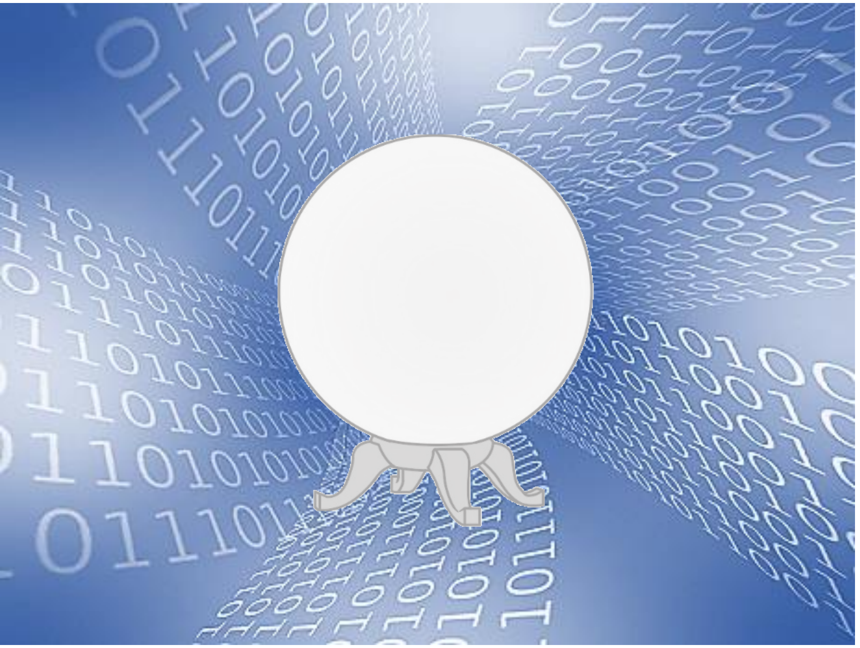
Prof. Dr. Bastian Amberg
School of Business & Economics
Department of Information Systems



Business Intelligence

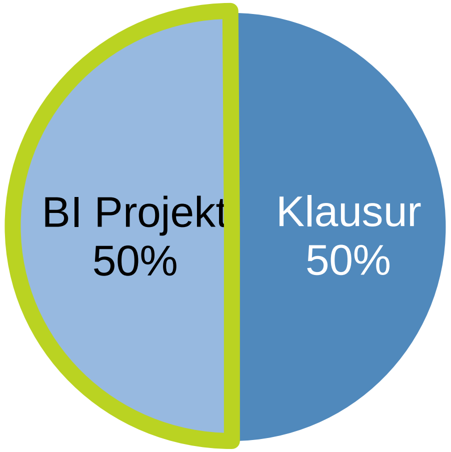
11 Fitting a Model

Prof. Dr. Bastian Amberg
(summer term 2024)
21.6.2024

Organisatorisches - Prüfungstermine und Prüfungsleistung

Freie Universität Berlin

Seit Ende Mai sind die Prüfungspläne vom Prüfungsbüro veröffentlicht und
seit dem 7.6. können Sie die BI-Projekte bearbeiten.



1/3: Präsentation

vorherige Abgabe via Blackboard (**16.7.‘**24 **bis 23.59 Uhr**)

am **17.7.‘**24 **und 19.7.‘**24, 20 Minuten Präsentation je Gruppe

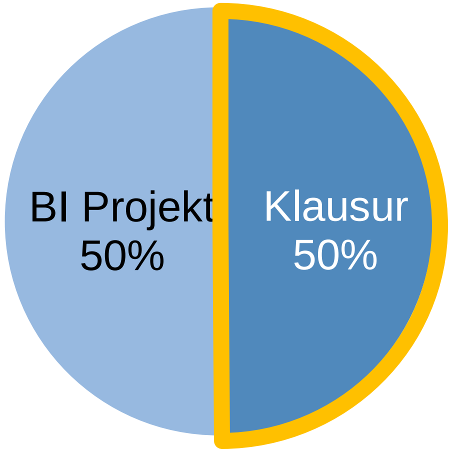
2/3: Kurzdokumentation

Gliederung gemäß CRISP-DM,

max. 12 Seiten (3 pro Person)

Abgabe via Blackboard (spätestens am **14.8.‘**24 (**bis 23.59 Uhr**).

Am 3.7. Hinweise zu den Abschlusspräsentationen inkl. Reihenfolge der Präsentationen (gemeinsame Auslosung via Python)



1x Klausur

1. Termin **31.7.‘**24**, 10.15 Uhr,** HS 108, 60 Minuten

oder **2. Termin** **2.10.‘**24**, 10.15 Uhr,** HS 108, 60 Minuten

Ab Mittwoch, 26.6., Wie könnten mögliche Klausuraufgaben aussehen? (Beispielaufgaben in Blackboard)

Prüfungszeitraum Termin 1: 22.07. - 03.08.2024
Prüfungszeitraum Termin 2: 23.09. - 05.10.2024

(https://www.wiwiss.fu-berlin.de/studium-lehre/pruefungsbuero/news/Pruefungsplaene-SoSe-2024.html)

Ref.

Hinweise

Freie Universität Berlin

Wie könnten mögliche Klausuraufgaben aussehen?

40 days left until the exam (Termin 1)

103 days left until the exam (Termin 2)

6 Woche

Zu den Inhalten:

- Die Inhalte aus den Python-Selbstlerneinheiten sowie aus den beiden Python-Übungen fließen nicht in die Klausur ein. Kein Python.
- Inhaltliche Schwerpunktsetzung beachten. **Aufteilung: DW/DE** ca. 10-15 Punkte, **DM/DS** ca. 45-50 Punkte.
- Für den Teil **DW/DE** haben Sie im Rahmen der Bonusaufgaben mit *Co-Create your exam* einen Fragenpool formuliert. *(BI_2024_DW-DE_Aufgabenpool.pdf ab dem 26.6. unter Kursmaterial/Übungen -- deckt vollständig diesen Teilbereich der Klausur ab)*
- Beispiele für Fragestellungen zum Teil ***DM/DS*** ergeben sich aus den Übungsaufgaben während der Vorlesung („Exercises“ beachten), bzw. aus dem in Blackboard bereit gestellten Zusatzmaterial ausgewählter Aufgaben vergangener Jahre *(BI_2024_DM-DS_Probeaufgaben.pdf ab dem 26.6. unter Kursmaterial/Übungen)*
- Insbesondere sollen die Probeaufgaben ein Gefühl dafür vermitteln, wie die Klausurfragen vermutlich formuliert sind. *(Die Vorlesungsinhalte zum Teil **DM/DS** bereiten Sie bereits aktuell durch die Bearbeitung der Projektaufgabe nach)*

Entropy ; IG

Linear Classifier

Erwartungen:

- Die gemeinsam bzw. im Selbststudium „geübten“ Methoden und Berechnungen (z.B. zu Decision Trees, in Kürze auch zu Expected Value Framework, Naive Bayes, k-Nearest Neighbors, ...) können selbstständig durchgeführt werden.
- Sollten darüber hinaus bei der kritischen Beurteilung bzw. Entscheidungsfindung Formeln zwingend notwendig sein, die wir nur überblicksartig behandelt haben (z.B. Korrelationsmaße), werden diese in der Aufgabenstellung mit angegeben.
- In jedem Fall sollten Sie in der Lage sein, Maße, Modelle und Methoden gegenüberzustellen, um sie in konkreten Fällen begründbar auszuwählen.

13.7.

(In der letzten Veranstaltung am 16.7. folgen noch ergänzende Informationen)

Ref.

Schedule

Freie Universität Berlin

		Wed., 10:00-12:00		Fr., 14:00-16:00 (Start at 14:30)		Self-study		
	W1	17.4.	(Meta-)Introduction	19.4.		Python-Basics	Chap. 1	
	W2	24.4.	Data Warehouse – Overview & OLAP	26.4.	*[Blockveranstaltung SE Prof. Gersch]*		Chap. 2	
Basics	W3	1.5.		3.5.			Chap. 3	
	W4	8.5.	Data Warehouse Modeling I & II	10.5.	Data Mining Introduction			
	W5	15.5.	CRISP-DM, Project understanding	17.5.	Python-Basics-Online Exercise	Python-Analytics	Chap. 1	
	W6	22.5.	Data Understanding, Data Visualization I	24.5.	*No lectures, but bonus tasks 1.) Co-Create your exam 2.) Earn bonus points for the exam*		Chap. 2	
	W7	29.5.	Data Visualization II	31.5.				
Main Part	W8	5.6.	Data Preparation	7.6.	Predictive Modeling I (10:00 -12:00)	BI-Project	Start	
	W9	12.6.	Predictive Modeling II	14.6.	Python-Analytics-Online Exercise		|	
	W10	19.6.	*Guest Lecture Dr. Ionescu*	21.6.	Fitting a Model		|	Case Study
	W11	26.6.	How to avoid overfitting	28.6.	What is a good Model?		|	
	W12	3.7.	Project status update Evidence and Probabilities	5.7.	Similarity (and Clusters) From Machine to Deep Learning I		|	
Deep-ening	W13	10.7.		12.7.	From Machine to Deep Learning II		|	
	W14	17.7.	Project presentation	19.7.	Project presentation		End	
					Klausur 1.Termin, 31.7.'24 Klausur 2.Termin, 2.10.'24	Projektbericht		

Ref.

