Recap

- We are discussing the SVM method for learning classifiers.
- Objective is to learn the optimal hyperplane – one that maximizes the ‘margin of separation’.
- It can be formulated as a constrained optimization problem.

Linear SVM – data linearly separable

- The optimal hyperplane is a solution of

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2} W^T W \\ \text{subject to} \quad & y_i (W^T X_i + b) \geq 1, \;\; i = 1, \ldots, n \end{aligned}$$

- We solve the dual given by

$$\begin{aligned} \max_{\boldsymbol{\mu}} \quad & q(\boldsymbol{\mu}) = \sum_{i=1}^{n} \mu_i - \frac{1}{2} \sum_{i,j=1}^{n} \mu_i \mu_j y_i y_j X_i^T X_j \\ \text{subject to} \quad & \mu_i \geq 0, \;\; i = 1, \ldots, n, \;\; \sum_{i=1}^{n} y_i \mu_i = 0 \end{aligned}$$

- Then the final solution is:

$$W^* = \sum \mu_i^* y_i X_i, \;\; b^* = y_j - X_j^T W^*, \;\; j \text{ such that } \mu_j > 0$$

Linear SVM

- The primal problem is

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2}W^T W \; + \; C\sum_{i=1}^{n} \xi_i \\ \text{subject to} \quad & y_i(W^T X_i + b) \geq 1 - \xi_i, \;\; i = 1, \ldots, n \\ & \xi_i \geq 0, \;\; i = 1, \ldots, n \end{aligned}$$

- The dual problem is:

$$\max_{\boldsymbol{\mu}} \quad q(\boldsymbol{\mu}) = \sum_{i=1}^{n} \mu_i - \frac{1}{2}\sum_{i,j=1}^{n} \mu_i \mu_j y_i y_j X_i^T X_j$$

$$\text{subject to} \quad 0 \leq \mu_i \leq C, \;\; i = 1, \ldots, n, \;\; \sum_{i=1}^{n} y_i \mu_i = 0$$

- We solve dual and the final optimal hyperplane is
$W^* = \sum \; \mu_i^* y_i X_i,$
$b^* = y_j - X_j^T W^*, \; (j \; s.t. \; 0 < \mu_j < C).$

The Linear SVM

- Given training data, $(X_i, y_i), i = 1, \cdots, n$, we solve

$$\max_{\boldsymbol{\mu}} \qquad q(\boldsymbol{\mu}) = \sum_{i=1}^{n} \mu_i - \frac{1}{2} \sum_{i,j=1}^{n} \mu_i \mu_j y_i y_j X_i^T X_j$$

$$\text{subject to} \qquad 0 \le \mu_i \le C, \;\; i = 1, \ldots, n, \;\; \sum_{i=1}^{n} y_i \mu_i = 0$$

- The SVM is a linear classifier specified by W^*, b^*:

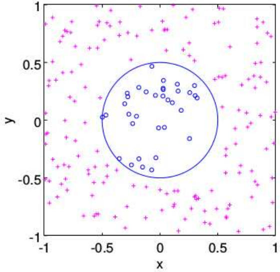
$$W^* = \sum \mu_i^* y_i X_i, \quad b^* = y_j - X_j^T W^*, \; (j \; s.t. \; 0 < \mu_j < C)$$

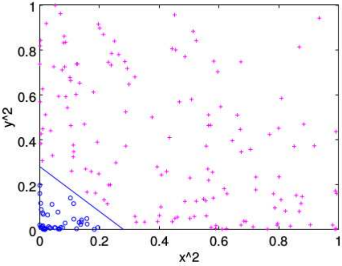
- Given a new pattern, X, its class is determined by sign of

$$\begin{aligned} f(X) &= X^T W^* + b^* = \sum_i \mu_i^* y_i X_i^T X + b^* \\ &= \sum_{i:\mu_i > 0} \mu_i^* y_i X_i^T X + (y_j - \sum_{i:\mu_i > 0} \mu_i^* y_i X_i^T X_j) \end{aligned}$$

- We first formulated SVM for linearly separable case to understand the idea.
- Then, using slack variables, ξ_i, we have a linear SVM where we make no assumptions about separability.
- In the dual, the only difference is an upperbound on μ_i.
- Using this formulation, we can find 'best' hyperplane classifier.
- The next question is how can we learn non-linear classifiers?
- Recall that the SVM idea is to transform X_i into some other high-dimensional space and learn a linear classifier there.

Transforming Patterns to become Linearly Separable





Non-linear classifiers

- In general, we can use a mapping, $\phi : \Re^m \rightarrow \Re^{m'}$.
- In $\Re^{m'}$, the training set is $\{(Z_i,\ y_i),\ i = 1, \ldots, n\}$, $Z_i = \phi(X_i)$.
- We can find optimal hyperplane by solving the dual (replacing $X_i^T X_j$ with $Z_i^T Z_j$).
- The dual problem now would be the following.

$$\max_{\mu} \qquad q(\mu) = \sum_{i=1}^{n} \mu_i - \frac{1}{2} \sum_{i,j=1}^{n} \mu_i \mu_j y_i y_j \phi(X_i)^T \phi(X_j)$$

$$\text{subject to} \qquad 0 \leq \mu_i \leq C, \ \ i = 1, \ldots, n, \ \ \sum_{i=1}^{n} y_i \mu_i = 0$$

- The optimization is over $\Re^n$ (with quadratic cost function & linear constraints) **irrespective of ϕ and m'**.
- But computationally expensive?

Kernel function

- Suppose we have a function, $K : \Re^m \times \Re^m \to \Re$, that satisfies

  $K(X_i,\ X_j) = \phi(X_i)^T \phi(X_j)$

  Called Kernel function.
- Suppose computation of $K(X_i,\ X_j)$ is about as expensive as that of $X_i^T X_j$.
- Replacing $\phi(X_i)^T \phi(X_j)$ by $K(X_i,\ X_j)$, we can solve dual without ever computing any $\phi(X_i)$. Efficient for obtaining optimal hyperplane.
- What about storing W^*? Computing $\phi(X)^T W^*$ for new patterns?

