A Tutorial on Support Vector Machine

Henok Girma

Center of expermental mechanichs

University of Ljubljana

2009

Abstract

Support vector machines (SVMs) are a set of related supervised learning algorithm developed by vladimir vapnik in the mid 90's for classification and regression.It is a new generation learning algorithms based on recent advances in statistical learning theory, and applied to large number of real-world applications, such as text categorization and hand-written character recognition. The elegance and the rigorous mathematical foundations from optimization and statistical learning theory have propelled SVMs to the very forefront of the machine learning field within the last decade.

In this paper we will see a method for appling SVM learning algorisim for data clasification and regression purpose.

1. Introduction

Support Vector Machines, are supervised learning machines based on statistical learning theory that can be used for pattern recognition and regression. Statistical learning theory can identify rather precisely the factors that need to be taken into account to learn successfully certain simple types of algorithms, however, real-world applications usually need more complex models and algorithms (such as neural networks), that makes them much harder to analyse theoretically. SVMs can be seen as lying at the intersection of learning theory and practice. They construct models that are complex enough (containing a large class of neural networks for instance) and yet that are simple enough to be analysed mathematically. This is because an SVM can be seen as a linear algorithm in a high-dimensional space .

In this document, we will primarily concentrate on Support Vector Machines as used in pattern Recognition and function approximation. In the first section we will introduce pattern recognition and hyperplane classifiers, simple linear machines on which SVMs are based. We will then proceed to see how SVMs are able to go beyond the limitations of linear learning machines by introducing the kernel function, which paves the way to find a nonlinear decision function , in this section we will see different kind of kernels and parameter selection. Finally, we will see regression.

2. Pattern Recognition and Hyperplane Classifiers

In pattern recognition we are given training data of the form

$$(\mathbf{x}_1,y_1), \ldots\ldots\ldots,(\mathbf{x}_m,y_m) \in R^n \times \{-1,1\} ,$$

that is n–dimensional patterns (vectors) x_i and their labels y_i. A label with the value of +1 denotes that the vector is classified to class +1 and a label of −1 denotes that the vector is part of class −1.We thus try to find a function $f(\mathbf{x}) = y : R^n \rightarrow \{+1,-1\}$ that apart from correctly classifying the patterns in the training data (a relatively simple task), correctly classifies unseen patterns too.This is called generalisation.

It is imperative that we restrict the class of functions that our machine can learn, otherwise learning the underlying function is impossible. It is for this reason that SVMs are based on the class of hyperplanes

$$\mathbf{w}.\mathbf{x} + b=0 , \ \mathbf{w}\in R^n , b \in R$$

Where the vector **w** defines a direction perpendicular to a hyperplane while varing the value of b move the hyperplane parallel to itself (figure 1). Which basically divide the input space into two: one part containing vectors of the class −1 and the other containing those that are part of class +1 (see Figure 1). If there exists such a hyperplane, the data is said to be linearly separable. To find the class of a particular vector **x**, we use the following decision function

$$f(\mathbf{x}) = \text{sign}(\mathbf{w} \cdot \mathbf{x} + b)$$

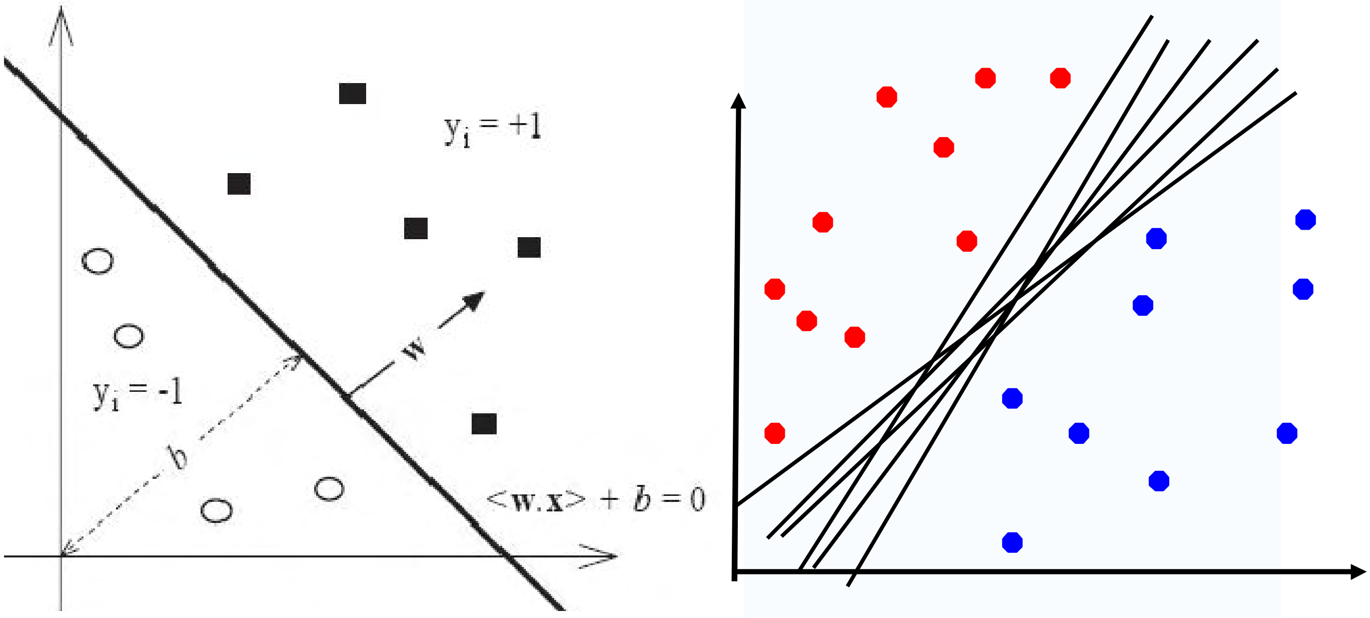


Fig. 1. A separating hyperplanes (**w**, b) for a two dimensional(2D) training set.

