Support Vector Machines

Introduction

- **Reference:** *"A Tutorial on Support Vector Machines for Pattern Recognition"* – `Christopher J.C. Burges.`
- Support Vector Machines (SVMs) are regarding a novel way of estimating a non-linear function by using a limited number of training examples.
- It is shown that in most of the cases, SVMs can achieve better performance on independently drawn test sets.
  1. Isolated handwritten digit recognition
  2. object recognition
  3. Face detection in images
  4. Text categorization
  5. Density estimation
  6. Etc. ...

Introduction

- Getting struck in local minima is not there!!
- It shows better generalization ability.
  That is, it can overcome the problem of overfitting.
  - It achieves a right balance between the accuracy attained on a particular training set, and the "capacity" of the machine, that is, the ability of the machine to learn any training set without error.
  - A machine with too much capacity is like remembering all training patterns. This achieves zero training error.
  - A machine with too less capacity is like randomly assigning class label. Or taking only prior probabilities into account. Or something like saying "a green thing is a tree".

Introduction

- SVMs can be said to have started in the late seventies (Vapnik, 1979).
- The books by *Vapnik* published in 1995 (*"The Nature of Statistical Learning Theory"*) , and 1998 (*"Statistical Learning Theory"*) gives detailed description on the subject.
- Only recently SVMs are gaining lot of attention. This is because of its success in various fields (This can be compared to the increased research interest on neural networks when the backpropagation is found in 1986).
- There are numerous extensions for the basic SVMs, like *rough SVM*, *Support Vector Clustering*, etc.

Generalization Ability

- Let the training set is $\{(X_1, y_1), \ldots, (X_n, y_n)\}$ where $X_i \in \mathcal{R}^d$ and $y \in \{+1, -1\}$.
- It is assumed that the training set is drawn i.i.d from a distribution $p(X, y)$, which might be unknown to us.
- We assume form of a function and try to learn its parameters to suit the training set. Let the final function obtained is $f(X; \alpha)$ where α is the set of parameters.
- The overall risk or actual risk $R(\alpha)$ is the error of the classifier over the entire feature space.
- The empirical risk or training risk $R_{emp}(\alpha)$ is the error of the classifier over the training set.

Generalization Ability

- The actual risk of the classifier $f(X;\alpha)$ is

$$R(\alpha) = \int \frac{1}{2} \left| y - f(X;\alpha) \right| p(X,y) dX dy$$

- The empirical risk is

$$R_{emp}(\alpha) = \frac{1}{n} \sum_{i=1}^{n} \frac{1}{2} \mid y_i - f(X_i;\alpha) \mid$$

Generalization Ability

- Risk Bound: The following is from VC-theory, the bound is true with a probability $(1-\eta)$

$$R(\alpha) \quad \leq \quad R_{emp}(\alpha) + \underbrace{\phi\left(\frac{h}{n}, \frac{log(\eta)}{n}\right)}_{\text{VC Confidence}}$$

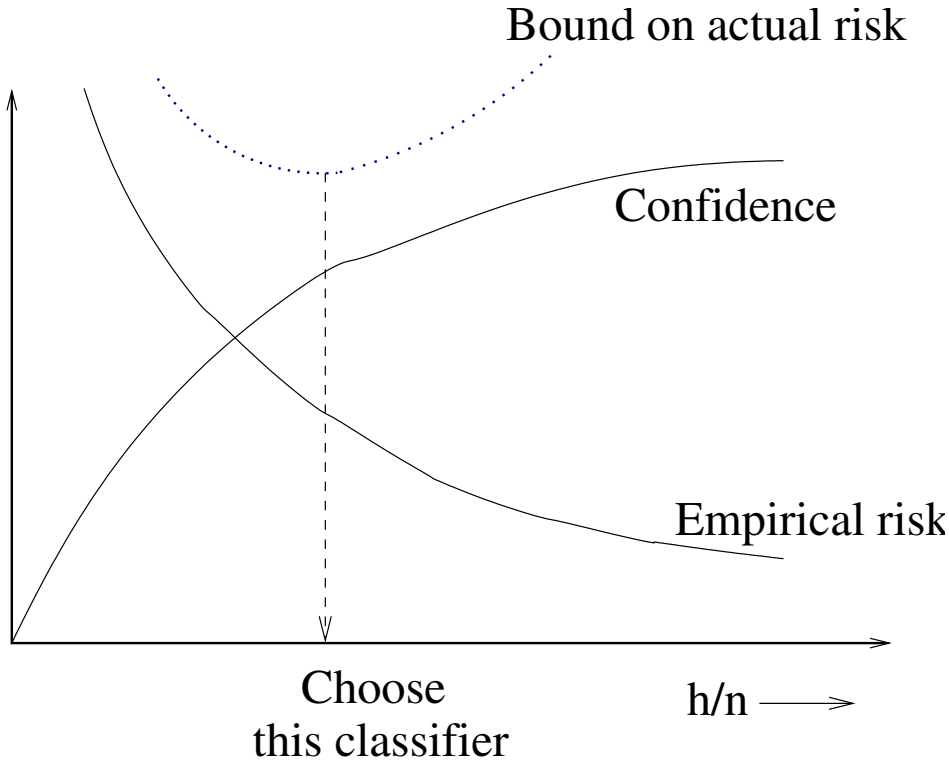
  where h is called the *VC-dimension* for the class of functions $f(\ )$, which is a measure for the *capacity* of the classifier.

