# FISHER LINEAR DISCRIMINANT ANALYSIS

## Overview

Fisher discriminant analysis or linear discriminant analysis represents basically same concept, but the terms are often used interchangeably. It is most often used for dimensionality reduction in the pre-processing step. It project a dataset onto a lower-dimensional space with good inter class-separability in order to avoid overfitting and reduce computational costs. Fisher's linear discriminant is a classification method performs in exact manner. It projects high-dimensional data onto a line which can be divided into classes by a threshold. Projection is done in such a way that distance between class-means is maximized while class variance is minimized. The cost function is a function of class means and variances. This is also called fisher criterion. For 2 class problem, the cost function of fisher criterion can be expressed as

where  represents a class mean,  represents a class variance, and the subscripts denote the two classes. This cost function is maximized for optimal class separation (classification). Fisher criterion is the classification analogue of signal to noise ratio (SNR) from signal processing. By maximizing this criterion, a closed form solution is yielded that involves the inverse of a covariance-like matrix. This method has strong parallels to linear perceptrons. For 2 class classification, the separating threshold is leaned by optimizing a cost function on the training set. The data projection onto a one dimensional line and resulting separation threshold from fisher criterion is shown in Figure 1

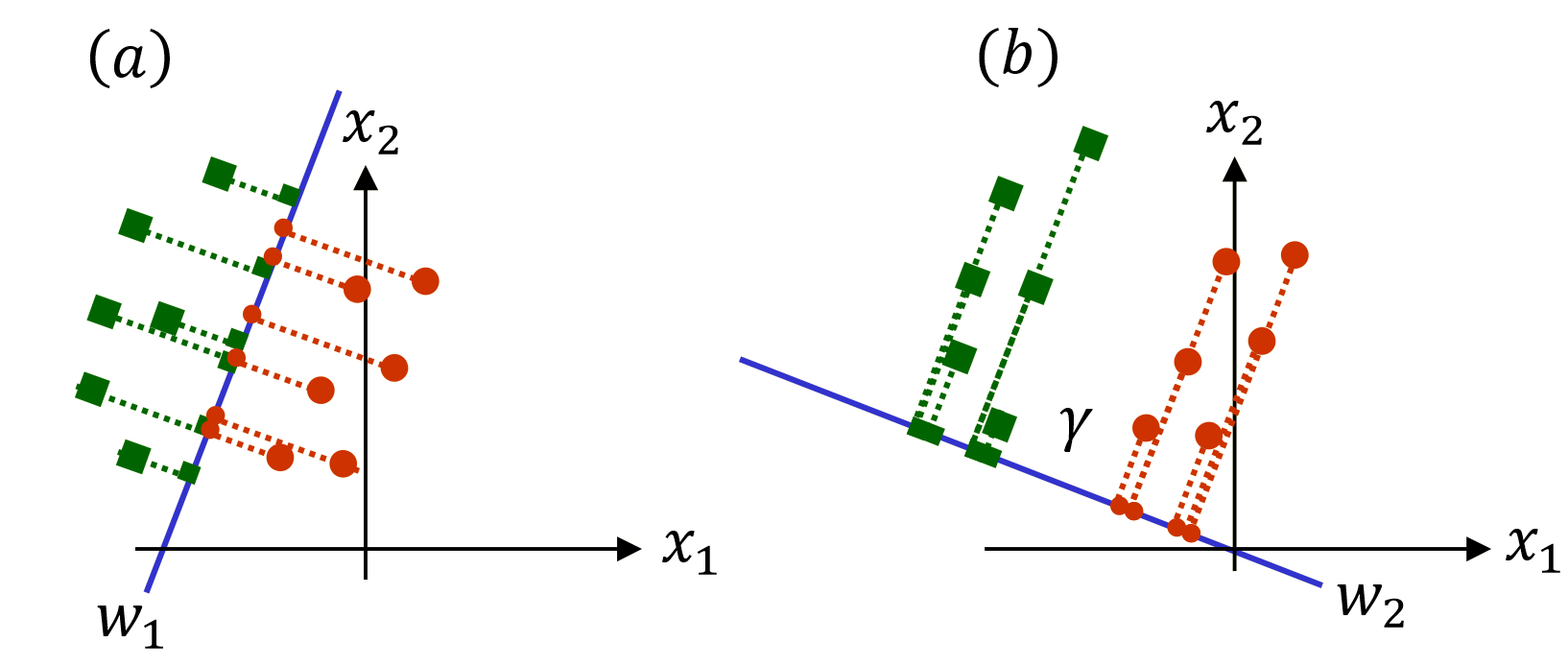


Figure 1: Projection of 2 dimensional data onto one dimensional line. Data is originally represented by two features. These are projected on a one dimensional line for better class separation. (a): shows bad projection which cannot separate classes. (b): shows a good projection which gives much better class separation. The separating threshold is determined by fisher criterion over a training set

