An Overview of Probabilistic Latent Variable Models with an Application to the Deep Unsupervised Learning of Chromatin States

Dissertation

Presented in Partial Fulfillment of the Requirements for the Degree Doctor of Philosophy in the Graduate School of The Ohio State University

By

Tarek Farouni, B.A., M.A., M.S.

Graduate Program in Psychology

The Ohio State University

2017

Dissertation Committee:

Robert Cudeck, Advisor

Paul DeBoeck

Zhong-Lin Lu

Ewy Mathé

Abstract

The following dissertation consists of two parts. The first part presents an overview of latent variable models from a probabilistic perspective. The main goal of the overview is to give a birds-eye view of the topographic structure of the space of latent variable models in light of recent developments in statistics and machine learning that show how seemingly unrelated models and methods are in fact intimately related to each other.

In the second part of the dissertation, we apply a Deep Latent Gaussian Model (DLGM) to high-dimensional, high-throughput functional epigenomics datasets with the goal of learning a latent representation of functional regions of the genome, both across DNA sequence and across cell-types. In the latter half of the dissertation, we first demonstrate that the trained generative model is able to learn a compressed two-dimensional latent representation of the data. We then show how the learned latent space is able to capture the most salient patterns of dependencies in the observations such that synthetic samples simulated from the latent manifold are able to reconstruct the same patterns of dependencies we observe in data samples. Lastly, we provide a biological interpretation of the learned latent manifold in terms of a continuous histone code and explain both the relevance and significance of the proposed generative approach to the problem of identifying functional regulatory regions from epigenomic marks.

This is dedicated to my wife Anna, my son Lev, and my immediate family.

Acknowledgments

I am extremely grateful to my advisor, Bob Cudeck, both for the freedom that allowed my mind to pursue interdisciplinary scientific questions and for the personal and academic support that emboldened me to attempt novel methodological approaches to answering them. I would like to thank Dr. Ewy Mathé for all that she has taught me in the last year and especially for her diligent review and commentary of the dissertation draft. I would also like to thank the rest of my dissertation committee members, Dr. Paul DeBoeck and Dr. Zhong-Lin Lu, for their advice, criticism, and support throughout many years. I am fortunate to have learned from all four committee members. This dissertation would not have been possible without their support. The ideas they introduced me to and their influence can be seen throughout this dissertation.

I would like to thank Slava Nikitin, Brenden Bishop, and Arkady Konovalov for their friendship, outstanding intellect, and stimulating discussions. Last by not least, I would like to thank Dr. Luca Pinello for all the help and feedback he provided me as I was writing the manuscript. It would not have been the same without all of you. Thank you.

Vita

July 26, 1977	Born - Amman, Jordan
2012	B.A. Psychology, The Pennsylvania State University
2014	M.A. Quantitative Psychology, The Ohio State University
2015	M.S. Statistics, The Ohio State University
2012-2013	University Fellow, The Ohio State University.
2013-present	Graduate Teaching Associate, Department of Psychology, The Ohio State University.

Publications

Research Publications

Baskin, E., Farouni, R., and Mathè, E. A. (2016). ALTRE: workflow for defining ALTered Regulatory Elements using chromatin accessibility data. *Bioinformatics*, btw688.

Fields of Study

Major Field: Psychology

Table of Contents

List of Tables

Table **Page**

List of Figures

Chapter 1: Introduction and Preliminaries

1.1 Summary of Proposed Contributions

The main contributions of the following dissertation are as follows. The first contribution, and the focus of Chapter 2, comes in the form a conceptual contemporary overview of latent variable models formulated within a generative probabilistic modeling framework. The chapter does not attempt to review recent literature, but rather aims to synthesize ideas and bring together in one place recent model developments in psychometrics, statistics, and machine learning. The main goal of the overview is to convey to the reader a birds-eye view of the topographic structure of the space of latent variable models and show how they relate to each other. Although model formulation and model inference are inextricable in practice, they are treated in this part of the dissertation as separate. By doing so, we can in our minds disentangle the two, elevating model building center stage while relegating model inference and its minutiae backstage, where the universal inference machine can quietly chug along.

The second contribution, and the focus of Chapter 3, investigates the application of a particular class of latent variable models, *Deep Latent Gaussian Models*, to high dimensional, high-throughput functional epigenomics data sets with the goal of learning a latent representation of chromatin states both across the genome and across cell types. Deep Latent Gaussian Models (Rezende, Mohamed, & Wierstra, 2014; D. P. Kingma & Welling,

2013) bring together the strengths of two streams of statistical learning: deep learning and probabilistic graphical modeling. Whereas deep learning allows us to build scalable models that can capture the most complex nonlinear dependencies in the data, probabilistic graphical modeling provides a framework in which we can model not just the variables we observe, but also latent variables that represent hidden patterns which can explain the observed data. By combining a generative probabilistic model with a deep neural network, we obtain a deep generative model (Salakhutdinov, 2015) that can capture a *hierarchy* of distributed dependencies between latent variables and learn an efficient lower dimensional representation of the data. Furthermore, generative models enable us to generate synthetic samples from the learned lower dimensional manifold, samples that would look indistinguishable from the observed data when the model does indeed learn an efficient latent representation. This latent representation equips generative models with an interpretive power that is absent from many black-box predictive machine learning algorithms. Although predictive models have proven to be valuable in many scientific applications, it can be argued that for the purposes of basic scientific research, their value is contingent upon the supportive role they play in building explanatory models of natural phenomena. Indeed, the generative approach to modeling keeps true to the principle of *verum factum*; namely, to understand an entity properly, one should be able to create it, or in the words of Richard Feynman "What I cannot create, I do not understand".

1.2 Definitions

Although the section can be skipped without loss of continuity, the definitions given here serve as the foundation upon which many concepts encountered later in the manuscript are built upon. The main goal of the next few pages is to start from the most basic notions and arrive at an intuitive mathematical definition of a *statistical model*. Although conceptual motivation and some degree of mathematical rigor is attempted at each step of constructing the definition, the final resulting definition does not say much about the nature of the relationship between a statistical model and its target phenomenon observed in the physical universe. Therefore, it is important to point out here that the modeling perspective adopted in this dissertation starts from the premise that a statistical model is a *phenomenological model* of reality describing relationships among observed variables in a mathematical formalism that is not derived from first principles (Hilborn & Mangel, 1997). A model is considered phenomenological in the Kantian sense (Kant & Guyer, 1998) in which the word *phenomenon* denotes an empirical object that is known through the senses and whose mental representation is constructed by experience. The *phenomenon* is contrasted with the *noumenon*, the "thing-in-itself" that is not directly knowable by the senses but which can be known by the mind in the form of a theoretical construct. For example, whereas the electromagnetic force is a theoretical construct arrived at from first principles and provides a causal explanation to empirical observations (e.g. effects of magnetic and electric fields), the empirical observations themselves constitute experienced phenomenon whose correlative nature can provide nothing more than evidence for noumenal knowledge.

1.2.1 Defining a Probability Model

Rigorous mathematical definitions of a probability space and a statistical model can be found in several excellent probability textbooks (Durrett, 2010; Athreya & Lahiri, 2006) and so we make no attempt here to reproduce them. Instead, we try to give rather brief, and consequently, less rigorous definitions that highlight the conceptual motivation behind their construction and theoretical necessity. In these few pages, the hope is to convey to the reader a view of a statistical model as a regularizer, a tamer of infinite possibilities which allows us to solve ill-posed problems.

Outcomes We begin with the **sample space** $\mathscr{X}$ (e.g. $\mathscr{X} = \{0, 1\}$), a nonempty **set** whose elements are called **outcomes** $x \in \mathscr{X}$. Most often, we are interested in defining probabilities not just for single outcomes, but also for all possible collections of outcomes (i.e. events). How can we enumerate all these possible events?

Events We generate events by equipping the sample space with structure. A set along with a collection of operations (e.g. $+, \times$) is called an **algebraic structure**. A **powerset** $(\mathscr{P}(\mathscr{X}), \cap, \cup, \neg, \emptyset, \mathscr{X})$ is a type of algebraic structure characterized by having only two binary operations $(\cap, \cup)$. So from $\mathscr{X}$ we can generate the powerset $\mathscr{P}(\mathscr{X})$ (e.g. $\mathscr{P}(\mathscr{X}) = \{\emptyset, \{0\}, \{1\}, \{0, 1\}\}$), the set of all **subsets** of $\mathscr{X}$. Subsets of $\mathscr{X}$ are termed **events**. But this only works when the sample space is countably finite. To equip a set with a structure, it should not be too large. When the set is the real line (i.e. $\mathscr{X} = \mathbb{R}$) for example, the powerset of $\mathbb{R}$, $\mathscr{P}(\mathbb{R})$, is simply too big to be manageable. So we need to restrict the set. But how big is the set (i.e. cardinality) in the first place and by how much should we restrict it? For a countably infinite set such as the natural numbers $\mathbb{N}$, for example, the cardinality

of $\mathbb{N}$ is denoted by beth null $\beth_0$. Since we are concerned with events, we need to think about the set of all subsets. The cardinality of the set of all subsets of the natural numbers $P(\mathbb{N})$ is denoted by $\beth_1$, which is also equal to the cardinality of the **continuum** $\mathbb{R}$. This brings us to the case of continuous variables where the sample space is the uncountably infinite set $\mathbb{R}$. Now, the number of subsets of $\mathbb{R}$ (i.e. number of events) is equal to the cardinality of $P(P(\mathbb{N}))$, the power set of the set of real numbers! The cardinality of this huge space is denoted by $\beth_2$.

The σ-algebra To stay within the limits of the continuum, we need to restrict ourselves to a subset of all possible events. This subset of interest is called a **σ-algebra** $\mathscr{B}(\mathscr{X})$. More specifically, a σ-algebra $\mathscr{B} \subset \mathscr{P}(\mathscr{X})$ is a sub-algebra of the powerset, completed to include countably infinite operations. The smallest possible σ-algebra is a collection of just two sets, $\{\mathscr{X}, \emptyset\}$. The largest possible σ-algebra is the collection of all the possible subsets of $\mathscr{X}$, the powerset.

Probability Space Now that we have found a way to enumerate all possible events, we need a function that assigns probabilities to events. That function $P : \mathscr{B} \to \mathbb{R}$, is called a **probability measure** and is defined on the **measurable space** $\Big(\mathscr{X}, \mathscr{B}(\mathscr{X})\Big)$. The combination of a measurable space and a probability measure gives us a **probability space** $(\mathscr{X}, \mathscr{B}, P)$, a space (i.e. a set equipped with some structure) in which the trinity of outcomes, events, and probabilities are rigorously defined.

1.2.2 Defining a Statistical Model

There are infinitely many possible probability measures we can choose from. Indeed, let $\mathscr{M}(\mathscr{X})$ denote the **space of all probability measures** on $\Big(\mathscr{X}, \mathscr{B}(\mathscr{X})\Big)$. Indeed, $\mathscr{M}(\mathscr{X})$

can be too large for many inference problems with limited data. To make inference a less daunting task we can restrict ourselves to a subset of the space by introducing a **parameter** $\theta \in \Theta$ that represents the pattern that explains the data, where the **parameter space** Θ is the set of all possible values of θ. As a result, the probability measures P_θ are now elements of $\mathscr{PM}(\mathscr{X})$, the **space of all probability measures** on Θ with elements $P_\theta \in \mathscr{PM}(\mathscr{X})$ indexed by a **parameter** $\theta \in \Theta$.

A **statistical model** $\mathscr{P}$ therefore is a subset $\mathscr{P} \subset \mathscr{PM}(\mathscr{X})$ such that $\mathscr{P} = \{P_\theta \mid \theta \in \Theta\}$ where $\theta \to P_\theta$ is a bijective and measurable assignment. Here is an example, if we let $\Theta = \mathbb{R}^2$ represent the set of linear functions, then $\theta \in \mathbb{R}^2$ determines the linear trend in simple linear regression.

The model $\mathscr{P}$ is **parametric statistical model** if $\Theta \subset \mathbb{R}^d$ for some $d \in \mathbb{N}$. The subset $\mathscr{P} \subset \mathscr{PM}(\mathscr{X})$ can be restricted further by specifying a family of parametric models $\mathscr{G} = \{G_\theta \mid \theta \in \Theta\}$ where $\theta \to G_\theta$ is smooth. For example, $\mathscr{G} = \{N(\theta, 1) : \theta \in \Theta\}$ specifies the one-dimensional normal location family of models. If on the other hand, Θ is infinite dimensional, then $\mathscr{P}$ is **a nonparametric statistical model**. In this case Θ is equivalent to $\mathscr{M}(\mathscr{X})$, the space of all probability measures on $\Big(\mathscr{X}, \mathscr{B}(\mathscr{X})\Big)$.

1.2.3 Defining a Bayesian Model

A **parametric Bayesian statistical model** $(\mathscr{P}, \Pi)$ consists of a model $\mathscr{P}$, the **observation model**, and a **prior distribution** Π on Θ such that θ is a random variable taking values in Θ and $\Pi\Big(\{P_\theta : \theta \in \Theta\}\Big) = 1$. In a Bayesian model, data is generated hierarchically in two stages:

$$\begin{aligned} \theta &\sim \Pi \\ x_1, x_2, \ldots \mid \theta &\sim_{iid} P_\theta \qquad \theta \in \Theta \subset \mathbb{R}^d \end{aligned} \tag{1.1}$$

After we observe data $(x_1, x_2, \cdots, x_N)$, the prior is then updated to the posterior distribution $\Pi(. \mid x_1, x_2, \cdots, x_N)$.

A **nonparametric Bayesian model** is a Bayesian model whose prior Π is defined on an infinite dimensional parameter space Θ. The corresponding two-stage hierarchical model is given as

$$\begin{aligned} P &\sim \Pi \\ x_1, x_2, \ldots \mid P &\sim_{iid} P \qquad P \in \mathscr{P} \end{aligned} \tag{1.2}$$

A prior distribution on an infinite dimensional space is a **stochastic process**. Defining an infinite dimensional prior distributions is not straightforward, but one way to construct a prior distribution Π on Θ is through De Finetti's Theorem.

De Finetti's Theorem. *A sequence of random variables $\{x_n\}_{n=1}^{\infty}$ with values on $\mathscr{X}$ is **exchangeable** if and only if there is a unique measure Π on Θ such that for all N*

$$\begin{aligned} p(x_1, x_2, ..., x_N) &= \int_{\Theta} \left(\prod_{n=1}^{N} \theta(x_n) \right) \Pi(d\theta) \quad \textit{General Form} \\ &\int_{\Theta} \left(\prod_{n=1}^{N} p(x_n \mid \theta) \right) p(\theta) d\theta \quad \textit{Specific Form} \end{aligned} \tag{1.3}$$

The theorem gives us a infinite mixture representation of the joint probability of the observations. More importantly, it shows that exchangeability implies conditional independence of the observations given θ.

Chapter 2: A Conceptual Overview of Latent Variable Models

This chapter has two main goals. The first goal is to present the generative probabilistic modeling perspective and illustrate its proper application with the aid of an example. The second goal is to provide a conceptual overview of latent variable models within a probabilistic modeling framework, an overview that emphasizes the compositional nature and the interconnectedness of the seemingly disparate models commonly encountered in statistical practice. We begin by explaining what a latent variable is, provide a general formulation of the latent variable models encountered in the chapter, and give a brief history of latent variable modeling. We then proceed to clarify the probabilistic modeling perspective using a concrete example in order to show how the probabilistic approach works in practice. This detailed foray into model building is important since it shows how the probabilistic framework forces the researcher to make explicit all the numerous assumptions that goes into model formulation. It is also important since it motivates much of the material found in the section, where we attempt to show how we can arrive at many of the most complex latent variable models by building and modifying on the basic foundations introduced earlier in the example.

2.1 What is a Latent Variable?

Many definitions for a latent variable have been proposed in the literature (see Bollen (2002) for a review). Although the definitions that have been put forward range from the informal to the mathematical, many of them rely on distinctions that can be conceptually problematic when working in a probabilistic modeling framework. For example, defining a variable as *latent* based on whether it is treated as random rather than fixed does not carry over to the probabilistic framework where all variables, known or unknown, are treated as random. Furthermore, definitions that are based on the notion of unobservability introduce some conceptual issues that complicate any attempt to give a simple definition based on that concept alone. In a probabilistic model, a variable is either observable or unobservable, but not all unobservable variables in a probabilistic model correspond to what we think of as a latent variable when we are working outside the probabilistic framework. In particular, definitions based on unobservability break down when considering how to treat the random noise term ε_n, whether it is found in a simple regression model or in an implicit model that transforms the distribution of the error into a more complicated distribution $g(\varepsilon_n;\theta)$ (Goodfellow et al., 2014). In both cases, the error term is an unobservable random quantity whose distribution deterministically induces a distribution on the response $\mathbf{y}_n$ through an injective function g whose inverse is g^{-1}. Nonetheless, it cannot be considered a latent variable since the posterior

$$p(\varepsilon_n|\mathbf{y}_n) = \mathbb{I}[\varepsilon_n = g^{-1}(\mathbf{y}_n)] \tag{2.1}$$

is a point mass, a deterministic not a random function (Tran, Ranganath, & Blei, 2017). Note that for a function to have an inverse, the dimension of ε_n should equal the dimension of $\mathbf{y}_n$. It could be argued that for variable to be considered a latent variable it needs be a

unobserved random variable whose posterior distribution is not degenerate. For example, in the Tobit model (Tobin, 1958)

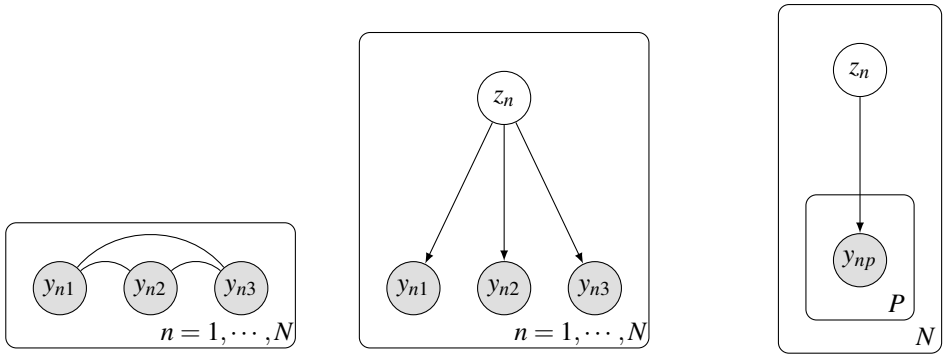
$$\begin{aligned} \varepsilon_i &\sim N(0,\sigma^2) \\ y_i^* &= \beta x_i + \varepsilon_i \\ y_i &= max(0, y_i^*) \end{aligned} \tag{2.2}$$

the error term ε_i accordingly would not be considered a latent variable since the function $g(x) = m + sx$ is invertible given that both ε_i and y_i^* are scalars (i.e. both of dimension one). In contrast, y_i^* can be considered a latent variable since its posterior is not a point mass and the function $g(x) = max(0,x)$ is not invertible, even though both y_i^* and the observed variable y_i are scalars. To complicate things further, one could argue that in the Tobit model, we are in fact dealing with a partially observable variable model $y_i = max(0, \beta x_i + \varepsilon_i)$ rather than with a latent variable model per se.

Requiring a latent variable to be an unobservable random variable *with a non-degenerate posterior distribution* can provide us with an adequate working definition of a latent variable. Still we argue that a latent variable can be better understood by the function it serves in a model. In particular, we propose that the inclusion of a latent variable in a statistical model allows us to accomplish two tasks: First, to capture statistical dependencies in the data and second, to learn a latent structure that potentially governs the data generating mechanism for the observations (i.e. learn the posterior distribution $p(\mathbf{z}|\mathbf{y})$). With this in mind, note that the notion of a latent variable as widely understood in the majority of modeling context is most intuitively explained by the *local independence definition* of a latent variable (Lord, 1952). We qualify the preceding statement by the word *majority* because in the local independence definition does not always hold for non-probabilistic models such as SEM where the latent variables can regressed on each other to induce dependencies

among the latent variable. Furthermore, the local independence definition might not be as intuitive in the case of models, whether estimable or not, in which there are a great number of latent variables that combine together to give rise to an observed response of a much lower dimension (Thomson, 1916).

To illustrate the basic idea, consider the following toy example. We observe N *multivariate observations*, N samples of three dependent variables $\{(y_{n1}, y_{n2}, y_{n3})\}_{n=1}^{N}$ (Figure 2.1a). The inclusion of a scalar latent variable z_n for the nth observation allows us to factorize the joint distribution of (y_{n1}, y_{n2}, y_{n3}) into a set of conditionally independent factors, as shown by the directed graph in Figure 2.1b. For example, if we have a set of P dependent variables where the dependency is restricted to positive linear association among them, then conditioning on just one single latent variable can be sufficient to explain the pattern of dependencies in the P dimensional data. The resulting latent variable model for the N observations can be concisely expressed by the directed graph in Figure 2.1c.



(a) MARGINAL MODEL **(b)** CONDITIONAL MODEL **(c)** FINITE-DATA MODEL

Figure 2.1: **(a)** Marginal graphical model for the three dependent variables. **(b)** Directed graph of the observed variables conditional on the latent variable. **(c)** The latent variable model as a directed graphical model for a sample of N multivariate observations, each a P-dimensional vector.

2.2 What is a Latent Variable Model?

Consider a multivariate data set consisting of N multivariate observations denoted by $\mathbf{y} = (\mathbf{y}_1, \cdots, \mathbf{y}_N)$ where each observation vector $\mathbf{y}_n = (y_1, \ldots, y_P)$ contains P measurements. A *latent variable model* can be defined as a statistical model of observed data that incorporates *latent variables*. Accordingly, to construct a latent variable model we start with incorporating an D-dimensional *local latent* variable vector $\mathbf{z}_n = (z_1, \ldots, z_D)$ for each observation and a M-dimensional *global latent* variable vector $\theta = (\theta_1, \ldots, \theta_M)$ shared across all observations (Blei, Kucukelbir, & McAuliffe, 2016). The local latent variable we include, $\mathbf{z}_n$, serves the function of capturing local dependencies *within* the observation vector $\mathbf{y}_n$ (e.g. clusters or common factors) whereas the global latent variable θ captures the dependencies *across* all N observation vectors. In a probabilistic modeling framework, the starting point is always the joint probability distribution of all observed and unobserved variables as such.

$$p(\mathbf{y}_1, \cdots, \mathbf{y}_N, \mathbf{z}_1, \cdots, \mathbf{z}_N, \theta) \tag{2.3}$$

We can see that there are countless many ways we could factor the joint distribution given by Equation 2.3. Each chosen factorization encodes a set of implicit assumptions that make the particular decomposition possible. For example, the following decomposition

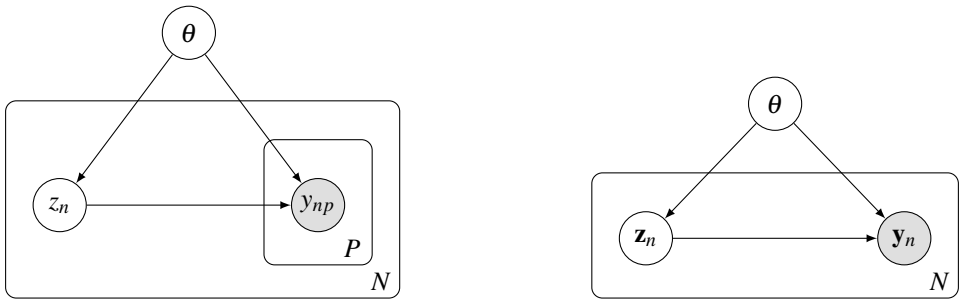
$$p(\mathbf{y}_{\leq N}, \mathbf{z}_{\leq N}, \theta) = p(\theta) \prod_{n=1}^{N} p(\mathbf{y}_n \mid f_x(\mathbf{z}_{\leq n}, \mathbf{y}_{<n}), \theta) p\big(\mathbf{z}_n \mid f_z(\mathbf{z}_{<n}, \mathbf{y}_{<n}), \theta\big) \tag{2.4}$$

supports a very general class of temporal generative models (Gemici et al., 2017), including hidden Markov models, non-linear state space models (Tornio, Honkela, & Karhunen, n.d.), and the Variational Recurrent Neural Network (VRNN) (Chung et al., 2015). If we are not modeling temporal or sequential data, then we can assume that the N multivariate

observations are *exchangeable* and obtain the following decomposition instead.

$$p(\mathbf{y}_1, \cdots, \mathbf{y}_N, \mathbf{z}_1, \cdots, \mathbf{z}_N, \theta) = p(\theta) \prod_{n=1}^{N} p(\mathbf{y}_n \mid \mathbf{z}_n, \theta) p(\mathbf{z}_n \mid \theta) \tag{2.5}$$

The decomposition subsumes a very broad class of latent variable models such as hierarchical Bayesian regression models (Gelman, Carlin, Stern, & Rubin, 2014), Bayesian mixture models, additive matrix decomposition models (e.g. factor analysis, PCA, and CCA), and Bayesian nonparametric models. A graphical representation of this general latent variable family of models is shown in Figure 2.2.



(a) Unidimensional Latent Variable Model **(b)** Multi-dimensional Latent Variable Model

Figure 2.2: **(a)** For each sample observation n, z_n is a local latent variable, and the $y_{np'}$ are P observed variables. θ is a vector of global latent variables (parameters) **(b)** Directed graph of the observed variables conditional on the latent variable.

Notice that the decomposition given by Equation 2.5 implies a second assumption; specifically, that the N local latent variables $\mathbf{z}_n$ are exchangeable in their joint distribution. The joint distribution here refers to the prior $p(\mathbf{z}_n \mid \theta)$ which is governed by the global latent variable θ. The role of global latent variable in the model is to control the extent of information pooling *across* the observations. To illustrate, if we set the variance of the prior distribution to infinity, we obtain a separate model for each group as a result. That

is, a different model is fit to each observation vector, where $\mathbf{y}_n$ can be thought of as an "experiment" with P measurements. On the other extreme, setting the variance to zero would result in *complete pooling* of information, effectively giving us a single model for all $N \times P$ collapsed observations (Gelman et al., 2014). Incidentally, in the context of mixed effects models, it is interesting to point out that a fixed effect can be thought of as a special case of a random effect, but one with an infinite variance prior.

Also note that the decomposition given by Equation 2.5 above presumes that the elements of the local latent variable vector $\mathbf{z}_n$ form a single layer. *Deep models* in contrast contain several hidden layers that form a rich hierarchical structure. To obtain a joint factorization that includes deep models as a general case, we just need to divide the vector of local unobserved variables $\mathbf{z}_n$ into L subsets $\mathbf{z}_n = \{\mathbf{z}_{n,1}, \cdots, \mathbf{z}_{n,L}\}$, such that each $\mathbf{z}_{n,1}$ corresponds to a separate level in the sampling hierarchy. Unlike shallow models, deep models with a hierarchy of latent variable dependencies are better able to capture rich latent structure (Ranganath, Tang, Charlin, & Blei, 2015).

2.3 Brief History of Latent Variable Models

Although latent variable modeling is a branch of multivariate statistics, many of the field's major developments originated and were motivated by research problems, not within statistics, but rather from the psychological and social sciences (Izenman, 2008). Whereas

the early breakthroughs mainly came from psychology, educational measurement, and economics, the more recent advances are increasingly coming from the field of machine learning, especially from the very active research area of unsupervised learning in which researchers are trying to accomplish the goal of endowing computers the ability to learn complex data patterns in an unsupervised automatic manner (Lake, Salakhutdinov, & Tenenbaum, 2015; Rezende, Mohamed, Danihelka, Gregor, & Wierstra, 2016).

The quest to understand learning and intelligence, whether in humans or machines, has always been a motivating force in the field from its early beginnings. Indeed, the psychologist Charles Spearman, influenced by Francis Galton's development of the correlation coefficient, introduced *factor analysis* in 1904 as a statistical method to study the organization of mental abilities in humans (Spearman, 1904). He proposed that from the correlations of multiple observed variables (e.g. tests of general ability), we can infer a single latent factor that can explain the common variance shared among all the observations. He referred to the latent variable common to all the measured ability tests as g, the general factor of intelligence.

2.3.1 Summary of Key Developments

The factor analytic approach paved the way for, what seemed to many in the social and psychological sciences, an objective and quantitative framework for measuring unobservable *hypothetical constructs* such as "self-esteem", "verbal ability", and "quality of life". Indeed Spearman's single factor model, in which both the one latent variable and the observed variables are assumed continuous, has laid the foundation for a long research program in psychometrics that has spanned well over a century. To account for the inadequacy

of a single factor to account for the covariance pattern in testing data, the psychologist Thurstone extended the single factor model to multiple factors (Thurstone & L.L., 1947). The factor model was constantly refined and generalized, culminating in the *structural equation model* introduced by (Jöreskog, 1970) in which linear structural relationships between the latent variables are also modeled. In educational testing, *item response theory* models (Birnbaum, 1968) were developed to accommodate categorical observed variables encountered in testing in which test items assume two values, correct and incorrect. In sociology, clustering analysis motivated the development of *latent class models* (i.e. mixture models) in which the latent variables are assumed to be categorical (Lazarsfeld, 1950). Since then, a large body of work has been built on top of these foundational early advances and many authors (Muthén, 2002; Bartholomew, Knott, & Moustaki, 2011; Hwang & Takane, 2014) have attempted to bring together the various models under one roof in a general framework. The most encompassing as of yet is the Generalized Linear Latent and Mixed model (GLLMM) introduced by Rabe-Hesketh, Skrondal, and Pickles (2004) as a unified general framework that brings together not just factor analysis, IRT, structural equations, and latent class models commonly encountered in psychometrics and social quantitative methods, but also mainstream statistical random effects models such as multilevel regression, and longitudinal models.

2.3.2 The Statistical Myth of Measuring Hypothetical Constructs

The enthusiasm regarding the factor analytic approach to uncover underlying theoretical constructs was not universally shared. In particular, the interpretation of a latent variable as a hypothetical construct was much criticized. One of the earliest criticism came from Spearman's colleague, Karl Pearson, widely regarded as the founder of modern mathematical

statistics. The two men constantly feuded with each other throughout their careers due to their opposing scientific philosophies (Hägglund, 2001). Whereas Spearman strove to discover the fundamental laws of psychology and invent a method by which he can objectively measure latent traits such as intelligence, Pearson did not see a place for unobserved and subsequently unmeasurable phenomena in the scientific enterprise. Incidentally, Pearson (1901) had pioneered a method closely related to, and often conflated with Factor Analysis (FA) - *principle component analysis* (PCA). Pearson's PCA, which was later formulated as multivariate technique by Hotelling (1933), was introduced not as an explanatory model, but as a dimensionality reduction technique in which the data is geometrically transformed (i.e. projected) into a lower dimensional subspace.