Kernel function based classifier

- Let μ_i^* be soln of Dual. Then $W^* = \sum \mu_i^* y_i \phi(X_i)$.
- We also have

$$b^* = y_j - \phi(X_j)^T W^* = y_j - \sum_i \mu_i^* y_i \phi(X_i)^T \phi(X_j)$$

- Given a new pattern X we only need to compute

$$\begin{aligned}
f(X) &= \phi(X)^T W^* + b^* \\
&= \sum_i \mu_i^* y_i \phi(X_i)^T \phi(X) + (y_j - \phi(X_j)^T W^*) \\
&= \sum_i \mu_i^* y_i K(X_i,\ X) + \left(y_j - \sum_i \mu_i^* y_i K(X_i,\ X_j) \right)
\end{aligned}$$

$$= \sum_{i:\mu_i > 0} \mu_i^* y_i K(X_i,\ X) + \left(y_j - \sum_{i:\mu_i > 0} \mu_i^* y_i K(X_i,\ X_j) \right)$$

- This is an interesting way of learning nonlinear classifiers.
- We solve the dual whose dimension is n, number of examples. (We do not need $\phi(X_i)$ for solving dual)
- All we need to store are:
  - non-zero Lagrange multipliers: $\mu_i^* > 0$,
  - Support vectors: X_i, i s.t. $\mu_i^* > 0$.
- Then, given an X, we compute

$$f(X) = \sum_{i:\mu_i>0} \mu_i^* y_i K(X_i,\ X) + \left(y_j - \sum_{i:\mu_i>0} \mu_i^* y_i K(X_i,\ X_j) \right)$$

  and classify X based on sign of $f(X)$.
- Never need to enter '$\phi(X)$' space!

Support Vector Machine

- Obtain μ_i^* by solving the Dual with $\phi(X_i)^T \phi(X_j)$ replaced by $K(X_i,\ X_j)$. (Choose a suitable Kernel function. Use 'penalty const', C as needed).
- Store non-zero μ_i^* and the corresponding support vectors.
- Classify any new pattern X by sign of

$$f(X) = \sum \mu_i^* y_i K(X_i,\ X) + \left(y_j - \sum_i \mu_i^* y_i K(X_i,\ X_j) \right)$$

- If we have a suitable Kernel function, we never need to compute $\phi(X)$.
- The range space of ϕ can even be infinite dimensional!

Example kernel function

- We start with an example kernel function in $\Re^2$.
- Consider $K(X_i,\ X_j) = (1 + X_i^T X_j)^2$.
- Let $X_i = (x_{i1},\ x_{i2})^T \in \Re^2$ and similarly for X_j.
- Then
$$K(X_i,\ X_j) = (1 + x_{i1}x_{j1} + x_{i2}x_{j2})^2$$
- We now show that there exists a mapping ϕ such that $K(X_i, X_j) = \phi(X_i)^T \phi(X_j)$.

- Consider $\phi : \Re^2 \rightarrow \Re^6$ given by

$$Z = \phi(X) = [1 \quad x_1^2 \quad x_2^2 \quad \sqrt{2}x_1 \quad \sqrt{2}x_2 \quad \sqrt{2}x_1x_2]$$

(Here, $X = (x_1 \;\; x_2) \in \Re^2$).

- It is easy to see that a linear discriminant function in terms of Z (i.e., in $\Re^6$) would be a quadratic discriminant function in terms of X (i.e., in $\Re^2$).
- Now we show that

$$K(X_i, X_j) = (1 + X_i^T X_j)^2 = Z_i^T Z_j = \phi(X_i)^T \phi(X_j)$$

- Recall

$$Z_i = \phi(X_i) = [1 \quad x_{i1}^2 \quad x_{i2}^2 \quad \sqrt{2}x_{i1} \quad \sqrt{2}x_{i2} \quad \sqrt{2}x_{i1}x_{i2}]$$

We have

$$\begin{aligned} Z_i^T Z_j &= 1 + x_{i1}^2 x_{j1}^2 + x_{i2}^2 x_{j2}^2 + 2x_{i1}x_{j1} + 2x_{i2}x_{j2} + 2x_{i1}x_{i2}x_{j1}x_{j2} \\ &= (1 + x_{i1}x_{j1} + x_{i2}x_{j2})^2 \\ &= \left(1 + X_i^T X_j\right)^2 = K(X_i,\ X_j) \end{aligned}$$

- Easy to see it works for $X \in \Re^n$ in general.
- Thus $K(X_i,\ X_j) = (1 + X_i^T X_j)^2$ results in a quadratic discriminant function or a quadratic classifier.

- From this example, it is also easy to see that for a given Kernel function, the mapping ϕ (or the dimension of its range space) is not unique.
- Consider the same Kernel fn $K(X_i, X_j) = (1 + X_i^T X_j)^2$.
- Consider the mapping $\phi : \Re^2 \rightarrow \Re^7$ given by

$$Z = \phi(X) = [1 \quad \sqrt{2}x_1 \quad \sqrt{2}x_2 \quad x_1^2 \quad x_2^2 \quad x_1 x_2 \quad x_1 x_2]$$

- It is easy to see that this mapping also works.

- We saw that the Kernel $K(X, X') = (1 + X^T X')^2$ results in a quadratic discriminant function (in the original feature space)
- This is because the effective ϕ function is such that each $x_i x_j$ term is a component of $\phi(X)$.
- Thus, if $X \in \Re^m$, then any reasonable ϕ function corresponding to this kernel would have range space with dimension $O(m^2)$.
- Hence, $\phi(X_i)^T \phi(X_j)$ would need $O(m^2)$ multiplications.
- If we are using a linear SVM, we only need $X_i^T X_j$ which needs m multiplications.
- When we use the Kernel for the quadratic case, we need only $m + 1$ multiplications.

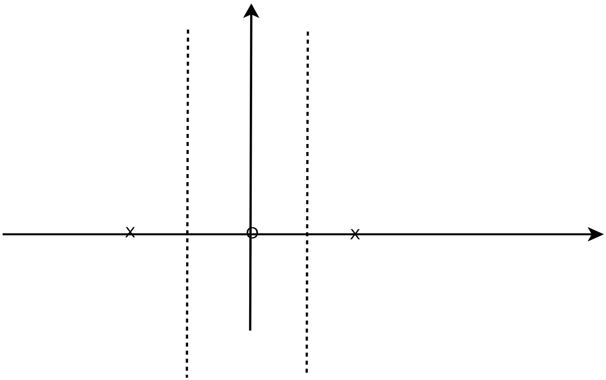
Example

- We will first consider a very simple example problem in $\Re^2$ to get a feel for the method of obtaining SVM.
- Suppose we have 3 examples:

$$X_1 = (-1, 0), \quad X_2 = (1, 0), \quad X_3 = (0, 0)$$

with $y_1 = y_2 = +1$ and $y_3 = -1$.

Example



- As is easy to see, a linear classifier is not sufficient here.
- Suppose we use the Kernel function:
  $K(X, X') = (1 + X^T X')^2$.

