Support Vector Machines

SS 2016

Dr. Holger Fröhlich
Global Statistical Sciences
UCB Biosciences GmbH, Monheim

Motivation

- **Goal:** Investigate 2 (or more) groups of patterns and learn to discriminate them
  - Learn a classification rule (supervised learning)
- Support Vector Machines (SVMs) are one of the most commonly employed classification methods in Bioinformatics
- SVMs are often used for classification of high dimensional omics data (e.g. gene expression)
  - more variables than samples

Example Application: Personalized Medicine

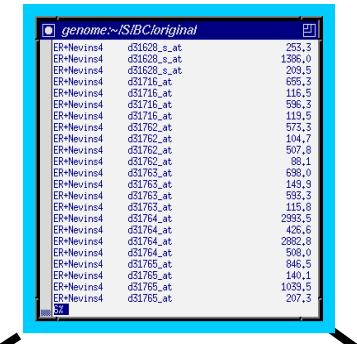
- Classical medicine: One drug and dosage for everybody

- Personalized medicine: individualized treatment for serious diseases
  - Cancer
  - Chronic diseases
  - ...

Example: Breast Cancer Treatment

- Breast cancer is not ONE disease
- There are (at least) 4 sub-types, which can be distinguished based on gene expression data.
- Sub-type should be considered in a personalized treatment.

Gene expression data



?

Tumor type A

?

Tumor type B

How our training data looks like

genes

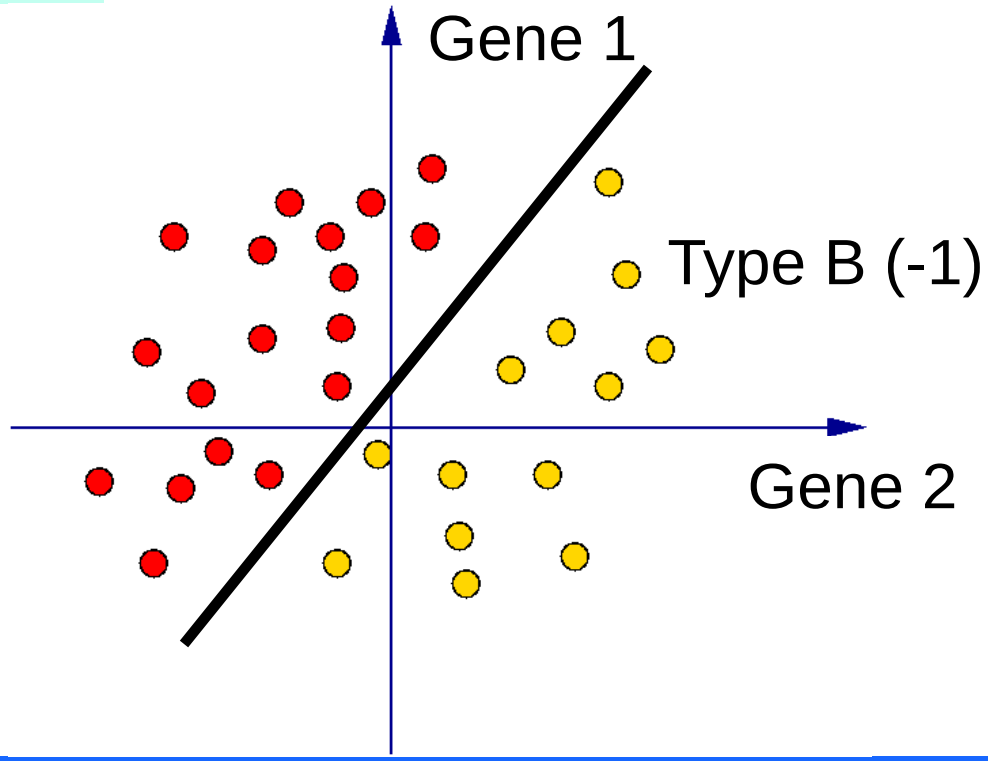
$$X = \begin{bmatrix} g_{11} & \cdots & g_{1m} \\ \vdots & \ddots & \vdots \\ g_{n1} & \cdots & g_{nm} \end{bmatrix}$$

patients

Typically 100 – 300 patients

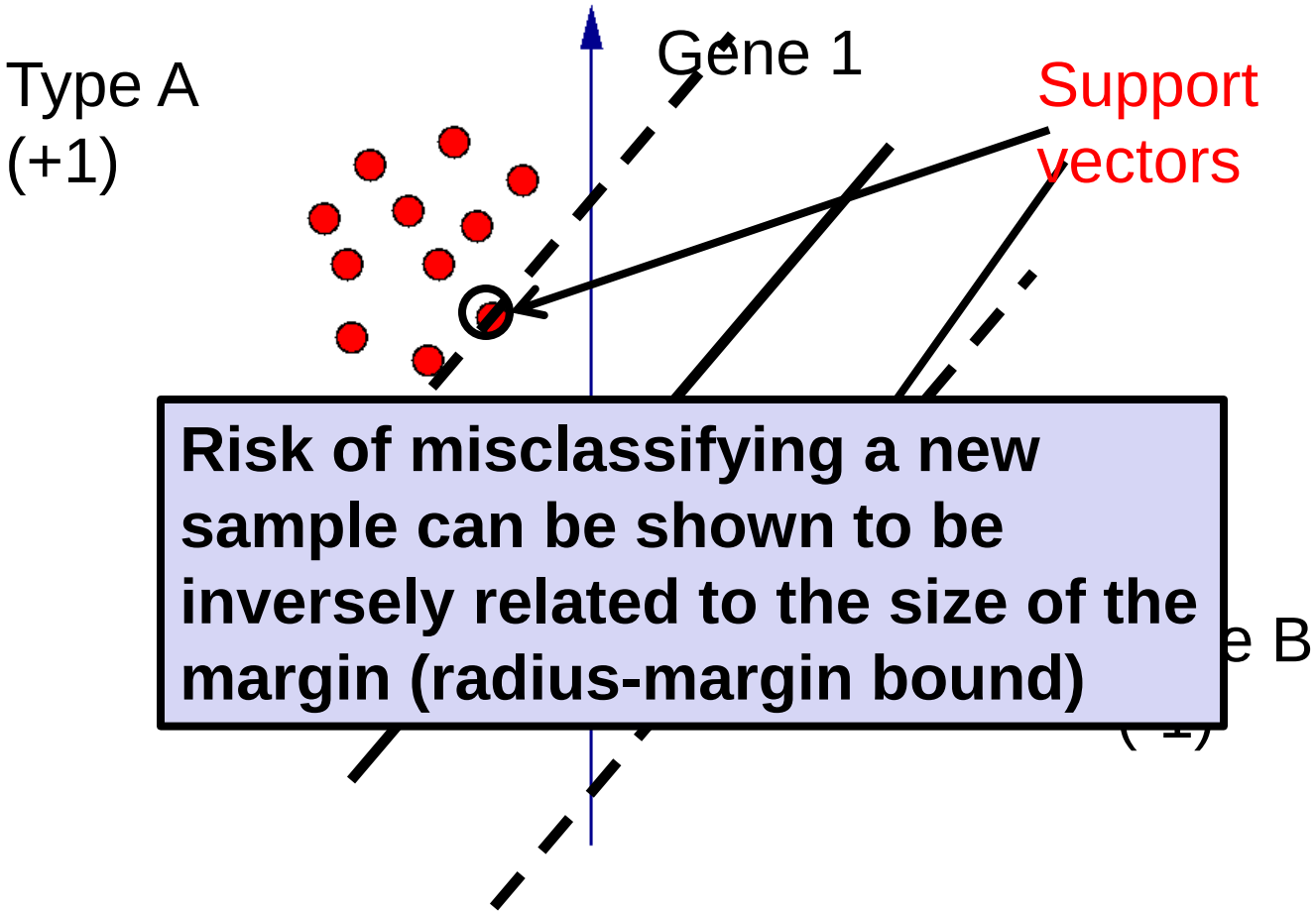
$$y = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} \in \{-1,1\}^n$$

Data with just 2 genes



Type A (+1)

