CLASSIFICATION WITH SUPPORT VECTOR MACHINES
CHAPTER 5        CLASSIFICATION WITH SUPPORT VECTOR MACHINES

5.1 Introduction

Being able to identify the minor variations in similar characters is the challenge underlying character recognition of alphabet of Indian languages. Of several existing methods, neural network models have been reported to achieve better performance in comparison, as mentioned in the previous chapter. But, just like any good thing neural networks have their own share of problems, some of them being the problem of multiple local minima and the dependence of the computational complexity on dimensionality of the input space. This is where support vector machine (SVM) enters into the scene. While the artificial neural network suffers from multiple local minima, the solution to SVM is global (Burges, 1998). And, unlike neural networks SVM does not depend on the dimensionality of the input space and works very well with high dimensional data (Tan et al., 2007). Other advantages of SVM include simple geometric interpretation and the insensitivity to the relative number of training examples. SVM has been observed to achieve reasonable generalization accuracy especially in implementing handwritten digit recognition and character recognition (Bin et al., 2000). This chapter mainly discusses the design of the classifier with support vector machines.

5.2 Support Vector Machines (SVM)

The SVM is a classification and regression prediction tool that uses machine learning theory to maximize the predictive accuracy while avoiding over fit to data. This classification technique has the advantage of being able to solve supervised classification in high dimension and it also does not suffer from the limitations of data dimensionality and limited samples. The unique aspect of SVM is that, it constructs the decision boundary using a subset of training examples known as support vectors. To understand the basic essence of SVM, the concept of the hyper plane or more precisely the maximal margin hyper plane has to be understood. SVM, which is basically a binary classifier, separates the P dimensional data points into two classes by constructing a P-1 dimensional hyper plane. As long as the two classes are linearly separable, looking for a hyperplane with the largest margin of separation is easy and the classifier is known as linear classifier. The principle of linear classifier can also be
extended to non-linear classifier by employing a penalty term for misclassification or by using a kernel function for projecting the original feature space to high dimensional space which facilitates classification with linear decision surface. In the subsequent parts of the chapter, the above mentioned 3 cases are outlined under the headings

- Linear SVM for separable Case
- Linear SVM for non-separable Case
- Non-Linear SVM

5.2.1 Linear SVM for Separable Case

The goal of a linear SVM is to separate the two classes by a function which is induced from the available examples and build a classifier that will work well on unseen examples. Given a set of data points in a P dimensional space, the linear classifier separates the data with a P-1 dimensional hyperplane. There are many hyperplanes that might classify the data as shown in Figure 5.1. But the interest here is to pick a hyper plane so that the distance from the hyper plane to the nearest data point is maximized as shown in Figure 5.2. Linear SVM searches for hyper plane with the largest margin and is therefore referred as maximal margin classifier.

![Figure 5.1: With Number of Hyper Planes](image)
To understand how SVM learns such a boundary, we consider a binary classification problem consisting of $n$ training examples where each example is denoted by a tuple $(x_i, y_i)$ for $i=1,2,…,n$ where $x_i=(x_{i1}, x_{i2},…, x_{ip})^T$ corresponds to the attribute set for the $i^{th}$ example and $y_i \in \{-1,1\}$ denotes the class label assuming that the data is linearly separable. The decision boundary or the hyper plane of the linear classifier can be described by the following equation

$$W x + b = 0 \hspace{2cm} 5.1$$

Where $W$ is normal to the hyper plane and $\frac{-b}{\|w\|}$ is the perpendicular distance of hyper plane from the origin. The aim of the support vector machine is to orientate this hyper plane in such a way as to be as far as possible from the closest members of both the classes. Training data referred in Figure 5.3 can be described by selecting the variables $W$ and $b$ such that
Figure 5.3: Hyper Plane through Two Linearly Separable Classes

