

# Classification by Nearest Shrunken Centroids and Support Vector Machines

**Florian Markowetz**

florian.markowetz@molgen.mpg.de  
Max Planck Institute for Molecular Genetics,  
Computational Diagnostics Group, Berlin



**Practical DNA microarray analysis 2005**

# Two roads to classification

1. model **class probabilities**  
→ QDA, LDA, ...
2. model **class boundaries** directly  
→ Optimal Separating Hyperplanes, SVM



# What's the problem?

In classification you have to trade off **overfitting vs. underfitting** and **bias vs. variance**.

In 12'000 dimensions even linear methods are very complex → high variance!

## Simplify your models



# Discriminant analysis and gene selection



# Comparing Gaussian likelihoods

**Assumption:** each group of patients is well described by a Normal density.

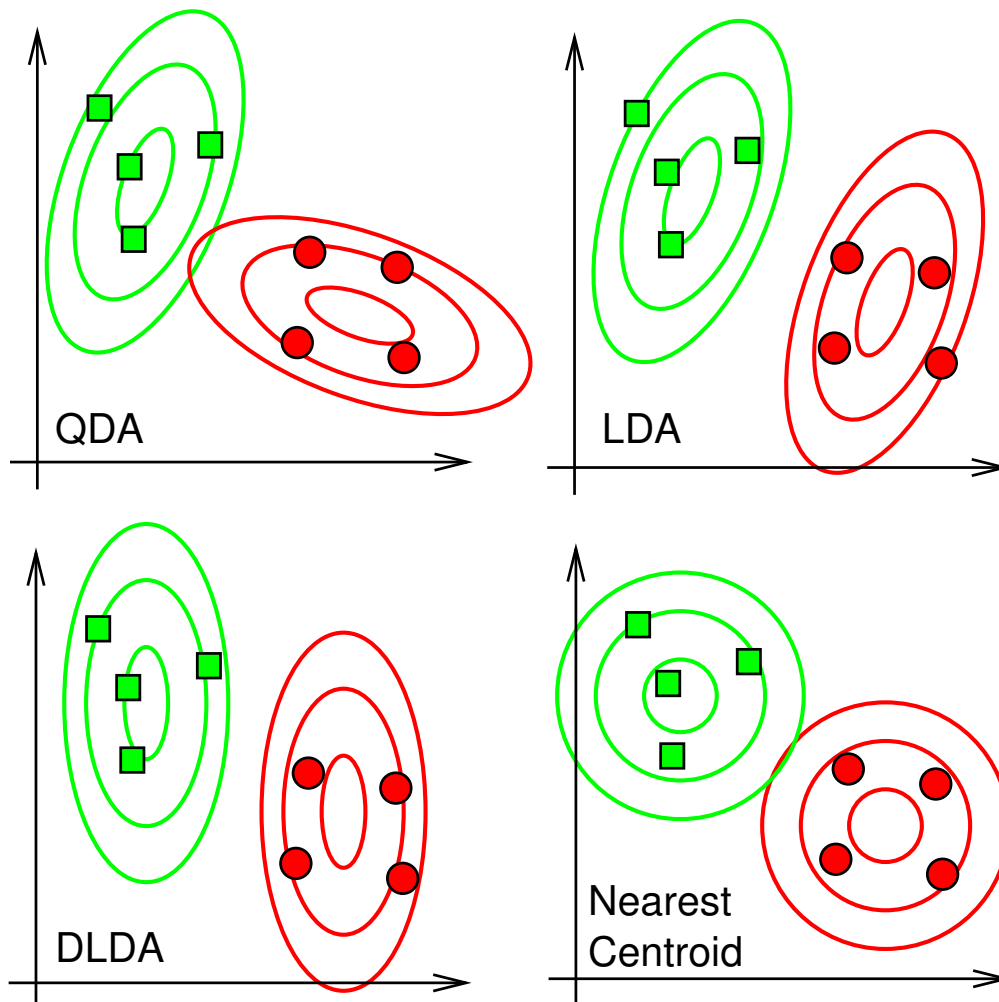
**Training:** estimate **mean** and **covariance matrix** for each group.

**Prediction:** assign new patient to group with higher likelihood.

**Constraints** on covariance structure lead to different forms of discriminant analysis.



# Discriminant analysis in a nutshell



Characterize each class by **mean** and **covariance structure**.

- **Quadratic D.A.**  
different COVs
- **Linear D.A.**  
requires same COVs.
- **Diagonal linear D.A.**  
same diagonal COVs.
- **Nearest centroids**  
forces COVs to  $\sigma^2 \mathbf{I}$ .



# Feature selection

## Next simplification:

Base the classification only on a small number of genes.

Feature selection: Find the **most discriminative genes**.

This task is different from testing for differential expression. Genes can be significantly differential expressed, but still useless for classification.



