Data Mining with Linear Discriminants

Exercise: Business Intelligence (Part 6)

Summer Term 2014

Stefan Feuerriegel

Today's Lecture

Objectives

1. Recognizing the ideas of artificial neural networks and their use in R
2. Understanding the concept and the usage of support vector machines
3. Being able to evaluate the predictive performance in terms of both metrics and the receiver operating characteristic curve
4. Distinguishing predictive and explanatory power

Outline

1 Recap

2 Linear Discriminants

3 Artificial Neural Networks

4 Support Vector Machines

5 Prediction Performance

6 Wrap-Up

Outline

1 Recap

2 Linear Discriminants

3 Artificial Neural Networks

4 Support Vector Machines

5 Prediction Performance

6 Wrap-Up

Supervised vs. Unsupervised Learning

Supervised learning

- Machine learning task of inferring a function from labeled training data
- Training data includes both the input and the desired results
  $\rightarrow$ correct results (target values) are given

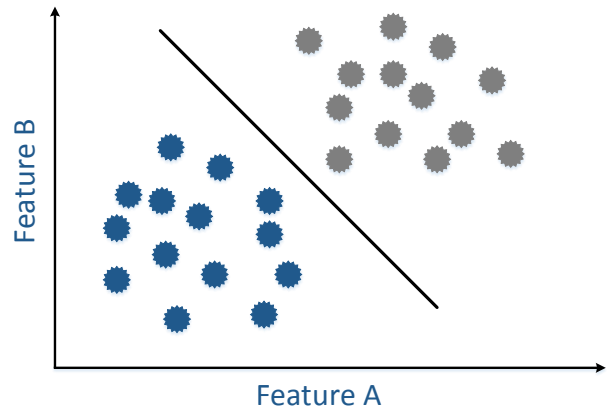
Unsupervised learning

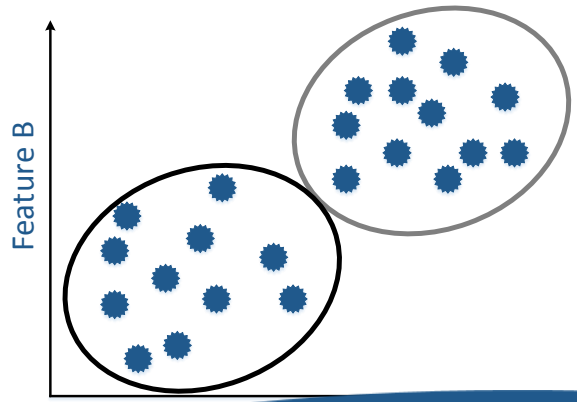
- Methods try to find hidden structure in unlabeled data
- The model is not provided with the correct results during the training
- No error or reward signal to evaluate a potential solution
- Examples:
  - Hidden Markov models
  - Dimension reduction (e. g. by principal component analysis)
  - Clustering (e. g. by k-means algorithm)
    $\rightarrow$ group into classes on the basis of their statistical properties only

Taxonomy of Machine Learning

- Machine learning estimates function and parameter in $y = f(x, w)$
- Type of method varies depending on the nature of what is predicted

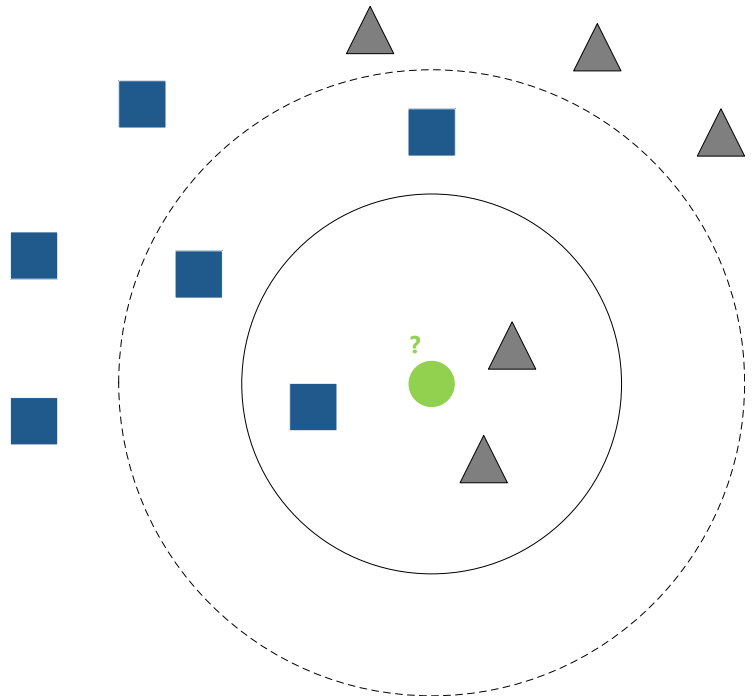
- Regression
  - Predicted value refers to a real number
  - Continuous y
- Classification
  - Predicted value refers to a class label
  - Discrete y (e. g. class membership)
- Clustering
  - Group points into clusters based on how "near" they are to one another
  - Identify structure in data





K-Nearest Neighbor Classification

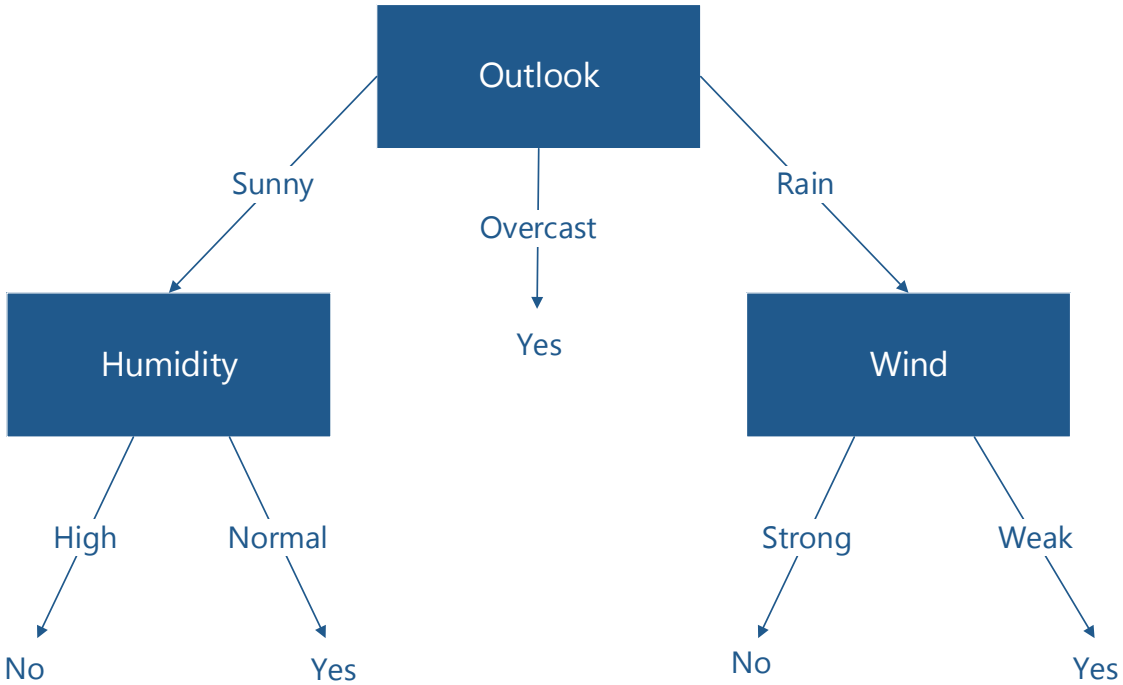
- Input: training examples as vectors in a multidimensional feature space, each with a class label
- No training phase to calculate internal parameters
- Testing: Assign to class according to k-nearest neighbors
- Classification as majority vote
- Problematic
  - Skewed data
  - Unequal frequency of classes



$\rightarrow$ What label to assign to the circle?