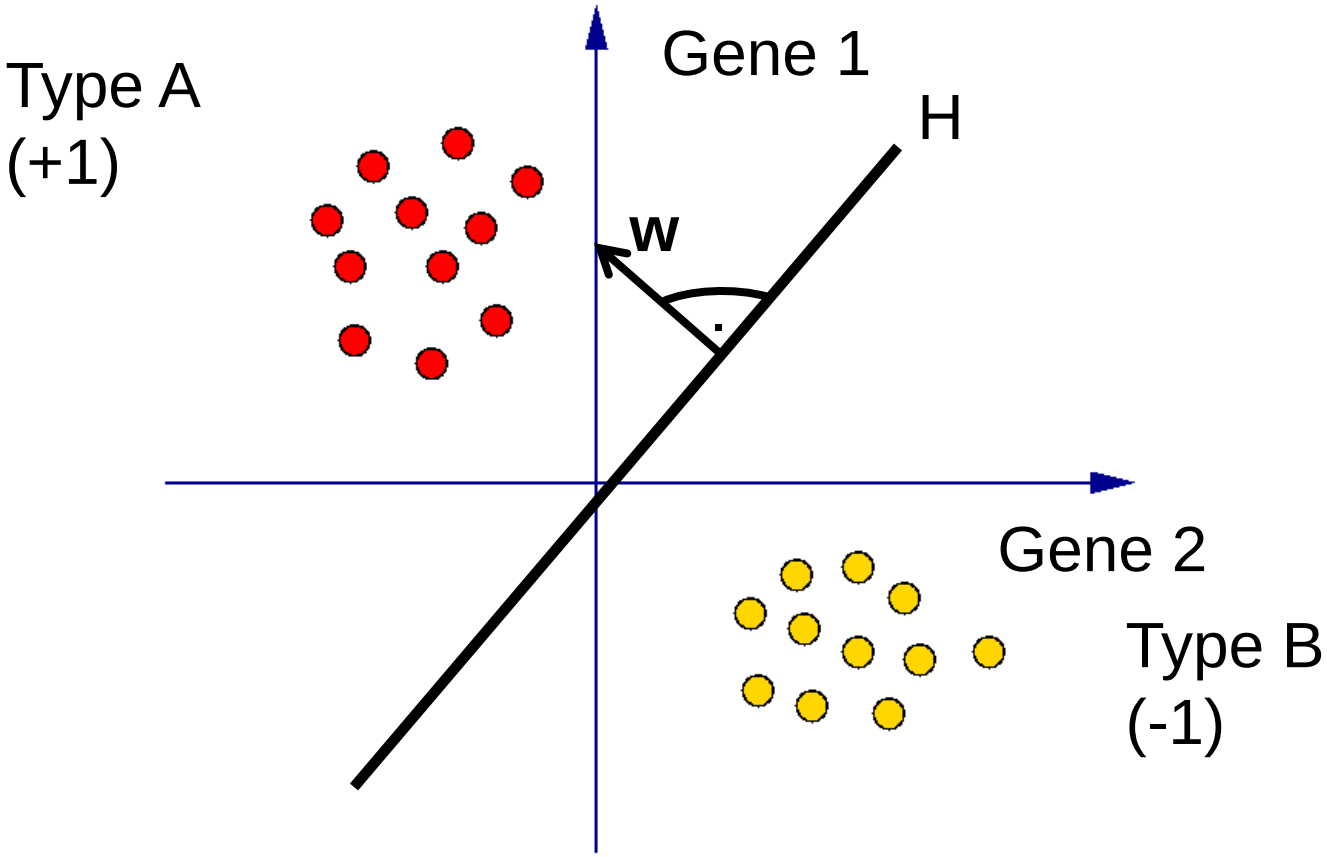
Support Vector Machines



Risk of misclassifying a new sample can be shown to be inversely related to the size of the margin (radius-margin bound)

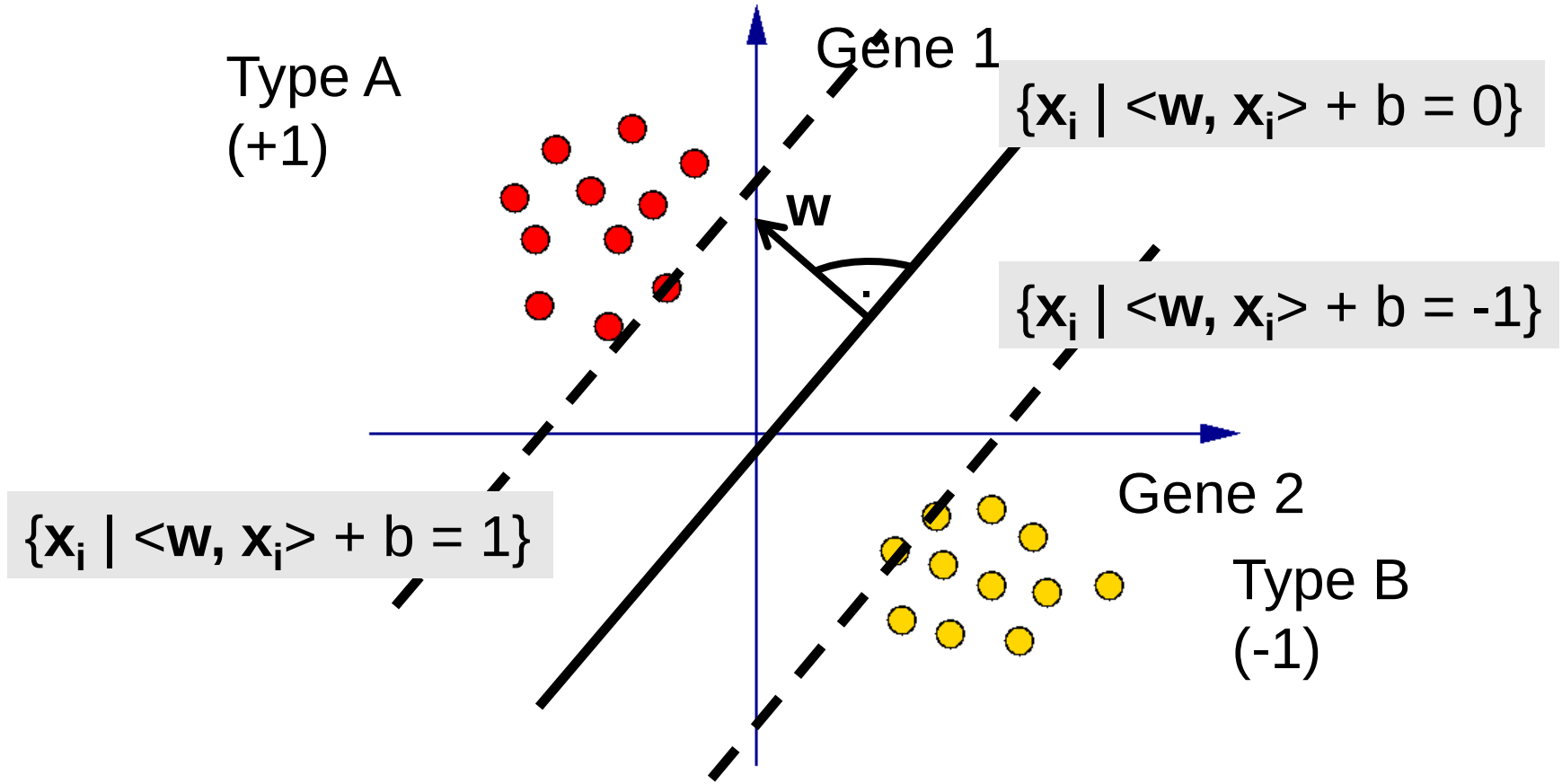
Choose the hyperplane that maximizes the margin between the two classes!

Hyperplanes



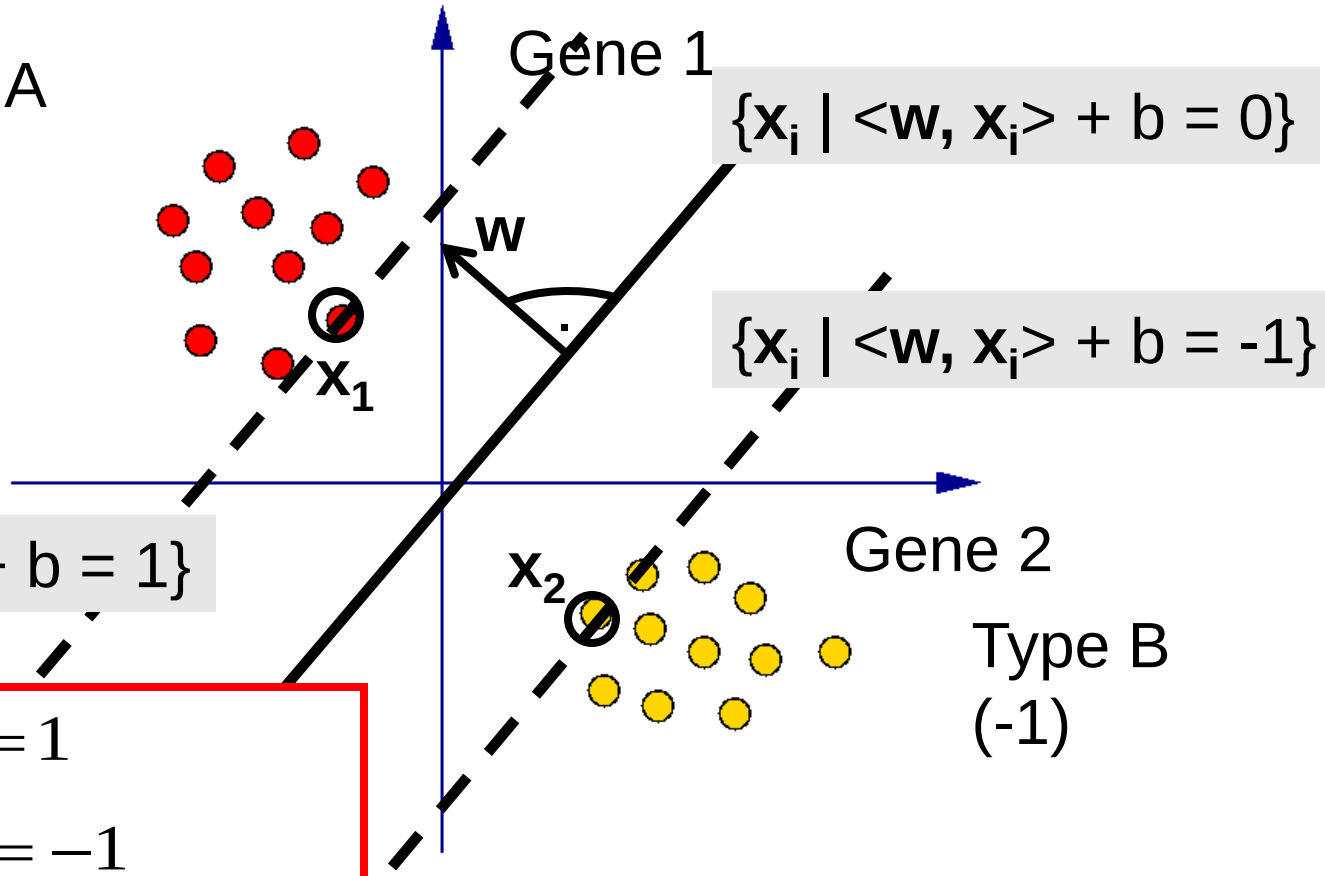
- Hyperplane can be described via the set of points being in the set $H = \{\mathbf{x}_i \mid \langle \mathbf{w}, \mathbf{x}_i \rangle + b = 0\}$
- $\mathbf{x}_i$ = i-th *row* of matrix X (expression profile for patient i)
- $\mathbf{w}$ = hyperplane normal vector