# Feature selection

## 1. Filter:

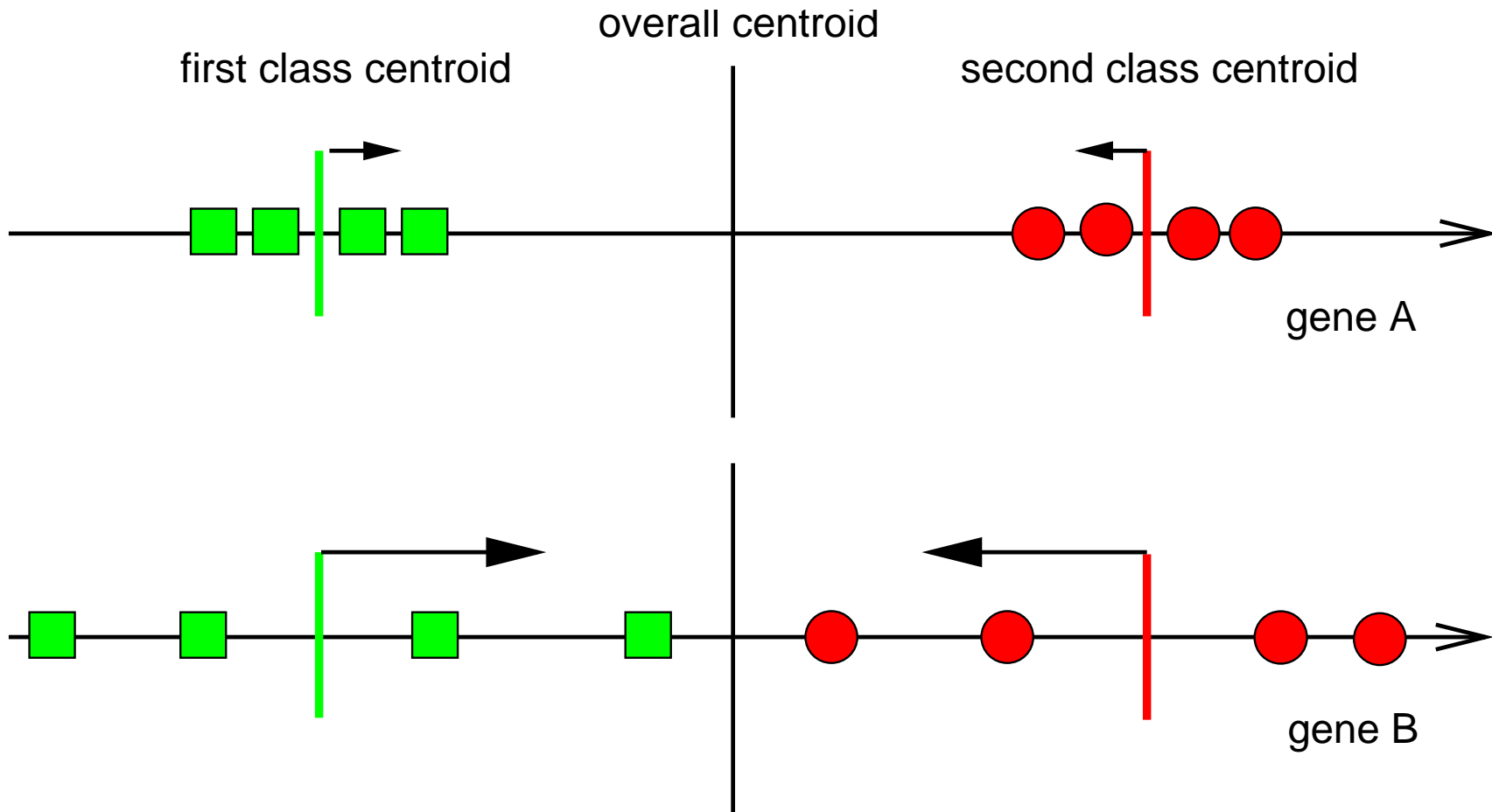
- Rank genes according to discriminative power by t-statistic, Wilcoxon, ...
- Use only the first  $k$  for classification.
- Discrete, hard thresholding.

## 2. Shrinkage:

- Continuously shrink genes until only a few have influence on classification.
- Example: [Nearest Shrunken Centroids](#).



# Shrunken Centroids



## Nearest Shrunken Centroids *cont'd*

The **group centroid**  $\bar{x}_{gk}$  for gene  $g$  and class  $k$  is compared to the **overall centroid**  $\bar{x}_g$  by

$$\bar{x}_{gk} = \bar{x}_g + m_k (s_g + s_0) d_{gk} ,$$

where  $s_g$  is the pooled within-class standard deviation of gene  $g$  and  $s_0$  is an offset to guard against genes with low expression levels.

**Shrinkage:** Each  $d_{gk}$  is reduced by  $\Delta$  in absolute value, until it reaches zero. Genes with  $d_{gk} = 0$  for all classes do not contribute to the classification.

(Tibshirani *et al.*, 2002)



# Shortcomings of filter and shrinkage methods

1. High **correlated genes** get similar score but offer no new information.  
But see (Jaeger *et al.*, 2003) for a cure.
2. Filter and Shrinkage work only on **single genes**.  
They don't find interactions between groups of genes.
3. Filter and Shrinkage methods are only **heuristics**.  
Search for *best subset* is infeasible for more than 30 genes.

