Chapter 8

Support Vector Machines for Face Recognition

8.1 Introduction

In chapter 7 we have investigated the credibility of different parameters introduced in the present work, viz., SSPD and ALR Feature Vectors by performing recognition experiments on the human face database using c-Means clustering technique, k-NN classifier, Bayesian Classifier and Artificial Neural Network. The recognition accuracies obtained in these experiments also reveal the need of an improved system that can effectively handle the large variations present in the face image samples caused by the various poses and other conditions. The best alternative suggested in this study is Support Vector Machines (SVM), pioneered by Vapnik [Vapnik 1998], [Cortes & Vapnik 1995], [Boser et al. 1992], [Vapnik 1982]. The support vector machine has the inherent ability to solve a pattern-classification problem in a manner close to the optimum for the problem of interest. Moreover, it is able to achieve such a remarkable performance with no problem domain knowledge built into the design of the machine.

Basically, Support Vector Machine is a linear machine with some nice properties. In the context of pattern classification, the main idea of a support vector machine is to construct a hyperplane as the decision surface in such
8. Support Vector Machines for Face Recognition

a way that the margin of separation between positive and negative examples is maximized. The machine achieves this desirable property by following a principled approach rooted in statistical learning theory, structural risk minimization. This induction principle is based on the fact that the error rate of a learning machine on test data (i.e., generalization error report) is bounded by the sum of the training-error rate and the term that depends on the Vapnik-Chervonenkis (VC) dimension [Vapnik 1982], [Vapnik 1998]. In the case of separable patterns, the support vector machine produces a value of zero for the first term and minimizes the second term. Accordingly, the support vector machine can provide a good generalization performance on pattern classification problems despite the fact that it does not incorporate problem-domain knowledge. This attribute is unique to support vector machines [Haykin 2001], [Cristianini & Shawe-Taylor 2000].

In this study, an effort is made to build a recognition system using the support vector machine for human face recognition. A notion that is central to the construction of the support vector learning algorithm is the inner-product kernel between a support vector $\mathbf{x}_i$ and the vector $\mathbf{x}$ drawn from the input space. The support vectors consist of small subset of the training data extracted by the algorithm. We used $SSPD$, and $ALR Feature Vectors$ discussed in the previous chapters for human face recognition. The recognition results show that this method is more efficient and can be adopted for developing a complete face recognition system for human face images.

A relatively new learning architecture is used in the present study: the Decision Directed Acyclic Graph (DDAG), which is used to combine many two-class classifiers into a multiclass classifier. For an N-class problem, the DDAG contains classifiers, one for each pair of classes. DAGSVM operates in a kernel- induced feature space and uses two-class maximal margin hyperplanes
8. Support Vector Machines for Face Recognition

at each decision-node of the DDAG. The DAGSVM is substantially faster to train and evaluate than either the standard algorithm or Max Wins, while maintaining comparable accuracy to both of these algorithms.

The necessary theoretical background required for the design of a SVM is presented in this chapter. The section 8.2 describes the optimal hyperplane for linearly separable patterns, section 8.3 gives the procedure and basic theory of support vector machine for pattern classification and details of DAGSVM method. Section 8.4 discusses simulation experiments and results obtained using support vector machines.

8.2 Optimal hyperplane for linearly separable face patterns

Consider training sample \((x_i, d_i)^N\), where \(x_i\) is the input pattern for the \(i^{th}\) example and \(d_i\) is the corresponding desired response (target output). To begin with, we assume that the pattern (class) represented by the subset \(d_i = +1\) and the pattern represented by the subset \(d_i = -1\) are linearly separable. The equation of surface in the form of a hyperplane that does the separation is

\[
\mathbf{w}^T \mathbf{x} + b = 0
\]  

(8.2.1)

where \(\mathbf{x}\) is an input vector, \(\mathbf{w}\) is an adjustable weight vector, and \(b\) is a bias. Then it could be written as

\[
\mathbf{w}^T \mathbf{x}_i + b \geq 0 \text{ for } d_i = +1
\]

\[
\mathbf{w}^T \mathbf{x}_i + b < 0 \text{ for } d_i = -1
\]  

(8.2.2)

For a given weight vector \(\mathbf{w}\) and a bias \(b\), the separation between the hyperplane defined in Eq. 8.2.1 and the closest data point is called the \textit{margin of separation}, denoted by \(\rho\). The goal of a support vector machine is to find
Figure 8.1: Optimal hyperplane for linearly separable patterns.

the particular hyperplane for which the margin of separation $\rho$ is maximized. Under this condition, the decision surface is the optimal hyperplane. Figure 8.1 illustrates the geometric construction of an optimal hyperplane for a two-dimensional input space. Let $w_o$ and $b_o$ denote the optimum values of the weight vector and bias, respectively. Correspondingly, the optimal hyperplane, representing a multidimensional linear decision surface in the input space, is defined by

$$w_o^T x + b_o = 0 \quad (8.2.3)$$

the discriminant function

$$g(x) = w_o^T x + b_o \quad (8.2.4)$$

gives an algebraic measure of the distance from $x$ to the optimal hyperplane [Duda & Hart 1973]. The easiest way to see this is to express $x$ as

$$x = x_p + r \frac{w_o}{\|w_o\|}$$
8. Support Vector Machines for Face Recognition

**Figure 8.2:** Geometric interpretation of algebraic distances of points to the optimal hyperplane for a 2D case.

where \( x_p \) is the normal projection of \( x \) onto the optimal hyperplane, and \( r \)
is the desired algebraic distance; \( r \) is positive if \( x \) on the positive side of the optimal hyperplane and negative if \( x \) is on the negative side. Since, by definition, \( g(x_p) = 0 \), it follows that

\[
g(x) = w_o^T x + b_o = r \| w_o \|
\]

or

\[
r = \frac{g(x)}{\| w_o \|} \tag{8.2.5}
\]

In particular, the distance from the origin (i.e., \( x=0 \)) to the optimal hyperplane is given by \( \frac{b_o}{\| w_o \|} \). If \( b_o > 0 \), the origin is on the positive side of the optimal hyperplane; if \( b_o < 0 \), it is on the negative side. If \( b_o = 0 \), the optimal hyperplane passes through the origin. A geometric interpretation of these algebraic results is shown in figure 8.2.