Decision Trees

- Flowchart-like structure in which nodes represent tests on attributes
- End nodes (leaves) of each branch represent class labels
- Example: Decision tree for playing tennis

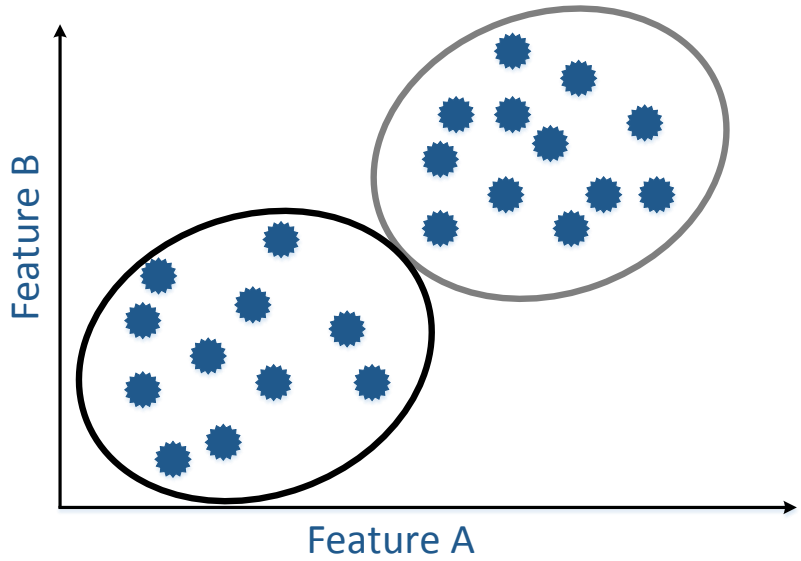


Decision Trees

- Issues
  - How deep to grow?
  - How to handle continuous attributes?
  - How to choose an appropriate attributes selection measure?
  - How to handle data with missing attributes values?
- Advantages
  - Simple to understand and interpret
  - Requires only few observations
  - Words, best and expected values can be determined for different scenarios
- Disadvantages
  - Information Gain criterion is biased in favor of attributes with more levels
  - Calculations become complex if values are uncertain and/or outcomes are linked

k-Means Clustering

- Partition *n* observations into *k* clusters in which each observation belongs to the cluster with the nearest mean, serving as a prototype of the cluster



- Computationally expensive; instead, we use efficient heuristics
- Default: Euclidean distance as metric and variance as a measure of cluster scatter

Outline

1 Recap

2 Linear Discriminants

3 Artificial Neural Networks

4 Support Vector Machines

5 Prediction Performance

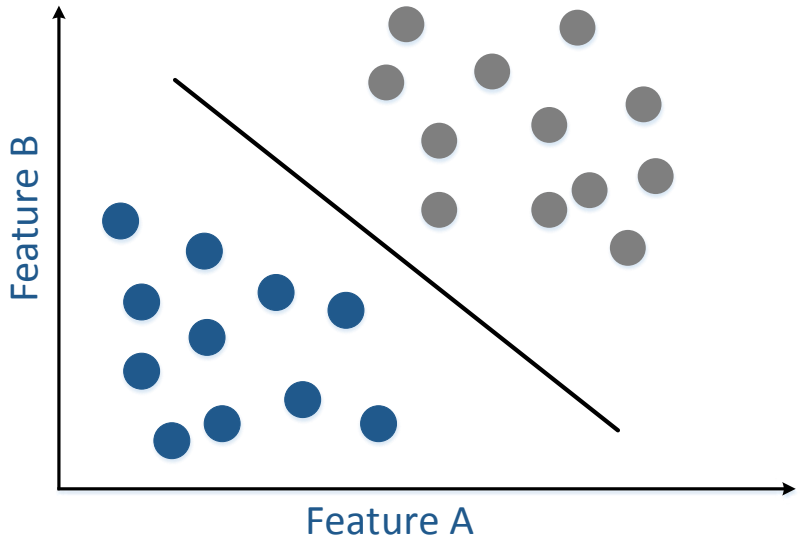
6 Wrap-Up

Classification Problems

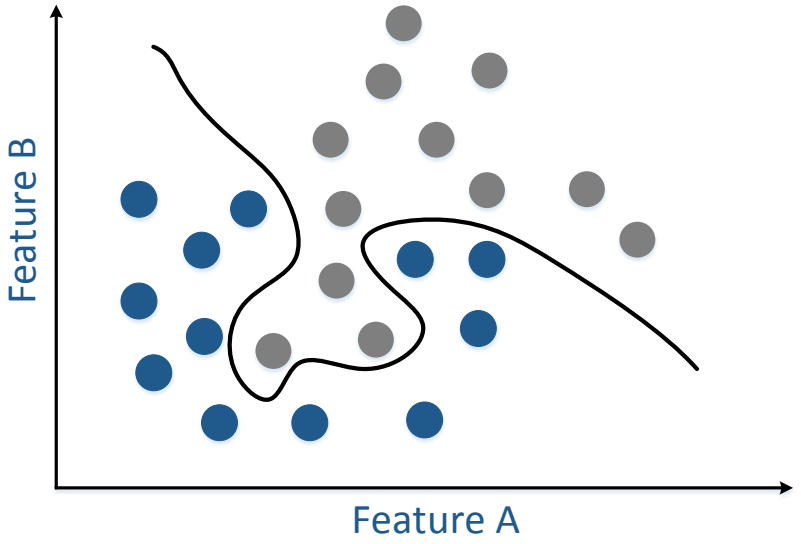
- General classification problem
  - Goal: Take a new input $\boldsymbol{x}$ and assign it to one of K classes C_k
  - Given training set $X = [\boldsymbol{x}_1 \mid \ldots \mid \boldsymbol{x}_n]^T$ with target values $T = [\boldsymbol{t}_1, \ldots, \boldsymbol{t}_n]^T$
  - Number of dimensions D, i. e. $\boldsymbol{x}_i \in \mathbb{R}^D$
  - Learn a discriminant function $y(\boldsymbol{x})$ to perform the classification
- 2-class problem with binary target values $t_i \in \{0,1\}$
  $\rightarrow$ Decide for class C_1 if $y(x) > 0$, else for class C_2
- K-class problem with 1-of-K coding scheme, i. e. $\boldsymbol{t}_i \in [0,1,0,0,0]^T$

Linear Separability

- If a data set can be perfectly classified by a linear discriminant, then we call it linearly separable



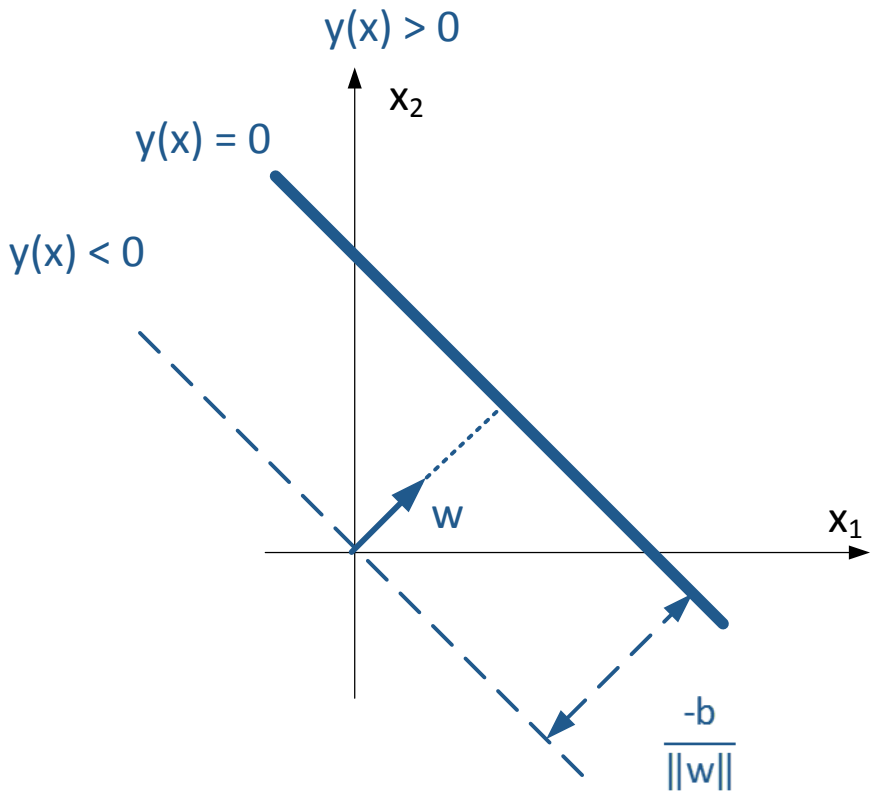
Linearly separable



Not linearly separable

Linear Discriminant Functions

- Decision boundary given by $y(\boldsymbol{x}) = \boldsymbol{w}^T\boldsymbol{x} + b = 0$ defines a hyperplane
- Classes labeled according to sign $\left(\boldsymbol{w}^T\boldsymbol{x} + b\right)$
- Normal vector w and offset $-\frac{b}{\|w\|}$



