ENGN8530:

Computer Vision and Image Understanding: Theories and Research

Topic 4

Bayesian analysis and classifiers

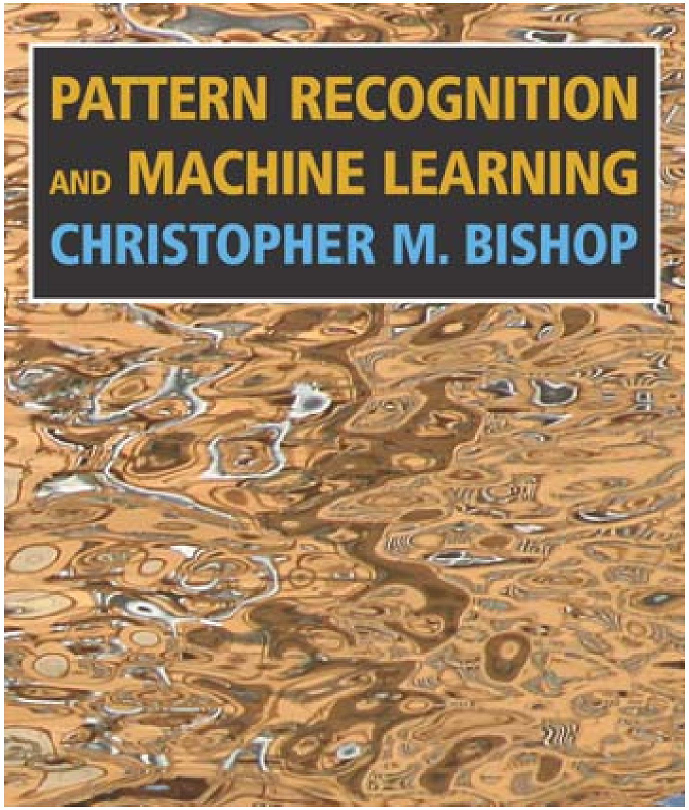
ViSTA NICTA and RSISE ANU

Textbooks

- Pattern Recognition (3rd edition), S. Theodoridis, K. Koutroumbas (2006), Elsevier, ISBN 0-12-369531-7.
- Information Theory, Inference, and Learning Algorithms, D. MacKay (2003). PDF version available online.
- Pattern classification (2nd edition), R. O. Duda, P. E. Hart, D. G. Stork (2001), Wiley, New York, ISBN 0-471-05669-3.
  covers classification but also contains much more material

One more book for background reading …

- **Pattern Recognition and Machine Learning** Christopher Bishop, Springer, 2006.
  - Excellent on classification and regression
  - Advanced topics



Statistical Learning for Prediction

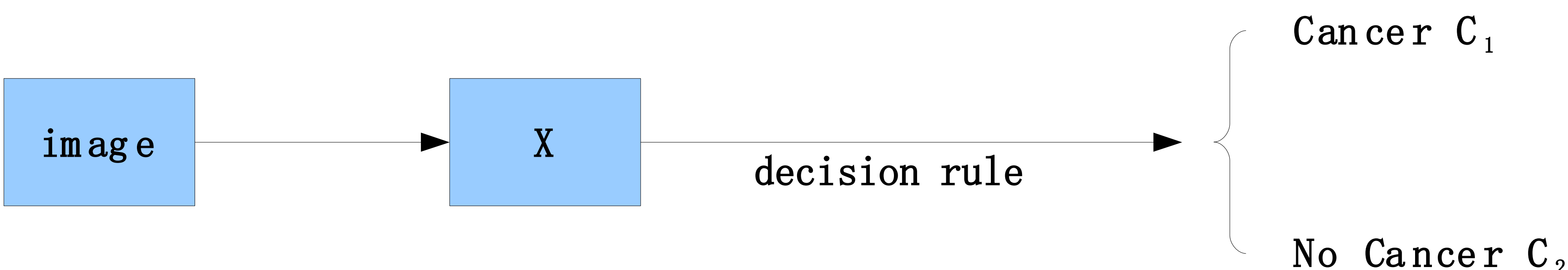
- The usual setting is that provided with a training set $(X_I , y_I), \ldots,$ (X_N , y_N) we want to guess a mapping f so that on a new sample (X, y) not seen during training, we have y = $f(X)$.
- This mapping is usually picked in a set of mappings defined a priori using a training algorithm.

Example of a Task (Face Classification)

For instance *x* can be a small grayscale image and *y* a boolean value standing for the presence of a centered face.

Bayesian Decision Theory

- Suppose we want to make measurements on a medical image and classify it as showing evidence of cancer or not

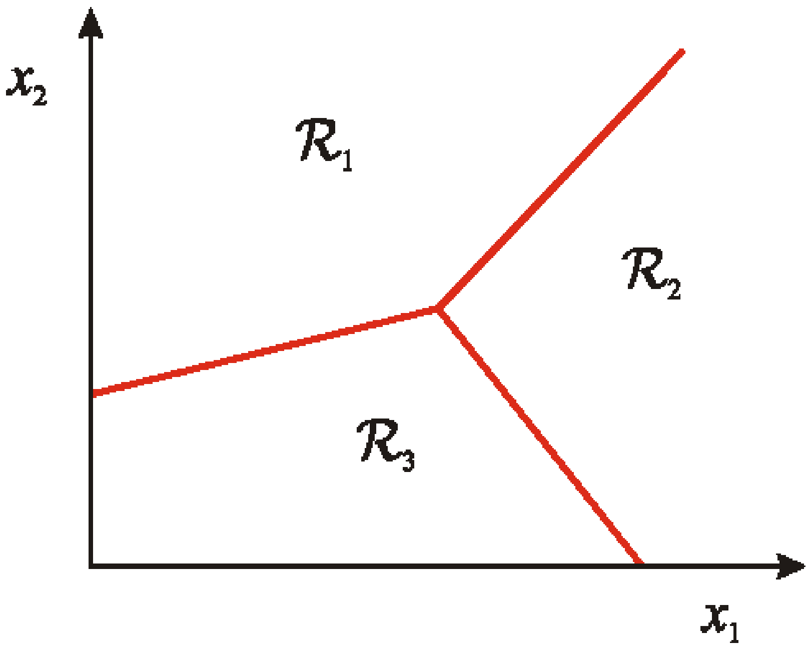


$$p(\mathbf{x}, C_i) = p(\mathbf{x}|C_i)p(C_i)$$

How do we make the best decision?

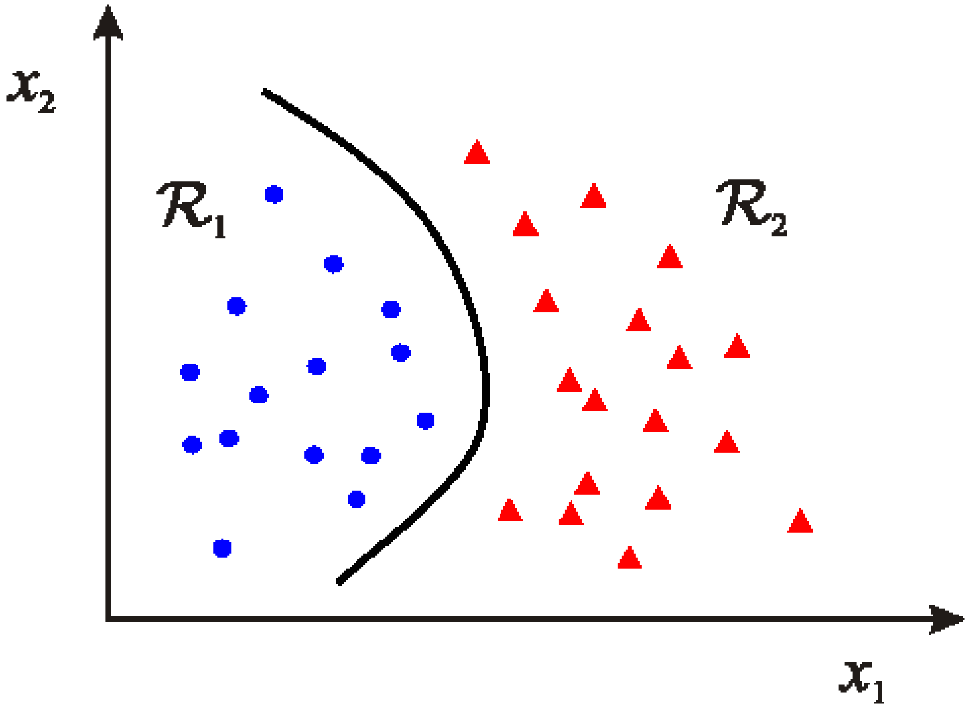
Classification

- Assign input vector X to one of two or more classes C_k
- Any decision rule divides input space into decision regions separated by decision boundaries.



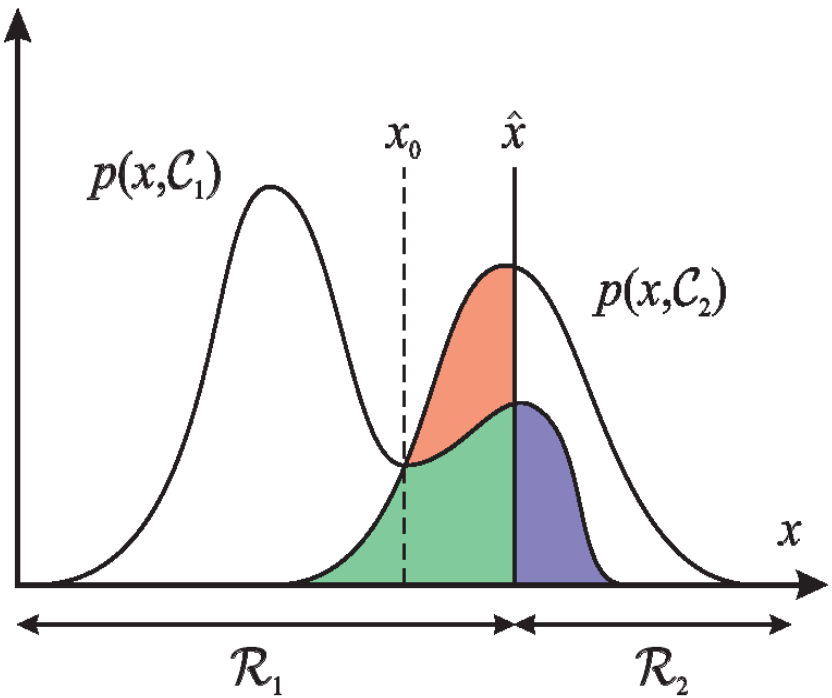
Classification

- Example: Two class decision depending on a 2D vector measurement.
- Also we would like a confidence measure (how sure are we that the input belongs to the chosen category?



Decision Boundary for Average Error

- Consider a two class decision depending on a scalar variable X



$$p(\text{error}) = \int_{-\infty}^{+\infty} p(\text{error}, x)\, dx$$

$$= \int_{\mathcal{R}_1} p(x, C_2)\, dx + \int_{\mathcal{R}_2} p(x, C_1)\, dx$$

minimise number of misclassifications if the decision boundary is at X_0

Bayes Decision Rule

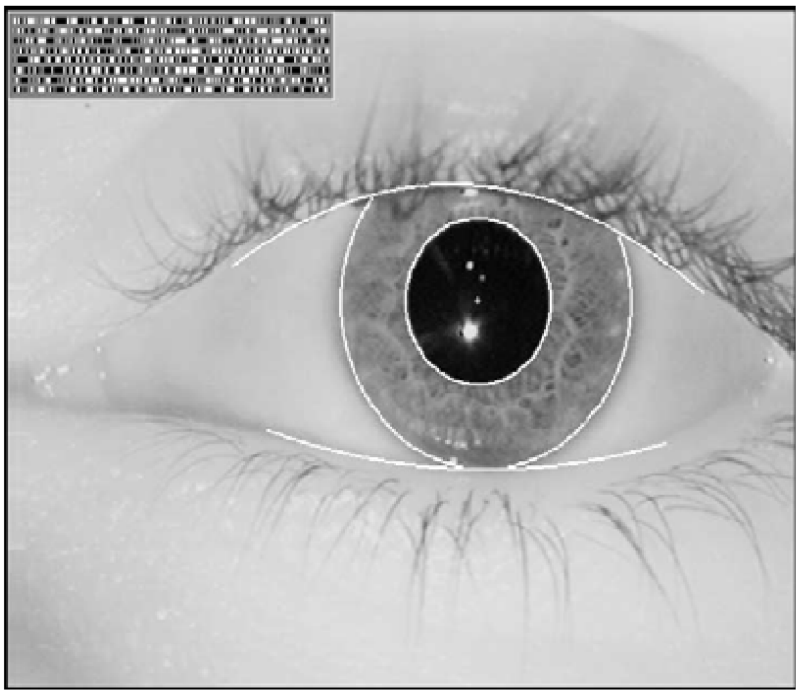
- Assign X to the class C_i for which $p(X, C_i)$ is largest. This is equivalent to
- Assign X to the class C_i for which $p(C_i \mid X)$ is largest because we have