2.1 The Optimal Hyperplane

As we can see from the right hand side of figure 1, ther are more than one hyperplane that correctly classifies the training examples. So the qeustion is which of the linear separator is optimal and how can we get this opitimal separtor?

As you can see from figuere 2 below, it has been shown that the hyperplane that guarantees the best generalisation performance is the one with the maximal margin of separation between the two classes ,$\frac{2}{\|\mathbf{w}\|}$. This type of hyperplane is known as the optimal or maximal margin hyperplane and is unique.

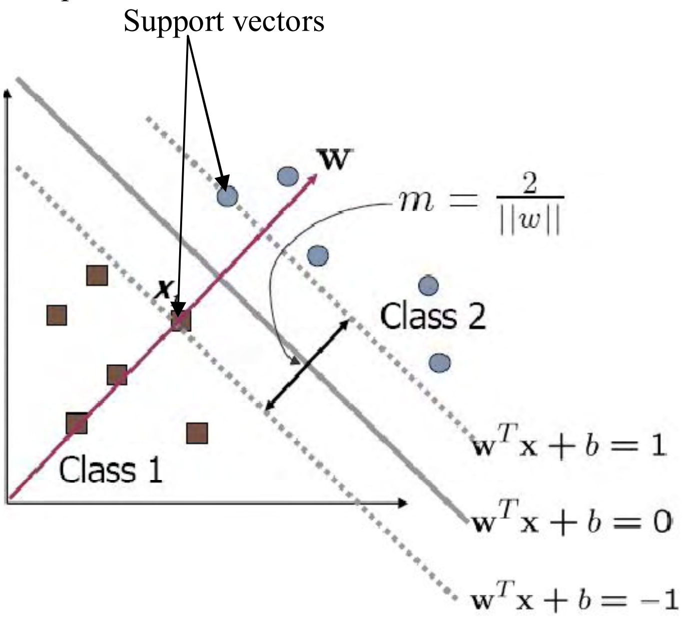

Support vectors
w
$m = \frac{2}{||w||}$
x
Class 2
Class 1
$\mathbf{w}^T\mathbf{x} + b = 1$
$\mathbf{w}^T\mathbf{x} + b = 0$
$\mathbf{w}^T\mathbf{x} + b = -1$

Fig. 2. A maximal margin hyperplane with its support vectors

To calculate the margin, two parallel hyperplanes are constructed, one on each side of the separating hyperplane (figure 2), which are "pushed up against" the two data sets. Intuitively, a good separation is achieved by the hyperplane that has the largest distance to the neighboring datapoints of both classes, since in general the larger the margin the better the generalization error of the classifier.

Assume we are given a set of S of point of $\mathbf{x}_i \in I\Re^n$ with i=1,2,3...m. each point $\mathbf{x}_i$ belongs to either of two classes and thus given a level $y_i \in \{-1,1\}$. The goal is to establish the equation of hyperplane which separates the positive from the negative examples (a “separating hyperplane”) while maximizing the minimum distance between either of the two classes and the hyperplane. For the linearly separable case, the support vector algorithm simply looks for the separating hyperplane with largest margin. This can be formulated as follows:

$$\begin{aligned} \mathbf{w}.\mathbf{x}_i + b &\geq 1 \quad \text{for } y_i = 1 \\ \mathbf{w}.\mathbf{x}_i + b &\leq -1 \quad \text{for } y_i = -1 \end{aligned} \qquad (1)$$

These can be combined into one set of inequalities:

$y_{i}(\mathbf{w}.\mathbf{x}_i + b) - 1 \geq 0 \quad \forall i$ (2)

the pair (**w**,b) define a hyperplane of equation

$\mathbf{w}.\mathbf{x} + b = 0$

named separating hyperplane.The quantity $\frac{2}{\|\mathbf{w}\|}$ which measures the distance between the two class in the direction of **w** is called margin.

$\frac{1}{\|\mathbf{w}\|}$ is the lower bound the distance between the point $\mathbf{x}_i$ and the separating hyperplane $(\mathbf{w},\mathbf{x}_i)$.

So given a linearly separable set **S**, the optimal separable hyperplane is the separating hyperplane which maximize the distance of the closese point of **S**.

Since the distance of the closest point equals to $\frac{1}{\|\mathbf{w}\|}$ the optimal separable hyperplane can be regards as the solution of the problem of maximizing subjected to constaints (2), i.e.

Mimimizing $\frac{\|\mathbf{w}\|2}{2}$

Subjected to:

$y_{i}(\mathbf{w}.\mathbf{x}_i + b) - 1 \geq 0$

Thus we can find the pair of hyperplanes which gives the maximum margin by minimizing $\|\mathbf{w}\|^2$ subject to constraints (2).Thus we expect the solution for a typical two dimensional case to have the form shown in Figure 2 . Those training points for which the equality in Eq. (2) holds (i.e. those which wind up lying on one of the hyperplanes **w.x** + b=1 or **w.x** + b= -1), and whose removal would change the solution found, are called **support vectors**; they are indicated in Figure 2 .

The above constraint problem can be solved by using method of lagrangian multipiers. There are two reasons for doing this. The first is that the constraints in (2) will be replaced by constraints on theLagrange multipliers themselves, which will be much easier to handle. The second is that in this reformulation of the problem, the training data will only appear (in the actual training and test algorithms) in the form of dot products between vectors. This is a crucial property which will allow us to generalize the procedure to the nonlinear case.

