

Class 01

Introduction to the Module



Computational Statistics

Prof. Dr. Jens Witkowski
September 4, 2023

Agenda

- Session 1
 - Catching up
 - Module overview and Organization
 - Forecasting Game
- Session 2
 - Artificial Intelligence vs Machine Learning vs Classical Statistics
 - Linear Regression re-visited

Update

What's new?

Organization

- **Attendance and Participation**
 - Mutual respect: risk-free learning environment.
 - No explicit participation grading
 - Bring to class:
 - Your name tags.
 - Fully charged laptops.
- **All In-Class, on Campus**
- **Office hours: by appointment**
 - Please reach out beforehand via email with your specific questions.
- **Communication**
 - Check Canvas regularly for course materials, announcements, and other updates.
 - Email is the preferred way to get in touch with me: j.witkowski@fs.de (I do not get notified about Canvas messages, so I might respond to those delayed.)

Respectful Conduct

- Arrive on time. This includes coming back from the break.
- If you will arrive late: write me an email beforehand.
- If you will have to leave early: write me an email or let me know at beginning of class.
- In interest of learning experience: turn off your cell phones and use laptops only as instructed.
- If you have to use the bathroom, leave and enter the room quietly.

General guidance: Think of each lecture as a business meeting.

Deliverables and Grading

Deliverables and associated points

3 Homeworks (groups of three)	60 Points
Final exam (individual)	60 Points
Total	120 Points

Homeworks:

- Student groups of three.
- All students in a group need to fully understand and be able to explain the solution to me.
- Demonstration of mastery of the contents from class as well as their application beyond mere replication.
- Contains both math exercises and programming / data analysis.
- May or may not contain a peer evaluation aspect.
- First homework handed out on September 7 (after class) and due on September 14 (before class).

Academic Integrity:

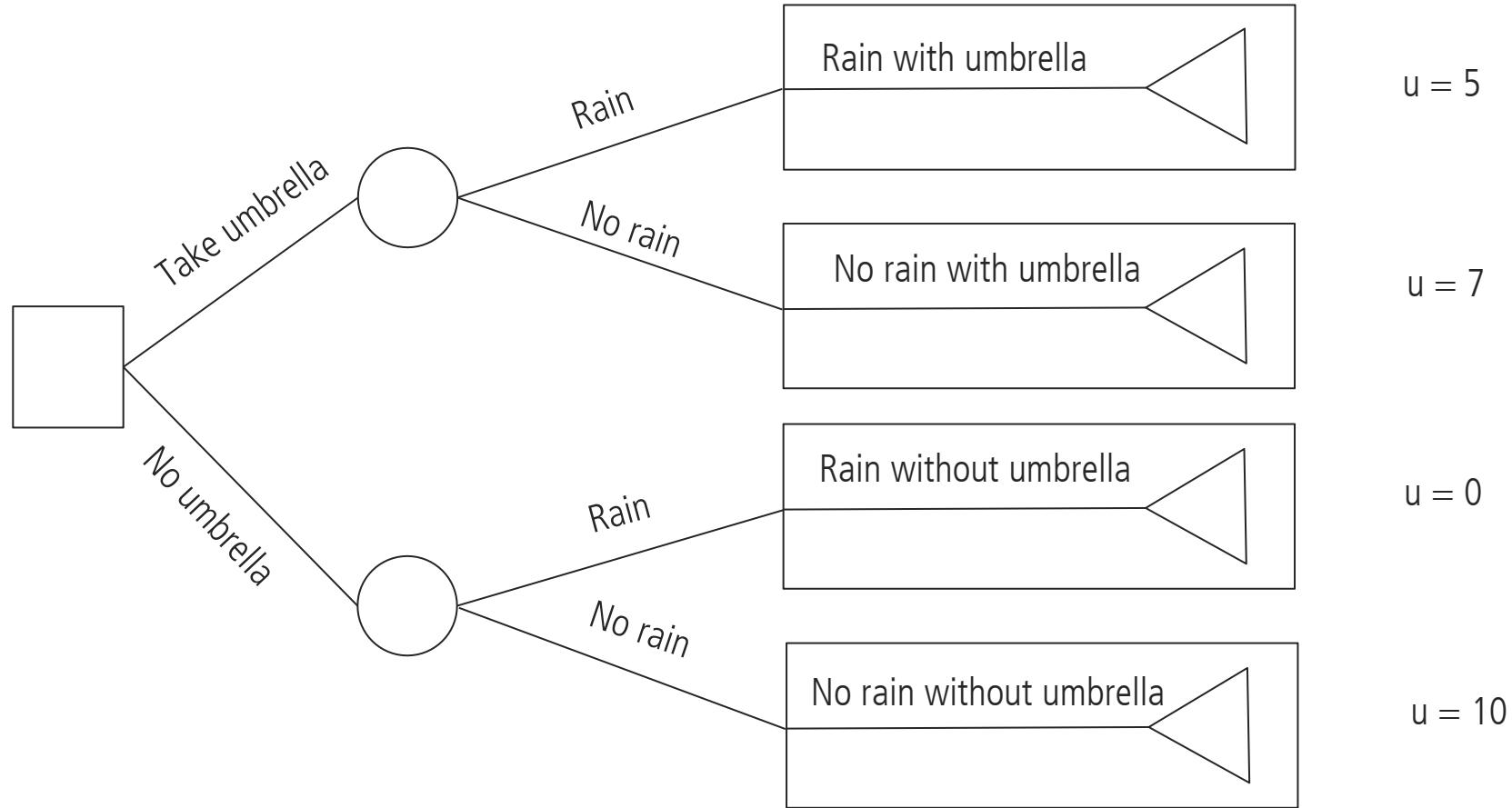
- Academic integrity is a core value of Frankfurt School's Code of Honor.
- You may consult with others but it is not allowed to copy or share solutions and/or code!
- I am taking cheating seriously.

High-Level Course Outline

- Part 1: Introduction to Supervised Machine Learning
- Part 2: Crowd Forecasting and Decision Making

Both parts are about predictions!

Why care about Predictions?



Predictions are basis of decision making:
need probability of rain to determine a decision maximizing expected utility!

First Part: Supervised Machine Learning

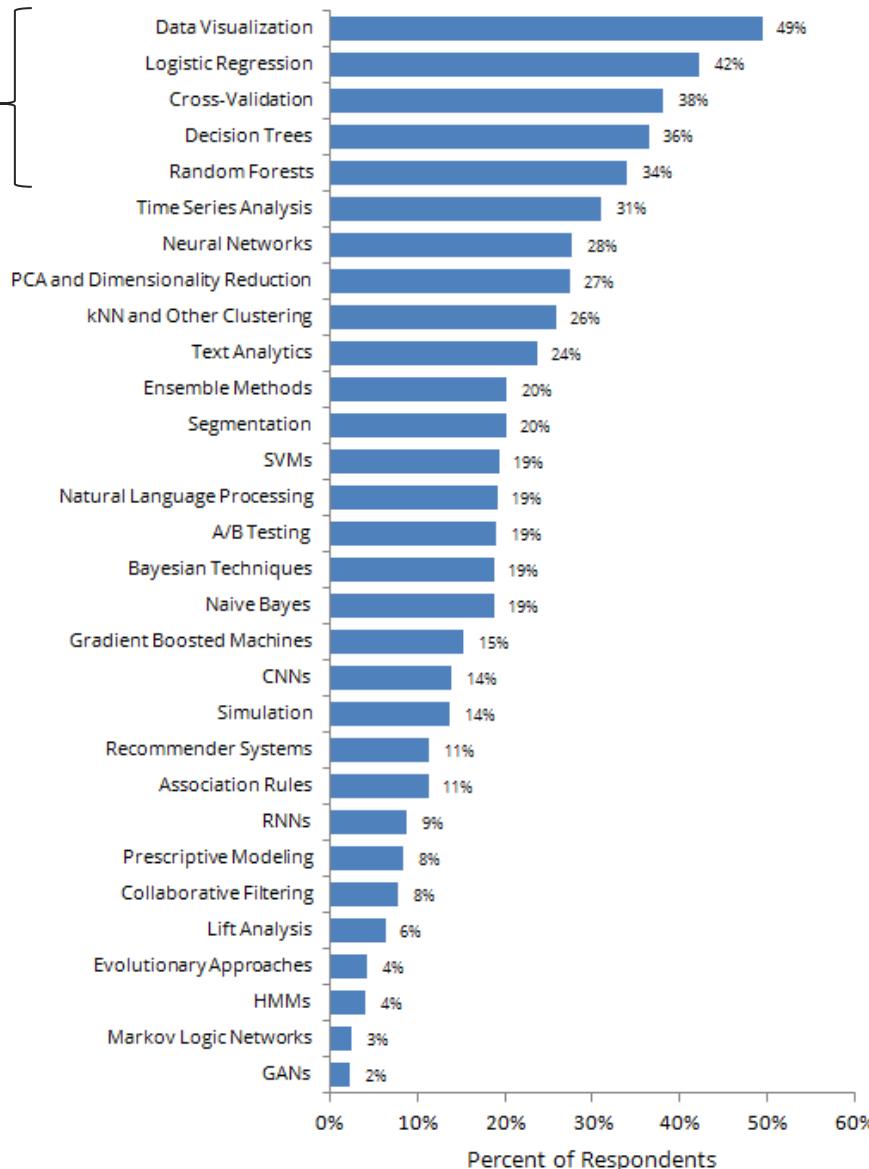
Example: Should a bank give a loan to person X?



- Depends on probability of default by X and bank's profit in cases of "default" and "no default"
- Machine learning approaches use large datasets of previous loans and their outcomes to estimate probability (make prediction) that X will default.

Data Science Methods Used at Work

Will get to know some of these



[Kaggle, 2018]

Second Part: Crowd Predictions

What is probability of armed conflict between Mainland China and Taiwan within the next 6 months?



- Not enough relevant historical data for pure data-driven approach
- Crowd forecasting methods aggregate forecasts of many individuals into single probabilistic forecast
- Used by governmental agencies, financial institutions, and retailers (e.g., demand predictions for particular style or color of clothing)

Most large-scale, strategic decisions are *supported* by data but there are too few similar situations with known outcomes ("labels") that pure machine learning approaches are practical!

Crowd Forecasting

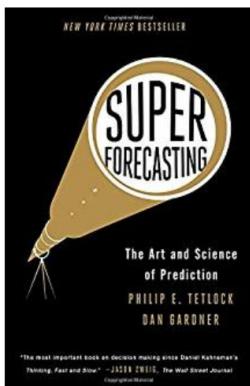
Wisdom of the Crowd: Eliciting and aggregating many people's beliefs (expert or non-expert).

The Good Judgment Project™



#1244 Will India and/or Brazil become a permanent member of the U.N. Security Council before 1 March 2015?

How likely is this event?  42 %



Superforecasting: The Art and Science of Prediction

by Philip E. Tetlock

Paperback

\$11.55 \$17.00 

You save \$5.45 (32%)

Get it by Tomorrow, Apr 4

66 offers from \$7.00

 (256)

4 Other Formats: Kindle Edition, Hardcover,
[+2 more](#)

Some questions we will address:

- How to identify good forecasters in a crowd?
- How to aggregate forecasts from different forecasters?
- How to incentivize forecasters to invest effort and report truthfully.

Data Science and Forecasting Competitions



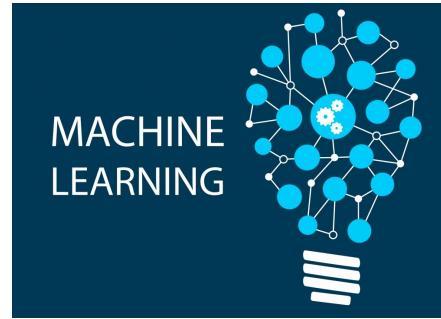
 A screenshot of a computer screen showing a Kaggle competition page. The competition is titled "Store Sales - Time Series Forecasting" and is described as "Use machine learning to predict grocery sales". It shows "867 teams" and is currently "Ongoing". The page includes a navigation bar with "Overview", "Data", "Code", "Discussion", "Leaderboard" (which is selected), and "Rules". Below the navigation is a "Leaderboard" section with a table of 10 entries. The table columns are "#", "Team", "Members", "Score", "Entries", "Last", and "Join". The top entry is "Jason_". The table is sorted by Score.

#	Team	Members	Score	Entries	Last	Join
1	Jason_		0.37786	1	20d	
2	Javier Reinoso Velasco		0.37793	19	1mo	
3	Yigit Celik		0.37948	8	1mo	
4	Jean Machado		0.37984	1	1mo	
5	Pedro R Mendonca		0.37984	13	1mo	
6	A0251441R		0.37984	6	2mo	
7	Zisheng Huang		0.37984	6	2mo	
8	Chong Zhen Jie		0.37984	2	2mo	
9	Kent Huang		0.37984	4	2mo	
10	Kriangkrai Tan		0.37984	1	2mo	

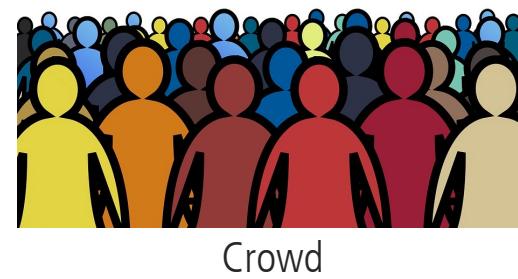
- Crowd-sourcing the Analytics: let the crowd come up with best machine learning model for you!
- Will discuss incentive issues with these tournaments and how to address those issues.

Human versus Machine Predictions

- With historical data for similar* situations:
- Without appropriate historical data:
- Or, potentially better:



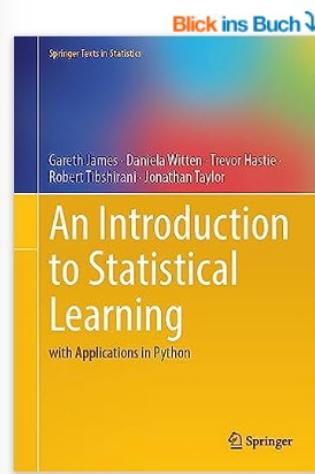
*will make this more clear



Forecasting Game

Go to www.menti.com and use code 8569 3666

Literature Pointer for First Part of this Class



Blick ins Buch ↓

An Introduction to Statistical Learning: with Applications in Python (Springer Texts in Statistics)



Gebundene Ausgabe – 1. Juli 2023

Englisch Ausgabe | von Gareth James (Autor), Daniela Witten (Autor), Trevor Hastie (Autor), Robert Tibshirani (Autor), & 1 mehr

5,0 ★★★★★ 1 Sternebewertung

[Alle Formate und Editionen anzeigen](#)

Gebundenes Buch

106,99 € prime

8 Neu ab 103,30 €

Taschenbuch

74,89 € prime

1 Neu ab 74,89 €

An Introduction to Statistical Learning provides an accessible overview of the field of statistical learning, an essential toolset for making sense of the vast and complex data sets that have emerged in fields ranging from biology to finance, marketing, and astrophysics in the past twenty years. This book presents some of the most important modeling and prediction techniques, along with relevant applications. Topics include linear regression, classification, resampling methods, shrinkage approaches, tree-based methods, support vector machines, clustering, deep learning, survival analysis, multiple testing, and more. Color graphics and real-world examples are used to illustrate the methods presented. This book is targeted at statisticians and non-statisticians alike, who wish to use cutting-edge statistical learning techniques to analyze their data.

Four of the authors co-wrote *An Introduction to Statistical Learning, With Applications in R* (ISLR), which has become a mainstay of undergraduate and graduate classrooms worldwide, as well as an important reference book for data scientists. One of the keys to its success was that each chapter contains a tutorial on implementing the analyses and methods presented in the R scientific computing environment. However, in recent years Python has become a popular language for data science, and there has been increasing demand for a Python-based alternative to ISLR. Hence, this book (ISLP) covers the same materials as ISLR but with labs implemented in Python. These labs will be useful both for Python novices, as well as experienced users.

[^ Weniger lesen](#)

[Falsche Produktinformationen melden.](#)



Dieses Bild anzeigen

Den Autoren folgen



Trevor
Hastie

Folgen



Gareth
James

Folgen

Excellent text book and available as free download: https://hastie.su.domains/ISLP/ISLP_website.pdf

- Recommended for Class 01 (today): Section 2.1
- Recommended for Class 02: Sections 3.1, 3.2, 5.1, 7.1

What is Artificial Intelligence (AI)?



"The science and engineering of making intelligent machines"
(John McCarthy, '56)

What does this mean in reality?

What is Artificial Intelligence: Examples



AI more than just learning; in many real-world systems combined, e.g.:

- Planning and Search
- Game Theory
- Machine Learning

J I will take Computational Statistics this semester. Should I be excited?

 Whether you should be excited about taking Computational Statistics this semester depends on your interests and goals. Computational Statistics is a field that combines statistical theory with computer science and data analysis techniques to solve real-world problems. Here are some factors to consider:

1. Interest in Statistics: If you have a genuine interest in statistics, data analysis, and using mathematical and computational tools to draw insights from data, then you might find this course exciting. Computational Statistics often involves solving complex problems and working with data, which can be intellectually stimulating.
2. Practical Skills: Computational Statistics courses often teach practical skills that are highly relevant in today's data-driven world. You'll likely learn how to use statistical software, programming languages like R or Python, and various data analysis techniques



Send a message 

Free Research Preview. ChatGPT may produce inaccurate information about people, places, or facts. [ChatGPT August 3 Version](#)

What is Machine Learning (ML)?



“Machine learning is the field of study that gives computers the ability to learn without being explicitly programmed.” (Arthur Samuel, 1959)

What is Machine Learning: Example

- Classify email messages as "spam" or "no spam"



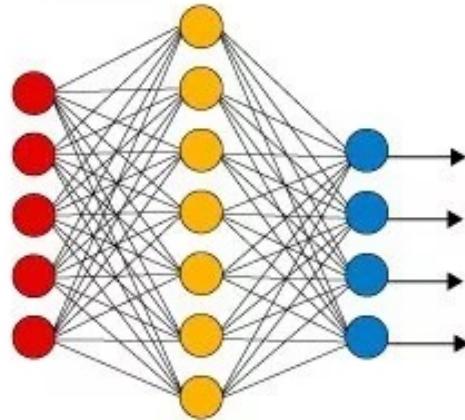
- Classical approach: manually write down rules:

IF text contains "50% off" AND email has attachment
 THEN classify as "spam" ELSE classify as "no spam"

- Machine Learning: automatic identification of rules from training data (examples)

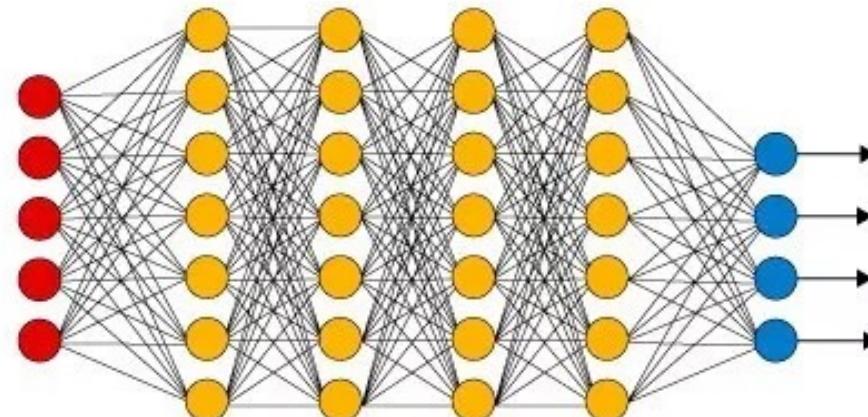
What is Deep Learning?

Simple Neural Network



● Input Layer

Deep Learning Neural Network



● Hidden Layer

● Output Layer

[Picture: <https://thedata scientist.com/what-deep-learning-is-and-isnt>]

- Sub-area of machine learning
- Employs neural network models, which are loosely inspired by the human brain
- “Deep” derives from using several hidden layers (until ~2010 believed to be unnecessary as single hidden layer proven to be “general” function approximator)
- Tremendous success in applications in recent years

What is Deep Learning: Example



German (detected) ▾

Auf einer neuen Online-Plattform bewerten die Frankfurt School of Finance & Management und der WWF Deutschland die Finanzmarktregelung in Deutschland und in der EU in Bezug auf Green Finance. Der am Donnerstag veröffentlichte „Finance fit for Paris“-Tracker“ (3fP) überprüft Gesetze und aktuelle Maßnahmen hinsichtlich ihrer Übereinstimmung mit dem Ziel des Pariser Klimaschutzabkommens, die Erderwärmung auf deutlich unter zwei Grad Celsius zu begrenzen. „Green Finance wird nach wie vor als Nische gedacht. Dabei ist essentiell, das gesamte Finanzsystem zu grünen, um es einerseits stabil gegenüber neuen Risiken zu halten, andererseits aber auch seinen Beitrag zu nationalen und internationale Klimaziel zu gewährleisten“, sagt Matthias Kopp, Leiter Sustainable Finance beim WWF Deutschland.

↑ Translate document

799/5000

Linguee ▾

DeepL Pro

Blog

Info ▾

👤 ▾

Translate into English ▾

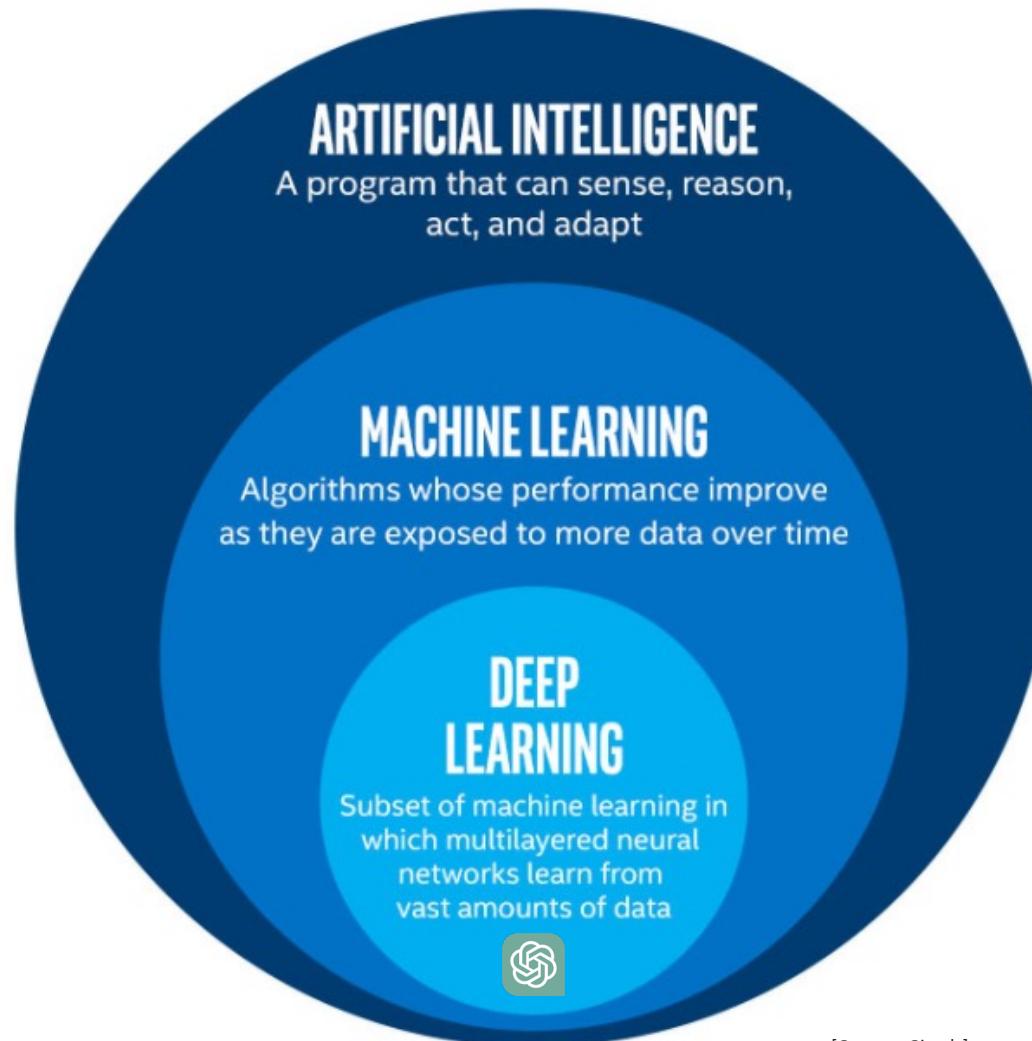
On a new online platform, the Frankfurt School of Finance & Management and WWF Germany assess financial market regulation in Germany and the EU with regard to green finance. The "Finance fit for Paris"-Tracker" (3fP), published on Thursday, reviews laws and current measures with regard to their compliance with the goal of the Paris Climate Protection Agreement to limit global warming to well below two degrees Celsius. "Green finance is still seen as a niche market. It is essential to green the entire financial system in order to keep it stable against new risks while at the same time ensuring its contribution to national and international climate targets," says Matthias Kopp, Head of Sustainable Finance at WWF Germany.



English version of fs.de

With a new online platform, Frankfurt School of Finance & Management and WWF Germany assess financial market regulation in Germany and the European Union with a focus on Green Finance. The finance fit for Paris tracker, made available on Thursday, evaluates laws and current measures against the imperative of the <2-degree target as anchored in the Paris Agreement. "Green Finance is still thought of as a niche topic. But it is essential to green the whole finance system to keep it stable against new risks, but also to ensure its contribution to national and international climate goals", says Matthias Kopp, Head of Sustainable Finance at WWF Germany.

Artificial Intelligence and Machine Learning



[Seema Singh]

Main Areas within Machine Learning

Supervised Learning: training data given to us contains “correct” target.

Regression: real-valued target, i.e. $y \in \mathbb{R}$

- Amount of coffee sold in a particular area
- Growth of the US economy in 2021
- A person’s height

Classification: categorial target, i.e. $y \in \{1, \dots, m\}$

- Emails: “spam” / “no spam”
- Credit card transactions: “regular” / “fraud”
- Image recognition: “dog” / “person” / “house”

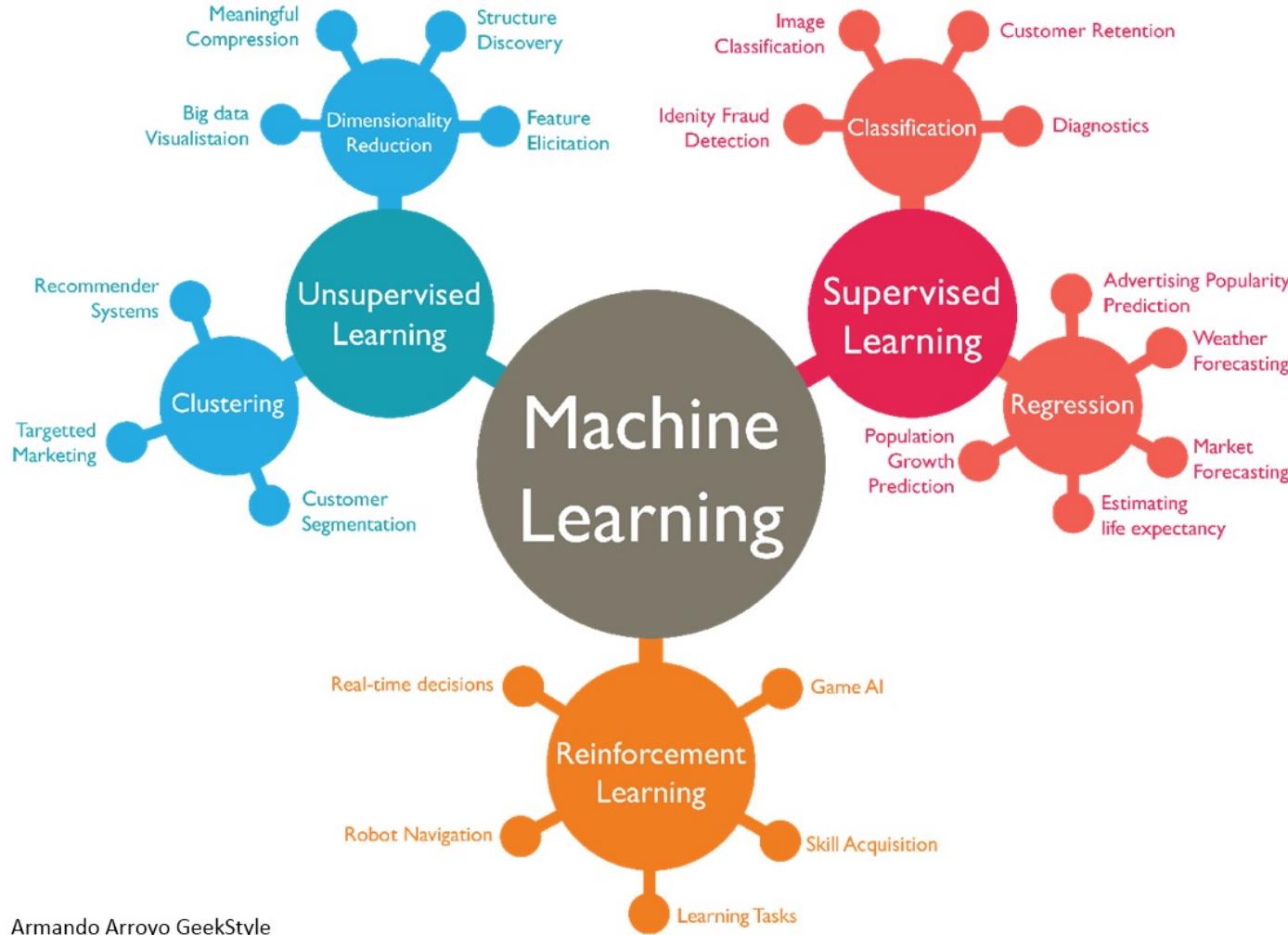
Unsupervised Learning: training data contains no targets.

Dimensionality reduction

Clustering: group “similar” data points, e.g., google photos grouping similar pictures

Reinforcement Learning: learning from trial and error with feedback from environment.

Main Areas within Machine Learning (cont.)



Supervised Machine Learning: Example

Image classification Easiest classes

red fox (100) hen-of-the-woods (100) ibex (100) goldfinch (100) flat-coated retriever (100)



tiger (100)



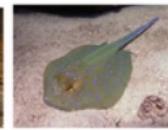
hamster (100)



porcupine (100)



stingray (100)



Blenheim spaniel (100)



Hardest classes

muzzle (71) hatchet (68) water bottle (68) velvet (68)



loupe (66)



hook (66)



spotlight (66)



ladle (65)



restaurant (64)

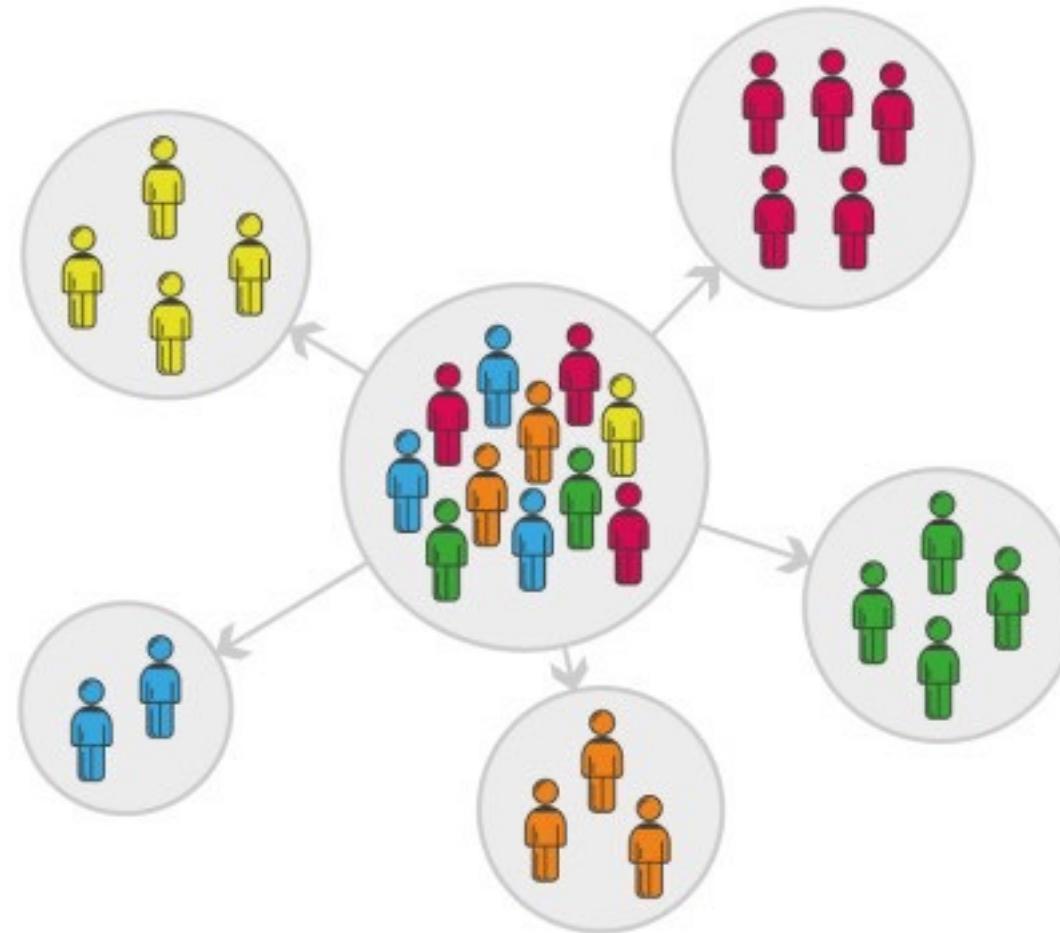


letter opener (59)



Unsupervised Machine Learning: Example

Customer segmentation

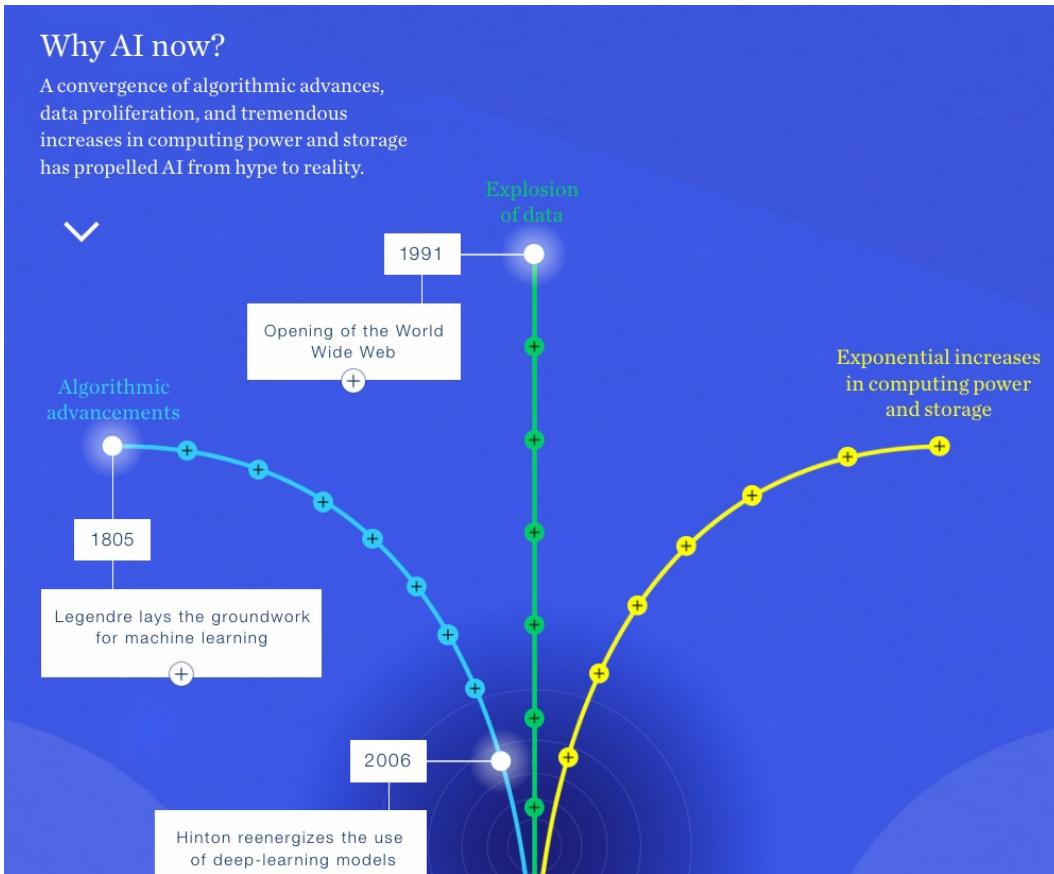


[Project A, 2015]

Reinforcement Learning: Example

- <https://www.youtube.com/watch?v=V1eYniJ0Rnk>

Why now?



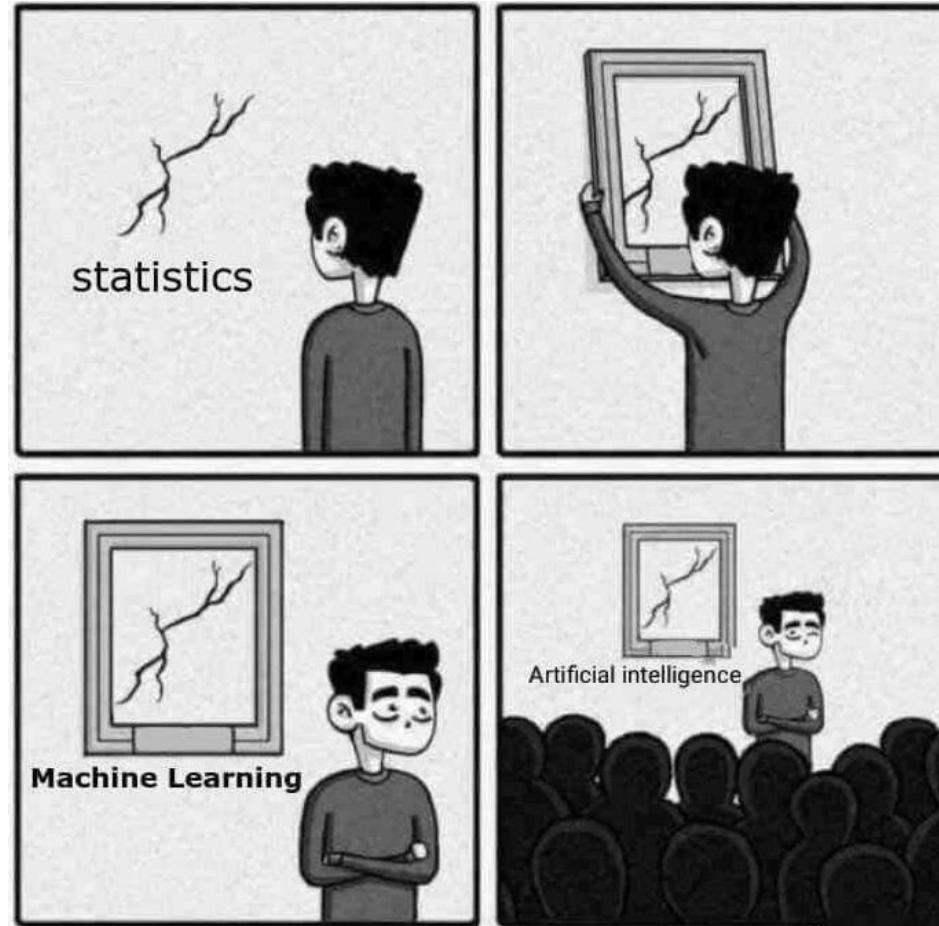
Main drivers of recent ML successes:

- More (labeled!) data
- More computing power
- New algorithms (note though: basic neural network structure has been around at least since the 1990s)

[McKinsey, 2018]

Machine Learning vs Statistics

The cynical view



[<https://i.imgur.com/pWiyz4Z.jpg>]

(not quite right as we have seen but still contains some truth)

Machine Learning vs Classical Statistics

- The two fields are converging: machine learning and statistical learning theory sometimes used interchangeably
- However, (supervised) **ML focuses on prediction**
 - Historically cares less about causality / explanation / interpretation / statistical significance
 - ML lets algorithm figure out which variables (and their interactions, e.g., $x_1 * x_2$) should be included
 - Often very much non-linear (don't need to be able to have explanation for why learned function has the form it has as long as it's accurate)
 - Evaluation on unseen data / cross validation
- ML developed out of computer science; hence: more computational
 - often: "bigger" data
 - often more "engineer-y" (and, say, less hypothesis testing)
- ML arguably more concerned with performance / accuracy than elegant theory (still a lot of research needed as to why deep learning works as well as it does)

Dictionary: Statistics \leftrightarrow Machine Learning

These terms are used inter-changeably:

- Regression coefficients = weights = parameters
- Residual variance = noise variance
- Inputs = predictors = features = explanatory variables = regressors = IVs = covariates
- Outputs = outcomes = targets = response variables = DVs = labels
- Training = learning = fitting (kind of; depends on context)

Supervised Machine Learning: Regression

Want to learn function $f(\mathbf{x})$ using samples (x_i, y_i) coming from

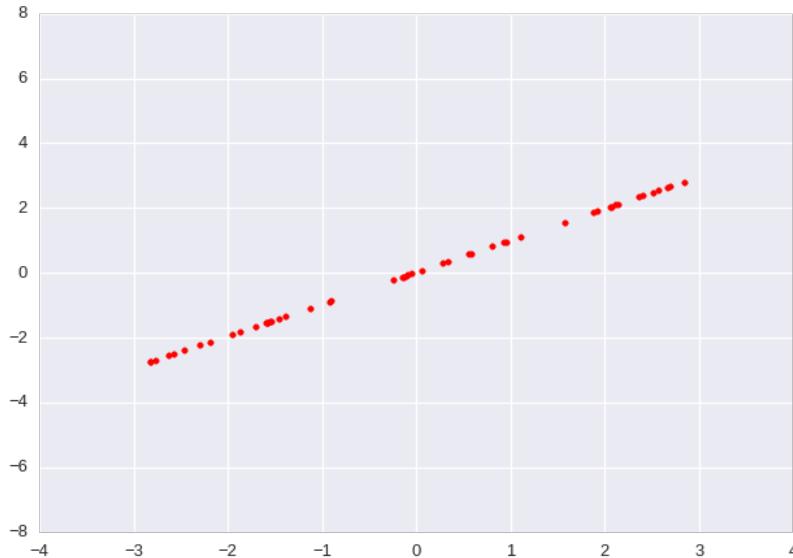
$$y = f(\mathbf{x}) + \epsilon$$

where ϵ is a noise term with $\mathbf{E}[\epsilon] = 0$.

