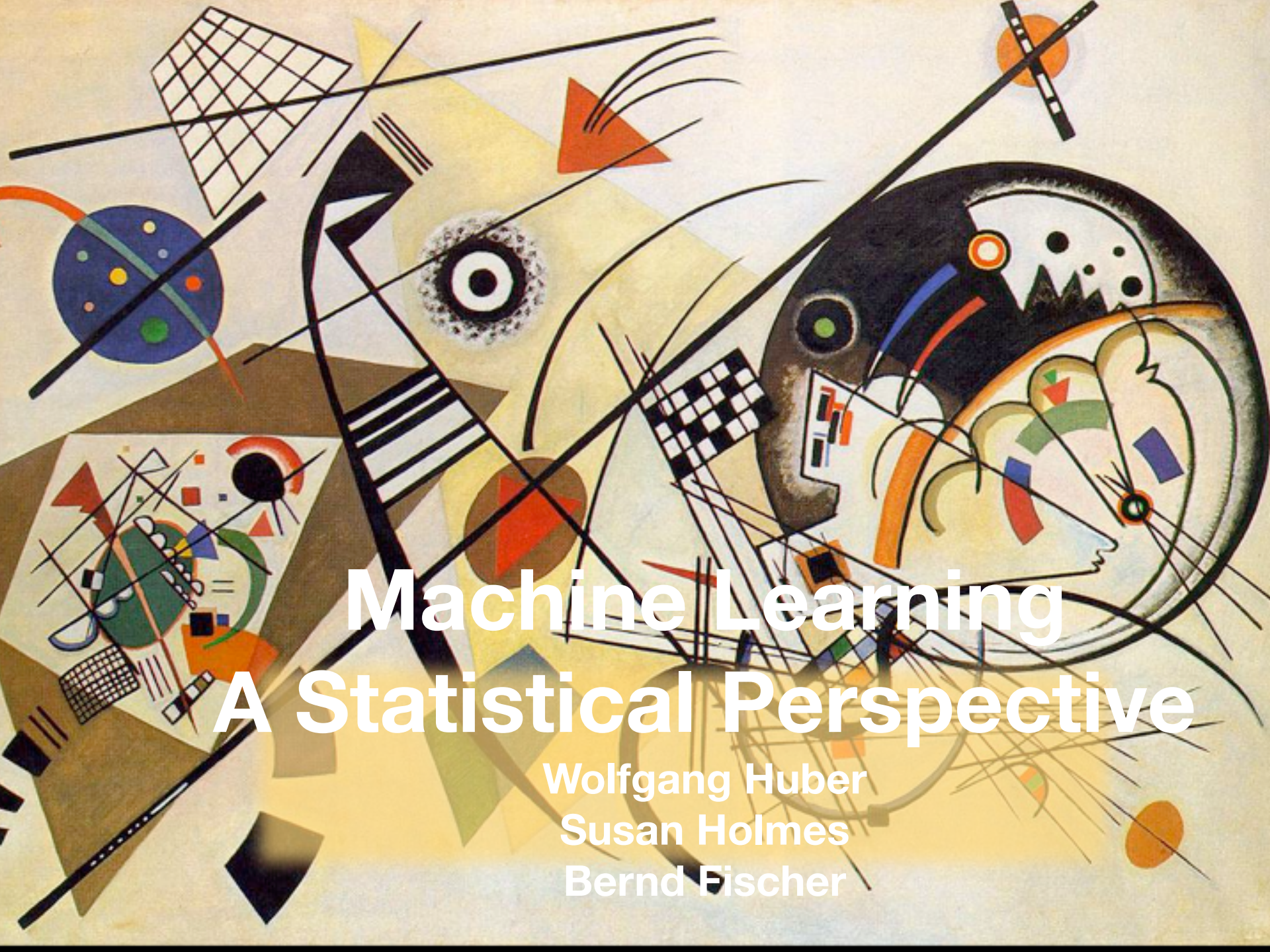


Machine Learning
A Statistical Perspective

Wolfgang Huber
Susan Holmes
Bernd Fischer

What you will learn in this lecture

- Exemplary applications
- Linear discriminant analysis
- How to get non-linear decision boundaries
- Hyperparameters
- Using cross-validation to
  - tune parameters
  - assess performance

Basics

- There are two basic kinds of machine learning algorithms, supervised and unsupervised
  - **Supervised - classification**: usually there is a training set with features and class labels
  - **Unsupervised - clustering:** there is no training set, or set with known class labels
- Typically we have **observations** (individuals) with **features** (covariates, phenotypes) measured on each observation
- All machine learning algorithms depend on finding some measure of similarity (or distance) between **observations**
- In many situations the features will need to be transformed, or manipulated (feature engineering) to better suit the task.
- Often feature selection - which features to use - or feature engineering are part of the process

The diabetes data

```
library("readr")
library("magrittr")
diabetes = read_csv("../data/diabetes.csv", col_names = TRUE)
diabetes

## # A tibble: 144 x 7
##       id relwt glufast glutest steady insulin group
##    <int> <dbl>   <int>   <int>  <int>   <int> <int>
##  1     1  0.81      80     356    124      55     3
##  2     3  0.94     105     319    143     105     3
##  3     5  1.00      90     323    240     143     3
##  4     7  0.91     100     350    221     119     3
##  5     9  0.99      97     379    142      98     3
##  6    11  0.90      91     353    221      53     3
##  7    13  0.96      78     290    136     142     3
##  8    15  0.74      86     312    208      68     3
##  9    17  1.10      90     364    152      76     3
## 10    19  0.83      85     296    116      60     3
## # ... with 134 more rows

diabetes$group %<>% factor
```

We used the forward-backward pipe operator `%<>%` to convert the `group` column into a factor. The plot is shown in Figure 13.4.

```
library("ggplot2")
library("reshape2")
ggplot(melt(diabetes, id.vars = c("id", "group")),
       aes(x = value, col = group)) +
 geom_density() + facet_wrap( ~variable, ncol = 1, scales = "free") +
 theme(legend.position="bottom")
```

https://cran.r-project.org/web/packages/candisc/vignettes/diabetes.html

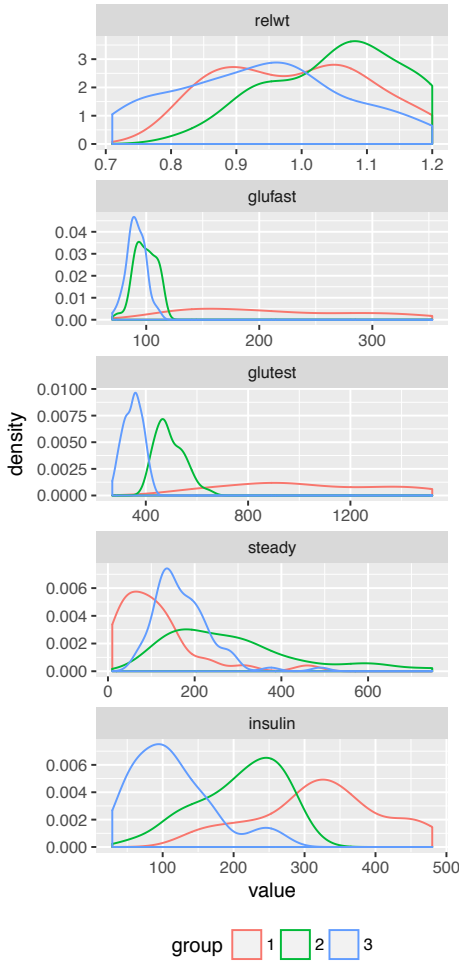


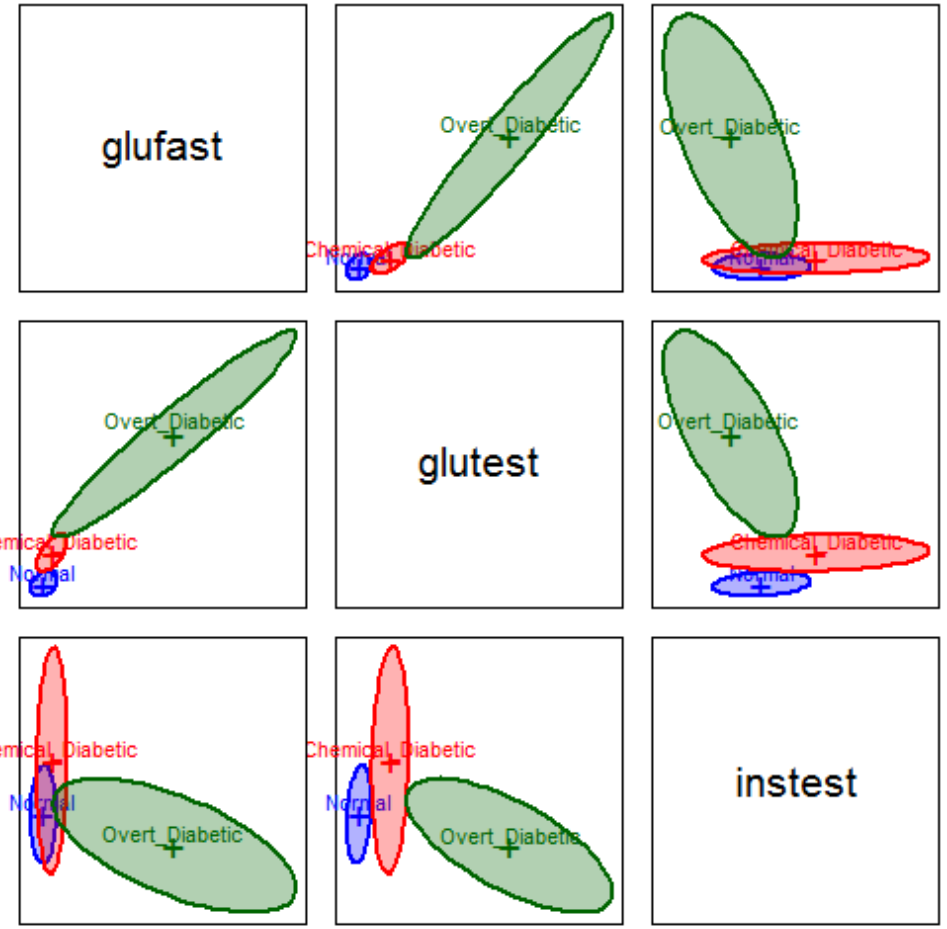
Figure 13.4: We see already from the one-dimensional distributions that some of the individual variables could potentially predict which group a patient is more likely to belong to. Our goal will be to combine variables to improve these one dimensional predictions.

Diabetes Data

The *candisc* package can give us some idea about how the data are distributed…

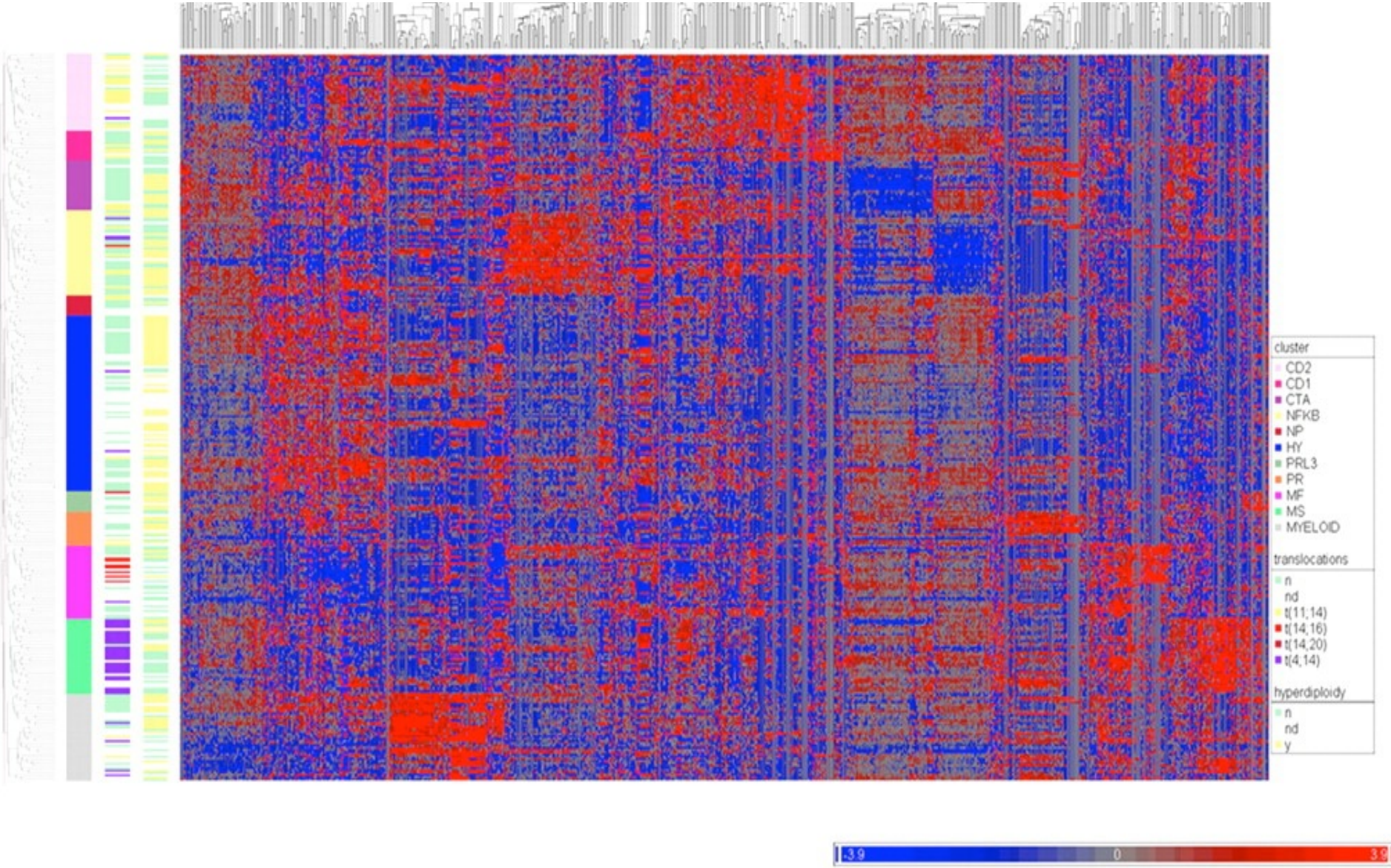
And potentially some ideas about how to best analyze it.

"It is clear from this that there is a problem of heterogeneity of variance-covariance matrices here. The normal group shows the smallest variances and the overt diabetic group the largest."



Molecular classification of cancer

(multiple myeloma in newly diagnosed patients, gene expression profiling)



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

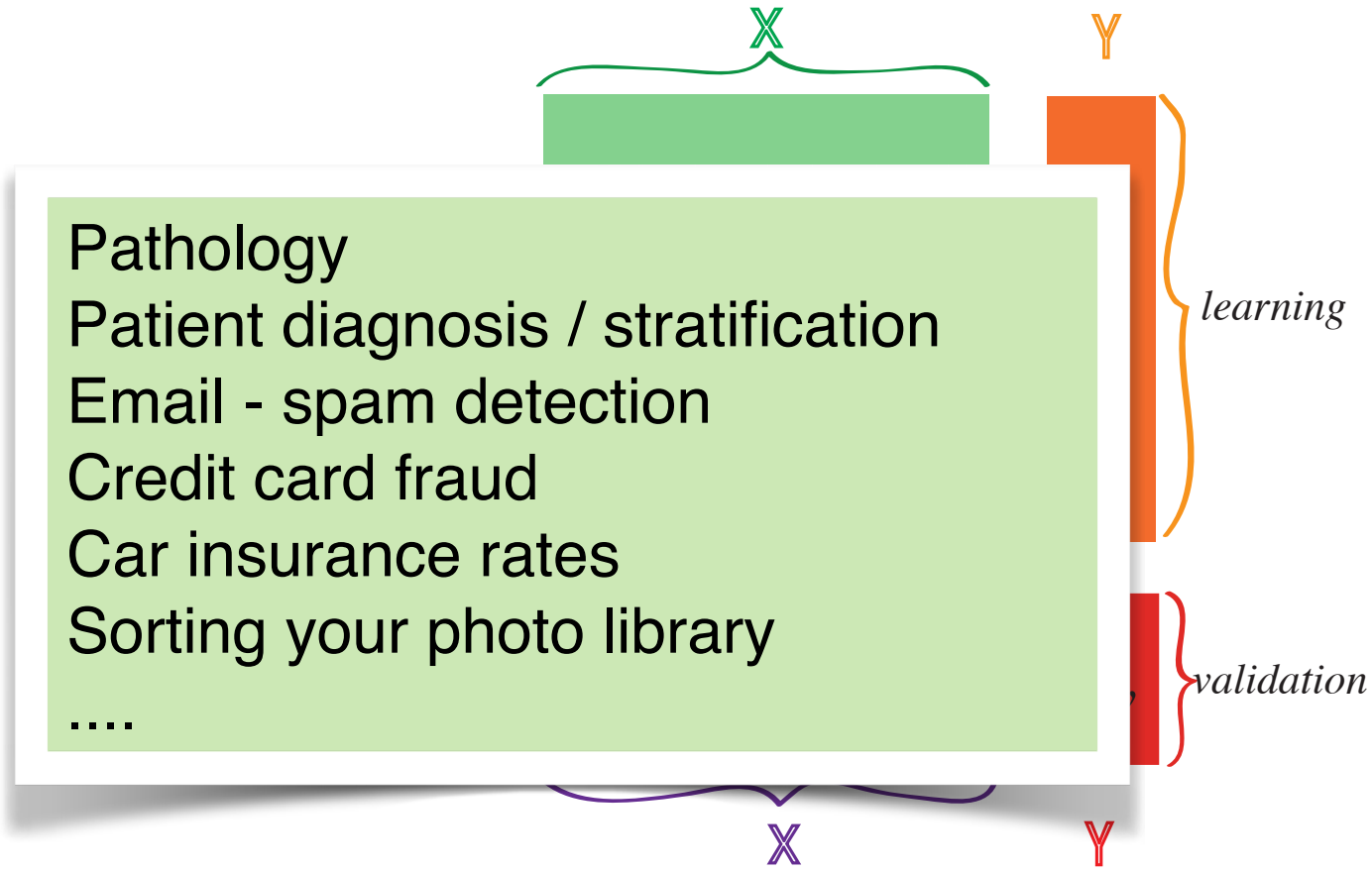


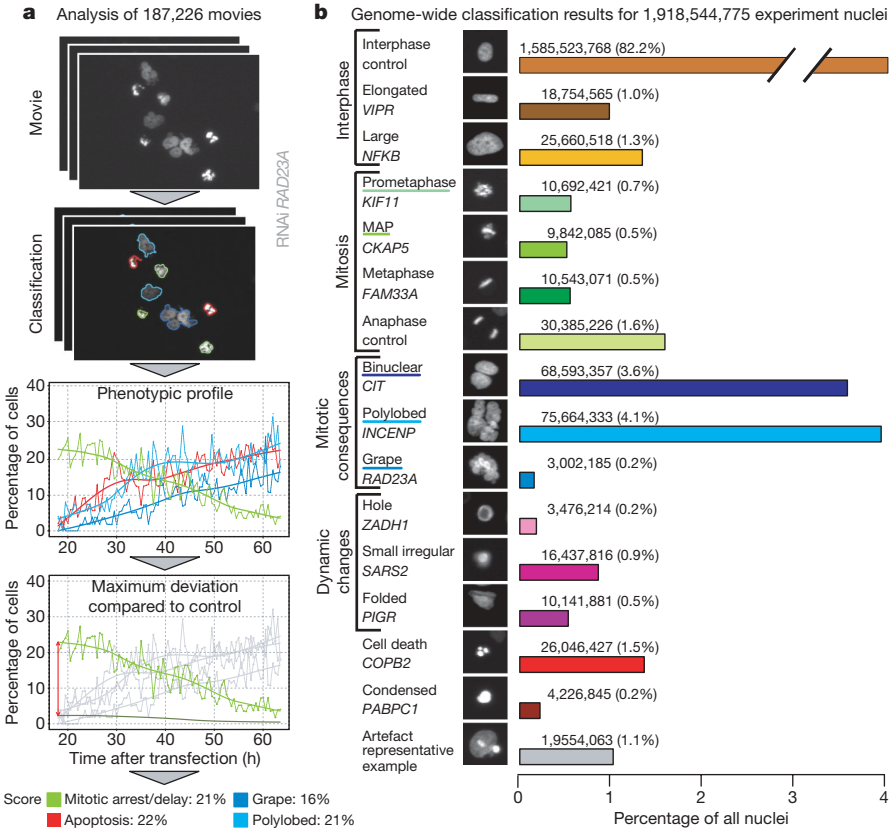
Figure 13.3: In supervised learning, we assign two different roles to our variables. We have labeled the explanatory variables X and the response variable(s) Y. There are also two different sets of observations: the training set X_ℓ and Y_ℓ and the validation set X_v and Y_v.