Thus, we introduce positive Lagrange multipliers α_i, i=1,2,...,m, one for each of the inequality constraints (2). Recall that the rule is that for constraints of the form $c_i \geq, 0$, the
constraint equations are multiplied by positive Lagrange multipliers and subtracted from the objective function, to form the Lagrangian. For equality constraints, the Lagrange multipliers are unconstrained. This gives Lagrangian:

$L = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{m} \alpha i y i(\mathbf{xi}.\mathbf{w} + b) + \sum_{i=1}^{m} \alpha i$ (3)

Now this is a convex quadratic programming problem,so the solution of the problem is equvalent to determning the saddle points of of the above lagrangian.

$$\frac{\partial L}{\partial b} = \sum_{i=1}^{m} yi\alpha i = 0 \qquad (4)$$

$$\frac{\partial L}{\partial w} = \mathbf{w} - \sum_{i=1}^{m} \alpha iyi\mathbf{xi} = 0$$

$$W = \sum_{i=1}^{m} \alpha iyi\mathbf{xi} \qquad (5)$$

If we Substitute equation (4) and (5) into equation 3 , we get

$$L(\alpha) = \sum_{i=1}^{m} \alpha i - \frac{1}{2}\sum_{i,j}^{m} \alpha i\alpha jyiyj\mathbf{xi.xj} \qquad (6)$$

The new problem is called dual formulation of our first primal problem (equation 3). It has the property that the maximum of L, subject to constraints $\boldsymbol{\alpha} \geq 0$, occurs at the same values of **w**, b and α , as the minimum of L , subjected to constraints

$y_i(\mathbf{w}.\mathbf{x}_i + b) - 1 \geq 0$.

so we can formulated the dual problem as

Maximizing $L(\alpha) = \sum_{i=1}^{m} \alpha i - \frac{1}{2}\sum_{i,j}^{m} \alpha i\alpha jyiyj\mathbf{xi.xj}$

Subjected to $\sum_{i=1}^{m} yi\alpha i = 0$

$\alpha \geq 0$

Note that the only αi that can be non zero in equation (6) are those for which the constarints in (2) are satisfied with equality sign.The corrosponding point $\mathbf{x}_i$,termed **support vectors**,are the points of **S** closes to the optimal separt hyperplane see figure 3 for geometrical interpretation .

So given the parameter **w** ,the parameter b can be obtain as:

b = yi - **w.x**i

Therefore , the problem of classifing a new data point **x** is now simply solved by computing

sign($\sum_{i=1}^{m} \alpha iyi\mathbf{x}i.\mathbf{x}$ + b)

where $\mathbf{x}_i$ is suppor vector.

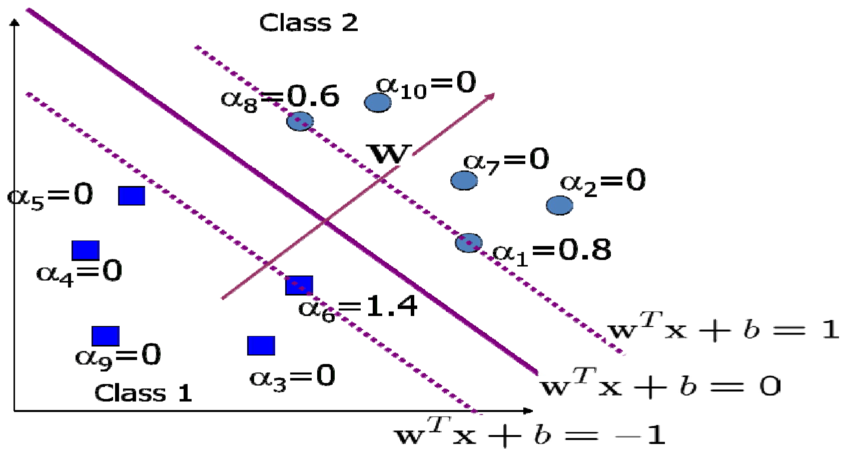


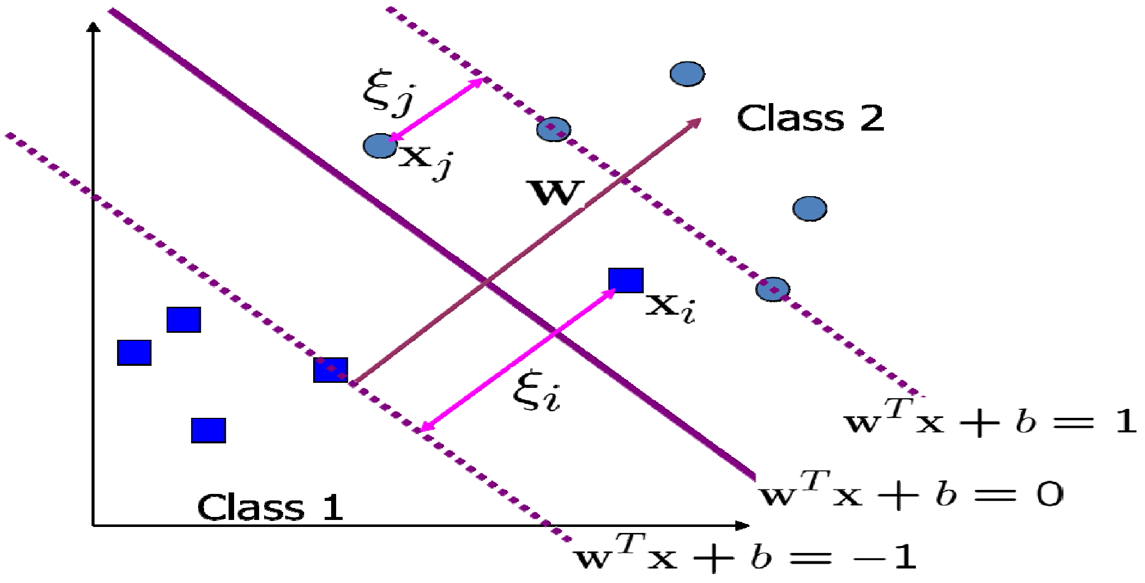
Figure 3 geometrical interpretation of α

As we saw till now basic linear learning machines are only suitable for linearly separable problems. But real world applications often require a more expressive hypothesis space than linear functions can provide.

