Unit-5

***Pattern Recognition* :** Introduction, Design principles of pattern recognition system, Statistical Pattern recognition, Parameter estimation methods - Principle Component Analysis (PCA) and Linear Discriminant Analysis (LDA), Classification Techniques – Nearest Neighbor (NN) Rule, Bayes Classifier, Support Vector Machine (SVM), K – means clustering.

5.1 What is Pattern Recognition?

Pattern recognition focuses on the problem of **how to automatically classify physical objects or abstract multidimensional patterns (*n* points in *d* dimensions) into known or possibly unknown categories.** Traditional example applications include *character recognition, handwriting recognition, document classification, fingerprint classification, speech and speaker recognition, white blood cell (leukocyte) classification, military target recognition*, and object recognition by machine vision systems in assembly lines among others.

Pattern recognition is the scientific discipline **whose goal is the classification of *objects* into a number of categories or *classes*.** Depending on the application, these objects can be images or signal waveforms or any type of measurements that need to be classified.

Pattern recognition *is a branch of machine learning that focuses on the recognition of patterns and regularities in data*. Pattern recognition is the science of making inferences from perceptual data, using tools from *statistics, probability, computational geometry, machine learning, signal processing, and algorithm design*. Thus, it is of central importance to artificial intelligence and computer vision, and has far-reaching applications in *engineering, science, medicine, and business.*

It is natural that we should seek *to design and build machines that can recognize patterns*. From automated speech recognition, fingerprint identification, optical character recognition, DNA sequence identification, and much more, it is clear that reliable, accurate pattern recognition by machine would be immensely useful.

1.2 Basics

Feature can be defined as any distinctive aspect, quality or characteristic which, may be symbolic (i.e., color) or numeric (i.e., height). The combination of d features is represented as a d-dimensional column vector called a **feature vector.** The d-dimensional space defined by the feature vector is called **feature space.** Objects are represented **as points in feature space**. This representation is called a scatter plot [3]. Pattern is defined as **composite of features** that are characteristic of an individual. In classification, a pattern is a pair of variables {x, w} where x is a collection of observations or features (feature vector) and w is the concept behind the observation (label).

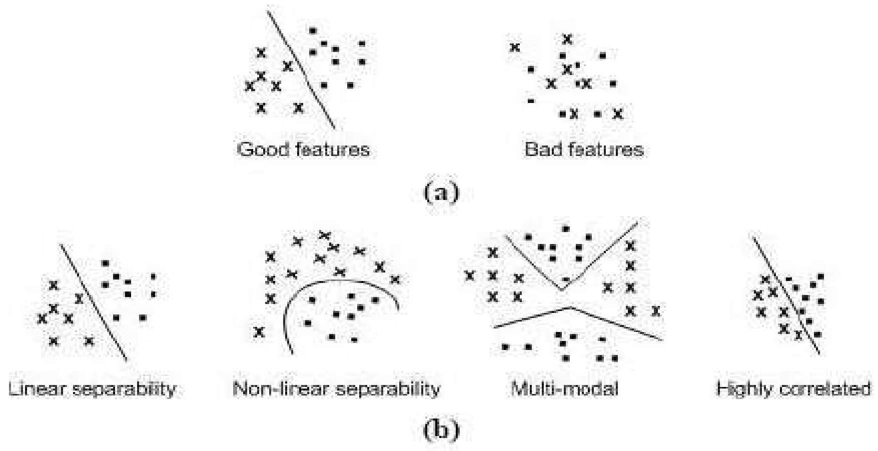


Figure 1.1: Characteristic (feature); a. the distinction between good and poor features, and b. feature properties.

The goal of a classifier is to partition feature space into class-labeled decision regions. Borders between decision regions are called decision boundaries (Figure 1.2).

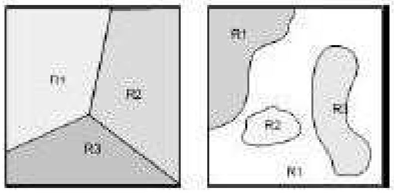


Figure 1.2: Classifier and decision boundaries.

If the characteristics or attributes of a class are known, individual objects might be identified as belonging or not belonging to that class. The objects are assigned to classes by observing patterns of distinguishing characteristics and comparing them to a model member of each class. Pattern recognition involves the extraction of patterns from data, their analysis and, finally, the identification of the category (class) each of the patterns belongs to.

1.3 An Example

Suppose that a fish-packing plant wants to automate the process of sorting incoming fish on a conveyor belt according to species. As a pilot project, it is decided to try to separate sea bass from salmon using optical sensing (Figure 1.4).

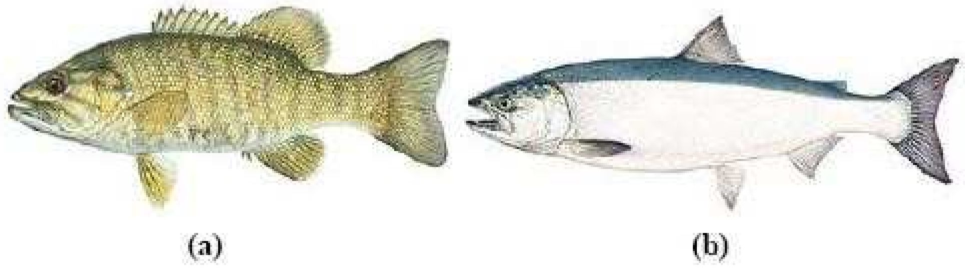


Figure 1.4: The objects to be classified; a. Sea bass, and b. salmon.

We set up a camera (see Figure 1.5), take some sample images, and begin to note some physical differences between the two types of fish, length, lightness, width, number and shape of fins, position of the mouth, and so on, and these suggest features to explore for use in our classifier. We also notice noise or variations in the images, variations in lighting, and position of the fish on the conveyor, even static due to the electronics of the camera itself.

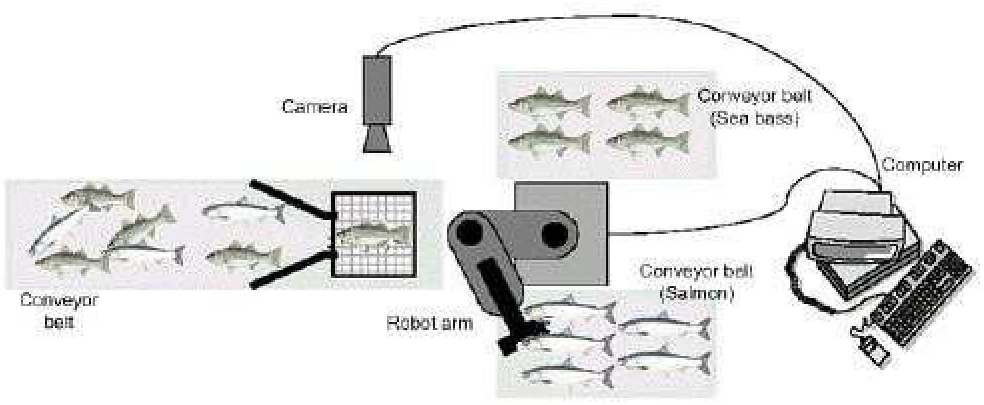


Figure 1.5: The fish-packing system.

1.4 Approaches to Pattern Recognition

There are two fundamental approaches for implementing a pattern recognition system: **statistical and structural**. Each approach employs different techniques to implement the description and classification tasks

Statistical pattern recognition draws from established concepts in *statistical decision theory to discriminate among data from different groups based upon quantitative features of the data*. Examples of statistical feature extraction techniques include mean and standard deviation computations, frequency count summarizations, Karhunen-Lóeve transformations, Fourier transformations, wavelet transformations, and Hough transformations

Structural pattern recognition, sometimes referred to as *syntactic pattern recognition due to its origins in formal language theory, relies on syntactic grammars to discriminate among data* *from different groups based upon the morphological interrelationships (or interconnections) present within the data.*

The semantics related with each feature are determined by the coding scheme (i.e., the selection of morphologies) used to identify primitives in the data. Feature vectors generated by structural pattern recognition systems contain a variable number of features (one for each primitive extracted from the data) in order to accommodate the presence of superfluous structures which have no impact on classification.