The issue at hand is to find the parameters \( w_o \) and \( b_o \) for the optimal hyperplane, given the training set \( \mathcal{S} = \{(x_i, d_i)\} \). In light of the result portrayed in Figure 8.2, we see that the pair \((w_o, b_o)\) must satisfy the constraint:

\[
w_o^T x_i + b_o \geq 1 \quad \text{for } d_i = +1
\]
Thus if the patterns are linearly separable (i.e., Eq. 8.2.2 holds), we can rescale $w_o$ and $b_o$ such that the equation 8.2.6 holds; this scaling operation leaves Eq. 8.2.3 unaffected.

The particular data points $(x_i, d_i)$ for which the first or the second line of Eq. 8.2.6 is satisfied with the equality sign are called support vectors, hence the name support vector machine. These vectors play a prominent role in the operation of this class of learning machines. In conceptual term the support vectors are those data points that lie closest to the decision surface and are therefore the most difficult to classify.

Consider a support vector $x^{(s)}$ for which $d^{(s)} = \pm 1$. Then by definition, we have

\[ g(x^{(s)}) = w_o^T x^{(s)} \mp b_o = \mp 1 \text{ for } d^{(s)} = \mp 1 \]  

(8.2.7)

from Eq. 8.2.5 the algebraic distance from the support vector $x^{(s)}$ to the optimal hyperplane is

\[ r = \frac{g(x^{(s)})}{\|w_o\|} \]

\[ = \begin{cases} 
\frac{1}{\|w_o\|} & \text{if } d^{(s)} = +1 \\
-\frac{1}{\|w_o\|} & \text{if } d^{(s)} = -1 
\end{cases} \]  

(8.2.8)

where plus sign indicates that $x^{(s)}$ lies on the positive side of the optimal hyperplane and minus sign indicates that $x^{(s)}$ lies on the negative side of the optimal hyperplane. Let $\rho$ denote the optimum value of the margin of separation between the classes that constitutes the training set. Then from Eq. 8.2.8 it follows that

\[ \rho = 2r \]

\[ = \frac{2}{\|w_o\|} \]  

(8.2.9)

\[ 173 \]
Equation 8.2.9 states that maximizing the margin of separation between classes is equivalent to minimizing Euclidean norm of the weight vector $\mathbf{w}$.

### 8.2.1 Quadratic optimization for finding the optimal hyperplane

Objective of the design of an SVM is to develop a computationally efficient procedure for using the training sample $\mathcal{S} = \{(\mathbf{x}_i, d_i)\}_{i=1}^N$ to find the optimal hyperplane, subject to the constraint

$$d_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 \quad \text{for } i = 1, 2, 3 \ldots N \quad (8.2.10)$$

This constraint combines the two lines of Eq. 8.2.6 with $\mathbf{w}$ used in place of $\mathbf{w}_0$. The constrained optimization problem that we have to solve may now be stated as:

*Given the training sample $\mathcal{S} = \{(\mathbf{x}_i, d_i)\}_{i=1}^N$, find the optimum values of the weight vector $\mathbf{w}$ and bias $b$ such that they satisfy the constraints

$$d_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 \quad \text{for } i = 1, 2, 3 \ldots N$$

and the weight vector $\mathbf{w}$ minimizes the cost function:

$$\Phi(\mathbf{w}) = \frac{1}{2} \mathbf{w}^T \mathbf{w}$$

The scaling factor $1/2$ is included here for convenience of presentation. This constrained optimization problem is called the *primal problem*. It is characterized as follows:

- The cost function $\Phi(\mathbf{w})$ is a *convex* function of $\mathbf{w}$
- The constraints are *linear* in $\mathbf{w}$

Accordingly, we can solve the constrained optimization problem using the method of *Lagrange multipliers* as follows[Bertsekas 1995].
First, we construct the **Lagrangian function**:

\[
J(w, b, \alpha) = \frac{1}{2} w^T w - \sum_{i=1}^{N} \alpha_i \left[d_i (w^T x_i + b) - 1\right]
\]  

(8.2.11)

where the auxiliary nonnegative variables \(\alpha_i\) are called Lagrangian multipliers. The solution to the constrained optimization problem is determined by the saddle point of the Lagrangian function \(J(w, b, \alpha)\), which has to be minimized with respect to \(w\) and \(b\); it also has to be maximized with respect to \(\alpha\). Thus, differentiating \(J(w, b, \alpha)\) with respect to \(w\) and \(b\) and setting the results equal to zero, we get the following two conditions of optimality:

**Condition 1:**

\[
\frac{\partial J(w, b, \alpha)}{\partial w} = 0
\]

**Condition 2:**

\[
\frac{\partial J(w, b, \alpha)}{\partial b} = 0
\]

Application of optimality condition 1 to the Lagrangian function of Eq. 8.2.11 yields

\[
w = \sum_{i=1}^{N} \alpha_i d_i x_i
\]

(8.2.12)

Application of optimality condition 2 to the Lagrangian function of Eq. 8.2.11 yields

\[
\sum_{i=1}^{N} \alpha_i d_i = 0
\]

(8.2.13)

The solution vector \(w\) is defined in terms of an expansion that involves the \(N\) training examples. Although this solution is unique by virtue of the convexity of the Lagrangian, the same cannot be said about the Lagrange coefficients, \(\alpha_i\).