Recap Python Exercise: Data mining and its use

Discerning model building and model deployment

150 samples

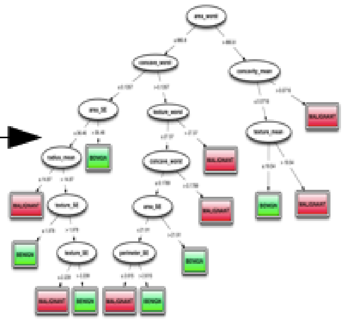
Historical data

Training data have all values specified

X	Y	Z	class
14	True	Red	accepted
6	True	Blue	rejected
...			
50.3	False	Red	accepted

Data mining

Model



Model is deployed

Mining

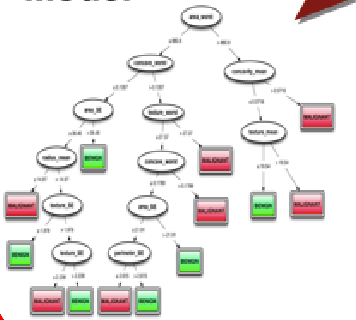
Use

New data item

X	Y	Z	class
30	False	Red	?

New data item has class value **unknown** (e.g., *will customer accept?*)

Model

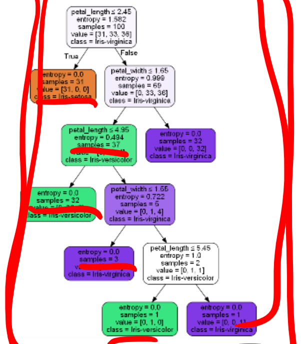


Class: accepted, Probability: 0.88

See Python-Analytics Exercise from Friday 14.6.

1,00

Build model using 100 samples (training set)



Validation

Test model using 50 samples (test set)

Overfitting!

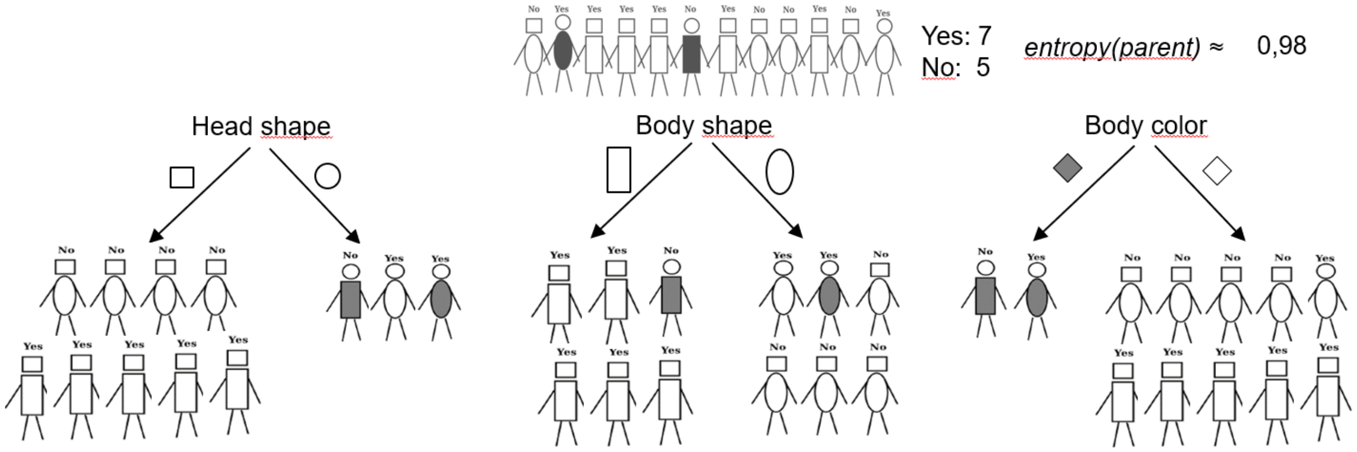
nächste Woche

Ref.

Last Lesson

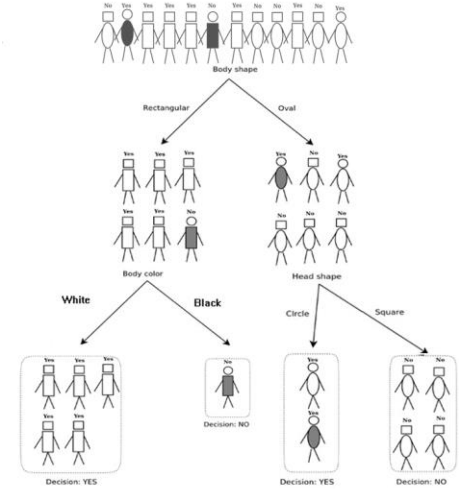
Predictive Modeling   Decision Trees

Which attribute to choose?



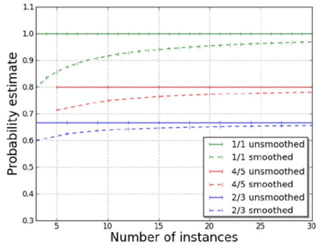
Recursively apply **attribute selection** to find the best attribute to partition the data set

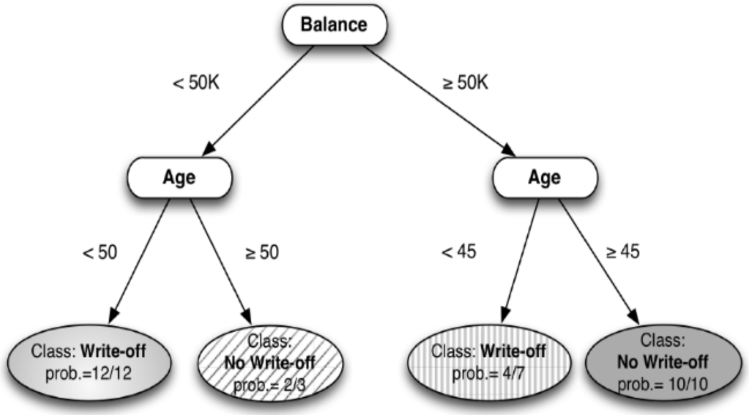
The goal at each step is to select an attribute to partition the current group into subgroups that are as pure as possible w.r.t. the target variable



- Decision Trees model non-linear relationships between attributes
- Different decision tree algorithms generate decision trees with different structure
- Tree induction can easily produce **probability estimation trees** instead of simple classification trees

  Smoothed version of frequency-based estimate by **Laplace correction**, which moderates the influence of leaves with only a few instances





Ref.

Agenda

Freie Universität Berlin

Classification via Mathematical Functions
Fitting a Model to Data

(1) Linear Classifiers

(2) Linear Regression

(3) Logistic Regression

(4) Tree-induction vs. logistic regression

Ref.

Introduction

$f(x) = w_0 + w_1 x_1 + w_2 x_2 + \ldots + w_n x_n$

$10 + 10\,Age - 7.5\,Balance$

$w_0 + w_1 Age + w_2 \cdot Balance$

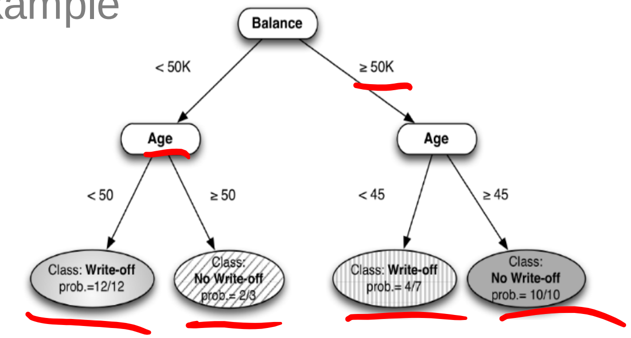
Freie Universität Berlin

Classification via Tree Induction

So far:

✓ we produced both the **structure of the model** (the particular tree model) and the **numeric parameters** of the model from the data

For example



Questions answered:

- ✓ How do we decide to classify data?
- ✓ Why do we not build „complete“ trees?
- ✓ If we have incomplete trees, we want to assess probabilities. What do we take into consideration?

Ref.

