LUND UNIVERSITY

FMAN-45: Machine Learning

Lecture 5: Support Vector Machines

Cristian Sminchisescu

Back to Binary Classification Setup…

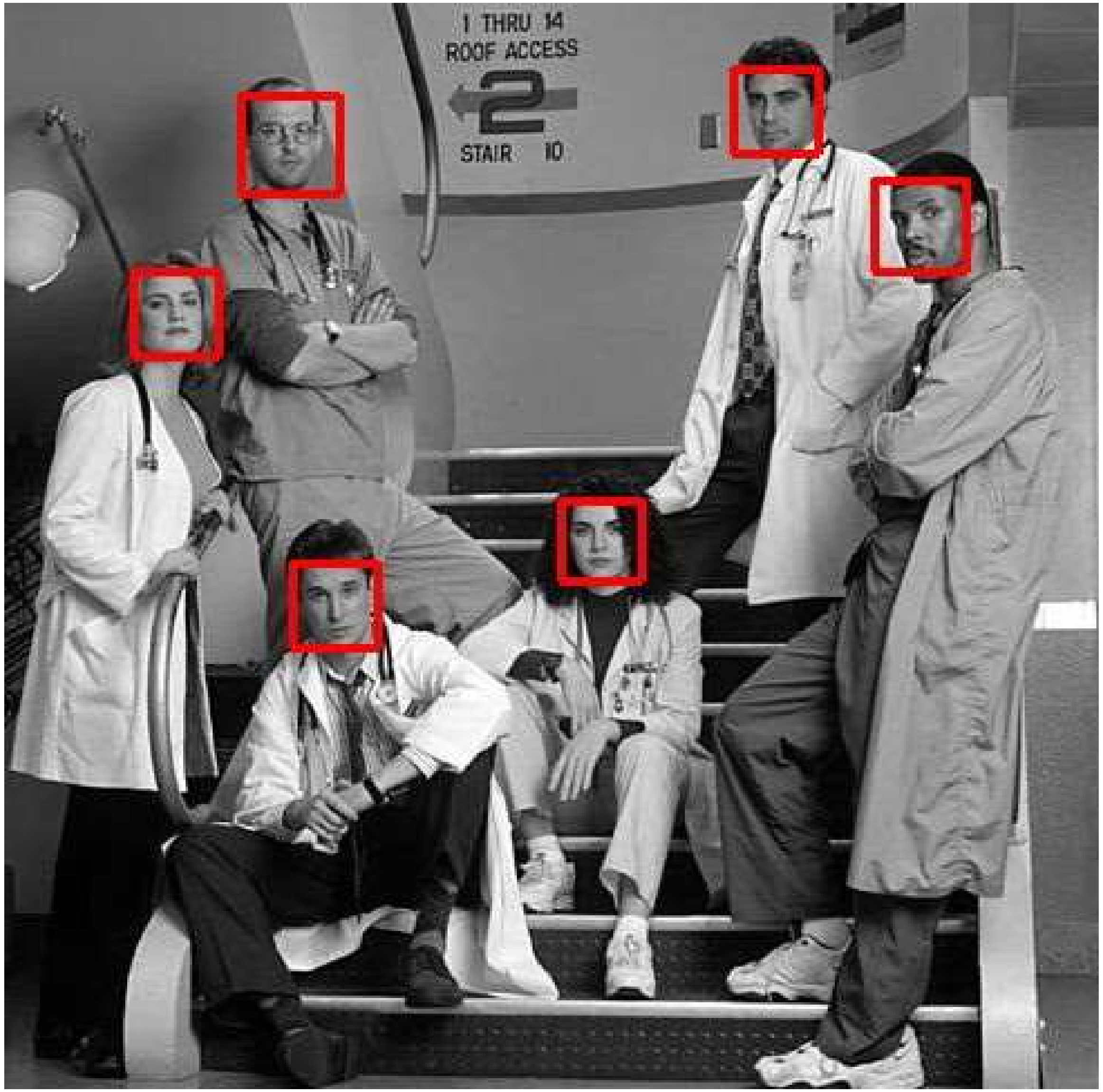
- We are given a finite, possibly noisy, set of training data: $\{\mathbf{x}_i, y_i\}, i = 1, .., N$. Each input $\mathbf{x}$ is paired with a binary output y $(+1 \text{ or } -1)$

- Based only on training data, construct a machine that generates outputs y, given inputs $\mathbf{x}$

- Now, a new sample is drawn from the same distribution as the training sample

- We wish to run the machine on the new sample input, and be able to classify it correctly, as either positive or negative

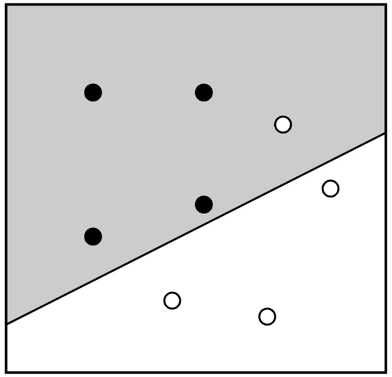
Example: Face Detection



Discriminant Function

Once again, we will restrict our attention to learning machines that separate the positive and negative examples using a linear function, with parameters $(\mathbf{w}, b)$

$$g(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + b$$



Linear Functions

Linear Discriminant Function

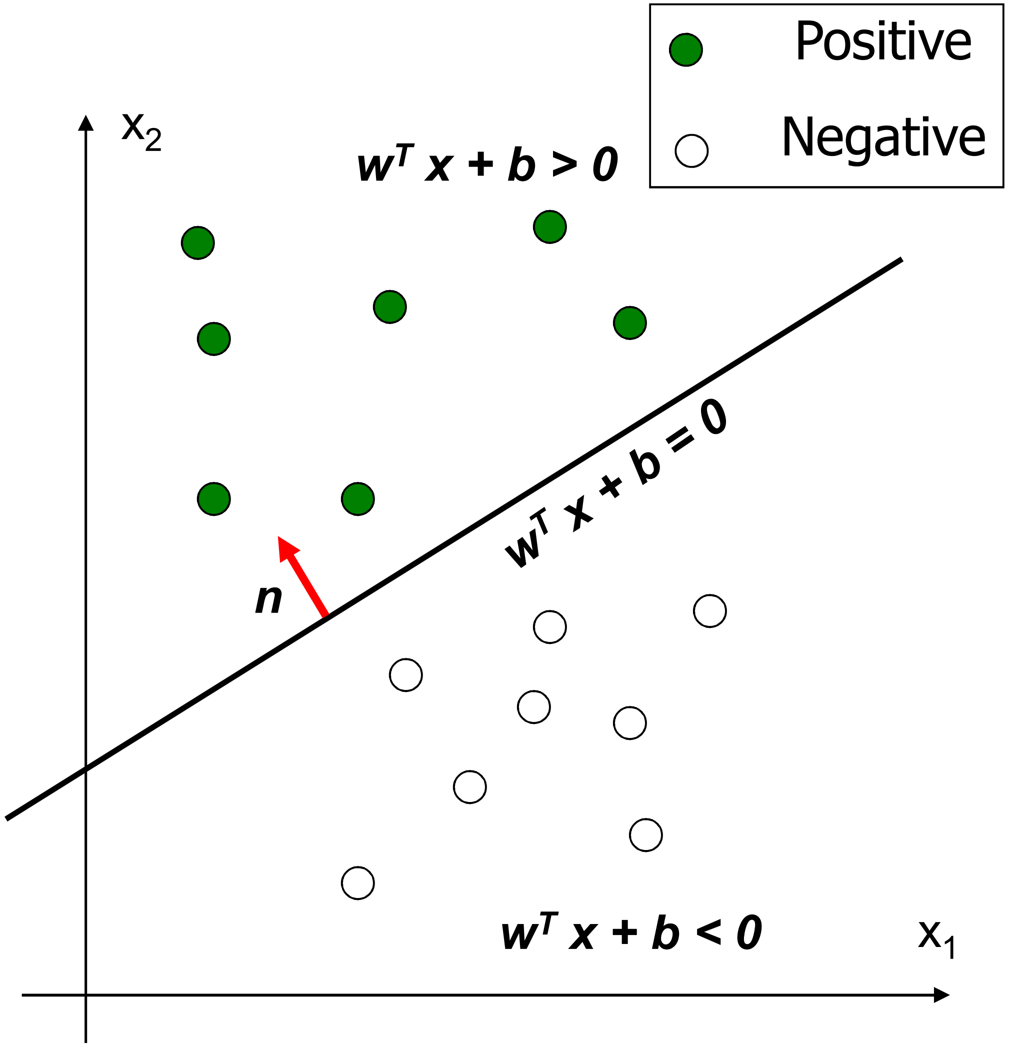
- $g(\mathbf{x})$ is a linear function:

$$g(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + b$$

- A hyper-plane in feature space $\mathbf{x}$

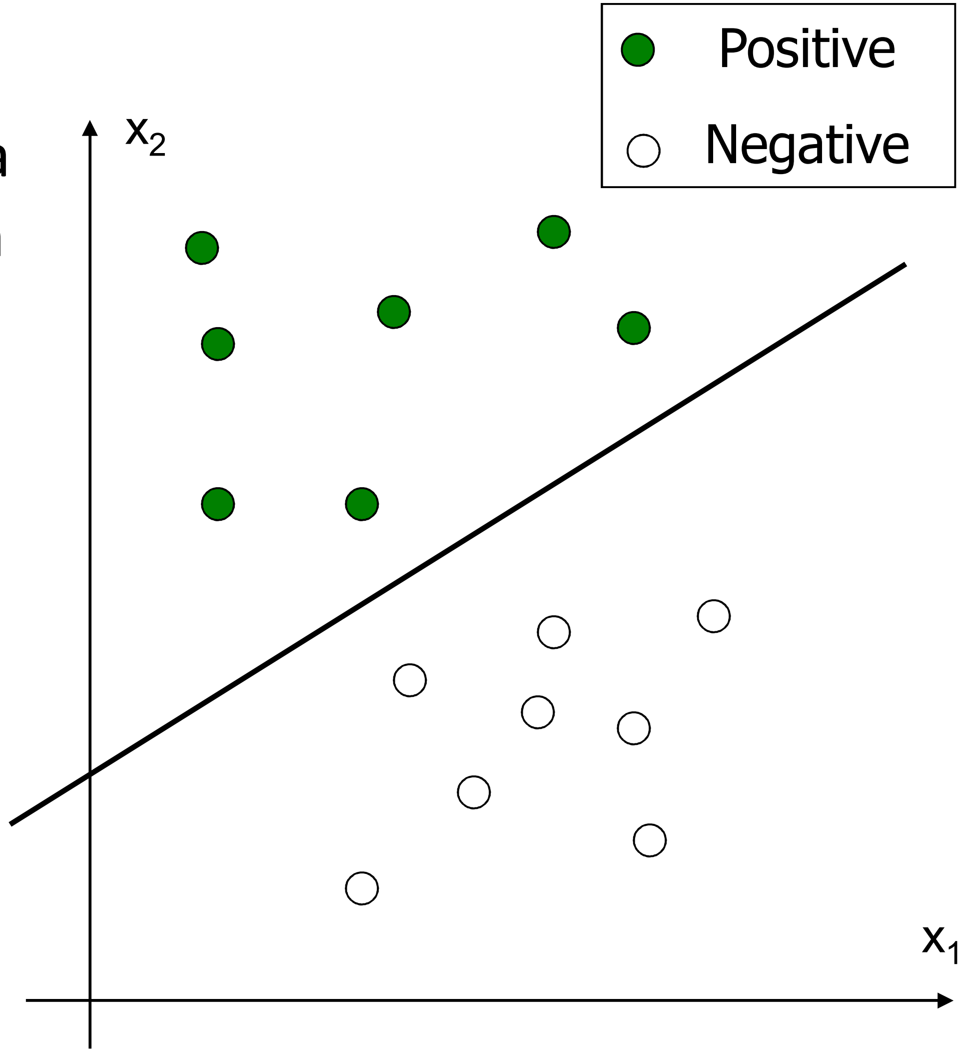
- The unit-length normal vector of the hyper-plane:

$$\mathbf{n} = \frac{\mathbf{w}}{\|\mathbf{w}\|}$$



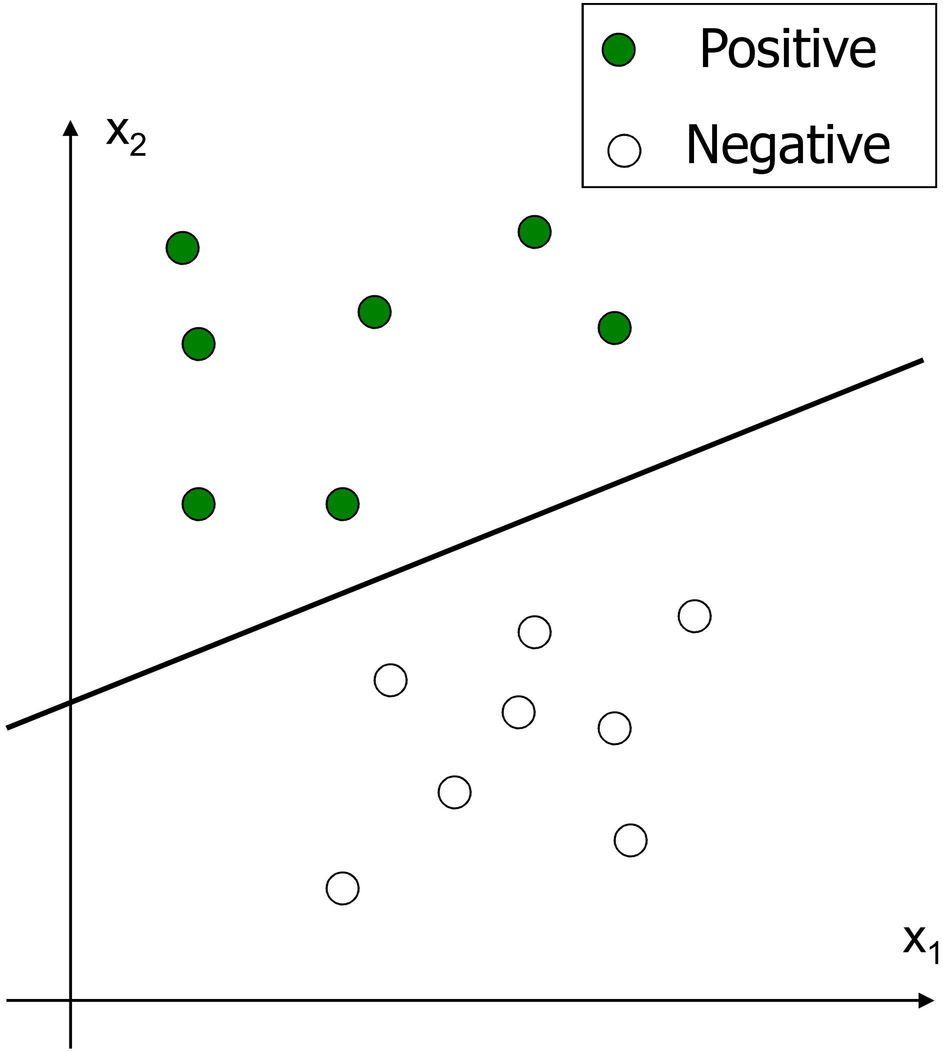
Linear Discriminant Function

How can we classify the data using a linear discriminant in order to minimize the error rate ?



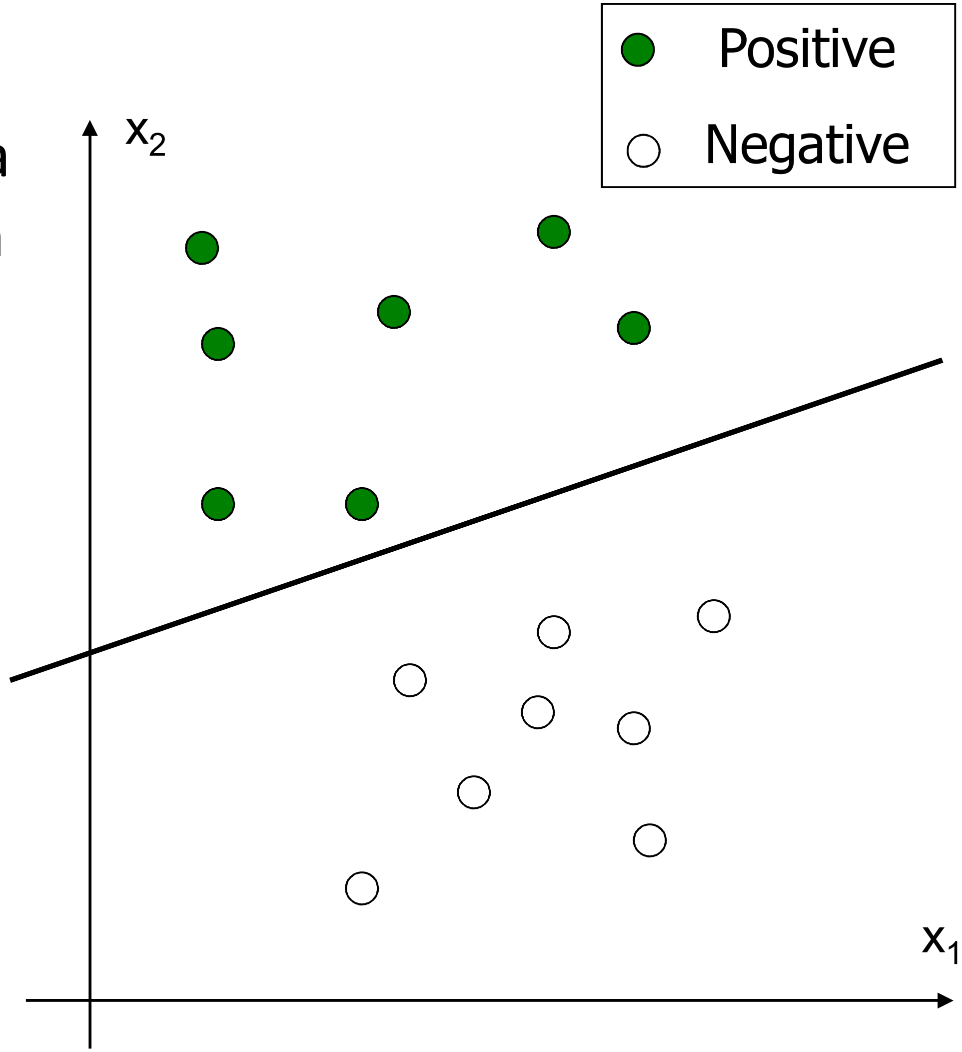
Linear Discriminant Function

How can we classify the data using a linear discriminant in order to minimize the error rate ?



Linear Discriminant Function

How can we classify the data using a linear discriminant in order to minimize the error rate ?

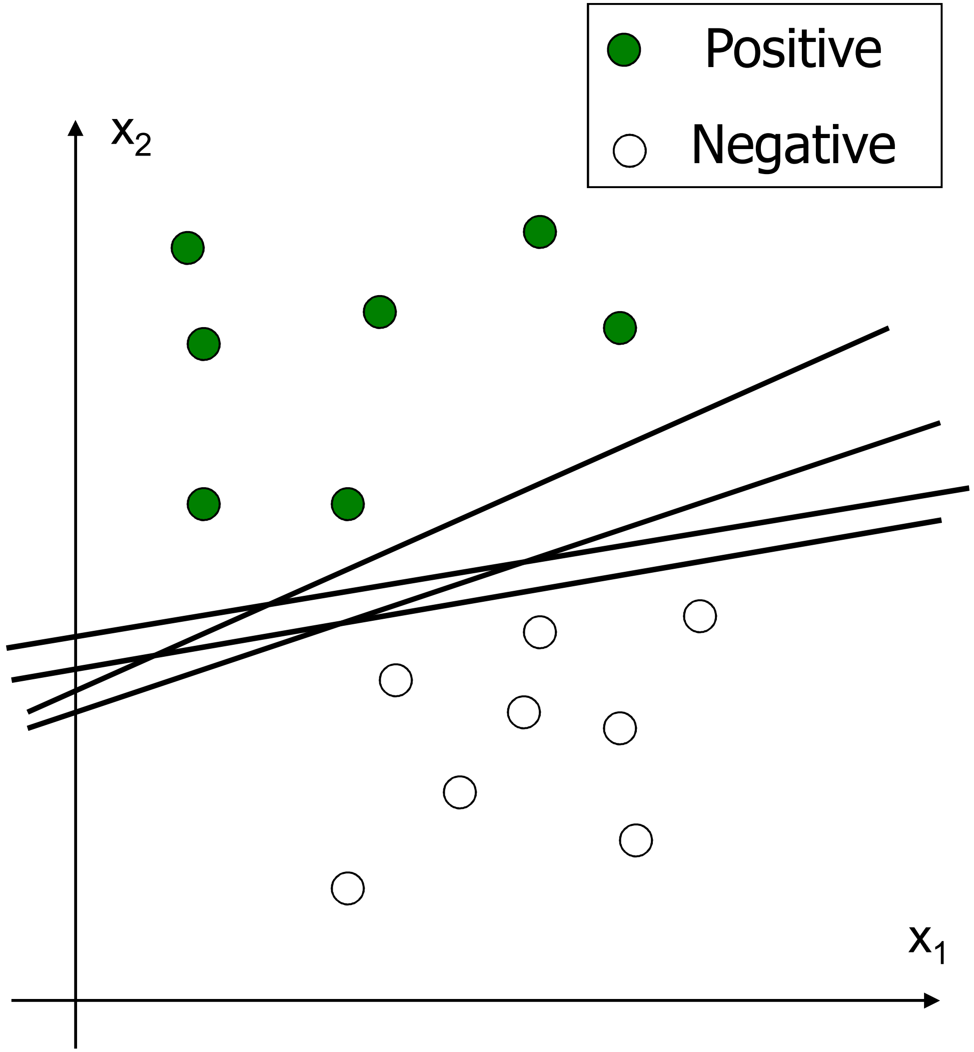


Linear Discriminant Function

How can we classify the data using a linear discriminant in order to minimize the error rate ?

- Many possible answers!
- Which one is the best?

A: *The best is the one that gives the lowest error on new test data!*



Large Margin Linear Classifier

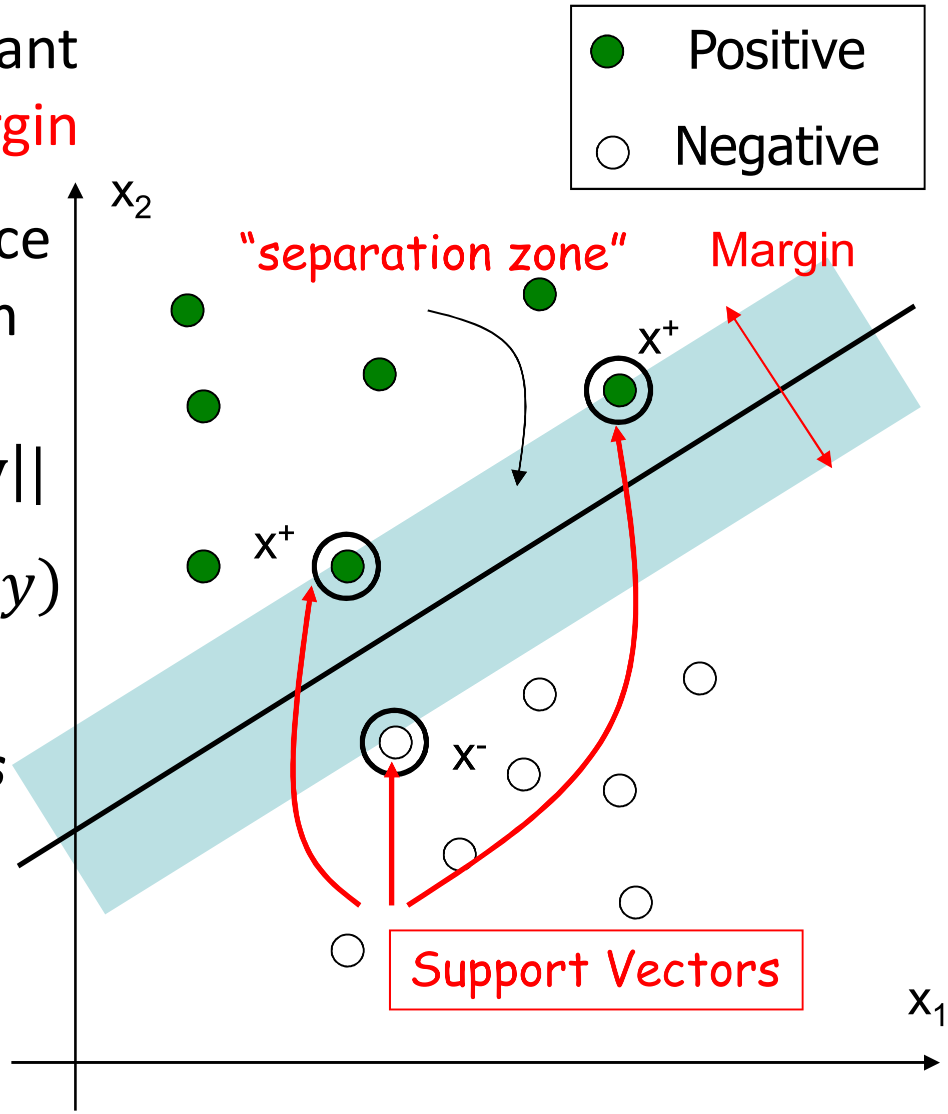
One option: the linear discriminant function with the maximum margin

- ***Geometric margin*** is the distance to a separating hyperplane from the point closest to it:

  $$\rho_{\mathbf{w},b}(\mathbf{x}, y) = y(\mathbf{w}^T\mathbf{x} + b)/||\mathbf{w}||$$

  $$\text{Margin } \rho_{\mathbf{w},b} := \min_{i=1,..,N} \rho_{\mathbf{w},b}(\mathbf{x}_i, y)$$

- Examples closest to the hyperplane are ***support vectors***
- *The discriminant* ***margin*** is the maximum width of the band that can be drawn, separating contrastive support vectors



Large Margin Linear Classifier

- Why is it good to focus on large margin?
  - Robust to outliers and thus strong generalization ability

- Fundamental result
  - If a d-dimensional dataset is enclosed inside a sphere of radius R, and the margin of the linear classifier is M, then the generalization error (difference between expected and empirical error) of the classifier is bounded by a function of the VC dimension, and

$$VC \leq \min\left\{d, \left\lceil \frac{4R^2}{M^2} \right\rceil\right\} + 1$$

- In particular, indepedently from the dimension d, we can reduce the VC dimension by increasing the margin.

VC Dimension

- Consider a binary classification problem, and a function class
- Each function of the class induces a labeling of patterns
- There are at most 2^N labelings for N patterns
- If a very rich function class might be able to realize all 2^N separations, it is said to **shatter** the N points
- However the function may not be rich enough
- The VC dimension is defined as the largest m such that there exist a set of m points which the class can shatter, or ∞ if no such m exists
- It is a one number summary for the capacity of the learning machine

VC Dimension Example

There exist VC dimension points that can be shattered, i.e. arbitrarily classified.
Here: There exist 3 points in 2 dimensions that can be shattered.

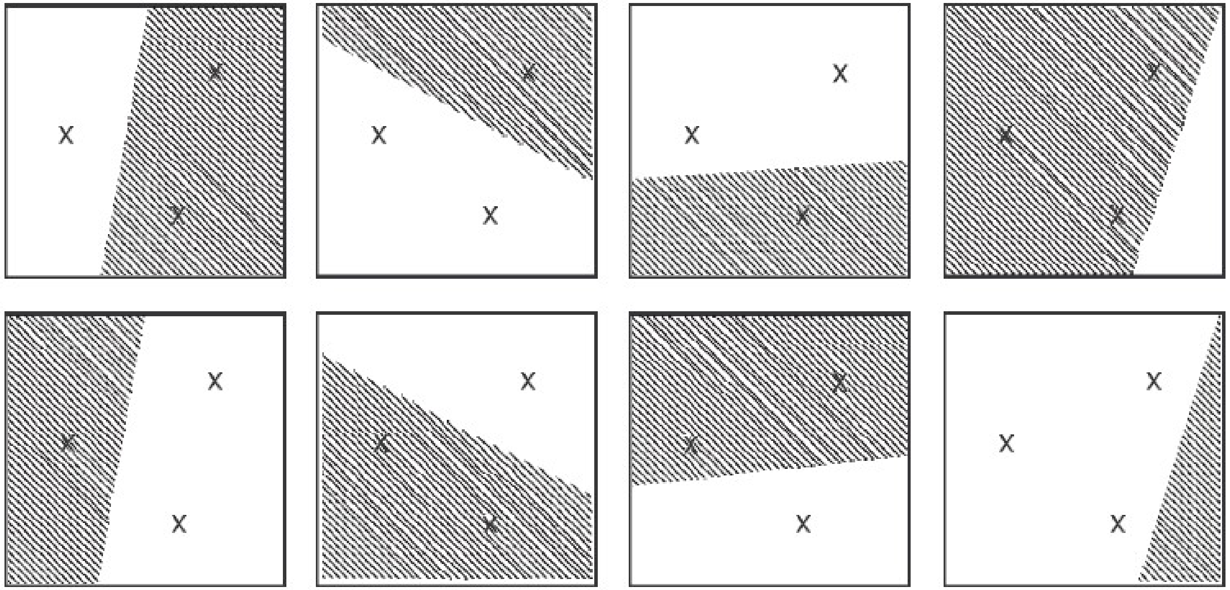


Figure 1.4 A simple VC dimension example. There are $2^3 = 8$ ways of assigning 3 points to two classes. For the displayed points in $\mathbb{R}^2$, all 8 possibilities can be realized using separating hyperplanes, in other words, the function class can shatter 3 points. This would not work if we were given 4 points, no matter how we placed them. Therefore, the VC dimension of the class of separating hyperplanes in $\mathbb{R}^2$ is 3.

Cover's Theorem

- Gives the number of possible linear separations of N points, in general position, in a d-dimensional space
- If $N \leq d+1$ then 2^N separations are possible $VC \dim = d+1$
- If $N > d+1$, the number of linear separations is

$$2\sum_{i=0}^{d}\binom{N-1}{i}$$

- As we increase d, there are more terms in the sum, VC$\nearrow$
- Points assumed in general position: however in practical applications points could be on lower-dimensional manifold

Large Margin Linear Classifier

- Given a set of data points: $\{(\mathbf{x}_i, y_i)\},\ i = 1, 2, \cdots, n$, where

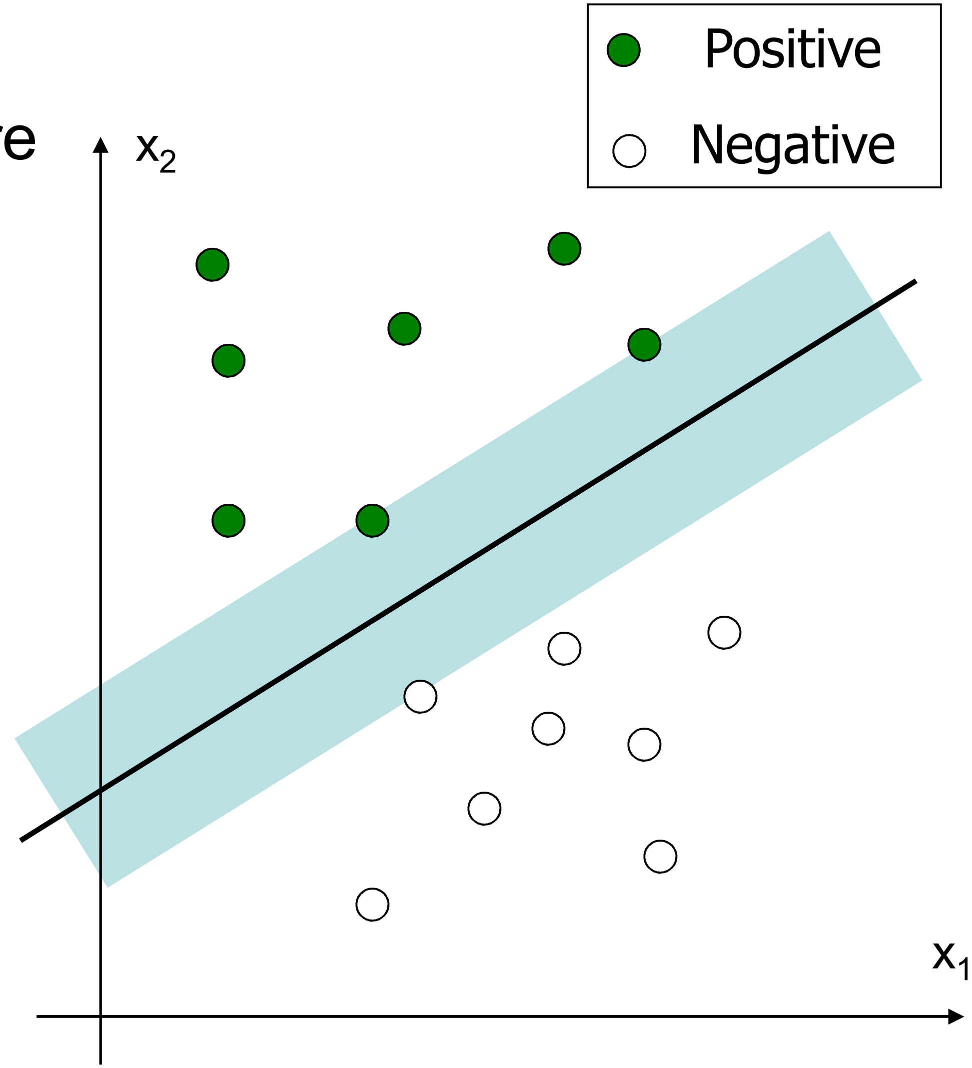
$$\text{For } y_i = +1,\quad \mathbf{w}^T\mathbf{x}_i + b > 0$$

$$\text{For } y_i = -1,\quad \mathbf{w}^T\mathbf{x}_i + b < 0$$

- **Cannonical Hyperplane**
Under a scale transformation on both $\mathbf{w}$ and b, we can remove gauge in the above

$$\text{For } y_i = +1,\quad \mathbf{w}^T\mathbf{x}_i + b \geq 1$$

$$\text{For } y_i = -1,\quad \mathbf{w}^T\mathbf{x}_i + b \leq -1$$



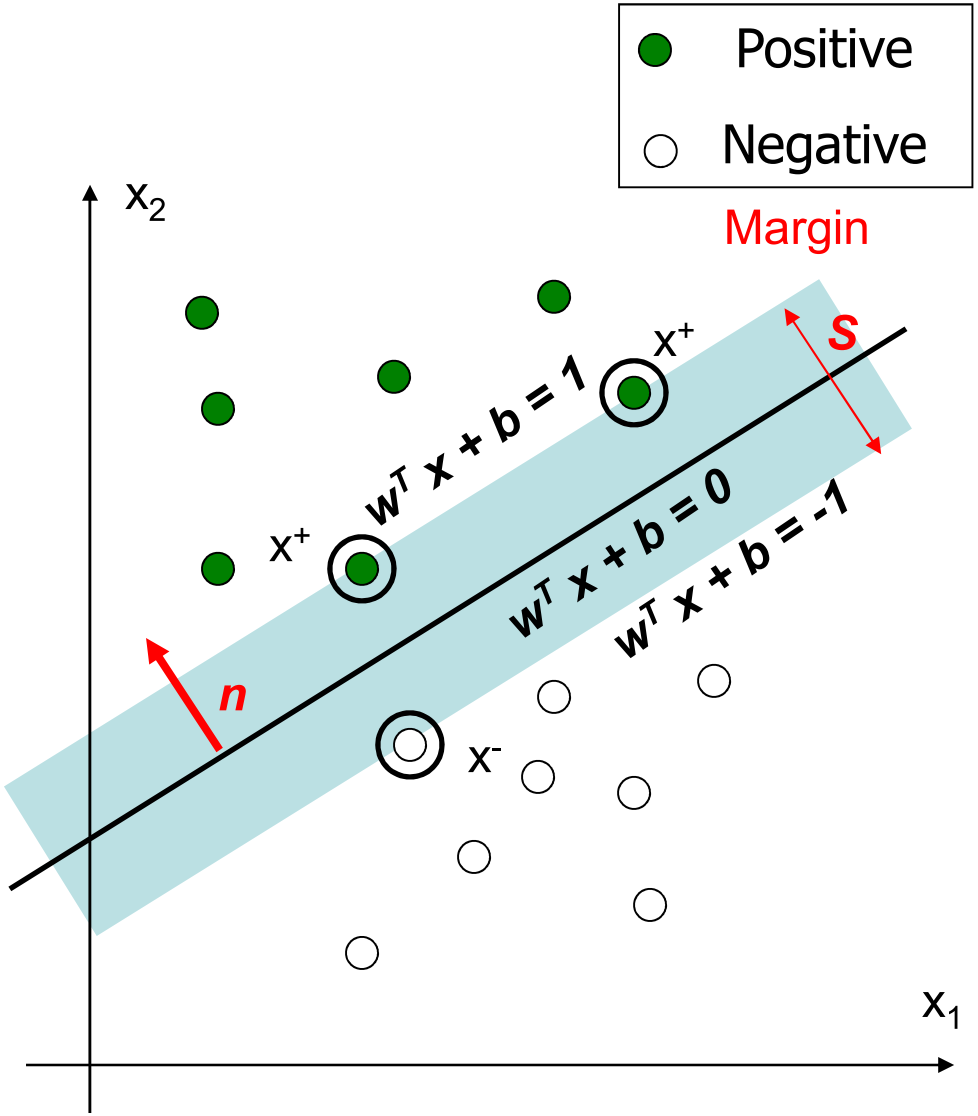
Large Margin Linear Classifier

- We know that

$$\mathrm{w}^T \mathrm{x}^+ + b = 1$$
$$\mathrm{w}^T \mathrm{x}^- + b = -1$$

- The separation is

$$S = \mathrm{n}^T \cdot (\mathrm{x}^+ - \mathrm{x}^-) =$$
$$= \frac{\mathrm{w}^T}{\|\mathrm{w}\|} \cdot (\mathrm{x}^+ - \mathrm{x}^-) = \frac{2}{\|\mathrm{w}\|}$$



Large Margin Linear Classifier

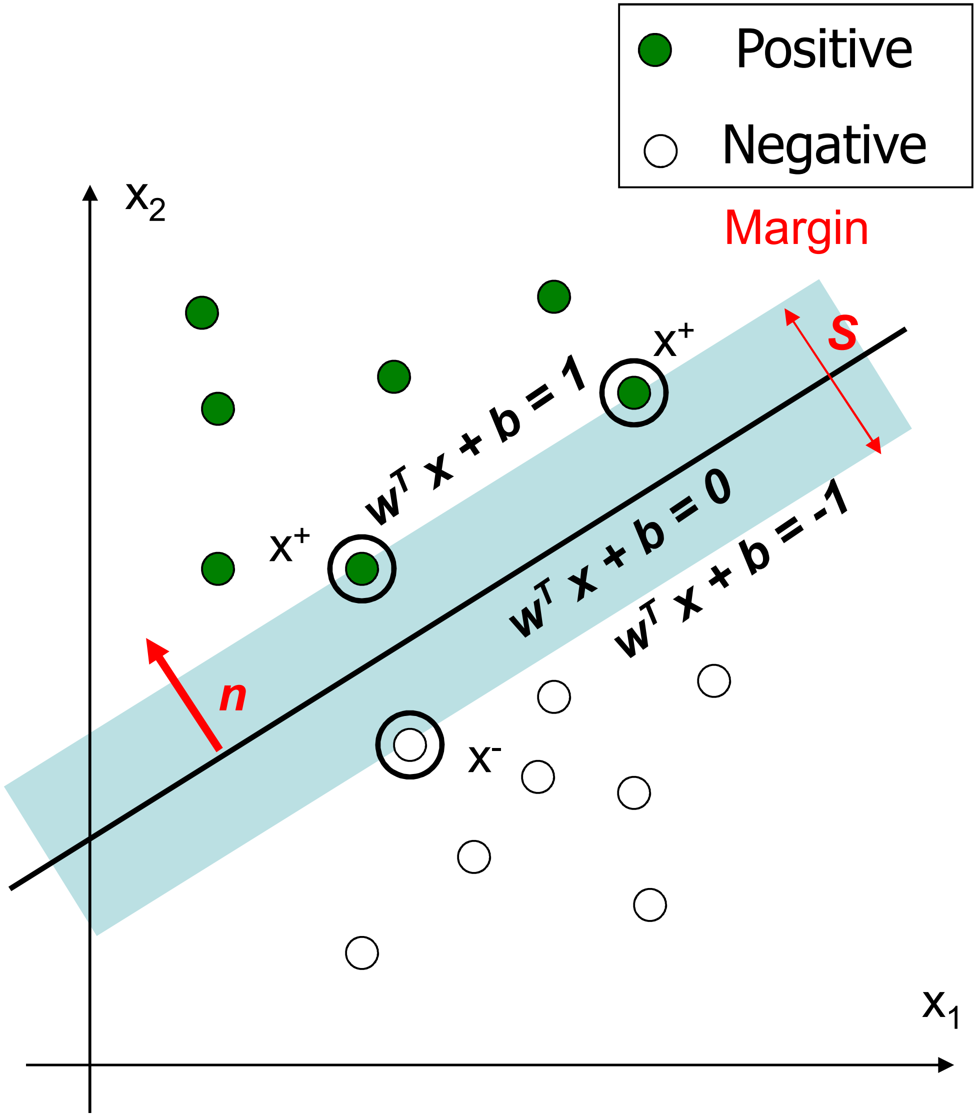
- Formulation:

$$\max_{\mathbf{w},b} \quad \frac{2}{\|\mathbf{w}\|}$$

such that

$$\text{For } y_i = +1, \quad \mathbf{w}^T \mathbf{x}_i + b \geq 1$$

$$\text{For } y_i = -1, \quad \mathbf{w}^T \mathbf{x}_i + b \leq -1$$



Large Margin Linear Classifier

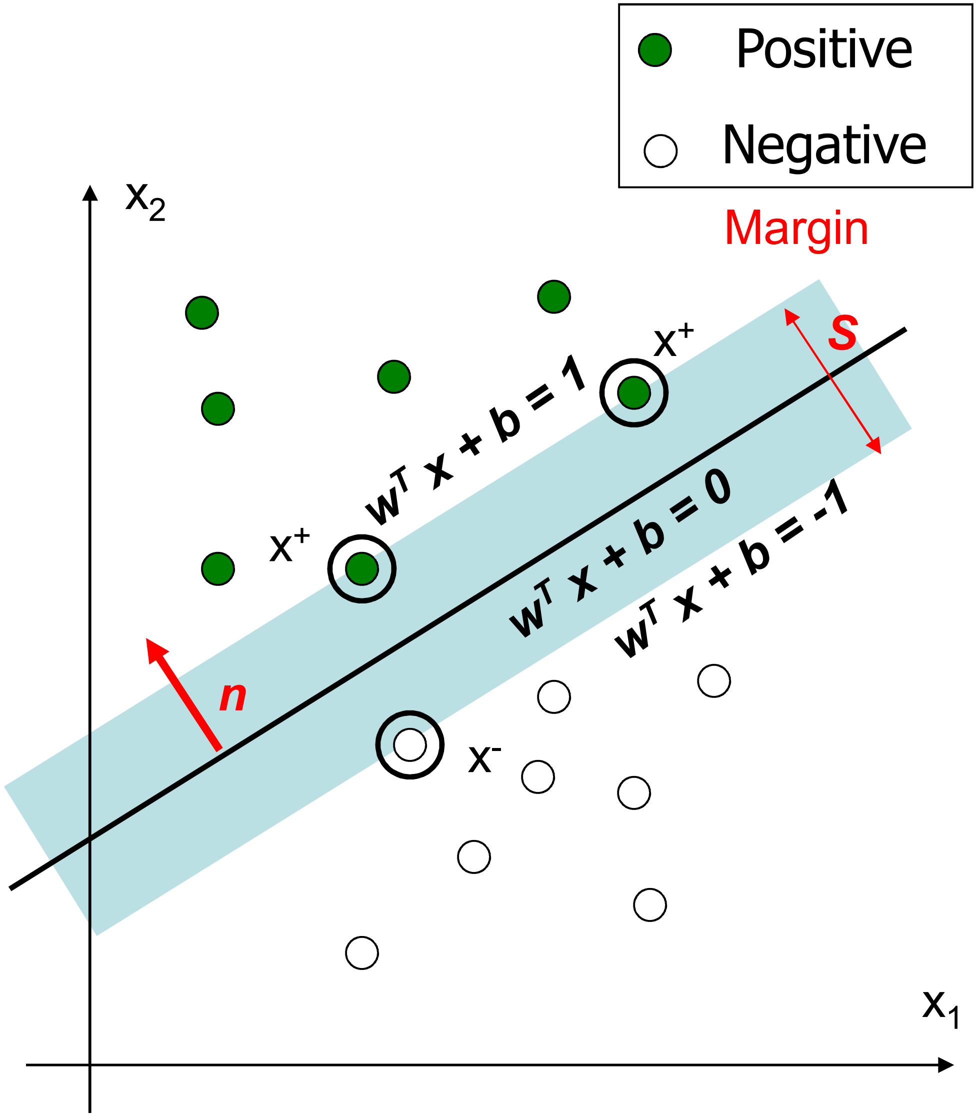
- Formulation:

$$\min_{\mathbf{w},b} \quad \frac{1}{2}\|\mathbf{w}\|^2$$

such that

$$\text{For } y_i = +1, \quad \mathbf{w}^T\mathbf{x}_i + b \geq 1$$

$$\text{For } y_i = -1, \quad \mathbf{w}^T\mathbf{x}_i + b \leq -1$$



Large Margin Linear Classifier

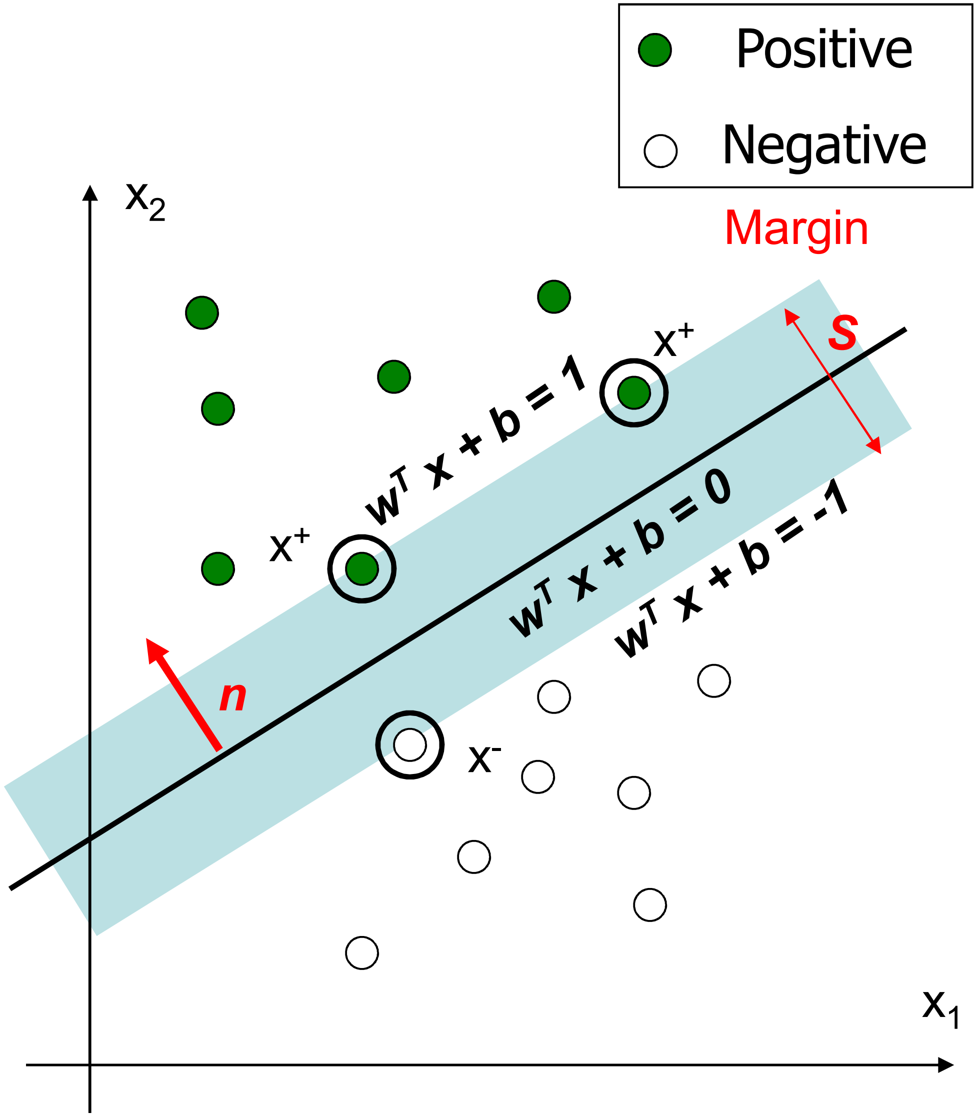
- Formulation:

$$\min_{\mathbf{w},b} \quad \frac{1}{2}\|\mathbf{w}\|^2$$

such that

$$y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1$$

- Quadratic program with linear constraints



Solving the Optimization Problem

Quadratic programming with linear constraints

$$\min_{\mathbf{w},b} \quad \frac{1}{2}\|\mathbf{w}\|^2$$

$$\text{s.t.} \quad y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1$$

$\Updownarrow$

Lagrangian Function

$$\min_{\mathbf{w},b} \max_{\boldsymbol{\alpha}} \ L_p(\mathbf{w}, b, \alpha_i) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{n} \alpha_i \left(y_i(\mathbf{w}^T\mathbf{x}_i + b) - 1 \right)$$

$$\text{s.t.} \quad \alpha_i \geq 0$$

The Lagrangian needs to be minimized w.r.t. $\mathbf{w}, b$, and maximized w.r.t α_i

Solving the Optimization Problem

$$\min_{\mathbf{w},b} \max_{\boldsymbol{\alpha}} \; L_p(\mathbf{w}, b, \alpha_i) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{n} \alpha_i \left(y_i(\mathbf{w}^T \mathbf{x}_i + b) - 1 \right)$$

$$\text{s.t.} \quad \alpha_i \geq 0$$

$$\frac{\partial L_p}{\partial \mathbf{w}} = 0 \quad \Longrightarrow \quad \mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i$$

Solution is an expansion in terms of training examples

$$\frac{\partial L_p}{\partial b} = 0 \quad \Longrightarrow \quad \sum_{i=1}^{n} \alpha_i y_i = 0$$

Due to strict convexity, $\mathbf{w}$ is unique although $\alpha_i's$ need not be

Solving the Optimization Problem

$$\min_{\mathbf{w},b} \max_{\boldsymbol{\alpha}} \; L_p(\mathbf{w}, b, \alpha_i) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{n} \alpha_i \left(y_i(\mathbf{w}^T \mathbf{x}_i + b) - 1 \right)$$

s.t. $\quad \alpha_i \geq 0$

Lagrangian Dual Problem

maximize

$$\max_{\boldsymbol{\alpha}} \; \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$$

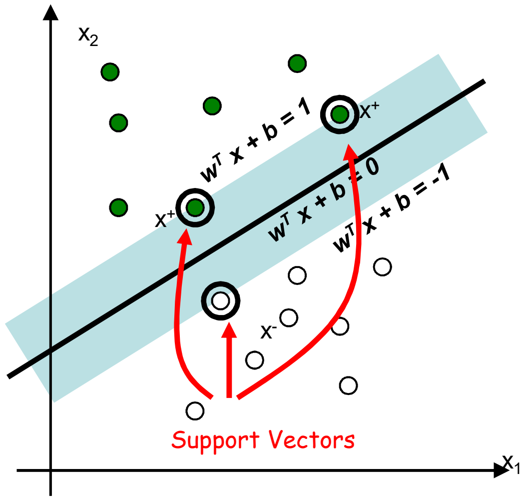
s.t. $\quad \alpha_i \geq 0$, and $\quad \sum_{i=1}^{n} \alpha_i y_i = 0$

Convex quadratic optimization problem. Using a QP solver, gives us the uniquely best α_i.

Solving the Optimization Problem

- Suppose we have found the optimal α
- From the KKT conditions, we know:

$$\alpha_i \left(y_i(\mathbf{w}^T \mathbf{x}_i + b) - 1 \right) = 0$$



- Thus, only support vectors have $\alpha_i \neq 0$

- The solution has the form:

$$\mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i = \sum_{i \in \mathrm{SV}} \alpha_i y_i \mathbf{x}_i$$

We can get b from $y_i(\mathbf{w}^T \mathbf{x}_i + b) - 1 = 0$

where $\mathbf{x}_i$ is any support vector

Solving for b

For support vectors

$$y_s(\mathbf{w}^\top \cdot \mathbf{x}_s + b) = 1$$

$$\mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i$$

$$y_s\left(\sum_{i\in S} \boldsymbol{\alpha_i y_i}\, \mathbf{x}_i^\top \cdot \mathbf{x}_s + b\right) = 1 \mid \times\ y_s$$

$$b = \frac{1}{N_{SV}} \sum_{i=1}^{N_{SV}} (y_s - \sum_{i\in S} \boldsymbol{\alpha_i y_i}\, \mathbf{x}_i^\top \cdot \mathbf{x}_s)$$

Better take an average over support vectors

Solving the Optimization Problem

- The linear discriminant function is:

$$g(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b = \sum_{i \in SV} \alpha_i y_i \mathbf{x}_i^T \mathbf{x} + b$$

- Relies on a *dot product* between the test point $\mathbf{x}$ and the support vectors $\mathbf{x}_i$
- Solving the optimization problem involved computing the **dot products**

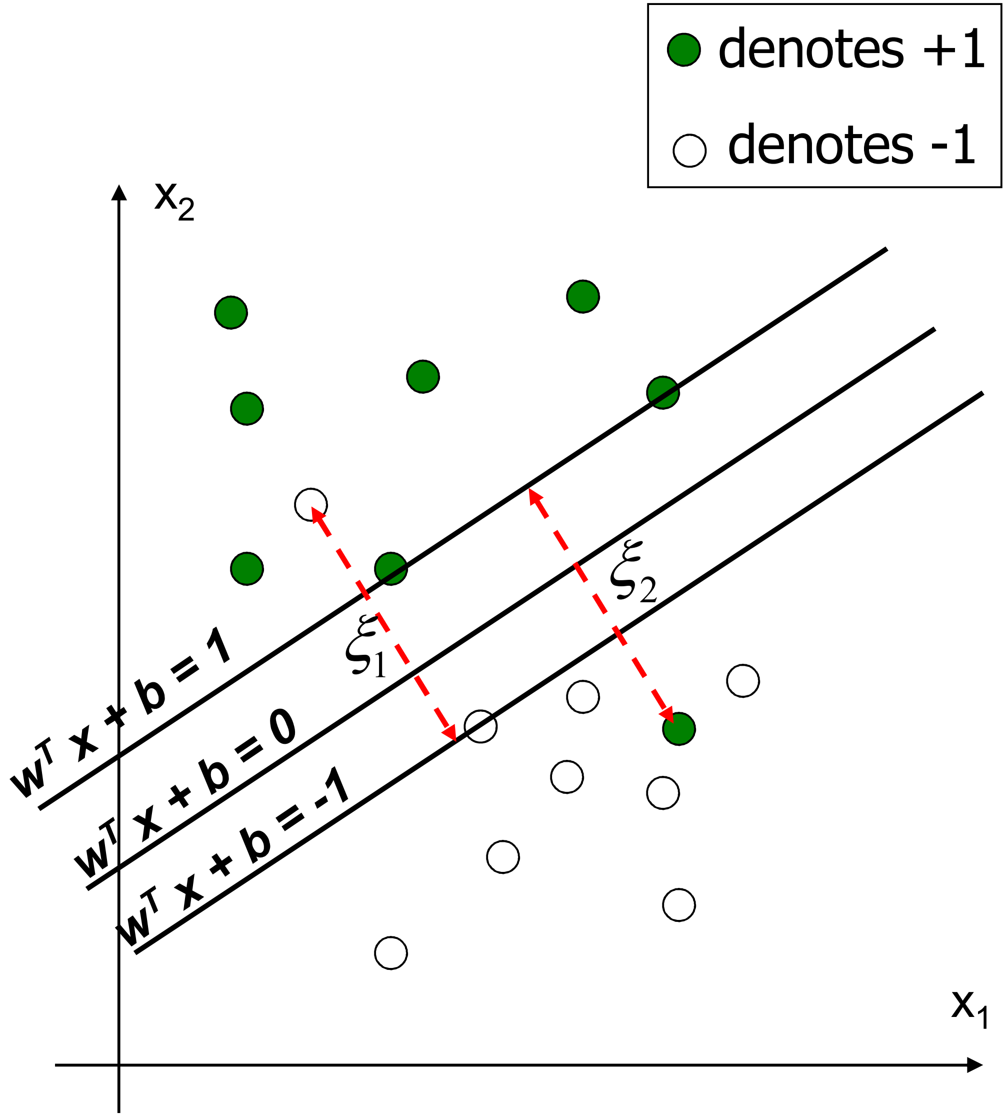
between all pairs of training points

- Negative side: with many points, this is expensive
- Positive side: The algorithm and solution only needs this matrix of products from the training points, not the points itself. We will take advantage of that.

`Soft Margin' Linear Classifier

- What if data is not linear separable due to noise or outliers?

- Slack variables ξ_i can be added to allow for the mis-classification of difficult or noisy data



`Soft Margin' Linear Classifier

- Formulation:

$$\min_{\mathbf{w},b,\boldsymbol{\xi}} \quad \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}\xi_i$$

such that

$$y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 - \xi_i$$

$$\xi_i \geq 0$$

- for $0 \leq \xi \leq 1$, point is between margin and correct side of hyperplane
- for $\xi > 1$, point is misclassified

- Parameter C can be viewed as a means to control over-fitting
  - small C allows constraints to be easily ignored: *large margin*
  - large C makes constraints hard to ignore: *narrow margin*
  - $C = \infty$ enforces all constraints: *hard margin*

`Soft Margin' Linear Classifier

- Formulation (Lagrangian Dual Problem)

$$\max_{\boldsymbol{\alpha}} \quad \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$$

such that

$$0 \le \alpha_i \le C$$

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

`Soft Margin' Interpretation (I)

- The constraint $y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 - \xi_i$ can be written more concisely as

$$y_i g(\mathbf{x}_i) \geq 1 - \xi_i \Leftrightarrow \xi_i = \max(0, 1 - y_i g(\mathbf{x}_i))$$

- Hence we need to solve the learning problem

$$\min_{\mathbf{w},b} ||\mathbf{w}||^2 + C \sum_{i=1}^{N} \max(0, 1 - y_i g(\mathbf{x}_i))$$

`Soft Margin' Interpretation (II)

We need to solve the learning problem

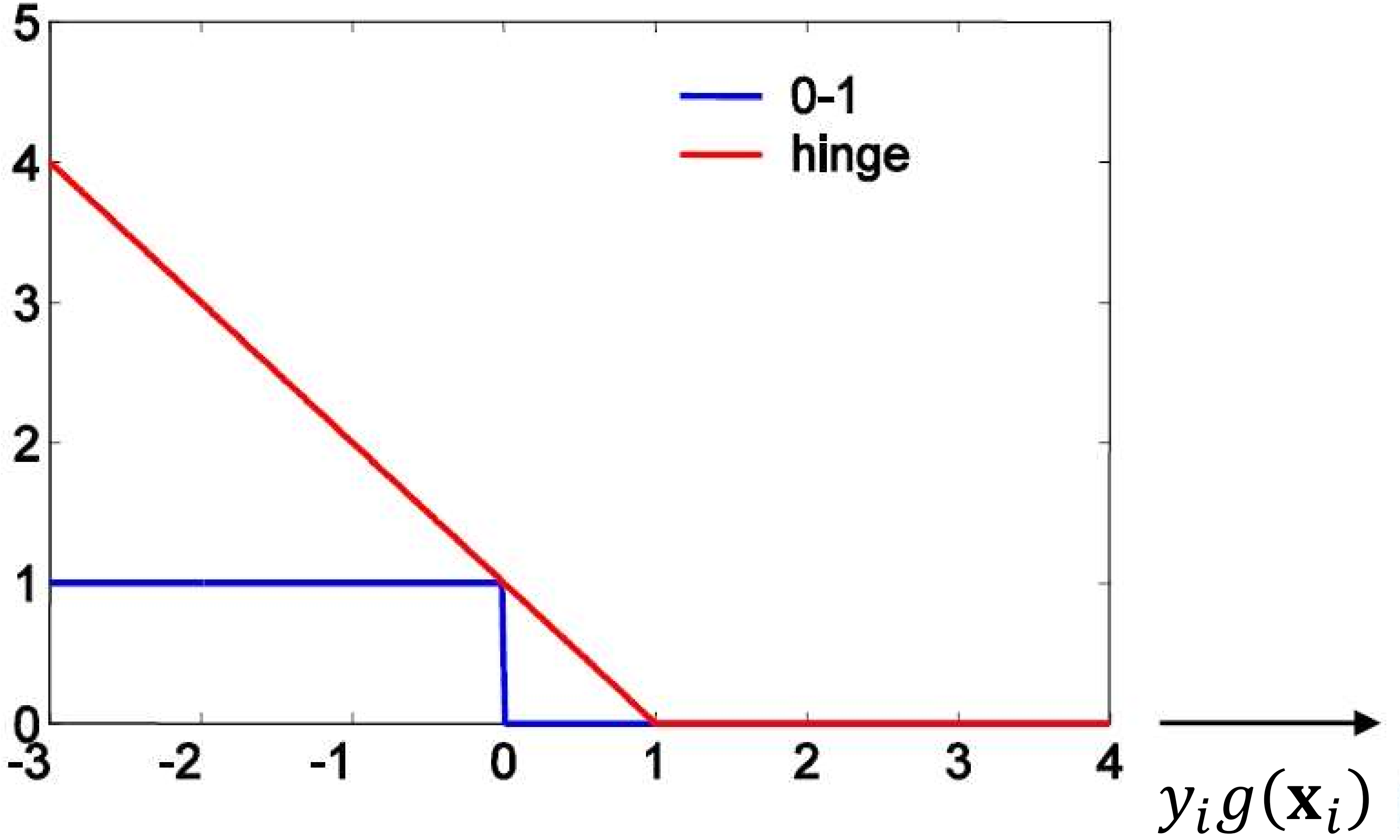
$$\min_{\mathbf{w},b} ||\mathbf{w}||^2 + C \sum_{i=1}^{N} \max(0, 1 - y_i g(\mathbf{x}_i))$$

- $y_i g(\mathbf{x}_i) > 1 \Rightarrow$ point is outside margin and does not contribute to loss

- $y_i g(\mathbf{x}_i) = 1 \Rightarrow$ point is on margin and does not contribute to loss (as in hard margin)

- $y_i g(\mathbf{x}_i) < 1 \Rightarrow$ point violates margin constraint and contributes to loss

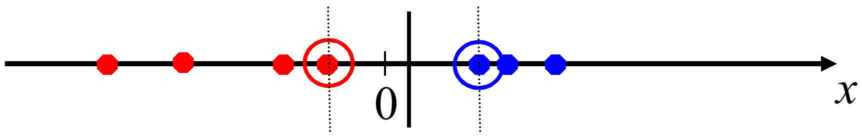
SVM uses Hinge Loss



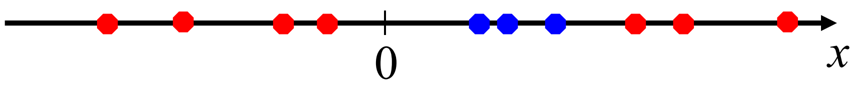
Can be viewed as an approximation to the 0-1 loss

Non-linear SVMs

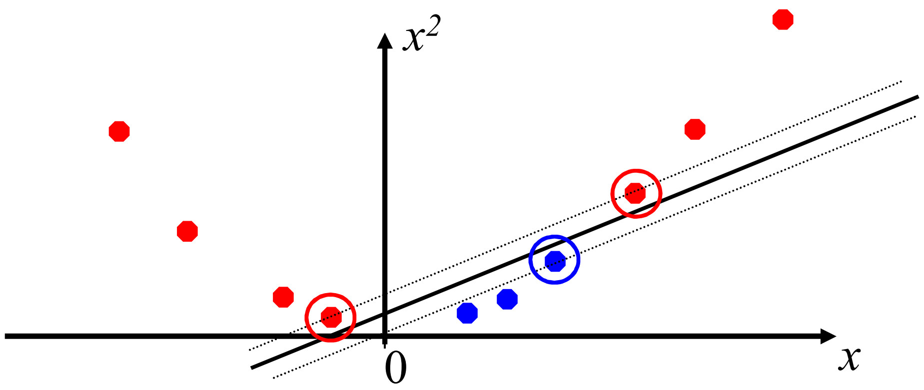
- Datasets that are linearly separable with noise work out great:



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

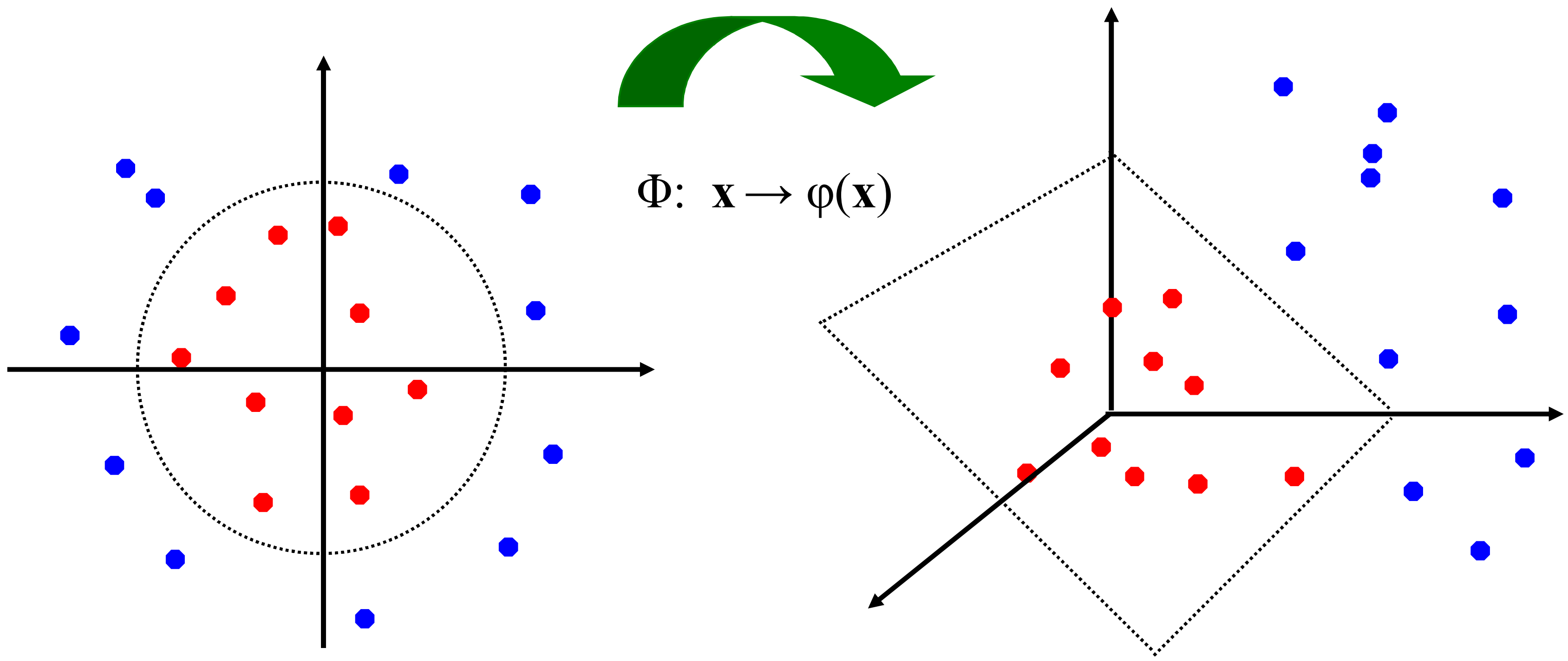


- How about… mapping data to a higher-dimensional space:



Non-linear SVMs: Feature Space

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



How to Use the Feature Space?

- The feature point $\mathbf{z} = \phi(\mathbf{x})$ corresponding to an input point $\mathbf{x}$ is called the image (or the lifting) of $\mathbf{x}$; the input point $\mathbf{x}$, if any, corresponding to a given feature vector $\mathbf{z}$ is called the pre-image of $\mathbf{z}$

- The naive way to use a feature space is to explicitly compute the image of every training and testing point, and run algorithm fully in feature space

- Two potential problems
  - The feature space may be very high dimensional or infinite dimensional, so direct (explicit) calculations in such feature space may not be practical, or even possible
  - We may sometimes want to map back an answer from feature space to the input space. This is called the pre-image problem. For some kernels, analytical expressions are available, but in most other cases some form of (local) optimization may be necessary

Nonlinear SVMs: The Kernel Trick

- With this mapping, our discriminant function is now:

$$g(\mathbf{x}) = \mathbf{w}^T \phi(\mathbf{x}) + b = \sum_{i \in \mathrm{SV}} \alpha_i \phi(\mathbf{x}_i)^T \phi(\mathbf{x}) + b$$

Nonlinear SVMs: The Kernel Trick

- With this mapping, our discriminant function is now:

$$g(\mathbf{x}) = \mathbf{w}^T \phi(\mathbf{x}) + b = \sum_{i \in \mathrm{SV}} \alpha_i \boxed{\phi(\mathbf{x}_i)^T \phi(\mathbf{x})} + b$$

- No need to know this mapping explicitly, because we only use the *dot product* of feature vectors both in training and in testing

Nonlinear SVMs: The Kernel Trick

- With this mapping, our discriminant function is now:

$$g(\mathbf{x}) = \mathbf{w}^T \phi(\mathbf{x}) + b = \sum_{i \in \mathrm{SV}} \alpha_i \boxed{\phi(\mathbf{x}_i)^T \phi(\mathbf{x})} + b$$

- No need to know this mapping explicitly, because we only use the *dot product* of feature vectors both in training and in testing

- A *kernel function* is defined as a function that corresponds to a dot product of two feature vectors in some expanded feature space:

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

- Instead of specifying and computing features, can define and compute kernel only.

Positive Definite Kernels

- **Gram Matrix.** Given a function $k: \mathrm{X}^2 \rightarrow \mathbf{R}$ (or $\mathbf{C}$), and patterns $x_1, \dots, x_m \in \mathrm{X}$, the $m \times m$ matrix K with elements $K_{ij} \coloneqq k(x_i, x_j)$ is called the Gram matrix (or kernel matrix) of k w.r.t $x_1, \dots, x_m$.

- **Positive definite kernel.** A complex $m \times m$ matrix K satisfying $\sum_{ij} c_i \overline{c_j} K_{ij} \geq 0$, $\forall\, c_i \in \mathbf{C}$ is called positive definite. Similarly, a real symmetric $m \times m$ matrix K satisfying the above for all $c_i \in \mathbf{R}$ is called positive definite.

positive definite kernels ≡Mercer kernels ≡reproducing kernels ≡admissible kernels ≡ support vector kernels ≡covariance functions

Examples of Kernels

Examples of commonly-used kernel functions:

- Linear kernel: $K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T \mathbf{x}_j$
- Polynomial kernel: $K(\mathbf{x}_i, \mathbf{x}_j) = (1 + \mathbf{x}_i^T \mathbf{x}_j)^p$
- Gaussian (Radial-Basis Function (RBF)) kernel:

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

- Sigmoid:

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

Generality of Kernel Trick

- Given an algorithm expressed in terms of a positive-definite kernel k, we can construct an alternative algorithm by replacing k with another positive-definite kernel $\tilde{k}$
- This is not limited to only cases when k is a dot product in the input domain
- Any algorithm that only depends on dot products (i.e. is rotationally invariant) can be kernelized
- Kernels are defined on general sets (rather than just dot product spaces!) and their use leads to an embedding of general data types in linear spaces

Nonlinear SVM: Optimization

- Formulation (Lagrangian Dual Problem)

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

such that

$$0 \le \alpha_i \le C$$

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

- The solution of the discriminant function is

$$g(\mathbf{x}) = \sum_{i \in SV} \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) + b$$

Support Vector Machine: Algorithm

1. Choose a kernel function

2. Choose a value for C

3. Solve the quadratic programming problem
   (many software packages available, e.g. libsvm)

4. Construct the discriminant function from the support vectors

Support Vector Machines

- A support vector machine (SVM) is nothing more than a kernelized maximum-margin hyperplane classifier
- You train it by solving the dual quadratic programming problem
- You run it by evaluating the kernel function between the test point and each of the “active” training points, called support vectors
- This combination of (1) kernel trick, (2) maximum margin (minimum norm) and (3) the resulting sparsity has turned out to be very effective and popular
- In practice, the hard part from a learning point of view is selecting the kernel function (there is a lot of research on this) and from a computational point of view it is solving the large QP efficiently (most popular method: Sequential Minimal Optimization, estimates pairs of parameters sequentially)

SVM Applet Demo

https://cs.stanford.edu/people/karpathy/svmjs/demo/

Properties of Kernels

- Kernels are symmetric in their arguments:

$$K(\mathbf{x}_1, \mathbf{x}_2) \ = \ K(\mathbf{x}_2, \mathbf{x}_1)$$

- They are positive valued for any inputs: $K(\mathbf{x}_1, \mathbf{x}_2) \geq \ 0$

- The Cauchy-Schwartz inequality holds:

$$K^2(\mathbf{x}_1, \mathbf{x}_2) \ \leq \ K(\mathbf{x}_1, \mathbf{x}_1)K(\mathbf{x}_2, \mathbf{x}_2)$$

- Technically, to use a function as a kernel, it must satisfy Mercer's conditions for a positive-definite operator

- The intuition is easy to grasp for finite spaces
  - Discretize $\mathbf{x}$ space as densely as desired into buckets $\mathbf{x}_i$
  - Between each two cells $\mathbf{x}_i, \mathbf{x}_j$, compute the kernel function, and write these values as a (symmetric) matrix $M_{ij} \ = \ K(\mathbf{x}_i, \mathbf{x}_j)$
  - If the matrix is positive definite, the kernel is OK

Kernel Closure Rules

Very useful for designing new kernels from existing kernels

– The sum of any two kernels is a kernel

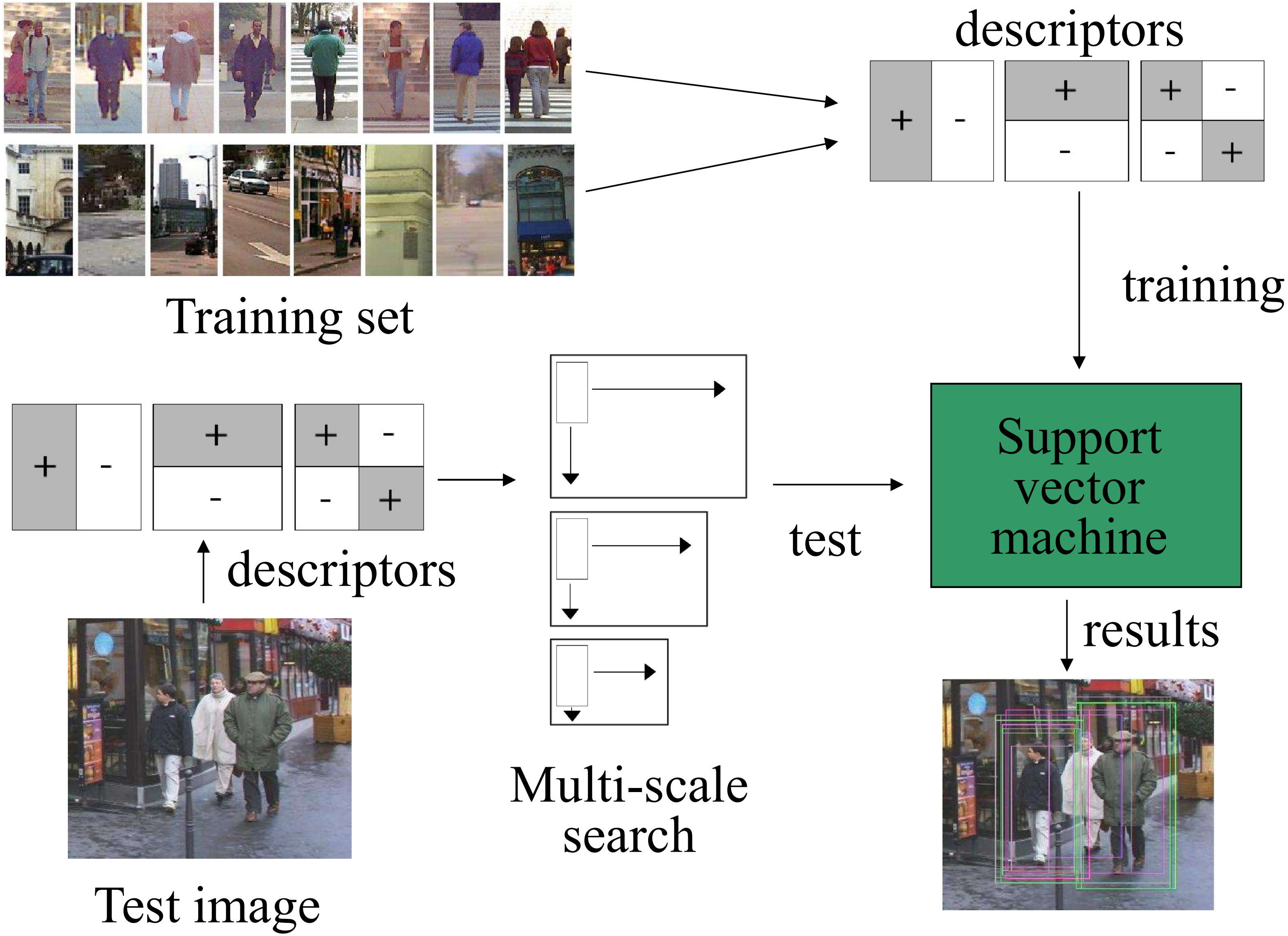
– The product of any two kernels is a kernel

– A kernel plus a constant is a kernel

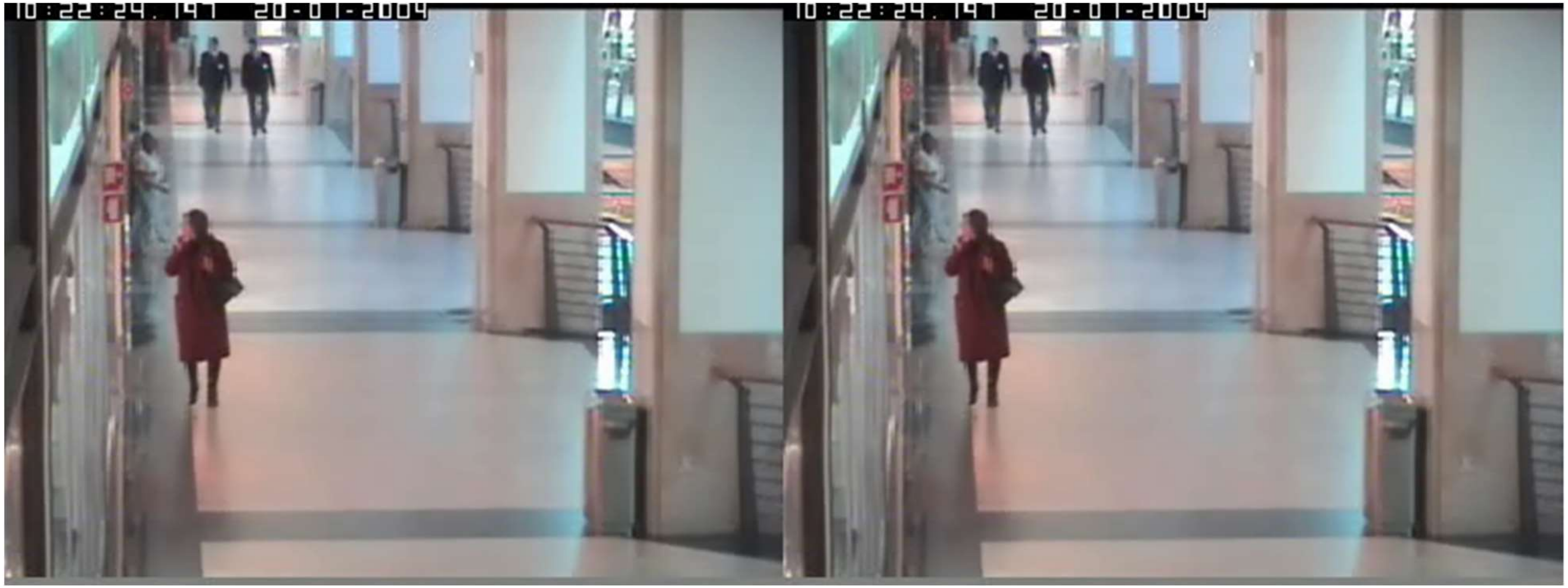
– A scalar times a kernel is a kernel

Support Vector Machine Detector

Papageorgiou, Poggio



Video: Pedestrian Detection



Scalability Issues

Although we circumvented infinite dimensionality,

$$f(\mathrm{x}) = \sum_i \alpha_i K(\mathrm{x}, \mathrm{x_i})$$

- In training:
  - # optimization variables = # training examples *N*
- In testing:
  - Need to evaluate kernel between test data and each training example

- Training and testing for millions of examples unfeasible
  - e.g. in ImageNet need to classify 9 million images

Linear versus Kernel Methods

	Linear	Kernel
Model	$f(x) = w^T x$	$f(x) = \sum_i \alpha_i k(x, x_i)$
Number of optimization variables	Input dimensionality d	# training examples N
Training time	O(Nd^2)	O(N^2d) ~ O(N^3d)
Testing time	O(d)	O(Nd)
Caltech-101 Accuracy (BOW feature)	49% (Vedaldi and Zisserman 2010)	64% (Vedaldi and Zisserman 2010)
Caltech-101 Accuracy (multiple kernels)	N/A	82% (Gehler & Nowozin 2009) (Li et al. 2010)

Good things are worth doing slowly