Distribution Agreement

In presenting this thesis or dissertation as a partial fulfillment of the requirements for an advanced degree from Emory University, I hereby grant to Emory University and its agents the non-exclusive license to archive, make accessible, and display my thesis or dissertation in whole or in part in all forms of media, now or hereafter known, including display on the world wide web. I understand that I may select some access restrictions as part of the online submission of this thesis or dissertation. I retain all ownership rights to the copyright of the thesis or dissertation. I also retain the right to use in future works (such as articles or books) all or part of this thesis or dissertation.

Signature:

_____ _____

Olgert Denas Date

Deep Models for Gene Regulation

By

Olgert Denas
Doctor of Philosophy

Computer Science

James Taylor, Ph.D.
Advisor

Eugene Agichtein, Ph.D.
Committee Member

Michelangelo Grigni, Ph.D.
Committee Member

Jeremy Goecks, Ph.D.
Committee Member

Accepted:

Lisa A. Tedesco, Ph.D.
Dean of the Graduate School

Date

Deep Models for Gene Regulation

By

Olgert Denas
Ph.D., Emory University, 2014

Advisor: James Taylor, Ph.D.

An abstract of
A dissertation submitted to the Faculty of the Graduate School
of Emory University in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
in Computer Science
2014

Abstract

Deep Models for Gene Regulation
By Olgert Denas

The recent increase in the production pace of functional genomics data has created new opportunities in understanding regulation. Advances range from the identification of new regulatory elements to gene expression prediction from genomic and epigenomic features. At the same time, this data-rich environment has raised challenges in retrieving and interpreting information contained therein.

Based on recent algorithmic developments, deep artificial neural networks (ANN) have been used to build representations of the input that preserve only the information needed to the task at hand. Prediction models based on these representations have achieved excellent results in machine learning competitions. The deep learning paradigm describes how to build these representations and train the prediction models in a single learning exercise.

In this work, we propose ANN as tools for modeling gene regulation and a novel technique for interpreting what the model has learned.

We implement software for the design of ANNs and for training practices over functional genomics data. As a proof of concept, use our software to model differential gene expression during cell differentiation. To show the versatility of ANNs, we train a regression model on measurements of protein-DNA interaction to predict gene expression levels.

Typically, input feature extraction from a trained ANN is formulated as an optimization problem whose solution is slow to obtain and not unique. We propose a new efficient feature extraction technique for classification problems that provides guarantees on the class probability of the features and their norm. We apply this technique to identify differential gene expression associated features that agree with previous empirical studies.

Finally, we propose building representations of functional features from protein-DNA interaction measurements using a deep stack of nonlinear transformations. We show that these reduced representations are informative and can be used to label parts of the gene, regulatory elements, and quiescent regions.

While widely successful, deep ANNs are considered to be hard to use and interpret. We hope that this work will help increase the adoption of such models in the genomics community.

Deep Models for Gene Regulation

By

Olgert Denas
Ph.D., Emory University, 2014

Advisor: James Taylor, Ph.D.

A dissertation submitted to the Faculty of the Graduate School
of Emory University in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
in Computer Science
2014

Acknowledgments

My advisor, James Taylor, for his patience and help.

The Mouse ENCODE Consortium in particular, Ross Hardison and Dave Gilbert. The members of the committee for their help and feedback. M. Gehring (Sauria) for great discussions.

My family and Barbara for their love and uninterrupted support.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Introduction

The information needed for the viability and identity of living cells is encoded in genomic DNA sequence. Two of the most important elements in genomic DNA are genes and *cis*-regulatory modules (CRM). The DNA sequence of genes can be transcribed by the cell and translated in a chain of amino acids forming a protein. On the other hand, CRM are binding locations for special proteins, called transcription factors (TFs), that are necessary for transcription.

Because, binding events at CRMs can potentially increase or decrease the transcription rate of a gene, CRMs have a direct impact on gene expression and eventually on the protein concentration levels in the cell. In this sense, we say that CRMs regulate gene expression. The effect of CRMs in regulation manifests itself in many aspects of the cell's life, such as determination of the cell type in an organism, control in a timely manner the developmental process of the organism, and influencing adaptation to environment and disease susceptibility.

Despite their importance, the mechanisms of transcriptional regulation regulation are not entirely understood, nor are catalogs of these CRM complete.

Until recently, the discovery of CRMs has relied heavily in computational approaches analyzing sequence data.

Computational approaches to CRM prediction can be viewed in three groups (Reviwed in [Hardison & Taylor, 2012]). One group uses existing motifs – a motif is a random vector of approximately 5-10bp in length, with each component a multinomial distributed random variable of the DNA alphabet. These methods identify putative CRM by scan the genome for significant matches with the given motif. These methods can enrich the catalog of CRM that share a common known signature (or motif). Extensions of these methods simultaneously perform motif discovery and motif matching, thus potentially reporting putative CRM not necessarily similar to already annotated motif.

Another group of methods is based on the tendency of regulatory elements to be located over genome regions that have remained largely unchanged during the evolution of species. Using alignments [Blanchette, 2007] of sequences from different species, statistical methods can identify these regions and other regions that have accumulated less mutations than expected.

Finally, a group of methods is based on directly measured DNA-protein interactions. Measurements provide binding evidence for a given protein throughout the genome. Based on this evidence, these methods infer the genome location of the binding event. While a binding event does not always imply the bound genome region has regulatory function, these methods have been very successful in predicting the location of a number of CRM.

It is important to realize that all three groups have their weaknesses. Motif based methods are biased toward already annotated elements. Methods that discover motifs have been shown to yield a large number of false positives. Methods based on conservation leverage the idea that purifying selection (rejecting mutations) is evidence of function. The advantage of these methods is that they do not require sequence patters or motif occurrences in the sequence. However, these methods are biased toward evolutionary conserved

sequence and tend to miss functional sequence that has not been conserved between species that are being compared. Finally, methods based on direct DNA-protein interactions, while having the lowest bias, are also subject to high false positives.

Another disadvantage of methods based on measured DNA-protein interactions is that they require an experiment for each protein and cell condition, which can be costly. In an effort to better understand transcriptional regulation, several projects have applied high throughput sequencing technologies to the production of DNA - protein binding data in many cell types and TFs from several types of experiments. Notably, the ENCODE [ENCODE Project Consortium, 2011; Mouse ENCODE Consortium et al., 2012; Gerstein et al., 2010; modENCODE Consortium et al., 2010] projects have been cataloging DNA-protein interactions for hundreds of proteins in hundreds of cell types and cell lines in Human, Mouse, Worm and Fly. These projects have produced the most complete functional genomics data to date. Coupled with whole genome comparisons and estimations of the level of sequence conservation, these data have been the basis of a number computational methods to predict CRM.

This thesis proposes computational approaches for the analysis of functional genomic data under the *deep learning* framework. Models with a deep architecture have been show to perform well in a number of applications including audio and visual pattern recognition. We apply deep models to functional genomic data in an effort (1) to better understand CRM and the rules that govern their regulatory activity and (2) to build low dimensional representations of multiple tracks of experimental functional genomics data.

1.2 Summary of remaining chapters

Chapter 2 In this chapter we introduce basic concepts in the biology of genomes. First, we look at the biological processes in gene regulation, the recent technologies developed to measure regulatory activities on the genome, and computational approaches to low-level interpretation of the raw data. Next, we introduce Artificial Neural Networks (ANNs). We describe fundamental architectures and training methods.

Chapter 3 In this chapter we consider the problem of interpreting trained ANNs. We review existing methods and propose a novel layer-wise feature extraction algorithm that identifies relevant class-related input features. We also describe a freely available and efficient implementation of the algorithm and models described above in a software library called `dimer` (Deep Models of Expression and Regulation).

Chapter 4 In this section we apply the algorithms proposed in Chapter 3 to two problems. First, we consider the problem of predicting and interpreting the direction of gene expression (repression or induction) during Erythroid differentiation. We propose an ANN architecture and apply our feature extraction algorithm. Next, we provide a biological interpretation of the extracted features. Second, we reapply the training and analysis process in the previous section to the quantitative prediction of gene expression on the Gm12878 cell line.

Chapter 5 In this chapter we apply unsupervised learning of genome features to two problems. First, we extract informative patterns of biochemical activity of the genome and synthesize that information in the form of labels. The aim is to provide high-level representations that can be generated systematically and are interpretable. Next, we apply unsupervised models to

the problem of characterizing the TF composition of specific genome regions called Topologically Associated Domains (TADs). The method discovers a partition of TADs induced by their TF compositions which aligns with a partition induced by their replication timing during cell division.

1.3 Contributions of this thesis

The material contained in Chapter 1 is the result of work done within the Mouse ENCODE project, as part of the regulatory elements analysis working group. Results were submitted as a research article and involved

- the development of a consistent method, based on multiple genome alignments, for the mapping of regulatory elements between human and mouse; the method was the basis of the *cis*-regulatory element comparison pipeline used by the mouse ENCODE consortium.

- Integration of ChIP-Seq assays of transcription factors (TFs) and chromatin accessibility with gene annotations revealed a strong divergence between sequence and function conservation of regulatory elements. Furthermore, it provided insights on the species, cell and TF specific evolutionary landscape of regulatory regions.

In particular, we observed that differences in functional versus sequence conservation reveal that information from sequence conservation alone is not enough to identify regulatory elements. Computational methods that use protein-DNA interaction measurements are in this regard a viable avenue. This observation served as motivation for the subsequent work.

We propose a novel method for interpreting a trained deep Neural Network. We improve upon existing results in the following ways:

- our method is parameter free and extracts a deterministic set of features,

- our method is efficient,
- we provide guarantees on both the parsimony of the features and their class probability assigned by the model.

We implemented software for modeling functional genomics data using supervised and unsupervised deep learning techniques. This work is in preparation for submission in *BMC Genomics*.

We apply our findings on three problems in functional genomics

- Predict differential gene expression during Erythroid differentiation.
- Predict quantitative gene expression from transcription factor occupancy data with excellent accuracy.
- Perform unsupervised genome-wide labeling based on multiple tracks of functional genomics data. Preliminary results of this work appeared as a poster in Systems Biology: Global Regulation of Gene Expression, Cold Spring Harbor Laboratories, NY (2014).

Given the limited use of deep ANNs in functional genomics, we hope our work will draw more attention to these models.

Chapter 2

Background

2.1 Importance of regulation

The dynamic equilibrium of gene expression levels and their products is maintained by direct or indirect execution of instructions encoded in the genome sequence. These instructions can be functional coding or functional noncoding. The functional coding portion is composed of genome regions that are transcribed into mRNA which, after processing, is translated into chains of amino acids. The noncoding functional regions are more elusive but equally important. These regions perform functions by virtue of their position in the genome or their sequence, and are usually referred to *cis*– regulatory elements.

The interaction between functional coding and functional noncoding genome elements results in a carefully tuned control of the expression levels of selected sets of genes. This orchestration is at the base of biological processes in the cell, including cell's response to environmental stimuli, its development, and its differentiation. At the organismal level, regulation of gene expression is at the basis of speciation, and disease susceptibility [Bulger & Groudine, 2010; E. Davidson, 2006].

In multicellular organisms, cells differentiate to perform highly specialized functions. For example, certain blood cells are specialized in the production of immunoglobulins by expressing the genes that encode the appropriate proteins. Similarly, pancreatic islet cells produce insulin as a response to high sugar concentration. This level of specialization is regulated through maintenance of appropriate gene expression levels. So, contrary to pancreatic islet cells, other cell types, like skin cells, cannot produce insulin at all as they have the insulin genes silenced. The choice and maintenance (at silenced or expressed levels) of a set of genes is a form of regulation, which occurs during cell differentiation, making this form of regulation time sensitive.

Another example that will be revisited later is the G1E biological model. It simulates stages of Erythropoiesis, the precess by which red blood cells are created from stem cells through a series of differentiation steps. These steps, start with G1E cells, which can be prevented from differentiating by removal of a regulatory protein called GATA1 (described in detail in [Weiss et al., 1997]). Restoration of GATA1 is followed by a chain of events, including the production of other important proteins, which will promote the expression of particular sets of genes. As a result, the cells resume their differentiation process.

Perhaps the most fascinating coordinations of gene expression take place during organism development. Regulation of gene expression levels results in substantial morphological changes during the early stages of embryo development. For example, the early fly (*Drosophila Melanogaster*) embryo maintains a delicate control of the concentration of maternal proteins. This cellular environment promotes the expression of specific set of genes and gene regions which correspond to body parts [Klug et al., 2007].

Another example, is the process of X chromosome inactivation. A battery of genes found in the Xic region of the X chromosome are co-expressed to produce a set of proteins that in turn will regulate the important Xist gene.

The product of the latter, a long RNA molecule, will coat most of chromosome X and inactivate it [Morey & Avner, 2011; Abramowitz & Bartolomei, 2012]. The process is very robust and manages X chromosome choosing, initiation, spreading, establishment of the X-ist product, and finally maintenance through the whole life of the cell and its progeny.

Species complexity is best reflected by the complexity of the functional noncoding genome [Meader et al., 2010]. Furthermore, mutations on the noncoding functional genome constitute the main mechanism of a species' evolution [E. Davidson, 2006]. Except for a modest number of protein coding genes related to the immune, reproductive, and language systems [Enard et al., 2002; Swanson et al., 2001; Hughes & Yeager, 1998], differences in human and chimpanzee are completely due to changes in functional noncoding regions [King & Wilson, 1975]. Evidence of correlation in human-chimp functional noncoding sequence divergence and gene expression divergence has also been found [Wilson & Odom, 2009].

Creation of species specific regulatory elements is still not well understood. However, measurements of DNA protein interaction have been useful in identifying putative regulatory regions. Recently, we performed a comparative analysis of DNA-protein interaction sites for 116 human and 35 mouse proteins over 52 human and 35 mouse cell types and tissues from the Human and Mouse ENCODE projects. Aside from a substantial amount (44% of mouse and 43% of human) of species specific regulatory material, the study suggested a mechanism for the creation of new regulatory material in a genome.

In contrast to the examples mentioned above, mutations on the functional noncoding genome are subject to purifying selection – i.e., selection against deleterious mutations at these sites. Evidence from comparisons of distant metazoans or even between species at the extremes of eukaryotes shows that the subset of the genome encoding TFs and signaling components (e.g., for temporal/spatial gene expression patterns) is largely the same [E. David-

son, 2006]. However, gene specializations within a gene family do occur in taxonomic units [Wilson & Odom, 2009; Levine & Tjian, 2003].

To accommodate these apparently contradicting observations, Davidson and Erwin [E. H. Davidson & Erwin, 2006] mapped evolutionary events with changes in specific components of the gene regulatory network into several hierarchical and modular network entities. On the top of the hierarchy stay *kernel modules*, evolutionary inflexible elements performing essential functions on which other processes depend. Below, are placed plugins which are modules that perform functions related to diverse developmental purposes. At the bottom, they place *I/O switches*, which allow or disallow developmental modules to act at certain contexts. Their proper use is reflected in proper body part sizes. Other lower level components are *gene batteries* as peripheral subnetworks composed of differentiation genes coding for proteins and used for specific purposes within a cell type.

The identity and viability of an organism and the constituting cells are largely defined from regulation of gene expression. In Eukaryotes, regulation happens in various steps of the protein synthesis process, including the predisposition of the DNA to be transcribed and the translation of mature mRNA (See Fig. 2.1). These steps include chromatin remodeling, transcription initiation, and post transcriptional modifications. In the next section we will look at each of these steps and the corresponding mechanisms of regulation.

2.2 Mechanisms of gene regulation

2.2.1 Transcriptional regulation

The main form of gene expression regulation in Eukaryotes is at the transcriptional level [Derman et al., 1981; Roop et al., 1978]. This form of regulation

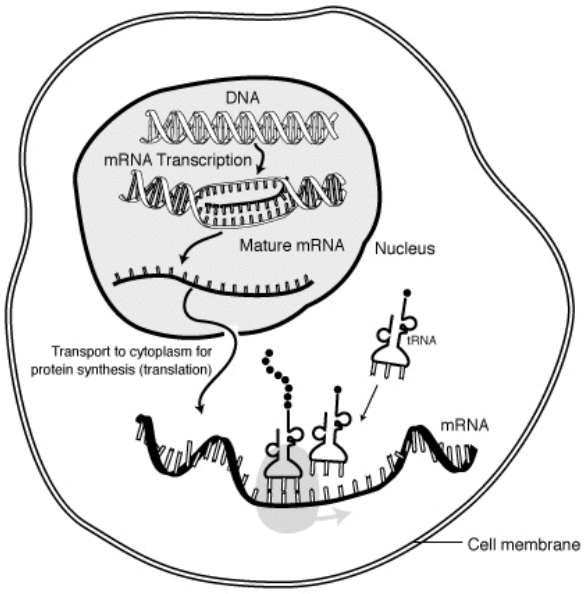


Figure 2.1: Diagram of the protein synthesis process. Inside the nucleus, the RNA Polymerase produces an RNA strand complementary to the DNA. Depending on the DNA sequence, the RNA can code for a protein (Mature mRNA in the figure) or perform other functions. Ribosomal RNA, for example, is the RNA component of the Ribosome, which is shown as a shaded area outside the nucleus. Transfer RNA (tRNA in the figure) mediates the formation of the amino acid chain. Outside the nucleus, the Messenger RNA is translated into a chain of amino acids which spell the sequence of a protein. Taken from Wikipedia `https://en.wikipedia.org/wiki/Protein_biosynthesis` on April, 9 2014.

reflects the ability of the cell to regulate gene expression through protein-DNA interactions prior to gene transcription. The regulatory DNA sites are recognized and bound by DNA-binding transcription factors (TFs), proteins that carry an enhancing or repressing function on the expression of the target gene.

