

Chapter 5

Effective Hand Gesture Classification Approaches

Hand gestures recognition goals can only be fulfilled when gesture isolation is coupled with an effective feature extraction followed by highly efficient classification. In the context of machine vision, feature extraction and classification can be jointly called pattern recognition in which, previous known patterns are matched with a query gesture.

In hand gesture recognition, separating any input gesture into a pre-assigned class suggest that the classification problem is a multiclass classification. For any effort in classification to be effective, the feature extraction should have generated adequate conditions such that the classes cluster far apart from each other. There are two prominent ways of solving the classification problem: linear and nonlinear. Linear classification approach involving more than two classes is known as a multi-class linear classification which would be discussed in detail in this chapter.

It is quite difficult to identify any classification problem as linear or non-linear without observing the feature data. As an example, moment invariant features usually leads to non-linear classification as the features would visibly not cluster as separable. If some gestures are removed, it is always feasible to find a linear classifier that separates gestures. When assessing the features' likelihood of belonging to different clusters in classification, it is imperative to use metrics to ascertain the affinity of data to clusters. Given that many features are multi-dimensional in hand gesture recognition, simple thresholding will not be effective. Fortunately, there is a large number of metrics that handle multi-dimensional data offering different types of distance metrics that will also be visited in this chapter. This will be followed by an in-depth view of both linear and nonlinear techniques in classification especially related to hand gesture recognition.

One important aspect in classification is to concisely represent extracted features. This is due to the fact that features extracted usually contain correlation which is not possible to foresee. There are many methodologies out there that decouple this redundant information and represent the feature data more elegantly. This can be achieved by decorrelating data using Eigen vectors and Karhunen Loeve Transform which are used in Support Vector Machines (SVM) or Principal Component Analysis (PCA).

In image processing, there is evidence of a long history in classification. Some of these classification approaches are related to classifying remote sensing data where images obtained by satellites or high-altitude aircrafts classifying data as vegetation, built-up areas or as waters. In hand gesture recognition, however, the gesture is fairly well-defined. The variability lies in size, the skin color, and lighting variation, movement of the hand and the other body movements. Therefore, even though the hand gesture is well-defined, the user may offer gestures which are quite similar to other gestures or some of the other variabilities will result in capturing a hand gesture looking similar to a different gesture. This undoubtedly led to complications and limits the number of gestures a system can recognize with an acceptable accuracy.

Since all classification methods make use of one or multiple distance metrics for declaring class affinity, in depth knowledge of the leading distance metrics will enhance the understanding of the new researchers. Next few sections will describe most of the known distance metrics which are commonly associated with hand gesture recognition.

5.1 Distance Metrics

Classification is the process of finding a data point of set of data points to another set of cluster representative data points. Since the affinity can be thought of as a distance in multi-dimensional space, measurement of these distances has a prominent role in any type of classification. There are many different notions of distance measure based on the context of the problem from biological cell growth to tornado prediction. Some of the prominent and widely used metrics are described next.

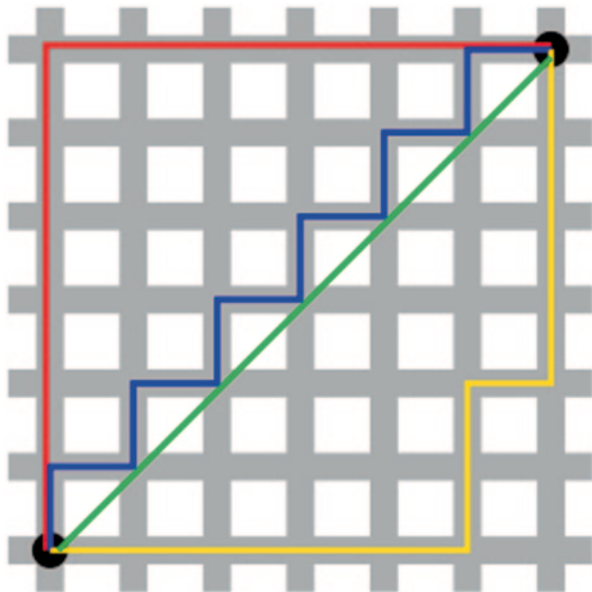
5.1.1 Euclidean Distance

Historically, Euclidean distance which was famously conceived by Euclidean provides the best direct measure between two points in multiple dimensions. The original distance was calculated using the Pythagorean triads [1, 2]. Consider two vectors with n-dimension; $V_1(a_1, a_2, a_3, a_4, \dots, a_n)$, $V_2(b_1, b_2, b_3, b_4, \dots, b_n)$. There Euclidean distance is measured as:

$$D_{Euclid} = \sqrt{(a_1 - b_1)^2 + (a_2 - b_2)^2 + (a_3 - b_3)^2 + (a_4 - b_4)^2 + \dots + (a_n - b_n)^2}$$

and is the simplest form of distance measures used by a vast number of applications.

Fig. 5.1 Two dimensional view of Manhattan distance. Green line shows Euclidean distance where as *Red, Blue* and *Yellow* all show different Manhattan distance of the same value



5.1.2 Manhattan Distance


This metric is also known as the *City Block Distance* assumes that in going from one vector to the other, it is only possible to travel directly along grid lines avoiding diagonal moves as illustrated in Fig. 5.1. Therefore the *Manhattan* distance is given by:

$$D_{\text{Manhattan}} = |a_1 - b_1| + |a_2 - b_2| + |a_3 - b_3| + |a_4 - b_4| + \dots + |a_n - b_n|$$

5.1.3 Chebyshev Distance

Concieved by Pafnuty Chebyshev, Chebyshev distance (or Tchebychev distance), Maximum metric is a metric defined on a vector space where the distance between two vectors is the greatest of their differences along any coordinate dimension [3, 4]. It is also known as chessboard distance, since in the game of chess, the minimum number of moves needed by a king to go from one square on a chessboard to another equals the Chebyshev distance between the centers of the squares. If the squares have side length one, as represented in 2-D spatial coordinates with axes aligned to the edges of the board [5].

Fig. 5.2 The Chebyshev distance between two spaces on a chess board gives the minimum number of moves a king requires to move between them. This is because a king can move diagonally, so that the jumps to cover the smaller distance parallel to a rank or column is effectively absorbed into the jumps covering the larger. Above are the Chebyshev distances of each square from the square f6

	a	b	c	d	e	f	g	h	
8	5	4	3	2	2	2	2	2	8
7	5	4	3	2	1	1	1	2	7
6	5	4	3	2	1		1	2	6
5	5	4	3	2	1	1	1	2	5
4	5	4	3	2	2	2	2	2	4
3	5	4	3	3	3	3	3	3	3
2	5	4	4	4	4	4	4	4	2
1	5	5	5	5	5	5	5	5	1
	a	b	c	d	e	f	g	h	

The metric is defined as:

$$D_{Chebyshev} = \max(|a_1 - b_1|, |a_2 - b_2|, |a_3 - b_3|, |a_4 - b_4|, \dots, |a_n - b_n|)$$

Figure 5.2 illustrates the *Chessboard Distance*, the metric assumes that moves can be made on the pixel grid as a ‘King’ making moves in chess, i.e. a diagonal move counts the same as a horizontal move.

5.1.4 Minkowski Distance

The *Minkowski distance* is a *superset* metric or a metric that can describe Euclidean, Manhattan and Chebyshev metrics as special cases. The definition is given by the following expression for a two vectors with dimension of n :

$$D_{Minkowski} = \left(|a_1 - b_1|^p + |a_2 - b_2|^p + |a_3 - b_3|^p + |a_4 - b_4|^p + \dots + |a^n - b^n|^p \right)^{1/p}$$

Minkowski distance reduces to Euclidean distance when $p=2$ and Manhattan distance when $p=1$. It also reduces to Chebyshev distance when p reaches infinity as a limiting case described by:

$$D_{Chebyshev} = \max_{i=1}^n |a_i - b_i| = \lim_{p \rightarrow \infty} \left(\sum_{i=1}^n |a_i - b_i|^p \right)^{1/p}$$

It is interesting to notice that the Manhattan and Chebyshev metrics can be computed faster than the Euclidean metric so that applications relying on speed can make use of Chebyshev when the accuracy is not critically important.

5.1.5 Mahalanobis Distance

The Mahalanobis distance is a descriptive statistic that provides a relative measure of a data point's distance (residual) from a common point. It is a unitless measure introduced by P. C. Mahalanobis in 1936 [6]. The Mahalanobis distance is used to identify and gauge similarity of an unknown sample set to a known one. It differs from Euclidean distance in that it takes into account the correlations of the data set and is scale-invariant. The distance measure is given by:

$$D_{Mahalanobis} = (\mathbf{a} - \mathbf{b})^T C^{-1} (\mathbf{a} - \mathbf{b}).$$

Where *Covariance Matrix* of \mathbf{a} and \mathbf{b} defined by

$$C = \sum_{i=1}^n \frac{(a_i - \bar{\mathbf{a}})(b_i - \bar{\mathbf{b}})}{n - 1}$$

The foundation of Mahalanobis measure can be understood by considering a classification problem. If given a point in n -dimensional space where its class affinity is in question, one would first estimate the average of the centre of mass of some given sample points whose class affinity is known. In determining the class affinity, the first logical step would be to the average or center of mass of the sample points. Intuitively, the closer the point in question is to this center of mass, the more likely it is to belong to the set.

However, it is important to know if the set is spread out over a large range or a small range. This would decide whether a given distance from the center is noteworthy or not. The simplistic approach is to estimate the standard deviation of the distances of the sample points from the center of mass. If the distance between the test point and the center of mass is less than one standard deviation, then it can be concluded that it is highly probable that the test point belongs to the set. The further away it is, the more likely that the test point should not be classified as belonging to the set. This intuitive approach can be made quantitative by defining the normalized distance between the test point and the set. This can be used in the normal distribution to derive the probability of the test point belonging to the set.

The drawback of the above approach was that it was assumed that the sample points are distributed about the center of mass in a spherical manner. Were the distribution to be decidedly non-spherical, for instance ellipsoidal, then the probability of the test point belonging to the set would depend not only on the distance from the center of mass, but also on the direction. In those directions where the ellipsoid has a short axis the test point must be closer, while in those where the axis is long the test point can be further away from the center.

Founding this on a mathematical basis, the ellipsoid that best represents the set's probability distribution can be estimated by building the covariance matrix of the samples. The Mahalanobis distance is simply the distance of the test point from the center of mass divided by the width of the ellipsoid in the direction of the test point.

In order to use the Mahalanobis distance to classify a test point as belonging to one of n classes, the covariance matrix has to be estimated for each class, usually based on samples known to belong to each class. This would be followed by computing the Mahalanobis distance from the test sample to each class. The minimum Mahalanobis distance would indicate the class affinity of the test sample.

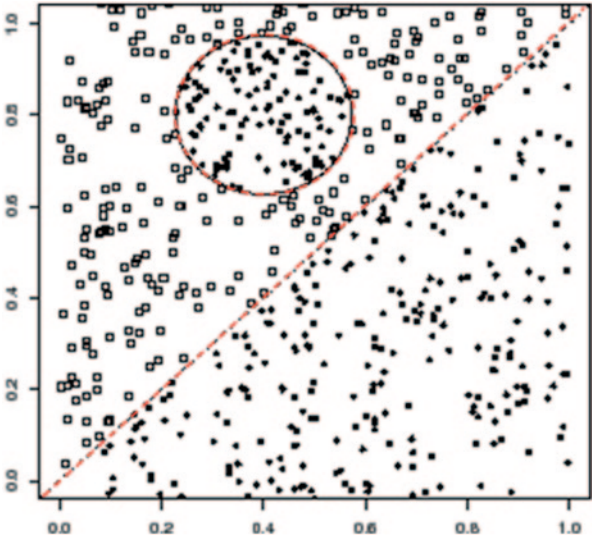
As described in Chap. 4, the most important step in hand gesture recognition is the ability to capture unique features representing a unique gesture yet with enough robustness such that other users may offer the gesture with some variability. The system should be able to resolve the difference among different gestures yet be able to capture the common traits of the unique gesture offered by the same user at different times and the gestures offered by different users with different skin colors under different lighting conditions. Armed with the knowledge of various metrics for assessing the effectiveness of classification approaches, the next sections of this chapter will present in-depth discussion on linear and nonlinear classifications algorithms for developing a better insight for effective classification in hand gestures recognition.

5.2 Linear vs Nonlinear Classification Approaches

Shape classification problem in computer vision is known to be a more difficult problem than any other classification problem as the features which are extracted from shapes sometimes can result in very high dimensions. Humans are capable of visualizing up to three dimensions at a time however; very high dimensions usually leave the human user without any insight into high dimensionality of the classification problem. As shown in the example of Fig. 5.3, the red straight broken line can separate two data types with some error. The data encircled with the red broken circle will be misclassified. However, if this error is acceptable, the system can make use of a simple linear classifier. The common approach to classification problem is to start with some linear approaches and classify the data. If such classification results in error, then it can be assumed that the problem needs a non-linear classification approach to lower misclassification.