## MATLAB Implementation

### Typographic Conventions

Functions (script) names starting with capital letters correspond to “GUI functions”

Functions (script) names starting with small letters correspond to “core mathematical functions” used to develop the network

All variable names start with capital letters

All structure names start with small letters

### Functions used

#### fisher\_training

This function is used to train a fisher object given training data. It projects the multi-dimensional data onto one dimensional plane. This is done by finding optimal weights for linear combination of inputs features. It then uses the training labels to estimate optimal separation threshold for the classes using fisher criterion. The fisher criterion is found by minimizing cost function expressed in Eq. (1). This multi-class fisher classifier is based on binary fisher classifier developed by Quan Wang (Signal Analysis and Machine Perception Laboratory, Department of Electrical, Computer, and Systems Engineering, Rensselaer Polytechnic Institute, Troy, NY 12180, USA). Multi-class training is done in **one-against-all** manner.

##### Syntax

fisher=fisher\_training(Train\_X,Train\_Y)

##### Input Arguments:

Train\_X: Training inputs (Dimensions: Number of entries Number of features)

Train\_Y: Training targets (Dimensions: Number of entries Number of classes if binary class coding is used, Dimensions: Number of entries 1 if actual class label is used instead of binary class coding). For example, there are two output classes, class 1 and class 2. In binary coding scheme, there are two columns of Train\_Y, once corresponds to class 1 and the second corresponds to class 2. All entries of column 1 where there is class 1 equals to 1 while rest are 0. Same goes for column 2 and class 2.

##### Output Arguments:

fisher: Trained fisher object. A typical fisher object contains following fields

**Weights:** Linear weights of each feature for linear combination while calculating one dimensional plane

**Thresh:** Optimal class separation threshold

A typical fisher object for training data with 20 features looks like

fisher = W: [20x1 double]

Thresh: -0.0044

Help: Type ‘help fisher\_training’ in command window to access help file for this function

#### fisher\_testing

This function is used to test fisher classifier on data. It takes features weights of the fisher object, uses them to find optimal one-dimensional plane. Once the plane is calculated, the optimal threshold is used to separate the data into two classes.

##### Syntax

[Predicted, Precision, Recall, Accuracy, F1, Actual] = fisher\_testing(fisher,X,Actual,display)

##### Input Arguments:

X: Test Inputs (Dimensions: Number of entries Number of features)

Actual: Test Targets (Dimensions: Number of rows equals to number of rows in X while number of columns equals to number of columns in Train\_Y)

fisher: Trained fisher object (with fields w and threshold)

display: Optional parameter to display classification results in command window. Its value should be 1 for displaying and 0 for not displaying. This parameter does not change training or testing conditions, it only helps us see the results in command window.

##### Output Arguments:

Predicted: Predicted labels for Test Inputs

Precision: Fraction of class labels retrieved that are relevant. Range [0 100]

Recall: Fraction of relevant class labels that are retrieved. Range [0 100]

Accuracy: Fraction of correctly classified labels. Range [0 100]

F1: harmonic mean of Precision and Recall. Range [0 100]

Actual: Actual labels of Test Inputs

Help: Type ‘help fisher\_testing’ in command window to access help file for this function

# LEAST SQUARE CLASSIFIER

Least squares is another example of linear classifiers in which the base assumption is that output is a linear combination of inputs features. The linear combination is often referred to as projected one dimensional space. In mathematical notion, if  is the predicted value and is the input feature of p-dimensional feature space

where the weight (coefficient) matrix and x is the input matrix. Figure 1 is a good illustration of projection represented by Eq. (). For two class classification problem, we have to find a good separation of the projected data which divides it in two classes. In least square classifier, the coefficients  w = (w_1, ..., w_p) are calculated to minimize the residual sum of squares between the observed responses in the dataset, and the responses predicted by the linear approximation. Mathematically it solves a problem of the form:

## MATLAB Implementation

### Typographic Conventions

Functions (script) names starting with capital letters correspond to “GUI functions”

Functions (script) names starting with small letters correspond to “core mathematical functions” used to develop the network

All variable names start with capital letters

All structure names start with small letters

### Functions used

#### lsc\_train

This function is used to train a least square classifier on training data. It performs least squares regression between the independent variables, Train\_X and dependent variable Train\_Y. After least squares regression, an argmax function is used to determine output class.

