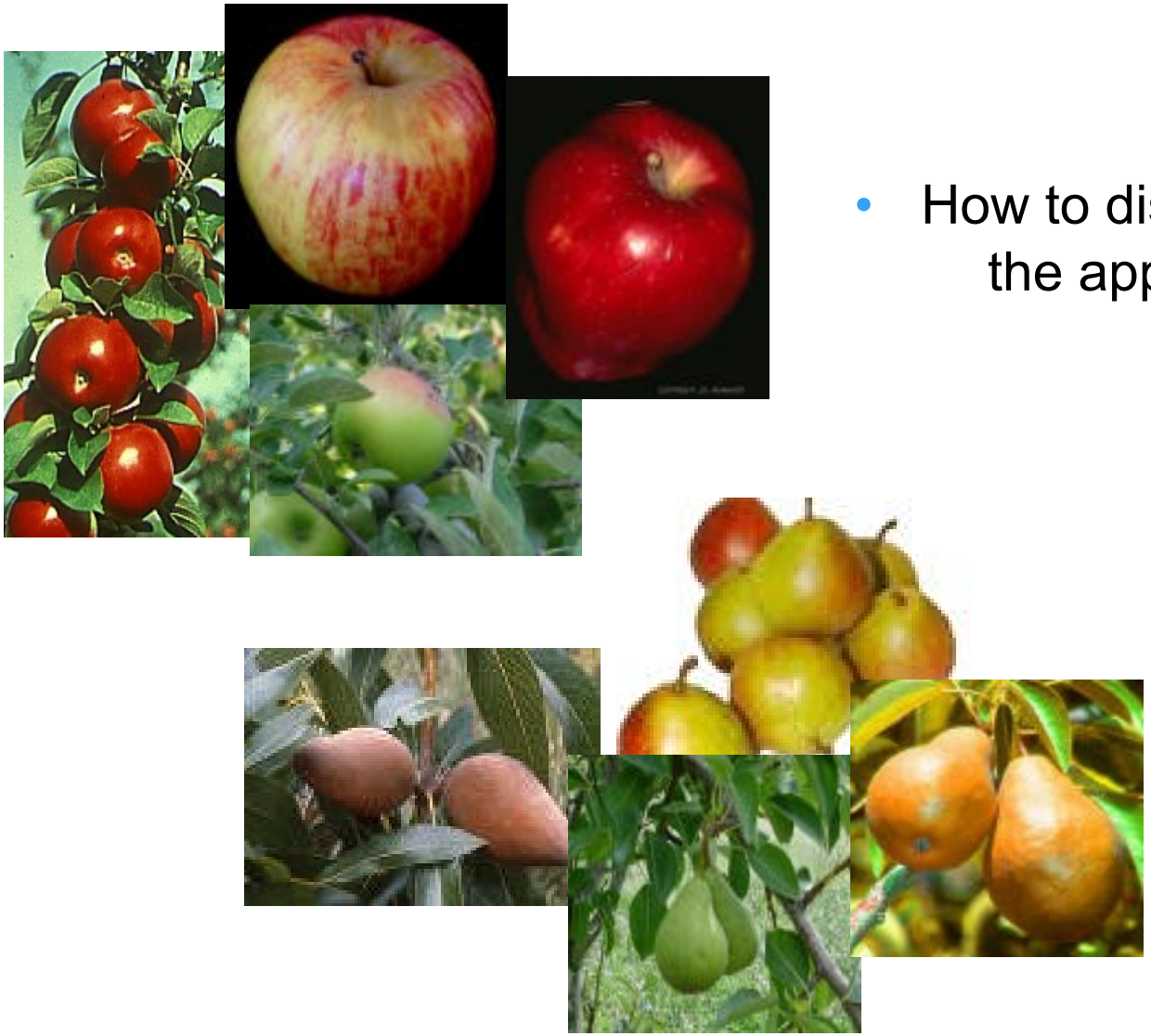
Machine Learning for Bioinformatics & Systems Biology

2. Classification

Perry Moerland *Amsterdam UMC, University of Amsterdam*
Marcel Reinders *Delft University of Technology*
Lodewyk Wessels *Netherlands Cancer Institute*

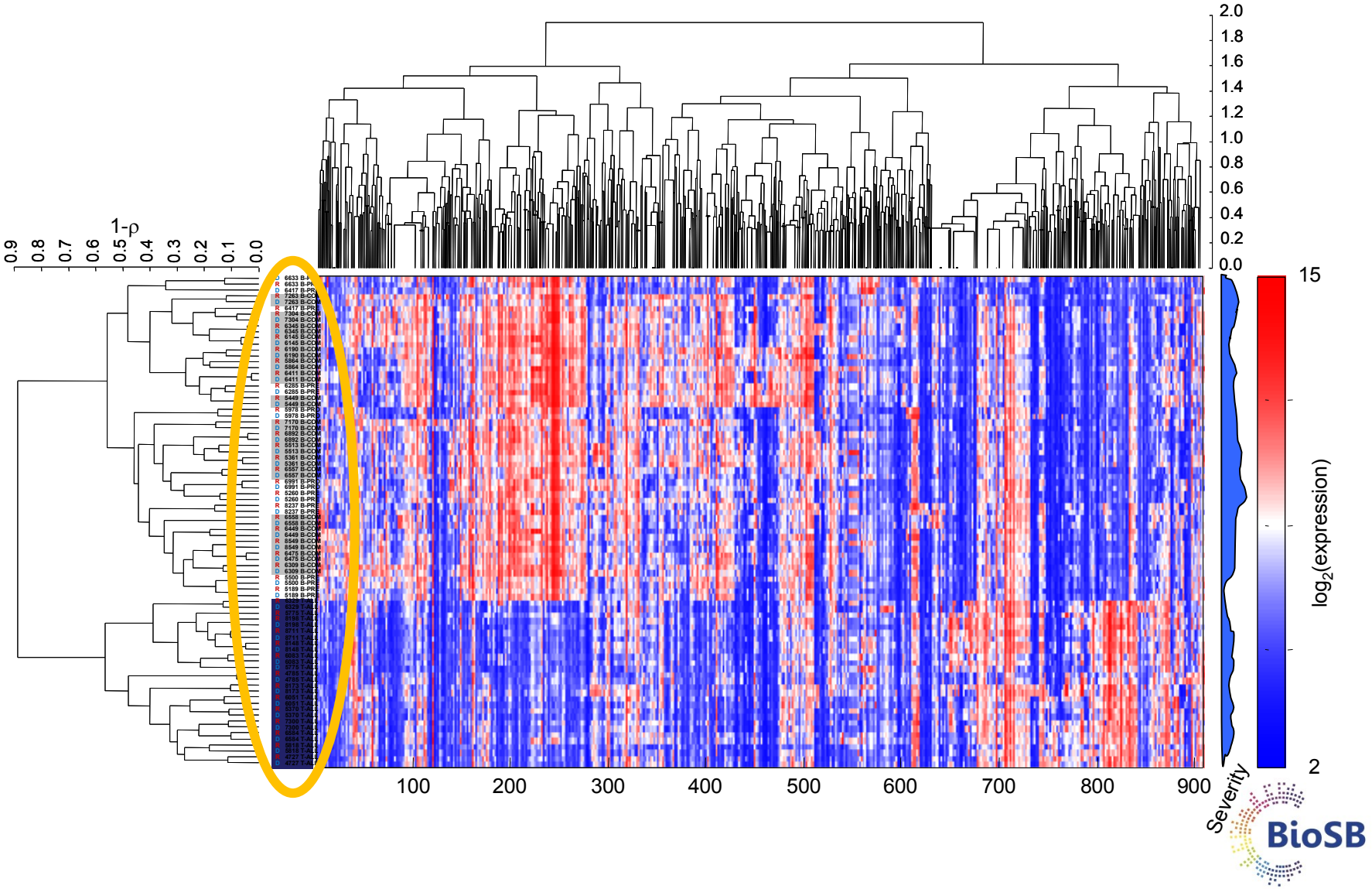
Some material courtesy of Robert Duin and David Tax

Classification



- How to distinguish between the apples and the pears?

Classification in bioinformatics



Classification in bioinformatics (2)

- Secondary structure prediction

  amino acids of a protein sequence → {H,E,-} = {alpha helix,beta strand,turn}

- Protein localization prediction

  {sequence,...} → {cell organelle}

- Genome annotation

  {sequence,...} → {exon,intron,splice site, …}

- ...

BioSB

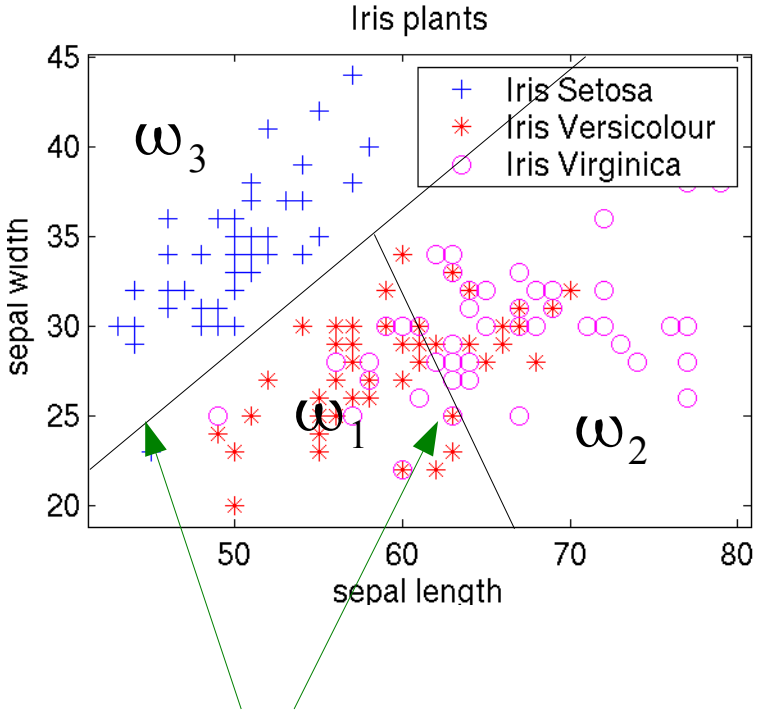
Classification (2)

- Formulation of two-class problems
- Logistic classifier
- Plug-in Bayes classifiers
  - Density-based classification: Parzen, nearest neighbour, Gaussian
- Linear discriminant analysis
  - Fisher classifier
- Decision trees and random forests

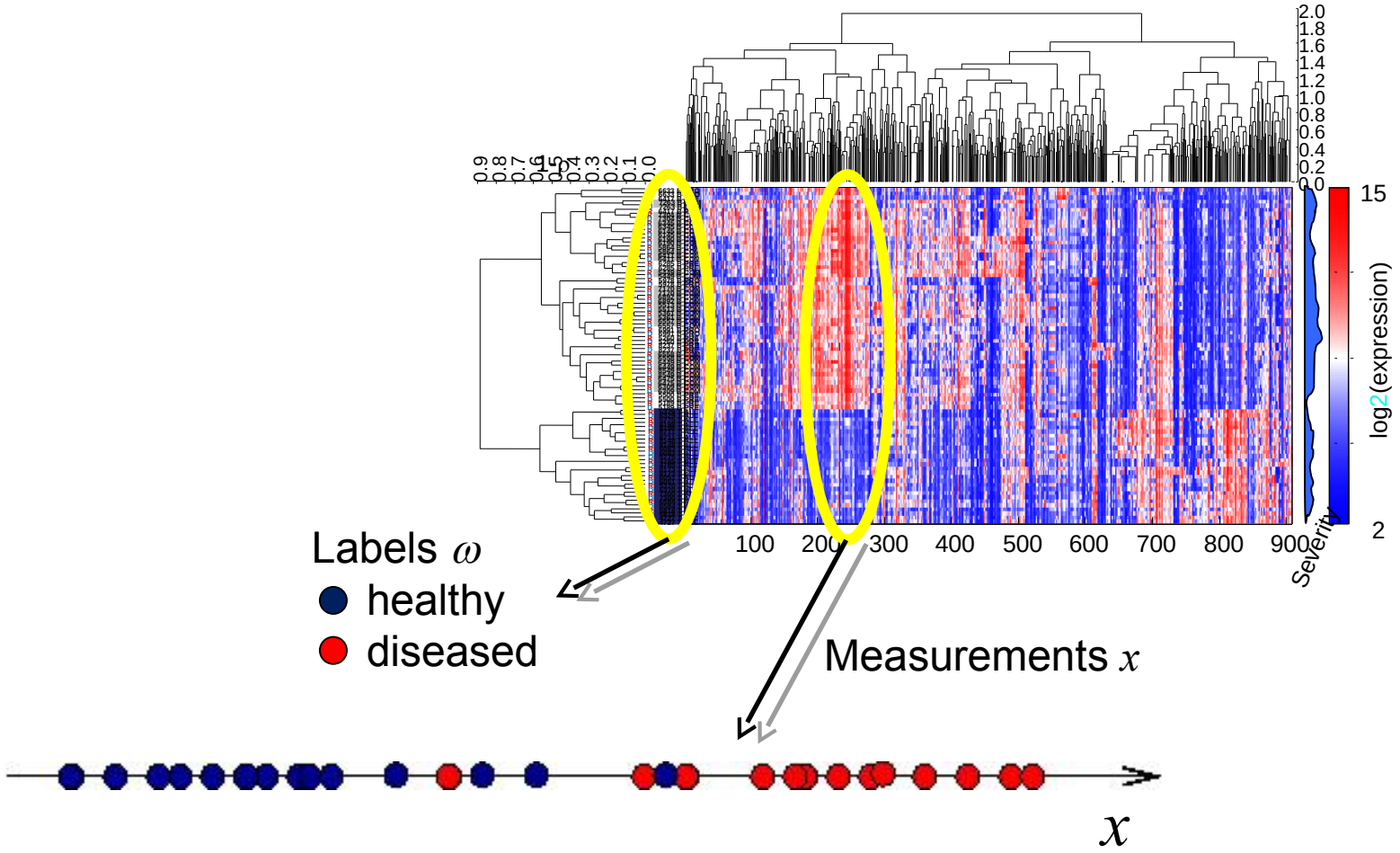
BioSB

Classification (3)

- Given labeled data: x
- Assign to each object a class label ω
- In effect splits the feature space in separate regions



Classification (4)



BioSB

Class posterior probability

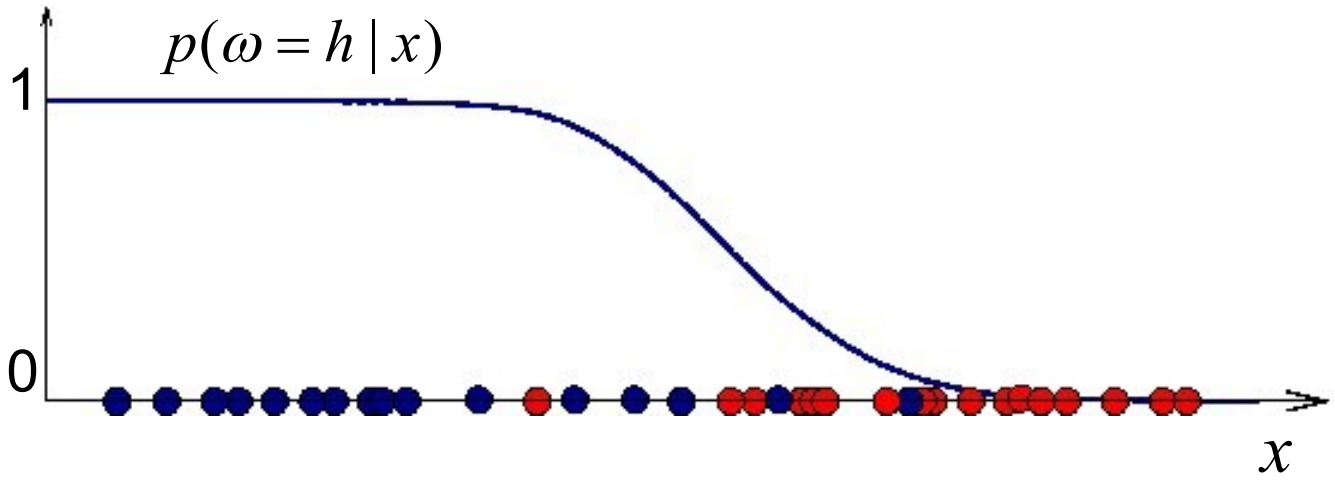
- For each object we have to estimate posterior $p(\omega = c | \mathbf{x})$

Labels ω
- healthy
- diseased

BioSB

Class posterior probability (2)

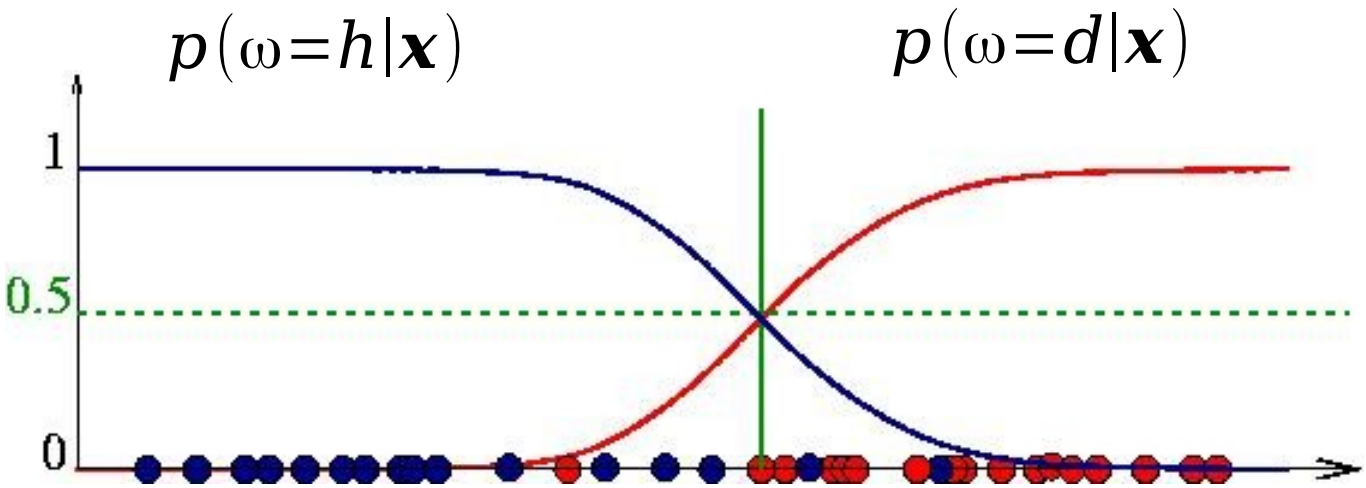
- For each object we have to estimate posterior $p(\omega = c | \mathbf{x})$



Labels ω
- healthy
- diseased

Class posterior probability (3)

- For each object we have to estimate posterior $p(\omega = c|\mathbf{x})$



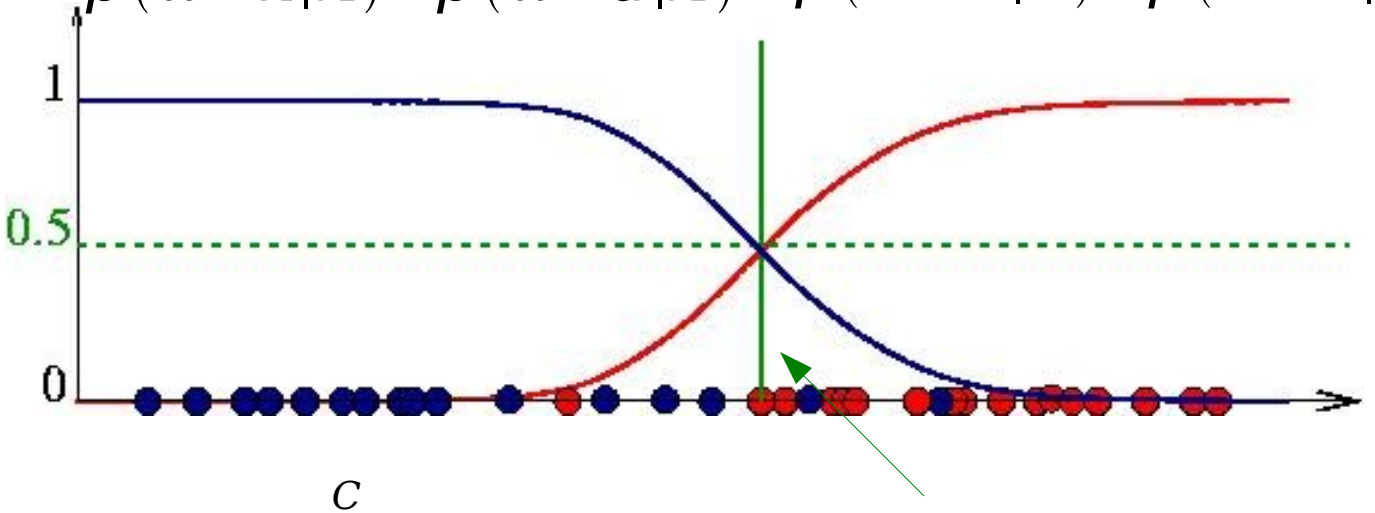
Labels ω
- healthy
- diseased

Of course: $\sum_{c=1}^{C} p(\omega = c|\mathbf{x}) = 1$

Class posterior probability (4)

- For each object we have to estimate posterior $p(\omega = c|\mathbf{x})$

$p(\omega = h|\mathbf{x}) > p(\omega = d|\mathbf{x})$ $p(\omega = d|\mathbf{x}) > p(\omega = h|\mathbf{x})$



Labels ω
- healthy
- diseased

Of course: $\sum_{c=1}^{C} p(\omega = c|\mathbf{x}) = 1$

decision boundary

- Assign label of class with the largest posterior probability

BioSB

Description of a classifier

There are several ways to describe the classifier:

- If $p(\omega=h|\mathbf{x})>p(\omega=d|\mathbf{x})$ then assign to h otherwise to d
- If $p(\omega=h|\mathbf{x})-p(\omega=d|\mathbf{x})>0$ then assign to h
- If $\frac{p(\omega=h|\mathbf{x})}{p(\omega=d|\mathbf{x})}>1$ then assign to h
- If $\ln(p(\omega=h|\mathbf{x}))-\ln(p(\omega=d|\mathbf{x}))>0$ then assign to h

A Bayesian classifier is a *threshold* on the difference between *posterior probabilities*

BioSB

Logistic classifier

- We can rewrite:

$$\ln\left(p(\omega=h|\boldsymbol{x})\right)-\ln\left(p(\omega=d|\boldsymbol{x})\right)=\ln\left(\frac{p(\omega=h|\boldsymbol{x})}{p(\omega=d|\boldsymbol{x})}\right)$$

logit, log-odds

- Assume we can approximate:

$$\ln\left(\frac{p(\omega=h|\boldsymbol{x})}{p(\omega=d|\boldsymbol{x})}\right)=w_0+\boldsymbol{w}^T\boldsymbol{x}$$

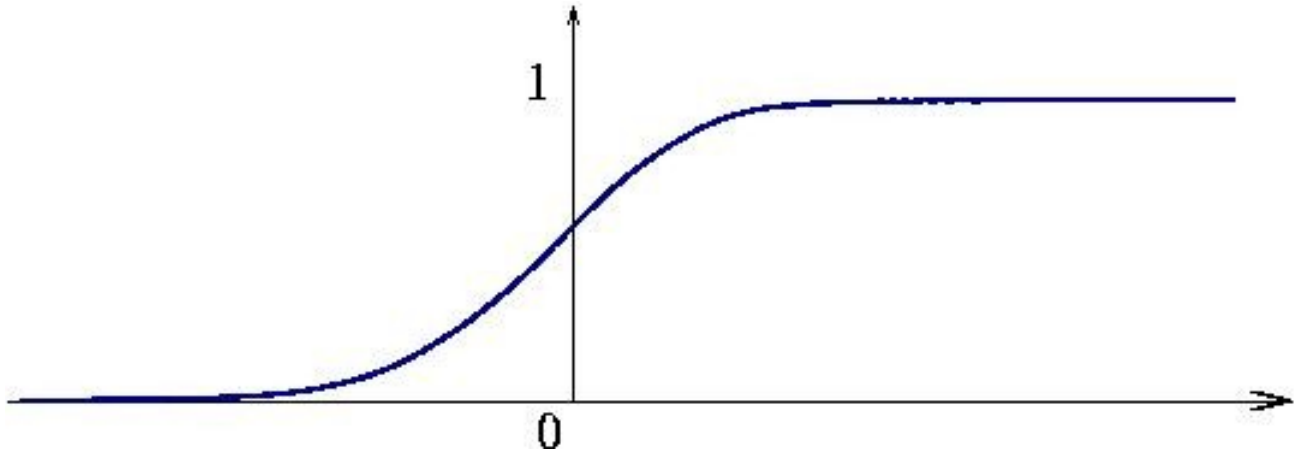
- The classifier becomes (computer lab exercise):

$$p(\omega=d|\boldsymbol{x})=\frac{1}{1+\exp(\boldsymbol{w}^T\boldsymbol{x}+w_0)}$$