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

ARTICLES

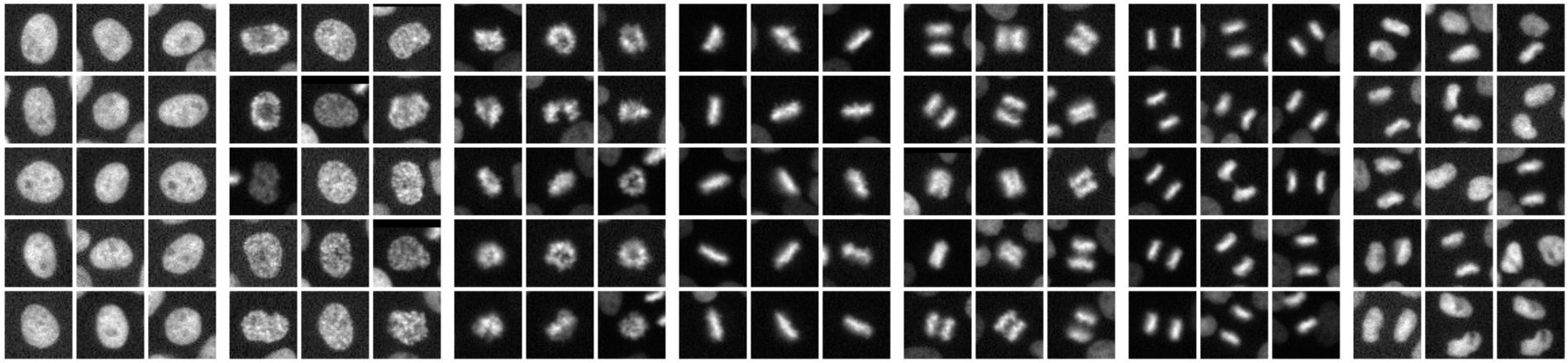
Phenotypic profiling of the human genome by time-lapse microscopy reveals cell division genes

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]

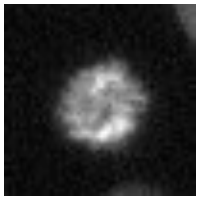


Morphological Phenotyping

Provide Human Annotation to a small set of cells:

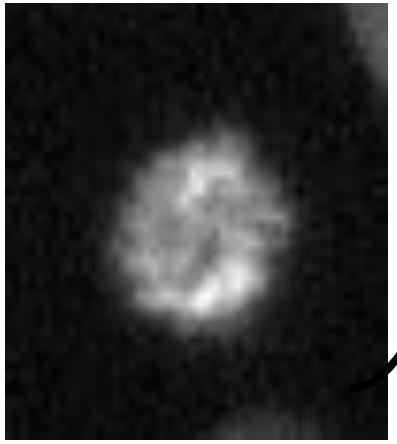


inter telo | pro | prometa | meta | earlyana | lateana



Which mitotic phase is this?
Can we do this automatically?

Automatic Classification Workflow



Preprocessing

e.g. normalization, background subtraction, …

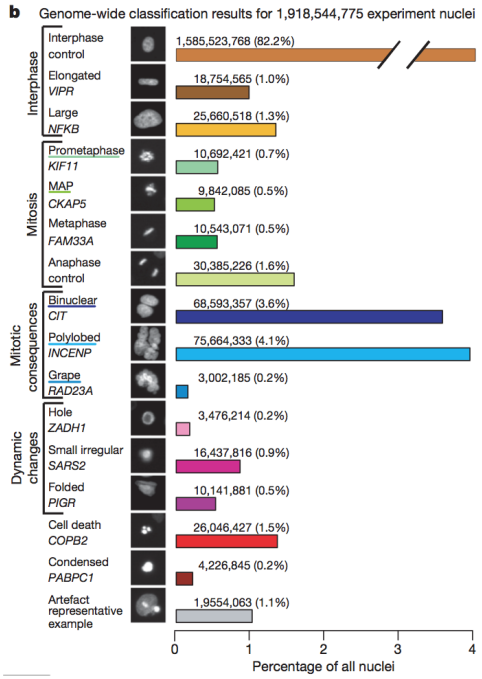
Feature extraction

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

Classification

Prophase

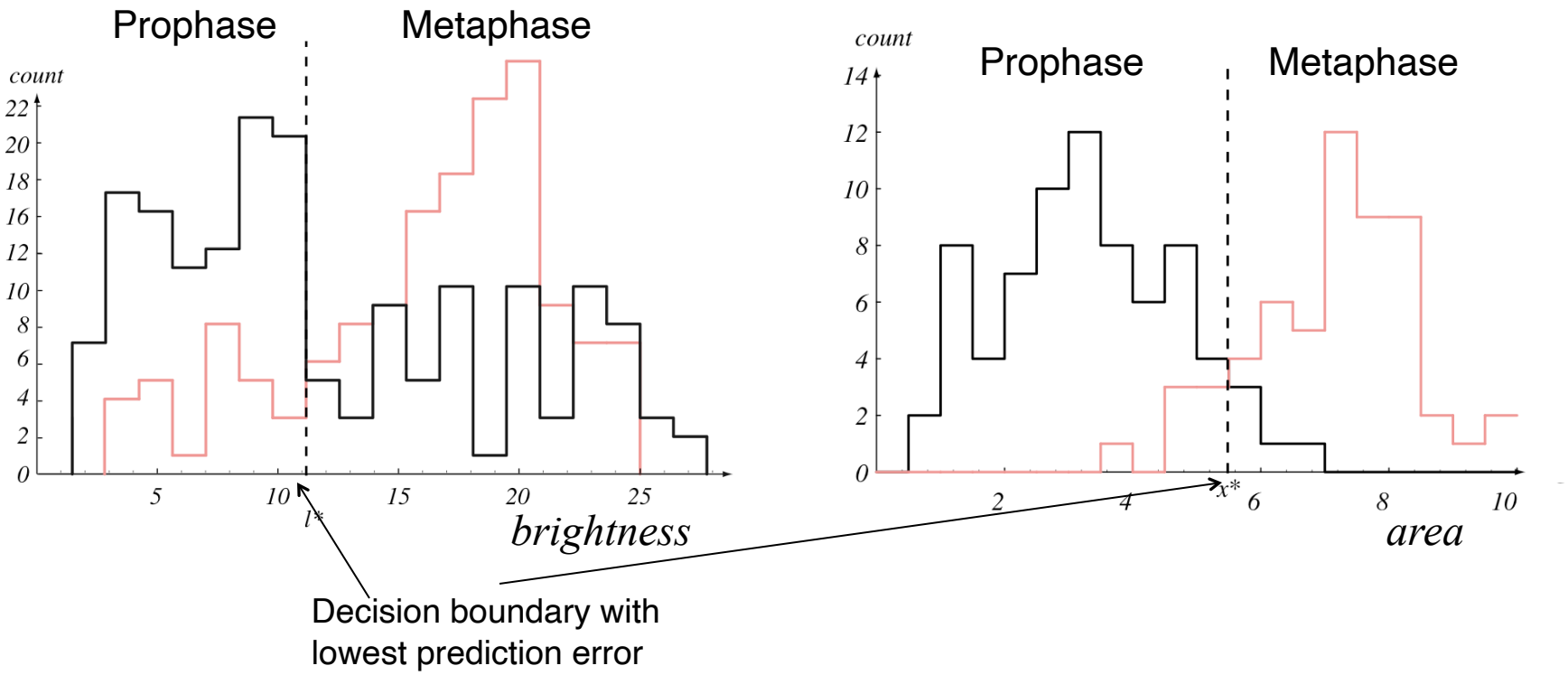
Metaphase



Prophase/ Metaphase Classification

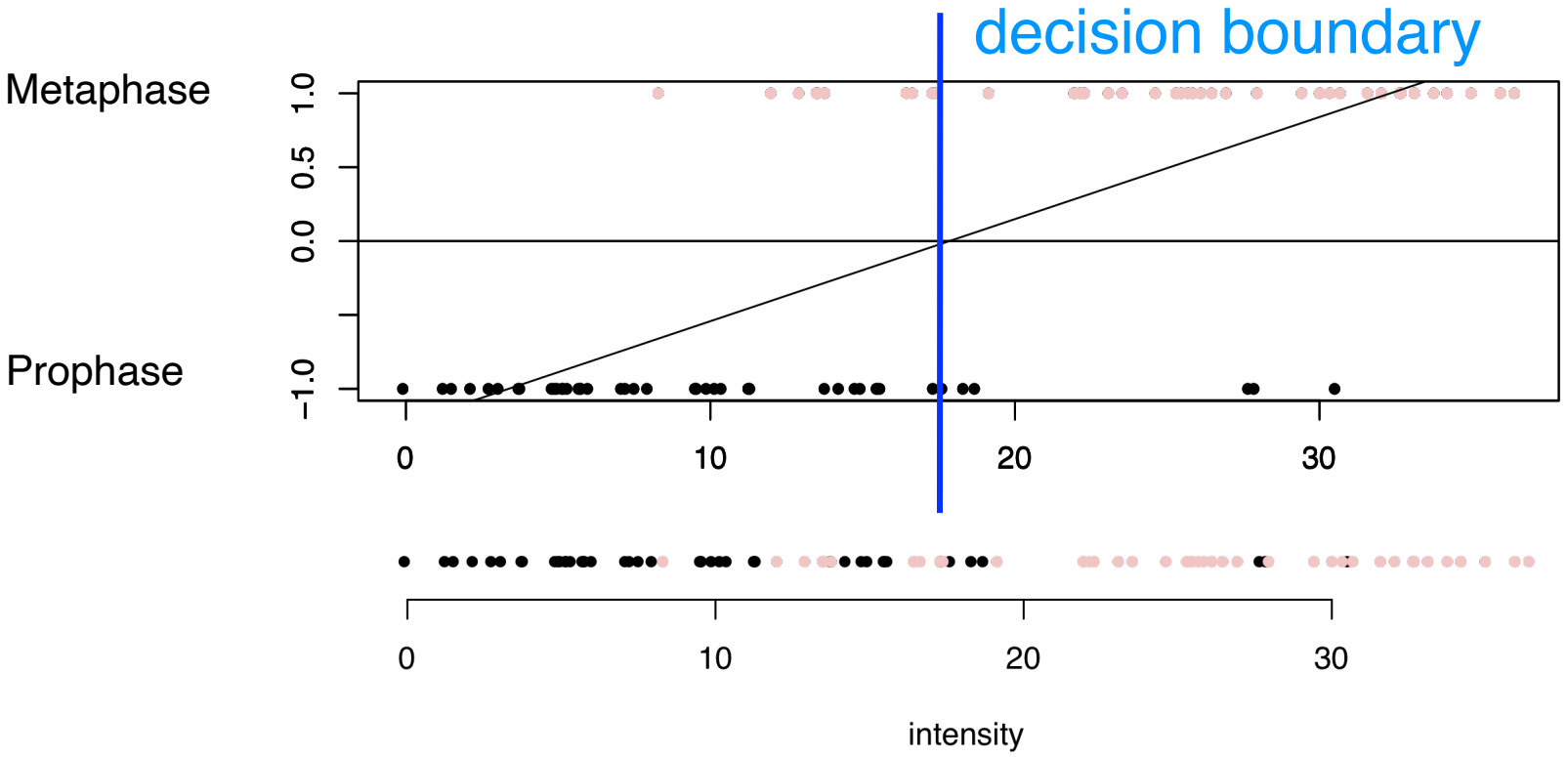
Predict mitotic state based on brightness

Predict mitotic state based on nucleus area



Both features are informative, but none of them individually has a good predictive power

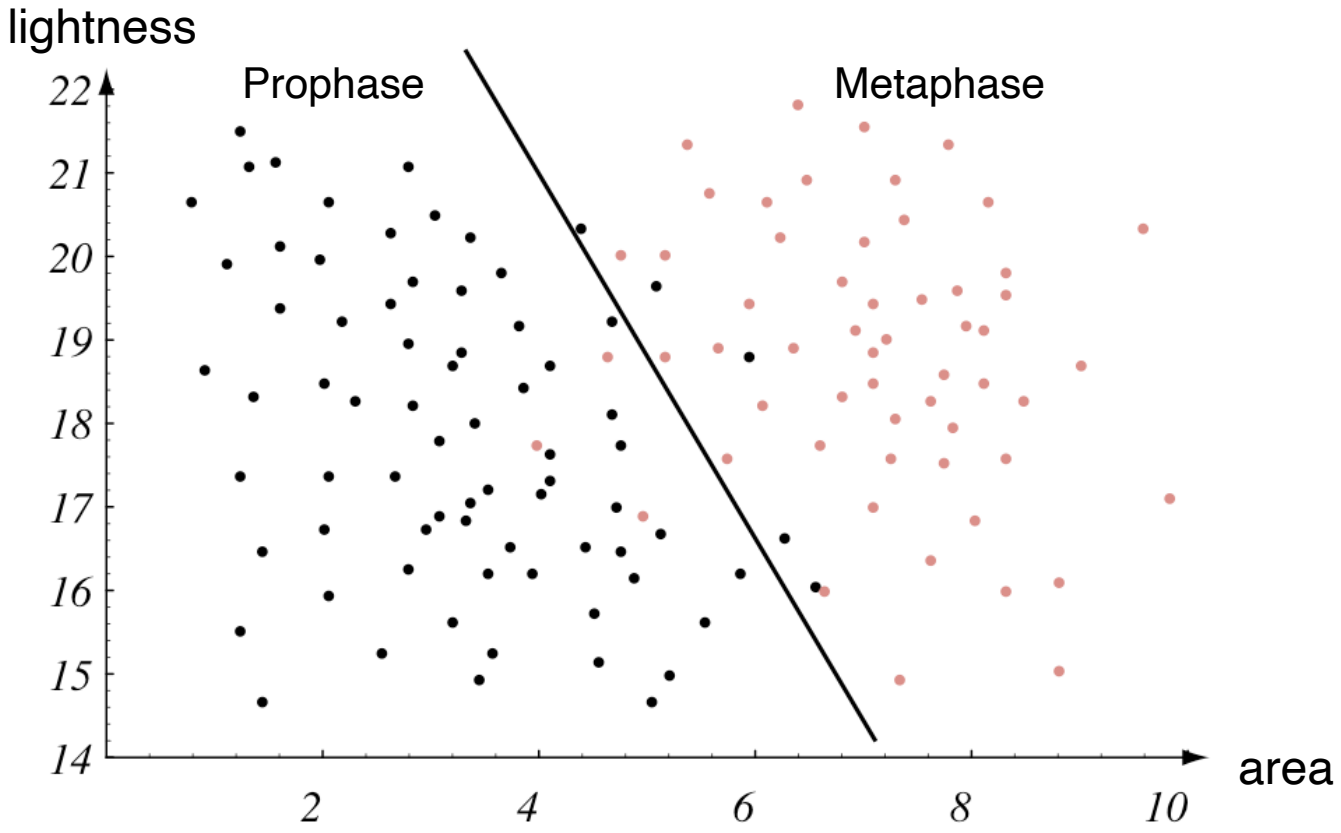
A Simple Least Squares Classifier (1D)




```
y[i] = -1 for prophase
y[i] = +1 for metaphase
x[i,] = intensity[i]
model = lm(y ~ x)
ynew = predict(model, newdata=newX)
ifelse(ynew < 0, “pro”, “meta”)
```

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

A Simple Least Squares Classifier (2D)




```
y[i] = +1 for prophase
y[i] = -1 for metaphase
x[i,] = c(area[i],lightness[i])
model = lm(y~x)
ynew = predict(model, Xnew)
            $fitted.values
ifelse(ynew < 0, "meta", "pro")
```

Linear discriminant Analysis (LDA) on the diabetes data

```
library("MASS")
```

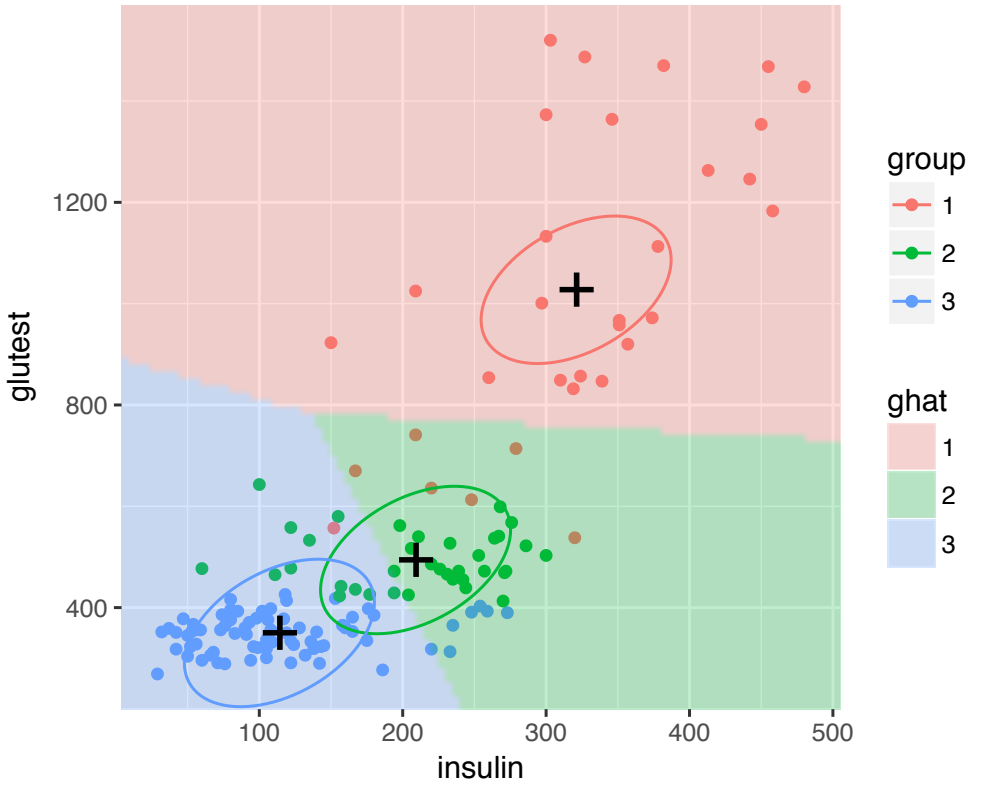
Why is the boundary between the prediction regions for groups 2 and 3 not half-way between the centers?

```
ghat = predict(diabetes_lda)$class
table(ghat, diabetes$group)

##
## ghat  1  2  3
##    1 25  0  0
##    2  6 24  6
##    3  1 12 70

mean(ghat != diabetes$group)

## [1] 0.1736111
```