Regulatory regions of the DNA can be *cis*– or *trans*–acting. In the former case, elements are found on the same chromosome as the target gene and provide binding locations for TFs which mediate directly the expression of the target gene. They are typically organized in modules called *cis*– regulatory modules (CRM) and bind a battery of TFs. The TFs and the genomic regions that code those TFs, on the other hand, are said to be acting in *trans*–, or to be *trans*–acting elements.

One fundamental *cis*–acting element is the promoter region of a gene – a region immediately upstream[1] of the transcription start site (TSS). This element, binds the general transcription factor (GTF) complex, which by itself is enough to start transcription. Interference with the assembly of the GTF is, thus, a potential form of regulation. Alone, this form of regulation is not enough for the fine tuned gene expression we see in high order organisms.

The cell, in fact, has a way of selectively promoting or demoting the formation and binding of the GTF at the promoter region, through *enhancers* and *silencers*, respectively. Another class of regulatory elements, called *insulators*, can block the effect of an enhancer to a gene. Enhancers, silencers and insulators are similar in principle to the promoter region, as they both bind TFs to regulate gene expression. However, they differ from each other in a number of ways. Enhancers and silencers can be located almost distally upstream of the gene's TSS, or downstream in intronic regions – an example is the heavy-chain gene of immunoglobulin, which contains an enhancer in an intron – and thus can act regardless of the direction in which the target gene

[1]The direction of the "stream" is defined by the direction in which the gene is read.

is located. It has been observed that certain insulators make use of spatial conformations of the chromatin, such as looping [Phillips & Corces, 2009]. Another fundamental difference is that enhancers, silencers, and insulators can be constitutive or tissue type specific (see also [Maston et al., 2006] for a review). Finally, it has been observed recently that enhancers are regularly transcribed in both directions [Kim et al., 2010] producing eRNA transcripts. It is not yet clear what is the function of eRNA or what are the implications of this mechanism, but extensive eRNA measurements in human cells by the FANTOM Consortium [Andersson et al., 2014] have been produced and used to better understand the cell specific regulatory landscape in humans.

2.2.2 Post-transcriptional regulation

If transcription of a DNA region was not repressed by one of the mechanisms described above, an mRNA molecule would have been created. At this point, there are several routes for the mRNA molecule: it may get exported outside of the nucleus and get translated into a protein, it may perform regulation functions (see below), or it may be degraded before it is able to perform any function.

Degradation of mRNA molecules is another way of gene expression regulation, called post-transcriptional regulation, which we review briefly.

Post-transcriptional regulation includes a number of ways the cell can act on the mRNA that resulted from gene transcription. One way, typical of Eukaryotes, is the process of alternative splicing. Agents, called *snRNPs* and composed of proteins and *small nuclear RNA*, remove large introns from the mRNA. The regulatory function of this mechanism becomes apparent in genes products that accept alternative splicing. It follows that the concentration of the possible protein products out of a transcript will depend on the splicing mechanism.

A more recently discovered form of regulation involves small noncoding RNA molecules. The first observations, revealed that these molecules tended to be long and create hairpin structures which tended to repress the genes that produced them. More evidence showed that these molecules are part of a complex gene regulating mechanism. Some long noncoding RNAs are diced by the *dicer* protein into either *miRNAs* or *siRNAs*. The former, simply binds to mRNA products preventing further processing of the transcript. *siRNAs*, on the other hand, bind a protein called RISC which is able to degrade mRNA sequences with regions complementary to the RISC bound *siRNA*.

2.2.3 Epigenetic regulation

The DNA is wrapped around proteins called histones (H2a, H2b H3, and H4, plus variants) to form a basic "string on beads" structure, which can further assemble in other structures. This structure is called chromatin and its unit is the nucleosome. Chromatin structure varies along the DNA, forming domains that are more or less accessible to proteins. For example, two large domains of the chromatin are the *euchromatin*, largely accessible, and the *heterochromatin*, condensed and non-accessible. Other smaller domains are maintained and they all contribute to the accessibility of the DNA by the TFs.

The mechanism of how small active or "dead" domains are created and maintained is not fully understood. However, a number of proteins called remodelers are known to perform nucleosome displacement. Remodelers can recognize post-translational modifications of the histones located in their N-terminus, or "tails". There are different types of histone tail modifications, such as Methylation, Acetylation, and Ubiquitation. Common combinations of modifications on the same tail have been compiled into a histone code[2]

[2]The code defines also succinct names of modifications. For example, H3K4me is a

[Turner, 2005].

Global chromatin environments have been associated with histone modifications. For example, it has been observed that heterochromatin is wrapped around histones with low Acetylation and and high on certain types of Methylation: H3K9, H3K27, and H4K20; euchromatin, on the other hand, is associated with Acetylation, H3K4me3, H3K36me3, H3K79me3; and bivalent chromatin contain modifications usually found in both euchromatin and heterochromatin.

Histone modifications are also directly related to regulation of transcription, by marking active or repressed genomic regions. Acetylation correlates with with activation of transcription and Deacetylation with repression of transcription; Lysine Methylation at, H3K4, H3K36 and H3K79 correlates with activation, but at H3K9, H3K27, H4K20 correlates with repression. Methylation of H3K36 is context dependent because it is associated with activation if found on coding sequence, and with repression if found on promoters. (These and other aspects of chromatin modifications are reviewed in [Kouzarides, 2007]).

The epigenetic landscape of the genome is often considered in the context of the 3D organization of the chromatin. Chromatin conformation capture technology [Lieberman-Aiden et al., 2009] has been used to measure distal chromatin interactions. These interactions define a set of genomic topologically associated domains (TADs) which have proved to be very informative in understanding distal regulatory mechanisms [Dixon et al., 2012]. Recently, the Mouse ENCODE Consortium performed an association analysis between TADs genome domains (RD) defined by synchronized replication timing during DNA replication. The study showed a strong correlation between TAD boundaries and RD boundaries [Pope et al., n.d.].

Methylation of the 4th Lysine amino acid in histone 3.

2.3 Computational methods for TF binding analysis

2.3.1 From HGP to ENCODE

The completion of the human reference genome by the Human Genome Project (HGP) uncovered the sequence that encodes the information necessary for the viability of the human organism. While other organisms had been already sequenced, the completion of the human genome was followed by an increase of data production and technological advances in the biological sciences. The reason of this increase lies in part in the need to interpret the newly sequenced genome. The common theme of advances following the completion of the HGP is the large scale low cost production of functional annotations of the genome sequence.

As the annotation of the genome became richer, it became clear that organism complexity does not lie on the number of genes it possesses, but on its faculty to combine their products [E. H. Davidson & Erwin, 2006; Meader et al., 2010]. In a way, the community found itself in a position similar to the one right before the start of the HGP, but with more tools and data at hand, and different questions in mind: instead of wondering about the location of cancer genes, with an annotated sequence at hand, the challenge was to understand how are these genes controlled, what are their roles and the roles of their manipulators [Dulbecco, 1986]. Inspired by the success of the HGP, several consortia, notably, US National Institutes of Health (NIH) Roadmap Epigenomics Mapping Consortium [Bernstein et al., 2010], the ENCODE, modENCODE, and the MouseENCODE, set to catalog and discover the regulatory nature of genome elements. One product of the ENCODE projects, was the publication of the largest number of uniformly processed measurements of biochemical activity on the genome [Plocik & Graveley, 2013]. As a

side effect new methods and technologies were created and validated. Today, the investigator possesses tens of methods for measuring different aspects of the sequence, functional, and structural nature of the genome.

2.3.2 Next generation sequencing

The routine sequencing of whole genomes has been one of the most coveted achievements in biology [Dulbecco, 1986; Collins & Galas, 1993]. The first sequencing methods were developed in the 80s [Sanger et al., 1982]. More, than a decade later, the process had been automated and the rate of sequencing was large enough for the Human Genome Project (HGP) to be feasible [Lander et al., 2001].

The preferred method of the HGP was the Hierarchical Shotgun (HS) Sequencing. In this approach, the DNA is broken in 150 Mb length sequences and replicated using BAC vectors. Once the correct order of these large sequences is determined a coarse scaffold is created. Next, each of the scaffold parts is sequenced and assembled just as if it were a genome. Since the order of the BAC inserts is known in advance it is possible to combine the inserts into the original genome (Figure 2.2).

A competing method to HS was the Whole Genome Shotgun (WGS) advocated by Celera [Weber & Myers, 1997] (Figure 2.2). This method skipped the coarse scaffolding step altogether and mapped the short reads directly to the genome. Concurrently to the HGP, a draft genome was published by Celera using this method [Venter et al., 2001]. The effectiveness of WGS versus HS has been debated extensively [Waterston et al., 2002; Myers et al., 2002]. One conclusion that can be safely drawn is that WGS paved the way for next generation sequencing methods (see below), but HS was indispensable for the creation of a reference error-free human genome sequence.

Celera's WGS was followed by the developement of a novel technology for

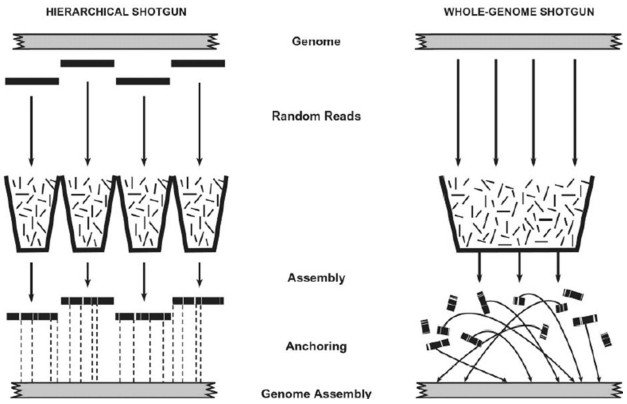


Figure 2.2: Schematic views of HS (left) and WGS (right). For a short description of each technology see the text. Adopted from [Waterston et al., 2002]. Permission granted from PNAS.

DNA sequencing, commonly called Next Generation Sequencing (NGS). This technology, with respect to the first generation, or Sanger sequencing, can produce a larger amount of reads in lower time and cost. NGS platforms capture fragmented DNA sequence and amplify it into fragment clusters. Then, they probe each fragment cluster for the next base through a process called sequencing by ligation or sequencing by synthesis, depending on the technology. Each probing step binds the primers with fluorescent nucleotides that can be detected and processed into sequence reads. Sequence reads from NGS instruments contain more errors and are much shorter than reads from the Sanger method, although this is constantly improving [Metzker, 2010].

In parallel to WGS, NGS has seen a widespread use for sequencing smaller targeted regions of the genome. The most widely used targeting technique is by chromatin immuno-precipitation (ChIP). This procedure is a general method by which it is possible to identify properties of sequence regions corresponding to some criteria. For example, one can identify the sequence of transcribed regions, or the sequence of regions that interaction with a given TF. It is often argued that the successful application of NGS with

ChIP experiments has somewhat shadowed its intended use in whole genome sequencing [Wold & Myers, 2008].

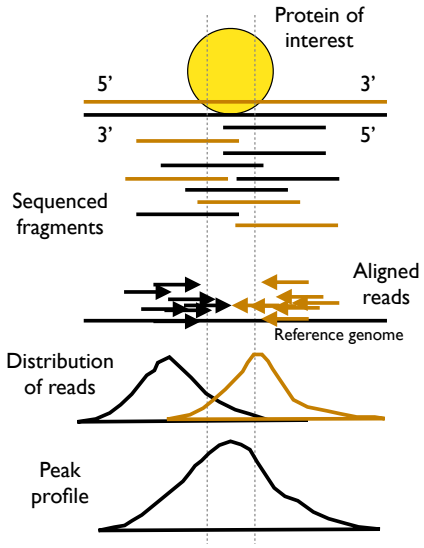


Figure 2.3: Schematic of a ChIP-Seq experiment. A protein of interest is linked to its binding location in the genome. The sequence is sheared and regions bound by the protein are selected with an antibody and sequenced from the 5' end. Only part of the fragment is sequenced (called read). Reads are aligned to the reference genome and histograms of reads from forward and reverse strands are computed. Further processing estimates the signal profile for the fragments. This can further be processed into a discrete peak call (See Section 2.3.3).

A typical ChIP-Seq assay starts with the immuno-precipitation of specific genomic regions of interest, such as promoter regions or enhancers. The resulting sample is then sequenced and the reads are aligned back to the genome (Figure 2.3). Suppose that we want to target genome regions occupied by a certain TF. A ChIP-Seq assay starts by linking the TF to the DNA and immuno-precipitating it by attaching an antibody specific to the TF. Next, cross-linking is reversed and, after fragmentation and size selection, the sample is sequenced by a NGS technology [Johnson et al., 2007]. To

improve the quality of the subsequent data analysis, a control experiment is performed in parallel where sequencing is applied to randomly chosen DNA fragments not bound by the TF. Sequence reads from the ChIP sample are highly enriched by DNA regions previously bound by the target TF, while the control sample reflects noise.

ChIP-Seq has a number of advantages over its predecessor technologies such as ChIP-chip [Ren et al., 2000]. First, not relying on hybridization it removes biases over GC content and issues related to the hybridization chemistry. Second, it improves accessibility as it does not depend on the design of genome tiling arrays; for example some repetitive regions are not easily accessible by microarrays. Finally, Chip-Seq can represent larger signal variations and achieve higher accuracy at lower cost.

2.3.3 From read counts to signal

One of the biggest challenges in completing a ChIP-Seq assay is the discrimination of the read counts into signal and noise – because reads accumulate visibly to TF occupancy sites, these step is also called peak calling. Peak calling is the problem of finding a suitable distribution for the background noise (i.e., signal in absence of TF binding) and then scanning the ChIP signal for regions where the signal is significantly higher than expected under the background distribution. A good estimation of the background distribution is thus essential for the quality of results [Zhang et al., 2008; Kharchenko et al., 2008].

One way to estimate the background distribution is to perform a ChIP-Seq experiment in absence of the protein of interest. This is a *control* experiment and can be done once, or each time a new experiment is performed. When control experiments cannot be performed, one can use standard background estimation inferred from others, or call the peaks from the signal experi-

ment only. Depending on the presence of a control experiment there are two strategies for modeling the background distribution.

The total number N of uniquely mapped reads is binned into windows of length w (typically 100bp). The number of reads in each window is written as a mixture $g(n) = \pi_0 f_0(n) + (1 - \pi_0) f_1(n)$, where $0 \leq \pi_0 \leq 1$, f_0 models the number of reads in absence of the protein, and f_1 models the number of reads when the protein binds. For a window i, the number of reads under the background distribution f_0 can be written as $n_i|\lambda_i \sim \text{Poisson}(\lambda_i)$ and $\lambda_I \sim \text{gamma}(\alpha, \beta)$[3]. This composed distribution can be also written as a negative binomial $NB(\alpha, \beta/(1+\beta))$. In the absence of a control experiment it is assumed that windows with less than three reads are regions that did not bind the protein of interest. Let then u_0, u_1, u_2 be the count of windows with 0, 1, and 2 mapped reads respectively. The estimated values of parameters α and β are $\alpha = (u_1/u_0)/\rho$ and $\beta = 1/\rho - 1$, where $\rho = (2u_0u_2 - u_1^2)/(u_0u_1)$ [Ji et al., 2008; Ladunga, 2010].

To estimate the mixture parameter π_0, one can write $g(0) \approx \pi_0 f_0(0)$, under the assumption that windows that did not contain any reads did not bind the protein of interest. Then

$$\pi_0 = \frac{u_0}{\sum_k u_k} \times \frac{1}{\hat{f}_0(0)}$$

where $\hat{f}_0$ is the above estimated f_0. With the estimated values of π_0, f_0, and g one can estimate the local false discovery rate (cite) for window w_i with n_i reads as $\pi_0 f_0(n_i)/g(n_i)$.