Rescaled hyperplane



- Hyperplane remains the same, if we multiply **w** and b by the same constant!
- <u>Consequence</u>: We can rescale **w** and b, such that the margin is normalized to 1

The size of the margin



$$\langle \mathbf{w}, \mathbf{x}_1 \rangle + b = 1$$

$$\langle \mathbf{w}, \mathbf{x}_2 \rangle + b = -1$$

$$\Rightarrow \langle \mathbf{w}, \mathbf{x}_1 - \mathbf{x}_2 \rangle = 2$$

$$\left\langle \frac{\mathbf{w}}{\| \mathbf{w} \|}, \mathbf{x}_1 - \mathbf{x}_2 \right\rangle = \frac{2}{\| \mathbf{w} \|}$$

← **maximize!**

Constraints

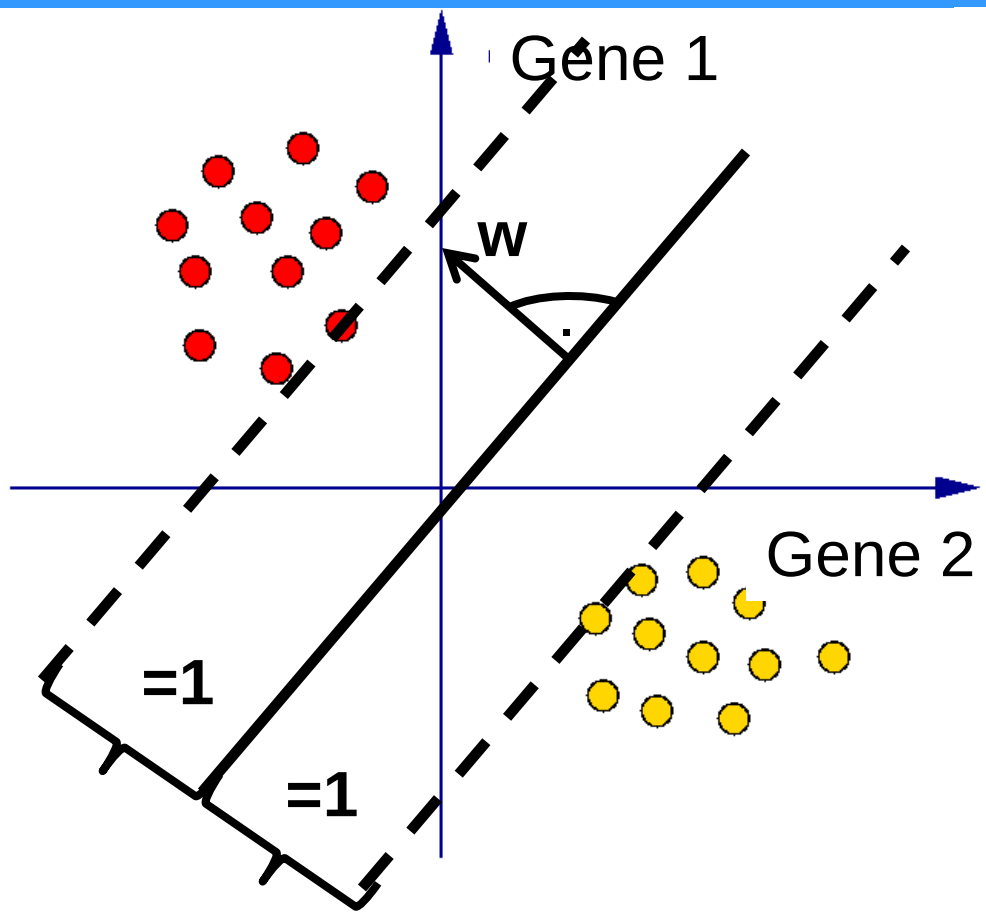
The hyperplane has to satisfy

$$\langle \mathbf{w}, \mathbf{x_i} \rangle + b \geq 1, \quad \text{if} \quad y_i = 1$$

$$\langle \mathbf{w}, \mathbf{x_i} \rangle + b \leq -1, \quad \text{if} \quad y_i = -1$$

which implies

$$y_i \left[\langle \mathbf{w}, \mathbf{x_i} \rangle + b \right] \geq 1$$



Computing the optimal hyperplane

- We have shown that we should *maximize* 2/||**w**||, i.e. *minimize* ||**w**||. Hence

$$\min_{\mathbf{w}} \frac{1}{2}\|\mathbf{w}\|^2$$

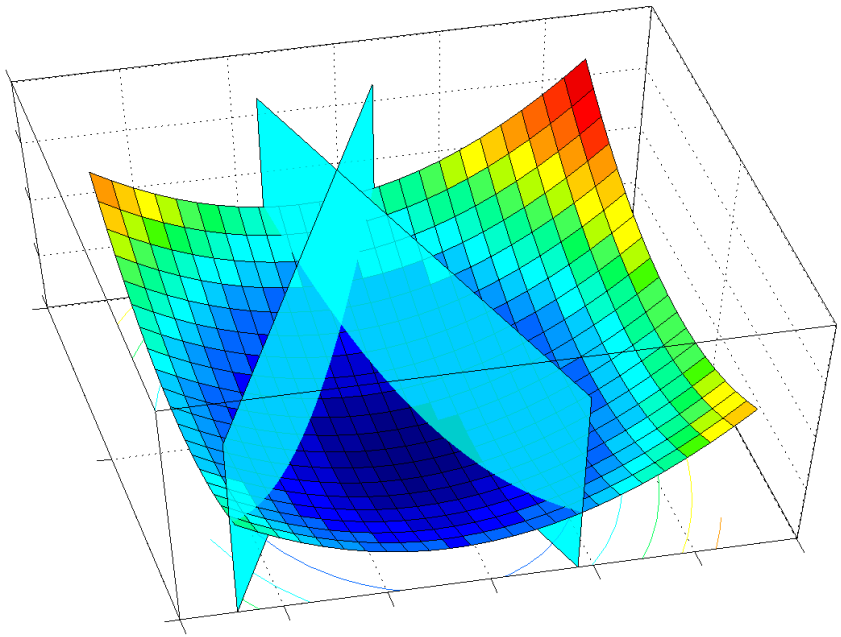
$$\text{subject to } y_i\left(\langle \mathbf{w}, \mathbf{x_i} \rangle + b\right) \geq 1 \quad (i = 1, \ldots, n)$$

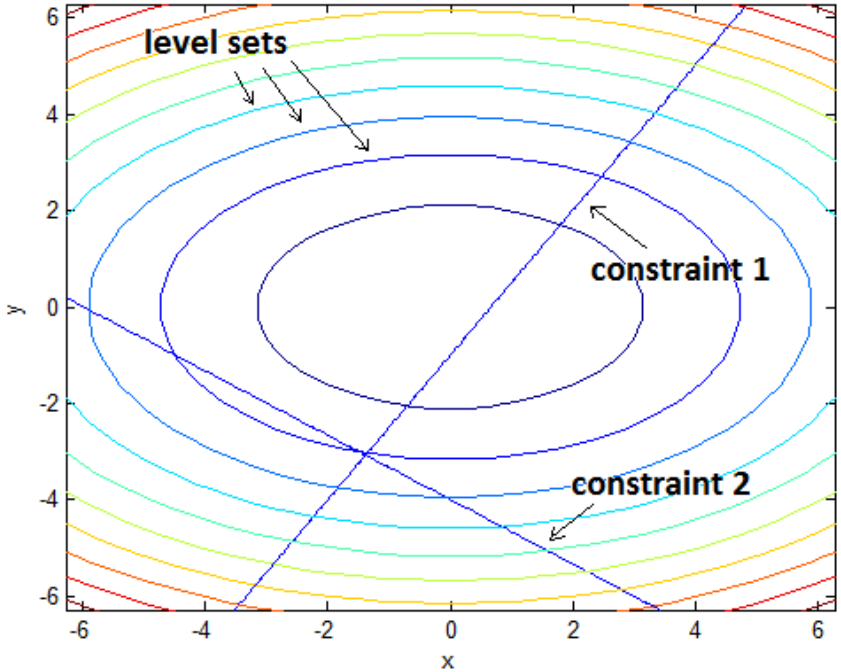
- So-called *quadratic program*
  - Objective function: $\boxed{\tfrac{1}{2}\,\|\mathbf{w}\|^2 = \tfrac{1}{2}\,\mathbf{w}^T\mathbf{w} = \tfrac{1}{2}\sum_i w_i^2}$
  - Constraints:

$$y_i\left[\langle \mathbf{w}, \mathbf{x_i} \rangle + b\right] \geq 1$$

- Objective function is convex (here: quadratic) function in **w**
- Constraints: linear in **w**
  - One constraint per data point ➔ n constraints in total

