CSE 555: Pattern Recognition
Problem Set 1: Bayesian Decision Theory

Upasana Ghosh
Department of Computer Science
University at Buffalo
Buffalo, NY 14214
upasanag@buffalo.edu

Problem 1:

The images are 28 x 28 pixels in gray-scale. The categories are 0, 1, ... 9. We concatenate the image rows into a 28 x 28 vector and treat this as our feature and assume the feature vectors in each category in the training data "train-images-idx3-ubyte.gz") have Gaussian distribution. Draw the mean and standard deviation of those features for the 10 categories as 28 x 28 images using the training images ("train-images-idx3-ubyte.gz"). There should be 2 images for each of the 10 digits, one for mean and one for standard deviation. We call those "mean digits" and "standard deviation digits" in CSE455/555.

Answer:

Procedure

1. The MNIST dataset is loaded and processed using the standard python library.

2. A dictionary is created using the data labels as the key and the features as the value. All the images are divided and stored in this dictionary with the keys representing the 10 classes and the images stored against their corresponding classes.

3. The Mean of the pixel values/ features is calculated using the following equation:

$$\bar{x} = \frac{1}{n}\left(\sum_{i=1}^{n} x_i\right) = \frac{x_1 + x_2 + \cdots + x_n}{n}$$

   where x_1, x_2 … x_n represents the pixel values/ features and $\bar{x}$ represents the mean of the values. There is one mean image for each class.

4. The Standard Deviation of the pixel values/ features is calculated using the following equation:

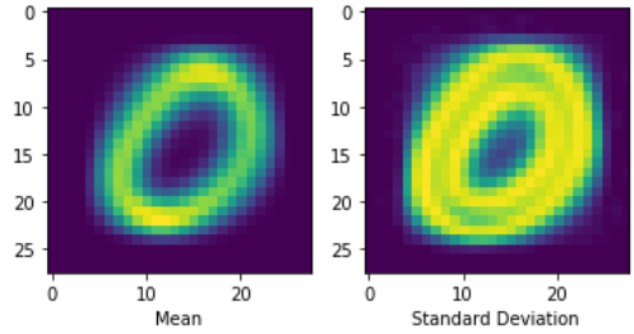
$$s = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(x_i - \bar{x})^2},$$

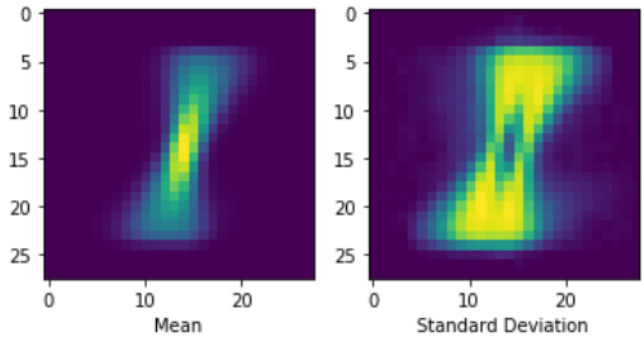
where x_1, x_2 … x_n are the pixel values/ features, $\bar{x}$ is the mean of these values, and N is the number of observations in the sample. Standard Deviation is a measure of the amount of dispersion or variation of a set of values from their mean.

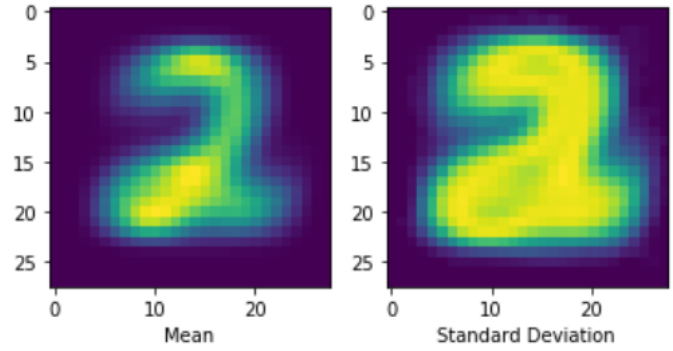
5. We finally persist the output images as 'img_digit_mean<digit>.png' for the mean and 'img_digit_stdev<digit>.png' for the standard deviation for each of the digits.