While one sample assays are easier and less expensive to perform, they make the assumption that the background signal follows a negative binomial distribution. This assumption has been observed to be unrealistic in some cases [Kharchenko et al., 2008]. With data from a control experiment, one can model the read counts from the control experiment as $m_i \sim \text{Poisson}(\mu_i)$

[3]We denote with I the random variable and i its values.

and the read counts from the ChIP experiment as $n_i \sim \text{Poisson}(\lambda_i)$. Because, the ChIP experiment windows contain reads due to both noise and signal, we write the rate $\mu_i = \mu_{i1} + \mu_{i0}$, where the rate due to background noise is also set to be $\mu_{i0} = c\lambda_i$. This latter equality assumes that at each window, the background noise in the two experiments differs by a multiplicative constant c. One can write $n_i|n_i + m_i \sim \text{Bin}(m_i + n_i, p_0)$, since a Poisson conditioned by the sum with another Poisson follows a Binomial. The parameter p_0 is the expected fraction of ChIP reads in a background window and is typically estimated from windows with a small number of reads. Given an estimation of p_0, the other quantities can be estimated from the data. The local FDR in this case can be estimated as the ratio between $f_{\text{Bin}}(n|t, \hat{p}_0)/g_{\text{obs}}(n|t) = P(X = n)$ and $X \sim \text{Bin}(t, p_0)$ and $g_{\text{obs}}(n|t)$, the frequency of windows with n ChIP reads among those with t total reads.

2.4 Artificial neural networks

2.4.1 Introduction

Simple versions of artificial neural networks (ANN) as pattern recognition systems were popularized in the late 50s – early 60s [Rosenblatt, 1958]. These early systems were simple two layer associative maps between input and output (layers), but they could map similar input configurations into similar output configurations. This allowed them to generalize reasonably well on input they had never seen before. This same property posed a limitation of these systems to recognize similar examples (e.g., inputs with the same label in the case of a classification problem) with a very different representation in the input. A classic example is the inability of a perceptron solve the XOR problem, where examples 00 and 11 have label 0 and examples 01 and 10 have label 1.

Two layer ANNs have to learn a mapping based on the input encoding provided by the external world. This is a limitation that manifests itself in a large and well studied class of problems [Minsky & Papert, 1969]. Adding the ability to create internal representations through one or more *hidden* layers, removes this limitation [Rumelhart et al., 1986]. This simple observation allowed a general use of ANNs.

The unit of computation in both an organic and artificial neural network is the neuron. Organic neurons communicate with other neurons though dendrites to axon connections called synapses. The efficiency with which a synapse transmits an impulse depends on physiological properties of the synapse. On the other hand, artificial neurons are far more simple – a typical artificial neuron, for example, implements no synaptic delay, and communicates with impulses [Neumann, 1958].

The first artificial neuron model, proposed by McCulloch and Pitts in 1943 [McCulloch & Pitts, 1943] is of the form

$$\begin{aligned} y &= s\left(\sum_j^p x_j w_j - u\right) \\ &= \begin{cases} 1 & \text{if } b + \sum_j^p x_j w_j \geq 0 \\ 0 & \text{else} \end{cases} \\ &= s\left(\sum_j^{p+1} x_j w_j\right) \end{aligned}$$

The first equation shows the model in the canonical form. The unit has weights w_j for each input x_j and an activation threshold u. The output y is a binary value which is on whenever the weighted sum of the inputs is greater than u. Typically, a bias term is added to the computation to cancel out u (second equation). For convenience, one can think of the unit as having $p+1$ inputs where one of them is always one. This trick incorporates the bias term into the summation of the third equation.

The step function s in (2.1) is called the activation function and is one of the many choices of activation functions. For example, a continuous squashing function such as the sigmoid $\sigma(x) = 1/(1 + e^{-x})$ allows for non-binary activation values at the output of the neuron. Other functions that result in faster training or with desirable application-specific properties have also been proposed (reviewed in [Montavon et al., 2012]).

2.4.2 Feed forward networks

Neurons can be arranged in a number of topologies to create networks. The topology refers to (1) the allowed connections between neurons, (2) the designation of select neurons as input units, and (3) the designation of select neurons as output units for the network. The ability of the network to learn concepts related to a problem depends on the topology, so designing network topologies is a problem specific task.

Units in feed forward networks are organized in layers and connections go from one (lower) layer to an upper layer. The topology allows for skip connections, even though these do not change the expressibility of the network. Each neuron combines the inputs and a bias term and then applies an activation function as in Equation (2.1).

$$y = f\left(b + \sum_{j}^{p} x_j w_j\right) \tag{2.1}$$

Following the notation in [Hastie et al., 2009], the three layered network for K-class classification in Figure 2.4 can be modeled as

$$\begin{aligned} Z_m &= \sigma(\alpha_{0m} + \alpha_m^T X), m = 1, \ldots, M, \\ T_k &= \beta_{0k} + \beta_k^T Z, k = 1, \ldots, K, \\ f_k(X) &= g_k(T), k = 1, \ldots, K, \end{aligned} \tag{2.2}$$

where $Z = (Z_1, \ldots, Z_m)$ and $T = (T_1, \ldots, T_k)$. The output function $g_k(T)$ is typically taken to be a softmax $g_k(T) = \exp(T_k)/\sum_{l=1}^{K}(\exp(T_l))$, which produces a probability distribution over the outputs.

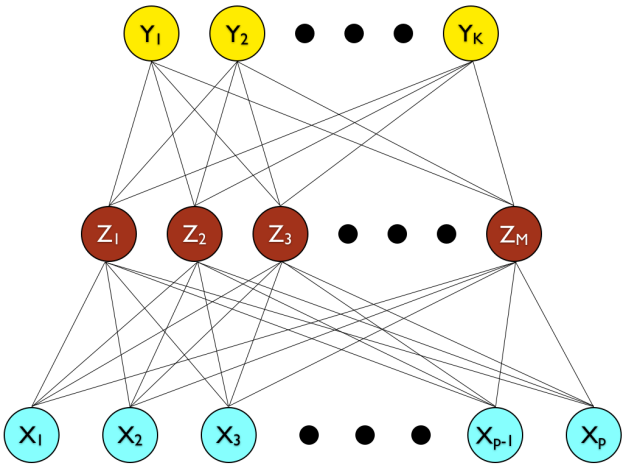


Figure 2.4: A three layer fully connected feed forward net.

To train an ANN, we try to fit the parameters (call them all θ) to the training data by minimizing a cost function $R(\theta)$. Typical choices for R are the L_2 norm or, for classification problems, a cross-entropy measure $-\sum_{i=1}^{N}\sum_{k=1}^{K} y_{ik} \log f_k(x_i)$. There are many choices for R, and in general iterative optimization methods can be used to fit the parameters. The most popular is the back-propagation algorithm. For the network in Figure 2.4 and a sum-of-squared cost function

$$R(\theta) = \sum_{i=1}^{N}\sum_{k=1}^{K}(y_{ik} - f_k(x_i))^2 = \sum_{i=1}^{N} R_i(\theta)$$

an online optimization method updates the parameters at iteration $r+1$ as

$$\beta_{km}^{(r+1)} = \beta_{km}^{(r)} - \gamma_r \sum_{i=1}^{N} \frac{\partial R_i}{\partial \beta_{km}^{(r)}} \tag{2.3}$$

$$\alpha_{ml}^{(r+1)} = \alpha_{lm}^{(r)} - \gamma_r \sum_{i=1}^{N} \frac{\partial R_i}{\partial \alpha_{ml}^{(r)}} \tag{2.4}$$

after observing each input example. The learning rate γ_r at step r scales the updates. The quantity $\frac{\partial R_i}{\partial \alpha_{ml}}$ (resp. $\frac{\partial R_i}{\partial \beta_{km}}$) can be interpreted as the product between the error at the output (resp. hidden) layer and the activation of unit z_m (resp. x_l). The stochastic version of the back-propagation algorithm computes the network prediction for a training example in a forward pass. Then in a backward pass uses the label of the example to compute the errors at each unit [Rumelhart et al., 1986].

2.4.3 Convolutional Neural Networks

A notable architecture is a convolutional NN (CNN). It contains one or more Convolutional Pooling layers followed by a feed forward layer and a regression or logistic regression layer. Convolutional layers perform a convolution of the input with 2D local receptive fields which can be written as

$$f_k(x) = \sigma(x * W_k + b), k = 1, \ldots K \tag{2.5}$$

The dimension of the receptive field W_k is typically small compared to that of the input, resulting in a drastic reduction on the number of parameters. This organization allows for distributed representations of simple input features, such as strokes in an image or peaks in a signal track, that can occur anywhere in the input. Furthermore, the representation is equivariant to translations, that is, translated input features correspond to translated features in the feature maps. A high number of receptive fields will create a rich hidden representation of the input. Optionally, a convolutional layer is followed by a max-pooling layer of dimensions p, q.

Subsequent layers are fully connected sigmoid feedforward layers like that in Equation (2.1).

Sparse, convolutional layers and max-pooling are at the heart of the LeNet family of models. While the exact details of the model will vary greatly, the figure below shows a graphical depiction of a LeNet model.

The lower-layers are composed to alternating convolution and max-pooling layers. The upper-layers however are fully-connected and correspond to a traditional MLP (hidden layer + logistic regression). The input to the first fully-connected layer is the set of all features maps at the layer below.

2.4.4 Modern ANNs and Representation learning

The richness of functions represented by a learning system depends on the number of computational units or parameters included in the system. Systems that can represent many functions compactly (i.e., with a small number of computational units) are clearly desirable; from a statistical point of view, these systems will tend generalize better [James et al., 2013] and from a computational point of view, they will tend to need a smaller number of training examples to train. Theoretical results from circuit complexity [Allender, 1996] and experimental results show that deeper (i.e., more hidden layers) ANNs result in more compact systems. Specifically a $k+1$ layer network might require an exponential number of units to be represented by a k layer network (See Chapter 2 of [Bengio, 2009] for a detailed description of these arguments).

While depth allows for more efficient deep models, it also increases the difficulty to train them. Until recently, deep ANNs performed poorly because their learned weights corresponded to local minima or plateaus [Glorot & Bengio, 2010]. This meant that the gradient based optimization algorithms tended to get stuck in these regions apparently because they were not powerful enough to find better solutions and because the initial weights of the network were far from regions with good solutions. Finding good initial weights, *pre-training* the weights of the network, resulted in networks that generalized much better. Pre-training consists in a layer-wise unsupervised training of weights [Hinton et al., 2006; Bengio et al., 2007].

The success of pre-trained deep ANNs had repercussions in the whole field of machine learning and artificial intelligence. One of the first steps in a machine learning problem is finding a data transformation that highlights as many relevant features as possible with minimal loss of information. The success of many machine learning methods depends heavily on the input data transformation. Therefore, systematical approaches to defining criterions for good transformations and designing systems for obtaining them are very important. The goal of representation learning is to define criterions for the quality of representations and to build good (under their criteria) representations.

A typical representation learning system learns a sequence of data transformations in an unsupervised way. Architecturally, representation learning systems are deep stacks of layers of neural units, where the representation given by a layer is the input of the next layer.

The objective of each transformation layer is to obtain representations that are *distributed*, *invariant*, and with *disentangled factors of variations* (reviewed in [Y. Bengio, 2013]). Distributed representations are expressive in that they require a small number of parameters to represent a large number of input regions. This property implies that the components of distributed representations can vary independently. On the contrary "one-hot" representations require n components to distinguish n input regions (or clusters). Invariance refers to representations that have reduced sensitivity with respect to variations of the input that is irrelevant to the task at hand. Finally, good representations require disentangling of sources of variation in the input data. Obtaining representations with disentangled factors of variation requires large amounts of un-labeled data, as typically, only a small number of factors change between consecutive input examples.

Chapter 3

Feature extraction in deep models

3.1 Introduction

Deep learning methods learn hierarchies of features in hidden layers, so that features of a layer are learned from those of the layer below. Recent algorithmic developments in initialization and training [Hinton et al., 2006; Bengio et al., 2007] have made it possible to train deep models of more than 2 or 3 layers – although deep convolutional networks could be trained before. Not only have deep models been successful in a of machine learning tasks over several areas of research (in vision, audio and natural language processing), but the hidden representations learned by them have been a topic of active study.

Insights on the nature of hidden representations in a deep model has been used to improve the training algorithms, the initial conditions, and the architecture of the models. An important related problem is to identify input features that are representative of the knowledge of the trained network or that exhibit the highest predictive power. This is an aspect of deep learn-

ing architectures that remains unexplored, partly because neural nets are used mainly in image or object recognition, where important features can be examined visually, and partly because of a strong interest in the predictive power of the models. Currently, it is very common to display the *weights* of the model, which convey little information on the relevant input features. The purpose of this chapter is to explore this problem and suggest a strategy for relevant input feature extraction.

We will write a k–layer ANN architecture as a series of compositions of functions on input vector X

$$\mathcal{N}(X) = f_{W_k} \circ f_{W_{k-1}} \circ \ldots \circ f_{W_1}(X) \tag{3.1}$$

where all functions are vector valued and defined on vector inputs. The functions in Expression (3.1) are architecture specific, but they usually take the form $\sigma(X^T W)$, with σ a squashing function like the sigmoid $\sigma(x) = 1/(1+exp(-x))$ called the activation function. In Expression (3.1), the input X is mapped in a series of intermediate representations, whose components are non-linear combinations of some (locally connected) or all (fully connected) components of the previous representation. These dependencies are also specified by the network architecture. The parameters $\{W_i\}_{1 \leq i \leq k}$, or *weights*, can be learned from training data in both a supervised or an unsupervised fashion.

As the parameters are learned, the intermediate representations retain only the information needed for the task at hand and become easier to classify or regress. This property makes the hidden representations interesting in their own right [Bengio, 2009]. For example, it has been observed that networks with large enough low levels are able to learn data transformations that exhibit feature de-correlation or sparsification. Furthermore, Lee et al. [Lee et al., 2009] proposed a deep model that learns representations from image data in an unsupervised way.

3.2 Relevant feature extraction from ANNs

The problem of finding a parsimonious set of input features that maximizes the probability of a given output class is related to understanding the activity of arbitrary neurons in a deep NN. The latter problem has been previously considered. A typical way of understanding what a unit in the first layer of a ANN is computing, is to visualize the weights of the first layer. Such visualizations give a qualitative idea of the input features that are relevant to the examined unit. Because ANNs have been widely used on image data, this technique provides images of weights that are easily interpretable. Published weight visualizations show edges, or simple strokes.

In the case of multiple layers, high or low weights at the first layer can be canceled or amplified by weight configurations in the subsequent layers. A more general technique that is applicable to the case of multiple layers is the activity maximization [Erhan et al., 2009]. The idea is to find an input vector that would maximize the activity of an arbitrary neuron in the network. For a 1-layered network the input that maximizes the activity of neuron $a_i = f(\sum_j w_{ij}x_j + b_i)$ is

$$\begin{aligned} &\max a_i \\ \text{s.t.,} \quad &||x||^2 = \textstyle\sum_j x_j^2 \leq 1 \end{aligned} \tag{3.2}$$

The solution to this is

$$x_j = \frac{w_{ij}}{\sqrt{\sum_k w_{ik}^2}}$$

In a multilayered ANN this is an optimization problem that can be approached, for example, with gradient ascent. In other words, start with a random vector x and update it by moving in the direction of the gradient $\frac{\partial a_i}{\partial x}$. This method has been reviewed by Erhan et al. [Erhan et al., 2009] and successfully used by Le et al. [Le et al., 2011]. Solving this problem multiple times might yield different solutions depending on the local maximum the

search ends in. This can be acceptable if the solutions are not too different. Another problem is that, depending on the size of the network, this method can be slow.

Another very simple way to show what the model has learned is to examine the best scoring examples. While looking at the best examples, might give clues to generalizable principles, one cannot rely on this method as an automatic feature extraction procedure. Another option is to compute the average of correctly classified examples weighted by the posterior probability assigned by the model. This method is more systematic, but can suffer from the small sample size and thus extract features that are in fact noise.

3.3 Convex optimization based method

The methods discussed in the previous section provide a qualitative measure of the of what an arbitrary unit in a ANN computes. Here we propose an approach that

(1) finds a deterministic set of features based on the weights of the model

(2) guaranties a quality and parsimoniousness of extracted features

(3) is efficient

The ideas of this section have been implemented in a Python library called dimer available at `https://bitbucket.org/gertidenas/dimer`. Furthermore, unless otherwise stated, we will consider the ANN in Equation (3.1) with convex activation functions and no pooling operations. The ANN $\mathcal{N}$ is trained on dataset $\mathcal{D} = \{(X^{(i)}, Y^{(i)}) : i = 1, \ldots, n\}$ where the outputs are discrete values $Y^{(i)} \in \{1, \ldots, L\}$. For ease of notation we will denote $f^{(i)} = f_{W_i} \circ \ldots \circ f_{W_1}$ and $f^{(0)}$ the identity function. We will also drop the W_i subscripts when they are clear from the context.

We start with

Definition 3.1. *Define the l-state at the ith layer of $\mathcal{N}$ as*

$$s_{l,i} = f^{(i)}\left(Avg_{j:Y^{(j)}=l}(X^{(j)})\right) \tag{3.3}$$

the ground at the ith layer of $\mathcal{N}$ as

$$g_i = f^{(i)}\left(Avg_j(X^{(j)})\right) \tag{3.4}$$

and the l-membership assigned by $\mathcal{N}$ as

$$m_l = Avg_{j:Y^{(j)}=l}\left(\mathcal{N}(X^{(j)})\right) \tag{3.5}$$

Notice that both $s_{l,0}$ and g_0 remain defined.

