Understanding human genome regulation through entropic graph-based regularization and submodular optimization

Maxwell W. Libbrecht

A dissertation
submitted in partial fulfillment of the
requirements for the degree of

Doctor of Philosophy

University of Washington

2016

Reading Committee:

William Stafford Noble, Chair

Jeffrey A Bilmes

Philip Green

Program Authorized to Offer Degree:
Computer Science and Engineering

University of Washington

Abstract

Understanding human genome regulation through entropic graph-based regularization and submodular optimization

Maxwell W. Libbrecht

Chair of the Supervisory Committee:
Professor William Stafford Noble
Genome Sciences

I am interested in developing computational methods to improve understanding of human genome regulation. This thesis is organized around two novel machine learning methods. First, I present a new method in the field of *posterior regularization*. The genomic neighborhood of a gene influences its activity, a behavior that is attributable in part to domain-scale regulation, in which regions of hundreds or thousands of kilobases known as domains are regulated as a unit. I developed a method called entropic graph-based posterior regularization that makes it possible to jointly model all available genomic data sets, including chromatin state information from ChIP-seq and chromatin conformation information from 3C-based assays. Using this approach, I produced a comprehensive model of chromatin domains in eight human cell types, thereby revealing the relationships among known domain types and identifying a new category of domain which I term "specific expression domains." Second, I present new methods from the field of *submodular optimization*. Due to the high cost of sequencing-based genomics assays such as ChIP-seq and DNase-seq, a panel of at most 3-10 assays is usually performed on each cell type. I present submodular selection of assays (SSA), a method for choosing a diverse panel of genomic assays that leverages methods from the field of submodular optimization. SSA performs better than alternative strategies in practice, is computationally efficient and extremely flexible, and is theoretically optimal under certain assumptions. This application may also serve as a model for how submodular optimization can be

applied to other discrete problems in biology. I applied a similar technique to remove redundancy in protein sequence data sets. This method applies submodular optimization to choose representative sets of protein sequences, achieving better results both in terms of reduction in redundancy and functional diversity. These methods are widely applicable to computational problems in genome biology and provide opportunities for the further development of methods.

TABLE OF CONTENTS

LIST OF FIGURES

Figure Number Page

LIST OF TABLES

Chapter 1

INTRODUCTION

More than a decade after the sequencing of the human genome, much is still unknown about how the genome functions. A primary function of the genome is the production of RNA transcripts of genes, which go on to produce the proteins that carry out most cellular activity. Gene regulation occurs through a number of biochemical processes. In the cell, most DNA is wrapped around histone proteins, forming a DNA-protein complex called chromatin. DNA-binding proteins (transcription factors) act on chromatin, changing its biochemical activity by recruiting other factors, covalently altering histone proteins, moving and removing histones to reveal the underlying DNA, and ultimately expressing or silencing a given gene. This complicated process is a key determinant of human traits, evolution and disease. I am interested in developing computational methods to improve our understanding of gene regulation.

1.1 Genomics assays

A class of experimental methods known as *genomics assays* are a powerful tool for understanding gene regulation. These experimental assays leverage DNA sequencing technology to assay various types of biochemical properties. These assays have become possible recently due to the dramatic increase in efficiency of high-throughput DNA sequencing technology. Due to this technological improvement, in the last decade, genomics assays have revolutionized the study of genome biology.

A representative genomics assay is *chromatin immunoprecipitation followed by sequencing* (ChIP-seq). This assay aims to identify the binding sites of a given transcription factor (i.e. DNA-binding protein)—call that factor X—in the genome of a population of human cells. These cells can be human cells grown in culture (cell lines), a sample of human primary tissue or any other source of cells in a homogenous state. In a ChIP-seq experiment, the researcher applies a series

of biochemical reactions to break up the genome into small fragments of DNA, then applies an antibody to extract the fragments that are bound to X (the term "immunoprecipitation" derives from this antibody step). The researcher then uses DNA sequencing technology to read the DNA sequence of each fragment. This sequence is known as a "read". If that sequence occurs exactly once in the human reference genome, the read is taken as evidence that the transcription factor binds to the matching position. A typical ChIP-seq experiment costs on the order of $500 and measures 10^7–10^8 such reads, so each binding event is measured hundreds of times to give statistical confidence. In that way, a ChIP-seq assay measures genome-wide binding of a given transcription factor in a population of cells relatively cheaply.

Many different types of genomics assays have been developed to query a wide variety of types of genome activity, including transcription factor binding, DNA accessibility, transcription, 3D genome architecture of the nucleus, and others. These assays generally proceed in similar ways as ChIP-seq, but replace the antibody step with a different step that extracts DNA fragments that exhibit some other activity of interest. In the last decade, several efforts have aimed to systematically apply genomics assays to a variety of human cells. The two largest such efforts are the ENCODE and Roadmap Epigenomics consortia. These two consortia have performed thousands of genomics assays on hundreds of human cell types. These rich data sets provide the opportunity to greatly improve our understanding of human genome biology. However, their scale and complexity necessitate the development of computational methods for their analysis.

1.2 Semi-automated genome annotation

A class of computational methods called *semi-automated genome annotation* (SAGA) algorithms are widely used to analyze genomics data sets. These algorithms take as input a collection of genomics data sets performed in a given cell type represented as real-valued tracks over the genome. They simultaneously partition the genome and label each segment with an integer label such that positions with the same label have similar patterns of activity. Each label is taken to represent a given type of genome element, and a researcher interprets what biological phenomenon each integer label corresponds to (hence the name "semi-automated"). These methods therefore both (1) identify

new types of genomic elements and (2) identify the function of each position in the genome. They have been shown [Ernst and Kellis, 2010, Hoffman et al., 2012] to re-discover many known types of elements, including genes, promoters, enhancers and heterochromatic (inactive) regions. I apply and extend SAGA methods throughout the work presented in this thesis.

1.3 This thesis

This thesis begins with a review of how machine learning is currently applied to genetic research (Chapter 2). The subsequent chapters are organized around novel machine learning methods in two fields, *posterior regularization* and *submodular optimization*. Chapter 3 presents a machine learning method called *entropic graph-based posterior regularization*. This method integrates prior work on unsupervised models such as SAGA with methods for semi-supervised learning, allowing new types of information to be efficiently incorporated into structured probabilistic models. In Chapter 4, I apply this method to investigate genomic *domain regulation*, in which regions of up to millions of base pairs are regulated as a unit. Chapters 5 and 6 present new methods in computational biology that apply techniques from *submodular optimization*. In Chapter 5, I present *Submodular Selection of Assays* (SSA), a method for choosing which genomics assays to perform on a new cell type. In Chapter 6, I present a submodular optimization-based method for removing redundancy in protein sequence data sets. In Chapter 7, I suggest several directions for future work in this field.

Chapter 2

MACHINE LEARNING IN GENETICS AND GENOMICS

Abstract

The field of machine learning promises to enable computers to assist humans in making sense of large, complex data sets. In this review, we outline some of the main applications of machine learning to genetic and genomic data. In the process, we identify some recurrent challenges associated with this type of analysis and provide general guidelines to assist in the practical application of machine learning to real genetic and genomic data.

Summary

- The field of machine learning is concerned with the development and application of computer algorithms that improve with experience.

- Machine learning methods can be divided into supervised, semisupervised and unsupervised methods. Supervised methods are trained on examples with labels (e.g. “gene” or “not gene”) and then are used to predict these labels on other examples, whereas unsupervised methods find patterns in data sets without the use of labels, and semisupervised methods leverage patterns in unlabeled data to improve power at predicting labels.

- An application requires different machine learning methods depending on whether one is interested in interpreting the output model or one is simply concerned with predictive power. Generative models, which posit a probabilistic distribution over input data, are generally best for interpretability, whereas discriminative models, which seek only to model labels, are generally best for predictive power.

- Prior information can be added to a model in order to train the model more effectively given limited data, limit the complexity of the model, or incorporate data not used by the model directly. Prior information can be incorporated explicitly in a probabilistic model or implicitly through the choice of features or similarity measures.

- Choosing an appropriate performance measure depends strongly upon the application task. Machine learning methods are most effective when they optimize an appropriate performance measure.

- Network estimation methods are appropriate when the data contains complex dependencies among examples. These methods work best when they take into account the confounding effects of indirect relationships.

2.1 Introduction

The field of machine learning is concerned with the development and application of computer algorithms that improve with experience [Mitchell, 1997]. Thus, for example, in genomics machine learning can be used to "learn" how to recognize the locations of transcription start sites (TSSs) in a genome sequence [Ohler et al., 2002]. The process typically proceeds in three stages (Figure 2.1). First, a machine learning researcher develops an algorithm that they believe will lead to successful learning. Second, the algorithm is provided with a large collection of TSS sequences as well as, optionally, a list of sequences that are known not to be TSSs. The annotation indicating whether a sequence is a TSS or not is known as the *label*. The algorithm processes these labeled sequences and stores a model. Third, novel, unlabeled sequences are given to the algorithm, and it uses the model to predict labels ("TSS" or "not TSS") for each sequence. If the learning was successful, then all or most of the predicted labels will be correct. If the labels associated with test set examples are known—i.e., if these examples were held out of the training set on purpose for use in testing the performance of the learning system—then the performance of the machine learning algorithm can be assessed immediately. Otherwise, in a prospective validation setting, the TSS predictions

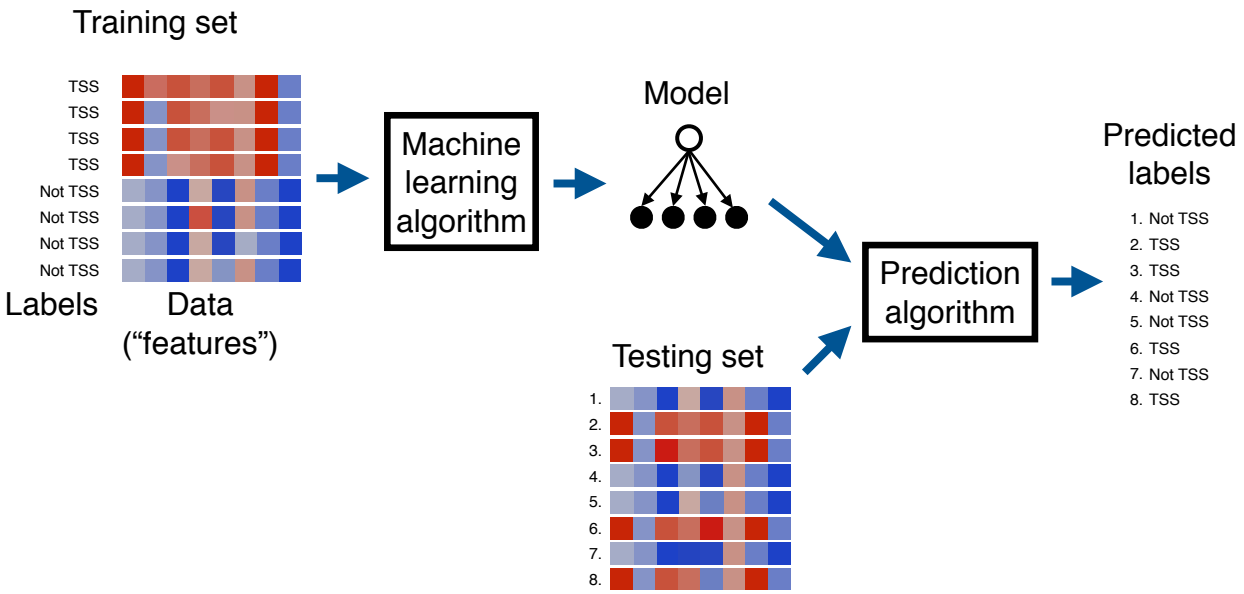


Figure 2.1: **Machine learning.** A canonical example of a machine learning application. A training set of DNA sequences is provided as input to a learning procedure, along with binary labels indicating whether each sequence is centered on a TSS or not. The learning algorithm produces a model which can then be subsequently used, in conjunction with a prediction algorithm, to assign predicted labels to unlabeled test sequences.

Input data	Task
DNA sequence	Identify transcription start sites, splice sites, exons, etc.
DNA sequence	Identify TF binding sites
DNA sequence	Identify genes
Gene expression	Predict regulatory relationships
Gene expression data	Identify biomarkers for a disease
Histone and TF ChIP-seq data	Partition and label the genome with chromatin state annotation
DNA sequence + gene expression + ...	Predict gene function
DNA sequence + histone mods + ...	Predict gene expression
DNA sequence + histone mods + ...	Predict variant deleteriousness
Sequence variants + gene expression + ...	Predict disease phenotype or prognosis

Table 2.1: **Selected applications of machine learning in genetics and genomics.**

produced by the machine learning system must be tested independently in the lab. Note that this is an example of a subtype of machine learning called "supervised learning," which is described in more detail below (Section 2.2).

This process of algorithm design, learning, and testing is simultaneously analogous to the scientific method on two different levels. First, the design-learn-test provides a principled way to test a hypothesis about machine learning: "I hypothesize that algorithm X can successfully learn to recognize TSSs." Second, the algorithm itself can be used as a hypothesis generator: "I hypothesize that sequence Y is a TSS." In this second setting, the resulting scientific theory is instantiated in the model produced by the learning algorithm. In this case, a key question, which we return to in Section 2.3, is whether and how easily a human can make sense of this model.

2.1.1 Machine learning application areas

Machine learning methods have been applied to a huge variety of problems in genomics and genetics (Table 2.1). Perhaps most significantly, machine learning has been used to annotate a wide variety of genomic sequence elements. In addition to TSSs, algorithms can be trained to identify splice sites [Degroeve et al., 2002], promoters [Bucher, 1990], enhancers [Heintzman et al., 2007], positioned nucleosomes [Segal et al., 2006], etc. In general, if you can compile a list of sequence elements of a given type, then you can probably train a machine learning method to recognize those

elements. Furthermore, models that each recognize an individual type of of genomic elements can be combined, along with (learned) logic about their relative locations, to build machine learning systems capable of annotating genes, including their full UTR/intron/exon structure, along entire eukaryotic chromosomes [Picardi and Pesole, 2010].

In addition to learning to recognize patterns in DNA sequences, machine learning can take as input data generated by other genomic assays, such as microarray or RNA-seq expression data, chromatin accessibility assays such as DNase-seq, MNase-seq, and FAIRE, or histone modification, transcription factor (TF) binding ChIP-seq data, etc. Gene expression data can be used to learn to distinguish between different disease phenotypes and, in the process, to identify potentially valuable disease biomarkers (Section 2.6). Chromatin data can be used, for example, to annotate the genome in an "unsupervised" fashion, thereby potentially allowing for the identification of novel classes of functional elements (Section 2.2).

Machine learning has also been used extensively to assign functional annotations to genes. Such annotations most frequently take the form of Gene Ontology term assignments [Gene Ontology Consortium, 2000b]. Predictive algorithms can take as input any one or more of a wide variety of data types, including the genomic sequence, gene expression profiles across various experimental conditions or phenotypes, protein-protein interactions, synthetic lethality data, open chromatin data, histone modification or TF binding ChIP-seq data, etc. As an alternative to GO term prediction, some predictors instead identify co-functional relationships; i.e., the machine learning method outputs a network in which genes are nodes and an edge between genes A and B indicates that the two genes share a common function [Fraser and Marcotte, 2004].

Finally, a wide variety of machine learning methods have been developed to help understand the mechanisms underlying gene expression. Some techniques aim to predict the expression of a gene based solely on the DNA sequence [Beer and Tavazoie, 2004], whereas others take into account ChIP-seq histone modification [Karlic et al., 2010] or TF binding [Ouyang et al., 2009] profiles at the gene promoter region. More sophisticated methods attempt to jointly model the expression of all genes in a cell by training a network model [Friedman, 2004]. Like a co-functional network, each node in a gene expression network is a gene; however, edges in this case represent, for example,

regulatory relationships between TFs and their targets.

Many of the problems listed above can also be solved using techniques drawn from the field of statistics. Indeed, the line between machine learning and statistics is at best blurry, with some preferring the term "statistical learning" over "machine learning" [Hastie et al., 2001]. Historically, the field of machine learning grew out of the artificial intelligence community, where the term "machine learning" became popular in the late 90s. In general, machine learning researchers have tended to focus on a subset of problems within statistics, emphasizing in particular the analysis of large, heterogeneous data sets. Accordingly, many core statistical concepts, such as calibration of likelihood estimates, statistical confidence estimation and power calculations, are essentially absent from the machine learning literature.

2.1.2 Scope of this review

This review provides an overview of machine learning as it is applied to problems in genomics. We discuss the main categories of machine learning methods and the key considerations that must be made when applying these methods to genomics problems. We do not not attempt to catalogue all machine learning methods or all reported applications of machine learning to genomics, nor do we discuss any particular method in great detail. Instead, the review begins by explaining several key distinctions in machine learning and then outlining some of the major challenges researchers face in applying machine learning methods to practical problems in genomics. We hope that the reader will take away from this review an idea of what problems machine learning approaches might be applicable to and which types of methods are likely to be effective for these problems. For a more detailed treatment of machine learning applied to particular subfields of genetics and genomics, the reader may consult reviews of these specific topics [Hamelryck, 2009, Swan et al., 2013, Upstill-Goddard et al., 2013, Yip et al., 2013].

The review will cover the following topics. Section 1 describes the major classes of machine learning problems: supervised, unsupervised and semisupervised. Section 2 concerns the tradeoff between maximizing a model's performance and interpretability. Sections 3-5 describe strategies a researcher can use to guide a machine learning model, through prior knowledge, means of integrating

Example	One data instance used in a machine learning task.
Feature	A single measurement or descriptor of an example used in a machine learning task.
Label	The target of a prediction task. In classification, the label is discrete (e.g., "expressed" or "not expressed"); in regression, the label is real-valued (e.g., a gene expression value).
Supervised algorithm	Machine learning algorithm that is trained on labeled examples and used to predict the label of unlabeled examples.
Unsupervised algorithm	Machine learning algorithm that does not require labels, such as a clustering algorithm.
Input space	Set of features chosen as input to a machine learning method.
Kernel methods	A class of machine learning methods (e.g. SVM) that employ a type of similarity measure between feature vectors called a *kernel*.
Bayesian network	A representation of a probability distribution that specifies the structure of dependencies between variables as a network.
Overfitting	A common pitfall in machine learning analysis where a complex model is trained on too little data and becomes specific to the training data, resulting in poor performance on other data.
Curse of dimensionality	The observation that analysis can sometimes become more difficult as the number of features increases, particularly because overfitting becomes more likely.
Feature selection	The process of choosing a smaller set of features from a larger set, either before applying a machine learning method or as part of training.
Class skew	A case where the two classes in a supervised learning problem are present with differing frequencies. Usually necessitates careful choice of performance measure.

Table 2.2: **Glossary of machine learning terminology.**

heterogeneous data sets and feature selection. Sections 6-9 describe several important pitfalls and special cases of machine learning: imbalance between classes in supervised problems, missing data and networks of dependencies between examples. Finally, in Section 10, we describe the outlook for machine learning in genomics in coming years.

2.2 Supervised versus unsupervised learning

Machine learning methods can usefully be segregated into two primary categories: *supervised* versus *unsupervised* methods. To illustrate the difference between the two, consider again the gene

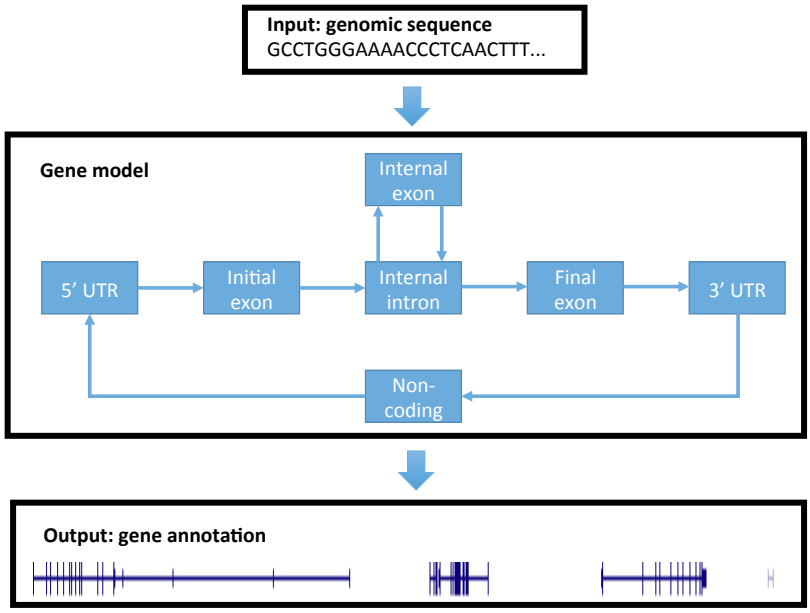


Figure 2.2: **A gene finding model.** A simplified gene finding model, which captures the basic properties of a protein-coding gene. The model takes as input the DNA sequence of a chromosome or a portion thereof and produces as output detailed gene annotations. Note that this simplified model is incapable of identifying overlapping genes or multiple isoforms of the same gene.

Method	Supervised	Unsupervised	Generative	Discriminative
Linear regression	✓		✓	
Artificial Neural Network (ANN)	✓	✓		✓
Support Vector Machine (SVM)	✓			✓
k-means clustering		✓	✓	
Hierarchical clustering		✓	✓	
Hidden Markov Model (HMM)	✓	✓	✓	

Table 2.3: **Selected machine learning methods.**

finding problem: given the DNA sequence of a chromosome, the goal of a gene finding algorithm is to predict the locations and detailed intron/exon structure of all of the protein-coding genes on the chromosome (Figure 2.2). The most straightforward solution to this problem leverages what we already know about the genome to help us build a predictive model. In particular, a supervised learning algorithm for gene finding requires as input a training set of *labeled* DNA sequences, where the labels specify the locations of the start and end of the gene (TSS and TTS) as well as all of the splice sites in between. The model then uses this training data to learn general properties of genes, such as what DNA sequence pattern typically occurs near a donor or acceptor splice site, the fact that in-frame stop codons should not occur within coding exons, the expected length distributions of 5' and 3' untranslated regions (UTRs) and of initial, internal and final introns, etc. The trained model can then use these learned properties to identify novel genes that resemble the genes in the training set.

On the other hand, supervised learning only makes sense when a labeled training set is available. Consider, for example, interpreting a heterogeneous collection of epigenomic data sets, such as those generated by the ENCODE and Epigenomics Roadmap consortia. A priori, we expect that the patterns of chromatin accessibility, histone modifications and TF binding along the genome should be able to provide us with a detailed picture of the biochemical and functional activity of the genome. We may also expect that these activities could be accurately summarized using a relatively small set of labels. If we are interested in discovering what types of labels best explain the data

rather than imposing a pre-determined set of labels on the data, then we must use unsupervised rather than supervised learning. In this type of approach, the machine learning algorithm takes as input only the unlabeled data and the desired number of different labels to assign [Day et al., 2007, Ernst and Kellis, 2012, Hoffman et al., 2012]. The algorithm then automatically partitions the genome into segments and assigns a label to each segment, with the goal of assigning the same label to segments that have similar data. Because the method is unsupervised, a human must subsequently assign semantics to each label, a step which is not required for a supervised gene finding algorithm. The benefit of the unsupervised approach, however, is the ability to train when labeled examples are unavailable and the ability to identify potentially novel types of genomic elements. The latter is fundamentally impossible with a supervised method, which can only identify more examples of the types of labels it learns from.

2.2.1 Semi-supervised learning

Intermediate between supervised and unsupervised learning is *semi-supervised learning* [Chapelle et al., 2006]. In supervised learning, the algorithm receives as input a collection of data points, each with an associated label, whereas in unsupervised learning, the algorithm receives the data but no labels. The semi-supervised setting is a mixture of these two: the algorithm receives a collection of data points, but only a subset of those points have associated labels. In practice, gene finding systems are often trained using a semi-supervised approach, where the input is a collection of annotated genes plus a complete (unlabeled) genome sequence. The learning procedure begins by constructing an initial gene finding model based solely on the labeled subset of the training data. Then the model is used to scan the genome, and tentative labels are assigned throughout the genome. These tentative labels can then be used to improve the learned model, and the procedure iterates until no new genes are found. The semi-supervised approach can work much better than a fully supervised approach because the model is able to learn from a much larger set of genes: all genes in the genome, rather than only the subset of genes that have been identified with high confidence.

2.2.2 Which type of method to use

When faced with a new machine learning task, the first question is often whether to use a supervised, unsupervised or semi-supervised approach. In some cases, the answer to this question is obvious. For example, if no labels are available, then you can only perform unsupervised learning. However, when labels are available, it is not always the case that taking a supervised approach is a good idea. This is because every supervised learning method rests upon the implicit assumption that the distribution responsible for generating the training data set is the same as the distribution responsible for generating the test data set. This assumption will be respected if, for example, you take a single labeled data set and randomly subdivide it into a training set and a testing set. However, it is often the case that you plan to train an algorithm on a training set that is generated differently from the testing data to which the trained model will eventually be applied. A gene finder trained using a training set of human genes will probably not work very well at finding genes in the mouse genome. Often, the train/test divergence is less obvious. For example, a TSS data set generated by cap analysis of gene expression (CAGE) data will not contain non-polyadenylated genes [Stamatoyannopoulos, 2010]. If such genes also exhibit differences around the TSS, then the resulting TSS predictor will be biased. In general, supervised learning should be employed only when the training set and test set are expected to exhibit similar statistical properties.

What about semi-supervised learning? When supervised learning is feasible, it is often the case that additional, unlabeled data points are easy to obtain. How do you decide whether to use a supervised or semi-supervised approach? In theory, semi-supervised learning requires making certain assumptions about the data set [Chapelle et al., 2006]). In practice, however, assessing these assumptions can be very difficult. Therefore, a good rule of thumb is to use semi-supervised learning if two conditions hold: you don't have very much labeled data and you have a very large amount of unlabeled data.

2.3 Generative versus discriminative modeling

Applications of machine learning methods generally have one of two goals: *prediction* or *interpretation*. Consider the problem of predicting, on the basis of a ChIP-seq experiment, the locations at which a given TF will bind genomic DNA. This task is analogous to the TSS prediction task (Figure 2.1), except that the labels are derived from ChIP-seq peaks. A researcher applying a machine learning method to this problem may either want to understand what properties of a sequence are most important for determining whether or not a TF will bind (interpretation) or simply predict the locations of TF binding as accurately as possible (prediction). There are tradeoffs between accomplishing these two goals—methods that optimize prediction accuracy often do so at the cost of interpretability.

The distinction between *generative* and *discriminative* models plays a large role in the tradeoff between interpretability and performance. Generative models build a full model of the distribution of features in each of the two classes and then compares how those two distributions differ from one another. In contrast, the discriminative approach focuses on accurately modeling just the boundary between the two classes. From a probabilistic perspective, the discriminative approach involves modeling just the conditional distribution of the label given the input feature data sets, as opposed to the joint distribution of the labels and features. Schematically, if we imagine that our task is to separate two groups of points in a two-dimensional space (Figure 2.3A), then the generative approach builds a full model of the distribution of points in each of the two classes and then compares how those two distributions differ from one another, while the discriminative approach focuses only on separating the two classes.

A researcher adopting a generative approach to the TF binding problem begins by imagining what procedure could be used to generate the observed data. A widely used, generative model of TF binding employs a position-specific frequency matrix (PSFM, Figure 2.3B), in which a collection of aligned binding sites of width w are summarized in a $4 \times w$ matrix M of frequencies, where the entry at position $M_{i,j}$ represents the empirical frequency of observing the ith DNA base at position j. We can generate a random bound sequence according to this PSFM model by drawing w random

numbers, each in the range $[0, 1\rangle$. For the jth random number we select the corresponding DNA base according to the frequencies in the jth column of M. Conversely, scoring a candidate binding site using the model corresponds to computing the product of the corresponding frequencies from the PSFM. This value is called the *likelihood*. Training a PSFM is simple—you simply compute the empirical frequency of each nucleotide at each position.

A simple example of a discriminative algorithm is the *support vector machine* (SVM, Figure 2.3C) [Boser et al., 1992, Noble, 2006]. The goal of the SVM is to learn to output a "1" whenever it is given a positive training example and a "-1" whenever it is given a negative training example. In the TF binding prediction problem, the input sequence of length w is encoded as a binary string of length $4w$, where each bit corresponds to the presence or absence of a particular nucleotide at a particular position.

This generative modeling approach offers several compelling benefits. First, the generative description of the data implies that the model parameters have well-defined semantics relative to the generative process. Accordingly, as shown in the example above, the model not only makes predictions for where a given TF binds but also provides a story about why the TF binds there. If we compare two different potential binding sites, we can see not only that the model prefers one site over another but also that the preference comes from, e.g., the preference for an A rather than a T at position 7 of the motif. Second, generative models are frequently stated in terms of probabilities, and the probabilistic framework provides a principled way to handle problems like missing data. For example, it is still possible for a PSFM to make a prediction for a binding site where one or more of the bound residues is unknown. This is accomplished by probabilistically averaging over the missing bases. The probabilistic framework is also helpful because the output has well-defined, probabilistic semantics. This can be helpful when making downstream decisions about how much to trust a given prediction.

In many cases, including the TF binding example, the training data set contains a mixture of positive and negative examples. In a generative setting, these two groups of examples are modeled separately, each with its own generative process. For example, for the PSFM model, the negative (or background) model is often a single set B of nucleotide frequencies, representing the overall mean

frequency of each nucleotide in the negative training examples. To generate a length-w sequence according to this model, we again generate w random numbers, but now each base is selected according to the frequencies in B. To use the foreground PSFM model together with the background model B, we compute a *likelihood ratio*, i.e., which is simply the ratio of the likelihoods computed with respect to the PSFM and with respect to B.

The primary benefit of the discriminative approach is that it provably achieves better performance with infinite training data [Ng and Jordan, 2002, Jordan, 1995]. In practice, analogous generative and discriminative approaches often converge to the same solution, and generative approaches can sometimes perform better with limited training data. However, when the amount of labeled training data is reasonably large, then the discriminative approach will tend to find a better solution, in the sense that it will predict the desired outcome more accurately when tested on previously unseen data (assuming, as usual, that the data are drawn from the same underlying distribution as the training data). To illustrate this phenomenon, we simulated data according to a simple PSFM model and trained a PSFM and SVM, varying the number of training examples. To train a model of width 19 nucleotides to discriminate between bound and non-bound sites with 90% accuracy requires eight training examples for a PSFM model and only four examples for an SVM model (Figure 2.3D). This improvement in performance is achieved because, by not bothering to accurately characterize the "easy" parts of the 2D space, the discriminative model does a better job at solving the discrimination task at hand. Thus, empirically, the discriminative approach will tend to give more accurate predictions

The flipside of this accuracy, however, is that by solving a single problem well, the discriminative approach fails to solve other problems at all. Specifically, because the internal parameters of a generatively trained model have well-defined semantics, we can use the model to ask a variety of related questions, e.g., not just "Does CTCF bind to this particular sequence?" but "Why does CTCF bind to this sequence more tightly than to some other sequence?" The discriminative model, in contrast, only allows us to answer the single question for which it was designed. Thus, choosing between a generative and discriminative model involves a trade-off between predictive accuracy and interpretability of the model.

While the distinction between generative and discriminative models plays a large role in determining the interpretability of a model, the model's complexity—that is, the number of parameters it has—can be just as imporant. Models of either type with a large number of parameters tend to be difficult to interpret, but can generally achieve higher accuracy than models with few parameters, given enough data. The complexity of a model can be limited either by choosing a simple model or by using a feature selection strategy to restrict the complexity of the learned model.

2.4 Incorporating prior knowledge

In many applications, the success or failure of a machine learning method depends upon the extent to which the method successfully encodes various types of prior knowledge about the problem at hand. Indeed, much of the "art" of practical machine learning involves understanding the details of a particular problem and then selecting an algorithmic approach that allows those details to be accurately encoded. There is provably no optimal machine learning algorithm that works best for all problems [Wolpert and Macready, 1997], so this selection of approach that matches the researcher's prior knowledge about the problem is crucial to the success of the analysis.

Often, the encoding of prior knowledge is implicit in the framing of the machine learning problem. Consider, for example, the prediction of nucleosome positioning from primary DNA sequence [Segal et al., 2006]. The "labels" for this prediction task can be derived, for example, from an MNase sequencing assay. In this case, after appropriate processing, each base is assigned an integer count of the number of nucleosome-sized fragments that cover the base. It might seem natural, therefore, to frame the problem as a regression, where the input is, say, a sequence of length 201 bp and the output is the predicted coverage at the center of the sequence. However, in practice we may be particularly interested in identifying nucleosome-free regions. Hence, rather than asking the algorithm to solve the problem of predicting exact coverage at each base, we might instead opt to predict whether each base occurs in a nucleosome-free region or not. Switching from regression to classification makes the problem easier, but more importantly, this switch also encodes the prior knowledge that regions of high MNase accessibility are of particular biological interest.

2.4.1 *Implicit prior knowledge*

In other cases, prior knowledge is encoded implicitly in the choice of the data that we provide as input to the machine learning algorithm. For example, Yip et al. [Yip et al., 2012] trained a collection of machine learning methods to distinguish among various types of genomic elements. Using chromatin data—DNaseI accessibility and ChIP-seq profiles of histone modifications and TF binding—one classifier distinguished between regulatory regions that are close to a gene versus regulatory regions that are far away from any gene. In this context, design of the *input space*, i.e., the features that are provided as input to the classifier, is of critical importance. In the Yip et al. study, features were computed by averaging over a 100 bp window. The choice of a 100 bp window likely reflects the prior knowledge that, at least for histone modification data, the data are arguably only meaningful at approximately the scale of a single nucleosome (147 bp) or larger. Moreover, replacing a single, averaged feature with 100 separate features may also be problematic (see Section 2.6). In contrast, for DNase accessibility data, it is plausible that averaging over 100 bp may remove some useful signal. Alternatively, prior knowledge may be implicitly encoded in the learning algorithm itself, where some types of solutions are preferred over others [Urbanowicz et al., 2012a]. Therefore, in general, the choice of input data sets, their representations, and any preprocessing must be guided by prior knowledge about data and application.

2.4.2 *Probabilistic priors*

In a probabilistic framework, some forms of prior knowledge can be represented explicitly by specifying a prior distribution over the data. A common prior distribution is the uniform, or "uninformative," prior which, despite the name, can be quite useful in some contexts. Consider, for example, a scenario in which we have gathered a collection of ten validated binding sites for a particular transcription factor (Figure 2.4), and we observe that the sequences cluster around a clear consensus motif. If we represent these sequences using a pure PSFM, then because our data set is quite small, a significant number of the entries in the PSFM will be zero. Consequently, the model will assign any sequence that contains one of these zero entries an overall probability of zero, even

if the sequence otherwise exactly matches the motif. This is counter-intuitive. The solution to this problem is to encode the prior knowledge that every possible DNA sequence has the potential to be bound by a given transcription factor. The result is that even sequences containing nucleotides that we have never observed in a given position can still be assigned a non-zero probability by the model.

In many probability models, much more sophisticated priors have been developed to capture more complex prior knowledge. A particularly successful example is the use of *Dirichlet mixture priors* in protein modeling [Brown et al., 1993]. Here, the idea is that if we are examining an alignment of protein sequences, and if we see many aligned leucines in a particular column, then because we know that leucine and valine are biochemically similar, we may want to assign a high probability to sequences that contain a valine in that same column, even if we have never seen a valine there in our training data. The name "Dirichlet mixture" refers to the fact that the prior distribution is represented as a *Dirichlet distribution*, and that the distribution is a mixture in which each component corresponds to a group of biochemically similar amino acids. Such priors lead to significantly improved performance in modeling evolutionarily related families of proteins [Brown et al., 1993] and in discovering protein motifs [Bailey and Elkan, 1995].

2.4.3 Prior information in non-probabilistic models

On the other hand, incorporation of prior knowledge into non-probabilistic methods can be more challenging. For example, discriminative classifiers like artificial neural networks (ANNs) or random forests do not provide any explicit mechanism for representing prior knowledge. The topology of a multi-layer ANN can represent information about dependencies between input features in the input space, but more general priors cannot be represented.

One class of discriminative methods does provide a more general mechanism for representing prior knowledge, when that knowledge can be encoded into a generalized notion of similarity. *Kernel methods* are algorithms that substitute in place of a simple similarity function (specifically, the cosine of the angle between two input vectors) a general class of mathematical functions called *kernels* [Schölkopf and Smola, 2002]. The flagship kernel method is the SVM classifier, which has been widely used in many fields including biological applications ranging from DNA and protein

sequence classification to mass spectrometry analysis [Noble, 2006]. Other methods that can employ kernels include support vector regression as well as classical algorithms like k-means clustering, principal components analysis, and hierarchical clustering. Prior knowledge can be provided to a kernel method by selecting or designing an appropriate kernel function. For example, a wide variety of kernels can be defined between pairs of DNA sequences, where the similarity can be based on shared k-mers, irrespective of their positions within the sequences [Leslie et al., 2002], on nucleotides occuring at particular positions [Rätsch and Sonnenburg, 2004], or on a mixture of the two [Zien et al., 2000]. A DNA sequence can even encode a simple model of molecular evolution, using algorithms similar to the Smith-Waterman alignment algorithm [Saigo et al., 2006]. Furthermore, the *Fisher kernel* provides a general framework for deriving a kernel function from any probability model [Jaakkola and Haussler, 1998]. In this way, formal probabilistic priors can be used in conjunction with any kernel method. Kernel methods have a rich literature, which is reviewed in more detail in [Shawe-Taylor and Cristianini, 2004].

2.5 Handling heterogeneous data

Another common challenge in learning from real biological data is that the data themselves are heterogeneous. For example, consider the problem of learning to assign Gene Ontology terms to genes. For a given term, such as "cytoskeleton dependent intracellular transport," a wide variety of types of data might be relevant, including the amino acid sequence of the gene's protein product, that protein's inferred evolutionary relationships to other proteins across a variety of species, the microarray or RNAseq expression profile of the gene across a variety of phenotypic or environmental conditions, the number and identity of neighboring proteins identified using yeast two-hybrid or tandem affinity purification tagging experiments, GFP-tagged microscopy images, etc (Figure 2.5). Such data sets are difficult to analyze jointly because of their heterogeneity: an expression profile is a fixed-length vector of real values; protein-protein interaction information is a binary network, and protein sequences are variable length strings drawn from a discrete alphabet. Many statistical and machine learning methods for classification assume that all of the data can be represented as fixed-length vectors of real numbers. Such methods cannot directly be applied to heterogeneous

data.

The most straightforward way to solve this problem is to transform each type of data into vector format prior to processing (Figure 2.5A). This was the approach taken, for example, by Peña-Castillo et al. in a critical assessment of methods for predicting gene function in mice [Peña-Castillo et al., 2008]. Participating research groups were provided with separate matrices, each representing a single type of data: gene expression, sequence patterns, protein interactions, phenotypes, conservation profiles, and disease associations. All matrices shared a common set of rows, one per gene, but the number and meanings of the columns differed from one matrix to the next. For example, gene expression data were represented directly, whereas protein sequence data were represented indirectly via annotations from repositories such as Pfam [Sonnhammer et al., 1997] and InterPro [Apweiler et al., 2001], and protein-protein interactions were represented as square matrices in which entries were shortest-path lengths between genes.

Alternatively, each type of data can be encoded using a kernel function (Figure 2.5B), with one kernel for each data type. Mathematically, using kernels is formally equivalent to transforming each data type into a fixed-length vector; however, in the kernel approach the vectors themselves are not always represented explicitly. Instead, the similarity between two data elements—amino acid sequences, nodes in a protein-protein interaction network, etc.—is encoded in the kernel function. The kernel framework allows more complex kernels to be defined from combinations of simple kernels. For example, a simple summation of all the kernels for a given pair of genes is itself a kernel function. Furthermore, the kernels themselves can encode prior knowledge by, e.g., allowing for statistical dependencies within a given data type but not between data types [Pavlidis et al., 2002]. Kernels also provide a general framework for automatically learning the relative importance of each type of data relative to a given classification task [Lanckriet et al., 2004].

Finally, probability models provide a very different method for handling heterogeneous data. Rather than forcing all the data into a vector representation or requiring that all data be represented using pairwise similarities, a probability model explicitly represents diverse data types in the model itself (Figure 2.5C). An early example of this type of approach assigned gene functional labels to yeast genes on the basis of gene expression profiles, physical associations from affinity purification,

two-hybrid and direct binding measurements, as well as genetic associations from synthetic lethality experiments [Troyanskaya et al., 2003]. The authors used a *Bayesian network*, which is a formal graphical language for representing the joint probability distribution over a set of random variables [Pearl, 1998]. Querying the network with respect to the variable representing a particular Gene Ontology label yields the probability that a given gene is assigned that label. In principle, the probabilistic framework allows a Bayesian network to represent any arbitrary data type and perform joint inference over heterogeneous data types using a single model.

In practice, such inference can be challenging because a joint probability model over heterogeneous data may contain a very large number of trainable parameters. Therefore, an alternative method for handling heterogeneous data in a probability model is to "chain" the data types together. This approach makes use of the general mechanism for handling prior knowledge by treating one type of data as "prior" to another. For example, as we have discussed, the problem of predicting where along the genome sequence a particular transcription factor will bind can be framed as a classification problem and solved using a PSFM. However, *in vivo*, the binding of a TF depends not only on the native affinity of the TF for a given DNA sequence, but also upon competitive binding by other molecules. Accordingly, measurements of chromatin accessibility, as provided by assays such as DNase-seq [Song and Crawford, 2010], can provide valuable information about the overall accessibility of a given genomic locus to TF binding. A joint probability model can take this accessibility into account [Wasson and Hartemink, 2009, Pique-Regi et al., 2011], but training such a model can be challenging. The alternative approach uses the DNaseI data to create a probabilistic prior and then applies this prior during the scanning of the PSFM [Cuellar-Partida et al., 2011].

2.6 Feature selection

In any application of machine learning methods, the researcher must decide which data to provide as input to the algorithm. As noted above, this choice provides a method for incorporating prior knowledge into the procedure, because the researcher can decide which features of the data are likely to be relevant and which are probably irrelevant. On the other hand, choosing relevant features can itself be a scientifically interesting question. Consider, for example, the problem of training a

multiclass classifier to distinguish, on the basis of gene expression measurements, among different types of cancers [Ramaswamy et al., 2001]. Such a classifier could be valuable in two ways. First, the classifier itself could help to establish accurate diagnoses in cases of atypical presentation or histopathology. Second, the model produced during the learning phase could perform *feature selection*, identifying subsets of genes whose expression patterns contribute specifically to different types of cancers. In general, feature selection can be carried out within any supervised learning algorithm, in which the algorithm is given a large set of features (or input variables) and then automatically decides to ignore some or all of the features, focusing on the subset of features most relevant to the task at hand.

In practice, it is important to distinguish among three distinct motivations for performing feature selection with respect to a given task. First, in some cases, we want to identify a very small set of features that yield the best possible classifier. For example, we may want to produce an inexpensive way to identify a disease phenotype on the basis of the measured expression levels of a handful of genes. Such a classifier, if it is accurate enough, might serve as the basis for an inexpensive clinical assay. Second, we may want to use the classifier to understand the underlying biology [Glaab et al., 2012, Tibshirani, 1996, Urbanowicz et al., 2012b]. In this case, we want the feature selection procedure to identify all and only the genes whose expression is actually relevant to the task at hand, in the hopes that the corresponding functional annotations or biological pathways might provide insight into the etiology of disease. Third, we often simply want to train the most accurate possible classifier [Tikhonov, 1943]. In this case, we hope that the feature selection enables the classifier to identify and eliminate noisy or redundant features. Researchers are often disappointed to find that feature selection cannot optimally perform more than one of these three tasks simultaneously.

Feature selection is especially important in the third case because analysis of high-dimensional data sets, including genomic, epigenomic, proteomic or metabolomic data, suffers from the *curse of dimensionality* [Keogh and Mueen, 2011]. The "curse" refers to the general observation that many types of analysis become more difficult as the number of input dimensions (i.e., data measurements) grows very large. For example, as the number of data features input to a machine learning classifier grows, it is increasingly likely that, by chance, one feature perfectly separates the training examples

Scenario	Example performance measure
All predictions are equally important.	Accuracy (fraction of all predictions that are correct)
Some number of follow-up tests are planned	Precision (fraction of positives that are correct) among a fixed number of predicted enhancers
Positive predictions will be published	Sensitivity (number of enhancers that are identified) among predictions with a pre-determined precision (e.g. 95%).

Table 2.4: **Measuring performance of a classifier.** Different applications for a machine learning classifier necessitate different performance measures.

into positive and negative classes. This phenomenon leads to good performance on the training data but poor generalization to data not used in training due to the model being *overfit* to the training data. Feature selection methods and dimensionality reduction techniques, such as principal components analysis or multidimensional scaling, aim to solve this problem by projecting the data from higher to lower dimensions.

2.7 Imbalanced class sizes

A common stumbling block in many applications of machine learning to genomics is the large imbalance (or *skew*) in the relative sizes of the groups being classified. For example, suppose you are trying to use a discriminative machine learning method to predict the locations of enhancers in the genome. Starting with a set of 641 known enhancers, you break the genome up into 1000 bp segments and assign each segment a label ("enhancer" or "not enhancer") based on whether or not it overlaps a known enhancer. This procedure produces 1,711 positive examples and roughly 3,000,000 negative examples—2,000 times as many negative examples as positive. Unfortunately, most software cannot handle 3,000,000 examples.

The most straightforward solution to this problem is to select a random, smaller subset of the data. However, in the case of enhancer prediction, selecting a random 50,000 examples results in 49,971 negative examples and just 28 positive examples. This many positive examples is far too small to train an accurate classifier. To demonstrate this problem, we simulated 93 noisy genomics

assays using a Gaussian model for enhancers and background positions based on data produced by the ENCODE Consortium [ENCODE Project Consortium, 2012]. We trained a logistic regression classifier to distinguish between enhancers and non-enhancers on the basis of these data. The overall *accuracy* of the predictions, i.e., the percentage of predictions that were correct, was 99.9%. This sounds reasonably good until you consider that a null classifier that simply predicts everything to be non-enhancer achieves nearly the same accuracy.

In this context, it is more appropriate to separately evaluate *sensitivity*, defined as the fraction of enhancers detected, and the *precision*, defined as the percentage of predicted enhancers that are truly enhancers. Our balanced classifier has a high precision but a very low sensitivity, detecting only 0.5% of enhancers. The behavior of the classifier can be improved by instead using all the enhancers to train and then picking a random set of 49,000 non-enhancer positions as negative training examples. However, balancing the classes in this way results in the classifier learning to reproduce this artificially balanced ratio. The resulting classifier achieves much higher sensitivity (81%) but very poor precision (40%). This classifier is thus not useful for finding enhancers that can be validated experimentally.

It is possible to trade off sensitivity and precision while retaining the training power of a balanced training set by placing weights on the training examples. In the enhancer prediction case, we use the balanced training set, but during training we weight each negative example 36 times as much as a positive example. Doing so results in an excellent sensitivity of 53% with a precision of 95%.

In general, which performance measure is most appropriate depends on the intended application of the classifier (Table 2.4). For problems such as identifying which tissue a given cell comes from, where it may be equally important to identify rare and abundant tissues, the overall number of correct predictions may be the most informative measure of performance. In others, such as enhancer detection, predictions in one class may be more important than the other. A wide variety of performance measures are used in practice, including the F1 measure, the receiver operating characteristic (ROC) or precision-recall (PR) curve [Manning and Schütze, 1999, Davis and Goadrich, 2006], and others [Cohen, 1968]. Machine learning classifiers perform best when they optimize for a realistic performance measure.

2.8 Handling missing data

Machine learning analysis can often be complicated by missing data values. Missing values can come from a variety of sources, such as defective cells in a gene expression microarray, unmappable genome positions in a functional genomics assay, or measurements that are unreliable because they saturate the detection limits of an instrument. Missing data values can be divided into two types: (1) values that are missing at random or for reasons unrelated to the task at hand, such as defective microarray cells, and (2) values whose absences provides information about the task at hand, such as saturated detectors. The presence or absence of values of the second type are usually best incorporated directly into the model.

The simplest way to deal with data that is missing at random is to impute the missing values [Luengo et al., 2012]. This can be done either with a very simple strategy, such as replacing all missing values with zero, or a more sophisticated strategy. For example, Troyanskaya et al. used the correlations between data values to impute missing microarray values [Troyanskaya et al., 2001]. For each target gene expression profile, the authors found the 10 most similar other expression profiles and replaced each missing value in the target profile with an average of the values in the similar profiles. Imputing data points this way can be used as a preprocessing step for any other analysis, but downstream analyses are blind to this information and cannot make use of either the fact that a data point is missing or the added uncertainty resulting from missing data values.

Another method for dealing with missing data is to include in the model informaton about the missingness of each data point. For example, Kircher et al. aim to predict the deleteriousness of mutations based on functional genomics data [Kircher et al., 2014]. The functional genomics data provided a feature vector associated with each mutation, but some of these features were missing. Therefore, for each feature, the authors added a Boolean feature indicating whether the corresponding feature value was present. The missing values themselves were then replaced with zeroes. A sufficiently sophisticated model will be able to learn the pattern that determines the relationship between the feature and presence or absence indicator. An advantage of this approach to handling missing data is that it is applicable whether or not the absence of a data point is

significant—if it is not, the model will learn to ignore the absence indicator.

Finally, probability models can explicitly model missing data by considering all the potential missing values. For example, this approach is used by Hoffman et al. to analyze functional genomics data, which contain missing values due to mappability issues from short read sequencing data [Hoffman et al., 2012]. The probability model annotates the genome by assigning a label to each position and modeling the probability of observing a certain value given the label. Missing data points are handled by summing over all possibilities for that random variable in the model. This approach, called *marginalization*, represents the case where a particular variable is unobserved. However, marginalization is only appropriate when data points are missing for reasons unrelated to the task at hand. When the presence or absence of a data point is likely to be correlated with the values themselves, then incorporating presence or absence explicitly into the model is more appropriate.

2.9 Modeling dependence among examples

So far we have focused on machine learning tasks that involve sets of independent instances of a common pattern. However, in some domains, individual entities, such as genes, are not independent, and the relationships among them are important. By inferring such relationships, it is possible to integrate many examples into a meaningful network. Such a network might represent physical interactions among proteins, regulatory interactions between genes, or symbiosis between microbes. Networks are useful both for understanding the biological relationships between entities and as input into a following analysis that makes use of these relationships.

The most straightforward way to infer the relationships among examples is to consider each pair independently. In this case, the problem of network learning reduces to a normal machine learning problem, defined on pairs of individuals rather than individual examples. Qiu et al. used an SVM classifier to predict whether a given pair of proteins physically interact based on data such as their sequences and cellular localization [Qiu and Noble, 2008].

A downside of any approach that considers each relationship independently is that such methods cannot take into account the confounding effects of indirect relationships (Figure 2.6). For example,

in the case of gene regulation, an independent model cannot infer whether a pair of genes directly regulates each other or they are both regulated by a third gene. Such spurious inferred relationships, called *transitive* relationships, can be removed by methods which infer the graph as a whole. For example, Friedman et al. inferred a Bayesian network on gene expression data that models which genes regulate each other [Friedman et al., 2000]. Such a network includes only direct effects, and models indirect correlations through multiple-hop paths in the network. Methods that infer a network as a whole are therefore more biologically interpretable, because they remove these indirect correlations. A large number of such methods have been described, as reviewed in [Bacardit and Llorà, 2013, Koski and Noble, 2012].

2.10 Future of machine learning in genomics

Machine learning is most effective in situations where one would like to make sense of large, complex data sets. Accordingly, machine learning methods are likely to become ever more important to genomics as more large sets become available, through projects such as the 1000 Genomes Project or the NIH's 4D Nucleome initiative. On the other hand, even in the presence of massive amounts of data, machine learning techniques cannot generally be applied in a completely arbitrary fashion. In practice, effective application of machine learning methods requires a certain amount of "art." In particular, theoretical and practical knowledge of both machine learning methodology and the particular application area are often necessary to achieve good performance. As new technologies for generating large genomic and proteomic data sets emerge, pushing beyond DNA sequencing to mass spectrometry, flow cytometry and high-resolution imaging methods, demand will increase not only for new machine learning methods but also for experts capable of applying and adapting them to big data sets. In this sense, both machine learning itself and machine learning researchers are likely to become increasingly important to genetics and genomics.

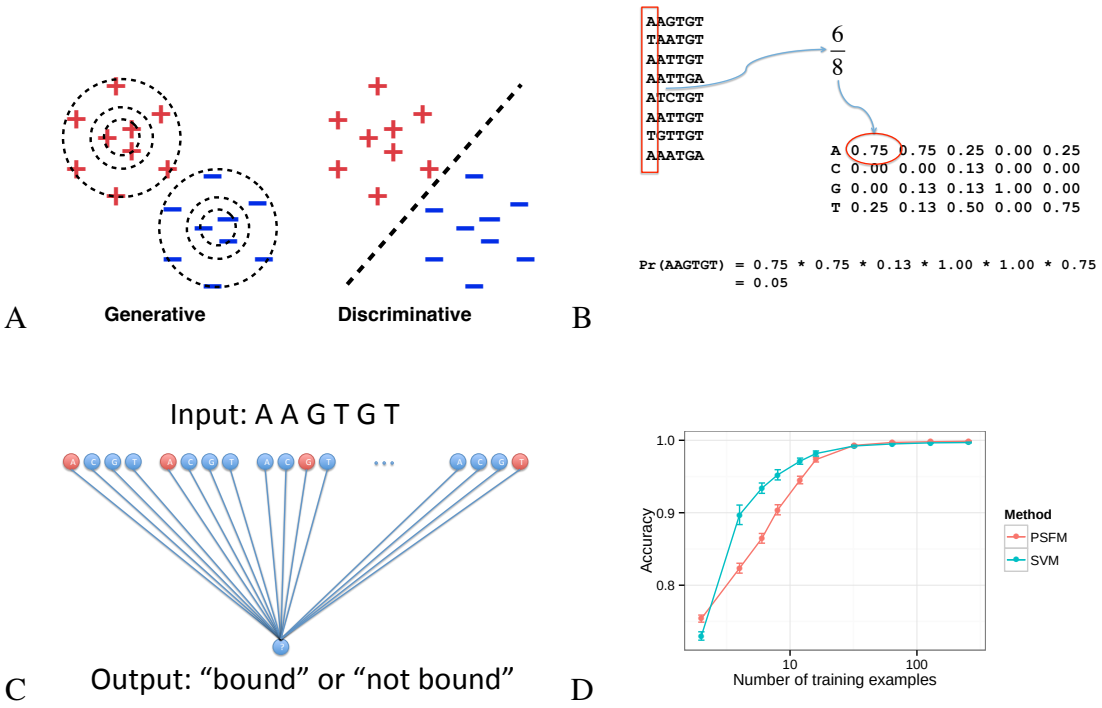


Figure 2.3: **Two models of TF binding.** (A) A position-specific frequency matrix model, in which the entry in row i and column j represents the frequency of the ith base occurring at position j in the training set. (B) A linear support vector machine model of TF binding. Labeled positive and negative training examples are provided as input, and a learning procedure adjusts the weights on the edges to predict the given label. (C) The figure plots the mean accuracy ($\pm$ 95% confidence intervals), on a set of 500 simulated test sets, of predicting TF binding as a function of the number of training examples. The two series correspond to a PSFM model or an SVM. (D) Schematic of generative versus discriminative modeling. The generative model characterizes both classes completely, whereas the discriminative model focuses on the boundary between the classes.

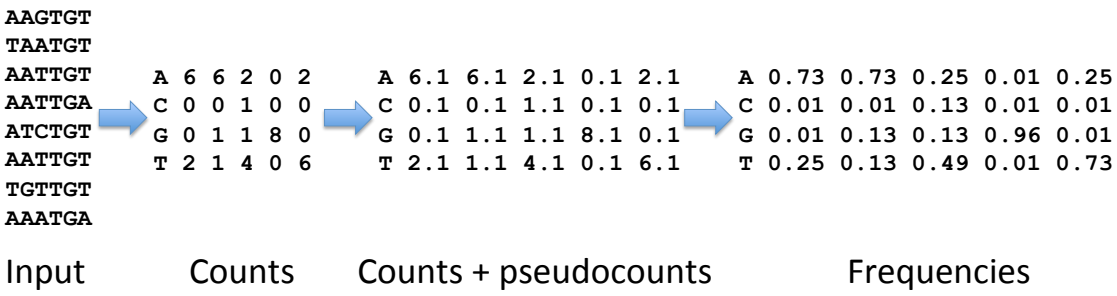


Figure 2.4: **Incorporating a prior into a PSFM.** A simple, principled method for putting a probabilistic prior on a PSFM involves augmenting the observed nucleotide counts with "pseudocounts" and then computing frequencies with respect to the sum. The magnitude of the pseudocount corresponds to the weight assigned to the prior.

