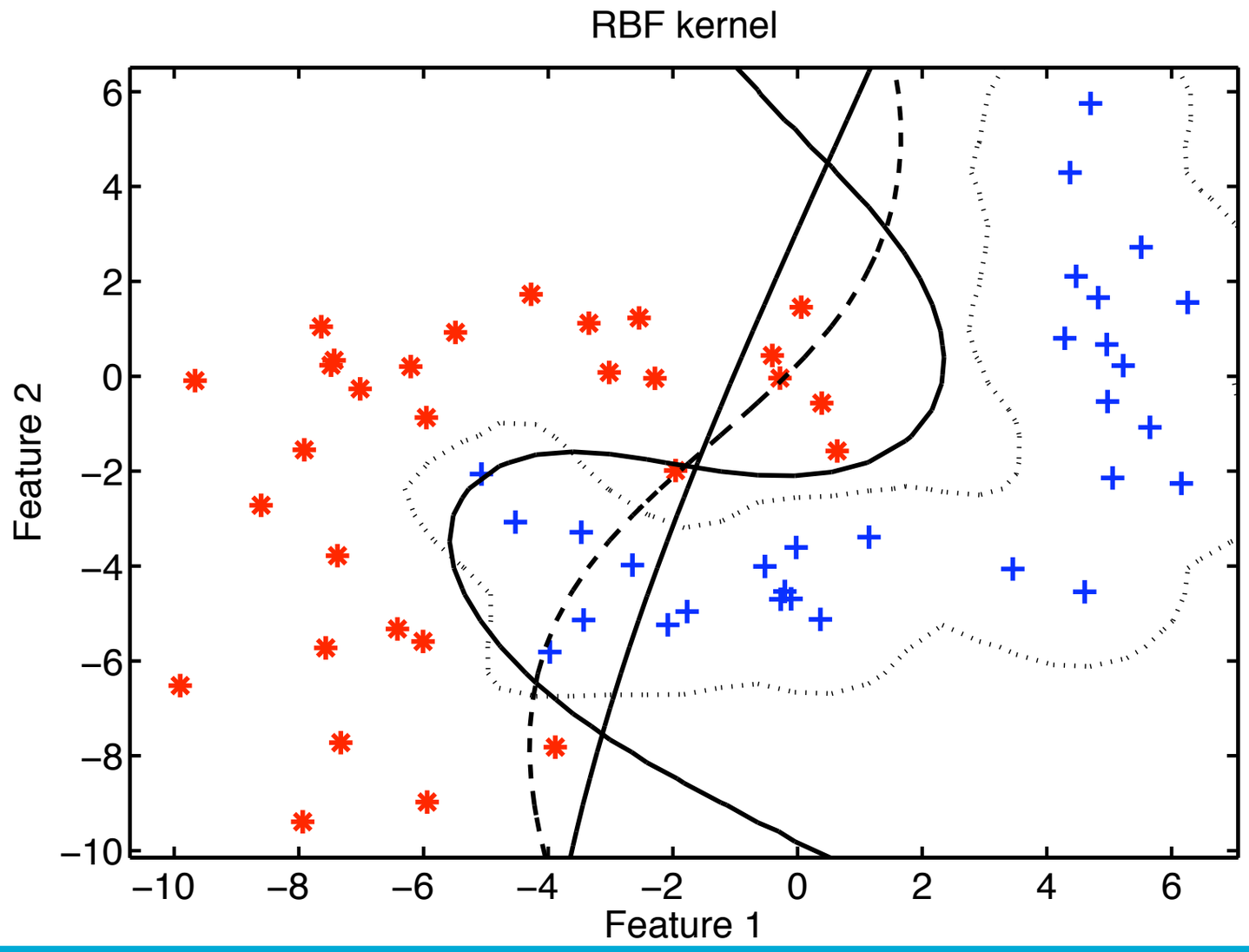
Complexity and Support Vector Classifiers

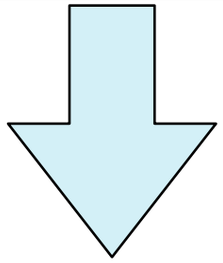


David M.J. Tax

TU Delft

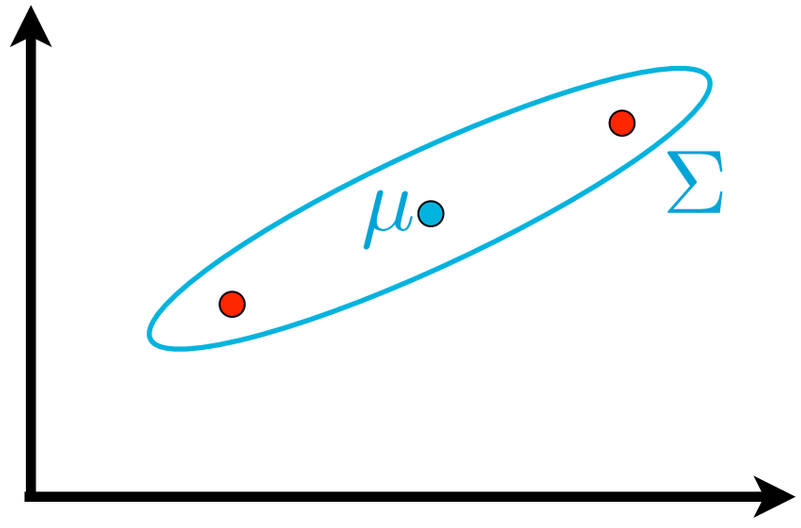
Contents

- How to characterise complexity?
- How to adapt the complexity?
- Measuring complexity: VC-dimension
- Support vector classifiers
  - Constrained optimisation
  - Advantages and disadvantages
- Examples
- Conclusions
- (Bonus?)



TU Delft

Quadratic classifier



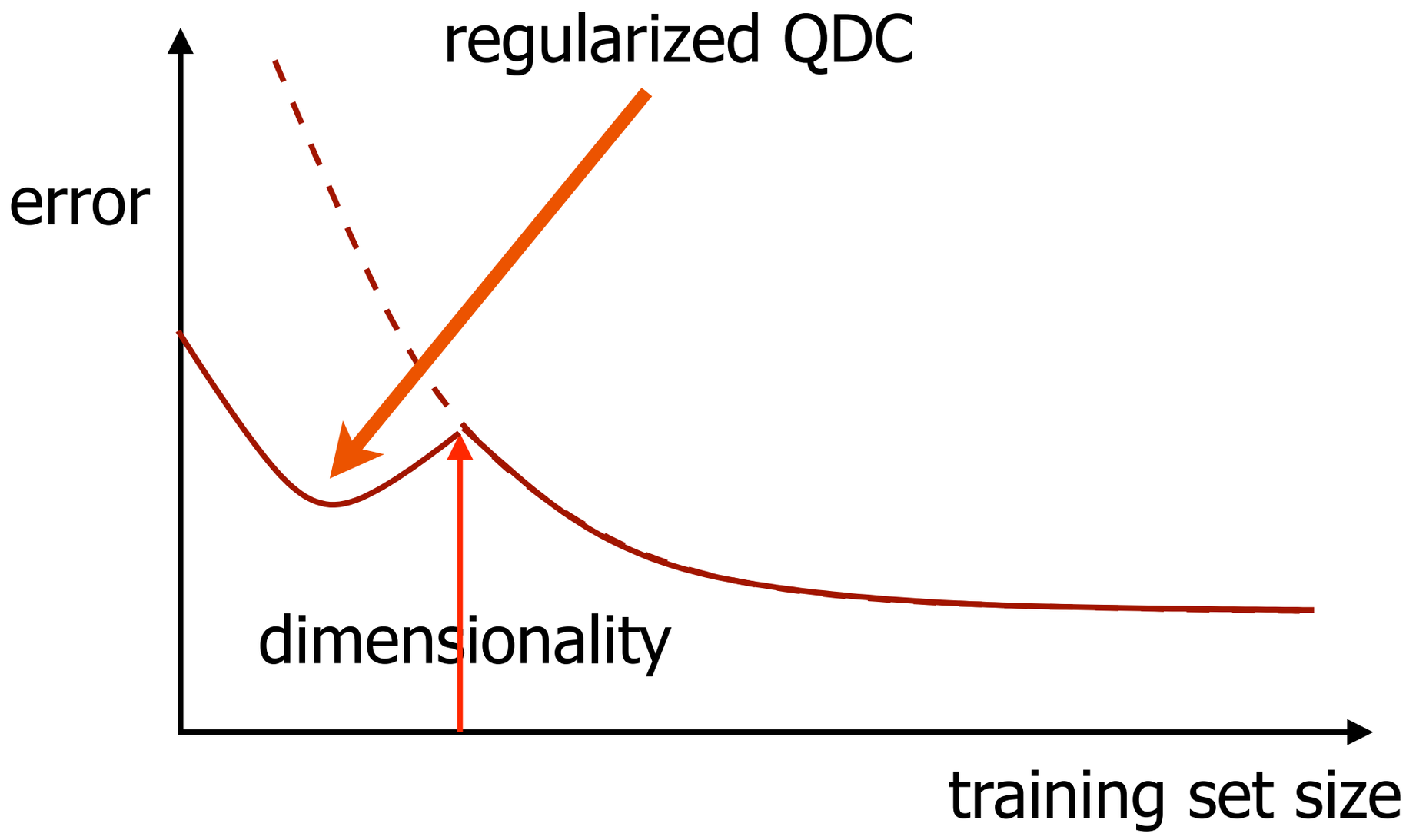
- When insufficient data is available (nr of obj's is smaller than dimensionality), the inverse covariance matrices are not defined: the classifier is not defined

- **Regularization** solves it: $\tilde{\Sigma}_i = \Sigma_i + \lambda \mathbb{I}$

- With very large regularisation $\lambda \to \infty$ you get: nearest mean classifier

TU Delft

Quadratic classifier



TU Delft

Bounding the true error

With probability at least $1-\eta$ the inequality holds:

$$\varepsilon \leq \varepsilon_A + \frac{\mathcal{E}(N)}{2}\left(1+\sqrt{1+\frac{\varepsilon_A}{\mathcal{E}(N)}}\right)$$

where

$$\mathcal{E}(N) = 4\frac{h(\ln(2N/h)+1)-\ln(\eta/4)}{N}$$

V.Vapnik, Statistical learning theory, 1998

- When h is small, the true error is close to the apparent error

TU Delft

Is this bound practical?

- The given bound (and others) is very loose
- The worst case scenario is assumed: objects can be randomly labeled
- In practice, features are chosen such that objects from one class are nicely clustered and can be separated from other classes

TU Delft

Compactness hypothesis

Representations of real world objects are close.
There is no ground for any generalization on representations that do not obey this demand.

A.G. Arkadev and E.M.Braverman,
Computers and Pattern Recognition, 1966

TU Delft

Changing h for linear classifiers

- The linear classifier had $h = p + 1$

- By putting some constraints on this linear classifier, the VC dimension can be reduced
- Assume a linearly separable dataset,

constrain the weights such that the output of the classifier is always larger than one:

$$\mathbf{w}^T \mathbf{x}_i + b \geq +1, \quad \text{for } y_i = +1$$
$$\mathbf{w}^T \mathbf{x}_i + b \leq -1, \quad \text{for } y_i = -1$$

TU Delft

Minimizing the Lagrangian

- Lagrangian:

$$\mathcal{L}(\mathbf{w}, \alpha) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_i \alpha_i \left(y_i(\mathbf{w}^T\mathbf{x}_i + b) - 1\right)$$

- Have to take the derivative with respect to **w**, b, and α_i and set the derivative to 0

$$\frac{\partial \mathcal{L}(\mathbf{w}, \alpha)}{\partial \mathbf{w}} = \mathbf{w} - \sum_i \alpha_i y_i \mathbf{x}_i = 0$$

- Solve:

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

TU Delft

Derivative of norm

- What is $\frac{1}{2}\frac{\partial \|\mathbf{w}\|^2}{\partial \mathbf{w}}$?
- Well, first $\frac{1}{2}\|\mathbf{w}\|^2 = \frac{1}{2}\sum_{i=1}^{p} w_i^2$
- Take derivative w.r.t. one parameter:

$$\frac{1}{2}\frac{\partial \|\mathbf{w}\|^2}{\partial w_1} = w_1 \qquad \frac{1}{2}\frac{\partial \|\mathbf{w}\|^2}{\partial w_2} = w_2$$

- In vector notation: $\frac{1}{2}\frac{\partial \|\mathbf{w}\|^2}{\partial \mathbf{w}} = \mathbf{w}$

TU Delft

Derivative of inner product

- What is $\frac{\partial \mathbf{w}^T \mathbf{x}_i}{\partial \mathbf{w}}$?
- Write it out:

$$\frac{\partial}{\partial w_1}(\mathbf{w}^T \mathbf{x}_i) = \frac{\partial}{\partial w_1}\left(\sum_{j=1}^{N} w_j x_{ij}\right)$$

$$= \frac{\partial}{\partial w_1}(w_1 x_{i1}) = x_1$$

- Also for other w_j
- In vector notation:

$$\frac{\partial \mathbf{w}^T \mathbf{x}_i}{\partial \mathbf{w}} = \mathbf{x}_i$$

TU Delft

Optimizing the classifier

- Solving w.r.t. **w** and b is "simple".
- Solving w.r.t. α_i gives:

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

$$\alpha_i \geq 0 \quad \forall i$$

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

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

- Quadratic Programming Problem

TU Delft

Support vectors

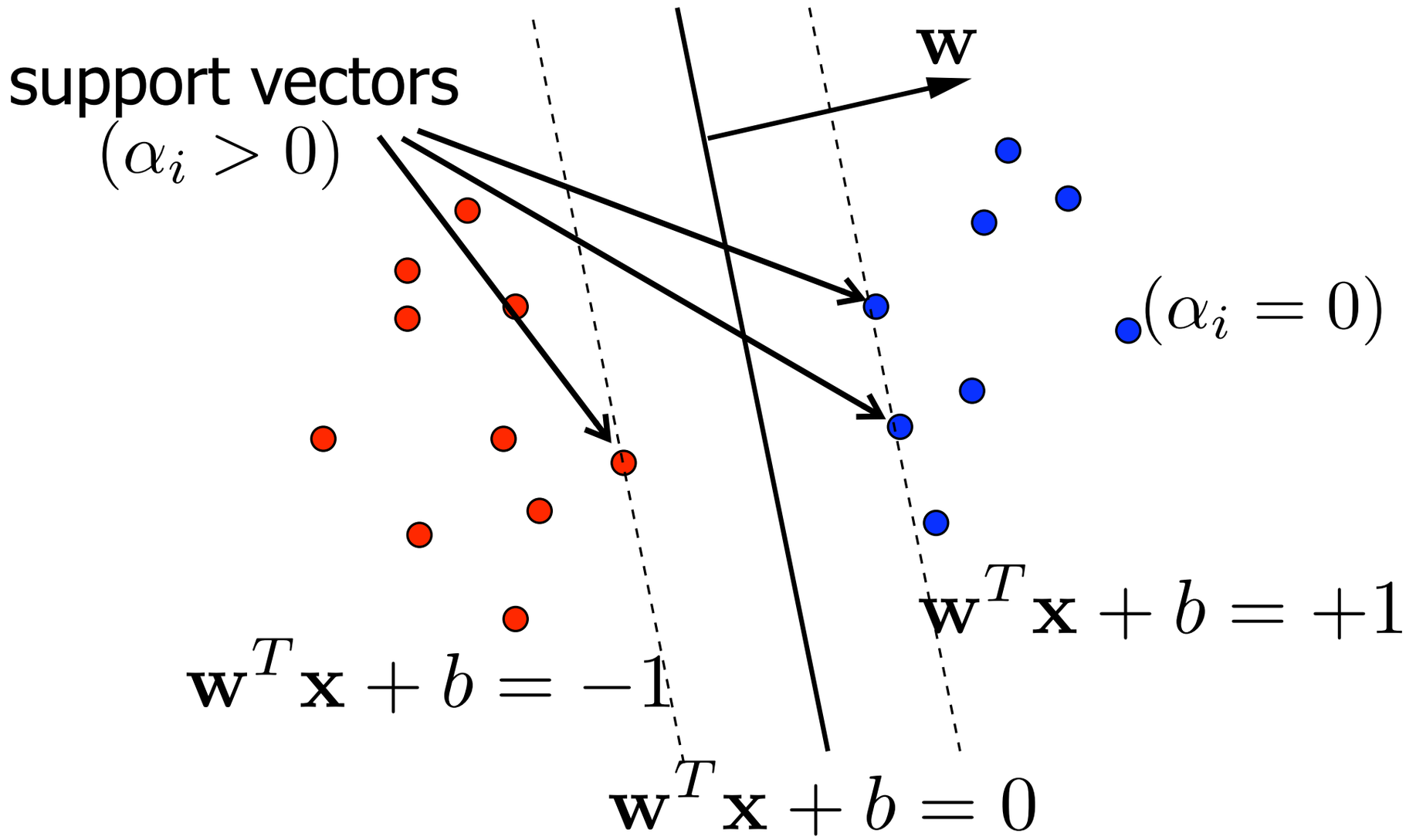
- The classifier becomes:

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

- The solution is expressed in terms of objects, not features.
- Only a few weights become non-zero
- The objects with non-zero weight are called the support vectors

TU Delft

Support vector classifier

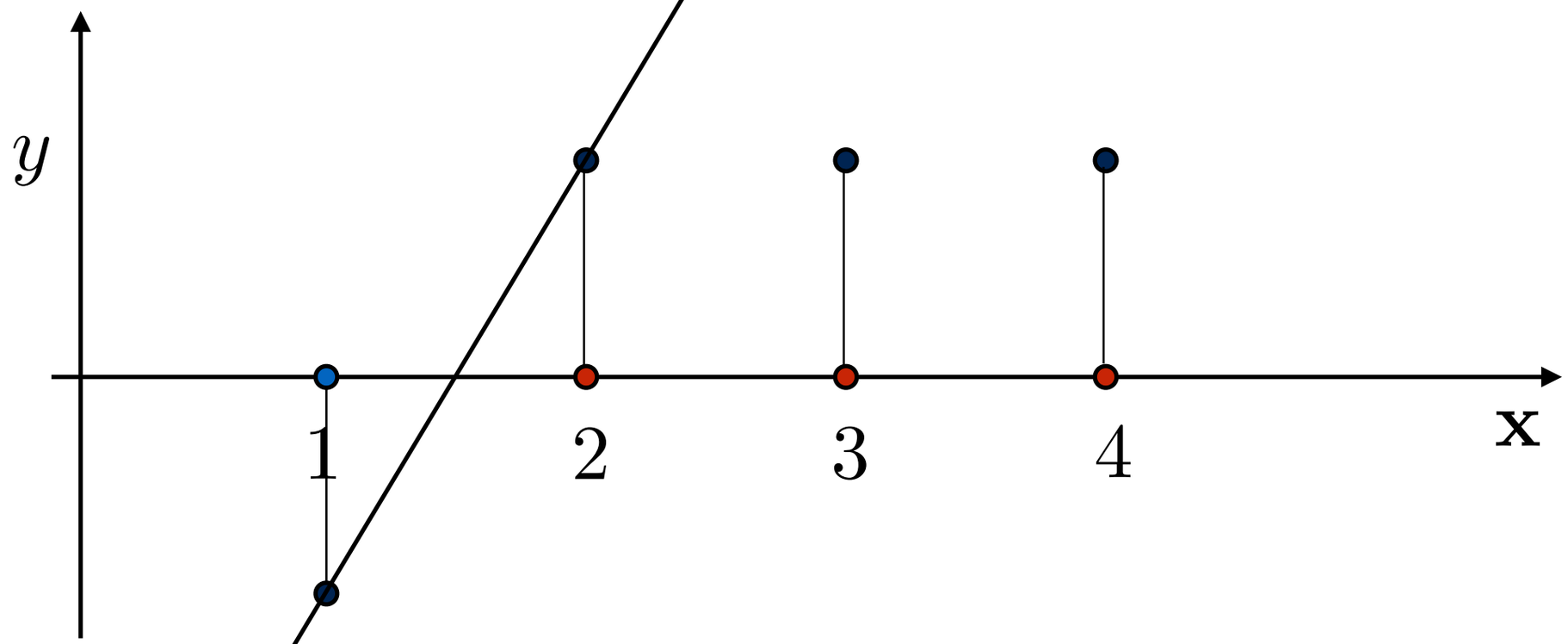


TU Delft

Use a linear classifier for:

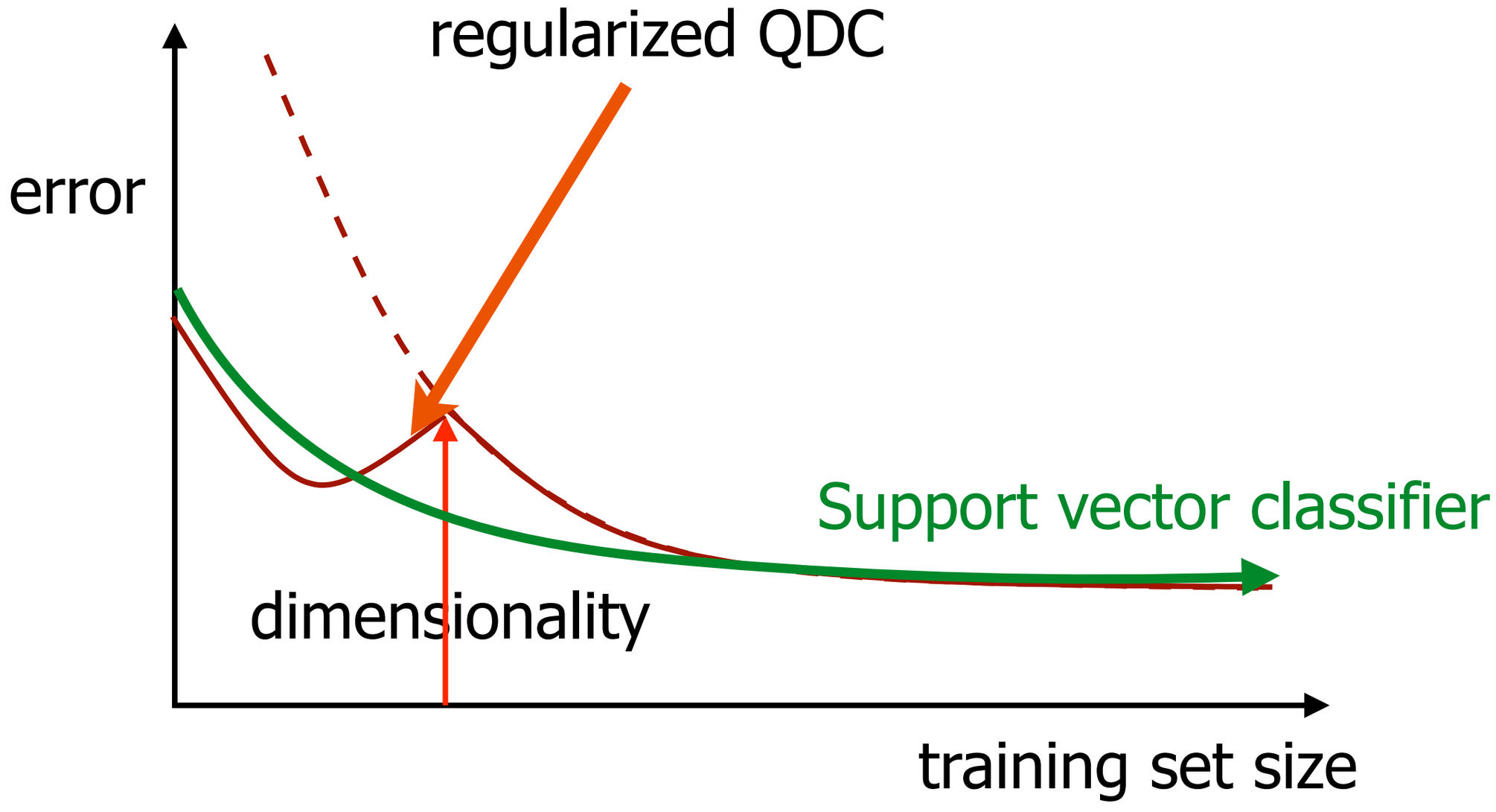
$$y = \text{sign}(\mathbf{w}^T \mathbf{x} + w_0)$$

- What are the optimal **w** and w_0?



