



Machine Learning (supervised)

Wolfgang Huber
based on material by Bernd Fischer

What you will learn in this lecture

- Exemplary applications
- Linear discriminant analysis
- How to get non-linear decision boundaries using kernel methods
- Overfitting
- Curse of dimensionality
- Regularization, generalizability and model complexity
- Using cross-validation to
 - tune parameters
 - assess performance

An example dataset: Diabetes

```
data("diabetes", package = "rrcov")  
head(diabetes)
```

	rw	fpg	glucose	insulin	sspg	group
1	0.81	80	356	124	55	normal
2	0.95	97	289	117	76	normal
3	0.94	105	319	143	105	normal
4	1.04	90	356	199	108	normal
5	1.00	90	323	240	143	normal
6	0.76	86	381	157	165	normal

rw: relative weight

fpg: fasting plasma glucose

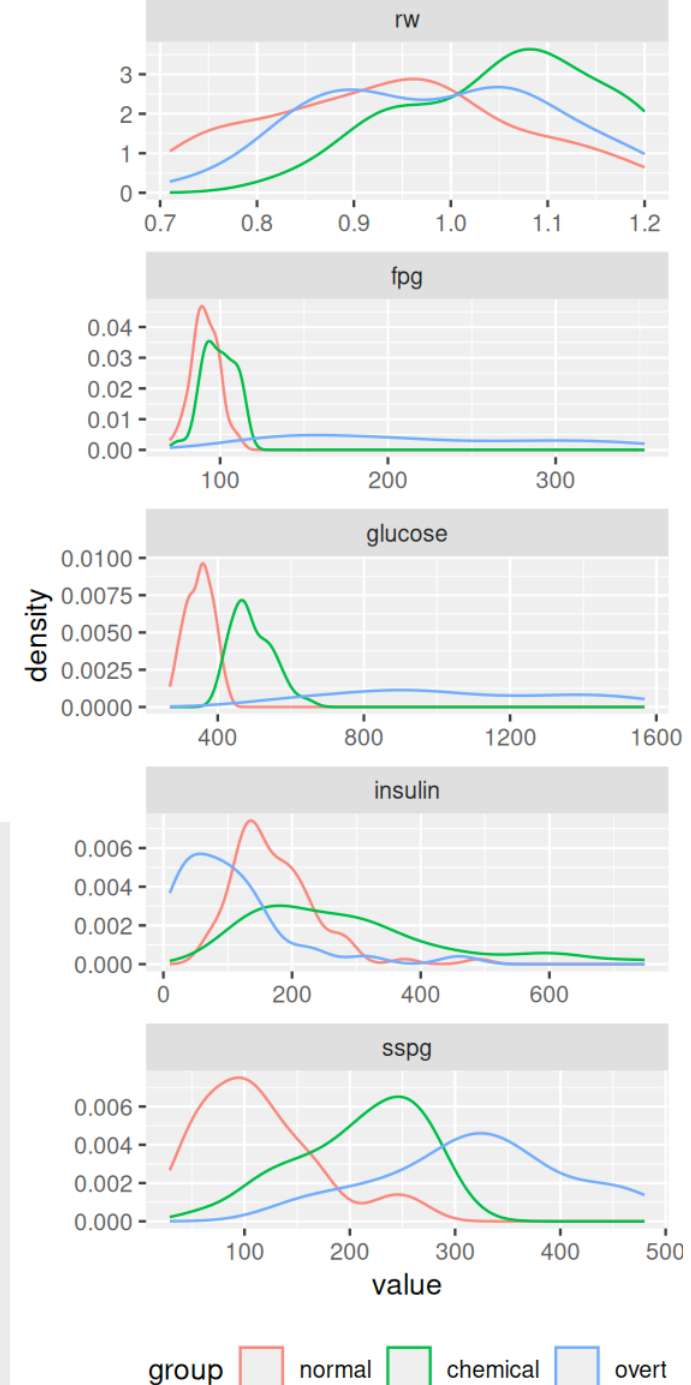
glucose: area under the plasma glucose curve for the three hour oral glucose tolerance test (OGTT)

insulin: area under the plasma insulin curve for the OGTT

sspg: steady state plasma glucose response

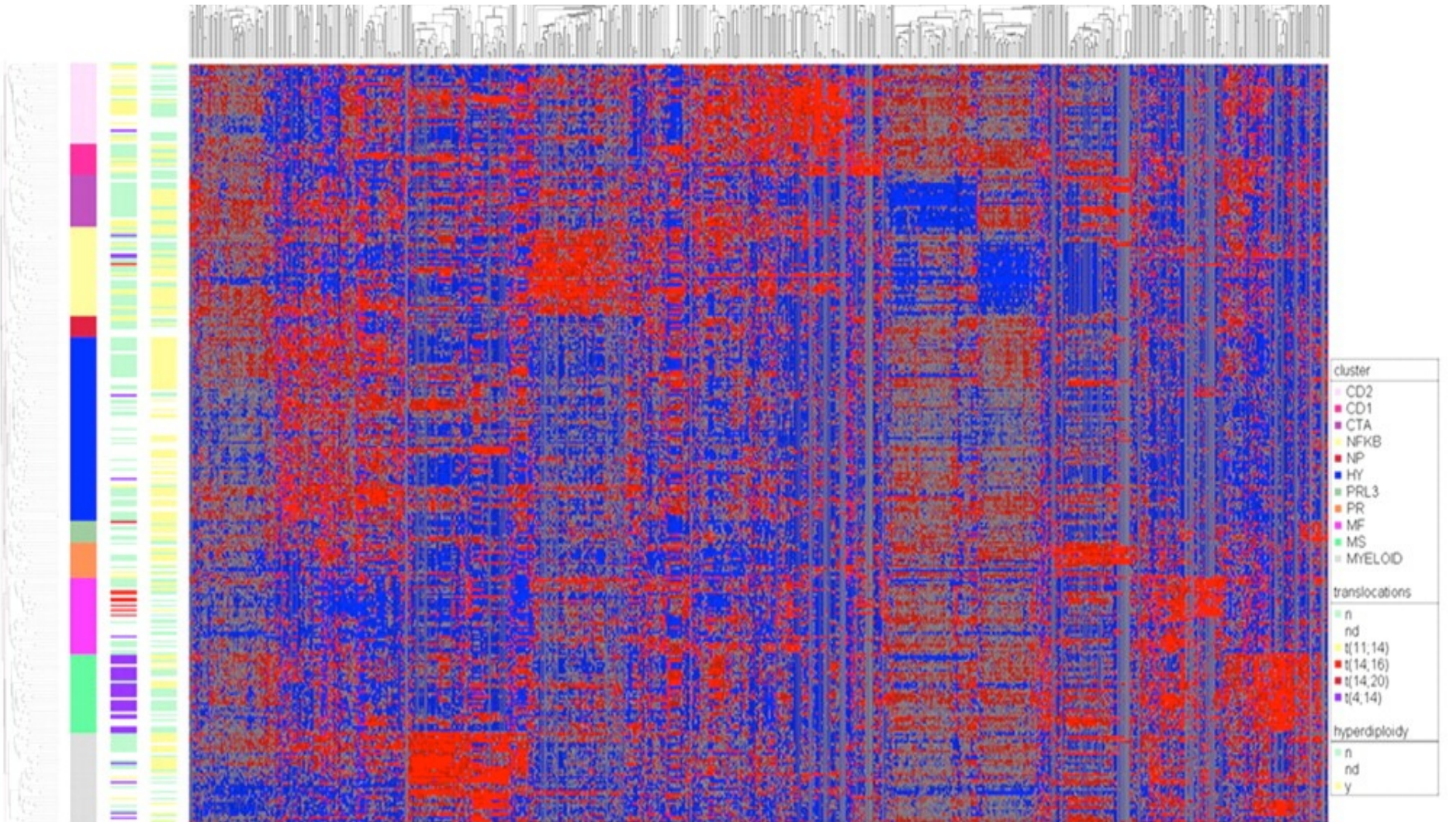
group: normal, chemical diabetes and overt diabetes

Reaven, GM, and RG Miller. 1979. "An Attempt to Define the Nature of Chemical Diabetes Using a Multidimensional Analysis." Diabetologia 16: 17.



Molecular classification of cancer

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



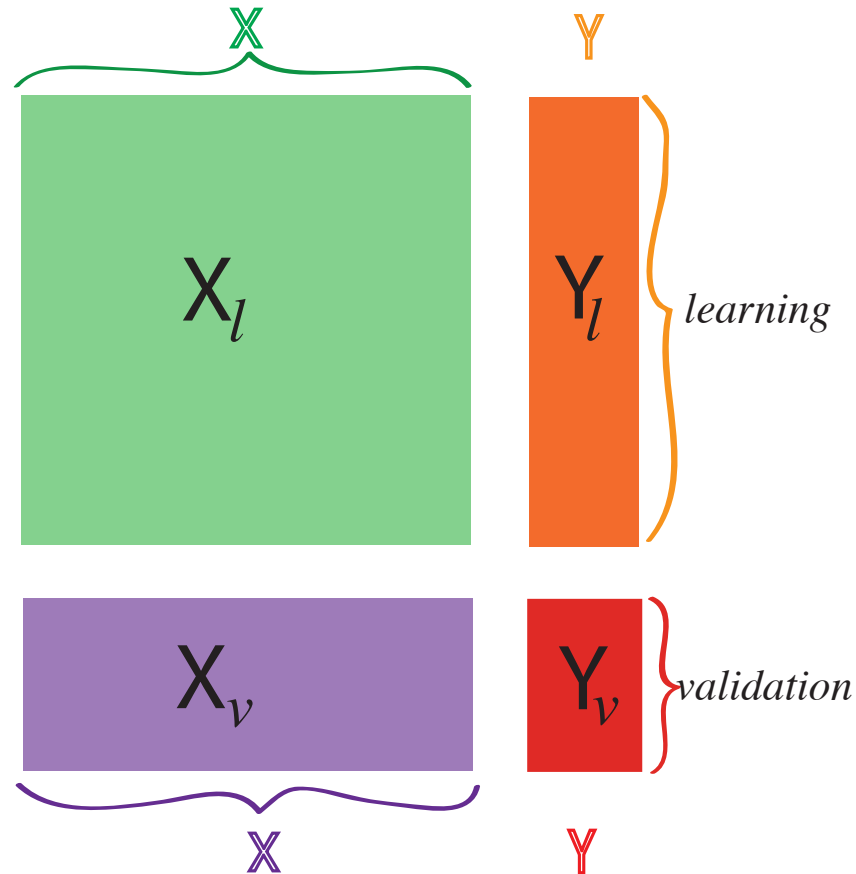


Figure 13.3: In supervised learning, we assign two different roles to our variables. We have labeled the explanatory variables X and the response variable(s) Y . There are also two different sets of observations: the training set X_ℓ and Y_ℓ and the validation set X_v and Y_v .

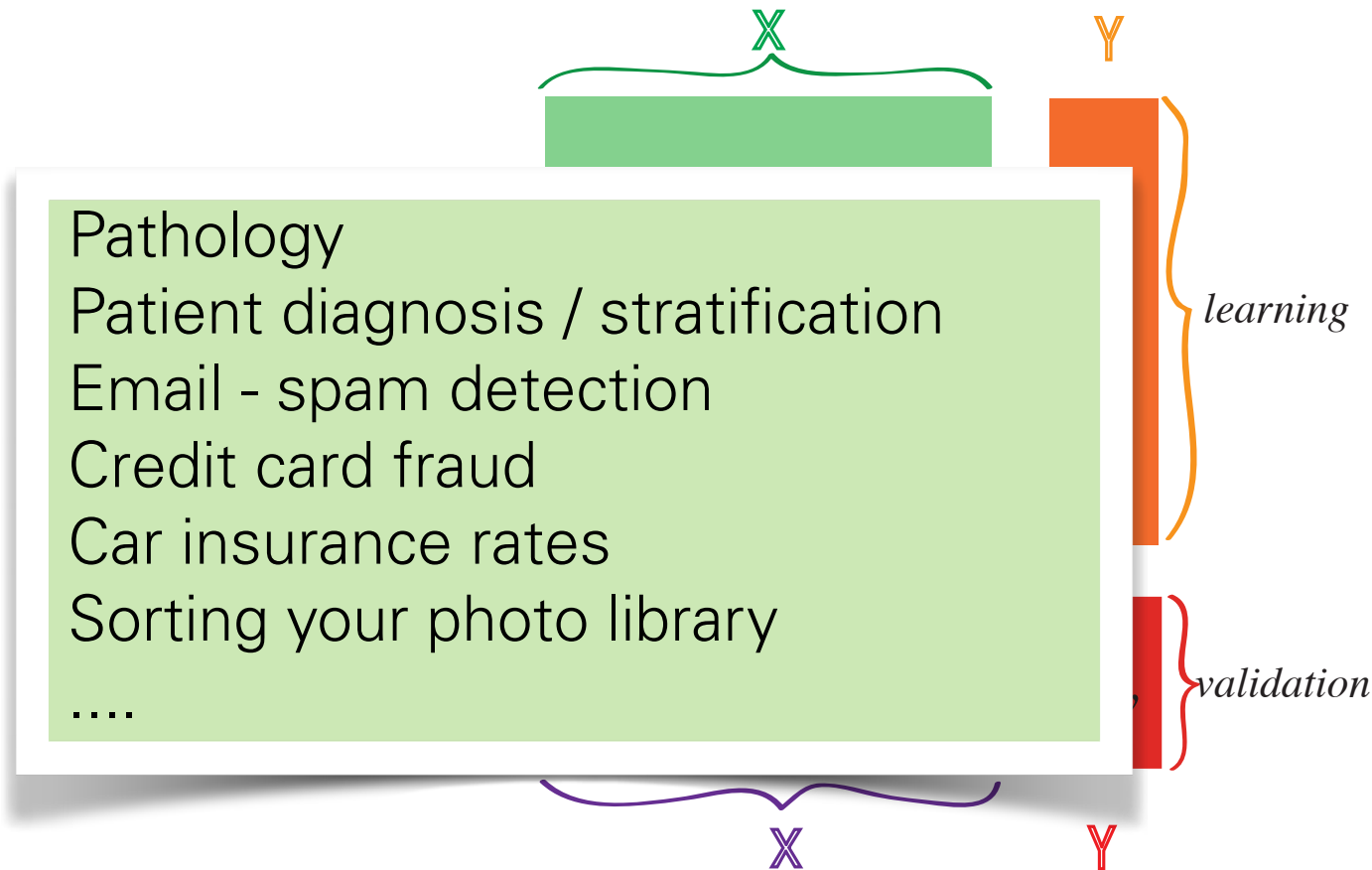
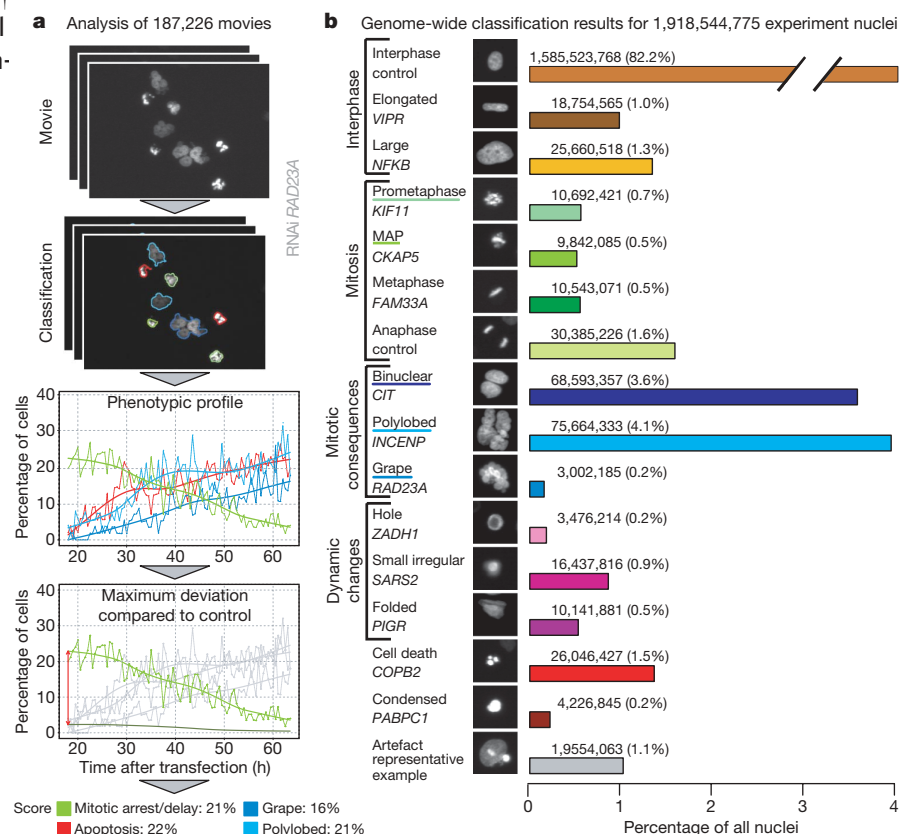


Figure 13.3: In supervised learning, we assign two different roles to our variables. We have labeled the explanatory variables X and the response variable(s) Y . There are also two different sets of observations: the training set X_ℓ and Y_ℓ and the validation set X_v and Y_v .