Final objective: **accurate predictions for unseen points!**

Generative Model of Linear Regression

Data is generated from a linear model with unknown parameters but corrupted with Gaussian noise:

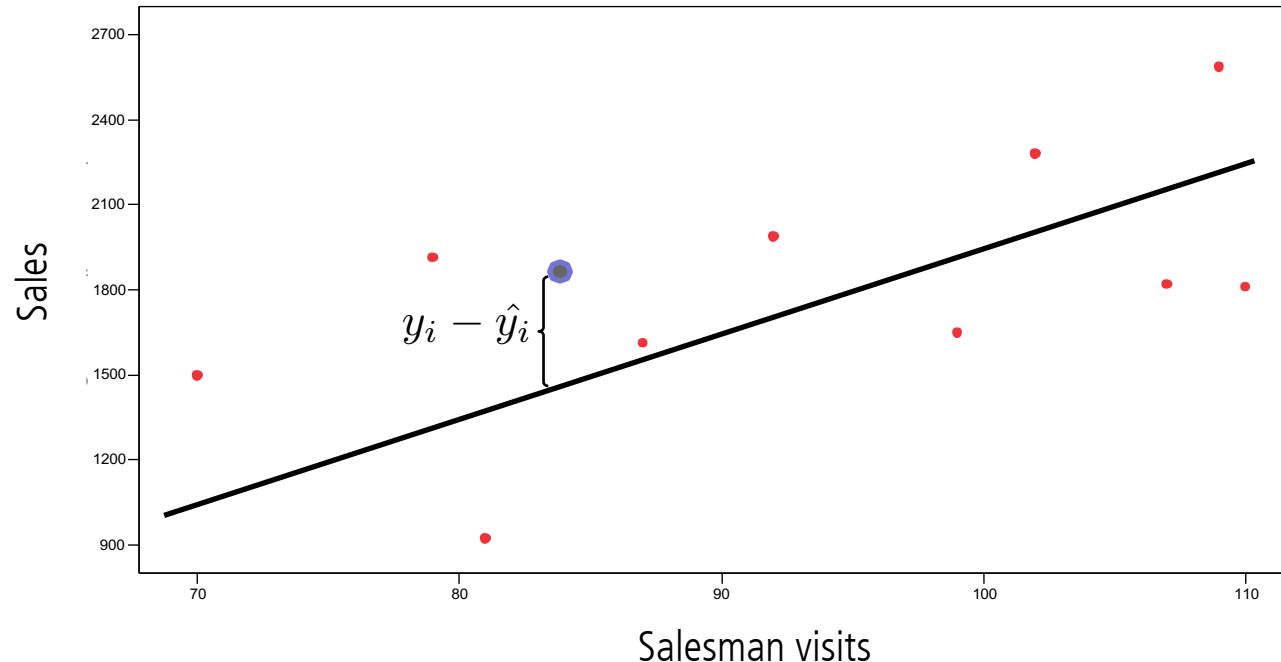


[Parkes and Rush, 2017]

Assume **data** comes from linear model, but we get to see only **noisy version**.

Linear Regression for Prediction

- After training a linear regression model, can use it to make predictions:



- How will we measure accuracy of the prediction (goodness of fit)?

Typically: Mean Squared Error (MSE), i.e. $\frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2$

Python Practice

Two standard libraries:

```
import numpy as np
import matplotlib.pyplot as plt
```

Let's create two simple Numpy arrays:

```
x = np.array([1.0, 2.0, 3.0, 4.0, 5.0, 6.0, 7.0, 2.5])
y = np.array([7.6, 5.0, 4.0, 3.5, 4.0, 5.0, 7.0, 4.0])
```

Let's plot this:

```
plt.scatter(x,y)
plt.show()
```

What would be a good fit for this data?

Fitting a Linear Regression Model in Python

First, we need a new library:

```
from sklearn.linear_model import LinearRegression
```

Create a linear regression object using

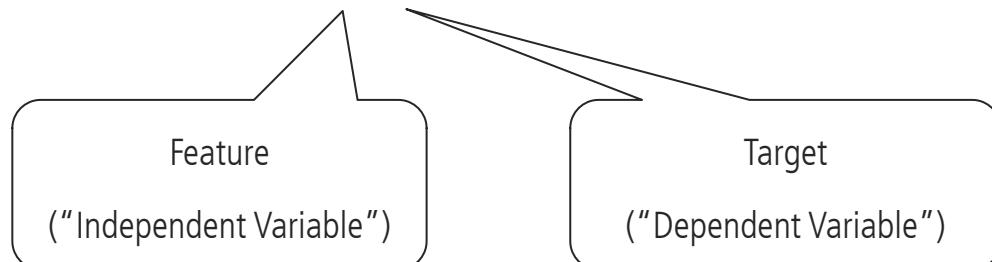
```
polyreg = LinearRegression()
```

Fitting a model requires a 2D feature array and reshape(-1,1) creates a 2D array from a 1D array:

```
X = x.reshape(-1, 1)
```

Fit the linear regression model using

```
polyreg.fit(X, y)
```



Predictions in Python

Use `predict` on your trained model to make predictions on unseen data, e.g., if you fitted your model using

```
polyreg.fit(X, y)
```

you can predict (unknown) value `y` for `x = 10` running

```
new_X = np.array([[10]])  
polyreg.predict(new_X)
```

Remember: 2D array

To predict for several values:

```
new_X = np.array([[-10], [15], [50]])  
polyreg.predict(new_X)
```

For a whole range of values, use `arange` or `linspace`:

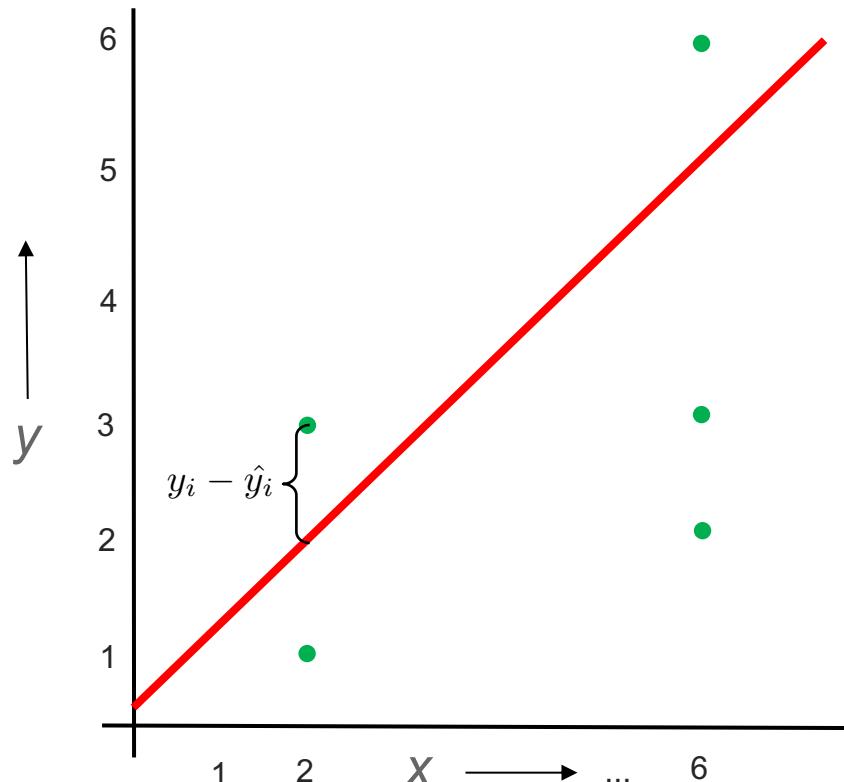
```
X_seq = np.linspace(0, 10, 100).reshape(-1, 1)  
polyreg.predict(X_seq)
```

Plotting the fitted model:

```
plt.plot(X_seq, polyreg.predict(X_seq), color="black")
```

Exercise: Mean Squared Error (MSE)

Measure goodness of fit using MSE: $\frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2$

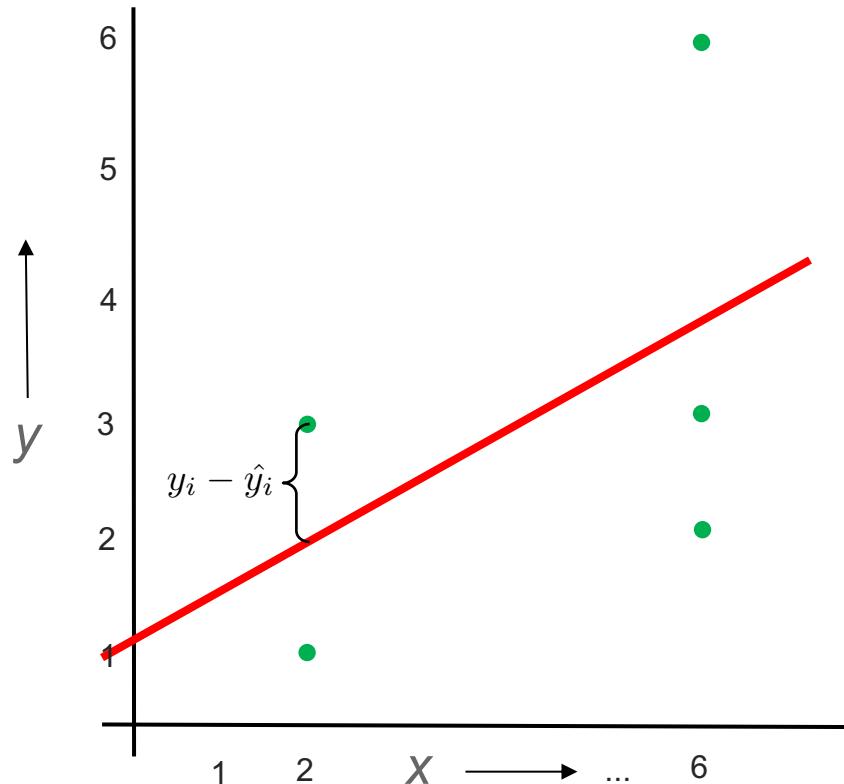


Is the red line the optimal fit to the data? Why? Why not?

What would happen to the optimal fit if we used mean *absolute* error $\frac{1}{n} \sum_{i=1}^n |y_i - \hat{y}_i|$ instead of MSE?

Solution: Mean Squared Error (MSE)

Measure goodness of fit using MSE: $\frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2$



The optimal fit does go through point (2,2) but not through (6, 5). Let \hat{y} be the estimated value at $x=6$ and minimize the squared error:

$$\begin{aligned} & \arg \min_{\hat{y}} \frac{1}{3} ((6 - \hat{y})^2 + (3 - \hat{y})^2 + (2 - \hat{y})^2) \\ & = \arg \min_{\hat{y}} 3\hat{y}^2 - 22\hat{y} + 49 \end{aligned}$$

Take derivative and set to 0:

$$6\hat{y} - 22 := 0 \Rightarrow \hat{y} = \frac{22}{6} = \frac{11}{3} = 3.667$$

That is, the optimal fit goes through the average value at $x = 6$.

Using mean absolute error, it would go through the median at $x = 6$.

Prof. Dr. Jens Witkowski

Assistant Professor of Computer Science and Management

Frankfurt School of Finance & Management
Adickesallee 32-34
60322 Frankfurt am Main
Germany

Email: j.witkowski@fs.de
Tel.: +49 (0) 69-154008-875
Fax: +49 (0) 69-154008-4875

Class 02

Overfitting and Model Selection



Computational Statistics

Prof. Dr. Jens Witkowski

September 7, 2023

Agenda

- Session 1
 - Recap: Linear Regression
 - Non-Linear Features
 - Python Exercise: Non-Linear Features
- Session 2
 - Overfitting
 - Cross validation and out-of-sample predictions

Recap: Linear Regression

Want to learn function $f(\mathbf{x})$ using samples (x_i, y_i) coming from

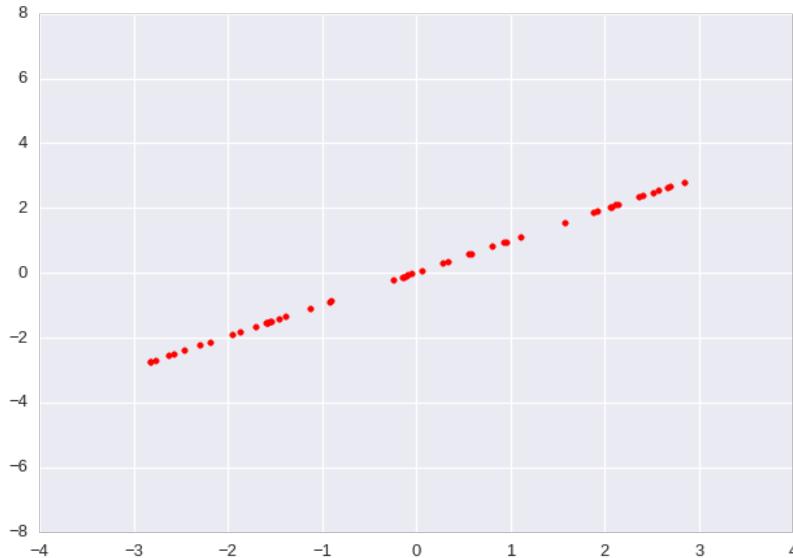
$$y = f(\mathbf{x}) + \epsilon$$

where ϵ is a noise term with $\mathbf{E}[\epsilon] = 0$ (typically assumed Gaussian).

Final objective: **accurate predictions for unseen points!**

Generative Model of Linear Regression

Data is generated from a linear model with unknown parameters but corrupted with Gaussian noise:



[Parkes and Rush, 2017]

Assume **data** comes from linear model, but we see **noisy version**.

Simple Linear Regression (from last time)

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression

x = np.array([1.0, 2.0, 3.0, 4.0, 5.0, 6.0, 7.0, 2.5])
y = np.array([7.6, 5.0, 4.0, 3.5, 4.0, 5.0, 7.0, 4.0])

X = x.reshape(-1, 1)
plt.scatter(X,y)

polyreg = LinearRegression()
polyreg.fit(X,y)

X_seq = np.linspace(0,10,100).reshape(-1,1)
print(polyreg.predict(X_seq))

plt.plot(X_seq, polyreg.predict(X_seq), color="black")
plt.show()
```

Python Exercise: MSE Function

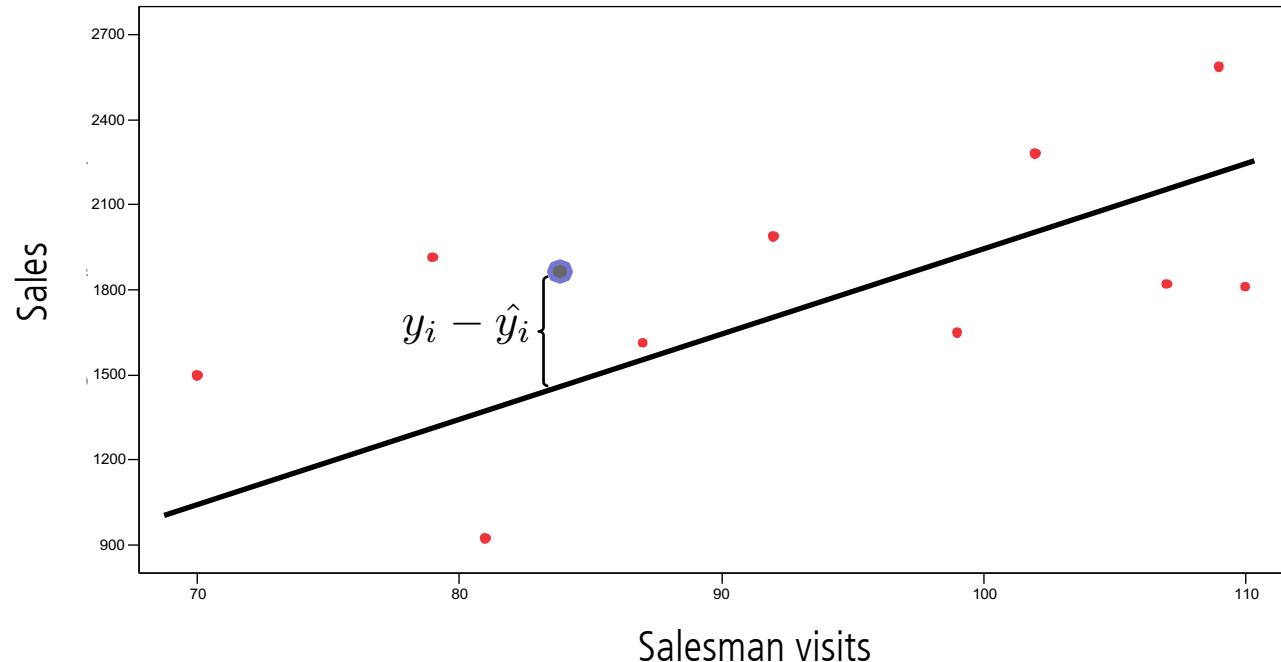
Write a function `mse` that returns the mean squared error and takes as input the `model`, the (original) feature matrix `X`, and the target vector `y`.

Solution:

```
def mse(model, X, y):  
    return ((y-model.predict(X))**2).mean()
```

Linear Regression for Prediction

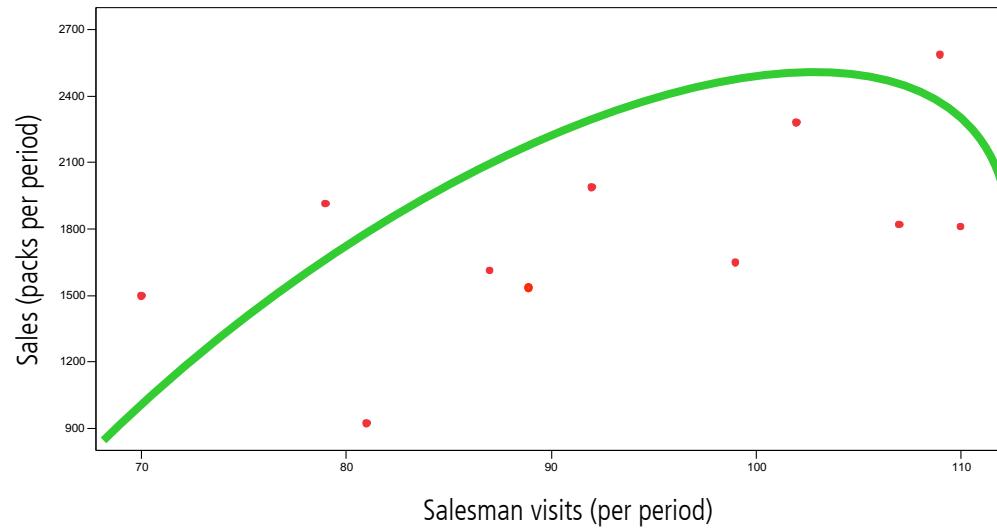
- After training a linear regression model, can use it to make predictions:



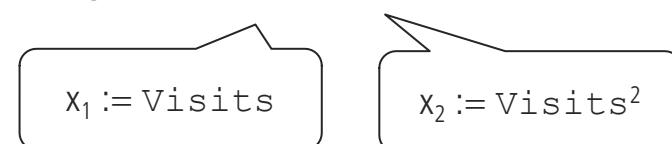
- But what if our data is not linear?

Non-linear Transformations

- A non-linear function (e.g., a quadratic function) might be a better fit to our data than a linear function



- Can do this by creating new features that are non-linear transformations of existing features, e.g. having `Visits` and `Visits2`
- The model we are fitting is *still a linear model* (in the features), e.g. $b_0 + b_1 x_1 + b_2 x_2$



- Can think of this as creating a new variable/column with `Visits2` before fitting linear model

Adding Transformed Features

- `np.append(a, b)` appends array `b` to array `a`:

```
np.append(np.array([[1], [2]]), np.array([[3], [4]]), axis = 0)
```

- Add a quadratic feature to the original feature space:

```
x_transformed = np.append(X, X**2, axis = 1)
```

- Fit linear model with original and one transformed feature (quadratic):

```
polyreg.fit(X_transformed, y)
```

- For input to `predict` method, also need transformed sequence:

```
X_seq_transformed = np.append(X_seq, X_seq**2, axis = 1)
```

- Linear model with two transformed features (quadratic and square root):

```
plt.plot(X_seq, polyreg.predict(X_seq_transformed), color="black")
```

Why both `X_seq` and `X_seq_transformed` in here?

Exercise

Try fitting a cubic regression, i.e., $b_0 + b_1 x + b_2 x^2 + b_3 x^3$.

(One) solution: Add

```
X_transformed = np.append(X_transformed, X**3, axis = 1)
```

and

```
X_seq_transformed = np.append(X_seq_transformed, X_seq**3, axis = 1)
```

Polynomial Regression with scikit-learn

- Becomes cumbersome and hard to read for larger polynomials; instead can use scikit tools:

```
from sklearn.preprocessing import PolynomialFeatures  
from sklearn.pipeline import make_pipeline
```

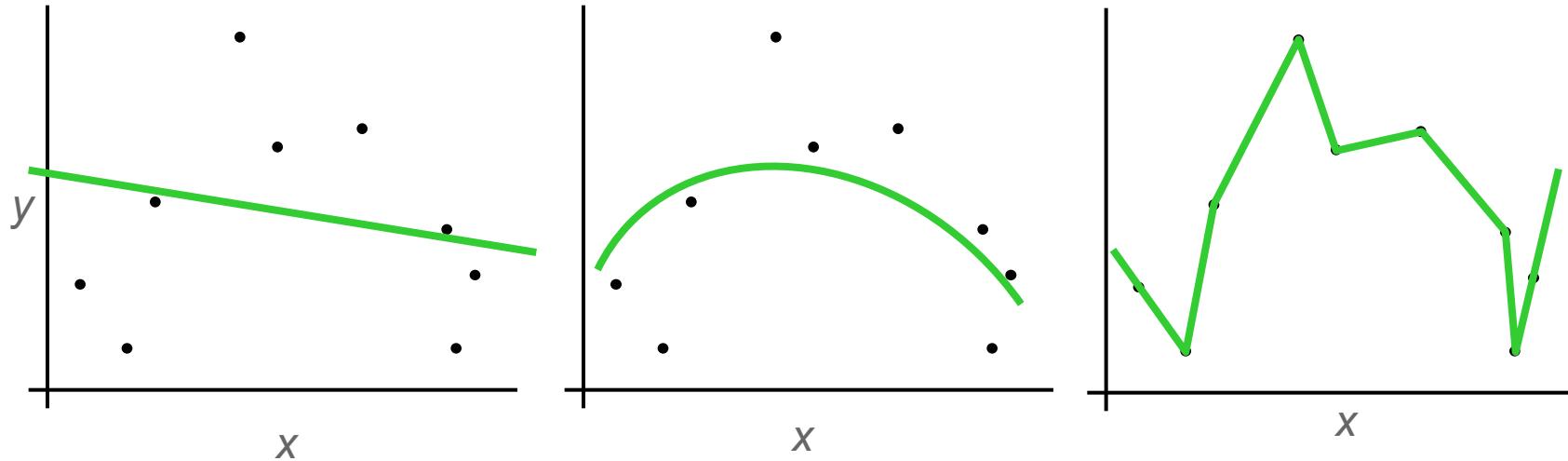
- Replace the corresponding lines with these here:

```
polyreg=make_pipeline(PolynomialFeatures(degree = 3), LinearRegression())  
plt.plot(X_seq, polyreg.predict(X_seq), color="black")
```

Play around with different values for degree. Which model has the lowest MSE and why?

Overfitting

Arguably the **most important challenge in Machine Learning!**



Why not choose model with best fit to data?

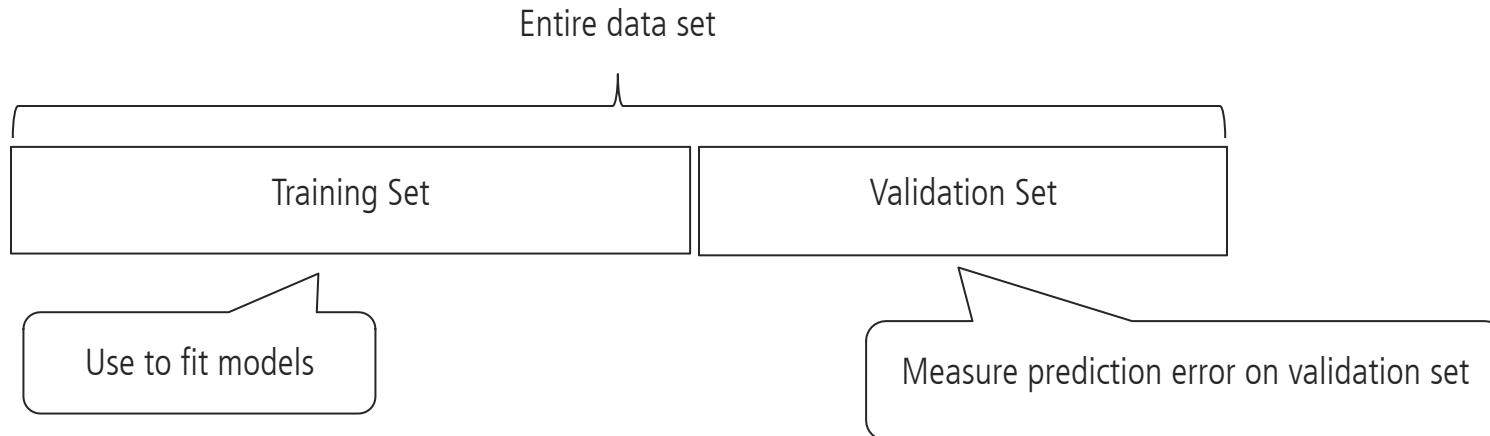
With higher complexity, at some point fitting to the noise ϵ instead of the function $f(\mathbf{x})$!

"How well is the model going to predict the target on *unseen data* drawn from the same distribution?"
 (i.e., on data that was not used for training the model)

The Validation Set Method (also: Holdout Method)

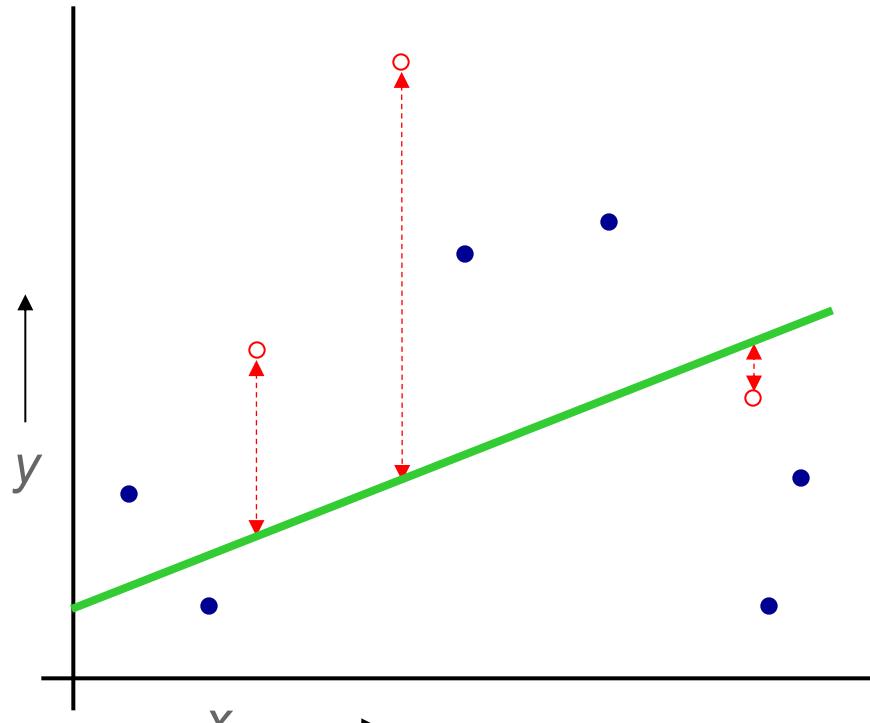
Want to *generalize* well to unseen data!

1. Randomly divide your data into a training set and a validation set:



2. Train different regression models with varying model complexity on training set (and compute *training error*)
3. Measure models' prediction errors on validation set (*validation error*)
4. Choose the model with the best validation error.

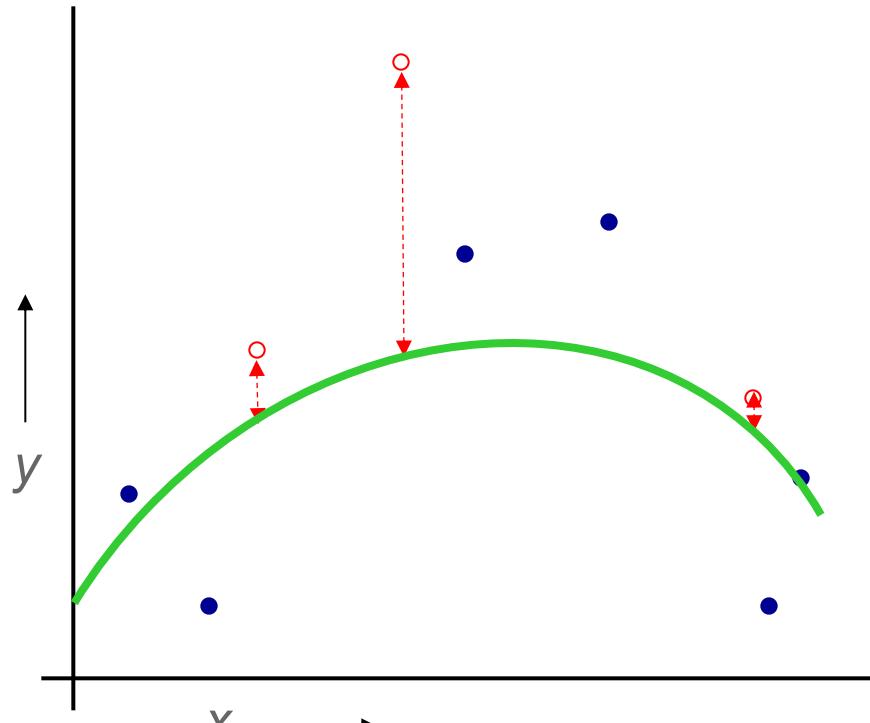
The Validation Set Method



Mean Squared Error = 2.4

1. Randomly choose (for example) 30% of the data to be in a **validation set**
2. The remainder is a **training set**
3. Train your regression models on training set
4. Evaluate model performance with the validation set

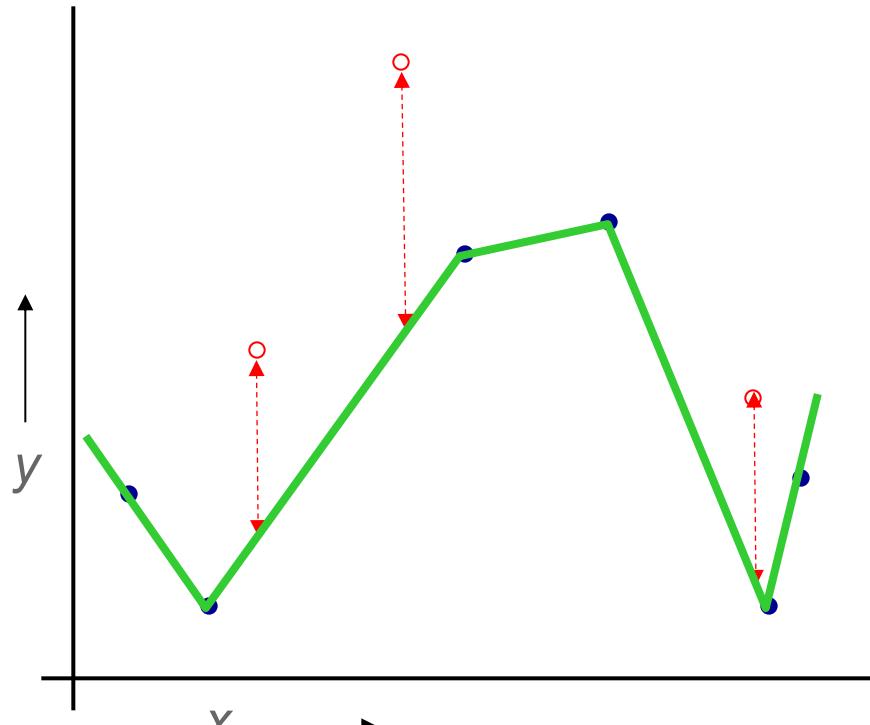
The Validation Set Method



Mean Squared Error = 0.9

1. Randomly choose (for example) 30% of the data to be in a **validation set**
2. The remainder is a **training set**
3. Train your regression models on training set
4. Evaluate model performance with the validation set

The Validation Set Method



Mean Squared Error = 2.2

1. Randomly choose (for example) 30% of the data to be in a **validation set**
2. The remainder is a **training set**
3. Train your regression models on training set
4. Evaluate model performance with the validation set

Validation Set Method: Pros and Cons

Good news:

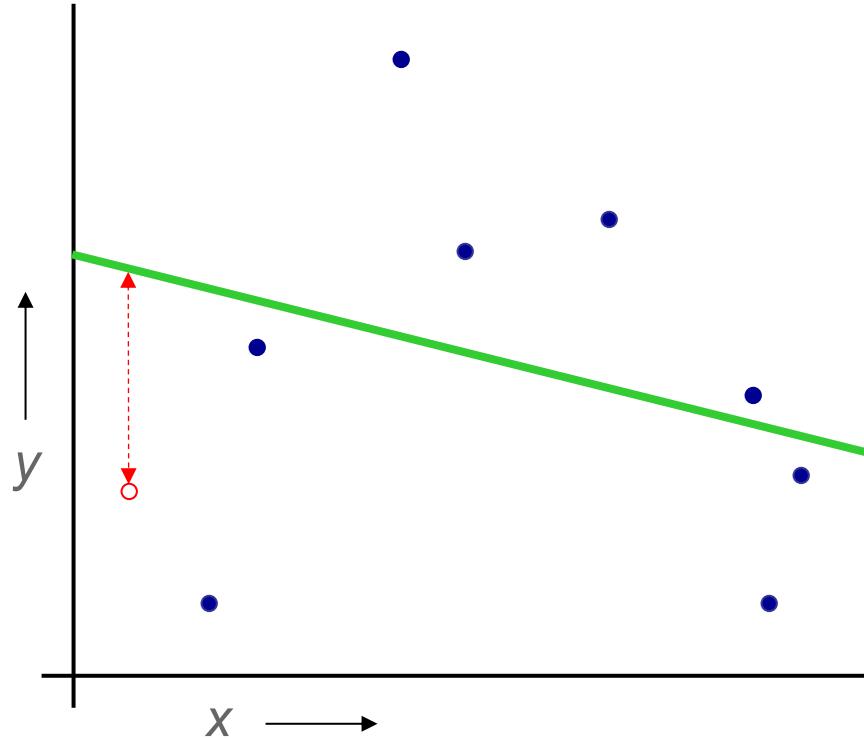
- Very simple

Bad news:

- Wastes data: only uses 70% (in example) of data for training
- Validation set sample unreliable (high variance): without much data, estimates for validation set error highly dependent on randomness of validation set sample

Question: How can we use all data but still predict "out of sample?"

LOOCV (Leave-one-out Cross Validation)

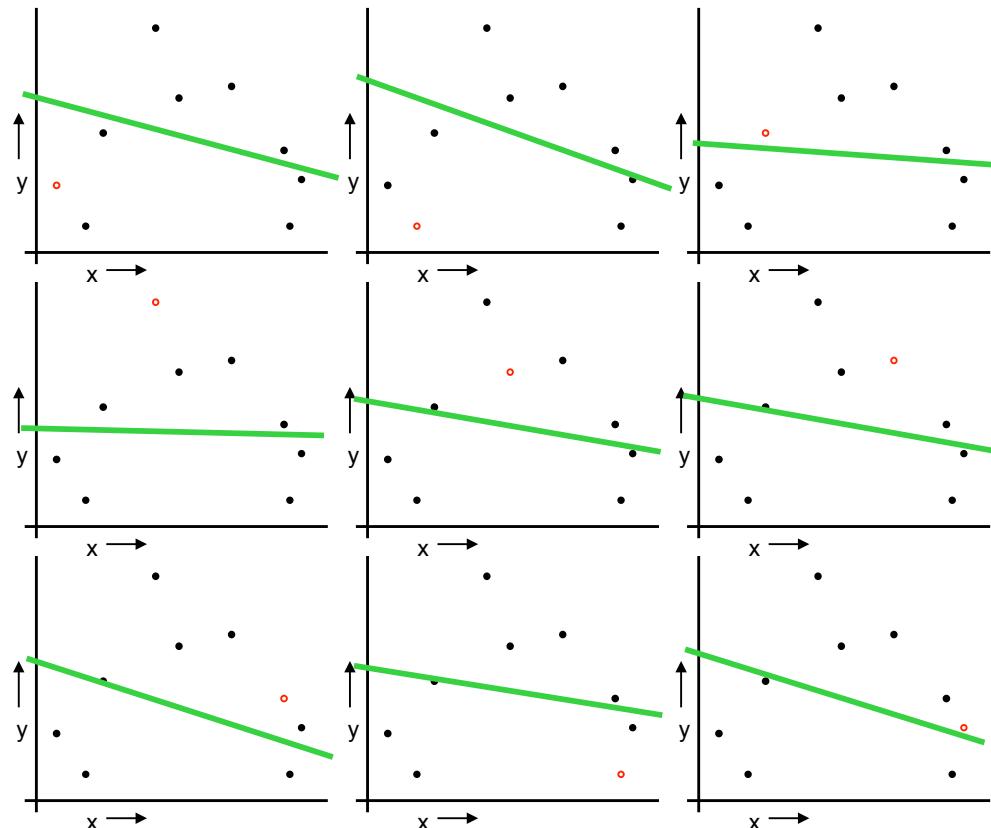


For $i = 1$ to n :

1. Let (x_i, y_i) be the i^{th} data point
2. Temporarily remove (x_i, y_i) from data set
3. Train model on remaining $n-1$ data points
4. Note error of model for (x_i, y_i)

When every point has been left out, report mean error.

LOOCV (Leave-one-out Cross Validation)



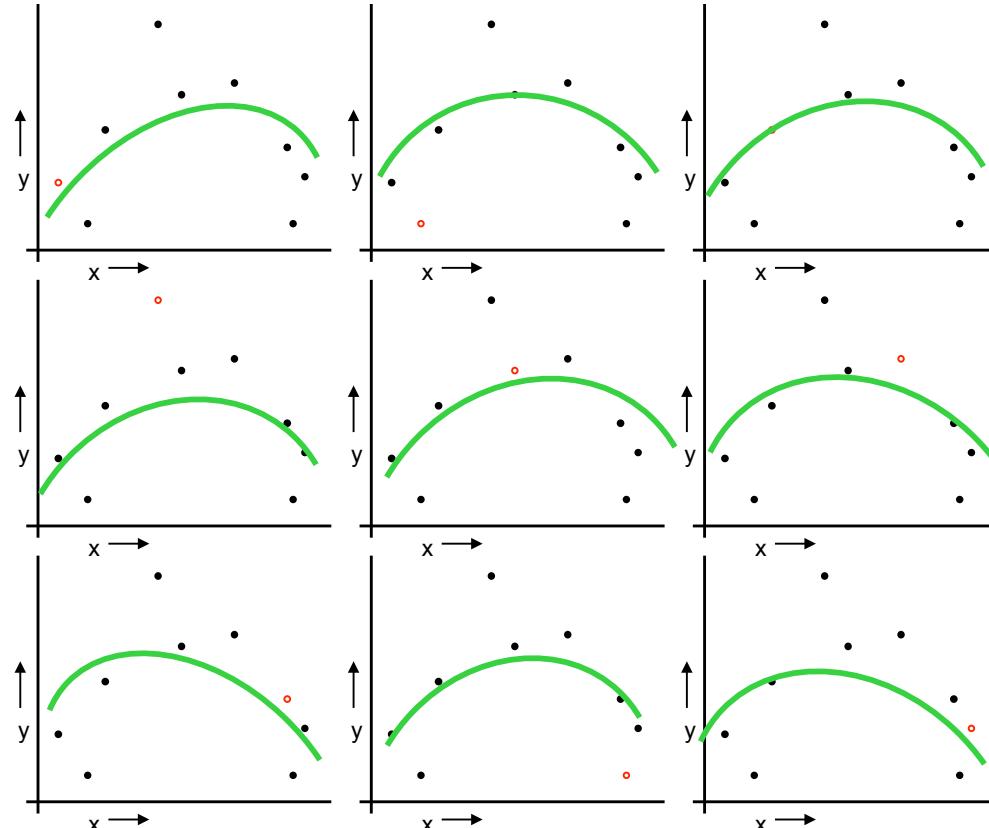
For $i = 1$ to n :

1. Let (x_i, y_i) be the i^{th} data point
2. Temporarily remove (x_i, y_i) from data set
3. Train model on remaining $n-1$ data points
4. Note error of model for (x_i, y_i)

When every point has been left out, report mean error.

$$\text{MSE}_{\text{LOOCV}} = 2.12$$

LOOCV (Leave-one-out Cross Validation)



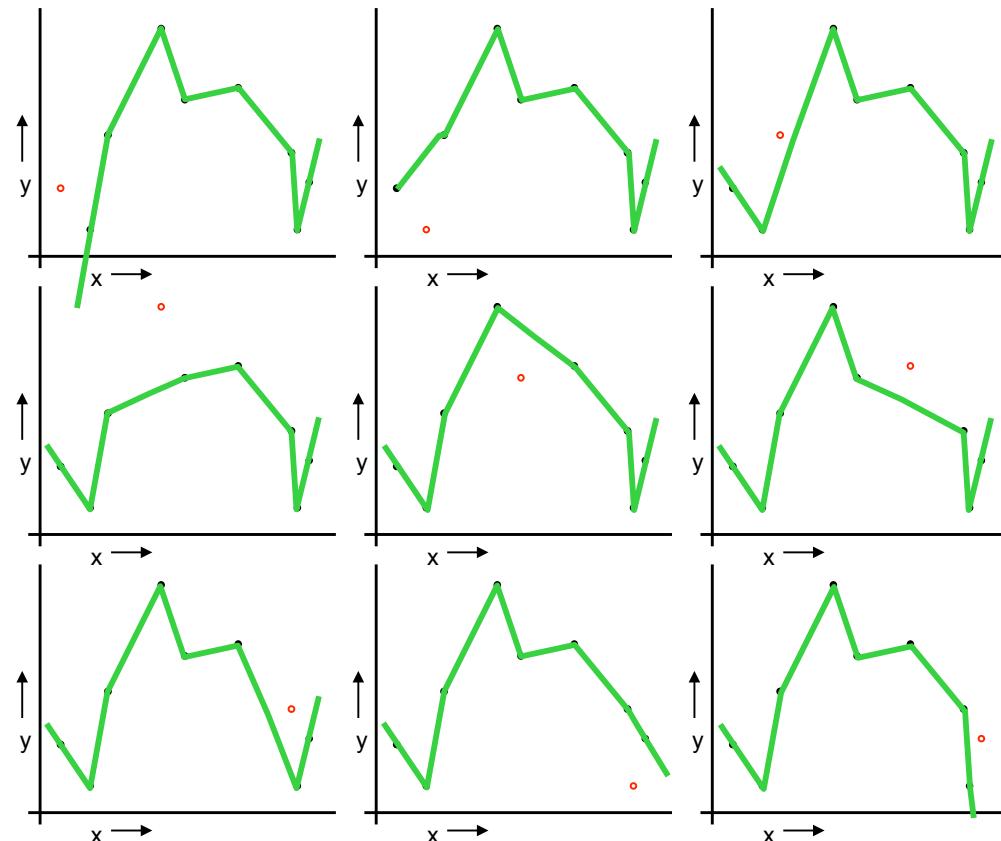
For $i = 1$ to n :

1. Let (x_i, y_i) be the i^{th} data point
2. Temporarily remove (x_i, y_i) from data set
3. Train model on remaining $n-1$ data points
4. Note error of model for (x_i, y_i)

When every point has been left out, report mean error.

$$\text{MSE}_{\text{LOOCV}} = 0.962$$

LOOCV (Leave-one-out Cross Validation)



For $i = 1$ to n :

1. Let (x_i, y_i) be the i^{th} data point
2. Temporarily remove (x_i, y_i) from data set
3. Train model on remaining $n-1$ data points
4. Note error of model for (x_i, y_i)

When every point has been left out, report mean error.

$$\text{MSE}_{\text{LOOCV}} = 3.33$$

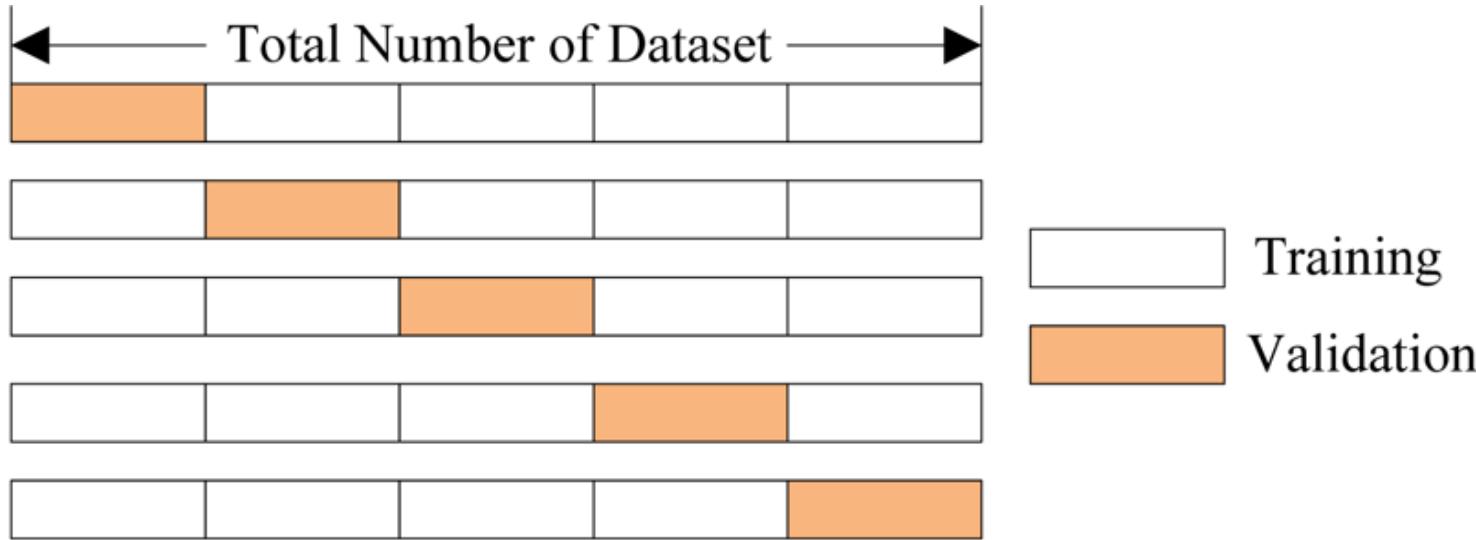
Which kind of Cross Validation?

	Downside	Upside
Validation Set Method	Unreliable estimate of future performance	Cheap
LOOCV	Expensive	Doesn't waste data

Can we get the best of both worlds?

k-Fold Cross Validation

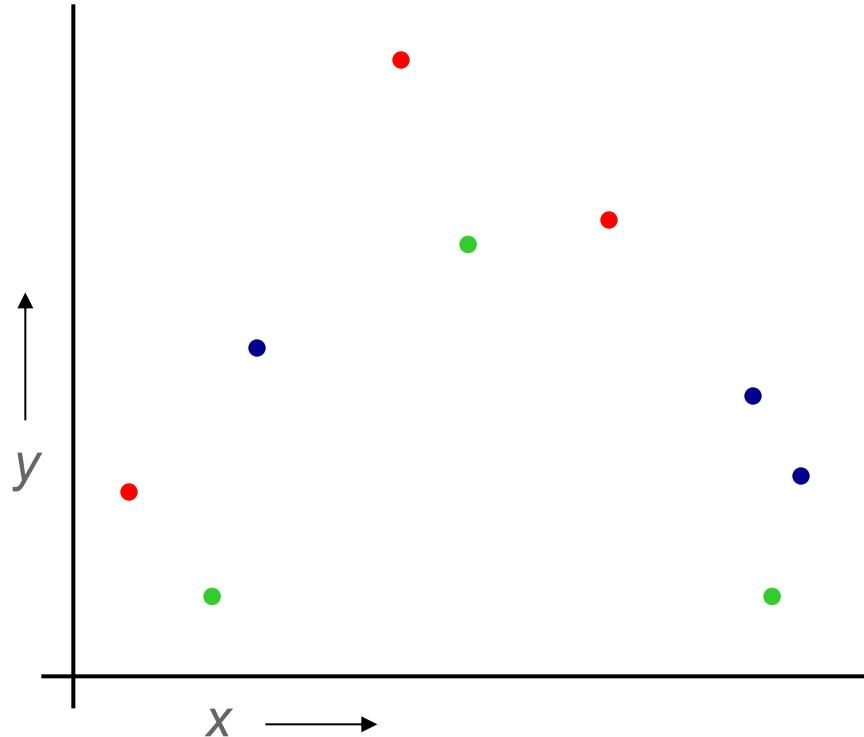
1. Randomly divide your data into k (here: 5) subsamples:



2. Repeat k times:
 - i. Train model using data from the training set.
 - ii. Measure model's prediction error on validation set.
3. Choose the model class with the best average validation error.

k-Fold Cross Validation

Randomly break data set into k partitions (in our example: k=3 with colors red, green, and blue)

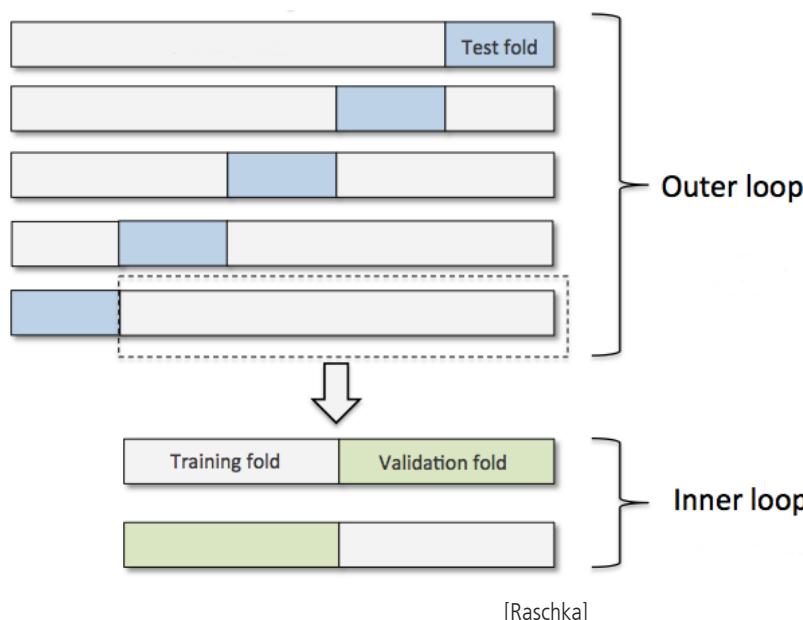


1. Train model on all points except for the red partition. Find test set error (MSE) on red points.
2. Train model on all points except for the green partition. Find test set error (MSE) on green points.
3. Train model on all points except for the blue partition. Find test set error (MSE) on blue points.
4. Report mean of MSE (average of averages).

Caveat 1: Reporting the Generalization Error

Is validation error too optimistic, too pessimistic, or just right?

- Validation error under-estimates generalization error of the selected model (i.e., validation error too optimistic)
- Reason: We have used the data on which the error is reported to select the best model
- To get unbiased estimate of generalization error, need to use three different sets: training set, validation set, and test set, where
 1. You fit the models on training set
 2. Select best-performing model on validation set
 3. Report error of selected model on test set
- Can use “nested k-fold cross validation:”



Caveat 2: Temporal Data

- Need to be careful with temporal data when selecting validation/test sets.
- In order to accurately simulate forecasting reality, forecaster must withhold all data about events that occur chronologically after the events used for fitting the model.
- Use “sliding window” (in graphic, “training set” consists of training and validation set):

Split 1:	Training set	Test set			
Split 2:		Training set	Test set		
Split 3:			Training set	Test set	
Split 4:				Training set	Test set
	Time 1	Time 2	Time 3	Time 4	Time 5

- Or “expanding window:”

Split 1:	Training set	Test set			
Split 2:		Training set	Test set		
Split 3:			Training set	Test set	
Split 4:				Training set	Test set
	Time 1	Time 2	Time 3	Time 4	Time 5

[A.G.]

A Note on "Randomness"

- Remember that we can generate "random" numbers using Numpy, e.g.:

