Term Paper Project (EE602)

Department of Electrical Engineering

Indian Institute of Technology, Kanpur

Statistical Pattern Recognition: A Review

Instructor:	**Prof. R. Hegde**
Submitted By:	**Ashish Kumar (Y5123)**
	Ashutosh Kumar (Y5128)
Date of Submission:	**17/04/2009**

1. Introduction

Pattern recognition is the study of how machines can observe the environment, learn to distinguish patterns of interest from their background, and make sound and reasonable decisions about the categories of the patterns. In spite of a lot of research in the area, design of a general purpose machine pattern recognizer remains an elusive goal. The best pattern recognizers in most instances are humans, yet we do not understand how humans recognize patterns. The work of Nobel Laureate Herbert Simon, whose central finding was that pattern recognition is critical in most human decision making tasks, suggest that the more relevant patterns are at our disposal, the better our decisions will be. This is hopeful news to proponents of artificial intelligence, since computers can surely be taught to recognize patterns. Our goal here is to introduce pattern recognition as the best possible way of utilizing available sensors, processors, and domain knowledge to make decisions automatically.

Automatic (machine) recognition, description, classification, and grouping of patterns are important problems in a variety of engineering and scientific disciplines such as biology, psychology, medicine, marketing, computer vision, artificial intelligence, and remote sensing. But what is a pattern? Watanabe defines a pattern "as opposite of a chaos; it is an entity, vaguely defined, that could be given a name." For example, a pattern could be a fingerprint image, a handwritten cursive word, a human face, or a speech signal. Given a pattern, its recognition/classification may consist of one of the following two tasks: 1) supervised classification (e.g., discriminant analysis) in which the input pattern is identified as a member of a predefined class, 2) unsupervised classification (e.g., clustering) in which the pattern is assigned to a hitherto unknown class. Note that the recognition problem here is being posed as a classification or categorization task, where the classes are either defined by the system designer (in supervised classification) or are learned based on the similarity of patterns (in unsupervised classification).

Interest in the area of pattern recognition has been renewed recently due to emerging applications which are not only challenging but also computationally more demanding. These applications include data mining (identifying a pattern, e.g., correlation, or an outlier in millions of multidimensional patterns), document classification (efficiently searching text documents), financial forecasting, organization and retrieval of multimedia databases, and biometrics (personal identification based on various physical attributes such as face and fingerprints).

The design of a pattern recognition system essentially involves the following three aspects: 1) data acquisition and preprocessing, 2) data representation, and 3) decision making. The problem domain dictates the choice of sensor(s), preprocessing technique, representation scheme, and the decision making model. Learning from a set of examples (training set) is an important and desired attribute of most pattern recognition systems. The four best known approaches for pattern recognition are: 1) template matching, 2) statistical classification, 3) syntactic or structural matching, and 4) neural networks. Here, our focus is mainly on statistical classification.

2. Statistical Pattern Recognition

Statistical pattern recognition has been used successfully to design a number of commercial recognition systems. In statistical pattern recognition, a pattern is represented by a set of d features, or attributes, viewed as a d-dimensional feature vector. Well-known concepts from statistical decision theory are utilized to establish decision boundaries between pattern classes. The recognition system is operated in two modes: training (learning) and classification (testing) (see Fig. 1). The role of the preprocessing module is to segment the pattern of interest from the background, remove noise, normalize the pattern, and any other operation which will contribute in defining a compact representation of the pattern. In the training mode, the feature extraction/selection module finds the appropriate features for representing the input patterns and the classifier is trained to partition the feature space. The feedback path allows a designer to optimize the preprocessing and feature extraction/selection strategies. In the classification mode, the trained classifier assigns the input pattern to one of the pattern classes under consideration based on the measured features.

