Performance of Logistic Regression and Softmax Regression with Different Settings on Classifying Traffic Signs

Yigit Korkmaz A59010764,
ykorkmaz@ucsd.edu

Marcus Schaller A59004764
mschalle@ucsd.edu

Abstract

Regression is the process of calculating the relationship between independent and dependent variables. It is used in many machine learning algorithms and is the backbone of deep learning. This paper will discuss the use of two regression methods, Logistic Regression and Softmax Regression, on a German Traffic Sign dataset and discuss the results of both.

1 Introduction

This paper will discuss how Logistic and Softmax Regression can be applied in conjunction with different gradient descent methods to a dataset in order to classify images. It will compare the two techniques and provide insight into how they differ and when one is more useful than the other. Additionally this paper will look into the use of Principle Component Analysis (PCA) and discuss its use as a prepossessing technique.

2 Background

The German Traffic Sign Recognition Benchmark was a multi-class image classification challenge held at the International Joint Conference on Neural Networks 2011. The dataset used in this paper includes 34799 total images that are labeled with up to 43 different classes. The provided dataset includes images of traffic signs that have been both center aligned in the picture as well as unaligned versions of them. In order to prepare the data for training, k-fold cross validation was done to create k different batches across the data. The validation and testing data is assigned to 2 sets and the remaining k-2 sets are assigned to the training data.

3 Related Work

There are many other related works that classify the signs with in this dataset. Many of which are more accurate than what we found in our project. Many papers used Convolutional Neural Networks which achieved an accuracy of greater than 99 percent [2]. A paper by a team from San Jose used tree classifiers in conjunction with Histogram of Oriented Gradients which obtained a classification rate of 97 percent [3]. This proves that there are many methods that can be used in order to classify these images.

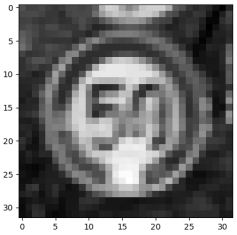


Figure 1: Example Dataset Image

4 Methods

4.1 Data Pre-processing

In order to prepare the data for training, some type of pre-processing must first be applied. This project included both z-score normalization as well as PCA. Z-score normalization involves normalizing the values within the dataset such that the mean of all values is 0 and the standard deviation is 1 by applying the following equation.

$$z = \frac{x - \mu}{\sigma} \tag{1}$$

PCA allows one to reduce the dimensions of a dataset. Given a dataset of length *N* PCA first calls for subtracting the dataset by its mean and using the following equations to calculate the covariance matrix of the dataset.

$$\mu_{reduced} = x - \mu \tag{2}$$

$$\frac{\mu_{reduced}^{T} \cdot \mu_{reduced}}{N - 1} \tag{3}$$

Next the normalised eigenvectors *v* and eigenvalues λ are calculated and sorted in descending order. Given k desired principle components, the eigenvectors v_k that correspond to the top *k* eigenvalues are selected. Finally the feature matrix is multiplied by v_k to return the transformed data, where each row is one sample which is represented by k principal components. Since in real world, we do not have access to validation and test data, we need to use the principal components calculated from training set. Thus, the test and validation sets are both transformed with the v_k calculated from the training set, after subtracting the mean of training set from each sample.

4.2 Logistic Regression

Logistic Regression involves using the sigmoid function to map the logits to a range between 0 and 1. In order to do so, the weights *w* and the feature *x* multiplied and given as input to sigmoid function to get the probability that it belongs to a given class.

$$h_\theta(x) = \frac{1}{1 + e^{-w^T x}} \tag{4}$$

Values greater than 0.5 belong to class 1 and values below belong to class 0. The cost function is given as the following.

$$J(\theta) = -\frac{1}{m} \sum_{i=1}^{m} \left[y^{(i)} \times \log\left(h_\theta\left(x^{(i)}\right)\right) + \left(1 - y^{(i)}\right) \times \log\left(h_\theta\left(x^{(i)}\right)\right) \right] \tag{5}$$

In this equation, y represents the target of that sample, i.e. true output and $h_\theta(x)$ represents the output of the network. By using gradient descent in the direction of the loss function and multiplying it with the learning rate α, one can update the weights in an iterative way.