There are other instances where noise in the data (features) simply complicates the classification problem. Considering a simple hypothetical scenario of two linearly separable classes with added noise; it might be that a particular training set is not linearly separable due to the noise, but that no nonlinear classifier can generalize better than a linear one (e.g. one could simulate data such that a linear shift was the true underlying difference between the classes). If a system that is trained to classify with linear classifier fails to separate one class from the rest, then the system needs

Fig. 5.3 Red straight line indicates a linear boundary that can separate two classes with some errors. If this linear classifier is used, data in the circle will be misclassified



to be non-linear in order to solve the problem. Alternatively, non-cross-validated measures that balance accuracy and complexity such as Bayesian model evidence could be used. When moving onto a non-linear solution, the improvement in accuracy should justify the use of a more complicated model that would also result in non-realtime classification especially in hand gesture recognition [7–14]. Figure 5.4 (left) indicates certain data distribution where linear classification whereas (right) indicates that a nonlinear approach is more suitable.

5.2.1 Linear Classifiers

Classification is the pivotal step in computer vision where a machine interprets the final outcome of any shape recognition. In applications such as face recognition,

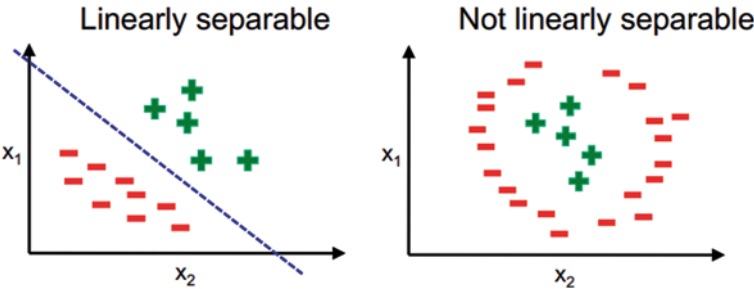


Fig. 5.4 Different data distribution requires different classification approaches [7]

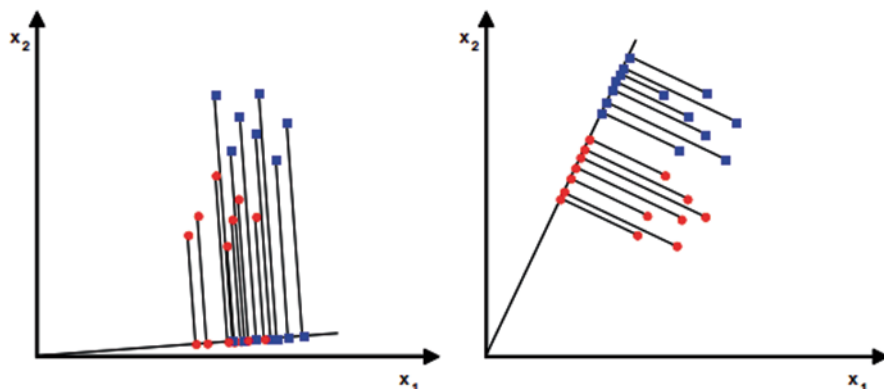


Fig. 5.5 LDA helps to classify data by maximizing the between class scatter and minimizing within class scatter when mapped to a different dimension

emotion detection or gesture recognition, the hurdle that limits success of any classification is the high dimensionality associated with large number of features needed to be extracted for improved classification. In order to handle data effectively and efficiently, its dimensionality needs to be reduced. Dimensionality reduction does not simply result in low dimensionality but leads to efficient, meaningful representation of reduced dimensionality. These lower dimensions now contain minimum number of features needed to represent the observed properties of the data. Dimension reduction techniques facilitate classification and compression of high-dimensional data. Principal Component Analysis (PCA) and Linear Discriminant Analysis (LDA) are very powerful techniques for dimension reduction and feature extraction. Many applications use these techniques for recognition purposes especially in face recognition where LDA plays a very important roll but its performance get effected if there are less number of observation as compare to the dimensionality of the given samples [15]. PCA is being used so that LDA performance will not get degraded due to small sample size problem. As many recognition systems use PCA (Principal components analysis) [16, 17, 18]. PCA is an effective method which can reduce the dimensionality of the data and can effectively extract the required information of the image. It provides data which has no redundancy as Gabor filter wavelet are not orthogonal wavelets. In case of image processing the complexity of grouping the images can be reduced. As among other dimension reduction methods, PCA is the fastest algorithm which will reduce the time complexity [19]. PCA also has good performance for a small data set. Linear discriminant analysis (LDA) and the related Fisher's linear discriminant are methods used in statistics, pattern recognition and machine learning 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 the final classification stage. LDA maps data from one dimension to another dimensional space wither the between class scatter is maximized while minimizing within class scatter as shown in Fig. 5.5.

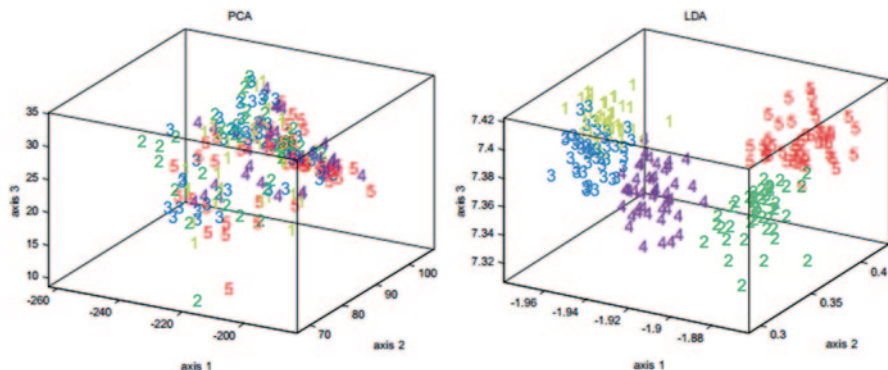


Fig. 5.6 From the 3D scatter plots it is clear that LDA outperforms PCA in terms of class discrimination, courtesy of [13]

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 [13]. 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 where a distinction between independent variables and dependent variables (also called criterion variables) must be made prior to classification. LDA is a parametric method based on unimodal Gaussian likelihoods. If LDA is used on distributions which are significantly non-Gaussian, the LDA projections may not preserve complex structure in the data needed for classification. Therefore, it is imperative that the user develops adequate knowledge about the feature distribution before any classification. This does not discourage a user from attempting to classify data using LDA involved approaches. In case the classification results are poor, non-Gaussian distributions may hold keys for its demise.

LDA's use can be better appreciated by looking at its performance in face recognition. In computerized face recognition, each face is represented by a large number of pixel values. Linear discriminant analysis is primarily used here to reduce the number of features to a more manageable number before classification. Each of the new dimensions is a linear combination of pixel values, which form a template. The linear combinations obtained using Fisher's linear discriminant are called Fisher faces, while those obtained using the related principal component analysis are called Eigenfaces. The ability for LDA to outperform PCA is shown in Fig. 5.6 for a multi-class problem.

If the number of classes is more than two, then a natural extension of Fisher Linear discriminant exists using multiple discriminant analysis [20]. As in two-class case, the projection is from high dimensional space to a low dimensional space and the transformation suggested still maximize the ratio of intra-class scatter to the inter-class scatter [8]. But unlike the two-class case, the maximization should be done among several competing classes.

5.2.1.1 Fisher's Discriminant for Multiple Classes

Linear Discriminant Analysis, or simply LDA, is a well-known classification technique that has been used successfully in many statistical pattern recognition problems as was stated earlier. 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 by transforming the data to a different space that is optimal for distinguishing between the classes. In case of multiple classes, FDA can be extended to find a subspace which appears to contain all of the class variability. Assuming that each of C classes has mean μ_i and the same covariance Σ and the number of features is greater than the number of classes. Then the between class variability can be defined by the sample covariance of the class means such that

$$\Sigma_b = \frac{1}{C} \sum_{i=1}^C (\mu_i - \mu)(\mu_i - \mu)^T$$

Where μ is the means of the class means and *class mean* is defined as follows:

$$\mu_i = \frac{1}{M_i} \sum_{m=1}^{M_i} \vec{X}_m^i.$$

Here $\vec{Y} = \vec{D}^T \vec{X}$ signifies that \vec{X} has been projected onto direction \vec{D} . The class separation or class scatter in a direction \vec{D} will be given by the following expression:

$$s_k = \frac{\vec{D}^T \Sigma_b \vec{D}}{\vec{D}^T \Sigma \vec{D}} \sum_{m=1}^{M_i} (\vec{X}_m^k - \mu)(\vec{X}_{im}^k - \mu)^T.$$

This implies that \vec{D} is an Eigenvector of $\Sigma^{-1} \Sigma_b$ where its separation will be equal to the corresponding Eigenvalue. If $\Sigma^{-1} \Sigma_b$ is diagonalizable, the variability between features will be contained in the subspace spanned by the Eigenvectors corresponding to the C -1 largest Eigenvalues. These eigenvectors are primarily used in feature reduction, as in PCA. The eigenvectors corresponding to the smaller Eigenvalues will tend to be very sensitive to the exact choice of training data, and it is often necessary to use regularization.

In practice, the class means and covariances are not known but can be estimated from the training set. Either the maximum likelihood estimate or the maximum a posteriori estimate may be used in place of the exact value in the above equations. Although the estimates of the covariance may be considered optimal in some sense, this does not mean that the resulting discriminant obtained by substituting these values is optimal in any sense, even if the assumption of normally distributed classes is correct.

Another complication in applying LDA and Fisher's discriminant to real data occurs when the number of observations of each sample does not exceed the number

Fig. 5.7 A simple two class problem with clear class separation

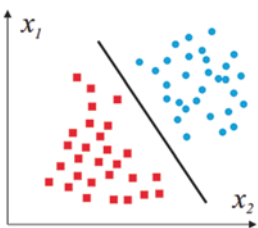
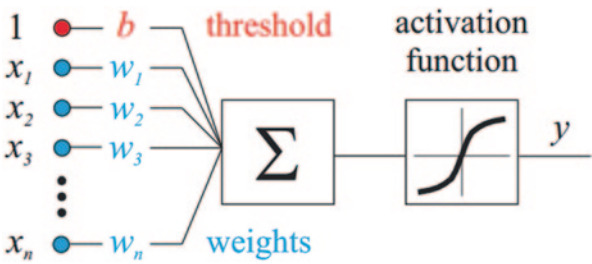


Fig. 5.8 Perceptron input/output relationship with weights, thresholds and activation functions



of samples [13]. In this case, the covariance estimates do not have full rank, and so cannot be inverted. There are a number of ways to deal with this. One is to use a pseudo inverse instead of the usual matrix inverse in the above formulae. However, better numeric stability may be achieved by first projecting the problem onto the subspace spanned by [21].

5.2.1.2 Multiclass Perceptron Classifier

In machine learning, the perceptron is an algorithm for supervised classification of an input into one of several possible non-binary outputs. It is known as a type of linear classifier as the classification algorithm makes its predictions based on a linear predictor function combining a set of weights with the feature vector describing a given input using the delta rule. The learning algorithm for perceptrons is an online algorithm, in that it processes elements in the training set one at a time. The perceptron algorithm was conceived in 1957 Frank Rosenblatt [22].

Even though, it was conceived as a two-class linear classifier, it has been extended to multiclass classification. Perceptron can be considered as the simplest example of Neural Network which would be discussed under nonlinear classification. The perceptron defines a hyperplane that split the representation space into two parts when handling two classes. Figure 5.7 shows a typical two-class problem with clear separation between classes and Fig. 5.8 shows the perceptron input-output relationship including all its intermediary functions.

Figure 5.9 depicts the online algorithm for two class-case where misclassifications are corrected continuously resulting in better classification results and Fig. 5.9 shows a commonly used activation function in perceptron classifier (Fig. 5.10).

1. $w_1 = 0$.
2. A wrongly classified observation x_j is sought, i.e., $\langle w_t, x_j \rangle < 0$, $j \in \{1, \dots, L\}$.
3. If there is no misclassified observation then the algorithm terminates otherwise
 $w_{t+1} = w_t + x_j$.
4. Goto 2.