- Recall, the examples are

$$X_1 = (-1, 0), \quad X_2 = (1, 0), \quad X_3 = (0, 0)$$

- The objective function involves $K(X_i, X_j)$. These are given in a matrix below.

$$\left[(1 + X_i^T X_j)^2\right] = \begin{bmatrix} 4 & 0 & 1 \\ 0 & 4 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

- The objective function to be maximized is

$$
\begin{aligned}
q(\boldsymbol{\mu}) &= \sum_{i=1}^{3} \mu_i - \frac{1}{2} \sum_{i,j=1}^{3} \mu_i \mu_j y_i y_j K(X_i, X_j) \\
&= \sum_{i=1}^{3} \mu_i - \frac{1}{2}(4\mu_1^2 + 4\mu_2^2 + \mu_3^2 - 2\mu_1\mu_3 - 2\mu_2\mu_3)
\end{aligned}
$$

- The constraints are

$$\mu_1 + \mu_2 - \mu_3 = 0; \quad \text{and} \quad -\mu_i \leq 0,\ i = 1, 2, 3.$$

- The lagrangian for this problem is

$$L(\boldsymbol{\mu}, \lambda, \boldsymbol{\alpha}) = q(\boldsymbol{\mu}) + \lambda(\mu_1 + \mu_2 - \mu_3) - \sum_{i=1}^{3} \alpha_i \mu_i$$

  Here, λ is the Lagrange multiplier for the equality constraint and α_i are the langrange multipliers for the inequality constraints.
- Using Kuhn-Tucker conditions, we have $\frac{\partial L}{\partial \mu_i} = 0$ and $\mu_1 + \mu_2 - \mu_3 = 0$.
- This gives us four equations; we have 7 unknowns (Three μ_i, three α_i and λ).
- By complementary slackness, we have $\alpha_i \mu_i = 0$. Essentially, we need to guess which $\mu_i > 0$.

- In this simple problem we know all $\mu_i > 0$.
- This is because all X_i would be support vectors.
- Hence we take all $\alpha_i = 0$.
- We have now four unknowns: $\mu_1, \mu_2, \mu_3, \lambda$.
- Using $\frac{\partial L}{\partial \mu_i} = 0$, $i = 1, 2, 3$ and feasibility, we can solve for μ_i.

- Recall

$$
\begin{aligned}
L(\boldsymbol{\mu}, \lambda, \boldsymbol{\alpha}) &= q(\boldsymbol{\mu}) + \lambda(\mu_1 + \mu_2 - \mu_3) - \sum_{i=1}^{3} \alpha_i \mu_i \\
q(\boldsymbol{\mu}) &= \sum_{i=1}^{3} \mu_i - \frac{1}{2}(4\mu_1^2 + 4\mu_2^2 + \mu_3^2 - 2\mu_1\mu_3 - 2\mu_2\mu_3)
\end{aligned}
$$

- Now $\frac{\partial L}{\partial \mu_i} = 0$, $i = 1, 2, 3$ and feasibility give

$$
\begin{aligned}
1 - 4\mu_1 + \mu_3 + \lambda &= 0 \\
1 - 4\mu_2 + \mu_3 + \lambda &= 0 \\
1 - \mu_3 + \mu_1 + \mu_2 - \lambda &= 0 \\
\mu_1 + \mu_2 - \mu_3 &= 0
\end{aligned}
$$

- These give us $\lambda = 1$ and $\mu_3 = 2\mu_1 = 2\mu_2$.
- Thus we get $\mu_1 = \mu_2 = 1$ and $\mu_3 = 2$.
- This completely determines the SVM

- If we used the penalty constant with $C \geq 2$ we get the same solution.
  (If $C < 2$, we can not get this solution).
- The calssification of any X by this SVM is by the sign of $f(X)$:

$$
\begin{aligned}
f(X) &= \sum_i \mu_i y_i K(X_i, X) \; + \; b^* \\
&= K(X_1, X) + K(X_2, X) - 2K(X_3, X) + b^*
\end{aligned}
$$

- Let us first calculate b^*.

- Recall the formula

$$b^* = y_j - \sum_i \mu_i y_i K(X_i, X_j), \quad j \; s.t. \; 0 < \mu_j$$

- Recall

$$[K(X_i, X_j)] = \left[(1 + X_i^T X_j)^2\right] = \begin{bmatrix} 4 & 0 & 1 \\ 0 & 4 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

- With $j = 1$ we get $b^* = 1 - (4 + 0 - 2) = -1$.
- With $j = 3$ we get $b^* = -1 - (1 + 1 - 2) = -1$.
- If we solved our optimization problem correctly, we should get same b^*!

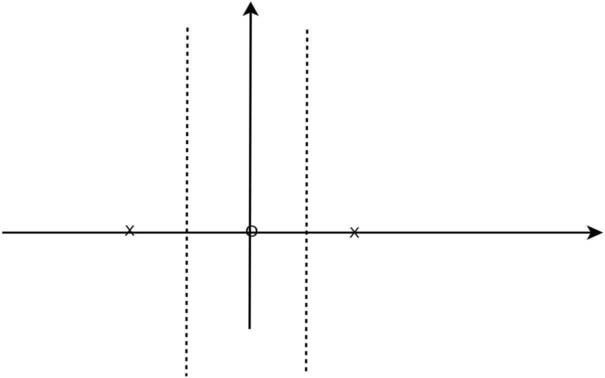
- We have $X_1 = (-1, 0), \quad X_2 = (1, 0), \quad X_3 = (0, 0)$ and $K(X, X') = (1 + X^T X')^2$.
- Hence, taking $X = (x_1, x_2)^T$, we have

$$\begin{aligned} f(X) &= K(X_1, X) + K(X_2, X) - 2K(X_3, X) + b^* \\ &= (1 - x_1)^2 + (1 + x_1)^2 - 2(1) - 1 \\ &= 2x_1^2 - 1 \end{aligned}$$

- Hence this SVM will assign class $+1$ to $X = (x_1, x_2)^T$ if

$$2x_1^2 \geq 1 \quad \text{or} \quad |x_1| \geq \frac{1}{\sqrt{2}}$$

Recall Example Patterns



- Hence this SVM will assign class $+1$ to $X = (x_1, x_2)^T$ if

$$2x_1^2 \geq 1 \quad \text{or} \quad |x_1| \geq \frac{1}{\sqrt{2}}$$

- Why not $|x_1| \geq (1/2)$?
- We are maximizing margin of the hyperplane in 'x^2'-space.
- The final SVM is intuitively very reasonable and we solve essentially the same problem whether we are seeking a linear classifier or a nonlinear classifier.

Kernel functions

- How do we obtain Kernel functions in general?
- What kind of symmetric functions capture the inner product in an appropriate space?
- We look at two important charectarizations for Kernel functions.

Mercer Kernels

- **Mercer Theorem**:
  Given a symmetric function, $K : \Re^m \times \Re^m \to \Re$,

  there exists an inner product space $\mathcal{H}$ and a mapping $\phi : \Re^m \to \mathcal{H}$ so that
  $K(X_1,\ X_2) = \phi(X_1)^T \phi(X_2)$

  if for all square-integrable functions g,

$$\int K(X_1,\ X_2) g(X_1) g(X_2) dX_1\, dX_2 \geq 0.$$

Positive definite kernels

- Let $\bar{K}$ be a $n \times n$ matrix with $\bar{K}_{i,j} = K(X_i,\ X_j)$.
- A **positive definite kernel** is the function K such that $\bar{K}$ is positive semi-definite for all n and all data sets $\{X_1, \ldots, X_n\}$.
- That is, given any n, and any feature vectors, $X_1, \cdots, X_n$, we have, for all scalars $c_1, \cdots, c_n$,

$$\sum_{i,j=1}^{n} c_i c_j K(X_i, X_j) \geq 0$$

- If input space is compact, both these notions are same.