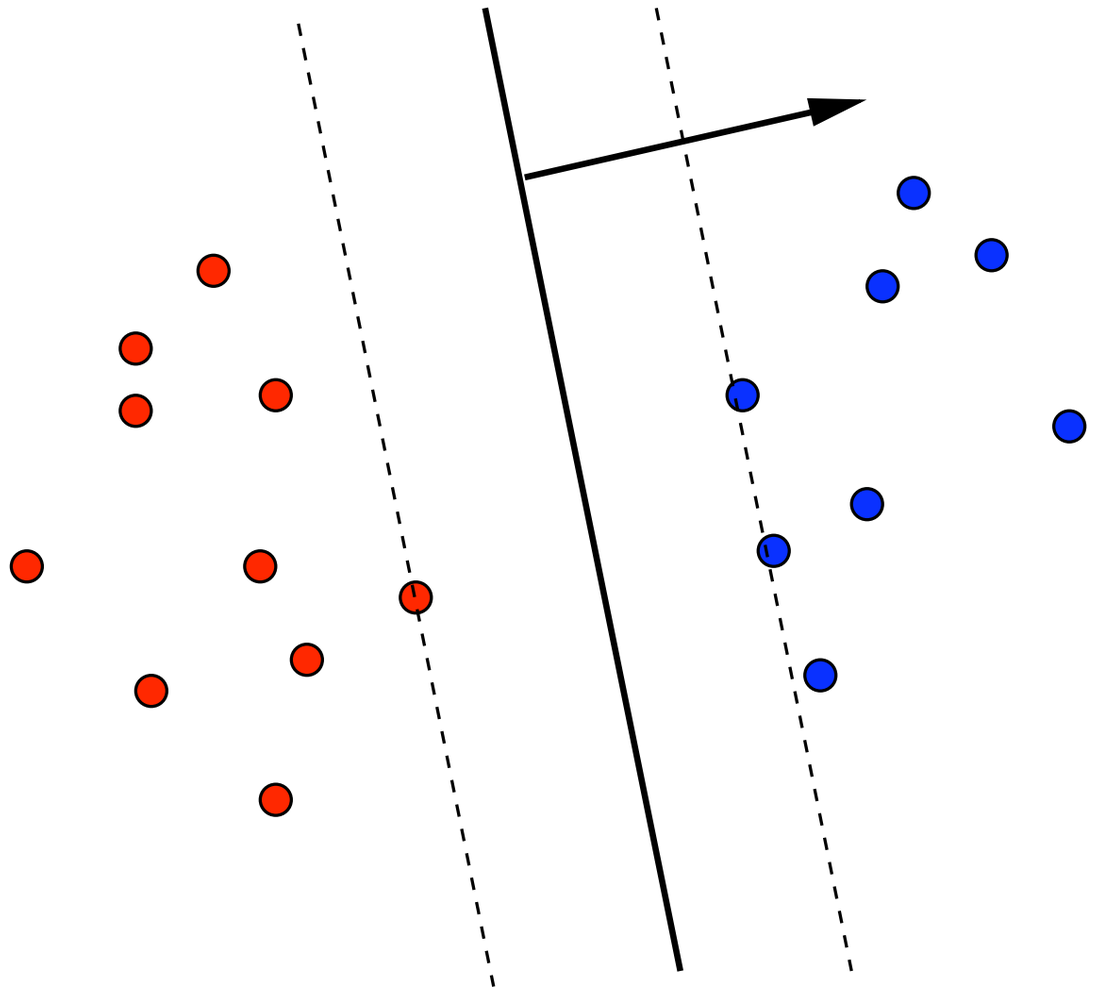
TU Delft

Quadratic classifier vs. SVC



TU Delft

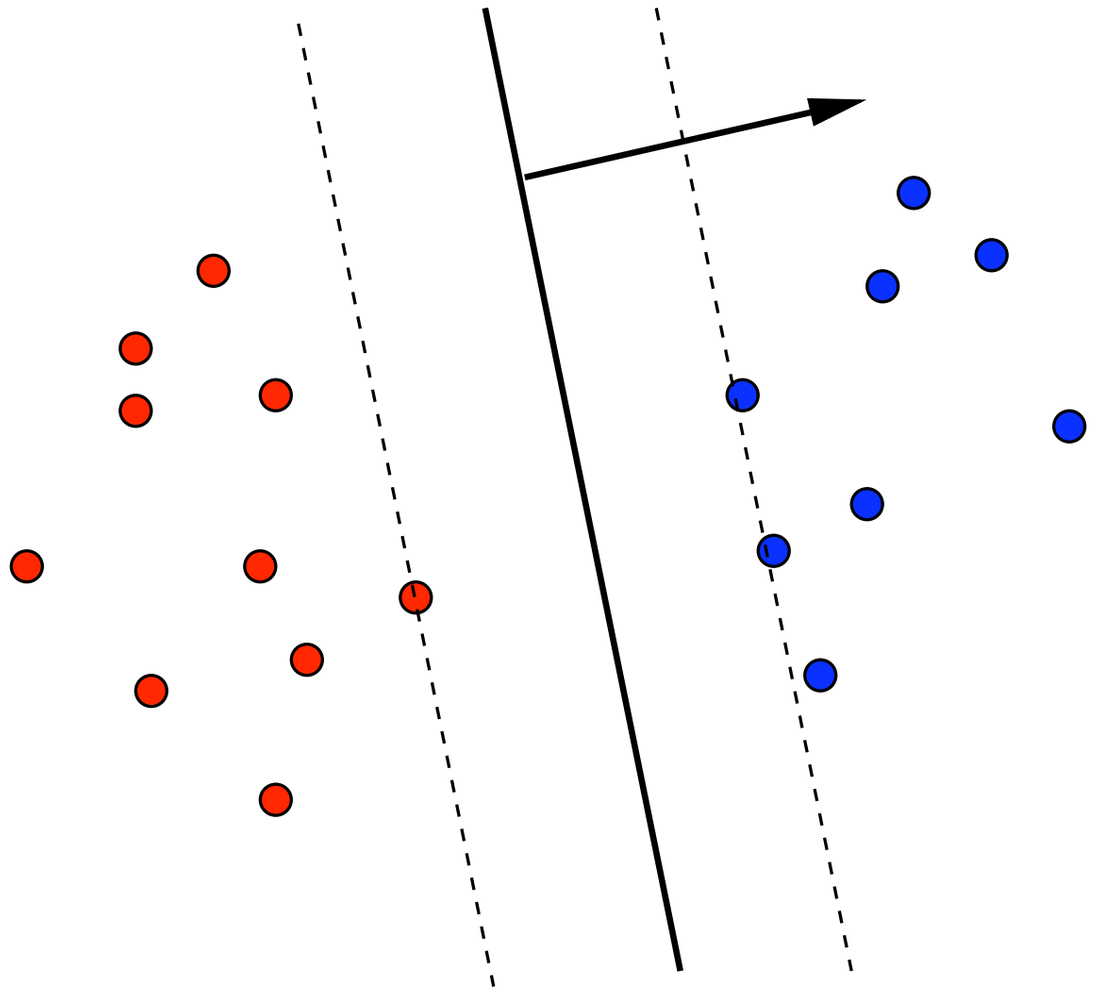
LOO Error estimation



Estimate the Leave-One-Out error on this data…

TU Delft

Error estimation

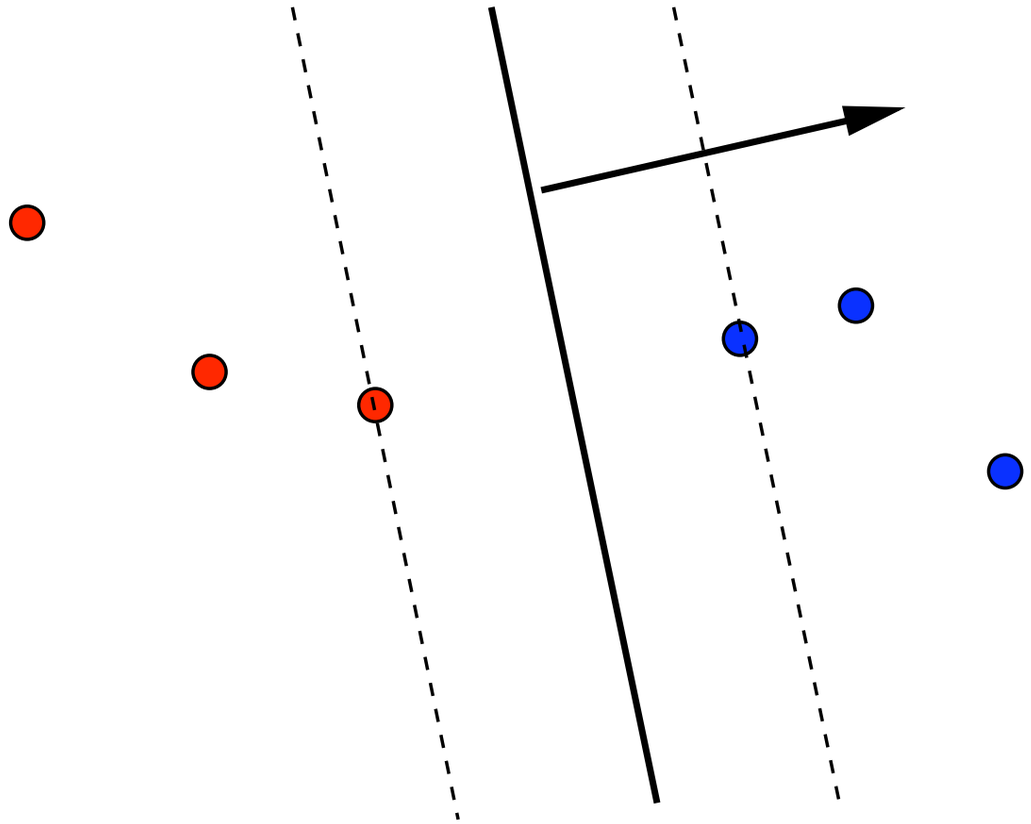


By leave-one-out you can obtain a bound on the error:

$$\varepsilon_{LOO} \leq \frac{\#\text{support vectors}}{N}$$

TU Delft

High dimensional feature spaces



The classifier is determined by objects, not features: the classifier tends to perform very well in high dimensional feature spaces.

TU Delft

Limitations of the SVM

1. The data should be separable

2. The decision boundary is linear

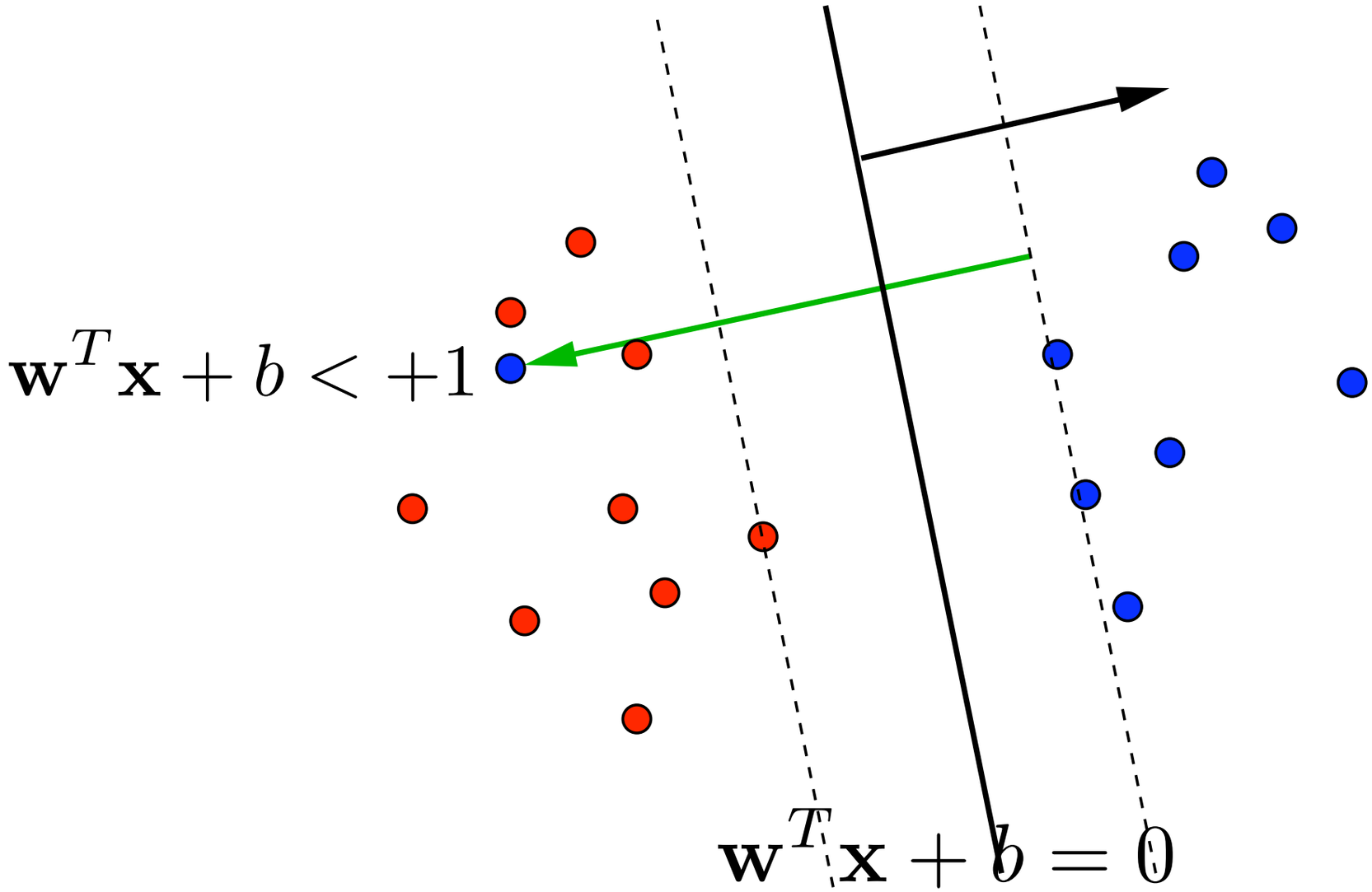
TU Delft

Problem 1

The classes should be separable...

TU Delft

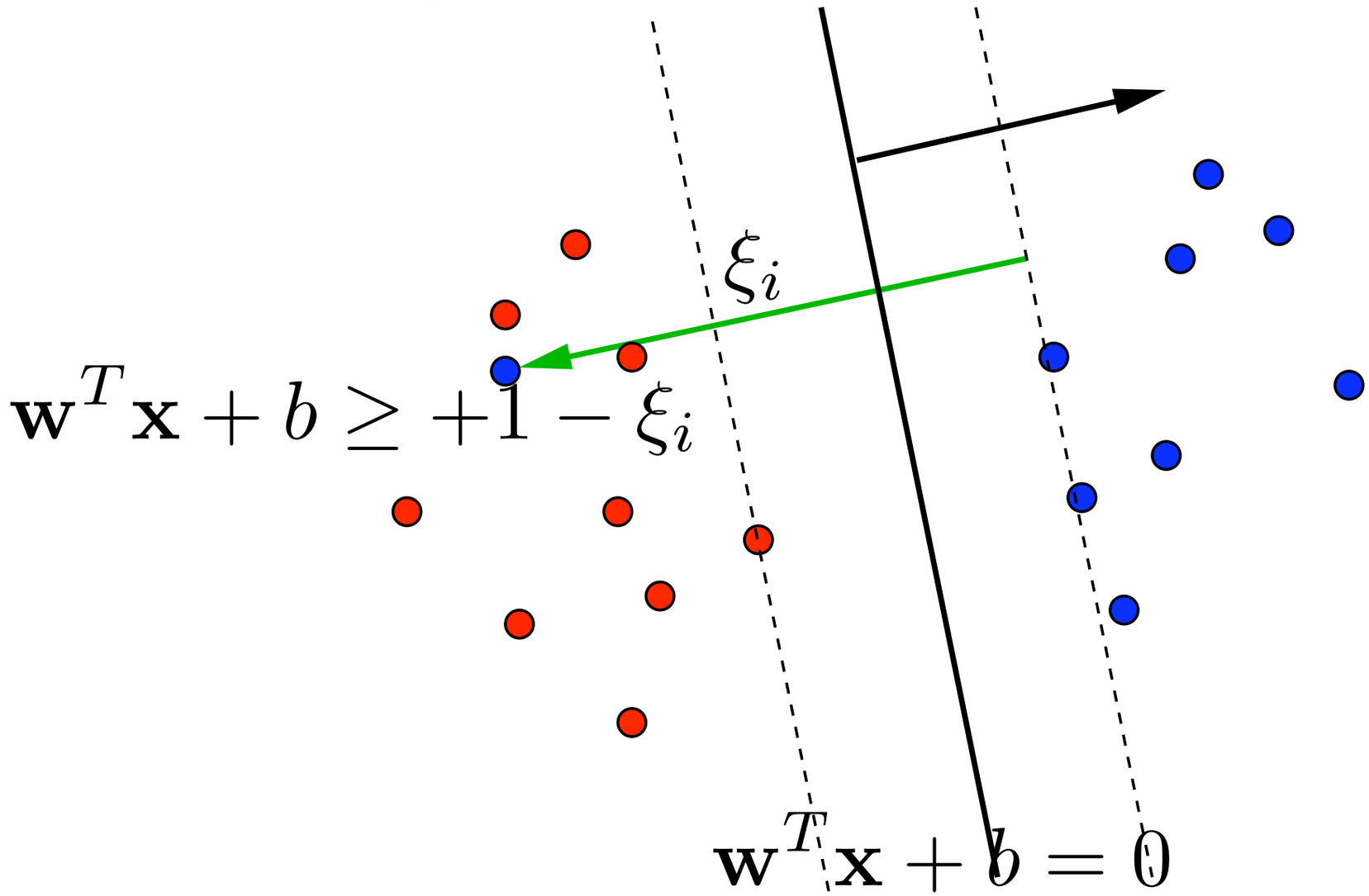
Problem 1



Classes should be separable

TU Delft

Weakening the constraints



Introduce a 'slack' variable ξ_i for each object, to weaken the constraints

TU Delft

SVM with slacks

- The optimization changes into:

$$\min \|\mathbf{w}\|^2 + C \sum_{i=1}^{N} \xi_i$$

$$\mathbf{w}^T \mathbf{x}_i + b \geq +1 - \xi_i, \quad \text{for } y_i = +1$$

$$\mathbf{w}^T \mathbf{x}_i + b \leq -1 + \xi_i, \quad \text{for } y_i = -1$$

$$\xi_i \geq 0 \quad \forall i$$

TU Delft

Optimizing the classifier

- Solving w.r.t. **w** and b is "simple".
- Solving w.r.t. α_i gives:

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

$$0 \leq \alpha_i \leq C \quad \forall i$$

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

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

- Quadratic Programming Problem

TU Delft

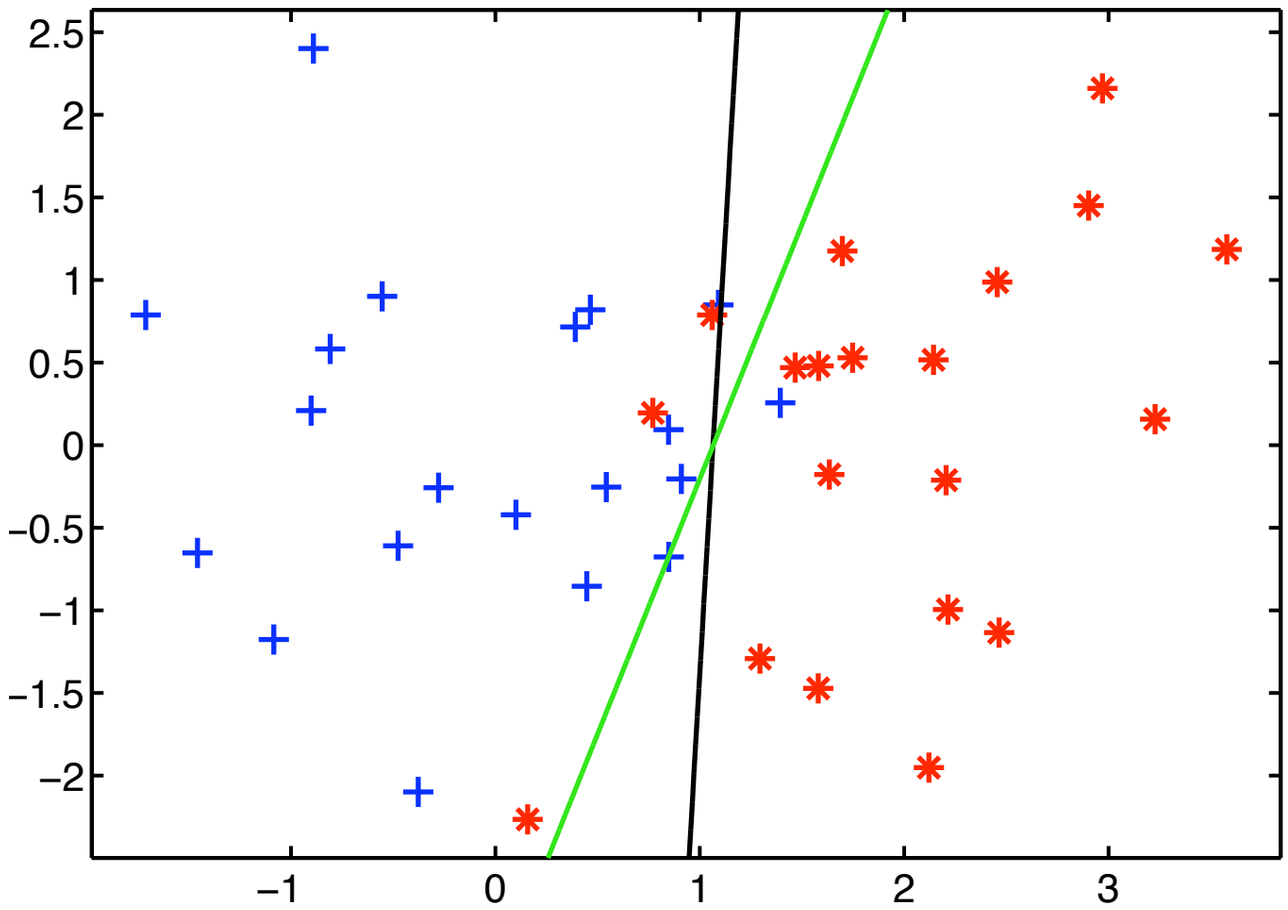
Tradeoff parameter C

$$\min \|\mathbf{w}\|^2 + C \sum_{i=1}^{N} \xi_i$$

- Notice that the tradeoff parameter C has to be defined beforehand
- It weighs the contributions between the training error and the structural error
- Its values are often optimized using crossvalidation.

TU Delft

Influence of C



C=1

C=1000

Erroneous objects have still large influence when C is not optimized carefully

TU Delft

Problem 2

The decision boundary is only linear...

TU Delft

Trick: transform your data

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

$$\alpha_i \geq 0 \quad \forall i$$

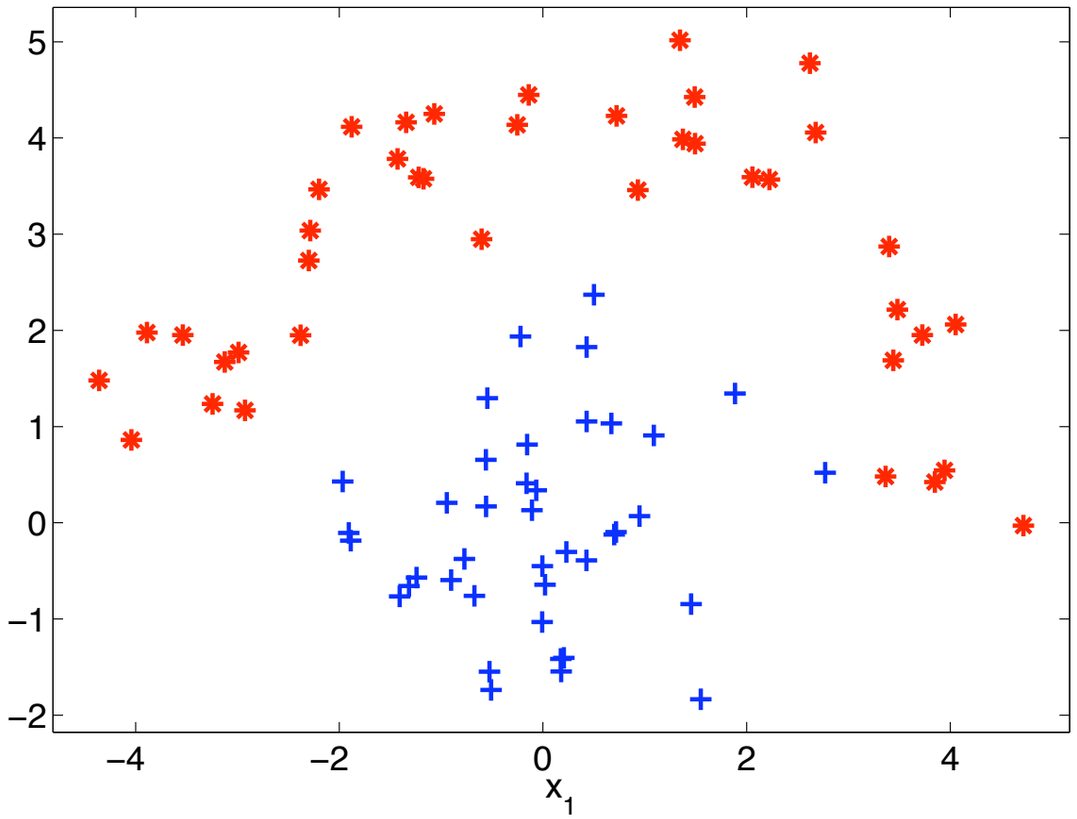
$$\sum_{i=1}^{N} \alpha_i y_i = 0$$

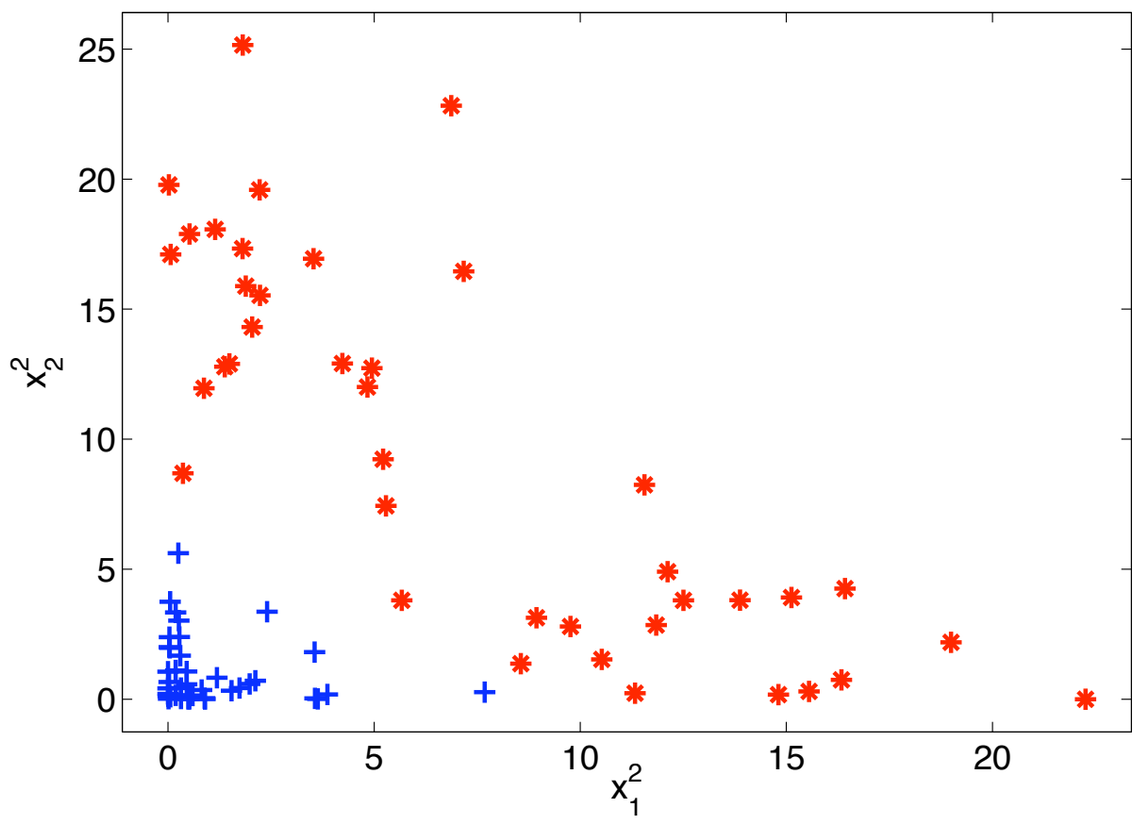
$$f(\mathbf{z}) = \mathbf{w}^T \mathbf{z} + b = \sum_{i=1}^{N} \alpha_i y_i \mathbf{x}_i^T \mathbf{z} + b$$

- Magic: all operations are on inner products between objects
- Assume I have a magic transformation of my data that makes it linearly separable...

TU Delft

Example transformation





- Original data: $\mathbf{x} = (x_1, x_2)$
- Mapped data: $\mathbf{\Phi}(\mathbf{x}) = (x_1^2, x_2^2, \sqrt{2}x_1x_2)$

TU Delft

Polynomial kernel

- When we have two vectors

$$\mathbf{\Phi}(\mathbf{x}) = (x_1^2, x_2^2, \sqrt{2}x_1x_2)$$

$$\mathbf{\Phi}(\mathbf{y}) = (y_1^2, y_2^2, \sqrt{2}y_1y_2)$$

$$\begin{aligned}\mathbf{\Phi}(\mathbf{x})^T\mathbf{\Phi}(\mathbf{y}) &= x_1^2y_1^2 + x_2^2y_2^2 + 2x_1x_2y_1y_2 \\ &= \left((x_1, x_2)(y_1, y_2)^T\right)^2 \\ &= \left(\mathbf{x}^T\mathbf{y}\right)^2\end{aligned}$$

it becomes very cheap to compute the inner product.