The criticisms nonetheless did not end with Pearson. The English educational psychologist Thomson (1916) proposed a sampling model that generated the same correlation structure as a general factor model but that did not assume a single factor. In particular, whereas Spearman's model, and later extensions, proposed that positive correlations among P variables suggest the existence of one or more factors fewer than P, Thomson's sampling model assumed the opposite: that there are more factors than we have variables and that each variable is the result of a great number of independent causes, much greater than P. Even though it seems more plausible to assume that there are a great number of factors (mental subprocesses) that contribute to a score on any one ability test or item questionnaire, psychometricians following Spearman's footsteps ignored these criticisms and pursued a research program based on the indefensible premise that factor analysis and its extensions are statistical methods that can explain correlations in the observed data, rather than just describe them. It can be argued that the explanatory approach in which the existence of a hypothetical construct is inferred by performing exploratory analysis on

limited sample self-report data cannot substitute the rigorous process of developing operational definitions of unobservable constructs on theoretical grounds or even on experimental measurements of the cognitive processes underlying the observed responses. Indeed, throughout the history of psychometrics, there have been repeated calls to guard against the *reification fallacy* in which a latent variable (e.g. mental ability) that captures a common variability in the data is treated as if it were a physical quantity (e.g. height) that is amenable to direct measurement (Gould, 1996; Sobel, 1994).

2.3.3 The First Latent Variable Model

If we restrict latent variable models to those models in which latent variables denote hypothetical constructs, arguably Spearman's factor model could be considered the first formulation of a latent variable model. However, such a restriction based on a narrowly defined interpretation is not universally accepted neither within psychometrics, the field Spearmen helped start, nor within the larger statistics community (Skrondal & Rabe-Hesketh, 2004). If we just consider the statistical form of the model, then it could be argued that the first latent variable model should be attributed to the astronomer Airy, who in 1861 proposed a variance components model (i.e. random effects model) for the repeated measurement of an astronomical object over several nights (Airy, 1861). Both Airy's variance components model and Spearman's factor model have the same graphical representation depicted in Figure 2.1c. The main difference lies in that the latent variable in Airy's model is a random effect that captures unobserved heterogeneity whereas in Spearman's model the latent variable denotes a hypothetical construct which we seek to measure.

In his model, Airy makes P repeated measurements of an astronomical phenomenon for each of N nights. See Searle, Casella, and McCulloch (2009) for a more details and

the original formulation. Assuming a balanced design, the model can be expressed as a probabilistic generative model by the following two-stage sampling hierarchy.

$$\begin{aligned} \mathbf{y}_n \mid z_n &\sim \mathcal{N}_P(\mathbf{1}\mu + \mathbf{1}z_n, \sigma_\varepsilon^2 \mathbf{I}) \\ z_n &\sim \mathcal{N}(0, \sigma_z^2) \quad \textit{for } n = 1, \cdots, N \end{aligned} \tag{2.6}$$

where $\mathbf{y}_n \in \mathbb{R}^P$ is the vector of repeated measurements and $z_n \in \mathbb{R}$ is a latent variable (i.e. random effect).

2.3.4 Latent Variable Models in Machine Learning

Although the GLLMM framework does indeed treat random coefficient models (which include hierarchical and longitudinal models) as latent variable models, the framework leaves out important multivariate data reduction methods such PCA, Independent Component Analysis (ICA), Canonical Correlation Analysis (CCA) from the formulation. The argument put forward by Skrondal and Rabe-Hesketh (2004) contends that PCA and similar procedures are not latent variable models because they cannot be formulated as statistical models with latent variables corresponding to lower dimensional space. However, their assertion does not seem to have taken into consideration results that had been previously published in the literature. More specifically, in recent years, research in latent variable modeling has undergone a very active resurgence in the field of unsupervised machine learning. Indeed, great progress has been made in developing new latent variable models and recasting existing clustering and dimensionality reduction methods in a probabilistic graphical model formulation. For example, building on previous research in neural networks (MacKay, 1995a; Bishop, Svensén, & Williams, 1998), Tipping and Bishop (1999) demonstrated that PCA can be indeed be formulated as a probabilistic latent variable model. In their paper, they showed that Probabilistic PCA is a special case of Factor Analysis (FA).

More specifically, whereas in FA the error covariance matrix is diagonal, in Probabilistic PCA, it is $\sigma\mathbf{I}$ and standard PCA is obtained when $\sigma \to 0$. Roweis and Ghahramani (1999) review several common methods showing that not just PCA and FA, but also ICA, Kalman Filters, and Hidden Markov Models (HMMs) can be considered as special cases of a general linear-Gaussian latent variable model. A probabilistic formulation of CCA (Bach & Jordan, 2005) later followed. Recently, a Group Factor Analysis (GFA) model has been proposed by Klami, Virtanen, Leppäaho, and Kaski (2015) that generalizes Probabilistic CCA to more than two data sets.

It is these developments that motivate the material in the next two sections. In the social sciences, latent variable models have been traditionally been viewed and used as tools to study hypothetical constructs. However, ongoing research in machine learning, deep learning, and probabilistic graphical modeling has given a new life and meaning to latent variable modeling. These recent developments present latent variables as capable of endowing statistical models not only with the ability to perform essentials data analysis tasks such as dimensionality reduction, statistical clustering, and accounting for unobserved heterogeneity, but also with explanatory and interpretative powers. The inclusion of latent variables allows us to capture statistical dependencies among observed variables and thus learn the hidden structure underlying the data generating mechanism. This is especially true when latent variable models are cast in a generative probabilistic framework, a perspective that the next section attempts to describe in detail. The following sections forms the foundation for subsequent sections where many of the most commonly encountered latent variable models are constructed from the most basic elements.

2.4 Constructing Generative Probabilistic Models

It is said that allowing oneself to be puzzled by the seemingly obvious is a useful capacity to cultivate and one that can lead to deeper insights. Indeed, even though some of the material in this section might seem obvious to the reader, we hope that by approaching data modeling from the basics, a more insightful picture emerges as the building blocks are put in place and several statistical insights that are implicitly and explicitly scattered in the statistical literature become slowly woven in the exposition.

Example With the purpose of motivating the material ahead, we first consider the following example. Consider a brain mapping experiment in which neural activity is recorded from multiple locations in the brain while images of natural objects are presented in the visual field of a human subject. Each two-dimensional image stimulus $\mathbf{x}$ consists of M pixels of varying light intensity. The presentation of an image evokes a pattern of neural activity that we measure as $\mathbf{y}$, a P-dimensional neural response sampled from a particular brain area. The data is then collected in a data set $\mathscr{D} = \{\mathbf{y}_n, \mathbf{x}_n\}_{n=1:N}$ consisting of N multivariate observations, where each observation is partitioned into a P-dimensional response vector $\mathbf{y}_n = (y_{n1}, \cdots, y_{nP})$ and an M-dimensional covariate vector $\mathbf{x}_n = (x_{n1}, \cdots, x_{nM})$. The data is provided as N by $(P+M)$ matrix consisting of a total of $N \times (P+M)$ measurements

$$\begin{bmatrix} (\mathbf{y}_1^{\mathrm{T}}, \mathbf{x}_1^{\mathrm{T}}) \\ \vdots \\ (\mathbf{y}_n^{\mathrm{T}}, \mathbf{x}_n^{\mathrm{T}}) \\ \vdots \\ (\mathbf{y}_N^{\mathrm{T}}, \mathbf{x}_N^{\mathrm{T}}) \end{bmatrix} = \begin{bmatrix} y_{11} & \cdots & y_{1P} & x_{11} & \cdots & x_{1M} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ y_{i1} & \cdots & y_{nP} & x_{n1} & \cdots & x_{nM} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ y_{N1} & \cdots & y_{NP} & x_{N1} & \cdots & x_{NM} \end{bmatrix} \tag{2.7}$$

Now we ask ourselves the following question. How should one best proceed with the task of predicting the neural responses at each of the brain regions evoked by an image stimulus?

Running Procedures Vs Building Models A simple *procedure-centric* approach to the above-posed prediction problem would proceed by fitting a set of P regression models to the data, a separate model for each response variables p'

$$y_{np'} \mid \mathbf{x}_n \sim \mathcal{N}(\beta_{p'}^{\mathrm{T}}\mathbf{x}_n, \sigma_{p'}^2) \quad for\ n = 1, \cdots, N \tag{2.8}$$

Accordingly, the data distribution model for the entire data is given by

$$\prod_{n=1}^{N}\prod_{p=1}^{P} \mathcal{N}(y_{np} \mid \beta_p^{\mathrm{T}}\mathbf{x}_n,\ \sigma_p^2) \tag{2.9}$$

The inference procedure would produce estimates for a total of $P \times (M+1)$ parameters, $\{\left(\beta_p^{\mathrm{T}}, \sigma_p^2\right)\}$, which we can then use to build a predictive model for future observations. If we are lucky, the assumptions implicit in the decomposition given by Equation 2.9 could turn out to be valid and we learn a model that is highly predictive and explains the data well. Unfortunately, more often than not, either the simplistic assumptions do not usually hold in reality or the resulting inference is far from optimal. For this reason, one important theme that this thesis attempts to highlight and argue for is that there is an inherent danger in approaching statistical inference as a problem that can be solved by throwing the "correct" off-the-shelf statistical procedure at it. We do not argue against a particular model per se (e.g. regression) or how data analysis should be conducted in general but rather we argue for how statistical modeling should be approached and practiced *in the sciences*. To understand and uncover the workings of natural phenomena, we need a disciplined and statistically rigorous approach to modeling. We argue that the probabilistic modeling

approach (Pearl, 2014; Gelman et al., 2014), which we attempt to detail in the next section, does provide a more disciplined approach to inference for a large majority of statistical modeling problems in the sciences.

2.4.1 The Probabilistic Modeling Approach

A quick insight into the essential aspect of the probabilistic modeling perspective can be gained by contrasting it with its opposite; i.e., with what *it is not*. The probabilistic approach does not construct statistical models using algebraic relations. More importantly, reasoning about how observed and unobserved quantities relate to each other and how their dependencies are induced does not involve specifying structural relationships using algebraic formulas. Instead, all observed and unobserved events that are thought to represent the phenomenon of interest are denoted by random variables and each event is allocated a finite probability of occurring. That is to say, the generative probabilistic modeling approach is Bayesian in spirit (Pearl, 2014; Lauritzen, 1996; Jebara, 2012). Model building starts with the joint distribution of all observable and unobservable variables and proceeds to factor the joint into a product of conditional and marginals distributions. Through conditioning, dependencies are introduced that transform the joint density into a mixture of densities. Implicit in this mixture is a data-generating process that gives rise to the observations. To illustrate the approach, let's begin by introducing θ, a parameter representing the pattern that governs the distribution of the data. The parameter θ can be a scalar, a multidimensional vector, or even an infinite dimensional stochastic process. We can now express the joint distribution for all $N \times (P+M)$ observations and the parameters as follows (where we use the response and stimulus vectors and grouped together in pairs).

$$p(\mathbf{y}_1, \mathbf{x}_1, \cdots, \mathbf{y}_N, \mathbf{x}_N, \theta) \tag{2.10}$$

Note that we are now treating the parameter, the image stimulus, and the neural response as random quantities. Before we proceed further, it seems instructive to pause and examine what the preceding statement implies. What does it mean to treat an image as a random vector for example? To explain, note that since our image stimulus is represented as an M dimensional vector, any particular image can be thought of as a point in a high-dimensional R^M image space from which a $\mathbf{x}_n$ is just one sample. Researchers in computer vision have observed that assuming random pixel values, the vast majority of possible images do not correspond to recognizable images of natural objects and accordingly we should not expect the distribution of natural images to be uniform over the entire image space. Indeed, according to the manifold hypothesis (Narayanan & Mitter, 2010), probable images are neighbors of other probable images, or more formally stated, the data points of all possible natural objects lie near a low-dimensional manifold embedded in a high-dimensional space of all possible images. The manifold hypothesis implies that the dimensionality of the space of natural images is much lower than that of the overall image space and that natural images are generated by a few factors of variation that can explain the structure in the data. The hypothesis is also associated with the assumption of natural clustering by which different categories of objects correspond to different manifolds separated by vast regions of low probability. The ideas of manifolds and natural clustering are important because they underlie the concept of representation that is fundamental not just to theories of object categorization, but also to the field of representation learning in which the latent variables are treated as random variables representing the underlying explanatory factors of variation that generate the data (Bengio, Courville, & Vincent, 2013).

The generative approach requires much more thought to apply in practice since there are a large number of ways we can decompose the joint distribution into a product of factors.

Since each possible decomposition implicitly encodes a particular number of modeling assumptions, the researcher is forced to examine the validity of and the motivation for all modeling assumptions. Furthermore, each component of the factorization needs to be assigned a probability distribution that respects the plausibility of what is known about the putative mechanism that gives rise to the data. For example, the set of P regression models introduced above imply the following factorization of the full joint probability distribution.

$$p(\mathbf{y}_1, \mathbf{x}_1, \cdots, \mathbf{y}_N, \mathbf{x}_N, \theta) = \prod_{n=1}^{N} \prod_{p=1}^{P} \mathscr{N}(y_{np} \mid \beta_p^{\mathrm{T}} \mathbf{x}_n,\ \sigma_p^2) \tag{2.11}$$

where both $\theta = \{\left(\beta_p^{\mathrm{T}}, \sigma_p^2\right)\}_{p=1}^{P}$ and the $\mathbf{x}_n$ variables are assumed to be fixed, or equivalently as random variables with infinite variance priors. To arrive at this particular factorization many assumptions that usually go unexamined need to be made. In what follows, we attempt to make explicit all that simplifying assumptions that would takes from the full joint probability distribution to the decomposition implied by the regression model.

Recording Measurements in Space and Time Rather than start with the observed data, we begin our model building task with examining the individual measurement events that produce the recorded data. The main point we wish to emphasize is that the event of taking a measurement occurs within a four-dimensional spatiotemporal frame of reference. Therefore, we denote an observation not as y, but rather as $y(d_1, d_2, d_3, t)$, a measurement recorded at spatial location $\mathbf{d} = (d_1, d_2, d_3)$ and time t, where $\mathbf{d} \in D \subset \mathbb{R}^3$ and $t \in T \subset \mathbb{R}$. Note that we are not considering here point process models in which D is random such that $\mathbf{d}$ denotes the location of random events. In many applications, spatiotemporal information is assumed to have a miniminal effect on the observed data and is therefore omitted. For example, if we wish to measure the heights of a number of individuals at a certain point in time, it could be safe to assume that the spatial arrangement of the subjects and the brief

time interval that passes between the measurements are inconsequential. If on the other hand, the height measurements for each subject are taken multiple times over a period of months, then the time index would need to be retained. To illustrate, if the height of individual i is recorded repeatedly over three occasions, we would then have the measurements $[y(t_1^*), y(t_2^*), y(t_3^*)]$. For uniform time intervals, the notation can be simplified and the vector of repeated measures can be denoted as $[y_1, y_2, y_3]$. In other applications, the location of measurements is an essential aspect of understanding the phenomenon and cannot be ignored. In either case, the form in which the data is presented (e.g. matrix) implicitly imposes certain assumptions on us. To illustrate, note that in the example provided the goal of conducting the experiment is to understand the temporal dynamics of neural activation spatial patterns. It follows then that we expect that each measured response y_{np} and each image pixel x_{nm} to have a corresponding spatiotemporal index. However, given that the data is presented to us as a matrix without including the spatiotemporal indices, there are a few questions that we need to consider before we start developing a model; namely the following.

- By grouping the response and covariate vectors as a pair $(\mathbf{y}_n, \mathbf{x}_n)$ are we assuming that the time period that elapses between presenting the stimulus and measuring the response is inconsequential?
- Does $\mathbf{d}$ vary continuously over D or is D discretized?
- Does t vary continuously over T or is T discretized?
- In case of discrete time, are we dealing with multivariate time series (i.e. the spatial coordinates are the same for each time period) or cross-sectional data (i.e. the spatial coordinates vary over the time periods)?

To reach our destination, we will then need to assume that both T and D are discretized uniformly, more specifically, that D is divided into a regular P dimensional lattice where $\mathbf{d}_p$ then represents the spatial coordinates of pth block and that T is divided into equal time intervals, where each measurement represent an average over an time interval and a spatial block. We will also need to assume that the spatial coordinates are fixed for all the time periods and that there is no lag between the presentation of the stimulus and the evoked neural response. Consequently, the time and space indices would no longer be needed since the row and column indices of the data matrix are sufficient to preserve the spatiotemporal structure of the measurements, in the sense that the complex 3D pattern of spatial dependencies can still be inferred from the one dimensional response vector.

As a matter of fact, much of the work that goes into preprocessing fMRI data for example, attempts to ensure that these basic assumptions are met before the preprocessed data is fed into downstream analysis steps (Poldrack, Mumford, & Nichols, 2011).

2.4.2 Spatiotemporal Models

The observations that populate the data matrix are not exchangeable neither in the spatial nor the temporal dimension. That is, neither the rows nor the columns of the matrix are invariant under permutation. The N samples of the neural response measurements can be characterized by a pattern of time dependencies. For example, if we let the subscript n index time, the response $\mathbf{y}_n$ can be dependent on the preceding response $\mathbf{y}_{n-1}$. Moreover, both the response and stimulus vectors can exhibit patterns of spatial dependencies that are unique to the activity of neural circuitry and the statistics of natural images, respectively. We can learn both the temporal and spatial dependencies of the data using a spatio-temporal model. With a non-separable spatio-temporal model, the spatial and temporal components

are allowed to interact. Such a situation is commonly encountered when modeling dynamical systems in which the current spatial component of the process evolves as a function of past spatial locations. Although non-separable models tend to be realistic for many real world applications, their estimation can be computationally intractable, especially in the case when the dimensionality of M or P is high. When adequate motivation exists, a separable spatio-temporal model can be used instead by assuming that the parameters that govern the distribution of the temporal component are independent of those that govern the spatial component $\theta = (\theta^{YX}, \theta^{T})$. Note that the parameter θ^{YX} governs the distribution of the pairs of $\mathbf{y}_n$ and $\mathbf{x}_n$ jointly, but if we wish to predict $\mathbf{y}_n$ from $\mathbf{x}_n$, we need to assume that the parameters that govern the distribution of the neural responses are independent of those that govern the distribution of the images. That is $\theta^{YS} = (\theta^{Y}, \theta^{X})$. After conditioning on the stimuli, the joint distribution $p(\mathbf{y}_1, \mathbf{x}_1, \cdots, \mathbf{y}_N, \mathbf{x}_N, \theta^{Y}, \theta^{X}, \theta^{T})$ can then be decomposed as such.

$$p(\mathbf{y}_1, \cdots, \mathbf{y}_N, \theta^{T} \mid \mathbf{x}_1, \cdots, \mathbf{x}_N, \theta^{Y}) p(\mathbf{x}_1, \cdots, \mathbf{x}_N, \theta^{T} \mid \theta^{X}) p(\theta^{Y}) p(\theta^{X}) \tag{2.12}$$

The factorization implies that the distribution of the stimuli does not influence the conditional distribution of the responses. The assumption allows us to ignore the distribution of the stimuli when the focus is on determining the parameter θ^{Y} - such as the case in many regression or path analysis models for example, where the $\mathbf{x}_n$ are considered exogenous.

2.4.3 Multivariate Models

A spatiotemporal model can be viewed either as a temporally varying spatial model or as spatially varying time series model (S. Banerjee, Carlin, & Gelfand, 2014). In either case, the data modeled is assumed to be characterized by both spatial and temporal dependencies. Multivariate models in contrast are appropriate for modeling data that exhibit

patterns of dependency in one dimension only (either space or time). Let's say that we would like to predict the brain's neural responses from image stimuli such that the N multivariate observations are treated *exchangeable*. Since exchangeability implies conditional independence, we can factor the joint distribution as such

$$p(\mathbf{y}_1, \cdots, \mathbf{y}_N, \mathbf{x}_1, \cdots, \mathbf{x}_N, \theta) = \prod_{n=1}^{N} p(\mathbf{y}_n, \mid \mathbf{x}_n, \theta^Y) \prod_{n=1}^{N} p(\mathbf{x}_n \mid \theta^X) p(\theta^Y) p(\theta^X) \tag{2.13}$$

where the parameter θ^T is no longer considered.

2.4.4 Normal Models

We now ignore the distribution of the image stimuli for the moment and instead focus on the conditional distribution of the neural responses given the images (the likelihood). If we assume that the responses come from a real-valued distribution with a mean $\mathbb{E}(\mathbf{y}_n \mid \mathbf{x}_n, \theta^Y) = \mu_n(\mathbf{x}_n; \theta^Y)$ and variance $\text{Var}(\mathbf{y}_n \mid \mathbf{x}_n, \theta^Y) = \sigma_n(\theta^Y)$ functions, such that higher moments do not play a role, then the normal distribution would be the natural choice. The resulting normal data distribution encompasses a broad class of normal linear and nonlinear regression models.

$$p(\mathbf{y}_n \mid \mathbf{x}_n, \theta^Y) \equiv \mathcal{N}_p(\mathbf{y}_n \mid \mu_n(\mathbf{x}_n; \theta^Y), \sigma_n(\theta^Y)) \tag{2.14}$$

Note that the subscript in σ_n is necessary when for example the variance is a function of the covariate vector $\mathbf{x}_n$ for the nth sample.

2.4.5 Normal Linear Models

If we split the vector of parameters into two components such that

$$\theta^Y = [vec(\mathbf{B}),\ vech(\Sigma)]$$

where $\mathbf{B}$ is matrix and Σ is positive definite matrix, and further assume that

$$\mu(\mathbf{x}_n;\mathbf{B}) = \mathbf{B}\mathbf{x}_n \quad \textit{and} \quad \sigma_n(\Sigma) = \Sigma \tag{2.15}$$

we arrive at following data distribution

$$\mathbf{y}_n \mid \mathbf{x}_n \sim \mathcal{N}_p\left(\begin{bmatrix}\beta_{11} & \cdots & \beta_{1m}\\ \beta_{21} & \cdots & \beta_{2m}\\ \vdots & \ddots & \vdots\\ \beta_{p1} & \cdots & \beta_{pm}\end{bmatrix}\mathbf{x}_n,\ \Sigma\right) \quad \textit{for} \quad n = 1:N \tag{2.16}$$

Note that since both $\mathbf{y}_n$ and $\mathbf{x}_n$ are multivariate vectors, $\mu(\mathbf{x}_n;\xi)$ is vector-valued function that transforms $\mathbf{x}_n$ into $\mathbf{y}_n$. The joint probability distribution can now be factored into

$$\prod_{n=1}^{N}\mathcal{N}_p(\mathbf{y}_n \mid \mathbf{B}\mathbf{x}_n,\ \Sigma)p(\mathbf{B},\Sigma)\prod_{n=1}^{N}\ p(\mathbf{x}_n \mid \theta^X)p(\theta^X) \tag{2.17}$$

2.4.6 Linear Regression Models

The specification of a particular covariance structure for Σ allows us to model different forms of dependent data. For example, to model stationary time series data, we can specify the Toeplitz matrix structure

$$\Sigma = \sigma^2\begin{bmatrix}1 & \rho_1 & \rho_2 & \cdots & \rho_{(p)}\\ \rho_1 & 1 & \rho_1 & & \vdots\\ \rho_2 & \rho_1 & \ddots & \ddots & \rho_2\\ \vdots & \ddots & \ddots & \ddots & \rho_1\\ \rho_p & \cdots & \rho_2 & \rho_1 & 1\end{bmatrix} \tag{2.18}$$

If we instead let $\Sigma = \mathrm{Diag}(\sigma_1^2,\cdots,\sigma_p^2)$ so that the elements of the response vector now become uncorrelated, the decomposition simplifies further since uncorrelated variables that are jointly multivariate normal imply univariate independent normal variables.

$$\prod_{n=1}^{N}\mathcal{N}_p(\mathbf{y}_n \mid \mathbf{B}\mathbf{x}_n,\ \Sigma) = \prod_{n=1}^{N}\prod_{p=1}^{P}\mathcal{N}(y_{np} \mid \beta_p^{\mathrm{T}}\mathbf{x}_n,\ \sigma_p^2) \tag{2.19}$$

The above simplification implies that a multivariate regression model can be broken down into a set of P multiple regression problems. For example, if we only consider one response variable $y \in \mathbf{y}$, we can then drop the subscript and obtain the following decomposition for a multiple linear regression model.

$$\prod_{n=1}^{N} p(y_n \mid \mathbf{x}_n) = \prod_{n=1}^{N} \mathcal{N}(\mathbf{y}_n \mid \beta^{\mathrm{T}} \mathbf{x}_n,\ \sigma^2) \tag{2.20}$$

2.5 Constructing Probabilistic Latent Variable Models

This section is concerned with demonstrating how to formulate probabilistic latent variable models from the ground up. Our starting point is the decomposition given in Equation 2.17 for the multivariate regression model - which is reproduced here.

$$\prod_{n=1}^{N} \mathcal{N}_p(\mathbf{y}_n \mid \mathbf{B}\mathbf{x}_n,\ \Sigma) p(\mathbf{B}, \Sigma) \prod_{n=1}^{N} p(\mathbf{x}_n \mid \theta^X) p(\theta^X) \tag{2.21}$$

If we consider the predictor $\mathbf{x}_n$ to be fixed, and the parameters that govern $\mathbf{x}_n$ and $\mathbf{y}_n$ to be independent, and the priors over the parameters $p(\mathbf{B}, \Sigma)$ to be uninformative, we reduce the decomposition to

$$\prod_{n=1}^{N} \mathcal{N}_p(\mathbf{y}_n \mid \mathbf{B}\mathbf{x}_n,\ \Sigma) \tag{2.22}$$

2.5.1 Reduced Rank Regression (RRR)

Now if we allow the rank of the regression coefficient matrix to be deficient such that $rank(\mathbf{B}) = d < min(m, p)$, then the matrix can be decomposed into two non-unique matrices as such $\overset{p\times m}{\mathbf{B}} = \overset{p\times d}{\mathbf{W}}\ \overset{d\times m}{\mathbf{D}}$. The decomposition results in what is known as Reduced Rank Regression (RRR) (Izenman, 1975)

$$\prod_{n=1}^{N} \mathcal{N}_p(\mathbf{y}_n \mid \mathbf{W}\mathbf{D}\mathbf{x}_n,\ \Sigma) \tag{2.23}$$

Now if set the predictor vector to be equal to the response vector then the resulting data distribution of the reduced rank regression can be expressed as

$$\prod_{n=1}^{N} \mathcal{N}_p(\mathbf{y}_n \mid \mathbf{WDy}_n,\ \Sigma) \tag{2.24}$$

By letting $\mathbf{z}_n = \overset{d\times m}{\mathbf{D}}\ \mathbf{y}_{m\times 1}$, we can now think of $\mathbf{y}_n$ as a message; $\mathbf{z}_n$ as an encoded transformation of the message; and $\mathbf{W}$ as the decoding transformation. The model can then be expressed as

$$\prod_{n=1}^{N} \mathcal{N}_p(\mathbf{y}_n \mid \mathbf{Wz}_n,\ \Sigma) \tag{2.25}$$

This representation allows us to see that several latent variable models are special cases of *RRR* (Izenman, 2008).

2.5.2 Principle Component Analysis (PCA)

To obtain our first and simplest latent variable model, we specify a distribution to the latent vector since the lower dimensional encoded message is unobservable. Assigning a standard Gaussian to the latent variables and restricting the covariance of the errors to be isotropic gives us probabilistic PCA (Tipping & Bishop, 1999). The model can be expressed as

$$\prod_{n=1}^{N} \mathcal{N}_p(\mathbf{y}_n \mid \mathbf{Wz}_n,\ \sigma^2\mathbf{I})\mathcal{N}_d(\mathbf{z}_n \mid 0, \mathbf{I}) \tag{2.26}$$

or as a generative model

$$\begin{aligned} \mathbf{y}_n \mid \mathbf{z}_n &\sim\ \mathcal{N}_p(\mathbf{Wz}_n, \sigma^2\mathbf{I}) \\ \mathbf{z}_n &\sim\ \mathcal{N}_d(0, \mathbf{I}) \quad \textit{for } n = 1, \cdots, N \end{aligned} \tag{2.27}$$

Standard PCA is obtained as the maximum likelihood solution if we let $\sigma^2 \to 0$ and constrain $\mathbf{W}$ to be orthogonal.

2.5.3 Factor Analysis (FA)

If we relax the constraint of homogeneous variance (i.e. σ^2 is the same for all components of the vector $\mathbf{y}_n$), the covariance matrix can be specified as

$$\Sigma = \text{Diag}(\sigma^2) = \begin{bmatrix} \sigma_1^2 & 0 & \dots & 0 \\ 0 & \sigma_2^2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \dots & 0 & \sigma_p^2 \end{bmatrix} \tag{2.28}$$

The result is a multidimensional factor analysis model with a diagonal covariance and d latent factors.

$$\prod_{n=1}^{N} \mathcal{N}_p(\mathbf{y}_n \mid \mathbf{W}\mathbf{z}_n,\ \text{Diag}(\sigma^2))\mathcal{N}_d(\mathbf{z}_n \mid 0, \mathbf{I}) \tag{2.29}$$

or as a generative model

$$\begin{aligned} \mathbf{y}_n \mid \mathbf{z}_n &\sim \mathcal{N}_p(\mathbf{W}\mathbf{z}_n, \text{Diag}(\sigma^2)) \\ \mathbf{z}_n &\sim \mathcal{N}_d(0, \mathbf{I}) \quad \textit{for } n = 1, \cdots, N \end{aligned} \tag{2.30}$$

Note that the standard normal prior over the latent variables resolves invariance in the scale and location of the latent space, but not in rotation.

2.5.4 Independent Component Analysis (ICA)

Independent Component Analysis (Hyvärinen, 2015) is a widely used model for signal source separation. In order to model signals as latent variables, their distribution cannot be normal because the Gaussian is the most random of all distributions. In fact, according to *Jaynes principle of maximum entropy* (Jaynes, 2003), a Gaussian random variable has maximum entropy (i.e.uncertainty) among all random variables with equal variance. In contrast, entropy tends to be small for spiked, or super-Gaussian distributions - distributions that characterize signals! So if we consider the *generalized Gaussian distribution* as a prior over the latent variables, we obtain a highly peaked supergaussian distribution (if

the shape parameter α set to be less than 2) that is appropriate for modeling signals that give rise to sparse data. Note that the version of the generalized Gaussian distribution we are considering is that of parametric symmetric distributions. The normal distribution is a special case obtained when $\alpha = 2$ and the Laplace distribution is another special case when $\alpha = 1$ (Gómez, Gomez-Viilegas, & Marin, 1998). ICA can be thought of as a linear latent generative model, a type of a Factor Analysis model but with nongaussian priors over the latent variables (i.e. sources). The model has the following decomposition.

$$\prod_{n=1}^{N} \mathscr{N}_p(\mathbf{y}_n \mid \mathbf{W}\mathbf{z}_n,\ \mathrm{Diag}(\sigma^2)) \prod_{n=1}^{N}\prod_{d=1}^{D} \mathscr{GG}_d(z_{d,n} \mid \alpha_d) \quad \alpha_d < 2 \tag{2.31}$$

or as a generative model

$$\begin{aligned} \mathbf{y}_n \mid \mathbf{z}_n \sim\ & \mathscr{N}_p(\mathbf{W}\mathbf{z}_n, \mathrm{Diag}(\sigma^2)) \\ z_{d,n}\ \sim\ & \mathscr{GG}_d(\alpha_d) \quad \alpha_d < 2 \quad d = 1:D; \quad n = 1,\cdots,N \end{aligned} \tag{2.32}$$

where $\mathscr{GG}$ is the generalized Gaussian distribution.