##### Syntax

[lsc] = lsc\_train(Train\_X,Train\_Y,tol)

##### Input Arguments:

Train\_X: Training inputs (Dimensions: Number of entries Number of features)

Train\_Y: Training targets (Dimensions: Number of entries Number of classes if binary class coding is used, Dimensions: Number of entries 1 if actual class label is used instead of binary class coding). For example, there are two output classes, class 1 and class 2. In binary coding scheme, there are two columns of Train\_Y, once corresponds to class 1 and the second corresponds to class 2. All entries of column 1 where there is class 1 equals to 1 while rest are 0. Same goes for column 2 and class 2.

tol: This parameter denotes the least square error tolerance which we are ready to allow for Eq. (3). As regression is done in the least square sense, this parameter dictates the value of least square error which we are ready to tolerate. We can specify any positive value for this parameter. Higher value generally means quicker convergence but bad classifier and vice versa.

Range [0 Inf]

##### Output Arguments:

lsc: Trained fisher object. A typical fisher object contains following fields

T score matrix of Inputs

P loading matrix of Inputs

U score matrix of Targets

Q loading matrix of Targets

B matrix of regression coefficients

W weight matrix of Inputs

Help: Type ‘help lsc\_train’ in command window to access help file for this function

#### lsc\_test

This function is used to test least squares classifier on data. It takes lsc object, inputs and targets as input arguments and apply Eq. (2, 3) on them. First it projects the inputs matrix onto another dimension by using Input loading matrix P and input weight matrix W, then projects it on target space by multiplying it with loading matrix of targets.

##### Syntax

[Predicted, Precision, Recall, Accuracy] = lsc\_test(lsc,X,Y)

##### Input Arguments:

X: Test Inputs (Dimensions: Number of entries Number of features)

Y: Test Targets (Dimensions: Number of rows equals to number of rows in X while number of columns equals to number of columns in Train\_Y)

lsc: Trained lscobject (with fields W,P and Q)

display: Optional parameter to display classification results in command window. Its value should be 1 for displaying and 0 for not displaying. This parameter does not change training or testing conditions, it only helps us see the results in command window.

##### Output Arguments:

Predicted: Predicted labels for Test Inputs

Precision: Fraction of class labels retrieved that are relevant. Range [0 100]

Recall: Fraction of relevant class labels that are retrieved. Range [0 100]

Accuracy: Fraction of correctly classified labels. Range [0 100]

Help: Type ‘help lsc\_test’ in command window to access help file for this function

# NAÏVE BAYES CLASSIFIER

The Bayesian Classification represents a statistical supervised learning method. It assumes an underlying probability distribution which allows it to capture uncertainty by determining probabilities of the outcomes. It is named after Thomas Bayes (1702-1761), who proposed the famous Bayes Theorem.

Bayesian classification works on the “naive” assumption of independence between every pair of features in the features space. Given a class variable  and a feature vector through, Bayes’ theorem states the relationship in Eq. (4) to be true

If naïve independence assumption is true, then

This relationship can be simplified for all as shown in Eq. (6)

Since  is constant given the input, classification can be derived as shown in Eq. 7(a, b):

Maximum-A-Posteriori (MAP) estimation can be used to estimate  and; where is the relative frequency of class  in the training set. Taking different distributions of  can result in different Bayesian classifiers.

In spite of their over-simplified assumptions, naive Bayes classifiers have proven to be powerful classification tools in many real-world application. They require a small amount of training data to estimate the necessary parameters. They can be extremely fast as well when compared to more sophisticated methods.