ARTICLES

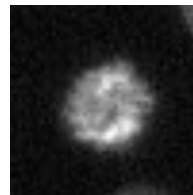
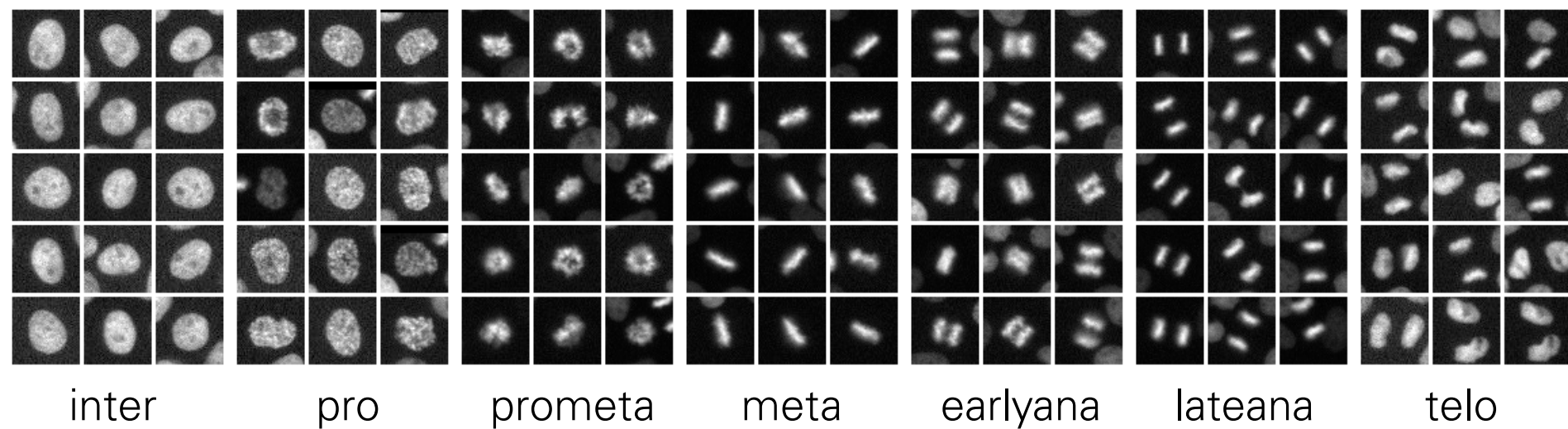
Phenotypic profiling of the human genome by time-lapse microscopy reveals cell division genes

Beate Neumann^{1*}, Thomas Walter^{1*}, Jean-Karim Hériché^{5†}, Jutta Bulkescher¹, Holger Erfle^{1,3†}, Christian Conrad^{1,3}, Phill Rogers^{1†}, Ina Poser⁶, Michael Held^{1†}, Urban Liebel^{1†}, Gregoire Pau⁹, Rolf Kabbe¹⁰, Annelie Wünsche², Venkata Satagopam⁴, Michael Daniel W. Gerlich⁷, Reinhard Schneider⁴, Roland Eils¹⁰, Wolfgang Huber⁹, Jan-Anthony A. Hyman⁶, Richard Durbin⁵, Rainer Pepperkok³ & Jan Ellenberg²



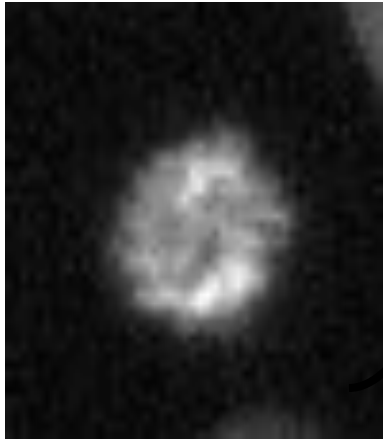
Morphological Phenotyping

Provide Human Annotation to a small set of cells:



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

Automatic Classification Workflow



Preprocessing

e.g. normalization, background subtraction, ...

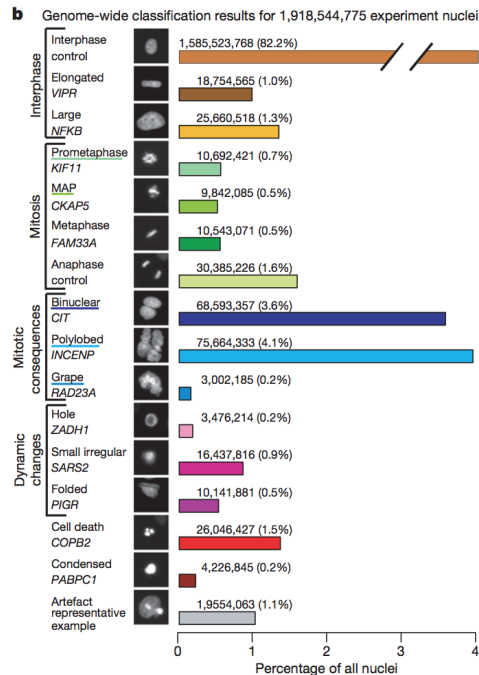
Feature extraction

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

Classification

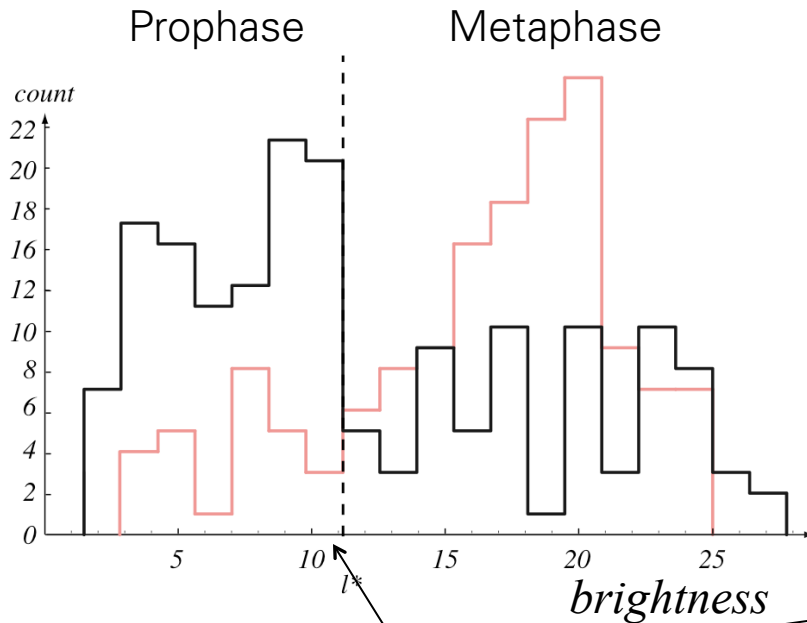
Prophase

Metaphase

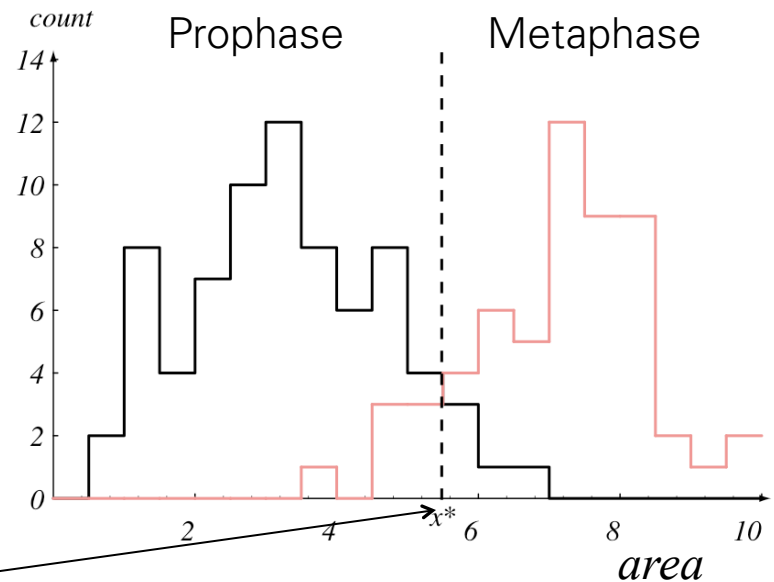


Prophase/ Metaphase Classification

Predict mitotic state based on brightness



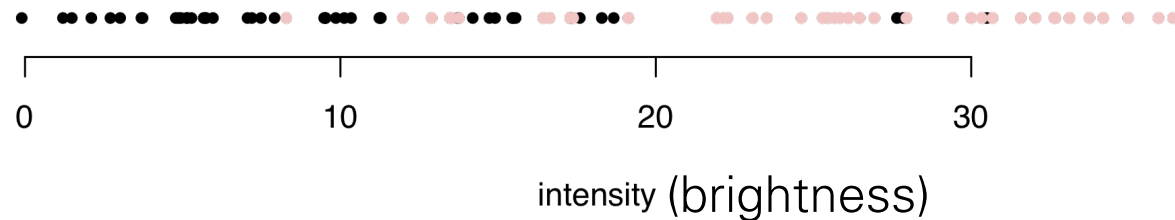
Predict mitotic state based on nucleus area



Decision boundary with lowest prediction error

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

A Simple Least Squares Classifier (1D)

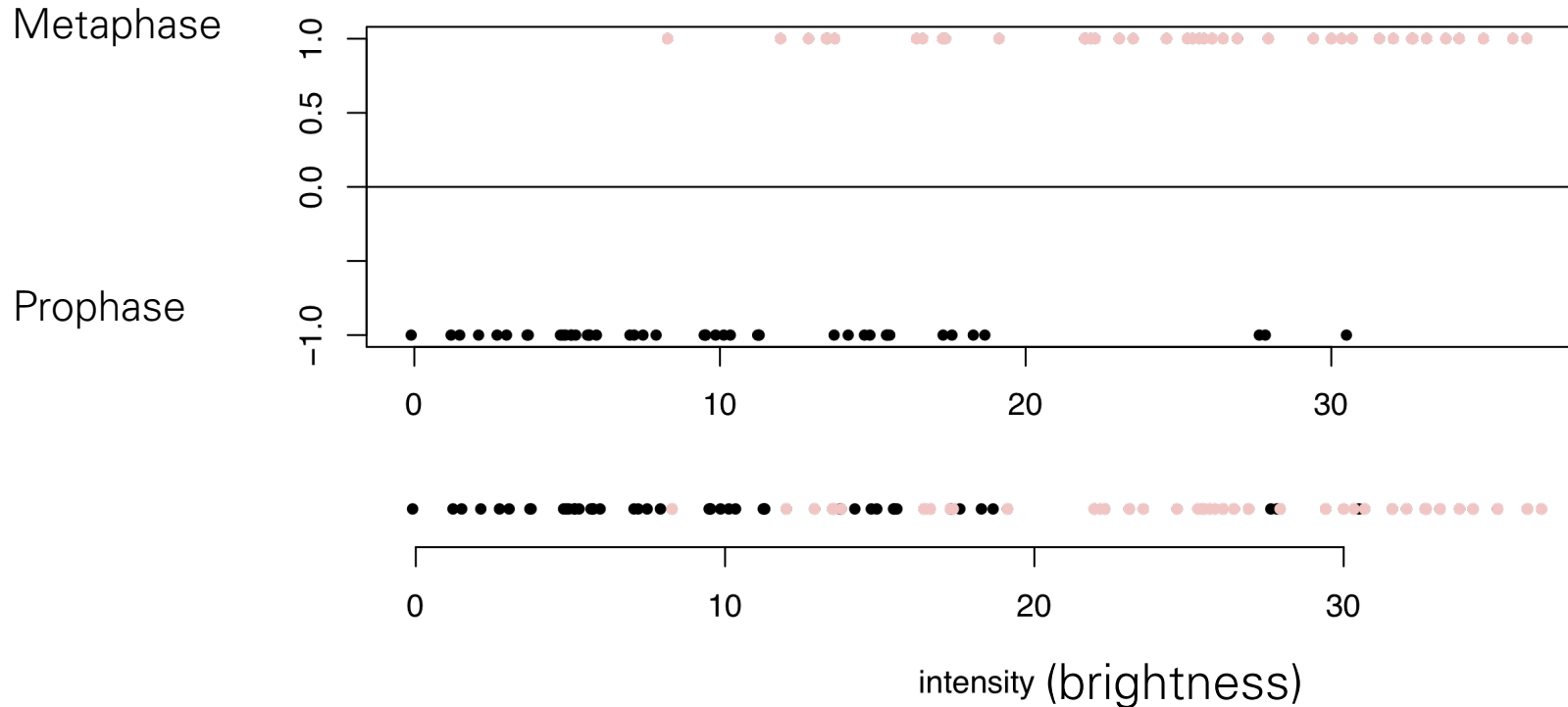


```
y[i] = -1 for prophase  
y[i] = +1 for metaphase  
x[i,] = c(area[i], intensity[i])  
model = lm(y ~ x)  
ynew = predict(model, newdata=newX)  
ifelse(ynew < 0, "meta", "pro")
```

Find β for which

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

A Simple Least Squares Classifier (1D)

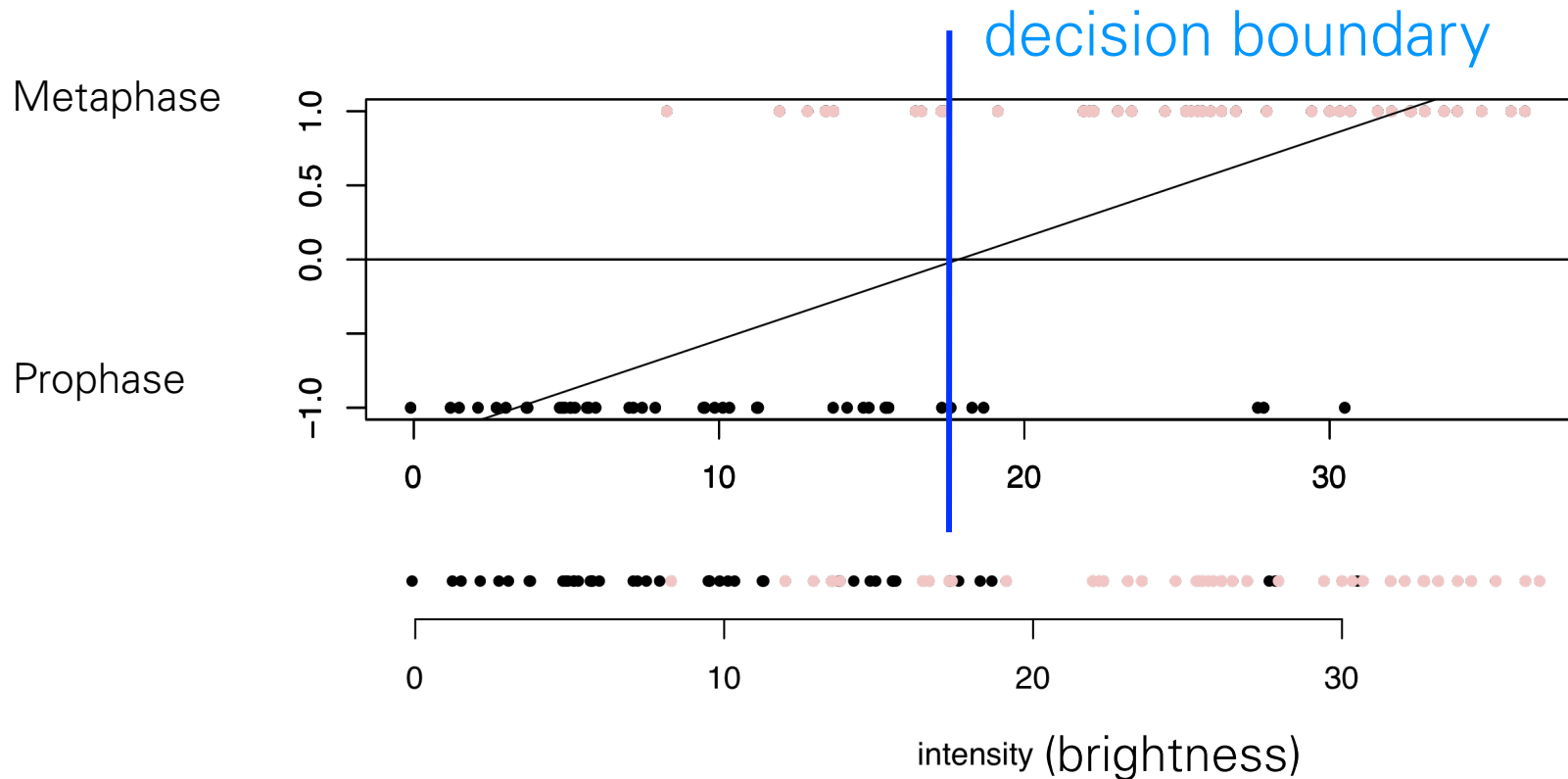


```
y[i] = -1 for prophase  
y[i] = +1 for metaphase  
x[i,] = c(area[i], intensity[i])  
model = lm(y ~ x)  
ynew = predict(model, newdata=newX)  
ifelse(ynew < 0, "meta", "pro")
```

Find β for which

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

A Simple Least Squares Classifier (1D)

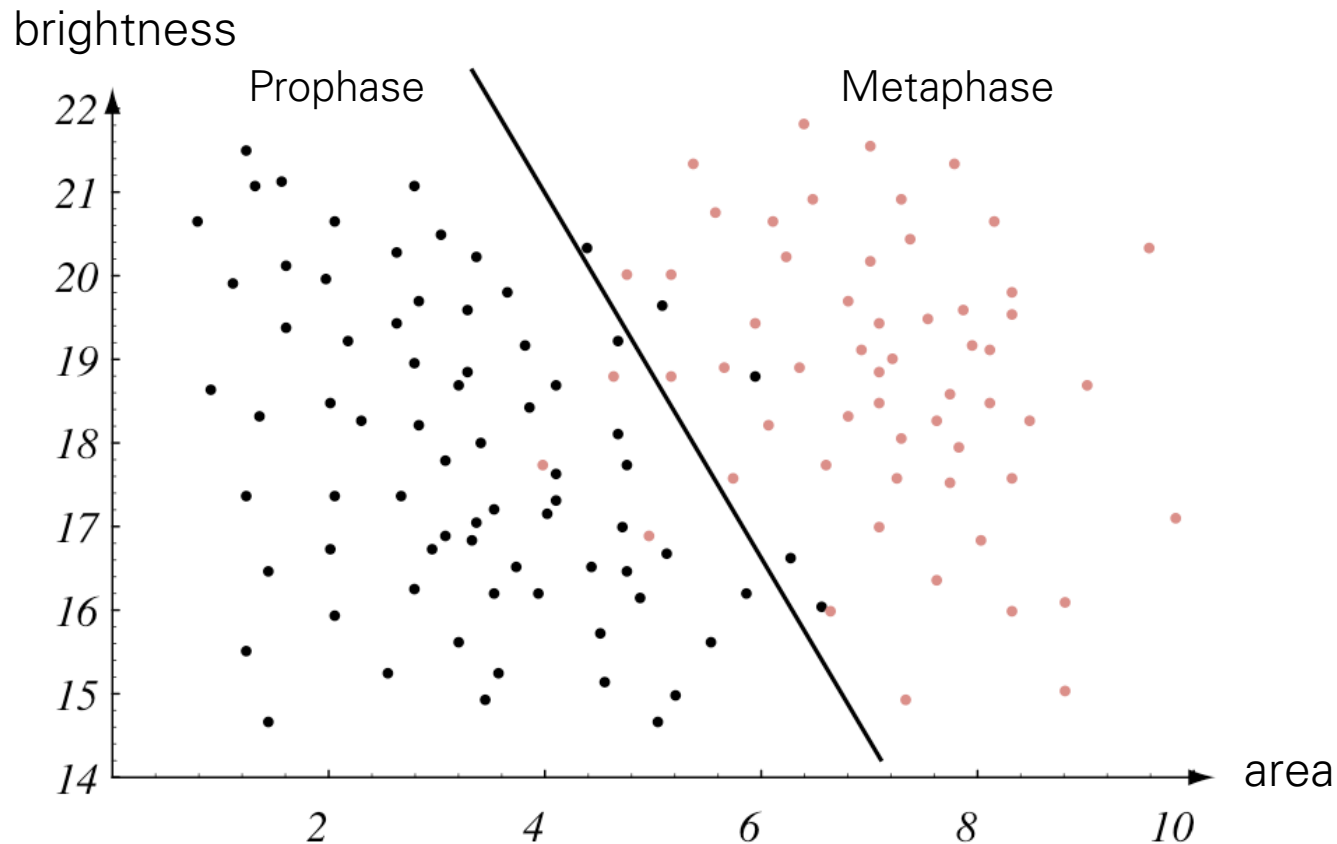


```
y[i] = -1 for prophase  
y[i] = +1 for metaphase  
x[i,] = c(area[i], intensity[i])  
model = lm(y ~ x)  
ynew = predict(model, newdata=newX)  
ifelse(ynew < 0, "meta", "pro")
```