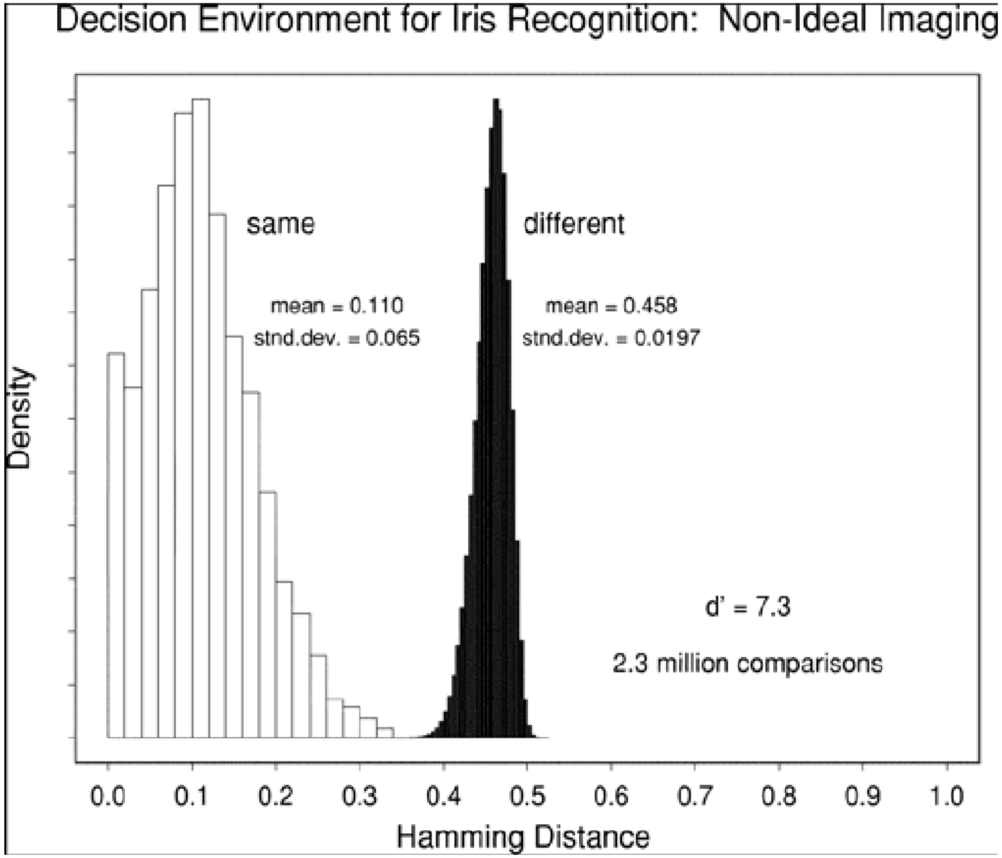
$$p(x, C_i) = p(C_i|x)\,p(x)$$

  - So usually we are interested in the **posterior** $p(C_i \mid X)$ for classification. If no prior information, then the posterior is same as the **likelihood** $p(X \mid C_i)$

Classification

- Example: Iris recognition





How Iris Recognition Works, J. Daugman (2004) IEEE TCSVT.

Discriminant v.s. Generative approaches

Discriminant

+ don't have to learn parameters which aren't used (e.g. covariance)
+ easy to learn
- no confidence measure
- have to retrain if dimension of feature vectors changed

Generative

+ have confidence measure
+ can use 'reject option'
+ easy to add independent measurements

$$
\begin{aligned}
p(\mathcal{C}_k|\mathbf{x}_\mathsf{A}, \mathbf{x}_\mathsf{B}) &\propto p(\mathbf{x}_\mathsf{A}, \mathbf{x}_\mathsf{B}|\mathcal{C}_k)p(\mathcal{C}_k) \\
&\propto p(\mathbf{x}_\mathsf{A}|\mathcal{C}_k)p(\mathbf{x}_\mathsf{B}|\mathcal{C}_k)p(\mathcal{C}_k) \\
&\propto \frac{p(\mathcal{C}_k|\mathbf{x}_\mathsf{A})p(\mathcal{C}_k|\mathbf{x}_\mathsf{B})}{p(\mathcal{C}_k)}
\end{aligned}
$$

- expensive to train (because many parameters)

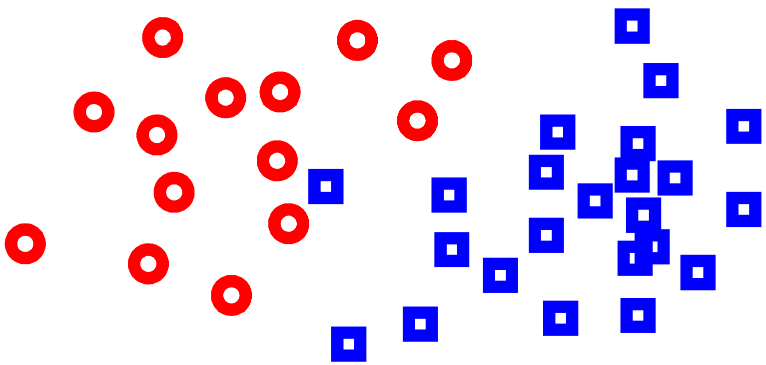
Discriminative Methods

- There are numerous techniques which bypass the modeling of the population of interest:
  - Multi-layer perceptrons (neural network)
  - K-nearest-neighbours
  - Decision trees
  - Boosting (AdaBoost, LPBoost ...)
  - Support vector machine
    - They directly approximate a mapping from the signal space into the space of the predicted values

K-nearest-neighbour

- The nearest neighbour classifier predicts that the class of an test data *X* belongs to the class of the closest training example.
- Very good performance with a good metric, but expensive prediction.
- It must store all training data.
- Curse of dimensionality: required amount of training data increases exponentially with dimension

Nearest Neighbour Rule

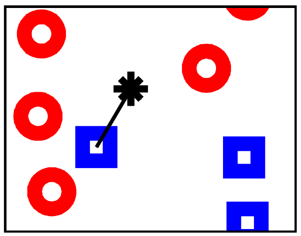


Non-parametric pattern classification

Consider a two class problem where each sample consists of two measurements (X,y).

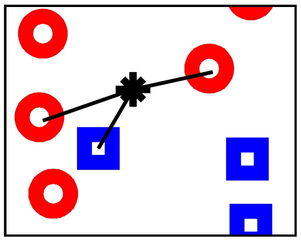
For a given query point q, assign the class of the nearest neighbour.

k = 1

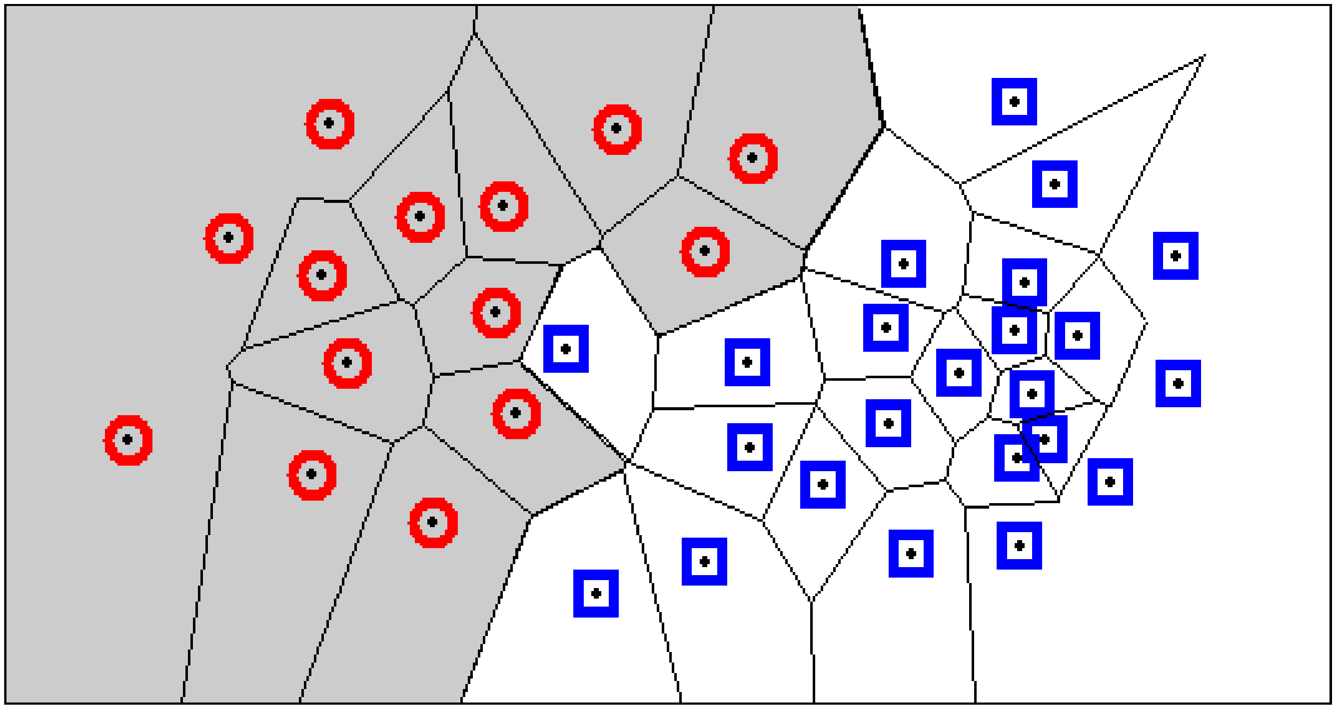


Compute the k nearest neighbours and assign the class by majority vote.

k = 3



Decision Regions



Each cell contains one sample, and every location within the cell is closer to that sample than to any other sample.

A Voronoi diagram divides the space into such cells.

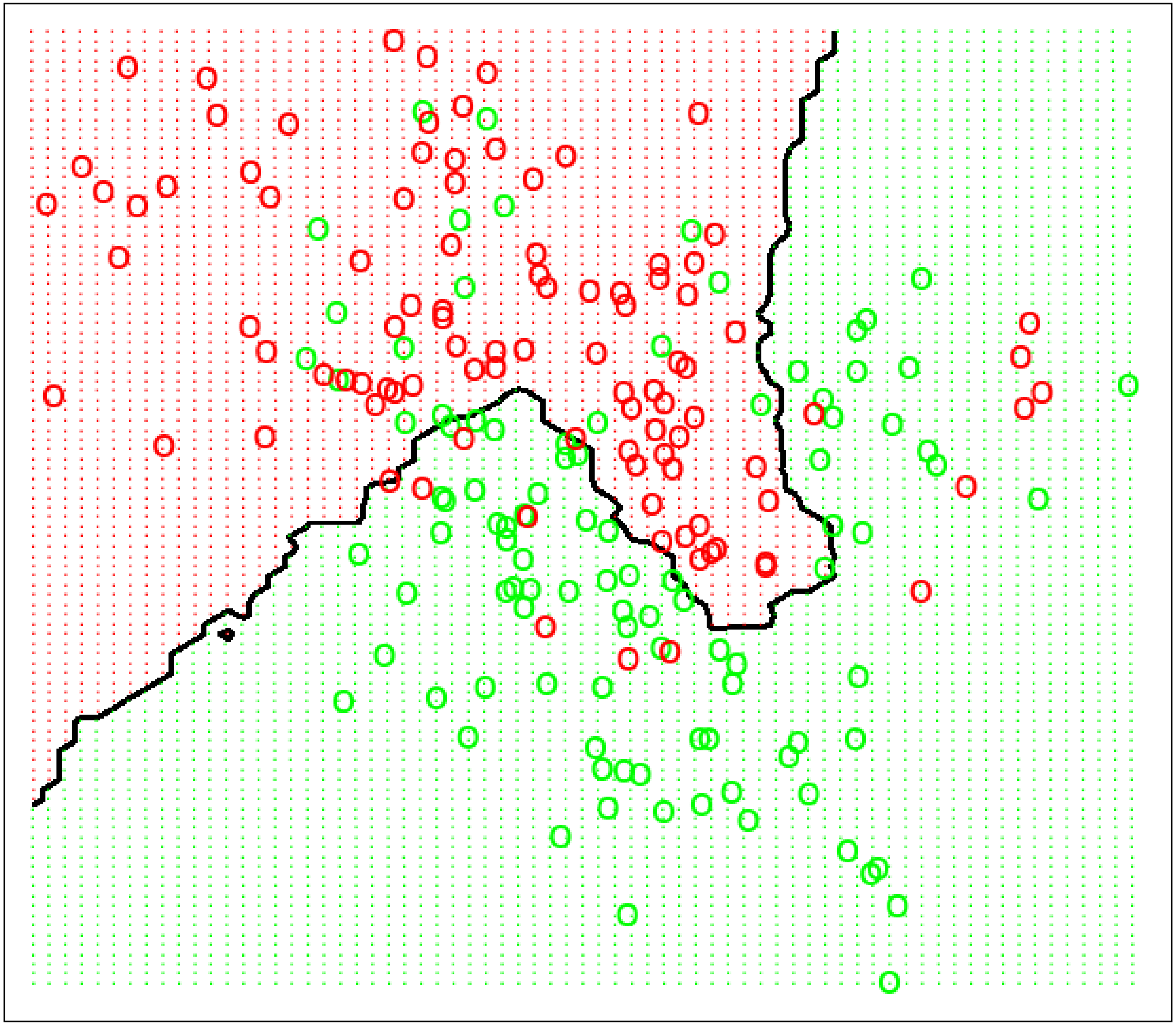
Every query point will be assigned the classification of the sample within that cell. The *decision boundary* separates the class regions based on the 1-NN decision rule.

Knowledge of this boundary is sufficient to classify new points.

The boundary itself is rarely computed; many algorithms seek to retain only those points necessary to generate an identical boundary.