## MATLAB Implementation

### Typographic Conventions

Functions (script) names starting with capital letters correspond to “GUI functions”

Functions (script) names starting with small letters correspond to “core mathematical functions” used to develop the network

All variable names start with capital letters

All structure names start with small letters

### Functions used

#### naïve\_bayes\_train

This function is used to train a naïve Bayesian classifier on training data. It trains a probabilistic classifier. This classifier is based on binary naïve Bayesian classifier developed for Gaussian distribution of input data. Multi-class implementation is done in one-againt-all manner. Let’s there are N classes in the training data, the naïve Bayesian classifier object is trained for identify all the classes, one by one. For instance, the classifier is being trained for class ‘a’, all the input values corresponding to class ‘a’ will be treated as class 1 and all the ‘non-a’ values will be treated as class 0. This way, we have N Likelihood-matrices and N-Evidence matrices in the final trained naïve Bayesian object.

##### Syntax

[naivebayes] = naive\_bayes\_train(Train\_X, Train\_Y)

##### Input Arguments:

Train\_X: Training inputs (Dimensions: Number of entries Number of features)

Train\_Y: Training targets (Dimensions: Number of entries Number of classes if binary class coding is used, Dimensions: Number of entries 1 if actual class label is used instead of binary class coding). For example, there are two output classes, class 1 and class 2. In binary coding scheme, there are two columns of Train\_Y, once corresponds to class 1 and the second corresponds to class 2. All entries of column 1 where there is class 1 equals to 1 while rest are 0. Same goes for column 2 and class 2.

##### Output Arguments:

naivebayes: Trained Naïve Bayesian classifier object. A typical naïve Bayesian classifier object contains following fields

Likelihood\_Matrix A likelihood matrix for each input feature against each class

Priors Prior probabilities of target classes

Evidence Relative importance of each input feature in from classification point of view

Help: Type ‘help naïve\_bayes\_train’ in command window to access help file for this function

#### naïve\_bayes\_test

This function is used to test naïve Bayesian squares classifier on data. It takes naïve Bayesian object, inputs and targets as input arguments and apply Eq. (6, 7, 8) on them. First it finds posterior probabilities of classes using feature evidence and likelihood matrix in association with prior probabilities. Then it uses argmax function to decide output class. Output class will be the one with maximum posterior probability.

##### Syntax

[Predicted, Precision, Recall, Accuracy] = naive\_bayes\_test(naivebayes,X,Y)

##### Input Arguments:

X: Test Inputs (Dimensions: Number of entries Number of features)

Y: Test Targets (Dimensions: Number of rows equals to number of rows in X while number of columns equals to number of columns in Train\_Y)

naivebayes: Trained naïve Bayesian classifier

##### Output Arguments:

Predicted: Predicted labels for Test Inputs

Precision: Fraction of class labels retrieved that are relevant. Range [0 100]

Recall: Fraction of relevant class labels that are retrieved. Range [0 100]

Accuracy: Fraction of correctly classified labels. Range [0 100]

Help: Type ‘help naïve\_bayes\_test’ in command window to access help file for this function

#### naïve\_bayes\_classify

This function is used to apply a naïve Bayesian classifier for two class target data

##### Syntax

[Predicted\_Classes, Posteriors] = naive\_bayes\_classify(naivebayes,X)

##### Input Arguments:

X: Test Inputs (Dimensions: Number of entries Number of features)

naivebayes: Trained naïve Bayesian classifier

##### Output Arguments:

Predicted\_Classes: Predicted labels for inputs X (there will be only two predicted labels)

Posteriors: Posterior probabilities for target classes

Help: Type ‘help naïve\_bayes\_classify’ in command window to access help file for this function

# PRINCIPAL COMPONENT ANALYSIS (PCA)

PCA is a dimensionality reduction technique which transforms correlated features into smaller number of uncorrelated features. A PCA model targets the variations in a set of variables by searching for uncorrelated linear combination of correlated variables while preserving the information as much as possible. These linear combinations are called principal components. Two primary objectives of PCA are

* To discover or to reduce the dimensionality of the data set.
* To identify new meaningful underlying variables.

PCA is based on Eigen analysis in eigenvalues and eigenvectors of a square symmetric matrix are solved with sums of squares and cross products. The eigenvector associated with the largest eigenvalue has the same direction as the first principal component. The eigenvector associated with the second largest eigenvalue determines the direction of the second principal component and so on. The sum of the eigenvalues equals the trace of the square matrix and the maximum number of eigenvectors equals the number of rows (or columns) of this matrix.

