CS 1675 Introduction to Machine Learning
Lecture 11

Topics:

- **Support vector machines (cont)**
- **ROC analysis**
- **Nonparametric methods**

Milos Hauskrecht

milos@cs.pitt.edu

5329 Sennott Square

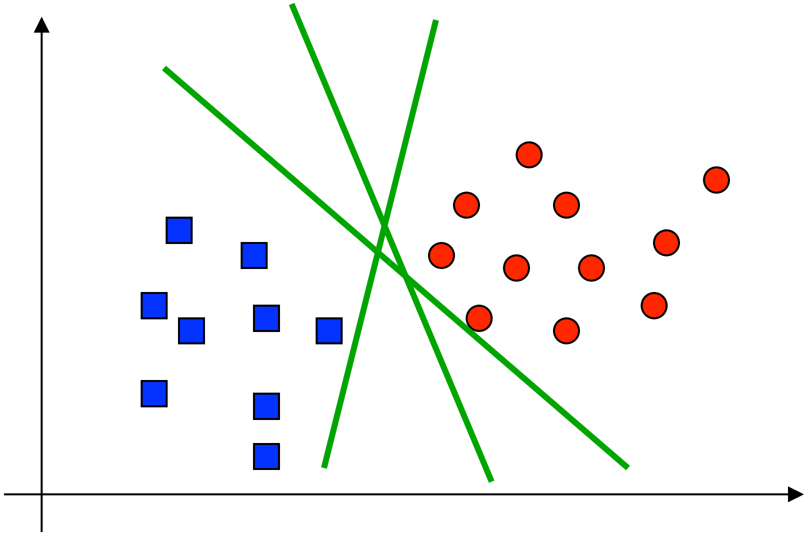
Last lecture outline

Outline:

- Algorithms for linear decision boundary
- **Support vector machines**
- Maximum margin hyperplane
- Support vectors
- Support vector machines learning
- Extensions to the linearly non-separable case
- Kernel functions

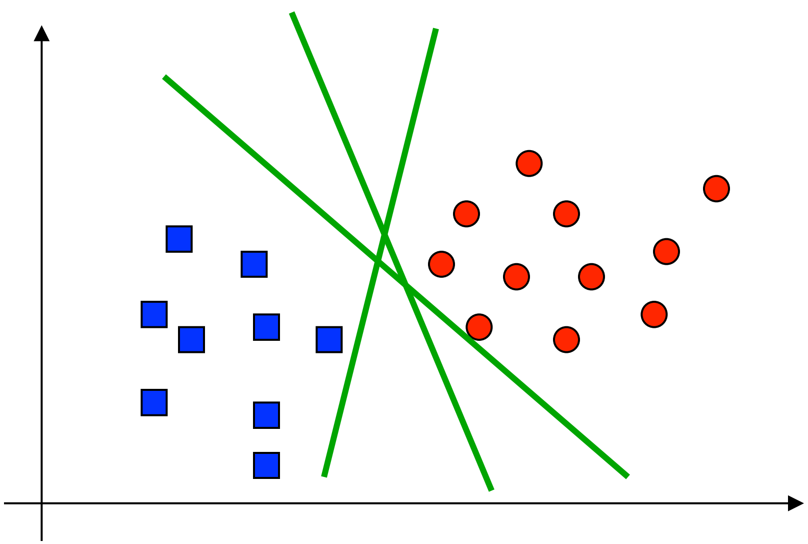
Optimal separating hyperplane

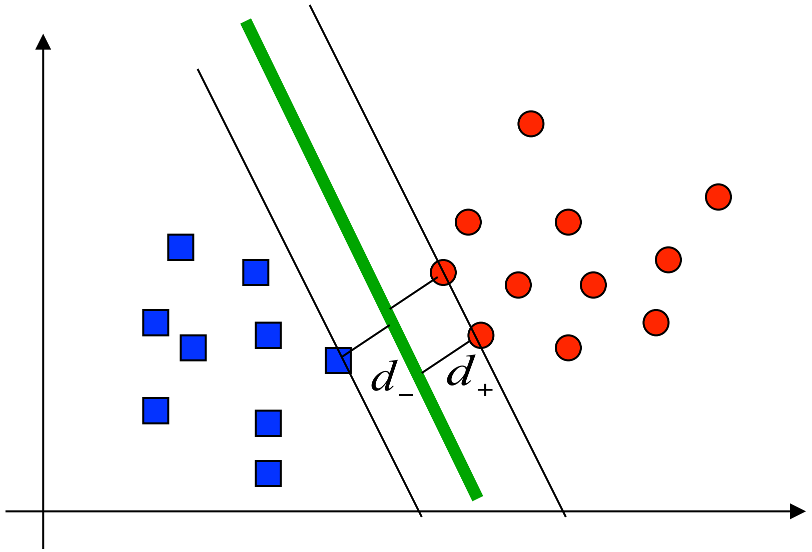
- **Problem:**
- There are multiple hyperplanes that separate the data points
- Which one to choose?



