# PA1 Report: Implementation and Evaluation of Logistic and Softmax Regression Classifier on various Human Facial Expressions

**Huimin Zeng**
Computer Science & Engineering
UC San Diego
La Jolla, CA 92093
*zenghuimin@yahoo.com*

**Zhenrui Yue**
Computer Science & Engineering
UC San Diego
La Jolla, CA 92093
*yuezrhb@gmail.com*

## Abstract

This document is the report of the first programming assignment (PA1) in CSE 253 Neural Networks/Pattern Recognition at UC San Diego in Winter 2020 by Huimin Zeng and Zhenrui Yue. The report is based on the requirements of the problems in the assignment and contains four sections for the three programming problems and the individual contributions of each team member in this group.

In this report, we first described the provided dataset and its preprocessing: Extended Cohn-Kanade Dataset (CK+), which consists of various pictures of human faces with different emotions. With preprocessed data, we implemented the cross-validation procedure to avoid overfitting, then, Principal Components Analysis (PCA) was applied to the dataset to reduce the dimensions of input data. In the following sections, we first developed a logistic regression classifier with Batch Gradient Descent and Stochastic Gradient Descent, the classifier was trained with the processed facial expressions and its performance on the dataset was evaluated and visualized in the report, our best performance on the Happiness vs Anger with logistic regression is 97.22%. A softmax regression classifier using the softmax function was also implemented to classify all given emotions: anger, disgust, fear, happiness, sadness and surprise, the section also evaluates the softmax classifier, two different updating methods and represents its performances visually, the best performance we had on all emotions with softmax regression is 66.67%. Finally, the last section addresses the individual contributions to this programming assignment.

## 1 Data Preprocessing, Cross-Validation and PCA

The first section contains three subsections: the dataset preprocessing and two basic methods adopted in this assignment: Cross-Validation and Principal Component Analysis.

### 1.1 Data Preprocessing (1)

The Extended Cohn-Kanade Dataset (CK+) was introduced in the first assignment to train and evaluate our regression model, CK+ was released by various researchers from Disney Research in 2010 based on Cohn-Kanade (CK) database in 2010. The dataset contains pictures of human facial expressions that represent visual information of seven emotions: angry, disgust, fear, happy, sadness, surprise and contempt. CK+ was tested and validated for the purpose of detecting different emotions, it also provides two baseline models and their performances using respectively Active Appearance Model and linear support vector machine.

The image data from CL+ was provided in both resized and aligned form from the assignment files and should be first preprocessed by the data loader given in the assignment code. The aligned data are processed images of aligned human faces in all seven emotions whereas the resized folder has resized images only available in two emotions: happiness and anger. The assignment code provides a few functions that automatically loads images from either aligned or resized folder and returns a dictionary with the emotions as keys and a list of images as values, the code also provides function that creates a balanced image set for the specified emotions and function that help to display an image based on matrix values.

### 1.2 Cross-Validation (2)

Before the data could be trained as input of a machine learning model, we first implemented cross validation, a widely adopted validation technique that splits data into different folds, so that some of the folds could be used to train a mathematical model and the rest folds to test and validate the model. Different combinations of training and testing folds could give a better overall estimation of the model behavior in predicting unknown data and its generalization ability.

In our implementation, we first wrote a function that split the emotions and returns a list with each facial emotion and their images in multiple folds according to the given dictionary data input and fold number as arguments. Followingly, a second function was created, where the preprocessed folded image list would be divided into training, testing and validation data. Afterwards, the training data would be processed using Principal Component Analysis (PCA, which will be introduced in the next subsection), the dimensions of the image data would be reduced by projecting them to the principal components in order to simplify the image data and reduce training time. Then, three data loaders would be returned for training, testing and validation sets, each data loader consists of the given emotion classes and corresponding image data in reduced dimension. We also implemented a simplified version of cross-validation, which returns the indices of the folds in a list. Before training the model, a for loop is still needed to traverse all possible combinations, we set a fold as validation set and the next one as test set, all the rest folds would be assigned to training set. Thus, all data entries would be used at least once for validation and testing.