```
np.random.randint(3, 18, 5)
>>> array([11, 12, 6, 11, 4])
```

Challenge for comparing code when we everybody gets different "random" draw (and for finding bugs in code)!

- "Fun" fact: computers can't generate truly random numbers but only produce pseudo-random numbers (typically initialized based on internal clock, which is slightly different for different machines)!
- Can set initialization of pseudo-random number-generating process ourselves by setting a seed:

```
np.random.seed(42)
np.random.randint(3, 18, 5)
>>> array([ 9,  6, 15, 17, 13])
np.random.randint(3, 18, 5)
>>> array([10, 15, 7, 9, 12])
np.random.seed(42)
np.random.randint(3, 18, 5)
>>> array([ 9,  6, 15, 17, 13])
np.random.randint(3, 18, 5)
>>> array([10, 15, 7, 9, 12])
```

Let's us compare results despite "random" sampling!

scikit learn: train_test_split

- Can divide dataset into training and validation/test set using scikit learn tools:

```
from sklearn.model_selection import train_test_split
X_train, X_validate, y_train, y_validate = train_test_split(X, y, test_size=0.2)
```

X_train contains 80% of rows from X, X_validate contains remaining 20% of rows from X (analogous for y).

- To simplify comparing results, we can set a seed (here given as randomstate):

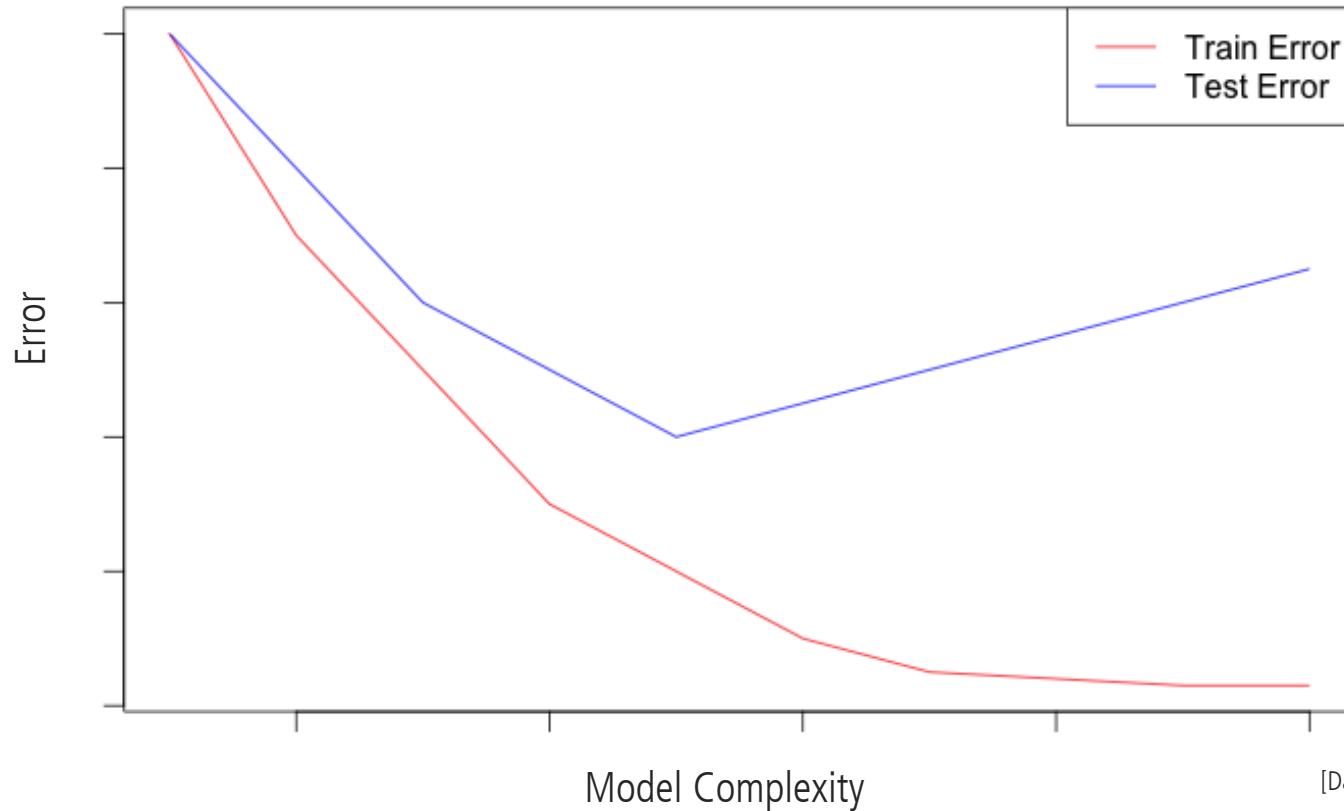
```
X_train, X_validate, y_train, y_validate = train_test_split(X,
y, test_size=0.2, random_state = 42)
```

(leads to everyone having the same train/validation split)

Python Practice

1. Load data set `Wage.csv` as `wage` into your Python environment. Split the data into training (90%) and validation (10%) sets and set a seed of 10. Subset to only the age (as feature) and wage (as target) of a person. Plot the training set in black and the validation set in grey.
2. Fit and plot (in blue) the linear function predicting `wage` from `age` using only the training set (call it `model1`) and compute the MSE of `model1` on the validation set.
3. Compute the validation MSE of the 5-degree polynomial function fitted on the training set (call it `model5`). Plot it in red.
4. Compute the validation MSE of the 12-degree polynomial function fitted on the training set (call it `model12`). Plot it in green.
5. Rerun everything with different seeds (20 and 30). Which model does seem best "out of sample?"
6. What would you expect to happen to the (out-of-sample) optimal degree if we only had 300 instead of 3000 observations / data points? (larger? smaller? same?) (If you are fast, you could try this out!)
7. What would you expect for the training MSE for all three models and different seeds? (If you are fast, you could try this out!)

Typical Situation: Training vs Test Error



[Daniel Nee]

- Training error decreases with higher model complexity (e.g., higher-order polynomials)
- Test/validation error first decreases (at slower speed) and then increases (that's the point where we overfit)
- Training error typically smaller than test/validation error.

Credits

Some of the figures in this presentation are taken from Andrew W. Moore
(<http://www.cs.cmu.edu/~awm/tutorials.html>) with permission from the author.

Prof. Dr. Jens Witkowski
Assistant Professor of Computer Science and Management

Frankfurt School of Finance & Management
Adickesallee 32-34
60322 Frankfurt am Main
Germany

Email: j.witkowski@fs.de
Tel.: +49 (0) 69-154008-875
Fax: +49 (0) 69-154008-4875

Class 03

Classification



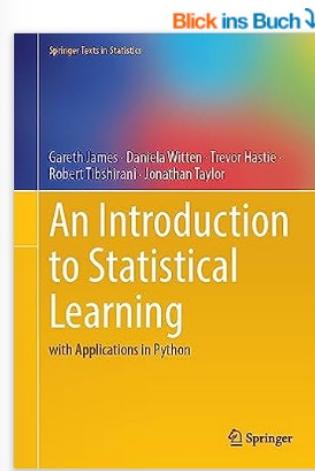
Computational Statistics

Prof. Dr. Jens Witkowski
September 7, 2023

Agenda

- Session 1
 - Recap
 - Model Selection (cont.)
- Session 2
 - Classification

Literature Pointer



Blick ins Buch ↓

An Introduction to Statistical Learning: with Applications in Python (Springer Texts in Statistics)



Gebundene Ausgabe – 1. Juli 2023

Englisch Ausgabe | von Gareth James (Autor), Daniela Witten (Autor), Trevor Hastie (Autor), Robert Tibshirani (Autor), & 1 mehr

5,0 ★★★★★ 1 Sternebewertung

[Alle Formate und Editionen anzeigen](#)

Gebundenes Buch

106,99 € ✓ prime

8 Neu ab 103,30 €

Taschenbuch

74,89 € ✓ prime

1 Neu ab 74,89 €

An Introduction to Statistical Learning provides an accessible overview of the field of statistical learning, an essential toolset for making sense of the vast and complex data sets that have emerged in fields ranging from biology to finance, marketing, and astrophysics in the past twenty years. This book presents some of the most important modeling and prediction techniques, along with relevant applications. Topics include linear regression, classification, resampling methods, shrinkage approaches, tree-based methods, support vector machines, clustering, deep learning, survival analysis, multiple testing, and more. Color graphics and real-world examples are used to illustrate the methods presented. This book is targeted at statisticians and non-statisticians alike, who wish to use cutting-edge statistical learning techniques to analyze their data.

Four of the authors co-wrote *An Introduction to Statistical Learning, With Applications in R* (ISLR), which has become a mainstay of undergraduate and graduate classrooms worldwide, as well as an important reference book for data scientists. One of the keys to its success was that each chapter contains a tutorial on implementing the analyses and methods presented in the R scientific computing environment. However, in recent years Python has become a popular language for data science, and there has been increasing demand for a Python-based alternative to ISLR. Hence, this book (ISLP) covers the same materials as ISLR but with labs implemented in Python. These labs will be useful both for Python novices, as well as experienced users.

[^ Weniger lesen](#)

[Falsche Produktinformationen melden.](#)

Dieses Bild anzeigen

Den Autoren folgen



Trevor
Hastie

Folgen



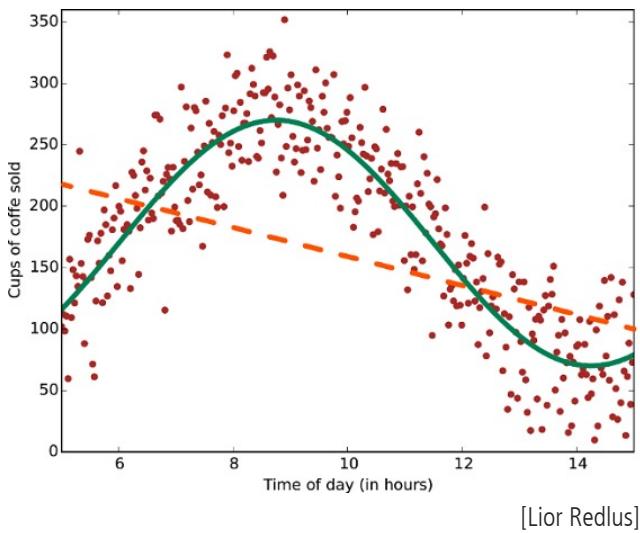
Gareth
James

Folgen

- Recommended for Class 01 (Sep 1): Section 2.1
- Recommended for Class 02 (Sep 6): Section 3.1, 3.2, 7.1
- Recommended for Class 03 (Sep 8): Sections 5.1

Recap: Model

$$\mathbf{y} = f(\mathbf{x}) + \boldsymbol{\epsilon}$$

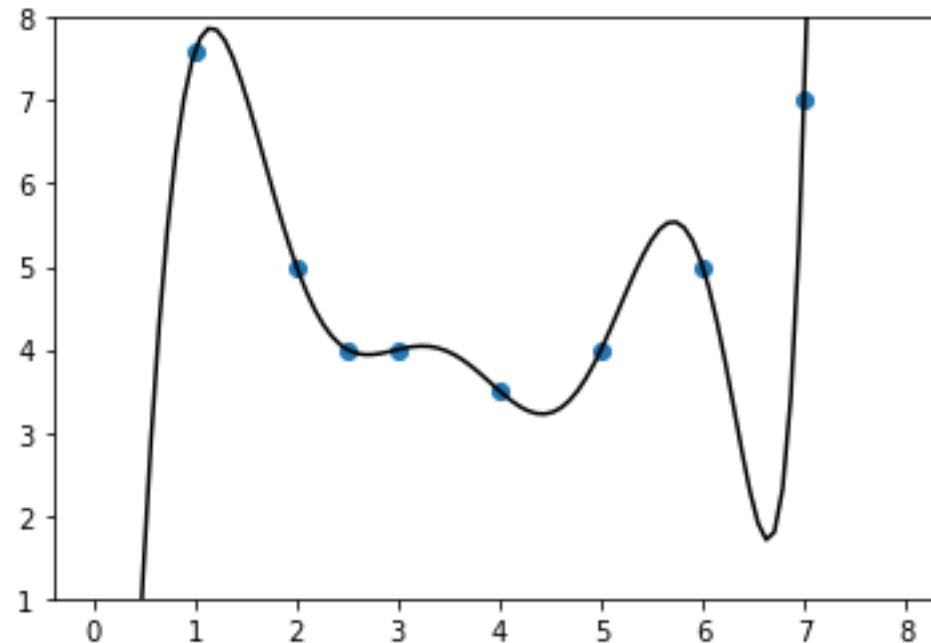


Assumptions of regression model in ML:

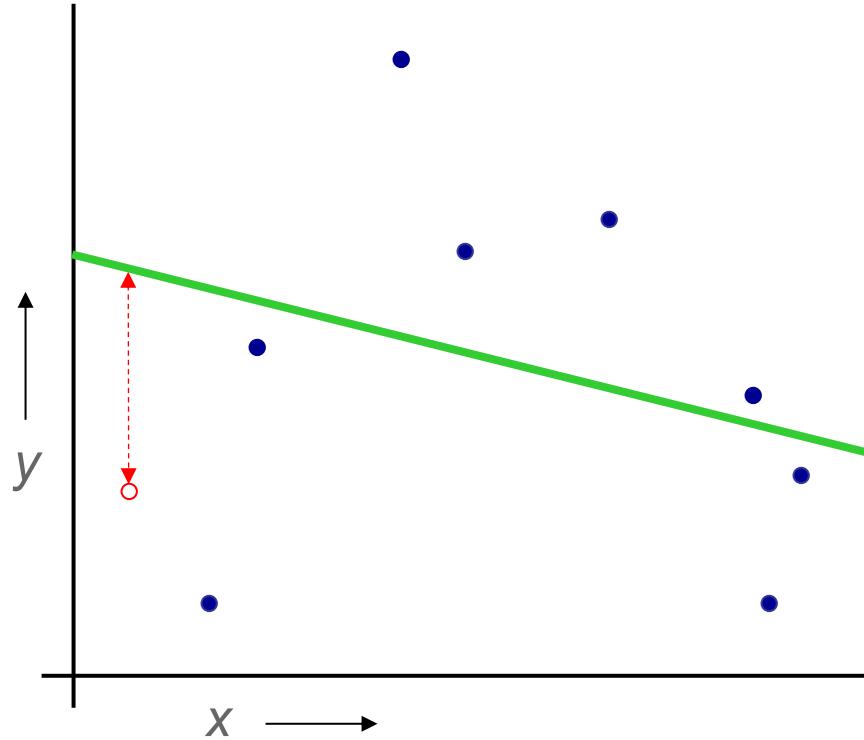
- There is a true, unknown function $f(\mathbf{x})$ (green)
- We get to observe noisy samples (x_i, y_i) (dark red)
- Objective: "learn" function $f(\mathbf{x})$ from samples.
- Reminder: noise $\boldsymbol{\epsilon}$ is assumed to be independent of \mathbf{x} and has expectation 0 (y values are distributed around $f(\mathbf{x})$).

Increased Model Complexity and Generalization

Fitting complex, non-linear functions (e.g., higher-order polynomials) fit the training data very well but might result in models that do not seem to generalize well to unseen data.



LOOCV (Leave-one-out Cross Validation)



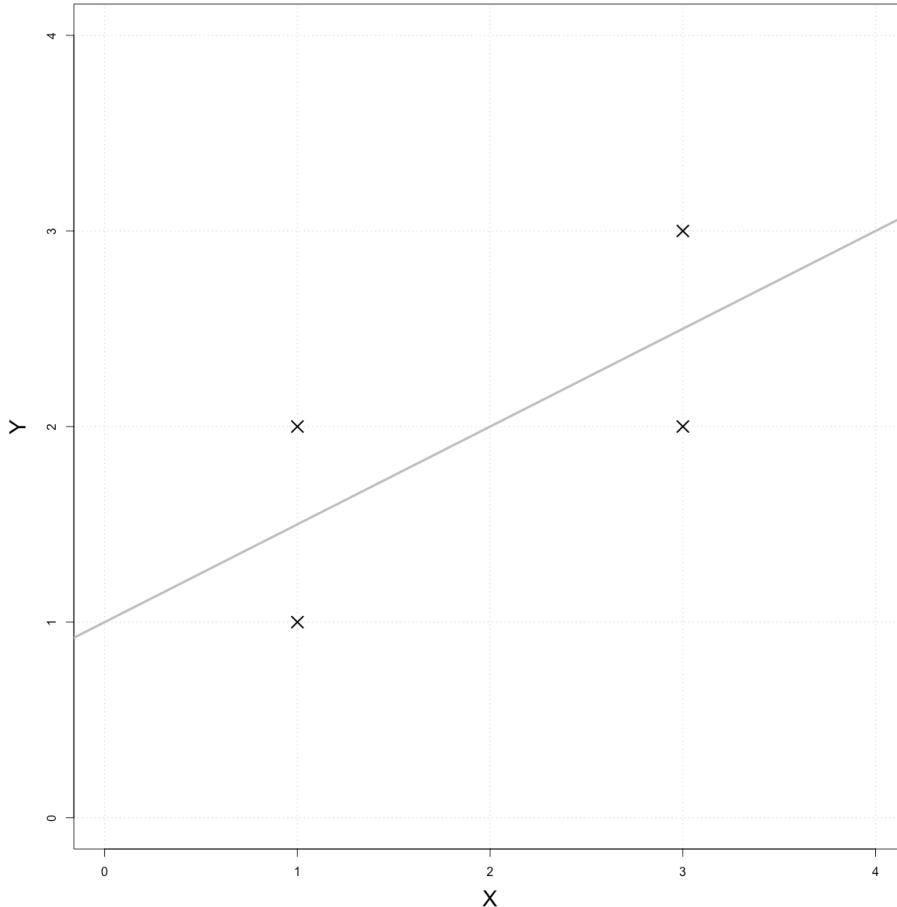
For $i = 1$ to n :

1. Let (x_i, y_i) be the i^{th} data point
2. Temporarily remove (x_i, y_i) from data set
3. Train model on remaining $n-1$ data points
4. Note error of model for (x_i, y_i)

When every point has been left out, report mean error.

Exercise

Data frame `df` contains 4 data points $(x_1, y_1) = (1, 1)$, $(x_2, y_2) = (1, 2)$, $(x_3, y_3) = (3, 2)$, and $(x_4, y_4) = (3, 3)$. Using scikit.learn, you fitted the linear model predicting Y from X without any additional features (the gray line is the fitted model).



- a) Compute the mean squared error (MSE) of `model1` over all 4 data points in `df`.
- b) Compute the Leave-One-Out-Cross Validation MSE for the linear model predicting Y from X.
Note that the model fit changes with the training data.

Reminder: the MSE is defined as $\frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2$.

Solution:

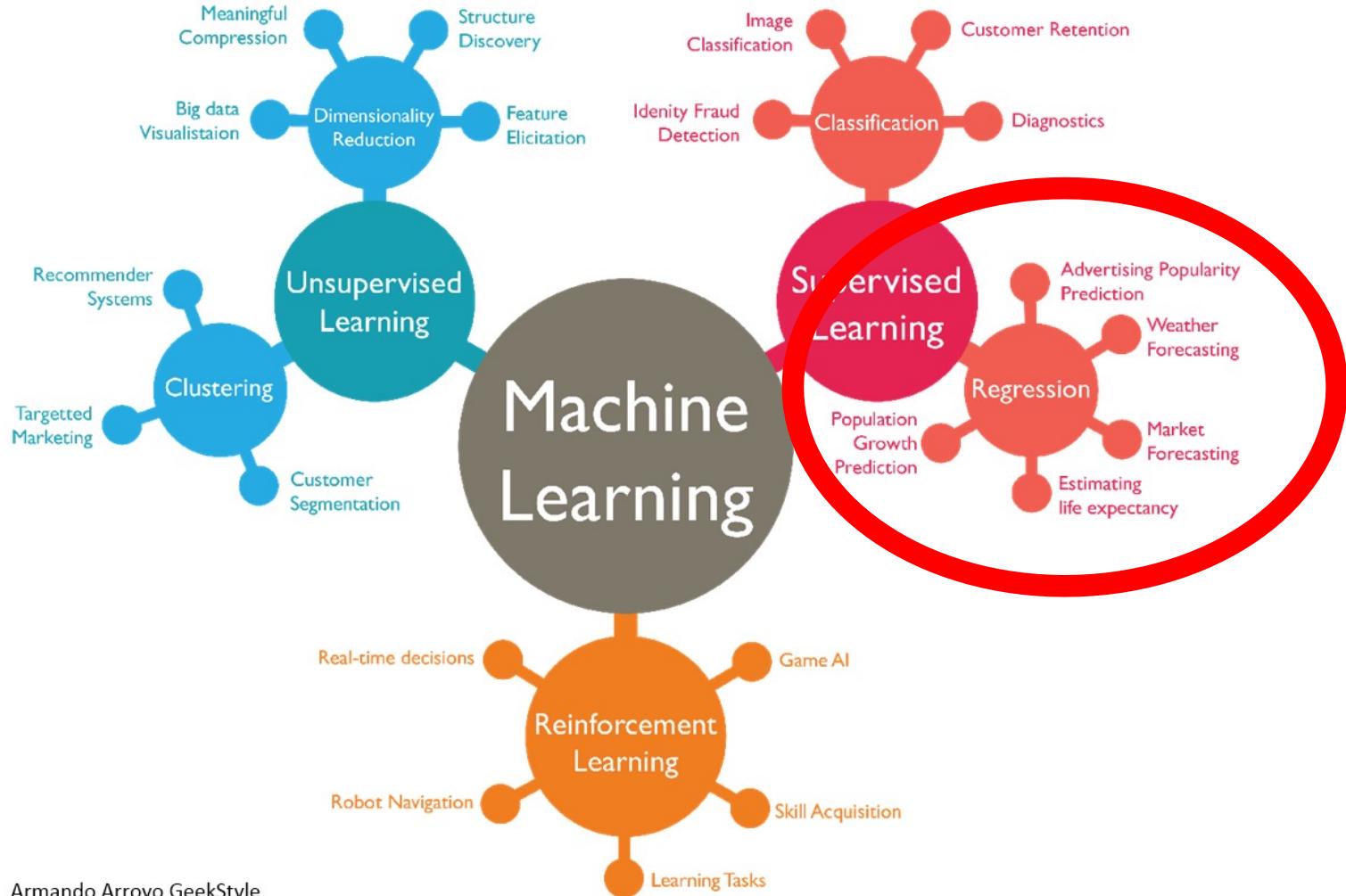
- a) $\frac{1}{4} * (0.5^2 + 0.5^2 + 0.5^2 + 0.5^2) = 0.5^2 = 0.25$
- b) $\frac{1}{4} * (1^2 + 1^2 + 1^2 + 1^2) = 1^2 = 1$

Controlling Model Complexity

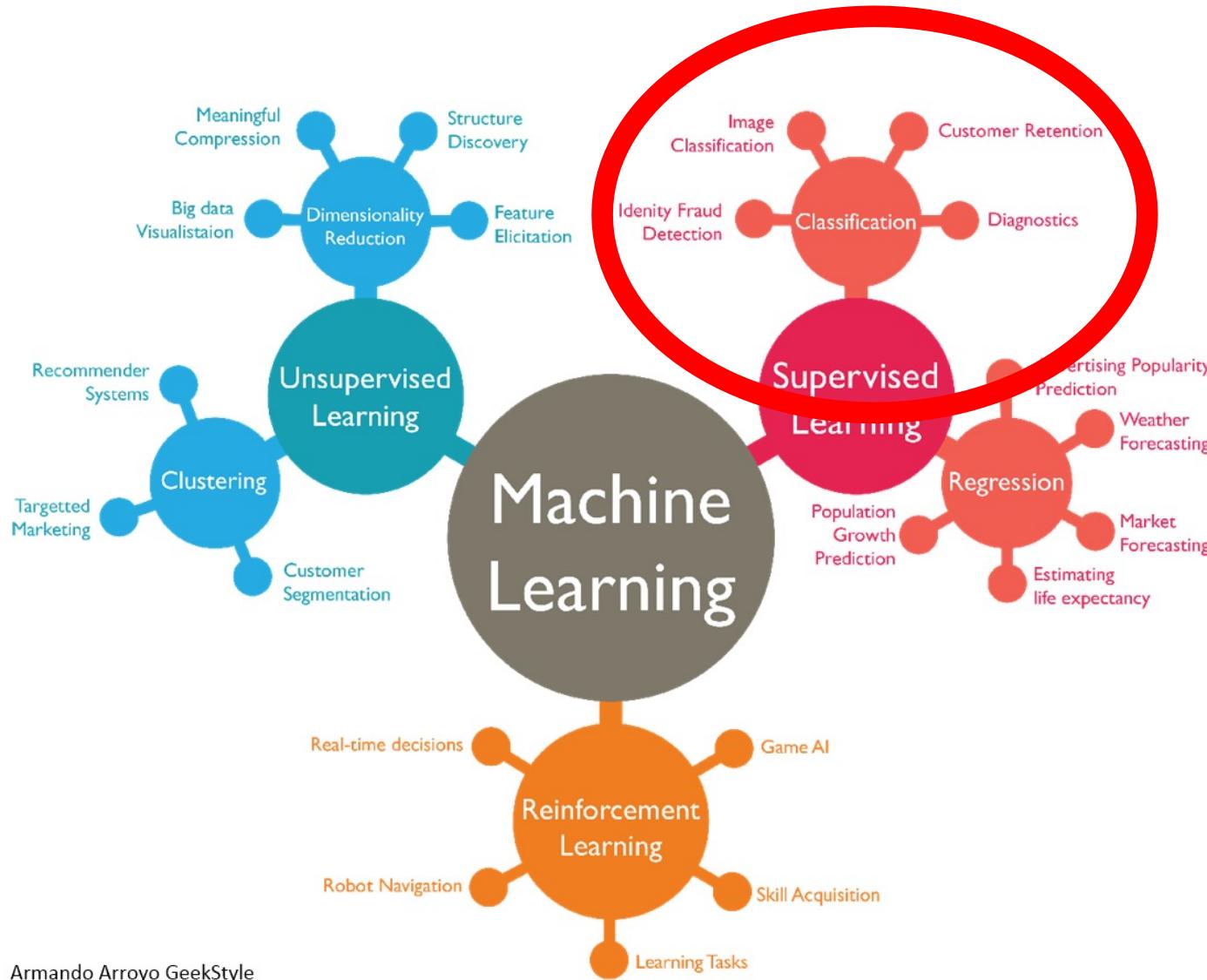
- So far: controlled model complexity through degree of polynomial (higher degree → higher model complexity)
- How can we control model complexity more generally (e.g., in linear regression with interaction terms or non-polynomial features such as $\log(x_1)$ or $\sqrt{x_2}$)?
- Typically, approaches fall into one of three classes:
 1. Subset selection: try out different subsets of features and select subset with best validation error.
 2. Dimension reduction methods: transform features into smaller feature space.
 3. Regularization: add punishment term for large coefficients to error minimization problem.

Most common today: dimension reduction and regularization.

So far: Regression



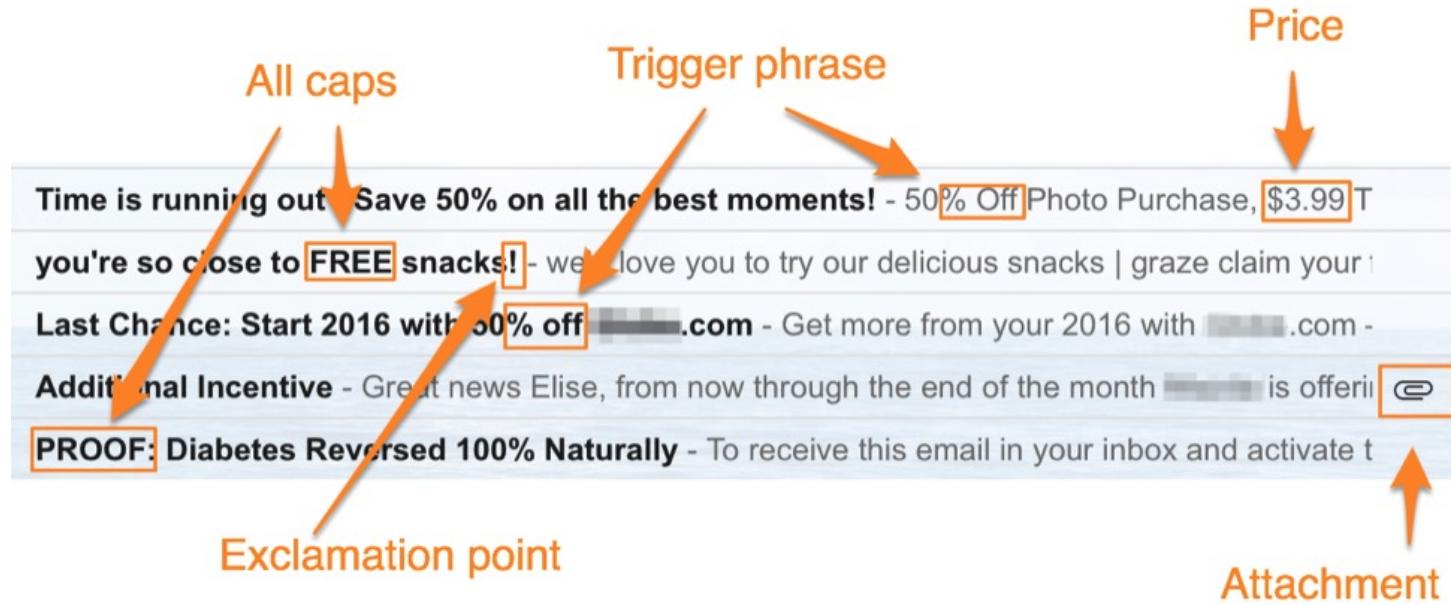
Today: Classification



Armando Arroyo GeekStyle

Example: Spam Filter

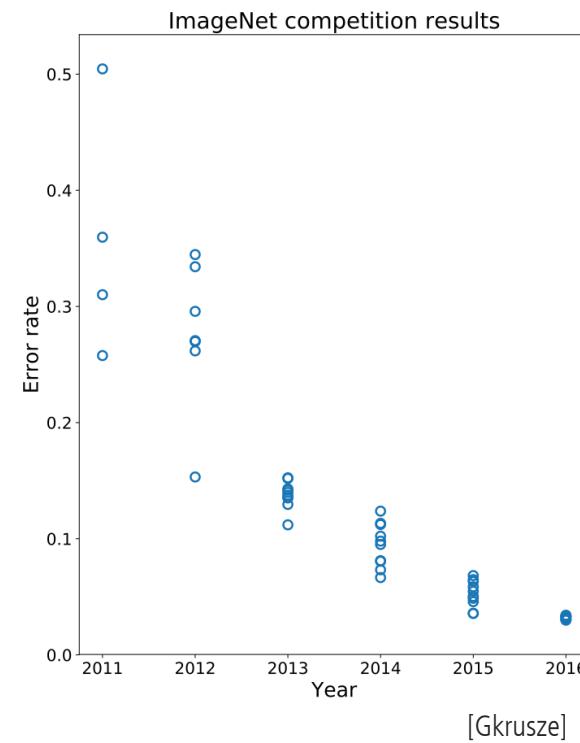
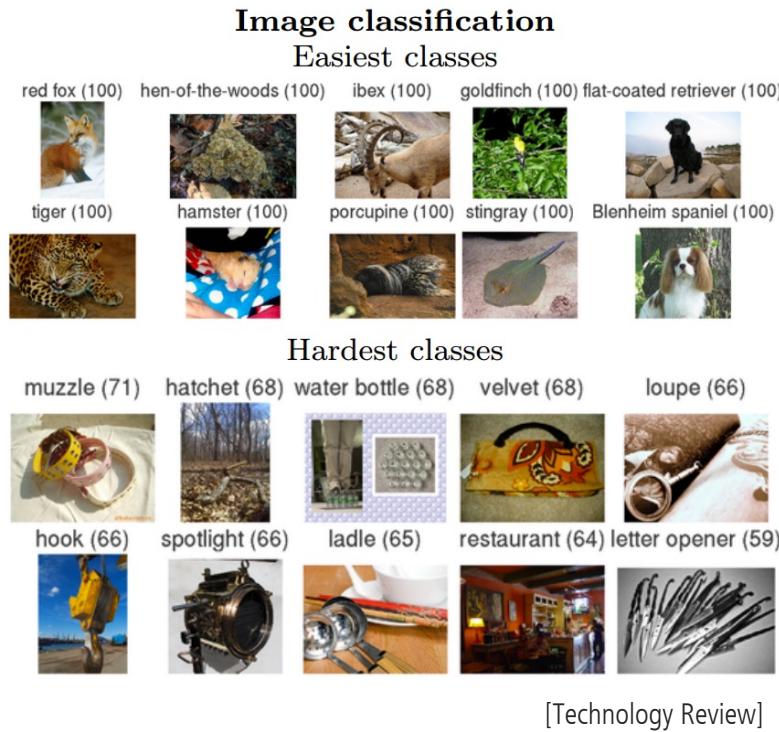
Classify email messages as "spam" or "no spam" (two classes):



Example: Image Recognition

ImageNet database:

- Over 14 million labeled images
- Over 20,000 different classes (e.g., "onion", "balloon", ...)
- ImageNet Challenge uses a reduced list of one thousand non-overlapping classes



For now: only binary (yes/no) classification!

Classification Setting

- Supervised learning where y to be predicted is categorical
- Want to assign data points \mathbf{x}
 - Documents
 - Queries
 - Images
 - Website visitor
 - ...

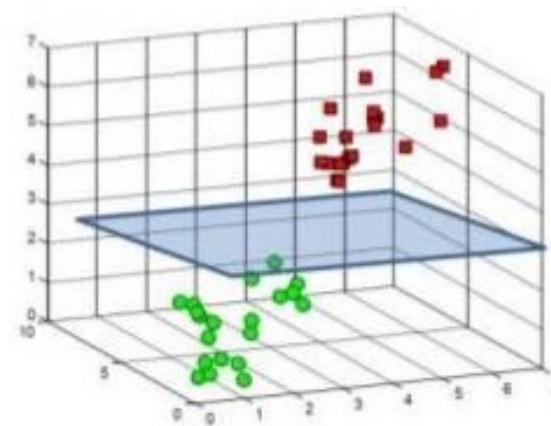
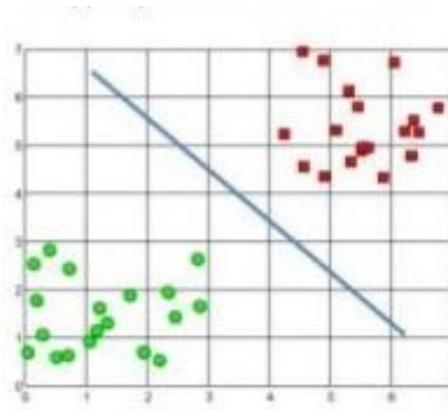
a label y ("spam"/"no spam"; contents of an image such as "onion"/"dog"/"house", "buy"/"no buy" etc.)

- As in regression: data points $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_n, y_n)$
- As in regression: $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n \in \mathbb{R}^d$ are the feature vectors of dimension d
- But now: $y_1, y_2, \dots, y_n \in \{-1, +1\}$ are the binary labels (e.g., -1 for "no spam" and +1 for "spam")
- Typically seek to minimize number of misclassifications ("0/1 loss")

(bold font refers to vectors; will sometimes write X_1 and X_2 for first and second dimension)

Separating Hyperplanes

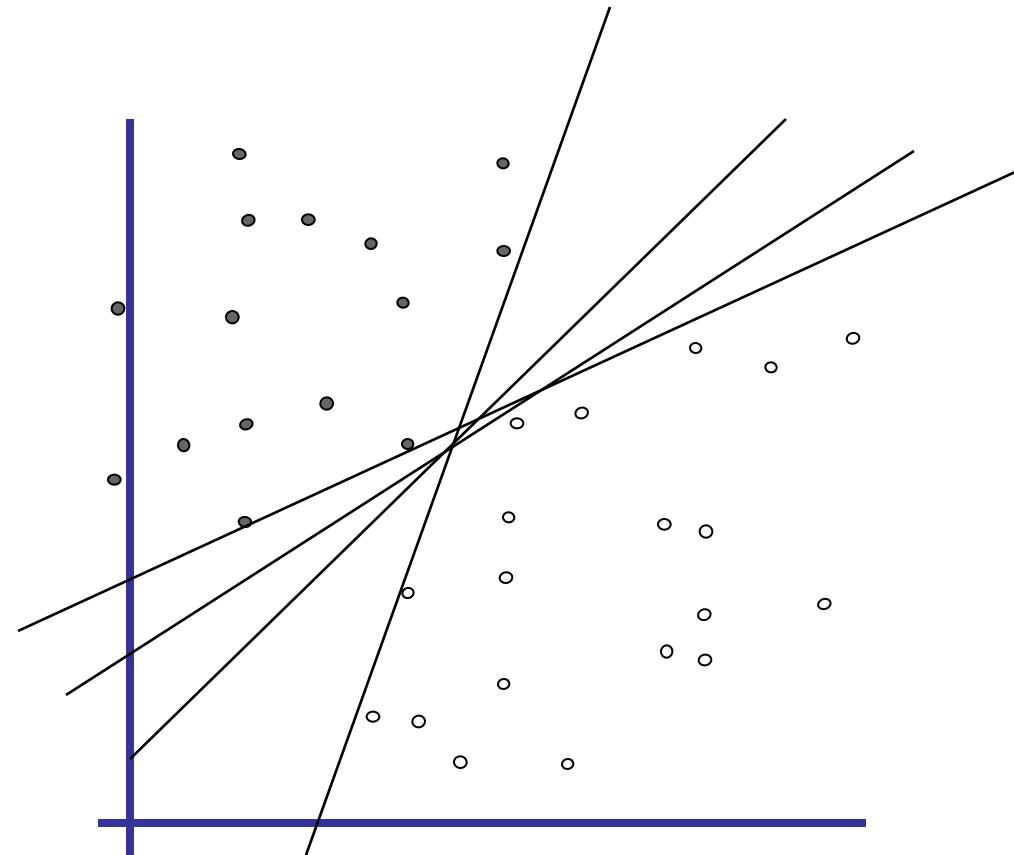
- Hyperplane: “In a p-dimensional space, a hyperplane is a flat affine subspace of dimension p-1.”
- For our purposes:
 - In 2 dimensions, a hyperplane is a line.
 - In 3 dimensions, a hyperplane is a plane.



- In 1 dimension?
Hyperplane is a point.
- Use hyperplane to divide feature space into two sides (one for each class)
- For new point with feature vector \mathbf{x} , predict class label depending on which side of hyperplane \mathbf{x} is

Possible Linear Separations

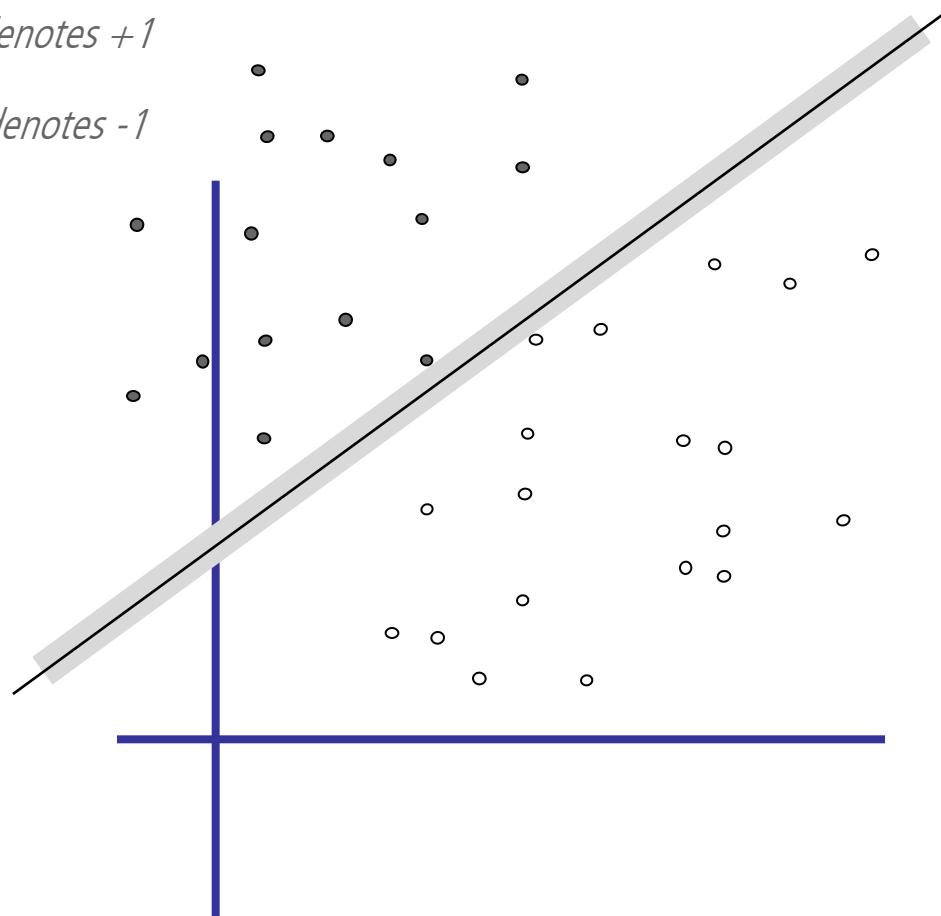
- denotes +1
- denotes -1



How would you separate the two classes?
Any of these seems to be fine on this data...?

Classifier Margin

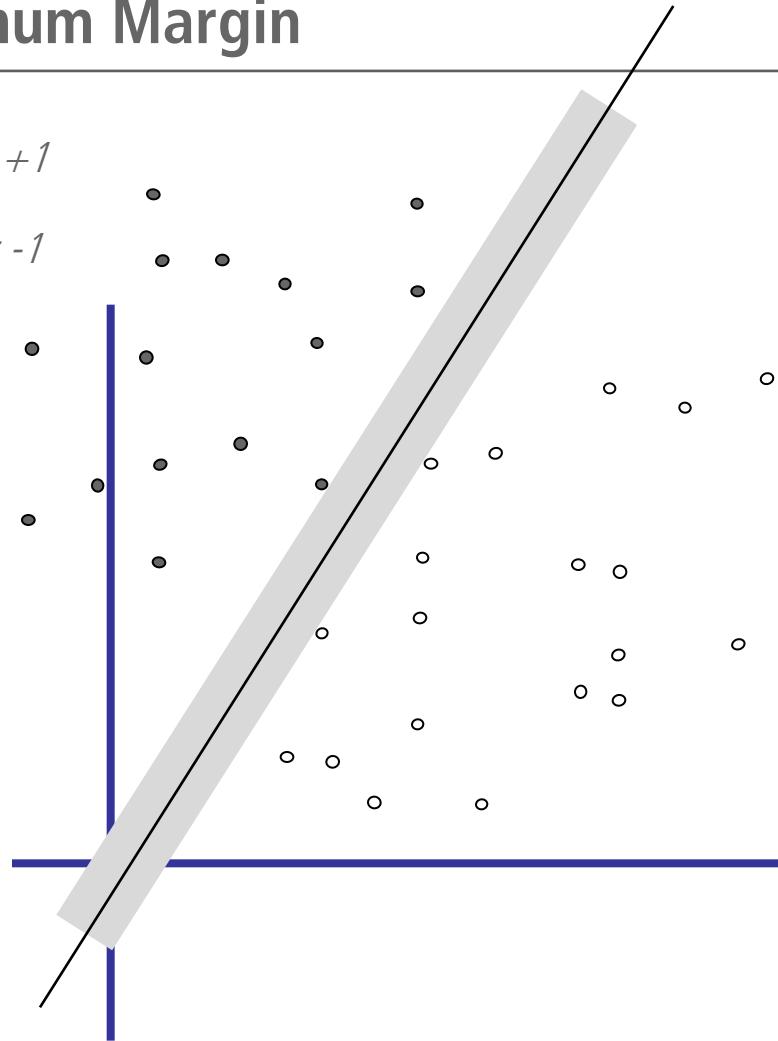
- denotes +1
- denotes -1



Margin of a linear classifier:
width that the boundary of a separating
hyperplane could be increased by before hitting
a data point.

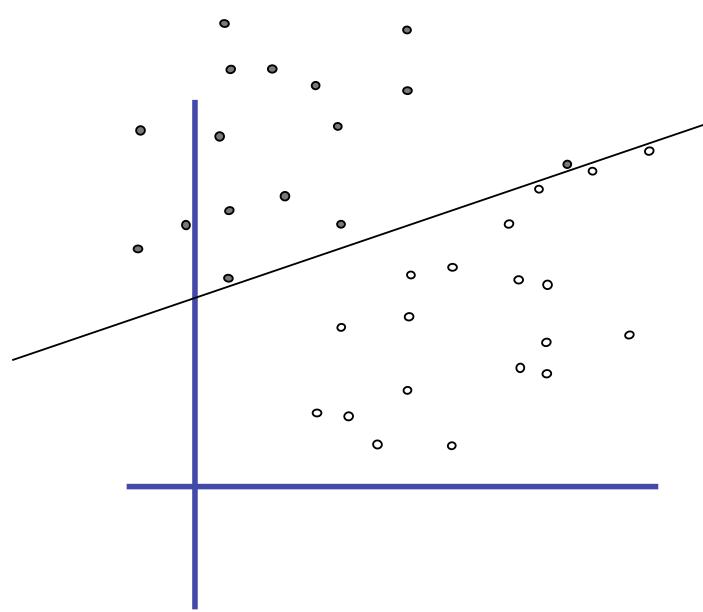
Maximum Margin

- denotes +1
- denotes -1

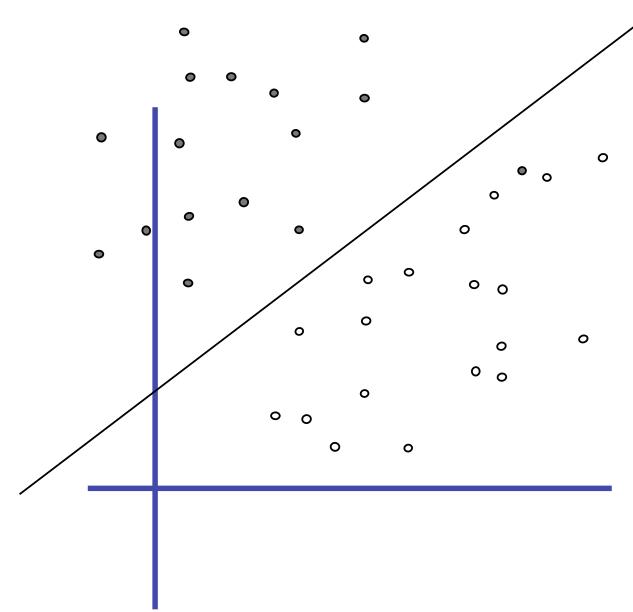


- *Maximum margin linear classifier* is the separating hyperplane with, well, maximum margin.
- Idea: maximum margin on training data is most robust for test data (supported by theory and works very well in practice)
- Can be formulated as an optimization problem that is efficiently computable.

Tricky Example



No misclassification but small margin

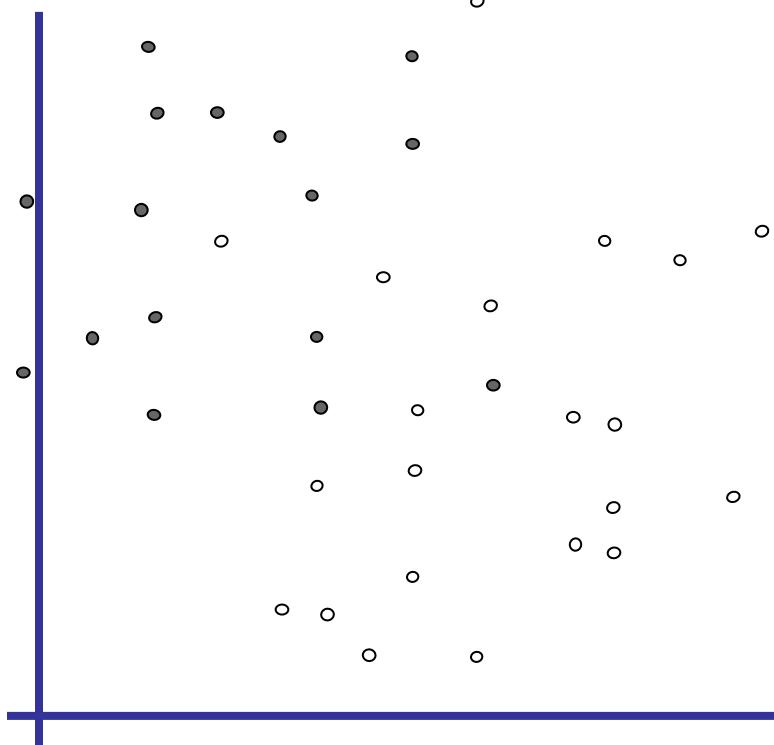


One misclassification but (ignoring that point) large margin

Why might you want to choose large margin? What is potential concern with small margin?

Overfitting to training data!

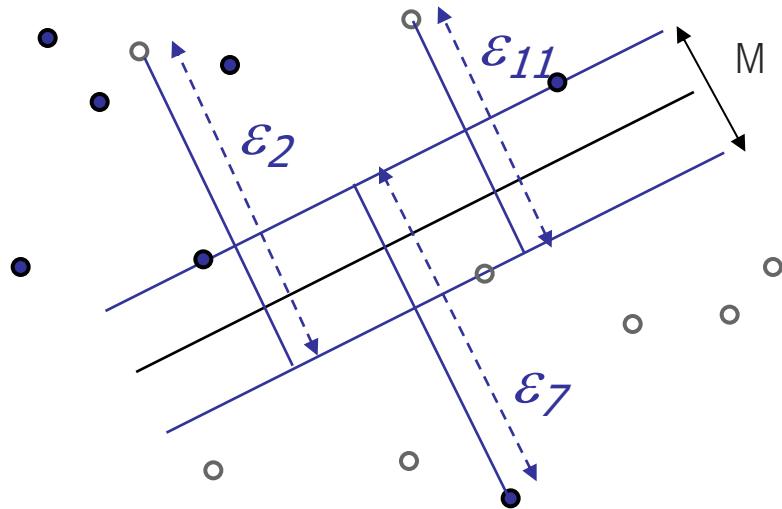
The Non-Separable Case



- Might not be able to linearly separate classes exactly
- Idea: extend concept of separating hyperplane to hyperplane that *almost* separates classes
- Might be worthwhile to misclassify a few training observations in order to do better job in classifying remaining points!

Maybe classifier can allow some misclassifications at a cost...

Soft Margin Classifier



Maximize M

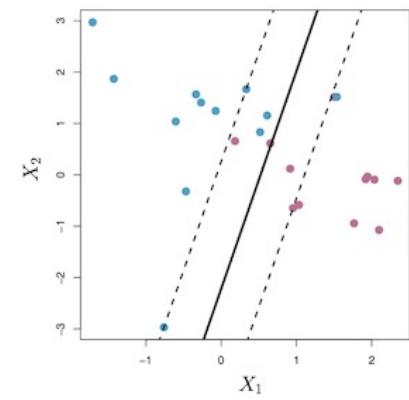
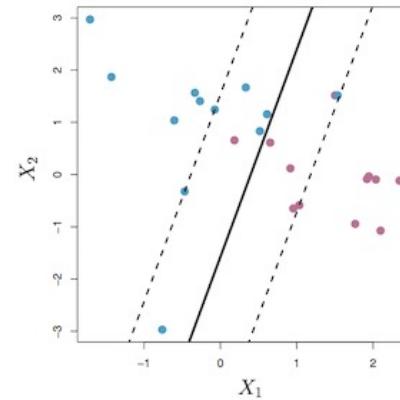
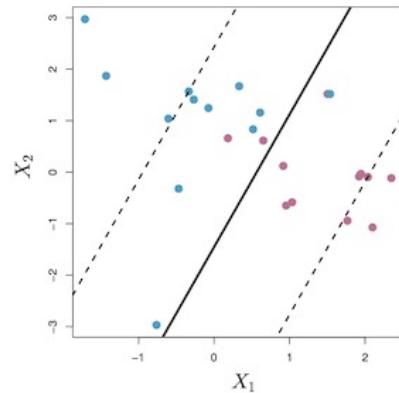
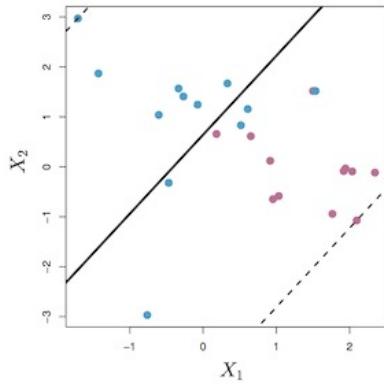
Subject to $\sum_{i=1}^n \varepsilon_i \leq B$

$$\varepsilon_i \geq 0$$

- Let $\varepsilon_1, \dots, \varepsilon_n$ be the “slack variables” that are positive only for points that are on the wrong side of the margin.
- Give classifier a “budget” B for the slack variables, which cannot be exceeded (i.e., budget for violations to the margin that we tolerate).
- Points that lie in the margin (including on its “borders”) or on the wrong side of the margin are called “support vectors.”
- Classifier’s decision based only on the support vectors: robust towards changes further away from hyperplane.

Behavior for Different Values of B

Four different values of B from large to small:

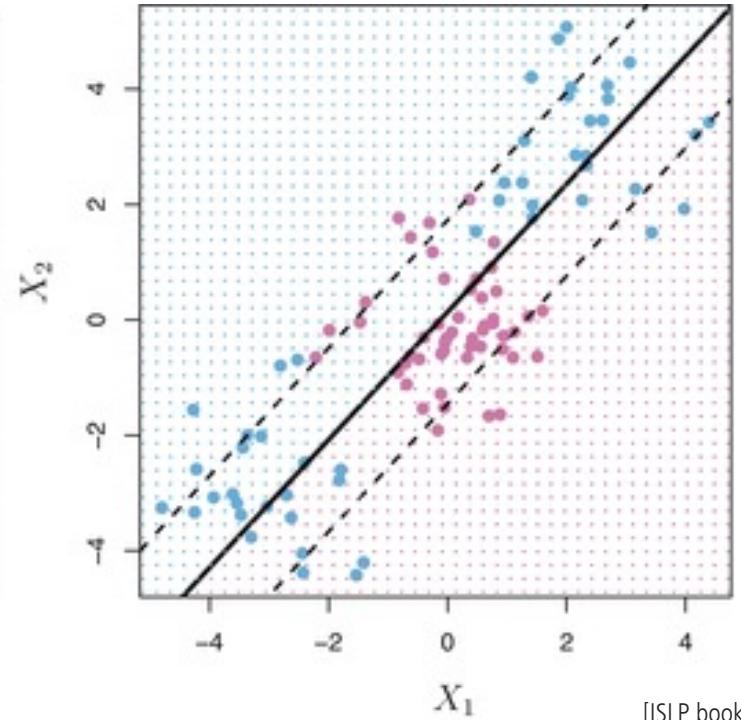
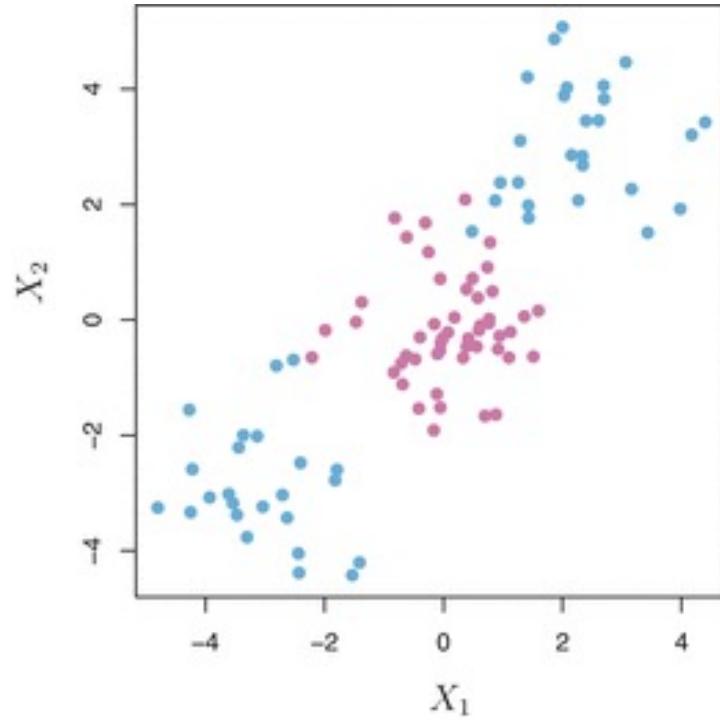


[ISLP book]

- Large B values lead to large margins with many violations but potentially better generalization error.
- Smaller B values allow less room for violations of the margin but might lead to overfitting.

How to choose parameter B?
Cross validation!

Non-Linear Class Boundaries: 2-D Example

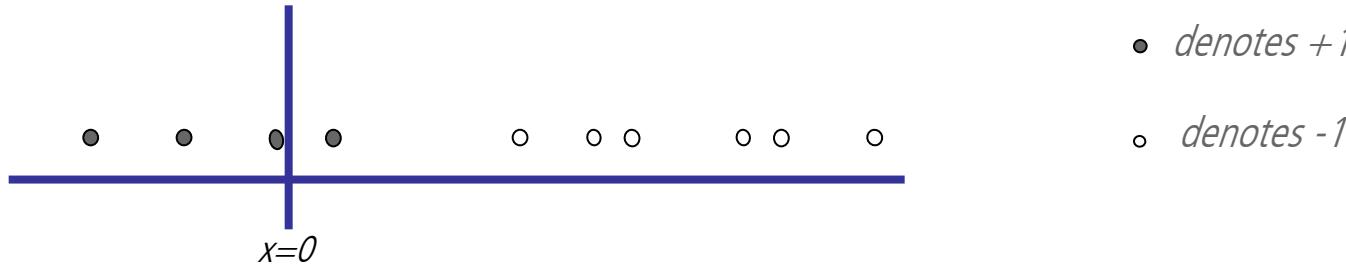


[ISLP book]

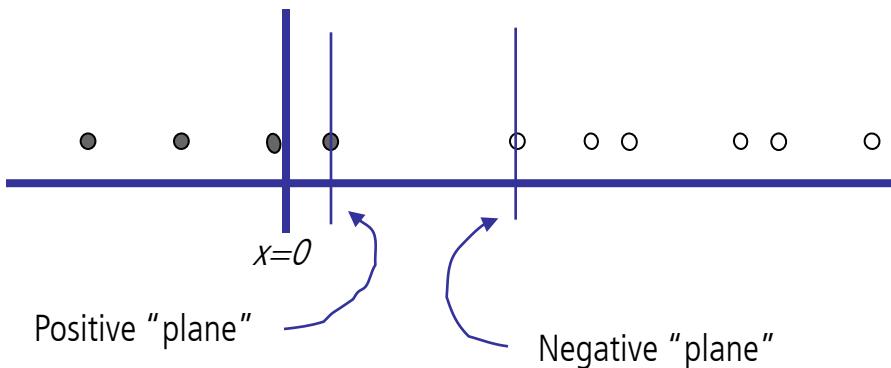
Linear soft-margin classifier performs poorly with non-linear class boundaries

Non-Linear Class Boundaries: 1-D Example

- This is linearly separable: (Reminder: hyperplane in 1-D is a point.)



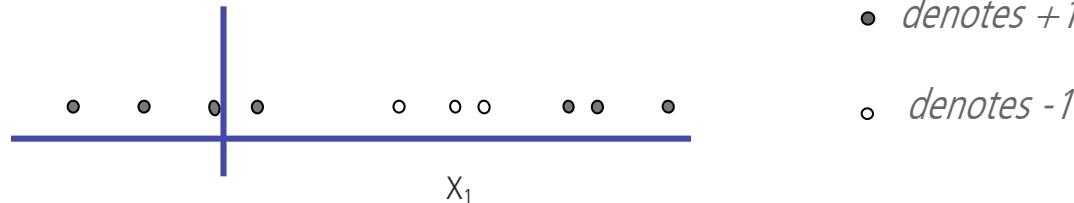
- What would maximum-margin classifier do?



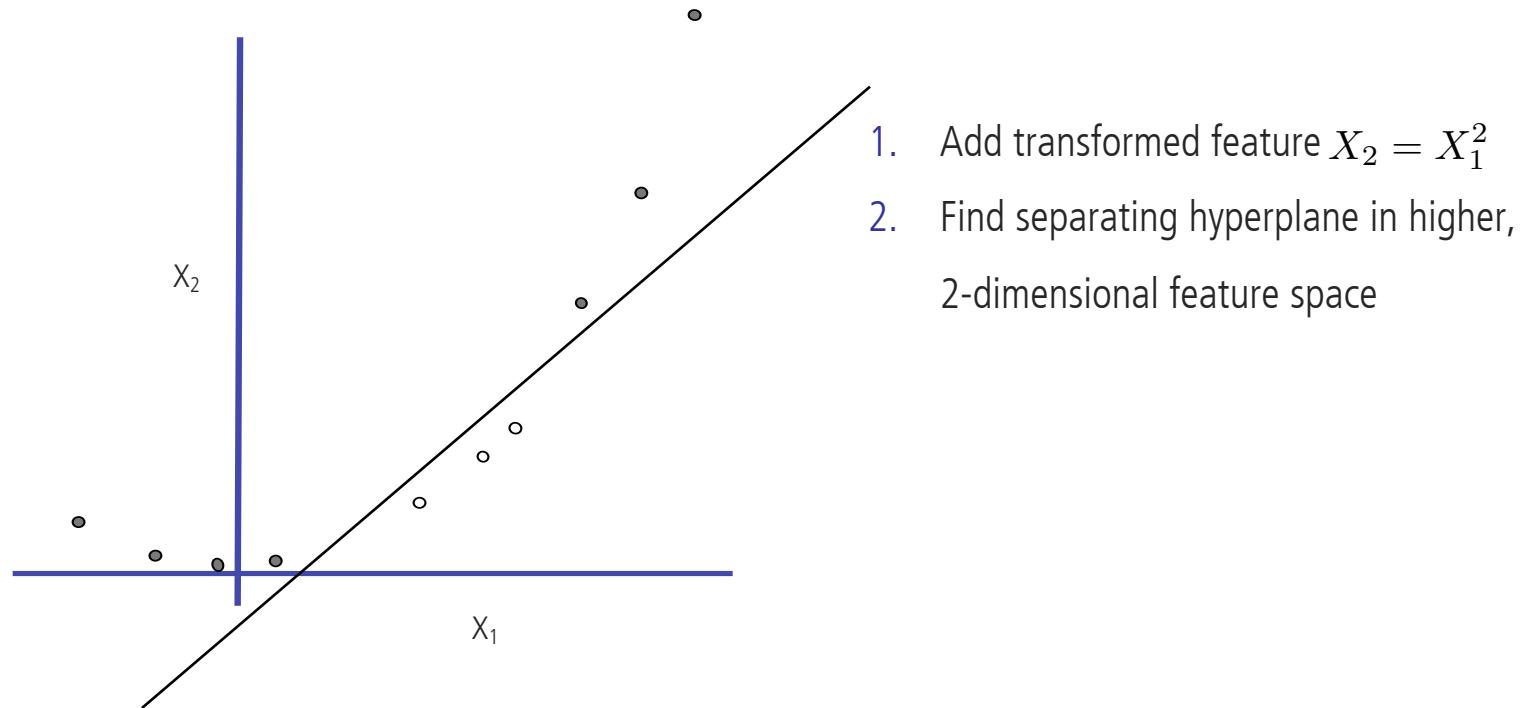
- OK, sure...

Non-Linear Class Boundaries: 1-D Example (cont.)

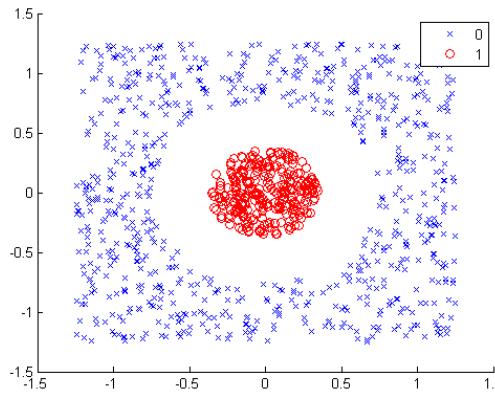
- No longer linearly separable:



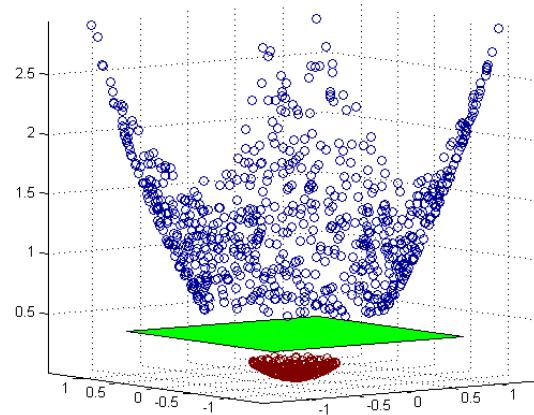
- Remember how we could transform features to obtain non-linear fits in regression? Let's do that here, too:



Feature Transformation 2D -> 3D



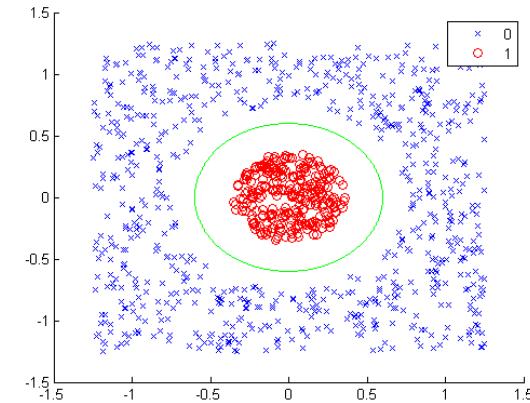
Data in original 2-D feature space



Data with added transformed feature

$$X_3 = X_1^2 + X_2^2$$

and separating hyperplane (green)
in enlarged 3-dimensional space



Resulting decision boundary in
original 2-D feature space

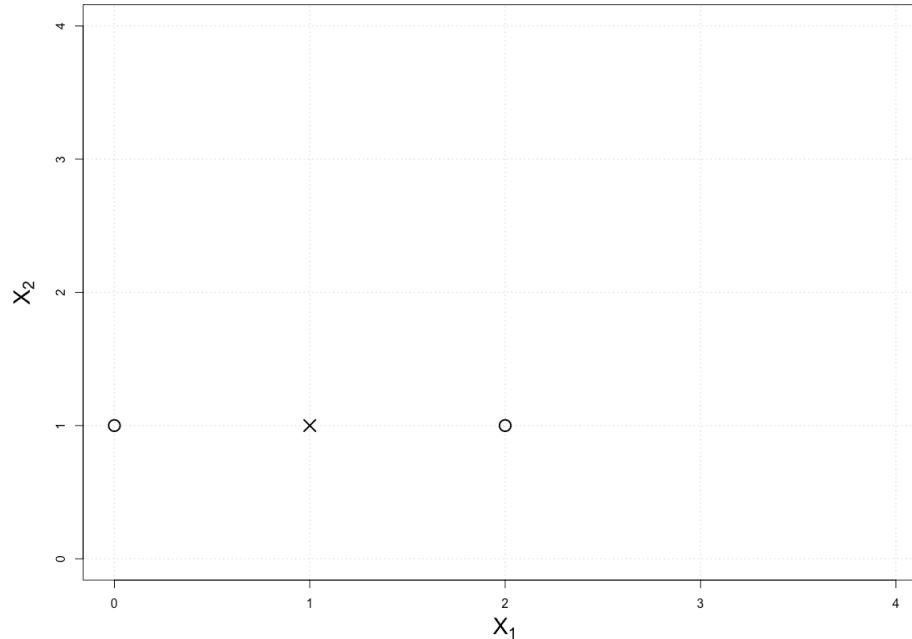
[P. López]

Can extend linear support vector classifier in same way we extended linear regression to fit non-linear functions by enlarging feature space with non-linear transformations of the original features!

Support Vector Machines (SVMs) are computationally more efficient approach of that same idea.

Exercise

Consider the following data set with 3 points (crosses for "+1", circles for "-1")



The two classes are not linearly separable in the original feature space. Which of the following transformed features would make the data linearly separable in the transformed space using only that single feature X'? (circle all correct answers)

- i) $X' := ((X_1 + X_2) - 2)^2$
- ii) $X' := -X_1^2 + X_2^2$
- iii) $X' := -(X_1 * X_2)^2$
- iv) $X' := (X_1 - 1)^2$

Solution: i) and iv)

Extend Binary Classifier to more than 2 Classes

One-Versus-One Classification:

1. Fit a separate binary classifier for each possible combination of classes (e.g., for $k=4$ classes, we would need 6 binary classifiers)
2. At prediction time: ask every binary classifier and tally number of times that a test observation is assigned to each of the k classes. Predict the class that has the highest number.

One-Versus-All Classification:

1. Let k again be the number of classes. Fit k binary classifiers, each time comparing one of the classes to the remaining $k-1$ classes.
2. At prediction time: for every binary classifier, look at how far the observation is on the "+1" side of the separating hyperplane, and predict the class where this is maximized (or where "-1" is minimized if observation is not on any classifier's "+1" side).

NB: Both are heuristics that can lead to ambiguities. Some classification algorithms generalize more directly.

Homework 1

1. Download `hw1.csv` and load it as `df` into your Python environment. You want to predict the data in column 1 from the data in column 0.
 - a) Plot the data with column 0 on the x axis and column 1 on the y axis.
 - b) Fit polynomial models with degree from 1 to 20 and implement 5-fold cross validation. Which of these 20 model classes has lowest validation MSE? Briefly discuss.
 - c) Set aside 10% of the data in a test set and then reuse your code from part b) to run 5-fold cross validation on the remaining data. Which of the 20 model classes has lowest validation MSE and what is that model's expected generalization error?
 - d) Train a model from the model class you found to be best on the entire data set and provide the resulting model as `best_model` in your code.

Homework 1 (cont.)

2. Extend the code in `linear_regression.py` (from lecture) with functions that take a feature matrix and return a feature matrix with new, transformed (typically: non-linear) features.

- a) Implement a short function `transformed_feature_matrix_quadratic(X)` that adds a quadratic feature to the feature space. For example, for the current dataset, calling `transformed_feature_matrix_quadratic(X)` should return:

```
>>> array([[ 1. ,  1. ],
       [ 2. ,  4. ],
       [ 3. ,  9. ],
       [ 4. , 16. ],
       [ 5. , 25. ],
       [ 6. , 36. ],
       [ 7. , 49. ],
       [ 2.5 ,  6.25]])
```

Apply this function to fit the `LinearRegression` object using these transformed features and to plot the model's predictions. Do not create new variables for these transformed features and do not add any additional libraries to `linear_regression.py`.

- b) Similarly, implement a short function `transformed_feature_matrix_singlecube(X)` that returns a feature matrix with a single feature that has been cubed. (Again, also fit and plot the model as in a)). For the current dataset, the function would return:

```
>>> array([[ 1. ],
       [ 8. ],
       [ 27. ],
       [ 64. ],
       [125. ],
       [216. ],
       [343. ],
       [ 15.625]])
```

- c) Can you find a single feature transformation that achieves a (training) MSE smaller than 0.06 on the full dataset with that single feature? (You do not need to code any optimization but are allowed to just try out different transformations.) Implement this as function `transformed_feature_matrix_singlemin(X)`. Also fit and plot the model as before and provide a short discussion of your result: How did you find it? What is the intuition behind it? What might be problematic about this from an applied data science perspective?

Homework 1 (cont.)

General notes:

- Only upload two ".py" files (one for each part, called `hw1_1.py` and `hw1_2.py`, respectively) with your code and your textual answers in comments right after the respective code.
- Elegant and easier-to-follow code and plots are part of this (and following) homeworks and will get more points than naive code and ugly plots. Your code should also have meaningful comments. Typos lead to point reductions.
- All students in a group need to upload the (identical) group solution before the deadline. Clearly state all names in the comments at the beginning of your code.
- Submission deadline is 1.15pm on September 14. For fairness, late submissions will not be accepted.
- **Start early!** Depending on your familiarity with Python and your current understanding of the course material, this could take longer to finish than you expect!
- Reminder: All students in a group need to fully understand the submitted solutions and no collaborations outside of your group are allowed! We will discuss your submitted solutions and you will need to be able to answer questions about them.

Any question?

Credits

Some of the figures in this presentation are taken from Andrew W. Moore
(<http://www.cs.cmu.edu/~awm/tutorials.html>) with permission from the author.

Prof. Dr. Jens Witkowski
Assistant Professor of Computer Science and Management

Frankfurt School of Finance & Management
Adickesallee 32-34
60322 Frankfurt am Main
Germany

Email: j.witkowski@fs.de
Tel.: +49 (0) 69-154008-875
Fax: +49 (0) 69-154008-4875

Class 04

Support Vector Classifier & Decision Trees

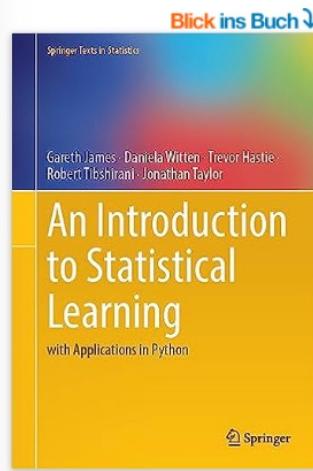


Computational Statistics

Prof. Dr. Jens Witkowski

September 11, 2023

Literature Pointer



Blick ins Buch ↓

An Introduction to Statistical Learning: with Applications in Python (Springer Texts in Statistics)

Gebundene Ausgabe – 1. Juli 2023

Englisch Ausgabe | von Gareth James (Autor), Daniela Witten (Autor), Trevor Hastie (Autor), Robert Tibshirani (Autor), & 1 mehr

5,0 ★★★★★ 1 Sternebewertung

[Alle Formate und Editionen anzeigen](#)



Gebundenes Buch

106,99 € ✓ prime

8 Neu ab 103,30 €

Taschenbuch

74,89 € ✓ prime

1 Neu ab 74,89 €

An Introduction to Statistical Learning provides an accessible overview of the field of statistical learning, an essential toolset for making sense of the vast and complex data sets that have emerged in fields ranging from biology to finance, marketing, and astrophysics in the past twenty years. This book presents some of the most important modeling and prediction techniques, along with relevant applications. Topics include linear regression, classification, resampling methods, shrinkage approaches, tree-based methods, support vector machines, clustering, deep learning, survival analysis, multiple testing, and more. Color graphics and real-world examples are used to illustrate the methods presented. This book is targeted at statisticians and non-statisticians alike, who wish to use cutting-edge statistical learning techniques to analyze their data.

Four of the authors co-wrote *An Introduction to Statistical Learning, With Applications in R* (ISLR), which has become a mainstay of undergraduate and graduate classrooms worldwide, as well as an important reference book for data scientists. One of the keys to its success was that each chapter contains a tutorial on implementing the analyses and methods presented in the R scientific computing environment. However, in recent years Python has become a popular language for data science, and there has been increasing demand for a Python-based alternative to ISLR. Hence, this book (ISLP) covers the same materials as ISLR but with labs implemented in Python. These labs will be useful both for Python novices, as well as experienced users.

[^ Weniger lesen](#)

[Falsche Produktinformationen melden.](#)

Dieses Bild anzeigen

Den Autoren folgen



Trevor
Hastie

Folgen



Gareth
James

Folgen

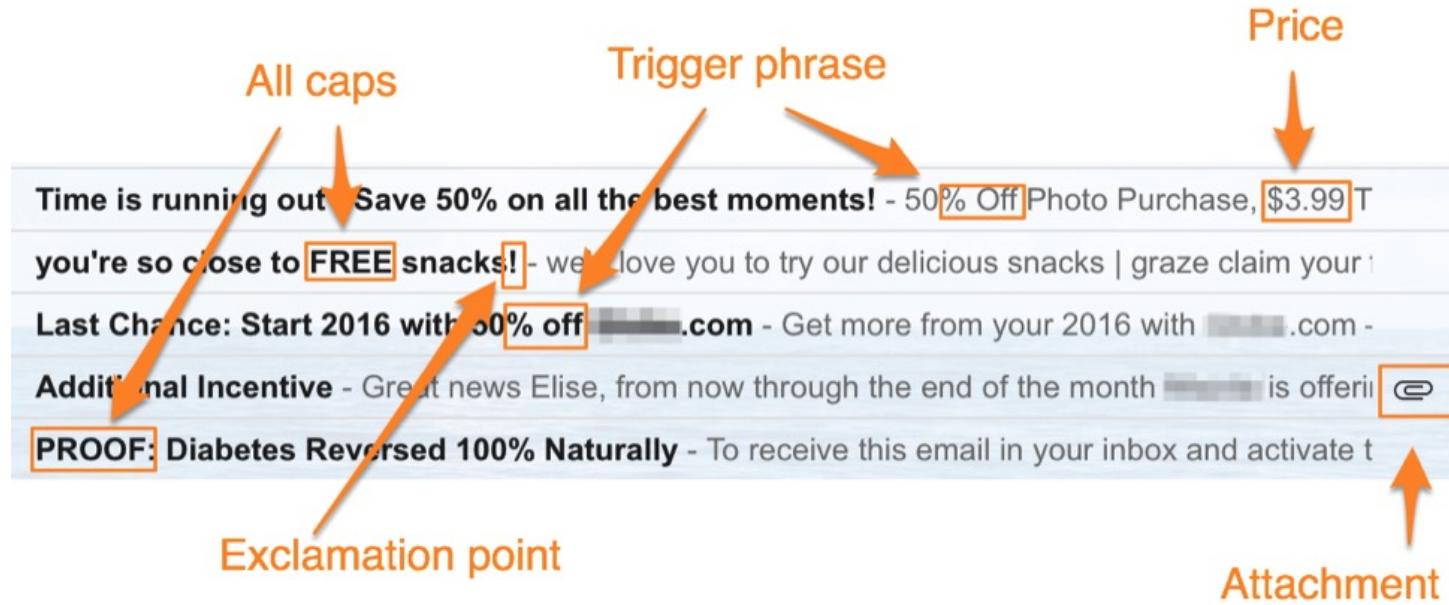
- Recommended for Class 01 (Sep 4): Section 2.1
- Recommended for Class 02 (Sep 7): Section 3.1, 3.2, 7.1
- Recommended for Class 03 (Sep 7): Sections 5.1, 9.1-9.4
- Recommended for Class 04 (today): Sections 9.6, 8.1

Agenda

- Session 1
 - Recap
 - Mathematical Representation
 - Python Exercise
- Session 2
 - Decision Trees: Regression

Example: Spam Filter

Classify email messages as "spam" or "no spam" (two classes):



Classification Setting

- Supervised learning where y to be predicted is categorical
- Want to assign data points \mathbf{x}
 - Documents
 - Queries
 - Images
 - Website visitor
 - ...

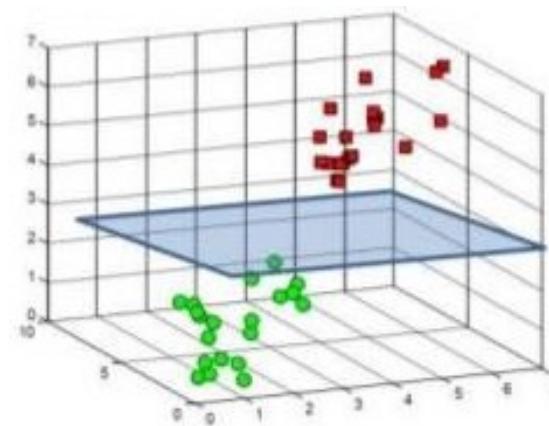
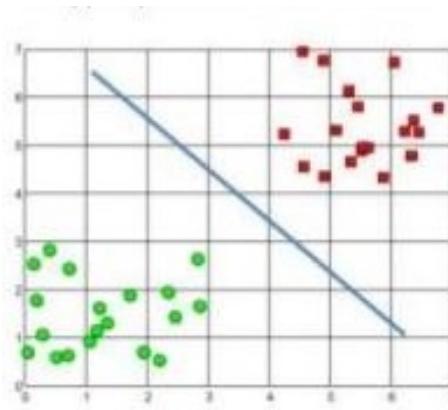
a label y ("spam"/"no spam"; contents of an image such as "onion"/"dog"/"house", "buy"/"no buy" etc.)

- As in regression: data points $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_n, y_n)$
- As in regression: $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n \in \mathbb{R}^d$ are the feature vectors of dimension d
- But now: $y_1, y_2, \dots, y_n \in \{-1, +1\}$ are the binary labels (e.g., -1 for "no spam" and +1 for "spam")
- Typically seek to minimize number of misclassifications ("0/1 loss")

(bold font refers to vectors; will sometimes write X_1 and X_2 for first and second dimension)

Separating Hyperplanes

- Hyperplane: “In a p-dimensional space, a hyperplane is a flat affine subspace of dimension p-1.”
- For our purposes:
 - In 2 dimensions, a hyperplane is a line.
 - In 3 dimensions, a hyperplane is a plane.



- In 1 dimension?
Hyperplane is a point.
- Use hyperplane to divide feature space into two sides (one for each class)
- For new point with feature vector \mathbf{x} , predict class label depending on which side of hyperplane \mathbf{x} is

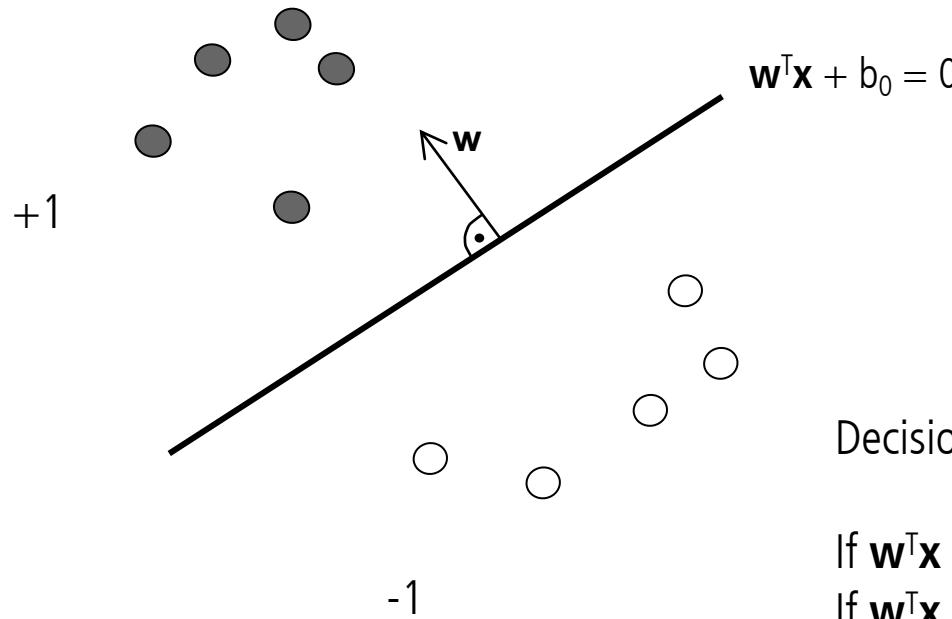
Mathematical Representation

A hyperplane can be described by equation:

$$b_0 + b_1 x_1 + b_2 x_2 + \dots + b_d x_d = 0$$

Let $\mathbf{w}^T = (b_1, b_2, \dots, b_d)$ be shorthand for the weight vector.

Hyperplane can then be written as $\mathbf{w}^T \mathbf{x} + b_0 = 0$.



Decision rule:

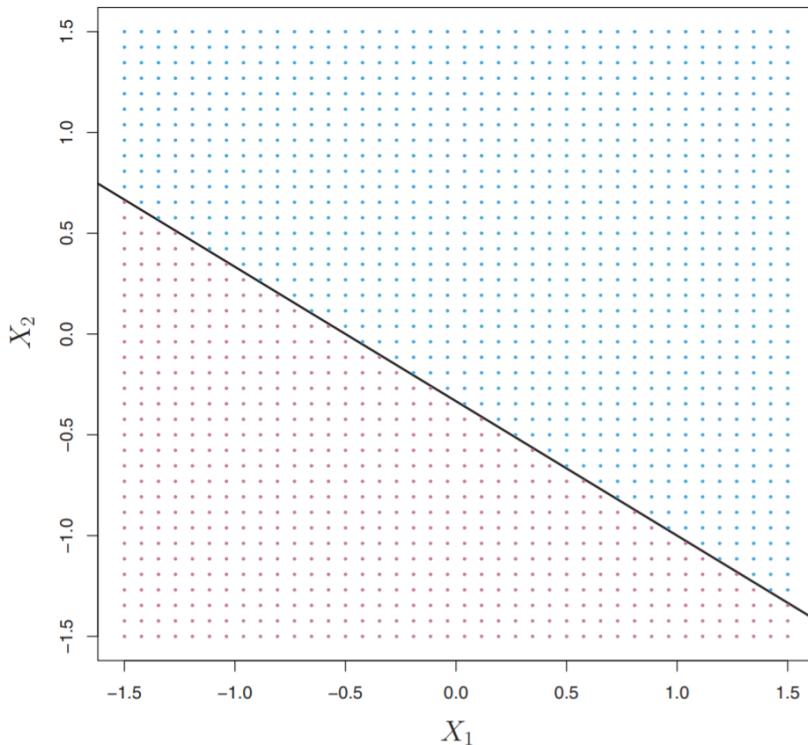
- If $\mathbf{w}^T \mathbf{x} + b_0 > 0$: predict class +1
- If $\mathbf{w}^T \mathbf{x} + b_0 < 0$: predict class -1

[Rossbach]

Side Note: Including the Bias in the Weight Vector

- It is common to include the bias term b_0 into the definition of the weight vector \mathbf{w} , i.e., $\mathbf{w}^T := (b_0, b_1, \dots, b_d)$.
- In that case, all feature vectors are augmented with a constant 1, i.e., $\mathbf{x}^T := (1, x_1, x_2, \dots, x_d)$.
- Advantage of this definition of \mathbf{w} and \mathbf{x} : "automatically" includes bias term and thus $b_0 + b_1 x_1 + b_2 x_2 + \dots + b_d x_d = \mathbf{w}^T \mathbf{x}$ becomes a simple dot product.
- Neat trick to keep things compact and efficient (used in both classification and regression models). All model parameters (weights) of a trained model then captured in \mathbf{w} .
- Example: To predict target in a linear regression problem, only need to compute a simple dot product $\mathbf{w}^T \mathbf{x}'$, where \mathbf{x}' is the feature vector for the point you want a prediction for.
- Going forward, I will make explicit when the bias is included in the weight vector.

Exercise: Hyperplane



Hyperplane: $1 + 2X_1 + 3X_2 = 0$
 (blue is +1, red is -1)

- What is a weight vector (including bias) representing the hyperplane on the left?

Sol.: $\mathbf{w}^T = (1, 2, 3)$

- Are there other weight vectors corresponding to this hyperplane?

Sol.: Yes, $\mathbf{w}^T = c * (1, 2, 3)$ for any $c \in \mathbb{R}^+$

- Let a to-be-predicted data point's feature vector be at $X_1 = 0.5$ and $X_2 = -0.5$. What is $\mathbf{w}^T \mathbf{x}$ and what does that mean for the predicted label?

Sol.: Adding bias term, $\mathbf{x}^T = (1, 0.5, -0.5)$. Thus, $\mathbf{w}^T \mathbf{x} = 1*1+2*0.5+3*(-0.5) = 0.5$.

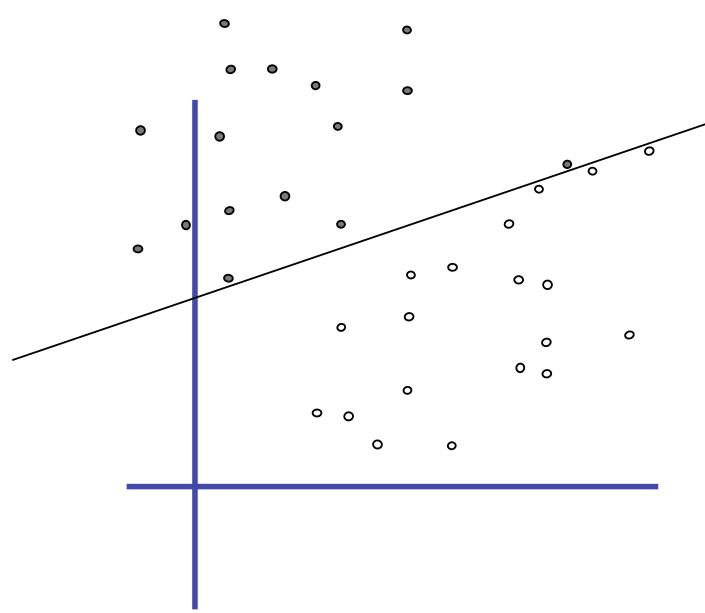
$0.5 > 0$, hence predicted label is +1.

- Let \mathbf{w} be the weight vector (including bias) for a hyperplane separating data points $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_n, y_n)$.

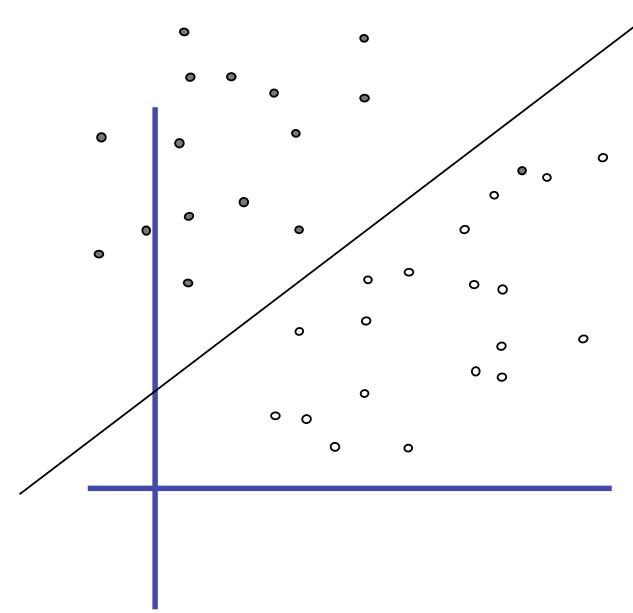
What can we say about $y_i \mathbf{w}^T \mathbf{x}_i$ for any i ?

Sol.: $y_i \mathbf{w}^T \mathbf{x}_i$ is positive for every correctly classified data point. Since \mathbf{w}^T is known to be a separating hyperplane for all points, it is positive for all.

Tricky Example



No misclassification but small margin

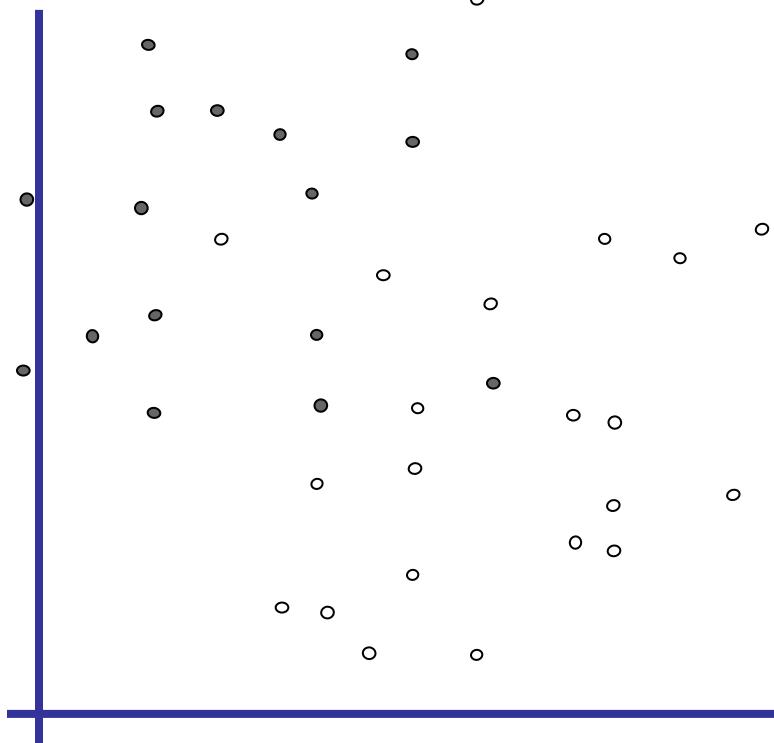


One misclassification but (ignoring that point) large margin

Why might you want to choose large margin? What is potential concern with small margin?

Overfitting to training data!

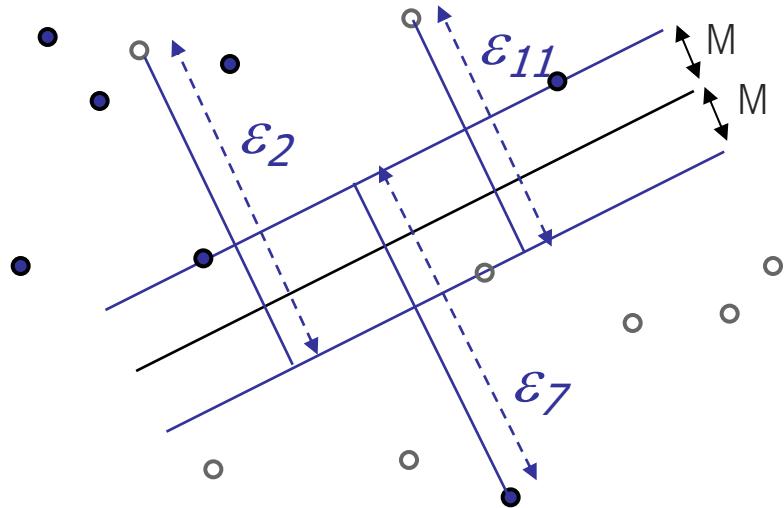
The Non-Separable Case



- Might not be able to linearly separate classes exactly
- Idea: extend concept of separating hyperplane to hyperplane that *almost* separates classes
- Might be worthwhile to misclassify a few training observations in order to do better job in classifying remaining points!

Maybe classifier can allow some misclassifications at a cost...

Soft Margin Classifier



Maximize M

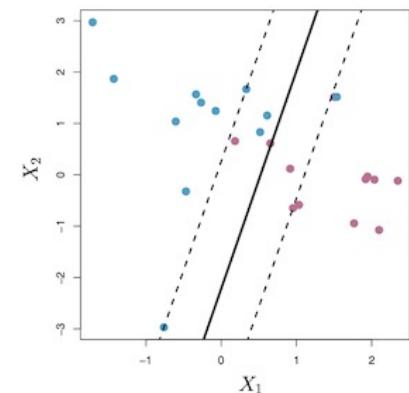
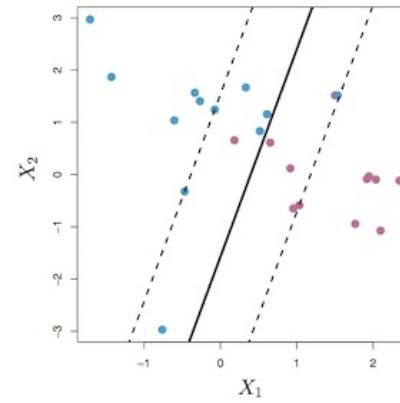
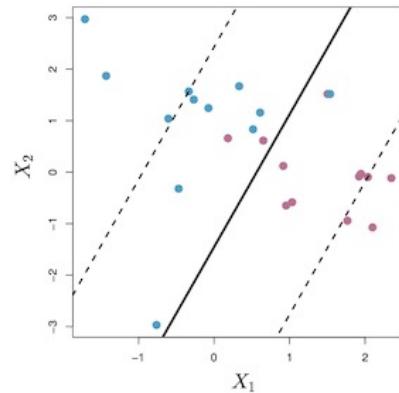
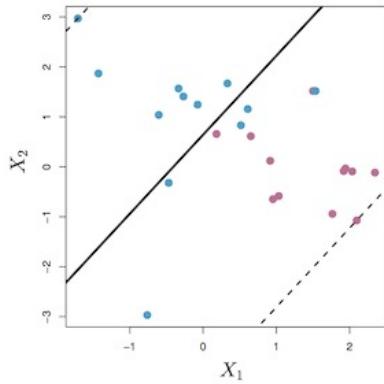
Subject to $\sum_{i=1}^n \varepsilon_i \leq B$

$$\varepsilon_i \geq 0$$

- Let $\varepsilon_1, \dots, \varepsilon_n$ be the “slack variables” that are positive only for points that are on the wrong side of the margin.
- Give classifier a “budget” B for the slack variables, which cannot be exceeded (i.e., budget for violations to the margin that we tolerate).
- Points that lie in the margin (including on its “borders”) or on the wrong side of the margin are called “support vectors.”
- Classifier’s decision based only on the support vectors: robust towards changes further away from hyperplane.

Behavior for Different Values of B

Four different values of B from large to small:

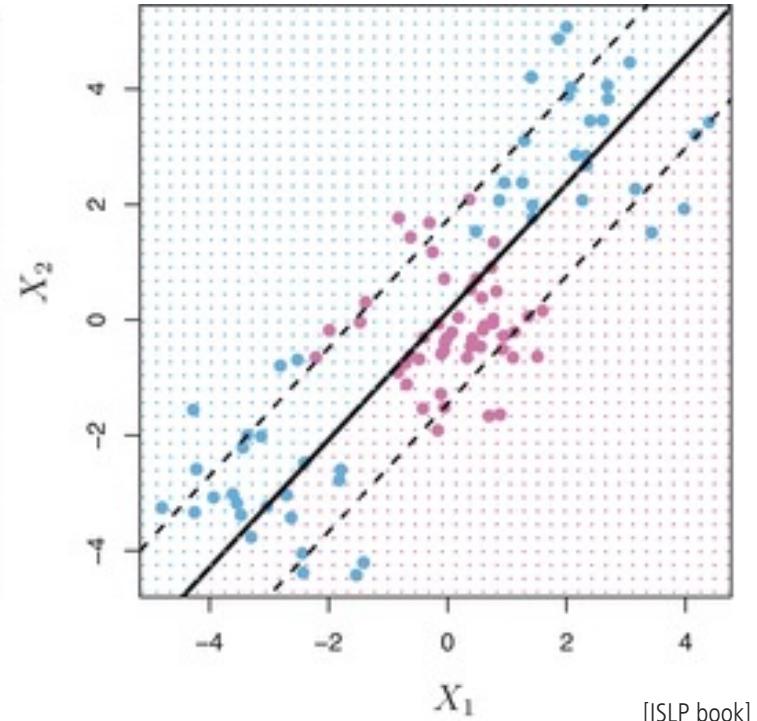
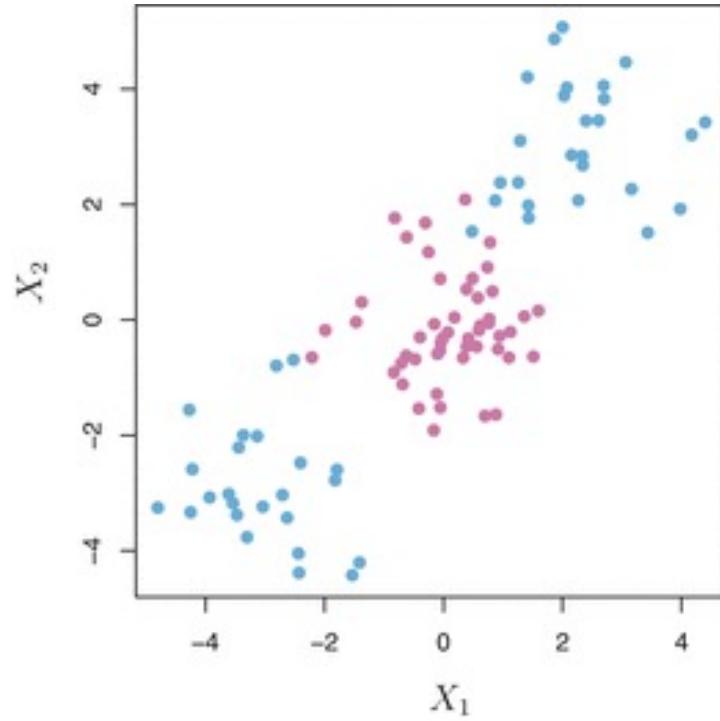


[ISLP book]

- Large B values lead to large margins with many violations but potentially better generalization error.
- Smaller B values allow less room for violations of the margin but might lead to overfitting.

How to choose parameter B?
Cross validation!

Non-Linear Class Boundaries: 2-D Example

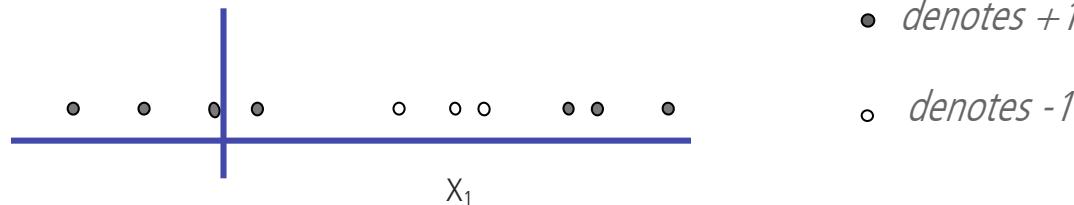


[ISLP book]

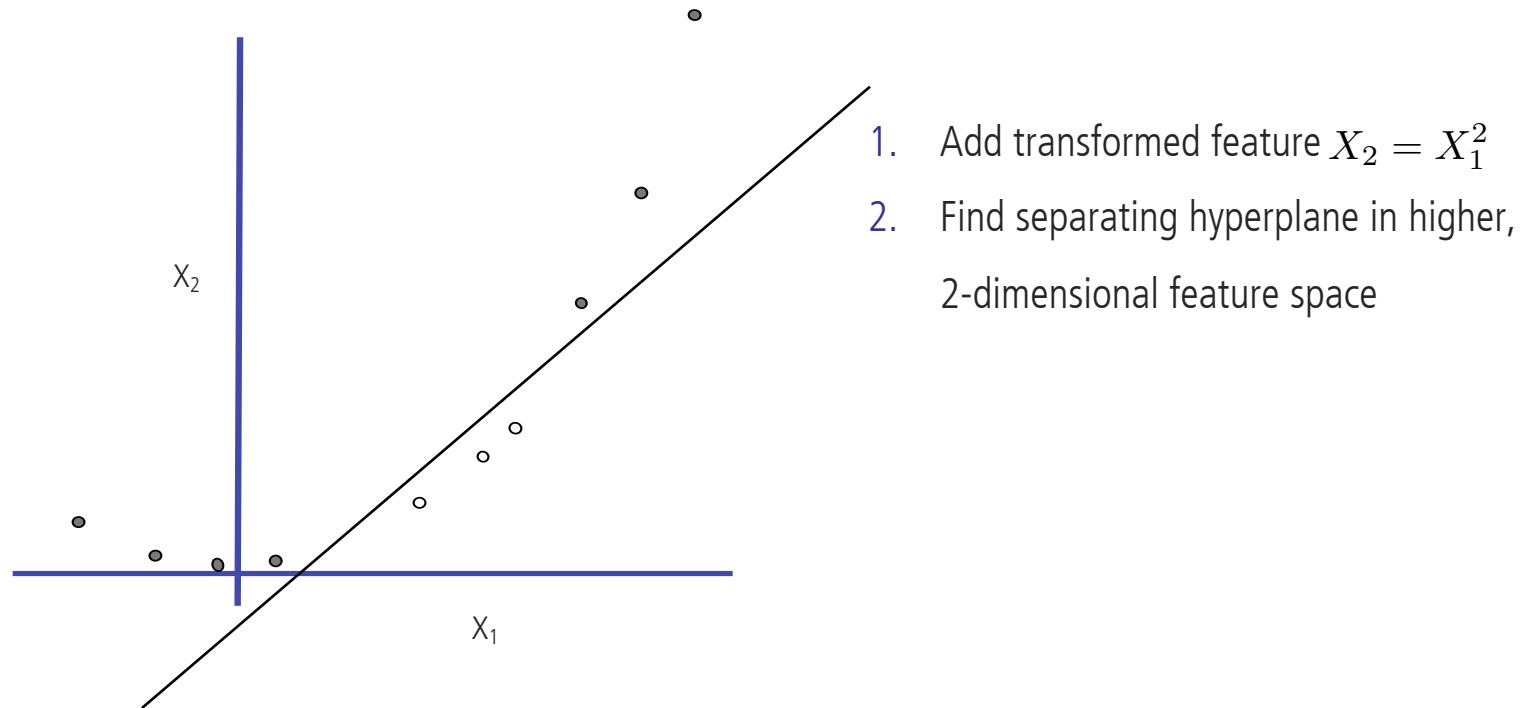
Linear soft-margin classifier performs poorly with non-linear class boundaries

Non-Linear Class Boundaries: 1-D Example (cont.)

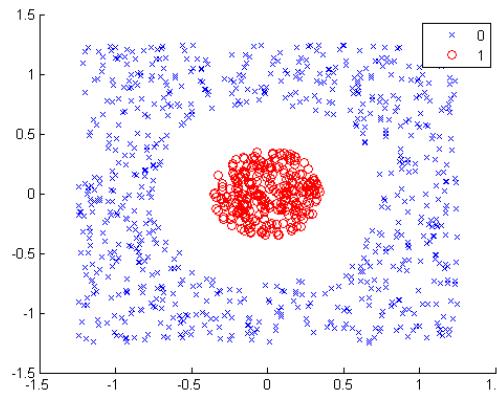
- No longer linearly separable:



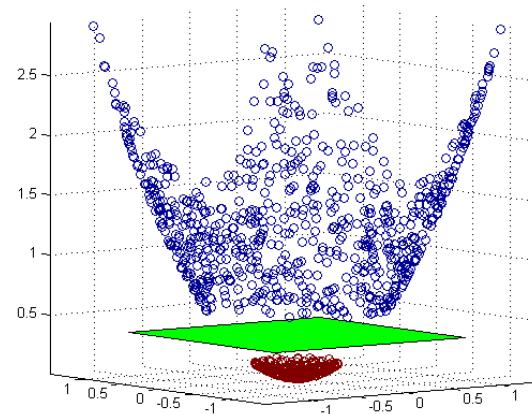
- Remember how we could transform features to obtain non-linear fits in regression? Let's do that here, too:



Feature Transformation 2D -> 3D



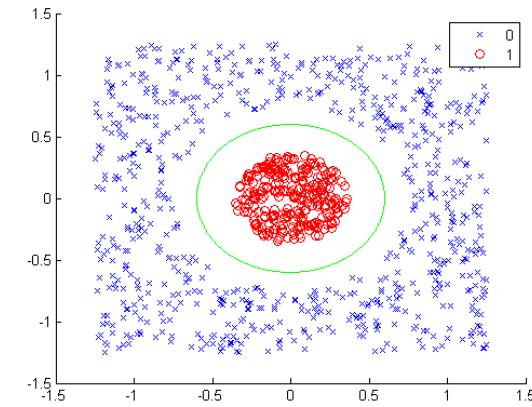
Data in original 2-D feature space



Data with added transformed feature

$$X_3 = X_1^2 + X_2^2$$

and separating hyperplane (green)
in enlarged 3-dimensional space



Resulting decision boundary in
original 2-D feature space

Can extend linear support vector classifier in same way we extended linear regression to fit non-linear functions by enlarging feature space with non-linear transformations of the original features!

Support Vector Machines (SVMs) are computationally more efficient approach to that same idea.

More Plotting

When we want to visualize classification datasets, it is helpful to plot points from different classes using different colors.

- The scatter function of matplotlib provides a functionality to do that:

```
plt.scatter(X1, X2, c=y, cmap=plt.cm.Accent)
```

Any distinct value in the array given in c will use a different color. Which color is determined by the colormap given in cmap. (You can find different colormaps online.)

- We can also change the size of the dots we are plotting using parameter s:

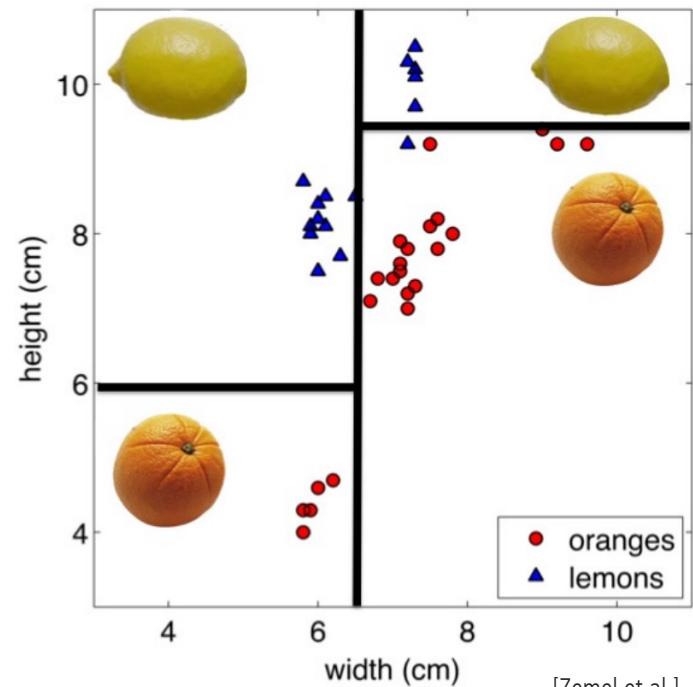
```
plt.scatter(X1, X2, s=70, c=y, cmap=plt.cm.Accent)
```

Python Practice

1. Download `SVC_pre.py` and import the SVM library using
`from sklearn.svm import SVC`. Note that the file includes a function `plot_svc`.
2. Create a small, random dataset with 20 data points, 10 for each class:
 - a) Use seed 10.
 - b) To generate `X`, first use `np.random.multivariate_normal` to generate 2 sets of 10 normally-distributed random numbers with centers (1,1) and (2,2), respectively, and the identity matrix as the covariance matrix.
 - c) To generate `y`, the (1,1) centered data points should be of class +1 and the (2,2) centered data points should be class -1.
3. Plot the data using the `plt.scatter` function and colormap `Paired`. Is the data linearly separable?
4. Create a Support Vector Classifier object using `svc1 = SVC(C= 1, kernel='linear')` and fit it using `svc1.fit(X, y)`. Plot the points and hyperplane of `svc1` using `plot_svc(svc1, X, y)`.
5. Scikit-Learn's `svc` package implements the support vector classifier slightly differently than what we have seen in the lecture. Instead of a budget `B` that can be used up for margin violations, it uses a parameter `C` with higher `C` implying less violations. That is, the higher the `C`, the lower the budget `B` (and vice versa). Increase the `C` parameter from 1 to 100, and 1000. How does the plot change?
6. Multiply the covariance matrices you used to generate the normal-distributed numbers by 0.1 and play around with `C` again. What changes?
7. Try out some of the non-trivial kernels (computationally more efficient versions of non-linear transformations), such as `poly` and `rbf` with different degrees for `poly` and different values of `C`. Eye-balling the results, which kernel seems to be working best here?

Decision Trees

- Non-linear methods used for both regression and classification
- Segment the feature space into simple regions
- Pro: Small trees can be interpreted fairly easily
- Con: Typically not as accurate as other methods (e.g., SVMs)
- But: often used as building block for very accurate ensemble methods (random forests / bagging / boosting)
- Today and next week:
 1. Regression Trees
 2. Classification Trees
 3. Ensemble Methods



[Zemel et al.]

Regression Trees: Example

Hitters dataset:

- Target: a baseball player's annual Salary (in thousands of dollars)
- Feature X_1 : Years (number of years he has played in major league)
- Feature X_2 : Hits (number of hits he made in previous year)

Preprocessing:

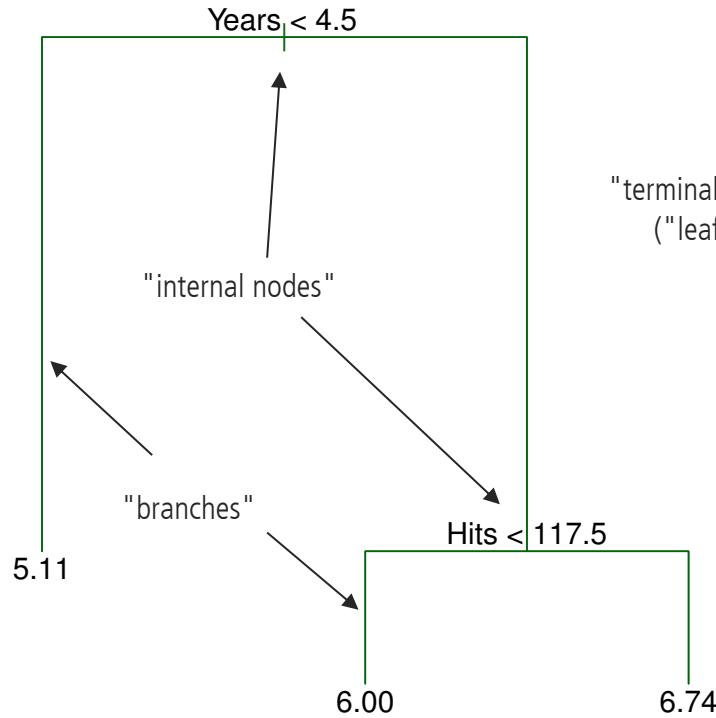
- Remove data points without Salary
- log-transform Salary (so it looks more like standard normal distribution)

NB: need to transform result back into original space at prediction time!

Regression tree fit to Hitters data:

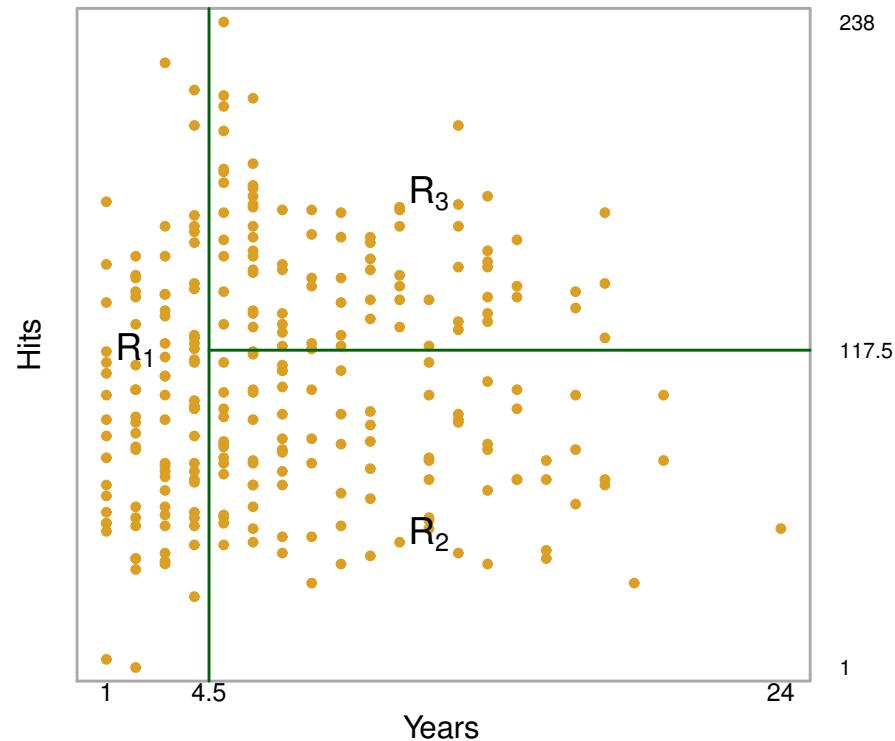


Regression Trees: Example (cont.)



Tree divides feature space into 3 regions:

$$\begin{aligned}
 R_1 &= \{X : \text{Years} < 4.5\} \\
 R_2 &= \{X : \text{Years} \geq 4.5, \text{Hits} < 117.5\} \\
 R_3 &= \{X : \text{Years} \geq 4.5, \text{Hits} \geq 117.5\}
 \end{aligned}$$



Prediction for (unseen) hitter:

- Player with less than 4.5 years experience:
 - predict log salary of 5.11, i.e., $e^{5.11}$ thousand dollars = 165.670 dollars.
- Player with at least 4.5 years experience further subdivided based on hits.

Growing a Regression Tree

- High level:
 1. Growing tree: Divide feature space into J distinct and non-overlapping regions R_1, \dots, R_J .
 2. Predicting w/ tree: Every point in R_j gets same prediction: mean of the target values in R_j .
- Step 2 easy once regions are defined.
- But how to construct regions R_1, \dots, R_J (Step 1)?
 - For simplicity: regions are (high-dimensional) rectangles ("boxes").
 - Goal: find boxes minimizing sum of squared error (SSE):

$$\sum_{j=1}^J \sum_{i \in R_j} (y_i - \hat{y}_{R_j})^2,$$

where \hat{y}_{R_j} is mean target value in R_j .

- Optimal solution computationally intractable.
- Alternative: use greedy* approach.

*widely-used concept in computer science: find optimal next step, ignoring impact on steps after next.

Greedy Approach: Recursive Binary Splitting

1. Select feature X_j and cutpoint s such that splitting feature space into regions $\{X : X_j < s\}$ and $\{X : X_j \geq s\}$ leads to greatest possible reduction in SSE. More precisely:

- a) For any j and s , define regions $R_1(j, s) = \{X : X_j < s\}$ and $R_2(j, s) = \{X : X_j \geq s\}$.
- b) Find j and s that minimize*

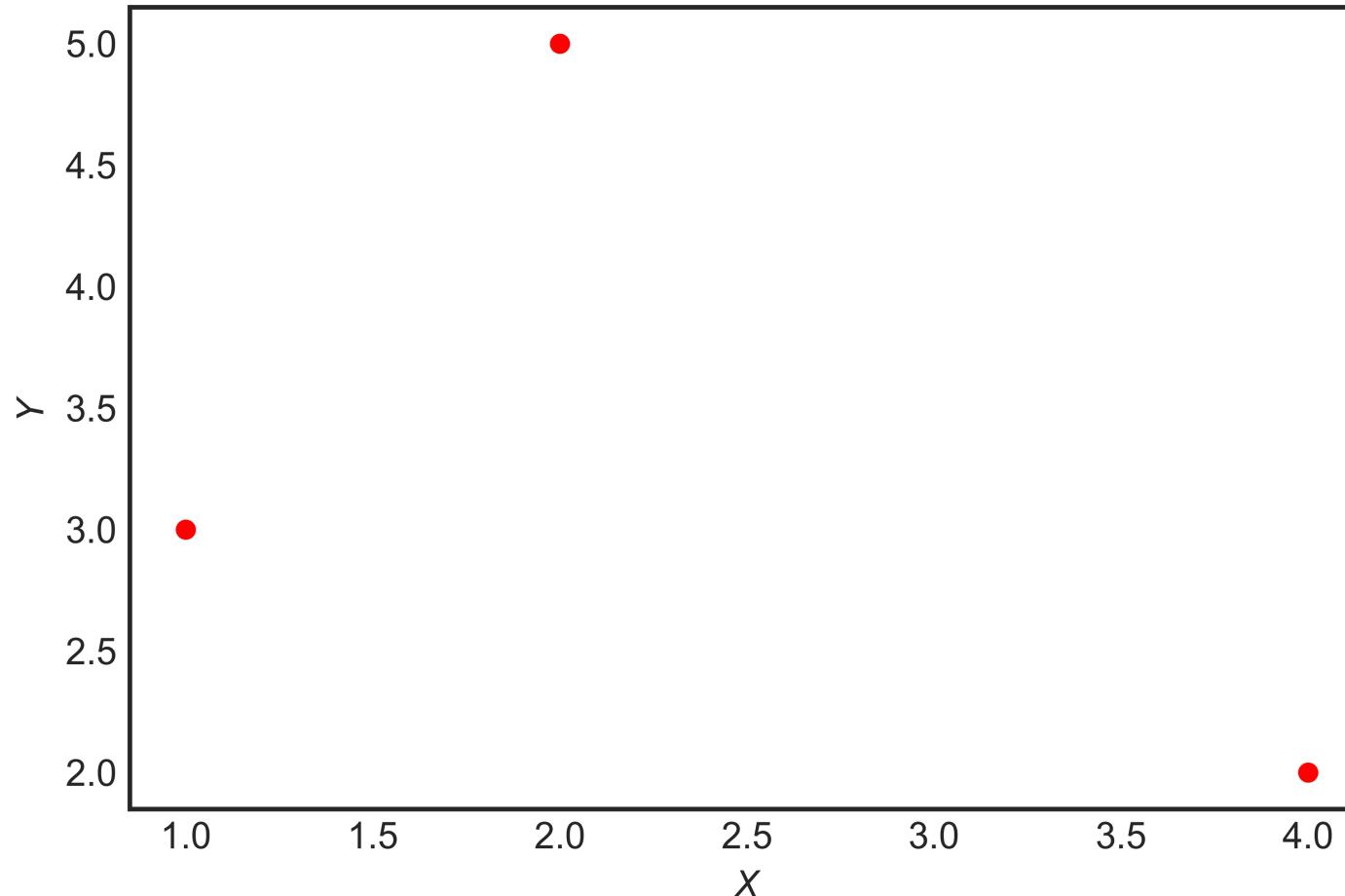
$$\sum_{i:x_i \in R_1(j,s)} (y_i - \hat{y}_{R_1})^2 + \sum_{i:x_i \in R_2(j,s)} (y_i - \hat{y}_{R_2})^2,$$

where \hat{y}_{R_j} is mean target value in R_j .

2. Repeat this process of finding best feature and cutpoint (now splitting already identified regions) until a stopping criterion is reached. For example, continue until no region contains more than 5 examples.
3. Once R_1, \dots, R_j have been created, predict target for a given test data point using the mean of the training data points in the region to which test data point belongs.

*Most implementations split at midpoints between two values and we will, too.

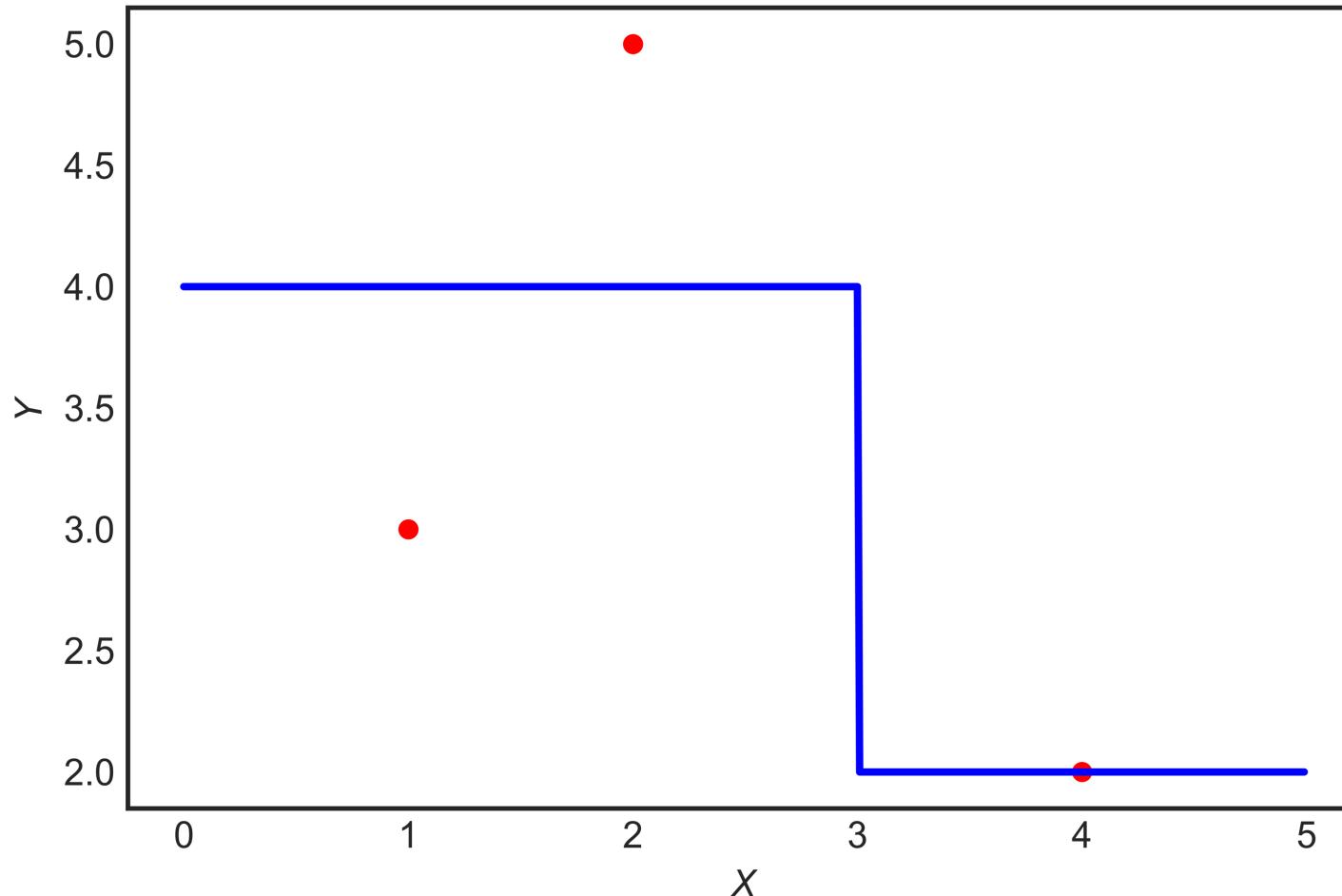
Exercise: Recursive Binary Splitting



1. What is the optimal binary split for this dataset and what is its SSE (on this training data)? Plot the predictions.
2. What is optimal split after that first one (with SSE on training data)? Also plot those predictions.

Exercise: Solution (1.)

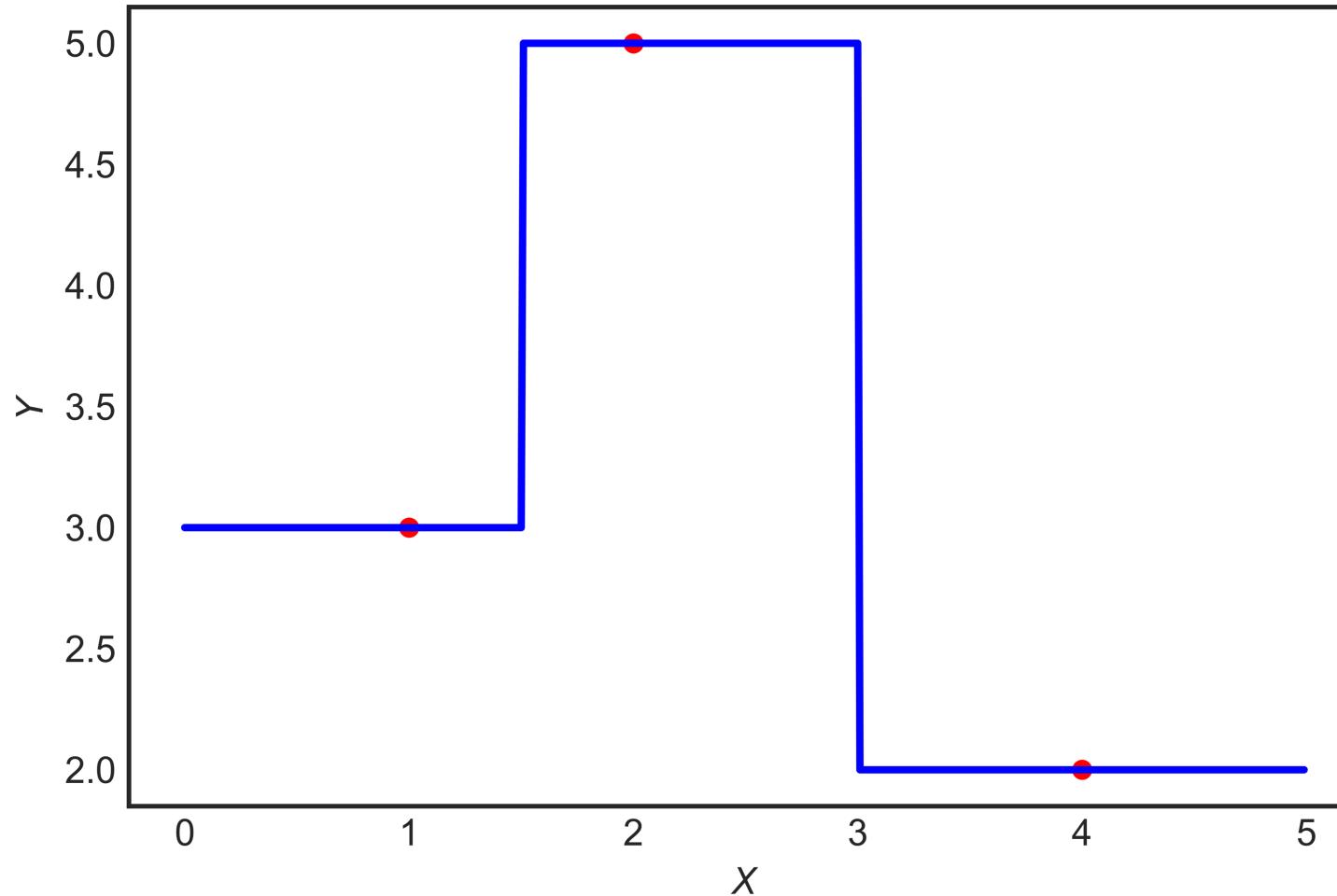
- Optimal binary split for this dataset is " $X > 3.0?$ " with $SSE = 1^2 + 1^2 + 0^2 = 2$. Plot of predictions:



NB: Split at " $X > 1.5?$ " would be worse with $SSE = 0^2 + 1.5^2 + 1.5^2 = 4.5$

Exercise: Solution (2.)

2. Optimal binary split following "no" to " $X > 3.0?$ " is " $X < 1.5?$ " with $SSE = 0^2 + 0^2 + 0^2 = 0$. Plot of predictions:



Credits

Some of the figures in this presentation are taken from Andrew W. Moore
(<http://www.cs.cmu.edu/~awm/tutorials.html>) with permission from the author.

Prof. Dr. Jens Witkowski

Assistant Professor of Computer Science and Management

Frankfurt School of Finance & Management
Adickesallee 32-34
60322 Frankfurt am Main
Germany

Email: j.witkowski@fs.de
Tel.: +49 (0) 69-154008-875
Fax: +49 (0) 69-154008-4875

Class 05

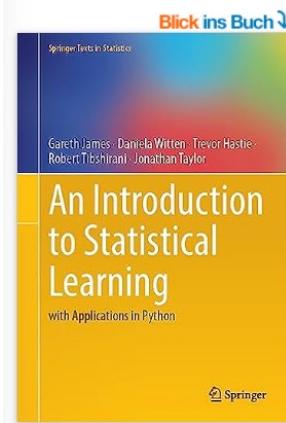
Decision Trees



Computational Statistics

Prof. Dr. Jens Witkowski
September 12, 2023

Literature Pointer



An Introduction to Statistical Learning: with Applications in Python (Springer Texts in Statistics)

Gebundene Ausgabe – 1. Juli 2023

Englisch Ausgabe | von [Gareth James](#) (Autor), [Daniela Witten](#) (Autor), [Trevor Hastie](#) (Autor), [Robert Tibshirani](#) (Autor), & 1 mehr

5,0 ★★★★☆ 1 Sternebewertung

[Alle Formate und Editionen anzeigen](#)

Gebundenes Buch 106,99 € prime
74,89 € prime

8 Neu ab 103,30 € 1 Neu ab 74,89 €

An Introduction to Statistical Learning provides an accessible overview of the field of statistical learning, an essential toolset for making sense of the vast and complex data sets that have emerged in fields ranging from biology to finance, marketing, and astrophysics in the past twenty years. This book presents some of the most important modeling and prediction techniques, along with relevant applications. Topics include linear regression, classification, resampling methods, shrinkage approaches, tree-based methods, support vector machines, clustering, deep learning, survival analysis, multiple testing, and more. Color graphics and real-world examples are used to illustrate the methods presented. This book is targeted at statisticians and non-statisticians alike, who wish to use cutting-edge statistical learning techniques to analyze their data.

Four of the authors co-wrote *An Introduction to Statistical Learning, With Applications in R* (ISLR), which has become a mainstay of undergraduate and graduate classrooms worldwide, as well as an important reference book for data scientists. One of the keys to its success was that each chapter contains a tutorial on implementing the analyses and methods presented in the R scientific computing environment. However, in recent years Python has become a popular language for data science, and there has been increasing demand for a Python-based alternative to ISLR. Hence, this book (ISLP) covers the same materials as ISLR but with labs implemented in Python. These labs will be useful both for Python novices, as well as experienced users.

[^ Weniger lesen](#)

[Falsche Produktinformationen melden.](#)

Dieses Bild anzeigen

Den Autoren folgen



Trevor
Hastie

Folgen



Gareth
James

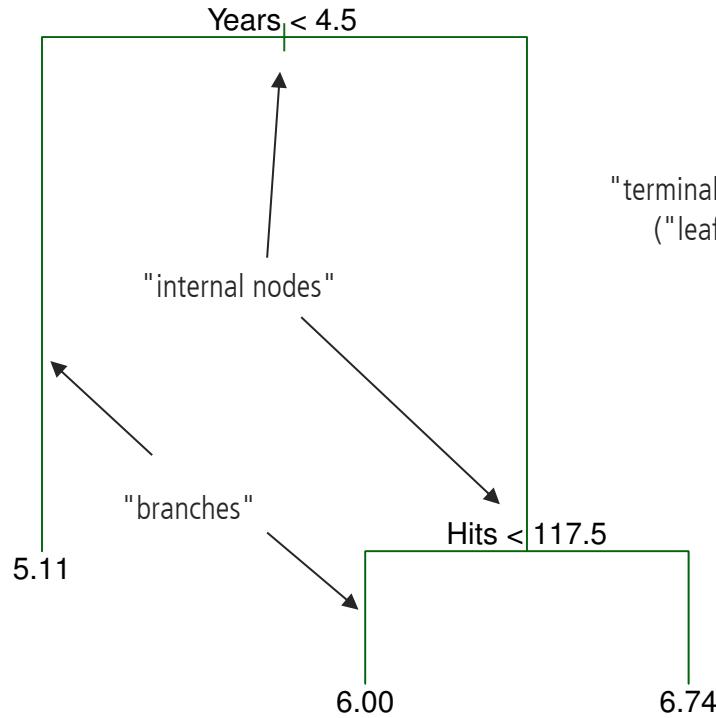
Folgen

- Recommended for Class 01 (Sep 4): Section 2.1
- Recommended for Class 02 (Sep 7): Section 3.1, 3.2, 7.1
- Recommended for Class 03 (Sep 7): Sections 5.1, 9.1-9.4
- Recommended for Class 04 (Sep 11): Sections 9.6, 8.1
- Recommended for Class 05 (today): Sections 8.1, 8.2

Agenda

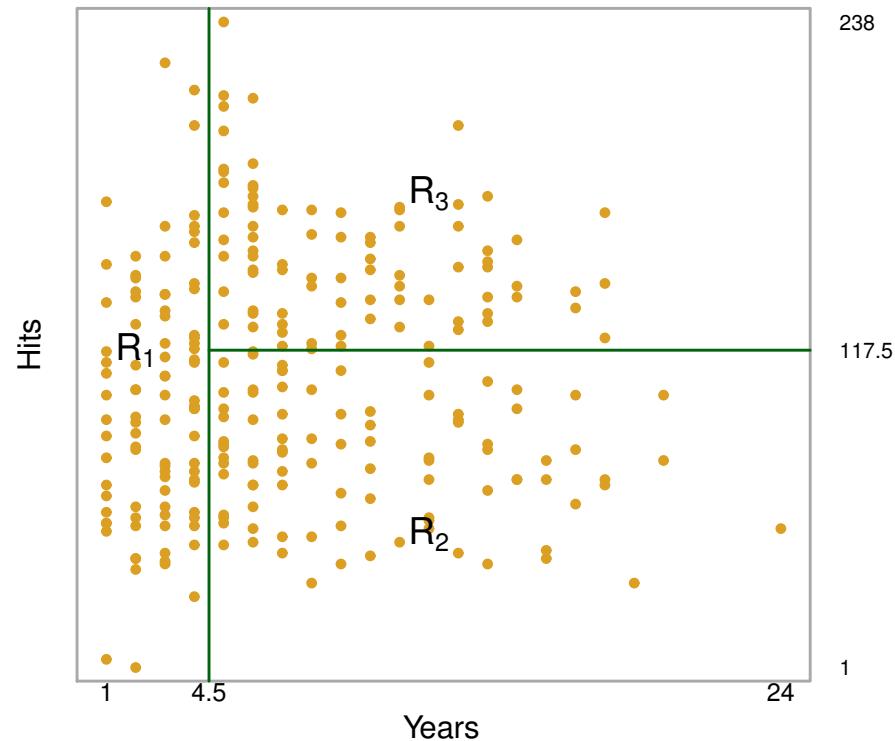
- Session 1
 - Decision Trees: Regression
- Session 2
 - Decision Trees: Classification

Regression Trees: Example (cont.)



Tree divides feature space into 3 regions:

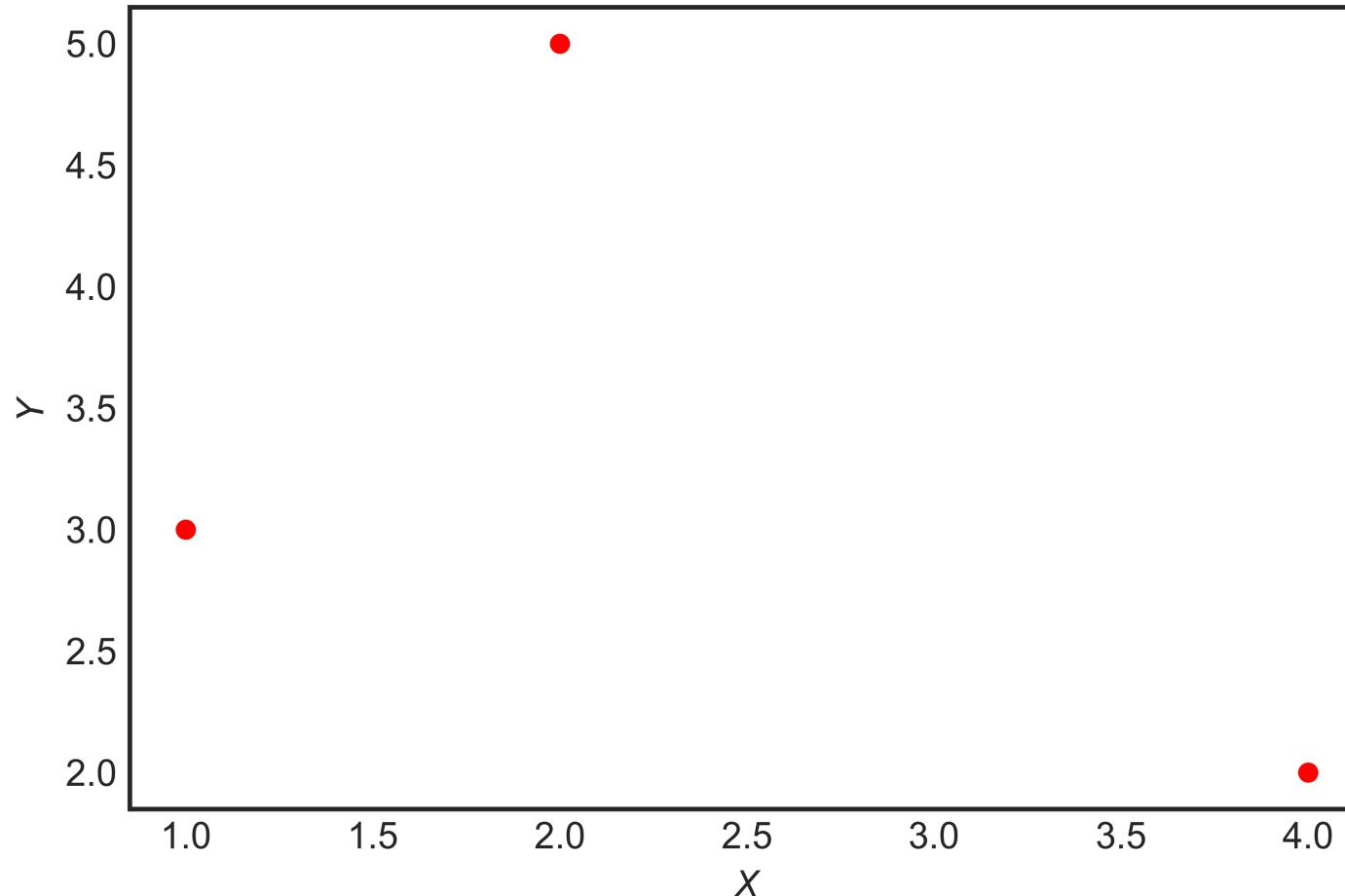
$$\begin{aligned}
 R_1 &= \{X : \text{Years} < 4.5\} \\
 R_2 &= \{X : \text{Years} \geq 4.5, \text{Hits} < 117.5\} \\
 R_3 &= \{X : \text{Years} \geq 4.5, \text{Hits} \geq 117.5\}
 \end{aligned}$$



Prediction for (unseen) hitter:

- Player with less than 4.5 years experience:
 - predict log salary of 5.11, i.e., $e^{5.11}$ thousand dollars = 165.670 dollars.
- Player with at least 4.5 years experience further subdivided based on hits.

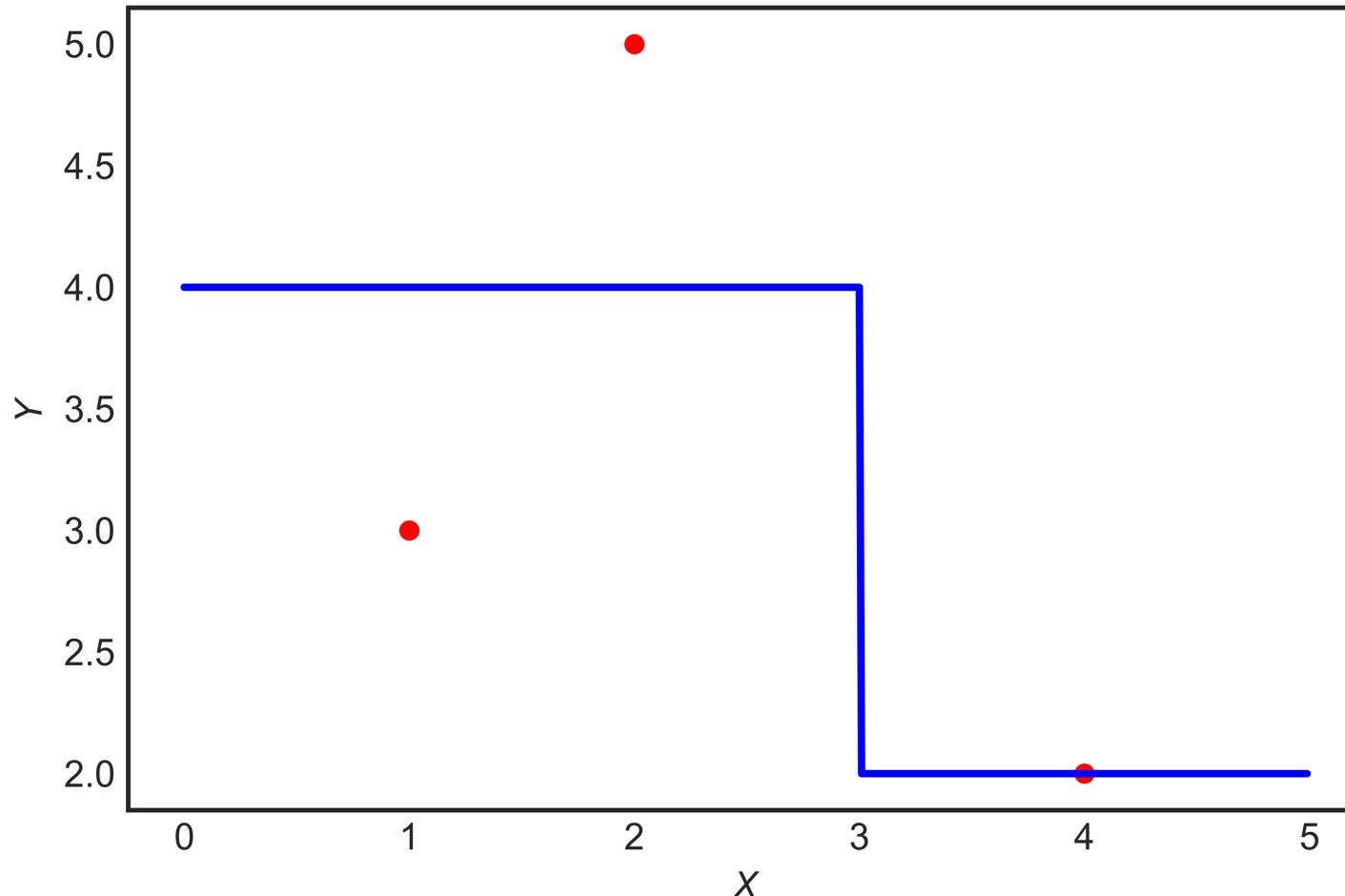
Exercise: Recursive Binary Splitting



1. What is the optimal binary split for this dataset and what is its SSE (on this training data)? Plot the predictions.
2. What is optimal split after that first one (with SSE on training data)? Also plot those predictions.

Exercise: Solution (1.)

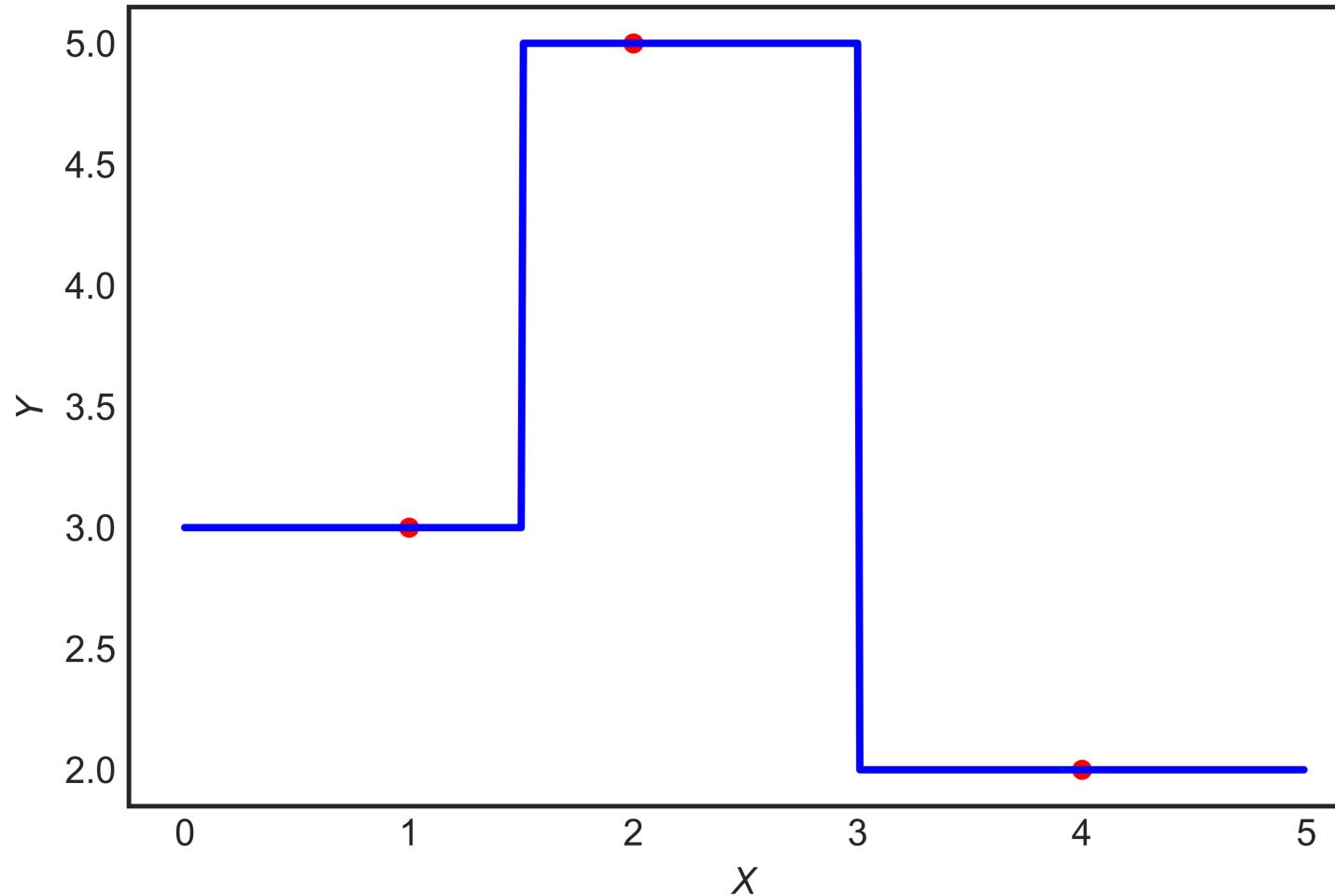
- Optimal binary split for this dataset is " $X > 3.0?$ " with $SSE = 1^2 + 1^2 + 0^2 = 2$. Plot of predictions:



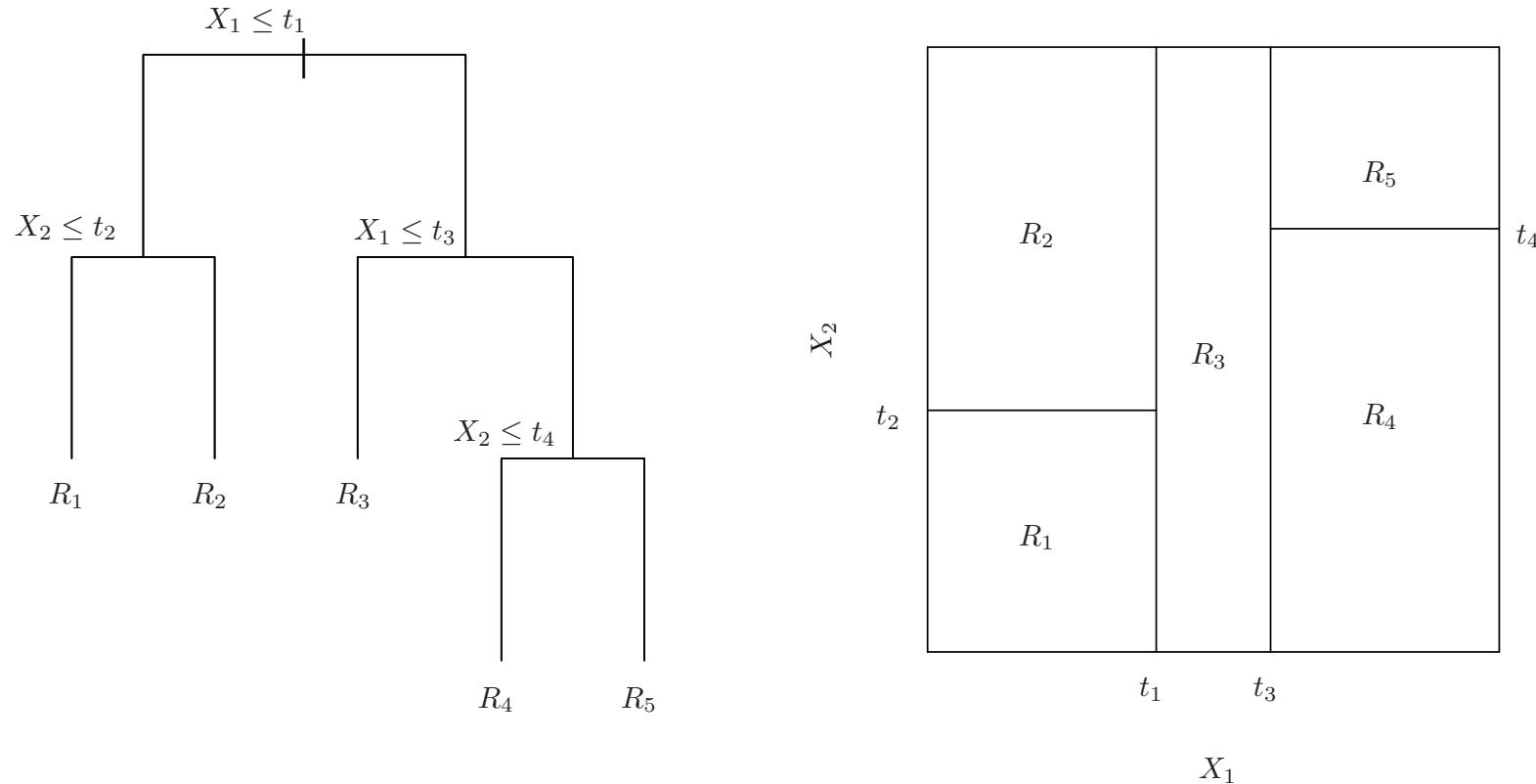
NB: Split at " $X > 1.5?$ " would be worse with $SSE = 0^2 + 1.5^2 + 1.5^2 = 4.5$

Exercise: Solution (2.)

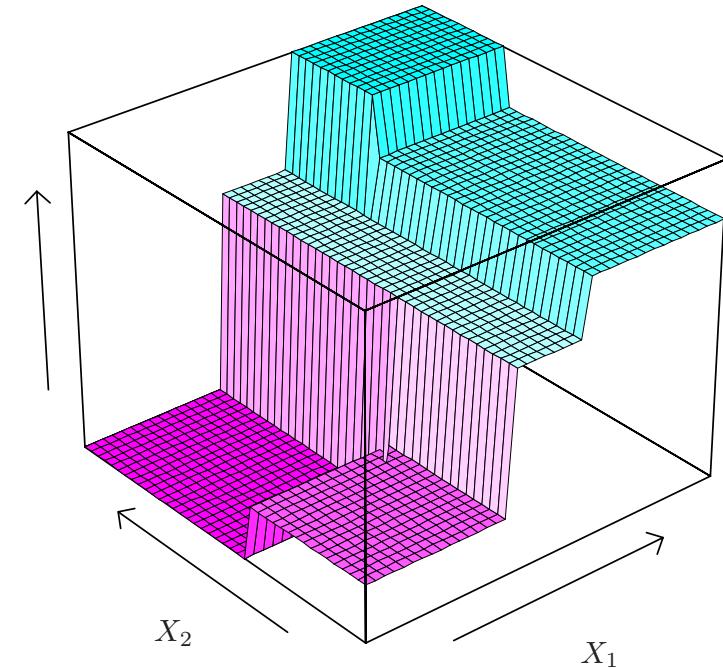
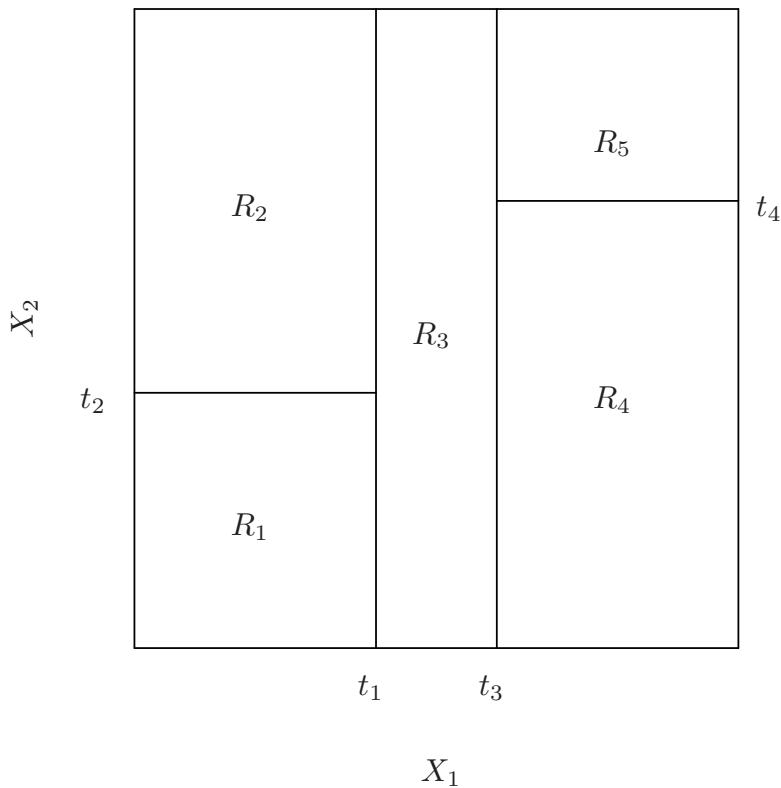
2. Optimal binary split following "no" to " $X > 3.0?$ " is " $X < 1.5?$ " with $SSE = 0^2 + 0^2 + 0^2 = 0$. Plot of predictions:



Recursive Binary Splitting: 2D Example

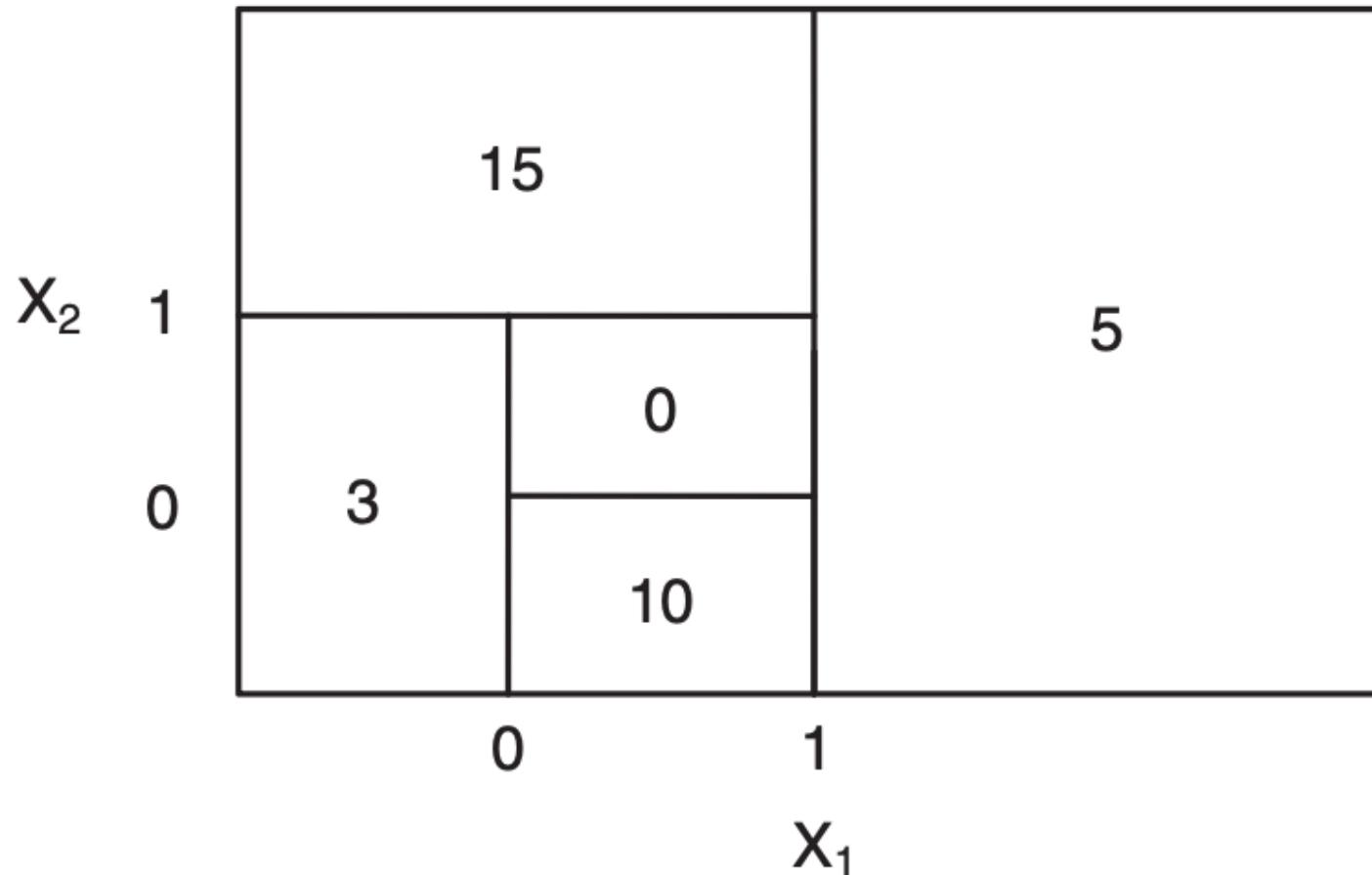


Recursive Binary Splitting: 2D Example (cont.)



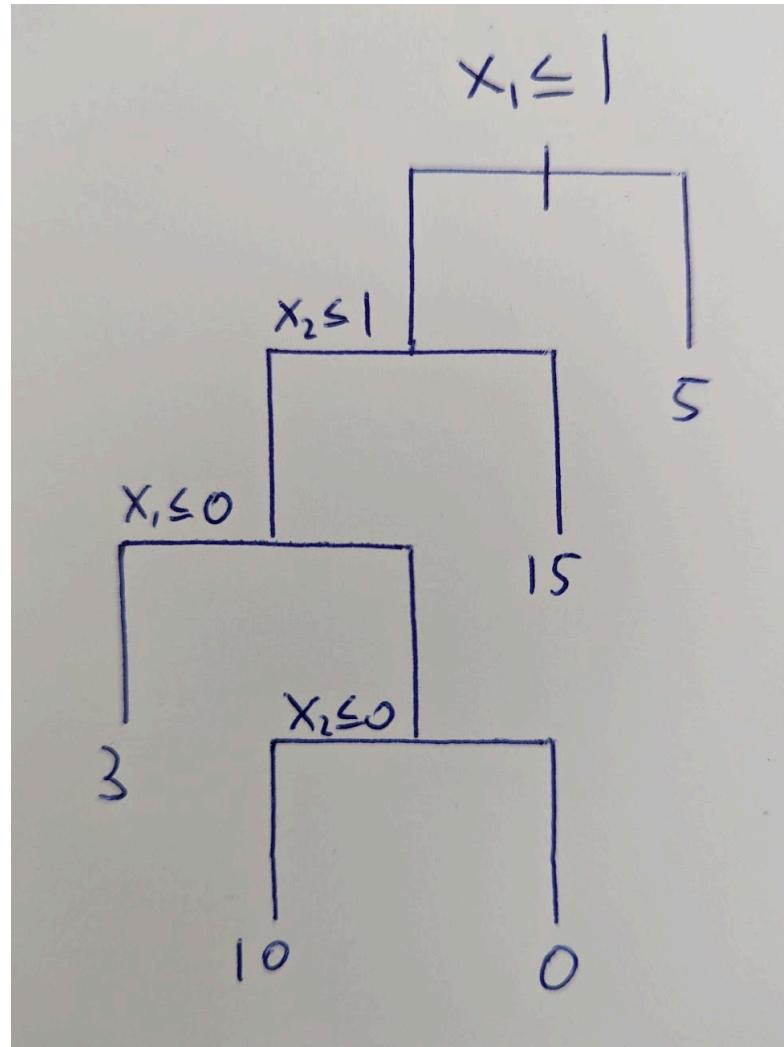
Exercise: Decision Trees

Sketch the tree corresponding to the partition of the feature space (using " \leq " and branching left if condition is true). The numbers inside the boxes indicate the mean of Y within each region:



Exercise: Decision Trees

Solution:



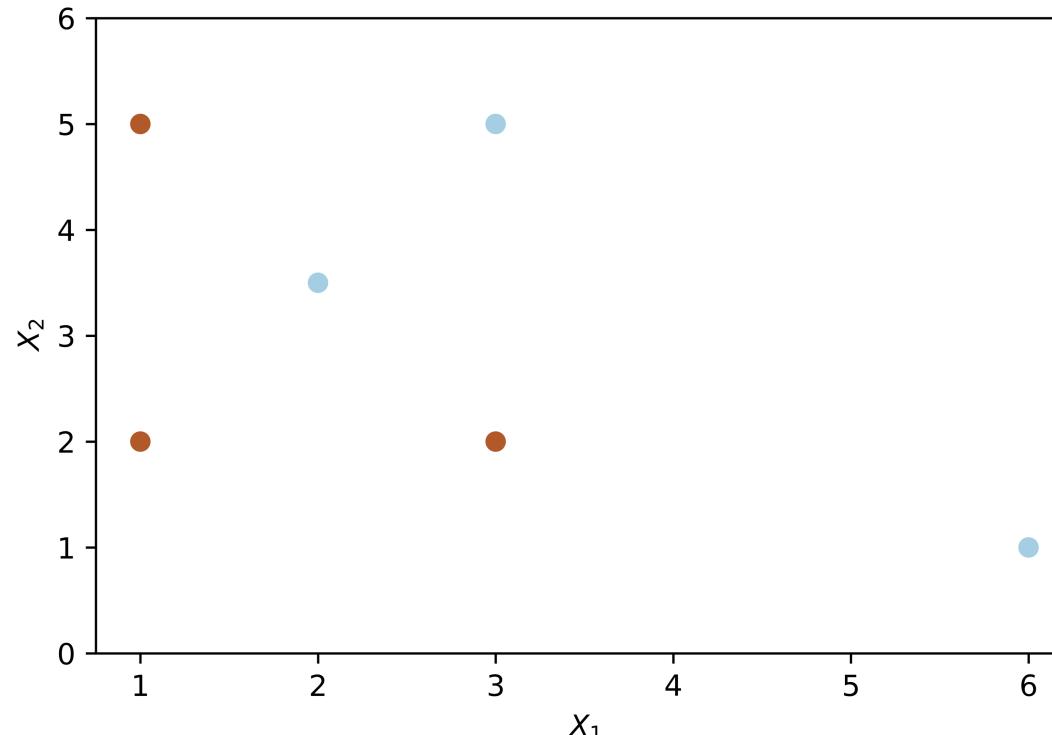
Python: Regression Trees

1. Load validation_set.py from Canvas (Class 03) and save it under regression_tree.py.
2. Add `from sklearn.tree import DecisionTreeRegressor` at the beginning.
3. Create a DecisionTreeRegressor object using `tree_regr = DecisionTreeRegressor()`.
4. Fit the model using the object's `fit` method.
5. Plot `tree_regr`'s predictions in yellow and also compute and print out its validation MSE (analogous to the other models). What do you see?
6. Do some research as to how the stopping criteria can be changed. Set it to "no region with 4 or fewer data points is split." What is the stopping criterion when none is given?

Classification Trees

Classification trees are very similar to regression trees, but:

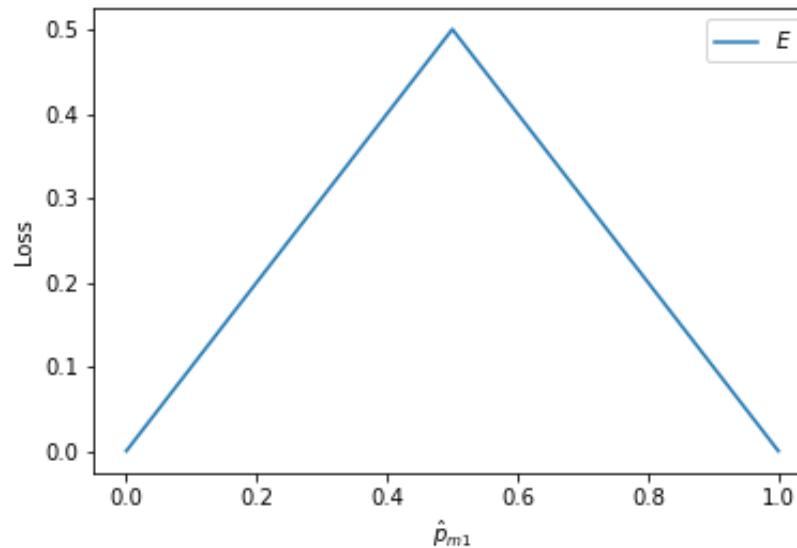
- At prediction time: each data point is predicted to belong to most commonly occurring class in training data of the region (instead of mean target value in training data lying in the region as in regression).
- When growing tree (i.e., deciding on splits), SSE no longer appropriate loss function. Let's consider 0/1 loss as a candidate for classification tree growing. What would recursive binary splitting look like here:



0/1 Loss ("Misclassification Error")

- Let \hat{p}_{mk} be the proportion of training data points in region m that are of class k ($K \geq 2$ classes).
- Predicted label for region m is $\hat{y} = \arg \max_k \hat{p}_{mk}$
- Fraction of misclassified points in region m is then $E = 1 - \max_k \hat{p}_{mk}$

For K=2 classes (plot for \hat{p}_{m2} is identical):



- Summed up over all regions and weighted by number of points N_m in region m to obtain overall error:

$$\sum_{m=1}^M N_m \left(1 - \max_k \hat{p}_{mk}\right)$$

Gini Index and Entropy

- 0/1 loss would be natural candidate for growing decision trees but runs into numerical issues.
- Two popular alternative measures to evaluate quality of particular split are Gini index and entropy.

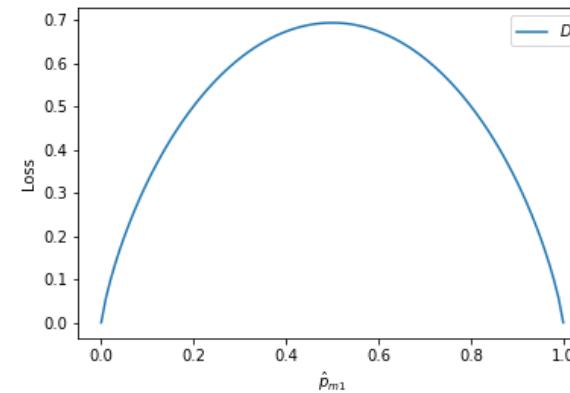
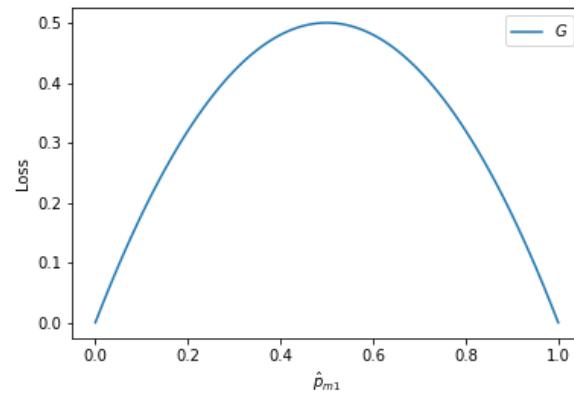
Gini index:

$$G = \sum_{k=1}^K \hat{p}_{mk} (1 - \hat{p}_{mk})$$

Entropy:

$$D = - \sum_{k=1}^K \hat{p}_{mk} \ln \hat{p}_{mk}$$

For K=2 classes (plot for \hat{p}_{m2} is identical):



Both take on small values if all \hat{p}_{mk} are close to either 0 or 1 (here shown for K=2).

As before, need to sum over all regions weighted by number of points in regions to obtain overall error.

Measures of "Impurity"

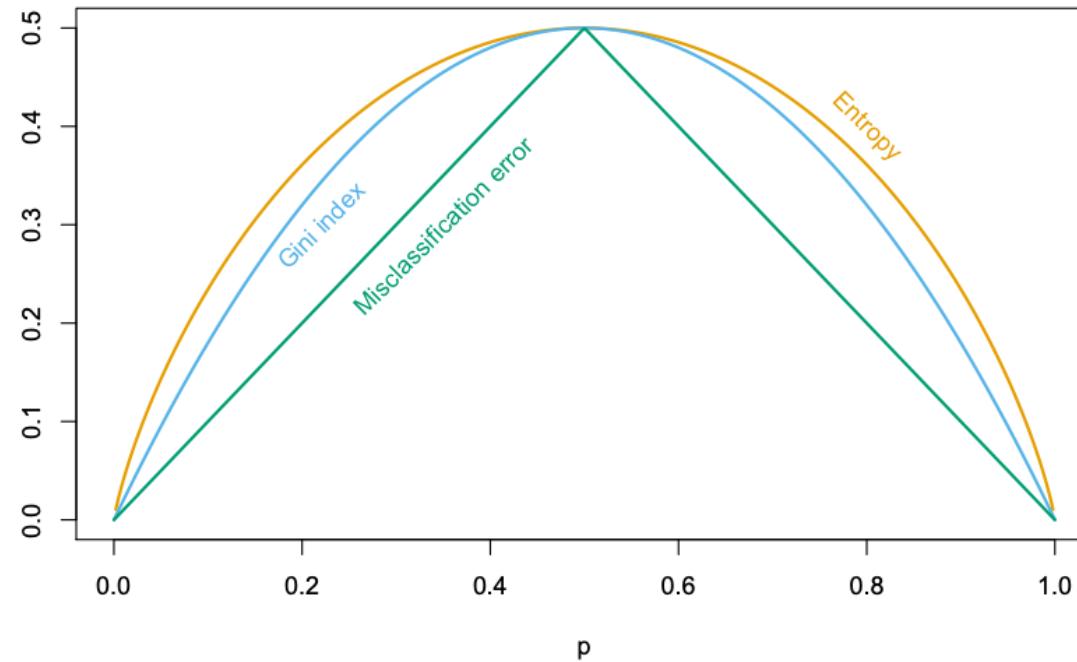


FIGURE 9.3. Node impurity measures for two-class classification, as a function of the proportion p in class 2. Cross-entropy has been scaled to pass through $(0.5, 0.5)$.

[Hastie et al.]

Exercise: Gini Index and Entropy

Let \hat{p}_{mk} be the proportion of training data points in region m that are of class k. Compute Gini index and entropy for K = 3 with $\hat{p}_{m1} = 0.2$, $\hat{p}_{m2} = 0.3$, and $\hat{p}_{m3} = 0.5$:

Solution:

- Gini index $G = 0.2*0.8 + 0.3*0.7 + 0.5*0.5 = 0.62$
- Entropy: $D = -0.2*\ln(0.2) - 0.3*\ln(0.3) - 0.5*\ln(0.5) = 1.03$

Python: Classification Trees

1. Download SVC_post.py from Canvas and save it under classification_tree.py.
2. Add `from sklearn.tree import DecisionTreeClassifier` at the beginning.
3. Create a `DecisionTreeClassifier` object using `tree_classifier = DecisionTreeClassifier()` and fit the model using the object's `fit` method.
4. Add `from sklearn import tree` at the beginning and plot the tree using `tree.plot_tree(tree_classifier)`.
5. Limit the number of `tree_classifier`'s leaf nodes to 2. Does the resulting split seem meaningful?
6. Set the number of `tree_classifier`'s leaf nodes to 3. Does the resulting split seem meaningful? Use entropy as the splitting criterion. What do you see?
7. Create a plot containing both the data points and the predicted labels from the DecisionTree.
 1. Create a 2D array `X_seq`, representing a grid of 100x100 points. For both dimensions, the grid should go from the minimal to the maximal value in your dataset.
 2. Save the predicted labels of your decision tree in a variable `predicted_labels` and plot the predicted labels for each point on the grid.

Credits

Some of the figures in this presentation are taken from Andrew W. Moore
(<http://www.cs.cmu.edu/~awm/tutorials.html>) with permission from the author.

Prof. Dr. Jens Witkowski
Assistant Professor of Computer Science and Management

Frankfurt School of Finance & Management
Adickesallee 32-34
60322 Frankfurt am Main
Germany

Email: j.witkowski@fs.de
Tel.: +49 (0) 69-154008-875
Fax: +49 (0) 69-154008-4875

Class 06

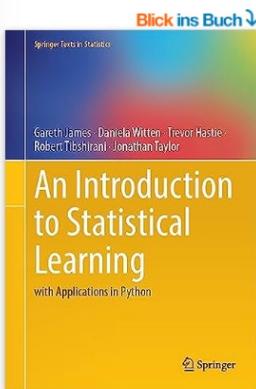
Ensemble Methods



Computational Statistics

Prof. Dr. Jens Witkowski
September 14, 2023

Literature Pointer



An Introduction to Statistical Learning: with Applications in Python (Springer Texts in Statistics)

Gebundene Ausgabe – 1. Juli 2023

Englisch Ausgabe | von Gareth James (Autor), Daniela Witten (Autor), Trevor Hastie (Autor), Robert Tibshirani (Autor), & 1 mehr

5,0 ★★★★★ 1 Sternebewertung Alle Formate und Editionen anzeigen

Gebundenes Buch
106,99 € prime
8 Neu ab 103,30 €

Taschenbuch
74,89 € prime
1 Neu ab 74,89 €

An Introduction to Statistical Learning provides an accessible overview of the field of statistical learning, an essential toolset for making sense of the vast and complex data sets that have emerged in fields ranging from biology to finance, marketing, and astrophysics in the past twenty years. This book presents some of the most important modeling and prediction techniques, along with relevant applications. Topics include linear regression, classification, resampling methods, shrinkage approaches, tree-based methods, support vector machines, clustering, deep learning, survival analysis, multiple testing, and more. Color graphics and real-world examples are used to illustrate the methods presented. This book is targeted at statisticians and non-statisticians alike, who wish to use cutting-edge statistical learning techniques to analyze their data.

Four of the authors co-wrote *An Introduction to Statistical Learning, With Applications in R* (ISLR), which has become a mainstay of undergraduate and graduate classrooms worldwide, as well as an important reference book for data scientists. One of the keys to its success was that each chapter contains a tutorial on implementing the analyses and methods presented in the R scientific computing environment. However, in recent years Python has become a popular language for data science, and there has been increasing demand for a Python-based alternative to ISLR. Hence, this book (ISLP) covers the same materials as ISLR but with labs implemented in Python. These labs will be useful both for Python novices, as well as experienced users.

[^ Weniger lesen](#)

[Falsche Produktinformationen melden.](#)

Dieses Bild anzeigen

Den Autoren folgen



Trevor
Hastie

[Folgen](#)



Gareth
James

[Folgen](#)

- Recommended for Class 01 (Sep 4): Section 2.1
- Recommended for Class 02 (Sep 7): Section 3.1, 3.2, 7.1
- Recommended for Class 03 (Sep 7): Sections 5.1, 9.1-9.4
- Recommended for Class 04 (Sep 11): Sections 9.6, 8.1
- Recommended for Class 05 (Sep 12): Section 8.1
- Recommended for Class 06 (today): Section 8.2

Agenda

- Session 1
 - Bagging
- Session 2
 - Random Forests

Bagging

- "Bagging" is short for "**bootstrap aggregating**" and is an example of an ensemble method. Ensemble methods combine several models into one "super" model, often achieving higher accuracy.
- General-purpose procedure : can be applied to a variety of different types of base models to reduce "variance." Roughly speaking: a model class suffers from high variance if splitting data in half and training two models – one for each half – tends to lead to very different predictions.
- Often used in conjunction with decision trees because individual trees tend to suffer from high variance and overfitting.
- Bagging procedure (B is a parameter):
 1. Randomly sample B new datasets from training dataset (of same size as training dataset and with replacement).
 2. Train B models, one for each of these new datasets.
 3. Use "average" of all B base model predictions as prediction.
(literal averaging for regression and majority vote for classification)
- Large B does not lead to overfitting. Choose B sufficiently large for test error to have settled down.

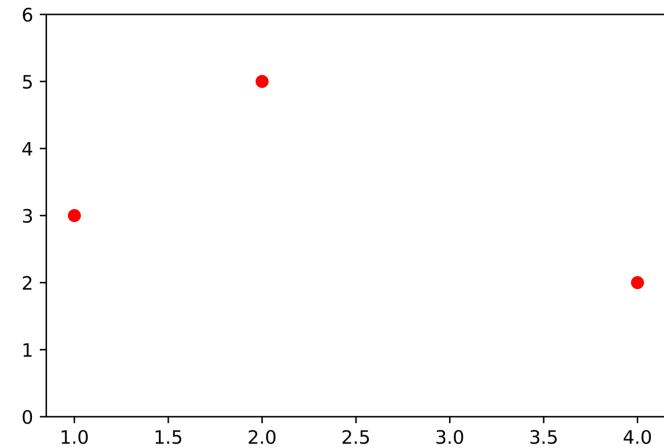
Exercise: Bagging with (Regression) Trees

1. Randomly sample $B=2$ new (artificial) datasets with replacement from the training dataset consisting of the 3 data points depicted on the right.

Solution (many others are possible):

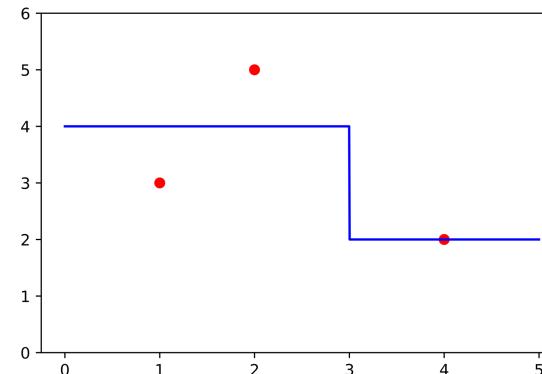
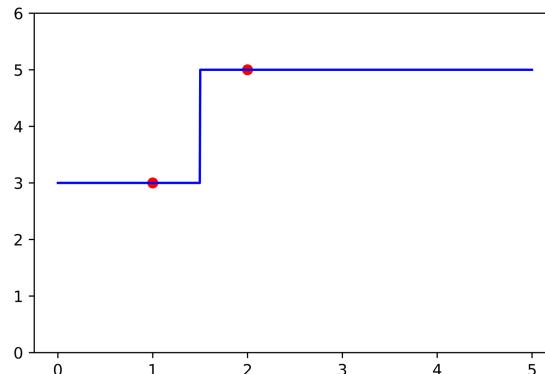
Dataset 1: $(x_1, y_1) = (1.0, 3.0)$, $(x_2, y_2) = (1.0, 3.0)$, $(x_3, y_3) = (2.0, 5.0)$

Dataset 2: $(x_1, y_1) = (2.0, 5.0)$, $(x_2, y_2) = (4.0, 2.0)$, $(x_3, y_3) = (1.0, 3.0)$



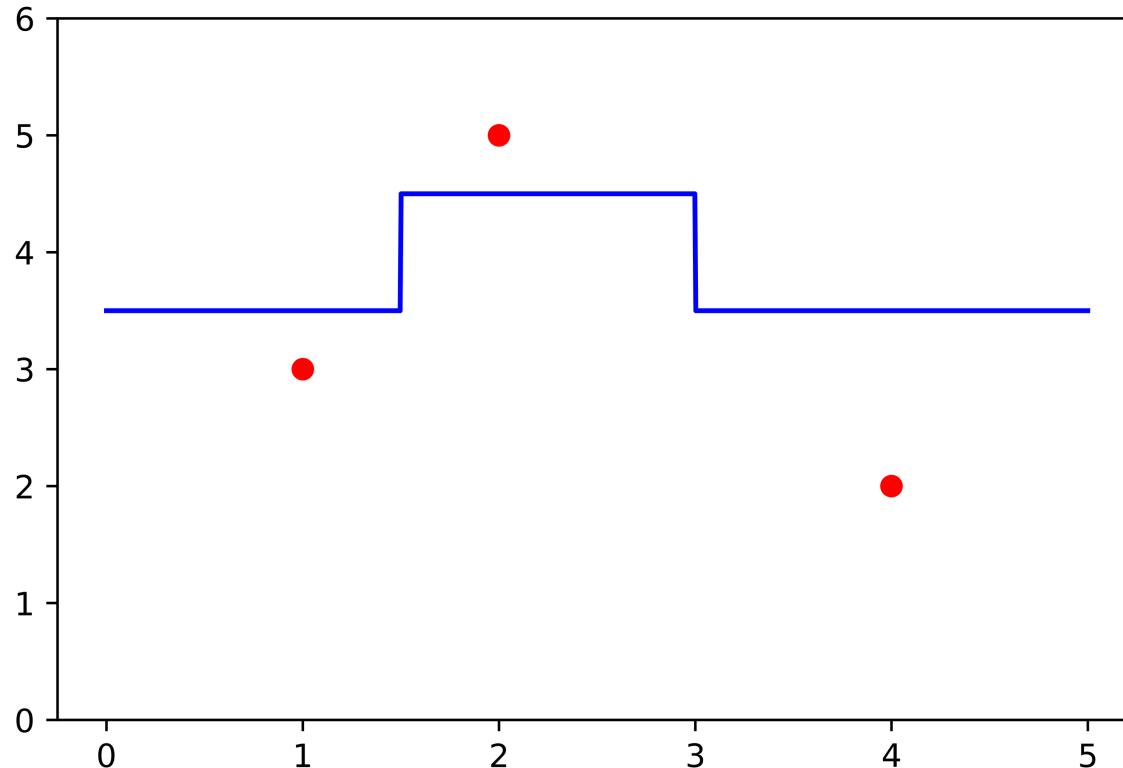
2. Train $B=2$ single-split models, one for each of these new datasets.

Draw the predictions for both models and all X :



Exercise: Bagging with (Regression) Trees

3. Draw the average of the 2 base model predictions (the ensemble prediction).



Python: Bagging Trees

1. Go to `regression_tree.py` from Canvas (Class 05) and save it under `bagging_regression_tree.py`.
2. Add `from sklearn.ensemble import BaggingRegressor` at the beginning.
3. Create a `BaggingRegressor` object using
`bagged_regression_tree = BaggingRegressor(base_estimator=DecisionTreeRegressor())`
and fit the model using the object's `fit` method.
4. Plot the model's prediction in brown and calculate its MSE.
5. Play around with the number of base models (`n_estimators`) and the `max_depth` of the base models.
What do you notice when you compare the `DecisionTreeRegressor` and the `BaggingRegressor` in terms of MSE (when created with same parameters)?
6. Set `train_size` of `train_test_split` to 1%. Plot again.

Random Forests

- Improvement over bagged trees through decorrelation of base trees.
- Procedure:
 1. As in bagged trees, randomly sample with replacement to create new datasets.
 2. When growing the base trees, however, any split can only pick from a random sample of m features from the full set of p features. A fresh sample is taken each split and we typically choose $m \approx \frac{p}{3}$ for regression and $m \approx \sqrt{p}$ for classification.
- Intuition:
 - Suppose there exists a strong feature in dataset along with moderately strong features.
 - Most or all the base trees in (regular) bagging will have that feature at root.
 - Hence, base trees will look very similar to each other and predictions will be highly correlated.
 - Problem with that: averaging correlated quantities does not lead to substantial variance reduction.
 - Random forests overcome this problem by ignoring the strong feature in most splits and other features will take its spot, leading to de-correlation of base trees.

Python: Random Forests

1. Go to `bagging_regression_tree.py` from earlier today and save it as `random_forest.py`.
2. Add `from sklearn.ensemble import RandomForestRegressor` at the beginning.
3. Create a `RandomForestRegressor` object using