Optimal separating hyperplane

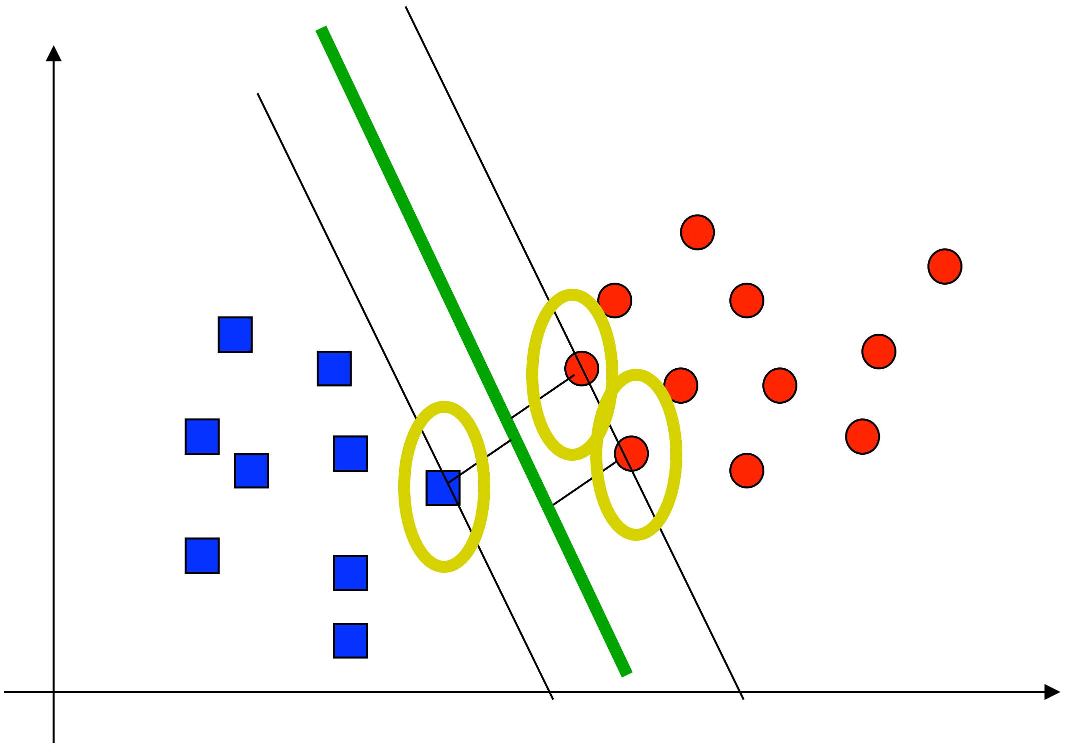
- **Problem:**
- There are multiple hyperplanes that separate the data points
- Which one to choose?
- The decision boundary that maximizes the distance of the +1 and -1 points from it





Maximum margin hyperplane

- For the maximum margin hyperplane only examples on the margin matter (only these affect the distances)
- These are called **support vectors**



Support vector machines: solution property

- **Decision boundary defined by a set of support vectors SV and their alpha values**
  - **Support vectors = a subset of datapoints in the training data that define the margin**

$$\hat{\mathbf{w}}^T \mathbf{x} + w_0 = \sum_{i \in SV} \hat{\alpha}_i y_i (\mathbf{x}_i^T \mathbf{x}) + w_0$$

Lagrange multipliers

- **Classification decision:**

$$\hat{y} = \operatorname{sign}\left[\sum_{i \in SV} \hat{\alpha}_i y_i (\mathbf{x}_i^T \mathbf{x}) + w_0 \right]$$

- **Note that we do not have to explicitly compute** $\hat{\mathbf{w}}$
  - This will be important for the nonlinear (kernel) case

Support vector machines: inner product

- Decision on a new **x** depends on the **inner product between two examples**
- **The decision boundary:**

$$\hat{\mathbf{w}}^T\mathbf{x} + w_0 = \sum_{i \in SV} \hat{\alpha}_i y_i (\mathbf{x}_i^T \mathbf{x}) + w_0$$

- **Classification decision:**

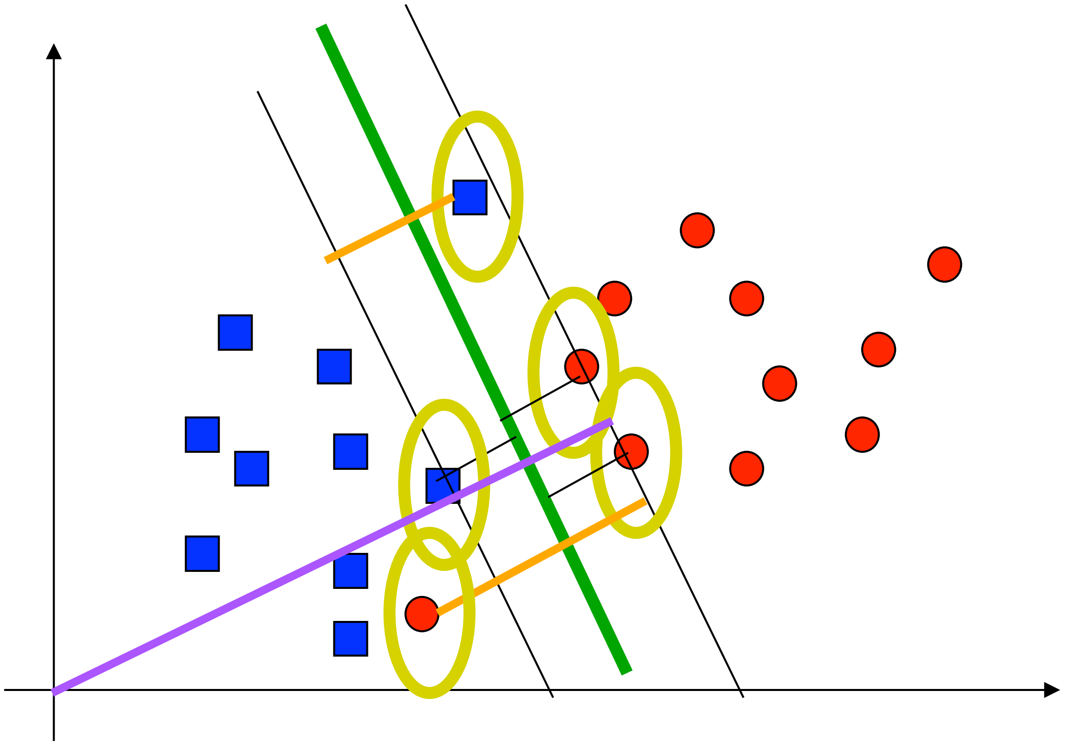
$$\hat{y} = \text{sign}\left[\sum_{i \in SV} \hat{\alpha}_i y_i (\mathbf{x}_i^T \mathbf{x}) + w_0\right]$$

- Similarly, the optimization depends on $(\mathbf{x}_i^T \mathbf{x}_j)$

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

Linearly non-separable case

- **Idea:** Allow some flexibility on crossing the separating hyperplane



Support vector machines: solution

- **The solution of the linearly non-separable case has the same properties as the linearly separable case.**
  - The decision boundary is defined only by a set of support vectors (points that are on the margin or that cross the margin)
  - The decision boundary and the optimization can be expressed in terms of the inner product in between pairs of examples

$$\hat{\mathbf{w}}^T \mathbf{x} + w_0 = \sum_{i \in SV} \hat{\alpha}_i y_i (\mathbf{x}_i^T \mathbf{x}) + w_0$$