For *p* variables, *p* principal components are formed as follows:

* The first principal component is the linear combination of the standardized original variables that has the greatest possible variance.
* Each subsequent principal component is the linear combination of the variables that has the greatest possible variance and is uncorrelated with all previously defined components.

Each principal component is calculated by taking a linear combination of an eigenvector of the correlation matrix (or covariance matrix or sum of squares and cross products matrix) with the variables. The eigenvalues represent the variance of each component.

## MATLAB Implementation

### Typographic Conventions

Functions (script) names starting with capital letters correspond to “GUI functions”

Functions (script) names starting with small letters correspond to “core mathematical functions” used to develop the network

All variable names start with capital letters

All structure names start with small letters

### Functions used

#### naïve\_bayes\_train

The function runs Principal Component Analysis on a set of data points. It works on Eigen value decomposition. Eigen analysis is performed on input matrix first and largest Eigen values along with their corresponding Eigen vectors are found. These Eigen values and Eigen vectors are then used to project the input matrix along its principal components. If user has specified smaller dimension as input parameter instead of original dimension, project matrix is returned with reduced dimensions along with projection map. This function is based on method proposed by Laurens van der Maaten, Maastricht University, 2007

##### Syntax

[MappedX, XMap] = prin\_comp\_analysis(X, NDims)

##### Input Arguments

X: Features matrix on which we want to run PCA for dimensionality reduction

NDims: Target dimensions for the reduced dimensionality matrix. Let’s say original dimensions of the input feature matrix X are. For dimensionality reduction,

##### Output Arguments

MappedX: Output feature vector with reduced dimensions

XMap: A structure which contains information about mapping of original feature matrix onto mapped features matrix. It has following fields

**mean:** Means of retained principal components

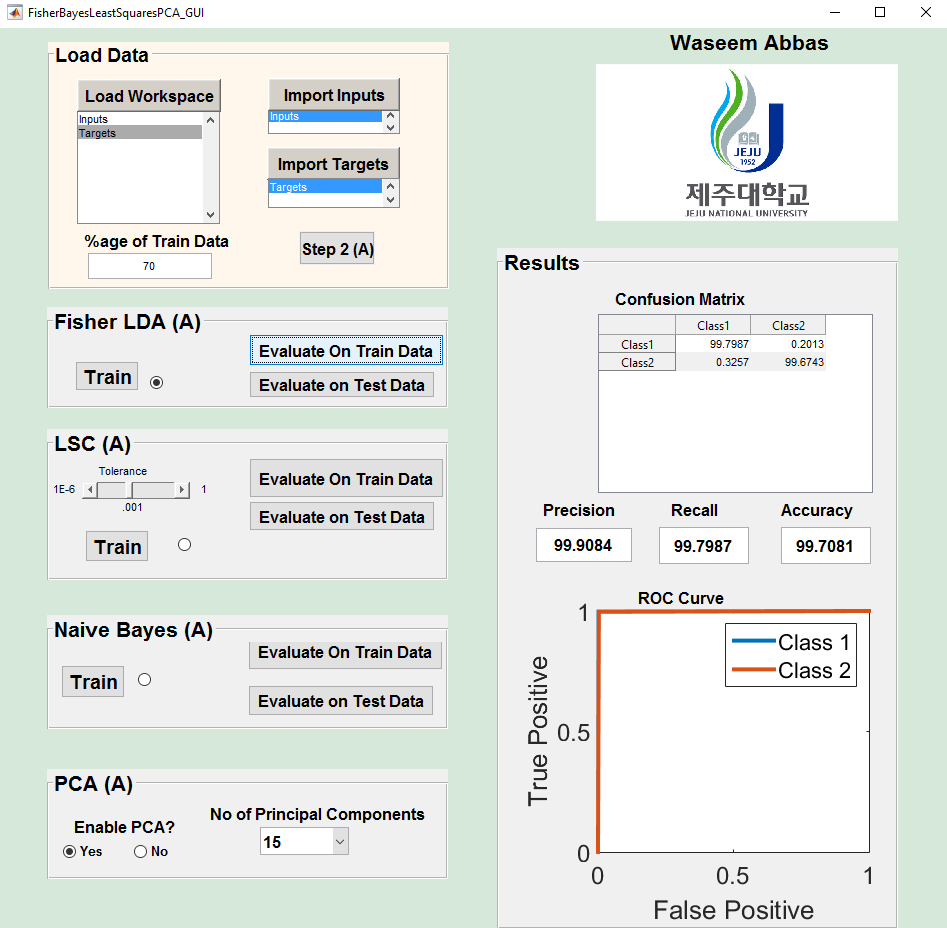
**M:** Mapping matrix of principal components. It can be used to reduce dimensions of input feature vector. Let’s input feature vector dimensions are and our target dimension is such that. Dimension of M in this case is. We can reduce dimensions of input matrix A by projecting it along as follows