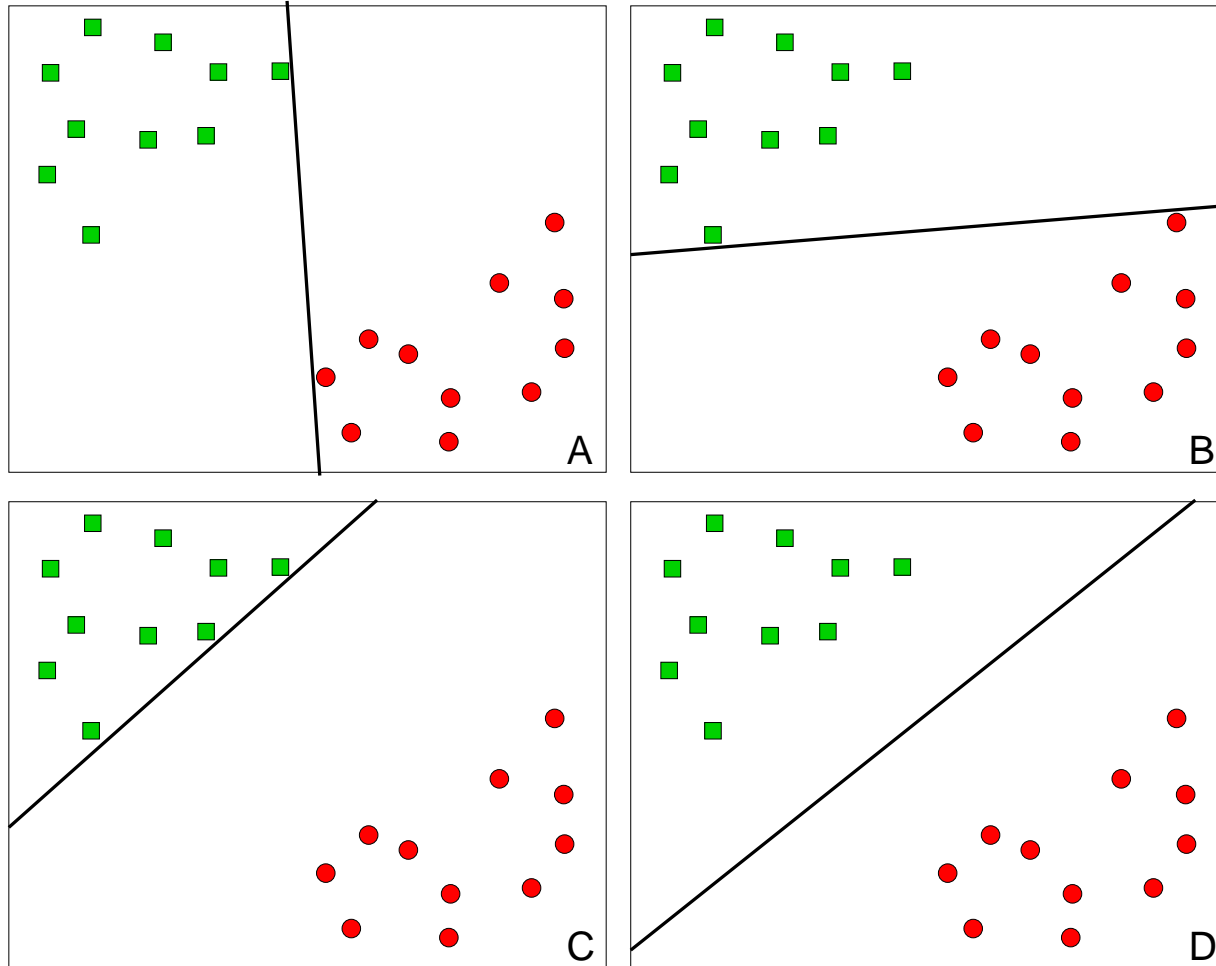


# Support Vector Machines

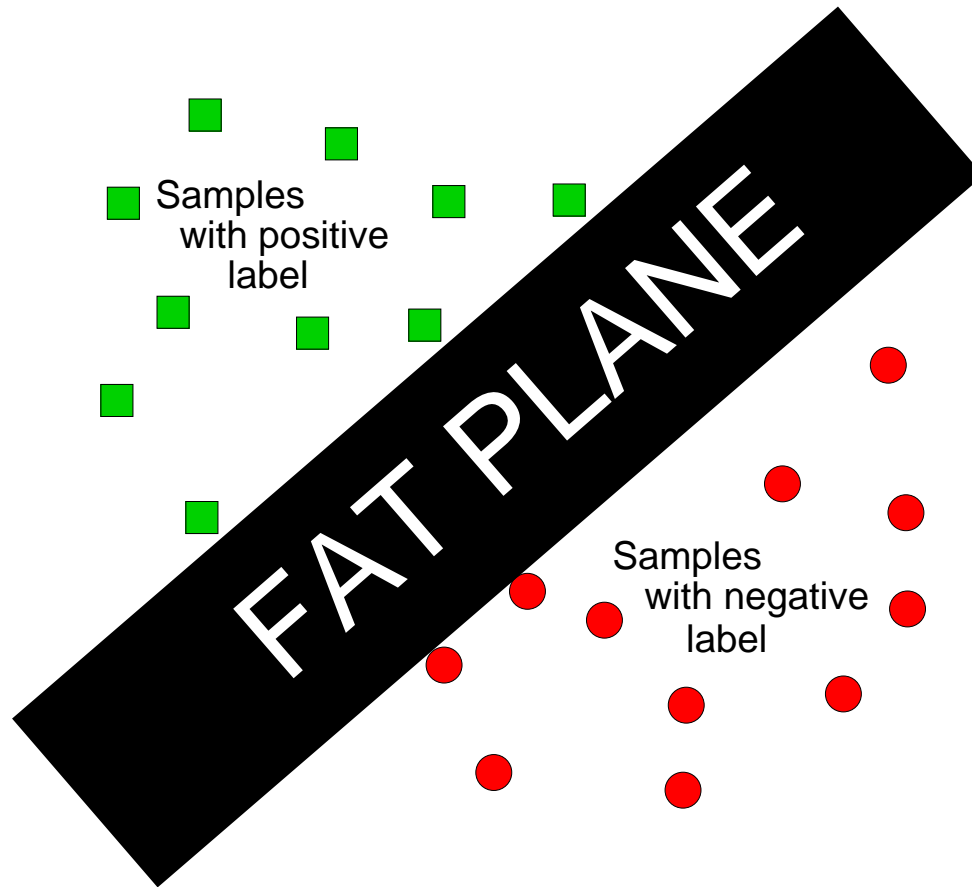
