Logistic and Softmax Regression for classifying MNIST dataset

Nishanth Rachakonda
Department of Computer Science
University of California, San Diego
nrachakonda@ucsd.edu

Soumya Ganguly
Department of Mathematics
University of California, San Diego
s1gangul@ucsd.edu

Abstract

We use logistic and softmax regression to classify handwritten '2s and 7s', '5s and 8s', and all the digits from 0 to 9 respectively, using the popular MNIST dataset. The models give test accuracy of above 97 % for logistic regression and above 92% for softmax regression. We also investigate the visual representation of the principal components used for dimensionality reduction and the final weight vector obtained for softmax regression.

1 Introduction

Deep Learning models have been used extensively used for modelling complex real world problems effectively. One such problem is classification of images. MNIST is a huge dataset which contains handwritten digits from 0 to 9. We classify the MNIST dataset effectively using simple regression models.

Through this paper, we show how to classify images of handwritten digits in different ways using logistic regression and softmax classifier. Logistic regression is normally used for 'binary classification', here we classify the handwritten digits '2 and 7' and '5 and 8'. The softmax classifier is generally used for multi-class classification and here it is used to classify all the handwritten digits from 0-9. Furthermore, the input images are reduced to lower dimensions to decrease the size of models needed to be trained.

It can be understood that these simple regression models generate great results. We were able to produce a test accuracy of over 90% for all the classification problems.

2 Related Work

In the lecture slides of CSE 251B by Cottrell [1], the theory behind logistic and softmax regression is discussed which creates the foundation of the work done in this paper. In [2], the same author has introduced cross-entropy which is our objective function here. The theory and techniques for performing principal component analysis (PCA) are discussed in [3].

3 Dataset

3.1 A brief description of data

Modified National Institute of Standards and Technology database, MNIST, is a handwritten image dataset of digits 0 to 9. This is a popular dataset used frequently for training multiple machine learning models. It is a huge dataset with 60,000 training images and 10,000 test images.

Some randomly sampled images of each handwritten digit class are shown in Figure 1.

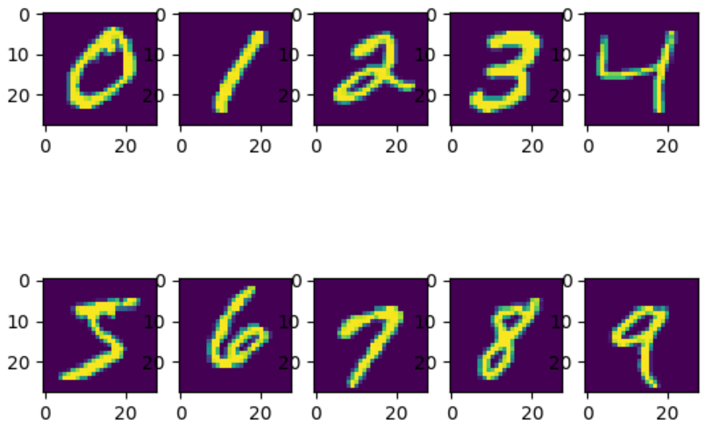

Figure 1: Random examples of different image classes in MNIST.

3.2 Pre-processing on the dataset

Before working on the data, we have to perform pre-processing on the dataset for getting good results on classification.

- **Shuffling**: Randomizing/Shuffling the dataset creates a homogeneous representation of the overall data in the minibatches. This allows for the model to look at a balanced subset of the data in each iteration and prevents it from over-fitting.
- **Normalizing**: Normalizing the data makes it more cohesive and easy to learn. We normalize the data using z-score-normalization where we subtract the mean from data and divide the result by the standard deviation which results in a more 'regular' looking plot for the loss functions.
- **One-Hot Encoding**: To compute the multi-class cross entropy in softmax regression, the target value is stretched into a target vector, $\vec{t}$, in $\mathbb{R}^c$ where c is the number of classes in the dataset, such that the $t_i = 1$ if i is the label value and $t_i = 0$ otherwise.
- **Appending Bias**: The image feature vectors are stacked by 1 at the end to seamlessly compute the addition of bias component while multiplying the feature vectors with weights. Hence, the dataset in $\mathbb{R}^{N \times d}$ is transformed to a dataset in $\mathbb{R}^{N \times (d+1)}$.