\[ X_i W + b \geq 1 \quad \text{for } Y_i = +1 \]  \hspace{1cm} 5.2

\[ X_i W + b \leq -1 \quad \text{for } Y_i = -1 \]  \hspace{1cm} 5.3

The equations 5.2 and 5.3 can be combined into a single equation 5.4

\[ Y_i (X_i W + b) - 1 \geq 0 \quad \forall \ i \] \hspace{1cm} 5.4

To identify the maximum margin hyper plane, consider the points that lie closest to the hyper plane i.e., the support vectors (shown with enclosed circles) and the planes \( H_1 \) and \( H_2 \) passing through the points and parallel to original hyper plane are described by the equations

\[ X_1 W + b = 1 \quad \text{for } H_1 \] \hspace{1cm} 5.5

\[ X_2 W + b = -1 \quad \text{for } H_2 \] \hspace{1cm} 5.6

The margin of the hyper plane \( Wx+b=0 \) is defined as the sum of the shortest distance from \( H_1 \) and shortest distance from \( H_2 \). The margin is simply \( \frac{2}{||w||} \), where \( ||w|| \) is the Euclidian norm of \( w \) and the maximum margin is obtained by minimizing \( ||w|| \) which is equivalent to minimizing \( \frac{1}{2} ||w||^2 \) and the use of this term makes it possible to perform quadratic programming (Fletcher, 2009).
The learning task of SVM can be formalized as constrained optimization problem to minimize

\[ \frac{1}{2} ||w||^2 \ \text{subjected to} \ Y_i (X_i \cdot W + b) - 1 \geq 0 \ \forall \ i \ \text{...............} \]

5.7

The function to be minimized is quadratic and the constraints are linear in parameters $W$ and $b$. The optimization is a convex optimization problem which can be solved by using the standard Lagrange multiplier method. The new objective function which takes into account the new constraints is known as Lagrangian for the optimization and can be written as

\[ L_p = \frac{1}{2} ||w||^2 - \sum_{i=1}^{N} \lambda_i (Y_i (X_i \cdot W + b) - 1) \]

5.8

Where $\lambda_i$ are called the Lagrange multipliers. To minimize the Lagrangian, derivative of $L_p$ with respect to $W$ and $b$ are set to zero.

\[ \frac{\partial L_p}{\partial W} = W - \sum \lambda_i Y_i X_i \]

\[ \Rightarrow \ W = \sum \lambda_i Y_i X_i \ \text{............................................} \]

5.9

\[ \frac{\partial L_p}{\partial b} = \sum \lambda_i Y_i \ \text{............................................} \]

5.10

As the Lagrange multipliers are unknown, the equation cannot be solved for $W$ and $b$. If the constraints in $L_p$ contains equality constraints instead of inequality, then feasible solutions for $W$, $b$ and $\lambda_i$ can be found. Such transformation leads to the following constraints on Lagrange multipliers which are known as Karush-Kuhn-Tucker(KKT) conditions and provide the sufficient and necessary conditions for optimization problem (Mavroforakis & Theodoridis, 2006).

\[ \lambda_i \geq 0 \ \text{..........................................................} \]

5.11
Many of the Lagrange multipliers become zero after applying the constraint in equation 5.12. This constraint states that the Lagrange multiplier must be zero unless the training instance \( X_i \) satisfies the equation \( Y_i (W \cdot X_i + b) = 0 \). The training instance with \( \lambda_i > 0 \) that lies along the hyper planes \( H_1 \) or \( H_2 \) is known as support vector. Training instances that do not reside along the hyper planes have \( \lambda_i = 0 \).

The problem of optimization of \( L_p \) can be simplified by transforming the Lagrangian into a function of the Lagrange multipliers by substituting 5.9 and 5.10 in 5.8 which leads to the dual formulation of the optimization problem

\[
L_D = \sum_{i=1}^{N} \lambda_i - \sum_{i,j} \lambda_i \lambda_j Y_i X_i \cdot X_j
\]

5.13

The differences between the dual and primary Lagrangians are,

- The dual Lagrangian involves only the Lagrange multipliers and the training data.
- The quadratic term in Equation 5.13 has negative sign, which means that original minimization problem involving the primary Lagrangian \( L_p \) has turned into a maximization problem involving the dual Lagrangian \( L_D \).