- The bound is independent of probability distributions and only assumes that the training and test sets are independently drawn.

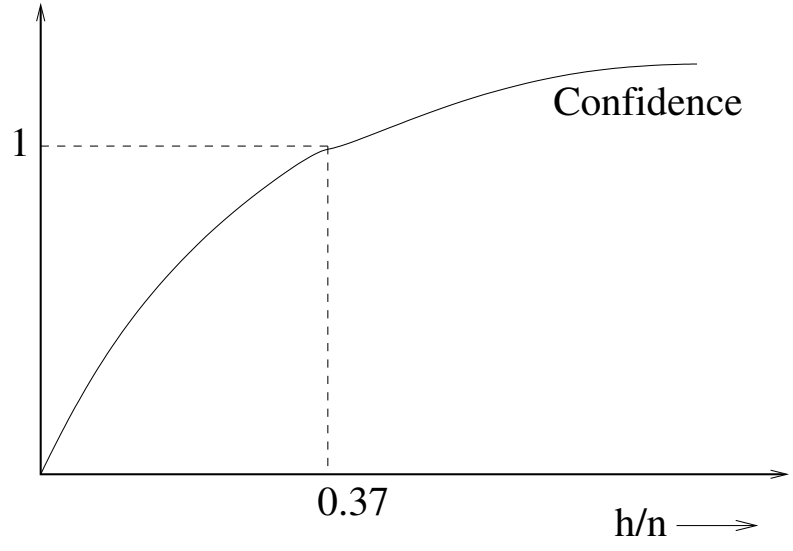
The Bound

- VC-Confidence:

$$\phi\left(\frac{h}{n}, \frac{log(\eta)}{n}\right) = \sqrt{\left(\frac{h(log(2n/h)+1) - log(\eta/4)}{n}\right)}$$



VC Confidence



- The bound is useless when $h/n > 0.37$ (for $\eta = 0.05$ and $n = 10000$).