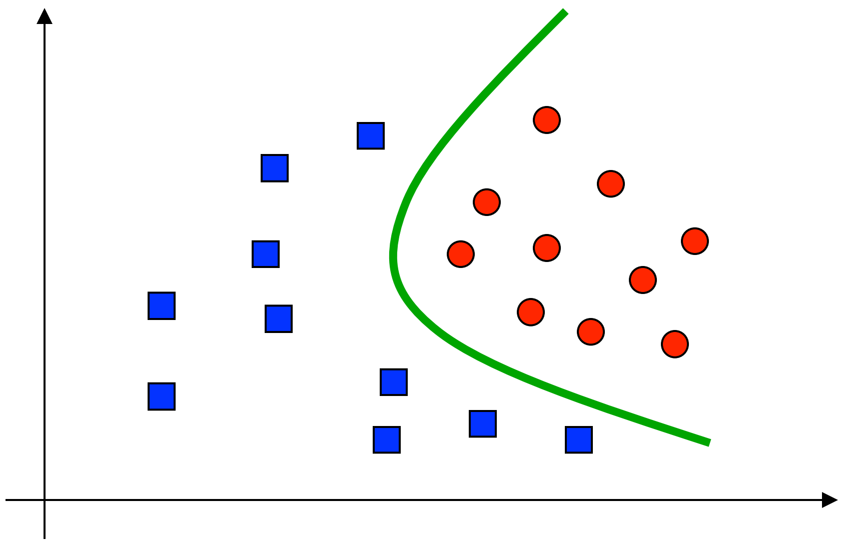
$$\hat{y} = \operatorname{sign}\left[\hat{\mathbf{w}}^T \mathbf{x} + w_0\right] = \operatorname{sign}\left[\sum_{i \in SV} \hat{\alpha}_i y_i (\mathbf{x}_i^T \mathbf{x}) + w_0\right]$$

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

Nonlinear decision boundary

So far we have seen how to learn a linear decision boundary

- **But what if the linear decision boundary is not good.**
- **How we can learn a non-linear decision boundaries with the SVM?**



Nonlinear decision boundary

- The non-linear case can be handled by using a set of features. Essentially we map input vectors to (larger) feature vectors

$$\mathbf{x} \rightarrow \boldsymbol{\varphi}(\mathbf{x})$$

  - **Example:** polynomial expansions
  - Note that feature expansions are typically high dimensional

- Given the nonlinear feature mappings, we can use the linear SVM on the expanded feature vectors

$$(\mathbf{x}^T \mathbf{x}') \longrightarrow \boldsymbol{\varphi}(\mathbf{x})^T \boldsymbol{\varphi}(\mathbf{x}')$$

- **Kernel function (measures similarity)**

$$K(\mathbf{x}, \mathbf{x}') = \boldsymbol{\varphi}(\mathbf{x})^T \boldsymbol{\varphi}(\mathbf{x}')$$

Support vector machines: solution for nonlinear decision boundaries

- **The decision boundary:**

$$\hat{\mathbf{w}}^T \mathbf{x} + w_0 = \sum_{i \in SV} \hat{\alpha}_i y_i K(\mathbf{x}_i, \mathbf{x}) + w_0$$

- **Classification:**

$$\hat{y} = \operatorname{sign}\left[\hat{\mathbf{w}}^T \mathbf{x} + w_0\right] = \operatorname{sign}\left[\sum_{i \in SV} \hat{\alpha}_i y_i K(\mathbf{x}_i, \mathbf{x}) + w_0\right]$$

- Decision on a new **x** requires to compute **the kernel function defining the similarity between the examples**
- Similarly, the optimization depends on the kernel

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

Kernel trick

- **Feature mapping:**

$$\mathbf{x} \rightarrow \boldsymbol{\varphi}(\mathbf{x})$$

- **Kernel function** defines the inner product in the expanded high dimensional feature vectors and let us use the SVM

$$K(\mathbf{x}, \mathbf{x}') = \boldsymbol{\varphi}(\mathbf{x})^T \boldsymbol{\varphi}(\mathbf{x}')$$

- **Problem:** after expansion we need to perform inner products in a very high dimensional $\boldsymbol{\varphi}(\mathbf{x})$ space

- **Kernel trick:**
  - If we choose the kernel function **K**(**x**,**x**') wisely we can compute linear separation in the high dimensional feature space implicitly by working in the original input space !!!!

Kernel function example

- Assume $\mathbf{x} = [x_1, x_2]^T$ and a feature mapping that maps the input into a quadratic feature set

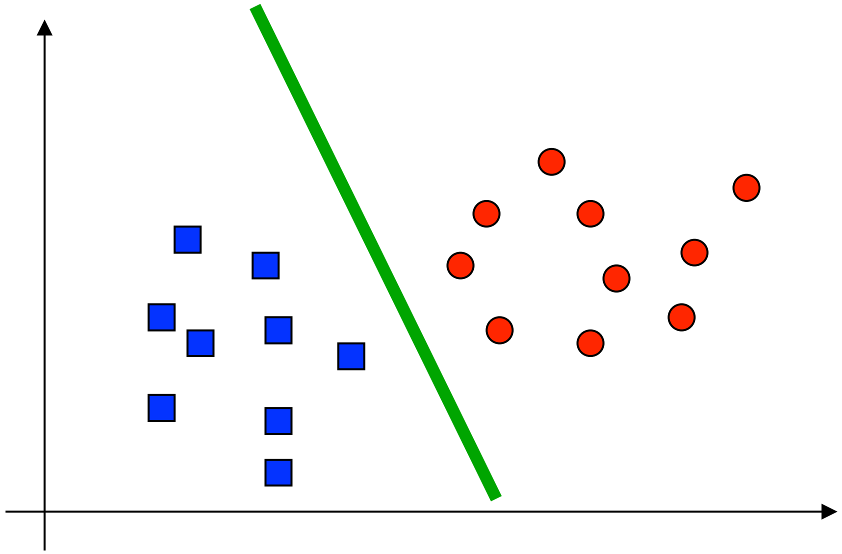
$$\mathbf{x} \rightarrow \boldsymbol{\varphi}(\mathbf{x}) = [x_1^2, x_2^2, \sqrt{2}x_1x_2, \sqrt{2}x_1, \sqrt{2}x_2, 1]^T$$