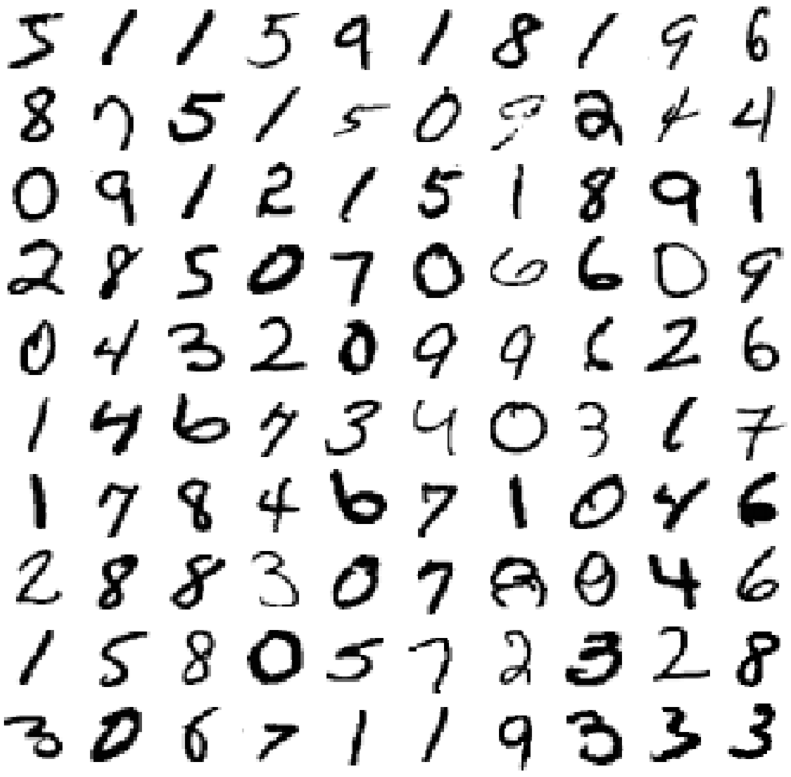
One more example

15-Nearest Neighbor Classifier



K-nearest-neighbour: Example

- The MNIST database is a free database of tens of thousands of handwritten digits.



K-nearest-neighbour: Example

- NN Classification results on MNIST

Method	Error Rate (%) LeCun *et al*	Error Rate (%) Claus	Notes
1-nearest neighbour		3.09	
3-nearest neighbour	5.0	2.83	1
1-nearest neighbour, de-skewed		2.04	2
3-nearest neighbour, de-skewed	2.4	1.92	1,2
2-layer neural network, 300 hidden units	4.7		
LeNet-4	1.1		
LeNet-5	0.95		

K-nearest-neighbour

- What distance measure to use?
  - Often Euclidean distance is used.
  - Locally adaptive metrics.
  - More complicated with non-numeric data, or when different dimensions have different scales.
- Choice of *k*?
  - Cross-validation.
  - 1-NN often performs well in practice.
  - k-NN needed for overlapping classes.
  - Re-label all data according to k-NN, then classify with 1-NN.

Neighbourhood Component Analysis

- What distance measure to use?

Neighbourhood components analysis, **Jacob Goldberger, Sam Roweis, Geoff Hinton and Ruslan Salakhutdinov. NIPS 2004**

Neighbourhood Component Analysis

The **expected** leave-one-out classification performance is:

$$\begin{aligned}
\phi &= \frac{1}{N}\sum_i p_i^+ \\
&= \frac{1}{N}\sum_i \sum_{j\in C_i} p_{ij} \\
&= \frac{1}{N}\sum_i \sum_{j\in C_i} \frac{e^{-d_{ij}}}{\sum_{k\neq i} e^{-d_{ik}}}
\end{aligned}$$

This is the objective function we will try to maximize during learning. It is **much smoother with respect to the distances** $\{d_{ij}\}$ than the actual leave one out classification error.

Neighbourhood Component Analysis

$$
\begin{aligned}
d_{ij} &= (x_i - x_j)^\top \mathbf{A}^\top \mathbf{A}(x_i - x_j) \\
&= (\mathbf{A}x_i - \mathbf{A}x_j)^\top (\mathbf{A}x_i - \mathbf{A}x_j) \\
&= (y_i - y_j)^\top (y_i - y_j)
\end{aligned}
$$

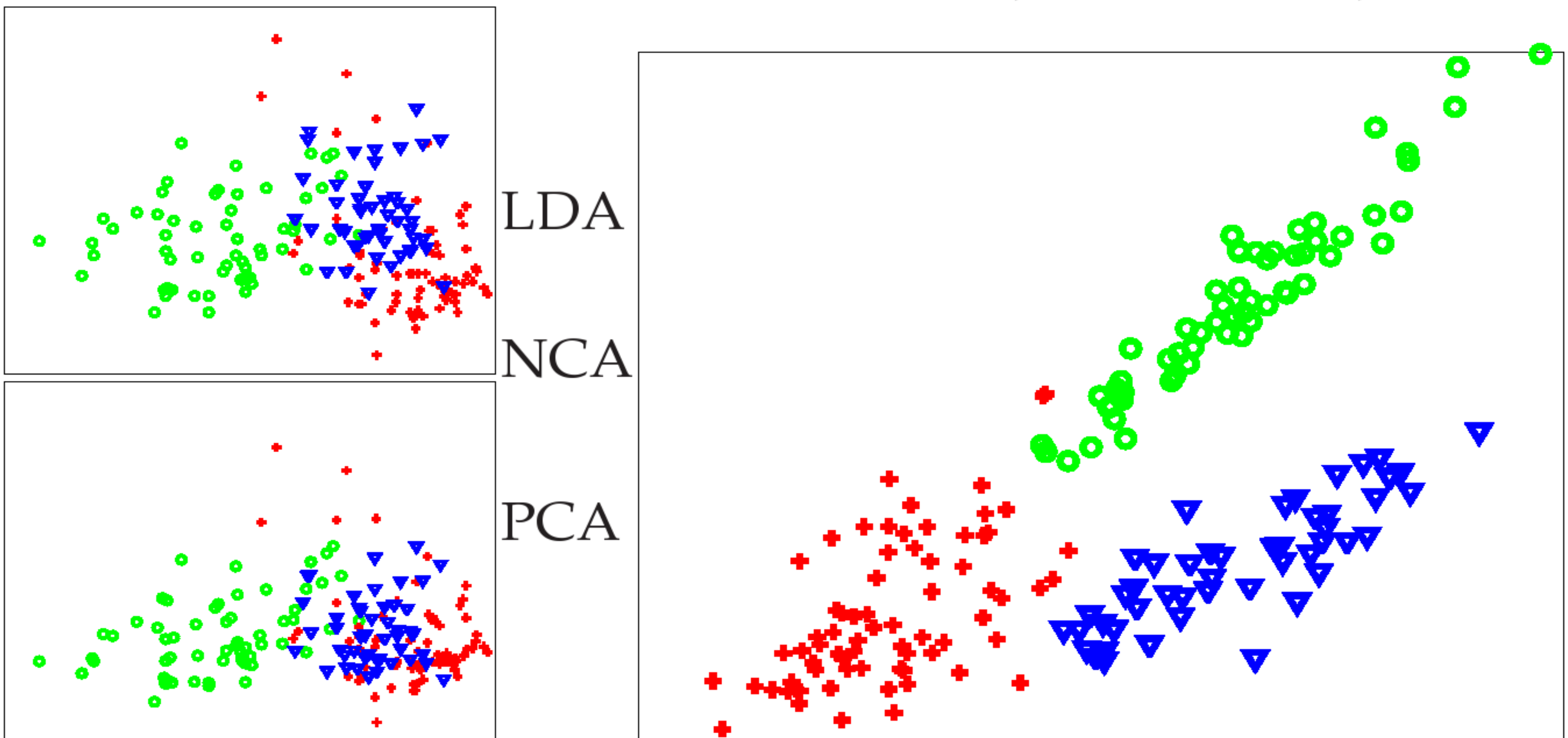
In other words, this is exactly equivalent to applying **a simple (spherical) Euclidean metric** to the points $\{y_i = \mathbf{A}x_i\}$.

Neighbourhood Component Analysis

UCI "Wine", N=178, D=13, 3 classes. Half train, half test.

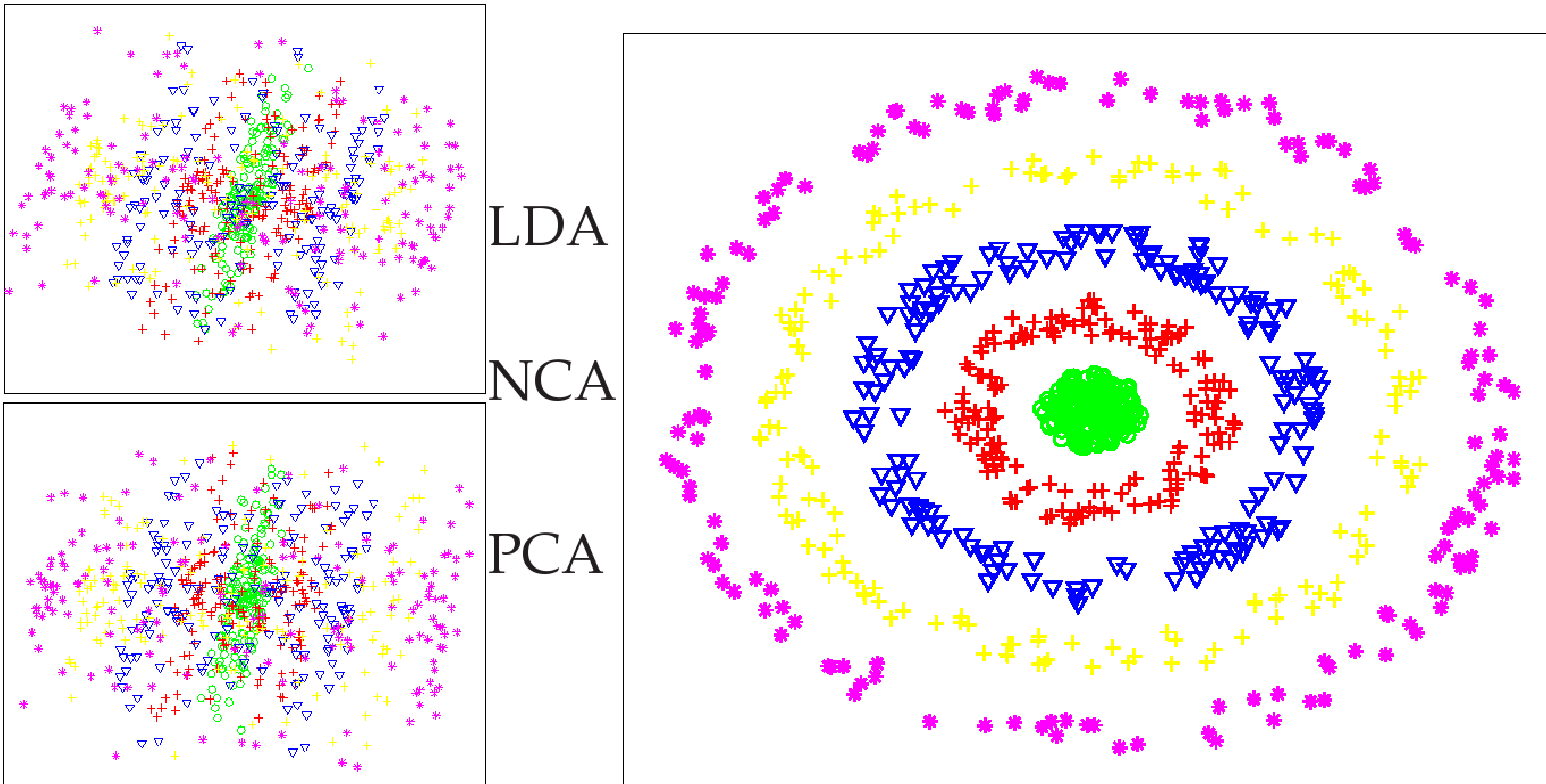
Test errors using d=D=13, and K chosen by LOO:

Euclidean=30%;Whiten=25%;NCA=7%



Test errors using KNN in 2D: LDA=28%; PCA=31%; NCA=5%

Neighbourhood Component Analysis

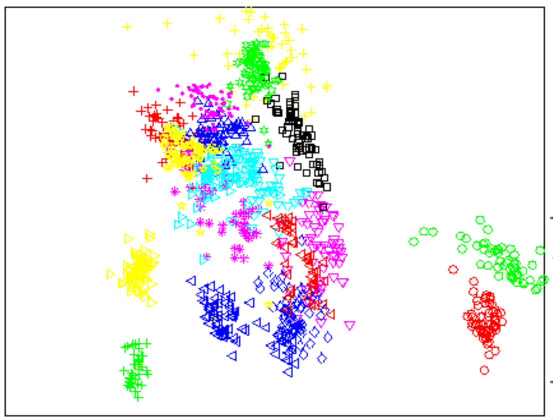


Neighbourhood Component Analysis

Grayscale images of faces taken from frames of a 20x28 video. (18 people as class labels, D=560, N=100 for training, 900 test).

