Chapter 4
Support Vector Machine

Section I: Theory

4.1 Introduction

The classical regression and Bayesian classification statistical techniques stand upon a strict assumption that the underlying probability distribution is known. However, in real life, oftentimes we are confronted with distribution-free regression or classification tasks with only recorded training patterns which are high-dimensional and empty in nature.

Support vector machine (SVM) is one of the relatively new and promising methods for learning separating functions in pattern recognition (classification) tasks, or for performing function estimation in regression problems. SVMs were originated from the statistical learning theory (SLT) by Vapnik (Vapnik 1995) for 'distribution-free learning from data'.

4.1.1 History and Background

Vladimir Vapnik and Alexey Chervonenkis developed the Vapnik-Chervonenkis theory (also known as VC theory) during 1960–1990 (Vapnik and Chervonenkis 1971). VC theory is a computational learning theory related to the statistical learning theory from distribution-free data. This was the basis and starting point for the support vector machine (SVM), a supervised learning technique for classification and regression. Although Vapnik (Vapnik 1995, 1998) introduced the subjects of linear classifier and optimal separating hyperplanes in the 1960s, it did not receive much attention until the nineties. Bernhard Boser, Isabelle Guyon and Vapnik (Boser et al. 1992) introduced a way to create nonlinear classifiers by applying the kernel trick to maximum-margin hyperplanes. The kernel trick is a method for converting a linear classifier algorithm into a nonlinear one by using a nonlinear function to map the original observations into a higher-dimensional space, originally proposed by Aizerman (Aizerman et al. 1964). SVMs have been successfully applied in various classification and machine learning applications. In 1996, Vapnik, Harris Drucker,

Chris Burges, Linda Kaufman and Alex Smola proposed a version of SVM for utilization in the regression tasks (Drucker et al. 1996). This is popularly known as support vector regression (SVR). SVM is an expanding field in the machine learning field. SVM incorporates different ideas like the VC-theory, statistical learning, maximum margin optimal hyperplane, kernel tricks and so on. This makes it particularly powerful over the traditional empirical risk minimization-based methods like the neural networks, etc, as we shall see later.

4.1.2 Applications

Since the nineties, support vector machines have been widely used in different machine learning, particularly classification as well as regression tasks. SVM is a growing field with more and more different kinds of application being explored. Some of these are mentioned below.

- Image processing
  - Object recognition
  - 3D object recognition
  - Face, facial expression recognition
  - Hand-written character recognition
  - Content-based image retrieval
  - Image clustering
- Speech processing
  - Speaker identification/verification
  - Speech recognition, synthesis
- Time-series prediction
  - Financial/currency prediction
  - Weather forecast
  - Load forecast
  - Marketing database
- Data mining
  - Data classification
  - Decision tree modeling
  - Text classification
  - Feature extraction
  - Noise classification and reduction
  - Signal conditioning
- Power systems
  - Fault classification
  - Disturbance analysis
  - Load forecasting

- Process Control
  - Density estimation
  - Generalized predictive control
  - Model predictive control
  - ANOVA decomposition
- Automotive
  - Combustion engine knock detection
  - Engine misfire detection
- Security
  - Intrusion detection
- Bioinformatics
  - Protein structure prediction
  - Breast cancer prognosis
  - Detection of remote protein homologies
  - Classification of microarray gene expression data
- Environmental sciences
  - Geo(spatial) and spatio-temporal environmental data analysis
  - Seismology: modeling of seismic liquefaction potential
  - Determining layered structure of earth
- High-energy Physics
  - Particle and quark flavor identification
  - Charmed quark detection
- Chaos Theory
  - Dynamic reconstruction of chaotic systems
  - Forecasting using chaos theory-based model.

4.1.3 Pros and Cons

Any new field comes with certain advantages and disadvantages. We highlight some specific advantages and disadvantages associated with the SVMs.

- Pros
  - Solution for nonlinear and unknown systems
  - Distribution-free learning from data
  - No overfitting, robust to noise
  - Better convergence guarantee
  - Very effective for sparse, high-dimensional data.

- Cons
  - Binary classification. For multi-class classifications, pair-wise classifications can be used (one class against all others, for all classes) which might not be optimum.
  - Computationally expensive, large memory requirement, runs slow
  - Training might be slow.

4.2 Basics about Statistical Learning Theory

Support vector machines are learning algorithm based on the statistical learning theory, overcoming certain typical problems like the overfitting (see Sect. 3.4.9.1) in machine learning. Therefore, before going into the details of support vector machine, it will be worthwhile to have a short look at the statistical learning theory and the associated topics.

4.2.1 Machine Learning & Associated Problem

Machine learning, in its simplest meaning, means mapping the input-output relationship from the observed data pairs (input-output) with the hope that this learned mapping would deduce the system response for unknown condition (unknown input data). However, the biggest challenge in such a training operation is the trade off between the accuracy and the generalization. This can be seen as having a learning machine which attains a particular *accuracy* level for particular training sets, with certain *capacity* to be able to learn any other (probably unseen) training set without error. Thus, the two main things to notice are the accuracy and the capacity.

Traditional neural network approaches have had difficulties in attaining appropriate generalization capacity in spite of achieving accuracy. Later, we will discuss in details about the difficulties of neural networks in a comparative manner over other approaches. Nevertheless, the goal in machine learning is to choose a model (input-output mapping) from the hypothesis space, which is closest (with respect to some error measure) to the underlying function in the target space (Gunn 1998). A model with too much capacity is overtrained (also referred as overfitted), while a model with very less capacity is undertrained or underfitted (see Chap. 3, Sect. 3.4.9 in context to the neural network). An overtrained model is like a hypothetical dog trainer who trained so many German shepherds (alsatian) that when presented with a greyhound, he concluded that it was not a dog because it did not match his known mapping for canine characteristics! In other way, an undertrained dog trainer when presented with a cat accepts it as a dog because it has four legs! This is the striking point of balancing the accuracy and the generalization in a learning process. Figure 4.1 depicts the over- and underfitting characteristics. Different fittings are shown in Fig. 4.1 for the two classes of square and circular points. A test square

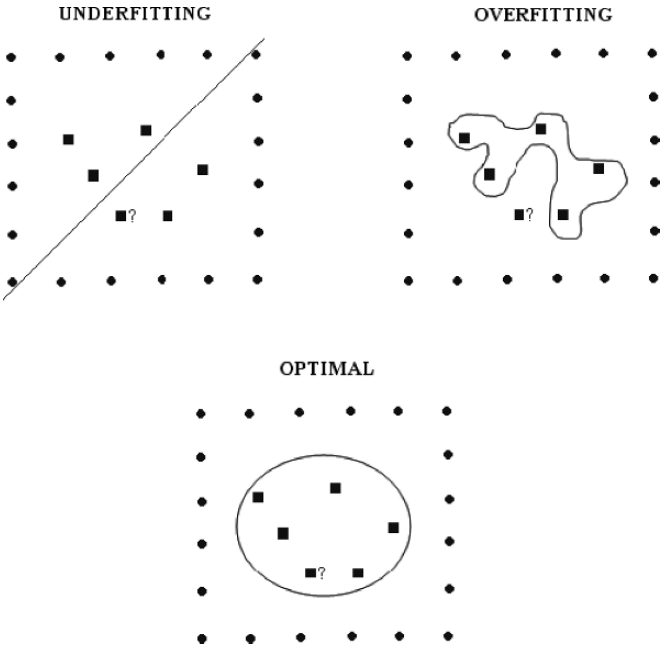

Fig. 4.1 Example of over- and underfitting

point (with a question mark) is checked for correct classification. For the top two plots (under- and overfitting case), the test point is incorrectly classified while the optimal fitting gives the correct classification. This indicates that neither the under- nor overfitting is a desired option. The theory of statistical learning takes on from here.

4.2.2 Statistical Learning Theory

Statistical learning theory (also known as SLT) is primarily concerned about the relation between the capacity of a learning machine and its performance (Burges 1998). Suppose we are given n observations, each consisting of an input-output data pairs $\{x_i, y_i, i = 1, 2, \ldots, n\}$, where x_i's are the input data and y_i's are the corresponding 'true' output data (from a trusted source). In a deterministic learning process, the machine can be defined to achieve the functional mapping $x_i \mapsto f(x_i, \alpha)$, governed by the adjustable learning parameter(s) α. For example, in case of a neural network, α corresponds to the weights and biases, for fuzzy logic-based modeling α corresponds to the fuzzy variables and rules, and so on.

Regarding these input-output data pairs, we also assume that there exists some unknown probability distribution $P(x, y)$, with probability density function (pdf) $p(x, y)$, from which these data are drawn independently and identically distributed (i.i.d). Then, the expectation of the test error or risk (R) can be cited in terms of the

learning parameter α as

$$R(\alpha) = \int_{x,y} \tfrac{1}{2}|y_i - f(x_i, \alpha)\,|\,p(x, y)dxdy. \tag{4.1}$$

However, without knowing the probability distribution and the pdf, we cannot compute the expected risk in (4.1). And this is commonly found in practice. The alternative is to compute the empirical risk (R_{emp}) which is nothing but the average error rate on the training data pairs,

$$R_{emp}(\alpha) = \frac{1}{2n}\sum_{i=1}^{n}|y_i - f(x_i, \alpha)|. \tag{4.2}$$

It is evident from (4.2) that R_{emp} does not depend on the underlying probability distribution, and is a fixed number for a particular choice of α and training set $\{x_i, y_i, i = 1, 2, \ldots, n\}$ (Burges 1998). The quantity on the right hand side of (4.2) is referred to as *loss* which depends on the range of the output.

Let us choose some parameter η such that $0 \leq \eta \leq 1$. It is to be noted that with proper scaling it is easily possible to associate the parameter η with the loss (in ideal case i.e., for no error $\eta = 0$, while in worst case, $\eta = 1$). For losses following the parameter η, Vapnik proved that the following bound (Vapnik 1995) holds with probability $1 - \eta$.

$$R(\alpha) \leq R_{emp}(\alpha) + \sqrt{\left(\frac{h(\log(2n/h) + 1) - \log(\eta/4)}{n}\right)}, \tag{4.3}$$

where, h a nonnegative integer, is called the Vapnik Chervonenkis (VC) dimension (Vapnik and Chervonenkis 1971). VC dimension is a quantitative measure of the capacity of the learning machine.

It can be easily noted from the expression of the VC bound in (4.3) that it does not depend on the probability distribution $P(x, y)$. Also, it is noticeable that knowing the VC dimension h, we can compute the second term in the right hand side of (4.3), enabling us to compute the expected risk of the left hand side, which otherwise is not usually computable.

From (4.3), we can intuitively see that we would like to have a learning process with the minimal expected risk (left hand side of (4.3)). From the right hand side of (4.3), we can interpret this as choosing a learning machine (having particular empirical risk) which gives the lowest upper bound (Burges 1998). This way, we have a structure for the risk and we try to minimize it without knowing the underlying probability distribution. This refers to the structural risk minimization (SRM) principle. Before going into the details of SRM, let's see something about the all important VC dimension.

4.2.3 Vapnik Chervonenkis (VC) Dimension

As mentioned before, the VC dimension is a scalar value that measures the capacity of a set of functions. Before discussing about the VC dimension, it is necessary to understand the concept of shattering of points. If a given set of n points can be categorized in all 2^n possible ways, and for each categorization, we can find a member of the set of functions which perfectly separates the set of points, we say that the set of points is shattered by the set of functions.

We see an example in Fig. 4.2. In Fig. 4.2, we see a set of three points in two dimensional space ($\mathbf{R}^2$). The separating function is a straight line. For all the different combinations, we see that any member of the set of function can separate out 2 points (say class 1) from the remaining 1 point.

The VC dimension h of a set of functions $\{f(x, \alpha)\}$ is defined as the maximum number of training points that can be shattered by the members of the set of functions (Vapnik 1995; Burges 1998). Therefore, for the example shown in Fig. 4.2, the VC dimension of the set of oriented lines in $\mathbf{R}^2$ is $3(= 2 + 1)$. The separating function in $\mathbf{R}^2$ is the set of oriented lines. In general, in an n-dimensional space $\mathbf{R}^n$, the separating functions would be set of oriented hyperplanes. And it can be shown that the VC dimension of the set of the hyperplanes in $\mathbf{R}^n$ is $n + 1$ (Burges 1998). Please note that for the $\mathbf{R}^2$ example above, we wrote the VC dimension as $2 + 1$, conforming to this general relation.

The VC dimension is an indicator of the (shattering) capacity of the set of functions, not only for the indicator functions like the one shown in Fig. 4.2, but also for real-valued functions. The VC dimension of a class of real-valued functions $\{Q(x, \alpha)\}$ is defined to be the VC dimension of the indicator class $\{I(Q(x, \alpha) - \beta > 0)\}$, where β takes values over the range of Q (Hastie et al. 2001). Figure 4.3 shows an example of the VC dimension for real-valued function.

From the definition of the VC dimension, it might appear that the VC dimension for learning machines (functions) with many parameters is higher than learning machines with few parameters. However, there are examples of univariate functions with single parameter having infinite VC dimension, i.e., the family of classifiers capable of shattering l points, no matter how large l (Vapnik 1995; Burges 1998). One classic example is the sinusoid, $f(x, \omega) = I(\sin \omega x > 0)$, the corresponding VC dimension $h = \infty$. Figure 4.4 shows the example of the infinite VC dimension

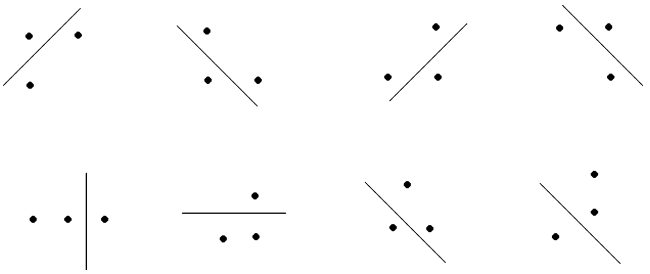

Fig. 4.2 Shattering of points in a two-dimensional space

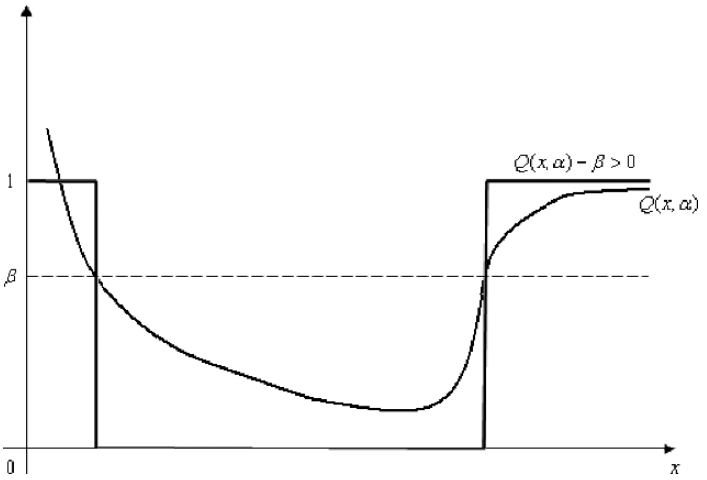

Fig. 4.3 VC dimension for real-valued functions

of the sinusoidal function which can shatter an arbitrarily large number of points by choosing an appropriately high frequency ω, the only parameter.

We can compute the VC bound or confidence (second term on the right hand side of (4.3)) knowing the VC dimension h. Figure 4.5 shows the plot of the VC confidence against the VC dimension h, which is a monotonically increasing function (Burges 1998). From (4.3) and the nature of the curve in Fig. 4.5, the following inference can be cited. For any nonzero empirical risk, the optimal learning machine would have associated set of functions with minimal VC dimension. This ensures minimization of the right hand side of (4.3), leading to a better upper bound on the actual error (Burges 1998).

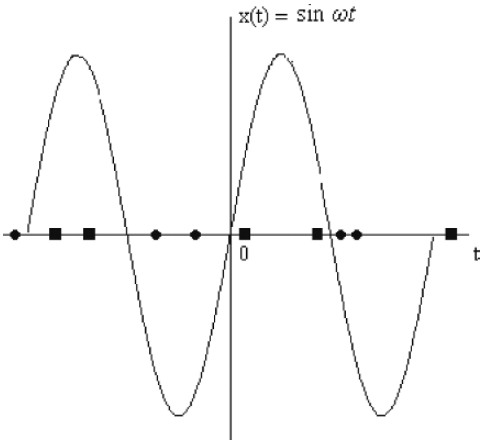

Fig. 4.4 Sinusoidal function with an infinite VC dimension

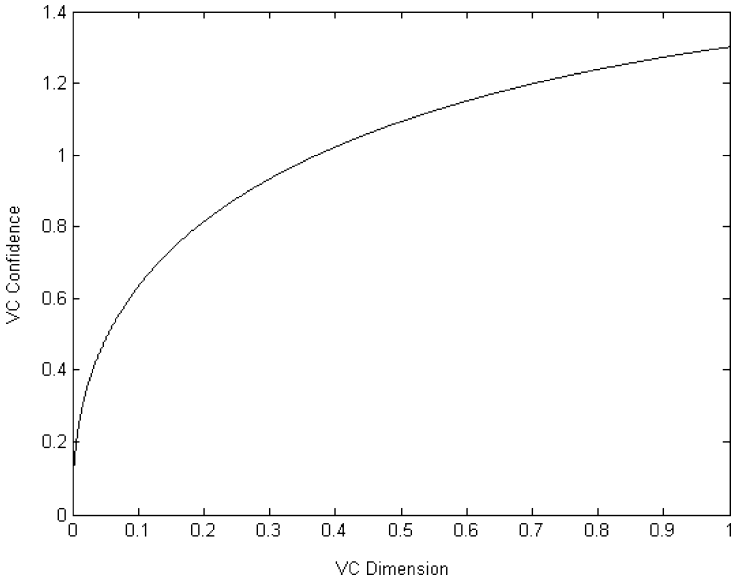

Fig. 4.5 VC confidence is monotonically increasing for the VC dimension h

4.2.4 Structural Risk Minimization

The main objective of the structural risk minimization (SRM) principle is to choose the model complexity optimally for a given training sample. For determining the optimal complexity, we note that the first term in the right hand side of (4.3) depends on a particular function from the set of functions. The second term in the right hand side of (4.3) depends mainly on the VC dimension of the set of functions. So, the goal is to find the subset of the chosen set of functions such that the risk bound for that subset is minimized. As the VC dimension h is an integer, we cannot arrange things so that h varies smoothly. Under SRM, the set S of loss functions $\{Q(x, \alpha)\}$ has a structure (Vapnik 1995). As shown in Fig. 4.6, the structure consists of nested subsets S_k such that

$$S_1 \subset S_2 \subset \ldots \subset S_k \subset \ldots, \text{ where } h_1 < h_2 < \ldots < h_k < \ldots. \tag{4.4}$$

From (4.3), it is to be noted that the VC confidence can be seen as the (estimated) difference between the true risk and the empirical risk. With SRM and the nested subset structure, we try to find the model with the right balance in order to avoid the over- and underfitting. Figure 4.7 shows the connection between the SRM-nested subset structure and how it is used to choose the optimal model with the balance of accuracy and generalization capacity.