Learning Discriminant Functions

- Linear discriminant functions given by

$$y(\boldsymbol{x}) = \boldsymbol{w}^T \boldsymbol{x} + b = \sum_{i=1}^{D} w_i x_i + b$$

$$= \sum_{i=0}^{D} \tilde{w}_i \tilde{x}_i \quad \text{with } \tilde{x}_0 = 1$$

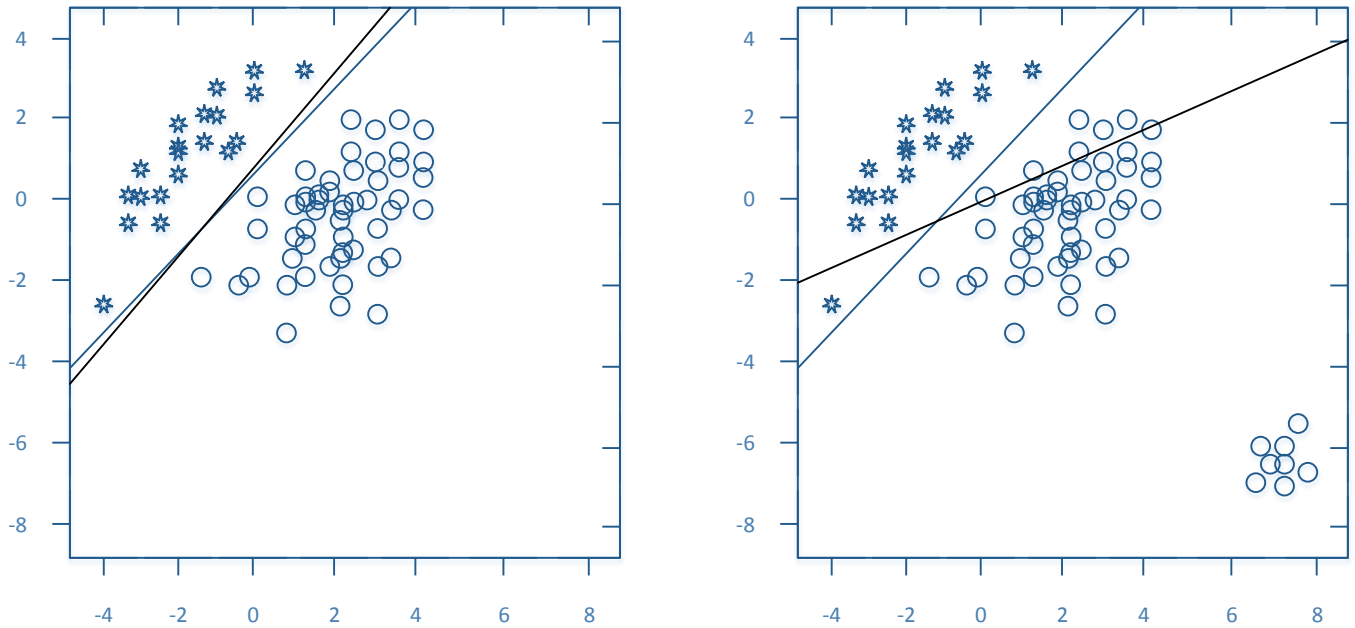
  - Weight vector w
  - "Bias" b, i. e. threshold

- Goal: Choose $\boldsymbol{w}$ and b, or $\tilde{\boldsymbol{w}}$ respectively, such that

$$\left(\boldsymbol{w}^T X + \begin{bmatrix} b \\ \vdots \\ b \end{bmatrix} \right) - \begin{bmatrix} t_1 \\ \vdots \\ t_n \end{bmatrix} \quad \Leftrightarrow \quad \tilde{\boldsymbol{w}}^T \tilde{X} - T \quad \text{is minimal}$$

Choosing Discriminant Function

- Solving $\tilde{\boldsymbol{w}}^T \tilde{X} - T$ by least-squares has drawbacks
  - Least-squares is very sensitive to outliers
  - Error function penalizes predictions that are "too correct"
  - Works only for linearly separable problems
  - Least-squares assumes Gaussian distribution



- Alternative solutions (e. g. in blue): Generalized linear models ($\rightarrow$ neural networks), support vector machines, etc.

$\rightarrow$ from Leibe (2010).

Outline

1 Recap

2 Linear Discriminants

3 Artificial Neural Networks

4 Support Vector Machines

5 Prediction Performance

6 Wrap-Up

Generalized Linear Model

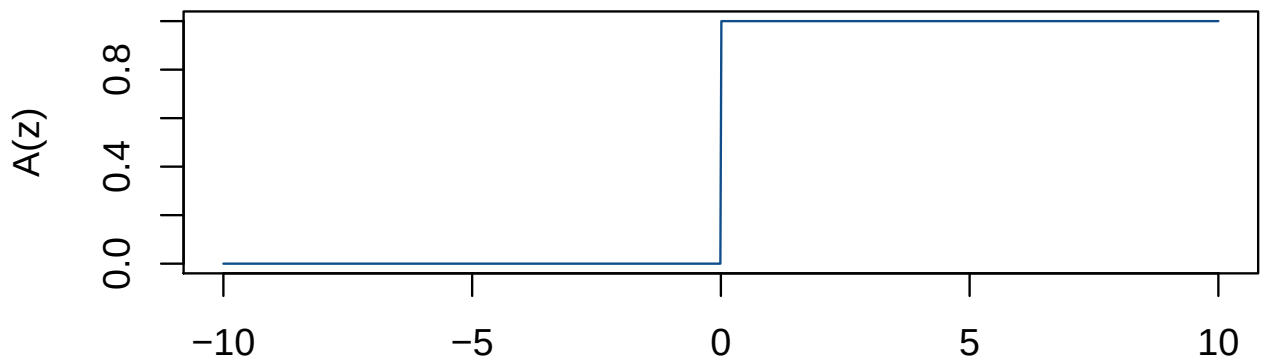
- Linear model

$$y(\boldsymbol{x}) = \boldsymbol{w}^T \boldsymbol{x} + b$$

- Generalized linear model with activation function A

$$y(\boldsymbol{x}) = \mathrm{A}\left(\boldsymbol{w}^T \boldsymbol{x} + b\right)$$

- Other than least-squares, choice of activation function should limit influence of outliers, e. g. using a threshold as A