Quadratic Programming in 2D

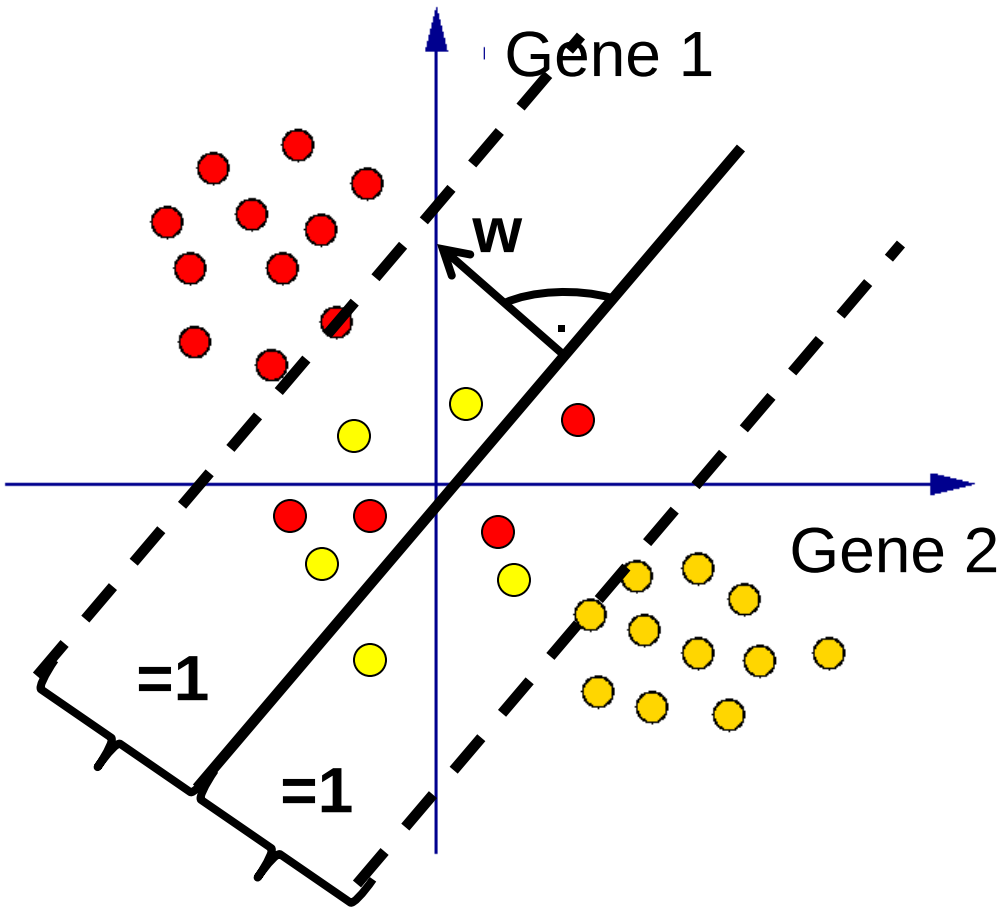




- **Quadratic programming** (QP) is a special type of mathematical optimization problem.
- It is the problem of optimizing (minimizing or maximizing) a quadratic function of several variables subject to linear constraints on these variables.
- For SVMs: unique, globally optimal solution
  - specialized solvers

Classification with noisy data

- What, if a hard margin hyperplane does not exist?
- Real data is noisy!



Soft Margin Hyperplanes (C-SVM)

- So far: no training errors allowed, perfect separation of the data
- Training errors might be acceptable ➔ otherwise no SVM solution or smaller margin
- Idea: Allow some violation of original constraints ➔ non-negative slack variables

$$\min_{\mathbf{w}} \frac{1}{2} \| \mathbf{w} \|^2 + C \sum_{i=1}^{n} \xi_i$$

$$\text{subject to } y_i \left(\langle \mathbf{w}, \mathbf{x} \rangle + b \right) \geq 1 - \xi_i \text{ for } i = 1, \ldots, n$$

- All non-zero slack variables correspond to *margin errors*.

- Parameter *C* trades margin errors against larger margin

SVMs as Regularized Models

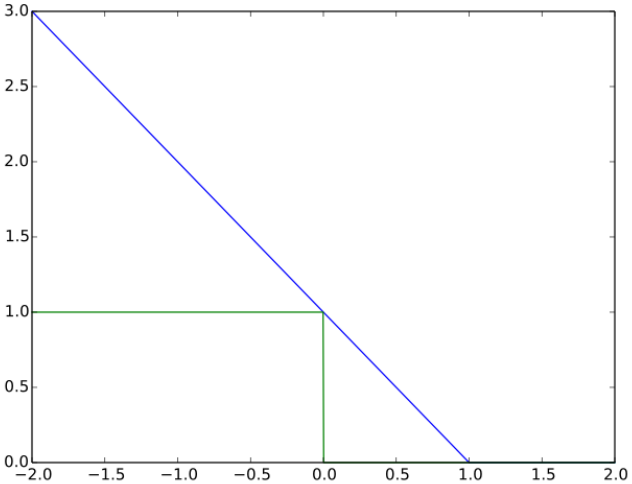
- We can re-write the SVM optimization problem as

$$\min_{\mathbf{w}} \frac{1}{2} \| \mathbf{w} \|^2 + C \sum_{i=1}^{n} \max(0, 1 - y_i (\langle \mathbf{w}, \mathbf{x} \rangle + b))$$

$$\Leftrightarrow \min_{\mathbf{w}} \sum_{i=1}^{n} \max(0, 1 - y_i (\langle \mathbf{w}, \mathbf{x} \rangle + b)) + \frac{1}{2C} \| \mathbf{w} \|^2$$

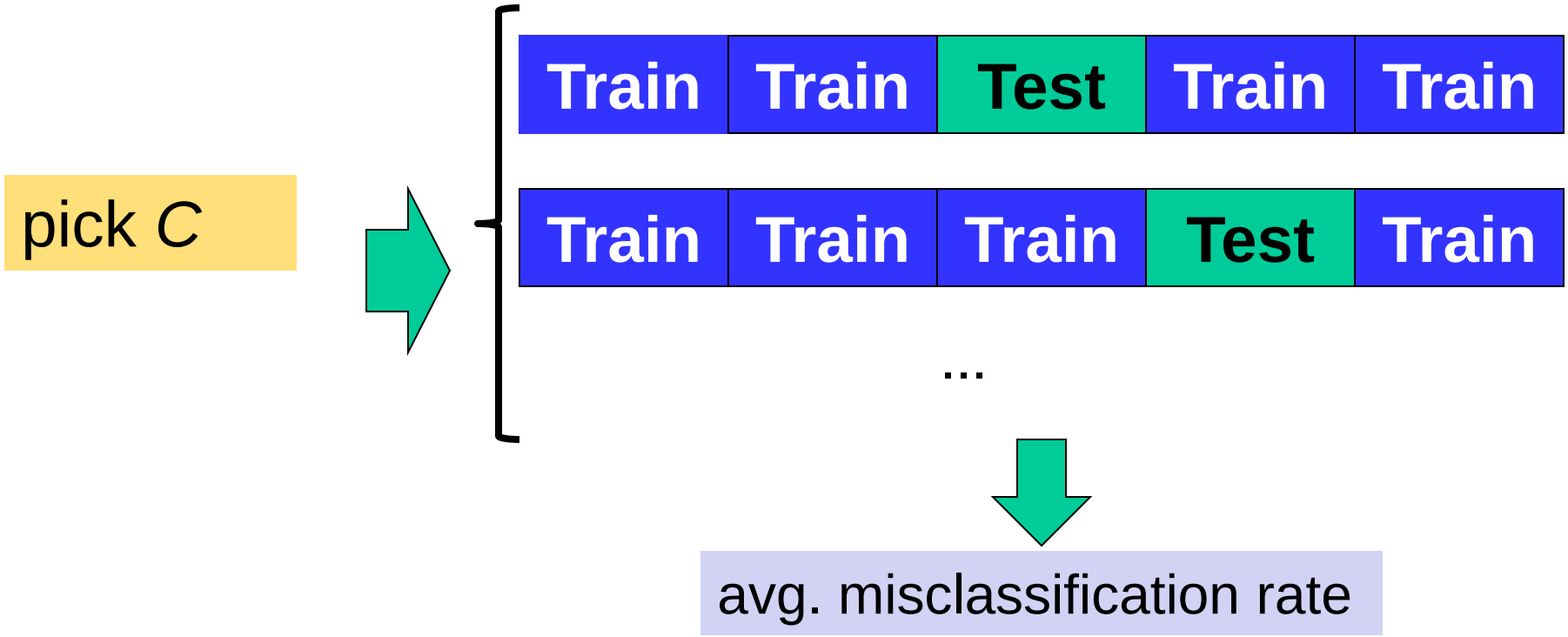
- This is exactly the general form of a regularized model with **hinge** loss function

$$\ell(y, y') = \max(0, 1 - yy')$$



- Standard SVM is <u>not</u> sparse in terms of parameters **w**
  - Most parameters will be non-zero (different to LASSO)
  - Classification depends on all variables

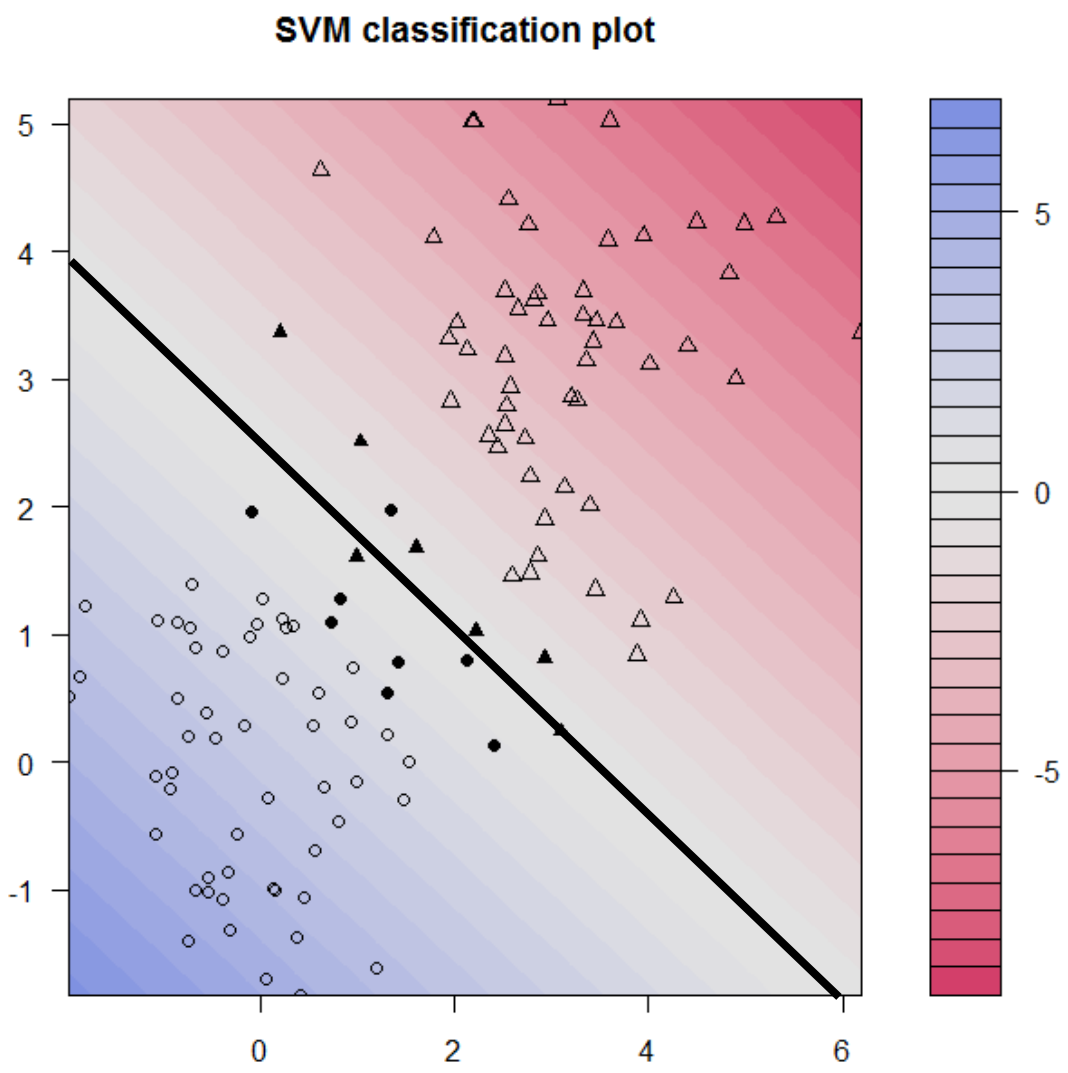
How to tune the *C*-parameter: cross-validation