TU Delft

Transform your data

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

$$\sum_{i=1}^{N} \alpha_i y_i = 0 \quad \alpha_i \geq 0 \quad \forall i$$

$$f(\mathbf{z}) = \sum_{i=1}^{N} \alpha_i y_i \mathbf{\Phi}(\mathbf{x}_i)^T \mathbf{\Phi}(\mathbf{z}) + b$$

- If we have to introduce the magic $\mathbf{\Phi}(\mathbf{x})$ we can as well introduce the magic kernel function:

$$K(\mathbf{x}, \mathbf{y}) = \mathbf{\Phi}(\mathbf{x})^T \mathbf{\Phi}(\mathbf{y})$$

TU Delft

Transform your data

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

$$\sum_{i=1}^{N} \alpha_i y_i = 0 \quad \alpha_i \geq 0 \quad \forall i$$

$$f(\mathbf{z}) = \sum_{i=1}^{N} \alpha_i y_i K(\mathbf{x}_i, \mathbf{z}) + b$$

- We only define the kernel function, and forget about the mapping $\mathbf{\Phi}(\mathbf{x})$
- For instance: $K(\mathbf{x}_i, \mathbf{z}) = \exp\left(-\frac{\|\mathbf{x} - \mathbf{z}\|^2}{\sigma^2}\right)$

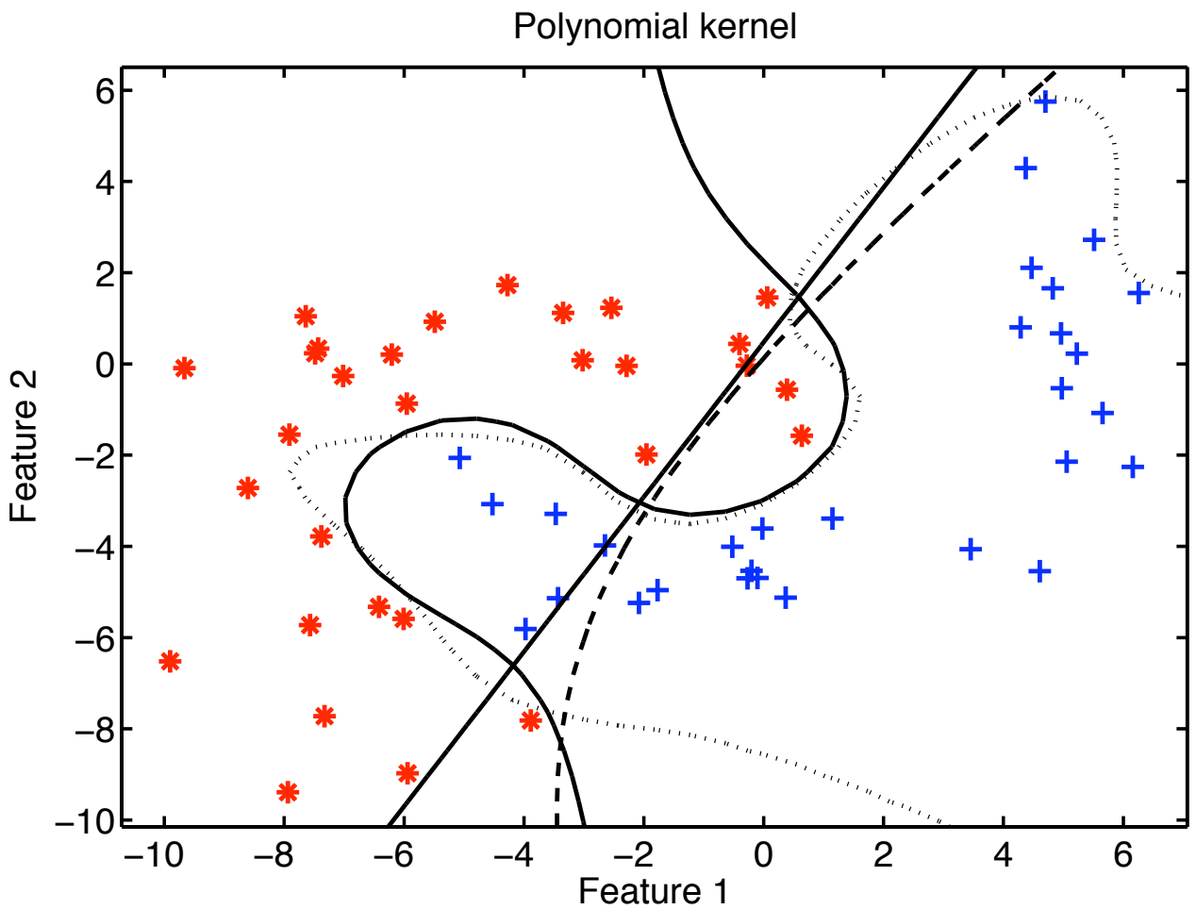
TU Delft

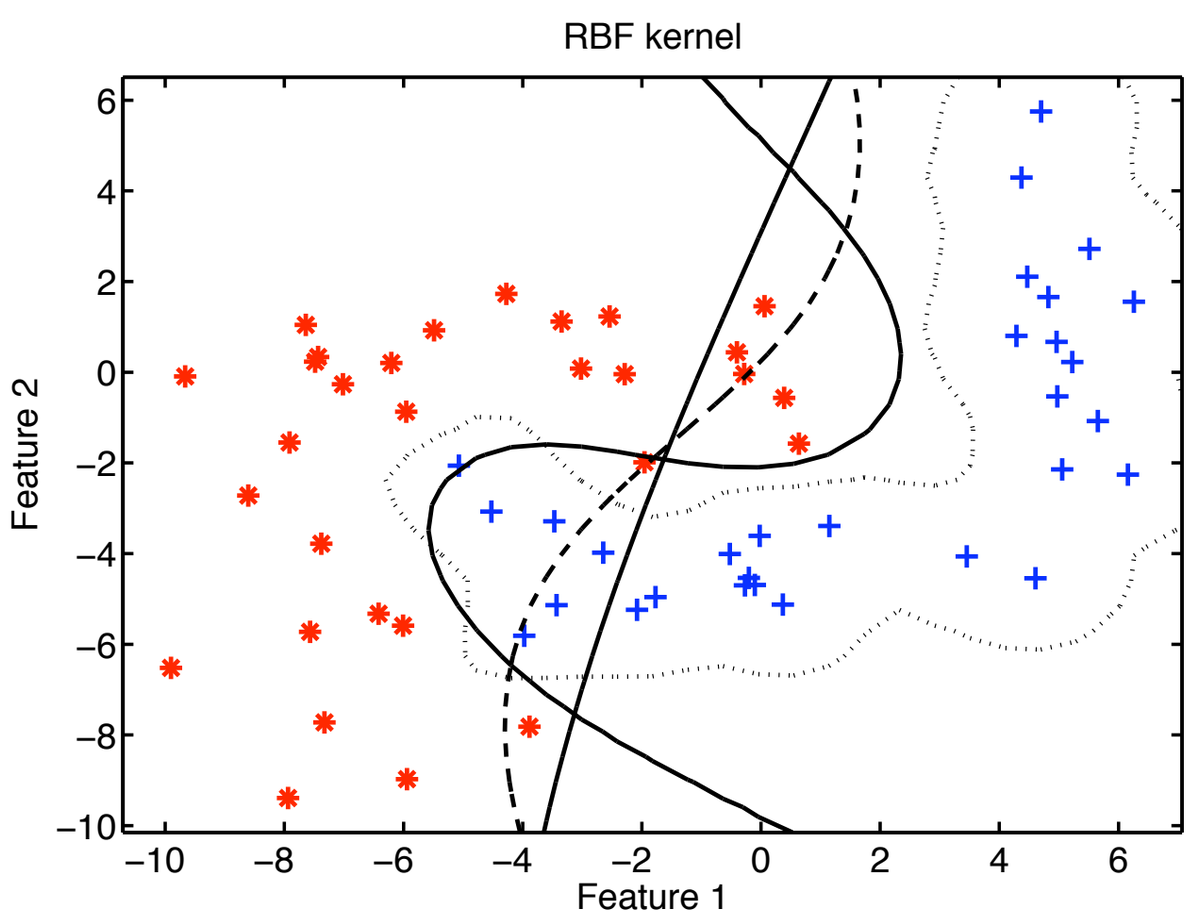
The 'kernel-trick'

- The idea to replace all inner products by a single function, the kernel function K, is called the kernel trick
- It implicitly maps the data to a (most often) high dimensional feature space
- The practical computational complexity does not change (except for computing the kernel function)

TU Delft

Popular kernel functions



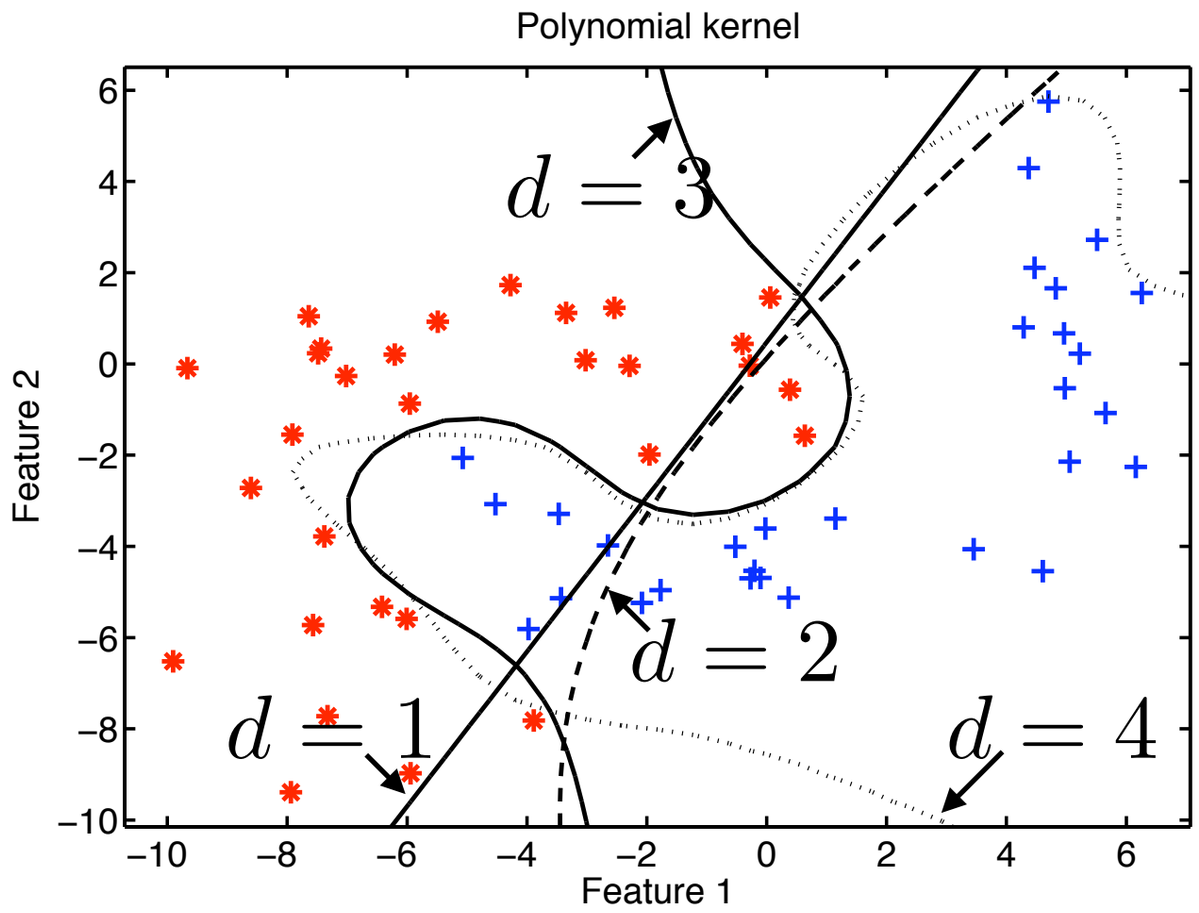


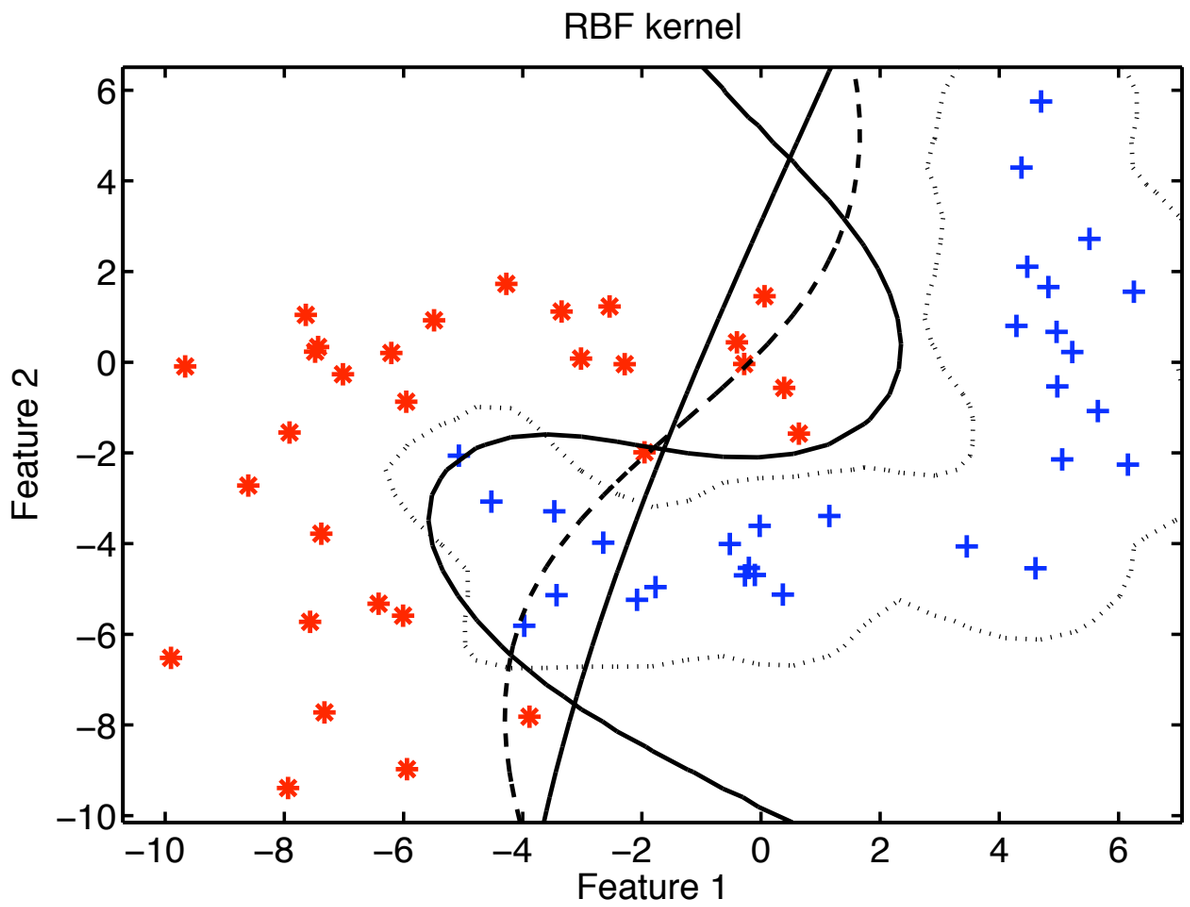
$$K(\mathbf{x}, \mathbf{y}) = (\mathbf{x}^T \mathbf{y} + 1)^d$$

$$K(\mathbf{x}, \mathbf{y}) = \exp\left(-\frac{\|\mathbf{x} - \mathbf{y}\|^2}{\sigma^2}\right)$$