Classification via Mathematical Functions

⇒ Deep learning

Now:

- ➢ We **specify the structure of the model**, but leave certain numeric parameters unspecified
- ➢ Data Mining calculates the best parameter values given a particular set of training data
- ➢ The form of the model and the attributes is specified
- ➢ The goal of DM is to tune the parameters so that the model fits the data as good as possible (**parameter learning**)

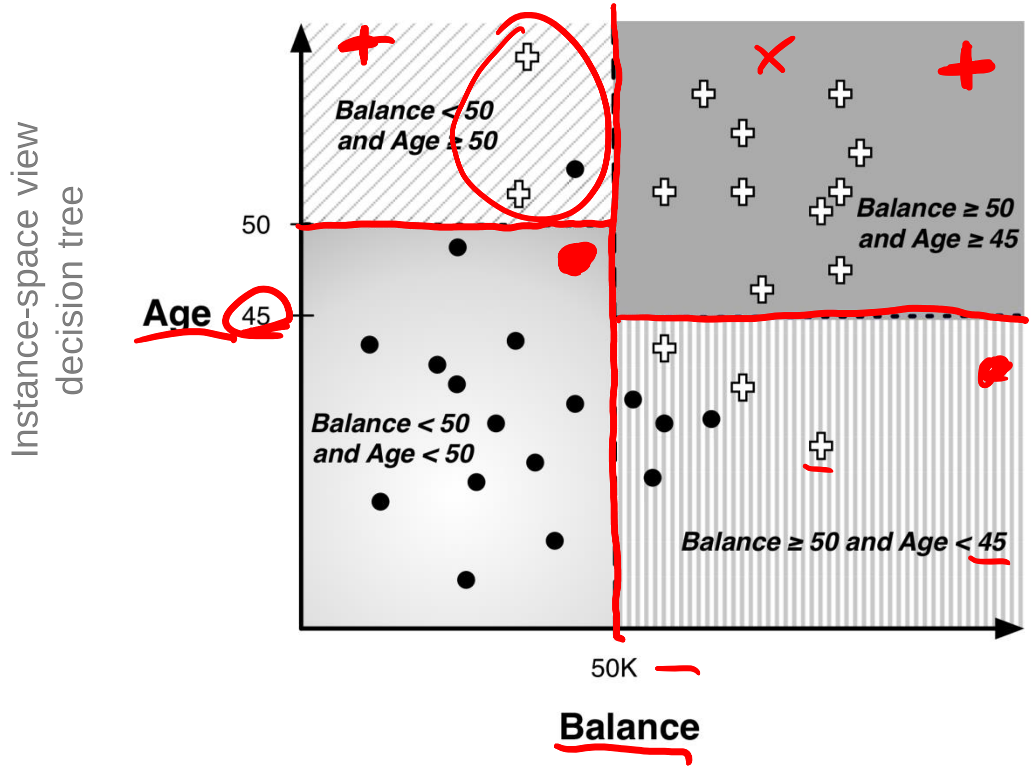
Simplifying assumptions:

- For classification and class probability estimation, we will consider **only binary classes**.
- We assume that **all attributes are numeric.** (→ see data preparation)
- We **ignore the need to normalize numeric** measurements to a common scale (→ see data preparation)

Linear classifiers

Instance-space view:
shows the space broken up into regions by decision boundaries

- Examples in each space should have similar values for the target variable
- Homogeneous regions help predicting the target variable of a new, unseen instance



$f(x) = w_0 + w_1 x_1 + \dots w_n x_n$

We can separate the instance almost perfectly (by class) if we are allowed to introduce a boundary that is still a straight line, but is not perpendicular to the axes

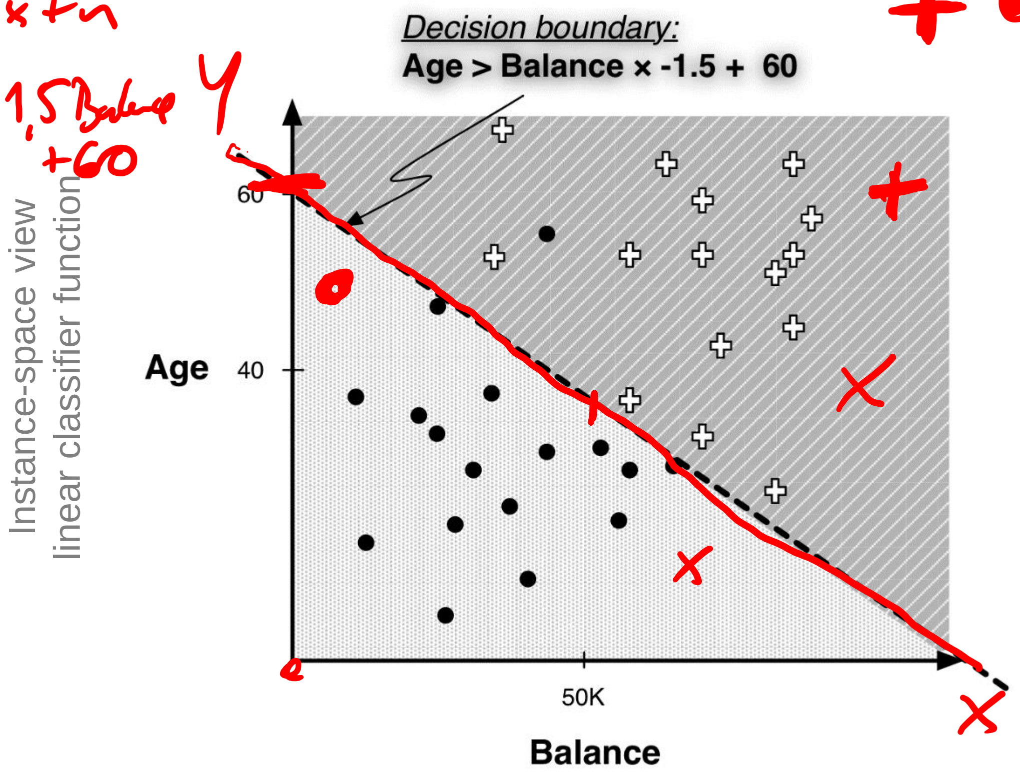
- Linear classifier

Age Balance

$y = mx + n$

Age = -1,5 Balance + 60

Y



Ref.

Linear discriminant functions (1/2)

Classifying with a linear function (*parametric learning*)

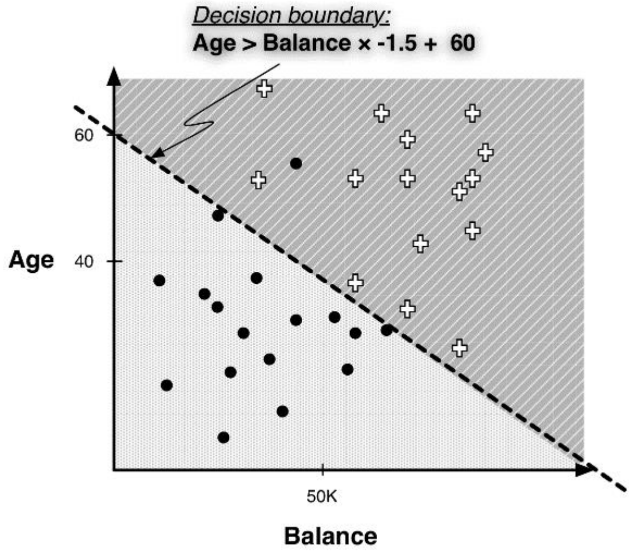
Equation of a line: $y = mx + b$
with m being the slope and b the y intercept

Line in figure:

$$Age = (-1.5) * Balance + 60$$

$\Leftrightarrow$

$- 1{,}0\ Age$
$- 1{,}5\ Balance$
$+ 60$



We would classify an instance x as a "+" if it is above the line, and as a "•" if it is below the line.

Ref.

