Chapter 7: Support Vector Machine

- Linear SVM
- Kernelized SVM
- Sequential Minimal Optimization

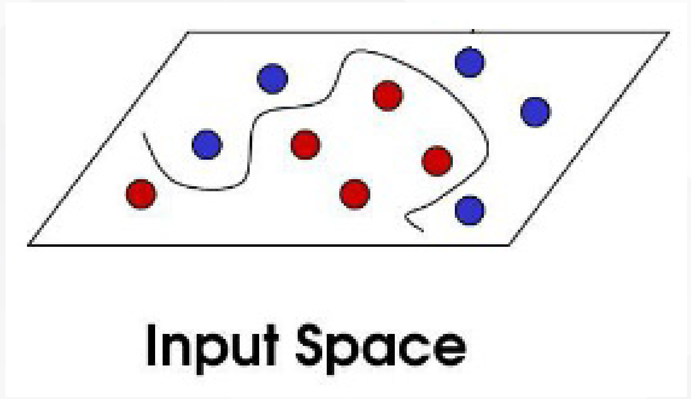
Intuitions of Kernel I: Non-Linear Support Vector Machine

- The primal optimization problem:

$$\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|_2^2 + C\sum_{i=1}^{N}\xi_i$$

$$s.t.\ \ y^i\left(\mathbf{w}^T\mathbf{x}^i + b\right) \geq 1-\xi_i,\ \ i=1,\dots,N,$$

$$\xi_i \geq 0,\quad i=1,\dots,N.$$



- Nonlinear Separable: if we can use a hypersurface to correctly separate the positive and negative examples.
- But the nonlinear problem is usually not easy to solve. Therefore, a typical method is to transform the nonlinear problem to a linear problem. Then we can use linear techniques to solve it.

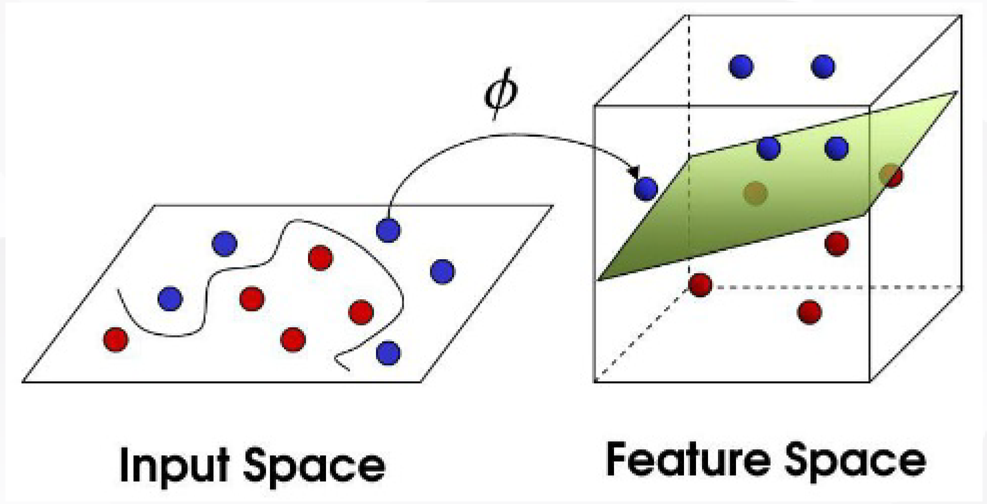
Intuitions of Kernel I: Non-Linear Support Vector Machine

- Transformation $\boldsymbol{z} = \phi(\boldsymbol{x})$
- The primal optimization problem:

$$\min_{w,b} \frac{1}{2}\|\boldsymbol{w}\|_2^2 + C\sum_{i=1}^{N}\xi_i$$

$$s.t.\ \ y^i\left(\boldsymbol{w}^T\boldsymbol{z}^i + b\right) \geq 1 - \xi_i,\ \ i = 1,\ldots,N,$$

$$\xi_i \geq 0,\quad i = 1,\ldots,N.$$



Intuitions of Kernel II: Dual Form of Support Vector Machine

- The dual optimization problem:

$$\max_{\boldsymbol{\alpha}} \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} y^i y^j \alpha_i \alpha_j \left(\boldsymbol{x}^i\right)^T \boldsymbol{x}^j$$

$$s.t.\ 0 \le \alpha_i \le C, i = 1, .., N,$$

$$\sum_{i=1}^{N} \alpha_i y^i = 0.$$

- Hyperplane:

$$\boldsymbol{w}^* = \sum_{i=1}^{N} \alpha_i^* y^i \boldsymbol{x}^i,\ b^* = y^j - \sum_{i=1}^{N} \alpha_i^* y^i \left(\boldsymbol{x}^i\right)^T \boldsymbol{x}^j.$$

- Largely dependent on a distance $< \boldsymbol{x}^i, \boldsymbol{x}^j >$.

▯

Intuitions of Kernel II: Dual Form of Regression

- Regularized sum of square error:

$$L(\mathbf{w})=\sum_{i=1}^{N}(\mathbf{w}^T\mathbf{x}^i-y^i)^2+\lambda\|\mathbf{w}\|^2$$

- Setting the derivatives of L with respect to $\mathbf{w}$ to zero, yields:

$$\mathbf{w}=-\frac{1}{\lambda}\sum_{i=1}^{N}(\mathbf{w}^T\mathbf{x}^i-y^i)\mathbf{x}^i$$

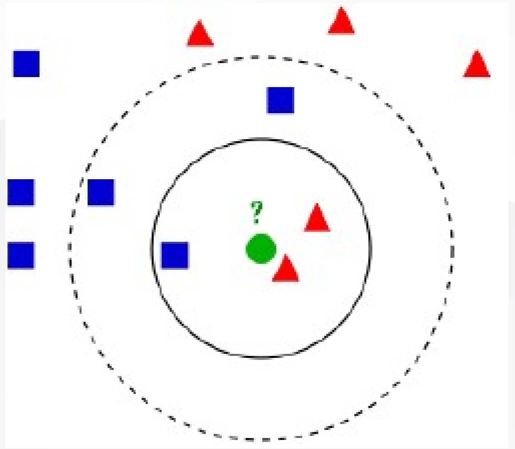
- Let $\alpha_i=\mathbf{w}^T\mathbf{x}^i-y^i$. Then we have $\mathbf{w}=-\frac{1}{\lambda}\sum_{i=1}^{N}\alpha_i\mathbf{x}^i$.
- Plug it back into $L(\mathbf{w})$,we get:

$$L(\mathbf{w})=\sum_{i=1}^{N}\alpha_i^2+\frac{1}{\lambda}\sum_{i,j=1}^{N}\alpha_i\alpha_j(\mathbf{x}^i)^T\mathbf{x}^j$$

- Largely dependent on a distance $<\mathbf{x}^i,\mathbf{x}^j>$.

Intuitions of Kernel III: Nonparametric Methods

- Two kinds of methods for prediction:
  - **Parametric models**: During learning phase, we either get a maximum likelihood estimate of $\boldsymbol{w}$ or a posterior distribution of $\boldsymbol{w}$. Training data is then discarded. We conduct our prediction based only on vector $\boldsymbol{w}$.
  - **Nonparametric (Memory based) models**: Training data points are used in prediction phase. Therefore, it largely depends on a defined distance (similarity), e.g. $< \boldsymbol{x}, \boldsymbol{x}^j >$.
- Nearest neighbor classification: An object is classified by a majority vote of its neighbors, with the object being assigned to the class most common among its k nearest neighbors.



Kernel

Definition: Kernel

Given the input space $\boldsymbol{X}$ and the feature space $\boldsymbol{H}$ (Hilbert Space), if there exists a mapping ϕ from $\boldsymbol{X}$ to $\boldsymbol{H}$, such that for any $\boldsymbol{x}, \boldsymbol{z} \in \boldsymbol{X}$, the function $K(\boldsymbol{x}, \boldsymbol{z})$ satisfies $K(\boldsymbol{x}, \boldsymbol{z}) = <\phi(\boldsymbol{x}), \phi(\boldsymbol{z})>$, then $K(\boldsymbol{x}, \boldsymbol{z})$ is called a kernel where ϕ is the mapping function.

