Partition-based Model Representation Learning

Yayun Hsu

Submitted in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy
under the Executive Committee
of the Graduate School of Arts and Sciences

COLUMBIA UNIVERSITY

2020

Abstract

Representation Learning – A Partition-based Model Approach

Yayun Hsu

Modern machine learning consists of both task forces from classical statistics and modern computation. On the one hand, this field becomes rich and quick-growing; on the other hand, different convention from different schools becomes harder and harder to communicate over time. A lot of the times, the problem is not about who is absolutely right or wrong, but about from which angle that one should approach the problem. This is the moment when we feel there should be a unifying machine learning framework that can withhold different schools under the same umbrella. So we propose one of such a framework and call it "representation learning". Representations are for the data, which is almost identical to a statistical model. However, philosophically, we would like to distinguish from classical statistical modeling such that (1) representations are interpretable to the scientist, (2) representations convey the pre-existing subject view that the scientist has towards his/her data before seeing it (in other words, representations may not align with the true data generating process), and (3) representations are task-oriented.

To build such a representation, we propose to use partition-based models. Partition-based models are easy to interpret and useful for figuring out the interactions between variables. However, the major challenge lies in the computation, since the partition numbers can grow exponentially with respect to the number of variables. To solve the problem, we need a model/representation

selection method over different partition models. We proposed to use I-Score with backward dropping algorithm to achieve the goal.

In this work, we explore the connection between the I-Score variable selection methodology to other existing methods and extend the idea into developing other objective functions that can be used in other applications. We apply our ideas to analyze three datasets, one is the genome-wide association study (GWAS) [1], one is the New York City Vision Zero [2], and, lastly, the MNIST handwritten digit database [3].

On these applications, we showed the potential of the interpretability of the representations can be useful in practice and provide practitioners with much more intuitions in explaining their results. Also, we showed a novel way to look at causal inference problems from the view of partition-based models.

We hope this work serve as an initiative for people to start thinking about approaching problems from a different angle and to involve interpretability into the consideration when building a model so that it can be easier to be used to communicate with people from other fields.

Table of Contents

List of Figures

Chapter 1: Introduction and Background

1.1 Motivation

Machine learning is the intersection where many disciplines meet, especially Statistics and Computer Science. Modern machine learning suffers from the trade-off between inference and prediction [4]. [1] For example, in traditional statistical linear regression, under the normality assumption, the model learned from maximum likelihood gives us direct inference results on the coefficients. If one attempts to use the resulting model for predicting out-of-sample future observations, it is usually not the best, compared with other methods such as nonparametric regression, kernel methods, or even neural networks. However, the current solution to this problem is a matter of choice – practitioners are forced to choose sides between these two cultures, without other middle-ground option(s). In this work, we would like to propose a new learning framework, which we will call it the "representation learning", that acts as another option other than traditional statistical modeling and black-box modeling.

From a statistical viewpoint, many modern machine learning methods are the retell of stories from earlier statistical toolbox with the aids of modern computation. As a statistician, I am glad to see how modern computation elevates statistics to a new scale that statisticians would never had imagined a few decades ago. From Computer Science (and even Artificial Intelligence) standpoint, researchers are thrilled to see the idea of learning a prediction model from past data, instead of human hard-coded rules, has opened a whole new world of problem-solving regimes. For example, AlphaGo, the first computer Go program to beat a human professional Go player, is based on modern deep-learning systems, instead of hard-coded systems such as Deep Blue [2] that appeared

[1] Throughout this work, we will refer to the former as the traditional statistical model, and the latter as the black-box model, following the same naming of Breiman.

[2] the first computer chess-playing system to win both a chess game and a chess match against a human reigning world champion

nearly 30 years ago.

As a result, we have seen almost exponential growth in machine learning because of the joint contribution of great minds from various disciplines. However, this rapid development is not without a price – if we look at modern, computational-approached methods through a statistical lens, many of them are not explainable, and even violate some statistical principles; however, they perform well in practice. For example, a regression coefficient is not a causal estimate unless a large set of assumptions are met. But in practice, many assumptions are actually uncheckable. In those cases, do we give up the methods entirely? Do we still apply the methods blindly? Or are there any other choices?

Given the rich developments and successes on both sides, instead of trying to deny their different mindsets entirely, we feel the need to construct a new framework, where people from the two sides of the spectrum can meet and share a common language. Hopefully, by constructing this bridge, people will be able to understand and appreciate what each other has been doing, thus helping the further development of each.

1.2 Representation Learning

The solution that we propose here, the new learning framework that we will construct, will be called representation learning. Indeed, the term "representation learning" has been around for quite a while, but there is no unifying definition for it. From Computer Science community, representation learning has become a field in itself, with regular workshops at the leading conferences such as *NIPS* and *ICML*, and a new conference dedicated to it, the *ICLR* (International Conference on Learning Representations), sometimes under the header of Deep Learning or Feature Learning. The rapid increase in scientific activity on representation learning has been accompanied and nourished by many empirical successes both in academia and in industry [5]. Fields such as signal processing, natural language processing, object recognition, and transfer learning, have all seen benefits from representation learning.

In this work, my goal is not to provide a universal definition for representation learning; rather, my goal is to borrow the name and to set up a new learning framework that was described in the previous section. Throughout the whole thesis, I will focus on (1) setting up the framework, (2) providing a proper loss function/ objective function, and (3) interpreting the results and implications.

First of all, we define representations to be hierarchies of decompositions or transformations of the raw data such that some of them are useful for some tasks. There are generous-purposed representations and specific-tasked representations. Let me explain these qualitative properties of representations using an intuitive example. Figure 1.1 illustrates the idea – we are given a bunch of pictures of cats, which is the observable data. In representation learning, we abstract the pictures of cats into hierarchies of representations. The two ends of the hierarchical representations are the data (in this example, the picture of the cats) and the task (in this example, predicting the label "cat"). The data is observable; the label may be observable or not. Furthermore, the label can change domains – the same picture of a cat can be labeled as "cat", "animal", "happy", "summer" or "black-and-white", etc. Between the data and the label, several hierarchies of representations can be constructed. The first layer above the raw data is for more generous purpose; as the layers move towards the label, those layers become more and more specific for the "task" – predicting the label. Here all the inner layers are legitimate representations, but not all of them are useful for all tasks. For example, if the task is to distinguish a cat from a dog, then probably the color of the picture doesn't help much; while if the task is to distinguish in which season the picture was taken, then color becomes an important representation.

I will use the term "task" to include most of the problems considered in machine learning, involving prediction, estimation, variable selection, etc.

We will see later that, using this proposed learning framework, we will be able to achieve (1) good interpretation (2) prevention of overfitting, and (3) variable selection all at once.

As discussed previously, the purpose of constructing such a framework is to give a common language to both sides of the culture. So in the following sections, I will explain how to use this newly defined "representations" to look at some traditional statistical models.

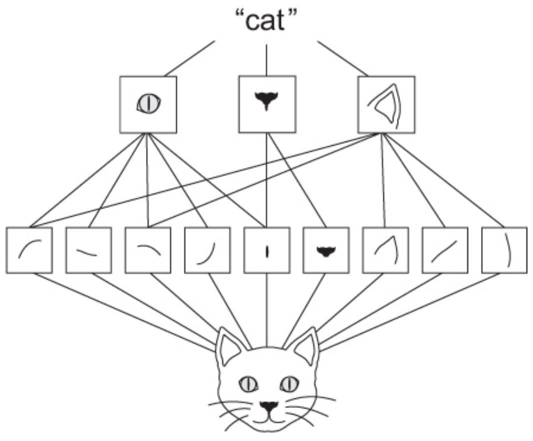


Figure 1.1: Example of a hierarchical representation of a cat.

1.2.1 Linear Regression as Representation

Suppose we have data $(Y_i, X_i^1, X_i^2, ...X_i^p)$, where i denotes the index for observations and p denotes the number of independent variables/ observed features. In the linear regression setting, we have model

$$\boldsymbol{Y} = \boldsymbol{X^T}\boldsymbol{\beta} + \boldsymbol{\epsilon}, \tag{1.1}$$

where $\boldsymbol{Y}$ is a known vector of observations with mean $\boldsymbol{X\beta}$, $\boldsymbol{\beta}$ is an unknown vector of constants, $\boldsymbol{\epsilon}$ is an unknown vector of random errors with mean 0 and variance $\boldsymbol{R}$, and $\boldsymbol{X}$ is a known design matrix. We can view Equation 1.1 as a decomposition of $\boldsymbol{Y}$ into the signal $\boldsymbol{X^T\beta}$ and the noise $\boldsymbol{\epsilon}$ parts. We call $\boldsymbol{X^T\beta}$ as the representation of $\boldsymbol{Y}$ under this model specification.

Traditionally in the modeling stage, the problem is that we don't know (1) how many independent variables we should use and (2) even if we know the answer to (1), we still don't know which variables compose the best subset. So many selection criteria have been proposed, such as AIC, BIC, and LASSO. On the other hand, from a black-boxed model point of view, if a better prediction accuracy can be achieved by adding more independent variables, there is no fault of doing so. It is a matter of the focus. In our proposed representation learning, we introduce a new focus – the interpretability of the representation. By interpretability, we mean the intuitive interpretation that we human can make sense about. In some cases, the interpretability may be obvious and even checkable in its own field knowledge. Such cases are like using random forests to build a genome-wide prediction for a certain disease. In other cases, the interpretability may not be obvious – in these cases, one may still want to stick with traditional criteria.

Now let's go back to the linear regression example, and elaborate more on the representation learning ideas. We know that, under the usual linear regression assumptions, the solution from machine learning theory says that the predicted Y is given by $\hat{Y} = (X(X^T X)^{-1} X^T)Y = HY$, where H is often called the hat matrix, and it stands for a linear transformation on Y [6] [7]. In our proposed representation learning framework, we say this is the empirical representation of Y. This representation is obtained through maximum likelihood estimation, so, by construction, it is interpretable for inference but may not for prediction. The interpretation is – $\hat{Y}$ is a projection of the vector $\vec{Y}$ onto the space spanned by $\boldsymbol{X}$, and it is perpendicular to the residual vector, $\vec{\epsilon}$; therefore, we have a orthogonal decomposition of Y into signal and noise.

However, in terms of prediction, as we know from machine learning theory that, for prediction purpose, it is often better to use slightly biased estimators obtained through penalization – this then becomes a black-boxed model. Is there another solution to the regression problem that can have both of the interpretability and the prediction accuracy?

Our proposed solution is that, linear regression can be done first by representation learning – this is the inference step that gives us interpretability – and then use the learned representations to build the prediction model. By doing so, we strike a balance between inference and prediction. Representation learning can use, for example, I-Score, principle component analysis (PCA), variational autoencoder (VAE), etc. The I-Score variable selection procedure is the main route we will pursue in this thesis, and hence more detailed introduction will be present later. But the basic idea is that we first use I-Score to perform variable selection, and then use the selected variables to build regression models or neural networks.

1.2.2 Mixed (Hierarchical) Models as Latent Representation

An additive mixed model is a statistical model containing both fixed effects and random effects. In matrix notation, a mixed model can be represented as

$$\boldsymbol{Y} = \boldsymbol{X}^T\boldsymbol{\beta} + \boldsymbol{Z}^T\boldsymbol{\mu} + \boldsymbol{\epsilon}, \tag{1.2}$$

where where $\boldsymbol{Y}$ is a known vector of observations with mean $\boldsymbol{X}^T\boldsymbol{\beta}$, $\boldsymbol{\beta}$ is an unknown vector of fixed effects, $\boldsymbol{\mu}$ is an unknown vector of random effects with mean 0 and variance–covariance matrix $\boldsymbol{G}$, $\boldsymbol{\epsilon}$ is an unknown vector of random errors with mean 0 and variance $\boldsymbol{R}$, and $\boldsymbol{X}$ and $\boldsymbol{Z}$ are known design matrices relating the observation $\boldsymbol{Y}$ to $\boldsymbol{\beta}$ and $\boldsymbol{\mu}$, respectively [8]. We will call $\boldsymbol{X}$ and $\boldsymbol{Z}$ as the "features" – $\boldsymbol{X}$ is the observable, raw feature, and $\boldsymbol{Z}$ is the latent feature (may or may not be observable).

In our representation learning language, we can view the above modeling as an additive, hierarchical decomposition of the observation $\boldsymbol{Y}$.

$$\boldsymbol{Y} = \underbrace{\underbrace{\boldsymbol{\mu_0} + \boldsymbol{\sigma_0\epsilon_0}}_{\mu_1} + \boldsymbol{\sigma_1\epsilon_1}}_{\mu_2} + \boldsymbol{\sigma_2\epsilon_2} + \ldots$$

assuming that each ϵ_i is independent with μ_0.

The modeling of a hierarchical model suffers from similar issues as linear regression – since we don't know how many hierarchies is "correct" (there may even not be a correct answer), we'd better first build representations in the inference step and then proceed with the model-building. The difference is that, since Z may be unobservable, it may require us to do some clustering on $\boldsymbol{Y}$ in building the representations.

For demonstration purpose (and without loss of generality), let us study a two-level model, albeit the concept can be generalized to even higher levels of hierarchies.

Several results from this model construction:

$$\mathbb{E}(Y) = \mu_0 \tag{1.4}$$

$$\mathbf{Var}(Y) = \sigma_0^2 + \sigma_1^2 + \sigma_2^2 \tag{1.5}$$

$$\mathbf{Cov}(Y, \mu_1) = \sigma_0^2 \tag{1.6}$$

$$\mathbf{Cov}(Y, \mu_2) = \sigma_0^2 + \sigma_1^2 \tag{1.7}$$

That is, even before we fit the model, we know the form of the variance-covariance structure. In some applications, such as kinship modeling, we can pre-assign the variance-covariance structure before fitting the model. This once again emphasizes the idea that an interpretable representation can go before model fitting, to ensure the learning results make intuitive sense in real applications.

1.2.3 Mixture Models as Representation

In mixture models, the whole population can be treated as a mixture of several mixture components. Since the mixture components are usually unseen, we refer to them as latent variables. The mixture components can be either discrete or continuous, but often for modeling convenience, they are assumed to be discrete. In general, a mixture model assumes the data are generated by the following process: first we sample from the latent variable Z, and then we sample the observables X from a distribution which depends on Z; that is,

$$p(x, z) = p(x|z)p(z), \tag{1.8}$$

where $p(x|z)$ is the representation based on latent components. Note that the above factorization doesn't suggest any ordering. So it is important to determine the ordering in the representation building stage.

Let us first review how traditionally people fit these kinds of model. Assuming that the true model is a Gaussian mixture model on the marginal data X with mixture components Z. X is observable while Z is not. We assume Z is a discrete random variable, and there are probabilities defined on $p(X|Z = k)$, for $k \in \{1, 2, ..., K\}$, where K is the total number of mixture components. Each $p(X|Z = k)$ is Gaussian with mean μ_k and variance σ_k^2. The parameters of interest are all the μ_k, σ_k^2 and $p(Z = k)$. We call each $p(X|Z = k)$ the probabilistic representation of X based on $Z = k$. If we follow maximum likelihood estimation for the parameters, there are two major issues: (1) different K may result in nonidentifiability, and (2) even if K is fixed, the MLE objective has no analytical solution.

In practice, to resolve issue (1), people pre-select a fixed K. To resolve (2), a more traditional statistical method is the expectation-maximization algorithm (EM), while in recent years, Bayesian methods such as topic modeling and variational inference have become more and more popular [9].

However, it is often hard to determine how many mixture components there should be. Later we will see that the method of I-Score representation learning can actually serve as a method for determining the number of mixtures by specifying some sort of "prior belief" [3].

On caveat is that, for a fixed dataset, there is no unique representation. It is simply because, representations are the subjective beliefs of the interpreter and they need not imitate the true data generating process, that we allow different practitioners to have their own representations, as long as they achieve the task well.

1.3 Partition-based Models

Here we define a partition over the feature space, where all features are discrete, is a collection of sets that (1) are mutually disjoint and (2) have a union as the entire composited feature space.

Suppose we have data $(Y_i, X_i^{(1)}, X_i^{(2)}, ...X_i^{(p)})$, where i denotes the index for observations, p denotes the number of independent variables/ observed features, all the Y_i's are continuous and all the $X_i^{(p)}$'s are discrete. The partition of the feature space is defined on the composition of $(X_i^{(1)}, X_i^{(2)}, ...X_i^{(p)})$. Assume that the partitions are $\{\Pi_1, \Pi_2, ...\Pi_K\}$, where K stands for the total number of partitions constructed.

[3]not the same kind of prior belief that people usually talk about in Bayesian inference

The model we consider is

$$Y_i = \sum_k \mu_k \cdot \mathbb{I}(Y_i \in \Pi_k) + \epsilon_i, \tag{1.9}$$

where μ_k is the "local mean" of Y in partition Π_k, and ϵ_i is the individual level error with mean 0 and independent with all other observations within the same partition, but may dependent with other observations across different partitions. The learning interests are all the local means μ_k and the clustering procedure $\mathbb{I}(Y_i \in \Pi_k)$. We interpret all the local means μ_k as the representation of Y under this partition-based model.

A partition-based model essentially clusters the observed Y's into these feature partitions. The desirable clustering should make observations inside each partition as homogeneous as possible. This has two consequences: First, we can then build local models within each partition, which yields a very expressive model. Second, we may find that some of the partitions contain no observations inside, which further implies the irrelevancy of the variables used to construct those partitions. Hence, variable selection is also achieved in the process.

The hardest part of building such a model is the choice of objective function and the corresponding algorithm to automate the clustering process. These two will be the focus of this thesis – the I-Score objective function and the corresponding backward dropping algorithm [10] [11].

Chapter 2: Technical Approach

The procedures of our representation learning involve the following: (1) latent structure learning of the predictors, (2) predictional modeling learning, (3) model checking, and (4) result interpretation. For (1), we use the I-Score, proposed by Chernoff, Lo and Zheng [12], to serve as a variable selection tool. For (2), we use the generalized additive model (GAM) and neural networks as the basis. We will see that the model learned in this way has a natural interpretation. Some applications on the full implementation will be discussed in Chapter 5-7.

2.1 I-score

The setting of the problem is that we are given a dataset $\{(Y_i, \boldsymbol{X_i})\}_{i=1}^N$ where N is the sample size. Although many variations have been proposed, here we stick to the setting that Y is a continuous random variable and all $\boldsymbol{X}$'s are discrete. There may be a few features in $\boldsymbol{X}$ that is not useful for predicting Y. We would like to sift them out, while not knowing what the true model is. The approach is to partition the N observations into several partitions, according to the composition of subsets of $\boldsymbol{X}$ with n_i observations in the i^{th} partition.

The composition of $\boldsymbol{X}$ is to redefine the high dimensional marginal X's into a one dimensional variable that takes the joint value. For example, if we originally have two-dimensional (X_1, X_2) in the data, and each of the random variables can only take values either 0 or 1. We can composite (X_1, X_2) into another random variable, say, X_3, such that X_3 can take four possible values that correspond to the situation when $(X_1, X_2) = \{(0,0),(0,1),(1,0),(1,1)\}$, respectively. We call each one in the composite set a partition, as it is a partition over the feature space.

A great advantage of this model is that we take into consideration the joint distribution of the original $\boldsymbol{X}$'s without making any implausible assumptions such as independence of the marginal

X's.

However, the price for being able to take care of the joint distribution in $\boldsymbol{X}$ is that this kind of composite random variables can easily become sparse in the sense that there are many possible partitions that don't have enough observations. So, instead of compositing all the raw features into one big new feature, we randomly sub-sample a few features and test their I-Score.

For a subset of $\boldsymbol{X}$ that composites i partitions, the I-Score is defined to be

$$I = \frac{1}{N}\sum_{i} n_i^2(\bar{Y}_i - \bar{Y})^2, \tag{2.1}$$

where n_i is the number of observations inside the i^{th} partition, $\sum_i n_i = N$, $\bar{Y}_i$ is the average of Y in the i^{th} partition (local mean), and $\bar{Y} = \sum_i \frac{n_i \bar{Y}_i}{N}$ is the overall average of Y (global mean). It can be a measure of influence based on how well the partition separates the subjects into relatively homogeneous subsets. It not only considers the deviation of local means to the global mean, but also take the number of observations inside each partition into account. The higher the I-Score, the more influential towards Y the set of features is.

One reason for explaining why partition models make such practical use is through local averaging. Many smoothers in nonparametric regression such as splines are known to be able to smooth the curve fitting by doing some kind of local averaging. However, the idea of locality becomes more and more vague as the dimension of features increases. This is the place where the partitions come in handy. In partitions, there is no ordering nor continuity. This methodology can then be related to tree-based methods, such as decision trees. However, the I-Score framework is different from trees in the sense that they take the interaction between features into account, by selecting a subset of features together at once; while in building decision trees, we only split the tree based on one marginal feature at a time.

Ideally, we would like to search over all possible subsets and find out the one with the highest I-Score. However, since the number of partitions grow exponentially with the number of features, it is impossible to do an exhaustive search in practice. Instead, they also proposed the backward dropping algorithm [10] [11]. Essentially it is a greedy search that every time we drop one variable that has the minimum contribution to the score until we reach the peak value.

2.1.1 Backward Dropping Algorithm

Suppose in our data, we have p features, $X_1, X_2, ...X_p$. The backward dropping algorithm is to randomly select a subset of features from the total p features, and to evaluate the I-Score of the current subset by gradually dropping one feature at a time, until we find the maximum I-Score generated by a certain subset of the current random subset – this is how we drop the redundant variables. In each dropping stage, we perform greedy search – to drop the one variable that gives the highest decrease of I-Score in each iteration. The procedure is practical in many real data experiments [10] [11] [13].

2.1.2 Comparison

In a partition model, where we have the local means $\bar{Y}_i$ and the global mean $\bar{Y} = \sum_i n_i \bar{Y}_i$, we have the following decomposition:

$$\sum_{i=1}^{n} \left(Y_i - \bar{Y}\right)^2 = \sum_{i=1}^{n} \left(Y_i - \bar{Y}_i\right)^2 + \sum_{i=1}^{n} \left(\bar{Y}_i - \bar{Y}\right)^2 . \tag{2.2}$$

In general, if we treat Equation 2.2 as a special case of the following equation, we can say that the local means in a partition-based model is an unbiased estimator for the response. So we have the following,

$$\sum_{i=1}^{n} \left(Y_i - \hat{Y}\right)^2 = \sum_{i=1}^{n} \left(Y_i - \hat{Y}_i\right)^2 + \sum_{i=1}^{n} \left(\hat{Y}_i - \bar{Y}\right)^2 . \tag{2.3}$$

When we try to minimize $\sum_{i=1}^{n} \left(Y_i - \bar{Y}_i\right)^2$, it is equivalent to maximize $\sum_{i=1}^{n} \left(\bar{Y}_i - \bar{Y}\right)^2$, since their sum, the sample variance, is fixed. Without any restriction on the number of partitions, the minimum value of $\sum_{i=1}^{n} \left(Y_i - \hat{Y}_i\right)^2$ is zero; that is, by giving each datapoint its own partition. This is the same phenomenon of overfitting in linear regression. In linear regression, we prevent overfitting by restricting the functional form of $\hat{Y}_i$ to be linear. On the contrary, in our proposed partition-based representation learning, we prevent overfitting by choosing partitions that prefer more than one observations to be inside each partition. This is guided by using the I-Score as the objective function, since the I-Score is essentially the weighted version of the second term on the right hand side of Equation 2.2. The rationale of doing this is to guide the learning process to learn a meaningful, interpretable representation, instead of blindly optimizing the objective function. The representation here is that, inside each partition, the data are homogeneously i.i.d. In other words, the proposed partition-based representation learns a smoother representation than the normal regression, without being completely smooth (the null model of regression). On the other hand, it improves the typical objective functions used in tree-based learning methods by taking the interactions of the features into account, since, instead of finding marginal signal each time, one whole partition that consists of many variables is considered every time the in I-Score method.

In sum, if we see there is a spectrum of methods, one side of which is linear regression and the other side being tree-based methods, the I-Score framework is in the middle. It has a natural interpretability just like trees, and meanwhile, it is computationally efficient as linear regression.

2.2 Generalized Additive Model (GAM)

Generalized additive models (GAM) are the generalization of general linear models [14] [15]. The idea is based upon the Kolmogorov-Arnold representation theorem that any multivariate function could be represented as sums and compositions of univariate functions. Mathematically, if we start from the basic linear regression, we can compare the assumptions of these similar models:

1. Standard Linear Regression: We assume the observable Y follows a Normal distribution, where the mean of the distribution is a linear combination of the observable features $X_1, X_2, \ldots$

$$Y \sim N(\mu, \sigma^2), \text{ where} \tag{2.4}$$

$$\mu = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots \tag{2.5}$$

2. Polynomial Regression: We assume the observable Y follows a Normal distribution, where

the mean of the distribution is a polynomial function of the observable features $X_1, X_2, \ldots$

$$Y \sim N(\mu, \sigma^2), \text{ where} \tag{2.6}$$

$$\mu = \beta_0 + \beta_1 X_1 + \beta_2 X_1^2 + \ldots \tag{2.7}$$

3. Generalized Linear Model: We assume the observable Y follows an exponential-family distribution, where the link function of the mean is a linear combination of the observable features $X_1, X_2, \ldots$

$$Y \sim \text{Exponential Family, where} \tag{2.8}$$

$$g(\mu) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots, \tag{2.9}$$

   where $g(\cdot)$ is the link function.

4. Generalized Additive Model: We assume the observable Y follows an exponential-family distribution, where the link function of the mean is a linear combination of functions in the observable features $X_1, X_2, \ldots$

$$Y \sim \text{Exponential Family, where} \tag{2.10}$$

$$g(\mu) = \beta_0 + f_1(X_1) + f_2(X_2) + \ldots, \tag{2.11}$$

   where $g(\cdot)$ is the link function, and $f_j(X_j)$ may be functions with a specified parametric form (for example, a polynomial) or may be specified non-parametrically.

The above comparison clearly shows how GAM generalizes based on GLM. What is more, since each feature is added to the model term by term, the model fitting is not only flexible, but also computationally efficient.

2.3 Neural Networks

In 1943, artificial neural networks (ANN) were introduced by Warren McCulloch and Walter Pitts [16] as a computational model [17]. Rosenblatt [18] created the perceptron, which later became the basis for building more complex networks [19]. The first functional networks with many layers were published by Ivakhnenko and Lapa in 1965, as the Group Method of Data Handling [20] [21] [22]. The basics of continuous backpropagation [20] [23] [24] [25] were derived in the context of control theory by Kelley [26] in 1960 and by Bryson in 1961 [27], using principles of dynamic programming.

Later in the seventies, the technique of automatic differentiation proposed by Seppo Linnainmaa was applied to discrete connected networks of nested differentiable functions [28] [29]. In 1973, Dreyfus used backward propagation to adapt parameters of controllers in proportion to error gradients [30]. Werbos's (1975) backpropagation algorithm enabled practical training of multilayer networks. In 1982, he applied Linnainmaa's automatic differentiation method to neural networks in the way that became widely used [23] [31].

In the eighties, computation power enjoyed a rapid growth because of very-large-scale integration (VLSI) – increasing transistor count in digital electronics provided more processing power – and hence many models that were considered computationally expensive became possible in practice. Geoffrey Hinton et al. (2006) proposed learning a high-level representation using successive layers of binary or real-valued latent variables with a restricted Boltzmann machine [32] to model each layer. In 2012, Ng and Dean created a network that learned to recognize higher-level concepts, such as cats, only from watching unlabeled images [33]. Unsupervised pre-training and increased computing power from GPUs and distributed computing allowed the use of larger networks, particularly in image and visual recognition problems, which became known as "deep learning" [34].

Once again, another technology advancement elevated the possibility to the next level – in 2010, Ciresan et al. showed that GPUs make backward propagation feasible for many-layered feedforward neural networks [35]. Between 2009 and 2012, ANNs began winning prizes and approaching human level performance on various tasks, initially in pattern recognition and machine learning [36] [37]. Nowadays ANNs have beaten many old models and even become the standard models in many fields, such as computer vision and natural language processing (NLP).

The structure of a neural network can be seen as a directed, weighted graph, as illustrated in Figure 2.1.

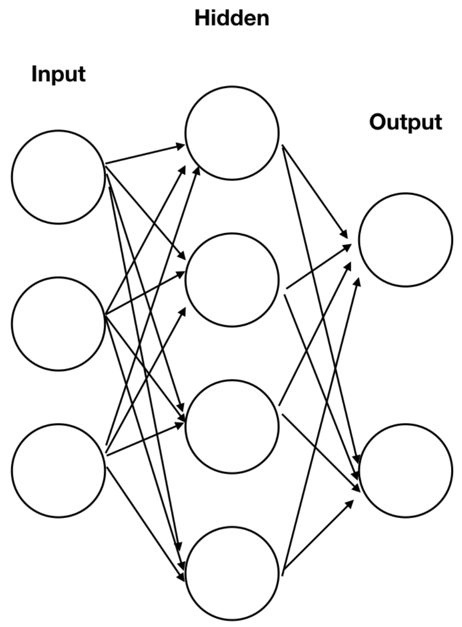


Figure 2.1: Illustration of the structure of an ANN.

An artificial neural network consists of a collection of simulated neurons. Each neuron is a node which is connected to other nodes via links. Each link has a weight, which determines the strength of one node's influence on another [38]. The neurons are typically organized into multiple layers, especially in deep learning. Neurons of one layer connect only to neurons of the immediately preceding and immediately following layers. The layer that receives external data is the input layer. The layer that produces the ultimate result is the output layer. In between them are zero or more hidden layers. Single layered networks are also used. Between two layers, multiple connection patterns are possible. They can be fully connected, with every neuron in one layer connecting to every neuron in the next layer. They can be pooling, where a group of neurons in one layer connect to a single neuron in the next layer, thereby reducing the number of neurons in that layer [39]. Neurons with only such connections form a directed acyclic graph and are known as feedforward networks. Alternatively, networks that allow connections between neurons in the same or previous layers are known as recurrent networks. Other than these, there have been many variations being proposed.