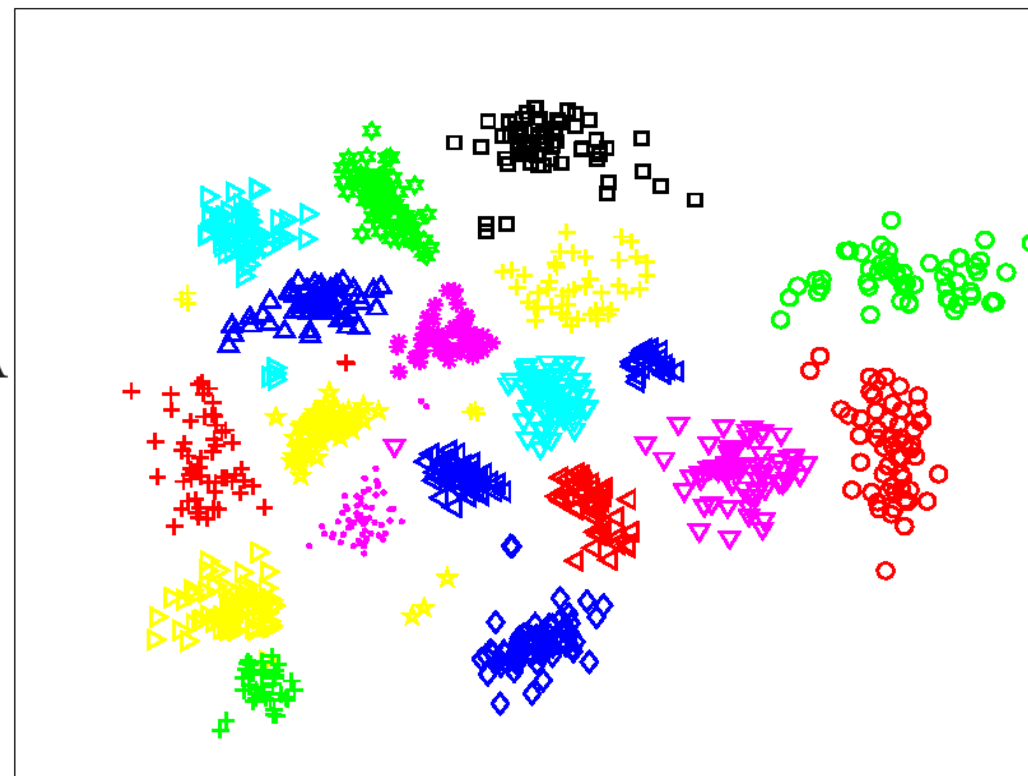
Test errors using d=D=560, and K chosen by LOO:

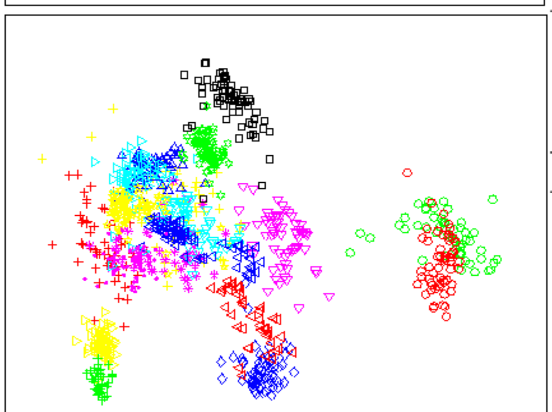
Euclidean=18%;Whiten=22%;NCA=4%



LDA

NCA





PCA

Test errors using KNN in 2D: LDA=25%; PCA=37%; NCA=5%

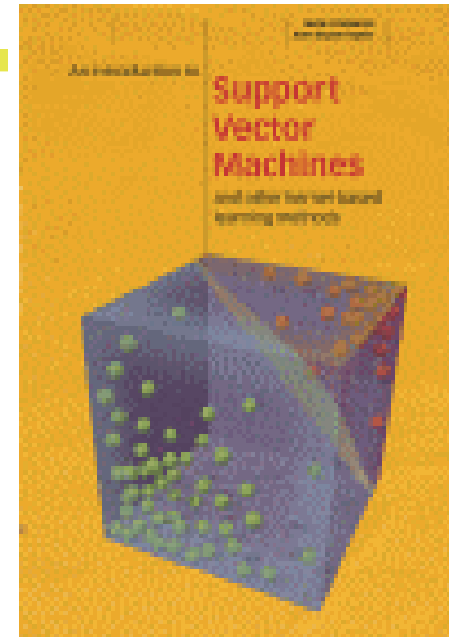
Support Vector Machine

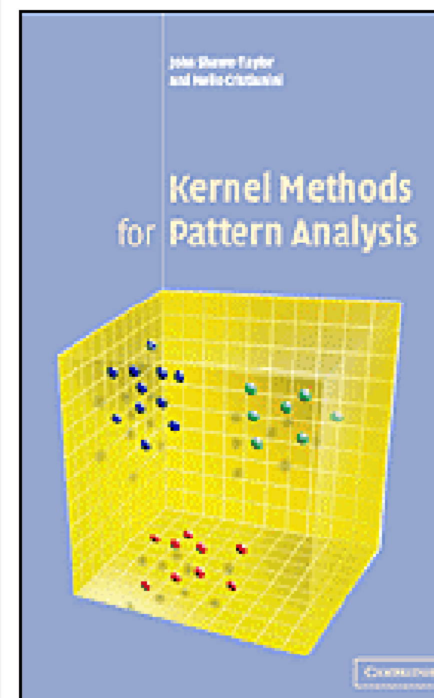
- SVM is related to statistical learning theory [3]
- SVM was first introduced in 1992 [1]
- SVM becomes popular because of its success in handwritten digit recognition
  - 1.1% test error rate for SVM. This is the same as the error rates of a carefully constructed neural network, LeNet 4.
    - See Section 5.11 in [2] or the discussion in [3] for details
- SVM is now regarded as an important example of *kernel methods*, one of the key area in machine learning
  - Note: the meaning of kernel is different from the kernel function for Parzen windows

[1] B.E. Boser *et al*. A Training Algorithm for Optimal Margin Classifiers. Annual Workshop on Computational Learning Theory 5 144-152, Pittsburgh, 1992.
[2] L. Bottou *et al*. Comparison of classifier methods: a case study in handwritten digit recognition. 12th IAPR Int. Conf. Pattern Recognition, vol. 2, pp. 77-82.
[3] V. Vapnik. The Nature of Statistical Learning Theory. 2nd edition, Springer, 1999.

Support Vector Machine

- Books
  - **Kernel Methods for Pattern Analysis**, John Shawe-Taylor & Nello Cristianini, Cambridge University Press, 2004.
  - **Support Vector Machines and other kernel◎based learning methods**, John Shawe-Taylor & Nello Cristianini, Cambridge University Press, 2000.
- Website: http://www.kernel-machines.org/

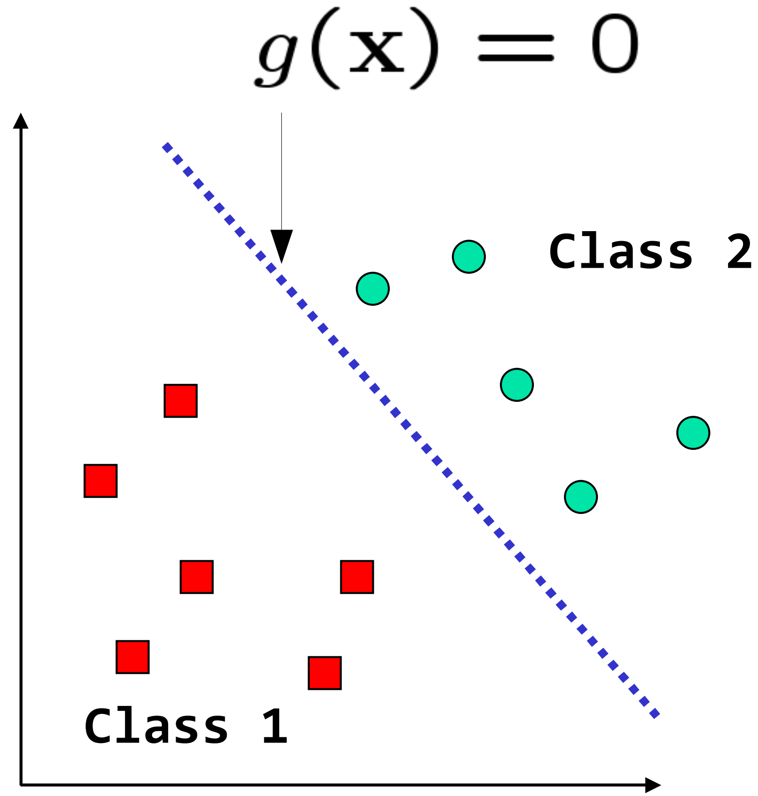




Linear Classifiers

- A linear discriminant has the form

$$g(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + w_0$$



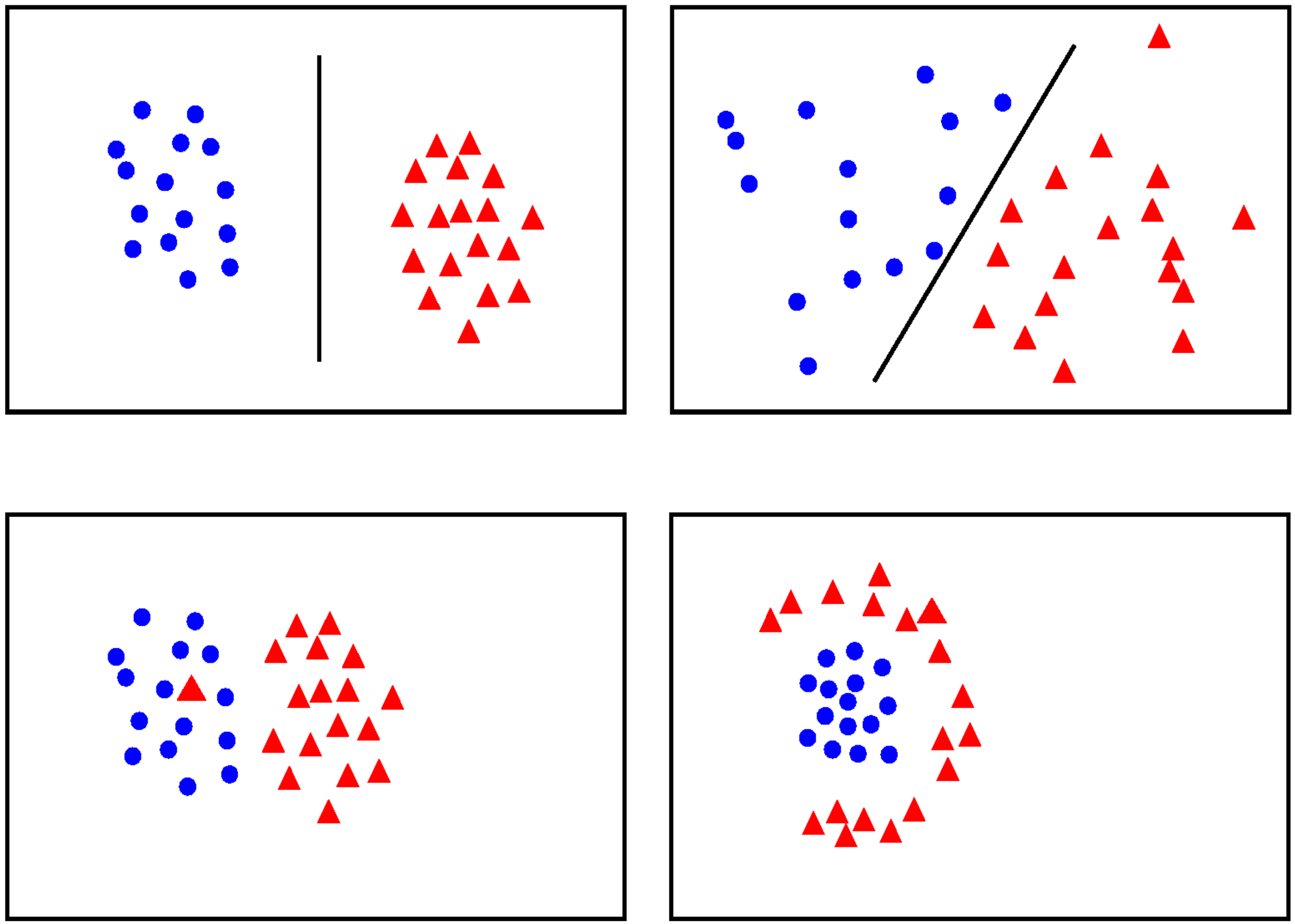
$\mathbf{w}$ is the normal to the plane, and w_0 the bias

