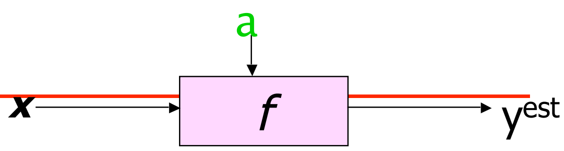
Support Vector Machines

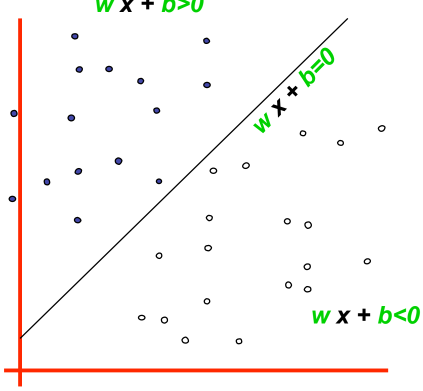
[With notes from Law, Tang, *Aliferis, Tsamardinos*, Tan]

Linear Classifiers



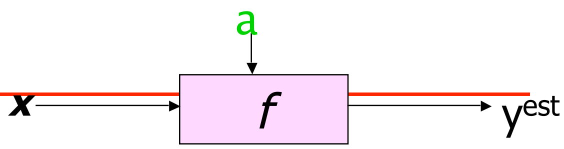
$f(\mathbf{x},\mathbf{w},b) = sign(\mathbf{w}\ \mathbf{x} + b)$

- denotes +1
- denotes -1

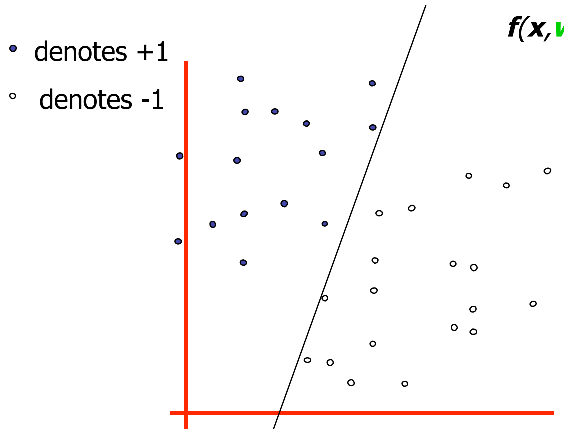


How would you classify this data?

Linear Classifiers

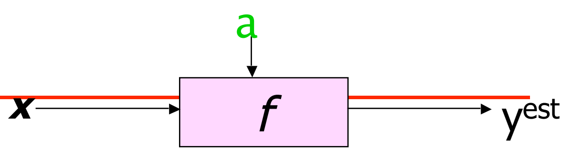


$f(\mathbf{x},\mathbf{w},b) = sign(\mathbf{w}\ \mathbf{x} + b)$

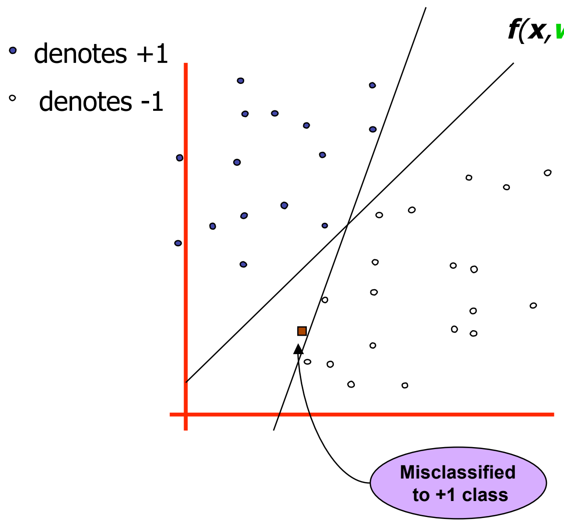


How would you classify this data?

Linear Classifiers

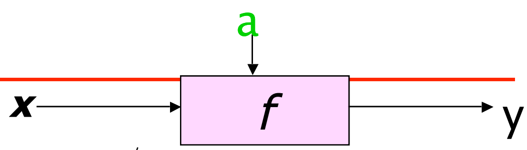


$f(\mathbf{x},\mathbf{w},b) = sign(\mathbf{w}\ \mathbf{x} + b)$



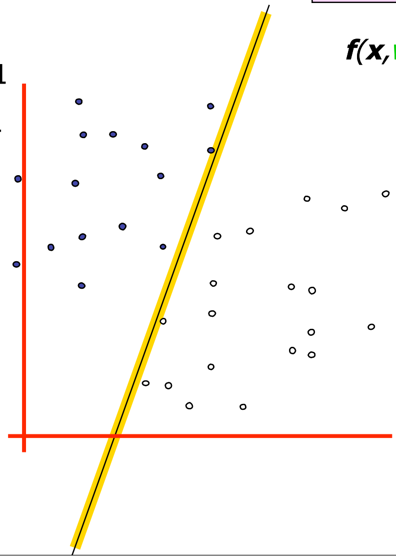
How would you classify this data?