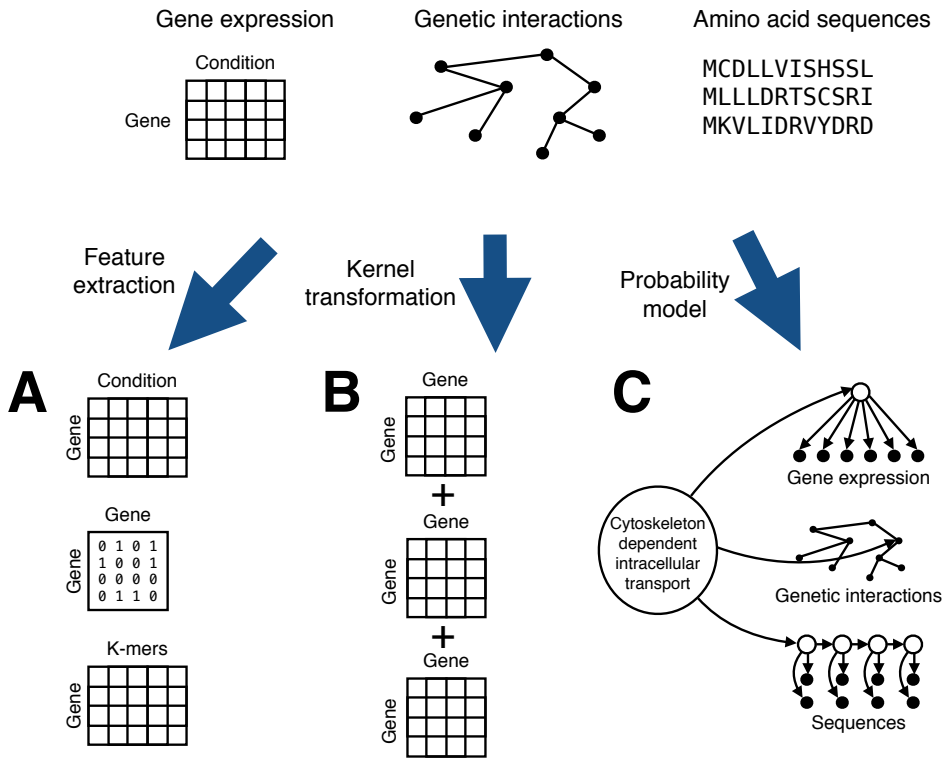


Figure 2.5: **Three ways to accommodate heterogeneous data in machine learning.** The task of predicting gene function labels requires methods that take as input gene expression data, protein sequences, protein-protein interaction networks, etc. These diverse data types can be encoded into fixed-length features, represented using pairwise similarities (kernels), or directly accommodated by a probability model.

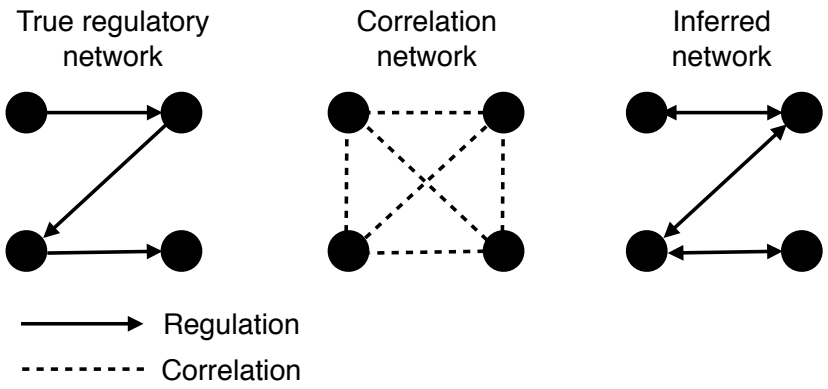


Figure 2.6: **Inferring network structure.** Methods that infer each relationship in a network separately, such as computing the correlation between each pair, can be confounded by indirect relationships. Methods that infer the network as a whole can identify only direct relationships. Inferring the direction of causality inherent in networks is generally more challenging than inferring the network structure [Pearl, 2000], so many network inference methods, such as Gaussian graphical model learning, infer only the network.

Chapter 3

ENTROPIC GRAPH-BASED POSTERIOR REGULARIZATION

Abstract

Graph smoothness objectives have achieved great success in semi-supervised learning but have not yet been applied extensively to unsupervised generative models. We define a new class of entropic graph-based posterior regularizers that augment a probabilistic model by encouraging pairs of nearby variables in a regularization graph to have similar posterior distributions. We present a three-way alternating optimization algorithm with closed-form updates for performing inference on this joint model and learning its parameters. This method admits updates linear in the degree of the regularization graph, exhibits monotone convergence, and is easily parallelizable. We are motivated by applications in computational biology in which temporal models such as hidden Markov models are used to learn a human-interpretable representation of genomic data. On a synthetic problem, we show that our method outperforms existing methods for graph-based regularization and a comparable strategy for incorporating long-range interactions using existing methods for approximate inference. Using genome-scale functional genomics data, we integrate genome 3D interaction data into existing models for genome annotation and demonstrate significant improvements in predicting genomic activity.

3.1 Introduction

Graph-based methods have recently been successful in solving many types of semi-supervised learning problems [Chapelle et al., 2006, Das and Smith, 2011, Joachims, 1999, Subramanya et al., 2010, Subramanya and Bilmes, 2009, 2011, Zhu et al., 2004, Zhu and Ghahramani, 2002]. These methods assume that data instances lie in a low-dimensional manifold that may be represented as a graph. They optimize a *graph smoothness* criterion, which states that data instances nearby in

the graph should be more likely to receive the same label. In a semi-supervised learning setting, optimizing this criterion has the effect of spreading labels from labeled to unlabeled instances.

Despite the success of graph-based methods for semi-supervised learning, there has not been as much study of the use of graph smoothness objectives in an unsupervised setting. In unsupervised problems, we do not have labels but instead have a generative model that is assumed to explain the observed data given the latent labels. While some types of relationships between instances (for example, the relationship between neighboring words in a sentence or neighboring bases in a genome) can easily be incorporated into the generative model, it is often inappropriate to encode a graph smoothness assumption into the model this way, for two reasons. First, in some cases, it is not clear what probabilistic process generated the labels with respect to the graph. Some objectives and distance measures that are successful for semi-supervised learning do not have probabilistic analogues. Second, large models must obey factorization properties (e.g., a tree or chain as in hidden Markov models) to facilitate the use of efficient dynamic programming algorithms such as belief propagation. Graphs representing similarity between variables do not in general satisfy these structure requirements because they tend to be densely clustered, leading to very high-order factors.

In this paper, therefore, we propose a new regularization approach for expressing a graph smoothness objective over a probabilistic model. We employ the posterior regularization (PR) framework of Ganchev et al. [2010], in which a probabilistic model is regularized through a term defined on an auxiliary posterior distribution variable. We define a powerful posterior regularizer which encourages pairs of variables to have similar posterior distributions by adding a penalty based on their Kullback-Leibler (KL) divergence. The pairs of penalized variables are encoded in a regularization graph which may be entirely different from the graphical model on which inference is performed. This regularizer graph need not have low treewidth and admits efficient optimization even when fully connected. We call our strategy of adding KL regularization penalties *entropic graph-based posterior regularization* (EGPR).

We show that inference and learning using this regularizer can be performed efficiently using a three-way alternating optimization algorithm with closed-form updates. This algorithm alternates between (1) smoothing marginal posteriors according to a regularization similarity graph, (2) per-

forming probabilistic inference in a graphical model with the same dependence structure as the unregularized model, and (3) updating model parameters. The updates are linear in the degree of the regularization graph and are easily parallelizable [Bilmes and Subramanya, 2012], in our experiments scaling to tens of millions of variables. We show that this procedure corresponds to a generalization of the EM algorithm.

We apply this approach to improve existing methods for annotating the human genome [Day et al., 2007, Hoffman et al., 2012, Ernst and Kellis, 2010]. Methods for genome annotation distill genomic data into a human-interpretable form by simultaneously partitioning the genome into non-overlapping segments and assigning labels to each segment. This type of analysis has recently had great success in interpreting the function of the human genome and formed an integral part of the analysis of the NIH-sponsored ENCODE project ([ENCODE Project Consortium, 2012, Hoffman et al., 2013], `http://www.nature.com/encode`).

However, exiting annotation methods use temporal models such as hidden Markov models and therefore cannot efficiently incorporate data on the genome's 3D structure. This 3D structure has been shown to play a key role in gene regulation and other genomic processes. In our experiments on synthetic data, a model using EGPR outperforms comparable models using either other regularization strategies (e.g., squared error) or loopy belief propagation. On ENCODE data, a model using EGPR predicts genome activity much more accurately than the currently-used chain models as well as other forms of regularizer. Thus EGPR provides a method for jointly modeling genome activity and 3D structure.

3.2 Proposed Method

In an unsupervised learning problem, we are given a set of vertices V that index a set of $n = |V|$ random variables $X_V = \{X_1, \ldots, X_n\}$ and a conditional dependence graph $G = (V, E)$. The graphical model describes a probability distribution parameterized by θ that can be factorized as $p_\theta(x_V) = \frac{1}{Z} \prod_{C \in \mathcal{C}} \phi_\theta^{(C)}(x_C)$ where each $C \subseteq V$ is a fully connected clique in G. We denote random variables with capital letters (e.g., X_H) and instantiations of variables with lower-case (e.g., $x_H \in \text{domain}(X_H)$). We also use capitals to denote sets and lowercase to denote set elements (e.g.,

X_h for $h \in H$).

Training graphical models involves a set of observed data $\bar{x}_O$, where a subset of variables $O \subseteq V$ is observed and the remainder $H = V \setminus O$ are hidden. In general we may have many samples of observed data, but for notational simplicity, in this work we assume the graphical model expresses the independence between samples, so it is sufficient to assume just one sample.

In this work we require that every hidden variable have the same domain $\mathcal{X}$, although our results can easily be extended to a case where we have several sets of variables, each with a common domain. When the probability distribution is governed by a set of parameters θ, penalized maximum likelihood training corresponds to the optimization

$$\text{maximize}_\theta \qquad J(\theta) \triangleq \mathcal{L}(\theta) + \mathcal{R}(\theta) \tag{3.1}$$

$$\text{where } \mathcal{L}(\theta) \triangleq \log p_\theta(\bar{x}_O) = \log \sum_{x_H} p_\theta(x_H, \bar{x}_O), \tag{3.2}$$

and where $\mathcal{R}(\theta)$ is a regularizer that expresses prior knowledge about the parameters. Many regularizers are used in practice, such as the ℓ_2 or ℓ_1 norms, which encourage parameters to be small or sparse, respectively.

Instead of placing a regularizer on the parameters themselves, it is often more natural to place a regularizer on the posterior distribution, a technique called *posterior regularization* [Ganchev et al., 2010]. This is done by introducing an auxiliary joint distribution $q(X_H)$, placing a regularizer on $q(X_H)$, and encouraging q to be similar to p_θ via a KL divergence penalty. The regularizer is

$$\mathcal{R}_{\text{PR}}(\theta) \triangleq \max_q \mathcal{R}'_{\text{PR}}(\theta, q) \tag{3.3}$$

$$\mathcal{R}'_{\text{PR}}(\theta, q) \triangleq -D(q(X_H) \| p_\theta(X_H | \bar{x}_O)) + \mathcal{PR}(q), \tag{3.4}$$

where $D(\cdot\|\cdot)$ is the KL divergence $D(p(X)\|q(X)) = \sum_x p(x) \log(p(x)/q(x))$ and $\mathcal{PR}(q)$ is a penalty term that expresses some prior knowledge about the posterior distribution. For notational convenience, we also define $J'(\theta, q) \triangleq \mathcal{L}(\theta) + \mathcal{R}'(\theta, q)$. Ganchev et al. [2010] showed how to optimize this combined objective efficiently when $\mathcal{PR}(q)$ is a sum of terms over individual cliques

in the model. Such regularizers can be used for constraining the posterior of individual variables in expectation, among other applications. However, graph smoothness objectives cannot be expressed this way, because they involve arbitrary pairs of variables.

When we have a graph smoothness assumption, we are given a weighted, undirected regularization graph over the hidden variables $G_R = (H, E_R)$, where $E_R \subseteq H \times H$ is a set of edges with non-negative similarity weights $w : E_R \to \mathbb{R}_+$, such that a large $w(u, v)$ indicates that we have strong belief that X_u and X_v should be similar. The regularization graph G_R is entirely separate from the conditional dependence graph G and, in particular, need not obey any decomposition or factorization properties to admit efficient inference.

He et al. [2013] introduced a regularizer of the following form. Let λ_G be a hyperparameter controlling the strength of regularization. The regularizer is

$$\mathcal{PR}_{\text{GPR}}(q) \triangleq -\lambda_G \sum_{(u,v)\in E_R} w(u, v)\|q(X_u) - q(X_v)\|_2^2.$$

He et al. showed how to optimize this regularizer using an exponentiated gradient descent method.

Although this regularizer shows good results for some problems, the use of squared error—or indeed any p-norm—to represent dissimilarity between probability distributions can be highly suboptimal, as we demonstrate empirically in Sections 3.6 and 4.2.9. Squared error is based on a Gaussian error model, which is not appropriate for probability values, and it under-penalizes differences between small probability values. p-norms are defined over all real numbers, while posteriors must lie within the range $[0, 1]$ (and live in a simplex).

A more justified way to measure divergence between probability distributions is to employ the KL divergence. The KL divergence measures the difference of exponents in the probability and so evaluates differences between small and large probabilities more uniformly. Also, Pinsker's inequality [Csiszár and Tusnády, 1984] combined with the relationship of ℓ-norms implies that $D(p\|q) \geq \frac{1}{2}\|p - q\|_1^2 \geq \frac{1}{2}\|p - q\|_\ell^2$, for all $\ell \geq 1$, where $\|\cdot\|_\ell$ is the ℓ-norm. Hence, minimizing KL divergence minimizes an upper bound on all ℓ-norms.

As a concrete example, consider two pairs of probability distributions over two events: $p_1 =$

$[0.55, 0.45]$ vs. $q_1 = [0.45, 0.55]$, and the second pair $p_2 = [0.1, 0.9]$ vs. $q_2 = [10^{-10}, 1 - 10^{-10}]$. The first pair of distributions (p_1, q_1) are fairly similar, with both events being roughly equally likely. The second pair (p_2, q_2) is quite dissimilar, with the first event being reasonably likely in the first case p_2 and astronomically unlikely in the second q_2. Squared error actually regards the first pair as more dissimilar than the second pair, while KL divergence identifies the second pair as much more dissimilar. Despite the advantages of KL divergence, although all posterior regularization objectives include a KL term binding q to be similar to p_θ, to our knowledge, no existing methods define the posterior regularizer **itself** using KL. Results in Sections 3.6 and 4.2.9 will show, moreover, that KL significantly improves over squared error across the board.

In this work, we propose such a posterior regularizer, which we term *entropic graph-based posterior regularization* (EGPR). The posterior regularizer is

$$\begin{aligned} &\mathcal{PR}_{\text{EGPR}}(q) \triangleq \\ &\quad -\lambda_G \sum_{(u,v)\in E_R} w(u,v) D(q(X_u) \| q(X_v)), \end{aligned} \tag{3.5}$$

and $J'_{\text{EGPR}}(\theta, q)$ and $\mathcal{R}'_{\text{EGPR}}(\theta, q)$ are defined according to Equations (4.3) and (4.5) respectively using the corresponding regularizers. That is,

$$\begin{aligned} &\text{maximize}_{\theta,q} \quad J'_{\text{EGPR}}(\theta, q) \triangleq \mathcal{L}(\theta) + \mathcal{R}'_{\text{EGPR}}(\theta, q), \\ &\mathcal{R}'_{\text{EGPR}}(\theta, q) \triangleq -D(q(X_H) \| p_\theta(X_H | \bar{x}_O)) + \mathcal{PR}_{\text{EGPR}}(q). \end{aligned}$$

The KL divergence in Equation 4.6 is symmetrized because G_R is undirected—that is $w(u, v) = w(v, u)$—so $D(q_u \| q_v)$ and $D(q_v \| q_u)$ appear with the same weight in the regularizer.

In the next section, we describe a novel alternating optimization algorithm for solving Equation (4.2) with an EGPR regularizer. Unlike other recent prominent examples of alternating optimization in machine learning [Wang et al., 2008], each update of this method has a closed-form solution. EGPR can be employed either as a regularizer for training the parameters or for inference directly. In the training case, an EM-like algorithm described in Section 3.3.3 is used to compute

and output θ, which can then be used for inference either with or without EGPR. In the inference case, q is computed and output as the posterior marginals, as described in Section 3.4.

3.3 EGPR for Training

We first describe how to compute $\operatorname{argmax}_q J'_{\text{EGPR}}(\theta, q)$, then describe how this algorithm can be used in combination with an EM-like algorithm for learning θ.

3.3.1 Optimizing q

The EGPR regularizer $\mathcal{R}'_{\text{EGPR}}(\theta, q)$ is convex in q; therefore, we could compute q using any convex optimization algorithm. However, general-purpose convex optimization algorithms do not scale to problems with millions or billions of variables such as those present in genomics. Therefore, we instead propose a novel alternating optimization strategy for performing this optimization more efficiently.

To enable closed form updates for this objective, we reformulate $J'_{\text{EGPR}}(\theta, q)$ by introducing a new variable $r^M(X_H)$. Like q, r^M is a distribution over X_H, but we require that r^M be factorizable as a product of marginals—that is $r^M(x_H) = \prod_h r_h^M(x_h)$. (In this manuscript, we use the notation $p^M(X_H)$ to indicate that p is a product of marginals.) We define the graph regularizer over r^M and add an additional term $\lambda_{\text{R1}} D(q(X_H) \| r^M(X_H))$, which encourages q and r^M to be similar. As we show below, restricting r^M in this way means that the reformulated objective is a lower bound on the original rather than being equivalent. We maximize this lower bound as an approximation to maximizing the original. The reformulated regularizer is

$$\begin{aligned} \mathcal{PR}'_{\text{EGPR-R1}}(q, r^M) \triangleq & -\lambda_{\text{R1}} D(q(X_H) \| r^M(X_H)) \\ & + f_{\text{R1}}(r^M) \end{aligned} \tag{3.6}$$

and

$$f_{\text{R1}}(r^M) \triangleq \\ -\lambda_G \sum_{(u,v)\in F_{EGPR}} w(u,v) D(r^M(X_u) \| r^M(X_v)), \tag{3.7}$$

where $J'_{\text{EGPR-R1}}(\theta, q, r^M)$ and $\mathcal{R}'_{\text{EGPR-R1}}(\theta, q, r^M)$ are defined according to Equations (4.3) and (4.5) respectively using the corresponding regularizers. That is,

$$\begin{aligned} &\text{maximize}_{\theta,q,r^M} \quad J'_{\text{EGPR-R1}}(\theta, q, r^M) \triangleq \\ &\qquad \mathcal{L}(\theta) + \mathcal{R}'_{\text{EGPR-R1}}(\theta, q, r^M), \end{aligned} \tag{3.8}$$

$$\begin{aligned} &\mathcal{R}'_{\text{EGPR-R1}}(\theta, q, r^M) \triangleq \\ &-D(q(X_H) \| p_\theta(X_H | \bar{x}_O)) + \mathcal{PR}_{\text{EGPR-R1}}(q, r^M). \end{aligned} \tag{3.9}$$

First, we show that $r^M \approx q$ for large values of λ_{R1}, so optimizing the reformulated regularizer is equivalent to optimizing a lower bound on the original.

Lemma 3.3.1. *For distributions $p \in \mathcal{P}$ and $q \in \mathcal{Q}$ where $\mathcal{P} \cap \mathcal{Q} \neq \emptyset$ and a continuous function $J(p, q)$, let $\tilde{J}(p, q; \lambda) = J(p, q) - \lambda D(p \| q)$, and $p^*_\lambda, q^*_\lambda \in \text{argmax}_{p\in\mathcal{P}, q\in\mathcal{Q}} \tilde{J}(p, q; \lambda)$. Then the following hold:*

$$\lim_{\lambda\to\infty} D(p^*_\lambda \| q^*_\lambda) = 0, \tag{3.10}$$

$$\lim_{\lambda\to\infty} \|p^*_\lambda - q^*_\lambda\|_\ell = 0 \tag{3.11}$$

for any ℓ, where $\|\cdot\|_\ell$ is the ℓ-norm, and

$$\lim_{\lambda\to\infty} \max_{p\in\mathcal{P}, q\in\mathcal{Q}} \tilde{J}(p, q; \lambda) \leq \max_{p\in\mathcal{P}} J(p, p). \tag{3.12}$$

Proof. Consider any $\epsilon > 0$ and any $p' \in \mathcal{P}, q' \in \mathcal{Q}$ such that $D(p' \| q') > \epsilon$. Let $\hat{p} \in \text{argmax}_{p\in\mathcal{P}\cap\mathcal{Q}} J(p, p)$

and consider any $\lambda' \geq (1/\epsilon)(J(p'\|q') - J(\hat{p}\|\hat{p}))$.

$$\begin{aligned}
\tilde{J}(p', q'; \lambda') &= J(p', q') - \lambda D(p'\|q') && (3.13)\\
&< J(p', q') - \lambda\epsilon && (3.14)\\
&\leq J(p', q') - \cancel{\epsilon}\cancel{(1/\epsilon)}(J(p'\|q') - J(\hat{p}\|\hat{p})) && (3.15)\\
&= J(\hat{p}, \hat{p}) && (3.16)
\end{aligned}$$

Therefore, $D(p^*\|q^*) \leq \epsilon$ when $\lambda \geq \lambda'$. This proves Proposition (3.10).

We have that

$$D(p\|q) \geq \frac{1}{2}\|p - q\|_1^2 \geq \frac{1}{2}\|p - q\|_\ell^2 \tag{3.17}$$

$$\tag{3.18}$$

for any ℓ-norm. The first inequality is Pinsker's inequality and the second follows from the relationship of ℓ-norms. Proposition (3.11) follows from this combined with Proposition (3.10).

Due to Proposition (3.11) and the continuity of $J(p, q)$,

$$\lim_{\lambda\to\infty} \max_{p\in\mathcal{P}, q\in\mathcal{Q}} \tilde{J}(p, q; \lambda) - \max_{p\in\mathcal{P}\cap\mathcal{Q}} J(p, p) = 0. \tag{3.19}$$

Proposition (3.12) follows from this and the fact that $\mathcal{P} \cap \mathcal{Q} \subseteq \mathcal{P}$. □

Therefore, for sufficiently large λ_{R1}, optimizing Equation (3.7) is equivalent to optimizing a lower bound on Equation (4.6). This form allows us to compute q efficiently, which is shown as follows.

Theorem 3.3.2. *Define* $q^*(X_H) \triangleq \operatorname{argmax}_q J'_{\text{EGPR-R1}}(\theta, q, r^M)$. *Then,*

$$q^*(x_H) = \frac{p_\theta(x_H, \bar{x}_O)^{1/(1+\lambda_{\text{R1}})} \prod_{h\in H} r_h^M(x_h)^{\lambda_{\text{R1}}/(1+\lambda_{\text{R1}})}}{\sum_{x'_H} p_\theta(x'_H, \bar{x}_O)^{1/(1+\lambda_{\text{R1}})} \prod_{h\in H} r_h^M(x'_h)^{\lambda_{\text{R1}}/(1+\lambda_{\text{R1}})}}. \tag{3.20}$$

Proof. For ease of notation, we group all terms that do not depend on q into one function $K_{3.3.2}(r)$. Since we must respect the sum-to-one property of q, we form the Lagrangian by adding the term $\lambda_{3.3.2}(1 - \sum_{x_H} q(x_H))$

$$L_{3.3.2}(q, \lambda_{3.3.2}) = -D(q(X_H) \| p_\theta(X_H|X_O)) - \lambda_{\text{R1}} D(q(X_H) \| r^M(X_H)) - \lambda_{3.3.2}(1 - \sum_{x_H} q(x_H)) + K_{3.3.2}(r) \tag{3.21}$$

$$= \sum_{x_H} q(x_H) \log \frac{p_\theta(x_H|\bar{x}_O) r^M(x_H)^{\lambda_{\text{R1}}}}{q(x_H)^{1+\lambda_{\text{R1}}}} - \lambda_{3.3.2}(1 - \sum_{x_H} q(x_H)) + K_{3.3.2}(r) \tag{3.22}$$

$$= \sum_{x_H} q(x_H) \log \frac{p_\theta(x_H, \bar{x}_O) r^M(x_H)^{\lambda_{\text{R1}}}}{q(x_H)^{1+\lambda_{\text{R1}}}} - \log p_\theta(\bar{x}_O) - \lambda_{3.3.2}(1 - \sum_{x_H} q(x_H)) + K_{3.3.2}(r) \tag{3.23}$$

$$0 = \frac{\partial L}{\partial q(x_H)} = -\log p_\theta(x_H, \bar{x}_O) r^M(x_H)^{\lambda_{\text{R1}}} + \log q(x_H)^{1+\lambda_{\text{R1}}} + 1 + \lambda_{\text{R1}} - \lambda_{3.3.2} \tag{3.24}$$

$$\implies q(x_H) \propto p_\theta(x_H, \bar{x}_O)^{\frac{1}{1+\lambda_{\text{R1}}}} r^M(x_H)^{\frac{\lambda_{\text{R1}}}{1+\lambda_{\text{R1}}}} \tag{3.25}$$

□

This is identical to the original model p_θ, but with one additional factor $r_h^M(x_h)^{\lambda_{\text{R1}}/(1+\lambda_{\text{R1}})}$ over each label. Critically, because r^M is factorizable such that each factor involves just one variable X_h, $q^*(X_H)$ is factorizable in the same way as the unregularized model $p_\theta(X_H, \bar{x}_O)$. For example, if the original model was an HMM, q still factors as a chain. Therefore, the normalization constant can be computed using any algorithm for exact or approximate probabilistic inference on factorized models, such as belief propagation, with similar computational cost as the unregularized model.

3.3.2 Optimizing r^M

While the last reformulation enabled closed form updates for q, the objective still does not admit closed-form updates for r^M. This is due to the fact that an objective of the form $D(p\|q) + D(p\|r)$

admits closed form updates for p, while $D(p\|q)+D(r\|p)$ does not. Therefore, we again reformulate $\mathcal{PR}_{\text{EGPR-R1}}(\theta, q, r^M)$ by adding a new variable s^M, where s^M is also a distribution over X_H restricted to be factorizable as a product of marginals. As before, we add a term $\lambda_{\text{R2}} D(s^M(X_H)\|r^M(X_H))$, which encourages $s^M \approx r^M$. We define the graph regularizer KL divergence terms to have s^M on the left and r^M on the right—that is, in the form $D(s_u^M(X_u)\|r_v^M(X_v))$—which will enable efficient optimization for both variables.

$$\begin{aligned}\mathcal{PR}'_{\text{EGPR-R2}}(q, r^M, s^M) &\triangleq -\lambda_{\text{R1}} D(q(X_H)\|r^M(X_H)) \\ &\quad + \max_{s^M} f_{\text{R2}}(r^M, s^M) \\ f_{\text{R2}}(r^M, s^M) &\triangleq -\lambda_{\text{R2}} D(s^M(X_H)\|r^M(X_H)) \\ &\quad - \lambda_G \sum_{(u,v)\in F_{EGPR}} w(u,v) D(s_u^M(X_u)\|r_v^M(X_v)).\end{aligned} \tag{3.26}$$

$J'_{\text{EGPR-R2}}(\theta, q, r^M, s^M)$ and $\mathcal{R}'_{\text{EGPR-R2}}(\theta, q, r^M, s^M)$ are defined according to Equations (4.3) and (4.5) respectively using the corresponding regularizers. That is,

$$\begin{aligned}&\text{maximize}_{\theta,q,r^M,s^M} \quad J'_{\text{EGPR-R2}}(\theta, q, r^M, s^M) \triangleq \\ &\quad \mathcal{L}(\theta) + \mathcal{R}'_{\text{EGPR-R2}}(\theta, q, r^M, s^M),\end{aligned} \tag{3.27}$$

$$\begin{aligned}&\mathcal{R}'_{\text{EGPR-R1}}(\theta, q, r^M, s^M) \triangleq \\ &\quad - D(q(X_H)\|p_\theta(X_H|\bar{x}_O)) + \mathcal{PR}_{\text{EGPR-R2}}(q, r^M, s^M).\end{aligned} \tag{3.28}$$

By Lemma 3.3.1, optimizing $\mathcal{R}_{\text{EGPR-R2}}(q)$ is equivalent to optimizing $\mathcal{R}_{\text{EGPR-R1}}(q)$ for large values of λ_{R2}. This regularizer can be optimized in r^M and s^M using closed-form updates, shown as follows.

Theorem 3.3.3. *For notational simplicity, define a new regularization graph with self-edges of weight* $\lambda_{\text{R2}}/\lambda_G$, $E'_{\text{EGPR}} \triangleq E_{\text{R}} \cup \{(h,h) \mid h \in H\}$, *and* $w'(u,v) \triangleq w(u,v) + \delta(u=v)\lambda_{\text{R2}}/\lambda_G$. *Let* $r^{M*}(X_H) \in \text{argmax}_{r^M} J'_{\text{EGPR-R2}}(\theta, q, r^M, s^M)$ *and* $s^{M*}(X_H) \in \text{argmax}_{s^M} J'_{\text{EGPR-R2}}(\theta, q, r^M, s^M)$.

Then,

$$r_v^M(x_v) = \frac{\lambda_{\text{R1}} q_v^M(x_v) + \lambda_G \sum_{(u,v)\in E'_{\text{EGPR}}} w'(u,v) s_u^M(x_v)}{\lambda_{\text{R1}} + \lambda_G \sum_{(u,v)\in E'_{\text{EGPR}}} w'(u,v)}, \tag{3.29}$$

$$s_u^{M^*}(x_u) = \frac{\exp \frac{\sum_{(u,v)\in E'_{\text{EGPR}}} w'(u,v) \log r_v^M(x_u)}{\sum_{(u,v)\in E'_{\text{EGPR}}} w'(u,v)}}{\sum_{x'_u} \exp \frac{\sum_{(u,v)\in E'_{\text{EGPR}}} w'(u,v) \log r_v^M(x'_u)}{\sum_{(u,v)\in E'_{\text{EGPR}}} w'(u,v)}}. \tag{3.30}$$

Proof. In its current form, $\mathcal{PR}_{\text{EGPR-R2}}(q)$ involves a sum over all values of $q(X_H)$. However, the following lemma shows how the factorizability of r^M facilitates expressing the objective in a form that involves only sum over values of each variable X_h.

Lemma 3.3.4. *For distribution $p(X_V)$ and factorizable distribution $q^M(X_V) = \prod_{v\in V} q_v^M(X_v)$, define $p_v^M(X_v) \triangleq \sum_{X_{V\setminus v}} p(X_V)$.*

$$D(p\|q^M) = \sum_{v\in V} D(p(X_v)\|q(X_v)) - H(p) + \sum_{v\in V} H(q(X_v)). \tag{3.31}$$

Proof.

$$D(p\|q^M) = -H(p) - \sum_{x_V} p(x_V) \log \left(\prod_{v \in V} q_v^M(X_v) \right) \tag{3.32}$$

$$= -H(p) - \sum_{x_V} p(x_V) \sum_{v \in V} \log q_v^M(X_v) \tag{3.33}$$

$$= -H(p) - \sum_{v \in V} \sum_{x_v} \sum_{x_V \neq v} p(x_V) \log q_v^M(X_v) \tag{3.34}$$

$$= -H(p) - \sum_{v \in V} \sum_{x_v} \left(\log q_v^M(X_v) \right) \sum_{x_V \neq v} p(x_V) \tag{3.35}$$

$$= -H(p) - \sum_{v \in V} \sum_{x_v} \left(\log q_v^M(X_v) \right) p_v^M(x_v) \tag{3.36}$$

$$= \sum_{v \in V} D(p(X_v)\|q(X_v)) - H(p) + \sum_{v \in V} H(q(X_v)) \tag{3.37}$$

□

Define q_h^M to be the marginal distribution of q over X_h, $q_h^M(X_h) \triangleq \sum_{X_{H \setminus h}} q(X_H)$. Using Lemma 3.3.4,

$$\mathcal{PR}_{\text{EGPR-R2}}(q) = \max_{r^M} \left(-\lambda_{\text{R1}} \sum_{h \in H} D(q_h^M(X_h)\|r_h^M(X_h)) + H(q) - \sum_{h \in H} H(q_h^M(X_h)) + f_{\text{R2}}(r^M) \right). \tag{3.38}$$

We now proceed to derive the update steps. We first derive the update for r^M. The Lagrangian

for the optimization of r_v^M is

$$L_{3.3.3\text{–}1}(r_v^M, \lambda_{3.3.3\text{–}1}) \tag{3.39}$$

$$= \lambda_{\text{R1}} D(q_v^M(X_v) \| r_v^M(X_v)) + \lambda_G \sum_{(u,v)\in E'_{\text{EGPR}}} w'(u,v) D(s_u^M(X_u) \| r_v^M(X_u)) \tag{3.40}$$

$$+ \lambda_{3.3.3\text{–}1}(1 - \sum_{x_v} r_v^M(X_v)) + K_{3.3.3\text{–}1}(q, s^M, r^M_{H\setminus v}) \tag{3.41}$$

$$0 = \frac{\partial L}{\partial r_v^M(x_v)} = -\left(\lambda_{\text{R1}} q_v^M(x_v) + \lambda_G \sum_{(u,v)\in E'_{\text{EGPR}}} w'(u,v) s_u^M(x_v)\right) \frac{1}{r_v^M(x_v)} + \lambda_{3.3.3\text{–}1} \tag{3.42}$$

$$\implies r_v^M(x_v) \propto \lambda_{\text{R1}} q_v^M(x_v) + \lambda_G \sum_{(u,v)\in E'_{\text{EGPR}}} w'(u,v) s_u^M(x_v) \tag{3.43}$$

$$\sum_{x_v} \left(\lambda_{\text{R1}} q_v^M(x_v) + \lambda_G \sum_{(u,v)\in E'_{\text{EGPR}}} w'(u,v) s_u^M(x_v)\right) \tag{3.44}$$

$$= \lambda_{\text{R1}} \sum_{x_v} q_v^M(x_v) \overset{1}{\nearrow} + \lambda_G \sum_{(u,v)\in E'_{\text{EGPR}}} w'(u,v) \sum_{x_v} s_u^M(x_v) \overset{1}{\nearrow} \tag{3.45}$$

$$\implies r_v^M(x_v) = \frac{\lambda_{\text{R1}} q_v^M(x_v) + \lambda_G \sum_{(u,v)\in E'_{\text{EGPR}}} w'(u,v) s_u^M(x_v)}{\lambda_{\text{R1}} + \lambda_G \sum_{(u,v)\in E'_{\text{EGPR}}} w'(u,v)} \tag{3.46}$$

We next derive the update for s^M. The Lagrangian for the optimization of s_u^M is

$$L_{3.3.3\text{–}2}(s_u^M, \lambda_{3.3.3\text{–}2}) \tag{3.47}$$

$$= \lambda_G \sum_{(u,v)\in E'_{\text{EGPR}}} w'(u,v) D(s_u^M(X_u) \| r_v^M(X_u)) + \lambda_{3.3.3\text{–}2}(1 - \sum_{x_u} s_u^M(X_u)) + K_{3.3.3\text{–}2}(q, r^M, s_{H\setminus u}^M) \tag{3.48}$$

$$= \lambda_G \sum_{x_u} s_u^M(x_u) \sum_{(u,v)\in E'_{\text{EGPR}}} w'(u,v) \log \frac{s_u^M(x_u)}{r_v^M(x_u)} + \lambda_{3.3.3\text{–}2}(1 - \sum_{x_u} s_u^M(X_u)) + K_{3.3.3\text{–}2}(q, r^M, s_{H\setminus u}^M) \tag{3.49}$$

$$0 = \frac{\partial L}{\partial s_u^M(x_u)} = \lambda_G \sum_{(u,v)\in E'_{\text{EGPR}}} w'(u,v) \log \frac{1}{r_v^M(x_u)} + \left(\lambda_G \sum_{(u,v)\in E'_{\text{EGPR}}} w'(u,v) \right) (1 + \log s_u^M(x_u)) - \lambda_{3.3.3\text{–}2} \tag{3.50}$$

$$\implies s_u^M(x_u) \propto \exp \frac{\sum_{(u,v)\in E'_{\text{EGPR}}} w'(u,v) \log r_v^M(x_u)}{\sum_{(u,v)\in E'_{\text{EGPR}}} w'(u,v)} \tag{3.51}$$

□

3.3.3 Updating θ

The preceding section described an algorithm for computing $\operatorname{argmax}_q \mathcal{R}_{\text{EGPR-R2}}$. This algorithm can be combined with an EM-like algorithm in order to learn a θ that (locally) optimizes $J_{\text{EGPR-R2}}$, as we describe in this section. We use an alternating EM-like algorithm to compute θ.

E-step:

$$q^{(t+1)} \in \operatorname{argmax}_{q, r^M, s^M} J'_{\text{EGPR-R2}}(\theta^{(t)}, q, r^M, s^M)$$

M-step: $\theta^{(t+1)} \in \operatorname{argmax}_\theta J'_{\text{EGPR}}(\theta, q^{(t+1)})$

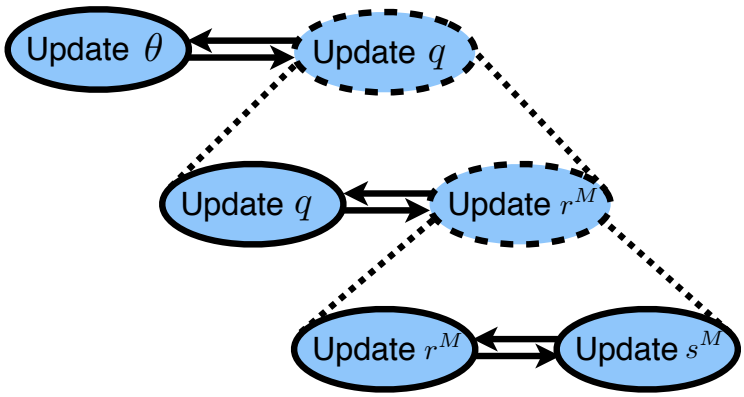


Figure 3.1: Illustration of optimization algorithm. Solid ovals denote closed-form update steps. Dashed ovals with dotted expansion lines denote updates that are implemented by alternating optimization. Pairs of opposing arrows indicate alternating optimization implemented by iterating each update to convergence.

The preceding section showed how to perform the E-step. To compute the M-step,

$$\begin{aligned} &\text{argmax}_\theta \, J'_{\text{EGPR}}(\theta, q^{(t+1)}) \\ &\quad = \text{argmax}_\theta \, E_{q^{(t+1)}(X_H)} \left[\log p_\theta(X_H, \bar{x}_O)\right] \end{aligned} \tag{3.52}$$

The M-step takes the same form as the EM algorithm presented in [Neal and Hinton, 1999]. The update for θ depends on the particular factorization and parameterization properties of the model. Because the posterior distribution $q(X_H)$ obeys the same factorization properties as the unregularized model $p_\theta(X_H, X_O)$, the same closed-form updates for θ can be used.

Therefore, the upper bound on the EGPR objective, $J_{\text{EGPR-R2}}$, can be minimized using a three-way alternating optimization algorithm, which proceeds by alternating closed-form updates to r^M and r^M to convergence, alternating this whole update of r^M/s^M with closed-form updates to q until convergence, then finally alternating updates to q and θ until convergence. A schematic of the algorithm is shown in Figure 3.1. The full algorithm in pseudocode is shown in Algorithm 1.

Theorem 3.3.5. *The modified EM algorithm monotonically increases the relaxed EGPR objective:*

$$J_{\text{EGPR-R2}}(\theta^{(t)}) \leq J_{\text{EGPR-R2}}(\theta^{(t+1)}). \tag{3.53}$$

Proof. Function $q^*(\cdot)$ of Algorithm 1 implements coordinate descent on q, r^M and s^M. $D(p\|q)$ and $D(p\|p)$ are jointly strictly convex in p and q and bounded below by 0. Thus, $J'_{\text{EGPR-R2}}$ is bounded below and jointly strictly convex in q, r^M and s^M. Convergence to the global optimum of $J'_{\text{EGPR-R2}}$ in q, r^M and s^M follows from its strict convexity [Warga, 1963].

$$\begin{aligned} J_{\text{EGPR-R2}}(\theta^{(t)}) &= J'_{\text{EGPR-R2}}(\theta^{(t)}, q^{(t+1)}, r^{M\,(t+1)}, s^{M\,(t+1)}) \\ &\leq J'_{\text{EGPR-R2}}(\theta^{(t+1)}, q^{(t+1)}, r^{M\,(t+1)}, s^{M\,(t+1)}) \\ &\leq J_{\text{EGPR-R2}}(\theta^{(t+1)}) \end{aligned}$$

The first equality follows from the global optimality of $q^{(t+1)}$, $r^{M\,(t+1)}$ and $s^{M\,(t+1)}$. The second inequality follows from the fact that $\theta^{(t+1)}$ is chosen to maximize $J'_{\text{EGPR-R2}}$. The third inequality follows from the fact that $J'_{\text{EGPR-R2}}(\theta, q, r^M, s^M)$ is a lower bound on $J_{\text{EGPR-R2}}(\theta)$. □

3.4 EGPR for Inference

In addition to its use for regularized learning, EGPR can be used directly as an inference algorithm. To do this, we compute $q^* \leftarrow \operatorname{argmax}_q J'_{\text{EGPR-R2}}(\theta, q)$ and use this distribution as the posterior. As discussed above, q^* obeys the same factorization properties as p_θ, so algorithms such as Viterbi and belief propagation can be used to compute the MAP solution and marginal distributions of q respectively. Using EGPR as an inference algorithm results in posteriors which are smooth with respect to the graph G_R.

3.5 Related work

The most straightforward way to express similarity information in an unsupervised model is to encode it in the graphical model. For example, one might add a factor between X_u and X_v as in

$\phi(x_u, x_v) = \lambda \mathbf{1}(x_u = x_v)$. This form of interaction is quite different from EGPR, because adding factors changes the "implementation" of the model, while EGPR regularizes what the model *does*. Moreover, there are two problems with this approach. First, it is not always clear what form of interaction is most appropriate. In particular, although KL-based penalties have been very successful in graph-based semi-supervised learning, they cannot be converted to an equivalent set of probability factors. Second, adding similarity edges to the probabilistic model results in a model that does not, in general, have low tree-width, so efficient exact inference algorithms such as belief propagation cannot be used, and one must resort to approximate inference. As we show in Section 4.7, EGPR performs better than the loopy belief propagation (LBP) approximate inference algorithm on an augmented graph.

Three methods take a similar approach to ours, by augmenting a probabilistic model with a graph regularizer. First, Altun et al. [2005] describe a graph regularization for max-margin models applied to pitch-accent prediction and optical character recognition. However, this method involves a matrix inversion step, and thus it cannot scale to large models. Second, Subramanya et al. [2010] combine a temporal conditional random field with a regularizer that expresses pairwise squared-error penalties derived from unlabeled data. They apply this method to the part-of-speech tagging task [Subramanya et al., 2010] and later to related problems in natural language [Das and Petrov, 2011, Das and Smith, 2011]. That work, however, resorts to a purely heuristic update step and lacks any optimality guarantees.

Third, He et al. [2013] present an approach based on an exponentiated gradient descent algorithm. Like our approach, He's approach exhibits monotone convergence. Although He's work has many similarities with our approach, He's work differs from ours in three important ways. First, He's method uses a squared-error penalty, which, as argued above, is less appropriate for probability distributions than Kullback-Leibler divergence [Bishop, 1995, p. 226]. As shown in Section 4.7, using squared error also results in worse performance in practice. Second, the exponentiated gradient descent method is applied to semi-supervised handwriting recognition and part-of-speech tagging, while we apply EGPR to an unsupervised genome annotation problem. Third, He et al. [He et al., 2013] use an exponentiated gradient descent strategy, while we use alternating optimization.

Our alternating optimization approach has several benefits over the exponentiated gradient descent method of He et al. [He et al., 2013]. First, the He et al. algorithm involves gradient calculations over each clique in the conditional dependence graph, with order $O(k^{|C|})$ for a variable of dimension k involved in cliques of size $|C|$. Our alternating minimization algorithm, by contrast, has closed-form updates of order $O(k)$ for q, r^M and s^M. (Updates for θ can still involve $O(k^{|C|})$ calculations, but these are generally very fast). Therefore, the alternating optimization algorithm is more appropriate for extremely large models or models with large cliques or high-order factors. Second, empirical studies of KL-based graph smoothness objectives have shown that alternating optimization algorithms perform better than gradient-based methods for these objectives [Subramanya and Bilmes, 2011, 2009]. Finally, the alternating optimization strategy is parallelizable [Bilmes and Subramanya, 2012] and extremely simple to implement because the graph regularization step has simple, closed-form updates with no learning rate hyperparameters, and the posterior calculation can be performed using any probabilistic inference method on a model with the same conditional dependence graph as the unregularized model.

3.6 Simulations

All implementations in the below use the GMTK, the graphical models toolkit [Bilmes, 2000], and its use of virtual evidence factors, for HMM and dynamic graphical model inference

3.6.1 Learning a Gaussian Mixture on Poorly-Separated Data

First, we consider a simple example that demonstrates the utility of EGPR (Figure 3.2). Suppose we wish to learn a mixture of four Gaussians on data lying in a circle in 2D, using EM training to find cluster centers. Clearly, the training problem is underspecified in this case, because any set of centers at 90 degrees from one another relative to the circle's center represents an optimum of the model likelihood. However, suppose we additionally have pairwise information that certain slices of the circle should form clusters. To represent this additional information, we form a regularization graph that connects all pairs of positions within the same cluster and run EM with EGPR using

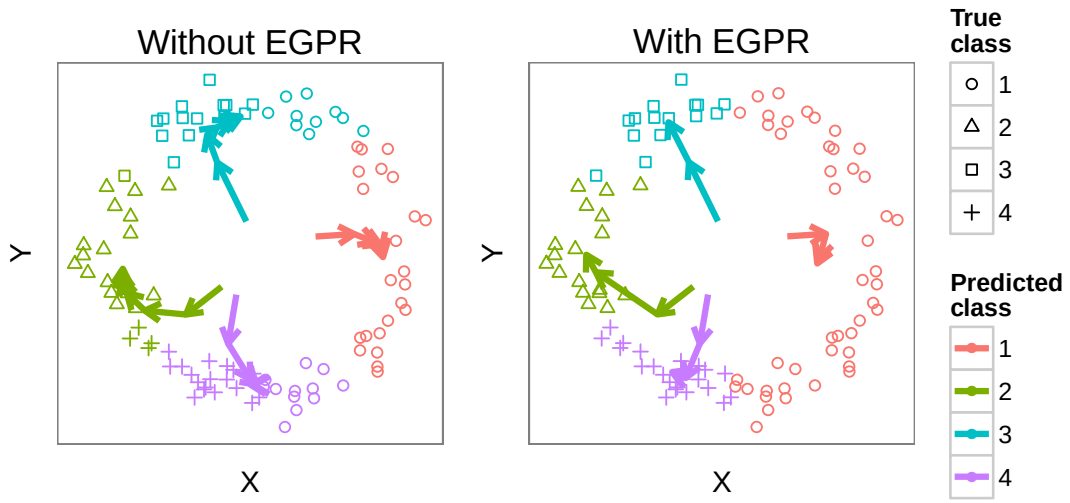


Figure 3.2: Using EGPR to learn ambiguous clusters. Shape denotes true class, color denotes predicted class, and colored arrows denote cluster means as they evolve between iterations of EM.

this graph. EGPR finds the true cluster centers and recovers the true labels with 100% accuracy compared to 74% accuracy with EM alone. This example demonstrates how pairwise information can be integrated in the training process to produce a trained model that implicitly incorporates this information.

3.6.2 Comparison with Related Inference Methods

To evaluate the efficacy of EGPR, we compared EGPR to two related methods: 1) approximate inference on a graphical model with the same dependence structure, and 2) GPR using squared-error penalties. We compared to the approximate inference method loopy belief propagation (LBP) because it is one of the most widely used approximate inference methods. While we would have preferred to perform this comparison using our genomics data sets (see Section 4.2.9), due to the large size of the models and dense connectivity of the regularization graphs, it appeared that even our implementations of these methods would take months to converge. Therefore, we instead performed this comparison using synthetic data. We generated a chain of length $n = 200$, with $(X_H, X_O) = (Z_{1:200}, Y_{1:200})$, where $Z_{1:200} \in \{0, 1\}^n$ and $Y_{1:200} \in \mathbb{R}^n$. We defined an HMM over this

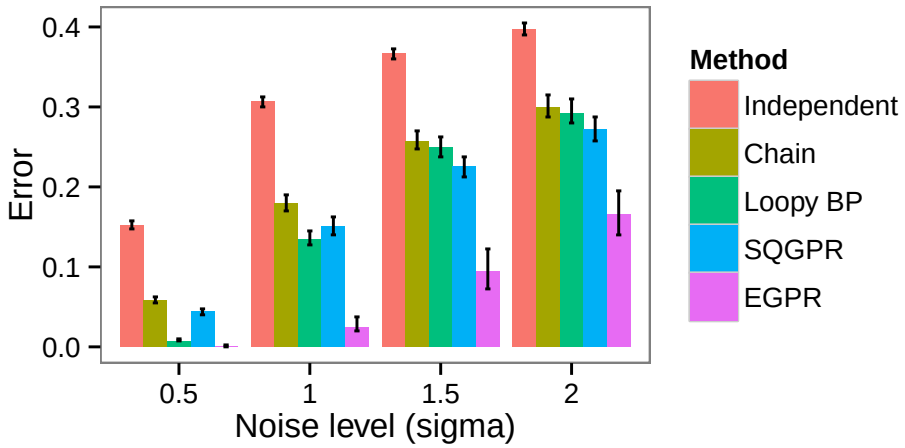


Figure 3.3: Comparison of EGPR with related inference methods. The X axis shows σ, a hyperparameter controlling the difficulty of inference. The Y axis shows the average accuracy over 200 simulations of MAP inference on the model in question (95% Wilcoxon test confidence intervals).

chain with transition probabilities $\Pr(Z_i = Z_{i+1}) = 0.9$ and emission probabilities $Y_i \sim N(Z_i, \sigma)$, where we vary σ to control the difficulty of the problem—higher σ results in more challenging inference. We generated a graph $W \in \mathbb{R}^{n \times n}$ over the vertices of the chain by setting $w_{ij} = 1$ with probability 0.4 if $Z_i = Z_j$, $w_{ij} = 1$ with probability 0.1 if $Z_i \neq Z_j$, and $w_{ij} = 0$ otherwise. This model is meant to simulate the task of labeling a chain (such as a genomic sequence) where we have noisy information about which pairs of positions have the same label.

We compared five methods of inference: 1) inference on each position independently, with no chain model; 2) inference on the chain alone, without using W; 3) LBP on the chain plus extra factors of $\Pr(X_i = X_j) = \text{sigmoid}(\lambda w_{ij})$, where λ controls the strength of these factors; 4) GPR using the regularization graph W and a squared-error penalties as described in [He et al., 2013] (SQGPR); and 5) EGPR using the regularization graph W. We chose hyperparameters for each model (λ_G, λ_{R1} and λ_{R2} for GPR and λ for LBP) using a training set of 200 simulations. We evaluated results according to the average accuracy over 200 simulations of MAP inference on the model in question.

EGPR significantly outperforms all other models for all experiments (Figure 3.3, providing nearly as much improvement in accuracy as does the chain model itself. The pattern of accuracy is

instructive in understanding the properties of each model. LBP performs very well when there is little noise, but becomes easily stuck in local optima on harder problems. GPR with squared error provides a modest improvement over the chain model, but has poor performance relative to KL penalties, consistent with previous work on semi-supervised methods [Subramanya and Bilmes, 2011, 2009].

3.7 Application: Genome Annotation Using Physical Interaction Information

Recently, many methods have been described that partition and label the human genome on the basis of a number of genome-wide real-valued signal tracks, generally employing temporal models such as HMMs [Day et al., 2007, Hoffman et al., 2012, Ernst and Kellis, 2010, Filion et al., 2010, Thurman et al., 2007, Lian et al., 2008]. Formally, these methods aim to learn a labeling $X_{H1:n} \in \{1..L\}^n$ which associates each position in the genome with one of L integer labels, such that positions that receive the same label exhibit similar patterns in the signal data. The input is comprised of a feature vector $X_{Oi} \in \mathbb{R}^F$ at each position that represents the output of biological experiments that measure local properties of the DNA, including its interaction with binding proteins, its local structure, and various types of chemical modifications. The process is "semi-automated" because a human assigns a semantic interpretation of the integer labels subsequent to the unsupervised learning phase.

However, existing genome annotation methods cannot incorporate the genome's 3D conformation. The 3D arrangement of the genome in the nucleus plays a central role in gene regulation, chromatin state and replication timing [Misteli, 2007, Dekker et al., 2002, Ryba et al., 2010, Dixon et al., 2012]. Genome conformation can be investigated using chromatin conformation capture experiments such as Hi-C [Lieberman-Aiden et al., 2009]. A Hi-C experiment outputs a matrix of contact counts, where the number of contact counts of a pair of genomic positions is inversely proportional to the positions' 3D distance in the nucleus [Lieberman-Aiden et al., 2009, Ay et al., 2014b]. Existing genome annotation methods can incorporate any data set that can be represented as a vector defined linearly across the genome, but they cannot incorporate inherently pairwise Hi-C data without resorting to simplifying transformations such as principle component analysis.

Algorithm 1 Efficient and scalable algorithm to optimize $J'_{\text{EGPR-R2}}$

function $r^{M*}(q)$
  for $h \in H$ **do**
    $q_h^M(x_h) \leftarrow \sum_{x_{H \neq h}} q(x_H)$ (belief propagation)
  end for
  Initialize $r^{M(0)}$, $s^{M(0)}$ arbitrarily.
  $t_1 \leftarrow 1$
  while not converged **do**
    for $v \in H$ **do**
      $$r_v^{M(t_1)}(x_v) \leftarrow \frac{\lambda_{\text{R1}} q_v^M(x_v) + \lambda_G \sum_{(u,v) \in E'_{\text{EGPR}}} w'(u,v) s_u^{M(t_1-1)}(x_v)}{\lambda_{\text{R1}} + \lambda_G \sum_{(u,v) \in E'_{\text{EGPR}}} w'(u,v)}$$
    end for
    for $u \in H$ **do**
      $$s_u^{M(t_1)}(x_u) \leftarrow \frac{\exp \frac{\sum_{(u,v) \in E'_{\text{EGPR}}} w'(u,v) \log r_v^{M(t_1-1)}(x_u)}{\sum_{(u,v) \in E'_{\text{EGPR}}} w'(u,v)}}{\sum_{x'_u} \exp \frac{\sum_{(u,v) \in E'_{\text{EGPR}}} w'(u,v) \log r_v^{M(t_1-1)}(x'_u)}{\sum_{(u,v) \in E'_{\text{EGPR}}} w'(u,v)}}$$
    end for
    $t_1 \leftarrow t_1 + 1$
  end while
  return $r^{M(t_1)}$
end function

function $q^*(\theta)$
  $t_2 \leftarrow 1$
  Initialize $r^{M(0)}$ arbitrarily.
  while not converged **do**
    $$q^{(t_2)}(x_H) \leftarrow \frac{p_\theta(x_H, \bar{x}_O)^{1/(1+\lambda_{\text{R1}})} \prod_{h \in H} r_h^{M(t_2-1)}(x_h)^{\lambda_{\text{R1}}/(1+\lambda_{\text{R1}})}}{\sum_{x'_H} p_\theta(x'_H, \bar{x}_O)^{1/(1+\lambda_{\text{R1}})} \prod_{h \in H} r_h^{M(t_2-1)}(x'_h)^{\lambda_{\text{R1}}/(1+\lambda_{\text{R1}})}}$$ (belief propagation)
    $r^{M(t_2)} \leftarrow r^{M*}(q^{(t_2)})$
    $t_2 \leftarrow t_2 + 1$
  end while
end function

Initialize $\theta^{(0)}$ arbitrarily.
$t_3 \leftarrow 1$
while not converged **do**
  $q^{(t_3)} \leftarrow q^*(\theta^{(t_3-1)})$
  $\theta^{(t_3)} \leftarrow \text{argmax}_\theta \, E_{q^{(t_3)}(X_H)} [\log p_\theta(X_H, \bar{x}_O))]$ (EM update)
end while
Output $\theta^{(t_3)}$

We therefore present a novel strategy for integrating 3D conformation information using EGPR in which we encourage pairs of positions which interact in 3D to receive the same label in the annotation by connecting these positions with edges in an EGPR graph (Figure 3.4). While the assumption that positions close in 3D space have similar regulatory state is not necessarily true at a small scale ($\sim$1 thousand base pair elements), it does generally hold at a large scale ($\sim$1 million base pair elements) [Lieberman-Aiden et al., 2009].

