Addressing Latent Biased Sampling in an Online Fashion

Chris Cremer
CSC2506 Course Project
ccremer@cs.toronto.edu

Abstract

Most learning algorithms assume that the training data distribution is reflective of the test data distribution. However, this assumption is commonly violated in many real world problems where there exists sampling biases that lead to some sub-groups being over represented in the training data. For classification problems, the sampling bias can be addressed by weighting each sample based on the class proportions. For unsupervised problems, the class distribution is latent. This project investigates how to address biased training data when the class distribution is latent. The method that achieves the best performance is re-weighting the samples based on their density estimation.

1 Introduction

Although most algorithms assume that the data are independent and identically distributed (IID), in many real world problems, sub-groups of samples exhibit a high degree of correlation amongst both features and labels. Often there exists sampling biases that lead to some sub-groups being over represented in the training data. Since most learning algorithms assume that samples are IID, the biases in the sample space cause the model to learn a misrepresentation of the true distribution. For a classification problem, this imbalance can be addressed by modifying the errors of each sample based on the known class distribution. Generalized linear mixed models (GLMM) address this by modeling the effects attributed to factors other than the independent variables. [1],[2]. However, for unsupervised problems, the class distribution is latent.

One option to address the problem of unbalanced sample space sampling in the unsupervised context could be to measure the similarity of each sample with all the others and then give higher weights to samples that are more dissimilar to the rest, in essence, balancing the sampling over the sample space. This method's runtime would scale quadratically with the size of the training set and would be impractical for very large datasets. In contrast, if the model could be trained in a online fashion, where data becomes available in a sequential manner as opposed to training on the entire dataset at once, then the size of the dataset would not be limited by the runtime.

In this project I propose several techniques to address the sampling bias problem in an online fashion. In essence, each model reduces the effect that similar samples have on the training of the model. Each model can be trained via gradient descent and thus can accommodate very large datasets.

A motivating application for solving this problem is protein structure prediction from evolutionary couplings. For biological sequence data there are strong sampling biases due to phylogenetic relations between species, due to the sequencing of different strains of the same species, and due to a non-random selection of sequenced species. The sampling is therefore clustered in sequence space, thereby introducing spurious non-functional correlations, whereas other viable parts of sequence space are statistically underrepresented [3]. Another application is balancing the learning of autoencoders. For instance, a group at Google and Stanford tried building highlevel, class-specific feature detectors from only unlabeled data of 10 million images from the Internet. Their results show that

they learned face detectors, cat face detectors, and human body detectors[4]. They learned those feature detectors because the Internet is highly biased towards those types of images. If they had balanced the sampling over the entire sampling space, they may have learned many other interesting feature detectors.

2 Methods

To evaluate the performance of various methods at dealing with bias sampling, I will working with the MNIST dataset [5].

2.1 Biased Sampling

The training and validation sets were subject to biased sampling, whereas the test set was left unaltered. The classes were split into two groups: a major and a minor. The major group included digits 0, 1, 2, 3, and 4, and the minor group included digits 5, 6, 7, 8, and 9. If the bias parameter is 0.5, then both groups have the same number of samples, ie. no bias. A bias parameter of 0.6 indicates that the major group contains 60% of the samples and the minor class has 40% of the samples. The number of each class within each group is divided evenly among the classes.

2.2 Regular Training

I trained a neural net with 784 input neurons, one hidden layer with 100 neurons, and an output neuron with 10 neurons. The hidden layer had a ReLu activation function and the output layer had a softmax function. For all experiments, the learning rate was 0.0001, the momentum was 0.9, and batch size was 1000. The cross-entropy error was minimized.

2.3 Weighted Learning

In a supervised classification setting, an easy way to address the bias sampling is to weight each sample by its class. More specifically, the error for each sample was scaled by the mean number of samples per class divided by the number of samples in that class. Thus the more samples a class has, the lower its error. This causes the gradient to be less influenced by the major classes. Note however that this will be the gold standard that the other methods will strive to achieve without knowing the class counts.

2.4 Squared Error Learning

Just as L1 error can make a method more robust to outliers compared to L2 error, squaring the cross entropy error can make the method less robust to outliers. In this scenario, the classes which are under-sampled are the 'outliers'. The cross-entropy normally is $-\sum_{i=0}^{N} t_i log(y_i)$, but in this model its been changed to $\sum_{i=0}^{N} (t_i log(y_i))^2$.

