Frequentist and Bayesian Perspectives on Logistic Regression and Neural Networks

Kelly Kung, Zihuan Qiao, Benjamin Draves

April 29, 2019

Abstract

Binary classification is a central problem in statistical practice. A plethora of statistical learning techniques have been developed to address this problem, each with their own benefits and limitations. In this paper, we set out to unify two such techniques, Logistic Regression and Neural Networks, and to analyze a larger model class that utilizes the binomial likelihood to connect covariate information with observed class assignments. We consider training and subsequent inference of these models under both a Frequentist and Bayesian paradigm which allows for a complete analysis of the similarity and differences that these models posit. Finally, we apply these methods to a resume dataset that contains binary features to predict whether an applicant receives an interview.

1 Introduction

A central problem in statistical practice and theory is binary classification. With applications in numerous disciplines, an arsenal of statistical techniques have been devised to relate the outcomes of an experiment to underlying covariates observed in practice. Models in this setting include decision trees, Support Vector Machines, Random Forests, Logistic Regression, and Neural Networks to name but a few. With much attention being dedicated to this problem from a multitude of disciplines, different methodologies are often discussed and analyzed within the context of its respective discipline. Two prominent examples of this behavior are the Logistic Regression model and the Neural Network model. Logistic Regression is often analyzed in a pure statistical setting, where the analysis focuses around the properties of the exponential family, maximum likelihood estimation, and statistical properties of parameter estimates (McCullagh and Nelder 1989). In contrast, Neural Networks are commonly analyzed through a computational guise, where gradient descent, penalization, and test set performance are the central focus of analysis (Hastie, Tibshirani, and Friedman 2009).

While these models are frequently discussed in different settings, they are a part of a larger model class that leverage the binomial likelihood to relate the parameters of the model to the parameters of the probability model of the data. In this project, we study this broad model class under both a Frequentist and Bayesian framework. In doing so, we will simultaneously examine the benefits and limitations of considering a

Logistic Regression framework as compared to a Neural Network model as well as adopting a Frequentist and Bayesian approach to parameter inference. We will compare these four approaches to binary classification with respect to model performance, running time, theoretical guarantees, as well as model interpretability.

To make our discussion rigorous, suppose that we have data $\mathcal{T} = \{(\mathbf{x}_i, y_i)\}_{i=1}^N$ with $\mathbf{x}_i \in \mathcal{X} \subseteq \mathbb{R}^p$ and $y_i = \{0, 1\}$. Moreover, suppose the conditional distribution of the class y_i is given by $y_i|\mathbf{x}_i, \theta \sim \text{Bern}(p(x_i) = \text{logit}(f_\theta(\mathbf{x}_i)))$ where θ parameterizes our estimate of probability $p(x_i)$. Under this model, we have (log)-likelihoods as follows.

$$\mathcal{L}(\theta|Y, X) = \prod_{i=1}^{n} p(x_i)^{y_i}(1 - p(x_i))^{1-y_i} \tag{1}$$

$$\ell(\theta|X, Y) = \sum_{i=1}^{n} \left\{ y_i \log\left(\frac{p(x_i)}{1 - p(x_i)}\right) + \log(1 - p(x_i)) \right\} \tag{2}$$

The goal of any method considered here is the accurate estimation of $p(x_i) = \text{logit}^{-1}(f_\theta(\mathbf{x}_i))$. Under this model, our goal reduces to the estimation of the parameters θ that parameterize f_θ. Here, we consider two functions. First we consider

$$f_\theta(\mathbf{x}) = \theta^T \mathbf{x} \quad \theta \in \mathbb{R}^{p \times 1}$$

which provides the Logistic Regression model. In addition, we consider the function

$$f_\theta(\mathbf{x}) = \theta^{(K)} \sigma(\theta^{(K-1)} \sigma(\dots \theta^{(2)} \sigma(\theta^{(1)} \mathbf{x}))) \quad \theta^{(1)} \in \mathbb{R}^{(m \times p)}, \theta^{(j)} \in \mathbb{R}^{m \times m}, \theta^{(K)} \in \mathbb{R}^{1 \times m}$$

which gives a K-layer Artificial Neural Network with activation function $\sigma(\cdot)$, m nodes in each hidden layer, and one output node. Notice that by removing the hidden layers from this Neural Network, setting $m = 1$ and $\sigma(\cdot) = \text{logit}(\cdot)$, we recover the Logistic Regression model. Therefore, we can interpret this Neural Network structure as a direct generalization of the Logistic Regression. We note here that we have chosen to parameterize the function f_θ. Indeed, as commonplace in statistical learning and non-parametric analysis, replacing this function with some other parametric or non parametric form will lead to different, often more complex models (Hastie, Tibshirani, and Friedman 2009). However, here we choose to focus on the two functions f_θ, introduced above as to compare and contrast the Logistic Regression model and the Neural Network model.

Having introduced these models, our focus is now centered around parameter inference. For a holistic approach, we consider both a Frequentist and Bayesian framework to complete this task. In the Frequentist setting, we seek

$$\hat{\theta} = \arg\max_{\theta \in \Theta} \mathcal{L}(\theta|Y, X) \tag{3}$$

However, under both the Logistic Regression model and the Neural Network Model we do not arrive at a closed form solution of $\hat{\theta}$. Therefore, we turn to gradient descent algorithms (e.g. Newton Raphson, Back-Propagation) to arrive at the estimated parameter values (McCullagh and Nelder 1989, Zhang 2000). In the Bayesian setting, we instead seek the posterior distribution $\pi(\theta|\mathbf{X}, \mathbf{Y})$. That is, given a prior distribution on the parameters of the model $\pi(\theta)$, we seek

$$\pi(\theta|\mathbf{X}, \mathbf{Y}) \propto p(\mathbf{X}, \mathbf{Y}|\theta)\pi(\theta) \propto p(\mathbf{Y}|\mathbf{X}, \theta)p(\mathbf{X}|\theta)\pi(\theta) \tag{4}$$

Now, by the definition of the likelihood given above, i.e. $p(\mathbf{Y}|\mathbf{X},\theta) = \mathcal{L}(\theta|\mathbf{Y},\mathbf{X})$, the corresponding posterior distribution does not have a closed form probability distribution associated with it (Tran et al. 2018). Therefore, traditional Gibbs sampling in either the Logistic Regression setting or the Neural Network setting will not suffice. We discuss alternative Markov Chain Monte Carlo (MCMC) sampling techniques (e.g. Metropolis-within-Gibbs Sampling) to infer the full posterior distribution $\pi(\theta|\mathbf{X},\mathbf{Y})$.

In this project, we look to compare two approaches to binary classification under a Frequentist and Bayesian paradigm. In Section 2, we study the Logistic Regression model and consider maximum likelihood estimation and a Bayesian estimation scheme. In Section 3, we study the Neural Network model and consider maximum likelihood estimation and a Bayesian estimation scheme over a much larger parameter space. In Section 4, we compare these methodologies in a theoretical setting. Finally, in Section 5 we complete a data analysis using these four different approaches to binary classification and conclude with a discussion in Section 6.

2 Regression Models

2.1 Logistic Regression

Logistic Regression is a generalized linear model (GLM) for modeling the binary data (McCullagh and Nelder 1989). For binary response $y_i \in \{0,1\}$, y_i follows a Bernoulli distribution with probability $\pi_i = p_i(x_i)$. In order to fit a generalized linear model, one of the assumptions is the distribution of the response belongs to the exponential family. For the binary response:

$$\log \mathbb{P}(\mathbf{Y}) = \sum_{i=1}^{n} \left\{ y_i \log\left(\frac{p(x_i)}{1-p(x_i)}\right) + \log(1-p(x_i)) \right\}$$

Thus, Bernoulli distribution belongs to the exponential family with canonical parameter $\gamma_i = log\frac{\pi_i}{1-\pi_i} = \text{logit}(\pi_i)$, cumulant function $b(\gamma_i) = -log(1-\pi_i)$, and with unit dispersion. Also note that the canonical link is the logistic function, i.e. $g^{-1}(\eta_i) = \frac{1}{1+e^{-\eta_i}}$ where $\eta_i = x_i^T\theta$.

The estimation for θ in the general framework for GLM is as follows. As the logistic regression takes the canonical link for binary data GLM, we assume a canonical link, e.g $\gamma_i = \eta_i = x_i^T\theta$, in the following derivation. Under the assumption that all the data $\{\mathbf{x}_i, y_i\}_{i=1}^N$ are independently and identically distributed, the log-likelihood becomes

$$l(\gamma,\phi;y) = \sum_{i=1}^{n} logf(y_i;\gamma_i,\phi) = \sum_{i=1}^{n} \frac{y_i\gamma_i - b(\gamma_i)}{a(\phi)} + c(y_i,\phi) = \sum_{i=1}^{n} \frac{y_i x_i^T\theta - b(x_i^T\theta)}{\phi} + c(y_i,\phi)$$

The score is then

$$U(\theta) = \frac{\partial l}{\partial \theta} = \frac{1}{\phi}\mathbf{X}^T(\mathbf{Y} - \mu(\theta))$$

Setting the score to zero, we can solve for θ using the Newton Raphson Method. Taking a first order Taylor Expansion of the score, we have

$$U(\tilde{\theta}) \approx U(\theta) + \frac{\partial U(\theta)}{\partial \theta^T}(\tilde{\theta} - \theta) = 0$$

This gives the update formula:

$$\begin{aligned}\theta^{(t+1)} &= \theta^{(t)} - J_U(\theta^{(t)})^{-1}U(\theta^{(t)}) = \theta^{(t)} - (\frac{\partial U}{\partial \theta^T}(\theta^{(t)}))^{-1}U(\theta^{(t)}) \\ &= \theta^{(t)} + (\mathbf{X}^T W(\theta^T)\mathbf{X})^{-1}\mathbf{X}^T W(\theta^{(t)})W(\theta^{(t)})^{-1}(\mathbf{Y} - \mu(\theta^{(t)}))\end{aligned}$$

where $W(\theta^T) = Diag_i\{b''(\mathbf{x_i}^T\theta)\}$ and $J_U(\theta^{(t)})$ is the Jacobian matrix of $\theta^{(t)}$. Simplifying the above equation gives the iterative reweighted least squares (IRLS) updating formula:

$$\mathbf{X^T}W(\theta^{(t)})\mathbf{X}\theta^{(t+1)} = \mathbf{X}W(\theta^{(t)})Z^{(t)}$$

where $Z^{(t)} = \eta^{(t)} + W(\theta^{(t)^{-1}})(\mathbf{Y} - \mu(\theta^{(t)}))$ is called the working response.

2.2 Bayesian Logistic Regression

In Bayesian Logistic Regression, we take prior information about the regression parameters $\theta \in \mathbb{R}^p$ into account, which allows for a more precise estimation (*Bayesian data analysis* 1995, Tran et al. 2018). Given prior distribution $\pi(\theta)$ and data $\mathcal{T} = \{(\mathbf{x}_i, y_i)\}_{i=1}^N$ with joint likelihood $p(\mathbf{Y}, \mathbf{X}|\theta) = p(\mathbf{Y}|\mathbf{X}, \theta)p(\mathbf{X}|\theta)$, the posterior distribution is given by Equation 4.