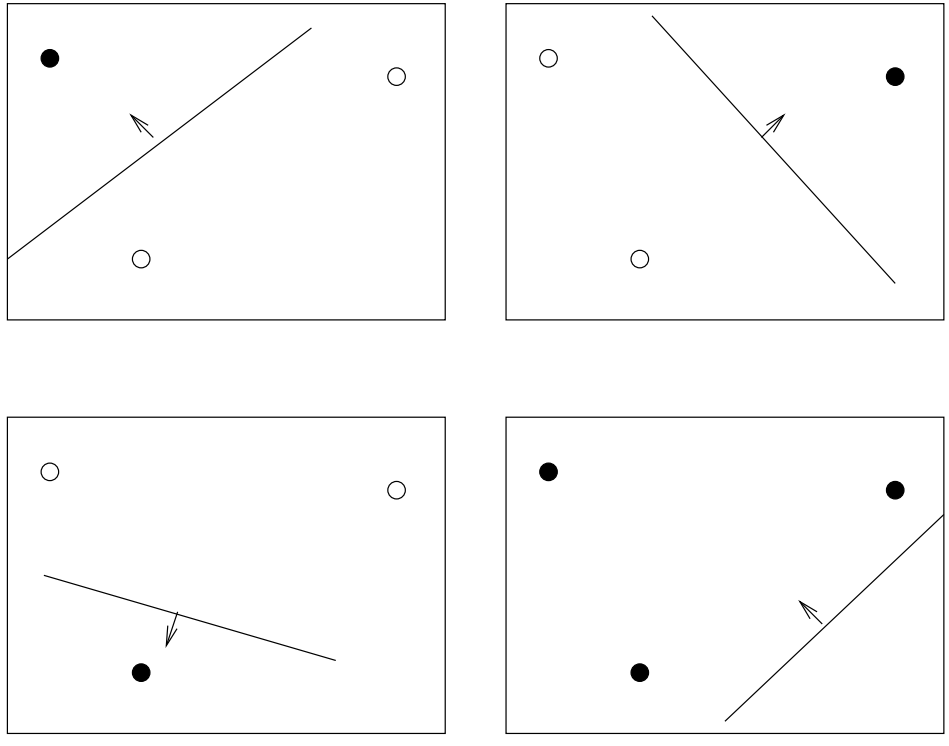
The VC-Dimension

- The VC-dimension (h) measures capacity (complexity in some sense!) of a classifier.
- ***Informally:*** If the classifier can correctly classify a set of atmost h points, for whatever be the labellings for the patterns, then we say that the VC-dimension for the classifier is h.
- ***Eg:*** VC-dimension of Perceptron for a 2D problem is 3.
- ***Eg:*** VC-dimension of 1-NNC is ∞.

The VC-Dimension (formally)

- $f(X;\alpha)$ is the learning machine.
- If α is allowed to vary we get a family (set) of functions $\{f(X;\alpha_1),\ldots,f(X;\alpha_m)\}$. Obviously m could be ∞, and even this set could be uncountable.
- We use $\{f(\alpha)\}$ to denote this set of functions.
- ***Shattering:*** For a given set of points $\{X_1,\ldots,X_n\}$ and for each assignment of labels in {+1, -1}, if there is a function in $\{f(\alpha)\}$ that correctly classifies the set, then we say $\{f(\alpha)\}$ shatters the set $\{X_1,\ldots,X_n\}$.

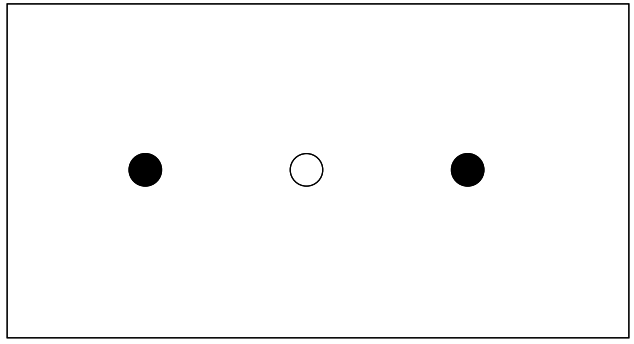
The VC-Dimension (formally)



- Perceptron in 2D can shatter 3 points.
- There are in fact $2^3 = 8$ ways of labellings the three points.

Perceptron in 2D

- It only means that there exists a set of 3 points and for every labeling there is a line which separates the classes correctly.
- It doesnot mean that “for every set of 3 points ...”
- In fact, for three points in a straight line, one cannot find a Perceptron for every labeling.



VC-Dimension

Myth: A classifier with few parameters will have less VC-dimension.

- It is shown (see the paper) that for $f(x) = Sign(\sin(\alpha x))$ there exists an infinite set of points which can be shattered by it.

Myth: Large VC-dimension $\Rightarrow$ Poor classifier.

Reality: For large VC-dimensions, the bound is invalid or useless.

- Eg: 1-NNC.

