

Support Vector Machines



Jaya Sil

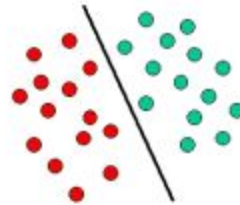
Department of Computer Science and Technology

Support Vector Machines (SVM)

- SVM were introduced by Vladimir Vapnik (Vapnik, 1995).
- The main objective in SVM is to find the hyperplane which separates the d -dimensional data points perfectly into two classes.
- However, since example data is often not linearly separable, SVM's introduce the notion of a “kernel induced feature space” which casts the data points (input space) into a higher dimensional feature space where the data is separable.
- SVM's higher-dimensional space doesn't need to be dealt with directly which eliminates overfitting.

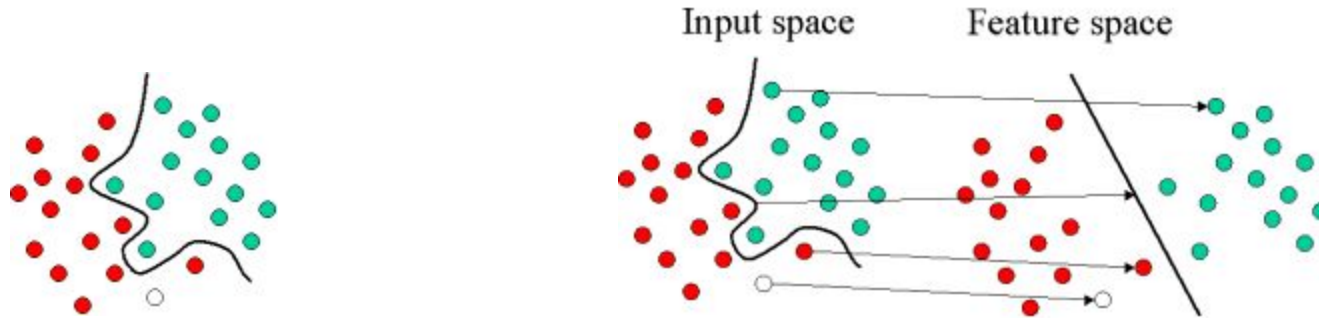
Support Vector Machines - Linear classifier

- Classification tasks are based on drawing separating lines to distinguish between objects of different class labels are known as hyperplane classifiers.
- A decision plane is one that separates between a set of objects having different class labels.



- Any new object falling to the right is labeled, i.e., classified, as GREEN (or classified as RED should it fall to the left of the separating line).
- The objects closest to the hyperplane are called support vectors

Input Space to Feature Space

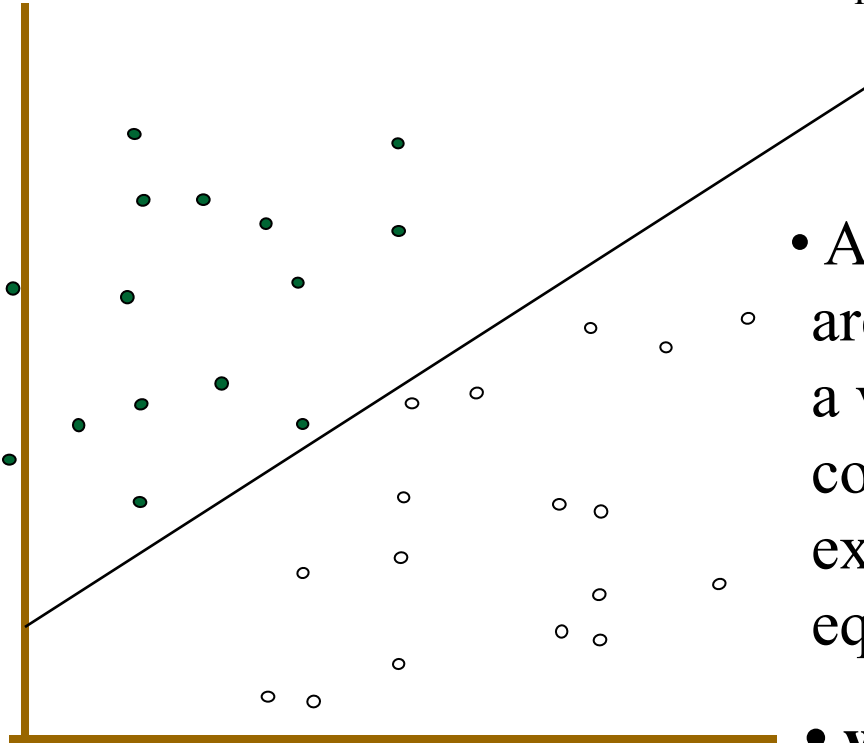


- The original objects are transformed, using a set of mathematical functions, known as kernels.
- Instead of constructing the complex curve, we find an optimal line that can separate the objects.

Linear Classifiers

We are given l training examples $\{x_i, y_i\}$; $i = 1..l$, where each example has d inputs ($x_i \in \mathbf{R}^d$), and a class label with one of two values ($y_i \in \{-1, 1\}$).