So how can we manage if the two classes are contain like oulier , noise or some kinde of error and can not be separated by linear clasifier? In the follpwing section we will give answer for this qeustion.

2.2 Soft Margin Classifier

If the training set not linearly separable, the standard approach is to allow the fat decision margin to make a few mistakes (some points - outliers or noisy examples - are inside or on the wrong side of the margin) see figure 4.



ξ_j
$\mathbf{x}_j$
Class 2
$\mathbf{w}$
$\mathbf{x}_i$
ξ_i
$\mathbf{w}^T\mathbf{x}+b=1$
Class 1
$\mathbf{w}^T\mathbf{x}+b=0$
$\mathbf{w}^T\mathbf{x}+b=-1$

Figure 4 soft margin classifier

We then pay a cost for each misclassified example, which depends on how far it is from meeting the margin requirement given in Equation (2). To implement this, we introduce *slack variables* ξ_i . *Slack variables* ξ_i can be added to allow misclassification of difficult or noisy examples, resulting margin called *soft* margin.

A non-zero value for all ξ_i allows $\mathbf{x_i}$ to not meet the margin requirement at a cost proportional to the value of ξ_i. So the condition for the optimal hyper-plane (equation 2) can be relaxed by including ξ_i:

$$y_i(\mathbf{w}.\mathbf{x_i}+b) \geq 1 - \xi_i \qquad (7)$$

if the point $\mathbf{x_i}$ satisfies iniquality to (2) ,then ξ_i is zero and the above equation reduce to equation (2). Instead if $\mathbf{x}$ does not satisfiey iniquality in equation (2), the term ξ_i is subtract from the right side of 2 to obtain the equation (7). The generalized OSH(optimal separable hyperplane) is regards as the solution to:

$$\text{minimizing } \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{m} \xi_i \qquad (8)$$

subjected to $y_i(\mathbf{w}.\mathbf{x_i}+b) \geq 1 - \xi_i$

$$\xi \geq 0$$

The term $\sum_{i=1}^{m} \xi_i$ can be thought of as some measure of amount of misclassicication. In other word the term $\sum_{i=1}^{m} \xi_i$ makes the OSH less sensitive to the presence of outliers in the training set. Here C is a regularization parameter that controls the trade-off between maximizing the margin and minimizing the training error. Small C tends to emphasize the margin while ignoring the outliers in the training data, while large C may tend to overfit the training data.

If we follow similar steps As We did for linear separable case we can see the dual problem is identical to separable case. And its Lagrangian Dual Problem formulation is:

$$\text{Maximizing } L(\alpha) = \sum_{i=1}^{m} \alpha i - \frac{1}{2}\sum_{i,j}^{m} \alpha i\alpha j y i y j \mathbf{xi.xj}$$

Subjected to:

$$\sum_{i=1}^{m} y i \alpha i = 0$$

$0 \leq \alpha_i \leq C$

This is very similar to the optimization problem in the linear separable case, except that there is an upper bound *C* on α_i now.

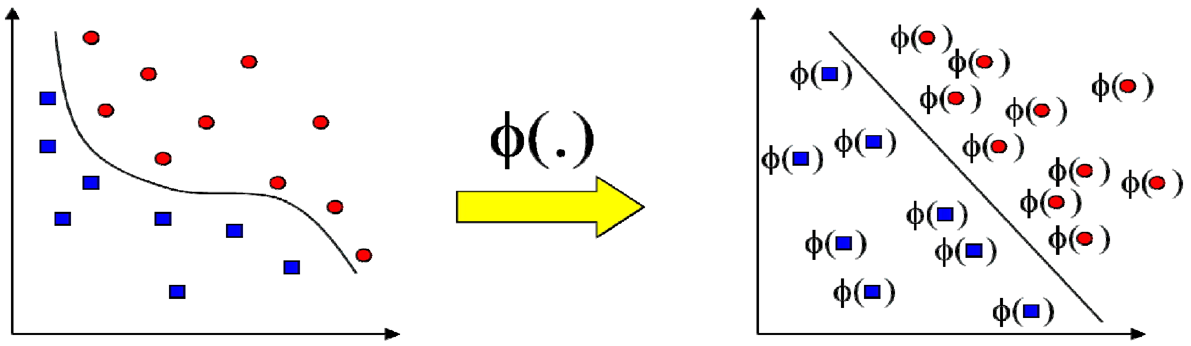
2.3. Nonlinear kernels

Till now we saw data sets that are linearly separable(with some noise) . But what are we going to do if the datasets are to hard to separate linearly?

Linear learning machines (such as the hyperplane classifier) have limited computational power and thus limited real-world value. In general, complex real-world applications require a learning machine with much more expressive power.

In most cases real world problems produce datasets which are hard to do linear separtion in inpute space. Fortunatly ,we can extend the linear sepration theory to nonlinear separating surfaces by mapping the input points into feature points and looking for the OSH in the crosponding future space (cortes and vapnik 1995).