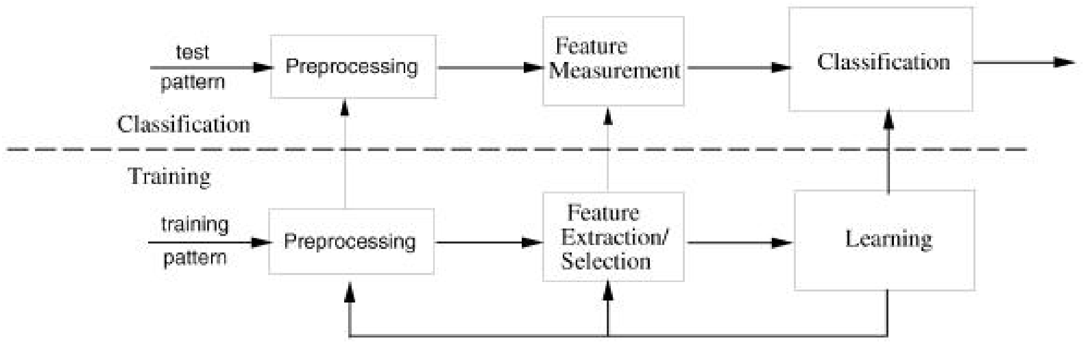

Fig. 1: Structure of statistical Pattern Recognition

2.1. Decision Making

The decision making process in statistical pattern recognition can be summarized as follows: A given pattern is to be assigned to one of c categories $w_1, w_2, \dots \dots \dots, w_c$ based on a vector of d feature values $\boldsymbol{x} = (x_1, x_2, \dots \dots \dots .. x_d)$. The features are assumed to have a probability density or mass (depending on whether the features are continuous or discrete) function conditioned on the pattern class. Thus, a pattern vector x belonging to class w_i is viewed as an observation drawn randomly from the class-conditional probability function $p(\boldsymbol{x}|w_i)$. A number of well-known decision rules, including the Bayes decision rule, the maximum likelihood rule (which can be viewed as a particular case of the Bayes rule), and the Neyman-Pearson rule are available to define the decision boundary.

2.1.1. Bayesian Decision Theory

The “optimal” Bayes decision rule for minimizing the risk (expected value of the loss function) can be stated as follows: Assign input pattern x to class w_i for which the conditional risk

$$R(w_i|\boldsymbol{x}) = \sum_{j=1}^{c} L(w_i, w_j).P(w_j|\boldsymbol{x})$$

Is minimum, where $L(w_i, w_j)$ is the loss incurred in deciding w_i when the true class is w_j and $P(w_j|\boldsymbol{x})$ is the posterior probability. In the case of the 0/1 loss function, as defined below, the conditional risk becomes the conditional probability of misclassification.

$$L(w_i, w_j) = \begin{cases} 0 \; i = j \\ 1 \; i \neq j \end{cases}$$

For this choice of loss function, the Bayes decision rule can be simplified as follows (also called the maximum a posteriori (MAP) rule): Assign input pattern x to class w_i if

$$P(w_i|\boldsymbol{x}) > P(w_j|\boldsymbol{x}) \quad for\ all\ j \neq i$$

Various strategies are utilized to design a classifier in statistical pattern recognition, depending on the kind of information available about the class-conditional densities. If all of the class-conditional densities are completely specified, then the optimal Bayes decision rule can be used to design a classifier. However, the class-conditional densities are usually not known in practice and must be learned from the available training patterns. If the form of the class-conditional densities is known (e.g., multivariate Gaussian), but some of the parameters of the densities (e.g., mean vectors and covariance matrices) are unknown, then we have a parametric decision problem. A common strategy for this kind of problem is to replace the unknown parameters in the density functions by their estimated values, resulting in the so-called Bayes plug-in classifier. The optimal Bayesian strategy in this situation requires additional information in the form of a prior distribution on the unknown parameters. If the form of the class-conditional densities is not known, then we operate in a nonparametric mode. In this case, we must either estimate the density function (e.g., Parzen window approach) or directly construct the decision boundary based on the training data (e.g., k-nearest neighbor rule). In fact, the multilayer perceptron can also be viewed as a supervised nonparametric method which constructs a decision boundary.