2.5.5 Canonical Correlation Analysis (CCA)

Hotelling (1936) introduced Canonical Correlation Analysis (CCA) as a dimensionality reduction method that projects two sets of random variables (i.e. two data sets) into a shared subspace such that the projections are maximally correlated. Although almost always commonly thought as data reduction technique, CCA can indeed be given a latent variable interpretation as Bach and Jordan (2005) have shown. More specifically, they showed that the equivalent probabilistic latent variable model can be formulated as

$$\begin{aligned} \mathbf{z}_n &\sim \mathscr{N}_d(0, \mathbf{I}) \quad n = 1,\cdots,N \\ \begin{bmatrix} \mathbf{y}_n{}^{(1)} \mid \mathbf{z}_n \\ \mathbf{y}_n{}^{(2)} \mid \mathbf{z}_n \end{bmatrix} &\sim \mathscr{N}_{(p1+p2)} \left(\begin{bmatrix} \mathbf{W}^{(1)} \\ \mathbf{W}^{(2)} \end{bmatrix} \mathbf{z}_n, \begin{bmatrix} \Sigma^{(1)} & 0 \\ 0 & \Sigma^{(2)} \end{bmatrix} \right) \end{aligned} \tag{2.33}$$

where $\mathbf{y}_n{}^{(1)} \in \mathbb{R}^{p_1}$ and $\mathbf{y}_n{}^{(2)} \in \mathbb{R}^{p_2}$, each denote a random vector of observations for Data set 1 and Data set 2, respectively. Note that by replacing the block diagonal covariance matrix

of the data with a diagonal matrix, the probabilistic CCA model reduces to a standard factor analysis model!

They also show that since any covariance matrix of rank D can be written as the following decomposition of the marginal covariance in the marginal joint distribution of the data

$$\begin{bmatrix} \mathbf{y}_n{}^{(1)} \\ \mathbf{y}_n{}^{(2)} \end{bmatrix} \sim \mathcal{N}\left(0, \begin{bmatrix} \mathbf{W}^{(1)}\mathbf{W}^{(1)\mathsf{T}} + \Sigma^{(1)} & \mathbf{W}^{(1)}\mathbf{W}^{(2)\mathsf{T}} \\ \mathbf{W}^{(2)}\mathbf{W}^{(1)\mathsf{T}} & \mathbf{W}^{(2)}\mathbf{W}^{(2)\mathsf{T}} + \Sigma^{(2)} \end{bmatrix}\right) \tag{2.34}$$

it follows that the MLE solution for the joint covariance matrix is such that the cross-covariance matrix is of rank D and consists of the canonical correlation directions that would be obtained in regular CCA.

2.5.6 Inter-Battery Factor Analysis (IBFA)

If we split the vector of latent variables $\mathbf{z}_n$ into three subsets $\{\mathbf{z}_{n,0}, \mathbf{z}_{n,1}, \mathbf{z}_{n,1}\}$ such that the latent vector $\mathbf{z}_{n,0}$ captures the common structure shared across both groups of variables, $\mathbf{y}_n{}^{(1)}$ and $\mathbf{y}_n{}^{(2)}$, while $\mathbf{z}_{n,1}$ and $\mathbf{z}_{n,1}$ each captures the latent structure specific to each group, then CCA model can be modified to have an equivalent specification to the Inter-Battery Factor Analysis (IBFA) model (Tucker, 1958; Browne, 1979).

$$\begin{aligned} \begin{bmatrix} \mathbf{z}_{n,0} \\ \mathbf{z}_{n,1} \\ \mathbf{z}_{n,2} \end{bmatrix} &\sim \mathcal{N}_d(0, \mathbf{I}) \quad n = 1, \cdots, N \\ \begin{bmatrix} \mathbf{x}^{(1)} \mid \mathbf{z}_{n,0}, \mathbf{z}_{n,1} \\ \mathbf{x}^{(2)} \mid \mathbf{z}_{n,0}, \mathbf{z}_{n,2} \end{bmatrix} &\sim \mathcal{N}\left(\begin{bmatrix} \mathbf{W}^{(1,0)} & \mathbf{W}^{(1,1)} & 0 \\ \mathbf{W}^{(2,0)} & 0 & \mathbf{W}^{(2,2)} \end{bmatrix} \begin{bmatrix} \mathbf{z}_{n,0} \\ \mathbf{z}_{n,1} \\ \mathbf{z}_{n,2} \end{bmatrix}, \begin{bmatrix} \Sigma^{(1)} & 0 \\ 0 & \Sigma^{(2)} \end{bmatrix}\right) \end{aligned} \tag{2.35}$$

Note that the above generative formulation makes it apparent that IBFA can be viewed as a special case of CCA, one obtained when we restrict the projection matrix $\mathbf{W}$ to have a

the structure given above; namely

$$\mathbf{W} = \begin{bmatrix} \mathbf{W}^{(1)} \\ \mathbf{W}^{(2)} \end{bmatrix} = \begin{bmatrix} \mathbf{W}^{(1,0)} & \mathbf{W}^{(1,1)} & 0 \\ \mathbf{W}^{(2,0)} & 0 & \mathbf{W}^{(2,2)} \end{bmatrix} \tag{2.36}$$

2.5.7 Multi-Battery Factor Analysis (MBFA)

The Inter-Battery Factor Analysis model can be readily extended to multiple groups of variables. So if we let $\mathbf{y}_n{}^{(g)} \in \mathbb{R}^{p_g}$ for $g = 1, \cdots, G$ denote the groups of variables (i.e. data sets), the vector of observations can be stacked as $P = \sum_g^G p_g$ dimensional vector and the Multi-Battery Factor Analysis (MBFA) model (Browne, 1980; Cudeck, 1982) can be expressed as follows.

$$\begin{gathered} \mathbf{z}_n \sim \mathcal{N}_d(0, \mathbf{I}) \quad n = 1, \cdots, N \\ \begin{bmatrix} \mathbf{y}_n{}^{(1)} \mid \mathbf{z}_n \\ \mathbf{y}_n{}^{(2)} \mid \mathbf{z}_n \\ \vdots \\ \mathbf{y}_n{}^{(G)} \mid \mathbf{z}_n \end{bmatrix} \sim \mathcal{N}_P \left(\begin{bmatrix} \mathbf{W}^{(1)} \\ \mathbf{W}^{(2)} \\ \vdots \\ \mathbf{W}^{(G)} \end{bmatrix} \mathbf{z}_n, \begin{bmatrix} \Sigma^{(1)} & 0 & \dots & 0 \\ 0 & \Sigma^{(2)} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \dots & 0 & \Sigma^{(G)} \end{bmatrix} \right) \end{gathered} \tag{2.37}$$

Again note that by replacing the block diagonal covariance matrix of the data with a diagonal matrix, MBFA reduces to a standard factor analysis model.

2.5.8 Group Factor Analysis (GFA)

Klami et al. (2015) proposed Group Factor Analysis (GFA) as a generalization of multi-battery factor analysis (MBFA). The generative model does still have the same general structure of the preceding latent variable models

$$\begin{gathered} \mathbf{z}_n \sim \mathcal{N}_d(0, \mathbf{I}) \quad n = 1, \cdots, N \\ \begin{bmatrix} \mathbf{y}_n{}^{(1)} \mid \mathbf{z}_n \\ \mathbf{y}_n{}^{(2)} \mid \mathbf{z}_n \\ \vdots \\ \mathbf{y}_n{}^{(G)} \mid \mathbf{z}_n \end{bmatrix} \sim \mathcal{N}_P \left(\begin{bmatrix} \mathbf{W}^{(1)} \\ \mathbf{W}^{(2)} \\ \vdots \\ \mathbf{W}^{(G)} \end{bmatrix} \mathbf{z}_n, \begin{bmatrix} \sigma_1^2\mathbf{I} & 0 & \dots & 0 \\ 0 & \sigma_2^2\mathbf{I} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \dots & 0 & \sigma_G^2\mathbf{I} \end{bmatrix} \right) \end{gathered} \tag{2.38}$$

where

$$\mathbf{W} = \begin{bmatrix} \mathbf{W}^{(1)} \\ \mathbf{W}^{(2)} \\ \vdots \\ \mathbf{W}^{(G)} \end{bmatrix} = \begin{bmatrix} \mathbf{w}_1^{(1)} & \mathbf{w}_2^{(1)} & \cdots & \mathbf{w}_D^{(1)} \\ \mathbf{w}_1^{(2)} & \mathbf{w}_2^{(2)} & \cdots & \mathbf{w}_D^{(2)} \\ \vdots & \vdots & \vdots & \vdots \\ \mathbf{w}_1^{(G)} & \mathbf{w}_2^{(G)} & \cdots & \mathbf{w}_D^{(G)} \end{bmatrix} \tag{2.39}$$

There are two differences however. First, GFA restricts the group specific error covariance to isotropic structure. That is, $\Sigma^{(g)} = \sigma_g^2 \mathbf{I}$ for $g = 1, \cdots, G$. Second, in order to model the dependency structure between any subsets of groups and resolve indeterminacy issues, GFA imposes a structural sparsity prior over the columns of the projection matrix $\mathbf{W}$. The Automatic Relevance Determination (ARD) prior (MacKay, 1995b) over the $G \times D$ group specific columns of the projection matrix $\mathbf{W}$ can be specified as follows.

$$\begin{aligned} \mathbf{w}_d^{(g)} &\sim \mathcal{N}\left(0, (\alpha_d^{(g)})^{-1} \mathbf{I}\right) \quad g = 1, \cdots, G; \quad d = 1, \cdots, D \\ log(\alpha) &= UV^\intercal + \mu_u \mathbf{1}^\intercal + \mathbf{1}\mu_v^\intercal \\ p(U) &= \prod_g^G \prod_r^R \mathcal{N}(u_{g,r} | 0, (\lambda = 0.1)^{-1}) \\ p(V) &= \prod_d^D \prod_r^R \mathcal{N}(v_{d,r} | 0, (\lambda = 0.1)^{-1}) \end{aligned} \tag{2.40}$$

for $\alpha \in \mathbb{R}^{G \times D}$, $U \in \mathbb{R}^{G \times R}$, and $V \in \mathbb{R}^{D \times R}$ and $R \ll min(G, D)$.

2.5.9 Structural Equation Models (SEM)

The preceding models can be extended by allowing the latent variables to regress on each other. The resulting general model is known as a Structural Equation Model (SEM) and was first introduced by Jöreskog (1970). SEM models are constructed using algebraic equations, not by formulating a probabilistic statistical model as is done in Bayesian Graphical Models (Pearl, 2014) for example.

A popular general formulation of SEMs is given by the LISREL model. The LISREL model consists of two parts. The first part is a structural component in which dependencies

among the latent variables are created by regressing them on each other. The vector of latent variables we introduce into the model is divided into two groups of independent and dependent variable $\mathbf{z}_n = (\mathbf{z}_{n,1}^\intercal, \mathbf{z}_{n,2}^\intercal)$ of dimensions D_1 and D_2, respectively, such that $D_1 + D_2 = D$. The second part consists of a measurement model that is similar in form to the factor analysis model (i.e. $\mathbf{y}_n = \mathbf{W}\mathbf{z}_n + \varepsilon_n$) but for which we also partition the P-dimensional observation vector (and correspondingly the error vector) into two groups $\mathbf{y}_n = (\mathbf{y}_{n,1}^\intercal, \mathbf{y}_{n,2}^\intercal)$ and $\varepsilon_n = (\varepsilon_{n,1}^\intercal, \varepsilon_{n,2}^\intercal)$ such that $\mathbf{y}_{n,1}$ depends only on $\mathbf{z}_{n,2}$ and $\mathbf{y}_{n,2}$ depends only on $\mathbf{z}_{n,1}$. The dimension of two groups is P_1 and P_2, such that $P_1 + P_2 = P$. The general algebraic form of the LISREL model is as follows.

$$\begin{aligned} \begin{bmatrix} \mathbf{y}_{n,1} \\ \mathbf{y}_{n,2} \end{bmatrix} &= \begin{bmatrix} \mathbf{W}_1 & 0 \\ 0 & \mathbf{W}_2 \end{bmatrix} \begin{bmatrix} \mathbf{z}_{n,1} \\ \mathbf{z}_{n,2} \end{bmatrix} + \begin{bmatrix} \varepsilon_{n,1} \\ \varepsilon_{n,2} \end{bmatrix} \\ \mathbf{B}\mathbf{z}_{n,2} &= \mathbf{C}\mathbf{z}_{n,1} + \xi_n \end{aligned} \tag{2.41}$$

where it is assumed that B is non-singular and

$$\begin{aligned} &\mathbb{E}(\mathbf{y}_n) = \mathbf{0}; \mathbb{E}(\mathbf{z}_n) = \mathbf{0} \;\; \mathbb{E}(\varepsilon_n) = \mathbf{0};\; \mathbb{E}(\xi_n) = \mathbf{0} \\ &\mathrm{Cov}(\varepsilon_n) = \Psi;\; \mathrm{Cov}(\mathbf{z}_{n,1}) = \Phi_{z_1};\; \mathrm{Cov}(\xi_n) = \Phi_\xi \end{aligned} \tag{2.42}$$

and the covariance matrices are diagonal. The regressions imply the following covariance matrix $\Sigma_{\mathbf{yy}}$ for the observations.

$$\begin{bmatrix} \mathbf{W}_1\Phi_{z_1}\mathbf{W}_1^\intercal + \Psi_1 & \mathbf{W}_1\Phi_{z_1}\mathbf{C}^\intercal(\mathbf{B}^\intercal)^{-1}\mathbf{W}_2^\intercal \\ \mathbf{W}_2\Phi_{z_1}\mathbf{C}^\intercal(\mathbf{B}^\intercal)^{-1}\mathbf{W}_1^\intercal & \mathbf{W}_2(\mathbf{B}^{-1}\mathbf{C}^\intercal\Phi_{z_1}\mathbf{C}(\mathbf{B}^\intercal)^{-1} + \mathbf{B}^{-1}\Phi_\xi(\mathbf{B}^\intercal)^{-1})\mathbf{W}_2^\intercal + \Psi_2 \end{bmatrix} \tag{2.43}$$

Notice that how the implied covariance structure of the data is determined by the coefficient matrices (i.e. $\mathbf{B}$, $\mathbf{C}$, and $\mathbf{W}$) and the covariance matrices (i.e. Ψ, Φ_{z_1} and Φ_ξ). More importantly, notice also that an SEM model implicitly assumes that the relationships between the observations are linear since covariances are measures of linear association. Another aspect of SEM models that is worth pointing out is that although we can assess

whether a particular implied decomposition is consistent with the data, we cannot determine whether it is the only decomposition that is equally consistent with the data. That is, the same data can be generated given different latent structural relations (Bartholomew et al., 2011).

Generalized Structured Component Analysis (GSCA) A generalization of the SEM framework has recently been proposed by Hwang and Takane (2004). The goal of the generalization is to unify two distinct approaches to SEM; namely the factor-based and component-based (i.e. partial least squares). The Generalized Structured Component Analysis (GSCA) model they propose has the following structural formulation.

$$\begin{bmatrix} \mathbf{I} \\ \mathbf{W} \end{bmatrix} \mathbf{y}_n = \begin{bmatrix} \mathbf{C} \\ \mathbf{B} \end{bmatrix} \mathbf{W}\mathbf{y}_n + \begin{bmatrix} \varepsilon_n \\ \xi_n \end{bmatrix} \tag{2.44}$$

where the latent variables are obtained as component scores using what the authors refer to as weighted relation model $\mathbf{z}_n = \mathbf{W}\mathbf{y}_n$. The assumption regarding the means and covariance of the variables are identical to the LISERL model. It is interesting to note that the weighted relation model equation is no different conceptually than the encoding transformation that Izenman (2008) used to construct latent variable models from the reduced rank regression framework (Section 2.5.1).

We argue that by relying on complicated algebraic equations to induce linear dependencies in the latent structure, the SEM framework risks losing statistical rigor especially when modeling complex phenomena. Real data, most often than not, is noisy and characterized by complex nonlinear dependencies. We think that we have a better and more statistically rigorous alternative in the probabilistic generative perspective that approaches the modeling endeavor by first proposing a probabilistic model for all the observed variables and any latent variables that we wish to incorporate in the model. As part of the

probabilistic model, we can also specify functional relations between the variables in order to capture any nonlinear dependencies in the data or to construct a rich latent space with complex structure. For example, a recent method, the Inverse Autoregressive Flow (IAF)(D. P. Kingma, Salimans, & Welling, 2016), has been proposed in the generative modeling literature that enables us to create models with rich latent structure. The key idea behind the IAF is that any multivariate normal variable with diagonal covariance can be transformed into a multivariate normal with linear dependencies (i.e. full covariance matrix) by expressing it as an autoregressive model

$$z_d = \mu_d(\mathbf{z}_{<d}) + \sigma_d(\mathbf{z}_{<d}) * \varepsilon \tag{2.45}$$

where μ_d and σ_d are the conditional mean vector and covariance matrix of z_d on the other elements of the latent vector $\mathbf{z}_n = (z_1, z_2, \cdots, z_d, \cdots, z_D)$. Accordingly, a general SEM model can be conceptually viewed as the following generative model.

$$\begin{aligned} \mathbf{y}_n \mid \mathbf{z}_n &\sim \mathcal{N}_p\Big(\mathbf{W}\mathbf{z}_n, \text{Diag}(\sigma^2)\Big) \\ \mathbf{z}_n &\sim \mathcal{N}_r(0, \Sigma_z) \end{aligned} \tag{2.46}$$

By including structural latent regressions, a specific SEM model would then correspond to a particular structure of the covariance matrix Σ_z. In the next section, we briefly describe Deep Latent Gaussian Models (DLGMs) as an alternative general framework to SEM, one that is more statistically disciplined and potentially much more powerful, especially when large amounts of data is available.

2.5.10 Deep Latent Gaussian Models (DLGM)

A Deep Latent Gaussian Model (DLGM) (Rezende et al., 2014) is a directed graphical model with Gaussian latent variables that are stacked in L stochastic layers and that are transformed at each layer by a differentiable nonlinear function parameterized by a neural

network $\mathbf{NN}(\mathbf{z};\theta)$ consisting of $K^{(l)}$ deterministic hidden layers. Whereas the latent variables in the first DLGMs such as the VAE (D. P. Kingma & Welling, 2013) are assumed to be independent, recent extensions (D. P. Kingma et al., 2016; Maaløe, Sønderby, Sønderby, & Winther, 2016) have introduced a deep latent hierarchy that endows the latent space with complex rich structure that can capture complex patterns of dependencies between the latent variables. A general DLGM model has the following generative formulation.

$$\begin{aligned}
\mathbf{z}_n^{(L)} &\sim \mathcal{N}_{d_{(L)}}(0,\mathbf{I}) \quad n=1,\cdots,N \\
\mathbf{z}_n^{(l)} &\sim \mathcal{N}_{d_{(l)}}\Big(\mathbf{z}_n^{(l)} \mid \mathbf{NN}^{(l)}(\mathbf{z}_n^{(l+1)};\theta^{(l)}),\Sigma^{(l)}\Big) \quad l=1,\cdots,L-1 \\
\mathbf{y}_n \mid \mathbf{z}_n &\sim \mathbf{Expon}\Big(\mathbf{y}_n \mid \mathbf{NN}^{(0)}(\mathbf{z}_n^{(1)};\theta^{(0)})\Big)
\end{aligned} \tag{2.47}$$

where

$$\begin{aligned}
\mathbf{NN}(\mathbf{z};\theta) &= h_K \circ h_{K-1} \circ \ldots \circ h_0(\mathbf{z}) \\
h_k(\mathbf{y}) &= \sigma_k(\mathbf{W}^{(k)}\mathbf{y} + \mathbf{b}^{(k)}) \\
\theta &= \{(\mathbf{W}^{(k)},\mathbf{b}^{(k)})\}_{k=0}^{K}
\end{aligned} \tag{2.48}$$

Such that the observation model **Expon** refers to the exponential family of models and the neural network $\mathbf{NN}(\mathbf{z};\theta)$ consists of the composition of K non-linear transformation functions, each of which is represented by a hidden layer $h_k(\mathbf{y})$. The function σ_k is an element-wise *activation function* such as the sigmoid function or the rectified linear unit (ReLU). The constant $d_{(l)}$ is the number of latent variables at the l layer so that the total number of latent variables equals $d = \sum_{l=1}^{L} d_{(l)}$

A Deep Latent Gaussian Model is considered a deep model in two distinct senses. First, the latent variables that make up the stochastic layers can be stacked in a deep hierarchy. Second, the neural networks that make up the deterministic nonlinear transformations can be made up of several hidden layers. In contrast, a nonlinear regression model is a *shallow*

nonlinear model in the sense that it uses only one non-linear transformation while a factor analysis model is a *shallow linear* model in the sense that the latent variables $\mathbf{z}_n$ are linearly projected by a matrix $\mathbf{W}$ to a higher dimensional vector.

Chapter 3: Unsupervised Learning of Chromatin States

3.1 Application Domain: Functional Epigenomics

This chapter investigates the application of a *deep generative latent model* to a set of 100 [1] whole-genome high-throughput functional epigenomics data sets with the goal of learning, from a set of nine epigenomic marks for each of ten ENCODE cell types, patterns of chromatin state variability that is indicative of the functional activity of regulatory DNA elements (e.g. enhancers, promoters) both within and across cell-types.

3.1.1 Introduction and Background

An individual's *phenotype* consists of all the observed characteristics that an individual manifests, ranging from biochemical properties at the subcelluar level to behavioral and cognitive traits at the organism's level. The phenotype is determined by an individual's *genotype* (i.e. the hereditary information encoded in the DNA) and non-hereditary environmental factors that regulates an organism's *phenotypic plasticity*. A concise way to mathematically conceptualize the causal relationship between the genotype and the phenotype can be given by the following genotype–phenotype map (Omholt, 2013).

$$\textbf{Organismal Phenotype} = \Phi(\textbf{Genome}, \textbf{Environment}) \tag{3.1}$$

[1] For each cell type there is a control input data set (ten in total).

As stated in Equation 3.1, the mapping between an entire genome and a corresponding phenotype is a computational problem of overwhelming complexity even in the case of a single cell. Predicting and understanding the effect of any single base-pair perturbation along the genome on an organismal phenotype is an overarching goal that is driving a great deal of contemporary research. Although we are still far from reaching our objective, recent technological advances in targeted genome editing using CRISPR/Cas9 has ushered a new era in biological research and has given us great hope in making progress toward achieving that end goal. Of particular note are several recent studies that have successfully used CRISPR/Cas9 mutagenesis screens to experimentally determine the effects of sequence perturbation in neighboring non-coding regions on the expression of target genes (Canver et al., 2015; Chen et al., 2015; Korkmaz et al., 2016; Fulco et al., 2016; Sanjana et al., 2016). Although these studies are not high-throughput, they are becoming the sine qua non experimental approach to testing whether a non-coding region is functional and to assessing its activity in an endogenous *in vivo* environment.

Given that the genome affects the phenotype by orchestrating cell activity through many dynamically interconnected biophysical processes, a more feasible approach that is often taken divides the general problem into sub-problems and restricts the initial focus to the level of a particular molecular phenotype, rather than to level of the organismal phenotype. More specifically, by limiting the output to a single molecular variable such as gene-expression, the genotype–phenotype mapping problem can be restated as follows.

$$\textbf{Gene Expression} = \Phi(\textbf{Genome}) \tag{3.2}$$

It is important to point out that the genome is no longer thought of as a static entity (Lanctôt, Cheutin, Cremer, Cavalli, & Cremer, 2007). We currently know that most regions in the genome are dormant for any given particular cell type or state, but the few specific regions

that are active essentially determine the identity and function of the cell. The realization that the genome is dynamic became apparent soon after the Human Genome Project (HGP) (Lander et al., 2001) was completed in 2003, achieving its stated goal of sequencing all 3 billion base-pairs of the human genome. After the protein coding genes were annotated, it was discovered that the coding regions of the genome made up only 1.5% of the human genome. The growing understanding that the genome is not static has given rise to an intensive, collaborative research projects in the field of functional genomics with a focus on understanding the dynamic aspects of the genome, particularly with regards to determining how the non-coding regions of the DNA sequence are involved in gene regulation. As a follow up of the HGP, the Encyclopedia of DNA Elements (ENCODE) project (Consortium, 2012) was launched soon afterwards with the goal of determining the functional role of the remaining 98.5% of what was initially thought of as "junk DNA". In particular, one of the main aims of the project is to identify which parts of the non-coding sequence are *non-regulatory regions* and which are *regulatory non-coding regions*. Even more recently, the PsychENCODE project (Akbarian et al., 2015), one of the largest collaborative efforts in neuroscience and psychiatry, was launched with the aim of understanding how variation in functional DNA elements and gene expression patterns in the central nervous system give rise to psychiatric disorders such as bipolar disorder, autism, and schizophrenia. In addition, multiple studies have already begun to shed light on how changes in the neuronal functional genome are associated both with learning and plasticity, as well as with normal and abnormal cognition. See Rajarajan, Espeso Gil, Brennand, and Akbarian (2016) for a recent review.

3.1.2 General Problem

The mechanism governing the regulation of gene expression in animals, including humans, is a complex unsolved problem of great importance. The difficulty of the problem partly stems from our inability to read the genome. Although we can currently sequence an entire human genome in less than a day, we are still unable to ascribe semantic meaning to all of the various regions that make up the sequence itself. Admittedly, we have made great progress in annotating the small part of the genome we presently know, including not only the 19,817 protein coding genes that produce the 80,531 protein-coding transcripts, but also 15,787 long non-coding RNA genes and 7,568 small non-coding RNA genes that produce non-coding transcripts (GENCODE annotation, V26). In contrast to the protein-coding genome which we know much about, the non-coding genome is still far from fully understood. Even for the majority of the non-coding genes we have identified, we have not ascribed a function to them yet.

Mapping and explaining the patterns of gene expression activity for different cell-types is fundamental to understanding how the genotype gives rise to the phenotype (Gjuvsland, Vik, Beard, Hunter, & Omholt, 2013). The human body, for example, is made up of approximately 400 somatic cell types, each of which contains the same exact DNA-sequence, yet each cell is different in form and function (i.e. its gene expression profile). The proximal reason for this apparent incongruity is that only a small subset of the approximately 20,000 protein coding genes are turned on in any cell type. The ultimate cause, however, of why a given subset of genes get expressed and not another is due to the epigenetic code. Whereas the genetic code is identical in all cells of a single individual, the epigenetic code is specific to each cell type and tissue (Turner, 2007). That is, although every cell contains information to make 20,000 proteins (i.e. ingredients), each cell-type at a particular point

in time uses a specific recipe, an *epigenetic code*, to produce and combine the protein building blocks in precise proportions, giving rise to a spatio-temporal gene expression profile pattern that defines the observed phenotype.

The main player mediating epigenetic regulatory control mechanisms is *chromatin*, defined as a DNA-protein complex consisting of DNA wrapped around proteins called *histones* that function to package and compact DNA into denser structures, *chromosomes*. In particular, it is the cell-specific three-dimensional structure of chromatin that serves as the conduit of gene regulation. More specifically, gene expression can be thought of as mainly controlled by the binding of roughly 2,000 transcription factor (TF) proteins to millions of regions on DNA known as transcription factor binding sites (TFBS) (Latchman, 2015). A single TF can regulate multiple genes by binding to multiple TFBS and any particular gene can be regulated by several TFs. A region of noncoding DNA where transcription factor binding sites cluster together is called a *cis-regulatory element*. Cis-regulatory elements can be sub-classified into promoters, enhancers, silencers, locus control regions, or insulators depending on their relative distances from the transcription start site (TSS), their repressive or activating effect on modulating gene expression, and the three-dimensional structure of chromatin. When the genome is packed into closed chromatin (its default state), no transcriptional activity occurs since TFBSs are not accessible. For chromatin to open, repressive chromatin marks such as methylated histones and methylated cytosines (in CpG islands sequences) need to be modified by enzymes and replaced by active chromatin marks. These post-translational modifications are termed epigenetic since they can regulate gene expression without modifying the genetic code.

3.1.3 Challenges

Based on the details provided in the previous section, the genotype–phenotype mapping problem which was introduced in Equation 3.2, can after linearizing the function Φ and decomposing it into two functions Φ_1 and Φ_2, be restated as follows.

$$\begin{aligned} \textbf{TF binding} &= \Phi_1(\textbf{Regulatory noncoding DNA}) \\ \textbf{Gene Expression} &= \Phi_2(\textbf{TF binding}) \end{aligned} \tag{3.3}$$

The first obstacle to solving this much simplified problem is given by the input argument to the first function Φ_1. The difficulty is due to the fact that the regions of the DNA sequence that make up the regulatory noncoding DNA are far from fully identified and annotated. As of yet, we do not have a complete and reliable list of identified regulatory elements. The mapping of the regulatory elements specific to each cell type and condition is the first step to understanding how these regulatory regions are responsible for determining cell identity, orchestrating cellular differentiation, and how their dysregulation causes abnormal gene expression leading to disease.

3.1.4 Proposed Approach

Out of the main cis-regulatory elements mentioned in Section 3.1.2, enhancers play a central role in modulating gene expression in a cell type-specific fashion. Indeed, enhancers, by virtue of functioning as the signaling switchboard that brings about changes in transcriptional gene regulation in response to external cues, are key in understanding how the genotype gives rise to the phenotype. Enhancers are the most variable class of regulatory elements across cell-types and are the most dynamically activated part of the genome (Heintzman et al., 2009). Although most of the estimated one million enhancers are cell

type specific, some enhancers can be active in multiple cell-types (Consortium, 2012). In particular, a regulatory region that is deplete of nucleosomes, is bound by an activating TF, and can upregulate the a target gene's transcriptional profile in a specific cell type is termed an *active enhancer* (Shlyueva, Stampfel, & Stark, 2014).

