CSE 555: Pattern Recognition: Project Report- Bayesian Decision Theory

Manisha Biswas
mbiswas2@buffalo.edu

Problem Statement: Apply Discriminant Analysis to recognize the digits in the MNIST data set. Also, classify the digits using 0-1 Loss Function and Bayesian Decision Rule to report the performance.

Given Datasets:

1. **Training Data Sets:**

   a. train-images-idx3-ubyte.gz: **Size – 60000 x 28 x 28**. Contains 60,000 sample images of handwritten MNIST digits from categories 0, 1, 2, … 9. The images are 28 X 28 pixels in gray- scale.

   b. train-labels-idx1-ubyte.gz: **Size – 60000 x 1.** Contains labels of the 60,000 image samples in training set, representing the class each image belongs to, from 0, 1, … 9.

2. **Testing Data Sets:**

   a. t10k-images-idx3-ubyte.gz: **Size – 10000 x 28 x 28**. Contains 10,000 sample images of handwritten MNIST digits from categories 0, 1, … 9 against which the classifier is to be tested. The images are 28 X 28 pixels in gray-scale.

   b. t10k-labels-idx1-ubyte.gz: **Size – 10000 x 1.** Similar to 1.b dataset, this dataset contains labels of the 10,000 image samples in testing set, representing the class each image belongs to, from 0, 1,… 9.

<u>Task 1</u>: Draw the mean and standard deviation of the features (28 x 28 vectors) for the 10 classes as 28 x 28 images, using the Training Data Sets.

The mean of a distribution is given as –

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

Hence, for calculating the mean images of every class 0, 1, … 9, from the training datasets, we calculate the sum of all the 28 x 28 pixels of the images belonging to a particular class and dividing the sum image by the total number of images belonging to that class.

After calculation, the following mean images were obtained –

Fig 1. Mean images obtained for the MNIST dataset

The standard deviation of a distribution is given as –

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

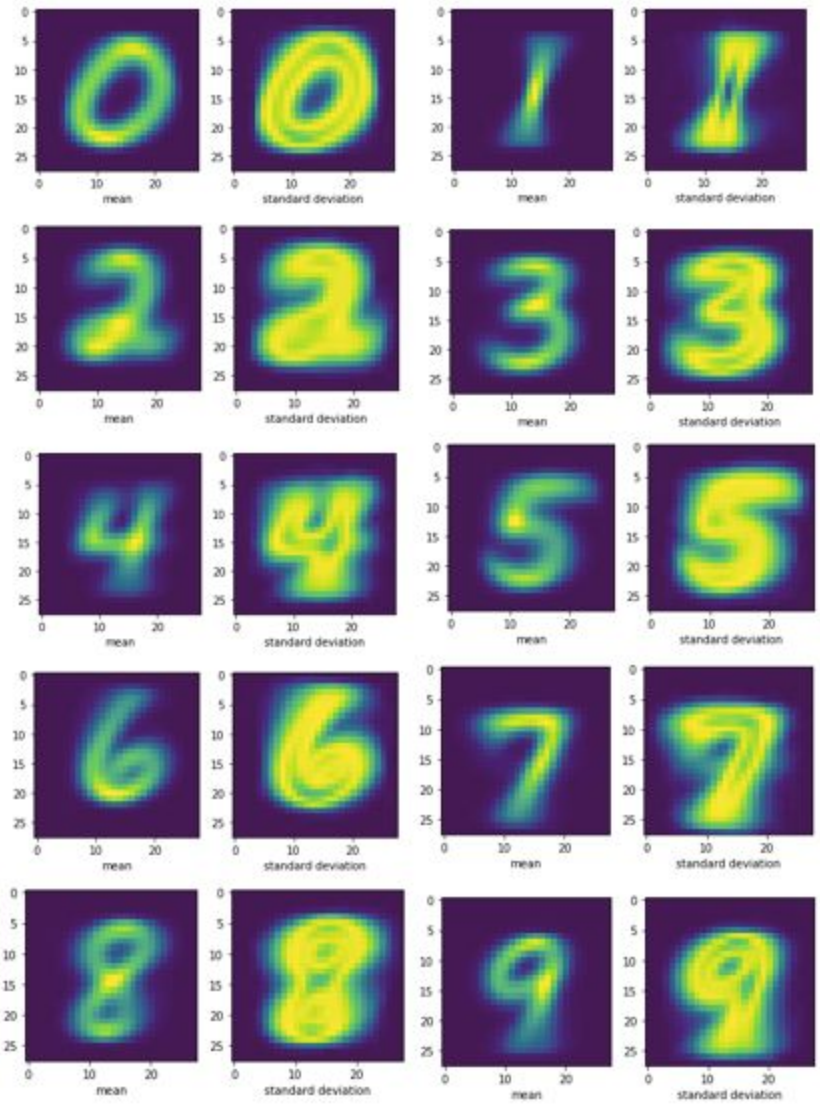
where $\{x_1, x_2, \ldots, x_N\}$ are the observed values of the sample items, $\bar{x}$ is the mean value of these observations, and N is the number of observations in the sample.