TU Delft

Popular kernel functions



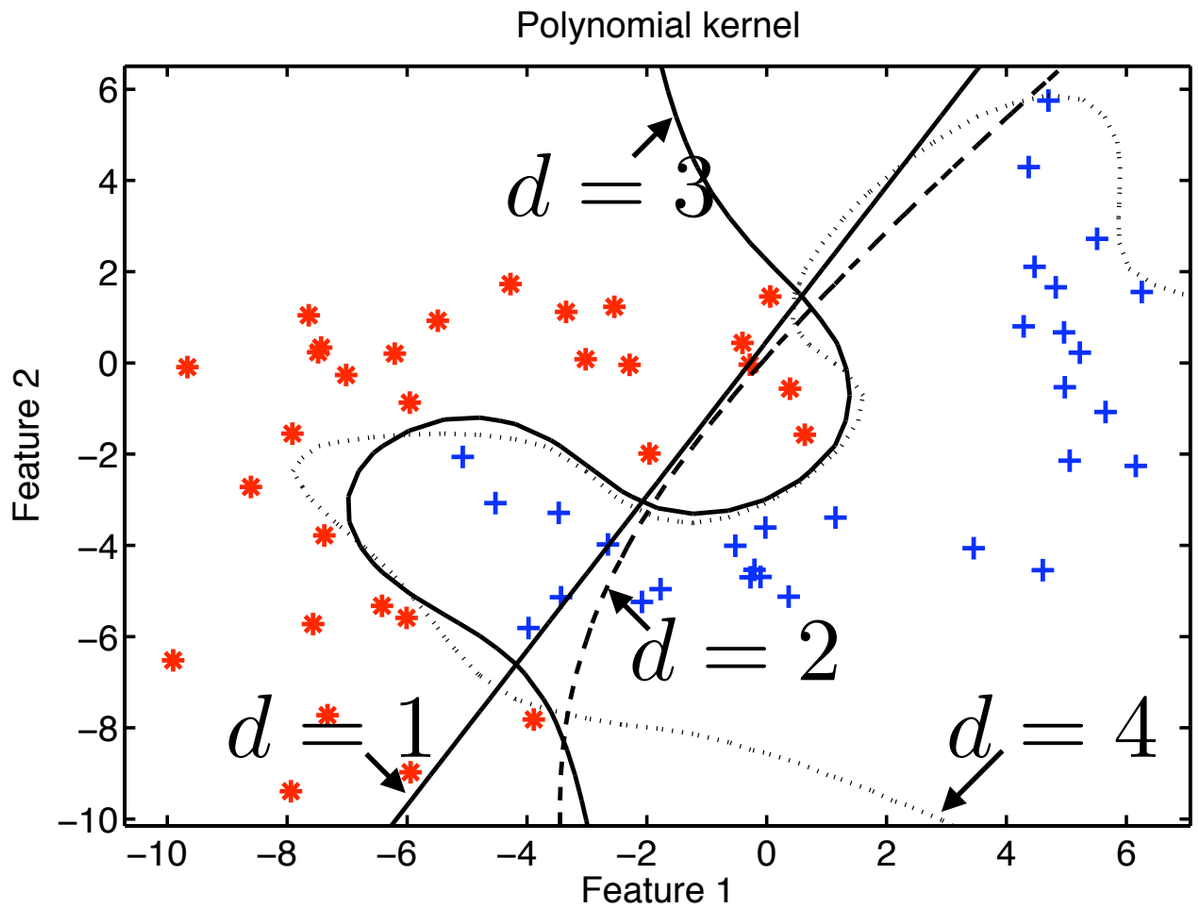


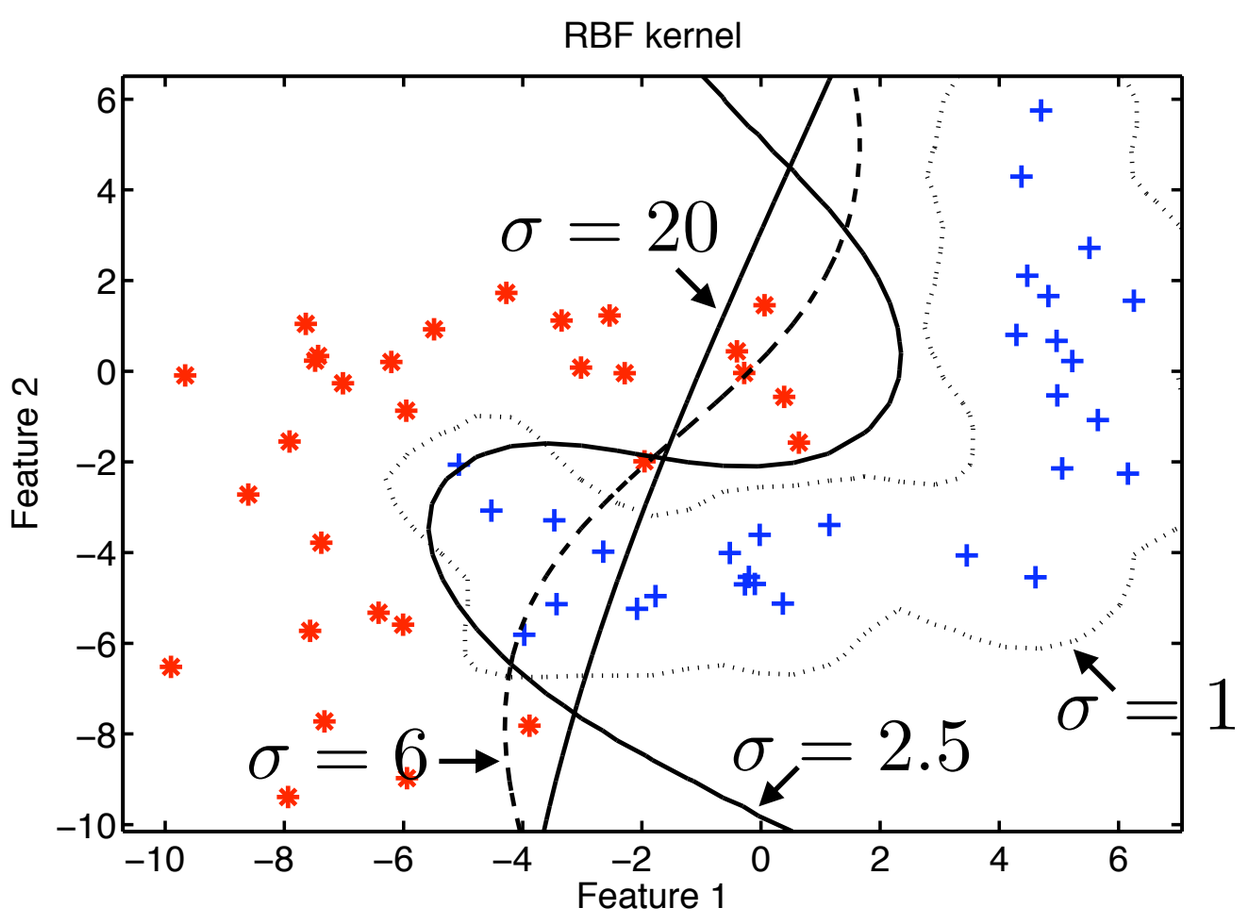
$$K(\mathbf{x}, \mathbf{y}) = (\mathbf{x}^T \mathbf{y} + 1)^d$$

$$K(\mathbf{x}, \mathbf{y}) = \exp\left(-\frac{\|\mathbf{x} - \mathbf{y}\|^2}{\sigma^2}\right)$$

TU Delft

Popular kernel functions



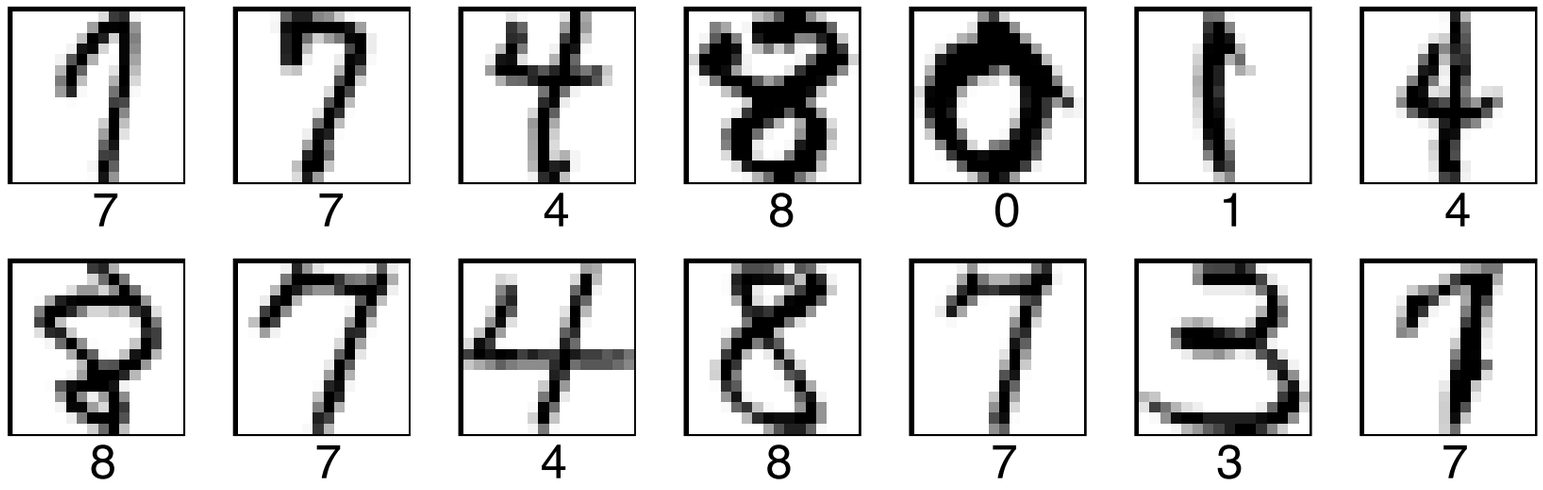


$$K(\mathbf{x}, \mathbf{y}) = (\mathbf{x}^T \mathbf{y} + 1)^d$$

$$K(\mathbf{x}, \mathbf{y}) = \exp\left(-\frac{\|\mathbf{x} - \mathbf{y}\|^2}{\sigma^2}\right)$$

TU Delft

Results



USPS handwritten digits dataset

Cortes & Vapnik, Machine Learning 1995

Classifier	raw error, %
Human performance	2.5
Decision tree, CART	17
Decision tree, C4.5	16
Best 2 layer neural network	6.6
Special architecture 5 layer network	5.1

degree of polynomial	raw error, %	support vectors	dimensionality of feature space
1	12.0	200	256
2	4.7	127	~ 33000
3	4.4	148	$\sim 1 \times 10^6$
4	4.3	165	$\sim 1 \times 10^9$
5	4.3	175	$\sim 1 \times 10^{12}$
6	4.2	185	$\sim 1 \times 10^{14}$
7	4.3	190	$\sim 1 \times 10^{16}$

TU Delft

Advanced kernel functions

- People construct special kernel functions for applications:
  - Tangent distance kernels to incorporate rotation/ scaling insensitivity (handwritten digits recognition)
  - String matching kernels to classify DNA/protein sequences
  - Fisher kernels to incorporate knowledge on class densities
  - Hausdorff kernels to compare image blobs
  - …

TU Delft

High dimensional feature spaces

- SVM appears to work well in high dimensional feature spaces
  - The class overlap is often not large
  - Minimizing h minimizes the risk of overfitting
  - The classifier is determined by the support objects and not directly by the features
  - No density is estimated

TU Delft

Advantages of SVM

- The SVM generalizes remarkably well, in particular in high-dimensional feature spaces with (relatively) low sample sizes
- Given a kernel and a C, there is one unique solution
- The kernel trick allows for a varying complexity of the classifier
- The kernel trick allows for especially engineered representations for problems

- No strict data model is assumed (when you can assume it, use it!)
- The foundation of the SVM is pretty solid (when no slack variables are used)
- An error estimate is available using just the training data (but it is a pretty loose estimate, and crossvalidation is still required to optimize K and C)

TU Delft

Disadvantages of SVM

- The quadratic optimization is computationally expensive (although more and more specialized optimizers appear)
- The kernel and C have to be optimized
- The SVM has problems with highly overlapping classes

TU Delft

Conclusions

- Always tune the complexity of your classifier to your data (#training objects, dimensionality, class overlap, class shape...)

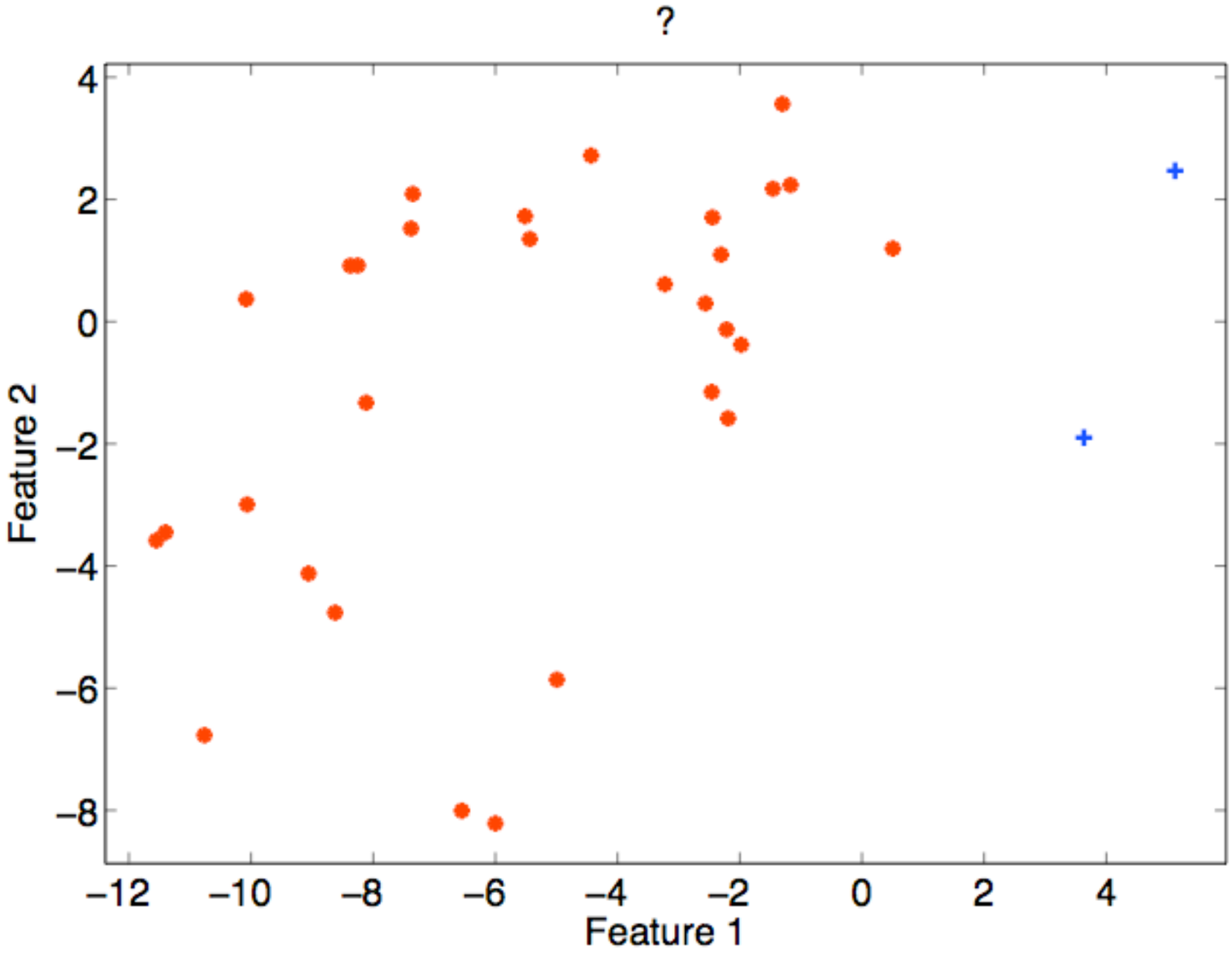
- 'Avoid solving a more general intermediate problem, when you have limited data' can be implemented by the SVM: no need to estimate (class-) densities