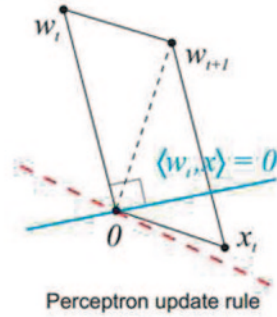
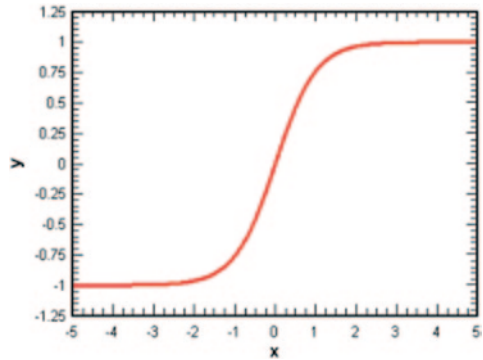


Fig. 5.9 Rosenblatt perceptron algorithm for two class classification [22]

Fig. 5.10 A typical activation function: hyperbolic tangent function given by the expression

$$f(x) = \frac{\sinh x}{\cosh x} = \frac{e^x - e^{-x}}{e^x + e^{-x}} = \frac{e^{2x} - 1}{e^{2x} + 1}$$



In multiclass perceptron algorithm, the classification starts with zero weights. Then each training vector is used one at a time and output is evaluated as follows:

$$\begin{aligned} y &= \arg \max_y w_y \cdot f(x) \\ &= \arg \max_y \sum_i w_{y,i} \cdot f_i(x). \end{aligned}$$

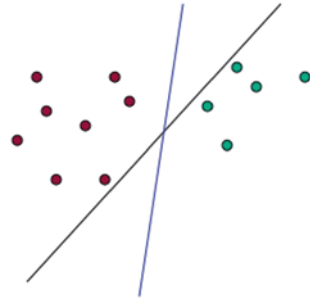
If the output generates the correct classification, no adaptations of weights are carried out. However, if the classification is wrong, the score of the wrong weights are adjusted as follows:

$$w_y = w_y - f(x).$$

The weights of the correct answer are increased such that:

$$w_{y*} = w_{y*} + f(x).$$

Fig. 5.11 When two classes can be separated using multiple decision boundaries, SVM determines the separation boundary in the most efficient way



The perceptron learning algorithm does not terminate if the learning set is not linearly separable. If the vectors are not linearly separable learning will never reach a point where all vectors are classified properly. The most famous example of the perceptron's inability to solve problems with linearly nonseparable vectors is the Boolean exclusive-OR problem. The solution spaces of decision boundaries for all binary functions and learning behaviors are studied in the reference [23].

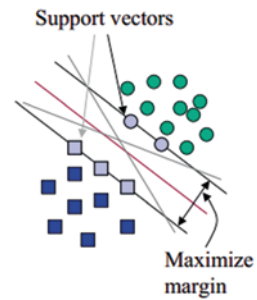
In the context of artificial neural networks, a perceptron is an artificial neuron using the Heaviside step function as the activation function. The perceptron algorithm is also termed the single-layer perceptron, to distinguish it from a multilayer perceptron, which is a misnomer for a more complicated neural network. As a linear classifier, the single-layer perceptron is the simplest feedforward neural network.

5.2.1.3 Linear Support Vector Machine (SVM)

Linear support vector machines are supervised learning models with associated learning algorithms that analyze data and recognize patterns, used for classification and regression analysis. The basic SVM takes a set of input data and predicts, for each given input, which of two possible classes forms the output, making it a non-probabilistic binary linear classifier [24]. Given a set of training examples as shown in Fig. 5.11, each marked as belonging to one of two classes; an SVM training algorithm builds a model that assigns new examples into one category or the other. An SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible. New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall on. SVM are based on statistical learning theory and have the aim of determining the location of decision boundaries that produce the optimal separation of classes [25].

As shown in Fig. 5.11 and 5.12, SVMs maximize the margin around the separating hyperplane. The decision function is fully specified by a subset of training samples, the support vectors. SVM has been found attractive by researchers as it is known as a quadratic programming problem with application found in hand gesture recognition [26–32], and text classification methods [33–37]. These applications

Fig. 5.12 Maximizing the separation between classes using support vectors [33]



are mainly feasible due to SVM's natural ability to handle large dimensional data with high accuracy and high flexibility with sparse representation of the solution using support vectors. They are also capable of making future predictions quickly without the need for probability estimates of data. Hence, they are also known as hard classifiers separated by Hyperplanes.

If the problem being investigated is separable and linear in two dimensions, a straight line can be found as discussed above. However, for higher dimensions, hyperplanes need to be found by linear programming such as perceptrons. They can be expressed as straight lines given by $ax + by = c$. Support vectors are the elements of the training set that would change the position of the dividing hyper plane if removed. Support vectors are also the critical elements of the training set and the problem of finding the optimal hyper plane is an optimization problem and can be solved by optimization techniques (use Lagrange multipliers to get into a form that can be solved analytically).

In the case of a two-class pattern recognition problem in which the classes are linearly separable, the SVM selects from among the infinite number of linear decision boundaries the one that minimizes the generalization error. Thus, the selected decision boundary will be one that leaves the greatest margin between the two classes, where margin is defined as the sum of the distances to the hyperplane from the closest points of the two classes [38–54]. This problem of maximizing the margin can be solved using standard Quadratic Programming (QP) optimization techniques. The data points that are closest to the hyperplane are used to measure the margin; hence these data points are termed 'support vectors'. Consequently, the number of support vectors is small [25].

If the two classes are not linearly separable, the SVM tries to find the hyperplane that maximizes the margin while, at the same time, minimizing a quantity proportional to the number of misclassification errors. The trade-off between margin and misclassification error is controlled by a user-defined constant [38]. SVM can also be extended to handle non-linear decision surfaces.

SVM were initially designed for binary (two-class) problems. When dealing with multiple classes, an appropriate multi-class method is needed. Vapnik in 1995 suggested comparing one class with the others taken together [25]. This strategy generates n classifiers, where n is the number of classes. The final output is the class that corresponds to the SVM with the largest margin, as defined above. For multi-

class problems one has to determine n -hyperplanes. Thus, this method requires the solution of n QP optimization problems, each of which separates one class from the remaining classes. This strategy can be described as ‘one against the rest’.

A second approach is to combine several classifiers (‘one against one’) to perform pair-wise comparisons between all n classes [55]. Thus, all possible two-class classifiers are evaluated from the training set of n classes, each classifier being trained on only two out of n classes, giving a total of $n(n-1)/2$ classifiers. Applying each classifier to the test data vectors gives one vote to the winning class. The data is assigned the label of the class with most votes. The results of a recent analysis of multi-class strategies are provided by Hsu and Lin [56].

SVM for Multiclass Classification Multiclass SVM aims to assign labels to instances by using support vector machines, where the labels are drawn from a finite set of several elements. The dominant approach for doing so is to reduce the single multiclass problem into multiple binary classification problems. Originally, SVMs were developed for binary classification. However, applications of binary classification are very limited especially in remote sensing land cover classification where most of the classification problems involve more than two classes. A number of methods to generate multiclass SVMs from binary SVMs have been proposed by researchers and is still a continuing research topic.

Instead of creating many binary classifiers to determine the class labels, this method attempts to directly solve a multiclass problem [48, 57, 58]. This is achieved by modifying the binary class objective function and adding a constraint to it for every class. The modified objective function allows simultaneous computation of multiclass classification and is given by [59] as shown next:

$$\min_{w, b, \xi} \left[\frac{1}{2} \sum_{i=1}^M \|w\|^2 + C \sum_{i=1}^k \sum_{r \neq y_i} \xi_i^r \right]$$

Subject to the constraints,

$$w_{y_i} \cdot x_i + b_{y_i} \geq w_r \cdot x_j + b_r + 2 - \xi_i^r \text{ and } \xi_i^r \geq 0 \text{ for } i = 1, \dots, k.$$

Where $y_i \in \{1, \dots, M\}$ are the multiclass labels of the data vectors and $r \in \{1, \dots, M\} / y_i$ are multiclass labelling excluding y_i [25].

Lee et al. and Schölkopf and Smola showed that the results from this method and the one-against-the-rest are similar [57–59]. However, in this approach, the optimization algorithm has to consider all the support vectors at the same time. Therefore, it may be able to handle massive data sets but the memory requirement and thus, the computational time requirements may be very high. To summarize, it may be said that the choice of a multiclass method depends on the problem in hand. A user should consider the accuracy requirement, the computational time, the resources available and the nature of the problem. For example, the multiclass objective function approach may not be suitable for a problem that contains a large number of training samples and classes due to the requirement of large memory and extremely long computational time.

Fig. 5.13 Nonlinear classifiers have nonlinear and possibly discontinuous decision boundaries [77]

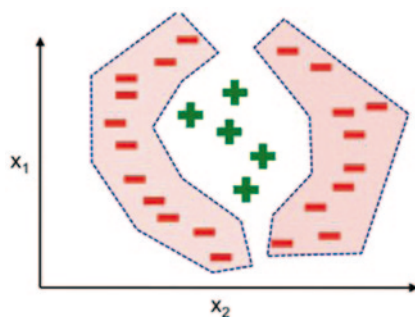
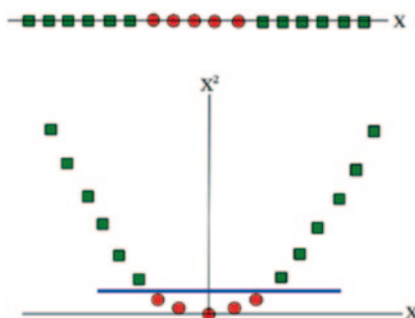


Fig. 5.14 Projecting data that is not linearly separable into a higher dimensional space can make it linearly separable [60]



5.2.2 NonLinear Classifiers

When class clusters cannot be separated effectively by a straight line, the problem becomes a nonlinear classification problem. This is illustrated in Fig. 5.13.

In some nonlinear classification approaches such as SVM (nonlinear SVM), nonlinear feature space is mapped to a linear space before classification. This is popularly known as the ‘Kernel Trick’. Figure 5.14 shows that the ‘red’ dots can be separated from ‘green’ squares easily when the data is mapped into a different dimension using a single threshold. There are other techniques such as Artificial Neural Networks (ANN) where no such transformation is required. As would be discussed in details in this chapter, ANN has been shown to be very effective in classifying known objects or hand gestures when effective features are extracted and adequately trained with sample data. Hidden Markov Models (HMM) have been increasingly being used in hand gesture recognition for its ability to describe postures and gestures as incremental steps which is not possible with many other classification approaches. K Nearest Neighbor method has also been extremely popular as a general nonlinear classification approach when ample amount of sample data or feature data are available. However, this is also plagued by its tendency to result in long processing time when large amount of sample data is available.

Fig. 5.15 Nonlinear Classification problem that can be attempted by nonlinear SVM [61]

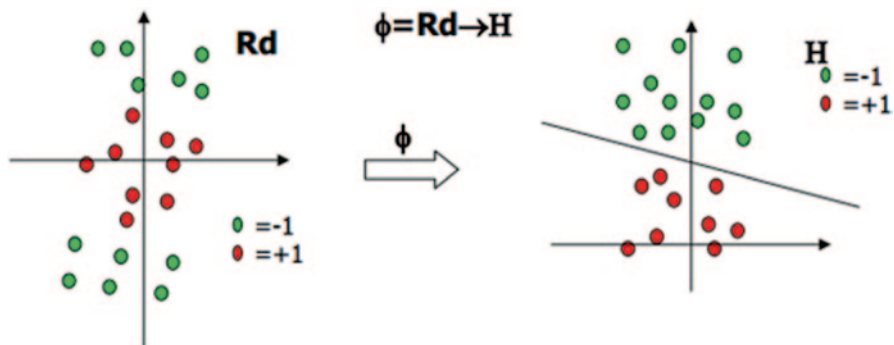
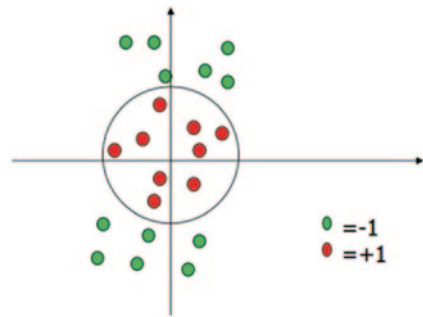


Fig. 5.16 Data can be mapped to a higher dimension for easy separation using SVM [59]

5.2.2.1 Non Linear SVM

The approach in nonlinear SVM is to map nonseparable data onto higher dimensions so that they are linearly separable as shown in Fig. 5.15. As depicted here, the red data cannot be effectively separated from the blue data in dimension x . When this data mapped to a higher dimension such that $x \mapsto \{x^2, x\}$, a linear separator can be found now to classify the data as shown on the bottom of the Fig 5.16. As would be obvious, such mapping could be extremely costly if the mapping for instance results in a quadratic space $x \mapsto \{x_1^2 x_2^2, \dots, x_D^2, x_1 x_2 x_3 \dots\}$. This leads to additional features and to inefficiency. The problem can be avoided by using Kernels approach which is extensively discussed in [62].

In many practical cases, it is not quite obvious as to what transform would result in a linear separable problem. SVM is a supervised learning method which is also a linear classifier that maximizes the distance between the decision lines. The main advantage of SVM is that it can use kernels for non-linear data transformation to cluster data into more separable ones in a new feature space as shown in Fig. 5.17. The main principle behind using SVM is to divide the given data into two distinct categories and then to get hyper-plane to separate the given classes.