## — SVM —



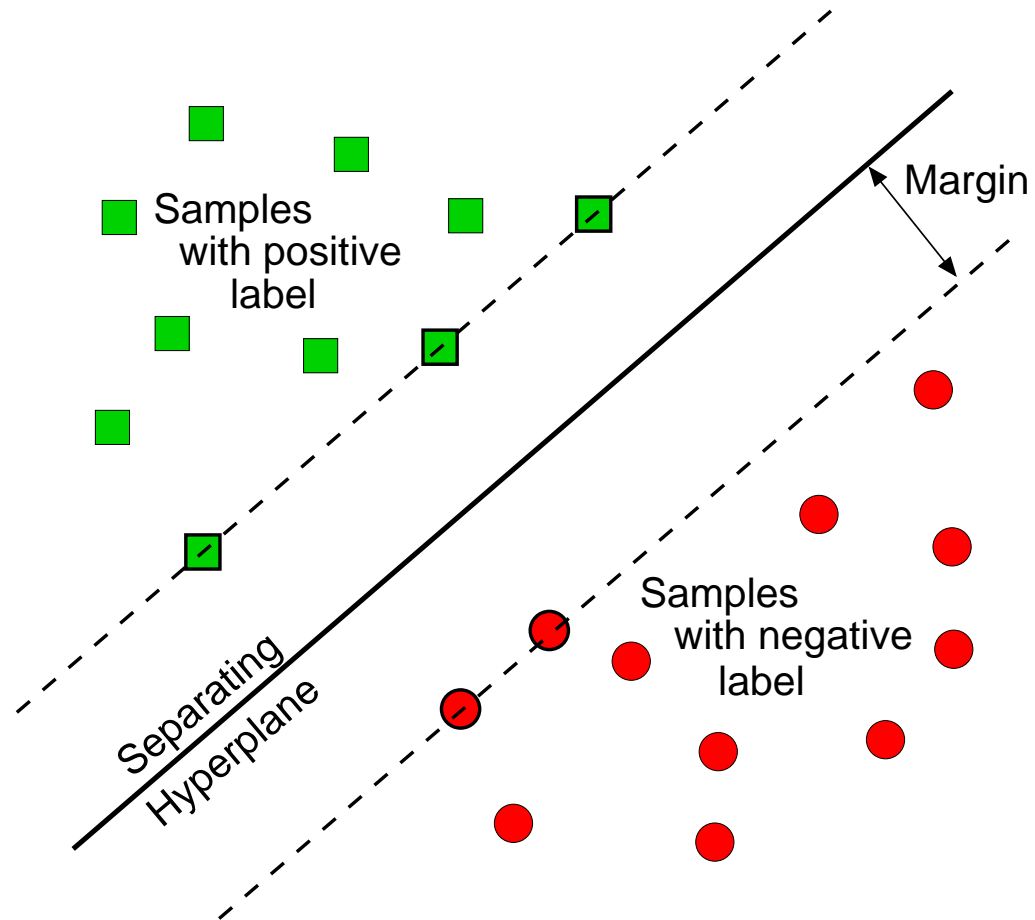
# Which hyperplane is the best?