- Kernel function for the feature space:

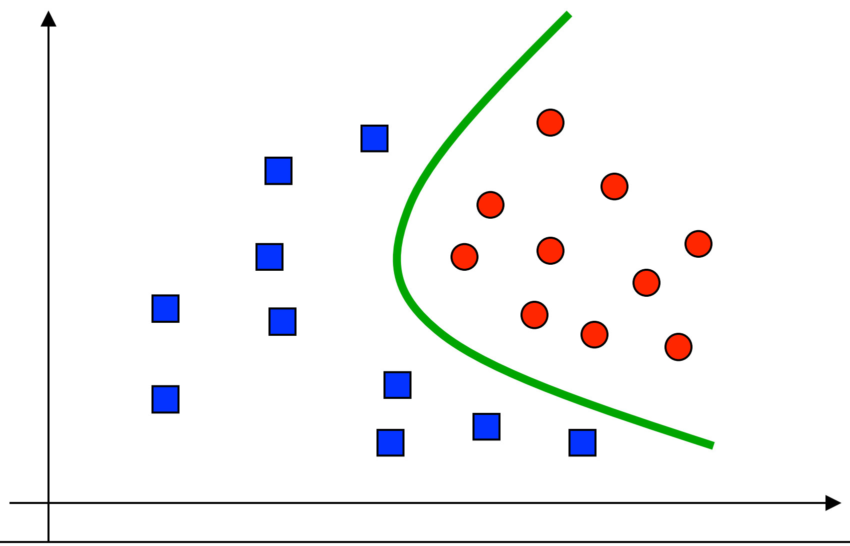
$$\begin{aligned} K(\mathbf{x}', \mathbf{x}) &= \boldsymbol{\varphi}(\mathbf{x}')^T \boldsymbol{\varphi}(\mathbf{x}) \\ &= x_1^2 x'^2_1 + x_2^2 x'^2_2 + 2x_1x_2x'_1\, x'_2 + 2x_1x'_1 + 2x_2x'_2 + 1 \\ &= (x_1x'_1 + x_2x'_2 + 1)^2 \\ &= (1 + (\mathbf{x}^T\mathbf{x}'))^2 \end{aligned}$$

- The computation of the linear separation in the higher dimensional space is performed implicitly in the original input space

Kernel function example



Linear separator
in the expanded
feature space



Non-linear separator
in the input space

Kernel functions

- **Linear kernel**

$$K(\mathbf{x}, \mathbf{x}') = \mathbf{x}^T \mathbf{x}'$$

- **Polynomial kernel**

$$K(\mathbf{x}, \mathbf{x}') = \left[1 + \mathbf{x}^T \mathbf{x}'\right]^k$$

- **Radial basis kernel**

$$K(\mathbf{x}, \mathbf{x}') = \exp\left[-\frac{1}{2}\left\|\mathbf{x} - \mathbf{x}'\right\|^2\right]$$

Kernels

- ML researchers have proposed kernels for comparison of variety of objects.
  - Strings
  - Trees
  - Graphs
- **Cool thing:**
  - SVM algorithm can be now applied to classify a variety of objects

Evaluation of binary classifiers
ROC analysis

Evaluation

For any data set we use to test the classification model on we can build a **confusion matrix:**

- Counts of examples with:
- class label ω_j that are classified with a label α_i

target

predict		$\omega = 1$	$\omega = 0$
	$\alpha = 1$	140	17
	$\alpha = 0$	20	54

Evaluation

For any data set we use to test the model we can build a confusion matrix:

target

		$\omega = 1$	$\omega = 0$
predict	$\alpha = 1$	140	17
	$\alpha = 0$	20	54

Accuracy = 194/231

Evaluation

For any data set we use to test the model we can build a confusion matrix:

target

		$\omega = 1$	$\omega = 0$
predict	$\alpha = 1$	140	17
	$\alpha = 0$	20	54

Accuracy = 194/231

Error = 37/231 = 1 - **Accuracy**

Evaluation for binary classification

Entries in the confusion matrix for binary classification have names:

target

		$\omega = 1$	$\omega = 0$
predict	$\alpha = 1$	TP	FP
	$\alpha = 0$	FN	TN

TP: True positive (hit)
FP: False positive (false alarm)
TN: True negative (correct rejection)
FN: False negative (a miss)

Additional statistics

- **Sensitivity (recall)**

$$SENS = \frac{TP}{TP + FN}$$

- **Specificity**

$$SPEC = \frac{TN}{TN + FP}$$

- **Positive predictive value (precision)**

$$PPT = \frac{TP}{TP + FP}$$

- **Negative predictive value**

$$NPV = \frac{TN}{TN + FN}$$

Binary classification: additional statistics

- **Confusion matrix**

		target		
		1	0	
predict	1	140	10	$PPV = 140/150$
	0	20	180	$NPV = 180/200$
		$SENS = 140/160$	$SPEC = 180/190$	

