Logistic Regression and Softmax Implementation

Andrew Ghafari
Department of Computer Science
University of California San Diego
aghafari@ucsd.edu

Jay Jhaveri
Department of Computer Science
University of California San Diego
jjhaveri@ucsd.edu

Abstract

In this assignment, we implemented logistic Regression and softmax for binary and M-ary classifications. In the former task, we tested our model on two sets of numbers: 5s vs 8s, which gave us a 93.9% test accuracy, and 7s vs. 2s which got us 96.5% accuracy. This can be explained by the idea that the first set of numbers has a lot more similarities than the second set where the two numbers are easily identifiable one from the other. In the latter one, we did a 10-class classification algorithm, and we got us 82.7%. One key finding was that the majority of important data in an image is contained in its first 30 principal components, which proved helpful in reducing dimensionality for classification algorithms. Secondly, after visualizing the weights of the softmax output layer, we realized that the 10 images represent the 0-9 digits we are trying to detect, which means that each weight converges to the object it is trying to detect in an image.

1 Introduction

We implemented logistic and softmax Regression in this study for binary and M-ary classification tasks. Logistic Regression is a statistical method used to predict a binary outcome's probability. In this task, we used logistic Regression to predict whether a given image is a 5 or 8, or a 7 or 2. The model uses the input features, in this case the 28x28 input image pixels, and a set of weights to calculate the probability of the image belonging to one of the classes, with the class with the highest probability being the chosen one. The weights are learned during the training process using a minimization algorithm of the binary cross entropy loss function using mini-batch gradient descent, which aims to minimize the error between the predicted and true labels.

Additionally, we used softmax to perform 10-class classification, where the goal is to classify the number represented by the 28x28 input bitmap image. Softmax is a generalization of logistic Regression to multiclass classification. In this task, the model uses the same input features and set of weights as logistic Regression. The weight terms learned during the training process also uses a minimization algorithm for the multiclass cross entropy objective loss function using mini-batch gradient descent. The output of the model is a probability distribution over all the classes, with the class with the highest probability being the predicted label of the image at hand.

One last important aspect of our study was using Principal Component Analysis (PCA) to reduce the dimensionality of the image inputs before running the abovementioned algorithms. PCA reduces the dimension of the image data by finding its directions of maximum variance. By projecting the data onto these principal components (vectors), we reduced the number of features while still preserving most of the information present in the data. In our study, we found that the majority of the important data in an image is contained in its first 30 principal components. By using PCA to reduce the dimensionality of the image inputs, we were able to significantly speed up the computation and storage requirements of the model while still maintaining a high level of accuracy. Additionally, by reducing the dimensionality, we also reduced the chances of overfitting, which is a common problem in high-dimensional data.

For the logistic regression part, after doing hyperparameter tuning, we realized that the best set of hyperparameters were p = 30, batch size = 64, no epochs = 100, learning rate = 0.001 and k folds = 10. We also tried both types of normalization of input data, min max and z score with the former giving the better results. After using the optimal weights for the 2 vs. 7 task, we achieved an accuracy of 96.5%, and for the 5 vs. 8, a 93.9%.

For the softmax regression part, the same said above applies, and we also got an accuracy of 82.7%

In the following sections, we will start off by talking about related work, or papers that inspired us in general. Next, we will be discussing pre-processing done on the images in the dataset and how it became ML-ready. Following this, we will be showing the results of implementing the algorithms implemented from PCA to logistic Regression to softmax regression. Finally, we will end this paper by providing team members' contributions and the source code.

2 Related work

We first would like to thank Prof Cottrell for the slides posted as we heavily relied on them while doing this programming assignment. More specifically, Lecture 1, Lecture 1A, and Lecture 2A from the Lecture Notes Winter 2023 on Piazza. In addition to that, we would also like the thank the TAs for the help they provided whether on piazza or in person during office hours.

We would also like to mention some of the articles that we have previously read about Deep learning techniques in general and for MNIST data in particular. Most of these papers were actually introduced to us during our computer vision course, when we were discussing history of neural networks for object classification and the ILSVRC competition.

"Gradient-Based Learning Applied to Document Recognition" by Yann et al. [1] was one of the first and most influential papers in the field. This paper presents LeNet, a neural network architecture trained on the MNIST dataset that achieved state-of-the-art performance at the time.