- Now we use Mercer's theorem to show that the function we gave earlier would be a Kernel function.
- Consider the function

$$K(U,\ V) = (U^T V)^p = \left(\sum_{i=1}^{m} u_i v_i \right)^p$$

where $p > 0$ is an integer and $U = [u_1 \cdots u_m]^T$ and $V = [v_1 \cdots v_m]^T$ are in $\Re^m$.
- We want to show that this satisfies the Mercer theorem.

- By expanding the $(U^TV)^p$ we get an expression

$$\left(\sum_{i=1}^{m} u_i v_i\right)^p = \sum_{r1,\,\cdots\, rm} \frac{p!}{r1!\, r2!\, \cdots\, rm!} \prod_{i=1}^{m} (u_i v_i)^{ri}$$

where the summation is over all non-negative integers, $r1,\, \cdots,\, rm$ such that

$$r1 + r2 + \cdots + rm = p$$

- We need to show

$$\int_{\Re^m} \int_{\Re^m} \left(\sum_{i=1}^{m} u_i v_i \right)^p g(U)\, g(V) dU\, dV > 0.$$

- This becomes a sum of integrals by expanding $(\sum u_i v_i)^p$.
- A typical term here is

$$\frac{p!}{r1!\, r2! \cdots rm!} \int \int (u_1 v_1)^{r1} (u_2 v_2)^{r2} \cdots (u_m v_m)^{rm} g(U) g(V) dU dV$$

$$= \frac{p!}{r1!\, r2! \cdots rm!} \int (u_1)^{r1} (u_2)^{r2} \cdots (u_m)^{rm} g(U) dU \int (v_1)^{r1} (v_2)^{r2} \cdots (v_m)^{rm} g(V)\, dV$$

$$= \frac{p!}{r1!\, r2! \cdots rm!} \left(\int u_1^{r1}\, u_2^{r2} \cdots u_m^{rm} g(U)\, dU\right)^2 \geq 0$$

- Now consider the function

$$K(U,V) = \sum_{j=0}^{p} a_j \, (U^T V)^j, \;\; a_j \geq 0$$

- We can show this also satisfies Mercer theorem

$$\begin{gathered} \int \sum_{j=0}^{p} a_j \, (U^T V)^j \, g(U) \, g(V) \, dU \, dV \\ = \sum_{j=0}^{p} a_j \, \int (U^T V)^j \, g(U) \, g(V) \, dU \, dV \\ \geq 0 \end{gathered}$$

- Hence functions of the form

$$K(X_1, X_2) = \sum_{j=0}^{p} a_j \, (X_1^T X_2)^j, \;\; a_j \geq 0$$

  are kernels (satisfying Mercer's theorem).
- A special case is

$$K(X_1, X_2) = (1 + X_1^T X_2)^p$$

  which is an example we considered earlier.
- This is called a polynomial kernel.

- We showed that the function

$$K(X_1, X_2) = (1 + X_1^T X_2)^p$$

  satisfies the Mercer theorem.
- Hence it is a (mercer) kernel.
- It is easy to show that it satisfies the definition of a positive definite kernel
- Hence it is also a positive definite kernel

- Now consider the functions of the type

$$K(U,V) = \sum_{j=0}^{\infty} a_j \, (U^T V)^j, \;\; a_j \geq 0$$

- Our proof only involved interchanging integration and summation.
- For finite sum it is always possible.
- For infinite sum, a sufficient condition is that the above sum is uniformly convergent
- Then the above would also satisfy Mercer's theorem.

- Consider the function

$$K(X_1,\ X_2) = e^{-\frac{(X_1-X_2)^T(X_1-X_2)}{2\sigma^2}}$$

- We can show it satisfies the theorem by noting

$$e^{-(X_1-X_2)^T(X_1-X_2)} = e^{-X_1^T X_1}\ e^{-X_2^T X_2} e^{2X_1^T X_2},$$

and

$$e^{2X_1^T X_2} = \sum_{p=0}^{\infty} \frac{(2X_1^T X_2)^p}{p!}$$

Some Popular Kernel functions

- Polynomial kernel:

$$K_p(X_1,\ X_2) = (1 + X_1^T X_2)^p$$

- Gaussian kernel

$$K_G(X_1,\ X_2) = e^{-\frac{||X_1 - X_2||^2}{\sigma^2}}$$

Generalization abilities

- SVM idea: learn linear classifier in a transformed space.
- We said that naively transforming patterns into a high dimensional space does not work.
- There were two issues in that ‘curse of dimensionality’
- Computational complexity – ‘elegantly’ solved by the kernel trick.
- But, does SVM generalize well?
- We are finding a hyperplane in a very high dimensional space. Do we need very large number of examples?

- In practice, SVMs perform well.
- They can learn classifiers that do well on test data without needing (correspondingly) large number of examples.
- The reason, essentially, is that we learn a hyperplane with large margin.
- There are different ways to analyze this.
- We would just state one theoretical result.

Some theoretical results

- Let P^n_{err} be the error rate on a test set for an SVM trained with n random examples.
- Then we can show (for SVM with no slack variables)

$$EP^n_{\mathsf{err}} \leq \min\left(\frac{s}{n}, \frac{[R^2||W||^2]}{n}, \frac{m}{n}\right)$$

where s is number of support vectors, R is the radius of smallest sphere enclosing all examples, $||W||^{-2}$ is the margin of the maximum margin hyperplane (in the feature space of dimension m).

- We have

$$EP^n_{\mathsf{err}} \leq \min\left(\frac{s}{n},\ \frac{[R^2||W||^2]}{n},\ \frac{m}{n}\right)$$

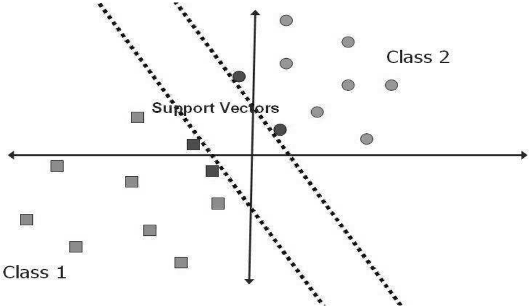
- Optimal hyperplane may generalize well because
    - 'data compression' is large
    - Margin is large
    - dimension of feature space is small

- The fraction of support vectors is an upperbound on expected generalization error.
- This is useful as a confidence measure on the learnt SVM.
- This essentially comes from the concept of ‘stability’ of a learning algorithm.