3.3 Statistics for the different splits of the dataset

The MNIST dataset has 60000 training samples and 10000 test samples. Each sample is a grayscale image of size 28 X 28. MNIST dataset has 10 classes representing handwritten digits from 0 to 9. We divide the training samples into k=10 folds. The training folds have total 54000 samples in them and each validation fold has 6000 samples.

From table 1, it can be observed that the handwritten digits are almost equally distributed. However, the classes 5, 4 and 8 are the least frequent classes and classes 1, 3 and 7 are the most frequent classes in the dataset.

Table 1: Class distribution in the dataset

Digit Class	Train (%)	Test(%)
0	9.87	9.8
1	11.24	11.35
2	9.93	10.32
3	10.22	10.1
4	9.74	9.82
5	9.04	8.92
6	9.86	9.58
7	10.44	10.28
8	9.75	9.74
9	9.92	10.09

4 PCA: Principal Component Analysis

MNIST dataset has grayscaled images of size 28×28, which is high dimensional ($784 = 28^2$). For efficient computation and memory use, we reduce the dataset dimensions using Principal Component Analysis (PCA). Principal components are the eigenvectors of the covariance matrix for a mean centered data. The significance of these eigen-vectors is determined by the magnitude of the corresponding eigenvalues. Then the dataset is projected onto the p most significant principal components. The number of principal components (p) is a hyperparameter which is to be determined based on the average validation accuracy.

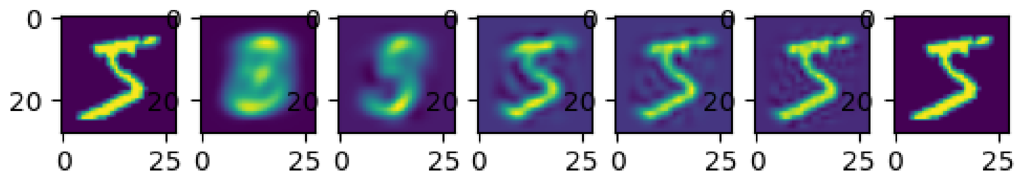

Figure 2: Image reconstruction using PCA

In Figure 2, a randomly sampled handwritten digit is reconstructed using p most significant principal components. The leftmost image is a randomly sampled image and then the reconstructed images using PCA follow. In the reconstructed images, p varies in ascending order from the set $\{2, 10, 50, 100, 200, 784\}$ as we go from left to right. It can be observed that with the increase in the principal components ('p'), the reconstructed image resembles the original image more closely. As the number of principal components increase, the reconstructed image retains more data from the original image. The rightmost image is reconstructed after retaining all the principal components (784) hence it exactly matches the original (leftmost) image. So in other words, loss of information is more when lesser number of principal components are taken.

4.1 Top 10 Principal Components (PCs) for different classes

In the following figures (Figure 3,4,5) we show the top 10 principal components for all the 3 classes concerned in this project i.e. 1)7s and 2s, 2) 8s and 5s and 3) all 10 classes, respectively:

We observe that for binary classification of classes 2 and 7, classes 5 and 8, the top 10 principal components resemble the digits being classified. For the 10-class classification, the principal components resemble key strokes of the handwritten digits, for example 0, 3,9,6 etc. We see the first