For the mini-batch gradient descent we also referred an article that explains the different types of gradient descent and how to implement them (Patrikar, 2021) [2]. Lastly, for implementing PCA, we also utilized the scikit-learn PCA documentation, which served handy for getting to know useful function for projections and inverse transform of the data.

3 Dataset

3.1 Description

The dataset provided to us is MNIST, a dataset of handwritten digits, consisting of 60,000 training images and 10,000 test images. Each image is a 28x28 grayscale image of a handwritten digit (0-9) and is associated with a label indicating the correct digit. We have previously used this dataset in our last quarter during our computer vision course and we are quite familiar with it. It is a staple dataset that is used everywhere for classification models.

To visualize the dataset, we randomly selected an image of each of the 10 classes in the dataset.

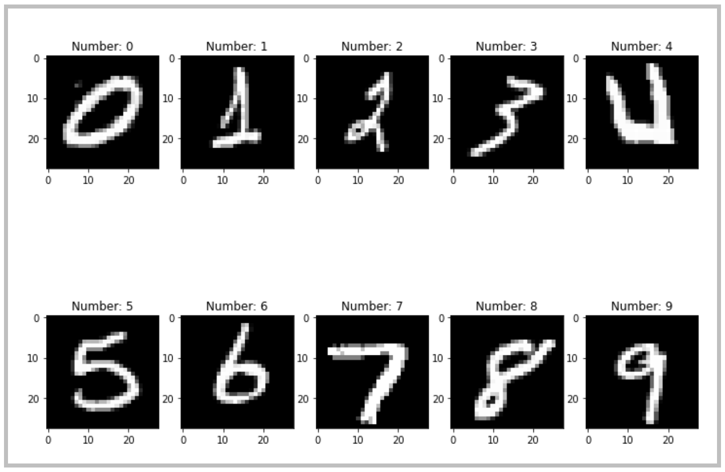

Figure 0: Random Depiction of Each Classes in the Dataset

These are some descriptive analysis we performed on the dataset.