QDA: Represent each group by a bivariate Normal $N(\mu_g, \Sigma_g)$

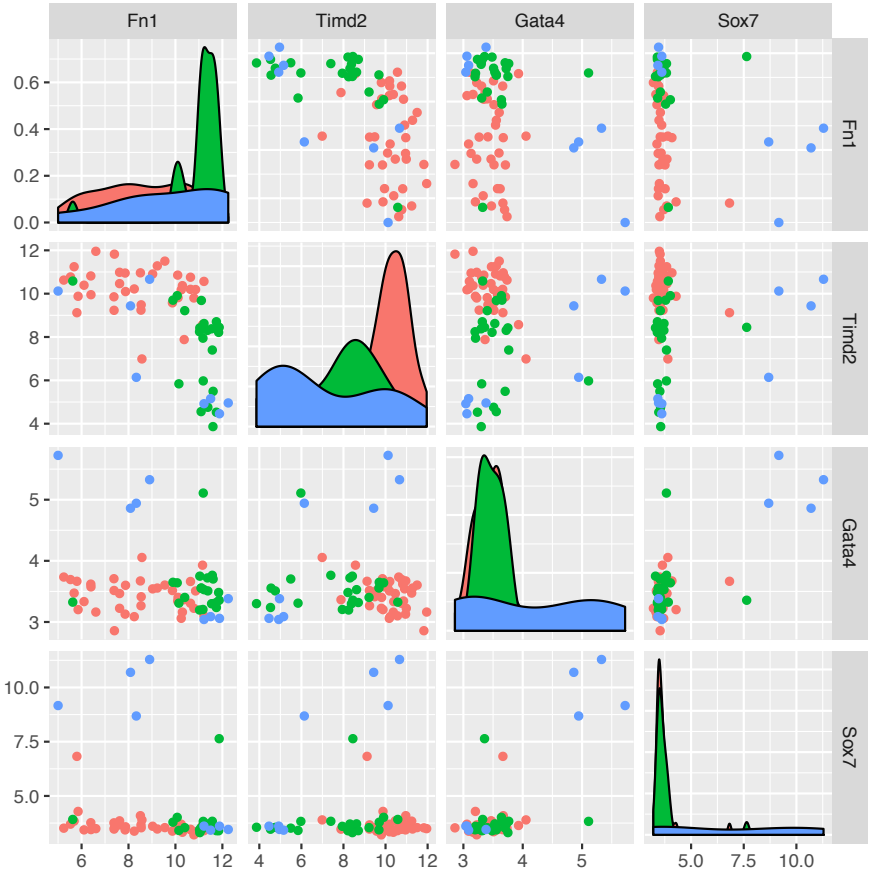
LDA: $\Sigma_g = \Sigma$

Hiiragi mouse embryo single cell expression data

```
library("Hiiragi2013")
data("x")
probes = c("1426642_at", "1418765_at", "1418864_at", "1416564_at")
embryoCells = t(exprs(x)[probes, ]) %>% as_tibble %>%
  mutate(Embryonic.day = x$Embryonic.day) %>%
  filter(x$genotype == "WT")
```

```
library("mouse4302.db")
anno = AnnotationDbi::select(mouse4302.db, keys = probes,
         columns = c("SYMBOL", "GENENAME"))
anno
```

```
##       PROBEID SYMBOL
## 1 1426642_at    Fn1
## 2 1418765_at  Timd2
## 3 1418864_at  Gata4
## 4 1416564_at   Sox7
##                                                GENENAME
## 1                                            fibronectin 1
## 2 T cell immunoglobulin and mucin domain containing 2
## 3                                  GATA binding protein 4
## 4                  SRY (sex determining region Y)-box 7
```



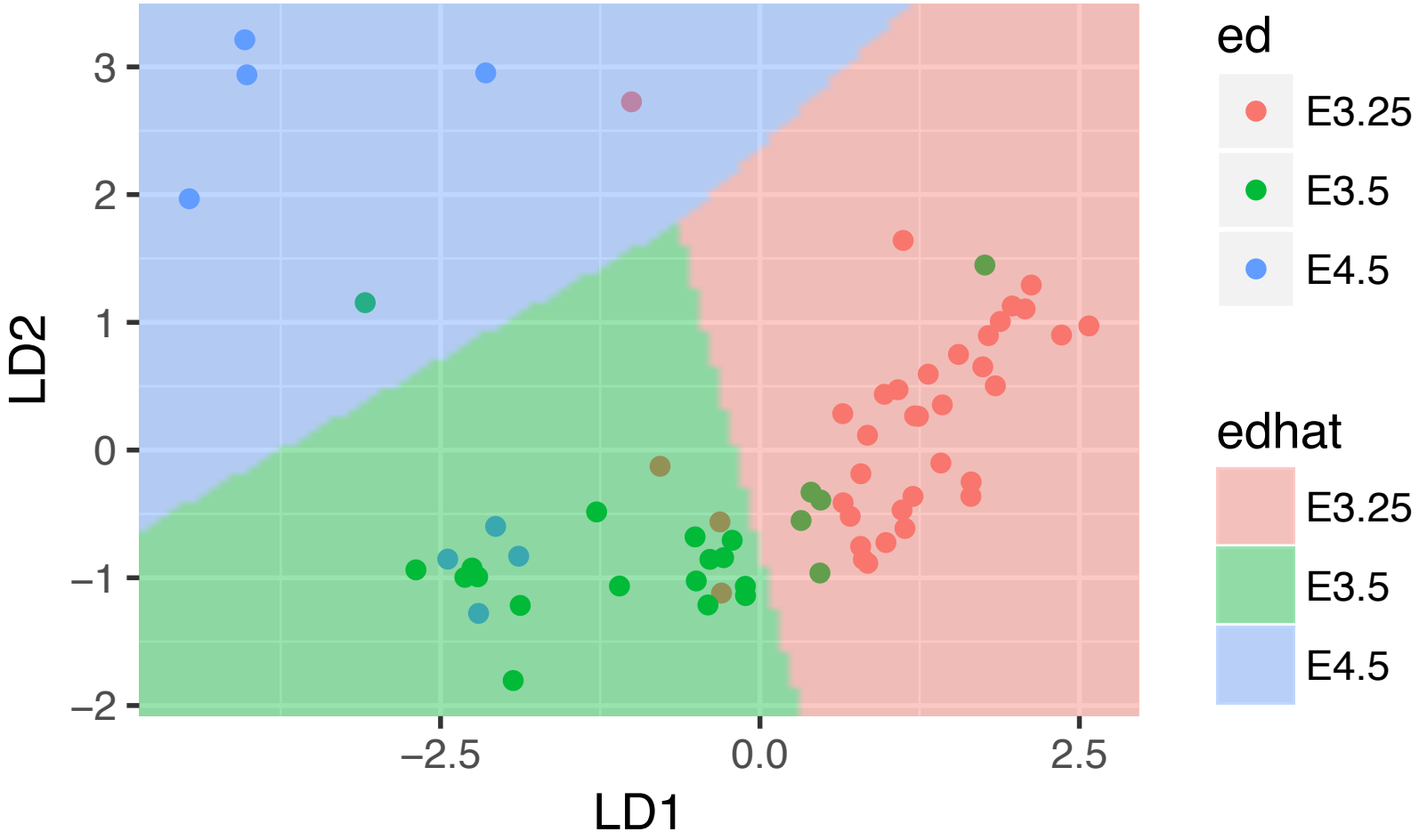


Figure 13.10: LDA classification regions for Embryonic.day.

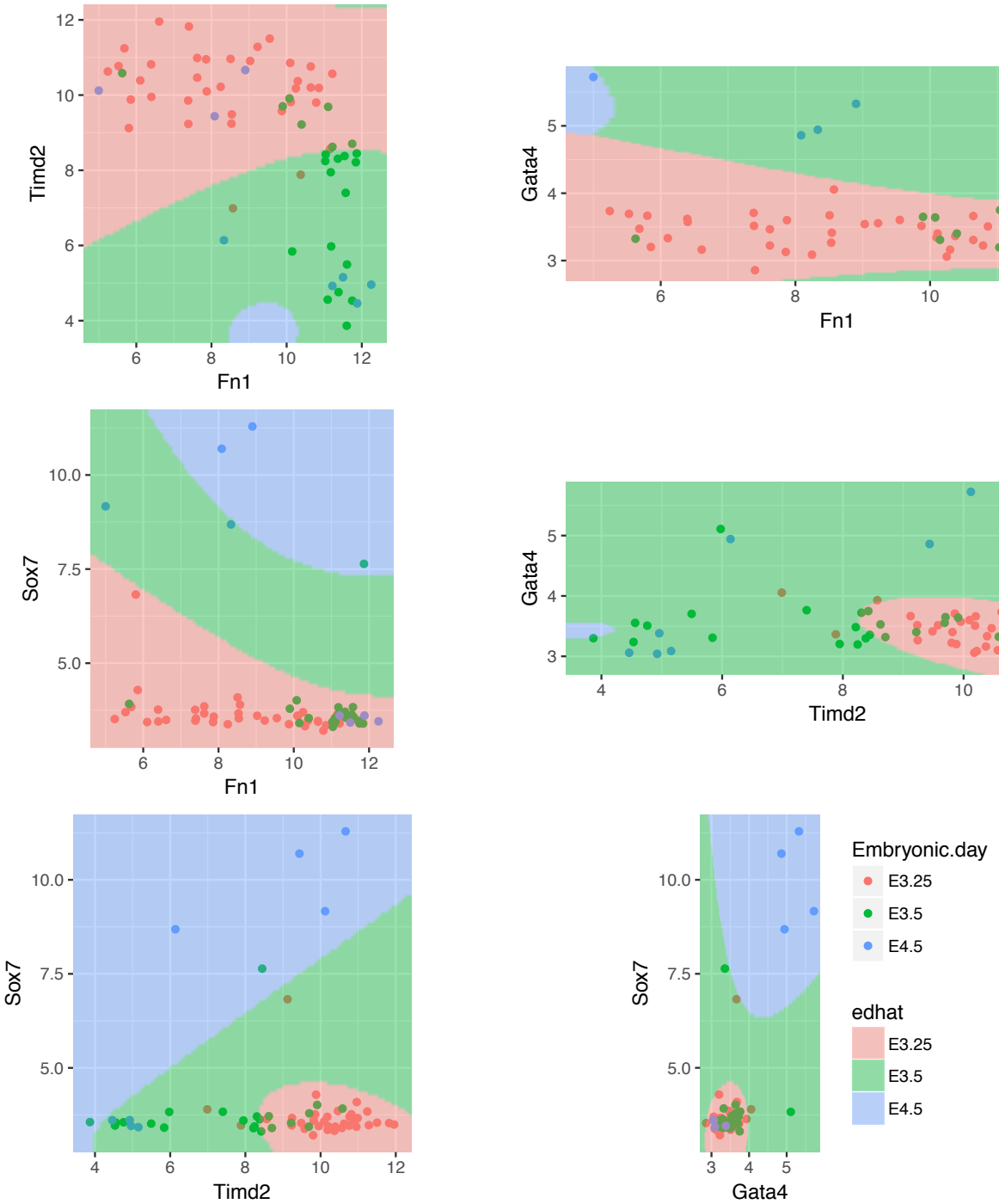
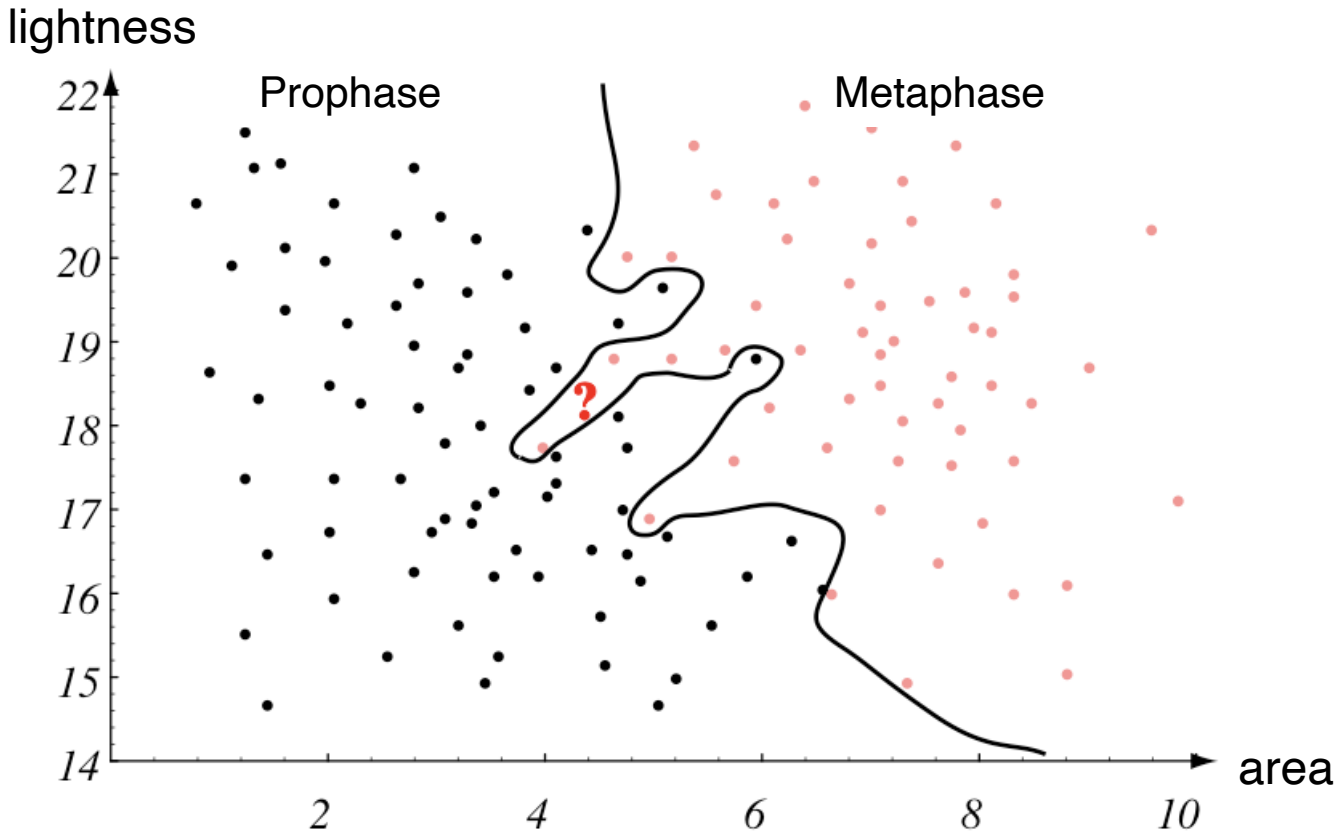


Figure 13.11: QDA for the mouse cell data. Shown are all pairwise plots of the four features.

k-Nearest-Neighbor Classifier

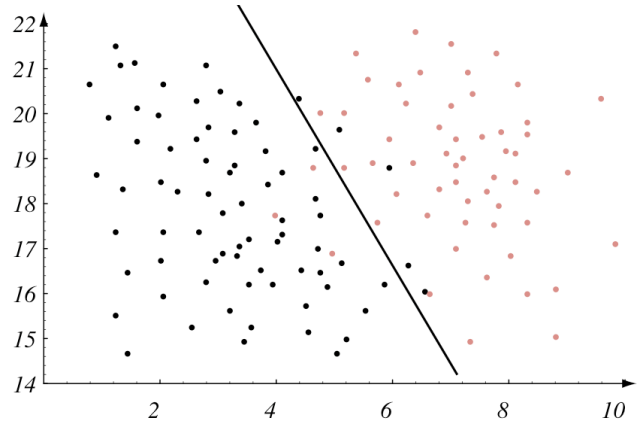


Assign each new cell to the class of its nearest neighbor.
Black line shows decision boundary

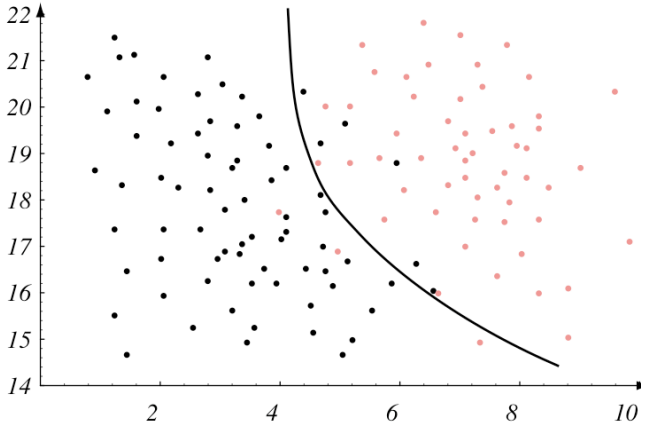
```
y[i] = +1 for pro phase
y[i] = -1 for meta phase
X[i,] = (area[i],lightness[i])
d = class::knn(X,Xnew,y,k=1)
```

Which Decision Boundary?

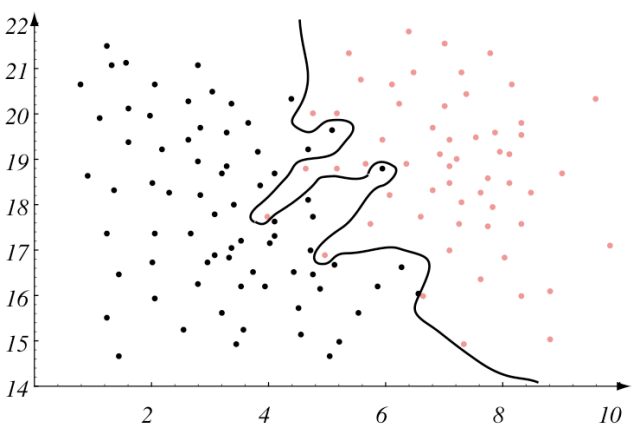
High bias
Low variance



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



Low bias
High variance



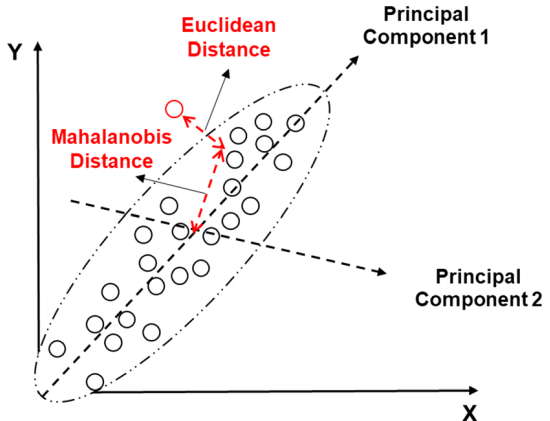
high model complexity
(need 100s of parameters to
describe decision boundary)

Which decision boundary has the lowest prediction error?