- The SVM can be used in almost all applications (by adjusting K and C appropriately)

TU Delft

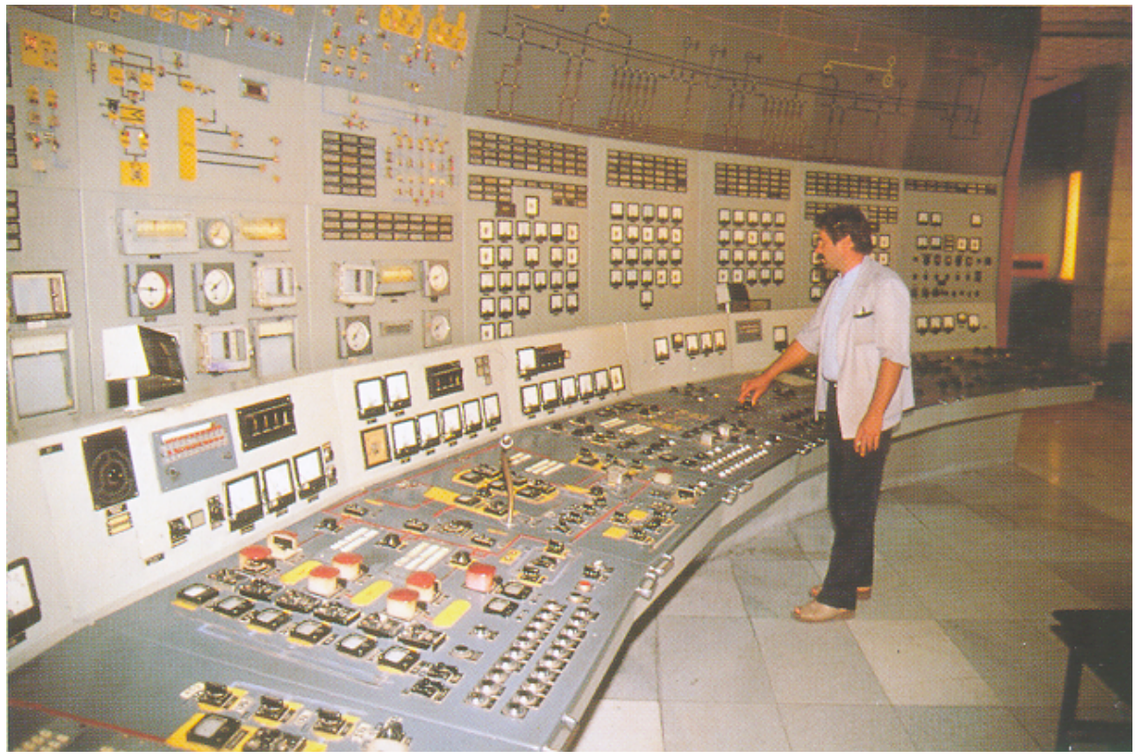
Classification problem

BONUS



TU Delft

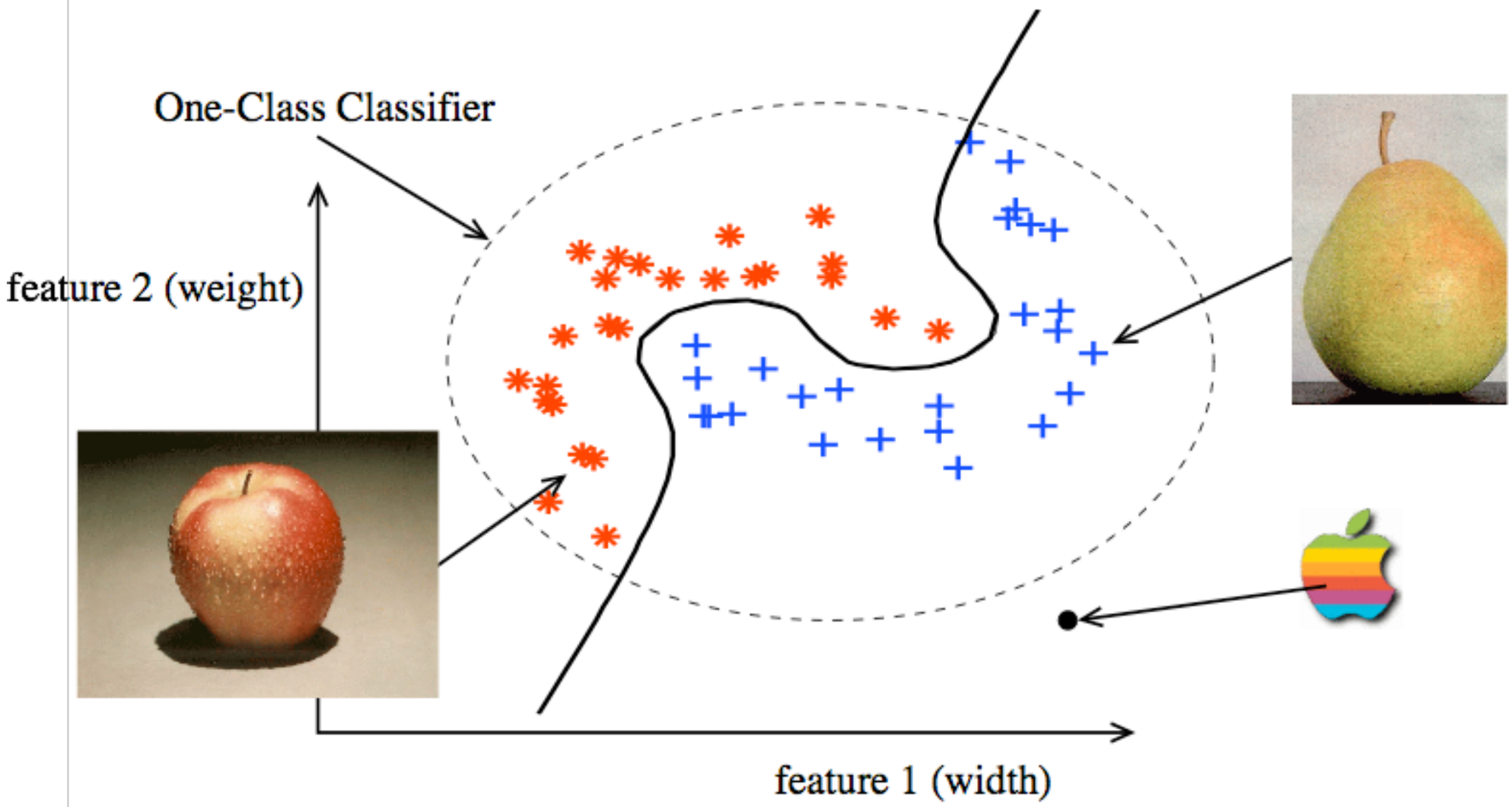
One-class classification



- Normal operating condition (target class) can be sampled well
- Abnormal conditions (outlier class) are rare, or are hard to sample reliably

TU Delft

One-class classification



- Many example target objects are present
- Few (reliable) outlier objects are available

TU Delft

From classifiers to outlier detection

$$p(y|\mathbf{x})$$

generative classifiers:

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

$$\hat{p}(\mathbf{x}) = \sum_i \hat{p}(\mathbf{x}|y_i)\hat{p}(y_i)$$

discriminative classifiers:

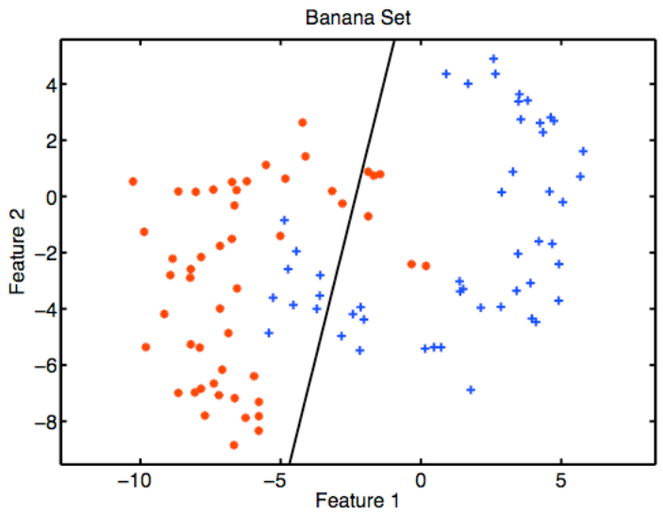
$$\hat{p}(y|\mathbf{x}) = f(\mathbf{x};\mathbf{w})$$

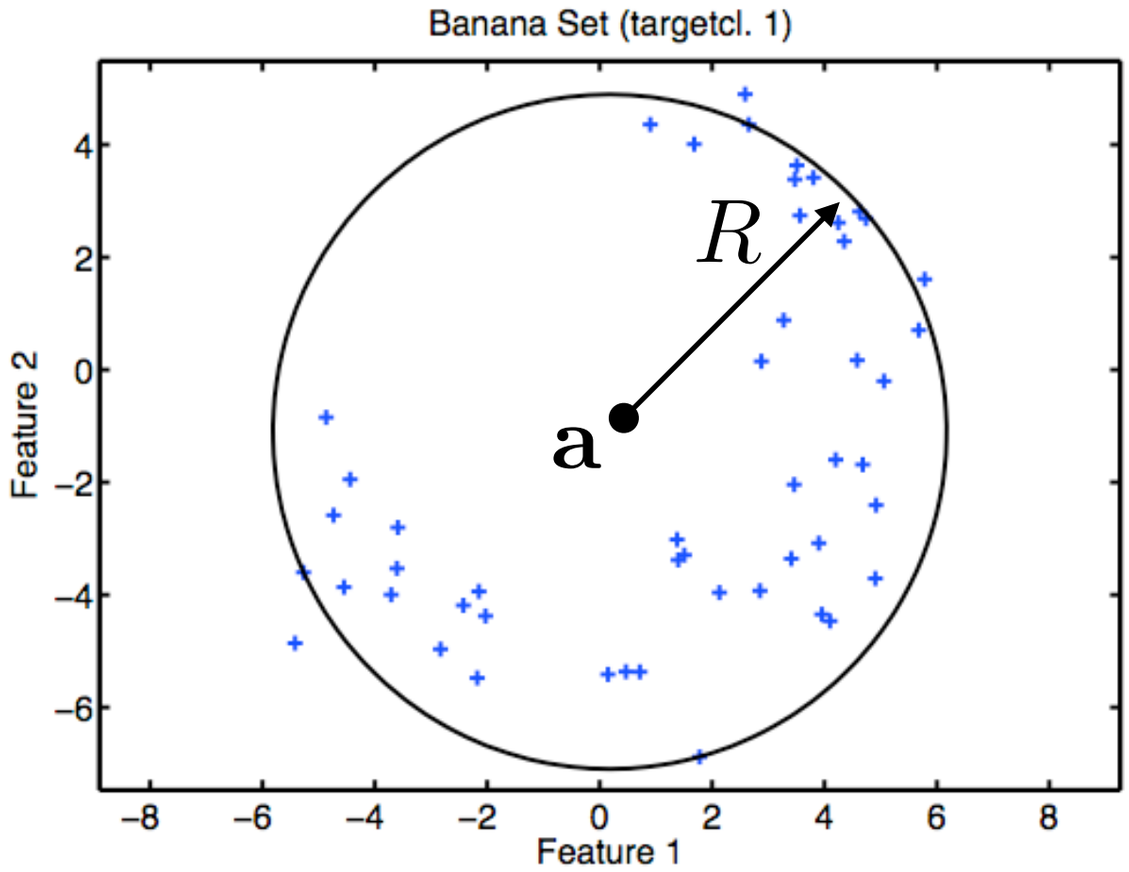
?

- Outlier detection without density estimation?