- For a given kernel $K(\boldsymbol{x}, \boldsymbol{z})$, the feature space $\boldsymbol{H}$ and mapping function ϕ is usually not unique.
- Example: $\boldsymbol{X} = R^2, K(\boldsymbol{x}, \boldsymbol{z}) = <\boldsymbol{x}, \boldsymbol{z}>^2$
  - Denote $\boldsymbol{x} = (x_1, x_2)$ and $\boldsymbol{z} = (z_1, z_2)$, we have:

$$<\boldsymbol{x}, \boldsymbol{z}>^2 = (x_1 z_1 + x_2 z_2)^2 = (x_1 z_1)^2 + 2x_1 z_1 x_2 z_2 + (x_2 z_2)^2$$

① $\boldsymbol{H} = R^3, \phi(\boldsymbol{x}) = (x_1^2, \sqrt{2}x_1 x_2, x_2^2)^T$

② $\boldsymbol{H} = R^3, \phi(\boldsymbol{x}) = \frac{1}{\sqrt{2}}((x_1 - x_2)^2, 2x_1 x_2, (x_1 + x_2)^2)^T$

③ $\boldsymbol{H} = R^4, \phi(\boldsymbol{x}) = (x_1^2, x_1 x_2, x_1 x_2, x_2^2)^T$

Kernel Track and Its Application in SVM

- Kernel Trick: Only the kernel $K(\boldsymbol{x},\boldsymbol{z})$ is used in learning and prediction. We do not need to define the mapping ϕ explicitly.
- The dual optimization problem of SVM:

$$\max_{\boldsymbol{\alpha}} \sum_{i=1}^{N} \alpha_i - \frac{1}{2}\sum_{i,j=1}^{N} y^i y^j \alpha_i \alpha_j K\left(\boldsymbol{x}^i, \boldsymbol{x}^j\right)$$

$$s.t. \sum_{i=1}^{N} \alpha_i y^i = 0,$$

$$0 \le \alpha_i \le C, i = 1,..,N.$$

- Hyperplane and Optimal Marginal Classifier

$$\boldsymbol{w}^* = \sum_{i=1}^{N} \alpha_i^* y^i \phi(\boldsymbol{x}^i),\ b^* = y^j - \sum_{i=1}^{N} \alpha_i^* y^i K\left(\boldsymbol{x}^i, \boldsymbol{x}^j\right).$$

$$f_{\boldsymbol{w}^*,b^*} = (\boldsymbol{w}^*)^T \phi(\boldsymbol{x}) + b^* = \sum_{i=1}^{N} \alpha_i^* y^i K(\boldsymbol{x}^i, \boldsymbol{x}) + b^*.$$

Kernelized SVM

- It can be viewed as we are mapping the original input space $\boldsymbol{X}$ to a feature space $\boldsymbol{H}$ by the mapping function ϕ , and learning a linear SVM in the new feature space.
- When the mapping function is nonlinear, we can obtain a nonlinear classifier.
- Given the kernel $K(\mathbf{x}, \mathbf{z})$, we can still use linear SVM methods to find the solution of the new nonlinear problem.
- Please note that the learning process is directly conducted in the feature space, we do not need to explicitly define the feature space $\boldsymbol{H}$ and the mapping function ϕ . This is one advantage of kernel, which can directly use the linear classification methods to solve the nonlinear problem.

Valid Kernel

- Question: How can we directly define a Kernel, not through ϕ ?
- Question: Given a kernel $K(x, z)$, how can we tell if it’ s a valid kernel, i.e., can we tell if there is some feature mapping ϕ so that $K(\mathbf{x},\mathbf{z}) = \phi(\mathbf{x})^T \phi(\mathbf{x})$ for all x, z.

Mercer’ s Theorem

Let K: $R^n \times R^n \rightarrow R$ be given. Then for $\boldsymbol{K}$ to be a valid (Mercer) kernel, it is necessary and sufficient that for any $\{\mathbf{x}^1, \cdots, \mathbf{x}^m\}$, ($m < \infty$), the corresponding kernel matrix is symmetric positive semi-definite.

- Kernel (Gram) Matrix: consider some infinite set of m points $\{\mathbf{x}^1, \cdots, \mathbf{x}^m\}$, let a square, m by m matrix be defined so that its $(i\ , j)$-th entry is given by $K_{ij} = K(\mathbf{x}^i, \mathbf{x}^j)$.

$$\mathbb{K} = \begin{pmatrix} K_{11} & K_{12} & \cdots & K_{1m} \\ K_{21} & K_{22} & \cdots & K_{2m} \\ \cdots & \cdots & \cdots & \cdots \\ K_{m1} & K_{m2} & \cdots & K_{m2} \end{pmatrix}.$$

Reproducing Kernel Hilbert Space (RKHS)

- A reproducing kernel Hilbert space (RKHS) is a Hilbert space associated with a kernel that reproduces every function in the space.
- Reproducing Kernel: A kernel $K : X \times X \to R$ is called reproducing if it satises the following two properties:
  - •For any $\boldsymbol{x}_0 \in X$, $K(\boldsymbol{x}, \boldsymbol{x}_0)$ is a function of x in the space H.
  - •For any $\boldsymbol{x} \in X$ and $f \in H$, we have $f(\boldsymbol{x}) = f(\cdot) \bullet K(\cdot, \boldsymbol{x})$.(Reproducing Property)
- The mapping function can be naturally be dened as $\phi(\boldsymbol{x}) = K(\cdot, \boldsymbol{x})$
- We can see that the following equation holds:

$$\phi(\boldsymbol{x})^T \phi(\boldsymbol{x}) = K(\cdot, \boldsymbol{x}) \bullet K(\cdot, \boldsymbol{z}) = K(\boldsymbol{x}, \boldsymbol{z})$$

- Note that for each $f(\cdot) = \sum_{i=1}^{m} \alpha_i K(\cdot, \boldsymbol{x}^i)$, we $f(\cdot) \bullet K(\cdot, \boldsymbol{x}) = \sum_{i=1}^{m} \alpha_i K(\boldsymbol{x}, \boldsymbol{x}^i) = f(\boldsymbol{x})$. It defines an inner product in the space H.

Some Common Kernel Functions

- Polynomial Kernel:

$$K(\boldsymbol{x},\boldsymbol{z}) = (\boldsymbol{x}^T\boldsymbol{z}+1)^p$$

- p is the degree
- If p =2 and $\boldsymbol{x}=(x_1,x_2)$

$$K(\boldsymbol{x},\boldsymbol{z}) = (x_1z_1 + x_2z_2 + 1)^2 = 1 + 2x_1z_1 + 2x_2z_2 + 2x_1z_1x_2z_2 + x_1^2z_1^2 + x_2^2z_2^2$$

- Mapping Function: $\phi(\boldsymbol{x}) = (1, \sqrt{2}x_1, \sqrt{2}x_2, \sqrt{2}x_1x_2, x_1^2, x_2^2)^T$
- The corresponding SVM is a polynomial classifier.

$$f(\boldsymbol{x}) = \operatorname{sgn}\Big(\sum_{i=1}^{n} \alpha_i^* y^i ((\boldsymbol{x}^i)^T \boldsymbol{x} + 1)^p + b^*\Big).$$

Some Common Kernel Functions

- Gaussian Kernel

$$K(\boldsymbol{x},\boldsymbol{z}) = \exp\{-\frac{\|\boldsymbol{x}-\boldsymbol{z}\|^2}{2\sigma^2}\}.$$

- Sometimes it is also called radial basis function (RBF) kernel. Can be generalized to the following form:

$$K(\boldsymbol{x},\boldsymbol{z}) = \exp\{-\frac{Dist(\boldsymbol{x},\boldsymbol{z})}{2\sigma^2}\}.$$

- The corresponding SVM is a gaussian radial basis function.

$$f(\boldsymbol{x}) = \operatorname{sgn}(\sum_{i=1}^{n}\alpha_i^* y^i(\exp\{-\frac{\|\boldsymbol{x}^i-\boldsymbol{x}\|^2}{2\sigma^2}\}+b^*).$$

Theoretical Analysis

Representer Theorem

Let $\boldsymbol{X}$ be a nonempty set and $\boldsymbol{K}$ a positive-definite real-valued kernel on $\boldsymbol{X} \times \boldsymbol{X}$ with corresponding reproducing kernel Hilbert space $H_{\boldsymbol{K}}$. Given training samples $\{(\boldsymbol{x}^1, y^1), \cdots, (\boldsymbol{x}^N, y^N)\}$, a strictly monotonically increasing real-valued function $g:[0,\propto) \to R$, and an arbitrary empirical risk function $\hat{E}$, then for $f^* \in H_K$ satisfying:

$$f^* = \arg\min_{f \in H_K} \{\hat{E} + g(\| f \|_K)\}.$$

f^* admits a representation of the form:

$$f^*(\cdot) = \sum_{i=1}^{N} \alpha_i K(\cdot, \boldsymbol{x}^i).$$

where $\alpha_i \in R$ for all $1 \le i \le N$.