# No sharp knife, but a fat plane



# Separate the training set with maximal margin

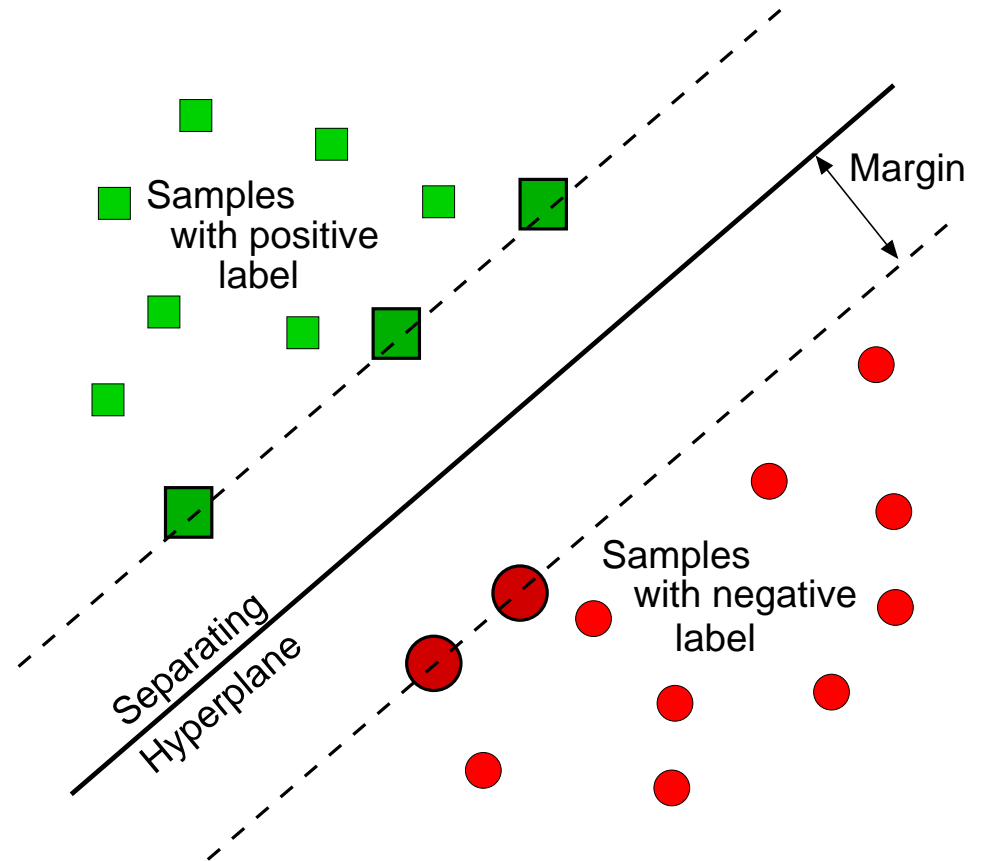


# What are Support Vectors?

The points nearest to the separating hyperplane are called **Support Vectors**.

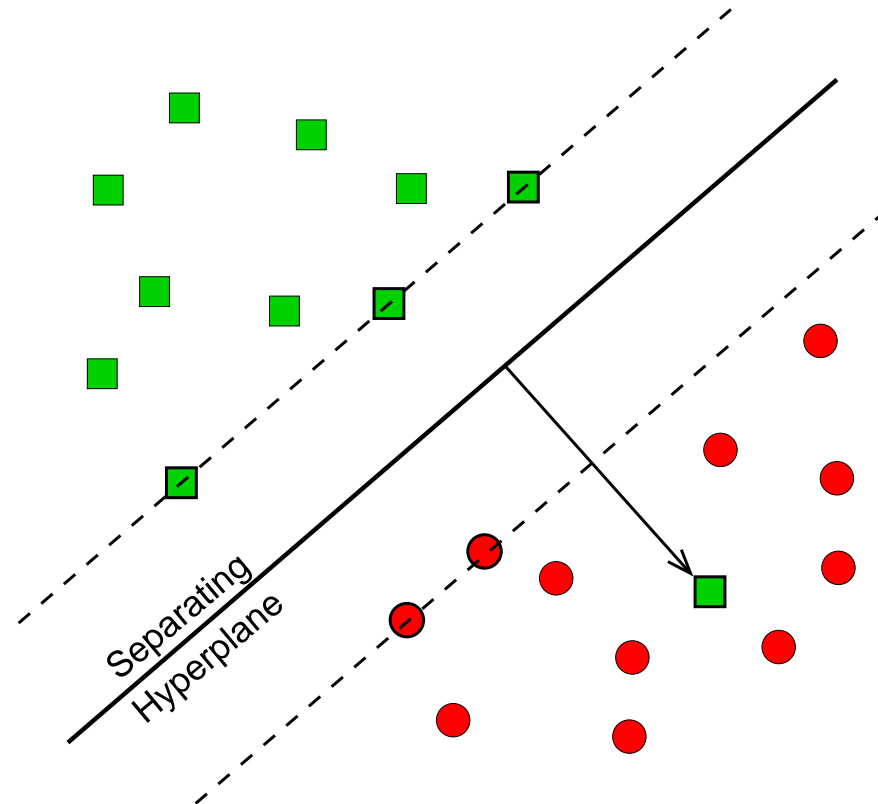
Only they determine the position of the hyperplane. **All other points have no influence!**

**Mathematically:** the weighted sum of the Support Vectors is the normal vector of the hyperplane.



# Non-separable training sets

Use linear separation, but admit training errors.



Penalty of error: distance to hyperplane multiplied by *error cost*  $C$ .



# The end?

The story of how to simplify your models is finished.

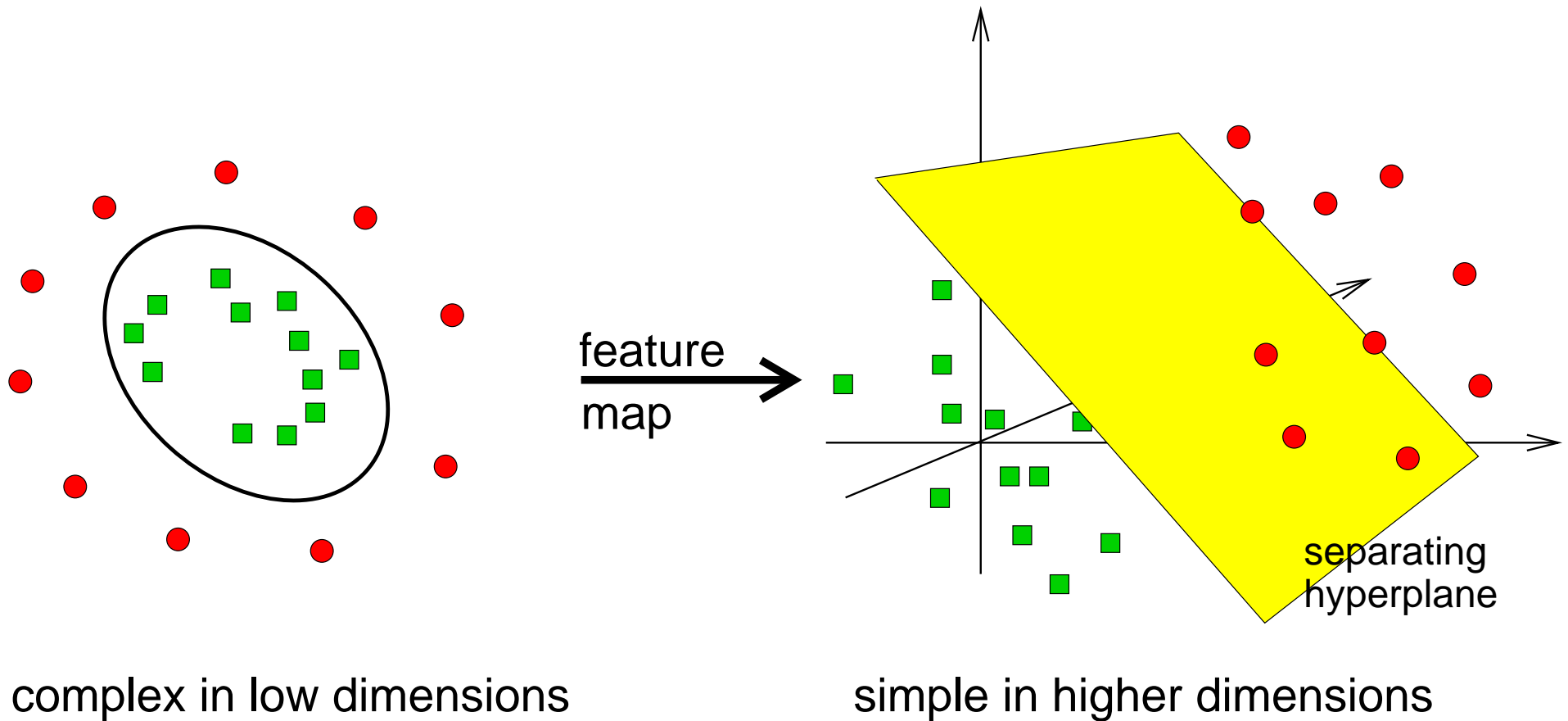
But for the sake of completeness:

How do we get from the simple linear Optimal Separating Hyperplane to a full-grown Support Vector Machine?

It's a trick, a **kernel trick**.



# Separation may be easier in higher dimensions



# The kernel trick

## Maximal margin hyperplanes in feature space

If classification is easier in a high-dimensional feature space, we would like to build a maximal margin hyperplane there.

The construction depends on inner products  $\Rightarrow$  we will have to evaluate inner products in the feature space.

This can be computationally intractable, if the dimensions become too large!

**Loophole** Use a kernel function that lives in low dimensions, but behaves like an inner product in high dimensions.



# Kernel functions

Expression profiles  $p = (p_1, p_2, \dots, p_g) \in \mathbb{R}^g$   
and  $q = (q_1, q_2, \dots, q_g) \in \mathbb{R}^g$ .

## Similarity in gene space: INNER PRODUCT

$$\langle p, q \rangle = p_1q_1 + p_2q_2 + \dots + p_gq_g$$

## Similarity in feature space: KERNEL FUNCTION

$$\mathcal{K}(p, q) = \text{polynomial, radial basis, ...}$$



# Examples of Kernels

**linear**  $\mathcal{K}(p, q) = \langle p, q \rangle$

**polynomial**  $\mathcal{K}(p, q) = (\gamma \langle p, q \rangle + c_0)^d$

**radial basis function**  $\mathcal{K}(p, q) = \exp(-\gamma \|p - q\|^2)$



# Why is it a trick?

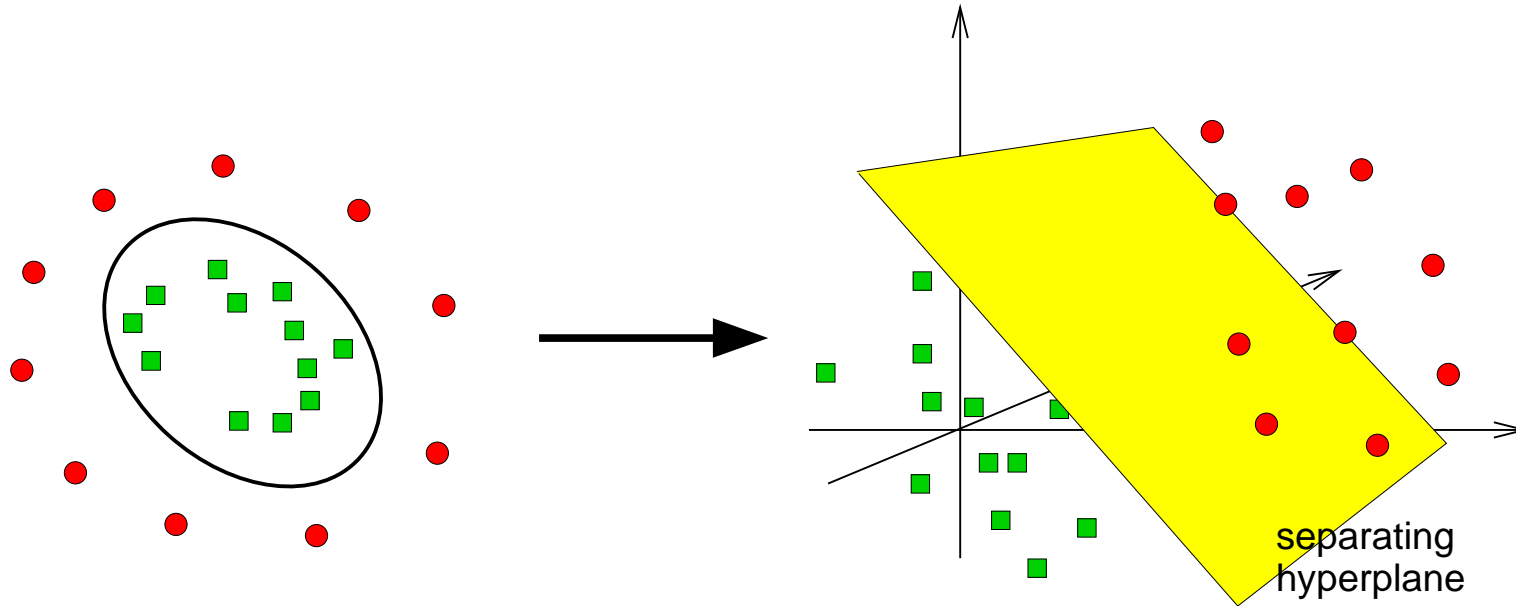
We do not need to know,  
how the feature space really looks like,  
we just need the kernel function as a measure of similarity.