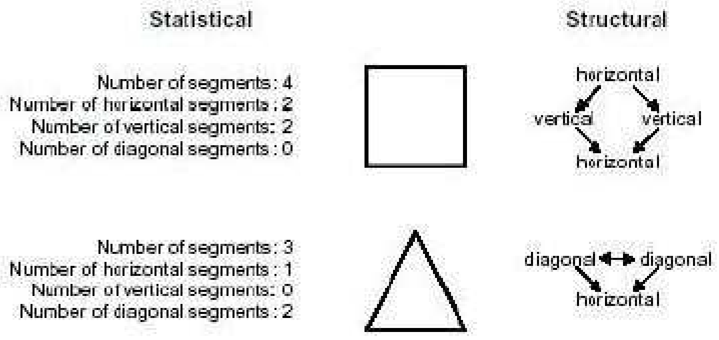
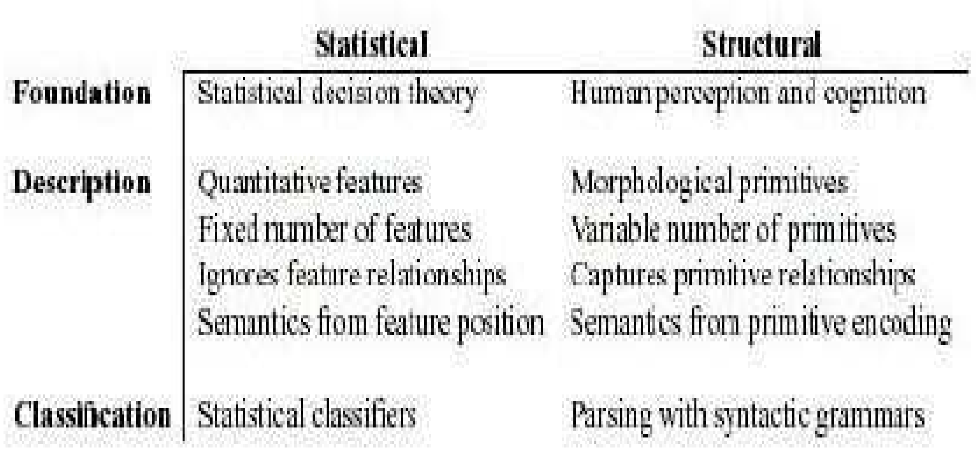

Statistical
Structural
Number of segments : 4
Number of horizontal segments : 2
Number of vertical segments: 2
Number of diagonal segments : 0
horizontal
vertical
vertical
horizontal
Number of segments : 3
Number of horizontal segments : 1
Number of vertical segments: 0
Number of diagonal segments : 2
diagonal
diagonal
horizontal

Figure 1.11: The statistical and structural approaches to pattern recognition applied to a common identification problem.

The essential dissimilarities are two-fold:

(1) The description generated by the statistical approach is **quantitative**, while the structural approach produces a description composed of **sub patterns or building blocks**.

(2) The statistical approach discriminates based upon *numeric differences among features from different group*s, while grammars are used by the structural approach to define a *language encompassing the acceptable configurations of primitives for each group.*



	Statistical	Structural
Foundation	Statistical decision theory	Human perception and cognition
Description	Quantitative features	Morphological primitives
	Fixed number of features	Variable number of primitives
	Ignores feature relationships	Captures primitive relationships
	Semantics from feature position	Semantics from primitive encoding
Classification	Statistical classifiers	Parsing with syntactic grammars

1.5 Pattern Recognition Systems

A typical pattern recognition system contains a *sensor, a preprocessing mechanism (segmentation), a feature extraction mechanism (manual or automated), a classification or description algorithm, and a set of examples (training set) already classified or described (post-processing)*(Figure 1.3).

1.5.1 Sensing

The sensors in a system which receive the data input, and they may vary depending on the purpose of the system. They are usually some form of transducers such as a camera or a microphone.

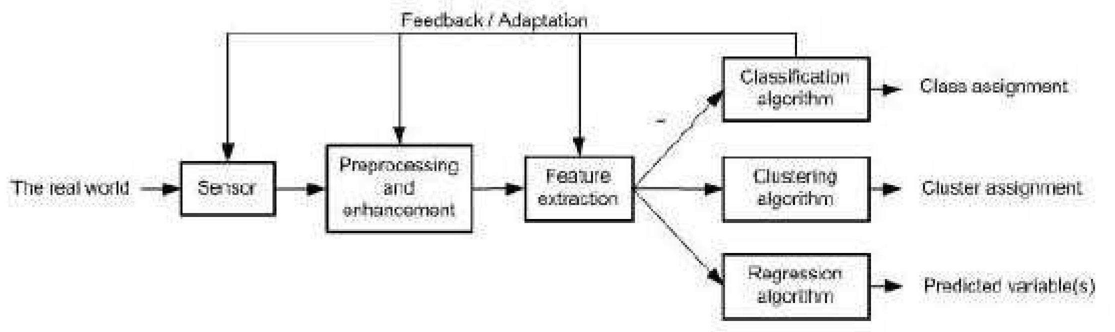


Figure 1.3: A pattern recognition system.

1.5.2 Segmentation and Grouping

How can we segment the images before they have been categorized, or categorize them before they have been segmented? It seems we need a way to know when we have switched from one model to another, or to know when we just have background or no category. How can this be done? Segmentation is one of the deepest problems in pattern recognition. Closely related to the problem of segmentation is the problem of recognizing or grouping together the various pans of a composite object.

1.5.3 Feature Extraction

The goal of the feature extractor is to characterize an object to be recognized by measurements whose values are very similar for objects in the same category, and very different for objects in different categories.

1.5.4 Classification

The task of the classifier component system is to use the feature vector provided by the feature extractor to assign the object to a category.

1.5.5 Post Processing

The post-processor uses the output of the classifier to decide on the recommended action. However, it may be much better to recommend actions that will minimize the total expected cost, which is called the risk.

- How do we incorporate knowledge about costs and how will this affect our classification decision?
- Can we estimate the total risk and thus tell when our classifier is acceptable even before we field it?
- Can we estimate the lowest possible risk of any classifier; to see how close ours meets this ideal, or whether the problem is simply too hard overall?

5.2 Principal Component Analysis

Principal Component Analysis, or simply PCA, *is a statistical procedure concerned with elucidating the covariance structure of a set of variables. In particular it allows us to identify the principal directions in which the data varies.*

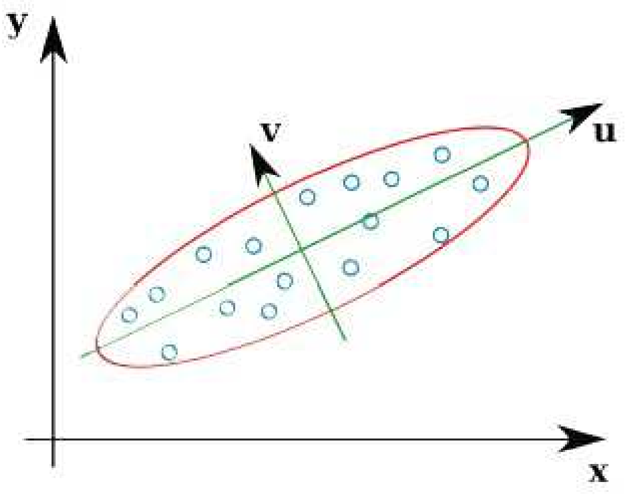


Figure 1: PCA for Data Representation

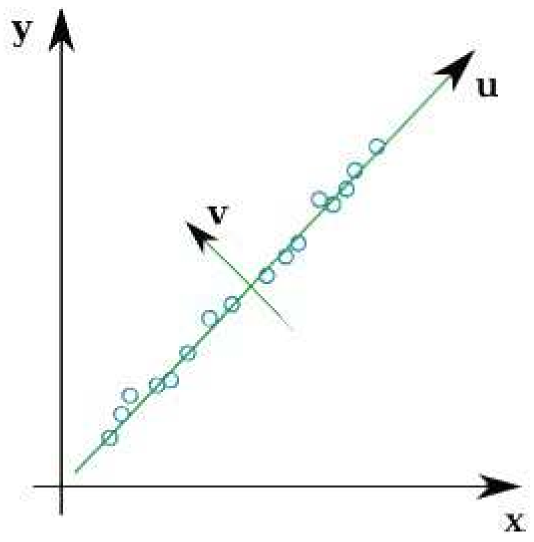


Figure 2: PCA for Dimension Reduction

For example, in figure 1, suppose that the triangles represent a two variable data set which we have measured in the X-Y coordinate system. The principal direction in which the data varies is shown by the U axis and the second most important direction is the V axis orthogonal to it. If we place the U- V axis system at the mean of the data it gives us a compact representation. If we transform each (X, Y) coordinate into its corresponding (U, V) value, the data is de-correlated, meaning that the co-variance between the U and V variables is zero.