SMO: Sequential Minimal Optimization

- We have mentioned that the primal SVM can be solved by traditional convex quadratic programming methods, which can guarantee the global optimal solution. However, these algorithm usually become slowly especially when the training data are large.
- SMO, proposed by John Platt in 1998, gives an efficient way of solving the dual problem arising from the derivation of the SVM.
- The dual optimization problem:

$$\max_{\boldsymbol{\alpha}} \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} y^i y^j \alpha_i \alpha_j \left(\boldsymbol{x}^i\right)^T \boldsymbol{x}^j$$

$$s.t. \ \sum_{i=1}^{N} \alpha_i y^i = 0,$$

$$0 \le \alpha_i \le C, i = 1,.., N.$$

Motivation of SMO: Coordinate Ascent

- Consider the following unconstrained optimization problem:

$$\max_{\boldsymbol{\alpha}} W(\alpha_1, \cdots, \alpha_N)$$

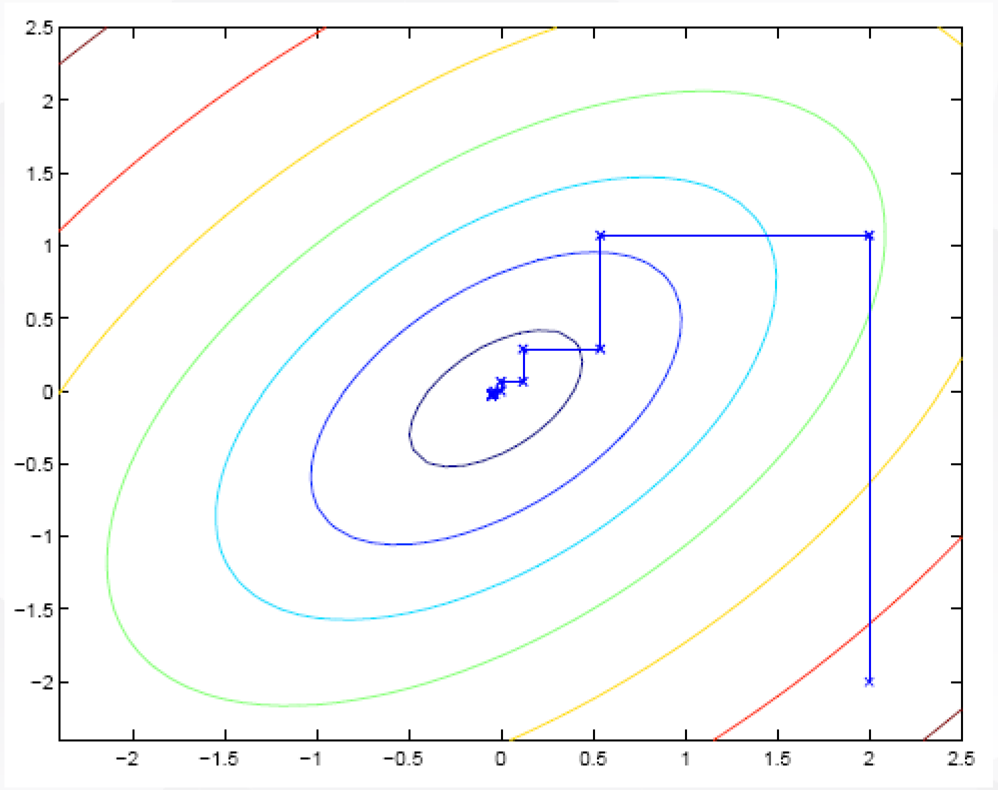
- Coordinate Ascent Optimization Algorithm:

```
Loop until convergence : {
    For $i = 1, \cdots, m$ :{
        $\alpha_i = \max_{\hat{\alpha}_i} W(\alpha_1, \cdots, \alpha_{i-1}, \hat{\alpha}_i, \alpha_{i+1}, \cdots, \alpha_N).$
    }
}
```

- In the innermost loop of this algorithm, we will hold all the variables except for some α_i fixed, and re-optimize W with respect to just the parameter α_i.

Illustration of Coordinate Ascent

- The ellipses in the figure are the contours of a quadratic function that we want to optimize.
- Coordinate ascent was initialized at (2,-2), and also plotted in the figure is the path that it took on its way to the global maximum.
- Notice that on each step, coordinate ascent takes a step that is parallel to one of the axes, since only one variable is being optimized at a time.



Coordinate Ascent for SVM?

- The dual optimization problem:

$$\max_{\boldsymbol{\alpha}} \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} y^i y^j \alpha_i \alpha_j \left(\boldsymbol{x}^i\right)^T \boldsymbol{x}^j$$

$$s.t. \sum_{i=1}^{N} \alpha_i y^i = 0,$$

$$0 \le \alpha_i \le C, i = 1,.., N.$$

- Suppose we want to hold $\alpha_1, \cdots, \alpha_N$ fixed, and take a coordinate ascent step and reoptimize the objective with respect to α_1 . Can we make any progress?
- The answer is no, because the constraint ensures that:

$$\alpha_1 = -y^1 \sum_{i=2}^{N} \alpha_i y^i$$

SMO: Update a Pair of

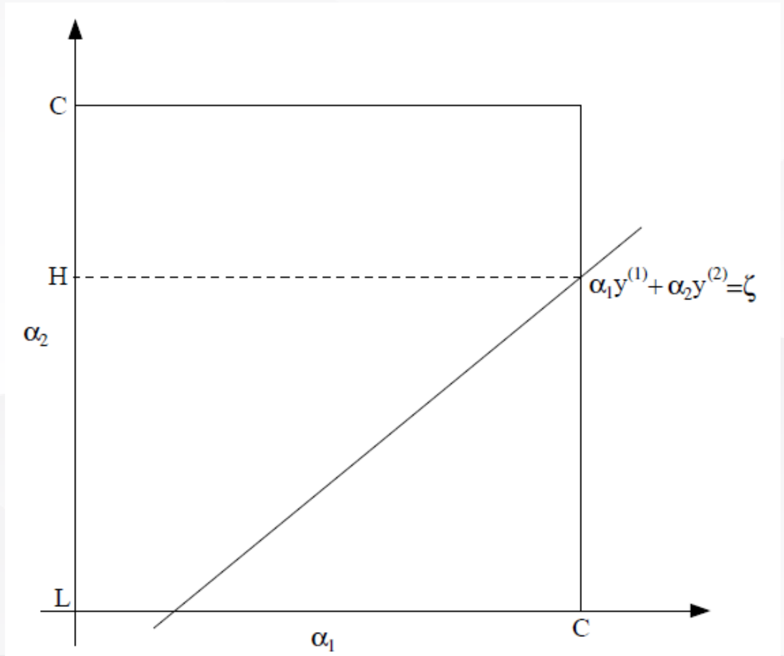
- If we want to update some subject of the α_is, we must update at least two of them simultaneously in order to keep satisfying the constraint.
- This is exactly the motivation of SMO, which simply does the following:

```
Repeat until convergence : {
(1) Select some pair to $\alpha_i$ and $\alpha_j$ update next
(using a heuristic that tries to pick the two that
will allow us to make the biggest progress towards
the global maximum):
(2) Re-optimize $W(\alpha)$ with respect to $\alpha_i$ and $\alpha_j$;
while holding all the other $\alpha_k s$ $(k = i, j)$ fixed:
}
```

- The key reason that SMO is an ecient algorithm is that the update to α_i, α_j can be computed very eciently.