Classifier Margin



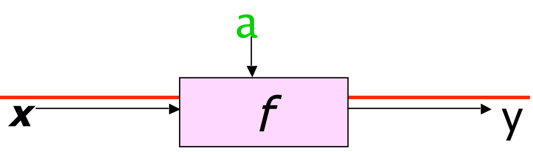
$f(\mathbf{x},\mathbf{w},b) = sign(\mathbf{w}\ \mathbf{x} + b)$

- denotes +1
- denotes -1



Define the margin of a linear classifier as the width that the boundary could be increased by before hitting a datapoint.

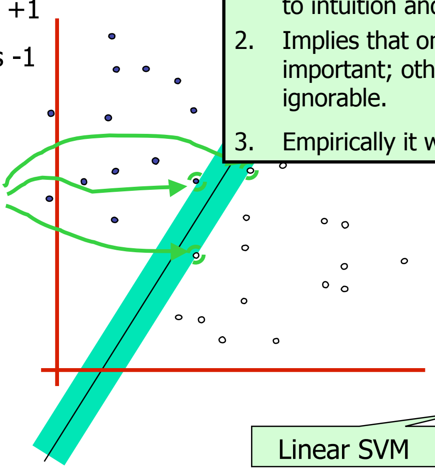
Maximum Margin



- denotes +1
- denotes -1

1. Maximizing the margin is good according to intuition and PAC theory
2. Implies that only support vectors are important; other training examples are ignorable.
3. Empirically it works very very well.

Support Vectors are those datapoints that the margin pushes up against

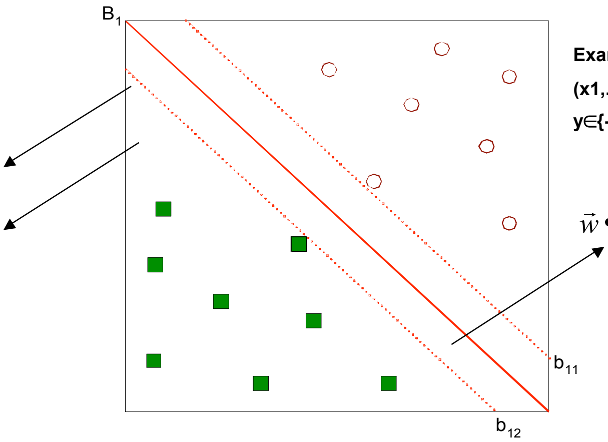


The maximum margin linear classifier is the linear classifier with the, um, maximum margin.

This is the simplest kind of SVM (Called an LSVM)

Linear SVM

Support Vector Machines



$\vec{w} \bullet \vec{x} + b = 0$

$\vec{w} \bullet \vec{x} + b = -1$

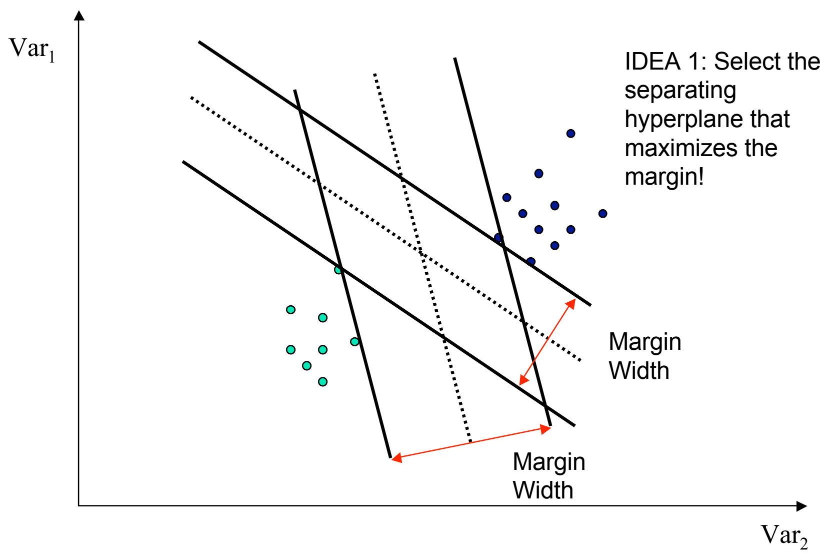
$\vec{w} \bullet \vec{x} + b = +1$

Examples are;
(x1,..,xn,y) with
y∈{-1.1}

$$f(\vec{x}) = \begin{cases} 1 & \text{if } \vec{w} \bullet \vec{x} + b \geq 1 \\ -1 & \text{if } \vec{w} \bullet \vec{x} + b \leq -1 \end{cases}$$