- Let $\{X_1, \cdots, X_n\}$ be a data set from which we learnt a classifier.
- let Z be another random example. If we gave $\{Z, X_2, \cdots, X_n\}$ as the training set we expect to learn a very similar classifier.
- Similarly we can make n data sets by replacing each X_i in turn with Z.
- The learning algorithm is stable if the errors on the set $\{Z, X_1, \cdots, X_n\}$ of all these classifiers are close.
- One can put a bound on generalization error based on the difference of errors of these classifiers.

Optimal hyperplane and the support Vectors

The optimal hyperplane does not change if we remove any ‘non-support-vector’ from the training data.

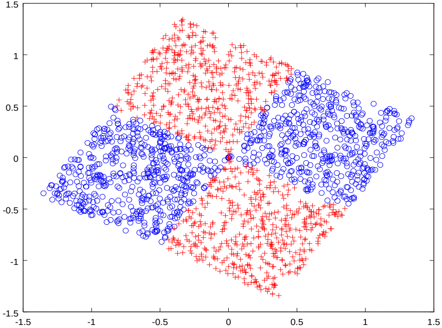


How good is SVM method ?

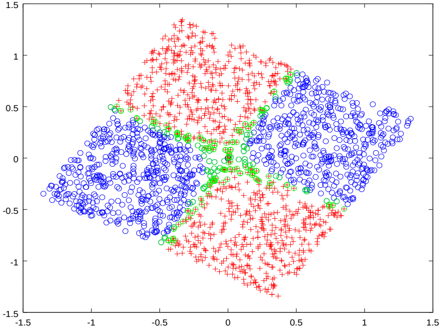
- A very competitive method for tackling many PR problems.
- Learning a nonlinear classifier only involves choosing a Kernel function.
- User needs to make choice of parameters – kernel function and C. Also parametrs of the Opt technique. Bad choices result in 'overfitting'.
- Support vectors are an imporatnt extra information we get.

An example

- Consider the training set for a 2-class problem in $\Re^2$ as below.

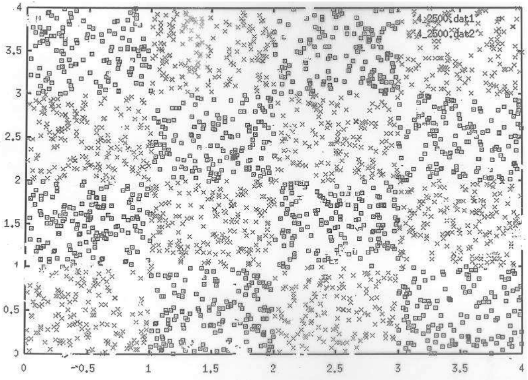


- The support vectors are shown below. (Gaussian Kernel)

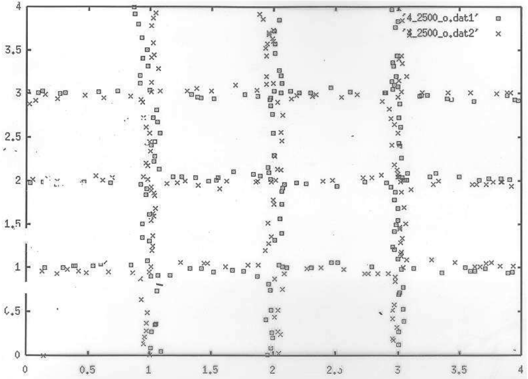


Another example

- More complicated 2-class problem (4×4 checker board).



- The support vectors in this example:



Solving the SVM optimization problem

- So far we have not considered any algorithms for solving for the SVM.
- We have to solve a constrained optimization problem to obtain the Lagrange multipliers and hence the SVM.
- Many specialized algorithms have been proposed for this.

- The optimization problem to be solved is

$$\max_{\boldsymbol{\mu}} \qquad q(\boldsymbol{\mu}) = \sum_{i=1}^{n} \mu_i - \frac{1}{2} \sum_{i,j=1}^{n} \mu_i \mu_j y_i y_j K(X_i, X_j)$$

$$\text{subject to} \qquad 0 \le \mu_i \le C, \;\; i = 1, \ldots, n, \;\; \sum_{i=1}^{n} y_i \mu_i = 0$$

- A quadratic programming (QP) problem with interesting structure.
- Due to the special structure, many efficient algorithms are proposed.

- One interesting idea – Chunking
- We optimize on only a few variables at a time.
- Dimensionality of the optimization problem is controlled.
- We keep randomly choosing the subset of variables.
- Gave rise to the first specialized algorithm for SVM – SVM Light

SMO Algorithm

- Taking chunking to extreme level – what is the smallest set of variables we can optimize on?
- We need to consider at least two variables because there is an equality constraint.
- Sequential Minimal Optimization (SMO) – works on optimizing two variables at a time.
- We can analytically find the optimum with respect to two variables.
- The algorithm (heuristically) decides which two we consider in each iteration.
- A very efficient algorithm
- There are many such algorithms.

Kernel Trick

- We use $\phi : \Re^n \rightarrow \mathcal{H}$ to map pattern vectors into appropriate high dimensional space.
- Kernel fuction allows us to compute innerproducts in $\mathcal{H}$ **implicitly** without using (or even knowing) ϕ.
- Through kernel functions, we learn nonlinear classifiers using 'linear techniques'.
- Algorithms that use only innerproducts (e.g., Fisher discriminant, regression etc) can be implicitly executed in a high dimensional, $\mathcal{H}$, by using a kernel function.
- We can elegantly construct non-linear versions of linear techniques.

Support Vector Regression

- Now we consider the regression problem.
- Given training data

$$\{(X_1,\, y_1), \ldots, (X_n,\, y_n)\}, \;\; X_i \in \Re^m,\; y_i \in \Re,$$

  want to find 'best' function to predict y given X.
- We search in a parameterized class of functions

$$\begin{aligned} g(X,\, W) &= w_1\, \phi_1(X) \,+\, \cdots + \, w_{m'}\, \phi_{m'}(X) \,+\, b \\ &= W^T \Phi(X) \,+\, b, \end{aligned}$$

  where $\phi_i : \Re^m \rightarrow \Re$ are some chosen functions.

- If we choose, $\phi_i(X) = x_i$ (and hence, $m = m'$) then it is the usual linear model.
- We are essentially learning a linear model in terms of $\Phi(X)$.
- We want to formulate the problem so that we can use the Kernel idea.
- Then, by using a kernel function, we never need to compute or even precisely specify the mapping Φ.

Loss function

- In a general regression problem, we need to find W to minimize

$$\sum_i L(y_i,\, g(X_i,\, W))$$

where L is a loss function.

- We consider a special loss function that allows us to use the kernel trick.