The outcome y_i follows a Bernoulli distribution with probability $p_i(x_i) = \text{logit}^{-1}(x_i^T\theta)$. We build a hierarchical model where we assume that $\theta \sim \text{MVN}(\mu, \Sigma)$, $\mu \sim \text{MVN}(\mu^*, \sigma^{2^*}I)$ and $\Sigma \sim \text{Inv-Wishart}(\Psi^*, \nu^*)$. These prior distributions are commonly used in practice in Bayesian regression models (Hoff 2009). We also assume that $\mathbb{P}(\mathbf{X}|\theta, \mu, \Sigma) \propto 1$. This mimics the use of an uninformative prior, which is typically used when no prior knowledge is known. Using these formulations, the posterior distribution becomes

$$\begin{aligned}\mathbb{P}(\theta, \mu, \Sigma|\mathbf{Y}, \mathbf{X}) &\propto \mathbb{P}(\mathbf{Y}|\mathbf{X}, \theta, \mu, \Sigma)\mathbb{P}(\mathbf{X}|\theta, \mu, \Sigma)\mathbb{P}(\theta|\mu, \Sigma)\mathbb{P}(\mu)\mathbb{P}(\Sigma) \\ &\propto \left(\prod_{i=1}^n p_i(x_i)^{y_i}(1-p_i(x_i))^{1-y_i}\right)\det(\Sigma)^{-1/2}\exp\left\{-\frac{1}{2}(\theta-\mu)^T\Sigma^{-1}(\theta-\mu)\right\} \times \\ &\quad \left(\frac{1}{\sigma^{2^*}}\right)^{1/2}\exp\left\{-\frac{1}{2\sigma^{2^*}}(\mu-\mu^*)^T(\mu-\mu^*)\right\}\det(\Sigma)^{-\frac{\nu^*+p+1}{2}}\exp\left\{-\frac{1}{2}tr(\Psi^*\Sigma^{-1})\right\}, \quad (5)\end{aligned}$$

where Σ and Ψ^* are $p \times p$ positive definite matrices. Note that this posterior distribution does not have a named distribution associated with it. In order to draw samples from the posterior distribution, we have to derive the full conditional distributions and use Monte Carlo Markov Chain (MCMC) algorithms to approximate the distribution. The full conditional distributions of the parameters μ, Σ, θ_j for $j = 1, \ldots, p$ are derived in Appendix A and are given as

$$\begin{aligned}\mu|\mathbf{X}, \mathbf{Y}, \theta, \Sigma &\sim MVN((\Sigma^{-1} + \frac{1}{\sigma^{2^*}}I)^{-1}(\Sigma^{-1}\theta + \frac{1}{\sigma^{2^*}}\mu^*), (\Sigma^{-1} + \frac{1}{\sigma^{2^*}}I)) \\ \Sigma|\mathbf{X}, \mathbf{Y}, \theta, \mu &\sim \text{Inv-Wishart}((\theta-\mu)(\theta-\mu)^T + \Psi^*, \nu^* + 1) \\ \theta_j|\mathbf{X}, \mathbf{Y}, \mu, \Sigma &\propto \prod_{i=1}^n \text{logit}^{-1}(\mathbf{x}_i^T\theta)^{y_i}(1-\text{logit}^{-1}(\mathbf{x}_i^T\theta))^{1-y_i}\exp\left\{-\frac{1}{2}(\theta-\mu)^T\Sigma^{-1}(\theta-\mu)\right\}\end{aligned}$$

The parameters μ and Σ can be sampled from known distributions, namely from the Multivariate Normal and Inverse Wishart distributions, respectively. However, there is no closed form probability distribution for the full conditional of θ_j. Therefore, we will use a Metropolis-within-Gibbs sampling algorithm to approximate the posterior distribution.

2.2.1 Metropolis-within-Gibbs Algorithm

To sample from the full conditionals, we use a Metropolis-within-Gibbs sampling algorithm since there is no closed form probability distribution for θ_j. In each step $t = 1, \ldots, T$ of the Metropolis-within-Gibbs algorithm, we first sample μ and Σ from the Multivariate Normal and Inverse Wishart distributions, respectively, using a Gibbs sampling step. Then, for $j = 1, \ldots, p$, we sample a proposal value θ_j^* from a Normal distribution centered around the current value of $\theta_j^{(t)}$ and with an adaptive variance $(\eta_j^{(t)})^2$, that depends on the acceptance probability. Let r^* be the target acceptance probability. If the proportion of acceptances is higher than r^*, the variance is increased, and if the proportion of acceptances is lower than r^*, the variance is decreased. The magnitude of the change in variance depends on the magnitude of the difference of the current acceptance probability and the target acceptance probability. We multiply this difference by a learning rate $\gamma_j^{(t)}$ that decreases as the number of iterations increases in order to stabilize the proposal variance.

The current acceptance probability of the j-th parameter is given by

$$\begin{aligned} r_j^{(t)} &= \frac{\pi(\theta_j^* | \theta_1^{(t+1)}, \ldots, \theta_{j-1}^{(t+1)}, \theta_{j+1}^{(t)}, \ldots, \theta_p^{(t)}, \mathbf{X}, \mathbf{y}, \mu, \Sigma)}{\pi(\theta_j^{(t)} | \theta_1^{(t+1)}, \ldots, \theta_{j-1}^{(t+1)}, \theta_{j+1}^{(t)}, \ldots, \theta_p^{(t)}, \mathbf{X}, \mathbf{y}, \mu, \Sigma)} \\ &= \frac{p(\mathbf{Y} | \theta_j^*, \theta_1^{(t+1)}, \ldots, \theta_{j-1}^{(t+1)}, \theta_{j+1}^{(t)}, \ldots, \theta_p^{(t)}, \mathbf{X}, \mu, \Sigma) p(\theta_j^* | \theta_1^{(t+1)}, \ldots, \theta_{j-1}^{(t+1)}, \theta_{j+1}^{(t)}, \ldots, \theta_p^{(t)}, \mu, \Sigma)}{p(\mathbf{Y} | \theta_j^{(t)}, \theta_1^{(t+1)}, \ldots, \theta_{j-1}^{(t+1)}, \theta_{j+1}^{(t)}, \ldots, \theta_p^{(t)}, \mathbf{X}, \mu, \Sigma) p(\theta_j^{(t)} | \theta_1^{(t+1)}, \ldots, \theta_{j-1}^{(t+1)}, \theta_{j+1}^{(t)}, \ldots, \theta_p^{(t)}, \mu, \Sigma)} \end{aligned} \tag{6}$$

We accept the proposed value θ_j^* as the updated value $\theta_j^{(t+1)}$ with probability $\min(r_j^{(t)}, 1)$. If θ_j^* is not accepted, then $\theta_j^{(t+1)}$ remains as $\theta_j^{(t)}$. In practice, the log ratio is used instead, and so we have

$$\begin{aligned} \log r_j^{(t)} = \log \Big(& p(\mathbf{Y} | \theta_j^*, \theta_1^{(t+1)}, \ldots, \theta_{j-1}^{(t+1)}, \theta_{j+1}^{(t)}, \ldots, \theta_p^{(t)}, \mathbf{X}, \mu, \Sigma) \\ & - p(\mathbf{Y} | \theta_j^{(t)}, \theta_1^{(t+1)}, \ldots, \theta_{j-1}^{(t+1)}, \theta_{j+1}^{(t)}, \ldots, \theta_p^{(t)}, \mathbf{X}, \mu, \Sigma) \\ & + p(\theta_j^* | \theta_1^{(t+1)}, \ldots, \theta_{j-1}^{(t+1)}, \theta_{j+1}^{(t)}, \ldots, \theta_p^{(t)}, \mu, \Sigma) \\ & - p(\theta_j^{(t)} | \theta_1^{(t+1)}, \ldots, \theta_{j-1}^{(t+1)}, \theta_{j+1}^{(t)}, \ldots, \theta_p^{(t)}, \mu, \Sigma) \Big) . \end{aligned} \tag{7}$$

Algorithm 1 shows the pseudocode for the Metropolis-within-Gibbs Algorithm.

Algorithm 1 Metropolis-within-Gibbs Algorithm for Finding Posterior Distribution of θ, μ, Σ

Input: data $\{\mathbf{x}_i, y_i\}_{i=1}^N$, initial parameter $\theta^{(0)}, \Sigma^{(0)}, \eta^{(0)}$, learning rates $(\gamma_j^{(0)})_{j=1}^p$, and target acceptance probability r^*

Output: Samples from $p(\theta_j|\mathbf{X}, \mathbf{Y})$ for all $j = 1, \ldots, p$

Create $\mathbf{M}_{p\times T}, \mathbf{TH}_{p\times T}, \mathbf{S}_{p\times p\times T}, \eta_{p\times T}$ to store parameters
for $t = 1, \ldots, T$ **do**
    Sample $\mu^{(t)} \sim MVN(\theta^{(t-1)}, \Sigma^{(t-1)})$
    Sample $\Sigma^{(t)} \sim Inv - Wishart((\theta^{(t-1)} - \mu^{(t)})(\theta^{(t-1)} - \mu^{(t)})^T + \Psi^*, \nu^* + 1)$
    for $j = 1, \ldots, p$ **do**
        Sample $\theta_j^* \sim N(\theta_j^{(t-1)}, (\eta_j{}^{(t-1)})^2)$
        Compute $\log r_j^{(t)}$ given in Equation (7)
        Sample $u \sim Unif(0, 1)$
        if $\log(u) < \log r_j^{(t)}$ **then**
            $\theta_j^{(t)} = \theta_j^*$
        else
            $\theta_j^{(t)} = \theta_j^{(t-1)}$
        end if
        Update $\eta_j^{(t)} = \eta_j^{(t-1)} + \gamma_j^{(t)}(r_j^{(t)} - r^*)$
        $TH[j, t] = \theta_j^{(t)}$
    end for
    $M[, t] = \mu^{(t)}, S[, , t] = \Sigma^{(t)}$
end for

3 Neural Network Models

3.1 Feed Forward Artificial Neural Network

In this work, we will be using the Artificial Neural Network (ANN) which is composed of one input layer, several hidden layers, and one output layer (Schmidhuber 2014, Hastie, Tibshirani, and Friedman 2009, Zhang 2000). Figure 1 shows an example of the framework of an ANN with an input of dimension 3, a single hidden layer with three nodes and the output layer. Nodes in the successive two layers are fully connected, and the information only moves forward in ANN. In addition, an activation function is applied to the outputs of each hidden layer. We will use the sigmoid function as the activation function as a good comparison to the generalized linear models. The sigmoid function is defined as:

$$f(z) = \frac{1}{1 + \exp(-z)}.$$

Suppose $\boldsymbol{h}^{(k)}$ is the output of the k-th layer, then

$$\boldsymbol{h}^{(k)} = f(\boldsymbol{b}^{(k)} + \boldsymbol{W}^{(k)}\boldsymbol{h}^{(k-1)})$$

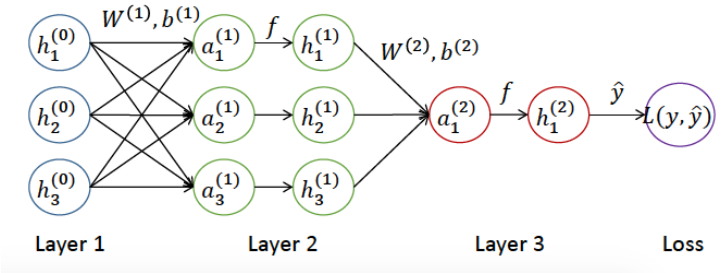


Figure 1: Framework of ANN with one input layer, single hidden layer with three nodes and one output layer. f is the activation function and $W^{(\bullet)}$ and $b^{(\bullet)}$ are coefficients to be estimated.

and $\boldsymbol{h}^{(0)} = \boldsymbol{x}$. For the loss function $J(\Theta)$, minimizing the loss function is equivalent to maximizing the likelihood. Thus,

$$J(\Theta) = -\frac{1}{n}\sum_{i=1}^{n}\left\{y_i \log\left(\frac{h_\Theta(x_i)}{1-h_\Theta(x_i)}\right) + \log(1-h_\Theta(x_i))\right\} = -\frac{1}{n}\ell(Y|X,\Theta) \tag{8}$$

which coincides with the cross-entropy loss. Our goal is to estimate the parameters $\boldsymbol{W}^{(k)}$'s and $\boldsymbol{b}^{(k)}$'s, which we achieve using the back propagation algorithm described in Algorithm 2.

