CHAPTER 2

2. MACHINE LEARNING

Machine learning is a technique that allows computers to automatically predict the accuracy on previous observations [33]. Primary aim of machine learning is to automate the process of extracting hidden information from data by deploying both computational and statistical techniques. These machine learning techniques [34] are extensively deployed to resolve issues related to natural language processing which includes speech recognition, document categorization, document segmentation, part-of-speech tagging, word-sense disambiguation, named entity recognition, parsing, machine translation and transliteration.

The two different kinds of learning are inductive and deductive. Massive data sets are used during inductive learning to extract rules and patterns. Learning/training and prediction are the two essential processes that comprise machine learning. Training data here is used by the system which is essentially a set of examples. The focus is to automatically derive effective and accurate model using the training data or set of examples. Training data provides knowledge to the domain which is the domain features based on which examples are drawn. The larger the training set will determine the higher quality of the model [35] [36].

Prediction forms the second phase of machine learning wherein a series of inputs are mapped according to their respective target values. Machine learning aims at creating a good prediction performance model that can use the test data and engender high quality generalization capability for unseen data.

Machine learning algorithms are segregated as supervised learning [37], unsupervised learning [38] [39], semi-supervised learning [40], reinforcement learning and transduction. The target function is identified using training data in supervised learning, wherein every example has a corresponding label. If the label is found to be discrete, the task is known as classification. However, if the label is a real value the task then poses regression issues. Target function is deployed to predict the label for new case. May infer that learning in the training set is not limited to being a task of remembering examples but also as a means of generalization for unseen cases [41].

The system during unsupervised learning, attempts to mine patterns that do not contain precise information about the target concept in the data. These systems focus on analyzing data and then categorize it using the training examples which have no class labels.

Semi-supervised learning deploys labeled and unlabeled examples to create a model. Learning involves incorporating a rule which can help to determine the action to be taken based on the observation. Transduction and supervised learning have certain similarities. The learning as such doesn’t exclusively create a model but the algorithm used focuses on predicting the output depending on the details of the training input, training output and test input inherent during training.

2.1. SUPERVISED CLASSIFICATION ALGORITHM

Supervised classification considered as automatic learning that focuses on the inherent modeling input/output relationships [42]. Supervised learning aims at optimal function identification mapped between input data X and output variable which is class label Y such that Y = f(X). The above mapping is performed and is based on observations drawn as a sample of input variables X that feature examples of the given problem.

Supervised learning main task is to chart out a classifier based on a set of classified training examples. The labeled example here consists of a pair of object and the respective class label. Labeled examples that provide the learning algorithm are referred to as the training set. Classifier is created based on training data that is provided to the classification algorithm. Major issue here related to supervised learning algorithm is that of generalization or in other words the ability for correct label prediction even in unseen data.

Performance evaluation of the classifier is determined by deploying several varying labeled examples referred to as the test set. Most learned classifiers can accurately predict the correct class label based on the model developed from training examples, it is not for new examples. Therefore using a different data set is apt while testing the learned model ability and for generalizing new data points. Percentage of correct classified test examples is referred to as the classification rate or also as the prediction accuracy. If the dataset is larger, valuing classification accuracy also reaches its height.

Some of the supervised learning algorithms used for solving classification problems are Linear Models, Neural Networks, k-Nearest Neighbor (k-NN), Decision Trees, Naïve Bayes, Support Vector Machine.

2.2. SUPPORT VECTOR MACHINE

Support Vector Machine is a newfangled approach deployed in supervised pattern classification and one whose application to a varied pattern recognition issues has proven to be successful. SVM as a supervised machine learning method is rewarding as well-developed learning theory which is apt for statistical learning theory [43]. SVM is primarily based on strong yet simplified mathematical foundations and results but extremely powerful algorithms [44]. A binary classifier used for constructing a hyperplane separating class members from non-members in the input space is a simple process. SVM additionally identifies nonlinear decision function in the input space through the process of data mapping into a higher dimensional feature space and furthermore separating it using maximum margin hyperplane. Automatically here the system selects its subset of informative points referred to as the support vectors. These are then used to represent the separating hyperplane which is a bare linear combination of these points. The SVM training algorithm thus resolves the simple convex optimization problem [45] [46].

2.2.1. Geometrical Interpretation of SVM

Typically, the machine is offered through a set of training examples, (x_i, y_i) where the x_i are the real world data instances and the y_i are the labels that show the class to which each of these instances belong. For the two class pattern recognition problem, $y_i = +1$or $y_i = -1$. A training example (x_i, y_i) is called positive if $y_i = +1$and negative in varying situations. SVMs construct a hyperplane demarcating these two classes. This facilitates the SVM algorithm to achieve maximum separation between the classes [47] as shown in Fig 2.1.

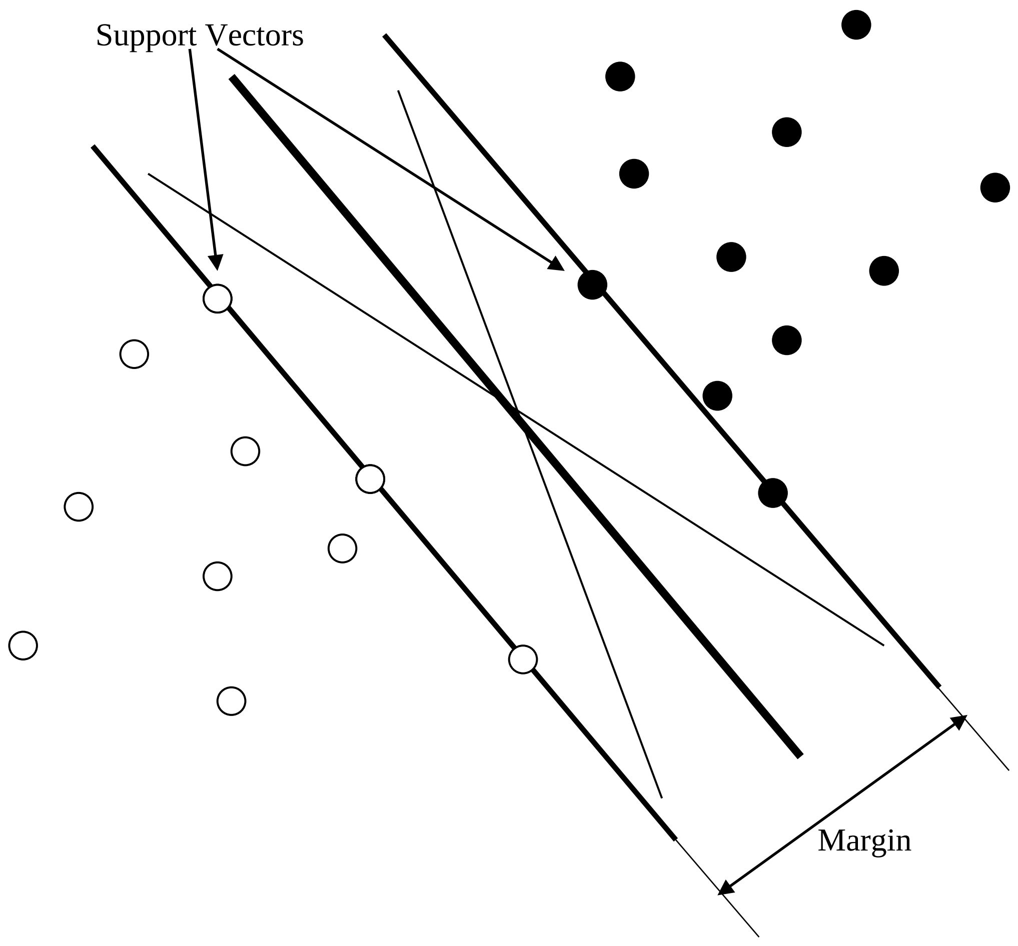


Fig. 2.1 Maximum Margin Classifier

Differentiating classes using a large margin almost eliminates effect on expected generalization error. A minimum generalization error here refers to the new examples (data points with unknown class values) that arrive for classification, possibility of error during prediction (of the class which it belongs) with respect to the learned classifier (hyperplane) can be possibly low.

It is obvious that such a classifier is one that achieves maximum separation-margin between classes. Fig 2.1 elucidates the concept of maximum margin. Parallel to the classifier the two planes and those in the data that pass through one or more points are known as bounding planes. Distance between bounding planes is known as margin and SVM learning, indicates finding a hyperplane that can maximize the margin.

The dataset shows points that fall on the bounding planes and are referred to as the support vectors. In classification these points have a significant role, thus the algorithm is known as the Support Vector Machines. The term machine refers to algorithm [47].

In [46] clearly shows that when the training vectors are separated using an optimal hyperplane without errors, test sample expected error rate is confined by the ratio of expectation of support vectors to number of training vectors. As this ratio is independent of any inherent problem regarding dimension and if number of support vectors is less, in that scenario excellent generalization is rest assured. However, is misclassification of data points as shown in Fig. 2.2**,** occurs when minimizing number of misclassifications whilst simultaneously maximizing the margin in correctly classified examples. In this scenario it can infer that SVM training algorithm allows a training error.

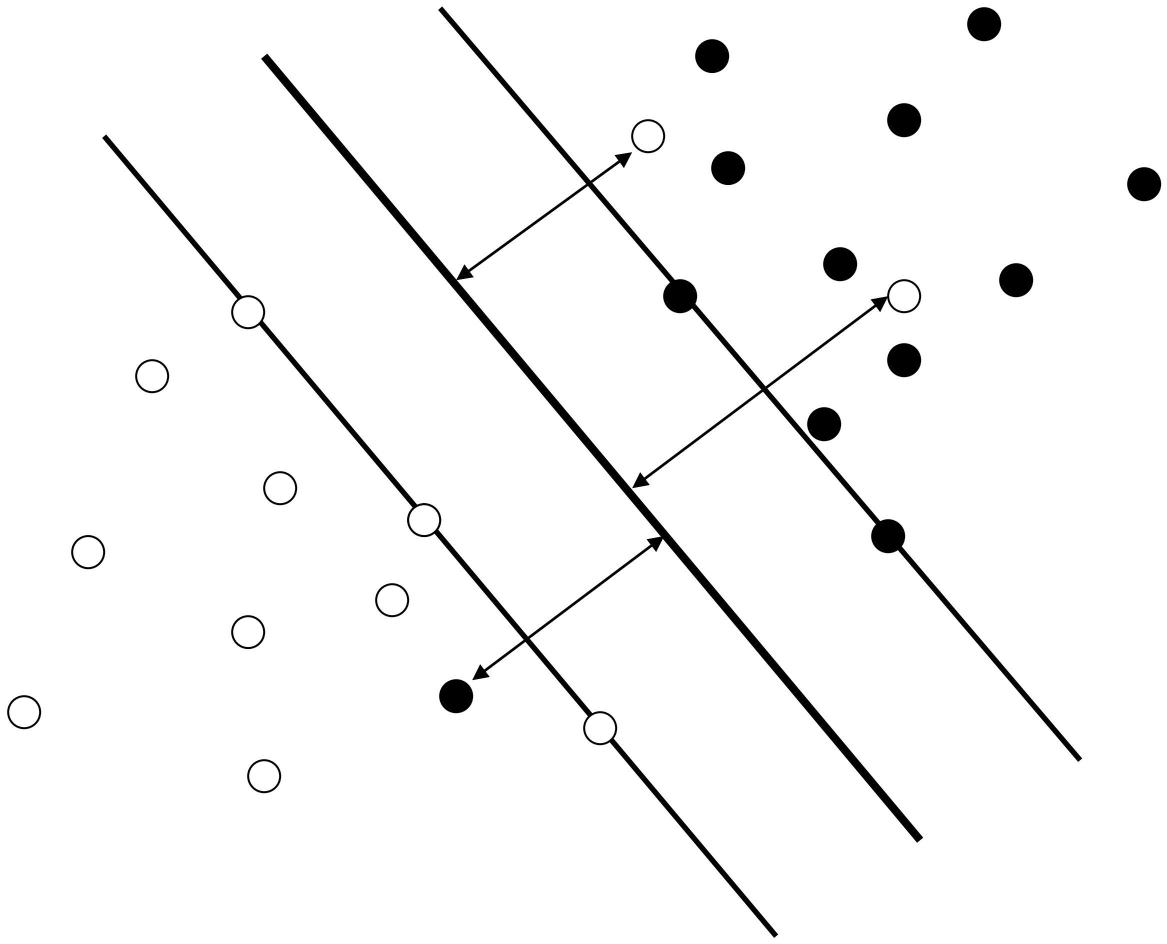


Fig. 2.2 Linear Classifier with Training Error

Another situation may exist in such a scenario where the clustered points have two classes and are not linearly separable as illustrated Fig. 2.3 and the linear classifier subsequently would be required to tolerate a considerably extensive training error. In these scenarios, non-linear mapping of data into some higher dimensional space is referred to as feature space, F, which allows it to become linearly separable. For distinguishing between these two types of spaces, the data point's original space is named as the input space. In the feature space the hyperplane corresponds to a highly non-linear that resolute the surface in the original input space. Thus the classifier is known as the non-linear classifier [47].