The SRM principle can be implemented by training a series of learning machines, one for each subset, where for a given subset the goal of training is to minimize the empirical risk (Burges 1998). From the series of trained machines, the optimal one

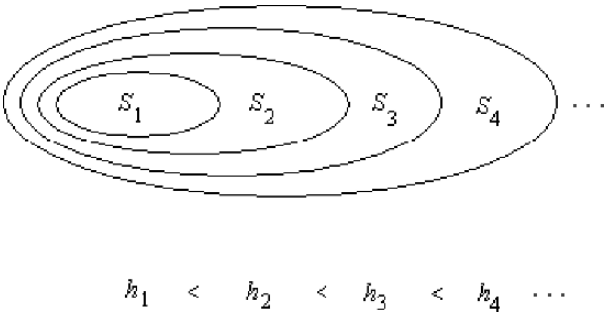

Fig. 4.6 Nested subsets of functions at par with the VC dimension

can be found for which the sum of the empirical risk and the VC confidence is minimum.

Multiple output problems can usually be reduced to a set of independent single output problems. Hence, irrespective of the number of outputs, the SRM principle can, in general, be applied to deduce the optimal model from multiple inputs.

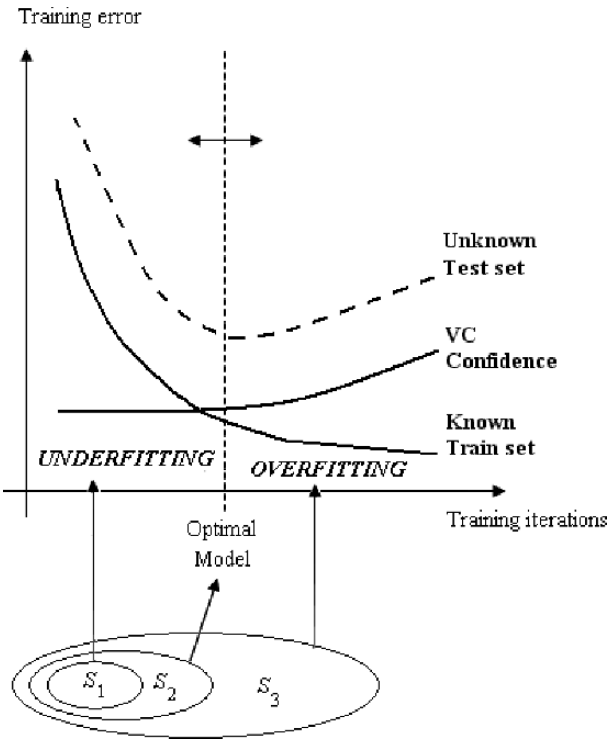

Fig. 4.7 Nested subset structure and choosing the optimal model

4.3 Support Vector Machine

4.3.1 Linear Classification

The aim of linear classification task is to construct linear decision boundaries that explicitly try to separate data into different classes. Figure 4.8 shows an example of mixture of two different data classes represented using the plus and the minus signs.

As shown in Fig. 4.8, in two-dimensional space, the separating hyperplane would be a straight line which can be defined, in general, by the weight vector w and the bias vector b. So, it can be generally represented as

$$w^T x + b = 0, \tag{4.5}$$

where, the superscript T indicates the transpose. In the classified data classes, the plus-sign data class can be represented as class 1 while the minus-sign data class can be represented as class -1. Hence, class 1 and -1 can be described by (4.6) and (4.7) respectively.

$$w^T x + b = 1. \tag{4.6}$$

$$w^T x + b = -1. \tag{4.7}$$

Concentrating on the example shown in Fig. 4.8, and the general hyperplane, and the two classes described by (4.5–4.7), we can observe the following equations corresponding to the five data points.

$$w_1.1 + w_2.1 + b = -1, \tag{4.8}$$

$$w_1.2 + w_2.1 + b = -1, \tag{4.9}$$

$$w_1.2 + w_2.2 + b = -1, \tag{4.10}$$

$$w_1.4 + w_2.4 + b = 1, \tag{4.11}$$

$$w_1.5 + w_2.5 + b = 1. \tag{4.12}$$

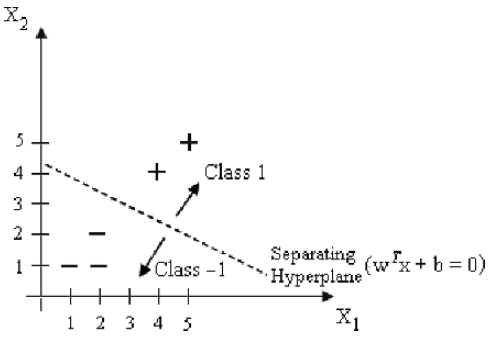

Fig. 4.8 Example of linear data classification

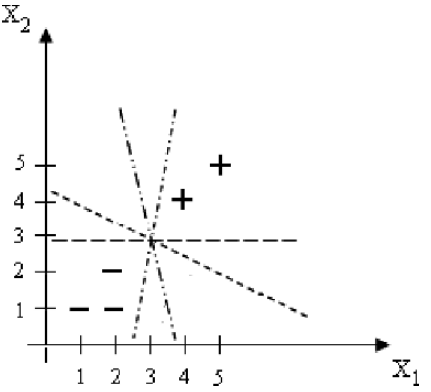

Fig. 4.9 Infinite solution to the classification problem

So, we have three unknown variables, w_1, w_2, b, and five equations. Hence, we cannot have a unique solution. In fact, for the data classes shown in Fig. 4.8, we have infinite number of classification solutions, i.e., possible separating hyperplanes as shown in Fig. 4.9. So, how to choose the optimal one? This is the question that support vector tries to answer as we shall see later.

4.3.1.1 Least Square Solution

We can represent (4.8–4.12) in the matrix form as

$$\begin{bmatrix} 1 & 1 & 1 \\ 2 & 1 & 1 \\ 2 & 2 & 1 \\ 4 & 4 & 1 \\ 5 & 5 & 1 \end{bmatrix} . \begin{bmatrix} w_1 \\ w_2 \\ b \end{bmatrix} = \begin{bmatrix} -1 \\ -1 \\ -1 \\ 1 \\ 1 \end{bmatrix} . \tag{4.13}$$

Equation (4.13) can be represented as

$$A.\beta = y. \tag{4.14}$$

Therefore, the least square estimate of the parameter vector β would be

$$\hat{\beta} = A^{-1}y. \tag{4.15}$$

Here, A^{-1} represents the matrix inversion operation.

$$A^{-1}y = (A^T)^{-1}A^{-1}A^T y = (A^T A)^{-1}A^T y = pinv(A).y. \tag{4.16}$$

Here, there term $(A^T A)^{-1}A^T$ is called the *pseudo-inverse* (*pinv*) (Golub and Van Loan 1996) of the matrix A. Using the least square estimate (4.15–4.16) of the parameter vector, $\hat{\beta}^T = \begin{bmatrix} \hat{w}_1 & \hat{w}_2 & \hat{b} \end{bmatrix}$, we can obtain the classification for any unseen data as

$$y = \text{sign}(w^T x + b). \tag{4.17}$$

However, we have a three-dimensional hyperplane for a two-dimensional classification problem.

4.3.1.2 Perceptron Learning

Perceptron learning (Schölkopf and Smola 2001) is oriented towards finding a separating hyperplane by minimizing the distances of the misclassified points to the decision boundary. Misclassified points mean the points which are categorized into wrong classes. This is shown in Fig. 4.10, by the circled data points which lie in the wrong data class. So, intuitively we can imagine that minimizing the distances of such misclassified points brings us close to the minimum error condition, i.e., towards the optimal solution.

Continuing further the description of the two-dimensional classification problem from the previous section, now we additionally introduce these misclassified points. So, we have two misclassification cases.

Case 1: If a response $y_i = 1$ is misclassified, then $w^T x + b < 0$,
Case 2: If a response $y_i = -1$ is misclassified, then $w^T x + b > 0$.

Combining the two cases, the goal is

$$\min \left[-\sum_{i \in D} y_i (w^T x_i + b) \right], \text{ with respect to } w, b, \tag{4.18}$$

where, D is the set of misclassified points. We can define the entity to minimize as

$$J = -\sum_{i \in D} y_i (w^T . x_i + b) = -\sum_{i \in D} y_i (x_i^T . w + b). \tag{4.19}$$

In (4.19), as w and x are operated using a scalar dot product, hence they are interchangeable. J is non-negative with respect to misclassification and proportional

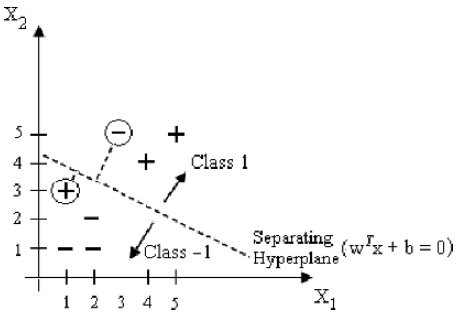

Fig. 4.10 Misclassified points

to the distance (perpendicular) of the misclassified points (shown by the dashed lines in Fig. 4.10) to the decision boundary which is defined by (4.5).

Using the gradient descent algorithm (see Chap. 3, Sect. 3.3.2), we get

$$\frac{\partial J}{\partial w} = -\sum_{i \in D} y_i x_i, \tag{4.20}$$

$$\frac{\partial J}{\partial b} = -\sum_{i \in D} y_i. \tag{4.21}$$

So, following the update rule (see (3.5) in Chap. 3, Sect. 3.3.1) we have

$$w_{i_{new}} = w_i + (-\alpha).\frac{\partial J}{\partial w} = w_i + \sum_{i \in D} \alpha y_i x_i. \tag{4.22}$$

$$b_{new} = b + (-\alpha).\frac{\partial J}{\partial b} = b + \sum_{i \in D} \alpha y_i. \tag{4.23}$$

Here, α is the learning rate.

4.3.2 Optimal Separating Hyperplane

From the classification example in Sect. 4.3.1, we've seen that the separating hyperplane is not unique in nature. There are many solutions when data are separable. There are three associated problems.

1. If data are separable, the solution found depends on the starting values, as can be seen in the cases of the least square solution and the perceptron learning.
2. Smaller the gap between the separable data classes, larger the time to find the separating hyperplane.
3. If data are not separable, the algorithm does not converge and cycles develop.

For the separable data, problem 1 can be eliminated by adding additional constraints to the separating hyperplane. Problem 2 can be eliminated by seeking a hyperplane not in the original space, but in a much enlarged spaced obtained by transformation. For example, we change the geometry of the original given two-dimensional input data set $x_i = \{x_1, x_2\}$ by projecting it into a higher dimensional space using a transformation to have new three-dimensional input data set, say, $X_i = \{X_1, X_2, X_1X_2\}$. We might not be able to find a suitable separating straight line in the two-dimensional space, but we could easily find a separating plane in the three-dimensional space. This way, we could also sometimes solve the problem 3 if the data are actually separable but apparently non-separable in a lower dimension. This is one of the reasons that makes this projecting and transformation technique (which is also the basis of the support vector machines) more robust in terms of

convergence than the neural network-based techniques. We will discuss more about this later.

4.3.2.1 Margin

For tackling the problem of choosing the optimal hyperplane from the many possible options one key parameter is the margin. In other words, the optimal separating hyperplane separates the classes (two in case of the problem in Sect. 4.3.1) and maximizes the distance between the closest point from each class, i.e., maximizes the margin. Intuitively, this is obvious because we are more confident about an optimal separation if we can maintain a maximum safety gap (margin). The effect of the margin is shown in Fig. 4.11.

4.3.2.2 Affine Set

Figure 4.12 shows a separating hyperplane and two data points on it.

From Fig. 4.12, we have the following description of the two data points.

$$W^T x_1 + b = 0. \tag{4.24}$$

$$W^T x_2 + b = 0. \tag{4.25}$$

Subtracting (4.25) from (4.24), we get

$$w^T (x_1 - x_2) = 0. \tag{4.26}$$

Hence, the difference vector $(x_1 - x_2)$ of the two data points is lying in the affine set (Vapnik 1998) or the hyperplane defined by (4.5).

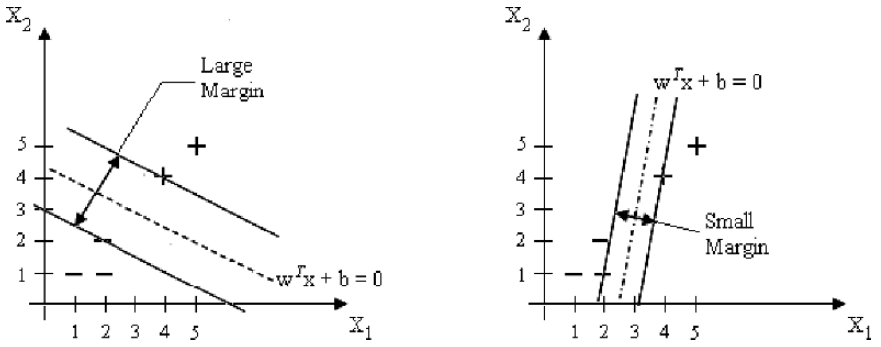

Fig. 4.11 Separating hyperplanes and margins

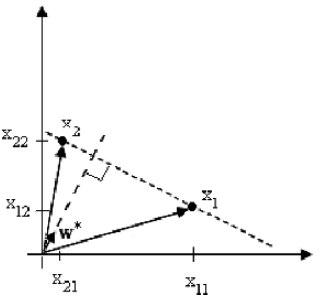

Fig. 4.12 Hyperplane with data points lying on it

4.3.2.3 Orthogonal Vector

Two vectors (**a**, **b**) are said to be orthogonal if the scalar (dot) product is zero.

$$\mathbf{a}^T.\mathbf{b} = 0. \tag{4.27}$$

Following this, we can define a vector **w** which is normal to the surface $w^T x + b = 0$. This is the orthogonal vector to the separating hyperplane. The length of the orthogonal vector is

$$\|\mathbf{w}\| = \sqrt{\mathbf{w}^T\mathbf{w}} = \sqrt{\mathbf{w}^2}. \tag{4.28}$$

We can also define a unit vector $\mathbf{w}^*$ as

$$\mathbf{w}^* = \frac{\mathbf{w}}{\|\mathbf{w}\|}. \tag{4.29}$$

4.3.2.4 Distance of a Point

Now, we want to deduce the distance of a point from the optimal separating hyperplane. Figure 4.13 shows a general point x at a distance d from the surface $w^T x + b = 0$.

The distance between two vectors is given by the scalar (dot) product equal to the length of one vector multiplied by the algebraic projection of the another vector on the direction of the first. For determining the distance of the point x, we utilize a difference vector of the point x and a point x_0 on the affine set and the unit orthogonal vector $\mathbf{w}^*$, as shown in Fig. 4.13. Then, we have,

$$d = \mathbf{w}^{*^T}(\mathbf{x} - \mathbf{x}_0) = \frac{\mathbf{w}^T}{\|\mathbf{w}\|}(\mathbf{x} - \mathbf{x}_0), \tag{4.30}$$

as x_0 is a point on the affine set, $\mathbf{w}^T\mathbf{x}_0 + b = 0$. Hence,

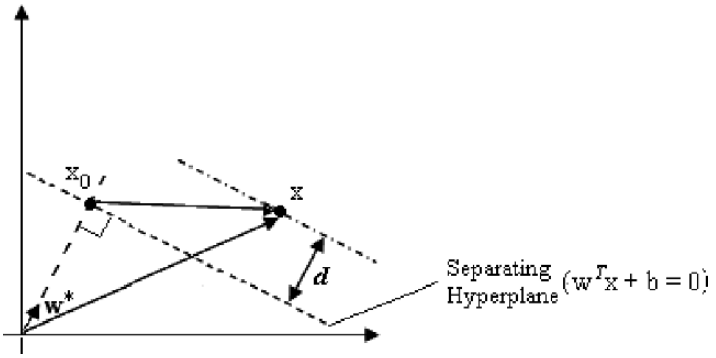

Fig. 4.13 Distance of a point

$$d = \frac{\mathbf{w}^T\mathbf{x} - \mathbf{w}^T\mathbf{x}_0}{\|\mathbf{w}\|} = \frac{\mathbf{w}^T\mathbf{x} + b}{\|\mathbf{w}\|}. \tag{4.31}$$

We can generalize (4.31) as follows. Let there be a given point $p(x_{1p}, x_{2p}, \ldots, x_{np})$ and a hyperplane $k(x, w, b) = 0$ defined by

$$w_1x_1 + w_2x_2 + \ldots + w_nx_n \pm b = 0. \tag{4.32}$$

The distance d_p from point p to the hyperplane k is given as

$$d_p = \frac{|(\mathbf{w}^T.\mathbf{x}_p) \pm b|}{\|\mathbf{w}\|} = \frac{|(\mathbf{w}.\mathbf{x}_p^T) \pm b|}{\|\mathbf{w}\|} = \frac{\|w_1x_{1p} + w_2x_{2p} + \ldots + w_nx_{np} \pm b|}{\sqrt{w_1^2 + w_2^2 + \ldots + w_n^2}}. \tag{4.33}$$

Let's see an example. We want to calculate the distance d between the point (1,1,1,1,1) and the 5-dimensional hyperplane defined as $x_1 + 2x_2 + 3x_3 + 4x_4 + 5x_5 + 7 = 0$. Using (4.33),

$$d = \frac{|[1 \quad 2 \quad 3 \quad 4 \quad 5][1 \quad 1 \quad 1 \quad 1 \quad 1]^T + 7|}{\sqrt{1^2 + 2^2 + 3^2 + 4^2 + 5^2}} = \frac{22}{\sqrt{55}}.$$

4.3.2.5 Distance of Margin

Figure 4.14 shows the optimal separating hyperplane separating the two classes $(1, -1)$ and the maximal margin. We define two data points x_1 and x_2 from the two classes on the two margins represented by the vectors $\mathbf{x}_1$ and $\mathbf{x}_2$. Figure 4.14 also shows the orthogonal vector $\mathbf{w}$ to the hyperplane. The distance between the margins is d. And the vectors $\mathbf{x}_1$ and $\mathbf{x}_2$ have angles of θ_1 and θ_2 with the orthogonal vector $\mathbf{w}$ respectively.