**lambda:** Eigen values of retained principal components

##### help

Help file of this function can be accessed by typing ‘help prin\_comp\_analysis’ in command window

## GUI



6

5

4

3

2

1

Figure 2: GUI for Fisher, Least Squares and Naive Bayesian Classifiers

There are total 6 fields of the GUI. Field 2, 3 and 4 are the actual classifiers. Panel 2 is for fisher classifier, panel 3 is for least square classifier and panel 4 is for naïve Bayesian classifier. Panel 1, 5 and 6 are shared by the three classifiers mentioned. All the panels are described below

### Panel 1: Loading Data

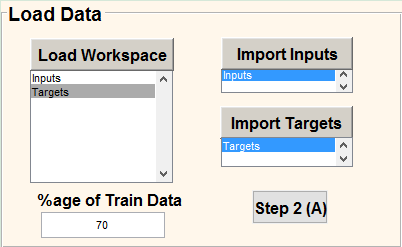


Figure 3: Loading Data panel

This panel is used for importing Input data and Targets to MATLAB GUI. Inputs and Targets can be selected in following manner.

* First of all, load your input data and target matrix to MATLAB workspace.
* Click on “Import Workspace” button. It will import all variables present in the workspace to the list below the button.
* Select the input variable and click on “Import Inputs”. The GUI will go to workspace and import variable with the same name into GUI space. It will also activate button for importing targets.
* Select the target variable and click on “Import Targets”. The GUI will go to workspace and import variable with the same name into GUI space. It will then check for compatibility of inputs and targets. If number of rows of inputs and targets is not equal, it means the user has imported wrong inputs and targets, so the GUI will generate an error message, stating either inputs or targets are not in recommended format or they have imported wrong inputs or targets.
* User can then change percentage of data they want to keep for training. Default is 70%. Once this is done, click on “Step 2” button. It will activate all the panels

### Panel 2: Fisher Classifier

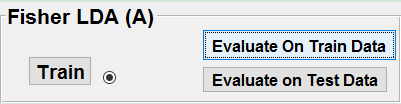


Figure 4: Panel for Fisher Classifier

This panel runs the fisher classifier in background. Once the inputs and targets are imported, this panel can be used to train a fisher classifier on training data. The classifier is then stored in memory by name “fisher.mat”. It can be loaded for later use, but it will be deleted if the GUI is launched again. The trained fisher object can be then used to see its performance on training data and test data by clicking the appropriate buttons.

### Panel 3: Least Square Classifier

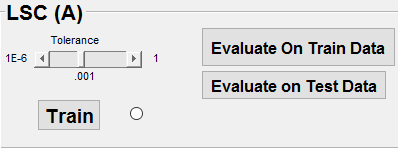


Figure 5: Panel for Least Squares Classifier

This panel is used to train multi-class linear least squares classifier in the background. After importing inputs and targets, this panel can be used to train a least square classifier on training data. As least squares classifier need a least-squares error tolerance, it can be set by setting the value of slider named “Tolerance”. User can pick any value of Tolerance in the range .The classifier is then stored in memory by name “lsc.mat”. It can be loaded for later use, but it will be deleted if the GUI is launched again. The trained LSC object can be then used to see its performance on training data and test data by clicking the appropriate buttons.

### Panel 4: Naïve Bayesian Classifier

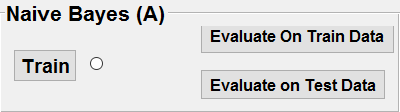


Figure 6: Panel for Naive Bayesian Classifier

This panel is used to train multi-class naïve Bayesian classifier in the background. After importing inputs and targets, this panel can be used to train the classifier on training data. The classifier is then stored in memory by name “naivebayes.mat”. It can be loaded for later use, but it will be deleted if the GUI is launched again. The trained Bayesian object can be then used to see its performance on training data and test data by clicking the appropriate buttons.