Find β for which

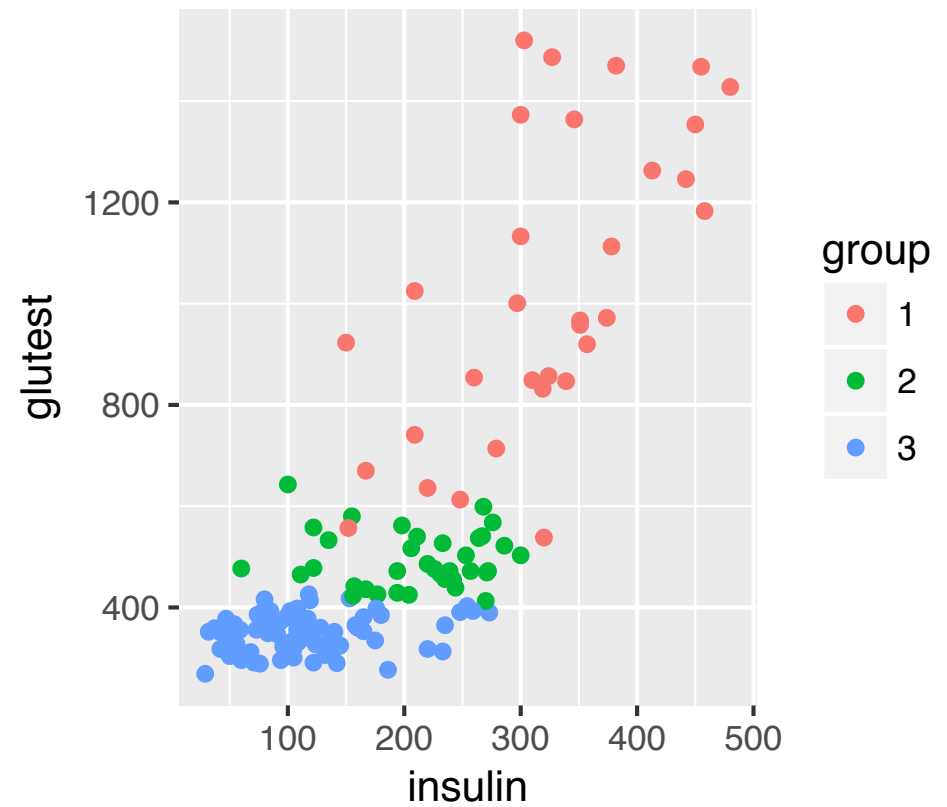
$$\sum_i (y_i - \beta x_i)^2 \rightarrow \min$$

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.fit(x, y)  
ynew = predict(model, Xnew)  
          $fitted.values  
ifelse(ynew < 0, "meta", "pro")
```

Linear discriminant Analysis (LDA) on the diabetes data



Linear discriminant Analysis (LDA) on the diabetes data

```
library("MASS")
diabetes_lda = lda(group ~ insulin + glutest, data = diabetes)
diabetes_lda

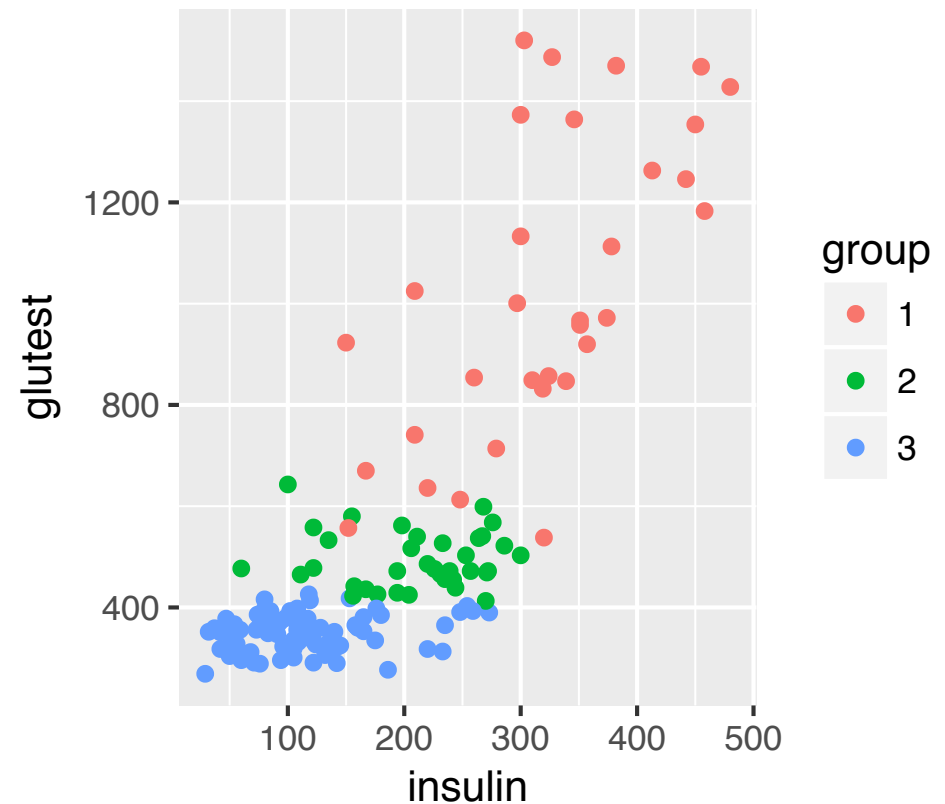
## Call:
## lda(group ~ insulin + glutest, data = diabetes)
##
## Prior probabilities of groups:
##      1      2      3
## 0.2222222 0.2500000 0.5277778
##
## Group means:
##      insulin  glutest
## 1 320.9375 1027.3750
## 2 208.9722  493.9444
## 3 114.0000  349.9737
##
## Coefficients of linear discriminants:
##              LD1          LD2
## insulin -0.004463900 -0.01591192
## glutest -0.005784238  0.00480830
##
## Proportion of trace:
##      LD1      LD2
## 0.9677 0.0323

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
```



Linear discriminant Analysis (LDA) on the diabetes data

```
library("MASS")
diabetes_lda = lda(group ~ insulin + glutest, data = diabetes)
diabetes_lda

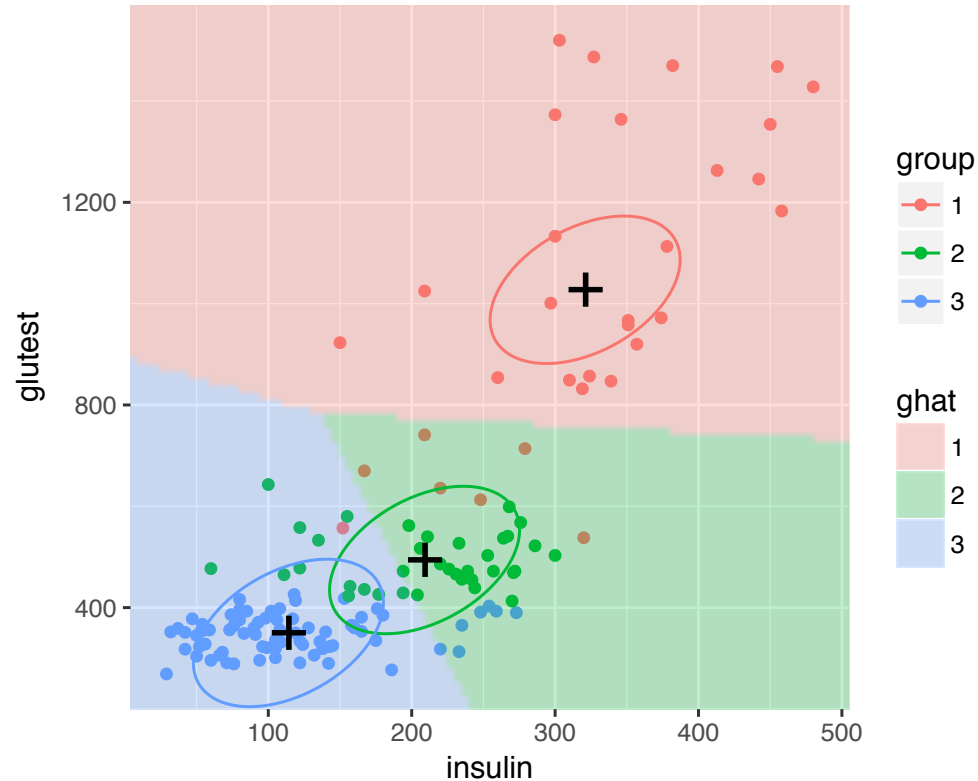
## Call:
## lda(group ~ insulin + glutest, data = diabetes)
##
## Prior probabilities of groups:
##      1      2      3
## 0.2222222 0.2500000 0.5277778
##
## Group means:
##      insulin  glutest
## 1 320.9375 1027.3750
## 2 208.9722  493.9444
## 3 114.0000  349.9737
##
## Coefficients of linear discriminants:
##              LD1      LD2
## insulin -0.004463900 -0.01591192
## glutest -0.005784238  0.00480830
##
## Proportion of trace:
##      LD1      LD2
## 0.9677 0.0323

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: $\Sigma_g = \Sigma$

Linear discriminant Analysis (LDA) on the diabetes data

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

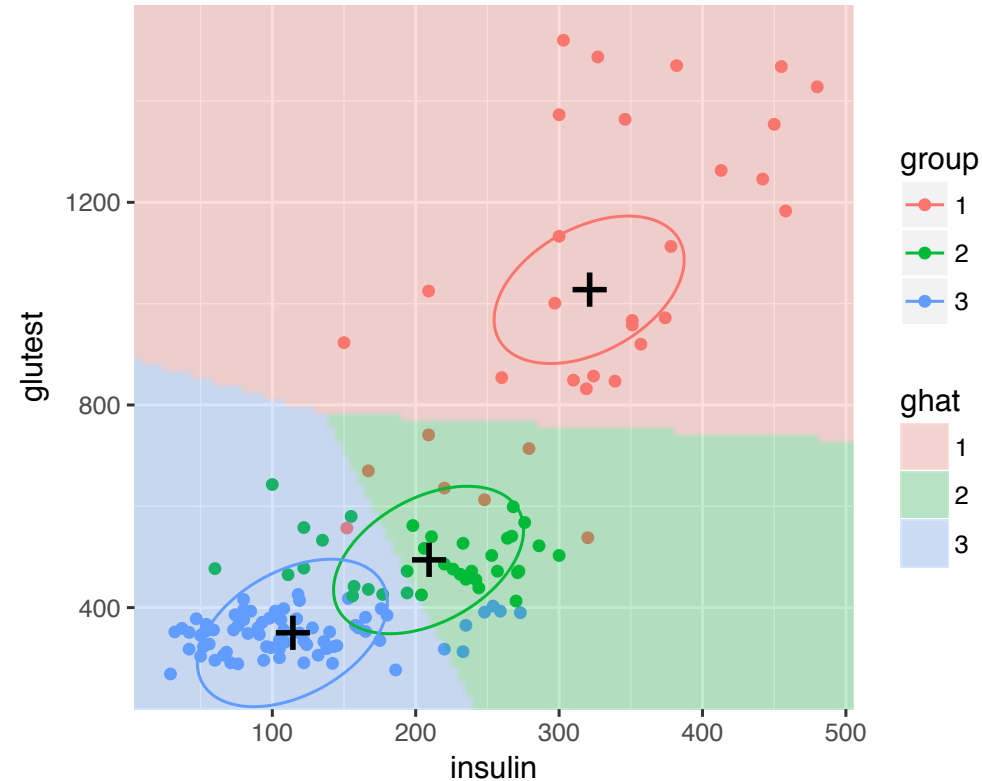
Why is the boundary between the prediction regions for groups 1 and 2 not perpendicular to the line between the cluster centers?

```
## 0.9877 0.0323
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 $N(\mu_g, \Sigma_g)$

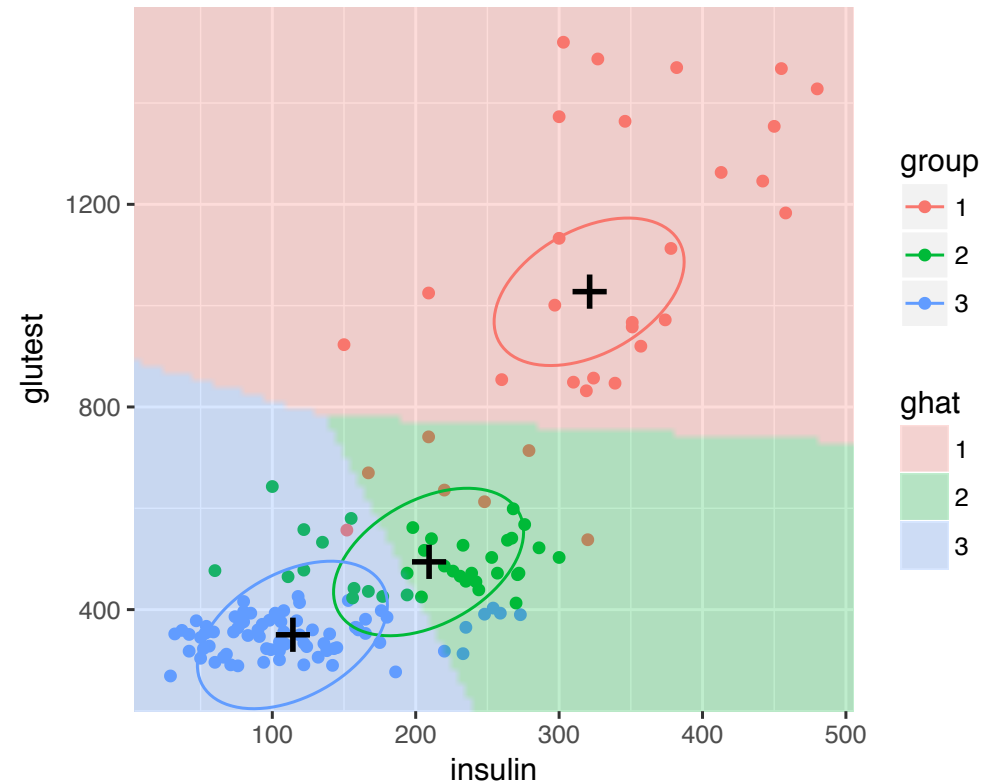
LDA: $\Sigma_g = \Sigma$

Linear discriminant Analysis (LDA) on the diabetes data

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

How confident would you be about the predictions in those areas of the 2D plane that are far from all of the cluster centers?

