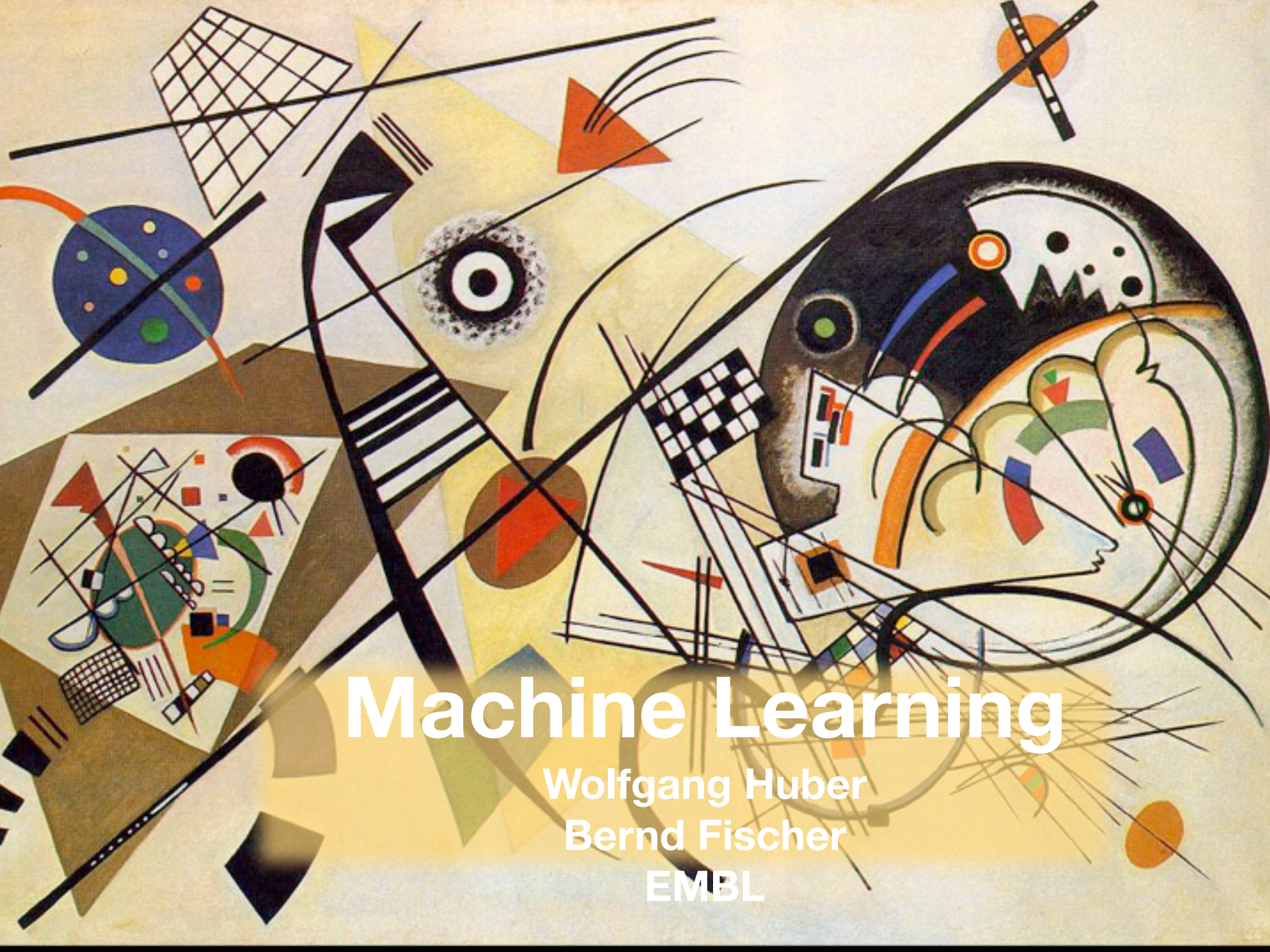


Machine Learning

Wolfgang Huber
Bernd Fischer
EMBL

What you will learn in this lecture

Motivating examples

Multivariate classification: least squares, support vector

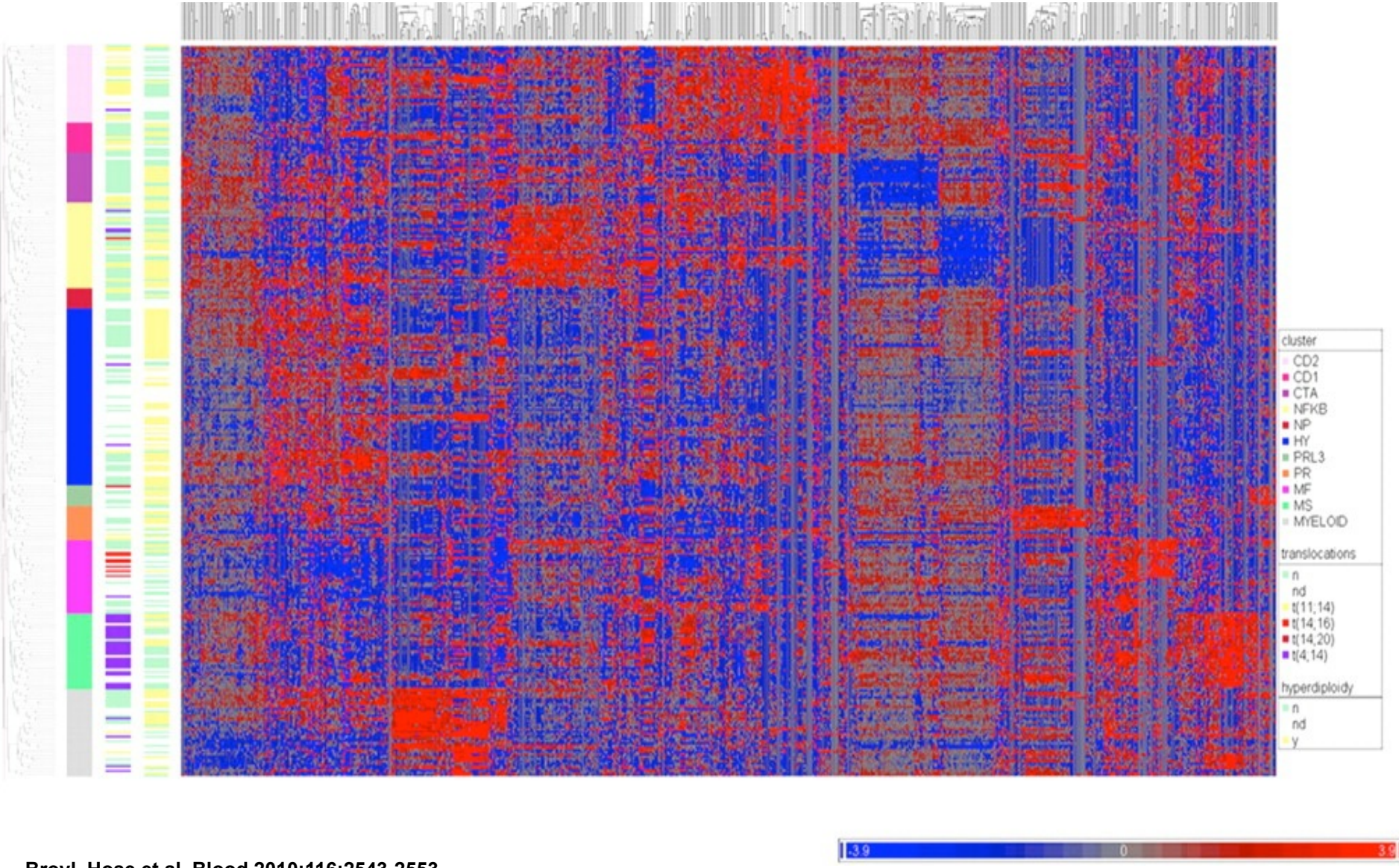
Model complexity - ‘overfitting’

Cross-validation

Kernel trick

Regularisation, Lasso & Co.

Gene expression profiling for molecular classification of multiple myeloma in newly diagnosed patients



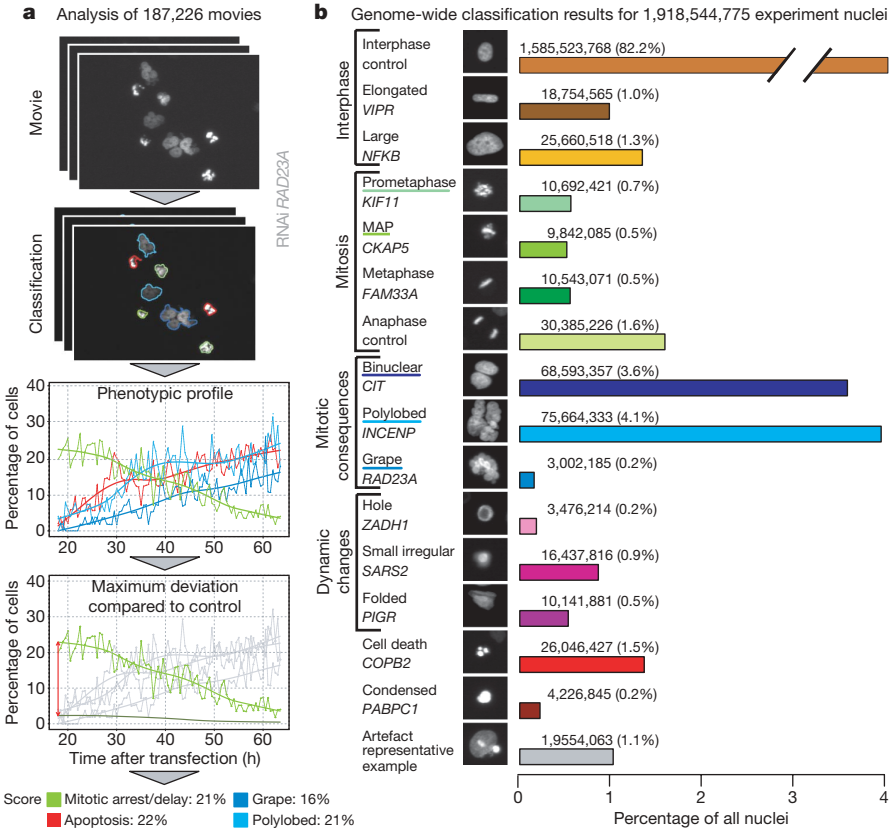
Broyl, Hose et al. Blood 2010;116:2543-2553

Vol 464|1 April 2010|doi:10.1038/nature08869

ARTICLES

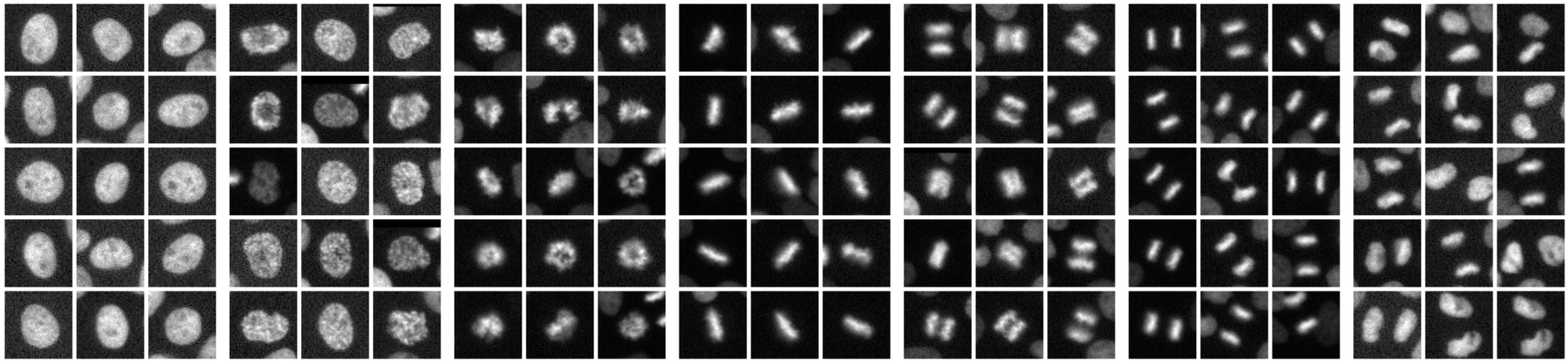
Phenotypic profiling of the human genome by time-lapse microscopy reveals cell division genes

Beate Neumann[1]*, Thomas Walter[1]*, Jean-Karim Hériché[5]†, Jutta Bulkescher[1], Holger Erfle[1,3]†, Christian Conrad[1,3], Phill Rogers[1]†, Ina Poser[6], Michael Held[1]†, Urban Liebel[1]† Gregoire Pau[9], Rolf Kabbe[10], Annelie Wünsche[2], Venkata Satagopam[4], Michael Daniel W. Gerlich[7], Reinhard Schneider[4], Roland Eils[10], Wolfgang Huber[9], Jan-Anthony A. Hyman[6], Richard Durbin[5], Rainer Pepperkok[3] & Jan Ellenberg[2]

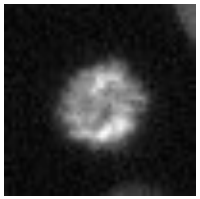


Morphological Phenotyping

- Provide Human Annotation to a small set of cells:



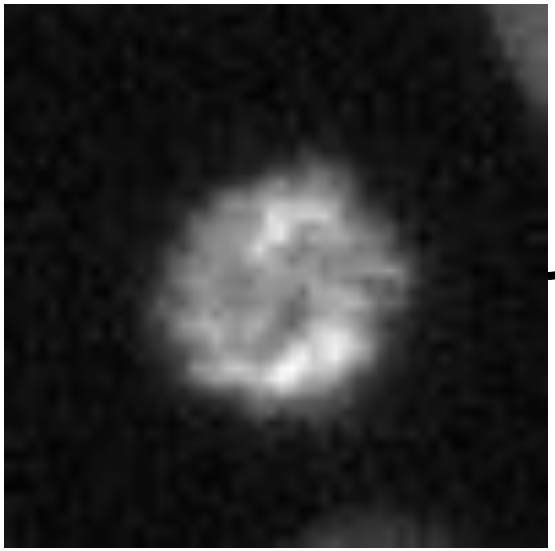
inter pro prometa meta earlyana lateana telo



Which mitotic phase?