Algorithm 2 Back Propagation Algorithm

1: Conduct the forward computation on the input data
2: Compute the gradient on the output layer: $\mathbf{g} \leftarrow \nabla_{\hat{y}} J$
3: **for** $k = l, l-1, ..., 1$ **do**
4: Convert the gradient on the layer's output into a gradient into the pre-nonlinearity activation: $\mathbf{g} \leftarrow \nabla_{a^{(k)}} J = \mathbf{g} \odot f'(a^{(k)})$
5: Compute gradients on biases and weights: $\nabla_{\boldsymbol{b}^{(k)}} J = \mathbf{g}$
6: $\nabla_{\boldsymbol{W}^{(k)}} J = \mathbf{g}\boldsymbol{h}^{(k-1)T}$
7: Propagate the gradients w.r.t. the next lower-level hidden layer's activations: $\mathbf{g} \leftarrow \nabla_{\boldsymbol{h}^{(k-1)}} J = \boldsymbol{W}^{(k)T}\mathbf{g}$
8: **end for**

3.2 Bayesian Artificial Neural Network

The previous section introduces the ANN model and attempts to find optimal parameters θ by iteratively minimizing the likelihood loss function $J(\Theta)$. In the Bayesian framework, we instead look to incorporate prior information $p(\theta)$ with our likelihood $p(\mathbf{Y}|\mathbf{X},\theta)$ and draw inferences based on the posterior $p(\theta|\mathbf{X},\mathbf{Y})$ (*Bayesian data analysis* 1995, Wan 1990, Tran et al. 2018). Recall under the Bayesian Logistic Regression framework, we consider the model

$$y_i|\mathbf{x}_i,\theta \overset{ind.}{\sim} \text{Bern}\left(\frac{\exp[f_\theta(\mathbf{x}_i)]}{1+\exp[f_\theta(\mathbf{x}_i)]}\right)$$
$$p(\mathbf{X}|\theta) \propto 1$$
$$\theta \sim p(\theta)$$

where the parameter θ is given by $\theta = \{\theta^{(1)} \in \mathbb{R}^{(m\times p)}, \theta^{(j)} \in \mathbb{R}^{m\times m}, \theta^{(K)} \in \mathbb{R}^{1\times m}\}$ and

$$f_\theta(\mathbf{x}) = \theta^{(K)}\sigma(\theta^{(K-1)}\sigma(\dots\theta^{(2)}\sigma(\theta^{(1)}\mathbf{x})))$$

for some activation function $\sigma(\cdot)$. For ease of notation and computational feasibility, we construct a *one-layer* neural network with M hidden nodes. In this case, we can rewrite f_θ and θ as

$$f_\theta(\mathbf{x}) = \sum_{m=1}^{M} w_m^{(2)}\sigma(b_m + (\mathbf{w}_m^{(1)})^T\mathbf{x})$$

for $\theta = \{\mathbf{w}^{(2)}, \mathbf{b} \in \mathbb{R}^M, \mathbf{w}_m^{(1)} \in \mathbb{R}^p\}$ and some activation function σ. For simplicity, we further assume the components of θ are i.i.d. from the distribution $N(0, c^2)$ for some known constant c.

To make inferences regarding the posterior $p(\theta|\mathbf{X}, \mathbf{Y})$, we look to develop a Gibbs sampler for this model. To this end, we need access to the full conditional distributions $p(\theta_j|\theta_{[-j]}, \mathbf{x}, \mathbf{y})$ where $\theta_{[-j]}$ denotes the usual $\theta_{[-j]} = (\theta_1, \dots, \theta_{j-1}, \theta_{j+1}, \dots, \theta_{M(2+p)})$. Under this model, we can write the full posterior up to a constant as follows

$$\begin{aligned}
p(\theta|\mathbf{X}, \mathbf{Y}) &\propto p(\mathbf{Y}|\mathbf{X}, \theta)p(\mathbf{X}|\theta)p(\theta) \\
&= \prod_{i=1}^{n} p(x_i)^{y_i}(1 - p(x_i))^{1-y_i} \prod_{m=1}^{M} p(w_m^{(2)})p(b_m)p(\mathbf{w}_m^{(1)}) \\
&= \prod_{i=1}^{n} p(x_i)^{y_i}(1 - p(x_i))^{1-y_i} \prod_{m=1}^{M} p(w_m^{(2)})p(b_m)p(\mathbf{w}_m^{(1)})
\end{aligned}$$

where $p_i = \exp(f_\theta(x_i))(1 - \exp(f_\theta(x_i)))^{-1}$. Focusing on this likelihood function we have

$$\begin{aligned}
p(\mathbf{Y}|\mathbf{X}, \theta) &= \prod_{i=1}^{n} p(x_i)^{y_i}(1 - p(x_i))^{1-y_i} \\
&= \exp\left\{\sum_{i=1}^{n} y_i \log\left(\frac{p(x_i)}{1 - p(x_i)}\right) + \log(1 - p(x_i))\right\} \\
&= \exp\left\{\sum_{i=1}^{n} y_i f_\theta(\mathbf{x}_i) - \log(1 + \exp(f_\theta(\mathbf{x}_i)))\right\}
\end{aligned}$$

which is the standard Logistic Regression likelihood form. Now turning to the prior distributions

$$\begin{aligned}
p(\theta) &= \prod_{m=1}^{M} p(w_m^{(2)})p(b_m)p(\mathbf{w}_m^{(1)}) \propto \prod_{m=1}^{M} \exp\left(-\frac{1}{2c^2}(w_m^{(2)})^2\right)\exp\left(-\frac{1}{2c^2}b_m^2\right)\exp\left(-\frac{1}{2c^2}(\mathbf{w}_m^{(1)})^T\mathbf{w}_m^{(1)}\right) \\
&= \exp\left(-\frac{1}{2c^2}\|\theta\|_2^2\right)
\end{aligned}$$

Therefore the full posterior distribution can be written up to a constant as

$$p(\theta|\mathbf{X}, \mathbf{Y}) \propto \exp\left\{-\frac{1}{2c^2}\|\theta\|_2^2 + \sum_{i=1}^{n} y_i f_\theta(\mathbf{x}_i) - \log\left(1 + \exp(f_\theta(\mathbf{x}_i))\right)\right\}. \tag{9}$$

Clearly, the distribution is not associated with a named probability distribution. Therefore we turn to Metropolis-within-Gibbs sampling to update the parameters θ_j. In order to do so, we need access to the full

conditional distributions. As the elements of θ are assumed to be independent, the conditional distribution of these parameters can be written as

$$p(\theta_j|\theta_{[-j]}, \mathbf{X}, \mathbf{Y}) \propto \exp\left\{-\frac{1}{2c^2}\theta_j^2 + \sum_{i=1}^{n} y_i f_\theta(\mathbf{x}_i) - \log\left(1 + \exp(f_\theta(\mathbf{x}_i))\right)\right\} \tag{10}$$

Clearly, this distribution is not simple to sample from. As an alternative, we turn to Metropolis-Hastings to sample from (10) within our Gibbs procedure. Therefore, at each stage of our Gibbs procedure, we produce an adaptive Gaussian Random Walk Metropolis-Hasting proposal with an accept-reject rule based on the density given in (10). The adaptive portion of this algorithm will come by adaptively tuning the variance of the random walk process that defines our proposals. This algorithm is formalized in Algorithm 3.

Algorithm 3 One-Layer NN: Metropolis-within-Gibbs Sampler

Input: data $\{\mathbf{x}_i, y_i\}_{i=1}^N$, initial parameter values $\theta^{(0)} \in \mathbb{R}^{M(2+p)}$, initial step sizes $\nu^{(0)} = [\nu_j^{(0)}]_{j=1}^{M(2+p)}$, learning rates $(\gamma_j^{(0)})_{j=1}^{M(2+p)}$, and desired acceptance probability r

Output: Samples from $p(\theta|\mathbf{X}, \mathbf{Y})$ for all

for $t = 0, 1, 2, \ldots$ **do**
  for $j \in [M(2+p)]$ **do**
    Sample proposal $\theta_j^* \sim N(\theta_j^{(t)}, (\nu_j^2)^{(t)})$
    Calculate acceptance probability $\alpha_{tj} = \min\left\{1, \frac{p\left(\theta_j^*|\theta_{[-j]}^{(t)}, \mathbf{y}\right)}{p\left(\theta_j^{(t)}|\theta_{[-j]}^{(t)}, \mathbf{y}\right)}\right\}$ ▷ Here $p(\cdot|\theta_{[-j]}, \mathbf{y})$ is from (10)
    Set $\theta_j^{(t+1)} = \begin{cases} \theta_j^* & \text{with probability} \quad \alpha_{tj} \\ \theta_j^{(t)} & \text{with probability} \quad 1 - \alpha_{tj} \end{cases}$
    Update $\nu_j^{(t+1)} = \nu_j^{(t)} + \gamma_j^{(t)}(\alpha_{tj} - r)$ ▷ Adaptive step size update
  end for
  if Convergence **then**
    break
  end if
end for

In this algorithm, we adaptively tune the variance of the random walk for *each* parameter θ_j. In that way, we do not impose that different components receive proposals from a common random walk process. Instead, we allow the random walk to explore and adjust in each parameter space independently and rely on our acceptance probability to inform whether the region is a high density area of the posterior. We implement this procedure in our data analysis in Section 5.

4 Methodological Comparison

The models studied in this paper are all likelihood based methods in the sense that our central focus is on modeling the parameter of this distribution. While these methods all address a unified task, there are several important differences among them. The Frequentist models use gradient based methods to find optimal

parameters to maximize the likelihood (McCullagh and Nelder 1989, Hastie, Tibshirani, and Friedman 2009), while the Bayesian models approximate the posterior distribution of the parameters using MCMC algorithms (*Bayesian data analysis* 1995, Wan 1990). Bayesian methods also allow for the incorporation of prior information which may prevent overfitting problems and have fewer assumptions than Frequentist methods. However, the accuracy of the posterior distribution in Bayesian methods depends on the convergence of the Markov Chain, which is not necessarily guaranteed in finite time. Even if the Markov Chain does converge, it may take many iterations until it happens.

The statistical Logistic Regression models and Neural Network models are also quite different. The former model assumes linearity on the predictor variables. In order to handle non-linearities or complex relationships of the predictor variables, we may need to manually construct non-linear transformations of the predictor variables. On the other hand, Neural Network models are equipped in learning non-linearities and projecting the original features to different feature spaces, which means they can be applied to more general cases. However, there may be a greater chance of overfitting in Neural Networks since the algorithm depends heavily on the training data. In essence, this feature of actively constructing covariates may be a limitation if the training data does not reflect the true process we are trying to understand. In short, the accuracy of predictions for Neural Networks depend on the size and quality of the given training set. While Logistic Regression methods also depend on the size and quality of the training set, there is a smaller dependency, and so they do not require as many training data. However, the overfitting issue can be eased by using techniques like introducing a regularization term into the loss function and doing cross-validation to tune parameters.

In addition to these differences, there is also the consideration of convexity of the likelihood function. Under the Logistic Regression model, we are guaranteed that the likelihood function is convex and therefore has a guaranteed global maximum (McCullagh and Nelder 1989). Alternatively, the Neural Network model has no such guarantee (Schmidhuber 2014). For this reason, these methods are quite sensitive to initial values of the gradient descent algorithm. This problem also appears in the Bayesian paradigm. Under the Metropolis-within-Gibbs MCMC setting, we require multiple evaluations of the likelihood function at each iteration of the algorithm. As the likelihood function is quite simple for the Logistic Regression setting, computing this efficiently is attainable. For the Neural Network, this task is much more complex. Therefore, the features of the likelihood function under the Logistic Regression and Neural Network models impact inference in both the Frequentist and Bayesian settings.

The previous three points are mostly technical in nature. While these points are critical to the understanding the differences under each approach considered in this report, there is a more central difference of the Logistic Regression and Neural Network models. The interpretation of relationships between predictor variables and the response in Neural Networks can be near impossible, while in the Logistic Regression setting, it is quite simple. As Neural Network models performance improves for deeper networks, this interpretability issue only compounds. This should not be surprising as the Neural Network model attempts to construct new features. As a practitioner, it is often quite important to understand which combination of features play a role in explaining the behavior of the response variable. Therefore, even if the Neural Network outperforms the Logistic Regression model, it may not be feasible in practice because it does not clearly address how a set of features affect a response variable.