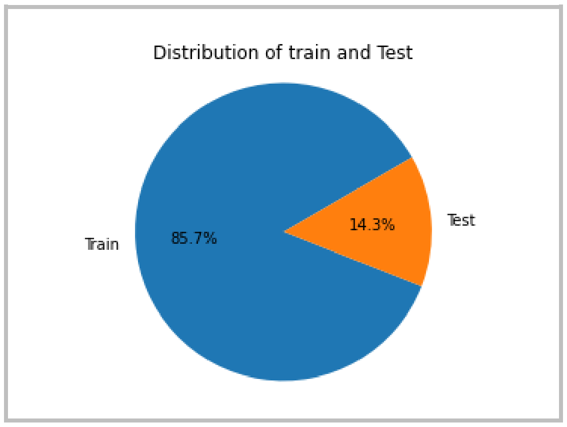

Figure 1: Percentage partition between train and test

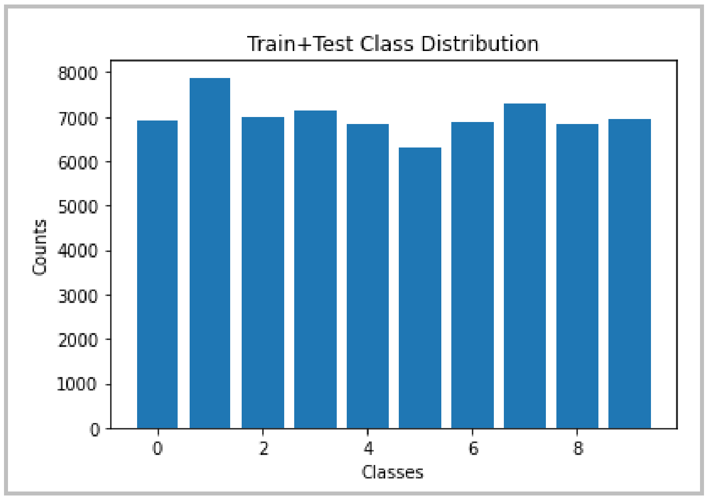

Figure 2: Total dataset partition between digits

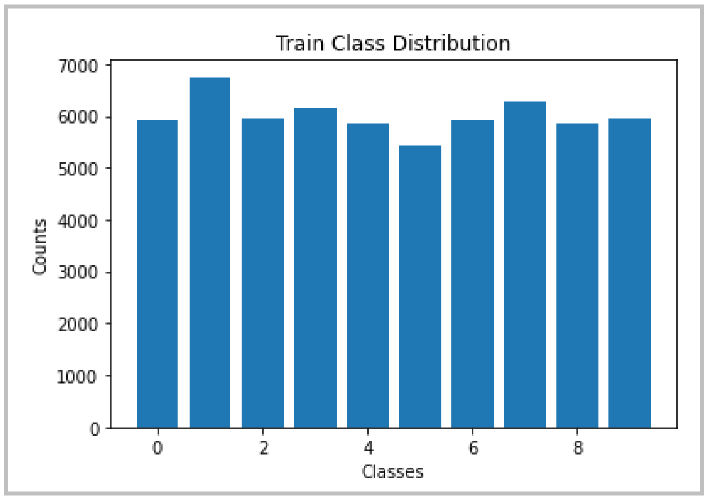

Figure 3: Train dataset partition between digits

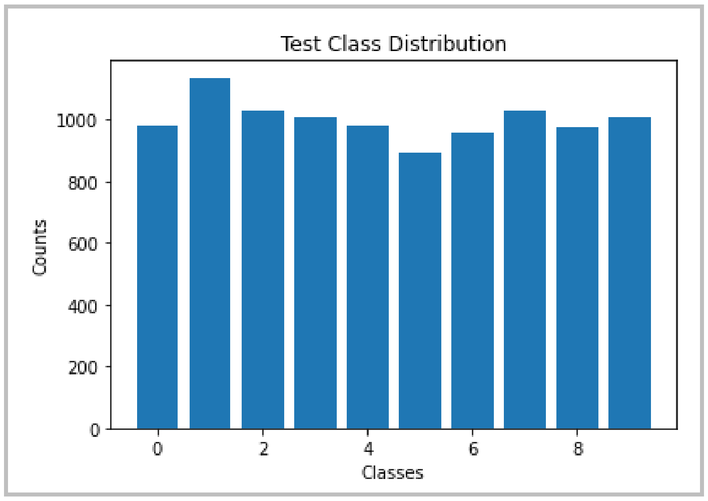

Figure 4: Test dataset partition between digits

As we can see, the data is fairly partitioned between train and test, and it is also uniformly spread across digits, which contributed to the model's success.

3.2 Pre-processing

The pre-processing that we did is quite simple and we are going to explain it now further step by step: we first started by normalizing the input matrices. We did that by applying Max-min normalization to scale the values between 0 and 1. This is a common pre-processing technique used in machine learning to ensure that the data is on a similar scale, which can improve the model's performance. Next, we appended a bias term to the input data. In machine learning, a bias term is a value added to the input data to adjust the model's output. This also helps bettering the performance of the model. Next, for the logistic regression part, we transformed the labels into binary values of 1 and 0. This is because logistic Regression is a binary classification problem, and the labels need to be in a binary format for the model to be able to predict the correct output.

For the multiclass softmax part, we did one-hot encoding for the labels. One-hot encoding is a technique used to convert categorical variables into a numerical format that a machine-learning model can use. In this case, we converted the labels from a single integer value to a vector of binary values, where each vector element represents one of the possible classes (only being 1 if it is that class else 0).

In order to improve the performance of our model and make it more robust to the underlying data distribution, we employed a number of techniques, including cross-validation, mini-batching, and PCA.

For PCA, we previously explained the gist of the technique and how it helped us in reducing the dimensionality of the data we have, but we just want to mention that we used sklearn's library to implement it since it was much faster than our grassroot implementation. In addition to that, this library contains several functions that made getting the principal components, projecting the data on them and reconstructing the original data back much simpler.

Cross-validation is a technique used to evaluate the model's performance on a variety of different data subsets. We do that by dividing the data into a number of subsets or "folds." Each fold is used as a validation set, and the remaining data is used as the training set. This technique usually helps with preventing overfitting but has memory and time complexity drawbacks, but since that dataset is relatively small, we were able to implement it without remarkable cons.