3.7.1 Synthetic Example

To motivate this approach, we first consider a simple synthetic model with 501 nodes and six EGPR edges. Let $Y_i \in \{0, 1\}$ for $i = 0 \ldots 500$. We assign $Y_1 = 0$ and $Y_{500} = 1$. Let $P(Y_{0:500}) = \prod_{i=0}^{499} 0.9^{\mathbf{1}(Y_i = Y_{i+1})} 0.1^{\mathbf{1}(Y_i \neq Y_{i+1})}$. In other words, the model places higher probability on neighboring positions taking the same label but provides no other information. We construct an EGPR graph with $w_{0,200} = w_{0,400} = w_{200,400} = w_{100,300} = w_{100,500} = w_{300,500} = 1$, corresponding to a hypothetical 2D arrangement of the chain. Without EGPR, the model learns a trivial labeling of the chain, whereas with EGPR, the model learns a labeling corresponding to the 2D arrangement (Figure 3.4).

3.7.2 Real data

Next, to evaluate the efficacy of our approach, we performed genome annotation of the human fibroblast cell line IMR90. We compared three models: (1) the chain model described in [Hoffman et al., 2012], without 3D structure data, (2) the chain model augmented with 3D structure data expressed with squared-error GPR (SQGPR), and (3) the chain model augmented with 3D structure data expressed with EGPR. We would have liked to compare against loopy belief propagation as well, but as mentioned previously, it appeared that our fastest implementation of this method would take months to converge on this large data set. In order to evaluate our performance in a variety of conditions, we ran each model separately once for each of the 29 available data sets in IMR90.

For SQGPR and EGPR, we used a GBR graph based on 3D structure data. To generate the GBR graph representation of 3D structure used by EGPR and SQGPR, we used the Hi-C data set

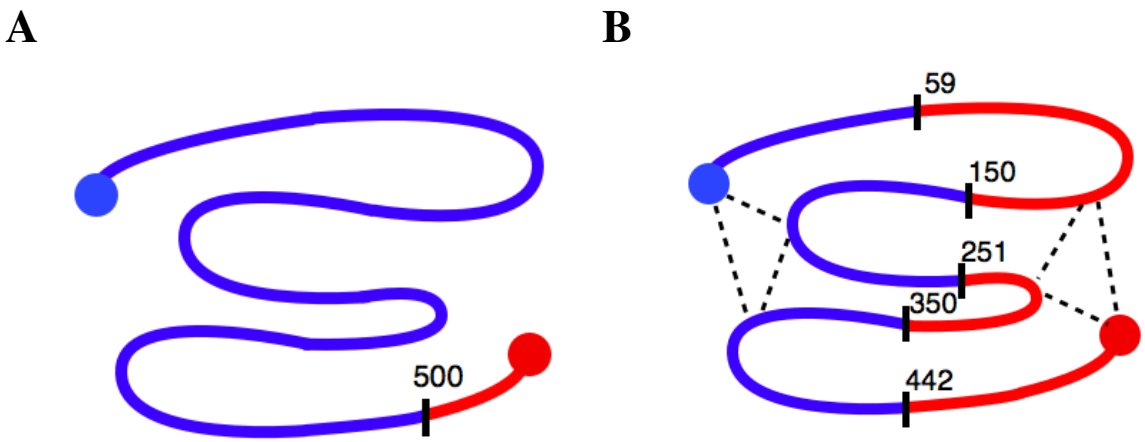


Figure 3.4: Synthetic model of genome spatial interactions. Color and labeled division lines indicate learned labels along the hypothesized 501 bp genome. Large filled circles indicate observed positions. Dotted lines indicate EGPR edges. (A) Without EGPR. (B) With EGPR.

of [Dixon et al., 2012] and processed the Hi-C data into a matrix of pairwise p-values using the Fit-Hi-C method [Ay et al., 2014a]. To remove noise and decrease the degree of the graph, we removed all contacts with uncorrected p-value $p > 10^{-6}$ and multiplied the remaining p-values by 10^6, similar to a Bonferroni correction. We generated the GPR graph by setting the weight between positions i and j, $w(i,j)$, to $w(i,j) \triangleq \max(0, -\log_e(p(i,j)/10^6))$, where $p(i,j)$ is the p-value of interaction between positions i and j. All annotations used four labels and binned the genome at 10,000 base pair resolution, comparable to the resolution of 40,000 bp used in [Dixon et al., 2012]. We chose the best-performing hyperparameters for each GPR model using a validation set.

To evaluate these annotations, we used the time during the cell cycle at which the DNA is replicated as a gold standard [Woodfine et al., 2004]. Replication time is highly correlated with gene expression and chromatin state and therefore is a good proxy for domain type [Lieberman-Aiden et al., 2009]. To evaluate the accuracy with which an annotation predicts replication time, we compute a prediction as follows. Let $a_i \in \{1 \ldots 4\}$ and x_i be the annotation label and replication time at position i, respectively. We compute the replication time mean over the positions assigned a

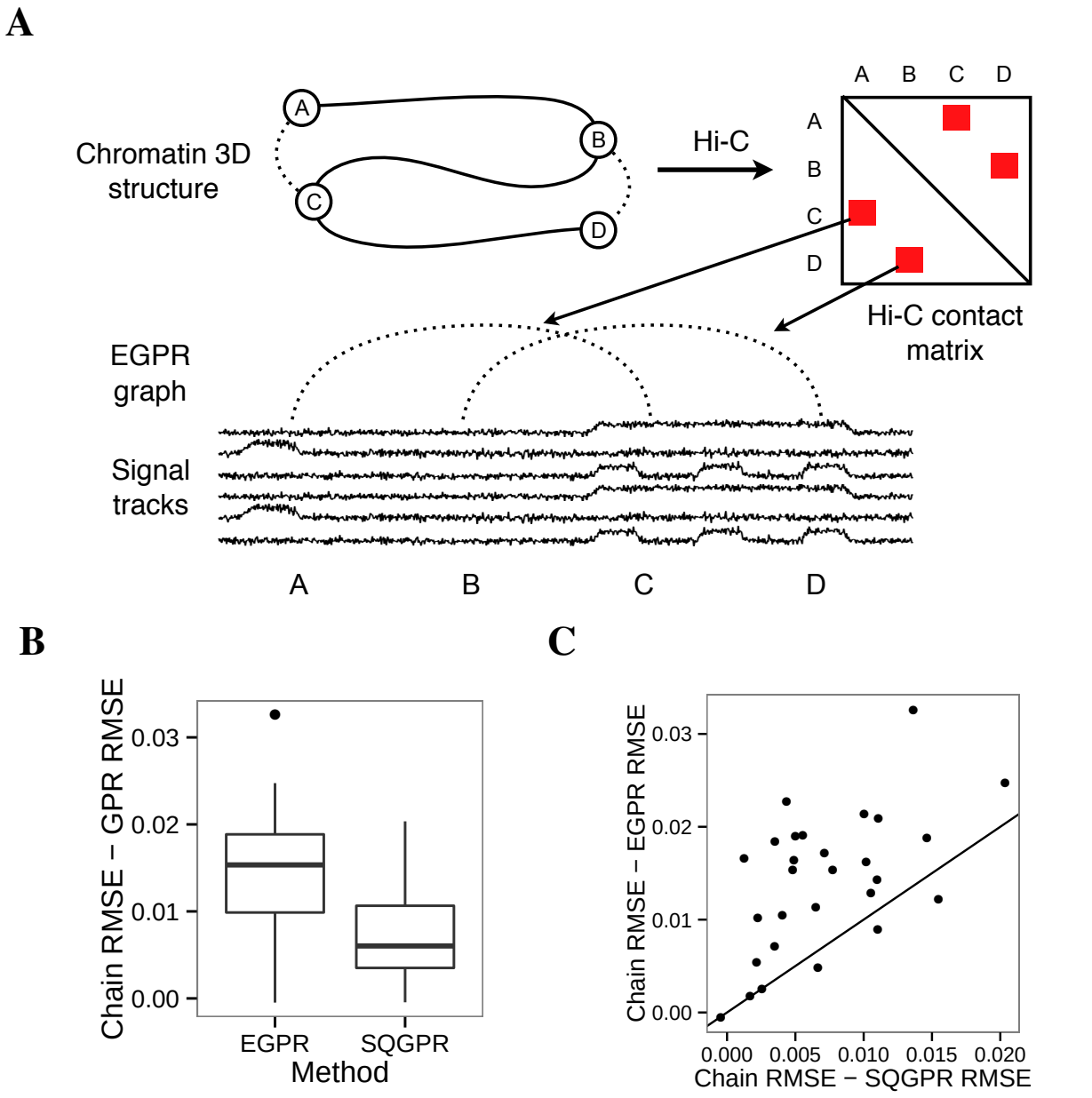


Figure 3.5: (A) Strategy for utilizing physical interaction information. (B) Improvement in RMSE over chain model for EGPR and SQGPR for 29 experiments. (C) Relative improvement in RMSE between EGPR and SQGPR for each of 29 experiments.

given label ℓ as

$$\mu_\ell \triangleq \frac{\sum_{i=1}^{n} \mathbf{1}(a_i = \ell) x_i}{\sum_{i=1}^{n} \mathbf{1}(a_i = \ell)} \quad \text{for } \ell \in \{1 \dots 4\}. \tag{3.54}$$

We define a predicted replication time vector $x_i^p = \mu_{a_i}$ and compute the root mean squared error (RMSE) of this prediction as RMSE $= \sqrt{\sum_i (x_i - x_i^p)^2}$. EGPR consistently outperforms both the chain model alone and SQGPR (Figure 3.5).

3.8 Discussion

We have defined entropic graph-based posterior regularization (EGPR), a method to encourage a model's posteriors to be smooth according to a regularization graph. This method is motivated by graph-based methods for semi-supervised learning, which have had great success in that setting but have not been studied thoroughly in an unsupervised setting. We showed that EGPR greatly outperforms both the approximate inference method loopy belief propagation and previous methods for graph-based regularization. We used EGPR to incorporate 3D structure data for semi-automated genome annotation and showed that EGPR greatly improved the quality of the resulting annotations. This method will thereby enable these methods to distill complicated pairwise contact matrices into human-interpretable genome annotations. Moreover, because EGPR can use any graphical model and similarity graph, it will likely have diverse applications in other fields such as natural language and time-series analysis.

3.9 Frequently Asked Questions

This section collects questions we have received when presenting this work.

1. **Why is KL divergence better than squared error for probability distributions?** Squared error is based on a Gaussian error model, which is not appropriate for probability values, and it under-penalizes differences between small probability values. p-norms are defined over all real numbers, while posteriors must lie within the range $[0, 1]$ (and live in a simplex). A more justified way to measure divergence between probability distributions is to employ the KL divergence. The KL divergence measures the difference of exponents in the probability and so evaluates differences between small and large probabilities more uniformly. Also, Pinsker's inequality [Csiszár and Tusnády, 1984] combined with the relationship of ℓ-norms implies

that $D(p\|q) \geq \frac{1}{2}\|p-q\|_1^2 \geq \frac{1}{2}\|p-q\|_\ell^2$, for all $\ell \geq 1$, where $\|\cdot\|_\ell$ is the ℓ-norm. Hence, minimizing KL divergence minimizes an upper bound on all ℓ-norms.

As a concrete example, consider two pairs of probability distributions over two possible events: $[0.55, 0.45]$ vs. $[0.45, 0.55]$, and $[0.1, 0.9]$ vs. $[10^{-10}, 1-10^{-10}]$. The first pair of distributions are very similar, with both events being roughly equally likely. The second pair is quite dissimilar, with the first event being reasonably likely in the first case and astronomically unlikely in the second. Squared error actually regards the first pair as more dissimilar, while KL divergence correctly identifies the second pair as much more dissimilar. Despite the advantages of KL divergence, although all posterior regularization objectives include a KL term binding q to be similar to p_θ, to our knowledge, no existing methods define the posterior regularizer **itself** using KL. This is also discussed in Section 3.2.

In practice, although a regularizer based on squared error shows good results for some problems, we demonstrate the better performance of KL divergence empirically in Sections 3.6 and 4.2.9.

2. **KL divergence is asymmetric in its two arguments. How does that affect the results?** As stated in Section 3.2, the KL divergence in Equation 4.6 is symmetrized because G_R is undirected—that is $w(u,v) = w(v,u)$—so $D(q_u\|q_v)$ and $D(q_v\|q_u)$ appear with the same weight in the regularizer. That is, $\sum_{u,v} w(u,v)D(q_u\|q_v) = \sum_{u \leq v} 2w(u,v)(D(q_u\|q_v) + D(q_v\|q_u))$.

3. **How does EGPR compare to other methods?** This topic is discussed in detail in Section 3.5.

   (a) **How does EGPR compare to approximate inference?** Posterior regularization is so named because it regularizes the model parameters indirectly via a penalty defined using the posterior distribution. It is therefore fundamentally different from approximate (or any) inference methods, which are used to perform probabilistic inference. Although

complementary, inference methods are quite a separate machine-learning concept from parameter regularization strategies.

In addition, we demonstrate empirically in Section 3.6 that EGPR outperforms the approximate inference algorithm loopy belief propagation (LBP) on a synthetic data set. (Note that LBP can be understood either in the context of belief propagation or as a variational method—see [Wainwright and Jordan, 2008].)

(b) **How does EGPR compare to the first posterior regularization method of [Ganchev et al., 2010]?** The method of [Ganchev et al., 2010] requires each posterior regularization term to involve variables in just one clique in the graphical model. Applying this method to a graph-based posterior regularization objective would result in a high tree-width model and intractable inference.

(c) **How does EGPR compare to the graph-based posterior regularization method of [He et al., 2013]?** He et al. [2013] present an approach based on an exponentiated gradient descent algorithm. Like our approach, He's approach exhibits monotone convergence. Although He's work has many similarities with our approach, He's work differs from ours in three important ways. First, He's method uses a squared-error penalty, which, as argued above, is less appropriate for probability distributions than Kullback-Leibler divergence [Bishop, 1995, p. 226]. As shown in Section 4.7, using squared error also results in worse performance in practice. Second, the exponentiated gradient descent method is applied to semi-supervised handwriting recognition and part-of-speech tagging, while we apply EGPR to an unsupervised genome annotation problem. Third, He et al. [He et al., 2013] use an exponentiated gradient descent strategy, while we use alternating optimization.

Our alternating optimization approach has several benefits over the exponentiated gradient descent method of He et al. [He et al., 2013]. First, the He et al. algorithm involves gradient calculations over each clique in the conditional dependence graph, with order $O(k^{|C|})$ for a variable of dimension k involved in cliques of size $|C|$. Our alternating

minimization algorithm, by contrast, has closed-form updates of order $O(k)$ for q, r^M and s^M. (Updates for θ can still involve $O(k^{|C|})$ calculations, but these are generally very fast). Therefore, the alternating optimization algorithm is more appropriate for extremely large models or models with large cliques or high-order factors. Second, empirical studies of KL-based graph smoothness objectives have shown that alternating optimization algorithms perform better than gradient-based methods for these objectives [Subramanya and Bilmes, 2011]. Finally, the alternating optimization strategy is parallelizable and extremely simple to implement because the graph regularization step has simple, closed-form updates with no learning rate hyperparameters, and the posterior calculation can be performed using any probabilistic inference method on a model with the same conditional dependence graph as the unregularized model.

3.10 Acknowledgments

This work was supported by NIH awards U41HG007000, R01ES024917, and R01GM103544 and by the Princess Margaret Cancer Foundation.

Chapter 4

JOINT ANNOTATION OF CHROMATIN STATE AND CHROMATIN CONFORMATION REVEALS RELATIONSHIPS AMONG DOMAIN TYPES AND IDENTIFIES DOMAINS OF CELL TYPE-SPECIFIC EXPRESSION

Abstract

The genomic neighborhood of a gene influences its activity, a behavior that is attributable in part to domain-scale regulation, in which regions of hundreds or thousands of kilobases known as domains are regulated as a unit. Previous studies using genomics assays such as chromatin immunoprecipitation (ChIP)-seq and chromatin conformation capture (3C)-based assays have identified many types of regulatory domains. However, due to the difficulty of integrating genomics data sets, the relationships among these domain types are poorly understood. Semi-automated genome annotation (SAGA) algorithms facilitate human interpretation of heterogeneous collections of genomics data by simultaneously partitioning the human genome and assigning labels to the resulting genomic segments. However, existing SAGA methods can incorporate only data sets that can be expressed as a one-dimensional vector over the genome and therefore cannot integrate inherently pairwise chromatin conformation data. We developed a new computational method, called graph-based regularization (GBR), for expressing a *pairwise prior* that encourages certain pairs of genomic loci to receive the same label in a genome annotation. We used GBR to exploit chromatin conformation information during genome annotation by encouraging positions that are close in 3D to occupy the same type of domain. Using this approach, we produced a comprehensive model of chromatin domains in eight human cell types, thereby revealing the relationships among known domain types. Through this model, we identified clusters of tightly-regulated genes expressed in only a small number of cell types, which we term "specific expression domains." We additionally

found that a subset of domain boundaries marked by promoters and CTCF motifs are consistent between cell types even when domain activity changes. Finally, we showed that GBR can be used for the seemingly unrelated task of transferring information from well-studied cell types to less well characterized cell types during genome annotation, making it possible to produce high-quality annotations of the hundreds of cell types with limited available data.

4.1 Introduction

Although the mechanism of regulation of a gene by a promoter directly upstream of its transcription start site is well understood, this type of local regulation does not explain the large effect of genomic neighborhood on gene regulation. The neighborhood effect is in part the consequence of domain-scale regulation, in which regions of hundreds or thousands of kilobases known as domains are regulated as a unit [Bickmore and van Steensel, 2013, Chakalova et al., 2005, Akhtar et al., 2013]. Current understanding of domain-scale regulation is based on a number of domain types, each defined based on a different type of data, such as histone modification ChIP-seq, replication timing, or measures of chromatin conformation. However, as a result of the difficulty of integrating genomics data sets, the relationships among these domain types are poorly understood. Therefore, a principled method for jointly modeling all available types of data is needed to improve our understanding of domain-scale regulation.

A class of methods we term *semi-automated genome annotation* (SAGA) algorithms are widely used to jointly model diverse genomics data sets. These algorithms take as input a collection of genomics data sets and simultaneously partition the genome and label each segment with an integer such that positions with the same label have similar patterns of activity. These algorithms are "semi-automated" because a human performs a functional interpretation of the labels after the annotation process. Examples of SAGA algorithms include HMMSeg [Day et al., 2007], ChromHMM [Ernst and Kellis, 2010], Segway [Hoffman et al., 2012] and others [Thurman et al., 2007, Lian et al., 2008, Filion et al., 2010]. These genome annotation algorithms have had great success in interpreting genomics data and have been shown to recapitulate known functional elements including genes, promoters and enhancers.

However, existing SAGA methods cannot model chromatin conformation information. The 3D arrangement of chromatin in the nucleus plays a central role in gene regulation, chromatin state and replication timing [Misteli, 2007, Dekker, 2008, Ryba et al., 2010, Dixon et al., 2012]. Chromatin architecture can be investigated using chromatin conformation capture (3C) assays, including the genome-wide conformation capture assay, Hi-C. A Hi-C experiment outputs a matrix of contact counts, where the contact frequency of a pair of positions is inversely proportional to the positions' 3D distance in the nucleus [Lieberman-Aiden et al., 2009, Ay et al., 2014b]. Existing SAGA methods can incorporate any data set that can be represented as a vector defined linearly across the genome, but they cannot incorporate inherently pairwise Hi-C data without resorting to simplifying transformations such as principle component analysis.

We present a method for integrating chromatin architecture information into a genome annotation method. Motivated by the observation that pairs of loci close in 3D tend to occupy the same type of domain, we encourage these pairs to be assigned the same label in a genome annotation through a *pairwise prior*. We developed a novel computational method, called graph-based regularization (GBR), which performs inference in the presence of such a pairwise prior, and we extended the existing SAGA algorithm Segway [Hoffman et al., 2012] to implement this method.

GBR can also be used for the seemingly-unrelated task of transferring information from well-studied cell types for the annotation of cell types with limited available data. Consortia such as ENCODE have characterized a small set of cell types in great detail using hundreds of genomics assays. However, due to the high cost of genomics experiments, it is feasible to perform only a few assays on any additional cell type of interest. For example, ENCODE and Roadmap Epigenomics have each performed 2-10 experiments in more than 100 cell types, and it is common for an individual lab to perform a small number of experiments on a particular cell type or perturbation of interest. In such settings, it is crucial to leverage information garnered from well-studied cell types to allow accurate annotation of other cell types using just a few experiments. We transfer information from well-studied cell types with GBR by using the pairwise prior that loci that were assigned the same label in many well-studied cell types should be more likely to receive the same label in a cell type of interest. Therefore, GBR makes it possible to produce high-quality annotations of the

hundreds of cell types with limited available data.

4.2 Results

4.2.1 Chromatin domains co-localize with domains of similar activity

Previous research has shown that large chromatin domains (∼1 Mb) tend to co-localize with domains of similar activity in 3D [Ryba et al., 2010, Lieberman-Aiden et al., 2009]. To further explore this trend, we compared the 1D genomics data at pairs of statistically significantly interacting loci (Supplementary Figure 4.1). As expected, we found that chromatin signals were highly consistent at pairs of positions nearby in 3D. First, histone modification and replication signal values at pairs of significantly interacting loci are more highly correlated than for a rotational permutation control, which controls for the 1D pattern of the signal (Supplementary Figure 4.1A). Second, Segway labels generated without using GBR from an annotation of the genome were assigned the same label much more often than a rotational permutation control (Supplementary Figure 4.1B). Note that this pattern is in stark contrast to small elements (∼100bp) such as promoters and enhancers, which do not, in general, cluster with elements of the same type. These observations suggest that chromatin conformation data might best be incorporated using a pairwise prior stating that a pair of positions should be more likely to receive the same label if the positions are close in 3D. Therefore, we sought to develop new methods for leveraging chromatin conformation data using this co-localization pattern.

4.2.2 Graph-based regularization expresses a pairwise prior in a SAGA method

Existing SAGA algorithms use dynamic programming algorithms to perform inference in a chain-structured Bayesian network such as a hidden Markov model. Dynamic programming algorithms such as the forward-backward algorithm can be used to perform inference efficiently in models with chain-structured dependencies; however, applying these methods in the presence of a pairwise prior that connects arbitrary pairs of positions results in inference costs which grow exponentially in the number of genomic positions. Therefore, these methods cannot be applied to genome annotation

problems with millions or billions of variables. We propose a novel convex optimization framework which allows for efficient inference in this case.

The method takes as input a set of genomics data sets and weighted graph over the genomic positions, where a large weight on a given pair of positions indicates that we have a strong prior belief that this pair should receive the same label. It outputs a probability distribution over the integer labels at each position. The method encourages pairs of positions connected by edges in the graph to be assigned the same label by minimizing a measure of dissimilarity between their output probability distributions called the Kullback-Leibler (KL) divergence.

We call this strategy of using a graph to incorporating a pairwise prior *graph-based regularization* (GBR) (Methods). Note that in this manuscript we use the word "prior" in the non-technical sense of "prior information," not in the sense of a prior distribution for a Bayesian model. We have developed an efficient, novel alternating minimization algorithm that optimizes this objective (Methods). Using synthetic data, we determined that graph-based regularization outperforms alternative methods based on approximate inference as well as existing methods for graph-based regularization (Chapter 3). We then extended the SAGA method Segway to implement this algorithm [Hoffman et al., 2012] (Section 4.4.3).

4.2.3 Using GBR to integrate 3D structure information improves prediction of replication and topological domains

We used graph-based regularization to integrate chromatin conformation information using the pairwise prior that positions close in 3D should be more likely to be identified as the same domain type (Figure 4.1A). To do this, we construct a GBR graph that connects each pair of positions with weight proportional to our statistical confidence that the positions physically interact [Ay et al., 2014a] (Methods). This measure of statistical confidence controls for the bias of Hi-C for positions close in 1D as well as biases for sequence features such as GC content and restriction site density.

Incorporating Hi-C data using GBR has the effect of both aligning domains to regions of self-interacting chromatin and helping to determine the label of each segment. We evaluated the first effect, as a sanity check, by computing the accuracy with which our annotation predicts

self-interacting regions of chromatin of size ∼1 Mb called topological domains [Dixon et al., 2012, Filippova et al., 2014]. We evaluated the second effect by comparing to replication time, which is highly correlated with gene expression and chromatin state and therefore is a good proxy for domain type. In order to evaluate our performance in a variety of conditions, we ran Segway augmented with GBR separately once for each of the 29 histone modification sets available in IMR90, in each case using as input a single histone modification data set and a GBR graph based on IMR90 Hi-C data. We found that the annotation's ability to identify both topological domains ($p < 10^{-16}$, t-test) and replication time ($p < 10^{-16}$, t-test) was greatly improved by using GBR (Figure 4.1B,C). To evaluate the degree to which an annotation matches topological domains, we computed, for each topological domain, the fraction of positions in the topological domain receiving the same label, controlling for the length and label distribution by comparing to a circularly permuted annotation (Methods). We measure the degree to which an annotation predicts replication time by the variance in replication timing explained by the annotation (Methods). The improvement from adding Hi-C with GBR was greater than the improvement achieved by instead adding another histone modification data set, for 26/30 and 27/30 histone modification data sets for topological domains and replication time, respectively. Furthermore, this improvement was consistent for a large range of hyperparameters (several order of magnitude around optimal, Figure 4.1B,C). These results demonstrate that incorporating Hi-C data using GBR greatly improves the quality of the resulting annotation, and moreover that Hi-C is more informative for determining domain identity than most other data types.

4.2.4 Joint domain annotation of chromatin state and chromatin conformation captures previously-described domain types

Having verified the utility of GBR for incorporating Hi-C data, we next sought to investigate domain-scale genome regulation using this method. Current understanding of domain-scale regulation is based on a number of domain types (seven, by our count), each defined based on a different type of data, such as histone modification, replication timing or 3C-based assays (Table 4.1). For example, ChIP-seq on the histone modification H3K27me3 has revealed repressive domains known

Name	Label(s)	Typical length	Relevant data types	References
Topological	All	0.8 Mb	Hi-C	Dixon et al. [2012], Filippova et al. [2014]
Open compartment / early replicating	BRD, SPC, FAC	10 Mb	Hi-C, Repli-(chip/seq)	Lieberman-Aiden et al. [2009] Ryba et al. [2010]
Closed compartment / late replicating	QUI, CON, FAC	10 Mb	Hi-C, Repli-(chip/seq)	Lieberman-Aiden et al. [2009] Ryba et al. [2010]
Quiescent	QUI	0.5 Mb	None	Ernst and Kellis [2010] Hoffman et al. [2012]
Constitutive heterochromatin	CON	0.5 Mb	H3K9me3 (ChIP-seq)	Lachner et al. [2003]
Facultative heterochromatin	FAC	0.5 Mb	H3K27me3 (ChIP-seq)	Morey and Helin [2010] Pauler et al. [2009]
Lamina	See text	0.6 Mb	lamin B1 (DamID)	Guelen et al. [2008] Wen et al. [2009]
Broad activity	BRD	0.5 Mb	Transcription (i.e. H3K36me3) (ChIP-seq)	Novel in human cells (see text)
Specific activity	SPC	0.5 Mb	Regulation (i.e. H3K27ac) (ChIP-seq)	Novel in human cells (see text)

Table 4.1: Known types of domains. Note that the lengths of domains depend greatly on the method used to define them.

as facultative heterochromatin [Morey and Helin, 2010, Pauler et al., 2009], and 3C-based assays have revealed regions of self-interacting chromatin known as topological domains [Dixon et al., 2012, Filippova et al., 2014]. Because all of these domain types are defined using different types of data, until now it has been difficult to understand the relationships among these domain types. GBR provides a principled method for integrating all types of data into a unified annotation of domains. We therefore used Segway with GBR to create such a unified annotation, in order to understand what types of domains exist and their interrelationships.

We annotated the cell type IMR90 using all 30 signal data sets we had available in IMR90 and a GBR graph derived from IMR90 Hi-C data, resulting in an annotation with median segment length of 0.4 Mb (Figure 4.2A,B,C, Methods, Supplementary Tables 4.1,4.2). Including Hi-C into this annotation using GBR changed the label of 6% of positions relative to an annotation without GBR, meaning Hi-C has a slightly larger influence than the $1/31 = 3\%$ difference one would expect from adding one additional data set (Supplementary Figure 4.2). Because determining the optimal number of labels for an annotation remains an open problem, we specified a somewhat larger number of

Data type	Figure	QUI	CON	FAC	BRD	SPC
Median segment length	4.2C	0.6 Mb	1.4 Mb	0.4 Mb	0.6 Mb	0.2 Mb
Histone modifications	4.2A,D	None	H3K9me3	H3K27me3	Transcription (i.e. H3K36me3)	Regulation (i.e. H3K27ac)
Replication timing	4.2A,D	Switching late	Constitutively late	Mixed	Constitutively early	Switching early
GC content	4.3D	Low	Low	High	Mid	High
Conservation	4.3E	Conserved	Nonconserved	Conserved or accelerated	Conserved	Conserved
Hi-C eigenvalue compartment	4.3F	Closed	Closed	Mixed	Open	Open
Gene density	4.3A	Low	Low	High	High	High
Gene expression	4.3B	N/A	N/A	Repressed	Average	Increased
Lamin	4.4	Core	Core	Boundaries	Flanks	Flanks

Table 4.2: Summary of learned domain types.

labels than we expected to be supported in the data (eight), then manually merged labels that we deemed to be redundant.

We compared our annotation to eight types of features (Table 4.2). On the basis of these analyses, we merged labels that appeared redundant and assigned names to each integer label (or group of labels) that best match our interpretation of their function. This procedure yielded five types of domains: (1) broad expression (BRD), (2) specific expression (SPC), (3) facultative heterochromatin (FAC), (4) constitutive heterochromatin (CON), and (5) quiescent (QUI). We describe the analyses that led us to these names in the following sections.

In order to understand how domains change state between cell types, we additionally annotated eight cell types using twelve data sets present in all eight types and a GBR graph representing common 3D contacts generated by combining IMR90 and H1-hESC Hi-C data sets (Methods, Supplementary Table 4.2). This strategy of combining Hi-C data sets is motivated by the consistency of Hi-C across cell types (Supplementary Figure 4.3). Again, we used eight labels and merged redundant labels, to which we assigned the same five names. We investigated the properties of these five domain types.

4.2.5 Repressive domains are divided into constitutive, facultative and quiescent heterochromatin

Previous studies have reported two types of repressive domains. The first type, best known as "constitutive heterochromatin" but sometimes referred to simply as "heterochromatin", is regulated by the HP1 complex and associated with the histone modification H3K9me3 [Lachner et al., 2003]. Constitutive heterochromatin is thought to repress permanently silent regions such as centromeres and telomeres. As expected, one output domain type "CON" exhibits all the known properties of constitutive heterochromatin. CON domains are associated with H3K9me3 (Figure 4.2A,D), are extremely depleted for genes (Figure 4.3A), are associated with low GC content and lack of evolutionary conservation (Figure 4.3D,E), appear within the Hi-C eigenvector closed compartment (Figure 4.3F, Methods), and cover regions which are constitutively late replicating in all cell types (Figure 4.3G). CON domains are depleted both for transcription factor motifs and for transcription factor binding at motifs (Figure 4.3H,I).

The second known type of repressive domain is best known as "facultative heterochromatin" but is also sometimes referred to as BLOCs or Polycomb-repressed chromatin [Morey and Helin, 2010, Pauler et al., 2009]. Facultative heterochromatin is regulated by the Polycomb complex and is associated with the histone modification H3K27me3. Facultative heterochromatin is thought to repress tissue-specific genes in cells where they are inactive. As expected, one output domain type "FAC" has all the known properties of facultative heterochromatin. FAC domains are marked by H3K27me3 (Figure 4.2A,D), and they are enriched for genes (Figure 4.3A), GC content (Figure 4.3D) and conservation (Figure 4.3E), but strongly depleted for gene expression relative to an average across cell types (Figure 4.3B), indicating that FAC domains have a direct repressive effect. FAC domains are mixed between the open and closed compartments, indicating that facultative repression is independent of compartment-driven repression (Figure 4.3F). However, FAC domains are almost completely absent from the annotation of the embryonic stem cell line H1-hESC, consistent with previous observations that H3K27me3 does not form domains in embryonic stem cells but rather occurs only at so-called poised or bivalent promoters (Supplementary Figure 4.4) [Bernstein et al., 2006].

Other semi-automated genome annotation analyses have reported a third type of repressive domain, characterized by a lack of signal from any mark, termed "quiescent domains" [Hoffman et al., 2012, Ernst and Kellis, 2010, Filion et al., 2010, Julienne et al., 2013]. We identified this domain type as the QUI label (Figure 4.2A). Note that Segway marginalizes over missing data rather than setting the values to zero (Section 4.4.3), so the QUI label is not simply an artifact of unmappable regions. QUI domains are highly depleted for genes (Figure 4.3A) and occur in the closed compartment (Figure 4.3F). QUI domains are depleted for transcription factor motifs but, unlike FAC and CON domains, are not depleted for transcription factor binding at motifs, indicating that QUI chromatin does not have a direct repressive effect (Figure 4.3H,I). The mechanism behind the activity of QUI domains is unknown, but these results are consistent with a model in which QUI domains lack any activating signals but are not directly repressed.

4.2.6 Active domains are divided between broad and specific gene expression

Previous studies of human domains have focused on various types of repressive domains but have assigned all active chromatin to one domain category [Julienne et al., 2013, Wen et al., 2009, Pauler et al., 2009]. However, studies in other organisms have reported multiple types of active domains [Filion et al., 2010, Liu et al., 2011]. We therefore investigated whether our IMR90 annotation can be used to identify types of human active domains. We found that active domains in IMR90 can be split into BRD ("broad expression") domains, characterized by transcription-associated marks such as H3K36me3, and SPC ("specific expression") domains, characterized by regulatory marks such as H3K27ac. Both domain types are highly enriched for genes (Figure 4.3A). However, while genes in BRD domains are mostly expressed across all cell types, a much larger fraction of active genes in SPC domains are expressed only in a small number of cell types (Figure 4.3C). Furthermore, when a gene is in a SPC domain, that gene is expressed at a much higher level than that gene's average across cell types, suggesting that SPC domains are highly activating (Figure 4.3B). In contrast, while genes in BRD domains are highly expressed, this high expression generally occurs consistently across cell types, indicating that BRD domains do not necessarily directly promote expression (Figure 4.3B). Moreover, while both BRD and SPC domains are generally early-replicating in

IMR90, regions covered by SPC domains typically switch replication time between cell types, while regions covered by BRD domains are typically early replicating in all cell types (Figure 4.3G). These results suggest a model in which genes performing housekeeping functions such as DNA repair have strong promoters but little other regulation, whereas genes specific to a given tissue are regulated by a complex web of regulatory elements, allowing the genome to specify precise conditions under which the gene is active.

To test this hypothesis, we computed the enrichment of Gene Ontology (GO) terms for genes in BRD and SPC domains respectively [Gene Ontology Consortium, 2000a, Boyle et al., 2004]. We found that genes in BRD domains were enriched for housekeeping functions such as cell cycle and DNA repair, while genes in SPC domains were enriched for IMR90-specific developmental functions such as vasculature development and stimulus response (Supplementary Tables 4.3–4.4, Supplementary Figure 4.5). In order to avoid hindsight bias, before looking at these GO term enrichments, we mixed the enriched terms with an equal number of decoy terms matched according to the number of genes associated with each term, and manually labeled which terms matched our hypothesized functions for each domain (housekeeping for BRD, IMR90-specific for SPC). We correctly identified 21/32 BRD enrichments ($p = 0.055$) and 54/64 SPC enrichments ($p = 1.4 \times 10^{-6}$). This demonstrates that active regions can be divided into domains of broadly-expressed housekeeping genes and domains of specifically-expressed developmental genes. To our knowledge, this is the first time a split between domains of broad and specific expression has been reported in human cells.

4.2.7 Lamina association is driven by a complex structure of domains

Previous work has shown that some repressive domains are marked with the histone modification H3K9me2, associate with the factor lamin B1 and localize to the nuclear lamina [Guelen et al., 2008, Wen et al., 2009]. We found that comparing lamina association to domain annotations based on many data sets reveals a much more complex interaction than comparing to each mark individually (Figure 4.4). As expected, repressive domains (QUI and FAC) are enriched inside lamina-associating chromatin domains, while active domains are depleted. However, this analysis also reveals that

CON domains are depleted immediately inside lamina-associating domain boundaries while being comparatively enriched at their centers. In contrast, FAC domains are highly enriched at lamina-associating domain boundaries while being comparatively depleted at their centers. In addition, while active domains (SPC and BRD) are depleted inside lamina-associating domains, they are highly enriched directly outside their boundaries. These observations suggest that lamina-associating domains form around a core of repressed chromatin and spread until they hit a strong active element.

4.2.8 Developmentally-consistent domain boundaries are marked by identifiable sequence elements

Previous research has shown that domain boundaries tend to be consistent between cell types even when the state of the domain changes. For example, when a region's replication time is perturbed by leukemia, the boundaries of the resulting replication domain tend to occur at the same positions as developmental replication timing domain boundaries [Ryba et al., 2012]. However, the cause of these consistent domain boundaries remains unclear. We investigated the consistency of domain boundaries using our domain annotations. As expected, domain boundaries frequently occurred at consistent positions across cell types, even when the domains' state changed (Figure 4.5A). To identify these consistent domain boundaries, we combined all boundaries occurring in at least one cell type and merged boundaries within 50 kb. We defined groups of five or more boundaries as *consistent* (Methods) (Figure 4.5B). As expected, these consistent boundaries are enriched for replication domain boundaries, but many consistent domain boundaries do not overlap a replication domain boundary (Supplementary Figure 4.6). We additionally found that consistent domain boundaries are highly enriched for promoters and CTCF motifs, suggesting that these elements may drive domain boundary formation (Figure 4.5C,D).

4.2.9 *Using GBR to transfer information between cell types improves accuracy of predicting functional elements*

Graph-based regularization can also be used for the seemingly-unrelated task of transferring information from well-studied cell types for the annotation of cell types with limited available data (Figure 4.6A). Existing SAGA methods work well on data from a single cell type, but integrating information between cell types remains an open problem. Existing methods for using data from multiple cell types for genome annotation fail to effectively address this problem (Supplementary Note 4.6.1). We propose a novel strategy for leveraging information from well-studied cell types using the pairwise prior that if two positions received the same label in many well-studied cell types, then they should be more likely to receive the same label in the target cell type (Figure 4.6A). To express this pairwise prior, we first perform a Segway annotation (without GBR) of each well-studied cell type and create a GBR graph which connects each pair of positions with weight proportional to the number of cell types in which the pair receive the same label, placing higher weight on cell types similar to the cell type of interest (Methods). We then use this graph in combination with the data sets available in the target cell type to produce an annotation of this cell type. Note that this GBR graph represents an entirely different type of information from the graphs used to represent Hi-C data in the previous sections, despite the fact that both types of data are represented as a graph.

To demonstrate the efficacy of this approach, we evaluated whether GBR improves an annotation's ability to predict enhancers and insulators. We simulated the case where the lymphoblastoid cell type GM12878 has only eight histone modifications available, a panel of data types similar to that assayed by Roadmap Epigenomics on hundreds of human tissues (Supplementary Note 4.2). Because there are enough well-studied cell types to ensure that at least one reference is reasonably closely related to any cell type of interest, we used the related leukemia cell type K562 as reference. We annotated GM12878 using these eight histone modifications and a GBR graph derived from an annotation of K562 (Methods). Incorporating information from K562 this way greatly improved the accuracy with which the annotation detected enhancers and insulators (Figure 4.6B,C). We evaluated the performance with which the GM12878 annotation predicts a certain type of functional

element by ordering the labels by their enrichment for the element on a training set and evaluating the recall as more labels are added (Methods). The GBR annotation detects one third of EP300 binding sites (a proxy for enhancers [Visel et al., 2009]) by predicting just 25 kb as EP300-binding, while the annotation produced without GBR predicts 43 kb before it detects this many sites (Figure 4.6B). Likewise, the GBR annotation detects one third of CTCF binding sites (a proxy for insulators [Burgess-Beusse et al., 2002]) by predicting 124 kb, compared to 241 kb without GBR (Figure 4.6C). Because the algorithm was not given any knowledge of enhancers or insulators as input, it is reasonable to expect that the annotations achieve similar performance at detecting other types of functional elements, for which we do not have gold-standard examples and therefore cannot evaluate against. These results demonstrate that GBR effectively leverages information from a reference cell type and therefore provides a method for producing high-quality annotations of the hundreds of cell types with limited available data.

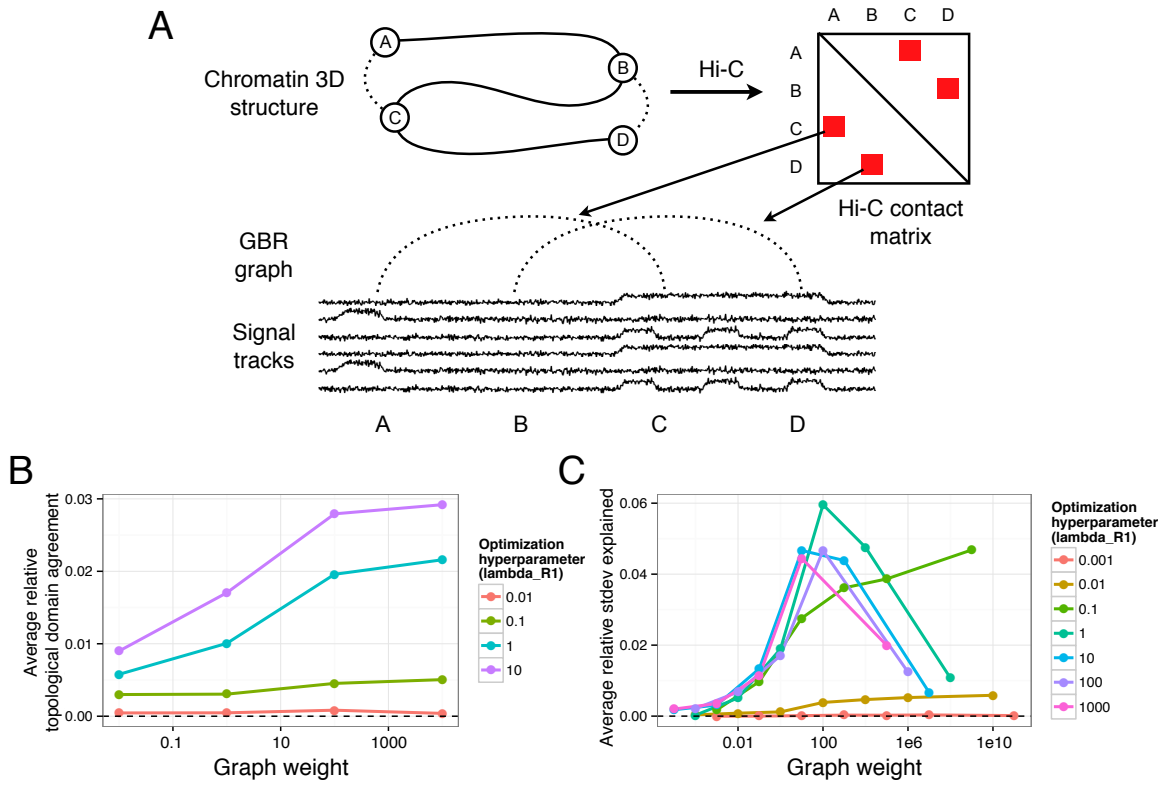


Figure 4.1: (A) Strategy for incorporating Hi-C data using GBR. (B) Effect of GBR hyperparameters on topological domain agreement. The X axis indicates the value of the graph weight hyperparameter λ_G. Y axis indicates the average over 10 annotations (one for each input histone modification data set) of the fraction improvement in topological domain label agreement by adding GBR over Segway without GBR. (C) Same as (B), but Y axis indicates improvement in replication timing standard deviation explained and average is over 29 annotations. Annotations used four labels, 10kb resolution.

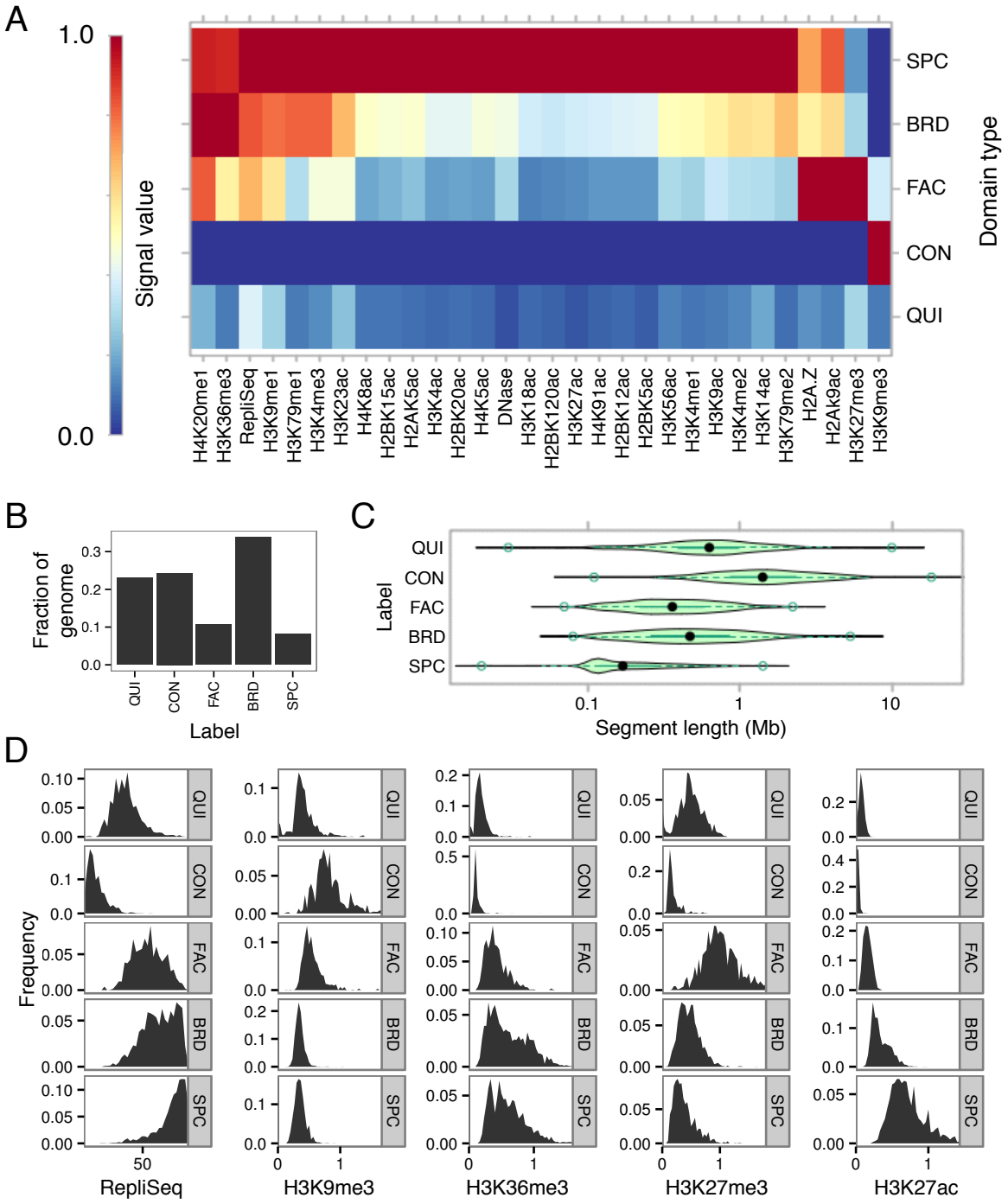


Figure 4.2: Statistics of domain annotation made with Segway augmented with GBR. (A) Heatmap of association of IMR90 domain labels to IMR90 data sets used in training. (B) Fraction of genome covered by each domain type. (C) Segment length distribution for each domain type. (D) Label-specific histograms of values for five notable data sets in IMR90.

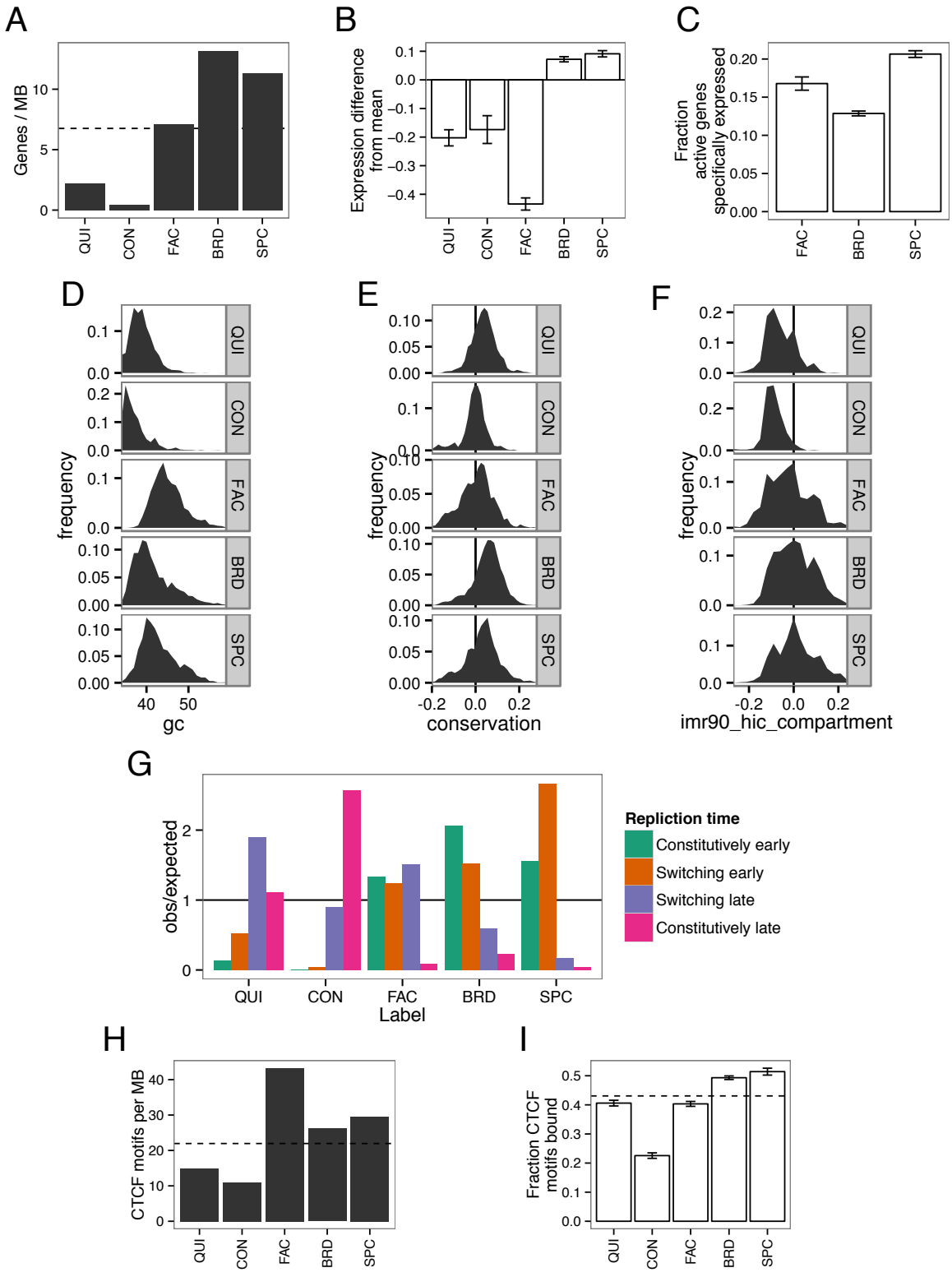


Figure 4.3: (Caption on following page.)

Figure 4.3: Characteristics of domain types. (A) Gene density in IMR90. (B) Gene expression relative to average over 33 cell types, averaged over eight annotations (t-test 95% confidence interval error bars). (C) Fraction of active genes also active in more than 15 other cell types, averaged over eight annotations (binomial test 95% confidence interval error bars). All gene expression data is from CAGE (Methods). (D-F) Histograms of (D) GC content, (E) conservation (PhyloP score) [Siepel et al., 2005] and (F) Hi-C compartment eigenvalues in IMR90. Positive PhyloP scores indicate evolutionary conservation, positive scores indicate accelerated evolution, and scores near zero indicate neutral evolution. Hi-C compartment values are computed according to the method of [Lieberman-Aiden et al., 2009] (Methods). Positive values indicate open compartment, negative values indicate closed compartment. (G) Enrichment of each domain type with respect to IMR90 replication time (early vs. late) and replication time dynamics across cell types (constitutive vs. switching) (V Dileep, F Ay, J Sima, WS Noble, DM Gilbert et al., unpublished data) (Methods). (H) CTCF motif density for each domain type. Dashed line indicates genome-wide average. (I) Fraction of CTCF motifs bound (overlapping a CTCF peak) in IMR90 (binomial test 95% confidence interval error bars). Dashed line indicates average over all motifs.

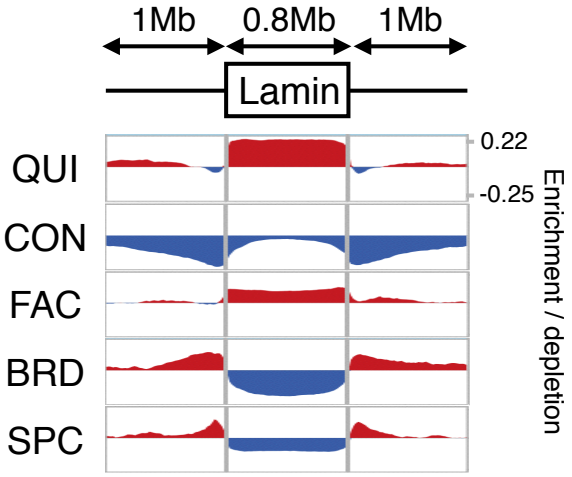


Figure 4.4: Enrichment of each domain type with respect to lamina-associating domain boundaries. X axis indicates position with respect to lamina-associating domains, with each domain stretched or shortened to the median length of 0.8 Mb. Y axis indicates label enrichment or depletion ($\log(\text{obs}/\text{expected})$).

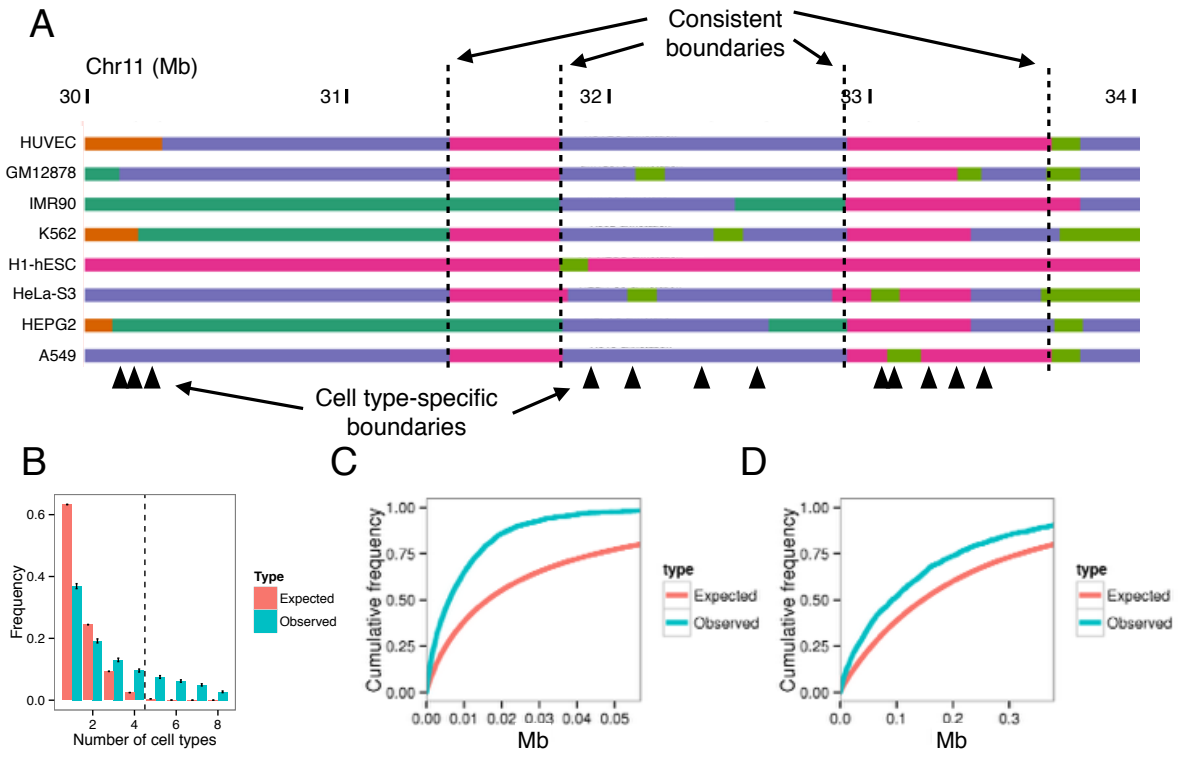


Figure 4.5: (A) Example of consistent domains. (B) Distribution of number of overlapping boundaries compared to a permutation control (binomial test 95% confidence interval error bars). Vertical dashed line denotes consistent boundary threshold. (C,D) Cumulative density of distances from consistent domain boundaries to the nearest (C) promoter and (D) distal CTCF motif. Distal CTCF motifs are defined as all CTCF motifs more than 5 kb from a promoter.