If $\mathbf{x} \in I\Re^n$ is an input point ,we let $\phi(\mathbf{x})$ be the corresponding feature point with ϕ a mapping from $I\Re^n$ to certain space Z called feature space . Clearly ,an OSH in Z corresponds a nonlinear separating surafce in inpute space (figure 5).



input space future space

Figure 5 projecting data that is not linearly separable into a higher dimensional space can make it linearly separable

We have alraedy notice in previous section in SVM formulation ,the training data only appear in the form of the dot products $\mathbf{x_i}.\mathbf{x_j}$ i.e. the optimal hyperplane classifier uses only dot products between vectors in input space. And the same is true in the decision function. In feature space this will translate to $\phi(\mathbf{x}_i)$. $\phi(\mathbf{x}_j)$.

$\Phi: I\Re^n \rightarrow Z$

Clearly, this is very computationally expensive, especially if the mapping is to a high-dimensional space.But many litrature show as kernel function can be used to accomplish the same result in a very simple and efficient way.

A kernel is a function $k(\mathbf{x_i}, \mathbf{x_j})$ that given two vectors in input space, returns the dot product of their images in feature space

$K(\mathbf{x_i}.\ \mathbf{x_j}) = \phi(\mathbf{x}_i).\ \phi(\mathbf{x}_j)$ (9)

So by computing the dot product directly using a kernel function, one avoid the mapping $\phi(\mathbf{x})$. This is desirable because Z has possibly infinite dimensions and $\phi(\mathbf{x})$ can be tricky or impossible to compute. Using a kernel function, one need not explicitly know what $\phi(\mathbf{x})$ is. By using a kernel function, a SVM that operates in infinite dimensional space can be constructed . And the decision function will be :

$$f(\mathbf{x}) = \sum_{i=1}^{m} \alpha i y i K(\mathbf{x}i.\mathbf{x_j}) + b \quad (10)$$

Where for every new test data, the kernel function for each SV need to be recomputed. and A kernel function K is such a function that corresponds to a dot product in some expanded feature space.

But what kind of kernel function is used? i.e. is there any constraint on the type of kernel function suitable for this task. For any kernel function suitable for SVM, there must exist at least one pair of $\{Z,\phi\}$, such that Eqn.(9) is true, i.e. the kernel function represents the dot product of the data in Z.

There are several different kernels, choosing one depends on the task at hand. The commonly used family of kernels are the following:

Polynomail kernel

$K(\mathbf{x_i}, \mathbf{x_j}) = (1 + \mathbf{x_i}.\ \mathbf{x_j})^d$

results in a classifier that is a polynomial of degree p = 2, 3.... in the data. And p is user defined parameter.

The polynomial kernel function is directional, i.e. the output depends on the direction of the two vectors in low-dimensional space. This is due to the dot product in the kernel. All vectors with the same direction will have a high output from the kernel. The magnitude of the output is also dependent on the magnitude of the vector $\mathbf{x_j}$.

Radial basis kernel

Commonly used radial basis kernel is gaussian kernel:

$K(\mathbf{x_i}, \mathbf{x_j}) = \exp\left(-\frac{\|\mathbf{xi}-\mathbf{xj}\|^2}{2\sigma^2}\right)$

Gives a Gaussian radial basis function classifier. The output of the kernel is dependent on the Euclidean distance of $\mathbf{x_j}$ from $\mathbf{x_i}$ (one of these will be the support vector and the other will be the testing data point). The support vector will be the centre of the RBF and σ will determine the area of influence this support vector has over the data space and it is user defined parameter. Larger value of σ will give a smoother decision surface and more regular decision boundary. This is because an RBF with large σ will allow a support vector to have a strong influence over a larger area. A larger σ-value will also reduce the number of support vectors. Since each support vector can cover a larger space, fewer are needed to define a boundary.

Sigmoid kernel:

$$K(\mathrm{x},\mathrm{y}) = \tanh(\kappa \mathrm{x}^T \mathrm{y} + \theta)$$

With user define parameter κ and θ.

It is originated from neural net works and it has similar feature like MLP.

Note: It does not satisfy equation (9) for all κ and θ values.

Example:

Lets see how ϕ look like for polynomial features.

Take polynomial kernel with p=2 and

Lets input be two dimentional vector $\mathbf{x}=[x_1\ \ x_2]$;

i.e. $K(\mathbf{x}_i,\mathbf{x}_j) = (1 + \mathbf{x}_i^T\mathbf{x}_j)^2$,

Need to show that $K(\mathbf{x}_i,\mathbf{x}_j) = \phi(\mathbf{x}_i)^T\phi(\mathbf{x}_j)$:

$K(\mathbf{x}_i,\mathbf{x}_j) = (1 + \mathbf{x}_i^T\mathbf{x}_j)^2$,

$$= 1 + x_{i1}^2x_{j1}^2 + 2\, x_{i1}x_{j1}\,x_{i2}x_{j2} + x_{i2}^2x_{j2}^2 + 2x_{i1}x_{j1} + 2x_{i2}x_{j2}$$

$$= [1\ \ x_{i1}^2\ \ \sqrt{2}\, x_{i1}x_{i2}\ \ x_{i2}^2\ \ \sqrt{2}x_{i1}\ \ \sqrt{2}x_{i2}]^T\, [1\ \ x_{j1}^2\ \ \sqrt{2}\, x_{j1}x_{j2}\ \ x_{j2}^2\ \ \sqrt{2}x_{j1}\ \ \sqrt{2}x_{j2}]$$

$$= \phi(\mathbf{x}_i)^T\phi(\mathbf{x}_j), \quad \text{where } \phi(\mathbf{x}) = [1\ \ x_1^2\ \ \sqrt{2}\, x_1x_2\ \ x_2^2\ \ \sqrt{2}x_1\ \ \sqrt{2}x_2]$$

2.4. Prametr Selection

SVM has only two major parameters which are defined by the user. There is the trade-off between the margin width and the classification error (*C*), and the kernel function. Most kernel functions will also have a set of parameters. The trade-off between maximum margin and the classification error (during training) is defined by the value *C* in Eqn. (8). The value *C* is called the Error Penalty. A high error penalty will force the SVM training to avoid classification errors. A larger *C* will result in a larger search space for the QP optimizer. This generally increases the duration of the QP search. Other experiments with larger numbers of data points (1200) fail to converge when *C* is set higher than 1000. This is mainly due to numerical problems.