Other Classification techniques can be summarized as in the Fig.2

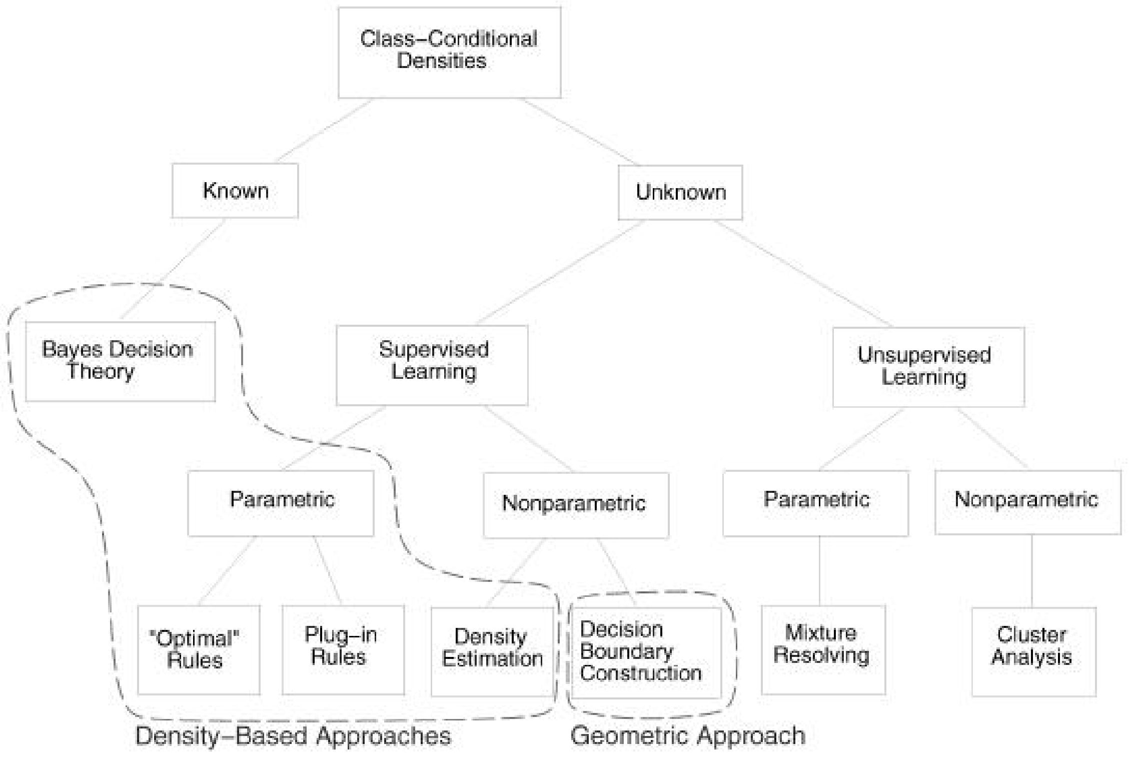

Fig. 2 Various Classification Approaches

3. Curse of Dimensionality and Peaking

The performance of a classifier depends on the interrelationship between sample sizes, number of features, and classifier complexity. A naive table-lookup technique (partitioning the feature space into cells and associating a class label with each cell) requires the number of training data points to be an exponential function of the feature dimension. This phenomenon is termed as "curse of dimensionality," which leads to the "peaking phenomenon" in classifier design. It is well-known that the probability of misclassification of a decision rule does not increase as the number of features increases, as long as the class-conditional densities are completely known (or, equivalently, the number of training samples is arbitrarily large and representative of the underlying densities). However, it has been often observed in practice that the added features may actually degrade the performance of a classifier if the number of training samples that are used to design the classifier is small relative to the number of features. This paradoxical behavior is referred to as the peaking phenomenon. A simple explanation for this phenomenon is as follows: The most commonly used parametric classifiers estimate the unknown parameters and plug them in for the true parameters in the class-conditional densities. For a fixed sample size, as the number of features is increased (with a corresponding increase in the number of unknown parameters), the reliability of the parameter estimates decreases. Consequently, the performance of the resulting plug-in classifiers, for a fixed sample size, may degrade with an increase in the number of features.