```
## 0.9677 0.0323  
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 $N(\mu_g, \Sigma_g)$

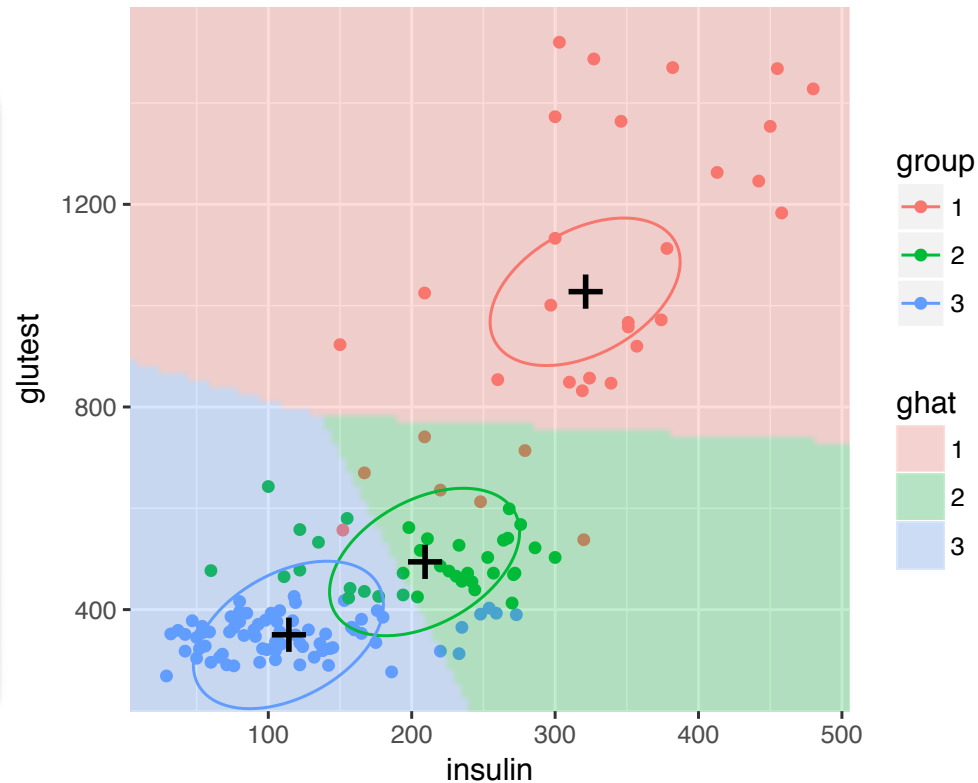
LDA: $\Sigma_g = \Sigma$

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?

```
## 0.9677 0.0323
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 $N(\mu_g, \Sigma_g)$

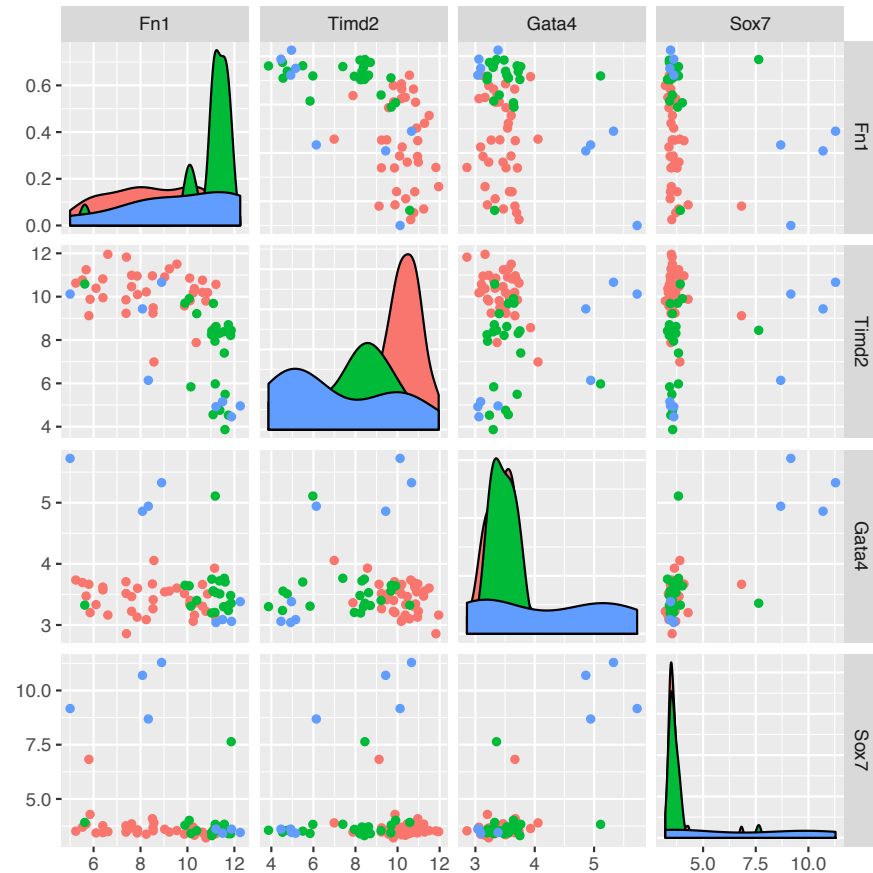
LDA: $\Sigma_g = \Sigma$

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



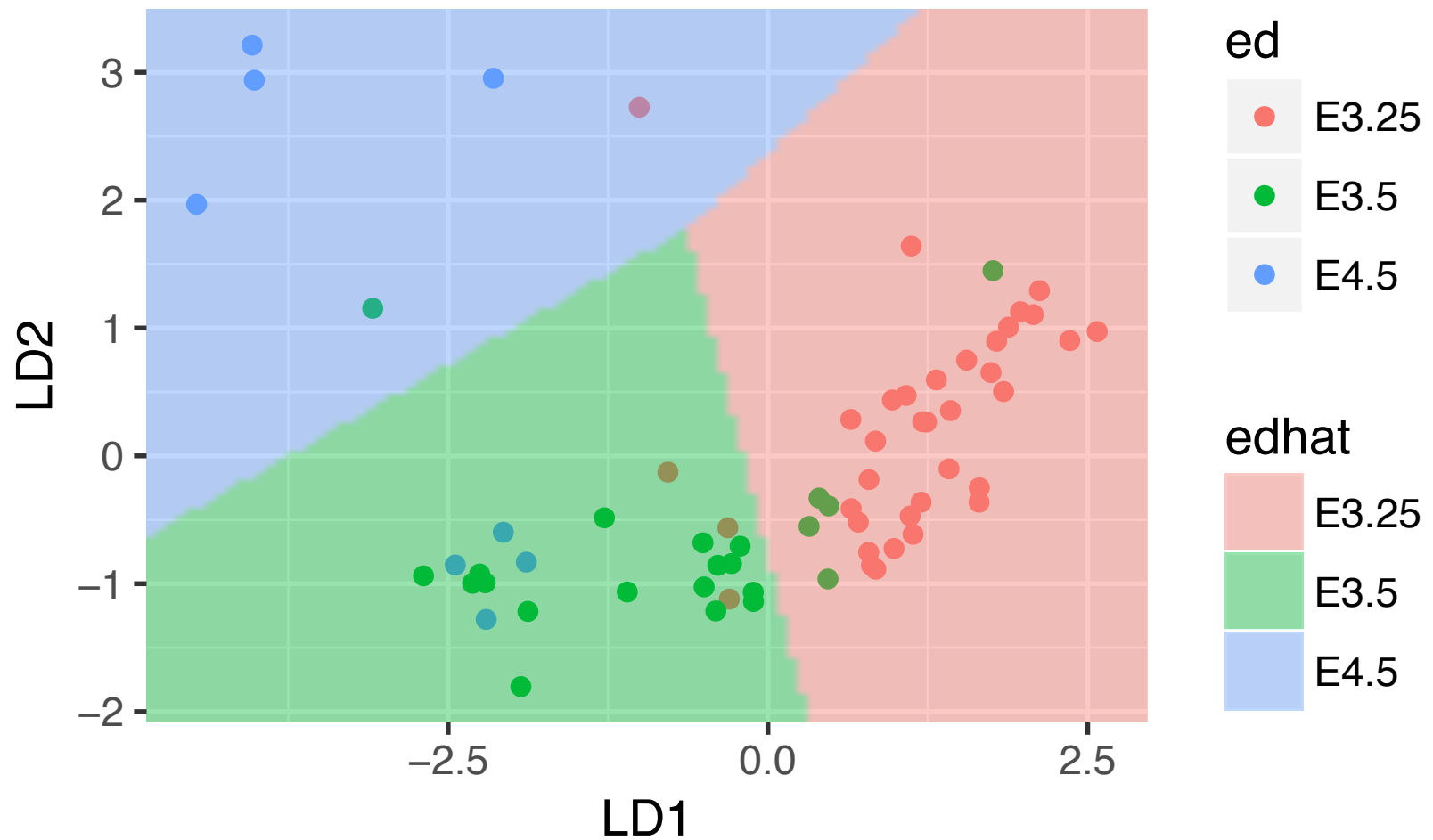


Figure 13.10: LDA classification regions for Embryonic.day.

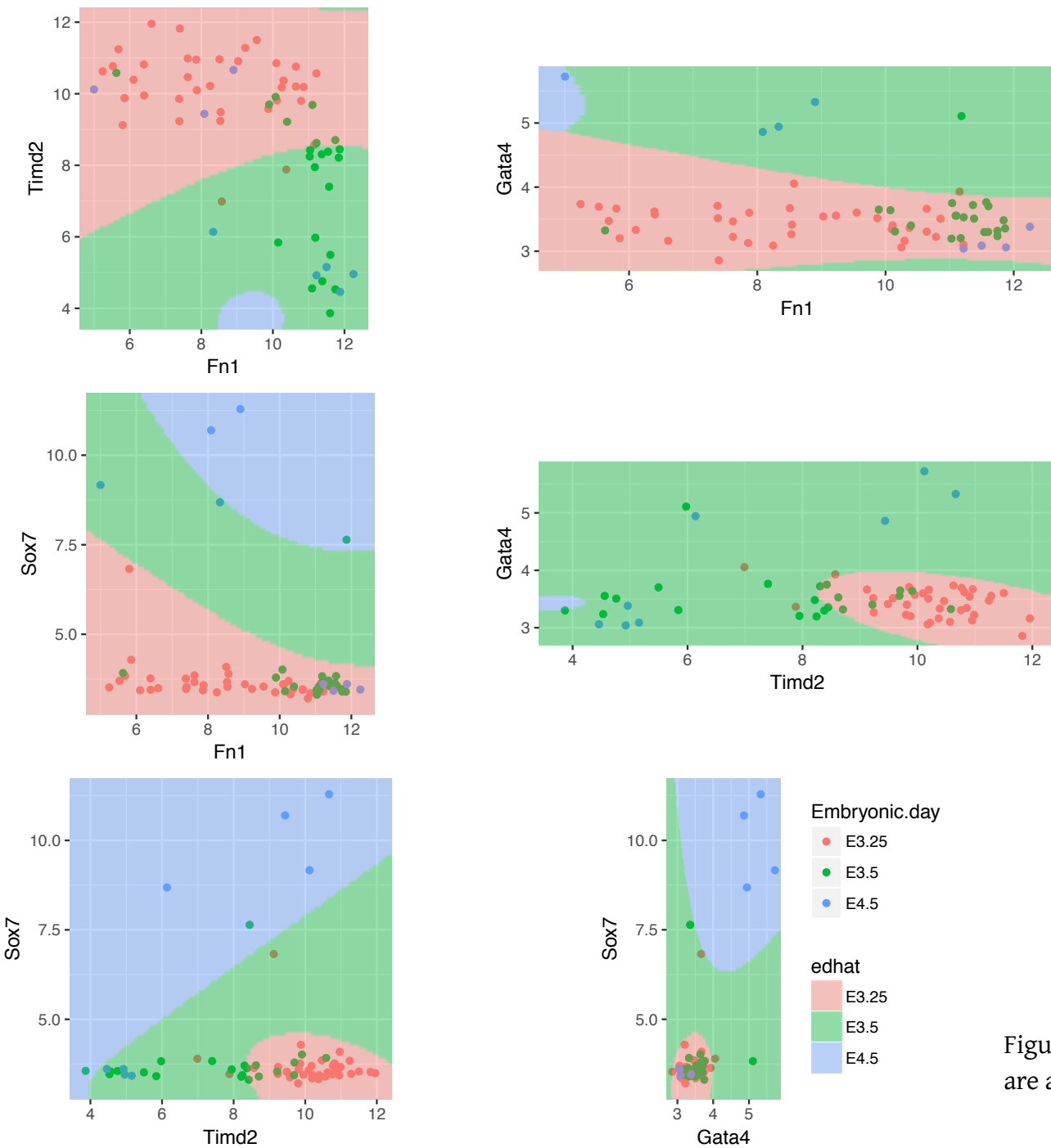
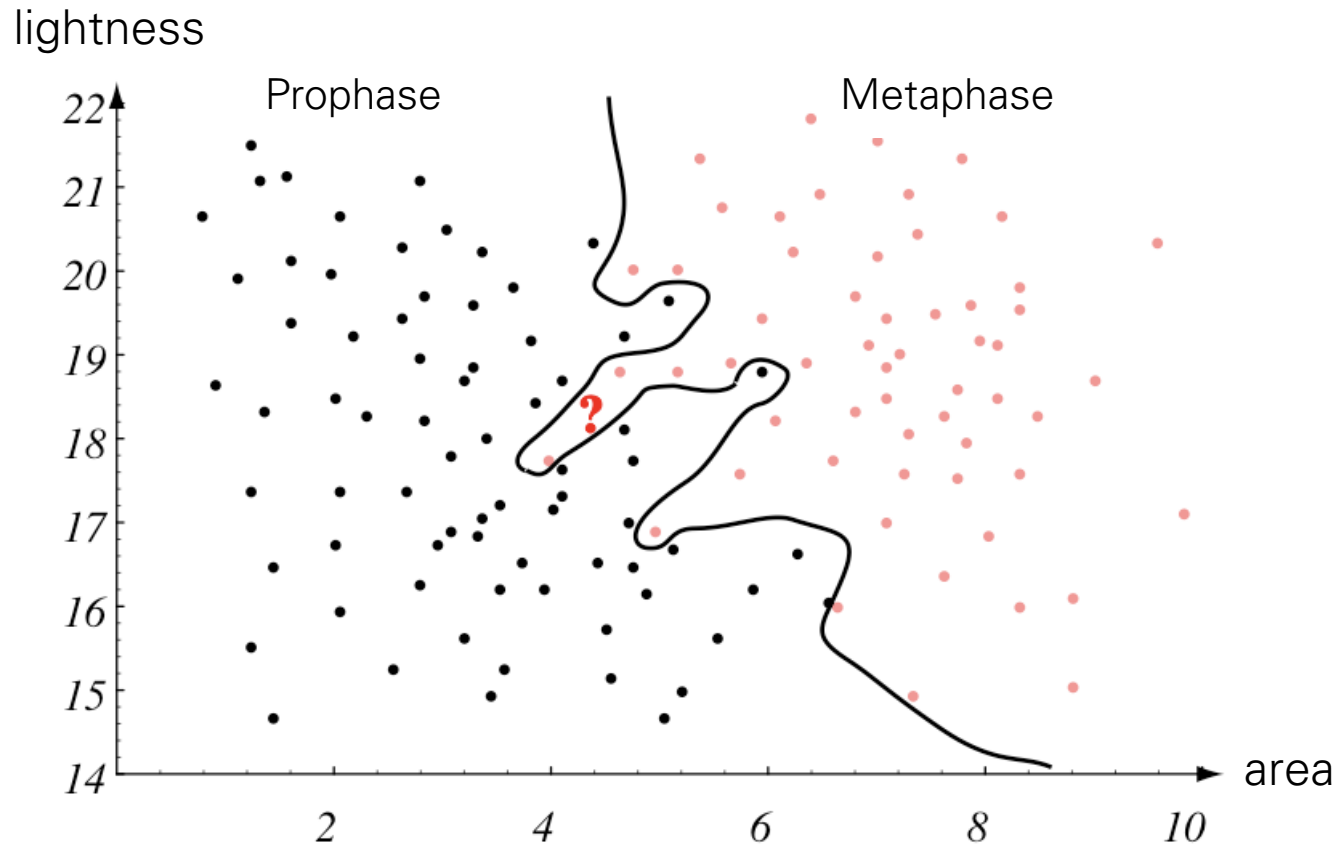


Figure 13.11: QDA for the mouse cell data. Shown are all pairwise plots of the four features.

k-Nearest-Neighbor Classifier



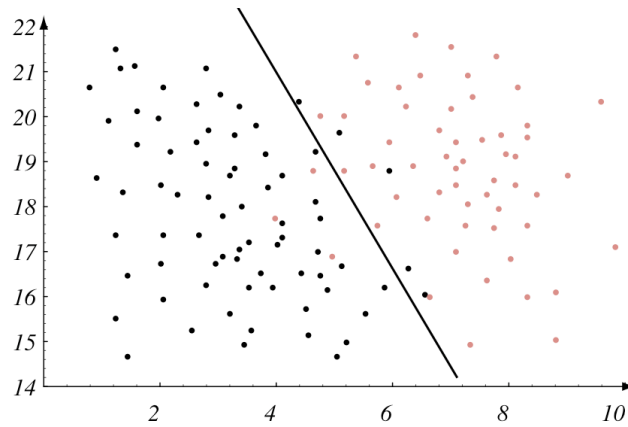
Assign each new cell to the majority class of its k nearest neighbours. Black line shows the resulting decision boundary

```
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

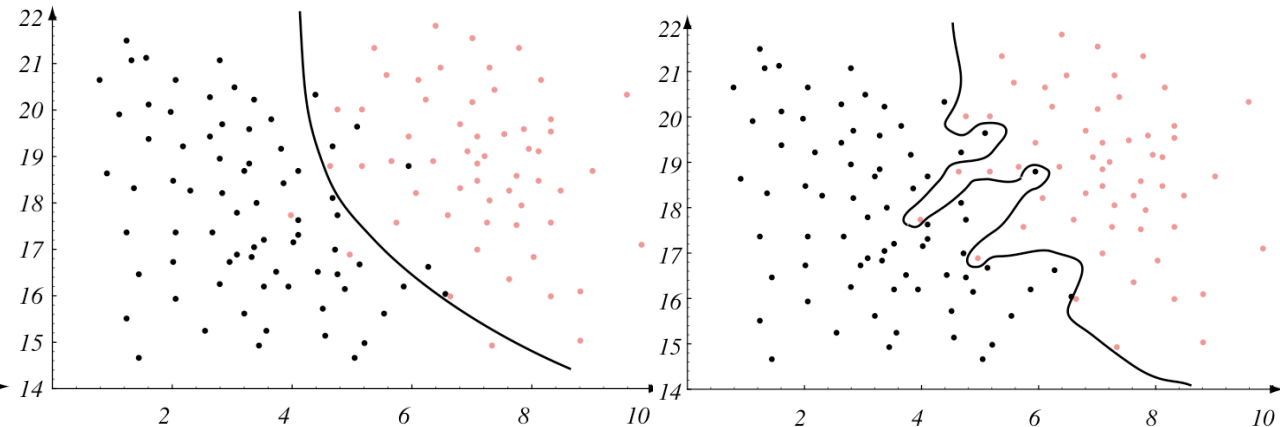
←-----



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

Low bias
High variance

-----→

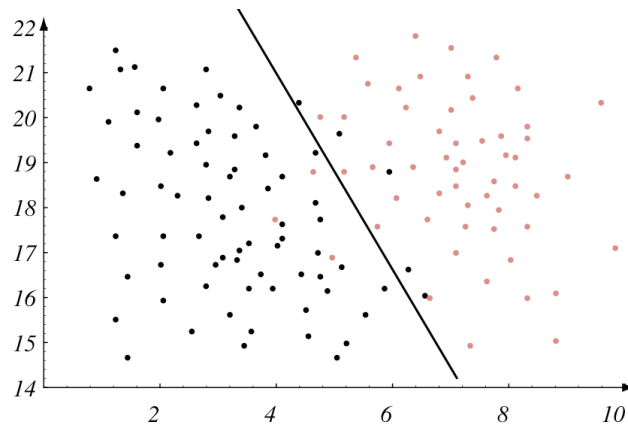


high model complexity
(need 100s of parameters to
describe decision boundary)

Which Decision Boundary?