$$\theta_j \leftarrow \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta) \tag{6}$$

4.3 Softmax Regression

Unlike Logistic regression, Softmax regression allows us to compare multiple classes at once. Now, given the feature input a vector y^n is outputted where each element, y_k^n represents the probability that x^n is in class k. The activation function is given as follows.

$$y_k^n = \frac{\exp\left(a_k^n\right)}{\sum_{k'} \exp\left(a_{k'}^n\right)} \tag{7}$$

$$a_k^n = w_k^T x^n \tag{8}$$

The multi-class cross-entropy cost function is then given as:

$$E = -\sum_n \sum_{k=1}^{c} t_k^n \ln y_k^n \tag{9}$$

Finally once again calculating the gradient and applying gradient descent at each iteration will result in Loss converging to a minimum.

$$-\frac{\partial E^n(w)}{\partial w_{jk}} = \left(t_k^n - y_k^n\right) x_j^n \tag{10}$$

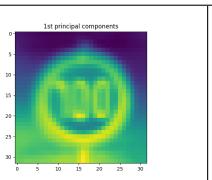
$$\theta_{nk} \leftarrow \theta_{nk} - \alpha \frac{\partial E^n(w)}{\partial w_{jk}} \tag{11}$$

5 Results

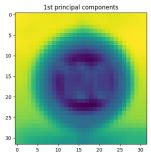
This section will discuss what we found in each section of the report and answer the questions that were included in the problem assignment.

5.1 Principal Component Analysis

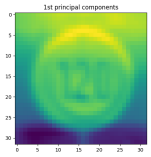
It is a good idea to change the inputs to have zero mean and unit standard deviation because it will make training faster since the error surface will be shaped better. Additionally it will normalize the dataset so the scale of every feature will be similar. It can be seen in the figures below that aligned data plays in important role in PCA as the images vary vastly between each principle component. Since, if the dataset is unaligned, it will be harder to produce effective principal components, since the position of the sign might be slightly different in each sample.



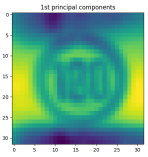
"First PC for aligned"



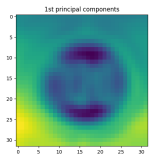
"Second PC for aligned"



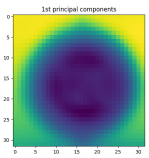
"Third PC for aligned"



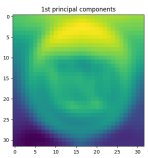
"Fourth PC for aligned"



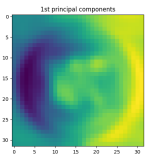
"First PC for unaligned"



"Second PC for unaligned"



"Third PC for unaligned"



"Fourth PC for unaligned"

5.2 Logistic Regression

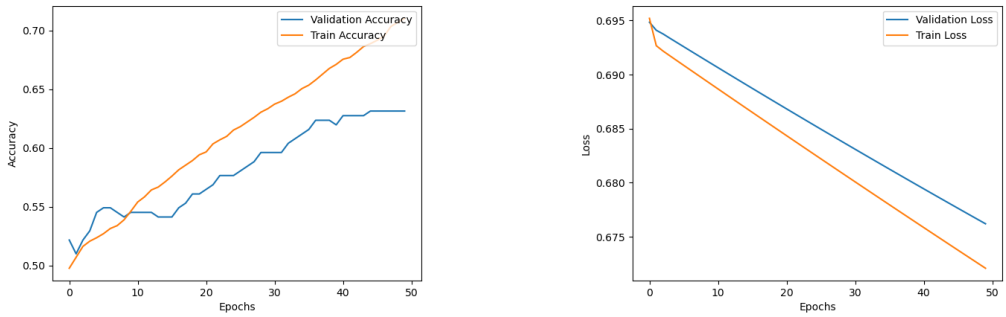
5.2.1 Implement Logistic Regression via Gradient Descent.

Logistic gradient descent only needs one output unit because we are only comparing between two classes. Therefore any time $P(y = 1|\lambda, x) > 0.5$ it will belong to class a otherwise it will belong

to class b. Important thing here is to convert class targets to binary since loss is calculated that way, i.e. 0 means class a, 1 means class b.