(Annotate automatically!)

Automatic Classification Workflow



Preprocessing
e.g. normalization, background subtraction, …

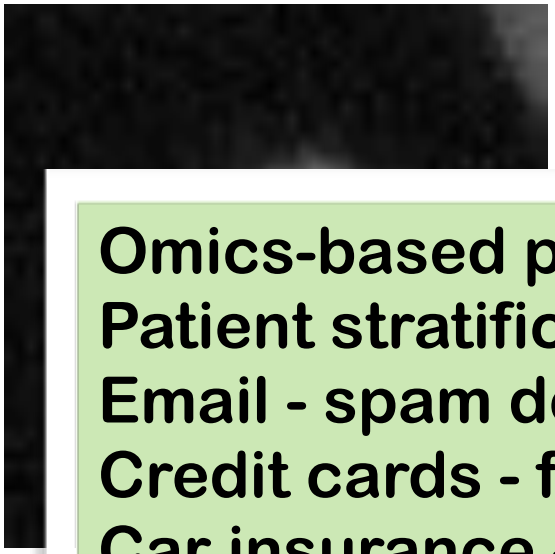
Feature Extraction
e.g. lightness, nucleus area, excentricity, …

Classification

Prophase

Metaphase

Automatic Classification Workflow



Omics-based pathology
Patient stratification
Email - spam detection
Credit cards - fraud
Car insurance - rates
....

Preprocessing
e.g. normalization, background subtraction, …

…action
…ess, nucleus area, excentricity, …

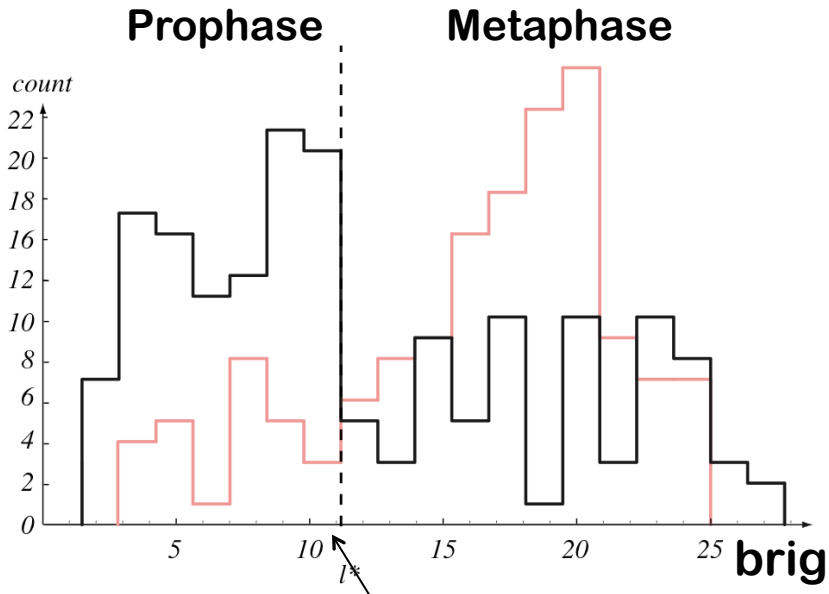
Classification

Prophase

Metaphase

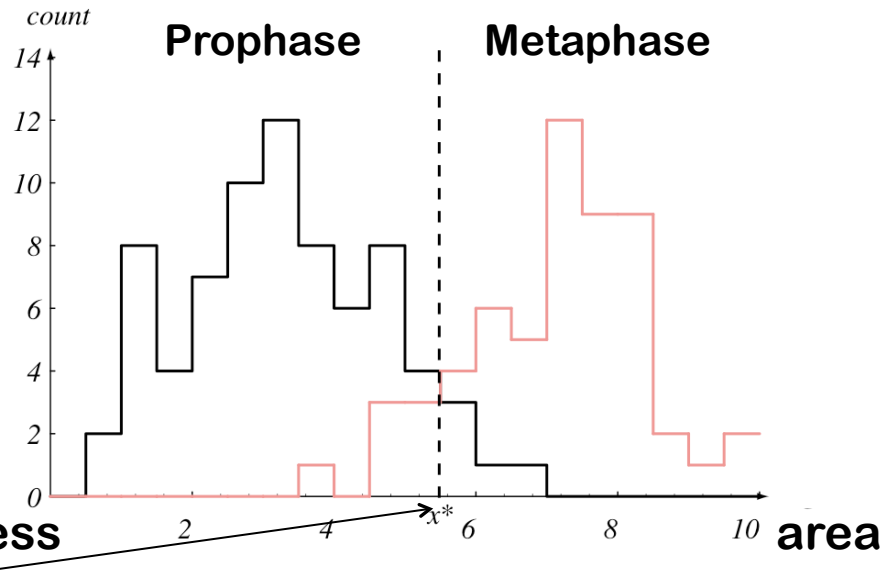
Prophase/ Metaphase Classification

Predict mitotic state based on brightness



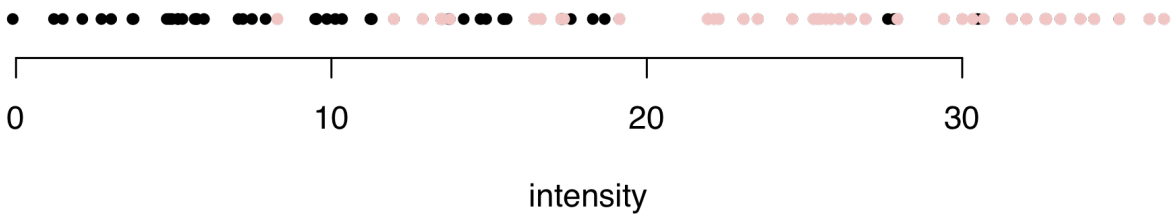
Decision boundary with lowest prediction error