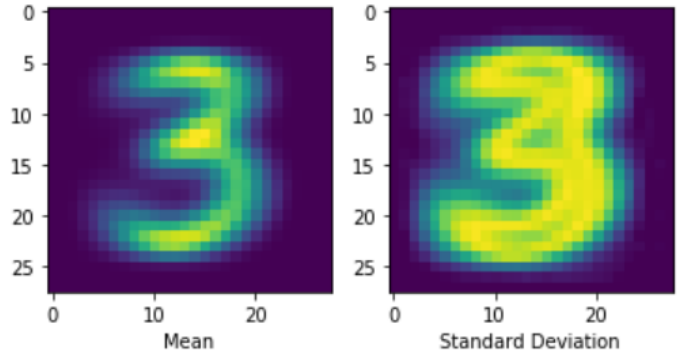
Results

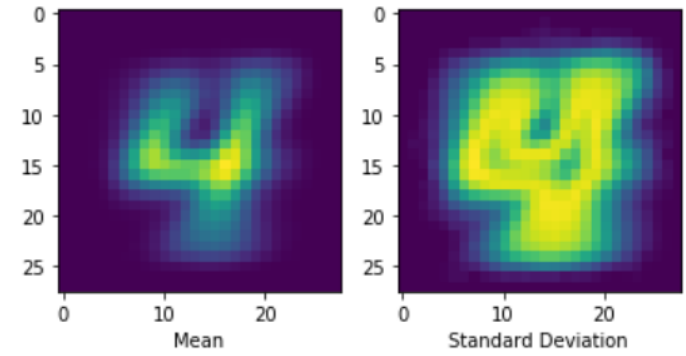
```
================================================================================================

        Displaying the Mean and Standard Deviation for each Digit (side-by-side)

================================================================================================
```

