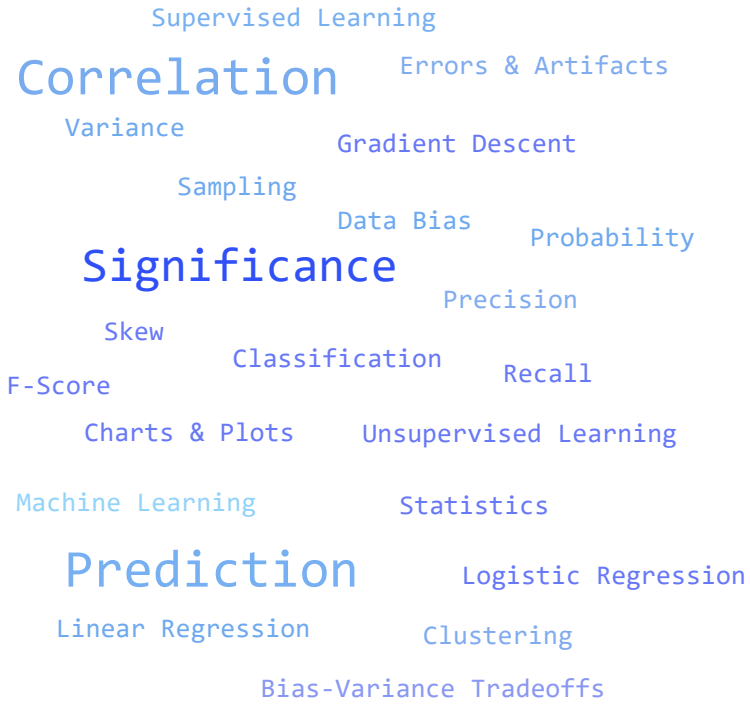


Data Science 1:
Introduction to Data Science

Logistic Regression & Classification

Winter 2025

Wolfram Wingerath, Jannik Schröder

Department for Computing Science
Data Science / Information Systems

Lecture slides based on content from “The Data Science Design Manual” (Steven Skiena, 2017) and associated course materials generously made available online by the author at https://www3.cs.stonybrook.edu/~skiena/data-manual/.

Special thanks to Professor Skiena for sharing these valuable teaching resources!

Semester Schedule

CW 42	14. Oct	Lecture	1	Orga & Intro	1-26
CW 43	21. / 23. Oct	Lecture + Exercises	2	Probability, Statistics & Correlation	27-56
CW 44	28. Oct	Lecture	3	Data Munging, Cleaning & Bias	57-94 / “Invisible Women”
CW 45	04. / 06. Nov	Lecture + Exercises	4	Scores & Rankings	95-120
CW 46	11. Nov	Lecture	5	Statistical Distributions & Significance	121-154
CW 47	18. / 20. Nov	Lecture + Exercises	6	Building & Evaluating Models	201-236
CW 48	25. Nov	Guest Lecture	7	Data Visualization	155-200
CW 49	02. / 04. Dec	Lecture + Exercises	8	Intro to Machine Learning	351-390
CW 50	09. Dec	Lecture	9	Linear Algebra	237-266
CW 51	16. / 18. Dec	Lecture + Exercises	10	Linear Regression & Gradient Descent	267-288
CW 02	06. Jan	Lecture	11	Logistic Regression & Classification	289-302
CW 03	13. / 15. Jan	Lecture + Exercises	12	Nearest Neighbor Methods & Clustering	303-350
CW 04	20. Jan	Lecture	13	Data Science in the Wild	391-426
CW 05	27. / 29. Jan	Lecture + Exercises	14	Q&A / Feedback	
CW 06	03. / 04. Feb	Oral Exams (Block 1)		Preparation in our last session („Oral Exam Briefing“)	
CW 13	24. / 25. Mar	Oral Exams (Block 2)			

Classification Problems

Often we are given collections of examples *labeled* by class:

- male / female?
- democrat / republican?
- spam / non-spam (ham)?
- cancer / benign?

Classification assigns a label to an input record.

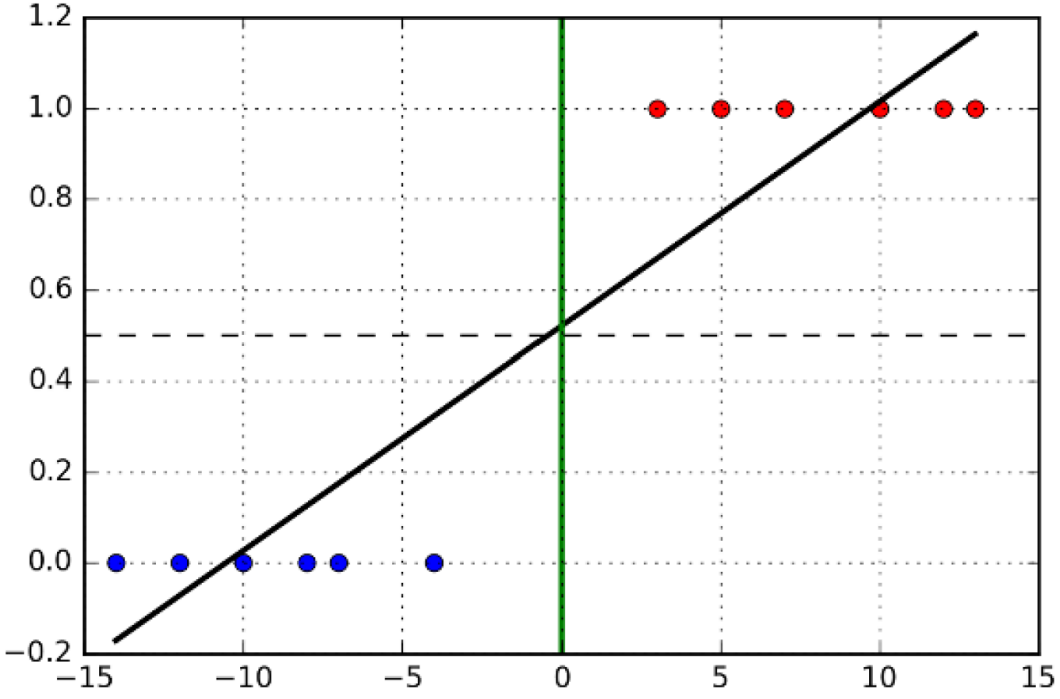
Regression for Classification

We *could* use linear regression to build from annotated examples by converting the class names to numbers:

- male=0 / female=1
- democrat=0 / republican=1
- spam=1 / non-spam=0
- cancer=1 / benign=0

Zero/one works for binary classifiers. By convention, the “positive” class gets 1 and the “negative” one 0.