BioSB

Logistic function

- The function looks like:



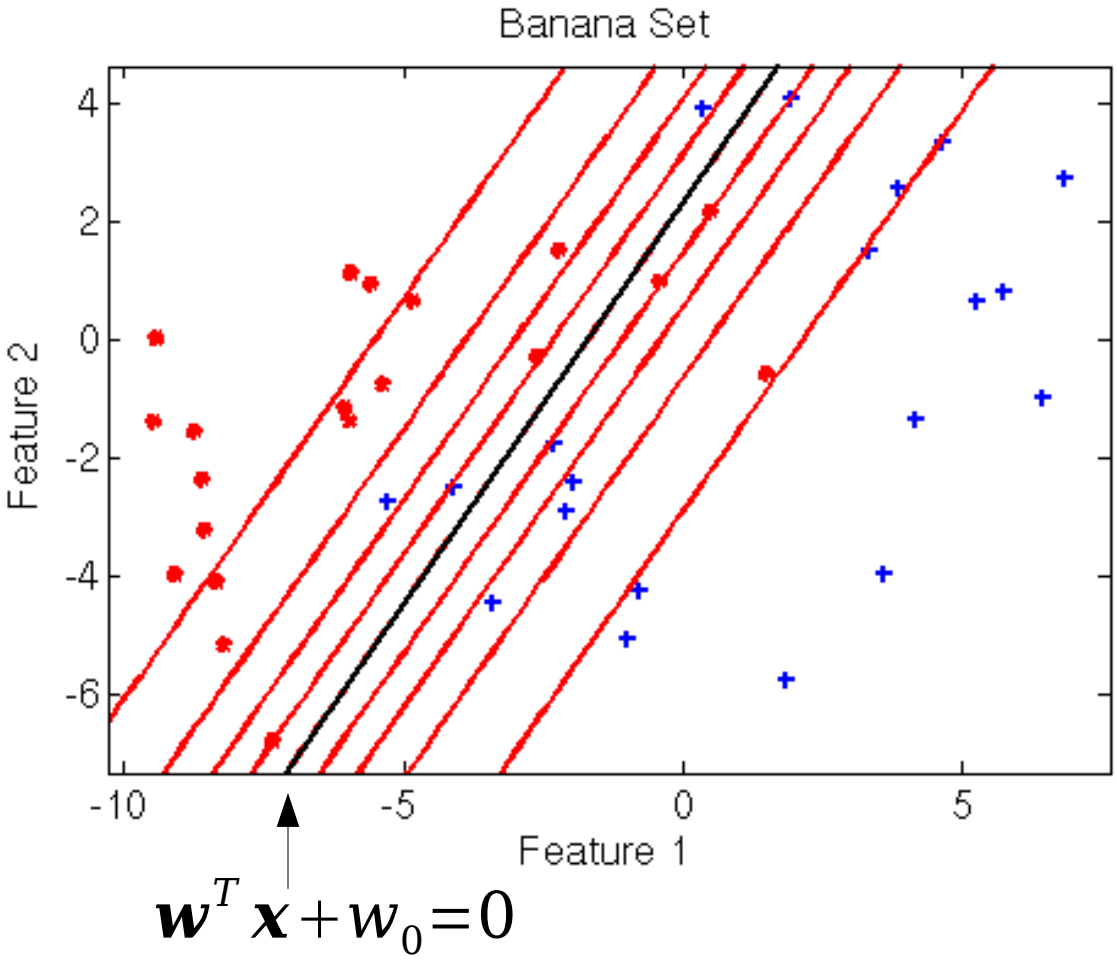
$$f(x)=\frac{1}{1+\exp(-x)}$$

logistic (sigmoid) function

BioSB

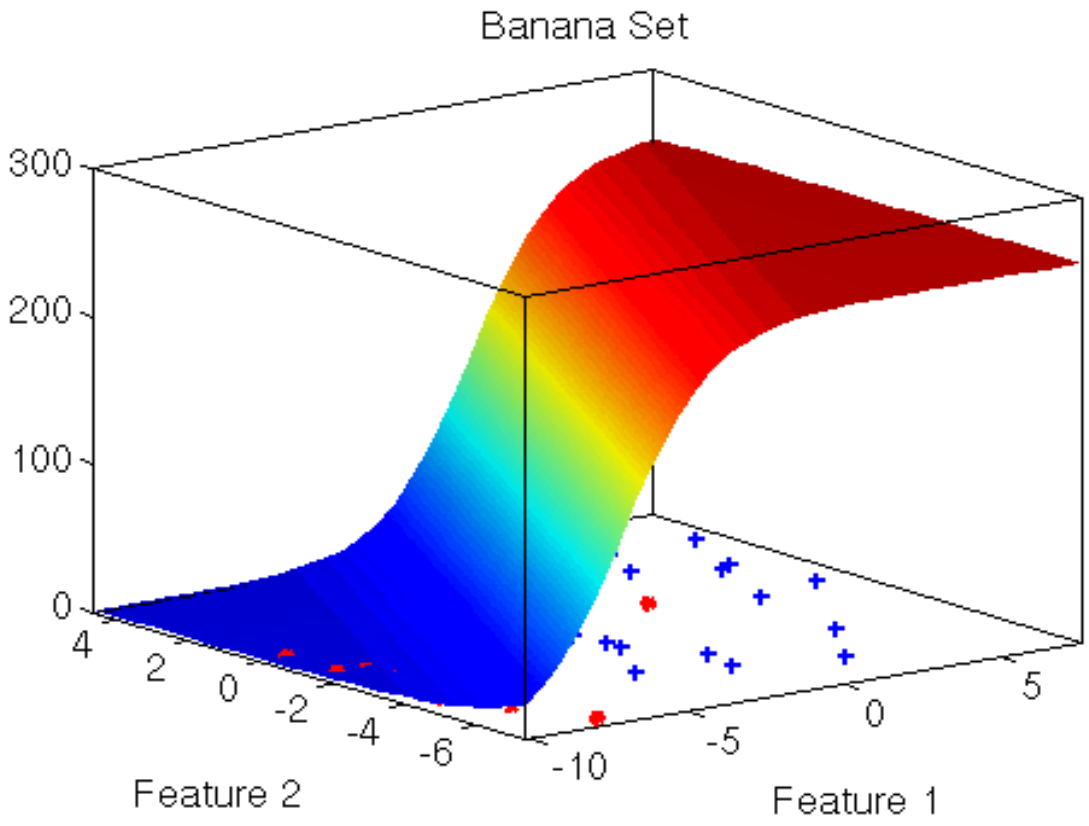
Logistic classifier (2)

- On a two-dimensional dataset it looks like:



Logistic classifier (3)

- On a two-dimensional dataset it looks like:



BioSB

Optimizing the logistic classifier

- To optimize the parameters on a training set, maximize the likelihood

$$L = \prod_{i=1}^{n_1} p(\mathbf{x}_i^{(1)}|\omega_1) \prod_{j=1}^{n_2} p(\mathbf{x}_j^{(2)}|\omega_2)$$

where $\mathbf{x}_i^{(j)}$ is the i-th object from class j

- Maximization using gradient ascent
- Appears to be easier to maximize log(L)
- Weights are iteratively updated as:

$$\mathbf{w}_{new} = \mathbf{w}_{old} + \eta \frac{\partial \log(L)}{\partial \mathbf{w}}$$

BioSB

Optimizing the logistic classifier (2)

- Function to maximize

$$L = \prod_{i=1}^{n_1} p(\boldsymbol{x}_i^{(1)}|\omega_1) \prod_{j=1}^{n_2} p(\boldsymbol{x}_j^{(2)}|\omega_2)$$

- Use log(L)

$$\log(L) = \sum_{i=1}^{n_1} \log(p(\boldsymbol{x}_i^{(1)}|\omega_1)) + \sum_{j=1}^{n_2} \log(p(\boldsymbol{x}_j^{(2)}|\omega_2))$$

- Use Bayes' theorem

$$\log p(\boldsymbol{x}_i^{(1)}|\omega_1) = \log p(\omega_1|\boldsymbol{x}_i^{(1)}) - \log p(\omega_1) + \log p(\boldsymbol{x}_i^{(1)})$$

constant

- Therefore

$$\log(L) = \sum_{i=1}^{n_1} \log(p(\omega_1|\boldsymbol{x}_i^{(1)})) + \sum_{j=1}^{n_2} \log(p(\omega_2|\boldsymbol{x}_j^{(2)})) + C$$

BioSB

Optimizing the logistic classifier (3)

- Filling in that

$$p(\omega_2|\boldsymbol{x}) = \frac{1}{1+\exp(\boldsymbol{w}^T\boldsymbol{x}+w_0)}$$

gives

$$\log(L) = \sum_{i=1}^{n_1}(w_0 + \boldsymbol{w}^T\boldsymbol{x}_i^{(1)}) - \sum_{j=1}^{n_1+n_2}\log(1+\exp(w_0+\boldsymbol{w}^T\boldsymbol{x}_j))$$

BioSB

Derivative of the log-likelihood

- The gradient of log(L) is

$$\frac{\partial \log(L)}{\partial w_0} = n_1 - \sum_{i=1}^{n_1+n_2} p(\omega_1|\boldsymbol{x}_i)$$

$$\frac{\partial \log(L)}{\partial w_j} = \sum_{i=1}^{n_1} (\boldsymbol{x}_i^{(1)})_j - \sum_{i=1}^{n_1+n_2} p(\omega_1|\boldsymbol{x}_i)(\boldsymbol{x}_i)_j, \; j=1,\ldots,p$$

- Take initial values: $w_0 = 0, \boldsymbol{w} = \boldsymbol{0}$
- Keep iterating $\boldsymbol{w}_{new} = \boldsymbol{w}_{old} + \eta \frac{\partial \log(L)}{\partial \boldsymbol{w}}$

till convergence

BioSB

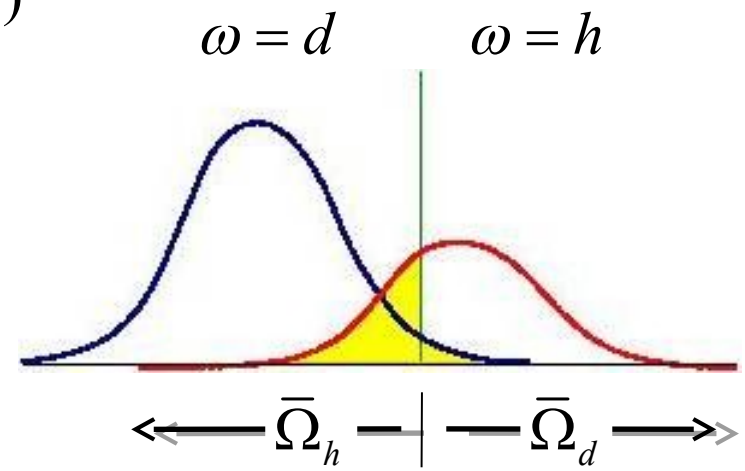
Bayes' error

- The error we make can be described as

$$p(error) = \sum_{c=1}^{C} p(error \mid \omega = c)\, p(\omega = c)$$

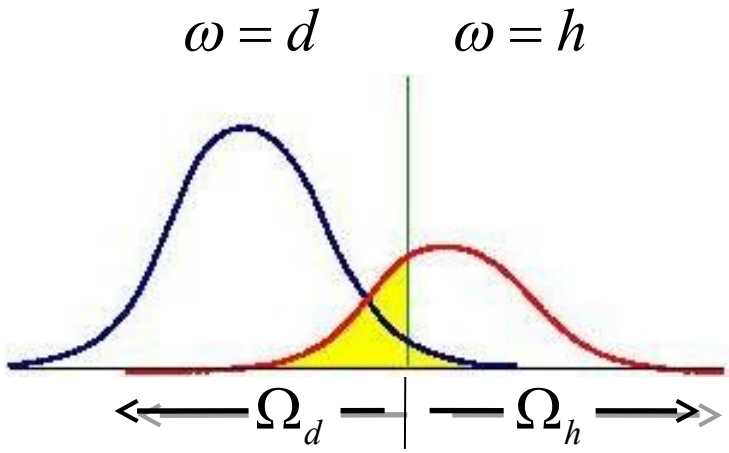
- For a single class:

$$p(error \mid \omega = c) = \int_{\bar{\Omega}_c} p(x \mid \omega = c)\, dx$$



where $\bar{\Omega}_c$ is the complement of the region Ω_c in which objects are assigned to class c

Bayes' error (2)



- Minimizing *p(error)* is equivalent to *maximizing*

$$\sum_{c=1}^{C} \int_{\Omega_c} p(x|\omega = c)\, p(\omega = c)\, dx$$

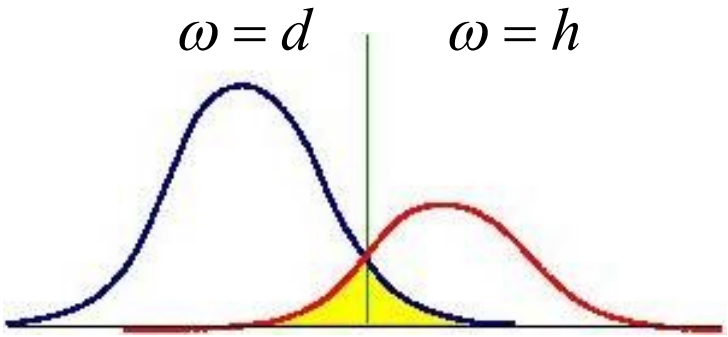
i.e. the probability of correct classification

- At every x, pick class label ω s.t. the above integral is maximal:

$$c_{opt} = \arg\max_c \; p(x \mid \omega = c)\, p(\omega = c)$$

- Bayes' error:

$$e = 1 - \int \max_c \; p(x \mid \omega = c)\, p(\omega = c)\, dx$$



BioSB

Misclassification error

- What is…
  - the maximum error for a problem with C classes?
  - the error of a rather dumb classifier, labeling all data to class c ?
  - the error of this classifier for a 10-class problem, with equal class priors?

Misclassification error

- What is…
  - the maximum error for a problem with C classes?
  - the error of a rather dumb classifier, labeling all data to class c ?
  - the error of this classifier for a 10-class problem, with equal class priors?

Bayes' risk

- Conditional risk of assigning object x to class c' :

$$r(\omega = c' \mid x) = \sum_{c=1}^{C} \Lambda(\omega = c', \omega = c)\, p(\omega = c \mid x)$$

- Average risk over class c' :

$$r(\omega = c') = \int_{\Omega_{c'}} r(\omega = c' \mid x)\, p(x)\, dx$$

$$= \int_{\Omega_{c'}} \sum_{c=1}^{C} \Lambda(\omega = c', \omega = c)\, p(\omega = c \mid x) p(x)\, dx$$

- Overall expected risk (at every x):

$$R = \sum_{c'=1}^{C} r(\omega = c') = \sum_{c'=1}^{C} \int_{\Omega_{c'}} \sum_{c=1}^{C} \Lambda(\omega = c', \omega = c)\, p(\omega = c \mid x) p(x)\, dx$$

BioSB

Bayes' risk (2)

- Overall expected risk is minimized if class label c' is chosen s.t.

$$c_{opt} = \arg\min_{c'} \sum_{c=1}^{C} \Lambda(\omega = c', \omega = c)\, p(\omega = c \mid x) p(x)$$

- For equal cost $\Lambda(.,.)$ this is identical to Bayes' rule for minimum error
- The minimum overall risk then is:

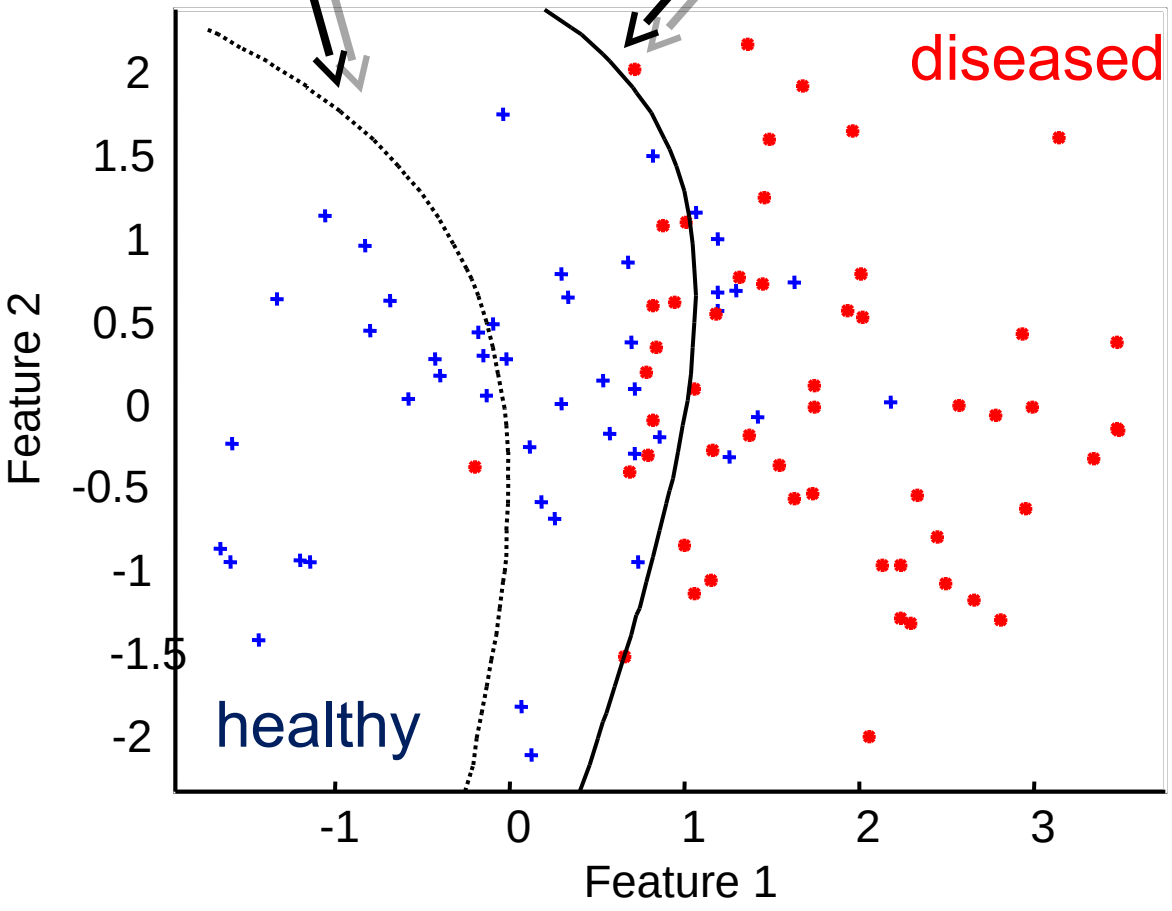
$$r^{*} = \int \min_{c'} \sum_{c=1}^{C} \Lambda(\omega = c', \omega = c)\, p(\omega = c \mid x) p(x) dx$$

BioSB

Example

$$\Lambda = \begin{pmatrix} 0 & 10 \\ 1 & 0 \end{pmatrix} \quad \begin{matrix} h & d \end{matrix}$$

$$\Lambda = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \quad \begin{matrix} h & d \end{matrix}$$



BioSB

Reject option

- Reject classification of objects with insufficient certainty (too low confidence in any class assignment)
- The reject area R can be written as:

$$R = \{x \mid \max_c \, p(\omega = c \mid x) < t\}$$

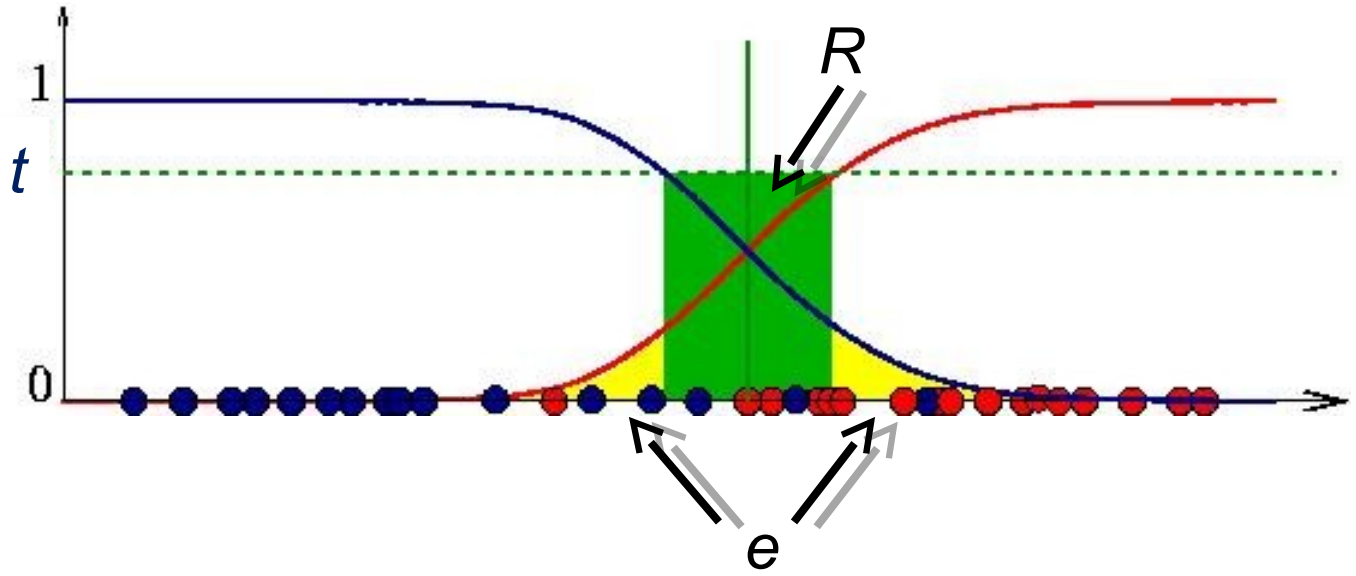
- Rejected objects should be classified by an expert, or by another classifier

- In Bayesian estimation, the reject option can be modeled as an additional class with certain (high) misclassification cost

BioSB

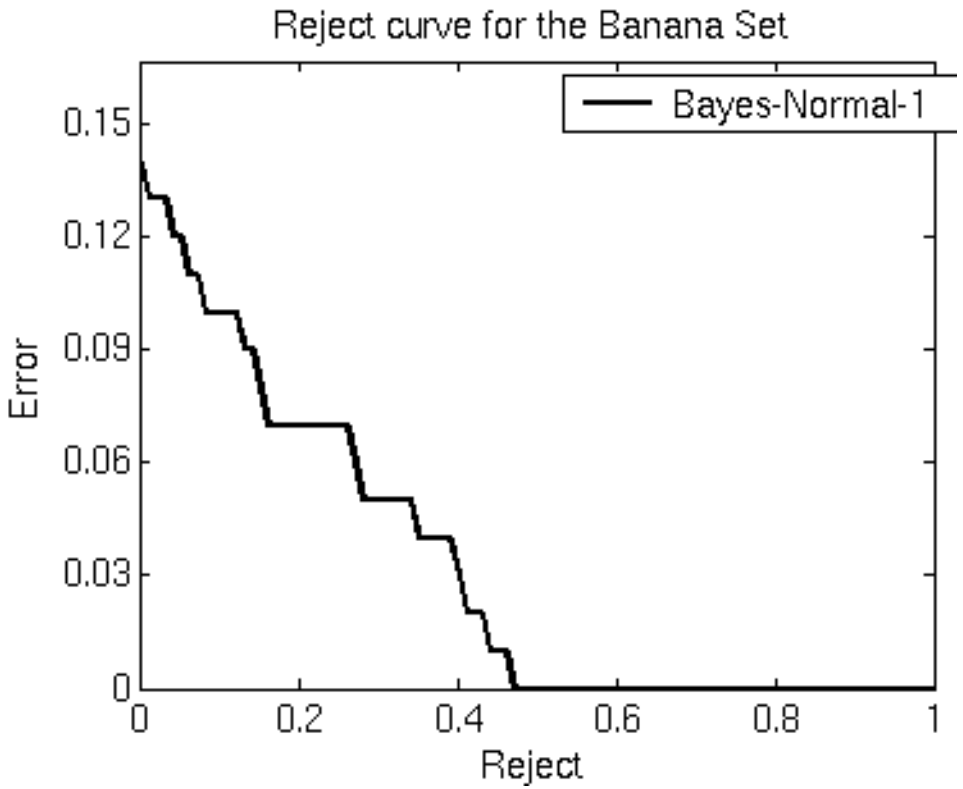
Reject option (2)