SMO: Optimizing with respect to Two Variables

- Lets say we currently have some setting of the α_is that satisfy the constraints, and suppose weve decided to hold $\alpha_3, \cdots, \alpha_N$ fixed, and want to re-optimize $W(\alpha_1, \cdots, \alpha_N)$ with respect to α_1 and α_2 .
- From the constraints, we require: $\alpha_1 y^1 + \alpha_2 y^2 = -\sum_{i=3}^{N} \alpha_i y^i$
- Since the right hand side is xed (as we have fixed $\alpha_3, \cdots, \alpha_N$), we can just let it be denoted by some constant ζ : $\alpha_1 y^1 + \alpha_2 y^2 = \zeta$.
- We can thus picture the constraints on α_1 and α_2 as follows:



- α_1 and α_2 must lie within the box $[0, C] \times [0, C]$ on the line $\alpha_1 y^1 + \alpha_2 y^2 = \zeta$.
- $L \le \alpha_2 \le H$. In this example, $L = 0$; but more generally, there will be some lower-bound L and some upper-bound H on the permissible values for α_2 that will ensure that α_1, α_2 lie within the box $[0, C] \times [0, C]$.

SMO: Optimizing with respect to One Variable

- Write α_1 as a function of α_2 : $\alpha_1 = (\zeta - \alpha_2 y^2) y^1$
- The objective $W(\alpha)$ can be written as:

$$W(\alpha_1, \cdots, \alpha_N) = W((\zeta - \alpha_2 y^2) y^1, \alpha_2, \cdots, \alpha_N)$$

- Treating other $\alpha_i s$ as constants, this is just a quadratic function in α_2.
- If we ignore the constraint of $L \le \alpha_2 \le H$, then we can easily maximize this quadratic function by setting its derivative to zero and solving. Let $\alpha_2^{new,unclipped}$ denote the resulting value of 2.
- If we had instead wanted to maximize W with respect to α_2 but subject to the box constraint, then we can find the resulting value optimal simply by taking $\alpha_2^{new,unclipped}$ and clipping it to lie in the *[L,H]* interval, to get:

$$\alpha_2^{new} = \begin{cases} H & \text{if } \alpha_2^{new,unclipped} > H \\ \alpha_2^{new,unclipped} & \text{if } \mathrm{L} \le \alpha_2^{new,unclipped} \le H \\ L & \text{if } \alpha_2^{new,unclipped} < L \end{cases}$$

SMO: Optimal Solution of α_1 and α_2

- Having found the α_2^{new} , we can use $\alpha_1 = (\zeta - \alpha_2 y^2) y^1$ to go back and find the optimal value of α_1^{new} as:

$$\alpha_1^{new} = \alpha_1^{old} + y^1 y^2 (\alpha_2^{old} - \alpha_2^{new})$$

- Recall that the optimal value of α_2^{new} is:

$$\alpha_2^{new} = \begin{cases} H & \text{if} \quad \alpha_2^{new,unclipped} > H \\ \alpha_2^{new,unclipped} & \text{if} \quad \mathrm{L} \leq \alpha_2^{new,unclipped} \leq H \\ L & \text{if} \quad \alpha_2^{new,unclipped} < L \end{cases}$$

SMO: Variable Selection for the First One

- Main Idea: Select the training example which most violate **KKT** condition, and treat the corresponding α_i as the first variable.
- Check whether the *i*-th training example (x^i, y^i) satisfies the KKT condition:

$$\alpha_i = 0 \Leftrightarrow y^i g(\boldsymbol{x}^i) \geq 1$$

$$0 < \alpha_i < C \Leftrightarrow y^i g(\boldsymbol{x}^i) = 1$$

$$\alpha_i = C \Leftrightarrow y^i g(\boldsymbol{x}^i) \leq 1$$

where $g(\boldsymbol{x}^i) = \sum_{j=1}^{N} \alpha_j y^j K(\boldsymbol{x}^i, \boldsymbol{x}^j) + b$

- We first check all of the examples satisfying $0 < \alpha_i < C$, i.e. support vectors on the boundary. If all these examples satisfy the KKT condition, then we will check all the training examples.

SMO: Variable Selection for the Second One

- Assume that we have found the first variable α_1 , now we are finding α_2 . The criteria is to make α_2 change much.

Theorem

$$\alpha_2^{new,unclipped} = \alpha_2^{old} + \frac{y^2(E_1 - E_2)}{\eta}$$

where $\eta = K_{11} + K_{22} - 2K_{12} = \left\|\phi(\mathbf{x}^1) - \phi(\mathbf{x}^2)\right\|^2, E_i = g(\mathbf{x}^i) - y^i.$

- A simple method is to directly choose α_2 to make the corresponding $|E_1 - E_2|$ largest.
- Since α_1 is fixed, E_1 is also fixed. If E_1 is positive, we can choose the smallest E_i as E_2 ; Otherwise, if E_1 is negative, we can choose the largest E_i as E_2 .
- For efficiency, we can save all of the E_i in one table.

SMO: Bias *b* and D-Value E_i

- Each time we nish the optimization over two variables, we need to compute the new bias b.
- If $0 < \alpha_1^{new} < C$, from KKT condition: $\sum_{i=1}^{N} \alpha_i y^i K_{i1} + b = y^1$.
- Then: $b_1^{new} = y^1 - \sum_{i=3}^{N} \alpha_i y^i K_{i1} - \alpha_1^{new} y^1 K_{11} - \alpha_2^{new} y^2 K_{21}$
- By the definition of E_1 , we have: $E_1 = \sum_{i=3}^{N} \alpha_i y^i K_{i1} + \alpha_1^{old} y^1 K_{11} + \alpha_2^{old} y^2 K_{21} + b^{old} - y^1$.
- Combine the above two equations, we have:
$$b_1^{new} = -E_1 - y^1 K_{11}(\alpha_1^{new} - \alpha_1^{old}) - y^2 K_{21}(\alpha_2^{new} - \alpha_2^{old}) + b^{old}.$$
- Similarly, we have:
$$b_2^{new} = -E_2 - y^1 K_{12}(\alpha_1^{new} - \alpha_1^{old}) - y^2 K_{22}(\alpha_2^{new} - \alpha_2^{old}) + b^{old}.$$
- If both α_1^{new} and α_2^{new} satisfy $0 < \alpha_i^{new} < C,\ i = 1, 2,\ b_1^{new} = b_2^{new}$.
- If $\alpha_1^{new}, \alpha_2^{new}$ is 0 or C, b_1^{new}, b_2^{new} and all the numbers between them satisfy the KKT condition, we choose the midpoint as b^{new} .
- Each time we nish the optimization over two variables, we need to update E_i :
$E_i^{new} = \sum_S y^j \alpha_j K(\mathbf{x}^i, \mathbf{x}^j) + b^{new} - y^i$, where S is the set of all support vectors.

SMO Algorithm

- Input: a training set $S=\{(x^i,y^i),i=1,\cdots,N\}$, accuracy ε ;
- Output: $\hat{\alpha}\hat{}\hat{}=(\alpha_1,\cdots,\alpha_N)$

① Initialize $\alpha^{(0)}=0,k=0.$

② Select optimization variables $\alpha_1^{(k)},\alpha_2^{(k)}$, find the solution of optimization problem, denoted as $\alpha_1^{(k+1)},\alpha_2^{(k+1)}$; update α to $\alpha^{(k+1)}$.