If the variation in a data set is caused by some natural property, or is caused by random experimental error, then we may expect it to be normally distributed. In this case we show the nominal extent of the normal distribution by a hyper-ellipse (the two dimensional ellipse in the example).

<u>Computing the Principal Components:-</u>

In computational terms the principal components are found by calculating the eigenvectors and eigenvalues of the data covariance matrix. This process is equivalent to finding the axis system in which the co-variance matrix is diagonal. The eigenvector with the largest eigenvalue is the direction of greatest variation, the one with the second largest eigenvalue is the (orthogonal) direction with the next highest variation and so on.

Let A be an n * n matrix. The Eigen values of A are defined as the roots of:

$$determinant(\boldsymbol{A} - \lambda \boldsymbol{I}) = |(\boldsymbol{A} - \lambda \boldsymbol{I})| = 0$$

where I is the n * n identity matrix. This equation is called the characteristic equation (or characteristic polynomial) and has n roots.

Let λ be an Eigen value of A. Then there exists a vector x such that:

$$\boldsymbol{A}\boldsymbol{x} = \lambda \boldsymbol{x}$$

The vector x is called an eigenvector of A associated with the Eigen value λ. Suppose we have a 3 * 3 matrix A with eigenvectors x1, x2, x3, and eigenvalues λ1, λ2, λ3 so:

$$\boldsymbol{A}\boldsymbol{x}_1 = \lambda_1 \boldsymbol{x}_1 \qquad \boldsymbol{A}\boldsymbol{x}_2 = \lambda_2 \boldsymbol{x}_2 \qquad \boldsymbol{A}\boldsymbol{x}_3 = \lambda_3 \boldsymbol{x}_3$$

Putting the eigenvectors as the columns of a matrix gives:

$$\boldsymbol{A}\begin{bmatrix} \boldsymbol{x}_1 & \boldsymbol{x}_2 & \boldsymbol{x}_3 \end{bmatrix} = \begin{bmatrix} \boldsymbol{x}_1 & \boldsymbol{x}_2 & \boldsymbol{x}_3 \end{bmatrix}\begin{bmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & \lambda_3 \end{bmatrix}$$

writing:

$$\Phi = \begin{bmatrix} \boldsymbol{x}_1 & \boldsymbol{x}_2 & \boldsymbol{x}_3 \end{bmatrix} \qquad \Lambda = \begin{bmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & \lambda_3 \end{bmatrix}$$

gives us the matrix equation:

$$\boldsymbol{A}\Phi = \Phi\Lambda$$

We normalised the eigenvectors to unit magnitude, and they are orthogonal, so:

$$\Phi\Phi^T = \Phi^T\Phi = \boldsymbol{I}$$

which means that:

$$\Phi^T \boldsymbol{A}\Phi = \Lambda$$

and:

$$A = \Phi\Lambda\Phi^T$$

Now let us consider how this applies to the covariance matrix in the PCA process. Let Σ be an $n \times n$ covariance matrix. There is an orthogonal $n \times n$ matrix Φ whose columns are eigenvectors of Σ and a diagonal matrix Λ whose diagonal elements are the eigenvalues of Σ, such that

$$\Phi^T \Sigma \Phi = \Lambda$$

We can look on the matrix of eigenvectors Φ as a linear transformation which, in the example of figure 1 transforms data points in the $[X, Y]$ axis system into the $[U, V]$ axis system. In the general case the linear transformation given by Φ transforms the data points into a data set where the variables are uncorrelated. The correlation matrix of the data in the new coordinate system is Λ which has zeros in all the off diagonal elements.

Face Recognition

Face recognition is one example where principal component analysis has been extensively used, primarily for reducing the number of variables. Let us consider the 2D case where we have an input image, and wish to compare this with a set of data base images to find the best match. We assume that the images are all of the same resolution and are all equivalently framed. (ie the faces appear at the same place and same scale in the images). Each pixel can be considered a variable thus we have a very high dimensional problem which can be simplified by PCA.

Formally, in image recognition an input image with n pixels can be treated as a point in an n-dimensional space called the image space. The individual ordinates of this point represent the intensity values of each pixel of the image and form a row vector: $p_{\boldsymbol{x}} = (i_1, i_2, i_3, ...i_n)$ This vector is formed by concatenating each row of image pixels, so for a modest sized image, say 128 by 128 resolution it will dimension 16384. For example:

$$= \begin{bmatrix} 150 & 152 & \cdot & 151 \\ 131 & 133 & \cdot & 72 \\ \cdot & \cdot & \cdot & \cdot \\ 144 & 171 & \cdot & 67 \end{bmatrix} \quad 128 \times 128$$

becomes the vector:

$$[150, 152, \cdots 151, 131, 133, \cdots 72, \cdots 144, 171, \cdots 67]_{16K}$$

Clearly this number of variables is far more than is required for the problem. Most of the image pixels will be highly correlated. For example if the background pixels are all equal then adjacent background pixels are exactly correlated. Thus we need to consider how to achieve a reduction in the number of the variables.

Dimension Reduction

Lets consider an application where we have N images each with n pixels. We can write our entire data set as an $N \times n$ data matrix $\boldsymbol{D}$. Each row of $\boldsymbol{D}$ represents one image of our data set. For example we may have:

$$\boldsymbol{D} = \begin{bmatrix} 150 & 152 & \cdots & 254 & 255 & \cdots & 252 \\ 131 & 133 & \cdots & 221 & 223 & \cdots & 241 \\ . & . & & . & . & & . \\ . & . & & . & . & & . \\ 144 & 171 & \cdots & 244 & 245 & \cdots & 223 \end{bmatrix} N \times n$$

The first step in PCA is to move the origin to mean of the data. In this application we achieve this by finding a mean image $\boldsymbol{\mu}$ by averaging the columns of $\boldsymbol{D}$. We then subtract the mean image from each image of the data set (ie each row of D) to create the mean centered data vector which we write as U. Let us suppose that the mean centered image is:

$$[120 \;\; 140 \;\; \cdots \;\; 230 \;\; 230 \;\; \cdots \;\; 240]$$

Then we have that:

$$U = \begin{bmatrix} 30 & 12 & \cdots & 24 & 25 & \cdots & 12 \\ 11 & -7 & \cdots & -9 & -7 & \cdots & 1 \\ . & . & & . & . & & . \\ 24 & 31 & \cdots & 14 & 15 & \cdots & -17 \end{bmatrix} N \times n$$