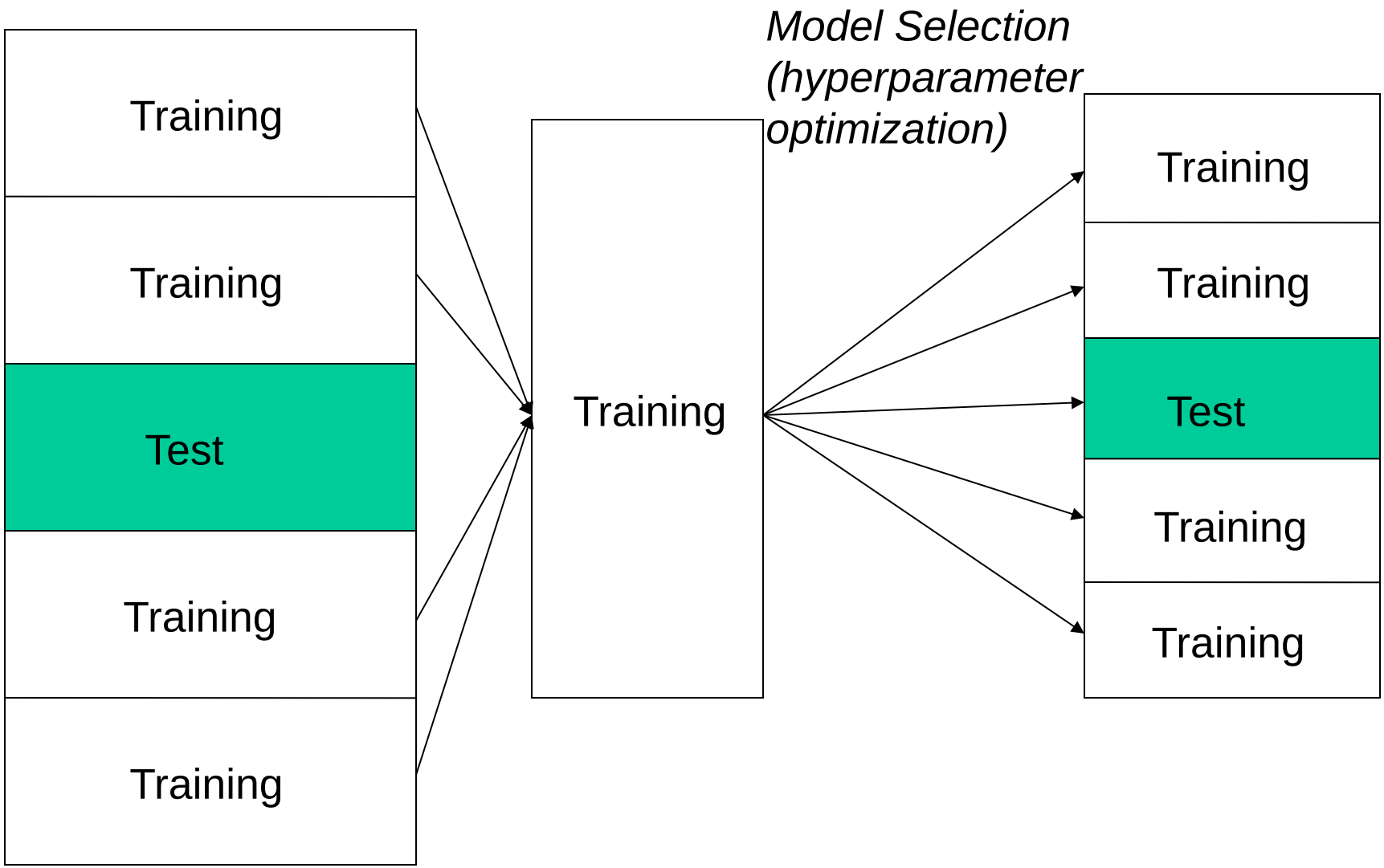
- Idea: sequentially split training data itself into training and test
- Evaluate predictions on (internal) test data.
- Only training data is used for learning the SVM model

Example

C	CV-error
0.01	2.5%
0.1	1.7%
1	3.3%
10	3.3%
100	3.3%

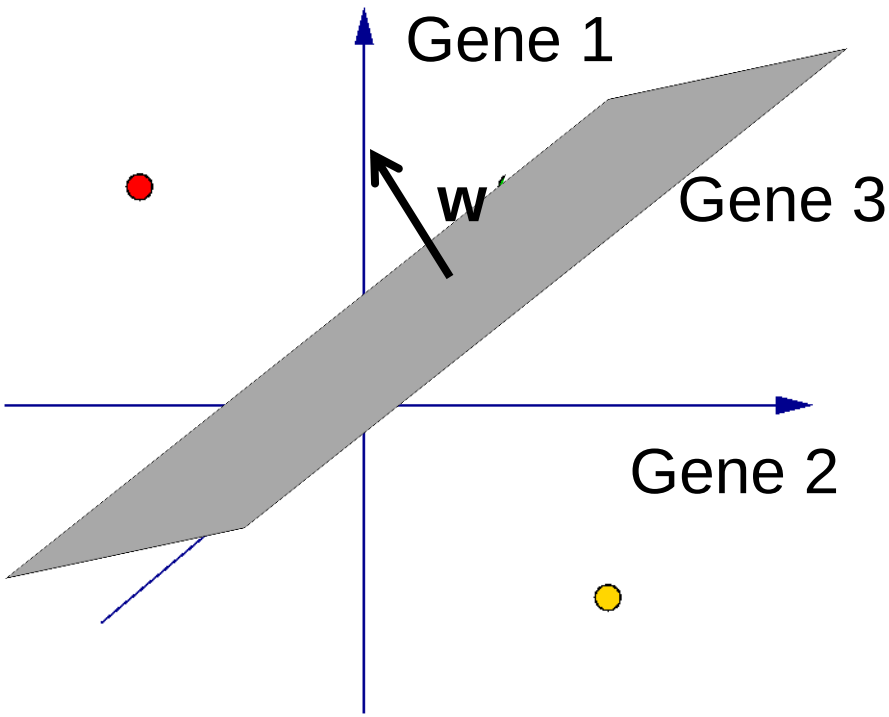


Nested Cross-Validation: Evaluation of classifier performance



SVMs for gene expression data

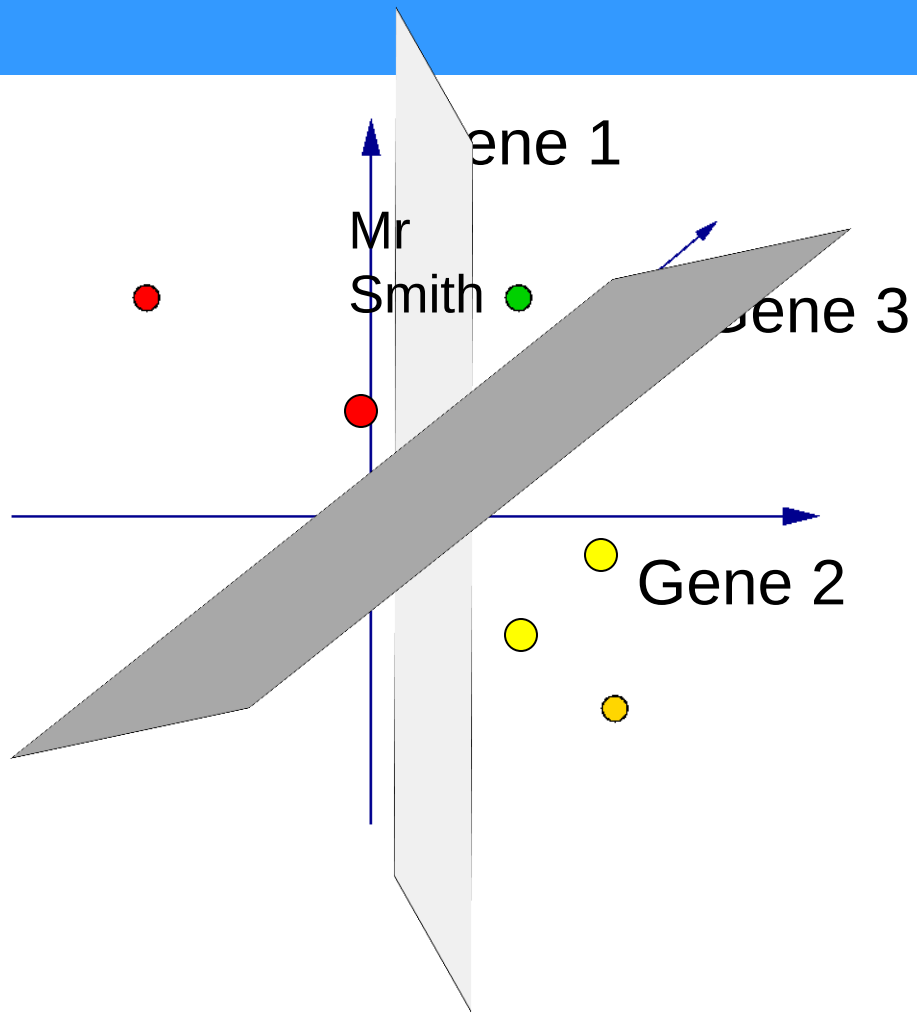
- SVMs are frequently employed for patient classification based on gene expression and other molecular data
- Challenge: This data is really high dimensional
  - Many more genes (~25,000) than patients (100 - 300)
  - Data points are extremely sparse within this space



With more and more gens (i.e. axis) training data points can be always separated perfectly.