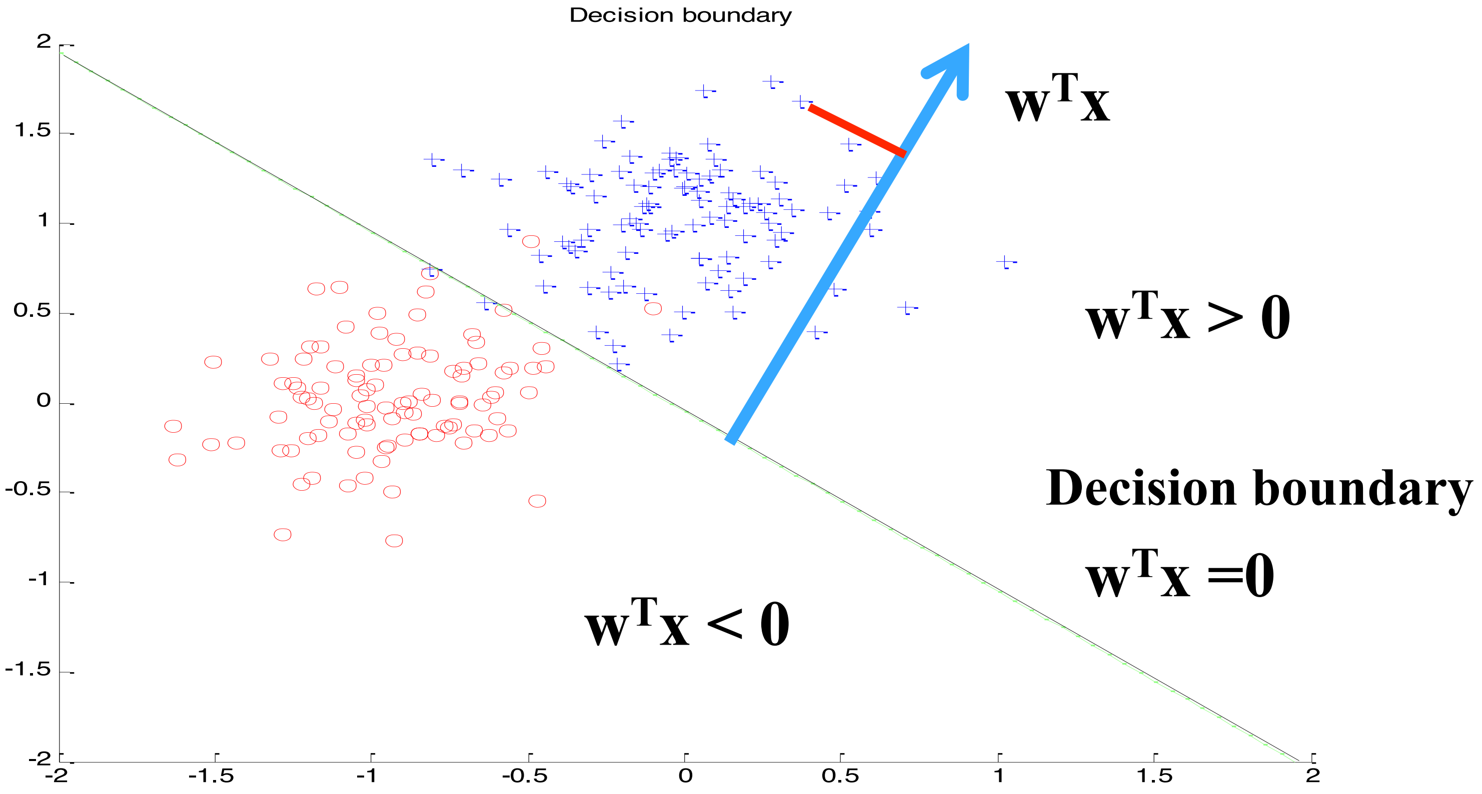
Row and column quantities:

- Sensitivity (SENS)
- Specificity (SPEC)
- Positive predictive value (PPV)
- Negative predictive value (NPV)

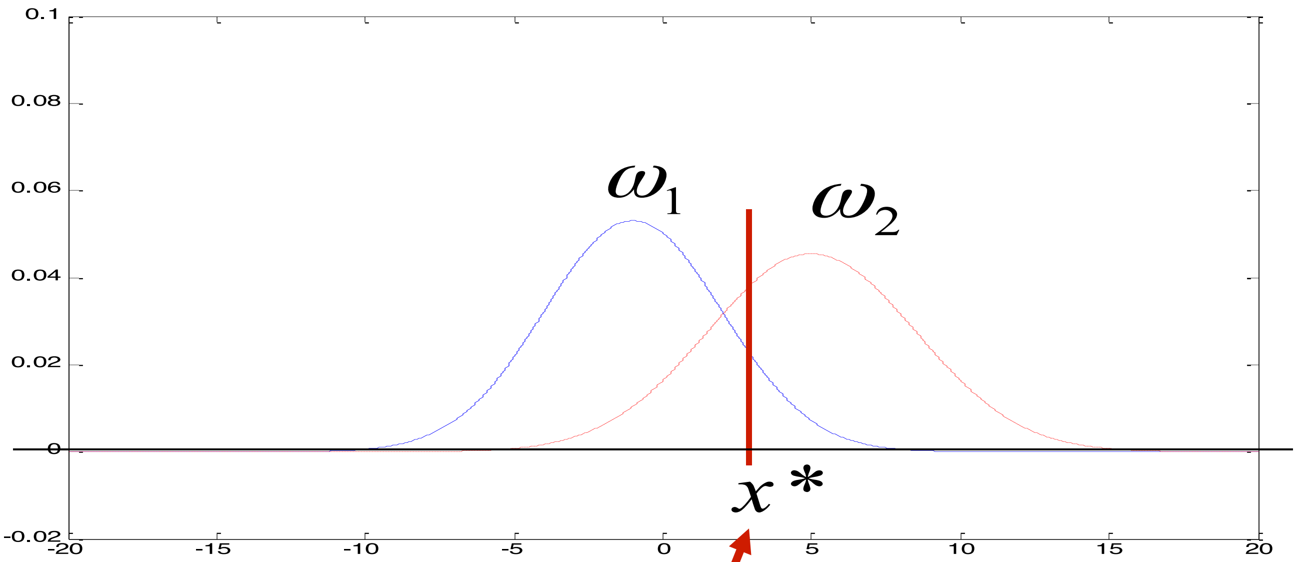
Classifiers

Project datapoints to one dimensional space:

Defined for example by: $\mathbf{w}^T\mathbf{x}$ or p(y=1|**x**,**w**)