Predict mitotic state based on nucleus area

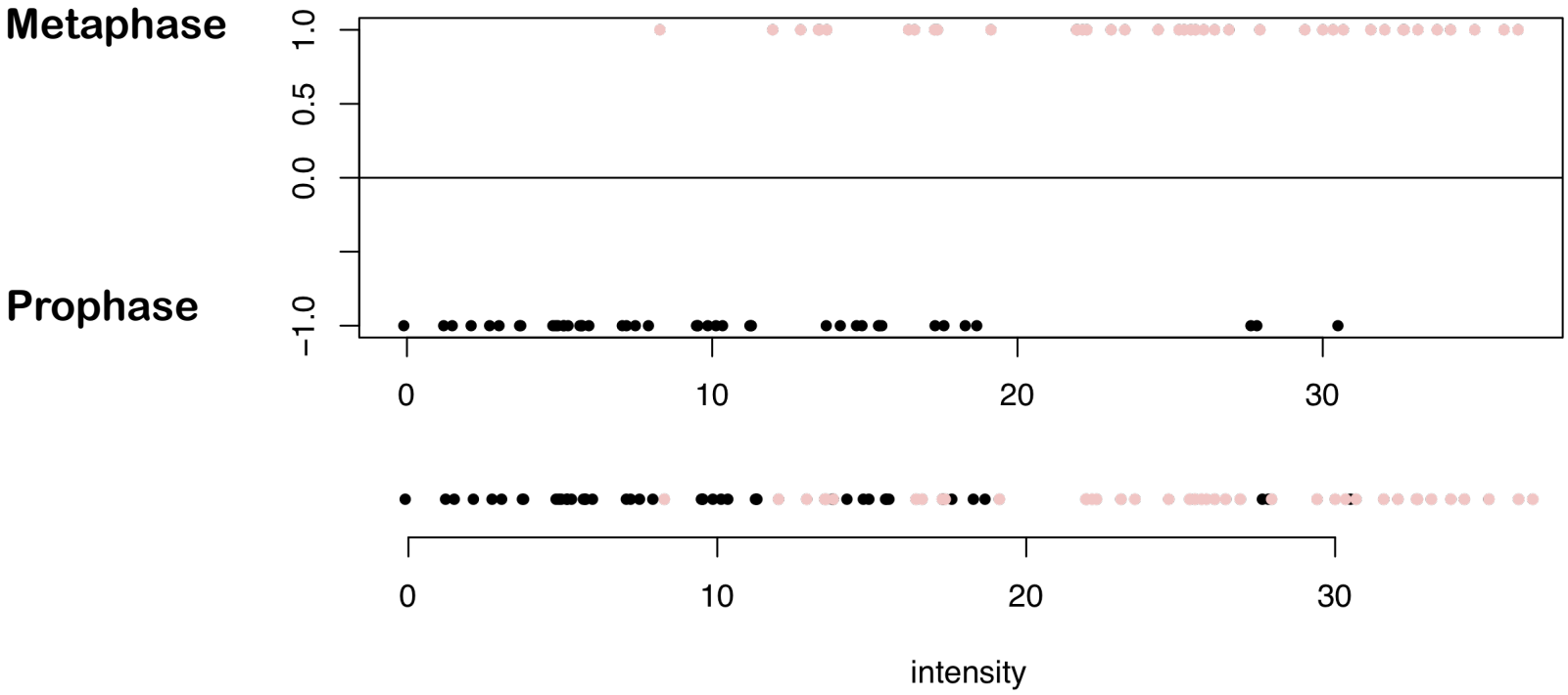


None of the two features individually has a good predictive power

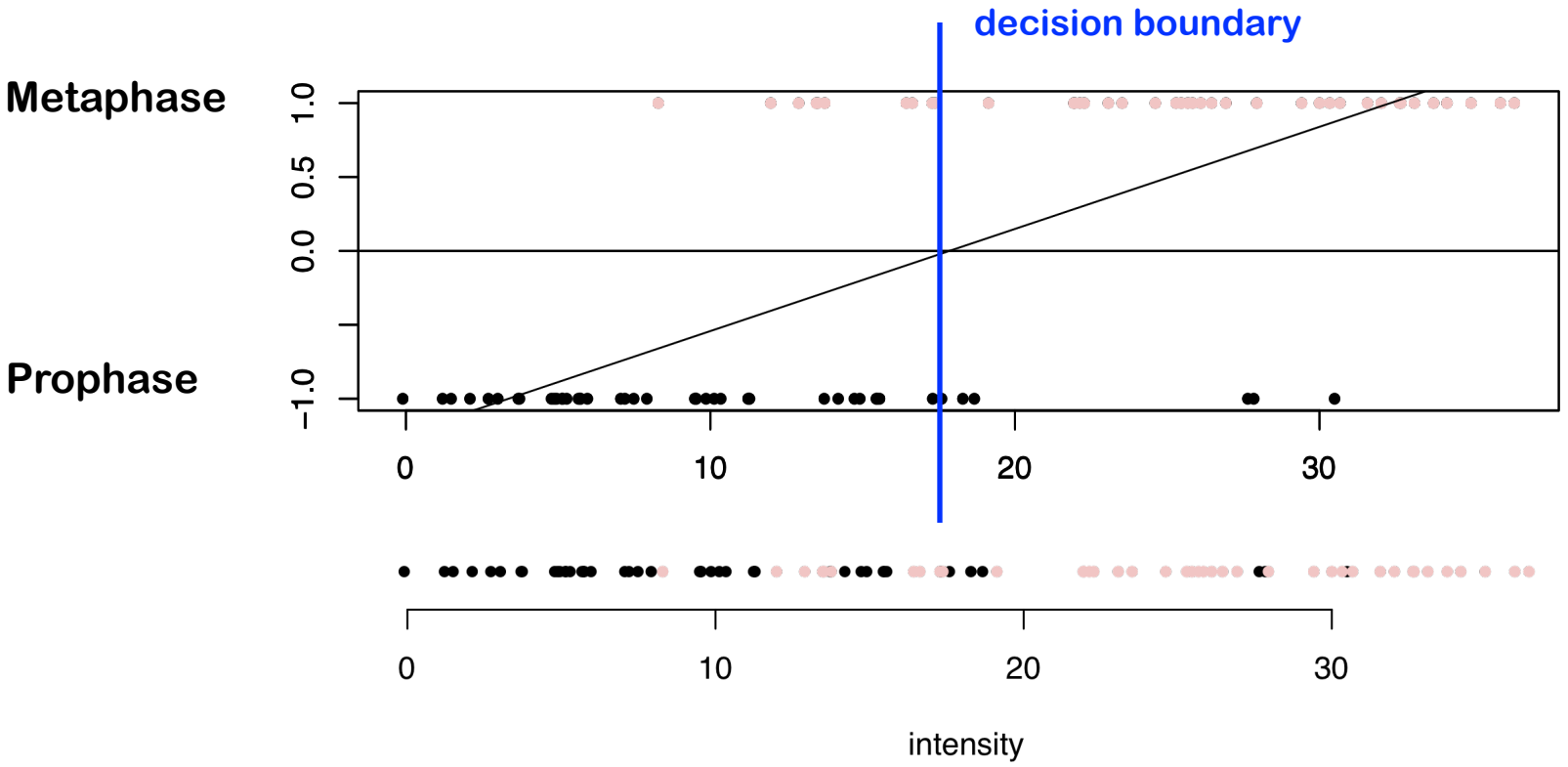
A Simple Least Squares Classifier: d=1



A Simple Least Squares Classifier: d=1



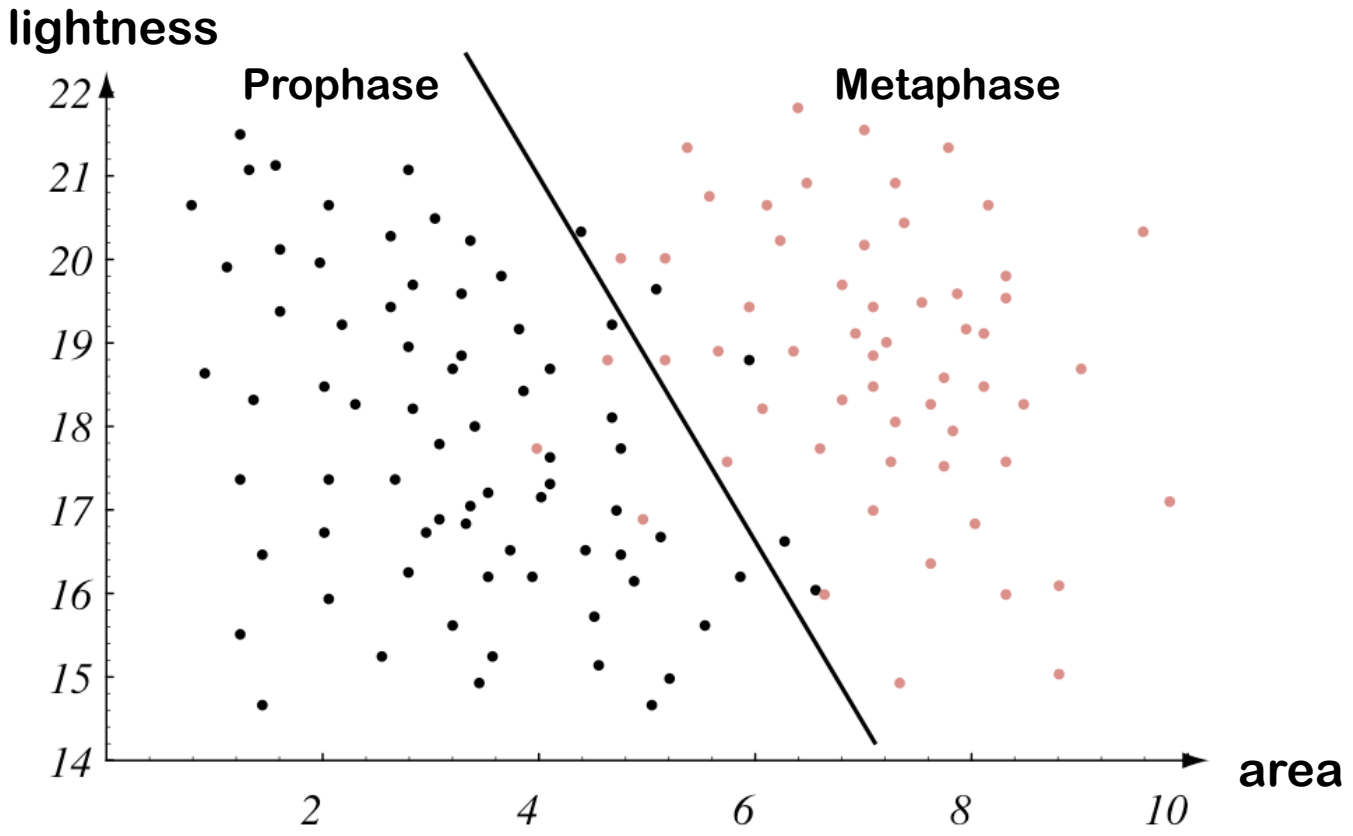
A Simple Least Squares Classifier: d=1




```
y[i] = -1 for prophase
y[i] = +1 for metaphase
X[i,] = c(area[i],intensity[i])
model = lm(y ~ X)
ynew = predict(model, newdata=newX)
ifelse(ynew < 0,-1,1)
```

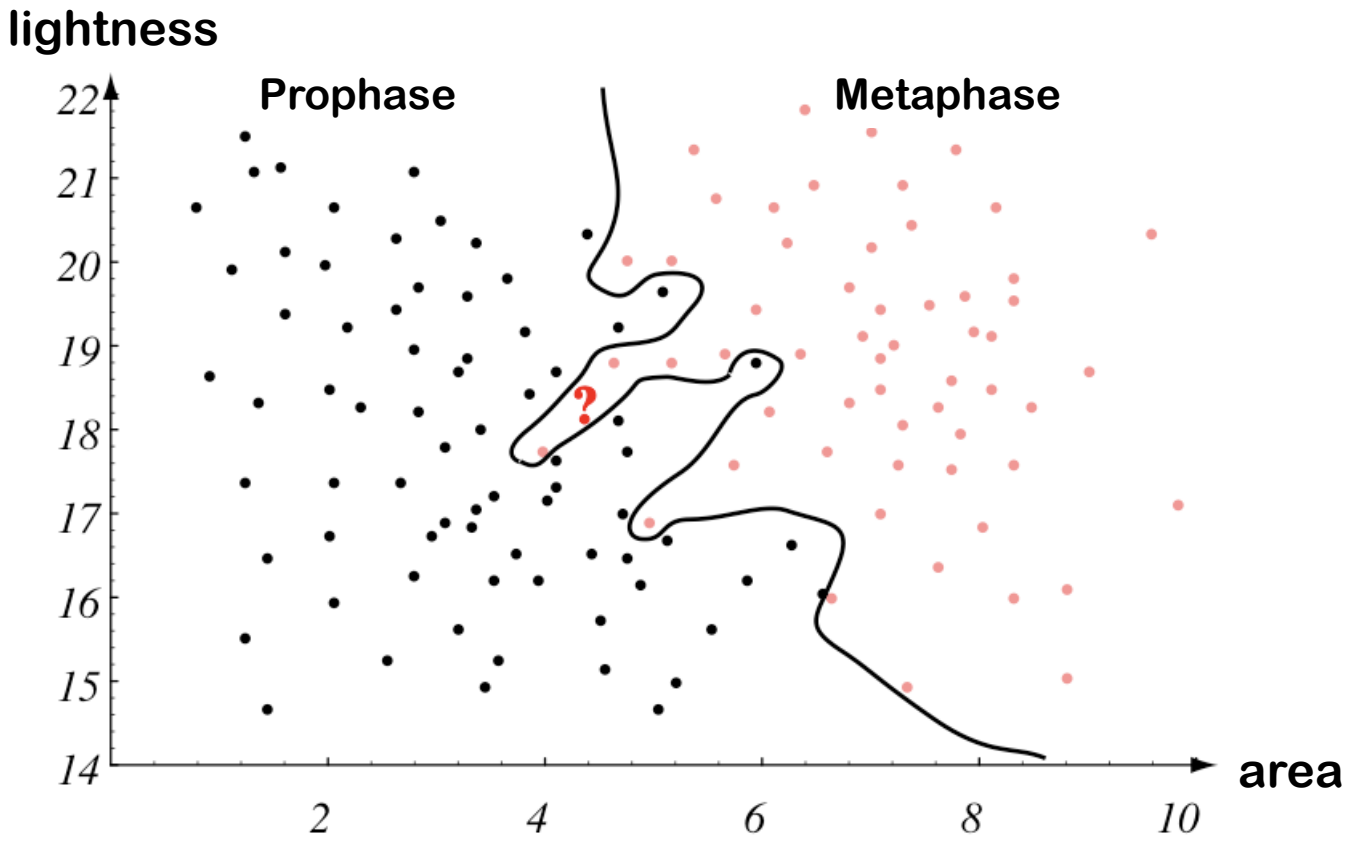
$$\Sigma_i (y_i - \beta x_i)^2 \rightarrow \min$$

A Simple Least Squares Classifier: d=2




```
y[i]=+1 for prophase
y[i]=-1 for metaphase
X[i,]=(area[i],lightness[i])
model <- lm.fit(X,y)
ynew <- predict(model,Xnew)
            $fitted.values
ifelse(ynew < 0,-1,1)
```

k-Nearest-Neighbor Classifier



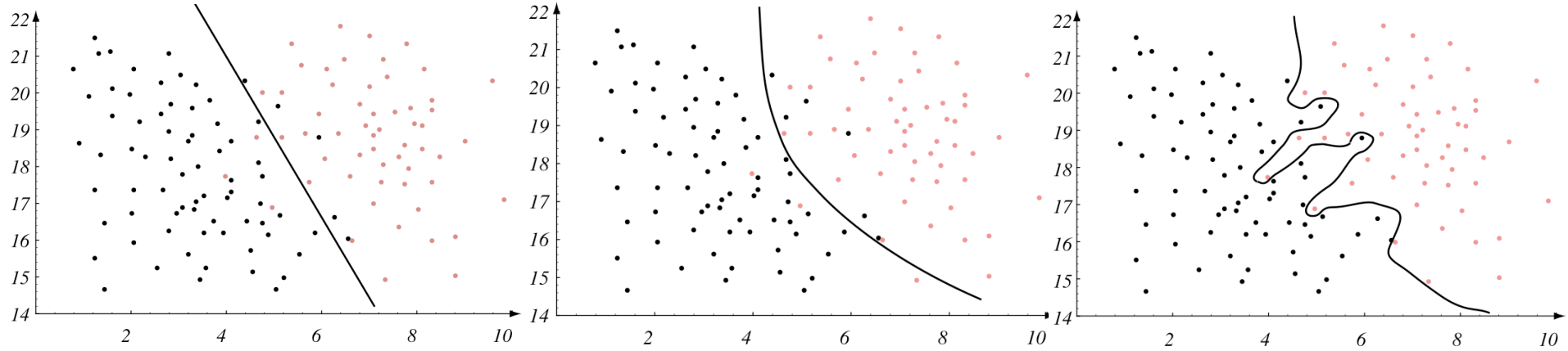
Assign each new cell to the class of its nearest neighbor.
Black line shows decision boundary

```
y[i]=+1 for pro phase
y[i]=-1 for meta phase
X[i,]=(area[i],lightness[i])
library(class)
d = knn(X,Xnew,y,k=1)
```

Which Decision Boundary?