5 Data Analysis

5.1 Exploratory Data Analysis

To compare the different models discussed in this paper, we analyze a resume skills dataset that was artificially generated based on a case study by the Princeton Dialogues on Artificial Intelligence and Ethics (Princeton 2017) in order to predict whether an applicant will get an interview. We begin our analysis of this dataset by completing Exploratory Data Analysis (EDA) on the development dataset. This dataset contains 619 artificially generated resumes of which 219 resulted in interviews along with 218 binary features describing if a skill appeared on their resume. This data is summarized in Figure 2. The y-axis of this plot are the binary features while the x-axis is the resume index. The corresponding coordinate is shaded if that resume (x) had that skill (y). We sorted the resumes so that all resumes to the left of the red line received an interview and all those to the right did not receive an interview. While the skills present on resumes that

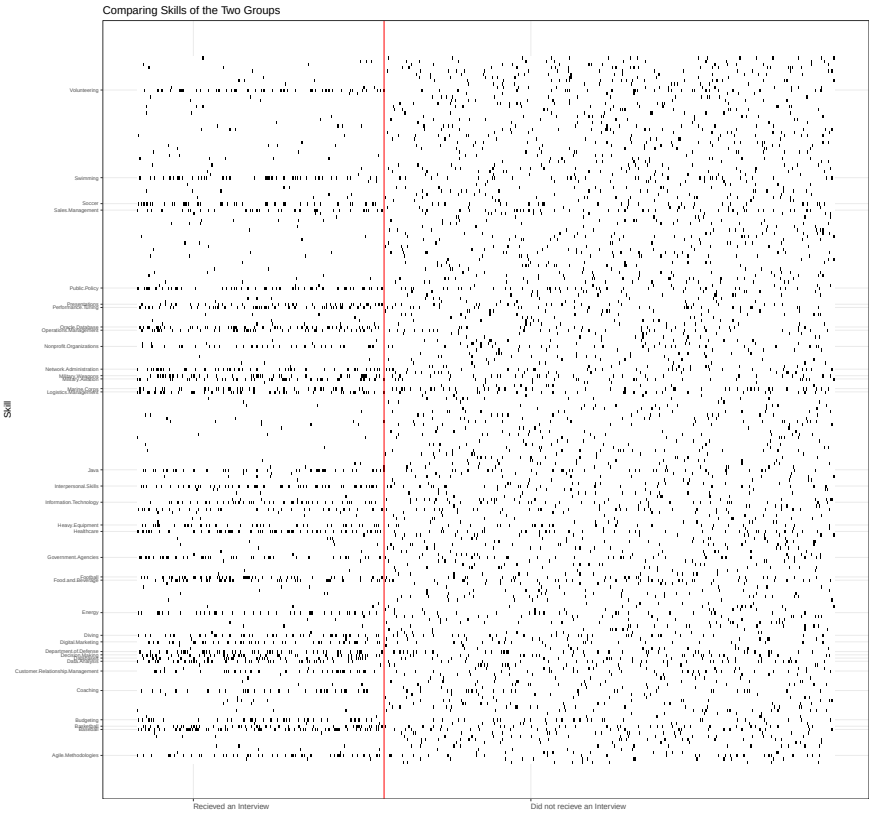


Figure 2: Plotting the presence of a skill on resume against if they received an interview.

did not receive an interview appear random, it is clear that resumes that did receive an interview have a certain set of skills and, perhaps importantly, do *not* have certain skills listed.

To investigate this trend more closely, we considered the number of times a certain skill appeared on a resume that received an interview. We then used a Kernel Density Estimator (KDE) to understand this distribution. This estimator is plotted in Figure 3.

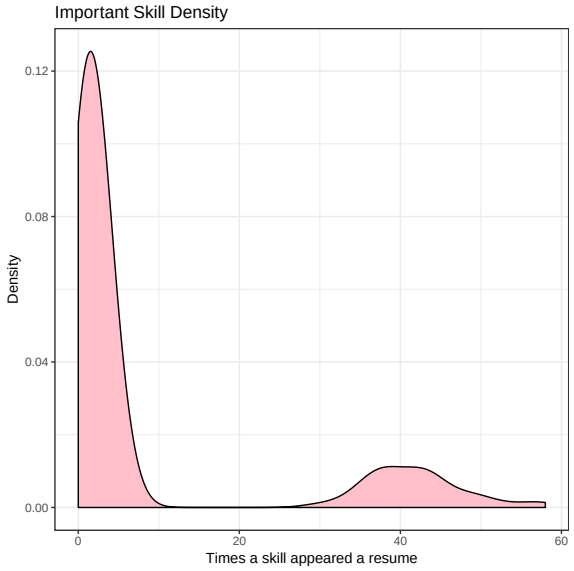


Figure 3: The KDE of the number of times a skill appeared on a successful resume.

Clearly, this distribution is bimodal. Moreover, it appears that skills are partitioned into two groups; those that often appear on resumes that received an interview and those that did not. The skills that often appear (appeared on more than 30 resumes) on these resumes we call *important skills* and are labeled in Figure 2.

To more closely see the difference between resumes that received an interview and those that did not, with respect to these important skills, we recreate Figure 2 by filtering out unimportant skills. This can be found in Figure 4. In this figure, it is evident that the resumes that received an interview were more likely to have one of these important skills on their resume as the blue shaded region contains more shaded squares than non-shaded squares. Having identified the most important skills, we further found the top 10 skills that appeared most frequently on resumes that received an interview. These 10 skills, along with the proportion of resumes that received an interview with said skill are given in Table 1.

Skill	Proportion
Baseball	0.2648402
Healthcare	0.2557078
Interpersonal.Skills	0.2374429
Oracle.Database	0.2283105
Public.Policy	0.2237443
Performance.Tuning	0.2191781
Volunteering	0.2191781
Football	0.2054795
Logistics.Management	0.2054795
Decision.Making	0.2009132

Table 1: The top 10 skills that appeared on resumes that received an interview.

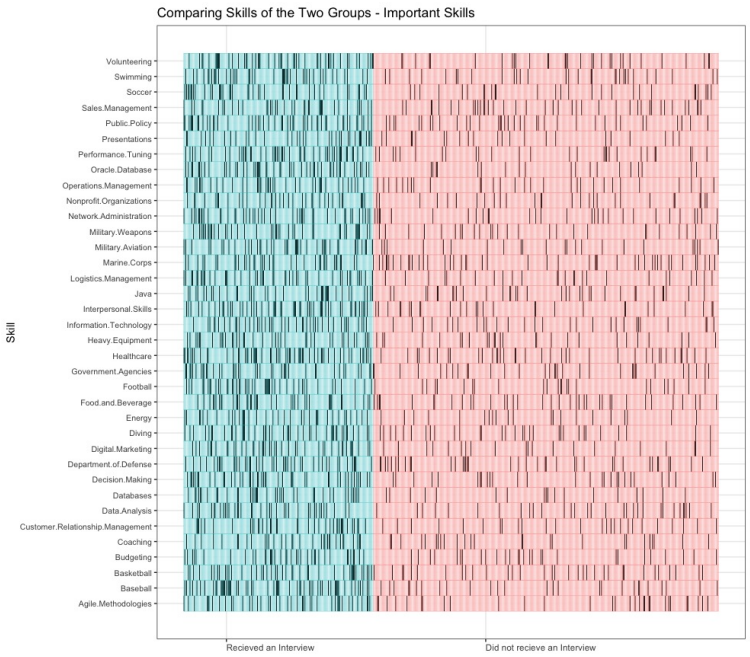


Figure 4: Plotting if an *important* skill appeared on a resume against if the applicant received an interview.

First, we notice that each skill has appeared on at least one out of five resumes that received an interview. It appears that the most important skills contain a mix of hard skills (e.g. Oracle Database, Logistics Management), communication skills (e.g. Interpersonal Skills, Volunteering, Decision Making), and domain specific skills (e.g. Healthcare, Public Policy). We were surprised, however, that both Baseball and Football appeared among the most important skills. This trend could be due to the fact that expressing an interest in hobbies in a resume may be something recruiters are seeking. Moreover, if the resume was submitted for a position involving either of these skills (e.g. Physical Education Instructor) then these skills are indeed quite important.

Upon further investigation, we learned that the Princeton case study (Princeton 2017) examined the machine learning algorithm PARiS that was developed by the company Strategeion to evaluate resumes. The case study follows an applicant who seemed like an ideal candidate, but was eventually rejected by PARiS. In an attempt to understand why the applicant was rejected, the hiring managers found that PARiS favored applicants who list sports on their resumes. They reasoned that many of Strategeion's employees are ex-veterans, and PARiS found a high correlation between ex-veterans and those who list sports on their resumes. This led to PARiS putting more weight on sports than on other skills. This dataset illustrates the importance of studying the different models and their implications. On the one hand, Neural Networks may be applied to more general settings, but on the other hand, it is much easier to interpret results from Logistic Regression models and notice abnormalities. Perhaps with a careful EDA and Logistic Regression model, the hiring managers would have been able to identify the strong relationship with sports and current employees at the beginning and taken it into account.

Having identified a set of possible important features and in an attempt to keep run time of the Bayesian methods reasonable, we choose to only consider these 36 important features. We can treat these EDA results

as prior knowledge which we take into account in the Bayesian models. Meanwhile, the run time of the Frequentist methods will still be feasible for the full feature set, and so we use all skills in the development of these models. However, it is important to note that by using a different set of skills in the models, the analysis using the Bayesian and Frequentist models answer slightly different questions. The former determines the top skills among those that are important in getting an interview, while the latter identifies which skills can lead to an interview. Although we cannot directly compare the two classes of models, we can still compare the Logistic Regression models and Neural Network models within the Frequentist and Bayesian classes. This difference reflects the advantages and disadvantages of the Bayesian and Frequentist methods, but more importantly, it motivates the joint use of these models in inference, as explored in Calibrated Bayesian methods (Little 2006).

5.2 Evaluation Metrics

For a binary classification problem, each prediction falls into one of four possible outcomes: true positives (TP), true negatives (TN), false positives (FP) and false negatives (FN). The results of classification in terms of these four possibilities can be presented by the confusion matrix in Table 2.

		Actual Values	
		Positive (1)	Negative (0)
Predicted Values	Positive (1)	TP	FP
	Negative (0)	FN	TN

Table 2: Confusion Matrix for Classification

In this report, class 1 means the applicant received an interview, while class 0 means the applicant did not receive an interview. To compare the classification performances of the different methods, we use accuracy, precision, recall and F1-measure as evaluation metrics which are defined as:

$$
\begin{aligned}
accuracy &= \frac{TP + TN}{TP + TN + FP + FN} \\
precision &= \frac{TP}{TP + FP} \\
recall &= \frac{TP}{TP + FN} \\
F1 - measure &= 2 * \frac{precision * recall}{precision + recall}
\end{aligned}
$$

These metrics evaluate the performance of classification from different aspects. Accuracy measures the performance in total, while recall focuses on the Type I error. F1-measure averages the effects of precision and recall, and we prefer a method with a higher F1-measure value.

5.3 Model Results

To test the models, we use the pilot dataset which consists of 1,986 labeled testing points. This corresponds to a $25\% - 75\%$ training - testing split. We train each of our models on the 25% training data and then test on the 75% testing data. The implementation details of the methods discussed in Section 3 are given in Appendix B. The results of each method are included in Table 3.

Classification Method	Accuracy	Precision	Recall	F1-measure	Run-Time
Logistic Regression	93.40%	88.51%	91.17%	0.8982	0.021s
Bayesian Logistic Regression	49.55%	36.39%	50.00%	0.4212	269.560s
Artificial Neural Networks	95.82%	93.52%	93.38%	0.9345	0.156s
Bayesian Artificial Neural Networks	69.33%	58.62 %	13.45%	0.2183	$>$ 24 hrs.

Table 3: Comparison between different classification methods.