Class Labels from Regression Lines

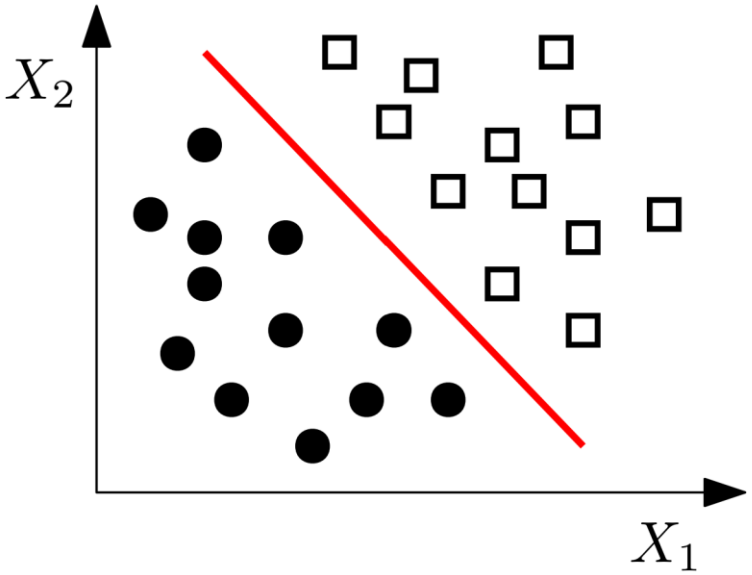


The regression line will cut through these classes, even though a separator exists.

Adding very +/- examples shifts the line but hurts the boundary

Decision Boundaries

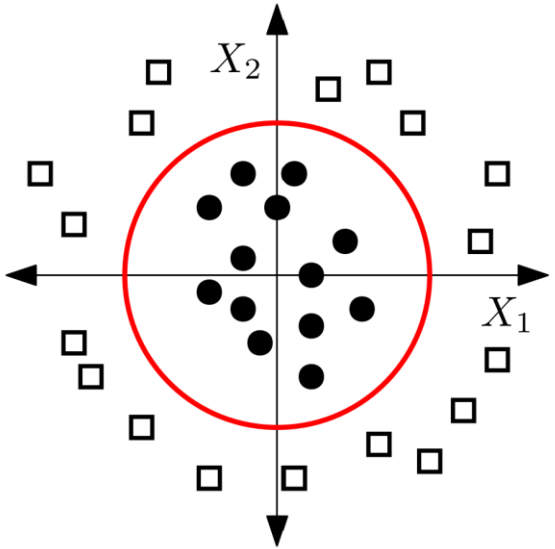
Ideally, our two classes will be well-separated in feature space, so a line can partition them.



Logistic regression is a method to find the best separating line for a given training set.

Non-Linear Decision Boundaries

Logistic regression can find non-linear boundaries if seeded with non-linear features.



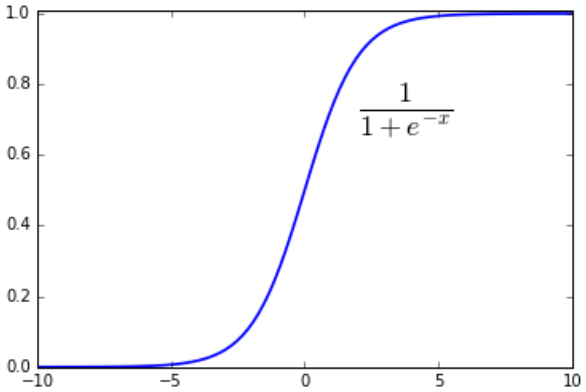
To get separating circles, we need to explicitly add features like x^2 and $x_1 \cdot x_2$.

The Logistic Function

We want a way to take a real variable and convert it to a probability:

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

- f(0) = ½
- f(infinity) = 1
- f(-infinity) = 0



The logistic function gives the probability of x being in a particular class.

Logistic Function for Classification

To extend the logistic function to m variables and set the threshold and steepness parameters, fit

$$h(x, w) = w_0 + \sum_{i=1}^{m-1} w_i \cdot x_i$$

Plug into the logistic function to get a classifier:

$$f(x) = \frac{1}{1 + e^{-h(x,w)}}$$

Scoring for Logistic Regression

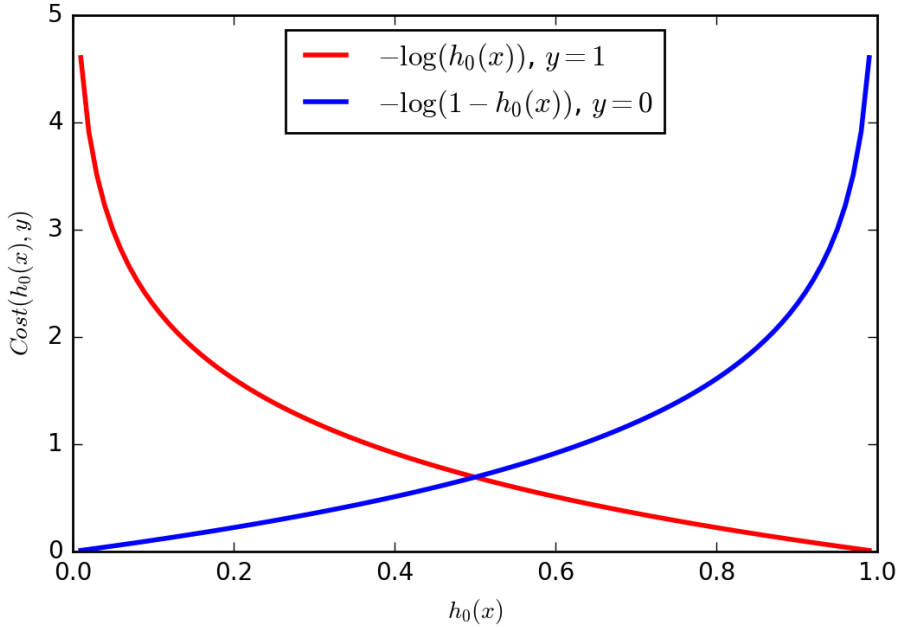
Assume each of our n training examples are labeled as to being in either class 0 or 1.

We seek to identify the coefficients w that give high probability on positive (class 1) items, and low probability on negative (class 0) items.

We need a cost function to value a probability p for an item of class c.

Costs for Positive / Negative Cases

We want zero error for the best probability, and increasing cost as the class prediction gets more and more wrong.



Logarithms of Probabilities

Observe that log(1) = 0. Thus it gives proper cost for correct predictions of positive items.

With $\log 0 \to -\infty$,we can use the cost function:

$$\text{cost}(x_i, 1) = -\log(\text{f}(x_i))$$

For negative instances, the cost function is:

$$\text{cost}(x_i, 0) = -\log(1 - \text{f}(x_i))$$

Cost / Loss Function

We can use the zero / one labels as indicator variables to compute the loss function:

$$\mathrm{J}(w) = \frac{1}{n}\sum_{i=1}^{n} \mathrm{cost}(\mathrm{f}(x_i, w), y_i)$$

$$= -\frac{1}{n}\sum_{i=1}^{n} y_i \log \mathrm{f}(x_i, w) + (1 - y_i) \log(1 - \mathrm{f}(x_i, w))$$

This works because the class indicator variables are 0 and 1.