Hence, we calculate the standard deviation images of every class using the above formula, where x_i represents the images belonging to each class i and x' represents the mean image of that class.

After calculation, the following standard deviation images were obtained –

Fig 2. Standard Deviation images obtained for the MNIST dataset

Results obtained:



Task 2: Classify the images in the testing data set using 0-1 loss function and Bayesian Decision Rule. Report the performance of the classifier.

The Bayes rule is defined as:

$$\text{Bayesian theorem: } P(\omega_j|x) = \frac{P(x|\omega_j)P(\omega_j)}{p(x)}, \text{ or}$$

$$\text{posterior} = \frac{\text{likelihood} \times \text{prior}}{\text{evidence}}$$

$$\text{Evidence: } p(x) = \sum_{j=1}^{c} P(x|\omega_j) * P(\omega_j)$$

In simpler terms, if *c* represents our class and *x* is the image from the test set, we can rewrite Bayes Rule as –

$$P(c\,|\,x) = \frac{P(x\,|\,c)P(c)}{P(x)}$$

x has been formatted as a vector, i.e. the 28 x 28 pixels are flattened to a vector of size 1 x 784.

The left side, called the posterior P(*c*|*x*) is the probability that the class is c given the data x

The right side, called the **likelihood** P(*x*|*c*) is the probability that the data x belongs to the class c.

To find the class that the image x belongs to, we can find-

$$c^* = argmax_c P(c\,|\,x) = argmax_c \frac{P(x\,|\,c)P(c)}{P(x)}$$

However, in this case, the **evidence,** *P(x)* is only a scaling factor as it will remain the same for all classes, *c*. So, the above expression reduces to –

$$c^* = argmax_c P(x|c)P(c)$$

Now, we derive a good model for the continuous data, i.e. the multi-variate Gaussian or the multi-variate Normal, given as –

$$P(x|c) = \frac{1}{\sqrt{(2\pi)^D|\Sigma|}} exp\left(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right)$$

Where, *D* = Dimension of the features = 784 in this case

$\sum$ = Covariance matrix of every class, thus **we have 10 covariances matrices**.

So, first, we calculate the covariance matrix $\sum$ for every class, and also its Determinant, $|\sum|$ and inverse as $\sum^{-1}$. The covariance is calculated by using the following formula –

$$\mathbf{Q} = \frac{1}{N-1}\sum_{i=1}^{N}(\mathbf{x}_i. - \bar{\mathbf{x}})(\mathbf{x}_i. - \bar{\mathbf{x}})^{\mathrm{T}}$$

Another approach is **Naïve Bayes Decision Theory** that considers all features to be linearly independent and hence, there exists a diagonal covariance matrix for every class, the diagonal elements of which are given by the pixel values of the standard deviation image of that class.