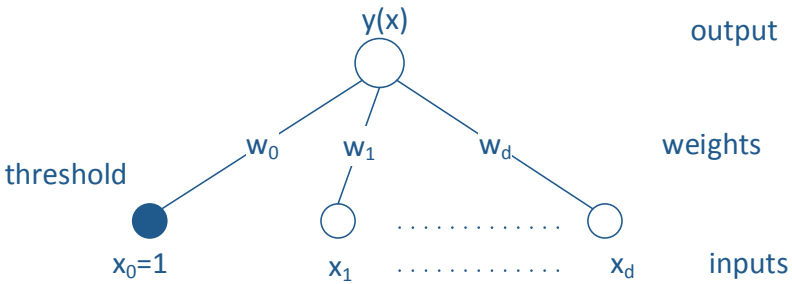
Relationship to Neural Networks

- In 2-class case

$$y(\boldsymbol{x}) = \sum_{i=0}^{D} \mathrm{A}\left(w_i x_i\right)$$

with $x_0 = 1$

- Single-layer perceptron

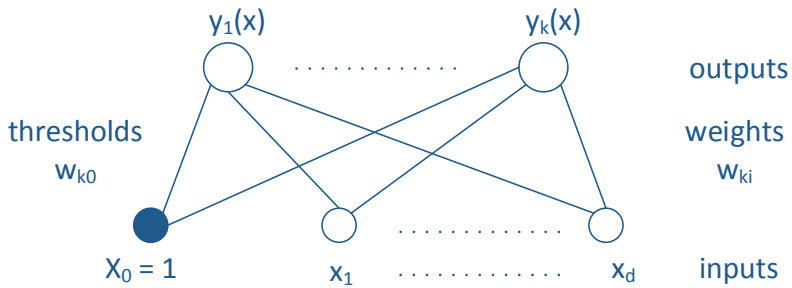


- In multi-class case

$$y_k(\boldsymbol{x}) = \sum_{i=0}^{D} \mathrm{A}\left(w_{k,i} x_i\right)$$

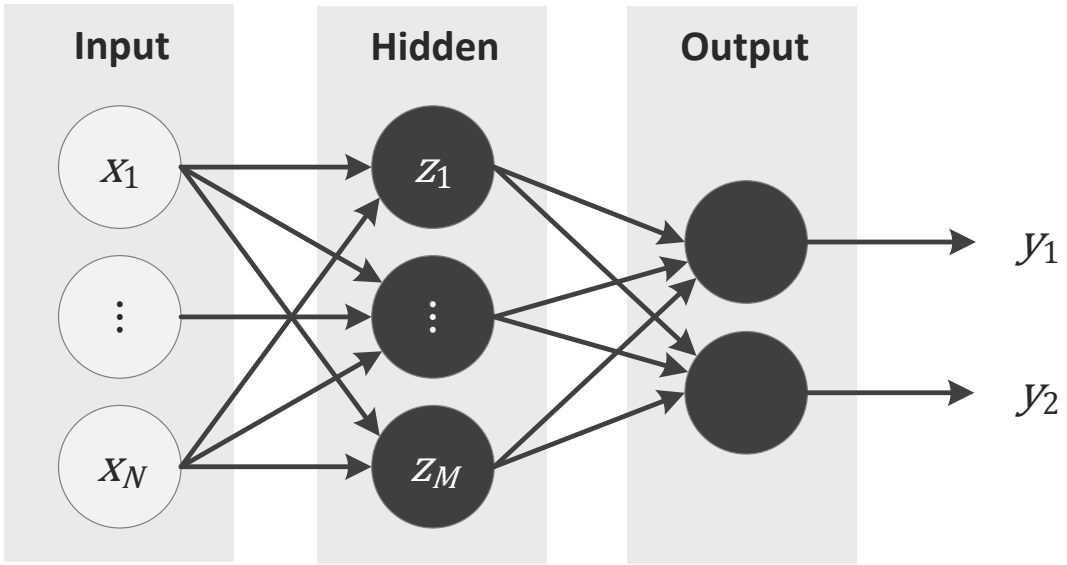
with $x_0 = 1$

- Multi-class perceptron



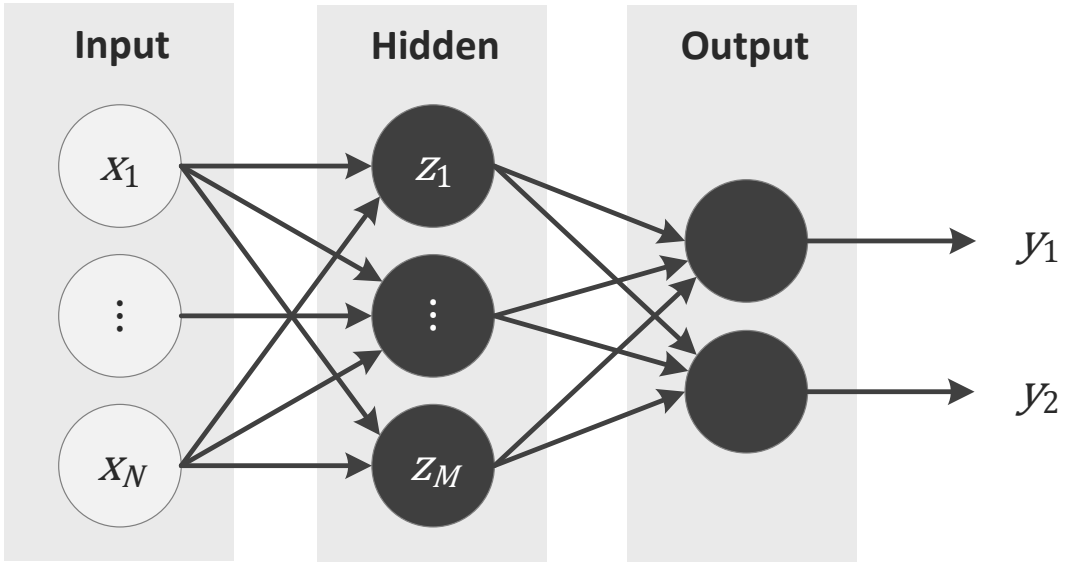
Artificial Neural Networks

- Artificial neural networks (ANN) are computational models to compute $f : X \mapsto Y$, inspired by the central nervous system
- Compute f by feeding information through the network
- Represented as a system of connected neurons
- ANNs are universal approximators among continuous functions (under certain mild assumptions)



Layers in Neural Networks

- Neurons are arranged in three (or more) layers
  - First layer: Input neurons receive the input vector $\boldsymbol{x} \in X$
  - Hidden layer(s): Connect input and output neurons
  - Final layer: Output neurons compute a response $\tilde{\boldsymbol{y}} \in Y$



- When neurons are connected as a directed graph without cycles, this is called a feed-forward ANN

Feeding Information through Neural Networks

- Input z_j of each neuron $j = 1, \ldots, M$ is a weighted sum of all previous neurons calculated as

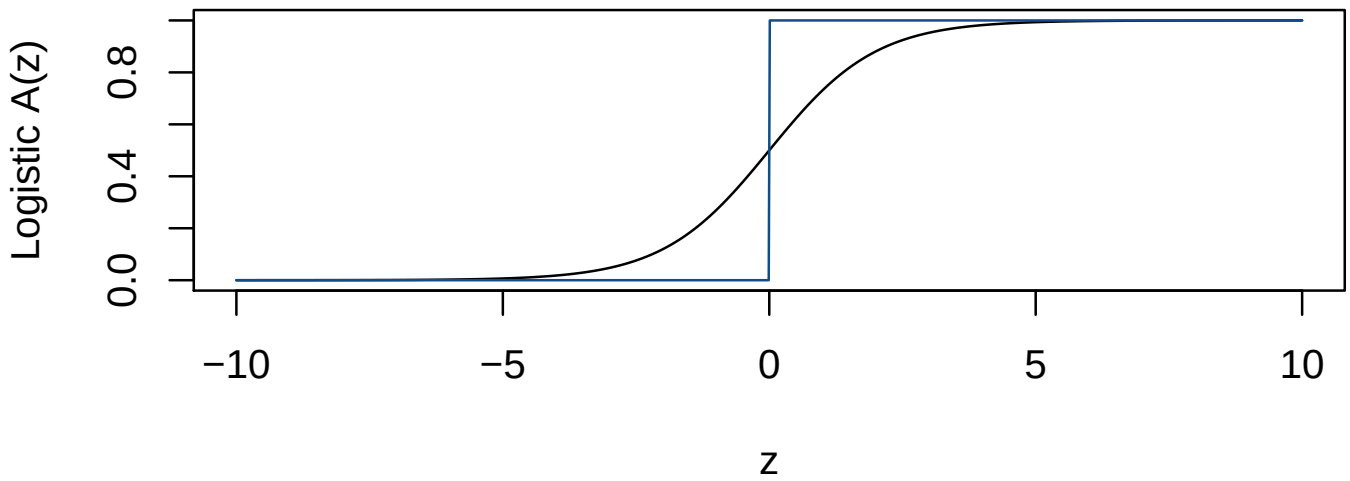
$$z_j = \mathrm{A}\left(w_{0,j} + \sum_{i=1}^{N} w_{i,j} x_i\right) = \mathrm{A}\left(w_{0,j} + \boldsymbol{w}_j^T \boldsymbol{x}\right)$$