High bias
Low variance

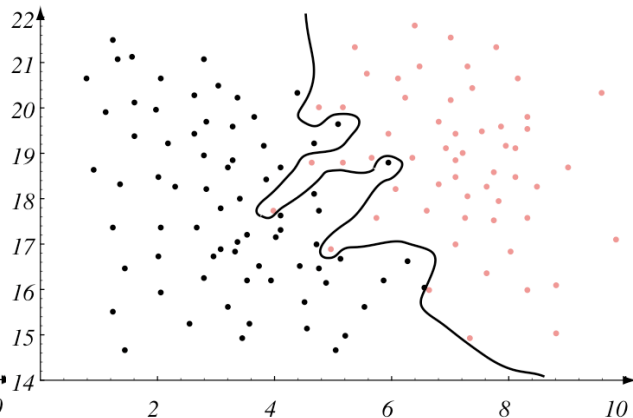
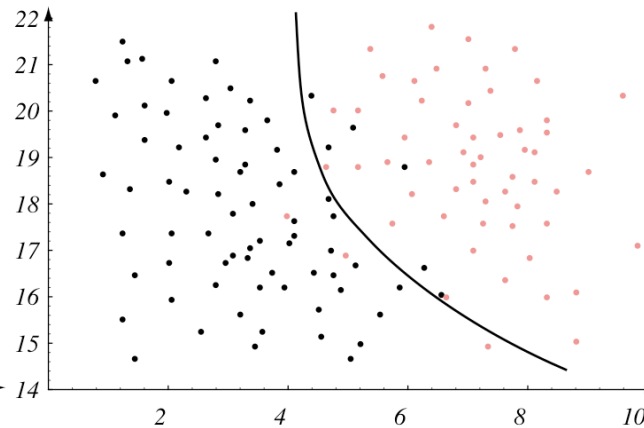
←-----



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

Low bias
High variance

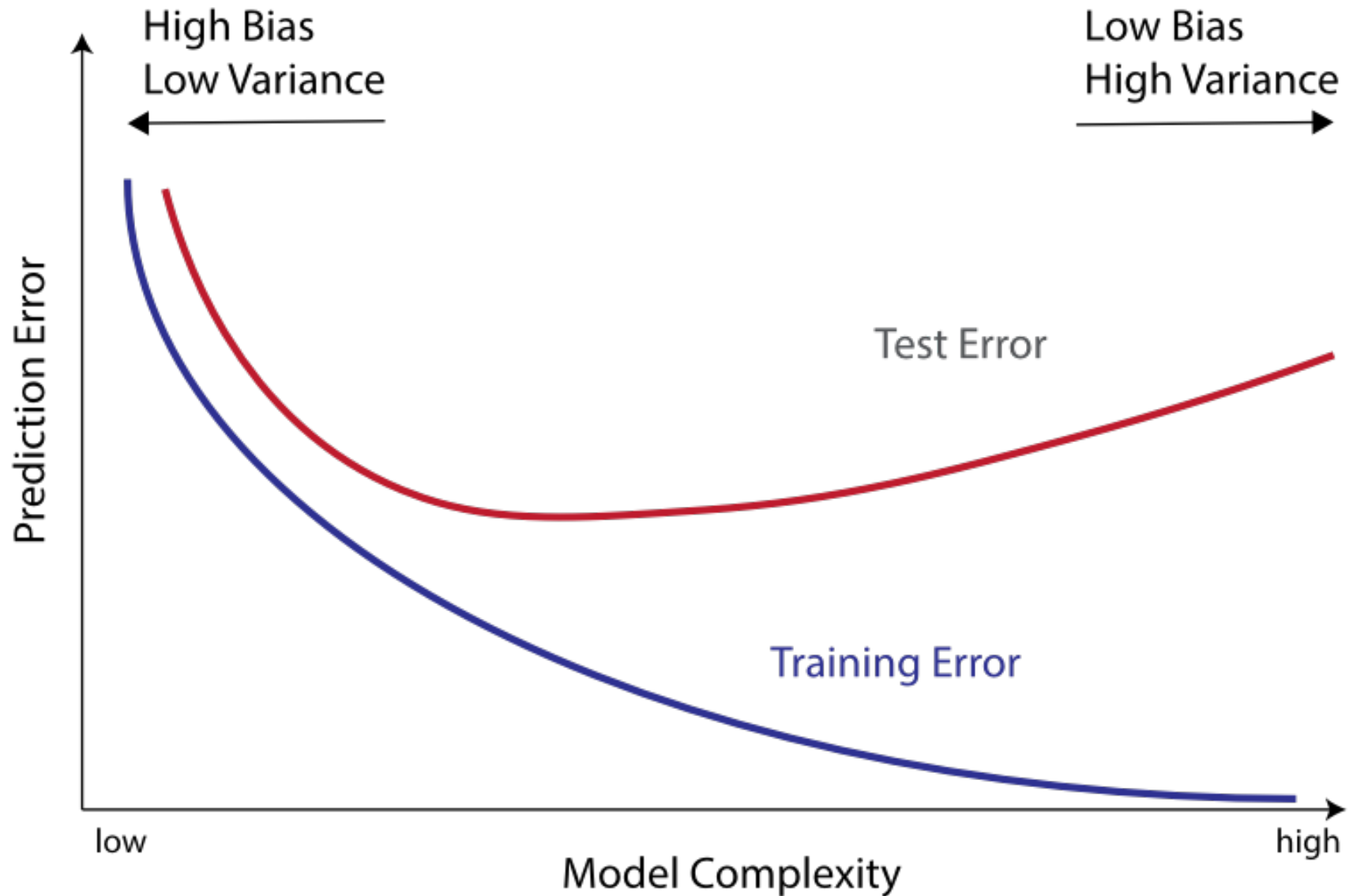
-----→



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

Which decision boundary has the
lowest prediction error?

Bias-Variance-Dilemma



Why are deep neural networks not overfitting?

Implicit Regularization by stochastic gradient descent

- does not explore all possible parameter configurations
- tends to find “flat minima”

Early Stopping

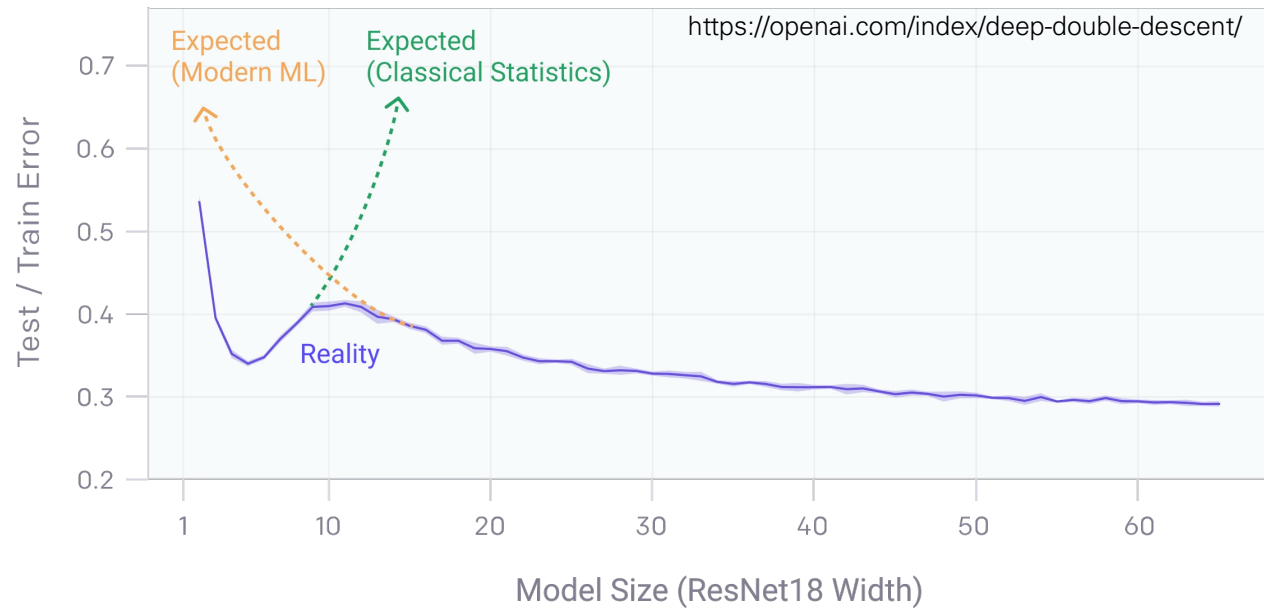
- training is halted before overfitting sets in

Architectural Inductive Biases

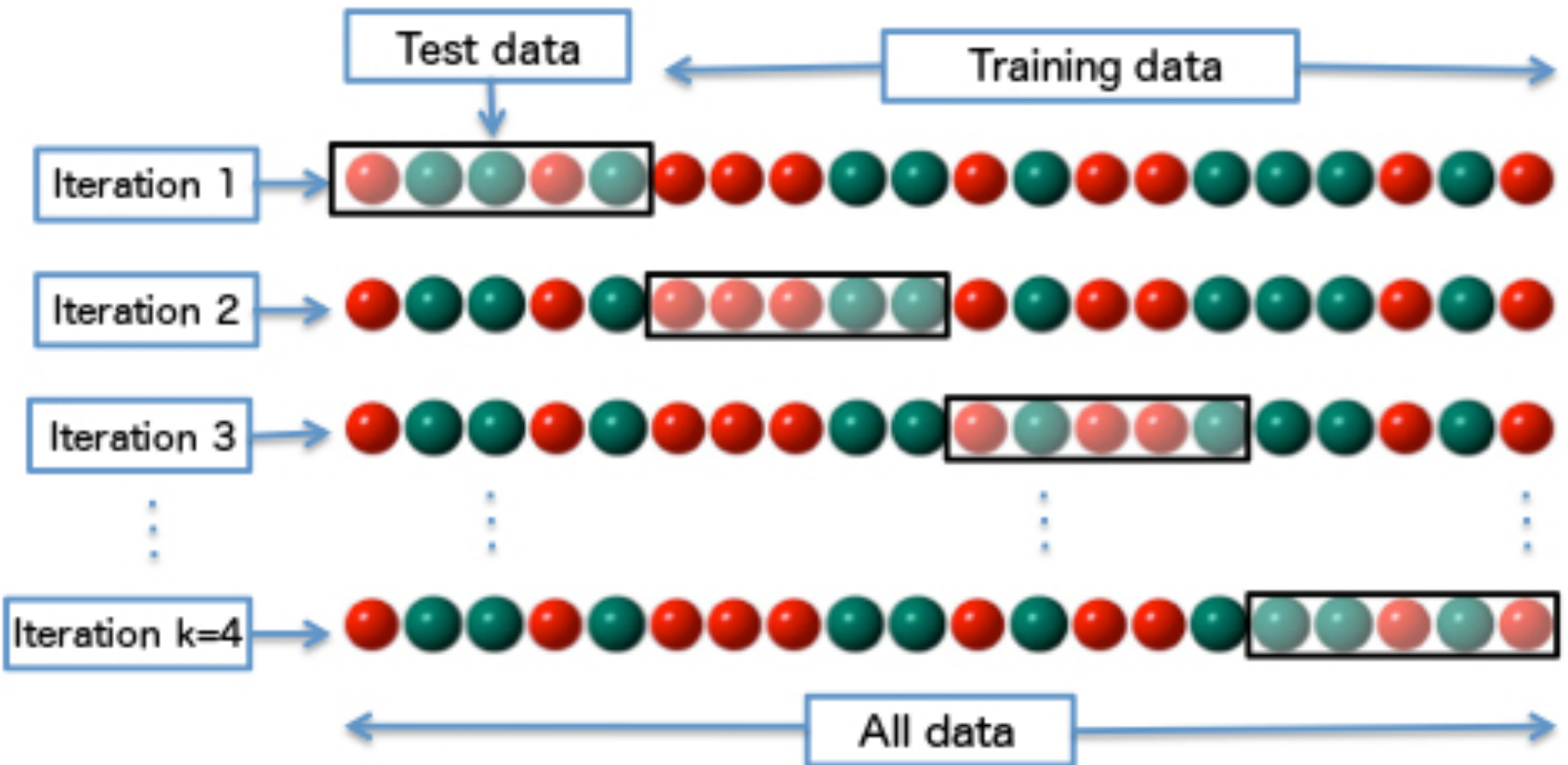
- convolutional neural networks: spatial locality, translation invariance
- transformers encode sequence position and attention
- such structures restrict space of functions the network can learn