- denotes +1
- denotes -1

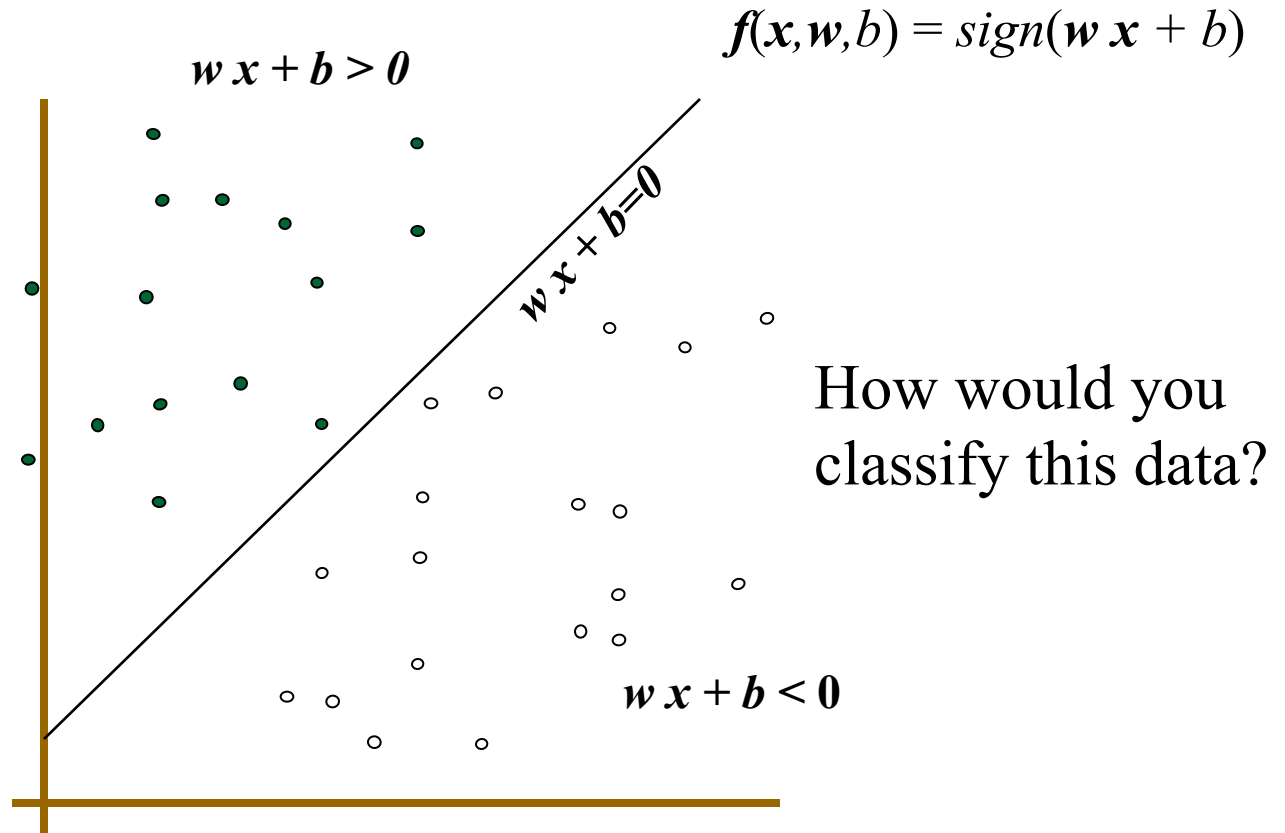


- All hyperplanes in \mathbf{R}^d are parameterized by a vector (\mathbf{w}) and a constant (b), expressed using the equation $\mathbf{w} \cdot \mathbf{x} + b = 0$
- \mathbf{w} is the vector orthogonal to the hyperplane

- Given such a hyperplane (\mathbf{w}, b) that separates the data, using function $f(x) = \text{sign}(\mathbf{w} \cdot \mathbf{x} + b)$