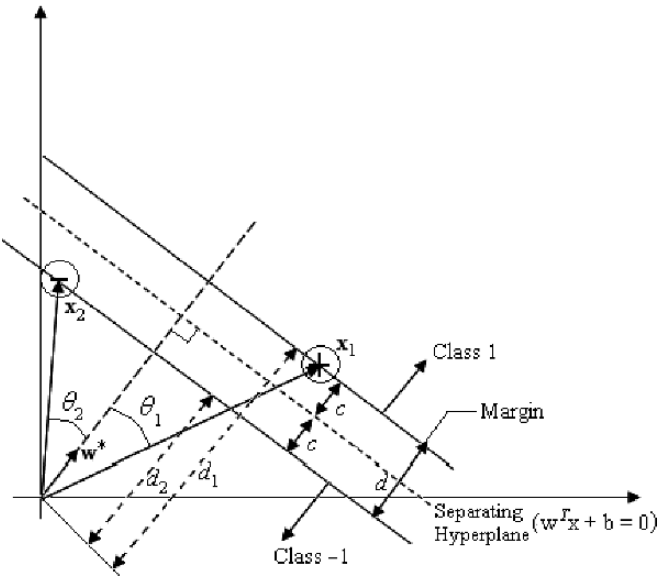

Fig. 4.14 Margin distance

Then, the distance between the margins is given by

$$d = d_1 - d_2. \tag{4.34}$$

$$d_1 = |\mathbf{x}_1| \cos\theta_1. \tag{4.35}$$

$$d_2 = |\mathbf{x}_2| \cos\theta_2. \tag{4.36}$$

So,

$$d = |\mathbf{x}_1| \cos\theta_1 - |\mathbf{x}_2| \cos\theta_2. \tag{4.37}$$

We have

$$\cos\theta_1 = \frac{\mathbf{w}.\mathbf{x}_1^T}{|\mathbf{w}||\mathbf{x}_1|}, \tag{4.38}$$

$$\cos\theta_2 = \frac{\mathbf{w}.\mathbf{x}_2^T}{|\mathbf{w}||\mathbf{x}_2|}. \tag{4.39}$$

Hence,

$$d = \frac{\mathbf{w}.\mathbf{x}_1^T}{|\mathbf{w}|} - \frac{\mathbf{w}.\mathbf{x}_2^T}{|\mathbf{w}|} = \frac{\mathbf{w}.\mathbf{x}_1^T - \mathbf{w}.\mathbf{x}_2^T}{|\mathbf{w}|} \tag{4.40}$$

For class 1 we have

$$\mathbf{w}.\mathbf{x}_1^T + b = 1, \tag{4.41}$$

$$\therefore \mathbf{w}.\mathbf{x}_1^T = 1 - b.$$

For class -1 we have

$$\begin{aligned} \mathbf{w}.\mathbf{x}_2^T + b &= -1, \\ \therefore \mathbf{w}.\mathbf{x}_2^T &= -1 - b. \end{aligned} \tag{4.42}$$

Using (4.40–4.42), we get

$$d = \frac{1 - b - (-1 - b)}{|\mathbf{w}|} = \frac{2}{|\mathbf{w}|} \tag{4.43}$$

From Fig. 4.14, we have the half-distance $c = d/2$. Hence, from (4.43)

$$c = \frac{d}{2} = \frac{1}{|\mathbf{w}|}. \tag{4.44}$$

4.3.3 Support Vectors

SVMs can be seen as a new method for training classifiers, regressors, like polynomial models, neural networks, radial basis function, fuzzy models and so on. SVMs are based on the structural risk minimization (SRM) principle, whereas neural networks (refer to Chap. 3) are based on the empirical risk minimization (ERM) principle (that is why SVMs usually generalize better than the NNs). Unlike the classical adaptation algorithm that work on L_1 or L_2 norms, i.e., by minimizing an absolute error or an error-square respectively, the SVMs minimize the Vapnik Chervonenkis (VC) bounds (Vapnik and Chervonenkis 1971; Vapnik 1995). According to Vapnik's theory, minimizing the VC bounds means the expected probability of error is low, i.e., good generalization, and that too with unknown probability distribution.

For a separable classification task, the idea is to map the training data into a higher dimensional feature space using a kernel function where an optimal separating hyperplane (defined by the w: weight vector, b: bias) can be found that maximizes margin or the distance between the closet data points as discussed in Sect. 4.3.2.

For the two-class classification problem (Fig. 4.14) we have the indicator function defined by (4.17). For the class 1,

$$y = 1; \quad \mathbf{w}^T.\mathbf{x} + b \geq 0. \tag{4.45}$$

For the class -1,

$$y = -1; \quad \mathbf{w}^T.\mathbf{x} + b < 0. \tag{4.46}$$

So, correctly classified points are represented as

$$y\left(\mathbf{w}^T.\mathbf{x} + b\right) \geq 0. \tag{4.47}$$

Distance d (of the points in either class) is defined by (4.31). So, for the correctly classified points we have

$$yd \geq 0. \tag{4.48}$$

For the optimal hyperplane, we need to find the maximum margin (defined by the half-distance $c = \frac{1}{|\mathbf{w}|}$ as in Sect. 4.3.2.5). So, in general, for n data points, the optimization problem can be described as

$$\max c, \text{with respect to}(\mathbf{w}, b). \tag{4.49}$$

Subject to the constraint

$$y_i d_i = \frac{y_i \left(\mathbf{w}^T.\mathbf{x}_i + b\right)}{|\mathbf{w}|} \geq c, \quad i = 1, 2, \ldots, n. \tag{4.50}$$

Substituting $c = \frac{1}{|\mathbf{w}|}$ in (4.50), the constraint becomes

$$y_i \left(\mathbf{w}^T.\mathbf{x}_i + b\right) \geq 1, \quad i = 1, 2, \ldots, n. \tag{4.51}$$

If we consider, in general, a monotonic function $\sqrt{f}$, then minimization of $\sqrt{f}$ is equivalent to the minimization of f. Consequently, a minimization of the norm $|\mathbf{w}|$ equals minimization of $\mathbf{w}^T\mathbf{w} = (\mathbf{ww}) = \sum_{i=1}^{n} w_i^2$ and this leads to a maximization of margin $d(= 2c)$. So, the transformed optimization problem (for finding the optimal hyperplane) is

$$\min \frac{1}{2}|\mathbf{w}|^2, \text{with respect to } (\mathbf{w}, b), \tag{4.52}$$

subject to the constraint described by (4.51).

The m number of points closest to the optimal separating hyperplane with the largest margin $2c = \frac{2}{|\mathbf{w}|}$, are specified as the *support vectors* (SVs) defined by

$$y_i \left(\mathbf{w}^T.\mathbf{x}_i + b\right) = 1, \quad i = 1, 2, \ldots, m. \tag{4.53}$$

4.3.3.1 Interpretation of SV in Higher Dimension

In the higher dimension, the cost function (analogous to (4.52)) will be quadratic and the constraints (analogous to (4.51)) linear, i.e., it would a quadratic programming (Vapnik 1995) problem. The optimum separating hyperplane can be represented based on the kernel function (Schölkopf and Smola 2001) as

$$f(x) = sign\left[\sum_{i=1}^{n} \alpha_i y_i \psi(x, x_i) + b\right], \tag{4.54}$$

where, n is the number of training samples, y_i is the label value of the example i, ψ represents the kernel function, and α_i coefficients must be found in a way to maximize a particular Lagrangian[13] representation. Subject to the constraints $\alpha_i \geq 0$ and $\sum \alpha_i y_i = 0 \, \forall i$, there is a Lagrange multiplier (α_i) for each training point and only those training points that lie close to the decision boundary have non-zero α_i. These are called the support vectors (SVs).

In real world problems, data are noisy and no linear separations are possible in the feature space. The hyperplane margin can be made more relaxed by penalizing the training points that the system misclassifies. Hence, the optimum separating hyperplane equation is defined as

$$y_i \left(\mathbf{w}^T \mathbf{x}_i + b \right) \geq 1 - \beta_i, \beta_i \geq 0, \tag{4.55}$$

where, β_i introduces a positive *slack* variable that measure the amount of violation from the constraints.

So, the optimization criterion to obtain the (optimum) separating hyperplane is

$$\min \left[\frac{1}{2} \|\mathbf{w}\|^2 + \mu \sum_{i=1}^{n} \beta_i \right]. \tag{4.56}$$

The penalty parameter μ is a regularization parameter that controls the trade off between maximizing the margin and minimizing the training error. Hence, the new learning machine utilizing the SVs, is called the support vector machine (SVM). SVM has two associated parameters:

1. the kernel function
2. the penalty parameter μ.

This is called the *soft-margin* (Burges 1998) approach.

4.3.4 Convex Optimization Problem

A convex optimization problem consists of a quadratic criterion with linear inequality constraints. So, for the support vector machines, we introduce a Lagrangian primal

[13] Lagrangian is a function of the dynamic variables of a system. In mathematical optimization problems, Lagrange multipliers are a method for dealing with constraints. We often encounter problem of extremizing a function of several variables subject to one or more constraints given by further functions of the variables to given values. The method introduces a new unknown scalar variable, the Lagrange multiplier, for each constraint; and forms a linear combination involving the multipliers as coefficients. This reduces the constrained problem to an unconstrained problem. It may then be solved, for example by the usual gradient method. For general description on Lagrangian see (Fletcher 1987).

$$L_P = \frac{1}{2}\mathbf{w}^T\mathbf{w} - \sum_{i=1}^{n}\alpha_i[y_i(\mathbf{w}^T\mathbf{x}_i + b) - 1], \tag{4.57}$$

where, α_i's are the Lagrange multipliers. So, the complete optimization problem for SVMs in terms of the Lagrangian function is given as follows.

$$\text{Objective function } L(\mathbf{w}, b, \alpha) = \frac{1}{2}\mathbf{w}^T\mathbf{w} - \sum_{i=1}^{n}\alpha_i\left[y_i\left(\mathbf{w}^T\mathbf{x}_i + b\right) - 1\right], \tag{4.58}$$

$$\text{Inequality constraints } y_i\left(\mathbf{w}^T.\mathbf{x}_i + b\right) \geq 1, \quad i = 1, 2, \ldots, n.$$

The Lagrangian L in (4.58) must be minimized with respect to $(\mathbf{w}, b)$, and has to be maximized with respect to nonnegative α_i ($\alpha_i \geq 0$).

The solution of the problem in the primal space (space described by $\mathbf{w}$ and b) is as follows. We search for the saddle point (i.e., $(\mathbf{w}_0, b_0, \alpha_0)$) using (4.57). At the saddle point,

$$\frac{\partial L_P}{\partial \mathbf{w}} = \mathbf{w} - \sum_{i=1}^{n}\alpha_i y_i \mathbf{x}_i = 0. \tag{4.59}$$

$$\frac{\partial L_P}{\partial b} = -\sum_{i=1}^{n}\alpha_i y_i = 0. \tag{4.60}$$

Equation (4.59) gives

$$\mathbf{w} = \sum_{i=1}^{n}\alpha_i y_i \mathbf{x}_i. \tag{4.61}$$

And (4.60) gives

$$\sum_{i=1}^{n}\alpha_i y_i = 0. \tag{4.62}$$

Substituting (4.61–4.62) in (4.57), we transform the Lagrangian primal to dual variable Lagrangian (L_D).

$$L_D = \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j {\mathbf{x}_i}^T\mathbf{x}_j. \tag{4.63}$$

It is to be noted that L_D is expressed in terms of the given data points and depend only on the scalar products of the input vectors $(\mathbf{x}_i, \mathbf{x}_j)$. In the dual space (space described by α_i), we have to maximize L_D subject to the following conditions

$$\alpha_i \geq 0,$$
$$\sum_{i=1}^{n}\alpha_i y_i = 0. \tag{4.64}$$

For finding the optimum of the constraint function in the dual space, we have to use the Karush-Kuhn-Tucker (KKT) (Karush 1939; Kuhn and Tucker 1951; Fletcher 1987) conditions. According to the KKT conditions, the solution must also satisfy

$$\alpha_i \left[y_i \left(\mathbf{w}^T \mathbf{x}_i + b \right) - 1 \right] = 0 \quad \forall i. \tag{4.65}$$

We have two cases for the (4.65).

Case 1:
If $\alpha_i > 0$, then $y_i \left(\mathbf{w}^T \mathbf{x}_i + b \right) - 1 = 0 \Rightarrow y_i \left(\mathbf{w}^T \mathbf{x}_i + b \right) = 1$. This indicates that the input vector $\mathbf{x}_i$ is on the boundary of the margin, i.e., they are support vectors.

Case 2:
If $\alpha_i = 0$, then $y_i \left(\mathbf{w}^T \mathbf{x}_i + b \right) - 1 > 0 \Rightarrow y_i \left(\mathbf{w}^T \mathbf{x}_i + b \right) > 1$. This indicates that the input vector $\mathbf{x}_i$ is not on the boundary of the margin.

After solving the dual problem for the two cases of $\alpha_i, i = 1, 2, \ldots, n$, the solution of the primal problem is obtained using the relation specified by (4.61). Solution vector $\mathbf{w}$ is defined in terms of a linear combination of the support points. b is calculated using $y_i \left(\mathbf{w}^T \mathbf{x}_i + b \right) = 1$, for any support points. So, the optimal separating hyperplane produces a function

$$f(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + b. \tag{4.66}$$

The classification (indicator) output is then given by $sign\ f(x)$. If the data classes are Gaussian in nature, then least square is the optimal solution (see Sect. 4.3.1.1). However, if data classes are non-Gaussian (which is more realistic in real life situation), the support vectors are more robust solutions.

4.3.5 Overlapping Classes

Non-overlapping data sets maintain the margin properly, i.e., no data points from any class leak into the margin region. However, such a case is rare in practice. Instead, overlapping classes consist of data points getting into the margin region and misclassified data points, i.e., points which should be confined to any specific class, gets into the other class(es). Figure 4.15 shows an example of overlapping class.

In Fig. 4.15, the optimal margin algorithm is generalized to the non-separable problems by the introduction of the nonnegative slack variables β_i in the statement of the optimization problem. We cannot use the quadratic programming solutions in the case of an overlapping class as the inequality constraints $y_i \left(\mathbf{w}^T \mathbf{x}_i + b \right) \geq 1, i = 1, 2, \ldots, n$ are not satisfied. So, the transformed optimization algorithm is

$$\max\ c, \text{with respect to } (\mathbf{w}, b), \tag{4.67}$$

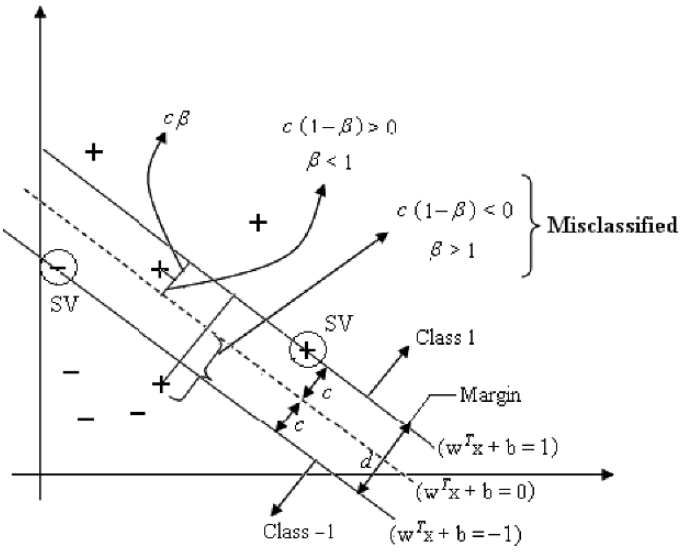

Fig. 4.15 Overlapping class

subject to,

$$y_i \frac{\left(\mathbf{w}^T.\mathbf{x}_i + b\right)}{|\mathbf{w}|} \geq c(1 - \beta_i), \quad \forall i \tag{4.68}$$

where,

$$\beta_i \geq 0, \tag{4.69}$$

$$\sum_i \beta_i \leq k. \tag{4.70}$$

Equation (4.70) bounds the total number of misclassifications at k. We have $c = \frac{1}{|\mathbf{w}|}$, therefore, the equivalent form of the optimization problem is given as follows.

$$\min |\mathbf{w}|, \text{ with respect to } (\mathbf{w}, \beta), \tag{4.71}$$

subject to,

$$y_i \left(\mathbf{w}^T \mathbf{x}_i + b\right) \geq 1 - \beta_i, \ \forall \beta_i \geq 0. \tag{4.72}$$

The bound of misclassification is given by (4.70).

For the generalized optimal separating hyperplane in the overlapping case, we introduce in the minimization functional an extra term accounting for the cost of the overlapping errors. The changed objective function with penalty parameter μ is

$$\min \left[\frac{1}{2}\mathbf{w}^T\mathbf{w} + \mu \sum_{i=1}^{n} \beta_i\right], \text{ with respect to } (\mathbf{w}, \beta), \tag{4.73}$$

subject to the inequality constraints given by (4.69 & 4.72).

Increasing the penalty parameter μ corresponds to assigning a higher penalty to the misclassification errors, side by side resulting in larger weights. This makes it a convex optimization (Vapnik 1995, 1998; Schölkopf and Smola 2001) problem.

4.3.6 Nonlinear Classifier

We made the linear classification example (Sect. 4.3.1) closer to the practical case by introducing the overlapping classes in the previous section. We can make it one step closer to the practical scenario by considering a nonlinear separating hypersurface (decision boundary) rather than the linear separation lines. This could be achieved by considering a linear classifier in so-called feature space (higher dimensional). Figure 4.16 shows an example of the nonlinear classifier.

In plot (a) of Fig. 4.16, the plus and minus data classes are separable using a linear separation line. However, if we interchange the positions of the circled plus and minus data points, we get a new class of data as shown in plot (b) of Fig. 4.16. We can still utilize the linear separation line, but this will result in two misclassifications. Instead, with the nonlinear decision boundary (plot (c) of Fig. 4.16), we would be able to separate the two classes without any error.

We can design the nonlinear SVM by mapping the input vectors $\mathbf{x} \in \mathbf{R}^n$, into vectors $\mathbf{z}$ of a higher dimensional feature space ($\mathbf{z} \in \psi(\mathbf{x})$), i.e., enlarging the input space using mapping (basis expansions).

$$\psi(\mathbf{x}_i) = (\psi_1(\mathbf{x}_i), \psi_2(\mathbf{x}_i), \ldots, \psi_M(\mathbf{x}_i)) \,. \tag{4.74}$$

We fit the SV classifier using input features $\psi_1(\mathbf{x}_i), \psi_2(\mathbf{x}_i), \ldots, \psi_M(\mathbf{x}_i); i = 1, 2, \ldots, n$, and produce the nonlinear boundary function $f(\mathbf{x}) = \mathbf{w}^T \psi(\mathbf{x}) + b$. By performing such a mapping, we hope that in a higher dimensional ψ-space, our learning algorithm will be able to linearly separate the projected images of the (original) input vector $\mathbf{x}$ by applying the linear SVM formulations.