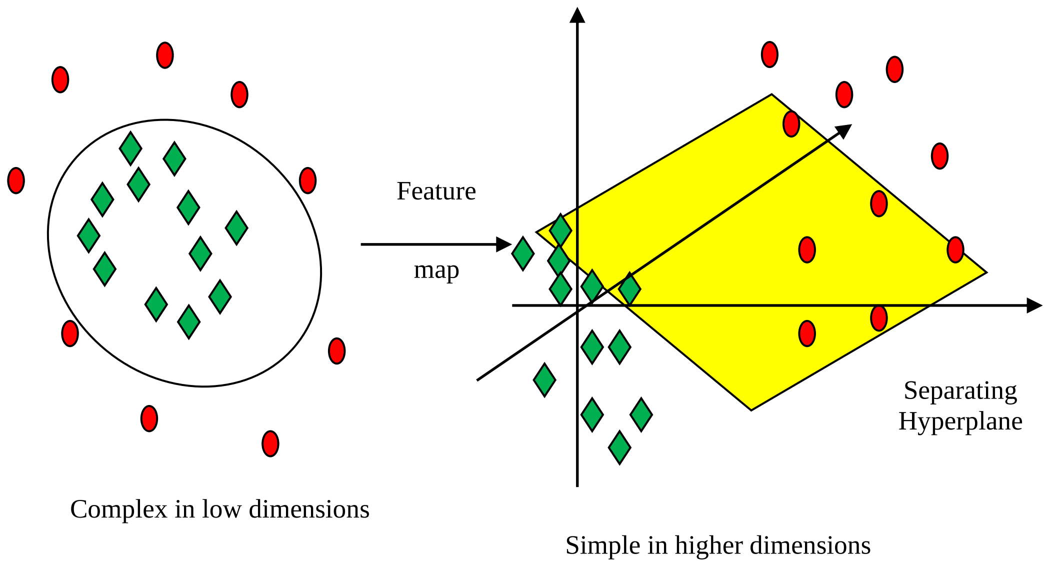


Fig. 2.3 Nonlinear Classifier

Data mapping into higher dimensional space includes substantial computation particularly when the date itself could be one with high dimensional is shown in Fig. 2.4. However, explicit mapping is not required for higher dimensional space to determine hyper plane or classifier, all computations are performed in the input space itself [47].

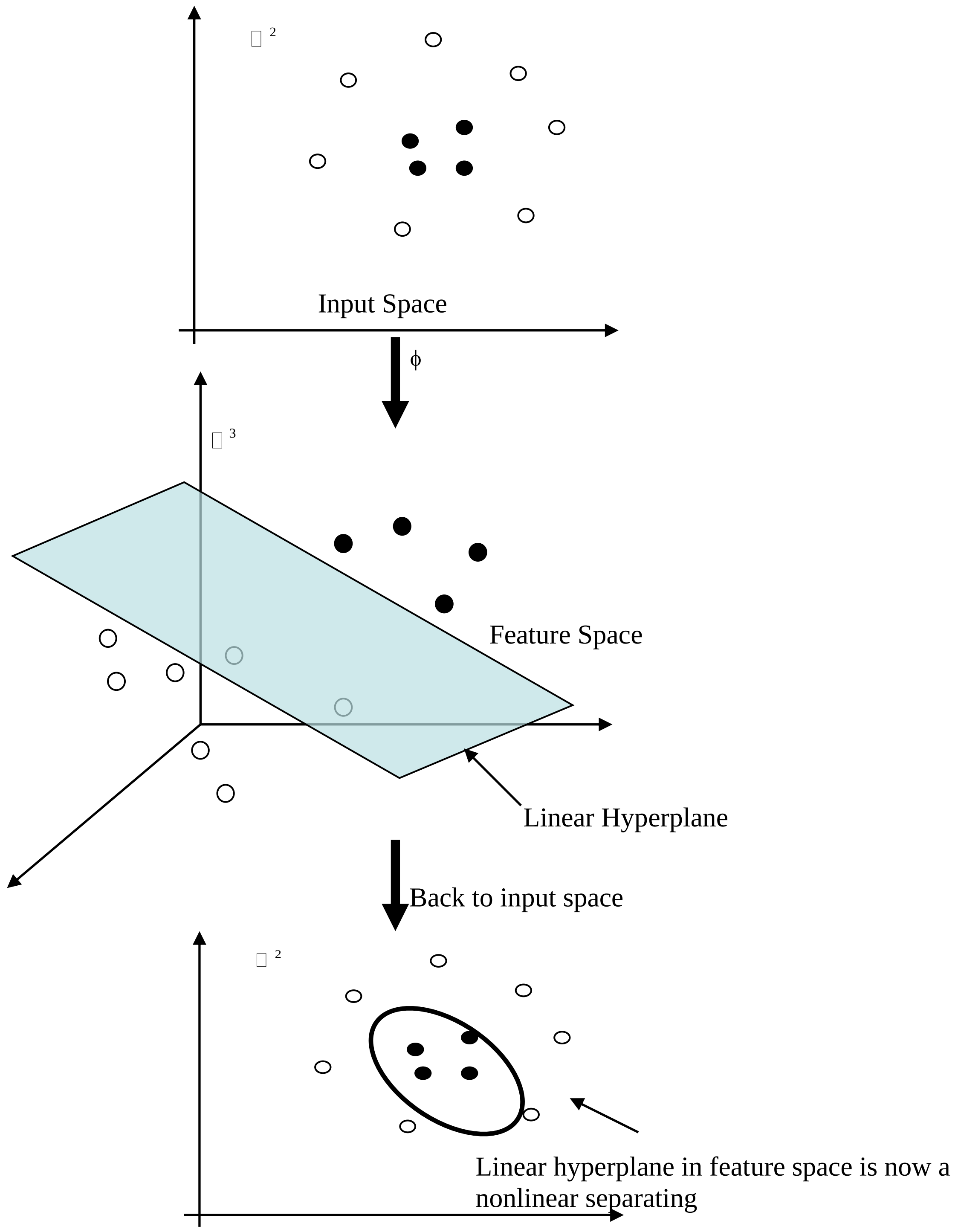


Fig. 2.4 Input Space and Feature Space

2.2.2. SVM Formulation - Hard Margin

The following notations are used

m = number of data points in the training set

n = number of features (variables) in the data

$\boldsymbol{x}_i = \begin{bmatrix} x_{i1} \\ x_{i2} \\ \vdots \\ x_{in} \end{bmatrix}$, n - dimensional vector which represent a data point in input space.

$d_i = D_{ii}$ = Target value of the ith data, it takes +1 or -1 value

$\boldsymbol{d} = \begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_m \end{bmatrix}$, a vector representing target value of m data points

$$\boldsymbol{D} = diag(\boldsymbol{d}) = \begin{bmatrix} d_1 & 0 & \cdots & 0 \\ 0 & d_2 & \dots & 0 \\ \vdots & \vdots & . & \vdots \\ 0 & 0 & \dots & d_m \end{bmatrix}$$

$\boldsymbol{w} = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \end{bmatrix}$, weight vector orthogonal to the hyper plane

$$w_1x_1 + w_2x_2 + \dots w_nx_n - \gamma = 0\,. \qquad (2.1)$$

γ is a scalar which is generally known as bias term

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{x}_1^T \\ \boldsymbol{x}_2^T \\ . \\ . \\ \boldsymbol{x}_m^T \end{bmatrix}, \ \boldsymbol{AA^T} = \begin{bmatrix} \boldsymbol{x}_1^T\boldsymbol{x}_1 & \boldsymbol{x}_1^T\boldsymbol{x}_2 & . & . & \boldsymbol{x}_1^T\boldsymbol{x}_m \\ \boldsymbol{x}_2^T\boldsymbol{x}_1 & \boldsymbol{x}_2^T\boldsymbol{x}_2 & . & . & \boldsymbol{x}_2^T\boldsymbol{x}_m \\ . & . & . & . & . \\ . & . & . & . & . \\ \boldsymbol{x}_m^T\boldsymbol{x}_1 & \boldsymbol{x}_m^T\boldsymbol{x}_2 & . & . & \boldsymbol{x}_m^T\boldsymbol{x}_m \end{bmatrix}$$

$\boldsymbol{AA^T}$ is called linear kernel of the dataset

$\phi(.) \rightarrow \boldsymbol{x}$ A nonlinear mapping function that maps input vector $\boldsymbol{x}$ into a high dimensional feature vector

$$\boldsymbol{K} = \begin{bmatrix} \phi(\boldsymbol{x}_1)^T\phi(\boldsymbol{x}_1) & \phi(\boldsymbol{x}_1)^T\phi(\boldsymbol{x}_2) & . & . & \phi(\boldsymbol{x}_1)^T\phi(\boldsymbol{x}_m) \\ \phi(\boldsymbol{x}_2)^T\phi(\boldsymbol{x}_1) & \phi(\boldsymbol{x}_2)^T\phi(\boldsymbol{x}_1) & . & . & \phi(\boldsymbol{x}_2)^T\phi(\boldsymbol{x}_m) \\ . & . & . & . & . \\ . & . & . & . & . \\ \phi(\boldsymbol{x}_m)^T\phi(\boldsymbol{x}_1) & \phi(\boldsymbol{x}_m)^T\phi(\boldsymbol{x}_2) & . & . & \phi(\boldsymbol{x}_m)^T\phi(\boldsymbol{x}_m) \end{bmatrix}$$

K is called the non-linear Kernel of input dataset.

$\boldsymbol{Q}$ **=** anmxm matrix whose (i,j)th element is $d_i d_j \phi(\boldsymbol{x}_i)^T \phi(\boldsymbol{x}_j)$

$\boldsymbol{Q}$ = K **.*** $\left(\boldsymbol{d} * \boldsymbol{d}^T\right)$**,** where **.*** represent element wise multiplication

The SVM build the optimal hyper plane with the view of geometrical calculation which is represented as $w^T x - \gamma = 0$ among 2 classes of samples. Here, ‘w’ represents the vector weight values and the ‘γ’ represents the threshold value. The main goal is to divide the combined places with maximum variation.

$$w^T x - \gamma = 1 \quad (2.2)$$

$$w^T x - \gamma = -1 \quad (2.3)$$

such that data points with d = -1 satisfy the constraints

$$w^T x - \gamma \leq -1 \quad (2.4)$$

and data points with d = +1 satisfy

$$w^T x - \gamma \geq 1. \tag{2.5}$$

The perpendicular distance of the bounding plane $w^T x - \gamma = 1$ from the origin is

$$|-\gamma + 1|/||w|| \tag{2.6}$$

and the perpendicular distance of the bounding plane $w^T x - \gamma = -1$ from the origin is

$$|-\gamma - 1|/||w||. \tag{2.7}$$

The margin between the optimal hyperplane and the bounding plane is $1/||w||$, and so the distance between the bounding hyperplanes is $2/||w||$. Then the learning problem is formulated as an optimization problem as given below

$$\begin{aligned} &\text{Minimize} \quad = \frac{1}{2}\|\mathbf{w}\|^2 \\ &\text{subject to} \quad D_{ii}(\mathbf{w}^T\mathbf{x}_i - \gamma) \geq 1, i = 1, \ldots, l. \end{aligned}$$

The training of SVM consists of finding w and γ, given the matrix of data points A and the corresponding class vector d**.** Once $\boldsymbol{w}$ and γ are obtained, the decision boundary can be obtained as $\boldsymbol{w}^T\boldsymbol{x} - \gamma = 0$. The decision function is given by $f(\boldsymbol{x}) = \text{sign}(\boldsymbol{w}^T\boldsymbol{x} - \gamma)$. This decision boundary is calculated based on decision function which is calculated as f(x) = sign ($w^Tx - \gamma$). Here sign value of $w^Tx - \gamma$ is used to classify newly arrived data point. By adapting the lagrangian dual variables, SVM learning can be done easily.