③ If with approximation error ε , we satisfy the following stopping conditions, turn to step 4; otherwise let $k=k+1$ and turn to step 2.

$$\sum_{i=1}^{N}\alpha_i y^i=0$$

$$0\le\alpha_i\le C,\ i=1,2,\cdots,N$$

$$y^i g(\boldsymbol{x}^i)=\begin{cases}\ge 1 & \{\boldsymbol{x}^i\mid a_i=0\}\\ =1 & \{\boldsymbol{x}^i\mid 0<a_i<C\}\\ \le 1 & \{\boldsymbol{x}^i\mid a_i=C\}\end{cases}$$

where $g(\boldsymbol{x}^i)=\sum_{j=1}^{N}\alpha_j y^j K(\boldsymbol{x}^j,\boldsymbol{x}^i)+b$

④ Let $\hat{\alpha}=\alpha^{(k+1)}$.

Summary of SMO

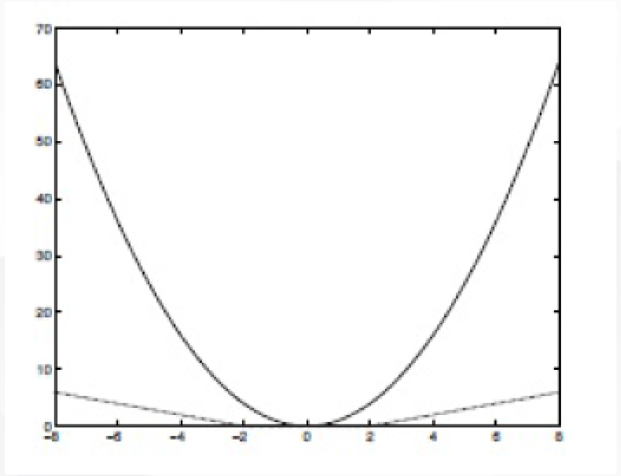
- **SMO** is a heuristic algorithm.
- The basic idea is that: If all variables satisfy the KKT condition, then we have obtained the optimal solution of the optimization problem. Otherwise, we can choose two variables, while keeping the other variables fixed, to construct a quadratic programming problem. The solution of the subproblem is nearer to the primal optimization problem, since it will make the objective function much smaller. More importantly, the subproblem can usually be efficiently solved by its closed form solution.
- The two variables are chosen as follows: the first variable is chosen to be the one who most violates the KKT condition, while the second one is automatically determined by the constraints. With the above strategies, **SMO** continuously divides the primal optimization problem to several subproblems, and it can find the optimal solution by iteratively solve these subproblems.

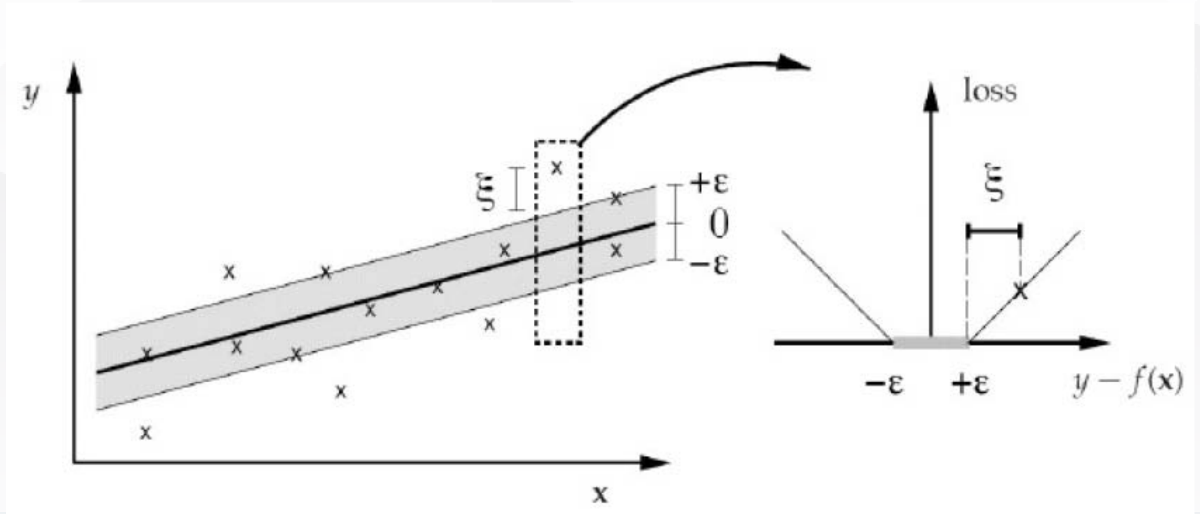
Support Vector Regression: -Insensitive Loss Function

- In order for the sparseness property of support vectors in SVM for classification to carry over to support vector regression (SVR), we do not use the squared loss but the -ε insensitive loss function:

$$\varepsilon_i(y^i, f(\mathbf{x}^i)) = \begin{cases} 0 & \text{if } |y^i - f(\mathbf{x}^i)| \le \varepsilon \\ |y^i - f(\mathbf{x}^i)| - \varepsilon & \text{otherwise} \end{cases}$$

- Two characteristics:
  - •Errors are tolerated up to a threshold of ε.
  - •Errors beyond ε have linear (rather than quadratic) effect so that the model is more robust against noise.





Support Vector Regression: Primal Optimization Problem

■ Primal Optimization Problem:

$$\min_{w,b} \frac{1}{2}\|\mathbf{w}\|_2^2 + C\sum_{i=1}^{N}(\xi_i^+ + \xi_i^-)$$

$$\begin{aligned} s.t.\ & y^i - (\mathbf{w}^T\mathbf{x}^i + b) \le \varepsilon + \xi_i^+,\ i=1,\dots,N, \\ & \mathbf{w}^T\mathbf{x}^i + b - y^i \le \varepsilon - \xi_i^-,\ i=1,\dots,N, \\ & \xi_i^+, \xi_i^- \ge 0,\quad i=1,\dots,N. \end{aligned}$$

■ Two Types of Slack Variables:

- ξ_i^+ : for positive deviation such that $y^i - (\mathbf{w}^T\mathbf{x}^i + b) > \varepsilon$
- ξ_i^- : for negative deviation such that $\mathbf{w}^T\mathbf{x}^i + b - y^i > \varepsilon$

■ If $y^i - (\mathbf{w}^T\mathbf{x}^i + b) \le \varepsilon$ and $\mathbf{w}^T\mathbf{x}^i + b - y^i \le \varepsilon$, then $\xi_i^+ = \xi_i^- = 0$ contributing no cost to the objective function.

Support Vector Regression: Dual and Kernel

- Similar to SVM for classication, the optimization problem for SVR can also be rewritten in the dual form.
- Nonlinear kernel extension is possible by introducing appropriate kernel functions.
- Due to the sparseness property of the -insensitive loss function, only a small fraction of the training instances are support vectors which are used in dening the regression function (like the discriminant function for classication).

Multi-Class SVM: One Versus All

- Approach: view as multi-class classication task, with every complex output y is one class.
- Training example: $S=\{(\boldsymbol{x}^i, y^i), i=1,\cdots,N\}, \ \boldsymbol{x}^i \in R^D, \ y^i \in \{1,\cdots,m\}$
- Optimization Problem:

$$\min_{\boldsymbol{w},b} \frac{1}{2}\|\boldsymbol{w}\|_2^2 + C\sum_{i=1}^{N}\xi_i$$

$$s.t. \ \ \forall j \neq y^1: \ \ \boldsymbol{w}_{y^1}^T \boldsymbol{x}^1 \geq \boldsymbol{w}_j^T \boldsymbol{x}^1 + 1 - \xi_1,$$

$$\cdots$$

$$\forall j \neq y^N: \ \ \boldsymbol{w}_{y^N}^T \boldsymbol{x}^N \geq \boldsymbol{w}_j^T \boldsymbol{x}^N + 1 - \xi_N,,$$

$$\xi_i \geq 0, \quad i=1,\dots,N.$$

Learning Theory: Notations

- Generalization Error: $\varepsilon(h) = P_{(\boldsymbol{x},y)\sim D} I_{\{h(\boldsymbol{x})\neq y\}}$
- Empirical Risk (Error): $\hat{\varepsilon}(h) = \frac{1}{N}\sum_{i=1}^{N} I_{\{h(\boldsymbol{x}^i)\neq y^i\}}$
- PAC (IID) Assumption: $(\boldsymbol{x}^i, y^i), i = 1, \cdots, N$ are drawn independently from the same distribution of D.
- Consider the setting of linear classication, and let $h_\theta(\boldsymbol{x}) = I_{\{\theta^T \boldsymbol{x}\geq 0\}}$.
- What's a reasonable way of tting the parameters θ ?
- Empirical Risk Minimization (ERM): $\hat{\theta} = \arg\min_\theta \hat{\varepsilon}(h_\theta)$.
- Question: Why should doing well on the training set tell us anything about generalization error? Specically, can we relate error on the training set to generalization error?
- Define the hypothesis class $\boldsymbol{H}$ used by a learning algorithm to be the set of all classiers considered by it. For linear classification $\boldsymbol{H} = \{h_\theta : h_\theta(\boldsymbol{x}) = I_{\{\theta^T \boldsymbol{x}\geq 0\}}\}$ is thus the set of all classifiers over $\boldsymbol{X}$ where the decision boundary is linear.
- ERM can now be written as: $\hat{h} = \arg\min_{h\in \boldsymbol{H}} (\hat{\varepsilon}(h))$.