For large data sets the dual optimization problem can be solved by Quadratic programming. Once \( \lambda_i \) are found, the solution for \( W \) and \( b \) can be found by using equations 5.9 and 5.12. The decision boundary can be expressed as

\[
\sum_{i=1}^{N} \lambda_i Y_i X_i \cdot X + b = 0
\]

5.14

Where \( b \) is obtained by solving equation 5.12 for the support vectors. Because the value of \( \lambda_i \) are calculated numerically and have numerical errors, the value computed for \( b \) may not be unique. It depends on the support vectors used in equation 5.12. In practice the average value for \( b \) is chosen to be the parameters of decision boundary.
5.2.2 Linear SVM: Non-Separable Case

In the non-separable cases where the classes overlap, the constraints in equation 5.4 cannot be satisfied for all the data points or in other words it is not possible to construct a separating hyper plane without encountering classification errors. This section addresses how the formulation in the section 5.2.1 can be modified to learn a decision boundary that is tolerable to small training errors using a method known as soft margin approach. The margin of separation between classes is said to be soft if a data point \((X_i, Y_i)\) violates the condition in 5.4 (H"aykin, 2006).

In order to extend the SVM methodology to handle data that is not fully linearly separable the equations 5.2 and 5.3 are slightly modified to allow for misclassified points. This is done by introducing a positive slack variable \(\xi_i\) in equation 5.2 and 5.3

\[
X_iW + B \geq +1 - \xi_i \quad \text{for } Y_i = +1 \quad \ldots \quad 5.15
\]

\[
X_iW + B \leq -1 + \xi_i \quad \text{for } Y_i = -1 \quad \ldots \quad 5.16
\]

Where \(\xi_i \geq 0\) \(\forall i\), which can be combined into

\[
Y_i(X_iW + b) - 1 + \xi_i \geq 0 \quad \text{where } \xi_i \geq 0 \quad \forall i \quad \ldots \quad 5.17
\]

To understand the meaning of the slack variable \(\xi_i\), consider the diagram shown in Figure 5.4.
Figure 5.4: Linear SVM for Non-separable Case

The circle P is one of the instances that violates the constraints in equation 5.4. The distance between this point and the hyper plane H2 is $\frac{\xi}{||W||}$. Thus $\xi$ provides an estimate of the decision boundary on the training examples P. $\sum \xi$ provides an upper bound on the classification errors which can be included in the objective function as an extra cost which is required to be minimized. The optimization problem in equation 5.7 can be modified with this extra term and the optimization problem can be reformulated as

$$\text{Minimize} \quad \frac{1}{2} ||w||^2 + c \sum_{i=1}^{N} \xi_i \quad \text{subjected to} \quad Y_i (X_i^TW + b) - 1 + \xi_i \geq 0 \quad \forall i \quad 5.18$$

Where the parameter c controls the tradeoff between the slack variable penalty and the size of the margin. To solve the reformulated optimization problem, Lagrangian is first constructed with primal variables as

$$L_p = \frac{1}{2} ||w||^2 + c \sum_{i=1}^{N} \xi_i - \sum_{i=1}^{N} \lambda_i \left(Y_i (X_i^TW + b) - 1 + \xi_i \right) - \sum_{i=1}^{N} \mu_i \xi_i \quad 5.19$$
Where the first two terms are the objective function to be minimized, the third term represents the inequality constraint associated with slack variables and the last term is the result of the non negativity requirements on the values of slack variables. The inequality constraints can be transformed into equality constraints using the KKT conditions given below

\[ \xi_i \geq 0, \lambda_i \geq 0, \mu_i \geq 0 \] \hspace{1cm} 5.20

\[ \lambda_i \left( Y_i (X_i W + b) - 1 + \xi_i \right) = 0 \] \hspace{1cm} 5.21

\[ \mu_i \xi_i = 0 \] \hspace{1cm} 5.22

Differentiating with respect to \( w, b, \xi_i \) and setting the first derivatives to zero

\[ \frac{\partial L_p}{\partial W} = 0 \Rightarrow W = \sum_{i=1}^{N} \lambda_i Y_i X_i \] \hspace{1cm} 5.23

\[ \frac{\partial L_p}{\partial b} = 0 \Rightarrow \sum_{i=1}^{N} \lambda_i Y_i = 0 \] \hspace{1cm} 5.24

\[ \frac{\partial L_p}{\partial \xi_i} = 0 \Rightarrow \lambda_i + \mu_i = c \] \hspace{1cm} 5.25