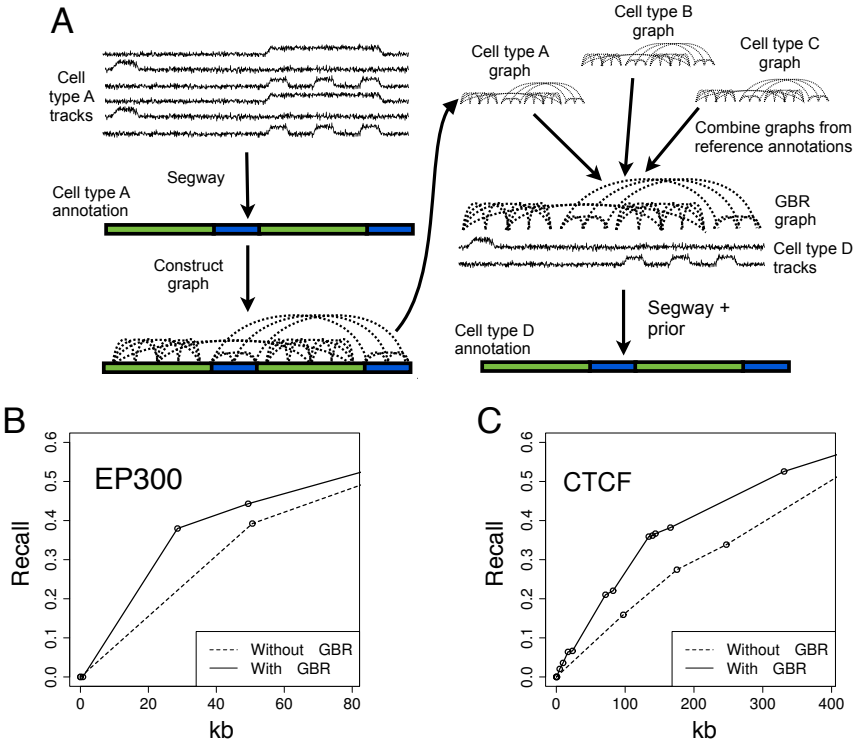


Figure 4.6: (A) Strategy for using GBR to transfer information between cell types. (B,C) Performance of predicting the locations of GM12878 (B) EP300 and (C) CTCF ChIP-seq peaks with and without using GBR to integrate information from K562. Each plot shows the fraction of elements detected as a function of the number of bases predicted (Methods). Results are shown on the test set (10 Mbp). Model training and label ordering was performed on the training set (10 Mbp). The X axis is plotted up to 1 kbp times the total number of elements. These plots can be interpreted, for example, in the context of an enhancer validation experiment, in which case it shows how many sequences would need to be tested in order to discover a certain number of enhancers.

4.3 Discussion

We introduced graph-based regularization (GBR), a method which allows probabilistic models to integrate a pairwise prior while maintaining efficient inference. We used GBR to model chromatin conformation data and thereby jointly model all available data types for the study of chromatin domains. To our knowledge, this represents the first method for integrating chromatin conformation information into SAGA methods without resorting to simplifying transformations. We showed that modeling Hi-C data with GBR improved the annotation's ability to predict replication time and topological domains. In addition, because graph-based regularization is a general method, it will likely prove useful for other applications involving dynamic Bayesian networks, such as methods for locating genes or predicting copy number.

The ability to integrate Hi-C data into an annotation allowed us to study the relationship between types of domains by integrating all available data into a single annotation (Figure 4.7A). This analysis revealed a set of five domain types that encompass all previously-described domain types: (1) quiescent domains, which lack any activity, (2) constitutive heterochromatin, which represses permanently silent regions and is marked with the histone modification H3K9me3, (3) facultative heterochromatin, which represses cell type-specific regions and is marked with the histone modification H3K27me3, (4) broadly-expressed domains, which cover genes that are highly expressed in all cell types, and (5) specifically-expressed domains, which exhibit high regulatory activity and cover genes that are expressed in a small number of cell types.

To our knowledge, domains of specific expression (SPC) have not been identified previously in human cells. These domains are likely the result of complex regulatory programs designed to precisely control the condition and level of genes important for a certain cell state or function. SPC domains are similar in some ways to dense clusters of regulatory elements important for cell identity known as superenhancers [Whyte et al., 2013, Lovén et al., 2013]. However, there are only small number of known superenhancers ($\sim$300) and each is much smaller than a SPC domain ($\sim$10 kb compared to $\sim$200 kb). Therefore, SPC domains and superenhancers may result from similar mechanisms, but on very different scales. However, the mechanisms underlying of both

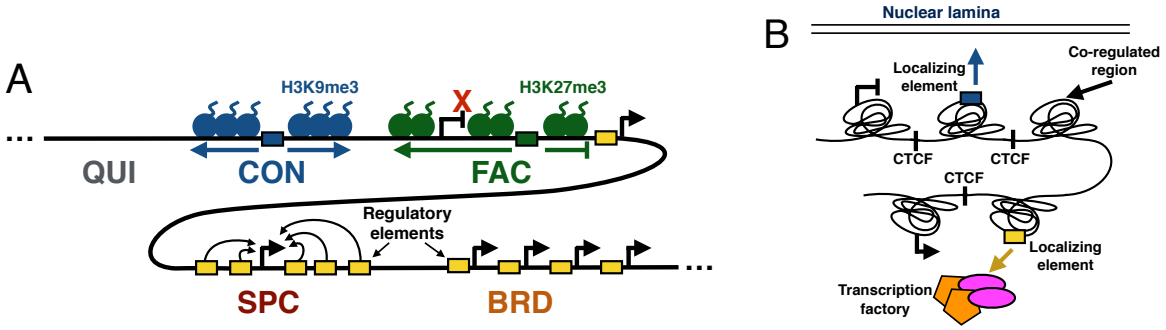


Figure 4.7: Models of (A) domain types and (B) co-regulated regions.

types of regions must be studied further in order to understand this relationship.

One likely mechanism of domain formation involves the spreading of heterochromatin [Weiler and Wakimoto, 1995, Talbert and Henikoff, 2006]. Under this hypothesis, heterochromatin nucleates at silencing elements such as telomeres, repeats or repressed promoters and sequentially assembles along chromatin. Spreading heterochromatin has been demonstrated mechanistically in *Saccharomyces* and *Drosophila* and for the SIR, HP1 and Polycomb complexes. While SIR is unique to yeast, HP1 and Polycomb have orthologs in humans that drive constitutive and facultative heterochromatin, respectively. Under the spreading hypothesis, heterchromatin can be halted by the presence of a strong active element. This halting mechanism is consistent with the observation that active domains (especially SPC) are strongly enriched directly outside of lamina-associating domains.

The consistency of domain boundaries between cell types also suggests a model in which core regions are regulated as a unit [Phillips and Corces, 2009, Dixon et al., 2012] (Figure 4.7B). Under this hypothesis, these units self-interact as topological domains and are co-regulated through availability of regulatory factors and elements such as enhancers. These co-regulated units are thought to be delimited by localizing sequences, particularly CTCF sites. Under this model, each of our annotated domains is actually composed of several such neighboring co-regulated regions with the same state. Therefore, while profiling a small number of cell types has allowed us to define a small number of consistent domain boundaries, profiling more cell types may lead to a complete

catalogue of potential boundary sites.

We have described five domain types because we this model allowed us to concisely summarize domain regulation, but we do not claim that this represents the "true" number of domain types. It is likely that new domain subtypes will be discovered in the future, thus increasing the number of known domain types. In addition, methods that discover the optimal number of domain types or that allow mixtures of domain types are an interesting direction for future work. To our knowledge, all existing SAGA methods require either a fixed number of labels [Day et al., 2007, Hoffman et al., 2012, Ernst and Kellis, 2010, Thurman et al., 2007, Lian et al., 2008, Filion et al., 2010] or a hyperparameter that indirectly controls the number of labels [Ho et al., 2013]. A method that allows for "mixed" domain labels at a given process could potentially circumvent the manual merging process that we used to reduce an eight-label model to a five-label one.

Finally, we presented a method for transferring information from well-studied cell types using GBR in order to improve the quality and interpretability of annotations of cell types with limited available data. This method enables a new strategy for understanding cell types, in which a small number of assays are performed on each cell type of interest to determine the unique characteristics of this cell type, then Segway with GBR is used to combine this data with the large body of available information from well-studied cell types. This method has the additional benefit of matching the label semantics of the target cell types to the semantics of the reference annotations, which allows the label interpretation process to be performed automatically. Because consortia such as ENCODE and Roadmap Epigenomics are already analyzing a large number of cell types with a small number of assays each, this strategy is immediately applicable. Determining which assays are most informative as input to this strategy is an interesting question for future work.

4.4 Methods

4.4.1 Histone modification, open chromatin and replication timing signal data

We acquired histone ChIP-seq, DNase-seq, and FAIRE-seq data for A549, K562, H1-hESC, GM12878, HeLa-S3, HepG2 and HUVEC from ENCODE and for IMR90 from Roadmap Epige-

nomics [ENCODE Project Consortium, 2012, Bernstein et al., 2010]. We used a uniform signal-processing pipeline to generate a genome-wide vector for each data set, as described in [Hoffman et al., 2013]. We also acquired Repli-seq data for IMR90 from ENCODE and smoothed this data using wavelet smoothing as described in [Thurman et al., 2007]. We applied the inverse hyperbolic sine transform $\text{asinh}(x) = \ln(x + \sqrt{x^2 + 1})$ to all signal data. This transform is similar to the log transform in that it depresses the magnitude of extremely large values, but it is defined at zero and amplifies the magnitude of small values less severely than the log transform does. This transform has been shown to be important for reducing the effect of large values in analysis of genomics data sets [Johnson, 1949, Hoffman et al., 2012].

We acquired transcription factor ChIP-seq data from ENCODE. Peaks were called for each factor using MACS using an IDR threshold of 0.05 [Zhang et al., 2008, Landt et al., 2012].

We acquired CAGE expression data for 33 cell types from GENCODE [Harrow et al., 2012].

The full list of data sets used is available in Supplementary Table 4.1.

4.4.2 Hi-C data

We used publicly available Hi-C data sets for two human cell lines (IMR90 and H1-hESC) [Dixon et al., 2012]. We processed raw paired-end libraries with a pipeline that combines reads from two replicates per cell line, maps these reads, extracts the read pairs for which each end maps uniquely, and removes potential PCR duplicates. We then partitioned the human genome into a collection of non-overlapping 10 kb windows and assigned each end of a read pair to the nearest 10 kb window mid-point. This process yielded a 303,641$\times$303,641 whole genome contact map. These contact maps consisted of both intra and interchromosomal contacts and contained only $\sim$0.3% non-zero entries. We assigned statistical confidence estimates to these contact counts using the method Fit-Hi-C, which jointly models the random polymer looping effect and technical biases [Ay et al., 2014a]. First, we applied the bias correction method ICE to the contact map to estimate a bias associated with each 10 kb locus, after eliminating all loci that have less than 50% uniquely mappable bases [Imakaev et al., 2012]. Second, using these computed biases and raw contact maps as input, we estimated a p-value of interaction for each pair of 10 kb loci with non-zero contact

counts (p-value was set to 1 for pairs with zero contacts). We used a slightly modified version of the original Fit-Hi-C algorithm which handles inter- as well as intrachromosomal contacts and omits the refinement step for fast computation. Because Fit-Hi-C normalizes for 1D genomic distance, the majority of significant contacts were at long distances (Supplementary Figure 4.7). Note that while the data sets we used have insufficient coverage to identify many high-confidence contacts at 10kb resolution, Segway with GBR aggregates information over roughly 400kb in order to make each domain call, so individual high-confidence interactions are not necessary.

We computed the genome chromatin compartment using eigenvalue decomposition on the normalized contact maps of IMR90 and H1-ESC cell lines at 1 Mb resolution as described in [Lieberman-Aiden et al., 2009]. For each chromosome, we calculated the Pearson correlation between each pair of rows of the intrachromosomal contact matrix and applied eigenvalue decomposition to the correlation matrix. Similar to [Lieberman-Aiden et al., 2009], we used the second eigenvector in cases where the first eigenvector values were either all positive or all negative to define the compartments. We used average GC content to map signs of eigenvectors to either open (higher GC content) or closed chromatin compartments.

4.4.3 Segway model

We used graph-based regularization to augment the Segway semi-automated genome annotation method [Hoffman et al., 2012]. Segway uses a dynamic Bayesian network model to perform genome annotation. The model is presented in detail in [Hoffman et al., 2012], but we describe it briefly here.

- We define a latent label variable $Y_i \in \{1..K\}$ for each position $i \in \{1..N\}$ in the genome, where K is the user-specified number of labels and N is the number of positions.

- We define observed signal data variables $X_{i,j}$ representing the value of signal data set $j \in \{1..M\}$ at genomic position i, where M is the number of signal data sets. We downsample the genome into bins of size R and average the signal data in each bin (after applying the inverse hyperbolic sine transform), so $N \approx 3 \times 10^9/R$. Because the sequencing depth of

existing Hi-C data sets is too low to achieve single base pair resolution, we used $R = 10000$ for experiments using GBR to integrate Hi-C data. We used $R = 1$ for experiments using GBR to transfer information between cell types.

- The observed data variable $X_{i,j}$ depends only on the label at position i, Y_i. We model the variable $X_{i,j}$ as a Gaussian distribution with data set- and label-specific mean parameter $\mu_{i,j}$ and data set-specific variance parameter σ_j. In the case that some data values are missing due to mappability, we weight the observation of $X_{i,j}$ by the proportion of mappable positions $\mathring{X}_{i,j}$ in bin i in data set j.

- The label variable Y_i depends only on the label at the previous position Y_{i-1}. We model the label transition from label a to label b using a transition parameter $Q_{a,b}$.

- We model segment length, determined by self-transitions $Q_{a,a}$, separately from label transitions, determined by $Q_{a,b}$ for $a \neq b$. The self-transition is weighted by a hyperparameter $\lambda_{\text{transition}}$, which weighs the importance of segment length relative to signal data.

- The parameters $\mu_{1:K,1:M}$, $\sigma_{1:M}$ and $Q_{1:K,1:K}$ are learned through EM.

The overall log-likelihood of the Segway model is defined as:

$$\begin{aligned}\log \Pr(X, Y \mid \mu, \sigma, Q) = &\sum_{i=1}^{N}\sum_{j=1}^{M} \mathring{X}_{i,j} \log N(X_{i,j} \mid \mu_{Y_i,j}, \sigma_j) \\ &+ \lambda_{\text{transition}} \sum_{i=1}^{N-1} \mathbf{1}(Y_i == Y_{i+1}) \log Q_{Y_i,Y_i} \\ &+ \sum_{i=1}^{N-1} \mathbf{1}(Y_i \neq Y_{i+1})(\lambda_{\text{transition}} \log(1 - Q_{Y_i,Y_i}) + \log Q_{Y_i,Y_{i+1}})\end{aligned} \tag{4.1}$$

where $\mu_{\ell,j}$ is the mean associated with signal data set j and label ℓ; σ_j is the variance associated with signal data set j (shared between all labels); $\mathring{X}_{i,j}$ is the proportion of mappable positions in bin

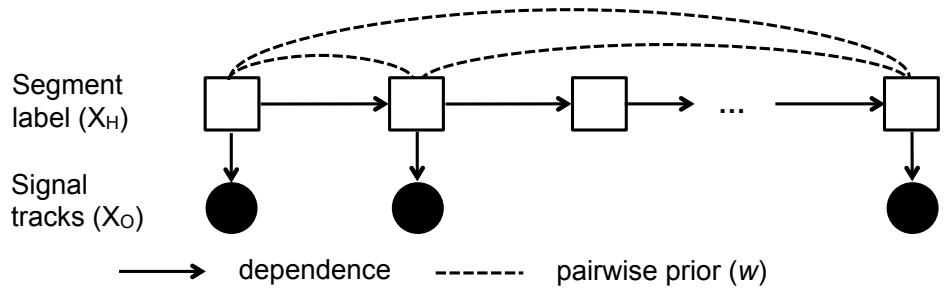


Figure 4.8: GBR model. Squares and circles denote discrete and continuous random variables respectively. Filled-in and unfilled shapes denote observed and unobserved variables, respectively.

i for data set j; $Q_{a,b}$ is the transition probability parameter from label a to label b; and $\lambda_{\text{transition}}$ is a weight on the transitions relative to the emissions of the model.

4.4.4 Graph-based regularization

In a SAGA method, we are given a set of vertices V that index a set of $n = |V|$ random variables $X_V = \{X_1, \ldots, X_n\}$ and a probability distribution parameterized by θ, $p_\theta(X_H, X_O)$. Different SAGA methods employ different distributions p_θ. Graph-based regularization could be applied to any probabilistic model, but in this work we use the Segway model (Supplementary Methods) because it can handle real-valued and missing data, and it can use non-geometric segment length distributions. We denote random variables with capital letters (e.g., X_H) and instantiations of variables with lower-case (e.g., $x_H \in \text{domain}(X_H)$). We use capitals to denote sets and lowercase to denote values (e.g., X_h for $h \in H$).

Training the model involves a set of observed data $\bar{x}_O$, where a subset of variables $O \subseteq V$ is observed, and the remainder $H = V \setminus O$ are hidden. The maximum likelihood training procedure optimizes the objective

$$\text{maximize}_\theta \qquad J(\theta) \triangleq \mathcal{L}(\theta) + \mathcal{R}(\theta) \tag{4.2}$$

$$\text{where} \qquad \mathcal{L}(\theta) \triangleq \log p_\theta(\bar{x}_O) = \log \sum_{x_H} p_\theta(x_H, \bar{x}_O), \tag{4.3}$$

where $\mathcal{R}(\theta)$ is a regularizer that expresses prior knowledge about the parameters. Many regularizers are used in practice, such as the ℓ_2 or ℓ_1 norms, which encourage parameters to be small or sparse respectively.

Dynamic programming algorithms such as the forward-backward algorithm can be used to perform inference in SAGA models, because all such existing models have dependencies in the form of a chain. That is, the variables associated with position i depend only on the variables associated with positions $i-1$ and $i+1$. Examples of such chain-structured models include hidden Markov models and dynamic Bayesian networks. However, these dynamic programming algorithms do not apply if a pairwise prior is added to the model, since the prior may have an arbitrary structure. Several techniques have been proposed to handle models with arbitrary structure (Supplementary Note 4.6.2). However, none of these techniques are optimal for expressing a pairwise prior.

Therefore, we instead employ a novel strategy based on *posterior regularization* [Ganchev et al., 2010] to integrate this prior. This is done by introducing an auxiliary joint distribution $q(X_H)$, placing a regularizer on $q(X_H)$, and encouraging q to be similar to p_θ through a KL divergence penalty. The regularizer is

$$\mathcal{R}_{\text{PR}}(\theta) \triangleq \max_q \mathcal{R}'_{\text{PR}}(\theta, q) \tag{4.4}$$

$$\mathcal{R}'_{\text{PR}}(\theta, q) \triangleq -D(q(X_H) \| p_\theta(X_H | \bar{x}_O)) + \mathcal{PR}(q), \tag{4.5}$$

where $D(\cdot\|\cdot)$ is the KL divergence $D(p(X_H)\|q(X_H)) = \sum_{x_H} p(x_H) \log(p(x_H)/q(x_H))$ and $\mathcal{PR}(q)$ is a posterior regularizer that expresses prior knowledge about the posterior distribution. KL divergence measures the dissimilarity of probability distributions, such that $D(p\|q)$ is zero if the distributions are identical and can be arbitrarily large if they are not. Several posterior regularizers have been proposed in the past, such as those that require posteriors to satisfy constraints in expectation [Ganchev et al., 2010].

We propose a new type of posterior regularizer that expresses a pairwise prior (Figure 4.8). We are given a weighted, undirected regularization graph over the hidden variables $G_R = (H, E_R)$, where $E_R \subseteq H \times H$ is a set of edges with non-negative similarity weights $w : E_R \to \mathbb{R}_+$, such

that a large $w(u, v)$ indicates that we have strong belief that X_u and X_v should be similar. (We describe how we generate this graph in the next two sections.) For a distribution $p(X_H)$, let $p_h^M(X_h)$ indicate the marginal distribution over X_h, $p_h^M(x_h) = \sum_{x_{H \setminus h}} p(x_H)$. Let λ_G be a hyperparameter controlling the strength of regularization. The posterior regularizer is

$$\mathcal{PR}_{\text{GBR}}(q) \triangleq -\lambda_G \sum_{(u,v) \in E_{\text{R}}} w(u, v) D(q_u^M(X_u) \| q_v^M(X_v)). \tag{4.6}$$

Thus the full objective is

$$\text{maximize}_{\theta, q} \quad J_{\text{GBR}}(\theta) \triangleq \mathcal{L}(\theta) - D(q(X_H) \| p_\theta(X_H | \bar{x}_O)) - \lambda_G \sum_{(u,v) \in E_{\text{R}}} w(u, v) D(q_u^M(X_u) \| q_v^M(X_v)). \tag{4.7}$$

We term this strategy of adding graph-based penalties *graph-based regularization* (GBR). See Chapter 3 for more detail.

4.4.5 GBR optimization

We have developed a novel algorithm for efficiently optimizing J_{GBR} in q. This algorithm alternates between using a method for probabilistic inference such as the forward-backward algorithm and applying a message passing algorithm over the regularization graph G_R. In the inference step, the model receives evidence from the message passing step in the form of a "virtual evidence" distribution $r_h^M(X_h)$ over each variable h. These virtual evidence distributions are used in conjunction with the original SAGA model to compute a posterior distribution over the labels using any algorithm for probabilistic inference on dynamic Bayesian networks, such as belief propagation or the forward-backward algorithm.

In the message passing step, the algorithm updates r^M to minimize the Kullback-Leibler penalties in the objective function J_{GBR}. This message passing step is itself performed using an alternating optimization algorithm, which passes messages over the regularization graph G_R. This algorithm is similar to one originally developed for the field of semi-supervised learning

[Subramanya and Bilmes, 2011].

The inference and message passing steps are iterated until convergence. These two updates are linear in the number of variables (for chain-structured models, which include all existing SAGA methods) and linear in the degree of the regularization graph, respectively. The algorithm exhibits monotonic convergence, similar to the EM algorithm. We derive the algorithm for optimizing J_{GBR} and prove its convergence in Chapter 3.

4.4.6 GBR graph for incorporating Hi-C data

When we are using GBR to incorporate Hi-C data, we are given a matrix of contact p-values $P \in \mathbb{R}^{n \times n}$, generated from a matrix of contact counts as described above. To remove noise and decrease the degree of the graph, we removed all contacts with uncorrected p-value $p > 10^{-6}$ and multiplied the remaining p-values by 10^6, similar to a Bonferroni correction. Note that due to the large number of hypotheses, performing a full Bonferroni correction would result in very few contacts. Moreover, the graph weights allow the algorithm to take into account the strength of each connection, so the choice of 10^6 was made for computational, not statistical, reasons. We computed the weights as

$$w(i,j) \triangleq \max(0, -\log_e(p(i,j)/10^6)). \tag{4.8}$$

As with the graph for transferring information between cell types, the multiplicative scale of the weights is arbitrary, since it is controlled by the graph weight hyperparameter λ_G. We used only intrachromosomal contacts for forming the GBR graph. To produce a GBR graph representing cell type-consistent chromatin conformation used in the domain annotation of eight cell types, we added the edge weights from the IMR90 and H1-hESC Hi-C GBR graphs.

4.4.7 GBR graph for annotation of multiple cell types

When we are using GBR to transfer information about cell type A to improve annotation of cell type B, we are given an annotation $a^A_{1:n} \in \{1 \dots k\}^n$ of cell type A, produced without GBR. We construct a GBR graph from this annotation by connecting each pair of positions that received the

same label in a^A with an undirected edge of weight 1. Note that the weight is arbitrary, since it is scaled by the regularization parameter λ_G. To mitigate the problem of quadratic growth in the degree of this graph, we randomly subsampled this graph such that each node had outgoing degree $17 \approx \log_e(n)$. We chose this graph degree because a randomly-subsampled graph with $n \log_e n$ edges has the same connected components as the full graph with high likelihood [Erdős and Rényi, 1960], and our experiments on synthetic data (not shown) showed that the sparse graph performed similarly to a complete graph.

4.4.8 Circular permutation

As a null model for several experiments, we performed a circular permutation of the genome along each chromosome arm as follows. We randomly choose a translation fraction $\theta \in [0, 1]$. For each coordinate $i \in \{1 \dots n\}$ within a chromosome arm that spans the range $[a, b)$, we translate i to $t(i)$, where

$$t(i) = \text{mod}_{(b-a)}(i + \lfloor \theta(b - a) \rfloor) + a. \tag{4.9}$$

To circularly permute a genome feature, such as an annotation or a Hi-C contact map, we translate each element from position i to $t(i)$. Thus, when a circularly-permuted feature is compared to an un-permuted feature, all positional correspondence between permuted and un-permuted features are removed, but each feature's spatial patterns are preserved. In each case, we performed this permutation 200 times and report the average over all permutations. If the feature includes any centromere- or telomere-defined elements, we remove these as a preprocessing step.

4.4.9 Topological domain agreement

To evaluate the degree to which an annotation A matches a set of topological domains, we computed the number $c_{d,\ell}$ of bases by which domain d is covered by label ℓ. We then computed $c^*_d = \max_\ell c_{d,\ell}$ to be the number of bases covered by the highest-coverage label for domain d, and divided c^*_d by the length of d to produce f^*_d, the fraction of d covered by its plurality label. The agreement f^*_d takes its maximum value of 1 if the domain d is covered by exactly one annotation label. We computed the

raw genome-wide agreement $f_{\text{raw}} = (1/|d|) \sum_d f_d^*$.

This raw genome-wide agreement f_{raw} can be improved simply by increasing the length of segments and decreasing the number of labels. Therefore, we circularly permuted A to form A^p, and used this permuted annotation to compute f_{raw}^p. Finally, we computed the *topological domain agreement* $a = f_{\text{raw}}/f_{\text{raw}}^p$ as the ratio of unpermuted and permuted raw agreements. This normalized agreement is large when the annotation has small segments that exactly match the topological domains and is small when the annotation's segments are not correlated with topological domains.

4.4.10 Signal variance explained

To evaluate the similarity between a genome-wide signal vector and a genome annotation, we use the following measure, which we term the *variance explained*. We are given a genome annotation with k labels $a_{1:n} \in \{1 \dots k\}^n$ and a vector $x_{1:n} \in \mathbb{R}^n$. We compute the signal mean over the positions assigned a given label ℓ as

$$\mu_\ell \triangleq \frac{\sum_{i=1}^n \mathbf{1}(a_i = \ell) x_i}{\sum_{i=1}^n \mathbf{1}(a_i = \ell)} \quad \text{for } \ell \in \{1 \dots k\}. \tag{4.10}$$

We define a predicted signal vector $x_i^p = \mu_{a_i}$ and compute the the prediction error as $d_i = x_i - x_i^p$. We compute the residual standard deviation of the signal vector as

$$s \triangleq \text{stdev}(d_{1:n}) = \sqrt{\frac{1}{n} \sum_{i=1}^n (d_i - \text{mean}(d_{1:n}))^2} = \sqrt{\frac{1}{n} \sum_{i=1}^n d_i^2} \tag{4.11}$$

The last equality holds because $\text{mean}(d_{1:n}) = 0$ by construction. We define the *variance explained* (VE) for annotation a and signal vector x as $\text{VE} \triangleq \text{stdev}(d_{1:n}) - \text{stdev}(x_{1:n})$. VE is bounded by the range $[0, \text{stdev}(x_{1:n})]$. VE is a measure of the extent to which a genome-wide signal data set and annotation are similar, where higher values indicate better agreement.

4.4.11 Genomic element prediction

We form a classifier for a set of genomic elements based on an annotation using the following strategy. We are given a genome annotation with k labels $a_{1:n} \in \{1 \dots k\}^n$ and a set of positions $S \subseteq \{1 \dots n\}$ which represent some set of elements of interest, such as enhancers or CTCF binding sites. Define $A_\ell = \{i \mid a_i = \ell\}$ to be the positions annotated by label ℓ To avoid biases caused by differing-size elements, we assume that each element occupies just 1 bp. In the case of larger elements (such as MACS-called TF binding sites, which are $\sim$200 bp), we define each element as the middle base pair of the range.

For each label, we compute the predictive precision of label ℓ as

$$\text{precision}(\ell) = \frac{|S \cap A_\ell|}{|A_\ell|} \quad \text{for } \ell \in \{1 \dots k\}. \tag{4.12}$$

We rank the labels in decreasing order of their precision on a training set to get an order $s_{1:k} \in \{1 \dots k\}^k$.

Using this ordering, we form k predictors, $P_j = \bigcup_{i=1}^{j} A_{si}$ for $j \in \{1 \dots k\}$. The true positives and false positives of a predictor P are $\text{TP}(P) = P \cap S$ and $\text{FP}(P) = P \cap (\{1 \dots n\} \setminus S)$ respectively. The predictors are in order of decreasing stringency—that is, $P_{j-1} \subseteq P_j$.

We can trace out the full sensitivity-specificity tradeoff (such as for an ROC or PR curve), by interpolating between each successive pair of predictors. To interpolate between a pair of predictors P_j and P_{j+1}, we form an interpolated predictor $P_{j,j+1,\theta}$ by sampling each position $i \in P_j \setminus P_{j-1}$ with probability $\theta \in [0, 1]$. The expected number of true positives and false positives of an interpolated predictor $P_{j,j+1,\theta}$ can be shown to be

$$E[|\text{TP}(P_{j,j+1,\theta})|] = |\text{TP}(P_j)| + \theta|\text{TP}(A_{j+1})| \quad \text{and} \tag{4.13}$$

$$E[|\text{FP}(P_{j,j+1,\theta})|] = |\text{FP}(P_j)| + \theta|\text{FP}(A_{j+1})|, \tag{4.14}$$

respectively. We report our performance using a test set disjoint from the training set used to order the labels.

4.4.12 Developmental replication domains

In order to evaluate the replication timing dynamics of different types of domains, we used a four-label (constitutive early/late, switching early/late) annotation of the human genome using published replication timing data for 16 different human cell types, gathered by V Dileep, F Ay, J Sima, WS Noble, DM Gilbert et al. (unpublished). This annotation first windowed replication timing data into 40 kb bins and then determined for each window whether it replicates early (RT value $>$ 0.5) or late (RT value $<$ -0.2) in all cell types. Such windows with consistent timing profiles across all cell types were labeled as "constitutively early" and "constitutively late", respectively. The remaining windows were labeled as either switching or left unlabeled. Switching windows are determined as those with an absolute value of replication timing larger than 0.5 in all cell types but with an opposite sign than others in at least one cell type. Switching windows that are early and late replicating in IMR90 were labeled as switching early and switching late, respectively.

4.4.13 Consistent domain boundaries

When we annotated domains in eight cell types, we found that domain boundaries were shared between annotations much more often than would be expected by chance. To identify developmentally-consistent domain boundaries, we first formed a list of all segment boundaries that occurred in at least one cell type. For each boundary, we computed the number of cell types with boundaries within 50 kb. We formed a set of representative by greedily selecting the boundary with the most nearby boundaries as a representative, removing all boundaries near the representative from the list, and repeating the process until no two boundaries in the list were within 50 kb of one another. While this problem is an instance of the NP-hard set cover problem, the greedy approach is guaranteed to result in a constant-factor approximation of optimal [Nemhauser et al., 1978a]. This yielded a set of 13,906 boundary groups, each more than 50 kb from all other groups. We defined the 2,967 boundary groups composed of at least five boundaries as consistent boundaries.

4.5 Data access

Domain annotations and code for Segway with GBR is available as Supplemental Material and online at http://noble.gs.washington.edu/proj/gbr.

4.6 Supplementary Notes

4.6.1 Review of existing SAGA methods for using data from multiple cell types

Existing methods for semi-automated genome annotation work well on data from a single cell type, but annotating multiple cell types remains an active area of research. There are three simple strategies for performing annotation of multiple cell types. First, the simplest strategy is to apply the same model to both genomes (sometimes called "concatenated" annotation) [Sheffield et al., 2013], but this requires that all cell types have the same set of available data, which is not generally true. Moreover, in practice, experimental artifacts lead to poor performance for models which model multiple data from multiple experiments with the same parameters, exhibiting effects such as assigning separate sets of labels to each cell type in the model. Second, one could perform annotation separately on each cell type and find a mapping between the labels (for example, by using the Hungarian algorithm [Kuhn, 1955]). However, since different cell types generally have different types of activity and different sets of signal data sets, such a mapping is generally very poor. Third, one could use all data from all cell types in one model (sometimes called "stacked" annotation), but this strategy must either give the same label to each position for every cell type or use a separate label for each pattern of labels across cell types, which requires an exponentially-large number of labels.

Two additional methods have been proposed to annotate multiple cell types. The first, called hiHMM ("hierarchically-linked infinite HMM") maintains a separate model for each cell type and uses a regularization penalty to encourage the models to have similar parameters [Ho et al., 2013]. This addresses the problem of requiring the same set of data across cell types, but does not share any position-specific information between cell types. The second method for performing multi-cell-line annotation, called TreeHMM, is given a tree over cell types and models the transition between

labels between neighboring positions and also between neighboring developmental states [Biesinger et al., 2013]. This model can integrate position-specific information between cell types, but requires that each cell type has the same available data and is sensitive to any cross-experiment artifacts. Moreover, the complexity of this model forced the authors to resort to approximate methods for inference, which likely decreases the quality of the resulting annotations.

These two problems—requiring a common set of data and failure to integrate evidence—are especially important because, although there are virtually limitless cell types and cell states that one would like to understand, very limited numbers of experiments have been performed in most of these cell types due to the cost of genomic experiments. For example, ENCODE has performed 335 experiments in its most-studied cell type, but has performed just 2-10 experiments in more than 100 cell types.

Transferring information with GBR removes the requirement for a common set of data across cell types and does integrate position-specific evidence across cell types. Therefore, GBR provides a method leveraging all available data in order to produce high-quality annotations of each cell type.

4.6.2 Related optimization methods

Clearly, the most straightforward way to express pairwise interactions in a graphical model is to encode them in the underlying graph and to use approximate methods (reviewed in [Wainwright and Jordan, 2008]) to enable inference. This form of interaction is quite general, in that when one adds a factor $\phi(y_i, y_j)$ between two random variables Y_i and Y_j, these random variables may have any type of interaction, expressed by $\phi(y_i, y_j)$. GBR, on the other hand, asks only for similarity between the marginals, meaning that $p(y_i|\cdot)$ and $p(y_j|\cdot)$ should be similar. Alternatively, a factor could encode such similarity, for example if $\phi(y_i, y_j) = \lambda \mathbf{1}(y_i = y_j)$. Such factors added to an HMM or CRF would result in a high treewidth model that can be dealt with using approximate inference. Doing so, however, loses any guarantee of optimality (which we preserve with GBR).

The posterior regularization framework of Ganchev et al. [Ganchev et al., 2010] takes an approach similar to ours, augmenting a simple model in a way that maintains tractable inference. This method adds a regularization term to an EM objective in order to require the posterior probabilities

to satisfy logical constraints in expectation. Ganchev et al. show how to optimize this combined objective efficiently when the regularization term is linear in the posterior distribution of the model. Unfortunately, pairwise similarity relationships cannot be expressed with such a linear regularization term.

The most similar work to ours are the following three methods for graph regularization. First, Altun el al. [Altun et al., 2005] describe a graph regularization applied to a max-margin model applied to pitch-accent prediction and optical character recognition. However, this method involves a matrix inversion step, and thus cannot scale to large models. Second, Subramanya et al. [Subramanya et al., 2010] combine a temporal CRF with a regularizer that expresses pairwise squared-error penalties derived from unlabeled data. They apply this method to the part-of-speech tagging task [Subramanya et al., 2010] and later to related problems in natural language [Das and Petrov, 2011, Das and Smith, 2011]. That work, however, resorts to a purely heuristic update step, and lacks any optimality guarantees. Third, He et al. [He et al., 2013] present an approach based on an exponentiated gradient descent algorithm. Like our approach, He's approach exhibits monotone convergence. Although He's work has many similarities with our approach, our methods were developed independently, and He's work differs from ours in three important ways. First, He et al. [He et al., 2013] use an exponentiated gradient descent strategy, while we use alternating minimization. Second, He's method uses a squared-error penalty, which is inappropriate for probability distributions, unlike our use of the Kullback-Leibler divergence [Bishop, 1995, p. 226]. Third, the exponentiated gradient descent method is applied to handwriting recognition and part-of-speech tagging, while we apply GBR to genome annotation.

4.7 Supplementary Figures/Tables

Data type	URL
CAGE	http://www.gencodegenes.org/releases/7.html
ENCODE (ChIP-seq, DNase, Replication timing)	http://hgdownload.cse.ucsc.edu/goldenPath/hg19/encodeDCC/
ChIP-seq (Roadmap)	http://www.roadmapepigenomics.org/data
Hi-C	http://yuelab.org/hi-c/download.html
Topological domains	http://www.cs.cmu.edu/∼ckingsf/software/armatus/

Supplementary Table 4.1: Data sources

	IMR90 domain annotation	Eight-cell type domain annotation	GM12878 reduced annotation
Input data sets	DNase H2aK5ac H2aK9ac H2a.Z H2bK120ac H2bK12ac H2bK15ac H2bK20ac H2bK5ac H3K14ac H3K18ac H3K23ac H3K27ac H3K27me3 H3K36me3 H3K4ac H3K4me1 H3K4me2 H3K4me3 H3K56ac H3K79me1 H3K79me2 H3K9ac H3K9me1 H3K9me3 H4K20me1 H4K5ac H4K8ac H4K91ac Repli-seq	DNase H2a.Z H3K27ac H3K27me3 H3K36me3 H3K4me1 H3K4me2 H3K4me3 H3K79me2 H3K9ac H3K9me3 H4K20me1	H3K4me1 H3K4me2 H3K3me3 H3K9ac H3K27ac H3K27me3 H3K36me3 H4K20me1
Number of labels	8	8	25
Transition weight ($\lambda_{\text{transition}}$)	48	12	1
Number of random EM initializations	10	10	10
GBR graph scale (λ_G)	1	1	1
GBR optimization hyperparameter (λ_{R1})	1	1	10

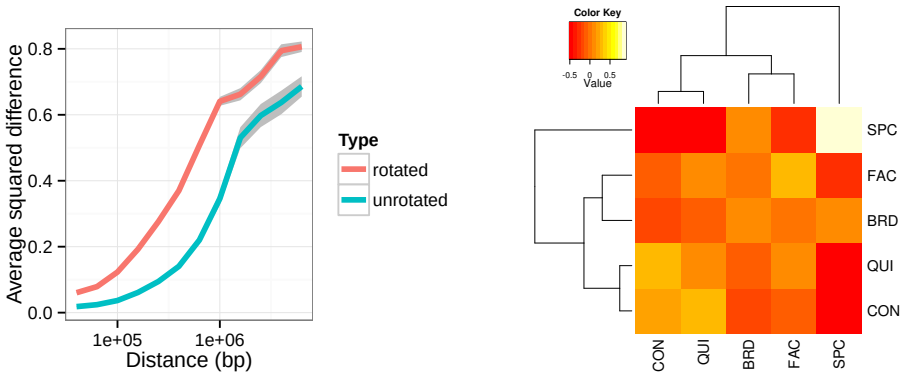
Supplementary Table 4.2: Parameters of all genome annotations

GOID	Bonferroni-corrected p-value	Name
GO:0008150	5.44809636039661e-49	biological process
GO:0070647	5.3669047773656 8e-14	protein modification by small protein conjugation or removal
GO:0016567	2.53980288482968e-13	protein ubiquitination
GO:0032446	3.48958938873894e-13	protein modification by small protein conjugation
GO:0071840	1.2816846277191e-07	cellular component organization or biogenesis
GO:0048522	3.15663523300463e-07	positive regulation of cellular process
GO:0016043	1.55013794212494e-06	cellular component organization
GO:0050953	2.07559208909527e-06	sensory perception of light stimulus
GO:0007601	2.79978686922173e-06	visual perception
GO:0006996	2.18396515274592e-05	organelle organization
GO:0016071	7.40708736771575e-05	mRNA metabolic process
GO:0048519	0.000123587368672724	negative regulation of biological process
GO:0023056	0.000191430490530164	positive regulation of signaling
GO:0048518	0.000239550248137195	positive regulation of biological process
GO:0032270	0.000250487526217915	positive regulation of cellular protein metabolic process
GO:0018146	0.000336577942863192	keratan sulfate biosynthetic process
GO:0007005	0.000402299808664309	mitochondrion organization
GO:0010647	0.000414643467676553	positive regulation of cell communication
GO:0032268	0.000530771233104358	regulation of cellular protein metabolic process
GO:0043928	0.000623789428174407	exonucleolytic nuclear-transcribed mRNA catabolic process involved in deadenylation-dependent decay
GO:0000288	0.000670978259119087	nuclear-transcribed mRNA catabolic process, deadenylation-dependent decay
GO:1903320	0.00117566239446111	regulation of protein modification by small protein conjugation or removal
GO:0009967	0.00131477493050438	positive regulation of signal transduction
GO:1902533	0.00136644459633516	positive regulation of intracellular signal transduction
GO:0048523	0.00148158842395948	negative regulation of cellular process
GO:0051340	0.00150521495507962	regulation of ligase activity
GO:0000291	0.00193533449146964	nuclear-transcribed mRNA catabolic process, exonucleolytic
GO:0031401	0.00194625332376889	positive regulation of protein modification process
GO:1903047	0.00215993757988771	mitotic cell cycle process
GO:0042531	0.00237481842965934	positive regulation of tyrosine phosphorylation of STAT protein
GO:0007267	0.0039286364798486	cell-cell signaling
GO:0006401	0.00399526165008678	RNA catabolic process
GO:0031396	0.00443702013802013	regulation of protein ubiquitination
GO:0000278	0.00470584872871359	mitotic cell cycle
GO:0051351	0.00495154943903681	positive regulation of ligase activity
GO:0051438	0.00524041402868326	regulation of ubiquitin-protein transferase activity
GO:0042339	0.0053487952592411	keratan sulfate metabolic process
GO:0051247	0.0056999635780996	positive regulation of protein metabolic process
GO:0016265	0.00629144608895318	death
GO:0046427	0.00671456368410498	positive regulation of JAK-STAT cascade
GO:0008219	0.0067422659494943	cell death
GO:0000956	0.00698342075165014	nuclear-transcribed mRNA catabolic process
GO:0031399	0.00786100228743956	regulation of protein modification process
GO:0044770	0.00795426575038111	cell cycle phase transition
GO:0051246	0.00937868333257509	regulation of protein metabolic process

Supplementary Table 4.3: GO terms enriched for genes in BRD domains

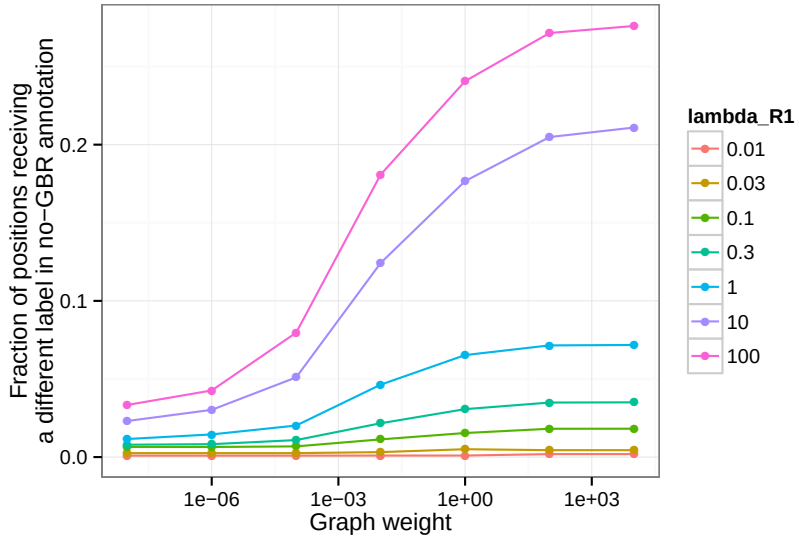
GOID	Bonferroni-corrected p-value	Name
GO:0001944	2.9592753850599e-11	vasculature development
GO:0001568	1.3492859759456 5e-09	blood vessel development
GO:0072358	1.8778596344277e-09	cardiovascular system development
GO:0072359	1.94865728031881e-09	circulatory system development
GO:0048598	2.33761516882996e-09	embryonic morphogenesis
GO:0032501	4.2950441880 5107e-09	multicellular organismal process
GO:0044707	6.45108090271442e-09	single-multicellular organism process
GO:0007275	9.54873387980616e-09	multicellular organismal development
GO:0032502	9.87403900226897e-09	developmental process
GO:0009653	1.13900576119261e-08	anatomical structure morphogenesis
GO:0048646	1.2756082632202e-08	anatomical structure formation involved in morphogenesis
GO:0048856	3.4402782464997 4e-08	anatomical structure development
GO:0048514	3.64379056813118e-08	blood vessel morphogenesis
GO:0044767	4.96188838214423e-08	single-organism developmental process
GO:0008150	1.09920334986419e-07	biological process
GO:0009888	1.1506207019 6739e-07	tissue development
GO:0048731	1.3400575699 2352e-07	system development
GO:0030198	2.5474465774 0361e-07	extracellular matrix organization
GO:0043062	2.7000849764 8116e-07	extracellular structure organization
GO:0040011	4.0968165755 1741e-07	locomotion
GO:0048523	6.0092473832 0409e-07	negative regulation of cellular process
GO:0048513	1.2755568481 7328e-06	organ development
GO:0009887	1.6722827183 8845e-06	organ morphogenesis
GO:0060429	1.7524848184 9761e-06	epithelium development
GO:0009605	2.1238836829 1444e-06	response to external stimulus
GO:0048519	2.7081271739 4184e-06	negative regulation of biological process
GO:0048869	3.9963976920 8689e-06	cellular developmental process
GO:0009790	4.814208141 0924e-06	embryo development
GO:0030154	8.1779942778 7657e-06	cell differentiation
GO:0048522	8.7903700402 3573e-06	positive regulation of cellular process
GO:0048870	9.7074854019 4382e-06	cell motility
GO:0051674	9.7074854019 4382e-06	localization of cell
GO:0048518	9.876252669 0946e-06	positive regulation of biological process
GO:1902533	1.7504728625 9699e-05	positive regulation of intracellular signal transduction
GO:0071840	2.3729929264 6679e-05	cellular component organization or biogenesis
GO:0030334	2.3884441143 6045e-05	regulation of cell migration
GO:0007389	3.4105764538 1657e-05	pattern specification process
GO:0051270	4.1579377202 4666e-05	regulation of cellular component movement
GO:2000145	7.2070085837 0221e-05	regulation of cell motility
GO:0040012	7.239813615 9996e-05	regulation of locomotion
GO:0016043	7.5777818241 9685e-05	cellular component organization
GO:0001501	9.1653146377 7006e-05	skeletal system development
GO:0008219	0.000103725100284498	cell death
GO:0001525	0.0001112117619674	angiogenesis
GO:0009966	0.000112414145595174	regulation of signal transduction
GO:0016265	0.000115956206339428	death
GO:0009967	0.000119265054523972	positive regulation of signal transduction
GO:0016477	0.000141195508435494	cell migration
GO:0048568	0.000148183903115669	embryonic organ development
GO:0001503	0.000148474000627697	ossification
GO:0006915	0.000174710834263385	apoptotic process
GO:0048729	0.000185804853094102	tissue morphogenesis
GO:0012501	0.000193299675551627	programmed cell death
GO:0010647	0.000264669391358318	positive regulation of cell communication
GO:0003007	0.000308615021011459	heart morphogenesis
GO:0010628	0.000312006014076936	positive regulation of gene expression
GO:0023056	0.000458831197316147	positive regulation of signaling
GO:0051239	0.000694243253157008	regulation of multicellular organismal process
GO:0031325	0.000823302329284296	positive regulation of cellular metabolic process
GO:0002009	0.000843967242091738	morphogenesis of an epithelium
GO:0048863	0.000980998073137653	stem cell differentiation
GO:0042325	0.00125849445147922	regulation of phosphorylation
GO:1902531	0.00144207328720399	regulation of intracellular signal transduction
GO:0044236	0.00151180971326667	multicellular organismal metabolic process
GO:0009893	0.00161958399151636	positive regulation of metabolic process
GO:0022603	0.00169126335423007	regulation of anatomical structure morphogenesis
GO:0035295	0.00223230684414793	tube development
GO:0006928	0.00230989986294592	cellular component movement
GO:0010604	0.00236857824547227	positive regulation of macromolecule metabolic process
GO:0010646	0.00252308440283661	regulation of cell communication
GO:0050793	0.002684781718963 7	regulation of developmental process
GO:0048562	0.00285877724714378	embryonic organ morphogenesis
GO:0032879	0.00295955595643228	regulation of localization
GO:0080134	0.00335551717091664	regulation of response to stress
GO:0048705	0.00370439844279299	skeletal system morphogenesis
GO:0048864	0.00389984096973906	stem cell development
GO:0023051	0.004015877459873 1	regulation of signaling
GO:0006334	0.00406223628626311	nucleosome assembly
GO:0045893	0.00414091776655817	positive regulation of transcription, DNA-templated
GO:0007167	0.00519580828357705	enzyme linked receptor protein signaling pathway
GO:0003002	0.00540223999981726	regionalization
GO:0051254	0.00549153180374605	positive regulation of RNA metabolic process
GO:1902680	0.00598393700399004	positive regulation of RNA biosynthetic process
GO:0023014	0.00676138544251521	signal transduction by phosphorylation
GO:0008283	0.007031041895687 8	cell proliferation
GO:0048583	0.007209600804982 2	regulation of response to stimulus
GO:0006333	0.00732272231577271	chromatin assembly or disassembly
GO:0007507	0.00736076227709712	heart development
GO:0042127	0.00771299785801664	regulation of cell proliferation
GO:2000026	0.008432344878453	regulation of multicellular organismal development
GO:0010941	0.00916678907171083	regulation of cell death
GO:0043408	0.00935020093242137	regulation of MAPK cascade
GO:0044259	0.00998773914798186	multicellular organismal macromolecule metabolic process

Supplementary Table 4.4: GO terms enriched for genes in SPC domains.

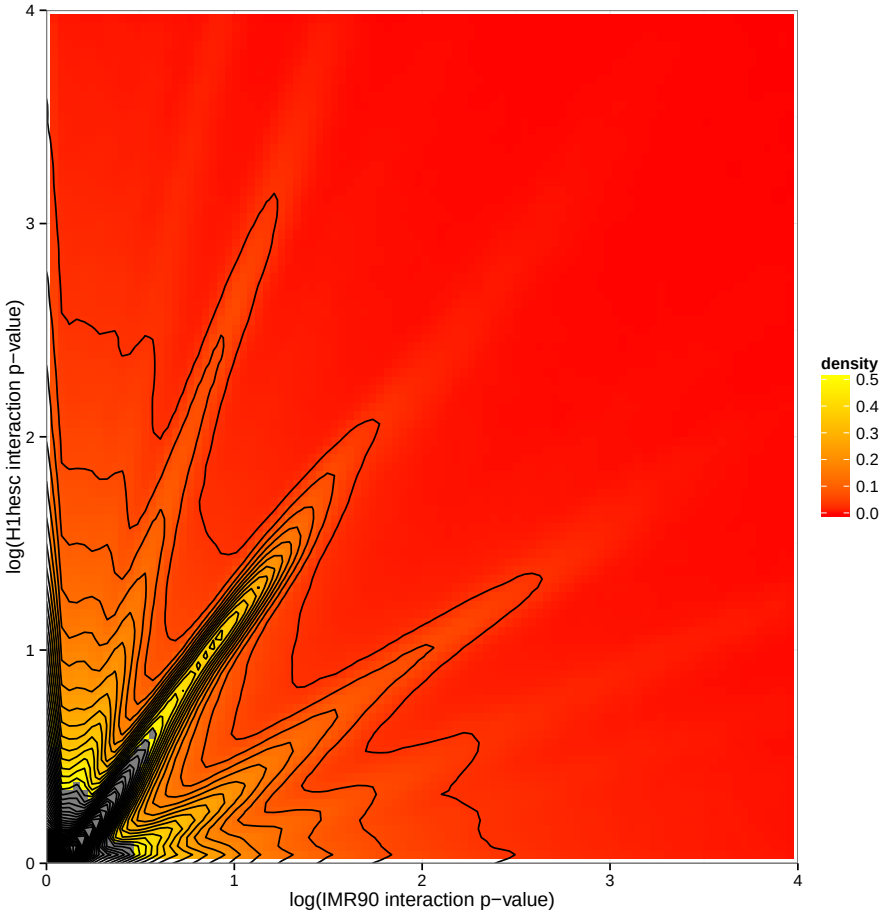


A B

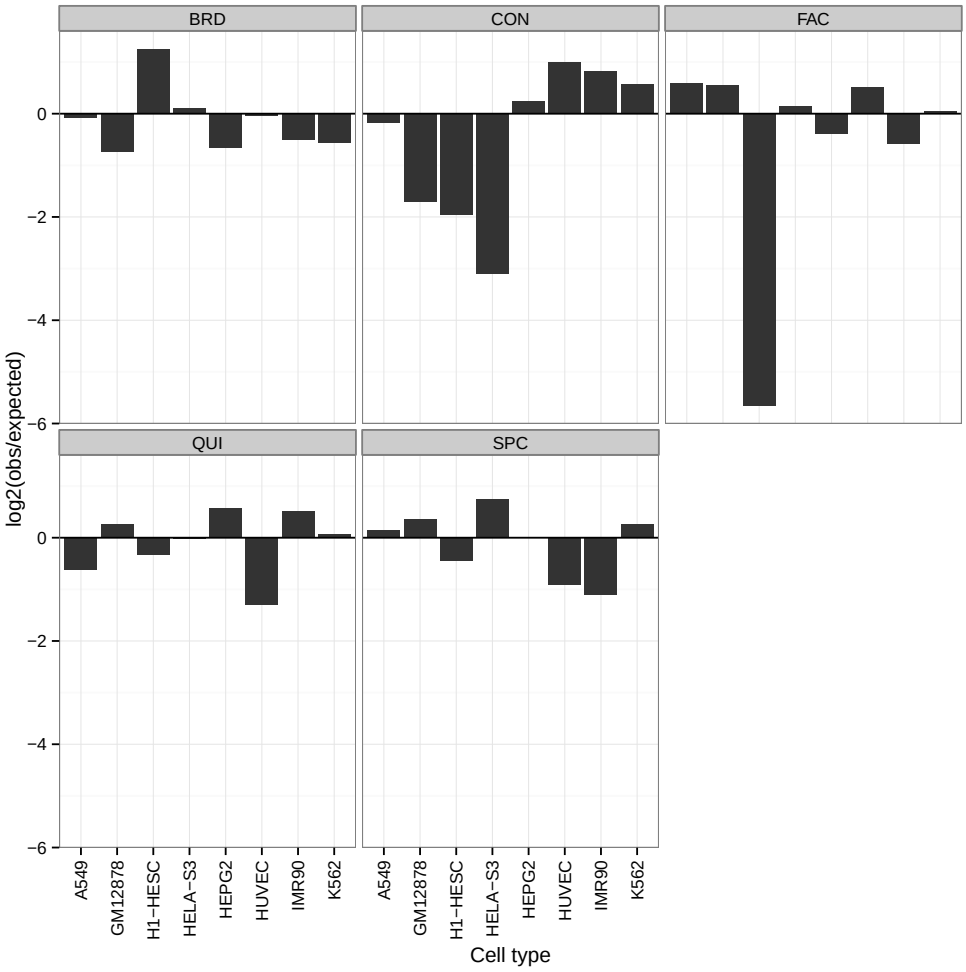
Supplementary Figure 4.1: (A) Average squared difference between replication timing values at left and right sides of significant contacts, as a function of genomic distance, relative to a permutation control (t-test 95% confidence interval grey error regions). (B) Confusion matrix of Segway annotation labels at left and right sides of significant contacts (without GBR). Color depicts $\log_2$(obs/expected) relative to a permutation control (Methods). Pairs of annotation labels at significantly interacting positions match more often than expected by chance (binomial test $p < 10^{-16}$).