Regularization Techniques

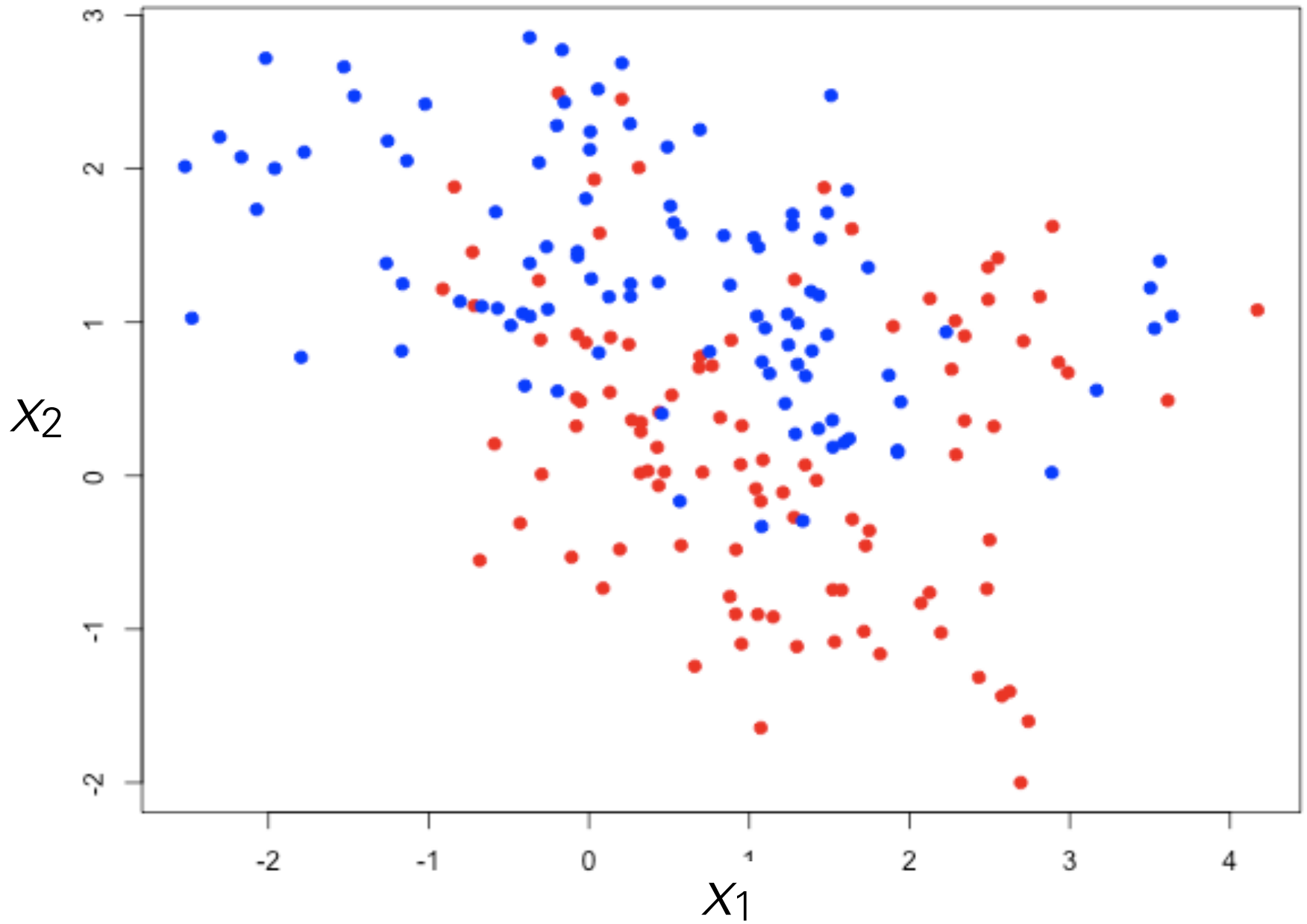
- Dropout, Batch norm, weight decay



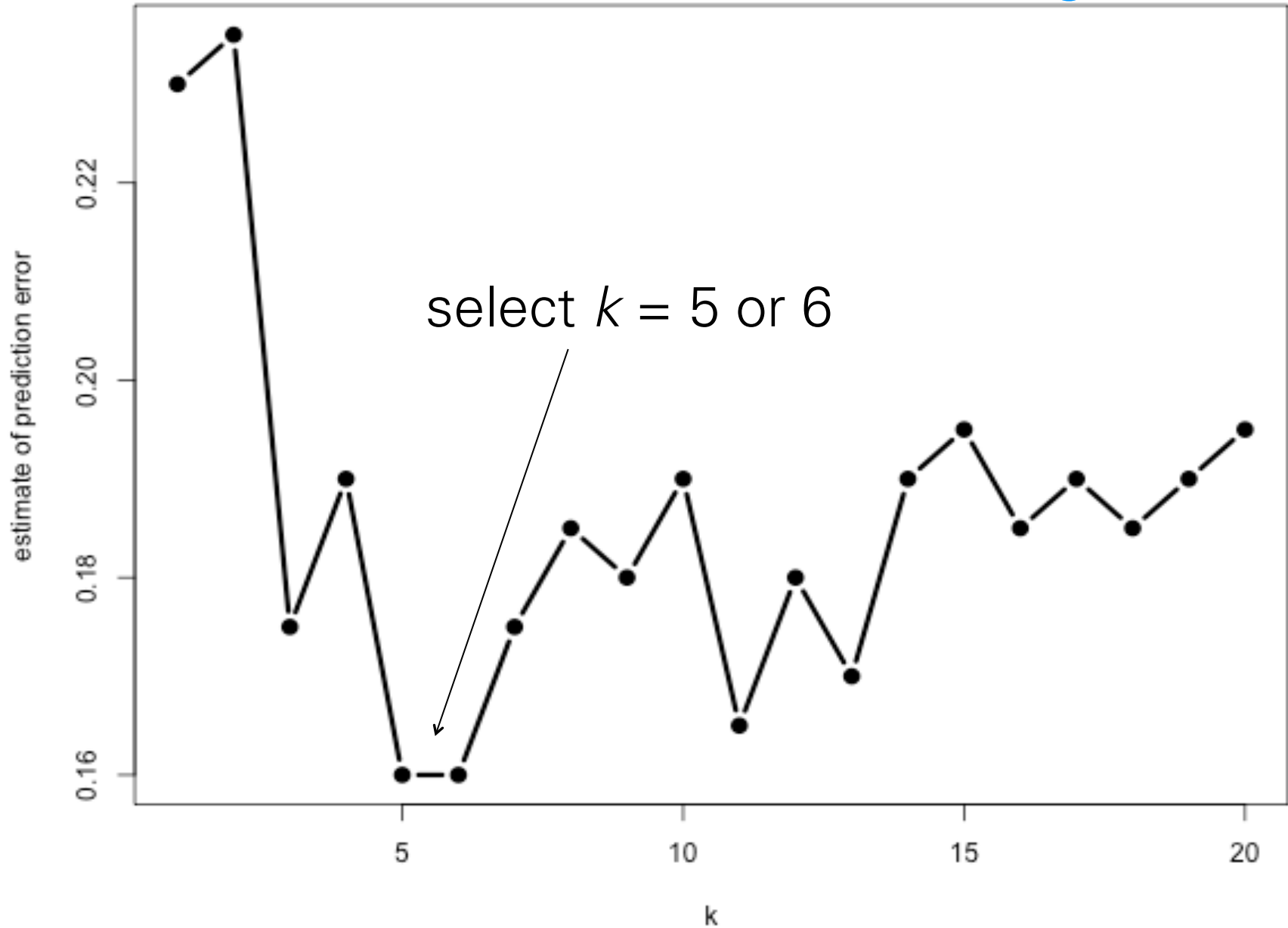
Cross-Validation



Example: 2 classes, 2 variables, 200 objects

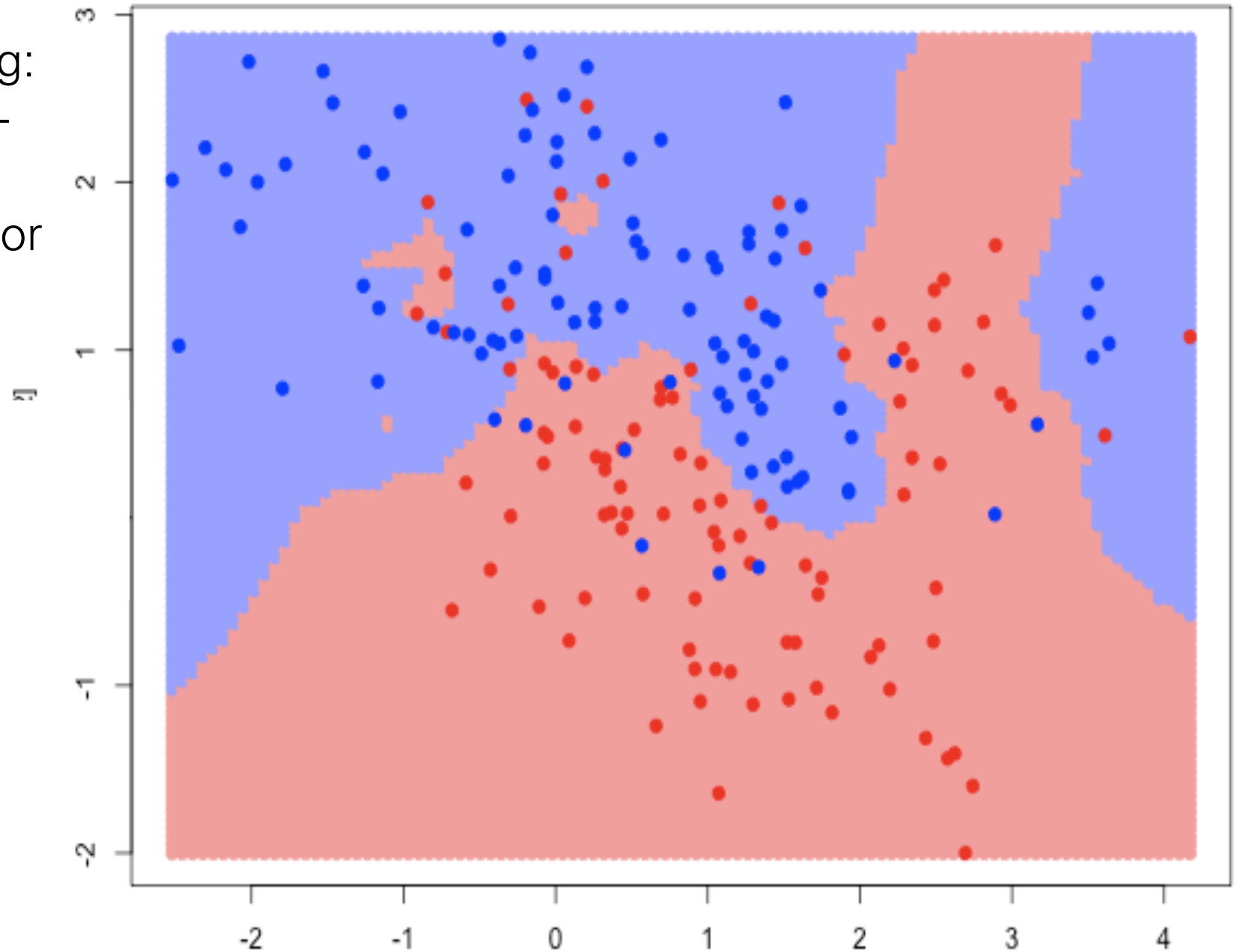


Cross-validation for k nearest neighbours



Cross-validation for k nearest neighbours

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



Recall the linear least squares classifier

X : $n \times d$ matrix with d -dimensional features for n samples

y : vector of length n : $y_i = 0$ for first, 1 for second class

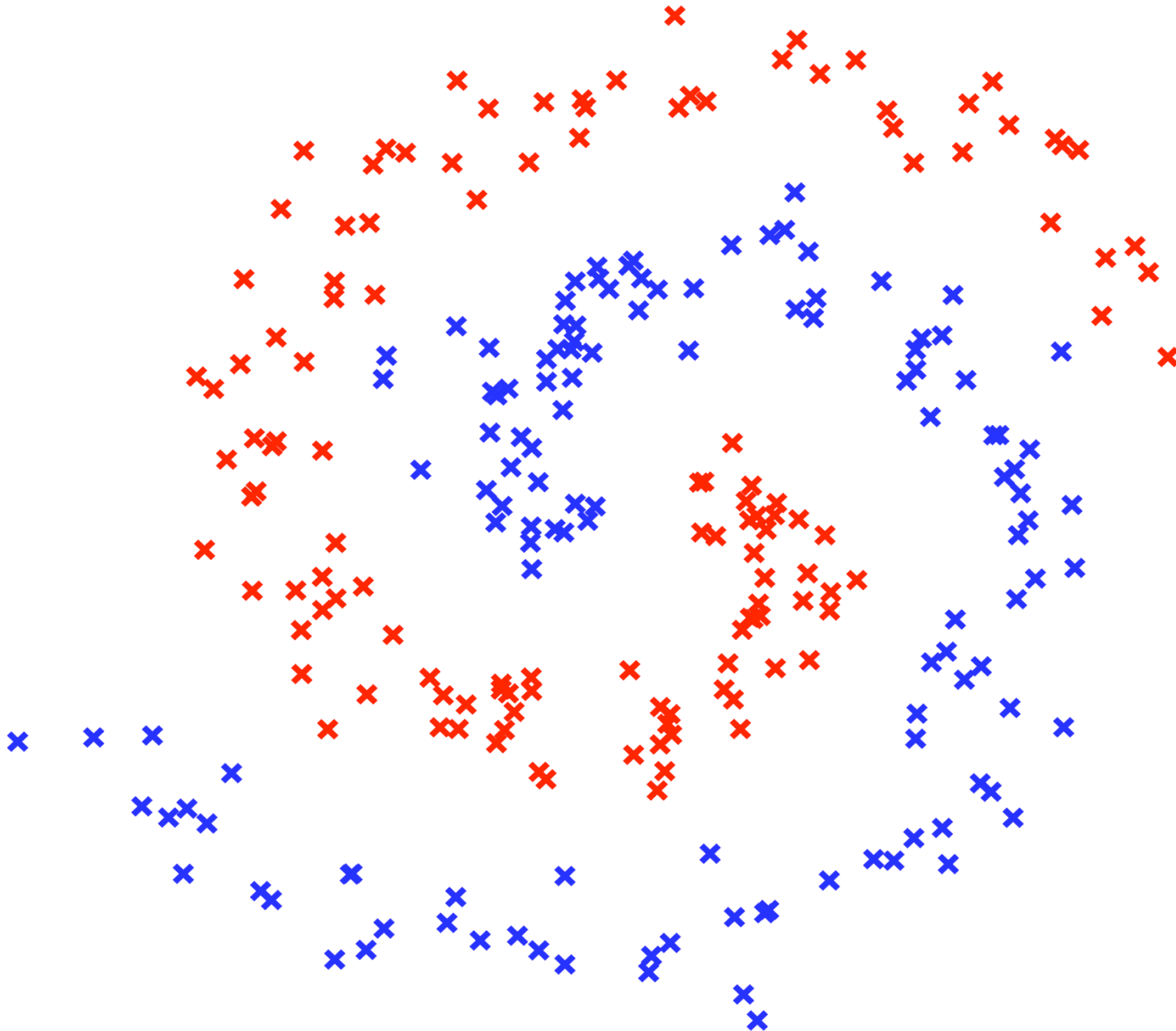
Fit linear model by minimizing the squared error:

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

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

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
simpler



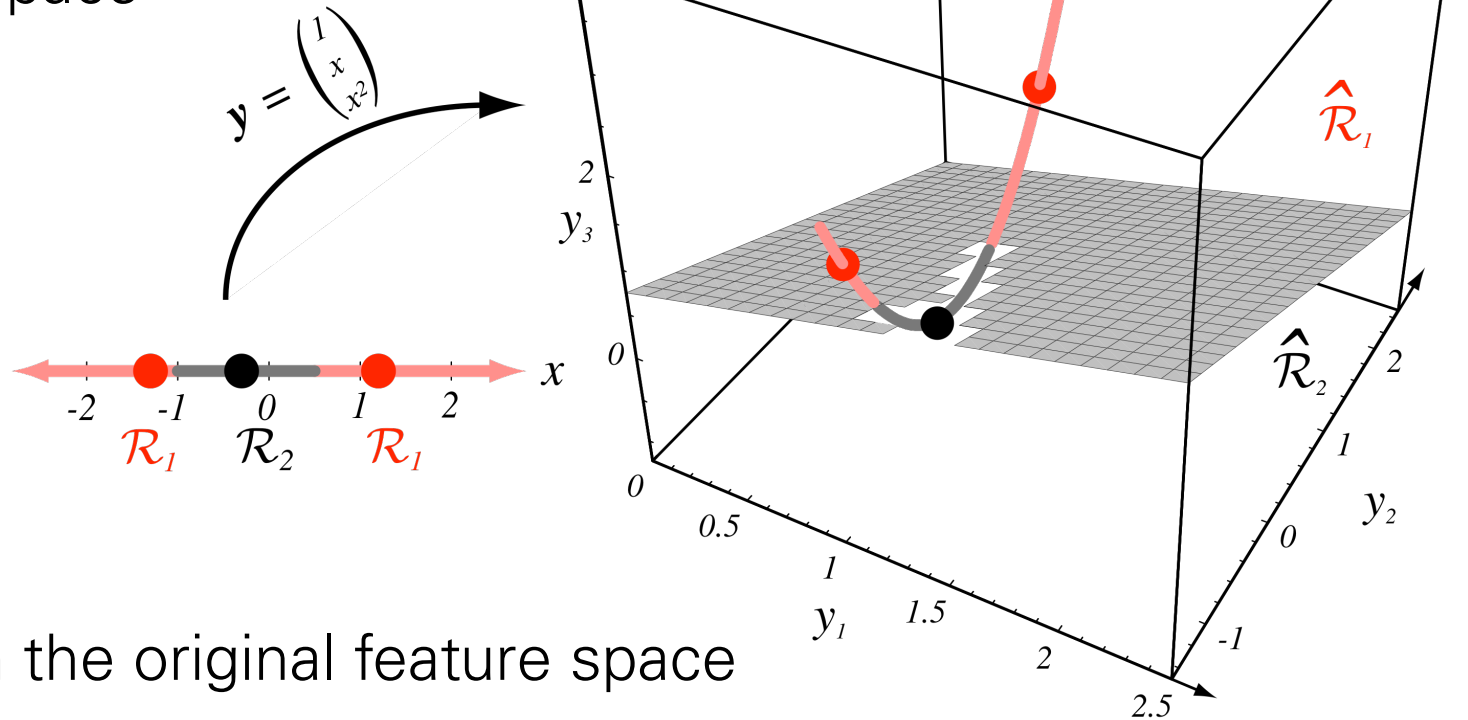
Danny MacAskill

Data Transformation & Augmentation

Apply non-linear functions, e.g.

$$f(x) = (1, x, x^2, x^3, \dots)$$

Train **linear** classifier in the new feature space

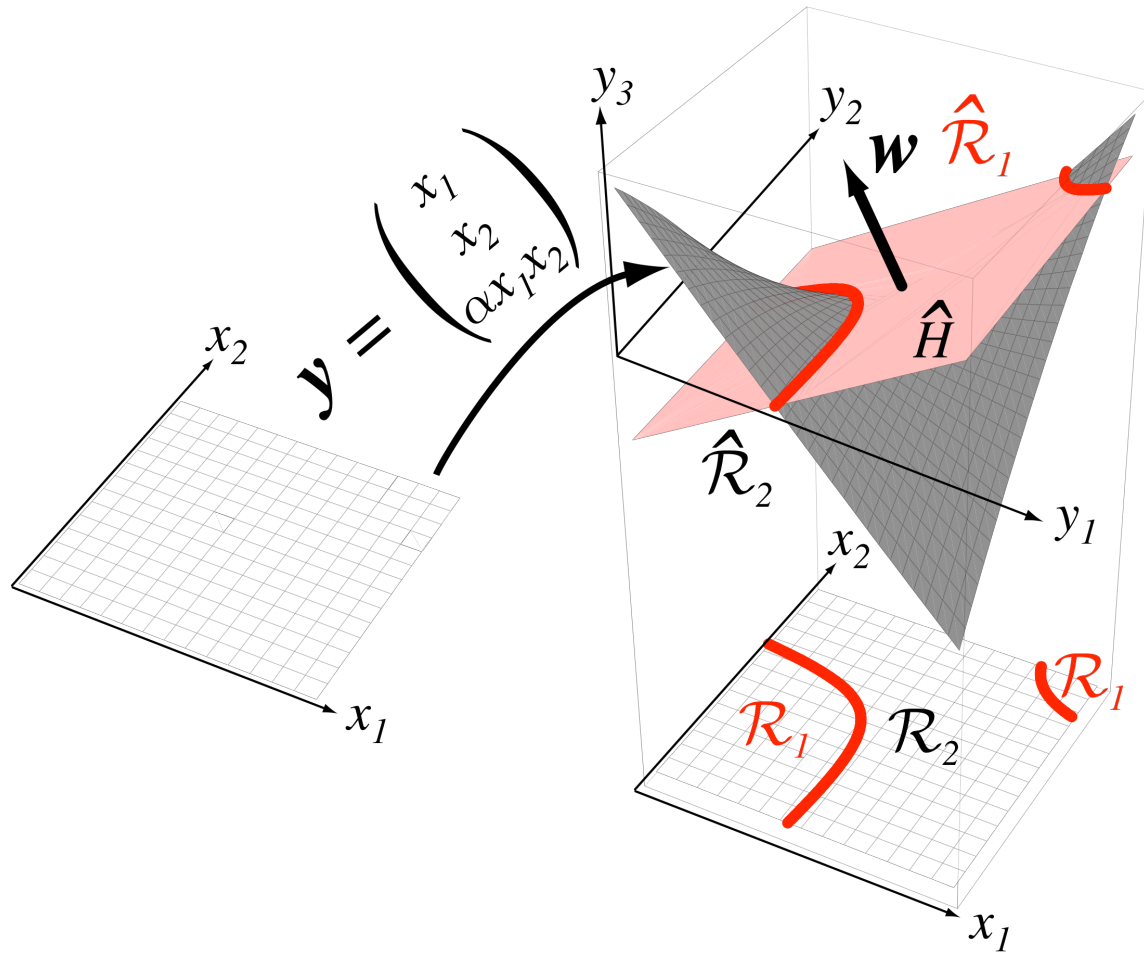


non-linear

classifier in the original feature space

Quadratic Extension

Parabolic decision boundaries can be achieved by using the product x_1x_2



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

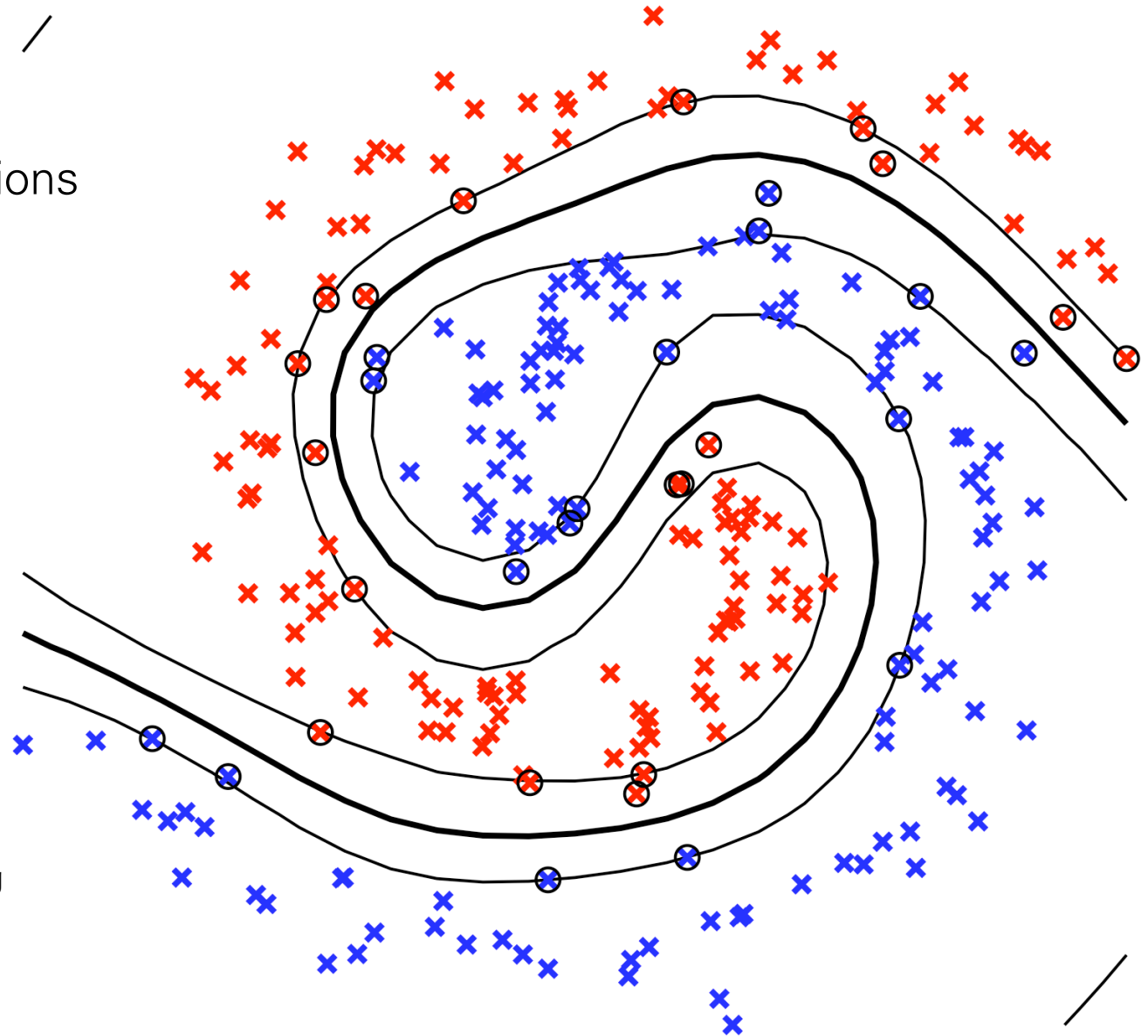
$$K(x_i, x_j) = \exp\left(-\frac{1}{2\sigma^2} \|x_i - x_j\|\right)$$