2.5 Background/Foreground Model

This model aims to reduce the gradient from samples that can be predicted using a ransom subset of samples. The predictive distribution for the kth output $\hat{y}_k$ given $\hat{x}_k$ and a random subset of the training samples D is defined as,

$$P(\hat{y}_k|\hat{x}_k, D, \pi) = \pi F(\hat{y}_k|\hat{x}_k, \theta) + (1-\pi)B(\hat{y}_k|\hat{x}_k, D) \tag{1}$$

where,

$$B(\hat{y}_k|\hat{x}_k, D) = \frac{\sum_{i=1}^{k-1} K(\hat{x}_k, x_i) y_i}{\sum_{i=1}^{k-1} K(\hat{x}_k, x_i)}. \tag{2}$$

The foreground model $F(\hat{y}_k|\hat{x}_k, \theta)$ can be any probabilistic prediction model. The variable θ represents the parameters of the foreground model. The variable D is a random subset of the training

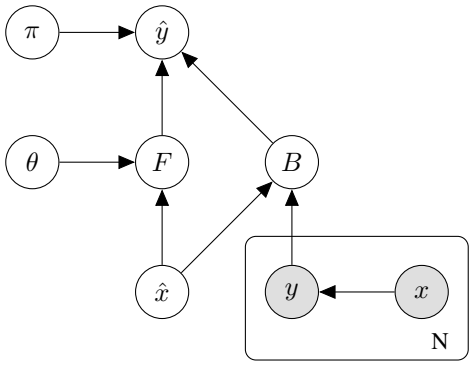


Figure 1: Graphical model of the background/foreground model

data. The parameter π is the ratio of the prediction attributed to the foreground model versus the background model. The function $K(a, b)$ measures the similarity between samples a and b, in this case the Euclidean distance is used.
Let's examine the derivative of the log probability of sample k with respect to the parameters of the foreground model,

$$\frac{\partial ln(P(\hat{y}_k|\hat{x}_k, D))}{\partial \theta} = \frac{\pi}{P(\hat{y}_k|\hat{x}_k, D)}\Big(\frac{\partial F(\hat{y}_k|\hat{x}_k, \theta)}{\partial \theta}\Big) \tag{3}$$

$$= \frac{\pi}{\pi F(\hat{y}_k|\hat{x}_k, \theta) + (1-\pi)B(\hat{y}_k|\hat{x}_k, D)}\Big(\frac{\partial F(\hat{y}_k|\hat{x}_k, \theta)}{\partial \theta}\Big) \tag{4}$$

We see that the parameters of the foreground model change in proportion to the performance of both the foreground and background models. Consequently, if the background model is able to predict the sample, then there will be little change in the foreground model.

This model addresses the sampling bias problem because if there exist samples that are very correlated, such that they don't provide any new information, then they will have little effect on the foreground model. Thus the foreground model will be trained on a more balanced distribution of the sample space. This model can be trained in an online manner via gradient descent.

2.6 Kernel Density Estimation

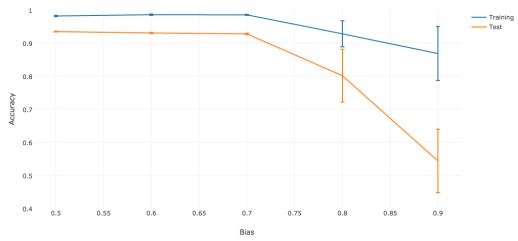
Another approach to dealing with the unbalanced classes is to weight the samples based on their density. Prior to training, a 1000 samples are taken from the training set and its density is estimated with a Gaussian kernel with a bandwidth of 0.5. The densities are shifted up so that they are all positive, then divided by the mean so that the mean density is 1. Finally, each sample's error is weighted by the reciprocal of its density.

3 Results

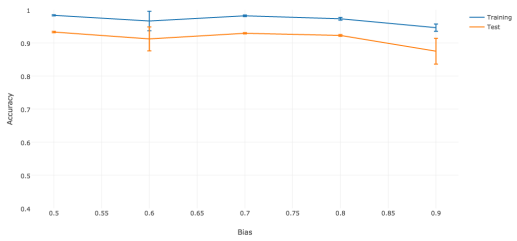
For each method, 10 iterations were performed. The mean of the top 5 training and test accuracies are shown in order to reduce the effect of the iterations which got caught in local minimas.

3.1 Regular Learning

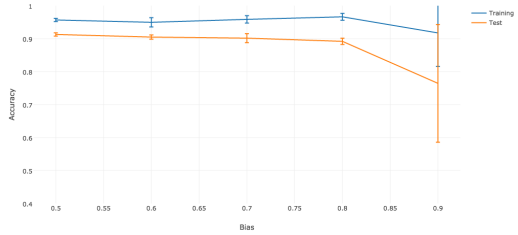
Fig. 2a shows the performance of training the neural net with varying levels of bias in the training data. Up to 0.7 bias, there is nearly no affect compared to 0.5 bias. At 0.5 the mean accuracy is 93%. At 0.9 bias, the mean accuracy is 54%, which means it is essentially only predicting the over-sampled classes.