It is very easy to compute the covariance matrix from the mean centered data matrix. It is just

$$\Sigma = \boldsymbol{U}^T \boldsymbol{U}/(N-1)$$

and has dimension $n \times n$. We now calculate the eigenvectors and eigenvalues of Σ using the standard technique outlined above. That is to say we solve for Φ and Λ that satisfy:

$$\Sigma = \Phi \Lambda \Phi^T$$

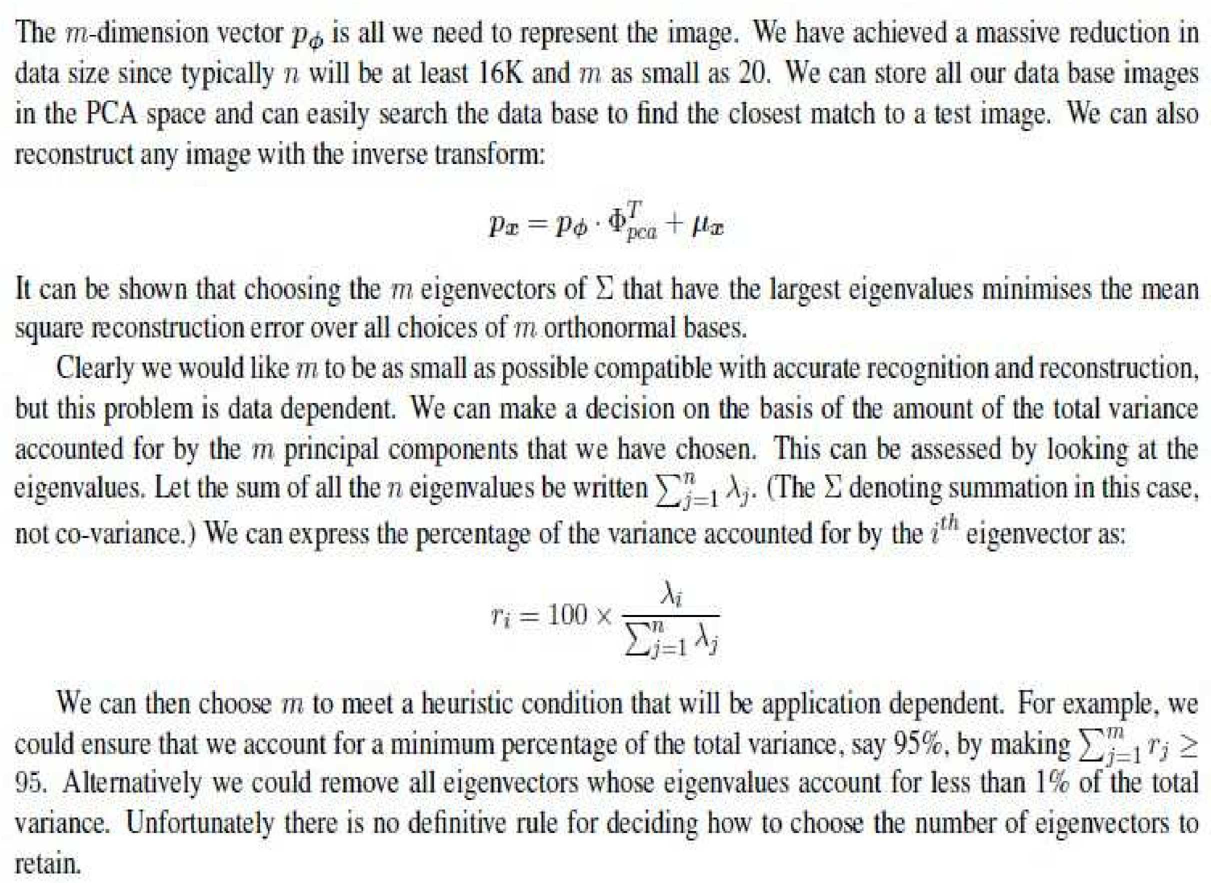
If we normalise the eigenvectors, then the system of vectors Φ forms an orthonormal basis, that is to say:

$$\forall \phi_i \, \phi_j \in \Phi, \; \phi_i \cdot \phi_j = \begin{cases} 1 & if\; i = j \\ 0 & if\; i \neq j \end{cases}$$

It is in effect an axis system in which we we can represent our data in a compact form.

We can achieve size reduction by choosing to represent our data we fewer dimensions. Normally we choose to use the set of $m (m \leq n)$ eigenvectors of Σ which have the m largest eigenvalues. Typically for face recognition system m will be quite small (around 20-50 in number). We can compose these in an $n \times m$ matrix $\Phi_{pca} = [\phi_1, \phi_2, \phi_3 \cdots \phi_m]$ which performs the PCA projection. For any given image $\boldsymbol{p_x} = (i_1, i_2, i_3, ...i_n)$ we can find a corresponding point in the PCA space by computing

$$\boldsymbol{p_\phi} = (\boldsymbol{p_x} - \boldsymbol{\mu_x}) \cdot \Phi_{pca}$$



The m-dimension vector p_ϕ is all we need to represent the image. We have achieved a massive reduction in data size since typically n will be at least 16K and m as small as 20. We can store all our data base images in the PCA space and can easily search the data base to find the closest match to a test image. We can also reconstruct any image with the inverse transform:

$$p_x = p_\phi \cdot \Phi_{pca}^T + \mu_x$$

It can be shown that choosing the m eigenvectors of Σ that have the largest eigenvalues minimises the mean square reconstruction error over all choices of m orthonormal bases.

Clearly we would like m to be as small as possible compatible with accurate recognition and reconstruction, but this problem is data dependent. We can make a decision on the basis of the amount of the total variance accounted for by the m principal components that we have chosen. This can be assessed by looking at the eigenvalues. Let the sum of all the n eigenvalues be written $\sum_{j=1}^{n} \lambda_j$. (The Σ denoting summation in this case, not co-variance.) We can express the percentage of the variance accounted for by the i^{th} eigenvector as:

$$r_i = 100 \times \frac{\lambda_i}{\sum_{j=1}^{n} \lambda_j}$$

We can then choose m to meet a heuristic condition that will be application dependent. For example, we could ensure that we account for a minimum percentage of the total variance, say 95%, by making $\sum_{j=1}^{m} r_j \geq 95$. Alternatively we could remove all eigenvectors whose eigenvalues account for less than 1% of the total variance. Unfortunately there is no definitive rule for deciding how to choose the number of eigenvectors to retain.

5.3 Linear Discriminant Analysis (LDA)

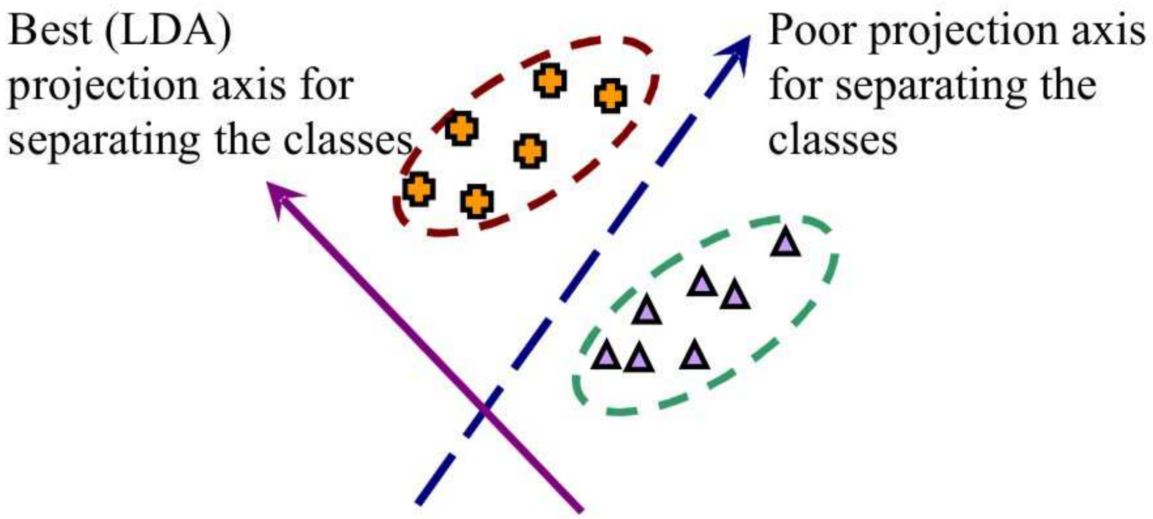
Linear Discriminant Analysis, or simply LDA, is a well-*known classification technique that has been used successfully in many statistical pattern recognition problems. It was developed by Ronald Fisher*, who was a professor of statistics at University College London, and is sometimes called Fisher Discriminant Analysis (FDA).