Polynomial

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

Example

Radial Basis Functions
Kernel



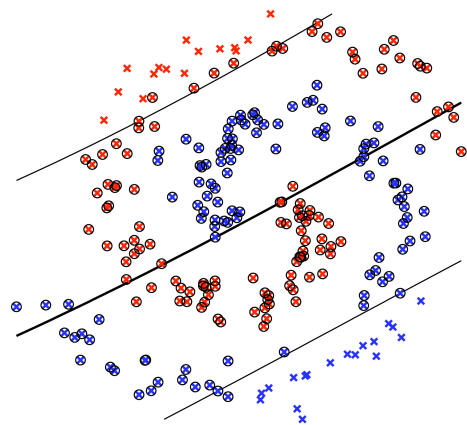
(SVM

Thick line: class separating
hyperplane

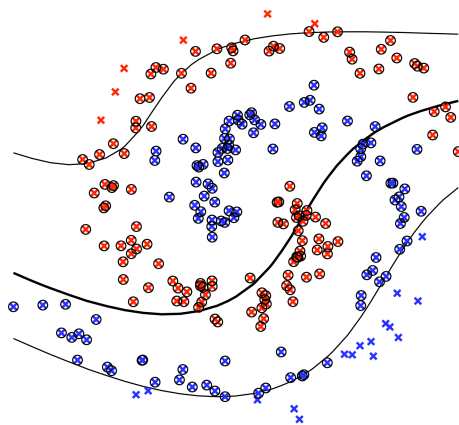
Thin line: margin

Circles: support vectors)

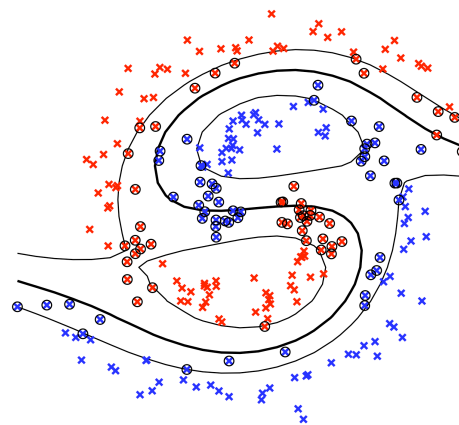
The Influence of the Kernel Parameter



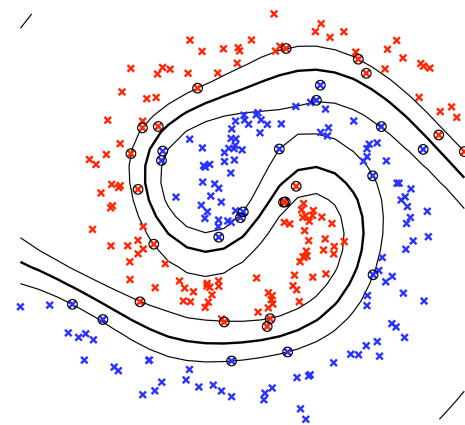
$\gamma = 0.001$



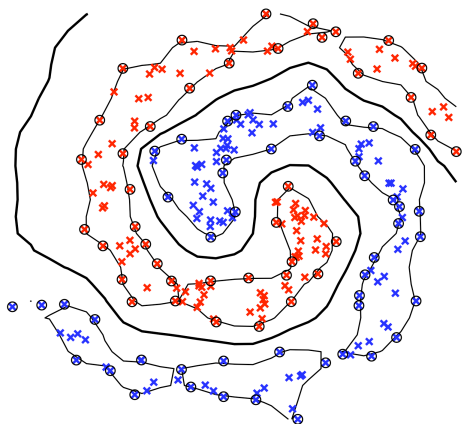
$\gamma = 0.005$



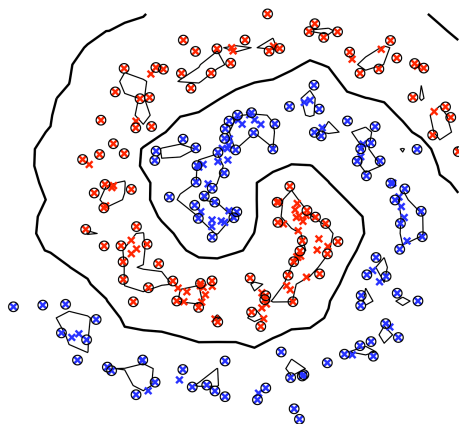
$\gamma = 0.03$



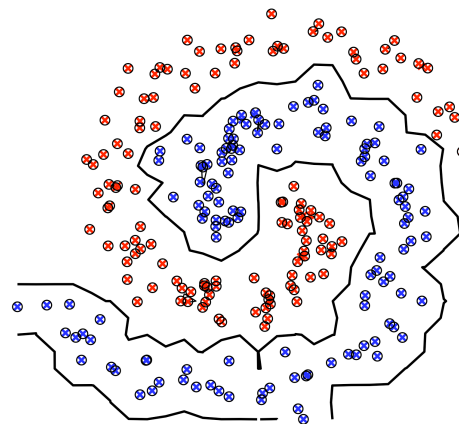
$\gamma = 0.1$



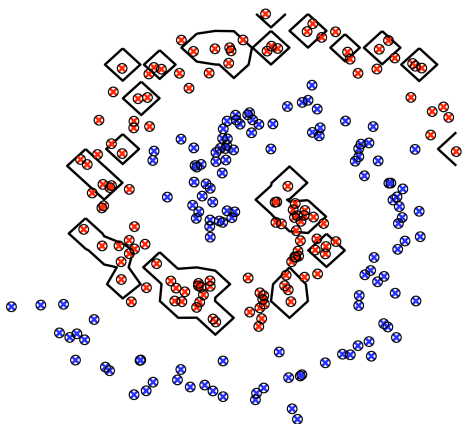
$\gamma = 1$



$\gamma = 2$



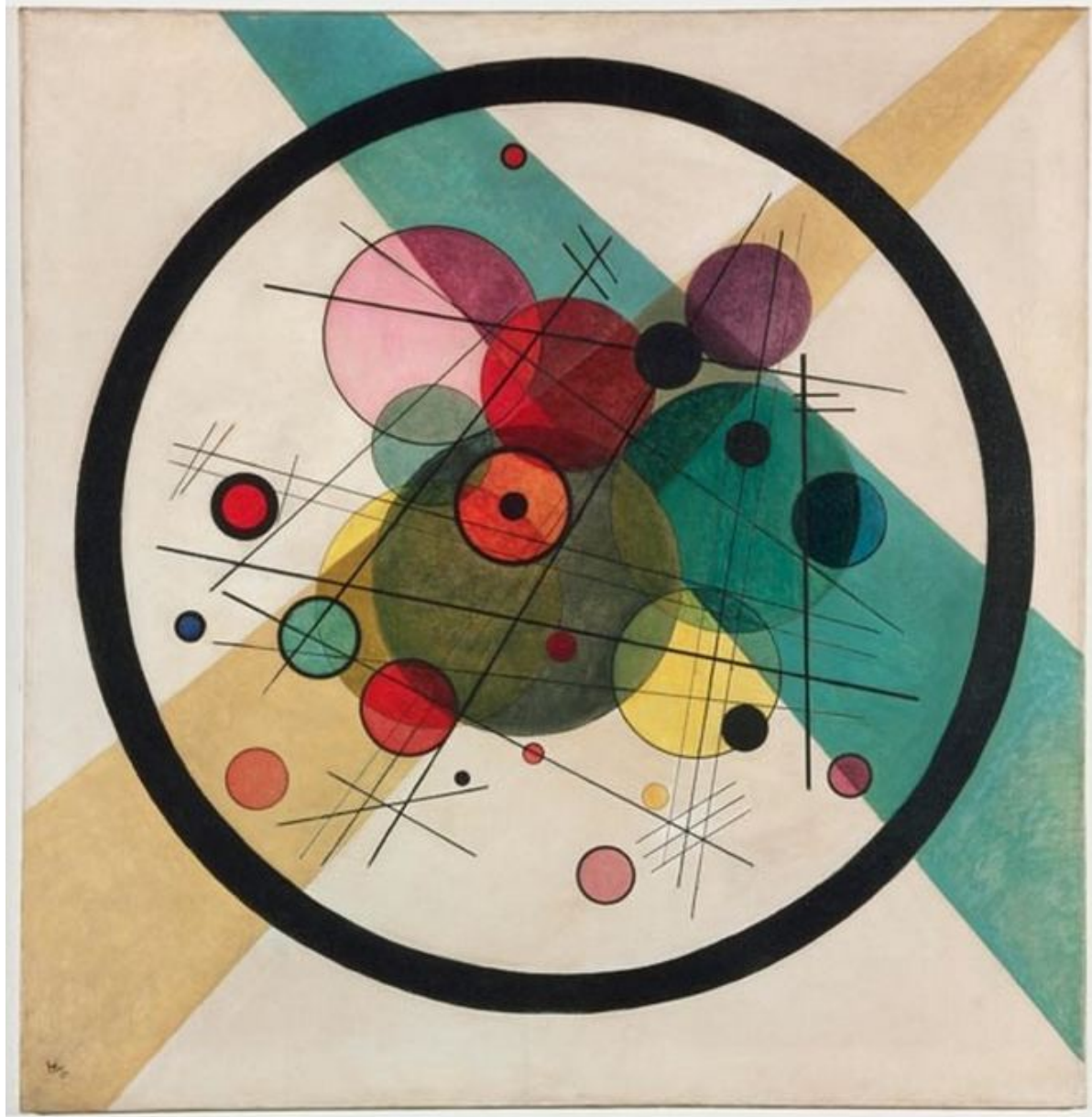
$\gamma = 20$

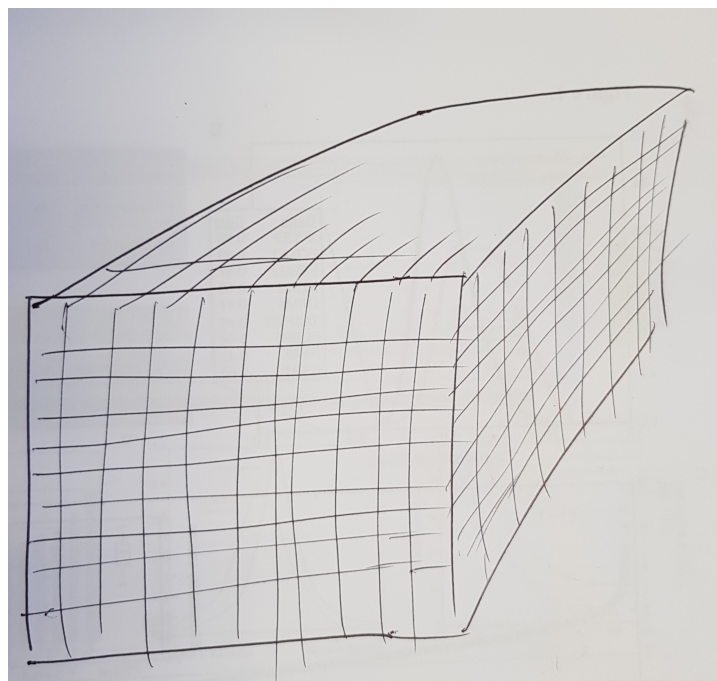


$\gamma = 200$

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

The curse of dimensionality



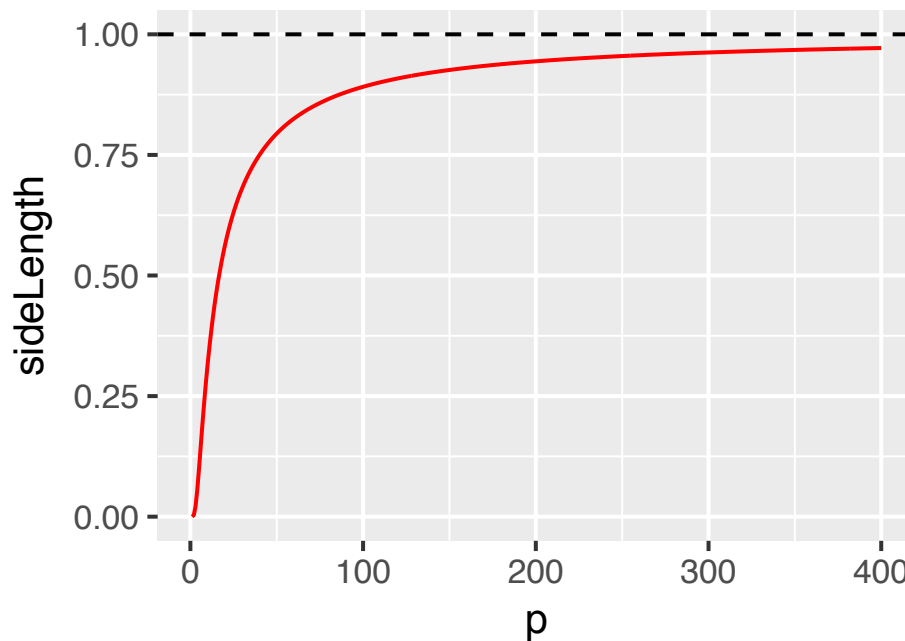


► **Question 13.12** Assume you have a dataset with 1 000 000 data points in p dimensions. The data are uniformly distributed in the unit hypercube (i. e., all features lie in the interval $[0, 1]$). What's the side length of a hypercube that can be expected to contain just 10 of the points, as a function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

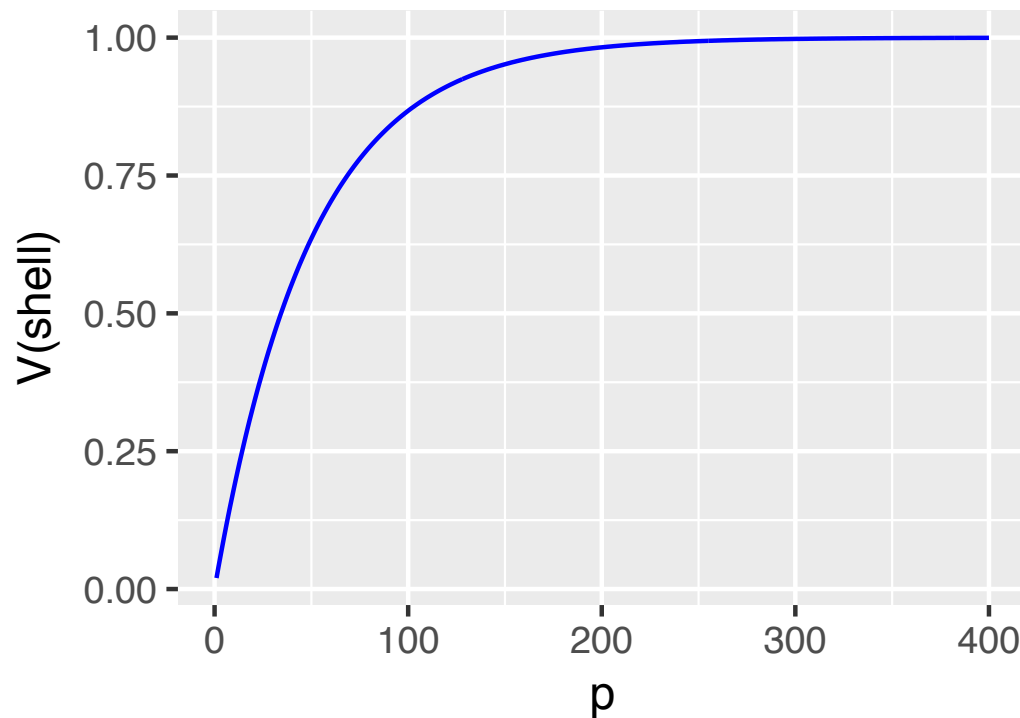


► **Question 13.12** Assume you have a dataset with 1 000 000 data points in p dimensions. The data are uniformly distributed in the unit hypercube (i. e., all features lie in the interval $[0, 1]$). What's the side length of a hypercube that can be expected to contain just 10 of the points, as a function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

► **Solution 13.12**

See Figure 13.16.