5.2.2 Evaluate the model using unaligned dataset. (class 7 vs class 8)

The learning rate we used in this section was 0.0023. In 50 epochs on unaligned data we achieved a accuracy of 62.35 percent and a loss of 0.6742 on the test set. It is understandable that the results will be poor since as shown in the figure above, and explained in PCA part, applying PCA on unaligned data will not be very helpful.

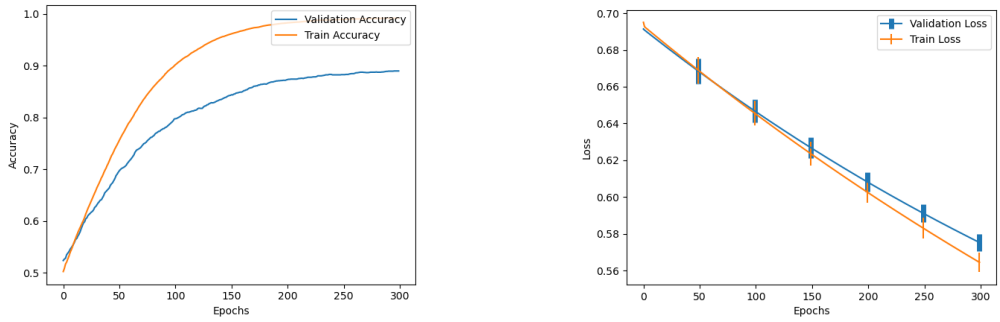


(a) (b)

Figure 2: (a) Accuracy on unaligned data, (b) Loss on unaligned data for part 5.b

5.2.3 Evaluate the model using aligned dataset. (class 7 vs class 8)

After 300 epochs trained on aligned data with a learning rate of 0.0023 we achieved a test accuracy of 88.43 percent and a loss of .5767. As seen in the figure above, applying PCA on aligned dataset produces better results than applying PCA on unaligned dataset, thus resulting in higher accuracy.



(a) (b)

Figure 3: (a) Accuracy on aligned data, (b) Loss on aligned data for part 5.c

Clearly learning rate plays an important role in gradient descent. As when learning rate is two high the loss quickly diverges. When it is too low it will take many more epochs to train.

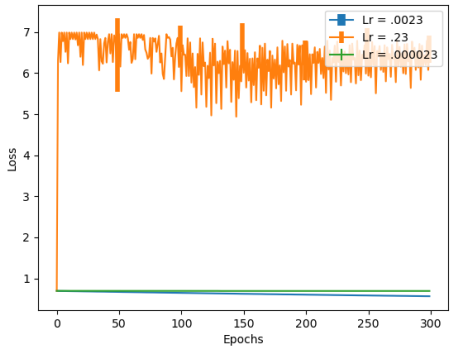
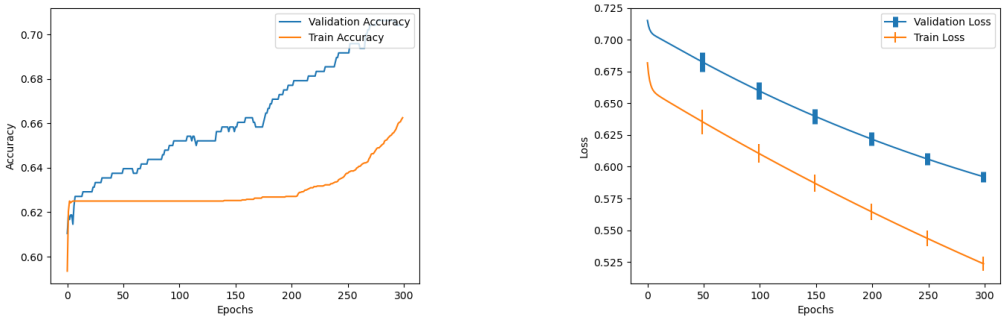


Figure 4: Loss on aligned data at varying learning rates for part 5.c

5.2.4 Evaluate the model using aligned dataset (class 19 vs class 20)

In this section we achieved an accuracy of 68.74 percent and a Loss of .6203. Using a learning rate of 0.0023 in 300 epochs. It should be noted that if we used a higher learning rate resulted in a accuracy closer to what was achieved in the previous class. This is likely because the dangerous curve signs are much more similar and take either a greater learning rate or more epochs to minimize loss. Another reason might be the sample set for these classes are smaller than the set for class 7 and 8, resulting in smaller set of principal components and longer time to learn the weights, ending up with a worse accuracy.