As an example, we consider the following function

$$y_i = \exp(\mathbf{x}_i), \quad i = 1, 2, \ldots, n. \tag{4.75}$$

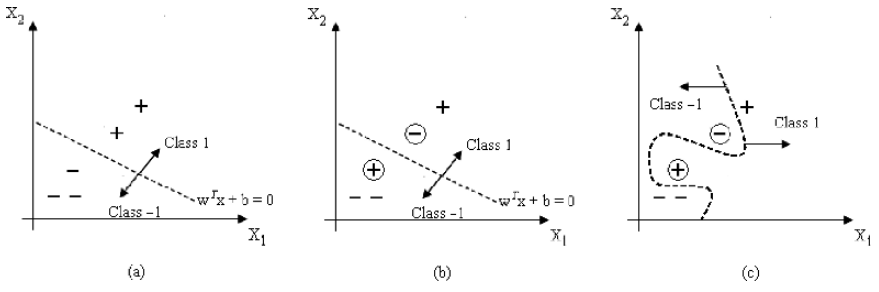

Fig. 4.16 Nonlinear classifier example

y_i is exponentially linearized with the following ψ-space mapping,

$$\psi(\mathbf{x}_i) = \left(\mathbf{x}_i, \mathbf{x}_i^2, \mathbf{x}_i^3, \ldots, \mathbf{x}_i^M\right). \tag{4.76}$$

And, linear coefficients **w** is equal to the Taylor series as given below.

$$y_i = \sum_{n=0}^{\infty} \frac{1}{n} \mathbf{x}_i^n \approx \sum_{n=1}^{M} \mathbf{w}_n \psi_n(\mathbf{x}_i) + \mathbf{w}_0. \tag{4.77}$$

4.3.7 Kernel Method

4.3.7.1 Introduction

Kernel method implicitly introduces a notion of similarity between data. Kernels indicate the choice of similarity patterns, implying the choice of features. Let's see an analogy for kernel. We consider two books with same number of pages but of entirely different subjects. Although these two books are different subject-wise, yet we see somewhat similarity based on the feature of number of pages. So, now we can think of a book with same number of blank pages as a kernel. Then, taking this blank book we fill up with the appropriate subject to arrive at the individual pattern.

Mathematically, kernel methods exploit information about the inner products between data items. Hence, kernel functions can be thought of as the inner products in some feature space. Once the kernel function for the data is mentioned, we no longer require any information about the features of the data. As we saw in the previous book example, once we have the blank book, we do not need to specify that we are working on the basis number of pages, we just use that blank book to get the individual books!

We've seen in the previous section that linear classifiers cannot particularly deal with nonlinearly separable data and noisy data. In those cases, we require a nonlinear classifier. Again, instead of using a nonlinear classifier in lower dimension, we can intelligently map the data into a richer feature space (higher dimension) which would include the problematic nonlinear features as well. Then, in that transformed higher dimensional feature space, we might have a linear separation. Actually from optimization point of view, we continue to search for this nice feature space where we can find a linear separation. This is shown in Fig. 4.17.

Oftentimes in this sort of classification tasks, the feature space or the basis functions (μ) has to be chosen high. For example, the classification task, $y = sign(x)$, requires infinite number of sinusoidal basis functions. The computational cost increases rapidly with the increasing dimension. The main reason for this increase is that all the inner products $\psi(\mathbf{x}_i)^T.\psi(\mathbf{x}_j)$, $i, j = 1, 2, \ldots, n$, have to be computed for solving the optimization problem. Computationally, these higher dimensions mean dealing with large vectors, requiring lots of memory. This dimension explosion and the associated computational demands can be effectively lowered by

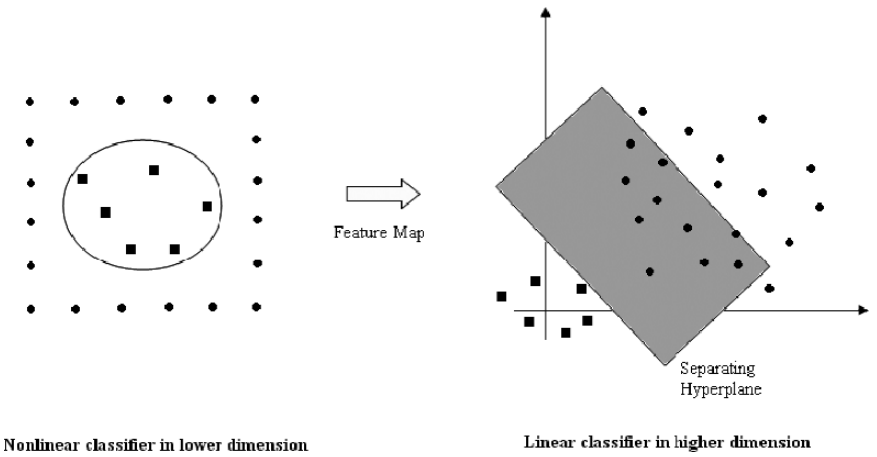

Fig. 4.17 Linear classification in higher dimension

using the kernel functions (Burges 1998; Schölkopf and Smola 2001). The kernel methods can be used to replace the higher dimension feature space (mathematically, by replacing the exploding inner products). Hence, also from the practical implementation point of view, kernel methods are important to tackle infinite dimensions efficiently in time and space.

So, the 4-step synopsis for the kernel methods are given below.

1. We've nonlinear, noisy data which we want to classify.
2. One way is to construct a nonlinear classifier in the given space. But this is complex!
3. We project the data into a higher dimensional feature space where linear classification is possible. But this results in huge dimensions, computation problem!
4. We continue to use the excellent concept in point 3 by transforming the data into an easier feature space where we can perform linear classification. But we get rid of the higher dimension by performing this transformation with the aid of kernel functions.

4.3.7.2 Definition of a Kernel

A kernel function is a function that returns the value of the dot product between the images of the two arguments.

$$K(x_1, x_2) = \langle \phi(x_1), \phi(x_2) \rangle. \tag{4.78}$$

An important concept associated with the kernel functions is the dot product which can be represented using the kernel matrix also known as the Gram matrix (Schölkopf and Smola 2001).

$$K = \begin{bmatrix} K(1,1) & K(1,2) & K(1,3) & \dots & K(1,n) \\ K(2,1) & K(2,2) & K(2,3) & \dots & K(2,n) \\ K(3,1) & K(3,2) & K(3,3) & \dots & k(3,n) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ K(n,1) & K(n,2) & K(n,3) & \dots & K(n,n) \end{bmatrix}. \tag{4.79}$$

The kernel matrix (Schölkopf and Smola 2001) is the central structure in the kernel machines, containing all the necessary information for the learning algorithm. The kernel matrix (aka gram matrix) shown in (4.79) has some interesting properties.

- The kernel matrix is symmetric positive definite.
- Any symmetric positive definite matrix can be regarded as a kernel matrix, that is as an inner product matrix in some space.

These points are formally known as Mercer's theorem (Mercer 1909; Burges 1998; Schölkopf and Smola 2001). According to Mercer's theorem, for a continuous symmetric nonnegative definite kernel K, there is an orthonormal basis for the mapping, consisting of the eigenfunctions such that the corresponding sequence of eigenvalues (λ_i) is nonnegative. Using this, the kernel function in (4.78) can be written as

$$K(x_1, x_2) = \sum_i \lambda_i \phi_i(x_1)\phi_i(x_2). \tag{4.80}$$

This implies that the eigenfunctions act as the features.

4.3.7.3 Properties of Kernel

We cite some useful properties of the kernel function below. The set of kernels is closed under some operations. If K, K' are kernels, then:

- $K + K'$ is a kernel.
- aK is a kernel, if $a > 0$.
- $aK + bK'$ is a kernel, if $a, b > 0$.

Using these properties, we can construct different complex kernels from simple ones. However, to utilize the power of kernels, we must pay attention whether the kernels that we can construct are good or bad ones. A bad kernel can be identified from the kernel or gram matrix. The gram matrix of a bad kernel would be mostly diagonal as shown in (4.81). This indicates that all points are orthogonal to each other, indicating no clusters or structures. In other words, if we try to map in a space with too many irrelevant features, the kernel matrix becomes diagonal. Therefore, to choose a good kernel, we always need some prior knowledge of the target.

$$K_{bad} = \begin{bmatrix} 1 & 0 & 0 & \dots & 0 \\ 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & 1 \end{bmatrix}. \tag{4.81}$$

4.3.7.4 Typical Kernel Functions

Some typical kernel functions are mentioned below.

- Linear kernel: $\psi(\mathbf{x}, \mathbf{x}_k) = \mathbf{x}_k^T \mathbf{x}$.
- Polynomial kernel of degree p: $\psi(\mathbf{x}, \mathbf{x}_k) = (\mathbf{x}_k^T \mathbf{x} + 1)^p$.
- Radial basis function (RBF) kernel:

$$\psi(\mathbf{x}, \mathbf{x}_k) = \exp\left[\frac{-\|\mathbf{x} - \mathbf{x}_k\|^2}{2\sigma^2}\right].$$

- Two layer neural kernel: $\psi(\mathbf{x}, \mathbf{x}_k) = \tanh\left[a\mathbf{x}_k^T \mathbf{x} + b\right]$.

A kernel function is defined in the input space and no explicit mapping into the higher dimensional feature space is required by using it. Using a kernel function $K(\mathbf{x}_i, \mathbf{x}_j)$ in the input space, we can avoid the mapping in to a higher dimensional feature space. For any given training data vectors, the required (scalar) inner products are calculated by computing the kernel.

We see an example of mapping into the polynomial feature space defined as

$$\psi: \mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2) \rightarrow \tilde{\mathbf{x}} = (\mathbf{x}_1^2, \sqrt{2}\mathbf{x}_1\mathbf{x}_2, \mathbf{x}_2^2). \tag{4.82}$$

In this feature space, the inner products of two vectors $\tilde{\mathbf{x}}$ and $\tilde{\mathbf{x}}_k$ are given as

$$\tilde{\mathbf{x}}^T \tilde{\mathbf{x}}_k = \mathbf{x}_1^2 \mathbf{x}_{k1}^2 + 2\mathbf{x}_1\mathbf{x}_2\mathbf{x}_{k1}\mathbf{x}_{k2} + \mathbf{x}_2^2\mathbf{x}_{k2}^2 = (\mathbf{x}^T\mathbf{x}_k)^2 = \psi(\mathbf{x}, \mathbf{x}_k). \tag{4.83}$$

In general, the mapping of an input vector $\mathbf{x} \in \mathbf{R}^d$ into a polynomial space with degree p implies the dimension $\begin{pmatrix} p+d-1 \\ d \end{pmatrix}$ of the feature space.

Any symmetric function $K(\mathbf{x}, \mathbf{y})$ in input space can represent a scalar product in feature space if

$$\int\int K(\mathbf{x}, \mathbf{y}) f(\mathbf{x}) f(\mathbf{y}) d\mathbf{x} d\mathbf{y} > 0, \tag{4.84}$$

where, $f(\mathbf{x})$ or $f(\mathbf{y})$ are any function with a finite L_2 norm in input space, i.e., a function for which

$$\int f^2(\mathbf{x}) d\mathbf{x} < \infty. \tag{4.85}$$

In previous sections, we discussed about the computation of the optimal boundary functions. We see them in light of the kernel functions below.

Linear case:

$$\begin{aligned} f(\mathbf{x}) &= \mathbf{w}^T \mathbf{x} + b, \\ \mathbf{w} &= \sum_{k=1}^{n} \alpha_k y_k \mathbf{x}_k, \\ \Rightarrow f(\mathbf{x}) &= \sum_{k=1}^{n} \alpha_k y_k \mathbf{x}_k^T \mathbf{x} + b. \end{aligned} \tag{4.86}$$

Nonlinear case:

$$\begin{aligned} f(\mathbf{x}) &= \mathbf{w}^T \mathbf{x} + b, \\ \mathbf{w} &= \sum_{k=1}^{n} \alpha_k y_k \psi(\mathbf{x}_k), \\ \Rightarrow f(\mathbf{x}) &= \sum_{k=1}^{n} \alpha_k y_k \psi^T(\mathbf{x}_k) \psi(\mathbf{x}) + b. \end{aligned} \tag{4.87}$$

The term $\mathbf{x}_k^T \mathbf{x}$ in (4.86) is an example of a linear kernel. The term $\psi^T(\mathbf{x}_k)\psi(\mathbf{x})$ in (4.87) is an example of a nonlinear kernel.

4.3.7.5 Kernel Hilbert Space

In the SVM literature, the aforesaid feature space (higher dimensional) is usually referred to as the kernel Hilbert space, named after David Hilbert. We will briefly review about the vector spaces and the kernel Hilbert space in the following section.

A norm represents the distance of a vector in $\mathbf{R}^2$ or $\mathbf{R}^3$, as shown in Fig. 4.18. Therefore, for $x = (x_1, x_2) \in \mathbf{R}^2$, the norm is $\|x\| = \sqrt{x_1^2 + x_2^2}$. For,

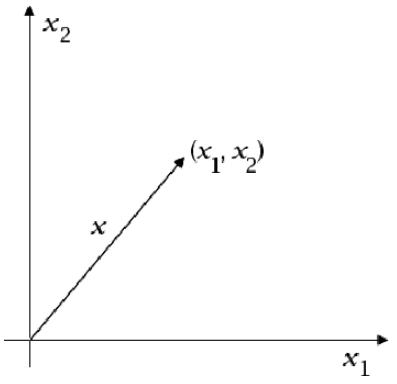

Fig. 4.18 Norm in $\mathbf{R}^2$

$x = (x_1, x_2, x_3) \in \mathbf{R}^3$, the norm is $\|x\| = \sqrt{x_1^2 + x_2^2 + x_3^2}$. In general, for $x = (x_1, x_2, \ldots, x_n) \in \mathbf{R}^n$, the norm is $\|x\| = \sqrt{x_1^2 + x_2^2 + \ldots + x_n^2}$.

However, the norm is not linear on $\mathbf{R}^n$. To impose linearity, we need to use the dot product. For $x, y \in \mathbf{R}^n$, the dot product is defined as

$$x.y = x_1y_1 + x_2y_2 + \ldots + x_ny_n. \tag{4.88}$$

The dot product is a number in $\mathbf{R}^n$, not a vector. Clearly, $x.x = \|x\|^2, \forall x \in \mathbf{R}^n$. Instead of being real, if x is complex, we can define

$$x = a + jb. \tag{4.89}$$

The magnitude $|x|$, and the complex conjugate x^* are defined as

$$\|x\| = \sqrt{a^2 + b^2}, \tag{4.90}$$

$$x^* = a - jb. \tag{4.91}$$

Equations (4.89–4.91) are related as

$$xx^* = \|x\|^2. \tag{4.92}$$

For $x = (x_1, x_2, \ldots, x_n) \in \mathbf{C}^n$, the norm is given as

$$\|x\| = \sqrt{|x_1|^2 + |x_2|^2 + \ldots + |x_n|^2}. \tag{4.93}$$

As x is complex, and we need a nonnegative number, we can think of the norm $\|x\|^2$. Using (4.92–4.93),

$$\|x\|^2 = |x_1|^2 + |x_2|^2 + \ldots + |x_n|^2 = x_1x_1^* + x_2x_2^* + \ldots + x_nx_n^*. \tag{4.94}$$

Similarly to the real space, we can think of $\|x\|^2$ as the inner product of $\|x\|$ with itself. Hence, the inner product of $y = (y_1, y_2, \ldots, y_n) \in \mathbf{C}^n$ with x will be

$$\langle x.y \rangle = y_1x_1^* + y_2x_2^* + \ldots + y_nx_n^*. \tag{4.95}$$

Comparing (4.88) with (4.95), we get that the inner product is a generalization of the dot product.

Thus, for a real or complex vector space H, every inner products produce a norm like

$$\|x\| = \langle x, x \rangle. \tag{4.96}$$

According to the Cauchy criterion, any sequence $\{x_n\}$ in this space is a Cauchy sequence if for every positive real number ε, there exists a natural number N so that

$$\|x_n - x_m\| < \varepsilon, \ \forall m, n > N. \tag{4.97}$$

If H is a Hilbert space, it is complete with respect to the norm in (4.96), i.e., every Cauchy sequence (defined by (4.97)) in H converges to some point. In other words, Hilbert space is a space of infinite dimensions in which distance is preserved by making the sum of squares of coordinates a convergent Cauchy sequence.

Two (or more) Hilbert spaces can be combined into a single Hilbert space by taking their direct sum or their tensor product. Example of Hilbert space is the Euclidian space in L_2. For SVM purpose, the higher dimensional feature space replaced by and represented as the kernel Hilbert space, can be thought of as a generalized Euclidian space that behaves in a gentlemanly fashion (Burges 1998).

4.3.7.6 Kernels & SVM Classifier Optimization

We review the SVM classifier optimization problem in terms of kernel functions in this section. The SVM classifier satisfies

$$\begin{aligned} \mathbf{w}^T \psi(\mathbf{x}_k) + b \geq 1, \quad & if \ y_k = 1, \\ \mathbf{w}^T \psi(\mathbf{x}_k) + b < 1, \quad & if \ y_k = -1. \end{aligned} \tag{4.98}$$

This is equivalent to

$$y_k \left[\mathbf{w}^T \psi(\mathbf{x}_k) + b \right] \geq 1, \quad k = 1, 2, \ldots, n. \tag{4.99}$$

For overlapping classes,

$$y_k \left[\mathbf{w}^T \psi(\mathbf{x}_k) + b \right] \geq 1 - \beta_k, \quad k = 1, 2, \ldots, n, \tag{4.100}$$

where, β_k's are nonnegative slack variables taking into account the overlaps. Involving the penalty parameter μ, the optimization problem becomes

$$\min \left[\frac{1}{2} \mathbf{w}^T \mathbf{w} + \mu \sum_{k=1}^{n} \beta_k \right], \text{ with respect to } (\mathbf{w}, \beta), \tag{4.101}$$

subject to the inequality specified by (4.100). Here, large μ implies discouragement of overlapping, i.e., overfitting, while small μ implies small $\|\mathbf{w}\|$, i.e., smooth boundary.