Supplementary Figure 4.2: Fraction of annotation changed by GBR. Y axis depicts the fraction of positions that receive a different label between an annotation without GBR and an annotation with GBR using a certain set of hyperparameters. X axis and color depict the λ_G and λ_{R1} hyperparameters respectively.



Supplementary Figure 4.3: Correlation of Hi-C contact strength between IMR90 and H1-hESC. X and Y axes are log p-values of association of a given pair of positions. Color indicates density of points. Black lines indicate density contours in 0.1% bins. Spearman $r = 0.57$.

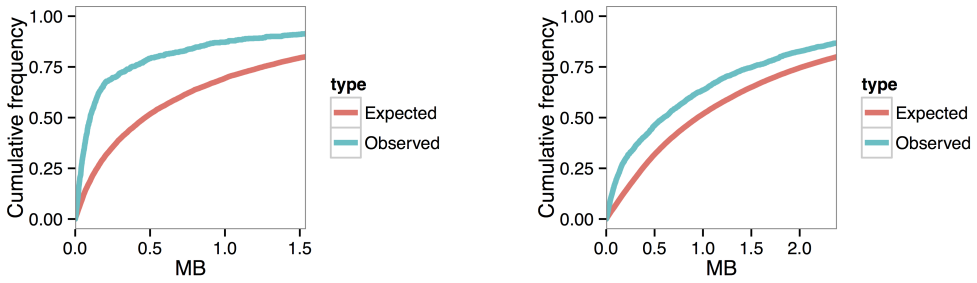


Supplementary Figure 4.4: Distribution of domain labels across eight cell types. Y axis indicates log2(bases covered by label ℓ in celltype A / (bases covered by label ℓ / 8)).

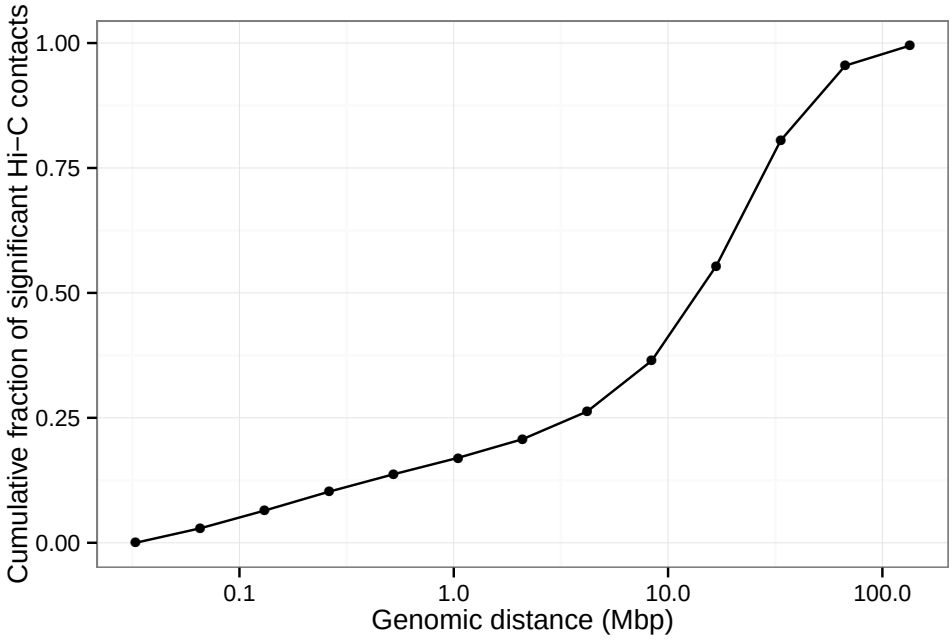
A

B

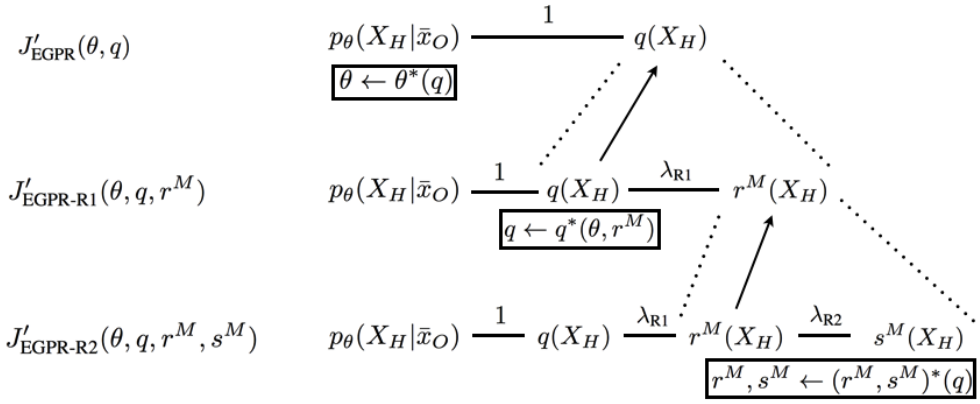
Supplementary Figure 4.5: Visualization of GO term enrichment for genes in IMR90 (A) BRD domains and (B) SPC domains using REVIGO [Supek et al., 2011]. Each bubble represents a cluster of related enriched GO terms. X and Y axes are projected semantic axes defined using multidimensional scaling on the semantic similarity of each pair of terms.



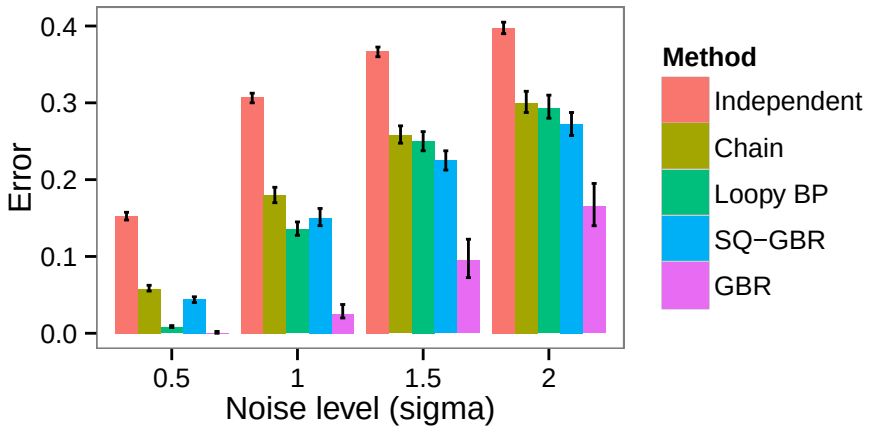
Supplementary Figure 4.6: Enrichment of consistent Segway boundaries for consistent replication domain boundaries. (A) Fraction of consistent replication domain boundaries overlapping consistent Segway domain boundaries as a function of the overlap distance. (B) Same as (A), but fraction of Segway domain boundaries. We used replication domain boundaries called by Pope et al. [2014]. We defined replication boundaries occurring in more than 10 out of 18 cell types as consistent.



Supplementary Figure 4.7: Genomic distance distribution of significant IMR90 Hi-C contacts ($q < 0.05$).



Supplementary Figure 4.8: Schematic of the three formulations of the objective and the alternating maximization strategy. Edges in this figure indicate KL terms, labeled according to their weight in the objective. Boxed formulae are update steps. We perform two reformulations, first splitting q into q and r^M linked by a KL term of weight λ_{R1}, then splitting r^M into r^M and s^M, linked by a KL term of weight λ_{R2}.



Supplementary Figure 4.9: Comparison between inference methods on synthetic data. The X axis shows σ, a hyperparameter controlling the difficulty of inference. The Y axis shows the average accuracy over 200 simulations of MAP inference on the model in question (95% Wilcoxon test confidence intervals). Bars correspond to five different inference methods: 1) inference on each position independently, with no chain model (Independent); 2) inference on the chain alone, without using W (Chain); 3) loopy belief propagation on the chain plus extra factors of $\Pr(X_i = X_j) = \text{sigmoid}(\lambda w_{ij})$, where λ controls the strength of these factors; (Loopy BP) 4) a variant of GBR using the regularization graph W and a squared-error penalties as described in [He et al., 2013] (SQ-GBR); and 5) GBR using the regularization graph W (our method, GBR).

Chapter 5

CHOOSING PANELS OF GENOMICS ASSAYS USING SUBMODULAR OPTIMIZATION

Abstract

Due to the high cost of sequencing-based genomics assays such as ChIP-seq and DNase-seq, a panel of at most 3–10 assays is usually performed on each cell type. We present *submodular selection of assays* (SSA), a method for choosing a diverse panel of genomic assays that leverages methods from the field of *submodular optimization*. SSA performs better than alternative strategies in practice, is computationally efficient and extremely flexible, and is theoretically optimal under certain assumptions. This application may also serve as a model for how submodular optimization can be applied to other discrete problems in biology. SSA is available at `http://github.com/melodi-lab/Submodular-Selection-of-Assays`.

5.1 Background

Genomics assays such as ChIP-seq, DNase-seq and RNA-seq can measure a wide variety of types of DNA activity, but the cost of these assays limits their application. In principle, to fully characterize a cell type, one would like to perform every possible type of assay, including ChIP-seq for a variety of histone modifications and dozens of transcription factors, several chromatin accessibility assays, and RNA-seq characterizing various types of RNA molecules. However, at current sequencing prices, performing a single genomics assay with reasonable sequencing depth costs on the order of $400 [scienceexchange]. As a point of comparison, consider the ENCODE and Roadmap Epigenomics consortia, which develop, perform and analyze genomics assays as their primary activity [Bernstein et al., 2010, ENCODE Project Consortium, 2012]. As of January 2015, the two consortia had performed a total of 216 types of assays on at least one cell type, and at least one assay on a total of

228 cell types (Methods). Applying all these assay types to all these cell types would require 49,248 assays; however, the two consortia have performed just 1,359 assays, 5% of the possible number (encodedcc.org; 2014). These consortia are worldwide efforts with large budgets; a typical lab might be able to perform at most several assays per cell type in order to analyze a particular tissue or perturbation that they are interested in. Moreover, there are virtually limitless perturbations and variations of a given cell type for which it would be interesting to examine the effect on DNA activity, including drug treatments, age, differentiation, etc.

Consequently, selecting a small panel of assays to perform on each cell type of interest—a problem we call *assay panel selection*—is a key step in any genomics project. To our knowledge, there has been little discussion in the literature of how to choose such a panel. In consortia such as ENCODE and Roadmap, the procedure for choosing which assay types to perform on each cell type is typically ad hoc. These decisions are made by the investigators involved, based on their intuition about the diversity of assay types, perhaps based on pairwise correlations between assays or similar simple metrics. Ernst and Kellis [Ernst and Kellis, 2015] proposed that imputation methods can evaluate the quality of a given of a given panel, but this approach cannot be used to select a panel (see Discussion).

In this work, we propose a principled method to solve the assay panel selection problem. Qualitatively, the method aims to identify, on the basis of existing data sets, assay types that yield complementary views of the genome. In practice, many pairs of assay types yield redundant information. For example, the transcription factors REST and RCOR1 are cofactors and therefore bind almost the same set of genomic positions [Andrés et al., 1999]. Similarly, the histone modification H3K36me3 primarily marks gene bodies, which are also transcribed and therefore measured by RNA-seq. Therefore, a great deal of what can be learned from the full set of assays can likely be learned by performing a small subset of the possible assays. This redundancy among assay types suggests that a carefully chosen panel of assays is likely to produce most of the information that would be obtained by performing all assays. Our solution to the assay panel selection problem is composed of two parts: an objective function that defines the quality of a panel, and an optimization algorithm that efficiently finds a panel that scores highly according to the objective function.

We propose to use an objective function called *facility location* (defined mathematically below), which measures what fraction of the information available in the full set of assay types is contained within the panel. This function has been previously applied in many fields, including document summarization [Lin and Bilmes, 2012], feature selection for machine learning [Liu et al., 2013], and exemplar-based clustering [Mirzasoleiman et al., 2013]. This function also corresponds to the objective function of the widely-used k-medoids clustering algorithm [Gomes et al., 2010]. Computing the facility location function requires a measure of similarity between assay types. For this purpose, we use the Pearson correlation between the two assays, averaged over the cell types in which the assays have been performed. We chose this objective function because it performs better than or comparably to the other methods we tried (Supplementary Notes 5.9–5.12).

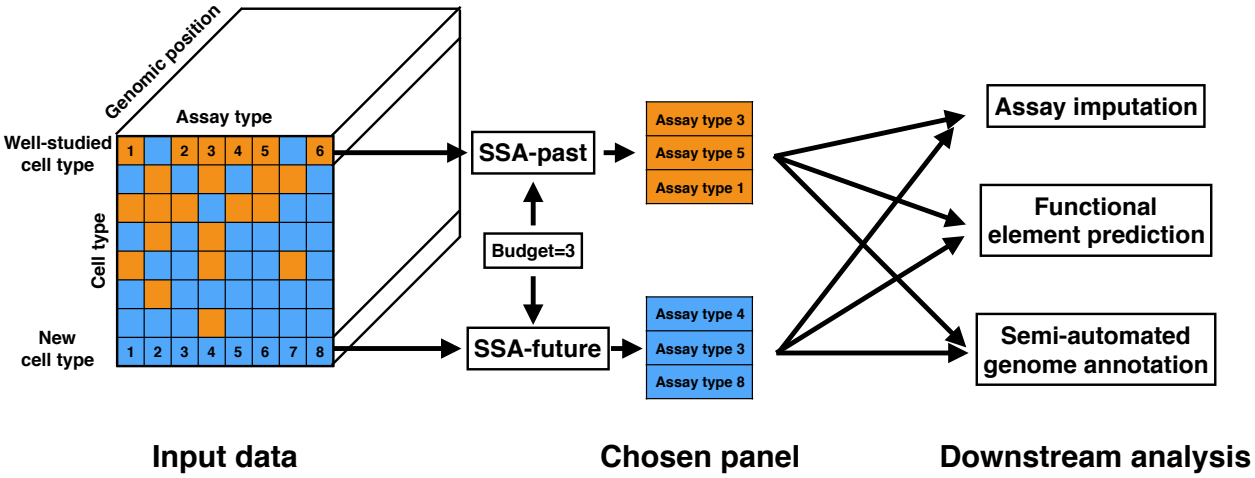
To optimize the facility location function, we borrow methods from the field of submodular optimization. A simple approach to selecting a panel of assays would evaluate the facility location function for every possible subset of assays and then choose the highest-scoring subset. Unfortunately, 216 possible assay types yield $2^{216} \approx 10^{65}$ possible panels of assays, so this approach is not feasible in practice. Fortunately, an efficient alternative selection method exists because the facility location function has the property of *submodularity*. The property of submodularity (defined mathematically below) is analogous to the property of convexity but is defined on discrete set functions rather than continuous functions. Submodular functions have a long history in economics [Vives, 2001, Carter, 2001], game theory [Topkis, 1998, Shapley, 1971], combinatorial optimization [Edmonds, 1970, Lovász, 1983, Schrijver, 2004], electrical networks [Narayanan, 1997], operations research [Cornunéjols et al., 1990], and more recently, machine learning [Narasimhan and Bilmes, 2005, Krause et al., 2008, Liu et al., 2013, Wei et al., 2014], but they are not yet widely used for problems in biology. Therefore, this application may serve as a model for how submodular optimization can be applied to biological problems more generally.

We apply existing submodular optimization algorithms to the facility location function to efficiently select a high-quality panel of assays, a method we call *submodular selection of assays* (SSA). There exists a large literature of methods for optimizing submodular functions. The optimization method we employ is very efficient and is theoretically guaranteed to find a solution whose quality

comes within a constant factor of the quality of the optimal solution [Nemhauser et al., 1978b].

In addition to proposing solutions to the assay panel selection problem, an important contribution of this work is development of three general methods for evaluating the quality of a selected panel of assays. These three methods correspond to three distinct practical applications of the selected panel: (1) the accuracy with which the panel can be used to impute the results of assays not included in the panel; (2) the accuracy with which the panel can be used to detect functional elements such as transcription factor binding sites, promoters, and enhancers; and (3) the quality of a whole-genome annotation produced using the panel. These evaluation metrics share the property that an informative and diverse set of assay types yields better performance, according to each metric, than does a redundant set. Note that these evaluation metrics differ from the objective function because they use information that is not available at the time a panel is chosen; therefore, the evaluation metrics themselves cannot be used directly to choose a panel. These three metrics will be useful for any future study of the quality of a panel of assays, independent of the particular procedure used to choose such a panel.

We consider two variants of the assay panel selection problem. We are primarily interested in the "future" variant, which arises when a researcher is interested in applying a panel of genomics assays to a new tissue type or cellular condition. In this case, the researcher must use previously performed assays in other cell types to choose a representative panel of assay types. However, we also consider the "past" variant, which arises when a researcher is interested in applying a computationally expensive analysis, such as a genome annotation method, that cannot efficiently be run on all available data sets. In this setting, the researcher must therefore choose a representative panel from the available data to use as input to the analysis. In this case, the researcher may use the data from assays performed on the cell type in question to inform their choice. We propose a variant of the method called SSA-past, which leverages this information to allow a researcher to choose such a representative panel in this setting.



Supplementary Figure 5.1: **Schematic of the selection process performed by submodular selection of assays (SSA).** The method takes as input all available existing genomics assays, where each assay is represented as a real-valued track over the genome. In the SSA-past mode, SSA selects a panel of already-performed assays to use as input to an expensive computational analysis. In the SSA-future mode, SSA chooses a panel of assay types to be performed in a new cell type. In both cases, the resulting data sets are provided as input to downstream analyses, which may include imputing assays that weren't performed, predicting the locations of functional elements, or semi-automated genome annotation.

5.2 Results

5.2.1 Submodular selection of assays identifies diverse panels of genomics assays

Submodular selection of assays (SSA) takes as input a collection of genomics data sets ("assays") and identifies a high-quality subset of those assays (Methods, Figure 5.1). Each input data set is represented as a real-valued signal vector over the genome. SSA begins by computing a pairwise similarity matrix that contains, for each pair of assay types, the mean Pearson correlation over all pairs of assays of those two types. The method employs a submodular function, called *facility location* (Methods), to estimate the quality of any possible panel of assay types. The facility location function takes a high value for a particular panel when all assay types have at least one similar

representative in the panel. We chose this strategy because it performed the best of those we tried according to our evaluation (Supplementary Notes 1-4). SSA then applies the *greedy submodular optimization* algorithm (Methods) to efficiently choose a panel of assays that maximizes this facility location function. The output of this method is an ordered list of assay types, where the top k assay types in this list represent a high-quality panel of size k. A detailed description and theoretical justification for the method is provided in Methods.

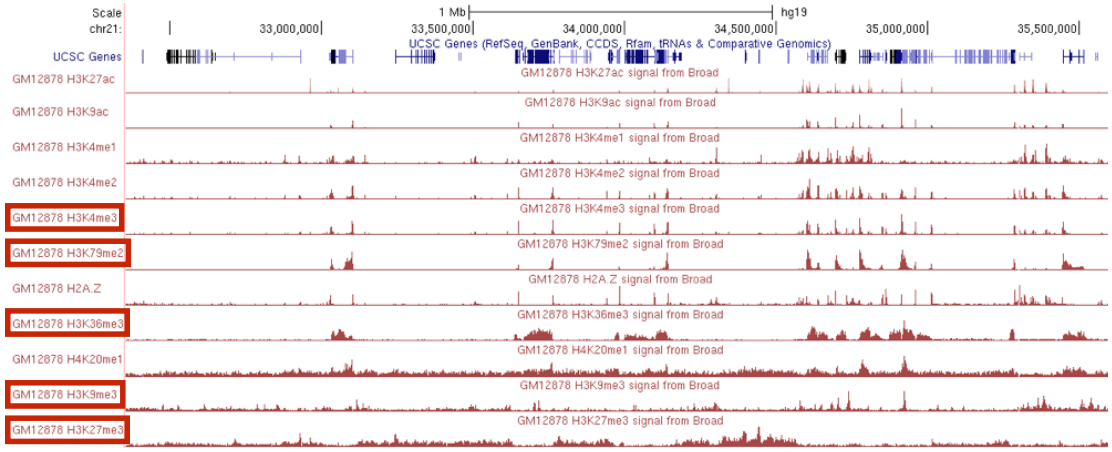
By analyzing data from the ENCODE and Roadmap Epigenomics Consortia, we found that SSA results in assay panels with diverse genomic functions. Because researchers generally perform panels of either histone modifications or transcription factor ChIP-seq assay types but rarely perform mixed panels, we ran the method separately on transcription factor and histone modification types (combined panels are shown in Supplementary Table 5.1). When choosing from transcription factors, SSA chooses factors that engage in diverse regulatory pathways (Figure 5.2A). The vast majority of transcription factors in our data set bind to promoters and enhancers and regulate the transcription of RNA Pol II-transcribed genes. The top five transcription factors chosen by SSA include three of these factors, each of which regulate very different regulatory pathways: SMARCB1, an ATP-dependent chromatin remodeler; PML, a tumor suppression factor; and STAT5A, a factor involved in developmental signal transduction [UniProt Consortium, 2014]. The top five also includes two factors—CTCF and BRF2—that are not solely involved in RNA Pol II-mediated transcription. CTCF, part of the cohesin complex, regulates chromatin conformation and enhancer-promoter insulation, and only about half of its binding sites occur in promoters or enhancers. BRF2 is part of the RNA Polymerase III complex, which transcribes rRNA, tRNA and other small RNAs. These two assay types each have low objective scores when in a panel by themselves ("singleton scores"), but are chosen by SSA because they measure different types of activity than the rest of the panel. Therefore, they are important to include in a diverse panel.

When choosing a panel of histone modifications, SSA selects marks that cover diverse types of genomic regions (Figure 5.2B,C). The top six histone modifications include a promoter mark (H3K4me3), an enhancer mark (H3K4me1), a gene mark (H3K79me2) and marks associated with both known types of repressive domains, facultative (H3K27me3) and constitutive (H3K9me3)

A

Choice order	Transcription factor	Function	Singleton score	Objective gain
1	SMARCB1	ATP-dependent chromatin remodeling	14.78	14.78
2	PML	Tumor suppression	14.12	1.84
3	STAT5A	Developmental signal transduction	14.17	0.91
4	CTCF	Chromatin conformation and insulation	8.40	0.81
5	BRF2	RNA polymerase III initiation complex	8.88	0.55

B



C

Choice order	Histone modification	Association	Singleton score	Objective gain	Objective loss if swapped in
1	**H3K4me3**	Promoters	3.18	3.18	
2	H3K79me2	Transcription	2.40	0.97	
3	**H3K9me3**	Constitutive heterochromatin	0.70	0.34	
4	**H3K27me3**	Facultative heterochromatin	0.86	0.25	
5	**H3K36me3**	Transcription	1.21	0.23	
6	**H3K4me1**	Enhancers	1.86	0.18	0.05 (H3K36me3)
7	H3K4me2	Regulatory	3.12	0.08	0.07 (H3K36me3)
8	H3K9ac	Regulatory	3.13	0.07	0.15 (H3K36me3)
9	H2A.Z	Promoters	2.47	0.05	0.16 (H3K27me3)
10	H4K20me1	Transcription	1.41	0.005	0.22 (H3K36me3)
11	H3K27ac	Regulatory	2.61	0	0.14 (H3K36me3)

Supplementary Figure 5.2: (Caption on following page)

Supplementary Figure 5.2: **Results of SSA-past on ENCODE+Roadmap data sets.** (A) Panels of transcription factors assays chosen by SSA-future. Each list is in the order assigned by SSA; for any size k, the top k assay types in the list are the chosen panel of this size. The "singleton score" is the objective value of a panel containing only the indicated assay type, and the "objective gain" indicates the improvement in objective that results from SSA adding the indicated assay type to the growing panel. Because there are 80 transcription factors, we display just the top five chosen by SSA. Associations are summarized from UniProt [UniProt Consortium, 2014]. (B) Redundancy in histone modification signal in the genome. The top five assay types chosen by SSA are boxed in red. (C) Similar to (A), but for the histone modification assays. There are only eleven histone modifications, so we can display the full list. Bold font indicates those histone modification assays chosen by the Roadmap Epigenomics consortium.

heterochromatin. The two repressive marks, H3K27me3 and H3K9me3, have the lowest singleton scores of all the histone modifications, but they give a high objective gain because they measure distinct activity from the rest of the panel. In contrast, even though H3K27ac is sometimes considered the best individual mark of enhancers and has a high singleton score, it is chosen last by SSA because it is redundant with other assay types in the panel, such as H3K4me1 and H3K9ac. The top six includes two different marks of transcription, H3K79me2 and H3K36me3, but these two modifications mark different parts of genes and are regulated differently relative to the gene's level of transcription [Li et al., 2007]. As expected, SSA ranks additional measures of regulation (H3K4me2, H2A.Z, H3K9ac and H3K27ac) low on the list because these marks are redundant with the regulatory marks H3K4me1 and H3K4me3.

Moreover, SSA almost exactly recapitulates the panel of histone modifications chosen by the Roadmap Epigenomics consortium (boldface entries in Table 5.2C). This consortium chose a set of five "core" histone modifications to assay across 111 human primary tissues. This choice was made by the members of this consortium based on their collective, expert knowledge. These five core histone modifications ranked among the top six modifications chosen by SSA. In fact, the SSA-chosen and Roadmap-chosen panels of size 5 have very similar scores according to the facility location function, ranking 1 and 16 respectively out of all $\binom{11}{5} = 2772$ possible panels of five histone modifications. Therefore, SSA closely reproduces careful, manual selection by experts in an

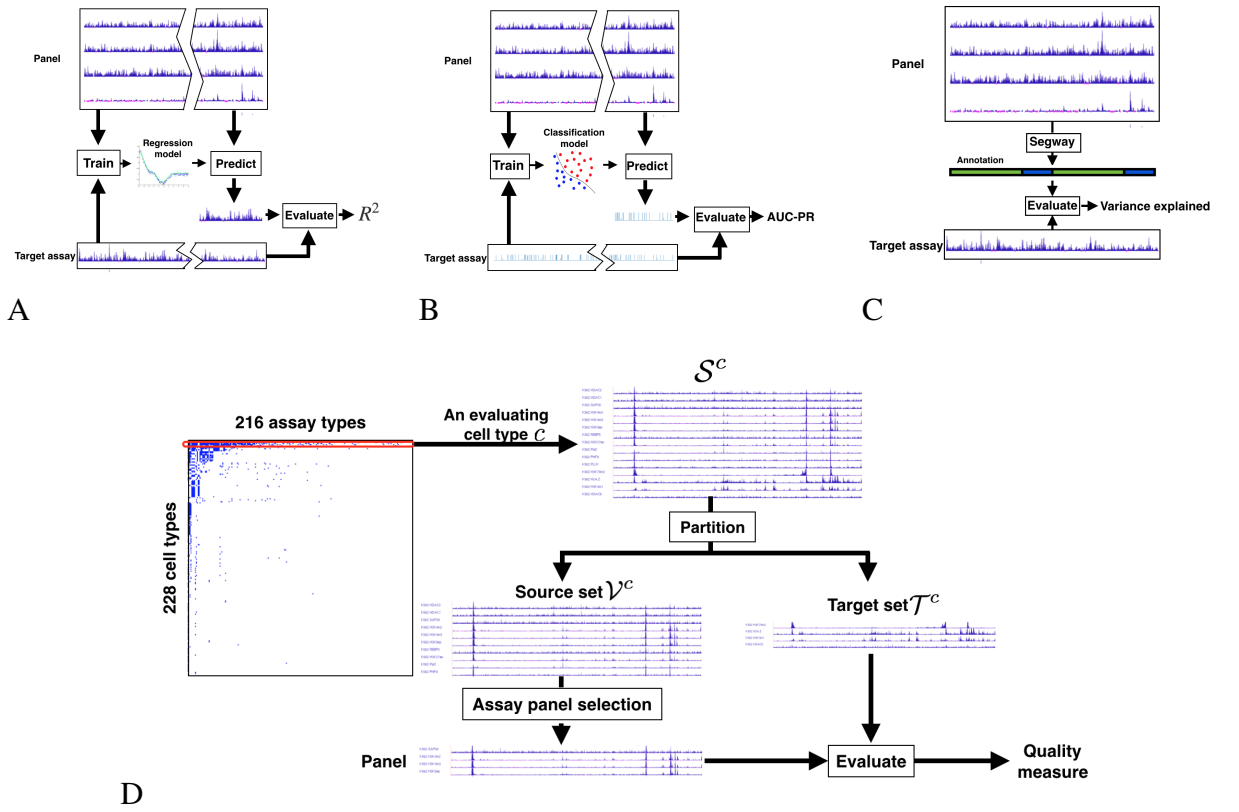
entirely automated and data-driven way.

In addition to its efficiency, SSA is flexible. First, in some circumstances, a researcher may have already performed a few assays in a given cell type or be confident that they want to perform them, and is therefore interested in choosing which assay types best complement these existing assays. SSA can be used in this scenario simply by restricting the returned set to include these assay types. For example, restricting the set of histone modifications to contain H3K27ac de-prioritizes other enhancer marks, such as H3K9ac and H3K4me1 (Supplementary Table 5.2). Second, a researcher might have a prior preference for certain classes of assays because they are easier to perform, less expensive, or measure features the researcher is especially interested in. SSA can include this prior preference by adding a weight to the objective for each assay type, reflecting the preference for that assay type. For example, selecting from all assay types (transcription factor, histone modification and DNA accessibility) jointly while placing a preference for (or against) histone modifications increases (or decreases) the fraction of histone modifications chosen to for the panel (Supplementary Table 5.3). Third, SSA can even alternatively be used to select cell types (instead of assay types) by transposing the cell type-assay type matrix and running the same analysis. Doing so produces a panel of cell types that includes a diverse mixture of mesodermal, endodermal and undifferentiated cells, and both normal and cancer-derived cells (Supplementary Table 5.4).

5.2.2 Three metrics evaluate the quality of a set of genomics assays.

In order to quantitatively evaluate SSA, we developed an evaluation framework for assay panel selection. We focused on three of the most common downstream applications of genomics data sets: (1) imputing assays that have not been performed, (2) locating functional elements such as promoters and enhancers, and (3) annotating the genome using a semi-automated method. We describe each metric briefly here, with full details provided in Methods.

The first evaluation metric, *assay imputation*, measures how well a chosen panel of assays can be used to predict assays that have not been performed (Figure 5.3A). We train a regression model to predict each assay outside of the panel on the basis of the assays within the panel, using random subsets of the genome for training and testing, respectively. High performance on the assay



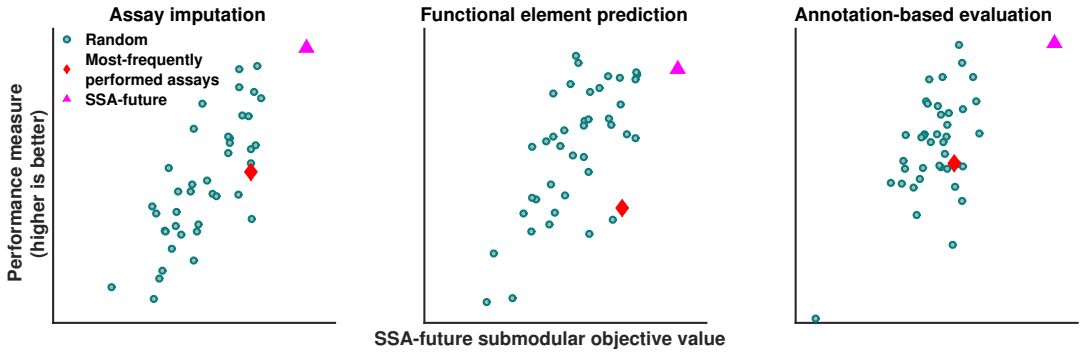
Supplementary Figure 5.3: **Evaluation strategies**. Schematics of the three evaluation metrics: (a) assay imputation, (b) functional element prediction, and (c) annotation-based evaluation, as well as (d) the overall cross-validation evaluation strategy.

imputation metric indicates that the panel contains all of the information in the assays outside of the panel. Moreover, recent work on imputation has showed that it is often effective to train a regression model on data from reference cell types and apply it to a target cell type (Discussion).

The second evaluation metric, *functional element prediction*, is similar to the assay imputation metric but focuses specifically on how well a chosen panel of assays can be used to locate functional elements such as promoters and enhancers (Figure 5.3B). Because there are few validated examples of each type of element, we use experimentally determined binding of transcription factors, as measured by transcription factor ChIP-seq, as a proxy for functional elements. We train a classifier model to predict the locations of these elements on the basis of the assays within the panel. High performance on the functional element prediction metric indicates that a panel can be used to accurately locate functional elements. Although both the assay imputation and functional element prediction evaluation metrics aim to predict genomics data sets, functional element prediction focuses on the small fraction of the genome corresponding to transcription factor binding sites.

The third evaluation metric, *annotation-based evaluation*, measures how effectively a given panel can be used to annotate the genome through a semi-automated genome annotation (SAGA) method (Figure 5.3C). SAGA methods, which include HMMSeg [Day et al., 2007], ChromHMM [Ernst and Kellis, 2010], Segway [Hoffman et al., 2012] and others [Thurman et al., 2007, Lian et al., 2008, Filion et al., 2010, Kharchenko et al., 2010, Jaschek and Tanay, 2009, Larson et al., 2013, Biesinger et al., 2013], annotate the genome on the basis of a panel of genomics assays. They simultaneously partition the genome and annotate each segment with an integer label such that positions with the same label exhibit similar patterns of activity. These methods are semi-automated because a person must interpret the biological meaning of each integer label. SAGA methods have been shown to recapitulate known functional elements including genes, promoters and enhancers. Given a particular panel of assays, we perform annotation-based evaluation by using this panel as input to Segway and measuring how well the resulting genome annotation corresponds to patterns observed in the assays outside of the panel. High performance on this metric indicates that the chosen panel can be used to produce a comprehensive annotation of the genome.

Applying these metrics to evaluate a method for choosing panels is complicated by two factors.



Supplementary Figure 5.4: **Relationship between the facility location objective function and evaluation metrics.** Each dot corresponds to one of 40 randomly-chosen panels. Pink triangles indicate results from maximizing the SSA-future facility location function; red diamonds indicate the panel of most-frequently performed assay types (Supplementary Table 5.5). These results were computed in GM12878, using panels of four assay types.

First, no cell type has had all assay types performed in it, so we perform evaluation separately on each cell type in order to evaluate against all available assay types. Second, these evaluation metrics must be used to compare to assays outside of the panel, so we use a cross-validation strategy in which we hold out a *target set* for evaluation and choose panels from the remaining *source set*, repeating this process for many choices of target set. This evaluation strategy enables principled evaluation that compares all methods against the same held-out standard while using only the available data sets (Methods, Figure 5.3D).

5.2.3 Panels chosen by SSA outperform alternative methods according to three evaluation metrics.

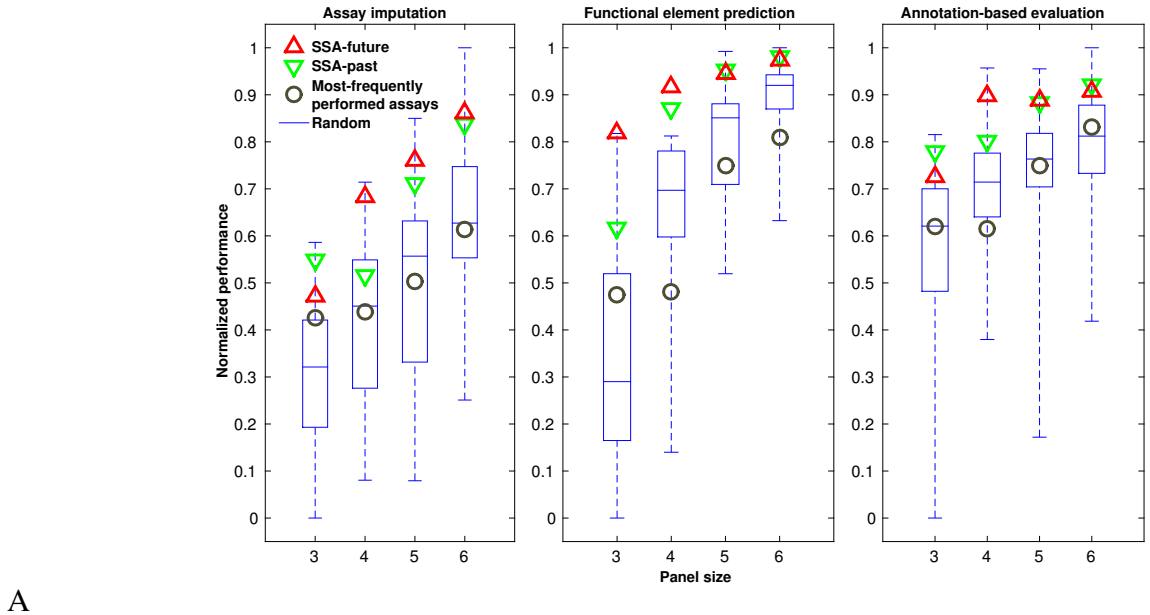
We applied this panel evaluation framework to evaluate SSA. First, to determine the most effective objective function, we compared the facility location function and four other potential objective functions based on the pairwise similarity matrix. We found that the facility location function had a higher Spearman correlation with the three evaluation metrics than the other objective functions we tried, and this trend was consistent across multiple cell types (Supplementary Note 5.9). In addition, we found that the facility location function produced the highest correlation with the three evaluation metrics when we defined the similarity between a pair of assay types as the mean Pearson correlation

between this pair, as opposed to the median, maximum or other aggregation function (Supplementary Note 5.10). We also compared between two strategies for deriving the Pearson correlation. The first computes the correlation based on the random samples of the genomics positions, and the second on the DNase peaks positions only. We found that almost comparable performance is achieved on the three evaluation metrics for the two strategies (Supplementary Note 5.11). We lastly compared the effectiveness between Pearson correlation and Spearman correlation for computing the similarity measure. We found that the consistently better performance is achieved when the similarity is defined using the Pearson correlation (Supplementary Note 5.12). These observations led us to choose our variant of facility location as the SSA's objective function.

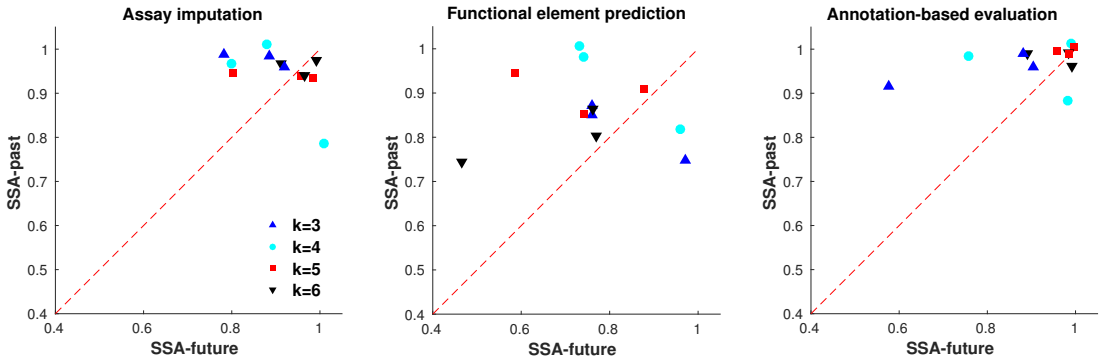
Next, we used our three evaluation metrics to compare SSA to alternative panel selection approaches. As a baseline, we considered randomly selected panels of a given size. We also considered the panel of most frequently performed assays (Supplementary Table 5.5) as a good proxy for a likely data-driven choice. We found that the panels reported by SSA perform among the top few percent out of the space of all possible panels, and this high performance is consistent across panel sizes, evaluation cell types and performance metrics (Figure 5.5A). We found that SSA also greatly outperforms the panel of most-frequently performed assay types. Indeed, this commonly-performed panel actually performs worse than the average panel in many cases, which may be a consequence of the fact that the most commonly-performed assay types measure broad marks of regulation, such as histone modifications and DNA accessibility, which do not have the specificity to identify pathway-specific elements. These results demonstrate quantitatively that panels chosen by SSA are effective when applied to their most common downstream tasks.

5.2.4 SSA can select a subset of performed assays as input to an expensive analysis

So far we have considered panel selection in the "future" setting, where a researcher is planning to experimentally perform a panel of assay types. Panel selection is sometimes also important in the "past" setting, where a researcher wishes to apply a computationally expensive analysis that cannot be efficiently applied to all assays together and therefore must be applied to a smaller panel. For example, training a statistical model to perform semi-automated genome annotation jointly on



A



B

Supplementary Figure 5.5: **Performance of panel selection strategies.** (a) Boxplots show the distribution of evaluation metrics over 40 random panels on data from cell type GM12878. The panels of most-frequently performed assays are composed of the top k most frequent assay types available in our data set, where k is the size of the panel. Each evaluation metric is normalized to lie within $[0, 1]$ by subtracting the lowest value and dividing by the highest. (b) Scatter plot between the performance of SSA-past and SSA-future across cell types K562, GM12878, and H1-hESC. Each dot in the plot corresponds to the two variants of SSA for a panel size evaluated using a metric in a cell type. The performance is measured as the fraction of panels that perform worse than the SSA-chosen panel, estimated by comparing to 40 randomly-selected panels.

dozens of assays across many cell types is computationally expensive. A strategy in which each cell type is represented by a smaller panel of assays might yield very similar annotations using a fraction of the computational resources. In this setting, the assays themselves are available to the selection algorithm, so we compute the similarity matrix based on these values themselves (SSA-past) rather than estimating the similarities by aggregating across cell types (SSA-future). Importantly, in the selection of past assays case, a different panel can be selected for each cell type, based on the available data. Alternatively, if the researcher would like to use the same panel of past assays for analyses several cell types, SSA-future can be used to select a panel that maximizes the average quality over all target cell types (Supplementary Table 5.6). To test SSA in the past setting, we used the same evaluation strategy as in the future setting, but using the source assays themselves to compute the similarity matrix. SSA performs consistently well according to these metrics, and it performs slightly better on some cell types in the past than the future setting due to the availability of this additional information (Figure 5.5B).

5.3 Discussion

The growing availability of a large number of types of genomics assays means that choosing a panel of genomics assays is a key step in any genomics project. Previously, these panels were chosen in an ad hoc fashion. We have developed submodular selection of assays (SSA), a method for choosing high-quality panels using submodular optimization. This method is computationally efficient, results in high-quality panels according to several quality measures, and is mathematically optimal under some assumptions. By applying SSA, researchers can now easily choose a high-quality panel of assay types to perform on any cell type of interest. These higher-quality panels will allow researchers to achieve the same utility from performing fewer assays, saving thousands of dollars in labor and reagent costs per cell type. This panel selection framework can also be used partway through the investigation of a cell type, when several assays are already available. By modifying the facility location function to include the availability of these assays, SSA can be used to determine the most-informative next experiments to perform. In doing so, SSA will take into account the information in these existing assays and choose additional assay types that measure

distinct genomic features.

A key feature of the submodular optimization approach is the flexibility afforded by the broad class of submodular objective functions, and the ability to encode appropriate prior knowledge into the selection of the objective. In this manuscript, we focused on optimizing the facility location function. However, the same submodular optimization framework can be used to optimize other objective functions that may prove to be more relevant for certain applications. Several other functions may be useful in practice. First, if some assays are more expensive or time consuming than others, then the objective function can be modified to incorporate this cost. Second, if some assays are inherently preferable to others, for example because they have better-established processing pipelines, then the objective can incorporate this preference and trade off choosing both diverse and established assay types. Third, entirely different types of panel attributes may be valuable for a particular application, which can be formalized as a different objective function, such as the alternatives we discuss in Supplementary Note 5.9. As long as the resulting objective function remains submodular, it will be efficiently optimizable using either the greedy algorithm for monotone non-decreasing functions or other efficient methods [Buchbinder et al., 2012, 2014] for non-monotone functions. Moreover, such modifications are intuitive to design and easy to implement.

The facility location function can also be used to guide manual assay panel selection. A researcher may seek to optimize hard-to-quantify characteristics of a panel, such as familiarity with the protocols involved or the panel's concordance with panels performed on other cell types. In this case, the researcher may choose to perform a panel that has slightly poorer quality as measured by the facility location function in order to optimize these other criteria. Still, in such a setting, manual investigation of the objective values associated with different panels can provide useful insights.

The framework presented here can also be extended to many related problems. We have discussed two such variants that apply in the scenarios, "I would like to select a panel of assay types to perform, taking cost into account", and "I have carried out many assays in this cell type and would like to choose a subset of this to use as input to an expensive computational analysis". This framework can also trivially be extended to the problem, "I have already performed several

assay types in a particular cell type of interest and would like to select several more to perform," by simply restricting the output panel to contain the previously-performed assays (Supplementary Table 5.2). The same framework also applies to the problem, "I have a set of assay types commonly performed in several cell types and would like to choose several assay types from the set that are the most informative and representative to study the cell types" by restricting the aggregation of the assay type similarity to the cell types at hand (Supplementary Table 5.6). In addition, by deriving a similarity measure between cell types, the methods presented here could be used to solve the problem, "Based on orthogonal data such as gene expression profiles, which new cell types should I perform assays in?" Finally, related methods may be able to apply to the problem "I have carried out assays across a variety of cell types and assay types, and I would like to select a set of additional assays to apply in any combination of cell and assay types."

The evaluation strategy we introduce here can be used to evaluate any proposed strategy for panel selection, whether or not this method is based on submodular optimization. This framework is composed of two parts. First, a cross-validation strategy allows for principled comparison of methods under the restriction that not all assays are available in all cell types, and that a panel must not be evaluated on an assay type that it contains. Second, three distinct metrics capture the three primary downstream applications of genomics data sets.

The problem of assay panel selection was posed previously by Ernst and Kellis [Ernst and Kellis, 2015]. These authors proposed the assay imputation evaluation metric for evaluating a panel of genomics assays. However, this type of evaluation metric cannot be used to choose an assay panel for two reasons. First, evaluating the accuracy of an imputation model requires that the target data set already be performed. Second, even if such data sets were available, evaluating all possible panels of size K from N assay types requires applying evaluation to $\binom{N}{K}$ possible panels, which is hopelessly computationally expensive. For example, the method of Ernst and Kellis was reported to take roughly one hour to train per panel, so choosing a panel of five assay types from the 188 previously-performed assay types would take on the order of 10^5 processor-years. In contrast, the facility location objective function we define does not require that the target data sets be available in order to evaluate the function, and the submodular optimization approach requires only a polynomial

number of evaluations. Moreover, we define two additional evaluation metrics (functional element prediction and annotation-based evaluation) that complement evaluation by evaluation by assay imputation.

In addition to be applicable to the assay panel selection problem, the same submodular optimization framework may also be useful in the following scenario: "Rather than deciding which assay types to choose, one wants to choose a set of informative and representative cell types to study". We call this problem *cell type panel selection*. Although this setting is out of scope of this work, it can be viewed as a dual variant of the assay panel selection problem. To address this problem, a similarity matrix among cell types can be derived in the same manner as the assay type similarity matrix using the similar aggregation strategies. Instantiated on this cell type similarity matrix, a submodular function (e.g., facility location function) may then serve as the optimization objective for outputting the selection order of the cell types. The selection order of the cell types using the SSA methodology is displayed in Supplementary Table 5.4. The flexibility in SSA (e.g., forcing to choose certain assay types, and weighing certain assay types with higher preference) easily carries over to this setting making the cell type selection framework also very general. Moreover, the methodologies proposed here for evaluating the quality of a panel of assay types (e.g., assay imputation, and annotation-based evaluation) can be easily extended to assess the goodness a cell type panel selection approach.

One limitation of any data-driven analysis is that it is limited by any imperfections in the data sets used. For example, if all available assays of a given type happen to be of particularly good or poor quality, then the correlations associated with this assay type will appear to the algorithm to be particularly strong or weak, respectively. Similarly, any mislabeled assays, batch effects, or other artifacts may also influence whether certain assay types will be chosen in a panel. Future assays of that type may not be expected to exhibit the same artifactual patterns, so the resulting panels could be suboptimal. Therefore, it is always important to scrutinize the results of data-driven approaches like this one to understand whether patterns in the available data are predictive of future experiments. Modifications of this approach that, for example, find and remove faulty assays before input into the algorithm might result in different panels. However, our evaluation metrics are also

entirely data-driven, so we cannot use them to explore these issues.

As noted above, submodular optimization is widely used for discrete problems in other fields but is not yet widely used in biology. We hope that the current work can serve as a model for how submodular optimization can be applied to other problems in biology. As with convex optimization, the same toolbox of submodular optimization methods can be applied to a wide variety of problems, and any innovations to this toolbox improve all solutions. Therefore we expect that in the future, submodular optimization will be used for other discrete problems in biology, such as for selecting panels of DNA mutations to test in a functional screen or removing redundancy in protein sequence data sets.

5.4 Methods

5.4.1 Genomics data

We acquired all public genomics data from the ENCODE (http://hgdownload.cse.ucsc.edu/goldenPath/hg19/encodeDCC/) and Roadmap Epigenomics (https://sites.google.com/site/anshulkundaje/projects/epigenomeroadmap) projects as of January 2015. These data sets were processed by the two consortia into real-valued data tracks, as described previously [Hoffman et al., 2013, Kundaje et al., 2015]. We omitted all assays with more than 1% unspecified positions, which may indicate errors during processing or mapping. We manually curated these assays to unify assay type and cell type terminology and, when multiple assays were available, we arbitrarily chose a representative assay for each (cell type, assay type) pair. This procedure resulted in a total of 1,359 assays comprised of a total of 216 assay types and 228 cell types. The assay types include ChIP-seq with a variety of targets (both histone modification and transcription factor), DNase-seq, FAIRE-seq, Repli-seq and RNA-seq. The full list of assays is given as supplementary data. We applied the inverse hyperbolic sine transform $\text{asinh}(x) = \ln(x + \sqrt{x^2 + 1})$ to all signal data. This function has the compressing effect of a function like $\log x$ for large values of x but it is defined at zero and has much less of a compressing effect for small values. The asinh transform has been shown to be important for reducing the effect of large values in analysis of

genomics data sets [Johnson, 1949, Hoffman et al., 2012]. Transcription factor ChIP-seq peaks were called by each consortium for each factor using MACS using an irreproducible discovery rate (IDR) threshold of 0.05 [Zhang et al., 2008, Landt et al., 2012].

5.4.2 Notation

We use the following notation below to facilitate the description of our method. We use the term "assay type" to mean a particular genomics assay protocol that may be performed in any cell type (for example "ChIP-seq targeting H3K27me3") and "assay" to mean a particular assay type performed in a particular cell type. The term "cell type" refers to any cellular state that may be queried with a genomics assay, which may refer to any combination of cell line, tissue type, disease state (such as cancer), individual, or drug perturbation. We refer to a cell type as c and the entire set of all cell types as $\mathcal{C}$ ($|\mathcal{C}| = 228$). We use a to refer to an assay type, A for a subset of assay types, and $\mathcal{A}$ for the set of all assay types ($|\mathcal{A}| = 216$). We use s to denote a single assay (that is, a given assay type performed in a given cell type), S for a set of assays, and $\mathcal{S}$ as the set of all available performed assays. Given any cell type $c \in \mathcal{C}$ we define the set of assay types performed in this cell type as $\mathcal{A}^c$ and the corresponding assays as $\mathcal{S}^c$. We define $I = \{1, \ldots, n\}$ as the set of all positions in a genome. An assay s is represented as a vector of length n; i.e., $s \in \mathbb{R}^n$. We denote its ith entry (i.e., the value of assay s at genomic position i) as $s(i)$.

5.4.3 Submodular optimization

A *submodular* function [Fujishige, 2005] is defined as follows: given a finite size m set $V = \{1, 2, \ldots, m\}$, a discrete set function $f : 2^V \rightarrow \mathbb{R}$ that offers a real value for any subset $S \subseteq V$ is submodular if and only if:

$$f(S) + f(T) \geq f(S \cup T) + f(S \cap T), \forall S, T \subseteq V. \tag{5.1}$$

Defining $f(s|S) \triangleq f(s \cup S) - f(S)$, submodularity can equivalently be defined as $f(s|S) \geq f(s|T), \forall S \subseteq T$ and $s \notin T$. That is, the incremental gain of adding item s to the set decreases when

the set to which s is added to grows from S to T. In this work, the whole set V represents a set of genomics assays and the set function $f(S)$ represents a measure of quality of a subset of assays $S \subseteq V$.

Two other properties of set functions are relevant to this setting. First, a set function f is defined as *monotone non-decreasing* if

$$f(s|S) \geq 0, \forall s \in V \setminus S, S \subseteq V. \tag{5.2}$$

Second, we say that f is *normalized* if $f(\emptyset) = 0$.

In this work we are interested in the problem of maximizing a submodular function subject to a constraint on the size of the reported set. That is, we are interested in solving the problem

$$\text{maximize } f(S), \quad \text{subject to } |S| \leq k \tag{5.3}$$

for some integer $k \leq |V|$. In this work, we require that f is submodular, monotone-nondecreasing and normalized.

While this problem is NP-hard, it can be approximately solved by a simple greedy algorithm with a worst-case approximation factor $(1 - e^{-1})$ [Nemhauser et al., 1978a]. This is also the best solution obtainable in polynomial time unless $P = NP$ [Feige, 1998]. The algorithm starts with the empty set $S_0 = \emptyset$ and at each iteration i adds the element s_i that maximizes the conditional gain $f(s_i|S_{i-1})$ with ties broken arbitrarily (i.e., finding $s_i \in \text{argmax}_{e \in V \setminus S_{i-1}} f(e|S_{i-1})$) and then updates $S_i \leftarrow S_{i-1} \cup \{s_i\}$. The algorithm stops when the cardinality constraint is met with equality. This algorithm has a time complexity of $O(km)$ function evaluations. The running time of this algorithm can be improved to a near-linear number without any performance loss by further exploiting the submodularity property [Minoux, 1978].

5.4.4 *Facility location function*

In this work we use the *facility location* function to measure the quality of panel of assay types. The facility location function [Cornunéjols et al., 1990] $f_{\text{fac}} : 2^V \to \mathbb{R}$ is defined as follows:

$$f_{\text{fac}}(S) = \sum_{s' \in V} \max_{s \in S} r_{s',s}, \tag{5.4}$$

where $r_{s',s}$ measures the pairwise similarity between assays s' and s (defined below). Intuitively, the facility-location function takes a high value when every assay in V has at least one similar representative in S. To instantiate the facility location function, one needs to compute and store a pair-wise similarity graph, which takes $O(m^2)$ memory complexity.

5.4.5 *Assay type similarity*

We use the following strategy to define the similarity between each pair of assay types in order to use this similarity to define a facility location function. We define this similarity differently depending on the application: In the selection of past assays setting, the particular assays performed in the cell type of interest c are available, while in the selection of future assays setting we must estimate this similarity from reference cell types.

In the selection of past assays setting, we directly use the signal vectors s_i and s_j to derive the similarity. We define this similarity as $r_{s_i,s_j} = |\rho_{s_i,s_j}| \in [0, 1]$, where ρ_{s_i,s_j} is the Pearson correlation between the signal vector s_i and s_j. Pearson correlation is frequently used to evaluate the similarity between genomics assays [ENCODE Project Consortium, 2012]. For efficiency, we compute the correlation measure ρ_{s_i,s_j} only across a subset of genomic positions $I' \subseteq I$, where I' is randomly subsampled from I, and $|I'| \approx 0.01|I|$.

In the selection of future assays setting, the assays in the cell type c are not available, but the assays performed in cell types other than c, $\mathcal{S} \setminus \mathcal{S}^c$, are available. Let $a_i, a_j \in \mathcal{A}$ be the assay types associated with the assay s_i and s_j, respectively. Let $\mathcal{S}^{a_i}$ be the set of assays in $\mathcal{S}$ with type a_i. We approximate the similarity between s_i and s_j by aggregating the all similarity between the pairs in

$\mathcal{S}^{a_i} \setminus s_i$ and $\mathcal{S}^{a_j} \setminus s_j$. We utilize the aggregation strategy by taking the average of these similarity scores. Mathematically, the aggregated similarity is defined below:

$$r_{s_i,s_j} \triangleq \frac{1}{|\mathcal{S}^{a_i} \setminus s_i|} \frac{1}{|\mathcal{S}^{a_j} \setminus s_j|} \sum_{s \in \mathcal{S}^{a_i} \setminus s_i} \cdot \sum_{s' \in \mathcal{S}^{a_j} \setminus s_j} |\rho_{s,s'}|, \tag{5.5}$$

We chose to use the average correlation r because the facility location function defined via the similarities aggregated in this way correlated best with our evaluation metrics. We provide the comparison of such aggregation strategy against other strategies in the Supplementary Note 5.10.