- *The primary purpose of LDA is to separate samples of distinct groups. We do this by transforming the data to a different space that is optimal for distinguishing between the classes.*
- LDA is used to find a linear combination of *features which characterizes or separates two or more classes of objects or events*. The resulting combination may be used as a linear classifier or, more commonly, for dimensionality reduction before later classification.
- LDA is also closely related to *principal component analysis (PCA) and factor analysis* in that they both look for linear combinations of variables which best explain the data.
- *LDA explicitly attempts to model the difference between the classes of data. PCA on the other hand does not take into account any difference in class,* and factor analysis builds the feature combinations based on differences rather than similarities.
- Discriminant analysis is also different from factor analysis in that *it is not an interdependence technique: a distinction between independent variables and dependent variables (also called criterion variables) must be made.*

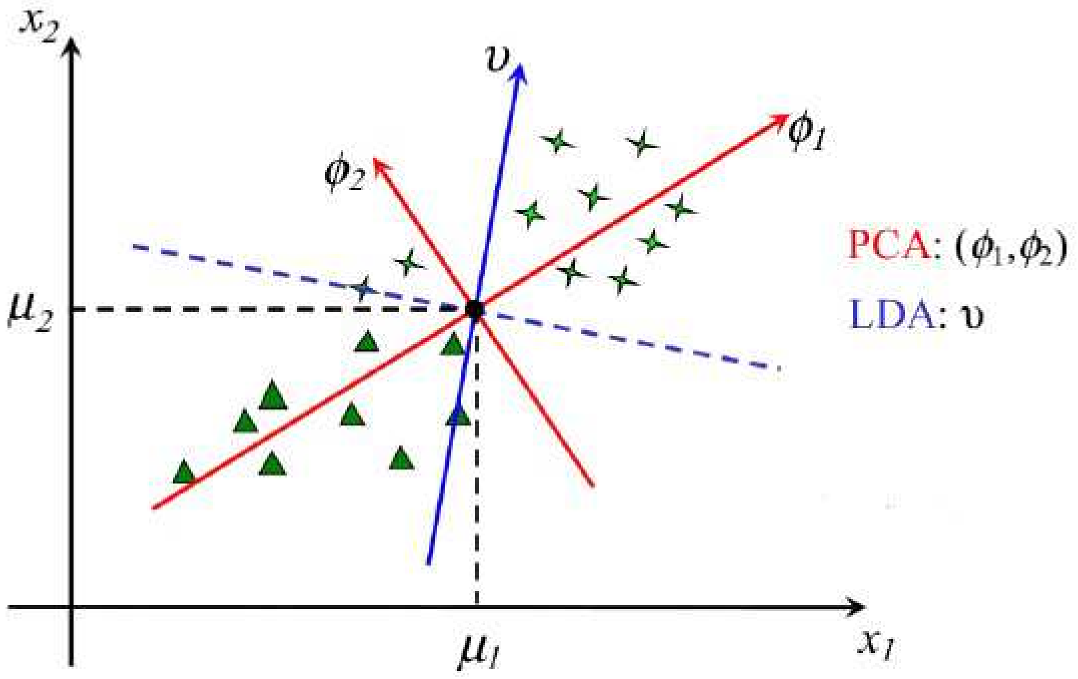
- LDA works when the measurements made on independent variables for each observation are continuous quantities. When dealing with categorical independent variables, the equivalent technique is discriminant correspondence analysis.

Example

For example, let us suppose we have a simple two variable problem in which there are two classes of objects.


Best (LDA) projection axis for separating the classes
Poor projection axis for separating the classes

Instead, the approach of the LDA is to project all the data points into new space, normally of lower dimension, which maximises the between-class separability while minimising their within-class variability.



The above figure illustrates the difference between PCA and LDA. In PCA we take the data as a whole and do not consider any division into classes. The axes are optimal for representing the data as they indicate where the maximum variation actually lies. The LDA axis is optimal for distinguishing between the different classes. In general the number of axes that can be computed by the LDA method is one less than the number of classes in the problem. In the figure the ϕ_1 axis is less effective for separating the classes than the ν (LDA) axis. The ϕ_2 axis is clearly completely useless for classification.

The first step in the LDA is finding two scatter matrices referred to as the "between class" and "within class" scatter matrices. Suppose in a given problem we have g different classes or (sample groups). Each sample group π_i has a class mean, which we denote $\overline{x_i}$

$$\overline{x_i} = \frac{1}{N_i}\sum_{j=1}^{N_i} x_{i,j}$$

where there are N_i data points in class π_i. We can also define a sample group covariance matrix:

$$\Sigma_i = \frac{1}{N_i - 1}\sum_{j=1}^{N_i}(x_{i,j} - \overline{x_i})(x_{i,j} - \overline{x_i})^T$$

and we can define a grand mean for the whole data set:

$$\overline{x} = \frac{1}{N}\sum_{i=1}^{g} N_i\overline{x_i} = \frac{1}{N}\sum_{i=1}^{g}\sum_{j=1}^{N_i} x_{i,j}$$

The between class scatter matrix is defined as:

$$S_b = \sum_{i=1}^{g} N_i(\overline{x_i} - \overline{x})(\overline{x_i} - \overline{x})^T$$

We could normalise this into a between class covariance matrix by dividing by $N - 1$, but it is not necessary to do so. If all classes were the same size then the $N - i$ could be removed from the equation. The within class matrix is defined as follows:

$$S_w = \sum_{i=1}^{g}(N_i - 1)\Sigma_i = \sum_{i=1}^{g}\sum_{j=1}^{N_i}(x_{i,j} - \overline{x_i})(x_{i,j} - \overline{x_i})^T$$

The S_w matrix is computed by pooling the estimates of the covariance matrices of each class. Since each Σ_i has rank $N_i - 1$ its rank can be at most $N - g$.

The main objective of LDA is to find a projection matrix Φ_{lda} that maximises the ratio of the determinant of S_b to the determinant of S_w. This can be written:

$$\Phi_{lda} = \arg\max_{\Phi}\frac{|\Phi^T S_b \Phi|}{|\Phi^T S_w \Phi|}$$

This ratio is known as Fishers criterion. To get an intuition of what is means, note that the determinant of the co-variance matrix tells us how much variance a class has. For example, for the co-variance matrix in the PCA (diagonal) projection, the value of the determinant is just the product of the diagonal elements which are the individual variable variances. The determinant has the same value under any ortho-normal projection. So Fishers criterion tries to find the projection that maximises the variance of the class means and minimises the variance of the individual classes.

It has been shown that Φ_{lda} is the solution of the following equation:

$$S_b\Phi - S_w\Phi\Lambda = 0$$

Multiplying by the inverse of S_w we get:

$$\begin{gathered} S_w^{-1}S_b\Phi - S_w^{-1}S_w\Phi\Lambda = 0 \\ S_w^{-1}S_b\Phi - \Phi\Lambda = 0 \\ S_w^{-1}S_b\Phi = \Phi\Lambda \end{gathered}$$

Thus, if S_w is a non-singular matrix, and can be inverted, then the Fisher's criterion is maximised when the projection matrix Φ_{lda} is composed of the eigenvectors of:

$$S_w^{-1}S_b$$

Notice that there will be at most $g - 1$ eigenvectors with non-zero real corresponding eigenvalues. This is because there are only g points to estimate S_b. This again can represent a massive reduction in the dimensionality of the problem. In face recognition for example there may be several thousand variables, but only a few hundred classes.

Once the projection is found all the data points can be transformed to the new axis system along with the class means and co-variances. Allocation of a new point to a class can be done using a distance measure such as the Mahalanobis distance. We will look at methods of classifying in the next lecture. LDA is essentially just a projection method.