Structural Risk Minimization

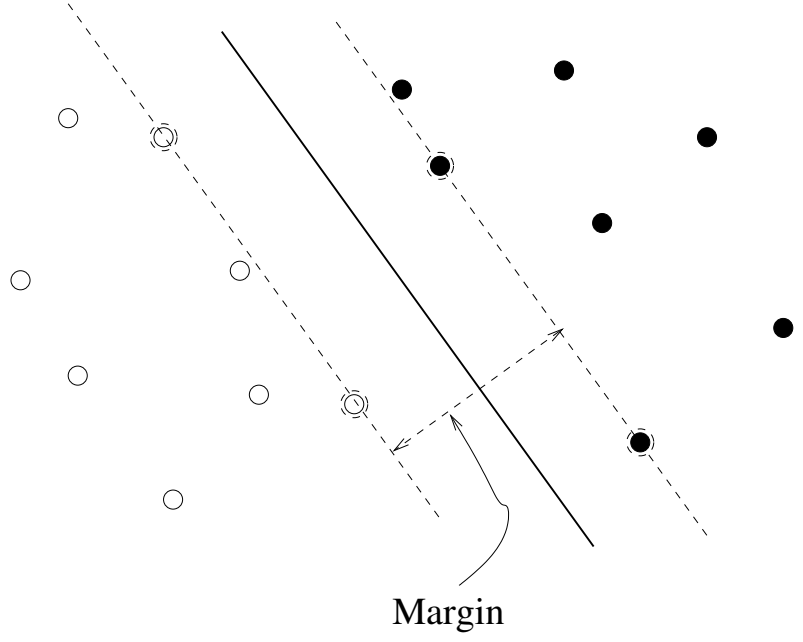
$$R(\alpha) \leq R_{emp}(\alpha) + \underbrace{\phi\left(\frac{h}{n}, \frac{log(\eta)}{n}\right)}_{\text{VC Confidence}}$$

- VC-confidence depends upon the chosen family of functions.
- Whereas the actual and the empirical risks depends upon the particular function.
- In most of the cases we cannot find the actual risk i.e., the LHS of the bound.

Structural Risk Minimization

- But once we know the h we can find the R.H.S.
  - Among various families choose the best ones based on $R_{emp}(\alpha)$.
  - Among these best ones choose that for which the RHS of the bound is low.

Linear SVM



- Assuming linearly separable data, this tries to find the best separating hyperplane.

Linear SVM

- We like to draw two hyperplanes such that

$$\begin{aligned} W \cdot X_i + b &\geq 1 \quad \textit{if } y_i = +1 \\ W \cdot X_i + b &\leq -1 \quad \textit{if } y_i = -1 \end{aligned}$$

$$\begin{aligned} y_i(W \cdot X_i + b) &\geq 1 \quad \textit{for all } i, \qquad \text{OR} \\ 1 - y_i(W \cdot X_i + b) &\leq 0 \quad ----- \quad -----(1) \end{aligned}$$

- The two parallel hyperplanes are

$$\begin{aligned} H_1 &: W \cdot X + b = 1 \\ H_2 &: W \cdot X + b = -1 \end{aligned}$$

Linear SVM

- Distance between origin and H_1 is $\frac{b-1}{||W||}$
- Distance between origin and H_2 is $\frac{b+1}{||W||}$
- Then, the margin $= \frac{b-1}{||W||} - \frac{b+1}{||W||}$
  $= \frac{2}{||W||}$
- Then the problem is :

  Minimize $\frac{1}{2}||W||^2$ (Objective function)

  Subject to constraints: $1 - y_i(W \cdot X_i + b) \leq 0$, for all i
- Note that the objective is *convex* and the constraints are *linear*
- Lagrangian method can be applied.

Constrained Optimization Problem

- Minimize $f(v)$
  Subject to the constraints $g_j(v) \leq 0, \;\; 1 \leq j \leq n$.
- Lagrangian,

$$\mathcal{L} = f(v) + \sum_{j=1}^{n} \alpha_j \; g_j(v)$$

  where v is called *primary* variables and α_j are the Lagrangian multipliers which are also called *dual* variables.
- $\mathcal{L}$ has to be minimized with respect to primal varibles and maximized with respect to dual variables.

Constrained Optimization Problem

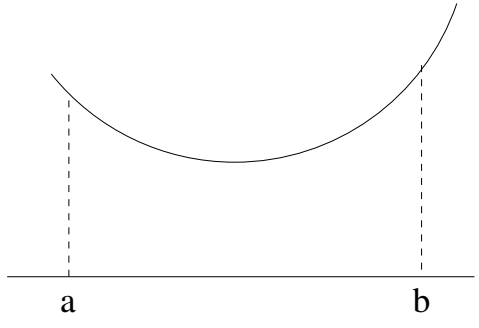
- The K.K.T (Karush-Kuhn-Tucker) conditions "necessary" at optimal v are:
  1. $\nabla_v \mathcal{L} = 0$
  2. $\alpha_j \geq 0, \quad$ for all $j = 1, \ldots, n$
  3. $\alpha_j g_j(v) = 0, \quad$ for all $j = 1, \ldots, n$
- If $f(v)$ is convex and $g_j(v)$ is linear for all j, then it turns out that K.K.T conditions are "necessary and sufficient" for the optimal v.