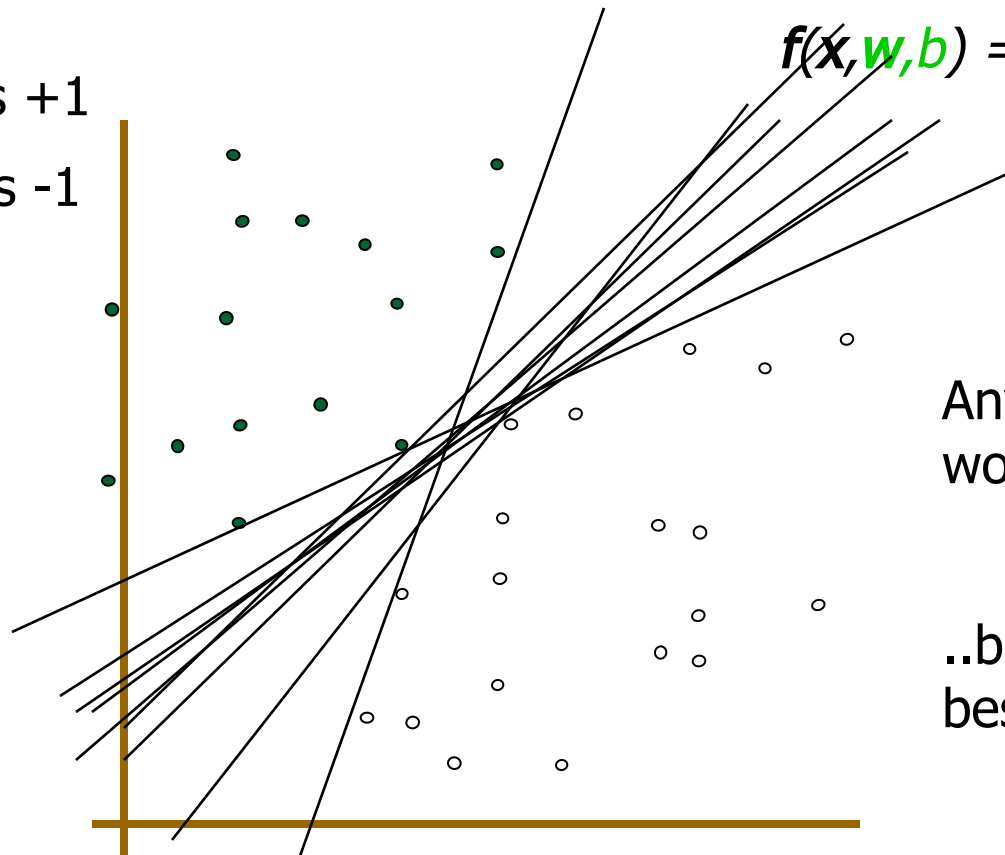
Linear Classifiers

- denotes +1
- denotes -1



Linear Classifiers

- denotes +1
- denotes -1



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

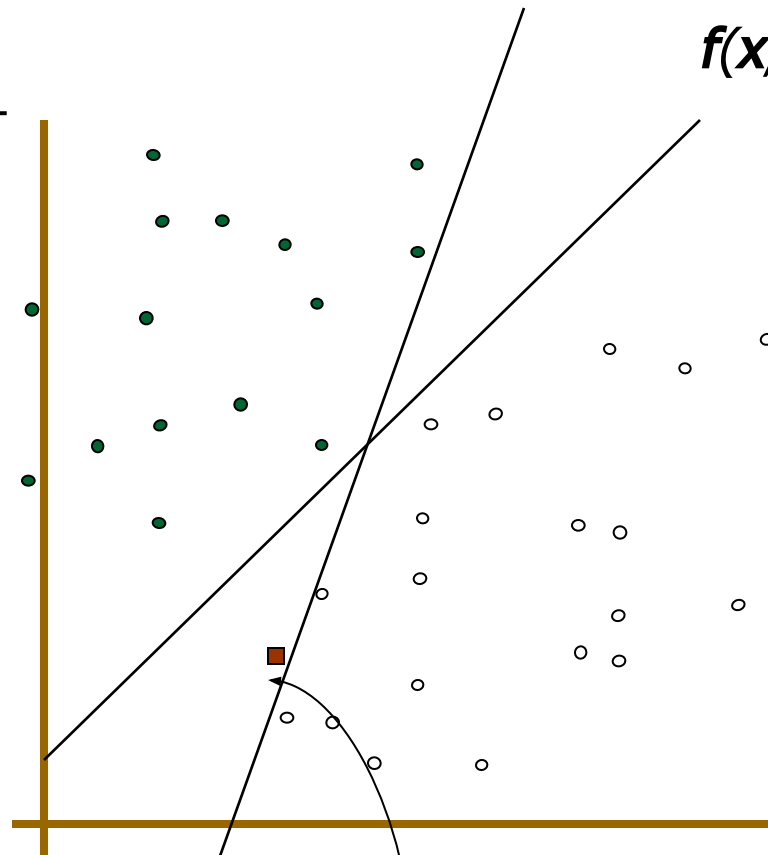
Any of these
would be fine..

..but which is
best?

Linear Classifiers

- denotes +1
- denotes -1

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



How would you classify this data?

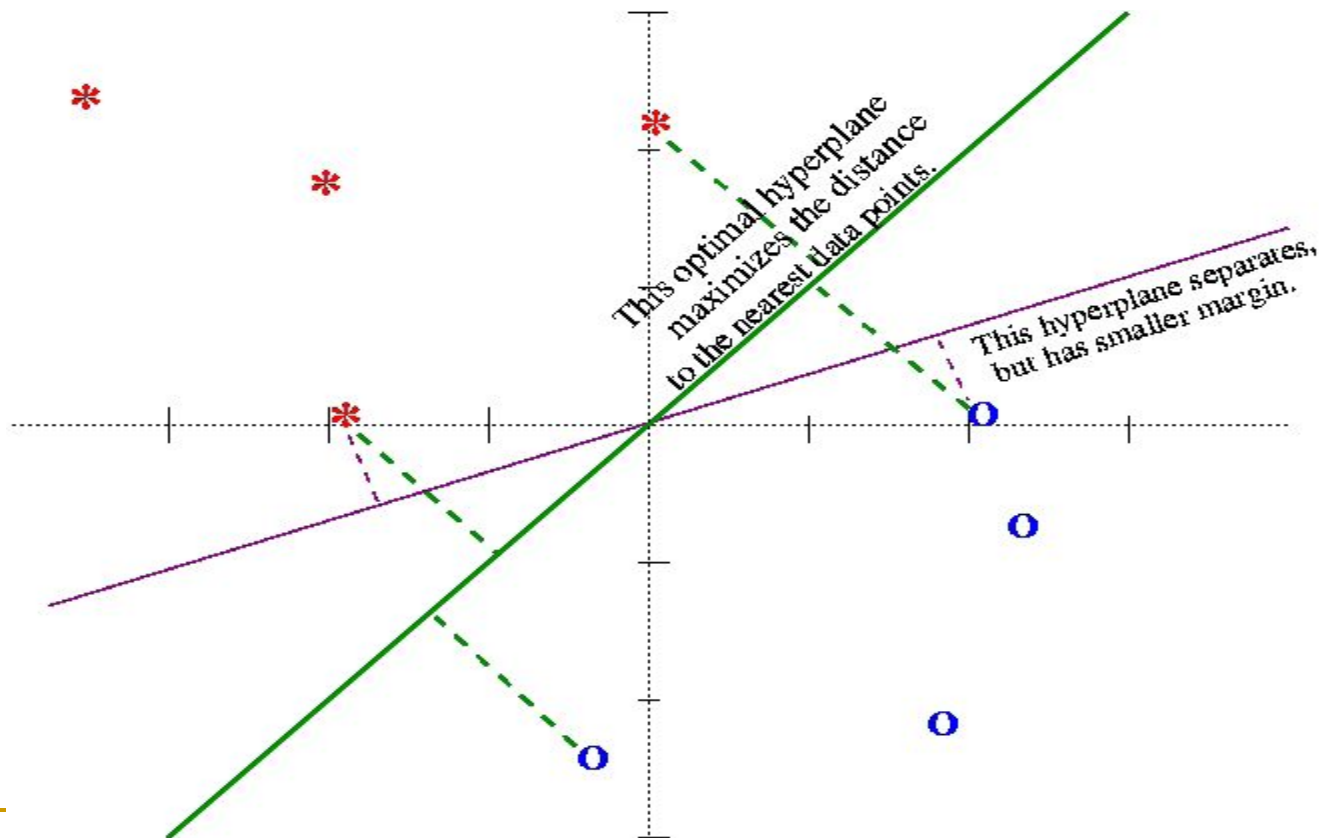
Misclassified
to +1 class

Hyperplane Classifier

- A given hyperplane represented by (\mathbf{w}, b) is equally expressed by all pairs $\{\lambda \mathbf{w}, \lambda b\}$ for $\lambda \in \mathbb{R}^+$.
- We define the hyperplane which separates the data from the hyperplane by a “distance” so that at least one example on both sides has a distance of exactly 1.
- That is, we consider those that satisfy:
- $\mathbf{w} \cdot \mathbf{x}_i + b \geq 1$ when $y_i = +1$
- $\mathbf{w} \cdot \mathbf{x}_i + b \leq -1$ when $y_i = -1$

$$y_i (\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1 \quad \forall i$$

- To obtain the geometric distance from the hyperplane to a data point, we normalize by the magnitude of \mathbf{w} .
 - We want the hyperplane that maximizes the geometric distance to the closest data points.
- $$d((\mathbf{w}, b), \mathbf{x}_i) = [y_i(\mathbf{w} \cdot \mathbf{x}_i + b)] / \|\mathbf{w}\| \geq 1 / \|\mathbf{w}\|$$