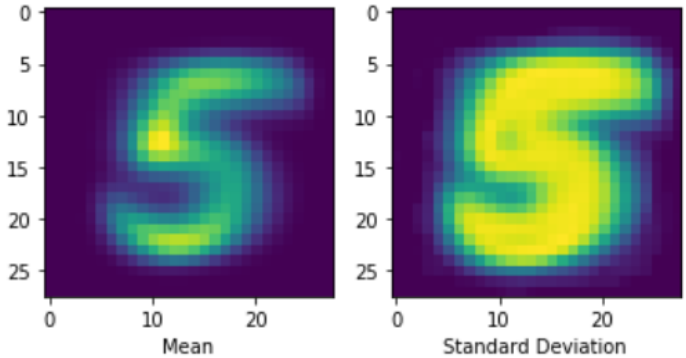


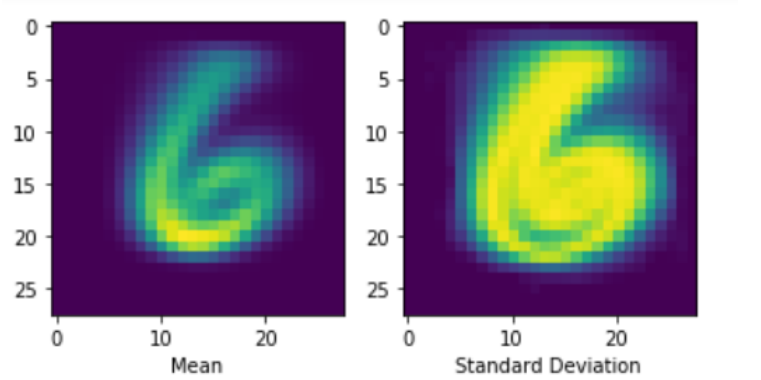


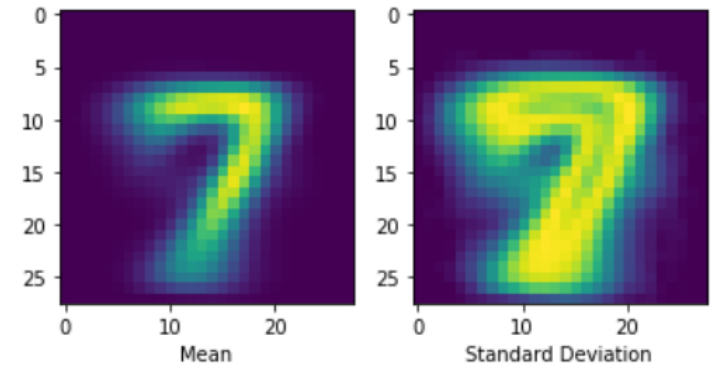

0
5
10
15
20
25
0
10
20
Mean
Standard Deviation

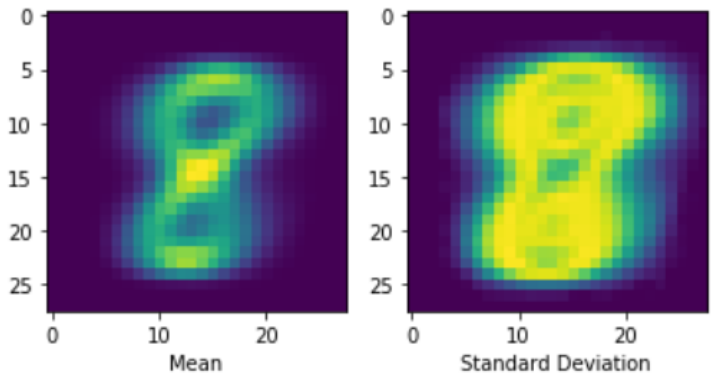

0
5
10
15
20
25
0
10
20
Mean
Standard Deviation

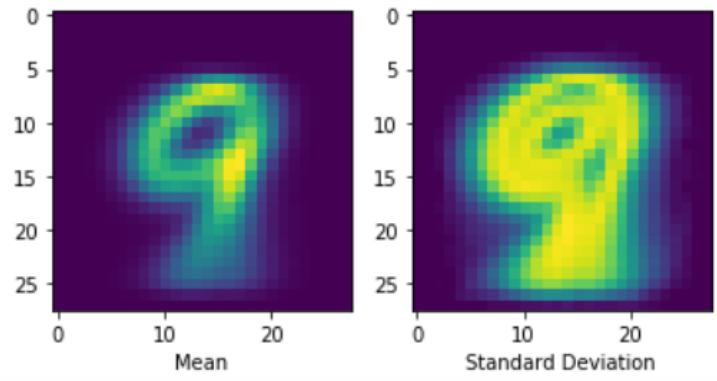

0
5
10
15
20
25
0
10
20
Mean
Standard Deviation


0
5
10
15
20
25
0
10
20
Mean
Standard Deviation


0
5
10
15
20
25
0
10
20
Mean
Standard Deviation


0
5
10
15
20
25
0
10
20
Mean
Standard Deviation


0
5
10
15
20
25
0
10
20
Mean
Standard Deviation


0
5
10
15
20
25
0
10
20
Mean
Standard Deviation

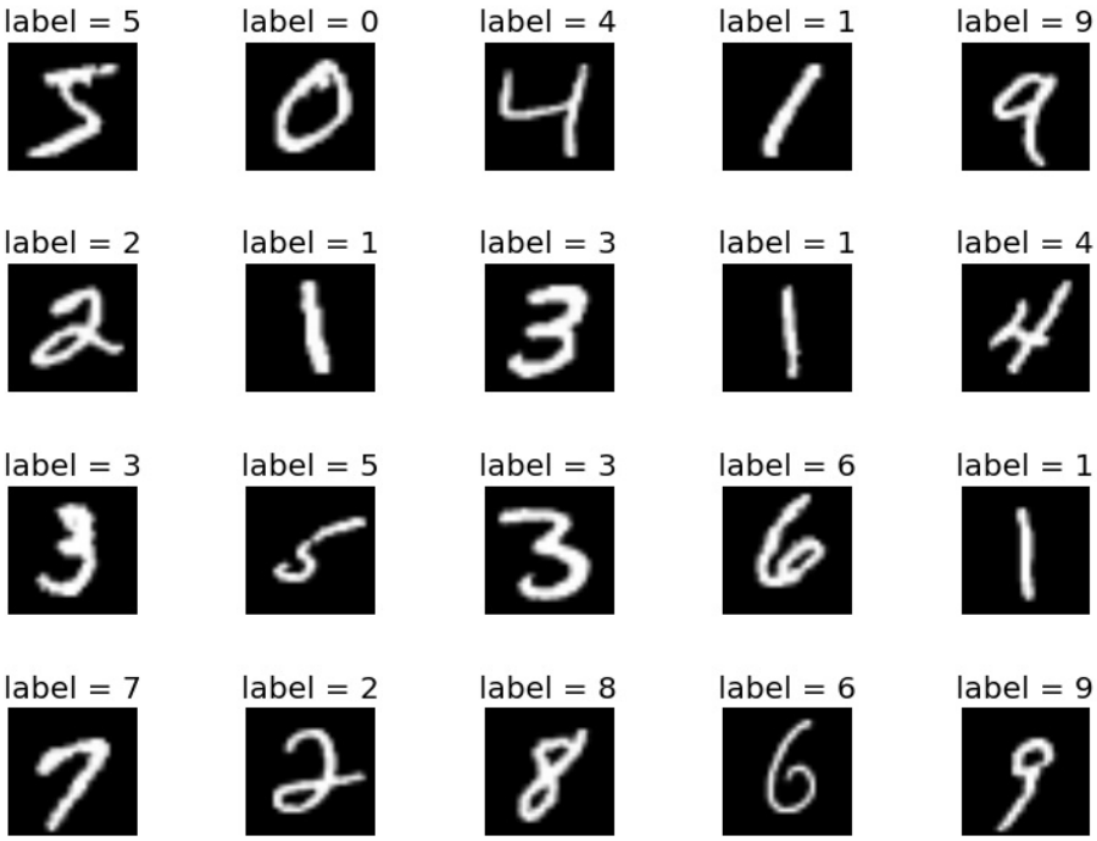
Problem 2:

Classify the images in the testing data set ("t10k-images-idx3-ubyte.gz") using 0-1 loss function and Bayesian decision rule and report the performance. Why it doesn't perform as good as many other methods on LeCun's web page? Before coding the discriminant functions, review Section 2.6.

Answer:

Dataset

We are using the MNIST Dataset. The MNIST database (Modified National Institute of Standards and Technology database) is a large database of handwritten digits that is commonly used for training various image processing systems. The database is also widely used for training and testing in the field of machine learning. It was created by "re-mixing" the samples from NIST's original datasets. The creators felt that since NIST's training dataset was taken from American Census Bureau employees, while the testing dataset was taken from American high school students, it was not well-suited for machine learning experiments. Furthermore, the black and white images from NIST were normalized to fit into a 28x28 pixel bounding box and anti-aliased, which introduced grayscale levels.



Discriminant Analysis

Discriminant analysis is a popular method for multiple-class classification. Linear discriminant analysis (LDA) or normal discriminant analysis (NDA) is a generalization of Fisher's linear discriminant, a method used in statistics, pattern recognition and machine learning to find a linear combination of features that characterizes or separates two or more classes of objects or events.

Discriminant analysis is used when groups are known a priori. Each case must have a score on one or more quantitative predictor measures, and a score on a group measure. In simple terms, discriminant function analysis is a classification method that is used to distribute similar type of data into groups, classes or categories.

In this project, we have made use of the Quadratic Discriminant Analysis, which has the following form:

$$g_i(x) = x^t W_i x + N_i^t x + B_{i0},$$

where $W_i = -\frac{1}{2}\sum_i^{-1}$, $N_i = \sum_i^{-1}\mu_i$ and $B_{i0} = -\frac{1}{2}\mu_i^t\sum_i^{-1}\mu_i + lnP(\omega_i) - \frac{1}{2}ln|\sum_i|$ and quadratic boundary.

Quadratic discriminant analysis (QDA) is closely related to linear discriminant analysis (LDA), where it is assumed that the measurements from each class are normally distributed. Unlike LDA however, in QDA there is no assumption that the covariance of each of the classes is identical. When the normality assumption is true, the best possible test for the hypothesis that a given measurement is from a given class is the likelihood ratio test.

Results

After successfully implementing and running the Quadratic Discriminant Analysis function it was observed that the classifier was able to classify the MNIST digit dataset with an accuracy of 82.49%.

```
================================================================================

 Bayes Decision Rule Accuracy:  82.49
 Naive Bayes Accuracy: 9.799999999999997

================================================================================
 Time taken for calculating 0-1 Loss Function: 1539.286013841629
================================================================================
```

Conclusion

The reasons behind '0-1 Loss Function' and 'Bayesian Decision Rule' not performing as good as many other methods mentioned in the LeCun's web page are discussed below:

1. Naive Bayes classifier makes a very strong assumption about the shape of the data distribution, i.e. any two features are independent given the output class. Due to this assumption, the result can be (potentially) not satisfactory. Hence, LeCun's web page discusses several methods which do not make this critical assumption.