Learning Theory: the case of finite H

- Consider a finite hypothesis class $\boldsymbol{H} = \{h_1, \cdots, h_k\}$
- We would like to give guarantees on the generalization error of $\hat{h}$
- Take any one fixed $h_i \in \boldsymbol{H}$. Consider a Bernoulli random variable Z whose distribution is defined as follows. Sample $(\boldsymbol{x}, y) \sim D$ and set $Z = I_{\{h_i(\boldsymbol{x}) \neq y\}}$. Similarly, we define $Z^j = I_{\{h_i(\boldsymbol{x}^j) \neq y^j\}}$. Since our data was drawn iid from D, Z and the Z^js have the same distribution.
- The training error can be written as: $\hat{\varepsilon}(h_i) = \frac{1}{N}\sum_{j=1}^{N} Z^j$
- Applying the Hoeffding inequality, and obtain:

$$P(|\varepsilon(h_i) - \hat{\varepsilon}(h_i)| > \gamma) \leq= 2\exp(-2\gamma^2 N)$$

Hoeffding Inequality (Chernoff bound)

Let $Z^1, \cdots, Z^N$ be N iid random variables drawn from a Bernoulli distribution. I.e., $P(Z^i = 1) = \phi$ and $P(Z^i = 0) = 1 - \phi$. Let $\hat{\phi} = \frac{1}{N}\sum_{i=1}^{N} Z_i$ be the mean of these random variables, and let any $\gamma > 0$ be fixed. Then:

$$P(|\phi - \hat{\phi}| > \lambda) \leq 2\exp(-2\gamma^2 n)$$

Learning Theory: the case of finite H

- This shows that, for our particular h_i , training error will be close to generalization error with high probability, assuming n is large. We want to prove that this will be true for simultaneously for all $h \in \boldsymbol{H}$.
- Let $A_i = \{|\varepsilon(h_i) - \hat{\varepsilon}(h_i)| > \gamma\}$, then we have already show that, for any particular $\boldsymbol{A_i}$, it holds true that $P(A_i) \leq 2\exp(-2\gamma^2 N)$.
- Using the union bound, we have that:

$$P(\exists h_i \in H, |\varepsilon(h_i) - \hat{\varepsilon}(h_i)| > \gamma) = P(A_1 \cup \cdots \cup A_k)$$

$$\leq \sum_{i=1}^{k} P(A_i) \leq \sum_{i=1}^{k} 2\exp(-2\gamma^2 N) = 2k\exp(-2\gamma^2 N)$$

- Then: $P(\forall h_i \in H, |\varepsilon(h_i) - \hat{\varepsilon}(h_i)| \leq \gamma) \geq 1 - 2k\exp(-2\gamma^2 N)$
- With probability at least $1 - 2k\exp(-2\gamma^2 N)$, we have that $\varepsilon(h)$ will be within of $\hat{\varepsilon}(h)$ for all $h \in \boldsymbol{H}$. This is called uniform convergence result, because this is a bound that holds simultaneously for all (as opposed to just one) $h \in \boldsymbol{H}$.

Learning Theory: the case of finite H

- There are three quantities of interest here: N, γ and the probability of error; we can bound either one in terms of the other two.
- **Given γ and some $\delta > 0$, how large must n be before we can guarantee that with probability at least $1-\delta$, training error will be within of generalization error?**
  - By setting $\delta = 2k\exp(-2\gamma^2 N)$ and solving for N, we find that if $N \geq \frac{1}{2\gamma^2}\log\frac{2k}{\delta}$, then with probability at least $1-\delta$, we have that $|\varepsilon(h) - \hat{\varepsilon}(h)| > \gamma$ for some $h \in \boldsymbol{H}$. ((Equivalently, this shows that the probability that $|\varepsilon(h) - \hat{\varepsilon}(h)| > \gamma$ for some $h \in \boldsymbol{H}$ is at most δ.))
  - This bound tells us how many training examples we need in order make a guarantee. The training set size n that a certain method or algorithm requires in order to achieve a certain level of performance is also called the algorithms sample complexity.
- we can also hold N and δ fixed, and solve for γ in the previous equation, and show that with probability $1-\delta$, we have that for all $h \in \boldsymbol{H}$

$$|\hat{\varepsilon}(h) - \varepsilon(h)| \leq \sqrt{\frac{1}{2N}\log\frac{2k}{\delta}}$$

Learning Theory: the case of finite $\boldsymbol{H}$

- Now, lets assume that uniform convergence holds, i.e., that for all $|\varepsilon(h) - \hat{\varepsilon}(h)| \le \gamma$. **What can we prove about the generalization of our learning algorithm that picked** $\hat{h} = \arg\min_{h \in \boldsymbol{H}} (\hat{\varepsilon}(h))$?
- Define $h^* = \arg\min_{h \in \boldsymbol{H}} (\varepsilon(h))$ to be the best possible hypothesis in $\boldsymbol{H}$.
- Note that h^* is the best that we could possibly do given that we are using $\boldsymbol{H}$, so it makes sense to compare our performance to that of h^*. We have: $\varepsilon(\hat{h}) \le \hat{\varepsilon}(h) + \gamma \le \varepsilon(h^*) + \gamma \le \varepsilon(h^*) + 2\gamma$.
- If uniform convergence occurs, then the generalization error of $\hat{h}$ is at most 2γ worse than the best possible hypothesis in $\boldsymbol{H}$!

Theorem

Let $|\boldsymbol{H}| = k$, any N, γ be fixed. Then with probability at least $1 - \delta$:

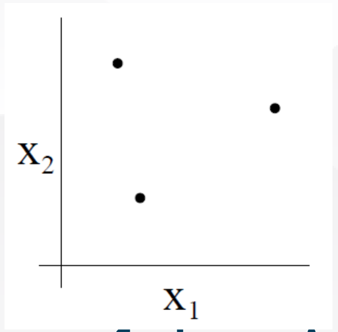
$$\hat{\varepsilon}(h) \le (\min_{h \in \boldsymbol{H}} \varepsilon(h)) + 2\sqrt{\frac{1}{2N} \log \frac{2k}{\delta}}$$

Learning Theory: the case of infinite H

- We have proved some useful theorems for the case of finite hypothesis classes. But many hypothesis classes, including any parameterized by real numbers (as in linear classication) actually contain an infinite number of functions. Can we prove similar results for this setting?
- Given a set $S=\{\boldsymbol{x}^1,\cdots,\boldsymbol{x}^d\}$ (no relation to the training set) of points $\boldsymbol{x}^1 \in \boldsymbol{X}$, we say that $\boldsymbol{H}$ shatters S if $\boldsymbol{H}$ can realize any labeling on S. I.e., if for any set of labels $\{y^1,\cdots,y^d\}$, there exists some $h \in \boldsymbol{H}$ so that $h(\boldsymbol{x}^i)=y^i$ for all $i=1,\cdots,d$.
- Given a hypothesis class $\boldsymbol{H}$, we then define its Vapnik-Chervonenkis dimension, written VC($\boldsymbol{H}$), to be the size of the largest set that is shattered by $\boldsymbol{H}$. (If $\boldsymbol{H}$ can shatter arbitrarily large sets, then VC($\boldsymbol{H}$) = ∞ .)