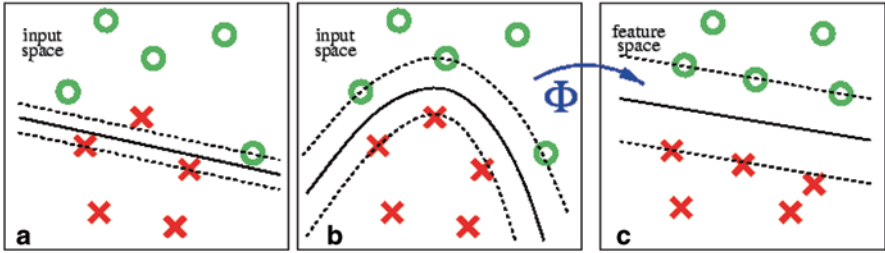


Fig. 5.17 Three different views on the same two class separation problem. **a** A linear separation of the input points is not possible without errors. Even allowing misclassification of one data point results in a small margin. **b** A better separation is provided by a non-linear surface in the input space. **c** This non-linear surface corresponds to a linear surface in a feature space. Data points are mapped from input space to feature space by the function Φ induced by the kernel function k [59]

Mathematically the well-known ‘kernel trick’ which brings a nonlinear problem to a linear one with ease can be describe as follows. Kernel-methods first preprocess the data by a certain non-linear mapping Φ and then apply the same linear algorithm as before but in the image space of Φ . (cf. Fig. 5.17 for an illustration). More formally the following mapping $\Phi: \mathbb{R}^N \rightarrow \mathcal{E}$ is performed.

$$x \mapsto \Phi(x)$$

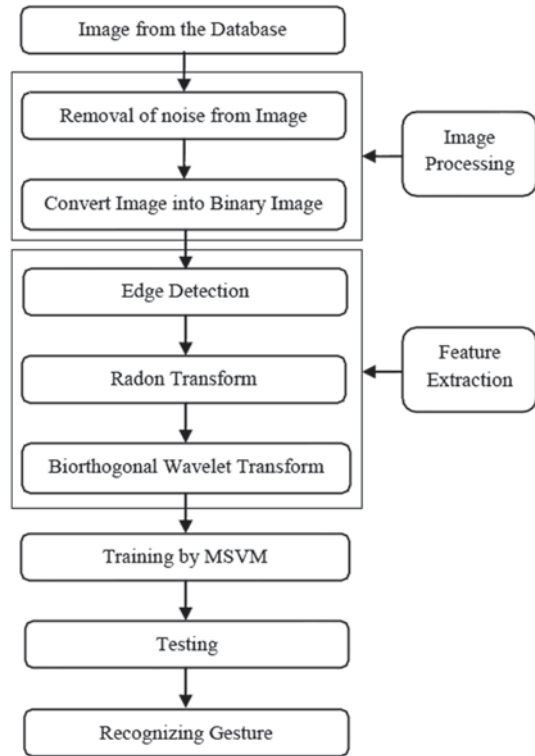
to the data $x_1, \dots, x_M \in \mathfrak{X}$ and consider our algorithms in \mathcal{E} instead of \mathfrak{X} . The sample is preprocessed as

$$\{(\Phi(x_1), y_1), \dots, (\Phi(x_M), y_M)\} \subseteq (\mathcal{E} \times \mathcal{Y})^M.$$

In certain applications, there would be sufficient knowledge about problem in hand such that an appropriate Φ can be designed [63, 64]. The mapping should be simple enough so that \mathcal{E} will not be overly high dimensional leading to a feasible mapping. Similar notions are attempted in (single hidden layer) neural networks [65], radial basis networks [66] or Boosting algorithms [67], where the input data are mapped to some representation given by the hidden layer, the RBF bumps or the hypotheses space, respectively [68]. One of the key advantages of kernel-methods is that a suitably chosen Φ results in an algorithm that has powerful nonlinearities but is still very intuitive and retains most of the favorable properties of its linear input space version.

Besides being useful tools for the computation of dot-products in high- or infinite-dimensional spaces, kernels possess some additional properties that make them an interesting choice in algorithms. It was shown [69] that using a particular positive definite kernel correspond to an implicit choice of a regularization operator. For translation-invariant kernels, the regularization properties can be expressed conveniently in Fourier space in terms of the frequencies [70, 71]. For example, Gaussian kernels correspond to a general smoothness assumption in all k^{th} order derivatives [70]. Alternatively, using the property of correspondence, kernels matching a certain prior about the frequency content of the data can be constructed so as

Fig. 5.18 SVM based hand gesture recognition, courtesy of Rahman et al. [83]



to reflect the known prior problem knowledge. Another particularly useful feature of kernel functions is that they are simply not restricted to kernels that operate on vectorial data, (e.g. $\mathcal{X} = \mathbb{R}^N$). In principle, it is also possible to define positive kernels for e.g. strings or graphs, i.e. making it possible to embed discrete objects into a metric space and apply metric-based algorithms (e.g. [63, 72, 73]). Furthermore, many algorithms can be formulated using so called *conditionally positive definite kernels* [70, 74] which are a superclass of the positive definite kernels. They can be interpreted as generalized nonlinear dissimilarity measures as opposed mere scalar products and are also applicable in kernel PCA. More information on nonlinear SVM and Kernel methods can be found in the following [75–82].

A hand gesture recognition techniques using SVM is proposed by Rahman and Afrin [83] in which they use features extracted from the skin segmented hand postures. The hand postures are Radon transformed and Biorthogonal Transform coefficients were used for multiclass SVM classification. In this usage of SVM, they find the optimal separating hyper plane such that error for unseen patterns is minimized. They used classification for 10 gestures denoting the letters from the alphabet, A, B, C, D, G, H, I, L, V, and Y. They had moderate success for their approach which can be summarized using the Fig. 5.18.

Chen et al. [84] reported a 3 camera angle image capture system that used 3 SVM classifiers for each angle that voted and fused the classification results. Their

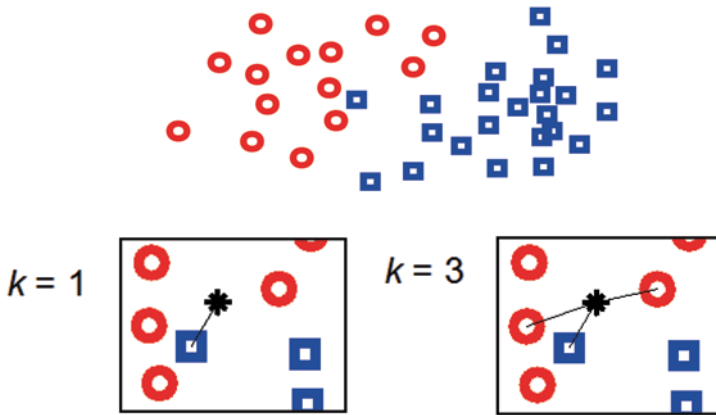
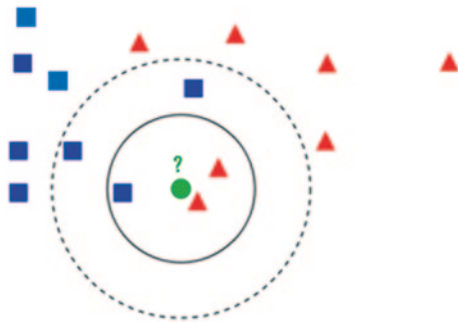


Fig. 5.19 KNN Algorithm. Two classes shown in *red circles* and *blue squares* on the *top* diagram. For a given query point q (shown in $*$), assign the class of the nearest neighbor. Compute the k nearest neighbors and assign the class by majority vote [86]

system was very sophisticated yet lacked proper feature extraction. They used a grayscale histogram equalized images and used a binary classification in the training phase with radial basis functions set as kernel functions. The reported performance accuracy was limited to be around 80% with large misclassification in some gesture classes.

5.2.2.2 Nearest Neighbor Classification

K-nearest neighbors (KNN) is a classification (or regression) algorithm that in order to determine the classification of a point, group affiliations of the nearest neighbors are considered. It is a non-parametric method for classifying objects based on closest training examples in the feature space. It is also known as a supervised classification technique as the membership of the other class members (points) is known. KNN is a type of instance-based learning, or lazy learning where the function is only approximated locally and all computation is deferred until classification. Many early work in pattern recognition relied on k-nearest neighbor algorithm as it is one of the simplest classification algorithms. In this classification, an object is classified by a majority vote of its neighbors, with the object being assigned to the class most common amongst its k nearest neighbors (k is a positive integer, typically small). If $k=1$, then the object is simply assigned to the class of that single nearest neighbor as shown in Fig. 5.19 [85]. Many early work on hand gesture recognition relied on KNN as it was highly effective inductive inference method for noisy training data and complex target functions. The target function or data points for a whole space may be described as a combination of less complex local approximations. The learning process is a simple and straight forward even though the classification is time consuming. Large number of data points could result in extremely high dimensions known as curse of dimensionality.

Fig. 5.20 KNN example

It can be useful to weight the contributions of the neighbors, so that the nearer neighbors contribute more to the average than the more distant ones. Usually, each neighbor is given a weight of $1/d$, where d is the distance to the neighbor. This scheme is a generalization of linear interpolation [86]. The neighbors are taken from a set of objects for which the correct classification or membership is known. This can be thought of as the training set for the algorithm, though no explicit training step is required. The k -nearest neighbor algorithm is sensitive to the local structure of the data [86]. Nearest neighbor rules in effect implicitly compute the decision boundary. It is also possible to compute the decision boundary explicitly, and to do so efficiently, so that the computational complexity is a function of the boundary complexity [87].

Algorithm The KNN algorithm can be better understood by following how the classification is carried out in Fig. 5.20. The test object shown in green circle should be classified either to the first class of dark blue squares, light blue squares or to the third class of red triangles. If $k=3$ (solid line circle) it is assigned to the third class because there are 2 triangles and only 1 dark blue square inside the inner circle. If $k=5$ (dashed line circle) it is assigned to the first class (3 dark blue squares vs. 2 triangles inside the outer circle).

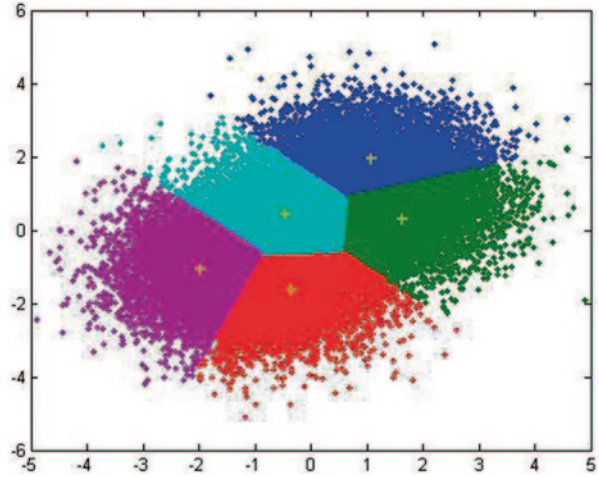
The training examples are vectors in a multidimensional feature space, each with a class label. The training phase of the algorithm consists only of storing the feature vectors and class labels of the training samples.

In the classification phase, k is a user-defined constant, and an unlabeled vector (a query or test point) is classified by assigning the label which is most frequent among the k training samples nearest to that query point. A commonly used distance metric for continuous variables is Euclidean distance which can be calculated using the following expression of Euclidean distance minimization:

$$\arg \min_s \sum_{i=1}^k \sum_{\mathbf{x}_j \neq \mathbf{x}_i} \|\mathbf{x}_j - \mathbf{x}_i\|.$$

For discrete variables, such as for text classification, another metric can be used, such as the overlap metric (or Hamming distance). Often, the classification accu-

Fig. 5.21 K-means nearest neighbor representation, courtesy of [88]



racy of KNN can be improved significantly if the distance metric is learned with specialized algorithms such as Large Margin Nearest Neighbor or Neighborhood components analysis.

One of the major drawbacks in KNN is its inability to handle uneven class distribution of data. If one class tends to have more data than another class, the “majority voting” results in poor or misclassification. Samples of a more frequent class tend to dominate the prediction of the new sample, because they tend to be common among the k nearest neighbors due to their large number [87]. One way to overcome this problem is to weight the classification, taking into account the distance from the test point to each of its k nearest neighbors as was mentioned before. The class of each of the k nearest points is multiplied by a weight proportional to the inverse of the distance from that point to the test point as was mentioned before. K-means nearest neighbor can handle some of these drawbacks.

5.2.2.3 K-Means Nearest Neighbor Classification

K-means is a clustering algorithm that tries to partition a set of points into K clusters such that each point belongs to the cluster with the nearest mean, serving as a prototype of the cluster as shown in Fig. 5.21. It is an unsupervised classification technique unlike the KNN as the points have no external classification.

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

$$\arg \min_S \sum_{i=1}^k \sum_{\mathbf{x}_j \in S_i} \|\mathbf{x}_j - \mu_i\|$$

Where μ_i is the mean of points in S_i .

Since the clusters are not labeled, this technique is not commonly used in hand gesture recognition. However, KNN problem can be extended to incorporate cluster mean instead of the k neighbors which would not require the distance to all the points in a classification field. Yet, it is up to individual researcher to report the success of such an approach.

Usage of k-Nearest Neighbor in Hand Gesture Recognition There have been few attempts to use KNN as a classification means for hand gesture recognition. Some of these attempts used KNN in combination of probabilistic approaches. The main trend for using KNN is its simplicity and the intuitive sense over other techniques. However, when the feature vector had multiple dimensions and the clustering in each domain was not distinctive, the researchers resorted to other methods. However, data or feature decorrelation techniques such as Vector Quantization can be used to regenerate clusters in higher dimensional space so that KNN can be more successful as a pattern classification technique.