TU Delft

Decision boundary OCC





- Fit only a boundary (inspired by the support vector classifier)
- Instead of a linear decision boundary, a hypersphere around the target class

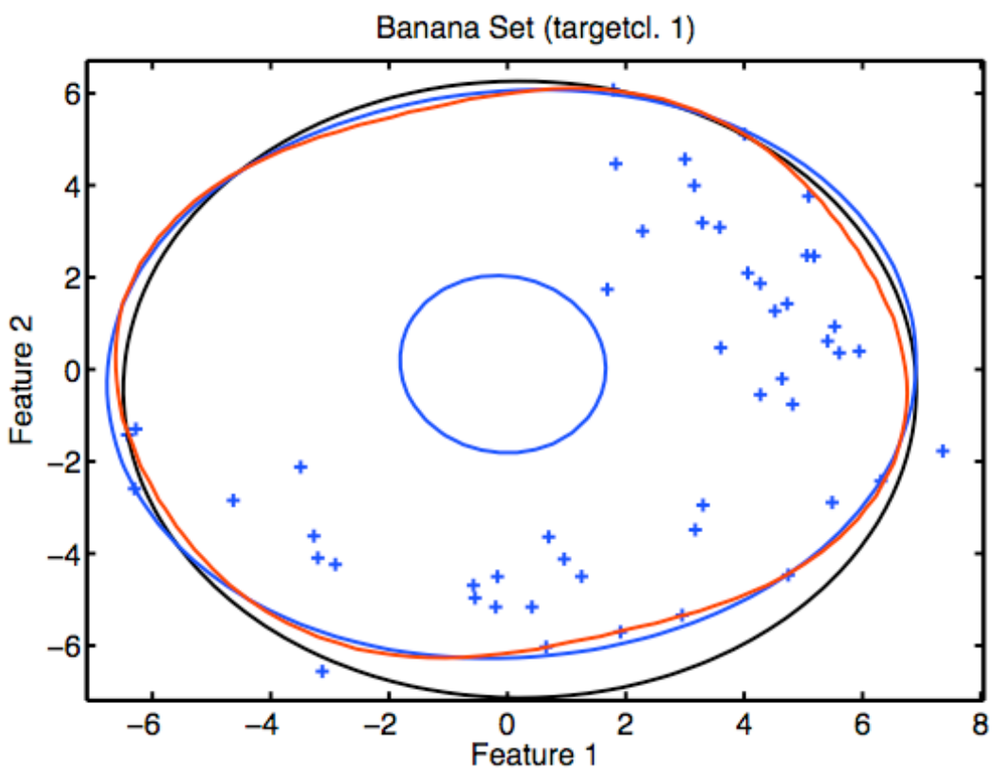
TU Delft

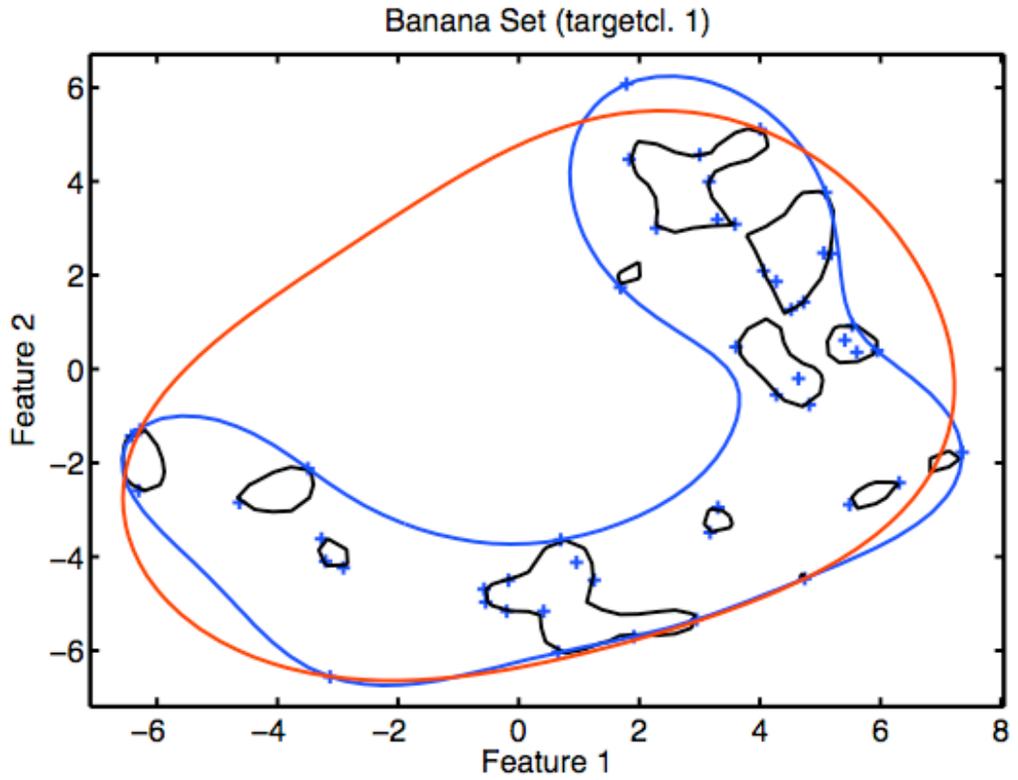
Support Vector Data Description

	Support Vector cl.	**Support vector DD**
model	hyperplane $\mathbf{w}, b$	hypersphere $\mathbf{a}, R$
complexity	$\|\mathbf{w}\|^2$	R^2
error	$\|\mathbf{w}\|^2 + C\sum_i \xi_i$	$R^2 + C\sum_i \xi_i$
SVs	objects on the plane	objects on the sphere
slacks	objects on the wrong side of the plane	objects outside the sphere

TU Delft

Different kernels





Polynomial

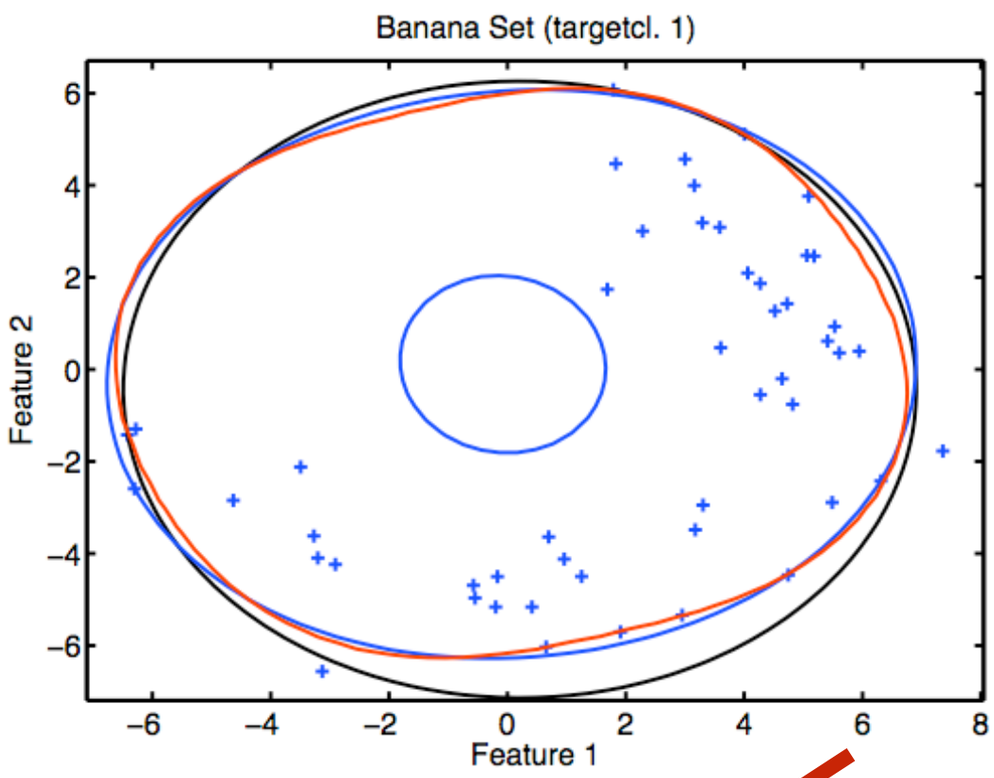
$$K(\mathbf{z}, \mathbf{x}) = (\mathbf{x}^T \mathbf{z} + 1)^d$$

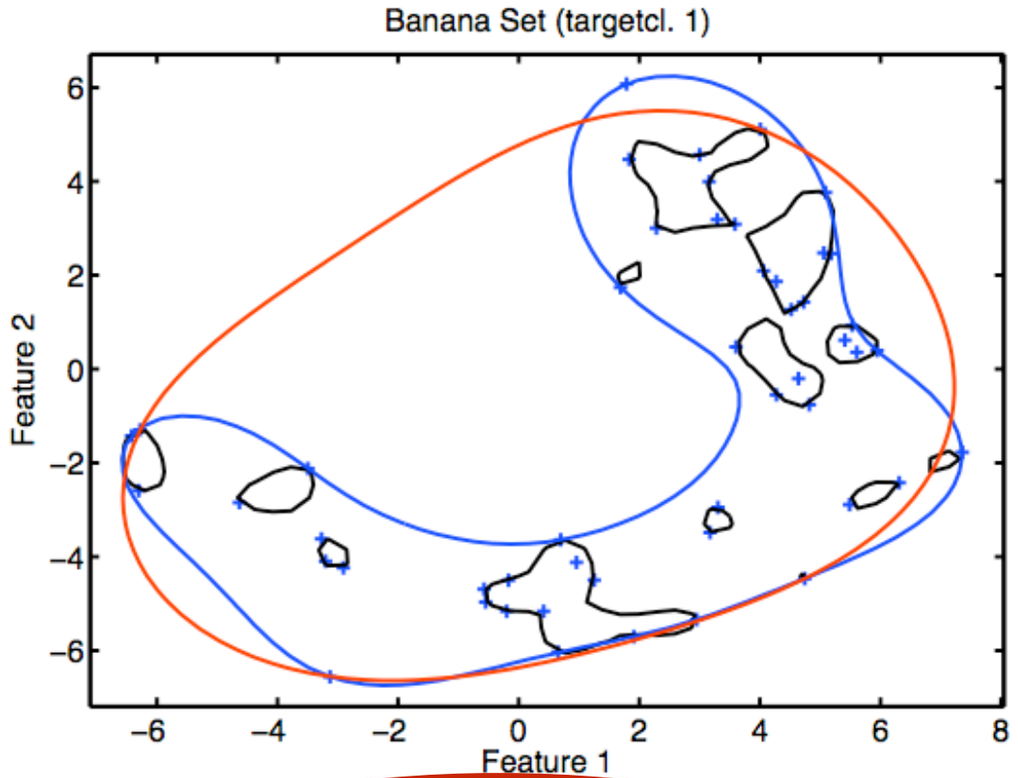
Radial basis (Gaussian)

$$K(\mathbf{z}, \mathbf{x}) = \exp\left(-\frac{\|\mathbf{x} - \mathbf{z}\|^2}{\sigma^2}\right)$$

TU Delft

Different kernels





Polynomial

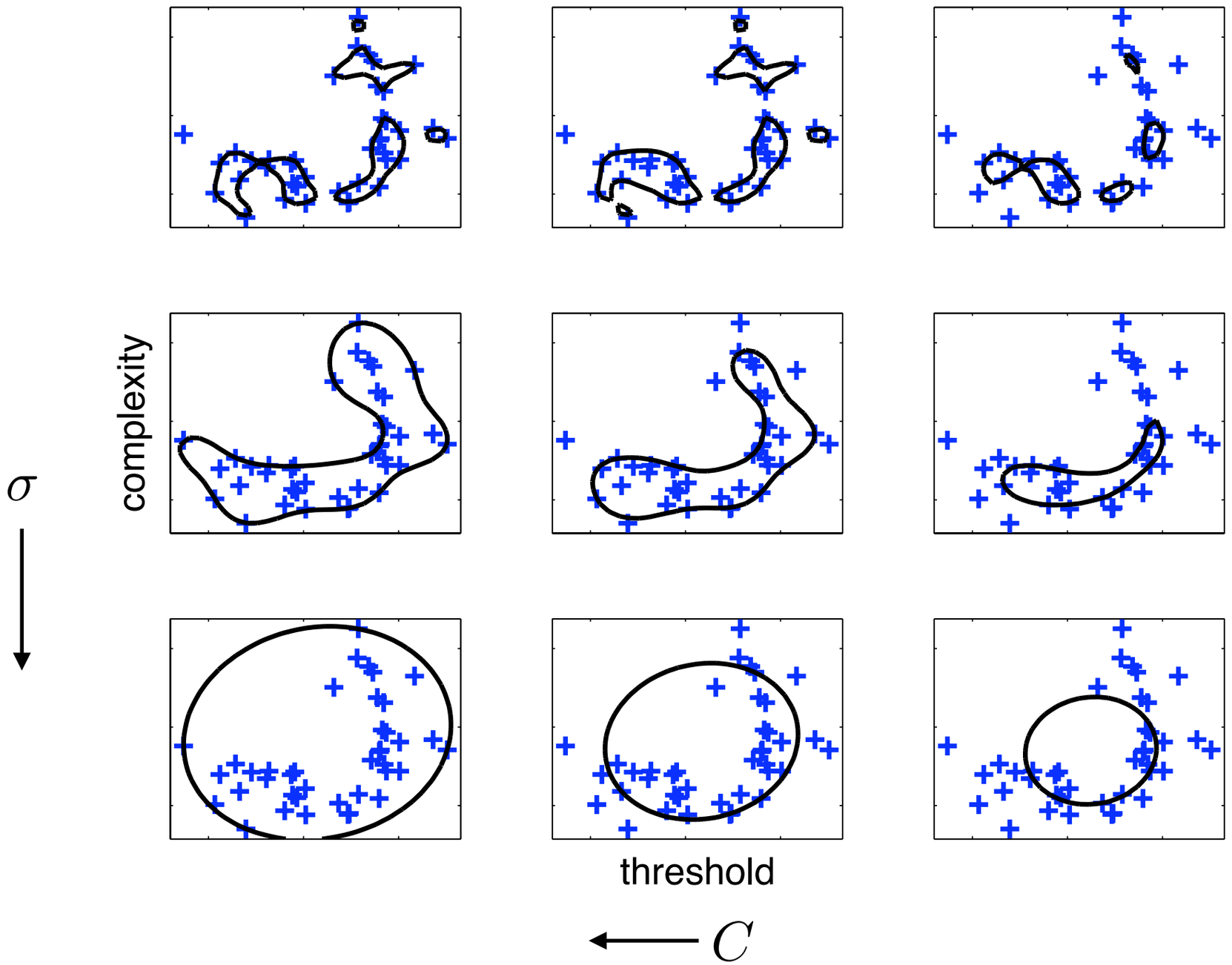
$$K(\mathbf{z},\mathbf{x}) = (\mathbf{x}^T\mathbf{z} + 1)^d$$

Radial basis (Gaussian)

$$K(\mathbf{z},\mathbf{x}) = \exp\left(-\frac{\|\mathbf{x}-\mathbf{z}\|^2}{\sigma^2}\right)$$

TU Delft

Complexity and threshold



TU Delft

Bonus conclusions

- Avoiding density estimations is also useful in other applications (outlier detection, regression)
- Requires a formulation where (training data) only occurs in the form of inner products

TU Delft