High bias
Low variance

Low bias
High variance

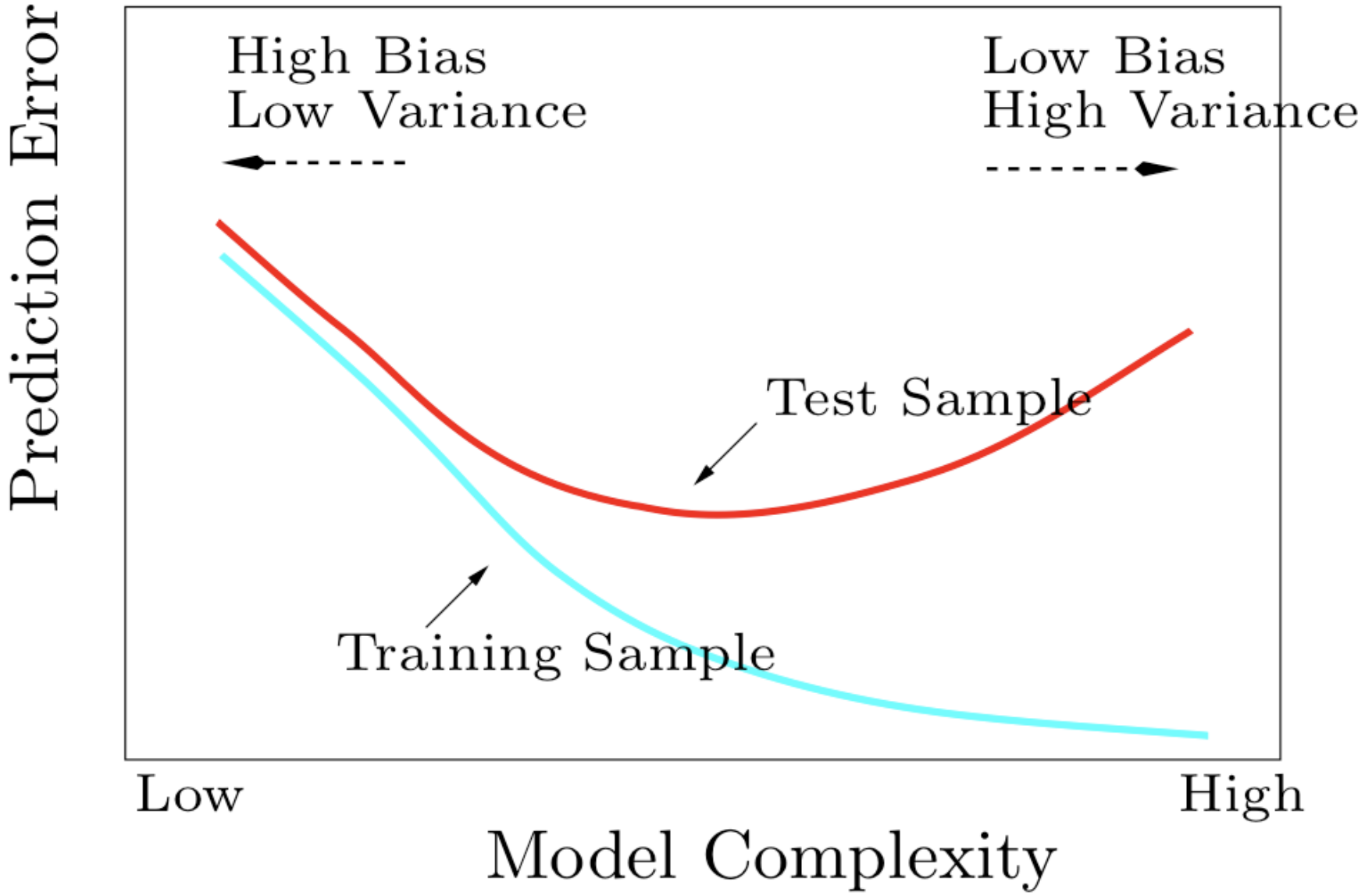


low model complexity
(needs 2 parameters to describe the decision boundary)

high model complexity
(needs hundreds of parameter to describe the decision boundary)

Which decision boundary has the lowest prediction error?

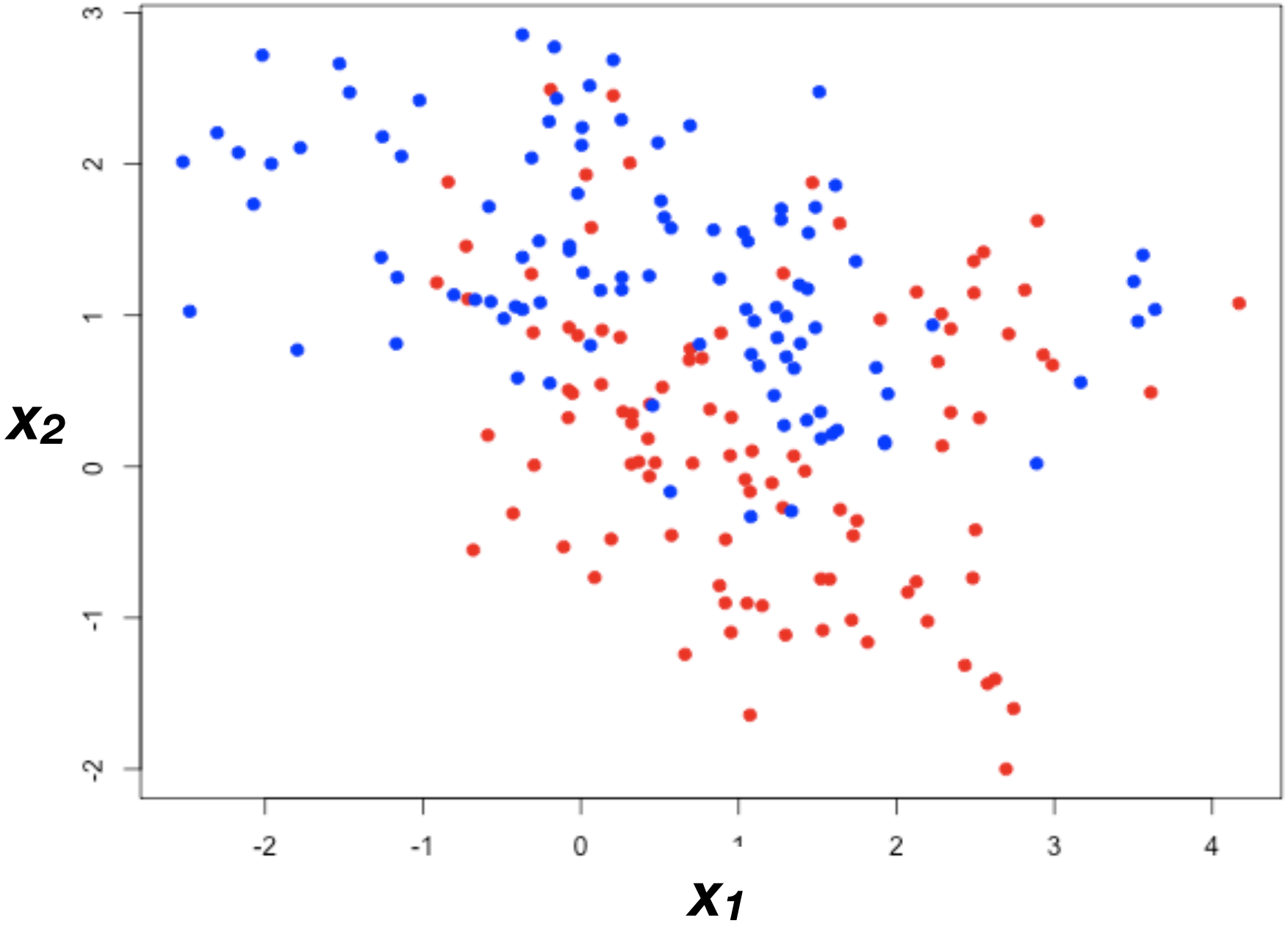
Bias-Variance-Dilemma



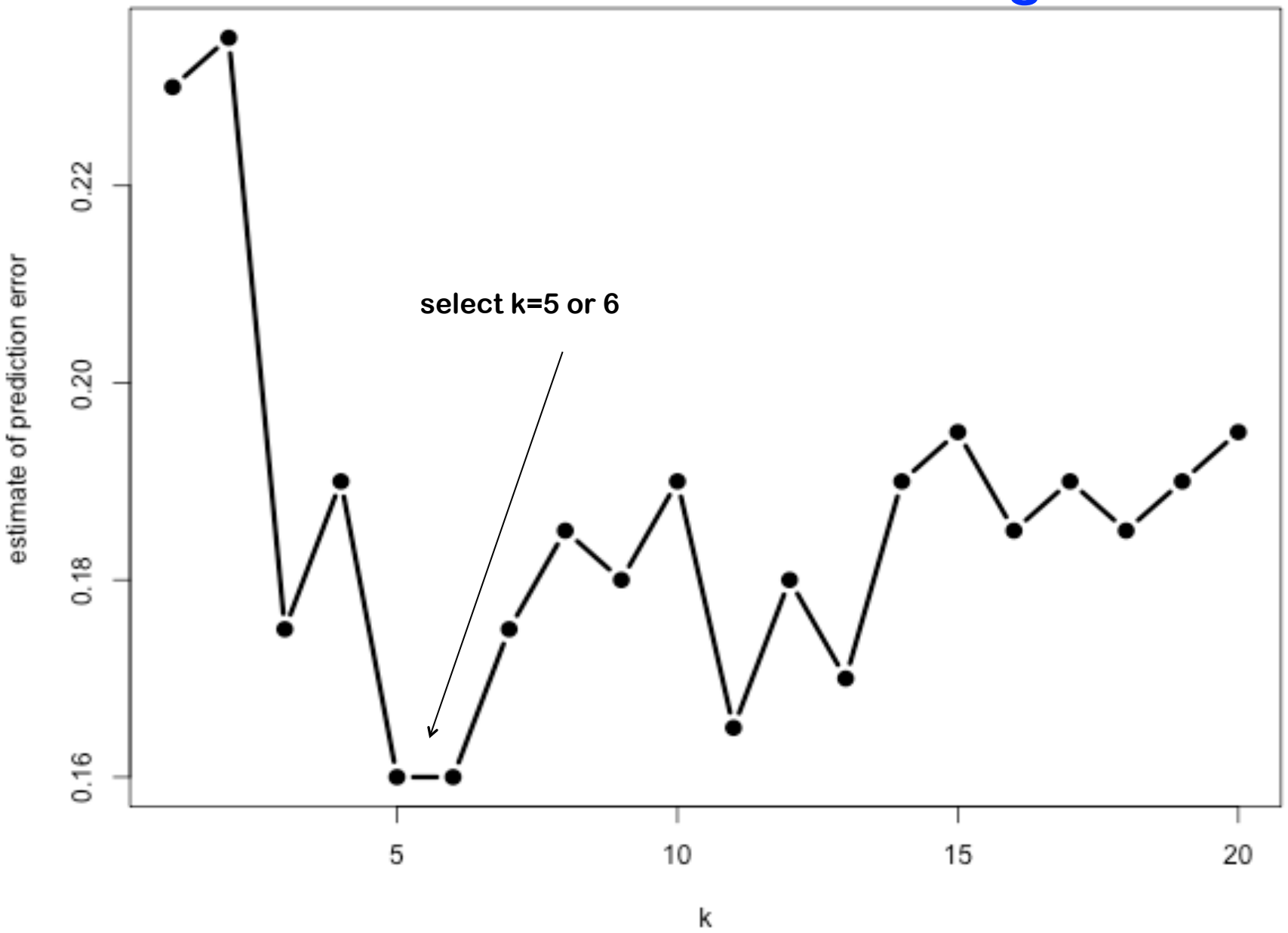
Cross-Validation

- cross validation is an easy & useful method to estimate the prediction error.
- data consist of *n* samples with *d* features and a known class label
- Method (*m*-fold cross-validation):
  - Split the data into m approximately equally sized subsets
  - Train the classifier on (*m*-1) subsets
  - Test the classifier on the remaining subset. Estimate the prediction error by comparing the predicted class label with the true class labels.
  - Repeat *m* times (i.e.: use each subset once as test set)