Mini-batching is a technique used to speed up the training process by breaking the data into small batches. Instead of processing the entire dataset at once, the model processes the data in small batches, which reduces the amount of memory required and speeds up the training process.

Lastly, we shuffled the data to make sure that the data will still be representative of the whole distribution even after dividing it into "k" folds for cross-validation and then mini-batches for training via gradient descent. This is very important because it helps prevent the model from overfitting to a specific subset of the data and ensures that the model can generalize well to new data.

4 PCA

As requested from us, we will now show four figures, showing the results of our PCA implementation, and we are going to discuss the findings. The first image in the batch shows the output of reconstructed images using different number of PCs from 2 till 784.

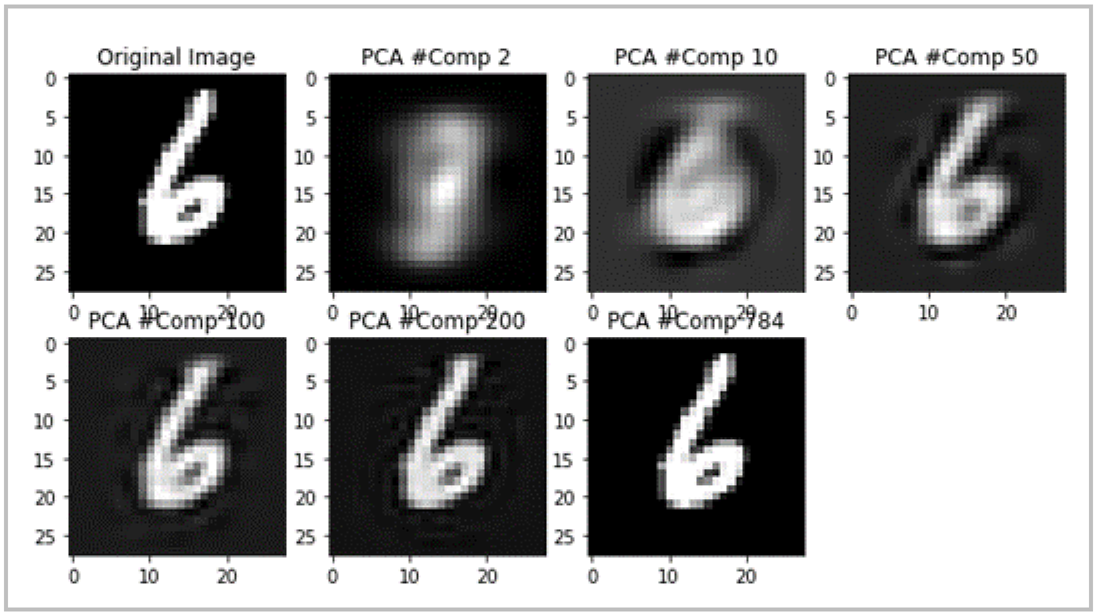

Figure 5: Reconstruction of images on different top p components

As we can see, the higher the number of PCs the more information the reconstructed image contains about the original image. For p = 2, it was blurry and indistinguishable, and then all the way to 784 where the reconstructed image is the same as the original one because it has all the principal components of it.

For the next three images, they are depicting the shape of the weights, one for 2s vs 7s, one for 5s vs 8s and the last for all classes.