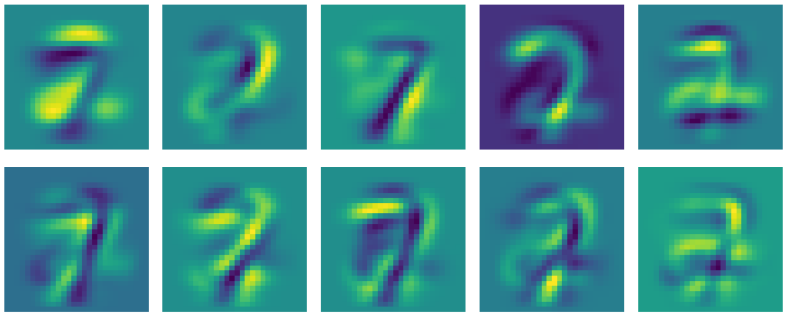

Figure 3: Top 10 PCA for 7s and 2s

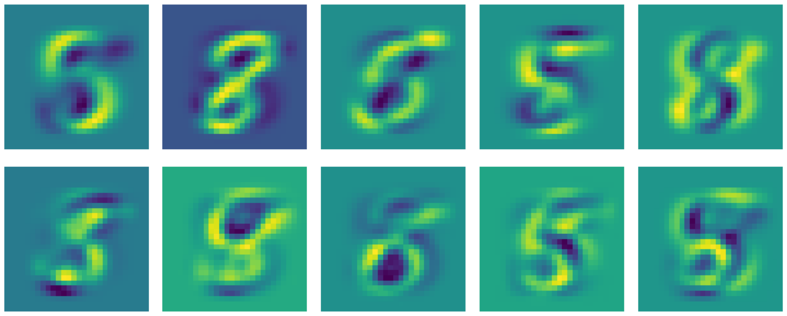

Figure 4: Top 10 PCA for 8s and 5s

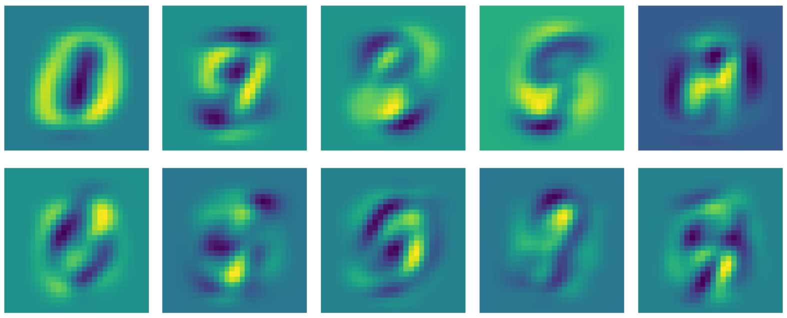

Figure 5: Top 10 PCA for all 10 classes

principal component looks like 0, the first item in our handwritten-digit-list. Since principal components of images encode the key features of the images and we are calculating principal components of images of handwritten digits from 0 to 9, we can totally expect such representation of the first 10 principal components as shown here.

However we see that the top 10 PCs for classification of '2 & 7' and '5 & 8' look much closer to the digits we are trying to classify than the top 10 PCs of all 10 handwritten digits. The PCs of images of digits '2 & 7' and '5 & 8' need to capture less number of feature whereas the top 10 PCs need to capture more no. of features, in same number of components- thus leading to less prominent images of digits in them.

5 Logistic Regression

5.1 Logistic Regression Model

Logistic regression is a binary classification method, where the output of the network is $y = g(a)$ where g is called the activation function given by the logistic function as follows:

$$g(a) = \frac{1}{1 + e^{-a}}$$

Here, a is the weighted sum of the inputs or the net input $a = \sum_{j=0}^{d} w_j x_j$, where $\vec{x}$ is the input and $\vec{w}$ is the weight vector, such that w_0 is the 'bias' and $x_0 = 1$. If $y > 0.5$, $\vec{x}$ is classified as 1 and 0 otherwise.

The weight vector $\vec{w}$ is chosen by performing gradient descent to minimize the loss function given by the following cross-entropy cost function, over training the set:

$$E(w) = -\sum_{n=1}^{N} \{t^n \ln(y^n) + (1 - t^n) \ln(1 - y^n)\}$$

Here N is the number of training samples, $t^n \in \{0, 1\}$ is the label for the sample n where $t^n = 1$ signifies that x^n is in class 1 and $t^n = 0$ signifies that x^n is in class 0. y^n is the predicted output which is computed as $g(x^n)$.