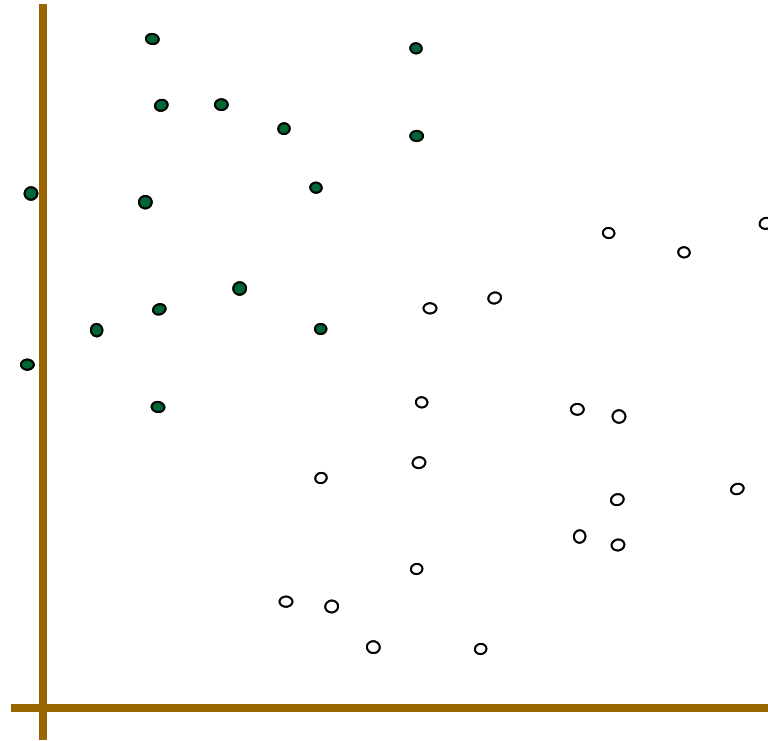


Choosing the hyperplane that maximizes the margin

Classifier Margin

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

- denotes +1
- denotes -1



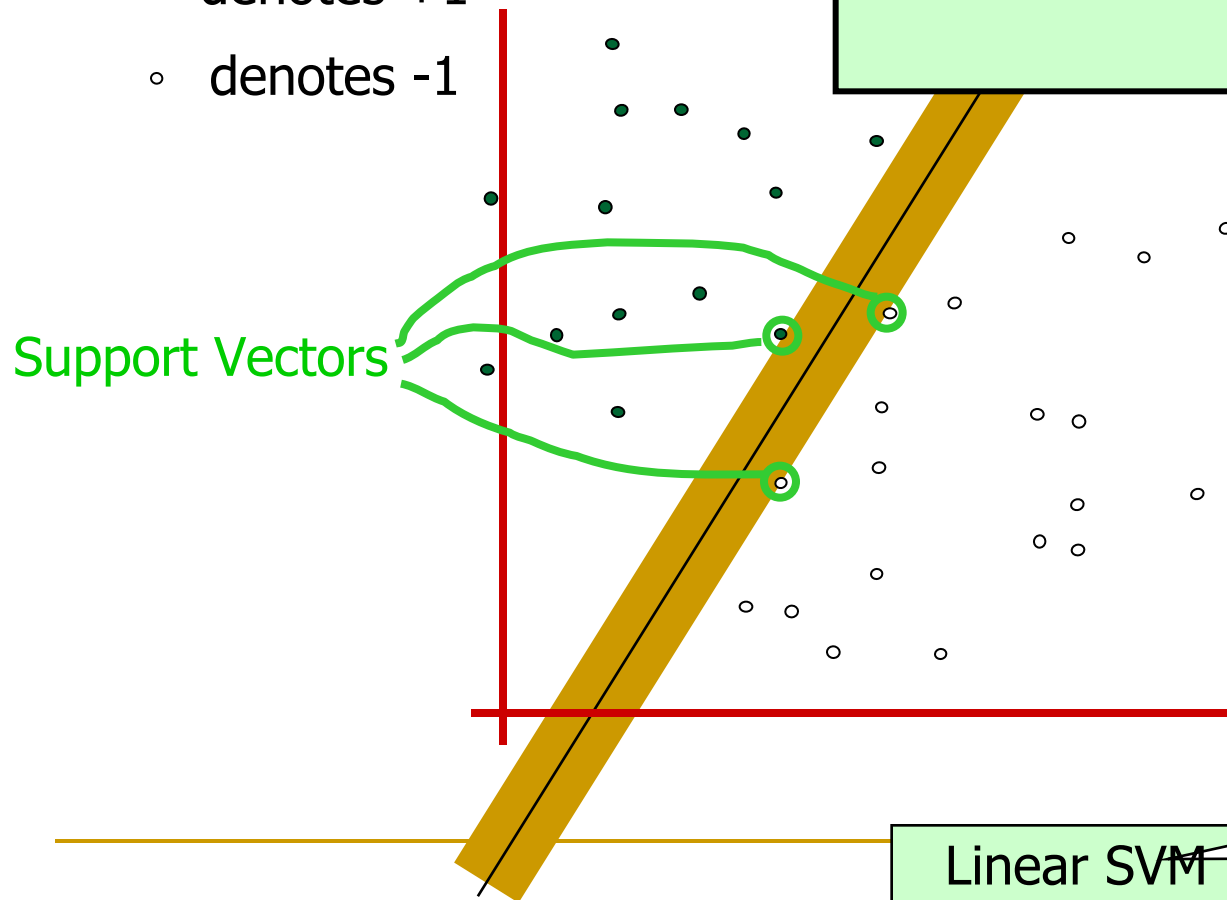
Define the **margin** of a linear classifier as the width that the boundary could be increased by before hitting a datapoint.