### 1.3 Principal Component Analysis (3)

2D Images are actually really high-dimension data, in our case, each aligned image has the resolution of 224x192, which equals 43008 features of each data point. Therefore, it is infeasible to stretch the pixel matrices into vectors and feed them into the model directly. High-dimensional data could be preprocessed by certain techniques, such as Principal Component Analysis (PCA). Here, we performed PCA, by computing the eigenvalue decomposition of the training images. We took the largest k eigenvalues as the normalization factors and the corresponding eigenvectors to form the projector matrix. Via matrix/vector multiplication, we can represent the raw high-dim images with the lower-dim principal vectors by mapping all training images as well as the validation images and the test images into a lower dimensional space, using the same projector matrix. For example, if we want to project a data point with 43008 features to a space with only 20 dimensions, we could multiply the first 20 principal components (corresponds to the first 20 columns of our projector matrix) with the data point, which will result in a vector with 20 features. The data processed with PCA carries the most variance of the originally dataset but is much smaller in size and could be trained in a significantly shorter period.

In our implementation, we applied the Turk and Pentland trick to simplify the calculation of eigenvectors (eigenfaces), we normalized the vectors and divided them by their corresponding eigenvalues. Afterwards, we output the average value of the training data (average face), before we project testing and validating data, we subtract these data by the average value and multiply the projector matrix with the centered data.

## 2 Logistic Regression (5a, see code)

We implemented logistic regression for the binary classification task. Specifically speaking,

the model firstly computes the dot product of the dimension-reduced inputs vectors and a weights vector and then feeds the scalar-valued output into a non-linear function, namely the sigmoid function, to project the outputs into a probability space, i.e. (0,1). When making decisions, the model simply compares the probabilistic outputs with a threshold and then makes the final predictions of the classification.

Well is known that it is of little efficiency to compute the Mean Squared Error (MSE) for the classification problem, since there might be relatively less information for the update step. Therefore, Binary Cross Entropy (BCE) is regarded as better loss function in the binary classification problem. Here, we firstly demonstrate the formal definition of the BCE:

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

Besides, we do not have a closed-form solution for the minimizer of this loss function. Therefore, we need to do gradient descent with the gradient of the loss function:

$$-\frac{\partial E(w)}{\partial w_j} = \sum_{n=1}^{N} (t^n - y^n) x_j^n$$

## 2.1 Happiness vs Anger using the resized dataset (5b)

After implementing the logistic regression model, the resized data was first tested on the model to test the performance of the model on unaligned facial expressions. Here we didn't apply the cross-validation and only split the data into three parts, training, testing and validation set with respectively 80%, 10% and 10% of the original data amount. We set the number of epochs to 100 and learning rate to 0.1, which results in a 68.06% of accuracy in training data and an identical 66.67% accuracy in both testing and validation set. The following figure shows the development of all three error values with increasing epochs, note that the testing and validation error start to rise after only 20 epochs, which is a sign of our model overfitting the training data.

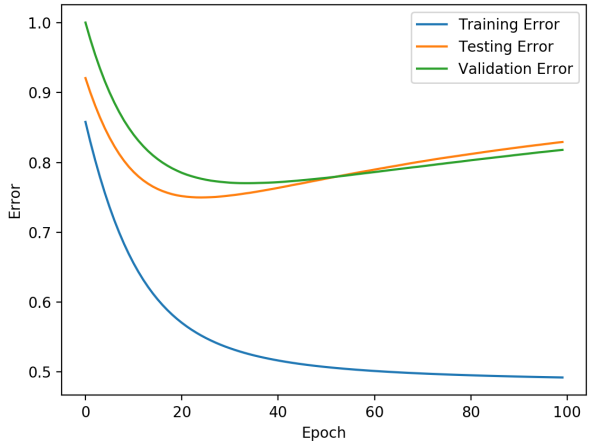


Figure 1: Error values on Happiness vs Anger (Resized) in 100 Epochs

Figure 2: First four Principal Components (Resized) as images

## 2.2 Happiness vs Anger on the aligned dataset (5c)

In this part, we start to use the aligned facial expression dataset, which contains all 6 emotions in the format of 224 x 192 resolution images. We repeated the test from previous section on the aligned dataset and applied cross-validation with 10 folds, epoch number and learning rate stay the same. After adjusting the code and training the model, we achieved an average of 99.58% of training accuracy, 97.78% validation accuracy and 97.22% accuracy in the testing dataset. Note that the chart contains information of the standard deviations in each epoch with error bars during training (see Figure 2).