5.2 Results

5.2.1 Plots of training and validation loss

Hyperparameter tuning is performed using the grid search algorithm over various hyperparameter values, which will be discussed in 6.1.3. Based on these experiments, we observe:

- The optimal set of hyperparameters for classifying the handwritten digits 2 and 7 will be: No. of principal components= 50, Minibatch size= 128, Learning rate = 0.1.
- The optimal set of hyperparameters for classifying the handwritten digits 5 and 8 will be: No. of principal components= 100, Minibatch size= 16, Learning rate = 0.001.

The plots of training and validation loss can be found in Figure 6 and Figure 7, which are done for these optimal sets of hyperparameters, in case of each classification.

We see, for the classification of 2 and 7 above, the model starts overfitting at around 40 epochs as the validation loss starts to increase in value. So in this case we would need early-stopping. However, for classification of 5 and 8, we see from the plots that the model is not overfitting within 100 epochs, which requires no early stopping.

5.2.2 Test performance

The model, after choosing the most optimal set of hyperparameters and weight vector, produces a test accuracy of 97.85% for classification of classes 2 and 7, and the model produces a test accuracy of 95.80% for classification of classes 5 and 8. The test accuracy of classification of classes 5 and 8 is lower than the classification of classes 2 and 7. This is probably because the key strokes of digits 5 and 8 match each other more closely than the key strokes of digits 2 and 7.

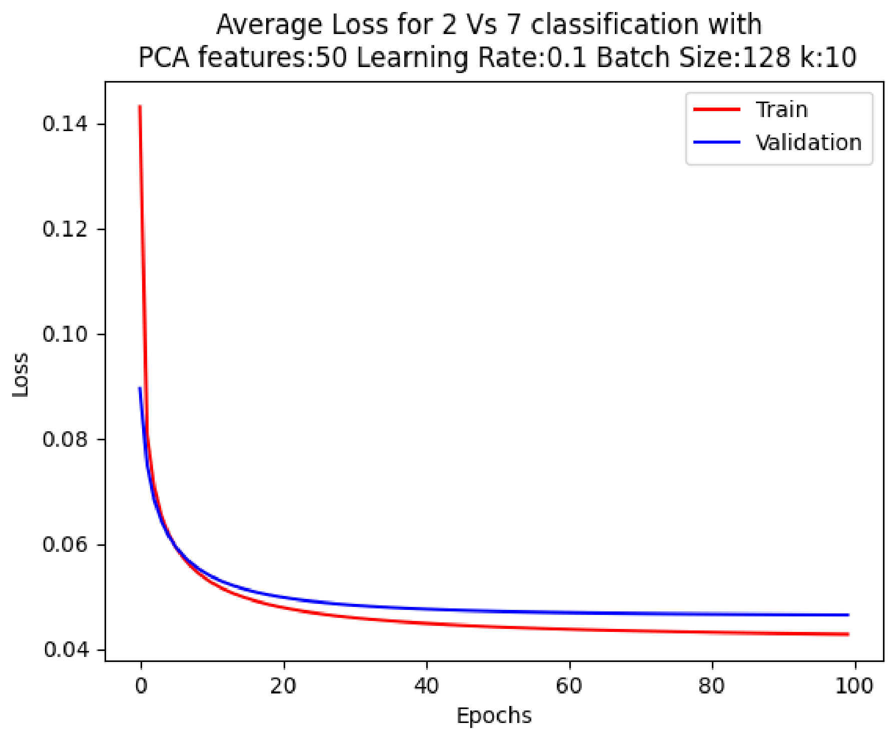

Figure 6: Loss curve for classifying classes 2 and 7

5.2.3 Discussion of the results

The plots given above for training and validation losses are for the hyperparameter combinations (i.e. no. of principal components (p), minibatch size, learning rate) that gave us the best accuracy when tested on the validation sets. These optimal sets were obtained when we performed hyperparameter tuning using grid search on several possible combinations of hyperparameters. We show some of the results of the grid search (which give us some of the highest and lowest validation accuracies) in the tables here. The optimal hyperparameter combination was chosen based on the highest average validation accuracy, which are also given in the tables.