2.4 Generative Neural Networks

There is a surge of generative deep neural networks in the recent years. To train a generative model, we first collect a large amount of data in some domain (e.g., think millions of images, sentences, or sounds, etc.) and then train a neural network to generate data like it. The most popular structure nowadays are variational autoencoders (VAE) and generative adversarial networks (GAN). [1] [40] [41] [42]

The trick is that the neural networks we use as generative models have a number of parameters significantly smaller than the amount of data we train them on, so the models are forced to discover and efficiently internalize the essence of the data in order to generate it.

Generative models have many short-term applications. But in the long run, they hold the potential to automatically learn the natural features of a dataset, whether categories or dimensions or something else entirely. This will be an important step towards artificial intelligence.

Currently, all of the different approaches to generative neural networks have their pros and cons. For example, VAEs allow us to perform both learning and efficient Bayesian inference in sophisticated probabilistic graphical models with latent variables. However, their generated samples tend to be slightly blurry. GANs currently generate the sharpest images but they are more difficult to optimize due to unstable training dynamics. All of these problems do not have certain solutions and are very active areas of research now.

[1]The more detailed introduction will be present in Chapter 7.

2.5 Entropy and Mutual Information

The entropy $H(X)$ of a discrete random variable X is defined as

$$H(X) = -\sum_{x \in X} p(x) \log p(x). \tag{2.12}$$

It is a function which attempts to characterize the unpredictability of a random variable, both in terms of the number of possible outcomes and its frequency.

The concept of mutual information is intimately linked to that of entropy of a random variable, a fundamental notion in information theory that quantifies the expected amount of information held in a random variable.

Not limited to real-valued random variables and linear dependence like the correlation coefficient, mutual information is more general and determines how different the joint distribution of the pair (X, Y) is to the product of the marginal distributions of X and Y. Formally, the definition of mutual information is (again, we only consider both X and Y to be discrete)

$$MI(X;Y) = \sum_{x \in X} \sum_{y \in Y} p(x,y) \log \frac{p(x,y)}{p(x)p(y)}. \tag{2.13}$$

2.6 Definition of Terminologies

In this coming sections, we will define the terminologies that will be used throughout the whole work.

2.6.1 Task and Objective Function

Definition 2.6.1. A task for a machine learning procedure is the optimization of an objective function.

Commonly-seen tasks are prediction, classification, inference, and variable selection. When defining a task, a corresponding objective function that is based on empirical data should also be defined. For example, when one defines “prediction” as the task, there should be a corresponding objective function, such as the sum of squares of the difference between the observed value and the predicted value.

Definition 2.6.2. Objective functions are statistics that we can develop optimization algorithms for.

2.6.2 Feature v.s. Latent Factor/Variable

Features are observable independent variables. Latent factors/variables are unobservable independent variables that are not present in the dataset. We will refer to the features present in the dataset as the raw feature, and any other transformation of the raw features will be called transformed features. Since latent factors/variables are unobservable, they are assumed by the scientists and then learned from data.

Historically, we see there is the trend of learning from raw to transformed features. It starts from feature engineering, the most concrete case, when people hard-code their prior knowledge about the underlying mechanism into the model. For example, in a facial recognition system, we may hard-code features such as the eye color, the face shape and the length of hair into the system. However, in order to expand the scope and applicability of machine learning methods, we actually would like to make learning algorithms to be less dependent on feature engineering [5]. Another rationale for doing this is about data-drivenality – most of the time, we actually have limited knowledge about what good features are, so it would be better for the machine to figure it out by itself from data. More importantly, if we want to make progress towards artificial intelligence, which fundamentally means to understand the environment around, the artificial intelligence agent must have the ability to figure out important underlying explanatory factors hidden in the observed sensory data [5].

In terms of a data generating process, we define the flow of the data generating factors C_i, the observable raw features X, and the transformed features$f_i(X)$, where i denotes the indexing. The idea is illustrated in Figure 2.2. In this example, we have four data generating factors C_1 to C_4, the observable raw feature X, and the transformed features $f_1(X)$ and $f_2(X)$.

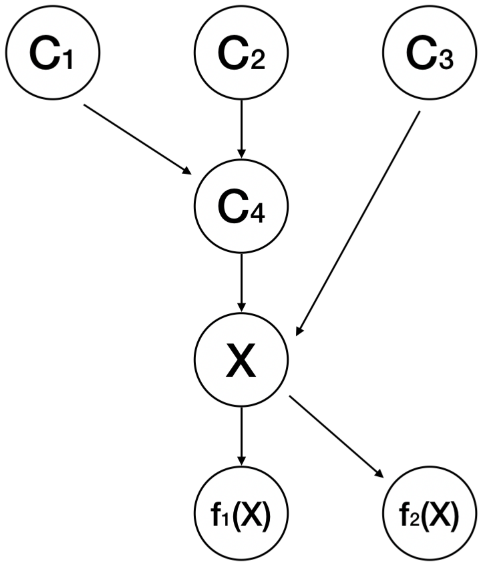


Figure 2.2: A data generating process example.

If we model the data generating process as a sequential procedure, we call everything happened before the raw features as the upstream transformations; similarly, we call everything happened after the raw features as the downstream transformations. So in the previous example, the raw feature is X, the upstream transformations include C_1 to C_4, and the downstream transformations include $f_1(X)$ and $f_2(X)$. The ordering may not impact much in the mathematical modeling stage, but they are essential in adding on interpretability of the model.

In sum, we make the following definitions:

Definition 2.6.3. Transformed features are the downstream transformations of the raw feature. Without seeing the data, neither features or transformed features can be formed.

Definition 2.6.4. Latent Factors/Variables are the upstreams transformations of the raw features. They can be formed without data.

2.6.3 Feature Selection v.s. Feature Embedding

Similarly, as the development of machine learning in many different applications, different names of similar procedures were generated. We will use the following unifying definition whenever we mention about these terminologies:

Definition 2.6.5. Feature selection is the process of choosing relevant and useful features for sufficing the task.

Definition 2.6.6. Feature embedding is the process of transforming features.

In fact, in this definition, feature selection can be treated as one special case of feature embedding, where each feature is transformed into the feature times the corresponding indicator function which indicates the selection decision.

2.6.4 Discriminative Model v.s. Generative Model

Definition 2.6.7. Discriminative models are models that could model the process and give results of either regression (continuous response) or classification (discrete response). They do not necessarily reproduce the true data generation process, but is sufficient for the discrimination task.

Definition 2.6.8. Generative models are models that try to reproduce the true data generation process. Why do we need generative models? First, they help us understand the data generation process, which may be potentially helpful if we would like to do causal inference or transfer learning. Second, they add on interpretability. And, thirdly, when we don't have enough data, they can generate more data in for semi-supervised learning purpose.

In some cases, discriminative models can be viewed as a hierarchy embedded inside a generative model. The decision of going for a discriminative or a generative model should be determined by the interpretability of the corresponding representation and the ultimate task.

Chapter 3: Sufficient Representation Learning Framework

3.1 Introduction

As introduced earlier, there is the dilemma between inference and prediction. As a statistician, I would like to begin with traditional statistical inference methods, and gradually move towards modern statistical methods that incorporate decision theory, graphical models and Bayesian methods. While each has its own origin and assumptions, practitioners nowadays seem to ignore these important distinctions, which may lead to misuse of the inferences. In this work, we would like to create a unifying framework that connects statistical modeling methods with computationally-inspired black-boxed models such as neural networks. We hope this will provide more insights and correct the practice of statistics into various fields of applications.

One of the major challenges of transporting traditional statistical methods into modern data is the scale of the dataset. At the time while Statistics was emerging, the applicable data wasn't really big or high-dimensional compared to nowadays data. So even though the model wasn't exactly "correct", it may not be too far away from the correct one. However, as more complicated and high-dimensional data emerge, such as genome data and image data, the correctness of model specification has become more important than ever before. So, in this work, we would like to discuss about the interaction between model, representation and optimization for a task – which could probably be omitted when the data was not so complicated, but no longer so once we enter the big data era.

So our first idea of the new learning framework is that, since "all models are wrong" [43], let's shift our focus on finding models that are interpretable. We call these interpretable models the representation of the data. As machine learning progresses, there are more and more demands into this criteria, especially when we enters the era of artificial intelligence. For example, instead of letting a black-boxed algorithm to tell us if the scan photo of a patient's liver has a tumor, we often feel more comfortable and reliable if the algorithm also tells us why. The consequence of this is that our desirable representations may go before data-fitting, which violates traditional statistical learning principles.

Hence, we would like to suggest that, instead of trying to find the true model (if it exists), we try to find a useful representation within a model that is interpretable and sufficient for a task. Traditionally in model fitting, we have the mindset that there is only one true model, so we have to perform model selection to choose the best one. However, in representation learning, there is no true or false. Representations simply reflect how the scientists interpret the data but they do not need to reflect the true status of the nature. In other words, there can be many different representations for the same data, as long as they are sufficient for their assigned tasks.

3.2 Sufficient Representation – Statistical Desirable Qualities

We are now in the process towards defining what a good representation is. Since the idea of being good may be too subjective, we, instead, identify some desirable properties of an appropriate representation. In Bengio 2014 [5], he discussed about many criteria for a good representation from a computational point of view. Essentially, in machine learning, the performance of the methods is highly dependent on the choice of good features [5]. However, in practice, people choose features based on the performance of the machine learning method. The rationale being that, if the performance of the machine learning method is good, then it must be that the underlying features are good. But is this the right way? Or how about doing it in the opposite way? As discussed previously, in representation learning, we are concerned about the interpretability – if we completely let the performance of a discriminative learning method guide the feature selection or embedding process, the resulting features may not be interpretable. One family of such examples is the L-x penalization methods in linear regressions. While computationally efficient, it can hardly explain the meaning of the results beyond statistical significance.

Even in the case if the goal of the learning is about the ultimate performance, where learning the optimum representations is not the most important consideration, we may still find representation learning as an important step for modern machine learning problems. For example, if we have a bunch of photos of cats and dogs, and the task is to do classification; while a logistic regression model may well suffice the needs, people still appreciate methods like neural networks that can provide some insights about the representations learned inside the network. Another example is that, in identifying possible cancer patients, decision trees are much preferred than regression methods because of its natural interpretability which provides much more insights for scientists and doctors to look further into the biological meanings of those variables. This is also mentioned in Bengio et al. [5] that the success of machine learning algorithms generally depends on data representation, because different representations can entangle and hide more or less the different explanatory factors of variation behind the data [5]. Although specific domain knowledge can be used to help design representations, learning with generic priors can also be used, and the quest for artificial intelligence is motivating the design of more powerful representation learning algorithms implementing such priors [5].

3.3 Sufficient Representation – Statistical Desirable Qualities

So far, we have discussed some qualitatively desired properties of representation learning. In this section, we would like to propose one statistical criteria, sufficiency, inspired from classical Statistics.

To apply the idea to our proposed partition-based model, the key is to find a partition-based representation such that, conditioned on the representation, the task and the raw features become independent. If that is the case, then we don't need to worry about the distributions of the raw features, which, in some sense, may help variable selection and dimensional reduction at the same time.

We need to assume that the raw data itself is always sufficient. This guarantees the existence of a sufficient representation for any specific dataset.

3.4 Definition of Sufficient Representation

Here we would like to define, quantitatively, what a sufficient representation is. The idea is the generalization of sufficient statistics. We have data including raw features $\boldsymbol{X} \in \mathbb{R}^p$, and there may or may not be corresponding label Y. Here we assume $Y \in \mathbb{R}$. Even if Y has more than one dimension, we can deal with one of the dimensions each time, and we assume all of them are independent. We also define a task θ. This θ can be, for example, estimating a parameter, estimating a function, choosing a model, ..., etc. Whether θ is stochastic or not depends on the design and application.

We would also like to define what we mean by "data" – a generalized idea. There are three kinds of data: the observed, the futured, and the unobserved. The observed (or the collected) data is the conventional data. They are the current data that we have already observed their values. We will use subscripts to denote them in order to tell them apart; for example, X_i means the i^{th} observed data X. The future data is the imaginary data that may exist in the future, but unknown to us at the current moment. The unobserved data can be either in the past, concurrent, or in the future. Since it is never observed, it does not matter when it would happen. A hidden feature also belongs to this category. We will see the importance of these distinctions later.

According to the task, a corresponding objective function $f_{obj}(\boldsymbol{X_i}, Y_i, \theta)$, or $f_{obj}(\boldsymbol{X_i}, \theta)$, depending on the availability of Y, will be defined. We define the achievement of the task to be obtaining the optimum solution, θ^*, to the objective function. This can be done analytically or computationally. In practice, the achievement of the exact solution may not be feasible, but, still, one can try to find a proxy or a subset to the optimum solution. The optimum solution can be written as

$$\theta^* = argmax_\theta f_{obj}(\boldsymbol{X_i}, Y_i, \theta), \text{ or} \tag{3.1}$$

$$\theta^* = argmax_\theta f_{obj}(\boldsymbol{X_i}, \theta). \tag{3.2}$$

Note that there may or may not be a "true" theoretical θ. Even if there is one, the optimum solution θ^* from the objective function may or may not equal to the true value θ. Furthermore, when θ^*

is obtained by an optimization algorithm, it becomes a function of the algorithm. This gives practitioners more flexibility in designing the objective function and the corresponding optimization algorithm(s).

We define a sufficient representation $\boldsymbol{Z}$ to be a function of the data $(\boldsymbol{X}, Y)$, such that the achievement of the task is conditionally independent with the observed data $(\boldsymbol{X_i}, Y_i)$, given the representation. That is,

$$\theta^* \perp\!\!\!\perp (\boldsymbol{X_i}, Y_i)_{i=1}^N | \boldsymbol{Z}, \text{ or} \tag{3.3}$$

$$\theta^* \perp\!\!\!\perp (\boldsymbol{X_i})_{i=1}^N | \boldsymbol{Z} \tag{3.4}$$

One quick example is to perform MLE on Normally-distributed data – the objective function 3.2, is the Normal likelihood function, and the task θ is to estimate the mean of the Normal distribution (assuming the variance is known). According to the theory of MLE, the sufficient statistic is the sample mean. We then interpret (in the representation learning setting) that, the sample mean is the representation for the data in achieving the task of estimating the population mean. In other words, since it is the representation, to estimate the population mean, all we need is the sample mean and we can throw away other information in the raw data. However, the missing piece in classical theory is that the task for MLE learning is for inference, so the results are for inference purpose. If we want more from the results of MLE, such as prediction, it is a wrong use of the model and may lose the interpretability as well.

Next, we would like to propose that, one dedicated objective function should be used for one task at a time. There may not be a universal mathematical benefit of doing this, but this will guarantee the interpretability of the representation. For example, if our goal is to perform dimension reduction and linear regression, we compare the following two methods: (1) a principle component analysis (PCA) is performed first, and then use the resulting principle components for a linear regression, and (2) a linear regression with L-1 penalty (i.e. LASSO). While there are pros and cons for these two methods, we can always say that (1) has a better interpretability than (2), because it has a dedicated step for dimension reduction and another one for regression. So we can interpret the results separately; while in (2), the dimension reduction is almost done inside a black-box – it is very hard to interpret the result.

Finally, there is a connection between the objective function and the algorithm that is used to optimize the objective. Even for the same objective function, different optimizing algorithms may yield different solutions. So it is important to specify the pair of "objective function + optimization algorithm" together.

More examples will be present later in this chapter. Before that, we would like to give a general guideline of the procedures that one should follow for performing our proposed representation learning.

3.5 Representation Learning Procedure

In order to implement our proposed sufficient representation learning, we propose the following procedures:

1. Define the task, the corresponding objective function, the optimization algorithm (if no analytical solution), and the desired form of the representation.
2. Define the joint distribution of every variable including the representation, and decompose the likelihood up to the desired level of hierarchy. Choose prior(s) if needed.
3. Find the optimum solution to the objective function and check the sufficiency of the representation.
4. Use data to perform the algorithm.
5. Interpret the results or assess the goodness of the learned representation.

3.6 Notation

In the following sections, more examples will demonstrate how this sufficiency can be achieved in different contexts. Here let us define the notations and terminologies that will be used later in this chapter.

If $\boldsymbol{X} \in \mathbb{R}^p$ is a vector that has p dimensions, we will denote each dimension component by $X^{(k)}$, where k means the i^{th} dimension component. In other words, $\boldsymbol{X} = (X^{(1)}, X^{(2)}, ..., X^{(p)})$. If $\boldsymbol{X}$ represents a dataset, then we call $X^{(1)}$ the first feature, $X^{(2)}$ the second feature, ..., etc. We will use subscriptions to denote the sample index; for example, $X_i^{(1)}$ means the value of the first feature $X^{(1)}$ for the i^{th} sample. Without special explanations, $\boldsymbol{X}$ is for the independent variables, or features; and $\boldsymbol{Y}$ is for the response variables, or labels.

For collected data, we will denote it by $(Y_i, X_i^{(1)}, X_i^{(2)}, ..., X_i^{(p)})_{i=1}^N$ or $(X_i^{(1)}, X_i^{(2)}, ..., X_i^{(p)})_{i=1}^N$, depending on the availability of Y. i is for the index of the (unordered) data values.

3.7 Toy Example

Suppose the true model is $Y = X^{(1)} + \log X^{(1)} + \epsilon$, where Y is the response variable, $X^{(1)}$ is the only true feature, ϵ is the error that follows a standard Normal distribution and is independent with $X^{(1)}$. In reality, we collect data $(Y_i, X_i^{(1)}, X_i^{(2)})_{i=1}^N$, where $X^{(1)}$ is the first feature and $X^{(2)}$ is the second feature in the dataset. In other words, there are redundant features in the dataset. However, in practice, we don't know which features are redundant, nor the true model. How should we model the data?

The common practice that we see is to jump right into some type of regression, hoping that one magical model will do it all.

However, we propose that let us pause for a second and define the task first, since, modeling the data is not done blindly; instead, it should be task-oriented.

Task 1: In our first example, let's say *Task 1* is to predict future Y; that is, to obtain the optimum estimator for future Y, denoted by $\hat{Y}$. The corresponding objective function is chosen to be minimizing the expected square loss $\mathbb{E}(\frac{1}{N}\sum_i (Y_i - \hat{Y})^2)$, where the predicted future value, $\hat{Y}$, is constructed from the current dataset, and Y_i's are the actual future data. We assume that the future $\boldsymbol{X}$ are known, so there is no need to be worried about the uncertainty for $\boldsymbol{X}$. From machine learning theory, we know the bias-variance decomposition:

$$\text{Assume } y = f(x) + \epsilon \text{ is the true model and } \hat{f} \text{ is the estimator for } f\text{, then we have} \tag{3.5}$$

$$\mathbb{E}(\frac{1}{N}\sum_i (Y_i - \hat{Y})^2) = \mathbb{E}(Y - \hat{Y})^2 \tag{3.6}$$

$$= \mathbb{E}(f(x) + \epsilon - \hat{f(x)})^2 \tag{3.7}$$

$$= \mathbb{E}(\epsilon)^2 + \mathbb{E}(f(x) - \mathbb{E}(\hat{f(x)}))^2 + \mathbb{E}(\hat{f(x)} - \mathbb{E}(\hat{f(x)}))^2 \tag{3.8}$$

$$= \mathbb{E}(\epsilon)^2 + (f(x) - \mathbb{E}(\hat{f(x)}))^2 + \mathbb{E}(\hat{f(x)} - \mathbb{E}(\hat{f(x)}))^2 \tag{3.9}$$

The minimum of Equation 3.9 is obtained at $\mathbb{E}(\epsilon)^2$, which is the variance of Y. This is often called the irreducible error. One obvious solution is when $\hat{Y} = f(x)$.

$$f(x) = \text{argmin}_{\hat{Y}}\, \mathbb{E}(\frac{1}{N}\sum_i (Y_i - \hat{Y})^2), \tag{3.10}$$

and it obviously satisfies the sufficiency condition Equation 3.3:

$$f(X) \perp\!\!\!\perp (Y_i, X_i^{(1)}, X_i^{(2)})_{i=1}^N | \hat{Y} = f(X), \tag{3.11}$$

where X without subscriptions is for future values, and X_i and Y_i with subscriptions are for values in the collected data.

However, to obtain the exact true function almost never happens in reality.

Since the task of prediction is actually much less than obtaining the whole true function, the solution becomes much more obtainable and practical, if we are willing to say that, we are not trying to find the true function; instead, we are trying to find a function that can perform the task of prediction as well as the true function. This new function that we construct has our own desired properties and is called a representation.

For example, if we use nonparametric regression methods such as a spline and find the estimated function $\hat{f}_{spline}(x)$. Let's assume that, for all x within our interest of prediction, $\hat{f}_{spline}(x) = f(x)$. Then we also have the sufficiency condition just like Equation 3.11:

$$f(X) \perp\!\!\!\perp (Y_i, X_i^{(1)}, X_i^{(2)})_{i=1}^N | \hat{f}_{spline}(X), \tag{3.12}$$

while the functional form of $\hat{f}_{spline}(x)$ can be totally different from the true $f(x)$. We call $\hat{f}_{spline}(x)$ a representation, because it contains our belief of looking at the data through the lens of the basis functions of the spline, and the results are interpretable using lingos in spline models.

In sum, in this example, we used useful representations of the raw features $\boldsymbol{X}$ for the task of predicting future Y. A representation is sufficient if we can fulfill the task using the representation only, instead of using the raw features.

Task 2: In our second example, let's say *Task 2* is to find influential variables that helps to predict Y, then identifying $X^{(1)}$ alone is sufficient for this task, regardless of the functional form. How can we correctly identify $X^{(1)}$? We propose using our partition-based model and use I-Score as the selection criteria with backward dropping algorithm. In this framework, the partitions in $X^{(1)}$ is sufficient for fulfilling this task, and the corresponding representation is each local mean of Y inside the partitions. More detailed discussion will be present in Chapter 4.

Identifying influential variables is not a typical task in statistical machine learning. However, it is useful for many modern applications. Take, for example, the transfer learning in the context of modern artificial neural networks. The idea is that, the labels Y of a picture may be "attached" a posteriori, and we would like to be able to switch between different labeling domains. For example, if we have a handful of pictures of cats and dogs, we can label them according to the animals inside the picture, or we can also label them according to colorful or black-and-white picture. In this kind of applications, we need to identify some general-purposed influential variables regardless of the functional form. Since the exact functional form may change, if we switch from one domain to another, the general-purposed influential variables are less likely to vary a lot from domain to domain.

3.8 Representation Learning v.s. Generative Model Learning

Now we have defined our proposed representation learning framework, and we would like to compare it with a generative modeling learning framework using the new viewpoints. The purpose of this is to provide one more example for distinguishing our method with other existing methods, as well as to pave the background for Chapter 7, when we will discuss the application in a generative model.

Representation learning and generative model learning are two different problems by nature. As discussed previously, representations are subjective in the sense that they reflect the way the scientists interpret the data; while (generative) models are (mimicking) the true, objective data generating process.

Let's illustrate the idea using Figure 3.1, which shows the general idea of representation learning. Here we follow the same notations from Figure 2.2, except for adding Y, the response variable, Z, the representation, and the task. It is obvious that, in representation learning, we want to learn Z, and all the required information can be taken from X and Y, which are observable. However, if we want to learn the generative model, we need to start with all the latent generating factors from C_1 to C_4, which may not be viable in practice.

However, as we will develop later, representation learning can actually be beneficial to generative model learning. With the invention of variational autoencoder (VAE), researchers have successfully combined these two kinds of learning together into one architecture. Kingma et al. [40] proposed the idea that learning a generative model can feedback for representation learning. Essentially, we can imagine the task in Figure 3.1 is connected back to the generating factors C_1 to C_4. On the flip side, to assess the goodness of the learned representation, one can assess the goodness of samples generated based on the learned representation.

Now we have laid out the framework of representation learning, and proposed one desirable

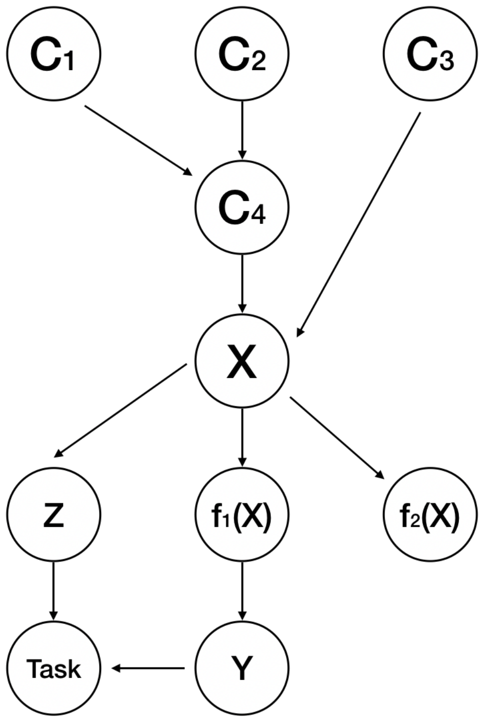


Figure 3.1: An illustration of representation learning.

property of the representation. Next, we will shift our focus to how to design an objective function for a specific set of representations, so that the learned representation is as sufficient as possible [1]. Let's first do this by reviewing the objectives in the framework of latent variable modeling.

3.8.1 Searching For the Objectives For A Generative Model

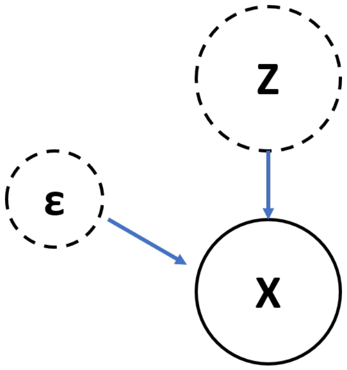


Figure 3.2: Latent variable model.

There are many ways of modeling a generative model, one of which is through latent variable

[1]In some cases, the assessment of sufficiency may not be quantifiable.

modeling. In the existing literature, the formulation of latent variable model can be defined through the graphical model as shown in Figure 3.2, where we have data $\{\boldsymbol{X_i}\}_{i=1}^N$ and latent variable $\boldsymbol{Z}$, where $\boldsymbol{X_i} \in \mathbb{R}^p$ and is observable, while $\boldsymbol{Z}$ is not observable and the dimension depends on the modeling.

The graphical model suggests the joint distribution of the observable $\boldsymbol{X}$ and the latent $\boldsymbol{Z}$ can be factorized into

$$p(\boldsymbol{X}, \boldsymbol{Z}) = p(\boldsymbol{X}|\boldsymbol{Z})p(\boldsymbol{Z}) \tag{3.13}$$

There are many different objectives for learning these types of model. Particularly, since $\boldsymbol{Z}$ is unobservable, if one marginalizes the latent variable, the typical maximum likelihood method can be applied on the observable $\boldsymbol{X}$.

$$p(\boldsymbol{X}) = \int p(\boldsymbol{X}|\boldsymbol{Z})p(\boldsymbol{Z})d\mathbf{Z} \tag{3.14}$$

Note that since only the data $\boldsymbol{X}$ is observable, if we start from Equation 3.14, there are many different ways of specifying latent variable $\boldsymbol{Z}$'s. In other words, there is no one unique solution to the latent variable $\boldsymbol{Z}$'s, even if there truly exists one.

One of the oldest method for learning this kind of model is the expectation-maximization (EM) algorithm. In that setting, we have a pre-defined parameterized distribution of $\boldsymbol{Z}$, denoted by $p(\boldsymbol{Z})$. The inference goal is to learn the parameters in the joint distribution of $\boldsymbol{X}$ and $\boldsymbol{Z}$, denoted by $p_\theta(\boldsymbol{X}, \boldsymbol{Z})$, where θ is the parameter of interest. The distribution is often assumed to be from an exponential family. Note that the key assumption for this method to work is that the joint distribution $p_\theta(\boldsymbol{X}, \boldsymbol{Z})$ has to be fixed. In other words, if we change the distribution on the "prior" $p(\boldsymbol{Z})$, it cannot change the joint distribution $p_\theta(\boldsymbol{X}, \boldsymbol{Z})$. However, how to specify $p(\boldsymbol{Z})$ remains difficult in practice.

Later, in the variational inference (VI) framework, people claimed that they don't need to specify the real distribution $p(\boldsymbol{Z})$; instead, it is only required to specify a proxy $q(\boldsymbol{Z})$ to $p(\boldsymbol{Z}|\boldsymbol{X})$. In our representation learning lingo, this $q(\boldsymbol{Z})$ is the representation for $p(\boldsymbol{Z}|\boldsymbol{X})$. And the goodness of the representation is assessed by the K-L divergence between the proxy $q(\boldsymbol{Z})$ and the true $p(\boldsymbol{Z}|\boldsymbol{X})$.

Since

$$\mathbb{KL}(q(\boldsymbol{Z})||p(\boldsymbol{Z}|\boldsymbol{X})) = -\mathbb{E}_q[logp(\boldsymbol{Z},\boldsymbol{X}) - logq(\boldsymbol{Z})] + logp(\boldsymbol{X}) \tag{3.15}$$

$$\Rightarrow logp(\boldsymbol{X}) - \mathbb{KL}(q(\boldsymbol{Z})||p(\boldsymbol{Z}|\boldsymbol{X})) = \mathbb{E}_q[logp(\boldsymbol{Z},\boldsymbol{X}) - logq(\boldsymbol{Z})], \tag{3.16}$$

minimizing $\mathbb{KL}(q(\boldsymbol{Z})||p(\boldsymbol{Z}|\boldsymbol{X}))$ is equivalent to maximizing the evidence lower bound (ELBO), $\mathbb{E}_q[logp(\boldsymbol{Z},\boldsymbol{X}) - logq(\boldsymbol{Z})]$. Furthermore, it is interpreted that, since ELBO is the lower bound of $logp(\boldsymbol{X})$, maximizing the ELBO is identical to maximizing the log-likelihood. So it's the best of both worlds – not only they find a way to perform Bayesian inference in a computationally-efficient fashion, they also preserve the good tradition from classical Statistics (the MLE).