SVMs for gene expression data

- The risk of misclassifying Mr Smith increases the more sparsity of our data we have
  - Adding more training data might change the SVM hyperplane drastically
  - Low training error, high prediction error (overfitting)



This has little to do with medicine. It is a geometrical problem.

Consequences

- Finding a good classification of patients on the training data is meaningless
- We need to find a combination of genes (signature) yielding high prediction performance!

- This is an instance of the so-called feature selection problem in machine learning.
  - Computationally difficult problem (NP complete)
  - Large body of literature, dominating approach
  - Other possibility: principal component analysis
    - Lower computational complexity
    - Disadvantages:
      - How many principal axis to choose?
      - Results are more difficult to interpret.

Feature Selection Approaches

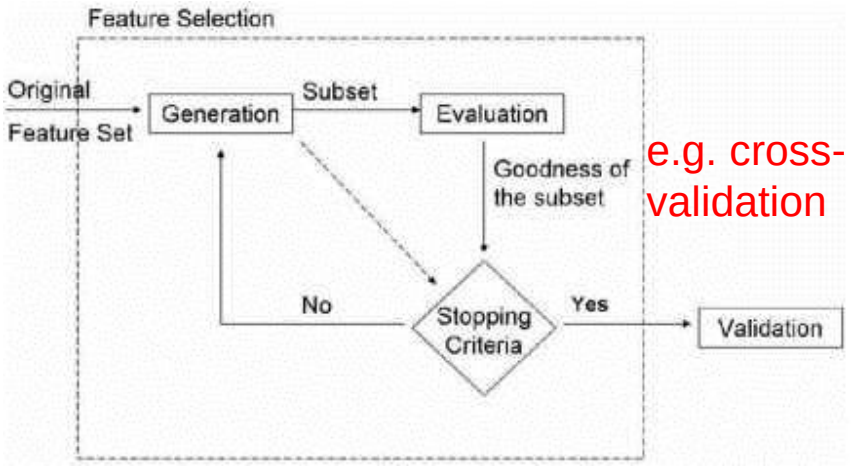
1. **Filter** methods
   - Define quality criterion (filter) for features or feature groups
   - Select features passing the filter
   - Train classifier

   - Pro: fast, well suited for very high dimensional data
   - Con:
     - success depends heavily on defined filter criterion: features are selected regardless of actual classifier performance
     - Difficult to evaluate feature groups: most often features are seen independent (bias!)

2. **Embedded** methods
   - Integrate feature selection into classification model
     - Example: LASSO: force coefficients for irrelevant features to be exactly 0 ➔ no contribution to model predictions
     - SVM does not have this property

Feature Selection Approaches (cont.)

3. **Wrapper** methods
   - Pro: Guided by classifier performance
   - Con: Highly computational demanding



e.g. cross-validation

- Example: *Greedy Forward Selection*
  1. Start with empty feature set
  2. Add feature improving classifier performance most
  3. Iterate, until classifier performance begins to decline

- 1st iteration: n features to evaluate
- 2nd iteration: n – 1 features
- 3rd iteration: n – 2 features
- ...
- **Note: improvement by given feature depends on already selected ones!**

→ $O(n^2)$ feature sets to evaluate!
➔ Disaster for high dimensional data

Example for Filter: Feature Selection via T-Test and Fold Change

- One particularly simple method for feature selection is to filter genes showing differential expression between the two patient groups.

genes

$$X = \begin{bmatrix} g_{11} & \cdots & g_{1m} \\ \vdots & \ddots & \vdots \\ g_{n1} & \cdots & g_{nm} \end{bmatrix}$$

group A

group B

T-test (p)	0.1	0.04	...	0.5
Log2 FC	-1.2	0.3	...	0.1

Note: a log-FC of 1 means a two-fold up-regulation, a log-FC of -1 a two-fold down-regulation

- Remember: The p-value is the probability to get a test statistic at least as extreme as the one observed under the null hypothesis.
- If we select a gene with p < 0.05, there is a 5% chance of a false positive
- With 10,000 genes we would thus expect 10,000 * 0.05 = 500 false positive genes!
- We need to lower the p-value cutoff to adjust for multiple testing!

Controlling False Discovery Rate (FDR)

- **Idea:** adjust p-value such that the expected proportion of falsely rejected null hypotheses is q% (e.g. 5%)

- <u>Benjamini-Hochberg method:</u>

1. Sort p-values in increasing order: $p_1, ..., p_m$
2. For a given q find the largest k such that

$$p_k \leq \frac{k}{m} q$$

3. Declare all $p_1, ..., p_k$ as significant

Summary: Classifier Development for Biomarker Signature Discovery

Gene selection

- T-test for each gene ➔ p-values ➔ FDR
- Select DE genes:
  - *e.g: FDR < 5% and |log FC| > 1 (2 fold up or down)*

SVM Training

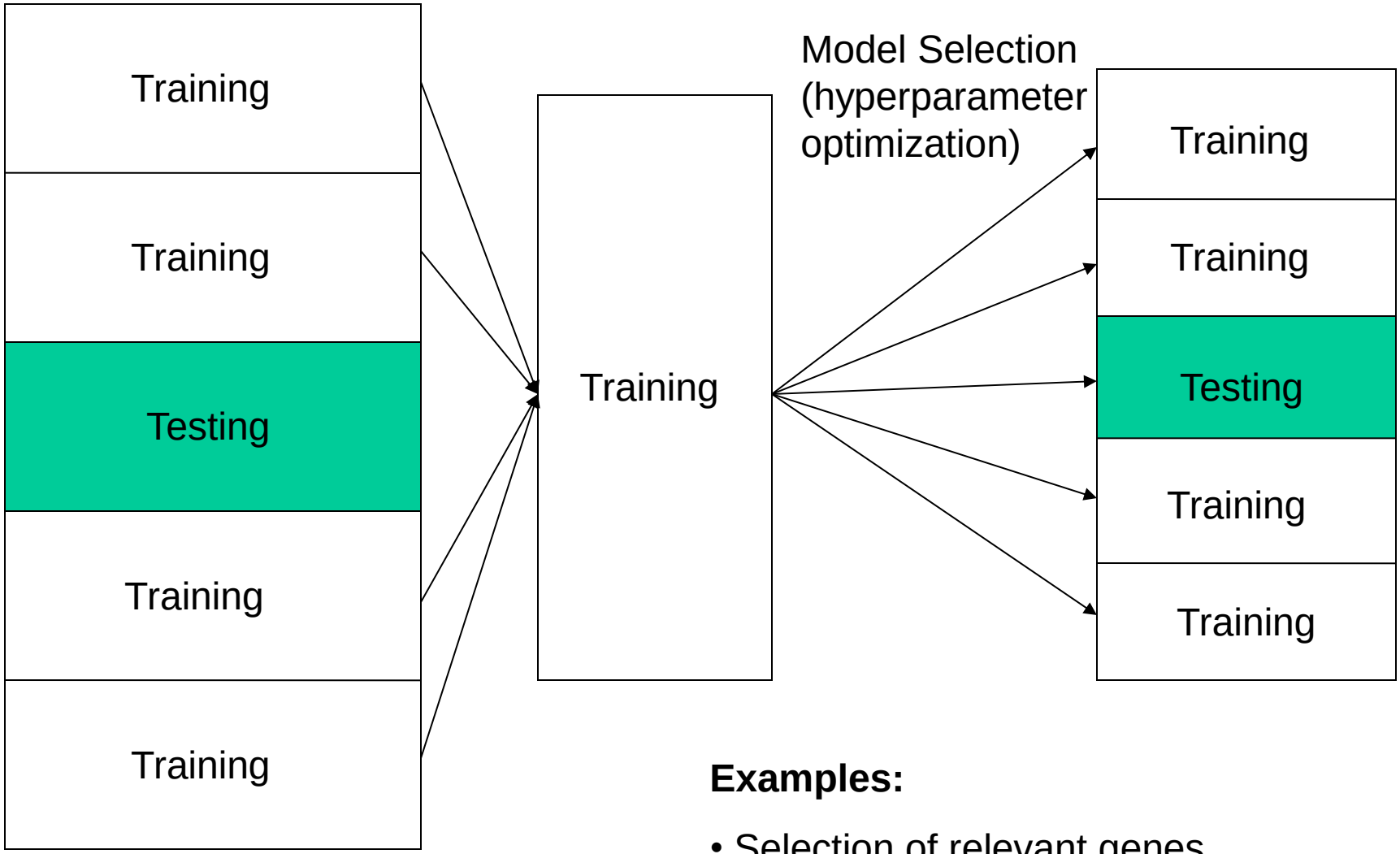
- Optimize SVM hyperparameters via k-fold CV
- for nonlinear SVMs: optimize kernel parameters via CV

Apply SVM on test set

- Select same genes as on training data

Training data only!

