Dr. George Karraz, Ph. D.

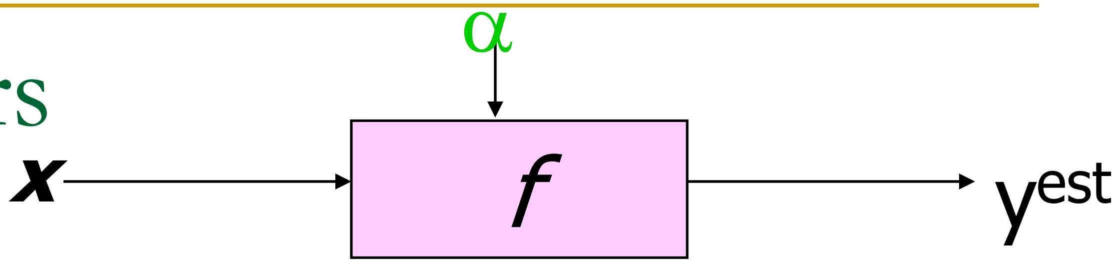
Support Vector Machine & Its Applications

George Karraz, Ph.D.

Overview

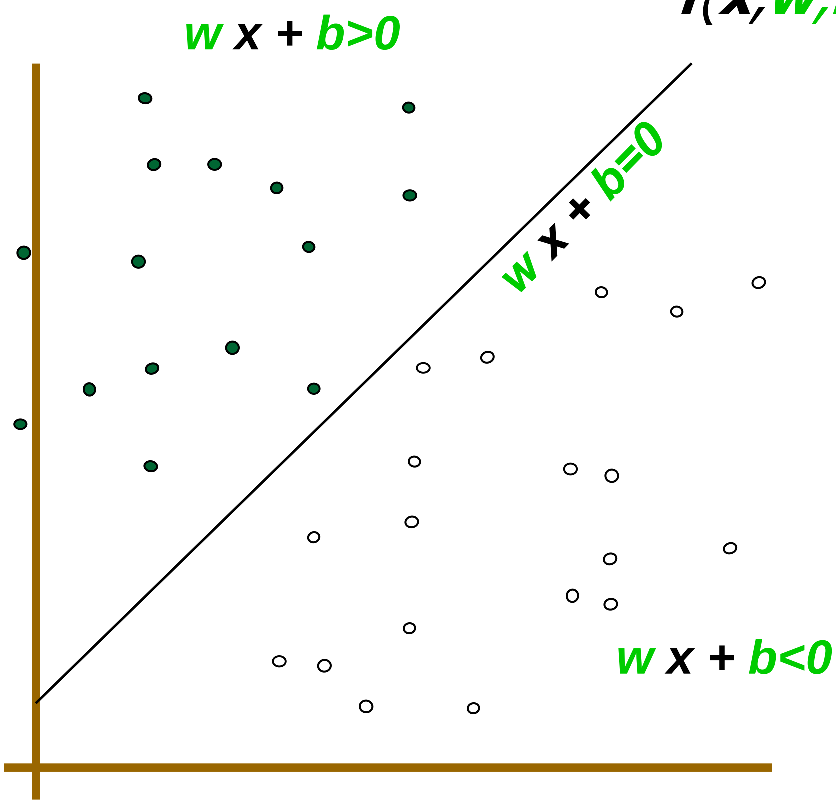
- Intro. to Support Vector Machines (SVM)
- Properties of SVM
- Applications
  - Gene Expression Data Classification
  - Text Categorization *if time permits*
- Discussion

Linear Classifiers



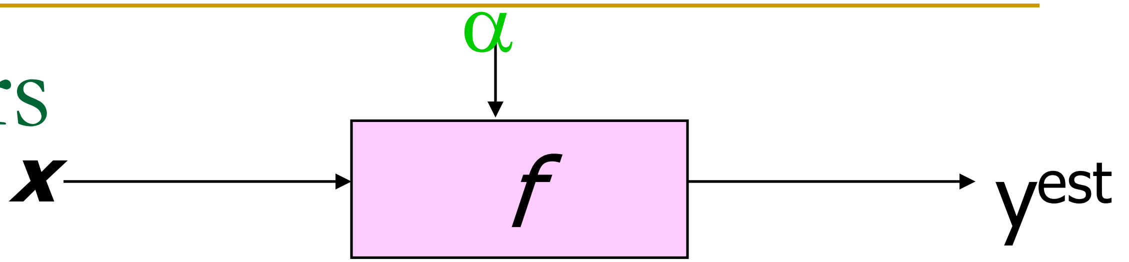
- denotes +1
- denotes -1

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



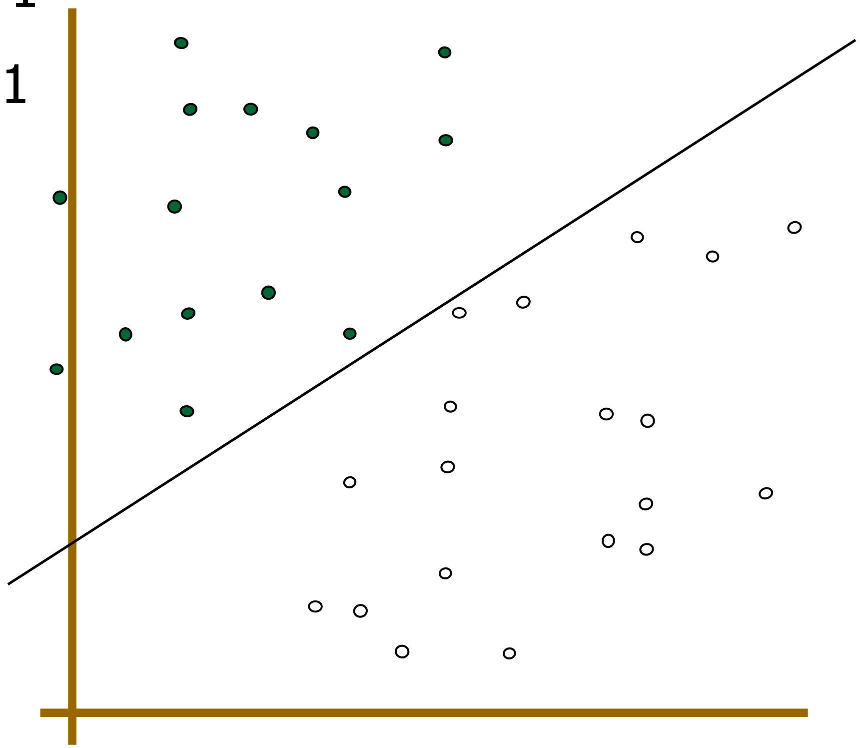
How would you classify this data?

Linear Classifiers



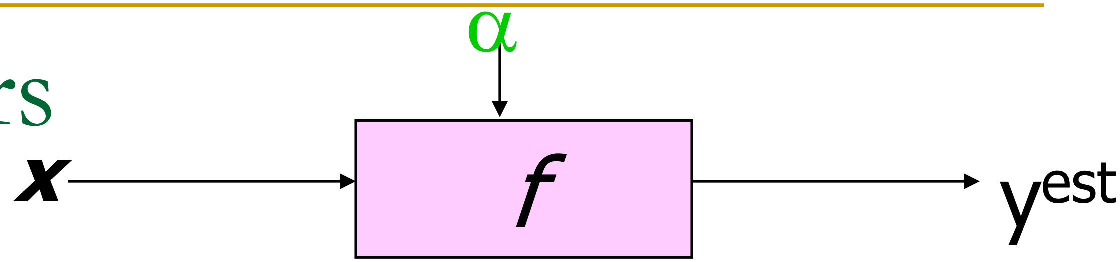
$f(\mathbf{x},\mathbf{w},b) = sign(\mathbf{w}\ \mathbf{x} + b)$

- ● denotes +1
- ○ denotes -1



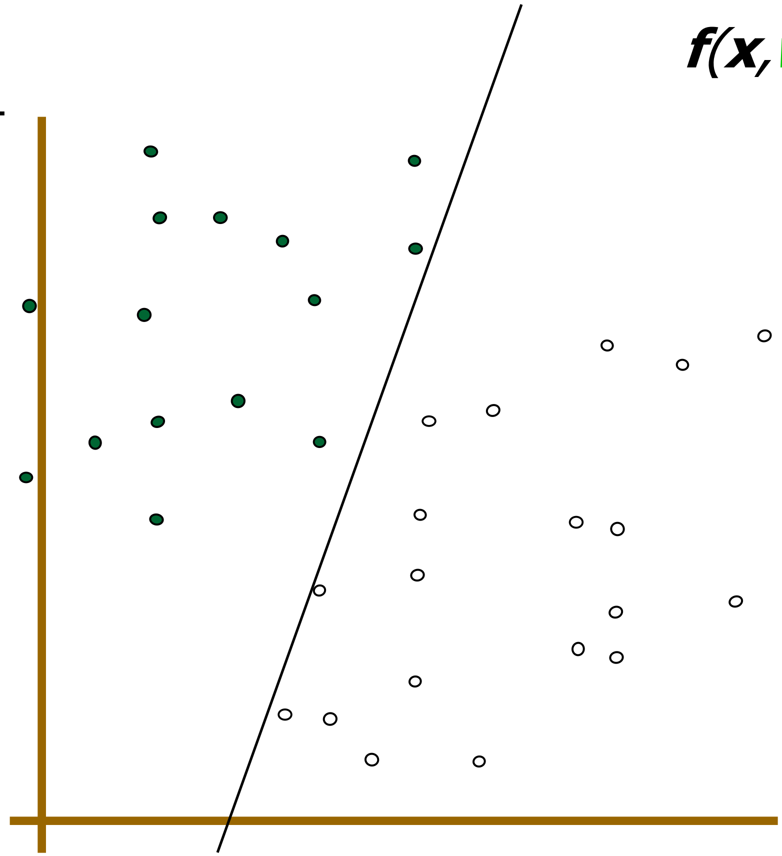
How would you classify this data?

Linear Classifiers



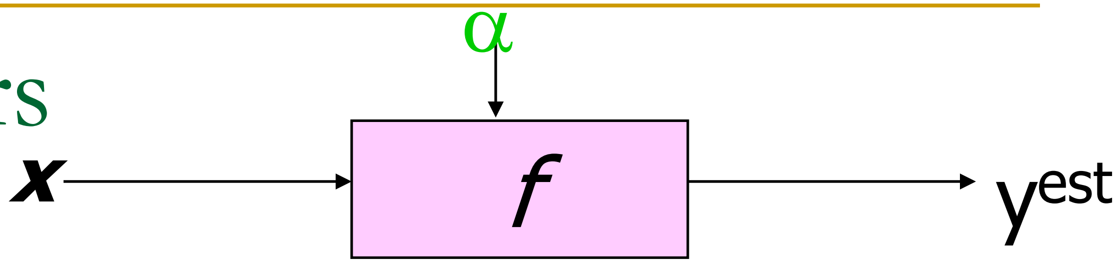
$$f(\mathbf{x},\mathbf{w},b) = sign(\mathbf{w}\ \mathbf{x} + b)$$

- denotes +1
- denotes -1



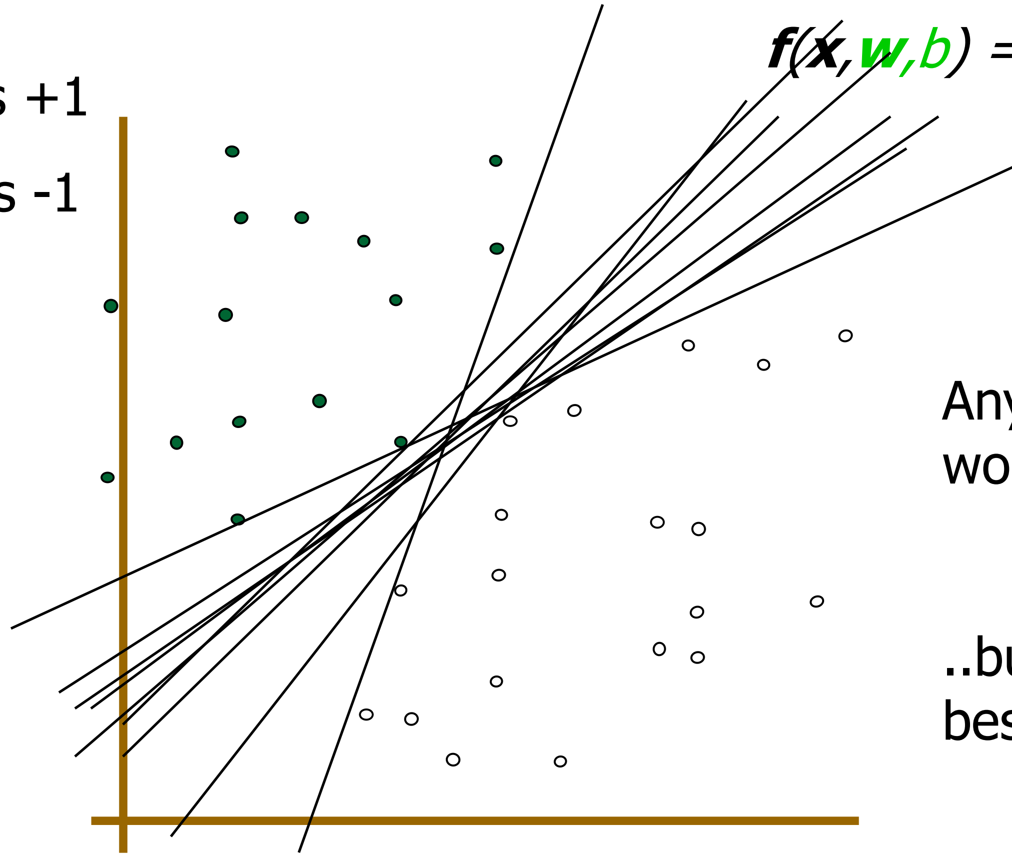
How would you classify this data?

Linear Classifiers



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

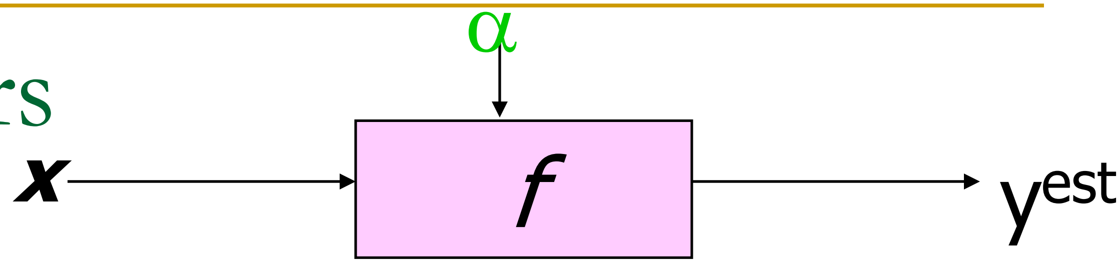
- ● denotes +1
- ○ denotes -1



Any of these would be fine..

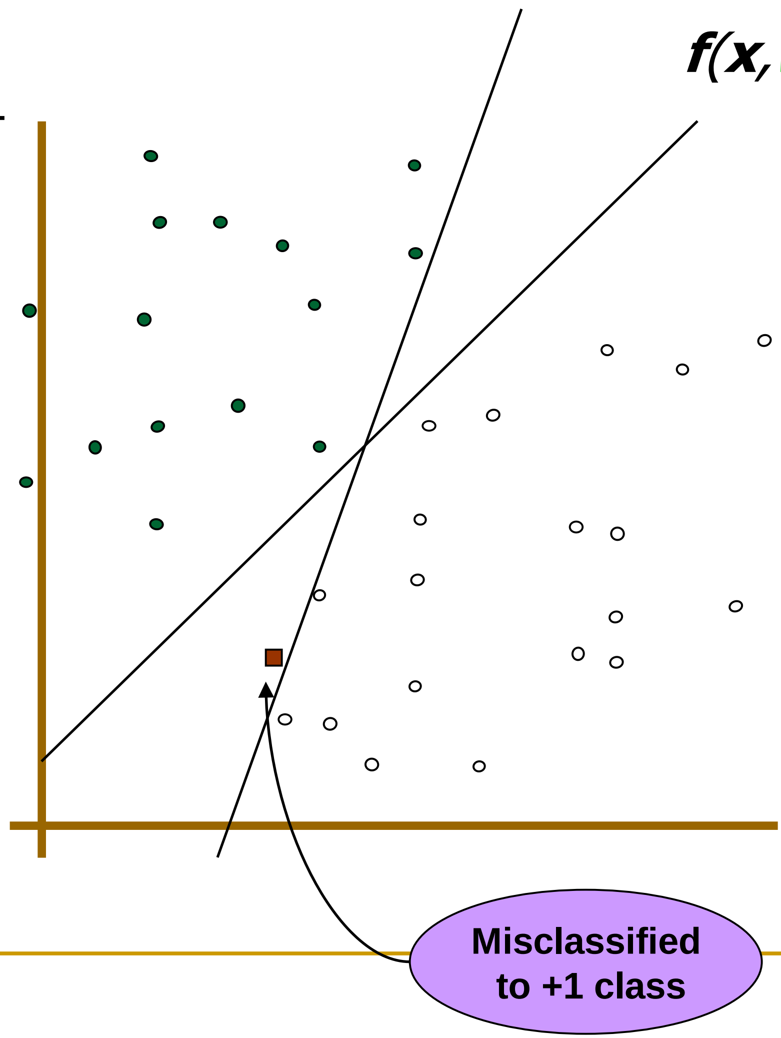
..but which is best?

Linear Classifiers



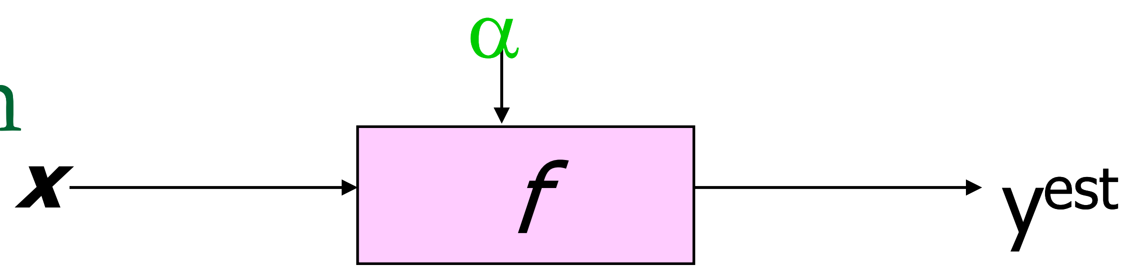
$\boldsymbol{f}(\mathbf{x},\mathbf{w},b) = sign(\mathbf{w}\ \mathbf{x} + b)$

- ● denotes +1
- ○ denotes -1

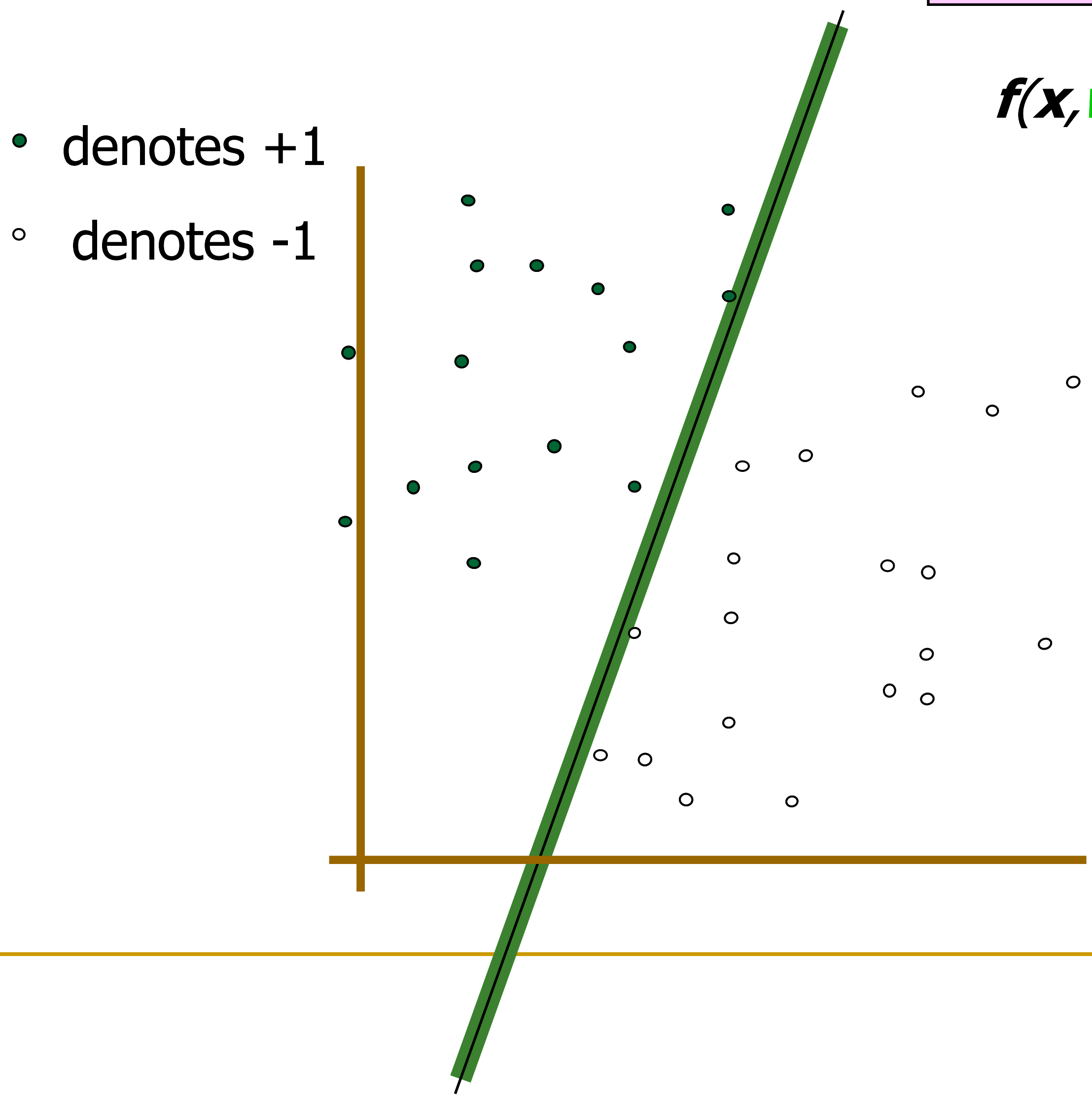


How would you classify this data?

Classifier Margin

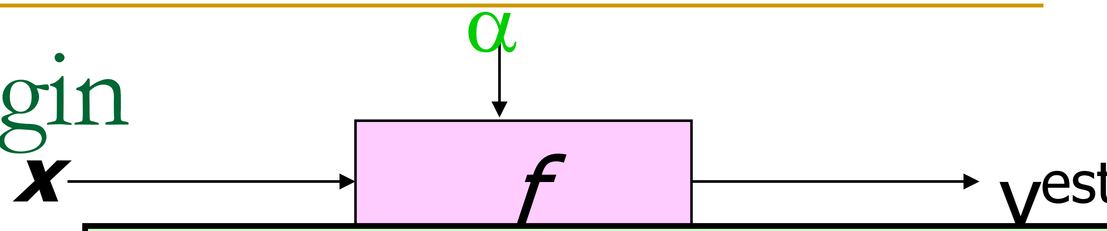


$f(\mathbf{x},\mathbf{w},b) = sign(\mathbf{w}\ \mathbf{x} + b)$



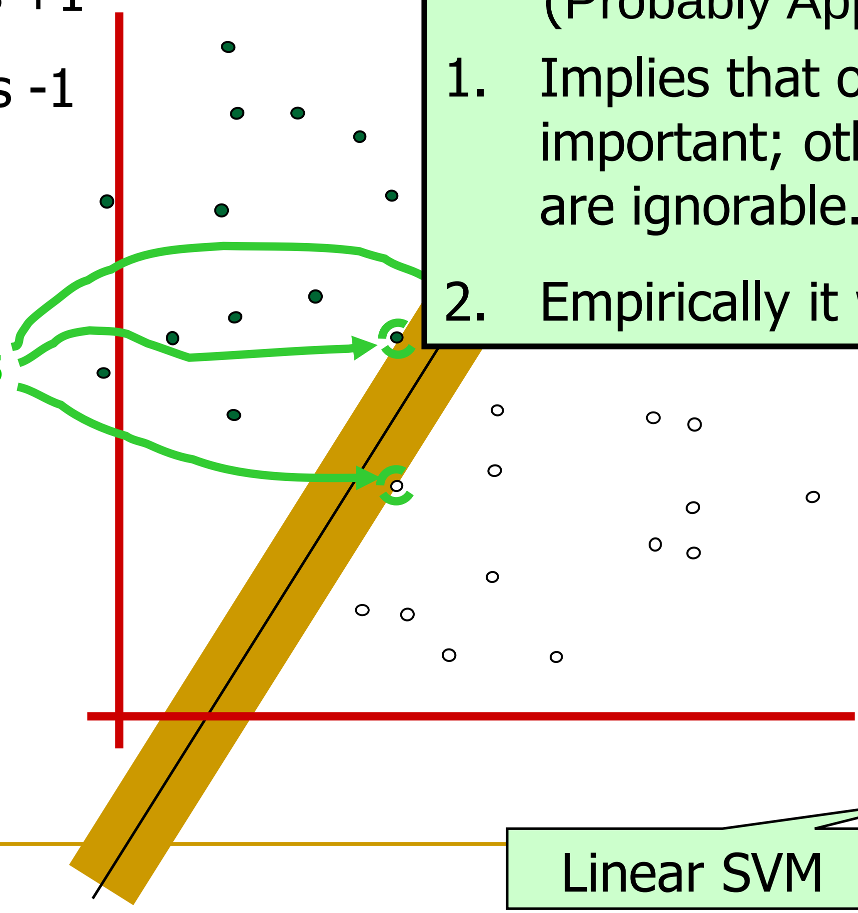
Define the margin of a linear classifier as the width that the boundary could be increased by before hitting a datapoint.

Maximum Margin



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

- denotes +1
- denotes -1



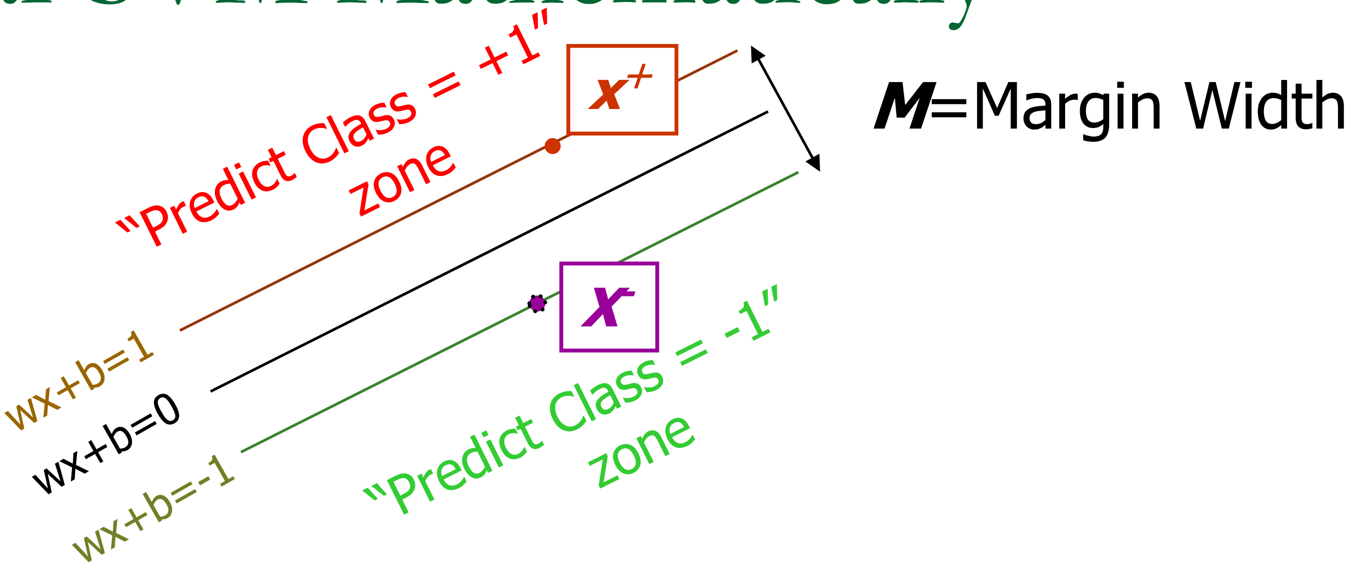
Support Vectors are those datapoints that the margin pushes up against

with the, um, maximum margin.

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

Linear SVM

Linear SVM Mathematically



What we know:

- $\boldsymbol{w} \, . \, \boldsymbol{x}^+ + b = +1$
- $\boldsymbol{w} \, . \, \boldsymbol{x}^- + b = -1$
- $\boldsymbol{w} \, . \, (\boldsymbol{x}^+ - \boldsymbol{x}^-) = 2$

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

Linear SVM Mathematically

- Goal: **1) Correctly classify all training data**

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

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

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

2) Maximize the Margin $\quad M = \dfrac{2}{|w|}$

same as minimize $\quad \dfrac{1}{2} w^t w$

- **We can formulate a Quadratic Optimization Problem and solve for w and b**

- Minimize $\quad \Phi(w) = \dfrac{1}{2} w^t w$

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

Solving the Optimization Problem

Find $\mathbf{w}$ and b such that
$\mathbf{\Phi}(\mathbf{w}) = \frac{1}{2}\, \mathbf{w}^T\mathbf{w}$ is minimized;
and for all $\{(\mathbf{x_i}, y_i)\}$: $y_i(\mathbf{w}^T\mathbf{x_i} + b) \geq 1$

- **Need to optimize a *quadratic* function subject to *linear* constraints.**
- **Quadratic optimization problems are a well-known class of mathematical programming problems, and many (rather intricate) algorithms exist for solving them.**
- **The solution involves constructing a *dual problem* where a *Lagrange multiplier* α_i is associated with every constraint in the primary problem:**

Find $\alpha_1 \ldots \alpha_N$ such that
$\mathbf{Q}(\boldsymbol{\alpha}) = \Sigma\alpha_i - \frac{1}{2}\Sigma\Sigma\alpha_i\alpha_j y_i y_j \mathbf{x_i}^T\mathbf{x_j}$ is maximized and
(1) $\Sigma\alpha_i y_i = 0$
(2) $\alpha_i \geq 0$ for all α_i

The Optimization Problem Solution

- The solution has the form:

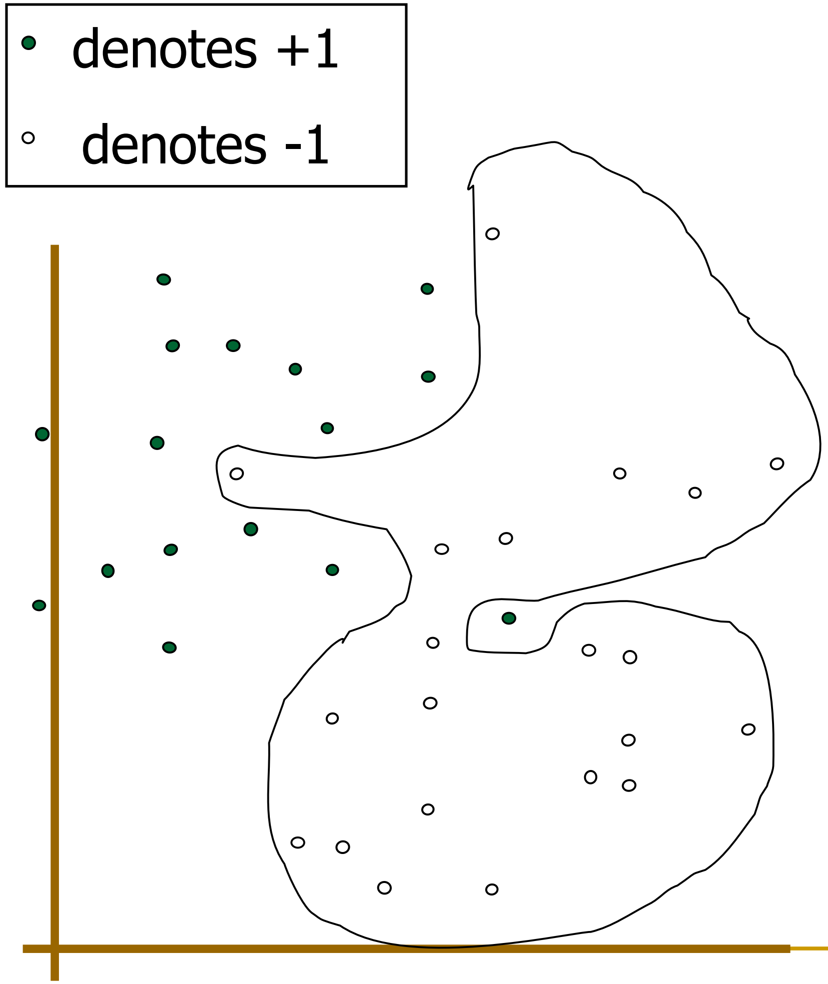
$$\mathbf{w} = \Sigma\alpha_i y_i \mathbf{x_i} \qquad b = y_k - \mathbf{w^T x_k} \text{ for any } \mathbf{x_k} \text{ such that } \alpha_k \neq 0$$

- Each non-zero α_i indicates that corresponding $\mathbf{x_i}$ is a support vector.
- Then the classifying function will have the form:

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

- Notice that it relies on an *inner product* between the test point $\mathbf{x}$ and the support vectors $\mathbf{x_i}$ – we will return to this later.
- Also keep in mind that solving the optimization problem involved computing the inner products $\mathbf{x_i^T x_j}$ between all pairs of training points.

Dataset with noise



- **Hard Margin:** **So far we require all data points be classified correctly**

  - No training error

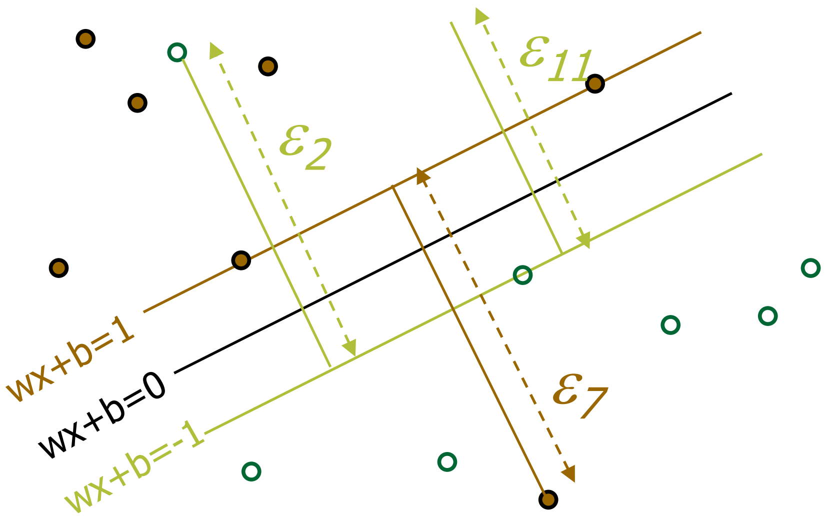
- **What if the training set is noisy?**

  - Solution 1: **use very powerful kernels**

OVERFITTING!

Soft Margin Classification

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



What should our quadratic optimization criterion be?

Minimize

$$\frac{1}{2}\mathbf{w}.\mathbf{w} + C\sum_{k=1}^{R}\varepsilon_k$$

Hard Margin v.s. Soft Margin

- **The old formulation:**

Find **w** and b such that

$\mathbf{\Phi(w)} = \frac{1}{2}\, \mathbf{w}^T\mathbf{w}$ is minimized and for all $\{(\mathbf{x_i}, y_i)\}$

$y_i\, (\mathbf{w^T x_i} + \mathrm{b}) \geq 1$

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

Find **w** and b such that

$\mathbf{\Phi(w)} = \frac{1}{2}\, \mathbf{w}^T\mathbf{w} + C\sum\xi_i$ is minimized and for all $\{(\mathbf{x_i}, y_i)\}$

$y_i\, (\mathbf{w^T x_i} + b) \geq 1 - \xi_i$ and $\xi_i \geq 0$ for all i

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

Linear SVMs: Overview

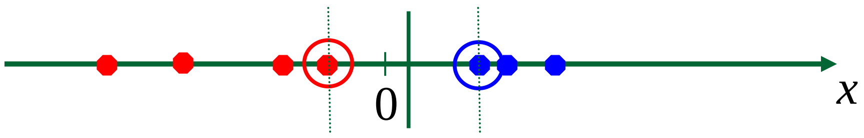
- **The classifier is a *separating hyperplane.***
- **Most "important" training points are support vectors; they define the hyperplane.**
- **Quadratic optimization algorithms can identify which training points $\mathbf{x}_i$ are support vectors with non-zero Lagrangian multipliers α_i.**
- **Both in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

Find $\alpha_1 \ldots \alpha_N$ such that
$Q(\alpha) = \Sigma\alpha_i - \frac{1}{2}\Sigma\Sigma\alpha_i\alpha_j y_i y_j \mathbf{x}_i^T\mathbf{x}_j$ is maximized and
(1) $\Sigma\alpha_i y_i = 0$
(2) $0 \leq \alpha_i \leq C$ for all α_i

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

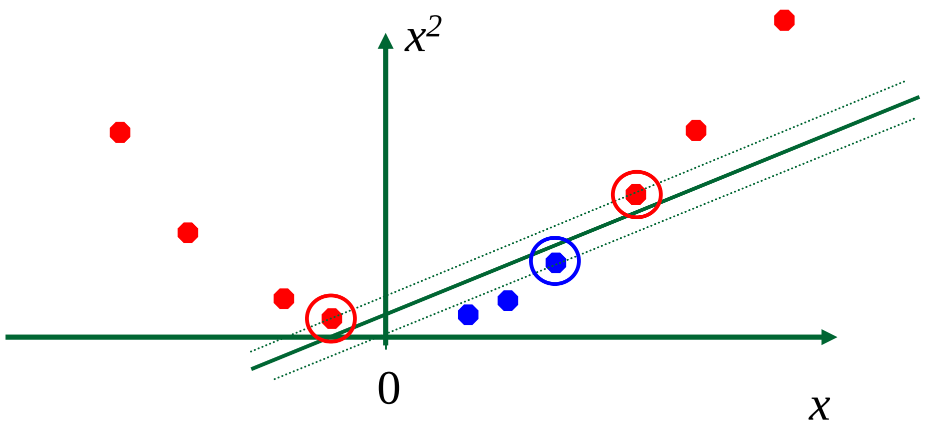
Non-linear SVMs

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



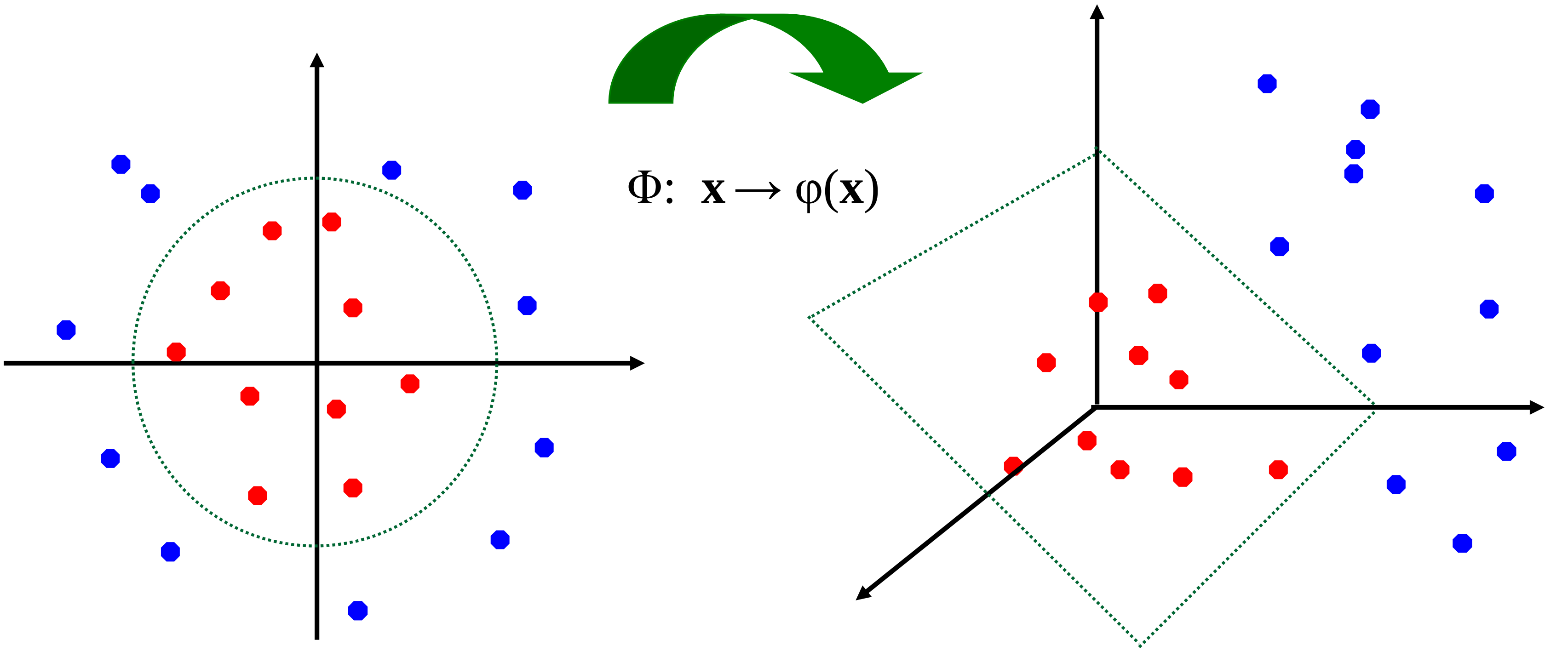
- But what are we going to do if the dataset is just too hard?

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



Non-linear SVMs: Feature spaces

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



The "Kernel Trick"

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

$$K(\mathrm{x}_i,\mathrm{x}_j)= \varphi(\mathrm{x}_i)^T\varphi(\mathrm{x}_j)$$

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

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

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

$$K(\mathrm{x}_i,\mathrm{x}_j)=(1 + \mathrm{x}_i^T\mathrm{x}_j)^2,$$

$$= 1+ x_{i1}^2x_{j1}^2 + 2\, x_{i1}x_{j1}\, x_{i2}x_{j2}+ x_{i2}^2x_{j2}^2 + 2x_{i1}x_{j1} + 2x_{i2}x_{j2}$$

$$= [1\ \ x_{i1}^2\ \ \sqrt{2}\, x_{i1}x_{i2}\ \ x_{i2}^2\ \ \sqrt{2}x_{i1}\ \ \sqrt{2}x_{i2}]^T[1\ \ x_{j1}^2\ \ \sqrt{2}\, x_{j1}x_{j2}\ \ x_{j2}^2\ \ \sqrt{2}x_{j1}\ \ \sqrt{2}x_{j2}]$$

$$= \varphi(\mathrm{x}_i)^T\varphi(\mathrm{x}_j), \quad \text{where } \varphi(\mathrm{x}) = [1\ \ x_1^2\ \ \sqrt{2}\, x_1x_2\ \ x_2^2\ \ \sqrt{2}x_1\ \ \sqrt{2}x_2]$$

What Functions are Kernels?

- **For some functions $K(x_i,x_j)$ checking that**

  $K(x_i,x_j)= \varphi(x_i)^T\varphi(x_j)$ can be cumbersome.
- **Mercer's theorem:**

  Every semi-positive definite symmetric function is a kernel
- **Semi-positive definite symmetric functions correspond to a semi-positive definite symmetric Gram matrix:**

K=

$K(\mathbf{x}_1,\mathbf{x}_1)$	$K(\mathbf{x}_1,\mathbf{x}_2)$	$K(\mathbf{x}_1,\mathbf{x}_3)$	…	$K(\mathbf{x}_1,\mathbf{x}_N)$
$K(\mathbf{x}_2,\mathbf{x}_1)$	$K(\mathbf{x}_2,\mathbf{x}_2)$	$K(\mathbf{x}_2,\mathbf{x}_3)$		$K(\mathbf{x}_2,\mathbf{x}_N)$
…	…	…	…	…
$K(\mathbf{x}_N,\mathbf{x}_1)$	$K(\mathbf{x}_N,\mathbf{x}_2)$	$K(\mathbf{x}_N,\mathbf{x}_3)$	…	$K(\mathbf{x}_N,\mathbf{x}_N)$

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: $K(\mathbf{x_i},\mathbf{x_j}) = \tanh(\beta_0\mathbf{x_i}^T\mathbf{x_j} + \beta_1)$

Non-linear SVMs Mathematically

- **Dual problem formulation:**

Find $\alpha_1 \ldots \alpha_N$ such that
$Q(\alpha) = \Sigma\alpha_i - \frac{1}{2}\Sigma\Sigma\alpha_i\alpha_j y_i y_j K(\mathrm{x}_i, \mathrm{x}_j)$ is maximized and
(1) $\Sigma\alpha_i y_i = 0$
(2) $\alpha_i \geq 0$ for all α_i

- **The solution is:**

$$f(\mathrm{x}) = \Sigma\alpha_i y_i K(\mathrm{x}_i, \mathrm{x}_j) + b$$

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

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**

SVM Applications

- **SVM has been used successfully in many real-world problems**

  - text (and hypertext) categorization

  - image classification

  - bioinformatics (Protein classification, Cancer classification)

  - hand-written character recognition

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.

Application 2: Text Categorization

- Task: The classification of natural text (or hypertext) documents into a fixed number of predefined categories based on their content.

  - email filtering, web searching, sorting documents by topic, etc..

- A document can be assigned to more than one category, so this can be viewed as a series of binary classification problems, one for each category

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