Ziaie et al. reported a classification scheme based on naïve Bayes classifier based on KNN with distance weighting for static hand gesture recognition [89, 90]. They reported a performance of 93% accuracy for limited hand postures which were used in a human-robot dialog system. One of the major drawbacks of the KNN technique is the system is required to calculate the vector distance of the test gesture to all other neighbors in the system in order to select the k nearest neighbors. When a system has far more training data than another system, this results in unacceptable amount of computing time. This has resulted in discouraging many researchers from using KNN in pattern classification.

Vafadar and Berhad [91] reported a hand gesture recognition system based on spatio-temporal volume analysis technique in which hand contour extraction over time represented gesture paths for dynamic gesture recognition. In their approach, they used three types of classification methods: KNN, learning Vector Quantization Neural Network and back propagation neural networks. They reported that they had better recognition rate 96.6–99.6% with a k value of 2 (two nearest neighbor). Koller et al. described a hand gesture recognition technique based on depth information from a Time of Flight camera [92]. The hand size information was extracted and projected on to x-y plane. The x-y data and the depth information were used in a KNN classification approach with 94% accuracy for 12 gestures from 34 persons and a classification time of 30 ms.

5.2.2.4 Multi Layer Neural Networks

Artificial Neural Network or simply Neural Network as they are commonly known, models the behavior of neurons found in all animal life. These models effectively represent some of the fundamental attributes of neurons such as memory and outputs due one or many inputs exceeding pre-assigned thresholds. Scientists have used this notion to mimic the fundamental behavior of neurons using building blocks

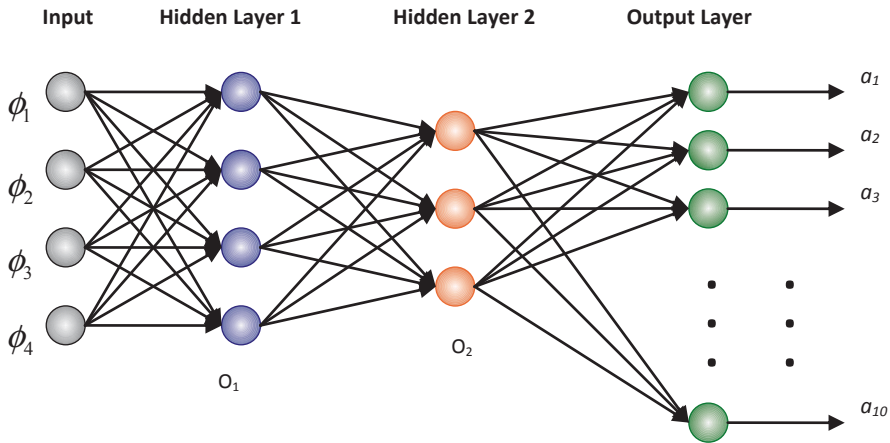


Fig. 5.22 Structure of the neural network classifier (conception perspective)

of mathematical functions with highly promising results. Today, there are myriad of scientific evidence to suggest that most non-linear classification problems have been attempted with Neural Networks. What has attracted the most interest in neural networks is the ability to memorize experience which is not available with many other classification approaches.

The artificial neural network refers to layers of inter connections between multiple inputs and multiple outputs. As shown in Fig. 5.22, many or one input may result in an output that would be used for many classification problems. The network nodes have the ability to remember their experience as a fundamental attribute of neurons due the weight each node is associated with. There are connections between input and output and some architectures have feedback and feed-forward from different layers. Each node executes a metric on incoming input to determine whether the input level exceeds a threshold to pass the communication of the signal to the next level. As it happens with human beings, for instance, some individuals may develop higher threshold for pain in which case their reaction to a level of pain will be different to a person experiencing the same level of pain.

It would be interesting to learn the historical developments of neural networks and their use in engineering problems. The first step toward artificial neural networks came in 1943 when Warren McCulloch, a neurophysiologist, and a young mathematician, Walter Pitts, wrote a paper on how neurons might work. They modeled a simple neural network with electrical circuits. Reinforcing this concept of neurons and how they work was a book written by Donald Hebb and the *The Organization of Behavior* was written in 1949 which pointed out that neural pathways are strengthened each time that they are used [93]. As computers advanced into their infancy of the 1950s, it became possible to begin to model the rudiments of these theories concerning human thought. Nathaniel Rochester from the IBM research laboratories led the first effort to simulate a neural network. That first attempt failed however, their later attempts succeeded.

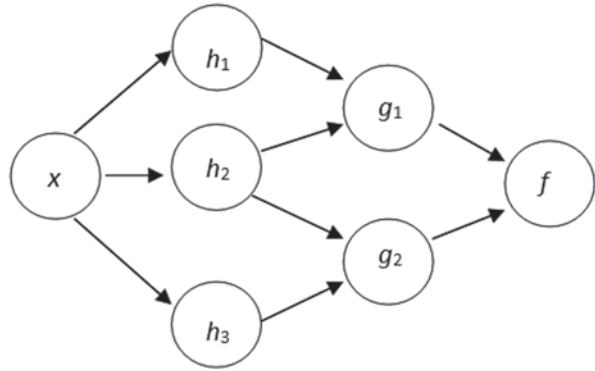
In 1956 the Dartmouth Summer Research Project on Artificial Intelligence provided a boost to both artificial intelligence and neural networks. One of the outcomes of this process was to stimulate research in both the intelligent side, AI, as it is known throughout the industry, and in the much lower level neural processing part of the brain [93]. In the years following the Dartmouth Project, John von Neumann suggested imitating simple neuron functions by using telegraph relays or vacuum tubes. Similarly, Frank Rosenblatt, a neuro-biologist of Cornell, began work on the Perceptron. He was intrigued with the operation of the eye of a fly as local level processing takes place in an eye of a fly which helps the fly to flee threatened. The Perceptron, which resulted from this research, was built in hardware and is the oldest neural network still in use today. A single-layer perceptron was found to be useful in classifying a continuous-valued set of inputs into one of two classes. The perceptron computes a weighted sum of the inputs, subtracts a threshold, and passes one of two possible values out as the result. Unfortunately, the perceptron was limited and was proven as such during the “disillusioned years” in Marvin Minsky and Seymour Papert’s 1969 book *Perceptrons* [93].

In 1959, Bernard Widrow and Marcian Hoff of Stanford developed models they called ADALINE and MADALINE. These models were named for their use of Multiple ADaptive LINear Elements. MADALINE was the first neural network to be applied to a real world problem. It was an adaptive filter which eliminates echoes on phone lines which is still found commercial use. Unfortunately, these earlier successes caused unsubstantiated expectations on neural networks, particularly in light of the limitation in the electronics available then. These fears, combined with unfulfilled, outrageous claims, caused respected voices to critique the neural network research which resulted in funding cuts to neural network based research.

This period of stunted growth lasted through 1981. In 1982 several events caused a renewed interest in neural networks especially when John Hopfield of University of California Technology presented a paper to the national Academy of Sciences. Hopfield’s approach was not to simply model brains but to create useful devices. With clarity and mathematical analysis, he showed how such networks could work and what they could do [93].

Pattern recognition is a powerful technique for harnessing information present in data for recognition purposes. Neural networks learn to recognize the patterns which exist in data sets. Neural network based system is developed through learning rather than programming. Programming needs a profound understanding of mechanisms when a system is developed through modeling. However, many pattern recognition problems are overly difficult to model as the pattern recognition problem can be thought of as in its infancy. Neural networks present the opportunity to classify data without in-depth knowledge of the pattern classification problem or accurate system modeling. Neural networks learn patterns that exist in data in a way which is still a mystery to the science and are also flexible in a changing environment. Rule based systems or programmed systems are limited to specific problems for which they were designed and when conditions change, they are no longer applicable. One of the disadvantages of neural networks is that learning time or network training typically requires a long time when more training data is available. They are also adept at building informative models where more conventional approaches fail.

Fig. 5.23 Input, hidden layers and output relationships of a neural network



Even though their ability to model complex interactions is a mystery, neural networks can easily model data which is too difficult to model with traditional approaches such as inferential statistics or programming logic. Performance of neural networks typically outperforms classical statistical modeling. They also have the ability to build models that are more reflective of the structure of the data in significantly less time.

An ANN is typically defined by three types of parameters:

1. The interconnection pattern between different layers of neurons
2. The learning process for updating the weights of the interconnections
3. The activation function that converts a neuron's weighted input to its output activation.

Figure 5.23 shows a diagram neural network which can be used to describe the mathematical relationship of input and output. Neuron's network function or the output $f(x)$ can be mathematically described using the following expression:

$$f(x) = K\left(\sum_i w_i g_i(x)\right)$$

Here, w_i 's are the weights of synapses g_i 's are the output function of previous layer and K is known as the activation function. There are few activation functions or transfer functions commonly use such as hyperbolic tangent functions that was described in the 5.2.1.2.

Learning Paradigms There are three major learning paradigms in neural network learning corresponding to a particular abstract learning task. These are supervised learning, unsupervised learning and reinforcement learning.

Supervised Learning It can be stated that the neural networks enjoy the popularity that it enjoys today due to the ability to learn under supervision. In supervised learning, the system has ample pre-classified data for with corresponding inputs. The system uses the input and generates output and compare with the expected output. If the system output differs from the expected output, then the error is fed back to the system to adjust the weights until the outputs totally agree with the expected output

which is known as training. This feeding error backwards through the network is known as *back-propagation*. Both the Multi-Layer Perceptron and the Radial Basis Function are supervised learning techniques. The Multi-Layer Perceptron uses the back-propagation while the Radial Basis Function is a feed-forward approach which trains on a single pass of the data.

Some systems may need many training cycles to adequately train the system so that system output matches exactly with the expected outputs. Different weight adjustment mechanisms are employed to optimize the weights so that the weights converge to their optimum values quickly [94–98].

Unsupervised Learning Unsupervised learning is used when the output is not expected to classify data but to group ‘similar looking’ entities together. The primary unsupervised technique is the Kohonen network which is used in cluster analysis. The advantage of the neural network for this type of analysis is that it requires no initial assumptions about what constitutes a group or how many groups there are. The system starts with a *clean slate* or *no prior knowledge* and is not biased about which factors should be most important [99–103]. Tasks that fall within the paradigm of unsupervised learning are in general estimation problems; the applications include clustering, the estimation of statistical distributions, compression and filtering.

Reinforcement Learning Reinforcement learning is learning by interacting with an environment where input is not explicitly given as in supervised or unsupervised learning. The input is generated by the system by observation. The system learns from the consequences of its actions, rather than from being explicitly taught and it selects its actions on basis of its past experiences (exploitation) and also by new choices (exploration), which is essentially trial and error learning. The aim is to discover a *policy* for selecting actions that minimizes some measure of a long-term cost when correct decisions are rewarded and incorrect ones are penalized [104–115].

Advantages of Neural Computing Neural networks are increasingly applied in many types of classification problems in divergent fields such as crop classification, predicting weather patterns to cell abnormalities in mammograms. The prominent reason for such applications is that to use neural networks for classification, no knowledge of underlying reasons are required as mentioned in previous sections. For instance, if wind speed, humidity, angle of the sun rays are determined to be prominent factors affecting the temperature of the air, then a neural network can be easily setup to predict air temperature using recorded data for training. The mathematical relationship among the above four factors and how they affect the temperature is not required to be established to predict the temperature. If the system is able to predict the temperature fairly accurately (if no other factors are affecting the temperature significantly), then temperature can be predicted by simply measuring other data without ever establishing their relationship. This example clearly illustrates how useful neural networks are and the reasons for their wide use in classification in diverse fields. It also highlights the potential knowledge vacuum in using neural network. The key limitation of using neural networks is its inability to explain the model and their relationship in a useful way.

Learning Algorithms Training a neural network model refers to adjusting weights to memorize the experience using training inputs to predict the training outputs. These weights are modified using different models that minimize a cost function to optimize the weights quickly. This approach usually uses gradient descent type algorithm. Expectation-maximization, non-parametric methods and particle swarm optimization [116], Evolutionary methods [117], gene expression programming [118], simulated annealing [119] are some commonly used methods for training neural networks.

As was mentioned in the previous section, neural networks are increasingly found in classification approaches due to their ability to be used as an arbitrary function approximation mechanism that ‘learns’ from observed data. However, a thorough understanding of the underlying theory is required to develop a robust neural network that is not overly complex and optimize its weights or training rapidly. The utility of artificial neural network models lies in the fact that they can be used to infer a function from observations. This is particularly useful in applications where the complexity of the data or task makes the design of such a function by hand impractical. Some of their usage is highlighted below:

- Function approximation, or regression analysis, including time series prediction, fitness approximation and modeling.
- Classification, including pattern and sequence recognition, novelty detection and sequential decision making.
- Data processing, including filtering, clustering, blind source separation and compression.
- Robotics, including directing manipulators, prosthesis.
- Control, including Computer numerical control

Application areas of neural networks include system identification and control (vehicle control, process control, natural resources management), quantum chemistry [120], game-playing and decision making (backgammon, chess, poker), pattern recognition (radar systems, face identification, object recognition), sequence recognition (gesture, speech, handwritten text recognition), medical diagnosis, financial applications (automated trading systems), data mining (or knowledge discovery in databases), visualization and e-mail spam filtering.