As evident by the table, it is clear that Logistic Regression and Neural Networks under the Frequentist paradigm outperformed their Bayesian counterparts. Indeed when comparing accuracy of the methods, Logistic Regression and Artificial Neural Networks correctly predicted if a candidate received an interview at 93% and 96%, respectively while Bayesian Logistic Regression and Bayesian Neural network models only classified at a rate of 50% and 70%, respectively. This could be to a number of reasons including improper prior specification, choice of hyperparameters, or due to the choice of only including 36 of the original 218 features. Moreover, it appears that the Bayesian methods on average took significantly longer to run than these Freqeuntist methods. This is due to the fact that under the Bayesian schemes presented here, we require a specialized MCMC sampling technique that can be quite slow under certain models.

Comparing Logistic Regression and Neural Network, it appears that the Neural Network model outperforms the Logistic Regression model in both the Frequentist and Bayesian approach. In the Frequentist approach, the Neural Network offers a 3% improvement in accuracy, a 5% improvement in precision, and a 2% improvement in Recall. Moreover, we see the F1-measure score improves from .89 to .93. In short, it appears that the Neural Network is better at predicting both resumes that received an interview and those that did not when compared to the Logistic Regression model. This, of course, is at the cost of interpretability. In several cases, a practitioner may prefer a 93% accurate model that they can interpret over a 96% accurate model that they cannot interpret. In the Bayesian setting, it appears that the Bayesian Logistic Model preforms no better than random chance, while the Bayesian Neural Network Model predicts if a resume receives an interview at 70% accuracy. Moreover, the Bayesian Neural Network improves precision by 22%, but has a significantly worse recall at 13%. These metrics can be explained by the fact that the Bayesian Neural Network model classified to group 0 (did not receive an interview) 93% of the time while the test and training data only had approximately 70% of applications that did not receive an interview. Therefore, it is unclear which method, the Bayesian Logistic Regression or the Bayesian Neural Network, preformed better in this application.

In this application, the Frequentist methods outperformed the Bayesian methods with respect to model accuracy and to run time. Moreover, it appears that the Nueral Network model gave moderate improvements

over the Logistic Regression model, while sacrificing model interpretability. By further tuning the hyperparameters, priors, and including the full feature set in the Bayesian approaches, we may see an improvement over the models presented here. This will only worsen the run time of these already quite slow algorithms but may offer more comparable prediction performance of the Frequentist methods.

6 Conclusion

In this report, we look at the Logistic Regression and Neural Networks from the Frequentist and Bayesian perspectives. The Logistic Regression is a classical classification model in Statistics and the Neural Networks is one of the most popular modern machine learning models. Under the settings for each classification model, our goal is to estimate the parameters for predictors that minimizes the likelihood-based loss function. From the Frequentist perspective, we resort to variants of the gradient descent algorithms, more specifically, the Newton Raphson method for Logistic Regression and Back Propagation for the Neural Networks. From the Bayesian perspective, Metropolis-within-Gibbs Sampling technique is used to infer the full posterior distribution. To conduct an empirical analysis, we implement all the models and apply them on a resume dataset to predict whether an applicant receives an interview, given the skills on their resumes. Although the results imply that ANN outperforms the other methods discussed, it does so at a cost of interpretability. For future work, we can further explore other choices of prior distributions for the Bayesian methods. We can also extend the discussion to continuous response variables, as we only compared Logistic Regression and Neural Networks from the Frequentist and Bayesian perspectives in the context of binary classification problem in this report. Although there are many benefits and costs of both Logistic Regression and Neural Networks, as well as Frequentist and Bayesian methods, perhaps there is way to jointly use all the models to obtain accurate results with easy interpretation.

7 Contributions

In this project, our work is composed of the following parts: literature review, theoretical derivations of different classification models, exploratory data analysis, experiment using four models, and the analysis of the result. Literature review and theoretical derivations are done by all members, while Zihuan mainly focused on the Logistic Regression model and Artificial Neural Networks, Kelly focused on the Bayesian Logistic Regression model and Ben focused on the Bayesian Neural Networks. For the data analysis section, Ben made contributions to the exploratory data analysis, experiments on the Logistic Regression and Bayesian Neural Networks, Kelly contributed to the Bayesian Logistic Regression and Zihuan contributed to the Logistic Regression and the Neural Networks. All team members did the model comparison theoretically and empirically together.

References

Bayesian data analysis (1995). eng. Texts in statistical science. London ; New York: Chapman & Hall. ISBN: 0412039915.

Hastie, Trevor, Robert Tibshirani, and Jerome H. Friedman (2009). *The elements of statistical learning: data mining, inference, and prediction, 2nd Edition*. Springer series in statistics. Springer. ISBN: 9780387848570. URL: http://www.worldcat.org/oclc/300478243.

Hoff, Peter D (2009). *A first course in Bayesian statistical methods*. Vol. 580. Springer.

Little, Roderick J (2006). "Calibrated Bayes: a Bayes/frequentist roadmap". In: *The American Statistician* 60.3, pp. 213–223.

McCullagh, P. and J.A. Nelder (1989). *Generalized Linear Models, Second Edition*. Chapman and Hall/CRC Monographs on Statistics and Applied Probability Series. Chapman & Hall. ISBN: 9780412317606. URL: http://books.google.com/books?id=h9kFH2%5C_FfBkC.

Princeton (2017). *Princeton case study: Hiring by machine*. https://aiethics.princeton.edu/wp-content/uploads/sites/587/2018/12/Princeton-AI-Ethics-Case-Study-5.pdf.

Schmidhuber, Jürgen (2014). "Deep Learning in Neural Networks: An Overview". In: *CoRR* abs/1404.7828. arXiv: 1404.7828. URL: http://arxiv.org/abs/1404.7828.

Tran, Minh-Ngoc et al. (2018). "Bayesian Deep Net GLM and GLMM". In: *arXiv e-prints*, arXiv:1805.10157, arXiv:1805.10157. arXiv: 1805.10157 [stat.CO].

Wan, E. A. (1990). "Neural network classification: a Bayesian interpretation". In: *IEEE Transactions on Neural Networks* 1.4, pp. 303–305. ISSN: 1045-9227. DOI: 10.1109/72.80269.

Zhang, G. P. (2000). "Neural networks for classification: a survey". In: *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 30.4, pp. 451–462. ISSN: 1094-6977. DOI: 10.1109/5326.897072.

Appendix A Derivations of the Full Conditionals for Logistic Regression

To derive the full conditionals, we aim to identify known distributions using parts of the posterior distribution that depend on the parameter of interest.

i. Derivation of full conditional for μ

From the posterior distribution, we have

$$\begin{aligned}
\mu|\mathbf{X},\mathbf{Y},\theta,\Sigma &\propto \exp\left\{-\frac{1}{2}(\theta-\mu)^T\Sigma^{-1}(\theta-\mu)\right\} exp\left\{-\frac{1}{2\sigma^{2^*}}(\mu-\mu^*)^T(\mu-\mu^*)\right\} \\
&\propto \exp\left\{-\frac{1}{2}\left[(\theta-\mu)^T\Sigma^{-1}(\theta-\mu)+\frac{1}{\sigma^{2^*}}(\mu-\mu^*)^T(\mu-\mu^*)\right]\right\} \\
&\propto \exp\left\{-\frac{1}{2}\left[\theta^T\Sigma^{-1}\theta-2\mu^T\Sigma^{-1}\theta+\mu^T\Sigma^{-1}\mu+\frac{1}{\sigma^{2^*}}\left(\mu^T\mu-2\mu^T\mu^*+\mu^{*T}\mu^*\right)\right]\right\} \\
&\propto \exp\left\{-\frac{1}{2}\left[\mu^T\Sigma^{-1}\mu-2\mu^T\Sigma^{-1}\theta+\frac{1}{\sigma^{2^*}}\left(\mu^T\mu-2\mu^T\mu^*\right)\right]\right\} \\
&\propto \exp\left\{-\frac{1}{2}\left[\mu^T\left(\Sigma^{-1}+\frac{1}{\sigma^{2^*}}I\right)\mu-2\mu^T\left(\Sigma^{-1}\theta+\frac{1}{\sigma^{2^*}}\mu^*\right)\right]\right\} \\
&\propto \exp\left\{-\frac{1}{2}\left[\mu^T\underbrace{\left(\Sigma^{-1}+\frac{1}{\sigma^{2^*}}I\right)}_{=\tilde{\Sigma}^{-1}}\mu-2\mu^T\left(\Sigma^{-1}+\frac{1}{\sigma^{2^*}}I\right)\underbrace{\left(\Sigma^{-1}+\frac{1}{\sigma^{2^*}}I\right)^{-1}\left(\Sigma^{-1}\theta+\frac{1}{\sigma^{2^*}}\mu^*\right)}_{=\tilde{\mu}^*}\right]\right\} \\
&\propto \exp\left\{-\frac{1}{2}\left[\mu^T\tilde{\Sigma}^{-1}\mu-2\mu^T\tilde{\Sigma}^{-1}\tilde{\mu}^*+\tilde{\mu}^{*T}\tilde{\Sigma}^{-1}\tilde{\mu}^*-\tilde{\mu}^{*T}\tilde{\Sigma}^{-1}\tilde{\mu}^*\right]\right\} \\
&\propto \exp\left\{-\frac{1}{2}\left[\mu^T\tilde{\Sigma}^{-1}\mu-2\mu^T\tilde{\Sigma}^{-1}\tilde{\mu}^*+\tilde{\mu}^{*T}\tilde{\Sigma}^{-1}\tilde{\mu}^*\right]\right\} \\
&\propto \exp\left\{-\frac{1}{2}(\mu-\tilde{\mu}^*)^T\tilde{\Sigma}^{-1}(\mu-\tilde{\mu}^*)\right\}.
\end{aligned}$$

Thus, we see that $\mu \sim MVN(\tilde{\mu}^*, \tilde{\Sigma})$ where $\tilde{\mu}^* = \left(\Sigma^{-1}+\frac{1}{\sigma^{2^*}}I\right)^{-1}\left(\Sigma^{-1}\theta+\frac{1}{\sigma^{2^*}}\mu^*\right)$ and $\tilde{\Sigma} = \left(\Sigma^{-1}+\frac{1}{\sigma^{2^*}}I\right)^{-1}$.

ii. Derivation of full conditional for Σ

The Inverse Wishart Distribution is given by

$$\begin{aligned}
f(\mathbf{X}|\Psi,\nu) &= \frac{|\Psi|^{\nu/2}}{2^{\nu p/2}\pi^{\frac{\binom{p}{2}}{2}}\prod_{j=1}^{p}\Gamma(\frac{\nu+1-j}{2})}|\mathbf{X}|^{-(\nu+p+1)/2}\times\exp\left\{-\operatorname{tr}\left(\Psi\mathbf{X}^{-1}\right)/2\right\} \\
&\propto |\mathbf{X}|^{-(\nu+p+1)/2}\times\exp\left\{-\operatorname{tr}\left(\Psi\mathbf{X}^{-1}\right)/2\right\}
\end{aligned}$$

where the parameters are ν, the degrees of freedom, and Ψ, the scale matrix. Here, $\mathbf{X}$ and $\mathbf{S}$ are $p\times p$ positive definite matrices. Looking at the joint posterior distribution, we can derive the conditional distribution of $\Sigma|\mathbf{X},\mathbf{Y},\theta,\mu$:

$$\begin{aligned}
\Sigma|\mathbf{X},\mathbf{Y},\theta,\mu &\propto \det(\Sigma)^{-1/2}\exp\left\{-\frac{1}{2}(\theta-\mu)^T\Sigma^{-1}(\theta-\mu)\right\}\det(\Sigma)^{-\frac{\nu^*+p+1}{2}}\exp\left\{-\frac{1}{2}tr(\Psi^*\Sigma^{-1})\right\} \\
&\propto \det(\Sigma)^{-\frac{1}{2}(1+\nu^*+p+1)}\exp\left\{-\frac{1}{2}\left[(\theta-\mu)^T\Sigma^{-1}(\theta-\mu)+tr(\Psi^*\Sigma^{-1})\right]\right\}
\end{aligned}$$

Let a_{ij} be the element in the i-th row, j-th column of Σ^{-1}. We will now focus on the first term in the

exponential.