Lagrangian dual

The dual of the primal problem is using Lagrangian dual is given by.

$$\max_{\mathbf{u}\geq 0}\left[\min_{\mathbf{w},\gamma} \mathrm{L}(\mathbf{w}, \gamma, \mathbf{u}\right] \tag{2.8}$$

Where

$$\mathrm{L}(\mathbf{w}, \gamma, \mathbf{u}) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{k=1}^{m} u_k[\mathrm{d}_k(\mathbf{w}^T\mathbf{x}_k - \gamma) - 1] \tag{2.9}$$

The problem is solved using Lagrangian duality to find values of w and γ which minimizes $\mathrm{L}(\mathbf{w},\gamma,\mathbf{u})$ for a given u.

$$\frac{\partial}{\partial\gamma}\mathrm{L}\left(\mathbf{w},\gamma,\mathbf{u}\right)=0 \text{ and } \frac{\partial}{\partial \mathrm{w}}\mathrm{L}\left(\mathbf{w},\gamma,\mathbf{u}\right)=0 \tag{2.10}$$

This leads to

$$\sum_{k=1}^{m} u_k d_k = 0 \tag{2.11}$$

and

$$\mathbf{w}=\sum_{\mathrm{k}=1}^{\mathrm{m}} u_{\mathrm{k}} d_{\mathrm{k}} \mathbf{x}_{\mathrm{k}} \tag{2.12}$$

Substituting

$$\mathbf{w}=\sum_{\mathrm{k}=1}^{\mathrm{m}} \mathrm{u_k d_k} \mathbf{x}_{\mathrm{k}} \text{ into } \mathrm{L}(\mathbf{w},\gamma,\mathbf{u}) \tag{2.13}$$

$$\mathrm{L}(\mathbf{w},\gamma,\mathbf{u})=\frac{1}{2}\|\mathbf{w}\|^2 - \sum_{\mathrm{i}=1}^{\mathrm{m}} \mathrm{u_i}[\mathrm{d_i}(\mathbf{w}^{\mathrm{T}}\mathbf{x}_{\mathrm{i}}-\gamma)-1]$$

$$= \frac{1}{2}(\mathbf{w}^{\mathrm{T}}\mathbf{w}) - \sum_{\mathrm{i}=1}^{\mathrm{m}} \mathrm{u_i}[\mathrm{d_i}(\mathbf{w}^{\mathrm{T}}\mathbf{x}_{\mathrm{i}}-\gamma)-1]$$

$$=\frac{1}{2}\sum_{\mathrm{i}=1}^{\mathrm{m}}\sum_{\mathrm{j}=1}^{\mathrm{m}} \mathrm{u_i u_j d_i d_j}(\mathbf{x}_{\mathrm{i}}{}^{\mathrm{T}}\mathbf{x}_{\mathrm{j}}) - \sum_{\mathrm{i}=1}^{\mathrm{m}} \mathrm{u_i d_i}(\mathbf{w}^{\mathrm{T}}\mathbf{x}_{\mathrm{i}}) + \gamma\sum_{i=1}^{m} u_i d_i + \sum_{i=1}^{m} u_i \tag{2.14}$$

Since $\sum_{i=1}^{m} u_i d_i = 0$

$$\mathrm{L}(\mathbf{w},\gamma,\mathbf{u}) = \frac{1}{2}\sum_{\mathrm{i}=1}^{\mathrm{m}}\sum_{\mathrm{j}=1}^{\mathrm{m}} \mathrm{u_i u_j d_i d_j}(\mathbf{x}_{\mathrm{i}}{}^{\mathrm{T}}\mathbf{x}_{\mathrm{j}}) - \sum_{\mathrm{i}=1}^{\mathrm{m}}\sum_{\mathrm{j}=1}^{\mathrm{m}} \mathrm{u_i u_j d_j d_i}(\mathbf{x}_{\mathrm{j}}{}^{\mathrm{T}}\mathbf{x}_{\mathrm{i}}) + \sum_{i=1}^{m} u_i$$

$$= \sum_{i=1}^{m} u_i - \frac{1}{2}\sum_{\mathrm{i}=1}^{\mathrm{m}}\sum_{\mathrm{j}=1}^{\mathrm{m}} \mathrm{u_i u_j d_i d_j}(\mathbf{x}_{\mathrm{i}}{}^{\mathrm{T}}\mathbf{x}_{\mathrm{j}}) \tag{2.15}$$

Now Lagrangian is a function of **u** and is given by

$$L_D(\mathbf{u}) = \sum_{i=1}^{m} u_i - \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m} u_i u_j d_i d_j (\mathbf{x}_i^T \mathbf{x}_j) \tag{2.16}$$

U must satisfy the condition

$$\sum_{k=1}^{m} u_k d_k = 0 \tag{2.17}$$

The optimization problem now reduces to

$$\text{Max} \;\; L_D(\mathbf{u}) = \sum_{i=1}^{m} u_i - \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m} u_i u_j d_i d_j (\mathbf{x}_i^T \mathbf{x}_j) \tag{2.18}$$

$$\text{Subject to} \sum_{i=1}^{m} u_i d_i = 0, \; u_i \geq 0, \; i = 1,2,\ldots,m$$

The above quadratic programming problem yields $u_i s$ and the decision function is given by

$$f(\boldsymbol{x}) = \text{sign}(\mathbf{w}^T\mathbf{x} - \gamma) = \text{sign}(\sum_{i=1}^{m} u_i d_i \mathbf{x}_i^T \mathbf{x} - \gamma) \tag{2.19}$$

Where γ is measured by considering the margin with $u_i \neq 0$. There is an presence of dual vairable u_i for every data x_i. Most of the time, value of u_i is zero and the remaining non zero u_i values are refereed as support vectors due to their property of decision function f(x). The decision function f(x) can be represented as given in following equation where the set of index of support vectors are declared as svmindex

$$\text{sign}(\sum_{i \in svindex} u_i d_i \boldsymbol{x}_i^T \boldsymbol{x} - \gamma) \tag{2.20}$$

2.2.3. SVM Formulation with Soft Margin - L_1 Norm

As shown in Fig. 2.5, the data points with noisy features can be divided optimally over the regions A_+ and A_- with reduced training error which would cause worst generalization. This can be avoided by dividing the plane maximally with the assignment of small number of points on negative side. This deviation occurred with the bounding place is called as error.

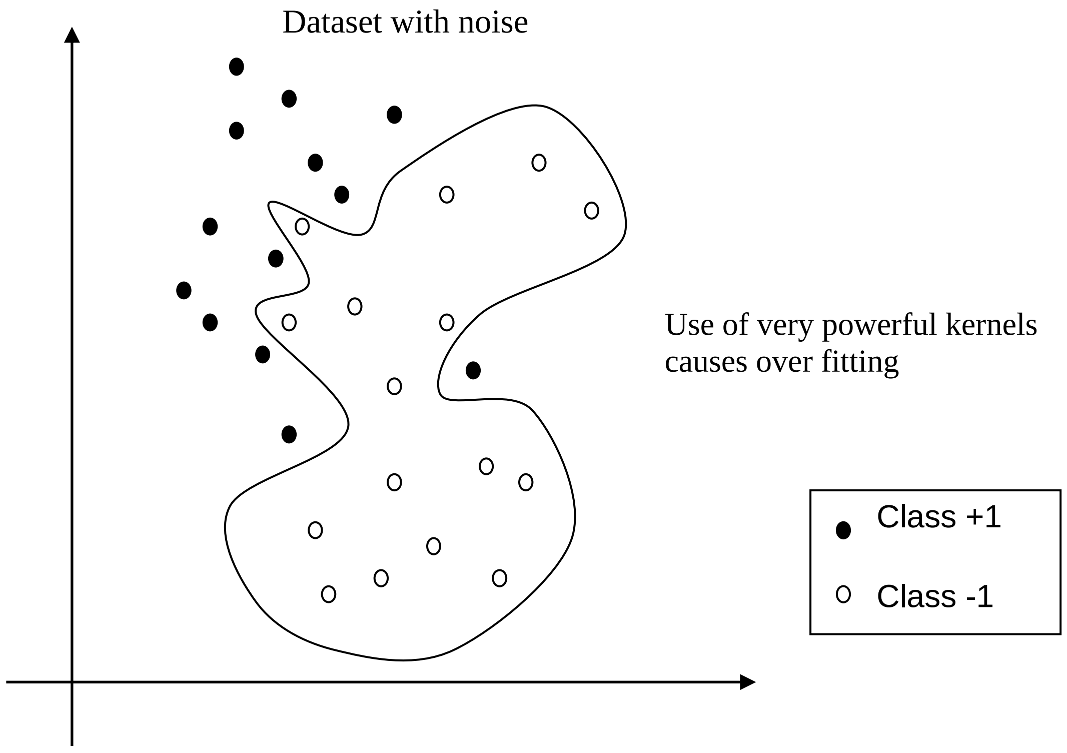


Fig. 2.5 Data which Require Linear SVM Formulation with Soft Margin

The main goal of the SVM is to find the classifier which can separate the places with maximum bounding value and reduced error value. However, these maximum margin value and reduced error for the smaller data point learning is contradict to each other. That is maximum value of margin would cause more error. This is resolved by introducing the control parameter C which can adjust the value of weight so that contradiction can be controlled efficiently [47].

Here there is a chance of misclassification error which can be handled and corrected by introducing the slack variable ξ_i, so that violation that occurs due to specifications of some constraints can be avoided. This is called as SVM with soft margin and it is calculated as

$$\begin{aligned}&\min_{\mathbf{w},\gamma,\xi} \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{m}\xi_i\\ &\text{subject to } d_i(\mathbf{w}^T\mathbf{x}_i - \gamma) + \xi_i - 1 \geq 0, \quad 1 \leq i \leq m \\ &\qquad\qquad\qquad\qquad\qquad \xi_i \geq 0, \quad 1 \leq i \leq m\end{aligned} \tag{2.21}$$

The attempt to minimizing error and quantity $C\mathbf{e}^{\mathrm{T}}\boldsymbol{\xi} + \frac{1}{2}\mathbf{w}^{\mathrm{T}}\mathbf{w}$ for the varying parameter values of w and £ leads to increased separation outcome of bounding places with increased number of data points. Here C is referred as regularization parameter value whose main goal is to achieve the trade-off between the different objective function. The optimal selection C value would ensure the best generalization outcome of the classifier. This can be represented as Lagrangian as follows:

$$\begin{aligned} L(\mathbf{w},\gamma,\boldsymbol{\xi},\boldsymbol{u},\boldsymbol{\mu}) &= \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{m}\xi_i - \sum_{i=1}^{m}u_i\left[d_i(\mathbf{w}^T\mathbf{x}_i - \gamma) + \xi_i - 1\right] - \sum_{i=1}^{m}\mu_i\xi_i \\ &= \frac{1}{2}\mathbf{w}^T\mathbf{w} + \sum_{i=1}^{m}(C - u_i - \mu_i)\xi_i - \left(\sum_{i=1}^{m}u_i d_i \mathbf{x}_i^T\right)\mathbf{w} - \left(\sum_{i=1}^{m}u_i d_i\right)\gamma + \sum_{i=1}^{m}u_i \end{aligned} \tag{2.22}$$

where $\boldsymbol{u},\boldsymbol{\mu}$ are the Lagrange multipliers, the Wolfe dual of the problem is:

$$\begin{aligned} &\max_{\mathbf{u},\boldsymbol{\mu}} L(\mathbf{w},\gamma,\boldsymbol{\xi},\boldsymbol{u},\boldsymbol{\mu}) \\ &\text{subject to } \frac{\partial L}{\partial \mathbf{w}} = \mathbf{0} \\ &\qquad \frac{\partial L}{\partial \gamma} = 0 \\ &\qquad \frac{\partial L}{\partial \xi_i} = 0 \qquad 1 \le i \le m \\ &\qquad \boldsymbol{u} \ge \mathbf{0} \\ &\qquad \boldsymbol{\mu} \ge \mathbf{0} \end{aligned} \tag{2.23}$$

The constraint $\frac{\partial L}{\partial \mathbf{w}} = 0$ implies

$$\mathbf{w}^T - \sum_{i=1}^{m}u_i d_i \mathbf{x}_i^T = 0 \tag{2.24}$$

That is

$$\mathbf{w} = \sum_{i=1}^{m}u_i d_i \mathbf{x}_i \tag{2.25}$$

The constraint $\frac{\partial L}{\partial \gamma} = 0$ implies

$$\sum_{i=1}^{m} \alpha_i d_i = 0 \qquad (2.26)$$

The constraints $\frac{\partial L}{\partial \xi_i} = 0$ imply

$$C - u_i - \mu_i = 0, \qquad 1 \le i \le m. \qquad (2.27)$$

Note that $\boldsymbol{u} \ge \mathbf{0}$, $\mu \ge \mathbf{0}$, and $0 \le u_i \le C$.

Substituting these results into $L(\mathbf{w}, \gamma, \boldsymbol{\xi}, \boldsymbol{u}, \boldsymbol{\mu})$:

$$\begin{aligned} L(\mathbf{w}, \gamma, \boldsymbol{\xi}, \boldsymbol{u}, \boldsymbol{\mu}) &= \frac{1}{2}\mathbf{w}^T\mathbf{w} + \sum_{i=1}^{m} 0 \times \xi_i - \mathbf{w}^T\mathbf{w} - 0 \times \gamma + \sum_{i=1}^{m} u_i \\ &= -\frac{1}{2}\mathbf{w}^T\mathbf{w} + \sum_{i=1}^{m} u_i \qquad (2.28) \\ &= \sum_{i=1}^{m} u_i - \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m} d_i d_j \mathbf{x}_i{}^T \mathbf{x}_j u_i u_j \end{aligned}$$

The dual problem is

$$\begin{aligned} &\max_{\boldsymbol{u}} L_D(\boldsymbol{u}) = \sum_{i=1}^{m} u_i - \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m} d_i d_j \mathbf{x}_i{}^T \mathbf{x}_j u_i u_j \\ &\text{subject to } \sum_{i=1}^{m} d_i u_i = 0 \qquad (2.29) \\ &\qquad 0 \le u_i \le C \qquad 1 \le i \le m \end{aligned}$$

or

$$\begin{aligned} &\min_{\boldsymbol{u}} L_D(\boldsymbol{u}) = \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m} d_i d_j \mathbf{x}_i{}^T \mathbf{x}_j u_i u_j - \sum_{i=1}^{m} u_i \\ &\text{subject to } \sum_{i=1}^{m} d_i u_i = 0 \qquad (2.30) \\ &\qquad 0 \le u_i \le C \qquad 1 \le i \le m \end{aligned}$$

In matrix form the dual problem can be written as follows:

Minimize $\mathrm{L_D}(\mathbf{u}) = \frac{1}{2}\mathbf{u}^{\mathrm{T}}\boldsymbol{Q}\mathbf{u} - \mathbf{e}^{\mathrm{T}}\mathbf{u}$

$$\mathbf{d}^{\mathrm{T}}\mathbf{u} = 0, \mathbf{0} \le \boldsymbol{u} \le C\boldsymbol{e}$$

Here $\boldsymbol{Q} = \boldsymbol{DAA^T D}$ and A is data matrix. Each data is a row vector in matrix A and u can be obtained by solving this quadratic programming problem.