In other words the l-state is the representation of the average l-labeled input according to $\mathcal{N}$. The ground, on the other hand, is the representation of the a vector that is the average of all of the training data. If $\mathcal{D}$ is standardized, then the ground is zero. The l-membership vector is such that the lth component of the l-membership vector m_l is the average posterior probability assigned by the $\mathcal{N}$ to the l-labeled input examples, we will use this value as a lower bound on the quality of our extracted features.

Our approach is based on the following idea. In a one layer ANN $\mathcal{N}(X) = f(X)$, for a fixed output label l, we would like an input vector X_l such that

$$\begin{aligned} &\min |\mathsf{X}_l - g_0| \\ &\text{s.t.,} |f(\mathsf{X}_l) - g_1| \quad \geq |m_l - g_1| \end{aligned} \tag{3.6}$$

There are two quantities of interest in this formulation. The quantity being minimized is a measure of the deviation of the feature vector from the ground at input – intuitively, the smaller this value the more parsimonious X_l. The second quantity is the distance from the ground at output – intuitively we

want the probability $P(\mathcal{N}(\mathsf{X}_l) = l)$ to be at least as large as that of the l-membership. Notice the difference between this approach and the activity maximization. In our approach we condition in all the components of $f(\mathsf{X}_l)$.

Because $(\mathcal{N}(\mathsf{X}_l) = l)$ is typically high, we introduce a slack parameter $0 \leq \epsilon \leq \min\{g_{l,k}, 1 - g_{l,k}\}$ which relaxes Inequality (3.6). Increasing ϵ typically results in more parsimonious features X_l and lower class probability. In Figure 3.1 we train a 2-layer model on a binary classification problem and extract features satisfying Expressions (3.6) for various values of ϵ. Increasing ϵ will lower the label probability required for X_l to achieve. If the effect on the class membership is negligible in practice, it is worth exploring different values of ϵ.

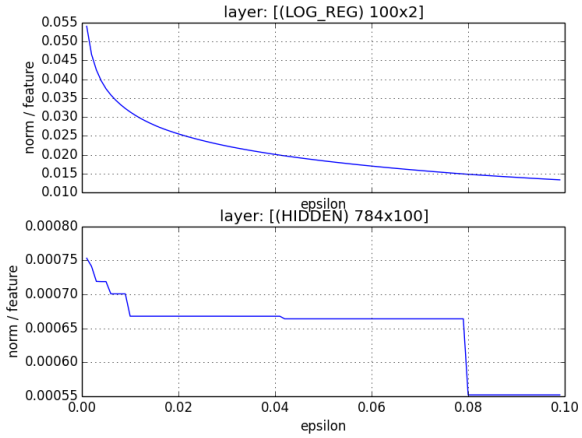


Figure 3.1: Average feature distance from ground as a function of ϵ. We extract features satisfying Expressions (3.6) for various values of ϵ on a 2-layer FF ANN for a binary classification problem. The plot reports the total L_1 distance of extracted features from the ground divided by the size of the layer.

We can now prove the following

Lemma 3.2. *Let $\mathcal{N}$ be a one layer NN. For each label l, there is a feature*

vector X_l *such that*

$$|\mathcal{N}(\mathsf{X}_l) - g_1| \geq |m_l - g_1|$$

Furthermore, $|\mathsf{X}_l - g_0| \leq |s_{l,0} - g_0|$.

Proof. We prove by construction. The formulation

$$\begin{array}{lll} \min & |\mathsf{X}_l - g_0| & \\ \text{s.t.} & \mathcal{N}(\mathsf{X}_l) & \begin{cases} \geq s_{l,1} - \epsilon & \text{if } s_{l,1} > g_1 \\ \leq s_{l,1} + \epsilon & \text{if } s_{l,1} < g_1 \\ = s_{l,1} & \text{else} \end{cases} \\ & \mathsf{X}_l \text{ free} & \end{array} \tag{3.7}$$

has $\mathsf{X}_l = s_{l,0}$ as a feasible solution. Because the activation function is convex $\smile$ and concave $\frown$ for values that are below and above the ground respectively and by Jensen's inequality $m_l \leq s_{l,1}$. □

ConvPool layers are treated in principle identically to fully connected sigmoid layers since convolution can be thought of as matrix multiplication. However, differences arise in the indexes of constraints and the inversion of pooling operations. A ConvPool layer maps F input feature maps of size $M \times N$ into K output feature maps. The dimensions of the output feature maps is $(M-L+1)\times(N-H+1)$ if the weights have size $H \times L$. A schematic view is shown in Figure 3.2. Let, $X_{[F\times M\times N]}$[1] be the input to the i-th convolutional layer of the network with weights $W_{[K\times F\times L\times H]}$. The formulation is

[1]This notation indicates an array X with dimensions $F \times M \times N$

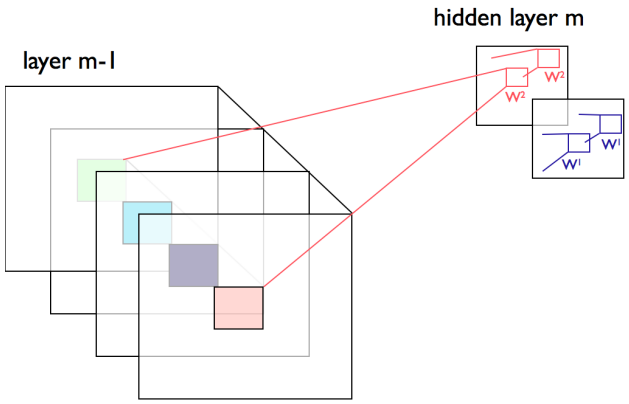


Figure 3.2: A convolutional layer. Here, we show two layers of a CNN, containing 4 feature maps at layer $(m-1)$ and 2 feature maps (h^0 and h^1) at layer m. Pixels (neuron outputs) in h^0 and h^1 (outlined as blue and red squares) are computed from pixels of layer $(m-1)$ which fall within their receptive field in the layer below (shown as colored rectangles). Notice how the receptive field spans all four input feature maps. The weights W^0 and W^1 of h^1 and h^2 are thus 3D weight tensors. The leading dimension indexes the input feature maps, while the other two refer to the pixel coordinates.

$$\begin{aligned} &\min |\mathsf{X}_l - g_0| \\ &\text{s.t. } \sigma(\sum_f X[f] * W_k[f]) \begin{cases} \geq s_{l,1}[k] - \epsilon & \text{if } s_{l,1} > \sigma(s_{l,0} * W_k) \\ \leq s_{l,1}[k] + \epsilon & \text{if } s_{l,1} > \sigma(s_{l,0} * W_k) \\ = c_l[k] - \epsilon & \text{else} \end{cases} \forall k \\ &\quad \mathsf{X}_l \text{ free} \end{aligned} \tag{3.8}$$

where the $*$ operation is a 2d convolution in "valid" mode and $k = 1, \ldots K$.

The solution of (3.8) can be obtained as in Lemma 3.2, however existence cannot be guarantied if the layer applied max-pooling to the activation. One

way to deal with pooling is to "un-pool" the output of the layer. Let $y_1, \ldots y_p$ be the output of the layer before max-pooling and y after max-pooling. With the "un-pooling" method we will use y/p instead of the corresponding component of c_l in formulation (3.8). This substitution has the potential to tighten some of the constraints in (3.8) and thus break its feasibility.

In practice, there are two ways to try to find a solution. First, one can relax the constraints by increasing ϵ. Second, find a relaxed formulation that minimizes the absolute sum of constraint violation. This functionality is provided with some of the LP-solvers [Gurobi Optimization, 2013]. The effectiveness of these strategies depends on the pool size and the variance of c_l.

Before we show how to extend the result of Lemma 3.2 to a multilayer NN, we show in Algorithm 1 how to obtain it. At the end of the loop C_l is the

Algorithm 1 Feature extraction from a stack of layers.

Require: $\mathcal{N}(X) = f_{W_k} \circ f_{W_{k-1}} \circ \ldots f_{W_1}(X)$

Require: l one of class labels $\{1, \ldots, L\}$

1: $C_l \leftarrow \mathcal{N}(E[X_l]) - \epsilon$

2: **for** $i \leftarrow k, \ldots, 1$ **do**

3: $\quad f \leftarrow f_{W_i} \circ \ldots \circ f_{W_1}$

4: $\quad g \leftarrow f_{W_{i-1}} \circ \ldots \circ f_{W_1}$

5: $\quad$ Find vectors X_l s. t., $|f_{W_i}(\mathsf{X}_l) - f(E[X])| \geq |C_l - f(E[X])|$

6: $\quad$ Find $X_l^* \leftarrow \min_{\mathsf{X}_l} |\mathsf{X}_l - g(E[X])|$

7: $\quad C_l \leftarrow X_l^*$

8: **end for**

9: **return** C_l

vector of features in the original input space. The minimization in step 6 reduces the domain of the extracted features at a layer, so that they can be used as targets for the layer below. At the same time it determines the

most parsimonious features. We now prove that the features extracted in Algorithm 1 have high class probability in the following

Lemma 3.3. *Given a multilayer ANN $\mathcal{N}$ and a label l, there is a feature vector $\mathsf{X}_{l,i}, i = 1, \ldots k$ such that*

$$|f_{W_k} \circ \ldots \circ f_{W_i}(\mathsf{X}_{l,i}) - g_k| \quad \geq \quad |m_l - g_k|$$

Furthermore, $|\mathsf{X}_{l,i} - g_{i-1}| \leq |s_{l,i-1} - g_{i-1}|$.

Proof. Consider the following formulation for the i-the layer of $\mathcal{N}$.

$$\begin{aligned} & \min |\mathsf{X}_{l,i} - g_i| \\ \text{s.t.} \quad & f_{W_k} \circ \ldots \circ f_{W_i}(\mathsf{X}_{l,i}) \quad \begin{cases} \geq \mathsf{X}_{l,i+1} & \text{if } s_{l,i} > g_i \\ \leq \mathsf{X}_{l,i+1} & \text{if } s_{l,i} < g_i \\ = \mathsf{X}_{l,i+1} & \text{else} \end{cases} \\ & \mathsf{X}_l \text{ free} \end{aligned} \tag{3.9}$$

and set

$$\begin{aligned} \mathsf{X}_{l,k+1} &= s_{k,l} \text{ if } s_{k,l} \geq g_{k,l} \\ \mathsf{X}_{l,k+1} &= s_{k,l} \text{ if } s_{k,l} < g_{k,l} \end{aligned} \tag{3.10}$$

by Lemma 3.2 the vector $s_{l,i-1}$ is a feasible solution to the formulation of the i-th layer for all $i = 1, \ldots, k$ satisfying the third case of (3.9). Adding the minimization requirement, will produce a solution $\mathsf{X}_{i,l}$ that is not further from g_{i-1}. As a result the constraints (3.9) for the $i-1$th layer can only be more relaxed. Thus, the solution space in the layers below is never reduced. □

Theorem 3.4. *Given a multilayer ANN $\mathcal{N}$ with no pooling operations trained over a dataset $\mathcal{D}$ for a classification problem. Then for each output label l, there is an input X_l to $\mathcal{N}$ such that*

$$\begin{aligned} P(\mathcal{N}(\mathsf{X}_l) = l | \mathsf{X}_l; \mathcal{N}) \quad &\geq \quad Avg_{j:Y^{(j)}=l}(P(\mathcal{N}(X^{(j)}) = l | X^{(j)}; \mathcal{N})) \\ |\mathsf{X}_l - g_0| \quad &\leq \quad |s_{l,0} - g_0| \end{aligned}$$

Proof. Trivially $\mathsf{X}_{l,0}$ from Lemma 3.3. □

If the extracted features $\mathsf{X}_{l,0}$ are informative, then we expect them to correlate with the information gain from input features. Following [Han & Kamber, 2001], the information gain of an input feature A is

$$Gain(A) = I(s_I, s_R) - E(A) \tag{3.11}$$

where

$$I(s_I, s_R) = -s_I/s \log_2(s_I/s) - s_R/s \log_2(s_R/s) \tag{3.12}$$

and

$$E(A) = \frac{s_{I0} + s_{R0}}{s} I(s_{I0}, s_{R0}) + \frac{s_{I1} + s_{R1}}{s} I(s_{I1}, s_{R1}) \tag{3.13}$$

where s_I (resp. s_R) is the count of induced (resp. repressed) examples and where s_{Ii} (resp. s_{Ri}) is the count of induced (resp. repressed) examples with $A = i$. The information gain in (3.11) is a measure of relevance of feature A in discriminating differential expression.

We considered a dataset of the 95 most differentially expressed genes during G1E differentiation (for details on the dataset see Chapter 4). Each training example is a 7×500 real matrix representing a 20Kb (we binned the signal into 40bp bins) TSS-centered neighborhood of TFos for 7 TFs. We trained 3 layer ConvNet with 10 convolutional kernels and a hidden layer of size 200 on this dataset. The score of the extracted features shows a strong alignment with the information content (See Figure 3.3).

3.4 Conclusion

ANNs are notorious for being hard to tune and train [Montavon et al., 2012] and finding the right parameters and meta-parameters requires a lot of intuition and tricks. Furthermore, the layer wise mapping of non-linear combinations of inputs make it hard to interpret the role of input features on the

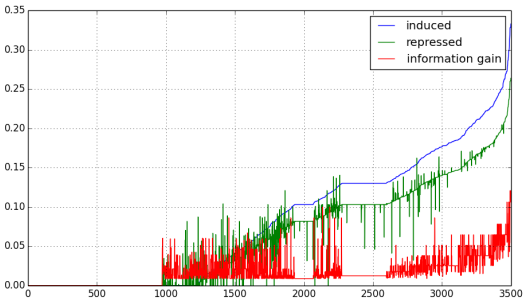


Figure 3.3: The plot shows the information gain as defined in Expression (3.11) and the score of the extracted features X_{I0} and X_{R0}. The data is sorted by increasing induced score, repressed score, and information gain.

output. Often, ANNs are used as black boxes for prediction. We presented a systematic method for extracting knowledge from a deep ANNs back in the input space, by using the weights of the trained model to infer layer-wise parsimonious input features with guarantied margin on the targets.

Chapter 4

Deep models for gene regulation

4.1 Introduction

Thus far, we have described deep ANNs and their use in supervised and unsupervised modeling. In the previous Chapter, we proposed an algorithm for extracting features based on their importance to predicting samples. In this Chapter, we use ANNs and our feature extraction method for modeling and interpreting functional genomics data. First, we consider classifying a set of differentially expressed genes during G1E differentiation and uncover a number of global regulatory mechanisms and interplays between the participating TFs supported by recent empirical studies. Next, we perform quantitative gene expression prediction from 41 ChIP-Seq assays targeting TF binding locations in in human K562 cells. These cases are a proof of concept for the use of deep ANNs as a supervised approach in represent biological knowledge in an interpretable way.

The modeling framework we present here relies on sequence census methods described in Section 2.1. Sequence census methods based on Chromatin Im-

munoprecipitation (IP) followed by high throughput sequencing (ChIP-Seq) are now available for multiple cell types, TFs, and other epigenetic features. Studies that integrate multiple types of IP based assays have had considerable success in the identification of regulatory elements and *cis*-regulatory modules [Heintzman et al., 2007; Pique-Regi et al., 2011], prediction of gene expression [C. Cheng et al., 2012], and a variety of interdependencies between TFs and histone histone modification profiles [Yu et al., 2008] or DnaseI Hypersensitive Sites and Gene expression [Thurman et al., 2012]. Computational analysis of TF occupied segments (TFos) identified by ChIP-Seq methods have also improved upon the performance of approaches based on sequence (including motifs and evolutionary conservation) data only (Reviewed in [Hardison & Taylor, 2012]).

Typical computational approaches have been based on Hidden Markov Models [Taylor et al., 2006], Bayesian Networks [Yu et al., 2008], and Random Forests [C. Cheng et al., 2012]. A paradigm that has been foreign to genomics studies is based on deep ANNs. Deep ANNs have been almost impossible to train for a long time. However, recent developments in the field and advances in computational power have made ANNs the best performing models in a number of machine learning competitions [Y. Bengio, 2013].

We have implemented a framework, dimer, that allows for functional genomics data aggregation, model design and model interpretation. `dimer` provides an environment for the definition and training of deep models. It implements typical configurations of supervised and unsupervised architectures and allows for the design of custom ones. `dimer` manages datasets, model parameters, and training monitors. `dimer` can also perform model introspection. Documentation and source code is available at `https://bitbucket.org/gertidenas/dimer`.