Binary decisions: Receiver Operating Curves



- **Probabilities:**
  - *SENS*
  - *SPEC*

threshold

$$p(x > x^* \mid \mathbf{x} \in \omega_2)$$

$$p(x < x^* \mid \mathbf{x} \in \omega_1)$$

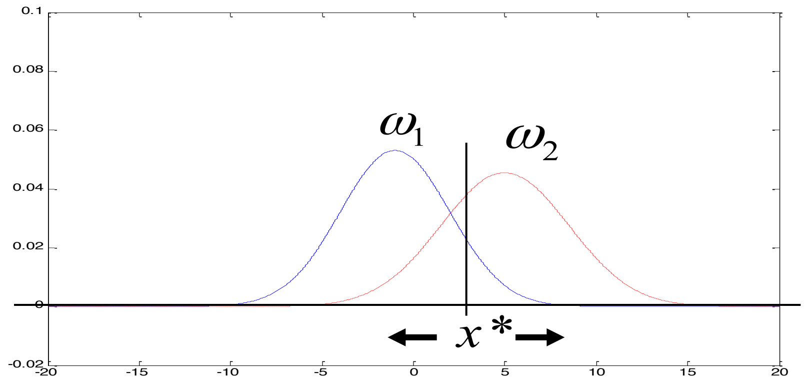
Receiver Operating Characteristic (ROC)

- **ROC curve plots :**

SN= $p(x > x^* \mid \mathbf{x} \in \omega_2)$

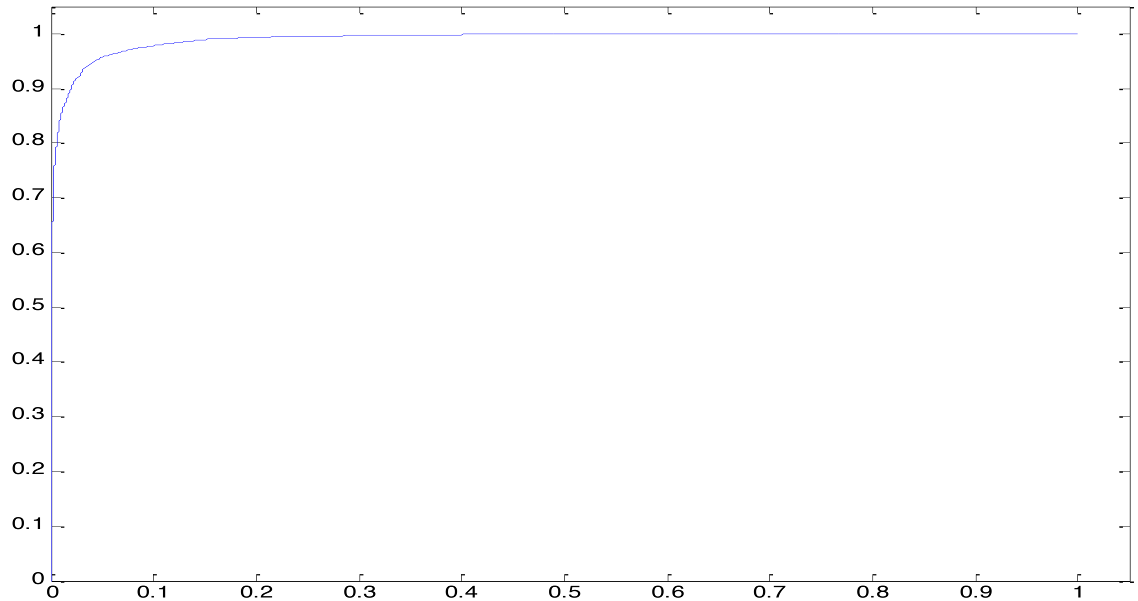
1-SP= $p(x > x^* \mid \mathbf{x} \in \omega_1)$