Example: Two classes, two variables, 200 objects

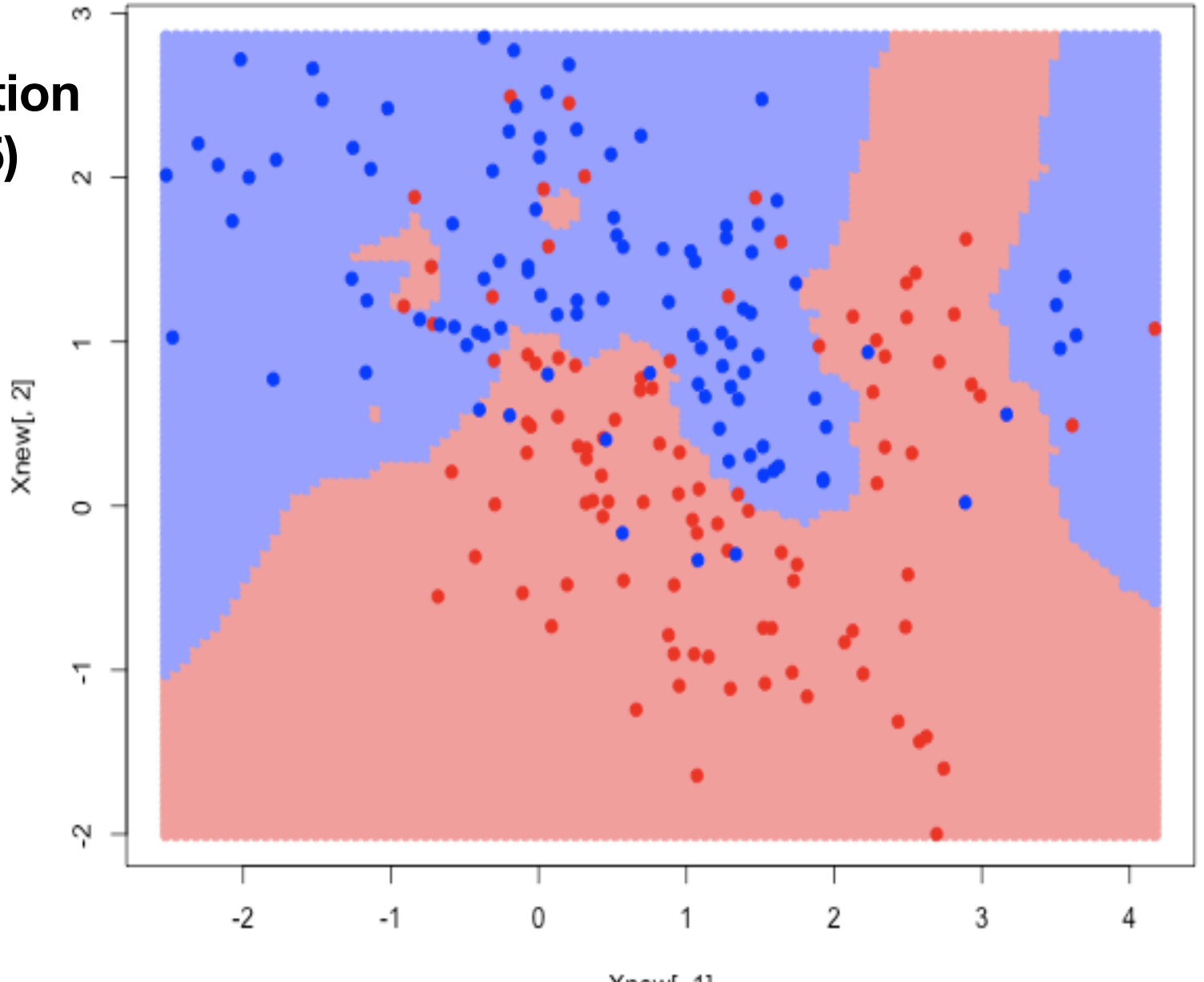


cross-validation for k-nearest neighbours



Demo: Cross-Validation for k-nearest neighbours

Classification result (k=5)



Least Squares Classifier

X: n x d matrix with d-dimensional features for n samples
y: vector of length n: $y_i = 0$ for first class, 1 for second class
Fit linear model by minimizing the squared error:

$$\hat{\beta} = \arg\min_{\beta} \|X\beta - y\|_2^2$$

```
model = lm.fit(X, y)
ynew = predict(model, Xnew)$fitted.values
ifelse(ynew < 0,-1,1)
```

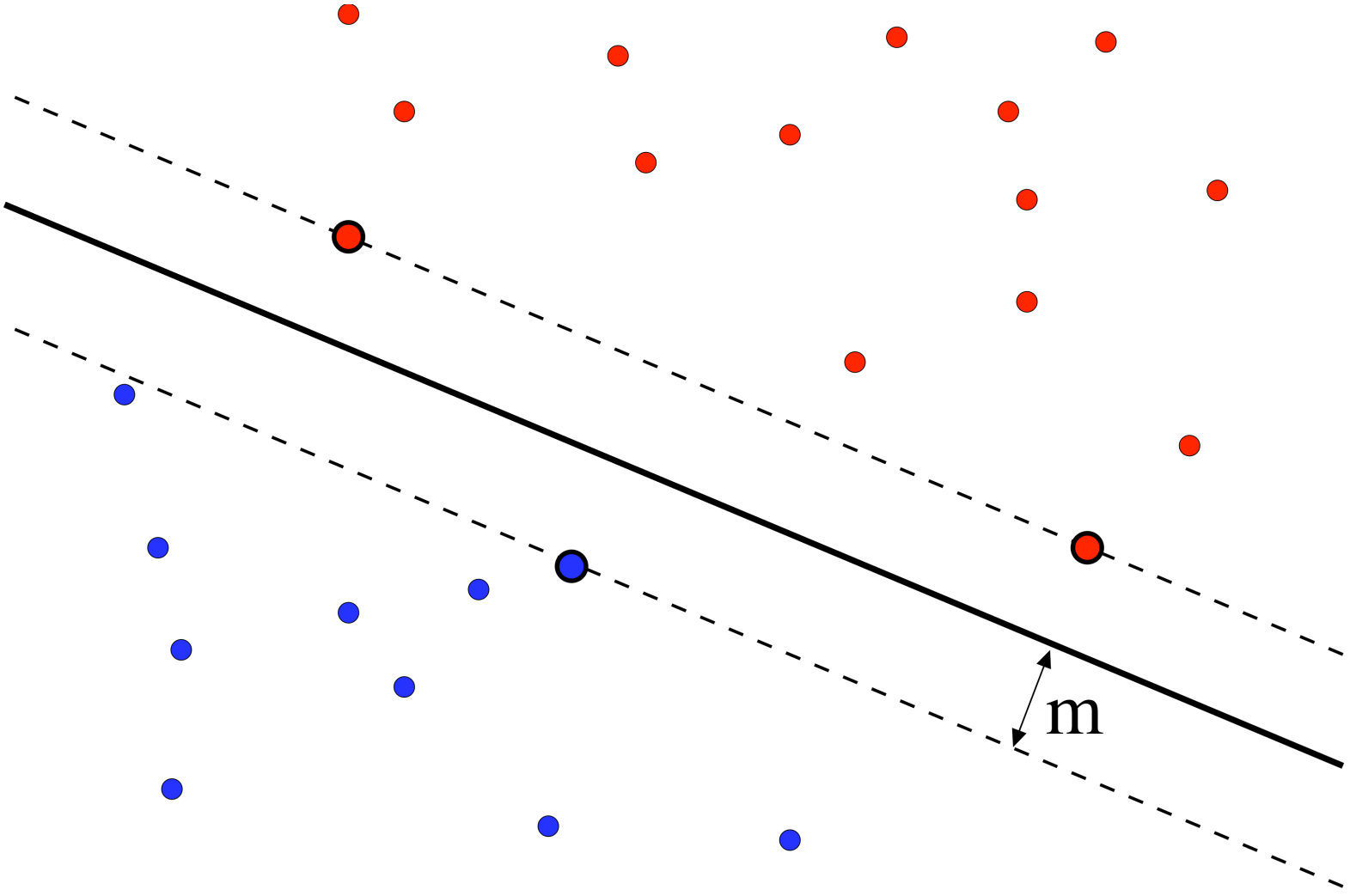
Extension to k classes:

Y an n x k indicator matrix; each row contains exactly one "1" at column j if the sample belongs to class j. All other entries are zero.

In practice: `lda` (R-package MASS)

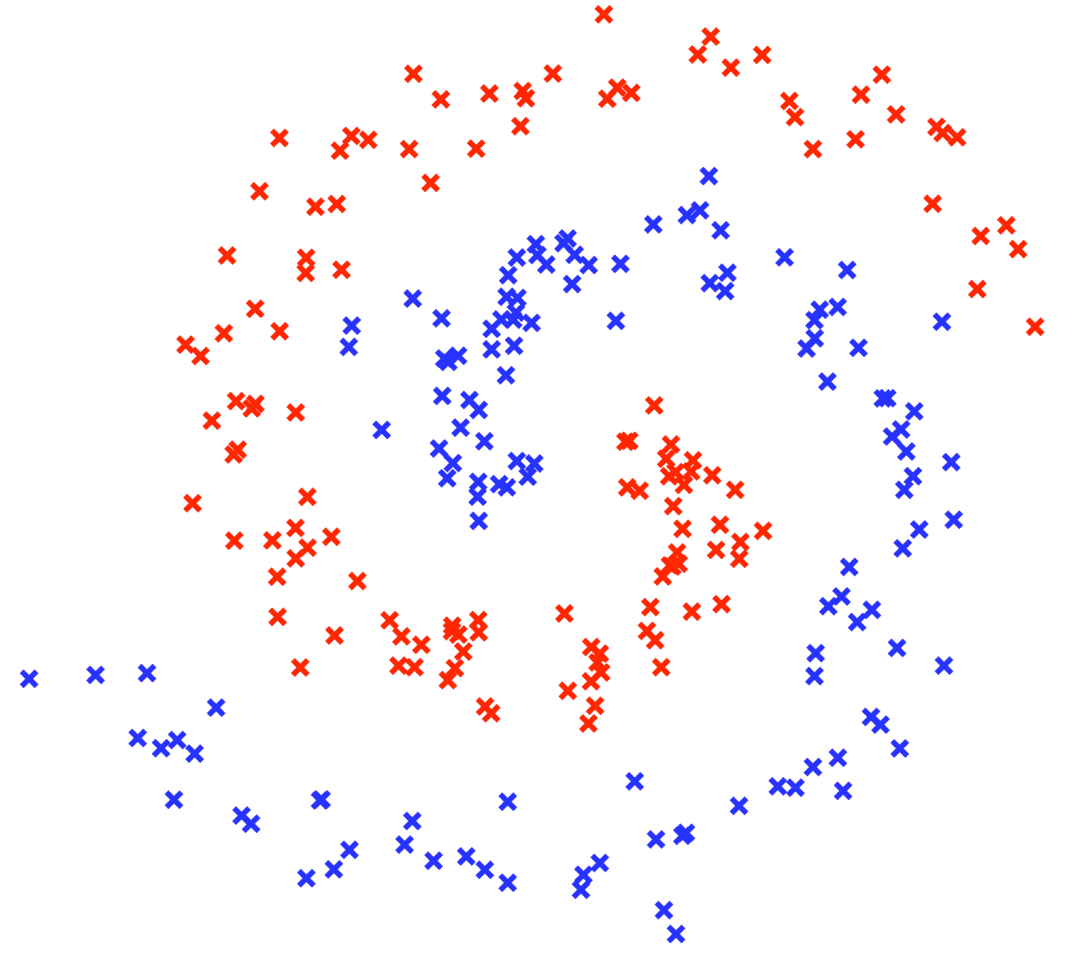
Support Vector Machine

Find a separating hyperplane with maximal margin to the samples



Non-Linear Classifiers

These classes can not be separated by a straight line (hyperplane)



Feature Transformation

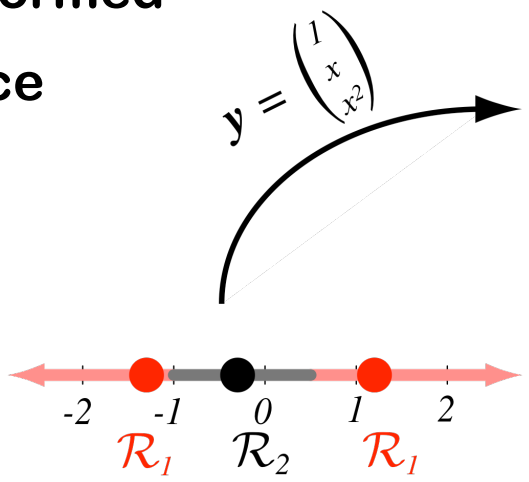
Transform the data with non-linear function, e.g.