It is also important to note that all the saddle point, for each Lagrange multiplier \(\alpha_i\), the product of that multiplier with its corresponding constraint vanishes, as shown by

\[
\alpha_i \left[d_i (w^T x_i + b) - 1\right] = 0 \quad \text{for} \ i = 1, 2, \ldots, N
\]

(8.2.14)
Therefore, only those multipliers exactly meeting Eq. 8.2.14 can assume nonzero values. This property follows from the Kuhn-Tucker conditions of optimization theory [Fletcher 1987], [Bertsekas 1995].

The primal problem deals with a convex cost function and linear constraints. Given such a constrained optimization problem, it is possible to construct another problem called the dual problem. This second problem has the same optimal value as the primal problem, but with the Lagrange multipliers providing the optimal solution. In particular we may state the following duality theorem[Bertsekas 1995], [Haykin 2001].

(a) If the primal problem has an optimal solution, the dual problem also has an optimal solution, and the corresponding optimal values are equal.

(b) In order for \( w_0 \) to be an optimal primal solution and \( \alpha_0 \) to be an optimal dual solution, it is necessary and sufficient that \( w_0 \) is feasible for the primal problem, and

\[
\Phi(w_0) = J(w_0, b_0, \alpha_0) = \min_w J(w, b, \alpha)
\]

Dual problem can be postulated first by expanding Eqn. 8.2.11 , term by term, as follows:

\[
J(w, b, \alpha) = \frac{1}{2} w^T w - \sum_{i=1}^{N} \alpha_i d_i w^T x_i - b \sum_{i=1}^{N} \alpha_i d_i + \sum_{i=1}^{N} \alpha_i \tag{8.2.15}
\]

The third term on the right hand side of Eq. 8.2.15 is zero by virtue of the optimality condition of Eq. 8.2.13. Furthermore, from Eq. 8.2.12 we have

\[
w^T w = \sum_{i=1}^{N} \alpha_i d_i w^T x_i = \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j d_i d_j x_i^T x_j
\]

Accordingly, setting the objective function \( J(w, b, \alpha) = Q(\alpha) \), Eq. 8.2.15 may be reformulated as

\[
Q(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j d_i d_j x_i^T x_j \tag{8.2.16}
\]
where the $\alpha_i$ are nonnegative.

The dual problem is stated as: Given the training sample $\{(x_i, d_i)\}_{i=1}^N$, find the Lagrange multipliers $\{\alpha_i\}_{i=1}^N$ that maximize the objective function

$$Q(\alpha) = \sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j d_i d_j x_i^T x_j$$

Subject to the constraints

1. $\sum_{i=1}^N \alpha_i d_i = 0$
2. $\alpha_i \geq 0$ for $i = 1, 2, \ldots, N$

The dual problem is cast entirely in terms of the training data. Moreover, the function $Q(\alpha)$ to be maximized depends only on the input pattern in the form of a set of dot products, $\{x_i^T x_j\}_{(i,j)=1}^N$. Having determined the optimum Lagrange multipliers, denoted by $\alpha_{0,i}$, optimum weight vector $w_o$ is computed using Eq. 8.2.12 and so write

$$w_o = \sum_{i=1}^N \alpha_{0,i} d_i x_i \quad (8.2.17)$$

Optimum bias $b_o$ is computed using $w_o$ by applying Eq. 8.2.7 pertaining to a positive support vector as

$$b_o = 1 - w_o^T x^{(s)} \quad \text{for } d^{(s)} = 1 \quad (8.2.18)$$

### 8.2.2 Optimal hyperplane for nonseparable patterns

When training data contains non-separable patterns, it is not possible to construct a separating hyperplane without encountering classification errors. In this context, optimal hyperplane that minimizes probability of classification error, averaged over the training set is found first. The margin of separation between classes is said to be soft if a data point $(x_i, d_i)$ violates following condition $d_i (w^T x_i + b) \geq 1$ for $i = 1, 2, 3 \ldots N$ This violation can arise in one of two ways:
• The data point \((x_i, d_i)\) falls inside the region of separation but on the right side of the decision surface, as illustrated in figure 8.3(a)

• The data point \((x_i, d_i)\) falls on the wrong side of the decision surface, as illustrated in figure 8.3(b)

To set the stage for a normal treatment of nonseparable data points, a new set of nonnegative scalar variables, \(\{\xi_i\}_{i=1}^{N}\), is introduced into the definition of separating hyperplane (i.e., decision surface) as

\[
d_i(w^T x_i + b) \geq 1 - \xi_i, \quad i = 1, 2, ..., N
\]  

(8.2.19)

The \(\xi_i\) are called slack variables; they measure the deviation of data point from the ideal condition of pattern separability. For \(0 \leq \xi_i \leq 1\), the data point falls inside the region of separation but on the right side of the decision surface, as illustrated in figure 8.3(a). For \(\xi_i > 1\), it falls on the wrong side of the separating hyperplane, as illustrated in figure 8.3(b). The support vectors are those particular data points that satisfy Eq. 8.2.19 precisely even if \(\xi_i > 0\). If an example with \(\xi_i > 0\) is left out of the training set, the decision surface would change. The support vectors are thus defined in exactly the same way for both linearly separable and nonseparable cases.