Artificial neural networks have also been used to diagnose several cancers. A NN-based hybrid lung cancer detection system named HLND improves the accuracy of diagnosis and the speed of lung cancer radiology [121]. These networks have also been used to diagnose prostate cancer. The diagnoses can be used to make specific models taken from a large group of patients compared to information of one given patient. The models do not depend on assumptions about correlations of different variables. Colorectal cancer has also been predicted using the neural networks and are known to predict the outcome for a patient with colorectal cancer with a higher accuracy than many clinical methods [122].

Some of the important criteria that should be exercised in developing a robust neural network classification system are:

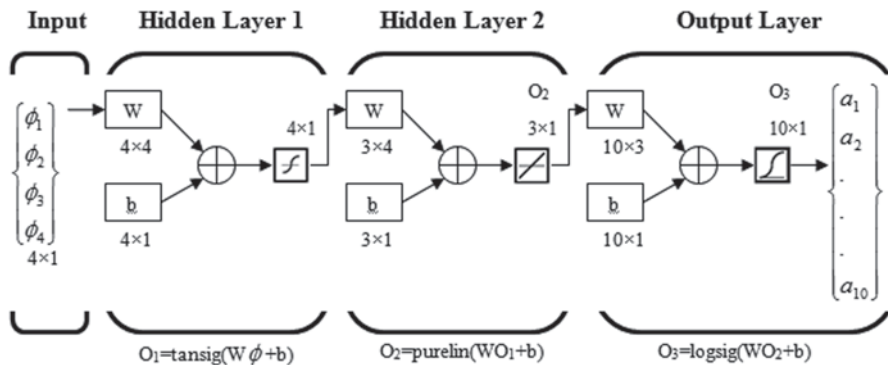


Fig. 5.24 Structure of the neural network classifier (implementation perspective)

1. Selecting an appropriate model: The application and the data (number of input parameters and output parameters) will determine this. Selecting complex models will take extremely long time to train and may never be optimized.
2. Learning algorithm: There are numerous trade-offs between learning algorithms. Almost any algorithm will work well with the *correct* hyper parameters for training on a particular fixed data set. However selecting and tuning an algorithm for training on unseen data requires a significant amount of experimentation.
3. System robustness: If the model, cost function and learning algorithm are selected appropriately the resulting neural network can be extremely robust.

Neural Network Classification for Hand Gesture Recognition In 2005, Premaratne et al. proposed a neural network based hand gesture recognition approach where invariant moments of binarized gesture images were used with extremely high accuracy for limited set of gestures [123]. The proposed neural network is a *backpropagation* network as shown in Fig. 5.24. In this particular network, the input vectors (the sample set of hand gestures) and the target vectors (the corresponding commands set) were used to train the network until it can approximate a function between the input and the output. There were a number of neuron layers between the input and the output. Each layer contains a number of nodes whose properties were characterized by a weight matrix W , a bias vector b and a transfer function f . Some popular transfer functions or activation functions used for testing were Log-Sigmoid transfer function, Tan-Sigmoid transfer function and linear transfer function [94].

In this design, there were only three layers due to the limited number of hand gestures to be classified. More complex network could possibly be designed and implemented, but it was not practical and necessary for this particular project. The first layer, i.e. the hidden layer 1, contained four nodes and it used the *tansig* transfer function. The second layer, i.e. the hidden layer 2, contained three nodes which used the *purelin* transfer function. This implied that the network could have had more hidden layers, but it was not necessary and practical for this particular project. The

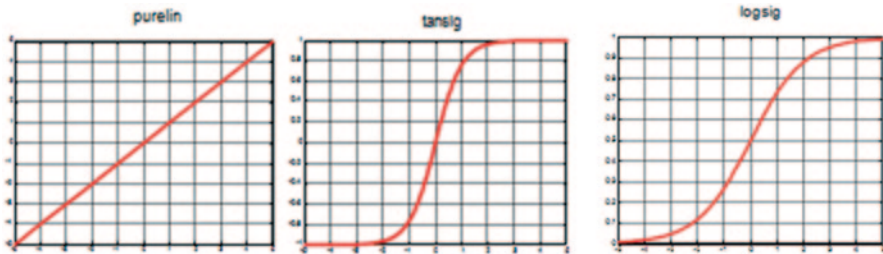


Fig. 5.25 Neural network activation (transfer) functions

last layer, i.e. the output layer, consisted of ten nodes and it used the *logsig* transfer function. Each of the above activation functions encompassed different properties and is illustrated in Fig. 5.25.

Purelin activation function is given by $f(x) = x$, and the tansig is given by $f(x) = \frac{2}{1 + e^{-2x}} - 1$ whereas logsig is expressed as $f(x) = \frac{1}{1 + e^{-x}}$.

The input was the set of moment invariant values derived from the sample set of hand gestures. The output was the target set of commands corresponding to each gesture. Some recent work on hand gesture recognition using neural networks can be found here [124–127].

5.2.2.5 HMM for Hand Gesture Classification

Hidden Markov Model (HMM) is a collection of finite states connected by transitions [128]. Each state is characterized by two sets of probabilities: a transition probability and either a discrete output probability distribution or continuous output probability density function which given the state, defines the condition probability of emitting each output symbol from a finite alphabet or a continuous random vector [129].

The use of Hidden Markov Model for hand gesture recognition stems from HMM's ability to successfully modeling speech recognition. The similarities between speech and gesture suggest that techniques effective for one problem may be effective for the other as well [129]. Hand gestures, like spoken language, vary according to the person, location or background, time, gender, age and social factors. There is common ground between speech and gestures which are known to have syntactic rules [128]. Since gesture is an expressive motion, it is natural to describe such a motion through sequential model [130].

HMMs can be used for object classification when given several models, it is possible to determine the model which will produce a given sequence of observations with the highest probability [131]. Thus, if for each class there is a model with the states, transitions and probabilities set appropriately, the Viterbi algorithm can be used to calculate the model that most probably resulted in the sequence of observations [132]. It is easy to see how this can be extended to a weak temporal

classification system where a single model is constructed for each class, and given an unlabelled instance; the probability that that sequence of output symbols was generated by each HMM is calculated. The model with the highest probability is identified as the predicted class [131].

However, it is not a trivial task to select the correct model and appropriate states and transitions. Knowledge in the problem domain and experience would play a significant role in succeeding with the application. Trial and error methods would identify the number of states and transitions when developing an efficient system.

The concept behind the use of HMM for gesture recognition is to use multi-dimensional HMM representing the parameters obtained from the training data using the chosen model. The trained models represent the most likely human performance and are used to evaluate the new incoming gestures. In using a HMM to classify incoming dynamic hand gestures, the first important step would be to develop a set of hand gestures which are distinct in terms of feature extraction. The HMM will use any type of parameters of features in classifying however, the success of the approach depends on the best choice of parameters or features. Therefore it is essential that the selection of the parameters of features are unique with properties such as rotation, scale, translation and illumination invariance as was outlined in the previous chapter on Feature Extraction. Such feature extraction step naturally to be preceded by preprocessing as already discussed in a previous chapter for the best classification results. Then these highly efficient features can be used to describe each gesture in terms of a multi-dimensional HMM. In other words, each gesture would have its own HMM with N distinct hidden states and r dimensional M distinct observable symbols. An HMM is characterized by a transition matrix A and r discrete output distribution matrices. Figure 5.26 shows a hand gesture recognition systems based on HMM.

Training the HMMs through training data is paramount to adjust the parameters in such a way that they can maximize the likelihood of known gestures for the given training data. The Baum-Welch [133] algorithm is commonly used to iteratively re-estimate model parameters to achieve the local maximum. Once the training is completed, any incoming dynamic gesture can be classified. The Forward-Backward algorithm or the Viterbi algorithm [132] can be used to classify both static and dynamic hand gestures.

Yang et al. [134] describes a dynamic gesture recognition method using state based spotting algorithm to split continuous gestures. Features using in the system include hand position, velocity, size and shape. They developed a data aligning algorithm to align feature vector sequences for training. Then a HMM is trained alone for each gesture with promising accurate recognition.

Chen et al. [135] used Fourier Descriptors (FD) (of dimension 10) to represent hand gesture boundary. When normalized, FDs are invariant to gesture size, rotation and translation. In their attempt to use HMM to recognize gestures, they used three parameters: the initial state probability vector, the state transition probability matrix and the observable symbol probability matrix. Since a dynamic gesture has more transitions, the likelihood of dynamic gesture being classified

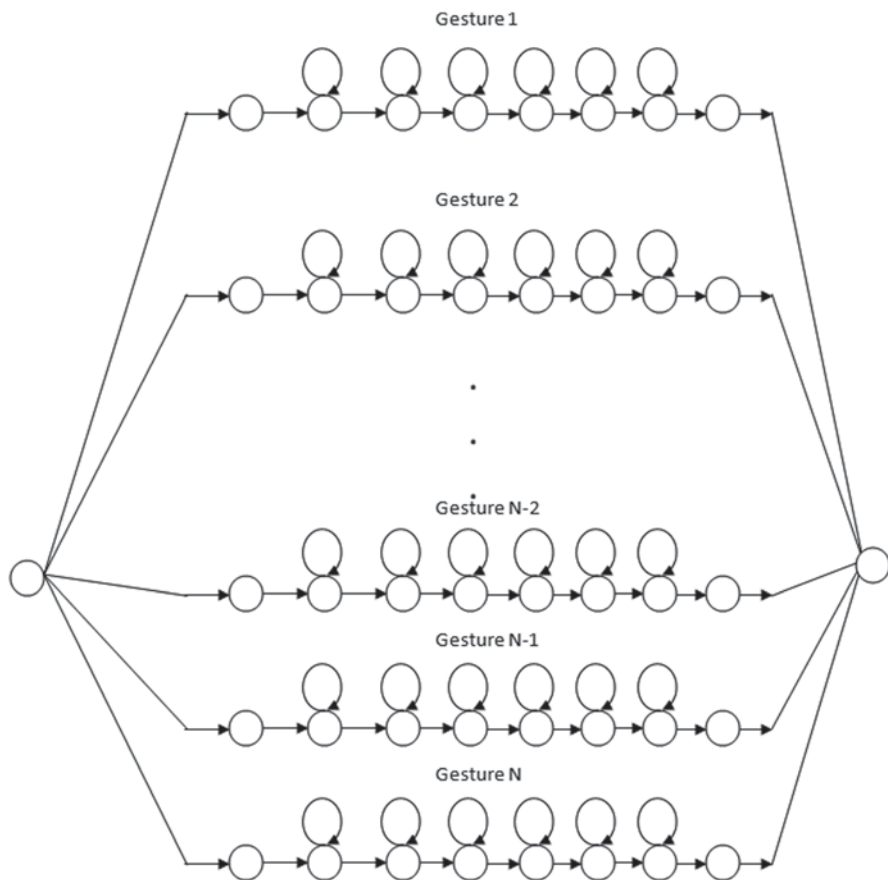
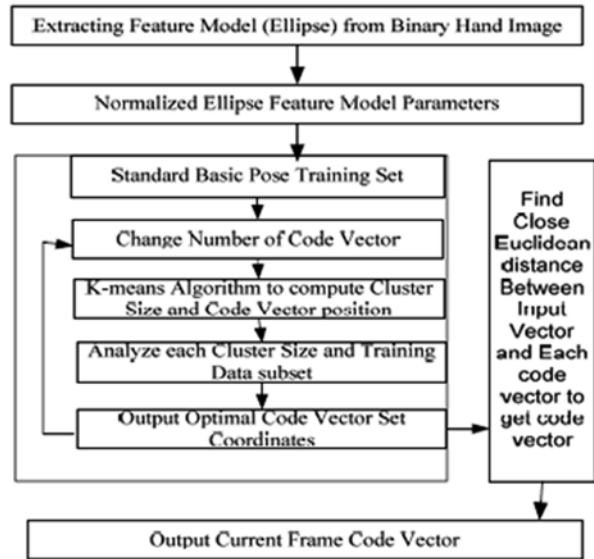


Fig. 5.26 A Typical HMM classification for dynamic hand gesture recognition, reproduced [129]

right is far higher than a single static gesture. To model various gesture expressions, they trained different HMMs to model different hand gestures. In order to remove redundancy in any gesture sequences, they applied vector quantization as preprocessing. They used a M -level VQ (referred to as a codebook with size M) to partition all k -dimensional training feature vectors into M clusters, whose centroid is the k -dimensional vector. They designed a technique where there would be a quantization error between the VQ and training data vector for each feature. The error decreases as the size of the codebook increases due to the availability of more training data.