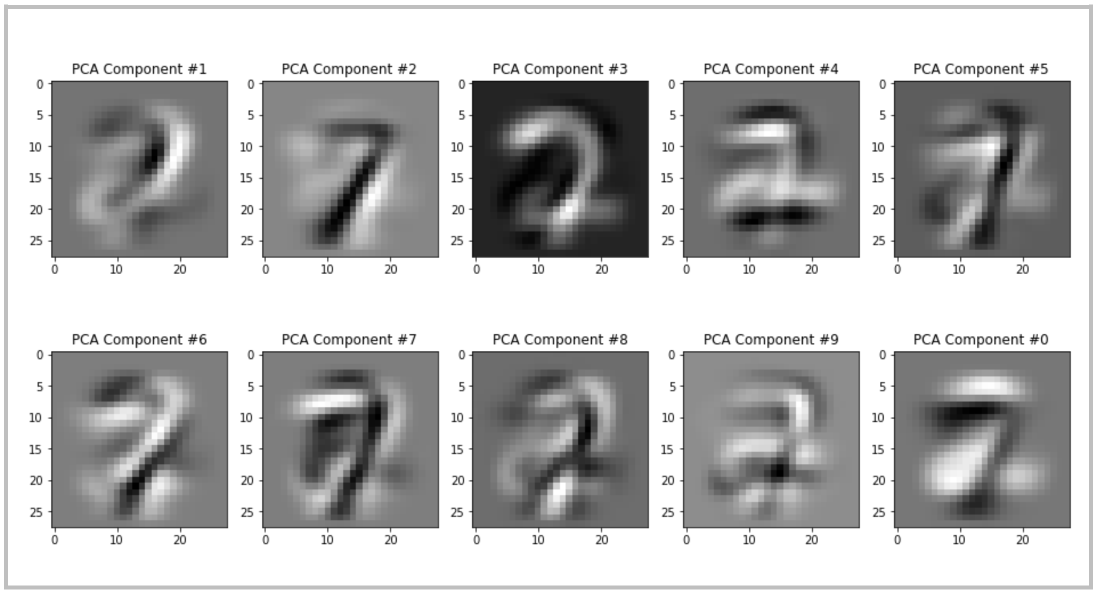

Figure 6: Top 10 PCs for 2s vs 7s task

We can see that the weights converged to either a 2 or a 7 shape which is how the weights later on determine to which category the inputted image it belongs. The same effect is also seen for 5s vs 8s as well.

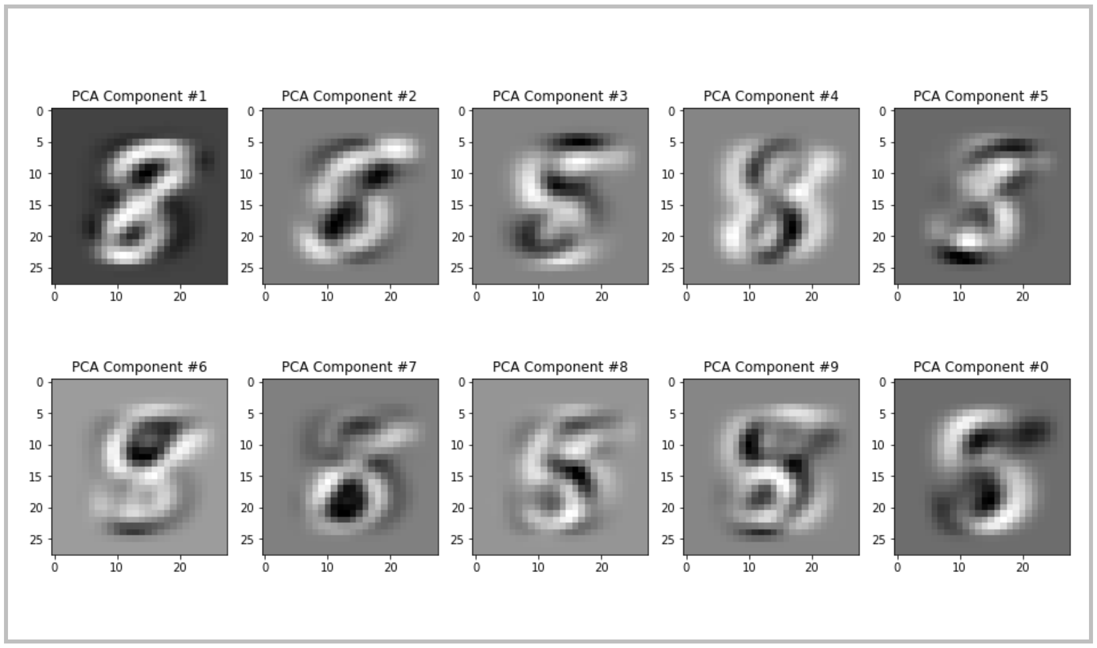

Figure 7: Top 10 PCs for 5s vs 8s task

Here also the weights took the shape of a 5 or an 8. This phenomenon was also present when we used the whole dataset but this time something different happened. Since we chose the 10 top PCs and there are 10 total classes, each weight took the shape of one output class (from 0 to 9). This can be seen in the image below as well. Now some images are more visible than others but we can see how each component focused on a specific part that is different from the other ones and that is what helps categorizing new data.