The goal here is to find a separating hyperplane for which the misclassification error, averaged on training set, is minimized. This is done by minimizing the function

\[
\Phi(\xi) = \sum_{i=1}^{N} I(\xi_i - 1)
\]

with respect to the weight vector \(w\), subject to the constraints described in Eq. 8.2.19 and the constraint on \(\|w\|^2\). The function \(I(\xi_i)\) is an indicator function, defined by

\[
I(\xi) = \begin{cases} 
0 & \text{if } \xi \leq 0 \\
1 & \text{if } \xi > 0 
\end{cases}
\]

178
(a) Data point $x_i$ (belonging to class $C_1$) falls inside the region of separation, but on the right side of the decision surface.

(b) Data points $x_i$ (belonging to class $C_2$) falls on the wrong side of the decision surface.

Figure 8.3: Support vectors and data points
Minimizing of $\Phi(\xi)$ with respect to $w$ is a nonconvex optimization problem that is NP-complete.

To make the optimization problem mathematically tractable, function $\Phi(\xi)$ is approximated by writing

$$\Phi(\xi) = \sum_{i=1}^{N} \xi_i$$

Moreover, the computation can further be simplified by formulating the function to be minimized with respect to the weight vector $w$ as follows:

$$\Phi(w, \xi) = \frac{1}{2} w^T w + C \sum_{i=1}^{N} \xi_i \quad (8.2.20)$$

As before, minimizing the first term in Eq. 8.2.20 is related to minimizing the VC dimension of support vector machine. As for the second term $\sum_i \xi_i$, it is an upper bound on the number of test errors. Formulation of the cost function $\Phi(w, \xi)$ in Eq. 8.2.20 is therefore in perfect accord with principle of structural risk minimization.

The parameter $C$ controls tradeoff between complexity of the machine and the number of nonseparable point; it may therefore be viewed as a form of a "regularization" parameter. The parameter $C$ has to be selected by the user. This can be done in one of two ways

- The parameter $C$ determined experimentally via the standard use of training/validation test set, which is a crude form of resampling
- It is determined analytically by estimating the VC dimension via Eq. 8.2.9 and then by using bounds on generalization performance of the machine based on the VC dimension.

Now the primal problem for the nonseparable case can be stated as:
Given the training sample $\mathcal{S} = \{(\mathbf{x}_i, d_i)\}_{i=1}^{N}$, find the optimum values of the weight vector $\mathbf{w}$ and bias $b$ such that they satisfy the constraints

\[ d_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 1 - \xi_i \quad \text{for } i = 1, 2, 3 \ldots N \]
\[ \xi_i \geq 0 \quad \text{for all } i \]

and such that the weight vector $\mathbf{w}$ and slack variables $\xi_i$ minimizes the cost function:

\[ \Phi(\mathbf{w}, \xi) = \frac{1}{2} \mathbf{w}^T \mathbf{w} + C \sum_{i=1}^{N} \xi_i \]

where $C$ is the user-specified positive parameter.

Using the method of Lagrange multiplier and proceeding in a similar to that described in section 8.2 we may formulate the dual problem for nonseparable patterns as

Given the training sample $\mathcal{S} = \{(\mathbf{x}_i, d_i)\}_{i=1}^{N}$, find the Lagrange multipliers $\{\alpha_i\}_{i=1}^{N}$ that maximize the objective function

\[ Q(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j d_i d_j \mathbf{x}_i^T \mathbf{x}_j \]

Subject to the constraints:

\[ (1) \sum_{i=1}^{N} \alpha_i d_i = 0 \]
\[ (2) 0 \leq \alpha_i \leq C \quad \text{for } i = 1, 2, \ldots, N \]

where $C$ is a user-specified positive parameter.

Neither the slack variables $\xi_i$ nor their Lagrange multipliers appear in the dual problem. The dual problem for the case of nonseparable patterns is thus similar to that for the simple case of linearly separable patterns except for minor but important difference. The objective function $Q(\alpha)$ to be maximized is the same in both cases. The nonseparable case differs from the separable case in that the constraint $\alpha_i \geq 0$ is replaced with the more stringent constraint.
Except for this modification, the constrained optimization for the nonseparable case and computations of the optimum values of the weight vector \( \mathbf{w} \) and bias \( b \) proceed in the same way as in exactly the same way as in the linearly separable case. Note also that the support vectors are defined in exactly the same way as before.

The optimum solution for the weight vector \( \mathbf{w} \) is given by

\[
\mathbf{w}_o = \sum_{i=1}^{N_s} \alpha_{o,i} d_i \mathbf{x}_i
\]  

(8.2.21)

where \( N_s \) is the number of support vectors. The determination of the optimum values of the bias also follows a procedure similar to that described before. Specifically, the Kuhn-Tucker now be defined by

\[
\alpha_i \left[ d_i (\mathbf{w}^T \mathbf{x}_i + b) - 1 + \xi_i \right] = 0, \quad i = 1, 2, \ldots N
\]  

(8.2.22)

and

\[
\mu_i \xi_i = 0, \quad i = 1, 2 \ldots, N
\]  

(8.2.23)

where \( \mu_i \) are Lagrange multipliers that have been introduced to enforce the non negativity of the slack variables \( \xi_i \) for all \( i \). At the saddle point the derivative of the Lagrangian function for the primal problem with respect to the slack variable \( \xi_i \) is zero, the evaluation of which yields

\[
\alpha_i + \xi_i = C
\]  

(8.2.24)

By combining Eqs. 8.2.23 and 8.2.24, we see that

\[
\xi_i = 0, \quad \text{if} \quad \alpha_i < C
\]  

(8.2.25)

We may determine the optimum bias \( b_0 \) by taking any data point \((\mathbf{x}_i, d_i)\) in the training set for which we have \( 0 < \alpha_{0,i} < C \) and therefore \( \xi_i = 0 \), and using that data point in Eq. 8.2.22. However, from a numerical perspective it is better to take the mean value of \( b_0 \) resulting from all such data points in the training sample [Burges 1998].
8.2.3 Building SVM for pattern recognition

In the previous section we discussed how to find the optimal hyperplane for non separable face patterns at hand. In this section we formally describe the construction of a support vector machine for face image pattern recognition task.