- Define the threshold t
- Reject all objects in the green area R
- Objects in the yellow area are still errors



Error-reject curve

- By changing the threshold t,
  the error decreases, but the percentage rejected increases



Recapitulation

- For classification we want the posterior $p(\omega|\boldsymbol{x})$
- We can approximate the posterior directly: logistic classifier
- Assigning an object to the class with maximum posterior probability gives the Bayes classifier
- Bayes classifier is the optimal classifier
- The Bayes' error is the smallest error attainable
- The Bayes' risk is the smallest risk attainable

BioSB

Plug-in Bayes classification

- In many cases the posterior is hard to estimate
- Often a functional form of the class distributions can be assumed
- Use Bayes' theorem to rewrite one into the other:

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

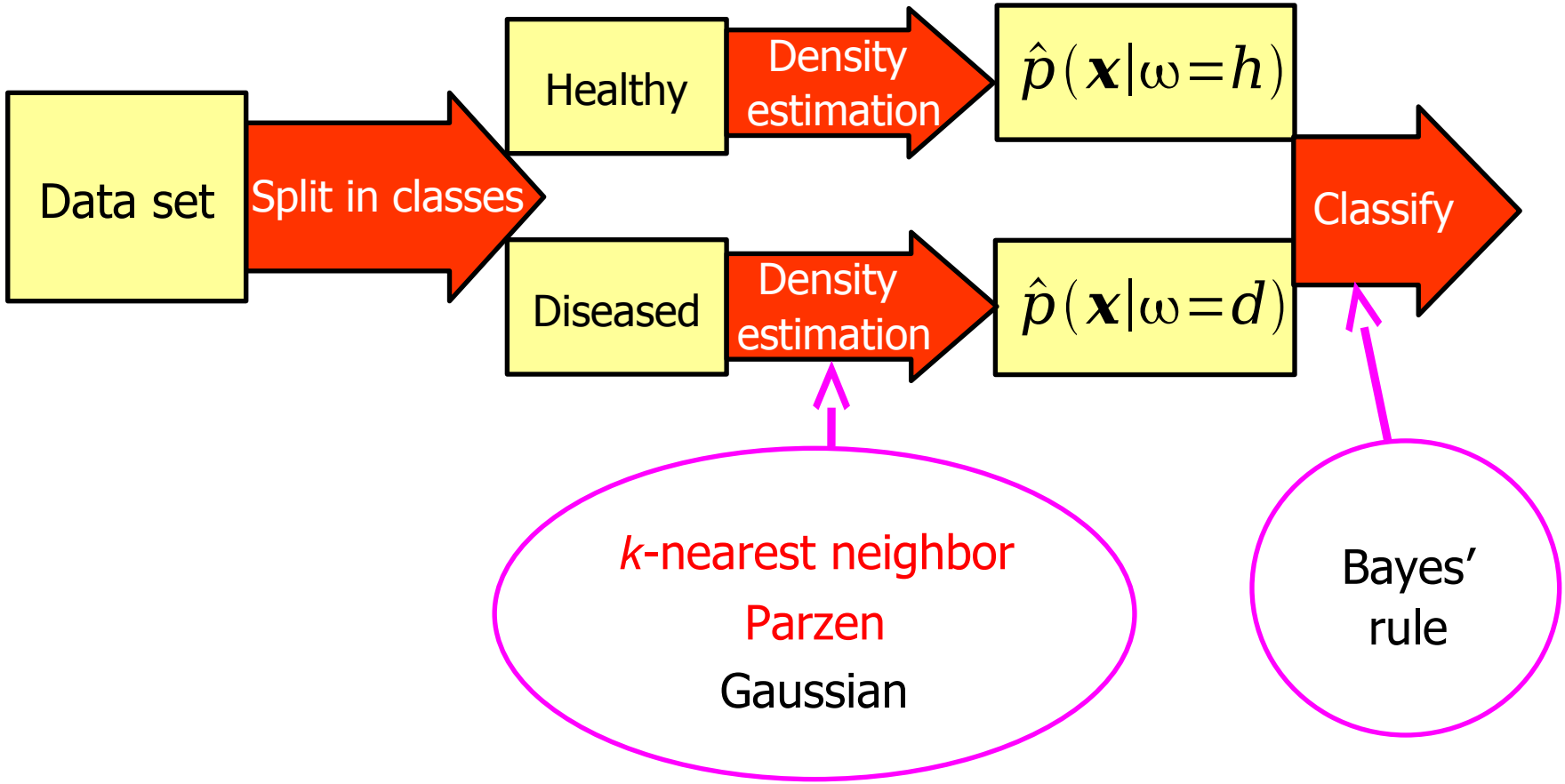
class-conditional distribution: $p(\mathbf{x}|\omega)$

prior distribution: $p(\omega)$

data distribution: $p(\mathbf{x})$

BioSB

Plug-in Bayes classification (2)

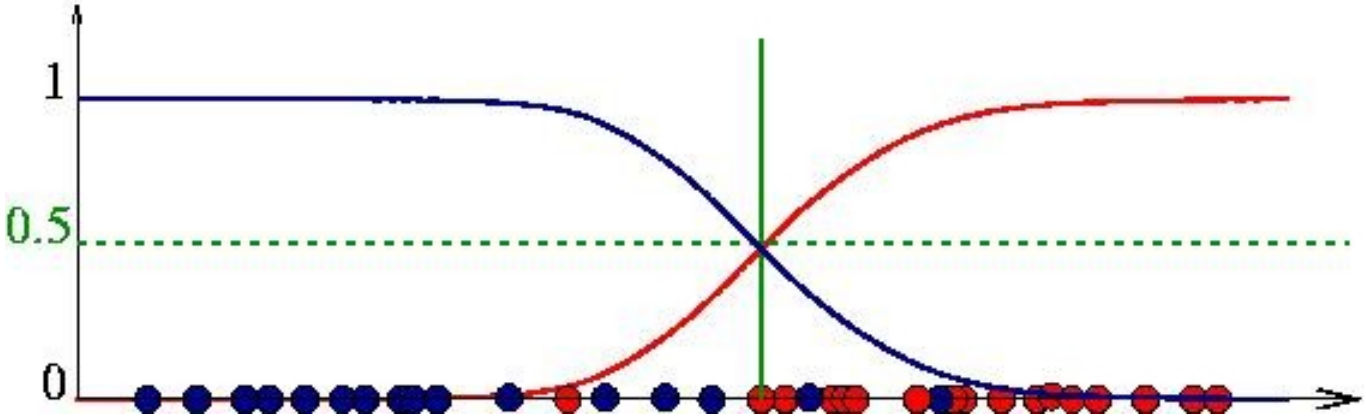


BioSB

Plug-in Bayes classification (3)

- For each object we estimate $p(\omega = c|\mathbf{x})$ using Bayes' rule

$$p(\mathbf{x}|\omega = h)\,p(\omega = h) > p(\mathbf{x}|\omega = d)\,p(\omega = d)$$



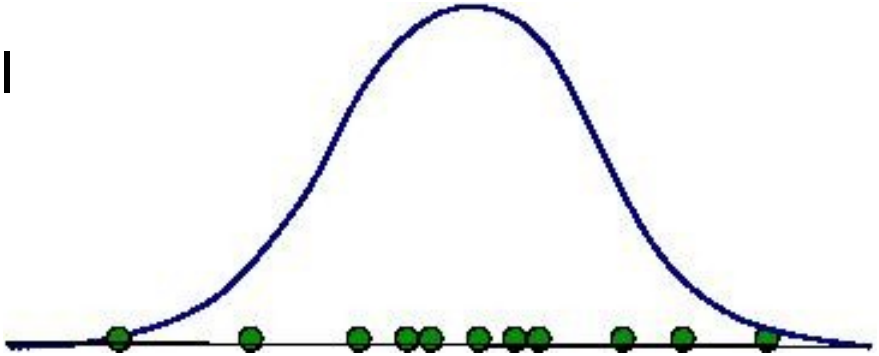
$$p(\mathbf{x}|\omega = d)\,p(\omega = d) > p(\mathbf{x}|\omega = h)\,p(\omega = h)$$

BioSB

Bayes decision making

$$\hat{p}(\boldsymbol{x}|\omega_i) = N(\boldsymbol{x}; \mu, \sigma)$$

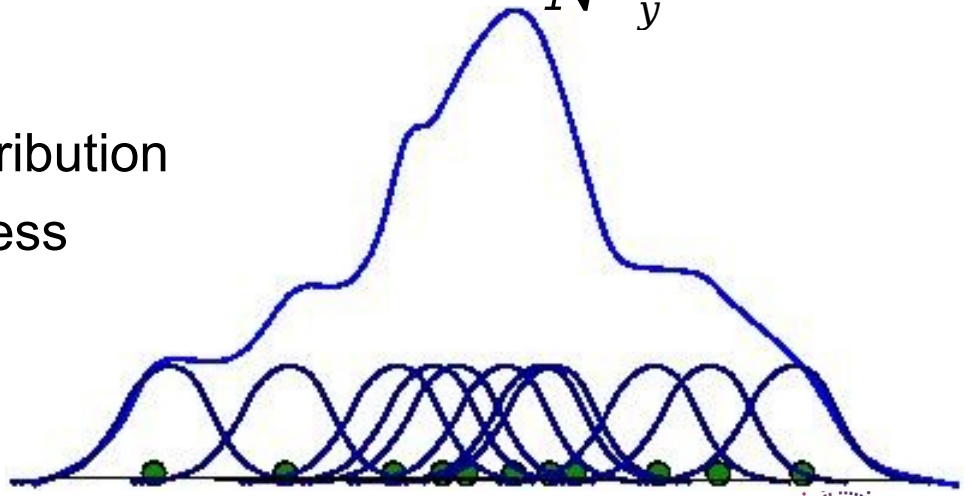
- Estimate the class-conditional density (Day 1) $\hat{p}(\boldsymbol{x}|\omega_i)$



- Parametric
  - Known distribution
  - Estimate parameters on training set

$$\hat{p}(\boldsymbol{x}|\omega_i) = \frac{1}{N}\sum_{y} K(\boldsymbol{x}, \boldsymbol{y})$$

- Non-parametric
  - No knowledge on distribution
  - Manage the smoothness of the distribution

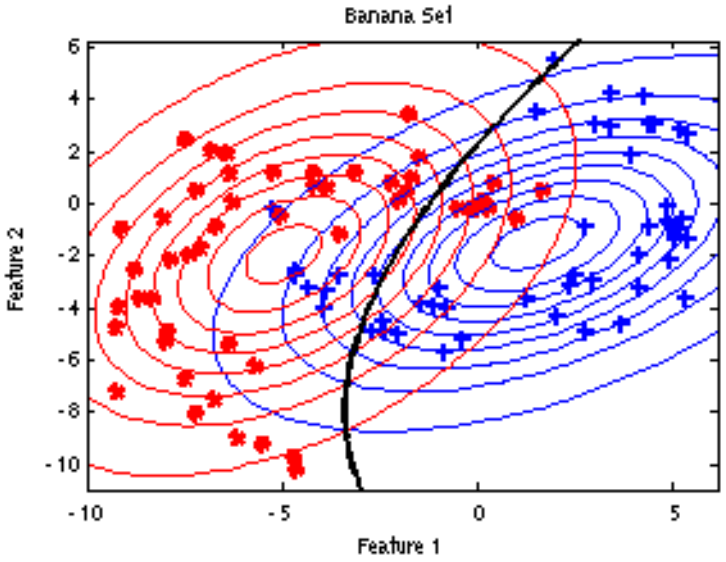


BioSB

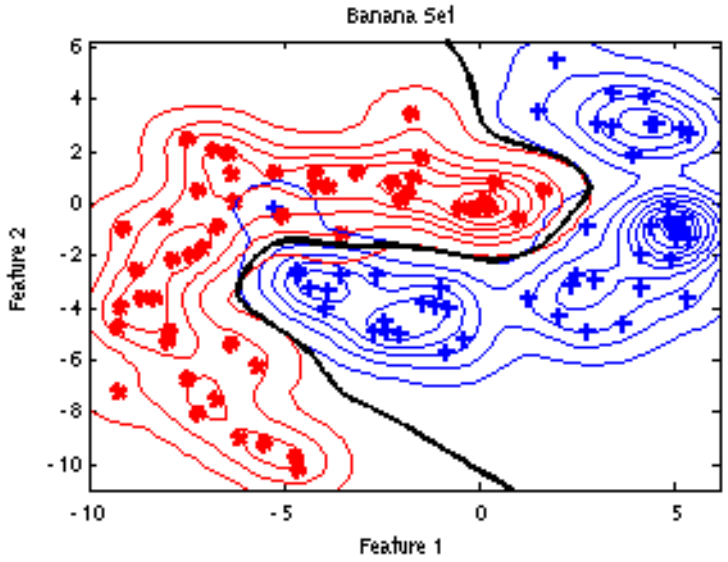
Example plugin

- Two examples

Normal density estimation



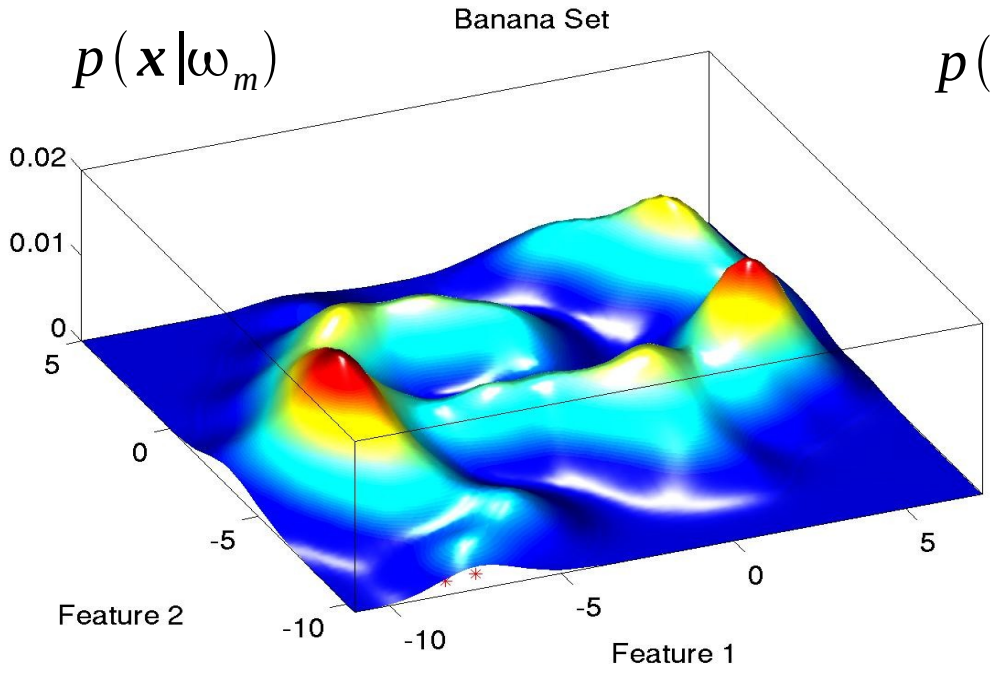
Parzen density estimation

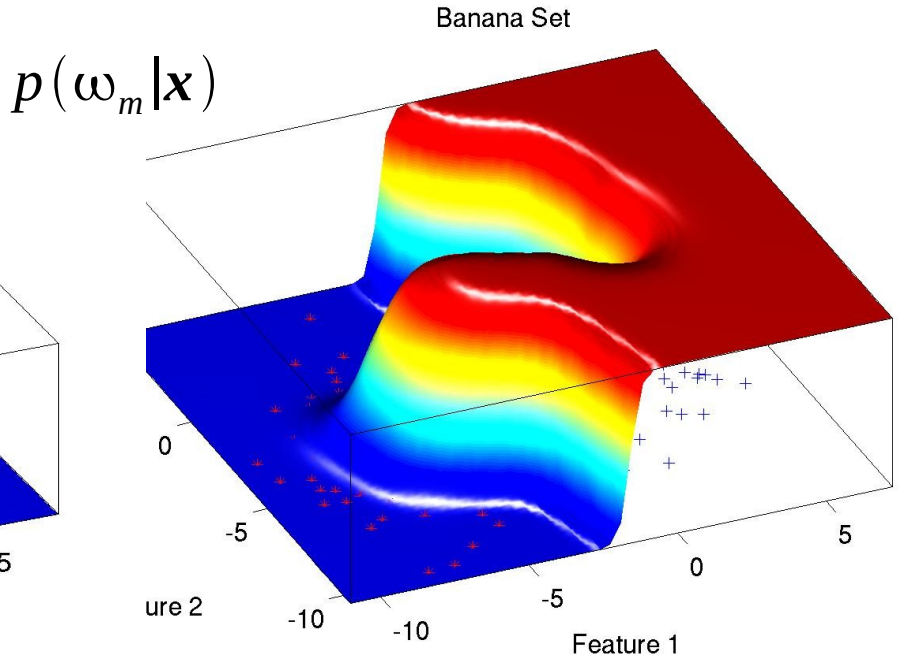


BioSB

Parzen classifier

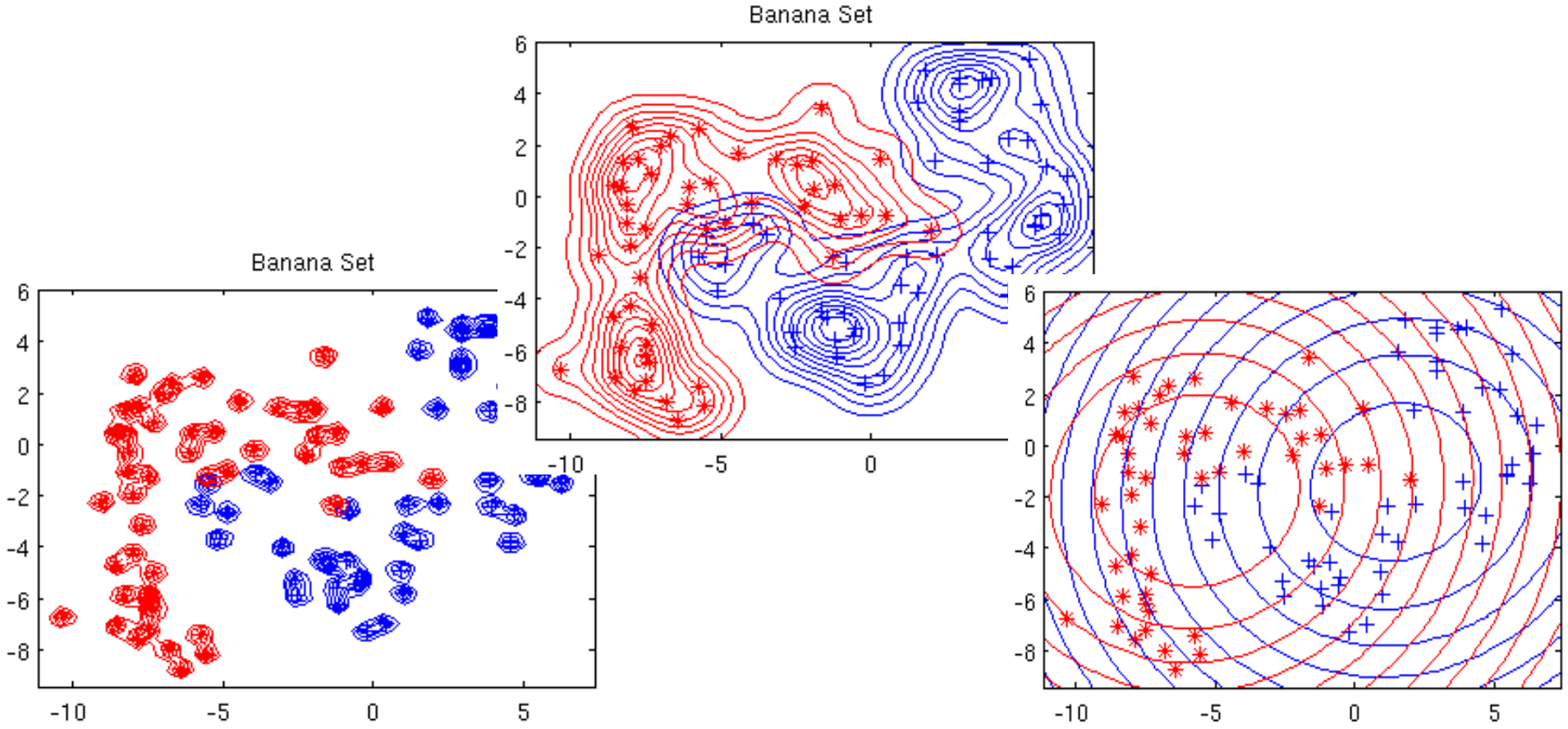
$$p(\boldsymbol{x}|\omega_m) = \frac{1}{N}\sum_{i=1}^{N_m} N(\boldsymbol{x}; \boldsymbol{x}_i, h\boldsymbol{I})$$





Parzen width parameter

- The width parameter *h* has a large influence



Optimization of *h*

- Use the average *k*-nearest neighbor distance (*k*=10 is suggested...)
- Use a heuristic $$h = \sigma \left(\frac{4}{p+2} \right)^{\frac{1}{p+4}} n^{\frac{-1}{p+4}}$$

$$\sigma^2 = \frac{1}{p} \sum_{i=1}^{p} s_{ii}$$

- Optimize the likelihood using cross-validation

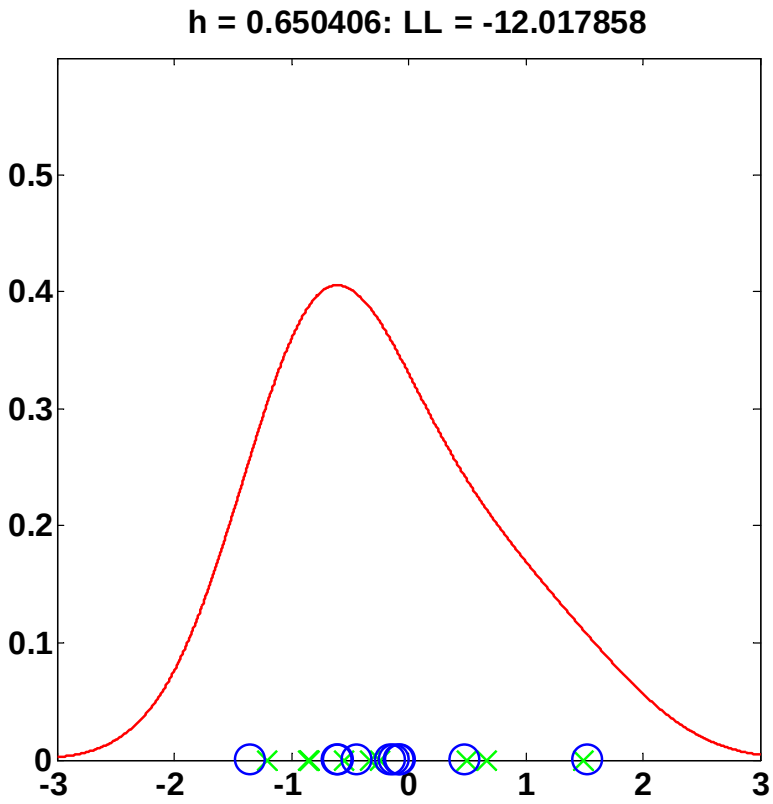
$$\prod_{i=1}^{n} \hat{p}(\boldsymbol{x}_i)$$

- and more...