Recently developed experimental approaches such as STARR-seq (Arnold et al., 2013) and CapStarr-seq (Vanhille et al., 2015) have enabled the genome-wide identification of enhancers activity. Although promising, these assays have a few limitations. First, they are not context specific since they do not measure enhancer activity in the endogenous chromatin environment. Second, unless sequencing is performed at very high coverage depths [2], they are generally subject to high noise. As a result of the limitations plaguing existing experimental approaches and the dramatic increase in publicly available "omics" data sets, several researchers (Yip, Cheng, & Gerstein, 2013; Leung, Delong, Alipanahi, & Frey, 2016) have called for the adoption of a different strategy for identifying functional genomic elements. The strategy they propose is focused on two goals. First, to leverage data-driven machine learning approaches to identify patterns in large data sets and second, to generate computational predictions that can then be experimentally validated in the lab.

Given that context-specific signals determine whether an enhancer is active, current computational approaches have increasingly been incorporating into the modeling framework, not only static *genomic features* such as transcription factor binding sites and the DNA sequence itself but also *epigenomic features* such as nucleosome positioning, histone modifications, and chromatin accessibility. Many of these computational approaches differ mainly in two respects. First, what genomic or epigenomic features to include in the model and second, whether to use supervised or unsupervised learning models. Notwithstanding,

[2] Although CapStarr-seq attempts to mitigate this particular shortcoming

what unites the majority of these approaches is the overarching goal of determining the identity and interactions of cell-specific regulatory regions from high-throughput molecular features such as chromatin accessibility, histone modifications, and transcription factor binding sites using statistical and machine learning models trained on large data sets.

Our proposed computational approach can be classified as an unsupervised learning model that uses epigenomic features toward the goal of identifying, not just enhancers, but also other regulatory regions of the genome such as promoters and CpG islands. More specifically, the unsupervised approach we propose consists of training a deep generative model to learn a continuous representation of chromatin states across the genome using a set of nine epigenomic features (Table 3.2); namely eight histone modifications marks and one transcription factor mark that provides indirect information about the 3-D structure of chromatin. After the model is learned, we assign genomic regions that have existing annotation with corresponding labels. In particular, given that on of the the most accurate features for predicting enhancers are enhancer RNA (eRNA) is considered (Li, Notani, & Rosenfeld, 2016), we use the list of enhancers taken from FANTOM5 CAGE expression atlas to annotate a subset of genomic regions as enhancers (Andersson et al., 2014). Enhancer RNAs (eRNAs) are defined as active enhancers that produce RNAs when they are bound by general transcription factors and RNA polymerase II (Natoli & Andrau, 2012).

3.1.5 Related Approaches

Our approach is thus rather similar in spirit to two earlier and well-established unsupervised generative models for learning chromatin states, *ChromHMM* (Ernst & Kellis, 2012) and *Segway* (Hoffman, Buske, et al., 2012). Although at first glance it might seem that the goal of our proposed approach is similar to that of *ChromHMM* and *Segway*; namely to

develop unsupervised computation methods to segment the genome into *chromatin states*, the generative model proposed here is more focused on uncovering the underlying generative mechanism that gives rise to the observed patterns of dependencies in the data, both across cell-types and epigenomic marks. A second goal of the approach is to obtain a latent manifold representation which we can later use to functionally annotate unlabeled genomic loci, especially those corresponding to enhancers.

Although the model we specify is, like *ChromHMM* and *Segway*, a generative latent variable model, there are some important differences. Unlike the Hidden Markov Model and the Dynamic Bayesian Network model that underlie *ChromHMM* and *Segway*, respectively, both of which are shallow models with sequentially dependent discrete latent variables, the generative model proposed here is a deep Gaussian latent variable model in which the latent variable are modeled as continuous and independent. Furthermore, unlike *ChromHMM* but similar to *Segway*, real valued signal data are used as input, not binarized data.

It is important to point out that both *ChromHMM* and *Segway* suffer from serious shortcomings. In particular, since *ChromHMM* models the input observations using an independent Bernoulli model conditional on the hidden state, the need to binarize the data to fit the model arguably may result in a substantial loss in information due to discretization and arbitrary thresholding. In contrast, *Segway* models the data as independent Gaussian conditional on the hidden state and consequently requires the data to be transformed to meet the normality assumption. Nonetheless, the monotonic transformation is unjustified since it fails to transform the variables into true Gaussian. Furthermore, for both *ChromHMM* and *Segway*, the conditional assumption that defines Hidden Markov Models is motivated by mathematical tractability, not biological reality. The patterns of dependencies in reality

are likely to be much more complex, bi-directional, and long-distance acting than what the Markov assumption would grant.

Building on *ChromHMM*, other methods also have been recently proposed in the literature. For example, Spectacle (Song & Chen, 2015) specifies a similar Hidden Markov Model as *ChromHMM*. However, it uses a spectral learning approach instead of the EM algorithm for inference. The Self-Organizing Map (SOM) model adopted by Mortazavi et al. (2013) is another approach that provides a higher resolution analysis of chromatin states allowing the visualization of more than 1,000 states on a two-dimensional map in which similar chromatin states cluster together. Although these two approaches can be considered unsupervised learning models, they are not generative probabilistic models in the same way *ChromHMM* and *Segway* are.

3.2 Data

A total of 100 ENCODE epigenomic data sets are used in this study. For each of ten ENCODE cell types (GM12878, H1-hESC, HMEC, HSMM, HUVEC, K562, NHEK, NHLF, HeLa-S3, and HepG2), genome-wide bigWig signal coverage tracks for ten ChIP-seq DNA binding assays are used: eight histone modifications tracks labeled as H3K4me1, H3K4me2, H3K4me3, H3K9ac, H3K27ac, H3K27me3, H3K36me3, and H4K20me1; one transcription factor track (CTCF); and one control signal track. ChIP-seq, which stands for Chromatin ImmunoPrecipitation followed by high-throughput sequencing, is a method that allows us to determine the regions in the genome in which selected targeted proteins most often bind to DNA (Johnson, Mortazavi, Myers, & Wold, 2007).

The 100 bigWig tracks were made publicly available as part of the ENCODE consortium Jan 2011 data release. The public data is provided in a processed form generated

using the WIGGLER software (Kundaje, n.d.). In particular, the generated signal tracks were normalized for differences in the number of mapped reads, number of replicates, fragment lengths, and local mappability (Hoffman, Ernst, et al., 2012). The data is available at `https://sites.google.com/site/anshulkundaje/projects/wiggler`. The exact names and repository URL addresses of the downloaded files are provided in Appendix B.1. The total size of all 100 compressed binary bigWig files is approximately 84GB. UCSC Genome Browser (Kent et al., 2002) screen-shots of a sample of the signal tracks for a small region of the genome are depicted in Figure 3.1 and Figure 3.2.

Additionally, existing functional annotations for enhancers, CpG islands, and GENCODE Genes were obtained and used as labels to validate the proposed model after it has been fit. It should be noted that functional labels are available for only a small fraction of the approximately 14 million observations eventually used to train the model. Sections 3.2.3 and 3.3.2 provide more detail about how these observations and annotations are obtained.

3.2.1 Description of ENCODE Cell Types

The ten ENCODE cell types are listed and described in Table 3.1. Only three out of the ten are cancer cell lines; i.e., K562, HeLa-S3, and HepG2. The other seven cell types come from normal body tissues. These particular ten cell types have been selected because they were previously used in other studies that proposed unsupervised method to annotate chromatin states. For example, the same ten cell types were used by Song and Chen (2015) and except for HeLa-S3, the other nine cell types are the same ones used in the study by Ernst et al. (2011).

Table 3.1: ENCODE Tissue/Cell Types

Tissue/Cell Type	Description
K562	Erythrocytic leukemia
GM12878	B-lymphocyte cell
H1-hESC	Embryonic stem cell
HeLa-S3	Cervix adenocarcinoma
HepG2	Hepatocellular carcinoma
HUVEC	Umbilical vein endothelial cell
HSMM	Skeletal muscle myoblasts
NHEK	Epidermal keratinocytes
HMEC	Mammary epithelial cell
NHLF	Lung fibroblasts

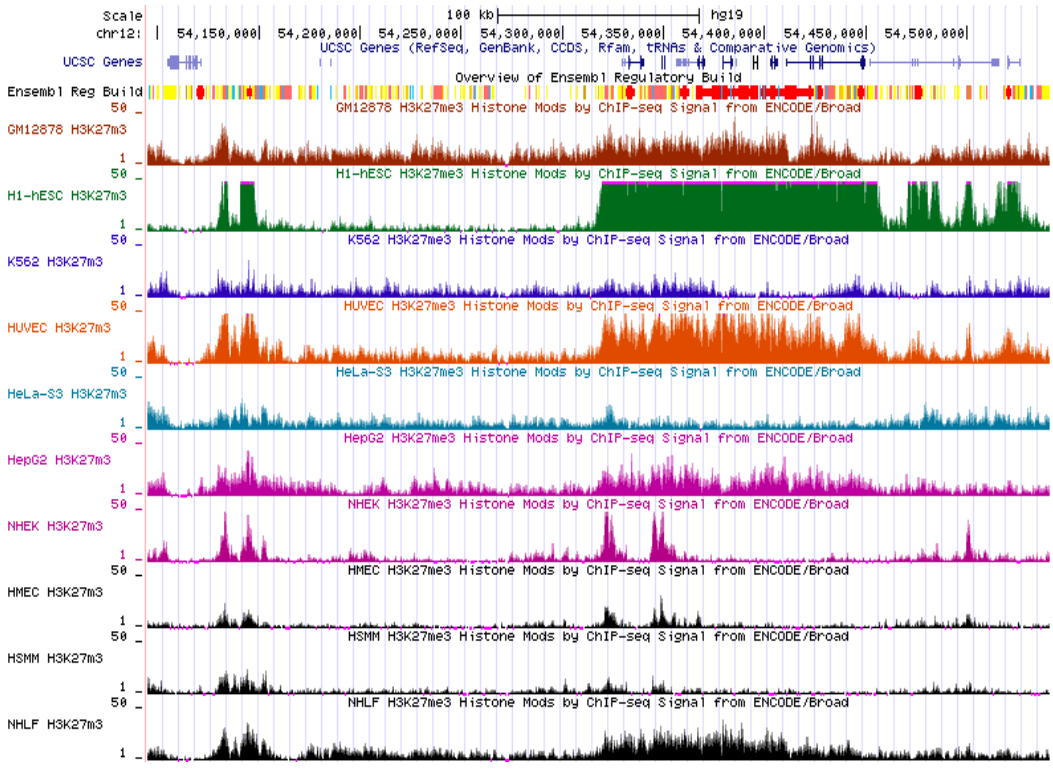


Figure 3.1: UCSC genome browser tracks for the H3k27me3 mark for all ten cells. The tracks are available at `http://genome.ucsc.edu/`

3.2.2 Description of Epigenomic Marks

Table 3.2 lists the nine epigenomic marks used for the study. Note that since the tenth data set for each cell type is a control signal that is used to quantify the baseline noise, it is not included in the table. The first epigenomic mark is for a CTCF transcription factor binding protein. CTCF defines the boundaries between euchromatin (open DNA) and heterchromatin (closed DNA). It also regulates the 3D structure of chromatin by allowing the formation of chromatin loops (Ong & Corces, 2014). The eight marks that follow in the table are histone marks of post-translational modifications. Post-translational modifications of histones can impact gene expression by altering chromatin structure. In particular, the methylation and acetylation of lysine residues of histone proteins can determine whether chromatin is accessible to transcription factor binding and consequently whether genes are transcribed (Carlberg & Molnár, 2014). The third column in the table lists the functional region of the genome each epigenomic mark is associated with (Kundaje et al., 2015).

Table 3.2: List of Epigenomic Marks and Corresponding Function

Epigenomic Mark (Target Protein)	**Description** (Modification/Residue)	**Associated Region**
CTCF	Transcription Factor	Insulator
H3k27ac	Acetylation/Lys27	Active enhancer/promoter
H3k27me3	Tri-Methylation/Lys27	Polycomb repressed Regions
H3k36me3	Tri-Methylation/Lys36	Transcribed regions
H3k4me1	Mono-Methylation/Lys4	Poised enhancer
H3k4me2	Di-Methylation/Lys4	Poised/active enhancer
H3k4me3	Tri-Methylation/Lys4	Promoter regions
H3k9ac	Acetylation/Lys9	Active gene
H4k20me1	Mono-Methylation/Lys20	Active gene

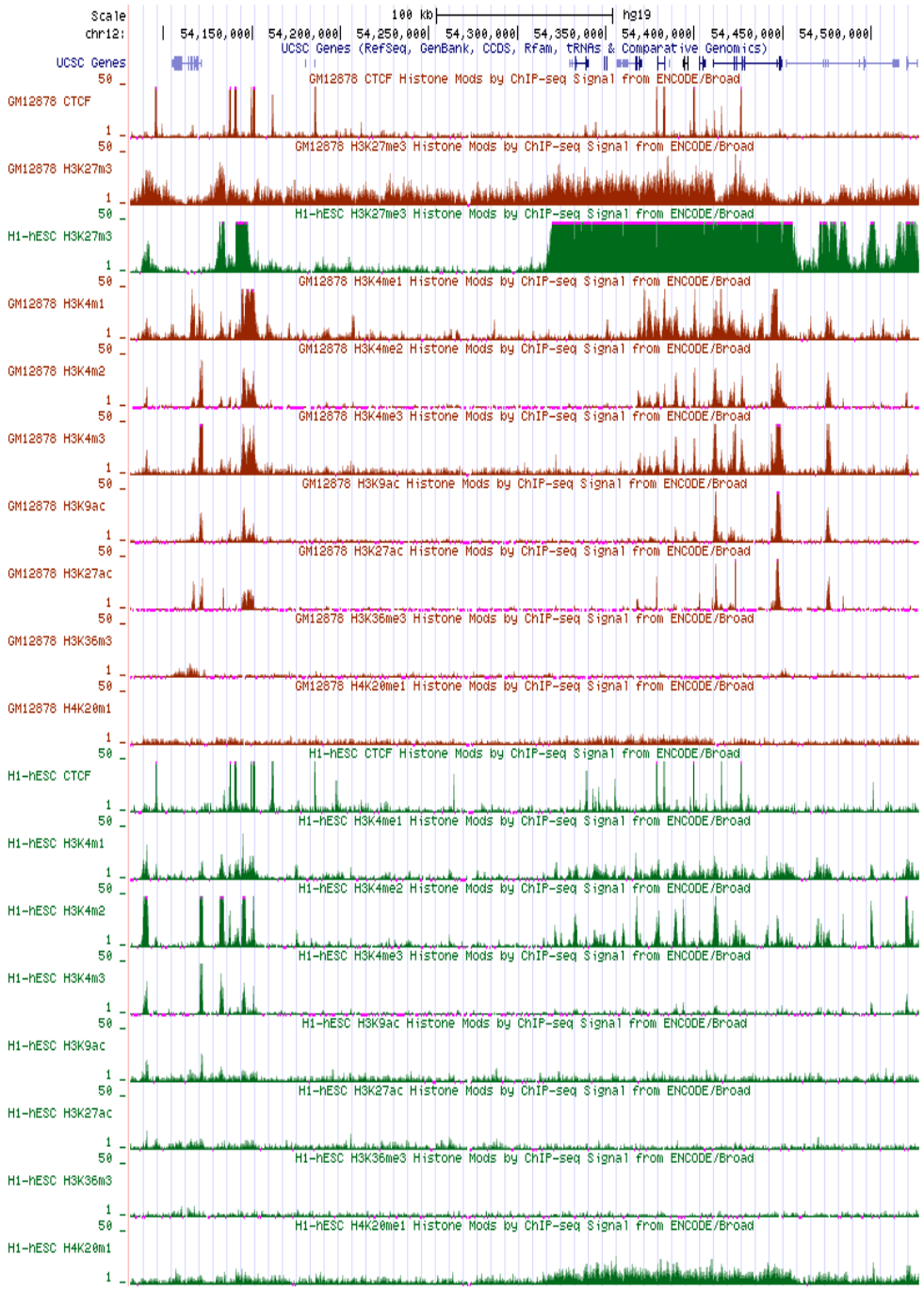


Figure 3.2: UCSC genome browser tracks for all nine marks for the GM12878 and H1-hESC cell types. The tracks are available at `http://genome.ucsc.edu/`

3.2.3 Functional Annotation Labels

The functional annotations that are used to label the observations are described next.

1. A list of 43,011 cataloged candidate enhancers *active* across a majority of human cell types and tissues obtained from the FANTOM5 CAGE expression atlas (Andersson et al., 2014).

2. A list of 52,502 CpG island regions from the unmasked CpG (cpgIslandExtUnmasked) Track. Although CpG islands are rare, there seems to be selective pressure to keep them due to their role in gene regulation (CpG islands are highly enriched for TF binding sites). The reason we should see fewer of them is that the Cs in CpG islands are normally methylated and over evolutionary time the Cs turn into Ts; i.e., spontaneous deamination (Gardiner-Garden & Frommer, 1987).

3. The GENCODE Basic Set Genes track (version 19, December 2013) (Harrow et al., 2012). A BED file containing a list of defined promoters was created from the GENCODE annotation track (wgEncodeGencodeBasicV19). Promoters are defined as genomic intervals that are 400 bps upstream and 100 bps downstream of a Transcription Start Site (TSS), the position in the sequence at which transcription of DNA into mRNA begins.

The BED file (permissive_enhancers.bed) containing a list of predicted enhancers was downloaded from the web address `http://enhancer.binf.ku.dk/presets/`. The GENCODE annotation and the CpG island files were downloaded using the UCSC Table Browser, available at `http://genome.ucsc.edu/cgi-bin/hgTables`.

3.3 Preprocessing

Data preprocessing consisted of the following five steps: (1) Creating a blacklist file from the merger of publicly available annotations of excludable genomic regions. (2) Segmenting the human reference genome HG19 into a set of 200bp resolution genomic intervals and discarding any region in the genome segmentation file that overlaps regions found in the blacklist file. (3) Combining the 100 bigWig signal files into a single data-frame and averaging the signal over the 200bp segments. (4) Normalizing the signals, linking labels (available functional annotation) to genomic regions, and storing the multiple data sets in a compressed data store divided by chromosome. (5) Finding regions that intersect with any of the regions defined in the functional annotation lists. These five steps are further described in the following sections.

3.3.1 Creating a Blacklist File

A blacklist of artifact regions of the human genome that have anomalous, unstructured signals was created by combining the following three annotation files.

1. ENCODE Data Analysis Consortium (DAC) Blacklisted Regions

2. Duke Excluded Regions

3. Gap locations (e.g. centromeres, telomeres, contigs, etc)

The merged blacklist file was created using the *GenomicRanges* package (Lawrence et al., 2013) in R (R Core Team, 2016). The R script in Appendix A.1.1 contains code that downloads the three files and merges them in one comprehensive blacklist of genomic intervals restricted to regions in the standard chromosomes minus the Y chromosome.

3.3.2 Segmenting the Human Reference Genome

The total sequence length of the human reference genome GRCh37 (Lander et al., 2001) is 3,137,144,693 base pairs (bp). To decrease the resolution of the signals, the entire length of the genome is divided into 15,181,531 bins of size 200 bp using the *window* function in *Bedtools* (Quinlan & Hall, 2010). The resulting file is filtered of all 200 bp segments that overlapped any of the blacklisted regions. The resulting filtered file contained 14,119,934 bins. For more details, please see the code in Appendix A.1.2.

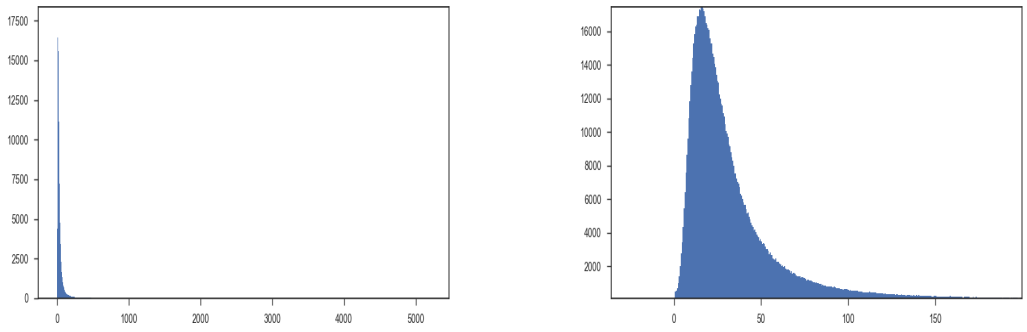
3.3.3 Combining Signal Tracks

The *multiBigwigSummary* function from *DeepTools2* (Ramírez et al., 2016) is used to combine the bigWig files into one data-frame with 100 columns corresponding to each mark/cell combination and 14,119,934 rows, each corresponding to a 200bp genomic segment. Each entry in the matrix is the computed average signal score for a mark/cell over a 200bp bin. For more details, please see the code in Appendix A.1.2.

3.3.4 Normalizing Signal Tracks

First, missing values are replaced by zeros. These missing values correspond to inaccessible regions of the genome that the ChIP-seq assay cannot survey due to technological limitations (i.e. regions with no signal). Second, the control signal for each cell type is subtracted from all other signal tracks of the same cell type. Any resulting negative values are thresholded to zero. The ten control signal are discarded afterwards and observations in which the sum of signal values across the 90 variables equals zero are omitted. As a result, the number of observations (i.e. bins) decreased to 13,367,943.

It is important to note that the marginal distribution of each signal track is highly variable across marks and cells. However, almost all are characterized by extremely long tails with some having maximum values in the hundreds or even thousands. The distribution of sums obtained from collapsing the observations over the columns is also characterized by very long tails. Figure 3.3 is a plot of the distribution of sums for Chromosome 1. Other chromosomes are characterized by a similar pattern. The distribution can be thought of as the univariate probability density obtained after projecting the joint probability distribution of the 90 variables onto $\mathbf{1}$, the vector of ones. The resulting one dimensional random variable can be denoted as $w = \mathbf{1}^{\mathsf{T}}\mathbf{y}$ where $\mathbf{y}$ is a 90 dimensional vector.



(a) Histogram of Summed Signal Observations **(b)** Restricted and Magnified View of Figure (a)

Figure 3.3: The distribution of 1,085,047 summed signal observations for Chromosome 1. The sum is taken over the 90 column variables (i.e. experiments)

In order to decrease the range of variability of the signals, the data is then transformed element-wise using the inverse hyperbolic sine function $asinh(y) = \log(y + \sqrt{y^2+1})$. The *asinh* transformation has the same effect as the log transformation for large values but in contrast has a weaker compressing effect for small values. Furthermore, unlike the log

transform, the *asinh* transform does not requires a pseudo-count to be added to the input. In order to restrict the range of possible values to the 0-1 interval, each variable is scaled by the maximum value over all the observations. However, since the signal for some tracks exhibit a few regions with extremely high values (in the thousands), all value exceeding a threshold of six in the transformed scale (200 in the original scale) are clipped to the threshold value. The scaling is necessary in order to use a Bernoulli model for the observations. Note that cross-experiment normalization was not attempted since any heuristic scaling solution might introduce unintended consequences.

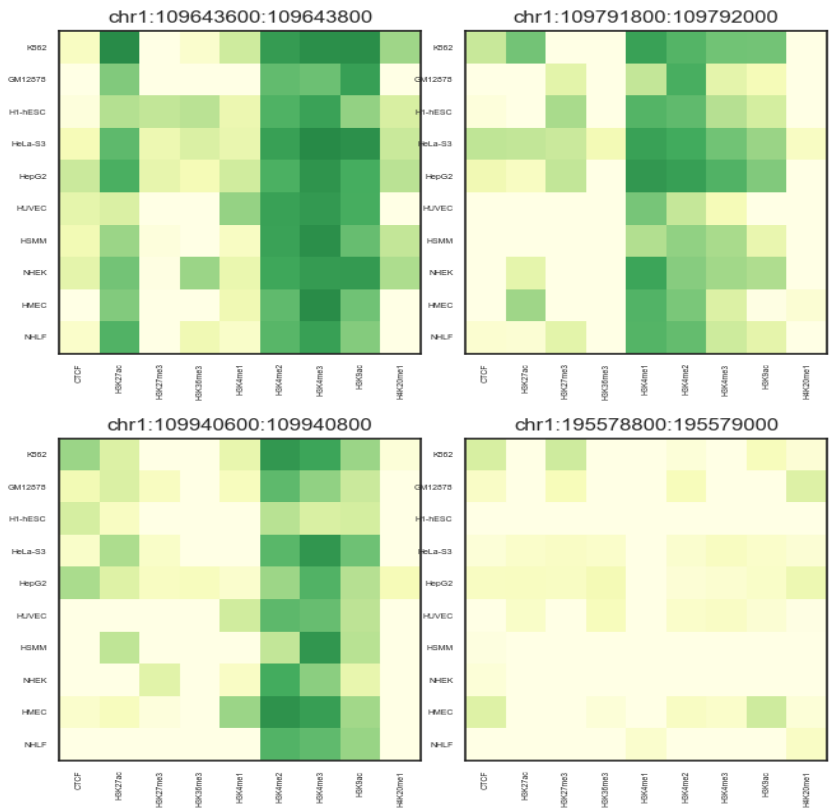


Figure 3.4: A sample of 4 observations from test data. Each observation has 10 rows (cells) and 9 columns (epigenomic marks)

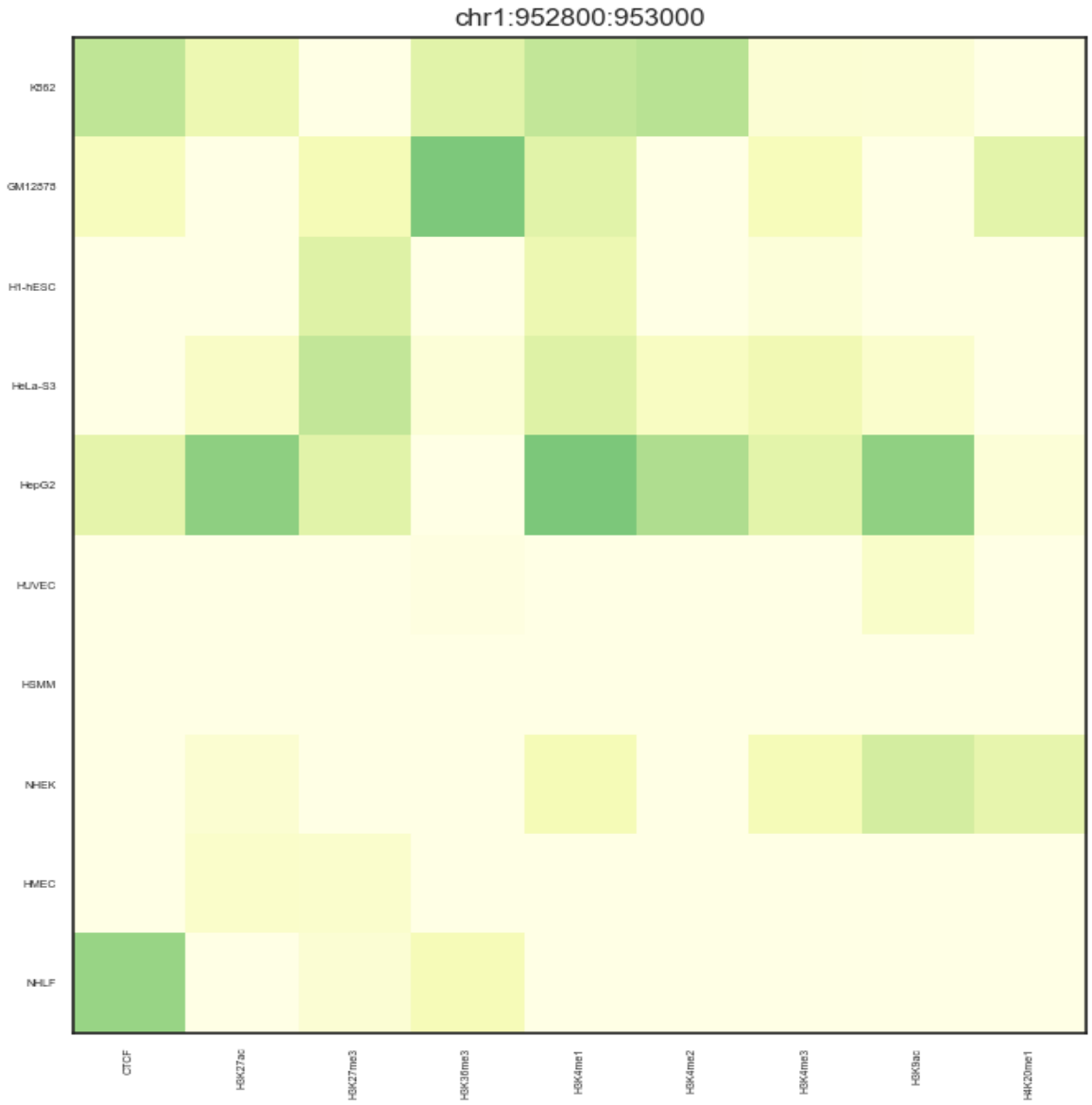


Figure 3.5: A single 90-dimensional observation reshaped as an image representing the scaled mean signal over a 200bps window for all 90 datasets. Zero corresponds to very light yellow and one corresponds to very dark green.

3.3.5 Creating Labels

The outcome of the pre-processing steps is a matrix containing 14,119,934 observations stacked in rows and where each row is a 90 dimensional normalized vector whose elements values range from 0 to 1. A sample of four observation vectors (i.e. rows) have been reshaped as two dimensional arrays and plotted as images (Figure 3.4).

The Python *pybedtools* package (Dale, Pedersen, & Quinlan, 2011) was used to determine which observations in the data intersect with any genomic region found in the functional annotation lists provided.

3.4 Model

The class of probabilistic models known as Deep Latent Gaussian Models (DLGMs), see Section 2.5.10 for more detail, have recently emerged as one of the most promising approaches to the automatic discovery of underlying feature representations in high-dimensional data sets and endowing computers the ability to learn complex data patterns in an unsupervised manner (Rezende et al., 2014). Unlike many common and successful unsupervised learning methods used for clustering and prediction developed in the machine learning and deep learning communities, a DLGM, by virtue of its formulation as generative probabilistic graphical model, allows us to accomplish a goal that tends to be of high importance in scientific applications; namely, to learn an interpretable model structure that can explain the observed high-dimensional data in terms of a meaningful lower-dimensional latent representation.

The specific model we describe and formulate for the data is known as the Variational Autoencoder (VAE) (D. P. Kingma & Welling, 2013). The VAE is a special case of a DLGM obtained by restricting the hierarchy of stochastic layers to just one layer composed

of independent Gaussian latent variables. But like DLGMs, a VAE consists of two components. The first is *generative model* that specifies how the latent variables give rise to the observations through a nonlinear mapping parametrized by a neural network. The second is an *inference model* that learns the inverse mapping from high-dimensional observations to lower dimensional latent variables. The inference model represents the approximate posterior distribution and uses a neural network to parameterize the nonlinear mapping.