Basically, the design of support vector machine for pattern recognition requires the following two operations

1. Non-linear mapping of an input vector into high dimensional feature space that is hidden from both the input and output.

2. Construction of an optimal hyperplane for separating the features discovered in step 1.

The rationale for each of these two operations is; operation 1 is performed in accordance with Cover’s theorem on the separability of patterns which states that a multidimensional space may be transformed into a new feature space where the patterns are linearly separable with high probability, provided two conditions are satisfied. First, the transformation is linear. Second, the dimensionality of the feature space is high enough. Operation 2 exploits the idea of building an optimal separating hyperplane in accordance with the theory explained in the previous section, but with a fundamental difference; the separating hyperplane is now defined as a linear function of vectors drawn from feature space rather than the original input space. Most importantly the construction of this hyperplane is performed in accordance with the principle of structural minimization that is rooted in VC dimension theory. The construction hinges on the evaluation of an inner-product kernel discussed below.
8.2.4 Inner-product kernel

Let \( x \) denote a vector drawn from the input training space, assumed to be of dimension \( m_0 \). Let \( \{ \varphi_j(x) \}_{j=1}^{m_1} \) denote a set of nonlinear transformations from the input space to the feature space, where \( m_1 \) is the dimension of the feature space. It is assumed that \( \varphi_j(x) \) is defined \emph{a priori} for all \( j \). Given such a set of nonlinear transformations, we can define a hyperplane acting as the decision surface as follows:

\[
\sum_{j=1}^{m_1} w_j \varphi_j(x) + b = 0 \tag{8.2.26}
\]

where \( \{w_j\}_{j=1}^{m_1} \) denotes a set of linear weights connecting the feature space to the output space, and \( b \) is the bias. For simplicity it can be written as

\[
\sum_{j=0}^{m_1} w_j \varphi_j(x) = 0 \tag{8.2.27}
\]

where it is assumed that \( \varphi_0(x) = 1 \) for all \( x \), so that \( w_0 \) denotes the bias \( b \).

Equation 8.2.27 defines the decision surface computed in the feature space in terms of the linear weights of the machine. The quantity \( \varphi_j(x) \) represents the input supplied to the weight \( w_j \) via the feature space. Define the vector

\[
\varphi(x) = [\varphi_0(x), \varphi_1(x), ..., \varphi_{m_1}(x)]^T \tag{8.2.28}
\]

where, by definition, we have

\[
\varphi(x) = 1 \quad \text{for all} \ x \tag{8.2.29}
\]

In effect the, the vector \( \varphi(x) \) represents the "image" induced in the feature space due to the input vector \( x \), as illustrated in Fig. 8.4. Thus, in terms of this image we may define the decision surface in the compact form:

\[
w_\varphi^T \varphi(x) = 0 \tag{8.2.30}
\]
Adapting Eq. 8.2.12 to our present situation involving a feature space where we now seek linear separability features, we may write

$$w_\varphi = \sum_{i=1}^{N} \alpha_i d_i \varphi(x_i) \quad (8.2.31)$$

where the feature vector $\varphi(x_i)$ corresponds to the test pattern $x_i$ in the $i$th example. Therefore, substituting Eq. 8.2.31 in 8.2.30, the decision surface computed in the feature space may be defined as:

$$\sum_{i=1}^{N} \alpha_i d_i \varphi^T(x_i) \varphi(x) = 0 \quad (8.2.32)$$

The term $\varphi^T(x_i) \varphi(x)$ represents the inner product of two vectors induced in the feature space by the input vector $x$ and the test pattern $x_i$ pertaining to the example. We may therefore introduce the inner-kernel-product denoted by $K(x, x_i)$ and defined by

$$K(x, x_i) = \varphi^T(x) \varphi(x_i) = \sum_{j=1}^{m_1} \varphi_j(x) \varphi_j(x_i) \quad \text{for } i = 1, 2, \ldots, N \quad (8.2.33)$$

From this definition we see that the inner-product kernel is symmetric function
of its arguments, shown by

\[ K(x, x_i) = K(x_i, x) \quad \text{for all } i \]  

(8.2.34)

We can use the inner product kernel \( K(x, x_i) \) to construct the optimal hyperplane in the feature space without having to consider the feature space itself in explicit form. This is readily seen by using equation Eq. 8.2.32 in 8.2.33, whereby the optimal hyperplane is now defined by [Haykin 2001], [Cristianini & Shawe-Taylor 2000].

\[ \sum_{i=1}^{N} \alpha_i d_i K(x, x_i) = 0 \]  

(8.2.35)

8.3 Multi-class classification using SVM

The problem of multi-class classification using SVMs doesn’t have an easy solution. It is generally simpler to construct classifier theory and algorithms for two mutually-exclusive classes than for \( N \) mutually-exclusive classes. Literatures reveal that constructing \( N \)-class SVMs is still an unsolved research problem [Platt et al. 2000].