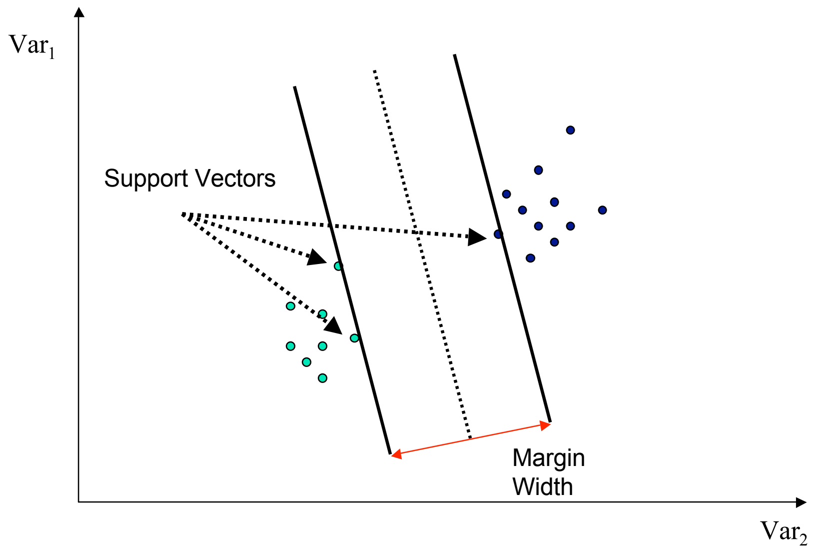
$$\text{Margin} = \frac{2}{\|\vec{w}\|^2}$$

Maximizing the Margin

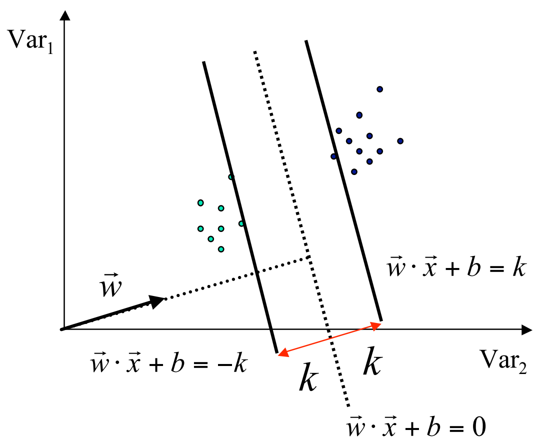


IDEA 1: Select the separating hyperplane that maximizes the margin!

Support Vectors



Setting Up the Optimization Problem



The width of the margin is:

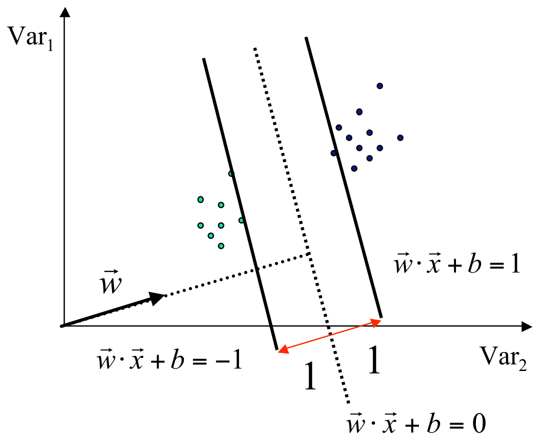
$$\frac{2|k|}{\|w\|}$$

So, the problem is:

$$\max \frac{2|k|}{\|w\|}$$

$$s.t.\ (w\cdot x+b)\geq k,\ \forall x \text{ of class } 1$$
$$(w\cdot x+b)\leq -k,\ \forall x \text{ of class } 2$$

Setting Up the Optimization Problem



There is a scale and unit for data so that *k=1*. Then problem becomes:

$$\max \frac{2}{\|w\|}$$

$$s.t.\ (w \cdot x + b) \geq 1,\ \forall x \text{ of class } 1$$

$$(w \cdot x + b) \leq -1,\ \forall x \text{ of class } 2$$

Finding the Decision Boundary

- Let $\{x_1, \ldots, x_n\}$ be our data set and let y_i Î {1,-1} be the class label of x_i
- The decision boundary should classify all points correctly Þ $y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1, \quad \forall i$
- The decision boundary can be found by solving the following constrained optimization problem

$$\text{Minimize } \frac{1}{2}||\mathbf{w}||^2$$

$$\text{subject to } y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 \qquad \forall i$$

- This is a constrained optimization problem. Solving it requires some new tools
  - Feel free to ignore the following several slides; what is important is the constrained optimization problem above

Recap of Constrained Optimization

- Suppose we want to: minimize f(**x**) subject to g(**x**) = 0
- A necessary condition for $\mathbf{x}_0$ to be a solution:

$$\begin{cases} \frac{\partial}{\partial \mathbf{x}}\Big(f(\mathbf{x}) + \alpha g(\mathbf{x})\Big)\Big|_{\mathbf{x}=\mathbf{x}_0} = \mathbf{0} \\ g(\mathbf{x}) = 0 \end{cases}$$