### Panel 5: Principal Component Analysis

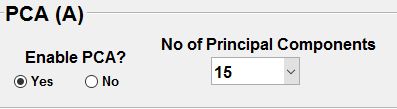
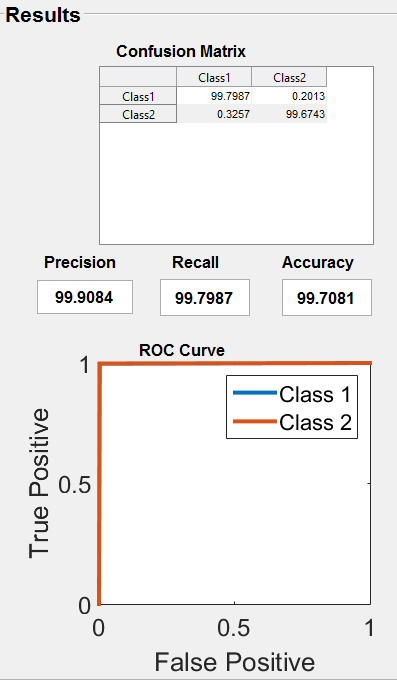


Figure 7: Panel for Principal Component Analysis

This panel is used to perform Principal Component Analysis (PCA) for dimensionality reduction on input data. By default, this option is disabled, but if the user wants to perform PCA, enable it by clicking on “Yes” radio button. The “Number of Principal Components” popup menu will become visible, and it will show total number of features present in the input data. Let’s say there are total 30 features, it will select 15 (half) features by default. If number of features is smaller than 4, the PCA will not be performed. It is to be noted that this panel is only a supporting tool to the three classifiers mentioned under panel 2, 3 and 4.

### Panel 6: Results



This panel is used to visualize the performance of the classifiers. The confusion matrix of predicted classes, accuracies and ROC curves can be visualized and observed in this panel. This panel is accessible to all three classifiers.

#### Confusion Matrix

A confusion matrix, also known as an error matrix, is a specific table layout that allows visualization of the performance of an algorithm, typically a supervised learning. Each column of the matrix represents the instances in a predicted class while each row represents the instances in an actual class (or vice-versa). The name stems from the fact that it makes it easy to see if the system is confusing two classes (i.e. commonly mislabeling one as another).

#### Precision

Precision signifies the fraction of class labels retrieved that are relevant. Range [0 100]

#### Recall

Recall signifies the fraction of relevant class labels that are retrieved. Range [0 100]

#### Accuracy

It signifies the fraction of correctly classified labels. Range [0 100]

#### ROC Curve

In terms of classification, an ROC curve for a class is a graphical representation of relationship between true positives and false positives.

## Running the GUI

Step 1: Start from panel 1. Import inputs and targets data using panel 1. Select percentage of data to be used for training and then click on step 2.

Step 2: All the panels will be activated at this stage. Select the classifier which you want to train and test by clicking on the radio button on its panel. Let’s say you have selected Fisher. All the other classifiers will be disabled automatically.

1. If you want to train the classifier without reducing dimension, just click on “Train”. Once the classifier is trained, the GUI will give you a message about training success.
2. If you want to reduce dimensionality of the input data, go to “PCA” panel. Enable PCA by clicking on “Yes” radio button below ‘Enable PCA’. A popup menu will become visible on right side of this radio button. Select number of dimensions (or principal components) you want to keep.
3. Once you have selected number of principal components to retain, click on “Train” button in Fisher panel. It will train the Fisher classifier with reduced dimensions.

Step 3: Once the training is complete, you can visualize performance of the classifier on training data as well as on test data by clicking on appropriate buttons.

Step 4: Repeat steps 1 to 3 if you want to train any other classifier.

Important notice: As these classifiers are more or less variants of linear classifiers, they project input data onto one dimensional line and then perform classification. Naïve Bayesian Classifier works on prior and posterior probabilities, so there are not much parameters to be tuned. Therefore, on some datasets, the classifier cannot achieve proper convergence. If that’s the case, the classifier training completion message might pop up, even the classifier object will be stored. But once we try it to run it on data, the GUI will generate errors because the classifier output will not be as expected by the GUI. If that’s the case, please use another classifier (like ELM or DBN) on such data.