However, there is no free lunch. We would like to point out two cautions for this interpretation. First, for generative model learning, it may be sufficient to use typical MLE as the learning objective, since the purpose of MLE is for inference (under the correct model specification); nevertheless, for representation learning, MLE may not suffice to be the optimum objective. Second, in this framework, people often overlook the importance of the population $p(\boldsymbol{X},\boldsymbol{Z})$ being fixed. As mentioned earlier, if we only observe $\boldsymbol{X}$, there is no way to identify one unique $\boldsymbol{Z}$. Thus resulting non-unique joint distribution $p(\boldsymbol{X},\boldsymbol{Z})$. If $p(\boldsymbol{X},\boldsymbol{Z})$ is not unique, ELBO is not unique, and it is unfair to compare across different models with different lower bounds.

A similar argument was proposed by [fixing the broken ELBO], in which the authors argued that the learning of representations in the context of variational inference should be expanded into a two-dimensional problem by both considering the reconstruction and the distortion rates. In our point of view, this is the same as saying the key population quantity of interest is no longer $p(\boldsymbol{X})$ alone, but the joint $p(\boldsymbol{X},\boldsymbol{Z})$. In the ELBO, the first term gives us the first dimension, which is the reconstruction of observable $\boldsymbol{X}$, given a chosen $\boldsymbol{Z}$, and the second term is about choosing one optimum $\boldsymbol{Z}$.

So far, although both EM and VI claim to be able to learn a latent representation based on a pre-specified $\boldsymbol{Z}$, we still need to clarify the difference between generative model learning and representation learning: If the learning goal is only pertaining to the marginal distribution of $p(\boldsymbol{X})$, any $p(\boldsymbol{Z})$ may be useful, since we will marginalize $p(\boldsymbol{Z})$ anyway. However, if the learning goal is to learn a good or even an optimum latent variable, Equation 3.14 formulation does not suffice. In fact, there is no one unique $p(\boldsymbol{Z})$, and there is no way for the machine to automatically learn a "meaningful" $p(\boldsymbol{Z})$ if we do not put any constraints on it. Furthermore, how "meaningful" $p(\boldsymbol{Z})$ really depends on how we want to use the latent variables for. This once again emphasizes the importance of pre-specifying the meaning of the representation before the learning starts.

In the following chapters, we will focus on partition-based models and the design the proper objective function for representation learning. In our setting, the latent variable creates partitions on the feature space. Conditioned on each partition, the distribution of the data is homogenous.

3.9 Summary

In this chapter, we proposed our definition for a statistically desirable data representation, and developed a learning framework for the representation. We will use this framework to further implement a learning methodology incorporating the I-Score, and also use the framework to examine other existing learning methodologies in the coming chapters.

Chapter 4: I-Score Sufficient Representation Learning

4.1 Partition-based Representation Learning

From a black-box model's point of view, one of the major challenges of modeling is to tell signals apart from noises. The culprit of this problem stems from (1) the observed data always contains noise (although this is an assumption, it is hard to be refuted) (2) the paradigm lets data to select the model, instead of the other way around. The result of learning from this paradigm is often over-fitting. From statistical inference viewpoint, (1) is also an accepted assumption, and people avoid (2) by choosing a pre-fixed model. However, although the statistical thinking is more "careful", in practice, it is hard to not wanting to select the model after seeing the data. In addition, no one ever knows the "true" model, which adds on even more restriction on the usage of a pure statistical inference procedure in practice.

In this chapter, we consider partition-based representations and study them in detail. Partition-based models caught our attention because of their niche in performing model selection without model fitting. It prevents overfitting by restricting the class of the representations to be partitions in the feature space.

Another property of partition-based models is that although the idea is similar to latent variable models in the sense that each observed data is implicitly being assigned to a hidden cluster, the potential number of clusters in a partition-based model can grow with the sample size (i.e. not pre-fixed), which makes it a nonparametric method.

One of the existing commonly-seen partition-based models are the decision trees. However, trees ignore the structures inside the features, especially correlations between the features. Our proposed method has an advantage over tree-based methods in this regard.

Another distinction between our proposed method and tree-based methods is that our method can be implemented either top-down or bottom-up, while tree-based methods are always top-down – building models from one variable at first and gradually grow the tree into further branches.

4.2 Problem Setup

Assume the true data generating process is depicted as the graphical model shown in Figure 4.1. First, there are some generating factors $\boldsymbol{C}$, which then generates the observed data $\boldsymbol{X}$. Then, the labels Y are generated accordingly. The labels Y may or may not be observable. Also, labels can switch domains, such as switching from classifying the type of animal in the photo to classifying the season in which the photo was taken.

The representation $\boldsymbol{Z}$ learning is illustrated in Figure 4.2. The desired representation $\boldsymbol{Z}$ should satisfy the conditional independence property, $Y \perp\!\!\!\perp \boldsymbol{X}|\boldsymbol{Z}$. We will find an objective function that encourages this learning process.

Depending on the task and the corresponding objective function, the sufficient representation changes accordingly. Let's take two examples:

4.2.1 Inference

If the task is to perform inference and the corresponding objective function is the likelihood function, then we have the following decomposition, when we have pairs of data $(\boldsymbol{X_i}, Y_i)_{i=1}^N$, where N is the sample size, C the true data generating factors and Z, the learned representations.

$$
\begin{aligned}
p(Y_i, \boldsymbol{X_i}) &= \int_Z p(Y_i, \boldsymbol{X_i}, \boldsymbol{Z}) dZ && (4.1)\\
&= \int_Z p(Y_i|\boldsymbol{X_i}, \boldsymbol{Z}) p(\boldsymbol{Z}|\boldsymbol{X_i}) p(\boldsymbol{X_i}) dZ && (4.2)\\
&= \int_Z p(Y_i|\boldsymbol{Z}) p(\boldsymbol{Z}|\boldsymbol{X_i}) p(\boldsymbol{X_i}) dZ && (4.3)\\
&= p(\boldsymbol{X_i}) \cdot \int_Z p(Y_i|\boldsymbol{Z}) p(\boldsymbol{Z}|\boldsymbol{X_i}) dZ && (4.4)
\end{aligned}
$$

Let me explain the above steps in detail: in Equation 4.1, this is the result of assuming that the data generation process follows Figure 4.1. Equation 4.2 is the ideal decomposition of the data generating process after our proposed representation learning, illustrated in Figure 4.2. Equation 4.3 follows from the desired property of representation learning; that is, the conditional independence property, $Y \perp\!\!\!\perp \boldsymbol{X}|\boldsymbol{Z}$. The result in Equation 4.4 shows that, by introducing such a representation, to model the joint distribution $p(Y_i, \boldsymbol{X_i})$, it can actually be broken into modeling $p(\boldsymbol{X_i})$, $p(Y_i|\boldsymbol{Z})$, and $p(\boldsymbol{Z}|\boldsymbol{X_i})$, separately.

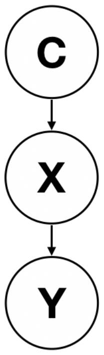


Figure 4.1: Data generating assumption.

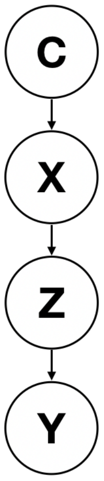


Figure 4.2: Proposed decomposition of representation learning.

4.2.2 Prediction

If the task is to predict future values of Y and the corresponding objective function is the empirical square loss. We know from statistical theory that the conditional expectation is the best predictor under square loss. That is,

$$\mathbb{E}(Y|X) \in \underset{g(x)}{\operatorname{argmin}}\, \mathbb{E}[(Y - g(X))^2] \tag{4.5}$$

If we have a representation Z such that $\mathbb{E}(Y|X) = \mathbb{E}(Y|Z)$, the representation is sufficient for prediction through squared loss.

4.3 Sufficient Gaussian-Partition Model – Unsupervised

In this section, we will demonstrate the proposed representation learning in detail using a parametric Gaussian mixture model.

4.3.1 Model Assumption and Learning Objective

In a Gaussian mixture model, we have observable random variable X and unobservable latent partition/cluster [1] assignment mechanism Z, where $X|Z$ is one-dimensional Gaussian, and θ_k is the corresponding parameter(s). We also assume that Z itself follows a categorical distribution, and $p(Z = \pi_k) = \eta_k$ is the corresponding parameter(s). Here the representation is $X|Z$.

When fitting the model, we have data $\{X_i\}_{i=1}^{N}$, where N is the sample size. We assume model M_k to be a k-partitioned model, $1 \leq k \leq n$, over which there is a categorical distribution. Here we assume that there exists a partition model such that the noises within partitions are iid from the same distribution, while the noises between different partitions are independent. Since we don't observe Z, we need to estimate the number of k. We will do this as part of the model selection.

[1] In this example, we will use these two terms interchangeably.

4.3.2 Example

Here we demonstrate an unsupervised learning example. The objective function is the log likelihood function. The data generating process is illustrated in Figure 4.3

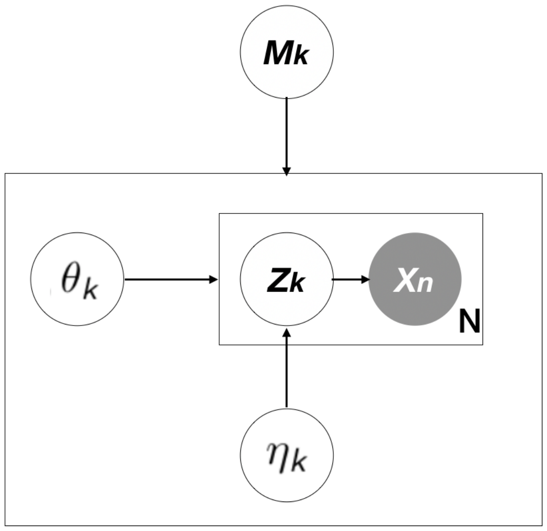


Figure 4.3: Data generating process assumption (unsupervised).

1. **Define the task, the corresponding objective function, and the desired representation.**

   We would like to cluster the observed X's into several homogeneous partitions, such that, conditioned on each partition, the distribution of the local observations follows a Gaussian distribution. We would like to use maximum likelihood as the objective function.

2. **Define the joint distribution of every variable and parameter, and decompose the like-**

lihood up to the desired level of hierarchy. Choose prior(s) if needed.

$$p(X_n, Z_k, \theta_k, \eta_k, M_k) = p(X_n|Z_k, \theta_k, \eta_k, M_k)p(Z_k|\theta_k, \eta_k, M_k)p(\theta_k|\eta_k, M_k)p(\eta_k|M_k)p(M_k) \tag{4.6}$$

$$= p(X_n|Z_k, \theta_k, M_k)p(Z_k|\eta_k, M_k)p(\theta_k|M_k)p(\eta_k|M_k)p(M_k) \tag{4.7}$$

$$= p(X_n|Z_k, \theta_k, M_k)p(Z_k|\eta_k, M_k)p(M_k) \tag{4.8}$$

$$= \Pi_k[[N(X_n|\mu_k, \sigma_k) \cdot p(Z_i = k|\eta_k, M_k)]^{I(Z_i=k)} \cdot p(M_k) \tag{4.9}$$

We assume that we don't have particular prior information to consider, so we dropped the prior terms for the hyperparameters, $p(\theta_k|M_k)$ and $p(\eta_k|M_k)$; however, we keep the prior on the model selection, namely, $p(M_k)$, as a controlling factor of model complexity. As a result, the marginal likelihood is $p(X_n|Z_n, \theta_k, M_k)$, and the priors are $p(Z_n|\eta_k, M_k)$ and $p(M_k)$.

Therefore, the log likelihood is

$$\log p(X_1, ..., X_n|\mu_k, \sigma_k, \eta_k.Z, M_k) \tag{4.10}$$

$$= \sum_i \log[\Pi_k[N(X_n|\mu_k, \sigma_k) \cdot p(Z_i = k|\eta_k, M_k)]^{I(Z_i=k)}] + \log p(M_k) \tag{4.11}$$

$$= \sum_i \sum_k I(Z_i = k) \log(\eta_k N(X_i|\mu_k, \sigma_k)) + \log p(M_k) \tag{4.12}$$

$$= \sum_i \{\sum_{k=1}^{K} I(Z_i = k) \log(\eta_k) + \sum_{k=1}^{K} I(Z_i = k) \log[N(X_i|\mu_k, \sigma_k)]\} + \log p(M_k) \tag{4.13}$$

$$= \sum_{k=1}^{K} n_k \log(\eta_k) + \sum_i \sum_{k=1}^{K} I(Z_i = k) \log[N(X_i|\mu_k, \sigma_k)] + \log p(M_k) \tag{4.14}$$

Note that the uncertainties considered here are (1) model uncertainty (how many clusters) and (2) uncertainty of assigning each data point to which cluster.

For model uncertainty, we propose a prior based on Stirling number of the second kind (or Stirling partition number). It is proportional to $\sum_{i=1}^{k} n_i^2$, where k is the number of clusters in model M_k. We will show more explanation about this prior in later section.

For assignment uncertainty, we try some possible assignments and use empirical estimation. Contrasted to common practice, in which people estimate the assignment uncertainty by the posterior probability, $p(Z_i = k|X_i)$, we use an algorithm called backward dropping to search for possible assignments.

3. **Define several candidate models, inside which the desired representation should be specified.**

   Here the candidate model M_k is the partition model with k partitions, where k can potentially be from 1 to N, the sample size. In other words, on one extreme case, there is one cluster for each data point; on the other side, all the data points belong to the same cluster. The representation inside each cluster is the Gaussian mixture components. That is, $p(X|Z = k)$ follows Normal(μ_k, σ_k^2).

4. **According to the objective function, develop an algorithm.**

In this example, we are able to further simplify Equation 4.14 – To eliminate Z from Equation 4.14, we can either (1) take expectation with respect to it, or (2) use empirical estimate to substitute for it. Here we can only do (2), since each assignment is hard assignment and observed in the sense that we try possible combinations all out.

When X is univariate, Equation 4.14 equals

$$\log p(X_1, ..., X_n|\mu_k, \sigma_k, \eta_k.Z, M_k) \tag{4.15}$$

$$= \sum_{k=1}^{K} n_k \log(\eta_k) + \sum_i \sum_{k=1}^{K} I(Z_i = k) \log[N(X_i|\mu_k, \sigma_k)] + \log p(M_k) \tag{4.16}$$

$$= \sum_{k=1}^{K} n_k \log(\eta_k) - \sum_i \sum_{k=1}^{K} I(Z_i = k)[\frac{n_k}{2} \log \sigma_k^2 + \frac{(X_i - \mu_k)^2}{2\sigma_k^2}] + \log p(M_k) \tag{4.17}$$

Then the objective function for selecting the optimum model is

$$M^* = \underset{M_k}{\operatorname{argmax}} \log p(X_1, ..., X_n | \hat{\mu_k}, \hat{\sigma_k}, \hat{\eta_k}, \hat{Z}, M_k) \tag{4.18}$$

$$= \underset{M_k}{\operatorname{argmax}} \sum_{k=1}^{K} n_k \log(\hat{\eta_k}) + \sum_{i=1}^{N} \sum_{k=1}^{K} I(\hat{Z_i} = k) \log[N(X_i | \hat{\mu_k}, \hat{\sigma_k})] + \log p(M_k) \tag{4.19}$$

$$= \underset{M_k}{\operatorname{argmax}} \sum_{k=1}^{K} n_k \log(\hat{\eta_k}) - \sum_{i=1}^{N} \sum_{k=1}^{K} I(\hat{Z_i} = k)[\frac{n_k}{2} \log \hat{\sigma_k^2} + \frac{(X_i - \hat{\mu_k})^2}{2\hat{\sigma_k^2}}] + \log p(M_k) \tag{4.20}$$

$$= \underset{M_k}{\operatorname{argmax}} \sum_{k=1}^{K} n_k \log(\hat{\eta_k}) - \sum_{i=1}^{N} \sum_{k=1}^{K} I(\hat{Z_i} = k)[\frac{n_k}{2} \log \hat{\sigma_k^2} + \frac{(X_i - \hat{\mu_k})^2}{2\hat{\sigma_k^2}}] + \log \frac{\sum_k n_k^2}{N}, \tag{4.21}$$

where $\hat{\eta_k} = \frac{n_k}{N}$, $\hat{\mu_k} = \sum_{i=1}^{n_k} \frac{X_i}{n_k}$, $\hat{\sigma_k^2} = \sum_{i=1}^{n_k} \frac{(X_i - \hat{\mu_k})^2}{n_k}$, and $\hat{Z_i}$ can done by simulating possible conditions.

So the basic algorithm is, we first select a random partition model, and then calculate the corresponding log-likelihood 4.52. Iterate this process several times and find out the one partition model with the highest log-likelihood 4.52 – this yields the optimum M^*. Once we find the optimum M^*, we can calculate the corresponding $\hat{\eta_k} = \frac{n_k}{N}$, $\hat{\mu_k} = \sum_{i=1}^{n_k} \frac{X_i}{n_k}$, and $\hat{\sigma_k^2} = \sum_{i=1}^{n_k} \frac{(X_i - \hat{\mu_k})^2}{n_k}$.

When the sample size is small, it is possible to iterate through all possible partitions (exhaustive search); otherwise, we first select a random partition of a pre-defined size, and then perform backward dropping algorithm within each iteration. Every time only the one model with the highest log-likelihood will survive. Finally we can compare all the iteration results and find the one with the highest log-likelihood.

Numerical example:

Suppose our data is $\{1, 2, 3, 4, 5, 6\}$ and we want to come up with a reasonable prior such that it can promote the middle-ground selection of number of clusters.

Intuitively, we know both ends of the extremes – treating all 6 points as one big cluster, or, contrarily, treating each point as a group by itself – are bad options, for one being under-fitting and the other over-fitting. Note that we are not saying it is impossible for either of them to be the "ground truth" – we are just saying that we want to construct a prior based on the data to encourage something not too extreme. Afterwards it will all depend on the practitioner's use, and our prior can serve at least as a reasonable initial guess.

If we consider all possible partitions, and calculate the corresponding value of Equation 4.52, we get the top-rated partitions as shown in Figure 4.4, which seems indeed doing what we would like to achieve:

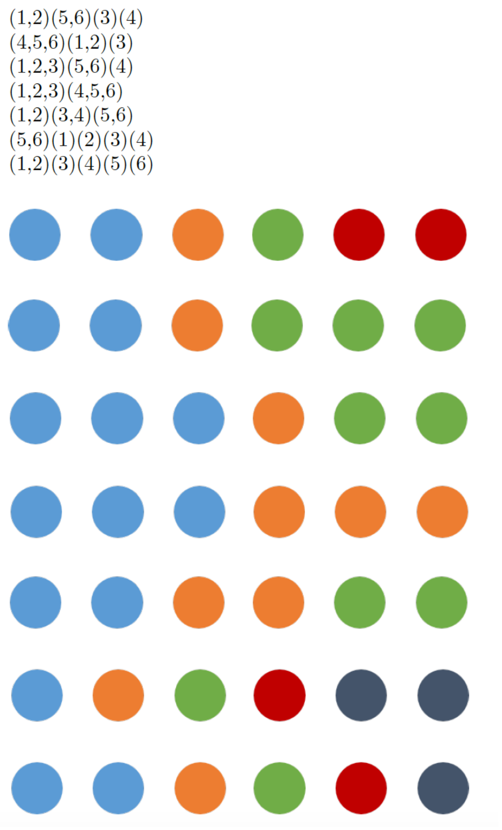


Figure 4.4: Partitions ranked by the proposed procedure.

4.3.3 Stirling prior

The motivation of the problem is – how to create an empirical distribution based on the number of observations in each partition that has a (slightly right-skewed,) bell-shape like distribution? If the prior is proportional to the sum of the counts, it stays constant, no matter which model is under consideration. Or we may call this the uniform prior. However, if we take the square sum or the sum of $n \log n$, the distribution will be something that gives the peak in the middle of the distribution. Here is one demonstration of the distribution of the sum of some power of the number of counts in each partition using ten observations:

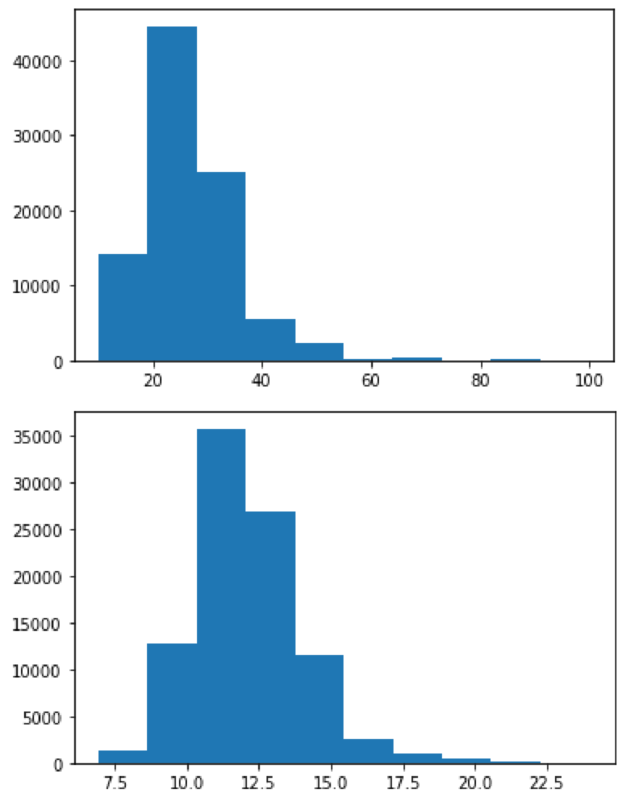


Figure 4.5: Histograms on the number of partitions.

One intuition of the distribution of the sum of squares when the sum is a constant can be shown graphically in the Figure 4.6:

Each side of the square stands for the sum of the lengths, which is a constant, and the sum of the shaded square areas are the sum of the square of the lengths. Obviously there are more possibilities when the partitions aren't on the extreme ends; thus putting more prior on the distribution that emphasizes not on the extremes. Also, one can think of this as partitioning the co variance matrix into several clusters.

Also note that the distributions above is independent from the value of the observations. In other words, the prior on the models $p(M_i)$ is motivated from the data, but it is not used for parameter estimation, and thus it won't cause overfitting.

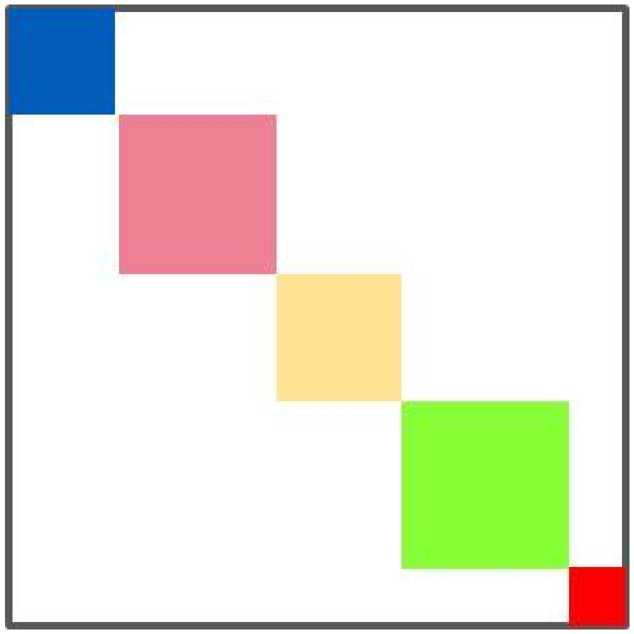


Figure 4.6: Intuition for the partition selection mechanism.

Using this neat trick to set the prior on models can help selecting the model "in the middle" and slightly towards the lower end. This helps prevent either overfitting or underfitting.

4.4 Sufficient Gaussian-Partition Model – Supervised

4.4.1 Notation

The data is a set of $\{(Y_i, \boldsymbol{X_i})\}_{i=1}^N$. Y is the label. $\boldsymbol{X} \in \mathbb{R}^p$ is the raw feature set consisting of p features. Here we consider all the features are discrete. If we composite all the p features, we obtain partitions over the feature space, π_1, π_2,..., π_K, where K is the total number of partitions. For example, if each feature takes 2 discrete values, then $K = 2^p$. We use Π to denote the random variable that has a categorical distribution over the partitions. We also assume that, given p features, not all of these features are necessary for prediction purpose. That being said, if we take a subset of m features, where $m < k$, the corresponding model will have less than K partitions. We use model [2] M_k to denote the k-partitioned model.

[2]Note that here what we mean by "model" is referring to the corresponding number of partitions in that representation, instead of the usual way people use in Statistics lingo.

4.4.2 Model Assumption and Learning Objective

Assume there exists a function $f(X)$ such that (one may interpret such $f(\cdot)$ as a hidden feature constructed from X) the following joint distribution and the decomposition holds:

$$
\begin{aligned}
p(Y_n, X_n, \theta_k, \eta_k, M_k) &= p(Y_n, X_n, f(X_n), \theta_k, \eta_k, M_k) && (4.22)\\
&= p(Y_n, X_n | f(X_n), \theta_k, \eta_k, M_k) p(f(X_n)|\theta_k, \eta_k, M_k) p(\theta_k|\eta_k, M_k) p(\eta_k|M_k) p(M_k) && (4.23)\\
&= p(Y_n, X_n | f(X_n), \theta_k, M_k) p(f(X_n)|\eta_k, M_k) p(\theta_k|M_k) p(\eta_k|M_k) p(M_k) && (4.24)\\
&= p(Y_n, X_n | f(X_n), \theta_k, M_k) p(f(X_n)|\eta_k, M_k) p(M_k) && (4.25)\\
&= p(Y_n | X_n, f(X_n), \theta_k, M_k) p(X_n | f(X_n), M_k) p(f(X_n)|\eta_k, M_k) p(M_k) && (4.26)\\
&= p(Y_n | f(X_n), \theta_k, M_k) p(f(X_n)|\eta_k, M_k) p(M_k) && (4.27)\\
& && (4.28)
\end{aligned}
$$

where θ_k is the parameters for $Y|X$ in model M_k, and η_k is the parameters for $f(X)$ in model M_k.

The data generating process is illustrated in the graphical model shown in Figure 4.7

The term $p(X_n|f(X_n), M_k)$ represents the amount of information lost when we use $f(X_n)$ only to predict Y, instead of using X_n. When $p(X_n|f(X_n), M_k) = 1$, it means that there is no information loss by just using $f(X_n)$. Such a condition can be achieved by the partition model.

In Equation 4.27, we assumed that the desirable latent variable $f(X_n)$ can satisfy $Y_n \perp\!\!\!\perp X_n|f(X_n)$. If $f(X_n)$ has lesser dimension than X, then it serves as a dimensional reduction technique. One example would be, if we first perform PCA on all the predictors X and then regress Y on the reconstructed X' (reduced dimension).

There are two tasks that we may be interested in, one is variable selection through Z, the other being prediction. Regarding to prediction, according to Bayesian model averaging theory, the

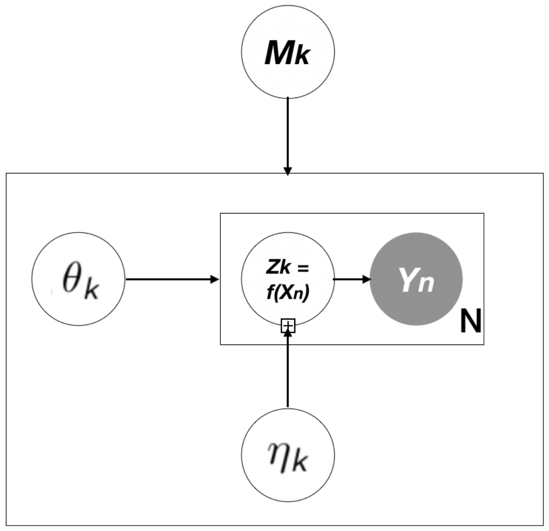


Figure 4.7: Data generating process assumption (supervised).

optimum prediction under square loss is the average posterior mean

$$\mathbb{E}(Y|Y_n, X_n) = \sum_{M_k} \mathbb{E}(Y|M_k, Y_n, X_n) p(M_k|Y_n, X_n) \tag{4.29}$$

In practice, we may substitute the model averaging part with the optimum model selected –

$$\begin{aligned} \mathbb{E}(Y|Y_n, X_n) &= \sum_{M_k} \mathbb{E}(Y|M_k, Y_n, X_n) p(M_k|Y_n, X_n) && (4.30) \\ &\approx \mathbb{E}(Y|M^*, Y_n, X_n) && (4.31) \end{aligned}$$

which is equivalent to the empirical Bayesian procedure.

4.4.3 I-score for Supervised Variable Selection With Discrete Features

Model Assumption

The scenario we consider is – We have data $\{(\boldsymbol{X_1}, Y_1)..., (\boldsymbol{X_n}, Y_n)\}$, where $\boldsymbol{X}$ is the feature/predictor and Y is the response/label. $\boldsymbol{X} \in \mathbb{R}^p$ and all of them are discrete. Our model assumption is that, conditioned on model M_k, there are k hidden partitions which are formed by the composition of a subset of $\boldsymbol{X}$, $1 \leq k \leq |\Pi_X|$. Observations Y that are from the same partition are i.i.d. from some $N(\mu_k, \sigma_k^2)$ distribution. Unlike the unsupervised counterpart, where the partitions are completely clueless, here we have the features $\boldsymbol{X}$ to guide the selection of partitions.

Algorithm

1. **Define the task, the corresponding objective function, and the desired representation.**

   We would like to cluster the observed Y's into several homogeneous partitions, such that, conditioned on each partition, the distribution of the local observations follows a Gaussian distribution. We would like to use maximum a posteriori as the objective function.