Classification Techniques

In machine learning and statistics, *classification is the problem of identifying to which of a set of categories (sub-populations) a new observation belongs, on the basis of a training set of data containing observations (or instances) whose category membership is known.*

- An example would be assigning a given email into "spam" or "non-spam" classes or assigning a diagnosis to a given patient as described by observed characteristics of the patient (gender, blood pressure, presence or absence of certain symptoms, etc.).

Relation to other problems

Classification and clustering are examples of the more general problem of pattern recognition, *which is the assignment of some sort of output value to a given input value.* Other examples are regression, *which assigns a real-valued output to each input; sequence labeling, which assigns a* *class to each member of a sequence of values* (for example, part of speech tagging, which assigns a part of speech to each word in an input sentence); parsing, which assigns a parse tree to an input sentence, describing the syntactic structure of the sentence; etc.

5.4 Nearest Neighbor (NN) Rule

Nearest neighbor classifiers is a class of non-parametric methods used in statistical classification (or pattern recognition). The method classifies objects based on closest training examples in the feature space.

Statistical setting

Suppose we have pairs $(X,Y),(X_1,Y_1),\ldots,(X_n,Y_n)$ taking values in $\mathbb{R}^d\times\{1,2\}$, where Y is the class label of X, so that $X|Y=r\sim P_r$ for $r=1,2$ (and probability distributions P_r). Given some norm $\|\cdot\|$ on $\mathbb{R}^d$ and a point $x\in\mathbb{R}^d$, let $(X_{(1)},Y_{(1)}),\ldots,(X_{(n)},Y_{(n)})$ be a reordering of the training data such that $\|X_{(1)}-x\|\leq\ldots\leq\|X_{(n)}-x\|$.

The 1-nearest neighbor classifier

The most intuitive nearest neighbor type classifier is the one nearest neighbor classifier that assigns a point x to the class of its closest neighbor in the feature space, that is $C_n^{1nn}(x)=Y_{(1)}$.

As the size of training data set approaches infinity, the one nearest neighbor classifier guarantees an error rate of no worse than twice the Bayes error rate.

The k-nearest neighbor classifier

The k-nearest neighbor classifier assigns a point x to a particular class based on a majority vote among the classes of the $k$nearest training points to x.

Properties

There are many results on the error rate of the $k$nearest neighbor classifiers. The k-nearest neighbor classifier is strongly (that is for any joint distribution on (X,Y)) consistent provided $k:=k_n$ diverges and k_n/nconverges to zero as $n\to\infty$

Let C_n^{knn} denote the k nearest neighbor classifier based on a training set of size$n$. Under certain regularity conditions, the excess risk yields the following asymptotic expansion

$$\mathcal{R}_{\mathcal{R}}(C_n^{knn}) - \mathcal{R}_{\mathcal{R}}(C^{Bayes}) = \left\{ B_1 \frac{1}{k} + B_2 \left(\frac{k}{n}\right)^{4/d} \right\} \{1 + o(1)\},$$

for some constants B_1 and B_2.

The choice $k^* = \lfloor Bn^{\frac{4}{d+4}} \rfloor$offers a tradeoff between the two terms in the above display, for which the k^*-nearest neighbor error converges the Bayes error at the optimal (minimax) rate $\mathcal{O}(n^{-\frac{4}{d+4}})$.

5.5 Naive Bayes classifier

In machine learning, **naive Bayes classifiers** are a family of simple *probabilistic classifiers based on applying Bayes' theorem with strong (naive) independence assumptions between the features.*

- In simple terms, a naive Bayes classifier assumes that the *value of a particular feature is unrelated to the presence or absence of any other feature, given the class variable.*

  For example, a fruit may be considered to be an apple if it is red, round, and about 3" in diameter. A naive Bayes classifier considers each of these features to contribute independently to the probability that this fruit is an apple, regardless of the presence or absence of the other features.

- In many practical applications, parameter estimation for naive *Bayes models uses the method of maximum likelihood; i*n other words, one can work with the naive Bayes model without accepting Bayesian probability or using any Bayesian methods.
- Despite their naive design and apparently oversimplified assumptions, *naive Bayes classifiers have worked quite well in many complex real-world situations.*
- An advantage of naive Bayes is that it only *requires a small amount of training data to estimate the parameters (means and variances of the variables) necessary for classification.* Because independent variables are assumed, only the variances of the variables for each class need to be determined and not the entire covariance matrix.
- The idea behind a Bayesian classifier is that, *if an agent knows the class, it can predict the values of the other features. If it does not know the class, Bayes' rule can be used to predict the class given (some of) the feature values*. In a Bayesian classifier, the learning agent builds a probabilistic model of the features and uses that model to predict the classification of a new example.
- A latent variable is a probabilistic variable that is not observed. *A Bayesian classifier is a probabilistic model where the classification is a latent variable that is probabilistically related to the observed variables.* Classifications then become inference in the probabilistic model.

Probabilistic model

Abstractly, the probability model for a classifier is a conditional model

$$p(C|F_1,\dots,F_n)$$

over a dependent class variable C with a small number of outcomes or *classes*, conditional on several feature variables F_1 through F_n. The problem is that if the number of features n is large or if a feature can take on a large number of values, then basing such a model on probability tables is infeasible. We therefore reformulate the model to make it more tractable.

Using Bayes' theorem, this can be written

$$p(C|F_1,\dots,F_n) = \frac{p(C)\ p(F_1,\dots,F_n|C)}{p(F_1,\dots,F_n)}.$$

In plain English, using Bayesian Probability terminology, the above equation can be written as

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

In practice, there is interest only in the numerator of that fraction, because the denominator does not depend on C and the values of the features F_i are given, so that the denominator is effectively constant. The numerator is equivalent to the joint probability model

$$p(C,F_1,\dots,F_n)$$

which can be rewritten as follows, using the chain rule for repeated applications of the definition of conditional probability:

$$\begin{aligned} p(C,F_1,\dots,F_n) &= p(C)\ p(F_1,\dots,F_n|C) \\ &= p(C)\ p(F_1|C)\ p(F_2,\dots,F_n|C,F_1) \\ &= p(C)\ p(F_1|C)\ p(F_2|C,F_1)\ p(F_3,\dots,F_n|C,F_1,F_2) \\ &= p(C)\ p(F_1|C)\ p(F_2|C,F_1)\ \dots p(F_n|C,F_1,F_2,F_3,\dots,F_{n-1}) \end{aligned}$$

Now the "naive" conditional independence assumptions come into play: assume that each feature F_i is conditionally independent of every other feature F_j for $j \neq i$, given the category C. This means that

$$p(F_i|C,F_j) = p(F_i|C),$$

$$p(F_i|C,F_j,F_k) = p(F_i|C),$$

$$p(F_i|C, F_j, F_k, F_l) = p(F_i|C),$$

and so on, for $i \neq j, k, l$. Thus, the joint model can be expressed as

$$\begin{aligned} p(C|F_1,\dots,F_n) &\propto p(C, F_1,\dots,F_n) \\ &\propto p(C)\ p(F_1|C)\ p(F_2|C)\ p(F_3|C)\ \cdots \\ &\propto p(C) \prod_{i=1}^{n} p(F_i|C)\,. \end{aligned}$$

This means that under the above independence assumptions, the conditional distribution over the class variable $C$is:

$$p(C|F_1,\dots,F_n) = \frac{1}{Z} p(C) \prod_{i=1}^{n} p(F_i|C)$$

where the evidence $Z = p(F_1, \dots, F_n)$is a scaling factor dependent only on $F_1, \dots, F_n$, that is, a constant if the values of the feature variables are known.