Logistic Regression via Gradient Descent

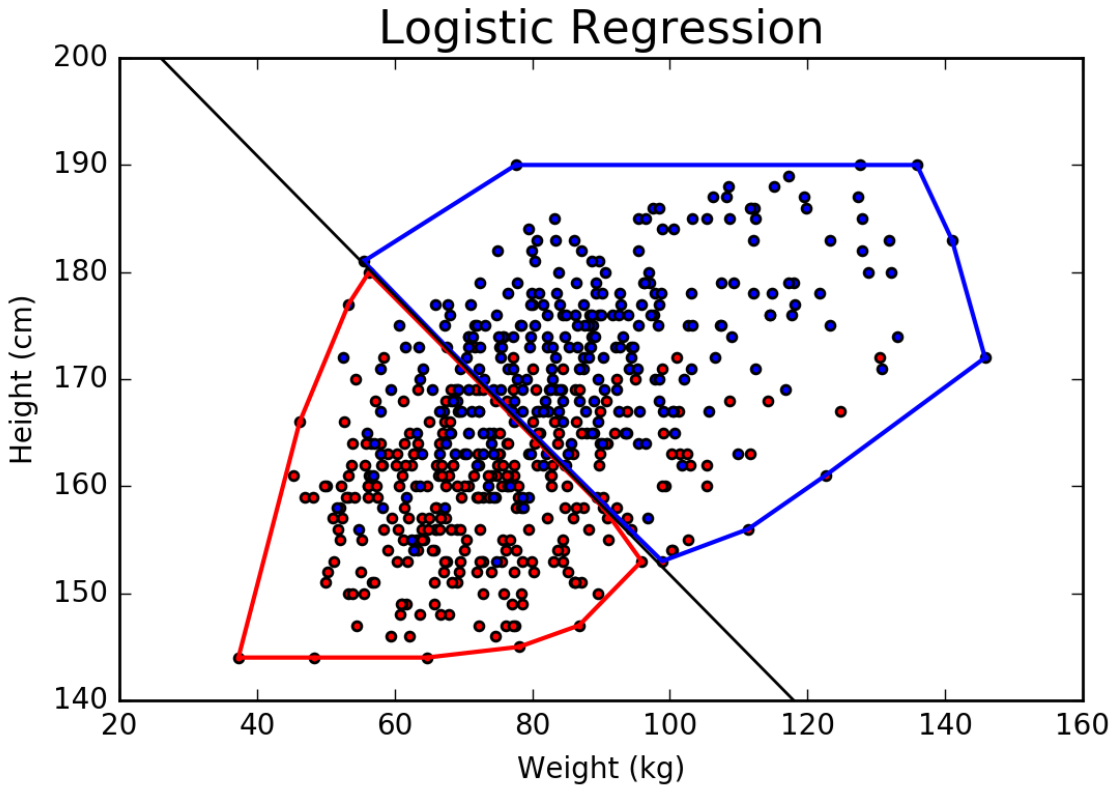
The loss function here is convex, so we can find the parameters which best fit it by gradient descent.

Thus we can find the best linear separator between two classes (linear in the possibly non-linear features).

Logistic Gender Classification

Red region:
229 w / 63 m

Blue region:
223 m / 65 w



Issues in Logistic Classification

- Balanced training class sizes
- Multi-class classification
- Hierarchical classification
- Partition functions

Balanced Training Classes

Consider the optimal separating line for grossly unbalanced class sizes, say 1 positive example vs. 1,000,000 negative examples.

The best scoring line from logistic regression will try to be very far from the big cluster instead of the midpoint between them.

Use equal numbers of positive / negative examples

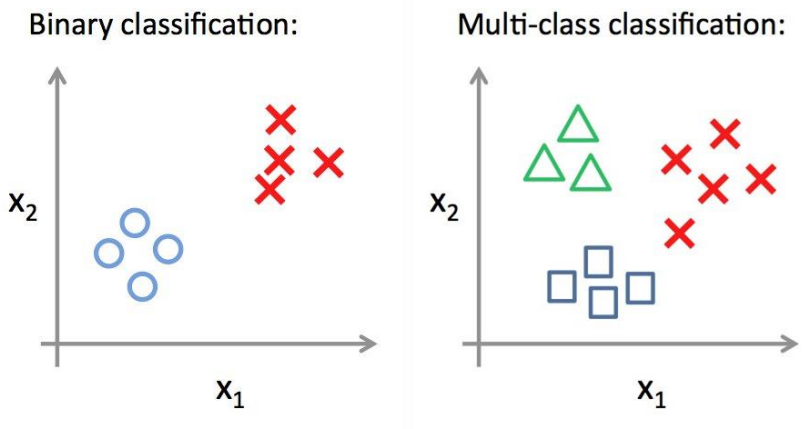
Ways to Balance Classes

- Work harder to find members of the minority class.
- Discard elements from the bigger class.
- Weigh the minority class more heavily, but beware of overfitting.
- Replicate members of the smaller class, ideally with random perturbation.

Multi-class Classification

Classification tasks are not always binary.

Is a given movie a comedy, drama, action, documentary, etc.?



Encoding Multi-Classes: Bad Idea

It is natural to represent multiple classes by distinct numbers: blond=0, brown=1, red=2.

But unless the ordering of the classes reflect an increasing scale, the numbering is meaningless.

Classes on a Likert scale are ordinal.

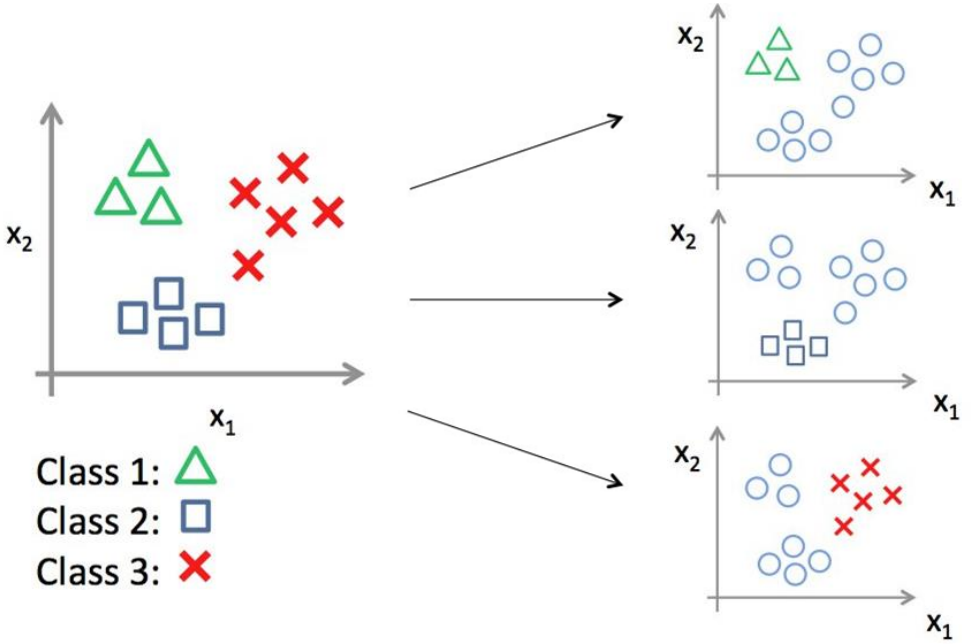
4 stars - 3 stars - 2 stars - 1 star

- Completely Agree
- Mostly Agree
- Slightly Agree
- Slightly Disagree
- Mostly Disagree
- Completely Disagree

One Versus All Classifiers

We can build multi-class classifiers by building multiple independent binary classifiers.