Introducing the Lagrange multipliers $\alpha_k \geq 0$, $\nu_k \geq 0$ for $k = 1, 2, \ldots, n$, input data points, we get the Lagrangian functional as follows.

$$L = \frac{1}{2} \mathbf{w}^T \mathbf{w} + \mu \sum_{k=1}^{n} \beta_k - \sum_{k=1}^{n} \alpha_k \left[y_k \left(\mathbf{w}^T \psi(\mathbf{x}_k) + b \right) - 1 + \beta_k \right] - \sum_{k=1}^{n} \nu_k \beta_k. \tag{4.102}$$

In the primal space (i.e., large amount of input data sets), solving (4.102) for the saddle points, we get

$$\frac{\partial L}{\partial \mathbf{w}} = 0,\ k = 1, 2, \ldots, n, \tag{4.103}$$
$$\Rightarrow \mathbf{w} = \sum_{k=1}^{n} \alpha_k y_k \psi(\mathbf{x}_k).$$
$$\frac{\partial L}{\partial b} = 0,\ k = 1, 2, \ldots, n, \tag{4.104}$$
$$\Rightarrow \sum_{k=1}^{n} \alpha_k y_k = 0.$$
$$\frac{\partial L}{\partial \beta_k} = 0,\ k = 1, 2, \ldots, n, \tag{4.105}$$
$$\Rightarrow 0 \leq \alpha_k \leq \mu.$$

In the dual space (i.e., high dimensional feature space), the quadratic programming problem becomes

$$\max \sum_{k=1}^{n} \alpha_k - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \psi(\mathbf{x}_i, \mathbf{x}_j),\ \text{with respect to } \alpha_k, \tag{4.106}$$

subject to,

$$\sum_{k=1}^{n} \alpha_k y_k = 0, \tag{4.107}$$
$$0 \leq \alpha_k \leq \mu,\ k = 1, 2, \ldots, n.$$

The kernel function is

$$K(\mathbf{x}_i, \mathbf{x}_j) = \psi^T(\mathbf{x}_i)\psi(\mathbf{x}_j). \tag{4.108}$$

Hence, we do not need to compute the $\mathbf{w}$ or $\psi(\mathbf{x}_k)$ (see (4.103)). That is, we bypass the scalar inner products by computing the kernels. This is a global solution.

4.3.8 Support Vector Regression

In regression, we typically use some measure or error of approximation instead of margin between an optimal separating hyperplane and the SVs. If y and f represent respectively the predicted and the measured values, the typical error functions are given as follows.

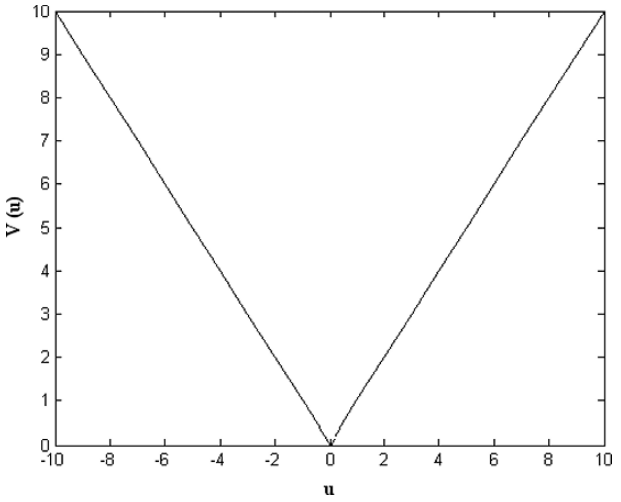

Fig. 4.19 Error function in L_1 norm

- L_1 norm (least modules): $|y - f|$, shown in Fig. 4.19.
- L_2 norm (square errors): $(y - f)^2$, shown in Fig. 4.20.
- Huber's loss function:

$$\frac{1}{2}(y-f)^2, \quad for \quad |y-f| < \xi$$

.

$$\xi\,|y-f| - \frac{\xi^2}{2}, \quad otherwise.$$

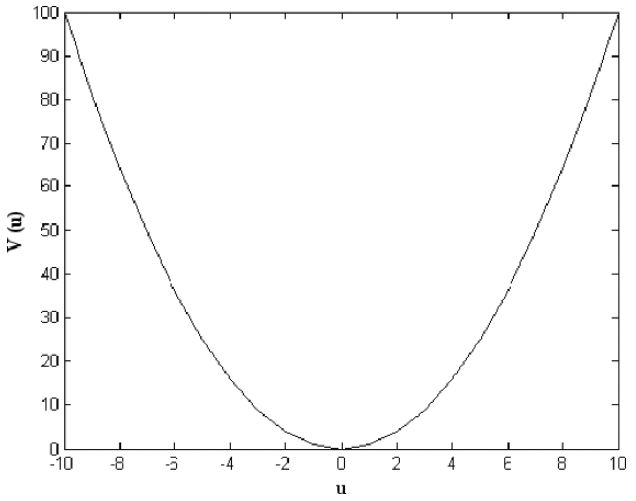

Fig. 4.20 Error function in L_2 norm

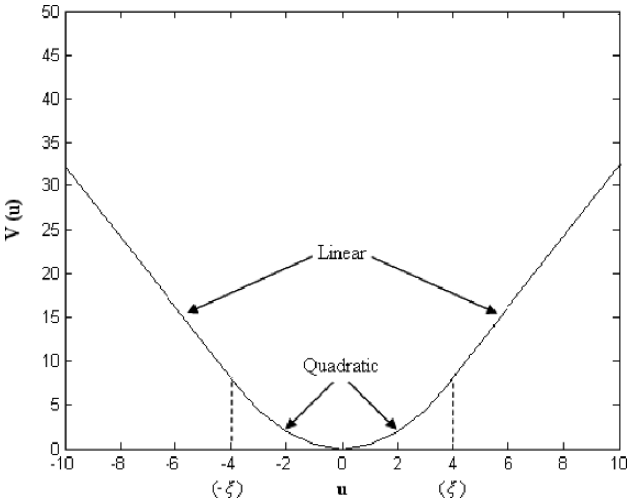

Fig. 4.21 Huber's loss function for *robust regression*

Huber's loss function results in robust regression. It is to be noted that Huber's loss function, shown in Fig. 4.21, is related to the quadratic (square) loss function (Fig. 4.20). If the quadratic loss function (Fig. 4.20) becomes linear after $|\xi|$, it becomes Huber's loss function as in Fig. 4.21.

- Vapnik's linear loss function with ε-insensitivity zone, which is shown in Fig. 4.22.

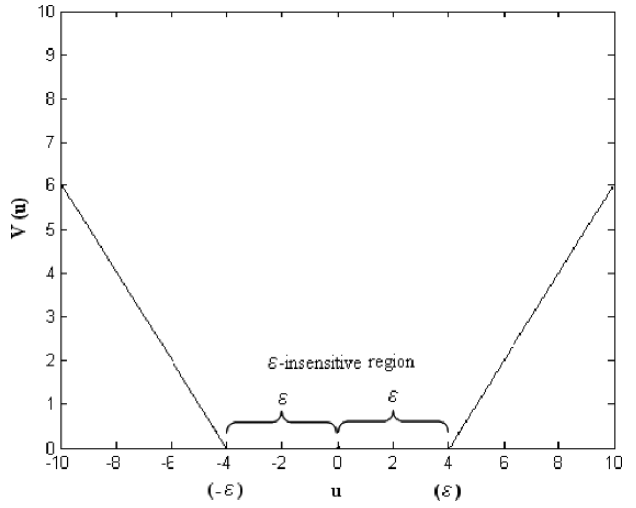

Fig. 4.22 Vapnik's linear loss function with ε-insensitivity zone

4.3.8.1 Linear Regression

For the SVM-based regression operation, we use the linear model

$$\mathbf{y} = f(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b. \tag{4.109}$$

For linear regression, to estimate $\mathbf{w}$, we need to minimize the following objective function,

$$\min \sum_{i=1}^{n} V\,(y_i - f(x_i)) + \frac{\lambda}{2}\|\mathbf{w}\|^2, \text{ with respect to } \mathbf{w}, \tag{4.110}$$

where,

$$V(u) = \begin{cases} 0, & if\ |u| < \varepsilon, \\ |u| - \varepsilon, & otherwise. \end{cases} \tag{4.111}$$

In (4.110), λ is a regularization parameter and can be estimated using cross-validation. In (4.111), ε is the insensitive error measure.

In formulating an SV algorithm for regression, we will try to minimize both empirical risk and $\|\mathbf{w}\|^2$ simultaneously. Figure 4.23 shows the SV-based regression (function approximation). The outliers are marked with squares while the support vectors (denoting sparse fit) are marked as circled.

As shown in Fig. 4.23, Vapnik's ε-insensitive linear loss function defines an ε-tube. If any predicted value lies within the tube, the error is zero. For all other predicted points lying outside the tube, the loss equals the difference between the predicted value and the radius ε of the tube.

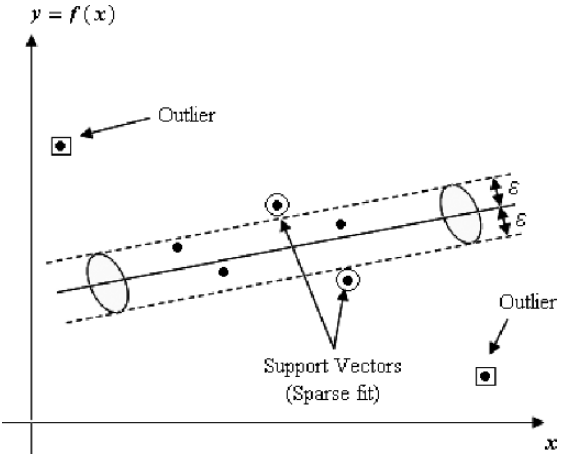

Fig. 4.23 Support vector-based regression using an ε-tube

4.3.8.2 Optimization Problem in Regression

Concerning (4.110–4.111) for SV-regression, the optimization problem can be formulated by introducing the Lagrange multipliers (α_i, α_i^*) like the SV classifier. The two Lagrange multipliers correspond to the data points outside the aforesaid ε-tube. More specifically, α_i and α_i^* correspond to the data points above and below the ε-tube respectively (see Fig. 4.23). Hence, each data point can only lie either above or below ε-tube. Therefore, like SV classifier, the optimization problem in dual space becomes

$$\min \varepsilon \sum_{i=1}^{n}(\alpha_i^* + \alpha_i) - \sum_{i=1}^{n} y_i(\alpha_i^* - \alpha_i) + \frac{1}{2}\sum_{i,j=1}^{n}(\alpha_i^* - \alpha_i)(\alpha_j^* - \alpha_j)\mathbf{x}_i^T\mathbf{x}_j, \tag{4.112}$$

$$\text{with respect to } (\alpha_i, \alpha_i^*),$$

subject to,

$$\begin{aligned} &\sum_{i=1}^{n}\alpha_i^* = \sum_{i=1}^{n}\alpha_i, \\ &\alpha_i^* \le 1/\lambda, \\ &0 \le \alpha_i. \end{aligned} \tag{4.113}$$

Solving (4.112) subject to the constraints given by (4.113) determines the Lagrange multipliers α_i, α_i^*. The regression function is given by (4.109), where

$$\mathbf{w} = \sum_{i=1}^{n}(\alpha_i^* - \alpha_i)\mathbf{x}_i, \tag{4.114}$$

$$b = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - \mathbf{x}_i^T\mathbf{w}\right). \tag{4.115}$$

The support vectors are given by the solutions which satisfy the Karush-Kuhn-Tucker (KKT) condition (Karush 1939; Kuhn and Tucker 1951; Fletcher 1987). These are

$$\alpha_i\alpha_i^* = 0,\ i = 1, 2, \ldots, n. \tag{4.116}$$

That is, the support vectors are given by the data points where exactly one of the Lagrange multipliers is nonzero. After finding the Lagrange multipliers, the optimal weight vector $\mathbf{w}$ and the optimal bias term b can be found using (4.114) and (4.115) respectively.

4.3.8.3 Nonlinear Regression

In most practical cases, a nonlinear model performs better modeling than a linear one. In the same manner as the nonlinear support vector classification approach, a nonlinear mapping can be used to map the data into a high dimensional feature space where linear regression is performed. The kernel approach is again employed to address the curse of dimensionality.

Using the ε-insensitive linear loss function for nonlinear regression, the optimization problem becomes

$$\min \varepsilon \sum_{i=1}^{n}(\alpha_i^* + \alpha_i) - \sum_{i=1}^{n} y_i(\alpha_i^* - \alpha_i) + \frac{1}{2}\sum_{i,j=1}^{n}(\alpha_i^* - \alpha_i)(\alpha_j^* - \alpha_j)K(\mathbf{x}_i, \mathbf{x}_j), \tag{4.117}$$

$$\text{with respect to } (\alpha_i, \alpha_i^*),$$

subject to the constraints given by (4.113). In (4.117), $K(\mathbf{x}_i, \mathbf{x}_j)$ is the kernel function to replace the problematic inner products. Solving (4.117) subject to the constraints given by (4.113) determines the Lagrange multipliers. Then, the regression function is given by

$$f(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b = \sum_{SVs}(\alpha_i - \alpha_i^*)K(\mathbf{x}_i, \mathbf{x}) + b. \tag{4.118}$$

The optimal bias term b in (4.118) can be estimated as

$$b = \frac{1}{2}\sum_{i=1}^{n}(\alpha_i^* - \alpha_i)\left[K(\mathbf{x}_i, \mathbf{x}_r) + K(\mathbf{x}_i, \mathbf{x}_s)\right]. \tag{4.119}$$

4.3.8.4 Regression Example

We show an example of nonlinear regression below. The input (X) and the output (Y) data for the regression are shown in Table 4.1.

The nonlinear regression functions using different kernels to map the input and the output data are shown in Fig. 4.24 to 4.27.

For the simulations, the LS-SVM toolbox (Pelckmans et al. 2003), which is based on the least square support vector machine, has been used in MATLAB® environment.

Table 4.1 Input and output data for nonlinear regression

X	Y
2	180
3	135
4	80
5	51
6	58
7	68
9	83
11	133
14	172
17	214

4.3.9 Procedure to use SVM

We've seen so far the theories behind the statistical learning theory, support vector machines for classification and regression tasks. In this section, we briefly describe the procedures to apply the SVMs in practice for classification and regression task.

1. **Scaling**. The input and the output data need to be scaled properly for better performance. This is important as the kernel values depend on the inner products of the feature vectors. Hence, large attribute values might cause numerical problems (Hsu et al. 2003). Simple linear scaling, like the input and the output data range having bounds like $[-1, 1]$ or $[0, 1]$, provides good result.

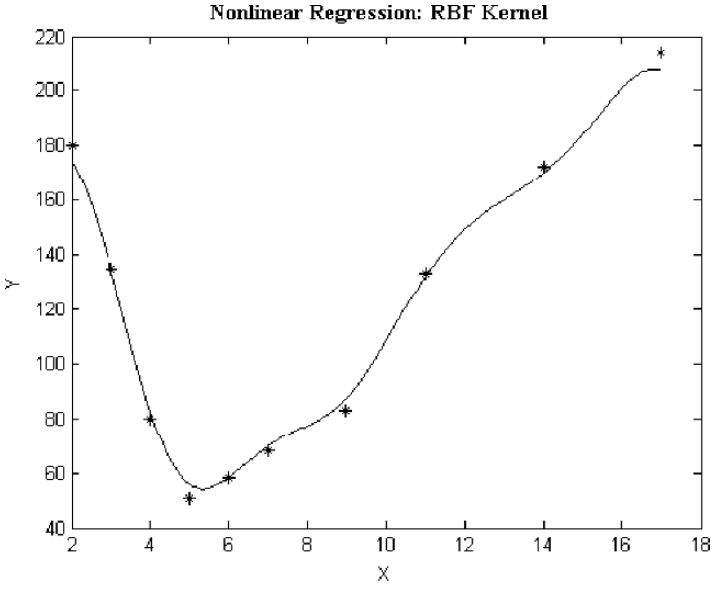

Fig. 4.24 Nonlinear regression using radial basis function kernel

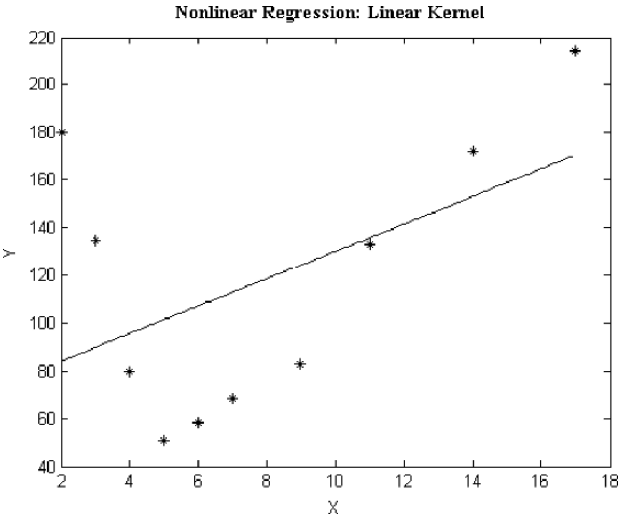

Fig. 4.25 Nonlinear regression using linear kernel

2. **Choosing the kernel**. The next step is to choose an appropriate kernel function (see Sect. 4.3.7.4 for different kernels). The best starting point is to use the RBF kernel. Keerthi and Lin (Keerthi and Lin 2003) showed that if the RBF kernel is used as the basis kernel, there is no need to use the linear kernel. Moreover, the neural kernel involving the sigmoid function may not be positive definite and its accuracy is lower than the RBF kernel (Hsu et al. 2003).
3. **Adjusting the kernel parameters**. The next step is to adjust the kernel parameters, e.g., variance for the RBF kernel, polynomial order for the polynomial

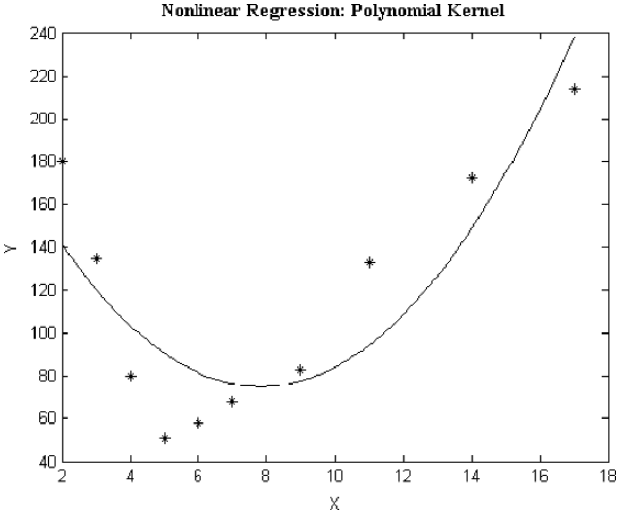

Fig. 4.26 Nonlinear regression using polynomial kernel

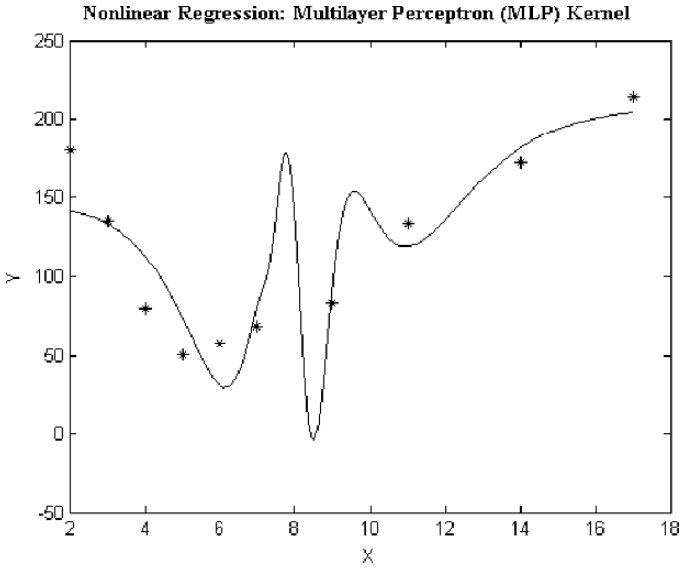

Fig. 4.27 Nonlinear regression using multilayer perceptron kernel

kernel, etc. It is to be noted here that a very high order polynomial might cause computational problem. An optimum value for the kernel parameter of any specific kernel might be achieved using cross-validation.