Convex Function

- A real valued function f defined in (a, b) is said to be convex if

$$f\left(\lambda x + (1-\lambda)y\right) \leq \lambda f(x) + (1-\lambda) f(y)$$

for $a < x, y < b$, and $0 < \lambda < 1$



- This definition can be extended to functions in higher dimensional spaces.

Lagrangian

- Minimize $\frac{1}{2}||W||^2$ (Objective function)
- Subject to constraints: $1 - y_i(W \cdot X_i + b) \leq 0$, for all i
- Lagrangian,

$$\mathcal{L}(W, b, \alpha) = \frac{1}{2}||W||^2 + \sum_i \alpha_i \left[1 - y_i(W \cdot X_i + b)\right]$$

- Here,

$$\alpha = \begin{pmatrix} \alpha_1 \\ . \\ . \\ . \\ \alpha_n \end{pmatrix}$$

K.K.T. Conditions

- $\nabla_W \mathcal{L} \;=\; W - \sum_i \alpha_i y_i X_i \;=\; 0$

  $\Rightarrow \qquad W \;=\; \sum_i \alpha_i y_i X_i \; - - - - - - - - - - \rightarrow (1)$
- $\frac{\partial \mathcal{L}}{\partial b} \;=\; -\sum \alpha_i y_i \;=\; 0$

  $\Rightarrow \qquad \sum \alpha_i y_i \;=\; 0 \; - - - - - - - - - - - - - \rightarrow (2)$
- $\alpha_i \;\geq\; 0,$ for $i = 1$ to $n \quad - - - - - - - - - - \rightarrow (3)$
- $\alpha_i \left[1 - y_i(W \cdot X_i + b)\right] \;=\; 0 \quad - - - - - - - - - \rightarrow (4)$

  for $i = 1$ to n.

- Solve these equations $(1), (2), (3), (4)$ to get W and b.
- While it is possible to do this, it is tedious !

Wolfe Dual Formulation

- Other easy and advantageous ways to solve the optimization problem does exist which can be easily extended to non-linear SVMs.
- This is to get $\mathcal{L}$ where we eliminate W and b and which has $\alpha_1, \ldots, \alpha_n$.
- We know, $\mathcal{L}$ has to be maximized w.r.t. the dual variables $\alpha_1, \ldots, \alpha_n$.

Wolfe Dual Formulation

- The Lagrangian $\mathcal{L}$ is :

$$= \frac{1}{2}||W||^2 + \sum_i \alpha_i - \sum_i \alpha_i y_i X_i \cdot W - \sum_i \alpha_i y_i b$$

$$= \frac{1}{2}||W||^2 + \sum_i \alpha_i - W \cdot \underbrace{\left[\sum_i \alpha_i y_i X_i\right]}_{W} - b \underbrace{\left[\sum_i \alpha_i y_i\right]}_{0}$$

$$= -\frac{1}{2}||W||^2 + \sum_i \alpha_i$$

$$= \sum_i \alpha_i - \frac{1}{2}\sum_i \sum_j \alpha_i \alpha_j y_i y_j X_i \cdot X_j$$

Wolfe Dual Formulation

- Maximize $\mathcal{L}$ w.r.t α

$$\mathcal{L} = \sum_i \alpha_i - \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y_i y_j X_i \cdot X_j$$

  such that $\sum \alpha_i y_i = 0$, and $\alpha_i \geq 0$ for all i.

- We need to find the Lagrangian multipliers α_i $(1 \leq i \leq n)$ only.
- Primal variables W and b are eliminated.
- There exists various numeric iterative methods to solve this constrained convex quadratic optimization problem.
- *Sequential minimal optimization (SMO)* is one such technique which is a simple and relatively fast method.