3.4.1 The Variational Autoencoder (VAE)

Let $\mathbf{Y} \in \mathbb{R}^{N \times P}$ denote a data matrix consisting of N observation vectors $\mathbf{y}_n \in \mathbb{R}^P$; let $\mathbf{z}_n \in \mathbb{R}^D$ denote the latent variable vector corresponding to the nth observation; and Let $\theta \in \mathbb{R}^M$ denote the vector of model parameters. The generative model can be obtained by decomposing the joint probability distribution of the observed and unobserved variables as follows.

$$p(\mathbf{y}_1, \cdots, \mathbf{y}_N, \mathbf{z}_1, \cdots, \mathbf{z}_N; \theta) = \prod_{n=1}^{N} \mathbf{Expon}(\mathbf{y}_n \mid \mathbf{NN}(\mathbf{z}_n; \theta)) \mathcal{N}_d(\mathbf{z}_n \mid 0, \mathbf{I}) \tag{3.4}$$

where **Expon** refers to the observation model whose distribution belongs to the exponential family of models (usually either a Gaussian or Bernoulli) and $\mathbf{NN}(\mathbf{z}; \theta)$ denotes a deterministic nonlinear function parameterized by the neural network

$$\begin{aligned} \mathbf{NN}(\mathbf{z}; \theta) &= h_K \circ h_{K-1} \circ \ldots \circ h_0(\mathbf{z}) \\ h_k(\mathbf{y}) &= \sigma_k(\mathbf{W}^{(k)}\mathbf{y} + \mathbf{b}^{(k)}) \\ \theta &= \{(\mathbf{W}^{(k)}, \mathbf{b}^{(k)})\}_{k=0}^{K} \end{aligned} \tag{3.5}$$

The neural network consists of the function composition of K non-linear transformation σ_k, the output of which is represented by a hidden layer $h_k(\mathbf{y})$. The function σ_k is an element-wise *activation function* such as the Rectified Linear Unit (ReLU); the softplus, a smooth

approximation to RelU; or the sigmoid function (i.e. logistic), the derivative of the softplus. These three activation functions are plotted in Figure 3.6 and defined as follows.

- ReLU: $\sigma(x) = \max(0, x)$
- Softplus: $\sigma(x) = \ln(1 + e^x)$
- Sigmoid: $\sigma(x) = \frac{1}{1+e^{-x}}$

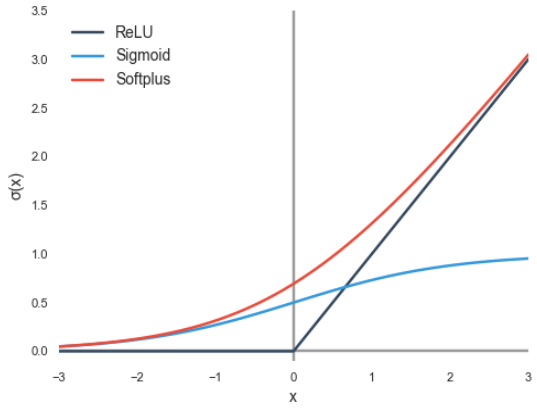


Figure 3.6: Common activation functions used in neural networks

For instance, the model for a Gaussian data distribution can be described by the following generative formulation:

$$\begin{aligned} \mathbf{z}_n &\sim \mathcal{N}_d(0, \mathbf{I})) \quad n = 1, \cdots, N \\ \mathbf{y}_n \mid \mathbf{z}_n &\sim \mathcal{N}_p(\mathbf{NN}_\mu(\mathbf{z}_n; \theta), \mathbf{NN}_\sigma(\mathbf{z}_n; \theta))) \end{aligned} \tag{3.6}$$

where both the p elements of the mean vector and p elements of the diagonal covariance matrix are the output of a deterministic nonlinear function parameterized by a deep neural network with two output layers.

We are mainly interested in learning a lower-dimensional latent representation of the data that captures the independent factors of variation that govern the observed data. This inferential task of explaining the data involves learning the posterior distribution of the latent variables; i.e. $p(\mathbf{z}_n \mid \mathbf{y}_n)$. By applying Bayes rule, we can obtain the posterior as such.

$$p(\mathbf{z}_n \mid \mathbf{y}_n) = \frac{p(\mathbf{y}_n \mid \mathbf{z}_n)p(\mathbf{z}_n)}{p(\mathbf{y}_n)} \tag{3.7}$$

Since both the marginal distribution $p(\mathbf{y}_n)$ and the data distribution $p(\mathbf{y}_n \mid \mathbf{z}_n)$ are intractable, the true posterior distribution is also intractable. To solve this intractable inference problem, the Variational Autoencoder model introduces an approximate posterior distribution $q(\mathbf{z}_n \mid \mathbf{y}_n; \nu)$. For example, for the generative model given by Equation 3.6, we can propose the following approximate posterior.

$$q(\mathbf{z}_n \mid \mathbf{y}_n; \nu) = \mathcal{N}_d(\mathbf{NN}_\mu(\mathbf{z}_n; \nu), \mathbf{NN}_\sigma(\mathbf{z}_n; \nu))) \tag{3.8}$$

To measure how well the approximate posterior $q(\mathbf{z}_n \mid \mathbf{y}_n; \nu)$ approximates the true posterior $p(\mathbf{z}_n \mid \mathbf{y}_n; \theta)$, we use the Kullback-Leibler divergence as a distance metric between the two distributions, where the distance measures the amount of information lost in nats when we use q to approximate p. The goal of inference is to minimize the following Kullback-Leibler divergence objective

$$\begin{aligned} D_{KL}\big(q(\mathbf{z}_n|\mathbf{y}_n;\nu) \parallel p(\mathbf{z}_n|\mathbf{y}_n;\theta)\big) &= \mathbf{E}_q[\log q(\mathbf{z}_n|\mathbf{y}_n;\nu)] - \mathbf{E}_q[\log p(\mathbf{z}_n,\mathbf{y}_n)] + \log p(\mathbf{y}_n) \\ &= -ELBO(\nu) + \log p(\mathbf{y}_n) \end{aligned}$$

where ELBO stands for the Evidence Lower Bound. Since the evidence; i.e., the marginal distribution $p(\mathbf{y}_n)$, is intractable and since $D_{KL} \geq 0$ by Jensen's inequality, minimizing the

Kullback-Leibler divergence is thus equivalent to maximizing the ELBO. By expanding the log joint (i.e. $\log p(\mathbf{z}_n, \mathbf{y}_n)$), the ELBO can be written as

$$ELBO(\nu, \theta) = \mathbb{E}_{q(\mathbf{z}_n|\mathbf{y}_n;\nu)}\big(\log p(\mathbf{y}_n \mid \mathbf{z}_n; \theta)\big) - D_{KL}\big(q(\mathbf{z}_n \mid \mathbf{y}_n; \nu) \parallel p(\mathbf{z}_n)\big)$$

The training objective that we need to minimize is therefore the negative of the ELBO

$$\mathscr{L}(\mathbf{y}_n, \nu, \theta) = -\mathbb{E}_{q(\mathbf{z}_n|\mathbf{y}_n;\nu)}\big(\log p(\mathbf{y}_n \mid \mathbf{z}_n; \theta)\big) + D_{KL}\big(q(\mathbf{z}_n \mid \mathbf{y}_n; \nu) \parallel p(\mathbf{z}_n)\big) \tag{3.9}$$

Here, the first term is the reconstruction error, which pushes the generative model to learn a good reconstruction the data, whereas the second term, the Kullback-Leibler divergence of the approximate posterior from the prior, functions as a regularizer such that if the inference model outputs a latent representation different from the prior (i.e. the standard normal), the loss objective gets penalized. Inference proceeds by finding the variational parameters that minimize the loss function given by Equation 3.9. Optimization is generally accomplished by stochastic gradient descent with automatic differentiation for gradients.

3.4.2 Explicit Models Vs Implicit Models

Most of the time, we propose models whose likelihoods and priors have an explicit closed-form probability densities. In some occasions, however, if we are able to sample from a simulator that is better at modeling the generative mechanism that gives rise to the data, then we can refrain from specifying an explicit probability density for either the data distribution or the prior. Models that do not specify explicit densities are known in the literature as *implicit models*, first introduced by Diggle and Gratton (1984) to approximate an intractable likelihood and later extended to approximating the posterior distribution by Tavaré, Balding, Griffiths, and Donnelly (1997) in a method that is now known as Approximate Bayesian Computation (ABC) (Pritchard, Seielstad, Perez-Lezaun, & Feldman,

1999). In an implicit model, instead of defining a likelihood, we define a deterministic function $f(.)$ whose input is a random noise source ε_n and whose output is the observation

$$\mathbf{y}_n = f(\varepsilon_n \mid \mathbf{z}_n, \theta); \quad \varepsilon_n \sim s() \tag{3.10}$$

The function $f(.)$ can for example be implemented as a deep neural network. Furthermore, the density for the local unobserved variable can also be defined implicitly as such

$$\mathbf{z}_n = f(\varepsilon_n \mid \theta); \quad \varepsilon_n \sim s() \tag{3.11}$$

For example, Figure 3.7, a reproduction of a similar figure found in Doersch (2016), shows how a 2-D Gaussian space can be transformed into a nonlinear manifold when ε_n undergoes the transformation. The VAE can be thought of as a deep implicit model in which Gaussian noise is transformed by a nonlinear function parameterized by a deep neural network (Goodfellow et al., 2014) into a complex latent structure for the model. To clarify, in an explicit linear latent model variable model such as a Factor Analysis (FA) model with 2-dimensional latent space, the transformation is given by $g(\mathbf{z}_n) = \Lambda \mathbf{z}_n$, where $\Lambda \in \mathbb{R}^{p \times 2}$. In the FA model, the transformation is linear and explicitly specified. In contrast, in an implicit model such as the VAE, the transformation can be highly nonlinear and is not explicitly specified. Instead it is learned by the neural network from the data. To illustrate, consider the VAE's approximate distribution given by Equation 3.8. We can sample from the posterior $q(\mathbf{z}_n \mid \mathbf{y}_n; \nu)$ for observation n

$$\mathbf{z}_n \sim \mathcal{N}_d\big(\mathbf{z}_n | \mathbf{NN}_\mu(\mathbf{y}_n; \nu), \mathbf{NN}_\sigma(\mathbf{y}_n; \nu)\big) \tag{3.12}$$

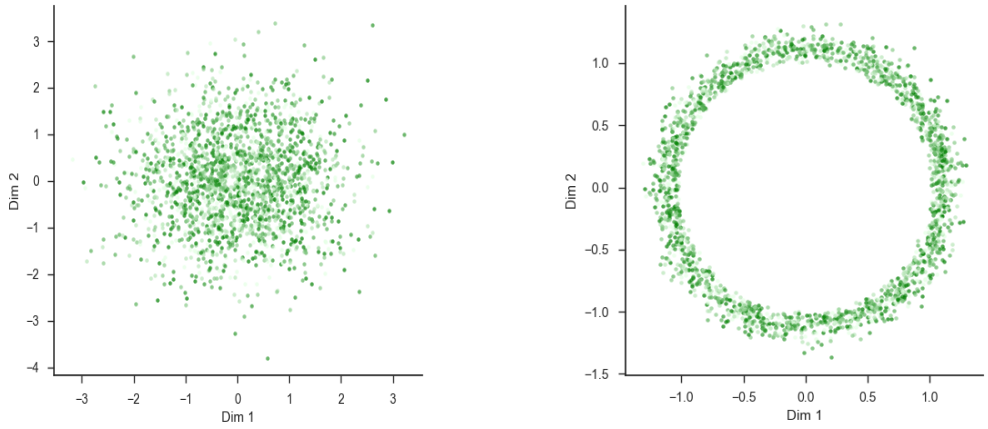
by using the implicit function g

$$\mathbf{z}_n = g(\varepsilon_n, \mathbf{y}_n; \nu) = \mathbf{NN}_\mu(\mathbf{y}_n; \nu) + \mathbf{NN}_\sigma(\mathbf{y}_n; \nu) \odot \varepsilon_n \tag{3.13}$$

where $\odot$ denotes the elementwise product and

$$\varepsilon_n \sim \mathcal{N}_d\big(0, \mathbf{I}\big) \tag{3.14}$$

If the neural networks **NN** are not invertible, as is usually the case, then the function g is not invertible and the likelihood is therefore intractable (i.e. implicit).



(a) 2-D Isotropic Gaussian Latent Space

(b) Transformed Latent Manifold

Figure 3.7: **(a)** Observations $\mathbf{z}_n$ sampled from a 2D multivariate Gaussian variable **(b)** Samples from the latent manifold induced by the transformation $g(\mathbf{z}_n) = \frac{\mathbf{z}_n}{10} + \frac{\mathbf{z}_n}{\|\mathbf{z}_n\|}$

3.4.3 Model Specification

Here we give the exact formulation of the VAE model that is fit to the data. The VAE is a directed graphical model that pairs a generative model, a differentiable generative network that maps the 2-dimensional normally distributed latent variables $\mathbf{z}_n$ into a distribution over samples $\mathbf{y}_n \mid \mathbf{NN}(\mathbf{z}_n; \theta)$, with a differentiable inference network that attempts to approximate the posterior of the latent variables $q_\nu(\mathbf{z}_n \mid \mathbf{y}_n; \nu)$.

The Generative Model

The generative model can be expressed as

$$\begin{aligned} p(\mathbf{y},\mathbf{z};\theta) &= \prod_{n=1}^{N} \text{Bernoulli}_p(\mathbf{y}_n \mid \mathbf{NN}(\mathbf{z}_n;\theta))\mathcal{N}_d(0,\mathbf{I})) \\ &= \prod_{n=1}^{N} \Big(\prod_{p=1}^{90} \text{Bernoulli}(\mathbf{y}_n \mid \mathbf{NN}(\mathbf{z}_n;\theta)[p]) \Big) \mathcal{N}_2(\mathbf{z}_n \mid 0,\mathbf{I}) \end{aligned} \tag{3.15}$$

or as a generative model as

$$\begin{aligned} \mathbf{z}_n &\sim \mathcal{N}_2(0,\mathbf{I}) \quad n = 1,\cdots,N \\ \mathbf{NN}(\mathbf{z}_n;\theta) &= \text{Sigmoid}\Big(\mathbf{W}_p^{(3)} \text{ReLU}\left(\mathbf{W}_p^{(2)} \text{ReLU}(\mathbf{W}_p^{(1)}\mathbf{z}_n + \mathbf{b}_p^{(1)}) + \mathbf{b}_p^{(2)}\right) + \mathbf{b}_p^{(3)}\Big)\Big) \\ \mathbf{y}_n \mid \mathbf{z}_n &\sim \text{Bernoulli}_{90}\Big(\mathbf{NN}(\mathbf{z}_n;\theta)\Big) \end{aligned} \tag{3.16}$$

where $\mathbf{y}_n \in \mathbb{R}^{90}$ is the observation vector for the nth sample, $\mathbf{z}_n \in \mathbb{R}^2$ is the corresponding the latent variable vector; $\mathbf{NN}(\mathbf{z}_n;\theta)$ denotes a function parameterized by a neural network with ReLU and Sigmoid activation functions (a.k.a. nonlinearities); and $\theta = \{(\mathbf{W}_p^{(k)}, \mathbf{b}_p^{(k)})\}_{k=1}^{3} \in \mathbb{R}^{178,522}$ denotes the vector of model parameters such that

$$\begin{aligned} \mathbf{W}_p^{(1)} &\in \mathbb{R}^{256\times 2} \quad \mathbf{b}^{(1)} \in \mathbb{R}^{256\times 1} \\ \mathbf{W}_p^{(2)} &\in \mathbb{R}^{512\times 256} \quad \mathbf{b}^{(2)} \in \mathbb{R}^{512\times 1} \\ \mathbf{W}_p^{(3)} &\in \mathbb{R}^{90\times 512} \quad \mathbf{b}^{(3)} \in \mathbb{R}^{90\times 1} \end{aligned} \tag{3.17}$$

The Inference Model

The approximate posterior of the latent variables can be expressed as

$$\begin{aligned} \mathbf{h}(\mathbf{y}_n;\nu) &= \text{ReLU}\left(\mathbf{W}_q^{(2)} \text{ReLU}\left(\mathbf{W}_q^{(1)}\mathbf{y}_n + \mathbf{b}_q^{(1)}\right)\right) \\ \mathbf{NN}_\mu(\mathbf{y}_n;\nu) &= \mathbf{W}_q^{(3)}\mathbf{h}(\mathbf{y}_n;\nu) + \mathbf{b}_q^{(3)} \\ \mathbf{NN}_\sigma(\mathbf{y}_n;\nu) &= \text{Softplus}\left(\mathbf{W}_q^{(4)}\mathbf{h}(\mathbf{y}_n;\nu) + \mathbf{b}_q^{(4)}\right) \\ q_\nu(\mathbf{z}_n \mid \mathbf{y}_n;\nu) &= \mathcal{N}_2(\mathbf{z}_n \mid \mathbf{NN}_\mu\left(\mathbf{y}_n;\nu\right), \mathbf{NN}_\sigma(\mathbf{y}_n;\nu)\big) \end{aligned} \tag{3.18}$$

where $\nu = \{(\mathbf{W}_q^{(k)}, \mathbf{b}_q^{(k)})\}_{k=1}^4 \in \mathbb{R}^{178,948}$ denotes the vector of variational parameters such that

$$\begin{aligned} \mathbf{W}_q^{(1)} &\in \mathbb{R}^{512\times 90} \quad \mathbf{b}^{(1)} \in \mathbb{R}^{512\times 1} \\ \mathbf{W}_q^{(2)} &\in \mathbb{R}^{256\times 512} \quad \mathbf{b}^{(2)} \in \mathbb{R}^{256\times 1} \\ \mathbf{W}_q^{(3)} &\in \mathbb{R}^{2\times 256} \quad \mathbf{b}^{(3)} \in \mathbb{R}^{2\times 1} \\ \mathbf{W}_q^{(4)} &\in \mathbb{R}^{2\times 256} \quad \mathbf{b}^{(4)} \in \mathbb{R}^{2\times 1} \end{aligned} \tag{3.19}$$

The total number of parameters in the both the inference and the generative models is equal to 357,470.

Note that the VAE model has a total of three deterministic layers for the inference network, one stochastic layer for the latent variables, and three deterministic layers for the generative network. The model can be considered a deep learning model for this reason and also for the historical reason that a fully connected neural network with more than two layers was difficult to learn until the introduction of recent advances that brought about the deep learning revolution. These include robust optimizers such ADAM, less problematic activation functions such as the Rectified linear Activation Unit (ReLU), better regularization techniques such as dropout, and powerful software platforms such as Tensorflow that provide automatic differentiation and utilize the computing power of GPUs (Goodfellow, Bengio, & Courville, 2016).

3.5 Inference

3.5.1 Methods

The model is written in Python using Keras (Chollet, 2015) with Tensoflow 1.0 (Abadi et al., 2016) as the backend. The model was run on a computer with the following software

and hardware specifications: Linux Ubuntu 16.04 operating system, Intel© Core(TM) i7-4790K CPU @ 4.00GHz, 32G of RAM, NVIDIA© GPU GeForce GTX 960 with 4G of memory running on CUDA Toolkit 8.0 and cuDNN v5.1.

Training, validation, and test data sets

Regions for all the standard set of chromosomes are used in the data except for those from Chromosome Y since the ten cells were taken both from the male and female subjects. The data is split into training, validation, and test data sets (i.e. $\mathscr{D}_{train}, \mathscr{D}_{valid}, \mathscr{D}_{test}$) as follows. First, Chromosome 1, Chromosome 8, and Chromosome 21 are merged and kept as held-out test data. The three chromosomes contained a total of 1,946,177 observations. The remaining data containing 11,748,445 regions from the other 20 chromosomes are shuffled then divided into an 80/20 training-validation split. The reason these three chromosomes were used as held-out data is partly due to the following observation. Specifically, the size and distribution of protein coding genes are not uniform across the chromosomes. Chromosome 1 is the largest chromosome in size and has the greatest number of protein coding genes (i.e. 2058). In contrast, Chromosome 21 is the smallest in size and has the fewest number protein coding genes (i.e. 238). Even after taking chromosome size into account, Chromosome 1 has the third highest density of protein coding regions. Chromosome 8 is of medium size, approximately 3 times larger than Chromosome 21, compared to Chromosome 1, which is 5 times larger than Chromosome 21 (Ensembl release 88).

A second important point that should be mentioned is the implicit assumption of *genomic invariance* that we make when we divide the genome into 200 bps segments. By dividing the genome into approximately 14 million regions and treating them as exchangeable observations, we obtain a large number of independent samples that we can use to train

high capacity models. In the context of functional genomics, the genomic invariance assumption implies that the regulatory processes that drive gene expression are similar across the genome (Leung et al., 2016). The assumption might not hold in particular regions, but it can be argued that treating experimental data as coming from a distribution can be justified given the pattern we see in Figure 3.3. The reasoning for why such an approach is justified is similar to the manifold hypothesis argument given in Section 2.4.1 with regards to treating an image of a natural object as a random observation sampled from a distribution.

Optimization

Optimization is run on the training data but the final model parameters are obtained after evaluating the loss on the validation data. Early stopping is implemented such that optimization would stop after four epochs (runs through the entire data) of unimproved validation loss. The test data set is used to evaluate the final generalization error and examine the latent posterior given unseen data.

For optimization, the Nesterov Accelerated Adaptive Moment Estimation (Nadam) algorithm is used. Nadam is a slight variation on the Adam algorithm (D. Kingma & Ba, 2014). In the same way that Adam is basically RMSprop with momentum, Nadam is RMSprop with Nesterov momentum (Sutskever, Martens, Dahl, & Hinton, 2013). All three algorithms, RMSprop, ADAM, and Nadam, are extensions of minibatch Stochastic Gradient Descent (SGD), a first-order stochastic gradient-based optimization method that divides the data into batches and tries to find the minima or maxima by iterating over the batches. The momentum incorporated in the extensions serve the function of remembering previous updates and accelerating the rate of learning for the next update accordingly.

The batch size used for optimization is 256. Accordingly the number of updates per epoch is 36,713 obtained by dividing the total number of observations in the test data by the batch size. The computational cost per update is approximately 1 GFLOP.

For parameter initialization, the Glorot (Xavier) uniform initializer (Glorot & Bengio, 2010) is used for the weight matrices and zeros for the bias vectors. These two settings are the default in Keras for dense fully-connected layers.

3.5.2 Results

In this dissertation, we are mainly concerned with two inferential tasks: understanding the data and understanding the model.

In the context of latent generative modeling and representation learning (Bengio et al., 2013), the first task of understanding the data entails learning a lower-dimensional latent representation of the high-dimensional data. For the concrete case of the VAE model we learn, this is given by the 2-dimensional approximate posterior distribution $q_\nu(\mathbf{z}_n \mid \mathbf{y}_n; \nu)$. According to the manifold hypothesis (Narayanan & Mitter, 2010), the dimensionality of the observed data is much higher than the lower-dimensional latent manifold in which the probability mass of the observations $p(\mathbf{y})$ concentrates. At each point of the manifold, there is a tangent plane that defines the local directions in the space that allow you to stay on the manifold. These directions (i.e. vector basis) define the factors of variation that generate the patterns of dependencies observed in the high-dimensional data. Uncovering these few factors of variation allows us to understand and explain and the structure of the data. For any single 90-dimensional data observation $\mathbf{y}_n$, we can output an approximate posterior distribution of the latent variable, or we can examine its location in the latent space defined by the distribution mean parameters. We expect that if different observations (i.e. genomic

regions) can be classified as different classes in reality, then according to the assumption of natural clustering associated with the manifold hypothesis, these observations would correspond to different manifolds separated by vast regions of low probability. We inspect the inference model component of the VAE model (i.e. the approximate posterior distribution) in the next section.

As for task of understanding the model, turn our attention to the generative model component of the VAE $p(\mathbf{y}, \mathbf{z}; \theta)$. One of the main advantages of using probabilistic generative models is that it allows us to perform model checking simply by sampling from the learned generative model. Discriminative models, whether they are regression or classification models, lack this important capability, making it hard to assess what these models have actually learned from the data, and more importantly what they have missed. In contrast, to sanity-check a generative model, we just generate observations from the model and compare the synthetic samples to real observations to see what the learned model is able to capture and what it has missed. We examine the generative model component of the VAE model in the section that follows.

The model we learn is the one that produced the smallest loss on the validation set. The loss function is the negative of the Evidence Lower Bound (ELBO). The lowest negative ELBO obtained is equal to 18.2881.

Examining the Inference Model

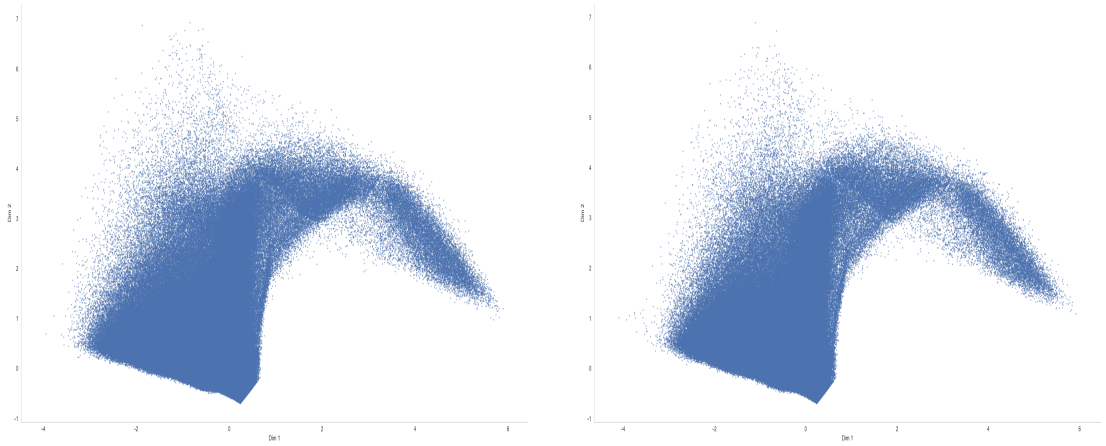
We start with examining the approximate posterior distribution

$$q_{\nu}(\mathbf{z}_n \mid \mathbf{y}_n; \nu) = \mathcal{N}_2\big(\mathbf{z}_n \mid \mathbf{NN}_{\mu}(\mathbf{y}_n; \nu), \mathbf{NN}_{\sigma}(\mathbf{y}_n; \nu)\big) \tag{3.20}$$

In particular, we restrict our attention first to the mean of the approximate posterior of the two-dimensional latent variables, represented by the output of the neural network

$\mathbf{NN}_{\mu}(\mathbf{y}_n; \nu)$. We first project the 90-dimensional observations in the validation data set $\mathscr{D}_{valid}$ into the 2-dimensional latent space by passing the data as input (i.e $\mathbf{y}_n$) to the learned neural network with parameters ν that produced the smallest loss on the validation set. For each observation, the corresponding output generated by the neural network is a two-dimensional mean vector. We do not discuss the parameters of the neural network further since the neural network itself is treated simply as a deterministic function that transforms the isotropic Gaussian into a more complicated distribution.

To evaluate the generalizability of the learned model, the held-out observations in the test data set $\mathscr{D}_{valid}$ are also passed through the same neural network. As can be seen in Figure 3.8, the pattern of clustering of the projected observations in the latent space is highly similar for both data sets. The observations seem to be confined to a particular funnel-shaped region of the two-dimensional space of mean function.

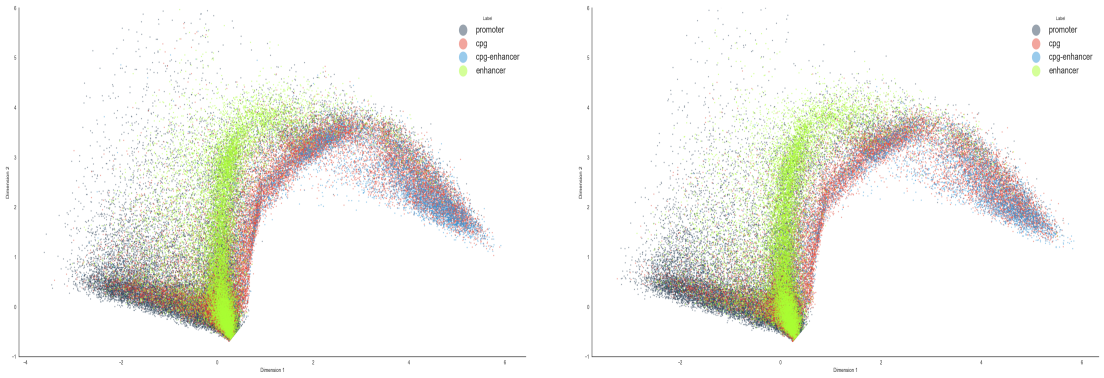


(a) Projection of Validation Data

(b) Projection of Test Data

Figure 3.8: Projections of unlabeled observations(a) Projection of validation data (2,349,696 observations) into the 2D latent manifold. (b) Projection of test data (1,946,177 observations) into the 2D latent manifold

To better understand the structure of the latent space, functional annotations were obtained for a subset of the observations for which there exists externally validated annotation as promoters, enhancers, or CpG islands. See Section 3.2.3 for details. The small fraction of projected observations that have the corresponding labels are then color coded and plotted in Figure 3.9. As can be seen, clear clusters with distinct boundaries in the case of enhancers and CpG island labels can be discerned as overhanging arcs. Also notice that the observations labeled both as "cpg-enhancer" and as "cpg" overlap each other, much as expected since the "cpg-enhancer" label refers to those observations that intersect with genomic regions that are both in the list of permissive enhancers and in the list of CpG islands.



(a) Projection of Labeled Validation Data

(b) Projection of Labeled Test Data

Figure 3.9: Projections of labeled observations (a) Projection of a subset of the validation data into the 2D latent space (84,156 labeled observations: 35,620 promoters, 25,699 CpGs; 6,394 that intersect CpGs-enhancers regions; and 16,443 enhancers) (b) Projection of a subset of test data into the 2D latent space. (73,176 labeled observations: 31,457 promoters, 20,637 CpGs; 5,154 that intersect CpGs-enhancers regions; and 15,928 enhancers)

Examining the Generative Model

To inspect the generative model we have learned (Equation 3.16), we proceed by making the model "hallucinate" observations from the latent space. To examine what the generative model has learned, we select several points in the 2-dimensional latent space of the prior, pass the latent variable values through the neural network, and let the model generate the corresponding 90-dimensional simulated observation. To explore a large extent of the latent space, we first divide the 2-dimensional latent space into a $16 \times 16 = 256$ grid and select a set of 256 somewhat uniformly spaced coordinates $\mathbf{z} = (z_1, z_2)$, where the values of z_1 are restricted to the range [-3.5, 5.5] and the values of z_2 are restricted to the range [-0.5, 5]. Notice that since the prior of the latent space is the standard normal, a sample value equal or greater than four sigmas would have an extremely very low probability of occurring, yet the generative model is able to capture rather well the patterns driven by these tail probabilities. The 256 synthetic samples are shown as a grid in Figure 3.10. It should also be noted that the output of the last layer of the neural network (the sigmoid layer), not the output of the Bernoulli likelihood, is what is being plotted and examined.