Select the class of highest probability as the predicted label.



The Challenges of Many Classes

Multi-class classification gets much harder as the number of classes increase.

The monkey is right only *1/k* times for *k* classes.

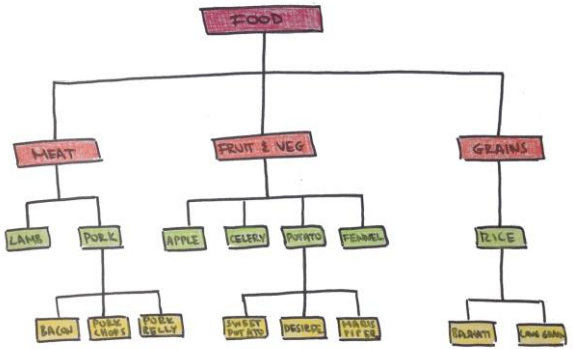
- With many available classes, certain mistakes are more acceptable than others.
- The actual population sizes of rare classes can easily become vanishingly small.

Hierarchical Classification

Grouping classes by similarity and building a taxonomy reduces the effective number of classes.

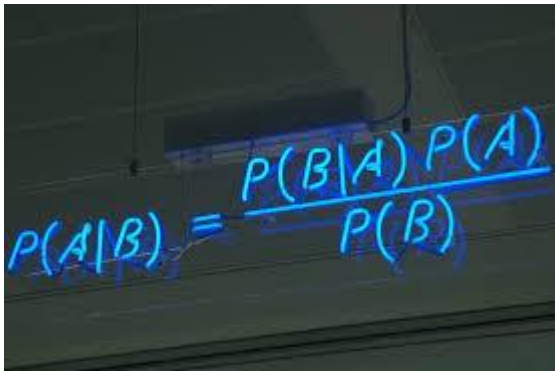
ImageNet has 27 categories (e.g. animal, appliance, food, furniture) on top of 21,841 subcategories.

Classify from top-down in tree.



Bayesian Priors

Having honest estimates for the probability distribution of classes is essential to avoid domination by rare classes (“rock stars”)



Here A is the given class, and B is the event your classifier returned a guess of class A.

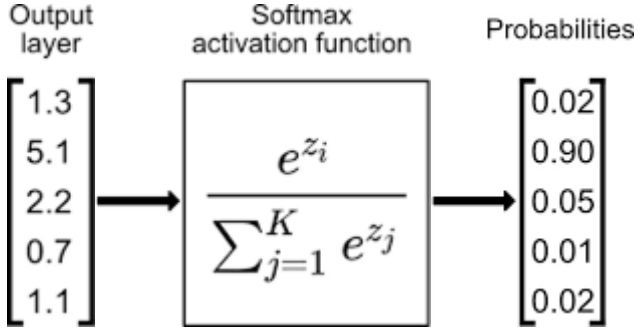
Partition Functions

There is no reason why independent binary classifiers yield probabilities which sum to 1.

To get real probabilities for Bayes theorem:

$$T = \sum_{i=1}^{c} F_i(x) \quad \text{and} \quad \mathrm{P(c)} = \frac{F_c(x)}{T}$$

Alternately, softmax is



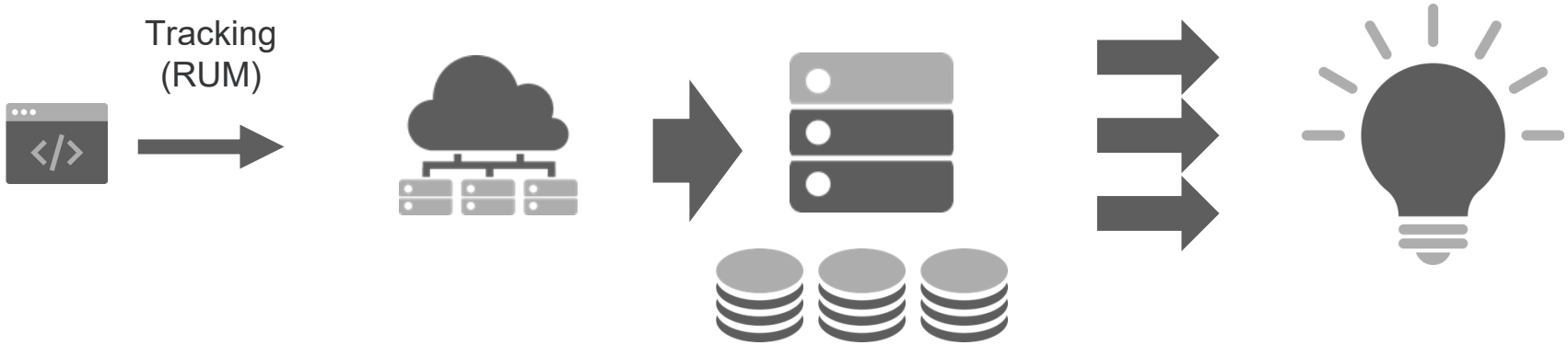
Classification Use Case: Bot Detection

Bot traffic creates costs, but it does not generate revenue → automated (**ML-based**) decisionmaking required (bots vs. users)!