  where
  - x_i are the values from the input layer
  - suitable coefficients $w_{i,j}$ for $i = 1, \ldots, N$ and $j = 1, \ldots, M$
- Predefined non-linear function A is referred to as the activation function
- Frequent choice: Logistic function

$$\mathrm{A}(z) = \frac{1}{1 + \mathrm{e}^{-z}}$$

- Coefficients $w_{i,j}$ learned from e. g. a back-propagation algorithm

Logistic Function



- Resembles a threshold function

$$A(z) \approx \begin{cases} 0, & z < 0, \\ 1, & z > 0 \end{cases}$$

Neural Networks in R

- Loading required library `nnet`

```
library(nnet)
```

- Accessing credit scores

```
library(caret)
data(GermanCredit)
```

- Split data into index subset for training (20 %) and testing (80 %) instances

```
inTrain <- runif(nrow(GermanCredit)) < 0.2
```

Neural Networks in R

- Train neural network with *n* nodes in the hidden layer via `nnet(formula, data=d, size=n ...)`

```
nn <- nnet(Class ~ ., data=GermanCredit[inTrain,],
           size=15, maxit=100, rang=0.1, decay=5e-4)

## # weights:  946
## initial  value 139.233471
## iter  10 value 120.009852
## iter  20 value 111.986450
## iter  30 value 92.182560
## iter  40 value 88.672309
## iter  50 value 85.914152
## iter  60 value 85.220372
## iter  70 value 85.121310
## iter  80 value 85.061444
## iter  90 value 84.634114
## iter 100 value 81.262052
## final  value 81.262052
## stopped after 100 iterations
```

Neural Networks in R

- Predict credit scores via `predict(nn, test, type="class")`

```
pred <- predict(nn, GermanCredit[-inTrain,],
                type="class")
```

- Confusion matrix via `table(pred=pred_classes, true=true_classes)`

```
cm <- table(pred=pred,
            true=GermanCredit$Class[-inTrain])
```

- Calculate accuracy

```
sum(diag(cm))/sum(sum(cm))

## [1] 0.7387
```

Outline

1 Recap

2 Linear Discriminants

3 Artificial Neural Networks

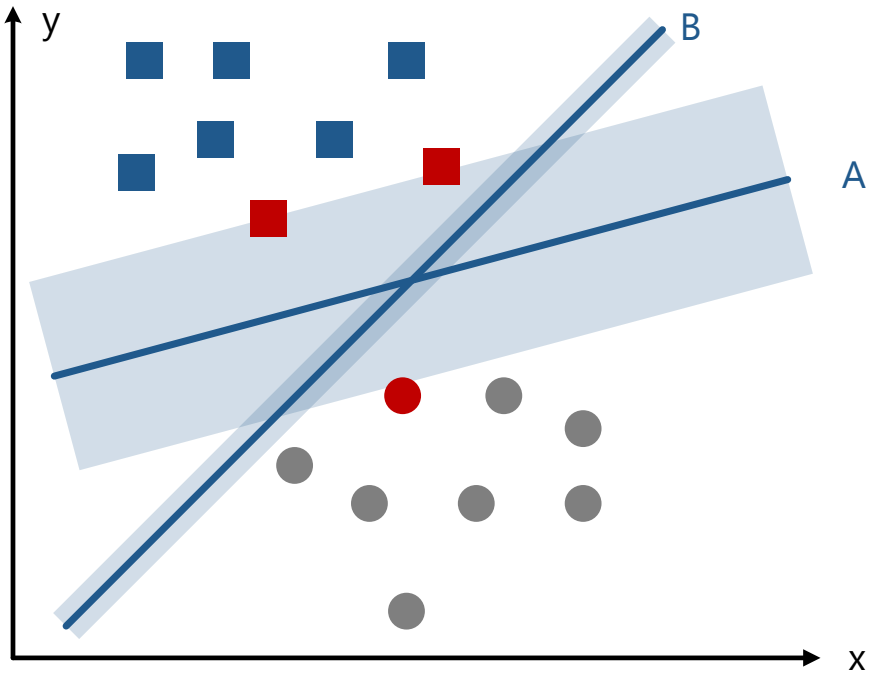
4 Support Vector Machines

5 Prediction Performance

6 Wrap-Up

Support Vector Machine (SVM)

- Which of these linear separators is optimal?
- Idea: Maximize separating margin (here: A)
  - Data points on the margin are called support vectors
  - When calculating decision boundary, only support vectors matter; other training data is ignored
  - Formulation as convex optimization problem with global solution



SVM in R

- Loading required library `e1071`

```
library(e1071)
```

- Accessing credit scores

```
library(caret)
data(GermanCredit)
```

- Split data into index subset for training (20%) and testing (80%) instances

```
inTrain <- runif(nrow(GermanCredit)) < 0.2
```

SVM in R

- Train support vector machine for classification via `svm(formula, data=d, type="C-classification")`

```
model <- svm(Class ~ ., data=GermanCredit[inTrain,],
             type="C-classification")
```

- Predict credit scores for testing instances `test` via `predict(svm, test)`

```
pred <- predict(model, GermanCredit[-inTrain, ])
head(cbind(pred, GermanCredit$Class[-inTrain]))
```

```
##   pred
## 2    2 1
## 3    2 2
## 4    2 2
## 5    2 1
## 6    2 2
## 7    2 2
```

- First row gives predicted outcomes, second are actual (true) values

SVM in R

- Confusion matrix via `table(pred=pred_classes, true=true_classes)`

```
cm <- table(pred=pred,
            true=GermanCredit$Class[-inTrain])
cm
```

```
##        true
## pred    Bad Good
##   Bad    57    1
##   Good  243  698
```

- Calculate accuracy

```
sum(diag(cm))/sum(sum(cm))
```

```
## [1] 0.7558
```

Outline

1 Recap

2 Linear Discriminants

3 Artificial Neural Networks

4 Support Vector Machines

5 Prediction Performance

6 Wrap-Up

Assessment of Models

1. Predictive performance (measured by accuracy, recall, F1, ROC, . . .)
2. Computation time for both model building and predicting
3. Robustness to noise in predictor values
4. Scalability
5. Interpretability $\rightarrow$ transparency, ease of understanding

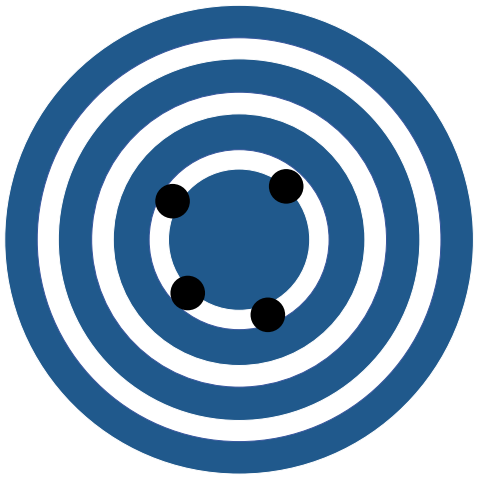
Model	**Allows** $n < k$	**Interpret.**	**# Tuning Parameters**	**Robust to Predictor Noise**	**Comp. Time**
Linear Regression	✗	✓	0	✗	✓
ANN	✓	✗	1–2	✗	✗
SVM/SVR	✓	✗	1–3	✗	✗
k-NN	✓	✗	1	o	✓
Single Tree	✓	o	1	✓	✓
Random Forest	✓	✗	0–1	✓	✗
Boosted Trees	✓	✗	3	✓	✗

$\rightarrow$ from Kuhn & Johnson (2013). Applied Predictive Modeling.