   $p(Y_n|\hat{\mu_k}, \hat{\sigma_k}, \hat{\eta_k}, \hat{Z_k}, M^*)$ is the representation, and we would like to learn the optimum $f(\cdot)^*$ from MAP:

$$f(\cdot)^* = \text{argmax}_{f(X_i)}\, p(f(X_i)|Y_1, ..., Y_n) \tag{4.32}$$

$$\propto \Pi_i \Pi_k [N(Y_i|\mu(f(X_i)), \sigma^2) \cdot p(f(X_i) = k|\eta_i)]^{\mathbb{I}(f(X_i)=k)} \cdot p(M_k) \tag{4.33}$$

2. **Define the joint distribution of every variable and parameter, and decompose the likelihood up to the desired level of hierarchy. Choose prior(s) if needed.**

$$
\begin{aligned}
p(Y_n, X_n, \theta_k, \eta_k, M_k) &= p(Y_n, X_n, f(X_n), \theta_k, \eta_k, M_k) && (4.34)\\
&= p(Y_n, X_n | f(X_n), \theta_k, \eta_k, M_k) p(f(X_n) | \theta_k, \eta_k, M_k) p(\theta_k | \eta_k, M_k) p(\eta_k | M_k) p(M_k) && (4.35)\\
&= p(Y_n, X_n | f(X_n), \theta_k, M_k) p(f(X_n) | \eta_k, M_k) p(\theta_k | M_k) p(\eta_k | M_k) p(M_k) && (4.36)\\
&= p(Y_n, X_n | f(X_n), \theta_k, M_k) p(f(X_n) | \eta_k, M_k) p(M_k) && (4.37)\\
&= p(Y_n | X_n, f(X_n), \theta_k, M_k) p(X_n | f(X_n), M_k) p(f(X_n) | \eta_k, M_k) p(M_k) && (4.38)\\
&= p(Y_n | f(X_n), \theta_k, M_k) p(f(X_n) | \eta_k, M_k) p(M_k) && (4.39)\\
&= N(Y_i | \mu(f(X_i)), \sigma^2) \cdot p(f(X_i) = k | \eta_k, M_k) && (4.40)
\end{aligned}
$$

where θ_k is the parameters for $Y|X$ in model M_k, and η_k is the parameters for $f(X)$ in model M_k. $1 \leq k \leq |\Pi_x|$, and the distribution of $p(f(X_n))$ is a categorical distribution where $\eta_k = \{p_1, ..., p_k\}$. The distribution of $p(Y_n | f(X_n), \theta, M_k)$ is assumed to be Gaussian with parameter $\theta_k = \{\mu_k, \sigma^2\}$. We dropped the prior on the model selection, $p(M_k)$ because of the presence of X. Here the prior isn't as crucial.

Next, let us write down the posterior (likelihood) as

$$
p(f(X_i) | Y_1, ..., Y_n) \propto \Pi_i \sum_k [N(Y_i | \mu(f(X_i)), \sigma^2) \cdot p(f(X_i) = k | \eta_i)]^{\mathbb{I}(f(X_i) = k)} \quad (4.41)
$$

Therefore, the log posterior likelihood is

$$\log p(f(X_i)|Y_1,...,Y_n) \propto \sum_i \log[\sum_k [N(Y_i|\mu(f(X_i)),\sigma^2)\cdot p(f(X_i)=k|\eta_i)]^{\mathbb{I}(f(X_i)=k)}] \tag{4.42}$$

$$= \sum_i \sum_k I(Z_i = k)\log(\eta_k N(X_i|\mu_k,\sigma_k)) + \log p(M_k) \tag{4.43}$$

$$= \sum_i \{\sum_{k=1}^K I(Z_i = k)\log(\eta_k) + \sum_{k=1}^K I(Z_i = k)\log[N(X_i|\mu_k,\sigma_k)]\} + \log p(M_k) \tag{4.44}$$

$$= \sum_{k=1}^K n_k \log(\eta_k) + \sum_i \sum_{k=1}^K I(Z_i = k)\log[N(X_i|\mu_k,\sigma_k)] + \log p(M_k) \tag{4.45}$$

Note that the uncertainties considered here are (1) model uncertainty (how many partitions) and (2) uncertainty of assigning each data point to which partition.

Still, just as the unsupervised case, for assignment uncertainty, we use the backward dropping algorithm to search for possible assignments.

3. **Define several candidate models, inside which the desired representation should be specified.**

   Here the candidate model M_k is the partition model with k partitions, where k can potentially be from 1 to N, the sample size. In other words, on one extreme case, there is one partition for each data point; on the other side, all the data points belong to the same partition. The representation inside each model is the Gaussian mixture components. That is, $p(Y|f(\boldsymbol{X}) = k)$ follows Normal(μ_k, σ_k^2).

4. **According to the objective function, develop an algorithm.**

   In this example, we are able to further simplify Equation 4.14 – To eliminate Z from Equation 4.14, we can either (1) take expectation with respect to it, or (2) use empirical estimate to substitute for it. Here we can only do (2), since each assignment is hard assignment and observed in the sense that we try possible combinations all out.

When X is univariate, Equation 4.14 equals

$$\log p(X_1, ..., X_n|\mu_k, \sigma_k, \eta_k.Z, M_k) \tag{4.46}$$

$$= \sum_{k=1}^{K} n_k \log(\eta_k) + \sum_{i} \sum_{k=1}^{K} I(Z_i = k) \log[N(X_i|\mu_k, \sigma_k)] + \log p(M_k) \tag{4.47}$$

$$= \sum_{k=1}^{K} n_k \log(\eta_k) - \sum_{i} \sum_{k=1}^{K} I(Z_i = k)[\frac{n_k}{2} \log \sigma_k^2 + \frac{(X_i - \mu_k)^2}{2\sigma_k^2}] + \log p(M_k) \tag{4.48}$$

Then the objective function for selecting the optimum model is

$$M^* = \underset{M_k}{\operatorname{argmax}} \log p(X_1, ..., X_n|\hat{\mu_k}, \hat{\sigma_k}, \hat{\eta_k}, \hat{Z}, M_k) \tag{4.49}$$

$$= \underset{M_k}{\operatorname{argmax}} \sum_{k=1}^{K} n_k \log(\hat{\eta_k}) + \sum_{i=1}^{N} \sum_{k=1}^{K} I(\hat{Z_i} = k) \log[N(X_i|\hat{\mu_k}, \hat{\sigma_k})] + \log p(M_k) \tag{4.50}$$

$$= \underset{M_k}{\operatorname{argmax}} \sum_{k=1}^{K} n_k \log(\hat{\eta_k}) - \sum_{i=1}^{N} \sum_{k=1}^{K} I(\hat{Z_i} = k)[\frac{n_k}{2} \log \hat{\sigma_k^2} + \frac{(X_i - \hat{\mu_k})^2}{2\hat{\sigma_k^2}}] + \log p(M_k) \tag{4.51}$$

$$= \underset{M_k}{\operatorname{argmax}} \sum_{k=1}^{K} n_k \log(\hat{\eta_k}) - \sum_{i=1}^{N} \sum_{k=1}^{K} I(\hat{Z_i} = k)[\frac{n_k}{2} \log \hat{\sigma_k^2} + \frac{(X_i - \hat{\mu_k})^2}{2\hat{\sigma_k^2}}] + \log \frac{\sum_k n_k^2}{N}, \tag{4.52}$$

where $\hat{\eta_k} = \frac{n_k}{N}$, $\hat{\mu_k} = \sum_{i=1}^{n_k} \frac{X_i}{n_k}$, $\hat{\sigma_k^2} = \sum_{i=1}^{n_k} \frac{(X_i - \hat{\mu_k})^2}{n_k}$, and $\hat{Z_i}$ can done by simulating possible conditions.

So the basic algorithm is, we first select a random partition model, and then calculate the corresponding log-likelihood 4.52. Iterate this process several times and find out the one partition model with the highest log-likelihood 4.52 – this yields the optimum M^*. Once we find the optimum M^*, we can calculate the corresponding $\hat{\eta_k} = \frac{n_k}{N}$, $\hat{\mu_k} = \sum_{i=1}^{n_k} \frac{X_i}{n_k}$, and $\hat{\sigma_k^2} = \sum_{i=1}^{n_k} \frac{(X_i - \hat{\mu_k})^2}{n_k}$.

When the sample size is small, it is possible to iterate through all possible partitions (exhaustive search); otherwise, we first select a random partition of a pre-defined size, and then perform backward dropping algorithm within each iteration. Every time only the one model with the highest log-likelihood will survive. Finally we can compare all the iteration results and find the one with the highest log-likelihood.

Chapter 5: Representation Learning In Genome-wide Association Studies

5.1 Introduction

In this chapter, we will implement the learning framework outlined in the previous chapter on real data. The application we consider here is the genome-wide association studies (GWAS). A genome-wide association study (GWAS) is an approach used in genetics research to associate specific genetic variations with particular diseases. The method involves scanning the genomes from many different people and looking for genetic markers that can be used to predict the presence of a disease. Once such genetic markers are identified, they can be used to understand how genes contribute to the disease and develop better prevention and treatment strategies [44].

Statistically, there are many challenges involved in modeling GWAS, such as the high-dimensionality (about 500 thousands to 5 millions of SNP's), non i.i.d. sample (due to family structures), and lack of sample size. Here we will address the family structure problem by attempting different mixed effect models.

Note: This chapter is adapted from Hsu et al. [45].

5.2 GWAS Background

To carry out a genome-wide association study, researchers use two groups of participants: people with the disease being studied and similar people without the disease. Researchers obtain DNA from each participant, usually by drawing a blood sample or by rubbing a cotton swab along the inside of the mouth to harvest cells.

Each person's complete set of DNA, or genome, is then purified from the blood or cells, placed on tiny chips and scanned on automated laboratory machines. The machines quickly survey each participant's genome for strategically selected markers of genetic variation, which are called single nucleotide polymorphisms, or SNPs.

If certain genetic variations are found to be significantly more frequent in people with the disease compared to people without disease, the variations are said to be "associated" with the disease. The associated genetic variations can serve as powerful pointers to the region of the human genome where the disease-causing problem resides.

However, the associated variants themselves may not directly cause the disease. They may just be "tagging along" with the actual causal variants. For this reason, researchers often need to take additional steps, such as sequencing DNA base pairs in that particular region of the genome, to identify the exact genetic change involved in the disease.

The impact on medical care from genome-wide association studies could potentially be substantial. Such research is laying the groundwork for the era of personalized medicine, in which the current one size-fits-all approach to medical care will give way to more customized strategies.In the future, after improvements are made in the cost and efficiency of genome-wide scans and other innovative technologies, health professionals will be able to use such tools to provide patients with individualized information about their risks of developing certain diseases. The information will enable health professionals to tailor prevention programs to each person's unique genetic makeup. In addition, if a patient does become ill, the information can be used to select the treatments most likely to be effective and least likely to cause adverse reactions in that particular patient.

5.3 Statistical Problem

It is well known that genome-wide association studies (GWAS) may lead to spurious findings if one fails to address the dependence among individuals such as that resulting from family structure. If the true dependence structure is known, the best practice is to explicitly incorporate this information into the analysis. However, the true dependence structure is rarely available. Consequently, different strategies have been proposed to address this issue [46] [47] [48] [49] [50] [51] [52] [53] [54] [55] [56] [57] [58].

Statistically, we use multilevel models to model complicated family structures. Multilevel models, also known as variance component models, random effects models, or hierarchical linear models, have seen rapid growth and development in many different fields. Multilevel models provide a flexible regression modeling framework for handling data sampled from clustered population structures, such as students within classes that are within schools, patients within hospitals, repeated measurements within individuals, or children within families. Ignoring the multilevel structure of the data can lead to incorrect inferences that result from underestimating the standard errors for the regression coefficients.

Linear mixed-effect regression models assume the family effect to be a random effect. The covariance structure for the random effect is generally assumed to correspond to that implied by a polygenic model, incorporating the genetic relationship (kinship) between each pair of individuals. Although the use of this linear mixed-effect regression model was originally proposed for pedigrees with known relationships [**4**] [47] [48] [49] [50], this approach is popular for use with samples of unknown or uncertain relationship [51] [52] [53] [54] [55] [56] [57] [58], including apparently unrelated samples that may nevertheless display distant levels of common ancestry.

5.4 Data

The data set we use for this analysis is collected under the Genetics of Lipid Lowering Drugs and Diet Network (GOLDN) study. It was designed to identify genetic determinants of lipid response to two interventions: (1) a high-fat meal challenge and (2) fenofibrate treatment for 3 weeks [59]. The dataset only includes families with at least two siblings. This family information allow us for the analysis of family structural dependencies. Volunteers were required to withhold lipid-lowering agents (pharmaceuticals or nutraceuticals) for at least 4 weeks prior to their initial visit. A total of 1053 met all eligibility requirements. For the current study, we evaluated fasting triglyceride (TG) and very-low-density lipoprotein cholesterol among 991 participants for whom epigenetic data were available [60].

5.5 Models

Using a linear mixed-effect model, we model the response to both the fixed effects X and random effects Z as follows:

$$Y = X\beta + Z\mu + \epsilon, \text{ where} \tag{5.1}$$

$$\mu \sim N(0, \sigma_\mu^2 I_m) \tag{5.2}$$

$$\epsilon \sim N(0, \sigma_\epsilon^2 I_n) \tag{5.3}$$

where Y is a vector of beta scores (the response in this study) with mean $X\beta$; β is an unknown vector of fixed effects; μ is the vector of unknown random effects, with mean 0 and the variance– covariance matrix proportional to the kinship matrix; and ϵ is an unknown vector of random errors, with mean 0 and the variance–covariance matrix assumed to be proportional to the identity matrix. X and Z are the observable and latent design matrices respectively and I_m and I_n are identity matrices with corresponding sizes $m \times m$ and $n \times n$.

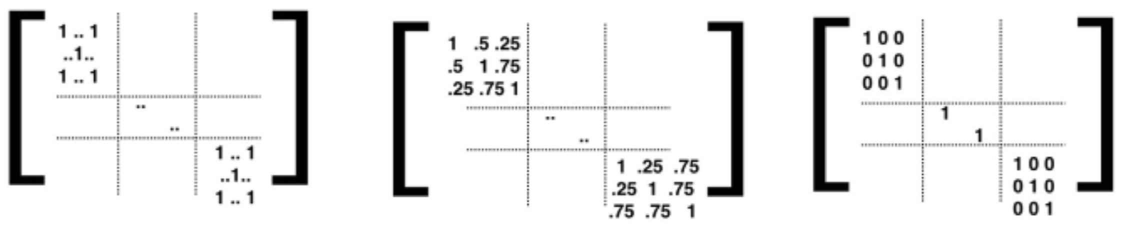
The variance–covariance matrix of Y is given by

$$\mathbf{Cov}(Y) = \sigma_{\mu}^{2} Z Z^{T} + \sigma_{\epsilon}^{2} I_{n} \tag{5.4}$$

Analysis of variance is one of the popular methods in the statistical literature to estimate the variance components σ_{μ}^{2} and σ_{ϵ}^{2}. However, as we discussed in the I-Score section, the decomposition of variance alone cannot be used as an objective function for selecting the representations between signal and noise. So we need other methods such as I-Score to put constraints on the representation selection mechanism. But the unique thing here for this dataset is the family structure, which can used to specify the latent random effects Z, is available.

So we assume that the latent random effects can totally be accounted by the family structure, and attempt three different strategies to model it. First, we recreate the Irvin study [58]: modeling the beta score as a function of TG level using mixed linear regression adjusted for age, gender, study site, and 4 methylation principal components as fixed effects (namely, X in Equation 5.1) and the family structure (Z in in Equation 5.1) as random effects. Throughout the 3 models we try, the matrices X are identical, as in [58]. We made this decision because our interest is in the evaluation of the covariance matrix introduced by the random effects Z, but not the fixed part X. Here, the kinship matrix is based on the theoretical estimates – the probability of sharing genetic relatedness. We refer to this ideal option as the kinship option. When the kinship information is not available, there are two other coping strategies: one that assumes everyone in this study is independent, ignoring family structure (denoted as the independence option), and one that uses only one randomly drawn representative individual from each family (referred as the representative option). While these 3 modeling options differ in the covariance matrix for the random effects, the fixed-effect parts remain the same. Figure 5.1 illustrates the assumed variance–covariance matrices.

Option 1, in expectation, is equivalent to using centroids for each family, which is not always



(a) (b) (c)

Figure 5.1: Kinship coefficient matrices used in 3 modeling options for fitting the linear mixed-effects model to data with family structures. (a) Option 1: representatives. (b) Option 2: kinship. (c) Option 3: independent.

feasible for all data types.

Under the assumption that the assumed model is correct, the important single-nucleotide polymorphisms (SNPs) identified by these 3 modeling options should have a nested structure as in Figure 5.2 in expectation, where the black box includes all SNPs and each circle includes the significant SNP sets concluded from each model. In practice, we also expect that the random variations in data and departures from model assumptions will lead to fewer overlapped identified SNP sets. If the analysis results from real data meet these expectations, it is only a matter of tradeoff between effective sample size and power. If any results from real data analysis deviates from this theoretical expectation, it might suggest the need for more detailed inspection of the coping strategies for family structures.

We use the *lmekin* function in *R* package *coxme* for the estimation of the mixed effect model and compare the results from the 3 modeling options. This is done by using user-specified variance functions. We use the 3 random effect structures explained previously in Figure 5.1 for the simulation. The results from the application to the GAW data set and a simple simulation study are shown and discussed in the next section.

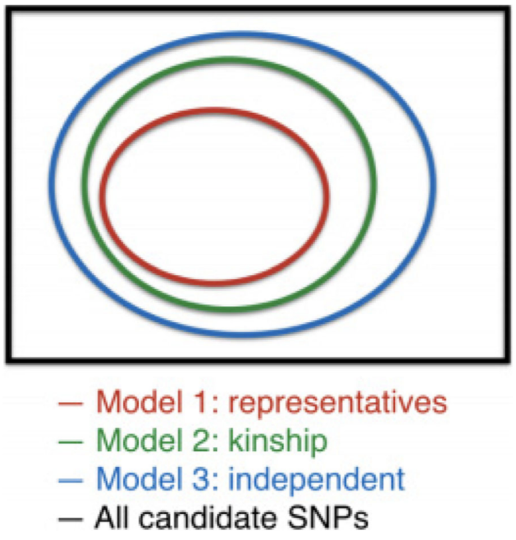


Figure 5.2: Ideal relation among significant SNP sets from 3 different modeling options.

5.6 Results

5.6.1 GWAS Data

We first apply the 3 modeling options for fitting linear mixed effects model to the GAW20 data. For each SNP, as in Irvin et al. [58], we test for statistical significant departure from the null hypothesis that the SNP has no effect on the beta scores. The Manhattan plots from these 3 modeling options are shown in Figure 5.3 and Figure 5.4. As expected, when the effective sample size is small, as with the representative option, we have less power in detecting significant SNPs. So the overall p values from option 1 are lower than those from options 2 and 3. Figure 5.4 plots the results from options 2 and 3 together, because of their similarity. While the chromatic dots show the results using the original Irvin et al. study [58], the black dots are the significant findings from option 3. The results from these 2 options are very similar. In fact, the top 100 significant SNPs identified using options 2 and 3 have 71 SNPs in common. Using option 1, cg12033043 from chromosome 8 is found to be the most significant SNP. Using option 2, cg27026926 from chromosome 8, cg05599320 from chromosome 1, cg13982695 from chromosome 11, cg27331738 from chromosome 8, and cg00223867 from chromosome 8 are identified to be the top 5 significant SNPs, which are among the top 100 SNPs identified by option 3.

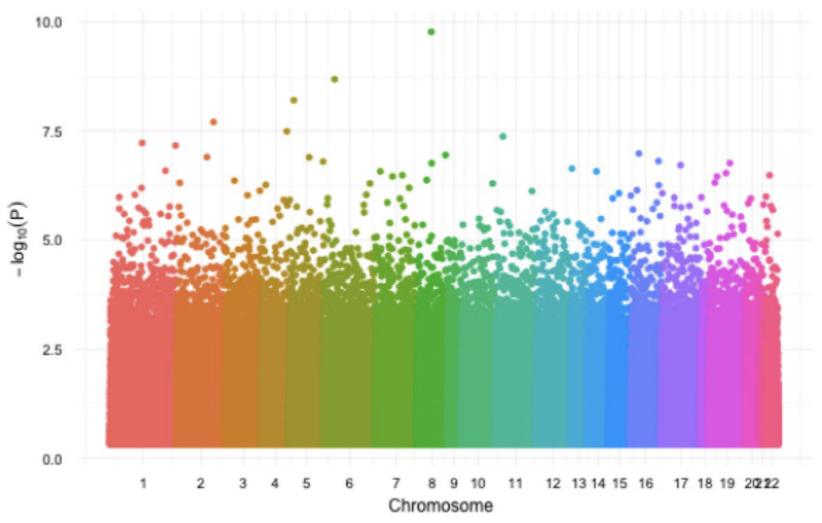


Figure 5.3: Manhattan plots for option 1.

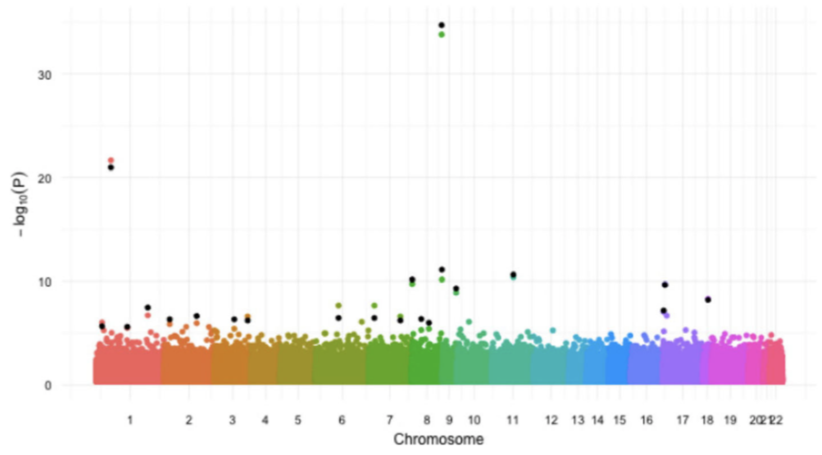


Figure 5.4: Manhattan plots for option 2 and option 3 (black dots).

This result does not completely meet our expectation. Both option 1 and option 2 are fitting statistical valid models to the data, while option 3 (independence) is based on an incorrect assumption. However, even though option 3 is under an incorrect assumption, it has a substantial overlap with option 2.

To understand what we observed in the real data analysis, we carry out a simple simulation study to further compare the three coping options. We ran independently 3 sets of simulations, in which we draw samples under the model assumed in eq. (1) to eq. (3), with random effects from multivariate normal distribution and known variance–covariance structure. The results showed that, while option 2 has the highest discovery rate, the significant SNP's found by option 3 is still closer to the that found in option 2 than option 1 is.

This suggests a potential hybrid modeling option that combines results from the independence option and the representative option when the kinship coefficients are not available.

5.7 Discussion

In this work, we evaluate different modeling options for coping with data with family structure in the context of genetics studies. Our analysis suggests the need for adjusting for kinships. When kinship information is not known, we compare two opposite strategies, one that treats all individuals in the study as independent and the other that approximately uses the family centroid by randomly sampling 1 representative from each family. Our results suggest that the cost of ignoring other members from a family (the representative option) is greater than that of ignoring dependence among all individuals in a study (the independence option). More research should be conducted to understand this phenomenon. From the results of a simple simulation, we suggest that both strategies should be used in practice and that the focus should be on SNPs that are identified by both.

From representation learning point of view, the surprising result may due to that (1) the kinship matrix isn't sufficient for modeling the family structure, or (2) there may be more hidden confounders that need to be adjusted for. All the hypotheses require more structured data for an entailed further investigation.

Chapter 6: I-Score Representation Learning in Causal Inference

6.1 Introduction

Philosophically, there have been many debates about the right framework and the proper definition of what causality is, and even debates about the existence of causality. Here we would not get involved into these debates; instead, we would like to provide insights from a representation perspective – we human understand the outside world through our own representations of the world; therefore, whether there really exists causality or not is not an issue for representations in human minds. All we care about is to use our own representation of the world to fulfill our tasks. As long as the tasks can be achieved, whatever the mental representations are isn't what we are interested in pursuing. Furthermore, different people may even have different mental representations.

To translate the above philosophical perspective into statistical language, we can say that, in different scenarios (tasks are different), people may use different representations of the data, and there is no "right" or "wrong" representation. As long as the representation can fulfill the task, it is a reasonable one.

In statistical causal inference literature, the most common framework is the Neyman–Rubin causal model. We will follow their setting and introduce how our proposed representation learning can be fit into and even improve the current methods.

6.2 Neyman–Rubin Causal Model

The Neyman–Rubin causal model is an approach to the statistical analysis of cause and effect based on the framework of potential outcomes – the name "Rubin causal model" was first coined by Paul W. Holland [61], and the potential outcomes framework was first proposed by Jerzy Neyman [62], who first discussed the problem only in the setting of completely randomized experiments. Later, Rubin extended it into a general framework for thinking about causation in both observational and experimental studies [63]. Because of its mathematical clarity and practicality, this framework has become more and more popular in many fields, including Statistics [61] [64], Medicine [65], Economics [66] [67] [68], Political Science [69] [70], Sociology [71] [72] [73] and even Law [74].

Basically, this model is based on the idea of potential outcomes. For example, a person got a flu and took an over-counter drug. Three days later, her symptoms became less and she felt energized again. Is it because of the drug that cured her flu, or is it because of her own immune system and there's nothing to do with the drug? To measure the causal effect of the drug, we need to compare the outcome for the same individual in both alternative futures. Since it is impossible to see both potential outcomes in reality, one of the potential outcomes is always missing. This dilemma is the "fundamental problem of causal inference".

Because of the fundamental problem of causal inference, unit-level causal effects cannot be directly observed. However, randomized experiments allow for the estimation of population-level causal effects [75] – this is what we normally use Statistical inference for, where independence between observations is required in the assumption. A randomized experiment can create an artificial environment that can satisfy this independence assumption by assigning people randomly to treatments: drug or no drug. Because of this random assignment, the groups are (on average) equivalent, and the mean difference in the outcomes (discovery level from the flu) can be attributed to the drug effect. An estimate of the average treatment effect (ATE) can then be obtained by computing the difference in means between the treated and the control observations.

In many circumstances, however, randomized experiments are not possible due to ethical or practical concerns. In such scenarios there is a non-random assignment mechanism. This is especially the case for clinical trials – not everyone is willing to or susceptible to take the same drug, and let alone the same dosage. Because of this practical challenge, many statistical methods have been developed for causal inference, such as propensity score matching. These methods attempt to correct for the assignment mechanism by finding control units similar to treatment units.

Another popular branch of methods is sample stratification. These methods attempt to find stratifications in data such that the outcome and the treatment assignment are independent within each stratum. If the stratifications are created based on some variables, we call these variables the confounders. Mathematically, we are interested in finding confounders C such that the outcome Y and the treatment assignment Z are conditionally independent: $Y \perp\!\!\!\perp Z|C$.

In proceeding with the above methodology, one precaution is to avoid Simpson's paradox, which is only apparent in the sense that the successes of groups seem reversed when the groups are combined [76]. This result is often encountered in medical science statistics, and occurs when frequency data are hastily given causal interpretation, the paradox disappears when causal relations are derived systematically, through formal analysis. This paradox is chiefly an issue of aggregated statistical analysis where separate in real life groups are analyzed together [77] [78]. The variable that should have been used to separate these groups are the confounders [76] [78].

6.3 Representation Learning in Causal Inference

As discussed in the previous section, the key probabilistic idea upon which statistical causal inference relies is conditional probability. But when we are making causal inferences, conditional probabilities are not themselves of direct interest. We use conditional probabilities to learn about counterfactuals of interest. This is when our proposed representation learning comes in handy. In other words, we would like to find representations $f(X)$, where X stands for some observable features/covariates and $f(X)$ is a proper functional transformation of X, such that the outcome Y and the treatment assignment Z are independent: $Y \perp\!\!\!\perp Z|f(X)$. These representations $f(X)$ also serve the role as confounders.

6.4 Definition and Proposition

(a) Definition of a “Strongly Relevant Variable":

Given a set of covariates $\mathbf{X} = \{X_1, X_2, ..., X_m\}$, a covariate $X_k \in \mathbf{X}$ is defined as strongly relevant in terms of predicting the response variable Y if

$$\begin{aligned} &\mathbb{P}(Y|\mathbf{X}) \neq \mathbb{P}(Y|\mathbf{X}_{-k}) \\ &\text{where } \mathbf{X}_{-k} = \mathbf{X} \setminus \{X_k\}, \\ &\text{and we will use this notation throughout this chapter.} \end{aligned} \tag{6.1}$$

(b) Definition of a “Weakly Relevant Variable":

Given a set of covariates $\mathbf{X} = \{X_1, X_2, ..., X_m\}$, a covariate $X_k \in \mathbf{X}$ is defined as weakly relevant in terms of predicting the response variable Y if

$$\begin{aligned} &\mathbb{P}(Y|\mathbf{X}) = \mathbb{P}(Y|\mathbf{X}_{-k}) \\ &\text{where } \mathbf{X}_{-k} = \mathbf{X} \setminus \{X_k\}, \end{aligned} \tag{6.2}$$

(c) Definition of a “Irrelevant Variable":

Given a set of covariates $\mathbf{X} = \{X_1, X_2, ..., X_m\}$, a covariate $X_k \in \mathbf{X}$ is defined as strongly relevant in terms of predicting the response variable Y if

$$\begin{aligned} &\mathbb{P}(Y|\mathbf{X}) = \mathbb{P}(Y|\mathbf{X}_{-k}) \\ &\text{where } \mathbf{X}_{-k} = \mathbf{X} \setminus \{X_k\}, \end{aligned} \tag{6.3}$$

(d) Definition of an “Unassociated Variable":

Given a set of covariates $\mathbf{X} = \{X_1, X_2, ..., X_m\}$, a covariate $X_k \in \mathbf{X}$ is defined as marginally unassociated to the response variable Y if

$$\mathbb{E}(Y|\mathbf{X}) = \mathbb{E}(Y|\mathbf{X}_{-k}) \tag{6.4}$$

(e) Definition of a “Noisy Variable":

Given a set of covariates $\mathbf{X} = \{X_1, X_2, ..., X_m\}$, a covariate $X_k \in \mathbf{X}$ is defined as noisy to the response variable Y if it is unassociated and is independent with all the other covariates $\mathbf{X_{-k}}$.

$$\mathbb{E}(Y|\mathbf{X}) = \mathbb{E}(Y|\mathbf{X}_{-k}) \text{ and } X_k \perp\!\!\!\perp \mathbf{X_{-k}} \tag{6.5}$$

(f) Definition of an “Association Set":

Given a set of covariates $\mathbf{X}$ and the response variable Y, after eliminating all the marginally noisy variables, we call the remaining set an association set of $\mathbf{X}$ to Y and will be denoted by $\mathbf{X_Y}$.