Constructing a classifier from the probability model

The naive Bayes classifier combines this model with a decision rule. One common rule is to pick the hypothesis that is most probable; this is known as the *maximum a posteriori* or *MAP* decision rule. The corresponding classifier, a Bayes classifier, is the function defined as follows:

$$\mathrm{classify}(f_1,\dots,f_n) = \underset{c}{\operatorname{argmax}}\ p(C=c) \prod_{i=1}^{n} p(F_i=f_i|C=c).$$

Parameter estimation and event models

The assumptions on distributions of features are called the *event model* of the Naive Bayes classifier. For discrete features like the ones encountered in document classification (include spam filtering), multinomial and Bernoulli distributions are popular.

Gaussian naive Bayes

When dealing with continuous data, a typical assumption is that the continuous values associated with each class are distributed according to a Gaussian distribution. For example, suppose the training data contain a continuous attribute, x. We first segment the data by the class, and then compute the mean and variance of x in each class. Let μ_c be the mean of the values in x associated with class *c*, and let σ_c^2 be the variance of the values in x associated with class *c*.

Then, the probability *density* of some value given a class, $p(x = v|c)$, can be computed by plugging $v$into the equation for a Normal distribution parameterized by $\mu_c$and σ_c^2. That is,

$$p(x = v|c) = \frac{1}{\sqrt{2\pi\sigma_c^2}}\, e^{-\frac{(v-\mu_c)^2}{2\sigma_c^2}}$$

Multinomial naive Bayes

With a multinomial event model, samples (feature vectors) represent the frequencies with which certain events have been generated by a multinomial $(p_1, \dots, p_n)$ where p_i is the probability that event *i* occurs (or *k* such multinomials in the multiclass case). The likelihood of observing a feature vector (histogram) *F* is given by

$$p(F|C) = \frac{(\sum_i F_i)!}{\prod_i F_i!} \prod_i p_i{}^{F_i}$$

The multinomial naive Bayes classifier becomes a linear classifier when expressed in log-space:

$$\begin{aligned} \log p(C|F) &\propto \log \left(p(C) \prod_{i=1}^{n} p(F_i|C) \right) \\ &= \log p(C) + \sum_{i=1}^{n} \log p(F_i|C) \\ &= b + \mathbf{w}_C^\top \mathbf{F} \end{aligned}$$

where $b = \log p(C)$and $w_{Ci} = \log p(F_i|C)$.

Bernoulli naive Bayes

In the multivariate Bernoulli event model, features are independent booleans (binary variables) describing inputs. Like the multinomial model, this model is popular for document classification tasks, where binary term occurrence features are used rather than term frequencies. If F_i is a boolean expressing the occurrence or absence of the *i*'th term from the vocabulary, then the likelihood of a document given a class *C* is given by

$$p(F_1, \dots, F_n|C) = \prod_{i=1}^{n} [F_i\, p(w_i|C) + (1 - F_i)(1 - p(w_i|C))]$$

where $p(w_i|C)$ is the probability of class *C* generating the term w_i. This event model is especially popular for classifying short texts. It has the benefit of explicitly modeling the absence of terms. Note that a naive Bayes classifier with a Bernoulli event model is not the same as a multinomial NB classifier with frequency counts truncated to one.

<u>5.6 Support Vector Machines (SVM)</u>

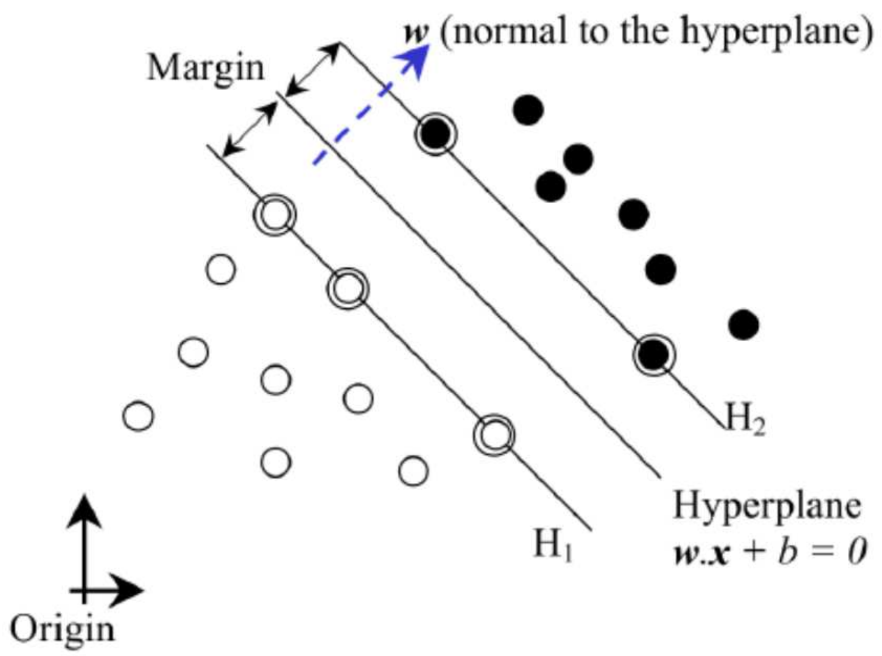
In machine learning, **support vector machines** (**SVMs**, also **support vector networks**) are supervised learning models with associated learning algorithms that analyze data and recognize patterns, used for classification and regression analysis.

- Given a set of training examples, *each marked as belonging to one of two categories, an SVM training algorithm builds a model that assigns new examples into one category or the other, making it a non-probabilistic binary linear classifier*.
- In addition to performing linear classification, *SVMs can efficiently perform a non-linear classification using what is called the kernel trick*, implicitly mapping their inputs into high-dimensional feature spaces.
- In contrast to classical methods of *statistics where in order to control performance one decreases the dimensionality of a feature space, the Support Vector Machine (SVM) dramatically increases dimensionality and relies on the so-called large margin factor.*
- *A support vector machine is primarily a two-class classifier. It is possible to solve multi class problems with support vectors by treating each single class as a separate problem. It aims to maximise the width of the margin between classes, that is, the empty area between the decision boundary and the nearest training patterns.*

Example

has been achieved a new pattern x can then be classified by testing which side of the hyper-plane

Let us consider the case where the class boundary is a hyper plane. Given a set of points $\{x_i\}$ in n-dimensional space with corresponding classes $\{y_i : y_i \in \{-1,1\}\}$ then the training algorithm attempts to place a hyper plane between points where y_i= 1 and points where y_i = -1. Once this has been achieved a new pattern x can then be classified by testing which side of the hyper-plane the point lies on.



In the simplest case, linear machines are trained on separable data. The training data is given by $(x_1, y_1), \cdots (x_N, y_N)$ where each x_i is a vector of real numbers and the corresponding y_i designates the class of x_i, $(1 or -1)$ Suppose the two classes can be separated by a hyperplane:

$$(w \cdot x) + b = 0$$

We say that this set is separated by the optimal hyperplane if it is separated without error and the distance between the closest points and the hyperplane is maximal. The training points which lie on one of the hyperplanes (H_1, H_2) and whose removal would change the solution found are called support vectors.

To describe the separating hyperplane let us suppose that all the training data satisfy the following constraints:

$$\begin{array}{ll} (w \cdot x_i) + b \geq +1 & \text{for} \quad y_i = +1 \\ (w \cdot x_i) + b \leq -1 & \text{for} \quad y_i = -1 \end{array}$$

which can be combined into one set of inequalities:

$$y_i(w.x_i + b) - 1 \geq 0 \text{ for } i = 1 \cdots N$$

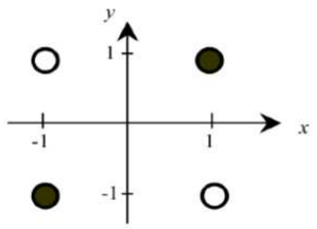
Now consider the points for which the equality in the previous equations holds. These points lie on one of two planes:

$$H_1 : (w \cdot x_i) + b = 1 \qquad \text{or} \qquad H_2 : (w \cdot x_i) + b = -1$$

with normal w and perpendicular distance from the origin respectively $|1-b|/|w|$ and $|-1-b|/|w|$. The shortest distance between H_1 and H_2 is the difference, $2/|w|$, and the distance between the separating hyperplane and the closest points is $1/|w|$.