Accuracy vs. Precision

Accuracy Closeness to the actual (true) value

Precision Similarity of repeated measurements under unchanged conditions



→ High accuracy, but low precision

→ High precision, but low accuracy

Confusion Matrix

Confusion matrix (also named contingency table or error matrix) displays predictive performance

	Condition (as determined by Gold standard)		
	True	**False**	
Positive Outcome	True Positive (TP)	False Positive (FP) $\rightarrow$ Type I Error $\rightarrow$ False Alarm	Precision or Positive Predictive Value $= \frac{TP}{TP+FP}$
Negative Outcome	False Negative (FN) $\rightarrow$ Type II Error / Miss	True Negative (TN)	
	Sensitivity† = TP Rate $= \frac{TP}{TP+FN}$	Specificity = TN Rate $= \frac{TN}{FP+TN}$	**Accuracy** $= \frac{TP+TN}{\text{Total}}$

† Equivalent with hit rate and recall

Confusion Matrix

Example: Blood probe to test for cancer

	Patient with Cancer		
	True	**False**	
Positive Blood Test Outcome	TP: Cancer correctly diagnosed	FP: Healthy person diagnosed cancer	Precision $= \frac{TP}{TP+FP}$
Negative Blood Test Outcome	FN: Cancer not diagnosed	TN: Healthy person diagnosed as healthy	
	Sensitivity = TP Rate $= \frac{TP}{TP+FN}$	Specificity = TN Rate $= \frac{TN}{FP+TN}$	**Accuracy** $= \frac{TP+TN}{\text{Total}}$

Different loss functions: Redundant, €1000 check in FP case, compared to lethal outcome in FN case

Assessing Prediction Performance

Imagine the following confusion matrix with an accuracy of 65 %

	Patient with Cancer		
	True	**False**	
Positive Blood Test Outcome	$TP = 60$	$FP = 5$	Precision $= \frac{TP}{TP+FP} \approx 0.92$
Negative Blood Test Outcome	$FN = 30$	$TN = 5$	
	Sensitivity $= \frac{TP}{TP+FN} \approx 0.67$	Specificity $= \frac{TN}{FP+TN} = 0.50$	**Accuracy** $= \frac{TP+TN}{\text{Total}} = 0.65$

Is this a "good" result? No, because of unevenly distributed data, a model which always guesses positive will score an accuracy of 60 %

Prediction Performance in R

- Confusion matrix in variable `cm`

```
##      True False
## Pos    30    10
## Neg    20    40
```

- Calculating accuracy

```
(cm[1,1]+cm[2,2])/
(cm[1,1]+cm[1,2]+cm[2,1]+cm[2,2])
```

```
## [1] 0.7
```

```
# Alternative that works also for multi-class data
sum(diag(cm))/sum(sum(cm))
```

```
## [1] 0.7
```

- Calculating precision

```
cm[1, 1]/(cm[1, 1] + cm[1, 2])
```

```
## [1] 0.75
```

Trade-Off: Sensitivity vs. Specificity/Precision

- Performance goals frequently place more emphasis on either sensitivity or specificity/precision
  - Example: Airport scanners triggered on low-risk items like belts (low precision), but reduce risk of missing risky objects (high sensitivity)
- Trade-Off: F1 score is the harmonic mean of precision and sensitivity

$$F1 = \frac{2\,TP}{2\,TP + FP + FN}$$

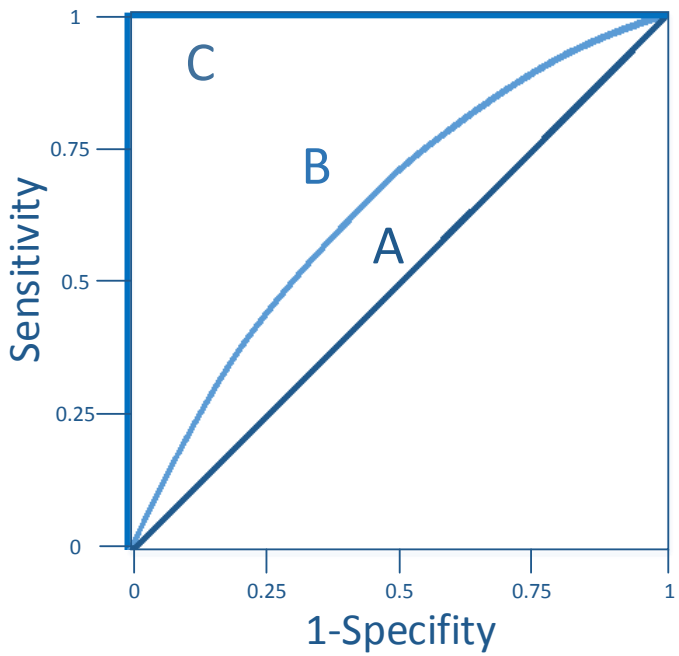
- Visualized by receiver operating characteristic (ROC) curve

Receiver Operating Characteristic (ROC)

ROC illustrates performance of binary classifier as its discrimination threshold $y(\boldsymbol{x})$ is varied

Interpretation:

- Curve A is random guessing (50 % correct guesses)
- Curve from model B performs better than A, but worse than C
- Curve C from perfect prediction



Area south-east of curve is named area under the curve and should be maximized

ROC in R

- Load required library `pROC`

```
library(pROC)
```

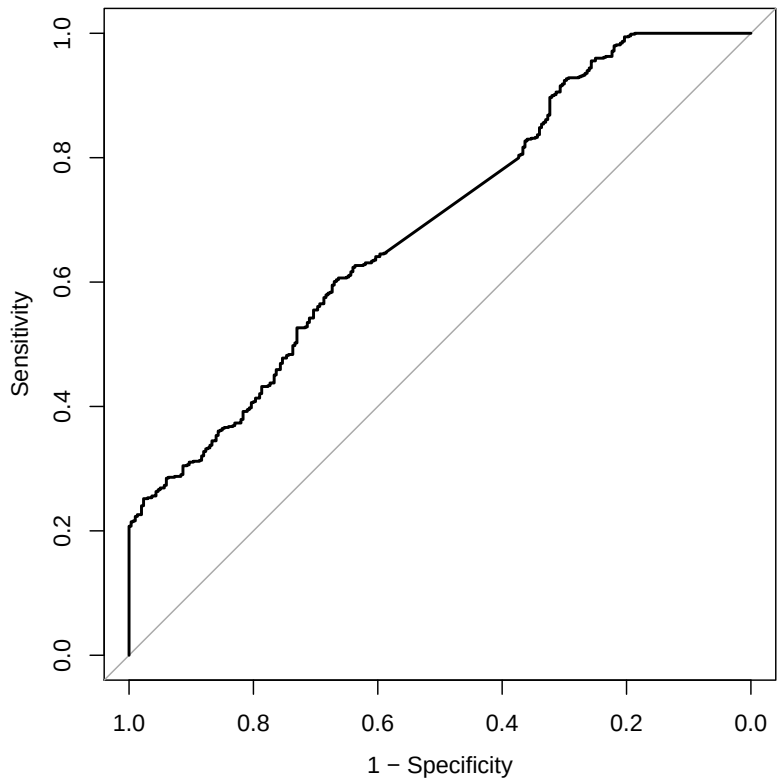
- Perform prediction and retrieve decision values `dv`

```
pred <- predict(model, GermanCredit[-inTrain,],
                decision.values=TRUE)
dv <- attributes(pred)$decision.values
```