- a: the Lagrange multiplier
- For multiple constraints $g_i(\mathbf{x}) = 0$, i=1, …, m, we need a Lagrange multiplier a_i for each of the constraints

$$\begin{cases} \frac{\partial}{\partial \mathbf{x}}\Big(f(\mathbf{x}) + \sum_{i=1}^{n} \alpha_i g_i(\mathbf{x})\Big)\Big|_{\mathbf{x}=\mathbf{x}_0} = \mathbf{0} \\ g_i(\mathbf{x}) = 0 \qquad \text{for } i = 1, \ldots, m \end{cases}$$

Recap of Constrained Optimization

- The case for inequality constraint $g_i(\mathbf{x}) \leq 0$ is similar, except that the Lagrange multiplier a_i should be positive
- If $\mathbf{x}_0$ is a solution to the constrained optimization problem

$$\min_{\mathbf{x}} f(\mathbf{x}) \quad \text{subject to} \quad g_i(\mathbf{x}) \leq 0 \quad \text{for } i = 1, \ldots, m$$

- There must exist a_i^3 0 for i=1, …, m such that $\mathbf{x}_0$ satisfy

$$\begin{cases} \frac{\partial}{\partial \mathbf{x}}\Big(f(\mathbf{x}) + \sum_{i} \alpha_i g_i(\mathbf{x})\Big)\Big|_{\mathbf{x}=jx_0} = \mathbf{0} \\ g_i(\mathbf{x}) \leq 0 \qquad \text{for } i = 1, \ldots, m \end{cases}$$

- The function $f(\mathbf{x}) + \sum_i \alpha_i g_i(\mathbf{x})$ is also known as the Lagrangrian; we want to set its gradient to **0**

Back to the Original Problem

$$\text{Minimize } \frac{1}{2}||\mathbf{w}||^2$$

$$\text{subject to } 1 - y_i(\mathbf{w}^T\mathbf{x}_i + b) \leq 0 \qquad \text{for } i = 1, \ldots, n$$

- The Lagrangian is

$$\mathcal{L} = \frac{1}{2}\mathbf{w}^T\mathbf{w} + \sum_{i=1}^{n} \alpha_i \left(1 - y_i(\mathbf{w}^T\mathbf{x}_i + b)\right)$$

  - Note that $||\mathbf{w}||^2 = \mathbf{w}^T\mathbf{w}$
- Setting the gradient of $\mathcal{L}$ w.r.t. **w** and b to zero, we have

$$\mathbf{w} + \sum_{i=1}^{n} \alpha_i(-y_i)\mathbf{x}_i = \mathbf{0} \quad \Rightarrow \quad \mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i$$

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

The Dual Problem

- If we substitute $\mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i$ to $\mathcal{L}$, we have

$$\begin{aligned}
\mathcal{L} &= \frac{1}{2}\sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i^T \sum_{j=1}^{n} \alpha_j y_j \mathbf{x}_j + \sum_{i=1}^{n} \alpha_i \left(1 - y_i(\sum_{j=1}^{n} \alpha_j y_j \mathbf{x}_j^T \mathbf{x}_i + b)\right) \\
&= \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j + \sum_{i=1}^{n} \alpha_i - \sum_{i=1}^{n} \alpha_i y_i \sum_{j=1}^{n} \alpha_j y_j \mathbf{x}_j^T \mathbf{x}_i - b\sum_{i=1}^{n} \alpha_i y_i \\
&= -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j + \sum_{i=1}^{n} \alpha_i
\end{aligned}$$

- Note that $\sum_{i=1}^{n} \alpha_i y_i = 0$
- This is a function of a_i only

The Dual Problem

- The new objective function is in terms of a_i only
- It is known as the dual problem: if we know **w**, we know all a_i; if we know all a_i, we know **w**
- The original problem is known as the primal problem
- The objective function of the dual problem needs to be maximized!
- The dual problem is therefore:

$$\text{max.} \quad W(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1,j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$$

$$\text{subject to } \alpha_i \geq 0, \quad \sum_{i=1}^{n} \alpha_i y_i = 0$$

Properties of a_i when we introduce the Lagrange multipliers

The result when we differentiate the original Lagrangian w.r.t. b

The Dual Problem

$$\text{max.} \quad W(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1,j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$$

$$\text{subject to } \alpha_i \geq 0, \sum_{i=1}^{n} \alpha_i y_i = 0$$

- This is a quadratic programming (QP) problem
  - A global maximum of a_i can always be found
- **w** can be recovered by $\mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i$