for different x*



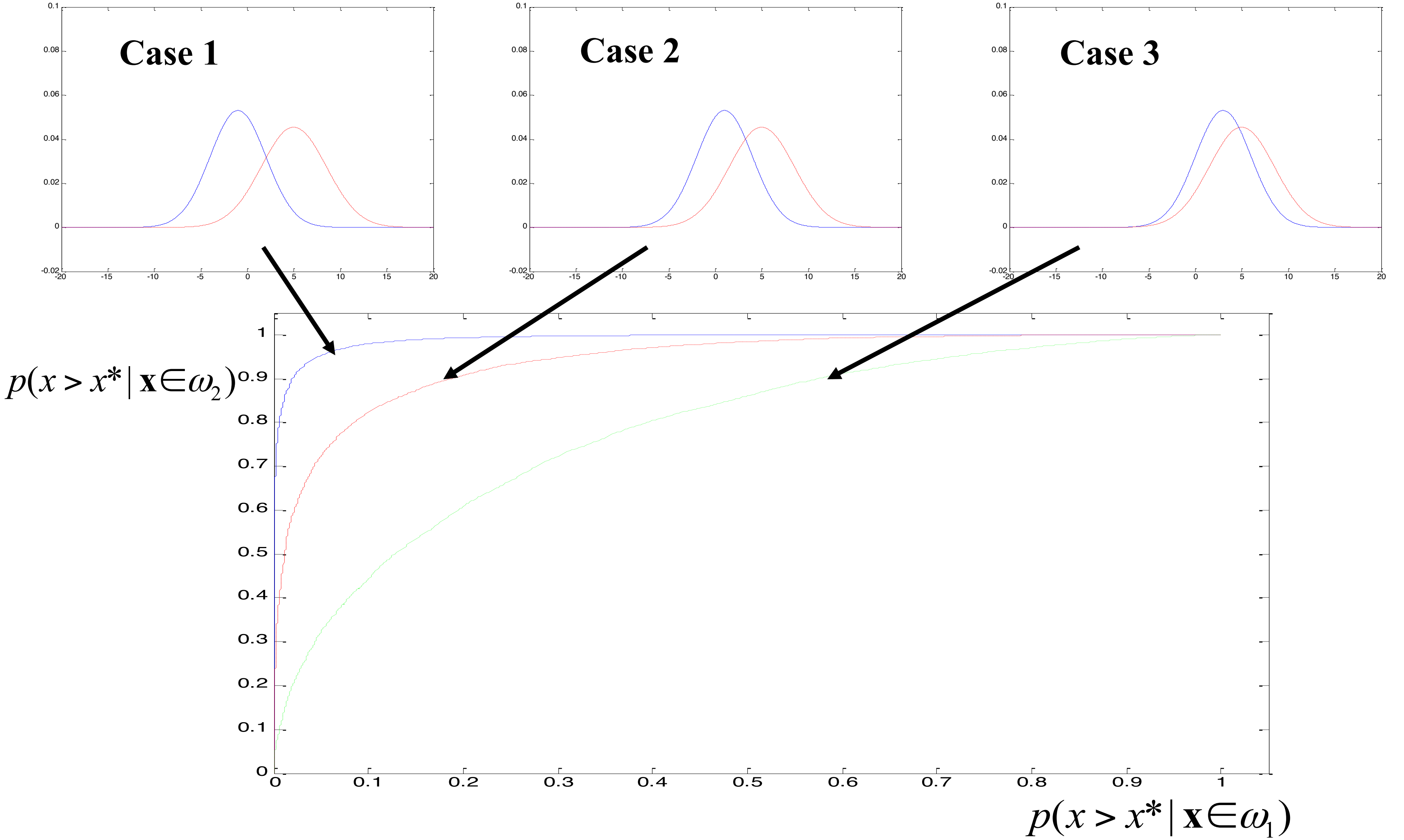
SENS

$p(x > x^* \mid \mathbf{x} \in \omega_2)$



1-SPEC $p(x > x^* \mid \mathbf{x} \in \omega_1)$

ROC curve



Case 1
Case 2
Case 3
$p(x > x^* \mid \mathbf{x} \in \omega_2)$
$p(x > x^* \mid \mathbf{x} \in \omega_1)$

Receiver operating characteristic

- **ROC**
  - shows the discriminability between the two classes under different decision biases
- **Decision bias**
  - can be changed using different loss function

- **Quality of a classification model:**
  - Area under the ROC
  - Best value 1, worst (no discriminability): 0.5

Nonparametric Methods

- **Parametric distribution models** are:
  - restricted to specific forms, which may not always be suitable;
  - Example: modelling a multimodal distribution with a single, unimodal model.
- **Nonparametric approaches:**
  - make few assumptions about the overall shape of the distribution being modelled.

Nonparametric Methods

Nonparametric Density Methods

Problem:

- We have a set D of data-points $\mathbf{x}_i$ for i = 1, 2, … n
- We want to calculate p($\mathbf{x}$) for a target value of $\mathbf{x}$

Parametric approach:

- represents p($\mathbf{x}$) using a parametric density model with parameters θ
- fits the parameters θ wrt the data

Nonparametric approach:

- Does not make any parametric assumption
- Estimates p($\mathbf{x}$) from all datapoints in D, as if all D are parameters

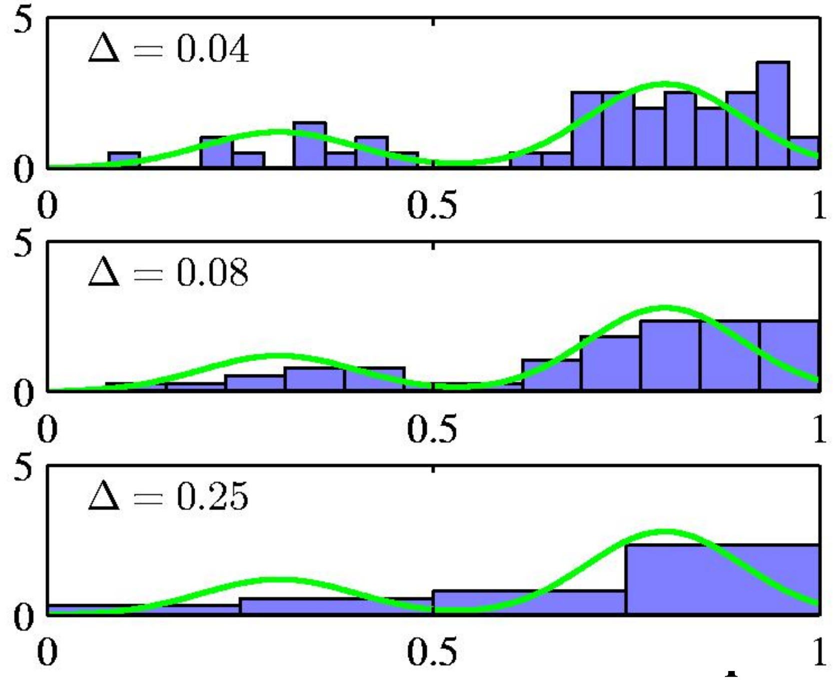
Nonparametric Density Methods

Histogram methods:

partition the data space into distinct bins with widths Δ_i and count the number of observations, n_i, in each bin.

$$p_i = \frac{n_i}{N\Delta_i}$$

- Often, the same width is used for all bins, $\Delta_i = \Delta$.
- Δ acts as a smoothing parameter.



M^D bins!

Nonparametric Density Methods

- Assume observations drawn from a density p(x) and consider a small region R containing x such that

$$P = \int_R p(x)dx$$

- The probability that K out of N observations lie inside R is *Bin(K,N,P)* and if N is large

$$K \cong NP$$

If the volume of R, *V*, is sufficiently small, p(x) is approximately constant over R and

$$P \cong p(x)V$$

Thus

$$p(x) = \frac{P}{V}$$

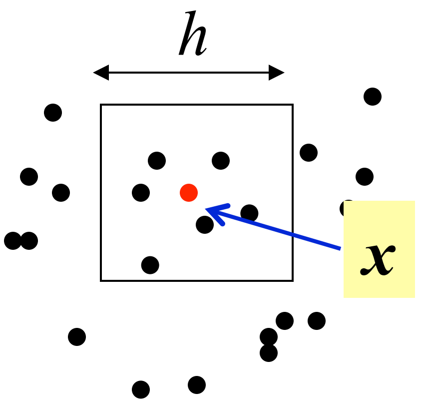
$$\boxed{p(x) = \frac{K}{NV}}$$

Nonparametric Methods: kernel methods

Kernel Density Estimation:

Fix V, estimate K from the data. Let R be a hypercube centred on **x** and define the kernel function (Parzen window)

$$k\left(\frac{x-x_n}{h}\right) = \begin{cases} 1 & |(x_i - x_{ni})|/h \le 1/2 \qquad i = 1,\ldots D \\ 0 & otherwise \end{cases}$$