$$
\begin{aligned}
(\theta-\mu)^T\Sigma^{-1}(\theta-\mu) &= \begin{bmatrix}\theta_1-\mu_1 & \dots & \theta_p-\mu_p\end{bmatrix}\begin{bmatrix}a_{11} & \dots & a_{1p}\\ a_{21} & \dots & a_{2p}\\ \dots & \dots & \dots\\ a_{p1} & \dots & a_{pp}\end{bmatrix}\begin{bmatrix}\theta_1-\mu_1\\ \theta_2-\mu_2\\ \dots\\ \theta_p-\mu_p\end{bmatrix}\\
&= \begin{bmatrix}\sum_{j=1}^p(\theta_j-\mu_j)a_{j1} & \dots & \sum_{j=1}^p(\theta_j-\mu_j)a_{jp}\end{bmatrix}\begin{bmatrix}\theta_1-\mu_1\\ \theta_2-\mu_2\\ \dots\\ \theta_p-\mu_p\end{bmatrix} + tr(\Psi^*\Sigma^{-1})\\
&= \sum_{i=1}^p\sum_{j=1}^p(\theta_j-\mu_j)a_{ji}(\theta_i-\mu_i)
\end{aligned}
$$

Now, consider $\mathbf{S}=(\theta-\mu)(\theta-\mu)^T$. We then have

$$
\begin{aligned}
\mathbf{S}\Sigma^{-1} &= (\theta-\mu)(\theta-\mu)^T\Sigma^{-1}\\
&= \begin{bmatrix}\theta_1-\mu_1\\ \theta_2-\mu_2\\ \dots\\ \theta_n-\mu_n\end{bmatrix}\begin{bmatrix}\theta_1-\mu_1 & \dots & \theta_n-\mu_n\end{bmatrix}\begin{bmatrix}a_{11} & \dots & a_{1n}\\ a_{21} & \dots & a_{2n}\\ \dots & \dots & \dots\\ a_{n1} & \dots & a_{nn}\end{bmatrix}\\
&= \begin{bmatrix}(\theta_1-\mu_1)^2 & \dots & (\theta_1-\mu_1)(\theta_n-\mu_n)\\ (\theta_2-\mu_2)(\theta_1-\mu_1) & \dots & (\theta_2-\mu_2)(\theta_n-\mu_n)\\ \dots & & \\ (\theta_n-\mu_n)(\theta_1-\mu_1) & \dots & (\theta_n-\mu_n)^2\end{bmatrix}\begin{bmatrix}a_{11} & \dots & a_{1n}\\ a_{21} & \dots & a_{2n}\\ \dots & \dots & \dots\\ a_{n1} & \dots & a_{nn}\end{bmatrix}\\
&= \begin{bmatrix}(\theta_1-\mu_1)\sum_{j=1}^n(\theta_j-\mu_j)a_{j1} & \dots & (\theta_1-\mu_1)\sum_{j=1}^n(\theta_j-\mu_j)a_{jn}\\ (\theta_2-\mu_2)\sum_{j=1}^n(\theta_j-\mu_j)a_{j1} & \dots & (\theta_2-\mu_2)\sum_{j=1}^n(\theta_j-\mu_j)a_{jn}\\ \dots & & \\ (\theta_n-\mu_n)\sum_{j=1}^n(\theta_j-\mu_j)a_{j1} & \dots & (\theta_n-\mu_n)\sum_{j=1}^n(\theta_j-\mu_j)a_{jn}\end{bmatrix}.
\end{aligned}
$$

Taking the trace of this matrix, we have that

$$
tr(\mathbf{S}\Sigma^{-1}) = \sum_{i=1}^n\sum_{j=1}^n(\theta_j-\mu_j)a_{ji}(\theta_i-\mu_i) = (\theta-\mu)^T\Sigma^{-1}(\theta-\mu).
$$

We can then rewrite the full conditional of Σ as

$$
\begin{aligned}
\Sigma|\mathbf{X},\mathbf{Y},\theta,\mu &\propto \det(\Sigma)^{-\frac{1}{2}(1+\nu^*+p+1)}\exp\left\{-\frac{1}{2}\left[(\theta-\mu)^T\Sigma^{-1}(\theta-\mu)+tr(\Psi^*\Sigma^{-1})\right]\right\}\\
&\propto \det(\Sigma)^{-\frac{1}{2}(1+\nu^*+p+1)}\exp\left\{-\frac{1}{2}\left[tr(\mathbf{S}\Sigma^{-1})+tr(\Psi^*\Sigma^{-1})\right]\right\}\\
&\propto \det(\Sigma)^{-\frac{1}{2}(1+\nu^*+p+1)}\exp\left\{-\frac{1}{2}\left[tr(\mathbf{S}\Sigma^{-1}+\Psi^*\Sigma^{-1})\right]\right\}\\
&\propto \det(\Sigma)^{-\frac{1}{2}(1+\nu^*+p+1)}\exp\left\{-\frac{1}{2}\left[tr\left((\mathbf{S}+\Psi^*)\Sigma^{-1}\right)\right]\right\}
\end{aligned}
$$

Hence, we have $\Sigma|\mathbf{X},\mathbf{Y},\theta,\mu \sim Inv-Wishart(\tilde{\Psi}^*,\tilde{\nu}^*)$ where $\tilde{\Psi}^*=(\theta-\mu)(\theta-\mu)^T+\Psi^*$ and $\tilde{\nu}^*=\nu^*+1$.

iii. Derivation of full conditional for θ_j

Lastly, the full conditional for θ_j is given by

$$\theta_j|\mathbf{X},\mathbf{Y},\mu,\Sigma \propto \prod_{i=1}^{n} \text{logit}^{-1}(\mathbf{x}_i^T\theta)^{y_i}(1-\text{logit}^{-1}(\mathbf{x}_i^T\theta))^{1-y_i} \exp\left\{-\frac{1}{2}(\theta-\mu)^T\Sigma^{-1}(\theta-\mu)\right\}$$

There is no known closed form probability distribution with this form.

Appendix B Implementation Details

B.1 Logistic Regression

The most simple model we considered was the classical Logistic Regression. Being one of the most popular approaches to binary classification, several fast implementations exist on numerous programming platforms. We use the *glm* function in base R for this implementation.

B.2 Bayesian Logistic Regression

To find the posterior distribution of the regression coefficients, we first specify the hyperparameters for the distributions of μ and Σ. We let $\mu \sim MVN(\mathbf{0}_{p\times 1}, I)$ and $\Sigma \sim Inv-Wishart(I,p)$ where I is the identity matrix of dimension p, and p is the number of parameters. These distributions were chosen as they are the standard Normal and Inverse-Wishart distributions. We also set the learning rate $\gamma_j^{(t)} = \gamma^{(t)} = \frac{1}{t^2}$, which ensures that the variance for the proposal values θ_j^* stabilizes fairly quickly. The values $\Sigma^{(0)}$, $\theta^{(0)}$ are initialized using the regression output from a Logistic Regression model, while $\eta^{(0)}$ is initialized as $\mathbf{1}_{p\times 1}$. Lastly, we set the target acceptance probability $r^* = 0.3$.

We run the Metropolis-within-Gibbs algorithm for 10,000 iterations to obtain an approximation of the posterior density. To account for initial bias and autocorrelation, we take a burn in period of 1,000 iterations and take every 15-th iteration. Appendix C.1 show the plots of the Markov Chain path and the autocorrelation function (ACF) for each of the 36 skills. For most of the skills, the Markov Chains seem well mixed, and the ACF plots show that the autocorrelation diminishes somewhat quickly. However, there are skills such as Agile Methodologies, Digital Marketing, Java, Marine Corps, Military Weapons, and Soccer, that have high autocorrelation, and the plots show that the Markov Chains were stuck in some areas. We keep this in mind when making inference about the parameters. Appendix C.2 contains plots of the posterior distributions, including the posterior means and 95% credible intervals. Looking at the credible intervals, we see that the significant skills include Baseball, Digital Marketing, Marine Corps, and Oracle Database. Using the posterior means as the regression coefficients, we fit the model on the test data to generate the probability that each applicant received an interview. Those with a predicted probability greater than 50% are classified into class 1, i.e. people who received an interview.

B.3 Artificial Neural Networks

When using the Artificial Neural Networks (ANN), we first need to specify the Neural Networks structure. Here we have one input layer with 218 nodes, corresponding to 218 skills of applicants and one output layer with one node which outputs a number between 0 and 1. This is a monotone function of the probability the particular applicant gets an interview. When the output is bigger than 0.5, we classify the corresponding applicant to receiving an interview. It remains to determine the number of hidden layers and the number of neurons in each hidden layer. Since there are no well-established theory on how to set these values, they are usually determined by a tuning parameter. Here we apply k-fold cross validation to determine the optimal number of nodes of a single hidden layer, then fix the number of neurons in each hidden layer to be the same, and find the approximate optimal number of hidden layers.

We use the development dataset for training and validation. In this dataset, there are 619 resume data, and we randomly sampled 610 out of the full dataset to do a 10-fold cross validation on choosing the optimal number of neurons for a single hidden layer. The cross validation result is shown in Figure 5. The figure on the left shows the cross entropy loss on the validation dataset, and the figure on the right shows the training time as a function of number of neurons. When the number of neurons increases from 0 to approximately 10, the cross entropy loss decreases dramatically. The classification error somewhat stabilizes when the neuron number is between 10 and 35, but starts to increase with more neurons. It is also clear in the figure on the right that more neurons cause the training time to increase exponentially. Thus, fixing the number of neurons in the hidden layer to be approximately 12 is reasonable in terms of classification performance and computational cost.

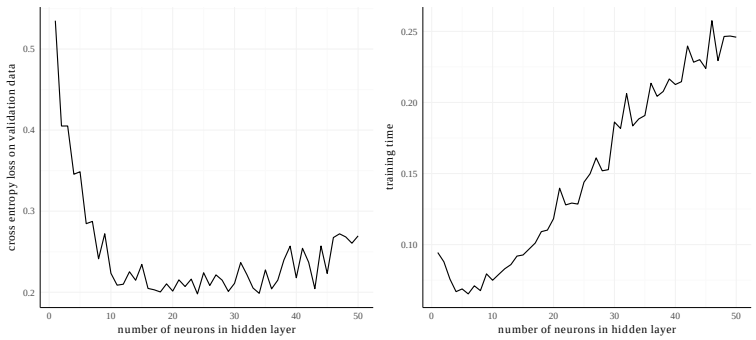


Figure 5: Cross validation for choosing optimal number of neurons for single hidden layer.

We fix the number of neurons to be 12 in each hidden layer and increase the number of hidden layers to approximately determine the best number of hidden layers to use. Again, we are using the 10-fold cross validation. Table 4 lists the experiment results, and the single hidden layer model performs the best. More hidden layers will slow down the training process since there are more weights to be estimated, which includes more calculations when doing backward propagation. However, more hidden layers do not enhance the classification accuracy. Thus in the final prediction phase, we fix one hidden layer with 12 neurons.

#hidden layers	1	2	3	4	5
cross entropy loss	0.28103	0.36692	0.49509	0.48392	0.68472
time	0.0889	0.0947	0.1334	0.1395	0.1636

Table 4: Cross validation for choosing best number of hidden layers.