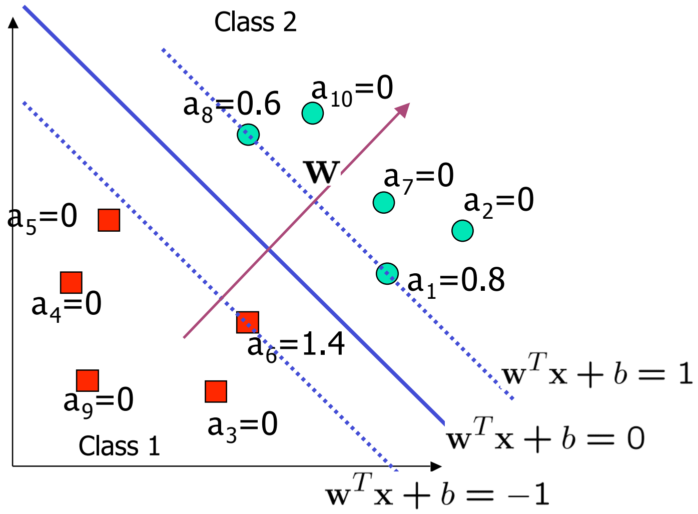
Characteristics of the Solution

- Many of the a_i are zero
  - **w** is a linear combination of a small number of data points
  - This "sparse" representation can be viewed as data compression as in the construction of knn classifier
- $\mathbf{x}_i$ with non-zero a_i are called support vectors (SV)
  - The decision boundary is determined only by the SV
  - Let t_j (j=1, ..., s) be the indices of the s support vectors. We can write $\mathbf{w} = \sum_{j=1}^{s} \alpha_{t_j} y_{t_j} \mathbf{x}_{t_j}$
- For testing with a new data **z**
  - Compute $\mathbf{w}^T\mathbf{z} + b = \sum_{j=1}^{s} \alpha_{t_j} y_{t_j} (\mathbf{x}_{t_j}^T \mathbf{z}) + b$ and classify **z** as class 1 if the sum is positive, and class 2 otherwise
  - Note: **w** need not be formed explicitly

The Quadratic Programming Problem

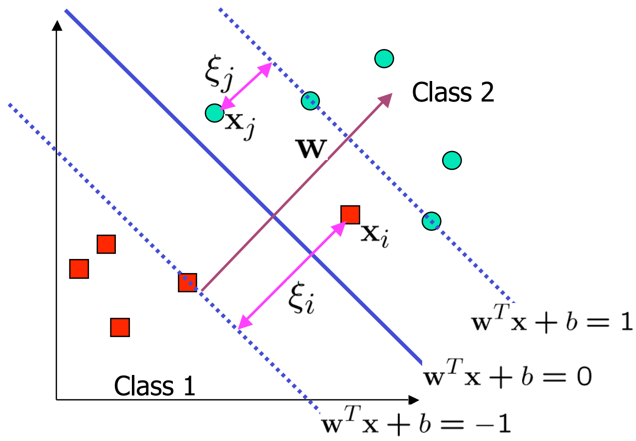
- Many approaches have been proposed
  - Loqo, cplex, etc. (see http://www.numerical.rl.ac.uk/qp/qp.html)
- Most are "interior-point" methods
  - Start with an initial solution that can violate the constraints
  - Improve this solution by optimizing the objective function and/or reducing the amount of constraint violation
- For SVM, sequential minimal optimization (SMO) seems to be the most popular
  - A QP with two variables is trivial to solve
  - Each iteration of SMO picks a pair of (a_i,a_j) and solve the QP with these two variables; repeat until convergence
- In practice, we can just regard the QP solver as a "black-box" without bothering how it works

A Geometrical Interpretation

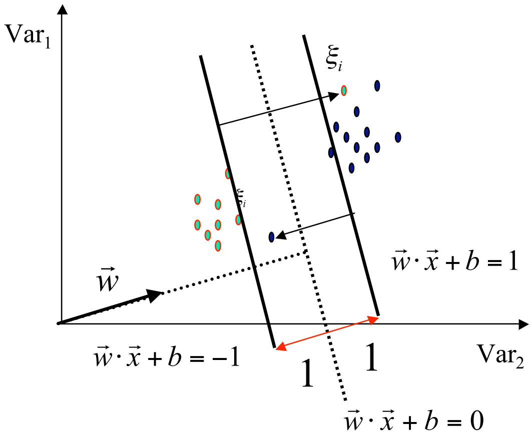


Non-linearly Separable Problems

- We allow "error" x_i in classification; it is based on the output of the discriminant function $\mathbf{w}^T\mathbf{x}$+b
- x_i approximates the number of misclassified samples



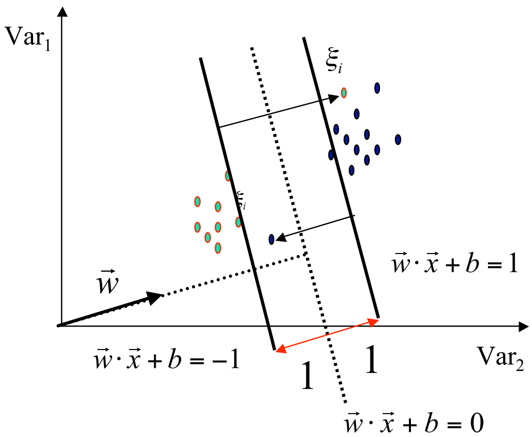
Non-Linearly Separable Data



Introduce slack ξ_i variables

Allow some instances to fall within the margin, but penalize them

Formulating the Optimization Problem



Constraint becomes :