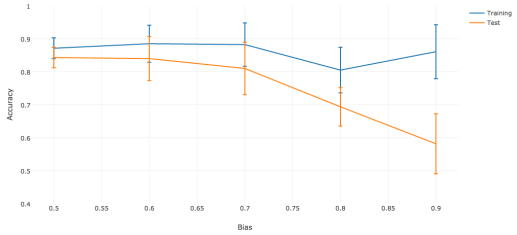
(a) Regular



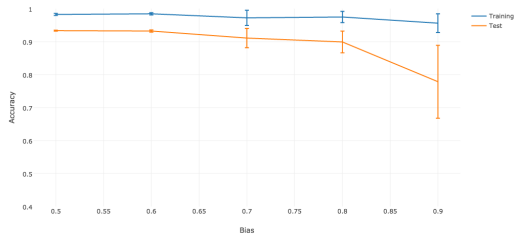
(b) Weighted



(c) Squared



(d) Background/Foreground



(e) Kernel Density Estimation

Figure 2: The training and test accuracies of each method with standard deviation error bars.

3.2 Weighted Learning

In a classification task, the counts of each class are known prior to running the model. Fig. 2b shows how the performance on the test set can be maintained even at 0.9 bias if the errors are weighted by the class proportions. At 0.9 bias the mean accuracy is 87%, which is impressive given that only 10% of the samples in the training set are among the minor group of classes. This is the ideal result that the other methods are striving to achieve without knowing the counts for each class.

3.3 Squared Learning

Squaring the error helps the model become more robust to the unbalanced classes. Fig. 2c shows the accuracy of squaring the error. Comparing Fig. 2a to Fig. 2c, we see that the mean test accuracy at 0.9 bias is 76% for squaring versus 54% without it. However, the standard deviation is much larger which is an indication that squaring the error causes the model to be more susceptible to local minimas.

3.4 Background/Foreground Learning

Fig. 2d is a plot of the performance of the background/foreground model. It performs noticeably worst than the other methods. Without any bias (0.5 bias) it only achieves 84% test accuracy. At 0.9

bias its mean test accuracy is 58%. There several possible explanations for this performance which are discussed briefly in the discussion section.

3.5 Kernel Density Estimation Learning

In KDE learning, we re-weight each sample by the reciprocal of its density. Fig. 2e is a plot of the performance of this model. At no bias, it has 93% mean test accuracy, and at 0.9 bias it has 77% accuracy. This is highest accuracy achieved by any model tested in this project.

3.6 All methods

To summarize and contrast all the methods in Fig. 2, the plot in Fig. 3 are all the mean test accuracies of each model. We see that squaring the error and weighting the error by the sample density perform very similarly.

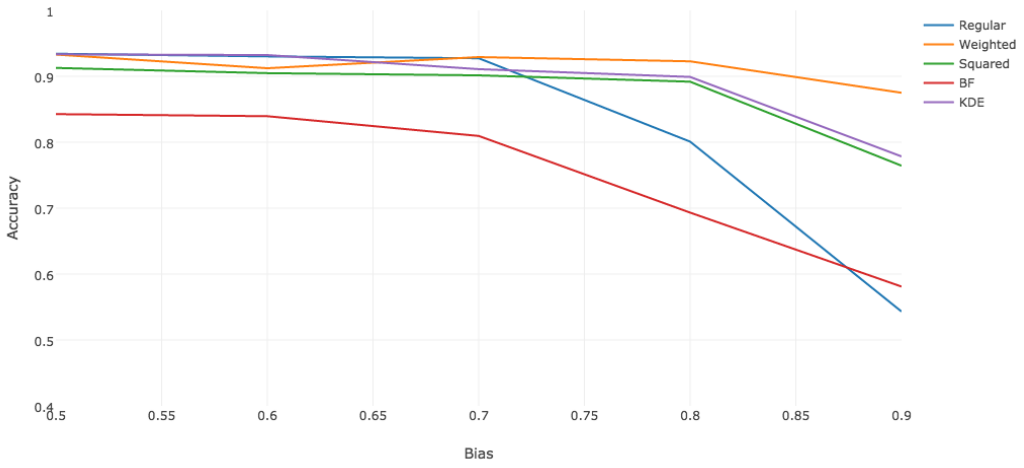


Figure 3: Mean test accuracy of each method versus amount of bias in the training data

4 Discussion

4.1 Background/Foreground Shortcomings

Out of all models tested, the one that performed noticeably worst than the others was the background/foreground model. There are several possible explanations for this. Firstly, since it was computationally expensive for the background model to make predictions for each sample, the number of samples it used to make predictions may not reflect the true distribution. Conversely, if too many samples are used in the background model, then its prediction may be too accurate, which defeats its purpose. Another likely reason is the distance metric (Euclidean distance) used to compare the similarity of samples was likely not precise, resulting in inaccurate predictions.