B.4 Bayesian Neural Network

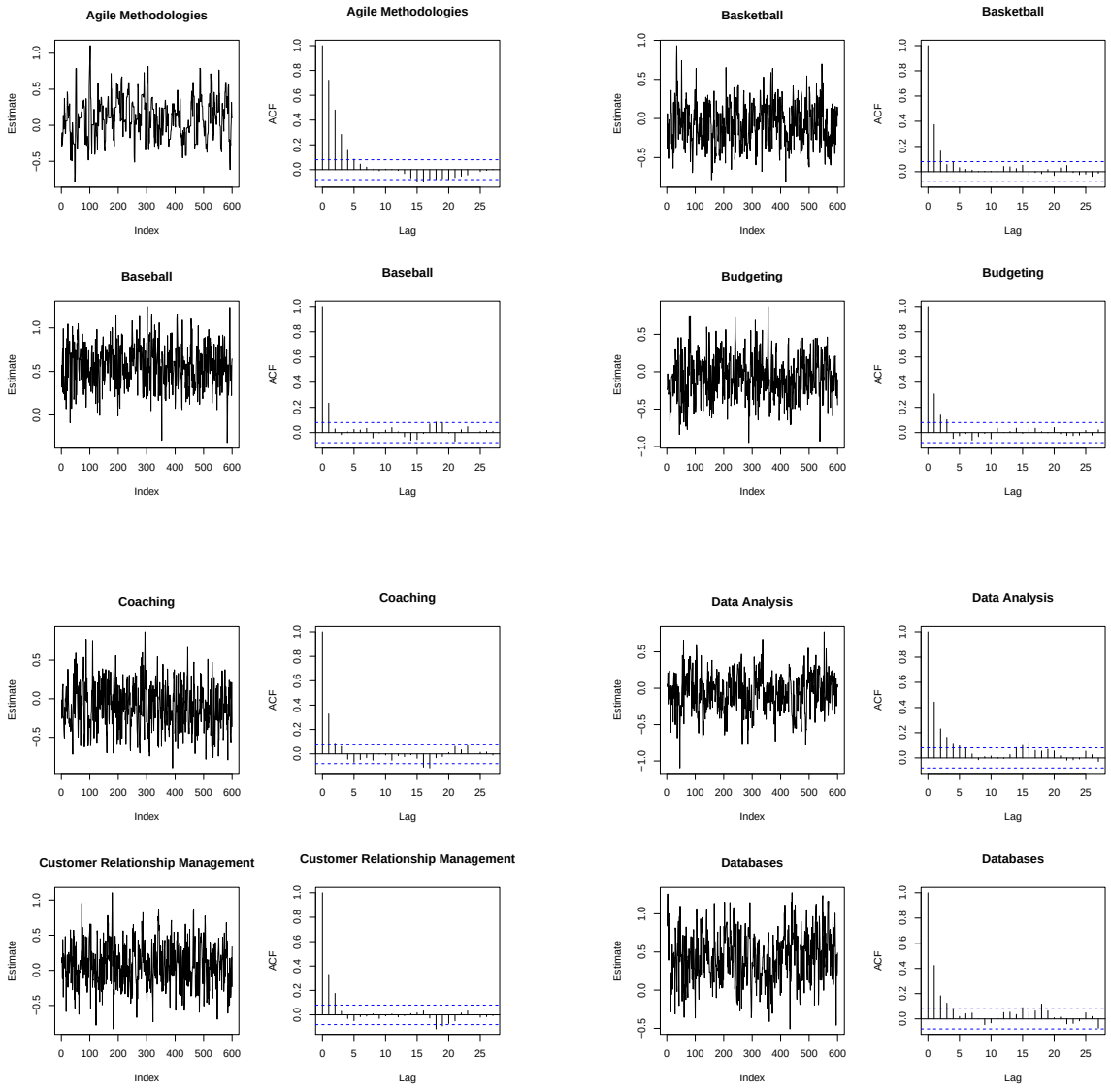
In order to arrive at the posterior distribution of each weight in the Bayesian Neural Network model, we specify the posterior distribution of each element of θ as coming from prior distribution $\theta_i \sim N(0, 1)$. That is, we set $c = 1$. Moreover, we set the adaptive step size for the random walk proposal as $(\nu_j)^{(t+1)} = (\nu_j)^{(t)} + \gamma_j^{(t)}(\alpha_{tj} - r)$ where we set $\gamma_j^{(t)} = t^{-2}$ and $r = .3$. In this way, we hope to reach an appropriate step size to where the acceptance rate is approximately 30%. To keep computational costs feasible, we only consider the 36 important skills identified in the Explanatory Data Analysis and run this Metropolis-within-Gibbs algorithm for 10,000 iterations. We use a 500 iteration burn in period and take every 10th iteration to avoid autocorrelation and initial condition bias.

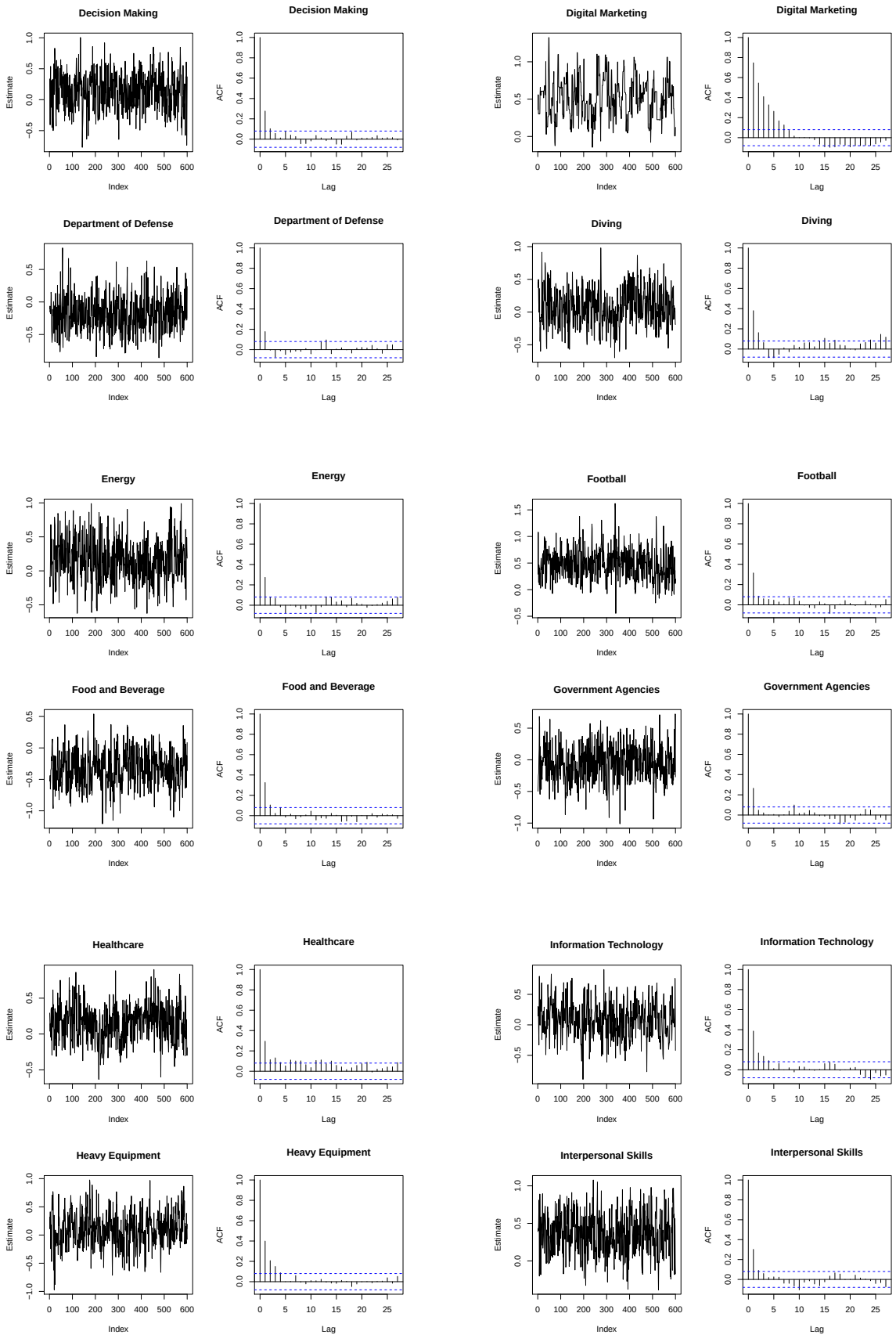
As there were 456 parameters in this model, we choose to not include the mixing or ACF plots as we did in the Bayesian Logistic Regression portion of the report. Generally, for non-intercept parameters, the mixing was quite conservative around the Neural Network starting conditions. That is, the mixing was tightly confined in a small neighborhood around this local minimum. With a larger c value, we could see that this mixing would occur over a larger portion of the parameter space. In addition, the intercept terms appeared to mix quite poorly, erratically jumping from point to point. However, under this model, the intercept, or bias terms, at each node are not identifiable. Therefore, this erratic behavior can simply be seen as the same set of biases being introduced across different nodes in the hidden layer. Generally speaking, tuning the variance parameter c to be larger and the step size $\gamma^{(t)} > \frac{1}{t^2}$ would encourage more exploration of the parameter space.

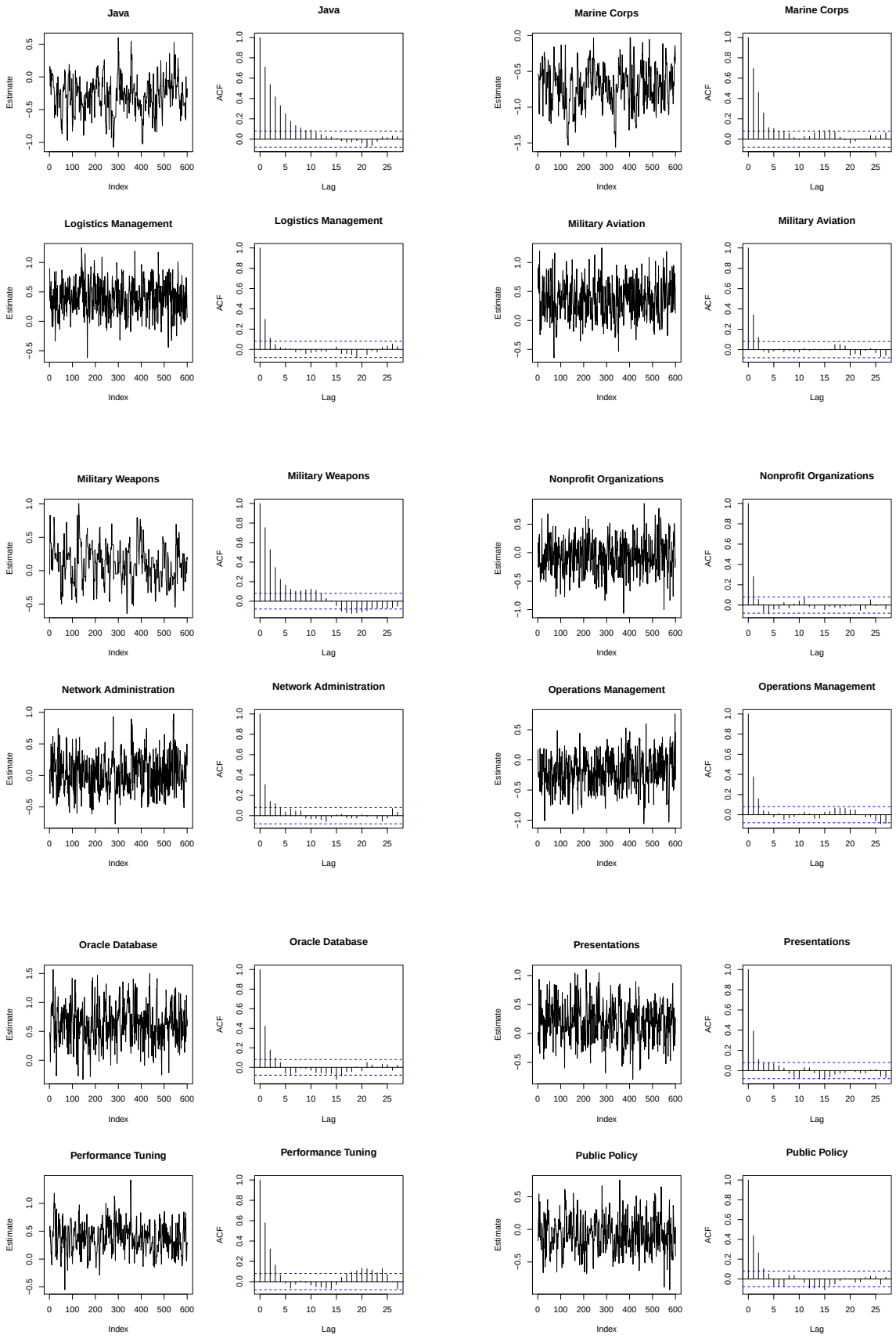
Lastly, we note that the run time of this network took considerably longer than the other methods considered here. Indeed, a single likelihood calculation (as needed in the calculation of the acceptance probability) took upwards of 6 seconds - longer than the full run time of the complete ANN. While this could be an artifact of inefficient coding by the authors, even with quite efficient implementation this method will be considerably slower than other methods consider in this report.