4. **Training**. Train the SVM using the chosen kernel with optimal parameter, the input and the output (for regression) data. The choice of the appropriate kernel could be influenced by the number of training data. However, in practice, usually cross-validation technique is the best way to judge the appropriateness of the kernel.
5. **Testing**. After training the SVM, test it using the validation data.

4.3.10 SVMs and NNs

Both the support vector machines (SVMs) and the neural networks (NNs) (Chap. 3) map the nonlinear relationship between the input (x_i) and the output (y_i). In this section, we will see a comparative discussion of the SVMs and the NNs.

Classical multilayer perceptron (MLP) operates like

$$e = \sum_{i=1}^{n} (y_i - f(\mathbf{x}_i, \mathbf{w}))^2, \tag{4.120}$$

where, e is the error measure. In (4.120), the right hand side term is a measure of the closeness to data.

Radial basis function (RBF) neural networks operate in the following manner.

$$e = \sum_{i=1}^{n} (y_i - f(\mathbf{x}_i, \mathbf{w}))^2 + \lambda \|\mathbf{P}f\|^2, \tag{4.121}$$

where, $\mathbf{P}$ is the regularization vector. In (4.121), the first term in the right hand side is a measure of the closeness to data similar to (4.120), and the additional second term indicates the smoothness of the operation.

Compared to the MLP and RBF, SVM operates in the following way.

$$e = \sum_{i=1}^{n} (y_i - f(\mathbf{x}_i, \mathbf{w}))^2 + \Omega(l, d, \eta). \tag{4.122}$$

As before, the first term in the right hand side of (4.122) is a measure of the closeness to data and the function Ω in the second term denotes the capacity of the machine. In (4.122), the structural risk minimization principle (SRM) uses the VC dimension d as a controlling parameter for minimizing the generalization error e (i.e., risk).

Vapnik (Vapnik 1995, 1998) basically introduced two basic, constructive approaches to the minimization of the right hand side of (4.122). These are as follows.

1. We choose an appropriate structure, for example, order of polynomials, number of hidden layer neurons in neural network, number of rules in fuzzy logic model etc. Keeping the confidence interval fixed in this way, we minimize the training error, i.e., we perform the empirical risk minimization (ERM). In simpler words, this way we concentrate on the structure hoping that the structure takes care of (minimizes) the associated risk (unseen!).
2. We keep the value of the training error fixed to zero or some acceptable level and minimize the confidence level. This way, we structure a model of the associated risk and try to minimize that (that's SRM!). The result is the optimal structure.

Classical NNs implement the first approach, while the SVMs follow the second strategy. In both cases, the resulting model should resolve the trade-off between under- and over-fitting. Nevertheless, the final model should ideally match the learning machines capacity with training data complexity.

It's evident from above discussion that, in the first ERM approach, as we start with an appropriate structure hoping to tackle the underlying risk of the input-output mapping, we might not be guaranteed that the structure achieves it. That's why NNs many times face the problem of convergence. Also, we do not have any idea about the underlying input-output mapping, hence, the structure, we started with, might get stuck at some local minima, as shown in Fig. 4.28. In comparison, in the second SRM approach, we start with the test error and arrive at an optimal structure. Hence, the SVMs, in general, generalize better. Also, surety of convergence is more for the

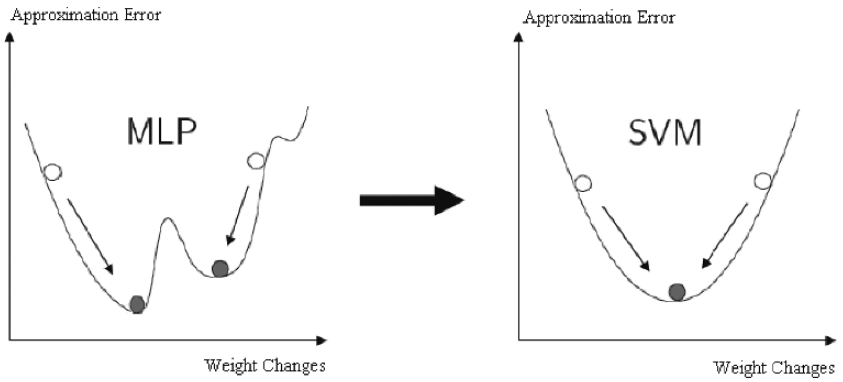

Fig. 4.28 Local minima, convergence problem of NN

SVMs than the NNs. This is due to the fact that the former does not face the problem of getting stuck at local minima (hidden and unpredicted).

We summarize the comparison of the SVM and the NN below.

4.3.10.1 Similarities between SVM and NN

- Both the SVMs and the NNs learn from experimental data, for which, more often in the practical scenario, the underlying probability distribution is not known.
- Struturally, the SVMs and the NNs are same.
- Both the SVMs and the NNs are universal approximators in the sense that they can approximate any function to any desired degree of accuracy.
- After the learning, they are given with the same mathematical model.

4.3.10.2 Differences between SVM and NN

- NNs are based on the ERM principle, starting with an appropriate structure to minimize the error. SVMs are based on the SRM principle, starting with the error to achieve an optimal structure.
- SVMs and NNs follow different learning methods. NNs typically use the back-propagation algorithm (see Sects. 3.3, 3.3.4) or other algorithms like the gradient descent (see Sect. 3.3.2), other linear algebra based approach. The SVMs learn by solving the linear and quadratic programming problem.

4.3.10.3 Advantages of SVM over NN

- SVMs can achieve a trade-off between the false positives and the false negatives using asymmetric soft margins.

- SVMs always converge to the same solution for a given data set regardless of the initial conditions.
- SVMs do not get stuck into the local minima, ensuring convergence of the operation.
- SVMs remove the danger of overfitting.

References

Aizerman M, Braverman E, Rozonoer L (1964) Theoretical foundations of the potential function method in pattern recognition learning. Automation and Remote Control 25: 821–837

Boser BE, Guyon IM, Vapnik V (1992) A training algorithm for optimal margin classifiers. In Proc. 5th Annual ACM Workshop on COLT, D. Haussler, editor, pp. 144–152, ACM Press, Pittsburgh, PA

Burges CJC (1998) A tutorial on support vector machines for pattern recognition. Data Mining and Knowledge Discovery 2:121–167

Drucker H, Burges CJC, Kaufman L, Smola A, Vapnik V (1996) Support vector regression machines. Advances in Neural Information Processing Systems NIPS 1996, pp. 155–161, The MIT Press, Cambridge

Fletcher R (1987) Practical methods of optimization, 2nd edition. John Wiley and Sons, Inc., Chichester

Golub GH, Van Loan CF (1996) Matrix Computations, 3rd edition. Johns Hopkins University Press, Baltimore

Gunn SR (1998) Support vector machines for classification and regression. Technical report, University of Southampton, UK

Hastie T, Tibshirani R, Friedman J (2001) The elements of statistical learning. Springer, New York

Hsu CW, Chang CC, Lin CJ (2003). A practical guide to support vector classification. Technical report, Dept. of Computer Science, National University of Taiwan

Karush W (1939) Minima of functions of several variables with inequalities as side constraints. M.Sc. Dissertation. Dept. of Mathematics, Univ. of Chicago, Chicago, Illinois

Keerthi SS, Lin CJ (2003) Asymptotic behaviors of support vector machines with Gaussian kernel. Neural Computation 15:1667–1689

Kuhn HW, Tucker AW (1951) Nonlinear programming. In Proc. of 2nd Berkeley Symposium, pp. 481–492, University of California Press, Berkeley

Mercer J (1909) Functions of positive and negative type and their connection with the theory of integral equations. Philos. Trans. Royal Society, London

Pelckmans K, Suykens J, Van Gestel T, De Brabanter J, Lukas L, Hamers B, De Moor B, Vandewalle J (2003) LS-SVMlab toolbox user's guide, version 1.5.

Schölkopf B, Smola A (2001) Learning with kernels. The MIT Press, Cambridge

Vapnik V (1995) The nature of statistical learning theory. Springer, New York

Vapnik V (1998) Statistical learning theory. Springer, New York

Vapnik V, Chervonenkis A (1971) On the uniform convergence of relative frequencies of events to their probabilities. Theory of Probability and its Applications 16:264–280

Section II: Application Study

4.4 Fault Classification

4.4.1 Introduction

The analysis of faults and disturbances has always been a fundamental foundation for a secure and reliable electrical power supply. The introduction of digital recording technology, e.g. digital fault recorder (DFR), digital protective relay (DPR), etc, opened up a new dimension in the quantity and quality of fault and disturbance data acquisition, resulting in the availability of a huge amount of new information to power systems engineers. Information from the analysis of digital records, particularly the fault data can provide much-needed insight towards analyzing the disturbance as well as performance of protection equipments.

In power systems, faults are either symmetrical (e.g., three phase short-circuit, three phase-to-ground faults, etc) or nonsymmetrical (e.g., single phase-to-ground, double phase-to-ground, double phase short-circuit, etc). Typical probability of occurrence of different types of faults is shown in Fig. 4.29.

The faults considered in Fig. 4.29 are three phase short-circuit (L-L-L), three phase-to-ground (L-L-L-G), line-line short-circuit (L-L), single line-to-ground (L-G) and double line-to-ground (L-L-G) faults, where the terms 'L' and 'G' refer to 'Line' and 'Ground' respectively. The probabilities associated with a fault type depend upon the operating voltage and can vary from system to system.

4.4.2 Fault Classification

We have already mentioned in the previous section the different types of faults. A particular important task is to perform the classification in an automated way. This

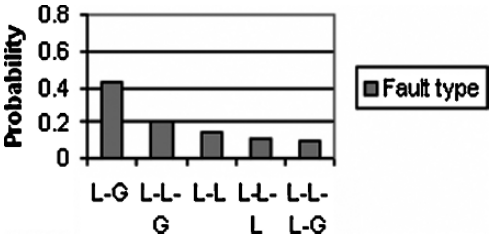

Fig. 4.29 Probability of occurrence of different faults

is particularly important for automated fault analysis (Kezunovic et al. 2001) which supersedes time-consuming and cumbersome manual analysis. The ultimate aim of automatic fault classification is to analyze the fault signals (three-phase voltage and current as well as neural voltage and current signals) to retrieve the following important information (Keller et al. 2005).

- Faulted phase(s)
- Fault type
- Total fault duration
- Fault location
- Fault resistance
- DC offset
- Breaker operating time
- Auto reclose time.

These are typical requirements of automatic fault analyzer, however, additional features might be included. Nevertheless, a good starting point is to have the fault signals classified according to the fault type. As can be seen form the above list, fault classification also answers the first two requirements.

4.4.3 Fault Classifier

Architecture of a fault classifier is shown in Fig. 4.30. The first module takes the fault signal as input. It is to be noted that a good practice is to use the general IEEE COMTRADE (**Com**mon Format for **Tra**nsient **D**ata **E**xchange) standard (IEEE 1991) for the input and output signal type. Use of COMTRADE allows flexible utilizations of variety of sources of transient data, such as DFRs, DPRs, transient simulation programmes from different manufacturers. The second module extracts essential features from the fault signals. These features are utilized in the third module by a SVM-based classifier to perform the automated fault classification task. The output of the classifier could be the information on the faulted phase and the type of the fault.

4.4.3.1 Feature Extraction

Feature extraction concentrates on the fundamental frequency of the fault signal, i.e., the 50 or 60 Hz component depending on the country of operation. Chapter 5 discusses in details the harmonic analysis of a signal. The magnitudes and the phase angle information of the three phase currents are particularly important to design the fault classification algorithm.

It is also important to remove the DC offsets from the fault signals, otherwise that might affect the classification outcome (see Chap. 5, Sect. 5.9.5.4 for more details).

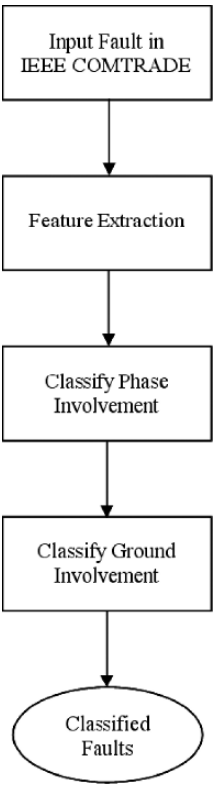

Fig. 4.30 General architecture of the fault classifier

4.4.3.2 Classification Algorithm

Here we focus on SVMs for two-class classification. First we distinguish between single-phase and multi-phase faults. After the determination of the number of phase involvement, we concentrate on whether the fault involved ground or not. So, we will use two-stage SVMs.

The inputs are the phase-currents. For this, we would utilize the sorted three-phase currents, i.e., we categorize the phase currents in a descending order (Henze 2005). This is depicted in Fig. 4.31. This is done to ensure the improved accuracy of the SVM-classifier.

The sufficient statistic for determining the number of phase involvement in the fault is the separating hyperplane between the plane of the maximum and the second highest phase current. The second classifier uses only the second highest and the lowest phase current. For three-phase faults, the lowest value will be also quiet high compared to the two-phase faults. Once we determine the phase involvement, we utilize the neutral currents for the different classes (maximum, second highest, lowest) to find out the involvement of the ground (Henze 2005; Keller et al. 2005). The flowchart of the classification algorithm is shown in Fig. 4.32.

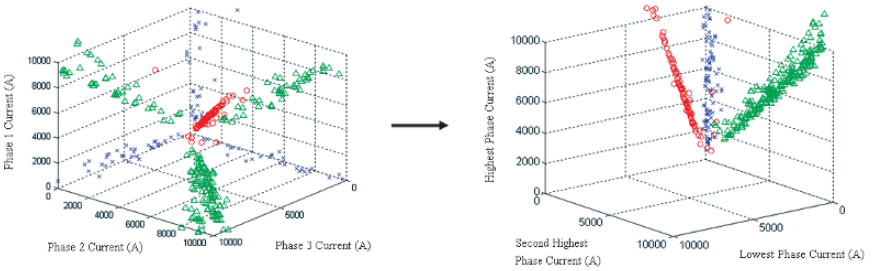

Fig. 4.31 Inputs: sorted three-phase currents

4.4.4 SVM Simulation

For the simulation we will use a least square support vector machine (Suykens et al. 2002) using the LSSVM toolbox (Pelckmans et al. 2003). A least square SVM is in general faster in speed. The modifications for the least square SVM with regard to the standard SVM classifiers are mentioned below.

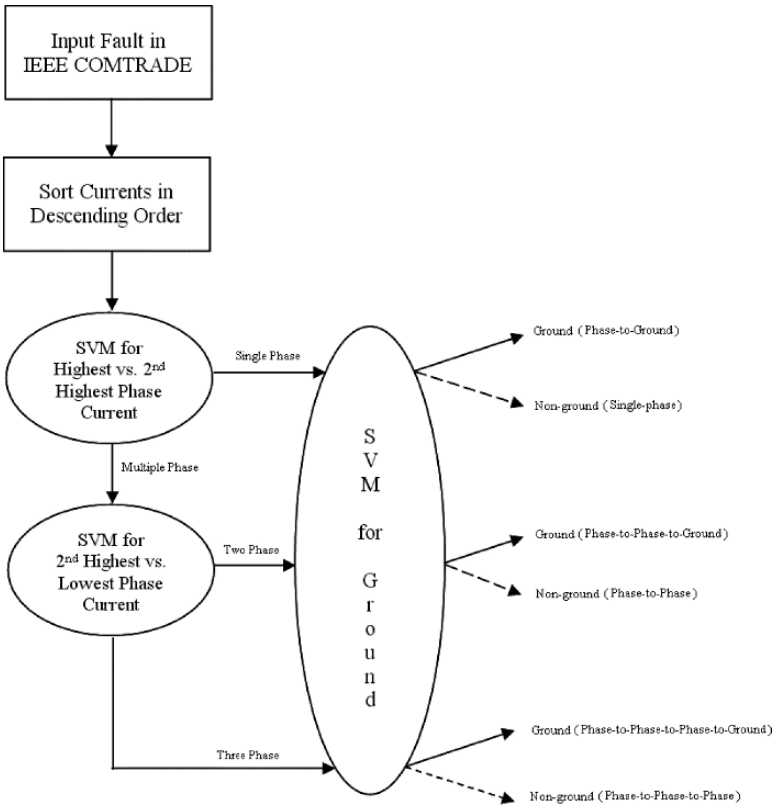

Fig. 4.32 Flowchart of the classification algorithm

- Target values instead of threshold values are used in the constraints.
- Simplifies the classification problem via equality constraints and least squares.

The LSSVM would be trained using about 100 records for each fault type.

4.4.4.1 Phase Involvement Determination

As per the number of phase involvement detection algorithm, we determine the separating hyperplane between the maximum and the second highest phase currents for bifurcating single and multiple-phase involvement. This is shown in Fig. 4.33. In Fig. 4.33, the pink points (closer to the Y-axis of highest phase current) indicate the single-phase involvement, while the white points (closer to the X-axis of second highest phase current) indicate the multiphase involvement.

We further take the multiphase faults (white points in Fig. 4.33) and plot the second highest ones against the lowest phase currents to bifurcate two-phase and

Fig. 4.33 Determining single- or multi-phase involvement

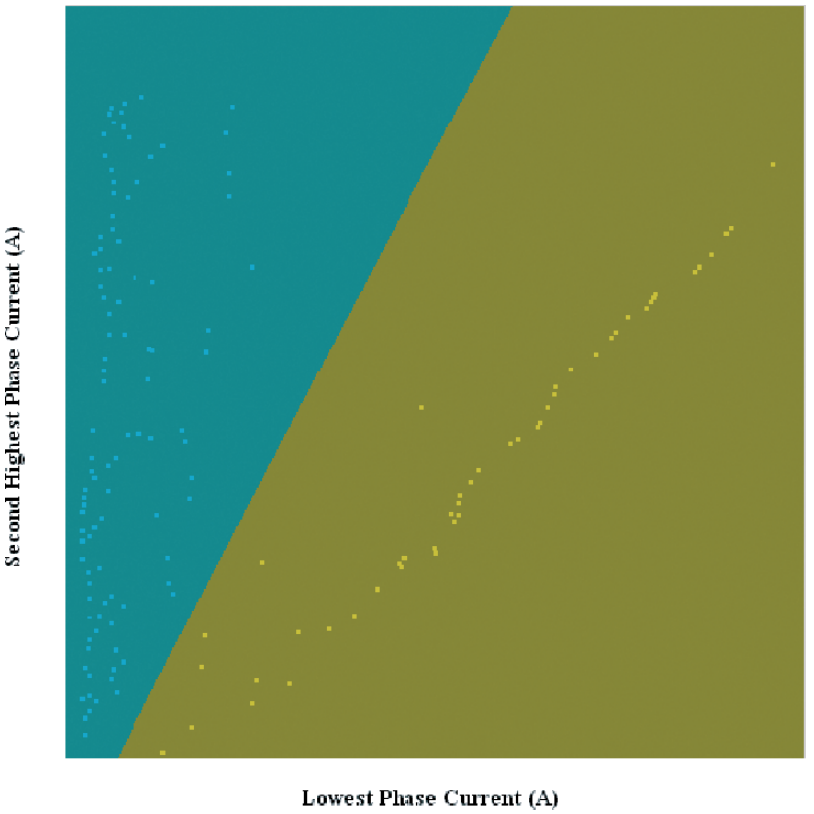

Fig. 4.34 Determining two- or three-phase involvement

three-phase involvement, as shown in Fig. 4.34. In Fig. 4.34, the white points (closer to the Y-axis of second highest phase current) indicate the two-phase involvement. Therefore, the rest of the points, i.e., the yellow points (closer to the X-axis of lowest phase current) indicate involvement of all the three-phases.