$\mathbf{w}$ is known as the weight vector

Linear Separability

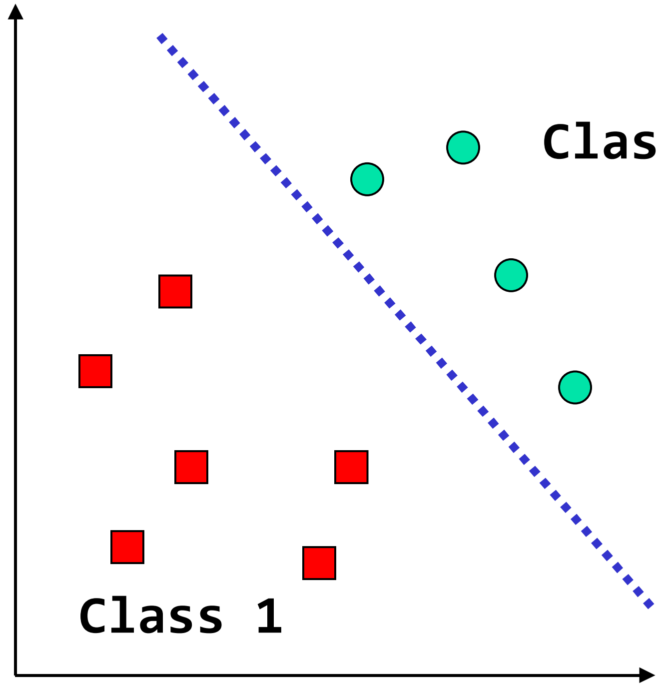
linearly separable

not linearly separable



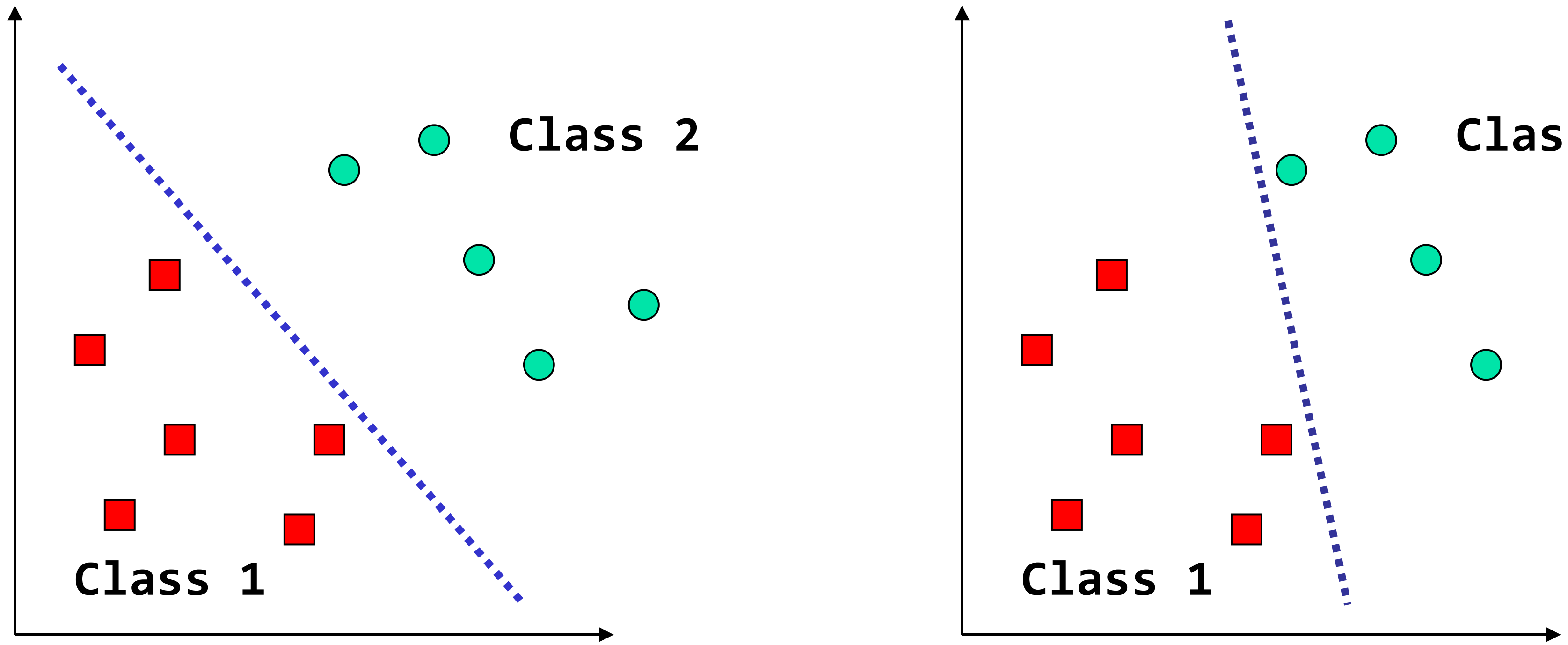
Support Vector Machine

- Consider a two-class, linearly separable classification problem
- Many decision boundaries!
  - The Perceptron algorithm can be used to find such a boundary
  - Different algorithms have been proposed



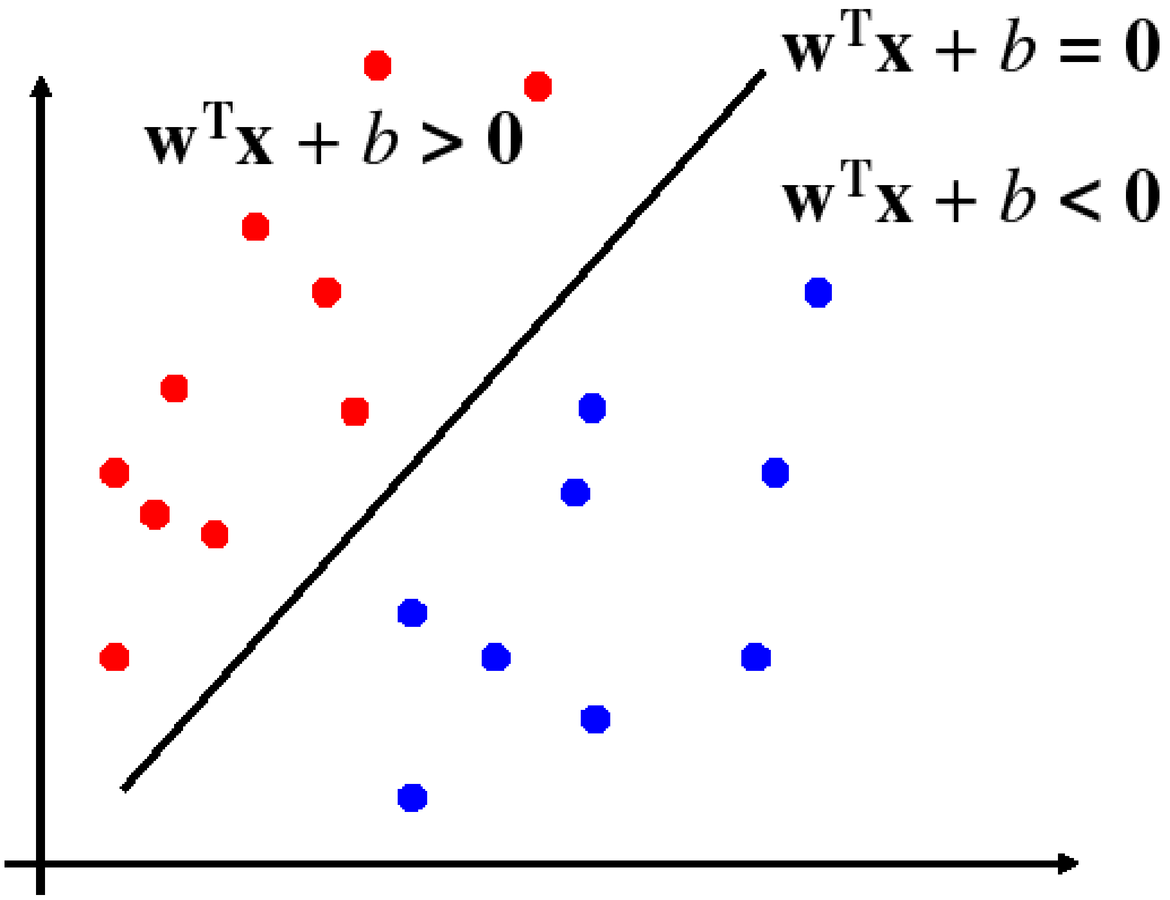
Are all decision boundaries are equally good for classification?

Decision Boundaries



Linear Classifiers Revisited

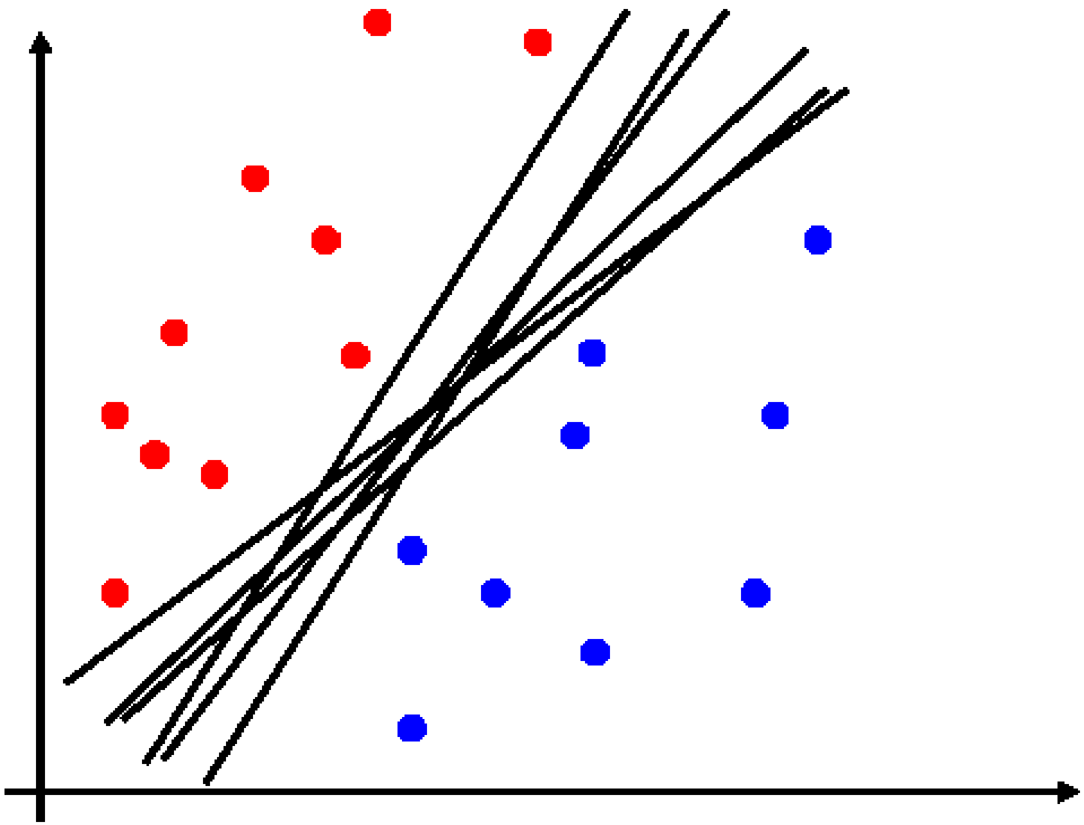
- Binary classification can be viewed as the task of separating classes in feature space:



$$f(\mathbf{x}) = \mathrm{sign}(\mathbf{w}^{\mathrm{T}}\mathbf{x} + b)$$

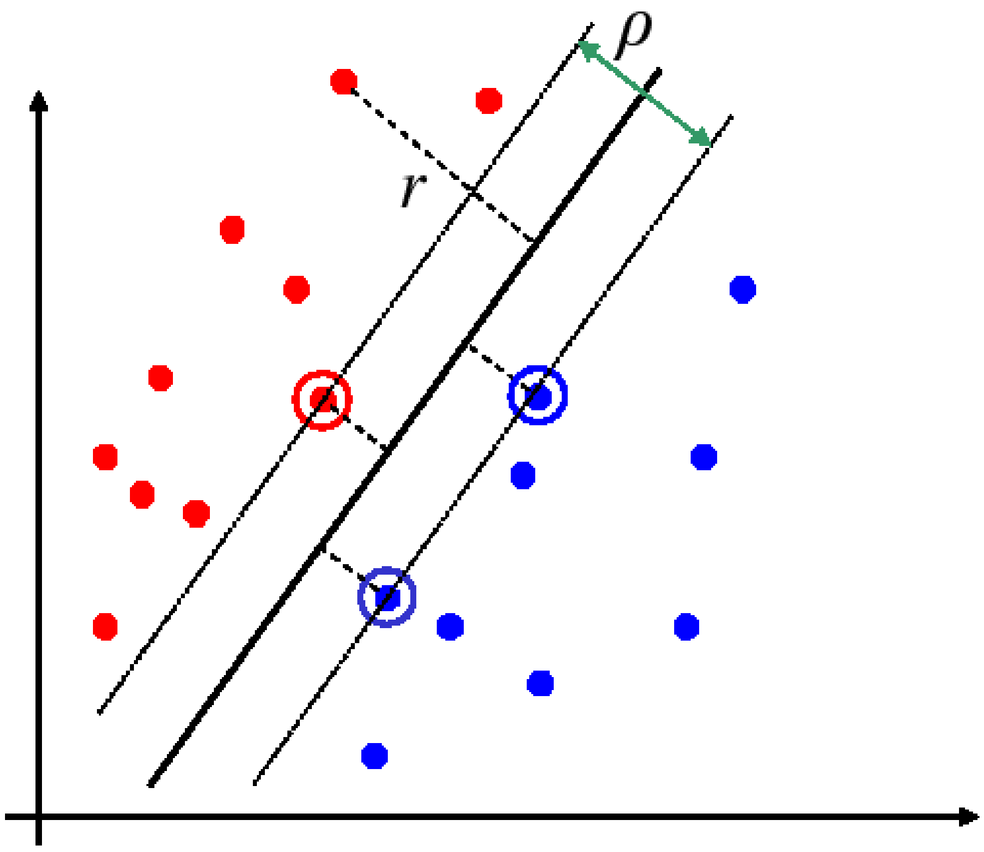
Linear Classifiers

Which of the linear separators is optimal?