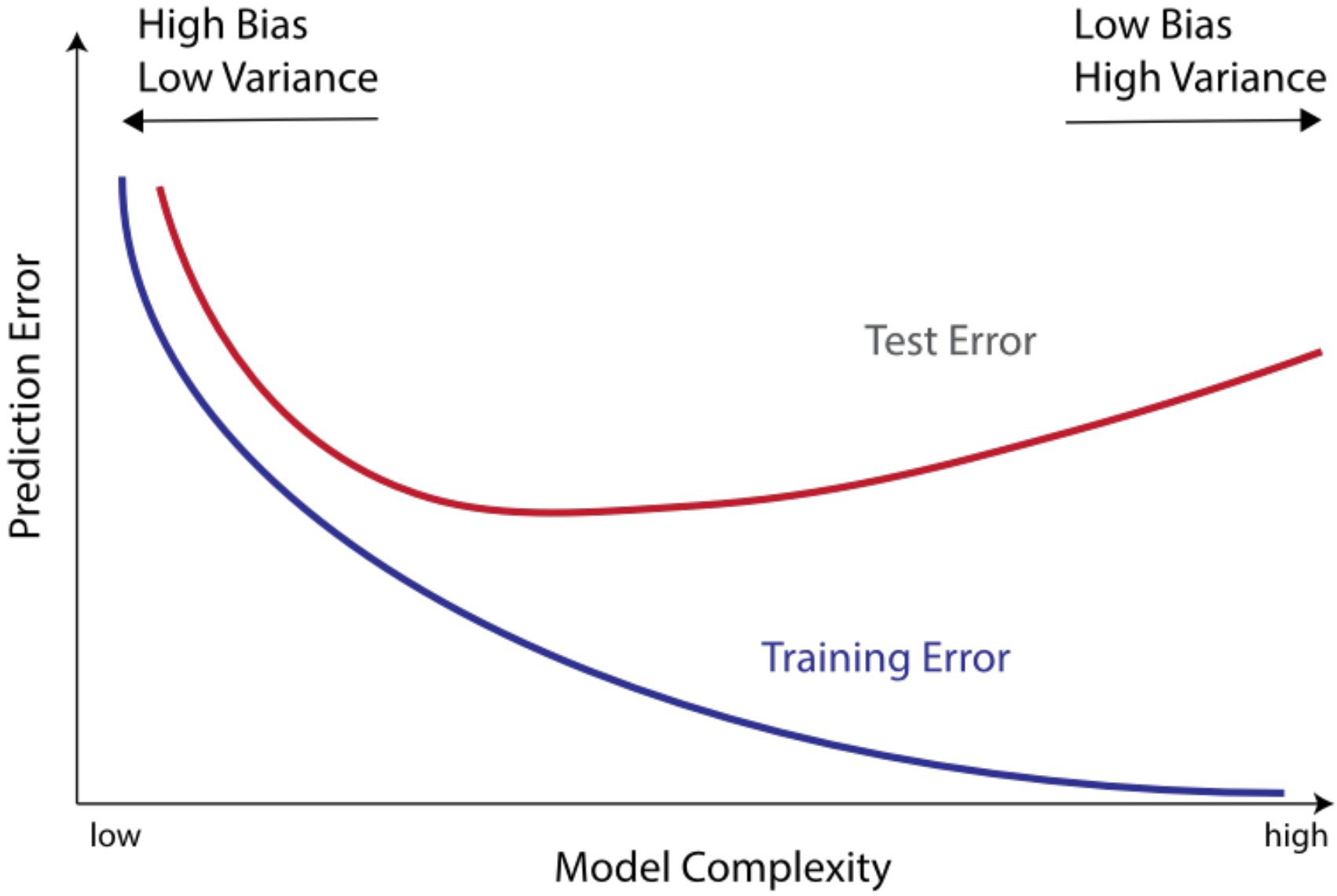
Two ways to think about classifiers

- One can think of classifiers as working in two different ways
- **One way** is to decide on (estimate) the decision boundary - thereby tiling the k dimensional space you are working in.
  - Then once you know what region you classify accordingly
- **An alternative:** decide where the cluster centroids are then assign according to whether the new observation is closer to one centroid or another.
  - You can use different measures of distance to effectively stretch or shrink in different directions (often accounting for different units of measurement, or directions of more variation in the data)

DOI:10.1007/s10845-019-01504-w



Bias-Variance-Dilemma



Building an AI/ML Model

The typical process involves identifying a machine learning algorithm that seems appropriate

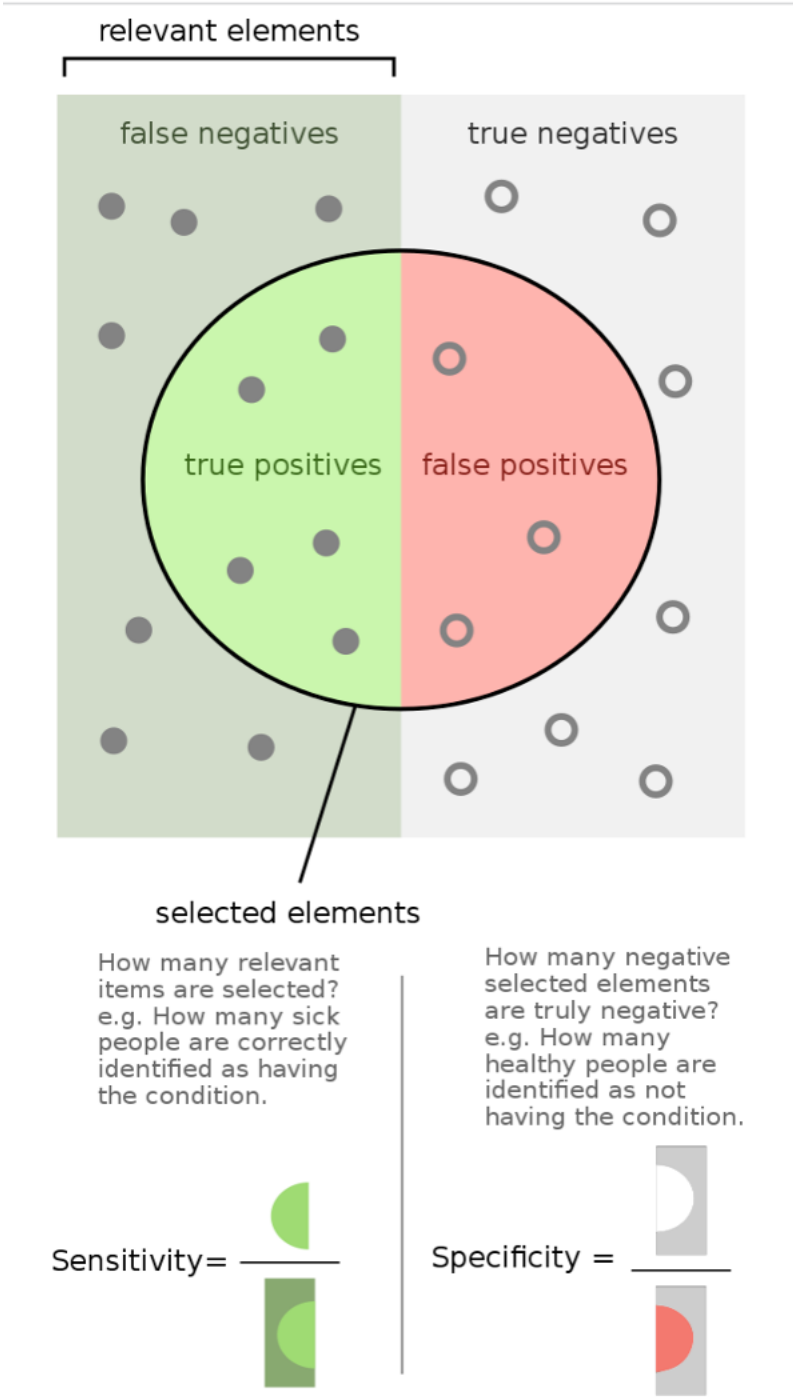
- eg convolutional neural networks, xgboost, etc
- in most cases there are:
- **model parameters:** these are parameters that are directly estimated by the underlying model step
- **hyperparameters:** these are parameters the analyst must specify, eg how many layers in the neural network, how much smoothing to do, parameters that affect the "rate of learning"
- sometimes **feature selection** is carried out
- **hyperparameters** are often examined by choosing a range of possible values, then fitting with each one and selecting one on the basis of some loss function (often using cross-validation)

Train/Test/Validate

- typical use is that the training data are used to fit model parameters
- testing data are used to adjust **hyperparameters** (eg how many layers in the neural network)
- validation is an independent set that is only used once a final model is chosen and it is used to assess prediction accuracy
- it is very typical for one large (homogeneous) data set to be used for all three
- the data are split, possibly at random – sometimes with some care – into three different data sets and labeled accordingly
- Usually **training >> testing > validation**

Sensitivity and Specificity

- in order to estimate sensitivity and specificity we typically divide our learning data into three groups
- G1: **training data** – usually the largest – fit model parameters
- G2: **testing data** – used to assess different hyperparameters
- G3: **validation data** – used to assess the overall predictions (eg sensitivity and specificity) with respect to the target population



Separation of Tasks

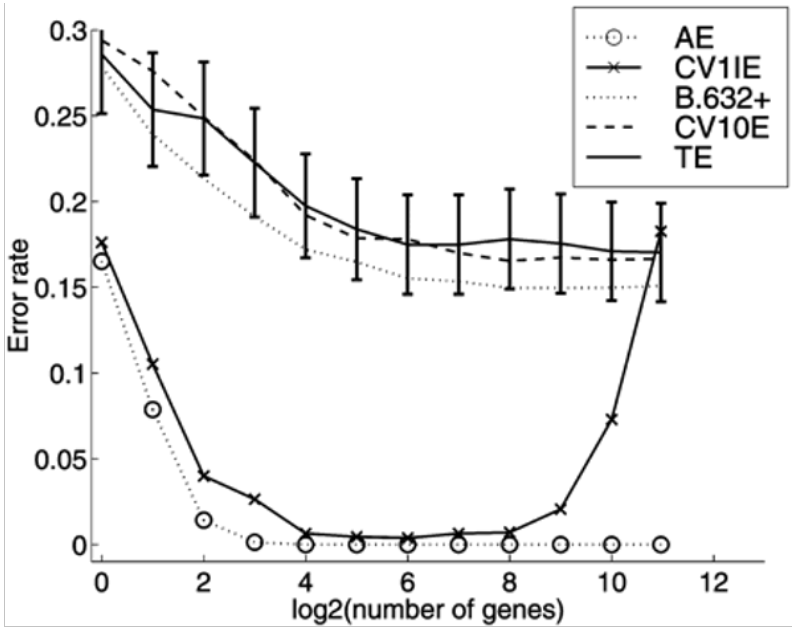
RESEARCH ARTICLE

Selection bias in gene extraction on the basis of microarray gene-expression data

Christophe Ambroise and Geoffrey J. McLachlan
+ See all authors and affiliations

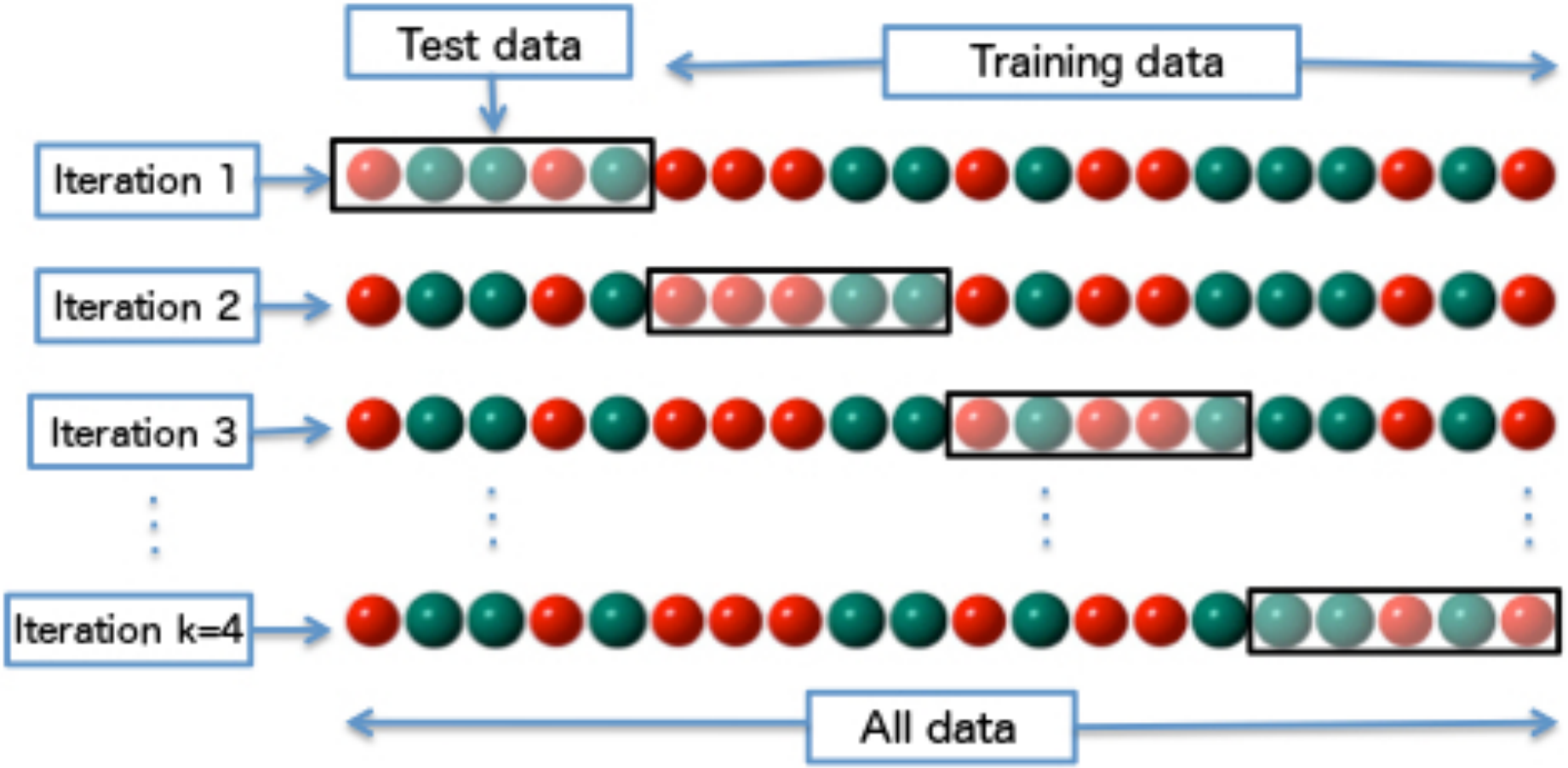
PNAS May 14, 2002 99 (10) 6562-6566; https://doi.org/10.1073/pnas.102102699

- one of the most important things to address is to ensure that you do not allow information to pass between different layers
- early experience with microarrays where feature selection was carried out using all the data, then data were colon cancer classification based on 62 samples (40 disease, 22 normal)
- microarray data – thousands of genes



- colon cancer classification based on 62 samples (40 disease, 22 normal)
- microarray data – thousands of genes
- the authors show how biased our estimates of the error rate are when feature selection is not included in each step of the cross-validation process

Cross-Validation



Wikipedia

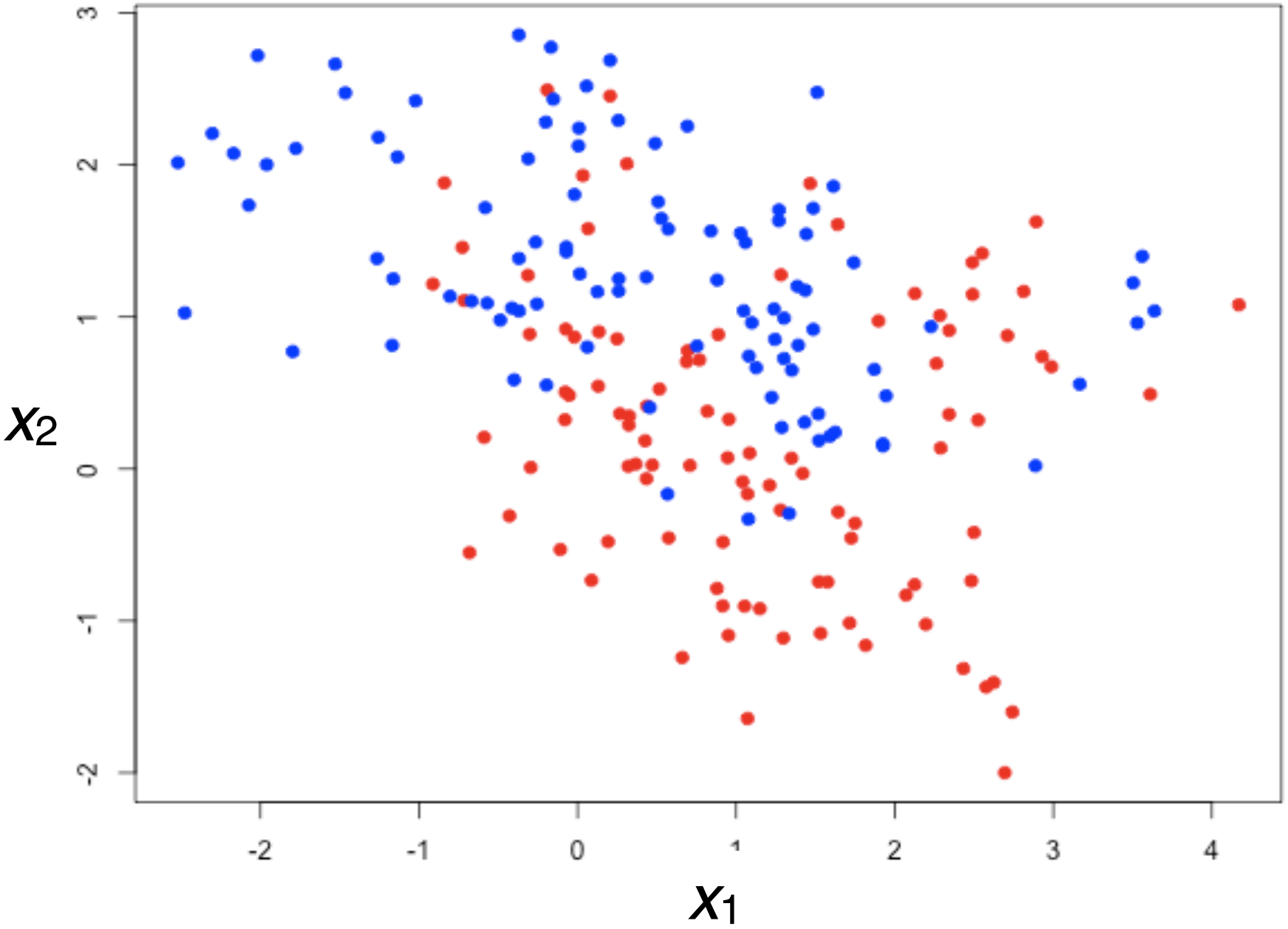
Cross-Validation

- Divide the data into a training, test and validation data set
- You should have some stochastic component
- You need to worry about balance if there are important features that are rare in the population (eg red hair)
- Cross-validation helps you choose hyper-parameters
- There is some overfitting as you are using the same data to select parameters as to train, test
- Dividing one large data set into three groups tends to lead to an overly optimistic impression of the operating characteristics