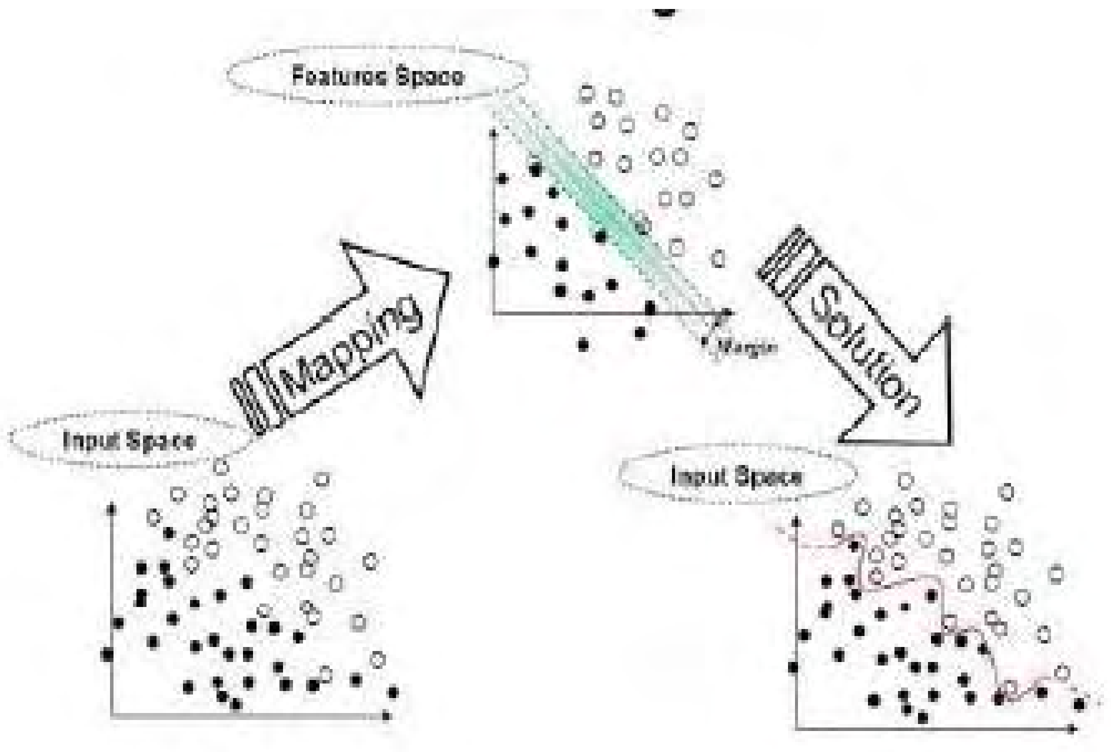
Selecting a specific kernel and parameters is usually done in a try-and-see manner. The followin Simple procedures help us to select these parameters:

- Conduct simple scaling on the data
- Consider RBF kernel , K($\mathbf{x_i}$, $\mathbf{x_j}$)=exp($-\frac{\|xi-xj\|^2}{2\sigma^2}$)
- Use cross-validation to find the best parameter C and σ
- Use the best C and to train the whole training set
- Test

And we can summerize SVM algorithm as follows:

- 1. Choose a kernel function
- 2. Choose a value for *C*
- 3. Solve the quadratic programming problem (many software packages available)
- 4. Construct the discriminant function from the support vectors

Generally we can summerize what we saw till now by using the graph below



3 . Support vector Machine Regression (SVR)

Support Vector method can also be applied to the case of regression, maintaining all the main features that characterize the maximal margin algorithm. The model produced by support vector classification (as described above) only depends on a subset of the training data, because the cost function for building the model does not care about training points that lie beyond the margin. Analogously, the model produced by SVR only depends on a subset of the training data, because the cost function for building the model ignores any training data that are close (within a threshold ε) to the model prediction(see figure 7).

The regression problem can be stated as:

Suppose we are given training data $\{(x_i, y_i) \mid i=1, 2,\dots, m)\}$, of input vectors x_i associated targets y_i, and the goal is to fit a function f(x) which approximates the relation inherited between the data set points and it can be used later on to infer the output y for a new input data point x. Any practical regression algorithm has a loss function L(y, f(x)), which describes how the estimated function deviated from the true one. Many forms for the loss function can be found in the literature. In this tutorial, Vapnik's loss function is used, which is known as" ε - insensitive loss function and defined as:

$$L(y, f(x)) = \begin{cases} 0 & if\ |y - f(x)| \leq \varepsilon \\ (y - f(x)) - \varepsilon & otherwise \end{cases}$$