4.4.4.2 Ground Fault Determination

After we classify the phase involvements, for each class we utilize the neural currents and seek the separating hyperplane. This is shown in Fig. 4.35. In Fig 4.35, we plot the neutral current against the highest phase current for the class with two-phase involvement. The white points (close to the origin) indicate the non-ground (phase-to-phase in this case) faults, while the yellow dots indicate the involvement of the ground, hence they can be classified as phase-to-phase-to-ground fault (L-L-G in Fig. 4.29).

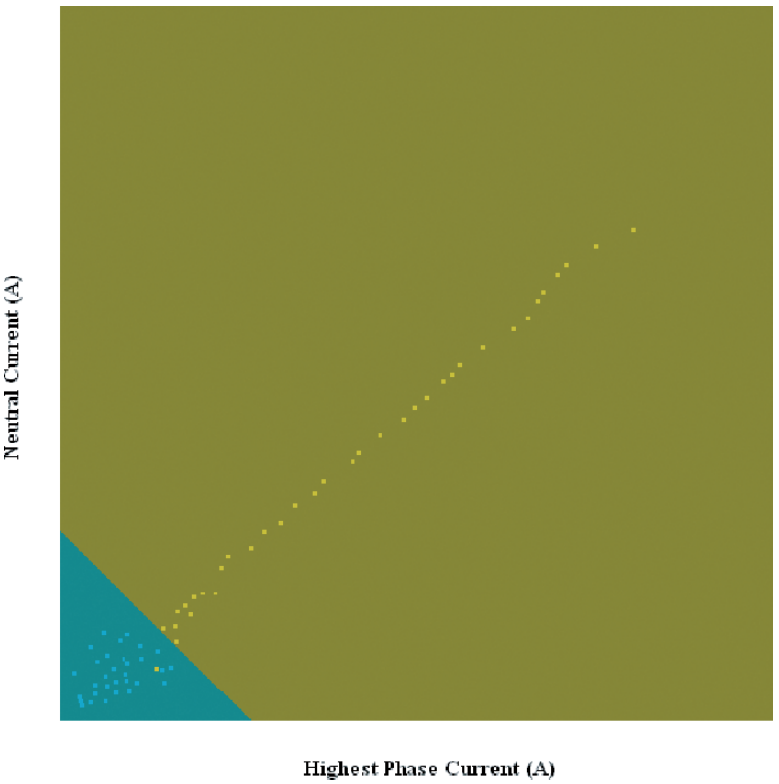

Fig. 4.35 Determining involvement of ground for two-phase faults

References

Henze C (2005) Automatic fault classification using support vector machines. Technical report, Department of Electrical Engineering and Information Technology, Dresden University of Technology, Germany

IEEE standard C37.111–1991 (1991) IEEE standard common format for transient data exchange, version 1.8.

Keller P, Henze C, Zivanovic R (2005) Automated analysis of digital fault records-a utility perspective. In Proc. SAUPEC Conf., Jo'burg, South Africa

Kezunovic M, Chen-Ching L, McDonald JR, Smith L (2001) IEEE tutorial on automated fault analysis.

Pelckmans K, Suykens J, Van Gestel T, De Brabanter J, Lukas L, Hamers B, De Moor B, Vandewalle J (2003) LS-SVMlab toolbox user's guide, version 1.5.

Suykens J, Van Gestel T, De Brabanter J, De Moor B, Vandewalle J (2002) Least squares support vector machines. World Scientific Pub. Co., Singapore

Section III: Objective Projects

4.5 Load Forecasting

Load forecasting is used to estimate the electrical power demand. There are many benefits of such prediction, like, proper generation planning, efficient operation and distribution, interruption-free power supply etc. Several techniques for short-, medium- and long-term load forecasting have been discussed, such as Kalman filters, regression algorithms and neural networks (IEEE 1980; Bunn and Farmer 1985).

4.5.1 Use of SVM in Load Forecasting

SVM is a very useful technique for performing task like non-linear classification, function estimation and density estimation and so on. Load forecasting is an application in the area of function estimation (regression).

4.5.1.1 Input and Output

The input and the output depends on the type of load forecasting. Nevertheless, the input and the output are, in general, load consumption in terms of megawatts (MW).

For short-term load forecasting, the output could be the load prediction for the next hour, and load values of the previous hours can be used as the inputs. We can additionally also use the day and hour information.

For medium- and long-term load forecasting, instead of hourly load input values, we could use monthly, yearly, or multiple year load consumption values as the input and the output in similar ranges. Additional inputs could be seasons, demand profile, industrial growth and so on.

Normalization of the input values often works better for the training purpose.

4.5.1.2 SVM Type

Different linear and nonlinear SVM regression could be utilized. One good candidate is nonlinear radial basis function (RBF) kernel SVM because the inputs and the outputs follow a nonlinear relationship in load forecasting.

4.5.1.3 SVM Tool

Several SVM-libraries for Matlab® are available for function approximation and regression. One good tool is the Least Squares SVM (LS-SVM) toolbox (Pelkmans et al. 2003).

4.5.2 Additional Task

Neural network is a traditional tool for load forecasting. Load forecasting using different types of neural networks have been discussed in details in Chap. 3, Sect. 3.6.

One interesting task would be to compare the performances of the neural networks with the SVM for different categories of load forecasting. Theoretical differences of the NN and the SVM have been addressed in Chap. 4.3.10.

References

Bunn DW, Farmer ED (1985) Comparative models for electrical load forecasting. John Wiley & Sons, New York

IEEE Committee (1980) Load forecasting bibliography phase I. IEEE Transactions on Power Applications and Systems 99:53–58

Pelckmans K, Suykens J, Van Gestel T, De Brabanter J, Lukas L, Hamers B, De Moor B, Vandewalle J (2003) LS-SVMlab toolbox user's guide, version 1.5. Catholic University Leuven, Belgium

4.6 Differentiating Various Disturbances

4.6.1 Magnetizing Inrush Currents

Energizing of a transformer often goes hand in hand with high magnetizing inrush currents. Transformer protection must be set so that the transformer does not trip for this inrush current. This high current is the result of the remnant flux in the transformer core when it was switched out, and depends where on the sine wave the transformer is switched back in (Ukil and Zivanovic 2007). A typical example of magnetizing inrush current is shown in Fig. 4.36.

An interesting application involving the SVM could be to differentiate the magnetizing inrush currents from the normal fault condition (Faiz and Lotfi-Fard 2006). This could be a linear or nonlinear classification task. The two-dimensional feature space could be formed using the magnetizing inrush current and the fault current. Then, linear, Gaussian or RBF kernel SVMs could be used for the classification task.

4.6.2 Power Swing

Power swing is an important phenomenon in the power systems disturbance analysis. Power systems under steady-state conditions operate typically close to their

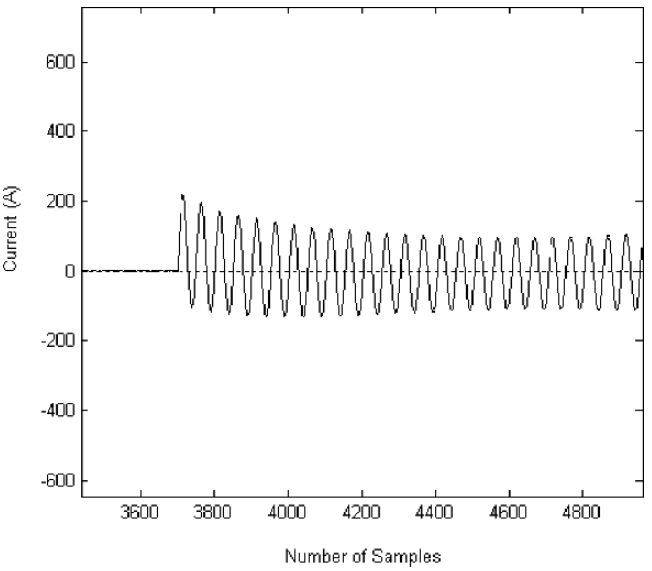

Fig. 4.36 High magnetizing inrush currents associated with transformer energizing

nominal frequency. A balance between generated and consumed active and reactive powers exists during steady-state operating conditions and the sending and the receiving end voltages are within limit (typically 5%). Power system transient events, like, faults, line switching, generator disconnection, and the large load change result in sudden changes to electrical power, whereas the mechanical power input to generators remains relatively constant. These system disturbances cause oscillations in machine rotor angles and can result in severe power flow swings. Depending on the severity of the disturbance and the actions of power system controls, the system may remain stable and return to a new equilibrium state experiencing what is referred to as a stable power swing. Severe system disturbances, on the other hand, could cause large separation of generator rotor angles, large swings of power flows, large fluctuations of voltages and currents, and eventual loss of synchronism between groups of generators or between neighboring utility systems. Large power swings, stable or unstable, can cause unwanted relay operations at different network locations, which can aggravate further the power-system disturbance and possibly lead to cascading outages and power blackouts (Kundur 1994; Taylor 1993).

Under fault condition also the voltage levels are decreased. Figure 4.37 depicts the decrease in the voltage level during power swing (plot i) and under fault condition (plot ii). Differentiation of the power swing from the normal fault condition is usually a difficult task.

SVM-based classification technique could be utilized to differentiate the power swings from the normal fault condition. The classification could be performed using

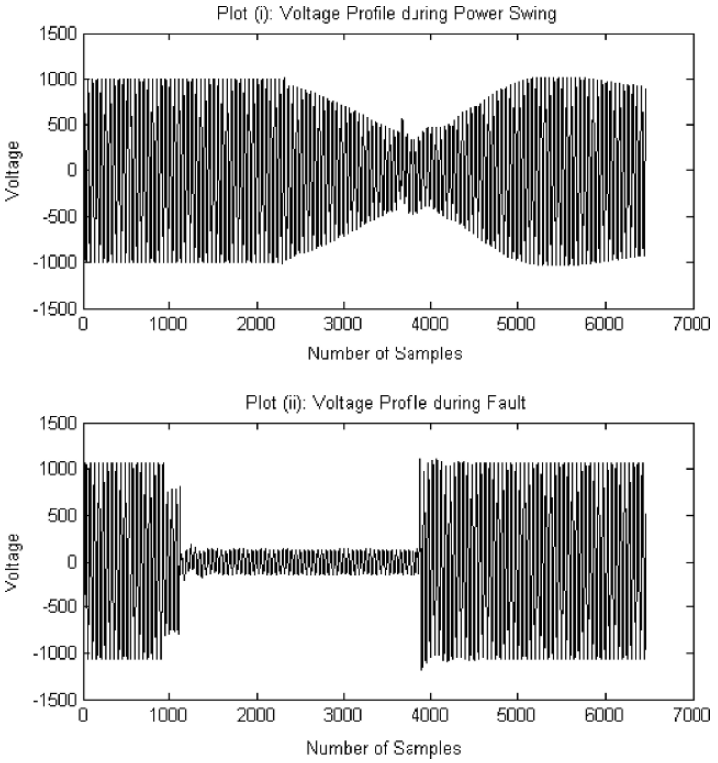

Fig. 4.37 Decrease of voltage level during power swing and fault

Gaussian or nonlinear RBF kernel SVMs in a feature space comprising of the voltage profiles under fault condition and power swing.

4.6.3 Reactor Ring Down

When a line reactor is connected to a line and the circuit-breakers are opened at both ends, the voltage does not disappear. Instead, an oscillating voltage waveform can be found which slowly reduces in magnitude. This phenomenon is called reactor ring down (Ukil and Zivanovic 2007). It is a result of the interaction between the reactor and the capacitance of the line. This forms an oscillatory circuit, a schematic being depicted in Fig. 4.38.

The voltage profile during the reactor ring down phenomenon is shown in Fig. 4.39.

From the discussion of the power swing in the previous section, it is evident that reactor ring down phenomenon adds more complexity to the disturbance differentiation problem. Hence, a composite SVM-based classifier, in conjunction with that from the Sect. 4.6.2, could be designed to differentiate from each other, the reactor

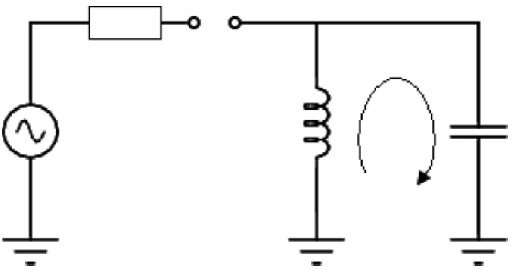

Fig. 4.38 Schematic circuit of the reactor ring down phenomenon

ring down, the power swing and the normal fault condition. A proposed architecture of such a classifier is shown in Fig. 4.40.

As shown in Fig. 4.40, the classification algorithm employs two SVMs. The first SVM differentiates between the fault and the power swing condition. Composite voltage profiles during the fault, power swing and reactor ring down condition can be used as the feature space on which the first SVM operates. The classification algorithm for the SVM-1 operates on the basis of the detection of abrupt changes (Basseville and Nikoforov 2006). As evident from Fig. 4.37, if the voltage profile undergoes abrupt changes then it is indicative of fault condition. For the other case we can further segment the voltage profile and check the parameters of the segments. Here, we employ the second SVM operating on the composite feature space minus the fault signals. The classification algorithm of the SVM should be designed to check gradually decreasing voltage profile segments. That indicates reactor ring down phenomenon, otherwise it will be power swing.

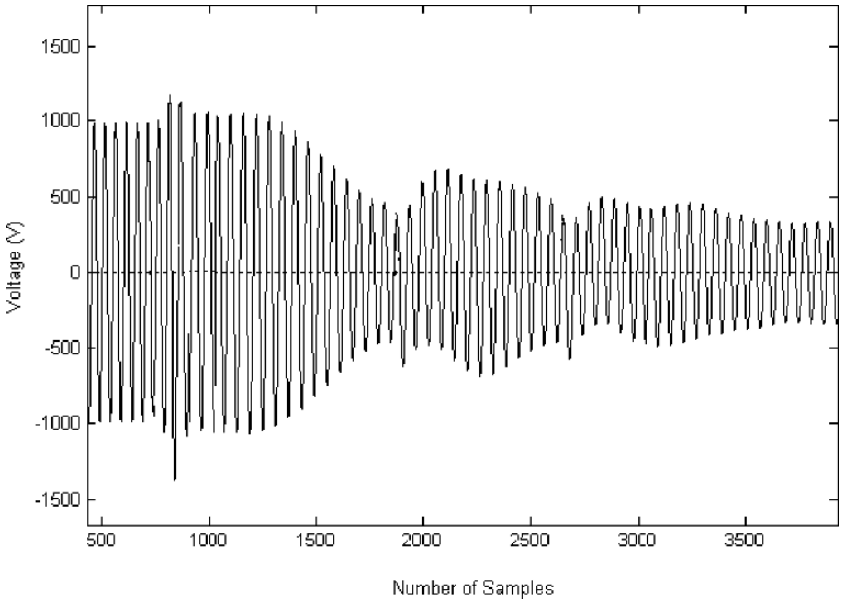

Fig. 4.39 Voltage profile during reactor ring down phenomenon

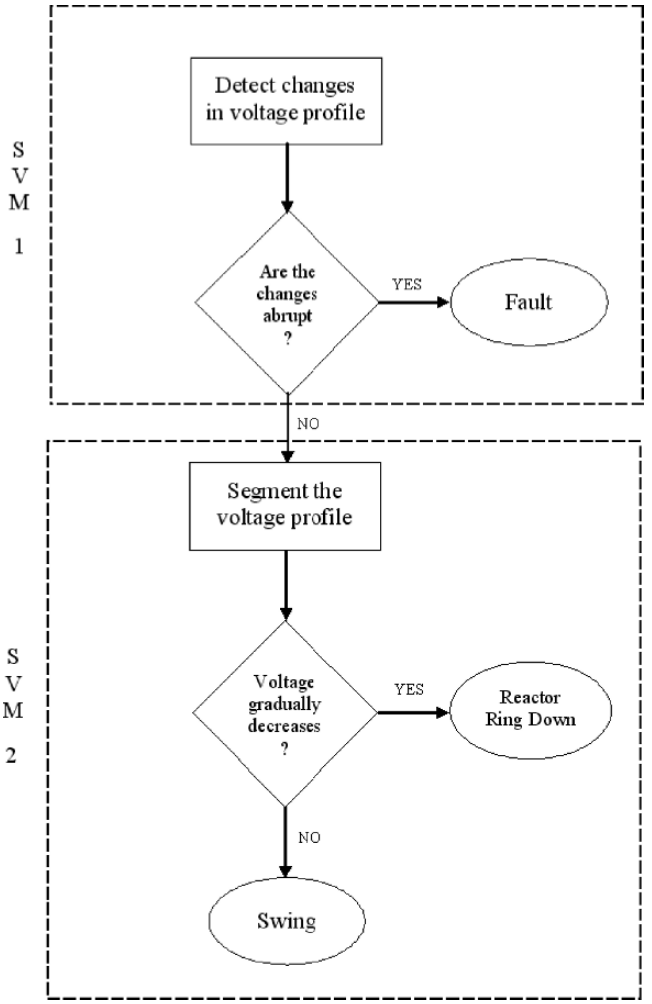

Fig. 4.40 Architecture for SVM-based classifier for reactor ring down, power swing and fault condition

References

Basseville M, Nikoforov IV (1993) Detection of abrupt changes – theory and applications. Prentice-Hall, Englewood Cliffs, NJ

Faiz J, Lotfi-Fard S (2006) A novel wavelet-based algorithm for discrimination of internal faults from magnetizing inrush currents in power transformers. IEEE Transactions on Power Delivery 21:1989–1996

Kundur P (1994) Power system stability and control. McGraw-Hill, New York

Taylor CW (1993) Power system voltage stability. McGraw-Hill, New York

Ukil A, Zivanovic R (2007) Application of abrupt change detection in power systems disturbance analysis and relay performance monitoring. IEEE Transactions on Power Delivery 22:59–66

Section IV: Information Section

4.7 Research Information

This section is organized towards different applications of support vector machines in various power engineering related problems. Research information on general SVM theory and power engineering specific applications in terms of relevant books, publications (journal, conference proceedings), reports, software, etc have been categorized into different sub-sections depending on the applications.