Mathematically:

Age | Balance
20 | 10.000

$$class = \begin{cases} + & \text{if } -1.0 * Age - 1.5 * Balance + 60 > 0 \\ \bullet & \text{if } -1.0 * Age - 1.5 * Balance + 60 \leq 0 \end{cases}$$

Linear discriminant discriminates between the Classes

Supervised segmentation by creating a mathematical function of multiple attributes

A **linear discriminant function** is a numeric classification model, which can be written as

$$f(\mathbf{x}) = w_0 + w_1 x_1 + w_2 x_2 + \cdots$$

$w_0, w_1 \ldots w_n$ are parameters to be estimated

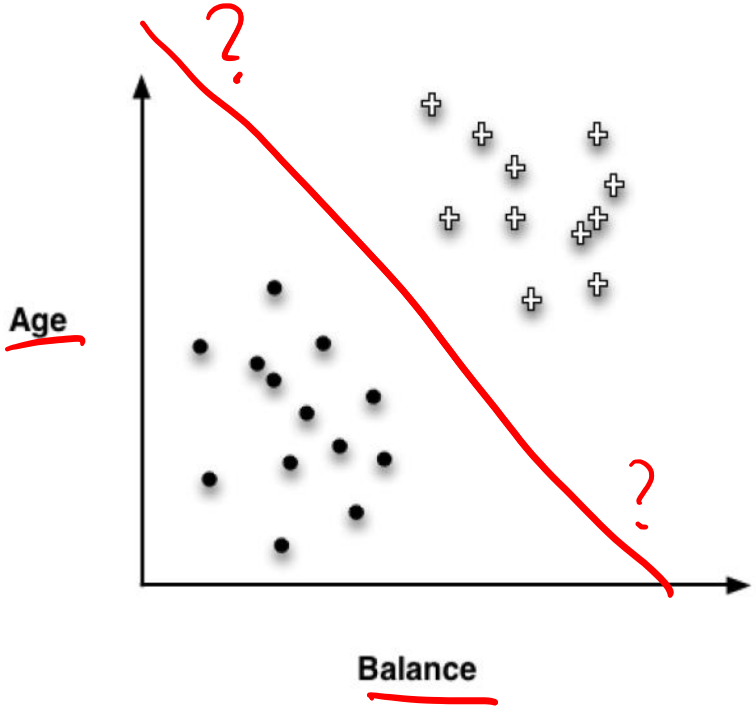
Linear discriminant functions (2/2)

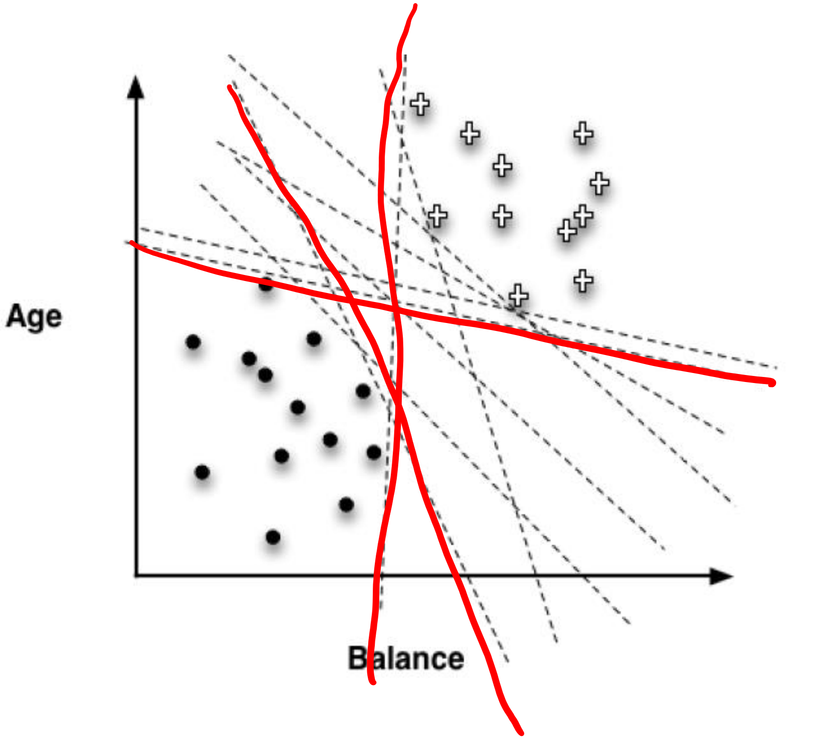
$f(x) = w_0 + w_1 x_1 + \ldots$

What is the „best“ line to separate the classes?

Fit parameters w_i to a particular data set

- Find a good set of weights (using a given set of features)
- Weights may be interpreted as importance indicators
- The larger the magnitude of a weight, the more important





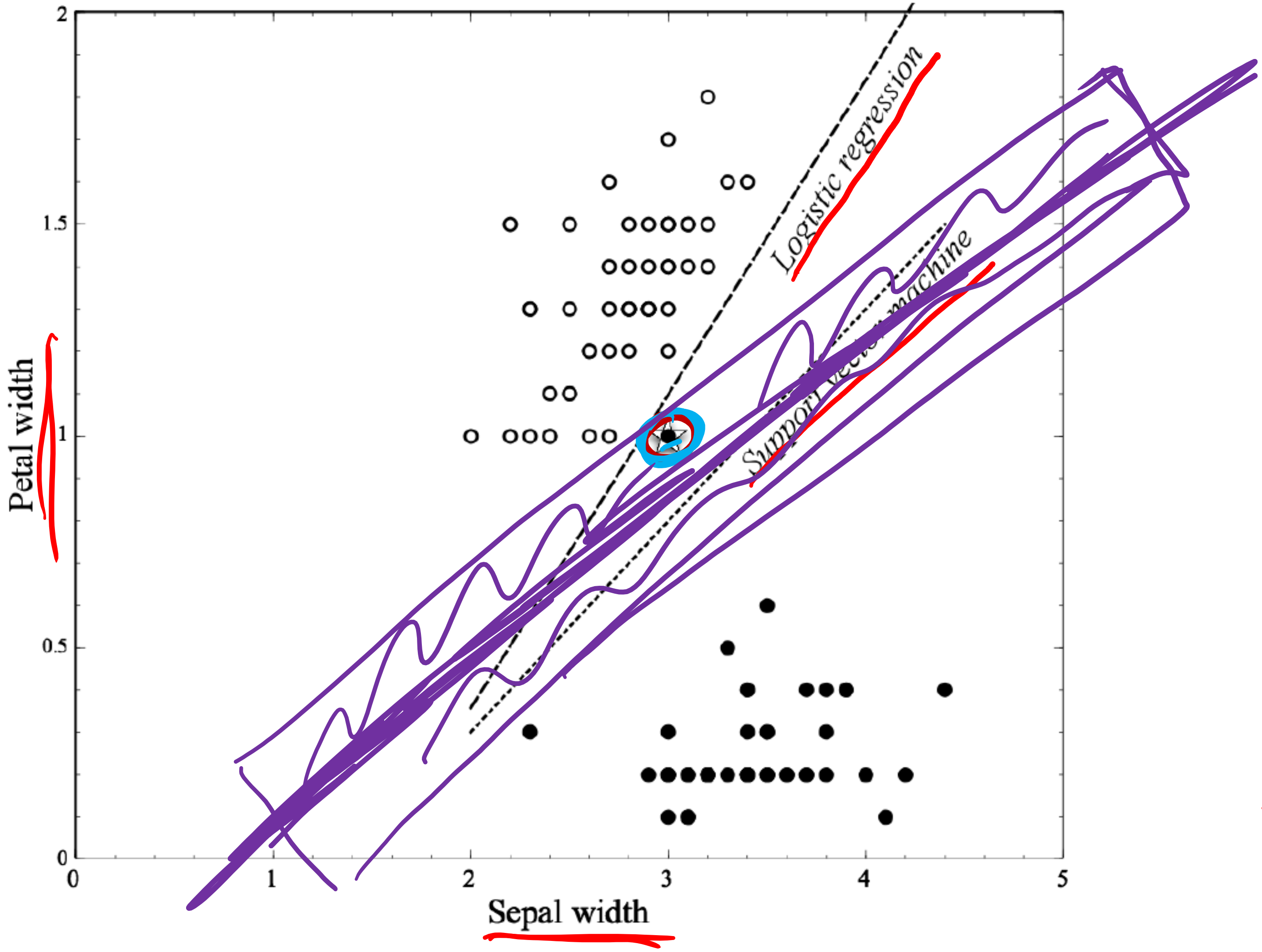
➡ Optimizing an objective function

What should be our objective in choosing the parameters?
(Which weights should we choose?)
We need to define an **objective function** that represents our goal sufficiently
(Optimal solution is found by minimizing or maximizing)

We will consider
- Support Vector Machines
- Linear regression
- Logistic regression

Ref.

Mining a linear discriminant for the Iris data set



Ref.