$\boldsymbol{Q}$ can be computed as $\boldsymbol{Q} = (\boldsymbol{A} * \boldsymbol{A^T}) \bullet *(\boldsymbol{d} * \boldsymbol{d}^{\mathrm{T}})$

KKT complementary conditions have the following three cases for u_i

1. $u_i = 0$. Then ξ_i =0 (corresponding data points do not require positive ξ_i). Thus $\boldsymbol{x}_i$ (ith data point in the training set) is correctly classified.
2. $0 < u_i < C$. Then $d_i\left(\boldsymbol{w}^T \boldsymbol{x}_i - \gamma\right) + \xi_i = 1$ and $\xi_i = 0$. Then $\boldsymbol{x}_i$ is a support vector. Such support vectors with $0 < u_i < C$ are called unbounded support vectors
3. $u_i = C$ then $d_i\left(\boldsymbol{w}^T \boldsymbol{x}_i - \gamma\right) + \xi_i = 1$ and $\xi_i > 0$. Then $\boldsymbol{x}_i$ is a support vector. Such support vectors with $u_i = C$ are called bounded support vectors. If $0 \le \xi_i < 1$, then $\boldsymbol{x}_i$ is correctly classified, and if $\xi_i \ge 1$ it is misclassified.

For computing γ, one of those points whose Lagrangian multiplier falls between 0 and C can be used, that is one of those points which falls on the hyper planes. For these points $\boldsymbol{w}^T \boldsymbol{x}_i - \gamma = d_i$.

Such that $\gamma = \boldsymbol{w}^T \boldsymbol{x}_i - d_i = \sum_{j \in svmindex} u_j d_j \boldsymbol{x}_j{}^T \boldsymbol{x}_i - d_i$ and γ can be measured from the available data points based on their average value. Here the SVM depends on the final set of support vector indices based on which data separation between hyper planes can be ensured.

2.2.4. L2 Norm Linear SVM

The L2 norm SVM ensures the minimization of sum of square values of data points and also reduction of the squared margin value between the bounding planes [47]. This formulation procedure is given as follows:

$$\begin{aligned}
&\min_{\mathbf{w},\gamma,\xi} \frac{1}{2}\mathbf{w}^T\mathbf{w} + \frac{C}{2}\sum_{i=1}^{m}\xi_i^2 \\
&\text{subject to } d_i(\mathbf{w}^T\mathbf{x}_i - \gamma) + \xi_i - 1 \geq 0, \quad 1 \leq i \leq m \\
&\qquad\qquad\qquad\qquad\qquad\quad \xi_i \geq 0, \quad 1 \leq i \leq m
\end{aligned} \tag{2.31}$$

The Lagrangian is:

$$\begin{aligned}
L(\mathbf{w},\gamma,\boldsymbol{\xi},\boldsymbol{u}) &= \frac{1}{2}\mathbf{w}^T\mathbf{w} + \frac{C}{2}\sum_{i=1}^{m}\xi_i^2 - \sum_{i=1}^{m}u_i\left[d_i(\mathbf{w}^T\mathbf{x}_i - \gamma) + \xi_i - 1\right] \\
&= \frac{1}{2}\mathbf{w}^T\mathbf{w} + \frac{C}{2}\sum_{i=1}^{m}\xi_i^2 - \left(\sum_{i=1}^{m}u_i d_i \mathbf{x}_i{}^T\right)\mathbf{w} - \left(\sum_{i=1}^{m}u_i d_i\right)\gamma - \sum_{i=1}^{m}u_i\xi_i + \sum_{i}u_i
\end{aligned} \tag{2.32}$$

where $\boldsymbol{u}$ are the Lagrange multipliers.

Lagrangian multiplies in terms of constraint values $\xi_i \geq 0, \quad 1 \leq i \leq m$ cannot be assumed as Lagrangian L due to property of ξ_i which is a function of u_i. Here the values of u_i s are unbounded (on positive side) which is measured as follows:

The Wolfe dual of the problem is:

$$\begin{aligned}
&\max_{\mathbf{u},\boldsymbol{\mu}} L(\mathbf{w},\gamma,\boldsymbol{\xi},\boldsymbol{u},\boldsymbol{\mu}) \\
&\text{subject to } \frac{\partial L}{\partial \mathbf{w}} = \mathbf{0} \\
&\qquad\qquad \frac{\partial L}{\partial \gamma} = 0 \\
&\qquad\qquad \frac{\partial L}{\partial \xi_i} = 0 \qquad 1 \leq i \leq m \\
&\qquad\qquad \boldsymbol{u} \geq \mathbf{0}
\end{aligned} \tag{2.33}$$

The constraint $\frac{\partial L}{\partial \mathbf{w}} = 0$ implies:

$$\mathbf{w}^T - \sum_{i=1}^{m} u_i d_i \mathbf{x}_i^T = 0 \tag{2.34}$$

or

$$\mathbf{w} = \sum_{i=1}^{m} u_i d_i \mathbf{x}_i \tag{2.35}$$

The constraint $\frac{\partial L}{\partial \gamma} = 0$ implies

$$\sum_{i=1}^{m} u_i d_i = 0 \tag{2.36}$$

The constraints $\frac{\partial L}{\partial \xi_i} = 0$ imply

$$C\xi_i - u_i = 0, \qquad 1 \le i \le m. \tag{2.37}$$

where $\boldsymbol{u} \ge \mathbf{0}$

Substituting these into $L(\mathbf{w}, \gamma, \boldsymbol{\xi}, \boldsymbol{u})$:

$$\begin{aligned} L(\mathbf{w}, \gamma, \boldsymbol{\xi}, \boldsymbol{u}) &= \frac{1}{2}\mathbf{w}^T\mathbf{w} + \frac{1}{2C}\sum_{i=1}^{m} u_i^2 - \mathbf{w}^T\mathbf{w} - 0 \times \gamma - \frac{1}{C}\sum_{i=1}^{m} u_i^2 + \sum_{i=1}^{m} u_i \\ &= -\frac{1}{2}\mathbf{w}^T\mathbf{w} + \sum_{i=1}^{m} u_i - \frac{1}{2C}\sum_{i=1}^{m} u_i^2 \\ &= \sum_{i=1}^{m} u_i - \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m} d_i d_j \mathbf{x}_i^T \mathbf{x}_j u_i u_j - \frac{1}{2C}\sum_{i=1}^{m} u_i^2 \end{aligned}$$

$$= \sum_{i=1}^{m} u_i - \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m} d_i d_j u_i u_j (\mathbf{x}_i^T \mathbf{x}_j + \frac{\delta_{ij}}{C}) \tag{2.38}$$

Therefore the dual problem is:

$$\max_{\boldsymbol{u}} L(\boldsymbol{u}) = \;= \sum_{i=1}^{m} u_i - \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m} d_i d_j u_i u_j (\mathbf{x}_i^T \mathbf{x}_j + \frac{\delta_{ij}}{C})$$

$$\text{subject to} \quad \sum_{i=1}^{m} d_i u_i = 0 \qquad (2.39)$$

$$u_i \ge 0 \qquad 1 \le i \le m$$

The standard form in matrix format is

$$\min_{\boldsymbol{u}} L(\boldsymbol{u}) = \frac{1}{2}\boldsymbol{u}^T \boldsymbol{D}(\boldsymbol{A}\boldsymbol{A}^T + \frac{\boldsymbol{I}}{C})\boldsymbol{D}\boldsymbol{u} - \boldsymbol{e}^T\boldsymbol{u}$$

$$\text{subject to} \quad \boldsymbol{d}^T\boldsymbol{u} = 0 \qquad (2.40)$$

$$\boldsymbol{u} \ge \mathbf{0}$$

or,

$$\min_{\boldsymbol{u}} L(\boldsymbol{u}) = \frac{1}{2}\boldsymbol{u}^T \boldsymbol{Q}\boldsymbol{u} - \boldsymbol{e}^T\boldsymbol{u}$$

$$\text{subject to} \quad \boldsymbol{d}^T\boldsymbol{u} = 0 \qquad (2.41)$$

$$\boldsymbol{u} \ge \mathbf{0}$$

where Q can be computed as $\boldsymbol{Q}=(\boldsymbol{A}*\boldsymbol{A}^{\mathrm{T}} + \boldsymbol{I}/C).*(\boldsymbol{d}*\boldsymbol{d}^T)$

γ is computed as below:

According to the KKT complementary theorem, at optimal point:

$$u_i\left(d_i(\mathbf{w}^T\mathbf{x}_i - \gamma) + \xi_i - 1\right) = 0\,. \qquad (2.42)$$

$$\text{So either } u_i = 0 \text{ or } \left(d_i(\mathbf{w}^T\mathbf{x}_i - \gamma) + \xi_i - 1\right) = 0 \qquad (2.43)$$

that is , if $u_i > 0$, then $\left(d_i(\mathbf{w}^T\mathbf{x}_i - \gamma) + \xi_i - 1\right) = 0$

Substituting the optimal values of $\boldsymbol{w}$ *and* ξ_{i} ,

$$d_i\left(\sum_{j=1}^{m} u_j d_j x_j^T x_i + \frac{d_i}{C}u_i - \gamma\right) - 1 = 0 \;\; ; \quad [\text{because } d_i d_i = +1]$$

$$d_i\left(\sum_{j=1}^{m}u_j d_j\left(x_j^T x_i+\frac{\delta_{ij}}{C}\right)-\gamma\right)-1=0$$

$$\sum_{j=1}^{m}u_j d_j\left(x_j^T x_i+\frac{\delta_{ij}}{C}\right)-\gamma=\frac{1}{d_i}=d_i$$

$$\gamma=\sum_{j=1}^{m}u_j d_j\left(x_j^T x_i+\frac{\delta_{ij}}{C}\right)-d_i \qquad (2.44)$$

Thus γ can be obtained by taking a point whose $u_i > 0$ **(**which corresponds to a support vector) and by using the above formula. The average of γ may be estimated from each such support vectors. The above quadratic programming problem yields $u_i s$ and the decision function is given by

$$f(\boldsymbol{x}) = \text{sign}(\mathbf{w}^{\mathrm{T}}\mathbf{x}-\gamma)=\text{sign}(\sum_{i=1}^{m}u_i d_i \mathbf{x}_i{}^{\mathrm{T}}\mathbf{x}-\gamma) \qquad (2.45)$$

If the set of index of support vectors (or data points whose $u_i \neq 0$) is denoted as svindexthen $f(\boldsymbol{x})$ can be written as $\text{sign}(\sum_{i\in svindex} u_i d_i \boldsymbol{x}_i{}^{\mathrm{T}}\boldsymbol{x}-\gamma)$ (2.46)

2.2.5 Non-linear SVM and Kernel Trick

It can be noted that the solution of SVM finally requires Q, d and C where Q is obtained from $\boldsymbol{A}*\boldsymbol{A}^T$ and $\boldsymbol{d}*\boldsymbol{d}^T$.