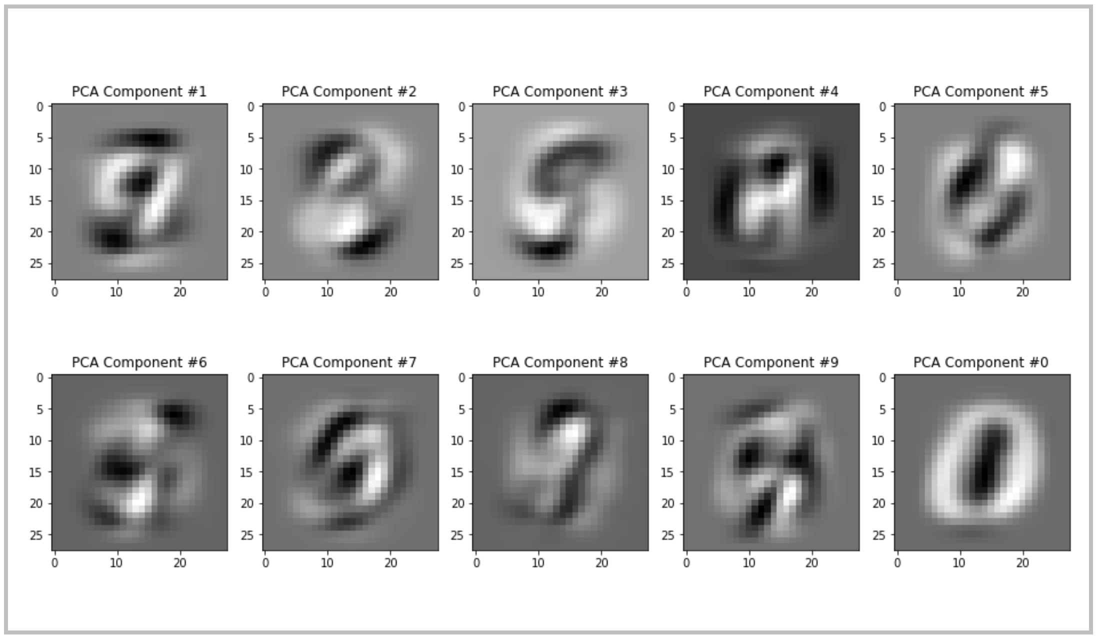

Figure 8: Top 10 PCs for all dataset

5 Logistic Regression

Logistic Regression is a statistical machine learning method that is used for binary classification tasks. Here, we applied logistic Regression to the MNIST dataset to classify images as 7 or 3 in the first part and 5 or 8 in the second part and compared the accuracies of these two examples. The model uses as input 28x28 images, and a set of weights to calculate the probability of the image belonging to one of the classes using the sigmoid activation function defined as follows:

$$p(z) = 1 / (1 + e^{-(W^T * x)})$$

Where W is the weight matrix. The output of the activation function, ie the class with the highest probability is chosen as the model's prediction. The weights are learned during training using mini-batch gradient descent which minimized the loss function which is in our case the binary cross-entropy loss function, which is defined as follows:

$$Loss = -(t\log(y) + (1-t)\log(1-y))$$

Where t is the true label and y is the predicted probability.

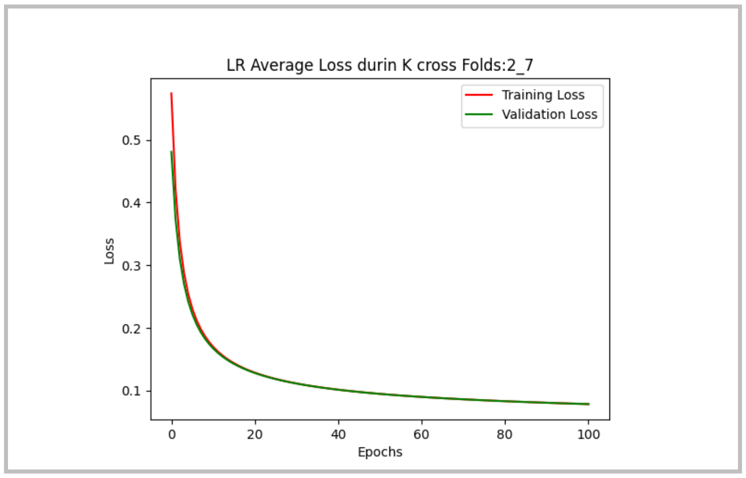

Figure 9: Logistic Regression average loss for 2s vs 7s [train and valid]