Proposition: Given an association set $\mathbf{X_Y} = \{X_1, X_2, ..., X_j\}$, it has the property that

$$\mathbb{E}(Y|\mathbf{X_Y}) \neq \mathbb{E}(Y|\mathbf{X_{Y}}_{-i}) \; \forall \, i \in \{1, 2, ...j\} \tag{6.6}$$

(g) Definition of the “Minimum Association Set":

Given a finite set of covariates $\mathbf{X} = \{X_1, X_2, ..., X_m\}$ and the response variable Y, the minimum association set out of $\mathbf{X}$ will be denoted by $\mathbf{X_{Y}}_{\min}$ and is defined as follows:

$\mathbf{X_Y} \supseteq \mathbf{X_Y}_{\min}$ has the property that

$$\begin{aligned}&\text{For all elements } X_i \text{ in } \mathbf{X_Y}_{\min},\\ &\nexists \mathbf{X_I} \supseteq X_i, \text{ s.t. } \mathbb{E}(Y|\mathbf{X_Y}_{\min}) = \mathbb{E}(Y|\mathbf{X_Y}_{\min} \setminus \mathbf{X_I}); i.e.\\ &\forall \mathbf{X_I} \supseteq X_i, \ \mathbb{E}(Y|\mathbf{X_Y}_{\min}) \neq \mathbb{E}(Y|\mathbf{X_Y}_{\min} \setminus \mathbf{X_I})\end{aligned} \tag{6.7}$$

On the other hand, $\mathbf{X_Y}^c_{\min} = \{X_1, X_2, ..., X_j\}$ has the property that

$$\begin{aligned}&\text{For all elements } X_i \text{ in } \mathbf{X_Y}^c_{\min},\\ &\exists \mathbf{X_I} \supseteq X_i, \ \mathbb{E}(Y|\mathbf{X_Y}^c_{\min}) = \mathbb{E}(Y|\mathbf{X_Y}^c_{\min} \setminus \mathbf{X_I})\end{aligned} \tag{6.8}$$

(h) Proposition:

For any p positive integers $x_1, x_2, ..., x_p$ that have a fixed sum N; that is, $\sum_{i=1}^N x_i = N$, $(\sum_{i=1}^N x_i)^2 \geq \sum_{i=1}^N x_i^2$.

(i) Theorem: Assume that $\mathbf{X}_{-m}$ and X_m both have discrete distributions.

If X_m is noisy for Y, then

$I(Y, \mathbf{X_Y}) \leq I(Y, \mathbf{X_{Y-m}})$, or, equivalently,

$$\sum_{\mathbf{X_Y}} (\mathbb{E}(Y|\mathbf{X_Y}) - \mathbb{E}(Y))^2 \cdot (\mathbb{P}(\mathbf{X_Y} = x)))^2 \leq \sum_{\mathbf{X}_{Y-m}} (\mathbb{E}(Y|\mathbf{X}_{Y_{-m}}) - \mathbb{E}(Y))^2 \cdot (\mathbb{P}(\mathbf{X_{Y_{-m}}} = x_{-m}))^2. \tag{6.9}$$

- Proof:

  Since X_m is noisy for Y, for all x_{-m}, $\mathbb{E}(Y|\mathbf{X_{Y_{-m}}} = x_{-m}) = \mathbb{E}(Y|\mathbf{X_Y} = x)$ for some x. Conditioned on each x_{-m}, $\mathbb{P}(\mathbf{X_{Y_{-m}}} = x_{-m}) = \sum_x \mathbb{P}(\mathbf{X_Y} = x)$ for the same set of x.

By proposition (h), conditioned on each x_{-m}, we have

$$\mathbb{P}^2(\mathbf{X}_{\mathbf{Y_{-m}}} = x_{-m}) \geq \sum_x \mathbb{P}^2(\mathbf{X}_{\mathbf{Y}} = x) \text{ for the same set of } x. \tag{6.10}$$

Therefore, by summing over x_{-m} with the corresponding x in Inequality 6.10, we get Inequality 6.9.

(j) I-score serves as the statistics to test against the null hypothesis that the given set of covariates $\mathbf{X}_{\text{data}}$ is all noisy.

(k) The I-score backward dropping algorithm can be used to search for the minimum association set out of the given set of covariates $\mathbf{X}_{\text{data}}$.

6.5 Causal Inference

(a) Notation: Let Y_{1i} denote the potential outcome for unit i if the unit receives treatment, and let Y_{0i} denote the potential outcome for unit i in the control regime. The treatment effect for observation i is defined by $\mu_{0i} = Y_{1i} - Y_{0i}$.

(b) Model:

$$\begin{aligned} &Y_{1i} = Y_{0i} + \mu_{0i} = Y_{0i} + (\mu_0 + \epsilon_i)\ \forall i, \\ &\text{where } \epsilon_i \text{ has mean 0 and is independent among all individual.} \end{aligned} \tag{6.11}$$

(c) Fundamental problem of causal inference: in an observational study, Y_{1i} and Y_{0i} can never both be observed for any i.

(d) Statistically, we can make use of the population distribution trying to solve this problem.

(e) If our goal is to estimate the average treatment effect μ_0, and if we assume

$$\begin{aligned}&\mathbb{E}(Y_0|Z=0,\mathbf{X}) = \mathbb{E}(Y_0|\mathbf{X}), \text{ and}\\ &\mathbb{E}(Y_1|Z=1,\mathbf{X}) = \mathbb{E}(Y_1|\mathbf{X})\end{aligned} \tag{6.12}$$

(which is slightly weaker than the conditional independence assumption) then

$$\begin{aligned}\mu_0 &= \mathbb{E}(Y_1) - \mathbb{E}(Y_0)\\ &= \mathbb{E}(Y_1 - Y_0)\\ &= \mathbb{E}(\mathbb{E}(Y_1 - Y_0)|\mathbf{X})\\ &= \mathbb{E}(\mathbb{E}(Y_1|\mathbf{X}) - \mathbb{E}(Y_0|\mathbf{X}))\\ &= \mathbb{E}(\mathbb{E}(Y_1|Z=1,\mathbf{X}) - \mathbb{E}(Y_0|Z=0,\mathbf{X}))\end{aligned} \tag{6.13}$$

Since μ_0 is a constant that doesn't depend on $\mathbf{X}$, we have

$$\mu_0 = \mathbb{E}(Y_1|Z=1,\mathbf{X}) - \mathbb{E}(Y_0|Z=0,\mathbf{X}) \tag{6.14}$$

For any proper subset $\mathbf{X_c} \subset \mathbf{X}$ such that the above equation still holds, i.e.

$$\begin{aligned}\mu_0 &= \mathbb{E}(Y_1|\mathbf{X}) - \mathbb{E}(Y_0|\mathbf{X})\\ &= \mathbb{E}(Y_1|Z=1,\mathbf{X}) - \mathbb{E}(Y_0|Z=0,\mathbf{X})\\ &= \mathbb{E}(Y_1|Z=1,\mathbf{X_c}) - \mathbb{E}(Y_0|Z=0,\mathbf{X_c}),\end{aligned} \tag{6.15}$$

we will call this $\mathbf{X_c}$ one proper adjustment set (for estimating the ATE), and the smallest among all of these sets is called the minimum adjustment set (for estimating the ATE).

Hence, $\mathbf{X_c} = \{X_1, X_2, ..., X_j\}$ has the property that

$$\begin{aligned} &\mathbb{E}(Y_0|Z = 0, \mathbf{X_c}) \neq \mathbb{E}(Y_0|Z = 0, \mathbf{X_{c-\mathit{i}}})\ \forall\ i \in \{1, 2, ...j\},\ \text{or} \\ &\mathbb{E}(Y_1|Z = 1, \mathbf{X_c}) \neq \mathbb{E}(Y_1|Z = 1, \mathbf{X_{c-\mathit{i}}})\ \forall\ i \in \{1, 2, ...j\} \end{aligned} \tag{6.16}$$

6.5.1 On the Search of Adjustment Sets

Relationship between the Adjustment Set and the Association Set

- In this section, we will show, under the causal inference model, what is the relationship between the association set $\mathbf{X_Y}$ and the adjustment set $\mathbf{X_c}$, and how to search for $\mathbf{X_c}$.

  Note: $\mathbf{X_Y}$ and $\mathbf{X_c}$ may not be their minimum.
- To search for $\mathbf{X_c}$, we define the following $Y'(\delta)$ and its corresponding $\mathbf{X_{Y'(\delta)}}$:
- Define $Y'(\delta) = Y_0 \cdot (1 - Z) + (Y_1 + \delta) \cdot Z = Y_0 \cdot (1 - Z) + (Y_0 + \mu_0 + \delta) \cdot Z$
- When $\delta = -\mu_0$, $Y'(\delta)$ is equivalent to Y_0.

Theories

- **Theorem I**

  $\mathbf{X_{Y_0}} = \mathbf{X_{Y_1}}$, and we will define
  $\mathbf{X_Y} = \mathbf{X_{Y_0}} = \mathbf{X_{Y_1}}$.
- **Theorem II**

  $\mathbf{X} \supseteq \mathbf{X_{Y_0}} \supseteq \mathbf{X_{Y_0}} \cap \mathbf{X_Z} \supseteq \mathbf{X_c}$.
  $\mathbf{X} \supseteq \mathbf{X_{Y_0}} \supseteq \mathbf{X_c}$.
  $\mathbf{X} \supseteq \mathbf{X_{Y_0}} = \mathbf{X_{Y_1}} \supseteq \mathbf{X_c}$.

- **Theorem III**

  $\mathbf{X_Y}^c \cap \mathbf{X_Z} \subset \mathbf{X_{Y'(\delta)}} \cap \mathbf{X_Z}\ \forall \delta \in \mathbb{R} \setminus -\mu_0.$

- **Theorem IV**

  For any $x_j \in \mathbf{X_Y} \cap \mathbf{X_Z}$, $\exists$ corresponding ϵ_j s.t. $x_j \in \mathbf{X_{Y'(\delta)}} \cap \mathbf{X_Z}\ \forall\ \delta \in (-\mu_0 - \epsilon_j, -\mu_0 + \epsilon_j)$. It is also true when $\epsilon_j = 0$.

- **Theorem V**

  $\mathbf{X_{Y'(\delta)}}^c \cap \mathbf{X_Z} \subset \mathbf{X_Y} \cap \mathbf{X_Z}\ \forall \delta \in \mathbb{R} \setminus -\mu_0.$

- **Theorem VI**

  $\mathbf{X_Y} \cap \mathbf{X_Z}^c = \mathbf{X_{Y'(\delta)}} \cap \mathbf{X_Z}^c\ \forall \delta \in \mathbb{R} \setminus -\mu_0.$

Proof of the Theorems

- **Proof of Theorem I**

  By definition, $\mathbf{X_{Y_0}} = \{X_1, X_2, ..., X_j\}$ has the property that

  $\mathbb{E}(Y_0 | \mathbf{X_{Y_0}}) \neq \mathbb{E}(Y_0 | \mathbf{X_{Y_0}}_{-i})\ \forall\ i \in \{1, 2, \dots j\}$

$$
\begin{aligned}
\Rightarrow \mathbb{E}(Y_1 | \mathbf{X_{Y_0}}) &= \mathbb{E}(Y_0 + \mu + \epsilon | \mathbf{X_{Y_0}}) = \mathbb{E}(Y_0 | \mathbf{X_{Y_0}}) + \mu, \\
&\neq \mathbb{E}(Y_0 | \mathbf{X_{Y_0}}_{-i}) + \mu\ \forall\ i \in \{1, 2, \dots j\} \\
&= \mathbb{E}(Y_1 | \mathbf{X_{Y_0}}_{-i})\ \forall\ i \in \{1, 2, \dots j\}
\end{aligned} \tag{6.17}
$$

Therefore, $\mathbf{X_{Y_0}} \subset \mathbf{X_{Y_1}}$. Similarly, we can prove that $\mathbf{X_{Y_1}} \subset \mathbf{X_{Y_0}}$. Hence we have

$$\mathbf{X_{Y_0}} = \mathbf{X_{Y_1}}. \tag{6.18}$$

We will denote $\mathbf{X_Y} = \mathbf{X_{Y_0}} = \mathbf{X_{Y_1}}$.

- **Proof of Theorem II**

$$\begin{aligned}
&\mathbb{P}(Z = 1|Y_0, \mathbf{X_{Y0}}) = \mathbb{E}(Z|Y_0, \mathbf{X_{Y0}}) = \mathbb{E}[\mathbb{E}(Z|Y_0, \mathbf{X_{Y0}})|Y_0, \mathbf{X_{Y0}}] \\
&\mathbb{E}(Y_0|Z = 0, \mathbf{X_c}) = \mathbb{E}(Y_0|\mathbf{X_c}) \neq \mathbb{E}(Y_0|\mathbf{X_{c-\mathit{i}}}). \ \forall\, i \in \{1, 2, ...j\}, \text{ and} \\
&\mathbb{E}(Y_1|Z = 1, \mathbf{X}) = \mathbb{E}(Y_1|Z = 1, \mathbf{X_c}) = \mathbb{E}(Y_1|\mathbf{X_c}) \neq \mathbb{E}(Y_1|\mathbf{X_{c-\mathit{i}}}). \ \forall\, i \in \{1, 2, ...j\}
\end{aligned} \tag{6.19}$$

For any element $x_j \in \mathbf{X_c} = \{X_1, X_2, ..., X_j\}$, we have the property by (the conditional independence) assumption

$$\begin{aligned}
&\mathbb{E}(Y_0|Z = 0, \mathbf{X}) = \mathbb{E}(Y_0|Z = 0, \mathbf{X_c}) = \mathbb{E}(Y_0|\mathbf{X_c}) \neq \mathbb{E}(Y_0|\mathbf{X_{c-\mathit{i}}}). \ \forall\, i \in \{1, 2, ...j\}, \text{ and} \\
&\mathbb{E}(Y_1|Z = 1, \mathbf{X}) = \mathbb{E}(Y_1|Z = 1, \mathbf{X_c}) = \mathbb{E}(Y_1|\mathbf{X_c}) \neq \mathbb{E}(Y_1|\mathbf{X_{c-\mathit{i}}}). \ \forall\, i \in \{1, 2, ...j\}
\end{aligned} \tag{6.20}$$

Similarly,

$$\mathbb{E}(Y_1|Z = 1, \mathbf{X_{c-\mathit{j}}}, X_j) = \mathbb{E}(Y_1|\mathbf{X_{c-\mathit{j}}}, X_j) \neq \mathbb{E}(Y_1|\mathbf{X_{c-\mathit{j}}}). \tag{6.21}$$

As a result, $\mathbf{X_c}$ is the association set in terms of the predictivity of $\mathbf{X_c}$ to both Y_0 and Y_1.

- **Proof of Theorem III**

First, we consider the set X_Z. For any element $x_j \in X_Z$, we have the property

$$\begin{aligned} & \mathbb{E}(Z|x_j = x_i) \neq \mathbb{E}(Z) \text{ for at least one } \mathbf{x_i} \in \{\mathbf{x_1}, \mathbf{x_2}, ..., \mathbf{x_{\Pi_{x_j}}}\} \\ \iff & \mathbb{P}(Z = 1|x_j = x_i) \neq \mathbb{P}(Z = 1) \text{ for at least one } \mathbf{x_i} \in \{\mathbf{x_1}, \mathbf{x_2}, ..., \mathbf{x_{\Pi_{x_j}}}\} \end{aligned} \tag{6.22}$$

and if also $x_j \in X_0^c$

$$\mathbb{E}(Y_0|x_j = x_i) = \mathbb{E}(Y_0) \ \forall \ \mathbf{x_i} \in \{\mathbf{x_1}, \mathbf{x_2}, ..., \mathbf{x_{\Pi_{x_j}}}\}. \tag{6.23}$$

Then,

$$\begin{aligned} \mathbb{E}(Y_0'|x_j = x_i) &= \mathbb{E}(Y_0|x_j = x_i) + (\mu_0 + \delta) \cdot \mathbb{P}(Z = 1|x_j = x_i) \ \forall \ \mathbf{x_i} \in \{\mathbf{x_1}, \mathbf{x_2}, ..., \mathbf{x_{\Pi_{x_j}}}\} \\ &= \mathbb{E}(Y_0) + (\mu_0 + \delta) \cdot \mathbb{P}(Z = 1|x_j = x_i) \ \forall \ \mathbf{x_i} \in \{\mathbf{x_1}, \mathbf{x_2}, ..., \mathbf{x_{\Pi_{x_j}}}\} \\ &\neq \mathbb{E}(Y_0) + (\mu_0 + \delta) \cdot \mathbb{P}(Z = 1) \text{ for at least one } \mathbf{x_i} \in \{\mathbf{x_1}, \mathbf{x_2}, ..., \mathbf{x_{\Pi_{x_j}}}\} \\ &= \mathbb{E}(Y_0') \end{aligned} \tag{6.24}$$

$$\implies \mathbb{E}(Y_0'|x_j = x_i) \neq \mathbb{E}(Y_0') \text{ for at least one } \mathbf{x_i} \in \{\mathbf{x_1}, \mathbf{x_2}, ..., \mathbf{x_{\Pi_{x_j}}}\}.$$

For all δ, except for $\delta = -\mu_0$, $\mathbb{E}(Y_0'|x_j = x_i) \neq \mathbb{E}(Y_0')$.

$\implies x_j$ is associated with Y_0'.

$\implies x_j \in X_0'$.

Therefore, $X_0^c \cap X_Z \subset X_0' \cap X_Z$, for all δ, except for $\delta = -\mu_0$.

- **Proof of Theorem IV**

First, we consider the set X_Z. For any element $x_j \in X_Z$, we have the property

$$\begin{aligned} &\exists \text{ at least one } \mathbf{x_i} \in \{\mathbf{x_1}, \mathbf{x_2}, ..., \mathbf{x_{\Pi_{x_j}}}\} \text{ s.t. } \mathbb{E}(Z|x_j = x_i) \neq \mathbb{E}(Z) \\ \iff &\exists \text{ at least one } \mathbf{x_i} \in \{\mathbf{x_1}, \mathbf{x_2}, ..., \mathbf{x_{\Pi_{x_j}}}\} \text{ s.t. } \mathbb{P}(Z = 1|x_j = x_i) \neq \mathbb{P}(Z = 1) \end{aligned} \tag{6.25}$$

Also, since $x_j \in X_0$,

$$\mathbb{E}(Y_0|x_j = x_i) \neq \mathbb{E}(Y_0) \text{ for at least one } \mathbf{x_i} \in \{\mathbf{x_1}, \mathbf{x_2}, ..., \mathbf{x_{\Pi_{x_j}}}\}. \tag{6.26}$$

Then,

$$\begin{aligned} &\mathbb{E}(Y_0'|x_j = x_i) - \mathbb{E}(Y_0') \\ &= [\mathbb{E}(Y_0|x_j = x_i) + (\mu_0 + \delta) \cdot \mathbb{P}(Z = 1|x_j = x_i)] - [\mathbb{E}(Y_0) + (\mu_0 + \delta) \cdot \mathbb{P}(Z = 1)] \\ &= [\mathbb{E}(Y_0|x_j = x_i) - \mathbb{E}(Y_0)] + (\mu_0 + \delta) \cdot [\mathbb{P}(Z = 1|x_j = x_i) - \mathbb{P}(Z = 1)] \\ &\neq 0 \text{ for many } \delta \end{aligned} \tag{6.27}$$

$$\implies \mathbb{E}(Y_0'|x_j = x_i) \neq \mathbb{E}(Y_0') \text{ for many } \delta.$$

Especially, when $\delta = -\mu_o + \epsilon$, the above equation becomes

$$\begin{aligned} &\mathbb{E}(Y_0'|x_j = x_i) - \mathbb{E}(Y_0') \\ &= [\mathbb{E}(Y_0|x_j = x_i) + (\mu_0 - \mu_o + \epsilon) \cdot \mathbb{P}(Z = 1|x_j = x_i)] - [\mathbb{E}(Y_0) + (\mu_0 - \mu_o + \epsilon) \cdot \mathbb{P}(Z = 1)] \\ &= [\mathbb{E}(Y_0|x_j = x_i) - \mathbb{E}(Y_0)] + \epsilon \cdot [\mathbb{P}(Z = 1|x_j = x_i) - \mathbb{P}(Z = 1)] \\ &= f(\epsilon) \end{aligned} \tag{6.28}$$

which is a continuous function in ϵ, since both $\mathbb{E}(Y_0|x_j = x_i) - \mathbb{E}(Y_0)$ and $\mathbb{P}(Z = 1|x_j = x_i) - \mathbb{P}(Z = 1)$ are fixed constants.

Therefore,

$$\lim_{\epsilon \to 0} f(\epsilon) = f(0) = \mathbb{E}(Y_0'(\delta = -\mu_0)|x_j = x_i) - \mathbb{E}(Y_0'(\delta = -\mu_0)) = \mathbb{E}(Y_0|x_j = x_i) - \mathbb{E}(Y_0) \neq 0$$

Similarly, when $\delta = -\mu_o - \epsilon$, we can prove the same result.

Hence, we have, for any $x_j \in \mathbf{X_0} \cap \mathbf{X_Z}$, $\exists \epsilon$ s.t. $x_j \in \mathbf{X_0}'(\delta \in (-\mu_0 - \epsilon, -\mu_0 + \epsilon)) \cap \mathbf{X_Z}$. It is also true when $\epsilon = 0$, for any $x_j \in \mathbf{X_0} \cap \mathbf{X_Z}$, $x_j \in \mathbf{X_0}'(\delta = -\mu_0) \cap \mathbf{X_Z}$.

- **Proof of Theorem V**

First, we consider the set X_Z^c. For any element $x_j \in X_Z^c$, we have the property

$$\begin{aligned} &\mathbb{E}(Z|x_j = x_i) = \mathbb{E}(Z)\ \forall \mathbf{x_i} \in \{\mathbf{x_1}, \mathbf{x_2}, ..., \mathbf{x_{\Pi_{x_j}}}\} \\ \Longleftrightarrow\ &\mathbb{P}(Z = 1|x_j = x_i) = \mathbb{P}(Z = 1)\ \forall \mathbf{x_i} \in \{\mathbf{x_1}, \mathbf{x_2}, ..., \mathbf{x_{\Pi_{x_j}}}\} \end{aligned} \tag{6.29}$$

and if also $x_j \in X_0$

$$\mathbb{E}(Y_0|x_j = x_i) \neq \mathbb{E}(Y_0) \text{ for at least one } \mathbf{x_i} \in \{\mathbf{x_1}, \mathbf{x_2}, ..., \mathbf{x_{\Pi_{x_j}}}\}. \tag{6.30}$$

Then,

$$
\begin{aligned}
\mathbb{E}(Y_0'|x_j = x_i) &= \mathbb{E}(Y_0|x_j = x_i) + (\mu_0 + \delta) \cdot \mathbb{P}(Z = 1|x_j = x_i) \;\forall\; \mathbf{x_i} \in \{\mathbf{x_1}, \mathbf{x_2}, ..., \mathbf{x_{\Pi_{x_j}}}\} \\
&= \mathbb{E}(Y_0|x_j = x_i)) + (\mu_0 + \delta) \cdot \mathbb{P}(Z = 1) \;\forall\; \mathbf{x_i} \in \{\mathbf{x_1}, \mathbf{x_2}, ..., \mathbf{x_{\Pi_{x_j}}}\} \\
&\neq \mathbb{E}(Y_0) + (\mu_0 + \delta) \cdot \mathbb{P}(Z = 1) \text{ for at least one } \mathbf{x_i} \in \{\mathbf{x_1}, \mathbf{x_2}, ..., \mathbf{x_{\Pi_{x_j}}}\} \\
&= \mathbb{E}(Y_0) + \mu_0 \cdot \mathbb{P}(Z = 1) + \delta \cdot \mathbb{P}(Z = 1) \\
&= \mathbb{E}(Y_0') + \delta \cdot \mathbb{P}(Z = 1)
\end{aligned}
\tag{6.31}
$$

When $\delta = 0$, $\mathbb{E}(Y_0'|x_j = x_i) \neq \mathbb{E}(Y_0')$.

$\implies$ x_j is associated with Y_0'.

$\implies$ $x_j \in X_0'$.

$\implies$ $X_0 \cap X_Z^c \subset X_0' \cap X_Z^c$.

Similarly, we can prove that $X_0' \cap X_Z^c \subset X_0 \cap X_Z^c$, which then implies $X_0 \cap X_Z^c = X_0' \cap X_Z^c$.

6.5.2 Inference

The estimation problem is to use the observed data to estimate the true average treatment effect, i.e. $\mu = \mathbb{E}(Y_1) - \mathbb{E}(Y_0)$. However, we only have $Y_1|Z = 1$ and $Y_0|Z = 0$ available in the data. In general, $\mathbb{E}(Y_1) - \mathbb{E}(Y_0) \neq \mathbb{E}(Y_1|Z = 1) - \mathbb{E}(Y_0|Z = 0)$. The equality only holds when there exists no confounders, which is usually not practical. Nevertheless, under the conditional independence assumption,

$$
\mathbb{E}(Y_1) - \mathbb{E}(Y_0) = \mathbb{E}(Y_1|Z = 1, \mathbf{X} = \mathbf{x_i}) - \mathbb{E}(Y_0|Z = 0, \mathbf{X} = \mathbf{x_i}) \;\forall \mathbf{x_i} \in \{\mathbf{x_1}, \mathbf{x_2}, ..., \mathbf{x_{\pi_X}}\} \tag{6.32}
$$

However, this is usually infeasible in practice because of the sparsity generated by the partitions of **X**. Our interest is in searching for a minimum subset $\mathbf{X_c} \subset \mathbf{X}$, such that

$$\begin{aligned} \mathbb{E}(Y_1) - \mathbb{E}(Y_0) &= \mathbb{E}(Y_1|Z = 1, X = x) - \mathbb{E}(Y_0|Z = 0, X = x) \ \forall x \in X \\ &= \mathbb{E}(Y_1|Z = 1, X = x_c) - \mathbb{E}(Y_0|Z = 0, X = x_c) \ \forall x_c \in X_c, \end{aligned} \tag{6.33}$$

where X_c is minimum in the sense that getting rid of any elements in the set X_c would result in inequality in Equation 6.24.

6.5.3 Simulation

In this section, we will use several examples to show the idea that, when the sweeping variable δ equals to the true causal effect, the intersection of the variables influential to the outcome and the variables influential to the treatment assignment will be the true confounders; in other cases when δ does not equal to the true causal effect, at least one of the true confounders will be included in the intersection of the variables influential to the outcome and the variables influential to the treatment assignment will be the true confounders.