BioSB

Cross-validation

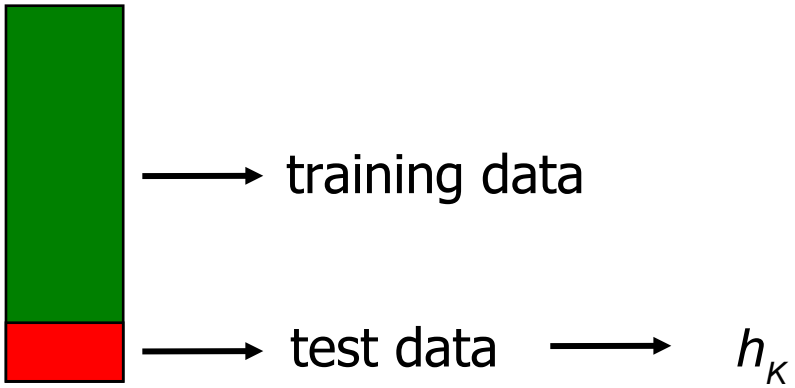
- Scheme:
  - Split data into *training set* and *test* set
  - Optimise h w.r.t. likelihood of test set, given Parzen model trained on training set
- Problems:
  - Uses a lot of valuable data
  - Sensitive to split of data



BioSB

Cross-validation (2)

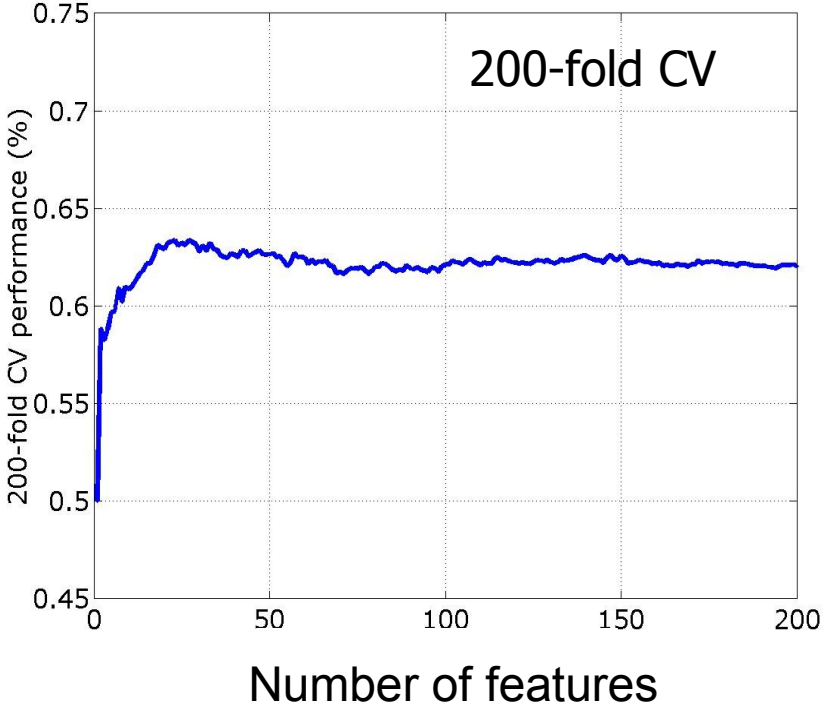
- Better solution: *K*-fold crossvalidation
  - Split data into K parts ($K = n$: leave-one-out)
  - Repeat K times:
    - Find h using $(K - 1)$ parts for training and 1 part for testing
  - Use average of h's as kernel width

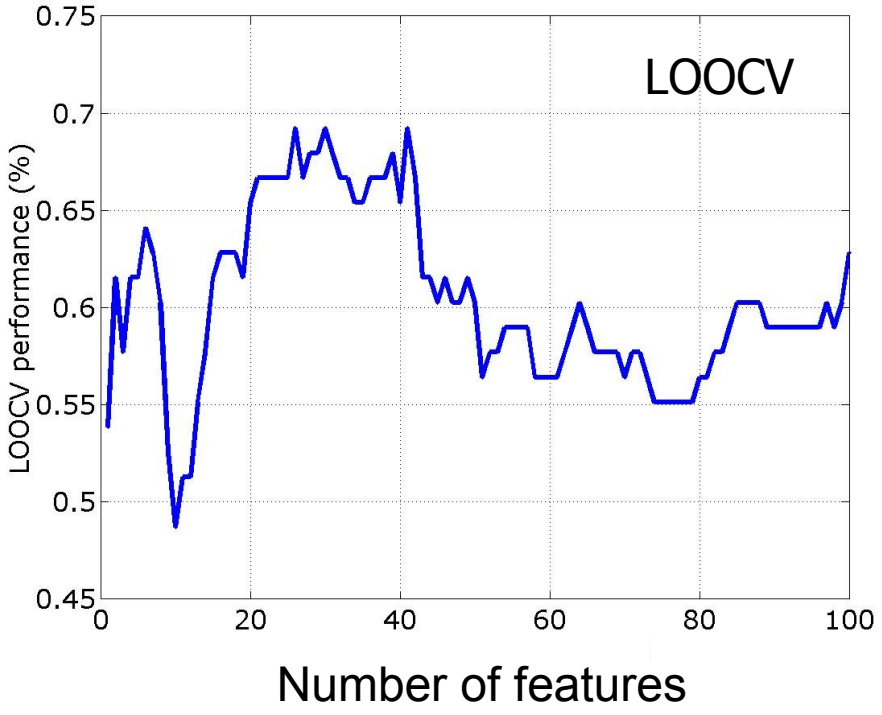


BioSB

Cross-validation (3)

- Prefer *K*-fold cross-validation over leave-one-out
  - Smoother (less variance) and more biased (conservative)





Bootstrap

- Alternative to cross-validation:
  - Repeat K times:
    - Draw n objects from the dataset, with replacement (some objects will be selected more than once)
    - Test using objects that were not selected
- Cross-validation and bootstrap estimates are *biased*
  - They are conservative (i.e. too pessimistic) because they do not use all data available

BioSB

Training, validation and test sets

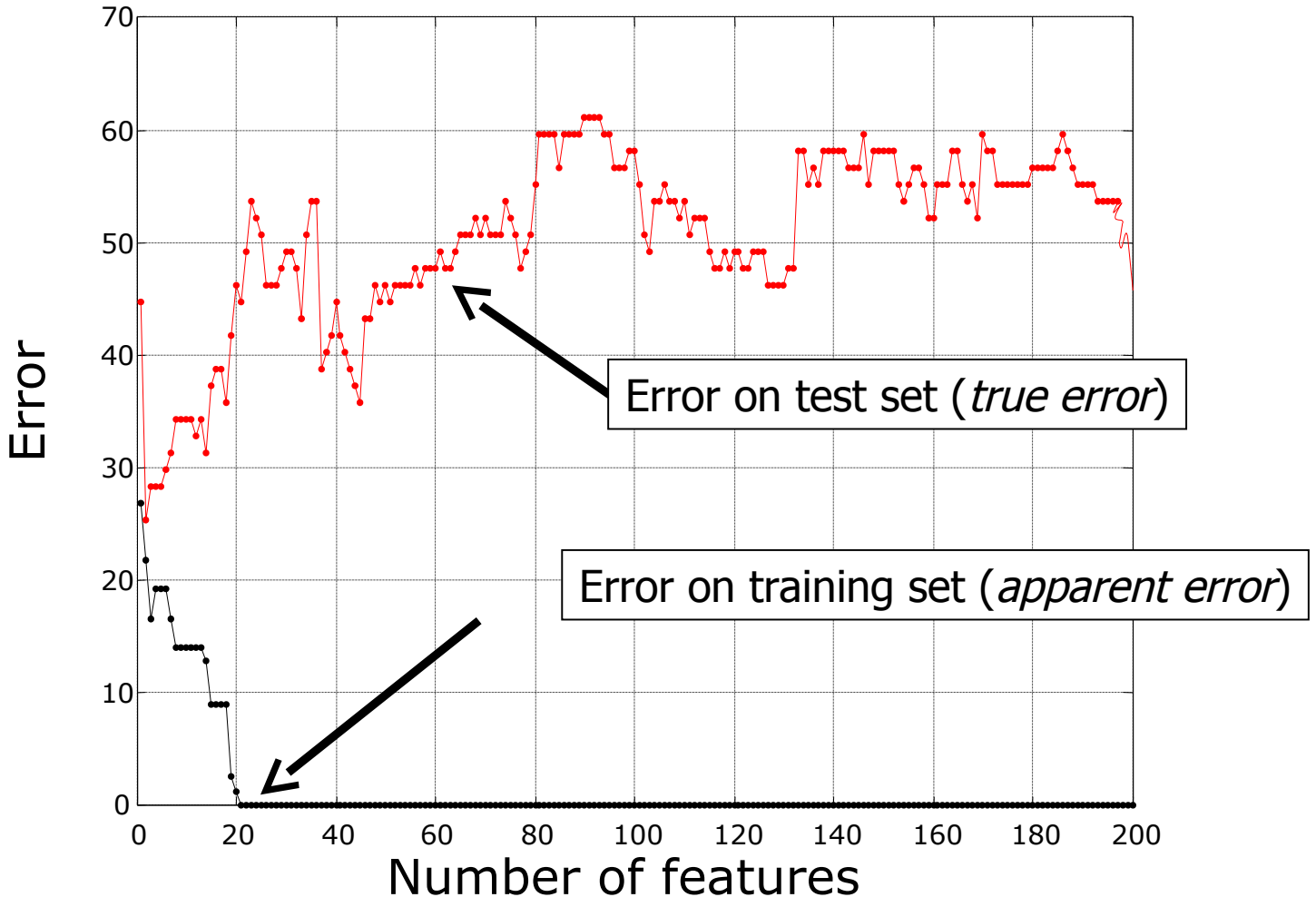
- Terminology:
  - A *training set* is used to estimate parameters
  - An optional *validation set* is used to optimise parameter settings, e.g. by calculating classifier error on this set
  - A *test set* is only used to judge performance of the entire classifier

- Error estimates:
  - On training set: *apparent error*
  - On test set: *true error*

- The test set should *never* be used to set any parameters! This leads to biased estimates of performance -- in practice we may do much worse than we predict

BioSB

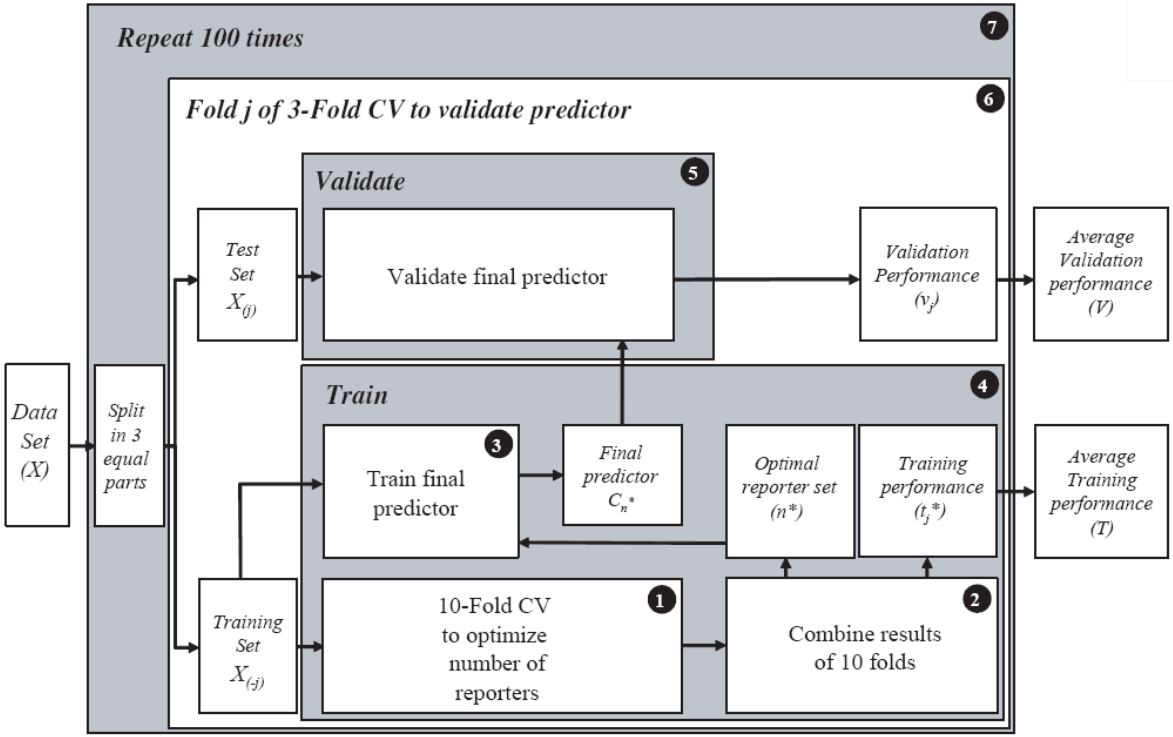
Training, validation and test sets (2)



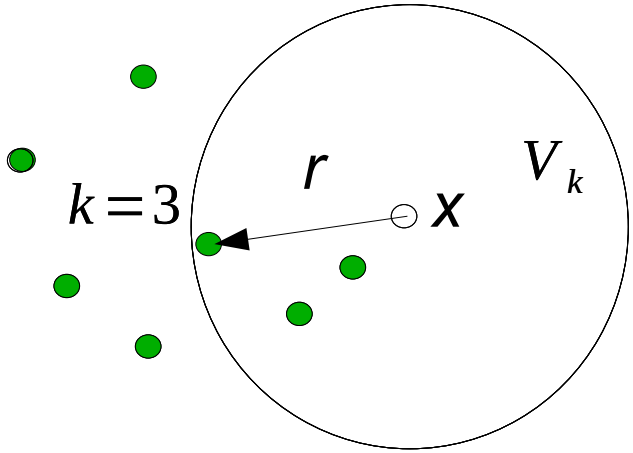
BioSB

Training, validation and test sets (3)

- Can lead to complicated schemes for estimating parameters, e.g. double cross-validation loops



Nearest neighbor classification



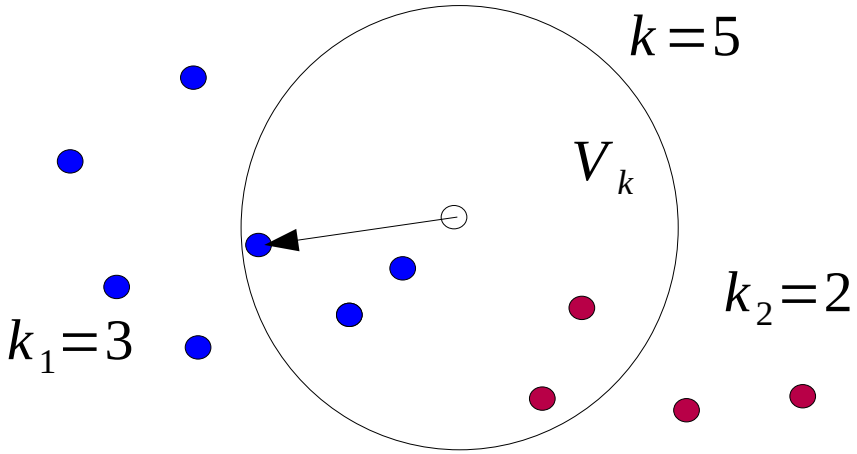
- For the k-nearest neighbor density we defined:

$$\hat{p}(\boldsymbol{x}) = \frac{k}{n V_k}$$

where V_k is the volume of the sphere centered at x, with radius r the distance to the k-th nearest neighbor

BioSB

Nearest neighbor classification (2)



- When more classes are present, count how many objects of each of the classes are members of the *k* neighbors
- Class-conditional density:

$$\hat{p}(\boldsymbol{x}|\omega_m)=\frac{k_m}{n_m V_k}$$

BioSB

Nearest neighbor classification (3)

- Using Bayes: $\hat{p}(\boldsymbol{x}|\omega_m)\,\hat{p}(\omega_m) \geq \hat{p}(\boldsymbol{x}|\omega_i)\,\hat{p}(\omega_i)$
- Estimate the prior probability by counting:

$$\hat{p}(\omega_m) = \frac{n_m}{n}$$

- Fill in:

$$\frac{k_m}{n_m V_k}\frac{n_m}{n} \geq \frac{k_i}{n_i V_k}\frac{n_i}{n} \quad \Rightarrow \quad k_m \geq k_i$$

- No density estimation is needed!

BioSB

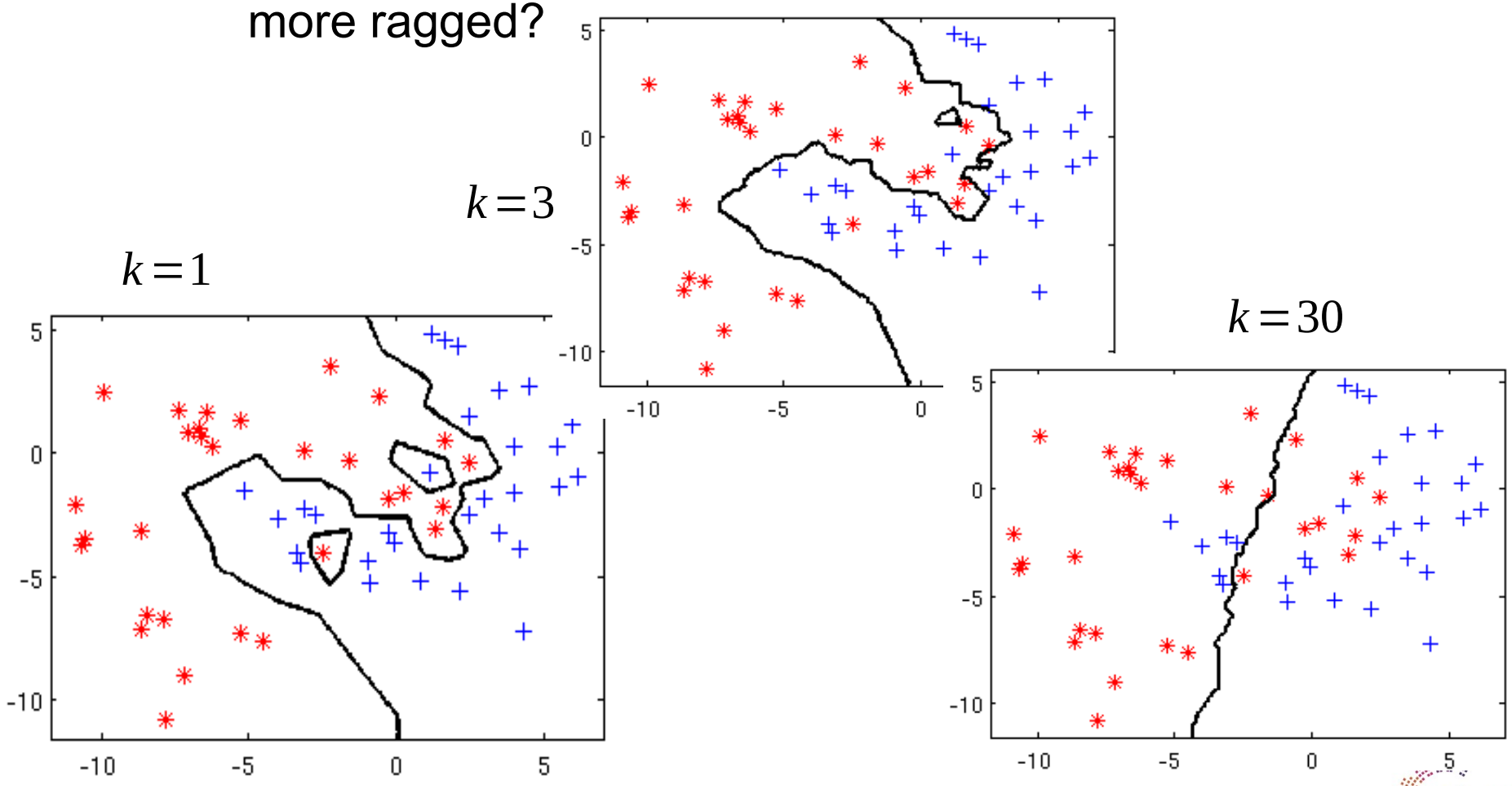
The choice of *k*

- When does the classifier become more smooth? When more ragged?

- What happens for $k = 1$, and $k = n$?

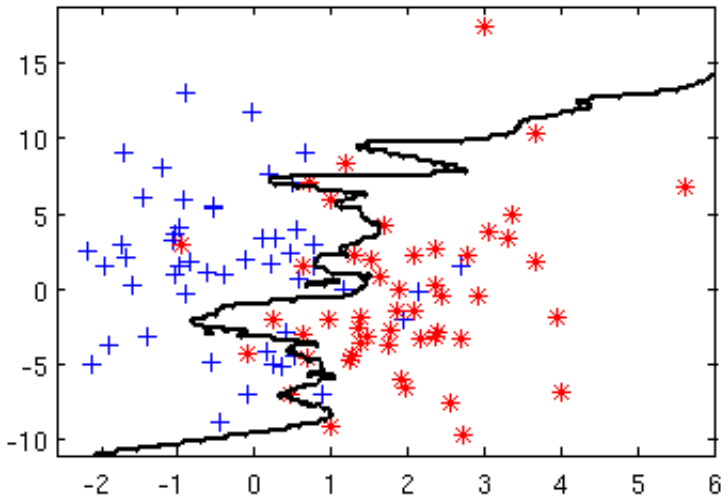
The choice of *k* (2)

- When does the classifier become more smooth? When more ragged?



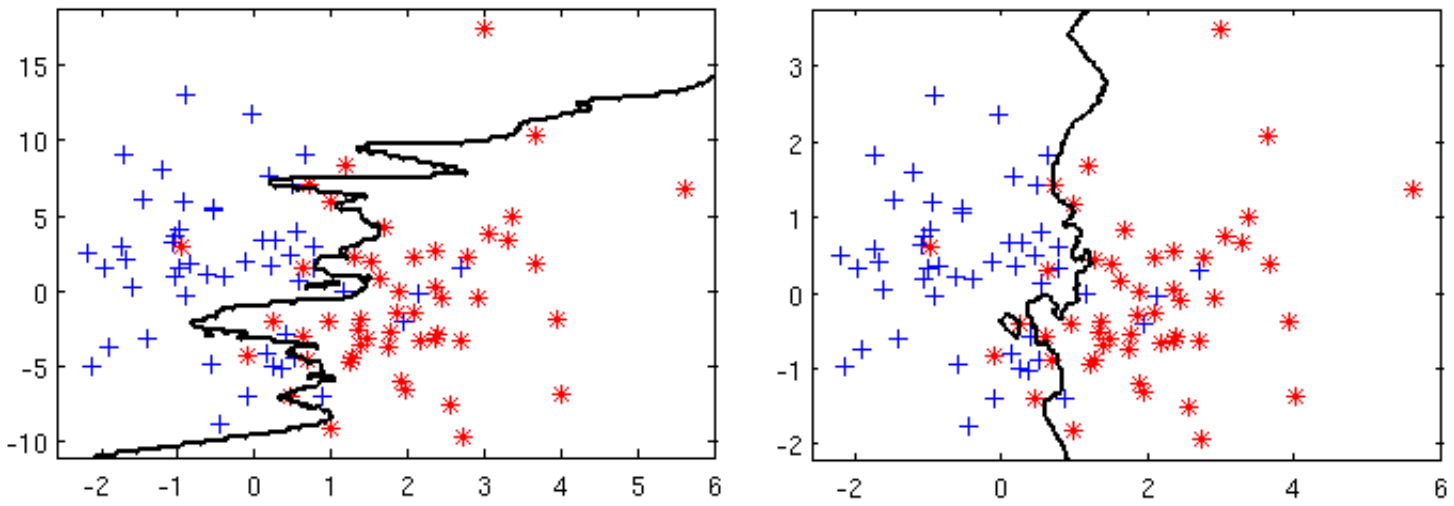
Sometimes strange results:

$k = 5$



Sometimes strange results (2):

$k = 5$



Rescaling the features has large influence!