The Optimization Problem

- $\alpha = (\alpha_1, \ldots, \alpha_n)^t$, let $\mathbf{1} = (1, \ldots, 1)^t$ then

$$\mathcal{L} = -\frac{1}{2}\alpha^t \hat{K} \alpha + \mathbf{1}^t \alpha$$

  where $\hat{K}$ is a $n \times n$ matrix with its $(i,j)^{th}$ entry being $(y_i y_j X_i \cdot X_j)$

- If $\alpha_i > 0$ (from (4)) $\Rightarrow$ X_i is on a hyperplane, i.e., X_i is a `support vector`.
  Note: X_j lies on the hyperplane $\not\Rightarrow$ $\alpha_i > 0$

- Similarly, X_i doesnot lie on hyperplane $\Rightarrow \alpha_i = 0$
  That is, for interior points $\alpha_i = 0$.

The Solution

- Once α is known, We can find $W = \sum_i \alpha_i y_i X_i$
- The classifier is

$$f(X) = Sign\{W \cdot X + b\} = Sign\left\{\sum_i \alpha_i y_i X_i \cdot X + b\right\}$$

- b can be found from (4)
  - For any $\alpha_j \neq 0$, we have $y_j(W \cdot X_j + b) = 1$
  - Multiplying with y_j on both sides we get, $W \cdot X_j + b = y_j$
  - So, $b = y_j - W \cdot X_j = y_j - \sum_i \alpha_i y_i X_i \cdot X_j$

Some observations

- In the dual problem formulation and in its solution we have only dot products between some of the training patterns.
- Once we have the matrix $\hat{K}$, the problem and its solution are independent of the dimensionality d.

Non-linear SVM

- We know that every non-linear function in X-space (input space) can be seen as a linear function in an appropriate Y-space (feature space).
- Let the mapping be $Y = \phi(X)$.
- Once the $\phi(\cdot)$ is defined, one has to replace in the problem as well as in the solution, for certain products, as explained below.
- Whenever we see $X_i \cdot X_j$, replace this by $\phi(X_i) \cdot \phi(X_j)$.
- While it is possible to explicitly define the $\phi(\cdot)$ and generate the training set in the Y-space, and then obtain the solution...
- it is tedious and amazingly unnecessary also.

Kernel Function

- For certain mappings, $\phi(X_i) \cdot \phi(X_j) = \kappa(X_i, X_j)$. That is, dot product in the Y-space can be obtained as a function in the X-space itself. There is no need to explicitly generate the patterns in the Y-space.
- Eg: Consider a two dimensional problem with $X = (x_1, x_2)^t$. Let $\phi(X) = (x_1^2, x_2^2, \sqrt{2}\, x_1 x_2)^t$. Then $\phi(X_i) \cdot \phi(X_j) = \kappa(X_i, X_j) = (X_i \cdot X_j)^2$.
- This *kernel trick* is one of the reasons for the success of SVMs.

Kernel Function

- $\kappa(X_i, X_j)$ is called the kernel function.
- We say $\kappa(\cdot, \cdot)$ is a valid kernel iff there exists a $\phi(\cdot)$ such that $\kappa(X_i, X_j) = \phi(X_i) \cdot \phi(X_j)$ for all X_i and X_j.
- Mercer's Theorem gives the necessary conditions for a kernel to be valid.
- While Mercer's theorem is a mathematically involved one, some of the properties of kernels can be used to verify whether a kernel is valid.

Kernel Function

- The following three properties are satisfied by any Kernel.
  1. $\kappa(X_i, X_j) = \kappa(X_j, X_i), \quad \forall X_i, X_j$
  2. $\kappa(X, X) \geq 0, \forall X$
  3. $[\kappa(X_i, X_j)]^2 \leq \kappa(X_i, X_i)\kappa(X_j, X_j), \quad \forall X_i, X_j$