In a typical workflow, `dimer` accepts tracks of real signal or discrete features from a number of genome regions. Tracks span genome regions of lengths

that can vary from several Kb to several Mb. For example, they can be transcription start site-centered neighborhoods or a sliding window along the genome. Labels or quantitative values can be assigned to each region in the case of a supervised problem. `dimer` will preserve topological information on the dataset – e.g. the organization of ChIP-Seq signal in tracks – which can be very informative. A generic training dataset in `dimer` is a set of input-label pairs $\mathcal{D} = \{(X^{(i)}, Y^{(i)}), i = 1, \ldots, N\}$. The i-th input example $X^{(i)}$ is a two dimensional matrix where value $X^{(i)}_{jk}$ represents the value of the j-th input track, at the k-th position with respect to a genomic mark (Figure 4.1). Given a model architecture, `dimer` can be used to pre-train its layers under some cost function and then fine tune the weights with supervised gradient descent.

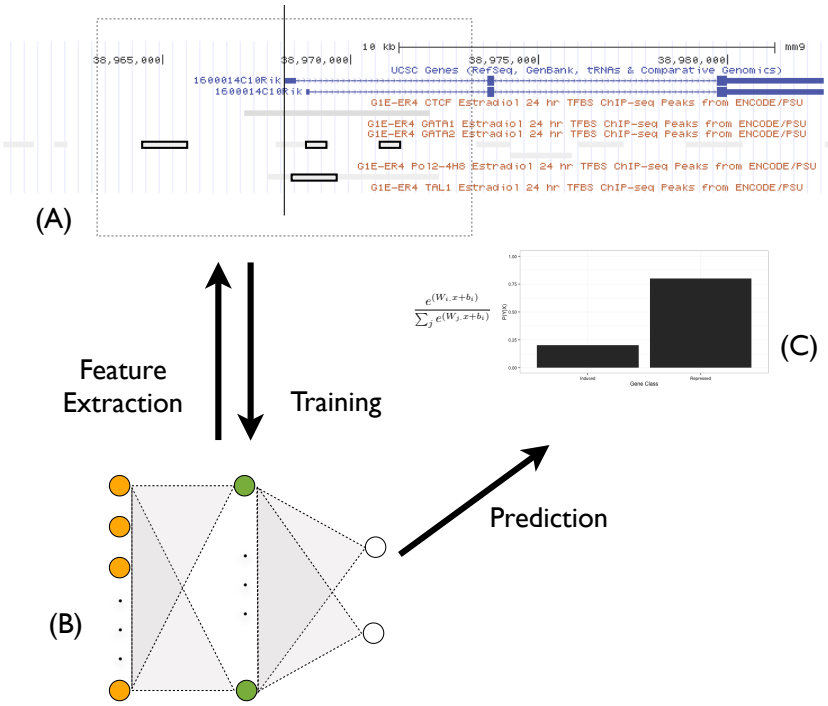


Figure 4.1: A typical workflow using `dimer` for modeling gene regulation: **(A)** TF occupancy sites on a TSS-centered neighborhood. The vertical black line indicates the TSS and the dashed rectangle a 10Kb neighborhood. The corresponding input example, in this case is a matrix X_{jk} with $1 \leq j \leq 6$ indicating the index of the track, and $-5Kb \leq k \leq 5Kb$ indicating the position with respect to the TSS. As a training example, a label or quantitative value Y could be also associated. **(B)** Schematic of a fully connected feed-forward NN with a single layer. The input X is shown in orange units. The output layer uses the softmax function to produce a unit norm vector whose dimension equals the number of output classes. **(C)** The output of the final layer of the NN interpreted as a class probability. Data from TF occupied segments are fed to the NN in the training phase (Training arrow). A trained model, can be used to identify relevant input features back in the input space (Feature Extraction arrow). The model can also be used to predict the expression of a gene based on its TF occupancy profile (Prediction arrow).

4.2 Differential gene expression modeling

4.2.1 The G1E biological model and data

The G1E biological model allows for the controlled differentiation of pluripotent GATA1^{-} erythroid cells (G1E). G1E cells are derived from mouse embryonic stem cells and can be induced to further differentiation into the G1E-ER4 sub-line [Weiss et al., 1997]. G1E-ER4 cells resemble normal erythroid progenitor cells, with the exception of estrogen activated GATA1 receptor (GATA1-ER). This receptor allows control of GATA1-ER expression by treating G1E-ER4 cells with estradiol (E2), which in turn will un pause differentiation (Figure 4.2). Resemblance of G1E sub-lines (G1E-ER4 and G1E-ER4+E2 cells) with normal erythroid progenitors and differentiating erythroblasts respectively makes the G1E-ER4 differentiation a very good model for normal protheoblast differentiation in mouse [Wu et al., 2011].

Because Erythroid differentiation depends heavily on the GATA1 TF [Weiss et al., 1994], GATA1-ER release on G1E-ER cells is an important event that causes changes on the TF binding locations and the chromatin structure as the cell differentiates into G1-ER4+E2 (See Figure 4.3). Whether the former induce the latter or vice-versa is still an open question, however empirical studies of the G1E model report that chromatin state profiles and accessibility remains largely unchanged during GATA1-induced erythroid differentiation. This suggests a scenario where the chromatin structure of G1E cells is established prior to G1E cell commitment and subsequent changes are limited to TF binding locations [Wu et al., 2011].

On the above assumption, we consider, in addition to GATA1, the ChIP-Seq profiles of three other important TFs in the G1E model: GATA2, TAL1, and CTCF. The GATA2 TF, a protein similar to GATA1 that recognizes similar motifs (WGATAR) and plays an important role as a regulator of the differentiation process [Yamamoto et al., 1990]. The TAL1 protein which

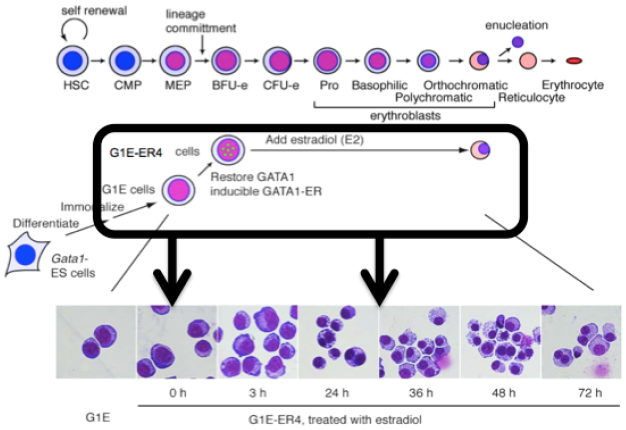


Figure 4.2: The G1E biological model. The diagram shows the differentiation process from GATA1 *null* Embryonic Stem Cells (GATA1$^-$ ESC). GATA1$^-$ ESC are induced to differentiate. GATA1 is restored on G1E cells and differentiation continues to G1E-ER4-E2. The G1E and G1E-ER4-E2 time points (0h and 30h respectively) are shows by the black arrows. We use ChIP-Seq data targeting a total of four TFs in these two time points as the input to our model. Figure adopted with permission from Dr. Ross Hardison.

is known to form multi protein complexes with both GATA1 and GATA2 [Wadman et al., 1997]. The CTCF protein, a highly conserved zinc finger protein implicated in diverse regulatory functions, including transcriptional activation/repression, insulation, imprinting, and X chromosome inactivation [Phillips & Corces, 2009].

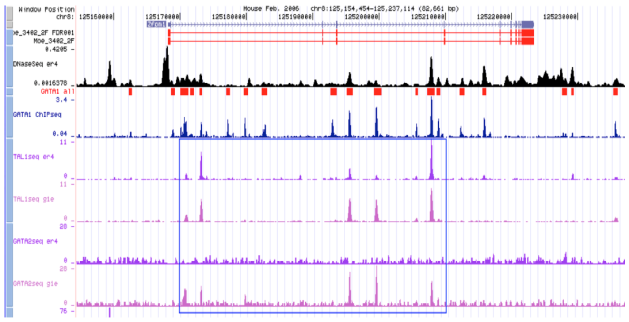


Figure 4.3: A snapshot from the PSU genome browser on a neighborhood centered at the Zfpm1 gene, which is induced during G1E differentiation. The gene track shows the gene position and name, the DnaseSeq track shows DNA accessibility, whether the rest of the tracks show raw occupancy signal of the region for 4 proteins in G1E and G1E-ER4-E2 (light and dark magenta respectively). Higher signal represents higher occupancy. This view shows how differentiation is accompanied by TAL1 reduction at particular sites and GATA1 enrichment in intronic regions.

We build a 3-layer ConvNet where the bottom layer is a convolution layer with several banks (Figure 4.5 shows two banks as W_k) of 2 dimensional layers of connections. Follow, a fully connected feed forward layer and a logistic regression layer with two outputs representing the state of the gene: *Induced* or *Repressed*. In this model architecture:

(i) with kernels spanning all the input tracks across a small genomic region, co-localized features across the all input tracks can be extracted

(ii) each bank will highlight simple features, such as combinations of peaks,

that can possibly occur along the entire input track

(ii) there is substantial parameter sharing in the convolutional layers. The number of parameters of a convolutional layer is the product of the number of kernels and the receptive field shape.

We train by minimizing the cost function

$$\mathcal{L}(\theta = \{W, b\}; \mathcal{D}) = \sum_{i=0}^{|\mathcal{D}|} \log(P(Y = y^{(i)}|x^{(i)}; \mathcal{D})) + \lambda_1||\theta||_1 + \lambda_2||\theta||_2 \quad (4.1)$$

computed over the training set $\mathcal{D}$. The validation contains 20% of the examples, which are randomly selected prior to each training and never showed to the model. We train the model for 10 epochs (an epoch is a sweep through all of the training examples) stop training if the validation cost would increase in all next 10 epochs. The model is trained on 600 (7×500) TSS-centered neighborhood profiles and achieves 70% accuracy on 5-fold cross validation.

The parameters of the model are the weights of the layers, θ, and a variable learning rate ρ_i indexed by the training epoch. We optimize the weights by batched stochastic gradient descent (batch size of 20) and update the learning rate at the nth epoch to $\rho_n = \rho_0 \times \sqrt{(n-1)/n}$ if the cost on the training set increased by 1%. We choose $\rho_0 = 0.2$, but the results are robust to this choice as the learning rate gets quickly adjusted.

The meta-parameters are: λ_1, λ_2, and the size of the layers. Using the above training strategy, we searched heuristically the meta-parameter space for training neural nets. First, we determined ranges of number of kernels in the convolutional layer and the size of the top layer by exploring the grid

$$\text{nr. kernels} = \{2, \ldots, 80\} \times \text{top size} = \{20, 40, 80, 160\} \times \text{rfield size} = \{(7, 21)\}$$

we noticed that both the un-regularized cost in Expression (4.1) and the miss-classification rate increases in models with more than 20 kernels (See Figure 4.4), suggesting that those models are likely to over-fit the data.

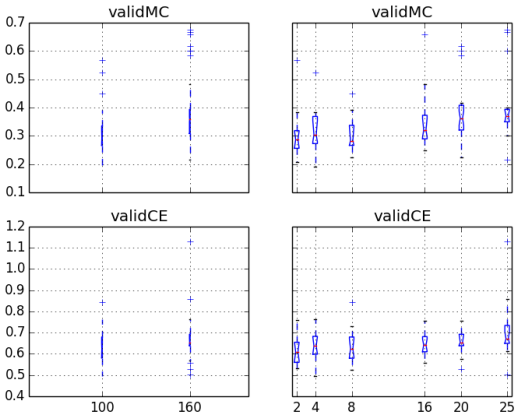


Figure 4.4: We trained a 3-layer convolutional network on our full tr dataset 20 for each choice of the number of kernels (right column) and the top layer size (left column). We then reported miss-classification rates (top row) and the cross-entropy (bottom row) for the validation set at the last training epoch.

4.2.2 Feature extraction

Following the techniques presented in the previous Chapter, we compute the most parsimonious input to the network that produces a given gene class with a given probability. We further process features quantitatively and qualitatively. For each extracted feature we consider its pair of score values with respect to the *induced* and *repressed* gene classes. The class of the feature, given the pair of scores is shown in Table 4.1 and the score of the feature is the average of the magnitudes in all cases.

To interpret the extracted features, we perform association tests between features of different tracks. Given a pair of query and target tracks of extracted features we ask how are the query features positioned with respect to the target features by evaluating the expected and observed relative distance between the two (Figure 4.6). Following Favorov et al. [Favorov et al., 2012], let d_i be the relative distance of a query feature from neighboring target

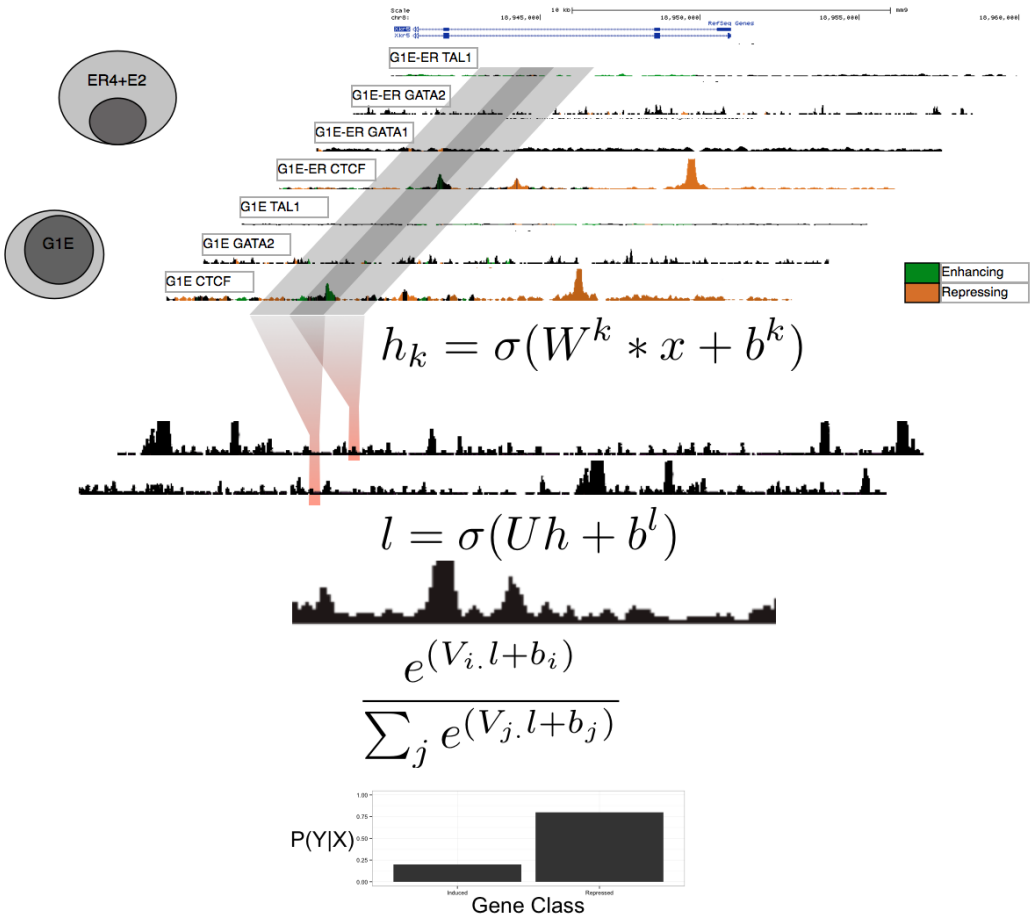


Figure 4.5: Schematic view of a 3-layer CNN. The bottom (top in figure) layer is a convolution layer with several banks (parametrized by 2 kernels $\{W^k, b^k\}$, in figure) with 2D layers of neurons. Each output value of the convolutional layer depends on a neighborhood of input from all the tracks centered at the corresponding position. Follow, a fully connected sigmoid feed forward layer parametrized by $\{U, b^l\}$ and a logistic regression layer parametrized by $\{V, b\}$ with two outputs representing the state of the gene: *Induced* or *Repressed*.

Score		
induced	*repressed*	classification
positive	negative (zero)	E strong (weak)
negative	positive (zero)	R strong (weak)
positive	positive (negative)	TFos strong (weak)
zero	zero	na

Table 4.1: We classify extracted features based on their score for induced and repressed genes. Those that exhibit enrichment on induced (resp. repressed) genes *and* feature depletion for repressed (resp. induced) genes are *strong* features associated to enhancing (resp. repressive) function. Those that show enrichment or depletion on one class of genes, but show no significant depletion (resp. enrichment) on the other class are *weak*.

features as shown in the top-right panel of Figure 4.6. Then, the measure

$$S = \int_0^{0.5} |ECDF(d) - ECDF_{ideal}(d)| \mathrm{d}d \tag{4.2}$$

where $ECDF_{ideal}(d)$ is the cumulative distribution of distances in the case of perfect independence of query and target features, is a correlation (up to normalization) measure.

We extract a parsimonious set of strong 56 repressor and strong 391 enhancer input features, and another 3053 features less directly associated to differential gene expression (Table 4.2). We map these features to the genome as TF-specific tracks of scored regions relative to the TSS (Figure 4.8). Importantly, the function attributed to these features is associated to the whole set of genes.