Classification Margin

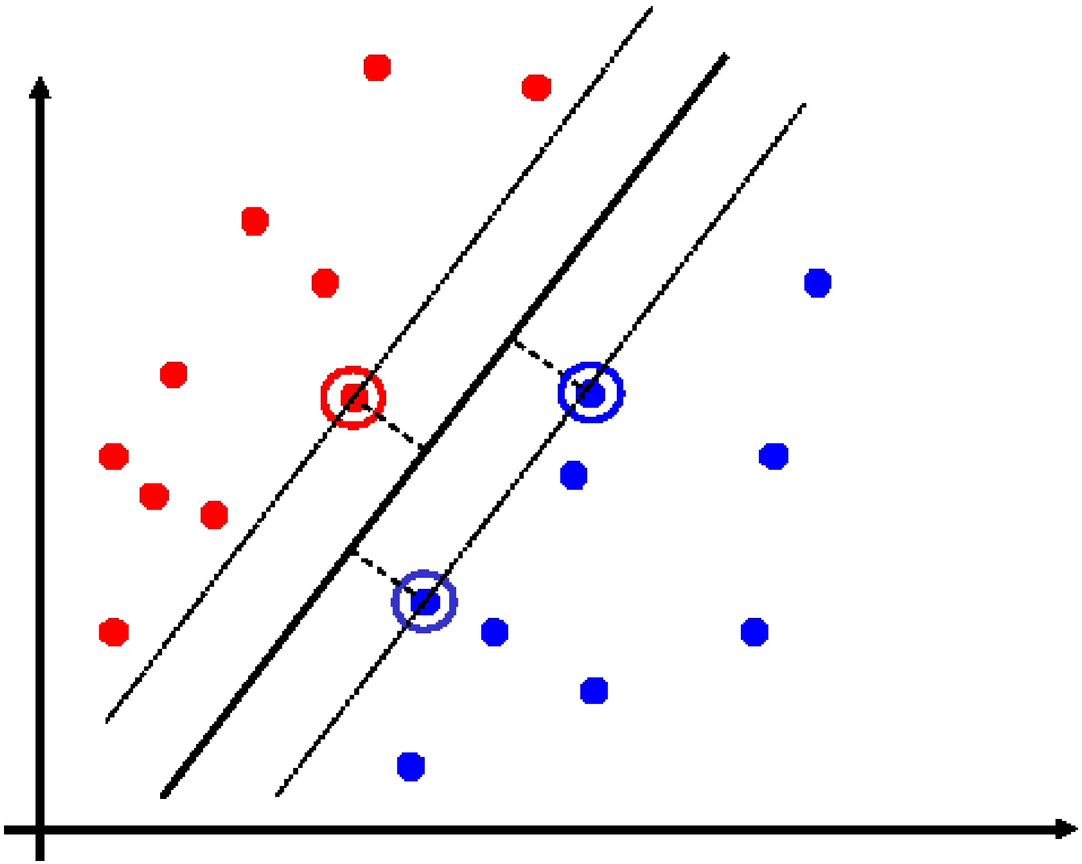
- Distance from example $\mathbf{x}_i$ to the separator is $r = \dfrac{\mathbf{w}^T \mathbf{x}_i + b}{\|\mathbf{w}\|}$
- Examples closest to the hyperplane are ***support vectors***.
- ***Margin*** ρ of the separator is the distance between support vectors.



Max Margin Classification

Maximizing the margin is good according to intuition and PAC theory.

Implies that only support vectors matter; other training examples are ignorable.



Linear SVM

Let training set $\{(\mathbf{x}_i, y_i)\}_{i=1..n}$, $\mathbf{x}_i \in \mathbf{R}^d$, $y_i \in \{-1, 1\}$ be separated by a hyperplane with margin ρ. Then for each training example $(\mathbf{x}_i, y_i)$:

$$\begin{array}{ll} \mathbf{w}^{\mathrm{T}}\mathbf{x}_i + b \leq -\rho/2 & \text{if } y_i = -1 \\ \mathbf{w}^{\mathrm{T}}\mathbf{x}_i + b \geq \rho/2 & \text{if } y_i = 1 \end{array} \quad \Leftrightarrow \quad y_i(\mathbf{w}^{\mathrm{T}}\mathbf{x}_i + b) \geq \rho/2$$

For every support vector $\mathbf{x}_s$ the above inequality is an equality. After rescaling $\mathbf{w}$ and b by $\rho/2$ in the equality, we obtain that distance between each $\mathbf{x}_s$ and the hyperplane is $r = \frac{y_s(\mathbf{w}^T\mathbf{x}_s + b)}{\|\mathbf{w}\|} = \frac{1}{\|\mathbf{w}\|}$

Then the margin can be expressed through (rescaled) $\mathbf{w}$ and b as:

$$\rho = 2r = \frac{2}{\|\mathbf{w}\|}$$

Linear SVM

- Then we can formulate the *quadratic optimization problem:*

Find $\mathbf{w}$ and b such that

$$\rho = \frac{2}{\|\mathbf{w}\|} \text{ is maximized}$$

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

Which can be reformulated as:

Find $\mathbf{w}$ and b such that

$$\boldsymbol{\Phi}(\mathbf{w}) = ||\mathbf{w}||^2 = \mathbf{w}^T\mathbf{w} \text{ is minimized}$$

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

Linear SVM

Find $\mathbf{w}$ and b such that
$\mathbf{\Phi}(\mathbf{w}) = \mathbf{w}^T\mathbf{w}$ is minimized
and for all $(\mathbf{x}_i, y_i)$, $i=1..n$: $y_i (\mathbf{w}^T\mathbf{x}_i + b) \geq 1$

Need to optimize a *quadratic* function subject to *linear* constraints.

Quadratic optimization problems are a well-known class of mathematical programming problems for which several (non-trivial) algorithms exist.

The solution involves constructing a *dual problem* where a *Lagrange multiplier* α_i is associated with every inequality constraint in the primal (original) 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

Linear SVM Solution

Given a solution $\alpha_1 \ldots \alpha_n$ to the dual problem, solution to the primal is:

$$\boxed{\mathbf{w} = \sum \alpha_i y_i \mathbf{x}_i \qquad b = y_k - \sum \alpha_i y_i \mathbf{x}_i^{\mathrm{T}} \mathbf{x}_k \quad \text{for any } \alpha_k > 0}$$

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

Then the classifying function is (note that we don't need **w** explicitly):

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

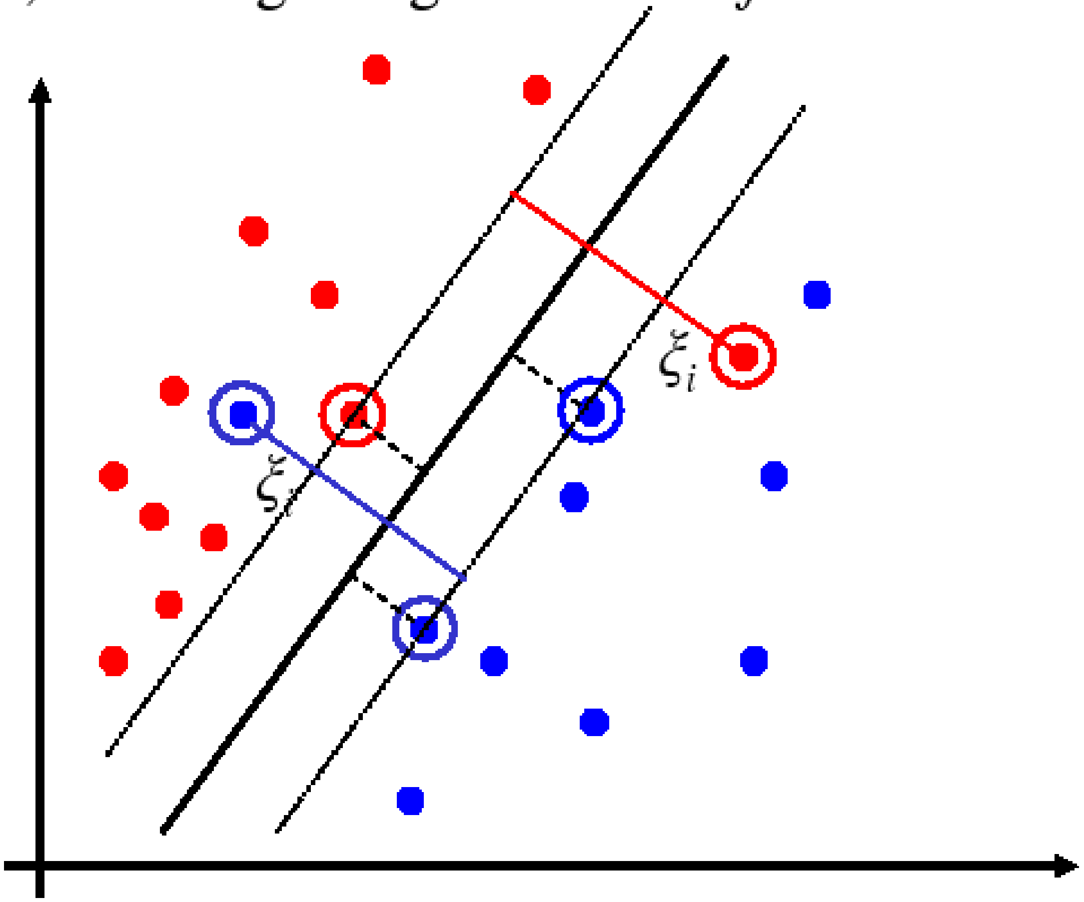
Notice that it relies on an *inner product* between the test point **x** and the support vectors $\mathbf{x}_i$ – we will return to this later.

Also keep in mind that solving the optimization problem involved computing the inner products $\mathbf{x}_i^{\mathrm{T}} \mathbf{x}_j$ between all training points.

Soft Margin SVM

What if the training set is not linearly separable?

Slack variables ξ_i can be added to allow misclassification of difficult or noisy examples, resulting margin called *soft*.



Soft Margin SVM

The old formulation:

Find $\mathbf{w}$ and b such that
$\mathbf{\Phi}(\mathbf{w}) = \mathbf{w}^{\mathrm{T}}\mathbf{w}$ is minimized
and for all $(\mathbf{x}_i, y_i)$, $i=1..n$: $y_i(\mathbf{w}^{\mathbf{T}}\mathbf{x}_i + b) \geq 1$

Modified formulation incorporates slack variables:

Find $\mathbf{w}$ and b such that
$\mathbf{\Phi}(\mathbf{w}) = \mathbf{w}^{\mathrm{T}}\mathbf{w} + C\sum\xi_i$ is minimized
and for all $(\mathbf{x}_i, y_i)$, $i=1..n$: $y_i(\mathbf{w}^{\mathbf{T}}\mathbf{x}_i + b) \geq 1 - \xi_{i,}$, $\xi_i \geq 0$

Parameter C can be viewed as a way to control overfitting: it "trades off" the relative importance of maximizing the margin and fitting the training data.

Soft Margin SVM

- Dual problem is identical to separable case (would *not* be identical if the 2-norm penalty for slack variables $C\Sigma\xi_i^2$ was used in primal objective, we would need additional Lagrange multipliers for slack variables):

Find $\alpha_1 \dots \alpha_N$ 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) $0 \leq \alpha_i \leq C$ for all α_i

- Again, $\mathbf{x}_i$ with non-zero α_i will be support vectors.
- Solution to the dual problem is:

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

$b = y_k(1 - \xi_k) - \Sigma\alpha_i y_i \mathbf{x}_i^T \mathbf{x}_k$ for any k s.t. $\alpha_k > 0$

Again, we don't need to compute $\mathbf{w}$ explicitly for classification:

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

Why Max Margin?

Vapnik has proved the following:

The class of optimal linear separators has VC dimension h bounded from above as

$$h \leq \min\left\{\left\lceil \frac{D^2}{\rho^2} \right\rceil, m_0\right\} + 1$$

where ρ is the margin, D is the diameter of the smallest sphere that can enclose all of the training examples, and m_0 is the dimensionality.

Intuitively, this implies that regardless of dimensionality m_0 we can minimize the VC dimension by maximizing the margin ρ.

Thus, complexity of the classifier is kept small regardless of dimensionality.

Linear SVM: 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.