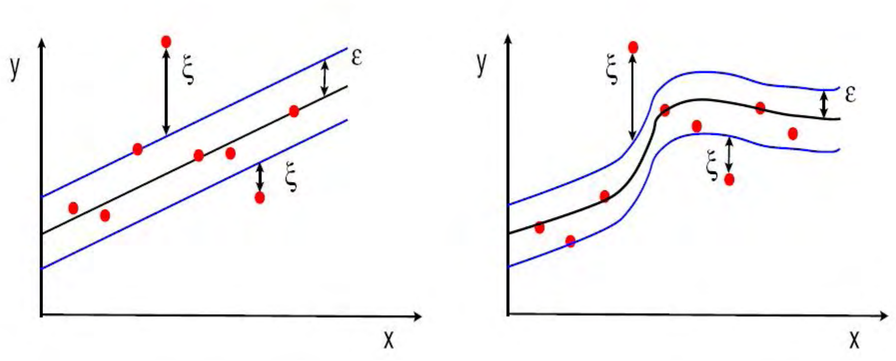


Figure 6 linear and non linear regressions with epsilon intensive band

Where ε > 0 is a predefined constant which controls the noise tolerance. With the ε-insensitive loss function, the goal is to find f(x) that has at most ε deviation from the actually obtained targets y_i for all training data. In other words, the regression algorithm does not care about errors as long as they are less than ε, but will not accept any deviation larger than this. In the same way as with classification approach there is motivation to seek and optimize the generalization bounds given for regression. They relied on defining the loss function that ignores errors, which are situated within the certain distance of the true value. This type of function is often called – epsilon intensive – loss function. Figure (6) above shows an example of linear and non linear regression function with – epsilon intensive – band. The variables measure the cost of the errors on the training points. These are zero for all points that are inside the band, figure (7).

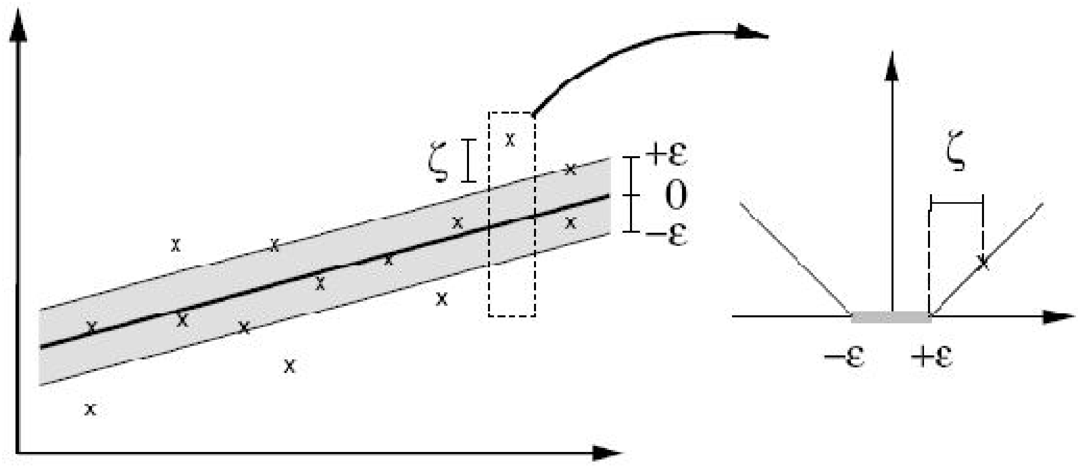


Figure 7 the soft margine loss setting correspondins for linear svm

One of the most important ideas in Support *Vector Classification* and Regression cases is that presenting the solution by means of small subset of training points gives enormous computational advantages. Using the epsilon intensive loss function we ensure existence of the global minimum and at the same time optimization of reliable generalization bound. In general, SVM classification and regression are performed using a nonlinear kernel $K(\mathbf{x}_i, \mathbf{x}_j)$.

For simplicity of notation, Lets see the case of linear functions f, taking the form

$$f(\mathbf{x})= \mathbf{w}.\mathbf{x} + b \tag{11}$$

the regression problem can be written as a convex optimization problem:

minimize $\frac{1}{2}||\mathbf{w}||^2$

subjected to $y_i - (\mathbf{w}.\mathbf{x}_i + b) \leq \varepsilon$ (12)

$(\mathbf{w}.\mathbf{x}_i + b) - y_i \leq \varepsilon$

The assumption in (12) was that such a function f actually exist that approximates all pairs (x,y) with ε precision, or in other words, that the convex optimization problem is feasible. Sometimes however ,this may not be the case,or we also may want to allow for some errors.

Analogously to the soft margin loss function ,one can introduce slack variables ξ_i , ξ^*_i to cope with, otherwise infeasible constraints of the optimization problem (12). So our optimization problem becomes:

Minimize $\frac{1}{2}\|w\|^2 + C\sum_{i=1}^{m} (\xi_i+\xi_i^*)$

Subject to $y_i - (\mathbf{w}.\mathbf{x}_i + b) \leq \varepsilon + \xi_i$ (13)

$(\mathbf{w}.\mathbf{x}_i + b) - y_i \leq \varepsilon + \xi_i^*$

$\xi_i , \xi_i^* \quad \geq 0$

The constant C > 0 determines the trade off between the flatness of f and the amount up to which deviations larger than ε are tolerated.

As shown in Fig.7, only the points outside the shaded region contribute to the cost insofar, as the deviations are penalized in a linear fashion. It turns out that in most cases the optimization problem Eq. (13) can be solved more easily in its dual formulation. Moreover, the dual formulation provides the key for extending SVM machine to nonlinear functions. Hence, a standard dualization method utilizing Lagrange multipliers will be described as follows.