Maximum Margin

- denotes +1
- denotes -1

1. Maximizing the margin
2. support vectors are important

Support Vectors

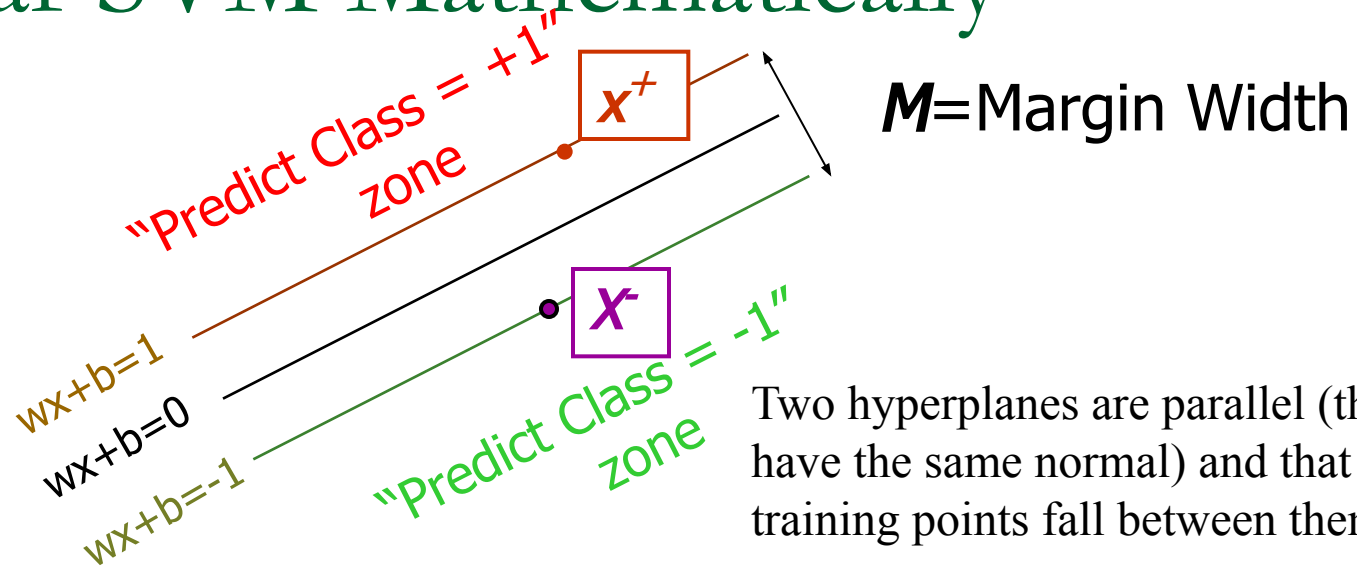


The **maximum margin linear classifier**

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

Linear SVM

Linear SVM Mathematically



Two hyperplanes are parallel (they have the same normal) and that no training points fall between them.

What we know:

- $w \cdot x^+ + b = +1$
- $w \cdot x^- + b = -1$
- $w \cdot (x^+ - x^-) = 2$

$$M = \frac{(x^+ - x^-) \cdot w}{\|w\|} = \frac{2}{\|w\|}$$

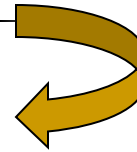
Linear SVM Mathematically

■ Goal: 1) Correctly classify all training data

$$wx_i + b \geq 1 \quad \text{if } y_i = +1$$

$$wx_i + b \leq -1 \quad \text{if } y_i = -1$$

$$y_i(wx_i + b) \geq 1 \quad \text{for all } i$$



2) Maximize the Margin $M = \frac{2}{\|w\|}$ or same as minimize $\frac{1}{2} w^t w$

■ We can formulate a constrained optimization Problem and solve for w and b

■ Minimize $\Phi(w) = \frac{1}{2} w^t w$

subject to $\forall i \quad y_i(wx_i + b) \geq 1$

Lagrange Multipliers

- Consider a problem: $\min_x f(x)$ subject to $h(x) = 0$
- We define the Lagrangian $L(x, \alpha) = f(x) - \alpha h(x)$
- α is called “Lagrange multiplier”
- Solve: $\min_x \max_{\alpha} L(x, \alpha)$ subject to $\alpha \geq 0$

Original Problem:

Find \mathbf{w} and b such that

$\Phi(\mathbf{w}) = \frac{1}{2} \mathbf{w}^T \mathbf{w}$ is minimized;

and for all $i \in \{1, \dots, n\}$: $y_i (\mathbf{w}^T \mathbf{x}_i + b) \geq 1$

Construct the Lagrangian Function for optimization

$$\mathcal{L}(w, b, \alpha) = \frac{1}{2} \|w\|^2 - \sum_{i=1}^m \alpha_i [y^{(i)} (w^T x^{(i)} + b) - 1] \quad \text{S. T. } \alpha_i \geq 0; \forall_i$$

Our goal is to: $\max_{\alpha \geq 0} \min_{w, b} \mathcal{L}(w, b, \alpha)$ OR $\min_{w, b} \max_{\alpha \geq 0} \mathcal{L}(w, b, \alpha)$

$$\frac{\partial}{\partial w} \mathcal{L}(w, b, \alpha) = w - \sum_{i=1}^m \alpha_i y^{(i)} x^{(i)} = 0$$

$$w = \sum_{i=1}^m \alpha_i y^{(i)} x^{(i)}$$

The derivative with respect to b

$$\frac{\partial}{\partial b} \mathcal{L}(w, b, \alpha) = \sum_{i=1}^m \alpha_i y^{(i)} = 0$$

Substituting we get:

$$\begin{aligned} \mathcal{L}(w, b, \alpha) &= \sum_{i=1}^m \alpha_i - \frac{1}{2} \sum_{i,j=1}^m y^{(i)} y^{(j)} \alpha_i \alpha_j (x^{(i)})^T x^{(j)} - b \sum_{i=1}^m \alpha_i y^{(i)} \\ &= \sum_{i=1}^m \alpha_i - \frac{1}{2} \sum_{i,j=1}^m y^{(i)} y^{(j)} \alpha_i \alpha_j (x^{(i)})^T x^{(j)} \end{aligned}$$

$$\max_{\alpha} : \sum_{i=1}^m \alpha_i - \frac{1}{2} \sum_{i,j=1}^m y^{(i)} y^{(j)} \alpha_i \alpha_j (x^{(i)})^T x^{(j)}$$

Subject to $0 \leq \alpha_i \leq C$ for $\forall i = 1 \dots m$ and $\sum_{i=1}^m \alpha_i y^{(i)} = 0$

α is the vector of m non-negative Lagrange multipliers to be determined,
and C is a constant

Optimal hyperplane : $\mathbf{w} = \sum_{i=1}^m \alpha_i y_i \mathbf{x}_i$

- The vector \mathbf{w} is just a linear combination of the training examples.