Rule-based bot detection, e.g.:
- CDN heuristics
- Useragent strings
- Specific metrics (e.g. First Input Delay)

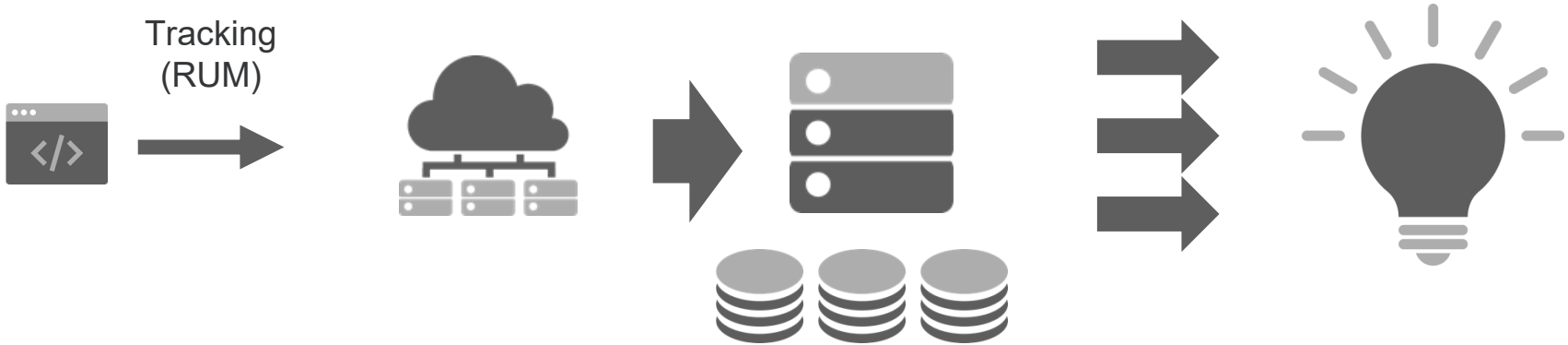
Detecting new bots via **manual analyses**

Collection | Storage & Serving | Aggregation | Analysis



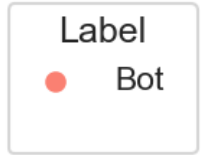
Classification Use Case: Bot Detection

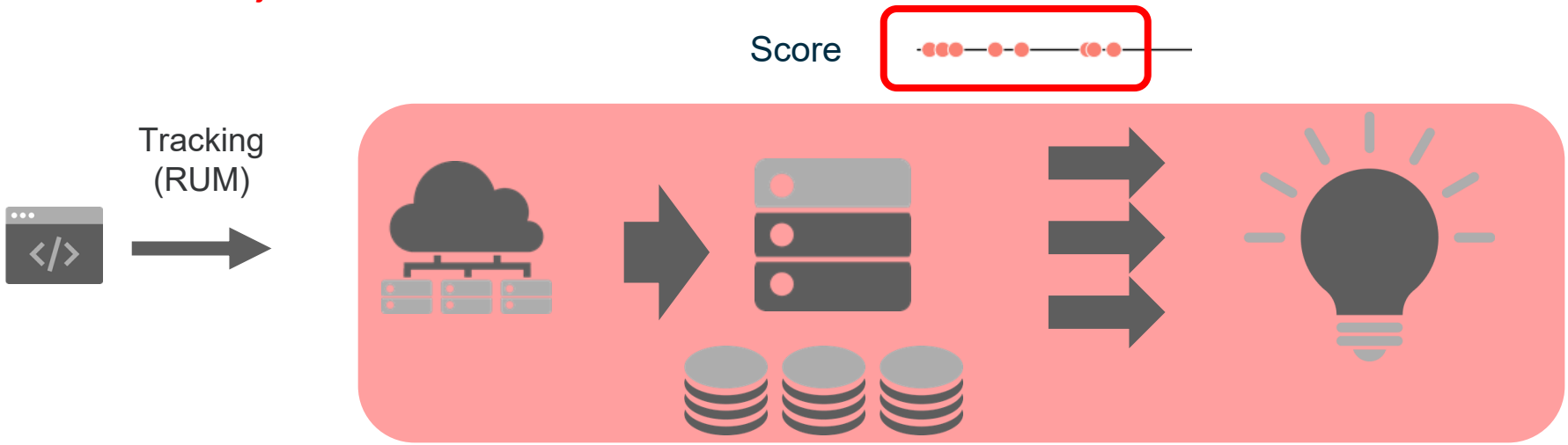
- **Input Data**: manually labeled tracking data
- **Goal**: automated prediction/assignment, i.e. decision at runtime where the new data points belong to users or bots
  - → Reduce infrastructure expenses: no costly optimizations for bots (e.g. caching)
  - → Increase data quality: fewer analyses are distorted by synthetic traffic



Classification Use Case: Bot Detection

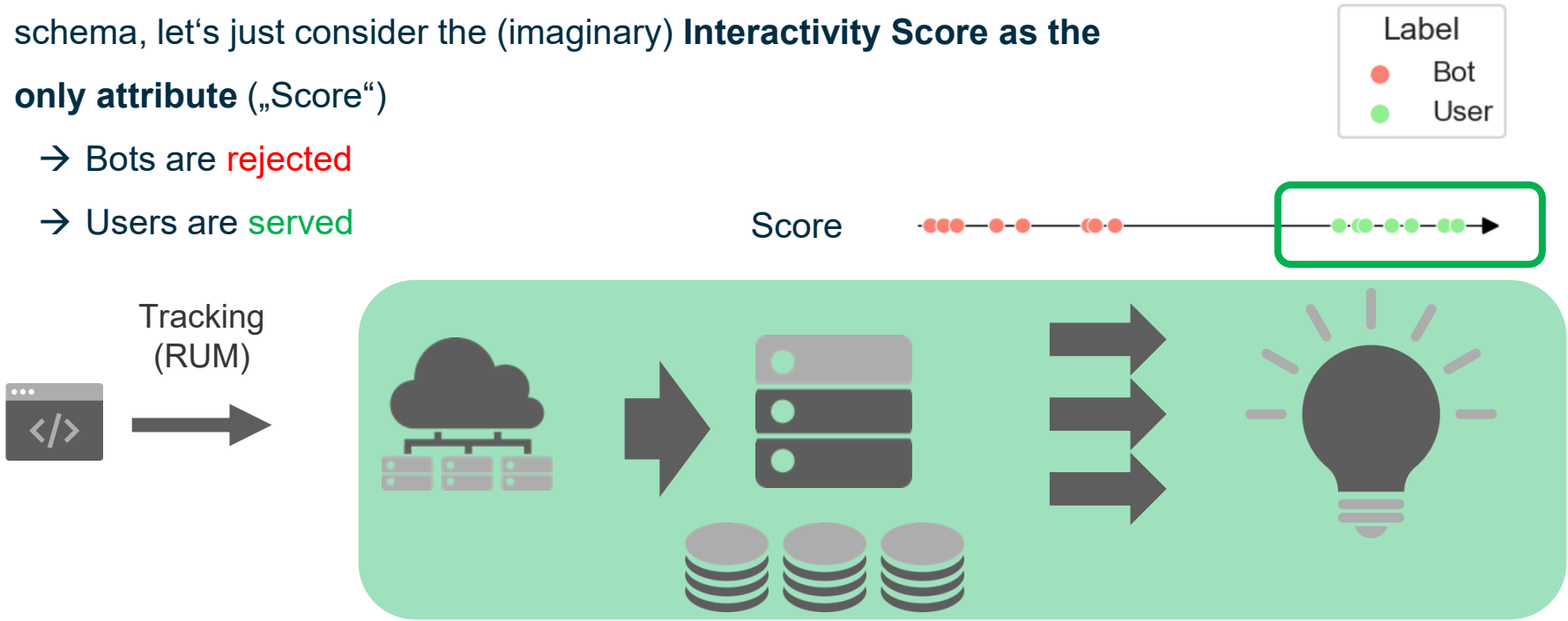
- **Simplification**: instead of the actual ~160 tributes in the tracking schema, let‘s just consider the (imaginary) **Interactivity Score as the only attribute** („Score“)
  → Bots are rejected