- **It follows that**
- **and hence** $$K = \sum_{n=1}^{N} k\left(\frac{x-x_n}{h}\right)$$

$$p(x) = \frac{1}{N}\sum_{n=1}^{N}\frac{1}{h^D}k\left(\frac{x-x_n}{h}\right)$$

Nonparametric Methods: smooth kernels

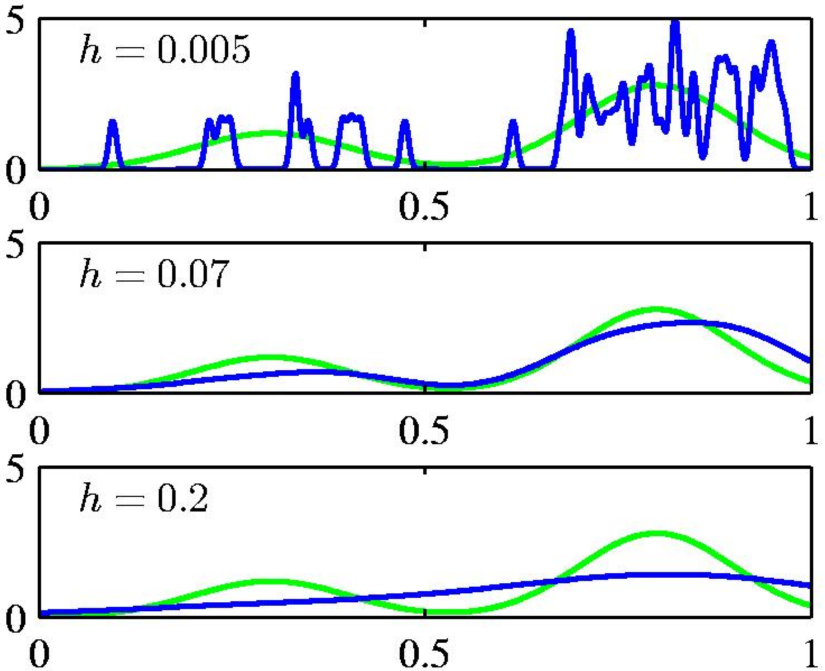
To avoid discontinuities in p(x) because of sharp boundaries use a **smooth kernel**, e.g. a Gaussian

$$p(\mathbf{x}) = \frac{1}{N}\sum_{n=1}^{N}\frac{1}{(2\pi h^2)^{D/2}}\exp\left\{-\frac{\|\mathbf{x}-\mathbf{x}_n\|^2}{2h^2}\right\}$$

- Any kernel such that

$$k(\mathbf{u}) \geqslant 0,$$
$$\int k(\mathbf{u})\,\mathrm{d}\mathbf{u} = 1$$

- will work.



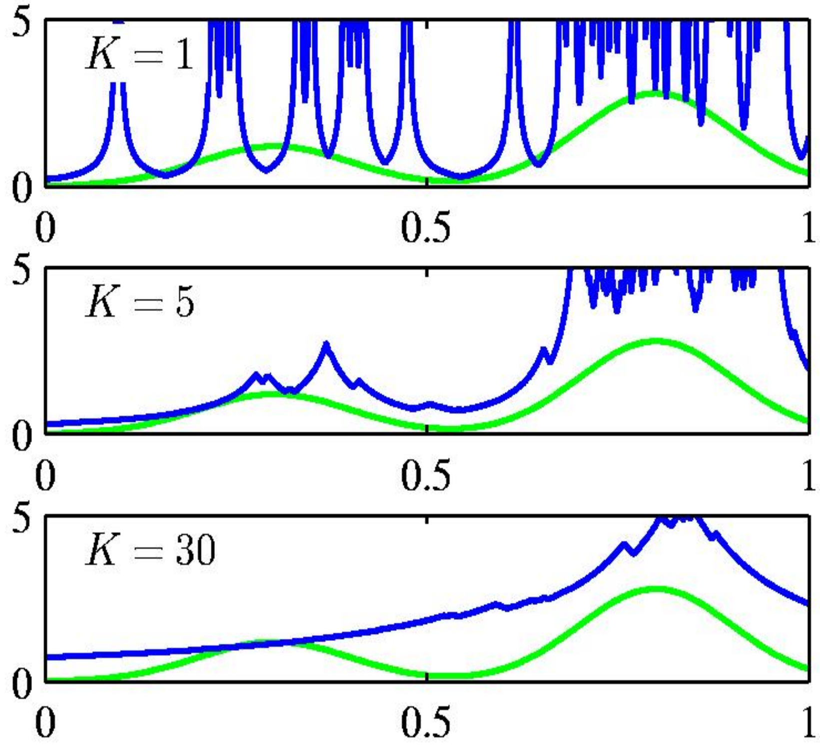
h acts as a smoother.

Nonparametric Methods: kNN estimation

Nearest Neighbour Density Estimation:

fix K, estimate V from the data. Consider a hyper-sphere centred on x and let it grow to a volume, V*, that includes K of the given N data points. Then

$$p(\mathbf{x}) \simeq \frac{K}{NV^\star}.$$



K acts as a smoother

Nonparametric vs Parametric Methods

Nonparametric models:

- More flexibility – no density model is needed
- But require storing the entire dataset
- and the computation is performed with all data examples.

Parametric models:

- Once fitted, only parameters need to be stored
- They are much more efficient in terms of computation
- But the model needs to be picked in advance

Nonparametric classification models

We have a set D of <**x**,y> pairs

We have a new data point **x** and want to assign it a class y

How ?

Algorithm 1

Step 1: Estimate p(y=1) and p(y=0)

Step 2: Estimate $p(\mathbf{x} \mid y=1)$ and $p(\mathbf{x} \mid y=0)$ using nonparametric estimation methods and labels

Step 3: choose a class by comparing

$$p(\mathbf{x} \mid y=1)\, p(y=1) \quad \text{with} \quad p(\mathbf{x} \mid y=0)\, p(y=1)$$

Nonparametric classification models

We have a set D of <**x**,y> pairs

We have a new data point **x** and want to assign it a class y

How ?

Algorithm 2 (K nearest neighbors)

Step 1: Find the closest K examples to x

Step 2: choose a class by considering the majority of the class labels

A special case: **the nearest neighbour algorithm**