Substituting equations 5.23, 5.24, 5.25 into the Lagrangian of equation 5.19 will produce the following dual Lagrangian

\[ L_D = \frac{1}{2} \sum_{i,j} \lambda_i \lambda_j Y_i Y_j X_i X_j + c \sum \xi_i - \sum \lambda_i \left( \sum \lambda_j Y_j X_j X_j + b \right) - 1 + \xi_i \right) - \sum (c - \lambda_i) \xi_i \]

\[ = \sum_{i=1}^{N} \lambda_i - \frac{1}{2} \sum_{i,j} \lambda_i \lambda_j Y_i Y_j X_i X_j \] \hspace{1cm} 5.26

Which turns out to be similar to the dual lagrangian for linearly separable data, but the constraints imposed on the Lagrange multipliers \( \lambda_i \) are slightly different than those in the linearly separable case. The Lagrange multipliers for non-linear separable data are restricted to \( 0 \leq \lambda_i \leq c \).
The dual problem can be solved similar to the method discussed in section 5.2.1 to obtain the parameters of the decision boundary.

### 5.2.3 Non-Linear SVM

This section addresses the methodology for applying SVM to data sets that have non-linear decision boundaries as shown in Figure 5.5. In 1992 Bosch et al., suggested a way to create a non-linear classifier by applying the kernel trick to construct maximum margin hyper plane. The resulting algorithm is formally similar, except that every dot product is replaced by non-linear kernel function. The trick here is to transform the data from its original co-ordinate space in $X$ into a new space $\phi(X)$ so that the linear decision boundary can be used to separate the instances in the transformed space.

![Figure 5.5: Nonlinearly Separable data](image)

The learning task for a non-linear SVM can be formalized as the following optimization problem:

$$\min \frac{||w||^2}{2} \text{ subject to } Y_i (w \cdot \phi(X_i) + b) \geq 1 \text{ for } i=1,2,3,\ldots,N \quad 5.27$$

The main difference between linear SVM and non-linear SVM is that instead of using the original attribute $X$, the learning task is performed on the transformed attribute $\phi(X)$. Following the same approach as in linear SVM for the dual Lagrangian, the constrained optimization may be formulated as
\[ L_D = \sum_{i=1}^{N} \lambda_i - \frac{1}{2} \sum_{i,j} \lambda_i \lambda_j Y_i Y_j \phi(X_i) \phi(X_j) \] ........................................ 5.28

\( \lambda \)'s are found using quadratic programming techniques, the parameters W and b can be derived using the following equations

\[ W = \sum \lambda_i Y_i \phi(X_i) \] ........................................ 5.29

\[ \lambda_i \left\{ Y_i \left( \sum \lambda_j Y_j \phi(X_j) \phi(X_i) + b \right) - 1 \right\} = 0 \] ........................................ 5.30

These two equations are similar to 5.9 and 5.10 in linear SVM. Finally the decision boundary can be expressed as

\[ \sum_{i=1}^{N} \lambda_i Y_i \phi(X_i) \phi(X_j) + b \] ........................................ 5.31

Equation 5.31 involves the calculation of dot product between the pairs of vectors in the transformed space \( \phi(X_i) \phi(X_j) \). Such computation may be quite cumbersome and may suffer from the curse of dimensionality. A breakthrough solution to this problem comes in the form of a method known as Kernel trick. This method enables the efficient computation of the inner product in the feature space.