BioSB

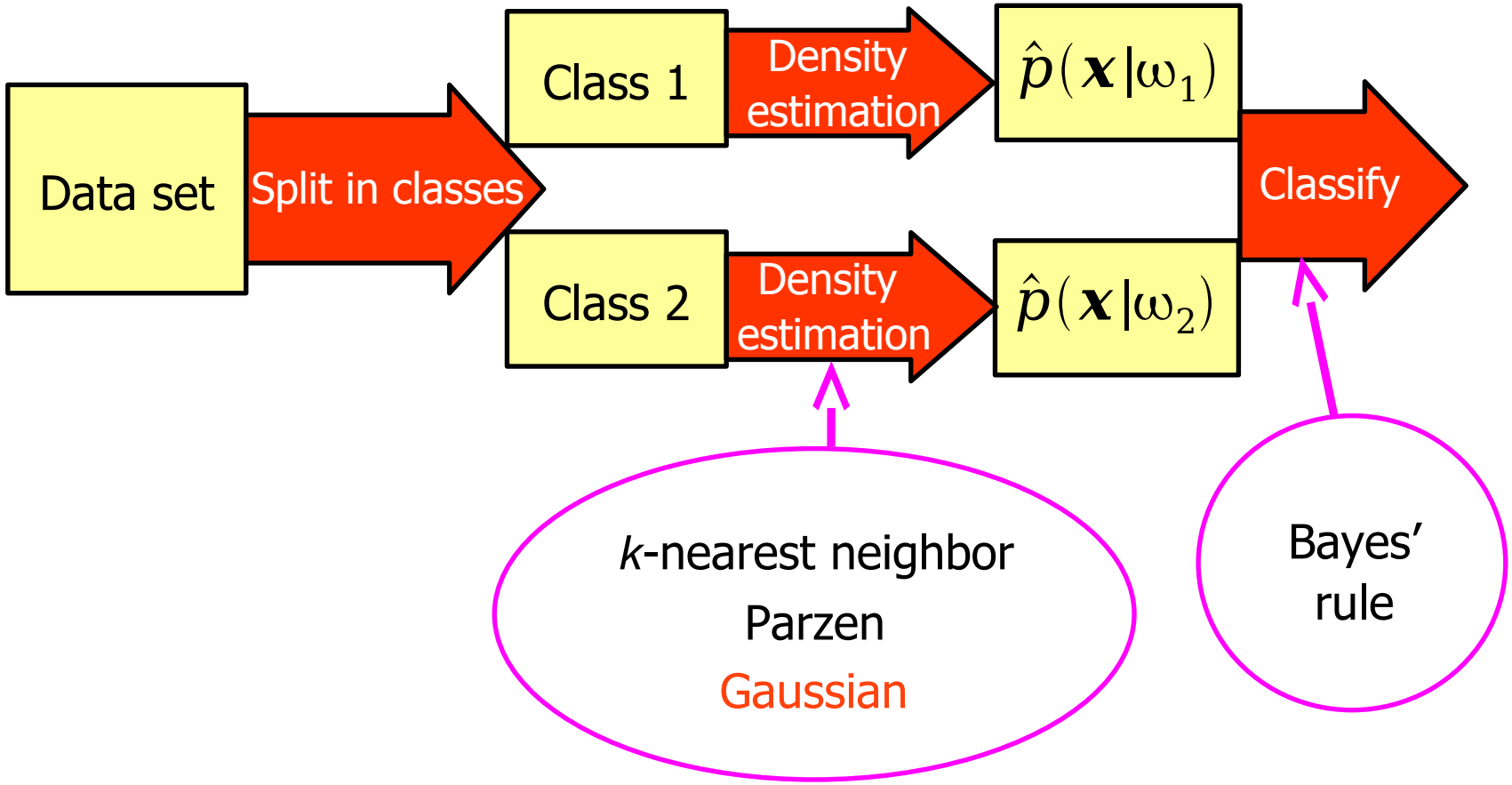
Advantages/disadvantages

- simple and flexible classifier
- often a very good classification performance
- it is simple to adapt the complexity of the classifier

- you have to store the complete training set
- distances to all training objects have to be computed
- scaling of the features should be sensible
- you have to optimize *k* or *h*

BioSB

Classifying with densities



BioSB

Plug-in Gaussian distribution

- Now take the most obvious choice: the Gaussian distribution

$$\hat{p}(\mathbf{x}|\omega)=\frac{1}{\sqrt{2\pi^{p}det(\hat{\Sigma}_{\omega})}}\exp\left(-\frac{1}{2}(\mathbf{x}-\hat{\mu_{\omega}})^{T}\hat{\Sigma}_{\omega}^{-1}(\mathbf{x}-\hat{\mu_{\omega}})\right)$$

- So-called parametric density estimation
- We have to estimate the parameters via maximum likelihood:

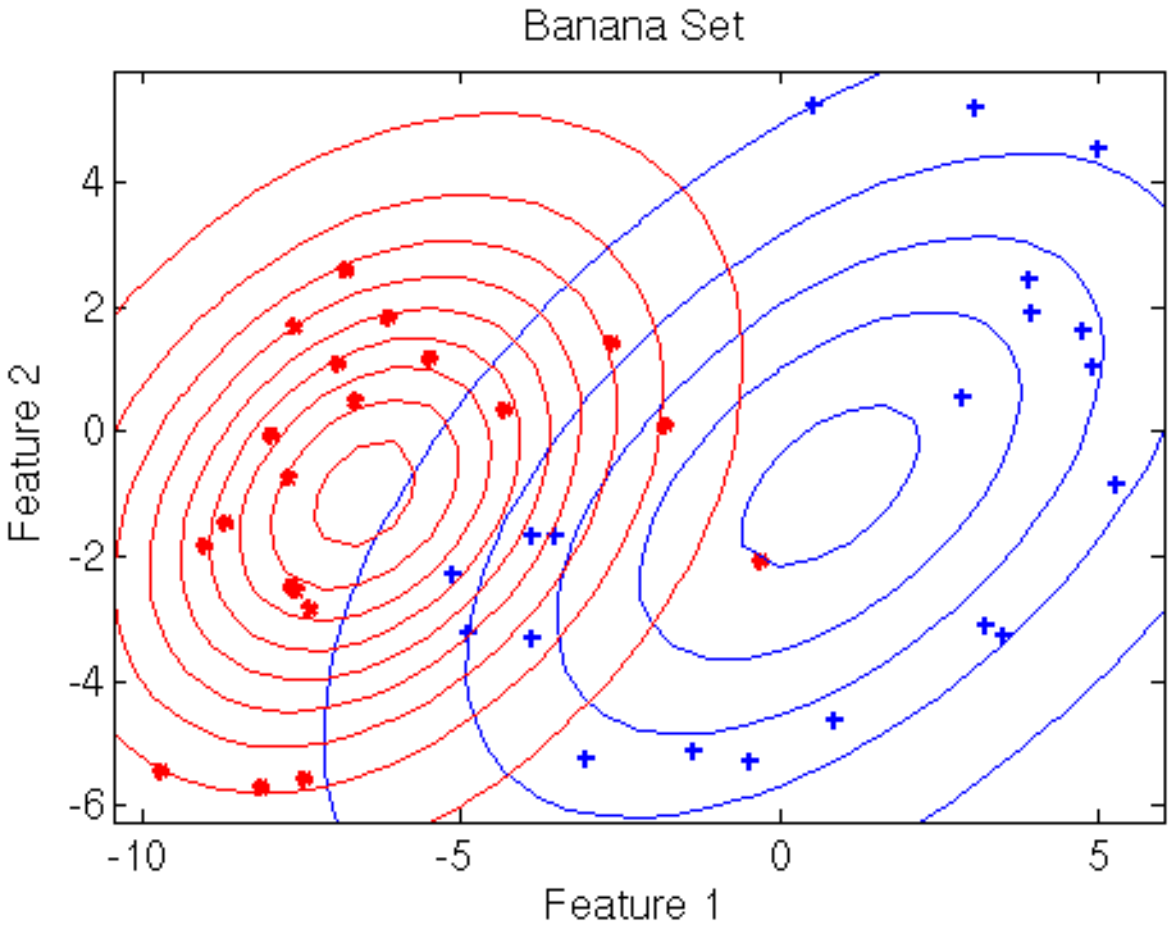
$$\hat{\mu}=\frac{1}{n}\sum_{i=1}^{n}\mathbf{x}_{i}$$

$$\hat{\Sigma}=\frac{1}{n}\sum_{i=1}^{n}(\mathbf{x}_{i}-\hat{\mu})(\mathbf{x}_{i}-\hat{\mu})^{T}$$

BioSB

Example on banana data

- A single Gaussian distribution on each class:



BioSB

Class-conditional densities

- Combining

$$\hat{p}(\mathbf{x}|\omega_i)=\frac{1}{\sqrt{2\pi^p det(\Sigma_i)}}\exp\left(-\frac{1}{2}(\mathbf{x}-\mu_i)^T\Sigma_i^{-1}(\mathbf{x}-\mu_i)\right)$$

$$p(\omega|\mathbf{x})=\frac{p(\mathbf{x}|\omega)p(\omega)}{p(\mathbf{x})}$$

we can derive for log(p):

$$\log(\hat{p}(\omega_i|\mathbf{x}))=-\frac{p}{2}\log(2\pi)-\frac{1}{2}\log(det\,\Sigma_i)$$

$$-\frac{1}{2}(\mathbf{x}-\mu_i)^T\Sigma_i^{-1}(\mathbf{x}-\mu_i)+\log(p(\omega_i))-\log(p(\mathbf{x}))$$

BioSB

Normal density-based classifier

- $p(\mathbf{x})$ is independent of the classes and can be dropped

$$g_i(\mathbf{x}) = -\frac{1}{2}\log(det\,\Sigma_i) - \frac{1}{2}(\mathbf{x}-\mu_i)^T\Sigma_i^{-1}(\mathbf{x}-\mu_i) + \log(p(\omega_i))$$

- Classifier becomes:

assign $\mathbf{x}$ to class ω_i when for all $i \neq j$: $g_i(\mathbf{x}) > g_j(\mathbf{x})$

BioSB

The two-class case

- Define the discriminant $f(\mathbf{x}) = p(\omega_1|\mathbf{x}) - p(\omega_2|\mathbf{x}) > 0$
- We get (laboratory exercise):

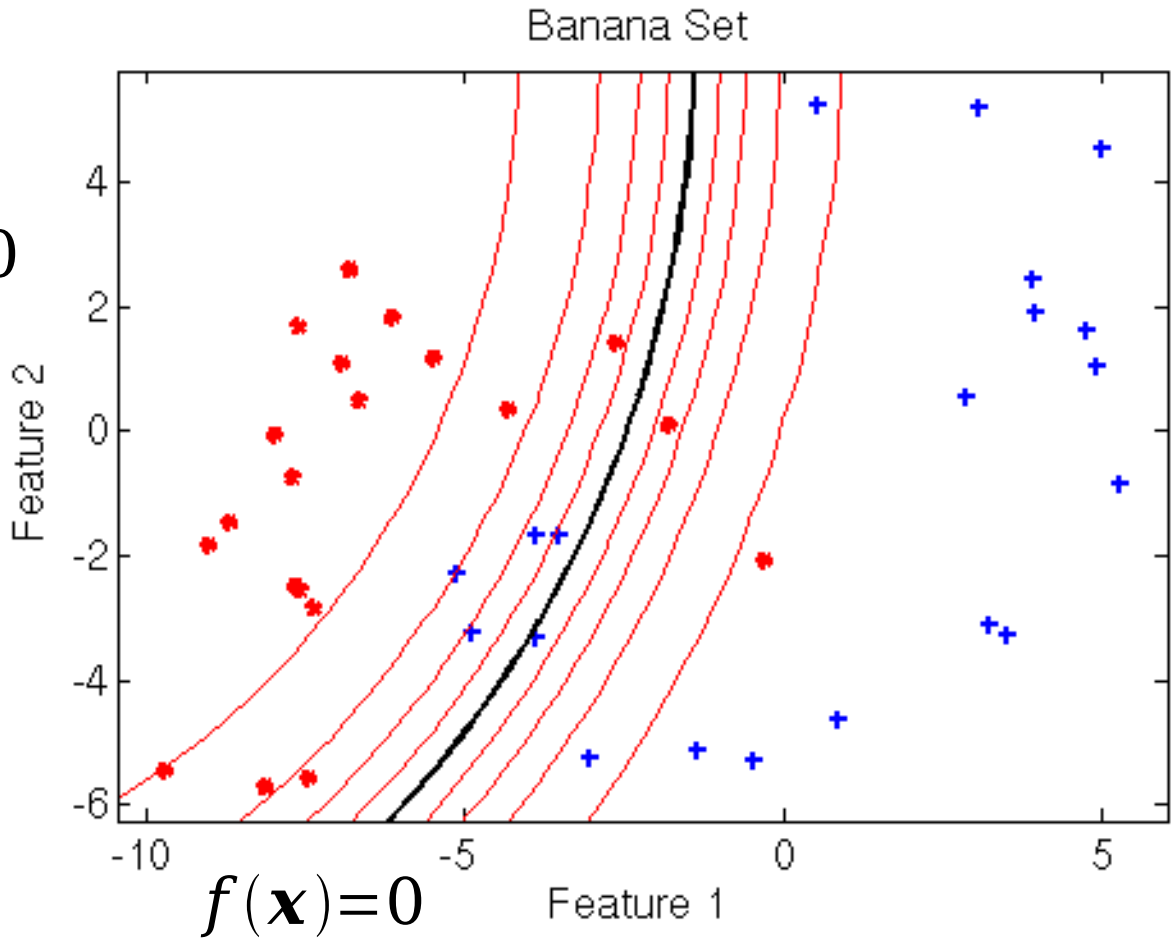
$$f(\mathbf{x}) = \mathbf{x}^T \boldsymbol{W} \mathbf{x} + \boldsymbol{w}^T \mathbf{x} + w_0$$

- This is a quadratic classifier because
  the decision boundary is a quadratic function of x

BioSB

Quadratic classifier on banana data

$f(\mathbf{x})>0$



$f(\mathbf{x})<0$

$f(\mathbf{x})=0$

Estimating the covariance matrix

- For the quadratic classifier you need to estimate

$$\hat{\boldsymbol{\Sigma}}_{\boldsymbol{k}} = \frac{1}{n} \sum_{i=1}^{n} (\boldsymbol{x}_i - \hat{\mu}_k)(\boldsymbol{x}_i - \hat{\mu}_k)^T$$

  for each of the classes!

- When you have insufficient data, this covariance matrix cannot be inverted
- Average over the covariance matrices of different classes:

$$\hat{\boldsymbol{\Sigma}} = \frac{1}{c} \sum_{k=1}^{c} \hat{\boldsymbol{\Sigma}}_{\boldsymbol{k}}$$

BioSB

Average covariance matrix

- When we use the averaged covariance matrix:

$$g_i(\boldsymbol{x}) = -\frac{1}{2}\log(det\,\hat{\Sigma}) - \frac{1}{2}(\boldsymbol{x}-\hat{\mu}_i)^T\hat{\Sigma}^{-1}(\boldsymbol{x}-\hat{\mu}_i) + \log(p(\omega_i))$$

- First term and quadratic term are always the same for all classes
- We end up with:

$$g_i(\boldsymbol{x}) = -\frac{1}{2}\hat{\mu}_i^T\hat{\Sigma}^{-1}\hat{\mu}_i - \frac{1}{2}\hat{\mu}_i^T\hat{\Sigma}^{-1}\boldsymbol{x} + \log(p(\omega_i))$$

- This classifier is *linear*:
  the linear normal density-based classifier.

BioSB

The two-class case (2)

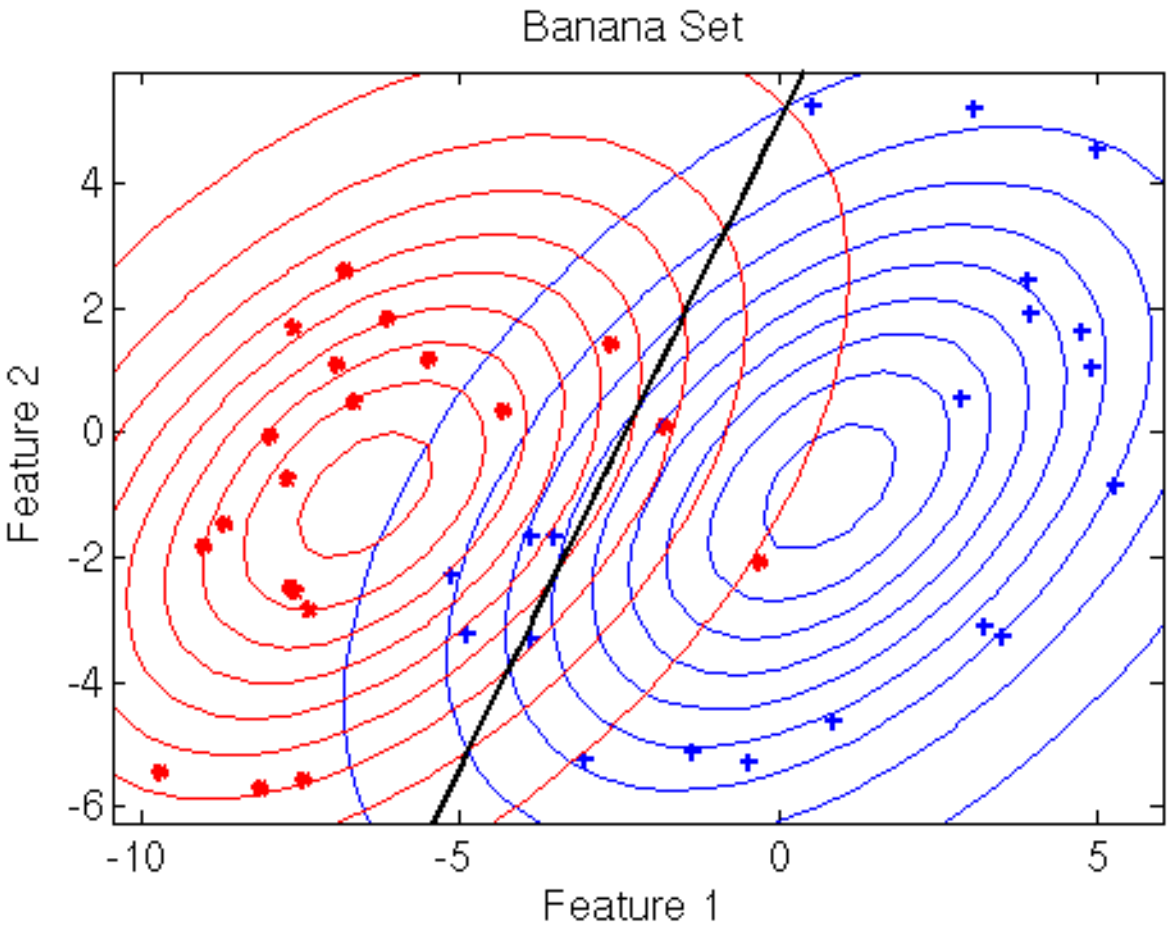
- Define the discriminant $f(\boldsymbol{x}) = p(\omega_1|\boldsymbol{x}) - p(\omega_2|\boldsymbol{x}) > 0$
- We get $f(\boldsymbol{x}) = \boldsymbol{w}^T \boldsymbol{x} + w_0$

with $\boldsymbol{w} = \hat{\Sigma}^{-1}(\hat{\mu}_1 - \hat{\mu}_2)$

$$w_0 = \frac{1}{2}\hat{\mu}_2^T \hat{\Sigma}^{-1} \hat{\mu}_2 - \frac{1}{2}\hat{\mu}_1^T \hat{\Sigma}^{-1} \hat{\mu}_1 + \log \frac{p(\omega_1)}{p(\omega_2)}$$

BioSB

Linear classifier on banana data



No estimated full covariance matrix

- In some cases even the averaged covariance matrix is too much to estimate
- Assume that all features have the same variance, and are uncorrelated:

$$\hat{\Sigma} = \sigma^2 I$$

- Then it becomes even simpler:

$$g_i(\mathbf{x}) = -\frac{1}{2\hat{\sigma}^2}(\hat{\mu}_i^T \hat{\mu}_i - \hat{\mu}_i^T \mathbf{x}) + \log(p(\omega_i))$$

BioSB

Nearest mean classifier

- Define the discriminant: $f(\boldsymbol{x}) = p(\omega_1|\boldsymbol{x}) - p(\omega_2|\boldsymbol{x}) > 0$
- We get $f(\boldsymbol{x}) = \boldsymbol{w}^T \boldsymbol{x} + w_0$

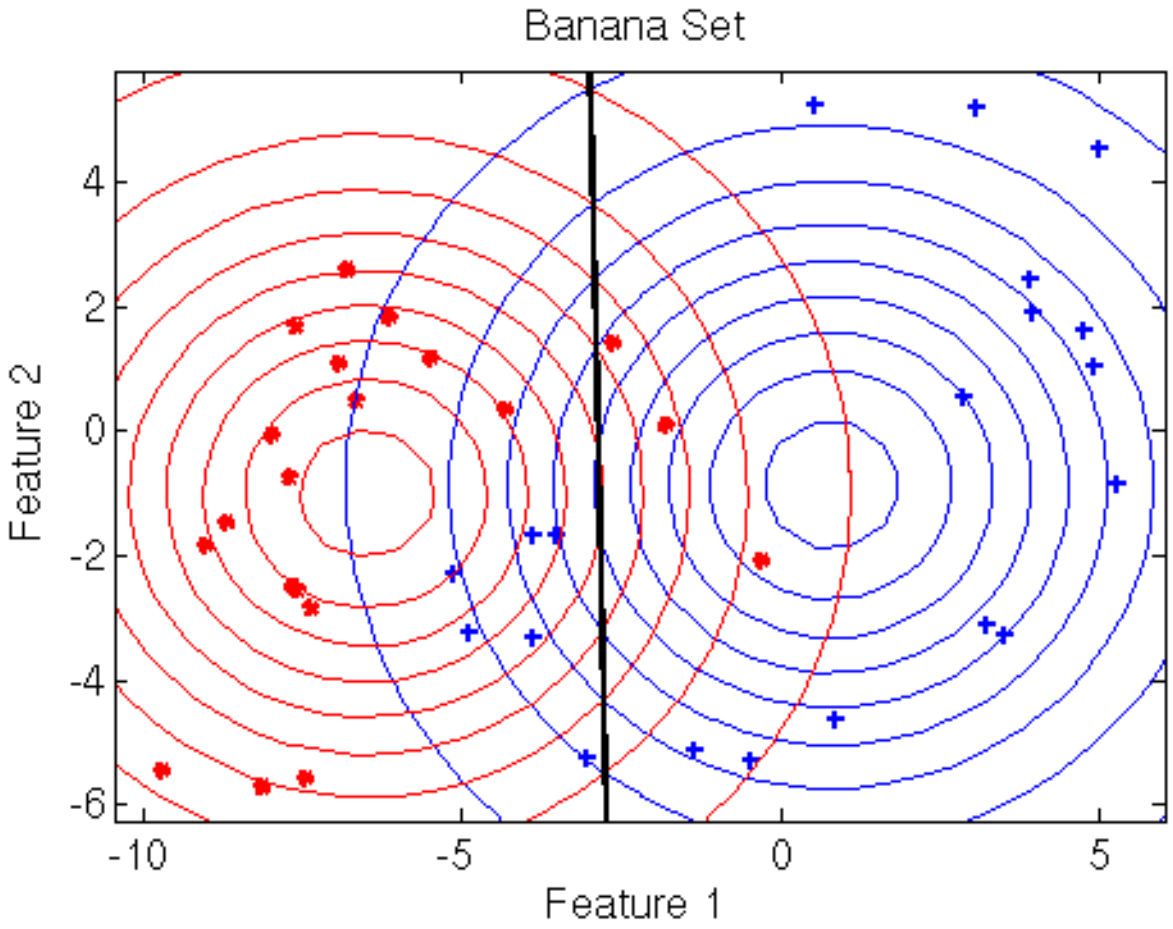
with $\boldsymbol{w} = \hat{\mu}_1 - \hat{\mu}_2$

$$w_0 = \frac{1}{2}\hat{\mu}_2^T \hat{\mu}_2 - \frac{1}{2}\hat{\mu}_1^T \hat{\mu}_1 + \hat{\sigma}^2 \log \frac{p(\omega_1)}{p(\omega_2)}$$