Accuracy = .6874 Loss = .6203



(a) (b)

Figure 5: (a) Accuracy on aligned data, (b) Loss on aligned data for classes 19 and 20 for part 5.d.

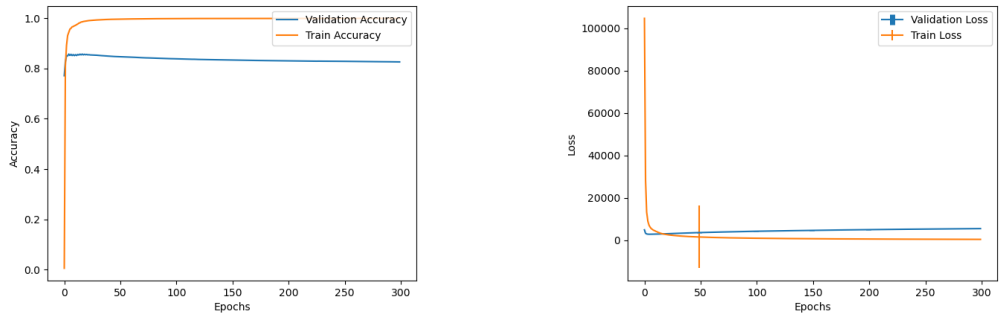
5.3 Softmax Regression

5.3.1 Implement Softmax Regression via Gradient Descent

In this part Softmax Regression is implemented by using the description mentioned above. However, one key part is, in Softmax Regression, each target should be converted to one-hot encoded versions, because the output of the network for each sample will be a vector which contains probabilities for different classes. To calculate accuracy, these vectors should be decoded to get the label.

5.3.2 Evaluate your network on all 43 traffic signs (aligned dataset).

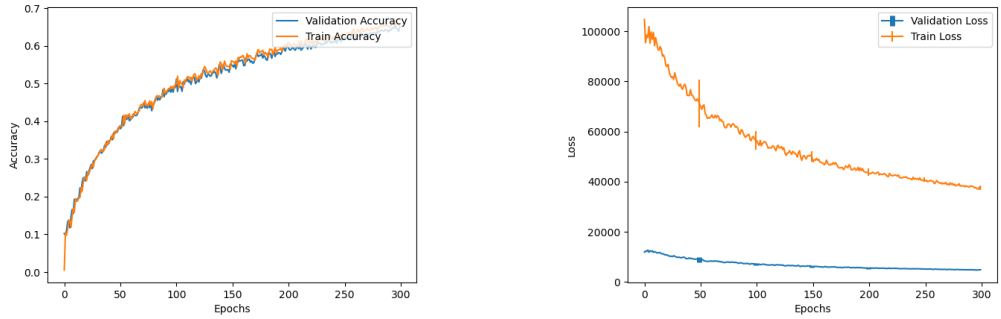
When using softmax regression with a learning rate of 0.002 we achieved a test accuracy of 85.5 percent and a loss of 2780.77.



(a) (b)

Figure 6: (a) Accuracy on aligned data, (b) Loss on aligned data for part 6.a.

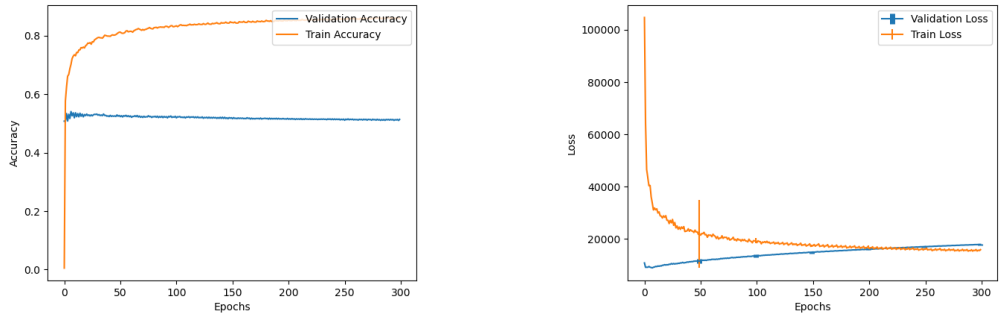
When training without PCA on alligned data we achieved a Test accuracy of 66.01 percent and a test loss of 4684.1.