Liu and Lovell [136] described a technique where they estimated a rotation angle of the hand direction after skin segmented hand region, and described a hand ‘blob’ using an ellipse model. They applied k -means Vector Quantization approach to determine the features using a HMM. In their quest to classify hand gestures using HMM, they made number of important conclusions:

Fig. 5.27 Vector quantization method used by Liu and Lovell [136]



1. Baum Welch is the traditional algorithm for HMM learning, while Viterbi path accounting is the new method (VPC). VPC performed steadily in comparison with Baum Welch, and is demonstrated to be a reliable method [136].
2. Model Structure variation. Structure doesn't greatly affect the recognition ratio. For Baum Welch, Left-Right is a slightly better than full connection, while in Viterbi path accounting, Full connection has a slightly higher correct ratio [136].
3. The number of states doesn't affect the correct ratio significantly. The effect on Baum Welch is greater than on Viterbi path accounting [136].

They reported an accuracy rate of 90% for 26 gestures. Their algorithm is denoted in Fig. 5.27 as a flowchart.

Classification completes the final link in associating hand gestures with their pre-assigned meanings in the context of hand gesture recognition. Yet, for any classification to be effective, feature extraction should provide the fundamental traits of any gesture by variety of users with many variables including, skin color, lighting condition, rotation and scale variations. This chapter discussed many techniques commonly used in linear and nonlinear classification while establishing their mathematical relationships providing in-depth insight into the problem of gesture recognition.

References

1. Chase, L.D.: Euclidean Distance (2008) <http://www.warnercnr.colostate.edu/~ldchase/Melinda's%20Final%20writeup.doc>. Accessed Oct. 12, 2013
2. Liberti, L., Lavor, C., Maculan, N., Mucherino, A.: Euclidean distance geometry and applications. Quantitative biology quantitative methods (2012)

3. Cantrell, C.D.: *Modern Mathematical Methods for Physicists and Engineers*. Cambridge University Press (2000)
4. Abello, J.M., Pardalos, P.M., Resende, M.G.C.: *Handbook of Massive Data Sets*. Springer (2002)
5. Tax, D.M.J., Duin, R., De Ridder, D.: *Classification, Parameter Estimation and State Estimation: An Engineering Approach Using MATLAB*. John Wiley and Sons (2004)
6. Mahalanobis, P.C.: On the generalised distance in statistics. *Proc. Natl. Inst. Sci. India* 2(1), 49–55 (1936)
7. Li, T., Zhu, S., Ogihara, M.: Using discriminant analysis for multi-class classification: an experimental investigation. *Knowl. Inf. Syst.* 10(4): 453–472 (2013)
8. Li, T., Zhu, S., Ogihara, M.: Using discriminant analysis for multi-class classification: an experimental investigation. *Knowl. Inf. Syst.* 10(4), 453–472 (2006)
9. Su, Y., Shan, S., Cao, B., Chen, X., Gao, W.: Multiple fisher classifiers combination for face recognition based on grouping AdaBoosted Gabor features. *Proceedings of the British Machine Vision Conference* (2005)
10. Fisher, R.A.: The use of multiple measurements in taxonomic problems. *Ann. Eugen.* 7(2), 179–188 (1936)
11. McLachlan, G.J.: *Discriminant Analysis and Statistical Pattern Recognition*. Wiley Interscience (2004)
12. Hendricks, D.: *Analyzing Quantitative Data: An Introduction for Social Researchers*, pp. 288–289 (2011)
13. Martinez, A.M., Kak, A.C.: PCA versus LDA. *IEEE Trans. Pattern Anal. Mach. Intell.* 23(2), 228–233 (2001)
14. Gupta, S., Jaafar, J., Ahmad, W.F.W.: Static hand gesture recognition using local gabor filter, international symposium on robotics and intelligent Sensors 2012. *Procedia Eng.* 41, 827–832 (2012)
15. Khan, A., Farooq, H.: Principal component analysis-linear discriminant analysis feature extractor for pattern recognition. *IJCSI Int. J. Comput. Sci.* 8(6), p276 (2011)
16. Suhas, S., Ajay, K., Khanale, P.: Face recognition using principal component analysis and linear discriminant analysis on holistic approach in facial images database. *IOSR J. Eng.* 2, 15–23 (2012)
17. Balakrishnama, S., Ganapathiraju, A.: Linear discriminant analysis- a brief tutorial. Institute for Signal and Information Processing. http://www.music.mcgill.ca/~ich/classes/mumt611/classifiers/lda_theory.pdf. Accessed Sept. 12, 2013
18. Khanale, P.B.: Face recognition against variation in pose and background. *IEEE International Conference on Electro/Information Technology* (2011)
19. Satonkar S.S., Kurhe A.B., Khanale P.B.: Face recognition methods & its applications. *J. Emerg. Technol. Appl. Eng., Technol. Sci. (IJ-ETA-ETS)* 4(2), 294–297 (2011)
20. Johnson, R.A., Wichern, D.W.: *Applied Multivariate Statistical Analysis*. Prentice Hall (1998)
21. Yu, H., Yang, J.: A direct LDA algorithm for high-dimensional data—with application to face recognition. *Pattern Recognit.* 34(10), 2067–2069 (2001)
22. Rosenblatt, F.: The perceptron—a perceiving and recognizing automaton. Report 85–460-1, Cornell Aeronautical Laboratory (1957)
23. Liou, D.-R., Liou, J.-W., Liou, C.-Y.: *Learning Behaviors of Perceptron*. iConcept Press (2013)
24. Mahesh P.: Multiclass approaches for support vector machine based land cover classification. *CORR* 2008 (2008)
25. Vapnik, V.N.: *The Nature of Statistical Learning Theory*. Springer-Verlag, New York (1995)
26. Chen, Y.T., Tseng, K.T.: Multiple-angle hand gesture recognition by fusing SVM classifiers, IEEE conference on Automation Science and Engineering, Scottsdale, AZ, USA, pp. 527–530 (2007)
27. Huang, D.-Y., Hu, W.-C., Chang, S.-H.: Vision-based hand gesture recognition using PCA + Gabor filters and SVM. *Fifth international conference on intelligent information hiding and multimedia signal processing* (2009)

28. Bonansea, L.: 3D Hand gesture recognition using a ZCam and an SVM-SMO classifier. Graduate Theses and Dissertations Graduate College, Iowa State University (2009)
29. Liu, Y., Gan, Z., Sun, Y.: Static Hand Gesture Recognition and its Application based on Support Vector Machines, pp. 517–521 (2008)
30. Chen, Y-T., Tseng, K-T.: Multiple-angle Hand Gesture Recognition by Fusing SVM Classifiers, pp. 527–530 (2007)
31. Bonansea, L.: Demonstration video of the 3D gesture recognition system using Zcam and SVM. http://www.youtube.com/watch?v=VsM0a_31I_Q (2009)
32. Ye, J., Yao, H., Jiang, F.: Based on HMM and SVM multilayer architecture classifier for Chinese sign language recognition with large vocabulary, pp. 377–380 (2004)
33. Joachims, T.: Text Categorization with Support Vector Machines: Learning with Many Relevant Features (1998) http://www.cs.cornell.edu/people/tj/publications/joachims_98a.pdf. Accessed April 18, 2013
34. Tong, S., Koller, D.: Support vector machine active learning with applications to text classification. *J Mach. Learn. Res.* 2001, 45–66 (2001)
35. Sassano, M.: Virtual Examples for Text Classification with Support Vector Machines. Fujitsu Laboratories Ltd (2003)
36. Basu, A., Watters, C., Shepherd, M.: Support vector machines for text categorization. *Proceedings of the 36th Hawaii International Conference on System Sciences* (2003)
37. Tong, S., Koller, D.: Support vector machine active learning with applications to text classification. In *Proceedings of the Seventeenth International Conference on Machine Learning (ICML-00)*, pp. 287–295 (1998)
38. Cortes, C., Vapnik, V. N.: Support-Vector Networks, *Machine Learning*, p. 20 (1995)
39. Press, W.H., Teukolsky, S.A., Vetterling, W.T., Flannery, B. P.: *Support Vector Machines. Numerical Recipes: The Art of Scientific Computing*, 3rd edn. Cambridge University Press, New York (2007)
40. Aizerman, M.A., Braverman, E.M., Rozonoer, L.I.: Theoretical foundations of the potential function method in pattern recognition learning. *Autom. Remote Control.* 25, 821–837 (1964)
41. Boser, B.E., Guyon, I.M., Vapnik, V.N.: A training algorithm for optimal margin classifiers. In: Haussler, D (ed.) *5th Annual ACM Workshop on COLT*, pp. 144–152 (1992)
42. Meyer, D., Leisch, F., Hornik, K.: The support vector machine under test, *Neurocomputing* 55(1–2), 169–186 (2003)
43. Hsu, C-W, Chang, C-C., Lin, C-J.: *A Practical Guide to Support Vector Classification* (Technical report). Department of Computer Science and Information Engineering, National Taiwan University (2003)
44. Duan, K-B., Keerthi, S. S.: Which is the best multiclass SVM method? An empirical study. *Proceedings of the Sixth International Workshop on Multiple Classifier Systems. Lecture Notes in Computer Science* vol. 3541, p. 278 (2005)
45. Hsu, C-W., Lin, C-J.: A comparison of methods for multiclass support vector machines. *IEEE Transactions on Neural Networks* (2002)
46. Platt, J., Cristianini, N., Shawe-Taylor, J.: Large margin DAGs for multiclass classification. In: Solla, S.A., Leen, T.K., Müller, K-R. (eds.) *Advances in Neural Information Processing Systems*, pp. 547–553. MIT Press (2000)
47. Dietterich, T.G., Bakiri, G.B.: Solving multiclass learning problems via error-correcting output codes. *J. Artif. Intell. Res.* 2(2), 263–286 (1995)
48. Crammer, K., Singer, Y.: On the algorithmic implementation of multiclass Kernel-based vector machines. *J. Mach. Learn. Res.* 2, 265–292 (2001)
49. Lee, Y., Lin, Y., Wahba, G.: Multicategory support vector machines. *Computing Science and Statistics*, p. 33 (2001)
50. Lee, Y., Lin, Y., Wahba, G.: Multicategory support vector machines, theory, and application to the classification of microarray data and satellite radiance data. *J. Am. Stat. Assoc.* 99(465), 67–81 (2004)

51. Joachims, T.: Transductive inference for text classification using support vector machines. *Proceedings of the 1999 International Conference on Machine Learning (ICML 1999)*, pp. 200–209 (1999)
52. Drucker, H., Burges, C.J.C., Kaufman, L., Smola, A.J., Vapnik, V.N.: Support vector regression machines. In *Advances in Neural Information Processing Systems 9*, NIPS 1996, pp. 155–161 (1997)
53. Suykens, J.A.K., Vandewalle, J.P.L.: Least squares support vector machine classifiers. *Neural Process. Lett.* 9(3), 293–300 (1999)
54. Ferris, M. C. and Munson, T. S.: Interior-point methods for massive support vector machines. *SIAM J Optim.* 13(3), 783–804 (2002)
55. Knerr, S., Personnaz, L., Dreyfus, G.: Single-layer learning revisited: a stepwise procedure for building and training neural network. *Neurocomputing: Algorithms, Architectures and Applications*, NATO ASI. Springer-Verlag, Berlin (1990)
56. Hsu, C.-W., Lin, C.-J.: A comparison of methods for multi-class support vector machines, *IEEE Trans. Neural Netw.* 13, 415–425 (2002)
57. JAMES, G.: Majority vote classifiers: Theory and Applications. Ph. D. Thesis, Department of Statistics, Stanford University, Stanford, CA (1998)
58. Lee, Y., Lin, Y., Wahba, G.: Multicategory support vector machines Tech. Rep. 1043, Department of Statistics, University of Wisconsin, Madison, (2001)
59. Schölkopf, B., Smola, A. J.: *Learning with Kernels—Support Vector Machines, Regularization, Optimization and Beyond*. The MIT Press, Cambridge (2002)
60. Weston, J., Watkins, C.: *Multi-class Support Vector Machines*. Royal Holloway, University of London, U. K., Technical Report CSD-TR-98–04 (1998)
61. Piyush, R.: *Kernel Methods and Nonlinear Classification CS5350/6350: Machine Learning* (2011)
62. Berwick, R.: An Idiot’s guide to Support vector machines (SVMs) <http://www.web.mit.edu/6.034/wwwbob/svm-notes-long-08.pdf>. Accessed Oct. 15, 2013
63. Scribe, M.I.J., Thibaux, R.: The kernel trick. *Advanced Topics in Learning & Decision Making* (2004) <http://www.cs.berkeley.edu/~jordan/courses/281B-spring04/lectures/lec3.pdf>. Accessed April 18, 2013
64. Zien, A., Rätsch, G., Mika, S., Schölkopf, B., Lengauer, T., Müller, K.-R.: Engineering support vector machine kernels that recognize translation initiation sites. *Bioinformatics* 16(9), 799–807 (2000)
65. Blankertz, B., Curio, G., Müller, K.-R.: Classifying single trial EEG: towards brain computer interfacing. In: Diettrich, T.G., Becker, S., Ghahramani, Z., (eds.) *Advances in Neural Information Processing Systems*, vol. 14, pp. 157–164 (2002)
66. Bishop, C.M.: *Neural Networks for Pattern Recognition*. Oxford University Press (1995)
67. Moody, J., Darken, C.: Fast learning in networks of locally-tuned processing units. *Neural Comput.* 1(2), 281–294 (1998)
68. Freund, Y., Schapire, R.E.: A decision-theoretic generalization of on-line learning and an application to boosting. *J. Comp. Syst. Sci.* 55(1), 119–139 (1997)
69. Rätsch, G., Mika, S., Schölkopf, B., Müller, K.-R.: Constructing boosting algorithms from SVMs: an application to one-class classification. *IEEE Patt. Anal. Mach. Intell. (IEEE PAMI)* 24(9), 1184–1199 (2002)
70. Girosi, F., Jones, M., Poggio, T.: Priors, stabilizers and basis functions: from regularization to radial, tensor and additive splines. Technical Report A.I. Memo No. 1430, Massachusetts Institute of Technology (1993)
71. Smola, A.J., Schölkopf, B., Müller, K.-R.: The connection between regularization operators and support vector kernels. *Neural Netw.* 11, 637–649 (1998)
72. Girosi, F.: An equivalence between sparse approximation and support vector machines. *Neural Comput.* 10, 1455–1480 (1998)
73. Haussler, D.: Convolution kernels on discrete structures. Technical Report UCSC-CRL-99–10, UC Santa Cruz (1999)