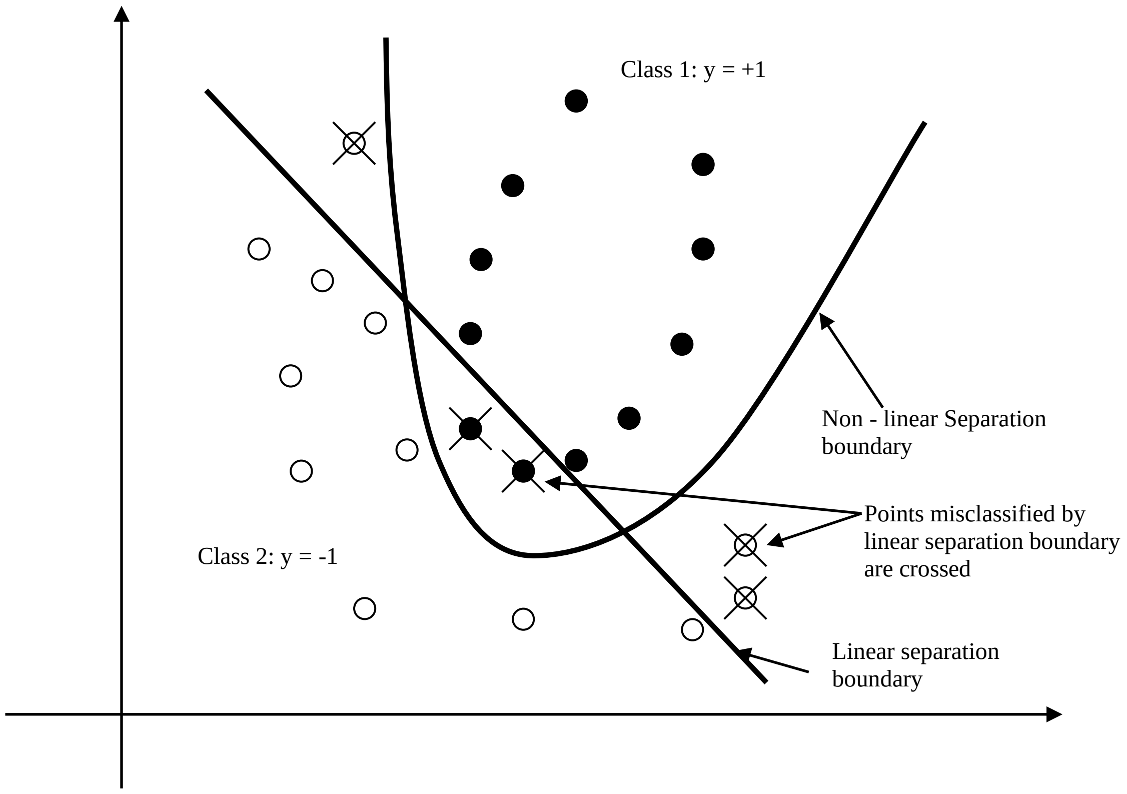
$$\boldsymbol{AA}^T=\begin{bmatrix} \boldsymbol{x}_1{}^T\boldsymbol{x}_1 & \boldsymbol{x}_1{}^T\boldsymbol{x}_2 & . & \boldsymbol{x}_1{}^T\boldsymbol{x}_m \\ . & . & \boldsymbol{x}_i{}^T\boldsymbol{x}_j & . \\ . & . & . & . \\ \boldsymbol{x}_m{}^T\boldsymbol{x}_1 & \boldsymbol{x}_m{}^T\boldsymbol{x}_2 & & \boldsymbol{x}_m{}^T\boldsymbol{x}_m \end{bmatrix}=\boldsymbol{K} \qquad (2.47)$$

Consider the (i,j)th element of $\boldsymbol{AA}^T$ is $\boldsymbol{x}_i{}^T\boldsymbol{x}_j$. This representation of matrix is called as linear kernel matrix. That is, required information would be gathered from the training process by utilizing dot products between training vectors. In case of non linear separable data as shown in Fig., 2.6, mapping of data x_i can be done by using the function $\phi(.)$ which is performed on high dimensional space. This is used to identify the maximally separating hyperplane for the corresponding classifier.

Here SVM training phase needs values for measuring $\phi(\boldsymbol{x}_i)^T\phi(\boldsymbol{x}_j)$ for all i and j for all i and j which requires more computation overhead for the measurement values of mapping data points in the high dimensional space. This is applied by integrating the kernel function which doesn't require dot product values for the mapping data to high dimensional space [47]. The merits of non-linear SVM classifier are given as follows:

The training algorithm is the same as that of the linear classifier.

- The problem of 'curse of dimensionality' is tackled in a simple way.



Class 1: y = +1
Non - linear Separation boundary
Points misclassified by linear separation boundary are crossed
Class 2: y = -1
Linear separation boundary

Fig. 2.6 Data which Require Non Linear Classifier

The kernel function K (x_i, x_j) is input value function which is mainly used to avoid the mapping $\phi(\boldsymbol{x})$ process. The kernel $\boldsymbol{K}(\boldsymbol{x}_i, \boldsymbol{x}_j)$ measurement leads to direct computation of scalar products from the input feature space $\phi(x_i)^T \phi(x_j)$ to train the input data vectors. Likewise we can process the high dimensional feature space F in the efficient way. The infinite dimension space for the SVM can be built by selecting the more suitable value of kernel $K(x_i, x_j)$. The kernel function K representation can be given as:

$$K(x_i, x_j) = \phi(x_i)^T \phi(x_j) \qquad (2.48)$$

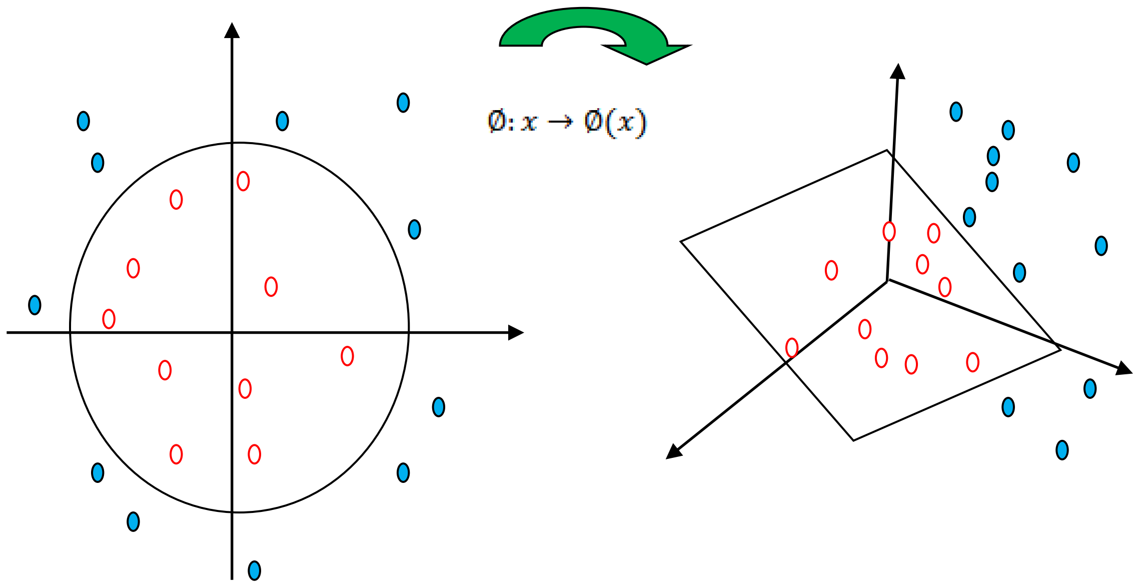


Fig. 2.7 Non-Linear Mapping into Feature Space

There are more traditional Kernel functions are available in real time namely linear, polynomial, Radial Basis Function (RBF) [48] [49]. Among this linear kernel found to be most popular and simpler kernel function which is defined as $K(x_i, x_j) = (x_i)^T (x_j)$.

The format of polynomial kernel is given as $K(x, y) = (x^T y + 1)^d$

The format of RBF kernel is represented as $k(x, y) = \exp(-\sigma \|x - y\|^2)$ where σ is a positive parameter controlling the radius.

The Mercer's condition is adapted by the kernel function to define that positive semi-definite kernel value k (x, y) is declared as dot product in a high dimensional space [50] - [52].

Normalization of Kernels

The support vector training process becomes more complex with increased size of number of variables which would lead to increased/decreased kernel size. This can be resolved by normalizing the value of kernel and the variables. The normalization of variables can be done between the range [0, 1] or [-3, +3] by applying the formula $\left(x_i - \mu_i\right)/\sigma_i$, where μ_i and σ_i are the mean and standard deviation of the i^th^ predictor variable. There would be presence of n number of kernels for the n number of variables which are normalized in the range of [0, 1]. The normalized variable value 1, it would lead to maximal value generation which forms polynomial kernel representation as $\left(n+1\right)^p$. Here the normalized kernel value is defined as $\mathrm{k}(\boldsymbol{x},\mathrm{y}) = \left(\frac{\boldsymbol{x}^T\mathrm{y}+1}{n+1}\right)^p$, where p is defined as degree of polynomial kernel. The maximum value of RBF kernel for n number of variable is $\|\boldsymbol{x} - \boldsymbol{y}\|^2$. And the normalized kernel is measured as $\mathrm{k}(\boldsymbol{x},\boldsymbol{y}) = exp(-\frac{\sigma}{n}\|\boldsymbol{x} - \boldsymbol{y}\|^2)$.

2.2.6. SVM Formulation of Non-Linear Kernels with Soft Margin - L1 Norm

The formulation can also be written as:

$$\begin{aligned}
&\min_{\mathbf{w},\gamma,\xi} \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{m}\xi_i \\
&\text{subject to } d_i(\mathbf{w}^T\phi(\mathbf{x}_i) - \gamma) + \xi_i - 1 \geq 0, \quad 1 \leq i \leq m \\
&\qquad\qquad\qquad\qquad\qquad \xi_i \geq 0, \quad 1 \leq i \leq m
\end{aligned} \tag{2.49}$$

The Lagrangian is

$$\begin{aligned}
L(\mathbf{w},\gamma,\boldsymbol{\xi},\boldsymbol{u},\boldsymbol{\mu}) &= \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{m}\xi_i - \sum_{i=1}^{m}u_i\left[d_i(\mathbf{w}^T\phi(\mathbf{x}_i) - \gamma) + \xi_i - 1\right] - \sum_{i=1}^{m}\mu_i\xi_i \\
&= \frac{1}{2}\mathbf{w}^T\mathbf{w} + \sum_{i=1}^{m}(C - u_i - \mu_i)\xi_i - \left(\sum_{i=1}^{m}u_i d_i\phi(\mathbf{x}_i^T)\right)\mathbf{w} - \left(\sum_{i=1}^{m}u_i d_i\right)\gamma + \sum_{i=1}^{m}u_i
\end{aligned} \tag{2.50}$$

where $\boldsymbol{u},\boldsymbol{\mu}$ are the Lagrange multipliers and the Wolfe dual problem are:

$$\max_{\mathbf{u},\boldsymbol{\mu}} L(\mathbf{w},\gamma,\boldsymbol{\xi},\boldsymbol{u},\boldsymbol{\mu})$$
$$\begin{aligned} \text{subject to } & \frac{\partial L}{\partial \mathbf{w}} = \mathbf{0} \\ & \frac{\partial L}{\partial \gamma} = 0 \\ & \frac{\partial L}{\partial \xi_i} = 0 \qquad 1 \le i \le m \\ & \boldsymbol{u} \ge \mathbf{0} \\ & \boldsymbol{\mu} \ge \mathbf{0} \end{aligned} \tag{2.51}$$

The constraint $\frac{\partial L}{\partial \mathbf{w}} = 0$ implies:

$$\mathbf{w}^T - \sum_{i=1}^{m} u_i d_i \phi(\mathbf{x}_i{}^T) = 0 \tag{2.52}$$

Or, equivalently,

$$\mathbf{w} = \sum_{i=1}^{m} u_i d_i \phi(\mathbf{x}_i) \tag{2.53}$$

The constraint $\frac{\partial L}{\partial \gamma} = 0$ implies:

$$\sum_{i=1}^{m} \alpha_i d_i = 0 \tag{2.54}$$

The constraints $\frac{\partial L}{\partial \xi_i} = 0$ imply:

$C - u_i - \mu_i = 0, \qquad 1 \le i \le m.$

Where $\boldsymbol{u} \ge \mathbf{0}$, $\mu \ge \mathbf{0}$, $0 \le u_i \le C$

Substituting these results into $L(\mathbf{w},\gamma,\boldsymbol{\xi},\boldsymbol{u},\boldsymbol{\mu})$:

$$L(\mathbf{w},\gamma,\boldsymbol{\xi},\boldsymbol{u},\boldsymbol{\mu}) = \frac{1}{2}\mathbf{w}^T\mathbf{w} + \sum_{i=1}^{m} 0\times\xi_i - \mathbf{w}^T\mathbf{w} - 0\times\gamma + \sum_{i=1}^{m} u_i$$

$$= -\frac{1}{2}\mathbf{w}^T\mathbf{w} + \sum_{i=1}^{m} u_i \qquad (2.55)$$

$$= \sum_{i=1}^{m} u_i - \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m} d_i d_j \phi(\mathbf{x}_i^T)\phi(\mathbf{x}_j) u_i u_j$$

Then the dual problem is:

$$\max_{\boldsymbol{u}} L(\boldsymbol{u}) = \sum_{i=1}^{m} u_i - \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m} d_i d_j \phi(\mathbf{x}_i^T)\phi(\mathbf{x}_j) u_i u_j$$

$$\text{subject to } \sum_{i=1}^{m} d_i u_i = 0 \qquad (2.56)$$

$$0 \le u_i \le C \qquad 1 \le i \le m$$

In matrix form it can be written as:

$$\text{Minimize } \mathrm{L_D}(\mathbf{u}) = \frac{1}{2}\mathbf{u}^{\mathrm{T}}\boldsymbol{Q}\mathbf{u} - \mathbf{e}^{\mathrm{T}}\mathbf{u} \qquad (2.57)$$

$\mathbf{d}^{\mathrm{T}}\mathbf{u} = 0, \mathbf{0} \le \boldsymbol{u} \le C\boldsymbol{e}$

where $\boldsymbol{Q} = \boldsymbol{DKD}$ and K is kernel matrix.

$\boldsymbol{Q}$ can be computed as $\boldsymbol{Q} = \mathrm{K} \bullet *(\boldsymbol{d} * \boldsymbol{d}^{\mathrm{T}})$

2.2.7 Multi-Class Support Vector Machines

If amount of classes is higher than two, then the problem is known as multiclass SVM. Two different types of methods are available for multiclass SVM. In first method which is said to be as indirect method where various binary SVM's has been constructed and classifier's output is made to combine in the sense of finding final class. In second approach namely direct method in which single optimization formulation has been considered [47].