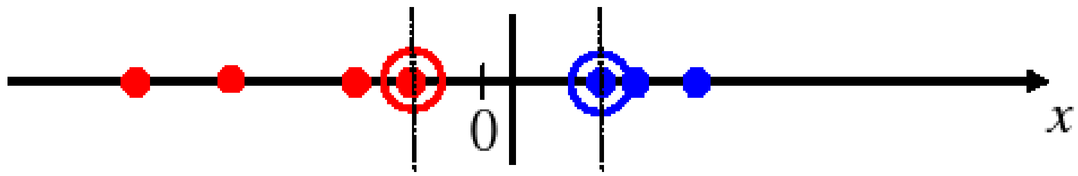
Both in the dual formulation of the problem and in the solution training points appear only inside inner products:

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

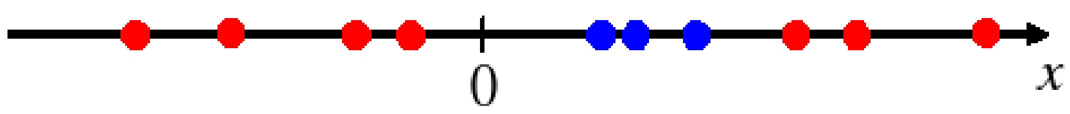
$$f(\mathbf{x}) = \Sigma\alpha_i y_i \boxed{\mathbf{x}_i^T\mathbf{x}} + b$$

Non-linear (Kernel) SVM

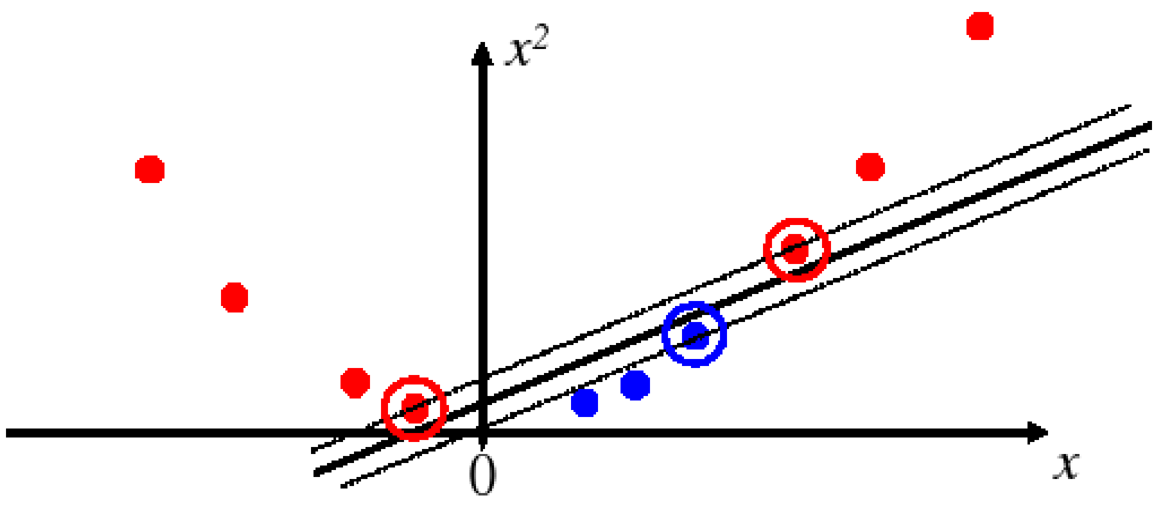
Datasets that are linearly separable with some noise work out great:



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

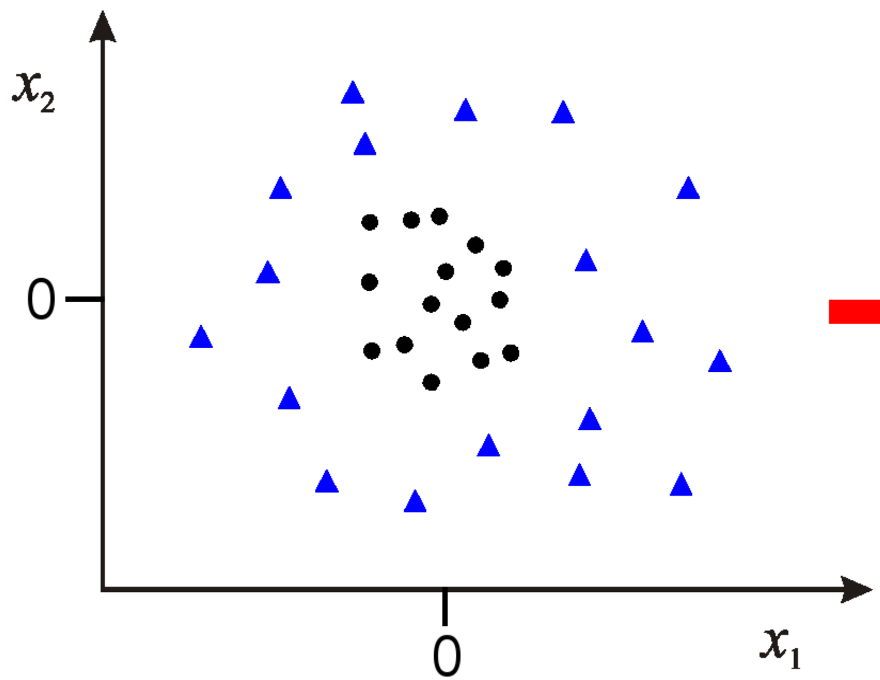


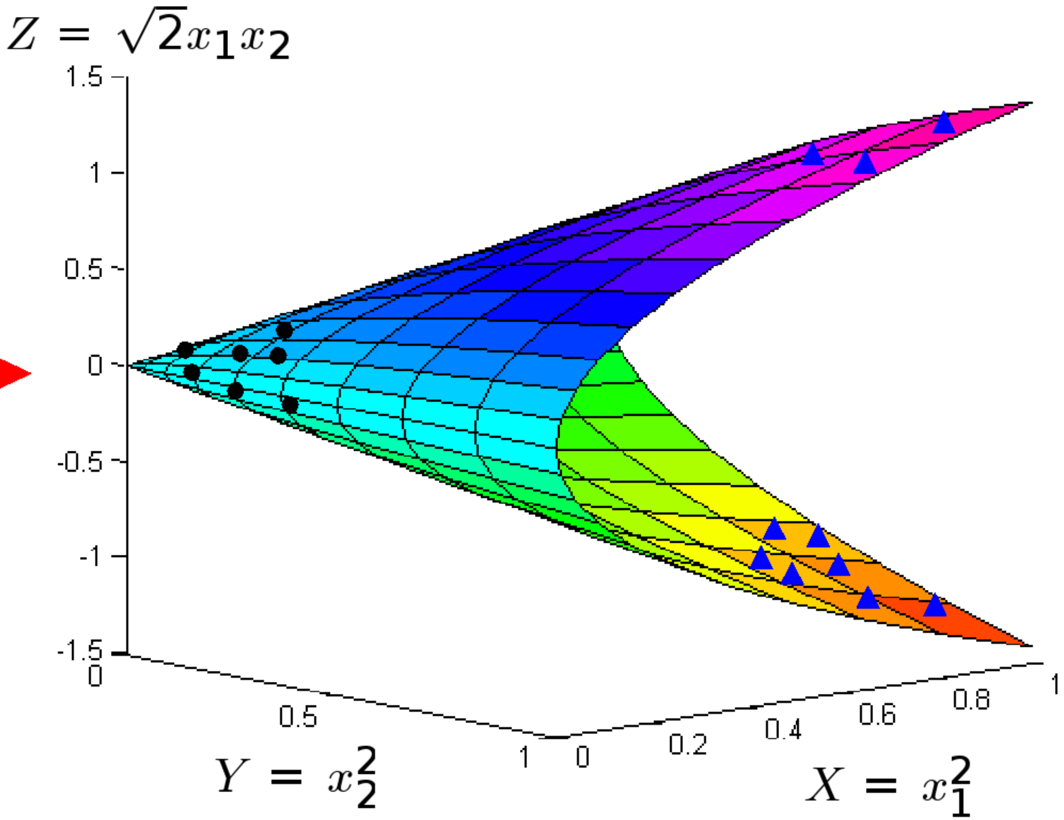
How about… mapping data to a higher-dimensional space:



Non-linear (Kernel) SVM

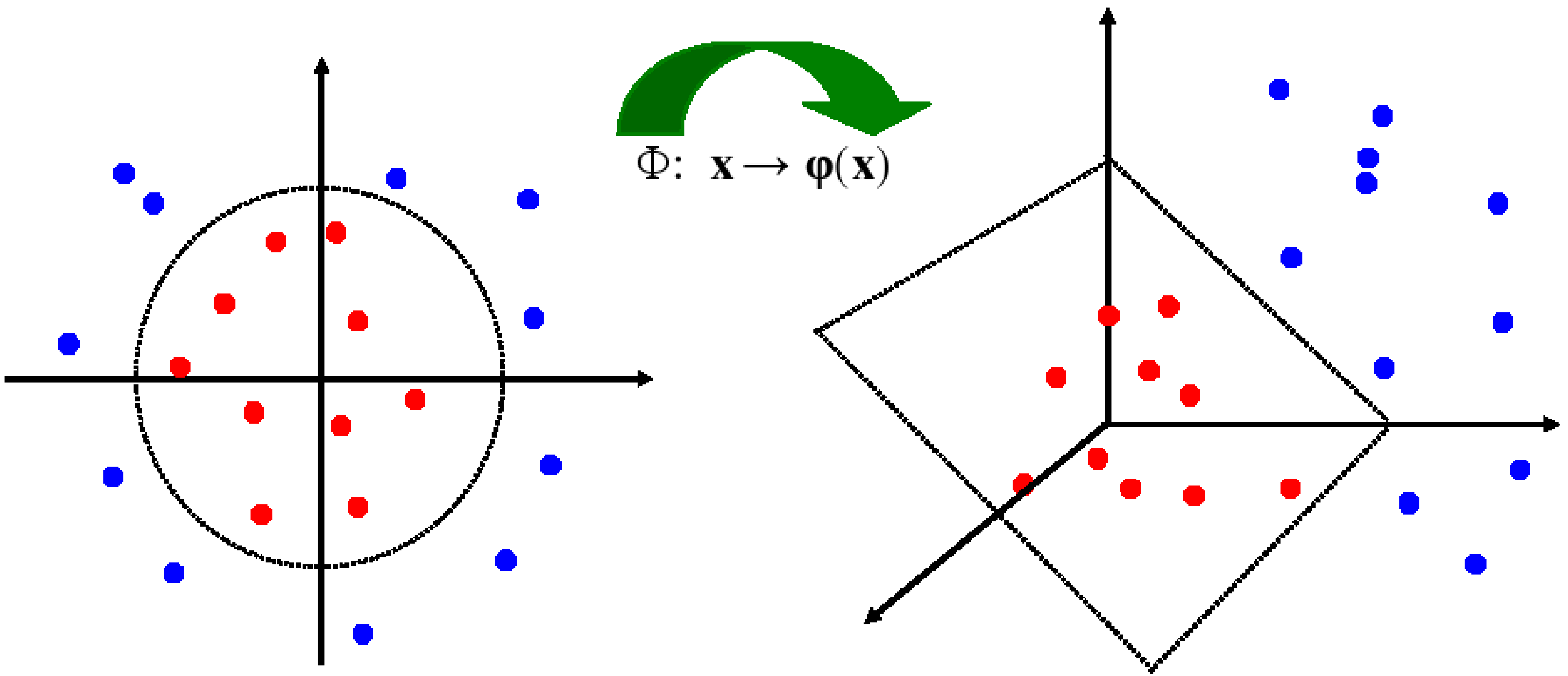
$$\phi(x_1, x_2) = (x_1^2, x_2^2, \sqrt{2}x_1x_2)$$





Nonlinear (Kernel) SVM

- General idea: the original feature 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 inner product between vectors $K(\mathbf{x}_i,\mathbf{x}_j)=\mathbf{x}_i^{\mathbf{T}}\mathbf{x}_j$

If every datapoint is mapped into high-dimensional space via some transformation Φ: $\mathbf{x} \rightarrow \varphi(\mathbf{x})$, the inner product becomes:

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

A *kernel function* is a function that is eqiuvalent to an inner product in some feature space.

Example:

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

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

$$K(\mathbf{x}_i,\mathbf{x}_j)=(1 + \mathbf{x}_i^{\mathbf{T}}\mathbf{x}_j)^2= 1+ x_{i1}^2x_{j1}^2 + 2\, x_{i1}x_{j1}\, x_{i2}x_{j2}+ x_{i2}^2x_{j2}^2 + 2x_{i1}x_{j1} + 2x_{i2}x_{j2}=$$

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

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

Thus, a kernel function *implicitly* maps data to a high-dimensional space (without the need to compute each $\boldsymbol{\varphi}(\mathbf{x})$ explicitly).

What Functions are Kernels?

For some functions $K(\mathbf{x}_i,\mathbf{x}_j)$ checking that $K(\mathbf{x}_i,\mathbf{x}_j)= \varphi(\mathbf{x}_i)^{\mathrm{T}}\varphi(\mathbf{x}_j)$ 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: $K(\mathbf{x}_i,\mathbf{x}_j)= \mathbf{x}_i^{\mathrm{T}}\mathbf{x}_j$

- Mapping Φ: $\mathbf{x} \rightarrow \boldsymbol{\varphi}(\mathbf{x})$, where $\boldsymbol{\varphi}(\mathbf{x})$ is $\mathbf{x}$ itself

Polynomial of power p: $K(\mathbf{x}_i,\mathbf{x}_j)= (1+ \mathbf{x}_i^{\mathrm{T}}\mathbf{x}_j)^p$

- Mapping Φ: $\mathbf{x} \rightarrow \boldsymbol{\varphi}(\mathbf{x})$, where $\boldsymbol{\varphi}(\mathbf{x})$ has $\binom{d+p}{p}$ dimensions

Gaussian (radial-basis function): $K(\mathbf{x}_i,\mathbf{x}_j) = e^{-\frac{\|\mathbf{x}_i-\mathbf{x}_j\|^2}{2\sigma^2}}$

- Mapping Φ: $\mathbf{x} \rightarrow \boldsymbol{\varphi}(\mathbf{x})$, where $\boldsymbol{\varphi}(\mathbf{x})$ is *infinite-dimensional*: every point is mapped to *a function* (a Gaussian); combination of functions for support vectors is the separator.