Nested Cross-Validation



Examples:

- Selection of relevant genes
- Tuning of SVM hyperparameters

DOs AND DON'Ts :

1. Decide on your classification model(s) (SVM, etc...) and do not change your mind later on
2. Think about how you want to measure classification accuracy

3. Use nested k-fold cross-validation procedure (repeated n times) to assess prediction performance:
   - Train and optimize your model using the data in the current ***training set*** only → (select genes, perform model selection ...)
   - Put the data in the current ***test set*** away ... far away
   - Do not even think of touching the test data at this time
   - Apply the model to the current ***test*** data ...
   - Do not even think of changing the model at this time

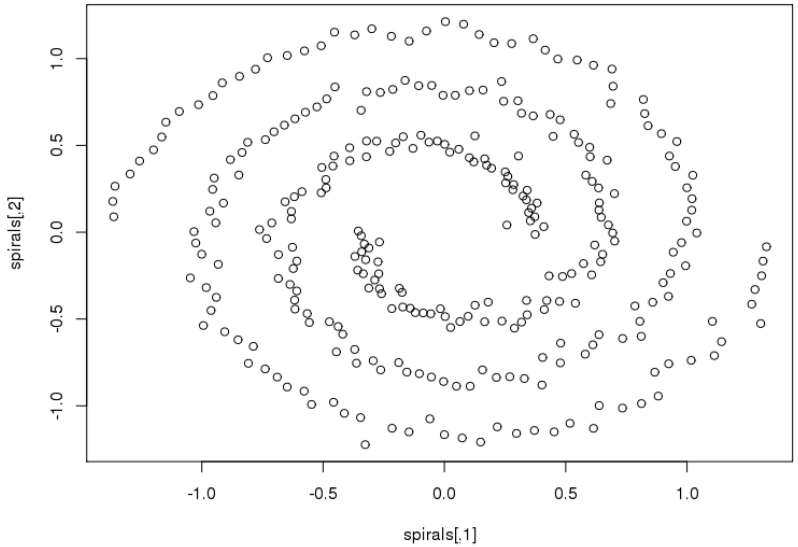
4. Do steps 1-3 only **once** and accept the result ...

- Do not even think of optimizing this procedure

Back to lower dimensional spaces

- Example: Can we predict HIV drug resistance?
- Data: amino acid sequence of parts of key protein of HI virus
  - extract features for few amino acids
  - 20 indicator variables for each amino acid
  - Few hundred features overall

- Data may become non-linear separable
  - Linear SVM does not work
  - Example: Can you learn to discriminate the two spirals?



Dual SVM Formulation

- Recall SVM optimization problem:

$$\min_{\mathbf{w}} \frac{1}{2} \| \mathbf{w} \|^2 + C \sum_{i=1}^{n} \xi_i$$

$$\text{subject to } y_i \left(\langle \mathbf{w}, \mathbf{x} \rangle + b \right) \geq 1 - \xi_i \text{ for } i = 1, \ldots, n$$

- We introduce Lagrangian multipliers to deal with the constraints:

$$L(\mathbf{w}, b, \boldsymbol{\alpha}, \boldsymbol{\xi}) = \frac{1}{2} \| \mathbf{w} \|^2 + C \sum_{i=1}^{n} \xi_i - \sum_{i=1}^{n} \alpha_i \left(y_i \left(\langle \mathbf{w}, \mathbf{x} \rangle + b \right) - 1 + \xi_i \right)$$

$$\frac{\partial}{\partial b} L(\mathbf{w}, b, \boldsymbol{\alpha}, \boldsymbol{\xi}) = \sum_{i=1}^{n} \alpha_i y_i$$

$$\frac{\partial}{\partial \mathbf{w}} L(\mathbf{w}, b, \boldsymbol{\alpha}, \boldsymbol{\xi}) = \mathbf{w} - \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i$$

$$\frac{\partial}{\partial \xi_i} L(\mathbf{w}, b, \boldsymbol{\alpha}, \boldsymbol{\xi}) = C - \alpha_i$$

Dual SVM Formulation

- At the optimum of L the first two derivatives have to equal 0:

$$\sum_{i=1}^{n} \alpha_i y_i = 0$$

$$\mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i$$

- Plugging back into *L* yields **dual** SVM optimization problem:

$$\max_{\boldsymbol{\alpha}} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle$$

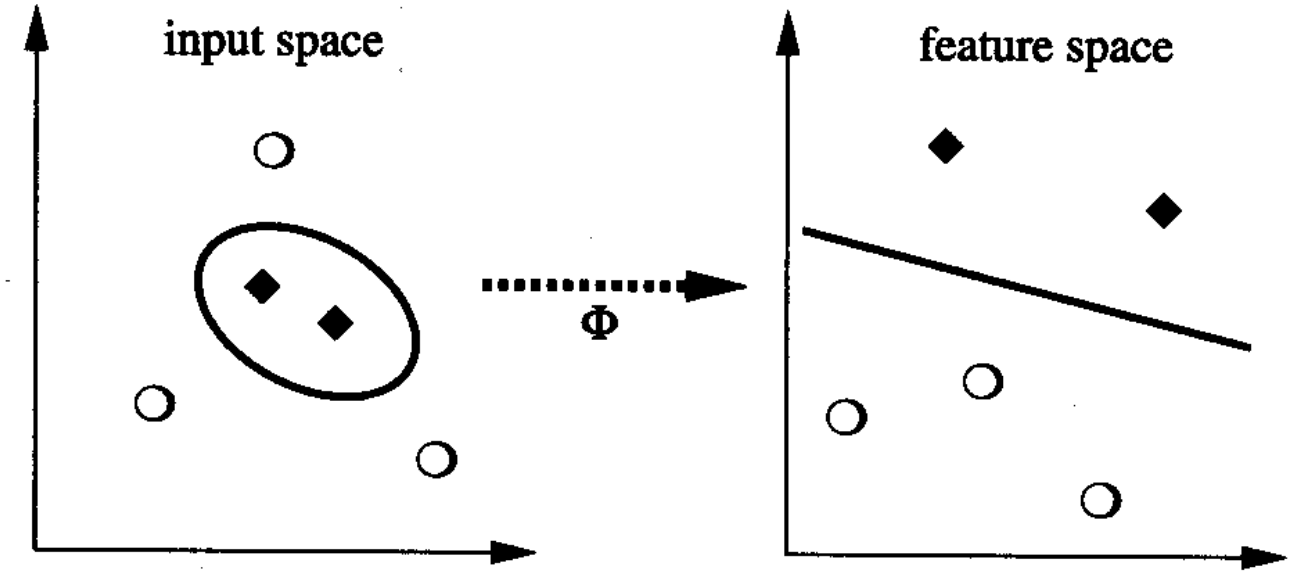
$$0 \le \alpha_i \le C$$

$$\sum_{i=1}^{n} \alpha_i y_i = 0$$

Dual SVM Formulation

- Most SVM algorithms solve the dual formulation
- Primal and dual SVM formulation yield exactly the same solution (because objective function is convex → KKT theorem), but there are also some differences:
- Primal formulation:
  - #parameters = #features
- Dual formulation:
  - #parameters = #data points
  - Only support vectors have non-zero α_i ➔ sparsity w.r.t. data points
  - Allows for non-linear classification (see next)

Nonlinear SVMs: the Kernel Trick

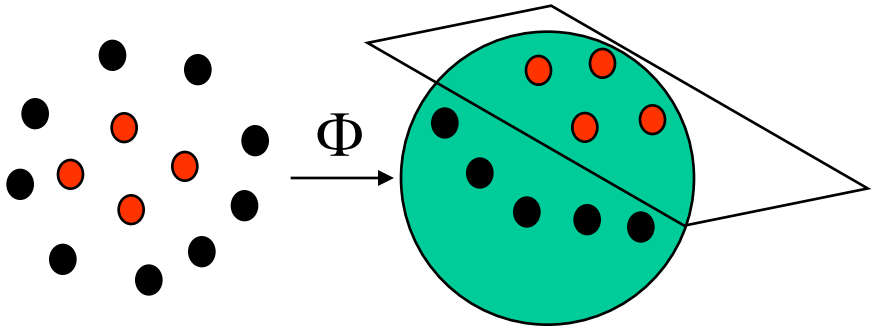


Source: Schölkopf, Smola, Learning with Kernels, MIT Press, 2002

$$\phi: X \to H, \text{ e.g. } \quad ([\mathbf{x}]_1, [\mathbf{x}]_2) \mapsto ([\mathbf{x}]_1^2, [\mathbf{x}]_2^2, [\mathbf{x}]_1[\mathbf{x}]_2)$$

Nonlinear SVMs: the Kernel Trick (Example)

- Projection of two-dimensional input data into three dimensional space (sphere) allows linear separation.



Nonlinear SVMs: the Kernel Trick

- For N-dimensional input space and monomials of 2nd order we have

$$([\mathbf{x}]_1,\ldots,[\mathbf{x}]_N) \mapsto ([\mathbf{x}]_1[\mathbf{x}]_{j_1},\ldots,[\mathbf{x}]_N[\mathbf{x}]_{j_N}),\ j_i \in \{1,\ldots,n\}$$

- => N(N+1)/2 coordinates in feature space
- In general, for degree *d* monomials:

$$N_{,} = \binom{d+N-1}{d} = \frac{(d+N-1)!}{d!(N-1)!}$$

- For N=256 and d=5 approx. dimension 10^{10} in feature space => curse of dimensionality!

Nonlinear SVMs: the Kernel Trick

- However: SVM algorithm can be expressed only in terms of dot products!
- We are only interested in dot products

$$\langle \phi(x), \phi(y) \rangle = k(x, y)$$

- k: $X \times X \rightarrow H$ is called a *kernel* function.