Indirect Methods of Multiclass SVM

Three approaches are comes under this category such as One against all which means one versus rest, One against one which means pair-wise and DAGSVM which means Directed Acyclic Graph SVM. These approaches mentioned above are simple in nature and highly used, since it does not cause any of numerical difficulties during training.

One against all

This method used for every class as a binary SVM classifier gets constructed, discrimination of data points in class against the one remain. If N classes are there, N binary SVMs are made to build. During testing process, every classifier would yields a decision value for test data point and classifier possessing highest positive decision value which assigns their label to data point. The comparison has been made between decision values which are produced by various SVMs that are still valid due to the presence of training parameters and dataset who remains as same. The testing process for a new data has been shown diagrammatically in the Fig. 2.8. In this three classes are there such as, H, E and C. Thus three SVM classifiers are there to which new data has subjected to testing.

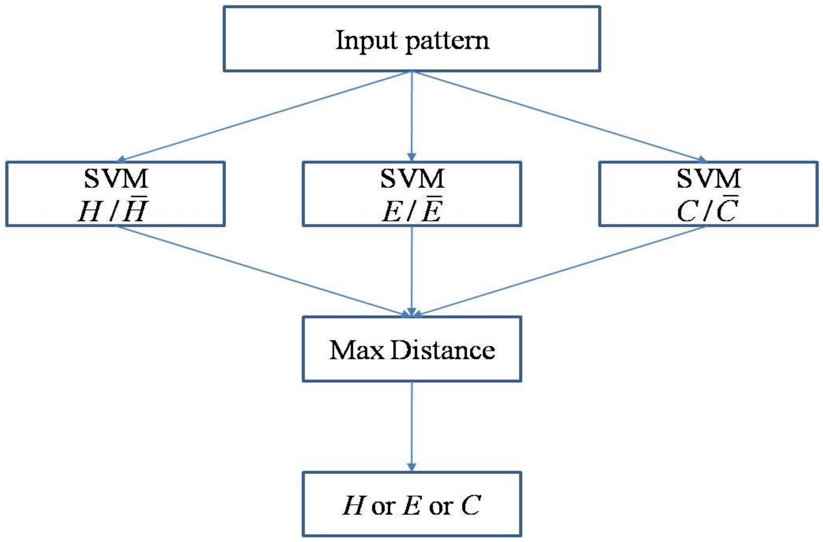


Fig. 2.8 Prediction using One Against All SVM

New data has been assigned for class label that depends on decision rule which follows

$$\arg\max_{k\in\{H,E,C\}} \left\{ \boldsymbol{w}_H^T\phi(\boldsymbol{x})-\gamma_H,\ \boldsymbol{w}_E^T\phi(\boldsymbol{x})-\gamma_E,\ \boldsymbol{w}_C^T\phi(\boldsymbol{x})-\gamma_C \right\} \qquad (2.58)$$

One against one

Another one is indirect way of extending the SVMs in the aim of solving the multiclass problems using one against one method. In such a case, for N classes, the N (N-1)/2 classifiers are made, one for the every pair of classes. Prediction of the class for new data point is depending on voting scheme. Fig 2.9 shows up the architecture of prediction scheme for specifically three classes that considered earlier.

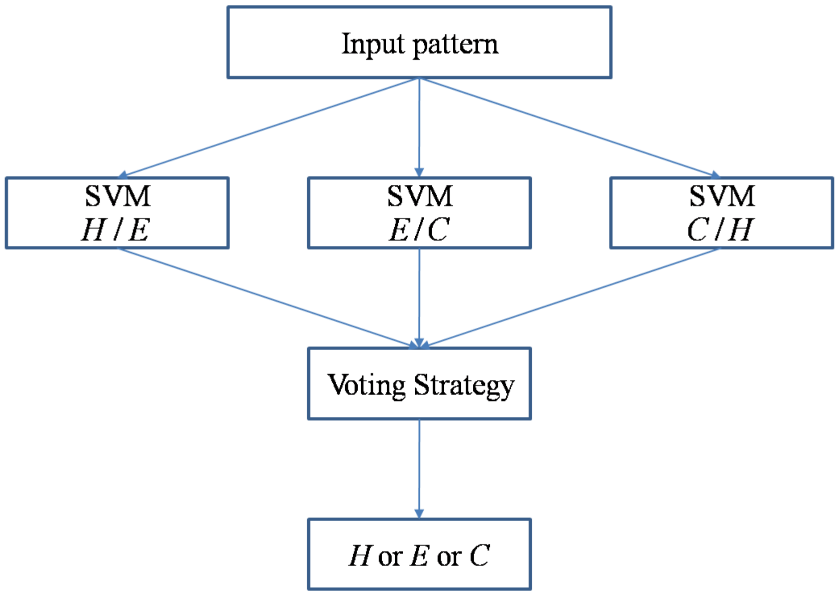


Fig. 2.9 Prediction using One Against One SVM

Directed Acyclic Graph SVM

In this type, the training phase is similar as one against one method. But for the testing, it has been used as directed acyclic graph [53]. Each node is act as binary classifier. For three classes namely H, E, C could be considered in above and prediction structure has shown in Fig. 2.10. Test data has been given, starting from root node, binary decision function of classifier H/E gets evaluated. Then node is exited at left edge if they do not belonging to H or the right edge if it does not belong to E. Next node's decision function is then made to evaluate. Finally input reaches the leaf node which indicates class label of input data. The benefits of using DAG is some analysis over generalization could establish [53].

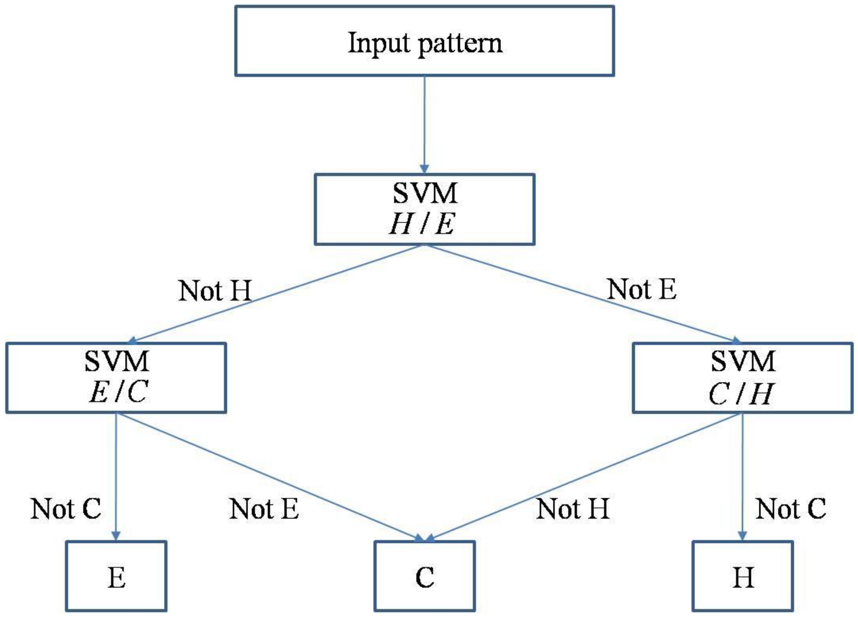


Fig. 2.10 Prediction using DAGSVM

Direct Methods of Multiclass SVM

Basically, two direct approaches are presented in [46] [54] [55]. These methods try to attempting the finding of separate boundaries for all classes in only one step. For every class, the methods could define the decision rule which is similar to binary SVMs and while during testing process of test data point has been assigned to the label of decision rule which yields the largest namely positive margin. The formulation for two popular approaches is mentioned below.

Vapnik and Weston Method

This approach is same as one – against - all approach. If N classes are present, then it constructs the N two-class rules where kth discriminant function $w_k^T \phi(x) - \gamma_k$ could separates the training vectors of class k from other vectors. The primal formulation of that multiclass SVM is written as

Minimize $\frac{1}{2}\sum_{k=1}^{N} w_k^T w_k + C\sum_{i=1}^{m}\sum_{k \neq k_i} \xi_k^i$ (2.59)

Subject to the constraints

$$\begin{aligned} &\boldsymbol{w}_{k_i}^T \phi(\boldsymbol{x}_i) - \gamma_{k_i} \geq \boldsymbol{w}_k^T \phi(\boldsymbol{x}_i) - \gamma_k + 2 - \xi_k^i, \ \forall k \neq k_i \\ &\xi_k^i \geq 0 \ , \forall k \neq k_i \end{aligned} \tag{2.60}$$

Where k_i is the class to which the training data $\boldsymbol{x}_i$ belong.

Decision function for new input data as x_j is by

$$\hat{d}_j = \arg \max_{\mathrm{k}} \left\{ f_k(\boldsymbol{x}_j) \right\}, \tag{2.61}$$

Where $f_k(\boldsymbol{x}_j) = \boldsymbol{w}_k^T \phi(\boldsymbol{x}_j) - \gamma_k$

Crammer and Singer Method

The formulation for this approach [55] is as below

Minimize

$$\frac{1}{2} \sum_{k=1}^{N} \boldsymbol{w}_k^T \boldsymbol{w}_k + C \sum_{i=1}^{n} \xi_i \tag{2.62}$$

Subject to the constraints

$$\boldsymbol{w}_{k_i}^T \phi(\boldsymbol{x}_i) - \boldsymbol{w}_k^T \phi(\boldsymbol{x}_i) \geq e_k^i - \xi^i \quad , \forall \mathrm{k} \neq \mathrm{k_i} \tag{2.63}$$

Where, k_i as class, the training data $\boldsymbol{x}_i$ belonging as, $e_k^i = 1 - c_k^i$,

$$c_k^i = \begin{cases} 1 \text{ if } \mathrm{k_i} = k \\ 0 \text{ if } \mathrm{k_i} \neq k \end{cases} \tag{2.64}$$

The main difference from method of Vapnik and the Weston is this approach has only n slack of variables ξ^i have used and this does not possess coefficients γ_k .

2.3. DEEP LEARNING

Deep learning [56] can also be said as deep structured learning or the hierarchical learning which is a part of broader family of the machine learning methods [57][58]. Most of the deep learning architecture consists of neural networks whereas some uses decision trees and other techniques. But deep learning with neural networks is most powerful with intellectual property of feature learning.

Learning can be supervised, partially supervised or unsupervised. There are different deep learning architectures namely the deep neural networks, the convolutional deep neural networks, the deep belief networks, and the recurrent neural networks. These networks are then applied for fields that including the computer vision, the speech recognition, the natural language processing, the audio recognition, the social network filtering, the machine translation, the bioinformatics and the drug design, are producing results that made to comparable and in some of cases they may superior to the human experts. This research focus towards development of writer identification model using convolutional neural network. A brief description of artificial neural network is presented in section 2.3.1 before elaborating the architecture of CNN.

2.3.1. Artificial Neural Network

Artificial Neural Networks shortly ANNs [57] [58] is a computing systems contains number of neurons like biological neural networks constituting animal brains. Those systems could learn to carry out the tasks by consider the examples, in general without the task-specific programming. They have been found as most using applications that are difficult in expressing a traditional computer approach uses rule-based programming.

It derives computational model stimulate by means of biological neural networks, and used for approximate the functions which are mysterious in general. It works like the behavior neurons and electrical signals that conveys between the input, processing, and the output from brain. The way of neurons could semantically communicate is area where in ongoing research [59] [60]. The majority of Artificial Neural Networks look somewhat like similar neuron which makes more complex biological counterparts, but they are effective at intended tasks.

ANNs process the information and the show some of acquaintance and they could behave the exhibiting intelligence in such way namely pattern recognition, learning and the generalization. ANN is depending on collection of units that are connected namely artificial neurons [61]. Every connection which means synapse that is between neurons which can transmit signal to another neuron. The receiving postsynaptic or neuron could process the signals and then signal downstream neurons which are connected to it. Neurons may state, and in general which are represented by the real numbers, between 0 and 1. Neurons and the synapses might have weight which varies as the learning proceeds, also increase or decrease strength of signal and is sends downstream.

Typically, the neurons are organized in layers. Different layers might perform by various kinds of transformations on inputs. Signals could travel from first input to last output layer, which are possibly after traversing the layers of multiple times. The goal of neural network approach aims to solve the issues in same way of human brain retain. Additional capability of neural network is to match specific mental capabilities, which entirely differ from biology namely back propagation, or passing of information in reverse direction and adjusting network in the sense of reflect the information.

Neural Networks shortly NNs have used for various tasks that includes speech recognition, computer vision, machine translation, , playing board, video games, medical diagnosis and social network filtering. NNs consist of number of neurons which are interconnected with each other. The neurons will act as the basic building blocks in brains that are biological neural networks.

Mathematical Model of the Neuron

Biological neural networks are found as very complex. By Contrast, mathematical model of network is as higher in simplicity and it is depending on several assumptions:

- All of neurons are made to synchronize which means the signal could pass from one neuron to the other neuron that takes similar time for all of the connections. Signal processing is also be synchronized and is same for every neurons.
- Every neuron has transfer function that determines the neuron's output signal which depending on input signal strength. That function is found as time-independent.
- While signal passes synapse, it can change by linearly i.e., signal value has multiplied by means of some number and that number is known as synaptic weight.