This is kind of **black magic**: we do not know what happens inside the kernel, we just get the output.

Still, we have the **geometric interpretation** of the maximal margin hyperplane, so SVMs are more transparent than e. g. Artificial Neural Networks.



# The kernel trick: summary



Non-linear separation  
between vectors  
**in gene space**  
using kernel functions

=

Linear separation  
between vectors  
**in feature space**  
using inner product



# Support Vector Machines

A Support Vector Machine is  
a **maximal margin hyperplane** in feature space  
built by using a **kernel function** in gene space.



# Parameters of SVM

Kernel Parameters	$\gamma$ : width of rbf coeff. in polynomial ( $= 1$ )
	$d$ : degree of polynomial
	$c_0$ additive constant in polynomial ( $= 0$ )
Error weight	$C$ : influence of training errors



# SVM@work: low complexity

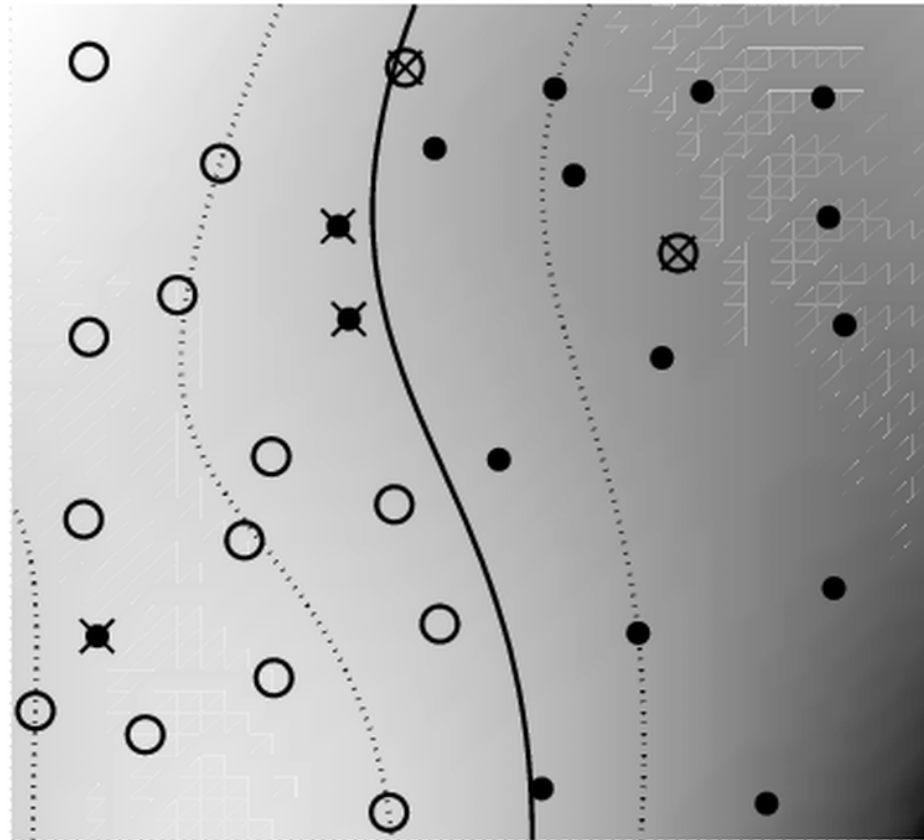


Figure taken from SCHÖLKOPF and SMOLA, *Learning with Kernels*, MIT Press 2002, p217