Linear discriminant functions

Scoring and ranking instances

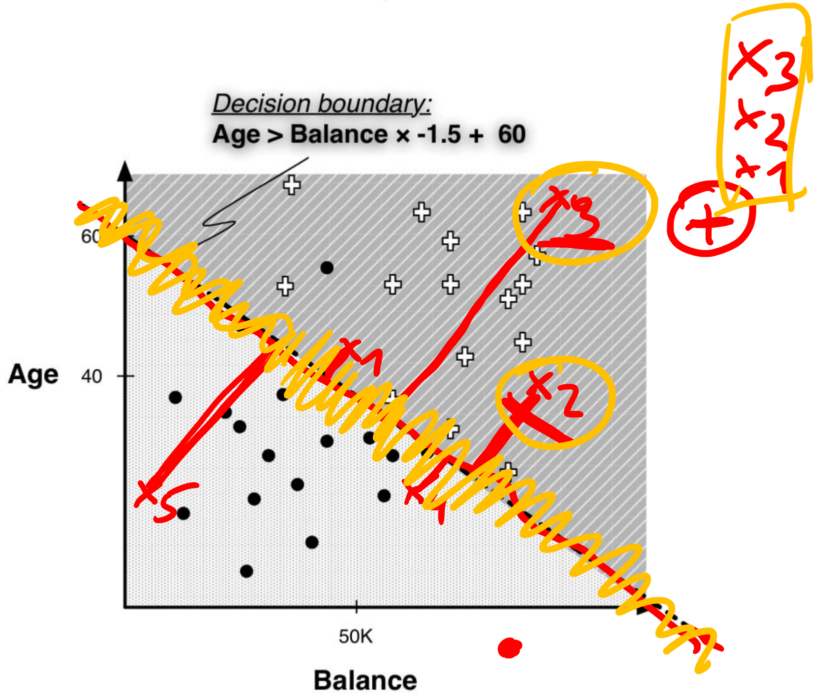
Sometimes, we want some notion of which examples **more or less likely** to belong to a class

- Which customers are most likely to respond to this offer?
- Remember class membership probability

Sometimes, we don‘t need a precise probability estimate – a **ranking** is sufficient

- Linear discriminant functions provide rankings
- $f(x)$ will be small when x is near the boundary
- $f(x)$ gives an intuitively satisfying ranking of the instances by their (estimated) likelihood of belonging to the class of interest

x_1, x_2, x_3, x_4, x_5



Ref.

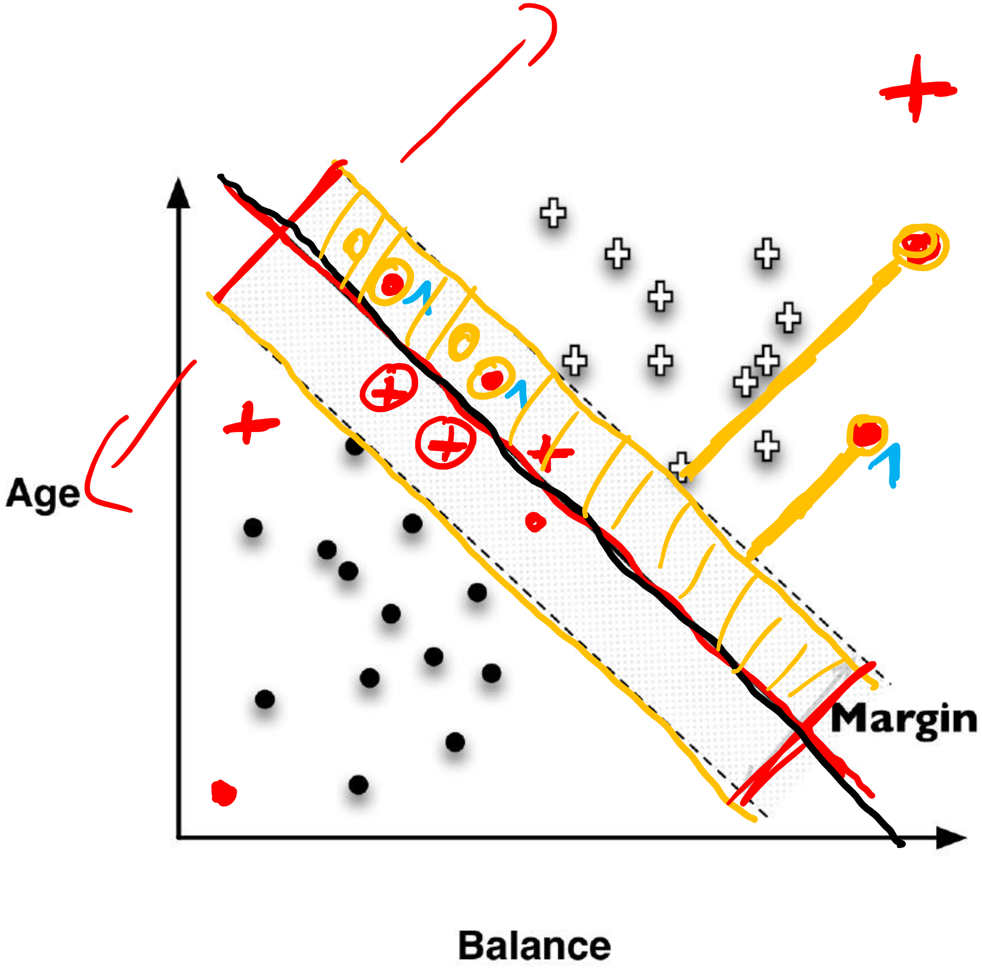
Support Vector Machines

An intuitive approach

$f(x) = w_0 + w_1 x_1 + \dots$

Support Vector Machines (SVM) are linear discriminants

- Classify instances based on a linear function of the features
- Objective function based on a simple idea: **maximize the margin**
  - Fit the broadest bar between the classes
  - Once the widest bar is found, the linear discriminant will be the center line through the bar
- The margin-maximizing boundary gives **the maximal leeway** for classifying new points



Ref.

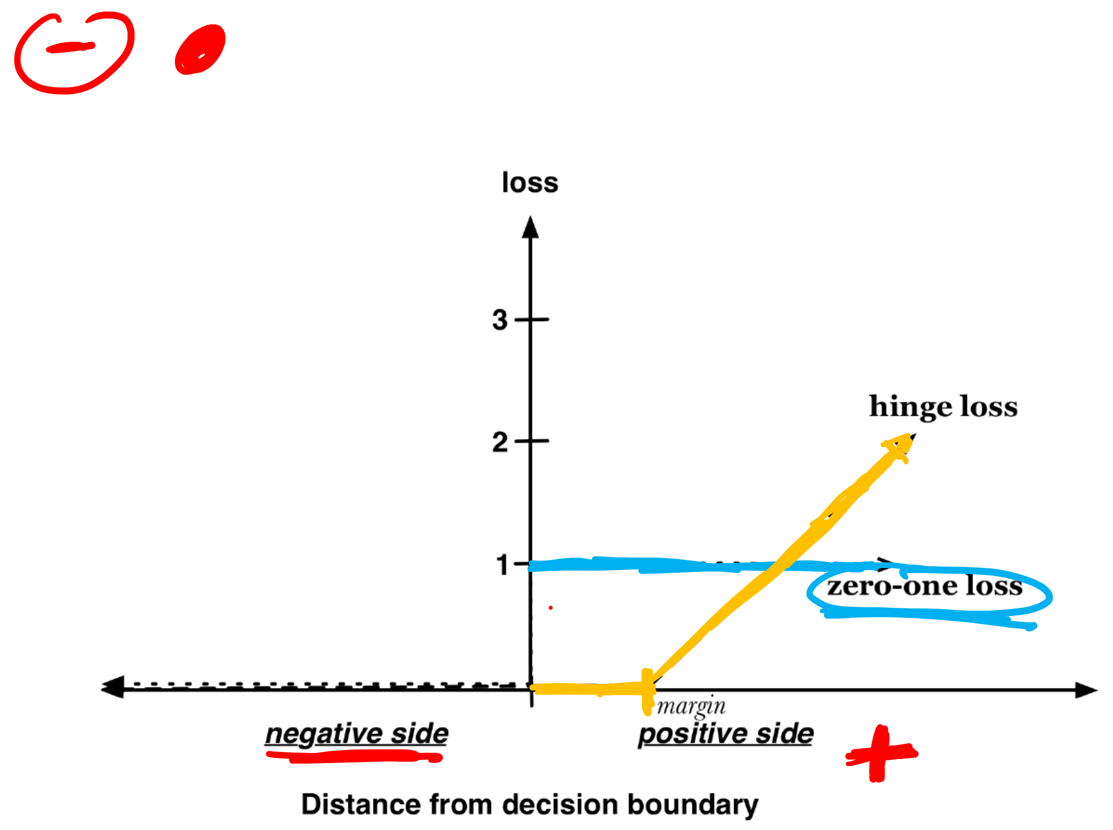
Support Vector Machines

An intuitive approach in finding a perfect separating function

How to handle data points that are misclassified by the model, i.e., if there is no perfect separating line?

In the objective function, a training point is **penalized** for being on the wrong side of the decision boundary

- If the data are linearly separable, no penalty is incurred and the margin is simply maximized
- If the data are not linearly separable, the best fit is **some balance between a broad margin and a low total error penalty**
- The penalty is proportional to the distance from the decision boundary (hinge loss)



Ref.