ROC in R

- Plot ROC curve via `plot.roc(classes, dv)`

```
plot.roc(as.numeric(GermanCredit$Class[-inTrain]), dv, xlab = "1 - Specificity")
```




```
##
## Call:
## plot.roc.default(x = as.numeric(GermanCredit$Class[-inTrain]),     predictor = dv, xlab =
##
## Data: dv in 300 controls (as.numeric(GermanCredit$Class[-inTrain]) 1) > 699 cases (as.nu
## Area under the curve: 0.692
```

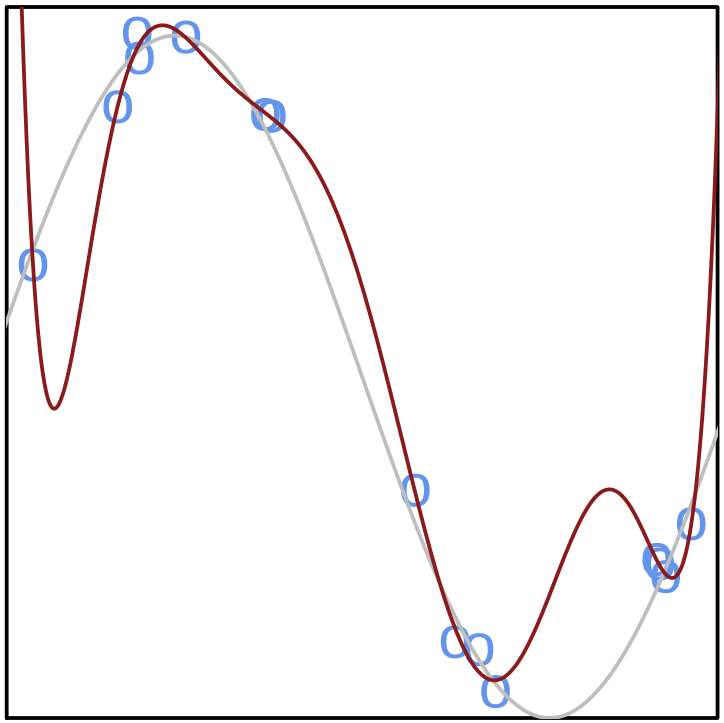
Predictive vs. Explanatory Power

Significant difference between predicting and explaining:

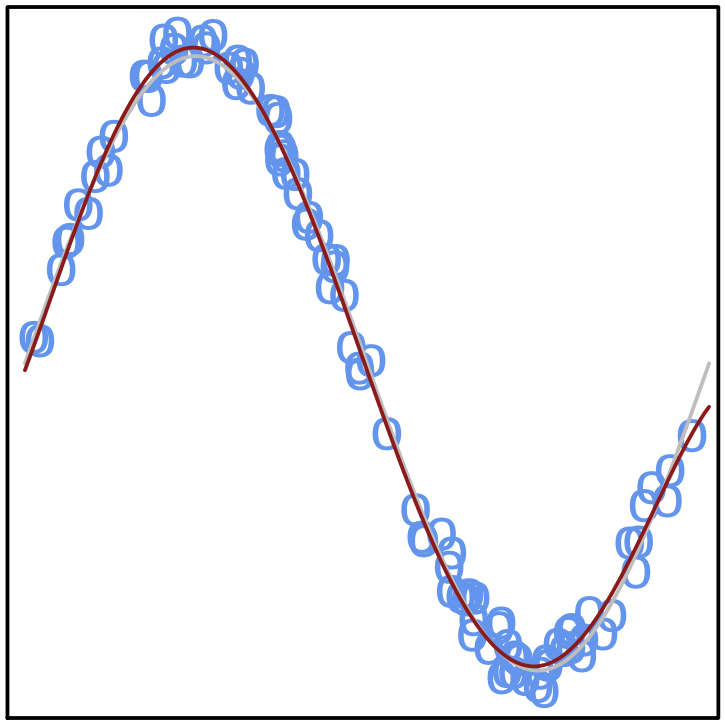
1. **Empirical Models for Prediction**
   - Empirical predictive models (e. g. statistical models, methods from data mining) designed to predict new/future observations
   - Predictive Analytics describes the evaluation of the predictive power, such as accuracy or precision
2. **Empirical Models for Explanation**
   - Any type of statistical model used for testing causal hypothesis
   - Use methods for evaluating the explanatory power, such as statistical tests or measures like R^2