ϵ-insensitive loss

- We employ ϵ-insensitive loss function:

$$
\begin{aligned}
L_\epsilon(y_i,\ g(X_i,\ W)) &= 0 && \text{If } |y_i - g(X_i,\ W)| < \epsilon \\
&= |y_i - g(X_i,\ W)| - \epsilon && \text{otherwise}
\end{aligned}
$$

  Here, ϵ is a parameter of the loss function.
- If prediction is within ϵ of true value, there is no loss.
- Using absolute value of error rather than square of error allows for better robustness.
- Also gives us optimization problem with the right structure.
- Empirical risk minimization under the ϵ-insensitive loss function would minimize

$$
\sum_{i=1}^{n} \max\left(|y_i - \Phi(X_i)^T W - b| - \epsilon,\ 0\right)
$$

- We want W, b to minimize

$$\sum_{i=1}^{n} \max\left(|y_i - \Phi(X_i)^T W - b| - \epsilon,\ 0\right)$$

- We can pose the problem as follows.

$$\begin{array}{ll} \min\limits_{W,b,\boldsymbol{\xi},\boldsymbol{\xi'}} & \sum_{i=1}^{n} \xi_i + \sum_{i=1}^{n} \xi_i' \\ \text{subject to} & y_i - W^T\Phi(X_i) - b \leq \epsilon + \xi_i, \quad i = 1, \ldots, n \\ & W^T\Phi(X_i) + b - y_i \leq \epsilon + \xi_i', \quad i = 1, \ldots, n \\ & \xi_i \geq 0,\ \xi_i' \geq 0 \quad i = 1, \ldots, n \end{array}$$

- This does not give a dual with the structure we want.
- So, we reformulate the optimization problem

The Optimization Problem

- Find W, b and ξ_i, ξ_i' to

$$
\begin{array}{ll}
\text{minimize} & \frac{1}{2}W^TW + C\left(\sum_{i=1}^{n}\xi_i + \sum_{i=1}^{n}\xi_i'\right) \\
\text{subject to} & y_i - W^T\Phi(X_i) - b \leq \epsilon + \xi_i, \;\; i = 1, \ldots, n \\
& W^T\Phi(X_i) + b - y_i \leq \epsilon + \xi_i', \;\; i = 1, \ldots, n \\
& \xi_i \geq 0, \; \xi_i' \geq 0 \;\; i = 1, \ldots, n
\end{array}
$$

- We have added the term W^TW in the objective function. This is like model complexity in a regularization context.

- Like earlier, we can form the Lagrangian and then, using Kuhn-Tucker conditions, can get the optimal values of W and b.
- Given that this problem is similar to the earlier one, we would get W^* in terms of the optimal lagrange multipliers as earlier.
- Essentially, the lagrange multipliers corresponding to the inequality constraints on the errors would be the determining factors.
- We can use the same technique as earlier to formulate the dual to solve for the optimal Lagrange multipliers.

The dual

- The dual of this problem is

$$
\begin{aligned}
\max_{\boldsymbol{\alpha},\boldsymbol{\alpha}} \quad & \sum_{i=1}^{n} y_i(\alpha_i - \alpha_i') - \epsilon \sum_{i=1}^{n} (\alpha_i + \alpha_i') \\
& -\frac{1}{2} \sum_{i,j} (\alpha_i - \alpha_i')(\alpha_j - \alpha_j')\Phi(X_i)^T \Phi(X_j) \\
\text{subject to} \quad & \sum_{i=1}^{n} (\alpha_i - \alpha_i') = 0 \\
& 0 \leq \alpha_i,\ \alpha_i' \leq C, \ \ i = 1, \ldots, n
\end{aligned}
$$

- Here α_i and α_i' are the Lagrange multipliers corresponding to the first two inequalities in the primal.

The solution

- We can use the Kuhn-Tucker conditions to derive the final optimal values of W and b as earlier.
- This gives us

$$
\begin{aligned}
W^* &= \sum_{i=1}^{n} (\alpha_i^* - \alpha_i^{*'})\Phi(X_i) \\
b^* &= y_j - \Phi(X_j)^T W^* + \epsilon, \;\; j \; s.t. \; 0 < \alpha_j^* < C/n
\end{aligned}
$$

- We have

$$
\begin{aligned}
W^* &= \sum_{i=1}^{n} (\alpha_i^* - \alpha_i^{*'}) \Phi(X_i) \\
b^* &= y_j - \Phi(X_j)^T W^* + \epsilon, \;\; j \; s.t. \; 0 < \alpha_j^* < C/n
\end{aligned}
$$

- Note that we have $\alpha_i^* \alpha_i^{*'} = 0$. Also, $\alpha_i^*, \; \alpha_i^{*'}$ are zero for examples where error is less than ϵ.
- The final W is a linear combination of some of the examples – the support vectors.
- Note that the dual and the final solution are such that we can use the kernel trick.

- Let $K(X, X') = \Phi(X)^T \Phi(X')$.
- The optimal model learnt is

$$\begin{aligned} g(X,\ W^*) &= \sum_{i=1}^{n} (\alpha_i^* - \alpha_i^{*'}) \phi(X_i)^T \phi(X) + b^* \\ &= \sum_{i=1}^{n} (\alpha_i^* - \alpha_i^{*'}) K(X_i,\ X) + b^* \end{aligned}$$

- As earlier, b^* can also be written in terms of the Kernel function.

Support vector regression

- Once again, the kernel trick allows us to learn non-linear models using a linear method.
- The parameters: C, ϵ and parameters of kernel function.
- The basic idea of SVR can be used in many related problems.

SV regression

- With the ϵ-insensitive loss function, points whose targets are within ϵ of the prediction do not contribute any 'loss'.
- Gives rise to some interesting robustness of the method. It can be proved that local movements of target values of points outside the ϵ-tube do not influence the regression.
- Robustness essentially comes through the support vector representation of the regression.

- In our formulation of the regression problem we did not explain why we added $W^T W$ term in the objective function.
- We are essentially minimizing

$$\frac{1}{2}W^T W \ + \ C \sum_{i=1}^{n} \max\left(|y_i - \Phi(X_i)^T W - b| - \epsilon, \ 0\right)$$

- This is 'regularized risk minimization'.
- Then $W^T W$ is the model complexity term which is intended to favour learning of 'smoother' models.
- Next we explain why $W^T W$ is a good term to capture degree of smoothness in case of linear models.

- Let $f : \Re^m \rightarrow \Re$ be a continuous function.
- Continuity means we can make $|f(X) - f(X')|$ as small as we want by taking $||X - X'||$ sufficiently small.
- There are ways to characterize the 'degree of continuity' of a function.
- We consider one such measure now.