Table 2: Table of different combinations of hyperparameters tested for classifying 2s and 7s

NO. OF PCs	MINIBATCH SIZE	LEARNING RATE	AVG. VALIDATION ACCURACY
50	128	0.1	0.9939236111111112
100	32	0.1	0.993421052631579
100	16	0.01	0.9925986842105263
50	16	0.1	0.9925986842105263
100	256	0.1	0.9921875
5	128	0.01	0.9704861111111112
5	256	0.001	0.9697265625
5	256	0.01	0.9697265625
10	128	0.001	0.9696180555555556
10	256	0.001	0.966796875

It can be seen that the top row of last table gives the best hyperparameter combination to be used for our model. From the table above, we can also observe that the higher values of validation accuracy

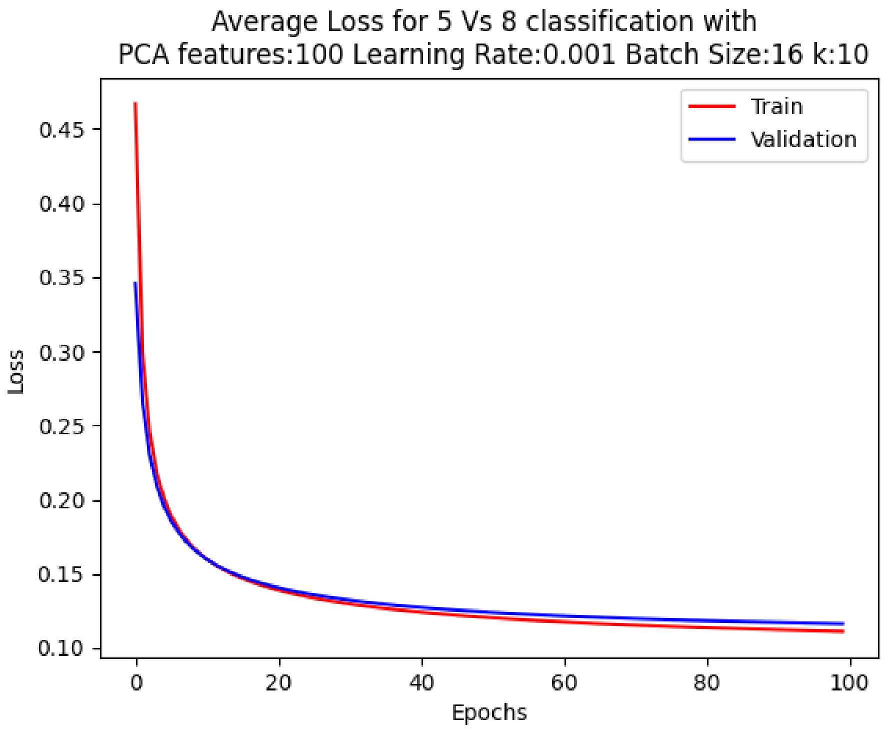

Figure 7: Loss curve for classifying classes 5 and 8

occur when no. of principal components are on the higher side (50 or 100) and learning rate is decent (0.1 and 0.01). As learning rate goes down (to 0.001), the validation accuracy also decreases. More no. of principal components encode more information about the image, so the model learns more and hence the behavior above is expected. We may also observe that all the rows above which have lower validation accuracy have batch size in the higher range.

Table 3: Table of different combinations of hyperparameters tested for classifying 5s and 8s

NO. OF PCs	**MINIBATCH SIZE**	**LEARNING RATE**	**AVG. VALIDATION ACCURACY**
100	16	0.001	0.9741071428571428
100	128	0.1	0.9736328125
100	256	0.1	0.97265625
100	32	0.01	0.9714285714285714
100	64	0.1	0.9705882352941176
10	64	0.001	0.9329044117647058
5	64	0.001	0.9310661764705882
10	256	0.001	0.9296875
5	256	0.001	0.927734375
5	128	0.001	0.927734375

Similar to the classification of classes 2 and 7, it can be observed that with the increase in the number of principal components and higher learning rate the validation accuracy increases. However, as mentioned in 5.2.2 the validation accuracy of classifying digits 5 and 8 is in general lower than that in the classification of digits 2 and 7.