Case 1:

$$
\begin{aligned}
&X_1, X_2, ...X_{20} \text{ iid Bernoulli}(0.5) \\
&\quad \mathbb{P}(Z = 1) = \text{logit}(X_1 + X_2 + X_3 + X_4 + X_5) \\
&\qquad\quad Y_0 = X_3 + X_4 + X_5 + X_6 + X_7 + N(0, 1) \\
&\qquad\quad \mu_0 = 3 + N(0, 1) \\
&\qquad\quad Y = Y_0 \cdot (1 - Z) + (Y_0 + \mu_0) \cdot Z = Y_0 \cdot (1 - Z) + Y_1 \cdot Z \\
&\qquad\quad Y' = Y_0 \cdot (1 - Z) + (Y_0 + \mu_0 + \delta) \cdot Z = Y_0 \cdot (1 - Z) + (Y_1 + \delta) \cdot Z \\
&\text{sample size = 20000, case = 10000, control = 10000}
\end{aligned}
\tag{6.34}
$$

δ	$X_0'(\delta) \cap X_Z(\text{ideal})$
-20	1,2,4,5
-19	1,2,4,5
-18	1,2,3,4,5
-17	1,2,3,4,5
-16	1,2,3,4,5
-15	1,2,3,4,5
-14	1,2,3,4,5
-13	1,2,4,5
-12	1,2,4,5
-11	1,2,5
-10	1,2
-9	1,2,5
-8	1,2
-7	1,2,3
-6	1,2
-5	3,4,5
-4	3,4,5
-3	2,3,4,5
-2	3,4,5
-1	3,4,5
0	1,3,4,5
+1	3,4,5
+2	1,2,3,4,5
+3	1,2,3,4,5
+4	1,2,3,4,5
+5	1,2,3,4,5
+6	1,2,3,4,5
+7	1,2,3,4,5
+8	1,2,3,4,5
+9	1,2,3,4,5
+10	1,2,3,4,5
+11	1,2,3,4,5
+12	1,2,3,4,5
+13	2,3,4,5
+14	1,2,3,4,5
+15	1,2,3,4,5
+16	1,2,3,4,5
+17	1,2,3,4,5
+18	1,2,3,4,5
+19	1,2,3,4,5
+20	1,2,3,4,5

Case 2:

$X_1, X_2, ...X_{20}$ iid Bernoulli(0.5)

$$\begin{aligned}
\mathbb{P}(Z = 1) &= \text{logit}(X_1 + X_2 + X_3 + X_4 + X_5) \\
Y_0 &= X_3 + X_4 + X_5 + X_6 + X_7 + N(0,1) \\
\mu_0 &= 3 + N(0,1) \\
Y &= Y_0 \cdot (1 - Z) + (Y_0 + \mu_0) \cdot Z = Y_0 \cdot (1 - Z) + Y_1 \cdot Z \\
Y' &= Y_0 \cdot (1 - Z) + (Y_0 + \mu_0 + \delta) \cdot Z = Y_0 \cdot (1 - Z) + (Y_1 + \delta) \cdot Z
\end{aligned} \tag{6.35}$$

sample size = 1000, case = 200, control = 800

δ	X_0'
-10	1,2,6,7
-9	1,2,6,7
-8	1,2,5,6,7
-7	1,2,3,5,6,7
-6	3,4,5,6,7
-5	1,2,3,4,5,6,7,19
-4	1,2,3,4,5,6,7,19
-3	3,4,5,6,7
-2	1,3,4,5,6,7
-1	2,3,4,5,6,7
0	1,2,3,4,5,6,7
+1	1,2,3,4,5,6,7
+2	1,3,4,5,6,7
+3	1,2,3,4,5,7
+4	1,2,3,4,5,7
+5	1,3,4,5
+6	1,3,4,5
+7	1,3,4,5
+8	1,3,4,5
+9	1,4
+10	4

Case 3:

$$
\begin{aligned}
&X_1, X_2, \ldots X_{20} \text{ iid Bernoulli}(0.5) \\
&\mathbb{P}(Z = 1) = \text{logit}(X_1 + X_2 + X_3 + X_4 + X_5) \\
&Y_0 = (X_3 + X_4 + X_5 + X_6 + X_7) \bmod 5 + N(0,1) \\
&\mu_0 = 3 + N(0,1) \\
&Y = Y_0 \cdot (1 - Z) + (Y_0 + \mu_0) \cdot Z = Y_0 \cdot (1 - Z) + Y_1 \cdot Z \\
&Y' = Y_0 \cdot (1 - Z) + (Y_0 + \mu_0 + \delta) \cdot Z = Y_0 \cdot (1 - Z) + (Y_1 + \delta) \cdot Z
\end{aligned}
\tag{6.36}
$$

sample size = 20000, case = 10000, control = 10000

δ	$X_0'(\delta) \cap X_Z(\text{ideal})$
-20	1,2,3,4
-19	1,2,3,4,5
-18	1,2,3,4
-17	1,2,3,4,5
-16	1,2,3,4,5
-15	1,2,3,4
-14	1,2,3,4
-13	1,2,3,4,5
-12	1,2,3,4,5
-11	1,2,3,4
-10	1,2,3
-9	1,2
-8	1,2,3
-7	1,2
-6	1,2
-5	1,2,3,4,5
-4	3,4,5
-3	3,4,5
-2	3,4,5
-1	3,4,5
0	3,4,5
+1	1,3,4,5
+2	1,3,4,5
+3	1,2,3,4,5
+4	1,2,3,4,5
+5	1,2,3,4,5
+6	2,3,4,5
+7	1,2,3,4,5
+8	1,3,4,5
+9	1,3,4,5
+10	1,2,3,4,5
+11	2,3,4,5
+12	2,3,4,5
+13	1,3,4,5
+14	1,3,4,5
+15	1,3,4,5
+16	1,3,4,5
+17	1,2,3,4,5
+18	1,3,4,5
+19	1,2,3,4,5
+20	1,3,4,5

Case 4:

$$
\begin{aligned}
&X_1, X_2, \ldots X_{20} \text{ iid Bernoulli}(0.5) \\
&\mathbb{P}(Z = 1) = \text{logit}(X_1 + X_2 + X_3 + X_4 + X_5) \\
&Y_0 = (X_3 + X_4 + X_5 + X_6 + X_7) \bmod 5 + N(0,1) \\
&\mu_0 = 3 + N(0,1) \qquad (6.37) \\
&Y = Y_0 \cdot (1 - Z) + (Y_0 + \mu_0) \cdot Z = Y_0 \cdot (1 - Z) + Y_1 \cdot Z \\
&Y' = Y_0 \cdot (1 - Z) + (Y_0 + \mu_0 + \delta) \cdot Z = Y_0 \cdot (1 - Z) + (Y_1 + \delta) \cdot Z
\end{aligned}
$$

sample size = 1000, case = 200, control = 800

δ	X_0'
-10	1,2,6,7
-9	1,2,6,7
-8	1,2,6,7
-7	1,2,6,7
-6	2,3,4,5,6,7
-5	3,4,5,6,7
-4	3,4,5,6,7
-3	3,4,5,6,7
-2	3,4,5,6,7
-1	3,4,5,6,7
0	3,4,5,6,7
+1	3,4,5,6
+2	3,4,5,6
+3	1,2,3,4,5,6
+4	1,3,4,5
+5	1,3,4,5
+6	1,3,4,5
+7	3,4,5
+8	3,4,5
+9	3,4,5
+10	3,4,5

6.6 Application to Vision Zero Data

6.6.1 Background

Vision Zero is a multi-national road traffic safety project that aims to achieve a highway system with no fatalities or serious injuries involving road traffic. It started in Sweden and was approved by their parliament in October 1997 [79]. A core principle of the vision is that "Life and health can never be exchanged for other benefits within the society" rather than the more conventional comparison between costs and benefits, where a monetary value is placed on life and health, and then that value is used to decide how much money to spend on a road network towards the benefit of decreasing risk [80]. The original Swedish theory hypothesizes that pedestrian deaths are not as much "accidents" as they are a failure of street design. As a result, the installment of the program emphasizes on the implementation of new safety facilities, such as increasing the numbers of speed bumps.

In New York City, Vision Zero was officially announced by New York City Mayor Bill de Blasio in 2014, whose goal is to eliminate all traffic deaths and serious injuries on New York City streets by 2024. This is an ongoing project and we shall be able to see more data as the project goes. In our analysis, we compare the increment of numbers of deaths before and after 2016.

6.6.2 Data

After the date of 01/01/2016, there are 1817 roads treated with Vision Zero in the dataset and 26657 roads that were never treated. We have 28 covariates in the original dataset, but not all of them are suitable for our analysis purpose, so we filtered some out and leave 6 major covariates. The covariates are *Borough*, *Priority Corridors*, *Priority Zones*, *Arterial Slow Zone*, *Enhanced Crossings*, and *Left Turn Traffic Calming*. The more detailed explanations will be provided in the next section. The main response is the increase in number of deaths (so the more negative the result, the more desirable). The project took place gradually over time, but here we do not take in the time effects into account; we assume that the effectiveness of the Vision Zero program has very few correlation with the time it was installed.

6.6.3 Analysis

The response we would like to study is the difference in the increment of numbers of deaths between the treated (Vision Zero installed) and the control group. Without any stratification, the grand difference is estimated by

$$\hat{\delta} = \frac{1}{N}(\sum_{i=1}^{N} Y_i \cdot \mathbb{I}(i \in \text{treated}) - \sum_{i=1}^{N} Y_i \cdot \mathbb{I}(i \in \text{control})), \tag{6.38}$$

where $\hat{\delta}$ is the estimated difference between the treated and the control, and Y_i is the individual i's increment of deaths before and after the installation of Vision Zero.

We suspect there exists some strata such that the causal effects inside each stratum is difference from the grand difference. Those strata form partitions in the space of Y, so we can decompose the grand difference into

$$\hat{\delta} = \sum_{z} \frac{n_z}{N}(\sum_{i=1}^{N} Y_i \cdot \mathbb{I}(i \in \text{treated}|Z = z) - \sum_{i=1}^{N} Y_i \cdot \mathbb{I}(i \in \text{control}|Z = z)) \tag{6.39}$$

where $\hat{\delta}$ is the estimated difference between the treated and the control, Y_i is the individual i's increment of deaths before and after the installation of Vision Zero, and Z denotes the variable that creates nonhomogeneous partitions. We would like to find out those influential covariates and their interaction that comprise these influential partitions.

Another aspect is the model of the treatment assignment. That is, in this study, we suspect the assignment of the treatment is not completely random. There may exist some confounders that both determines the treatment assignment mechanism and the outcome. However, unlike what practitioners usually perform a logistic regression to model the treatment assignment mechanism, we perform I-Score variable selection, simply because here our task is only to find out the influential variables so that we can stratify on them – hence we do not need to model the whole data.

After filtering out several unusable (e.g. too sparse) variables, we focus on the following covariates:

- *Borough* – The boroughs of New York City except Staten Island (too few samples); namely, Brooklyn, Manhattan, Queens, and The Bronx.
- *Priority Corridors* – TRUE/FALSE whether intersection was in a corridor prioritized by Vision Zero.
- *Priority Zones* – TRUE/FALSE whether intersection was in a zone prioritized by Vision Zero.
- *Arterial Slow Zone* – The date slow zone built.
- *Enhanced Crossings* – The date enhanced crossing built.
- *Left Turn Traffic Calming* – The date traffic calming built.

We performed I-Score variable selection with backward dropping algorithm – one for the response variable, and the other for the treatment assignment. Each time there are four covariates being selected randomly and the backward dropping algorithm proceeds until we find the highest I-Score. The results suggest that the combination of Prioritized Corridor and the Borough, and the combination of Arterial Slow Zone and Prioritized Corridor gave the top two highest I-Scores. Below we demonstrate the covariate effects using different graphs:

First, in Figure 6.1, the grand difference between the treated and the control is shown using a violin plot, where the green one is the treated and the red one is the control. We can see that the treated roads have much less increase in the number of deaths. However, there is no decrease in the number of deaths being observed in the treated roads, while there are some decrease being observed in the control group. This may suggest that, since the City Government prioritizes to treat roads with more serious past accidents, the outcome of the policy so far have decreased the number of deaths that would have been had the Vision Zero not been done on them, but not so intense to the point to actually reverse the trend.

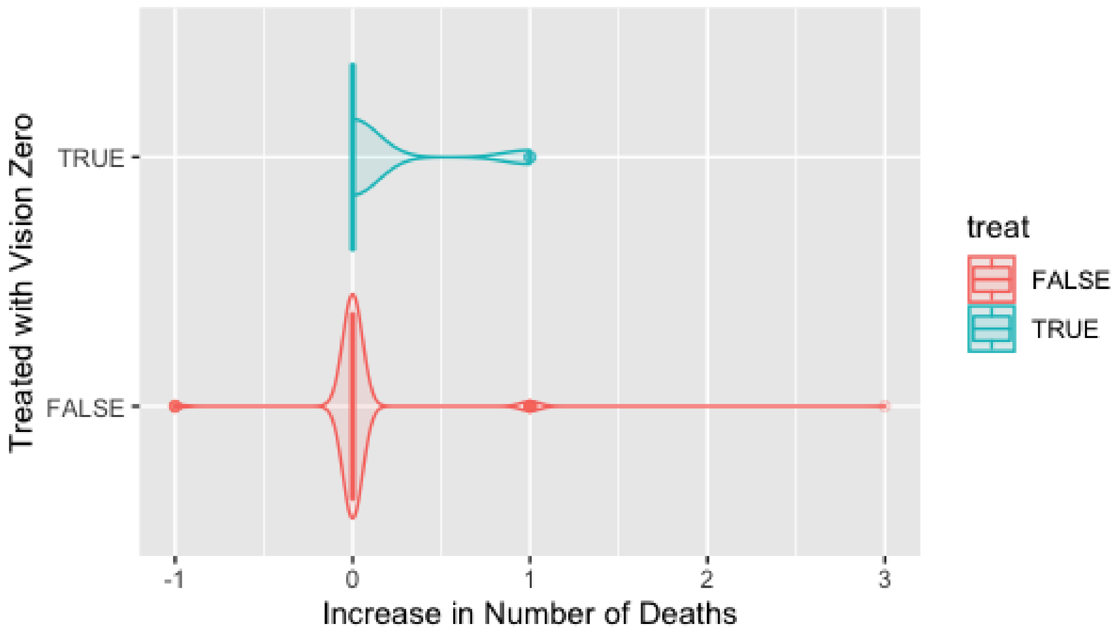


Figure 6.1: The comparison between case and control without any conditioning.

Next, let us look at the marginal effects of each covariate. From here on, we can interpret each subgraph as one partition. Figure 6.2 shows the comparison of these two groups when conditioned on different boroughs. Although they are slightly deviated from the grand trend, but the deviation isn't too far.

Figure 6.3 shows the comparison of these two groups when conditioned on the date when enhanced crossings were built. Figure 6.4 shows the comparison of these two groups when conditioned on if the corridor was prioritized. Figure 6.4 clearly shows that whether the corridor was prioritized makes a difference in the increment of number of deaths in the two groups.

Figure 6.5 shows the comparison of these two groups when conditioned on if the zone was prioritized. Figure 6.6 shows the comparison of these two groups when conditioned on the

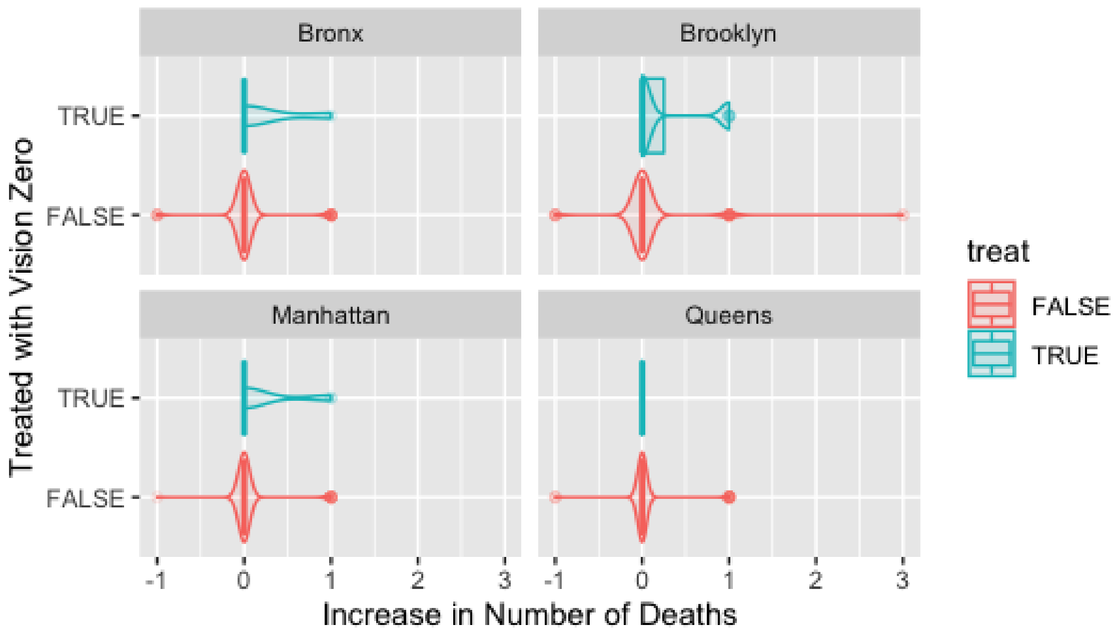


Figure 6.2: The comparison between case and control conditioned on borough.

date when slow zones were built. Figure 6.6 shows conditioning on slow zones has an impact on the distribution of the increase in number of deaths between the two groups.

For interaction effects, the top two combination suggested by I-Score variable selection algorithm are the combination of Prioritized Corridor and the Borough, and the combination of Arterial Slow Zone and Prioritized Corridor. The conditional violin plots are shown in Figure 6.7 and 6.8, respectively.

Finally, we combine the three variables together and have the following partitions created as the representation of the data, shown in Figure 6.9. We can clearly see that most of the distribution of the treated and the control inside each partition differ some the grand trend. We suggest that it is worthy of more investigation given more future data. Since now many partitions have imbalance data inside, conclusions made from the current data may not be sufficient for explaining the whole story.

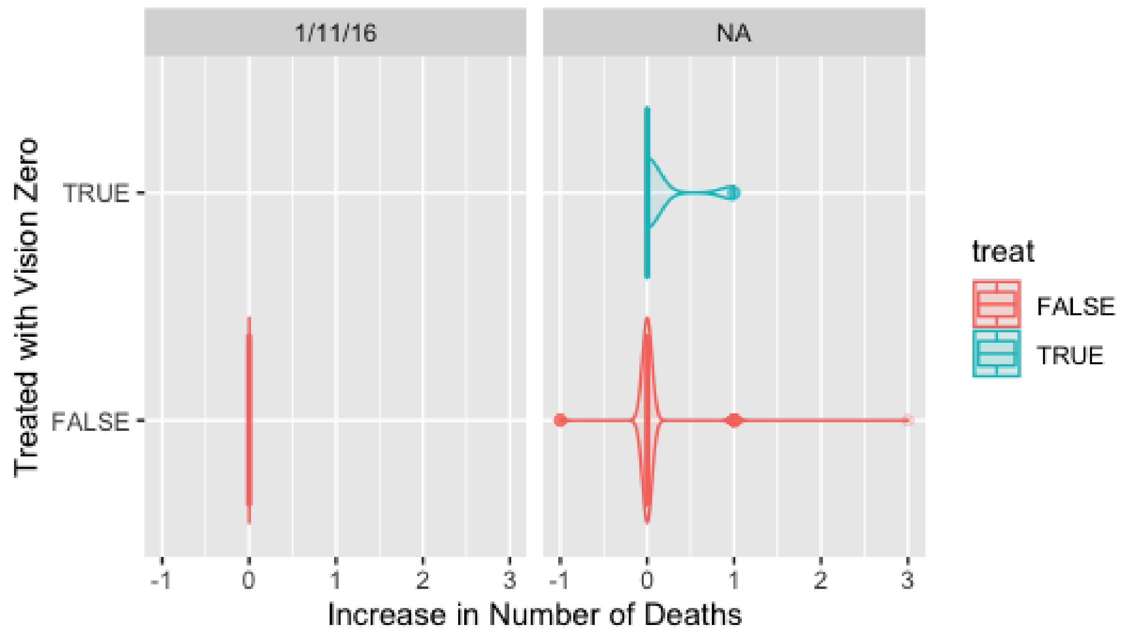


Figure 6.3: The comparison between case and control conditioned on the date when enhanced crossings were built.

6.7 Conclusion

In this chapter, we proposed the following novel ideas and methods and how representation learning can incorporate causal inference inside to yield a richer model and results.

(a) Provide a quantitative definition of the confounder set under the potential outcome framework.

(b) Provide several possible candidates for the adjustment sets.

(c) Provide a practical variable selection method for searching for the adjustment sets (1) as a whole (i.e. taking the joint interactions into account), and (2) without testing for confounding.

(d) Applied the proposed procedure on New York City Vision Zero data for causal analysis.

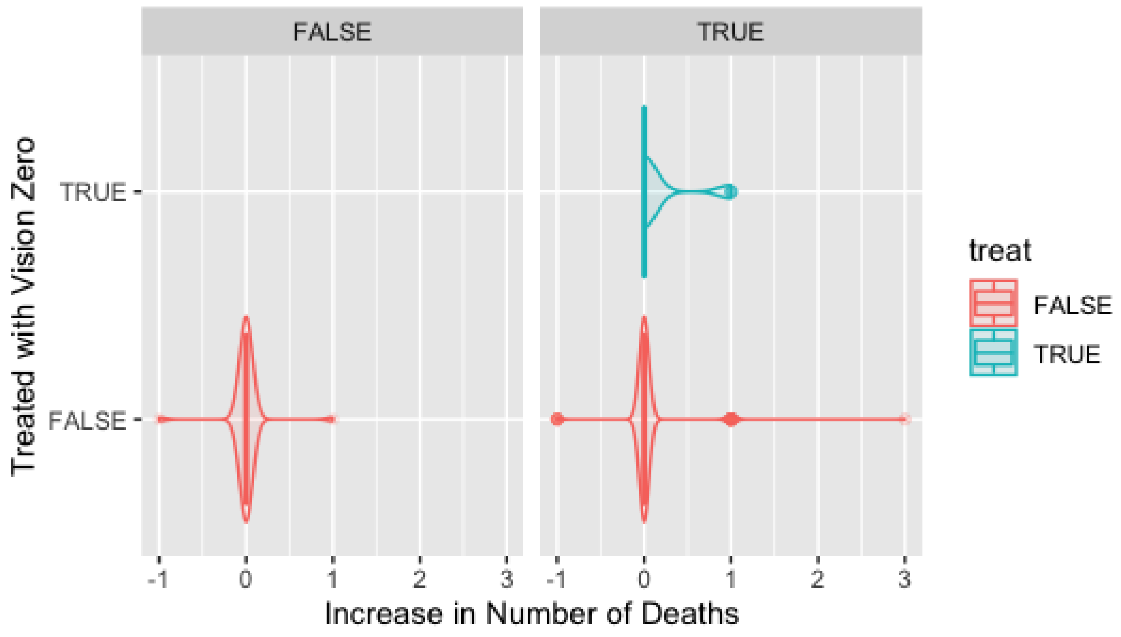


Figure 6.4: The comparison between case and control conditioned on if the corridor was prioritized.

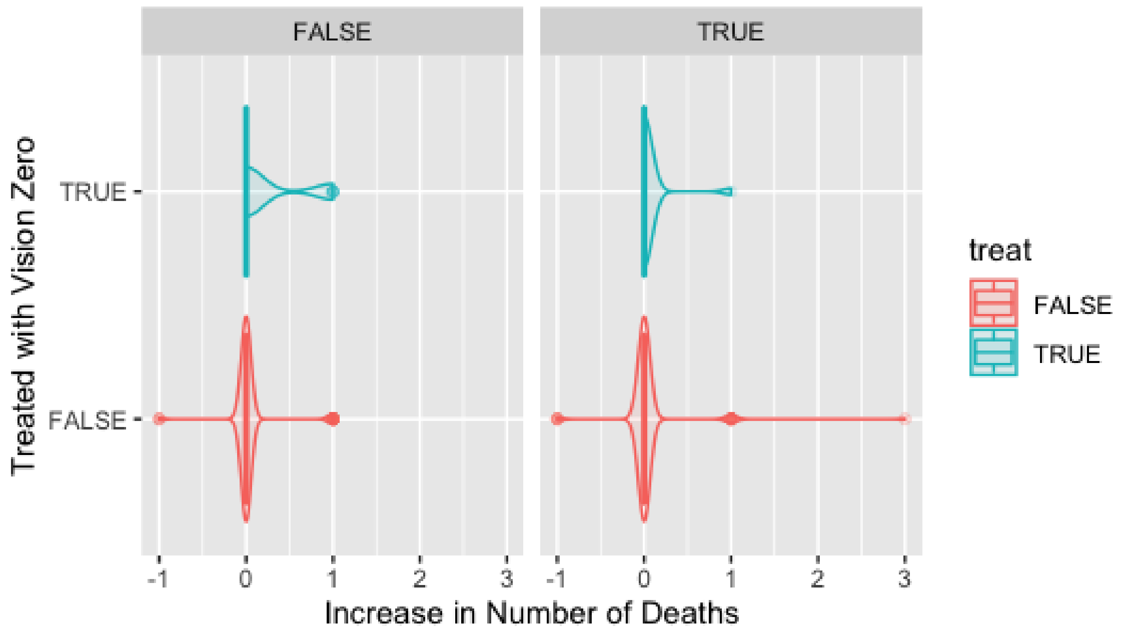


Figure 6.5: The comparison between case and control conditioned on if the zone was prioritized.

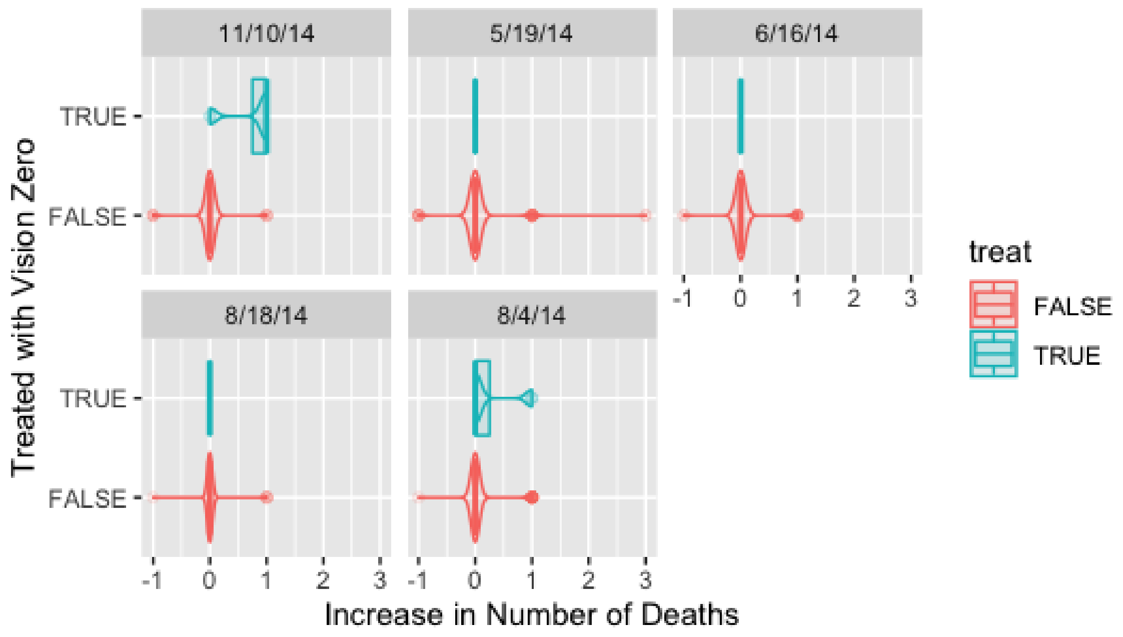


Figure 6.6: The comparison between case and control conditioned on the date when slow zones were built.

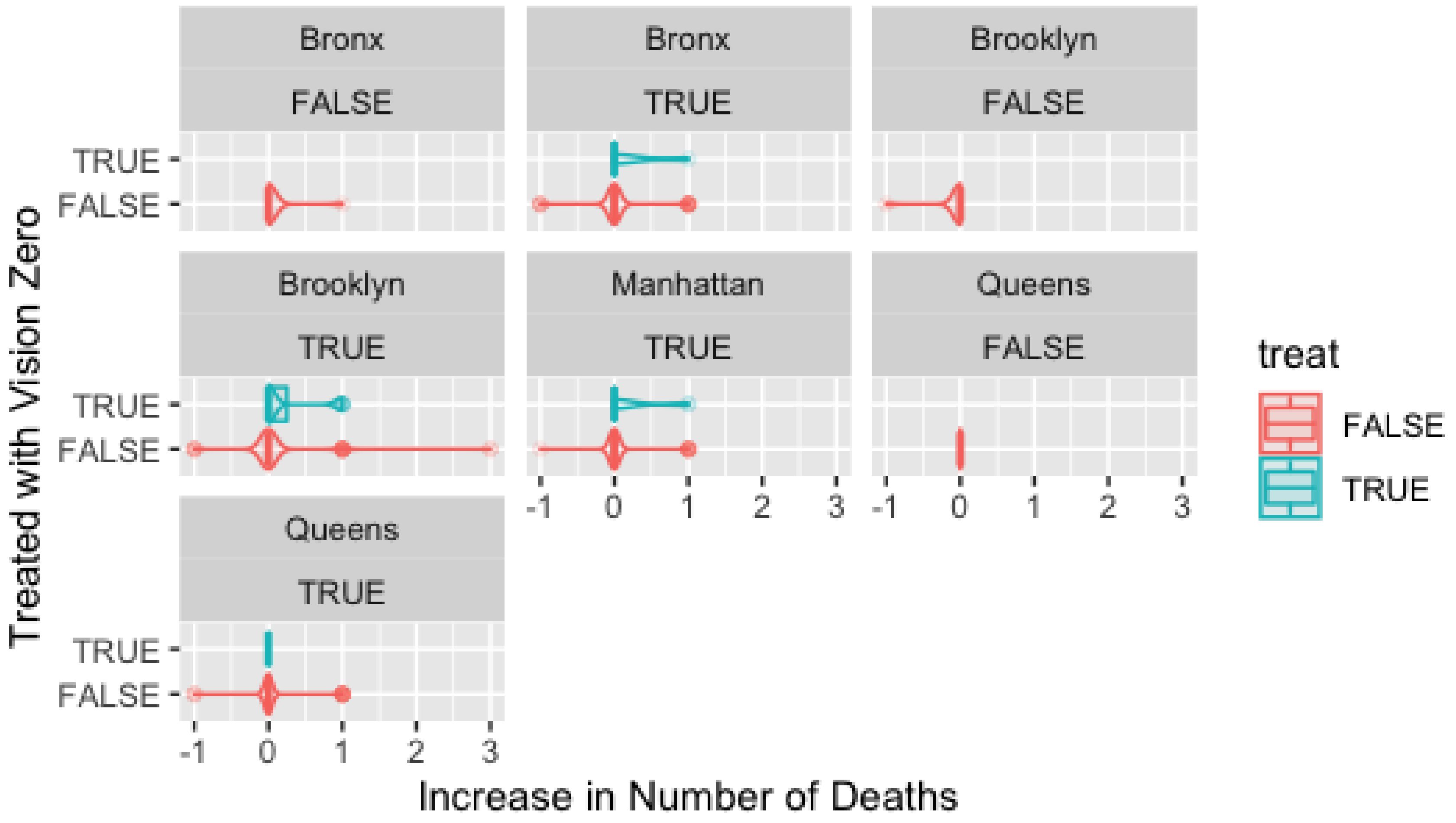


Figure 6.7: The comparison between case and control conditioned on the borough and the prioritized corridor.