(a) (b)

Figure 7: (a) Accuracy on aligned data without PCA, (b) Loss on aligned data without PCA for part 6.a.

When training on data that is unaligned with PCA we achieved a test accuracy of 52.82 percent and a test loss of 8991.0.



(a) (b)

Figure 8: (a) Accuracy on unaligned data, (b) Loss on unaligned data for part 6.a.

These differences in the accuracy clearly shows the importance of PCA and aligning on the performance of the classification. One needs to standardize the samples as much as possible to prevent

perturbations related to difference in configuration of the samples and the features, such as scale of features and position of the objects.

The confusion matrix shows that the trained weights have difficulty classifying class 19 and class 29.

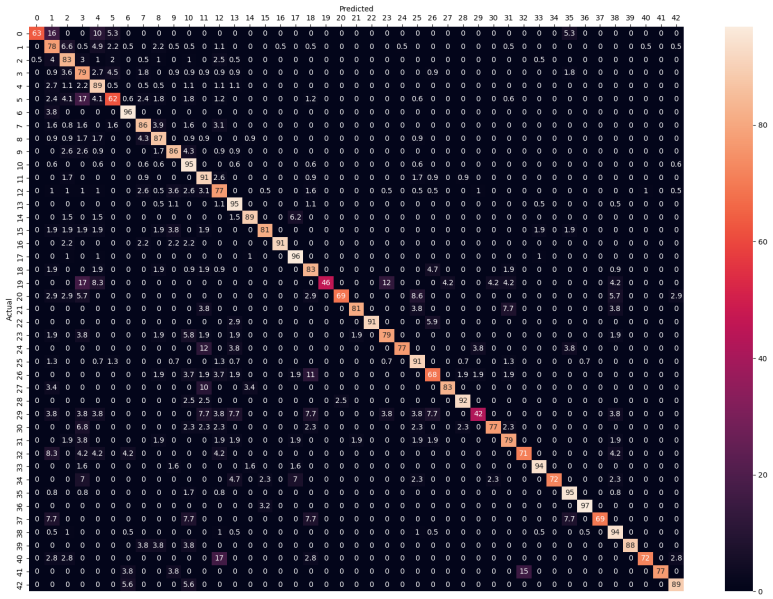
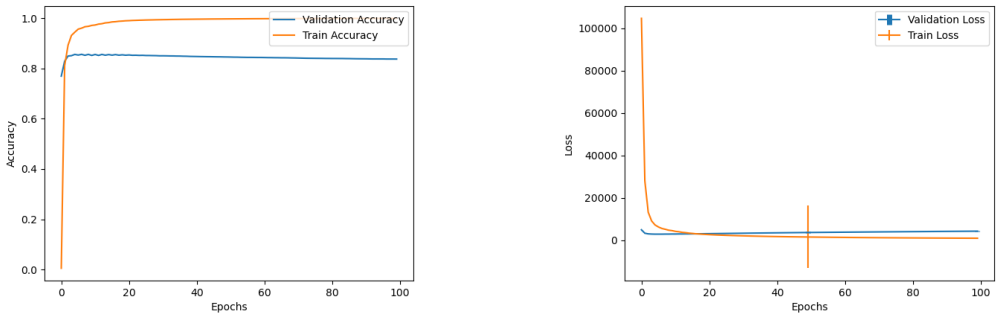


Figure 9: Confusion Matrix for part 6.a

5.3.3 Batch versus stochastic gradient descent.

Results of using stochastic gradient descent for 100 epochs can be found below. Note that, while using this method, training set is reshuffled at every epoch to get better learning.



(a) (b)

Figure 10: (a) Accuracy on aligned data, (b) Loss on aligned data for part 6.b.

Normally stochastic gradient descent should minimize the error in fewer epochs since it takes into account every sample specifically. While in batch gradient descent, whole training set is used at once. However it was found that in this problem there is no clear improvement between two methods.