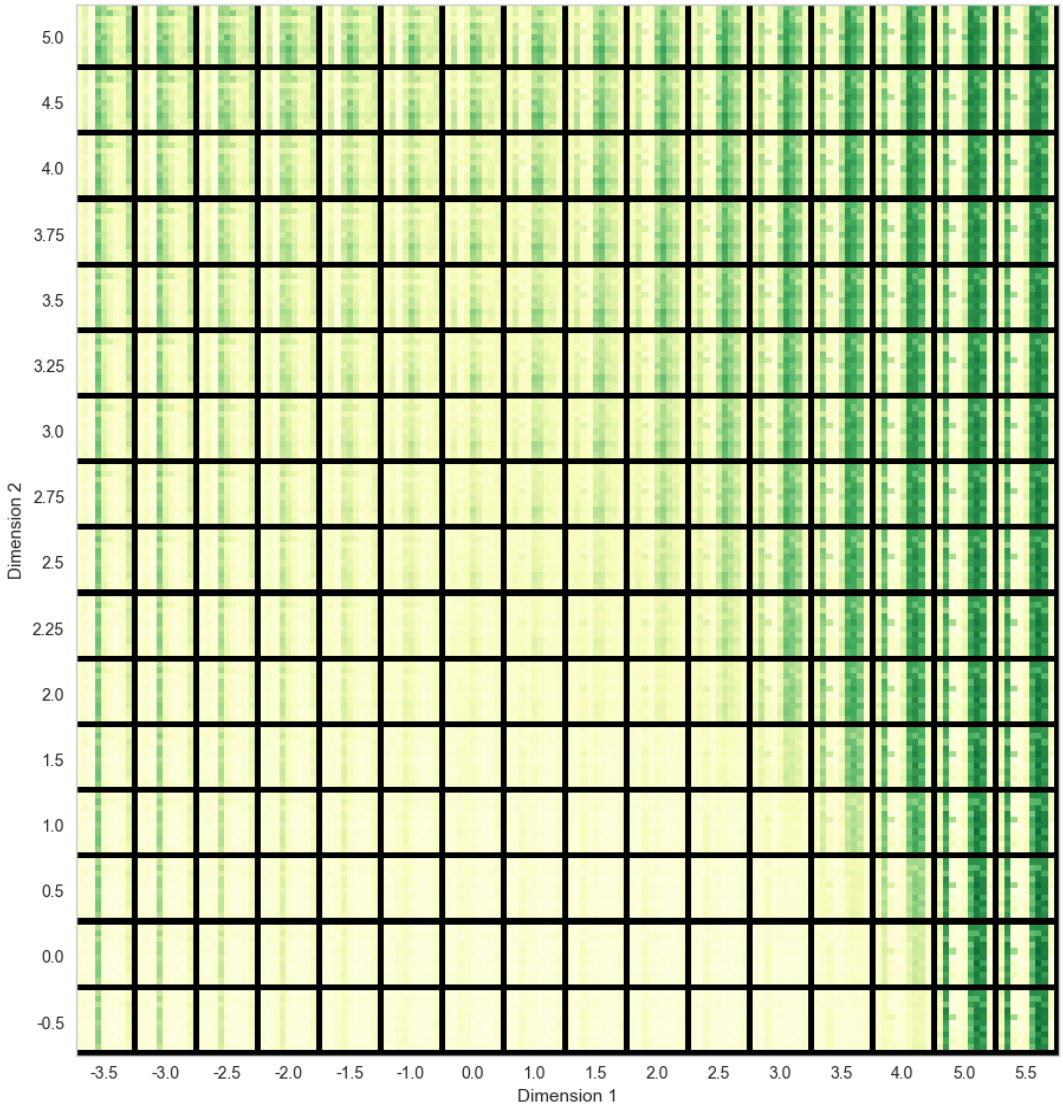


Figure 3.10: A 16×16 grid of simulated samples from the generative model $\mathbf{y}_n \mid \mathbf{z}_n \sim \text{Bernoulli}_{90}\left(\mathbf{NN}(\mathbf{z}_n; \theta)\right)$. Each of the 256 cells contains a 90-dimensional observation reshaped as a 10×9 image. Note that the values depicted in the plots are in fact probabilities (i.e. the output of the final sigmoid layer given by $\mathbf{NN}(\mathbf{z}_n; \theta)$), not the discrete outcomes of the Bernoulli likelihood. The greater the probability value, the darker the color. Zero corresponds to very light yellow and one corresponds to very dark green.

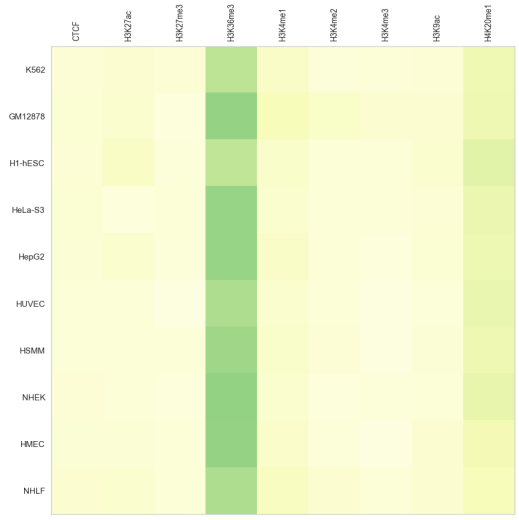
3.5.3 Biological Interpretation

What Patterns We Expect to Observe Several studies have elucidated the association of epigenomic marks with regulatory regions of the genome such as enhancers (Li et al., 2016). For example, two of the most commonly used epigenomic markers for annotating putative enhancers are

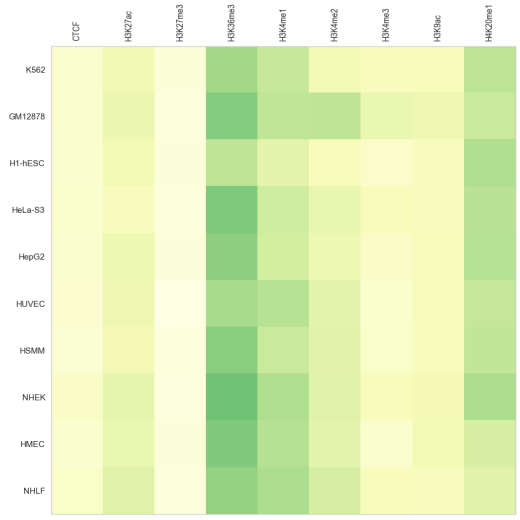
- The presence of histone acetylation. For example, the histone mark H3k27ac has been shown to correlate with active enhancers (Heintzman et al., 2009).
- A high ratio of H3K4me1 compared to H3K4me3. Admittedly, The cut-off threshold for this ratio can be arbitrary and may leave out a large portion of functional enhancers (Pekowska et al., 2011). Whereas H3K4me3 tend to be associated with active promoters (Barski et al., 2007), H3K4me1 is usually found in enhancer regions (Heintzman et al., 2009), even though its presence does not indicate whether an enhancer is actually active or not (Bonn et al., 2012).

Histone marks have also been associated with other regulatory regions; most notably

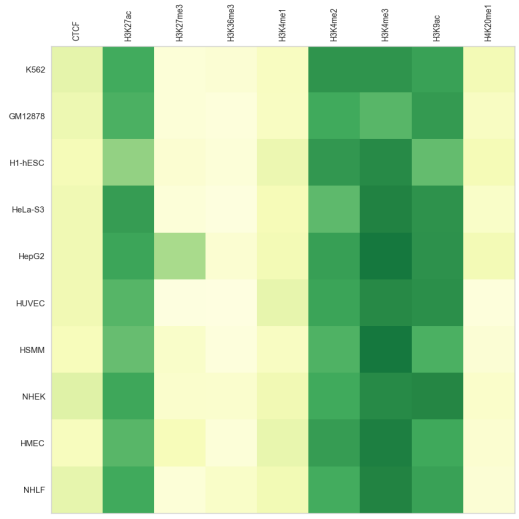
- H3k36me3 is associated with active promoters and gene body transcription (Koch et al., 2011).
- H3k27me3 is associated with Polycomb repression. Polycomb-group proteins are regulatory proteins that silence gene transcription by remodeling chromatin (Bonasio, Tu, & Reinberg, 2010)



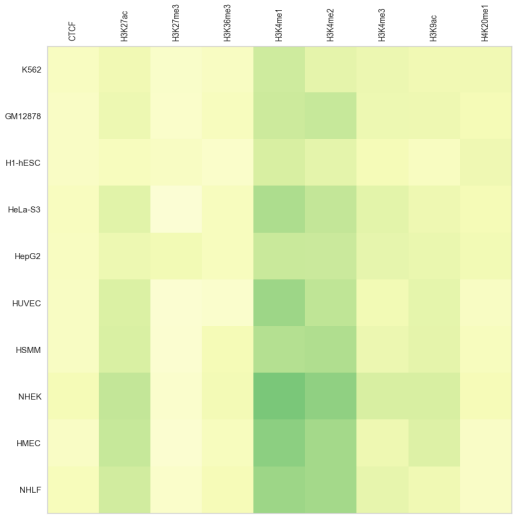
(a) A synthetic observation sampled from point $\mathbf{z} = [-3, 1]$ in the latent space



(b) A synthetic observation sampled from point $\mathbf{z} = [-3, 4]$ in the latent space



(c) A synthetic observation sampled from point $\mathbf{z} = [5, 2]$ in the latent space



(d) A synthetic observation sampled from point $\mathbf{z} = [0.5, 4]$ in the latent space

Figure 3.11: Four Samples simulated from the generative model $\mathbf{y}_n \mid \mathbf{z}_n \sim \text{Bernoulli}_{90}\left(\mathbf{NN}(\mathbf{z}_n; \theta)\right)$. Note that the values depicted in the plots are in fact probabilities (i.e. the output of the final sigmoid layer given by $\mathbf{NN}(\mathbf{z}_n; \theta)$), not the discrete outcomes of the Bernoulli likelihood. The greater the probability value, the darker the color. Zero corresponds to very light yellow and one corresponds to very dark green.

What Patterns We Actually Observe As Figure 3.10 and Figure 3.11 show, the combination of high signal values for the marks corresponding to Columns 2, 6, 7, 8 seem to be driving the variation in the lower right quadrant. These columns correspond to the H3k27ac, H3k4me2, H3k4me3, and H3k9ac histone marks respectively, all four marks of which are associated with active or poised regulatory elements. The variation in the left hand side seems to be mainly driven by a combination of two histone marks, H3k27me3 and H3k36me3, corresponding to Columns 3 and 4, respectively. In both the upper left and lower left quadrants, increasingly high values of H3k36me3 seems to dominate the pattern of variation as we go from right to left. In contrast, as we go up starting from the bottom left quadrant, very small signal values for H3k27me3 seems to dominate. These two main global patterns are present across all ten cell-types (rows). The only exception is the pattern manifested in the samples found in the lower right corner (Figure 3.11c). In particular, for Rows 3 and 5, corresponding to the H1-hESC and HepG2 cells, the signal values for the H3k27ac and H3k27me3 show a contrasting relationship breaking the consistency that is evident across the other eight cell-types. HepG2, a human liver cancer immortal cell line, is characterized by a high signal value for the H3k27me3 histone mark. The opposite pattern is seen for H1-hESC, a embryonic male stem cell, where the signal value for H3k27ac is markedly lower than those for the other nine cell types. As we have hoped, we do not see a major role for H3K4me1 histone mark, a mark that is no longer considered a marker for active enhancers.

3.6 Criticism

3.6.1 Posterior Predictive Checks

At the core of the variational autoencoder is the optimization of the ELBO given by Equation 3.9. Maximizing the ELBO entails maximizing the first term, the reconstruction ability of the model, subject to minimizing the second term, the Kullback-Leibler divergence D_{KL} between the approximate posterior distribution $q_\nu(\mathbf{z}_n|\mathbf{y}_n)$ and the prior distribution $p_\theta(\mathbf{z}_n)$. The D_{KL} serves as a penalty term representing the complexity of the posterior distribution. The effect of the KL term is to drive the model to learn a high capacity function that can transform a latent variable $\mathbf{z}_n$ into a distribution that can reconstruct a given observation $\mathbf{y}_n$.

To measure the reconstruction ability of the model, we can propagate sample observations into the inference and generative models and observe the output. If the model can reproduce the pattern of dependencies we see in the observations, we then have more confidence in the generative ability of the model. In Figure 3.12, 6 random samples taken from the test data set $\mathscr{D}_{test}$ are depicted next to their reconstructions. These 6 samples are labeled as enhancers. As can be seen, the reconstructions are able to capture the pattern of the dependencies of the 4th mark across the ten cell types for four out the six samples. However, the pattern of complex dependency evident in the 9th histone mark is smoothed over and attenuated.

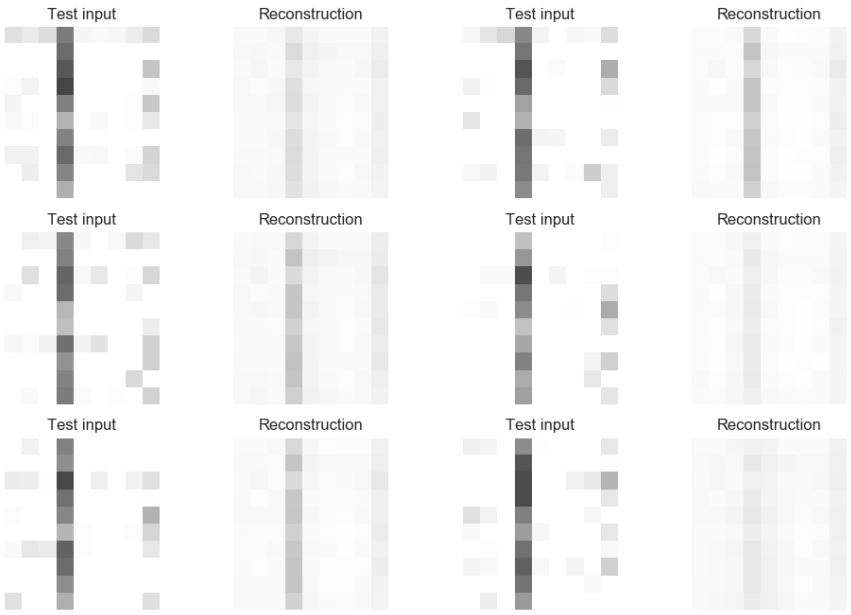


Figure 3.12: Reconstructions obtained by projecting then generating 6 test input samples from the latent manifold

Given that the latent variable in the formulated model is only two dimensional, the generative models representational capacity is rather limited and accordingly we should expect the re-constructive accuracy to suffer as a result. We also expect that the variational autoencoder model to learn the factors of variation that are most needed for reconstructing the most salient observed dependencies in the data. Increasing the dimensionality of the latent space should improve both the representational power of the latent space and the reconstructive ability of the model. However, visualizing latent spaces of higher dimensionality using two dimensional plots becomes increasingly difficult beyond three dimensions.

3.6.2 Discussion

The main goal of the deep generative model implemented in this work is to learn the structure of the lower dimensional manifold in which the probability mass $p(\mathbf{y})$ concentrates. A second related goal is to learn a good representation of the data in which the axes (i.e. features) of the lower-dimensional learned representational space correspond to independent factors of variation that generate the observed data. Unlike common statistical models in which the goal is to reject a particular hypothesis or compare different models, the aim of the *representation learning* approach (Bengio et al., 2013) taken here is to build a generative model that is able to capture the pattern of dependencies in our 90 dimensional data in such a way that the model we ultimately learn is, first, able to reconstruct the patterns observed in the data from a much compressed two dimensional latent representation, and second, can help us both explain and interpret the underlying latent structure that determines the observed dependencies in the data. Whereas interpretation can be obtained by assigning meaningful labels to the axes of the lower dimensional latent space, explanation comes in the form of our ability to examine the latent space and observe how our model's output changes in response to modifying the input; that is, moving from one point to another in the latent space.

In particular, unlike the approach taken here, the methods surveyed in Section 3.1.5, treat chromatin state as the output of a compressed histone regulatory code, similarly to how in the genetic code the set of $4^3 = 64$ possible trinucleotides are compressed into a set of 21 amino acids (B. Banerjee & Sherwood, 2017). More precisely, according to the Histone Code model (Jenuwein & Allis, 2001), chromatin structure, and gene expression activity as a result, is determined by the combinatorial effect of a finite number of post-translational modifications (e.g. acetylation and methylation) of histone proteins. Since

only a few of all possible histone marks combinations seem to have a functional role, these combinations that are functionally important can be referred to by distinct chromatin states, labels with which we can use to define regulatory elements and functionally annotate the human genome. In contrast, the compressed 2-dimensional representation learned by the proposed generative model views the histone code not only as continuous but also as more extensive. The learned latent manifold can be used to explain the patterns of dependencies not only across the nine epigenomic marks but also across the ten cell-types and even among the interaction of the cell-types and the marks. By modeling the latent space as continuous, we no longer need to set an arbitrary threshold to separate the states, instead we can consider the full extent of activity levels.

Deep latent generative models have several advantages over explicit, shallow, fully observable, or discriminative models. Indeed, the deep latent generative approach allows us to specify very *scalable* and *flexible* models in a disciplined way, to perform model checking by sampling, to learn compact representations, to understand the factors of variation that govern the data, and to take advantage of unlabeled data with semi-supervised learning. Indeed, it is problem of semi-unsupervised learning that the approach we presented in this dissertation holds the greatest promise. Given that the vast majority of the observations (i.e. genomic regions) are functionally unlabeled, learning a good representation of the data is an important first step in successfully performing further tasks. For example, if the goal is to predict enhancers using a supervised model, then we would expect that any supervised method approach trained on the data would most likely result in severe overfitting since that learning is restricted to just a fraction of the observations; namely those that have existing labels. Examples of existing supervised approaches to enhancer prediction include RFECS (Rajagopal et al., 2013), EnhancerFinder (Erwin et al., 2014), and GKM-SVM (Ghandi,

Lee, Mohammad-Noori, & Beer, 2014). A potential benefit of the approach proposed in this work to the enhancer prediction task and the problem of overfitting can be gained by using a *semi-supervised* approach in which the representation we have learned from the unlabeled data are incorporated in a supervised predictive model. Alternatively, if the goal is to identify enhancers using an unsupervised model, then through the structure of the latent manifold we learned, we can devise ways to segment or threshold regions of the latent space such that projected observations that fall in particular regions get labeled and defined according to how other projected observations are labeled. A successful semi-unsupervised extension to the generative model presented can have important consequences. For each observation correctly labeled and subsequently experimentally validated, we would be able to shed light on a small window of the dark matter of the genome and better understand the functional role of a particular regulatory region of interest. Notwithstanding, it is important to emphasize that statistical models cannot substitute for a theoretical and a causal understanding of the phenomenon of interest. With regards to the general problem outlined in this dissertation, the ultimate usefulness of any statistical modeling approach can only be evaluated by its ability to help us explain and predict how an experimentally induced perturbation in DNA sequence affects changes in epigenomic features and any subsequent modulation in cell-type specific gene expression, especially for cell types for which we have not obtained any training data.

Appendix A: Analysis Code

A.1 Preprocessing Code

A.1.1 Create Blacklist

```
library(data.table)
library(GenomicRanges)

dirpath <- "./genome_data/annotation/Exclude"

dac_filepath <- file.path(dirpath,"consensusBlacklist.bed.gz")
duke_filepath <- file.path(dirpath,"dukeExcludeRegions.bed.gz")
gaps_filepath <- file.path(dirpath, "gap.txt.gz")

download.file("http://hgdownload.cse.ucsc.edu/goldenPath/hg19/database/gap.txt.gz",
              gaps_filepath)
download.file("http://hgdownload.cse.ucsc.edu/goldenPath/hg19/encodeDCC/wgEncodeMapability/wgEncodeDacMapabilityConsensusExcludable.bed.gz",
              dac_filepath)
download.file("http://hgdownload.cse.ucsc.edu/goldenPath/hg19/encodeDCC/wgEncodeMapability/wgEncodeDukeMapabilityRegionsExcludable.bed.gz",
              duke_filepath)

dac  <- fread(input = sprintf("zcat < %s", dac_filepath), sep = "\t")
duke <- fread(input = sprintf("zcat < %s", duke_filepath), sep = "\t")
gaps <- fread(input = sprintf("zcat < %s", gaps_filepath), sep = "\t")

dac_grange <- GRanges(seqnames = dac$V1,
                      ranges   = IRanges(start = dac$V2,
                                         end = dac$V3))
duke_grange <- GRanges(seqnames = duke$V1,
                       ranges   = IRanges(start = duke$V2,
                                          end = duke$V3))
gaps_grange <- GRanges(seqnames = gaps$V2,
                       ranges   = IRanges(start = gaps$V3,
                                          end = gaps$V4))
```

```

chromosomes <- c('chr1','chr2','chr3','chr4','chr5',
                 'chr6','chr7','chr8','chr9','chr10',
                 'chr11','chr12','chr13','chr14','chr15',
                 'chr16','chr17','chr18','chr19',
                 'chr20','chr21','chr22','chrX')

dac_grange <- keepSeqlevels(dac_grange, chromosomes)
duke_grange <- keepSeqlevels(duke_grange, chromosomes)
gaps_grange <- keepSeqlevels(gaps_grange, chromosomes)

blacklist <-  c(dac_grange, duke_grange, gaps_grange)
blacklist <- reduce(blacklist)
blacklist <- as.data.frame(blacklist)[,1:3]
fwrite(blacklist,
       file.path(dirpath, "blacklist.bed"),
       col.names = FALSE, sep = "\t")
```

A.1.2 Combine Signals

```
import numpy as np
import os.path as op
import subprocess as sp
import time

from pybedtools import BedTool, Interval
import pyfasta
import parmap
# Standard library imports
import itertools
import pyBigWig


rootpath = op.abspath('/hd2/DeepChrome')
annotation = op.join(rootpath, 'annotation')
genome_sizes_file = op.join(annotation,'hg19.chrom.sizes')
blacklist = op.join(annotation,'Exclude/blacklist.bed')
out_dir = op.join(rootpath, 'output')
genome_filtered_filepath = op.join(out_dir,'hg19_200bp_filtered.bed')

genome_sizes_info = np.loadtxt(genome_sizes_file, dtype=bytes).astype(
    str)
chromosomes = list(genome_sizes_info[:, 0])

numthreads= 6
out_dir_multibigwig = op.join(out_dir, "multibigwig")


genome_window_size = 200
genome_windows = BedTool().window_maker(g=genome_sizes_file,
                                        w=genome_window_size,
```

```
                                                output=op.join(out_dir,'
    hg19_200bp.bed'))


genome_windows_filtered = genome_windows.intersect(blacklist,
                                                   wa=True,
                                                   v=True,
                                                   sorted=True,
                                                   output=
    genome_filtered_filepath)


############

datapath = op.join(rootpath, 'experimental_data')
metadata_path = op.join(rootpath,"experimental_data/metadata.txt")

metadata = np.loadtxt(metadata_path, dtype=bytes).astype(str)
marks = list(metadata[:, 0])
cells = list(metadata[:, 1])
filesname = list(metadata[:, 2])

labels_list= [ c + m for c, m  in zip(cells,marks)]
labels_list= np.array(labels_list)
labels_list= ' '.join(labels_list)

files_list = [ op.join(datapath, c, f) for c, f in zip(cells, filesname)
    ]
files_list =' '.join(files_list)




for region in chromosomes:
    cmd = ['multiBigwigSummary ', 'BED-file',
           '-b', '%s' % files_list,
           '-l', '%s' % labels_list,
           '--BED', genome_filtered_filepath,
           '-r', region,
           '-out', op.join(out_dir_multibigwig, '%s_scores.npz' % region
    ),
           '--outRawCounts', op.join(out_dir_multibigwig, '%s_scores.tab
    ' % region),
           '-p', '%s' % numthreads]
    cmd = ' '.join(cmd)
    sp.check_call(cmd, shell=True)

#############

out_dir_datasets = op.join(out_dir,'datasets')
cpg_bed = BedTool(op.join(annotation,'cpgisland.bed.gz'))
```

```
enhancer_bed = BedTool(op.join(annotation,'permissive_enhancers.bed'))
promoter_bed = BedTool(op.join(annotation,'wgEncodeGencodeBasicV19.promoter.merged.bed.gz'))
cds_bed = BedTool(op.join(annotation,'wgEncodeGencodeBasicV19.cds.merged.bed.gz'))


def createDatasets(region):
    start_time = time.monotonic()
    data_tab = np.loadtxt(op.join(out_dir_multibigwig, '%s_scores.tab' % region), dtype=bytes).astype(str)
    bigwig_data = data_tab[:, 3:].astype(np.float32)
    print('data loaded')
    #del data_tab
    bigwig_data[np.isnan(bigwig_data)] = 0
    # subtract control signal from other tracks of each cell-type
    for i in range(0, bigwig_data.shape[1], 10):
        bigwig_data[:, i:(i + 10)] = bigwig_data[:, i:(i + 10)] - bigwig_data[:, i:(i + 1)]

    bigwig_data[bigwig_data < 0] = 0
    # remove control signals
    bigwig_data = np.delete(bigwig_data, range(0, bigwig_data.shape[1], 10),1)

    # remove rows where there is no signal
    ind = bigwig_data.sum(axis=1) != 0
    bigwig_data = bigwig_data[ind, :]
    # transform the signal
    bigwig_data = np.arcsinh(bigwig_data)

    print('Number of observations thresholded: %d ' % np.sum(bigwig_data > 6) )

    # restrict range
    bigwig_data[bigwig_data > 6] = 6
    #restrict range to 0-1
    bigwig_data = bigwig_data / np.max(bigwig_data, axis=0)

    positions_data = data_tab[ind, 0:3]
    print('Number Rows removed: %d ' % np.sum(~ind) )

    peaks = []
    for chrom, start, stop in zip(positions_data[:, 0], positions_data[:, 1].astype(int), positions_data[:, 2].astype(int)):
        peaks.append(Interval(chrom, start, stop))
    peaks = BedTool(peaks)

    peaks_cpg = peaks.intersect(cpg_bed, wa=True, c=True)
    peaks_promoter = peaks.intersect(promoter_bed, wa=True, c=True)
    peaks_enhancer = peaks.intersect(enhancer_bed, wa=True, c=True)
```

```
    peaks_cds = peaks.intersect(cds_bed, wa=True, c=True)


    data_list = []
    data_dict = {}
    for  cpg,  enhancer, promoter, cds in zip(peaks_cpg,
                                              peaks_enhancer,
                                              peaks_promoter,
                                              peaks_cds):

        #position = p.chrom+':'+ str(p.start) + ':'+ str(p.stop)
        annotation = str(int(bool(cpg.count))) + str(int(bool(enhancer.
    count))) + \
                     str(int(bool(promoter.count))) + str(int(bool(cds.
    count)))

        data_list.append( annotation)
    data_dict['cellmarks'] = bigwig_data
    data_dict['annotations'] = np.array(data_list)  #[x[0] for x in
    data_list])
    data_dict['positions'] = positions_data #np.array([x[1] for x in
    data_list])
    fileout = op.join(out_dir_datasets, '%s_datasets.npz' % region)
    f = open(fileout, "wb")
    np.savez_compressed(f,
                        cellmarks=data_dict['cellmarks'],
                        annotations=data_dict['annotations'],
                        positions=data_dict['positions'])
    f.close()
    print('File saved: %s ' % region)
    elapsed_time = time.monotonic() - start_time
    print('total time: %f ' % elapsed_time )

#region='chr1'

for chrom in chromosomes:
    createDatasets(chrom)


################################

import matplotlib.pyplot as plt

a = bigwig_data[:,1]
plt.hist(aa, bins='auto')
plt.show()


################
```