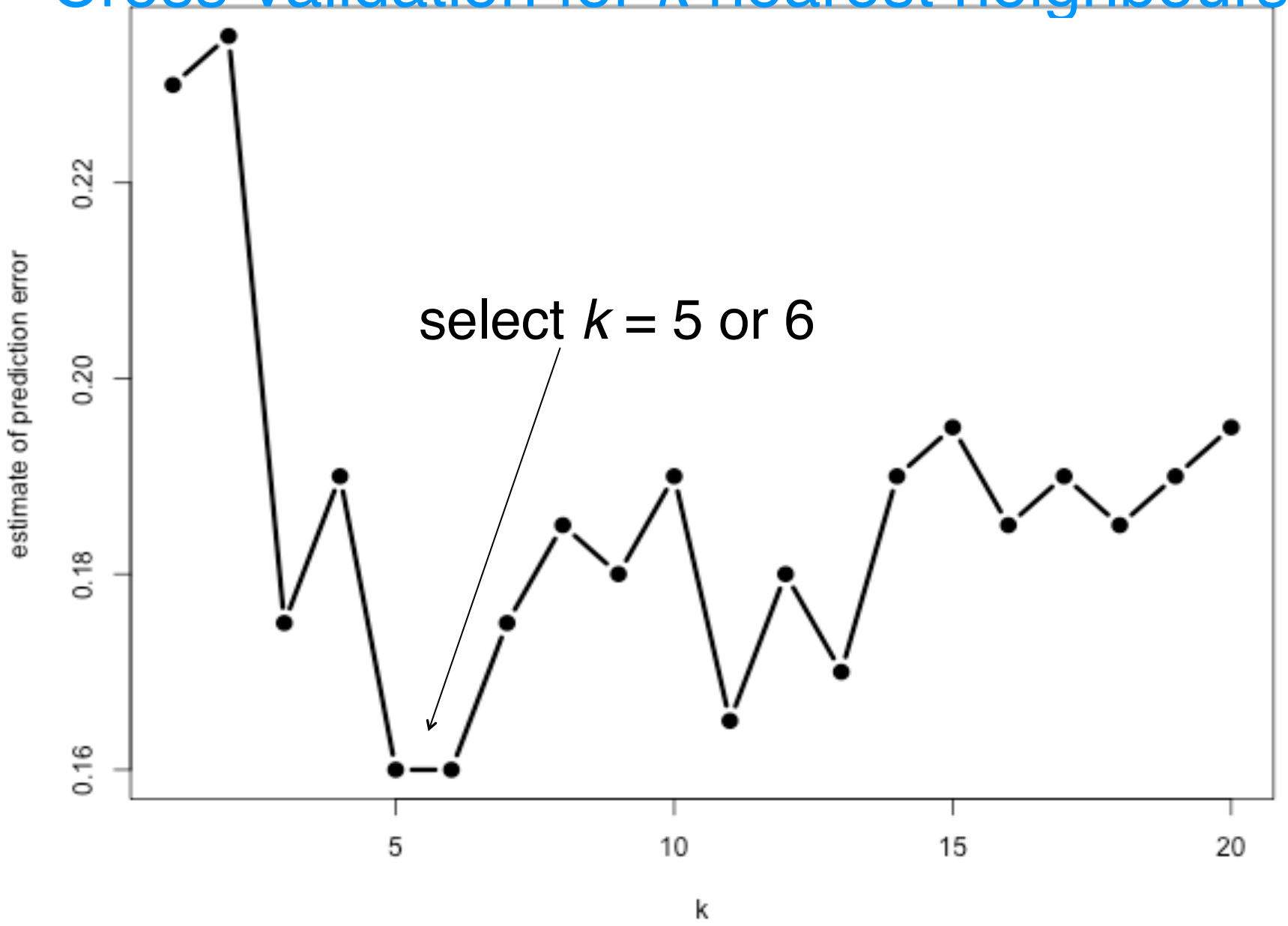
Loss Functions

- In many cases not all errors are equal and you may need to balance the cost of the error against the probability of making it
- Eg: screening for serious diseases - the cost of a FP is not as big as the cost of a FN (and for other diseases it can be the other way)
- Predicting rare conditions: eg we want to use genetics to predict the probability that someone has red hair. In the population of interest about 10% of people have red hair.
  - So a naive classifier - "No one has red hair" is right 90% of the time….
  - We might want to use a loss function that balances the errors differently in order to ensure that we are predicting the outcome of interest….

Example: 2 classes, 2 variables, 200 objects

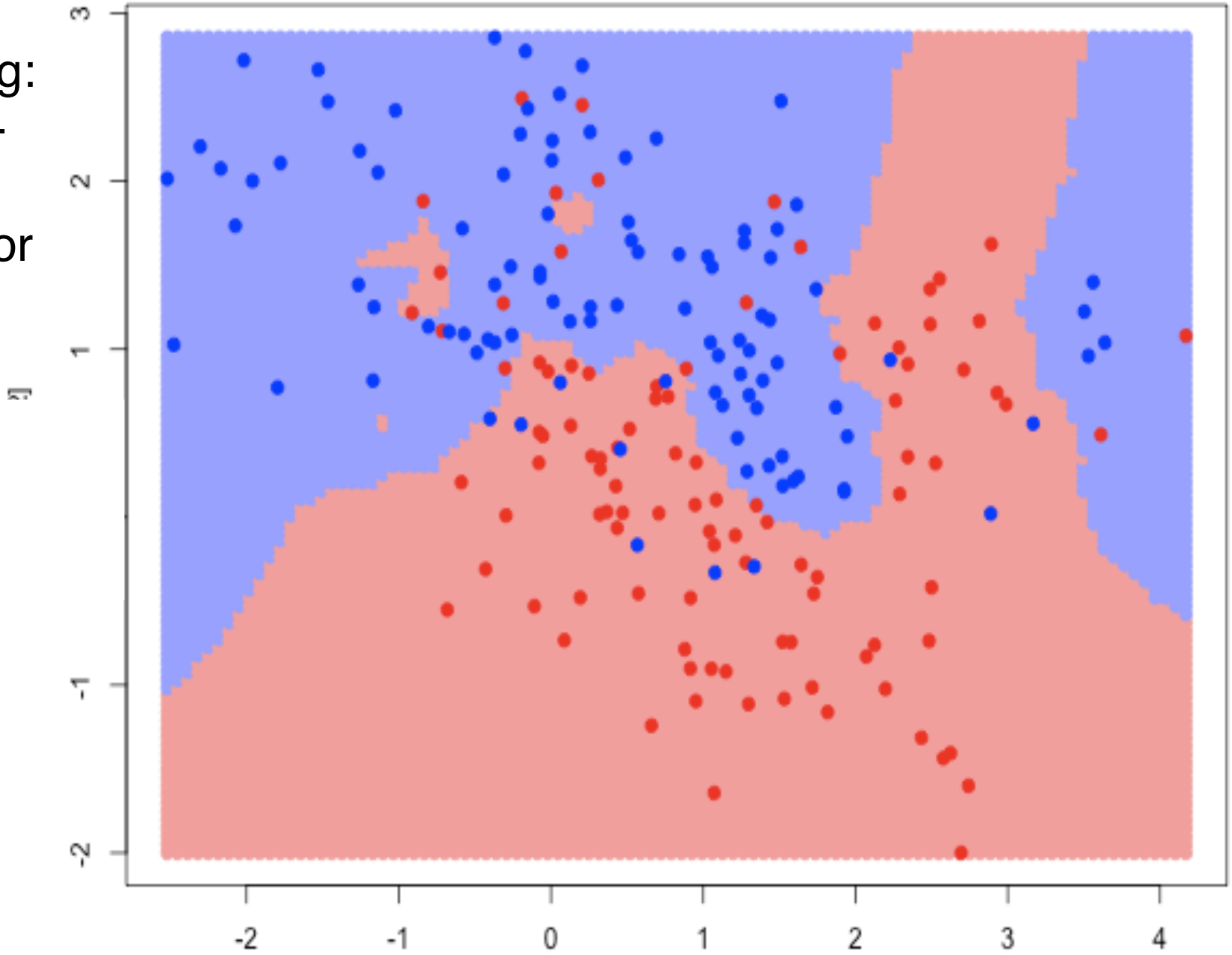


Cross-validation for *k* nearest neighbours



Cross-validation for *k* nearest neighbours

Shading: classifi-cation result for $k = 5$



Come back to the linear 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 *d* matrix of indicator variables

Class	1	2	3
a	1	0	0
b	0	1	0
c	0	0	1
b	0	1	0
a	1	0	0

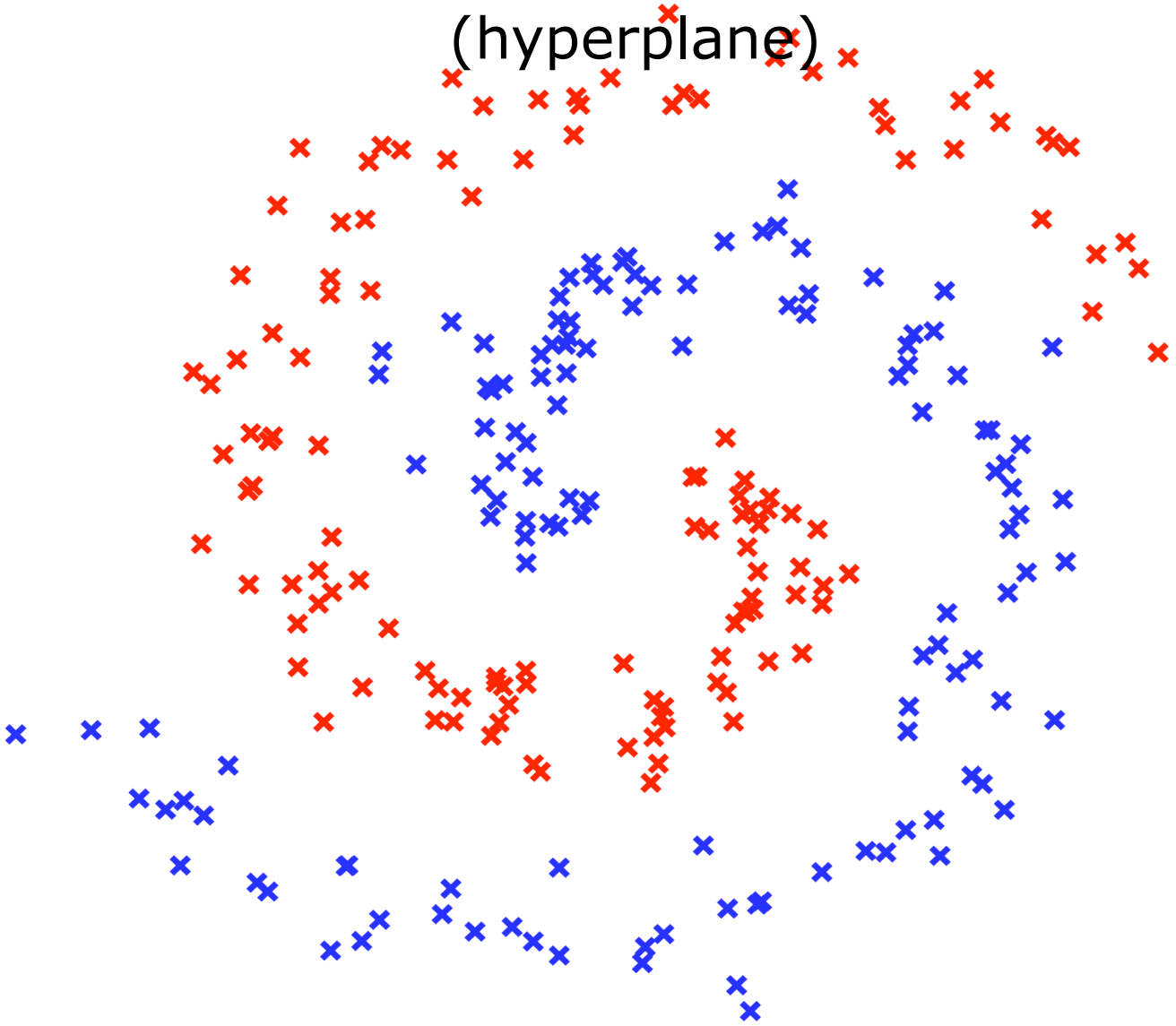
In practice: lda (R-package MASS)

Some Other Considerations

- In some cases we are very certain about the classification of a new observations, but in other cases we might want to express some amount of "doubt".
  - The point might lie very close to the boundary
- Sometimes you might want to just refuse to attempt to classify - perhaps because the new data is very different from the data used to train the classifier.
  - This requires access to the original training data - and there are some reasons why that information should be retained and used.

Non-Linear Classifiers?

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



We could either

- bend our decision boundaries to be curvy, or
- bend our data space and stick with linear boundaries.

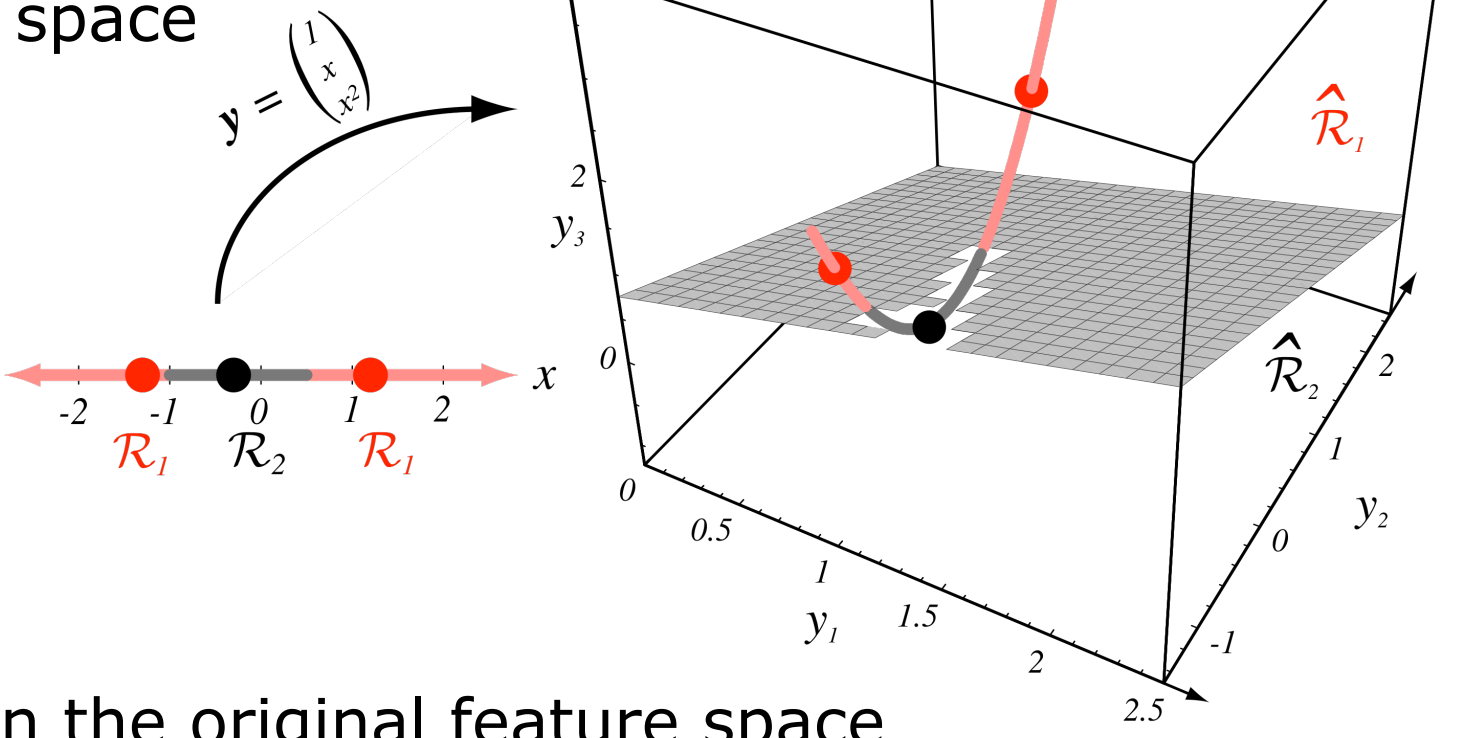
Guess what is

Data Transformation & Augmentation

Apply non-linear functions, e.g.