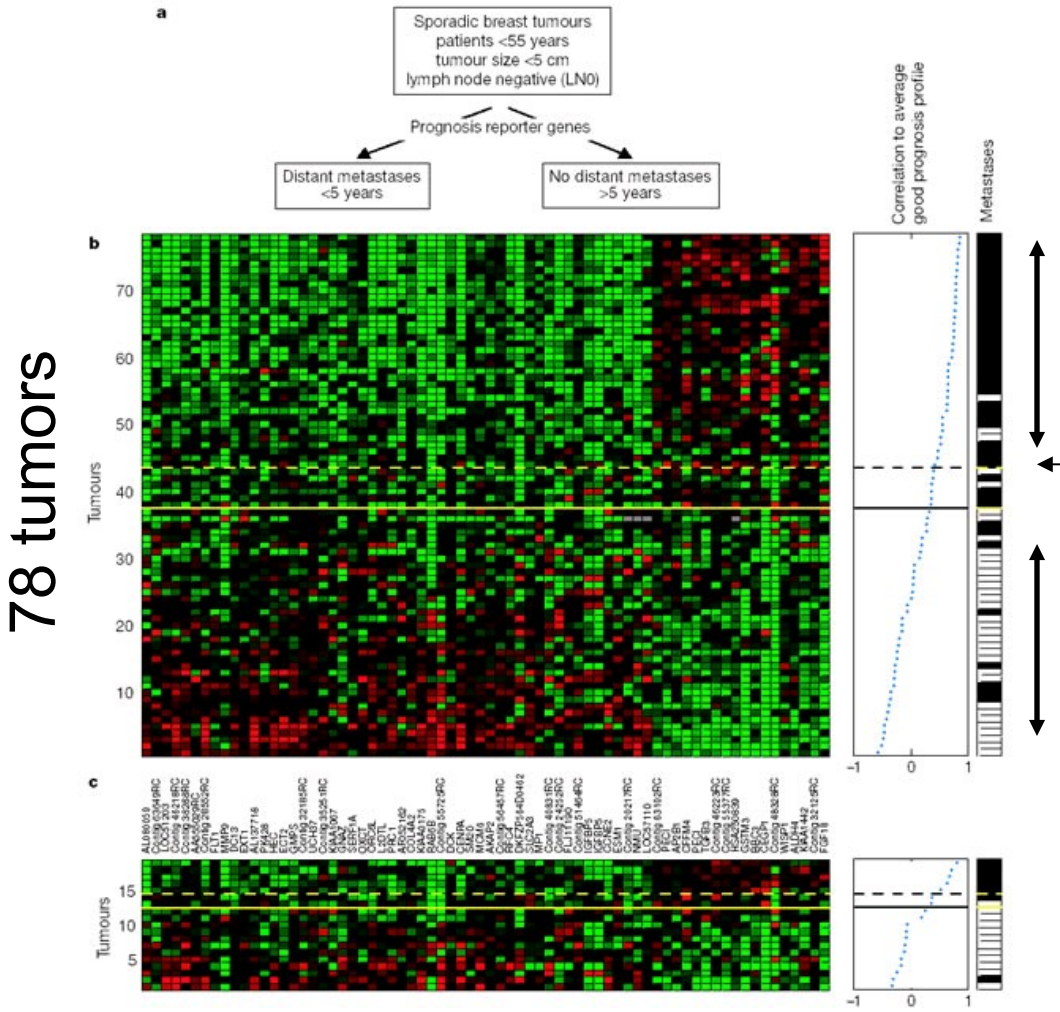
- Again a linear classifier, but it only uses the distance to the mean of each of the classes: *nearest mean* classifier

BioSB

Nearest mean on banana data



Nearest mean on gene expression data



78 tumors

70 genes (= biomarkers)

Two classes:

no metastases > 5yr

metastases < 5yr

good prognosis

threshold

bad prognosis

Van 't Veer et al, Nature **415**, 530 (2002)

BioSB

ROC curve

- Recall minimum cost classification:

$$c_{opt} = \arg\min_{c'} \sum_{c=1}^{C} \Lambda(\omega = c', \omega = c)\, p(\omega = c \mid x) p(x)$$

$$= \arg\min_{c'} \sum_{c=1}^{C} \Lambda(\omega = c', \omega = c)\, p(x \mid \omega = c) p(\omega = c)$$

- In the two-class case, cost can be absorbed into prior:

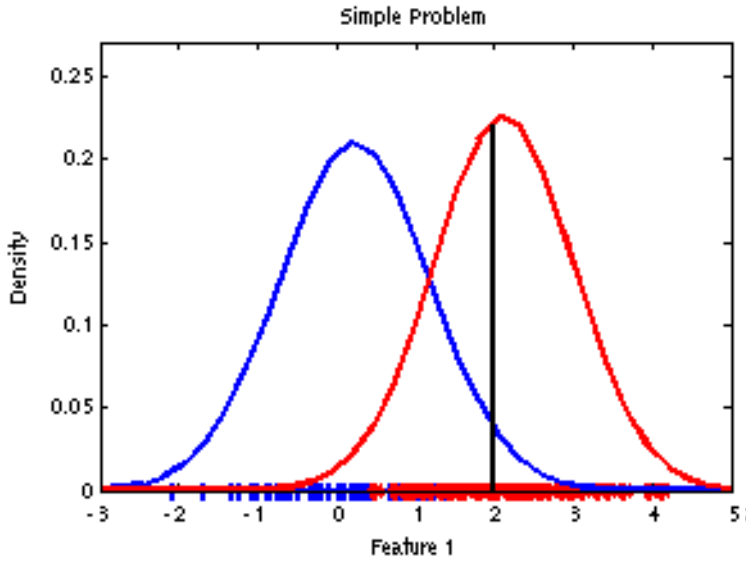
$$c_{opt} = \arg\min_{c'} \sum_{c=1}^{2} p(x \mid \omega = c) \tilde{p}(\omega = c)$$

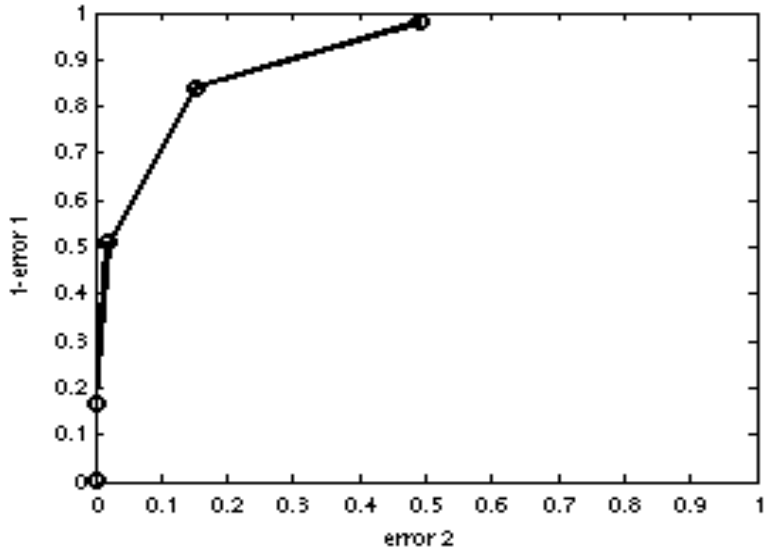
i.e. changing the costs is like changing the class priors

BioSB

ROC curve (2)

- Error as a function of the threshold gives an overview of all possible cost/prior scenarios: *receiver-operator characteristic curve*
- Classifier: any x left of the threshold belongs to the blue class, any x to the right to the red class

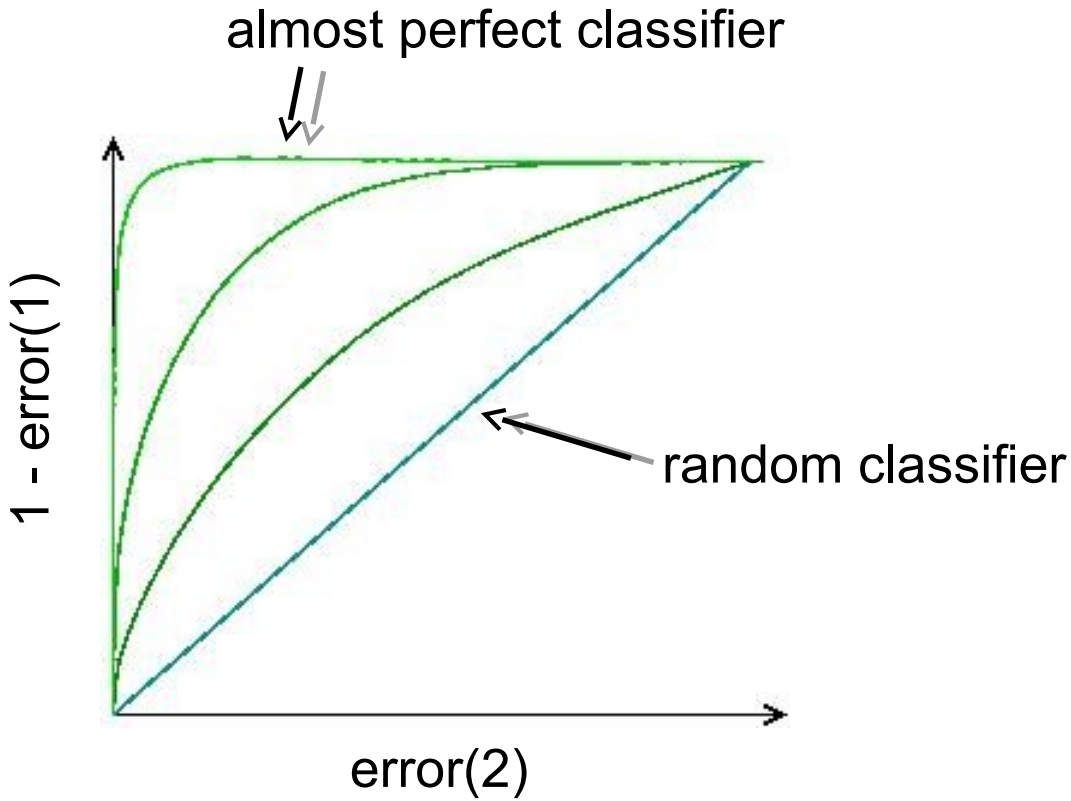




BioSB

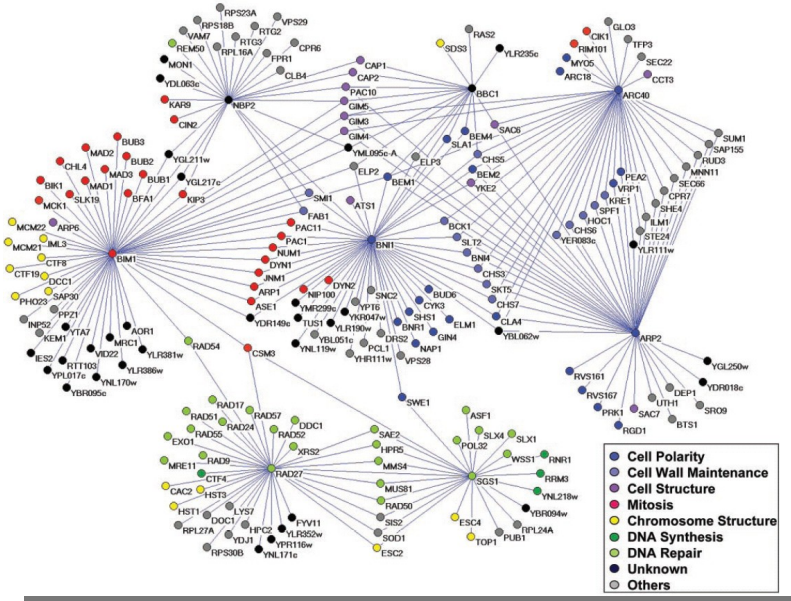
ROC curve (3)

- Different classifiers have different ROC curves

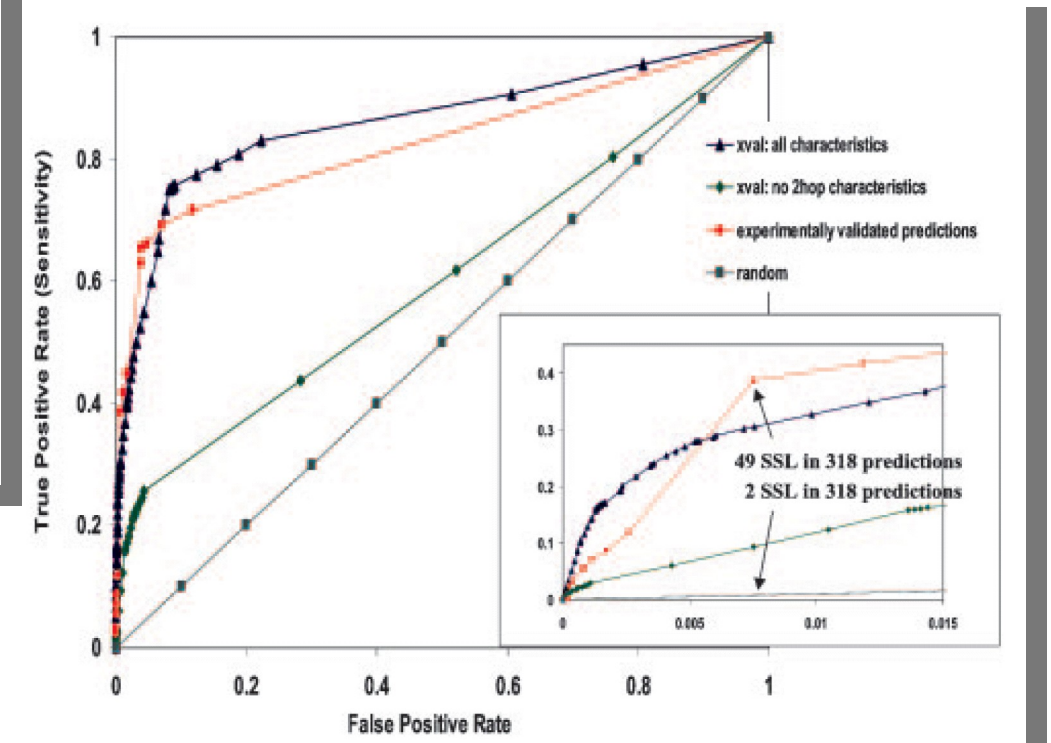


ROC curve (4)

- Example: prediction of synthetic genetic interactions (SGAs)



Wong et al., PNAS 2004



ROC for two-class problems: changing threshold

- Changing class costs = changing priors = moving the decision boundary = changing threshold
- Look at the general form of the normal-based classifiers:

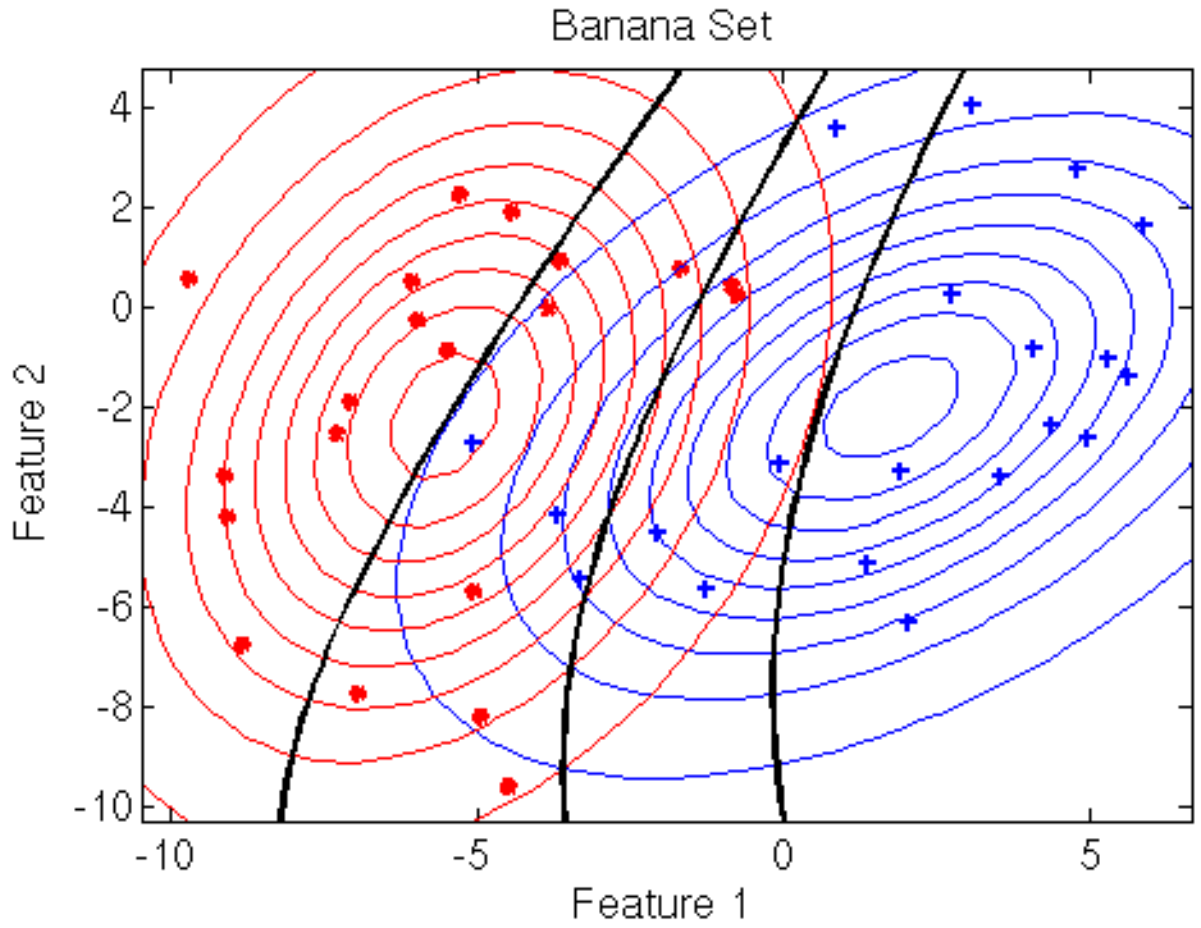
$$g_i(\boldsymbol{x}) = -\frac{1}{2}\log(det\,\Sigma_i) - \frac{1}{2}(\boldsymbol{x}-\mu_i)^T\Sigma_i^{-1}(\boldsymbol{x}-\mu_i) + \log(p(\omega_i))$$

- Changing the prior affects only the 'offset' (=threshold)
- It means only the thresholds have to be adapted:

$$f(\boldsymbol{x}) = \boldsymbol{w}^T\boldsymbol{x} + w_0$$
$$f(\boldsymbol{x}) = \boldsymbol{x}^T\boldsymbol{W}\boldsymbol{x} + \boldsymbol{w}^T\boldsymbol{x} + w_0$$

BioSB

Changing threshold in banana data



BioSB

Recapitulation

- Using the Parzen density and nearest neighbor density we can derive the Parzen classifier and nearest neighbor classifier
- Using the plug-in Bayes' rule with a normal distribution for each of the classes gives different classifiers
  - Separate mean and covariance matrix per class gives the quadratic classifier
  - Separate mean, equal covariance matrix per class gives the linear classifier (see Fisher classifier, for two classes)
  - Separate mean, identity covariance matrix per class gives the nearest mean classifier
- By changing the thresholds a ROC curve is obtained, showing the error on both classes.

BioSB

Discriminant analysis

- Different approach to classifiers: avoid estimating the (class conditional) probabilities altogether

  - Linear discriminant
  - Fisher classifier

BioSB

Avoid density estimation

- From the *k*-nearest neighbor we saw already that we don't need to explicitly estimate a density
- Estimating densities is hard, in particular when we have a high number of features (high dimensional feature space, curse of dimensionality)
- Now, we start from the other end:
  - Assume we have a function to describe the decision boundary
  - Optimize the free parameters of this function directly
  - No Bayes' theorem, no density estimates

Linear discriminant

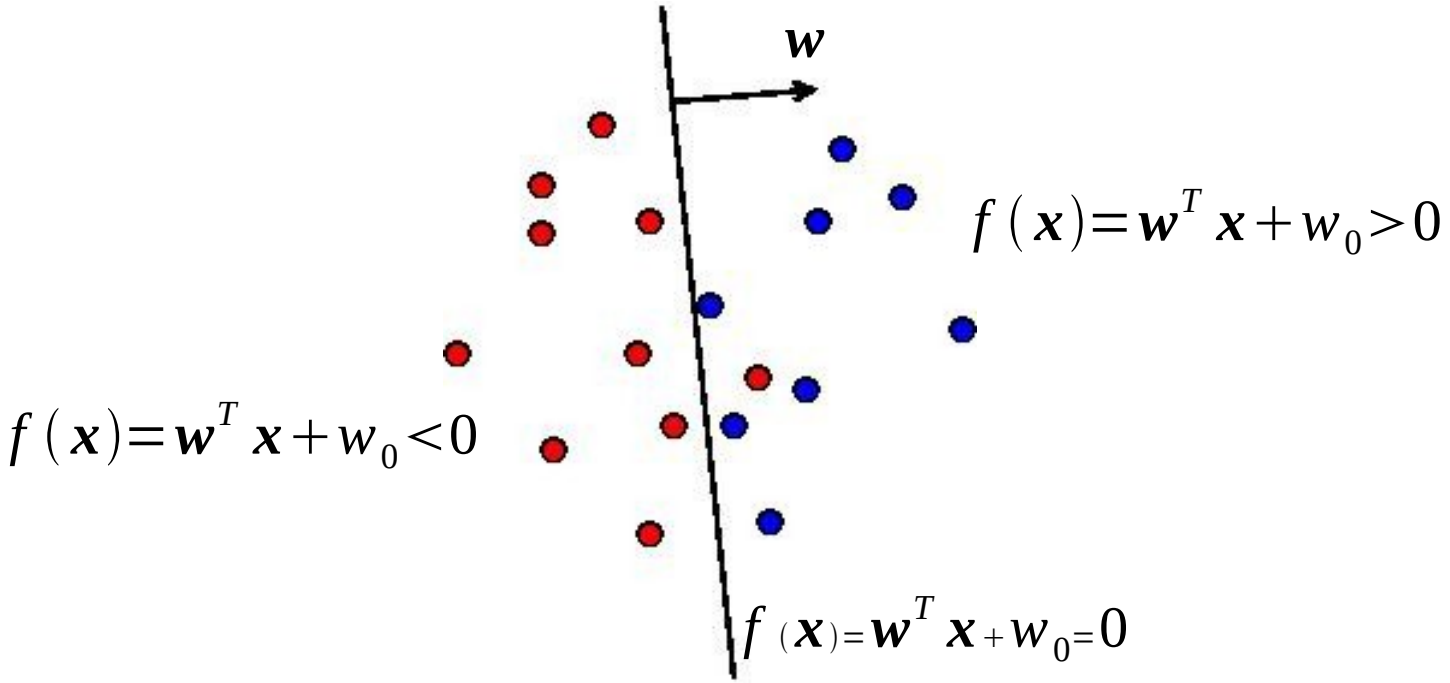
- Let us assume we can describe the discriminant by:

$$f(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + w_0$$

- There are several ways to optimize $\mathbf{w}$ and w_0
- This is generally called linear discriminant analysis

BioSB

Linear discriminant (2)



- Classifier is a linear function of the features
- The classification depends on whether the weighted sum of the features is above or below 0

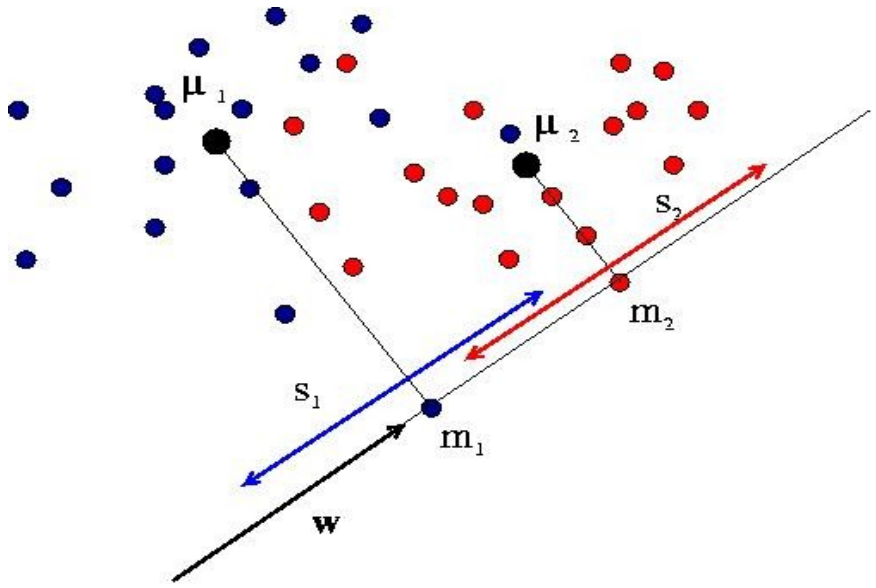
Fisher classifier

- Linear projection onto 1-D:

$$y = \boldsymbol{w}^T \boldsymbol{x}$$

- Maximize Fisher criterion:

$$J = \frac{|m_1 - m_2|^2}{(s_1^2 + s_2^2)}$$



- Maximizing *J* implies that after projection:
  - Means should be far apart
  - Variances should be small
- Find a projection direction $\boldsymbol{w}$ for which *J* is optimized

BioSB

Derivation Fisher classifier

- Map the means on w:

$$m_1 = \boldsymbol{w}^T \boldsymbol{\mu}_1, \qquad m_2 = \boldsymbol{w}^T \boldsymbol{\mu}_2$$

- Map the differences in mean:

$$|m_1 - m_2|^2 = (\boldsymbol{w}^T \boldsymbol{\mu}_1 - \boldsymbol{w}^T \boldsymbol{\mu}_2)^2$$
$$= \boldsymbol{w}^T (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)^T \boldsymbol{w} = \boldsymbol{w}^T \boldsymbol{S}_B \boldsymbol{w}$$

- Compute the mapped variance:

$$s_i^2 = \sum_j (\boldsymbol{w}^T \boldsymbol{x}_j^{(i)} - \boldsymbol{w}^T \boldsymbol{\mu}_i)^2$$
$$= \sum_j \boldsymbol{w}^T (\boldsymbol{x}_j^{(i)} - \boldsymbol{\mu}_i)(\boldsymbol{x}_j^{(i)} - \boldsymbol{\mu}_i)^T \boldsymbol{w} = \boldsymbol{w}^T \boldsymbol{S}_i \boldsymbol{w}$$

BioSB

Derivation Fisher discriminant

- Combine both results from the previous slide.
- The Fisher criterion $$J = \frac{|m_1 - m_2|^2}{(s_1^2 + s_2^2)}$$

can be written in terms of the weights

$$J(\boldsymbol{w}) = \frac{\boldsymbol{w}^T \boldsymbol{S}_B \boldsymbol{w}}{\boldsymbol{w}^T \boldsymbol{S}_W \boldsymbol{w}}$$

where $\boldsymbol{S}_W = \sum_i \frac{n_i}{n} \boldsymbol{S}_i$ is the 'within scatter matrix'

and $\boldsymbol{S}_B = (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)^T$ the 'between scatter matrix'.