But, calculating *P(x|c)* with the above method is tedious and involves high complexity. We can reduce it further by taking natural log (*ln*) on both sides and get the following equation –

$$logP(x|c) = -\frac{D}{2}ln(2\pi) - \frac{1}{2}ln|\Sigma| - \frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)$$

We wanted to find the class a particular digit belongs to using $argmax_c$ function. However, we can also get it as –

$$c^* = argmax_c(logP(x|c) + logP(c))$$

0-1 Loss Function:

Once we have achieved the classes every image in the testing set belongs to, we pass it through the 0-1 loss function to calculate the accuracy or performance of the model. The 0-1 loss function is given as –

$$\lambda(\alpha_i|\omega_j) = \begin{cases} 0 & i = j \\ 1 & i \neq j \end{cases}, \text{ where } i, j \in \{1, \cdots, c\}$$

Counting the number of 0s predicted by the loss function, we can derive the accuracy.

The accuracy calculated by the Bayes Decision Theory was 82.49%

Accuracy predicted by Naïve Decision Theory = 85.04%

<u>Question:</u> Why doesn't the Bayes Decision Theory perform as good as many other methods of LeCun's web page?

Answer :

Based on my understanding, there are multiple reasons that attribute to a worse performance of Bayes Decision Theory than other methods on LeCun's web page –

- Firstly, **no image pre-processing has been done in Bayes Decision Theory** that would augment the pixels of every image to lead to a better perception of the class of the image by the classifier. However, the other methods listed on his web-page that have better accuracy have some extent of pre-processing on the images. For example, SVMs and CNNs have such low error rate as the images are convolved by a feature mask

before passing to the classifier models (Neural Networks) and there comes the image pre-processing.

- Bayesian Decision Theory assumes that the samples have a Gaussian distribution which may not necessarily be true.
- Linear Discriminant Analysis does not work well if the degree of variation in the number of classes is high, i.e. if the number of objects in various classes are different to a great extent. Also, LDA is **sensitive to overfitting and takes difficult methods of evaluation.**
- **LDA is not applicable to non-linear problems.**
- Naïve Bayes considers all features to be linearly independent. This assumption is most often incorrect. For example, if a pixel in the middle is totally bright or has high intensity, it is highly probable that the neighbouring pixels are also of high intensity.
- **Naïve Bayesian decision rule always selects the class *i* that maximizes the posterior probability P(x|c), not always necessary.** It is also possible that with very minute difference in probability, the actual class does not get predicted by the classifier as due to some artefacts, the next closest digit was predicted with higher posterior probability. For e.g., the digits 0, 8 and 6 are very similar to each other and some noise or artefact in 0 can result in it appearing as 8 or 6. Thus prediction would be incorrect.
- 0-1 loss function is always accuracy centric, which is not always the objective. The 1's become indicators for misclassified items, regardless of how the classification or misclassification was made. However, if we change the weighting on the loss function, the same interpretation won't hold true anymore. The 0-1 loss function only cares whether the classification is correct or incorrect, without checking how the errors are made. This may lead to costlier situations. For example, misclassifying a positive case of cancer as negative, i.e. false negative is costlier than correctly classifying a negative case of cancer as positive, false positive. Hence, the 0-1 loss function is not always dependable.

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 n subscript 1 asterisk times n subscript 2 divided by n 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:

The “Fisher Digits” from the MNIST data set were constructed using the following steps:

Within class differences can be 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 ith sample of class j, μ_j is the mean of class j, and n_j the number of samples in class j.

Likewise, the between class differences are computed using the between-class scatter matrix,

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

where μ represents the mean of all classes.

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

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

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 $\mathbf{V}$ is (as above) the matrix of eigenvectors and $\mathbf{\Lambda}$ is a diagonal matrix of corresponding eigenvalues.

The eigenvectors of $\mathbf{V}$ associated with non-zero eigenvalues are the Fisherfaces. There is a maximum of $C-1$ Fisherfaces. This can be readily seen from the definition of $\mathbf{S}_b$. Note that in our definition, $\mathbf{S}_b$ is a combination of C feature vectors. 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: