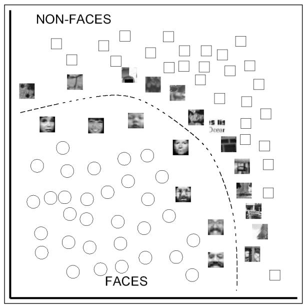
CSE 473

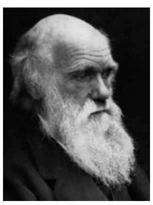
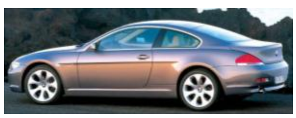
Lecture 25
(Chapter 18)

Linear Classification, SVMs and Nearest Neighbors



Motivation: Face Detection

How do we build a classifier to distinguish between faces and other objects?

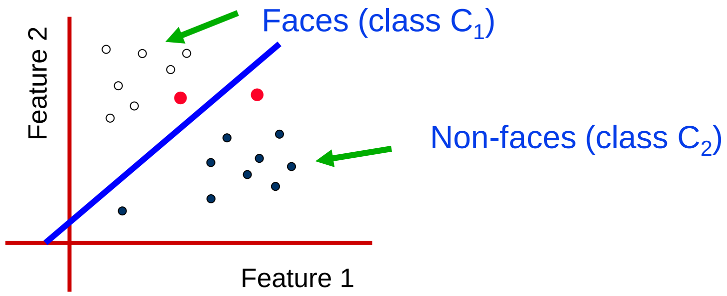






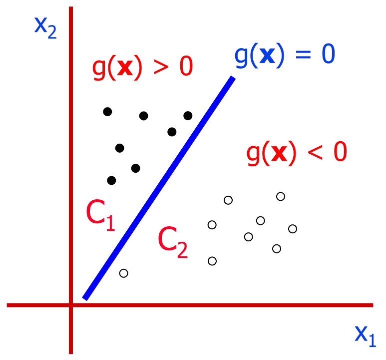


Binary Classification: Example



How do we classify new data points?

Binary Classification: Linear Classifiers



Find a line (in general, a hyperplane) separating the two sets of data points:

$g(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} + b = 0$, i.e.,

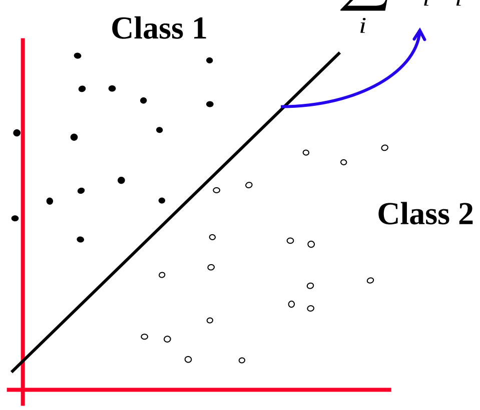
$w_1 x_1 + w_2 x_2 + b = 0$

For any new point **x**, choose:

class C_1 if $g(\mathbf{x}) > 0$ and class C_2 otherwise

Separating Hyperplane

$$\sum_i w_i x_i + b = 0$$

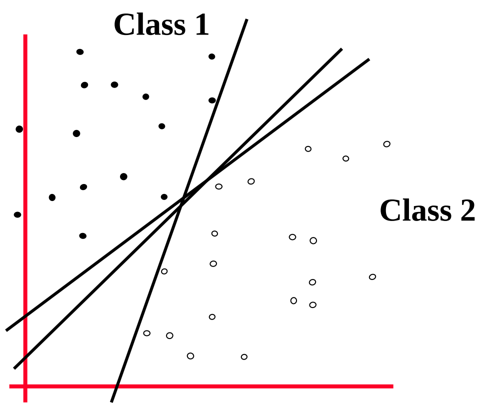


- denotes +1 output
- ◦ denotes -1 output

Need to choose w_i and b based on training data

Separating Hyperplanes

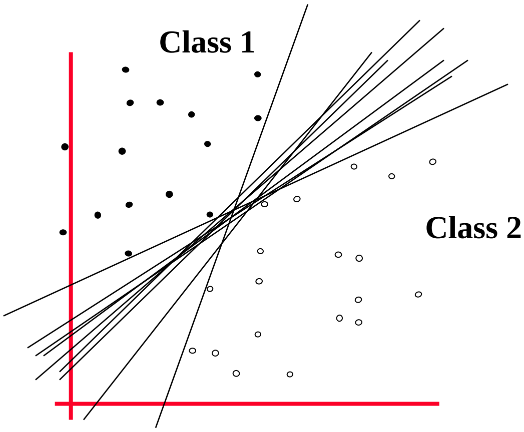
Different choices of w_i and b give different hyperplanes



- denotes +1 output
- ◦ denotes -1 output

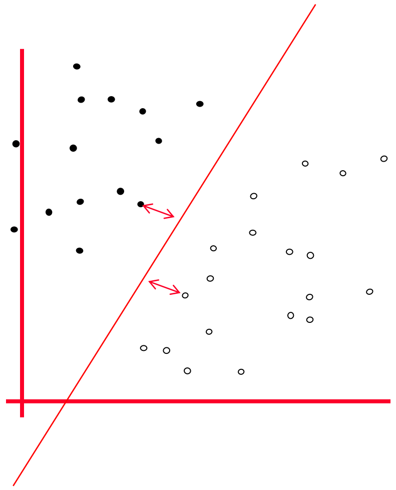
(This and next few slides adapted from Andrew Moore's)

Which hyperplane is best?



- • denotes +1 output
- ◦ denotes -1 output

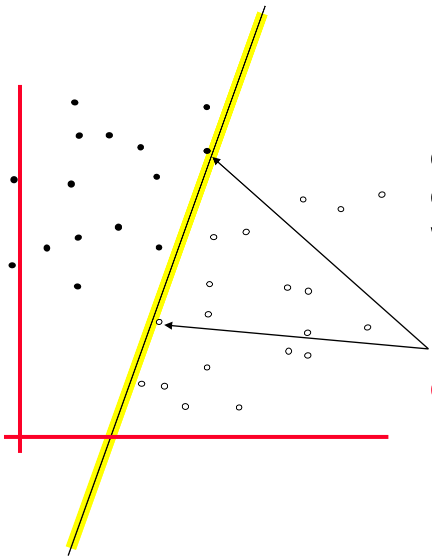
How about the one right in the middle?



Intuitively, this boundary seems good

Avoids misclassification of new test points if they are generated from the same distribution as training points

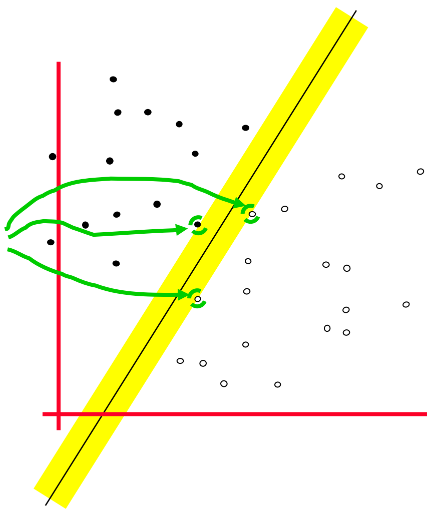
Margin



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

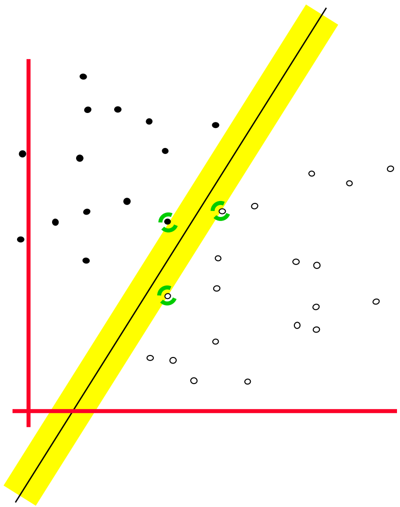
Maximum Margin and Support Vector Machine

Support Vectors are those datapoints that the margin pushes up against



The maximum margin classifier is called a Support Vector Machine (in this case, a Linear SVM or LSVM)

Why Maximum Margin?



- Robust to small perturbations of data points near boundary
- There exists theory showing this is best for generalization to new points
- Empirically works great

11

Finding the Maximum Margin

(For Math Lovers Eyes Only)

Can show that we need to maximize:

$$2/\|\mathbf{w}\| \text{ subject to } y_i(\mathbf{w}\cdot\mathbf{x}_i + b) \geq +1, \forall i$$

Margin ↗

Constrained optimization problem that leads to:

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

where the α_i are obtained by maximizing:

$$\sum_i \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j (\mathbf{x}_i \cdot \mathbf{x}_j)$$ ← Depends on *dot product* of inputs

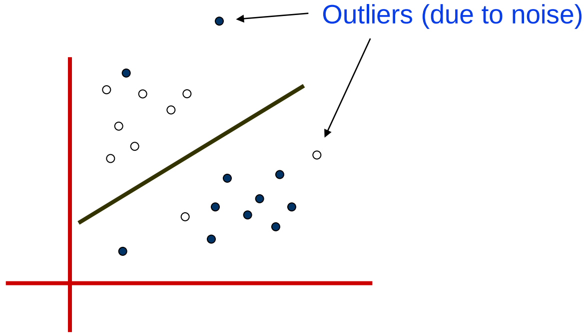
$$\text{subject to } \alpha_i \geq 0 \text{ and } \sum_i \alpha_i y_i = 0$$

Quadratic programming (QP) problem
- A global maximum can always be found

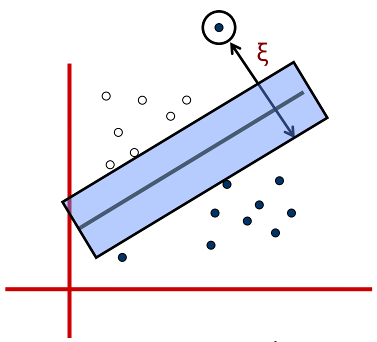
(Interested in more details? see Burges' SVM tutorial online)

12

What if data is not linearly separable?



Soft Margin SVMs

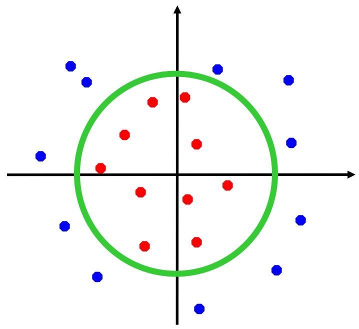


Allow *errors* ξ_i (deviations from margin)

Trade off margin with errors

Minimize: $$\frac{1}{2}\|\mathbf{w}\|^2 + C\sum_i \xi_i \text{ subject to:}$$
$$y_i(\mathbf{w}\cdot\mathbf{x}_i + b) \geq 1 - \xi_i \quad \text{and } \xi_i \geq 0, \forall i$$

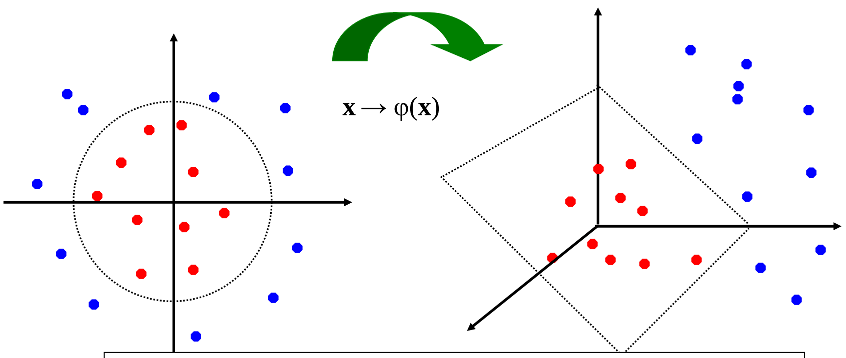
Another Example



Not linearly separable

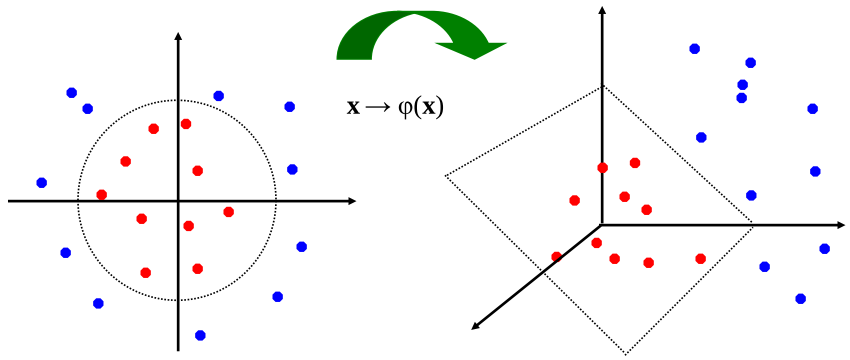
Handling non-linearly separable data

Idea: Map original input space to higher-dimensional feature space; use linear classifier in higher-dim. space



$$\Phi : \mathbb{R}^2 \rightarrow \mathbb{R}^3$$
$$(x_1, x_2) \mapsto (z_1, z_2, z_3) := (x_1^2, \sqrt{2}\, x_1 x_2, x_2^2)$$

Problem: High dimensional spaces



Computation in high-dimensional feature space is costly

The high dimensional projection function φ(x) may be too complicated to compute

Kernel trick to the rescue!

The Kernel Trick

Recall: SVM maximizes the quadratic function:

$$\sum_i \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j (\mathbf{x}_i \cdot \mathbf{x}_j)$$

subject to $\alpha_i \geq 0$ and $\sum_i \alpha_i y_i = 0$

Insight:

The data points only appear as dot product

- No need to compute high-dimensional φ(x) explicitly! Just replace inner product $x_i \cdot x_j$ with a "kernel" function $K(x_i,x_j) = \varphi(x_i) \cdot \varphi(x_j)$
- E.g., Gaussian kernel
  $K(x_i,x_j) = \exp(-||x_i-x_j||^2/2\sigma^2)$
- E.g., Polynomial kernel
  $K(x_i,x_j) = (x_i \cdot x_j+1)^d$

Example of the Kernel Trick

Suppose $\phi(.)$ is given as follows:

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

Dot product in the feature space is

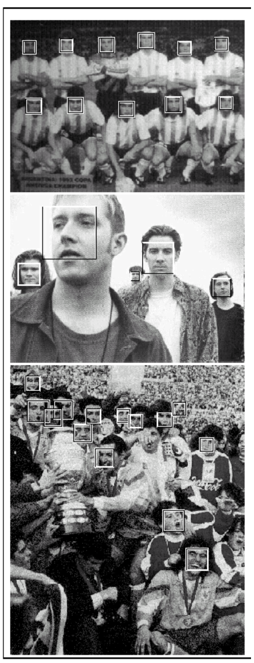
$$\langle \phi(\left[\begin{smallmatrix} x_1 \\ x_2 \end{smallmatrix}\right]), \phi(\left[\begin{smallmatrix} y_1 \\ y_2 \end{smallmatrix}\right]) \rangle = (1 + x_1y_1 + x_2y_2)^2$$

So, if we define the kernel function as follows, there is no need to compute $\phi(.)$ explicitly

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

Use of kernel function to avoid computing $\phi(.)$ explicitly is known as the **kernel trick**