$$y_i(w\cdot x_i+b)\geq 1-\xi_i,\ \forall x_i$$

$$\xi_i \geq 0$$

Objective function penalizes for misclassified instances and those within the margin

$$\min \frac{1}{2}\|w\|^2 + C\sum_i \xi_i$$

C trades-off margin width and misclassifications

Soft Margin Hyperplane

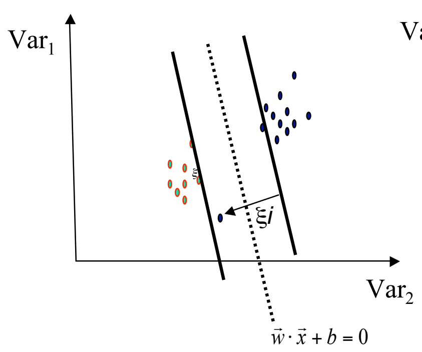
- If we minimize å$_i$x$_i$, x$_i$ can be computed by

$$\begin{cases} \mathbf{w}^T\mathbf{x}_i + b \geq 1 - \xi_i & y_i = 1 \\ \mathbf{w}^T\mathbf{x}_i + b \leq -1 + \xi_i & y_i = -1 \\ \xi_i \geq 0 & \forall i \end{cases}$$

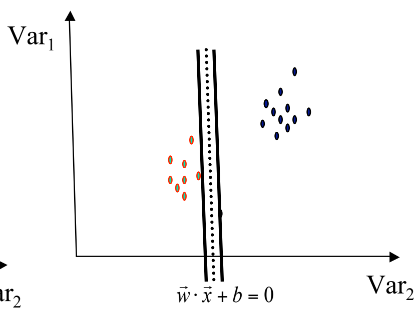
  - x$_i$ are "slack variables" in optimization
  - Note that x$_i$=0 if there is no error for $\mathbf{x}_i$
  - x$_i$ is an upper bound of the number of errors
- We want to minimize $\frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{n}\xi_i$
  - C : tradeoff parameter between error and margin
- The optimization problem becomes

$$\text{Minimize } \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{n}\xi_i$$
$$\text{subject to } y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 - \xi_i, \quad \xi_i \geq 0$$

Robustness of Soft vs Hard Margin SVMs



Soft Margin SVN

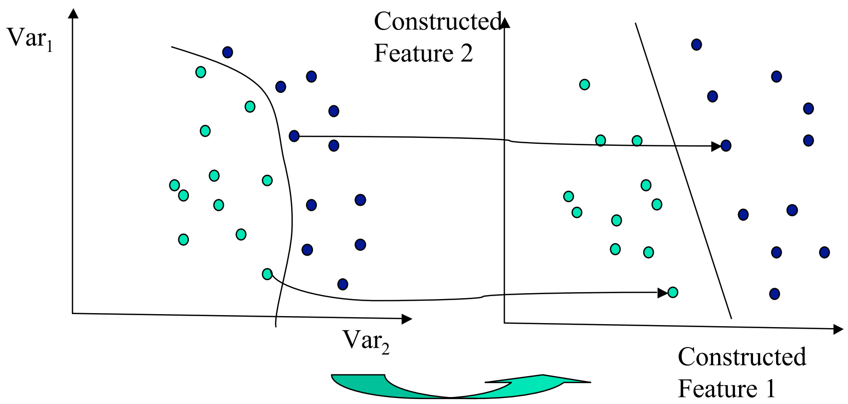


Hard Margin SVN

Soft vs Hard Margin SVM

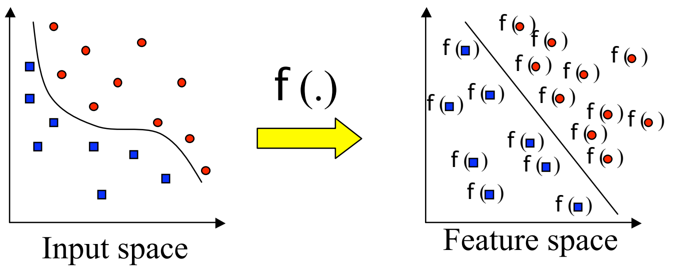
- Soft-Margin always have a solution
- Soft-Margin is more robust to outliers
  - Smoother surfaces (in the non-linear case)
- Hard-Margin does not require to guess the cost parameter (requires no parameters at all)

Linear Classifiers in High-Dimensional Spaces



Find function $\Phi(x)$ to map to a different space

Transforming the Data



Note: feature space is of higher dimension than the input space in practice

- Computation in the feature space can be costly because it is high dimensional
  - The feature space is typically infinite-dimensional!
- The kernel trick comes to rescue

The Dual of the SVM Formulation

- Original SVM formulation
  - *n* inequality constraints
  - *n positivity constraints*
  - *n* number of ξ variables

$$\min_{w,b} \frac{1}{2}\|w\|^2 + C\sum_i \xi_i$$

$$s.t.\ \ y_i(w \cdot \Phi(x) + b) \geq 1 - \xi_i, \forall x_i$$
$$\xi_i \geq 0$$