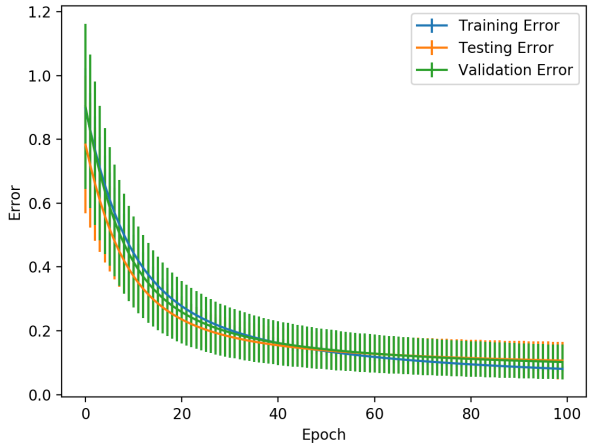


Figure 2: Error values on Happiness vs Anger (Aligned) in 100 Epochs

Then, we tried out different learning rates to test the learning behavior the logistic regression model, we used 6 different learning rates and picked 3. Beside the appropriate learning rate of 0.01 in the middle, the left chart with the learning rate of 10 represents overfitting and divergence during the learning process, as the training and testing error starting to increase after a few epochs and the standard deviation rises rapidly, whereas the chart on the right side with a much lower learning rate of 0.0001 would cause slow convergence and much longer time of the training process. This comparison implies that a proper learning rate could be crucial for logistic regression as the learning rate directly affect the convergence and training period of the model.

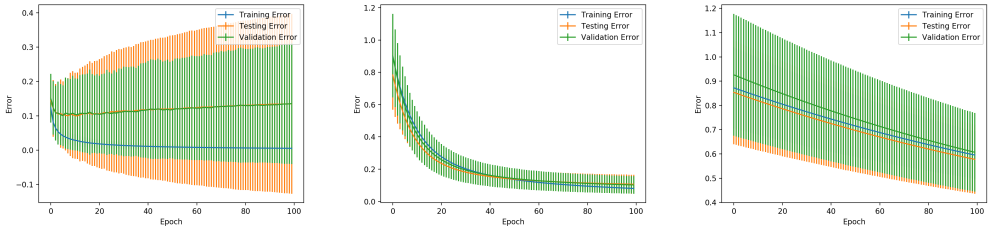


Figure 3: Error values on Happiness vs Anger (Aligned) with different learning rates

## 2.3 Fear vs Surprise on the aligned dataset (5d)

After training the model with two emotions: happiness and anger, we switched to two other emotions in the aligned dataset: fear and surprise. This is because compared the happiness and anger, the two emotions of fear and surprise could be more similar and thus, causes worse prediction accuracy in logistic regression model. So, we performed the same cross-validation training process (with 10 folds, 0.1 learning rate and 100 epochs), and the results are shown in the following figure. The accuracy value of the training set is at 96.25%, while the testing and validation accuracy are much lower, respectively 86.00% and 84.00%.

Obviously, the training process is much slower as the error curve still converging at the 100th epoch, and the values of standard deviation are much higher, the accuracy valued are also much less satisfactory compared to all accuracies of over 97% with the previous emotions of happiness and anger. This is partly because that fear and surprise facial expressions are somewhat alike and telling the difference of these two emotions is also harder than telling the difference between happy and angry, which has been proved by both of the team member after viewing all the facial images of these two combinations.