74. Watkins, C.: Dynamic alignment kernels. In: Smola, A.J., Bartlett, P.L., Schölkopf, B., Schuurmans, D. (eds.) *Advances in Large Margin Classifiers*, pp. 39–50 (2000)
75. Schölkopf, B.: The kernel trick for distances. In: Leen, T.K., Diettrich, T.G., Tresp, V. (eds.) *Advances in Neural Information Processing Systems 13*. MIT Press (2001)
76. Osuna, E., Freund, R., Girosi, F.: Training support vector machines: an application to face detection. In *Proceedings CVPR'97* (1997)
77. Platt, J.: Fast training of support vector machines using sequential minimal optimization. In: Schölkopf, B., Burges, C.J.C., Smola, A.J. (eds) *Advances in Kernel Methods—Support Vector Learning*, pp. 185–208 (1999)
78. Ralaivola, L., d'Alché Buc, F.: Incremental support vector machine learning: a local approach. *Lect. Notes Comput. Sci.* 2130, 322–329 (2001)
79. Schölkopf, B., Burges, C.J.C., Vapnik, V.N.: Extracting support data for a given task. In: Fayyad, U.M., Uthurusamy, R. (eds.) *Proceedings, First International Conference on Knowledge Discovery & Data Mining* (1995)
80. Schölkopf, B., Smola, A., Williamson, R.C., Bartlett, P.L.: New support vector algorithms. *Neural Comput.* 12, 1207–1245 (2000)
81. Schölkopf, B., Smola, A.J.: *Learning with Kernels*. MIT Press, Cambridge (2002)
82. Schölkopf, B., Smola, A.J., Müller, K.-R.: Nonlinear component analysis as a kernel eigenvalue problem. *Neural Comput.* 10, 1299–1319 (1998)
83. Simard, P.Y., LeCun, Y.A., Denker, J.S., Victorri, B.: Transformation invariance in pattern recognition—tangent distance and tangent propagation. In: Orr, G., Müller, K.-R. (eds.) *Neural Networks: Tricks of the Trade*, LNCS 1524, pp. 239–274 (1998)
84. Afrin, M.H.RH.: Hand gesture recognition using multiclass support vector machine. *Int. J. Comput. Appl.* 74(1), 39–43 (2013)
85. Chen, Y.-T., Tseng, K.-T.: Multiple-angle hand gesture recognition by fusing SVM classifiers. *Proceedings of the 3rd Annual IEEE Conference on Automation Science and Engineering*, pp. 527–530 (2007)
86. Altman, N.S.: An introduction to kernel and nearest-neighbor nonparametric regression. *Am. Stat.* 46(3), 175–185 (1992)
87. Coomans, D., Massart, D.L.: Alternative k-nearest neighbor rules in supervised pattern recognition: Part 1. K-Nearest neighbor classification by using alternative voting rules. *Anal. Chim. Acta* 136, 15–27 (1982)
88. Bremner D., Demaine E., Erickson J., Iacono J., Langerman S., Morin P., Toussaint G.: Output-sensitive algorithms for computing nearest-neighbor decision boundaries. *Discret. Comput. Geom.* 33(4), 593–604 (2005)
89. <http://www.analyticbridge.com/forum/topics/clustering-idea-for-very-large-datasets>
90. Pujan, Z., Müller, T., Foster, M.E., Knoll, A.: A Naïve Bayes Classifier with Distance Weighting for Hand-Gesture Recognition. *CSICC 2008, CCIS 6*, 308–315 (2008)
91. Pujan, Z., Müller, T., Foster, M.E., Knoll, A.: Using a Naïve Bayes classifier based on k-nearest neighbors with distance weighting for static hand-gesture recognition in a human-robot dialog system. *Adv. Comput. Sci. Eng.* 6(1–8), 308–315 (2008)
92. Vafadar, M., Behrad, A.: Human hand gesture recognition using Spatio-temporal volumes for human-computer Interaction. *International Symposium on Telecommunications*, pp. 713–718 (2008)
93. Kollorz, E., Penne, J., Horneegger, J., Barke, A.: Gesture recognition with a time-of-flight camera. *Int. J. Intell. Syst. Technol. Appl.* 5–34, 334–343 (2008)
94. http://www.byclb.com/TR/Tutorials/neural_networks/ch6_1.htm
95. Haykin, S.: *Neural Network—a Comprehensive Foundation; a Computational Approach to Learning and Machine Intelligence*, Macmillan (1994)
96. Zurada, J.M.: *Introduction to Artificial Neural Networks System*. Jaico Publishing House (1992)
97. Freeman: *Artificial Neural Network Algorithm. Applications and Programming*, Comp and Neural Systems Series, Addison-Wesley Pub (Sd) (1990)

98. Kulkarni, A.: Artificial Neural Network for Image Understanding. Reinhold, New York (1994)
99. Anderson, J.: An Introduction to Neural Network. A Bradford Book (1995)
100. Ranjan, A.: A New Approach for Blind Source Separation of Convolutional Sources (2008)
101. Carpenter, G.A., Grossberg, S.: The ART of adaptive pattern recognition by a self-organizing neural network. *Computer* **21**, 77–88 (1998)
102. Hinton, G., Sejnowski, T.J. (ed.): Unsupervised Learning: Foundations of Neural Computation, MIT Press (1999)
103. Duda, R.O., Hart, P.E., Stork, D.G.: Unsupervised Learning and Clustering, Chapter 10 in *Pattern classification*, 2nd edn. Wiley, New York, p. 571 (2001)
104. Ghahramani, Z.: Unsupervised Learning (2004) <http://mlg.eng.cam.ac.uk/zoubin/papers/ul.pdf>. Accessed April 18, 2013
105. Williams, R.J.: A class of gradient-estimating algorithms for reinforcement learning in neural networks. *Proceedings of the IEEE First International Conference on Neural Networks* (1987)
106. Sutton, R.S.: Learning to Predict by the Method of Temporal Differences. *Machine Learning* (Springer), vol. 3, pp. 9–44 (1998).
107. Bradtke, S.J., Barto, A.G.: Learning to Predict by the Method of Temporal Differences. *Machine Learning* (Springer), vol. 22, pp. 33–57 (1996)
108. Bertsekas, D.P., Tsitsiklis, D.: *Neuro-Dynamic Programming*. Athena Scientific, Nashua (1996)
109. Kaelbling, L.P., Littman, M.L., Moore, A.W.: Reinforcement learning: a survey. *J. Artif. Intell. Res.* **4**, 237–285 (1996)
110. Peters, J., Vijayakumar, S., Schaal, S.: Reinforcement learning for humanoid robotics. *IEEE-RAS International Conference on Humanoid Robots* (2003)
111. Powell, W.: *Approximate Dynamic Programming: Solving the Curses Of Dimensionality*. Wiley-Interscience (2007)
112. Auer, P., Jaksch, T., Ortner, R.: Near-optimal regret bounds for reinforcement learning. *J. Mach. Learn. Res.* **11**, 1563–1600 (2010)
113. Szita, I., Szepesvari, C.: Model-based Reinforcement Learning with Nearly Tight Exploration Complexity Bounds. *ICML 2010*, pp. 1031–1038 (2008)
114. Bertsekas, D.P.: *Approximate Dynamic Programming. Dynamic Programming and Optimal Control II*, 3rd edn. (2010)
115. Busoniu, L., Babuska, R., De Schutter, B., Ernst, D.: *Reinforcement Learning and Dynamic Programming using Function Approximators*. Taylor & Francis CRC Press (2010)
116. Tokic, M., Palm, G.: Value-difference based exploration: adaptive control between Epsilon-Greedy and Softmax. *KI 2011: advances in Artificial intelligence. Lecture Notes in Computer Science*, vol. 7006, pp. 335–346 (2011)
117. Wu, J., Chen, E., Wang, H., Shen, Y., Huang, T., Zeng, Z.: A Novel nonparametric regression ensemble for rainfall forecasting using particle swarm optimization technique coupled with artificial neural network. *6th International Symposium on Neural Networks* (2009)
118. De Rigo, D., Castelletti, A., Rizzoli, A.E., Soncini-Sessa, R., Weber, E.: A selective improvement technique for fastening Neuro-Dynamic programming in water resources network management. In: Zitek, P. (ed.) *Proceedings of the 16th IFAC World Congress* (2005)
119. Ferreira, C.: Designing neural networks using gene expression programming. In: Abraham, A., de Baets, B., Köppen, M., Nickolay, B. (eds.) *Applied Soft Computing Technologies: The Challenge of Complexity*, pp. 517–536 (2006)
120. Da, Y., Xiurun, G., Villmann, T.: An improved PSO-based ANN with simulated annealing technique. *New Aspects in Neurocomputing: 11th European Symposium on Artificial Neural Networks*. Elsevier (2005)
121. Balabin, R.M., Lomakina, E.I.: Neural network approach to quantum-chemistry data: accurate prediction of density functional theory energies. *J. Chem. Phys.* **131**(7), 1–8 (2009)

122. Ganesan, N. Venkatesh, K., Rama M.A.: Application of neural networks in diagnosing cancer disease using demographic data. *International Journal of Computer Applications* 1(26), 76–85 (2010)
123. Bottaci, L. Drew, P.J., Hartley, J.E., Hadfield, M.B., Farouk, R., Lee, P.W., Macintyre, I.M., Duthie, G.S., Monson, J.R.: Artificial Neural Networks Applied to Outcome Prediction for Colorectal Cancer Patients in Separate Institutions. *The Lancet* 350(9076), 469–472 (1997)
124. Premaratne, P., Safaei, F., Nguyen, Q.: Moment invariant based control system using hand gestures: book intelligent computing in signal processing and pattern recognition. *Book Series Lecture Notes in Control and Information Sciences*, vol. 345, pp. 322–333 (2006)
125. Gutta, S., Imam, I.F., Wechsler, H.: Hand gesture recognition using ensembles of radial basis functions (RBF) networks and decision trees. *Int. J. Patt. Recognit. Artif. Intell.* 11(6) (1997)
126. Murthy, G.R.S., Jadon, R.S.: Hand gesture recognition using neural networks. *IEEE 2nd International Advance Computing Conference Artificial Intelligence*, pp. 134–138 (2010)
127. Hasan, H., Abdul-Kareem, S.: Static hand gesture recognition using neural networks. *Artificial Intelligence Review* 41:147–181 (2012)
128. Zheng, X., Koenig, S.: A Project on Gesture Recognition with Neural Networks for Introduction to Artificial intelligence Classes (2010)
129. Min, B-W., Yoon, H-S., Soh, J., Yang, Y-M., Ejima, T.: Hand gesture recognition using hidden Markov models. *1997 IEEE International Conference on Systems, Man, and Cybernetics on Computational Cybernetics and Simulation*, vol. 5, pp. 4232–4235 (1997)
130. Yang, L., Xu, Y., Chen, C.S.: Human action learning via hidden Markov model. *IEEE Trans. Syst. Man. Cybern.* 27(1), 34–44 (1997)
131. Yang, L., Xu, Y.: Hidden Markov model for gesture recognition. Thesis, The Robotics Institute Carnegie Mellon University (1994)
132. Kadous, W.: Machine learning is a subfield of artificial intelligence. PhD Thesis, University of New South Wales (2002)
133. Viterbi, A.J.: Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Trans. Inf. Theory* 13(2), 260–269 (1967)
134. Rabiner, L.: First Hand: The Hidden Markov Model. *IEEE Global History Network*. http://www.ieeeeghn.org/wiki/index.php/First-Hand:The_Hidden_Markov_Model. Accessed Aug. 24, 2013
135. Yang, Z., Li, Y., Chen, W., Zheng, Y.: Dynamic hand gesture recognition using hidden Markov models. *7th International Conference on Computer Science & Education (ICCSE)*, pp. 360–365 (2012)
136. Chen, F.S., Fu, C.M., Huang, C.L.: Hand gesture recognition using a real-time tracking method and hidden Markov models. *Image Vision Comput.* 21(8), 745–758 (2003)