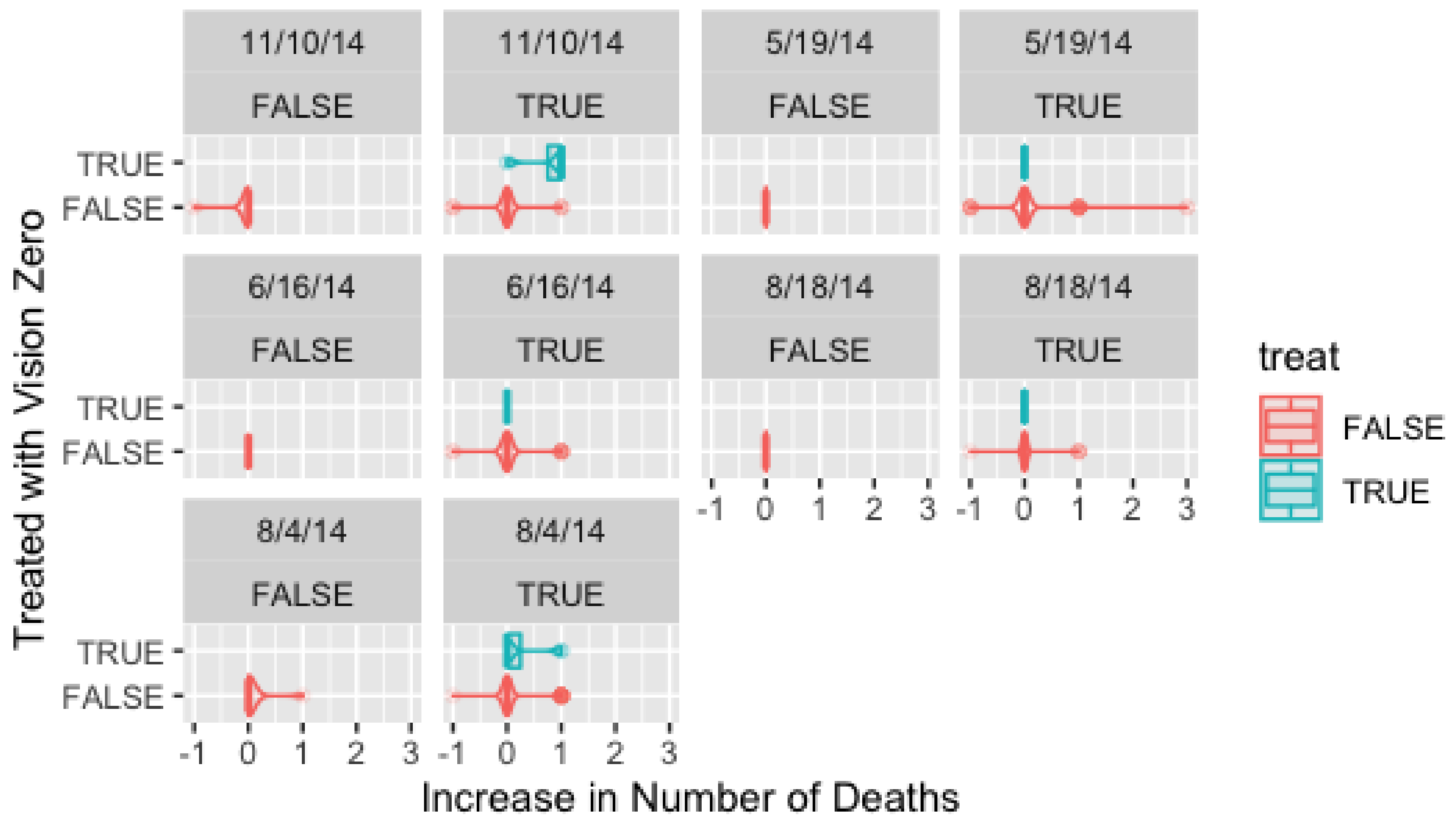


Figure 6.8: The comparison between case and control conditioned on the slow zone and the prioritized corridor.

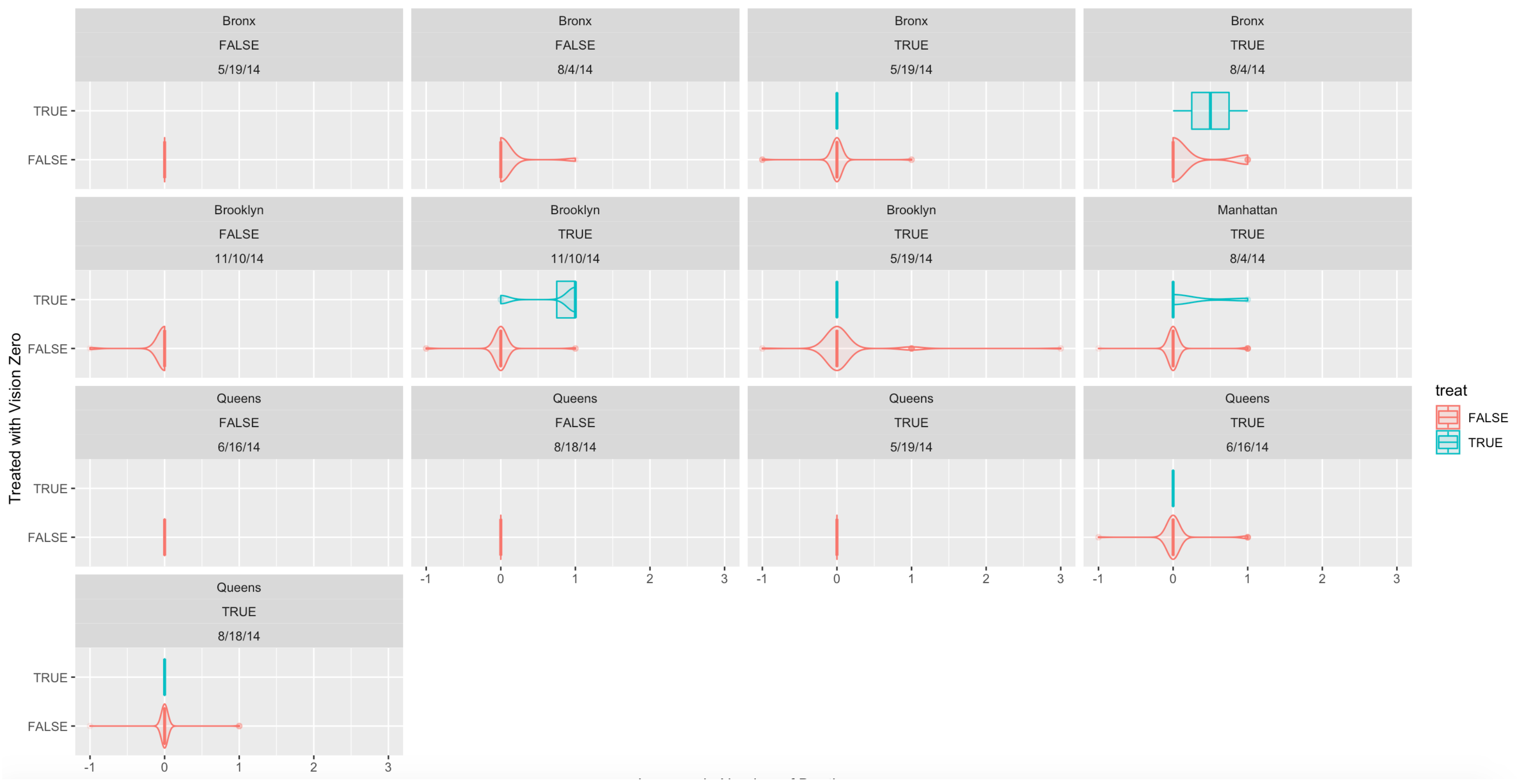


Figure 6.9: The final partitions chosen from I-Score procedure.

Chapter 7: I-Score Representation Learning in Neural Networks

7.1 Introduction

Neural networks have been credited as one of the major breakthroughs in artificial intelligence of the recent decades [81]. Neural networks are a set of algorithms, modeled loosely after the human brain, that are designed to recognize patterns. Most of today's neural networks are organized into layers of nodes, and they are feed-forward, meaning that data moves through them in only one direction. An individual node might be connected to several nodes in the layer beneath it, from which it receives data, and several nodes in the layer above it, to which it sends data.

Neural networks help us cluster and classify data. We may think of them as a clustering and classification layer on top of the data you store and manage. They help to group unlabeled data according to similarities among the example inputs, and they classify data when they have a labeled dataset to train on. Neural networks can also extract features that are fed to other algorithms for clustering and classification.

Although the idea of neural networks is decent, the model contains too many parameters that wasn't able to be handled easily. The first trainable neural network, the Perceptron, was demonstrated by the Cornell University psychologist Frank Rosenblatt in 1957. The Perceptron's design was much like that of the modern neural net, except that it had only one layer with adjustable weights and thresholds, sandwiched between input and output layers.

It was not until later when modern computational power took flight did the training of neural networks become possible. What's more, researchers started to experiment on stacking layers of neural networks together, which much resembles the real structure of neurons in a brain. By stacking layers of neurons, the model becomes "deep" and hence the name deep learning emerged.

Based on the success of basic neural networks, researchers came up with many different new model architectures, each aiming at different tasks. Among which, the one we are going to see in this chapter is the variational autoencoder (VAE) and the generative adversarial network (GAN).

7.2 Variational Autoencoder and Information Bottleneck

7.2.1 Introduction

Generally, a neural network can be treated as a sequence of representation learning. Tishby [82] argued that the success of deep learning is in part due to the capability of neural networks to build incrementally better representations that expose the relevant variability, while at the same time discarding nuisances. This interpretation is intriguing, as it establishes a connection between machine learning, probabilistic inference, information theory, and even artificial intelligence. However, common training practice does not seem to stem from this insight, and indeed deep networks may maintain even in the top layers dependencies on easily ignorable nuisances. In this work, we actually would like to elaborate on Tishby's original information bottleneck [83] , through the lens of our sufficient representation notion. Our main augmentation to the original theory is that not all latent representations are qualified for being the information bottleneck, they have to satisfy the sufficiency condition. Same arguments were also proposed by Alemi et al. [84].

7.2.2 Variational Autoencoder

As shown in Figure 7.1, we have the model that starts with the latent generating factor $\boldsymbol{C}$ (Let's assume it is the "true" generating factor of X.) and goes through a cascade of transformations until reaching the observed data $\boldsymbol{X}$. It is because of this special feed-forward structure that we can interpret the model as building up a series of "representations" of the data. Note that since $\boldsymbol{C}$ may be stochastic, even if each layer of cascading functions are deterministic, the outputs at each layer can still be stochastic. However, we can decompose or re-parameterize each layer into the deterministic part and the stochastic part.

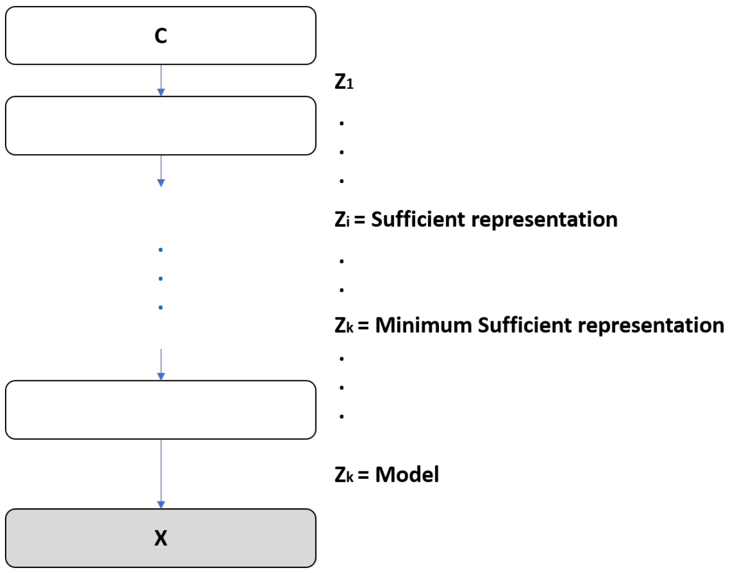

C
Z1
Zi = Sufficient representation
Zk = Minimum Sufficient representation
Zk = Model
X

Figure 7.1: Deep artificial neural network as hierarchical representation learning.

Variational autoencoders (VAEs) were defined in 2013 by Kingma et al. and Rezende et al.. In neural net language, a variational autoencoder consists of an encoder, a decoder, and a loss function. As illustraed in Figure 7.2, the encoder is a neural network, whose input is a datapoint X and output is a hidden representation Z. and it has weights and biases θ. For example, X can be a 28-by-28 pixel photo of a handwritten number. The encoder "encodes" the data which is 784-dimensional into a latent (hidden) representation space Z, which is much less than 784 dimensions. This is typically referred to as a "bottleneck"' because the encoder must learn an efficient compression of the data into this lower-dimensional space. Let's denote the encoder $q_\theta(z|x)$. Note that the lower-dimensional space is stochastic: the encoder outputs parameters to $q_\theta(z|x)$, which is a Gaussian probability density. We can sample from this distribution to get noisy values of the representations Z.

The decoder is another neural network. Opposite to the encoder, the input of a decoder is the representation Z. It outputs the parameters to the probability distribution of the data, and has weights and biases ϕ. The decoder is denoted by $p_\phi(x|z)$. Running with the handwritten digit example, let's say the photos are black and white and each pixel can take value only either 0 or 1. The probability distribution of a single pixel can be then represented using a Bernoulli distribution. The decoder gets as input the latent representation of a digit Z and outputs 784 Bernoulli parameters, one for each of the 784 pixels in the image. The decoder "decodes" the real-valued numbers in Z into 784 real-valued numbers between 0 and 1. Information from the original 784-dimensional vector cannot be perfectly transmitted, because the decoder only has access to a summary of the information (in the form of a less than 784-dimensional vector Z.

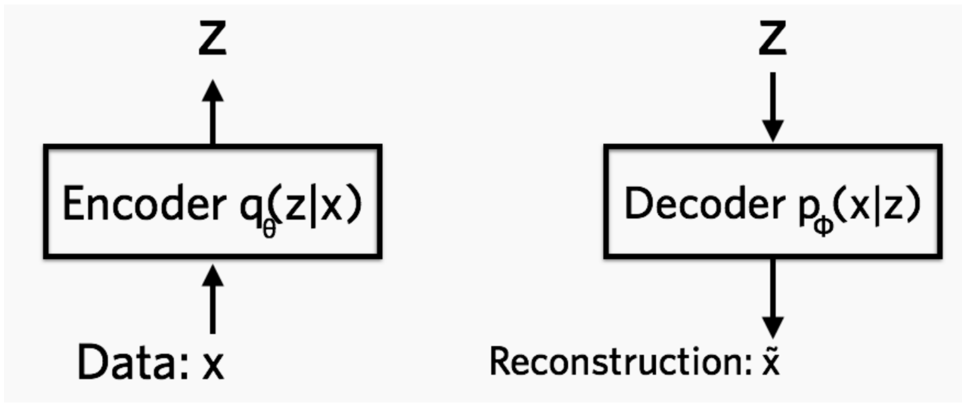


Figure 7.2: The encoder compresses data into a latent representation, and the decoder reconstructs the data given the hidden representation.

7.2.3 Representation Learning In Neural Networks

The main idea here is to establish cascading representations and its translation to probabilistic language. Assume we have total l layers in the network, we denote the output of each layer as $\{\boldsymbol{Z_1}, \boldsymbol{Z_2}, ..., \boldsymbol{Z_l}\}$. By construction, the network defines a generative process

$$p(\boldsymbol{X}) = \sum_{c,z_1,...z_l} p(\boldsymbol{X}|\boldsymbol{Z_l}, \boldsymbol{Z_{l-1}}, ..., \boldsymbol{C})p(\boldsymbol{Z_l}|\boldsymbol{Z_{l-1}}, \boldsymbol{Z_{l-2}}, ..., \boldsymbol{C})...p(\boldsymbol{Z_1}|\boldsymbol{C})p(\boldsymbol{C}) \tag{7.1}$$

Each of the terms $p(\boldsymbol{Z_i}|\boldsymbol{Z_{i-1}})$ may denote a complicated non-linear relationship; however, it is assumed that each $p(\boldsymbol{Z_i}|\boldsymbol{Z_{i-1}})$ can be sampled using a neural network. At each layer, we also need the conditional distribution to be independent from random noise; namely,

$$p(\boldsymbol{Z_i}|\boldsymbol{Z_{i-1}}, \epsilon) = p(\boldsymbol{Z_i}|\boldsymbol{Z_{i-1}}), \tag{7.2}$$

where ϵ is any arbitrary noise that is independent to the data generation. This expresses the idea that, at each layer, $\boldsymbol{Z_{i-1}}$ explains away the causal relationship to $\boldsymbol{Z_i}$, against all other noisy signals.

We define the sufficient representation $\boldsymbol{Z_k}$ appears when

Definition 7.2.1.

$\boldsymbol{Z_k}$ is the sufficient representation in deep learning when

$$\begin{aligned}
p(\boldsymbol{X}) &= \sum_{c,z_1,\dots z_l} p(\boldsymbol{X}|\boldsymbol{Z_l}, \boldsymbol{Z_{l-1}}, ..., \boldsymbol{C})p(\boldsymbol{Z_l}|\boldsymbol{Z_{l-1}}, \boldsymbol{Z_{l-2}}, ..., \boldsymbol{C})...p(\boldsymbol{Z_1}|\boldsymbol{C})p(\boldsymbol{C}) && (7.3)\\
&= \sum_{z_k,\dots z_l} p(\boldsymbol{X}|\boldsymbol{Z_l}, \boldsymbol{Z_{l-1}}, ..., \boldsymbol{Z_k})p(\boldsymbol{Z_l}|\boldsymbol{Z_{l-1}}, \boldsymbol{Z_{l-2}}, ..., \boldsymbol{Z_k})...p(\boldsymbol{Z_{k+1}}|\boldsymbol{Z_k})p(\boldsymbol{Z_k}) && (7.4)\\
&= \sum_{z_k} \sum_{z_{k+1},\dots z_l} p(\boldsymbol{X}|\boldsymbol{Z_l}, \boldsymbol{Z_{l-1}}, ..., \boldsymbol{Z_k})p(\boldsymbol{Z_l}|\boldsymbol{Z_{l-1}}, \boldsymbol{Z_{l-2}}, ..., \boldsymbol{Z_k})...p(\boldsymbol{Z_{k+1}}|\boldsymbol{Z_k})p(\boldsymbol{Z_k}) && (7.5)\\
&= \sum_{z_k} p(\boldsymbol{Z_k}) \sum_{z_{k+1},\dots z_l} p(\boldsymbol{X}|\boldsymbol{Z_l}, \boldsymbol{Z_{l-1}}, ..., \boldsymbol{Z_k})p(\boldsymbol{Z_l}|\boldsymbol{Z_{l-1}}, \boldsymbol{Z_{l-2}}, ..., \boldsymbol{Z_k})...p(\boldsymbol{Z_{k+1}}|\boldsymbol{Z_k}) && (7.6)
\end{aligned}$$

In other words, conditioned on the $\boldsymbol{Z_k}$ layer, all the other previous layers are independent from layers after $\boldsymbol{Z_k}$.

We can also interpret this model as the normalizing flow [], where each layer simply acts as adding on the complexity and expressiveness of the model.

The part $\sum_{z_{k+1},\dots z_l} p(\boldsymbol{X}|\boldsymbol{Z_l}, \boldsymbol{Z_{l-1}}, ..., \boldsymbol{Z_k})p(\boldsymbol{Z_l}|\boldsymbol{Z_{l-1}}, \boldsymbol{Z_{l-2}}, ..., \boldsymbol{Z_k})...p(\boldsymbol{Z_{k+1}}|\boldsymbol{Z_k}) = p(\boldsymbol{X}|\boldsymbol{Z_k})$ is what we defined as the representation. Furthermore, we say this $\boldsymbol{Z_k}$ is the sufficient representation in deep learning, which is sufficient for the reconstruction of $\boldsymbol{X}$. $\boldsymbol{X}$ can be modeled as a stochastic function of $\boldsymbol{Z_k}$. The idea is similar to probabilistic PCA, except that the transformation from $\boldsymbol{X}$ to the hidden space $\boldsymbol{Z}$ can be non-linear, and the solution isn't unique.

7.2.4 Related Work – Factor Analysis/Probabilistic PCA

In classical statistics, factor analysis is one of the tools that can be used to find out the latent structure inside the data. If we have data $\boldsymbol{X}$ and latent factor $\boldsymbol{F}$, the model can be written as

$$\boldsymbol{X} = \boldsymbol{\mu} + \boldsymbol{\Lambda F} + \boldsymbol{\epsilon}, \tag{7.7}$$

where $X \in \mathbb{R}^{p\times 1}, \mu$ (the global mean) $\in \mathbb{R}^{p\times 1}, \Lambda \in \mathbb{R}^{p\times m}, F \in \mathbb{R}^{m\times 1}$, and $\epsilon \in \mathbb{R}^{p\times 1}$.

If we assume that $\mathbb{E}(F) = 0, \mathbb{E}(\epsilon) = 0, \mathbf{Cov}(F) = \mathbb{E}(FF^T) = I, \mathbf{Cov}(\epsilon) = \Psi$, and $\mathbf{Cov}(F, \epsilon) = 0$, then

$$\begin{aligned}
\mathbf{Cov}(\boldsymbol{X}) &= \mathbb{E}[(X-\mu)(X-\mu)^T] && (7.8)\\
&= \mathbb{E}[(\Lambda F + \epsilon)(\Lambda F + \epsilon)^T] && (7.9)\\
&= \mathbb{E}[\Lambda F F^T \Lambda^T + \Lambda F \epsilon^T + \epsilon F^T \Lambda^T + \epsilon\epsilon^T] && (7.10)\\
&= \Lambda \mathbb{E}(FF^T)\Lambda^T + \Lambda \mathbb{E}(F\epsilon^T) + \mathbb{E}(\epsilon F^T)\Lambda^T + \mathbb{E}(\epsilon\epsilon^T) && (7.11)\\
&= \Lambda \boldsymbol{I} \Lambda^T + \Lambda 0 + 0\Lambda^T + \Psi && (7.12)\\
&= \Lambda\Lambda^T + \Psi && (7.13)
\end{aligned}$$

Consider a data set $\mathbf{X} = \{\mathbf{x}_n\}$ of N data points, where each data point is D-dimensional, $\mathbf{x}_n \in \mathbb{R}^D$. We aim to represent each $\mathbf{x}_n$ under a latent variable $\mathbf{z}_n \in \mathbb{R}^K$ with lower dimension, $K < D$. The set of principal axes $\mathbf{W}$ relates the latent variables to the data.

Specifically, we assume that each latent variable is normally distributed, $\mathbf{z}_n \sim N(\mathbf{0}, \mathbf{I})$.

The corresponding data point is generated via a projection, $\mathbf{x}_n \mid \mathbf{z}_n \sim N(\mathbf{W}\mathbf{z}_n, \sigma^2\mathbf{I})$, where the matrix $\mathbf{W} \in \mathbb{R}^{D\times K}$ are known as the principal axes. In probabilistic PCA, we are typically interested in estimating the principal axes $\mathbf{W}$ and the noise term σ^2.

Probabilistic PCA generalizes classical PCA. Marginalizing out the the latent variable, the distribution of each data point is $\mathbf{x}_n \sim N(\mathbf{0}, \mathbf{W}\mathbf{W}^Y + \sigma^2\mathbf{I})$.

Classical PCA is the specific case of probabilistic PCA when the covariance of the noise becomes infinitesimally small, $\sigma^2 \to 0$.

Conceptually, a VAE has the same property to transform the observed data into some latent factors. What is different is that a VAE does so by using a simulated neural network, instead of an analytical matrix decomposition.

7.2.5 Inference Method

The objective function for a VAE is the negative log-likelihood with a regularizer; namely,

$$l_i(\theta, \phi) = -\mathbf{E}_{z\sim q_\theta(z|x_i)}[\log p_\phi(x_i|z)] + \mathbf{KL}[q_\theta(z|x_i)||p(z)] \tag{7.14}$$

We have introduced this formulation earlier when we discussed about variational inference. The reuse of the variational equality into a neural network model is the insight of Kingma et al. [40] The objective function is actually a penalization of model complexity for the KL-divergence. So really what is happening is that we want some latent variable model with priors and its corresponding objective function. The first term is the reconstruction loss, or expected negative log-likelihood of the i^{th} datapoint. The expectation is taken with respect to the encoder's distribution over the representations. This term encourages the decoder to learn to reconstruct the data. If the decoder's output does not reconstruct the data well, the decoder parameterizes a likelihood distribution that does not place much probability mass on data. The second term is a regularizer that we throw in. This is the KL-divergence between the encoder's distribution $q_\theta(z|x)$ and $p(z)$. This divergence measures how close q is to p and thus quantifies the information lost when using q to represent p. In the VAE, p is specified as a standard Normal distribution. If the encoder outputs representations z that are different than those from a standard normal distribution, it will receive a penalty in the loss.

We now discuss the inference in terms of representation learning. The notion of sufficient representation z_k satisfies the condition $MI(x; z_k) = MI(x; c)$, which is equivalent to $\mathbb{E}_{x,z}(\log p(x|z_k)) = \mathbb{E}_{x,c}(\log p(x|c))$. Since the mutual information has a natural bound; that is, $MI(x; x) \geq MI(x; z)$, we will use it as the measure of dependency between variables, as opposed to other measures such as covariance that relies on linearity assumption.

Here we present the information bottleneck method proposed by Tishby et al. [82]. The assumption in this framework is that, to predict Y, all the information is in the observed X. To satisfy all the requirements, the requirement is equivalent to these mutual information criteria:

$$MI(X; Z^*) = MI(X; C), \tag{7.15}$$

$$MI(X; C|Z^*) = 0 \tag{7.16}$$

In the context of a VAE, since the task is the reconstruction of X, so $Y = \hat{X}$. In other words,

$$Z^* = argmax_z MI(\hat{X}; X) - \alpha MI(\hat{X}; Z) - \beta MI(X; \hat{X}|Z), \tag{7.17}$$

where the first term is the usual VAE learning objective, and the second and third terms are our proposed penalization terms.

7.2.6 Generative Adversarial Network and BiGAN

Generative adversarial networks (GAN) is a class of machine learning frameworks proposed by Ian Goodfellow et al. [85]. It consists of two neural networks, one the generative network and the other the discriminative network. These two networks contest with each other in a game – the generator scores high when it generates new sample that resembles the observed data so that the discriminator cannot tell them apart; while the discriminator scores high when it can accurately tell the true data apart from the synthesized. [85] [86] The main structure is illustrated in Figure 7.2.6.

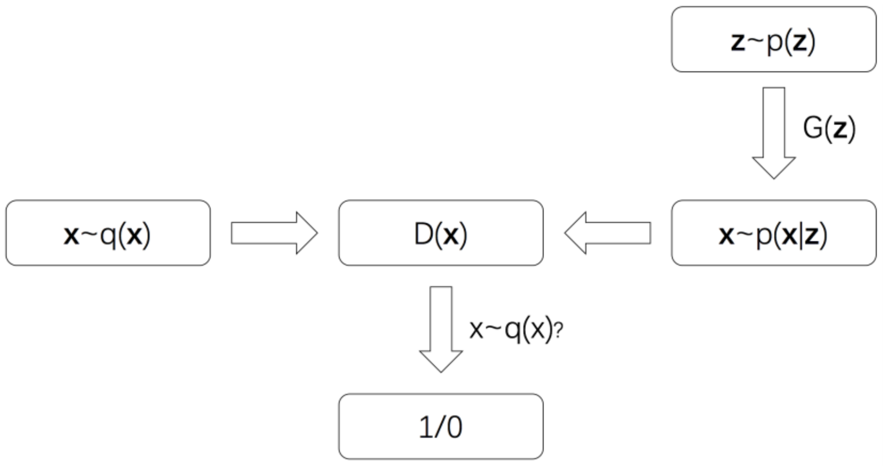


Figure 7.3: GAN architecture.

This idea shred new lights on the direction of future machine learning. Yann LeCun described it as “the most interesting idea in the last 10 years in machine learning”. [87] The novelty is that, instead of human hard-coded objective function, the objective function itself can also be learned using a model (in this case, a neural network) that is adjustable throughout the learning process. The overfitting is prevented by the competition between the two networks that have opposite interests by design.

After the initial GAN, many variants have been proposed, one of which is Bidirectional GAN (BiGAN) [88]. Similar to the initial GAN, there is a generator and a discriminator. What is different is that, since the goal of BiGAN is to infer a latent code Z, the discriminator has to tell if the joint input consisting the observation and the code is a real one or a synthesized one. The main structure is illustrated in Figure 7.2.6.