Face Detection using SVMs

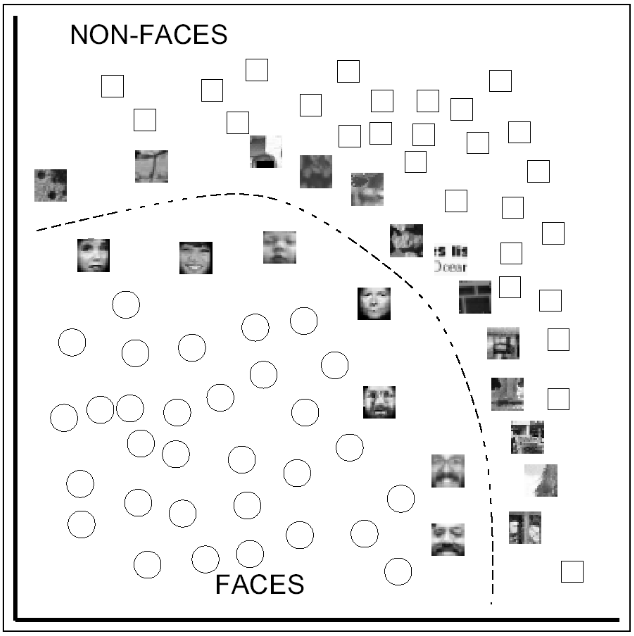


	Test Set A		Test Set B	
	Detect Rate	False Alarms	Detect Rate	False Alarms
SVM	97.1 %	4	74.2%	20
Sung *et al.*	94.6 %	2	74.2%	11

Kernel used: Polynomial of degree 2

(Osuna, Freund, Girosi, 1998)

Support Vectors



21

K-Nearest Neighbors

Idea:

- "Do as your neighbors do!"
- Classify a new data-point according to a *majority vote* of your k nearest neighbors

How do you measure "near"?

x discrete (e.g., strings): Hamming distance

$d(x_1,x_2)$ = # features on which x_1 and x_2 differ

x continuous (e.g., images): Euclidean distance

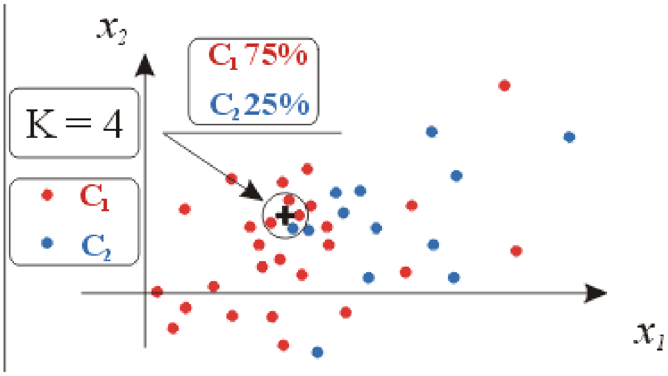
$d(x_1,x_2) = || x_1 - x_2 ||$ = square root of sum of squared differences between corresponding elements of data vectors

22

Example

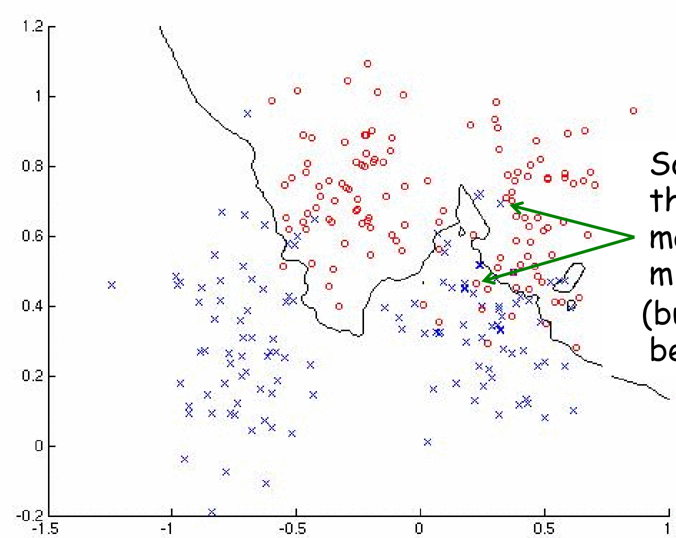
Input Data: 2-D points (x_1,x_2)

Two classes: C_1 and C_2. New Data Point +



K = 4: Look at 4 nearest neighbors.
3 are in C_1, so classify + as C_1

K-NN produces a Nonlinear Decision Boundary



Some points near the boundary may be misclassified (but perhaps okay because of noise)

Next Time

Regression (Learning functions with continuous outputs)

- Linear Regression
- Neural Networks

To Do:

- Project 4
- Read Chapter 18