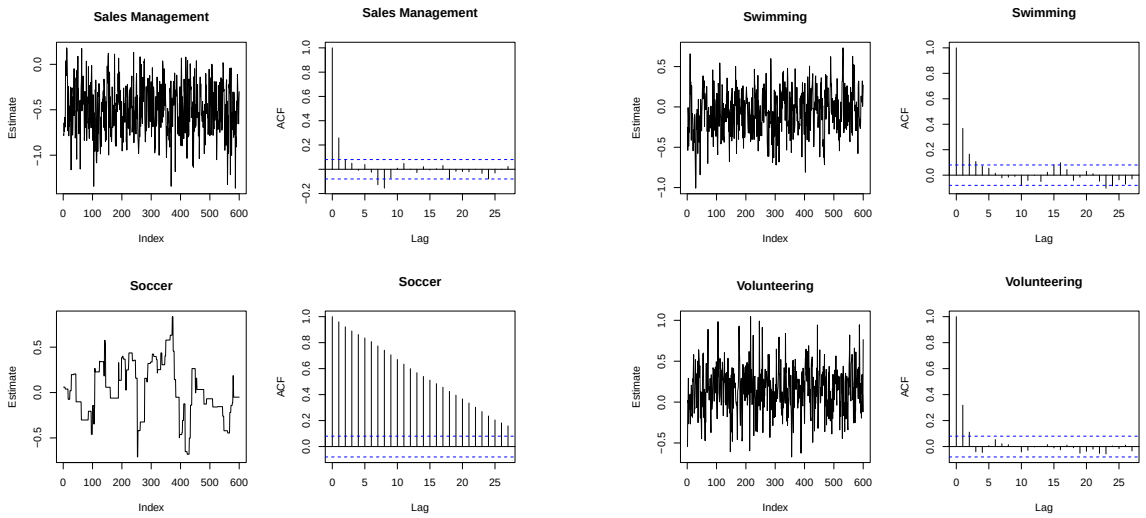
Appendix C Bayesian Logistic Regression Output

C.1 Markov Chain Path and ACF Plot

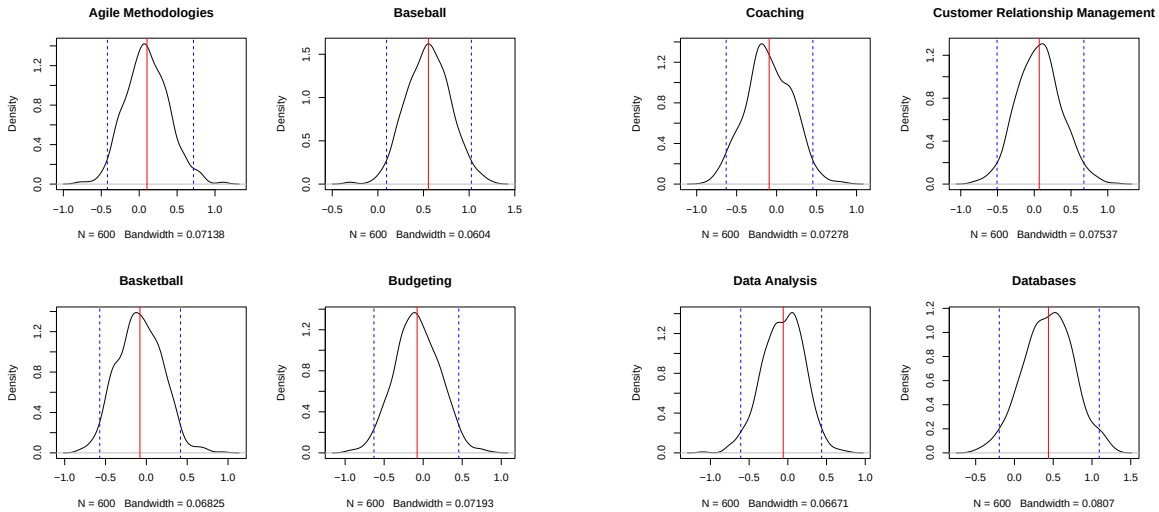


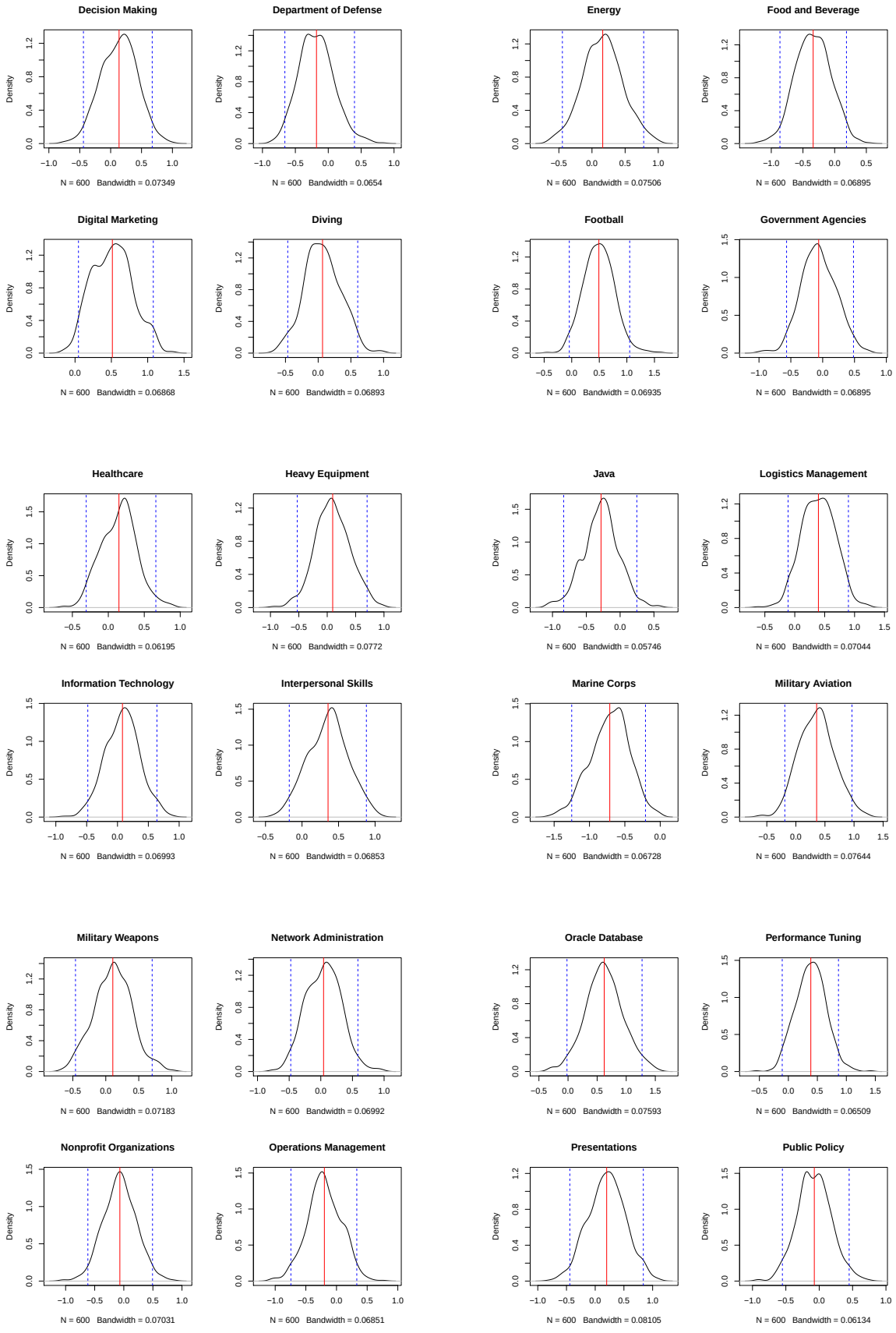

Decision Making
Estimate
Index
Decision Making
ACF
Lag
Digital Marketing
Estimate
Index
Digital Marketing
ACF
Lag
Department of Defense
Estimate
Index
Department of Defense
ACF
Lag
Diving
Estimate
Index
Diving
ACF
Lag
Energy
Estimate
Index
Energy
ACF
Lag
Football
Estimate
Index
Football
ACF
Lag
Food and Beverage
Estimate
Index
Food and Beverage
ACF
Lag
Government Agencies
Estimate
Index
Government Agencies
ACF
Lag
Healthcare
Estimate
Index
Healthcare
ACF
Lag
Information Technology
Estimate
Index
Information Technology
ACF
Lag
Heavy Equipment
Estimate
Index
Heavy Equipment
ACF
Lag
Interpersonal Skills
Estimate
Index
Interpersonal Skills
ACF
Lag

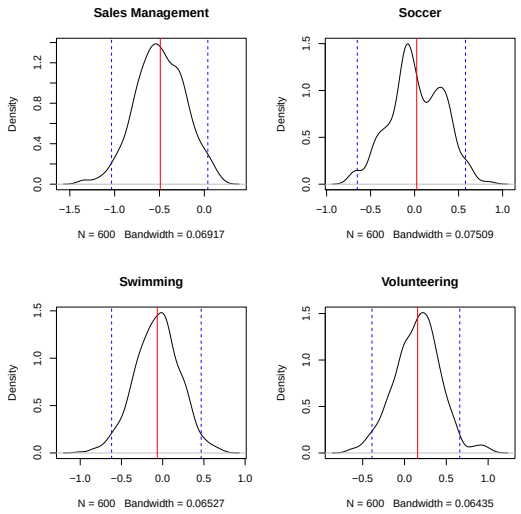

Java
Java
Marine Corps
Marine Corps
Logistics Management
Logistics Management
Military Aviation
Military Aviation
Military Weapons
Military Weapons
Nonprofit Organizations
Nonprofit Organizations
Network Administration
Network Administration
Operations Management
Operations Management
Oracle Database
Oracle Database
Presentations
Presentations
Performance Tuning
Performance Tuning
Public Policy
Public Policy
Estimate
ACF
Index
Lag



C.2 Posterior Density




Decision Making
N = 600 Bandwidth = 0.07349
Department of Defense
N = 600 Bandwidth = 0.0654
Energy
N = 600 Bandwidth = 0.07506
Food and Beverage
N = 600 Bandwidth = 0.06895
Digital Marketing
N = 600 Bandwidth = 0.06868
Diving
N = 600 Bandwidth = 0.06893
Football
N = 600 Bandwidth = 0.06935
Government Agencies
N = 600 Bandwidth = 0.06895
Healthcare
N = 600 Bandwidth = 0.06195
Heavy Equipment
N = 600 Bandwidth = 0.0772
Java
N = 600 Bandwidth = 0.05746
Logistics Management
N = 600 Bandwidth = 0.07044
Information Technology
N = 600 Bandwidth = 0.06993
Interpersonal Skills
N = 600 Bandwidth = 0.06853
Marine Corps
N = 600 Bandwidth = 0.06728
Military Aviation
N = 600 Bandwidth = 0.07644
Military Weapons
N = 600 Bandwidth = 0.07183
Network Administration
N = 600 Bandwidth = 0.06992
Oracle Database
N = 600 Bandwidth = 0.07593
Performance Tuning
N = 600 Bandwidth = 0.06509
Nonprofit Organizations
N = 600 Bandwidth = 0.07031
Operations Management
N = 600 Bandwidth = 0.06851
Presentations
N = 600 Bandwidth = 0.08105
Public Policy
N = 600 Bandwidth = 0.06134
Density


Sales Management
Density
N = 600 Bandwidth = 0.06917
Soccer
Density
N = 600 Bandwidth = 0.07509
Swimming
Density
N = 600 Bandwidth = 0.06527
Volunteering
Density
N = 600 Bandwidth = 0.06435