5.4.6 Evaluation cross-validation strategy

We would prefer to apply our method once to select a single panel of assay types. However, doing so could result in a panel of assay types that have not been performed in any cell type (or very few cell types), which would prohibit evaluating the quality of this panel. Therefore, we apply a cross-validation strategy that repeatedly holds out a subset of assay types for evaluation and selects a panel from the remaining assay types, and we perform this cross-validation separately for each cell type in turn (Figure 5.3D). To evaluate the quality of our method with respect to a cell type c, we restrict ourselves to selecting from the set of assays performed in c ($\mathcal{S}^c$). We randomly partition $\mathcal{S}^c$ into 10 equally-sized, disjoint folds. Of the 10 folds, a single fold is retained as the target set $\mathcal{T}^c$, and the remaining 9 blocks are used as the source set $\mathcal{V}^c$. We select a panel of assays $S \subseteq \mathcal{V}^c$ from the source set $\mathcal{V}^c$ and evaluate the panel on the assays relative to the target set $\mathcal{T}^c$ using the three evaluation metrics described below. The process is then repeated ten times, with each of the ten folds used once as the target set. We average the ten results are averaged to produce a single number representing the performance.

5.4.7 Assay imputation

The *assay imputation* evaluation metric measures the ability of a panel of assay types to be used to impute the results of other assay types outside the panel (Figure 5.3A). We formalize assay

imputation metric as a regression problem in which the assays in the panel S are used as features to predict the target set assays, $s' \in \mathcal{T}^c$. In this regression problem we have one labeled example for each position in the genome.

As our regression model, we use support vector regression with a Gaussian kernel. To construct the training and test data, we randomly choose disjoint sets of genomic positions $I^{Tr}, I^{Te} \subseteq I$, where $I^{Tr} \cap I^{Te} = \emptyset$. In our experiments, we set $|I^{Tr}| = 5,000$ and $|I^{Te}| = 2,000$. Given the panel $S = \{s_1, \ldots, s_{|S|}\}$, a target assay $s' \in \mathcal{T}^c$, and the training genomic positions I^{Tr}, we create the training data as $\mathcal{D}^{Tr} = \{x^i, y^i\}_{i \in I^{Tr}}$, where $x^i = [s_1(i), s_2(i), \ldots, s_{|S|}(i)]^T$ and $y^i = s'(i)$. Similarly, the test data set is constructed as $\mathcal{D}^{Te} = \{x^i, y^i\}_{i \in I^{Te}}$. The hyperparameters of the regression model are tuned using 5-fold cross validation. We measure the performance of the trained model on the test data $\mathcal{D}^{Te}$ as the squared correlation coefficient $\theta_{s'}$. We repeat this evaluation process for every target assay in $\mathcal{T}^c$ and report the performance of the panel S as the average squared correlation coefficient $\theta = \frac{1}{|\mathcal{T}^c|} \sum_{s' \in \mathcal{T}^c} \theta_{s'}$.

5.4.8 Functional element prediction

The *functional element prediction* evaluation metric evaluates how well a panel of assays can predict the genomic locations of functional elements such as promoters, enhancers and insulators. Because there are few validated examples of each type of element, we use experimentally-determined binding of transcription factors, as determined by transcription factor ChIP-seq peaks, as a proxy for functional elements. Most known types of functional elements can be characterized by the binding of particular transcription factors [Visel et al., 2009, Burgess-Beusse et al., 2002]. Note that functional element prediction is similar to assay imputation in the sense that both evaluation metrics aim to predict the output of a genomics assay; however, functional element prediction focuses on just transcription factor binding sites, whereas assay imputation focuses on the whole genome. Similar to assay imputation, we consider this metric separately for each cell type. For an evaluation cell type c, we denote the set of transcription factor ChIP-seq assays performed in c as $\hat{\mathcal{S}}^c \subseteq \mathcal{S}^c$. Given a bi-partition of $\mathcal{S}^c$ into the source set $\mathcal{V}^c$ and the target set $\mathcal{T}^c$, we choose from the source set $\mathcal{V}^c$ a panel of assays, and we evaluate functional element prediction only on the target assays in the

set $\hat{\mathcal{T}}^c = \mathcal{T}^c \cap \hat{\mathcal{S}}^c$, in contrast to the assay imputation metric where all assays in $\mathcal{T}^c$ are used for evaluation.

For a target transcription factor assay $s \in \hat{\mathcal{T}}^c$, let p be a binary vector $\{0,1\}^n$ indicating the genomic positions where s has a peak as called by the peak-calling algorithm. That is, $p(i) = 1$ if there is a peak at position i, and $p(i) = 0$ otherwise. We use a support vector machine (SVM) with Gaussian kernel to predict p given a panel of assays $S \subseteq \mathcal{V}^c$. For a given testing factor p, we refer to the positions where $p = 1$ as I_+ and the set of positions where $p = 0$ as $I_- = I \setminus I_+$. We randomly choose $I_+^{Tr} \subseteq I_+$ and $I_-^{Tr} \subseteq I_-$ as the positive and negative positions to generate training samples. Similarly, the testing samples are randomly chosen from $I_+^{Te} \subseteq I_+ \setminus I_+^{Tr}$ and $I_-^{Te} \subseteq I_- \setminus I_-^{Tr}$. Given the panel $S = \{s_1, \ldots, s_{|S|}\}$ of assays and the set of positive training genomic positions I_+^{Tr}, we construct the set of positive training samples as $\mathcal{D}_+^{Tr} = \{x^i, +1\}_{i \in I_+^{Tr}}$ where $x^i = [s_1(i), \ldots, s_{|S|}(i)]^T$. Similarly, we construct the negative training samples, positive test samples, and negative test samples as $\mathcal{D}_-^{Tr}$, $\mathcal{D}_+^{Te}$, and $\mathcal{D}_-^{Te}$, respectively. The SVM is first trained on the training data set $\mathcal{D}^{Tr} = \{\mathcal{D}_+^{Tr}, \mathcal{D}_-^{Tr}\}$, and then evaluated on the testing data set $\mathcal{D}^{Te} = \{\mathcal{D}_+^{Te}, \mathcal{D}_-^{Te}\}$.

Because there are far more genomic positions that are not a functional element than there are positions that are, measures of predictive accuracy such as the total fraction of correct predictions ("accuracy") and the area under the receiver operating characteristic curve do not offer a reasonable measure of performance. Instead, we compute the area under the curve of a precision-recall plot (AUC-PR), which is particularly well suited for settings with imbalanced class distributions [Craven and Bockhorst, 2005, Davis and Goadrich, 2006]. In our experiments we set $|I_+^{Tr}| = 200$, $|I_-^{Tr}| = 20,000$, $|I_+^{Te}| = 100$ and $|I_-^{Te}| = 10,000$. We apply 5-fold cross validation for tuning the hyperparameters of the SVM. Let $\gamma_{s'}$ be the normalized area under curve for the precision-recall plot (i.e., $\gamma_{s'} \in [0,1]$) for each target assay $s' \in \hat{\mathcal{T}}^c$. We illustrate this procedure schematically in Figure 5.3B. We report the performance as the average AUC-PR on all target assays, i.e., $\gamma = \frac{1}{\hat{\mathcal{T}}^c} \sum_{s' \in \hat{\mathcal{T}}^c} \gamma_{s'}$.

5.4.9 *Annotation-based evaluation*

The *annotation-based evaluation* metric measures the quality of a panel of genomics assays according to the quality of the genome annotation that is obtained by inputting the panel into a semi-automated genome annotation (SAGA) algorithm. SAGA algorithms are widely used to jointly model diverse genomics data sets. These algorithms take as input a panel of genomics assays and simultaneously partition the genome and label each segment with an integer such that positions with the same label have similar patterns of activity. These algorithms are considered "semi-automated" because a human performs a functional interpretation of the labels after the annotation process. Examples of SAGA methods include HMMSeg [Day et al., 2007], ChromHMM [Ernst and Kellis, 2010], Segway [Hoffman et al., 2012] and others [Thurman et al., 2007, Lian et al., 2008, Filion et al., 2010]. These genome annotation algorithms have had great success in interpreting genomics data and have been shown to recapitulate known functional elements including genes, promoters and enhancers. We use the SAGA method Segway in this work.

In order to apply annotation-based evaluation to a panel of assays, we input this panel into a SAGA algorithm and evaluate the resulting annotation (Figure 5.3C). Intuitively, a diverse panel of assays input to a SAGA algorithm should more accurately capture important biological phenomena than a redundant panel. To evaluate the quality of an annotation relative to a particular genomics data set, we use the *variance explained* measure [Libbrecht et al., 2015]. Given an evaluation cell type c we randomly partition $\mathcal{S}^c$ into a source set $\mathcal{V}^c$ and a target set $\mathcal{T}^c$. For a given panel of assays $S \subseteq \mathcal{V}^c$, we first train a Segway model based on the panel and then obtain an annotation y. Segway outputs an annotation $y \in \mathcal{Y}^n$, where $\mathcal{Y} = \{1, 2, \ldots, k\}$ is a set of k labels that an annotation can take on at each genomic position. For each target assay $s' \in \mathcal{T}^c$, we measure the quality of the annotation y as how well it explains the variance of the assay s'. We first compute the signal mean of s' over the positions assigned a given label ℓ as

$$\mu_\ell \triangleq \frac{\sum_{i=1}^{n} \mathbf{1}(y(i) = \ell) s'(i)}{\sum_{i=1}^{n} \mathbf{1}(y(i) = \ell)} \quad \text{for } \ell \in \{1 \ldots k\}. \tag{5.6}$$

We then define a predicted signal vector $\hat{s}'$ with $\hat{s}'(i) = \mu_{y(i)}$ and compute the prediction error as

$d_i = \hat{s}'(i) - s'(i)$. We compute the residual standard deviation of the signal vector as

$$\sigma_{\text{res}} \triangleq \text{stdev}(d_{1:n}) = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(d_i - \text{mean}(d_{1:n}))^2} = \sqrt{\frac{1}{n}\sum_{i=1}^{n} d_i^2}. \tag{5.7}$$

The last equality holds because $\text{mean}(d_{1:n}) = 0$ by construction. σ_{res} measures the residual standard deviation of the target assay s' accounting for the annotation y. Let $\sigma_{\text{ov}} = \text{stdev}(s'(1:n))$ be the overall standard deviation of the assay s'. The normalized variance explained by the annotation y is then

$$\alpha_{s'} = \frac{\sigma_{\text{ov}} - \sigma_{\text{res}}}{\sigma_{\text{ov}}}. \tag{5.8}$$

Observe that σ_{ov} always upper bounds σ_{res}. The measure $\alpha_{s'} \in [0, 1]$ represents the fraction of the variance of the assay s' explained by the annotation y, where larger values indicate better agreement.

In our experiments, we trained the Segway model with 10 EM random initializations (using GMTK [Bilmes and Rogers, 2015]) and 15 labels at 100 base-pair resolution. We report the performance as the averaged measure on all target assays as $\alpha = \frac{1}{|\mathcal{T}^c|}\sum_{s' \in \mathcal{T}^c} \alpha_{s'}$.

5.5 Source code

Source code for SSA and computed assay type similarity matrix are available as Supplemental Material and online at `http://github.com/melodi-lab/Submodular-Selection-of-Assays`.

5.6 Author contributions

All authors devised the method and designed the analysis. KW and MWL analyzed the data, implemented the method, developed the software and wrote the initial manuscript. All authors edited and approved the final manuscript. WSN and JAB jointly supervised the project.

5.7 Competing interests

The authors declare no competing interests.

5.8 Supplementary Information

5.9 Supplementary Note 1: Comparison of potential objective functions

We compared four classes of submodular objectives for selecting panels of assays. All functions are defined via a pairwise similarity graph, where we denote $r_{v,v'}$ as the similarity measure between the v and $v' \in V$.

Facility location function:

$$f_{\text{fac}}(A) = \sum_{v \in V} \max_{a \in A} r_{v,a} \tag{5.9}$$

This function is monotone and submodular.

Saturated coverage function:

$$f_{\text{sat}}(A) = \sum_{v \in V} \min\{\sum_{a \in A} r_{v,a}, \beta \sum_{v' \in V} r_{v',v}\}, \tag{5.10}$$

In this function, $0 \leq \beta \leq 1$ is a hyperparameter that controls the saturation of the coverage for each item $v \in V$. In the experiment we set $\beta = k/n$, where k is the target number of assays to select, and n is the size of entire set V of all available assays. The saturated coverage function is also monotone submodular.

Diversity function:

$$f_{\text{div}}(A) = -\sum_{a \in A} \sum_{a' \in A} r_{a,a'} \tag{5.11}$$

The diversity function evaluates a subset A as the sum of the pairwise dissimilarity among all items in the subset. Maximizing this function naturally encourages choosing a diverse set of items. The diversity function is submodular, but not monotone.

Log determinant function:

$$f_{\text{log-det}}(A) = \log\det(I + \lambda S_A) \tag{5.12}$$

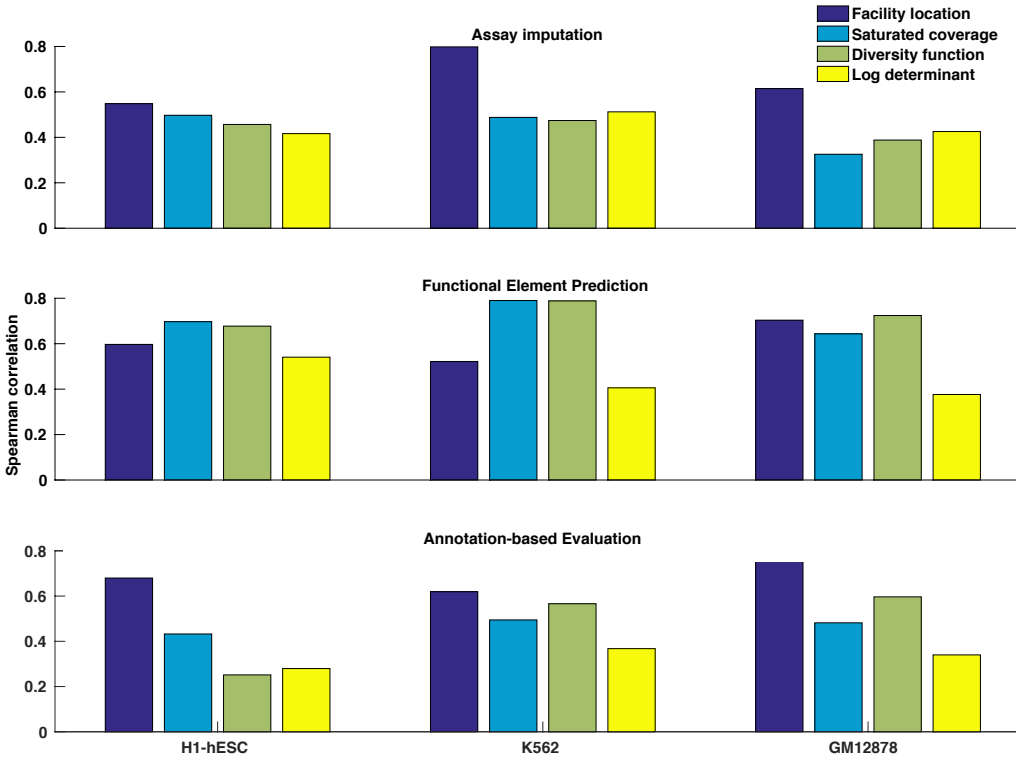
In this function, $\lambda > 0$ is a hyperparameter, and S_A is the pairwise similarity matrix indexed by the subset $A \subseteq V$. The log determinant function satisfies submodularity and monotonicity. This function has been shown to naturally capture the notion of diversity in a data set; therefore, maximizing this function always leads to a set of diverse items.

We measure the performance of these submodular objectives by examining how their objective valuations correlate with the three proposed evaluation metrics (assay imputation, functional element prediction, and annotation-based evaluation). The experiment is performed under the selection of past assays setting, where we compute the similarity between a pair of assays using the Pearson correlation. Given an evaluation metric, a cell type c, a selection budget K, and a submodular objective f, we randomly draw n subsets $\{A_i\}_{i=1}^n$ of assays from the cell type c, where each subset A_i is of size K. We then compute the Spearman correlation between the submodular valuation $\{f(A_i)\}_{i=1}^n$ and the performance measure $\{y_i\}_{i=1}^n$ under the evaluation metric. We report the correlation measure averaged over budget constraints $K = 3, 4, 5, 6$. In the experiment we set $n = 20$, and we test on the cell types K562, H1-hESC, and GM12878.

The results (Figure 5.6) suggest that the facility location function, in most cases, yields the highest correlation with the three evaluation metrics.

5.10 Supplementary Note 2: Comparison among various similarity measures

In this section, we study how the performance of SSA for the selection of future assays scenario varies with different choices of similarity aggregation strategies. We focus on strategies for defining the pairwise similarity between assay types in an evaluating cell type c given the assays performed on cell types other than c. We consider the six aggregation strategies r^1, r^2, r^3, r^4, r^5, and r^6, where they take the average, 0^{th}, 25^{th}, 50^{th}, 75^{th}, and 100^{th} percentile over the available similarity scores,



Supplementary Figure 5.6: Comparison among various submodular functions in terms of the Spearman correlation between function valuation and the performance metrics.

respectively. For completeness, we also give their corresponding mathematical definition below:

$$r_{s_i,s_j} \triangleq \frac{1}{|\mathcal{S}^{a_i} \setminus s_i|} \frac{1}{|\mathcal{S}^{a_j} \setminus s_j|} \sum_{s \in \mathcal{S}^{a_i} \setminus s_i} \cdot \sum_{s' \in \mathcal{S}^{a_j} \setminus s_j} |\rho_{s,s'}|, \tag{5.13}$$

$$r^2_{s_i,s_j} \triangleq \text{percentile}(\{|\rho_{s,s'}| : s \in \mathcal{S}^{a_i}, s' \in \mathcal{S}^{a_j} \setminus s_j\}, 0), \tag{5.14}$$

$$r^3_{s_i,s_j} \triangleq \text{percentile}(\{|\rho_{s,s'}| : s \in \mathcal{S}^{a_i}, s' \in \mathcal{S}^{a_j} \setminus s_j\}, 25), \tag{5.15}$$

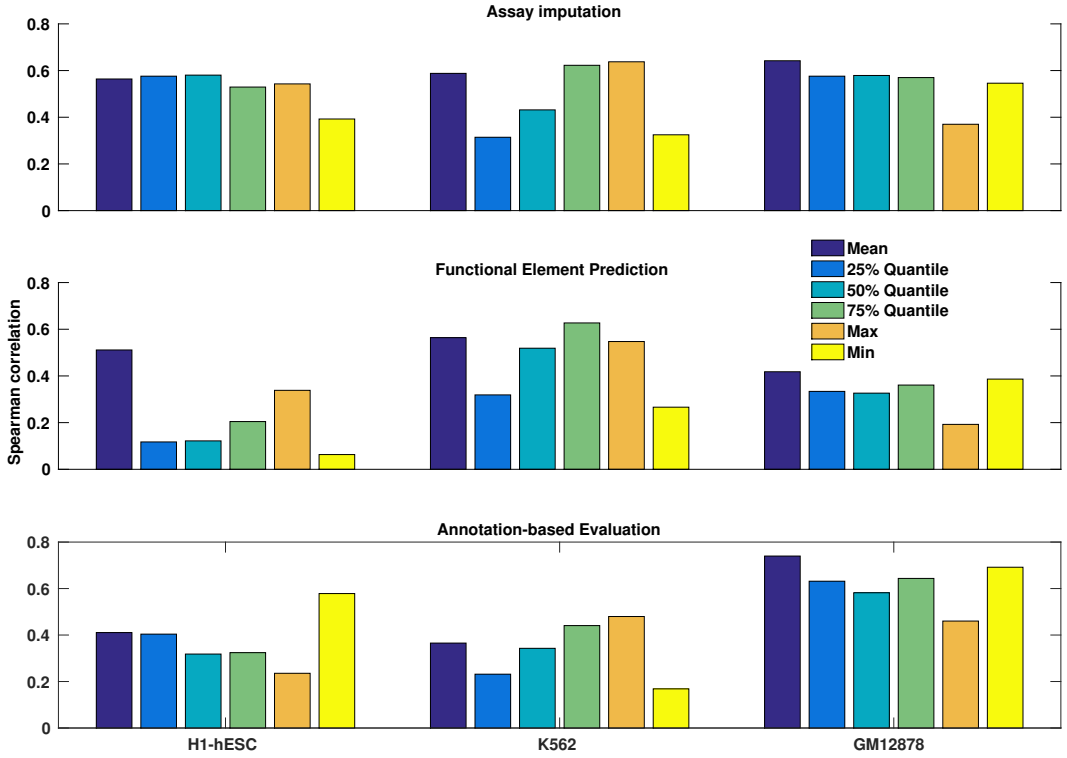
$$r^4_{s_i,s_j} \triangleq \text{percentile}(\{|\rho_{s,s'}| : s \in \mathcal{S}^{a_i}, s' \in \mathcal{S}^{a_j} \setminus s_j\}, 50), \tag{5.16}$$

$$r^5_{s_i,s_j} \triangleq \text{percentile}(\{|\rho_{s,s'}| : s \in \mathcal{S}^{a_i}, s' \in \mathcal{S}^{a_j} \setminus s_j\}, 75), \tag{5.17}$$

$$r^6_{s_i,s_j} \triangleq \text{percentile}(\{|\rho_{s,s'}| : s \in \mathcal{S}^{a_i}, s' \in \mathcal{S}^{a_j} \setminus s_j\}, 100), \tag{5.18}$$

where the function percentile(C, p) returns the p^{th} percentile of the items in the list C sorted in non-decreasing order.

We evaluate the performance of the aggregation strategies using the facility location function. In particular we utilize the six different aggregation strategies to compute the pairwise similarity measure for defining the facility location function. We test separately on three cell types: H1-hESC, K562, and GM12878. Similar to the previous experiment for comparing among different submodular objectives, we measure the performance of an aggregation strategy as how the valuation given by facility location function defined via the similarity matrix computed using this strategy correlates with the three proposed evaluation metrics. Following the same evaluation procedure we show the Spearman correlation measure for each aggregation strategy on each cell type and evaluation metric in Figure 5.7. We observe that the aggregation strategies of taking the mean or 75^{th} percentile yield the best performance for most cell types and most evaluation metrics. Between these two strategies we observe that the aggregation by mean performs marginally better. Therefore we choose it as the strategy for computing the similarity between assay types in the propose approach SSA under the selection of future assays setting.



Supplementary Figure 5.7: Comparison among various similarity aggregation strategies in terms of the Spearman correlation between the facility location function valuation instantiated by the similarity measure and the performance metrics.

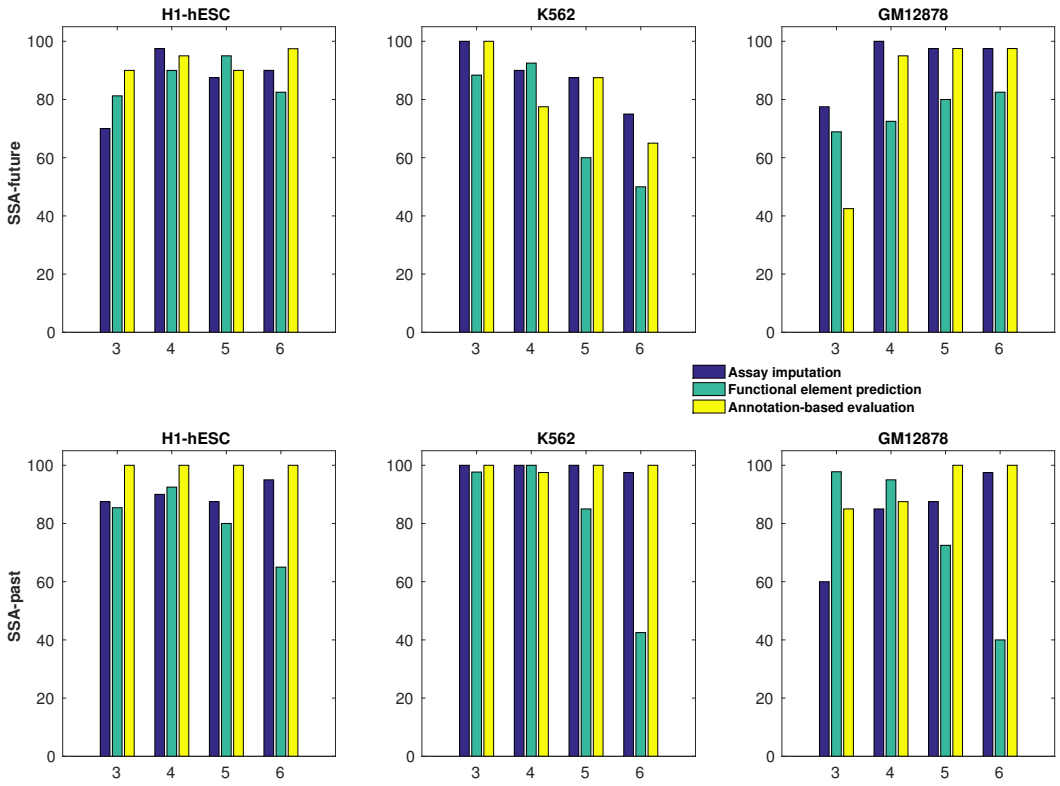
5.11 Supplementary Note 3: Comparison of different supports for similarity computation

In this section, we examine how the similarity measure may be affected if we change the support for deriving the similarity between assay types. In the main paper, we derive the similarity between assays as the Pearson correlation between the values of these two assays on a randomly chosen subset of genomic positions. In this section, we explore deriving the similarity based only on the set of genomic positions that are identified as DNase peaks, since the response in the transcription factors is only captured on these DNase peaks positions.

In Figure 5.8, we report the performance of the DNase peaks based SSA on all evaluation metrics as well as the three cell types. For completeness, we also show the corresponding performance of the variant of SSA that we used in the main main paper in Figure 5.9. Note that this variant of SSA has the similarity measure computed over a randomly sampled subset of genomic positions. To better illustrate the comparison between these two variants, we plot the performance measure of the DNase peaks approach (each bar in Figure 5.8) against the random genomic positions approach (each bar in Figure 5.9) in the form of scatter plot (Figure 5.10). We observe that consistent and significant improvements over the random selection baseline are achieved by the variant of SSA using DNase peaks only. Between the two variants of SSA, it is hard to establish the superiority of one variant over the other according to Figure 5.10.

5.12 Supplementary Note 4: Comparison of different correlation metrics as the similarity measure

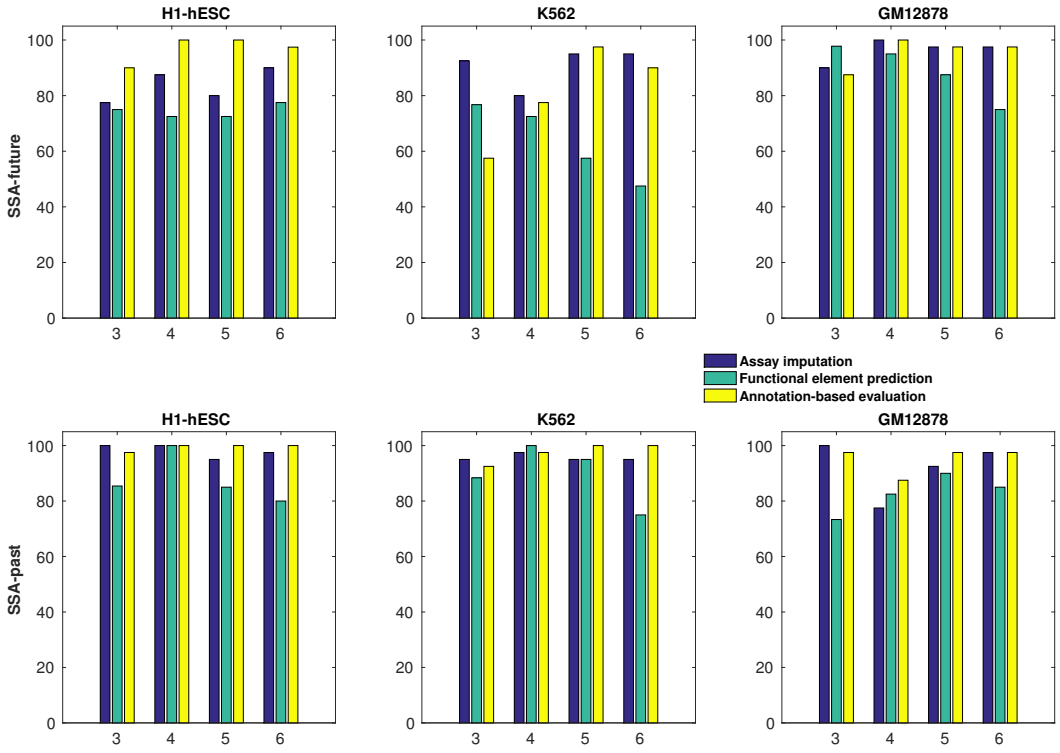
In this section, we examine how the similarity measure may be affected by different choice of correlation metrics. For the results reported in the main paper, we derive the similarity between assays via the absolute Pearson correlation. Here, we explore a variant of SSA whose similarity measure is defined via the absolute Spearman correlation. In Figure 5.11, we report the performance of such variant on all evaluation metrics as well as the three cell types. To contrast the performance difference between the Pearson correlation-based SSA and Spearman correlation-based SSA, we again show a scatter plot of the performance achieved between these two variants in Figure 5.12.



Supplementary Figure 5.8: Performance of the Dnase peaks-based SSA relative to an estimate of the performance on all possible panels. The vertical axis shows the fraction (%) of panels that perform worse than the SSA-chosen panel for a given setting, estimated by comparing to 40 randomly-selected panels.

We observe that the Pearson correlation based SSA significantly and consistently outperforms the Spearman correlation counterpart suggesting that Pearson correlation is more effective for capturing similarity between assays in our tasks.

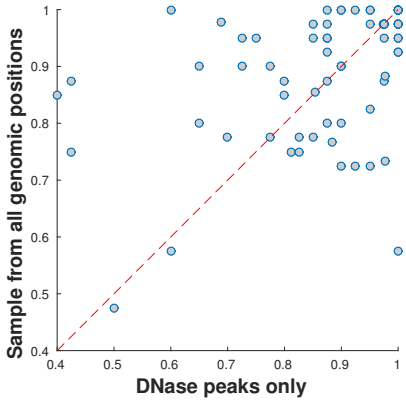
5.13 Supplementary Tables



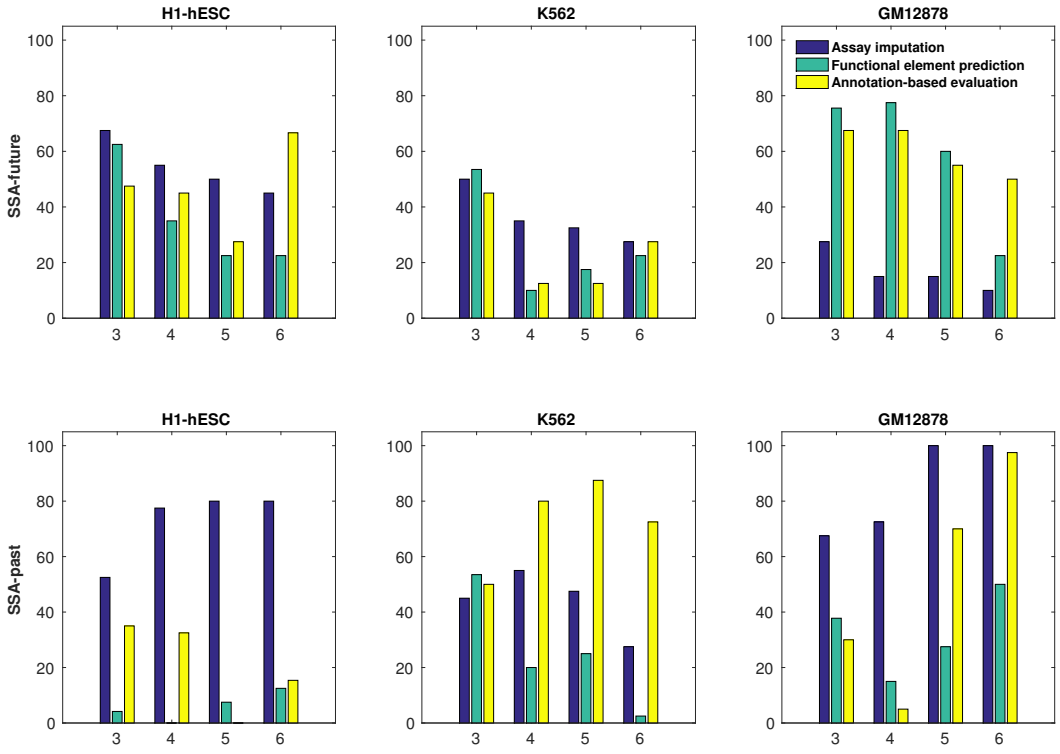
Supplementary Figure 5.9: Performance of the random genomics positions-based SSA relative to an estimate of the performance on all possible panels.

Choice order	Assay type	Singleton score	Objective gain
1	NFATC1	36.38	36.38
2	ILF2	33.37	4.60
3	PHF8	27.01	3.06
4	POU5F1	22.36	2.26
5	SMARCC1	13.63	1.93
6	IRF4	28.22	1.42
7	HNF4G	15.78	1.33
8	HMGN3	24.93	1.27
9	ILF3	29.09	1.04
10	ZNF217	16.00	1.03

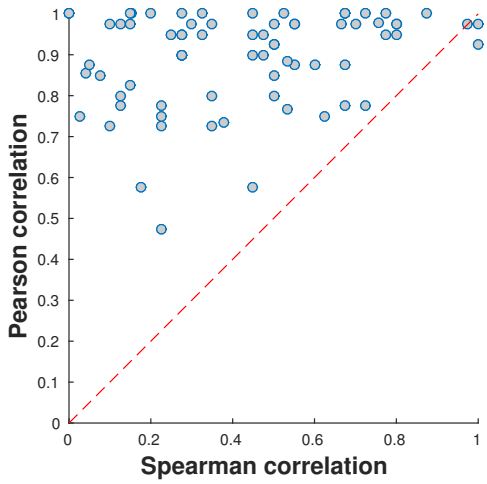
Supplementary Table 5.1: The choice order of SSA on all assay types.



Supplementary Figure 5.10: Scatter plot between the two variants of SSA with different genomic support for similarity computation. Each dot in the plot corresponds to the performance (again measured as relative to an estimate of the performance on all possible panels) of the two variants of SSA for a selection budget evaluated using a metric in a cell type (i.e., one bar in Figure 5.8 and its counterpart in Figure 5.5). Its x- and y-values are the performance measure for the DNase peaks-based SSA and random genomic positions-based SSA, respectively.



Supplementary Figure 5.11: Performance of the absolute Spearman correlation-based SSA relative to an estimate of the performance on all possible panels.



Supplementary Figure 5.12: Scatter plot between the two variants of SSA with different correlation metrics as the similarity measure. Each dot in the plot corresponds to the performance (again measured as relative to an estimate of the performance on all possible panels) of the two variants of SSA for a selection budget evaluated using a metric in a cell type (i.e., one bar in Figure 5.11 and its counterpart in Figure 5.5). Its x- and y-values are the performance measure for the Spearman correlation-based SSA and the Pearson correlation-based SSA, respectively.

Order	None	H3K4me1	H3K4me2	H3K9ac	H2A.Z	H4K20me1	H3K27ac
1	H3K4me3	H3K4me1	H3K4me2	H3K9ac	H2A.Z	H4K20me1	H3K27ac
2	H3K79me2	H3K4me3	H3K79me2	H3K79me2	H3K79me2	H3K4me3	H3K79me2
3	H3K9me3	H3K79me2	H3K4me3	H3K9me3	H3K4me3	H3K79me2	H3K4me3
4	H3K27me3	H3K9me3	H3K9me3	H3K27me3	H3K9me3	H3K9me3	H3K9me3
5	H3K36me3	H3K27me3	H3K27me3	H3K4me3	H3K36me3	H3K27me3	H3K27me3
6	H3K4me1	H3K36me3	H3K36me3	H3K36me3	H3K27me3	H3K36me3	H3K36me3
7	H3K4me2	H3K4me2	H3K4me1	H3K4me1	H3K4me1	H3K4me1	H3K4me1
8	H3K9ac	H3K9ac	H3K9ac	H3K4me2	H3K4me2	H3K4me2	H3K4me2
9	H2A.Z	H2A.Z	H2A.Z	H2A.Z	H3K9ac	H3K9ac	H2A.Z
10	H4K20me1	H4K20me1	H4K20me1	H4K20me1	H4K20me1	H2A.Z	H3K9ac
11	H3K27ac	H3K27ac	H3K27ac	H3K27ac	H3K27ac	H3K27ac	H4K20me1

Order	None	MAFF	ZNF263	BDP1	E2F6	RBBP5
1	SMARCB1	MAFF	ZNF263	BDP1	E2F6	RBBP5
2	PML	SMARCB1	SMARCB1	SMARCB1	SMARCB1	PML
3	STAT5A	PML	PML	PML	PML	SMARCB1
4	CTCF	CTCF	STAT5A	STAT5A	CTCF	STAT5A
5	BRF2	STAT5A	CTCF	CTCF	STAT5A	CTCF
6	MAFF	BRF2	BRF2	BRF2	BRF2	BRF2
7	ZNF263	ZNF263	MAFF	MAFF	MAFF	MAFF
8	BDP1	BDP1	BDP1	ZNF263	ZNF263	ZNF263
9	E2F6	E2F6	E2F6	E2F6	BDP1	BDP1
10	RBBP5	RBBP5	RBBP5	RBBP5	RBBP5	E2F6

Supplementary Table 5.2: **Performing SSA after one assay has already been performed.** We ran SSA while restricting the output panel to include each assay type in turn, by initializing the submodular greedy algorithm with the assay type in question and then running the algorithm as normal. As expected, restricting the panel to include a particular assay type de-prioritizes assay types that measure similar types of activity. (Top) Choice order of histone modification assays by SSA if one assay type is required to be included. (Bottom) Choice order of transcription factors by SSA if one assay type is required to be included. In both tables, each assay type in the table is encoded with a different color to allow easy comparison to different columns.

Order	$\alpha = 0$	$\alpha = 0.5$	$\alpha = 1$	$\alpha = 5$	$\alpha = 10$
1	NFATC1	NFATC1	NFATC1	NFATC1	H2A.Z
2	ILF2	ILF2	ILF2	H3K4me3	H3K4me3
3	PHF8	PHF8	H3K4me3	H3K9me3	H3K79me2
4	POU5F1	POU5F1	POU5F1	H3K79me2	H3K9me3
5	SMARCC1	SMARCC1	SMARCC1	H3K27me3	H3K27me3
6	IRF4	IRF4	PML	H3K36me3	H3K36me3
7	HNF4G	HNF4G	H3K79me2	H3K4me1	H3K4me2
8	HMGN3	HMGN3	MBD4	H2A.Z	H3K4me1
9	ILF3	H3K4me3	H3K27me3	H3K4me2	H3K9ac
10	ZNF217	H3K79me2	HMGN3	H3K9ac	H4K20me1
11	CCNT2	ILF3	H3K36me3	H4K20me1	H3K27ac
12	PML	ZNF217	H3K9me3	H3K27ac	NFATC1
13	CEBPZ	CCNT2	H3K4me1	ILF2	ILF2
14	NR2F2	PML	IRF4	POU5F1	POU5F1
15	MLL5	CEBPZ	H3K4me2	PML	PML
16	SREBF2	NR2F2	H3K9ac	SMARCC1	SMARCC1
17	XRCC4	MLL5	H2A.Z	MBD4	MBD4
18	IKZF1	SREBF2	ZNF217	HMGN3	HMGN3
19	MBD4	XRCC4	H4K20me1	IRF4	IRF4
20	PRDM1	IKZF1	CCNT2	ZNF217	ZNF217

Supplementary Table 5.3: **Incorporating modular weighting to encourage certain assay types to be chosen.** We ran SSA with a variant that weights certain assay types more highly in the selection process. In this experiment, we examine the choice order of assay types by changing the weights on histone modification assays in a selection experiment involving both histone modification and transcription factor assays. We implement this experiment by first defining a combined objective $h^\alpha(A) = f(A) + \alpha m(A)$, where f is the facility location objective and m is the modular function, with each modular score capturing the importance of the item. Note that a set function m is modular if $m(A) = \sum_{a \in A} m(a)$ holds for all $A \subseteq V$. We define a hyperparameter $\alpha > 0$ that controls the trade-off between f and m. We define the modular function m by assigning the score $m(a)$ for each histone mod assay as 1 and for other assay types as 0. Given a choice of α, we use the greedy algorithm to optimize h^α leading to an ordering of the assays. The table displays the resulting order for various choices of α. All histone modification assay types are color encoded for ease of comparison.

Order	Cell types	Description	Singleton value	Objective gain
1	HBMEC	brain microvascular endothelial cells (mesoderm blood vessel)	48.31	48.31
2	LNCaP	prostate adenocarcinoma (endoderm prostate cancer)	46.77	1.08
3	GM12864	B-lymphocyte (mesoderm blood)	43.74	0.39
4	IMR90	fetal lung fibroblasts (endoderm lung)	43.26	0.37
5	H7-hESC	undifferentiated embryonic stem cells	40.13	0.33

Supplementary Table 5.4: The choice order of cell types using SSA methodology.

	Assay Type
1	DnaseSeq
2	H3K4me3
3	CTCF
4	FaireSeq
5	POLR2A
6	H3K27me3
7	H3K36me3
8	H3K79me2
9	H3K4me2
10	H3K9me3

Supplementary Table 5.5: The 10 most frequently performed assay types.

Choice order	Assay type	Singleton score	Objective gain
1	DnaseSeq	4.311	4.311
2	H3K4me3	4.224	1.338
3	CTCF	2.515	0.837
4	H3K79me2	2.997	0.825
5	FaireSeq	2.624	0.808
6	POLR2A	3.293	0.788
7	CEBPB	3.001	0.78
8	TAF1	3.047	0.773
9	REST	2.641	0.694
10	GABPB1	3.684	0.692
11	MAX	3.515	0.677
12	EP300	3.7	0.664
13	H3K9ac	2.84	0.62
14	RAD21	4.245	0.192

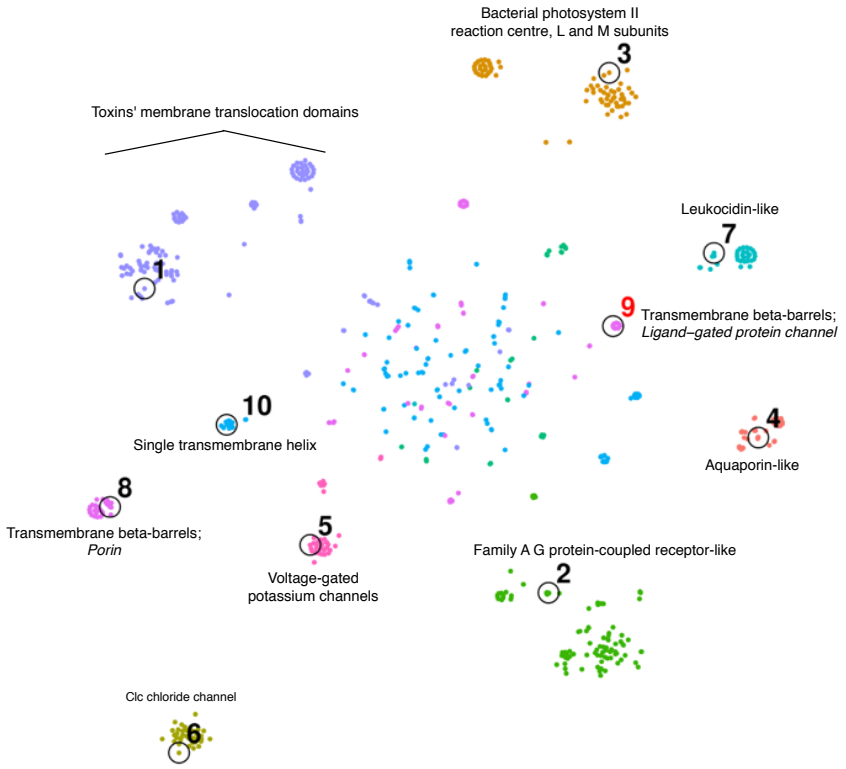
Supplementary Table 5.6: The choice order of SSA-past on common assay types shared among the six ENCODE tier 1+2 cell types: K562, GM12878, H1-hESC, A549, HepG2, and HeLa-S3.

Chapter 6

ELIMINATING REDUNDANCY AMONG PROTEIN SEQUENCES USING SUBMODULAR OPTIMIZATION

Abstract

Submodular optimization, a discrete analogue to continuous convex optimization, has been used with great success in many fields but is not yet widely used in biology. We apply submodular optimization to the problem of removing redundancy in protein sequence data sets. This is a common step in many bioinformatics and structural biology workflows, including creation of non-redundant training sets for sequence and structural models as well as selection of "operational taxonomic units" from metagenomics data. We demonstrate that the submodular optimization approach results in representative protein sequence subsets with greater structural diversity than sets chosen by existing methods. In particular, we compare to a widely used, heuristic algorithm implemented in software tools such as CD-HIT, as well to as a variety of standard clustering methods, using as a gold standard the SCOPe library of protein domain structures. In this setting, submodular optimization consistently yields protein sequence subsets that include more SCOPe domain families than sets of the same size selected by competing approaches. We also show how the optimization framework allows us to design a mixture objective function that performs well for both large and small representative sets. The framework we describe is theoretically optimal under some assumptions, and it is flexible and intuitive because it applies generic methods to optimize one of a variety of objective functions. This application serves as a model for how submodular optimization can be applied to other discrete problems in biology. Source code is available at https://github.com/mlibbrecht/submodular_sequence_repset.



Supplementary Figure 6.1: **Protein sequence representative set selection.** Points represent protein sequences, projected into 2D based on their pairwise similarities according to BLAST using t-SNE [Van der Maaten and Hinton, 2008]. All sequences are depicted from the SCOPe Class "Membrane and cell surface proteins and peptides." Color with text labels indicates SCOPe fold, and italicized text indicates SCOPe family within a given fold. Black circles indicate the top ten representative sequences chosen by the greedy algorithm on a mixture of facility-location (Rankprop sim; 0.5 weight) and sum-redundancy (percent ID sim; 0.5 weight). Numbers labeling these circles indicate choice order. Choice #9 is colored red because it is the second chosen sequence within the same fold. To produce the 2D embedding, we first computed a $n \times n$ pairwise distance matrix (for a set of n sequences) with the formula: $\text{distance}(i, j) = 1 - \text{percent\_id}(i, j)/100$. We used multidimensional scaling (MDS) to project this distance matrix to a $30 \times n$ feature matrix . This MDS preprocessing step is similar to the standard preprocessing step used by tSNE that applies principle component analysis (PCA) to project a high-dimensional feature matrix to a $30 \times n$ feature matrix [Van der Maaten and Hinton, 2008]. We used tSNE to project this $30 \times n$ matrix into a $2 \times n$ matrix, which is plotted. The diffuse points in the center are a common feature of tSNE plots, and are a consequence of the algorithm imperfectly recapitulating the n-dimensional similarity matrix in two dimensions.

6.1 Introduction

Redundancy is ubiquitous in biological sequence data sets, and this redundancy often impedes analysis. For example, because some proteins, such as those involved in disease or industrial applications, are studied by many more researchers than proteins with less important or unknown function, protein sequence databases contain hundreds or thousands of slight variants of well-studied proteins and only one or a few copies of the sequences of less-studied proteins. Removing this redundancy has two benefits. First, doing so removes computational and statistical issues introduced by this redundancy. Second, members of a non-redundant set of sequences can be interpreted as representatives of the whole set, such as using sequences to represent species or "operational taxonomic units" in a metagenomics study [Human Microbiome Project Consortium, 2012].

Redundancy can be handled by selecting a *representative subset* of the original data set. This representative subset is then used in downstream analysis. However, the task of choosing this subset can be computationally challenging because finding the best representative subset out of N items requires, in principle, considering all 2^N possible subsets.

In many fields, representative subsets are selected using some form of *submodular optimization*. The class of submodular functions is analogous to the class of convex functions. Submodular functions, however, are defined over subsets of a given set of data items, whereas convex functions are defined over real-valued vectors. Submodular functions are those that satisfy the property of diminishing returns. If we think of a function f as measuring the quality of a given subset, then the submodular property means that the incremental "value" of an item v decreases as the context in which v is considered grows. Formally, for any two subsets A and B of the set V, where $A \subseteq B \subset V$, and for any $v \in V \setminus B$, f is submodular if and only if it satisfies $f(A \cup \{v\}) - f(A) \geq f(B \cup \{v\}) - f(B)$. In many applications, it is common to search for a subset of maximal quality, as measured by a function f. The resulting optimization problem is hopelessly difficult for an arbitrary set function, but when the set function is submodular, the quality can be maximized within a constant factor of optimal in low-order polynomial time [Fisher et al., 1978, Nemhauser et al., 1978b, Minoux, 1978] (Methods). Moreover, these optimization algorithms are provably the best achievable in

polynomial time under some assumptions (Methods). For these reasons, submodular optimization has a long history in economics [Vives, 2001, Carter, 2001], game theory [Topkis, 1998, Shapley, 1971], combinatorial optimization [Edmonds, 1970, Lovász, 1983, Schrijver, 2004], electrical networks [Narayanan, 1997], and operations research [Cornunéjols et al., 1990]. Furthermore, submodular optimization has recently been used with great success for selecting representative subsets of text documents [Lin et al., 2007, Lin and Bilmes, 2011, 2012], recorded speech [Liu et al., 2013, Wei et al., 2013, 2014], machine translation [Kirchhoff and Bilmes, 2014], and images [Tschiatschek et al., 2014]. However, submodular optimization is not yet widely used in biology.

To illustrate how submodular optimization can be applied to biological data sets, we focus on the problem of choosing representative subsets from a database of protein sequences. Many methods have been developed for this problem [Hobohm et al., 1992, Holm and Sander, 1998, Li et al., 2001, Edgar, 2010, Parsons et al., 1992, Altschul et al., 1997, Enright and Ouzounis, 2000, Rice et al., 2000], including the popular CD-HIT [Weizhong and Adam, 2006] and PISCES [Wang and Dunbrack, Jr., 2003] algorithms. These sequence selection methods are very widely used—for example, the CD-HIT papers have been cited a total of >3,000 times (Google Scholar)—and are a standard preprocessing step applied to data sets of protein sequences, cDNA sequences and microbial DNA. All of these existing methods are based on the following simple algorithm: start with an empty set; order the sequences by length; then for each sequence, add the sequence to the set if no sequence currently in the set is more similar to it than some threshold (usually 90% sequence identity). Despite the ubiquity of this algorithm, it is simply a heuristic, apparently without any theoretical or empirical justification for its use over any other.

We propose a principled framework for representative protein sequence subset selection using submodular optimization. This approach involves defining a submodular objective function that quantifies the desirable properties of a given subset of sequences, and then applying a submodular optimization algorithm to choose a representative subset that maximizes this function. For illustrative purposes, we applied this optimization framework to protein domain sequences drawn from a single class ("Membrane and cell surface proteins and peptides") within the Structural Classification of Proteins—extended (SCOPe) database [Murzin et al., 1995]. The availability of protein structures

provide an orthogonal gold standard for judging the quality of a given representative subset. The results (Figure 6.1) suggest that the submodular approach does a good job of selecting protein sequences with diverse structural properties. In particular, among the nine folds that comprise our selected class, the submodular method successively chooses one exemplar sequence from fold before selecting two sequences from the same fold. The only exception is that the method chooses two "transmembrane beta-barrel" protein domains before selecting the first "single transmembrane helix" domain; however, as illustrated in Figure 6.1, this choice is not particularly surprising, given that the "transmembrane beta-barrel" fold is comprised of two superfamilies with quite different structural properties.

In the remainder of this paper, we expand upon this example, demonstrating that submodular optimization method does a better job of selecting representative protein sequence sets than competing methods, when evaluated relative to protein structure gold standards. We also demonstrate how optimization-based method development offers practical advantages over an approach in which the algorithm and the property it tries to optimize (its objective function) are inextricably intertwined. In particular, we demonstrate how a hybrid optimization function allows us to design a method that excels at selecting either large or small representative sets.

Note that the task of representative subset selection is distinct from that of clustering. Clustering focuses on sets of similar sequences, whereas representative subset selection focuses on the individual sequences chosen to represent a larger set. A large number of methods have been proposed for protein sequence clustering, including those based on Markov clustering [Enright et al., 2002], connected component detection [Enright and Ouzounis, 2000], generative modeling [Yang and Wang, 2003], minimal spanning trees [Guan and Du, 1998], and spectral clustering [Paccanaro et al., 2006]. Applying these methods to representative subset selection requires specifying, in addition, a procedure for selecting an exemplar sequence from each cluster. In practice, clustering methods are rarely used for reducing redundancy in protein sequence sets. Nonetheless, we empirically demonstrate below that our submodular selection approach outperforms several commonly used clustering algorithms.

6.2 Methods

6.2.1 Submodularity

The property of *submodularity* is important for the optimization of the set functions defined below. A *submodular* function [Fujishige, 2005] is defined as follows: given a finite set $V = \{1, 2, \ldots, n\}$, a discrete set function $f : 2^V \rightarrow \mathbb{R}$ is submodular if and only if:

$$f(S \cup \{v\}) - f(S) \geq f(T \cup \{v\}) - f(T), \quad \forall S \subseteq T \subseteq V, v \notin T. \tag{6.1}$$

In other words, the incremental gain of adding item s to the set decreases when the set to which s is added to grows from S to T. Intuitively, most reasonable measures of the quality or "informativeness" of a set of items are submodular because the gain from adding a given item is reduced when the set already contains other items similar to it.

Two other properties of set functions are relevant to their optimization. First, a set function f is defined as *monotone non-decreasing* if

$$f(S \cup \{v\}) - f(S) \geq 0, \quad \forall v \in V \setminus S, S \subseteq V. \tag{6.2}$$

Second, we say that f is *normalized* if $f(\emptyset) = 0$.

6.2.2 Similarity functions

The measures of subset quality described below are based on pairwise similarities of sequences. Such similarity measures are real-valued functions $s(u, v)$ defined on pairs of sequences. We considered two measures to quantify the similarity between a pair sequences: (1) the fraction of residues that match between the two sequences (fraction identical), and (2) the similarity measure used by the Rankprop method, defined as $\exp(-\text{E-value}/100)$ [Weston et al., 2004]. Both measures have been used successfully in previous work on protein sequences, and can be calculated efficiently using an index-based method like BLAST [Altschul et al., 1990]. In both cases, we define the

similarity between all pairs of sequences not reported by BLAST as 0. Note that, due to differing sequence lengths, both measures are not symmetric.

6.2.3 Subset quality measures

Let V be the full set of sequences and let $s(u, v)$ be a measure of similarity between a pair of sequences $u, v \in V$. A subset quality function is defined over subsets $f(A)$, where $A \subseteq V$. We consider several such subset quality functions, each of which measures some property of a set of sequences. In each case, a higher value of the function is desired.

First, we define the *facility-location* function as

$$f_{\text{facility-location}}(A) \triangleq \frac{1}{|V|} \sum_{v \in V} \max_{a \in A} s(a, v). \tag{6.3}$$

Intuitively, this function takes a high value when every sequence in V has at least one similar representative in A. This function is also the objective function of the k-medoids clustering method. The facility-location function is submodular, normalized and monotone non-decreasing.

Second, we define the *sum-redundancy* function as

$$f_{\text{sum-redundancy}}(A) \triangleq \kappa_{\text{sum}} - \sum_{a_1, a_2 \in A} s(a_1, a_2), \tag{6.4}$$

where $\kappa_{\text{sum}} = \sum_{a,b \in V} s(a, b)$. Intuitively, for sets A of a given size $|A|$, this function takes a very small value when A includes many pairs of similar sequences, and it takes a large value when all the sequences in A are very dissimilar from one another. In other words, the sum-redundancy function penalizes the redundancy in A. Optimizing the sum-redundancy function with the monotone greedy algorithm (defined below) is identical to the commonly-used representative subset heuristic of iteratively choosing the example that is most dissimilar from all previously-chosen examples. The sum-redundancy is submodular and normalized, but not monotone non-decreasing (in fact, it is monotone non-increasing).