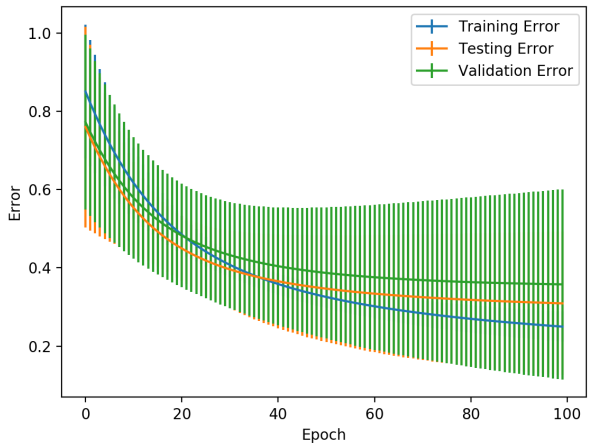


Figure 4: Error values on Fear vs Surprise (Aligned) in 100 Epochs

# 3 Softmax Regression (see code)

Logistic Regression could well perform the task of classifying two different objects, however, it's often the case that a dataset has multiple label, which is not possible for logistic classifiers. Therefore, we need a new classification model that could perform multi-class regression, which is a generalization of the logistic regression with the softmax function:

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

With $y_k^n$ representing the probability of the n data point in class k, the softmax regression model could compute probabilities of the data in all possible classes and pick the class with highest value. This cost function of softmax regression is defined with the cross entropy:

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

Taking the gradient of the above equation, we found that the update rule is basically identical to the logistic regression model. So, similarly, we implemented the softmax regression, with only differences in one-hot code function of data label, the softmax function and cross-entropy function.

## 3.1 Softmax Regression on all six emotions (6a)

After implementing the softmax regression classifier, we evaluated the model with all six emotions. It doesn't make sense of we train our model only with happy faces, this would cause lower classification accuracy on all other emotions, as the model was only trained with happiness facial expression. Thus, we trained our model with the entire aligned dataset, the following chart reveals the results of our softmax regression model with cross validation (10 folds), learning rate of 0.1 and 50 epochs. The best training accuracy of the model is 84.92%, while the testing accuracy and validation accuracy are much lower with the values of 65.33% and 68.67%.

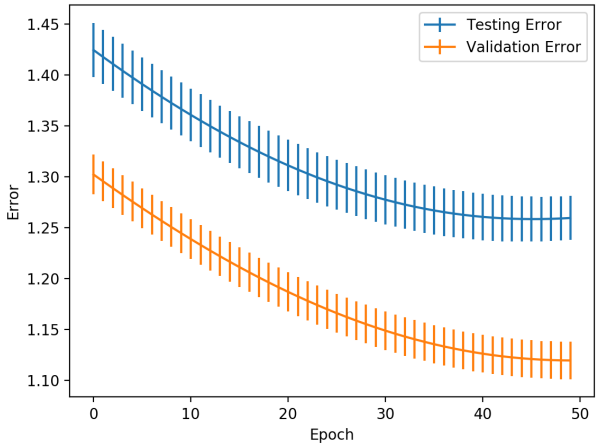


Figure 4: Error values on 6 emotions (Aligned) in 50 Epochs

We also created a 6x6 confusion matrix on the test data set, the left column represents the prediction of our model while the first row represents the true label of the data. As show in the table, the diagonal data entries are the correct predictions, where the prediction equals the true label. The other parts of the table are the fractions of different misclassified data, with a sum of approximately 35%.

| Label / Prediction | Anger | Disgust | Fear | Happiness | Sadness | Surprise |
|---|---|---|---|---|---|---|
| Anger | 0.113 | 0.02 | 0 | 0 | 0.033 | 0 |
| Disgust | 0.027 | 0.12 | 0 | 0.013 | 0.067 | 0 |
| Fear | 0.007 | 0 | 0.12 | 0.007 | 0.02 | 0.013 |
| Happiness | 0.007 | 0.007 | 0.013 | 0.14 | 0 | 0 |
| Sadness | 0.04 | 0.02 | 0.027 | 0 | 0.067 | 0.013 |
| Surprise | 0.02 | 0 | 0.007 | 0 | 0.02 | 0.12 |