$$C(\mathbf{w}, b, \alpha) = \frac{1}{2} \mathbf{w} \cdot \mathbf{w} - \sum_i \alpha_i [(\mathbf{w} \cdot \mathbf{x}_i + b) y_i - 1]$$
$$\alpha_i \geq 0, \forall_i$$



$$y_i (\vec{w} \cdot \vec{x}_i + b) = 1 \quad (1)$$

$$y_i y_i (\vec{w} \cdot \vec{x}_i + b) = y_i \quad (2)$$

$$(\vec{w} \cdot \vec{x}_i + b) = y_i \quad (3)$$



$$\mathbf{w} = \sum \alpha_i y_i \mathbf{x}_i$$

$$b = y_k - \mathbf{w}^T \mathbf{x}_k \text{ for any } \mathbf{x}_k \text{ such that } \alpha_k \neq 0$$

$$\begin{aligned} w^T x + b &= \left(\sum_{i=1}^m \alpha_i y^{(i)} x^{(i)} \right)^T x + b \\ &= \sum_{i=1}^m \alpha_i y^{(i)} \langle x^{(i)}, x \rangle + b. \end{aligned}$$

- If we've found the α_i 's, in order to make a prediction, we have to calculate a quantity that depends only on the inner product between x and the points in the training set.

- Each non-zero α_i indicates that corresponding \mathbf{x}_i is a **support vector**.

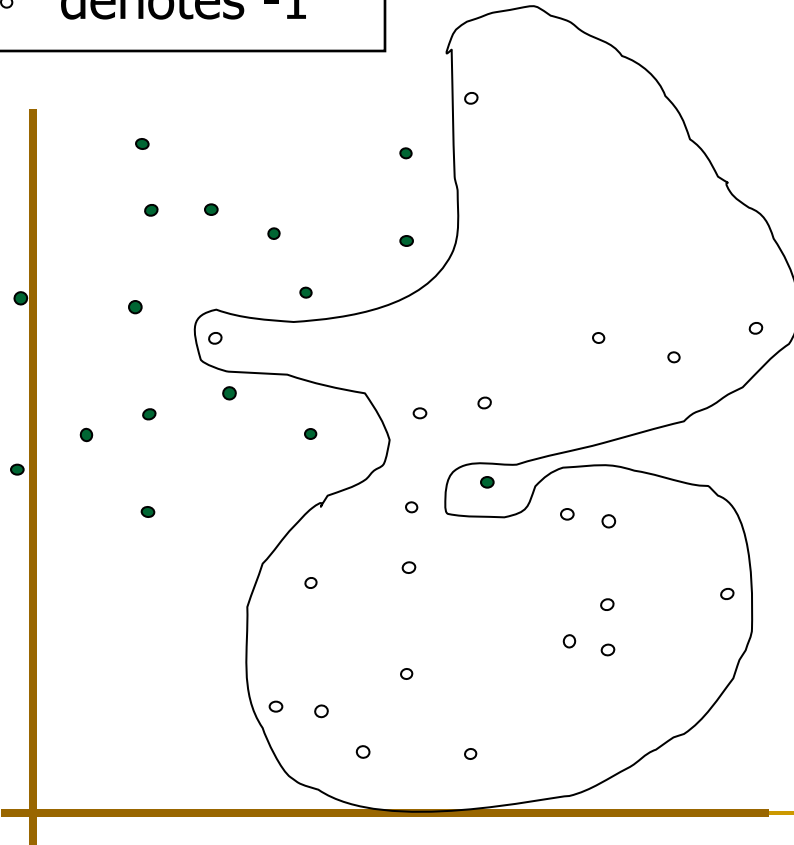
- Then the classifying function will have the form:

$$f(\mathbf{x}) = \sum \alpha_i y_i \mathbf{x}_i^T \mathbf{x} + b$$

- Notice that it relies on an *inner product* between the test point \mathbf{x} and the support vectors \mathbf{x}_i .

Dataset with noise

- denotes +1
- denotes -1

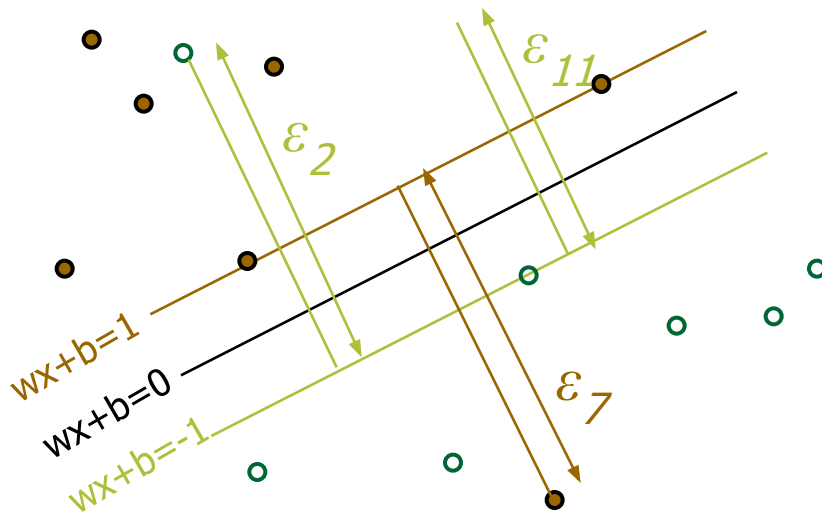


- **Hard Margin:** So far we require all data points be classified correctly.
 - No training error
- What if the training set is noisy?
 - Solution 1: use very powerful kernels

OVERFITTING!

Soft Margin Classification

Slack variables ξ_i can be added to allow misclassification of difficult or noisy examples.



What should our quadratic optimization criterion be?

Minimize

$$\frac{1}{2} \mathbf{w} \cdot \mathbf{w} + \lambda \sum_{k=1}^R \xi_k$$

Hard Margin v.s. Soft Margin

- **The old formulation:**

Find \mathbf{w} and b such that

$$\Phi(\mathbf{w}) = \frac{1}{2} \mathbf{w}^T \mathbf{w} \text{ is minimized and for all } \{(\mathbf{x}_i, y_i)\}$$
$$y_i (\mathbf{w}^T \mathbf{x}_i + b) \geq 1$$

- **The new formulation incorporating slack variables:**

Find \mathbf{w} and b such that

$$\Phi(\mathbf{w}) = \frac{1}{2} \mathbf{w}^T \mathbf{w} + \lambda \sum \xi_i \text{ is minimized and for all } \{(\mathbf{x}_i, y_i)\}$$
$$y_i (\mathbf{w}^T \mathbf{x}_i + b) \geq 1 - \xi_i \text{ and } \xi_i \geq 0 \text{ for all } i$$