- The (Wolfe) dual of this problem
  - one equality constraint
  - *n* positivity constraints
  - *n* number of α variables (Lagrange multipliers)
  - Objective function more complicated

$$\min_{a_i} \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j (\Phi(x_i) \cdot \Phi(x_j)) - \sum_i \alpha_i$$

$$s.t.\ \ C \geq \alpha_i \geq 0, \forall x_i$$
$$\sum_i \alpha_i y_i = 0$$

- NOTICE: Data only appear as $\Phi(x_i) \cdot \Phi(x_j)$

The Kernel Trick

- $\Phi(x_i) \cdot \Phi(x_j)$: means, map data into new space, then take the inner product of the new vectors
- We can find a function such that: $K(x_i \cdot x_j) = \Phi(x_i) \cdot \Phi(x_j)$, i.e., the image of the inner product of the data is the inner product of the images of the data
- Then, we do not need to explicitly map the data into the high-dimensional space to solve the optimization problem (for training)
- How do we classify without explicitly mapping the new instances? Turns out

$$\mathrm{sgn}(wx+b) = \mathrm{sgn}(\sum_i \alpha_i y_i K(x_i, x) + b)$$

$$\text{where } b \text{ solves } \alpha_j (y_j \sum_i \alpha_i y_i K(x_i, x_j) + b - 1) = 0,$$

$$\text{for any } j \text{ with } \alpha_j \neq 0$$

The Kernel Trick

- Recall the SVM optimization problem

$$\text{max. } W(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1,j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$$

$$\text{subject to } C \geq \alpha_i \geq 0, \sum_{i=1}^{n} \alpha_i y_i = 0$$

- The data points only appear as inner product
- As long as we can calculate the inner product in the feature space, we do not need the mapping explicitly
- Many common geometric operations (angles, distances) can be expressed by inner products
- Define the kernel function K by

$$K(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)$$

An Example for f (.) and K(.,.)

- Suppose f (.) is given as follows

$$\phi(\left[\begin{smallmatrix} x_1 \\ x_2 \end{smallmatrix}\right]) = (1, \sqrt{2}x_1, \sqrt{2}x_2, x_1^2, x_2^2, \sqrt{2}x_1x_2)$$

- An inner product in the feature space is

$$\langle \phi(\left[\begin{smallmatrix} x_1 \\ x_2 \end{smallmatrix}\right]), \phi(\left[\begin{smallmatrix} y_1 \\ y_2 \end{smallmatrix}\right]) \rangle = (1 + x_1y_1 + x_2y_2)^2$$

- So, if we define the kernel function as follows, there is no need to carry out f (.) explicitly

$$K(\mathrm{x}, \mathrm{y}) = (1 + x_1y_1 + x_2y_2)^2$$

- This use of kernel function to avoid carrying out f (.) explicitly is known as the kernel trick

Kernel Functions

- In practical use of SVM, the user specifies the kernel function; the transformation f (.) is not explicitly stated
- Given a kernel function $K(\mathbf{x}_i, \mathbf{x}_j)$, the transformation f (.) is given by its eigenfunctions (a concept in functional analysis)
  - Eigenfunctions can be difficult to construct explicitly
  - This is why people only specify the kernel function without worrying about the exact transformation
- Another view: kernel function, being an inner product, is really a similarity measure between the objects

Examples of Kernel Functions

- Polynomial kernel with degree *d*

$$K(\mathbf{x},\mathbf{y}) = (\mathbf{x}^T\mathbf{y}+1)^d$$

- Radial basis function kernel with width s

$$K(\mathbf{x},\mathbf{y}) = \exp(-||\mathbf{x}-\mathbf{y}||^2/(2\sigma^2))$$

  - Closely related to radial basis function neural networks
  - The feature space is infinite-dimensional
- Sigmoid with parameter k and q

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

  - It does not satisfy the Mercer condition on all k and q

Modification Due to Kernel Function

- Change all inner products to kernel functions
- For training,

Original

$$\text{max. } W(\boldsymbol{\alpha}) = \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i=1,j=1}^{n}\alpha_i\alpha_j y_i y_j \mathbf{x}_i^T\mathbf{x}_j$$

$$\text{subject to } C \geq \alpha_i \geq 0, \sum_{i=1}^{n}\alpha_i y_i = 0$$

With kernel function

$$\text{max. } W(\boldsymbol{\alpha}) = \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i=1,j=1}^{n}\alpha_i\alpha_j y_i y_j K(\mathbf{x}_i,\mathbf{x}_j)$$

$$\text{subject to } C \geq \alpha_i \geq 0, \sum_{i=1}^{n}\alpha_i y_i = 0$$

Modification Due to Kernel Function

- For testing, the new data **z** is classified as class 1 if *f* ^{3} 0, and as class 2 if *f* <0