ϵ-Margin of a function

- The ϵ-margin of a function, $f : \Re^n \rightarrow \Re$ is

$$m_\epsilon(f) = \inf\{||X - X'|| \,:\, |f(X) - f(X')| \geq 2\epsilon\}$$

- The intuitive idea is:
  How small can $||X - X'||$ *be, still keeping* $|f(X) - f(X')|$ *'large'*

- The larger $m_\epsilon(f)$, the smoother is the function.

$$m_\epsilon(f) = \inf\{||X - X'|| \,:\, |f(X) - f(X')| \geq 2\epsilon\}$$

- Obviously, $m_\epsilon(f) = 0$ if f is discontinuous.
- $m_\epsilon(f)$ can be zero even for continuous functions,

  e.g., $f(x) = 1/x$.

- $m_\epsilon(f) > 0$ for all $\epsilon > 0$ iff f is uniformly continuous.
- Higher margin would mean the function is 'slowly varying' and hence is a 'smoother' model.

Linear Models and margin

- Consider regression with linear models. Then,

$$|f(X) - f(X')| = |W^T(X - X')|.$$

- For all X, X' with $|W^T(X - X')| \geq 2\epsilon$, we want the smallest $||X - X'||$
- It would be smallest if $|W^T(X - X')| = 2\epsilon$ and $(X - X')$ is parallel to W.

  That is, $X - X' = \pm\frac{2\epsilon W}{W^T W}$.
- Thus, $m_\epsilon(f) = || \pm \frac{2\epsilon W}{W^T W}|| = \frac{2\epsilon}{||W||}$.
- Thus in our optimization problem adding the term $W^T W$ promotes learning of smoother models.
- As we have seen linear regression models use this as the regularization term.

- The basic idea of kernel functions, as we saw in SVM, has been extended in many ways.
- There have been many extensions of the basic SVM method also.
- Some of them are essentially formulations of approximate solutions to make the algorithm more efficient.
- Some of them are reformulations to add additional features to the SVM method.
- We consider a couple of simple examples of such extensions.

- Suppose the optimization problem is changed to

$$\begin{aligned} \min_{W,b,\boldsymbol{\xi}} \quad & \frac{1}{2}W^TW \ + \ b^2 \ + \ C\sum_{i=1}^{n} \xi_i \\ \text{subject to} \quad & y_i(W^TX_i + b) \geq 1 - \xi_i, \ \ i = 1, \ldots, n \\ & \xi_i \geq 0, \ \ i = 1, \ldots, n \end{aligned}$$

- We have added the b^2 term to the objective function. The main reason is that it simplifies the dual.

- The dual turns out to be

$$\begin{aligned} \max_{\boldsymbol{\mu}} \quad & \sum_{i=1}^{n} \mu_i - \frac{1}{2} \sum_{i,j=1}^{n} \mu_i \mu_j y_i y_j K(X_i, X_j) \\ & \qquad - \frac{1}{2} \sum_{i,j=1}^{n} \mu_i \mu_j y_i y_j \\ \text{subject to} \quad & 0 \le \mu_i \le C, \quad i = 1, \ldots, n, \end{aligned}$$

- The equality constraint is absent.
  Only bound constraints on variables.
- Allows for efficient optimization.
  (Successive overrelaxation).

- Next, we consider a reformulation of SVM optimization problem, known as ν-SVM.
- Recall that the primal problem for SVM with slack variables is

$$\begin{array}{ll} \min\limits_{W,b,\boldsymbol{\xi}} & \frac{1}{2}W^TW \;+\; C\sum\limits_{i=1}^{n}\,\xi_i \\ \text{subject to} & y_i(W^T\phi(X_i)+b) \geq 1-\xi_i, \;\; i=1,\ldots,n \\ & \xi_i \geq 0, \;\; i=1,\ldots,n \end{array}$$

ν-SVM

- In the SVM formulation with slack variables, we do not know how to choose C.
- Consider a changed optimization problem

$$\begin{aligned} \min_{W,b,\boldsymbol{\xi},\rho} \quad & \frac{1}{2}W^TW - \nu\rho + \frac{1}{n}\sum \xi_i \\ \text{subject to} \quad & y_i[W^T\phi(X_i) + b] \geq \rho - \xi_i \\ & \xi_i \geq 0. \end{aligned}$$

  where ν is a user-chosen constant.
- Note that $W, b, \rho, \xi_i = 0$ is a feasible solution.
- We do not need $\rho \geq 0$ constraint.

- The Lagrangian for this problem is

$$
\begin{aligned}
L(W, b, \boldsymbol{\xi}, \rho, \boldsymbol{\eta}, \boldsymbol{\mu}) &= \frac{1}{2} W^T W - \nu\rho + \frac{1}{n} \sum_{i=1}^{n} \xi_i \\
&\quad - \sum_{i=1}^{n} \eta_i \xi_i + \sum_{i=1}^{n} \mu_i \left(\rho - \xi_i - y_i [W^T \phi(X_i) + b] \right)
\end{aligned}
$$

- The μ_i are the Lagrange multipliers for the separability constraints and η_i are the Lagrange multipliers for the constraints $\xi_i \geq 0$.

The Kuhn-Tucker conditions give us

- $\nabla_W L = 0 \Rightarrow W = \sum_i \ \mu_i y_i \phi(X_i)$
- $\frac{\partial L}{\partial b} = 0 \Rightarrow \sum \mu_i y_i = 0$
- $\frac{\partial L}{\partial \xi_i} = 0 \Rightarrow \mu_i + \eta_i = \frac{1}{n}, \ \forall i$
- $\frac{\partial L}{\partial \rho} = 0 \Rightarrow \sum \mu_i = \nu$
- $\rho - \xi_i - y_i(W^T \phi(X_i) + b) \leq 0; \ \ \xi_i \geq 0; \ \forall i$
- $\mu_i \geq 0; \ \eta_i \geq 0, \ \forall i$
- $\mu_i(\rho - \xi_i - y_i(W^T \phi(X_i) + b)) = 0; \ \ \eta_i \xi_i = 0, \ \forall i$

- Suppose $\xi_i > 0$ for some i. Then we have $\eta_i = 0$ and hence $\mu_i = \frac{1}{n}$. Hence

$$\begin{aligned}
\nu &= \sum_{i=1}^{n} \mu_i = \sum_{i\,:\,\xi_i>0} \mu_i + \sum_{i\,:\,\xi_i=0} \mu_i \\
&\geq \sum_{i\,:\,\xi_i>0} \mu_i = \frac{|\{i : \xi_i > 0\}|}{n}
\end{aligned}$$

- Hence we have:
  ν is an upper bound on the fraction of 'margin errors'.

- We also have, because $0 \leq \mu_i \leq \frac{1}{n}$,

$$\begin{aligned} \nu &= \sum_{i=1}^{n} \mu_i = \sum_{i\,:\,\mu_i > 0} \mu_i + \sum_{i\,:\,\mu_i = 0} \mu_i \\ &\leq \sum_{i\,:\,\mu_i > 0} \mu_i \leq \frac{|\{i : \mu_i > 0\}|}{n} \end{aligned}$$

- Hence we have:
  ν is a lower bound on the fraction of support vectors.