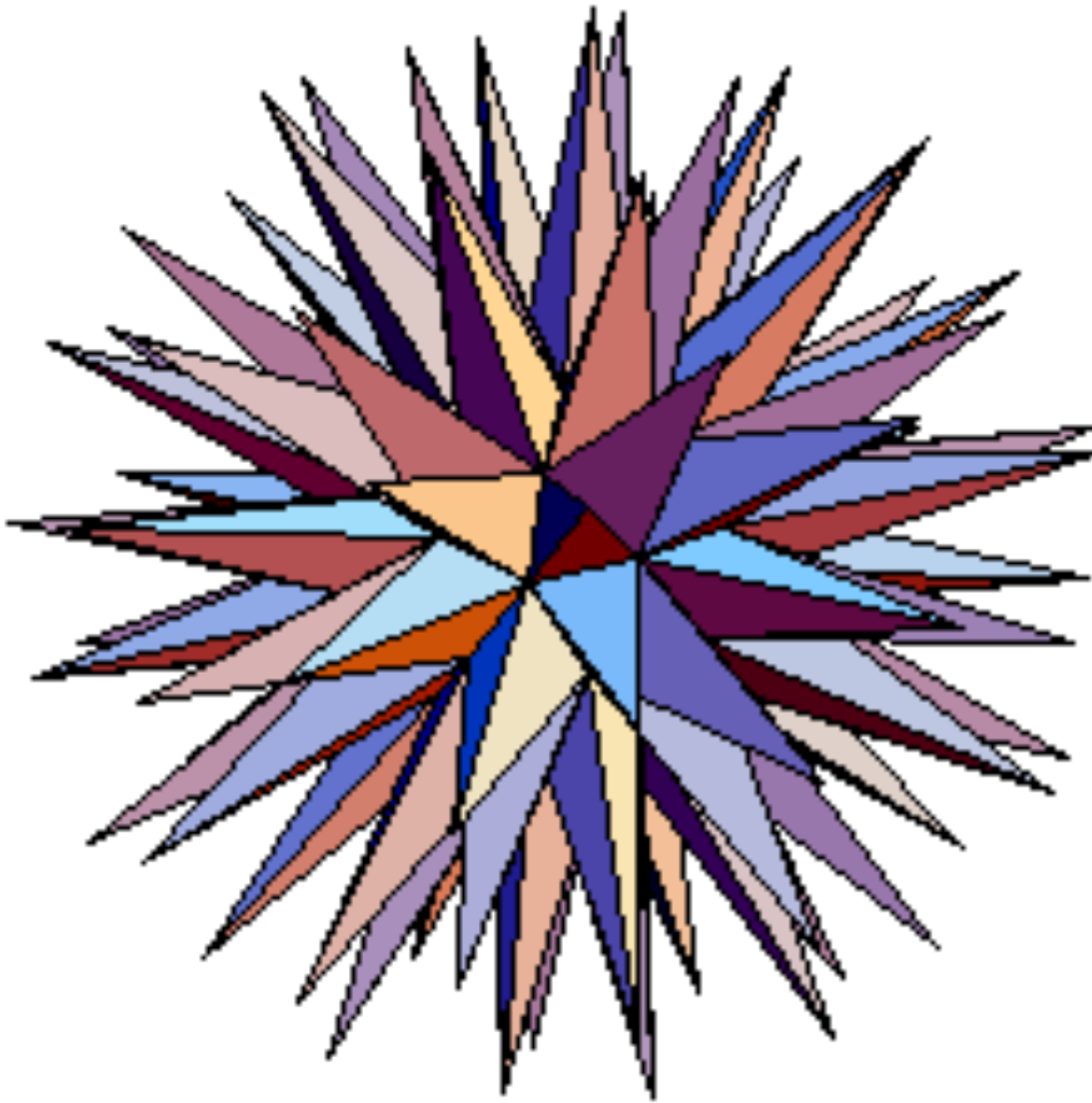
```
sideLength = function(p, pointDensity = 1e6, pointsNeeded = 10)
  (pointsNeeded / pointDensity) ^ (1 / p)
ggplot(tibble(p = 1:400, sideLength = sideLength(p)),
  aes(x = p, y = sideLength)) + geom_line(col = "red") +
  geom_hline(aes(yintercept = 1), linetype = 2)
```

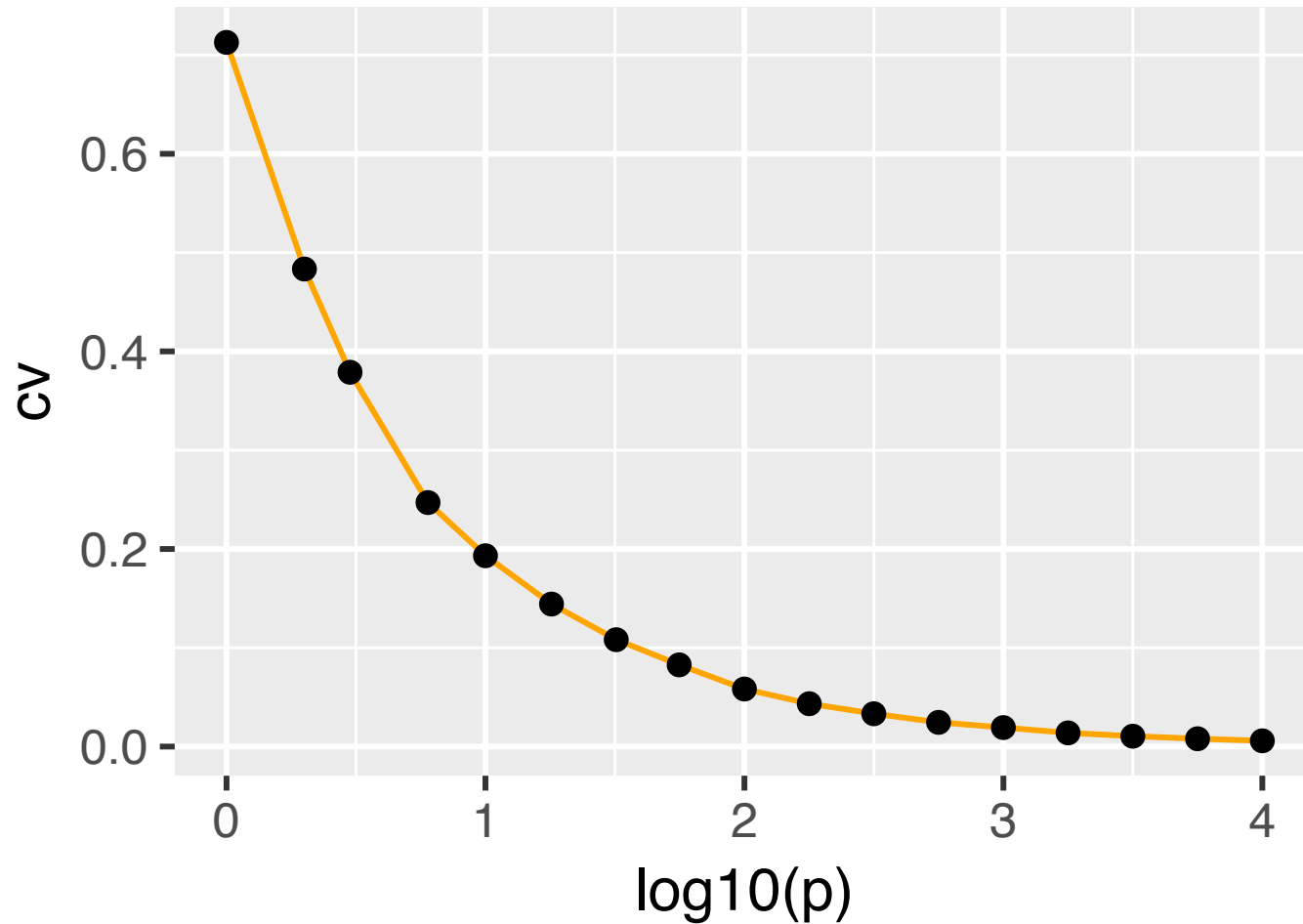
► **Question 13.13** What fraction of a unit cube's total volume is closer than 0.01 to any of its surfaces, as a function of the dimension? ◀

► **Solution 13.13**

```
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")
```



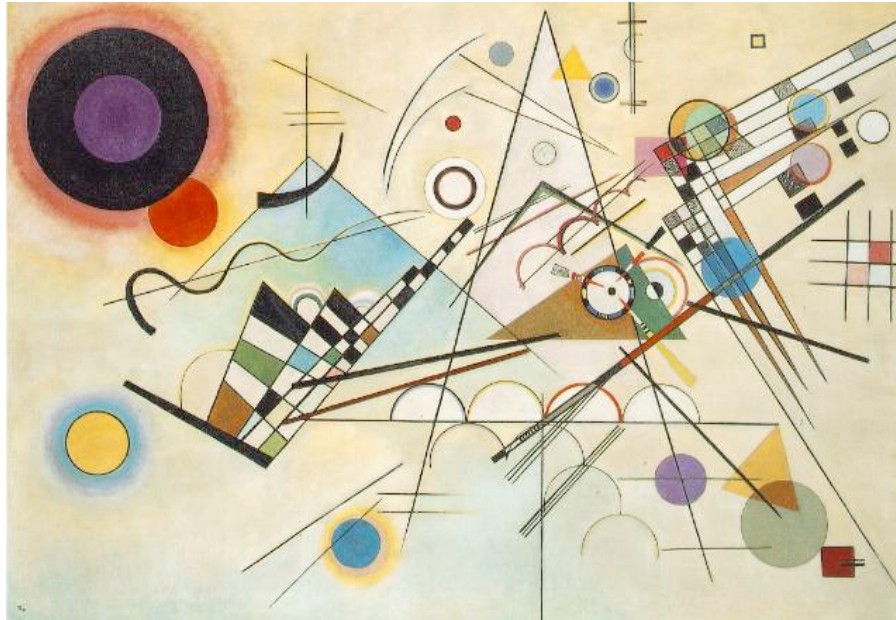
An attempt to
visualize a 7-dim
hypercube
($2^7 = 128$ corners)



► **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 be (always) too complex: overfitting
- There is a need to lower the complexity even below that of the linear classifier

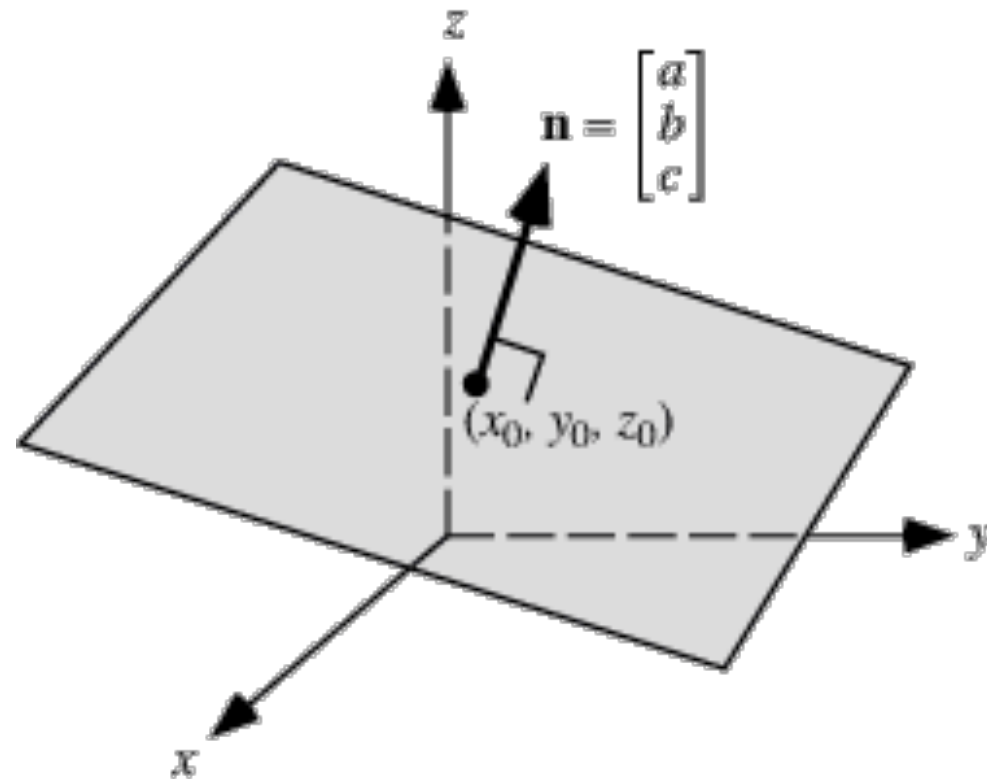


Regularisation

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:** $p(\beta) = \|\beta\|_2^2$

$$\begin{aligned}\hat{\beta} &\in \arg \min_{\beta} \ell(\beta) + \lambda \|\beta\|_1 \\ &= \arg \min_{\beta} \ell(\beta) \text{ s.t. } \|\beta\|_1 \leq t,\end{aligned}$$

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

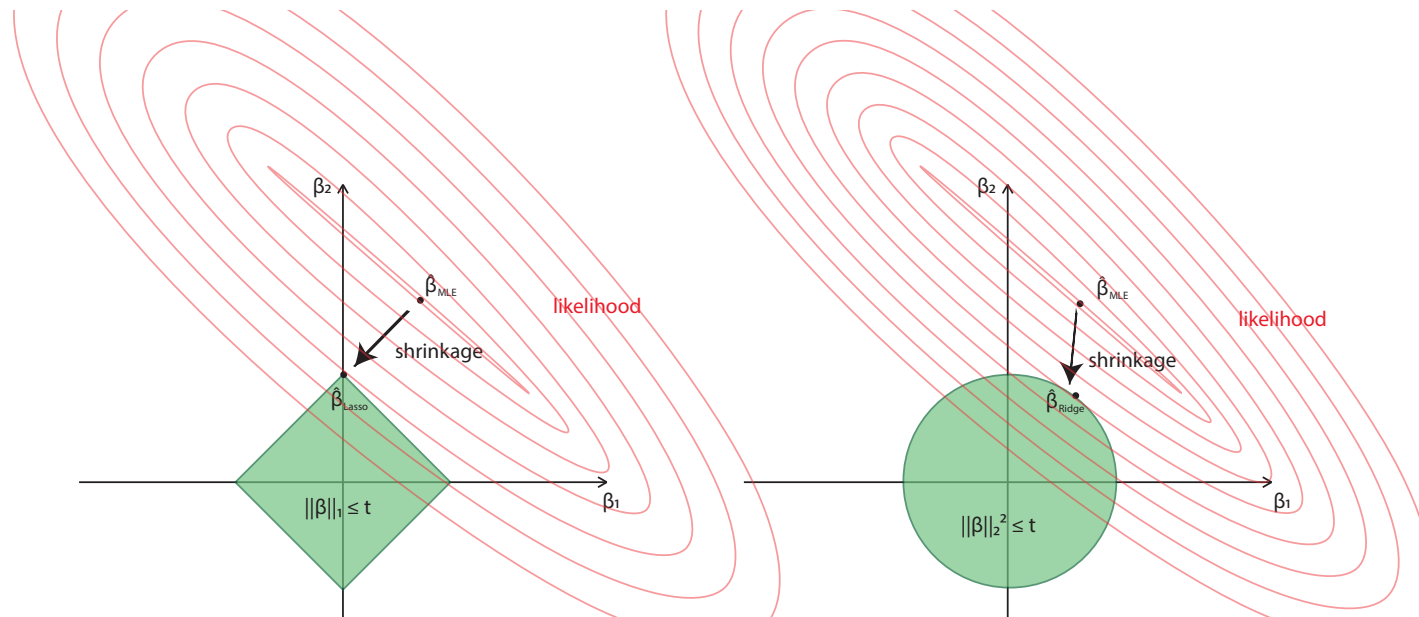


Figure:
Britta
Velten

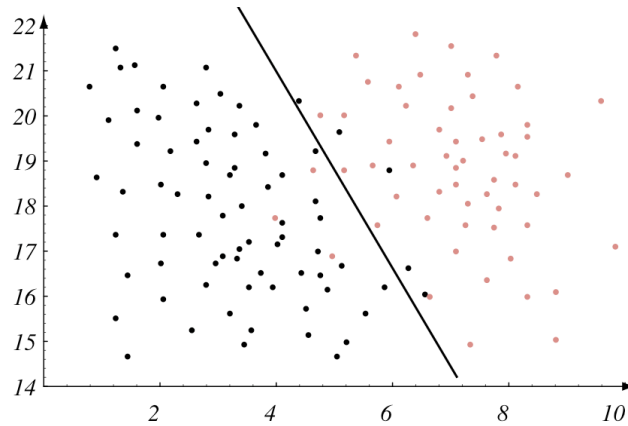
- Ridge just wants the β to be small
- Lasso snaps many of the elements to 0 ('sparsity')

(least absolute shrinkage and selection operator)

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

High bias
Low variance

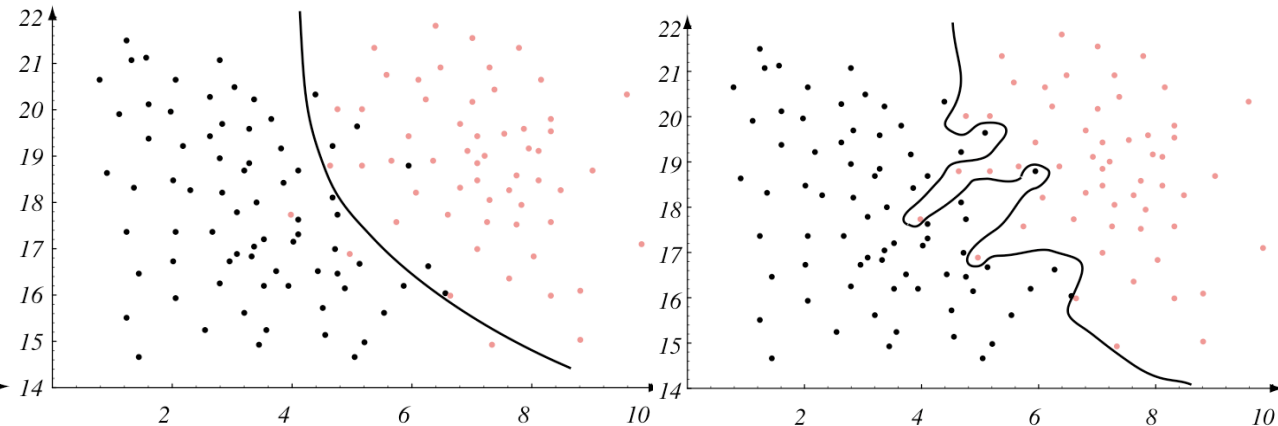
←-----



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)

Reduce complexity: regularization (Lasso, ridge, ...)

Increase complexity: data transformation, augmentation, kernels

Always assess classifiers by **cross-validation**

Springer Series in Statistics

Trevor Hastie
Robert Tibshirani
Jerome Friedman

The Elements of Statistical Learning

Data Mining, Inference, and Prediction

Second Edition

Free PDF download

 Springer

Christopher M. Bishop
with Hugh Bishop

Deep Learning

Foundations
and Concepts

 Springer