2. For any possible value of a feature, we need to estimate a likelihood value by a frequentist approach. This can result in probabilities going towards 0 or 1, which in turn leads to numerical instabilities and worse results. In this case, we need to smoothen the probabilities in some way, or to impose some prior on our data. However, it may be argued that the resulting classifier is not naive anymore. This shortcoming is absent in the methods involving ANNs and Neural Networks.

3. It is common to use a binning procedure to make them discrete, but if we are not careful a lot of information can be lost in the process. Another possibility is to use Gaussian distributions for the likelihoods.

4. Bayes or Naïve Bayes classifiers tend to work well with small training datasets, as compared to methods involving Neural Networks and ANNs. Here we have 60000 data in the training data set and the number of classes are 10. Instead of this, if we had two classes and a lower number of training data items then we could have got a better accuracy rate. As the classifiers are trained with increasing training datasets, the performance of the Naive Bayes classifier plateaus above a certain threshold. The simplicity of these kinds of classifiers prevents them from benefiting incrementally from the increasing training data past a certain point.

5. Naive Bayes' simplicity prevents it from fitting it's training data too closely. In contrast, due to their complexity Neural Networks can very easily over fit training data, especially when provided with large data sets.

6. Here in the Naïve Bayes' classifier, we must train each class one by one with the training data where as if we considered the implementation of a neural network classifier as mentioned in the website, they are trained simultaneously for different classes on large datasets.

Problem 3:

Construct the "Fisher digits" from the MNIST data set according to Sections 3.8.2 and 3.8.3. This web page on Fisher faces (http://www.scholarpedia.org/article/Fisherfaces) and this web page (https://www.bytefish.de/blog/fisherfaces/) might be helpful. Answer two questions about these sections: (a) Why should the vector w minimizing Eq. (103) satisfy Eq. (104)? (b) Why should the between-class scatter matrix in Eq. (115) is times the one in Eq. (102) in two-class case (i.e., c=2)? In addition, convince ourselves that Eq. (125) is the quotient between two "volumes" by referring the Wikipedia page on determinant (https://en.wikipedia.org/wiki/Determinant).

Answer:

Fisher's LDA is a dimension reduction technique. Such techniques can primarily be used to reduce the dimensionality for high-dimensional data. We do this for multiple reasons- dimension reduction as feature extraction, dimension reduction for classification or for data visualization.

Procedure

The "Fisher Digits" from the MNIST data set were constructed using the following procedure:

1. Within class differences was estimated using the within-class scatter matrix, given by

$$\mathbf{S}_w = \sum_{j=1}^{C} \sum_{i=1}^{n_j} (\mathbf{x}_{ij} - \mu_j)(\mathbf{x}_{ij} - \mu_j)^T$$

   where $\mathbf{x}_{ij}$ is the i^{th} sample of class j, μ_j is the mean of class j, and n_j the number of samples in class j

2. The between class differences were computed using the between-class scatter matrix, given by

$$\mathbf{S}_b = \sum_{j=1}^{C} (\mu_j - \mu)(\mu_j - \mu)^T$$

   where μ represents the mean of all classes.

3. We now find the basis vectors **V** where $\mathbf{S}_w$ is minimized and $\mathbf{S}_b$ is maximized, where **V** is a matrix whose columns $\mathbf{v}_i$ are the basis vectors defining the subspace. These are given by,

$$\frac{|\mathbf{V}^T \mathbf{S}_b \mathbf{V}|}{|\mathbf{V}^T \mathbf{S}_w \mathbf{V}|}$$

4. The solution to this problem is given by the generalized eigenvalue decomposition

$$\mathbf{S}_b\mathbf{V} = \mathbf{S}_w\mathbf{V}\mathbf{\Lambda}$$

where **V** is the matrix of eigenvectors and **Λ** is a diagonal matrix of corresponding eigenvalues.

The eigenvectors of **V** associated to non-zero eigenvalues are the Fisher digits. There is a maximum of $C-1$ Fisher digit. This can be readily seen from the definition of $\mathbf{S}_b$. In our definition, $\mathbf{S}_b$ is a combination of C feature vectors and any C vectors define a subspace of $C-1$ or less dimensions. The equality holds when these vectors are linearly independent from one another.

Results

```
================================================

          Displaying Fisher Digits

================================================
```