The very necessary property of synaptic weight is that can changes by time. This feature makes that possible for the brain to react vary on similar input in the various moments. The assumptions made would simplify the process carry out in biological neural network. For instance the brain signal transmission time could naturally depend on distance between the neurons. But despite those simplifications, the artificial networks still could preserve most necessary characteristics of the biological networks such as adaptability and the ability for learn. The first mathematical approach of the neuron has been introduced as higher than half century ago and but did not change in much. Finally the neuron has seen as straight forward automate that transforms input signals into output signal as shown in Fig. 2.11.

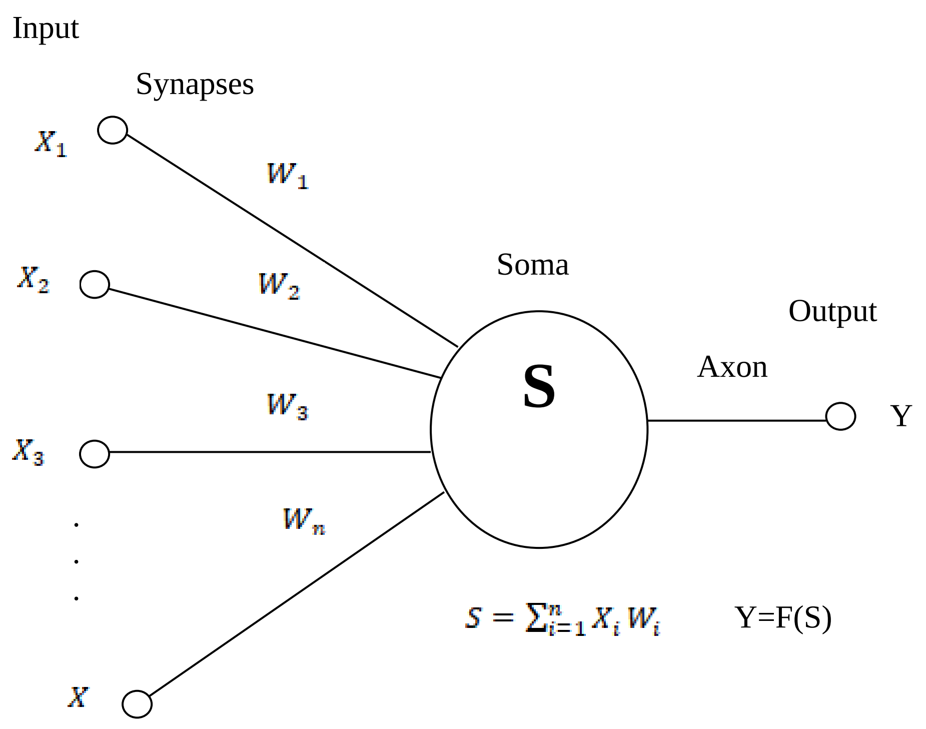


Fig. 2.11 Artificial Neuron Model

The model functions as inputs of neuron's synapses which receive the N signals [X_1, . . . ,X_n]. Now every synapse would makes a linear changes of signal which uses its synaptic weight. Then the neuron's body could receive signals [X_1w_1, . . . ,X_nw_n] thus w_i denotes the synaptic weight and the sums signal(X_i):

$$S = \sum_{i=1}^{n} X_i w_i \tag{2.65}$$

Finally it is applied on given function namely F and that could say as activation function which sends final signals to output.

$$Y = F(S) \tag{2.66}$$

There are various functions which are widely using as activation functions. In network, it is not important for using the similar function for all of neurons. Thus, it is common in practice. In many cases, the activation functions are non-linear in various scenarios. On the other hand, whole network would implement some of linear transformation and that would be equivalent to one artificial neuron - perception only.

Types of Neural Networks

Types of the Neural Networks are Multi-Layer Perceptron (MLP), Back Propagation Network (BPNN), Radial Biases Function (RBF) networks and the Probabilistic Networks. Two commonly used NNs such as MLP and BPNN are presented below.

Multilayer Perceptron

A Multilayer Perceptron is one of the classes for feedforward artificial neural network. An MLP would comprise of at least 3 layers of the nodes. Except for input nodes, every node is neuron which uses the nonlinear activation function. MLP could utilize the supervised learning technique and it is also said as Back Propagation for the training. Its multiple layers and the non-linear activation would distinguish the MLP from specifically a linear perceptron. It can also distinguish the data which is not linearly in separable.

Input neurons - Input neurons are taken as input vector which encodes the some action or the information about external environment. Input neuron does not perform any of computation types and only pass input vector to the neurons that are subsequent.

Output neurons - It would receive the signals from preceding neurons and it transforms using the equations (2.65-2.66) and values obtained represent the output of whole neural network.

Hidden neurons - These neurons are considered as basis of Neural Networks. Those neurons would receive signal from input neurons or preceding hidden neurons, and process is based on accordance of equations (2.65-2.66). Then the resultant signal is passed to subsequent which means hidden or output neurons.

In the multi-layer perceptron neurons, they are dividing in to layers. The Input and the output neurons of separate layer each input and output layer. Hidden neurons form the one or many hidden layers. Each MLP neuron, with exception of the input neurons has been connected through synapses with all the neurons of previous layer. MLP architecture is illustrated below in Fig. 2.12.

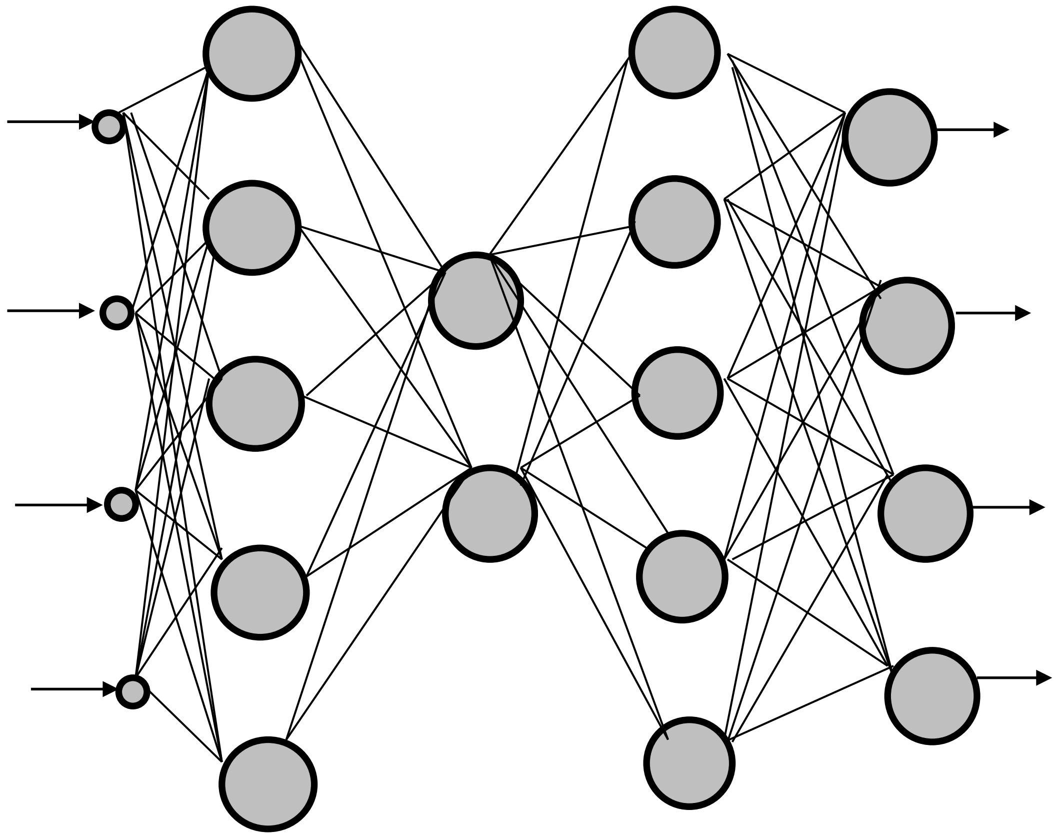


Fig. 2.12 Multi-Layer Perceptron Neural Network

This network would receive the 4-dimensional based input since it possesses 4 input neurons. The result has been represented by means of 4-dimensional output vector. Additionally, it comprises of 3 hidden layers. In beginning of MLP's learning, the network's weights are made to initialize in randomly by the small random values. Then network could

- Processes one by one of every item in training set
- Compares the output with desired output
- Computes the error value

After checking out entire training set, network could modify synaptic weights in the aim of minimizing error. Most of MLP training schemes are basis on gradient descent approach which is iteratively changes the network’s synaptic weights and that would gradually reduce the error on training set.

Back Propagation Neural Network Algorithm

Neural networks are one of the powerful tools helpful to solve the several problems namely prediction or classification. Additionally, in the sense of solve the some specific problem, network must set up and trained properly.

The first step is network architecture selection. In this number of the input and the output neurons, the hidden neurons and the number of layers are comprised in network. If network is expected to producing the boolean values namely 1 or 0 and it found as clear as it should retain one output only. In some cases, no obvious way has been available to determine best number of the hidden neurons by without training the various networks and could estimates the generalization error for all. Many hidden units are lead to the low training error and additionally still they had high generalization error because overfitting. Number of hidden units and the layers could discuss in [62].

The second stage is training by its own. BPNN algorithm belonging to group named supervised learning. Since it had performed depend on labeled trained the network must learn pairs input, the expected output where all together form training set. By contrast some classes of the networks namely Kohonen’s self-organizing maps, which belongs to unsupervised learning group. It also means that training set do not contains of expected outputs for given inputs. BPNN is also described as follows:

Initialization - The first process is of selecting network architecture and setting of initial synaptic weights of it. The most widely using method of synaptic weights initialization has been choosing a small random numbers with zero mean value and the uniform distribution.

Training pair’s presentation - During this phase, the elements of training set are made to pass in the manner of one by one to network. For each training sample, two steps are present namely forward and backward computation could take place.

Forward computation - Assume the training set comprises of the N of pairs (x_i , d_i):

$$T = \{(x_i, d_i)\}, i = 1, \ldots N, x_i \in \mathbb{R}^n, d_i \in \mathbb{R}^m \qquad (2.67)$$

Where, x_i is as an input vector and the d_i as desired output vector. Network passes the inputting vector from the layer to layer and that could sequentially compute outputs for all layer's neurons. The outputs form input vector for next layer. For each neuron namely j in layer represented by l the output such as y_j^l signals are made to computed from previous layer of output signals namely y_j^{l-1} as following,

$$S_j = \sum_{i=1}^{m_l} w_{ij}^l y_i^{l-1} \tag{2.68}$$

$$y_j^l = F(S_j) \tag{2.69}$$

Here m_l as amount of synaptic weights for all neurons of layer l. The number is identified by means of number of the neurons of those previous layers and it is similar for each and every layer's neuron. By initially the neuron would calculate linear combination of the outputs of neurons of previous layer (2.68). The output value has been calculated by means of applying the neuron's activation function to value which are computed in first step (2.69).

For the input layer namely layer 0, the formula is found as simpler:

$$y_j^0 = x_{ij} \tag{2.70}$$

Where, x_{ij} is jth element of input vector namely x_i. After input vector xi would pass all of layers, network would produce the output vector $o_i = [y_1^{out}, \dots, y_m^{out}]$, which is helpful for error signal computation:

$$e_i = d_i - o_i \tag{2.71}$$

Backward computation - While carry out this step the synaptic weights are made to adjust in the intent of producing smaller error value. These adjustments are taken place and derived from error signal. For each and every output unit j calculatesδ_j^{out} :

$$\delta_j^{out} = e_{ij} F^{'}(S_j) \tag{2.72}$$

Here e_{ij} is the j^{th} element of error vector represented as e_i and F' as derivative function for function namely F. For the hidden neuron j of layer l would calculates the δ_j^{hid}:

$$\delta_j^{hid} = F'(S_j) \sum_k \delta_k^{l+1} w_{jk} \tag{2.73}$$

Where summarization is done over k neurons present for following layer, and the w_{jk} is synaptic weight which connects the neurons j and k. When all δ are getting calculated, the synaptic weights are adjusted by means of (2.74) and (2.75):

$$w_{jk}^{t+1} = w_{jk}^{t+1} + \Delta w_{jk}^{t} \tag{2.74}$$

$$\Delta w_{jk}^{t} = \eta \delta_j y_j^{l-1} + \mu\left(w_{jk}^{t} - w_{jk}^{t-1}\right) \tag{2.75}$$

Where η is known as learning rate and y_j^{l-1} is output of neuron j of previous layer and μ is a momentum. Learning rate and the momentum are numbered between 0 and 1. Learning rate is numbering between 0 and 1 which identifies how speedy the neural network that adjusts by itself to patterns in training data. It they do not be constant and could dynamically reduce or enhances the time. The parameter must choose carefully by too small that makes learning process as slow and too high may leads to divergence. Momentum in term influences the way for how previous weights would affects the current one. It also helps algorithm in preventing from being stuck in local minimum. The value should choose carefully and experimentally identified. The usage of momentum could be removed. It enhances the performances of neural networks greatly and are commonly using.