4. Dimensionality Reduction

There are two main reasons to keep the dimensionality of the pattern representation (i.e., the number of features) as small as possible: measurement cost and classification accuracy. A limited yet salient feature set simplifies both the pattern representation and the classifiers that are built on the selected representation. Consequently, the resulting classifier will be faster and will use less memory. Moreover, as stated earlier, a small number of features can alleviate the curse of dimensionality when the number of training samples is limited. On the other hand, a reduction in the number of features may lead to a loss in the discrimination power and thereby lower the accuracy of the resulting recognition system.

The main issue in dimensionality reduction is the choice of a criterion function. A commonly used criterion is the classification error of a feature subset. But the classification error itself cannot be reliably estimated when the ratio of sample size to the number of features is small. In addition to the choice of a criterion function, we also need to determine the appropriate dimensionality of the reduced feature space. The answer to this question is embedded in the notion of the intrinsic dimensionality of data. Intrinsic dimensionality essentially determines whether the given d-dimensional patterns can be described adequately in a subspace of dimensionality less than d. For example, d-dimensional patterns along a reasonably smooth curve have an intrinsic dimensionality of one, irrespective of the value of d. Note that the intrinsic dimensionality is not the same as the linear dimensionality which is a global property of the data involving the number of significant Eigen values of the covariance matrix of the data.

Now we will briefly discuss some of the methods for feature extraction and feature selection.

5. Feature Extraction

Feature extraction methods determine an appropriate subspace of dimensionality m (either in linear or a nonlinear way) in the original feature space of dimensionality d ($m \leq d$). Linear transforms, such as principal component analysis, factor analysis, linear discriminant analysis, and projection pursuit have been widely used in pattern recognition for feature extraction and dimensionality reduction. The best known linear feature extractor is the principal component analysis (PCA) or Karhunen-LoeÁve expansion, that computes the m largest eigenvectors of the $d\ x\ d$ covariance matrix of the n d-dimensional patterns. The linear transformation is defined as

$$Y = XH$$

Where, X is the given $n\ x\ d$ pattern matrix, Y is the derived $n\ x\ m$ pattern matrix, and H is the $d\ x\ m$ matrix of linear transformation whose columns are the eigenvectors. Since PCA uses the most expressive features (eigenvectors with the largest Eigen values), it effectively approximates the data by a linear subspace using the mean squared error criterion. Other methods, like projection pursuit and independent component analysis (ICA) are more appropriate for non-Gaussian distributions since they do not rely on the second-order property of the data. ICA has been successfully used for blind-source separation; extracting linear feature combinations that define independent sources. This demixing is possible if at most one of the sources has a Gaussian distribution. Whereas PCA is an unsupervised linear feature extraction method, discriminant analysis uses the category information associated with each pattern for (linearly) extracting the most discriminatory features. In discriminant analysis, interclass separation is emphasized by replacing the total covariance matrix in PCA by a general separability measure like the Fisher criterion, which results in finding the eigenvectors of $S_w^{-1}S_b$ (the product of the inverse of the within-class scatter matrix, S_w, and the between-class matrix S_b). There are various ways to define non linear feature extraction but we will not discuss them here.

6. Feature Selection

The problem of feature selection is defined as follows: given a set of d features, select a subset of size m that leads to the smallest classification error. There has been a resurgence of interest in applying feature selection methods due to the large number of features encountered in the following situations: 1) multi sensor fusion: features, computed from different sensor modalities, are concatenated to form a feature vector with a large number of components; 2) integration of multiple data models: sensor data can be modeled using different approaches, where the model parameters serve as features, and the parameters from different models can be pooled to yield a high-dimensional feature vector.