**5.3 Kernel Trick**

The kernel trick is a method for using a linear classifier algorithm to solve a non-linear problem by mapping the original nonlinear observations into a high dimensional space. Subsequently the linear classifier is used to make a linear classification in the new space, which is equivalent to the non-linear classification in the original space. Here, the kernel trick is being used to compute the similarity in the transformed space with the original attribute set using the kernel function to represent the dot product. This results in the decision boundary which is represented as,

\[ \sum_{i=1}^{N} \lambda_i Y_i K(X_i, X_j) + b = 0 \] ........................................ 5.32

Where, \( K \) is the kernel function.
The kernel function used in SVM must satisfy the mathematical principle known as Mercer’s theorem (Tan et al., 2006). This principle ensures that kernel function can always be expressed as the dot product between the two input vectors on some high dimensional space. The transformed space of SVM kernel is called Reproducing Kernel Hilbert Space (RKHS). The kernel functions that satisfy the Mercer’s theorem are called positive definite kernel functions and commonly used such functions are

- Polynomial Kernel
- Radial Basis Function Kernel

### 5.3.1 Polynomial Kernel

Polynomial mapping is a popular method for non-linear modelling and is given by

\[
K\left(X_i, X\right) = (X_i X + 1)^d
\]

Where \( d \) is the degree of the polynomial and kernel function of order \( d \) can be used to transform them into linearly independent vectors on those \( d \) dimensions. The dimensionality of the feature space increases with the value of \( d \) dimensions. As a result the number of support vectors becomes large which increases the computational complexity and computation time. Among polynomial kernels with different degrees, the performance of the second and third degree polynomial classifiers is much better and number of support vectors is relatively small. (Cao et al., 2007)

### 5.3.2 Radial Basis Function Kernel

The RBF kernel is widely used because it offers good generalization, universal approximation properties and good performance in solving practical problems. The RBF kernel equation is described by

\[
K\left(X_i, X_j\right) = \exp\left(-\frac{||X_i - X_j||^2}{2\sigma^2}\right)
\]

The error can be minimized by choosing a larger margin which is the distance between hyper plane and the closest data points by changing width. The number of
RBFs and their centers are determined automatically by the number of support vectors and their values (Muller et al., 2001).

5.4 Design and Implementation

Support vector machines have gained immense popularity in recent years. SVMs have achieved excellent recognition results in various pattern recognition applications (Papageorgiou and Poggio, 2000). This section addresses the problem of classification of handwritten Telugu characters using support vector machines whose architecture is already presented in the previous section. SVM performs classification task by constructing hyperplanes in the feature space in case of linearly separable data. In case of non-linearly separable data the original feature space is projected to high dimensional feature space to facilitate the construction of linear decision boundary.

SVMs work very well with high dimensional data, so all the 41 features extracted from the images as described in Chapter 3 are used as attributes for classification. The members of the data set which we considered have similar structures and comprise a non-linearly separable data. So a non-linear mapping of an input vector into high dimensional feature space is performed before the construction of optimal hyper plane using kernel functions. In this work radial basis function kernel is used with a \( \sigma \) value of 4.5 to project the input features space to high dimensional feature space. The advantage of radial basis function over polynomial kernel is discussed in the previous section. The optimal separating hyper plane can be determined without any computations in the high dimensional feature space. The dual Lagrangian for binary classification in the non-linear SVM can be formulated as given in equation 5.28

\[
L_D = \sum_{i=1}^{N} \lambda_i - \frac{1}{2} \sum_{i,j} \lambda_i \lambda_j Y_i Y_j \phi(X_i) \phi(X_j)
\]

Where radial basis function kernel is used to map \( X \) to \( \phi(X) \). The advantage of choosing radial basis function over polynomial kernel is discussed in 5.3.2. The steps to solve the above dual Lagrangian can be explained with simple algorithm as shown in figure 5.6.
Step 1: Create Hessian matrix $H$ where $H_{ij} = Y_i Y_j \phi(X_i) \phi(X_j)$

Step 2: Choose how significantly misclassification should be treated, by selecting a suitable value for parameter $C$.

Step 3: Find $\lambda$ so that $\sum \lambda_i - \frac{1}{2} \sum H_{ij} \lambda_i \lambda_j$ is maximized subject to the constraints $0 \leq \lambda_i \leq c$ and $\sum_{i=1}^{N} \lambda_i Y_i = 0$

(This step is solved using Quadratic programming).