Support Vector Machines

Modeling non-linear relationships with the kernel trick

PCA

Excursus

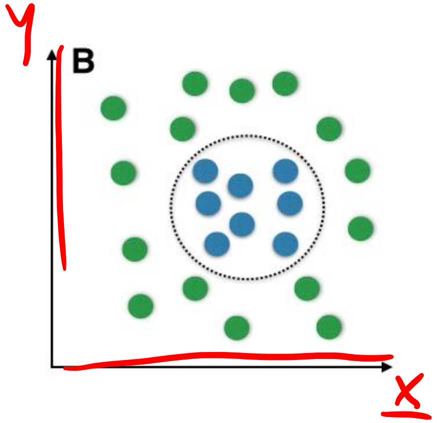
$f(x)=w_0+w_y y+w_x x$

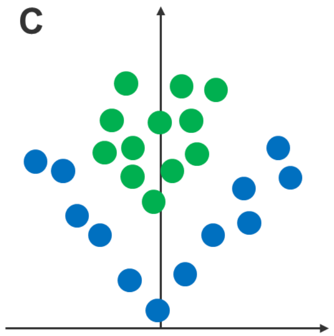
x, y, x^2+y^2

$f(x)=w_0+w_y y+w_x x + w_{xy}(x^2+y^2)$

We can model non-linear relationships in SVM using the **kernel trick**

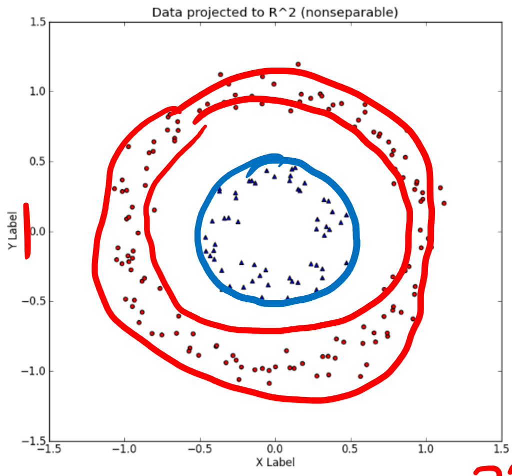
We increase the number of dimensions, to discriminate the classes.

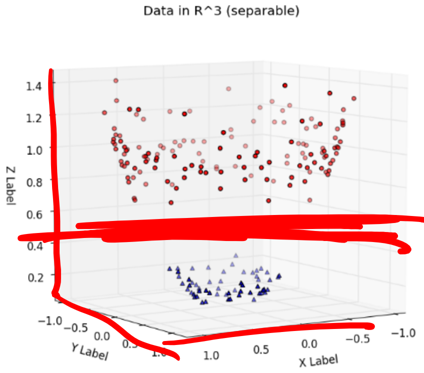




We separate the classes by adding a feature using existing information:

$$z = x^2 + y^2 \qquad\qquad z = |x|$$





2D -> 3D

You can fine-tune SVM by choosing

kernels (automatically optimized), like radial basis functional kernel (RBF) or polynomial kernels

and regularization parameter c, which tells the optimizer how broad the margin should be (smoothness vs. correct classification)

Ref.

Image: Eric Kim (2013)

Agenda

Freie Universität Berlin

Classification via Mathematical Functions
Fitting a Model to Data

(1) Linear Classifiers

(2) Linear Regression

Zahl

(3) Logistic Regression

Klassen-wahrscheinlichkeit

(4) Tree-induction vs. logistic regression

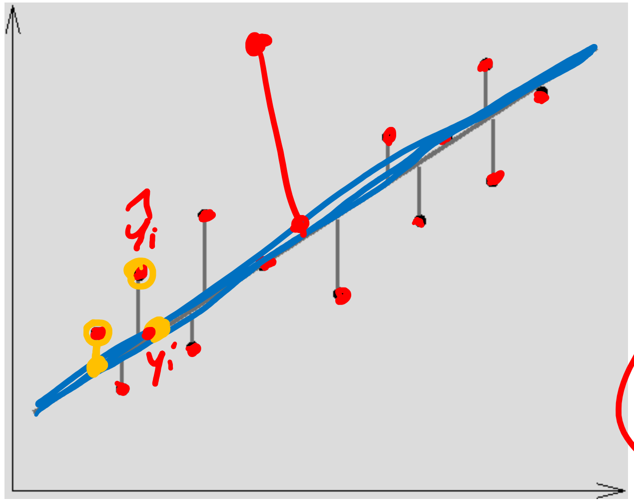
Ref.

Linear regression

$$f(\mathbf{x}) = w_0 + w_1 x_1 + w_2 x_2 + \cdots$$

Remember

- Which **objective function** should we use to optimize a model's fit to the data?
- Most common choice: how far away are the estimated values from the true values of the training data?
- **Minimize the error of the fitted model**, i.e., minimize the distance between estimated values and true values!



Min

e.g., ~~sum~~ of errors

$$\sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

$|y_i - \hat{y}_i|$

- **Regression procedures** choose the model that fits the data best w.r.t. the sum of errors
  - Sum of **absolute** errors
  - Sum of **squared** errors
- Standard linear regression is convenient (mathematically)!

A linear regression minimizes the squared error

Squared error strongly penalizes large errors

Squared error is very sensitive to the data

- erroneous or outlying data points can severely skew the resulting linear function
- For systems that build and apply models automatically, modeling needs to be much more robust

Choose the objective function to optimize with the ultimate business application in mind

Ref.

Agenda

Freie Universität Berlin

Classification via Mathematical Functions
Fitting a Model to Data

(1) Linear Classifiers

(2) Linear Regression

(3) Logistic Regression

Klassen-
wahrscheinlichkeit

(4) Tree-induction vs. logistic regression

Ref.

Logistic regression (1/4)

Estimation of the Probability

$[0 \dots 1]$

For many applications, we would like to **estimate the probability** that a new instance belongs to the class of interest

- *Fraud detection: where is the company's monetary loss expected to be the highest?*
- *Spam: What is the probability of the new mail being spam?*

Select different objective function to give accurate estimates of class probability

- Well calibrated and discriminative

$f(x) = w_0 + w_1 x_1 + w_2 x_2 + \dots$

$[-\infty \dots \infty]$

Recall:

An instance being further from the separating boundary leads to a higher probability of being in one class or the other, and $f(x)$ gives the distance from the separating boundary

$[0 \dots 1]$

But a probability ranges from zero to one.

Be careful: Distinguish between ***target variable*** and ***probability of class membership***!

Ref.

Logistic regression (2/4)

Likelihood and odds

$f(x) = w_0 + w_1 x_1 \ldots$

$-\infty \ldots +\infty$

Verhältnis

The likelihood of an event can be expressed by **odds**

The **odds of an event** is the ratio of the probability of the event occurring to the probability of the event not occurring

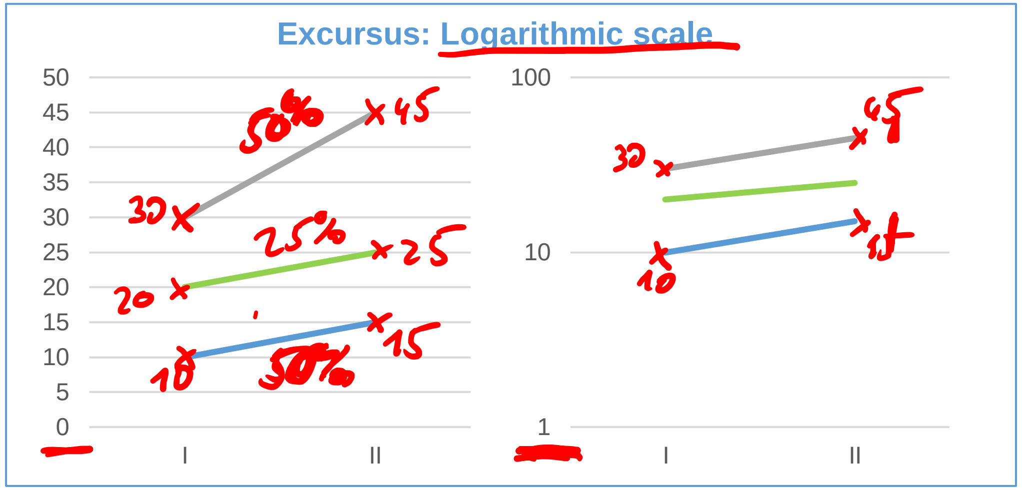
Log-odds convert the scale to $-\infty$ to $+\infty$

$[0 \ldots 1]$ $[0 \ldots \infty]$ $[-\infty \ldots \infty]$

log x, ln(x), $\log_{10}(x)$, x

Probability	Odds	Corresponding log-odds*	
0.5	50:50 or 1	0	0
0.9	90:10 or 9	2.19	0,95
0.999	999:1 or 999	6.9	3,00
0.01	1:99 or 0.0101	-4.6	-2,00
0.001	1:999 or 0.001001	-6.9	-3,00

*ln(x) is used | with $\log_{10}(x)$



Logistic regression model:

$f(x)$ is used as a measure of the log-odds of the "event" of interest

$f(x)$ is a an estimation of the log-odds that x belongs to the positive class

Ref.