A.2 Tensorflow Model Code

```
"""Variational Autoencoder
see: Kingma & Welling - Auto-Encoding Variational Bayes
(http://arxiv.org/abs/1312.6114)
"""
from __future__ import absolute_import
from __future__ import division
from __future__ import print_function

import tensorflow as tf
import matplotlib.pyplot as plt
import numpy as np
from collections import OrderedDict
from keras import backend as K
from keras.layers import Dense, Input, Lambda
from keras.models import Model
from keras.callbacks import ModelCheckpoint, EarlyStopping, CSVLogger

from keras import objectives
import os.path as op

def LoadData(data_dir, training_chromosomes):
    x_all = OrderedDict()
    y_all = OrderedDict()
    loci_all = OrderedDict()
    for chr in training_chromosomes:
        train_data = op.join(data_dir, '%s_datasets.npz' % chr)
        with np.load(train_data) as data:
            x_all['x_%s' % chr] = data['cellmarks']
            y_all['y_%s' % chr] = data['annotations']
            loci_all['loci_%s' % chr] = data['positions']
    x_all = np.concatenate([x_all[x] for x in x_all], 0)
    y_all = np.concatenate([y_all[y] for y in y_all], 0)
    loci_all = np.concatenate([loci_all[loci] for loci in loci_all], 0)
    return x_all, y_all, loci_all


def ShuffleSplitDataset(x, y, loci, batch_size, validation_split=0.2):
    # randomize observations
    random_indx = np.arange(x.shape[0])
    np.random.shuffle(random_indx)
    x = x[random_indx, :]
    y = y[random_indx]
    loci = loci[random_indx, :]
    # split and round up the nearst hundred
    split_index = int(len(random_indx) * (1 - validation_split))
    split_index -= split_index % -batch_size + batch_size
    x = np.array_split(x, [split_index], axis=0)
    y = np.array_split(y, [split_index], axis=0)
    loci = np.array_split(loci, [split_index], axis=0)
```

```
    train_data = {'x': x[0], 'y': y[0], 'loci': loci[0]}
    valid_data = {'x': x[1], 'y': y[1], 'loci': loci[1]}
    # trim valid data to have multiples of 100
    len_valid = valid_data['x'].shape[0]
    last_ind = (len_valid // batch_size) * batch_size
    valid_data = {v: np.delete(valid_data[v], range(last_ind, len_valid)
    , 0) for v in valid_data}
    return train_data, valid_data


rootpath = op.abspath('/hd2/vae_chromatin')
data_dir = op.join(rootpath, 'output/datasets')

heldout_chromosomes = ['chr1', 'chr8', 'chr21']
training_chromosomes = ['chr2', 'chr3', 'chr4', 'chr5', 'chr6', 'chr7',
                        'chr9', 'chr10', 'chr11', 'chr12', 'chr13',
                        'chr14', 'chr15', 'chr16', 'chr17', 'chr18',
                        'chr19', 'chr20', 'chr22', 'chrX']
# optimization
BATCH_SIZE = 256
EPOCHS = 15

x_all, y_all, loci_all = LoadData(data_dir,
                                  training_chromosomes)

train_data, valid_data = ShuffleSplitDataset(x_all,
                                             y_all,
                                             loci_all,
                                             batch_size=BATCH_SIZE)
x_train = train_data['x']
x_valid = valid_data['x']

# Model
#The code in this block borrows from the example script found in
#the Keras repository https://github.com/fchollet/keras/blob/
# master/examples/variational_autoencoder.py
#####################################

# architecture
INPUT_DIM = 90
HIDDEN_DIM_1 = 512
HIDDEN_DIM_2 = 256
LATENT_DIM = 2
ACTIVATION = 'relu'

### Instantiate Layers

pDense_1 = Dense(HIDDEN_DIM_2,
                 activation=ACTIVATION)
pDense_2 = Dense(HIDDEN_DIM_1,
                 activation=ACTIVATION)
```

```
pSquash = Dense(INPUT_DIM,
                activation='sigmoid')
#
qDense_1 = Dense(HIDDEN_DIM_1,
                 activation=ACTIVATION)
qDense_2 = Dense(HIDDEN_DIM_2,
                 activation=ACTIVATION)
qLinear = Dense(LATENT_DIM)
qSoftplus = Dense(LATENT_DIM,
                  activation='softplus')

def NN_p(z):
    p_layer_1 = pDense_1(z)
    p_layer_2 = pDense_2(p_layer_1)
    x_out = pSquash(p_layer_2)
    return x_out


def NN_q(x_in):
    layer_1 = qDense_1(x_in)
    layer_2 = qDense_2(layer_1)
    mu_z = qLinear(layer_2)
    var_log_z = qSoftplus(layer_2)
    return mu_z, var_log_z


def Normal(args):
    mu, var_log = args
    epsilon = K.random_normal((BATCH_SIZE, LATENT_DIM), 0, 1)
    # z = mu + sigma*epsilon
    z_sim = mu + K.exp(var_log / 2) * epsilon
    return z_sim


def KLqp_loss(x_in, x_out):
    reconstr_loss = INPUT_DIM * objectives.binary_crossentropy(x_in,
    x_out)
    latent_loss = -0.5 * K.sum(1 + var_log_z
                               - K.square(mu_z)
                               - K.exp(var_log_z), axis=-1)
    loss = reconstr_loss + latent_loss
    return loss


# Inference model
x_in = Input(batch_shape=(BATCH_SIZE, INPUT_DIM))
mu_z, var_log_z = NN_q(x_in)
# Generative model
z = Lambda(Normal)([mu_z, var_log_z])
x_out = NN_p(z)
# x     = Bernoulli(logits= logits)
```

```
150
151 vae = Model(inputs=x_in, outputs=x_out)
152 vae.compile(optimizer='Nadam', loss=KLqp_loss)
153
154 checkpoint_filepath = op.join(rootpath, 'output/weights.best.hdf5')
155 log_dir = op.join(rootpath, 'output/logs')
156
157 callbacks_list = [
158     EarlyStopping(monitor='val_loss',
159                   patience=6,
160                   verbose=1,
161                   mode='min'),
162     ModelCheckpoint(checkpoint_filepath,
163                     monitor='val_loss',
164                     save_best_only=True,
165                     verbose=1),
166     CSVLogger(op.join(rootpath, 'output/training.log'))
167 ]
168
169 vae.fit(x_train, x_train,
170         shuffle=True,
171         epochs=EPOCHS,
172         batch_size=BATCH_SIZE,
173         validation_data=(x_valid, x_valid),
174         callbacks=callbacks_list)
175
176 ################################################
177
178 # Print trainable variable parameter statistics to stdout.
179 param_stats = tf.contrib.tfprof.model_analyzer.print_model_analysis(
180     tf.get_default_graph(),
181     tfprof_options=tf.contrib.tfprof.model_analyzer.
182         TRAINABLE_VARS_PARAMS_STAT_OPTIONS)
183
184 # param_stats is tensorflow.tfprof.TFProfNode proto. It organize the
    statistics
185 # of each graph node in tree scructure. Let's print the root below.
186 sys.stdout.write('total_params: %d\n' % param_stats.total_parameters)
187
188 # Print to stdout an analysis of the number of floating point
189 # operations in the model broken down by individual operations.
190 tf.contrib.tfprof.model_analyzer.print_model_analysis(
191     tf.get_default_graph(),
192     tfprof_options=tf.contrib.tfprof.model_analyzer.FLOAT_OPS_OPTIONS)
193 ########################
194
195 # load weights
196 vae.load_weights(checkpoint_filepath)
197 # Compile model (required to make predictions)
198 vae.compile(optimizer='Nadam', loss=KLqp_loss)
199
```

```
####################################
encoder = Model(x_in, mu_z)
x_valid_encoded = encoder.predict(x_valid, batch_size=BATCH_SIZE)

########################
x_test, y_test, loci_test = LoadData(data_dir, heldout_chromosomes)

test_data = {'x_test': x_test,
             'y_test': y_test,
             'loci_test': loci_test}
len_valid = x_test.shape[0]
last_ind = (len_valid // BATCH_SIZE) * BATCH_SIZE
test_data = {v: np.delete(test_data[v], range(last_ind, len_valid), 0)
    for v in test_data}

test_encoder = Model(x_in, mu_z)
x_test_encoded = test_encoder.predict(test_data['x_test'],
                                      batch_size=BATCH_SIZE)

##############################

import pandas as pd
import seaborn as sns

params = {'legend.loc': 'upper right',
          'legend.fontsize': 18,
          'legend.frameon': True,
          'legend.fancybox': True,
          'legend.shadow': True,
          'legend.markerscale': 8}
sns.set(rc=params)
sns.set_style("white", {"axes.grid": False})

###
labels = {'1000': "cpg", '0100': "enhancer", '0010': "promoter",
          '1001': "cpg", '0101': "enhancer", '0011': "promoter",
          '1010': "cpg-enhancer", '1011': "cpg-enhancer"}

levels = ["promoter", "cpg", "cpg-enhancer", "enhancer"]
palette = sns.color_palette(["#34495e", "#e74c3c", "#3498db", '#aaff32'
    ])

### Plot validation

df_valid = pd.DataFrame({'Dim 1': x_valid_encoded[:, 0],
                         'Dim 2': x_valid_encoded[:, 1],
                         'Label': valid_data['y']})

sns.lmplot(x='Dim 1', y='Dim 2',
           # col="label",
           data=df_valid,
```

```
            # col_wrap=2,
            ci=None,
            fit_reg=False,
            size=4,
            scatter_kws={"s": 6, "alpha": 0.6})

df_valid = df_valid.query('Label in
    ["1000","1001","0100","0101","0010","0011","1010","1011"]')
df_valid['Label'].replace(labels, inplace=True)

sns.lmplot(x='Dim 1', y='Dim 2',
            # col="Label",
            hue="Label",
            hue_order=levels,
            data=df_valid,
            legend_out=False,
            # col_wrap=2,
            ci=None,
            fit_reg=False,
            palette=palette,
            size=4,
            scatter_kws={"s": 7, "alpha": 0.5})

#### Plot test

df_test = pd.DataFrame({'Dim 1': x_test_encoded[:, 0],
                        'Dim 2': x_test_encoded[:, 1],
                        'Label': test_data['y_test']})

sns.lmplot(x='Dim 1', y='Dim 2',
            # col="label",
            data=df_test,
            # col_wrap=2,
            ci=None,
            fit_reg=False,
            size=4,
            scatter_kws={"s": 6, "alpha": 0.6})

df_test = df_test.query('Label in ["1000", "1001", "0100", "0101",
    "0010", "0011", "1010", "1011"]')

df_test['Label'].replace(labels, inplace=True)

sns.lmplot(x='Dim 1',
            y='Dim 2',
            # col="label",
            hue="Label",
            hue_order=levels,
            data=df_test,
            # col_wrap=2,
            ci=None,
```

```
            legend_out=False,
            fit_reg=False,
            palette=palette,
            size=4,
            scatter_kws={"s": 7, "alpha": 0.5})

sns.lmplot(x='Dim 1',
           y='Dim 2',
           col="Label",
           hue="Label",
           data=df_test,
           col_wrap=4,
           ci=None,
           fit_reg=False,
           palette=palette,
           size=3,
           scatter_kws={"s": 7, "alpha": 0.5})

########################
# 2D Visualization of the Learned Latent manifold

_z = Input(shape=(LATENT_DIM,))
_x_out = NN_p(_z)
generator = Model(_z, _x_out)

grid_y = np.array([5., 4.5, 4.0, 3.75, 3.50, 3.25, 3.0, 2.75,
                   2.5, 2.25, 2.0, 1.5, 1.00, 0.5, 0.0, -0.5])
grid_x = np.array([-3.5, -3.0, -2.5, -2.0, -1.5, -1.0, 0.0, 1.0,
                   1.5, 2.0, 2.5, 3.0, 3.5, 4.0, 5.0, 5.5])

n = 16
digit_size1 = 10
digit_size2 = 9
figure = np.zeros((digit_size1 * n, digit_size1 * n))

for j, xi in enumerate(grid_x):
    for i, yi in enumerate(grid_y):
        z_sample = np.array([[xi, yi]])
        print(z_sample)
        x_decoded = generator.predict(z_sample)
        digit = x_decoded[0].reshape(digit_size1, digit_size2)
        figure[
        i * digit_size1: (i + 1) * digit_size1,
        j * digit_size1: (j + 1) * digit_size1] = \
            np.concatenate((digit, np.zeros((10, 1))), axis=1)

idx = np.arange(1, n + 1) * 10
figure = np.insert(figure, idx, 0, axis=0)
figure2 = np.ones((figure.shape[0], figure.shape[1]))
# figure.shape
dx, dy = 11, 10
```

```
# Modify the image to include the grid
figure2[:, 9::dy] = 0
figure2[10::dx, :] = 0
my_cmap = plt.cm.YlGn
my_cmap.set_under('k', alpha=0)
fig, ax = plt.subplots()
# plt.figure(figsize=(10, 10))
img2 = ax.imshow(figure2, cmap=plt.cm.gray)
img1 = ax.imshow(figure, cmap=my_cmap, clim=[0.00001, 1])
# plt.axis('off')
xticks = np.arange(4, dy * n + 4, dy)
yticks = np.arange(5, dx * n + 5, dx)
ax.set_yticks(yticks)
ax.set_xticks(xticks)
ax.set_xticklabels(grid_x)
ax.set_yticklabels(grid_y)
plt.show()

########################
### plot one sample

x_labels = ['CTCF', 'H3K27ac', 'H3K27me3', 'H3K36me3', 'H3K4me1',
            'H3K4me2', 'H3K4me3', 'H3K9ac', 'H4K20me1']
y_labels = ['K562', 'GM12878', 'H1-hESC', 'HeLa-S3', 'HepG2',
            'HUVEC', 'HSMM', 'NHEK', 'HMEC', 'NHLF']

z_sample = np.array([[5, 2]])
# z_sample = np.array([[-3, 1]])
# z_sample = np.array([[-3, 4]])
# z_sample = np.array([[0.5, 4]])

x_decoded = generator.predict(z_sample)
digit = x_decoded[0].reshape(digit_size1, digit_size2)

fig, ax = plt.subplots(figsize=(10, 12))
ax.imshow(digit, vmin=0, vmax=1, cmap=plt.cm.YlGn)
xticks = np.arange(0, 9)
yticks = np.arange(0, 10)
ax.set_yticks(yticks)
ax.set_xticks(xticks)
ax.set_xticklabels(x_labels, rotation='vertical')
ax.set_yticklabels(y_labels)
ax.xaxis.tick_top()
########################

reconstruct = Model(x_in, x_out)
x_test_encoded = reconstruct.predict(test_data['x_test'],
                                     batch_size=BATCH_SIZE)

plt.figure(figsize=(12, 8))
for i in range(12):
```

```
    plt.subplot(3, 4, 2 * i + 1)
    plt.imshow(test_data['x_test'][i + 200000].reshape(10, 9),
               vmin=0, vmax=1)
    plt.axis('off')
    plt.title("Test input")
    plt.subplot(3, 4, 2 * i + 2)
    plt.imshow(x_test_encoded[i + 200000].reshape(10, 9),
               vmin=0, vmax=1)
    plt.title("Reconstruction")
    plt.axis('off')
plt.tight_layout()

############################
ind_samples = (y_test == '1000')
samples = x_test[ind_samples, :]

loci_samples = loci_test[ind_samples, :]

xticks = np.arange(0, 9)
yticks = np.arange(0, 10)
fig = plt.figure(figsize=(8, 6))
for i in range(4):
    ind = 10 * i + 4000
    ax = fig.add_subplot(2, 2, i + 1)
    ax.imshow(samples[ind].reshape(10, 9),
              vmin=0,
              vmax=1,
              cmap=plt.cm.YlGn)

    locus = loci_samples[ind][0] + ':' + \
            str(loci_samples[ind][1]) + ':' + \
            str(loci_samples[ind][2])

    ax.set_yticks(yticks)
    ax.set_xticks(xticks)
    ax.tick_params(axis='both', which='major', labelsize=5)
    ax.tick_params(axis='both', which='minor', labelsize=5)
    ax.set_xticklabels(x_labels, rotation='vertical')
    ax.set_yticklabels(y_labels)
    ax.set_title("%s" % locus)
plt.tight_layout()
```

Appendix B: Supplementary Materials

B.1 URLs for the 100 Epigenetic Marks bigWig Files

```
1  ftp.ebi.ac.uk/pub/databases/ensembl/encode/supplementary/
       integration_data_jan2011/byDataType/signal/jan2011/bigwig/
       wgEncodeBroadHistoneK562CtcfStdAln_2Reps.norm5.rawsignal.bw
2  ftp.ebi.ac.uk/pub/databases/ensembl/encode/supplementary/
       integration_data_jan2011/byDataType/signal/jan2011/bigwig/
       wgEncodeBroadHistoneK562H3k27acStdAln_2Reps.norm5.rawsignal.bw
3  ftp.ebi.ac.uk/pub/databases/ensembl/encode/supplementary/
       integration_data_jan2011/byDataType/signal/jan2011/bigwig/
       wgEncodeBroadHistoneK562H3k27me3StdAln_2Reps.norm5.rawsignal.bw
4  ftp.ebi.ac.uk/pub/databases/ensembl/encode/supplementary/
       integration_data_jan2011/byDataType/signal/jan2011/bigwig/
       wgEncodeBroadHistoneK562H3k36me3StdAln_2Reps.norm5.rawsignal.bw
5  ftp.ebi.ac.uk/pub/databases/ensembl/encode/supplementary/
       integration_data_jan2011/byDataType/signal/jan2011/bigwig/
       wgEncodeBroadHistoneK562H3k4me1StdAln_2Reps.norm5.rawsignal.bw
6  ftp.ebi.ac.uk/pub/databases/ensembl/encode/supplementary/
       integration_data_jan2011/byDataType/signal/jan2011/bigwig/
       wgEncodeBroadHistoneK562H3k4me2StdAln_2Reps.norm5.rawsignal.bw
7  ftp.ebi.ac.uk/pub/databases/ensembl/encode/supplementary/
       integration_data_jan2011/byDataType/signal/jan2011/bigwig/
       wgEncodeBroadHistoneK562H3k4me3StdAln_2Reps.norm5.rawsignal.bw
8  ftp.ebi.ac.uk/pub/databases/ensembl/encode/supplementary/
       integration_data_jan2011/byDataType/signal/jan2011/bigwig/
       wgEncodeBroadHistoneK562H3k9acStdAln_2Reps.norm5.rawsignal.bw
9  ftp.ebi.ac.uk/pub/databases/ensembl/encode/supplementary/
       integration_data_jan2011/byDataType/signal/jan2011/bigwig/
       wgEncodeBroadHistoneK562H4k20me1StdAln_2Reps.norm5.rawsignal.bw
10 ftp.ebi.ac.uk/pub/databases/ensembl/encode/supplementary/
       integration_data_jan2011/byDataType/signal/jan2011/bigwig/
       wgEncodeBroadHistoneGm12878CtcfStdAln_2Reps.norm5.rawsignal.bw
11 ftp.ebi.ac.uk/pub/databases/ensembl/encode/supplementary/
       integration_data_jan2011/byDataType/signal/jan2011/bigwig/
       wgEncodeBroadHistoneGm12878H3k27acStdAln_2Reps.norm5.rawsignal.bw
12 ftp.ebi.ac.uk/pub/databases/ensembl/encode/supplementary/
       integration_data_jan2011/byDataType/signal/jan2011/bigwig/
       wgEncodeBroadHistoneGm12878H3k27me3StdAln_2Reps.norm5.rawsignal.bw
```

```
13 ftp.ebi.ac.uk/pub/databases/ensembl/encode/supplementary/
       integration_data_jan2011/byDataType/signal/jan2011/bigwig/
       wgEncodeBroadHistoneGm12878H3k36me3StdAln_2Reps.norm5.rawsignal.bw
14 ftp.ebi.ac.uk/pub/databases/ensembl/encode/supplementary/
       integration_data_jan2011/byDataType/signal/jan2011/bigwig/
       wgEncodeBroadHistoneGm12878H3k4me1StdAln_2Reps.norm5.rawsignal.bw
15 ftp.ebi.ac.uk/pub/databases/ensembl/encode/supplementary/
       integration_data_jan2011/byDataType/signal/jan2011/bigwig/
       wgEncodeBroadHistoneGm12878H3k4me2StdAln_2Reps.norm5.rawsignal.bw
16 ftp.ebi.ac.uk/pub/databases/ensembl/encode/supplementary/
       integration_data_jan2011/byDataType/signal/jan2011/bigwig/
       wgEncodeBroadHistoneGm12878H3k4me3StdAln_2Reps.norm5.rawsignal.bw
17 ftp.ebi.ac.uk/pub/databases/ensembl/encode/supplementary/
       integration_data_jan2011/byDataType/signal/jan2011/bigwig/
       wgEncodeBroadHistoneGm12878H3k9acStdAln_2Reps.norm5.rawsignal.bw
18 ftp.ebi.ac.uk/pub/databases/ensembl/encode/supplementary/
       integration_data_jan2011/byDataType/signal/jan2011/bigwig/
       wgEncodeBroadHistoneGm12878H4k20me1StdAln_2Reps.norm5.rawsignal.bw
19 ftp.ebi.ac.uk/pub/databases/ensembl/encode/supplementary/
       integration_data_jan2011/byDataType/signal/jan2011/bigwig/
       wgEncodeBroadHistoneH1hescCtcfStdAln_2Reps.norm5.rawsignal.bw
20 ftp.ebi.ac.uk/pub/databases/ensembl/encode/supplementary/
       integration_data_jan2011/byDataType/signal/jan2011/bigwig/
       wgEncodeBroadHistoneH1hescH3k27acStdAln_2Reps.norm5.rawsignal.bw
21 ftp.ebi.ac.uk/pub/databases/ensembl/encode/supplementary/
       integration_data_jan2011/byDataType/signal/jan2011/bigwig/
       wgEncodeBroadHistoneH1hescH3k27me3StdAln_2Reps.norm5.rawsignal.bw
22 ftp.ebi.ac.uk/pub/databases/ensembl/encode/supplementary/
       integration_data_jan2011/byDataType/signal/jan2011/bigwig/
       wgEncodeBroadHistoneH1hescH3k36me3StdAln_2Reps.norm5.rawsignal.bw
23 ftp.ebi.ac.uk/pub/databases/ensembl/encode/supplementary/
       integration_data_jan2011/byDataType/signal/jan2011/bigwig/
       wgEncodeBroadHistoneH1hescH3k4me1StdAln_2Reps.norm5.rawsignal.bw
24 ftp.ebi.ac.uk/pub/databases/ensembl/encode/supplementary/
       integration_data_jan2011/byDataType/signal/jan2011/bigwig/
       wgEncodeBroadHistoneH1hescH3k4me2StdAln_2Reps.norm5.rawsignal.bw
25 ftp.ebi.ac.uk/pub/databases/ensembl/encode/supplementary/
       integration_data_jan2011/byDataType/signal/jan2011/bigwig/
       wgEncodeBroadHistoneH1hescH3k4me3StdAln_2Reps.norm5.rawsignal.bw
26 ftp.ebi.ac.uk/pub/databases/ensembl/encode/supplementary/
       integration_data_jan2011/byDataType/signal/jan2011/bigwig/
       wgEncodeBroadHistoneH1hescH3k9acStdAln_2Reps.norm5.rawsignal.bw
27 ftp.ebi.ac.uk/pub/databases/ensembl/encode/supplementary/
       integration_data_jan2011/byDataType/signal/jan2011/bigwig/
       wgEncodeBroadHistoneH1hescH4k20me1StdAln_2Reps.norm5.rawsignal.bw
28 ftp.ebi.ac.uk/pub/databases/ensembl/encode/supplementary/
       integration_data_jan2011/byDataType/signal/jan2011/bigwig/
       wgEncodeBroadHistoneHelas3CtcfStdAln_2Reps.norm5.rawsignal.bw
29 ftp.ebi.ac.uk/pub/databases/ensembl/encode/supplementary/
       integration_data_jan2011/byDataType/signal/jan2011/bigwig/
       wgEncodeBroadHistoneHelas3H3k27acStdAln_2Reps.norm5.rawsignal.bw
```

```
30 ftp.ebi.ac.uk/pub/databases/ensembl/encode/supplementary/
       integration_data_jan2011/byDataType/signal/jan2011/bigwig/
       wgEncodeBroadHistoneHelas3H3k27me3StdAln_2Reps.norm5.rawsignal.bw
31 ftp.ebi.ac.uk/pub/databases/ensembl/encode/supplementary/
       integration_data_jan2011/byDataType/signal/jan2011/bigwig/
       wgEncodeBroadHistoneHelas3H3k36me3StdAln_2Reps.norm5.rawsignal.bw
32 ftp.ebi.ac.uk/pub/databases/ensembl/encode/supplementary/
       integration_data_jan2011/byDataType/signal/jan2011/bigwig/
       wgEncodeBroadHistoneHelas3H3k4me1Std_1Reps.norm5.rawsignal.bw
33 ftp.ebi.ac.uk/pub/databases/ensembl/encode/supplementary/
       integration_data_jan2011/byDataType/signal/jan2011/bigwig/
       wgEncodeBroadHistoneHelas3H3k4me2StdAln_2Reps.norm5.rawsignal.bw
34 ftp.ebi.ac.uk/pub/databases/ensembl/encode/supplementary/
       integration_data_jan2011/byDataType/signal/jan2011/bigwig/
       wgEncodeBroadHistoneHelas3H3k4me3StdAln_2Reps.norm5.rawsignal.bw
35 ftp.ebi.ac.uk/pub/databases/ensembl/encode/supplementary/
       integration_data_jan2011/byDataType/signal/jan2011/bigwig/
       wgEncodeBroadHistoneHelas3H3k9acStdAln_2Reps.norm5.rawsignal.bw
36 ftp.ebi.ac.uk/pub/databases/ensembl/encode/supplementary/
       integration_data_jan2011/byDataType/signal/jan2011/bigwig/
       wgEncodeBroadHistoneHelas3H4k20me1StdAln_2Reps.norm5.rawsignal.bw
37 ftp.ebi.ac.uk/pub/databases/ensembl/encode/supplementary/
       integration_data_jan2011/byDataType/signal/jan2011/bigwig/
       wgEncodeBroadHistoneHepg2CtcfStdAln_2Reps.norm5.rawsignal.bw
38 ftp.ebi.ac.uk/pub/databases/ensembl/encode/supplementary/
       integration_data_jan2011/byDataType/signal/jan2011/bigwig/
       wgEncodeBroadHistoneHepg2H3k27acStdAln_2Reps.norm5.rawsignal.bw
39 ftp.ebi.ac.uk/pub/databases/ensembl/encode/supplementary/
       integration_data_jan2011/byDataType/signal/jan2011/bigwig/
       wgEncodeBroadHistoneHepg2H3k27me3StdAln_2Reps.norm5.rawsignal.bw
40 ftp.ebi.ac.uk/pub/databases/ensembl/encode/supplementary/
       integration_data_jan2011/byDataType/signal/jan2011/bigwig/
       wgEncodeBroadHistoneHepg2H3k36me3StdAln_2Reps.norm5.rawsignal.bw
41 ftp.ebi.ac.uk/pub/databases/ensembl/encode/supplementary/
       integration_data_jan2011/byDataType/signal/jan2011/bigwig/
       wgEncodeBroadHistoneHepg2H3k4me1Std_1Reps.norm5.rawsignal.bw
42 ftp.ebi.ac.uk/pub/databases/ensembl/encode/supplementary/
       integration_data_jan2011/byDataType/signal/jan2011/bigwig/
       wgEncodeBroadHistoneHepg2H3k4me2StdAln_2Reps.norm5.rawsignal.bw
43 ftp.ebi.ac.uk/pub/databases/ensembl/encode/supplementary/
       integration_data_jan2011/byDataType/signal/jan2011/bigwig/
       wgEncodeBroadHistoneHepg2H3k4me3StdAln_2Reps.norm5.rawsignal.bw
44 ftp.ebi.ac.uk/pub/databases/ensembl/encode/supplementary/
       integration_data_jan2011/byDataType/signal/jan2011/bigwig/
       wgEncodeBroadHistoneHepg2H3k9acStdAln_2Reps.norm5.rawsignal.bw
45 ftp.ebi.ac.uk/pub/databases/ensembl/encode/supplementary/
       integration_data_jan2011/byDataType/signal/jan2011/bigwig/
       wgEncodeBroadHistoneHepg2H4k20me1StdAln_2Reps.norm5.rawsignal.bw
46 ftp.ebi.ac.uk/pub/databases/ensembl/encode/supplementary/
       integration_data_jan2011/byDataType/signal/jan2011/bigwig/
       wgEncodeBroadHistoneHuvecCtcfStdAln_3Reps.norm5.rawsignal.bw
```

```
47 ftp.ebi.ac.uk/pub/databases/ensembl/encode/supplementary/
       integration_data_jan2011/byDataType/signal/jan2011/bigwig/
       wgEncodeBroadHistoneHuvecH3k27acStdAln_3Reps.norm5.rawsignal.bw
48 ftp.ebi.ac.uk/pub/databases/ensembl/encode/supplementary/
       integration_data_jan2011/byDataType/signal/jan2011/bigwig/
       wgEncodeBroadHistoneHuvecH3k27me3StdAln_2Reps.norm5.rawsignal.bw
49 ftp.ebi.ac.uk/pub/databases/ensembl/encode/supplementary/
       integration_data_jan2011/byDataType/signal/jan2011/bigwig/
       wgEncodeBroadHistoneHuvecH3k36me3StdAln_3Reps.norm5.rawsignal.bw
50 ftp.ebi.ac.uk/pub/databases/ensembl/encode/supplementary/
       integration_data_jan2011/byDataType/signal/jan2011/bigwig/
       wgEncodeBroadHistoneHuvecH3k4me1StdAln_3Reps.norm5.rawsignal.bw
51 ftp.ebi.ac.uk/pub/databases/ensembl/encode/supplementary/
       integration_data_jan2011/byDataType/signal/jan2011/bigwig/
       wgEncodeBroadHistoneHuvecH3k4me2StdAln_2Reps.norm5.rawsignal.bw
52 ftp.ebi.ac.uk/pub/databases/ensembl/encode/supplementary/
       integration_data_jan2011/byDataType/signal/jan2011/bigwig/
       wgEncodeBroadHistoneHuvecH3k4me3StdAln_3Reps.norm5.rawsignal.bw
53 ftp.ebi.ac.uk/pub/databases/ensembl/encode/supplementary/
       integration_data_jan2011/byDataType/signal/jan2011/bigwig/
       wgEncodeBroadHistoneHuvecH3k9acStdAln_3Reps.norm5.rawsignal.bw
54 ftp.ebi.ac.uk/pub/databases/ensembl/encode/supplementary/
       integration_data_jan2011/byDataType/signal/jan2011/bigwig/
       wgEncodeBroadHistoneHuvecH4k20me1StdAln_3Reps.norm5.rawsignal.bw
55 ftp.ebi.ac.uk/pub/databases/ensembl/encode/supplementary/
       integration_data_jan2011/byDataType/signal/jan2011/bigwig/
       wgEncodeBroadHistoneHsmmCtcfStdAln_2Reps.norm5.rawsignal.bw
56 ftp.ebi.ac.uk/pub/databases/ensembl/encode/supplementary/
       integration_data_jan2011/byDataType/signal/jan2011/bigwig/
       wgEncodeBroadHistoneHsmmH3k27acStdAln_2Reps.norm5.rawsignal.bw
57 ftp.ebi.ac.uk/pub/databases/ensembl/encode/supplementary/
       integration_data_jan2011/byDataType/signal/jan2011/bigwig/
       wgEncodeBroadHistoneHsmmH3k27me3StdAln_2Reps.norm5.rawsignal.bw
58 ftp.ebi.ac.uk/pub/databases/ensembl/encode/supplementary/
       integration_data_jan2011/byDataType/signal/jan2011/bigwig/
       wgEncodeBroadHistoneHsmmH3k36me3StdAln_2Reps.norm5.rawsignal.bw
59 ftp.ebi.ac.uk/pub/databases/ensembl/encode/supplementary/
       integration_data_jan2011/byDataType/signal/jan2011/bigwig/
       wgEncodeBroadHistoneHsmmH3k4me1StdAln_2Reps.norm5.rawsignal.bw
60 ftp.ebi.ac.uk/pub/databases/ensembl/encode/supplementary/
       integration_data_jan2011/byDataType/signal/jan2011/bigwig/
       wgEncodeBroadHistoneHsmmH3k4me2StdAln_2Reps.norm5.rawsignal.bw
61 ftp.ebi.ac.uk/pub/databases/ensembl/encode/supplementary/
       integration_data_jan2011/byDataType/signal/jan2011/bigwig/
       wgEncodeBroadHistoneHsmmH3k4me3StdAln_2Reps.norm5.rawsignal.bw
62 ftp.ebi.ac.uk/pub/databases/ensembl/encode/supplementary/
       integration_data_jan2011/byDataType/signal/jan2011/bigwig/
       wgEncodeBroadHistoneHsmmH3k9acStdAln_2Reps.norm5.rawsignal.bw
63 ftp.ebi.ac.uk/pub/databases/ensembl/encode/supplementary/
       integration_data_jan2011/byDataType/signal/jan2011/bigwig/
       wgEncodeBroadHistoneHsmmH4k20me1StdAln_2Reps.norm5.rawsignal.bw
```

```
64 ftp.ebi.ac.uk/pub/databases/ensembl/encode/supplementary/
       integration_data_jan2011/byDataType/signal/jan2011/bigwig/
       wgEncodeBroadHistoneNhekCtcfStdAln_3Reps.norm5.rawsignal.bw
65 ftp.ebi.ac.uk/pub/databases/ensembl/encode/supplementary/
       integration_data_jan2011/byDataType/signal/jan2011/bigwig/
       wgEncodeBroadHistoneNhekH3k27acStdAln_3Reps.norm5.rawsignal.bw
66 ftp.ebi.ac.uk/pub/databases/ensembl/encode/supplementary/
       integration_data_jan2011/byDataType/signal/jan2011/bigwig/
       wgEncodeBroadHistoneNhekH3k27me3StdAln_3Reps.norm5.rawsignal.bw
67 ftp.ebi.ac.uk/pub/databases/ensembl/encode/supplementary/
       integration_data_jan2011/byDataType/signal/jan2011/bigwig/
       wgEncodeBroadHistoneNhekH3k36me3StdAln_3Reps.norm5.rawsignal.bw
68 ftp.ebi.ac.uk/pub/databases/ensembl/encode/supplementary/
       integration_data_jan2011/byDataType/signal/jan2011/bigwig/
       wgEncodeBroadHistoneNhekH3k4me1StdAln_3Reps.norm5.rawsignal.bw
69 ftp.ebi.ac.uk/pub/databases/ensembl/encode/supplementary/
       integration_data_jan2011/byDataType/signal/jan2011/bigwig/
       wgEncodeBroadHistoneNhekH3k4me2StdAln_3Reps.norm5.rawsignal.bw
70 ftp.ebi.ac.uk/pub/databases/ensembl/encode/supplementary/
       integration_data_jan2011/byDataType/signal/jan2011/bigwig/
       wgEncodeBroadHistoneNhekH3k4me3StdAln_3Reps.norm5.rawsignal.bw
71 ftp.ebi.ac.uk/pub/databases/ensembl/encode/supplementary/
       integration_data_jan2011/byDataType/signal/jan2011/bigwig/
       wgEncodeBroadHistoneNhekH3k9acStdAln_3Reps.norm5.rawsignal.bw
72 ftp.ebi.ac.uk/pub/databases/ensembl/encode/supplementary/
       integration_data_jan2011/byDataType/signal/jan2011/bigwig/
       wgEncodeBroadHistoneNhekH4k20me1StdAln_3Reps.norm5.rawsignal.bw
73 ftp.ebi.ac.uk/pub/databases/ensembl/encode/supplementary/
       integration_data_jan2011/byDataType/signal/jan2011/bigwig/
       wgEncodeBroadHistoneHmecCtcfStdAln_2Reps.norm5.rawsignal.bw
74 ftp.ebi.ac.uk/pub/databases/ensembl/encode/supplementary/
       integration_data_jan2011/byDataType/signal/jan2011/bigwig/
       wgEncodeBroadHistoneHmecH3k27acStdAln_2Reps.norm5.rawsignal.bw
75 ftp.ebi.ac.uk/pub/databases/ensembl/encode/supplementary/
       integration_data_jan2011/byDataType/signal/jan2011/bigwig/
       wgEncodeBroadHistoneHmecH3k27me3StdAln_2Reps.norm5.rawsignal.bw
76 ftp.ebi.ac.uk/pub/databases/ensembl/encode/supplementary/
       integration_data_jan2011/byDataType/signal/jan2011/bigwig/
       wgEncodeBroadHistoneHmecH3k36me3StdAln_2Reps.norm5.rawsignal.bw
77 ftp.ebi.ac.uk/pub/databases/ensembl/encode/supplementary/
       integration_data_jan2011/byDataType/signal/jan2011/bigwig/
       wgEncodeBroadHistoneHmecH3k4me1StdAln_2Reps.norm5.rawsignal.bw
78 ftp.ebi.ac.uk/pub/databases/ensembl/encode/supplementary/
       integration_data_jan2011/byDataType/signal/jan2011/bigwig/
       wgEncodeBroadHistoneHmecH3k4me2StdAln_2Reps.norm5.rawsignal.bw
79 ftp.ebi.ac.uk/pub/databases/ensembl/encode/supplementary/
       integration_data_jan2011/byDataType/signal/jan2011/bigwig/
       wgEncodeBroadHistoneHmecH3k4me3StdAln_2Reps.norm5.rawsignal.bw
80 ftp.ebi.ac.uk/pub/databases/ensembl/encode/supplementary/
       integration_data_jan2011/byDataType/signal/jan2011/bigwig/
       wgEncodeBroadHistoneHmecH3k9acStdAln_2Reps.norm5.rawsignal.bw
```

```
81 ftp.ebi.ac.uk/pub/databases/ensembl/encode/supplementary/
       integration_data_jan2011/byDataType/signal/jan2011/bigwig/
       wgEncodeBroadHistoneHmecH4k20me1StdAln_2Reps.norm5.rawsignal.bw
82 ftp.ebi.ac.uk/pub/databases/ensembl/encode/supplementary/
       integration_data_jan2011/byDataType/signal/jan2011/bigwig/
       wgEncodeBroadHistoneNhlfCtcfStdAln_3Reps.norm5.rawsignal.bw
83 ftp.ebi.ac.uk/pub/databases/ensembl/encode/supplementary/
       integration_data_jan2011/byDataType/signal/jan2011/bigwig/
       wgEncodeBroadHistoneNhlfH3k27acStdAln_3Reps.norm5.rawsignal.bw
84 ftp.ebi.ac.uk/pub/databases/ensembl/encode/supplementary/
       integration_data_jan2011/byDataType/signal/jan2011/bigwig/
       wgEncodeBroadHistoneNhlfH3k27me3StdAln_2Reps.norm5.rawsignal.bw
85 ftp.ebi.ac.uk/pub/databases/ensembl/encode/supplementary/
       integration_data_jan2011/byDataType/signal/jan2011/bigwig/
       wgEncodeBroadHistoneNhlfH3k36me3StdAln_3Reps.norm5.rawsignal.bw
86 ftp.ebi.ac.uk/pub/databases/ensembl/encode/supplementary/
       integration_data_jan2011/byDataType/signal/jan2011/bigwig/
       wgEncodeBroadHistoneNhlfH3k4me1StdAln_3Reps.norm5.rawsignal.bw
87 ftp.ebi.ac.uk/pub/databases/ensembl/encode/supplementary/
       integration_data_jan2011/byDataType/signal/jan2011/bigwig/
       wgEncodeBroadHistoneNhlfH3k4me2StdAln_2Reps.norm5.rawsignal.bw
88 ftp.ebi.ac.uk/pub/databases/ensembl/encode/supplementary/
       integration_data_jan2011/byDataType/signal/jan2011/bigwig/
       wgEncodeBroadHistoneNhlfH3k4me3StdAln_3Reps.norm5.rawsignal.bw
89 ftp.ebi.ac.uk/pub/databases/ensembl/encode/supplementary/
       integration_data_jan2011/byDataType/signal/jan2011/bigwig/
       wgEncodeBroadHistoneNhlfH3k9acStdAln_3Reps.norm5.rawsignal.bw
90 ftp.ebi.ac.uk/pub/databases/ensembl/encode/supplementary/
       integration_data_jan2011/byDataType/signal/jan2011/bigwig/
       wgEncodeBroadHistoneNhlfH4k20me1StdAln_3Reps.norm5.rawsignal.bw
91 ftp.ebi.ac.uk/pub/databases/ensembl/encode/supplementary/
       integration_data_jan2011/byDataType/signal/jan2011/bigwig/
       wgEncodeBroadHistoneNhekControlStdAln_2Reps.norm5.rawsignal.bw
92 ftp.ebi.ac.uk/pub/databases/ensembl/encode/supplementary/
       integration_data_jan2011/byDataType/signal/jan2011/bigwig/
       wgEncodeBroadHistoneK562ControlStdAln_1Reps.norm5.rawsignal.bw
93 ftp.ebi.ac.uk/pub/databases/ensembl/encode/supplementary/
       integration_data_jan2011/byDataType/signal/jan2011/bigwig/
       wgEncodeBroadHistoneNhlfControlStdAln_2Reps.norm5.rawsignal.bw
94 ftp.ebi.ac.uk/pub/databases/ensembl/encode/supplementary/
       integration_data_jan2011/byDataType/signal/jan2011/bigwig/
       wgEncodeBroadHistoneHmecControlStdAln_1Reps.norm5.rawsignal.bw
95 ftp.ebi.ac.uk/pub/databases/ensembl/encode/supplementary/
       integration_data_jan2011/byDataType/signal/jan2011/bigwig/
       wgEncodeBroadHistoneNhekControlStdAln_2Reps.norm5.rawsignal.bw
96 ftp.ebi.ac.uk/pub/databases/ensembl/encode/supplementary/
       integration_data_jan2011/byDataType/signal/jan2011/bigwig/
       wgEncodeBroadHistoneHsmmControlStdAln_1Reps.norm5.rawsignal.bw
97 ftp.ebi.ac.uk/pub/databases/ensembl/encode/supplementary/
       integration_data_jan2011/byDataType/signal/jan2011/bigwig/
       wgEncodeBroadHistoneHuvecControlStdAln_2Reps.norm5.rawsignal.bw
```

```
 98 ftp.ebi.ac.uk/pub/databases/ensembl/encode/supplementary/
      integration_data_jan2011/byDataType/signal/jan2011/bigwig/
      wgEncodeBroadHistoneHepg2ControlStdAln_1Reps.norm5.rawsignal.bw
 99 ftp.ebi.ac.uk/pub/databases/ensembl/encode/supplementary/
      integration_data_jan2011/byDataType/signal/jan2011/bigwig/
      wgEncodeBroadHistoneHelas3ControlStdAln_2Reps.norm5.rawsignal.bw
100 ftp.ebi.ac.uk/pub/databases/ensembl/encode/supplementary/
      integration_data_jan2011/byDataType/signal/jan2011/bigwig/
      wgEncodeBroadHistoneGm12878ControlStdAln_1Reps.norm5.rawsignal.bw
```