Step 4: Calculate $W = \sum_{i=1}^{L} \lambda_i Y_i \phi(X_i)$ Where $L$ is the number of Lagrangian multipliers

Step 5: Determine the set of support vectors $S$ by finding the indices such that $0 \leq \alpha_i \leq c$

Step 6: Calculate $b = \frac{1}{N} \sum_{s=1}^{S} Y_s - \sum_{m=1}^{S} \alpha_m Y_m \phi(X_m) \phi(X_s)$

Step 7: Classify new point $X'$ by evaluating $Y' = \text{sign}(W \phi(X') + b)$

Figure 5.6: Algorithm for Solving Dual Lagrangian

Although SVMs are originally designed as binary classifiers, as explained by the above algorithm, a variety of techniques for decomposition of the multiclass problem into several binary class problem using support vector machines have been used previously

5.5 Multi-Class Support Vector Machines Formulation

Since SVMs are originally designed for two class problems, when there are more than two classes in the problem, the problem is formulated differently. There are four types of SVMs that handle multi-class problem (Abe, 2005). They are

- One against all support vector machines
- Pair wise support vector machines
- Error correcting code (ECOC) support vector machines
- All at once support vector machine

In the first approach, one against all support vector machines, an N class problem is converted into N two class problems. In the second approach, a pair wise support vector machine converts N class problem to N (N-1)/2 two class problems covering all pairs of classes. In the third approach, error correcting codes which detect and
correct errors in the data transmission channels are used to encode classifier outputs to improve generalization ability. These codes are called error correcting codes. In the first three methods there is a probability for unclassifiable regions. In the fourth method unclassifiable regions are resolved by determining all the decision functions simultaneously. In this work one against all support vector machines is used and the unclassifiable regions are resolved by using decision tree based support vector machines.

5.5.1 One Against All Support Vector Machines

For an N class problem, the one against all support vector machines constructs N number of SVM models. The i\(^{th}\) SVM is trained with all of the training examples in the i\(^{th}\) class with positive labels and the others with negative labels. The proposed method constructs N hyper planes.

![Hyper Planes for One against All Support Vector Machines](image)

Figure 5.7: Hyper Planes for One against All Support Vector Machines

Figure 5.7 shows the four hyper planes constructed for a four class problem and the shaded portions represent the unclassified regions. In the recognition phase, a test example is presented to all N SVMs and is labeled according to maximum output among the N classifiers. The disadvantage of this method is its training complexity. When the number of training samples is large each of the N classifiers is trained using all available samples. In this work the data set considered is a multi-class problem.
with 10 classes. The 10 decision hyper planes are constructed using the algorithm given in the previous section. The SVM was tested with 600 images using 10-fold cross validation with 540 images as training set i.e., to create the decision boundaries and 60 images are tested in each fold. Out of 600 images, the results are as follows:

- Number of character unclassified 54
- Number of character misclassified 71
- Number of character correctly classified 475

The results of the one against all support vector machine is shown in Table 5.1. Where the number of correctly classified characters, percentage of characters classified and unclassified characters, false positives, and false negatives are given for each class. Because of the unclassified regions in the one against all support vector machines the error rate is high. The number of support vectors needed and the value of b for one fold are shown in Table 5.2. The accuracy of this method can be improved with small modification by constructing N-1 hyper planes or N-1 support vector machines by constructing decision tree based support vector machines.