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

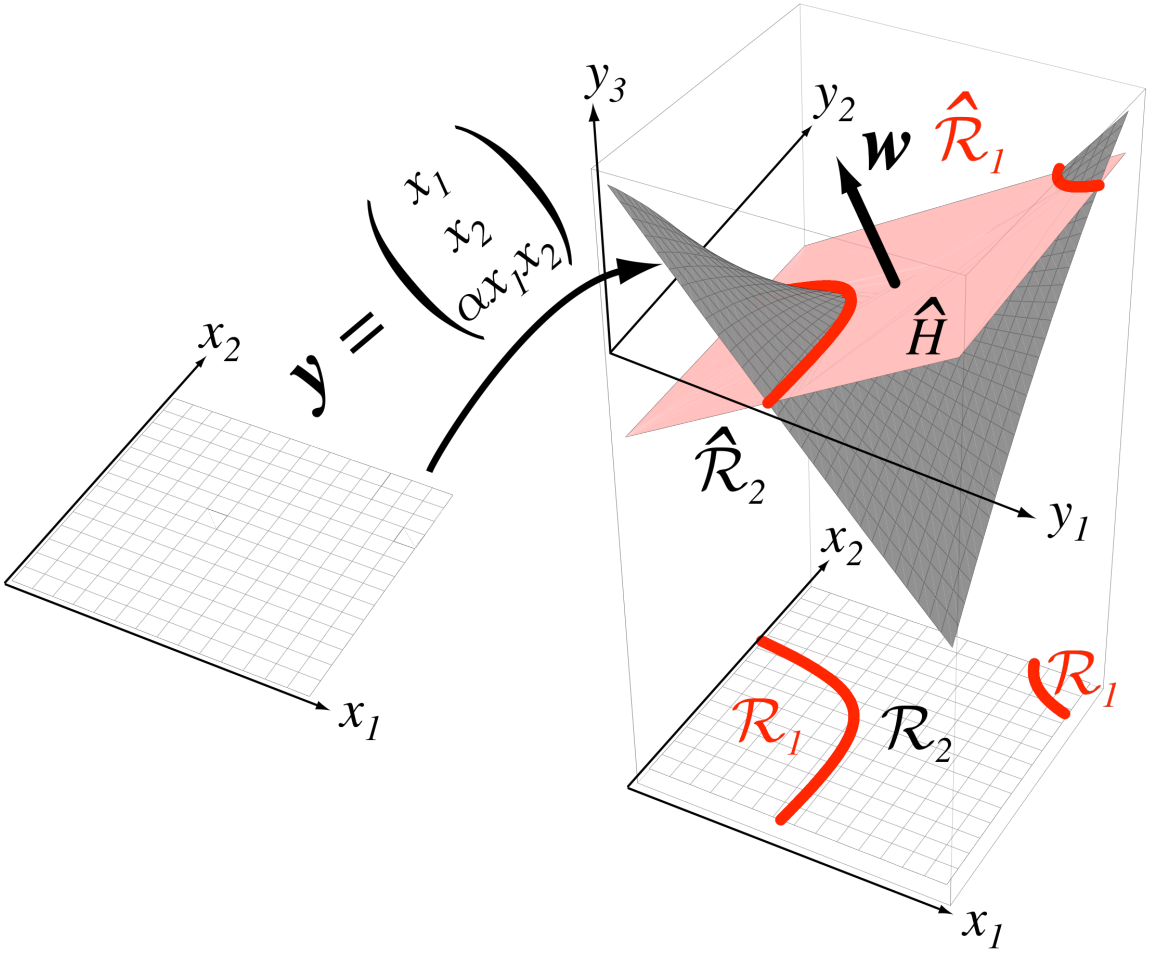
Train linear classifier in the new feature space



non-linear classifier in the original feature space

Quadratic Extension

Parabolic decision boundaries can be achieved by using the product



The Kernel Trick

You don’t even need to do the augmentation explicitly.

Remember that relative positions of all data points can be encoded by their distance matrix.

Now just replace Euclidean distance with some other, called “kernel”.

The Kernel Trick

Popular choices

Linear kernel

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

Radial basis functions

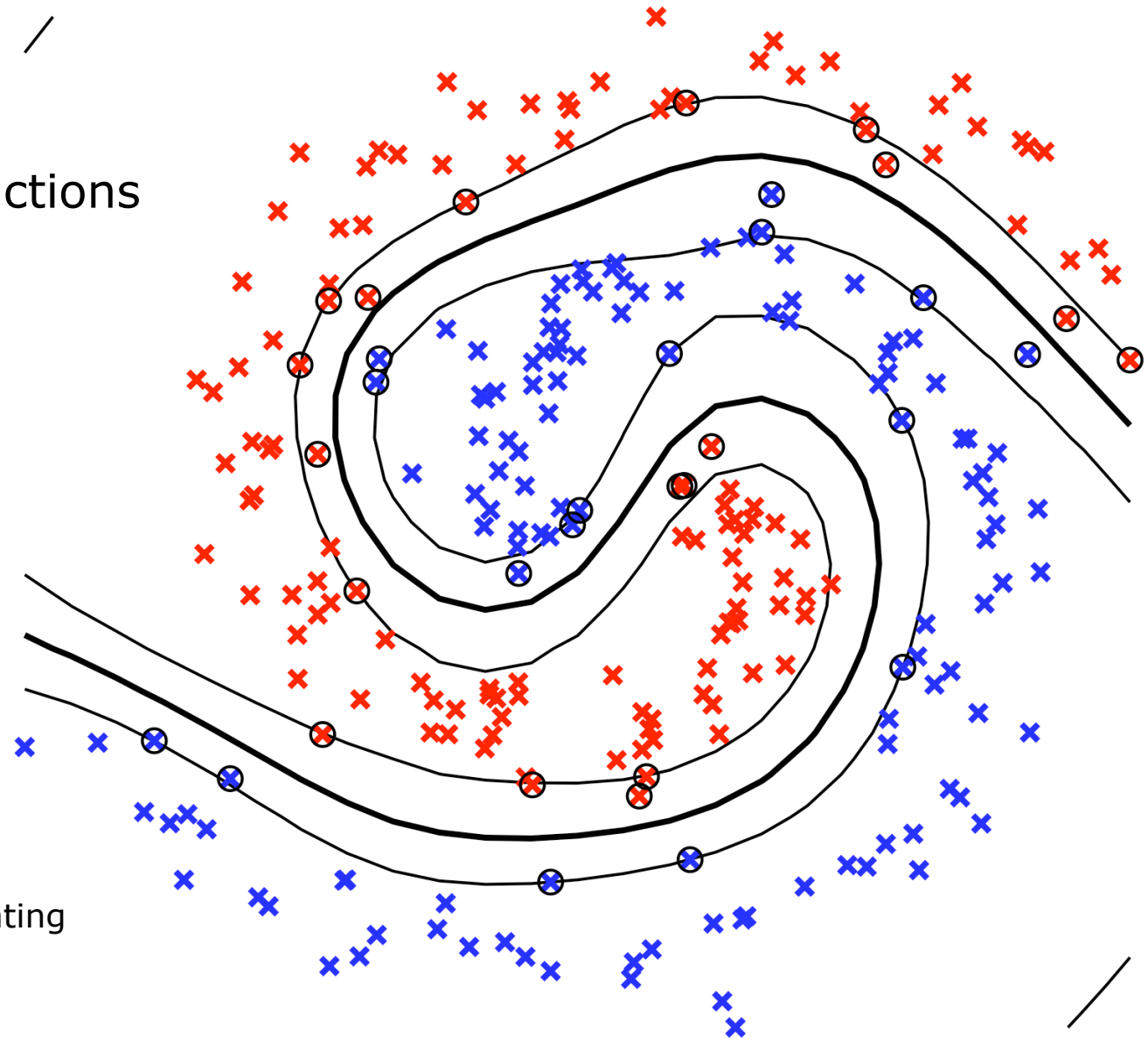
$$K(x_i, x_j) = \exp\left(-\frac{1}{2\sigma^2}\|x_i - x_j\|\right)$$

Polynomial

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

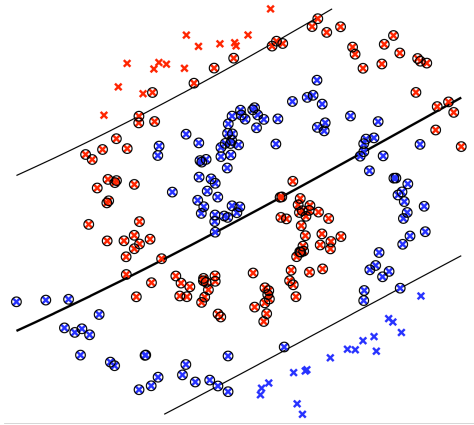
Example

Radial Basis Functions Kernel

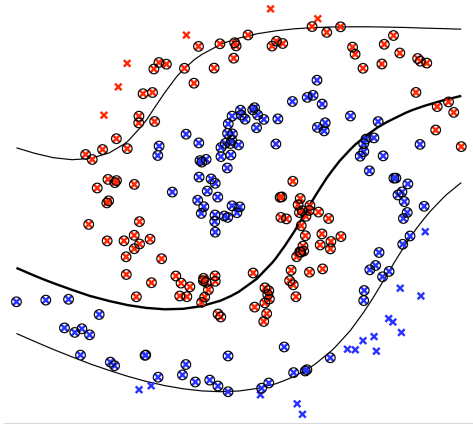


(SVM
Thick line: class separating hyperplane
Thin line: margin
Circles: support vectors)

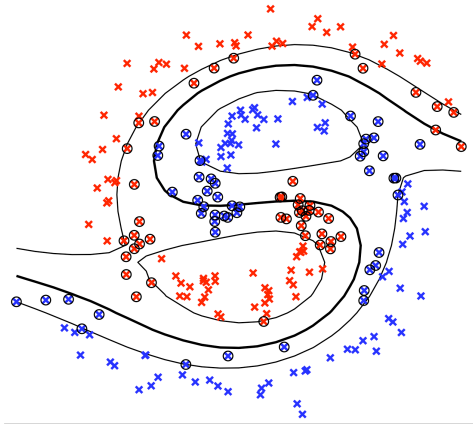
The Influence of the Kernel Parameter



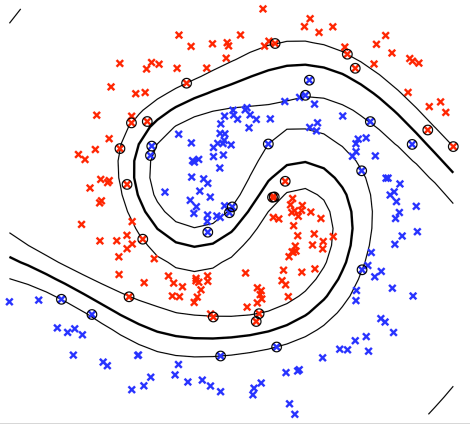
$\gamma = 0.001$



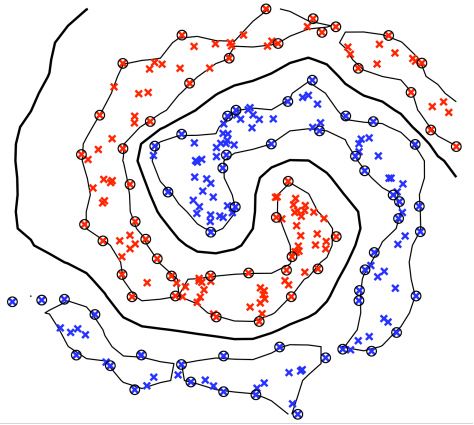
$\gamma = 0.005$



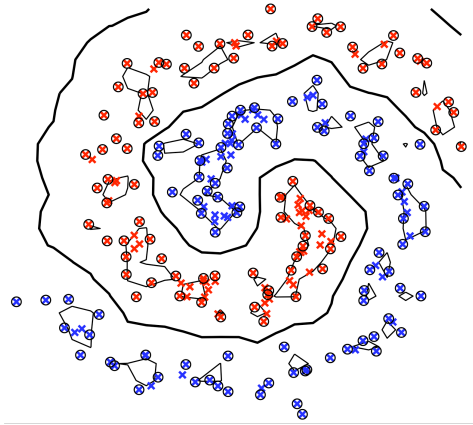
$\gamma = 0.03$



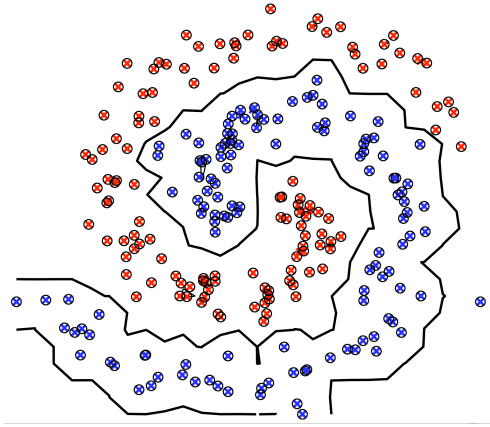
$\gamma = 0.1$



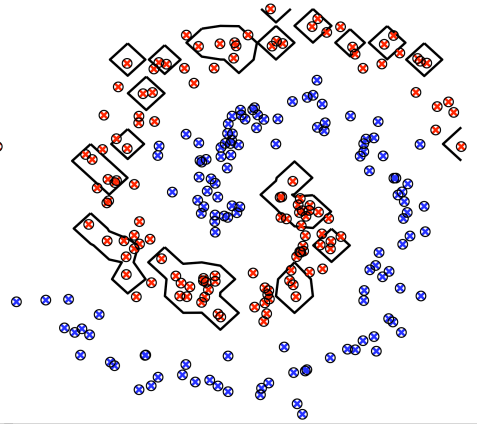
$\gamma = 1$



$\gamma = 2$



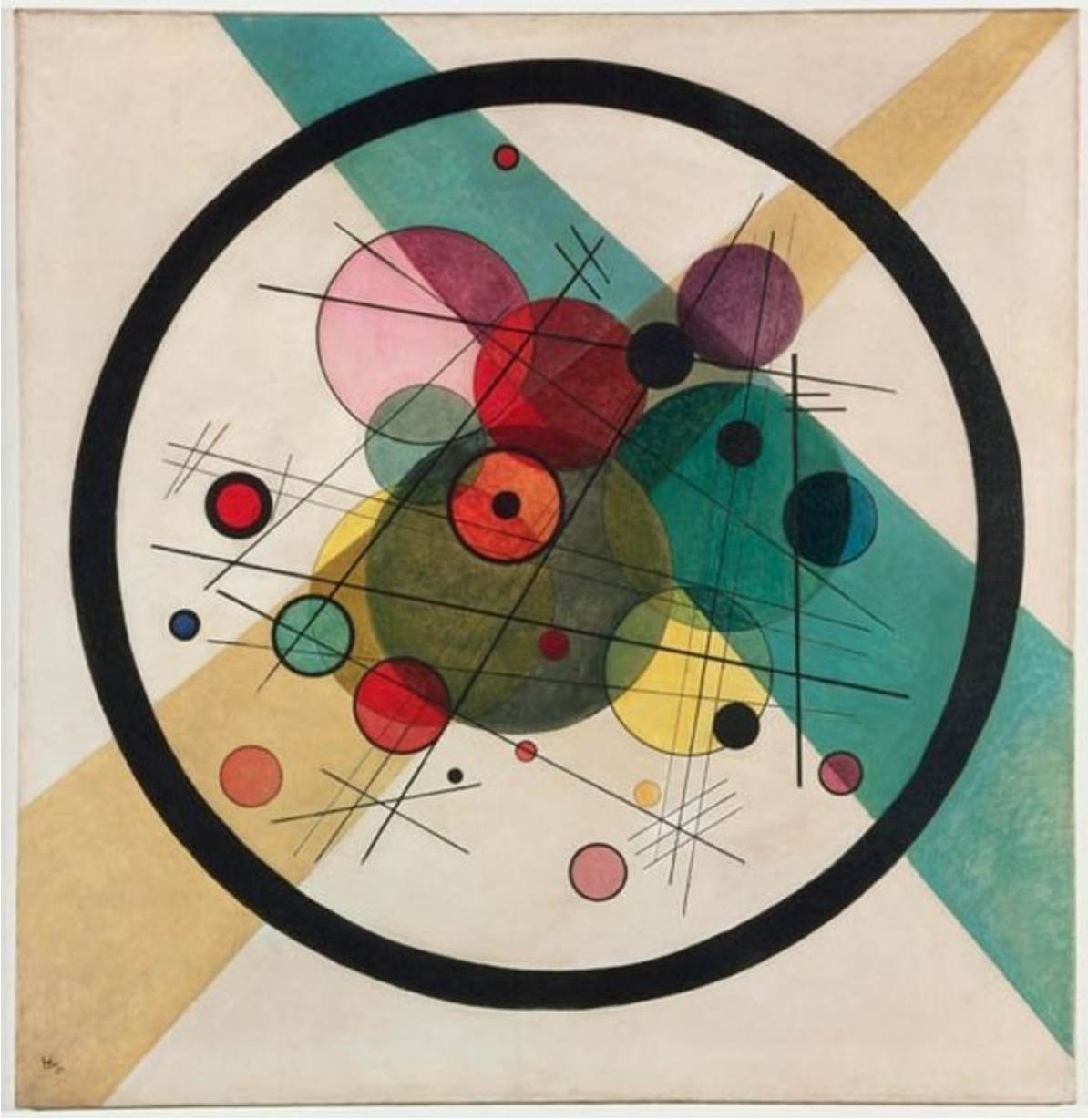
$\gamma = 20$

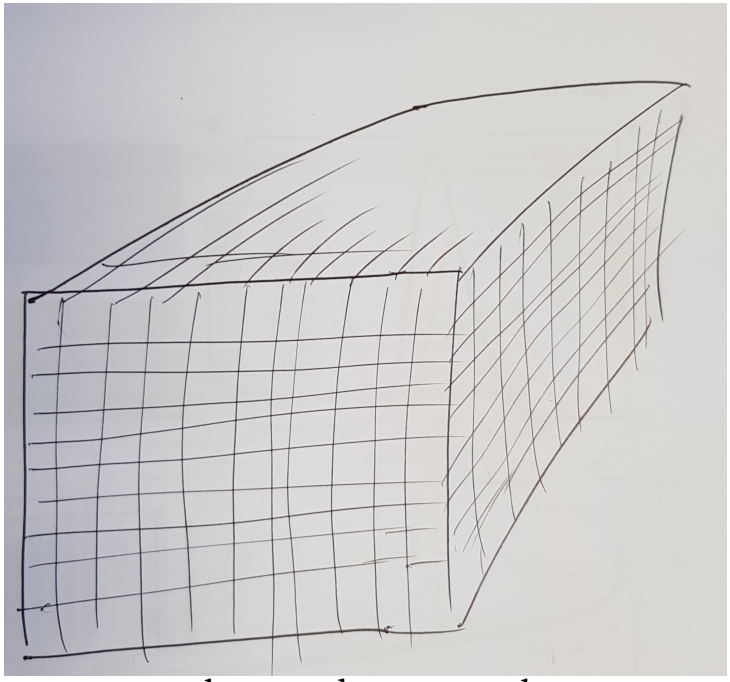


$\gamma = 200$

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

The curse of dimensionality



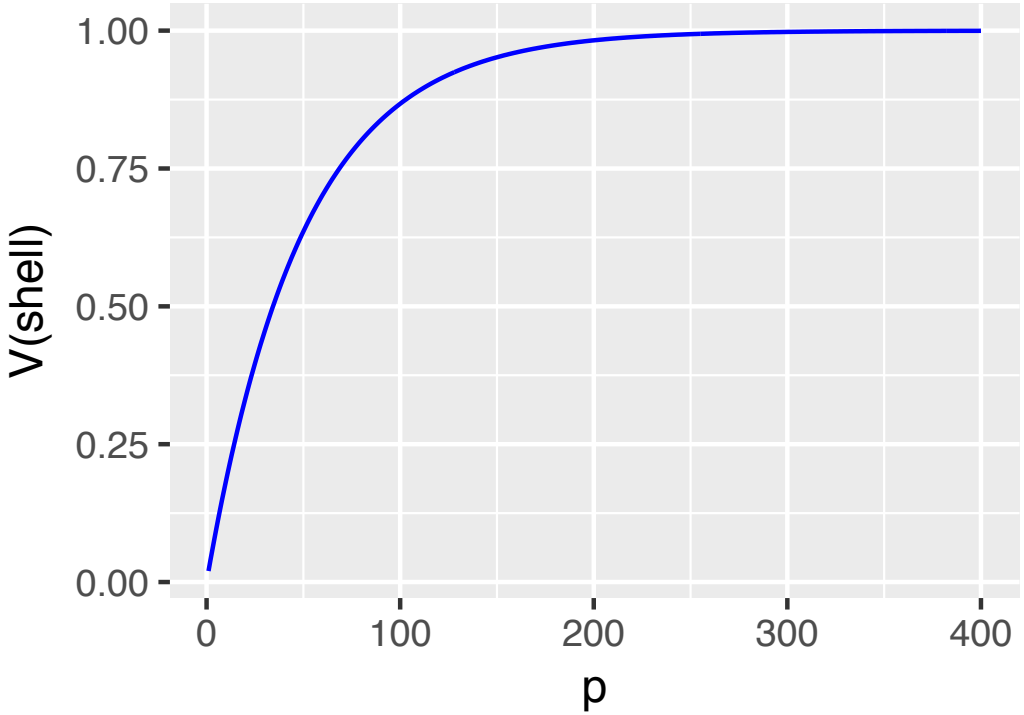


▶ Question **13.12** Assume you have a dataset with 1 000 000 data points in p dimensions. The data are uniformly distributed in the unit hybercube (i. e., all features lie in the interval $[0, 1]$). What's the side length of a hybercube that can be expected to contain just 10 of the points, as a function of p? ◀