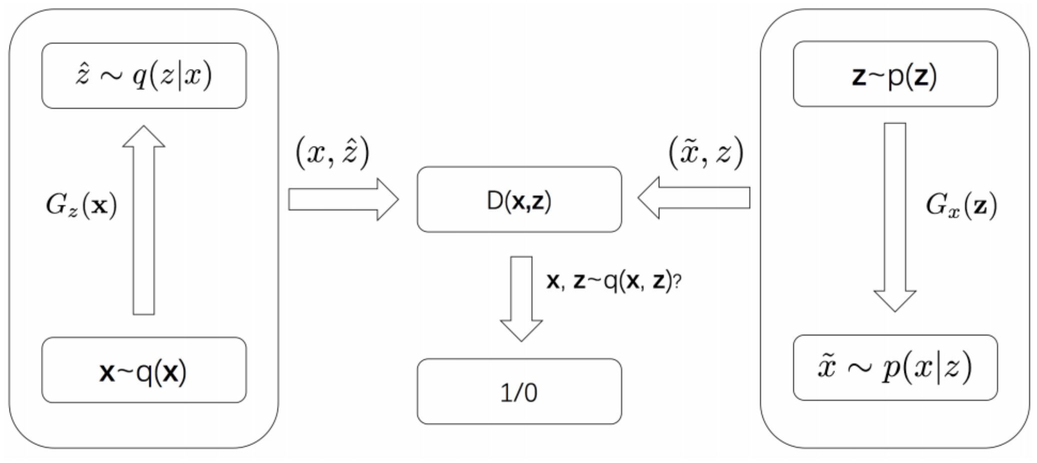
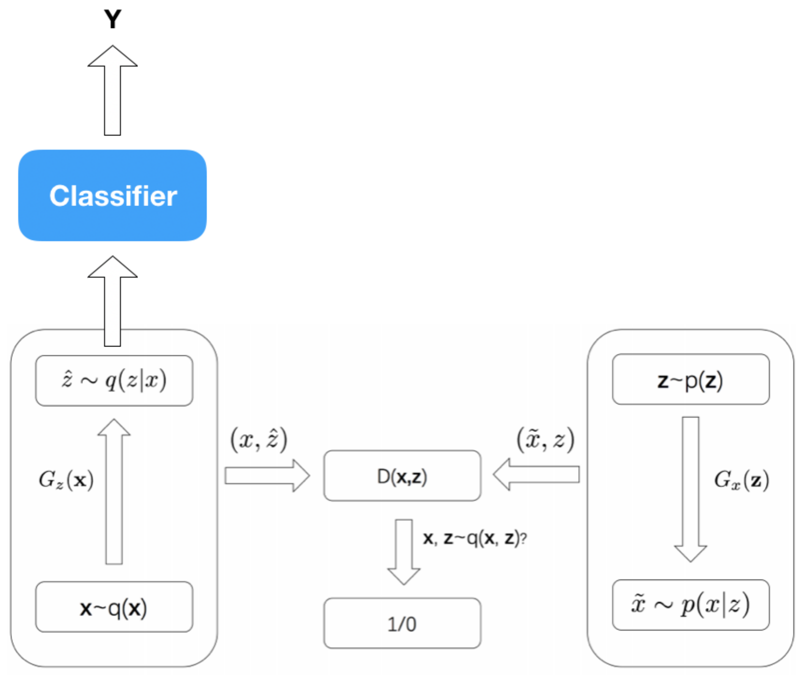


Figure 7.4: BiGAN architecture.

7.2.7 Info-BiGAN Implementation

Network Structure

We propose the following network structure:



In addition to a regular BiGAN, one of the main difference of this network structure to other fully or semi supervised network is that we build the auxiliary classifier based on the latent representation Z. From our empirical experiments, this shows an improvement in preventing mode collapse. The same phenomenon was also observed in the work of Creswell et al. [89] The intuitive explanation is that, when the auxiliary classifier is built based on Y, in order to minimize the loss, the generator would be encouraged to generate more images of some specific class that is easier to generate. In hand-written digits example, this may be the number "1". However, when the auxiliary classifier is built based on Z, then the generator just needs to focus on generating real images without the burden of prediction. It is then the encoder's job to find a good representation that can do the job of prediction and reconstruction.

Learning Procedure

Based upon the above Info-BiGAN structure, we propose to augment the objective function using the one we proposed in Equation 7.17.

$$Z \sim N(\mu(X), \sigma^2(X)) \tag{7.18}$$

$$\hat{X} = f(Z) \tag{7.19}$$

We demonstrate the learning procedure as follows:

(a) **Define the joint distribution of every variable and parameter, and decompose the likelihood up to the desired level of hierarchy. The marginal distribution of the observable variables should be invariant to different configuration of the model and hyperparameter(s).**

Let's define X_n to be the observed data, Z_n to be the corresponding latent code, θ_k to be the parameter of $X|Z$, η_k to be the parameter of Z, and M_k to be neural network k. Then we have the following decomposition.

$$p(X_n, Z_n, \theta_k, \eta_k, M_k) = p(X_n|Z_n, \theta_k, \eta_k, M_k)p(Z_n|\theta_k, \eta_k, M_k)p(\theta_k|\eta_k, M_k)p(\eta_k|M_k)p(M_k) \tag{7.20}$$

$$= p(X_n|Z_n, \theta_k, M_k)p(Z_n|\eta_k, M_k)p(\theta_k|M_k)p(\eta_k|M_k)p(M_k) \tag{7.21}$$

$$= p(X_n|Z_n, \theta_k, M_k)p(Z_n|\eta_k, M_k)p(M_k) \tag{7.22}$$

(b) **Define several candidate models, inside which the condition when signals and noises become independent should be specifically specified.**

We can try many architectures of different neural networks to serve as candidate models. The main structure here is a BiGAN, but we can try different implementations of the generator and discriminator.

(c) **Decide to use prior distributions or not. Separate the probability decomposition into the likelihood part and the prior part.**

We will follow the convention of specifying a normal distribution on the latent code Z.

(d) **Define a task for at least one of the representations.**

The encoder should be able to automatically encode observed data into a latent space where different classes are separate as much as possible.

(e) **According to the task, develop an objective function. For example, it could be MLE, MAP, or information criteria. Our general suggestion is that, for model interpretability, one would like to use likelihood based objectives; while for generative purpose, one may want to use information criteria to measure the difference between simulated and real data. For prediction purpose, usually it is more likely to be associated with likelihood based objectives.**

The target here is to make the "true" $p(X_n)$ to be close to the synthesized $p(X_n|M_k) = \mathbb{E}_{\hat{Z}} p(X_n|\hat{Z}_n, \theta_k, M_k)p(\hat{Z}_n|\eta_k, M_k)p(M_k)$ within model M_k.

(f) **Select the model and/or parameter(s) that optimizes the above objective function, call them M^* and θ^* respectively.**

We follow the usual neural networks training implemented in TensorFlow 2. The optimization is done through gradient descent.

7.2.8 Empirical Results

We use the MNIST dataset as an example to compare different generative models. Figure 7.2.8 shows the disentangling results from original BiGAN, normal BiGAN, and uniform BiGAN:

Figure 7.2.8 shows a few samples generated from the generator.

7.2.9 Observations

1. The higher dimension the representation is, the more realistic the generated images are.
2. The disentanglement effect becomes stronger as training epoch increases.
3. Different choices of $p(z)$ changes the latent space structure significantly.

7.3 Conclusion

In this chapter, we proposed a new BiGAN neural network structure and implemented it. The training involves an I-Score inspired objective function, and it showed some interesting results. Especially we demonstrated that this new objective function can be used to encourage a better disentangled latent codes.

Another novelty is to explain the mechanism of general neural networks using our representation learning framework.

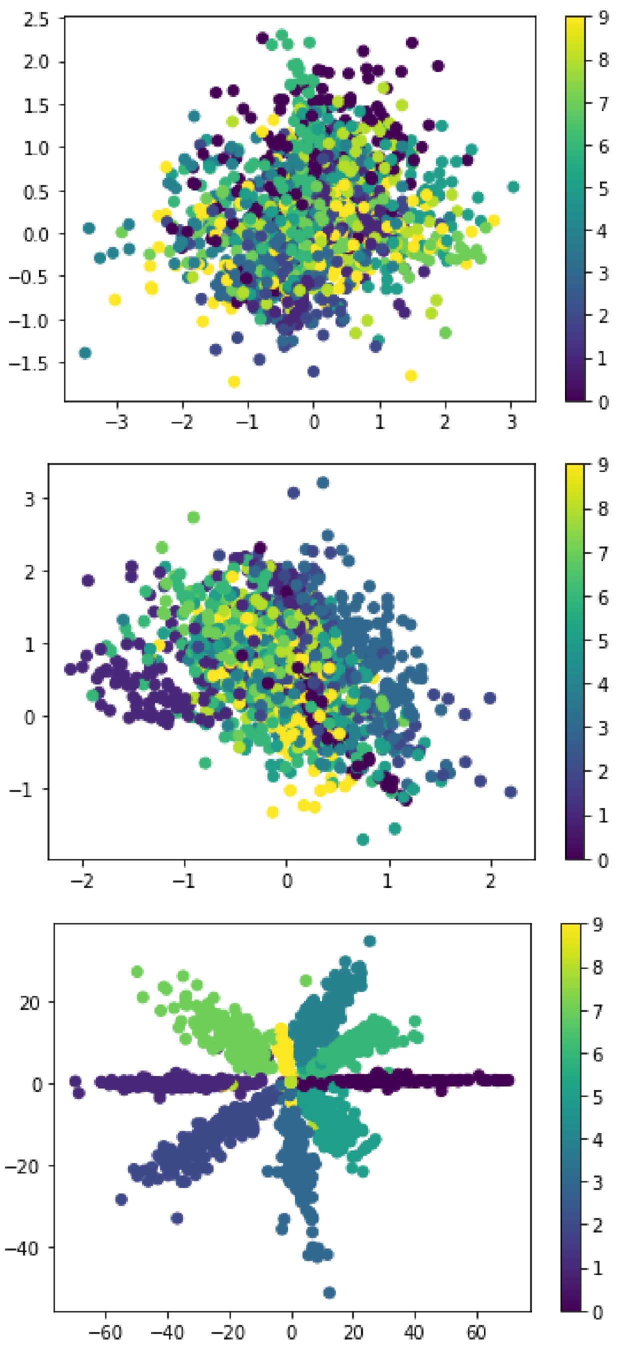


Figure 7.5: The latent space learned from VAE and Info-BiGAN with different $p(z)$. Top: VAE; Middle: Info-BiGAN with Normal prior; Bottom: Info-BiGAN with Uniform prior.

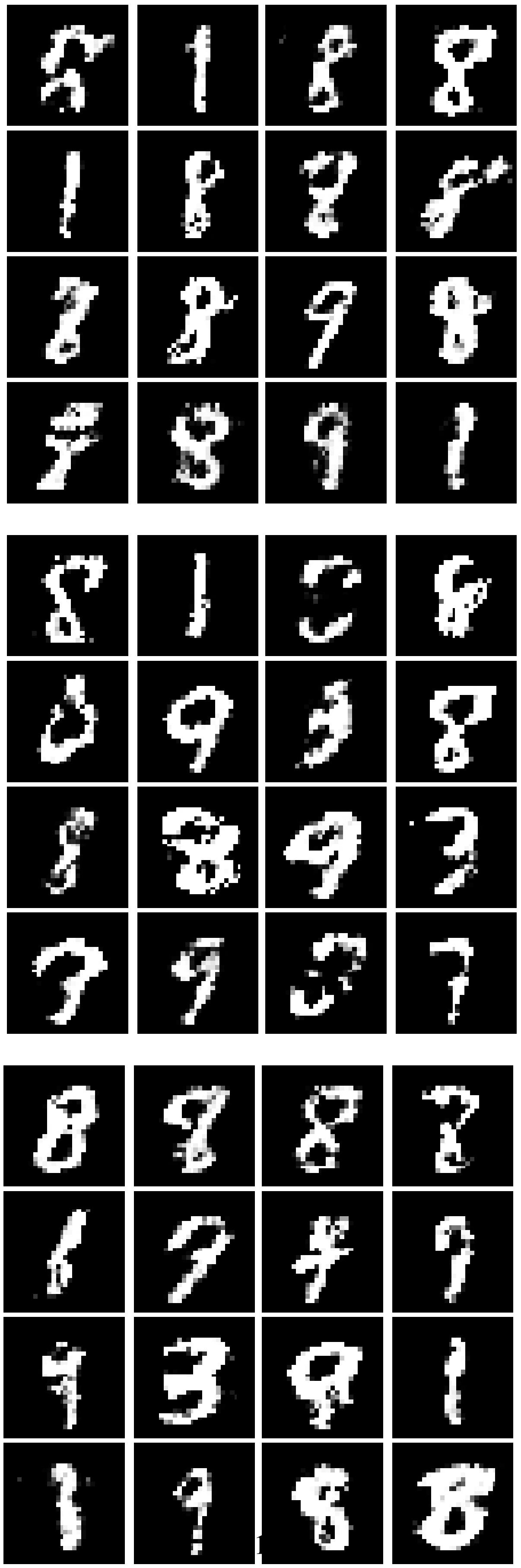


Figure 7.6: Generated hand-written digits from Info-BiGAN.

Epilogue

In this work, we proposed a new learning framework called representation learning. We introduced the motivation, the core elements, and did a few implementations on different datasets. We hope this change of mindset can help statistical machine learning more flexible and inclusive to more black-box models.

We also demonstrated the I-Score method is useful for selecting influential variables for prediction purpose without building a prediction model. The method is mainly built on partition-based models, which has a natural interpretability.

The completeness of the unifying framework still requires collaboration of people from various backgrounds. We hope this work serve as an initiative for people to start thinking about approaching problems from a different angle and to involve interpretability into the consideration when building a model and communicate with people from other fields.

References

[1] W. T. C. C. Consortium et al., “Genome-wide association study of 14,000 cases of seven common diseases and 3,000 shared controls,” Nature, vol. 447, no. 7145, p. 661, 2007.

[2] N. E. D. L. Shahum, “Vision zero,”

[3] Y. LeCun, C. Cortes, and C. Burges, “Mnist handwritten digit database,” 2010.

[4] L. Breiman et al., “Statistical modeling: The two cultures (with comments and a rejoinder by the author),” Statistical science, vol. 16, no. 3, pp. 199–231, 2001.

[5] Y. Bengio, A. C. Courville, and P. Vincent, “Unsupervised feature learning and deep learning: A review and new perspectives,” CoRR, vol. abs/1206.5538, 2012. arXiv: `1206.5538`.

[6] S. Weisberg, Applied linear regression. John Wiley & Sons, 2005, vol. 528.

[7] M. H. Kutner, C. J. Nachtsheim, J. Neter, W. Li, et al., Applied linear statistical models. McGraw-Hill Irwin New York, 2005, vol. 5.

[8] G. Molenberghs and G. Verbeke, Linear mixed models for longitudinal data. Springer, 2000.

[9] C. M. Bishop et al., Neural networks for pattern recognition. Oxford university press, 1995.

[10] S.-H. Lo and T. Zheng, “Backward haplotype transmission association (bhta) algorithm–a fast multiple-marker screening method,” Human heredity, vol. 53, no. 4, pp. 197–215, 2002.

[11] ——, “A demonstration and findings of a statistical approach through reanalysis of inflammatory bowel disease data,” Proceedings of the National Academy of Sciences, vol. 101, no. 28, pp. 10 386–10 391, 2004.

[12] H. Chernoff, S.-H. Lo, and T. Zheng, “Discovering influential variables: A method of partitions,” Ann Appl Stat, vol. 3, Sep. 2010.

[13] H. Wang, S.-H. Lo, T. Zheng, and I. Hu, “Interaction-based feature selection and classification for high-dimensional biological data,” Bioinformatics, vol. 28, no. 21, pp. 2834–2842, 2012.

[14] S. N. Wood, Generalized additive models: an introduction with R. CRC press, 2017.

[15] T. J. Hastie and R. J. Tibshirani, Generalized additive models. CRC press, 1990, vol. 43.

[16] D. Hebb, "Organization of behavior. new york: Wiley," J. Clin. Psychol, vol. 6, no. 3, pp. 335–307, 1949.

[17] B. Farley and W Clark, "Simulation of self-organizing systems by digital computer," Transactions of the IRE Professional Group on Information Theory, vol. 4, no. 4, pp. 76–84, 1954.

[18] J. Schmidhuber, "Deep learning in neural networks: An overview," Neural networks, vol. 61, pp. 85–117, 2015.

[19] A. G. Ivakhnenko and V. G. Lapa, Cybernetic predicting devices. CCM Information Corporation, 1973.

[20] A. Ivakhnenko, "Cybernetics and forecasting techniques,"

[21] J. Schmidhuber, "Deep learning," Scholarpedia, vol. 10, no. 11, p. 32 832, 2015.

[22] S. E. Dreyfus, "Artificial neural networks, back propagation, and the kelley-bryson gradient procedure," Journal of guidance, control, and dynamics, vol. 13, no. 5, pp. 926–928, 1990.

[23] E. Mizutani, S. E. Dreyfus, and K. Nishio, "On derivation of mlp backpropagation from the kelley-bryson optimal-control gradient formula and its application," in Proceedings of the IE
IEEE, vol. 2, 2000, pp. 167–172.

[24] H. J. Kelley, "Gradient theory of optimal flight paths," Ars Journal, vol. 30, no. 10, pp. 947–954, 1960.

[25] A. E. Bryson, "A gradient method for optimizing multi-stage allocation processes," in Proc. Harvard Univ. Symposium on digital computers and their applications, vol. 72, 1961.

[26] S. Linnainmaa, "The representation of the cumulative rounding error of an algorithm as a taylor expansion of the local rounding errors," Master's Thesis (in Finnish), Univ. Helsinki, pp. 6–7, 1970.

[27] ——, "Taylor expansion of the accumulated rounding error," BIT Numerical Mathematics, vol. 16, no. 2, pp. 146–160, 1976.

[28] S. Dreyfus, "The computational solution of optimal control problems with time lag," IEEE Transactions on Automatic Control, vol. 18, no. 4, pp. 383–385, 1973.

[29] P. J. Werbos, "Applications of advances in nonlinear sensitivity analysis," in System modeling and opt
Springer, 1982, pp. 762–770.

[30] M. Minsky and S. A. Papert, Perceptrons: An introduction to computational geometry.
MIT press, 2017.

[31] J. Weng, N. Ahuja, and T. S. Huang, "Cresceptron: A self-organizing neural network
which grows adaptively," in [Proceedings 1992] IJCNN International Joint Conference on Neural Ne
IEEE, vol. 1, 1992, pp. 576–581.

[32] I. Goodfellow, Y. Bengio, and A. Courville, Deep learning. MIT press, 2016.

[33] D. C. Cireşan, U. Meier, L. M. Gambardella, and J. Schmidhuber, "Deep, big, simple
neural nets for handwritten digit recognition," Neural computation, vol. 22, no. 12,
pp. 3207–3220, 2010.

[34] D. Scherer, A. Müller, and S. Behnke, "Evaluation of pooling operations in convolutional architectures for object recognition," in International conference on artificial neural networks,
Springer, 2010, pp. 92–101.

[35] J Schmidhuber, "On the eight competitions won by his deep learning team 2009–
2012," A. Kurzweil, Interviewer) Retrieved from http://www. kurzweilai. net/how-bio-inspired-deep
2012.

[36] A. Graves and J. Schmidhuber, "Offline handwriting recognition with multidimensional recurrent neural networks," in Advances in neural information processing systems,
2009, pp. 545–552.

[37] A. Graves, M. Liwicki, S. Fernández, R. Bertolami, H. Bunke, and J. Schmidhuber, "A
novel connectionist system for unconstrained handwriting recognition," IEEE transactions on pattern
vol. 31, no. 5, pp. 855–868, 2008.

[38] M. F. Abbod, J. W. Catto, D. A. Linkens, and F. C. Hamdy, "Application of artificial intelligence to the management of urological cancer," The Journal of urology,
vol. 178, no. 4, pp. 1150–1156, 2007.

[39] M. Miljanovic, "Comparative analysis of recurrent and finite impulse response neural
networks in time series prediction," Indian Journal of Computer Science and Engineering,
vol. 3, no. 1, pp. 180–191, 2012.

[40] D. P. Kingma and M. Welling, "Stochastic gradient vb and the variational auto-encoder,"
in Second International Conference on Learning Representations, ICLR, vol. 19, 2014.

[41] A. Radford, L. Metz, and S. Chintala, “Unsupervised representation learning with deep convolutional generative adversarial networks,” arXiv preprint arXiv:1511.06434, 2015.

[42] I. Goodfellow, “Nips 2016 tutorial: Generative adversarial networks,” arXiv preprint arXiv:1701.0016 2016.

[43] G. E. Box, “All models are wrong, but some are useful,” Robustness in Statistics, vol. 202, p. 549, 1979.

[44] Y. S. Kim, P. Maruvada, and J. A. Milner, “Metabolomics in biomarker discovery: Future uses for cancer prevention,” 2008.

[45] Y. Hsu, J. Auerbach, T. Zheng, and S.-h. Lo, “Coping with family structure in genome-wide association studies: A comparative evaluation,” in BMC proceedings, BioMed Central, vol. 12, 2018, pp. 235–240.

[46] R. A. Fisher, “The correlation between relatives on the supposition of mendelian inheritance.,” Earth and Environmental Science Transactions of the Royal Society of Edinburgh, vol. 52, no. 2, pp. 399–433, 1919.

[47] C. R. Henderson, “Estimation of variance and covariance components,” Biometrics, vol. 9, no. 2, pp. 226–252, 1953.

[48] E Boerwinkle, R Chakraborty, and C. Sing, “The use of measured genotype information in the analysis of quantitative phenotypes in man: Models and analytical methods,” Annals of human genetics, vol. 50, no. 2, pp. 181–194, 1986.

[49] M. Abney, C. Ober, and M. S. McPeek, “Quantitative-trait homozygosity and association mapping and empirical genomewide significance in large, complex pedigrees: Fasting serum-insulin level in the hutterites,” The American Journal of Human Genetics, vol. 70, no. 4, pp. 920–934, 2002.

[50] W.-M. Chen and G. R. Abecasis, “Family-based association tests for genomewide association scans,” The American Journal of Human Genetics, vol. 81, no. 5, pp. 913–926, 2007.

[51] Y. S. Aulchenko, D.-J. De Koning, and C. Haley, “Genomewide rapid association using mixed model and regression: A fast and simple method for genomewide pedigree-based quantitative trait loci association analysis,” Genetics, vol. 177, no. 1, pp. 577–585, 2007.

[52] H. M. Kang, J. H. Sul, S. K. Service, N. A. Zaitlen, S.-y. Kong, N. B. Freimer, C. Sabatti, E. Eskin, et al., “Variance component model to account for sample structure

in genome-wide association studies," Nature genetics, vol. 42, no. 4, pp. 348–354, 2010.

[53] Z. Zhang, E. Ersoz, C.-Q. Lai, R. J. Todhunter, H. K. Tiwari, M. A. Gore, P. J. Bradbury, J. Yu, D. K. Arnett, J. M. Ordovas, et al., "Mixed linear model approach adapted for genome-wide association studies," Nature genetics, vol. 42, no. 4, pp. 355–360, 2010.

[54] S Sawcer, G Hellenthal, M Pirinen, C. Spencer, N. Patsopoulos, L Moutsianas, A Dilthey, Z Su, C Freeman, S. Hunt, et al., "International multiple sclerosis genetics consortium wellcome trust case control consortium 2 genetic risk and a primary role for cell-mediated immune mechanisms in multiple sclerosis," Nature, vol. 476, no. 7359, pp. 214–219, 2011.

[55] J Yu, G Pressoir, and W. Briggs, "A unified mixed-model method for association mapping that accounts for multiple levels of relatedness," Nat genet, vol. 38, no. 2, pp. 203–208, 2006.

[56] N. Amin, C. M. Van Duijn, and Y. S. Aulchenko, "A genomic background based method for association analysis in related individuals," PloS one, vol. 2, no. 12, e1274, 2007.

[57] M. Fakiola, A. Strange, H. J. Cordell, E. N. Miller, M. Pirinen, Z. Su, A. Mishra, S. Mehrotra, G. R. Monteiro, G. Band, et al., "Common variants in the hla-drb1–hla-dqa1 hla class ii region are associated with susceptibility to visceral leishmaniasis," Nature genetics, vol. 45, no. 2, pp. 208–213, 2013.

[58] J Eu-Ahsunthornwattana, E. Miller, and M Fakiola, "Comparison of methods to account for relatedness in genome-wide association studies with family-based data," PLoS Genet, vol. 10, no. 7, e1004445, 2014.

[59] M. R. Irvin, E. K. Kabagambe, H. K. Tiwari, L. D. Parnell, R. J. Straka, M. Tsai, J. M. Ordovas, and D. K. Arnett, "Apolipoprotein e polymorphisms and postprandial triglyceridemia before and after fenofibrate treatment in the genetics of lipid lowering and diet network (goldn) study," Circulation: Cardiovascular Genetics, vol. 3, no. 5, pp. 462–467, 2010.

[60] M. R. Irvin, D. Zhi, R. Joehanes, M. Mendelson, S. Aslibekyan, S. A. Claas, K. S. Thibeault, N. Patel, K. Day, L. W. Jones, et al., "Epigenome-wide association study of fasting blood lipids in the genetics of lipid-lowering drugs and diet network study," Circulation, vol. 130, no. 7, pp. 565–572, 2014.

[61] P. W. Holland, "Statistics and causal inference," Journal of the American statistical Association, vol. 81, no. 396, pp. 945–960, 1986.

[62] J Neyman, “Sur les applications de la theorie des probabilites aux experiences agricoles: Essai des principes (masters thesis); justification of applications of the calculus of probabilities to the solutions of certain questions in agricultural experimentation. excerpts english translation (reprinted),” Stat Sci, vol. 5, pp. 463–472, 1923.

[63] D. B. Rubin, “Causal inference using potential outcomes: Design, modeling, decisions,” Journal of the American Statistical Association, vol. 100, no. 469, pp. 322–331, 2005.

[64] P. R. Rosenbaum, “Overt bias in observational studies,” in Observational studies, Springer, 2002, pp. 71–104.

[65] N. A. Christakis and T. J. Iwashyna, “The health impact of health care on families: A matched cohort study of hospice use by decedents and mortality outcomes in surviving, widowed spouses,” Social science & medicine, vol. 57, no. 3, pp. 465–475, 2003.

[66] A. Abadie and G. W. Imbens, “Large sample properties of matching estimators for average treatment effects,” econometrica, vol. 74, no. 1, pp. 235–267, 2006.

[67] S. Galiani, P. Gertler, and E. Schargrodsky, “Water for life: The impact of the privatization of water services on child mortality,” Journal of political economy, vol. 113, no. 1, pp. 83–120, 2005.

[68] S. Galiani, P. Gertler, E. Schargrodsky, and F. Sturzenegger, “The benefits and costs of privatization in argentina: A microeconomic analysis,” Privatization in Latin America—Myths and R
2005.

[69] J. Bowers and B. Hansen, “Attributing effects to a get-out-the-vote campaign using full matching and randomization inference,” in Prepared for presentation at the Annual Meeting of th
2005.

[70] J. S. Sekhon, “Quality meets quantity: Case studies, conditional probability, and counterfactuals,” Perspectives on Politics, vol. 2, no. 2, pp. 281–293, 2004.

[71] S. L. Morgan and D. J. Harding, “Matching estimators of causal effects: Prospects and pitfalls in theory and practice,” Sociological methods & research, vol. 35, no. 1, pp. 3–60, 2006.

[72] T. A. DiPrete and H. Engelhardt, “Estimating causal effects with matching methods in the presence and absence of bias cancellation,” Sociological Methods & Research, vol. 32, no. 4, pp. 501–528, 2004.

[73] C. Winship and S. L. Morgan, “The estimation of causal effects from observational data,” Annual review of sociology, vol. 25, no. 1, pp. 659–706, 1999.

[74] D. B. Rubin, "Using propensity scores to help design observational studies: Application to the tobacco litigation," Health Services and Outcomes Research Methodology, vol. 2, no. 3-4, pp. 169–188, 2001.

[75] ——, "Estimating causal effects of treatments in randomized and nonrandomized studies.," Journal of educational Psychology, vol. 66, no. 5, p. 688, 1974.

[76] A. Kyrgidis, E. Verrou, K. Kitikidou, C. G. Andreadis, I. Katodritou, and K. Vahtsevanos, "Reply to i. abraham," Journal of Clinical Oncology, vol. 28, no. 9, e145–e147, 2010.

[77] N. S. Abramson, S. F. Kelsey, P. Safar, and K. Sutton-Tyrrell, "Simpson's paradox and clinical trials: What you find is not necessarily what you prove," Annals of emergency medicine, vol. 21, no. 12, pp. 1480–1482, 1992.

[78] C. H. Wagner, "Simpson's paradox in real life," The American Statistician, vol. 36, no. 1, pp. 46–48, 1982.

[79] S. Goodyear, "The swedish approach to road safety:"the accident is not the major problem"," written account of Goodyear's interview with Matts-Åke Belin, traffic safety strategist w vol. 5, 2014.

[80] E. Hauer, "Computing what the public wants: Some issues in road safety cost–benefit analysis," Accident Analysis & Prevention, vol. 43, no. 1, pp. 151–164, 2011.

[81] M. Ford, Architects of Intelligence: The truth about AI from the people building it. Packt Publishing Ltd, 2018.

[82] N. Tishby and N. Zaslavsky, "Deep learning and the information bottleneck principle," in 2015 IEEE Information Theory Workshop (ITW), IEEE, 2015, pp. 1–5.

[83] N. Tishby, F. C. Pereira, and W. Bialek, "The information bottleneck method," arXiv preprint physics/ 2000.

[84] A. A. Alemi, I. Fischer, J. V. Dillon, and K. Murphy, "Deep variational information bottleneck," arXiv preprint arXiv:1612.00410, 2016.

[85] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in Advances in neural information processing 2014, pp. 2672–2680.

[86] P. Luc, C. Couprie, S. Chintala, and J. Verbeek, "Semantic segmentation using adversarial networks," arXiv preprint arXiv:1611.08408, 2016.

[87] B. Biggio and F. Roli, “Wild patterns: Ten years after the rise of adversarial machine learning,” Pattern Recognition, vol. 84, pp. 317–331, 2018.

[88] R. Bao, S. Liang, and Q. Wang, “Featurized bidirectional gan: Adversarial defense via adversarially learned semantic inference,” arXiv preprint arXiv:1805.07862, 2018.

[89] A. Creswell, A. A. Bharath, and B. Sengupta, “Conditional autoencoders with adversarial information factorization,” space, vol. 19, p. 24, 2017.