Finally, association analysis between all pairs of tracks shows a number of patterns in line with results in the literature. We summarize these as follows

- GATA1 and TAL1-ER4 are by far the most important players on differential gene up regulation during erythroblast differentiation, account-

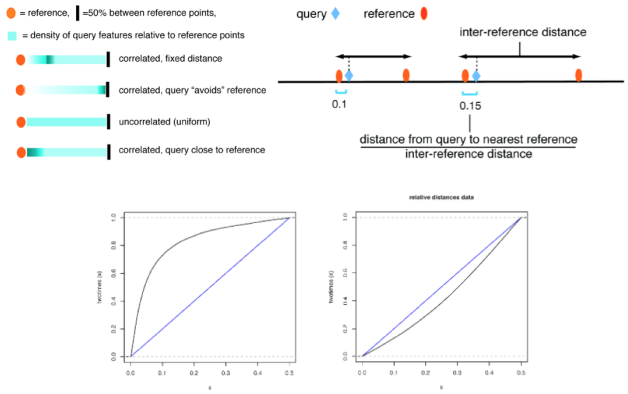


Figure 4.6: Distribution of query features with respect to the target features [Favorov et al., 2012]. The top figures show several types of distribution. The bottom plots (left to right) show the empirical cumulative distribution of query features positively and negatively associated with target features in black. The blue line indicates the ecdf. of non associated features. Figure adopted from the GenometriCorr documentation.

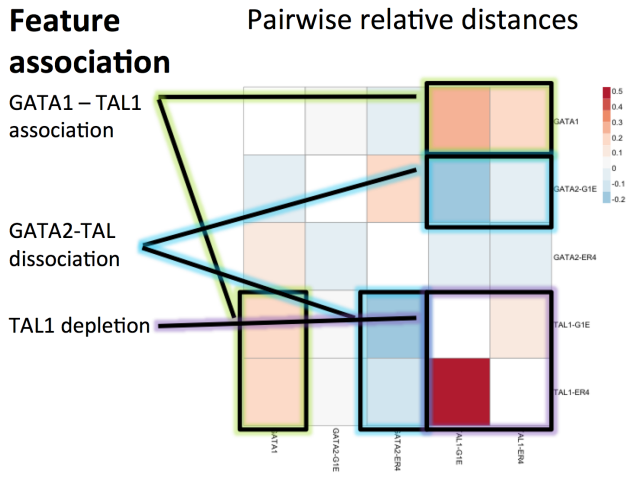


Figure 4.7: Correlation values for the relative distance statistic described by Equation (4.2) color-coded in a matrix. Notice that the matrix is not esymmetric specially in the case of a different number of features between query and target.

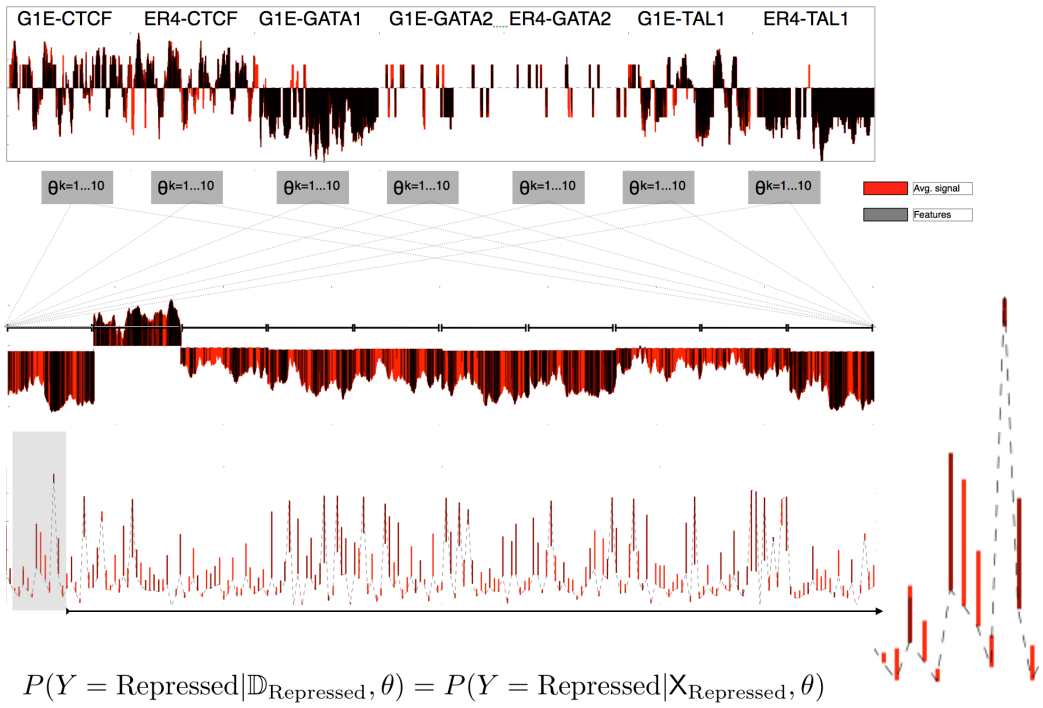


Figure 4.8: Feature extraction on the network shown in 4.5. The method extracts features top-down starting from a given gene class (*Repressed in the Figure*). The input layer is shown on the top with the tracks concatenated. The convolutional layer maps the input into 10 feature maps which are further mapped to a hidden state (bottom track in figure). An input representation shows in red the average signal for *Repressed* genes and in black the extracted features. It can be seen (blown up region on the right) that extracted features are either a subset or a partial match of the features obtained by the average signal.

	E-	E+	na	R-	R+
CTCF_ER4	64	0	175	235	26
CTCF_G1E	93	1	355	50	1
GATA1_ER4	301	82	105	12	0
GATA2_ER4	4	0	471	25	0
GATA2_G1E	16	4	453	26	1
TAL1_ER4	96	304	100	0	0
TAL1_G1E	28	0	341	103	28

Table 4.2: Catalog of feature coverage (measured in 40bp bins) associated to differential gene expression during erythroblastoid differentiation. Feature classes are defined above, while 'na' stands for 'no activity'

ing for 38.57% and 40.28% of the enhancer signal on both cell types for induced genes respectively.

- We notice that repression associated features are dominated by CTCF signal in ER4 (CTCF R+ features account for 100% in ER4).

- GATA1 enhancer features show significant overlap with TAL1_ER4 features (Scaled absloute distance tests p-val < 0.03 and < 0.03 for TAL1 over GATA1 and viceversa respectively) suggesting co-operation of the two factors for gene induction during differentiation [Y. Cheng et al., 2009; Wu et al., 2011]. See also Figure 4.7.

- The concentration of strong GATA1 enhancer features proximal to the TSS, shows a 28.05% increase with respect to global average (from 8.2 features / Kb to 10.5 features / Kb), suggesting a preferential enhancing activity of GATA1 proximal to the TSS.

- We were able to discover that features extracted from the model support the GATA switch mechanism by which, a global replacement of

GATA2 by GATA1 takes place during differentiation [Wu et al., 2011] (Scaled absolute distance test p-val < 0.03). See also Figure 4.7.

The model presented here showed that ANNs can be successfully used to model and interpret functional genomics data in a classification setting. In the next section we adopt an almost identical ANN for regression of gene expression levels.

4.3 Gene expression prediction from TF occupied segments

4.3.1 Data

The human ENCODE project has performed ChIP-Seq experiments targeting a total of 400 TFs across a total of 120 cell lines. These measurements are not distributed uniformly across cell lines, with the best coverage being in Tier 1 cell types. We choose in our experiment GM12878 lymphoblastoid cells and all 30 TFos measurements available for this cell type.

We use raw TFos as predictors of gene expression. Of the three (RNA-Seq, RNA-PET, and CAGE) technologies used by ENCODE to measure gene expression, we choose CAGE as it seems to show better correlation with TFos [C. Cheng et al., 2012; ENCODE Project Consortium et al., 2012]. For the same reasons we choose whole cell transcripts from the PolyA+ RNA extraction protocol. We sample the TFos peak signal around a 2Kb TSS centered window and average it across 100 bp bins for a randomly chosen sample of 10000 GENCODE V7 annotated TSSs. Thus, similarly to the G1E data (Section 4.2.1) each input example i is a 2-dimensional matrix where X_{jk} represents the activity (i.e. peak enrichment) of protein j, k bins away from the TSS of gene i.

4.3.2 Regression model

The regression model is similar to the one for differential gene expression classification in the G1E model in Figure 4.5. In this model the top layer is a regression layer followed by a sigmoid nonlinearity, but the cost function and the training procedure remains the same. Minor changes to the model in Figure 4.5 to adapt it to a quite different problem show the versatility of ANNs. We find that a model with a single Convolutional Pool layer performs best.

To observe the effectiveness of TFos as a distance from the TSS, we impose the Convolutional Pool layer filters to cover the whole TSS-centered window. Not surprisingly, the trained filters of the top layer show a high value for signal 1Kb upstream of the TSS. Importantly, we observe a high weight magnitude at +2Kb and +3Kb suggesting, contrary to previous studies [C. Cheng et al., 2012], that TFos in these locations are helpful in predicting expression (Figure 4.9). The model was able to explain 66% of the variability in the data. and the correlation between the predicted and actual outputs is 82% (Figure 4.10).

4.4 Conclusion

In light of a number of algorithmic breakthroughs, deep neural networks and other classes of deep models have taken the lead in a number of machine learning competitions and have found wide use in predictive settings and in tasks that are of special interest to artificial intelligence. However, their use in the genomics community has been limited. We propose `dimer`, a framework for the design and analysis of deep neural networks. The `dimer` framework also implements a method for model introspection.

As a proof of concept, we analyzed the effect of TF binding in differential

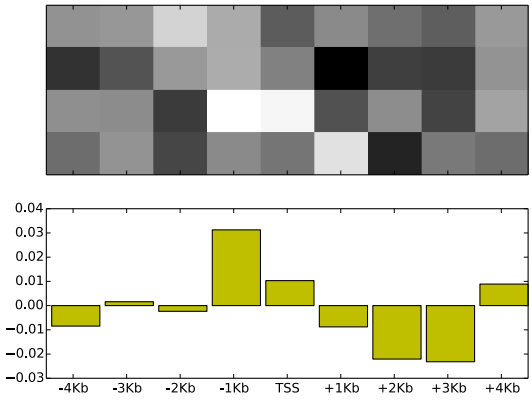


Figure 4.9: The input is convoluted with the weights of the kernels in the first layer. Since, kernels are sliced across the tracks, there is a one to one correspondence between weights in a column and the signal at a fixed distance from the TSS. The top plot shows the average kernel weight in the first layer. The bottom plot is the marginal sum of the values of the weights in the above plot. A bar is the sum of weights of a column. Nonzero values at distal sites, suggest that the model finds predictive value in the signal at those sites.

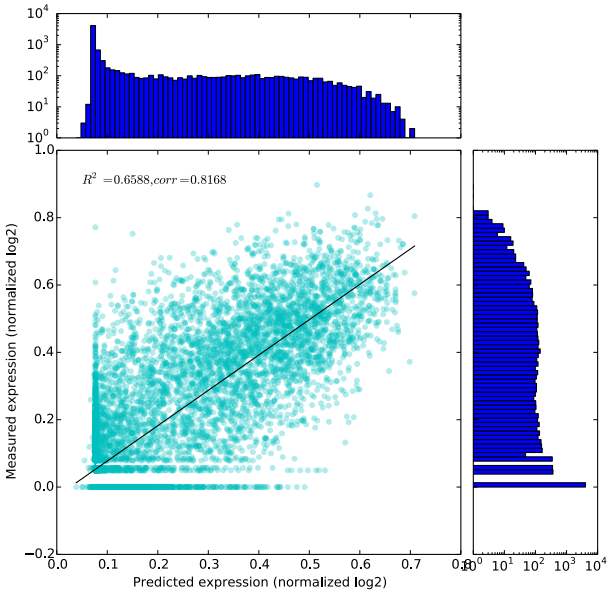


Figure 4.10: Accuracy of the 3-layer CNN model in predicting expression levels as measured by CAGE whole cell PolyA+ reads. We train and test on 10000 random GENCODE V7 annotated TSSs. Each point in the scatterplot is a gene whose coordinates represent the predicted and actual expression levels. The histograms show the marginal distribution of values.

gene expression in the G1E model. We showed how extraction of knowledge from the trained model can help identify important patterns in gene regulation. Then, we used TFos data from the human ENCODE to quantitatively predict gene expression in the Gm12878 cell line and show that our model achieves excellent prediction accuracy.

We believe that our modeling pipeline and the feature interpretation technique will be a useful resource for the modeling and explanation of biological processes that involve high dimensional data.

Chapter 5

Unsupervised modeling of functional genomics data

5.1 Introduction

The availability of complementary functional genomics assays described in Section 2.1 has made it possible to draw the most complete map of the genome to date. This map includes information on many aspects of the nature and function of the DNA, including DNA-protein interactions, chromatin conformation, histone modification, gene expression, and evolutionary conservation. This abundance of data calls for systematic data synthesis methods, for example tagging genome regions according to function or other epigenetic properties – e.g., enhancer, exon, or accessible regions.

In the previous chapter we described supervised methods that model the function of putative CRE or other genome elements. While insightful, these results are limited in scope by the particular datasets they analyze. In the case of IP based methods, it can be argued that the results are applicable to the particular cell condition, TFs considered, and the cell type in which the data were collected. A class of machine learning methods, called unsuper-

vised, do not require previous annotation. Instead, these methods search the parameter space for a configuration that best explains the data under some cost or energy function. Thus, these methods can potentially find patterns of a more global nature.

There have a number of attempts to apply unsupervised methods to multiple functional genomics datasets. Filion et al. [Filion et al., 2010], for example, applied PCA to identify a set of five chromatin domains with specific functional properties in Drosophila.

A group of methods use sequential state models with a carefully designed state structure and variable dependencies. At the training phase, these models scan part of the genome to learn transition and emission probabilities from data tracks. After training, they can assign labels with probabilities the whole genome sequence. One such model, Segway [Hoffman et al., 2012], uses a Dynamic Bayesian network, a form of a Probabilistic Graphical Model [Bilmes, 2010], to detect common patterns of chromatin modification and TFos occurring on the genome. Based on the patterns it has seen during training, the model can label an arbitrary genome location. These can be visualized in the form of non-overlapping segments that span the whole genome. Segway [Hoffman et al., 2012], was successfully used to identify patterns associated with transcription start sites, gene ends, enhancers, regions bound by transcriptional regulator CTCF and repressed regions.

Other similar methods based on Hidden Markov Models have been proposed. ChromHMM [Ernst & Kellis, 2012] and HMMSeg [Hon et al., 2008] both use HMMs to model commonly occurring Histone Marks. ChromHMM was used in the mouse ENCODE project pipeline whether HMMSeg has been used to assign chromatin signatures to existing and novel promoter and enhancer regions in HeLa cells.

Sequential state models, can handle multiple tracks (e.g., chromatin modification and TFos tracks), but they have a number of limitations. First,

these methods have a "shallow" architecture which is not very efficient with respect to deep architectures, in learning complex representations. Furthermore, they model conditional dependence between adjacent sites (relying on the Markov assumption) but, assume all tracks are conditionally independent given a single latent state at a site. There is empirical evidence on the benefit of extending traditional DBNs with hidden layers in phone recognition [Gunawardana et al., 2005; Andrew & Bilmes, 2012].

We employ stacks of denoising auto encoders (dAE) [Vincent et al., 2008] as models for unsupervised interpretation of multiple genome tracks. We have already described the benefits and properties of hidden representations in deep models (See Section 2.4.4). The goal is to use these representation as summarizing labels of the input.

A dAE is a pair of conjugate vectorial mappings parametrized by the same weight matrix $W_{[q\times p]}$

$$\begin{aligned} h: \quad & \mathbb{R}^p \rightarrow \mathbb{R}^q \\ & \tilde{X} \rightarrow H = \quad \sigma(W\tilde{X}) \\ h': \quad & \mathbb{R}^q \rightarrow \mathbb{R}^p \\ & H \rightarrow Z = \quad \sigma(W^T H) \end{aligned} \tag{5.1}$$

where $\tilde{X}$ is a noisy version of the input X and W are the weights of the layer. The vector H is called the code and is a representation of the input that is forced to keep as much information as possible in order to reconstruct X. The weights of a dAE are initialized at small random values, and trained with gradient descent over a cost function of the type

$$C(W) = |Z - X|_1 + \lambda_1 |W|_1 + \lambda_2 |W|_2 \tag{5.2}$$

where the $|a|_1 = \sum_i |a_i|$ and $|a|_2 = \sum_i a_i^2$. Figure 5.1(A) shows a dAE schematic with a hidden layer of size 5. The heat-maps give a visual representation of 31 functional genomic tracks over 100 randomly chosen 100bp

genome sites in human chromosome 21, their representation, and their reconstruction.