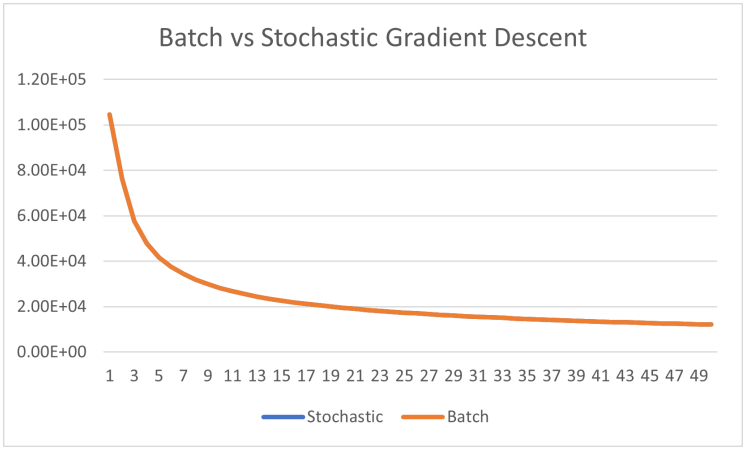
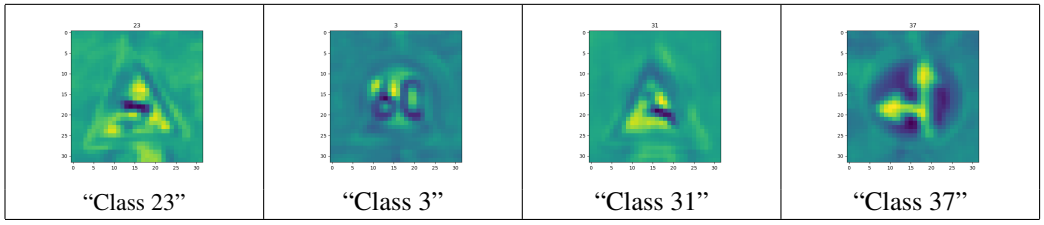


Figure 11: Stochastic vs Batch gradient descent

5.3.4 Visualize the weights.



The weights appear to represent the images. This is because after training, weights are meant to categorise the images when they are multiplied with them. Thus it is very likely for the weights to look like that class' image after training, which the case here as well.

5.3.5 Extra Credit

As mentioned in the problem, data in this case is imbalanced, i.e. each category has different number of samples. This inequality effects the accuracy of our classification as well. There are some ways to solve this issue. One of them might be determining on a limit (for example this limit can be the average number of samples that a class has) and then throwing away the samples in a class if that class has more number of samples than the average. However this will also reduce total number of samples we have, which effects the accuracy. Thus, we can use this method if we are sure that we already have more than enough samples.

Another method can be producing synthetic data for the classes that have few number of samples. However, we need to be careful when creating the synthetic data, our new samples should be consistent with the rest of the data in those classes. One way to create such samples can be taking the average image in the class that has fewer samples, and creating new data for the class randomly in the range of (average image $\pm$ standard deviation) of the samples in that class. We believe this method can be used when there is already fewer number of examples in the whole dataset and we cannot apply throwing-away method.

6 Conclusion

While there are many ways to classify images this paper describes two of the most common methods. When used in conjunction with other deep learning methods regression will allow us to classify these signs at a rate of 99 percent or more.

6.1 Individual contributions to project

6.1.1 Yigit

I was mainly responsible for the implementation of PCA and z-score normalization, softmax regression with both SGD and BGD, as well as the general structure and flow of the code. Also, I worked on writing the report, especially the parts about softmax regression.

6.1.2 Marcus

My primary contribution was the logistic regression section of the project as well as running and collecting the data for all of the experiments. This includes writing the code that involves plotting and the confusion matrix. Additionally I worked on writing up the report.

References

[1] G. W. Cottrell, "Logistic Regression Multinomial Regression," in CSE 251b Lecture 2.

[2] J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel, "Man vs. computer: Benchmarking machine learning algorithms for Traffic sign recognition," Neural Networks, vol. 32, pp. 323–332, 2012.

[3]F. Zaklouta, B. Stanciulescu, and O. Hamdoun, "Traffic sign classification using K-D trees and random forests," The 2011 International Joint Conference on Neural Networks, 2011.