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., ... Devin, M. (2016). Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*.

Airy, G. B. (1861). *On the algebraical and numerical theory of errors of observations and the combination of observations*. Macmillan&Company.

Akbarian, S., Liu, C., Knowles, J. A., Vaccarino, F. M., Farnham, P. J., Crawford, G. E., ... Geschwind, D. H. (2015). The psychencode project. *Nature neuroscience*, *18*(12), 1707–1712.

Andersson, R., Gebhard, C., Miguel-Escalada, I., Hoof, I., Bornholdt, J., Boyd, M., ... Suzuki, T. (2014). An atlas of active enhancers across human cell types and tissues. *Nature*, *507*(7493), 455–461.

Arnold, C. D., Gerlach, D., Stelzer, C., Boryń, Ł. M., Rath, M., & Stark, A. (2013). Genome-wide quantitative enhancer activity maps identified by starr-seq. *Science*, *339*(6123), 1074–1077.

Athreya, K. B., & Lahiri, S. N. (2006). *Measure theory and probability theory*. Springer Science & Business Media.

Bach, F. R., & Jordan, M. I. (2005). *A probabilistic interpretation of canonical correlation analysis* (techreport No. 688). Berkeley: Department of Statistics, University of California, Berkeley.

Banerjee, B., & Sherwood, R. I. (2017). A crispr view of gene regulation. *Current Opinion in Systems Biology*.

Banerjee, S., Carlin, B. P., & Gelfand, A. E. (2014). *Hierarchical modeling and analysis for spatial data*. Crc Press.

Barski, A., Cuddapah, S., Cui, K., Roh, T.-Y., Schones, D. E., Wang, Z., ... Zhao, K. (2007). High-resolution profiling of histone methylations in the human genome. *Cell*, *129*(4), 823–837.

Bartholomew, D. J., Knott, M., & Moustaki, I. (2011). *Latent variable models and factor analysis: A unified approach* (Vol. 904). John Wiley & Sons.

Bengio, Y., Courville, A., & Vincent, P. (2013). Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, *35*(8), 1798–1828.

Birnbaum, A. (1968). *Some latent trait models and their use in inferring an examinee's ability.*

Bishop, C. M., Svensén, M., & Williams, C. K. (1998). Gtm: the generative topographic mapping. *Neural computation*, *10*(1), 215–234.

Blei, D. M., Kucukelbir, A., & McAuliffe, J. D. (2016). Variational inference: A review for statisticians. *Journal of the American Statistical Association (to appear). arXiv preprint arXiv:1601.00670*.

Bollen, K. A. (2002). Latent variables in psychology and the social sciences. *Annual review of psychology*, *53*(1), 605–634.

Bonasio, R., Tu, S., & Reinberg, D. (2010). Molecular signals of epigenetic states. *science*, *330*(6004), 612–616.

Bonn, S., Zinzen, R. P., Girardot, C., Gustafson, E. H., Perez-Gonzalez, A., Delhomme, N., ... Furlong, E. E. (2012). Tissue-specific analysis of chromatin state identifies temporal signatures of enhancer activity during embryonic development. *Nature genetics*, *44*(2), 148–156.

Browne, M. W. (1979). The maximum-likelihood solution in inter-battery factor analysis. *British Journal of Mathematical and Statistical Psychology*, *32*(1), 75–86.

Browne, M. W. (1980). Factor analysis of multiple batteries by maximum likelihood. *British Journal of Mathematical and Statistical Psychology*, *33*(2), 184–199.

Canver, M. C., Smith, E. C., Sher, F., Pinello, L., Sanjana, N. E., Shalem, O., ... others (2015). Bcl11a enhancer dissection by cas9-mediated in situ saturating mutagenesis. *Nature*.

Carlberg, C., & Molnár, F. (2014). *Mechanisms of gene regulation*. Springer.

Chen, S., Sanjana, N. E., Zheng, K., Shalem, O., Lee, K., Shi, X., ... others (2015). Genome-wide crispr screen in a mouse model of tumor growth and metastasis. *Cell*, *160*(6), 1246–1260.

Chollet, F. (2015). *Keras.* `https://github.com/fchollet/keras`. GitHub.

Chung, J., Kastner, K., Dinh, L., Goel, K., Courville, A. C., & Bengio, Y. (2015). A recurrent latent variable model for sequential data. In *Advances in neural information processing systems* (pp. 2980–2988).

Consortium, E. P. (2012). An integrated encyclopedia of dna elements in the human genome. *Nature*, *489*(7414), 57–74.

Cudeck, R. (1982). Methods for estimating between-battery factors. *Multivariate Behavioral Research*, *17*(1), 47–68.

Dale, R. K., Pedersen, B. S., & Quinlan, A. R. (2011). Pybedtools: a flexible python library for manipulating genomic datasets and annotations. *Bioinformatics*, *27*(24), 3423–3424.

Diggle, P. J., & Gratton, R. J. (1984). Monte carlo methods of inference for implicit statistical models. *Journal of the Royal Statistical Society. Series B (Methodological)*, 193–227.

Doersch, C. (2016). Tutorial on variational autoencoders. *arXiv preprint arXiv:1606.05908*.

Durrett, R. (2010). *Probability: theory and examples*. Cambridge university press.

Ernst, J., & Kellis, M. (2012). *ChromHMM: automating chromatin-state discovery and characterization* (Vol. 9) (No. 3). doi: 10.1038/nmeth.1906

Ernst, J., Kheradpour, P., Mikkelsen, T. S., Shoresh, N., Ward, L. D., Epstein, C. B., ... Coyne, M. (2011). Mapping and analysis of chromatin state dynamics in nine human cell types. *Nature*, *473*(7345), 43–49.

Erwin, G. D., Oksenberg, N., Truty, R. M., Kostka, D., Murphy, K. K., Ahituv, N., ... Capra, J. A. (2014). Integrating diverse datasets improves developmental enhancer prediction. *PLoS Comput Biol*, *10*(6), e1003677.

Fulco, C. P., Munschauer, M., Anyoha, R., Munson, G., Grossman, S. R., Perez, E. M., ... Engreitz, J. M. (2016). Systematic mapping of functional enhancer–promoter connections with crispr interference. *Science*, *354*(6313), 769–773.

Gardiner-Garden, M., & Frommer, M. (1987). Cpg islands in vertebrate genomes. *Journal of molecular biology*, *196*(2), 261–282.

Gelman, A., Carlin, J. B., Stern, H. S., & Rubin, D. B. (2014). *Bayesian data analysis* (Vol. 2). Chapman & Hall/CRC Boca Raton, FL, USA.

Gemici, M., Hung, C.-C., Santoro, A., Wayne, G., Mohamed, S., Rezende, D. J., ... Lillicrap, T. (2017). Generative temporal models with memory. *arXiv preprint arXiv:1702.04649*.

Ghandi, M., Lee, D., Mohammad-Noori, M., & Beer, M. A. (2014). Enhanced regulatory sequence prediction using gapped k-mer features. *PLoS computational biology*, *10*(7), e1003711.

Gjuvsland, A. B., Vik, J. O., Beard, D. A., Hunter, P. J., & Omholt, S. W. (2013). Bridging the genotype–phenotype gap: what does it take? *The Journal of physiology*, *591*(8), 2055–2066.

Glorot, X., & Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In *Aistats* (Vol. 9, pp. 249–256).

Gómez, E., Gomez-Viilegas, M., & Marin, J. (1998). A multivariate generalization of the power exponential family of distributions. *Communications in Statistics-Theory and Methods*, *27*(3), 589–600.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT Press.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., ... Bengio, Y. (2014). Generative adversarial nets. In *Advances in neural information processing systems* (pp. 2672–2680).

Gould, S. J. (1996). *The mismeasure of man*. WW Norton & Company.

Hägglund, G. (2001). Milestones in the history of factor analysis. In R. Cudeck, K. G. Jöreskog, & D. Sörbom (Eds.), *Structural equation modeling, present and future: A festschrift in honor of karl jöreskog* (pp. 11–38). Lincolnwood, IL: Scientific Software International.

Harrow, J., Frankish, A., Gonzalez, J. M., Tapanari, E., Diekhans, M., Kokocinski, F., ... Searle, S. (2012). Gencode: the reference human genome annotation for the encode project. *Genome research*, *22*(9), 1760–1774.

Heintzman, N. D., Hon, G. C., Hawkins, R. D., Kheradpour, P., Stark, A., Harp, L. F., ... Ching, C. W. (2009). Histone modifications at human enhancers reflect global cell-type-specific gene expression. *Nature*, *459*(7243), 108–112.

Hilborn, R., & Mangel, M. (1997). *The ecological detective: confronting models with data* (Vol. 28). Princeton University Press.

Hoffman, M. M., Buske, O. J., Wang, J., Weng, Z., Bilmes, J. a., & Noble, W. S. (2012). Unsupervised pattern discovery in human chromatin structure through genomic segmentation. *Nature methods*, *9*(5), 473–6. doi: 10.1038/nmeth.1937

Hoffman, M. M., Ernst, J., Wilder, S. P., Kundaje, A., Harris, R. S., Libbrecht, M., ... Birney, E. (2012). Integrative annotation of chromatin elements from encode data. *Nucleic acids research*, gks1284.

Hotelling, H. (1933). Analysis of a complex of statistical variables into principal components. *Journal of Educational Psychology*, *24*(6), 417–441. doi: 10.1037/h0071325

Hotelling, H. (1936, dec). Relations Between Two Sets of Variates. *Biometrika*, *28*(3/4), 321. doi: 10.2307/2333955

Hwang, H., & Takane, Y. (2004). Generalized structured component analysis. *Psychometrika*, *69*(1), 81–99.

Hwang, H., & Takane, Y. (2014). *Generalized structured component analysis: A component-based approach to structural equation modeling.* CRC Press.

Hyvärinen, A. (2015). A unified probabilistic model for independent and principal component analysis. *Advances in Independent Component Analysis and Learning Machines*, 1–9.

Izenman, A. J. (1975). Reduced-rank regression for the multivariate linear model. *Journal of multivariate analysis*, *5*(2), 248–264.

Izenman, A. J. (2008). *Modern multivariate statistical techniques*. Springer.

Jaynes, E. T. (2003). *Probability theory: The logic of science*. Cambridge university press.

Jebara, T. (2012). *Machine learning: discriminative and generative* (Vol. 755). Springer Science & Business Media.

Jenuwein, T., & Allis, C. D. (2001). Translating the histone code. *Science*, *293*(5532), 1074–1080.

Johnson, D. S., Mortazavi, A., Myers, R. M., & Wold, B. (2007). Genome-wide mapping of in vivo protein-dna interactions. *Science*, *316*(5830), 1497–1502.

Jöreskog, K. G. (1970). A general method for analysis of covariance structures. *Biometrika*, *57*(2), 239–251. doi: 10.1093/biomet/57.2.239

Kant, I., & Guyer, P. (1998). *Critique of pure reason*. Cambridge University Press.

Kent, W. J., Sugnet, C. W., Furey, T. S., Roskin, K. M., Pringle, T. H., Zahler, A. M., & Haussler, D. (2002). The human genome browser at ucsc. *Genome research*, *12*(6), 996–1006.

Kingma, D., & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kingma, D. P., Salimans, T., & Welling, M. (2016). Improving variational inference with inverse autoregressive flow. *arXiv preprint arXiv:1606.04934v2*.

Kingma, D. P., & Welling, M. (2013, dec). Auto-Encoding Variational Bayes. *ICLR*(Ml), 1–14.

Klami, A., Virtanen, S., Leppäaho, E., & Kaski, S. (2015). Group factor analysis. *IEEE transactions on neural networks and learning systems*, *26*(9), 2136–2147.

Koch, F., Fenouil, R., Gut, M., Cauchy, P., Albert, T. K., Zacarias-Cabeza, J., ... others (2011). Transcription initiation platforms and gtf recruitment at tissue-specific enhancers and promoters. *Nature structural & molecular biology*, *18*(8), 956–963.

Korkmaz, G., Lopes, R., Ugalde, A. P., Nevedomskaya, E., Han, R., Myacheva, K., ... Agami, R. (2016). Functional genetic screens for enhancer elements in the human genome using crispr-cas9. *Nature biotechnology*, *34*(2), 192–198.

Kundaje, A. (n.d.). *Wiggler kernel description.* `http://github.com/akundaje/align2rawsignal`. (Accessed: 2017-03-11)

Kundaje, A., Meuleman, W., Ernst, J., Bilenky, M., Yen, A., Heravi-Moussavi, A., ... others (2015). Integrative analysis of 111 reference human epigenomes. *Nature*, *518*(7539), 317–330.

Lake, B. M., Salakhutdinov, R., & Tenenbaum, J. B. (2015). Human-level concept learning through probabilistic program induction. *Science*, *350*(6266), 1332–1338.

Lanctôt, C., Cheutin, T., Cremer, M., Cavalli, G., & Cremer, T. (2007). Dynamic genome architecture in the nuclear space: regulation of gene expression in three dimensions. *Nature Reviews Genetics*, *8*(2), 104–115.

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

Latchman, D. (2015). *Gene regulation*. Taylor & Francis.

Lauritzen, S. L. (1996). *Graphical models* (Vol. 17). Clarendon Press.

Lawrence, M., Huber, W., Pagès, H., Aboyoun, P., Carlson, M., Gentleman, R., ... Carey, V. (2013). Software for computing and annotating genomic ranges. *PLoS Computational Biology*, *9*. doi: 10.1371/journal.pcbi.1003118

Lazarsfeld, P. F. (1950). The logical and mathematical foundation of latent structure analysis. *Measurement and prediction*, *4*, 362–412.

Leung, M. K. K., Delong, A., Alipanahi, B., & Frey, B. J. (2016). Machine learning in genomic medicine: A review of computational problems and data sets. *Proceedings of the IEEE*, *104*(1), 176–197. doi: 10.1109/JPROC.2015.2494198

Li, W., Notani, D., & Rosenfeld, M. G. (2016). Enhancers as non-coding rna transcription units: recent insights and future perspectives. *Nature Reviews Genetics*, *17*(4), 207–223.

Lord, F. M. (1952). The relation of test score to the trait underlying the test. *ETS Research Report Series*, *1952*(2), 517–549.

Maaløe, L., Sønderby, C. K., Sønderby, S. K., & Winther, O. (2016). Auxiliary deep generative models. *arXiv preprint arXiv:1602.05473*.

MacKay, D. J. (1995a). Bayesian neural networks and density networks. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, *354*(1), 73–80.

MacKay, D. J. (1995b). Probable networks and plausible predictions - a review of practical bayesian methods for supervised neural networks. *Network: Computation in Neural Systems*, *6*(3), 469–505.

Mortazavi, A., Pepke, S., Jansen, C., Marinov, G. K., Ernst, J., Kellis, M., ... Wold, B. J. (2013). Integrating and mining the chromatin landscape of cell-type specificity using self-organizing maps. *Genome research*, *23*(12), 2136–2148.

Muthén, B. O. (2002). Beyond sem: General latent variable modeling. *Behaviormetrika*, *29*(1), 81–117.

Narayanan, H., & Mitter, S. (2010). Sample complexity of testing the manifold hypothesis. In *Advances in neural information processing systems* (pp. 1786–1794).

Natoli, G., & Andrau, J.-C. (2012). Noncoding transcription at enhancers: general principles and functional models. *Annual review of genetics*, *46*, 1–19.

Omholt, S. W. (2013). From sequence to consequence and back. *Progress in biophysics and molecular biology*, *111*(2), 75–82.

Ong, C.-T., & Corces, V. G. (2014). Ctcf: an architectural protein bridging genome topology and function. *Nature reviews Genetics*, *15*(4), 234–246.

Pearl, J. (2014). *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann.

Pearson, K. (1901, nov). LIII. On lines and planes of closest fit to systems of points in space. *Philosophical Magazine Series 6*, *2*(11), 559–572. doi: 10.1080/14786440109462720

Pekowska, A., Benoukraf, T., Zacarias-Cabeza, J., Belhocine, M., Koch, F., Holota, H., ... Spicuglia, S. (2011). H3k4 tri-methylation provides an epigenetic signature of active enhancers. *The EMBO journal*, *30*(20), 4198–4210.

Poldrack, R. A., Mumford, J. A., & Nichols, T. E. (2011). *Handbook of functional mri data analysis*. Cambridge University Press.

Pritchard, J. K., Seielstad, M. T., Perez-Lezaun, A., & Feldman, M. W. (1999). Population growth of human y chromosomes: a study of y chromosome microsatellites. *Molecular biology and evolution*, *16*(12), 1791–1798.

Quinlan, A. R., & Hall, I. M. (2010). Bedtools: a flexible suite of utilities for comparing genomic features. *Bioinformatics*, *26*(6), 841–842.

R Core Team. (2016). R: A language and environment for statistical computing [Computer software manual]. Vienna, Austria. Retrieved from `https://www.R-project.org/`

Rabe-Hesketh, S., Skrondal, A., & Pickles, A. (2004). Generalized multilevel structural equation modeling. *Psychometrika*, *69*(2), 167–190.

Rajagopal, N., Xie, W., Li, Y., Wagner, U., Wang, W., Stamatoyannopoulos, J., ... Ren, B. (2013). Rfecs: a random-forest based algorithm for enhancer identification from chromatin state. *PLoS Comput Biol*, *9*(3), e1002968.

Rajarajan, P., Espeso Gil, S., Brennand, K. J., & Akbarian, S. (2016). Spatial genome organization and cognition. *Nature Publishing Group*, *17*. doi: 10.1038/nrn.2016.124

Ramírez, F., Ryan, D. P., Grüning, B., Bhardwaj, V., Kilpert, F., Richter, A. S., ... Manke, T. (2016). deeptools2: a next generation web server for deep-sequencing data analysis. *Nucleic acids research*, gkw257.

Ranganath, R., Tang, L., Charlin, L., & Blei, D. M. (2015). Deep exponential families. In *Aistats*.

Rezende, D. J., Mohamed, S., Danihelka, I., Gregor, K., & Wierstra, D. (2016). One-shot generalization in deep generative models. *arXiv preprint arXiv:1603.05106*.

Rezende, D. J., Mohamed, S., & Wierstra, D. (2014). Stochastic backpropagation and approximate inference in deep generative models. *arXiv preprint arXiv:1401.4082*.

Roweis, S., & Ghahramani, Z. (1999). A Unifying Review of Linear Gaussian Models. *Neural Computation*.

Salakhutdinov, R. (2015). Learning deep generative models. *Annual Review of Statistics and Its Application*, *2*, 361–385.

Sanjana, N. E., Wright, J., Zheng, K., Shalem, O., Fontanillas, P., Joung, J., ... Zhang, F. (2016). High-resolution interrogation of functional elements in the noncoding genome. *Science*, *353*(6307), 1545–1549.

Searle, S. R., Casella, G., & McCulloch, C. E. (2009). *Variance components* (Vol. 391). John Wiley & Sons.

Shlyueva, D., Stampfel, G., & Stark, A. (2014). Transcriptional enhancers: from properties to genome-wide predictions. *Nature Reviews Genetics*, *15*(4), 272–286.

Skrondal, A., & Rabe-Hesketh, S. (2004). *Generalized Latent Variable Modeling* (Vol. 20041561). Chapman and Hall/CRC. doi: 10.1201/9780203489437

Sobel, M. E. (1994). Causal inference in latent variable models.

Song, J., & Chen, K. C. (2015). Spectacle: fast chromatin state annotation using spectral learning. *Genome biology*, *16*(1), 33.

Spearman, C. (1904, apr). "General Intelligence," Objectively Determined and Measured. *The American Journal of Psychology*, *15*(2), 201. doi: 10.2307/1412107

Sutskever, I., Martens, J., Dahl, G. E., & Hinton, G. E. (2013). On the importance of initialization and momentum in deep learning. *ICML (3)*, *28*, 1139–1147.

Tavaré, S., Balding, D. J., Griffiths, R. C., & Donnelly, P. (1997). Inferring coalescence times from dna sequence data. *Genetics*, *145*(2), 505–518.

Thomson, G. H. (1916, sep). A HIERARCHY WITHOUT A GENERAL FACTOR. *British Journal of Psychology, 1904-1920*, *8*(3), 271–281.

Thurstone, & L.L. (1947). *Multiple Factor Analysis.* Chicago.

Tipping, M. E., & Bishop, C. M. (1999, aug). Probabilistic Principal Component Analysis. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, *61*(3), 611–622.

Tobin, J. (1958). Estimation of relationships for limited dependent variables. *Econometrica: journal of the Econometric Society*, 24–36.

Tornio, M., Honkela, A., & Karhunen, J. (n.d.). Time series prediction with variational bayesian nonlinear state-space models..

Tran, D., Ranganath, R., & Blei, D. M. (2017). Deep and hierarchical implicit models. *arXiv preprint arXiv:1702.08896*.

Tucker, L. R. (1958). An inter-battery method of factor analysis. *Psychometrika*, *23*(2), 111–136.

Turner, B. M. (2007). Defining an epigenetic code. *Nature cell biology*, *9*(1), 2–6.

Vanhille, L., Griffon, A., Maqbool, M. A., Zacarias-Cabeza, J., Dao, L. T., Fernandez, N., ... Spicuglia, S. (2015). High-throughput and quantitative assessment of enhancer

activity in mammals by capstarr-seq. *Nature communications*, *6*.
Yip, K. Y., Cheng, C., & Gerstein, M. (2013). Machine learning and genome annotation: a match meant to be? *Genome biology*, *14*(5), 205.