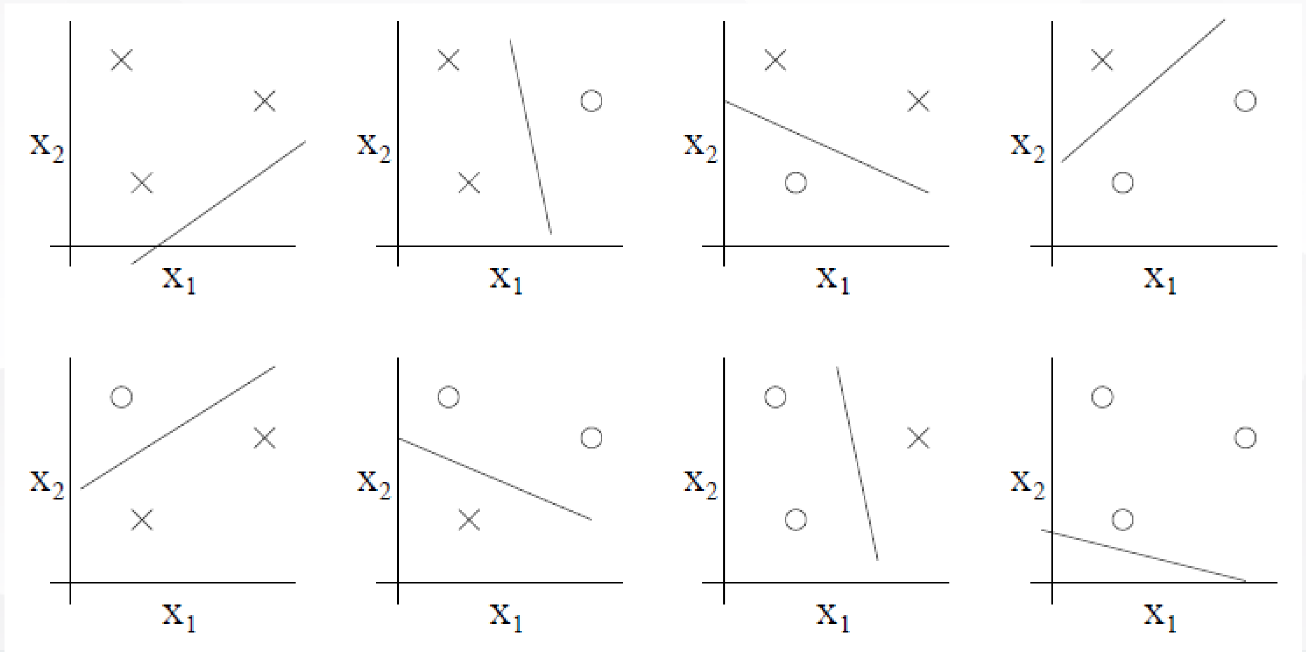
VC Dimension

- For instance, consider the following set of three points:

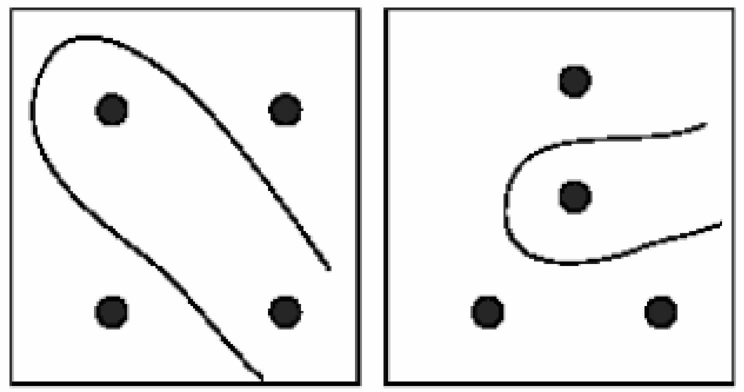


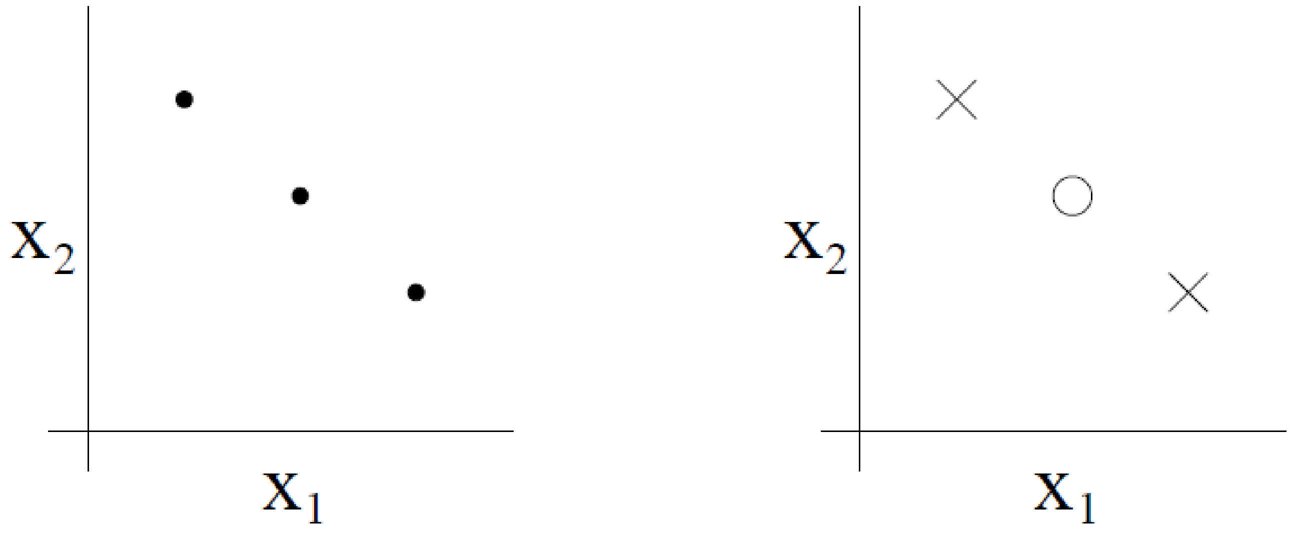
Can the set $\boldsymbol{H}$ of linear classifiers in two dimensions ($h(x) = I_{\{\theta_0 + \theta_1 x_1 + \theta_2 x_2 \geq 0\}}$) shatter the set above? The answer is yes.

- For any of the eight possible labelings of these points, we can find a linear classier that obtains zero training error on them:



VC Dimension





- There is no set of 4 points that this hypothesis class can shatter. Thus, the largest set that ***H*** can shatter is of size 3, i.e., VC(***H***) = 3.
- Note that the VC dimension of ***H*** here is 3 even though there may be sets of size 3 that it cannot shatter. In order words, in order to prove that VC(***H***) is at least d, we need to show only that theres at least one set of size d that ***H*** can shatter.

VC Dimension

Theorem

Let $\boldsymbol{H}$ be given , and let $d = VC(\boldsymbol{H})$ be fixed. Then with probability at least $1-\delta$, we have that for all $h \in \boldsymbol{H}$,

$$|\varepsilon(h) - \hat{\varepsilon}(h)| \leq O(\sqrt{\frac{d}{N}\log\frac{N}{d} + \frac{1}{N}\log\frac{1}{\delta}})$$

Thus with probability at least $1-\delta$, we also have that :

$$\varepsilon(\hat{h}) \leq \varepsilon(h^*) + O(\sqrt{\frac{d}{N}\log\frac{N}{d} + \frac{1}{N}\log\frac{1}{\delta}})$$

- In other words, if a hypothesis class has nite VC dimension, then uniform convergence occurs as N becomes large. As before, this allows us to give a bound on $\varepsilon(h)$ in terms of $\hat{\varepsilon}(h)$.