- **Parameter λ can be viewed as a way to control overfitting.**

Linear SVMs: Overview

- The classifier is a *separating hyperplane*.
- Most “important” training points are support vectors; they define the hyperplane.
- Quadratic optimization algorithms can identify which training points \mathbf{x}_i are support vectors with non-zero Lagrangian multipliers α_i .
- Both in the dual formulation of the problem and in the solution training points appear only inside dot products:

Find $\alpha_1 \dots \alpha_N$ such that

$Q(\alpha) = \sum \alpha_i - \frac{1}{2} \sum \sum \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$ is maximized and

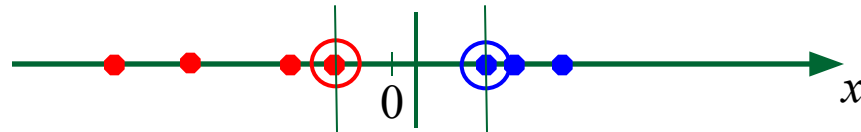
(1) $\sum \alpha_i y_i = 0$

(2) $0 \leq \alpha_i \leq C$ for all α_i

$$f(\mathbf{x}) = \sum \alpha_i y_i \mathbf{x}_i^T \mathbf{x} + b$$

Non-linear SVMs

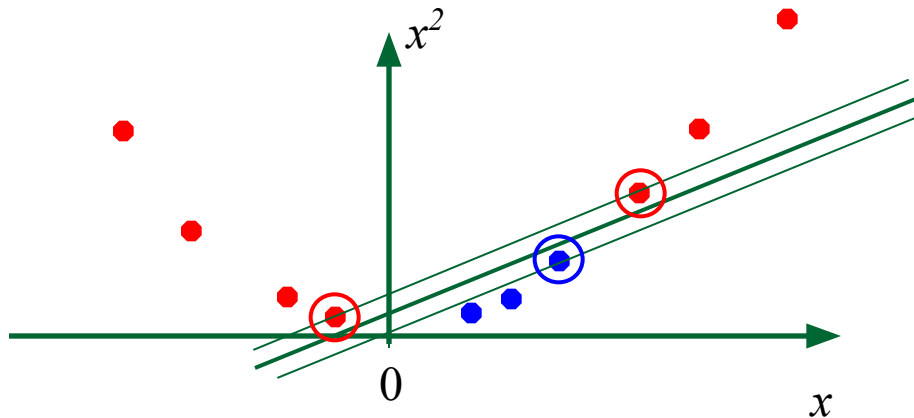
- Datasets that are linearly separable with some noise work out great:



- But what are we going to do if the dataset is just too hard?

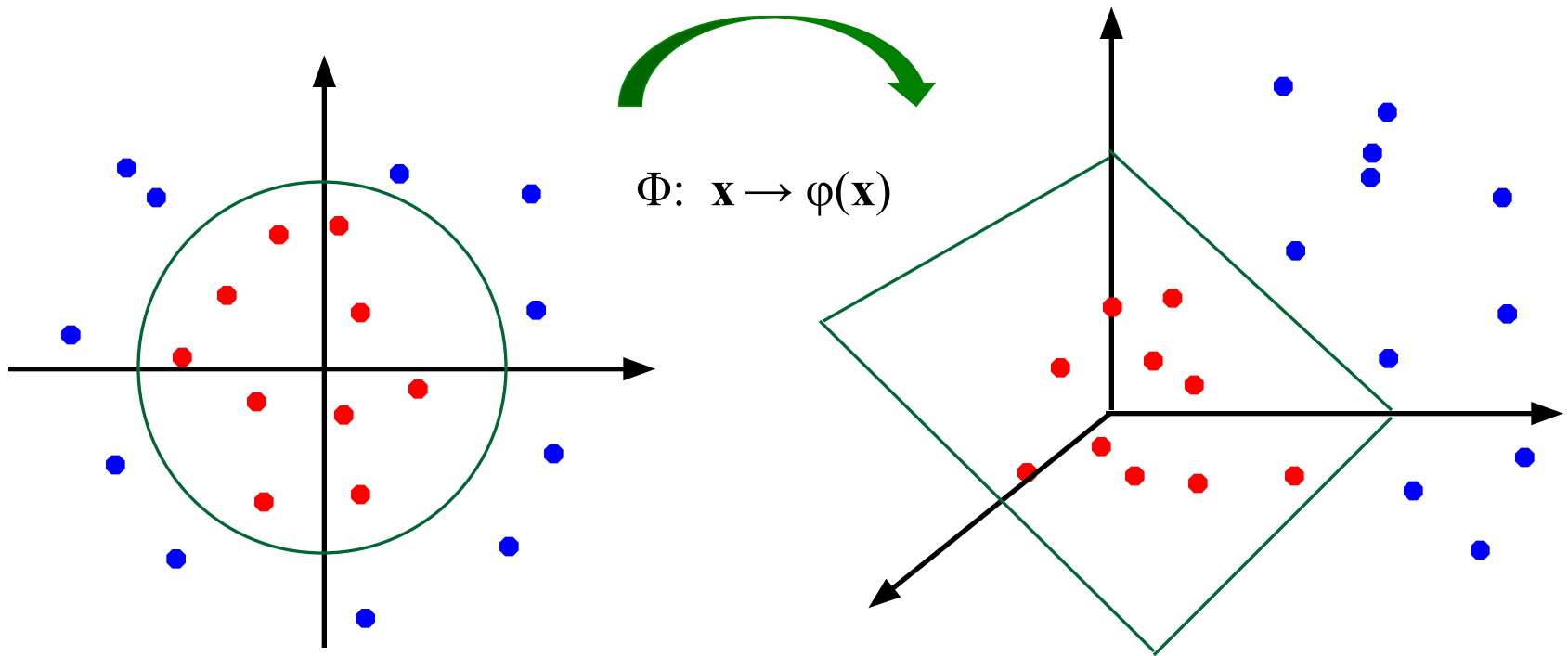


- How about... mapping data to a higher-dimensional space:



Non-linear SVMs: Feature spaces

- General idea: the original input space (nonlinear separable data) can always be mapped to some higher-dimensional feature space where the training set is linearly separable:



Mapping the Inputs to other dimensions - the use of Kernels

- Finding the optimal curve to fit the data is difficult.
- There is a way to “pre-process” the data in such a way that the problem is transformed into one of finding a simple hyperplane.
- We define a mapping $z = \phi(x)$ that transforms the d -dimensional input vector x into a (usually higher) d^* -dimensional vector z .
- We hope to choose a $\phi()$ so that the new training data $\{\phi(x_i), y_i\}$ is separable by a hyperplane.
- How do we go about choosing $\phi()$?