Original

$$\mathbf{w} = \sum_{j=1}^{s} \alpha_{t_j} y_{t_j} \mathbf{x}_{t_j}$$

$$f = \mathbf{w}^T \mathbf{z} + b = \sum_{j=1}^{s} \alpha_{t_j} y_{t_j} \mathbf{x}_{t_j}^T \mathbf{z} + b$$

With kernel function

$$\mathbf{w} = \sum_{j=1}^{s} \alpha_{t_j} y_{t_j} \phi(\mathbf{x}_{t_j})$$

$$f = \langle \mathbf{w}, \phi(\mathbf{z}) \rangle + b = \sum_{j=1}^{s} \alpha_{t_j} y_{t_j} K(\mathbf{x}_{t_j}, \mathbf{z}) + b$$

Example

- Suppose we have 5 1D data points
  - x_1=1, x_2=2, x_3=4, x_4=5, x_5=6, with 1, 2, 6 as class 1 and 4, 5 as class 2 $\Rightarrow$ y_1=1, y_2=1, y_3=-1, y_4=-1, y_5=1
- We use the polynomial kernel of degree 2
  - K(x,y) = $(xy+1)^2$
  - C is set to 100
- We first find a_i (*i*=1, …, 5) by

$$\text{max.} \sum_{i=1}^{5} \alpha_i - \frac{1}{2} \sum_{i=1}^{5} \sum_{j=1}^{5} \alpha_i \alpha_j y_i y_j (x_i x_j + 1)^2$$

$$\text{subject to } 100 \geq \alpha_i \geq 0, \sum_{i=1}^{5} \alpha_i y_i = 0$$

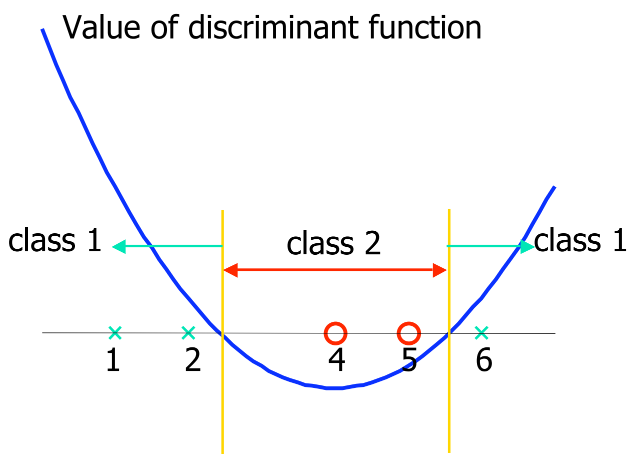
Example

- By using a QP solver, we get
  - $a_1=0$, $a_2=2.5$, $a_3=0$, $a_4=7.333$, $a_5=4.833$
  - Note that the constraints are indeed satisfied
  - The support vectors are $\{x_2=2, x_4=5, x_5=6\}$
- The discriminant function is

$$
\begin{aligned}
& f(z) \\
&= 2.5(1)(2z+1)^2 + 7.333(-1)(5z+1)^2 + \overset{\alpha_5}{4.833}\overset{y_5}{(1)}\overset{K(z,x_5)}{(6z+1)^2} + b \\
&= 0.6667z^2 - 5.333z + b
\end{aligned}
$$

- *b* is recovered by solving f(2)=1 or by f(5)=-1 or by f(6)=1, as x_2 and x_5 lie on th $\phi(\mathbf{w})^T\phi(\mathbf{x})+b=1$ and x_4 lies on t $\phi(\mathbf{w})^T\phi(\mathbf{x})+b=-1$
- All three give b=9 $\Longrightarrow$ $f(z) = 0.6667z^2 - 5.333z + 9$

Example



The Mercer Condition

- Is there a mapping $\Phi(x)$ for any symmetric function $K(x,z)$? No
- The SVM dual formulation requires calculation $K(x_i, x_j)$ for each pair of training instances. The array $G_{ij} = K(x_i, x_j)$ is called the Gram matrix
- There is a feature space $\Phi(x)$ when the Kernel is such that G is always semi-positive definite (Mercer condition)

Why SVM Work?

- The feature space is often very high dimensional. Why don't we have the curse of dimensionality?
- A classifier in a high-dimensional space has many parameters and is hard to estimate
- Vapnik argues that the fundamental problem is not the number of parameters to be estimated. Rather, the problem is about the flexibility of a classifier
- Typically, a classifier with many parameters is very flexible, but there are also exceptions
  - Let $x_i=10^i$ where i ranges from 1 to n. The classifier $y = \text{sign}\big(\sin(\alpha x)\big)$ can classify all x_i correctly for all possible combination of class labels on x_i
  - This 1-parameter classifier is very flexible

Why SVM works?

- Vapnik argues that the flexibility of a classifier should not be characterized by the number of parameters, but by the flexibility (capacity) of a classifier
  - This is formalized by the “VC-dimension” of a classifier
- Consider a linear classifier in two-dimensional space
- If we have three training data points, no matter how those points are labeled, we can classify them perfectly