Table 1: Confusion Matrix of 6 emotions (Aligned) on test data

### 3.2 Batch versus stochastic gradient descent (6b)

As mentioned above, we implemented the stochastic gradient descent training method similarly to the previous logistic regression model. Here, we compared two different method graphically with the training error, the corresponding model parameters are: learning rate of 0.1, epoch number of 50. The chart shows only limited differences in batch and stochastic gradient, the stochastic gradient descend method has a slightly faster converging rate, but after approximately 40 epochs, the loss value was surpassed by batch gradient descend method. On the other side, the batch gradient descend has though a slower converging speed, but ultimately better performance on the loss function, since this updating method consider the whole training data before updating the weights. In our case, we only have very little differences in their convergence and very similar accuracy values (around 65%).

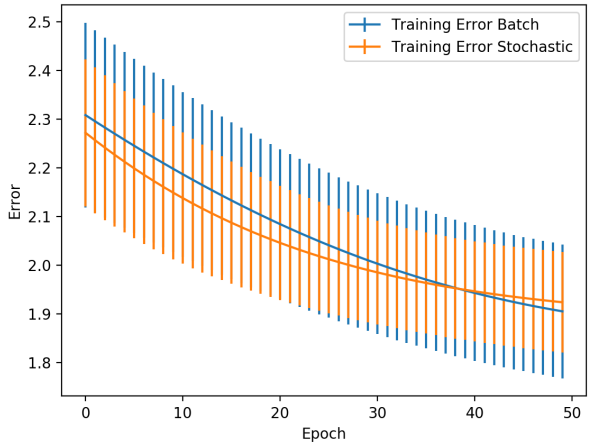


Figure 5: Training Error Comparison of Batch and SGD in 50 Epochs

### 3.3 Visualizing the weights (6c)

After training the model, we get a theta (weights of the model) that has 6 rows, each respectively represent a unique facial expression (see figure below, from left to right: Anger, Disgust, Fear, Happiness, Sadness, Surprise). We could visualize this and rebuild "eigenfaces" based on the weights. Since we applied the Principal Component Analysis (with 20 pcs) on softmax regression, each vector has to be multiplied with the original eigenvectors (eigenvector matrix with only the first 20 columns), so that we could project the vectors with only 20 features back to the image space with 43008 features. After multiplying with the matrix, we use the display image function provided in the data loader file and get all 6 facial expressions as below.

Figure 6: Visualized weights (Anger, Disgust, Fear, Happiness, Sadness, Surprise)

### 3.4 Unbalanced Dataset (6d, Extra Credit)

The performance of the classifier, either the logistic regression or the softmax regression tends to suffer from the imbalanced-distributed data. Intuitively speaking, it is likely that the model will learn more features of the domaining class and relatively fewer features of the rest classes. Therefore, model tends to predict the majority more frequently and might treat the rest data as noise or outliers. In a word, the predictions of the model, which is trained on imbalanced dataset, will be biased.

One possible solution is to do the data augmentation. We could manually create more samples for the minority classes to prevent biased regression model.

## 4 Individual Contributions (7)

This part included the individual contributions of the two team members Huimin Zeng and Zhenrui Yue on the assignment.

### 4.1 Huimin Zeng

I implemented cross validation function by splitting the data into different folds and shuffling them into different orders, I also implemented a simplified version of cross-validation by only assigning the corresponding indices of each fold. Moreover, I implemented PCA and PCA project by using the eigenvectors and eigenvalues of the data matrix based on the Turk and Pentland trick. I built the logistic regression class, computing the forward pass, gradient, the weights updates, predict method and generated relevant charts, compared different regression results and wrote some part of the report.

### 4.2 Zhenrui Yue

I also individually implemented PCA and projection function, computing the average face (eigenfaces) and eigenvalue decomposition and forming the projector matrix. I completed the softmax regression class, including the one-hot coding, loss function (cross entropy) computation, update and predict methods, batch / stochastic gradient descent as well as the confusion matrix part and visualized weights of the softmax regression. I also created some charts, images, table and wrote the most parts of this report.