6 Softmax Regression

6.1 Softmax Regression Model

Softmax regression is used for multi-class classification where the output of the network is a c dimensional vector $\vec{y}$ with c being the number of classes we are classifying the input into. We have $y_k = g(a_k)$ for $k \in \{1, 2, ...c\}$, where g is the activation function given as follows:

$$g(a_k) = \frac{e^{a_k}}{\sum_{j=1}^{c} e^{a_j}}$$

Here, $a_k = \sum_{j=0}^{d} w_{jk} x_j = w_k^T \vec{x}$ with $x_0 = 1$ and w_0 being the bias as before. Now, in this case, we have:

$$\text{Probability of the input being in class } k = P(C_k|\vec{x}) = y_k$$

So for a given input $\vec{x}$, the class for which we have the highest probability $P(C_k|\vec{x})$- becomes the label on that input.

Similar to logistic regression above, the weight vector $\vec{w}$ is chosen by performing gradient descent to minimize the loss function given by the Cross-entropy ('multi-class' this time) cost function, over training examples, which is shown below:

$$E(w) = -\sum_{n=1}^{N} \sum_{i=1}^{c} \{t_i^n \ln{(y_i^n)}\}$$

Here, N is the number of training samples, $\vec{t^n}$ is the one-hot-encoding of the label on the training sample no. n (defined above).

6.1.1 Plots of training and validation loss

Hyperparameter tuning is performed using the grid search algorithm over various hyperparameter combinations, just like we did for logisitic regression above. Based on these experiments, we observe: The optimal set of hyperparameters for classifying all the handwritten digits will be: No. of principal components= 100 , Minibatch size= 32, Learning rate = 0.01.

The plots of training and validation loss are done for these optimal sets of hyperparameters only.

We see, for the softmax classification, the model starts overfitting at around 50 epochs as the validation loss starts to increase in the plot above.

6.1.2 Test performance

After choosing the best set of hyperparameters and optimizing our weight matrix for this set, out model produces a test accuracy of 92.09% for softmax classification of all the handwritten digits from 0-9.

6.1.3 Discussion of the results

The plots given above for training and validation losses are for the hyperparameter combinations (i.e. no. of principal components (p), minibatch size, learning rate) that gave us the best accuracy when tested on the validation sets. These optimal sets were obtained when we performed hyperparameter tuning using grid search on several possible combinations of hyperparameters. We show some of the results of the grid search (which give us some of the highest and lowest validation accuracies) in the table below. The optimal hyperparameter combination was chosen based on the highest average validation accuracy, which are also given in the table.

It can be seen that the top row of last table gives the best hyperparameter combination to be used for our model. From the table above, we can also observe that the higher values of validation accuracy occur when no. of principal components are high (100). Just like logistic regression, we can argue that more no. of principal components encode more information about the image, so the model learns more and hence the behavior above is expected. All the lowest validation accuracies in the