Let Y be the given set of features, with cardinality d and let m represent the desired number of features in the selected subset $X, X \subseteq Y$. Let the feature selection criterion function for the set X be represented by $J(X)$. Let us assume that a higher value of J indicates a better feature subset; a natural choice for the criterion function is $J = 1 - P_e$, where P_e denotes the classification error. The use of P_e in the criterion function makes feature selection procedures dependent on the specific classifier that is used and the sizes of the training and test sets. The most straightforward approach to the feature selection problem would require 1) examining all $\binom{d}{m}$ possible subsets of size m, and 2) selecting the subset with the largest value of $J(.)$. However, the number of possible subsets grows combinatorially, making this exhaustive search impractical for even moderate values of m and d. It is shown that no non-exhaustive sequential feature selection procedure can be guaranteed to produce the optimal subset. It is further shown that any ordering of the classification errors of each of the 2^d feature subsets is possible. Therefore, in order to guarantee the optimality of, say, a 12-dimensional feature subset out of 24 available features, approximately 2.7 million possible subsets must be evaluated. The only "optimal" (in terms of a class of monotonic criterion functions) feature election method which avoids the exhaustive search is based on the branch and bound algorithm. This procedure avoids an exhaustive search by using intermediate results for obtaining bounds monotonocity property of the criterion function $J(.)$; given two features subsets X1 and X2, if X1 < X2, then J(X1) < J(X2). In other words, the performance of a feature subset should improve whenever a feature is added to it. Most commonly used criterion functions do not satisfy this monotonicity property.

7. Classifiers

Once a feature selection or classification procedure finds a proper representation, a classifier can be designed using a number of possible approaches. In practice, the choice of a classifier is a difficult problem and it is often based on which classifier(s) happen to be available, or best known, to the user.

There are basically three approaches to design a classifier:

1. Template Matching
   This is based on the intuitive way of concept of similarity.
2. Probabilistic Approach
3. Geometric Approach

Here, we will discuss two classification algorithms one from probabilistic approach (Gaussian Mixture Models) and another from geometric approach (Support Vector Machines)

Gaussian Mixture Model (GMM)

Pattern classes can be modeled using GMM. The multivariate Gaussian distribution function for a d-dimensional vector x is written as

$$P(x) = \frac{e^{\frac{-1(x-u)^t \sum^{-1}(x-u)}{2}}}{(2\Pi)^{\frac{d}{2}}}$$

Where, μ is the d-component mean vector and ∑ is the d-by-d covariance matrix.

The number of mixtures in a pattern type is a function of the pattern class s, denoted as n(s).

Let Θ_1 , $\Theta_{2,\dots\dots}$,$\Theta_{n(s)}$ be the vectors corresponding to the mixtures M_1, M_2,……,$M_{n(s)}$, where Θ_i is the vector with components μ_i and $\sum_I$ of the mixture mi. Given the feature vector, x, the Bayes formula to determine the (posteriori) probability of x being in the ith mixture, M_i, is given as

$P(Q|x)=P(m_i)p(x|Q_i)$

An estimate of the probability of a mixture (prior) is calculated as:

$P(m_i)=n_m/N$

where n_{mi} the number of vectors in M_i, and N is the total number of vectors.

We use to determine the probability for each vector in the data set. The parameters (dimensions of the vector) are packet train length and packet train size. The following algorithm is used in the training phase.

1. Initialize the prior probability of each mixture with equal value such that their sum is 1.
2. Initialize the mean (vector) of each mixture by selecting a vector randomly, such that no two mixtures have the same mean.
3. Initialize the covariance matrix of each mixture to the d-by-d identity matrix.
4. Until the mean and variance of mixtures converge

{ For each vector in the given data set, classify it into mixture m_i if

$P(Q_{i|\mathbf{x}}) > P(Q_{j|\mathbf{x}})$ for all $j \neq i$

where Qi corresponds to mixture Mi. If there are two or more mixtures with maximum probabilities, one among them is chosen arbitrarily.

- Re-compute the probability of each mixture.
- Re-compute the mean vector of each mixture.
- Re-compute the covariance matrix of each mixture

The above algorithm basically computes the vector Qi and the probability corresponding to every mixture of a traffic class.

The testing phase uses the probabilities and the means and variances of mixtures of different traffic classes obtained from the training phase. In Bayesian clustering, the probability that the

i th vector, Xi belongs to a traffic class is found using:

$P(s|\mathbf{x}_i) = \sum P(Q_j|\mathbf{x}_i) \qquad 1 <= j <= n(s)$

where n(s) is the number of mixtures in the traffic class denoted by s.