Third, we define the *max-redundancy* function as

$$f_{\text{max-redundancy}}(A) \triangleq \kappa_{\text{max}} - \sum_{a_1 \in A} \max_{a_2 \in A, a_2 \neq a_1} s(a_1, a_2). \tag{6.5}$$

where $\kappa_{\text{max}} = \sum_{a_1 \in V} \max_{a_2 \in V, a_2 \neq a_1} s(a_1, a_2)$. The max-redundancy function is very similar to sum-redundancy, except that it penalizes only the most similar neighbor for each sequence in A instead of all neighbors. The max-redundancy is submodular and normalized, but not monotone non-decreasing and (like above) is monotone non-increasing.

6.2.4 Optimization algorithms

We are interested in choosing a subset of sequences A that maximizes a given measure of quality of the subset. Because all of the set functions defined above are either monotone increasing or monotone decreasing in the size of the set, optimizing the function over all subsets would result in either the empty set or the full set, depending on the function. Therefore, we additionally control the size of the returned set using one of two strategies. In the first setting, we constrain the size of the returned set to a specific size k and solve the optimization problem

$$\begin{aligned} &\text{maximize } f(A) \\ &\text{subject to } |A| = k. \end{aligned} \tag{6.6}$$

In the second setting, we add a term to the objective that is proportional to the size of the subset, scaled by a hyperparameter λ, and we optimize the unconstrained function

$$\text{maximize } f(A) + \lambda |A|. \tag{6.7}$$

We use the constrained strategy when using the monotone greedy algorithm (defined below) because that algorithm supports such constraints. The bidirectional greedy algorithm (defined below) works only for unconstrained problems, so we use the second strategy for that algorithm, varying λ to

indirectly control the resultant size. In practice, if a particular subset size (or some other feature of the set such as minimum objective value) is desired, this can be achieved using binary search.

Maximizing an arbitrary set function in general requires considering all $2^{|V|}$ possible subsets and is therefore impractical for large sets of sequences. In fact, even when this set function is submodular, maximizing it exactly is NP-hard. However, submodular functions can be efficiently approximated within a constant factor of optimal. In this work we apply the following two optimization algorithms.

The first algorithm is the *monotone greedy* algorithm (Supplementary Algorithm 3). At each iteration, the monotone greedy algorithm computes the difference in the objective function obtained by adding each sequence and adds the sequence with the largest difference. It turns out that when the objective function is submodular, monotone nondecreasing and normalized, this simple algorithm is guaranteed to produce a solution within a factor of $1 - 1/e \approx 0.63$ of optimal [Nemhauser et al., 1978b]. This is the best approximation ratio achievable in polynomial time unless P=NP [Feige, 1998]. Furthermore, this algorithm can be sped up using the fact that the submodular property guarantees that, if adding a sequence v to A would result in a difference of d in the objective, then adding v to a larger set $A \cup B$ will result in an objective difference $d' \leq d$. This property allows the algorithm to skip considering many sequences, and in practice often results in near-linear running time [Minoux, 1978].

The second algorithm is the *bidirectional greedy* algorithm (Supplementary Algorithm 4). This algorithm, which is randomized, maintains a growing set, which is initialized to the empty set, and a shrinking set, which is initialized to the full set. It considers each sequence v in turn, and either adds v to the growing set or removes it from the shrinking set randomly with probability proportional to the gain in the objective resulting from either change, respectively. At the end of the algorithm, the growing set and shrinking set are identical, and this set is returned. When the objective function is submodular and normalized (but not necessarily monotone nondecreasing), the mean of the objective values found over many randomized runs of this algorithm is guaranteed to be at least $\frac{1}{2}$ of optimal [Buchbinder et al., 2012]. This is the best approximation ratio achievable in polynomial time (independent of whether P=NP) [Feige et al., 2011].

In the experiments below, we applied the monotone greedy algorithm to optimize the (mono-

tone) facility-location function ($1 - 1/e$ approximation guarantee). We applied the bidirectional greedy algorithm optimize the (nonmonotone) sum-redundancy and max-redundancy functions ($1/2$ approximation guarantee). The bidirectional greedy algorithm can be used on mixtures of these functions to maintain a $1/2$ approximation guarantee. However, we achieved better results on the development data set using the greedy algorithm to optimize these mixtures (no theoretical guarantee; Supplementary Figure 6.13), so we used this strategy in our subsequent experiments.

Note that the 1/2 approximation guarantee can be maintained by running both algorithms and choosing the result with the better objective value. Moreover, although the greedy algorithm does not have a mathematical guarantee for non-monotone problems, we have found that for many problems it nonetheless performs better than the mathematically-guaranteed bidirectional greedy algorithm (results not shown). As future work, we are interested in investigating mathematical properties of these functions that may explain why the greedy algorithm is performing better than the current theory predicts.

6.2.5 Comparison to threshold algorithm (CD-HIT)

We call the algorithm used by most previous sequence subset selection methods, including CD-HIT and PISCES, the *threshold* algorithm (Supplementary Algorithm 2). This algorithm starts with an empty return set. It considers each sequence in decreasing order of sequence length, adding a given sequence to the return set if there is no sequence currently in the return set with similarity greater than some threshold τ. The value of τ controls the size of subset returned: a small τ results in a small set, while a large τ results in a large set.

In order to facilitate comparison, we implemented our own version of CD-HIT. The primary difference between our implementation and the CD-HIT source is that our implementation uses pre-computed BLAST alignments, while the CD-HIT source uses an on-the-fly index-based alignment method similar to BLAST. The main advantage of this on-the-fly method is that it does not require an expensive database construction preprocessing step. However, when comparing multiple methods on a single data set, it is much more efficient to perform a single alignment step. Using a single set of alignments also removes the possibility for performance differences due to slightly different

alignment results. Note that the on-the-fly method could in principle be used in the context of either the CD-HIT or submodular optimization. Our implementation of CD-HIT and the CD-HIT source result in nearly-identical subsets (Supplementary Note 1).

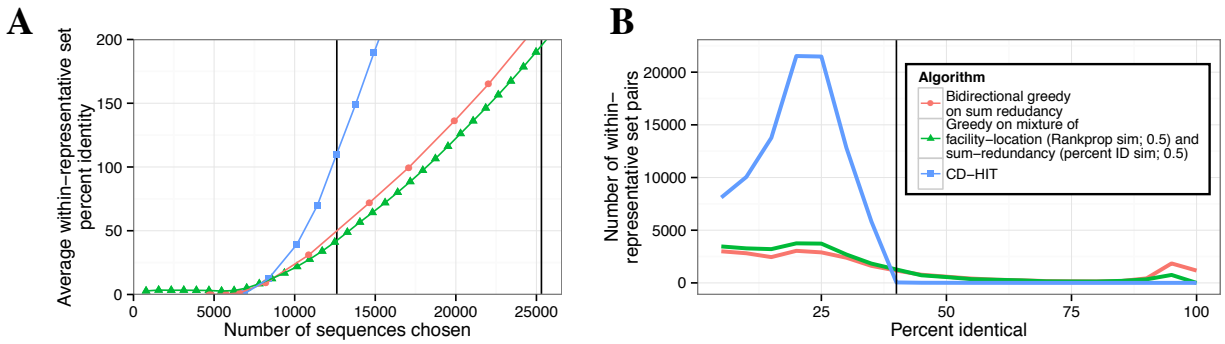
6.2.6 Protein sequence data

To test these methods, we downloaded the full Structural Classification Of Proteins–extended (SCOPe) database version 15 from `http://scop.berkeley.edu` [Murzin et al., 1995]. SCOPe is a hierarchical structural classification database, consisting of four levels with increasing granularity: class, fold, superfamily, and family. In this work, we focus primarily on the family and superfamily levels because these represent a reasonable level of granularity for subset selection.

We also downloaded the ASTRAL database from `http://scop.berkeley.edu` [Brenner et al., 2000]. ASTRAL is a database of protein domain sequences, some of which are associated with SCOPe categories. We used only ATOM-derived sequence records, and we chose the subset of ASTRAL sequences that are associated with a SCOPe family, removing eight sequences that were labeled with multiple SCOPe families. This resulted in a set of 78,048 sequences.

We also downloaded the results from all-by-all BLAST on the ASTRAL sequences from `http://scop.berkeley.edu`. BLAST was run using the command line `blastpgp -d DBNAME -i SEQ.fa -j 1 -h 1e-2 -v 5000 -b 5000 -a2 -z 100000000 -J T -O OUTPUT`. This results in all pairwise relationships with an E-value less than 0.01, where the similarity of each pair is measured by an E-value and a fraction sequence identity.

In order to overfitting to SCOPe, we performed all development on just the 16,880 ASTRAL sequences with the SCOPe Class "All beta proteins." Unless otherwise noted, the figures show performance on all of ASTRAL; we computed this performance only when preparing this manuscript.

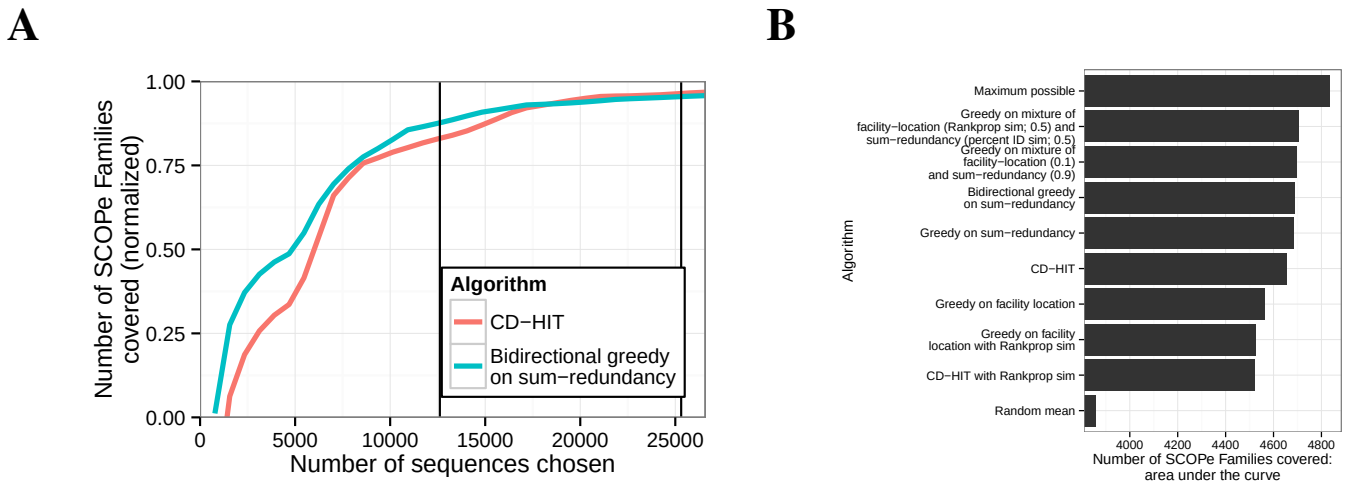


Supplementary Figure 6.2: (A) Average redundancy in representative sets. The vertical axis is calculated for a representative set A as $(1/|A|)\sum_{a_1,a_2 \in A} s(a_1, a_2)$, where $s(a_1, a_2)$ is percent identity. Vertical lines correspond to the sizes of subsets selected by CD-HIT using 40% and 90% sequence identity thresholds. (B) Histogram of pairwise similarities within representative sets. We produced subsets of size 12,614 sequences (the size reported by CD-HIT with a 40% identity threshold) for all methods by finding the smallest subset larger than the target size and randomly removing sequences to reach this size. Vertical axis indicates the number of pairs of sequences with a given percent identity within each size-12,614 representative set, in 5% bins. Vertical line indicates 40% identity threshold used by CD-HIT.

6.3 Results

6.3.1 Submodular optimization effectively eliminates redundancy in sequence data sets

To evaluate how well submodular optimization can remove redundancy in protein sequence data sets, we applied it to ASTRAL sequences. We focus on the sum-redundancy function with percent identity objective function because this function is most comparable to CD-HIT and it performed best in our later evaluation (see below). As expected, submodular optimization produced representative subsets with less total redundancy (Figure 6.2A). This because submodular optimization is a theoretically-based algorithm that explicitly aims to minimize redundancy, while the threshold algorithm is a heuristic that lacks any guarantee. The improvement primarily comes from the fact that the threshold method chooses many pairs of sequences with identity just under the threshold, whereas optimizing sum-redundancy chooses very few sequences with any detectable identity at the cost of choosing a small number of closely-related pairs (Figure 6.2B). The small number of closely-related pairs that



Supplementary Figure 6.3: Coverage of SCOPe families. (A) The vertical axis indicates is the number of families with at least one representative, normalized to the range $[0, 1]$ using the formula $f'(x) = \frac{f(x)-E(x)}{M(x)-E(x)}$, where x is a subset size, $f(x)$ is the value for the method in question on that subset size, $E(x)$ is expected (random mean) and $M(x)$ is maximum possible (defined as $M(x) = x$ if $x < F$, and $M(x) = F$ otherwise, where F = 4,994 is the total number of SCOPe families). Vertical lines correspond to the sizes of subsets selected by CD-HIT using 40% and 90% sequence identity thresholds. (B) Area under the curve comparison. Bars indicate selection algorithms, ordered by their area under the curve. Horizontal axis indicates area under the "family" coverage curve, calculated as follows. Let V be the full set of sequences, $i \in 1 \ldots |V|$ be a subset size, m be a particular method, $A_i^{(m)}$ be the subset of size i chosen by m, and $f(A)$ be the number of SCOPe Families covered by A. Area under the curve $= \frac{1}{|V|} \sum_{i=1}^{|V|} f(A_i^{(m)})$. For subset sizes for which we did not compute a subset, $f()$ is imputed with linear interpolation.

are accepted is likely due to the randomized nature of the algorithm.

6.3.2 Representative subsets chosen by submodular optimization exhibit high structural diversity

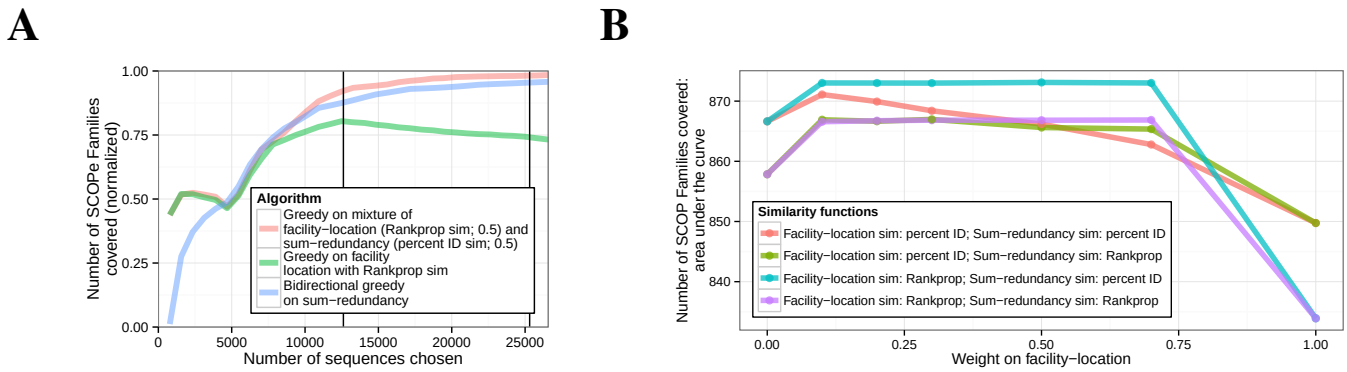
In order to evaluate the quality of a subset of sequences independently of a specific objective function, we compared these subsets to the Structural Classification Of Proteins–extended (SCOPe) database. SCOPe contains 78,048 protein domain sequences, each assigned to one of 4,994 protein families. These assignments are based on experimentally determined protein structures, so they are independent from the sequence-based information used by the optimization methods. Intuitively, a

good representative subset should include one sequence from each family, whereas a poor subset may include many sequences from the same family while leaving other families with no representatives. Therefore, we defined the quality of a subset as the number of SCOPe protein families with at least one sequence from that family in the subset.

Among the three submodular objective functions that we considered, we found that optimizing the sum-redundancy function produced the best results on this evaluation metric. Our evaluation considered 12 possible approaches: two submodular optimization algorithms on three different objective functions, each computed using two different measures of sequence similarity. To avoid overfitting to SCOPe, we performed all development and comparison of objective functions on a subset of the data ($\sim$21%) and applied only the three best methods to the full data set. Optimizing sum-redundancy with the *bidirectional greedy algorithm* results in much better coverage of SCOPe families than the threshold heuristic method that is employed by methods such CD-HIT (Figure 6.3). For representative sets of size 12,614 (the size chosen by CD-HIT with threshold 40%), the set chosen by submodular optimization leaves 27% fewer SCOPe families uncovered (276 compared with 379) than the threshold method does. We reached a similar conclusion when performing the evaluation on SCOPe superfamilies and folds rather than families (Supplementary Figures 6.8-6.13).

6.3.3 A mixture of submodular objective functions outperforms any single component objective

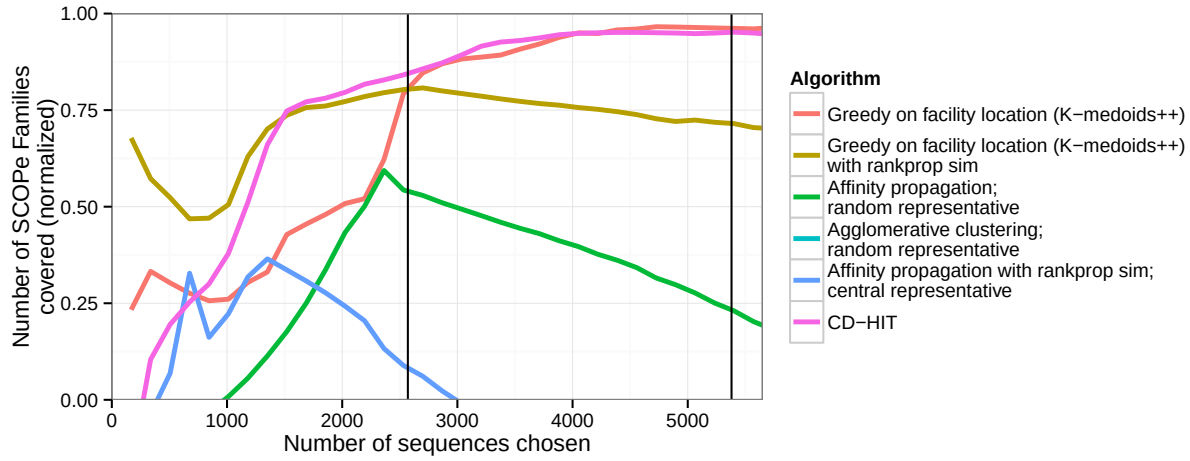
Although optimizing sum-redundancy performs best on the SCOPe-based evaluation metric for large subsets, we found that optimizing a different function resulted in better performance for very small subsets (<5,000 sequences; Figure 6.4A). Specifically, this good performance resulted from (1) optimizing a function called *facility-location* instead of sum-redundancy, and (2) using the similarity function used by the Rankprop method [Weston et al., 2004] that takes into account the statistical confidence of an alignment, instead of percent identity. The facility-location function is defined as $f_{\text{facility-location}}(A) \triangleq \frac{1}{|V|}\sum_{v \in V} \max_{a \in A} s(a, v)$ where V is the set of all sequences. Intuitively, this function takes a high value when every sequence in V has at least one similar representative in A. This function makes up for a weakness of redundancy-minimizing objectives such as sum-redundancy: such functions cannot differentiate representative sets with no detectable



Supplementary Figure 6.4: (A) Same as 6.3A, but including mixture objectives. (B) Comparison of mixture weights. Each mixture objective is a combination of facility-location and sum-redundancy objectives, each with either a percent ID or Rankprop similarity function. Horizontal axis indicates weight on facility-location (weight on sum-redundancy equals one minus the weight on facility-location). Line color indicates mixture objective. Vertical axis indicates area under the curve for SCOPe family coverage (see Supplementary Figure 6.10). To avoid overfitting, the statistics in (A) were computed on the subset of sequences with the SCOPe Class "All beta proteins".

similarity between any representatives. In contrast, facility-location can differentiate sets of unrelated representatives by how well they cover unpicked sequences.

These observations suggest that sum-redundancy with the percent identical similarity metric and facility-location with the Rankprop similarity metric may provide different types of information. To test this hypothesis, we optimized a mixed objective function using the greedy algorithm and evaluated the resulting representative subsets against the SCOPe evaluation metric. We found that this mixture performed much better on SCOPe coverage than either function individually, both for small and large subsets (Figures 6.3B and 6.4A). Moreover, this performance is quite insensitive to mixture weight (Figure 6.4B). These results show that a mixture of objective functions can better represent the quantity of interest than a single function. Optimizing mixtures of objective functions this way is very natural using the submodular optimization framework, but would be very difficult in a non-optimization-based framework.



Supplementary Figure 6.5: Performance of clustering methods repurposed for representative set selection. Plot is same as Figure 6.3, but includes clustering-based methods instead of submodular optimization-based. Results are computed on our development set, composed of the subset of the data ($\sim$21%) with the SCOPe Class "All beta proteins".

6.3.4 Representative subset selection methods perform better than repurposed clustering methods

An alternative to applying a representative subset selection method is to apply a clustering method and then choose a representative from each identified cluster. Using the SCOPe evaluation metric, we compared representative subset selection using submodular optimization to eight variants of this clustering-based approach. We compared two similarity functions: percent identity and Rankprop; three clustering methods: k-medoids++, affinity propagation and agglomerative hierarchical clustering; and two methods for choosing a representative from each cluster: "random" and "central". Central representative selection chooses the cluster member with the highest total similarity to the other cluster members (i.e. it optimizes a facility-location objective for a single item). Random representative selection chooses a random member. Note that k-medoids++ is identical to optimizing facility-location with the submodular greedy algorithm.

We found that none of these clustering methods performed as well as either the submodular optimization-based methods or the threshold heuristic (Figure 6.5 and Supplementary Figure 6.13). The exception, as noted above, is that optimizing facility-location with the Rankprop similarity

metric performs well for small representative sets. This is likely because clustering methods aim to optimize the quality of the entire cluster, whereas representative set selection methods focus solely on identifying good representatives.

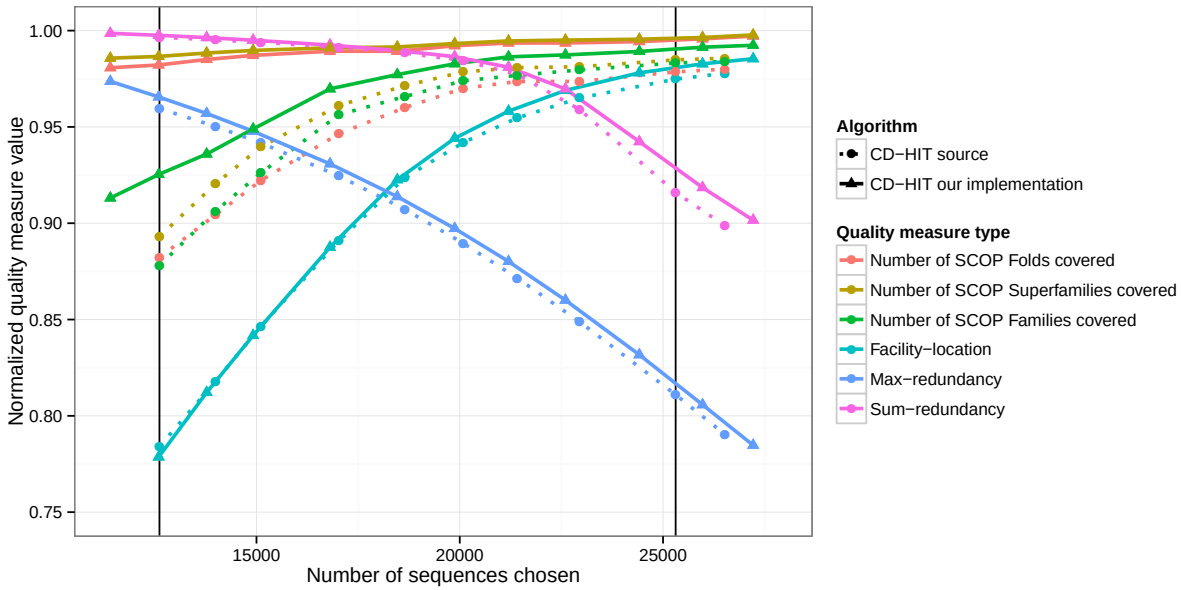
6.4 Discussion

The submodular optimization framework we present here results in theoretically guaranteed performance in some circumstances, performs well in practice, and has the flexibility to express a wide variety of desired properties of subsets of sequences. This approach is therefore preferable to existing, widely used, heuristic methods for selecting representative subsets of biological sequences. In this work we have explored only a small number of submodular functions; the submodular optimization framework supports many other objective functions that may provide even better results, including those based on cluster coverage, those based on the sequences directly rather than alignments, and others. Because the SCOPe evaluation function is itself submodular, a submodular objective function could also be efficiently learned from training data to correlate well with it. Moreover, this application may serve as a model for how submodular optimization can be applied to other discrete problems in biology, such as selecting representative sets of protein structures, genomic loci, or mass spectra.

Acknowledgements This work was supported by the National Institutes of Health award R01 CA180777.

Supplementary Note 1: Implementation of threshold algorithm

In order to verify that our implementation of the threshold algorithm matches existing packages, we ran our implementation of the threshold algorithm and the CD-HIT packages on the same data set. The methods use slightly different sequence alignments, so we cannot expect them to choose exactly the same subsets. Instead, we compared the values of relevant summary statistics: our three submodular objective functions and SCOPe coverage. As expected, we found that the methods compared very similarly (SI Appendix, Fig. 6.6). Both implementations performed nearly identically on the three submodular objective functions, with a slight advantage to our implementation due



Supplementary Figure 6.6: Comparison of the CD-HIT source and our implementation of the same algorithm. Horizontal axis indicates number of sequences in a given subset, focused on the region between 40% and 90% percent ID thresholds (horizontal black lines indicates these thresholds). Line color indicates quality measure; vertical axis indicates quality value. All quality measures are linearly transformed to fall between zero and one. Solid lines with triangles indicate performance with our implementation; dotted lines with circles indicate the CD-HIT source.

to the fact that these objectives and our implementation use the same sequence alignments. Our implementation performed somewhat better on SCOPe coverage, perhaps due to more accurate sequence alignments. Importantly, our implementation performed at least as well or better on all metrics.

Supplementary Figures/Algorithms

Algorithm 2 Threshold algorithm. V is the set of sequences and $s : V \times V \to \mathbb{R}$ is a similarity function over V. $N : V \to 2^V$ is the set of neighbors of v with nonzero similarity. τ is a hyperparameter threshold determining the size of the returned subset (low τ results in a small subset).

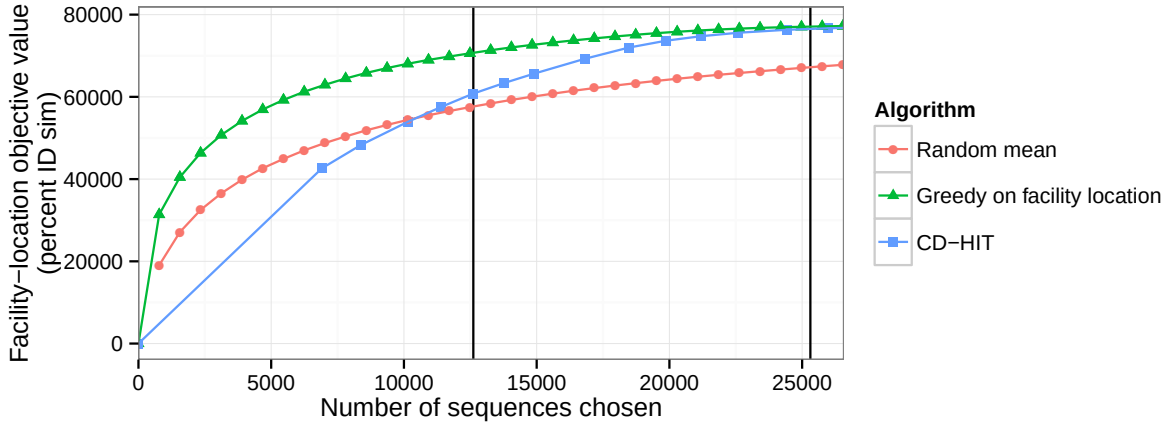
function THRESHOLD(V, s)
    Order V into a list $(v_1, v_2, \ldots, v_n)$ in decreasing order of sequence length.
    $A \leftarrow \{v_1\}$
    for $i = 2 \ldots N$ **do**
        $m \leftarrow \max\{s(v_i, v') : v' \in A \cap N(v_i)\}$
        if $m < \tau$ **then**
            $A \leftarrow A \cup \{v_i\}$
        end if
    end for
    return A
end function

Algorithm 3 Monotone greedy submodular maximization algorithm for optimizing a function $f(A)$. V is the set of sequences, and k is the size of subset to return.

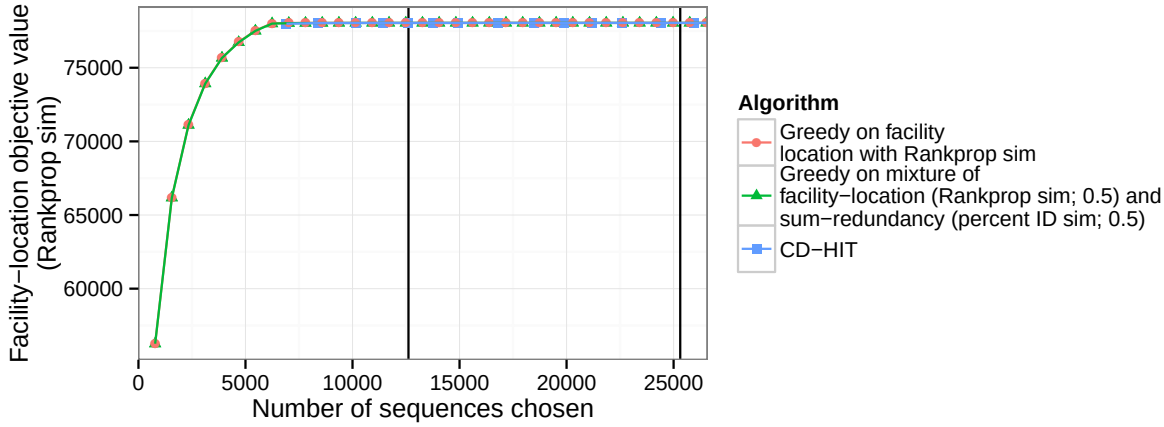
function MONOTONE_GREEDY(V, f)
    $A \leftarrow \{\}$
    for $1 \ldots k$ **do**
        for $v \in V \setminus A$ **do** ▷ Can be accelerated by considering only certain $v \in V$ (Methods).
            $\delta[v] \leftarrow f(A \cup \{v\}) - f(A)$
        end for
        $v^* \leftarrow \text{argmax}_{v \in V \setminus A} \delta[v]$
        $A \leftarrow A \cup \{v^*\}$
    end for
    return A
end function

Algorithm 4 Bidirectional greedy maximization algorithm for optimizing a function $f(A)$. V is the set of sequences.

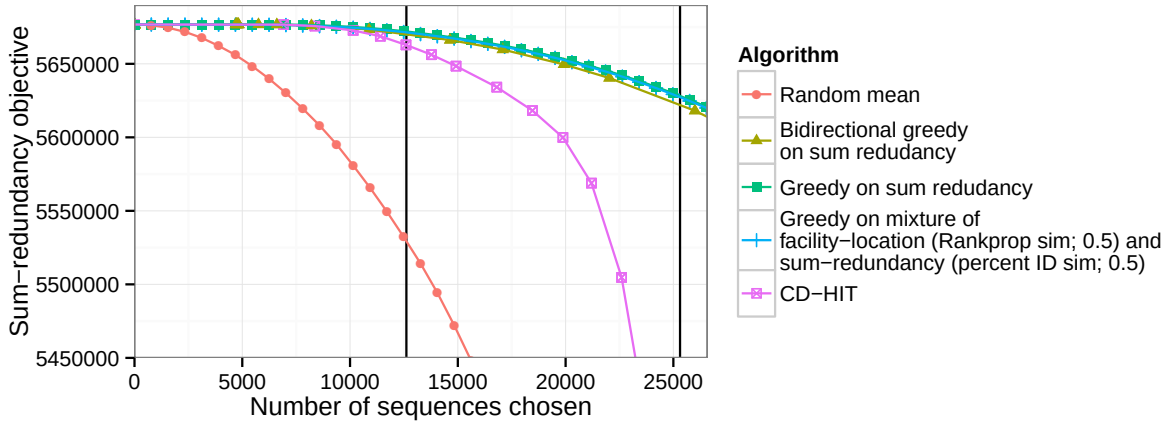
function BIDIRECTIONAL_GREEDY(V, f)
    $A \leftarrow \{\}, B \leftarrow V$
    for $v \in V$ **do**
        $\delta_A \leftarrow \max(0, f(A \cup \{v\}) - f(A))$
        $\delta_B \leftarrow \max(0, f(B \setminus \{v\}) - f(B))$
        if $\delta_A = \delta_B = 0$ **then**
            p = 1/2
        else
            p = $\delta_A/(\delta_A + \delta_B)$
        end if
        if random$(0, 1) \leq p$ **then**
            $A \leftarrow A \cup \{v\}$
        else
            $B \leftarrow B \setminus \{v\}$
        end if
    end for
    return A (or B, equivalently)
end function



(a)

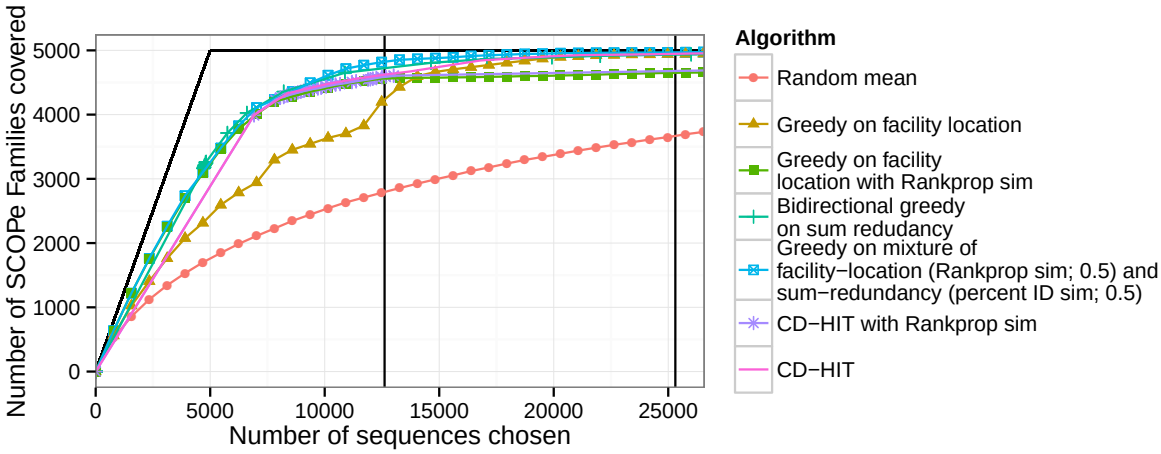


(b)

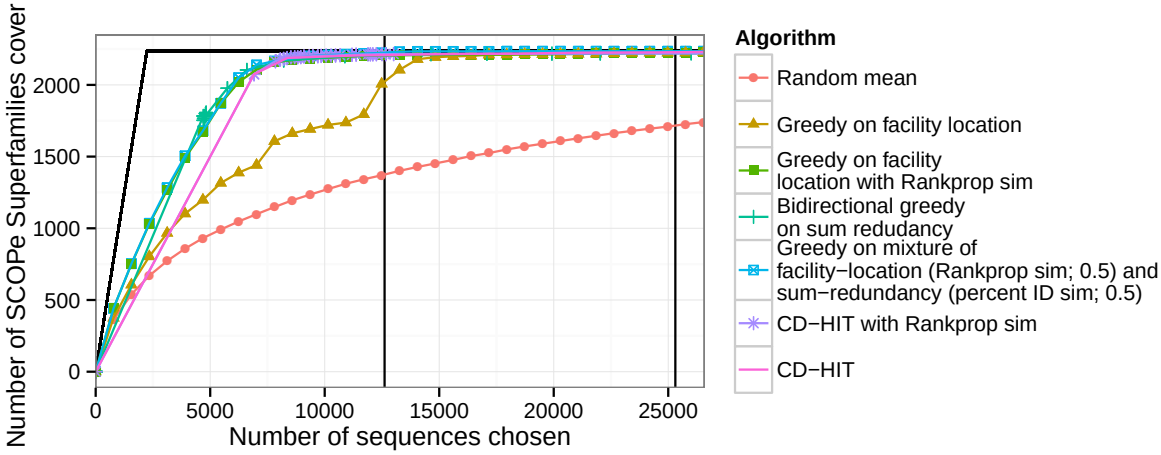


(c)

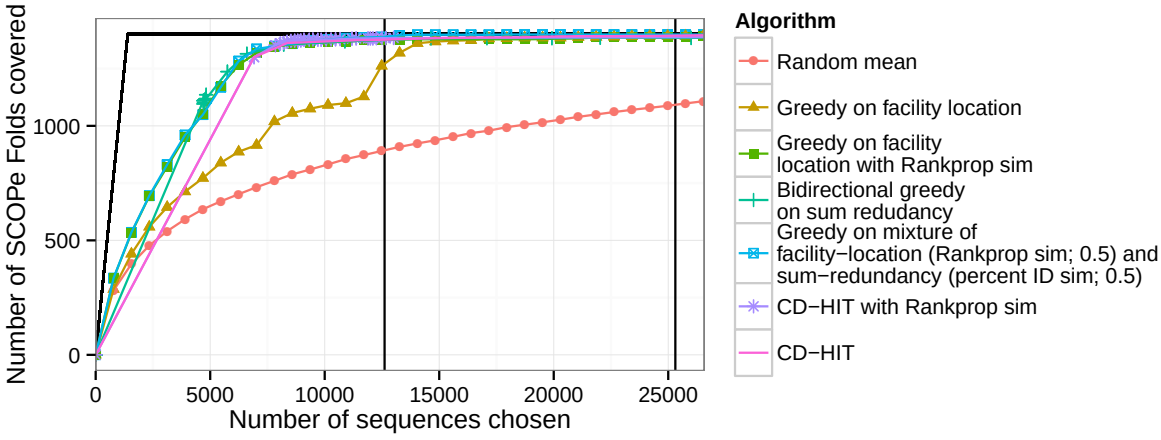
Supplementary Figure 6.7: Optimization performance on each submodular objective: (A) facility-location with percent ID sim; (B) facility-location with Rankprop sim; (C) sum-redundancy with percent ID sim. Horizontal axis indicates size of representative set. Vertical axis indicates facility-location objective value (higher is better). Color and point type indicates choice method. The highly similar lines in panel (B) result from the fact that the Rankprop similarity function has a small dynamic range (∼0.9-1.0).



(a)

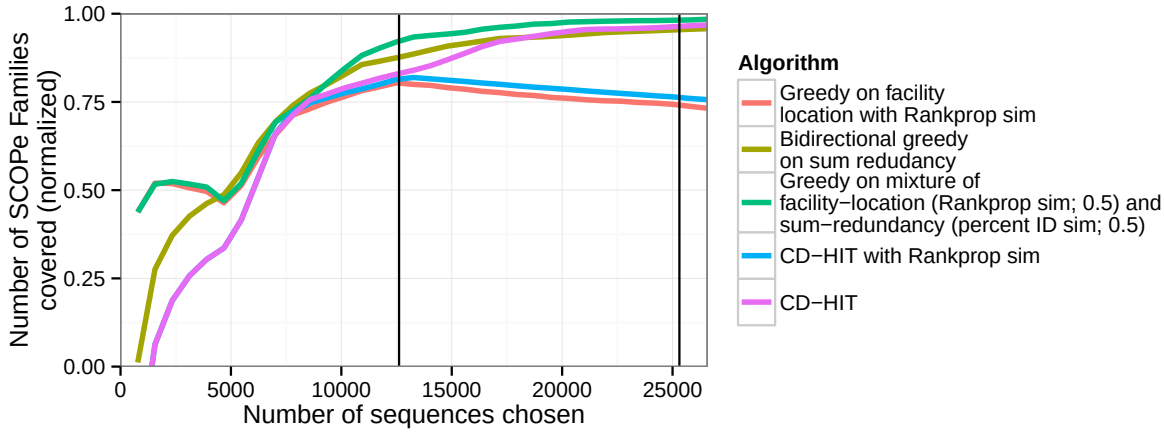


(b)

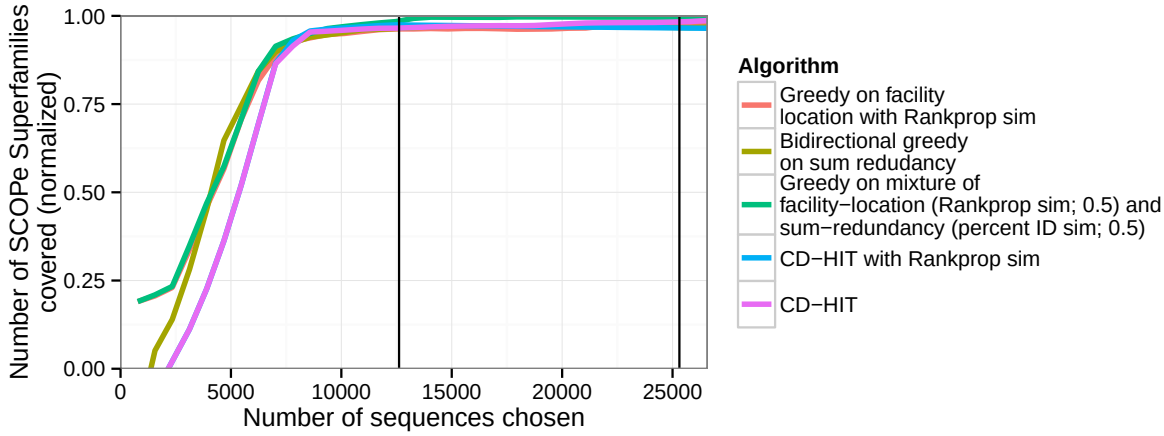


(c)

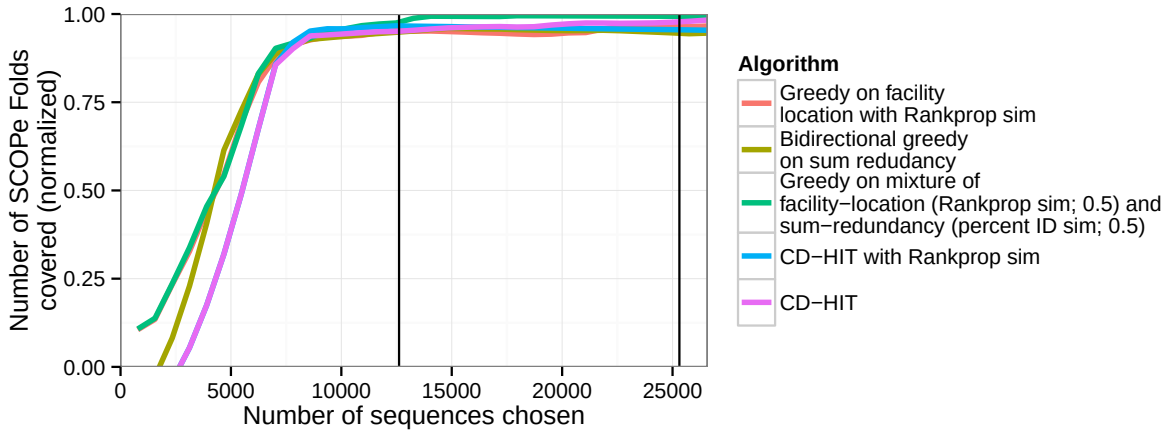
Supplementary Figure 6.8: Coverage of SCOPe categories: (A) Families; (B) Superfamilies, and (C) Folds. Horizontal axis indicates size of representative set. Vertical axis indicates number of SCOPe categories with at least one representative. Color and point type indicates choice method. Black curve indicates maximum possible performance. Vertical black lines indicate subset sizes reported by CD-HIT for thresholds of 40% and 90%.



(a)

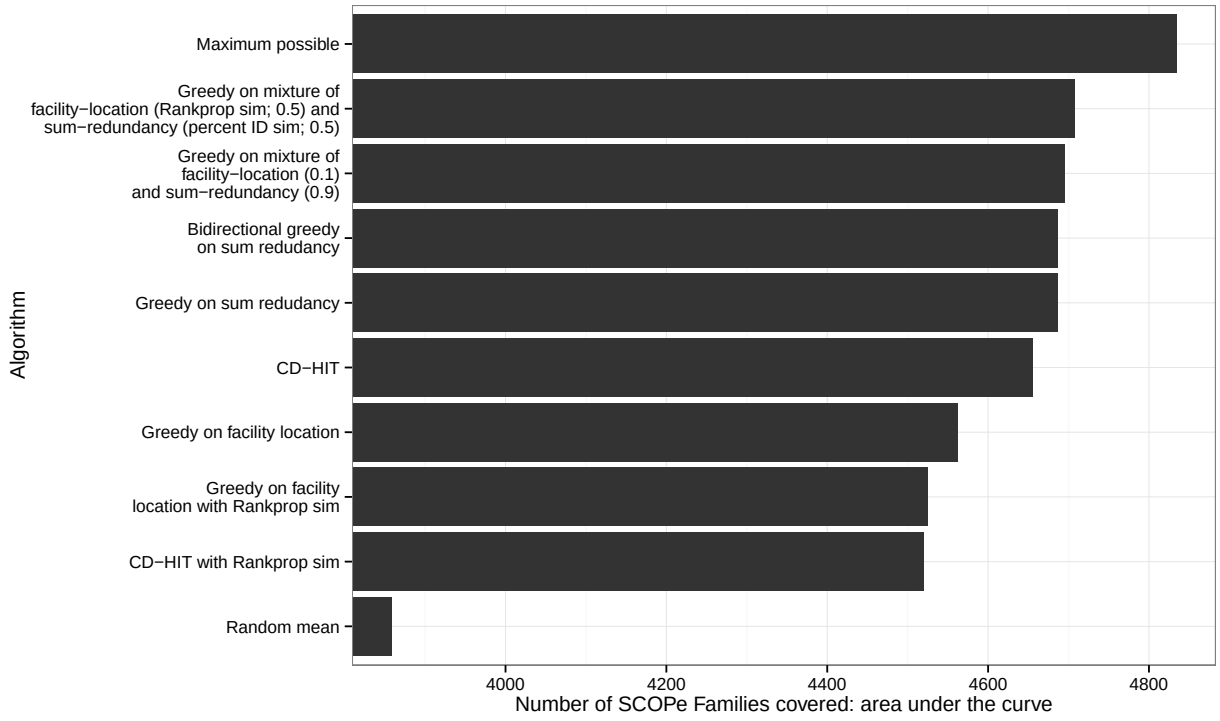


(b)

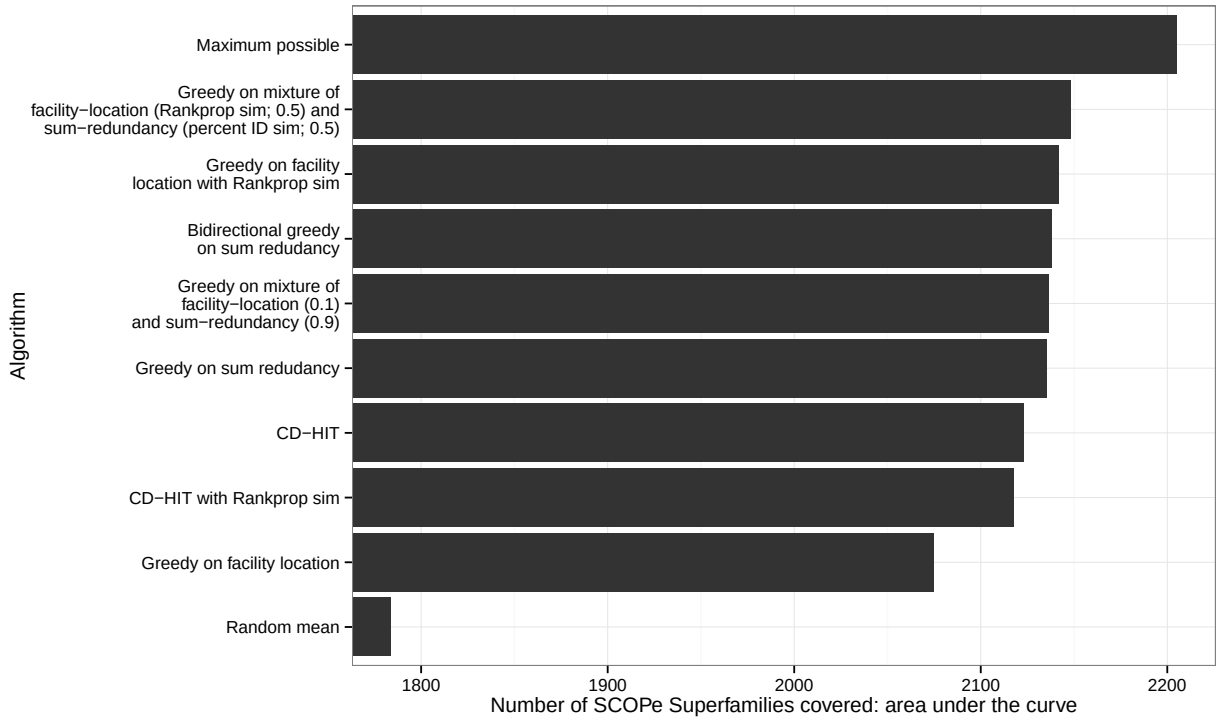


(c)

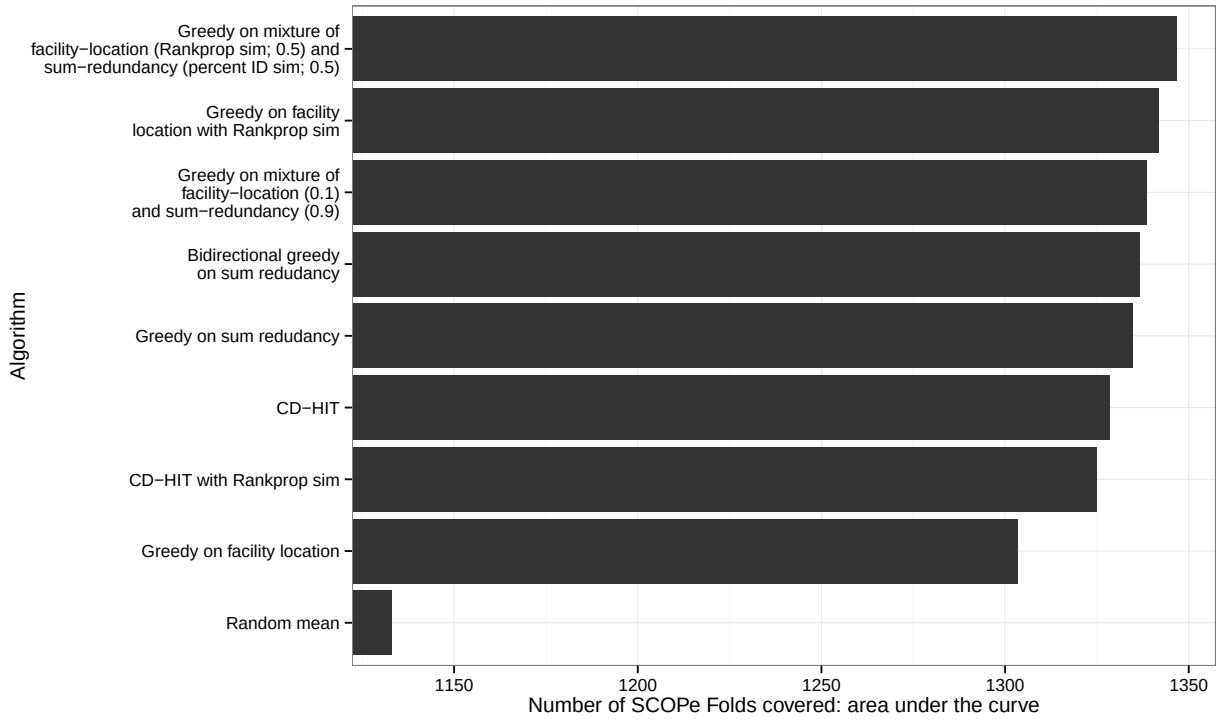
Supplementary Figure 6.9: Same as Supplementary Figure 6.8, but vertical axis is normalized as in Figure 2.



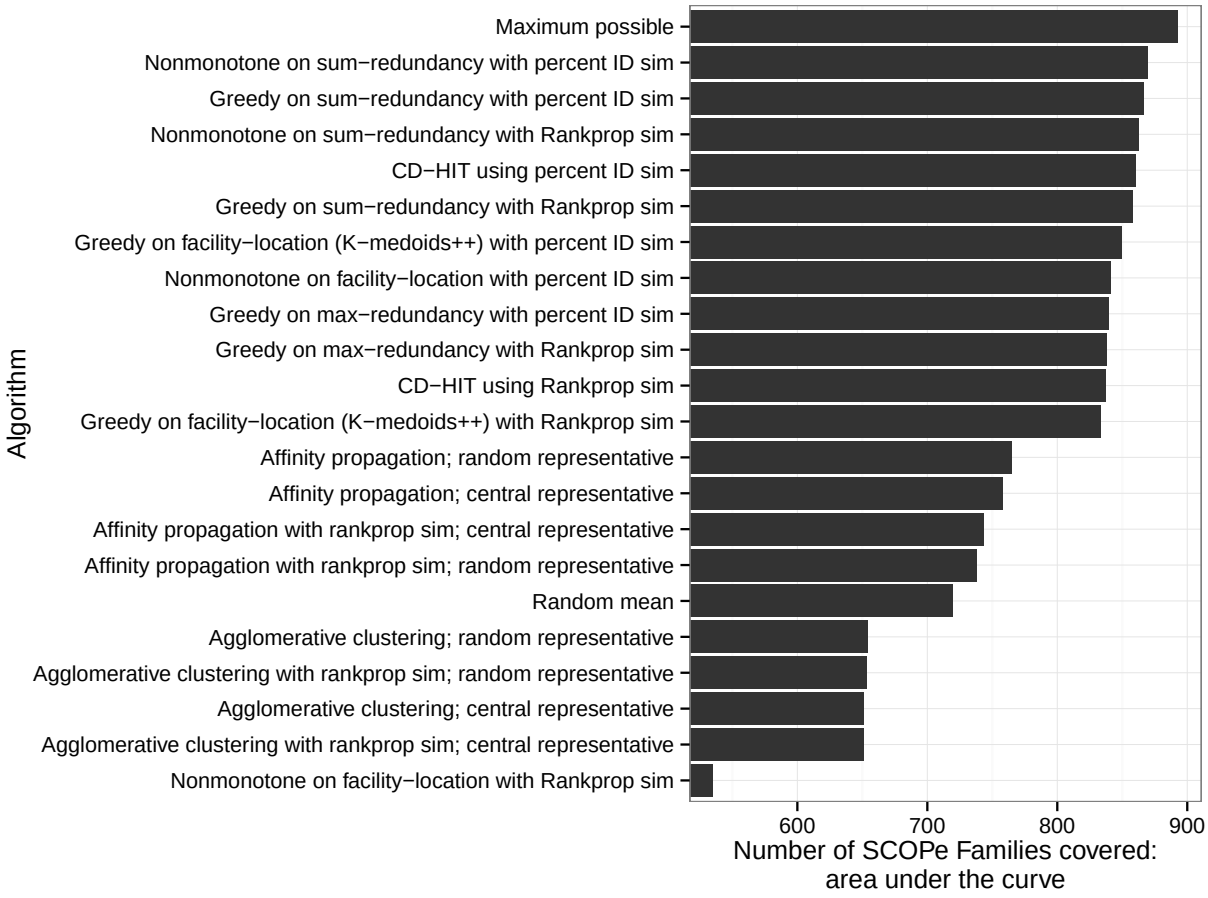
Supplementary Figure 6.10: Area under the curve for SCOPe Family coverage. Bars indicate selection algorithms, ordered by their area under the curve. Horizontal axis indicates area under the curve, calculated as follows. Let V be the full set of sequences, $i \in 1 \ldots |V|$ be a subset size, m be a particular method, $A_i^{(m)}$ be the subset of size i chosen by m, and $f(A)$ be the number of SCOPe Families covered by A. Area under the curve $= \frac{1}{|V|} \sum_{i=1}^{|V|} f(A_i^{(m)})$. For subset sizes for which we did not compute a subset, $f()$ is imputed with linear interpolation.