4.7.1 General Support Vector Machine

Burges CJC (1998) A tutorial on support vector machines for pattern recognition. Data Mining and Knowledge Discovery 2:121–167

Cristianini N, Taylor JS (2000) An introduction to support vector machines and other kernel-based learning methods. Cambridge University Press, Cambridge, Available: http://www.support-vector.net

Hastie T, Tibshirani R, Friedman JH (2001) The elements of statistical learning. Springer, New York

Schölkopf B, Burges CJC, Smola AJ (Eds.) (1998) Advances in kernel methods: support vector learning. The MIT Press, Cambridge

Schölkopf B, Smola A (2002) Learning with kernels. The MIT Press, Cambridge

Suykens J, Van Gestel T, De Brabanter J, De Moor B, Vandewalle J (2002) Least squares support vector machines. World Scientific Pub. Co., Singapore

Vapnik V (1995) The nature of statistical learning theory. Springer, New York

Vapnik V (1998) Statistical learning theory. Springer, New York

Vojislav K (2001) Learning and soft computing–support vector machines, neural networks and fuzzy logic models. The MIT Press, Cambridge

4.7.2 Support Vector Machine Software, Tool

(The website addresses provided in this section are checked to be valid till the publishing date. However, those might be changed by their owners. Hence, readers are advised to check the validity.)

Chang CC, CJ Lin (2001) LIBSVM: a library for support vector machines.
Software available from: http://www.csie.ntu.edu.tw/∼cjlin/libsvm
http://www.csie.ntu.edu.tw/∼cjlin/libsvmtools

Gunn SR (1997) Support vector machines for classification and regression. Technical report, Image Speech and Intelligent Systems Research Group, University of Southampton
Software available from: http://www.isis.ecs.soton.ac.uk/resources/svminfo

Kernel Machine: Various resources on kernel machines, support vector machines.
Available from: http://www.kernel-machines.org

Pelckmans K, Suykens J, Van Gestel T, De Brabanter J, Lukas L, Hamers B, De Moor B, Vandewalle J (2003) LS-SVMlab toolbox user's guide, version 1.5.
Software available from: http://www.esat.kuleuven.ac.be/sista/lssvmlab

4.7.3 Load Forecasting

Lin Z, Xian-Shan L, He-Jun Y (2004) Application of support vector machines based on time sequence in power system load forecasting. Power System Technology 28:38–41

Mohandes M (2002) Support vector machines for short-term electrical load forecasting. International Journal of Energy Research 26:335–345

Müller KR (1999) Predicting time series with support vector machines, Advances in kernel methods-support vector learning. MIT Press, MA

Pai PF, Hong WC (2005) Forecasting regional electricity load based on recurrent support vector machines with genetic algorithms. Electric Power Systems Research 74:417–425

Pai PF, Hong WC (2005) Support vector machines with simulated annealing algorithms in electricity load forecasting. Energy Conv. Manage. 46:2669–2688

Pan F, Cheng HZ, Yang JF, Zhang C, Pan ZD (2004) Power system short-term load forecasting based on support vector machines. Power System Technology 28: 39–42

Sun W, Lu JC, Meng M (2006) Application of time series based SVM model on next-day electricity price forecasting under deregulated power market. In Proc. Int. Conf. on Machine Learning and Cybernetics, pp. 2373–2378

Wu H, Chang X (2006) Power load forecasting with least squares support vector machines and chaos theory. In Proc. 6th World Congress on Intelligent Control and Automation, WCICA 2006, pp. 4369–4373

Xu H, Wang JH, Zheng SQ (2005) Online daily load forecasting based on support vector machines. Int. Conf. on Machine Learning and Cybernetics, ICMLC 2005, pp. 3985–3990

Yang J, Stenzel J (2006) Short-term load forecasting with increment regression tree. Electric Power Systems Research 76:880–888

Yang YX, Liu D (2005) Short-term load forecasting based on wavelet transform and least square support vector machines. Power System Tech. 29:60–64

Zhang MG (2005) Short-term load forecasting based on Support Vector Machines regression. Int. Conf. on Machine Learning and Cybernetics, ICMLC 2005, pp. 4310–4314

4.7.4 Disturbance & Fault Analysis

Dash PK, Samantaray SR, Panda G (2007) Fault classification and section identification of an advanced series-compensated transmission line using support vector machine. IEEE Transactions on Power Delivery 22:67–73

Jack LB, Nandi AK (2002) Fault detection using support vector machines and artificial neural networks: augmented by genetic algorithms. Mech. Syst. Signal Process. 16:373–390

Peng L, Da-Ping X, Yi-Bing L (2005) Study of fault diagnosis model based on multi-class wavelet support vector machines. In Proc. Int. Conf. Machine Learning and Cybernetics, 7:4319–4321

Pöyhönen S, Arkkio A, Jover P, Hyötyniemi H (2005) Coupling pairwise support vector machines for fault classification. Control Engg. Practice 13: 759–769

Salat R, Osowski S (2004) Accurate fault location in the power transmission line using support vector machine approach. IEEE Transactions on Power Systems 19:979–986

Thukaram D, Khincha HP, Vijaynarasimha HP (2005) Artificial neural network and support vector machine approach for locating faults in radial distribution systems. IEEE Transactions on Power Delivery 20:710–721

Yan WW, Shao HH (2002) Application of support vector machine non-linear classifier to fault diagnoses. In Proc. 4th World Congress Intelligent Control and Automation, Shanghai, China, pp. 2670–2697

Zhang J, He R (2004) A new algorithm of improving fault location based on SVM. In Proc. IEEE/PES Conf. Power Systems 2:609–612

4.7.5 Transient Analysis

Jayasekara B, Annakkkage UD (2007) Transient security assessment using multivariate polynomial approximation. Elec. Power Sys. Research 77:704–711

Li, D.H.; Cao, Y.J. (2005) SOFM based support vector regression model for prediction and its application in power system transient stability forecasting. In Proc. 7th Int. Conf. on Power Engg., IPEC 2005, 2:765–770

Moulin LS, Alves da Silva AP, El-Sharkawi MA, Marks II RJ (2001) Neural networks and support vector machines applied to power systems transient stability

analysis. International Journal of Engineering Intelligent Systems for Electrical Engineering and Communications 9:205–211
Moulin LS, Alves da Silva AP, El-Sharkawi MA, Marks II RJ (2002) Support Vector and multilayer perceptron neural networks applied to power systems transient stability analysis with input dimensionality reduction. In Proc. of the IEEE Power Engg. Society Transm. Distrib. Conf. 3 (SUMMER):1308–1313
Moulin LS, Alves Da Silva AP, El-Sharkawi MA, Marks II RJ (2004) Support vector machines for transient stability analysis of large-scale power systems. IEEE Transactions on Power Systems 19:818–825
Wang S, Wu S, Li Q, Wang X (2005) v-SVM for transient stability assessment in power systems. In Proc. Autonomous Decentralized Systems pp. 356–363
Zhonghong Y, Xiaoxin Z, Zhongxi W (2005) Fast transient stability assessment based on data mining for large-scale power system. In Proc. IEEE/PES Asia Pacific Conf. Transmission and Distributione, pp. 1–6

4.7.6 Harmonic Analysis

Li M, Kaipei L, Lanfang L (2005) Harmonic and inter-harmonic detecting based on support vector machine. In Proc. IEEE/PES Asia and Pacific Conf. Transm. Distrib., pp. 1–4
Lobos T, Kozina T, Koglin HJ (2001) Power systems harmonics estimation using linear least squares methods and SVD. IEE Proc. Generation, Transmission, Distribution 148:567–572
Zhan Y, Cheng H (2005) A robust support vector algorithm for harmonic and interharmonic analysis of electric power system. Electric Power Systems Research 73:393–400

4.7.7 Power Systems Equipments & Control

Hao L, Lewin PL, Tian Y, Dodd SJ (2005) Partial discharge identification using a support vector machine. Annual Report Conf. on Electrical Insulation and Dielectric Phenomena, pp. 414–417
Li X, Cao G, Zhu XJ (2006) Modeling and control of PEMFC based on least squares support vector machines. Energy Conversion Manage. 47:1032–1050
Schauder C, Mehta H (1991) Vector analysis and control of advanced static VAR compensators. IEE Conference Publication 345:266–272
Shutao Z, Baoshu L, Chengzong P, Jinsha Y (2006) Study on power instrument symbols identifying based on support vector machine. In Proc. Int. Conf. Power System Technology, PowerCon '06, pp. 1–4

4.7.8 Power Systems Operation

Chan WC, Chan CW, Cheung KC, Harris CJ (2001) On the modeling of nonlinear dynamic systems using support vector neural networks. Eng. Appl. Artif. Intell. 14:105–113

Chao Y, Neubauer C, Cataltepe Z, Brummel HG (2005) Support vector methods and use of hidden variables for power plant monitoring. In Proc. IEEE Int. Conf. Acoustics, Speech, Signal Processing, ICASSP '05, 5: v/693–v/696

Gottlieb C, Arzhanov V, Gudowski W, Garis N (2006) Feasibility study on transient identification in nuclear power plants using support vector machines. Nuclear Technology 155:67–77

Liu W, Han ZX (2005) Distribution system reconfiguration based on the support vector machine. Dianli Xitong Zidonghua/Automation of Electric Power Systems 29:48–52

Onoda, T.; Ito, N.; Yamasaki, H. (2006) Unusual condition mining for risk management of hydroelectric power plants. In Proc. 6th IEEE Int. Conf. Data Mining Workshops, ICDM '06, pp. 694–698

Sun W, Shen HY, Yang CG (2006) Comprehensive evaluation of power plants' competition ability with SVM Method. In Proc. Int. Conf. Machine Learning and Cybernetics, pp. 3568–3572

Wang R, Lasseter RH (2000) Re-dispatching generation to increase power system security margin and support low voltage bus. IEEE Transactions on Power Systems 15:496–501

Wang Y, Liu JH, Liu XJ (2006) Modeling of superheated steam temperature using sparse least squares support vector networks. In Proc. Int. Conf. Machine Learning and Cybernetics, pp. 3615–3620

Wang Y, Liu JH, Liu XJ, Tan W (2006) Research on power plant superheated steam temperature based on least squares support vector machines. In Proc. 6th World Cong. Intelligent Control and Automation, WCICA '06 1: 4752–4756

4.7.9 Power Quality

Hu GS, Xie J, Zhu FF (2005) Classification of power quality disturbances using wavelet and fuzzy support vector machines. Int. Conf. Mchine Learning and Cybernetics, ICMLC 2005, 7:3981–3984

Janik P, Lobos T, Schegner P (2004) Classification of power quality events using SVM networks. IEE Conference Publication 2:768–771

Janik P, Lobos T (2006) Automated classification of power-quality disturbances using SVM and RBF networks. IEEE Trans. Power Delivery 21:1663–1669

Lin WM, Wu CH, Lin CH, Cheng FS (2006) Classification of multiple power quality disturbances using support vector machine and one-versus-one approach. In Proc. Int. Conf. Power System Technology, PowerCon '06, l:1–8

Peisheng G, Weilin W (2006) Power quality disturbances classification using wavelet and support vector machines. In Proc. 6th Int. Conf. Intelligent Systems Design and Applications, ISDA '06, 1:201–206

Vivek K, Gopa M, Panigrahi BK (2006) Knowledge discovery in power quality data using support vector machine and S-transform. In Proc. 3rd Conf. Information Technology: New Generations, ITNG 2006, pp. 507–512

4.7.10 Load Flow

Jin M, Renmu H, Hill DJ (2006) Load modeling by finding support vectors of load data from field measurements. IEEE Transactions Power Systems 21:726–735

4.7.11 Power Systems Oscillation

Chen J, Lie TT, Vilathgamuwa DM (2004) Damping of power system oscillations using SSSC in real-time implementation. International Journal of Electrical Power & Energy Systems 26:357–364

Kakimoto N, Sugumi M, Makino T, Tomiyama K (2006) Monitoring of interarea oscillation mode by synchronized phasor measurement. IEEE Transactions on Power Systems 21:260–268

Lee KC, Poon KP (1990) Analysis of power system dynamic oscillations with beat phenomenon by Fourier transformation. IEEE Transactions on Power Systems 5:148–153

4.7.12 Power Systems Security

Andersson, C.; Solem, J.E. (2006) Improvements in classification of power system security. In Proc. IEEE PES General Meeting 6:18–22

Gavoyiannis AE, Vogiatzis DG, Georgiadis DP, Hatziargyriou ND (2001) Combined support vector classifiers using fuzzy clustering for dynamic security assessment. In Proc. of the IEEE Power Engineering Society Transmission and Distribution Conference 2:1281–1286

Kim H, Singh C (2005) Power system probabilistic security assessment using Bayes classifier. Electric Power Systems Research 74:157–165

Marei MI, El-Saadany EF, Salama MMA (2004) Estimation techniques for voltage flicker envelope tracking. Electric Power Systems Research 70:30–37

Sidhu TS, Cui L (2000) Contingency screening for steady-state security analysis by using FFT and artificial neural networks. IEEE Transactions on Power Systems 15:421–426

Wehenkel L (1997) Machine learning approaches to power-system security assessment. Expert IEEE 12:60–72

4.7.13 Power Systems Stability

Andersson C, Solem JE, Eliasson B (2005) Classification of power system stability using support vector machines. In Proc. 2005 IEEE Power Engineering Society General Meeting 1:650–655

Boonprasert U, Theera-Umpon N, Rakpenthai C (2003) Support vector regression based adaptive power system stabilizer. In Proc. – IEEE Int. Symposium on Circuits and Systems 3:III371–III374

Niu L, Zhao J, Du Z, Jin X (2005) Application of time series forecasting algorithm via support vector machines to power system wide-area stability prediction. In Proc. IEEE/PES Transm. Distrib. Conf.: Asia Pacific, Iss. 2005:1–6

Quoc TT, Sabonnadiere JC, Fandino J (1993) Method for improving voltage stability based on critical voltage. In Proc. of the IEEE International Conference on Systems, Man and Cybernetics 3:746–750

4.7.14 Energy Management

Dong B, Cao C, Lee SE (2005) Applying support vector machines to predict building energy consumption in tropical region. Energy and Buildings 37:545–553

4.7.15 Energy Market

Dianmin Z, Feng G, Xiaohong G (2004) Application of accurate online support vector regression in energy price forecast. In Proc. 5th World Cong. Intelligent Control and Automation, WCICA 2004, 2:1838–1842

Zheng H, Zhang L, Xie L, Li X, Shen J (2004) SVM model of system marginal price based on developed independent component analysis. In Proc. Power System Technology Conf., PowerCon '04, 2:1437–1440

4.7.16 Renewable Energy

Hansen T, Wang CJ (2005) Support vector based battery state of charge estimator. Journal of Power Sources 141:351–358

Junping W, Quanshi C, Binggang C (2006) Support vector machine based battery model for electric vehicles. Energy Conversion and Management 47:858–864

Mohandes MA, Halawani TO, Rehman S, Hussain AA (2004) Support vector machines for wind speed prediction. Renewable Energy 29:939–947

4.7.17 Transformers

Ganyun Lv, Cheng H, Zhai H, Dong L (2005) Fault diagnosis of power transformer based on multi-layer SVM classifier. Elec. Power Sys. Res. 75:9–15
Koley C, Purkait P, Chakravorti S (2006) Wavelet-aided SVM tool for impulse fault identification in transformers. IEEE Trans. Power Deliv. 21:1283–1290
Lee FT, Cho MY, Shieh CS, Fang FU (2006) Particle swarm optimization-based SVM application: power transformers incipient fault syndrome diagnosis. In Proc. Int. Conf. on Hybrid Information Technology, ICHIT '06, pp. 468–472
Lv G, Cheng H, Zhai H, Dong L (2005) Fault diagnosis of power transformer based on multi-layer SVM classifier. Electric Power Systems Research 74:1–7

4.7.18 Rotating Machines

Poyhonen S, Negrea M, Arkkio A, Hyotyniemi H, Koivo H (2002) Support vector classification for fault diagnostics of an electrical machine. In Proc. 6th Int. Conf. Signal Processing 2:1719–1722
Poyhonen S, Negrea M, Arkkio A, Hyotyniemi H, Koivo H (2002) Fault diagnostics of an electrical machine with multiple support vector classifiers. In Proc. IEEE Int. Symp. Intelligent Control, pp. 373–378
Thomson WT, Fenger M (2001) Current signature analysis to detect induction motor faults. IEEE Transactions on Industry Applications 7:26–34
Widodo A, Yang BS, Han T (2007) Combination of independent component analysis and support vector machines for intelligent faults diagnosis of induction motors. Expert Systems with Applications 32:299–312
Widodo A, Yang BS (2007) Application of nonlinear feature extraction and support vector machines for fault diagnosis of induction motors. Expert Systems with Applications 33:241–250
Yang BS, Han T, Hwang WW (2005) Application of multi-class support vector machines for fault diagnosis of rotating machinery. Journal of Mechanical Science and Technology 19:845–858

4.7.19 Power Electronics

Al-Khalifa S, Maldonado-Bascon S, Gardner JW (2003) Identification of CO and NO/sub 2/ using a thermally resistive microsensor and support vector machine. Science, Measurement and Technology, IEE Proceedings 150:11–14
Boni, A.; Pianegiani, F.; Petri, D. (2007) Low-power and low-cost implementation of SVMs for smart sensors. IEEE Transactions on Instrumentation and Measurement 56:39–44
Genov R, Cauwenberghs G (2003) Kerneltron: support vector "machine" in silicon. IEEE Transactions on Neural Networks 14:1426–1434

Ling LP, Azli NA (2004) SVM based hysteresis current controller for a three phase active power filter. In Proc. Power Energy Conf., PECon '04, 132–136

Peng D, Lee FC, Boroyevich D (2002) A novel SVM algorithm for multilevel three-phase converters. In Proc. Power Electronics Special. Conf. 2:509–513

Rohwer JA, Abdallah CT, Christodoulou CG (2003) Least squares support vector machines for fixed-step and fixed-set CDMA power control. In Proc. IEEE Conference Decision and Control 5:5097–5102

Ruijian C, Zheng X, Yixin N (2004) A new current control method based on SVM in /spl alpha//spl beta/0 coordinate system. In Proc. IEEE Power Engineering Society General Meeting, pp. 353–357

Sanchis E, Maset E, Carrasco JA, Ejea JB, Ferreres A, Dede E, Esteve V, Jordan J, Garcia-Gil R (2005) Zero-current-switched three-phase SVM-controlled buck rectifier. IEEE Transactions on Industrial Electronics 52:679–688