# SVM@work: medium complexity

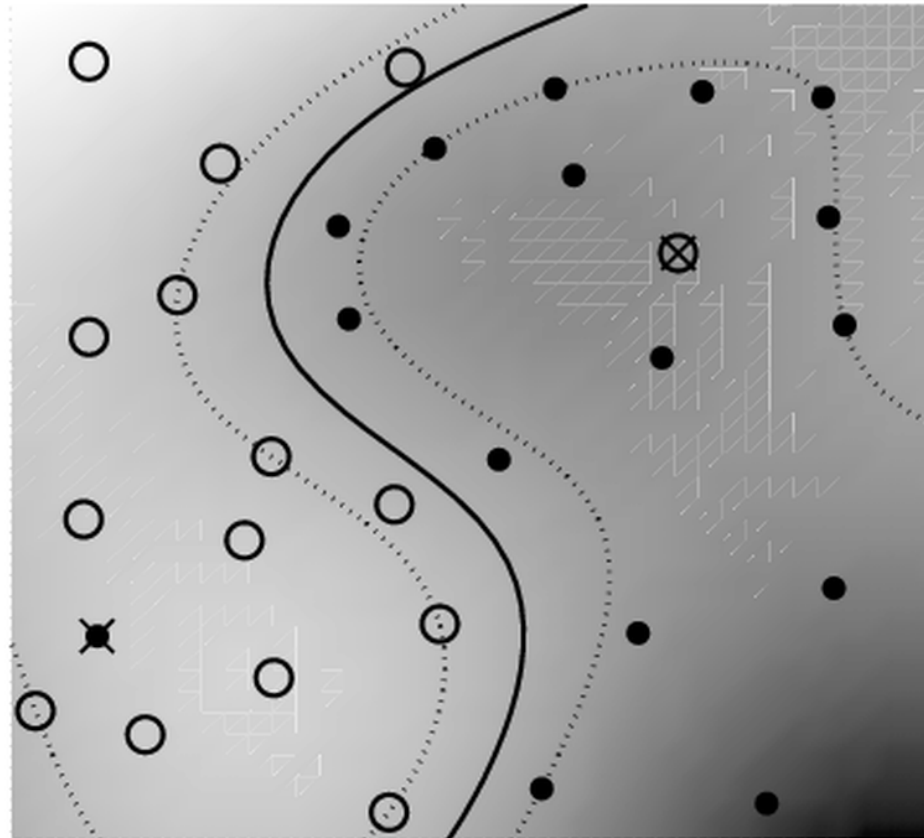


Figure taken from SCHÖLKOPF and SMOLA, *Learning with Kernels*, MIT Press 2002, p217



# SVM@work: high complexity

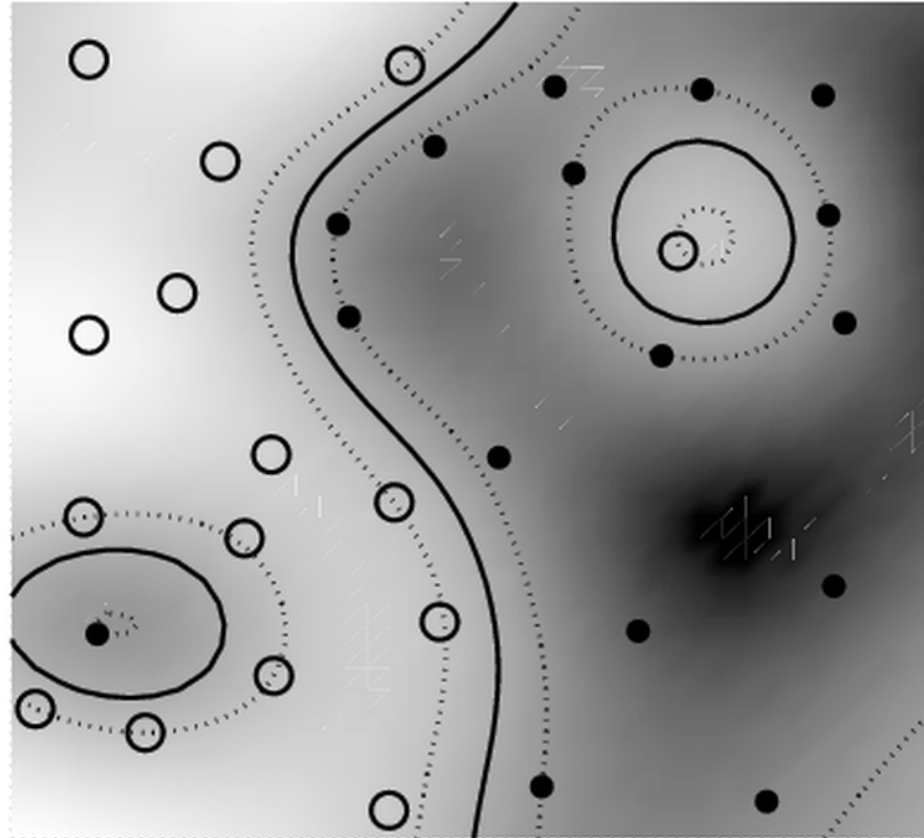


Figure taken from SCHÖLKOPF and SMOLA, *Learning with Kernels*, MIT Press 2002, p217



# References

1. Trevor Hastie, Robert Tibshirani, Jerome Friedman  
**The Elements of Statistical Learning**. Springer 2001.
2. Bernhard Schölkopf and Alex Smola.  
**Learning with Kernels**. MIT Press, Cambridge, MA, 2002.
3. Robert Tibshirani, Trevor Hastie, Balasubramanian Narasimhan, Gilbert Chu  
**Diagnosis of multiple cancer types by shrunken centroids of gene expression**, PNAS, 99(10), 6567–6572, 2002.
4. Jochen Jäger, R. Sengupta and W.L. Ruzzo  
**Improved Gene Selection for Classification of Microarrays**, Proc. PSB 2003



# Intro into practical session



# Computational Diagnosis

## TASK:

For 3 new patients in your hospital, decide whether they have a chromosomal translocation resulting in a BCR/ABL fusion gene or not.

## IDEA:

Learn the difference between the cancer types from an archive of 76 expression profiles, which were analyzed and classified by an expert.



# Training ... tuning ... testing

## TRAINING:

```
model <- svm(data = "76 profiles",  
             labels = "by an expert",  
             kernel = "..",  
             parameters = "..")
```

## TUNING:

```
svm.doctor <- tune.svm( data, labels,  
                       different.parameter.values )
```

## TESTING:

```
diagnosis <- predict(svm.doctor, new.patients)
```



## Training ... tuning ... testing

### TRAINING:

```
model      <- pamr.train( data , labels )
```

### TUNING:

```
pamr.cv( data, labels )
```

### TESTING:

```
diagnosis <- pamr.predict(new.patients, best.treshhold)
```