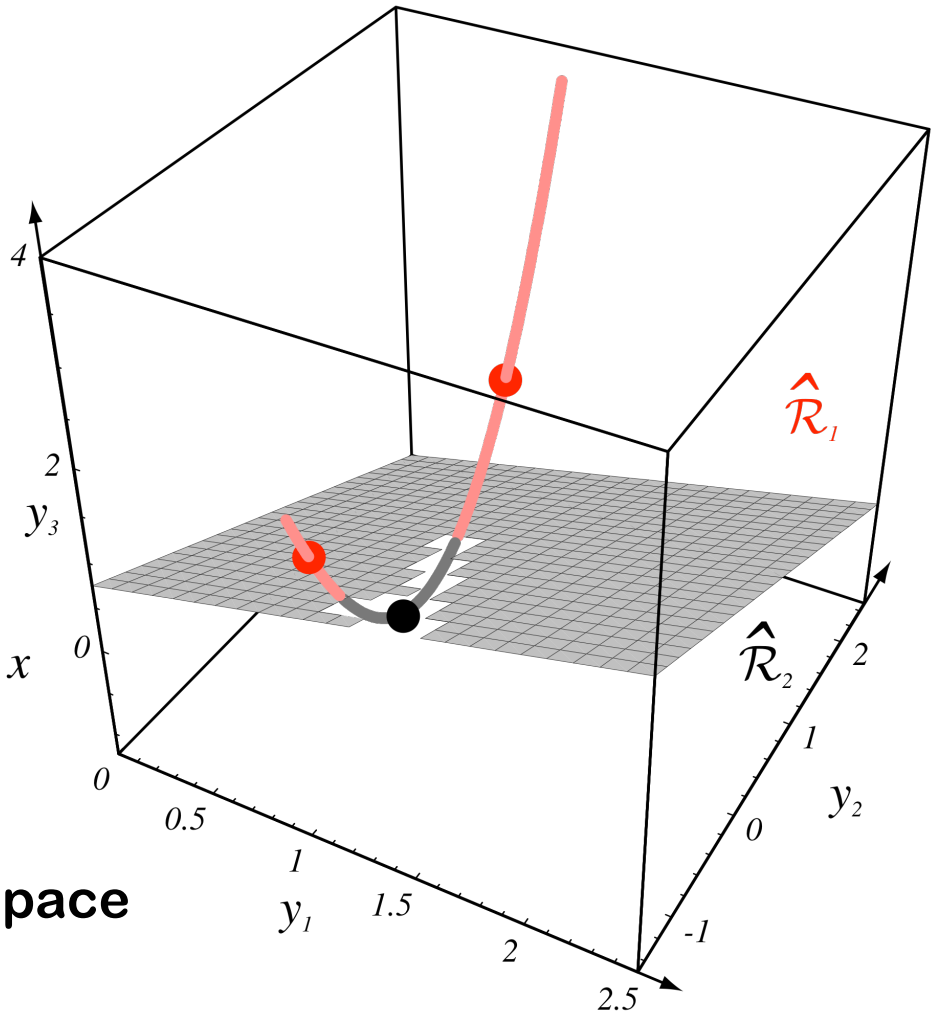
$$f(x) = \left(1, x, x^2, x^3, \ldots\right)$$

Train linear classifier

in the transformed

feature space



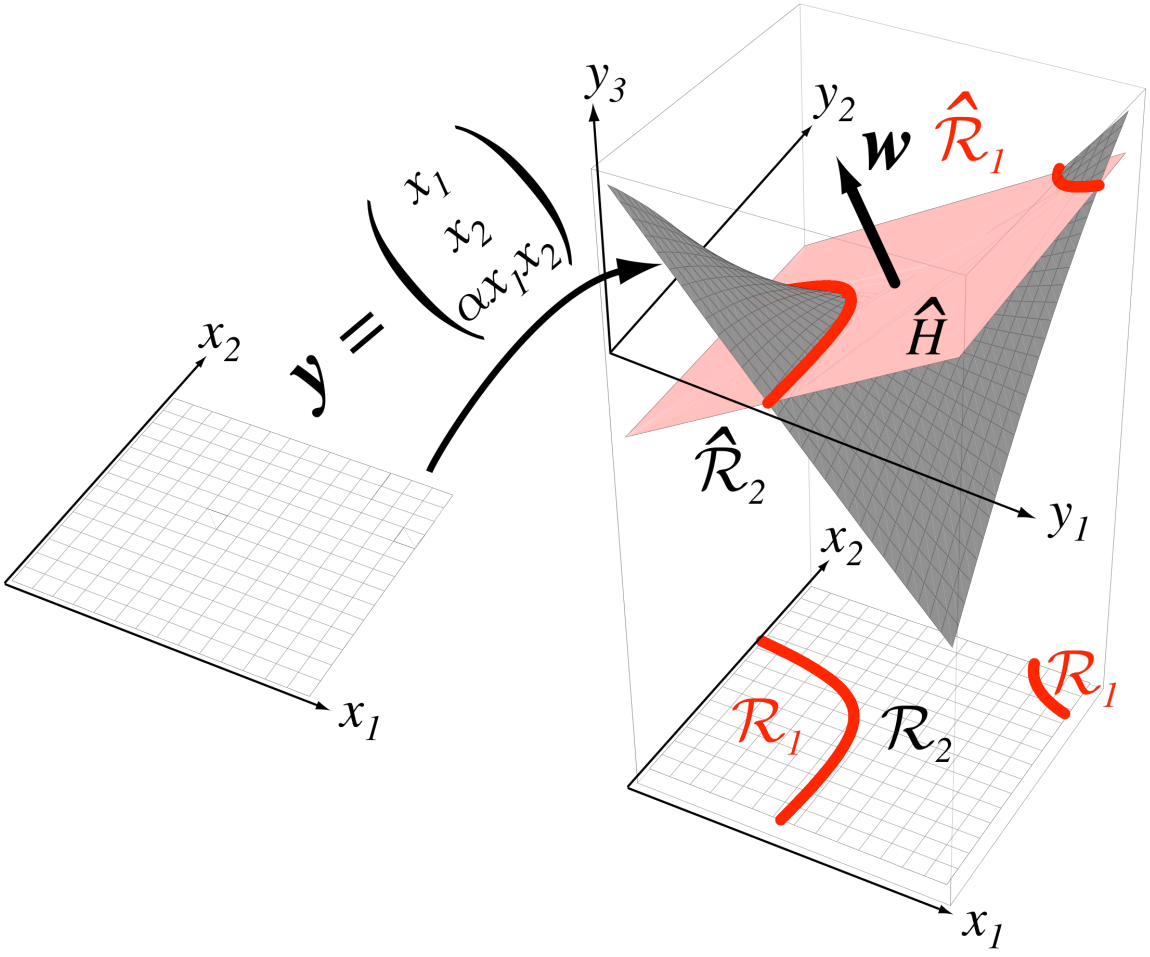


non-linear

classifier in the original feature space

Quadratic Extension

Parabolic decision boundaries can be achieved by using the product x_1x_2



The Kernel Trick

Rewrite the model such that the data *X* no longer appear directly, but only within scalar products.

Example: least squares

$$\sum_i (y_i - \beta \cdot \mathbf{x}_i)^2 \rightarrow \min$$

$$\beta = \left(X^t X\right)^{-1} X^t \mathbf{y}$$

The least squares criterion can be reformulated as a scalar product.

The matrix XX^t (i.e. $X_{ik}X_{kj}$) contains all scalar products. Replace it by

$$K_{ij} = K(x_i, x_j)$$

Implicit feature transformation. The kernel has to be positive semi-definite.

The Kernel Trick

Popular choices

Linear kernel:

$$K(x_i, x_j) = x_i x_j$$

Radial basis functions:

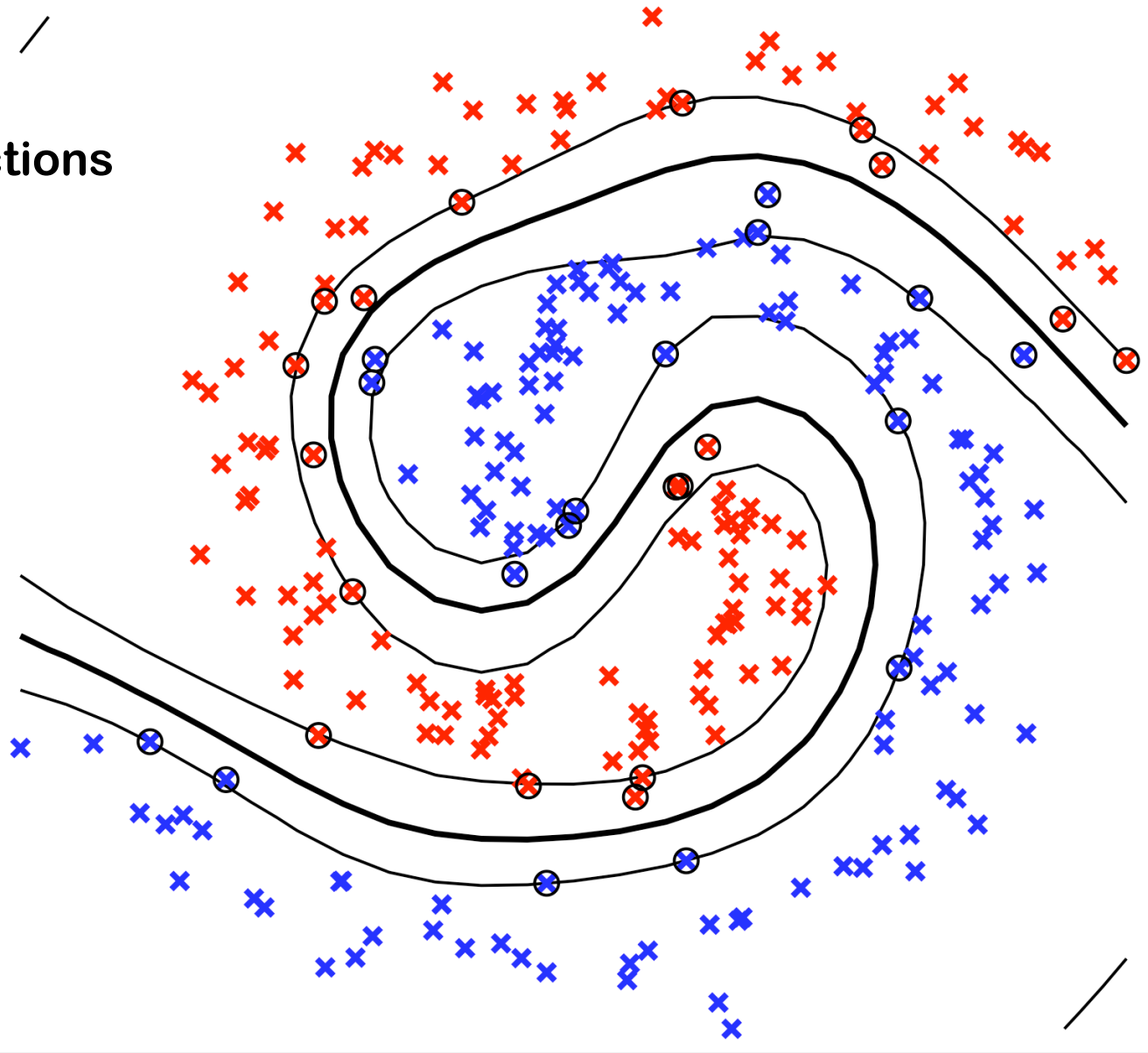
$$K(x_i, x_j) = \exp\left(-\frac{1}{2\sigma^2}\|x_i - x_j\|\right)$$

Polynomial kernel

$$K(x_i, x_j) = (x_i x_j + 1)^d$$

Examples for SVM-Classification

SVM with
Radial Basis Functions
(RBF-kernel)