BioSB

Derivation Fisher discriminant (2)

- To optimize *J*, we set the derivative to 0:

$$(\boldsymbol{w}^T \boldsymbol{S}_B \boldsymbol{w}) \boldsymbol{S}_W \boldsymbol{w} = (\boldsymbol{w}^T \boldsymbol{S}_W \boldsymbol{w}) \boldsymbol{S}_B \boldsymbol{w}$$

- Because $\boldsymbol{S}_B = (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)^T$

$\boldsymbol{S}_B \boldsymbol{w}$ will always be in the direction $(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)$

- We get: $(\boldsymbol{w}^T (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)) \boldsymbol{S}_W \boldsymbol{w} = (\boldsymbol{w}^T \boldsymbol{S}_W \boldsymbol{w})(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)$

BioSB

Derivation Fisher discriminant (3)

- Ignoring scalar factors, we get:

$$(\boldsymbol{w}^T(\boldsymbol{\mu}_1-\boldsymbol{\mu}_2))\boldsymbol{S}_W\boldsymbol{w}=(\boldsymbol{w}^T\boldsymbol{S}_W\boldsymbol{w})(\boldsymbol{\mu}_1-\boldsymbol{\mu}_2)$$

$$\boldsymbol{S}_W\boldsymbol{w}=C\cdot(\boldsymbol{\mu}_1-\boldsymbol{\mu}_2)$$

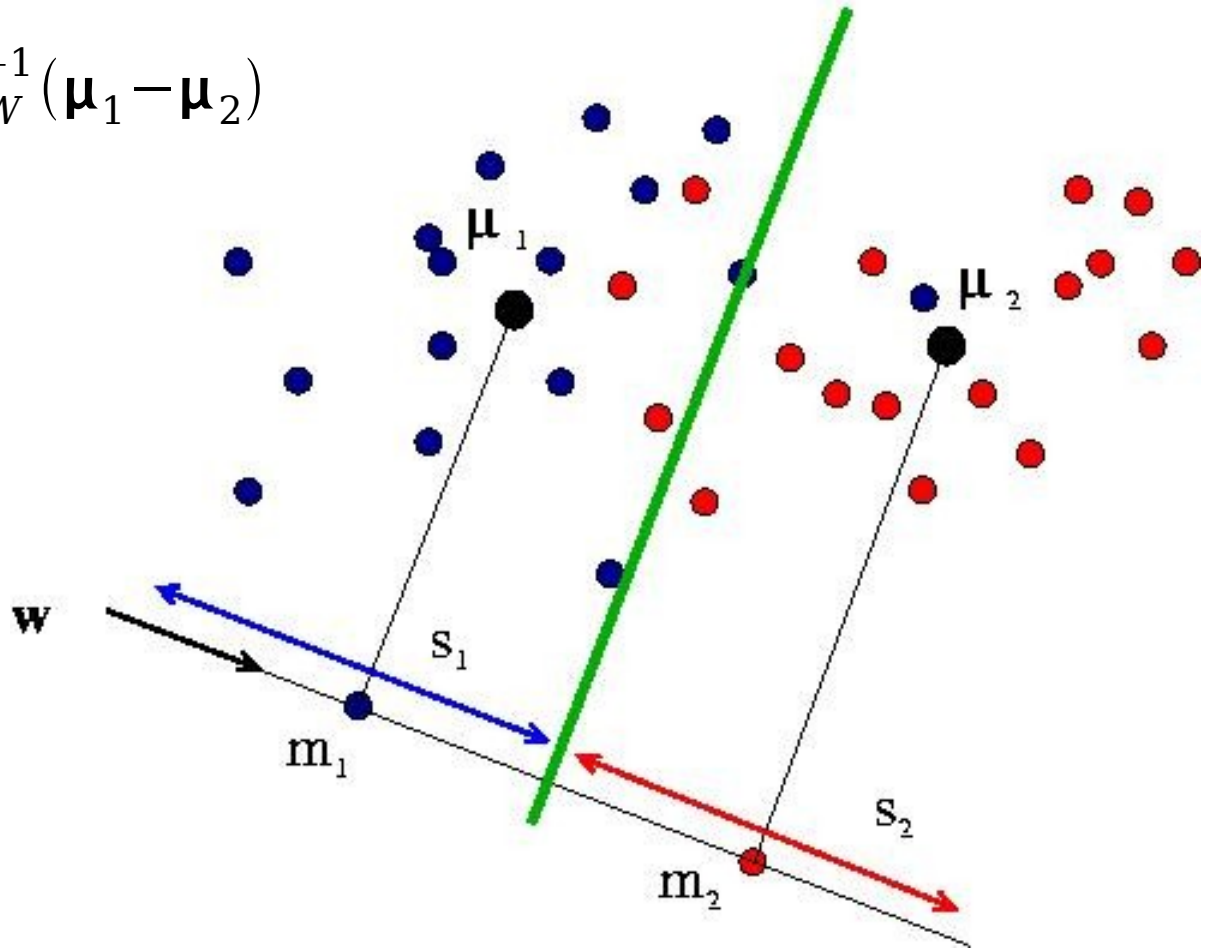
$$\boldsymbol{w}\sim\boldsymbol{S}_W^{-1}(\boldsymbol{\mu}_1-\boldsymbol{\mu}_2)$$

- Strictly speaking, we don't have a classifier yet, only a direction on which to project our data
- In practice, take the decision boundary in the middle

BioSB

The result

$$\boldsymbol{w} \sim \boldsymbol{S}_W^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)$$



BioSB

This is familiar...

- The expression for the Fisher discriminant

$$\boldsymbol{w} \sim \boldsymbol{S}_W^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)$$

looks like the linear normal-based classifier:

$$f(\boldsymbol{x}) = \boldsymbol{w}^T \boldsymbol{x} + w_0$$

$$\boldsymbol{w} = \hat{\Sigma}^{-1}(\hat{\boldsymbol{\mu}}_1 - \hat{\boldsymbol{\mu}}_2)$$

$$w_0 = \frac{1}{2}\hat{\boldsymbol{\mu}}_2^T \hat{\Sigma}^{-1}\hat{\boldsymbol{\mu}}_2 - \frac{1}{2}\hat{\boldsymbol{\mu}}_1^T \hat{\Sigma}^{-1}\hat{\boldsymbol{\mu}}_1 + \log\frac{p(\omega_1)}{p(\omega_2)}$$

- For a two-class problem, both classifiers are identical

BioSB

Comparison Fisher and Gauss

- The normal-based linear classifier assumes a density per class
- Fisher classifier just tries to optimise the Fisher criterion
  - For the Fisher classifier the bias term is (in principle) still free to optimise

- Both classifiers rely on the inverse of $\boldsymbol{S}_W$, so it can therefore become undefined when insufficient data is available

BioSB

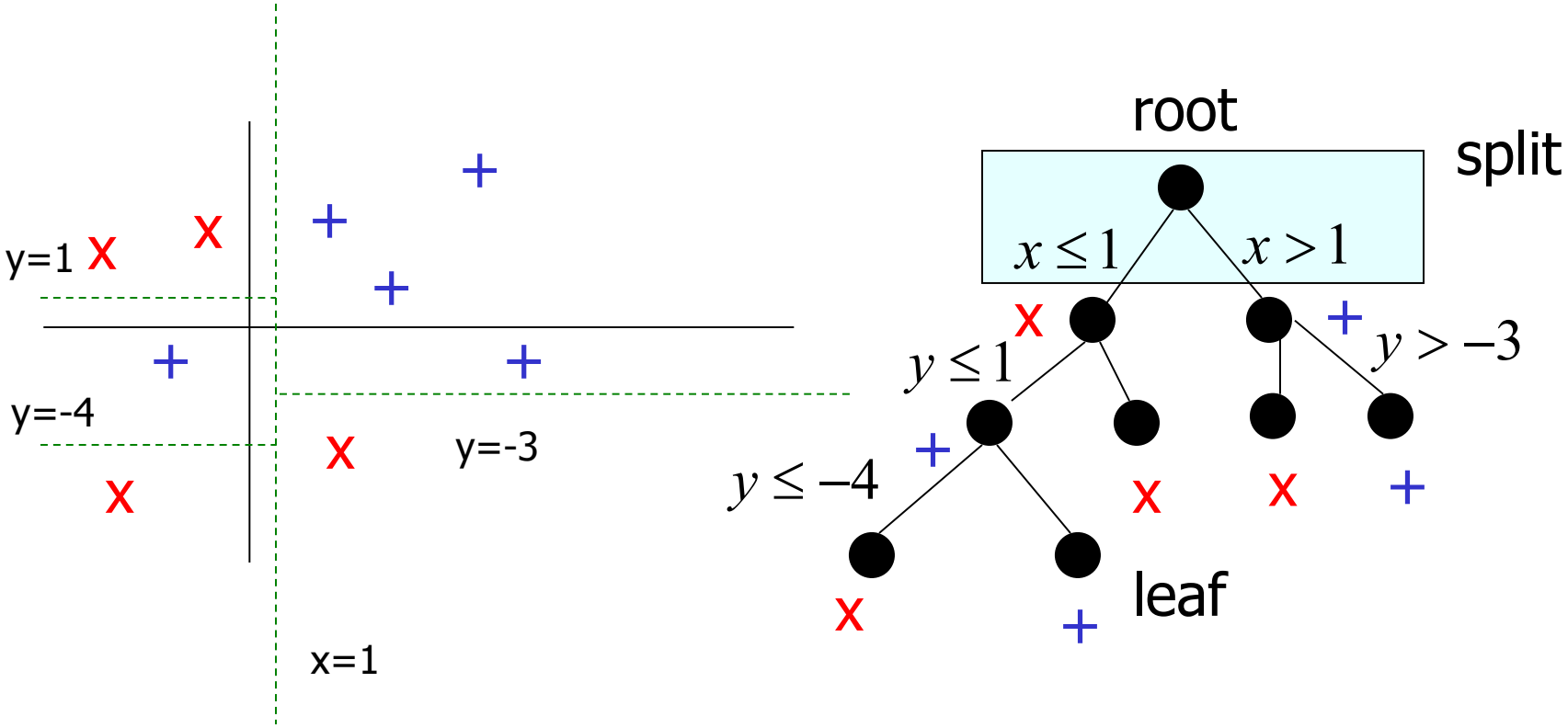
Tree-based models

- Until now: mainly linear and quadratic decision surfaces, often real data is more complex

- Classification trees
  - Feature selection
- Random forests
  - Ensemble of trees
  - Randomization
  - Bootstrapping
- More on Day 5: neural networks, support vector machines

BioSB

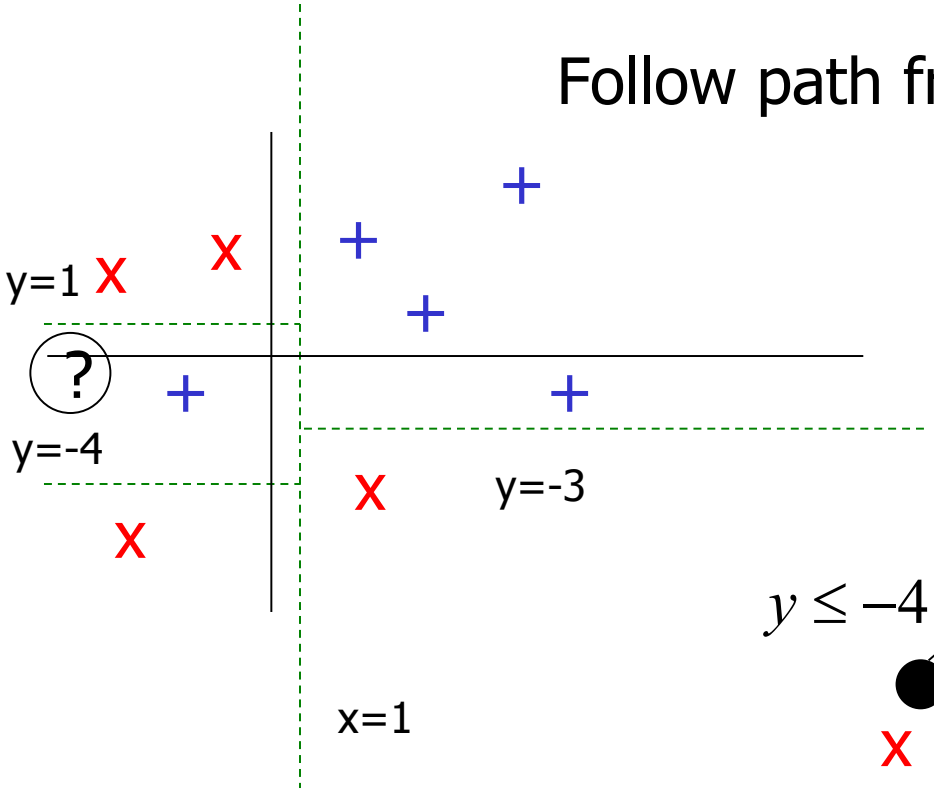
Classification trees

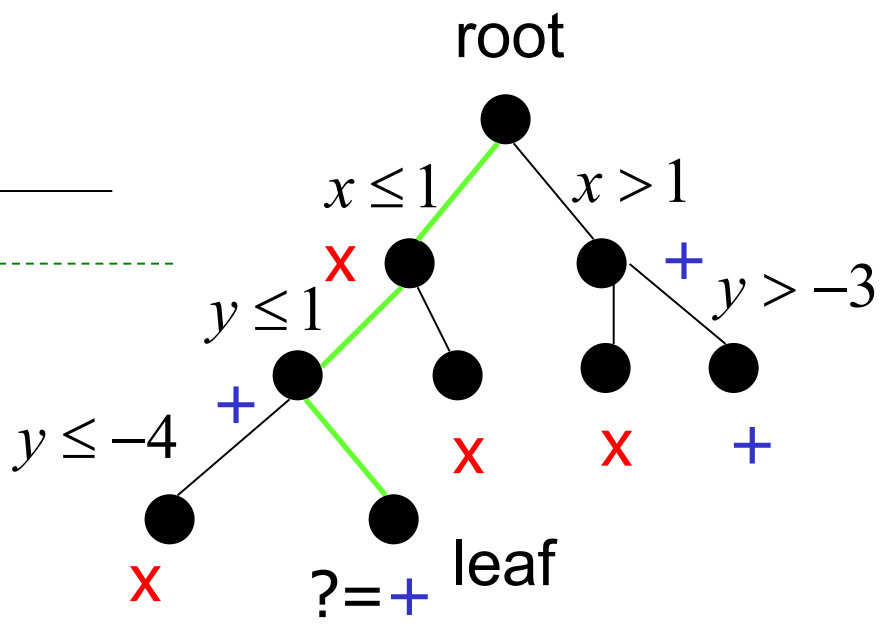


Build a tree of (binary) splits parallel to the axes in a greedy (=one by one) way.

Classification trees: new data

Follow path from root to one of the leaves

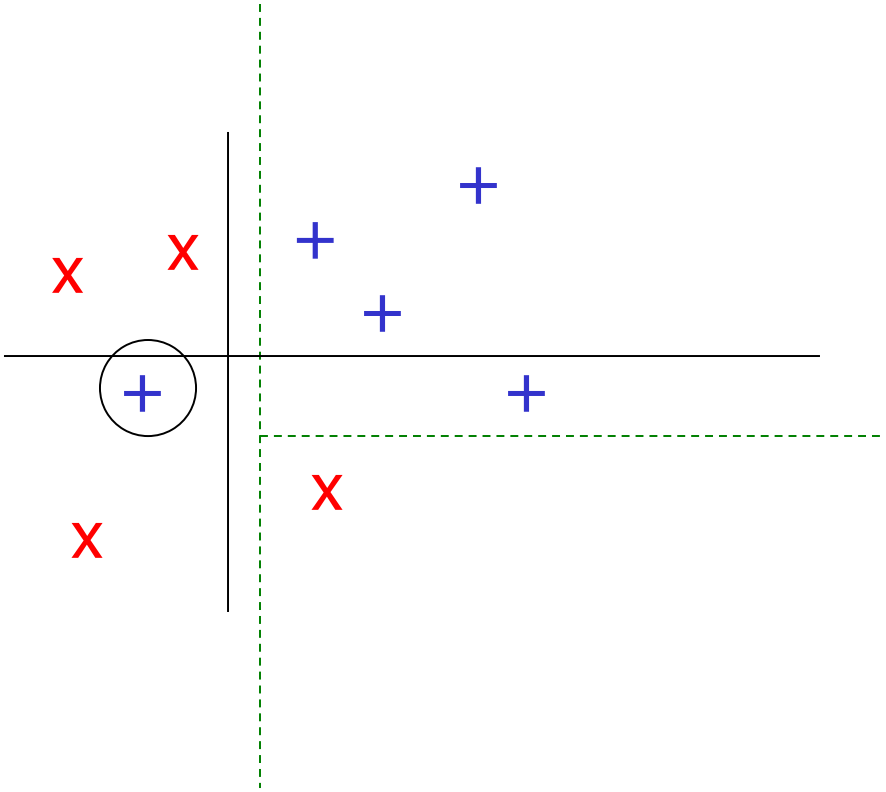


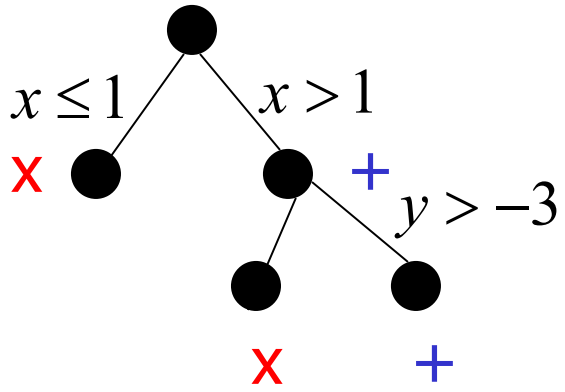


Can perfectly fit the data: overfitting

BioSB

Classification trees: pruning





Allow errors on training data in order to reduce overfitting

Tree ingredients

Trees are constructed in a greedy way:
starting with an empty tree and adding splits one by one
(and never coming back on a decision taken)

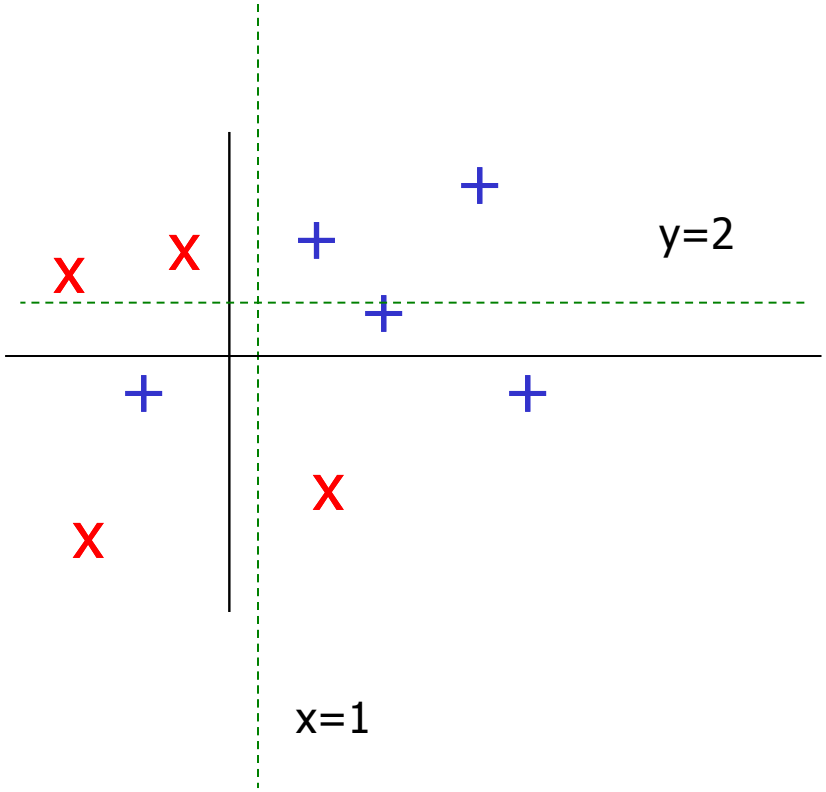
Main questions:

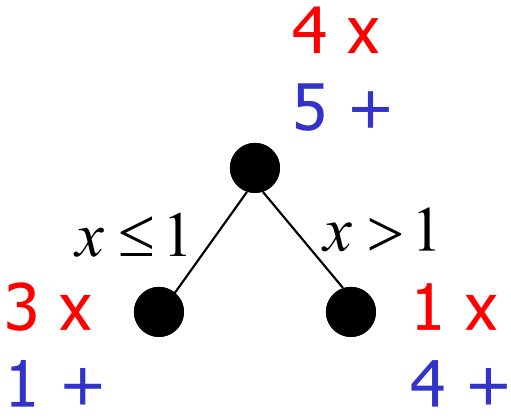
- How to choose a split
- How to choose a final tree?
  - Amount of pruning

Rest: details (but might be important …)

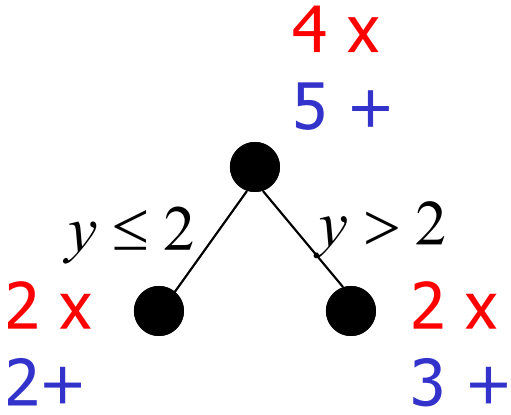
BioSB

How to choose a split?