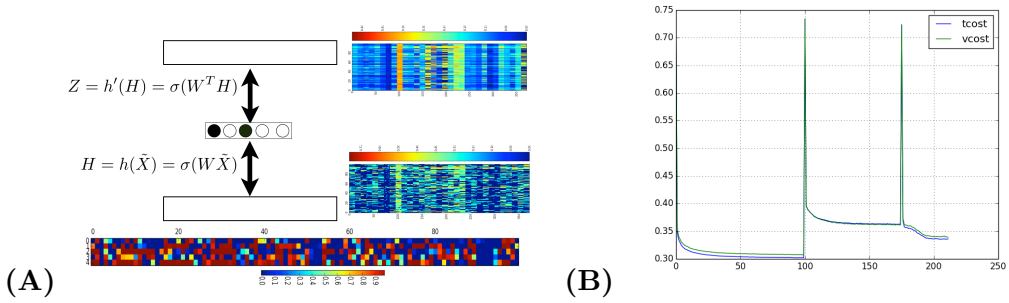
A dAE is similar to the PCA, but because the functions h, h' are non-linear, the dAE can in principle model a richer set of inputs. One can, however, achieve a more complex model by stacking dAEs so that each layer will produce a representation of the output of the layer below – the first layer will produce a representation of the raw input. The representation of the top layer can be interpreted as a combination of labels (one label per component) or as a meta-label of input patterns. As a result the representation allows for labels to a site that are described by any combination of these patterns. A stack of dAEs is trained layer-wise bottom-up. Figure 5.1(B) shows the training and validation cost in Expression (5.2) as a function of training epochs.

We implemented in `dimer` functionality for the design and analysis of deep unsupervised models. Currently `dimer` supports fully connected architectures with Binomial input corruption and sigmoid non-linearities. It also implements various training strategies and data and model parameter management.

5.2 Deep representations of the genome

5.2.1 Experimental setting

Functional genomic tracks have a sequential nature and it is thus important, at any position, to take into consideration the data distribution in a neighborhood of that position. DBN-based models achieve this naturally. We instead define as input to the model a small neighborhood centered in that position. We have experimented with windows ranging from 5 bp to 100 bp. For training, we simply sample the whole input space (the genome or the



(A) **(B)**

Figure 5.1: **(A)** Schematic of a dAE [Vincent et al., 2008]. A dAE is a pair of conjugate functions h, h that, given input X compute a hidden representation (H) and an input reconstruction (Z) respectively. Typically the dimension of H is smaller than that of X. The training criterion favors representations that produce accurate reconstructions ($Z \approx X$). To improve generalization, the model is in fact shown a noisy version of the input (hence the tilde). The pair of heat-maps show 100 input examples and their reconstructions from a 1-layer dAE with a hidden state of size 5. The hidden state is shown in the bottom heat-map. A column of the hidden state heat-map recapitulates a row of the input heat-map. **(B)** Train and validation cost as a function of training epochs over the same dataset. The y-axis shows the normalized cost in Expression (5.2) $C(W)/|W|_1$. The spikes show the cost function of each layer before the first epoch when the layer has random weights. In this example, the input is of size 310 and the hidden representations of size 201, 116, 31 respectively.

chromosome) and feed the the samples in a random order to the model. The more signal configurations we capture with the training data, the better.

During the identification phase we slide a fixed size window spanning all the tracks along the genome. Each input sample centered at a certain position will be recapitulated by the representation of the top layer of the dAE stack. Each component of this representation can be thought as a (discrete) label and the whole representation as a *meta-label* for that position (See Figure 5.2). This interpretation has the advantage of being dense so that a meta-label of size l with k-level discretization can represent up to k^l states.

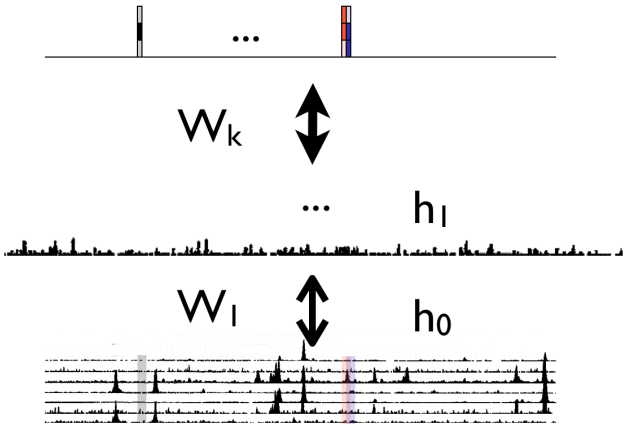


Figure 5.2: Higher levels of a stack of dAE can learn to represent higher level features by using hidden representations of layers below as input. Imposing decreasing output sizes for each label, forces hidden representations to be informative. The output of the last layer can be interpreted as a meta-label for the corresponding input. We use fixed-length windows over all input tracks as input to the dAE. The resulting meta-label, is thus a small dimensional representation of the input.

Fully connected dAEs lack translation equivariance – translated inputs do not correspond to translated outputs – posing a challenge in the interpretation of meta-labels (See Figure 5.3). We considered substituting fully connected layers of a dAE with convolutional ones, which are translation

equivariant. However, this approach would require a receptive field to span, in principle, the whole input range making it hard to cope with missing data. A sliding window dAE can instead easily skip over missing data.

Figure 5.3: Labeling of a genome region in chromosome 21 using a 4-layer dAE with meta-label size of 8 trained over genes (shown in blue). We expect perfect fit of meta-labels since the model has enough capacity to reproduce the input without loss of information. However, as the sliding window approaches the gene, the output signal fluctuates.

5.2.2 Data and Model

We consider 31 assays over 9 Histone Modifications, 2 DNaseI Hypersensitive Sites, and 20 TFs (See Table 5.1). While more tracks are now available, we chose this dataset because it had been previously used to generate segmentations [Hoffman et al., 2012].

We train each of the layers of the model, starting from the bottom. We apply 20% noise (i.e., we set the value of an input example to zero with probability 0.2) to each training example before showing it to the model. We use an early stopping strategy with memory on the validation cost. We perform a grid search over the parameter space $W \times B \times L$ (window size, binning size, model depth) and choose parameters with the smallest reconstruction error over the training set .

The translation equivariance problem (Figure 5.3) makes it difficult to interpret the raw meta-labels. To deal with this problem, we trained a dAE

Track type	Features targeted
TFs	cFos, cJun, cMyc (Iyer), cMyc (Snyder), CTCF (Bernstein), CTCF (Iyer), GABP, GATA1, JunD, Max, NFE2, NRSF, Pol2_8WG16 (Myers), Pol2_8WG16 (Snyder), Pol2_CTD4H8, Rad21, SRF, XRCC4, ZNF263
Open chromatin	DNaseI (Crawford), DNaseI (Stam), FAIRE
HM	H3K27ac, H3K27me3, H3K36me3, H3K4me1, H3K4me2, H3K4me3, H3K9ac, H3K9me1, H4K20me1

Table 5.1: Tracks for the Segway dataset. Names in parenthesis indicate the laboratories that produced the data.

and fed the output to Segway. Then, we compared the results with the output of a Segway run on the raw data. Figure 5.4 shows one example of a dAE + Segway versus a simple Segway runs over the same dataset. The dAE + Segway system scans input examples of length 100bp binned in bins of size 10bp and has a top layer output size of 31 – these numbers correspond to a 10-fold dimensionality reduction by the dAE. Segway is fed with both the raw 31 track input and the dAE output with all parameters unchanged at their default value.

5.3 Transcription factor composition analysis of timing replication domains

5.3.1 Introduction

The replication-timing program maintains a temporal order of DNA replication in Eukaryotes. The whole genome can be partitioned into regions of uniform replication timing (early or late) bounded by timing transition

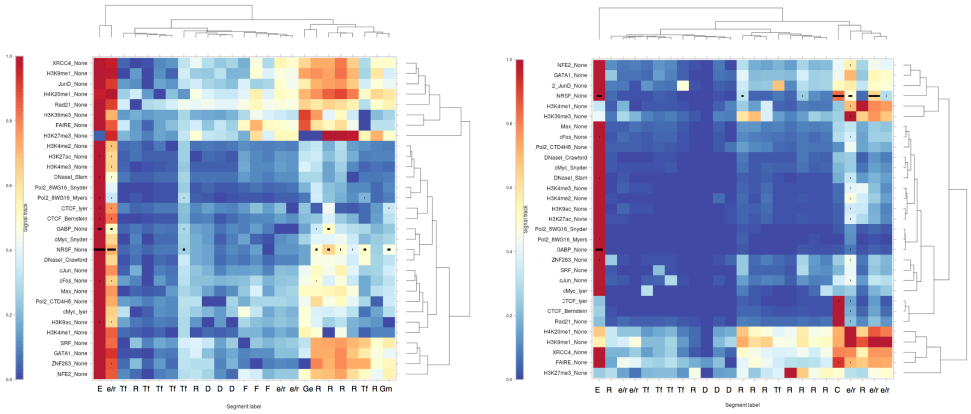


Figure 5.4: We considered 31 ENCODE signal tracks that measured binding of 15 different TFs, 9 different histone modifications, and regions of open chromatin on the K562 cell line. We trained a 3-layer dAE in unsupervised mode on a small (1%) genomic region of chromosome 21 and then labeled the whole chromosome allowing for 31 simultaneous states per 200 bp window binned in 10bp bins. The hidden representation of the top layer resulted in a 10-fold reduction of the dimension of input. Heat-maps above show label enrichment for input tracks for Segway alone and dAE+Segway respectively.

regions (TTR). During development changes in replication timing occur in 400-800Kb units called replication domains (RD) [Hiratani & Gilbert, 2010]. So, there are multiple uniformly replicating regions separated by TTRs within a domain, but the boundary between two RDs is not necessarily marked by a TTR. RDs are an important topological unit for the genome as they are preserved across cell types and species and seem to confine long-range effects of genome rearrangement [Pope et al., n.d.].

Recently, studies of the 3D conformation of the chromatin have discovered other domains induced by chromatin organization (1) topologically associating domains (TADs), which are genomic regions that align with H3K27 and H3K9 blocks and (2) lamina-associated domains (LADs), and coordinately regulated gene clusters.

It is thus, of interest to find a uniform domain based chromatin structure that explains the relationship between TADs, LADs and RDs. Recent studies have attempted to find such a structure using data from the Human and Mouse ENCODE projects. One aspect of this attempt lead by Pope et al. [Pope et al., n.d.] investigated the extent to which transcription factor binding could be used to distinguish hypothetical regulatory classes of RDs. By combining available ENCODE data for seven factors (Ctcf, Hcfc1, MafK, P300, PolII, Zc3h11a, and Znf384), we assigned data driven labels for each mESC TAD using a stack of six denoising autoencoders [Vincent et al., 2008], which is a powerful data reduction technique for unsupervised machine learning [Y. Bengio, 2013]. We trained the stack in an unsupervised fashion on 200 random samples for each of the six layers and obtained two labels or groups of TADs, termed A0 and B0.

5.3.2 Data and Model

Raw data from genome-wide ChIP-Seq in vivo detection of TF binding were processed in 200bp bins over a 1Mb window around the center of each mESC TAD. Each sample can be thought as a 2-dimensional matrix with rows for seven TF activity profiles and each row represented as a real vector with values for each bin in the 1 Mb window (number of bins = 5000). The value X_{ijk} represented the activity (i.e. peak enrichment) of TF j, k bins from the center of TAD i. The model consists of a stack of 7 sigmoid denoising autoencoders, which define parameterized feature extraction (encoder) and reconstruction functions (decoder). The encoder projects the data into a smaller dimension feature space and the decoder reconstructs the input from the feature space. The parameters of the functions are optimized to give the smallest reconstruction error over all the training data samples. While similar in principle to PCA (they both can be used as dimensionality reduction techniques), the denoising auto-encoder learns a nonlinear mapping between the input and its representation. Furthermore, constraints on the input and feature dimension sizes and the addition of noise to the input force it to learn important structure in the input. The stack reduces input dimensions gradually from (7 x 5000), (7 x 1000), (7 x 200), (7 x 60), (7 x 30), (7 x 20), to (7 x 10). Each autoencoder injects an additive binomial noise with a 20% corruption rate. We initialized weights at small random values with mean zero and used minibatch stochastic gradient descent [Rumelhart et al., 1986] to minimize the mean squared input reconstruction error. We trained on 200 random samples for 500 epochs each layer then used the model to transform all other samples. The model output was evaluated using gold standard labels based on both the means and standard deviations of mESC replication timing within each TAD. Early TADs had mean replication timing > 0 and standard deviation < 0.25, Late TADs had mean replication timing < 0 and standard deviation < 0.25, and all others TADs were considered

"TTR".

5.3.3 Results

Interestingly, 94% of all TADs were already labeled as A0 or B0 in the fourth layer and 100% were labeled in the fifth and sixth layers, indicative of clearly recognizable differences in the transcription factor composition of these two groups of TADs, as well as clear similarities within each group of TADs. With an overall error rate of 16%, A0 TADs corresponded to RDs within either TTRs or late regions, while B0 TADs corresponded exclusively to early RDs (Table 5.3.3). The relatively high enrichment of Hcfc1, MafK, and PolII within early versus late RDs may account for the two labels (Figure 5.5). These results demonstrate that combinations of transcription factors can accurately predict the compartmentalization of RDs and provide additional evidence that TTRs and late RDs have indistinguishable chromatin composition and are partitioned from early replicating TADs at TTR-present RD boundaries [Pope et al., n.d.].

	Score		
TF model prediction	*Early*	*TTR*	*Late*
A0 (TTR/Late)	7%	39%	17%
B0 (Early)	28%	9%	0%

Table 5.2: True versus predicted classification rates are shown comparing the labels of an unsupervised model trained on the profiles of seven transcription factors (Ctcf, Hcfc1, MafK, P300, PolII, Zc3h11a, and Znf384) versus actual replication timing for all mESC TADs. TADs considered either TTR or Late by replication timing predominantly composed label A0, while Early TADs predominantly composed label B0.

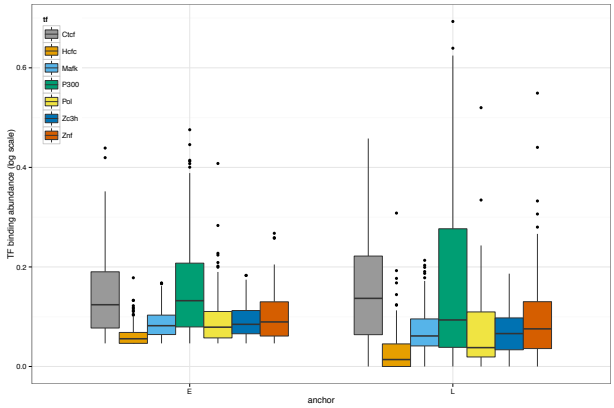


Figure 5.5: TF composition of mESC TADs grouped by a 7-layer dAE. The plot shows the distribution of the sum of the normalized TF profile signal, $\sum_k X_{ijk}$ grouped by the model assigned label.

5.4 Conclusion

In this chapter we proposed a framework for the use of deep unsupervised models for modeling functional genomics data. First, we produced segmentations of the genome based on Histone Modifications, DnaseI Hypersensitive Sites, and TFos. The resulting labels align with functional features defined by the input. Next, we investigated the extent to which transcription factor binding could be used to distinguish hypothetical regulatory classes of RDs. By combining available ENCODE data for seven factors (Ctcf, Hcfc1, MafK, P300, PolII, Zc3h11a, and Znf384), we obtained a grouping of TADs induced by their TF composition that aligned with the structure of RDs.

Both applications show that deep dAEs are a viable alternative as unsupervised models of functional genomic data. Hidden representations produced by these models are informative and can be used for dimensionality reduction.

One difficulty in the use of dAEs is the adaptation of these models to sequential data – specially when dependencies between sites of the same track

are important. To address this problem, combinations of deep models with sequential models are a possible avenue [Andrew & Bilmes, 2012]. Another possible pitfall with dAEs is the lack of translation equivariance. Due to this phenomenon, hidden layers are hard to interpret by the human eye and further processing is needed to extract information. However, this is a problem of the particular architecture (Fully Connected Feed Forward) used here and has been addressed in the past by use of convolutional layers [LeCun & Bengio, 1995].

References

Abramowitz, L. K., & Bartolomei, M. S. (2012, Apr). Genomic imprinting: recognition and marking of imprinted loci. *Curr Opin Genet Dev*, *22*(2), 72-8.

Allender, E. (1996). Circuit complexity before the dawn of the new millennium. In *16th annual conference on foundations of software technology and theoretical computer science* (pp. 1 – 18).

Andersson, R., Gebhard, C., Miguel-Escalada, I., Hoof, I., Bornholdt, J., Boyd, M., et al. (2014, Mar). An atlas of active enhancers across human cell types and tissues. *Nature*, *507*(7493), 455-61.