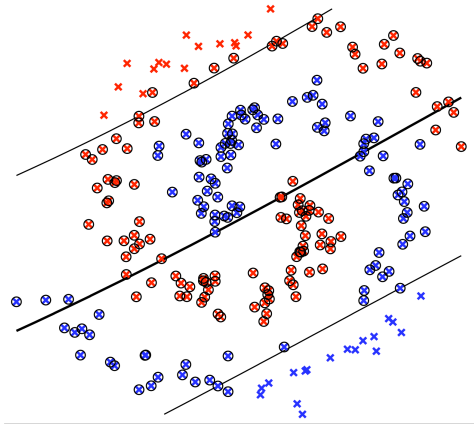


Thick line:
class separating
hyperplane

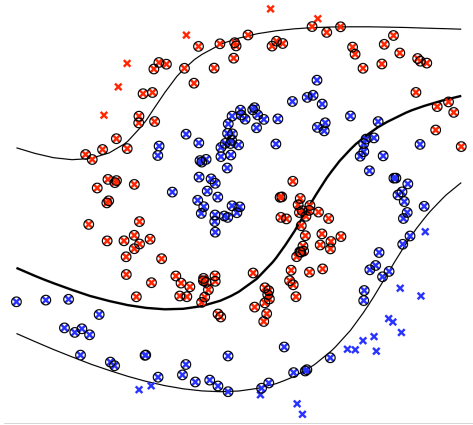
Thin line:
margin

Circles:
support vectors

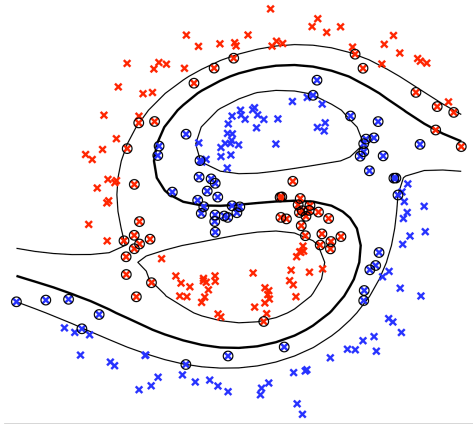
The Influence of the Kernel Parameter



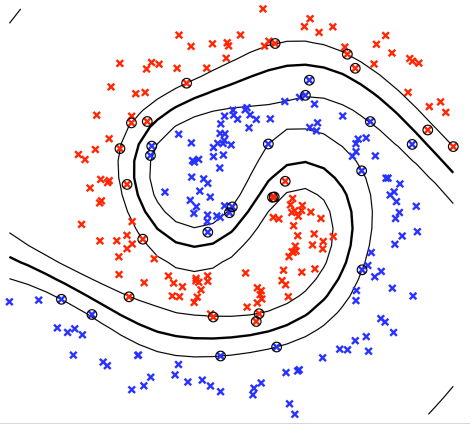
$\gamma = 0.001$



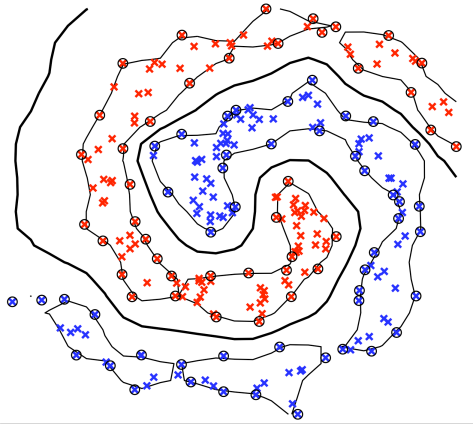
$\gamma = 0.005$



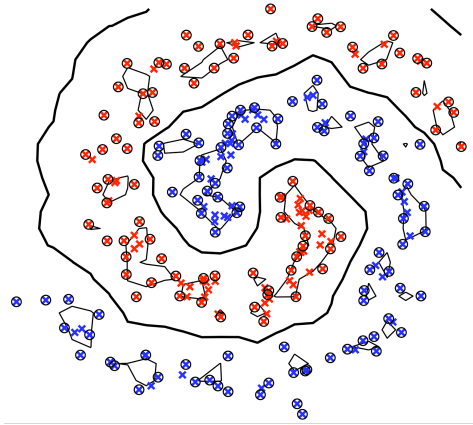
$\gamma = 0.03$



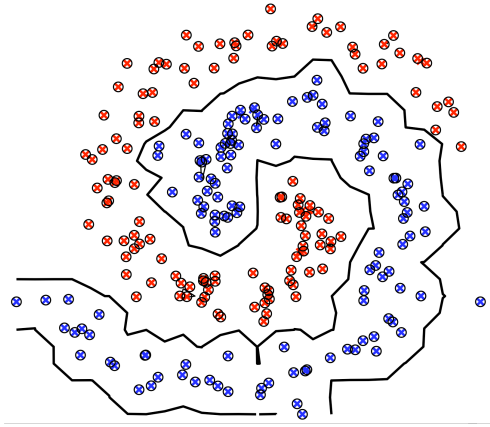
$\gamma = 0.1$



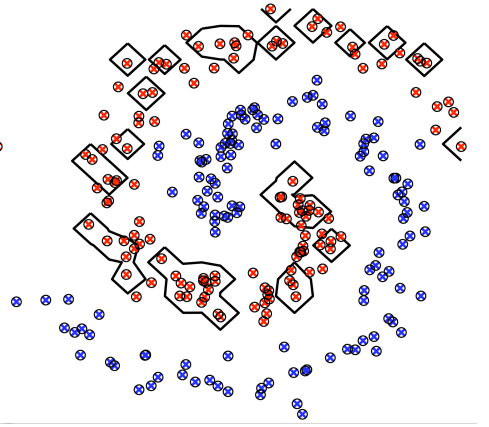
$\gamma = 1$



$\gamma = 2$



$\gamma = 20$

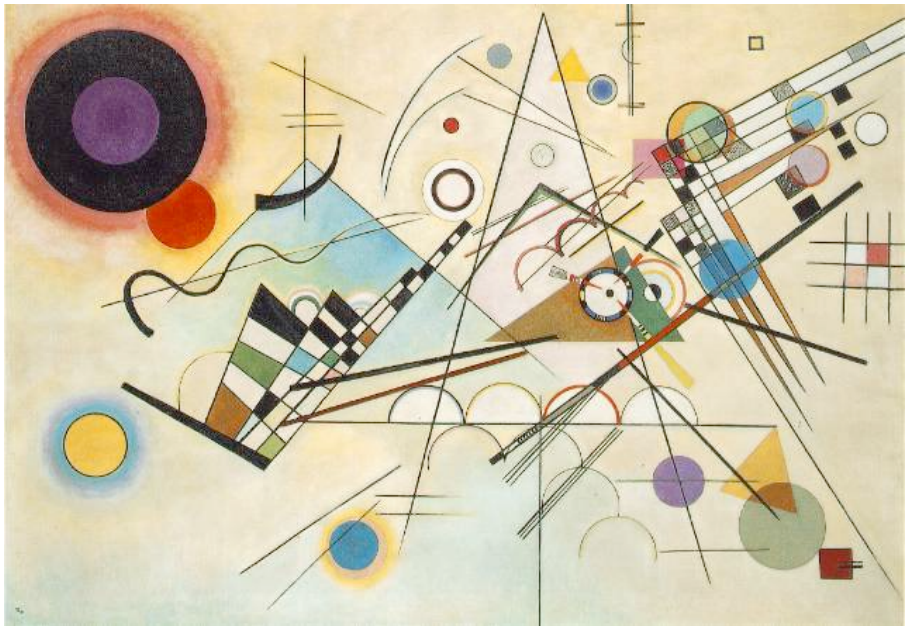


$\gamma = 200$

$\gamma = \sigma^{-2}$, RBF

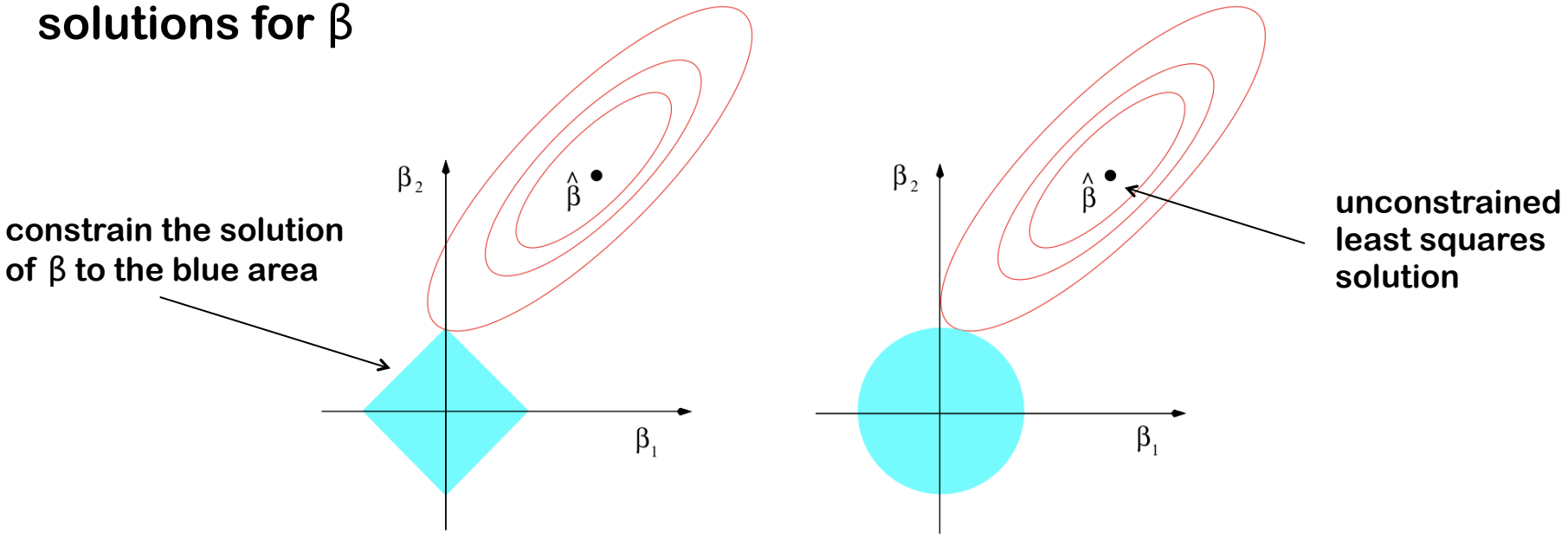
Curse of Dimensionality: overfitting guaranteed

- Consider:
  - 10 samples per class
  - Each sample is characterised by several hundred features.
- Even a linear classifier will be (always) too complex: overfitting
- There is a need to lower the complexity even below that of the linear classifier



Regularization

- Reduce the complexity by reducing the space of permissible solutions for β



Lasso:

$$\hat{\beta} = \arg\min_{\beta} \|X\beta - y\|_2^2 + \lambda\|\beta\|_1^1$$

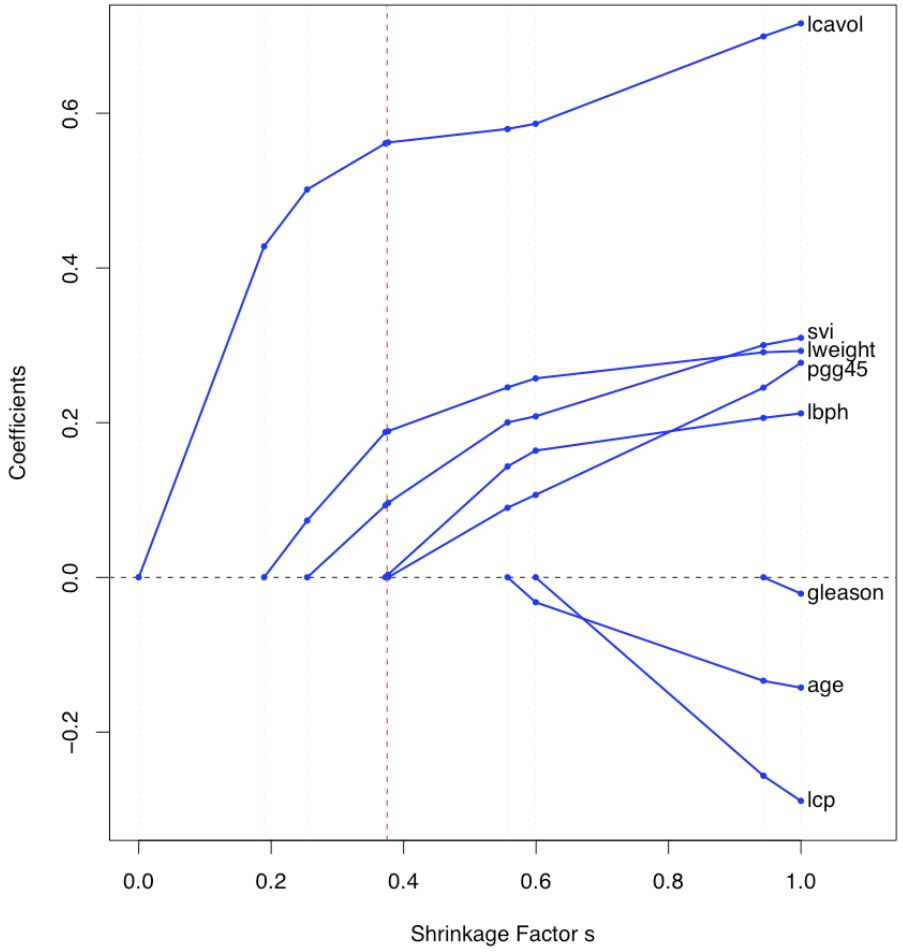
Ridge Regression

$$\hat{\beta} = \arg\min_{\beta} \|X\beta - y\|_2^2 + \lambda\|\beta\|_2^2$$

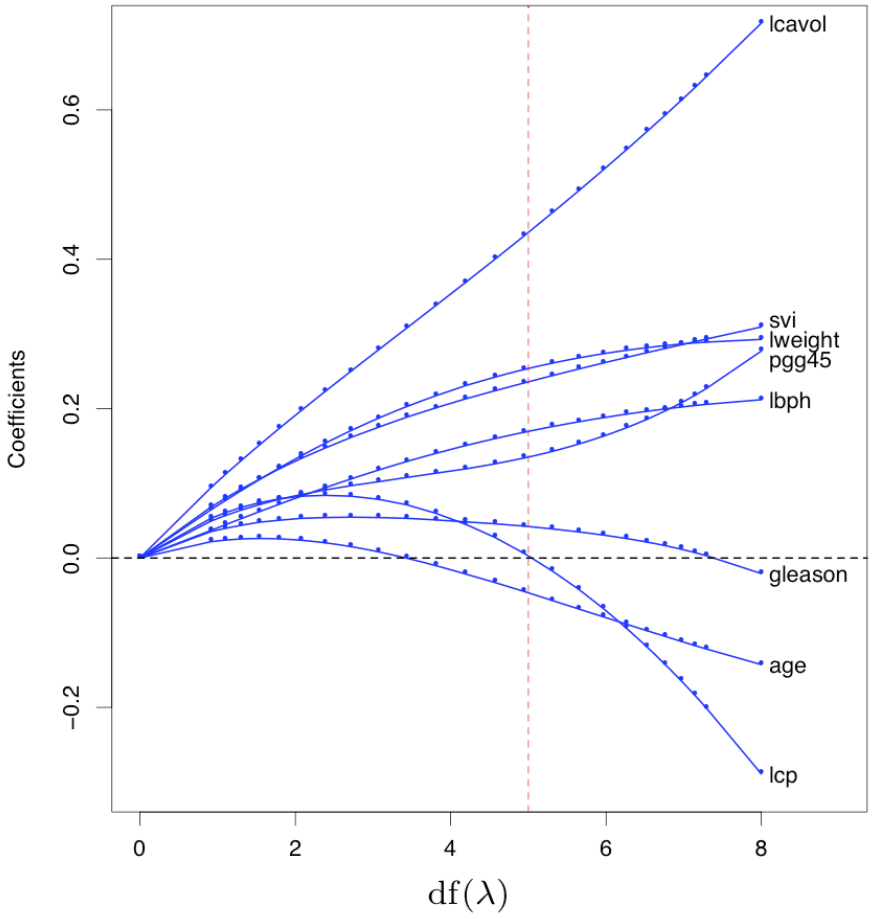
Lagrangian formulation of constrained optimization.
The blue area becomes larger, the smaller λ.
Lasso: sparse solution. Many coefficients βi become 0. Only a few coefficients are used for prediction. Implicitly selects features.