The standard method for \( N \)-class SVMs [Vapnik 1998] is to construct \( N \)-SVMs. The \( i \)th SVM will be trained with all of the examples in the \( i \)th class with positive labels, and all other examples with negative labels. SVMs trained in this way are referred as \( 1 - v - r \) SVMs (one-versus-rest). The final output of the \( N \) \( 1 - v - r \) SVMs is the class that corresponds to the SVM with the highest output value. Unfortunately, there is no bound on the generalization error for the \( 1 - v - r \) SVM, and the training time of the standard method scales linearly with \( N \).

Another method for constructing \( N \)-class classifiers from SVMs by combining two-class classifiers was proposed by Knerr et al [Knerr et al. 1990]. Knerr
suggested constructing all possible two-class classifiers from a training set of \( N \) classes, each classifier being trained on only two out of \( N \) classes. There would thus be \( K = N(N - 1)/2 \) classifiers. This is referred to as 1-v-1 SVMs (one-versus-one). Knerr suggested combining these two-class classifiers with an AND gate [Knerr et al. 1990]. Friedman [Friedman & Kandel 1999] suggested a Max Wins algorithm: each 1-v-1 classifier casts one vote for its preferred class, and the final result is the class with the most votes. Friedman shows circumstances in which this algorithm is Bayes optimal. KreBel [KreBel 1999] applies the Max Wins algorithm to Support Vector Machines with excellent results.

A significant disadvantage of the 1-v-1 approach, however, is that, unless the individual classifiers are carefully regularized, the overall \( N \)-class classifier system will tend to overfit. The "AND" combination method and the Max Wins combination method do not have bounds on the generalization error. Finally, the size of the 1-v-1 classifier may grow super linearly with \( N \), and hence, may be slow to evaluate on large problems. To overcome these shortcomings Platt suggested \textit{Directed Acyclic Graph SVM} (DAGSVM) algorithm based on \textit{Decision Directed Acyclic Graphs} (DDAG) for multiclass classification problem as explained in the following section [Platt et al. 2000].

### 8.4 The DAGSVM algorithm

Given a space \( X \) and a set of boolean functions \( \mathcal{F} = \{ f : X \to \{0, 1\} \} \) the class \( \text{DDAG}(\mathcal{F}) \) of Decision DAGs on \( N \) classes over \( \mathcal{F} \) are functions which can be implemented using a rooted binary DAG with \( N \) leaves labeled by the classes where each of the \( N(N - 1)/2 \) internal nodes is labeled with an element of \( \mathcal{F} \). The nodes are arranged in a triangle with the single root node at the top, two nodes in the second layer and so on until the final layer of \( N \) leaves.
The $i_{th}$ node in layer $j < N$ is connected to the $i^{th}$ and $(i+1)^{th}$ node in the $(j+1)^{th}$ layer.

To evaluate a particular DDAG $G$ on input $x \in X$, starting at the root node, the binary function at a node is evaluated. The node is then exited via the left edge, if the binary function is zero; or the right edge, if the binary function is one. The next node’s binary function is then evaluated. The value of the decision function $D(x)$ is the value associated with the final leaf node (see Figure 8.5). The path taken through the DDAG is known as the evaluation path. The input $x_i$ reaches a node of the graph, if that node is on the evaluation path for $x$. It is referred that the decision node distinguishing classes $i$ and $j$ as the $ij$-node. Assuming that the number of a leaf is its class, this node is the $i^{th}$ node in the $(N-j+i)^{th}$ layer provided $i < j$. Similarly the $j$-nodes are those nodes involving class $j$, that is, the internal nodes on the two diagonals containing the leaf labeled by $j$.

![Figure 8.5](image-url): The decision DAG for finding the best class out of four classes. The equivalent list state for each node is shown next to that node.
The DDAG is equivalent to operating on a list, where each node eliminates one class from the list. The list is initialized with a list of all classes. A test point is evaluated against the decision node that corresponds to the first and last elements of the list. If the node prefers one of the two classes, the other class is eliminated from the list, and the DDAG proceeds to test the first and last elements of the new list. The DDAG terminates when only one class remains in the list. Thus, for a problem with $N$ classes, $(N - 1)$ decision nodes will be evaluated in order to derive an answer. Figure 8.5 shows the decision DAG for finding the best class out of four classes. The equivalent list state for each node is shown next to that node. Figure 8.6 shows a diagram of the input space of a four-class problem. A $1-v-1$ SVM can only exclude one class from consideration. The current state of the list is the total state of

\begin{figure}
\centering
\includegraphics[width=\textwidth]{decision_dag}
\caption{The decision DAG for finding the best class out of four classes. The equivalent list state for each node is shown next to that node.}
\end{figure}

the system. Therefore, since a list state is reachable in more than one possible path through the system, the decision graph the algorithm traverses is a DAG,
not simply a tree.

Decision DAGs naturally generalize the class of Decision Trees, allowing for a more efficient representation of redundancies and repetitions that can occur in different branches of the tree, by allowing the merging of different decision paths. The class of functions implemented is the same as that of Generalized Decision Trees [Bennett et al. 2000], but this particular representation presents both computational and learning-theoretical advantages [Knerr et al. 1990], [Platt et al. 2000].

In this type of DDAGs node-classifiers are considered as hyperplanes. We define a Perceptron DDAG to be a DDAG with a perceptron at every node. Let \( w \) be the (unit) weight vector correctly splitting the \( i \) and \( j \) classes at the \( ij \)-node with threshold \( \theta \). We define the margin of the \( ij \)-node to be \( c(x) \) is the class associated to training example \( x \).