2.3.2. Deep Learning Architecture

Deep learning [63] is a kind of machine learning which achieves the great power and the flexibility by means of learning in the sense of representing the world as a nested hierarchy of the concepts and with every concept is defined in the relation to concepts in simpler, and finally more abstract based representations are computed by less abstract ones.

Deep learning [56] is one of the classes in machine learning algorithms that use a cascading of the multiple layers of the nonlinear processing units for particular feature selection and for transformation. Every successive layer is using the output from previous layer as an input.

Learning has been performed either in the supervised or the unsupervised in manner. Learning the multiple levels in representations that corresponding to various levels of the abstraction. The levels are forming the hierarchy in terms of concepts. The learning could perform via the Back Propagation shortly BP. The example of DNNs has presented in Fig. 2.13.

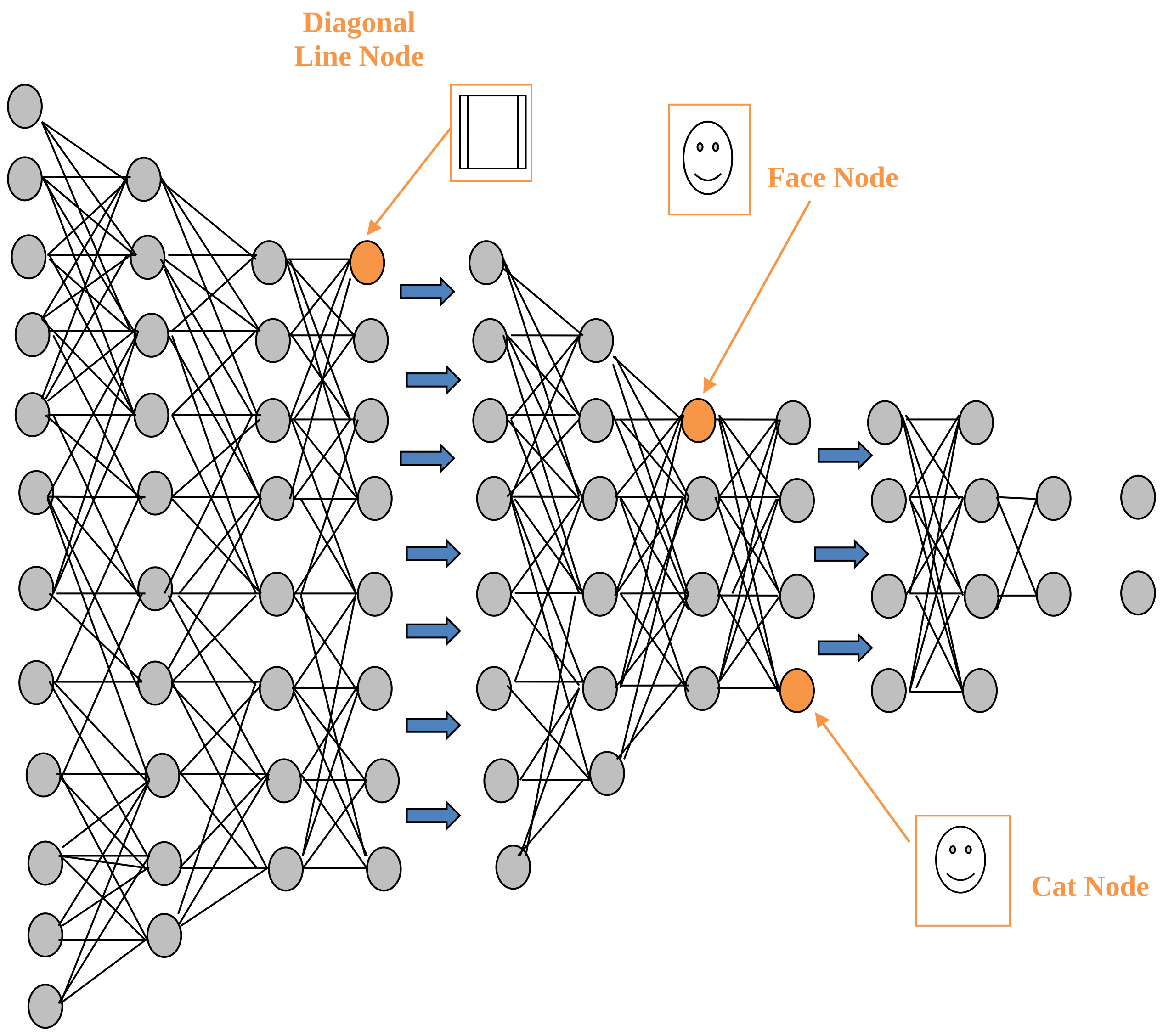


Fig. 2.13 Example of DNNs

Deep learning could automatically find the features using for classification. Deep learning algorithms are tried in learning high-level features from the data. This is very distinctive side of the Deep Learning and major step should ahead of the traditional Machine Learning. Thus, the deep learning would also reduce the task for developing the new feature from extractor for all problems. Comparing with Artificial Neural Network, the Convolutional Neural Network in the deep learning could help to providing promising outputs.

Deep Neural Networks

A Deep Neural Network simply DNN is an ANN with the multiple hidden layers in between input and the output layers [64]. As same as shallow ANNs, the DNNs could model the complex non-linear based relationships. Object has expressed as layered composition of the primitives in DNN architectures will generate the computational models. The additional layers helps to enable the structure of the features from those lower layers, possibly by modeling the multifaceted data with few of units than correspondingly performed shallow network.

Deep architectures may include high variants of few primary approaches. Each of architecture has identifies the success in particular domains. Additionally, this could not possible always for comparing the performance of those multiple architectures present, and they had evaluated on same data sets. DNNs are made to typically feedforward in networks for which the data could flows from input layer to output layer by without loop back.

Convolutional Neural Network

In machine learning, Convolutional Neural Network is a class of deep architecture with deep, feed-forward in Artificial Neural Network which can be applied specifically for visual imagery. CNNs use variation of the Multilayer Perceptron that designed for requiring the minimal preprocessing [65]. It can also be referred as shift invariant or the Space Invariant Artificial Neural Networks represented as SIANN. This is based on shared-weights architecture and the translation invariance characteristics [66].

Convolutional networks is designed based on biological human brain [67] where connectivity pattern between the neurons has inspired by organization of animal visual cortex. Individual cortical neurons are responding to stimuli that only in particular restricted region of visual field called as receptive field. These receptive fields of various neurons are partially overlaps in such way that they could cover entire visual field.

Pre-processing is a task performed in CNNs are very less compared to other image classification algorithms. This means, the network could learns the filters in traditional algorithms of hand-engineered. This independence obtains from prior knowledge and the human effort in the field of feature design as main advantage.

A CNN would consist of an input layer and an output layer, and additionally the multiple hidden layers. The hidden layers are seems to be convolutional, pooling or may fully connected. In general, the architecture of CNN is presented in Fig. 2.14.

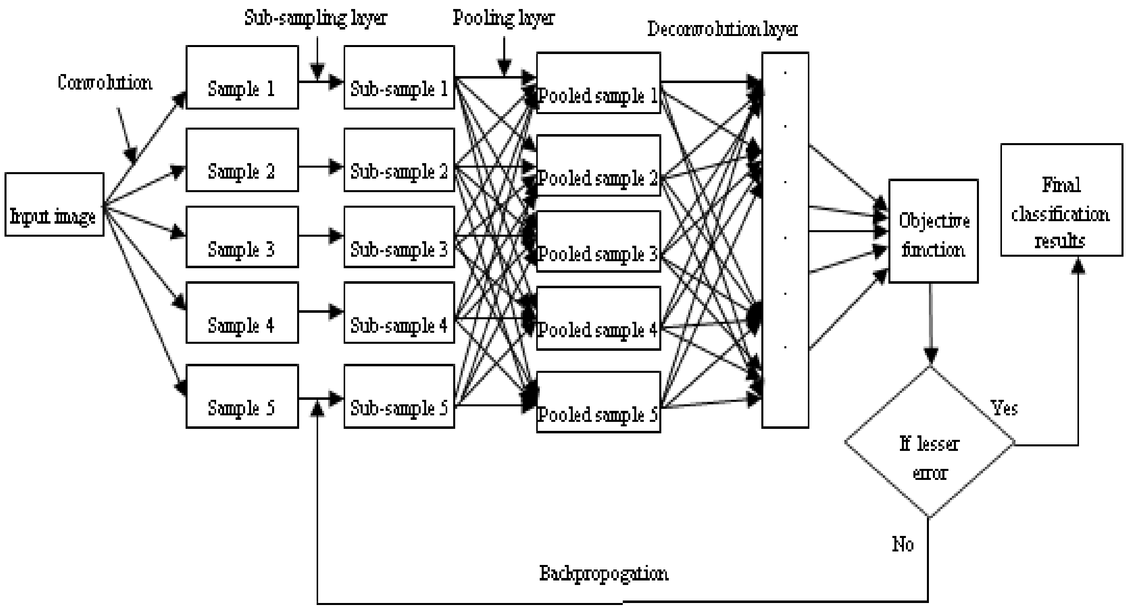


Fig. 2.14 Typical CNN Architecture

Convolutional Layer

The convolutional layer has the building block [68] as human brain, would connect the neurons to activate to receptive field. The parameter of layer's would consist of set of the learnable filters, having little receptive field, but also extends the full depth of input volume. In the forward pass, every filter is convolved across width and the height of input volume that computes the dot product of entries in the filter and the input will produces two-dimensional activation map of those filters. Finally, the network may help to learn the filters which activates when it detects the specific type of the feature at some of spatial position in input.

Stacking activation maps for all of filters along with depth dimension would form full outputting volume of convolution layer. Each and every entry in output volume could be interpreted as an output of the neuron which looks at little region in input and that shares the parameters with the neurons in similar activation map.

The convolutional layer is the basic building block of Convolutional Network which does mostly all computational heavy lifting.

Pooling Layer

Another necessary concept of the CNNs is called pooling, which is in form of non-linear down-sampling. There are various non-linear functions that implement the pooling among which the max pooling is most common [69] [70]. It partitions an input image in to set of non-overlapping rectangles and additionally for sub-region, which outputs the maximum. The instinct is exact location of specific feature is found as less enough than the rough location of it which are relative to various other features.

The pooling layer has its originality to act in every depth slice of the input and it will resize the data. Pooling layer has the filters with size of 2x2 applied with a stride of down samples at ever depth slice in the input by 2 along both width and height which will discard 75% of the activations. It is a common practice by periodically inserting pooling layer between the successive convolutional layers as in the CNN architecture. The pooling operation may provide another form of the translation invariance.

The pooling layer may operate independently on every of depth slice of input and would resize it spatially in manner. The most vital using form is pooling layer with the filters of size such as 2x2 applied with stride of the 2 down samples at each depth slice in input by means of 2 along with width and height that discarding 75% of activations. In such case, the each max operation has been over for 4 numbers. Then the depth dimension is remains unchanged.

The pooling layer may consist of following steps:

- Accepts volume of the size $W_1 \times H_1 \times D_1 \mathrm{W}_1 \times \mathrm{H}_1 \times \mathrm{D}_1$

- Requires 2 hyper parameters:
  - their spatial extent the *F*F,
  - the stride namely *S*S,
- Produces volume of the size *W*2$W_2 \times H_2 \times D_2$ thus:

  $W_2 = (W_1 - F)/S + 1$

  $H_2 = (H_1 - F)/S + 1$

  $D_2 = D1$
- Introduces the zero parameters and it computes a fixed function of input
- Note it is not usual for using zero-padding for the Pooling layers

In addition to the max pooling, pooling units could use the other functions namely average pooling or the L2-norm pooling. In historic information pooling layer plays a major role but had recently fallen out of the favor that compares for max pooling that works better by practice.

ReLU layer

ReLU stands for Rectified Linear Units. This layer may applies non-saturating activation function represented as f(x) = max (0, x). This will helps to improve the nonlinear properties of the decision function with overall network without affecting the fields corresponding to convolutional layer.

Other functions are used for increasing nonlinearity, for instance, saturating hyperbolic tangent f(x) = tanh (x), f(x) = |tanh(x)|, and sigmoid function $f(x) = \left(1 + e^{\{-x\}}\right)^{-1}\}$ ReLU is to prefer for other functions, because they trains neural network in several times speedy without significant penalty to the generalization accuracy.

Fully connected layer

After various convolutional and the max pooling layers, high-level reasoning in neural network has done through fully connected layers [71]. Fully connected layer contains neurons which have connections to all previous layers like traditional neurons. Their activations are been computed with the matrix multiplication that are followed by bias offset.

Loss layer

The loss layer might specify that how the training could penalizes the deviation between predicted and the true labels and has normally final layer. Various loss functions are appropriate for the various tasks which may uses. Softmax loss is useful for predicting the single class of the K mutually exclusive classes. Sigmoid cross-entropy loss helps in predicting K independent probability values in the [0, 1]. Euclidean loss is for regressing to real-valued labels.

2.4. SUMMARY

This chapter presents the description of supervised learning and deep learning algorithms which are used for implementing writer identification models. This chapter also presents the detailed description of mathematical modeling of Support Vector Machine with linear and non -linear kernels. The working principal of Convolutional Neural Network has been presented in this chapter. The new form of linear kernels with parameters estimated through parameter estimation methods are explained in subsequent chapters.