Regularization Path

The coefficients for varying regularization parameter λ

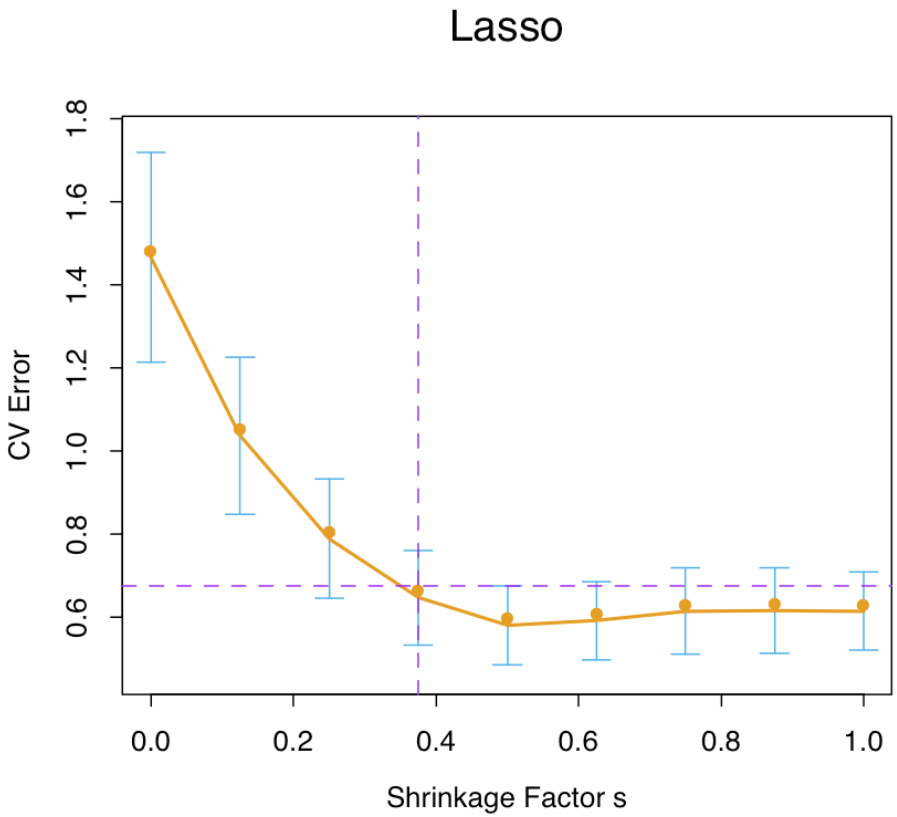


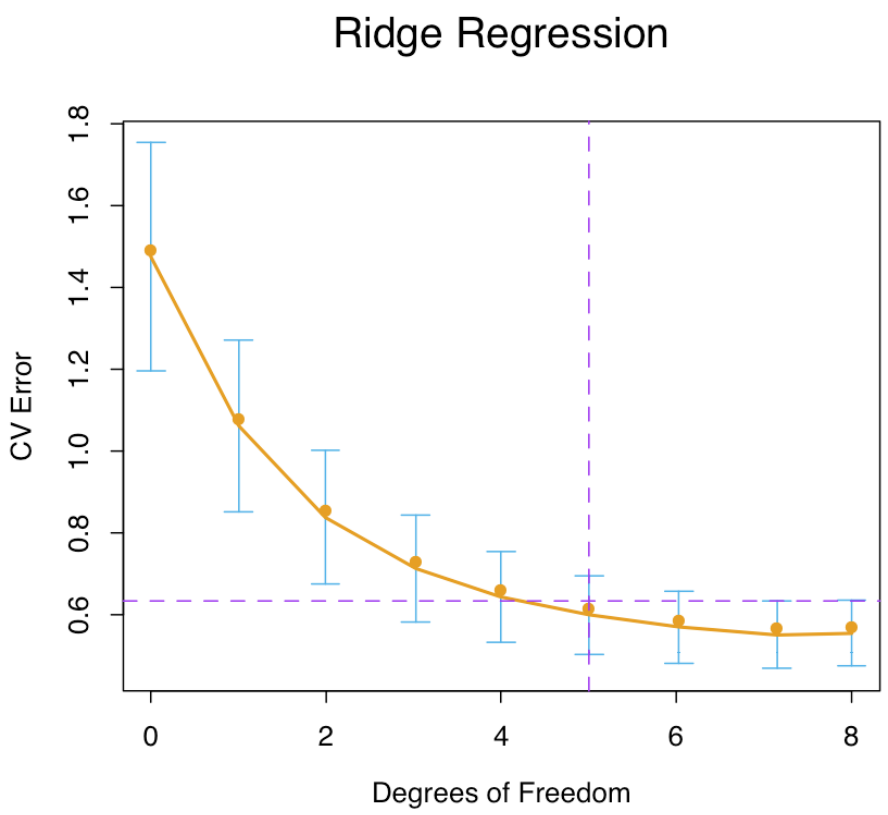
Lasso



Ridge Regression

Cross-Validation for Regularized Regression

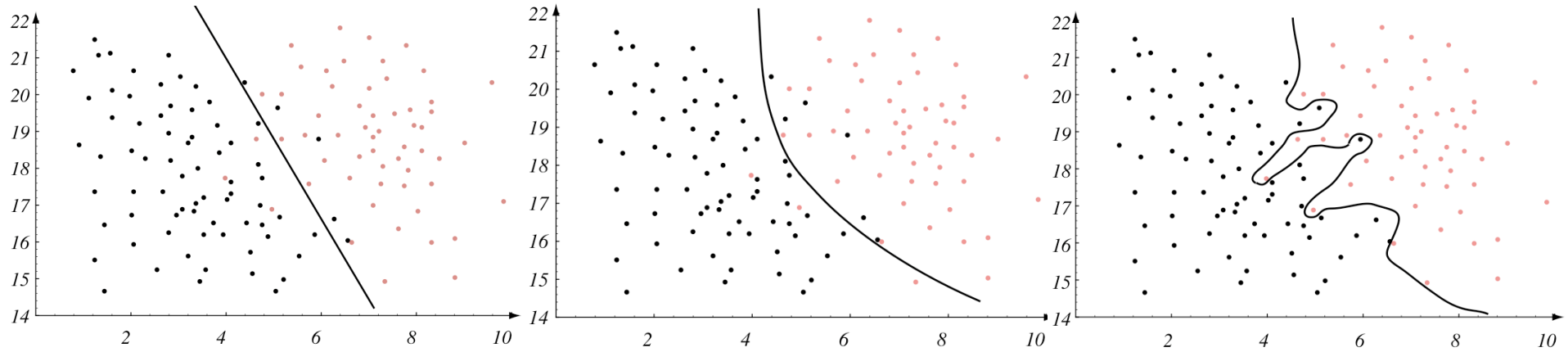




Summary: It's all about adapting the complexity of the model to that of the data

High bias
Low variance

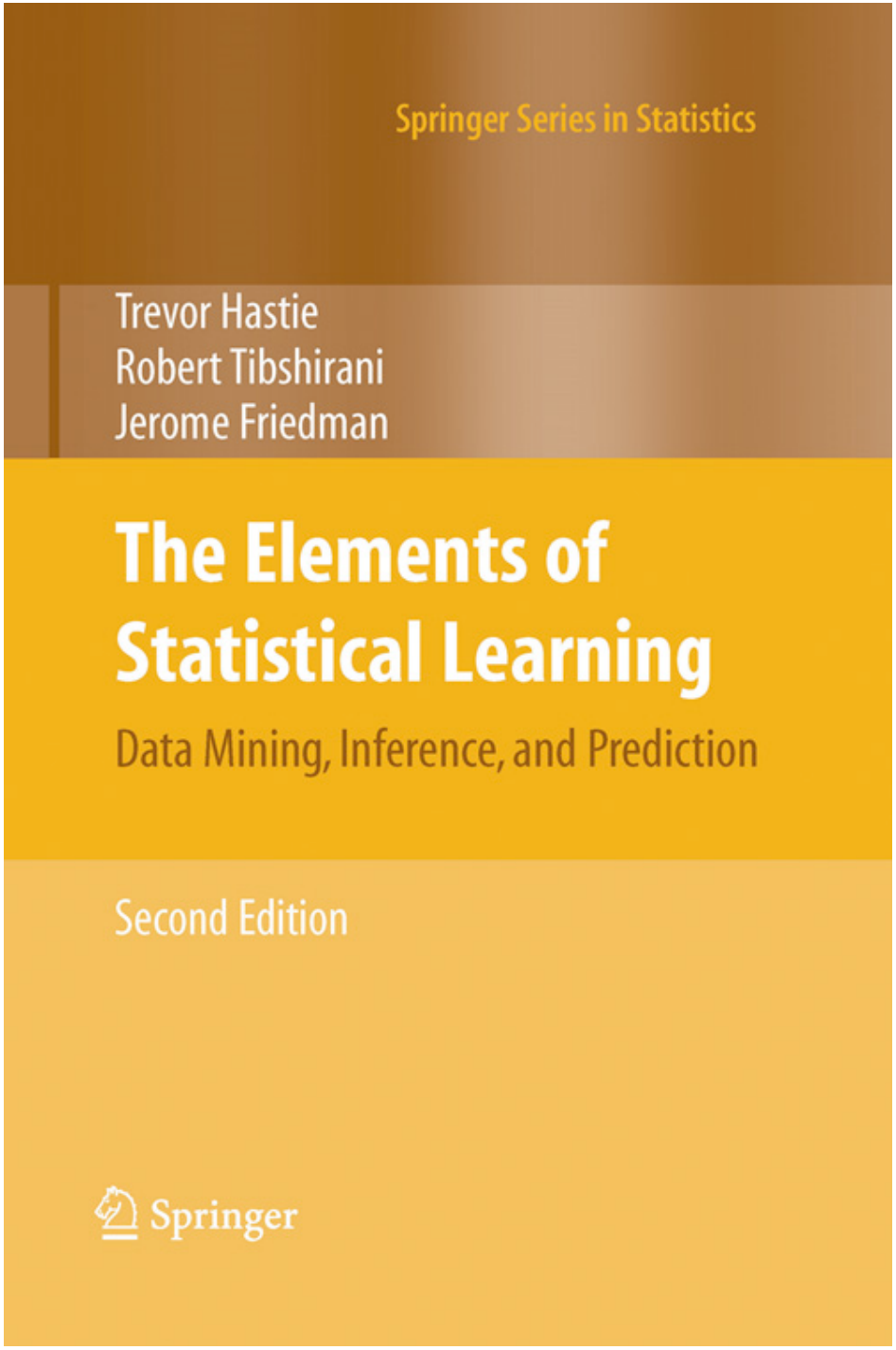
Low bias
High variance



low model complexity
(2 parameters describe the decision boundary)

high model complexity
(hundreds of parameter to describe the decision boundary)

Reduce complexity by regularization (Lasso, ridge, …)
Increase complexity by feature transformation or kernel functions
Always assess classifiers by cross-validation



Free PDF download