▶ Solution **13.12**

See Figure 13.16.

```
sideLength = function(p, pointDensity = 1e6, pointsNeeded = 10)
   (pointsNeeded / pointDensity) ^ (1 / p)
ggplot(tibble(p = 1:400, sideLength = sideLength(p)),
       aes(x = p, y = sideLength)) + geom_line(col = "red") +
  geom_hline(aes(yintercept = 1), linetype = 2)
```

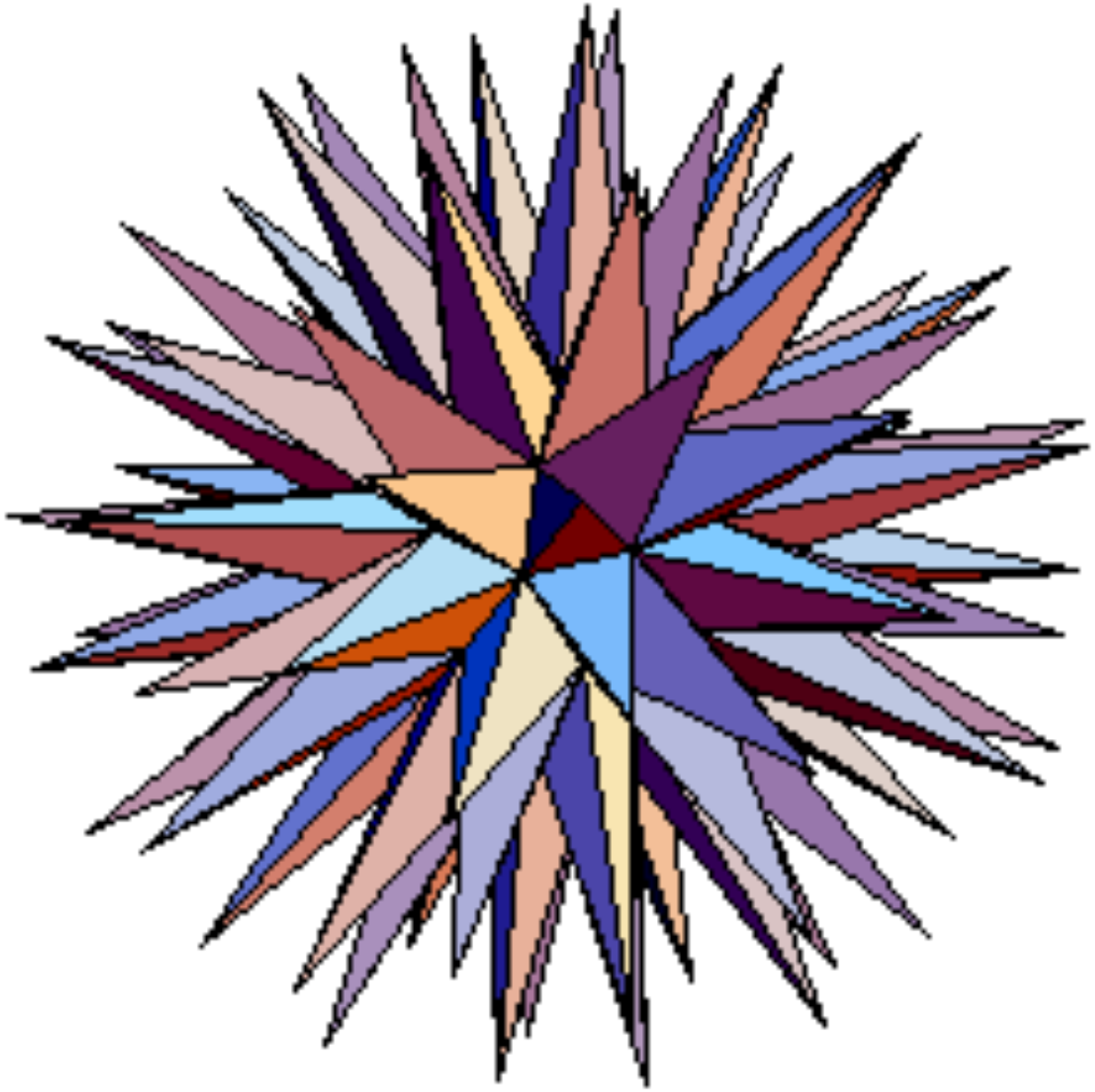


▶ Question **13.13** What fraction of a unit cube's total volume is closer than 0.01 to any of its surfaces, as a function of the dimension? ◀

▶ Solution **13.13**

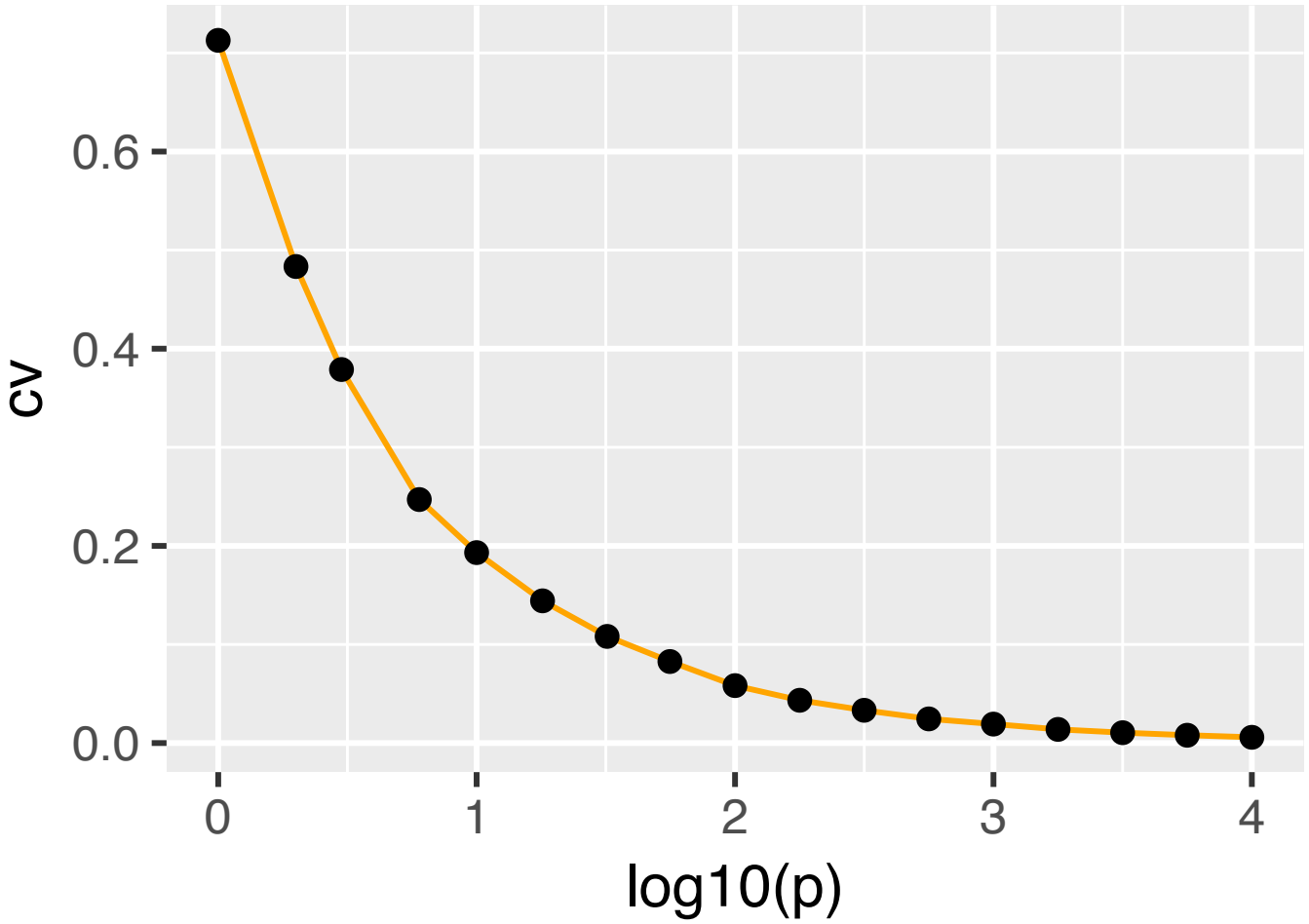
See code below and Figure 13.17.

```
tibble(
  p = 1:400,
  volOuterCube = 1 ^ p,
  volInnerCube = 0.98 ^ p,   # 0.98 = 1 - 2 * 0.01
  `V(shell)` = volOuterCube - volInnerCube) %>%
ggplot(aes(x = p, y =`V(shell)`)) + geom_line(col = "blue")
```



An attempt to visualize a 7-dim hypercube ($2^7 = 128$ corners)

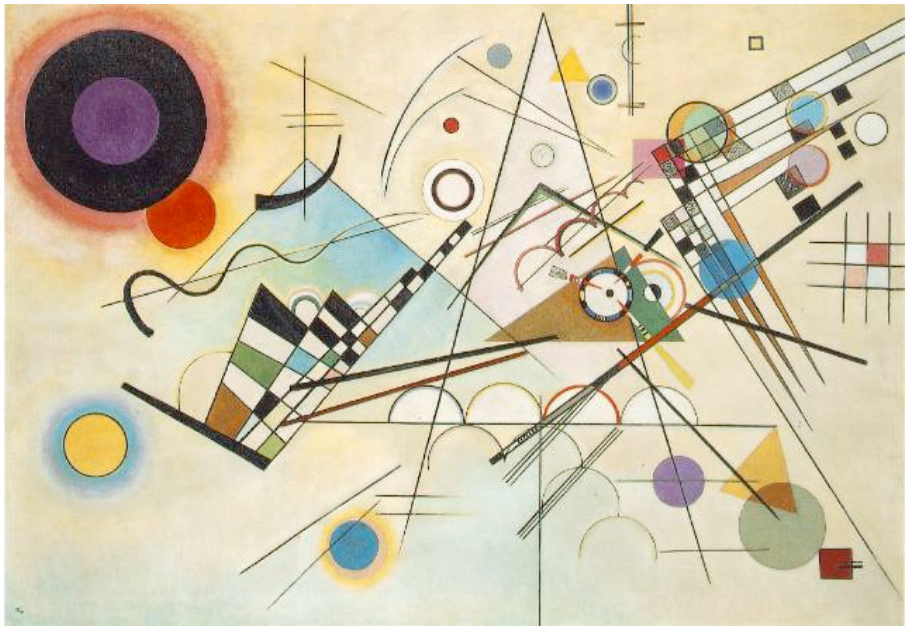
http://yaroslavvb.blogspot.com/2006/05/curse-of-dimensionality-and-intuition.html



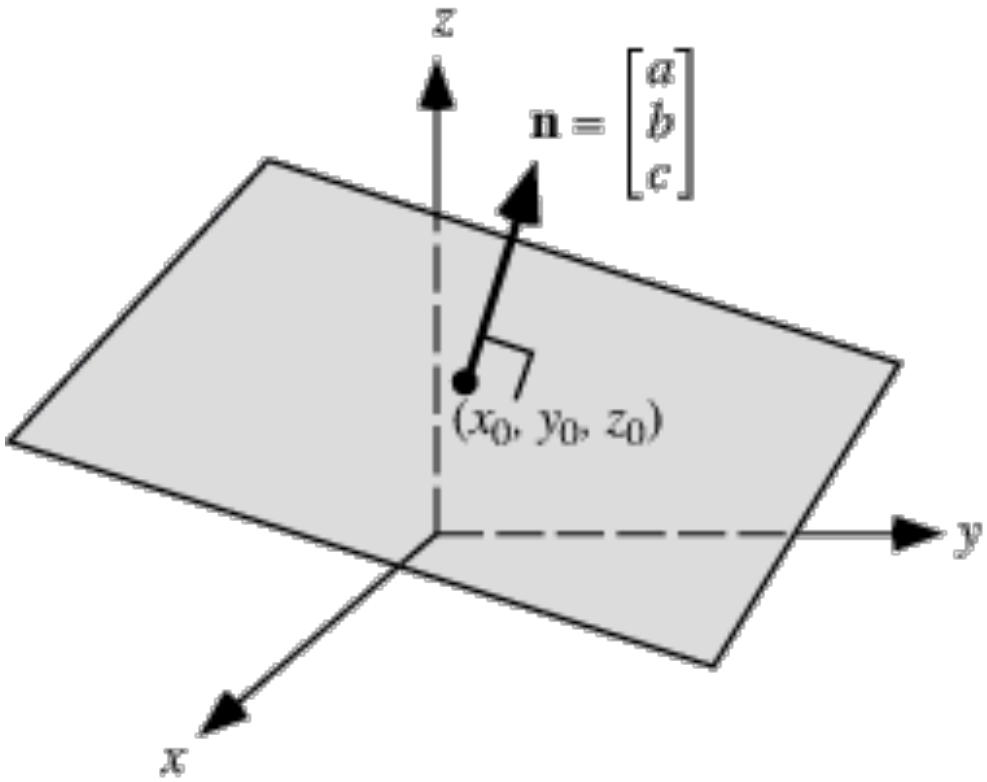
▶ Question **13.14** What is the coefficient of variation (ratio of standard deviation over average) of the distance between two randomly picked points in the unit hypercube, as a function of the dimension? ◀

Curse of Dimensionality: overfitting guaranteed

- Consider:
  - 10 samples per class
  - Each sample is characterised by several hundred features.
- Even a linear classifier will (always) be too complex: overfitting



Regularisation



Remember that a plane in 3D space can be represented by its normal vector.

Same for *n*-dim. space

Idea: rather than allowing general vectors, ask for many of the coefficients to be small, or even zero.

E.g.: decision rule:

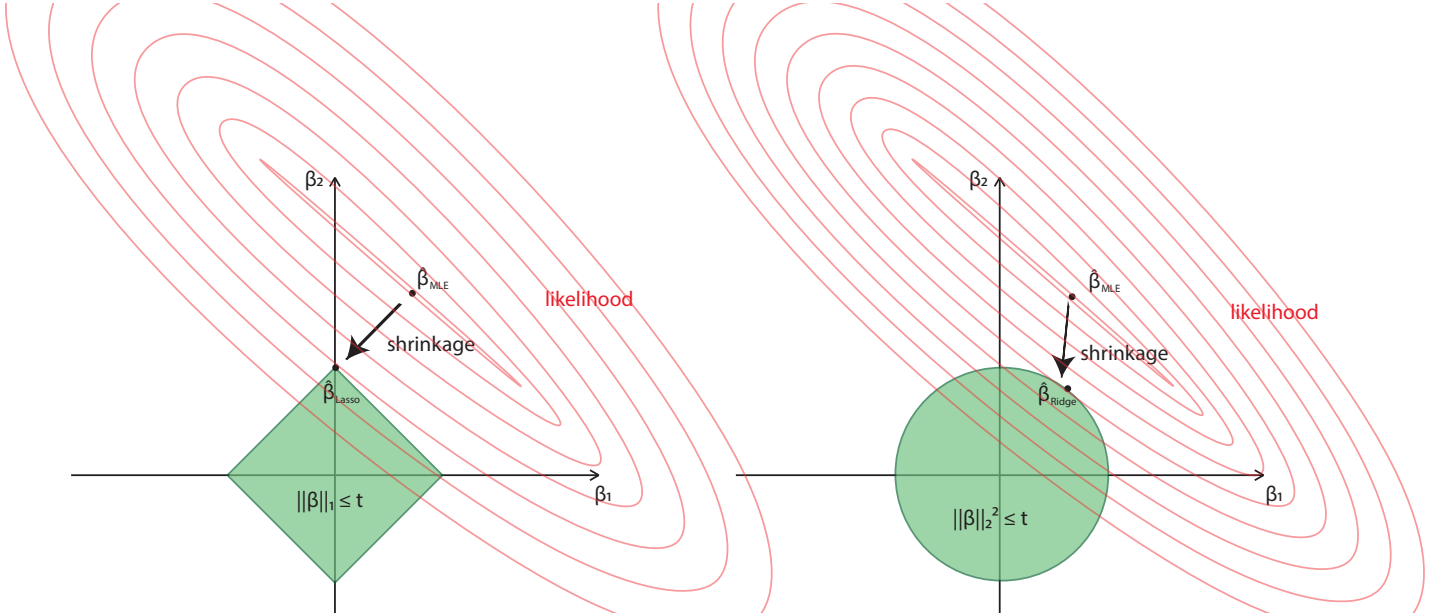
$\mathbf{x}\,\mathbf{n} - 1 > 0$

Commonly used penalizations and their geometry

Lasso estimator: $p(\beta) = ||\beta||_1$ **Ridge estimator**: $p(\beta) = ||\beta||_2^2$

$$\hat{\beta} \in \arg\min_{\beta} \ell(\beta) + \lambda||\beta||_1 = \arg\min_{\beta} \ell(\beta) \; s.t. ||\beta||_1 \leq t,$$

$$\hat{\beta} \in \arg\min_{\beta} \ell(\beta) + \lambda||\beta||_2^2 = \arg\min_{\beta} \ell(\beta) \; s.t. ||\beta||_2^2 \leq t,$$



- Ridge just wants the $\boldsymbol{\beta}$ to be small
- Lasso snaps many of the elements to 0 ('sparsity')

(least absolute shrinkage and selection operator)