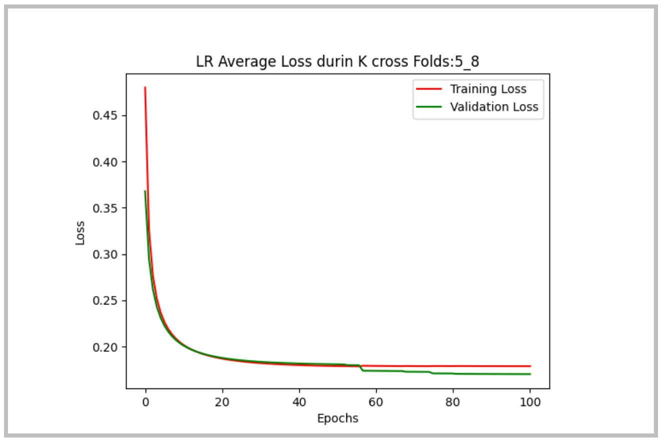

Figure 10: Logistic Regression average loss for 5s vs 8s [train and valid]

As we can see, both models are very accurate, and yielded impressive results. Both training and testing losses for both tasks converged smoothly to the minimum. The 2s vs 7s beat the second one in loss and accuracy by quite a bit, for the same reasons we mentioned above throughout this paper.

For testing we got an accuracy of 96.5% and a loss of 0.095 for the first task and a 93.4% accuracy and a loss of 0.2 for the second task.

All the results shown here are based on these parameters that we got after trying several different ones, p = 30, batch size = 64, no epochs = 100, learning rate = 0.001 and k folds = 10.

6 Softmax Regression

Softmax regression, also known as multinomial logistic Regression, is a statistical method that is used for multiclass classification problems. In this task, we applied it to the MNIST dataset of handwritten digits to classify images into 10 classes representing digits from 0 to 9.The model uses the input features, in this case, the 784 pixels of each 28x28 image and bias, and a set of weights to calculate the probability of the image belonging to each of the classes. The class with the highest probability is chosen as the model's prediction.

The weights are learned during the training process the same algorithm used for logistic Regression but this time to minimize a different object function which is the multiclass cross-entropy loss function, which is defined as:

Loss = - ∑(t * ln(y))

The activation function is softmax activation function which is defined as follows:

f(x) = e^(x) / ∑(e^(x))

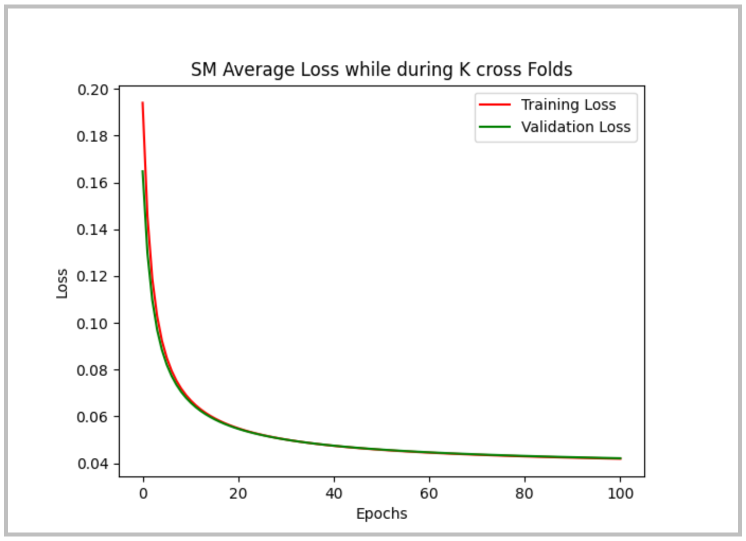

Figure 11: Softmax Regression average loss [train and valid]

Softmax algorithm also performed really well, we can see the convergence of the training and testing losses to 0.05. For testing, when we used the best weights we got an accuracy of 96.5% and a testing loss of 0.053. We used the same hyperparameters as stated above. p = 30, batch size = 64, no epochs = 100, learning rate = 0.001 and k folds = 10.