Maximizing the margin of all of the nodes in a DDAG will minimize a bound on the generalization error [Platt et al. 2000]. This bound is also independent of input dimensionality. Therefore, we can create a DDAG whose nodes are maximum margin classifiers over a kernel-induced feature space. Such a DDAG is obtained by training each \( ij \)-node only on the subset of training points labeled by \( i \) or \( j \). The final class decision is derived by using the DDAG architecture, illustrated in figure 8.5. The DAGSVM separates the individual classes with large margin. It is safe to discard the losing class at each 1-v-1 decision because, for the hard margin case, all of the examples of the losing class are far away from the decision surface (see Figure 8.6).

The DAGSVM algorithm is superior to other multiclass SVM algorithms in both training and evaluation time. Empirically, SVM training is observed to scale super-linearly with the training set size \( m \) [Platt et al. 2000], according to a power law: where for algorithms based on the decomposition method
with some proportionality constant $c$. For the standard 1-v-r multiclass SVM training algorithm, the entire training set is used to create all $N$ classifiers. The training time for 1-v-r will be

$$T_{1-v-r} = c N m^\gamma$$  \hspace{1cm} (8.4.1)

Assuming that the classes have the same number of examples, training each 1-v-1 SVM only requires $2m/N$ training examples. Thus, training $K$ 1-v-1 SVMs require

$$T_{1-v-r} = c \frac{N(N-1)}{2} \left(\frac{2m^\gamma}{N}\right) \approx 2^{\gamma-1} c N^{2-\gamma} m^\gamma$$  \hspace{1cm} (8.4.2)

For a typical case, where the amount of time required to train all of the 1-v-1 SVMs is independent of $N$, and is only twice that of training a single 1-v-r SVM. Using 1-v-1 SVMs with a combination algorithm is thus preferred for training time.

### 8.5 Simulation experiment and results

Simulation experiment were conducted to implement the DAGSVM algorithm using MATLAB. In chapter 7, it has been experimentally shown that SSPD parameter gives better recognition results in all the four face recognition algorithms used, viz., c-Means clustering technique, k-NN Classifier, Bayes Classifier and Artificial Neural Network. Therefore the credibility of the DAGSVM based face recognition system is verified using the SSPD parameters. The face images of the KNUFDB face database as well as AT&T face database are used in the simulation study. The recognition accuracies based on the SSPD feature using DAGSVM technique are given in Table 8.1. Graphical representation of these recognition results based on SSPD feature using DAGSVM architecture is shown in figure 8.7.
8. Support Vector Machines for Face Recognition

Table 8.1: Classification results using SSPD feature vector and DAGSVM algorithm on KNUFDB and AT&T face databases