The probability that the given set of vectors belong to a particular traffic class is determined by the joint probability of all the vectors

Support Vector Machines (SVM)

Support Vector Machines(SVM) is a supervised method of learning used for classification and regression problems. A Support Vector Machine (SVM) performs classification by constructing an N-dimensional hyper-plane that optimally separates the data into two categories .

A predictor variable is called an attribute, and a transformed attribute that is used to define the hyper-plane is called a feature. The task of choosing the most suitable representation is known as feature selection. A set of features that describes one case (i.e., a row of predictor values) is called a vector. So the goal of SVM modeling is to find the optimal hyper-plane that separates clusters of vector in such a way that cases with one category of the target variable are on one side of the plane and cases with the other category are on the other size of the plane. The vectors near the hyper-plane are the support vectors.

There may exist many such solutions that separate the classes exactly. If there are multiple solutions all of which classify the training set correctly then we should try to find out the find the one with the smallest generalized error .The support vector uses the concept of margin. The margin is the smallest distance between the decision boundary and any of the samples as shown in Figure 1b .

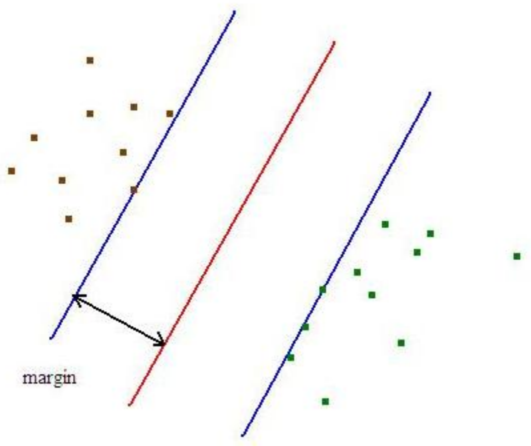

Figure 1b : Shows the concept of margin in two dimensional data

In support vector machines the decision boundary is chosen to be the one in which the margin is maximized.
Figure 1c shows graphically the classification of data containing three clusters using LIBSVM [16] ,a package a package kit for SVM classification.

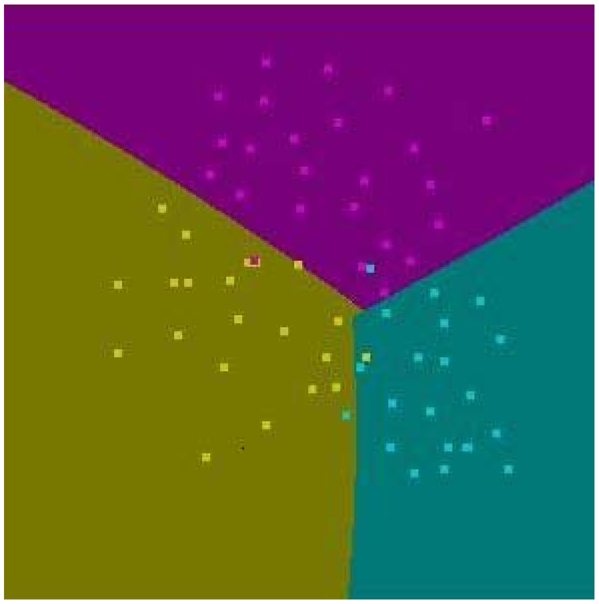

Figure 1c : Shows boundaries for two dimensional data with three clusters

8. References

[1] Anil K. Jain, Fellow, IEEE, Robert Duin, and Jianchang Mao, Senior member, IEEE “Statistical Pattern Recognition: A Review”

[2] H.M. Abbas and M.M. Fahmy, ªNeural Networks for Maximum Likelihood Clustering,º Signal Processing, vol. 36, no. 1, pp. 111- 126, 1994.

[3] H. Akaike, ªA New Look at Statistical Model Identification,º IEEE Trans. Automatic Control, vol. 19, pp. 716-723, 1974.