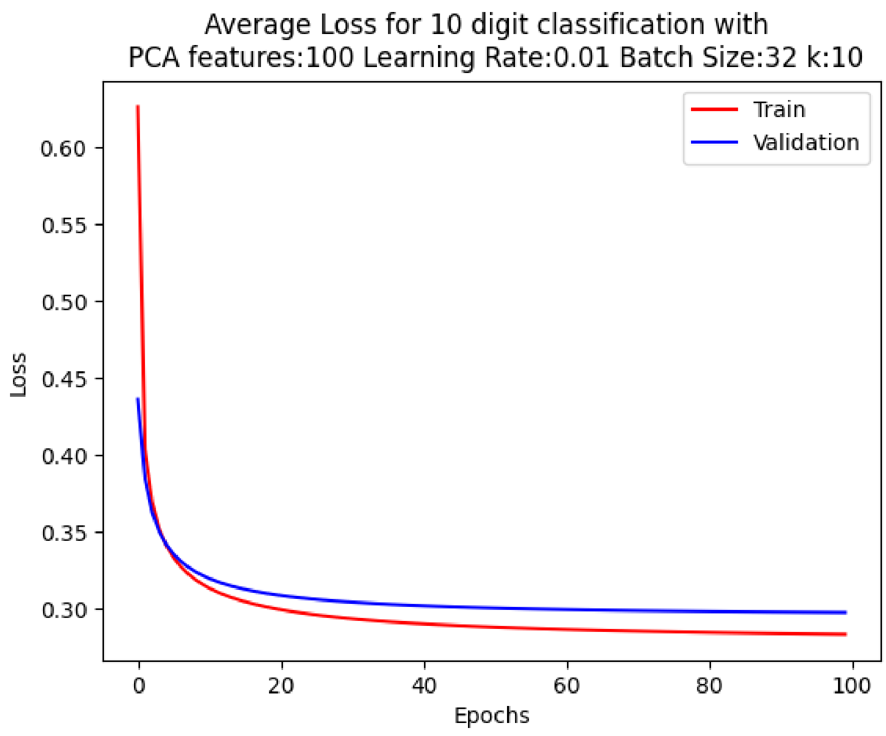

Figure 8: Loss curve for classifying classes 2 and 7

Table 4: Table of different combinations of hyperparameters tested for classifying 0-9

NO. OF PCs	MINIBATCH SIZE	LEARNING RATE	AVG. VALIDATION ACCURACY
100	32	0.01	0.9239639037433155
100	64	0.01	0.9232190860215054
100	64	0.1	0.9232190860215054
100	16	0.01	0.923
100	128	0.1	0.9228940217391305
5	128	0.01	0.6786684782608695
5	32	0.001	0.6721256684491979
5	64	0.001	0.6616263440860215
5	128	0.001	0.6489470108695652
5	256	0.001	0.6307744565217391

table above have very less number of principal components (5) and learning rates are considerably low there as well.

Visualization of the final weights:

After getting the optimal set of hyperparameters, we record the the final weights for our softmax model. The weights are then unnormalized using the mean and standard deviation of the train data set for visualizing it. The final, unnormalized weights can be found out as images in Figure 9 below.

From Figure 9, we can see that the final weight vectors look like the digits 0-9 that we are trying to classify, using our softmax model. This means our model is successfully learning the representation of the keystrokes of the digits.

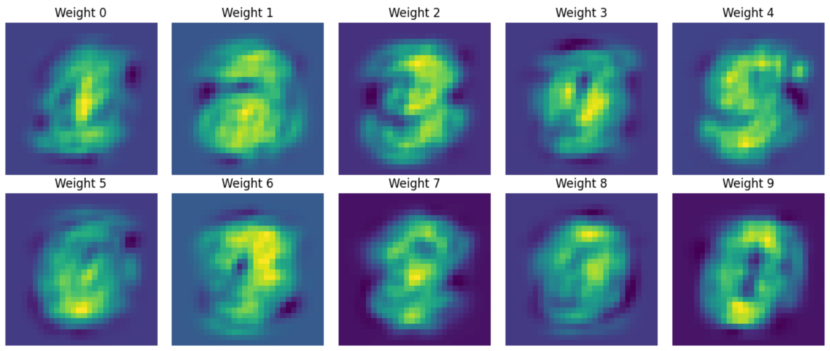

Figure 9: Reconstructed Softmax Regression Weight Vectors

7 Team Contributions

The project is a result of a team effort where both of us contributed as a group. However, roughly the contribution of each team member is listed below:

- Nishanth primarily worked on implementing the loss functions, activation functions, network object and training loops. He also wrote the abstract, introduction and description of dataset in the project report along with generating the necessary plots and tables.
- Soumya primarily worked on implementing data pre-processing, and grid search algorithm. He implemented algorithm for identifying optimal hyperparameters. He was also in charge of writing about PCA, the models of regressions being used, discussion of results etc.

8 References

[1] Cottrell, G.W. (2023) Logistic Regression Multinomial Regression *CSE 251B lecture slides*: 11-41.

[2] Cottrell, G.W. (2023) Maximum Likelihood: The theoretical basis for our objective functions. *CSE 251B lecture slides*: 34-49.

[3] Cottrell, G.W (2023) PA1: Logistic and Softmax Regression *CSE 251B discussion notes*: 15-20.