Efficient dot-product of polynomials

Polynomials of degree exactly d

$d=1$

$$\phi(u) \cdot \phi(v) = \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} \cdot \begin{pmatrix} v_1 \\ v_2 \end{pmatrix} = u_1 v_1 + u_2 v_2 = u \cdot v$$

$d=2$

$$\begin{aligned} \phi(u) \cdot \phi(v) &= \begin{pmatrix} u_1^2 \\ u_1 u_2 \\ u_2 u_1 \\ u_2^2 \end{pmatrix} \cdot \begin{pmatrix} v_1^2 \\ v_1 v_2 \\ v_2 v_1 \\ v_2^2 \end{pmatrix} = u_1^2 v_1^2 + 2u_1 v_1 u_2 v_2 + u_2^2 v_2^2 \\ &= (u_1 v_1 + u_2 v_2)^2 \\ &= (u \cdot v)^2 \end{aligned}$$

For any d :

$$\phi(u) \cdot \phi(v) = (u \cdot v)^d$$

- Taking a dot product and exponentiating gives same results as mapping into high dimensional space and then taking dot product

The “Kernel Trick”

- The linear classifier relies on dot product between vectors $K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T \mathbf{x}_j$
- If every data point is mapped into high-dimensional space via some transformation $\Phi: \mathbf{x} \rightarrow \phi(\mathbf{x})$, the dot product becomes:

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

- A *kernel function* is some function that corresponds to an inner product in some expanded feature space.
- Example:

2-dimensional vectors $\mathbf{x} = [x_1 \ x_2]$; let $K(\mathbf{x}_i, \mathbf{x}_j) = (1 + \mathbf{x}_i^T \mathbf{x}_j)^2$,

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

$$\begin{aligned} K(\mathbf{x}_i, \mathbf{x}_j) &= (1 + \mathbf{x}_i^T \mathbf{x}_j)^2, \\ &= 1 + x_{i1}^2 x_{j1}^2 + 2 x_{i1} x_{j1} x_{i2} x_{j2} + x_{i2}^2 x_{j2}^2 + 2 x_{i1} x_{j1} + 2 x_{i2} x_{j2} \\ &= [1 \ x_{i1}^2 \ \sqrt{2} x_{i1} x_{i2} \ x_{i2}^2 \ \sqrt{2} x_{i1} \ \sqrt{2} x_{i2}]^T [1 \ x_{j1}^2 \ \sqrt{2} x_{j1} x_{j2} \ x_{j2}^2 \ \sqrt{2} x_{j1} \ \sqrt{2} x_{j2}] \\ &= \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j), \quad \text{where } \phi(\mathbf{x}) = [1 \ x_1^2 \ \sqrt{2} x_1 x_2 \ x_2^2 \ \sqrt{2} x_1 \ \sqrt{2} x_2] \end{aligned}$$

Non-linear SVMs Mathematically

- **Dual problem formulation:**

Find $\alpha_1 \dots \alpha_N$ such that

$Q(\alpha) = \sum \alpha_i - \frac{1}{2} \sum \sum \alpha_i \alpha_j y_i y_j K(x_i, x_j)$ is maximized and

(1) $\sum \alpha_i y_i = 0$

(2) $\alpha_i \geq 0$ for all α_i

- **The solution is:**

$$f(x) = \sum \alpha_i y_i K(x_i, x_j) + b$$

- **Optimization techniques for finding α_i 's remain the same!**

Examples of Kernel Functions

- Linear: $K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T \mathbf{x}_j$
- Polynomial of power p : $K(\mathbf{x}_i, \mathbf{x}_j) = (1 + \mathbf{x}_i^T \mathbf{x}_j)^p$
- Gaussian (radial-basis function network):

$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}\right)$$

- Sigmoid: $K(\mathbf{x}_i, \mathbf{x}_j) = \tanh(\beta_0 \mathbf{x}_i^T \mathbf{x}_j + \beta_1)$

Nonlinear SVM - Overview

- SVM locates a separating hyperplane in the feature space and classify points in that space
 - It does not need to represent the space explicitly, simply by defining a kernel function
 - The kernel function plays the role of the dot product in the feature space.
-

Properties of SVM

- **Flexibility in choosing a similarity function**
 - **Sparseness of solution when dealing with large data sets**
 - only support vectors are used to specify the separating hyperplane
 - **Ability to handle large feature spaces**
 - complexity does not depend on the dimensionality of the feature space
 - **Overfitting can be controlled by soft margin approach**
 - **Nice math property: a simple convex optimization problem which is guaranteed to converge to a single global solution**
 - **Feature Selection**
-

Weakness of SVM

- **It is sensitive to noise**

- A relatively small number of mislabeled examples can dramatically decrease the performance

- **It only considers two classes**

- how to do multi-class classification with SVM?

- Answer:

1) with output arity m , learn m SVM's

- SVM 1 learns "Output==1" vs "Output != 1"
- SVM 2 learns "Output==2" vs "Output != 2"
- :
- SVM m learns "Output== m " vs "Output != m "

2) To predict the output for a new input, just predict with each SVM and find out which one puts the prediction the furthest into the positive region.

Some Issues

- **Choice of kernel**
 - Gaussian or polynomial kernel is default
 - if ineffective, more elaborate kernels are needed
 - domain experts can give assistance in formulating appropriate similarity measures
- **Choice of kernel parameters**
 - e.g. σ in Gaussian kernel
 - σ is the distance between closest points with different classifications
 - In the absence of reliable criteria, applications rely on the use of a validation set or cross-validation to set such parameters.
- **Optimization criterion** – Hard margin v.s. Soft margin
 - a lengthy series of experiments in which various parameters are tested

Additional Resources

- **An excellent tutorial on VC-dimension and Support Vector Machines:**
C.J.C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):955-974, 1998.
- **The VC/SRM/SVM Bible:**
Statistical Learning Theory by Vladimir Vapnik, Wiley-Interscience; 1998

<http://www.kernel-machines.org/>