Logistic regression (3/4)

How to translate log-odds into the probability of class membership?

$p_+(x) > 0{,}5$

$1 - p_+(x) = 0{,}5$

$\frac{0{,}5}{0{,}5} = 1$

$60 + 0{,}2x_1 + 0{,}4x_2$

$70 - 0{,}1x_1 + 9x_2$

Sigmoidfunktion

$\hat{p}_+(x)$ represents the model's estimate of the probability of class membership of a data item by feature vector x

$+$ is the class for the (binary) event we are modeling

$1 - \hat{p}_+(x)$ is the estimated probability of the event *not* occurring

Odd

$[-\infty \ldots \infty]$

$$ln\left(\frac{\hat{p}_+(\mathbf{x})}{1-\hat{p}_+(\mathbf{x})}\right) = f(\mathbf{x}) = w_0 + w_1x_1 + w_2x_2 + \cdots$$

$[-\infty \ldots \infty]$

Solve for $\hat{p}_+(x)$ yields $\quad \hat{p}_+(\mathbf{x}) = \frac{1}{1+e^{-f(\mathbf{x})}}$

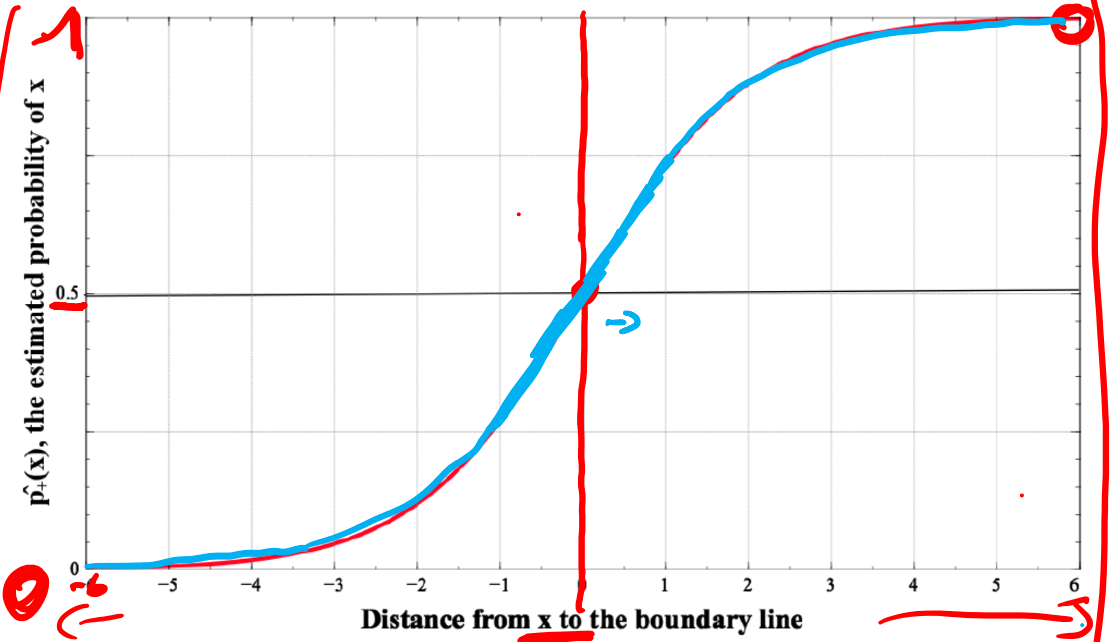
$[0..1]$

Logistische Regression

$f(x) = w_0 + w_1x_1 + \ldots w_nx_n$

Probability at distance $x = 0$ is 0.5
Probability varies approximately linearly near to the decision boundary, but then approaches certainty farther away

Determine the slope of the almost linear part within fitting



Ref.

Logistic regression (4/4)

What does the objective function look like?

Ideally, any positive example x_+ would have $\hat{p}_+(\boldsymbol{x}_+) = 1$ and any negative example $x_\bullet$ would have $\hat{p}_+(\boldsymbol{x}_\bullet) = 0$

Probabilities are never pure when real-world data is considered

Compute the likelihood of a particular labeled example given a set of parameters w that produces class probability estimates $\hat{p}_+(\boldsymbol{x})$

The g function gives the model's estimated probability of seeing x's actual class given x's features

For different parameterized models, sum the g values across all instances in a labeled (training) dataset to get "the maximum likelihood" model

The identified maximum likelihood model "on average" gives the highest probabilities to the positive examples and the lowest probabilities to the negative examples.

$$g(\mathbf{x}, \mathbf{w}) = \begin{cases} \hat{p}_+(\mathbf{x}) & \text{if } \mathbf{x} \text{ is a } + \\ 1 - \hat{p}_+(\mathbf{x}) & \text{if } \mathbf{x} \text{ is a } \bullet \end{cases}$$

$$\underset{w}{\arg\max} \quad g\,(x, w)$$

Ref.

Agenda

Classification via Mathematical Functions
Fitting a Model to Data

(1) Linear Classifiers

(2) Linear Regression

(3) Logistic Regression

(4) Tree-induction vs. logistic regression

Ref.

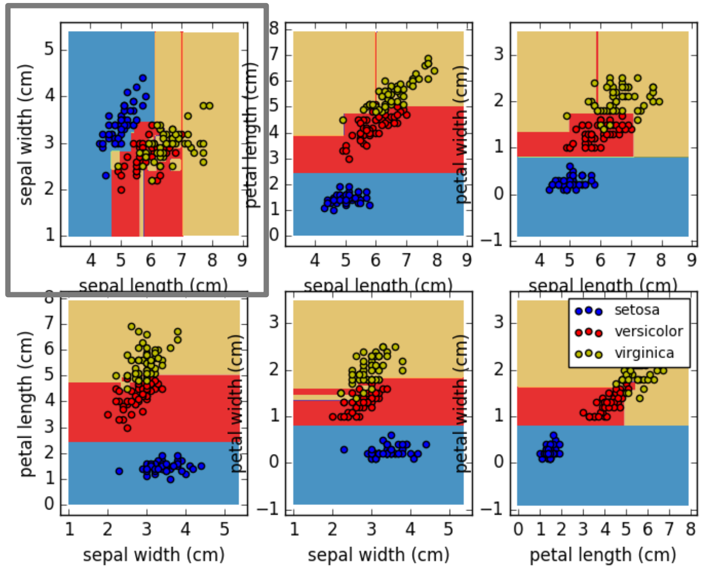
Tree induction vs. linear classifier (in general)

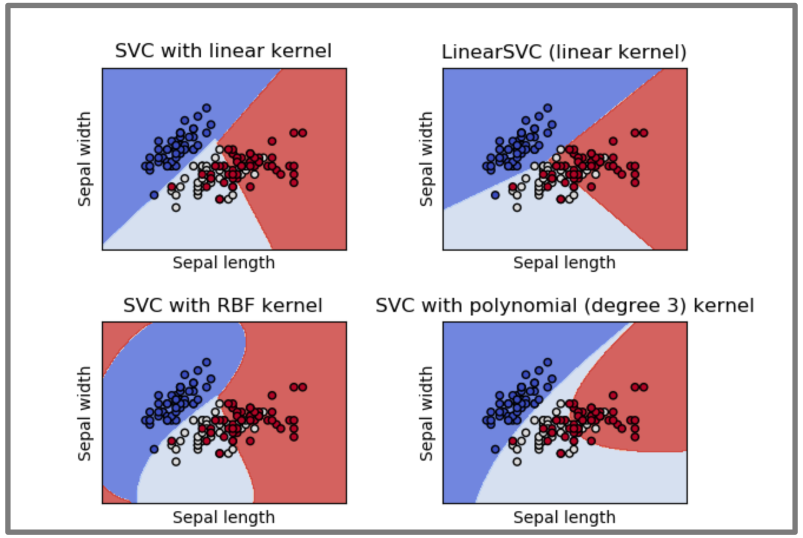
Important differences between trees and linear classifiers

- A classification tree uses decision boundaries that are **perpendicular** to the instance-space axes.
- The linear classifier can use **decision boundaries of any direction** or orientation
- A classification tree is a **"piecewise" classifier** that segments the instance space recursively → cut in arbitrarily small regions possible.
- The linear classifier places **a single decision** surface through the entire space.

Which of these characteristics are a better match to a given data set?

Decision surface of a decision tree using paired features





Ref.

https://scikit-learn.org

Tree induction vs. logistic regression

Example: Breast Cancer Dataset

Consider the background of the stakeholders

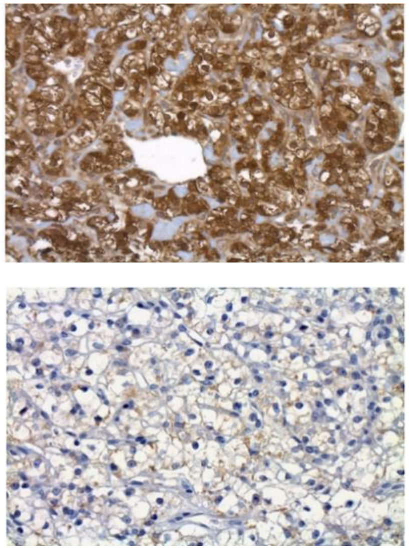
- A decision tree may be considerably **more understandable** to someone without a strong background in statistics
- Data Mining team does not have the ultimate say how models are used or implemented!