VC Dimension

- We also have the following corollary:

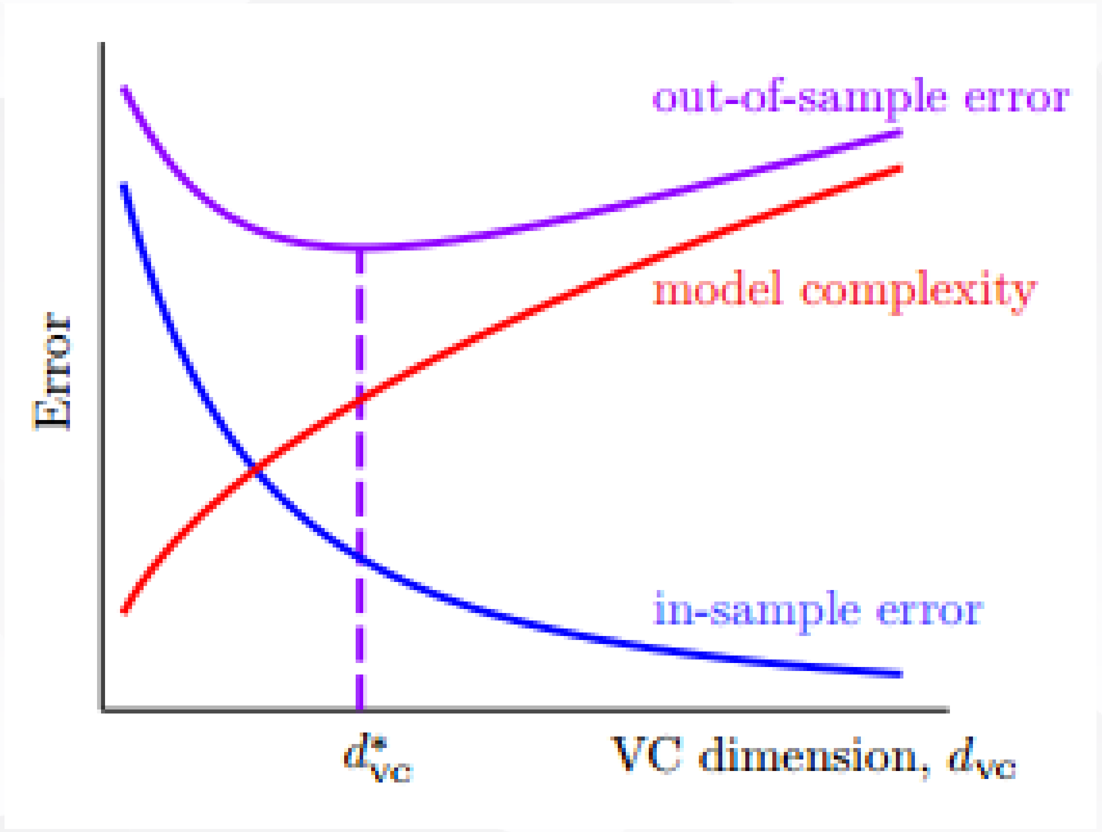
Corollary

For $|\varepsilon(h) - \hat{\varepsilon}(h)| \leq \gamma$ to hold for all $h \in \boldsymbol{H}$ (and hence $\varepsilon(\hat{h}) \leq \varepsilon(h^*) + 2\gamma$) with probability at least $1-\delta$, it suffices that $N = O_{\gamma,\delta}(d)$.

- In other words, the number of training examples needed to learn well using H is linear in the VC dimension of H. It turns out that, for most hypothesis classes, the VC dimension (assuming a reasonable parameterization) is also roughly linear in the number of parameters. Putting these together, we conclude that (for an algorithm that tries to minimize training error) the number of training examples needed is usually roughly linear in the number of parameters of H.

Summary of VC Dimension

- VC dimension is a measure of model complexity.



- When n is large, we can accommodate models with large complexity (VC dimension).

Generalization Analysis of SVM

- The following Theorem by Vapnik (1982) provides the essential link between margin and realized classier class complexity for SVMs.

Theorem: Vapnik 1982

The class of optimal linear separators has VC dimension h bounded from above as:

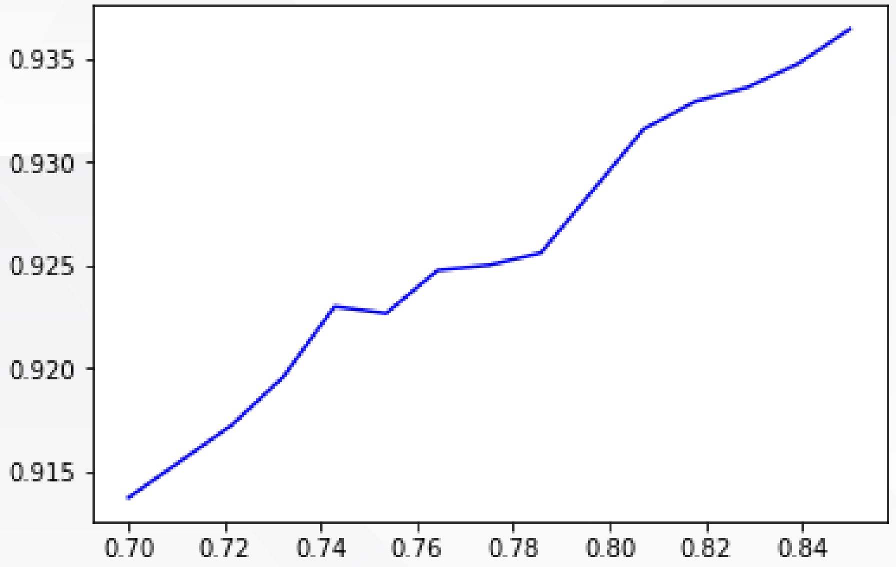
$$h \le \min\{\left\lceil \frac{4r^2}{\rho^2} \right\rceil, m\} + 1,$$

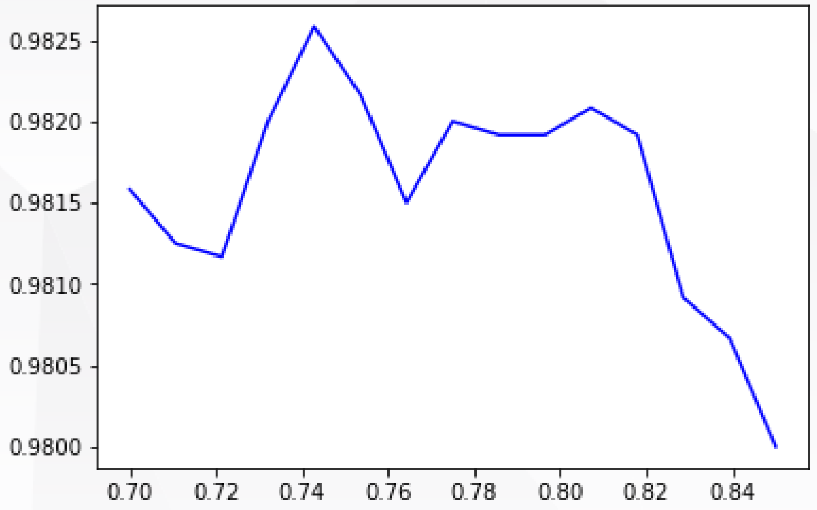
where ρ is the margin, r is the radius of the smallest sphere that can enclose all of the training examples, and m is the dimensionality of $\boldsymbol{X}$.

- Intuitively, this implies that regardless of dimensionality m we can minimize the VC dimension by maximizing the margin ρ .
- Thus, complexity of the classier is kept small regardless of dimensionality.

Summary of Supervised Learning

- Framework of Statistical Machine Learning
- Bias and Variance Decomposition
- Overfitting and Regularization
- Discriminative and Generative Approach
- Linear Regression
- Linear Classification
- Naive Bayesian
- Stochastic Gradient Descent
- SVM and Dual
- Kernel and Learning Theory





保留的信息量
降维后，特征维度越高，保留的信息量越多

- 10类 准确率：(RBF)0.9820982098

- 10类 准确率：0.98209820