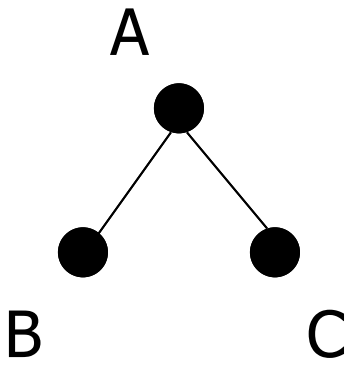


good



bad

How to choose a split? (2)

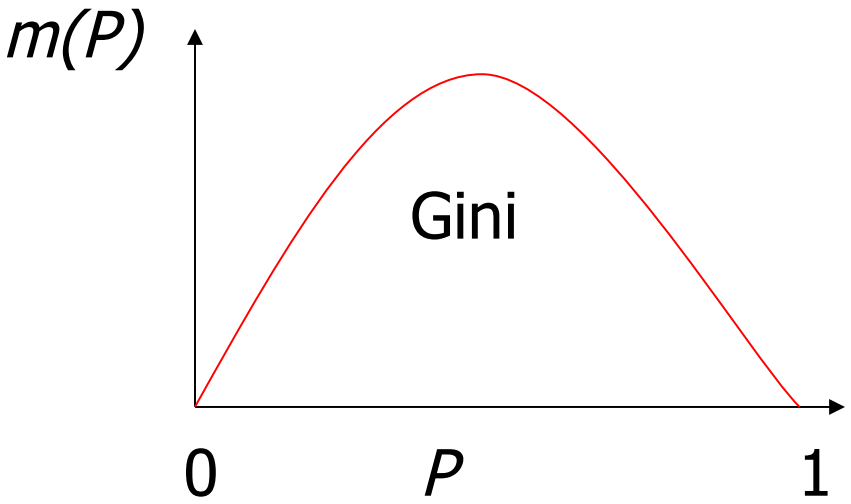


Good split at A:

- few x & many + in B, C

- many x & few + in B, C

Find some measure *m* that captures goodness



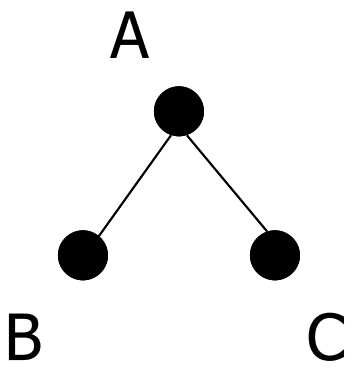
$m(0)=0$

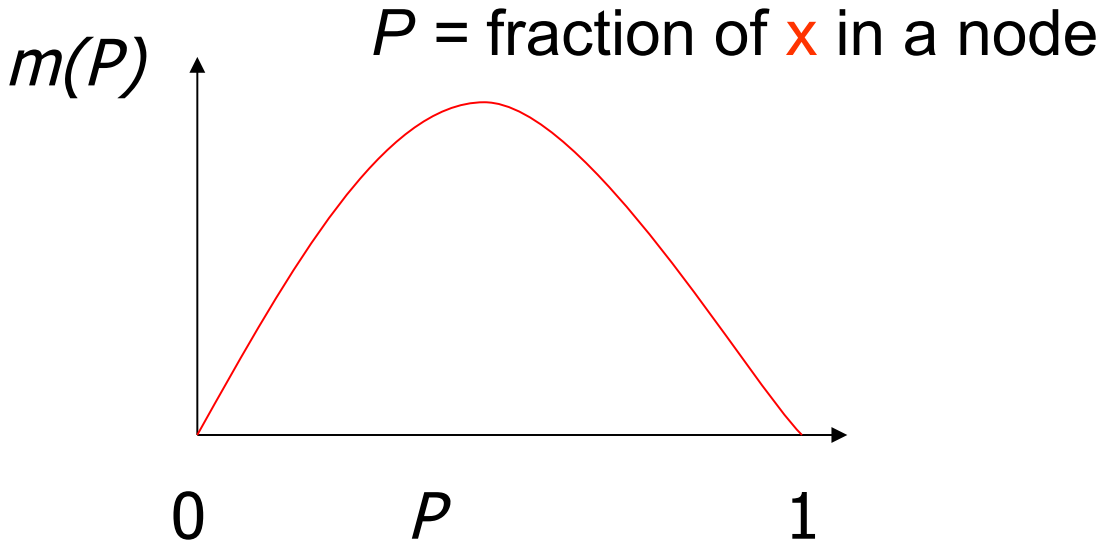
$m(1)=0$

$m(P)=P(1-P)$

P = fraction of x in a node

How to choose a split? (3)

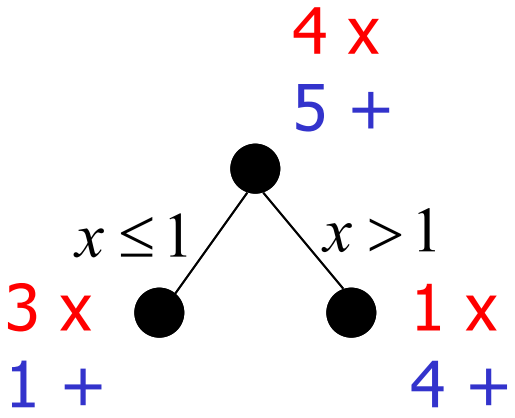




maximize $m(P_A) - P(B)m(P_B) - P(C)m(P_C)$

$P(X)$: determined by number of x and + at node X

How to choose a split? (4)



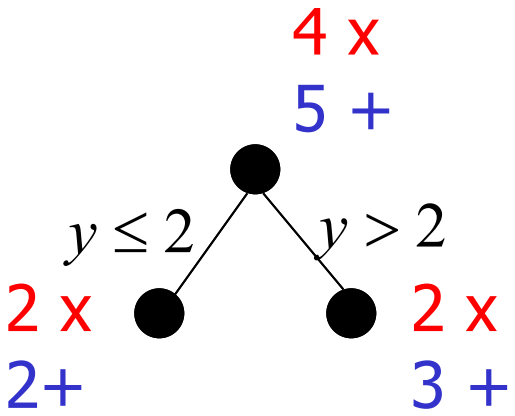
good

$$m(P_A) - P(B)m(P_B) - P(C)m(P_C)$$

$$\frac{4}{9}\frac{5}{9} - \frac{4}{9}\frac{3}{4}\frac{1}{4} - \frac{5}{9}\frac{1}{5}\frac{4}{5} =$$

0.075

maximum



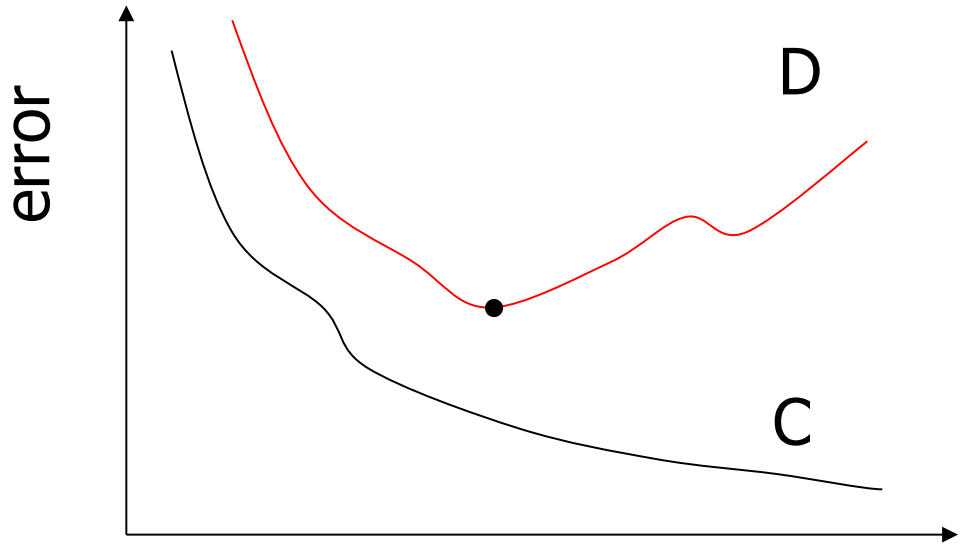
bad

$$\frac{4}{9}\frac{5}{9} - \frac{4}{9}\frac{2}{4}\frac{2}{4} - \frac{5}{9}\frac{2}{5}\frac{3}{5} =$$

0.0025

BioSB

Pruning: one step back

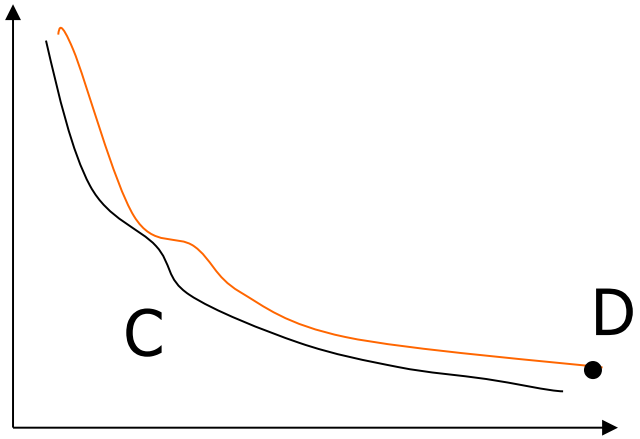


minimize: D = C + *k*(# of leaf nodes in the tree)

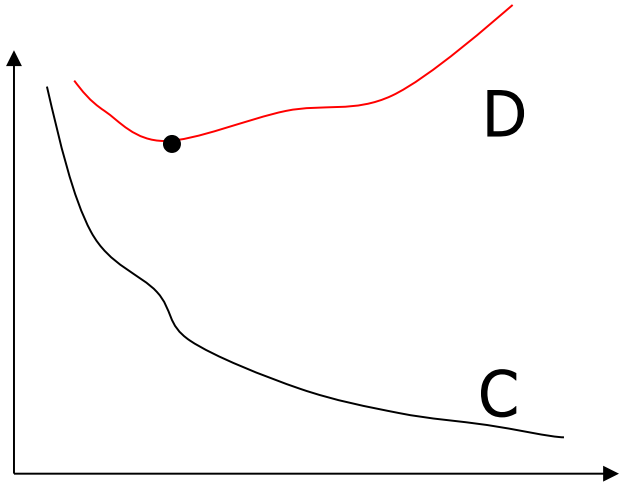
$0 \leq k$ *k*: complexity parameter

k penalizes big trees

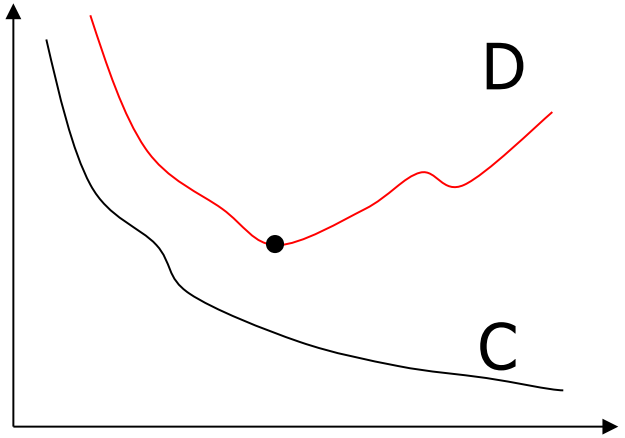
Pruning: one step back (2)



small *k*: big tree



large *k*: small tree



medium *k*: medium tree

BioSB

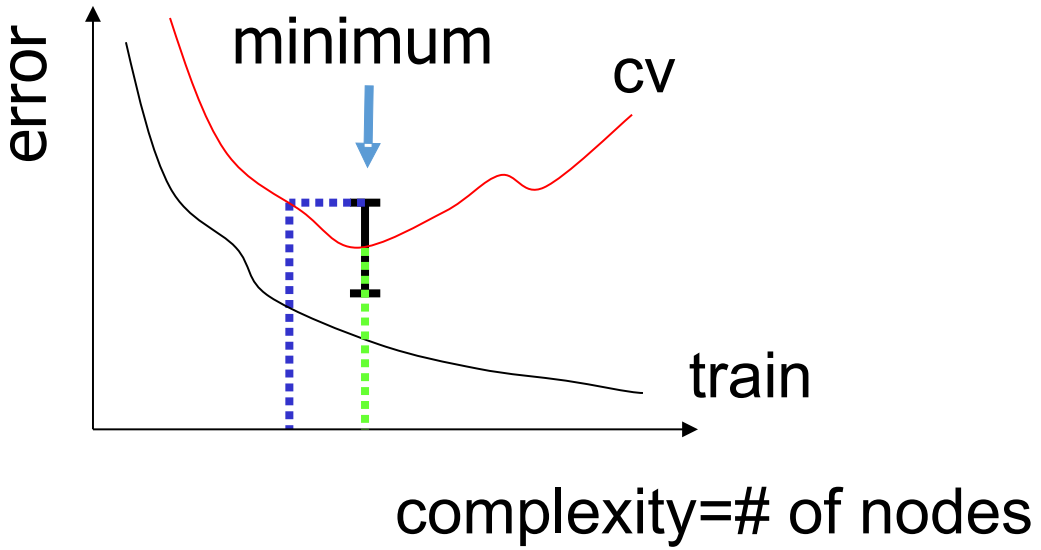
Pruning: CART

- Build a complete tree *T*
- With each subtree of *T* corresponds a choice of *k*

Cannot make choice of *k* on training set: overfitting

Optimal choice of *k* is made by cross-validation

Pruning: model selection



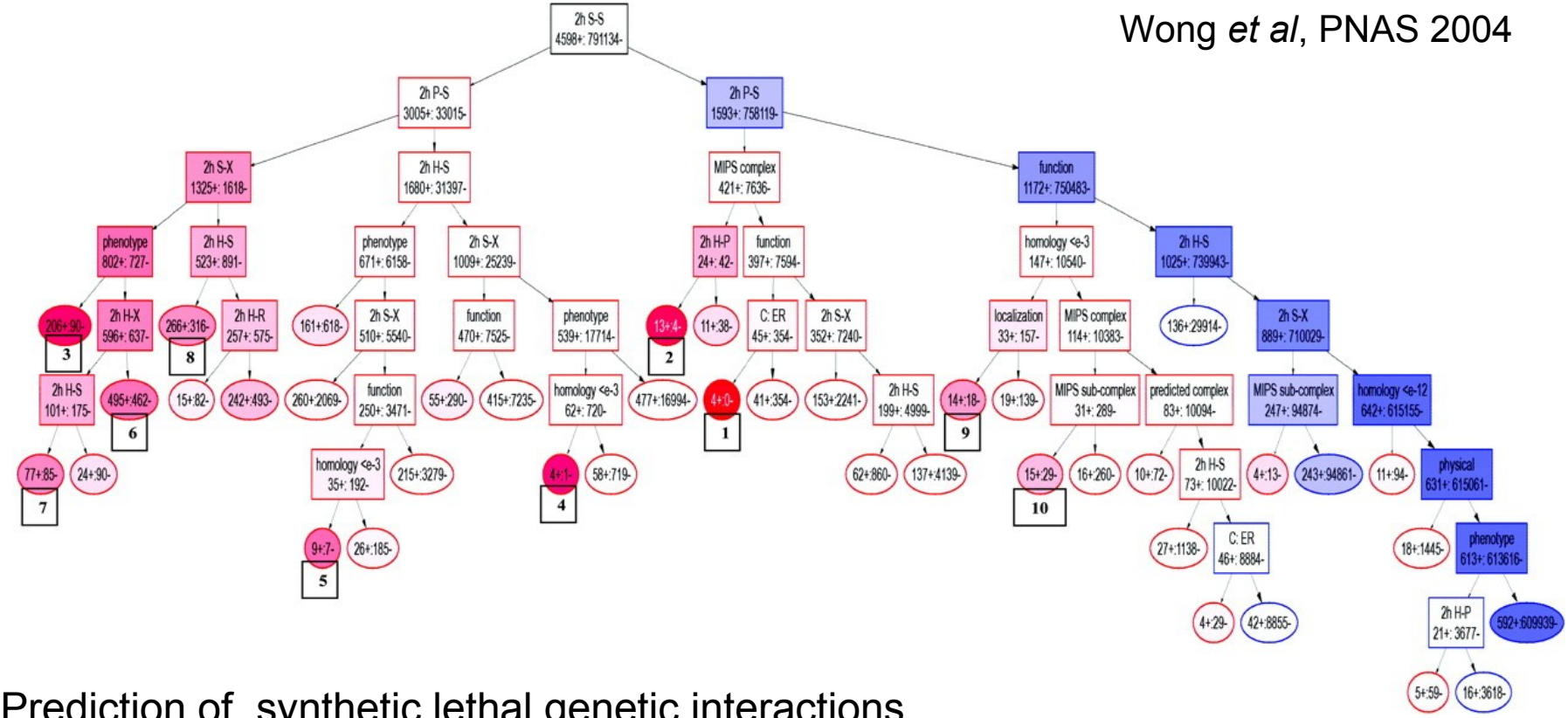
minimum

standard error (1-se)

10-fold cross-validation: mean +/- std. error

Decision tree: application

Wong *et al*, PNAS 2004



- Prediction of synthetic lethal genetic interactions
- Integrate multiple types of data: localization, mRNA expression, physical interaction, protein function, and characteristics of network topology

BioSB

Advantages/disadvantages

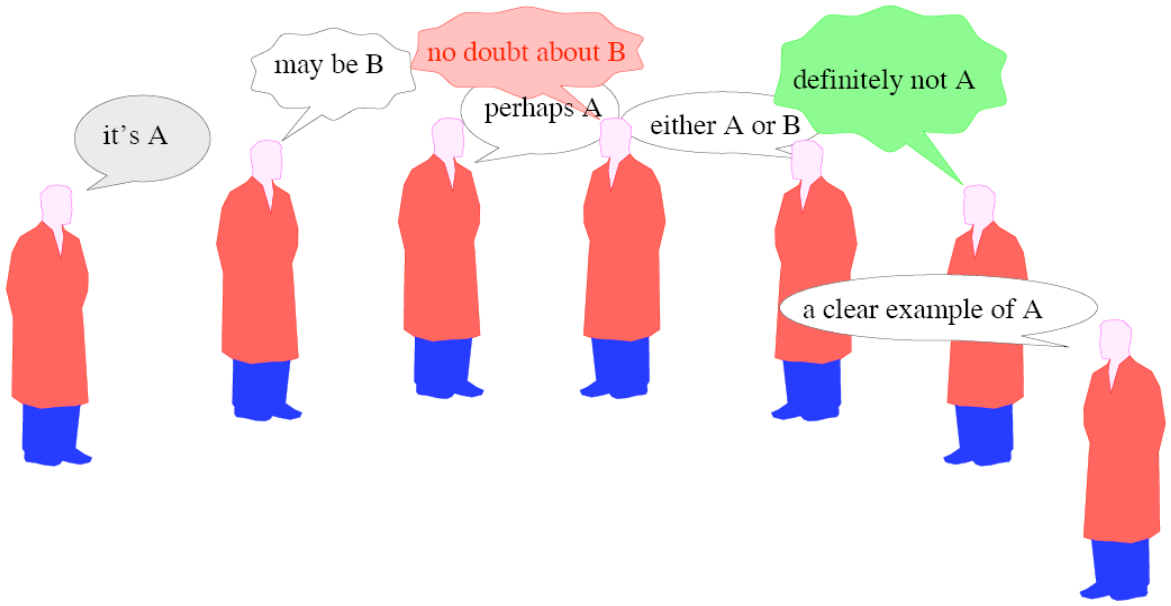
- simple and flexible classifier
- combination of discrete and continuous features
- feature selection (Day 3)
- interpretability

- hard splits
- splits are axis-aligned
- sensitive to small variations in data (high variance, Day 5)

BioSB

Classifier combination

- Idea: combine different classifiers and have them vote
- Design choices:
  - Identical or different?
    - Base classifiers, feature spaces, training sets, initialisations, etc.
  - Combination by a fixed rule or by another classifier?



BioSB

Example: random forests

- General overview: Day 5
- Specific example: random forest – an ensemble of decision trees

- Choices to be made:
  - Base classifiers: identical – decision trees
  - Feature spaces: for each node in each tree sample randomly *m* features
    - *m* << total number of features
  - Training sets: sampling with replacement (bootstrapping)
    - About two-third of the cases are used for training each tree
- Combination: majority vote

BioSB

Characteristics

- Out-of-bag error (oob) estimate:
  - Each tree can be tested on about one-third of the cases – the out-of-bag samples

- Variable importance:
  - For each tree: predict the class for oob cases and count the number of votes cast for the correct class
  - For each tree: randomly permute the values of variable *n* in the oob cases and count the number of votes cast for the correct class
  - Importance: rank (from high to low) based on average difference of these two scores

BioSB

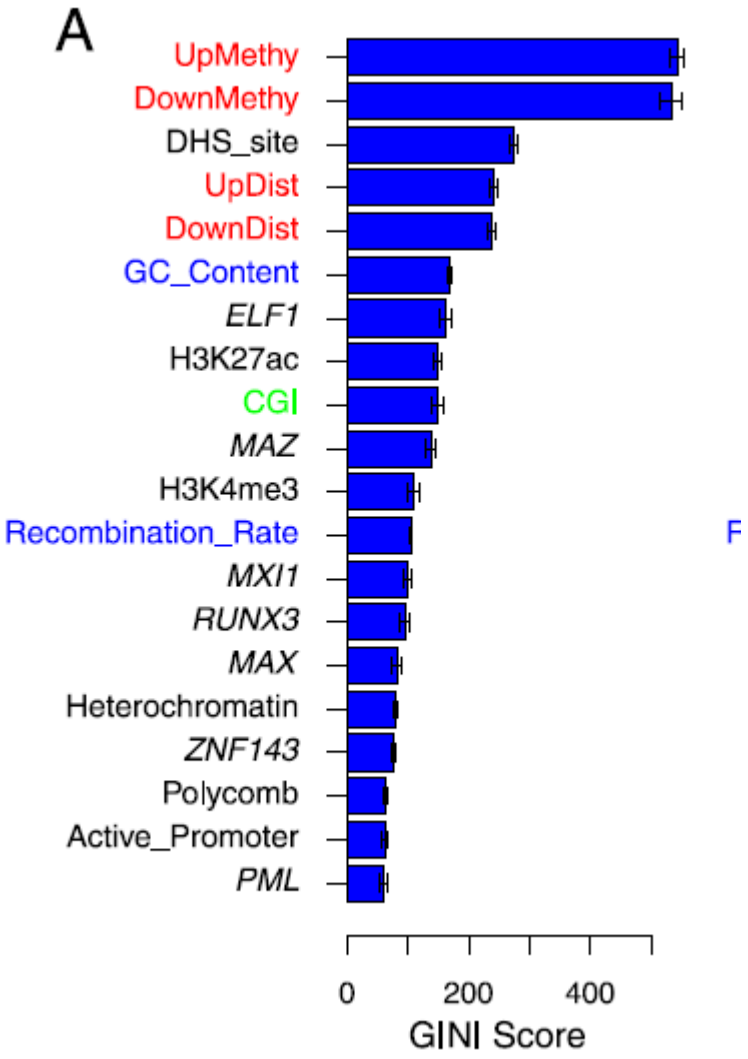
Some intuition

- Breiman *et al.*, Machine Learning (2001) paper
- Accuracy depends on two factors:
  - Correlation between any two trees in the forest. Decreasing correlation increases the forest accuracy: diversity
  - Accuracy of each individual tree (strength) in the forest. Increasing strength of individual trees increases the forest accuracy
- Trade-off:
  - Reducing *m* reduces correlation and strength
  - Increasing *m* increases correlation and strength
- Solution: somewhere in between is an *optimal* range of *m* - usually quite wide. Using the oob error rate a value of *m* in the range can be found

BioSB

Random forests: example

- Prediction of genome-wide DNA methylation
- Features:
  - Neighbors
  - Genomic position
  - DNA sequence properties
  - Cis-regulatory elements
- Random forest: feature selection



Zhang *et al.*, Genome Biology (2015)

BioSB

Recapitulation

- Decision trees: simple and flexible classifier
  - Incorporates feature selection
  - Interpretable
  - Hard, axis-aligned splits
  - Pruning is essential to avoid overfitting
- Random forest: example of ensemble method
  - Ensemble of decision trees
  - Variation between members introduced via randomness
  - When number of features is large and percentage of truly informative features is small (gene expression-based diagnostics): performance tends to decline significantly

BioSB