- In the ν-SVM formulation, the ν is the user chosen constant.
- Unlike the parameter C, the ν has an interesting interpretation.
- It is simultaneously the upperbound on fraction of errors and lower bound on fraction of support vectors.
- If for the chosen ν, the problem has a solution with $\rho > 0$, then the bounds would be met.
- This gives us a good way to choose this 'penalty constant'.

- The dual for the ν-SVM turns out to be

$$\max_{\boldsymbol{\mu}} \qquad q(\boldsymbol{\mu}) = -\frac{1}{2}\sum_{i,j=1}^{n} \mu_i \mu_j y_i y_j K(X_i, X_j)$$

$$\text{subject to} \qquad 0 \le \mu_i \le \frac{1}{n}, \forall i; \;\; \sum_{i=1}^{n} y_i \mu_i = 0; \;\; \sum_{i=1}^{n} \mu_i = \nu$$

- This a simple optimization problem similar to that of 'C-SVM'.
- One can show that if we have a solution for ν-SVM then if we choose $C = 1/\rho n$, we get the same solution with 'C-SVM'.

ν SVR

- This idea can be extended to the regression problem also.
- In support vector regression, we had two user defined constants: ϵ and C.
- The ϵ specifies the 'tolerable error' and it is difficult to know what value to choose for it.
- We can reformulate SVR so that we can optimize on ϵ also.
- This will be very similar to the ν-SVM formulation.

- Recall the optimization problem in SVR:

$$
\begin{array}{ll}
\min\limits_{W,b,\boldsymbol{\xi},\boldsymbol{\xi'}} & \frac{1}{2}W^TW + C\left(\sum_{i=1}^{n} \xi_i + \sum_{i=1}^{n} \xi_i'\right) \\
\text{subject to} & y_i - W^T\Phi(X_i) - b \leq \epsilon + \xi_i, \;\; i = 1, \ldots, n \\
 & W^T\Phi(X_i) + b - y_i \leq \epsilon + \xi_i', \;\; i = 1, \ldots, n \\
 & \xi_i \geq 0, \; \xi_i' \geq 0 \;\; i = 1, \ldots, n
\end{array}
$$

- We change the optimization problem to the following:

$$
\begin{aligned}
\min_{W,b,\epsilon,\boldsymbol{\xi},\boldsymbol{\xi'}} \quad & \frac{1}{2}W^TW + C\left(\nu\epsilon + \frac{1}{n}\sum_{i=1}^{n}(\xi_i + \xi_i')\right) \\
\text{subject to} \quad & y_i - W^T\phi(X_i) - b \leq \epsilon + \xi_i, \;\; i = 1, \ldots, n \\
& W^T\phi(X_i) + b - y_i \leq \epsilon + \xi_i', \;\; i = 1, \ldots, n \\
& \xi_i \geq 0, \; \xi_i' \geq 0 \; \epsilon \geq 0, \; i = 1, \ldots, n
\end{aligned}
$$

  where ν is a user-chosen constant.
- We get similar results as in ν-SVM.

Risk minimization view of SVM

- We posed the support vector regression problem as a (regularized) risk minimization under a special loss function.
- It was then reformulated into an (equivalent) constrained optimization problem.
- In contrast, we formulated the SVM directly as a constrained optimization problem.
- However, it can also be seen to be minimization of (regularized) empirical risk under a special loss function.

- The optimization problem for SVM is

$$\begin{aligned} \min_{W,b,\boldsymbol{\xi}} \quad & \frac{1}{2}W^TW \; + \; C\sum_{i=1}^{n} \xi_i \\ \text{subject to} \quad & y_i(W^TX_i + b) \geq 1 - \xi_i, \;\; i = 1, \ldots, n \\ & \xi_i \geq 0, \;\; i = 1, \ldots, n \end{aligned}$$

- Given any W, b, the ξ_i have to satisfy

$$\xi_i \geq \max(0,\; 1 - y_i(W^TX_i + b))$$

- Since we need to minimize $\sum \xi_i$, we need to take the value above for each ξ_i.

- Hence we can find SVM by solving the following unconstrained optimization problem:

$$\min_{W,b} \;\; \frac{1}{2}W^TW \; + \; C\sum_{i=1}^{n} \max(0, \; 1 - y_i(W^TX_i + b))$$

- The model (or classifier) we are learning is $f(X) = W^TX + b$.
- For this model, we already saw W^TW is a good regularization term.

- Consider the loss function defined by

$$L_{\mathsf{hinge}}(y, f(X)) = \max(0, 1 - yf(X))$$

- Then the optimization problem is same as

$$\min_{W,b} \ \frac{1}{n}\sum_{i=1}^{n} L(y_i, f(X_i)) \ + \ C'\frac{1}{2}W^TW$$

- Thus, our SVM formulation is empirical risk minimization under hinge-loss along with a regularization term.

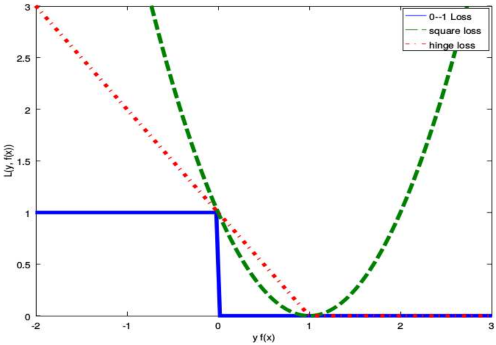
- As we saw earlier, the hinge-loss and square-loss are good convex approximations of the 0–1 loss.
- For 0–1 loss $L(y,\ f(X))$ is one if $yf(X)$ is negative and zero otherwise.
- The squared error loss can be written as

$$L_{\text{square}}(y,\ h(X)) = (1 - yf(X))^2$$

- The hinge loss is given by

$$L_{\text{hinge}}(y, h(X)) = \max(0,\ 1 - yf(X))$$

- We can plot all the functions as follows.



(Here we plot $yf(X)$ on x-axis and $L(y, f(X))$ on y-axis).

- Hinge loss is also called soft-margin loss.
- Supoose we want to minimize, over all f,

$$E[\max(0,\ 1 - yf(X))], \quad y \in \{+1,\ -1\}$$

- Intuitively the best we can do is to make sign of $f(X)$ to be same as sign of the corresponding y.
- Hence, intuitively, the best f is

$$f(X) > 0, \quad \text{if} \ \ P[y = +1|X] > 0.5; \quad \text{else} \ \ f(X) < 0$$

- This is indeed a good classifier.

- In SVM method, there are two important ingradients.
- One is the Kernel function.
- Kernel functions allow us to learn nonlinear models using essentially linear techniques.
- Second is the 'support vector' expansion – the final model is expressed as a ('sparse') linear combination of some of the data vectors.
- Kernels are a good way to capture 'similarity' and are useful in general.
- The support vector expansion is also a general property of Kernel based methods.
- We look at this general view of Kernels next.