**Table 5.1: Results of One against All Support Vector Machines**

<table>
<thead>
<tr>
<th>Class Label</th>
<th>Number of Correctly Classified Characters</th>
<th>Unclassified</th>
<th>False Negatives</th>
<th>False Positives</th>
<th>% Characters Classified</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>54</td>
<td>5</td>
<td>1</td>
<td>12</td>
<td>90</td>
</tr>
<tr>
<td>2</td>
<td>51</td>
<td>7</td>
<td>2</td>
<td>13</td>
<td>85</td>
</tr>
<tr>
<td>3</td>
<td>55</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td>91.7</td>
</tr>
<tr>
<td>4</td>
<td>55</td>
<td>5</td>
<td>0</td>
<td>6</td>
<td>91.7</td>
</tr>
<tr>
<td>5</td>
<td>43</td>
<td>5</td>
<td>12</td>
<td>2</td>
<td>71.67</td>
</tr>
<tr>
<td>6</td>
<td>42</td>
<td>8</td>
<td>10</td>
<td>8</td>
<td>70</td>
</tr>
<tr>
<td>7</td>
<td>54</td>
<td>5</td>
<td>1</td>
<td>11</td>
<td>90</td>
</tr>
<tr>
<td>8</td>
<td>49</td>
<td>3</td>
<td>8</td>
<td>10</td>
<td>81.7</td>
</tr>
<tr>
<td>9</td>
<td>47</td>
<td>3</td>
<td>10</td>
<td>1</td>
<td>78.3</td>
</tr>
<tr>
<td>10</td>
<td>25</td>
<td>10</td>
<td>25</td>
<td>5</td>
<td>41.7</td>
</tr>
</tbody>
</table>
Table 5.2: Number of Support Vectors and the values of b for one against All SVM

<table>
<thead>
<tr>
<th>Class</th>
<th>Number of Support Vector</th>
<th>b</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>169</td>
<td>-0.6605</td>
</tr>
<tr>
<td>2</td>
<td>200</td>
<td>-0.9219</td>
</tr>
<tr>
<td>3</td>
<td>133</td>
<td>-0.6682</td>
</tr>
<tr>
<td>4</td>
<td>169</td>
<td>-0.6471</td>
</tr>
<tr>
<td>5</td>
<td>191</td>
<td>-0.8547</td>
</tr>
<tr>
<td>6</td>
<td>231</td>
<td>-1.0658</td>
</tr>
<tr>
<td>7</td>
<td>178</td>
<td>-0.5202</td>
</tr>
<tr>
<td>8</td>
<td>204</td>
<td>-1.4592</td>
</tr>
<tr>
<td>9</td>
<td>123</td>
<td>-0.9798</td>
</tr>
<tr>
<td>10</td>
<td>226</td>
<td>0.9801</td>
</tr>
</tbody>
</table>

5.5.2 Decision Tree Based Support Vector Machines

The unclassifiable regions in the one against all support vector machines are resolved by using decision tree based support vector machines (Liu et al., 2007). In this N-1 hyper planes or N-1 support vector machines are constructed, where the $i^{th}$ support vector machine is trained so that it separates the class $i$ data from data belonging to one of the class $i+1, i+2, i+3, ..., N$. The construction of hyper planes for a four class problem is shown in the Figure 5.8. After training, classification is performed with the first to the $(N-1)^{th}$ support vector machines,
If the $i^{th}$ support vector machine classifies a datum into class $i$, classification terminates. Otherwise, classification is performed until the datum is classified into the definite class. The number of support vectors for the SVM with decision tree approach is less when compared to the one against all method. The number of support vectors and the value of $b$ for one fold are given below in Table 5.3.

**Table 5.3: Number of Support Vectors and the Values of $b$ for Decision Tree Based Support Vector Machine**

<table>
<thead>
<tr>
<th>Class</th>
<th>Number of Support Vector Machines</th>
<th>$b$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>169</td>
<td>-0.6605</td>
</tr>
<tr>
<td>2</td>
<td>192</td>
<td>-0.8136</td>
</tr>
<tr>
<td>3</td>
<td>126</td>
<td>-0.6124</td>
</tr>
<tr>
<td>4</td>
<td>152</td>
<td>-0.5205</td>
</tr>
<tr>
<td>5</td>
<td>127</td>
<td>-0.4935</td>
</tr>
<tr>
<td>6</td>
<td>169</td>
<td>-0.5189</td>
</tr>
<tr>
<td>7</td>
<td>124</td>
<td>-0.4771</td>
</tr>
<tr>
<td>8</td>
<td>96</td>
<td>-0.2850</td>
</tr>
<tr>
<td>9</td>
<td>61</td>
<td>-0.2921</td>
</tr>
</tbody>
</table>
This method is tested with 10 fold cross validation similar to one against all approach by choosing parameter $c=6$ and $\sigma =4.5$ to get the maximum accuracy. The confusion matrix and summary of the confusion matrix are shown in Figure 5.9 and Table 5.4 respectively.