Classification Use Case: Bot Detection

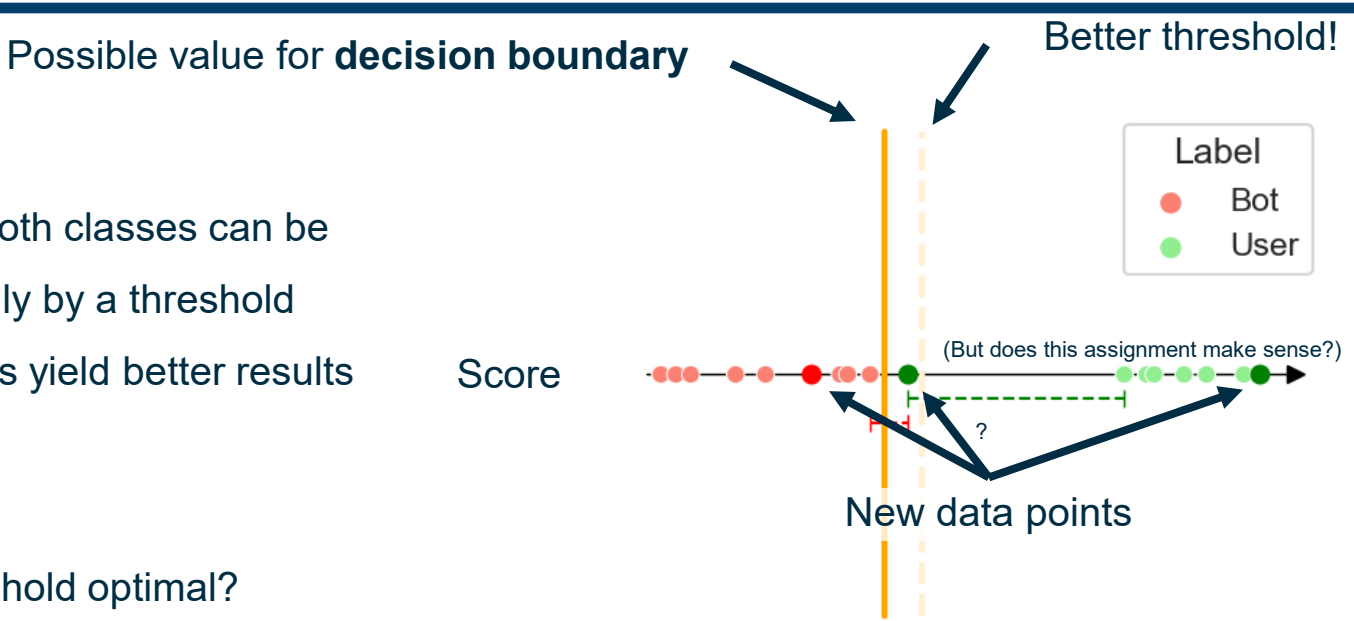
- **Simplification**: instead of the actual ~160 attributes in the tracking schema, let‘s just consider the (imaginary) **Interactivity Score as the only attribute** („Score“)
  - → Bots are rejected
  - → Users are served



Naïve Threshold-Based Classification

- **Observations**:
  - Data points of both classes can be separated cleanly by a threshold
  - Some thresholds yield better results than others
- **Question**:

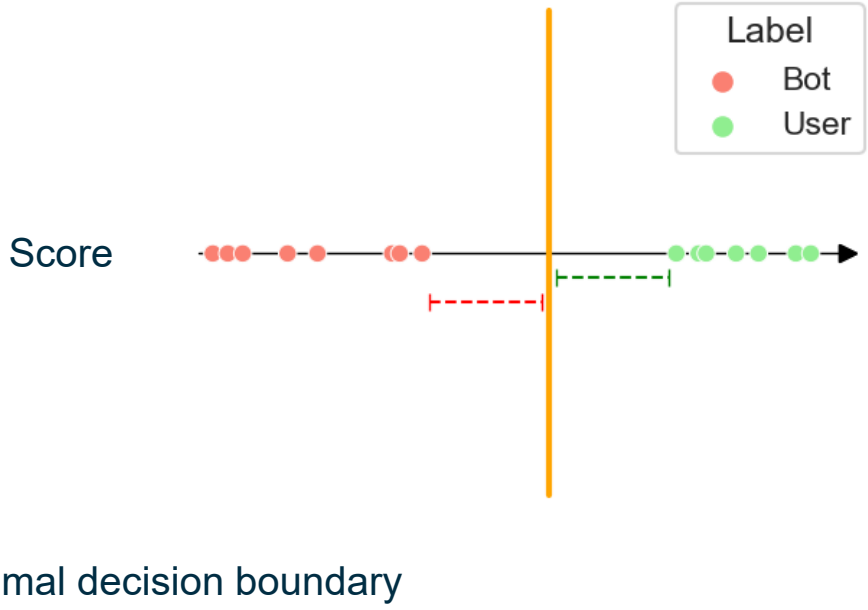
  What makes a threshold optimal?