Supplementary Figure 6.11: Area under the curve for SCOPe Superfamily coverage. Plot is same as Supplementary Figure 6.10, except that horizontal axis indicates Superfamily coverage.



Supplementary Figure 6.12: Area under the curve for SCOPe Fold coverage. Plot is same as Supplementary Figure 6.10, except that horizontal axis indicates Fold coverage.



Supplementary Figure 6.13: Area under the curve for SCOPe Family coverage on development set. Plot is same as Supplementary Figure 6.10, except that results are computed on our development set, composed of the subset of the data (∼21%) with the SCOPe Class “All beta proteins”.

Chapter 7

CONCLUSION

In this conclusion section I propose some promising directions for future work in this field.

7.1 Genome annotation

Chapters 3 and 4 covered the topic of genome annotation using dynamic Bayesian networks. There are a number of directions in which these types of methods could be extended and improved.

7.1.1 Annotation with multiple length scales

The genome exhibits different patterns of activity at different length scales. Most research so far has focused on elements on the scale of 100-1000 bp, such as promoters and enhancers. At a finer scale, individual regulatory elements can be divided into individual TF binding sites (~ 10 bp) and these TF binding sites can be further subdivided into their characteristic DNase accessibility profile [Hesselberth et al., 2009], which vary at a single-base pair scale. At a coarser scale, I discussed in Chapter 4 how the genome is composed of five types of domains at the scale of 100-1000 kb. However, the SAGA approach in Chapter 4 models only these large-scale patterns. Ideally, a single model would be able to analyze multiple scales at the same time.

The only existing work that simultaneously models multiple scales is [Larson et al., 2013], in which researchers defined two random variables at each position: a fine-scale label (promoters, enhancers) and a coarse-scale label (active and inactive domains). The fine-scale label was allowed to vary every 200 bp; the large-scale label was allowed to vary only every 10 kb. Each large-scale label was associated with a certain probability distribution over fine-scale labels: for example, the coarse-scale "active domain" label would have a higher probability for the fine-scale "enhancer" label than would the coarse-scale "heterochromatin" label. However, this work had two disadvantages. First,

the model ended up learning near-uniform distributions for each coarse-scale label; this is likely due to the fact that the model included no explicit penalty for this behavior. Second, the model was an extension of ChromHMM [Ernst and Kellis, 2010], which has a minimum resolution of 200 bp, which prevented the authors from additionally adding a third scale at the resolution of $\sim$1 bp. It would be valuable to extend this approach to explicitly encourage low-entropy label distributions and adding an additional 1 bp resolution scale.

An alternate design for this model might instead associate each coarse-scale label with an entirely independent set of fine-scale labels. For example, a 100 bp-scale "enhancer" label might be composed of 1 bp-scale elements of "H3K27ac region upstream of enhancer TF binding site", "enhancer nucleosome free TF binding site" and "downstream eRNA transcribed region". This approach has the disadvantage that it results in more sub-labels to manually interpret, but it may give the model more power to uncover diverse types of elements.

7.1.2 Hierarchical label complexity

Our understanding of genome elements has multiple levels of complexity. At the simplest level, one might divide the genome into three categories: regulatory, transcribed, and inactive. Regulatory elements could be further divided into promoters and enhancers; enhancers could be divided into strong, weak and poised, and so on. Existing SAGA elements do not respect this hierarchy; they learn the parameters associated with each label independently.

A better SAGA method might learn a hierarchical tree of labels rather than a flat collection. Each node (or leaf, alternatively) in this tree would correspond to a set of emission parameters. Transition probabilities would be defined over the tree, such that nodes nearby in the tree have high transition probabilities to one another. This tree could be learned dynamically as part of training, by splitting nodes that are assigned to many diverse data patterns in the training data and merging nodes that learn redundant patterns.

Such a tree-based model would greatly aid interpretation of the learned parameters. For example, if the researcher is attempting to interpret a given node and has already verified that the nearby nodes correspond to known sub-compartments of an enhancer, it is very likely that the node in

question represents another (potentially novel) enhancer sub-component. In general, when trying to understand the biological function of a given pattern (i.e. node in the tree), a researcher can use the nearby nodes to guide their interpretation. Moreover, even without a tree structure, this splitting and vanishing strategy for initializing label parameters is likely to be less prone to local optima because labels would be initialized to data-dense parts of the parameter space (i.e. similar parameters to a label that was a assigned to many of positions), rather than random parameters.

7.1.3 Multiple cell types

There is a great deal of room for further work in simultaneously annotating multiple cell types. Two primary strategies have been "stacked" and "concatenated" annotation. In stacked annotation, instead of inputting data from just one cell type to the method, the user inputs all available data from all cell types. However, this approach is generally not effective because, in order to represent all the possible patterns of five activity types across ten cell types, it takes $5^{10} \approx 10^7$ labels, each of which must be interpreted independently. The second approach, "concatenated" annotation, applies the same model to a number of cell types. (The name concatenated comes from the fact that this strategy can be implemented by concatenating the data from each cell type as through it came from additional chromosomes of the first cell type.) Concatenated annotation works well when all the data has been produced consistently, but it has two disadvantages. First, it requires that all cell types have the same set of data available, which does not hold in practice. Second, artifactual experimental differences can result in learning cell type-biased labels. Several extensions have been proposed to these framework, such as the ones I developed in Chapter 4 using EGPR, but there are several other potential directions for improvement.

Latent activity mixture model. One way to better model activity in multiple cell types would be to learn "potential activity types" at each genomic position. For example, a given position might exhibit enhancer activity in 75% of cell types and be quiescent in 25%. In general, the model would learn a vector of probabilities over the different annotation labels at each position, and suppose that the annotation label in a given cell type is drawn from this distribution. Such a model

would discourage one cell type from calling a position "enhancer" (say) if no other cell types show evidence for having an enhancer in that position.

A model similar to this was proposed in [Biesinger et al., 2013]. These researchers defined a tree over cell types and modeled transition probabilities between neighboring cell types in the tree (in addition to the usual transition probabilities between neighboring genomic positions). However, this work had two disadvantages. First, like in [Larson et al., 2013], the model learned a near-uniform transition distribution, so the multiple cell type aspect of the model had little effect. Second, when this work was performed, there was only data available in nine human cell types. It would be valuable to extend this work to explicitly encourage low-entropy label distributions and to apply an approach like this to the hundreds of cell types that now have available data.

State space model. An alternative approach would be to take a "stacked" approach, but switch to a continuous (rather than discrete) model. Such a model would assign a vector of real values $x_1 \ldots x_k$, called *factors*, to each position instead of a discrete label. The expected emission for a given track t under this model would be $\sum_{i=1}^{k} x_i \mu_{it}$, where each μ_{it} is a learned parameter associated factor i and track t. This type of model is called a *state space* model, and can be optimized with dynamic programming approaches comparable to the forward-backward algorithm used for discrete HMMs. A factor in this model might represent, for example, "the extent to which this position is an enhancer active in mesodermal cells." The advantage of this model is that, because each position is modeled as a mixture of multiple types of activity, it would likely extend to hundreds of cell types more naturally than a discrete stacked approach. This type of model is similar to the class of methods known as "topic modeling" that have been successful in natural language tasks.

7.2 Statistical evaluation of differences between genomics assays

In order to understand genome activity, it is important to be able to ask whether or not a given drug, tissue, or disease produces a statistically significant change in activity relative to a particular cellular condition. This question is difficult to answer because the output of a genomics assay—the number of reads mapping to a given genomic position—has no natural units. In particular, a

difference between 0 and 30 reads may have a very different statistical importance from a difference between 1,000 and 1,030 reads. This property is due to a relationship between the mean of the data-generating process (i.e. the genomics assay) and its variance. Currently-used methods that do not model this mean-variance relationship, such as a Poisson model, result in poor evaluation of statistical differences. It would be valuable to develop methods to learn the mean-variance relationship of functional genomics data sets using the variation among biological replicates. Such a method would make it possible to produce a statistically principled measure of differences between functional genomics data sets. Such measurements would enable interrogation of the effects of cellular perturbations, yielding insights into the mechanisms of disease, aging and drug activity.

Learning the mean-variance relationship of these data sets has the additional benefit that it would make it possible to place genomics data in interpretable units. The lack of natural units for genomics data sets makes it difficult to analyze these data sets by eye and confounds any downstream analyses that does not internally model the mean-variance relationship (including SAGA methods). Currently, to attempt to handle this phenomenon, many analyses of genomics data sets transform the data using a log or inverse hyperbolic sine transform to attempt to normalize for the mean-variance relationship of the sequencing read counts. However, there is no theoretical basis for using these particular transformations rather than any other, so they almost certainly do not properly account for the mean-variance relationship as intended. Given a known mean-variance relationship, a data set can be put into interpretable units using the variance-stabilizing transformation $f(x) = \int \frac{1}{\sigma(u)} du$, where $\sigma(x)$ is the standard deviation of a variable with a mean of x. The resulting signal is in units of standard deviation, so it has the useful property that all data points have a 95% confidence intervals of $\sim$2 units. Therefore, variance-stabilized data sets are a natural by-product of a principled method for evaluating statistical differences. These data sets are much more easily analyzed by eye, and are necessary for any downstream analysis that does not internally model the mean-variance relationship.

7.3 Improved submodular objective for choosing panels of genomics assays

In Chapter 5, I proposed a method based on submodular optimization for choosing panels of assays to perform. This method was based around the facility location submodular objective function. This function determines the putative quality of a panel of assays that is optimized, and therefore is central to the behavior of the method. There are several potential improvements to the objective function I proposed.

First, the similarity function used by the facility location was defined as the pairwise correlation over all pairs of assays of each type. This calculation involves computing the correlation of assays performed in different cell types. This is not optimal, because even identical assay types might have low correlation when performed in very different cell types. However, it is impossible to directly compute the correlation between each pair of assay types without cross-cell type comparisons, because there is no one cell type in which all assay types have been performed. Instead, it may be better to compute correlations between the subset of pairs of assay types that have been performed in the same cell type, and then use a matrix completion method to impute rest of the correlations.

Second, the proposed method did not take into account our prior information about assay types, cell type or genomic positions. Taking into account this prior information might suggest different choices of assay types. For example, if assay type A is exactly predictable based on assay type B and the genome sequence, it does not make sense to include A and B in a panel together. Our prior information about the genome sequence can be incorporated into the submodular framework by including the genome sequence as a "free" assay performed in each cell type. Likewise, it might be possible to induce a similarity between ChIP-seq assays for transcription factors A and B by comparing the structure of the two proteins, even if ChIP-seq has never been performed targeting those factors. These types of prior knowledge were not taken into account by our previous method, which considered each assay as an abstract vector over the genome. Incorporating them would further improve the panels identified by this approach.

7.4 Future problems in computational genomics

Upcoming technologies for single-cell measurement, better measurement of genome conformation and genome editing promise to present new opportunities for computational methods. As with existing types of genomics data sets, analyzing such data requires methods that are scalable and that accurately model the underlying phenomena while remaining interpretable enough to afford biological understanding. In addition to their original purposes, the strategies presented in this dissertation may provide launching points to address these new problems.

BIBLIOGRAPHY

Waseem Akhtar, Johann de Jong, Alexey V Pindyurin, Ludo Pagie, Wouter Meuleman, Jeroen de Ridder, Anton Berns, Lodewyk FA Wessels, Maarten van Lohuizen, and Bas van Steensel. Chromatin position effects assayed by thousands of reporters integrated in parallel. *Cell*, 154(4): 914–927, 2013.

S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman. A basic local alignment search tool. *Journal of Molecular Biology*, 215:403–410, 1990.

S. F. Altschul, T. L. Madden, A. A. Schaffer, J. Zhang, Z. Zhang, W. Miller, and D. J. Lipman. Gapped BLAST and PSI-BLAST: A new generation of protein database search programs. *Nucleic Acids Research*, 25:3389–3402, 1997.

Y. Altun, M. Belkin, and D. A. Mcallester. Maximum margin semi-supervised learning for structured variables. In *Advances in neural information processing systems*, pages 33–40, 2005.

María E Andrés, Corinna Burger, María J Peral-Rubio, Elena Battaglioli, Mary E Anderson, Julia Grimes, Julia Dallman, Nurit Ballas, and Gail Mandel. Corest: a functional corepressor required for regulation of neural-specific gene expression. *Proceedings of the National Academy of Sciences*, 96(17):9873–9878, 1999.

R. Apweiler, T. K. Attwood, A. Bairoch, A. Bateman, E. Birney, M. Biswas, P. Bucher, L. Cerutti4, F. Corpet, M. D. R. Croning, R. Durbin, L. Falquet, W. Fleischmann, J. Gouzy, H. Hermjakob, N. Hulo, I. Jonassen, D. Kahn, A. Kanapin, Y. Karavidopoulou, R. Lopez, B. Marx, N. J. Mulder, T. M. Oinn, M. Pagni, F. Servant, C. J. A. Sigrist, and E. M. Zdobnov. The interpro database, an integrated documentation resource for protein families, domains and functional sites. *Nucleic Acids Research*, 29(1):37–40, 2001.

F. Ay, T. L. Bailey, and W. S. Noble. Statistical confidence estimation for Hi-C data reveals regulatory chromatin contacts. *Genome Research*, 24:999–1011, 2014a.

F. Ay, E. M. Bunnik, N. Varoquaux, S. M. Bol, J. Prudhomme, J.-P. Vert, W. S. Noble, and K. G. Le Roch. Three-dimensional modeling of the *P. falciparum* genome during the erythrocytic cycle reveals a strong connection between genome architecture and gene expression. *Genome Research*, 24:974–988, 2014b.

Jaume Bacardit and Xavier Llorà. Large-scale data mining using genetics-based machine learning. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 3(1):37–61, 2013.

T. L. Bailey and C. P. Elkan. The value of prior knowledge in discovering motifs with MEME. In C. Rawlings, D. Clark, R. Altman, L. C. Hunter, and L. C. Rawlings, editors, *Proceedings of the Third International Conference on Intelligent Systems for Molecular Biology*, pages 21–29. AAAI Press, 1995.

M. A. Beer and S. Tavazoie. Predicting gene expression from sequence. *Cell*, 117:185–198, 2004.

B. E. Bernstein, T. S. Mikkelsen, X. Xie, M. Kamal, D. J. Huebert, J. Cuff, B. Fry, A. Meissner, M. Wernig, K. Plath, R. Jaenisch, A. Wagschal, R. Feil, S. L. Schreiber, and E. S. Lander. A bivalent chromatin structure marks key developmental genes in embryonic stem cells. *Cell*, 125 (2):315–326, 2006.

Bradley E Bernstein, John A Stamatoyannopoulos, Joseph F Costello, Bing Ren, Aleksandar Milosavljevic, Alexander Meissner, Manolis Kellis, Marco A Marra, Arthur L Beaudet, Joseph R Ecker, Peggy J Farnham, Martin Hirst, Eric S Lander, Tarjei S Mikkelsen, and James A Thomson. The NIH Roadmap Epigenomics Mapping Consortium. *Nature Biotechnology*, 28(10):1045–1048, Oct 2010. doi: 10.1038/nbt1010-1045. URL http://dx.doi.org/10.1038/nbt1010-1045.

Wendy A Bickmore and Bas van Steensel. Genome architecture: Domain organization of interphase chromosomes. *Cell*, 152(6):1270–1284, 2013.

J. Biesinger, Y. Wang, and X. Xie. Discovering and mapping chromatin states using a tree hidden Markov model. *BMC Bioinformatics*, 14(Suppl 5):S4, 2013.

J. Bilmes and A. Subramanya. Parallel graph-based semi-supervised learning. In R. Bekkerman, M. Bilenko, and J. Langford, editors, *Scaling Up Machine Learning*, chapter 15, pages 307—330. Cambridge UP, 2012.

Jeff Bilmes. Dynamic Bayesian multinets. In *UAI '00: Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence*, pages 38–45, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc. ISBN 1-55860-709-9.

Jeff Bilmes and Richard Rogers. The Graphical Models Toolkit GMTK source distribution, 2015. https://melodi.ee.washington.edu/gmtk.

C. Bishop. *Neural Networks for Pattern Recognition*. Oxford UP, Oxford, UK, 1995.

B. E. Boser, I. M. Guyon, and V. N. Vapnik. A training algorithm for optimal margin classifiers. In D. Haussler, editor, *5th Annual ACM Workshop on COLT*, pages 144–152, Pittsburgh, PA, 1992. ACM Press.

E. I. Boyle, S. Weng, J. Gollub, H. Jin, D. Botstein, J. M. Cherry, and G. Sherlock. Go::TermFinder – open source software for accessing Gene Ontology information and finding significantly enriched Gene Ontology terms associated with a list of genes. *Bioinformatics*, 20(18):3710–3715, 2004.

S. E. Brenner, P. Koehl, and M. Levitt. The ASTRAL compendium for sequence and structure analysis. *Nucleic Acids Research*, 28:254–256, 2000.

M. Brown, R. Hughey, A. Krogh, I. Mian, K. Sjolander, and D. Haussler. Using Dirichlet mixture priors to derive hidden Markov models for protein families. In C. Rawlings, editor, *Proceedings of the First International Conference on Intelligent Systems for Molecular Biology*, pages 47–55. AAAI Press, 1993.

N. Buchbinder, M. Feldman, J. S. Naor, and R. Schwartz. Submodular maximization with cardinality constraints. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1433–1452. SIAM, 2014.

Niv Buchbinder, Moran Feldman, Joseph Naor, and Roy Schwartz. A tight linear time (1/2)-approximation for unconstrained submodular maximization. In *Foundations of Computer Science (FOCS), 2012 IEEE 53rd Annual Symposium on*, pages 649–658. IEEE, 2012.

P. Bucher. Weight matrix description of four eukaryotic RNA polymerase II promoter elements derived from 502 unrelated promoter sequences. *Journal of Molecular Biology*, 4(20):563–578, 1990.

B. Burgess-Beusse, C. Farrell, M. Gaszner, M. Litt, V. Mutskov, F. Recillas-Targa, M. Simpson, A. West, and G. Felsenfeld. The insulation of genes from external enhancers and silencing chromatin. *Proceedings of the National Academy of Sciences of the United States of America*, 99 (Suppl 4):16433, 2002.

M. Carter. *Foundations of Mathematical Economics*. The MIT Press, 2001.

Lyubomira Chakalova, Emmanuel Debrand, Jennifer A Mitchell, Cameron S Osborne, and Peter Fraser. Replication and transcription: Shaping the landscape of the genome. *Nature Reviews Genetics*, 6(9):669–677, August 2005.

O. Chapelle, B. Schölkopf, and A. Zien, editors. *Semi-supervised Learning*. MIT Press, Cambridge, MA, 2006.

Jacob Cohen. Weighted kappa: Nominal scale agreement provision for scaled disagreement or partial credit. *Psychological bulletin*, 70(4):213, 1968.

G. Cornunéjols, G. L. Nemhauser, and L. A. Wolsey. The uncapacitated facility location problem. In P.B. Mirchandani and R.L. Franci, editors, *Discrete Location Theory*, chapter 3. Wiley/Interscience, New York, 1990.

M. Craven and J. Bockhorst. Markov networks for detecting overlapping elements in sequence data. *Advances in Neural Information Processing Systems*, 17:193, 2005.

I. Csiszár and G. Tusnády. Information geometry and alternating minimization procedures. *Statistics and decisions*, page 205, 1984.

G. Cuellar-Partida, F. Buske, R. McLeay, T. Whitington, W. S. Noble, and T. L. Bailey. Epigenetic priors for identifying active transcription factor binding sites. *Bioinformatics*, 28:56–62, 2011.

D. Das and S. Petrov. Unsupervised part-of-speech tagging with bilingual graph-based projections. In *NAACL*, pages 600–609, 2011.

D. Das and N.A. Smith. Semi-supervised framesemantic parsing for unknown predicates. In *Proc. of ACL*, 2011.

J. Davis and M. Goadrich. The relationship between precision-recall and ROC curves. In *Proceedings of the International Conference on Machine Learning*, 2006.

N. Day, A. Hemmaplardh, R. E. Thurman, J. A. Stamatoyannopoulos, and W. S. Noble. Unsupervised segmentation of continuous genomic data. *Bioinformatics*, 23(11):1424–1426, 2007.

S. Degroeve, B. De Baets, Y. Van de Peer, and P. Rouz. Feature subset selection for splice site prediction. *Bioinformatics*, 18:S75–S83, 2002.

J. Dekker, K. Rippe, M. Dekker, and N. Kleckner. Capturing chromosome conformation. *Science*, 295(5558):1306–1311, 2002.

Job Dekker. Gene regulation in the third dimension. *Science*, 319(5871):1793–1794, 2008. doi: 10.1126/science.1152850. URL http://www.sciencemag.org/content/319/5871/1793.abstract.

J. R. Dixon, S. Selvaraj, F. Yue, A. Kim, Y. Li, Y. Shen, M. Hu, J. S. Liu, and B. Ren. Topological domains in mammalian genomes identified by analysis of chromatin interactions. *Nature*, 485 (7398):376–380, 2012.

R. C. Edgar. Search and clustering orders of magnitude faster than BLAST. *Bioinformatics*, 19 (2460–2461), 2010.

J. Edmonds. Matroids, submodular functions, and certain polyhedra. *Combinatorial Structures and Their Applications*, pages 69–87, 1970.

ENCODE Project Consortium. An integrated encyclopedia of DNA elements in the human genome. *Nature*, 489:57–74, 2012.

A. J. Enright and C. A. Ouzounis. GeneRAGE: a robust algorithm for sequence clustering and domain detection. *Bioinformatics*, 16(5):451–457, 2000.

A. J. Enright, S. Van Dongen, and C. A. Ouzounis. An efficient algorithm for large-scale detection of protein families. *Nucleic Acids Research*, 30(7):1575–1584, 2002.

P. Erdős and A. Rényi. On the evolution of random graphs. *Magyar Tud. Akad. Mat. Kutató Int. Közl*, 5:17–61, 1960.

J. Ernst and M. Kellis. Discovery and characterization of chromatin states for systematic annotation of the human genome. *Nature Biotechnology*, 28(8):817–825, 2010.

J. Ernst and M. Kellis. ChromHMM: automating chromatin-state discovery and characterization. *Nat Methods*, 9(3):215–216, 2012.

Jason Ernst and Manolis Kellis. Large-scale imputation of epigenomic datasets for systematic annotation of diverse human tissues. *Nature biotechnology*, 33(4):364–376, 2015.

U. Feige. A threshold of *ln n* for approximating set cover. *Journal of the ACM*, 45(4):634–652, 1998.

Uriel Feige, Vahab S Mirrokni, and Jan Vondrak. Maximizing non-monotone submodular functions. *SIAM Journal on Computing*, 40(4):1133–1153, 2011.

G. J. Filion, J. G. van Bemmel, U. Braunschweig, W. Talhout, J. Kind, L. D. Ward, W. Brugman, I. J. de Castro, R. M. Kerkhoven, H. J. Bussemaker, and B. van Steensel. Systematic protein location mapping reveals five principal chromatin types in *Drosophila* cells. *Cell*, 143(2):212–224, 2010.

D. Filippova, R. Patro, G. Duggal, and C. Kingsford. Identification of alternative topological domains in chromatin. *Algorithms Mol Biol.*, 9:14, 2014.

M.L. Fisher, G.L. Nemhauser, and L.A. Wolsey. An analysis of approximations for maximizing submodular set functions—II. *Polyhedral combinatorics*, pages 73–87, 1978.

A. G. Fraser and E. M. Marcotte. A probabilistic view of gene function. *Nature Genetics*, 36(6): 559–564, 2004.

N. Friedman. Inferring cellular networks using probabilistic graphical models. *Science*, 303(5659): 799–805, 2004.

N. Friedman, M. Linial, I. Nachman, and D. Pe'er. Using Bayesian networks to analyze expression data. *Journal of Computational Biology*, 7(3-4):601–620, 2000.

S. Fujishige. *Submodular functions and optimization*, volume 58. Elsevier Science, 2005.

K. Ganchev, J. Graça, J. Gillenwater, and B. Taskar. Posterior regularization for structured latent variable models. *Journal of Machine Learning Research*, 11:2001–2049, 2010.

Gene Ontology Consortium. Gene ontology: tool for the unification of biology. *Nature Genetics*, 25(1):25–9, 2000a.

Gene Ontology Consortium. Gene ontology: tool for the unification of biology. *Nature Genetics*, 25(1):25–29, 2000b.

Enrico Glaab, Jaume Bacardit, Jonathan M Garibaldi, and Natalio Krasnogor. Using rule-based machine learning for candidate disease gene prioritization and sample classification of cancer gene expression data. *PloS one*, 7(7):e39932, 2012.

R. Gomes, A. Krause, and P. Perona. Discriminative clustering by regularized information maximization. In *Advances in Neural Information Processing Systems*, 2010.

Xiaojun Guan and Lei Du. Domain identification by clustering sequence alignments. *Bioinformatics*, 14(9):783–788, 1998.

L. Guelen, L. Pagie, E. Brasset, W. Meuleman, M. B. Faza, W. Talhout, B. H. Eussen, A. de Klein, L. Wessels, W. de Laat, and B. van Steensel. Domain organization of human chromosomes revealed by mapping of nuclear lamina interactions. *Nature*, 453(7197):948–951, 2008.

Thomas Hamelryck. Probabilistic models and machine learning in structural bioinformatics. *Statistical methods in medical research*, 2009.

Jennifer Harrow, Adam Frankish, Jose M Gonzalez, Electra Tapanari, Mark Diekhans, Felix Kokocinski, Bronwen L Aken, Daniel Barrell, Amonida Zadissa, Stephen Searle, et al. GENCODE: the reference human genome annotation for The ENCODE Project. *Genome Research*, 22(9):1760–1774, 2012.

T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data mining, Inference and Prediction*. Springer, New York, NY, 2001.

L. He, J. Gillenwater, and B. Taskar. Graph-based posterior regularization for semi-supervised structured prediction. In *Seventeenth Conference on Computational Natural Language Learning*, 2013.

N. Heintzman, R. Stuart, G. Hon, Y. Fu, C. Ching, R. D. Hawkins, L. Barrera, S. Van Calcar, C. Qu, K. Ching, W. Wang, Z. Weng, R. Green, G. Crawford, and B. Ren. Distinct and predictive chromatin signatures of transcriptional promoters and enhancers in the human genome. *Nature Genetics*, 39:311–318, 2007.

J. Hesselberth, X. Chen, Z. Zhang, P. J. Sabo, R. Sandstrom, A. P. Reynolds, R. E. Thurman, S. Neph, M. S. Kuehn, W. S. Noble, S. Fields, and J. A. Stamatoyannopoulos. Global mapping of

protein-DNA interactions *in vivo* by digital genomic footprinting. *Nature Methods*, 6(4):283–289, 2009.

J. W. K. Ho, T. Liu, Y. L. Jung, B. H. Alver, S. Lee, K. Ikegami, K. Sohn, A. Minoda, M. Y. Tolstorukov, A. Appert, S. C. J. Parker, T. Gu, A. Kundaje, N. C. Riddle, E. Bishop, T. A. Egelhofer, S. S. Hu, A. A. Alekseyenko, A. Rechtsteiner, D. Asker, J. A. Belsky, S. K. Bowman, Q. B. Chen, R. A. Chen, D. S. Day, Y. Dong, A. C. Dosé, X. Duan, C. B. Epstein, S. Ercan, E. A. Feingold, J. M. Garrigues, N. Gehlenborg, P. J. Good, P. Haseley, D. He, M. Herrmann, M. M. Hoffman, T. E. Jeffers, P. V. Kharchenko, P. Kolasinska-Zwierz, C. V. Kotwaliwale, N. Kumar, S. A. Langley, E. N. Larschan, I. Latorre, M. W. Libbrecht, X. Lin, R. Park, M. J. Pazin, H. N. Pham, A. Plachetka, B. Qin, Y. B. Schwartz, N. Shoresh, P. Stempor, A. Vielle, C. Wang, C. M. Whittle, H. Xue, R. E. Kingston, J. H. Kim, B. E. Bernstein, A. F. Dernburg, V. Pirrotta, M. I. Kuroda, W. S. Noble, T. D. Tullius, M. Kellis, D. M. MacAlpine, S. Strome, S. C. R. Elgin, J. Ahringer, X. S. Liu, G. H. Karpen, J. D. Lieb, and P. J. Park. Comparative analysis of metazoan chromatin architecture. Submitted., 2013.

Uwe Hobohm, Michael Scharf, Reinhard Schneider, and Chris Sander. Selection of representative protein data sets. *Protein Science*, 1(3):409–417, 1992.

M. M. Hoffman, O. J. Buske, J. Wang, Z. Weng, J. A. Bilmes, and W. S. Noble. Unsupervised pattern discovery in human chromatin structure through genomic segmentation. *Nature Methods*, 9(5):473–476, 2012.

M. M. Hoffman, J. Ernst, S. P. Wilder, A. Kundaje, R. S. Harris, M. Libbrecht, B. Giardine, P. M. Ellenbogen, J. A. Bilmes, E. Birney, R. C. Hardison, I. Dunham, M. Kellis, and W. S. Noble. Integrative annotation of chromatin elements from ENCODE data. *Nucleic Acids Res*, 41(2): 827–41, 2013.

L. Holm and C. Sander. Removing near-neighbour redundancy from large protein sequence collections. *Bioinformatics*, 14(5):423–429, 1998.

Human Microbiome Project Consortium. Structure, function and diversity of the healthy human microbiome. *Nature*, 486:207–214, 2012.

M. Imakaev, G. Fudenberg, R. P. McCord, N. Naumova, A. Goloborodko, B. R. Lajoie, J. Dekker, and L. A. Mirny. Iterative correction of Hi-C data reveals hallmarks of chromosome organization. *Nat Methods*, 9:999–1003, 2012.

T. Jaakkola and D. Haussler. Exploiting generative models in discriminative classifiers. In *Advances in Neural Information Processing Systems 11*, San Mateo, CA, 1998. Morgan Kauffmann.

R. Jaschek and A. Tanay. Spatial clustering of multivariate genomic and epigenomic information. In *Proceedings of the Thirteenth Annual International Conference on Computational Molecular Biology*, volume 5541, pages 170–183, 2009.

T. Joachims. Transductive inference for text classification using support vector machines. In I. Bratko and S. Dzeroski, editors, *Proceedings of the Sixteenth International Conference on Machine Learning*, pages 200–209, San Francisco, CA, 1999. Morgan Kaufmann.

Norman L Johnson. Systems of frequency curves generated by methods of translation. *Biometrika*, pages 149–176, 1949.

M. I. Jordan. Why the logistic function? a tutorial discussion on probabilities and neural networks. Technical report, Massachusetts Institute of Technology, 1995.

Hanna Julienne, Azedine Zoufir, Benjamin Audit, and Alain Arneodo. Human genome replication proceeds through four chromatin states. *PLoS Computational Biology*, 9(10):e1003233, 2013.

R. Karlic, H. R.Chung, J. Lasserre, K. Vlahovicek, and M. Vingron. Histone modification levels are predictive for gene expression. *Proceedings of the National Academy of Sciences of the United States of America*, 107(7):2926–2931, 2010.

E. Keogh and A. Mueen. *Encyclopedia of Machine Learning*, chapter Curse of Dimensionality, pages 257–258. Springer, 2011.

P. V. Kharchenko, A. A. Alekseyenko, Y. B. Schwartz, A. Minoda, N. C. Riddle, J. Ernst, P. J. Sabo, E. Larschan, A. A. Gorchakov, T. Gu, D. Linder-Basso, A. Plachetka, G. Shanower, M. Y. Tolstorukov, L. J. Luquette, R. Xi, Y. L. Jung, R. W. Park, E. P. Bishop, T. P. Canfield, R. Sandstrom, R. E. Thurman, D. M. MacAlpine, J. A. Stamatoyannopoulos, M. Kellis, S. C. R. Elgin, M. I. Kuroda, V. Pirrotta, G. H. Karpen, and P. J. Park. Comprehensive analysis of the chromatin landscape in *Drosophila melanogaster*. *Nature*, 471:480–485, 2010. doi: 10.1038/nature09725.

M. Kircher, D. M. Witten, P. Jain, B. J. O'Roak, G. M. Cooper, and J. Shendure. A general framework for estimating the relative pathogenicity of human genetic variants. *Nature Genetics*, 46(3):310–315, 2014. **Uses a machine learning approach to estimate pathogenicity of genetic variants using a framework that takes advantage of the fact that natural selection removes deleterious variation.**

K Kirchhoff and J Bilmes. Submodularity for data selection in machine translation. In *Empirical Methods in Natural Language Processing (EMNLP)*, 2014.

Timo JT Koski and John Noble. A review of bayesian networks and structure learning. *Mathematica Applicanda*, 40(1):51–103, 2012.

A. Krause, A. Singh, and C. Guestrin. Near-optimal sensor placements in gaussian processes: Theory, efficient algorithms and empirical studies. *The Journal of Machine Learning Research*, 9: 235–284, 2008.

Harold W Kuhn. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955.

Anshul Kundaje, Wouter Meuleman, Jason Ernst, Misha Bilenky, Angela Yen, Alireza Heravi-Moussavi, Pouya Kheradpour, Zhizhuo Zhang, Jianrong Wang, Michael J Ziller, et al. Integrative analysis of 111 reference human epigenomes. *Nature*, 518(7539):317–330, 2015.

Monika Lachner, Roderick J O'Sullivan, and Thomas Jenuwein. An epigenetic road map for histone lysine methylation. *Journal of Cell Science*, 116(11):2117–2124, 2003.

G. R. G. Lanckriet, T. De Bie, N. Cristianini, M. I. Jordan, and W. S. Noble. A statistical framework for genomic data fusion. *Bioinformatics*, 20(16):2626–2635, 2004.

Stephen G Landt, Georgi K Marinov, Anshul Kundaje, Pouya Kheradpour, Florencia Pauli, Serafim Batzoglou, Bradley E Bernstein, Peter Bickel, James B Brown, Philip Cayting, Yiwen Chen, Gilberto Desalvo, Charles Epstein, Katherine I Fisher-Aylor, Ghia Euskirchen, Mark Gerstein, Jason Gertz, Alexander J Hartemink, Michael M Hoffman, Vishwanath R Iyer, Youngsook L Jung, Subhradip Karmakar, Manolis Kellis, Peter V Kharchenko, Qunhua Li, Tao Liu, X. Shirley Liu, Lijia Ma, Aleksandar Milosavljevic, Richard M Myers, Peter J Park, Michael J Pazin, Marc D Perry, Debasish Raha, Timothy E Reddy, Joel Rozowsky, Noam Shoresh, Arend Sidow, Matthew Slattery, John A Stamatoyannopoulos, Michael Y Tolstorukov, Kevin P White, Simon Xi, Peggy J Farnham, Jason D Lieb, Barbara J Wold, and Michael Snyder. ChIP-seq guidelines and practices of the ENCODE and modENCODE consortia. *Genome Research*, 22(9):1813–1831, Sep 2012. doi: 10.1101/gr.136184.111. URL http://dx.doi.org/10.1101/gr.136184.111.

Jessica L Larson, Curtis Huttenhower, John Quackenbush, and Guo-Cheng Yuan. A tiered hidden markov model characterizes multi-scale chromatin states. *Genomics*, 102(1):1–7, 2013.

C. Leslie, E. Eskin, and W. S. Noble. The spectrum kernel: A string kernel for SVM protein classification. In R. B. Altman, A. K. Dunker, L. Hunter, K. Lauderdale, and T. E. Klein, editors, *Proceedings of the Pacific Symposium on Biocomputing*, pages 564–575, New Jersey, 2002. World Scientific.

Bing Li, Michael Carey, and Jerry L Workman Workman. The role of chromatin during transcription. *Cell*, 128(4):707–719, 2007.

W. Li, L. Jeroszewski, and A. Godzik. Clustering of highly homologous sequences to reduce the size of large protein databases. *Bioinformatics*, 17(3):282–283, 2001.

H. Lian, W. Thompson, R. E. Thurman, J. A. Stamatoyannopoulos, W. S. Noble, and C. Lawrence. Automated mapping of large-scale chromatin structure in ENCODE. *Bioinformatics*, 24(17): 1911–1916, 2008.

M. Libbrecht, F. Ay, M. M. Hoffman, D. M. Gilbert, J. A. Bilmes, and W. S. Noble. Joint annotation of chromatin state and chromatin conformation reveals relationships among domain types and identifies domains of cell-type-specific expression. *Genome Research*, 25(4):544–557, 2015.

E. Lieberman-Aiden, N. L. van Berkum, L. Williams, M. Imakaev, T. Ragoczy, A. Telling, I. Amit, B. R. Lajoie, P. J. Sabo, M. O. Dorschner, R. Sandstrom, B. Bernstein, M. A. Bender, M. Groudine, A. Gnirke, J. Stamatoyannopoulos, L. A. Mirny, E. S. Lander, and J. Dekker. Comprehensive mapping of long-range interactions reveals folding principles of the human genome. *Science*, 326 (5950):289–293, 2009.

G. Lin, M. K. Chawla, K. Olson, C. A. Barnes, J. F. Guzowski, C. Bjornsson, W. Shain, and B. Roysam. A multi-model approach to simultaneous segmentation and classification of heteregenous populations of cell nuclei in 3d confocal microscope images. *Cytometry A.*, 71(9):724–736, 2007.

H. Lin and J. Bilmes. A class of submodular functions for document summarization. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 510–520. Association for Computational Linguistics, 2011.

H. Lin and J. Bilmes. Learning mixtures of submodular shells with application to document summarization. In *Uncertainty in Artificial Intelligence (UAI)*, Catalina Island, USA, July 2012. AUAI.

Tao Liu, Andreas Rechtsteiner, Thea A Egelhofer, Anne Vielle, Isabel Latorre, Ming-Sin Cheung, Sevinc Ercan, Kohta Ikegami, Morten Jensen, Paulina Kolasinska-Zwierz, et al. Broad chromosomal domains of histone modification patterns in *C. elegans*. *Genome Research*, 21(2):227–236, 2011.

Y. Liu, K. Wei, K. Kirchhoff, Y. Song, and J. Bilmes. Submodular feature selection for high-dimensional acoustic score spaces. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 7184–7188. IEEE, 2013.

L. Lovász. Submodular functions and convexity. In M. Grotchel A. Bachem and B. Korte, editors, *Mathematical Programming – The State of the Art*, pages 235–257. Springer-Verlag, 1983.

Jakob Lovén, Heather A Hoke, Charles Y Lin, Ashley Lau, David A Orlando, Christopher R Vakoc, James E Bradner, Tong Ihn Lee, and Richard A Young. Selective inhibition of tumor oncogenes by disruption of super-enhancers. *Cell*, 153(2):320–334, 2013.

Julián Luengo, Salvador García, and Francisco Herrera. On the choice of the best imputation methods for missing values considering three groups of classification methods. *Knowledge and information systems*, 32(1):77–108, 2012.

Christopher D Manning and Hinrich Schütze. *Foundations of Statistical Natural Language Processing*. MIT press, 1999.

M. Minoux. Accelerated greedy algorithms for maximizing submodular set functions. *Optimization Techniques*, pages 234–243, 1978.

B. Mirzasoleiman, A. Karbasi, R. Sarkar, and A. Krause. Distributed submodular maximization: Identifying representative elements in massive data. In *Advances in Neural Information Processing Systems*, 2013.

T. Misteli. Beyond the sequence: Cellular organization of genome function. *Cell*, 128(4):787–800, 2007.

T. Mitchell. *Machine Learning*. McGraw-Hill, 1997.

Lluís Morey and Kristian Helin. Polycomb group protein-mediated repression of transcription. *Trends in biochemical sciences*, 35(6):323–332, 2010.

A. G. Murzin, S. E. Brenner, T. Hubbard, and C. Chothia. SCOP: A structural classification of proteins database for the investigation of sequences and structures. *Journal of Molecular Biology*, 247:536–540, 1995.

M. Narasimhan and J. Bilmes. A submodular-supermodular procedure with applications to discriminative structure learning. In *Uncertainty in Artificial Intelligence (UAI)*, Edinburgh, Scotland, July 2005. Morgan Kaufmann Publishers.

H. Narayanan. Submodular functions and electrical networks. *Annals of Discrete Mathematics*, 54, 1997.

R.M. Neal and G.E. Hinton. A view of the EM algorithm that justifies incremental, sparse, and other variants. In *Learning in graphical models*, pages 355–368. MIT Press, 1999.

G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher. An analysis of approximations for maximizing submodular set functions. *Mathematical Programming*, 14(1):265–294, 1978a.

G.L. Nemhauser, L.A. Wolsey, and M.L. Fisher. An analysis of approximations for maximizing submodular set functions—I. *Mathematical Programming*, 14(1):265–294, 1978b.

A. Y. Ng and M. I. Jordan. On discriminative versus generative classifiers: a comparison of logistic regression and naive Bayes. In T. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems*, 2002.

W. S. Noble. What is a support vector machine? *Nature Biotechnology*, 24(12):1565–1567, 2006.

W. Ohler, C. Liao, H. Niemann, and G. M. Rubin. Computational analysis of core promoters in the *drosophila* genome. *Genome Biology*, 3(12), 2002.

Z. Ouyang, Q. Zhou, and H. W. Wong. ChIP-Seq of transcription factors predicts absolute and differential gene expression in embryonic stem cells. *Proceedings of the National Academy of Sciences of the United States of America*, 106:21521–21526, 2009.

Alberto Paccanaro, James A Casbon, and Mansoor AS Saqi. Spectral clustering of protein sequences. *Nucleic acids research*, 34(5):1571–1580, 2006.

JD Parsons, S Brenner, and MJ Bishop. Clustering cDNA sequences. *Computer applications in the biosciences: CABIOS*, 8(5):461–466, 1992.

Florian M Pauler, Mathew A Sloane, Ru Huang, Kakkad Regha, Martha V Koerner, Ido Tamir, Andreas Sommer, Andras Aszodi, Thomas Jenuwein, and Denise P Barlow. H3k27me3 forms blocs over silent genes and intergenic regions and specifies a histone banding pattern on a mouse autosomal chromosome. *Genome Research*, 19(2):221–233, 2009.

P. Pavlidis, J. Weston, J. Cai, and W. S. Noble. Learning gene functional classifications from multiple data types. *Journal of Computational Biology*, 9(2):401–411, 2002.

J. Pearl. *Probabilistic Reasoning in Intelligent Systems : Networks of Plausible Inference*. Morgan Kaufmann, 1998.

Judea Pearl. *Causality: Models, Reasoning and Inference*, volume 29. Cambridge Univ Press, 2000.

L Peña-Castillo, M Tasan, CL Myers, H Lee, T Joshi, C Zhang, Y Guan, M Leone, A Pagnani, WK Kim, C Krumpelman, W Tian, G Obozinski, Y Qi, S Mostafavi, GN Lin, G Berriz, F Gibbons, G Lanckriet, J Qiu, C Grant, Z Barutcuoglu, DP Hill, D Warde-Farely, C Grouios, D Ray, JA Blake, M Deng, M Jordan, WS Noble, Q Morris, J Klein-Seetharaman, Z Bar-Joseph, T Chen, F Sun, OG Troyanskaya, EM Marcotte, D Xu, TR Hughes, and FP Roth. A critical assessment of *M. musculus* gene function prediction using integrated genomic evidence. *Genome Biology*, 9(Suppl 1):S2, 2008.

J. E. Phillips and V. G. Corces. CTCF: master weaver of the genome. *Cell*, 137(7):1194–1211, 2009.

E. Picardi and G. Pesole. Computational methods for ab initio and comparative gene finding. *Methods in Molecular Biology*, 609:269–284, 2010.

R. Pique-Regi, J. F. Degner, A. A. Pai, D. J. Gaffney, Y. Gilad, and J. K. Pritchard. Accurate inference of transcription factor binding from DNA sequence and chromatin accessibility data. *Genome Research*, 21(3):447–455, 2011.

B. D. Pope, T. Ryba, V. Dileep, F. Yue, W. Wu, O. Denas, D. L. Vera, Y. Wang, R. S. Hansen, T. K. Canfield, R. E. Thurman, Y. Cheng G. Gulsoy, J. H. Dennis, M. P. Snyder, J. A. Stamatoyannopoulos, J. Taylor, R. C. Hardison, T. Kahveci, B. Ren, and D. M. Gilbert. Topologically associating domains are stable units of replication-timing regulation. *Nature*, 515(7527):402–405, 2014.

J. Qiu and W. S. Noble. Predicting co-complexed protein pairs from heterogeneous data. *PLoS Computational Biology*, 4(4):e1000054, 2008.

S. Ramaswamy, P. Tamayo, R. Rifkin, S. Mukherjee, C. H. Yeang, M. Angelo, C. Ladd, M. Reich, E. Latulippe, J. P. Mesirov, T. Poggio, W. Gerald, M. Loda, E. S. Lander, and T. R. Golub. Multiclass cancer diagnosis using tumor gene expression signatures. *Proceedings of the National Academy of Sciences of the United States of America*, 98(26):15149–54, 2001.

G. Rätsch and S. Sonnenburg. *Kernel Methods in Computational Biology*, chapter Accurate splice site detection for *Caenhoribditis elegans*, pages 277–298. MIT Press, 2004.

Peter Rice, Ian Longden, Alan Bleasby, et al. EMBOSS: the European molecular biology open software suite. *Trends in genetics*, 16(6):276–277, 2000.

T. Ryba, I. Hiratani, J. Lu, M. Itoh, M. Kulik, J. Zhang, T. C. Schulz, A. J. Robins, S. Dalton, and D. M. Gilbert. Evolutionarily conserved replication timing profiles predict long-range chromatin interactions and distinguish closely related cell types. *Genome Res*, 20(6):761–770, 2010.

Tyrone Ryba, Dana Battaglia, Bill H Chang, James W Shirley, Quinton Buckley, Benjamin D Pope, Meenakshi Devidas, Brian J Druker, and David M Gilbert. Abnormal developmental control of replication-timing domains in pediatric acute lymphoblastic leukemia. *Genome Research*, 22(10): 1833–1844, 2012.

H. Saigo, J.-P. Vert, and T. Akutsu. Optimizing amino acid substitution matrices with a local alignment kernel. *BMC Bioinformatics*, 7(246), 2006.

B. Schölkopf and A. Smola. *Learning with Kernels*. MIT Press, Cambridge, MA, 2002.

A. Schrijver. *Combinatorial Optimization*. Springer, 2004.

scienceexchange. ChIP-seq prices at scienceexchange.com. https://www.scienceexchange.com/services/chip-seq, 2016.

E. Segal, Y. Fondufe-Mittendorf, L. Chen, A. Thøaström, Y. Field, I. K. Moore, J. Z. Wang, and J. Widom. A genomic code for nucleosome positioning. *Nature*, 44(17):772–778, 2006.

L. S. Shapley. Cores of convex games. *International Journal of Game Theory*, 1(1):11–26, 1971.

J. Shawe-Taylor and N. Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge UP, Cambridge, UK, 2004.

Nathan C Sheffield, Robert E Thurman, Lingyun Song, Alexias Safi, John A Stamatoyannopoulos, Boris Lenhard, Gregory E Crawford, and Terrence S Furey. Patterns of regulatory activity across diverse human cell types predict tissue identity, transcription factor binding, and long-range interactions. *Genome Research*, 23(5):777–788, 2013.

A. Siepel, G. Bejerano, J. S. Pedersen, A. S. Hinrichs, M. Hou, K. Rosenbloom, H. Clawson, J. Spieth, L. W. Hillier, S. Richards, G. M. Weinstock, R. K. Wilson, R. A. Gibbs, W. J. Kent, W. Miller, and D. Haussler. Evolutionarily conserved elements in vertebrate, insect, worm, and yeast genomes. *Genome Research*, 15(8):1034–1050, 2005.

L. Song and G. E. Crawford. DNase-seq: a high-resolution technique for mapping active gene regulatory elements across the genome from mammalian cells. *Cold Spring Harbor Protocols*, 2, 2010.

E. Sonnhammer, S. Eddy, and R. Durbin. Pfam: a comprehensive database of protein domain families based on seed alignments. *Proteins*, 28(3):405–420, 1997.

J. A. Stamatoyannopoulos. Illuminating eukaryotic transcription start sites. *Nature Methods*, 7: 501–503, 2010.

A. Subramanya and J. Bilmes. Entropic graph regularization in non-parametric semi-supervised classification. In Y. Bengio, D. Schuurmans, J. Lafferty, C. K. I. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems 22*, pages 1803–1811. NIPS Foundation, 2009.

A. Subramanya and J. Bilmes. Semi-supervised learning with measure propagation. *Journal of Machine Learning Research*, 12:3311–3370, 2011.

A. Subramanya, S. Petrov, and F. Pereira. Efficient graph-based semi-supervised learning of structured tagging models. In *Proc. of EMLNP 2010*, pages 167–176. Association for Computational Linguistics, 2010.

Fran Supek, Matko Bošnjak, Nives Škunca, and Tomislav Šmuc. REVIGO summarizes and visualizes long lists of gene ontology terms. *PloS ONE*, 6(7):e21800, 2011.

Anna Louise Swan, Ali Mobasheri, David Allaway, Susan Liddell, and Jaume Bacardit. Application of machine learning to proteomics data: classification and biomarker identification in postgenomics biology. *Omics: a journal of integrative biology*, 17(12):595–610, 2013.

Paul B Talbert and Steven Henikoff. Spreading of silent chromatin: inaction at a distance. *Nature Reviews Genetics*, 7(10):793–803, 2006.

R. E. Thurman, N. Day, W. S. Noble, and J. A. Stamatoyannopoulos. Identification of higher-order functional domains in the human ENCODE regions. *Genome Research*, 17:917–927, 2007.

R. J. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society B*, 58(1):267–288, 1996.

Andrey Nikolayevich Tikhonov. On the stability of inverse problems. In *Dokl. Akad. Nauk SSSR*,

volume 39, pages 195–198, 1943. **First described the now-ubiquitous method known as either L2 regularization or ridge regression.**

D. M. Topkis. *Supermodularity and complementarity*. Princeton University Press, 1998.

O. Troyanskaya, M. Cantor, G. Sherlock, P. Brown, T. Hastie, R. Tibshirani, D. Botstein, and R. Altman. Missing value estimation methods for DNA microarrays. *Bioinformatics*, 17:520–525, 2001.

O. G. Troyanskaya, K. Dolinski, A. B. Owen, R. B. Altman, and D. Botstein. A Bayesian framework for combining heterogeneous data sources for gene function prediction (in *S. cerevisiae*). *Proceedings of the National Academy of Sciences of the United States of America*, 100(14): 8348–8353, 2003.

Sebastian Tschiatschek, Rishabh K Iyer, Haochen Wei, and Jeff A Bilmes. Learning mixtures of submodular functions for image collection summarization. In *Advances in Neural Information Processing Systems*, pages 1413–1421, 2014.

UniProt Consortium. UniProt: a hub for protein information. *Nucleic acids research*, page gku989, 2014.

Rosanna Upstill-Goddard, Diana Eccles, Joerg Fliege, and Andrew Collins. Machine learning approaches for the discovery of gene–gene interactions in disease data. *Briefings in bioinformatics*, 14(2):251–260, 2013.

Ryan Urbanowicz, Delany Granizo-Mackenzie, and Jason Moore. An expert knowledge guided michigan-style learning classifier system for the detection and modeling of epistasis and genetic heterogeneity. In *Proc. Parallel Problem Solving From Nature 12*, pages 266–275, 2012a.

Ryan J Urbanowicz, Ambrose Granizo-Mackenzie, and Jason H Moore. An analysis pipeline with statistical and visualization-guided knowledge discovery for michigan-style learning classifier systems. *Computational Intelligence Magazine, IEEE*, 7(4):35–45, 2012b.

Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9(2579-2605):85, 2008.

A. Visel, M. J. Blow, Z. Li, T. Zhang, J. A. Akiyama, A. Holt, I. Plajzer-Frick, M. Shoukry, C. Wright, F. Chen, V. Afzal, B. Ren, E. M. Rubin, and L. A. Pennacchio. ChIP-seq accurately predicts tissue-specific activity of enhancers. *Nature*, 457(7231):854–858, Feb 2009. doi: 10.1038/nature07730. URL http://dx.doi.org/10.1038/nature07730.

X. Vives. *Oligopoly pricing: Old ideas and new tools*. The MIT Press, 2001.

Martin J Wainwright and Michael I Jordan. Graphical models, exponential families, and variational inference. *Foundations and Trends® in Machine Learning*, 1(1-2):1–305, 2008.

G. Wang and R. L. Dunbrack, Jr. PISCES: a protein sequence culling server. *Bioinformatics*, 19: 1589–1591, 2003.

Jun Wang, Tony Jebara, and Shih-Fu Chang. Graph transduction via alternating minimization. In *Proceedings of the 25th international conference on Machine learning*, pages 1144–1151. ACM, 2008.

J. Warga. Minimizing certain convex functions. *Journal of the Society for Industrial and Applied Mathematics*, 11(3):588–593, 1963.

T. Wasson and A. J. Hartemink. An ensemble model of competitive multi-factor binding of the genome. *Genome Research*, 19(11):2102–2112, 2009.

Kai Wei, Yuzong Liu, Katrin Kirchhoff, and Jeff Bilmes. Using document summarization techniques for speech data subset selection. In *HLT-NAACL*, pages 721–726, 2013.

Kai Wei, Yuzong Liu, Katrin Kirchhoff, Christopher Bartels, and Jeff Bilmes. Submodular subset selection for large-scale speech training data. In *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*, pages 3311–3315. IEEE, 2014.

Karen S Weiler and Barbara T Wakimoto. Heterochromatin and gene expression in *Drosophila*. *Annual review of genetics*, 29(1):577–605, 1995.

L. Weizhong and G. Adam. Cd-hit: a fast program for clustering and comparing large sets of protein or nucleotide sequences. *Bioinformatics*, 22(13):1658–1659, 2006.

Bo Wen, Hao Wu, Yoichi Shinkai, Rafael A Irizarry, and Andrew P Feinberg. Large organized chromatin K9-modifications (LOCKs) distinguish differentiated from embryonic stem cells. *Nature Genetics*, 41(2):246, 2009.

J. Weston, A. Elisseeff, D. Zhou, C. Leslie, and W. S. Noble. Protein ranking: from local to global structure in the protein similarity network. *Proceedings of the National Academy of Sciences*, 101 (17):6559–63, 2004.

W. A. Whyte, D. A. Orlando, D. Hnisz, B. J. Abraham, C. Y. Lin, M. H. Kagey, P. B. Rahl, T. I. Lee, and R. A. Young. Master transcription factors and mediator establish super-enhancers at key cell identity genes. *Cell*, 153(2):307–319, 2013.

David H Wolpert and William G Macready. No free lunch theorems for optimization. *Evolutionary Computation, IEEE Transactions on*, 1(1):67–82, 1997.

K. Woodfine, H. Fiegler, D. M. Beare, J. E. Collins, O. T. McCann, B. D. Young, S. Debernardi, R. Mott, I. Dunham, and N. P. Carter. Replication timing of the human genome. *Human molecular genetics*, 13(2):191–202, 2004.

Jiong Yang and Wei Wang. CLUSEQ: efficient and effective sequence clustering. In *Data Engineering, 2003. Proceedings. 19th International Conference on*, pages 101–112. IEEE, 2003.

K. Y. Yip, C. Cheng, N. Bhardwaj, J. B. Brown, J. Leng, A. Kundaje, J. Rozowsky, E. Birney, P. Bickel, M. Snyder, and M. Gerstein. Classification of human genomic regions based on experimentally determined binding sites of more than 100 transcription-related factors. *Genome Biology*, 13(9):R48, 2012.

Kevin Y Yip, Chao Cheng, and Mark Gerstein. Machine learning and genome annotation: a match meant to be? *Genome biology*, 14(5):205, 2013.

Y. Zhang, T. Liu, C. A. Meyer, J. Eeckhoute, D. S. Johnson, B. E. Bernstein, C. Nusbaum, R. M. Myers, M. Brown, W. Li, and X. S. Liu. Model-based analysis of ChIP-Seq (MACS). *Genome Biology*, 9(9):R137, 2008. doi: 10.1186/gb-2008-9-9-r137. URL http://dx.doi.org/10.1186/gb-2008-9-9-r137.

Xiaojin Zhu and Zoubin Ghahramani. Learning from labeled and unlabeled data with label propagation. Technical report, Carnegie Mellon University, 2002.

Xiaojin Zhu, Jaz S Kandola, Zoubin Ghahramani, and John D Lafferty. Nonparametric transforms of graph kernels for semi-supervised learning. In *NIPS*, volume 17, pages 1641–1648, 2004.

A. Zien, G. Rätsch, S. Mika, B. Schölkopf, T. Lengauer, and K.-R. Müller. Engineering support vector machine kernels that recognize translation initiation sites. *Bioinformatics*, 16(9):799–807, 2000.