Andrew, G., & Bilmes, J. (2012). Sequential deep belief networks. In *International conference on acoustics, speech and signal processing.*

Bengio, Y. (2009, January). Learning Deep Architectures for AI. *Found. Trends Mach. Learn.*, *2*(1), 1–127.

Bengio, Y., Lamblin, P., D., P., & Larochelle, H. (2007). Greedy layer-wise training of deep networks." advances in neural information processing systems 19 (2007): 153. *NIPS*.

Bernstein, B. E., Stamatoyannopoulos, J. A., Costello, J. F., Ren, B., Milosavljevic, A., Meissner, A., et al. (2010, Oct). The NIH Roadmap Epigenomics Mapping Consortium. *Nat Biotechnol*, *28*(10), 1045-8.

Bilmes, J. (2010, November). Dynamic graphical models. *IEEE Signal Processing Magazine*, *27*(6), 29 – 42.

Blanchette, M. (2007). Computation and analysis of genomic multi-sequence alignments. *Annu Rev Genomics Hum Genet*, *8*, 193-213.

Bulger, M., & Groudine, M. (2010, Mar). Enhancers: the abundance and function of regulatory sequences beyond promoters. *Dev Biol*, *339*(2), 250-7.

Cheng, C., Alexander, R., Min, R., Leng, J., Yip, K. Y., Rozowsky, J., et al. (2012, Sep). Understanding transcriptional regulation by integrative analysis of transcription factor binding data. *Genome Res*, *22*(9), 1658-67.

Cheng, Y., Wu, W., Kumar, S. A., Yu, D., Deng, W., Tripic, T., et al. (2009, Dec). Erythroid GATA1 function revealed by genome-wide analysis of transcription factor occupancy, histone modifications, and mRNA expression. *Genome Res*, *19*(12), 2172-84.

Collins, F., & Galas, D. (1993, Oct). A new five-year plan for the U.S. Human Genome Project. *Science*, *262*(5130), 43-6.

Davidson, E. (2006). *The regulatory genome. gene regulatory networks in developement and evolution.* Academic Press.

Davidson, E. H., & Erwin, D. H. (2006, Feb). Gene regulatory networks and the evolution of animal body plans. *Science*, *311*(5762), 796-800.

Derman, E., Krauter, K., Walling, L., Weinberger, C., Ray, M., & Darnell, J. E., Jr. (1981, Mar). Transcriptional control in the production of liver-specific mRNAs. *Cell*, *23*(3), 731-9.

Dixon, J. R., Selvaraj, S., Yue, F., Kim, A., Li, Y., Shen, Y., et al. (2012, May). Topological domains in mammalian genomes identified by analysis of chromatin interactions. *Nature*, *485*(7398), 376-80.

Dulbecco, R. (1986, Mar). A turning point in cancer research: sequencing the human genome. *Science*, *231*(4742), 1055-6.

Enard, W., Przeworski, M., Fisher, S. E., Lai, C. S. L., Wiebe, V., Kitano, T., et al. (2002, Aug). Molecular evolution of FOXP2, a gene involved in speech and language. *Nature*, *418*(6900), 869-72.

ENCODE Project Consortium. (2011, Apr). A user's guide to the encyclopedia of DNA elements (ENCODE). *PLoS Biol*, *9*(4), e1001046.

ENCODE Project Consortium, Bernstein, B. E., Birney, E., Dunham, I., Green, E. D., Gunter, C., et al. (2012, Sep). An integrated encyclopedia of DNA elements in the human genome. *Nature*, *489*(7414), 57-74.

Erhan, D., Bengio, Y., Courville, A., & Vincent, P. (2009). *Visualizing higher-layer features of a deep network* (Tech. Rep.). Departement d'Informatique et Recherche Operationnelle.

Ernst, J., & Kellis, M. (2012, Mar). ChromHMM: automating chromatin-state discovery and characterization. *Nat Methods*, *9*(3), 215-6.

Favorov, A., Mularoni, L., Cope, L. M., Medvedeva, Y., Mironov, A. A., Makeev, V. J., et al. (2012, May). Exploring massive, genome scale datasets with the GenometriCorr package. *PLoS Comput Biol*, *8*(5), e1002529.

Filion, G. J., Bemmel, J. G. van, Braunschweig, U., Talhout, W., Kind, J., Ward, L. D., et al. (2010, Oct). Systematic protein location mapping reveals five principal chromatin types in drosophila cells. *Cell*, *143*(2), 212-24.

Gerstein, M. B., Lu, Z. J., Van Nostrand, E. L., Cheng, C., Arshinoff, B. I., Liu, T., et al. (2010, Dec). Integrative analysis of the Caenorhabditis elegans genome by the modENCODE project. *Science*, *330*(6012), 1775-87.

Glorot, X., & Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In *13th international conference on artificial intelligence and statistics (aistats).*

Gunawardana, A., Mahajan, M., Acero, A., & Platt, J. (2005). Hidden conditional random fields for phone classification. In citeseer (Ed.), *Interspeech* (Vol. 2, p. 1).

Gurobi Optimization, I. (2013). *Gurobi optimizer reference manual.* Available from `http://www.gurobi.com`

Han, J., & Kamber, M. (2001). *Data mining: Concepts and techniques.* MORGAN KAUFMANN PUBLISHERS.

Hardison, R. C., & Taylor, J. (2012, Jul). Genomic approaches towards finding cis-regulatory modules in animals. *Nat Rev Genet*, *13*(7), 469-83.

Hastie, T., Tibishrani, R., & Friedman, J. (2009). *Elements of statistical learning: Data mining, inference, and prediction.* Springer.

Heintzman, N. D., Stuart, R. K., Hon, G., Fu, Y., Ching, C. W., Hawkins, R. D., et al. (2007, Mar). Distinct and predictive chromatin signatures of transcriptional promoters and enhancers in the human genome. *Nat Genet*, *39*(3), 311-8.

Hinton, G. E., Osindero, S., & Teh, Y.-W. (2006). A fast learning algorithm for deep belief nets. *Neural Comput.*, *18*(7), 1527–1554.

Hiratani, I., & Gilbert, D. M. (2010). Autosomal lyonization of replication domains during early mammalian development. *Adv Exp Med Biol*, *695*, 41-58.

Hoffman, M. M., Buske, O. J., Wang, J., Weng, Z., Bilmes, J. A., & Noble, W. S. (2012, May). Unsupervised pattern discovery in human chromatin structure through genomic segmentation. *Nat Methods*, *9*(5), 473-6.

Hon, G., Ren, B., & Wang, W. (2008, Oct). Chromasig: a probabilistic approach to finding common chromatin signatures in the human genome. *PLoS Comput Biol*, *4*(10), e1000201.

Hughes, A. L., & Yeager, M. (1998). Natural selection at major histocompatibility complex loci of vertebrates. *Annu Rev Genet*, *32*, 415-35.

James, G., Witten, D., Hastie, T., & Tibishrani, R. (2013). *An introduction to statistical learning*. Springer.

Ji, H., Jiang, H., Ma, W., Johnson, D. S., Myers, R. M., & Wong, W. H. (2008, Nov). An integrated software system for analyzing ChIP-chip and ChIP-seq data. *Nat Biotechnol*, *26*(11), 1293-300.

Johnson, D. S., Mortazavi, A., Myers, R. M., & Wold, B. (2007, Jun). Genome-wide mapping of in vivo protein-DNA interactions. *Science*, *316*(5830), 1497-502.

Kharchenko, P. V., Tolstorukov, M. Y., & Park, P. J. (2008, Dec). Design and analysis of ChIP-seq experiments for DNA-binding proteins. *Nat Biotechnol*, *26*(12), 1351-9.

Kim, T.-K., Hemberg, M., Gray, J. M., Costa, A. M., Bear, D. M., Wu, J., et al. (2010, May). Widespread transcription at neuronal activity-regulated enhancers. *Nature*, *465*(7295), 182-7.

King, M. C., & Wilson, A. C. (1975, Apr). Evolution at two levels in humans and chimpanzees. *Science*, *188*(4184), 107-16.

Klug, W. S., Cummings, M. R., & Spencer, C. A. (2007). *Essentials of genetics*. Pearson prentice hall.

Kouzarides, T. (2007). Chromatin modifications and their function. *Cell*, *128*(4), 693 - 705.

Ladunga, I. (2010). *Computational biology of transcription factor binding*. Springer.

Lander, E. S., Linton, L. M., Birren, B., Nusbaum, C., Zody, M. C., Baldwin, J., et al. (2001, Feb). Initial sequencing and analysis of the human genome. *Nature*, *409*(6822), 860-921.

Le, Q., Monga, R., Devin, M., Corrado, G., Chen, K., Ranzato, M., et al. (2011). Building high-level features using large scale unsupervised learning. *arXiv preprint arXiv:1112.6209*.

LeCun, Y., & Bengio, Y. (1995). Convolutional networks for images, speech, and time-series. In M. A. Arbib (Ed.), *The handbook of brain theory and neural networks.* MIT Press.

Lee, H., Grosse, R., Ranganath, R., & Ng, A. (2009). Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *Proceedings of the 26th annual international conference on machine learning* (pp. 609–616).

Levine, M., & Tjian, R. (2003, Jul). Transcription regulation and animal diversity. *Nature*, *424*(6945), 147-51.

Lieberman-Aiden, E., Berkum, N. L. van, Williams, L., Imakaev, M., Ragoczy, T., Telling, A., et al. (2009, Oct). Comprehensive mapping

of long-range interactions reveals folding principles of the human genome. *Science*, *326*(5950), 289-93.

Maston, G. A., Evans, S. K., & Green, M. R. (2006). Transcriptional regulatory elements in the human genome. *Annu Rev Genomics Hum Genet*, *7*, 29-59.

McCulloch, & Pitts. (1943). A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, *7*, 115 - 133.

Meader, S., Ponting, C. P., & Lunter, G. (2010, Oct). Massive turnover of functional sequence in human and other mammalian genomes. *Genome Res*, *20*(10), 1335-43.

Metzker, M. L. (2010, Jan). Sequencing technologies - the next generation. *Nat Rev Genet*, *11*(1), 31-46.

Minsky, M. L., & Papert, S. (1969). *Perceptrons*. MIT press.

modENCODE Consortium, Roy, S., Ernst, J., Kharchenko, P. V., Kheradpour, P., Negre, N., et al. (2010, Dec). Identification of functional elements and regulatory circuits by Drosophila modENCODE. *Science*, *330*(6012), 1787-97.

Montavon, G., Orr, G. B., & Müller, K.-R. (Eds.). (2012). *Neural networks: Tricks of the trade*. Springer.

Morey, C., & Avner, P. (2011, Jul). The demoiselle of x-inactivation: 50 years old and as trendy and mesmerising as ever. *PLoS Genet*, *7*(7), e1002212.

Mouse ENCODE Consortium, Stamatoyannopoulos, J. A., Snyder, M., Hardison, R., Ren, B., Gingeras, T., et al. (2012, Aug). An encyclopedia of mouse DNA elements (Mouse ENCODE). *Genome Biol*, *13*(8), 418.

Myers, E. W., Sutton, G. G., Smith, H. O., Adams, M. D., & Venter, J. C. (2002, Apr). On the sequencing and assembly of the human genome. *Proc Natl Acad Sci U S A*, *99*(7), 4145-6.

Neumann, J. von. (1958). *The computer and the brain.* Yale University Press.

Phillips, J. E., & Corces, V. G. (2009, Jun). CTCF: master weaver of the genome. *Cell*, *137*(7), 1194-211.

Pique-Regi, R., Degner, J. F., Pai, A. A., Gaffney, D. J., Gilad, Y., & Pritchard, J. K. (2011, Mar). Accurate inference of transcription factor binding from dna sequence and chromatin accessibility data. *Genome Res*, *21*(3), 447-55.

Plocik, A. M., & Graveley, B. R. (2013, Feb). New insights from existing sequence data: Generating breakthroughs without a pipette. *Mol Cell*, *49*(4), 605-17.

Pope, B. D., Ruba, T., Dileep, V., Yue, F., Wu, W., Denas, O., et al. (n.d.). *Topologically-associating domains are stable structural units of replication timing regulation.* (Submited as a companion paper in the Mouse ENCODE project)

Ren, B., Robert, F., Wyrick, J. J., Aparicio, O., Jennings, E. G., Simon, I., et al. (2000, Dec). Genome-wide location and function of DNA binding proteins. *Science*, *290*(5500), 2306-9.

Roop, D. R., Nordstrom, J. L., Tsai, S. Y., Tsai, M. J., & O'Malley, B. W. (1978, Oct). Transcription of structural and intervening sequences in the ovalbumin gene and identification of potential ovalbumin mRNA precursors. *Cell*, *15*(2), 671-85.

Rosenblatt, F. (1958, Nov). The perceptron: a probabilistic model for information storage and organization in the brain. *Psychol Rev*, *65*(6), 386-408.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, *323*, 533 – 536.

Sanger, F., Coulson, A. R., Hong, G. F., Hill, D. F., & Petersen, G. B. (1982, Dec). Nucleotide sequence of bacteriophage lambda DNA. *J Mol Biol*, *162*(4), 729-73.

Swanson, W. J., Yang, Z., Wolfner, M. F., & Aquadro, C. F. (2001, Feb). Positive darwinian selection drives the evolution of several female reproductive proteins in mammals. *Proc Natl Acad Sci U S A*, *98*(5), 2509-14.

Taylor, J., Tyekucheva, S., King, D. C., Hardison, R. C., Miller, W., & Chiaromonte, F. (2006, Dec). ESPERR: learning strong and weak signals in genomic sequence alignments to identify functional elements. *Genome Res*, *16*(12), 1596-604.

Thurman, R., Rynes, E., Humbert, R., Vierstra, J., Maurano, M. T., Haugen, E., et al. (2012). The accessible chromatin landscape of the human genome. *Nature*, *489*(7414), 75-82.

Turner, B. M. (2005, Feb). Reading signals on the nucleosome with a new nomenclature for modified histones. *Nat Struct Mol Biol*, *12*(2), 110-2.

Venter, J. C., Adams, M. D., Myers, E. W., Li, P. W., Mural, R. J., Sutton, G. G., et al. (2001, Feb). The sequence of the human genome. *Science*, *291*(5507), 1304-51.

Vincent, P., Larochelle, H., Bengio, Y., & Manzagol, P.-A. (2008). Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on machine learning* (pp. 1096–

1103). New York, NY, USA: ACM. Available from http://doi.acm.org/10.1145/1390156.1390294

Wadman, I. A., Osada, H., Grütz, G. G., Agulnick, A. D., Westphal, H., Forster, A., et al. (1997, Jun). The LIM-only protein Lmo2 is a bridging molecule assembling an erythroid, DNA-binding complex which includes the TAL1, E47, GATA-1 and Ldb1/NLI proteins. *EMBO J*, *16*(11), 3145-57.

Waterston, R. H., Lander, E. S., & Sulston, J. E. (2002, Mar). On the sequencing of the human genome. *Proc Natl Acad Sci U S A*, *99*(6), 3712-6.

Weber, J. L., & Myers, E. W. (1997, May). Human whole-genome shotgun sequencing. *Genome Res*, *7*(5), 401-9.

Weiss, M. J., Keller, G., & Orkin, S. H. (1994, May). Novel insights into erythroid development revealed through in vitro differentiation of GATA-1 embryonic stem cells. *Genes Dev*, *8*(10), 1184-97.

Weiss, M. J., Yu, C., & Orkin, S. H. (1997, Mar). Erythroid-cell-specific properties of transcription factor GATA-1 revealed by phenotypic rescue of a gene-targeted cell line. *Mol Cell Biol*, *17*(3), 1642-51.

Wilson, M. D., & Odom, D. T. (2009, Dec). Evolution of transcriptional control in mammals. *Curr Opin Genet Dev*, *19*(6), 579-85.

Wold, B., & Myers, R. M. (2008, Jan). Sequence census methods for functional genomics. *Nat Methods*, *5*(1), 19-21.

Wu, W., Cheng, Y., Keller, C. A., Ernst, J., Kumar, S. A., Mishra, T., et al. (2011, Oct). Dynamics of the epigenetic landscape during erythroid differentiation after GATA1 restoration. *Genome Res*, *21*(10), 1659-71.

Yamamoto, M., Ko, L. J., Leonard, M. W., Beug, H., Orkin, S. H., & Engel, J. D. (1990, Oct). Activity and tissue-specific expression of the transcription factor NF-E1 multigene family. *Genes Dev*, *4*(10), 1650-62.

Y. Bengio, P. V., A. Courville. (2013). Representation learning: A review and new perspectives. In *30th international conference on machine learning.*

Yu, H., Zhu, S., Zhou, B., Xue, H., & Han, J.-D. J. (2008). Inferring causal relationships among different histone modifications and gene expression. *Genome Res*.

Zhang, Y., Liu, T., Meyer, C. A., Eeckhoute, J., Johnson, D. S., Bernstein, B. E., et al. (2008). Model-based analysis of ChIP-Seq (MACS). *Genome Biol*, *9*(9), R137.