Some ways to generate new Kernels

- κ_1, κ_2 are kernels $\Rightarrow \alpha_1\kappa_1 + \alpha_2\kappa_2$ is a kernel, $\alpha_1 \geq 0,\ \alpha_2 \geq 0$
- κ_1, κ_2 are kernels $\Rightarrow \kappa(X_i, X_j) = \kappa_1(X_i, X_j)\kappa_2(X_i, X_j)$ is also a valid kernel. That is, $\kappa = \kappa_1\kappa_2$ is a valid kernel.
- For any symmetric, positive semi-definite $d \times d$ matrix B, $\kappa(X_i, X_j) = X_i^t B X_j$ is a valid kernel.
- Let $p(x)$ be a polynomial with positive coefficients. Let $\kappa(X_i, X_j)$ is a kernel then $p(\kappa(X_i, X_j))$ is a valid kernel.
- $\kappa(X_i, X_j)$ is a kernel $\Rightarrow$ exp($\kappa(X_i, X_j)$) (called *exponential kernel*) is also valid kernel.
- $\kappa(X_i, X_j) = \exp(-||X_i - X_j||^2/\sigma^2)$ (called *Gaussian kernel*)is a kernel.

Soft Margin Formulation

- Until now, we assumed that the data is linearly (or non-linearly) separable.
- The SVM derived is sensitive to noise.
- Soft Margin formulation allows violation of the constraints to some extent. That is, we allow some of the boundary patterns to be misclassified.

Soft Margin Formulation

$$W \cdot X_i + b \geq 1 - \xi_i \quad \textit{if } y_i = +1$$
$$W \cdot X_i + b \leq -1 + \xi_i \quad \textit{if } y_i = -1$$

$$y_i(W \cdot X_i + b) \geq 1 - \xi_i \quad \textit{for all } i, \quad \text{OR}$$
$$1 - \xi_i - y_i(W \cdot X_i + b) \leq 0 \quad --(1)$$

- ξ_i are called *slack variables*, and $\xi_i \geq 0$ for all i $\quad ---------(2)$
- Now, the objective to minimize: $\frac{1}{2}||W||^2 + C\sum \xi_i$
- C is called *penalty* parameter, and $C \geq 0$.

Soft Margin Formulation

- Lagrangian

$$\begin{aligned}\mathcal{L} = \quad & \tfrac{1}{2}||W||^2 + C\sum \xi_i \\ & + \sum \alpha_i \left[1 - \xi_i - y_i(W \cdot X_i + b)\right] \\ & - \sum \beta_i \xi_i\end{aligned}$$

Soft Margin: KKT Conditions

$$
\begin{aligned}
1) \quad \nabla_W \mathcal{L} &= W - \sum \alpha_i y_i X_i = 0 \\
&\Rightarrow \; W = \sum \alpha_i y_i X_i \qquad \qquad ---(1)
\end{aligned}
$$

$$
\begin{aligned}
2) \quad \frac{\partial \mathcal{L}}{\partial b} &= -\sum \alpha_i y_i = 0 \\
&\Rightarrow \; \sum \alpha_i y_i = 0 \qquad \qquad ---(2)
\end{aligned}
$$

$$
\begin{aligned}
3) \quad \frac{\partial \mathcal{L}}{\partial \xi_i} &= C - \alpha_i - \beta_i = 0 \\
&\Rightarrow \; \alpha_i + \beta_i = 0 \qquad \qquad ---(3)
\end{aligned}
$$

Soft Margin: KKT Conditions

$$4) \quad \alpha_i \geq 0 \qquad ---(4)$$

$$5) \quad \beta_i \geq 0 \qquad ---(5)$$

$$6) \quad \alpha_i(1 - \xi_i - y_i(W \cdot X_i + b)) = 0 \qquad ---(6)$$

$$7) \quad \xi_i \beta_i = 0 \qquad ---(7)$$

Wolfe Dual Formulation

- Maximize $\mathcal{L}$ w.r.t α

$$\mathcal{L} = \sum_i \alpha_i - \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y_i y_j X_i \cdot X_j$$

such that $\sum \alpha_i y_i = 0$, and $0 \leq \alpha_i \leq C$ for all i.

- $C = \infty \Rightarrow$ Hard Margin
- $C = 0 \Rightarrow$ Very Very Soft Margin

Training Methods

- Any convex quadratic programming technique can be applied.
- But with larger training sets, most of the standard techniques can become very slow and space occupying. For example, many techniques needs to store the kernel matrix whose size is n^2 where n is the number of training patterns.
- These considerations have driven the design of specific algorithms for SVMs that can exploit the sparseness of the solution, the convexity of the optimization problem, and the implicit mapping into feature space.
- One such a simple and fast method is Sequential Minimal Optimization (SMO).

Next Class ...

- SMO algorithm.