Predictive vs. Explanatory Power

$N = 15$



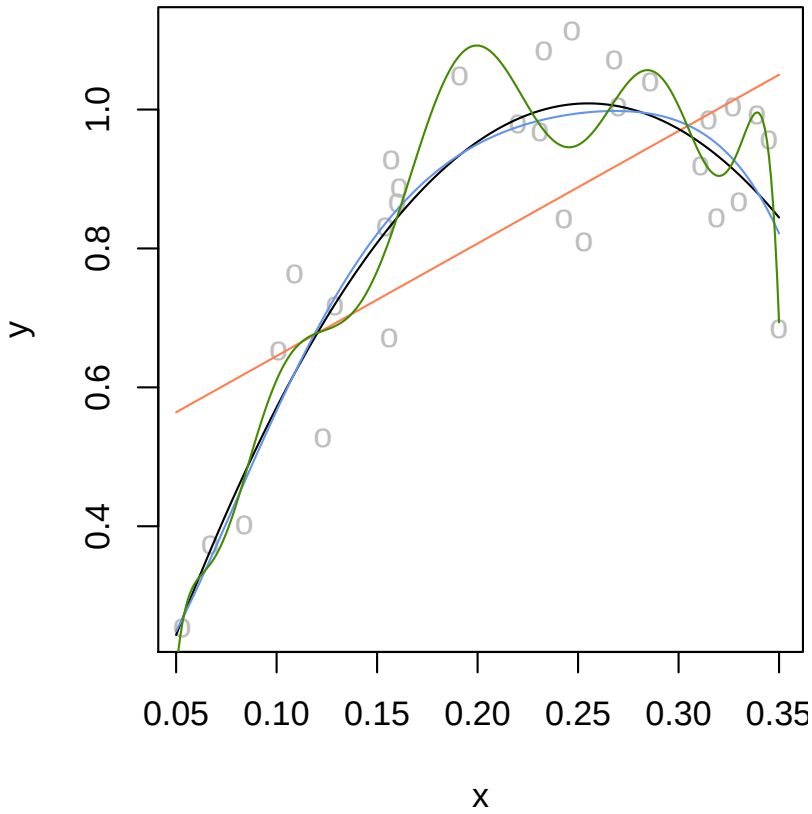
$N = 100$

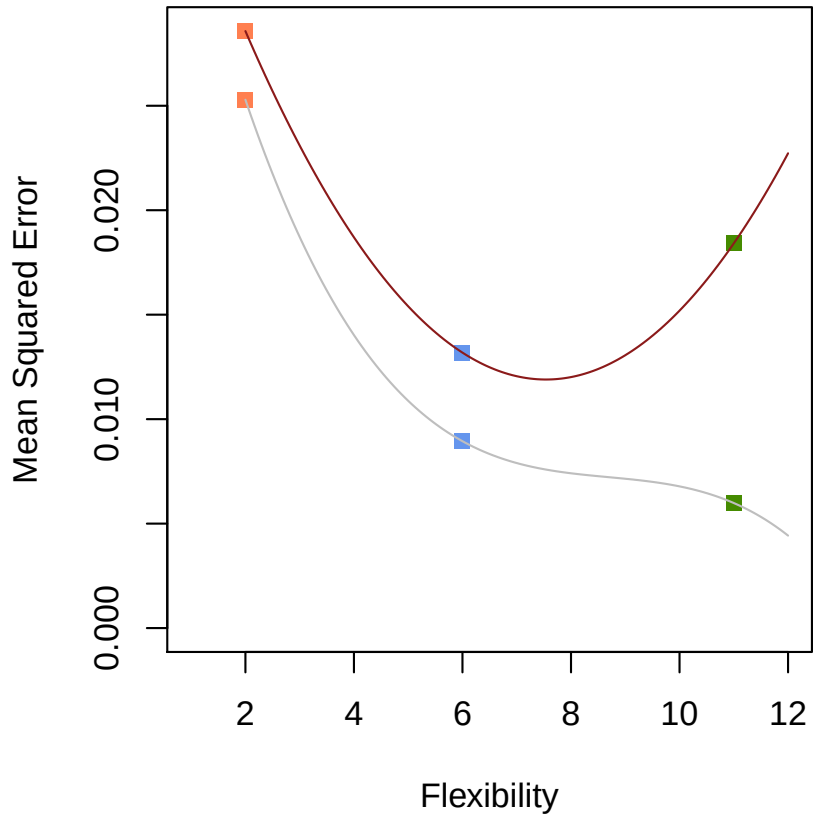


- Explanatory power does not imply predictive power
  - Red is the best explanatory model; gray the best predictive
  - In particular, dummies do not translate well to predictive modes
- Do not write something like "the regression proves the predictive power of regressor x_i"

Overfitting

- When learning algorithm is performed for too long, the learner may adjust to very specific random features not related to the target function
- Overfitting: Performance on training data (in gray) still increases, while the performance on unseen data (in red) becomes worse

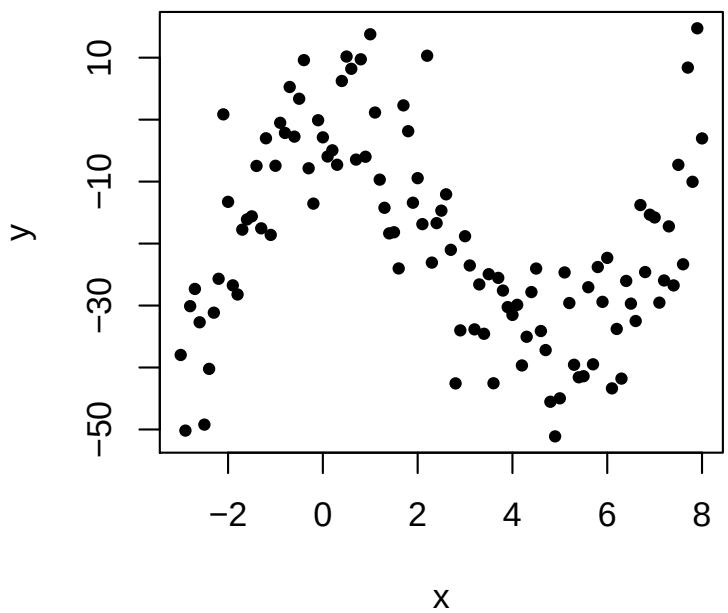




Overfitting in R

Given data `x` to predict `y`

```
plot(x, y, pch = 20)
```



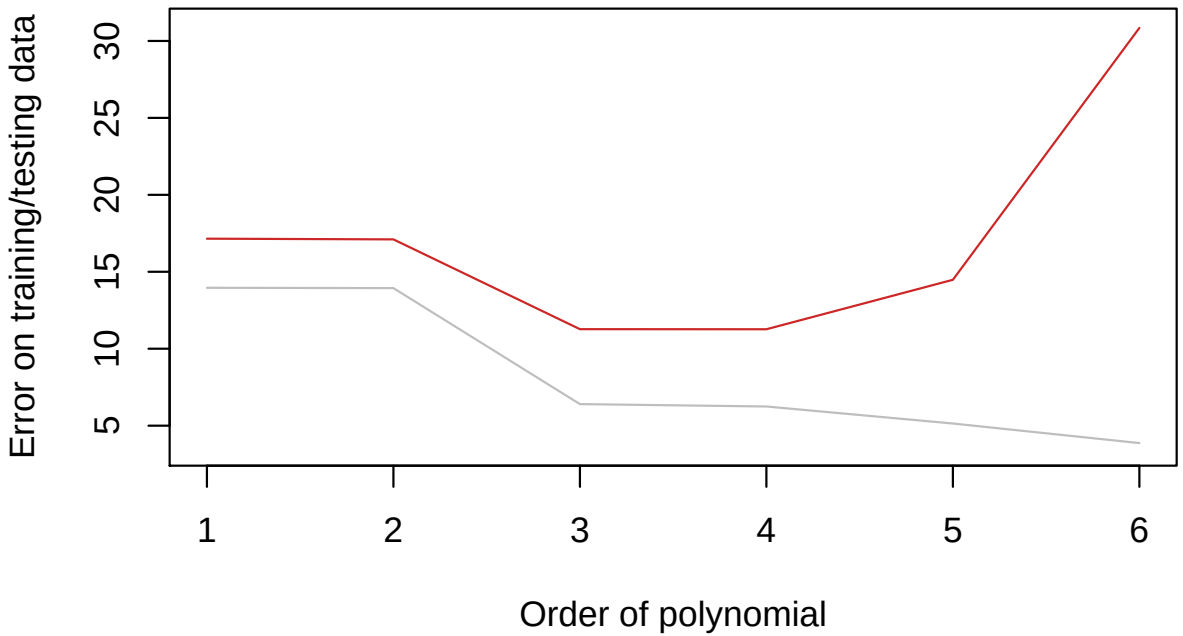
Estimate polynomials of order $P = 1, \ldots, 6$ and compare errors on training and testing data

```
inTrain <- runif(length(x)) < 0.2

err.fit <- c()
err.pred <- c()

for (i in 1:6)
{
  # Explanatory model
  xx <- x[inTrain]
  m <- lm(y[inTrain] ~
          poly(xx, i, raw=TRUE))
  err.fit[i] <- sqrt(mean(m$residuals^2))

  # Predictive model
  xx <- x[-inTrain]
  err <- predict(m,
           poly(xx, i, raw=TRUE))-y[-inTrain]
  err.pred[i] <- sqrt(mean(err^2))
}
```

Overfitting in R

- Performance on training data (in gray) still increases, while the performance on unseen data (in red) becomes worse

```
plot(1:6, err.fit, type='l', ylim=c(3.5, 31), col="gray",
     xlab="Order of polynomial", ylab="Error on training/testing data")
lines(1:6, err.pred, col="firebrick3")
```



Support Vector Regression in R

- Support vector regression (SVR) predicts continuous values
- Accessing weekly stock market returns for 21 years

```
library(ISLR)
data(Weekly)
inTrain <- runif(nrow(Weekly)) < 0.2
```

- When training, use `type="eps-regression"` instead

```
model <- svm(Today ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5,
             data=Weekly[inTrain,], type="eps-regression")
```

- Compute root mean square error (RMSE)

```
pred <- predict(model, Weekly[-inTrain, ])
sqrt(mean(pred - Weekly$Today[-inTrain])^2)

## [1] 0.1278
```

given by $\text{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\text{pred}_i - \text{true}_i)^2}$ and compare to standard deviation of $\sigma^2 = 2.3536$ (N in denominator, not $N-1$)

Outline

1 Recap

2 Linear Discriminants

3 Artificial Neural Networks

4 Support Vector Machines

5 Prediction Performance

6 Wrap-Up

Summary: Data Mining with Linear Discriminants

Linearly separable	Data can be perfectly classified by linear discriminant
Artificial neural network	Feeding information through the network → `nnet(formula, data=d, size=n ...)`
Support vector machine	Maximize separating margin → `svm(formula, data=d, type=...)`
Confusion matrix	Tabular outline of correct/incorrect guesses
Predictive performance	Measured by accuracy, precision, ...
ROC curve	Visualize trade-off between sensitivity and specificity
Overfitting	Performance on training data increases, different from unseen observations