Figure 5.9: Confusion Matrix with Decision Tree Based Support Vector Machine

Table 5.4: Summary of Performance Metrics for Decision Tree Based Support Vector Machine

<table>
<thead>
<tr>
<th>Class</th>
<th>Sensitivity</th>
<th>Specificity</th>
<th>PPV</th>
<th>NPV</th>
<th>Accuracy</th>
<th>F-Measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>90</td>
<td>97.8</td>
<td>81.8</td>
<td>98.9</td>
<td>97</td>
<td>0.86</td>
</tr>
<tr>
<td>2</td>
<td>86.7</td>
<td>97.41</td>
<td>78.8</td>
<td>98.5</td>
<td>96.3</td>
<td>0.83</td>
</tr>
<tr>
<td>3</td>
<td>91.7</td>
<td>99.6</td>
<td>96.5</td>
<td>99.1</td>
<td>98.8</td>
<td>0.94</td>
</tr>
<tr>
<td>4</td>
<td>91.7</td>
<td>98.9</td>
<td>90.2</td>
<td>99.1</td>
<td>98.2</td>
<td>0.91</td>
</tr>
<tr>
<td>5</td>
<td>75</td>
<td>99.6</td>
<td>95.74</td>
<td>97.3</td>
<td>97.2</td>
<td>0.84</td>
</tr>
<tr>
<td>6</td>
<td>73.3</td>
<td>98.3</td>
<td>83</td>
<td>97</td>
<td>95.8</td>
<td>0.78</td>
</tr>
<tr>
<td>7</td>
<td>91.7</td>
<td>98.1</td>
<td>84.6</td>
<td>99.1</td>
<td>97.5</td>
<td>0.88</td>
</tr>
<tr>
<td>8</td>
<td>81.7</td>
<td>96.9</td>
<td>74.2</td>
<td>97.9</td>
<td>95.3</td>
<td>0.78</td>
</tr>
<tr>
<td>9</td>
<td>83.3</td>
<td>98.5</td>
<td>86.2</td>
<td>98.2</td>
<td>97</td>
<td>0.85</td>
</tr>
<tr>
<td>10</td>
<td>58.3</td>
<td>95</td>
<td>56.5</td>
<td>95.4</td>
<td>91.3</td>
<td>0.57</td>
</tr>
</tbody>
</table>
5.6 Results and Discussion

In this chapter, a classification model for character recognition using support vector machines has been discussed. As SVMs are basically binary classifiers, two different approaches were used to construct hyper planes for a multi-class problem. The first approach is one against all support vector machines and the other one is decision tree based support vector machine. As the data set considered comes under nonlinear separable case and the data was projected onto high dimensional space using the Gaussian kernel. The model has been implemented in MATLAB (R2009b). The classification was performed using 10-fold cross validation with 60 test images and 540 images for learning. But only a few images of the 540 images were used for constructing the decision boundary and these images are referred to as support vectors. In the first approach the number of support vectors was high when compared to the number of support vectors in the second approach. Because of the unclassifiable regions in the first approach the percentage of characters classified correctly is less than when compared to second approach. Out of the 600 images 494 images were classified correctly with decision tree based support vector machine i.e., the accuracy is 82.3%.
5.7 Conclusions

This chapter was presented with the aim of demonstrating the role that can be played by SVM in recognizing handwritten character images. As mentioned in an earlier part of the chapter, the SVMs are insensitive to the size of the training set and the learning phase is decided by the number of support vectors which is very less than the number of samples in the training set. The accuracy of the classification which was performed by the SVM is also high when compared to the neural networks indicating high generalizing ability. The values of F-measure are relatively high with support vector machines when compared to neural networks discussed in the previous chapter for all the classes except for the class with label 10. But the disadvantage with SVM is that the time required is higher than that required by neural networks because a number of hyper planes are to be constructed for a multi class problem.