4.2 Oversampling

An interesting observation while performing various types of sampling bias was that if a class is over-sampled with replacement while keeping the other classes unchanged, there was no noticeable affect to its accuracy (results not shown). For instance, the data of class 0 was replicated 20 times more than any other class but this didn't affect the final accuracy. This observation suggests that, not the counts, but rather the variation within a class is important for affecting model learning.

4.3 Density of Samples

It is interesting to investigate how weighting the samples based on their densities performed better than the other methods. Fig. 4 is a scatter plot of the 3000 samples from the validation set which is biased 90% towards the major classes. The blue dots are the samples from the major (over-sampled) classes (0,1,2,3,4) whereas the orange are from the minor (under-sampled) classes (5,6,7,8,9). Since we are projecting onto only the top two principal components, much of the variation is lost, however, we can see that the under-sampled classes are among the over-sampled classes. This means that their densities were likely very similar, and thus weighted equally. In contrast, the points to the right of PC1 seem to strongly clustered together, resulting in high density, thus they received low weight. This figure helps visualize how some samples have higher density and thus a smaller weight.

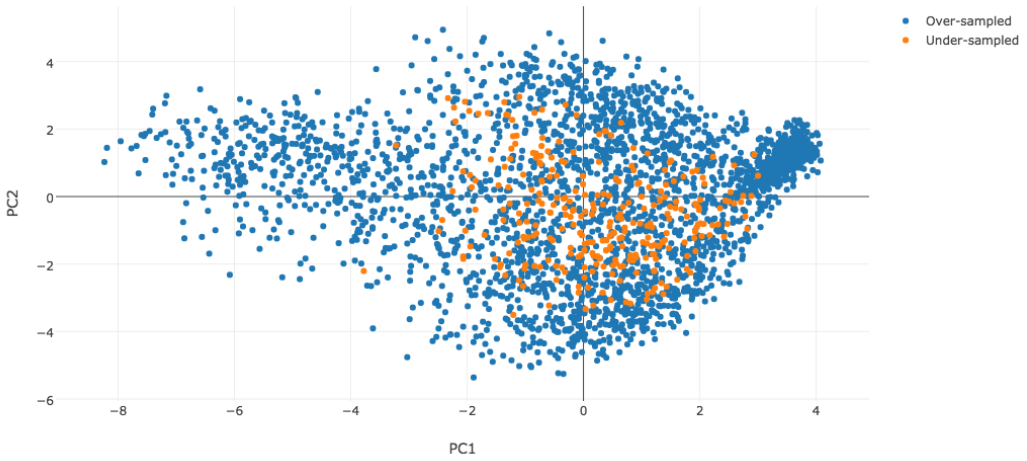


Figure 4: The distribution of the validation set with 0.9 bias

5 Conclusion

This project showed the performance of several methods at dealing with training data distributions that don't reflect the test data distribution. The assumption was that the test distribution is uniform and thus the methods try to push the model towards more balanced class distributions. The method that performed best was re-weighting samples based on their kernel density estimation. These results have implications in applications where datasets have a high level of sampling bias.

References

[1] C. Lippert, J. Listgarten, Y. Liu, C. M. Kadie, R. I. Davidson, and D. Heckerman, "Fast linear mixed models for genome-wide association studies," *Nature methods*, vol. 8, no. 10, pp. 833–835, 2011.

[2] M. Dundar, B. Krishnapuram, J. Bi, and R. B. Rao, "Learning classifiers when the training data is not iid.," in *IJCAI*, pp. 756–761, 2007.

[3] C. Baldassi, M. Zamparo, C. Feinauer, A. Procaccini, R. Zecchina, M. Weigt, and A. Pagnani, "Fast and accurate multivariate gaussian modeling of protein families: predicting residue contacts and protein-interaction partners," *PloS one*, vol. 9, no. 3, 2014.

[4] Q. V. Le, "Building high-level features using large scale unsupervised learning," in *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pp. 8595–8598, IEEE, 2013.

[5] Y. Lecun and C. Cortes, "The MNIST database of handwritten digits,"