Example: SVM for the XOR Problem

The exclusive OR is the simplest problem that cannot be solved using a single linear function operating on the features. We will lift the data from the 2 dimensional space to a six dimensional space using a quadratic polynomial kernel:



$$K(u,v) = (u \cdot v + 1)^2$$

For our two dimensional problem we can write the kernel as:

$$\begin{aligned} K(u,v) = & \ (u_1 v_1 + u_2 v_2 + 1)^2 \\ = & \ (u_1 v_1)^2 + (u_2 v_2)^2 + 2u_1 v_1 u_2 v_2 + 2u_1 v_1 + 2u_2 v_2 + 1 \end{aligned}$$

Mercer's condition holds for all polynomial kernels, so we are guarenteed a lifting function exists. We don't need to find one, but for example:

$$\phi(x_1, x_2) = (x_1^2, x_2^2, \sqrt{2}x_1 x_2, \sqrt{2}x_1, \sqrt{2}x_2, 1)$$

is one possibility. You can easily verify that it satisfies $K(u,v) = \phi(u) \cdot \phi(v)$. The new space has dimension 6, and, for the sake of example, we calculate the new six dimensional coordinates corresponding to our four original points:

Finding the separating hyperplane

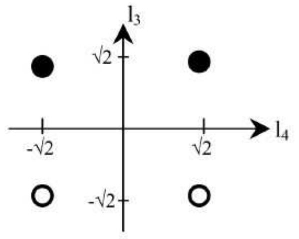
The distance separating the classes is $2/|w|$, the optimal hyperplane therefore is the one for which $|w|$ is minimal. Finding the hyperplane is formulated as the following optimisation problem:

$$\text{Minimise } |w|^2 \text{ subject to constraints } \forall(x_i, y_i) : \ y_i(w.x_i + b) - 1 \geq 0$$

We will not consider in depth the solution to this optimisation problem, but it is worth making a some observations about it. The solution uses Lagrange multipliers which have two advantages for this problem. First, the constraints will be replaced by constraints on the Lagrangian multipliers α_i, which will be much easier to handle. Secondly, the training data will only appear in the form of dot products between vectors. This is an important property that will allow us to generalise the method to the non-linear case.

	x_1	x_2	$l_1 = x_1^2$	$l_2 = x_2^2$	$l_3 = \sqrt{2}x_1x_2$	$l_4 = \sqrt{2}x_1$	$l_5 = \sqrt{2}x_2$	$l_6 = 1$
Class 1	1	1	1	1	$\sqrt{2}$	$\sqrt{2}$	$\sqrt{2}$	1
	-1	-1	1	1	$\sqrt{2}$	$-\sqrt{2}$	$-\sqrt{2}$	1
Class 2	1	-1	1	1	$-\sqrt{2}$	$\sqrt{2}$	$-\sqrt{2}$	1
	-1	1	1	1	$-\sqrt{2}$	$-\sqrt{2}$	$\sqrt{2}$	1

It can be seen that the points are separable on the l_3 axis. For this solution we have that in the high dimension space the optimal $w = [0, 0, 1, 0, 0, 0]$. Remember that in the actual algorithm we do not calculate (or even need to know) the lifting function ϕ.



Applications

SVMs can be used to solve various real world problems:

- SVMs are helpful in text and hypertext categorization as their application can significantly reduce the need for labeled training instances in both the standard inductive and transductive settings.
- Classification of images can also be performed using SVMs. Experimental results show that SVMs achieve significantly higher search accuracy than traditional query refinement schemes after just three to four rounds of relevance feedback.
- SVMs are also useful in medical science to classify proteins with up to 90% of the compounds classified correctly.
- Hand-written characters can be recognized using SVM

5.7 K - means clustering

- ***k*-means clustering** is a method of vector quantization, originally from signal processing, that is popular for cluster analysis in data mining.
- ***k-means clustering aims to partition n observations into k clusters in which each observation belongs to the cluster with the nearest mean, serving as a prototype of the cluster. This results in a partitioning of the data space into Voronoi cells***.
- The problem is computationally *difficult (NP-hard*); however, there are efficient heuristic algorithms that are commonly employed and converge quickly to a local optimum. These are usually similar to the *expectation-maximization algorithm for mixtures of Gaussi*an distributions via an iterative refinement approach employed by both algorithms.

- Additionally, they both *use cluster centers to model* the data; however, k-means clustering tends to find clusters of comparable spatial extent, while the expectation-maximization mechanism allows clusters to have different shapes.

Description

Given a set of observations ($\mathbf{x}_1$, $\mathbf{x}_2$, …, $\mathbf{x}_n$), where each observation is a d-dimensional real vector, k-means clustering aims to partition the n observations into k ($\leq n$) sets $\mathbf{S} = \{S_1, S_2, \ldots, S_k\}$ so as to minimize the within-cluster sum of squares (WCSS). In other words, its objective is to find:

$$\underset{\mathbf{S}}{\arg\min} \sum_{i=1}^{k} \sum_{\mathbf{x} \in S_i} \|\mathbf{x} - \boldsymbol{\mu}_i\|^2$$

where $\boldsymbol{\mu}_i$ is the mean of points in S_i.

Standard algorithm

The most common algorithm uses an iterative refinement technique. Due to its ubiquity it is often called the ***k*-means algorithm**; it is also referred to as **Lloyd's algorithm**, particularly in the computer science community.

Given an initial set of k means $m_1^{(1)},\ldots,m_k^{(1)}$, the algorithm proceeds by alternating between two steps:

Assignment step: *Assign each observation to the cluster whose mean yields the least within-cluster sum of squares (WCSS). Since the sum of squares is the squared Euclidean distance,* this is intuitively the "nearest" mean.

$$S_i^{(t)} = \{x_p : \|x_p - m_i^{(t)}\|^2 \leq \|x_p - m_j^{(t)}\|^2 \; \forall j, 1 \leq j \leq k\},$$

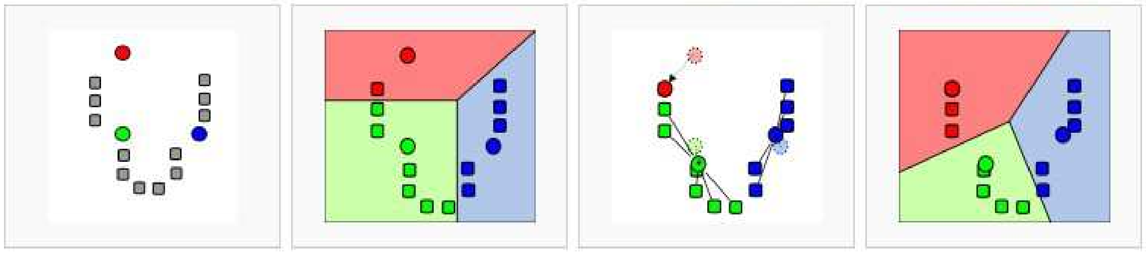
where each x_p is assigned to exactly one $S^{(t)}$, even if it could be assigned to two or more of them.

Update step: *Calculate the new means to be the centroids of the observations in the new clusters.*

$$m_i^{(t+1)} = \frac{1}{|S_i^{(t)}|} \sum_{x_j \in S_i^{(t)}} x_j$$

Since the arithmetic mean is a least-squares estimator, this also minimizes the within-cluster sum of squares (WCSS) objective.

Demonstration of the standard algorithm



1) *k* initial "means" (in this case *k*=3) are randomly generated within the data domain (shown in color).

2) *k* clusters are created by associating every observation with the nearest mean. The partitions here represent the Voronoi diagram generated by the means.

3) The centroid of each of the *k* clusters becomes the new mean.

4) Steps 2 and 3 are repeated until convergence has been reached.

- The algorithm has *converged when the assignments no longer change*. Since both steps optimize the WCSS objective, and there only *exists a finite number of such partitioning, the algorithm must converge to a (local) optimum*. There is no guarantee that the global optimum is found using this algorithm.