13.6.2 Example: predicting colon cancer from stool microbiome composition

Zeller et al. (2014) studied metagenome sequencing data from fecal samples of 156 humans that included colorectal cancer patients and tumor-free controls. Their aim was to see whether they could identify biomarkers (presence or abundance of certain taxa) that could help with early tumor detection. The data are available from Bioconductor through its **ExperimentHub** service under the identifier EH359.

```
library("ExperimentHub")
eh = ExperimentHub()
zeller = eh[["EH361"]]
```

```
table(zeller$disease)
```

```
##
##        cancer large_adenoma             n small_adenoma
##            53            15            61            27
```

```
pData(zellerNC)[ sample(ncol(zellerNC), 3), ]

##                    subjectID age gender bmi country disease
## CCIS71578391ST-4-0     FR-187  70   male  25  france       n
## CCIS50003399ST-4-0     FR-194  66 female  28  france       n
## CCIS38765456ST-20-0    FR-723  79 female  22  france  cancer
##                    tnm_stage ajcc_stage localization     fobt
## CCIS71578391ST-4-0       <NA>       <NA>         <NA> negative
## CCIS50003399ST-4-0       <NA>       <NA>         <NA> negative
## CCIS38765456ST-20-0    t4n1m1         iv           lc positive
##                    wif-1_gene_methylation_test   group
## CCIS71578391ST-4-0                    negative control
## CCIS50003399ST-4-0                    negative control
```

```
rownames(zellerNC)[1:4]

## [1] "k__Bacteria"
## [2] "k__Viruses"
## [3] "k__Bacteria|p__Firmicutes"
## [4] "k__Bacteria|p__Bacteroidetes"

rownames(zellerNC)[nrow(zellerNC) + (-2:0)] %>% formatfn

## [[1]]
## [1] "k__Bacteria| p__Proteobacteria| c__Deltaproteobacteria|"
## [2] "o__Desulfovibrionales| f__Desulfovibrionaceae|"
## [3] "g__Desulfovibrio| s__Desulfovibrio_termitidis"
##
## [[2]]
## [1] "k__Viruses| p__Viruses_noname| c__Viruses_noname|"
## [2] "o__Viruses_noname| f__Baculoviridae|"
## [3] "g__Alphabaculovirus|"
## [4] "s__Bombyx_mori_nucleopolyhedrovirus|"
## [5] "t__Bombyx_mori_nucleopolyhedrovirus_unclassified"
##
## [[3]]
## [1] "k__Bacteria| p__Proteobacteria| c__Deltaproteobacteria|"
## [2] "o__Desulfovibrionales| f__Desulfovibrionaceae|"
## [3] "g__Desulfovibrio| s__Desulfovibrio_termitidis|"
## [4] "t__GCF_000504305"
```

glmnet on the Zeller data

```
library("glmnet"
glmfit = glmnet
```

```
predTrsf = predi
```

```
table(predTrsf,
```

```
##
## predTrsf canc
##    cancer
##    n
```

```
plot(glmfit, co
```

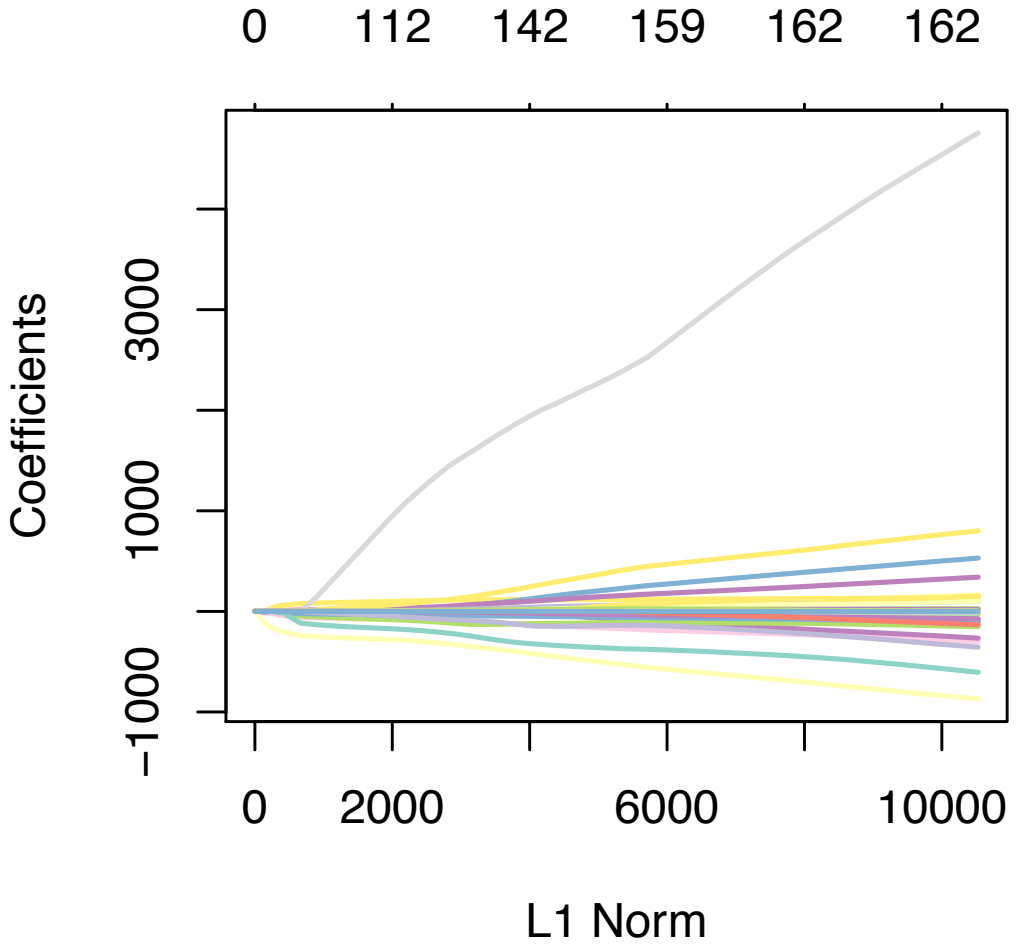
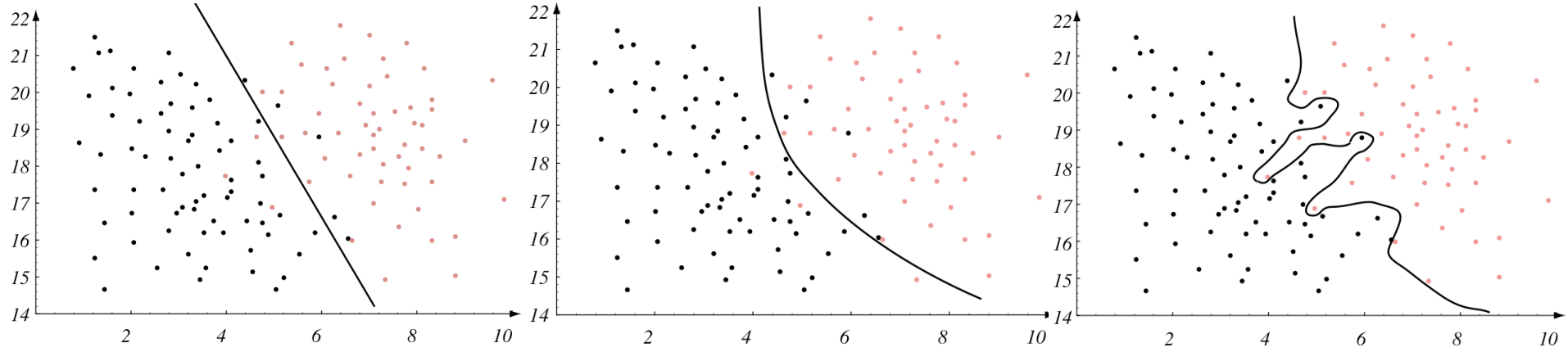


Figure 13.21: Regularization paths for `glmfit`.

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

High bias
Low variance

Low bias
High variance

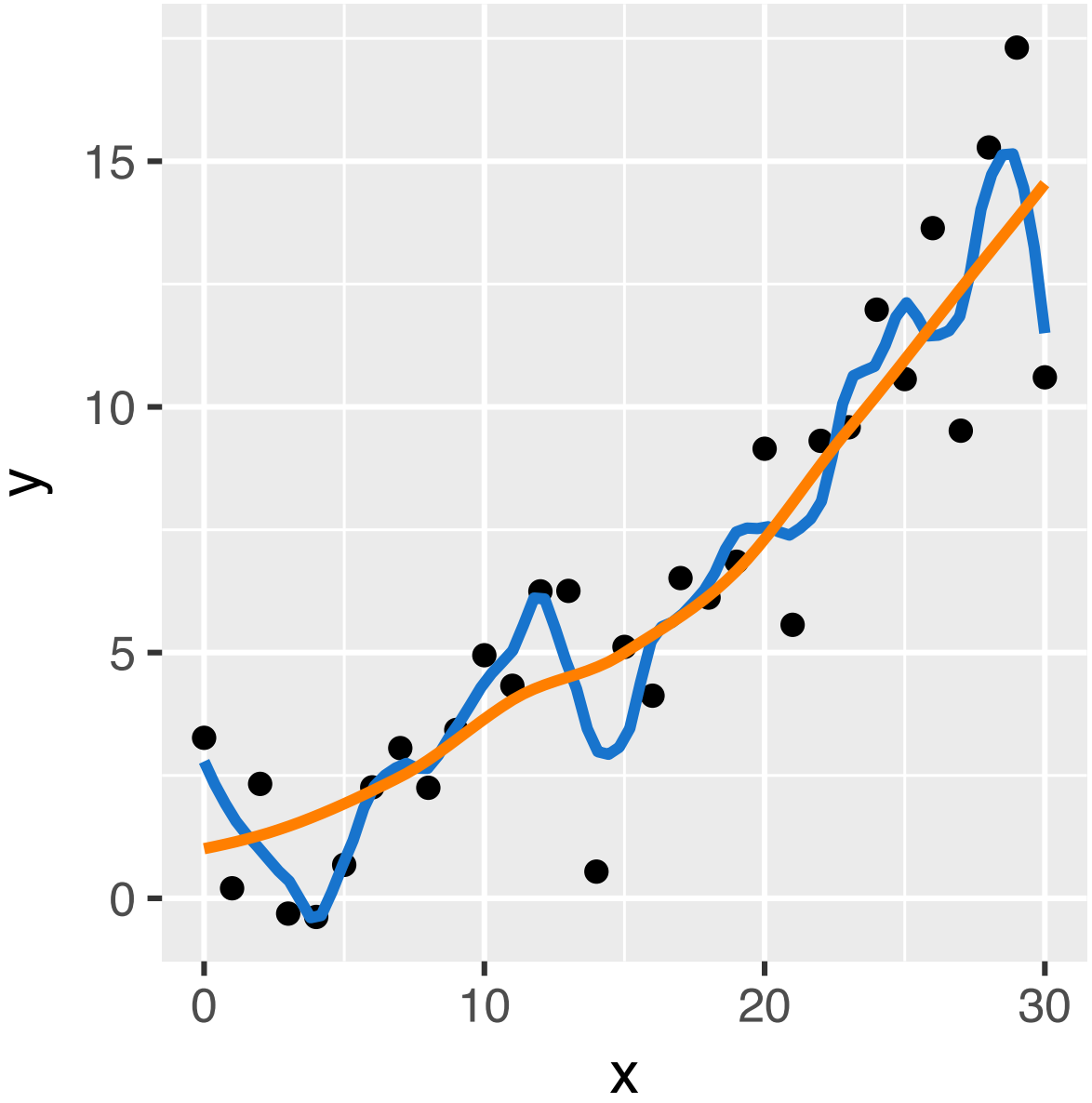


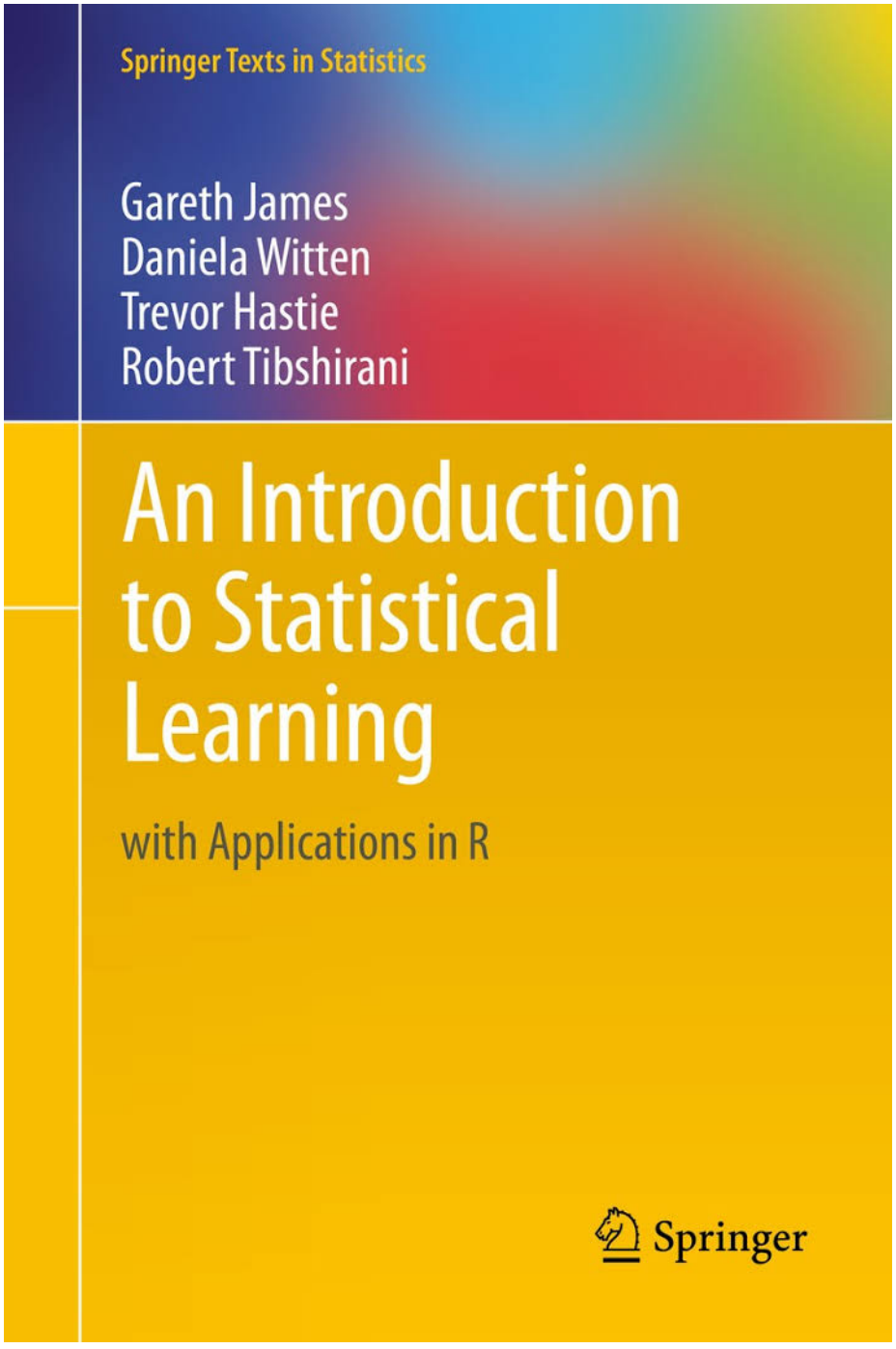
low model complexity
(needs 2 parameters to describe the decision boundary)

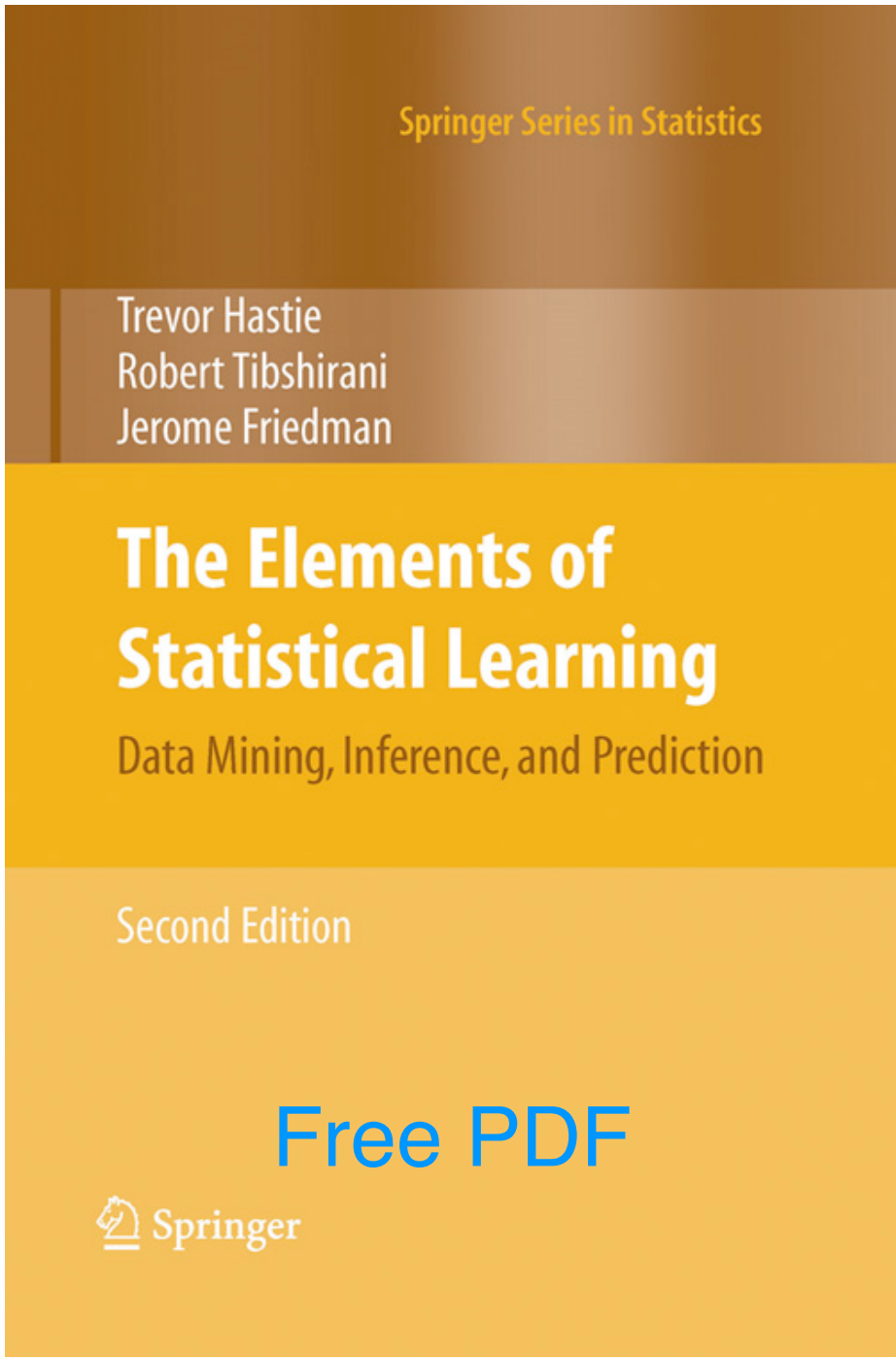
high model complexity
(need 100s of parameters to describe decision boundary)

Reduce complexity: regularization (Lasso, ridge, …)
Increase complexity: data transformation, augmentation, kernels
Always assess classifiers by cross-validation

Another example of overfitting




Springer Texts in Statistics
Gareth James
Daniela Witten
Trevor Hastie
Robert Tibshirani
An Introduction to Statistical Learning
with Applications in R
Springer


Springer Series in Statistics
Trevor Hastie
Robert Tibshirani
Jerome Friedman
The Elements of Statistical Learning
Data Mining, Inference, and Prediction
Second Edition
Free PDF
Springer