Higher-dimensional space still has *intrinsic* dimensionality d (the mapping is not *onto*), but linear separators in it correspond to *non-linear* separators in original space.

Non-linear (Kernel) SVM

Dual problem formulation:

Find $\alpha_1 \ldots \alpha_n$ such that

$\mathbf{Q}(\boldsymbol{\alpha}) = \Sigma\alpha_i - \frac{1}{2}\Sigma\Sigma\alpha_i\alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j)$ is maximized and

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

(2) $\alpha_i \geq 0$ for all α_i

The solution is:

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

Optimization techniques for finding α_i's remain the same!

Properties of SVM: Summary

- Duality
- Kernels
- Margin
- Convexity
- Sparseness
- Much more than just a replacement for neural networks
- General and rich class of pattern recognition methods
- Effective for awide range of problems

SVM Toolbox

- LibSVM (http://www.csie.ntu.edu.tw/~cjlin/libsvm/)
  - C/C++ implementation; many interfaces including Python, Matlab, Java etc.
  - Quite handy; easy to use.
- SVMlight (http://svmlight.joachims.org/)
  - C implementation.
  - Fast optimisation algorithm.

SVM Applications

SVMs were originally proposed by Boser, Guyon and Vapnik in 1992 and gained increasing popularity in late 1990s.

SVMs are currently among the best performers for a number of classification tasks ranging from text to genomic data.

SVMs can be applied to complex data types beyond feature vectors (e.g. graphs, sequences, relational data) by designing kernel functions for such data.

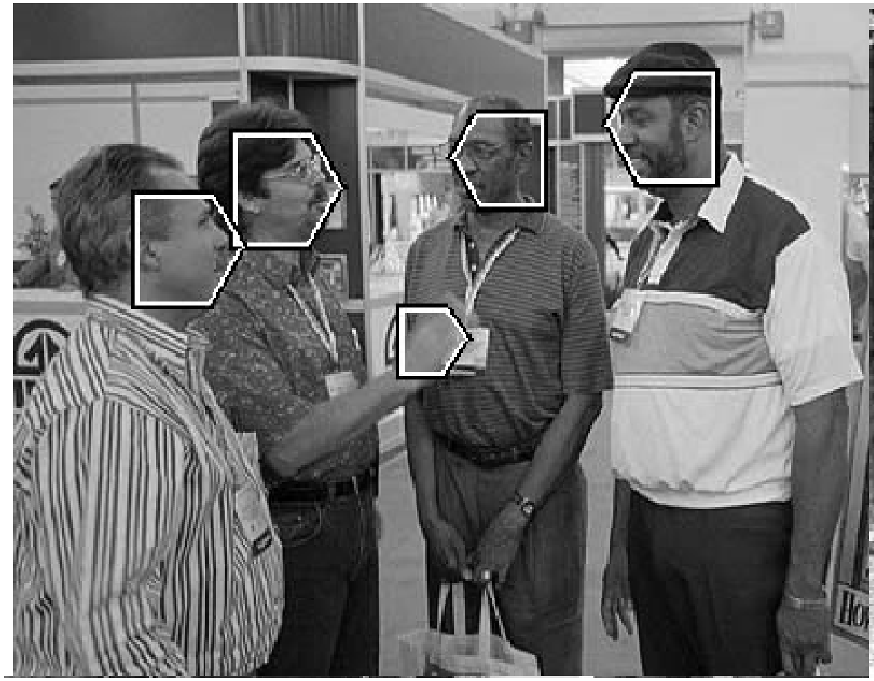
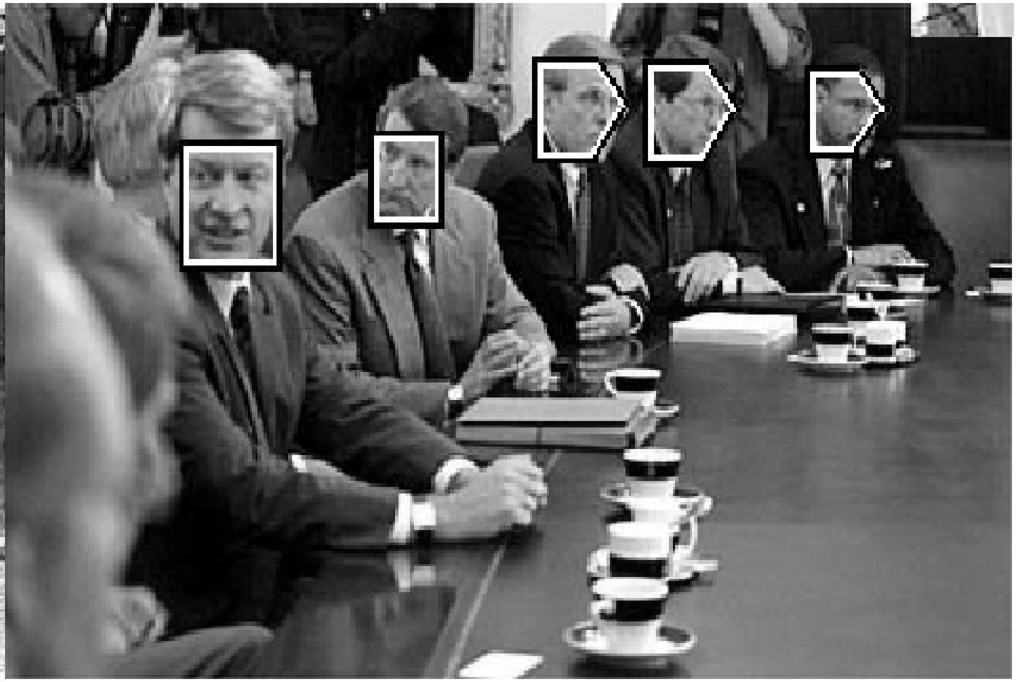
SVM techniques have been extended to a number of tasks such as regression [Vapnik *et al.* '97], principal component analysis [Schölkopf *et al.* '99], etc.

Most popular optimization algorithms for SVMs use *decomposition* to hill-climb over a subset of α_i's at a time, e.g. SMO [Platt '99] and [Joachims '99]

Tuning SVMs remains a black art: selecting a specific kernel and parameters is usually done in a try-and-see manner.

SVM Applications in Vision

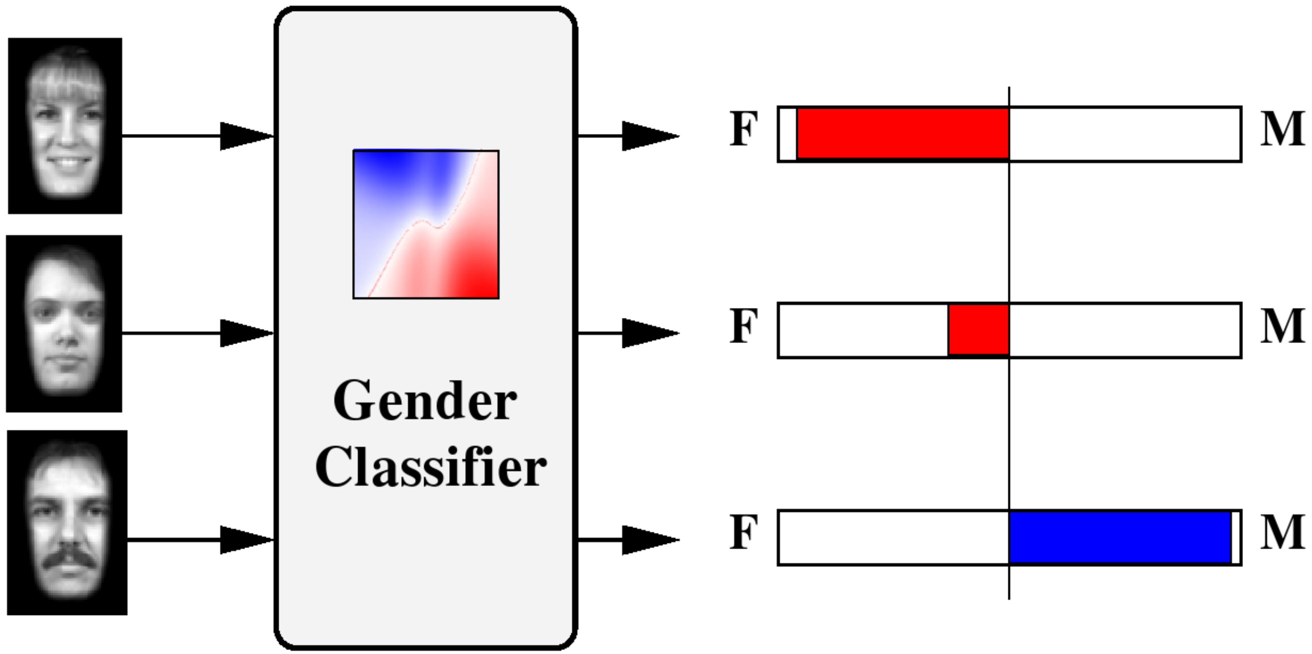
- Face detection



E. Osuna, R. Freund, and F. Girosi, *Training Support Vector Machines: An Application to Face Detection*, Proc. IEEE Conf. Computer Vision and Pattern Recognition, pp. 130-136, 1997.

SVM Applications in Vision

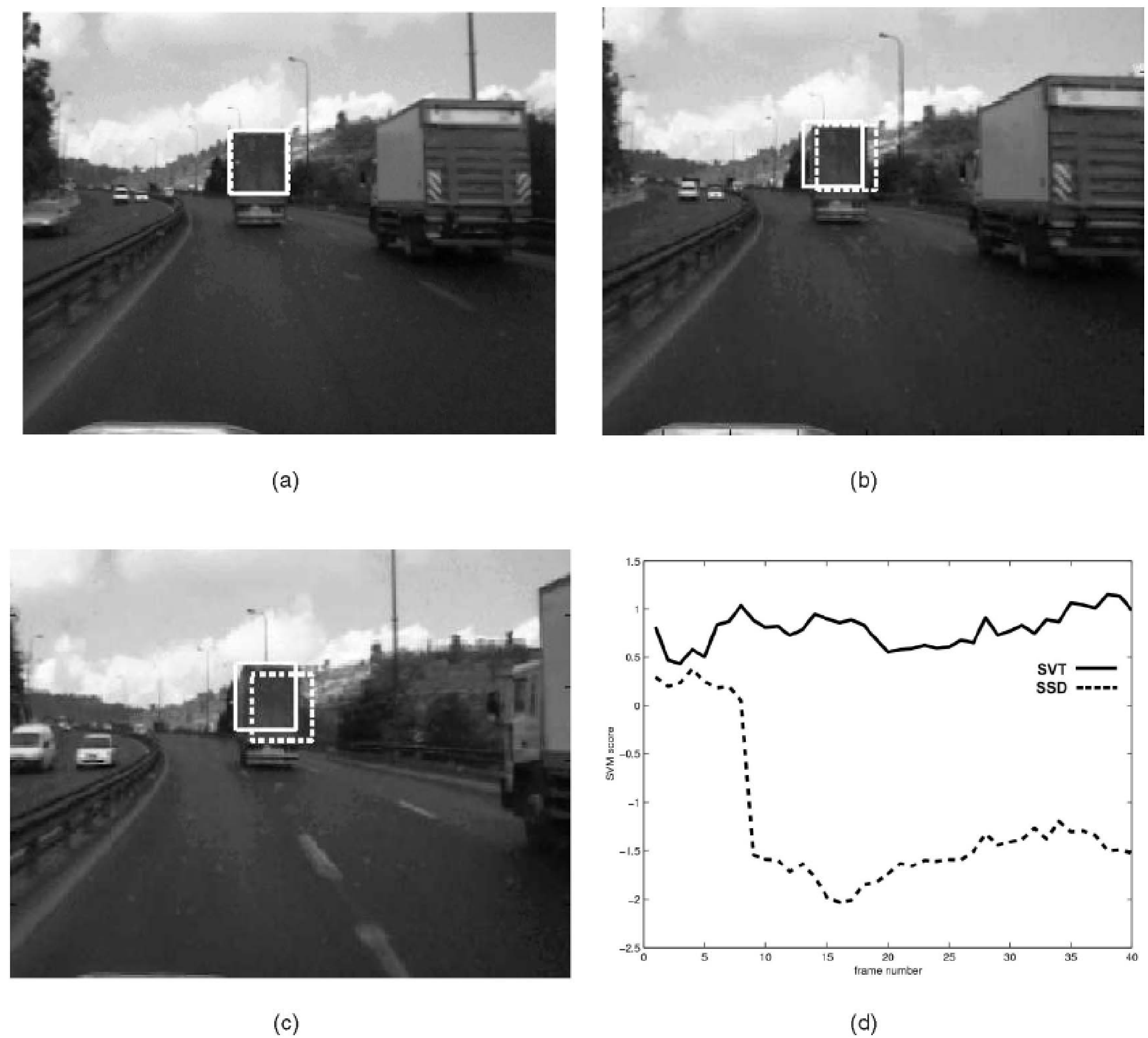
- Sex classification



B. Moghaddam, M.-H. Yang, *Sex with Support Vector Machines*, NIPS 2001.

SVM Applications in Vision

- Support vector tracking



S. Avidan, *Support vector tracking*, TPAMI 2004.

Summary

- A brief introduction to Bayesian inference.
- Discriminant classifiers
- KNN
- SVM