The Optimal Classification Threshold

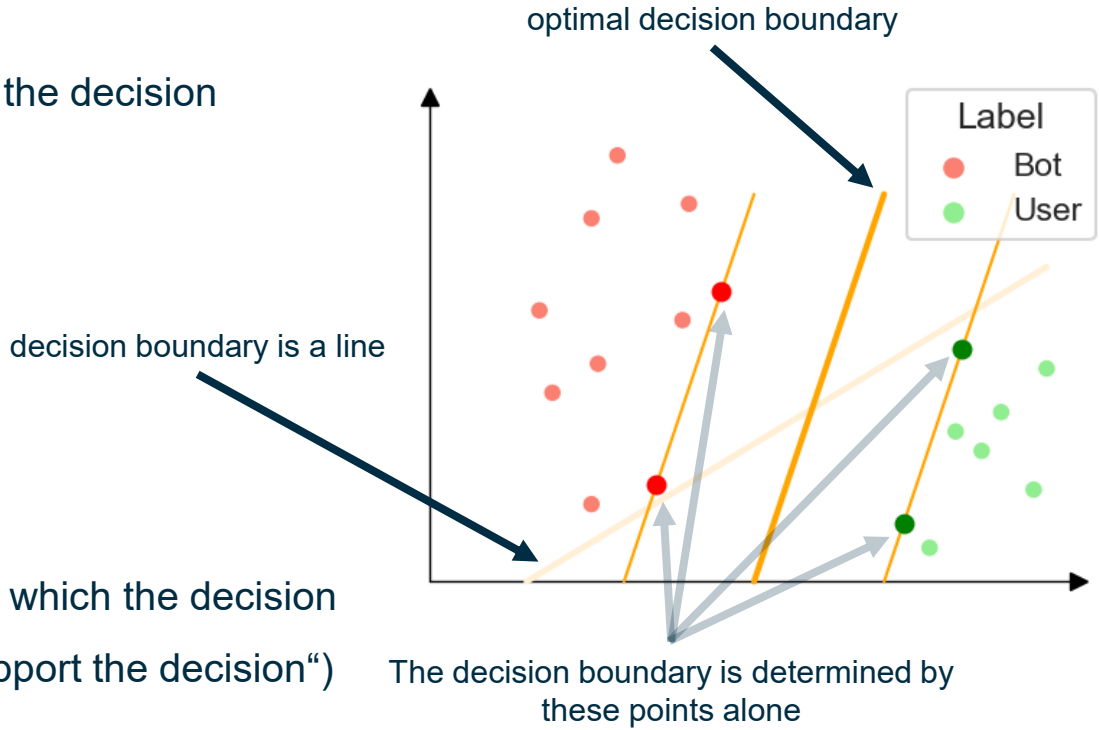
- **Margin**:
  - Distance between threshold and training data
  - Is maximized when distance is the same on both sides

→ **Maximal Margin Classifier**



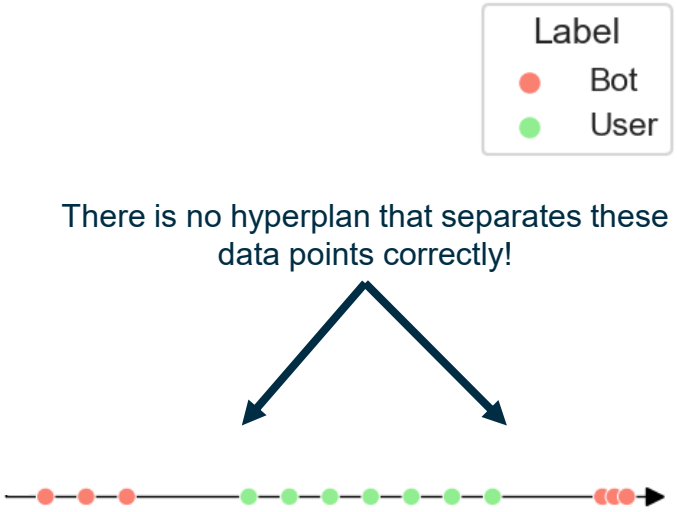
What Exactly Does “Optimal” Mean?

- **Hyperplane**:
  - In multidimensional spaces, the decision boundary is a hyperplane
  - → 2D: separating line
  - → 3D: separating plane
  - → …
- **Support Vectors**:
  - The data points (vectors) on which the decision boundary depends („that support the decision“)



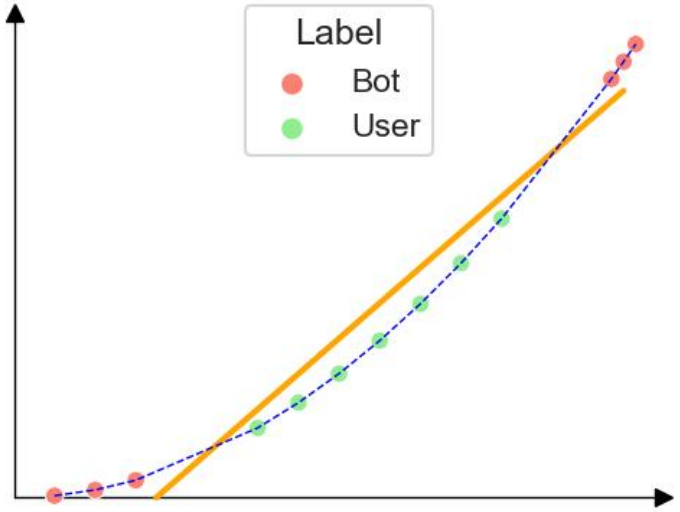
But What About More Complex Data?

- **Problem**: Data points may not be separable by a simple threshold!



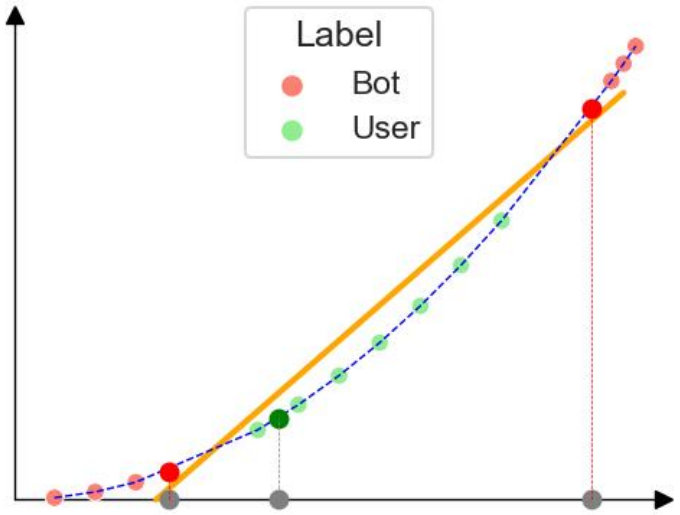
But What About More Complex Data?

- **Problem**: Data points may not be separable by a simple threshold!
- **Idea**: We add another feature to our dataset (Y-Axis) by squaring X values
  - → In this new 2-dimensional space, the data points are linearly separable



But What About More Complex Data?

- **Problem**: Data points may not be separable by a simple threshold!
- **Idea**: We add another feature to our dataset (Y-Axis) by squaring X values
  - → In this new 2-dimensional space, the data points are linearly separable
  - → For new data points:
    1. Transform from 1D to 2D
    2. Classify in 2D



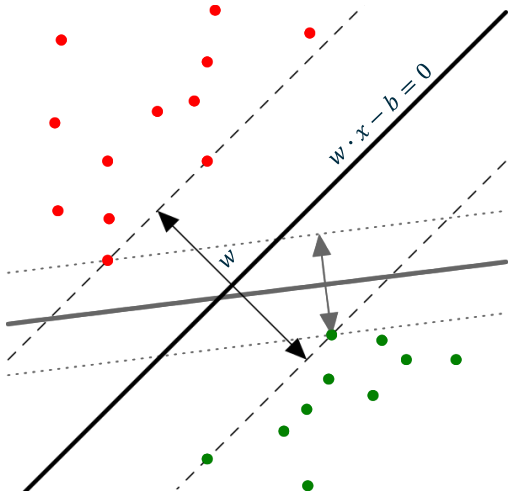
Support Vector Machines

- **Classification** by the following procedure:

Paul Henckels. Die Feuerzangenbowle, 1944

Support Vector Machines

- **Classification** by the following procedure:
  1. Transform data into higher-dimensional space
  2. Find Maximal Margin (Support Vector) Classifier
  3. New data points: first transform, then classify



Complex Optimization Problem:

We have to choose the hyperplane that maximizes the smallest **distance d** to the plane of a **point x** (given by **normal vector w** and **constant b**).

$$d = \frac{w \cdot x + b}{\|w\|}$$

Adapted from: Jens Pönisch. Grundlagen von Support Vector Machines (SVM), TU Chemnitz, 2019

The Kernel (Trick)

Problem: Frequent computation of dot product of two vectors in transformed space

- Input Space: $x \in \mathbb{R}$
- Transformation: $\phi(x) = (x, x^2, 1/2)$ (d.h. $\phi(x) \in \mathbb{R}^3$)

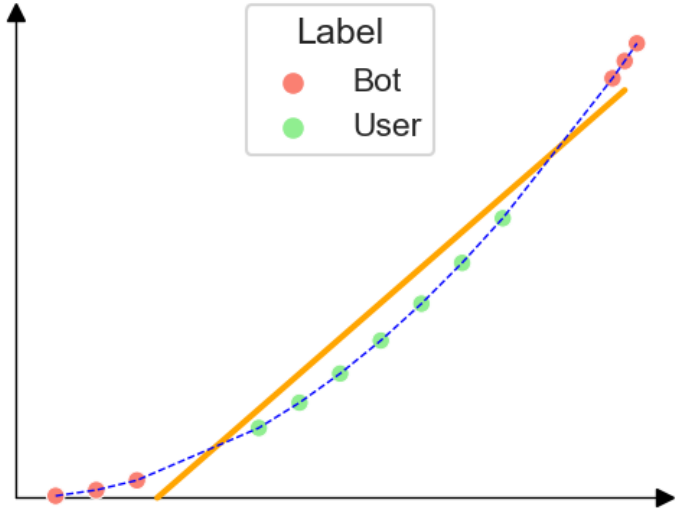
irrelevant since constant

- Dot product: $\phi(x) \cdot \phi(y) = xy + x^2y^2 + 1/4$

$$= (xy + 1/2)^2$$

$K(x, y) = (\gamma(xy) + r)^d$ with $\gamma = 1, r = 1/2, d = 2$

Kernel function $K(x, y)$ (polynomial)

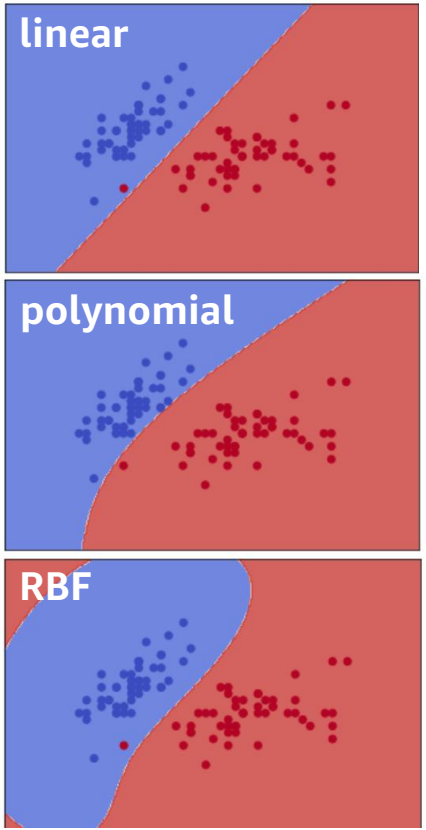


Solution: computation $f: \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}^3 \times \mathbb{R}^3 \rightarrow \mathbb{R}$ is simplified to $f': \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}$

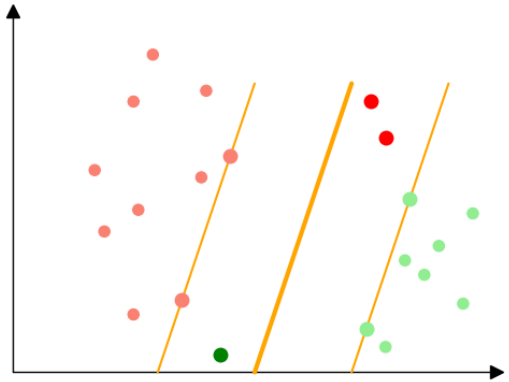
Using SVMs in Practice

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn import svm, datasets, metrics

# read data
bd = read_data(BOTS)
X = bd.data[:, 0:2] # only 2 features for better illustration
y = bd.target

# instantiate classifier, tune parameters (approximate optimal hyperplane)
svc_linear = SVC(kernel = 'linear').fit(X, y)
svc_poly = SVC(kernel = 'poly', degree = 5, gamma = 1, coef0 = 0.5).fit(X, y)
svc_rbf = SVC(kernel = 'rbf', gamma = 0.7).fit(X, y)

# define plotting area
x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1
y_min, y_max = X[:, 1].min() - 1, X[:, 1].max() + 1
xx, yy = np.meshgrid(np.arange(x_min, x_max, h), np.arange(y_min, y_max, h))

# create plots
titles = ['linear', 'polynomial', 'RBF`]

for i, clf in enumerate((svc_linear, svc_poly, svc_rbf)):
    ... # plot decision boundary und training data
plt.show()
```

Note: no split between training and test data!



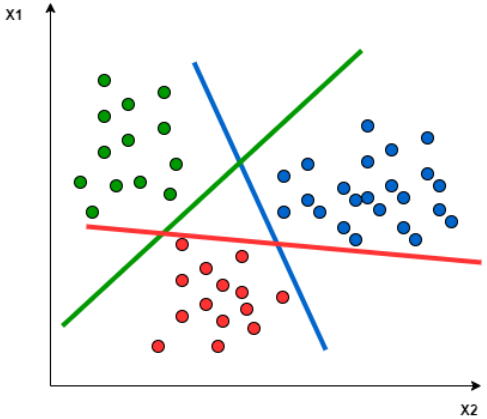
Advanced SVM Topics



Classification with > 2 Classes

Combining multiple SVMs

(One-to-One / One-to-Rest)



Soft Margins

Tolerance against misclassification, trade-off against margin size, increased robustness against outliers

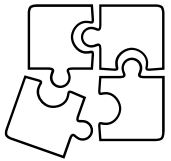
Kernels

Strengths, weaknesses & para-metrization of different Kernels

Eugen Baeldung. Multiclass Classification Using Support Vector Machines, Baeldung.com, 2022

Wrapup:

Log. Regression & Classification



- Classification is a subtopic of supervised ML
- Regression can be used for classification
- Challenges include balancing classes, non-binary decisions, and computing probabilities
- SVMs have high practical relevance due to their many advantages (e.g. geometric representation, ease of implementation, high efficiency → Kernel trick)