- Idea: read equation "backwards"
  - Define kernel function k, such that it corresponds to a dot product $<\phi(x), \phi(y)>$
  - Consequence: ϕ is defined <u>implicitly</u>
  - Which functions fulfill this property?

Examples of kernel functions

- Linear kernel: $k(\mathbf{x},\mathbf{y})=\langle \mathbf{x},\mathbf{y} \rangle$
- Polynomial kernel: $k(\mathbf{x},\mathbf{y})=(\langle \mathbf{x},\mathbf{y} \rangle +1)^d$
- RBF kernel: $k(\mathbf{x},\mathbf{y})=\exp(-||\mathbf{x}-\mathbf{y}||^2/\sigma^2)$

- All sums and products of these kernels:
  - Possibility to define dependent kernels, e.g. for chemical compounds, biological sequences

- General: Kernel matrix $\mathbf{K} = \left(k(\mathbf{x}_i, \mathbf{x}_j)\right)_{ij}$ has to be provably symmetric and positive semi-definite (for all possible training data)

Nonlinear SVMs with Kernels

$$\max_{\boldsymbol{\alpha}} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j k(x_i, x_j)$$

$$\text{subject to } \; 0 \le \alpha_i \le \mathrm{C} \text{ and } \sum_{i=1}^{n} \alpha_i y_i = 0$$

dual SVM formulation

$$\Leftrightarrow$$

$$\max_{\boldsymbol{\alpha}} \mathbf{1}^T \boldsymbol{\alpha} - \frac{1}{2} \boldsymbol{\alpha}^T \mathbf{Q} \boldsymbol{\alpha}$$

$$\text{subject to } \mathbf{y}^T \boldsymbol{\alpha} = 0 \text{ and } \mathbf{0} \le \boldsymbol{\alpha} \le \mathbf{C}$$

$$\text{with } Q = \left(y_i y_j k(x_i, x_j) \right)_{ij}$$

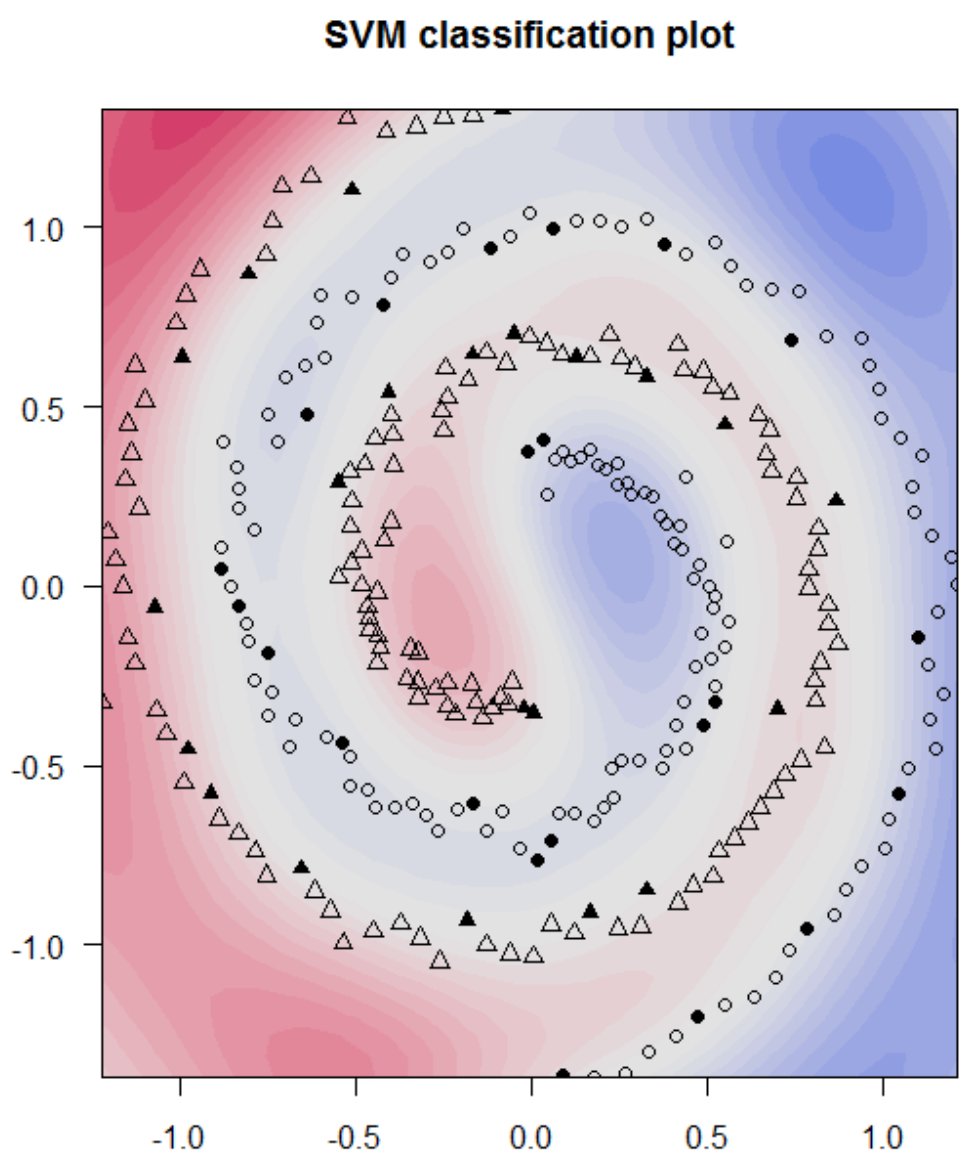
$K=(k(x_i,x_j))_{ij}$ is called *kernel matrix*.

dual SVM formulation in matrix-vector notation

- There is one coefficient (Lagrangian multiplier) per data point
- Only coefficients for support vectors are non-zero!

Example: SVM with RBF Kernel

- RBF kernel allows for non-linear classification of the two spirals with 0 training errors
- Filled dots: Support Vectors



How to classify a new data point?

- Linear SVM: $f(\mathbf{x}) = \mathrm{sgn}(<\mathbf{w}, \mathbf{x}> + b)$
- Kernelized SVM:

$$f(\mathbf{x}) = \mathrm{sgn}\left(\sum_{i \in SV} \alpha_i y_i k(\mathbf{x}, \mathbf{x}_i) + b \right)$$

- SVM decision function depends only on support vectors!
  - Thus the name „support vector machine“

- SVM is not sparse in terms of selected features
- It is sparse in terms of relevant data points (support vectors)

Multi-Class Classification

- So far: binary classification $f: X \rightarrow \{+1,-1\}$.
- Now: multi-class classification $f: X \rightarrow \{1,\ldots,k\}$.

- Most prominent solutions
  - One versus the rest
  - Pair-wise classification

- Multi-class SVM also exists, but is less common (slow computation)

Multi-Class Classification
- One versus the rest -

- Construct k classifier $f_1,\ldots,f_k$, each trained to separate one class from the rest
- Assign a pattern x to that class for which $g_i(x)$ is maximal (winner-takes-all strategy)

- Problem: unclear whether the g_i are on comparable scales (different binary problems!)
- Asymmetric data: more negatives than positive examples (especially with many classes)

Multi-Class Classification
- Pairwise Classification -

- Train a classifier for each possible pair of classes => k(k-1)/2 binary classifiers
- Assign pattern x to the class which gets the highest number of votes from all classifiers

- Advantages
  - Problem of asymmetry not as serious as in 1-vs.-rest
  - smaller individual problems
- Disadvantages
  - May be slower than 1-vs.-rest

Summary

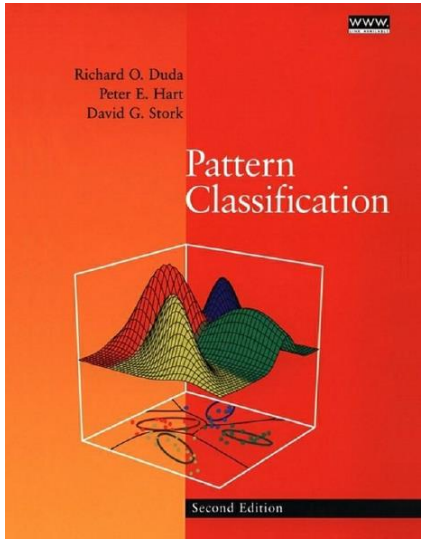
- SVMs
  - SVMs yield an optimal hyperplane with maximal margin
  - Convex optimization problem ➔ unique, globally optimal solution
  - Extension for non-linear classification (so-called kernel trick)
  - Extension for multi-class classification
  - One of the most popular classification algorithms today
  - Large body literature, many variations

- SVMs for gene expression data
  - Special challenge: many more genes than patient samples
  - Consequence: overfitting
  - Feature selection problem
    - Simple filtering approach: t-test + log-FC cutoff

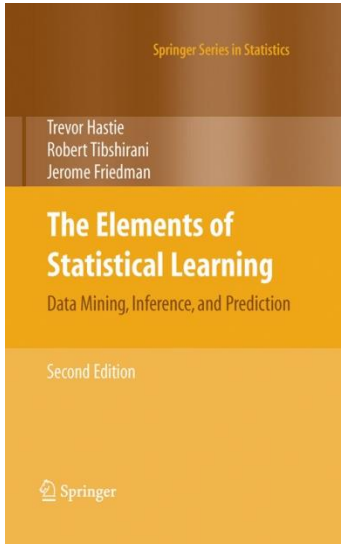
Acknowledgements

- A few slides were adapted from Prof. Dr. Rainer Spang‘s (University of Regensburg) talk on microarray classification within the BMBF funded Practical Microarray Analysis courses

- Further sources:



http://davinci22.tach.ula.ve/documents/vermig/Pattern%20Classification.pdf



http://statweb.stanford.edu/~tibs/ElemStatLearn/