$L=\frac{1}{2}\|w\|^2 + C\sum_{i=1}^{m} (\xi_i+\xi_i^*) - \sum_{i=1}^{m}\alpha_i(\varepsilon+\xi_i - yi + \mathbf{w}.\mathbf{x}i + b) - \sum_{i=1}^{m}\alpha_i^*(\varepsilon+\xi_i^* + yi - \mathbf{w}.\mathbf{x}i + b) - \sum_{i=1}^{m}(\lambda_i\xi_i + \lambda_i^*\xi_i^*)$ (14)

Here L is the Lagrangian and α_i, α_i^* , λ_i , λ_i^* are Lagrange multipliers. Hence the dual variables in Eq. (7) have to satisfy positivity constraints:

$\alpha_i, \alpha_i^* , \lambda_i , \lambda_i^* \geq 0$ (15)

It follows from the saddle point condition that the partial derivatives of L with respect to the primal variables (w, b ,ξ_i , ξ_i^*) have to vanish for optimality. And if we follow similar steps like what we did for classifiction optimization problem above we end up with the dual optimization problem. By deriving the dual problem we already eliminated the dual variables λ_i , λ_i^*. Thus

$f(x)= \sum_{i=1}^{m}(\alpha_i - \alpha_i^*)(x_i . x_j) + b$ (16)

Note that the complete algorithm can be described in terms of dot products between the data. Even when evaluating f(x) we need not compute **w** explicitly. These observations will come in handy for the formulation of a nonlinear extension. For new given **x** the crossponding value given by:

$$f(\mathbf{x}) = \sum_{i=1}(\alpha_i^* - \alpha_i)K(\mathbf{x}_i, \mathbf{x}) + b$$

Only data for which $\alpha_i \neq \alpha_i^*$ contribute.This occurs only if

$|f(\mathbf{x}_i) - y_i| \geq \epsilon$ (outside tube)

4. SVM Vs Neural network

A Support Vector Machine (SVM) performs classification by constructing an *N*-dimensional hyperplane that optimally separates the data into two categories. SVM models are closely related to neural networks. In fact, a SVM model using a sigmoid kernel function is equivalent to a two-layer, perceptron neural network.

Support Vector Machine (SVM) models are a close to classical multilayer perceptron neural networks. Using a kernel function, SVM's are an alternative training method for polynomial, radial basis function and multi-layer perceptron classifiers in which the weights of the network are found by solving a quadratic programming problem with linear constraints, rather than by solving a non-convex, unconstrained minimization problem as in standard neural network training. Generally we can characterized the two as follows.

Neural network

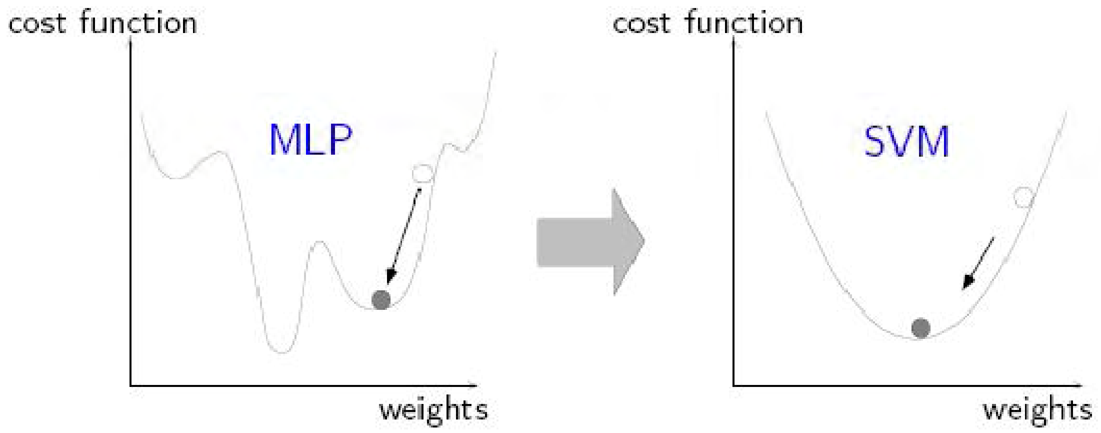
- Universal approximation of continuous nonlinear functions
- Learning from input-output patterns; either off-line or on-line learning
- Parallel network architecture, multiple inputs and outputs
- feedforward and recurrent networks
- supervised and unsupervised learning applications

Problems:

- Existence of many local minima!
- How many neurons and layers needed for a given task?

Support Vector Machine

- Learning involves optimization of a convex function(no false minima, unlike a neural network)



- Nonlinear classification and function estimation by convex optimization with a unique solution and primal-dual interpretations.

- Number of neurons (support vectors) automatically follows from a convex program.
- Learning and generalization in huge dimensional input spaces (able to avoid the curse of dimensionality!).
- Use of kernels (e.g. linear, polynomial,RBF...)

Problem: SVM formulation does not include criteria to select kernel function that give good generalization.so getting kernel function and parameters are some how trial and error.

5. Summary

Support Vector Machines have been applied to many real-world problems, producing state of the art results. These include text categorisation , image classification, and hand written character recognition.

SVMs provide a new approach to the problem of pattern recognition (together with regression estimation and linear operator inversion) with clear connections to the underlying statistical learning theory. They differ radically from comparable approaches such as neural networks: SVM training always finds a global minimum, and their simple geometric interpretation provides fertile ground for further investigation. An SVM is largely characterized by the choice of its kernel, and SVMs thus link the problems they are designed for with a large body of existing work on kernel based methods.

References

Christopher J.C.Burges. A Tutorial on Support Vector Machines forPattern

Recognition, Kluwer Academic Publishers, Boston, 1998.

Massimiliano Pontil and Alessandro Verri.properties of support vector machines,1998.

Pai-Hsuen Chen, Chih-Jen Lin, and Bernhard Scholkopf. A Tutorial on *ν*-Support Vector Machines

Steven Busuttil. Support Vector Machines

Aly Farag and Refaat M Mohamed.regression using support vectot machines,december 2007

Wikipedia, the free enczclopedia.Support vector mahine

http://www.svms.org/tutorials/

http://www.support-vector.net/tutorial.html