Example: Wisconsin Breast Cancer Dataset

http://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Diagnostic)

- Each record describes characteristics of a cell nuclei image, which has been labeled as either “benign” or “malignant” (**cancerous**)
- Ten fundamental characteristics were extracted and summarized in a mean (_mean), standard error (_SE) and mean of the three largest values (_worst)
  → 30 measured attributes
- 357 benign images and 212 malignant images

569



From Mu et al. (2011) doi:10.1038/ncomms1332

Ref.

569 sample

Trainingsdaten 569 sample

Tree induction vs. logistic regression

Example: ~~Breast Cancer Dataset~~

Results of logistic regression

$f(x) = -17{,}70 + 22{,}3 \times 1 + 19{,}47 x_2 \ldots$

- Weights of linear model
- Ordered from highest to lowest
- Performance: only six mistakes on the entire dataset, accuracy 98.9%

$$\frac{1}{1+e^{-f(x)}} = P_B(x)$$

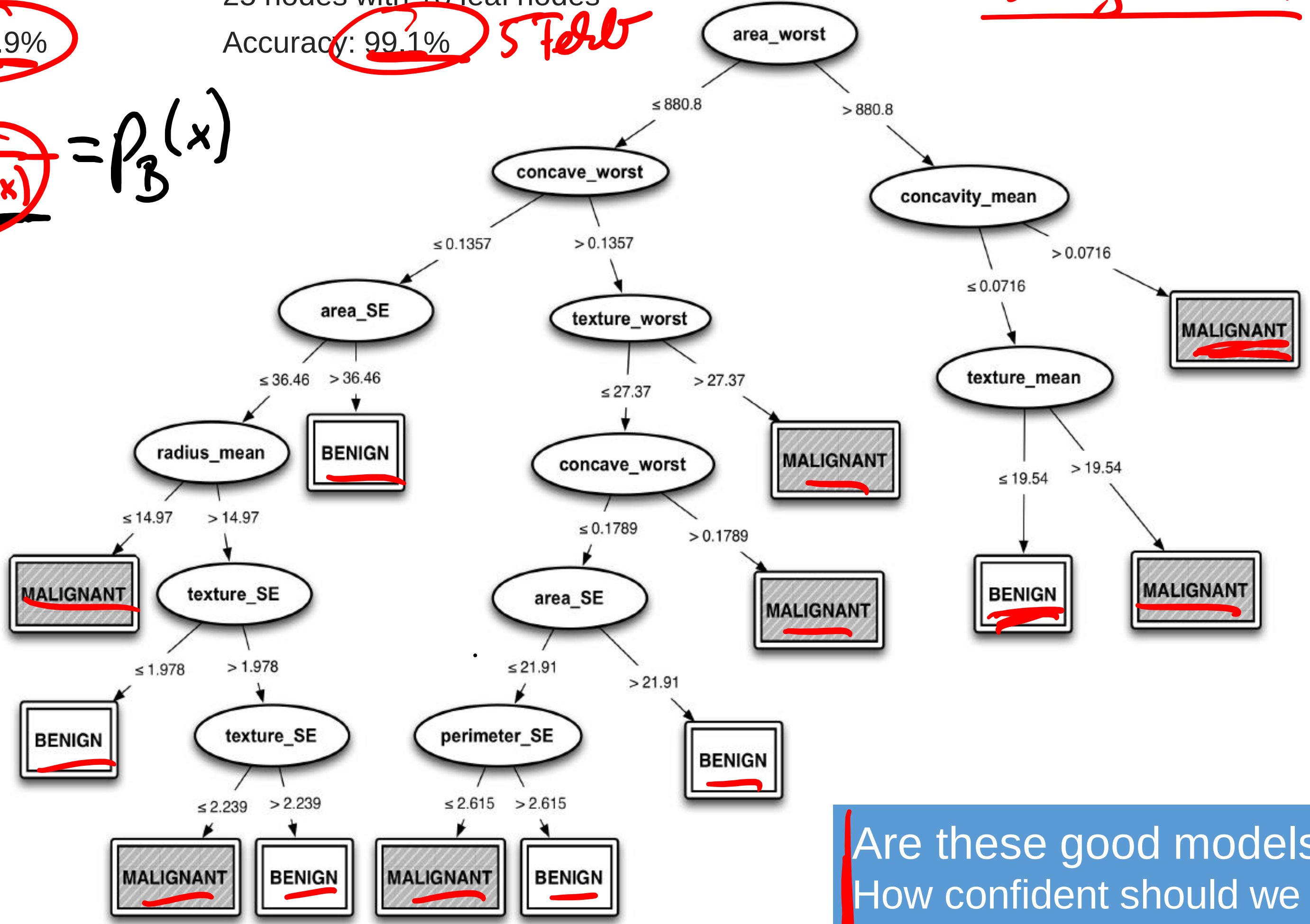
Attribute		Weight
Smoothness_worst	x_1	22.30
Concave_mean	x_2	19.47
Concave_worst	x_3	11.68
Symmetry_worst	x_4	4.99
Concavity_worst	⋮	2.86
Concavity_mean		2.34
Radius_worst		0.25
Texture_worst		0.13
Area_SE		0.06
Texture_mean		0.03
Texture_SE		-0.29
Compactness_mean		-7.10
Compactness_SE		-27.87
w_0 (intercept)		-17.70

$f(x) = w_0 + w_1 x_1 + w_2 x_2 + \ldots x_i \ldots$

Comparison with classification tree from the same dataset

13 Segmente

- Weka's J48 implementation
- 25 nodes with 13 leaf nodes
- Accuracy: 99.1% 5 Fehler



Are these good models?
How confident should we be in this evaluation?

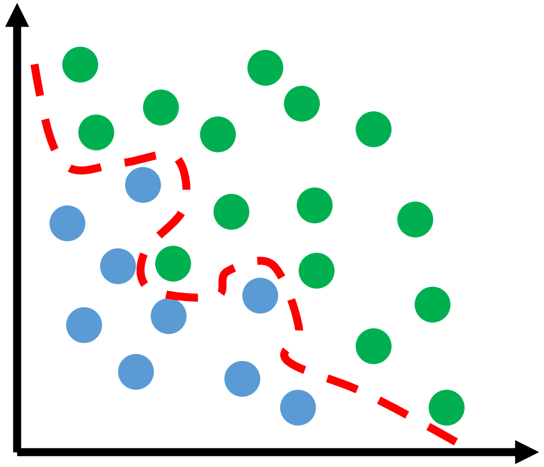
How to avoid overfitting? – Next Lesson

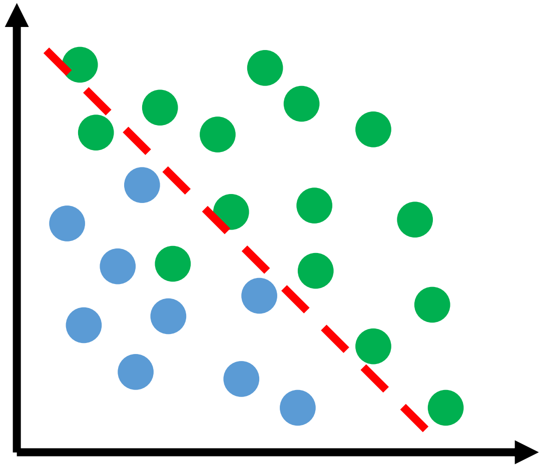
Introduction

Fundamental trade-off in DM between **overfitting** and **generalization**

If we allow ourselves enough flexibility in searching, we *will* find patterns

Unfortunately, these patterns may be just occurences by chance





Overfitting: finding chance occurences in data that *look like* interesting patterns, but which do *not generalize*

We are interested in patterns that **generalize**, i.e., that predict well for instances that we have not yet observed

Ref.

Fragen?

- ✓ Tree induction vs. linear classifier (in general)

- ✓ Linear regression
- ✓ Logistic regression
- ✓ Tree-induction vs. logistic regression

Recommended reading

Fitting a Model:

Provost, F., Fawcett, T.	Data Science for Business Chapter 4
Berthold et al.	Guide to Intelligent Data Analysis Chapter 8.3

Almost all introductory books on statistics include regression

How to avoid Overfitting (next lesson):

Provost, F., Fawcett, T.	Data Science for Business Chapter 5
Berthold et al.	Several subchapters

Ref.