Additionally, we also plotted the weights of the softmax output layer, and we clearly saw how each weight converged to a single digit that it focused on, which was fascinating to see.

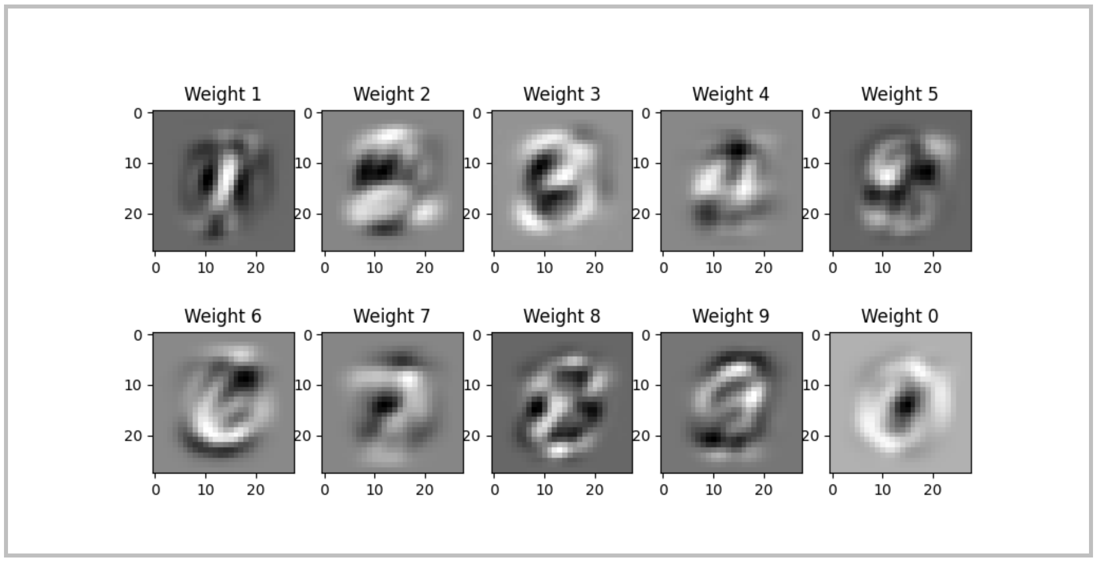

Figure 12: Weights of Softmax output layer.

7 Team contribution

We divided the project equally in all aspects. In fact, we are roommates so we did the whole project part together. For programming, it was pair programming with both of us on discord one guy sharing his screen and coding on a shared google colab notebook , alternating this role between different sections of this assignment. Jay took the lead for the logistic regression part (from loss function, activation function and gradient descent), Andrew took the lead for the Softmax function (same thing here, loss function, activation function and optimization algorithm.) In addition to that, Jay implemented the PCA algorithm and Andrew implemented the cross validation and mini batch integration that was common for both logreg and softmax. For the project report, it was pretty straightforward to do, since we had done everything together, we split up the writing in half, each one of us took the lead in writing the sections that he took the lead in coding.

8 Source code

COLAB LINK (driver code):
https://colab.research.google.com/drive/1QZcD2DC8U7Jabe1sDMofadr_AkNVkSgX?usp=sharing

DRIVE LINK (has all the main/network code):
https://drive.google.com/drive/folders/162ql8iESDd_UQU2N99lAxXaf6XnzpWPz?usp=share_link

README LINK (Guide to get started):
https://drive.google.com/file/d/1QTPyDNgrHA1is_77JObZCIJ7bcMeQV6J/view?usp=sharing

References

[1]. Y. Lecun, L. Bottou, Y. Bengio and P. Haffner, "Gradient-based learning applied to document recognition," in Proceedings of the IEEE, vol. 86, no. 11, pp. 2278-2324, Nov. 1998, doi: 10.1109/5.726791.

[2]. Patrikar, S. (2021, December 12). Batch, Mini Batch & Stochastic Gradient Descent - Towards Data Science. Medium. https://towardsdatascience.com/batch-mini-batch-stochastic-gradient-descent-7a62ecba642a