<table>
<thead>
<tr>
<th>Person ID</th>
<th>No. samples classified correctly</th>
<th>Accuracy (%) KNUFDB</th>
<th>No. samples classified correctly</th>
<th>Accuracy (%) AT&amp;T</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>18</td>
<td>60.00</td>
<td>3</td>
<td>60.00</td>
</tr>
<tr>
<td>2</td>
<td>19</td>
<td>63.33</td>
<td>4</td>
<td>80.00</td>
</tr>
<tr>
<td>3</td>
<td>20</td>
<td>66.67</td>
<td>5</td>
<td>100.00</td>
</tr>
<tr>
<td>4</td>
<td>19</td>
<td>63.33</td>
<td>5</td>
<td>100.00</td>
</tr>
<tr>
<td>5</td>
<td>20</td>
<td>66.67</td>
<td>5</td>
<td>100.00</td>
</tr>
<tr>
<td>6</td>
<td>23</td>
<td>76.67</td>
<td>5</td>
<td>100.00</td>
</tr>
<tr>
<td>7</td>
<td>25</td>
<td>83.33</td>
<td>5</td>
<td>100.00</td>
</tr>
<tr>
<td>8</td>
<td>22</td>
<td>73.33</td>
<td>5</td>
<td>100.00</td>
</tr>
<tr>
<td>9</td>
<td>22</td>
<td>73.33</td>
<td>5</td>
<td>100.00</td>
</tr>
<tr>
<td>10</td>
<td>24</td>
<td>80.00</td>
<td>5</td>
<td>100.00</td>
</tr>
<tr>
<td>11</td>
<td>26</td>
<td>86.67</td>
<td>5</td>
<td>100.00</td>
</tr>
<tr>
<td>12</td>
<td>24</td>
<td>80.00</td>
<td>5</td>
<td>100.00</td>
</tr>
<tr>
<td>13</td>
<td>24</td>
<td>80.00</td>
<td>5</td>
<td>100.00</td>
</tr>
<tr>
<td>14</td>
<td>25</td>
<td>83.33</td>
<td>5</td>
<td>100.00</td>
</tr>
<tr>
<td>15</td>
<td>24</td>
<td>80.00</td>
<td>5</td>
<td>100.00</td>
</tr>
<tr>
<td>16</td>
<td>26</td>
<td>86.67</td>
<td>5</td>
<td>100.00</td>
</tr>
<tr>
<td>17</td>
<td>22</td>
<td>73.33</td>
<td>5</td>
<td>100.00</td>
</tr>
<tr>
<td>18</td>
<td>24</td>
<td>80.00</td>
<td>5</td>
<td>100.00</td>
</tr>
<tr>
<td>19</td>
<td>26</td>
<td>86.67</td>
<td>5</td>
<td>100.00</td>
</tr>
<tr>
<td>20</td>
<td>25</td>
<td>83.33</td>
<td>5</td>
<td>100.00</td>
</tr>
<tr>
<td>21</td>
<td>25</td>
<td>83.33</td>
<td>5</td>
<td>100.00</td>
</tr>
<tr>
<td>22</td>
<td>24</td>
<td>80.00</td>
<td>5</td>
<td>100.00</td>
</tr>
<tr>
<td>23</td>
<td>25</td>
<td>83.33</td>
<td>5</td>
<td>100.00</td>
</tr>
<tr>
<td>24</td>
<td>24</td>
<td>80.00</td>
<td>5</td>
<td>100.00</td>
</tr>
<tr>
<td>25</td>
<td>25</td>
<td>83.33</td>
<td>5</td>
<td>100.00</td>
</tr>
<tr>
<td>26</td>
<td>26</td>
<td>86.67</td>
<td>5</td>
<td>100.00</td>
</tr>
<tr>
<td>27</td>
<td>25</td>
<td>83.33</td>
<td>5</td>
<td>100.00</td>
</tr>
<tr>
<td>28</td>
<td>26</td>
<td>86.67</td>
<td>5</td>
<td>100.00</td>
</tr>
<tr>
<td>29</td>
<td>24</td>
<td>80.00</td>
<td>5</td>
<td>100.00</td>
</tr>
<tr>
<td>30</td>
<td>25</td>
<td>83.33</td>
<td>5</td>
<td>100.00</td>
</tr>
<tr>
<td>31</td>
<td>27</td>
<td>90.00</td>
<td>5</td>
<td>100.00</td>
</tr>
<tr>
<td>32</td>
<td>27</td>
<td>90.00</td>
<td>5</td>
<td>100.00</td>
</tr>
<tr>
<td>33</td>
<td>26</td>
<td>86.67</td>
<td>5</td>
<td>100.00</td>
</tr>
<tr>
<td>34</td>
<td>26</td>
<td>86.67</td>
<td>5</td>
<td>100.00</td>
</tr>
<tr>
<td>35</td>
<td>26</td>
<td>86.67</td>
<td>5</td>
<td>100.00</td>
</tr>
<tr>
<td>36</td>
<td>25</td>
<td>83.33</td>
<td>5</td>
<td>100.00</td>
</tr>
<tr>
<td>37</td>
<td>27</td>
<td>90.00</td>
<td>5</td>
<td>100.00</td>
</tr>
<tr>
<td>38</td>
<td>26</td>
<td>86.67</td>
<td>5</td>
<td>100.00</td>
</tr>
<tr>
<td>39</td>
<td>26</td>
<td>86.67</td>
<td>5</td>
<td>100.00</td>
</tr>
<tr>
<td>40</td>
<td>25</td>
<td>83.33</td>
<td>5</td>
<td>100.00</td>
</tr>
<tr>
<td>41</td>
<td>24</td>
<td>80.00</td>
<td>5</td>
<td>100.00</td>
</tr>
<tr>
<td>42</td>
<td>26</td>
<td>86.67</td>
<td>5</td>
<td>100.00</td>
</tr>
<tr>
<td>43</td>
<td>25</td>
<td>83.33</td>
<td>5</td>
<td>100.00</td>
</tr>
<tr>
<td>44</td>
<td>26</td>
<td>86.67</td>
<td>5</td>
<td>100.00</td>
</tr>
<tr>
<td>45</td>
<td>26</td>
<td>86.67</td>
<td>5</td>
<td>100.00</td>
</tr>
<tr>
<td>46</td>
<td>24</td>
<td>80.00</td>
<td>5</td>
<td>100.00</td>
</tr>
</tbody>
</table>

continued on next page
8. Support Vector Machines for Face Recognition

Table 8.1 – continued from previous page

<table>
<thead>
<tr>
<th>Person ID</th>
<th>No. samples classified correctly</th>
<th>Accuracy (%) KNUFDB</th>
<th>No. samples classified correctly</th>
<th>Accuracy (%) AT&amp;T</th>
</tr>
</thead>
<tbody>
<tr>
<td>47</td>
<td>27</td>
<td>90.00</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>48</td>
<td>25</td>
<td>83.33</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>49</td>
<td>26</td>
<td>86.67</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>50</td>
<td>27</td>
<td>90.00</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>51</td>
<td>25</td>
<td>83.33</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>52</td>
<td>25</td>
<td>83.33</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>53</td>
<td>27</td>
<td>90.00</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>54</td>
<td>23</td>
<td>76.67</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>55</td>
<td>26</td>
<td>86.67</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>56</td>
<td>26</td>
<td>86.67</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>57</td>
<td>27</td>
<td>90.00</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>58</td>
<td>27</td>
<td>90.00</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>59</td>
<td>26</td>
<td>86.67</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>60</td>
<td>27</td>
<td>90.00</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Overall Recognition 82.39% 98.50%

Figure 8.7: Recognition accuracies for DAGSVM classifier using SSPD feature vectors

The overall recognition accuracy obtained is 82.39% for KNUFDB. For AT&T face image database the recognition accuracy obtained using DAGSVM
was comparable to that of ANN classifier. However, there is a notable improvement in the recognition accuracy for KNUFDB database. This gives an inference that DAGSVM classifier is suitable for face image data acquired under most natural conditions which may contain spurious noise, illumination variations and pose variations.

8.6 Conclusion

A face recognition system using DAGSVM is designed and developed. SSPD features extracted from human face images are used as the feature parameter. The DAGSVM algorithm is superior to other multiclass SVM algorithms in both training and evaluation time. The entire system is implemented using MATLAB codes. Usage of SSPD for face recognition in this work is the first of its kind in face recognition. The overall recognition accuracy obtained for face recognition is 82.39% for KNUFDB face database and 98.50% for AT&T face database respectively. This is highly promising compared to the recognition results obtained in various experiments conducted in this study using different parameters and classifiers.