```
RF_regression = RandomForestRegressor()
```

and fit the model using the object's `fit` method.

4. Plot the model's prediction and calculate its MSE.
5. Play around with the number of base models (`n_estimators`) and the `max_depth` of the base models.

What do you notice when you compare the `DecisionTreeRegressor`, `BaggingRegressor`, and `RandomForestRegressor` in terms of MSE (when created with same parameters)? What would you have expected?

Out-Of-Bag (OOB) Error

Clever trick to get test error "for free" in bagging and random forests (no need to do cross validation):

- Remember: sampling with replacement.
- On average: each base tree uses only around 2/3 of data points; 1/3 of data points not used in training.
- Idea: use those data points not used in training to evaluate the ensemble "out of sample":
 - For each data point (x_i, y_i) , evaluate loss by averaging predictions from those trees corresponding to new datasets in which (x_i, y_i) was not sampled.
- Can be shown: For large B , OOB error essentially equivalent to LOOCV error!

Exercise: Let h_b be the regressor tree trained on dataset b and let $h_b(x_i)$ be its prediction for point (x_i, y_i) .

Your $B = 3$ bootstrapped datasets contain the following points. Give the calculation for the OOB MSE.

$$D_1 = \{(x_1, y_1), (x_2, y_2), (x_1, y_1), (x_2, y_2)\}$$

$$D_2 = \{(x_1, y_1), (x_1, y_1), (x_1, y_1), (x_1, y_1)\}$$

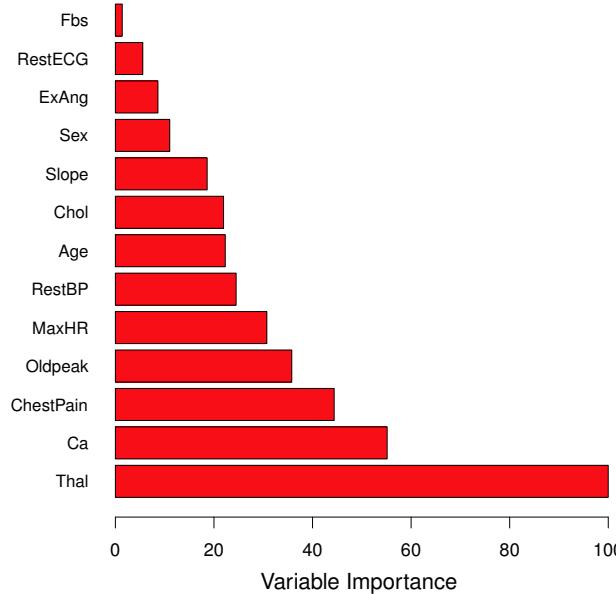
$$D_3 = \{(x_2, y_2), (x_3, y_3), (x_3, y_3), (x_3, y_3)\}$$

Solution:

$$\frac{1}{4} \left((h_3(x_1) - y_1)^2 + (h_2(x_2) - y_2)^2 + \left(\frac{h_1(x_3) + h_2(x_3)}{2} - y_3 \right)^2 + \left(\frac{h_1(x_4) + h_2(x_4) + h_3(x_4)}{3} - y_4 \right)^2 \right)$$

Variable Importance

- Disadvantage of bagged trees and random forests over simple decision trees: harder to interpret.
- Bagging improves prediction accuracy at cost of interpretability.
- However: Bagged trees and random forests allow for simple way to obtain summary of feature importance:
For each feature, look at every base tree and every split the feature is involved in, and take the average reduction in loss. Features associated with large loss reductions are deemed important.
- Example:



- Interesting to look at most important features and retrain models only for those, hopefully still doing well (i.e., can use random forests for *feature selection*)

sklearn built-in Datasets

- `sklearn` comes with a few small datasets built in. To access them, first add

```
from sklearn import datasets
```

at the beginning.

- Then load a particular one (here: "iris") into your environment using

```
iris = datasets.load_iris()
```

followed by

```
features = pd.DataFrame(data=iris.data)
```

- To name the columns properly, use

```
features = pd.DataFrame(data=iris.data, columns=iris.feature_names)
```

- The targets can be obtained using

```
targets = iris.target
```

Note: It can be useful to convert the features dataframe into an array before fitting a model on it.

Python: (More) Random Forests and k-fold CV

- The number of base trees B is given by parameter `n_estimators`.
- The number $m \leq p$ of features subsampled before each split is given by `max_features`. (Look into sklearn documentation for possible values.)
- If you want to use OOB, you need to set parameter `oob_score` to True.
- The variable importance can be obtained using attribute `feature_importances_`

We will use k-fold CV instead of the validation set method. Code stub:

```
from sklearn.model_selection import KFold

k = 10
kf = KFold(n_splits=k, shuffle=True)

for train_indices, validate_indices in kf.split(X):
    X_train, X_validate = X.iloc[train_indices,:], X.iloc[validate_indices,:]
    y_train, y_validate = y[train_indices], y[validate_indices]
    ...

```

Python: Random Forests

1. Load `sklearn's breast_cancer` dataset into your environment.
2. Use 5-fold cross validation on this dataset.
3. Create and fit a decision tree model and a random forest (RF) model on the training data.
4. ...

Credits

Some of the figures in this presentation are taken from the book "An Introduction to Statistical Learning" with permission from the authors.

Prof. Dr. Jens Witkowski

Assistant Professor of Computer Science and Management

Frankfurt School of Finance & Management
Adickesallee 32-34
60322 Frankfurt am Main
Germany

Email: j.witkowski@fs.de
Tel.: +49 (0) 69-154008-875
Fax: +49 (0) 69-154008-4875

Class 07

Ensemble Methods II



Computational Statistics

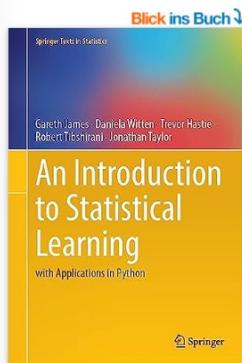
Prof. Dr. Jens Witkowski

September 18, 2023

Agenda

- Session 1
 - Random Forests recap
 - Python Exercise
- Session 2
 - Python Exercise (cont.)
 - Class Imbalance

Literature Pointer



An Introduction to Statistical Learning: with Applications in Python (Springer Texts in Statistics)

Gebundene Ausgabe – 1. Juli 2023

Englisch Ausgabe | von [Gareth James](#) (Autor), [Daniela Witten](#) (Autor), [Trevor Hastie](#) (Autor), [Robert Tibshirani](#) (Autor) & 1 mehr

5,0 ★★★★★ 1 Sternbewertung [Alle Formate und Editionen anzeigen](#)

Gebundenes Buch
106,99 € prime

Taschenbuch
74,89 € prime

8 Neu ab 103,30 €

1 Neu ab 74,89 €

An Introduction to Statistical Learning provides an accessible overview of the field of statistical learning, an essential toolset for making sense of the vast and complex data sets that have emerged in fields ranging from biology to finance, marketing, and astrophysics in the past twenty years. This book presents some of the most important modeling and prediction techniques, along with relevant applications. Topics include linear regression, classification, resampling methods, shrinkage approaches, tree-based methods, support vector machines, clustering, deep learning, survival analysis, multiple testing, and more. Color graphics and real-world examples are used to illustrate the methods presented. This book is targeted at statisticians and non-statisticians alike, who wish to use cutting-edge statistical learning techniques to analyze their data.

Four of the authors co-wrote *An Introduction to Statistical Learning, With Applications in R* (ISLR), which has become a mainstay of undergraduate and graduate classrooms worldwide, as well as an important reference book for data scientists. One of the keys to its success was that each chapter contains a tutorial on implementing the analyses and methods presented in the R scientific computing environment. However, in recent years Python has become a popular language for data science, and there has been increasing demand for a Python-based alternative to ISLR. Hence, this book (ISLP) covers the same materials as ISLR but with labs implemented in Python. These labs will be useful both for Python novices, as well as experienced users.

[Weniger lesen](#)

[Falsche Produktinformationen melden.](#)



Dieses Bild anzeigen

Den Autoren folgen



Trevor
Hastie

Folgen



Gareth
James

Folgen

- Recommended for Class 01 (Sep 4): Section 2.1
- Recommended for Class 02 (Sep 7): Section 3.1, 3.2, 7.1
- Recommended for Class 03 (Sep 7): Sections 5.1, 9.1-9.4
- Recommended for Class 04 (Sep 11): Sections 9.6, 8.1
- Recommended for Class 05 (Sep 12): Section 8.1
- Recommended for Class 06 (Sep 14): Section 8.2
- Recommended for Class 07 (first half today): Section 8.2

Bagging

- "Bagging" is short for "**bootstrap aggregating**" and is an example of an ensemble method. Ensemble methods combine several models into one "super" model, often achieving higher accuracy.
- General-purpose procedure : can be applied to a variety of different types of base models to reduce "variance." Roughly speaking: a model class suffers from high variance if splitting data in half and training two models – one for each half – tends to lead to very different predictions.
- Often used in conjunction with decision trees because individual trees tend to suffer from high variance and overfitting.
- Bagging procedure (B is a parameter):
 1. Randomly sample B new datasets from training dataset (of same size as training dataset and with replacement).
 2. Train B models, one for each of these new datasets.
 3. Use "average" of all B base model predictions as prediction.
(literal averaging for regression and majority vote for classification)
- Large B does not lead to overfitting. Choose B sufficiently large for test error to have settled down.

Random Forests

- Improvement over bagged trees through decorrelation of base trees.
- Procedure:
 1. As in bagged trees, randomly sample with replacement to create new datasets.
 2. When growing the base trees, however, any split can only pick from a random sample of m features from the full set of p features. A fresh sample is taken each split and we typically choose $m \approx \frac{p}{3}$ for regression and $m \approx \sqrt{p}$ for classification.
- Intuition:
 - Suppose there exists a strong feature in dataset along with moderately strong features.
 - Most or all the base trees in (regular) bagging will have that feature at root.
 - Hence, base trees will look very similar to each other and predictions will be highly correlated.
 - Problem with that: averaging correlated quantities does not lead to substantial variance reduction.
 - Random forests overcome this problem by ignoring the strong feature in most splits and other features will take its spot, leading to de-correlation of base trees.

Out-Of-Bag (OOB) Error

Clever trick to get test error "for free" in bagging and random forests (no need to do cross validation):

- Remember: sampling with replacement.
- On average: each base tree uses only around 2/3 of data points; 1/3 of data points not used in training.
- Idea: use those data points not used in training to evaluate the ensemble "out of sample":
 - For each data point (x_i, y_i) , evaluate loss by averaging predictions from those trees corresponding to new datasets in which (x_i, y_i) was not sampled.
- Can be shown: For large B , OOB error essentially equivalent to LOOCV error!

Exercise: Let h_b be the regressor tree trained on dataset b and let $h_b(x_i)$ be its prediction for point (x_i, y_i) .

Your $B = 3$ bootstrapped datasets contain the following points. Give the calculation for the OOB MSE.

$$D_1 = \{(x_1, y_1), (x_2, y_2), (x_1, y_1), (x_2, y_2)\}$$

$$D_2 = \{(x_1, y_1), (x_1, y_1), (x_1, y_1), (x_1, y_1)\}$$

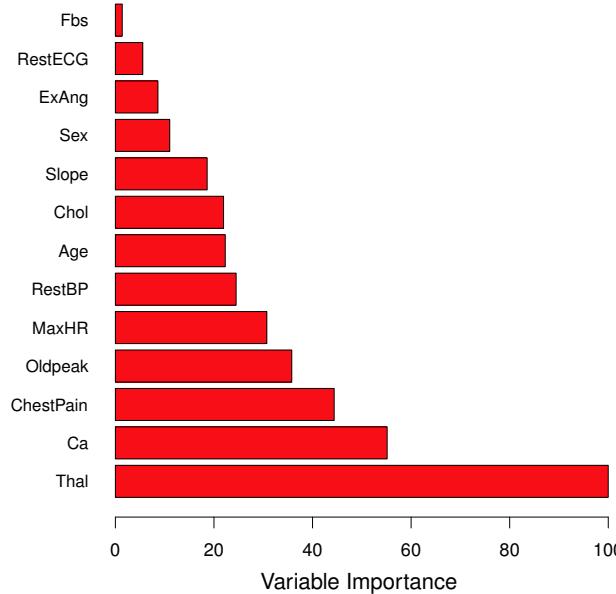
$$D_3 = \{(x_2, y_2), (x_3, y_3), (x_3, y_3), (x_3, y_3)\}$$

Solution:

$$\frac{1}{4} \left((h_3(x_1) - y_1)^2 + (h_2(x_2) - y_2)^2 + \left(\frac{h_1(x_3) + h_2(x_3)}{2} - y_3 \right)^2 + \left(\frac{h_1(x_4) + h_2(x_4) + h_3(x_4)}{3} - y_4 \right)^2 \right)$$

Variable Importance

- Disadvantage of bagged trees and random forests over simple decision trees: harder to interpret.
- Bagging improves prediction accuracy at cost of interpretability.
- However: Bagged trees and random forests allow for simple way to obtain summary of feature importance:
For each feature, look at every base tree and every split the feature is involved in, and take the average reduction in loss. Features associated with large loss reductions are deemed important.
- Example:



- Interesting to look at most important features and retrain models only for those, hopefully still doing well (i.e., can use random forests for *feature selection*)

Python: Random Forests

1. Load `sklearn's breast_cancer` dataset into your environment.
2. Use 5-fold cross validation on this dataset.
3. Create and fit a decision tree model and a random forest (RF) model on the training data.
4. Create a function `accuracy(model, x, y)` that returns the fraction of correctly classified points of `model` for `x` and `y`.
5. Print the average accuracy of the decision tree and the RF over all k=5 folds.
6. Also include a bagged-tree model and output the average error of that model over all folds.
7. Create a new random forest object (`RF_classifier2`), fit it on the entire dataset, and print its OOB error.
8. Define a variable `best_features` that contains `RF_classifier2`'s 2 most important features.
9. Fit an RF using only those 2 features and output its OOB error. Change the number of most important features.
What would you have expected?
10. Bonus: Use the plotting routine from Class 05 to visualize the data using only the 2 most important features together with the RF's predictions.

Pros and Cons of Random Forests

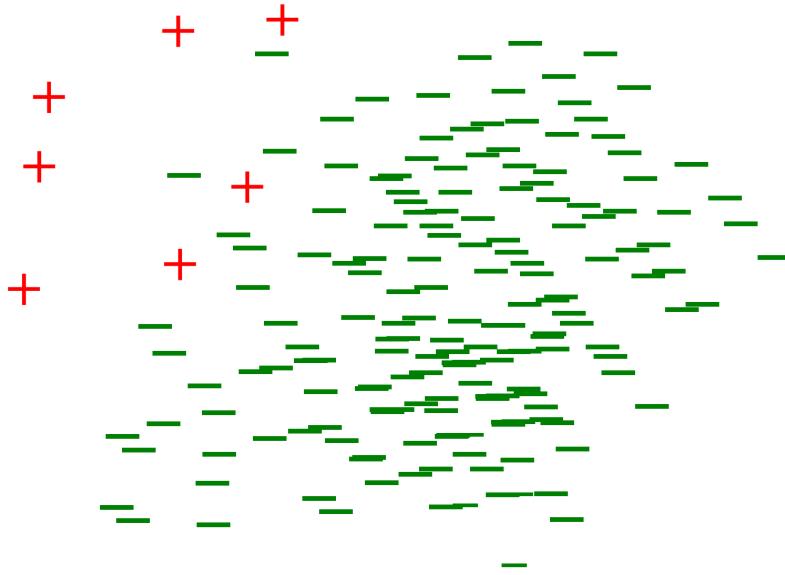
Advantages:

- Very accurate in many real-world scenarios!
- Practically no parameter tuning: no choosing depth of tree or degree of polynomials -- works out of the box!
- Fairly fast and easily parallelizable (even on single computer with several cores).
- OOB error: automatically gives unbiased estimate of the generalization error.
- Can estimate variable importance.
- Handles large numbers of input variables.
- No feature scaling required.

Disadvantages:

- Harder to interpret than simple decision trees.
- Do not handle large numbers of irrelevant features very well.
- Prediction accuracy very high but typically not single best (e.g., well-tuned boosted trees are typically slightly more accurate but they require tuning of parameters).

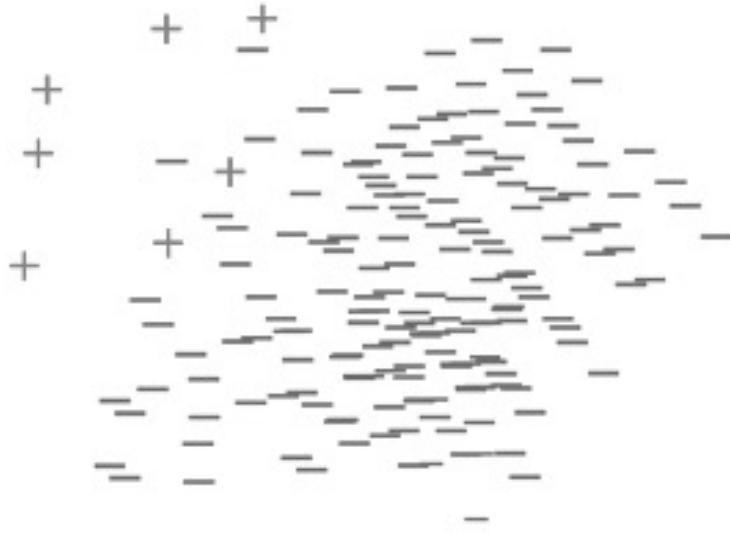
Class Imbalance



How impressive is a classifier with 95% accuracy here?

Accuracy (fraction of correct classifications) might not be the appropriate measure for highly imbalanced data.

Alternative Metrics for Imbalanced Data



Example of confusion matrix for given dataset and classifier:

		True label	
		Positive	Negative
Predicted label	Positive	TP 5	FP 11
	Negative	FN 2	TN 97

[adapted from Krause, 2016]

Accuracy?

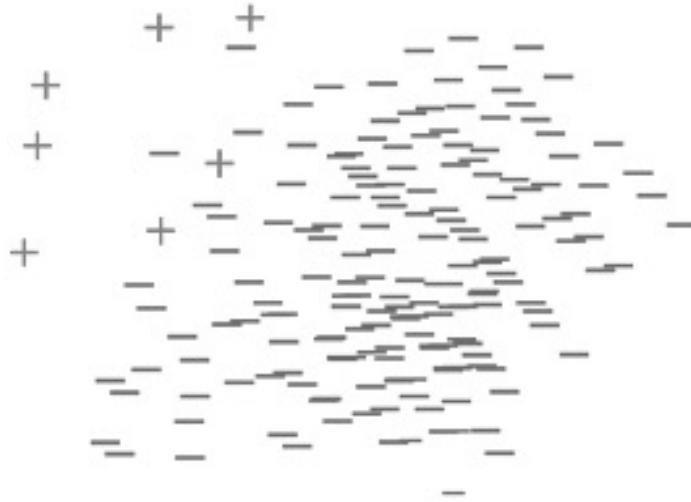
Recall (AKA "Sensitivity"): "Fraction of positive points that are correctly classified."

Specificity: "Fraction of negative points that are correctly classified."

Precision: "Fraction of points predicted positive that really are positive."

What is a classifier with very high recall? Very high specificity? Very high precision?

Alternative Metrics for Imbalanced Data



Predicted label

Confusion Matrix

True label

	Positive	Negative
Positive	TP	FP
Negative	FN	TN

Accuracy: $(TP + TN) / (TP + TN + FP + FN)$

[Krause, 2016]

Recall (AKA "Sensitivity"): $TP / (TP + FN)$ "Fraction of positive points that are correctly classified."

Specificity: $TN / (TN + FP)$ "Fraction of negative points that are correctly classified."

Precision: $TP / (TP + FP)$ "Fraction of points predicted positive that really are positive."

F1 score: $2TP / (2TP + FP + FN)$ (harmonic mean of precision and recall)

Many others: FPR, ROC AUC, MCC, ...

Related concept: cost-sensitive classification (beyond the scope of this lecture)

Credits

Some of the figures in this presentation are taken from the book "An Introduction to Statistical Learning" with permission from the authors.

Prof. Dr. Jens Witkowski
Assistant Professor of Computer Science and Management

Frankfurt School of Finance & Management
Adickesallee 32-34
60322 Frankfurt am Main
Germany

Email: j.witkowski@fs.de
Tel.: +49 (0) 69-154008-875
Fax: +49 (0) 69-154008-4875

Class 08

Probabilistic Predictions



Computational Statistics

Prof. Dr. Jens Witkowski

September 19, 2023

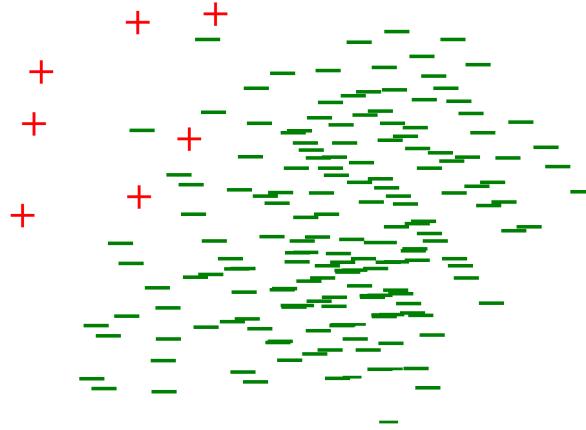
Agenda

- Session 1
 - Probabilistic Predictions
- Session 2
 - Forecasting Game Analysis

Probabilistic Predictions

(Some) classifiers can also predict probabilities.

How to judge whether a given probabilistic classifier is good? No right/wrong anymore...



Always reporting 0% for "+" class? Always reporting 5%?

Classifier always predicting 5% would be well calibrated (if 5% is the relative frequency of class in dataset).

But that classifier does not discriminate between positive and negative points. Want predicted probabilities for class "+" to be higher for data points with true label "+" than for those with true label "-" (higher "refinement").

But what about a classifier that predicts 80% for "+" class for those data points where true label is "+" and 60% for "+" class where the true label is "-"?

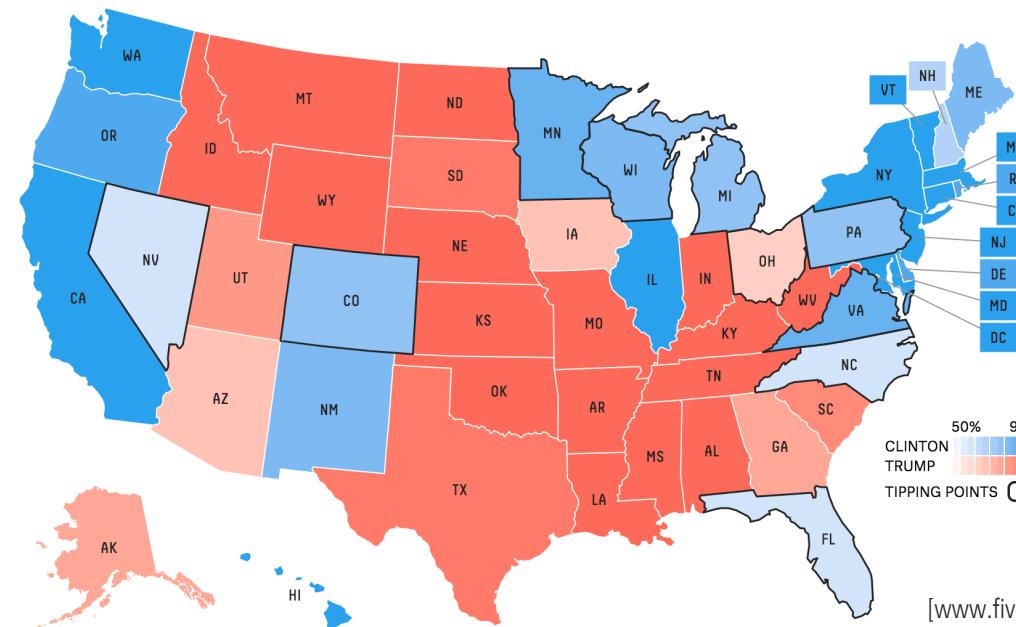
Proper scoring rules / proper losses incorporate both calibration and refinement.

Accuracy of Forecasts

Who will win the presidency?



Chance of winning



[www.fivethirtyeight.com]

Was this a good forecast? A bad forecast? How could we measure this?

Main Questions

1. How can we incentivize a forecaster to report her belief truthfully?
2. How should we determine how accurate a forecast or forecaster is?
3. How can we incentivize forecasters to invest costly effort into forming accurate beliefs in the first place?
4. How should we aggregate forecasts from several forecasters?

Tight connection between crowd forecasting and supervised machine learning.

Incentives for Truth Revelation

Probabilistic Predictions



“What is the probability that transaction X is fraudulent?”

Probabilistic Predictions



"What do you think is the probability that Donald Trump will be US president-elect on January 1, 2025?"

Proper Scoring Rules

Assuming we can pay a probabilistic forecaster: How can we incentivize her to be truthful?

Timeline:

1. Assume forecaster has true belief $p \in [0, 1]$. (We are agnostic as to how she formed it.)
2. Forecaster reports forecast $y \in [0, 1]$.
3. Event materializes to $x = 1$ ("event occurred") or $x = 0$ ("event did not occur").
4. Pay forecaster $R(y, x)$.

Objective:

Scoring rule $R(y, x)$ is *truthful* (or: *proper*) if reporting $y = p$ maximizes expected score, i.e. if

$$p = \arg \max_y \mathbf{E}[R(y, X)],$$

where expectation about X is taken with respect to true belief p .

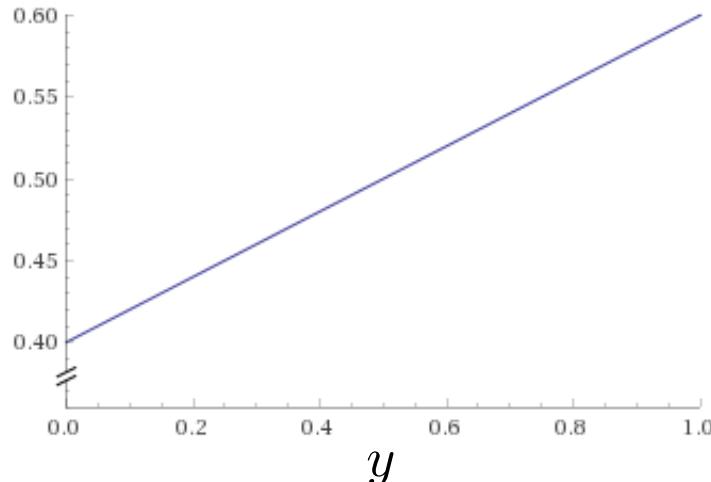
Scoring rule is *strictly* truthful (or: *strictly* proper) if $y = p$ is *only* report maximizing expected score.

First try: Linear Scoring Rule

$$R_{\text{lin}}(y, x) = \begin{cases} y, & \text{if } x = 1 \\ 1 - y, & \text{if } x = 0 \end{cases}$$

Example:

- True belief $p = 0.6$. What is expected score?
- Expected score: $0.6 \cdot y + 0.4 \cdot (1 - y) = 0.4 + 0.2y$



- $y = 1 \neq p = 0.6$ maximizes expected score!

Linear Scoring Rule is not proper!

Quadratic Scoring Rule ("Brier Score")

$$R_q(y, x) = 1 - (y - x)^2$$

Exercise:

- Calculate expected score given true belief p (similar to last slide)

$$\begin{aligned}
 \mathbf{E}[R_q(y, X)] &= p \cdot R_q(y, 1) + (1 - p) \cdot R_q(y, 0) \\
 &= p \cdot (1 - (y - 1)^2) + (1 - p) \cdot (1 - (y - 0)^2) \\
 &= p \cdot (1 - (y^2 + 1 - 2y)) + (1 - p) \cdot (1 - y^2) \\
 &= p \cdot (1 - y^2 - 1 + 2y) + (1 - y^2 - p + py^2) \\
 &= -py^2 + 2py + 1 - y^2 - p + py^2 \\
 &= 2py + 1 - y^2 - p
 \end{aligned}$$

Quadratic Scoring Rule (cont.)

Expected score (from previous slide): $\mathbf{E}[R_q(y, X)] = 2py + 1 - y^2 - p$

2. Take derivative w.r.t. y and set to 0.

$$2p - 2y := 0 \Rightarrow y = p$$

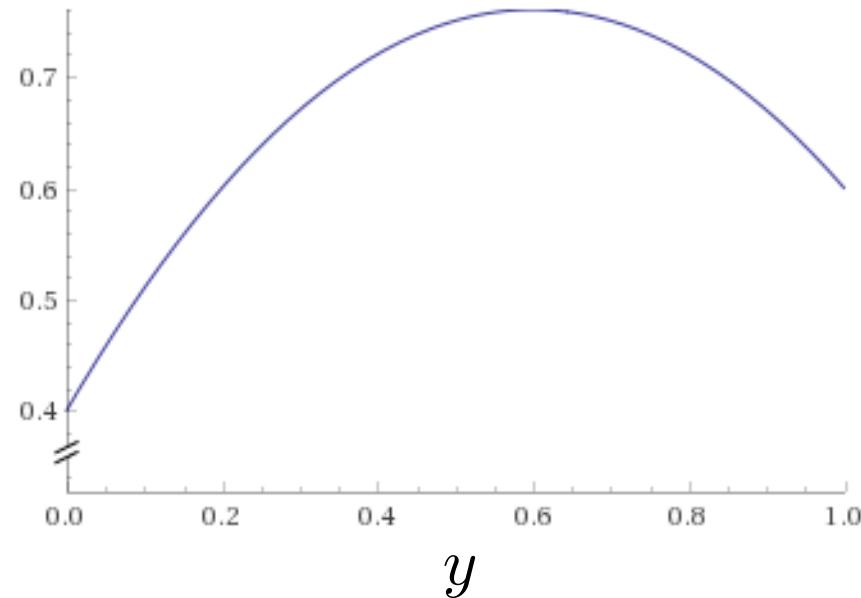
3. Minimum or maximum?

Maximum since second derivative of expected score is -2 .

Quadratic scoring rule is *proper*, i.e. truthfully elicits probabilistic beliefs about future events!

Quadratic Scoring Rule: Example

Expected payment of quadratic scoring rule if $p=0.6$:



Also Proper: Logarithmic Scoring Rule

$$R_l(y, x) = \begin{cases} \ln(y), & \text{if } x = 1 \\ \ln(1 - y), & \text{if } x = 0 \end{cases}$$

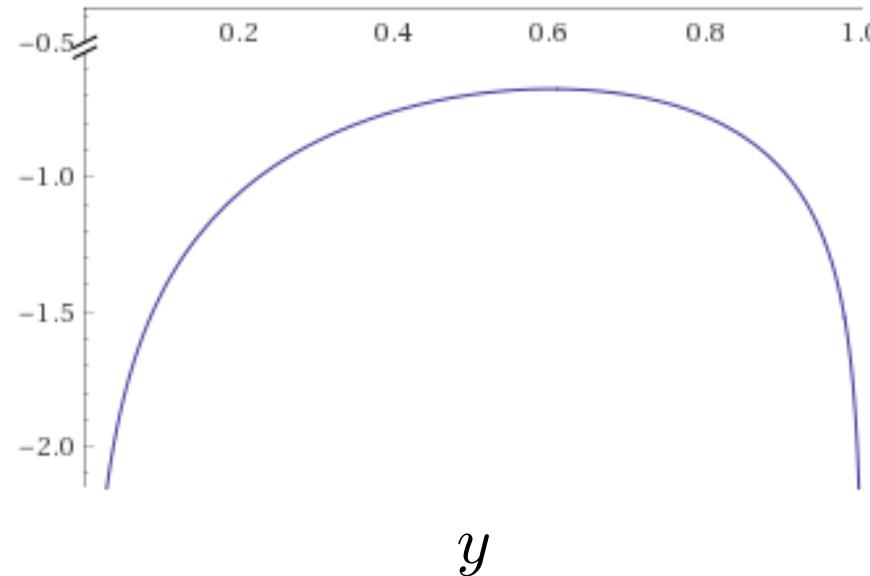
1. Expected score: $p \cdot \ln(y) + (1 - p) \cdot \ln(1 - y)$
2. Take derivative w.r.t. y and set to 0:

$$\frac{p}{y} + \frac{1-p}{1-y} \cdot (-1) = \frac{p}{y} + \frac{1-p}{y-1} = \frac{p(y-1)}{y(y-1)} + \frac{y(1-p)}{y(y-1)}$$

$$= \frac{py - p + y - py}{y(y-1)} = \frac{y - p}{y^2 - y} := 0 \Rightarrow y = p$$
3. Can confirm that $y = p$ is unique maximum with second derivative.

Logarithmic Scoring Rule: Example

Expected payment of logarithmic scoring rule if $p=0.6$:



Potential issues with logarithmic rule:

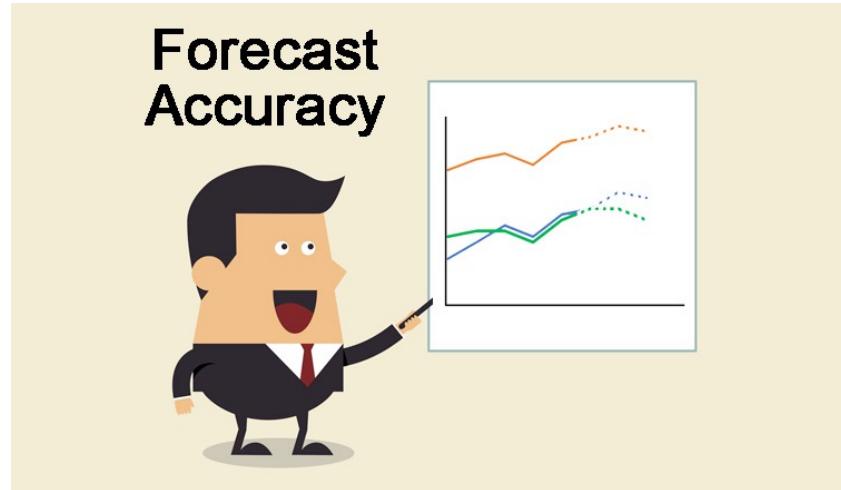
- Always negative (not ideal when scores are payments)
- Score is $-\infty$ if $(y = 0 \text{ and } x = 1) \text{ or } (y = 1 \text{ and } x = 0)$

There exist infinitely many proper scoring rules but will mostly focus on these two!

Judging Forecast Accuracy

Proper Scoring Rules as Accuracy Measures

- Proper scoring rules also used “ex post” to measure accuracy of probabilistic forecasts



[www.callcentrehelper.com]

- Report y and outcome x same as before.
- Instead of subjective belief p , we have objective “true probability” θ (which is latent/unknown)
- In this context, proper means that if θ is true probability, reporting $y = \theta$ obtains highest expected score (same math as before but different perspective)

Proper Scoring Rules: Accuracy

Timeline:

1. Forecaster reports forecast $y \in [0, 1]$.
2. Event materializes to $x = 1$ ("event occurred") or $x = 0$ ("event did not occur").
3. Give accuracy score $R(y, x)$ to forecaster.

Objective:

Scoring rule $R(y, x)$ is *proper* if reporting $y = \theta$ maximizes expected score, i.e.

$$\theta = \arg \max_y \mathbf{E}_{X \sim \theta} [R(y, X)]$$

Comparing Imperfect Forecast(er)s

Probability of heads (H) for biased  ?

Assume: $\theta = 0.7$

Two forecasters with reports:



$$y_M = 0.79$$



$$y_L = 0.6$$

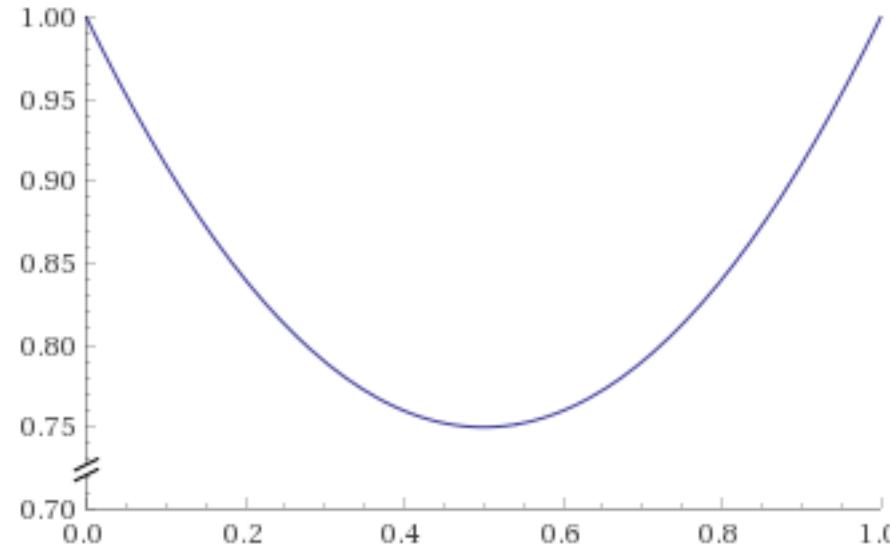
Which forecaster should be considered better?

Expected Loss of Quadratic Scoring Rule

With proper scoring rule, expected score is maximized when reporting true probability θ

For quadratic scoring rule, expected score is $2\theta y + 1 - y^2 - \theta$.

Thus, reporting $y = \theta$ yields expected score $2\theta^2 + 1 - \theta^2 - \theta = \theta^2 - \theta + 1$:



The expected score loss of reporting y instead of θ is:

$$\begin{aligned}
 & \theta^2 - \theta + 1 - (2\theta y + 1 - y^2 - \theta) \\
 &= \theta^2 - \theta + 1 - 2\theta y - 1 + y^2 + \theta \\
 &= (\theta - y)^2
 \end{aligned}$$

Comparing Imperfect Forecasts (cont.)

Probability of heads for biased  ?

Assume: $\theta = 0.7$



$$y_M = 0.79$$



$$y_L = 0.6$$

Which forecaster should be considered better?

Quadratic rule: Mario has higher expected score since $(0.79 - 0.7)^2 < (0.6 - 0.7)^2$.

Log rule: Luigi better since $0.7 \cdot \ln(0.6) + 0.3 \cdot \ln(0.4) = -0.632 > 0.7 \cdot \ln(0.79) + 0.3 \cdot \ln(0.21) = -0.633$.

Rankings not necessarily unambiguous: different proper scoring rules rank imperfect forecast(er)s differently!

Forecasting Game Outcomes I

Will Novak Djoković win the US Open 2023?

U.S. Open Men's Final

How Novak Djokovic Beat Daniil Medvedev

Momentum seemed to be against him at times, as he fought either fatigue or injury, but in the end, Djokovic handled Medvedev in three sets.

Sept. 10, 2023

 Share full article  



Michelle V. Agins/The New York Times

Men's Singles Final		
	Final	
3 Daniil Medvedev	3	6 ⁵ 3
2 Novak Djokovic ✓	6	7 ⁷ 6

Outcome: yes!

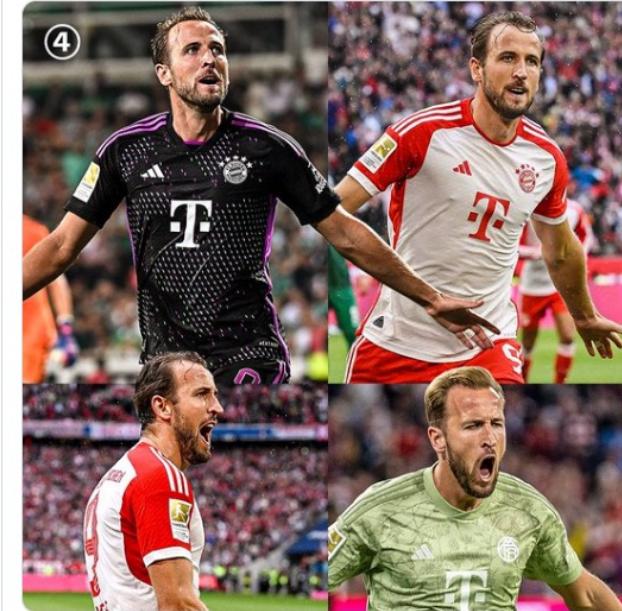
Will Harry Kane shoot at least one goal for Bayern Munich in their game against Bayer Leverkusen on September 15, 2023?

WHAT A START TO LIFE IN GERMANY FOR HARRY KANE

Was it ever in doubt?


Bayern & Football 
 @MunichFanpage · Follow

Harry Kane, 4 games, 4 goals! 🔥

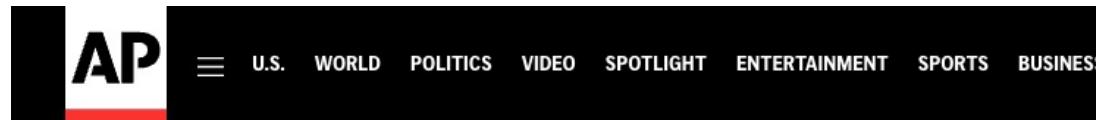


9:19 PM · Sep 15, 2023

Outcome: yes!

Forecasting Game Outcomes II

Will the Dow Jones Industrial index be over 35000 points on closing of September 14, 2023?



On Thursday:

OTHER NEWS



**Stock market today:
Wall Street closes
lower, giving S&P
500 another losing
week**



**Arm Holdings shares
gain nearly 25% in
biggest initial public
offering since late
2021**



**Stock market today:
Asian shares gain,
led by China after
Beijing eases
required bank
reserves**

The S&P 500 rose 37.66 points, or 0.8%, to 4,505.10.

The Dow Jones Industrial Average rose 331.58 points, or 1%, to 34,907.11.

Outcome: no!

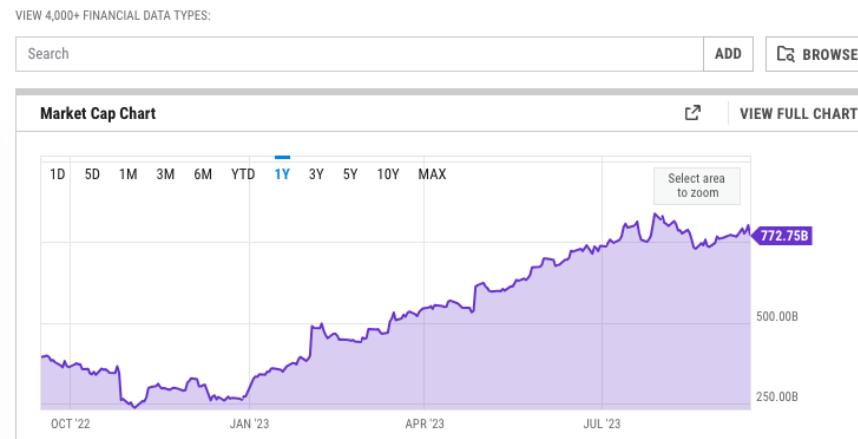
Forecasting Game Outcomes III

Will Tesla's market capitalization be higher than Meta's at closing on Sep. 14 (according to ycharts.com)?

Tesla Market Cap: 870.91B for Sept. 15, 2023



Meta Platforms Market Cap: 772.75B for Sept. 15, 2023



Historical Market Cap Data

View and export this data back to 2010. [Upgrade now.](#)

Date	Value	Date	Value
September 15, 2023	870.91B	August 10, 2023	778.71B
September 14, 2023	876.15B	August 09, 2023	768.71B

Historical Market Cap Data

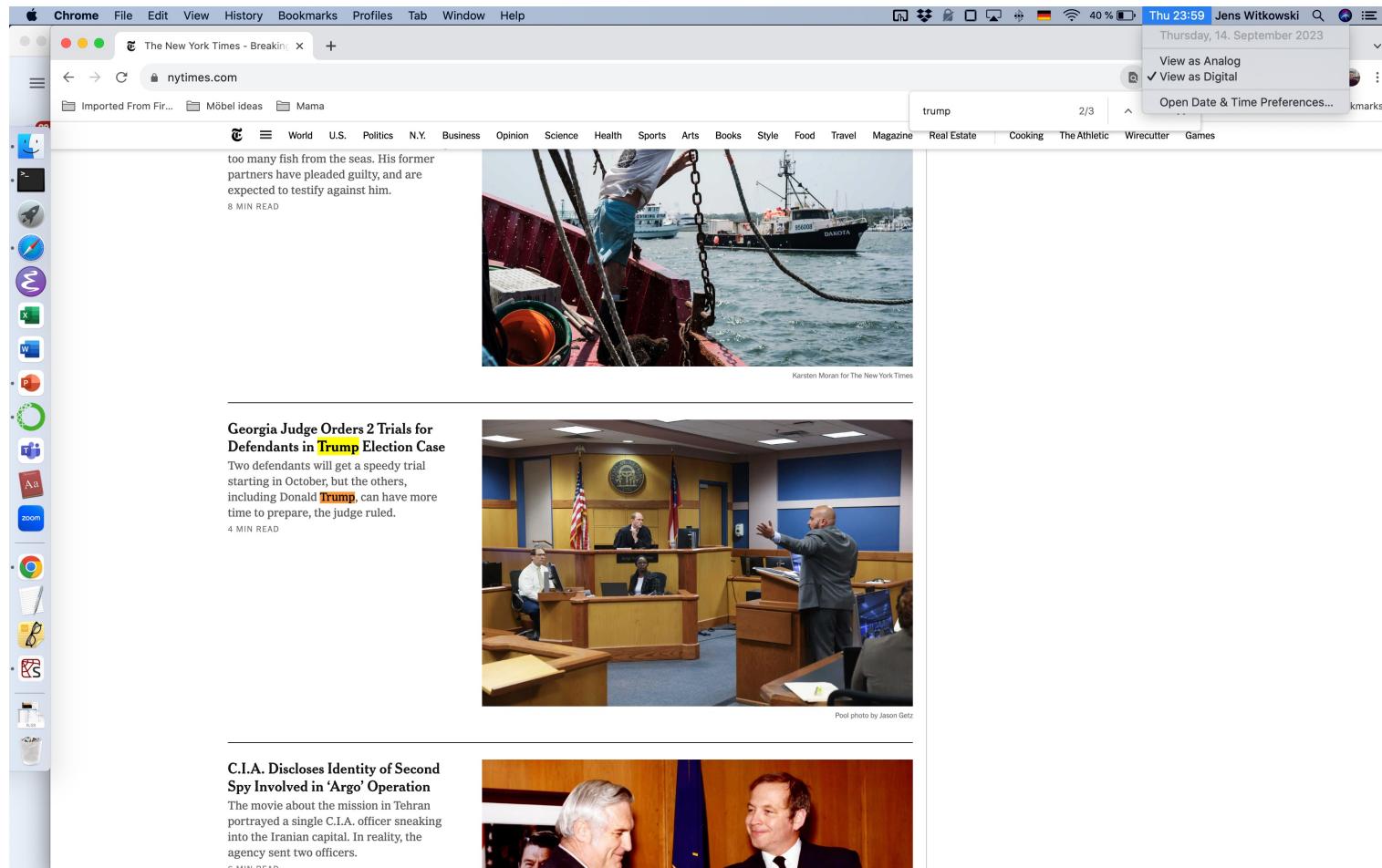
View and export this data back to 2012. [Upgrade now.](#)

Date	Value	Date	Value
September 15, 2023	772.75B	August 10, 2023	786.72B
September 14, 2023	802.11B	August 09, 2023	785.35B

Outcome: yes!

Forecasting Game Outcomes IV

How often will the word "Trump" appear on the main site of www.nytimes.com at 11.59p on Sep. 14?



The New York Times - Breaking x +

nytimes.com

Imported From Fir... Möbel ideas Mama

World U.S. Politics N.Y. Business Opinion Science Health Sports Arts Books Style Food Travel Magazine Real Estate Cooking The Athletic Wirecutter Games

trump 2/3

Thursday, 14. September 2023

View as Analog

View as Digital

Open Date & Time Preferences...

too many fish from the seas. His former partners have pleaded guilty, and are expected to testify against him.
8 MIN READ



Karsten Moran for The New York Times

Georgia Judge Orders 2 Trials for Defendants in Trump Election Case
Two defendants will get a speedy trial starting in October, but the others, including Donald Trump, can have more time to prepare, the judge ruled.
4 MIN READ



Pool photo by Jason Getz

C.I.A. Discloses Identity of Second Spy Involved in 'Argo' Operation
The movie about the mission in Tehran portrayed a single C.I.A. officer sneaking into the Iranian capital. In reality, the agency sent two officers.
6 MIN READ



Outcome: "3 or more times"!

Forecasting Game Outcomes V

Will there be any Covid-related masking requirements at Frankfurt School by September 14, 2023?

Outcome: no!

Forecasting Game Outcomes VI

Will the Cleveland Browns win against the Cincinnati Bengals in their game on September 10, 2023?



'We just lost to some elves': Browns win 100th 'Battle of Ohio' and rest of Sunday's Week 1 NFL games

By Ben Morse, CNN

Updated 6:30 AM EDT, Mon September 11, 2023



Outcome: yes!

Forecasting Game Outcomes VII

Will Germany win against Japan in the soccer test match on September 9, 2023?

Six observations from Germany's embarrassing 4-1 loss to Japan

Below not good.

By [CSmith1919](#) | [@TheBarrelBlog](#) | Updated Sep 10, 2023, 4:56pm CEST | 94 Comments / 94 New

[!\[\]\(e254acbc1557bff3559ee2336624e6ca_img.jpg\)](#) [!\[\]\(00eee6626692d02a6ee28b9cf6065f82_img.jpg\)](#) [!\[\]\(5c3e599568580d48225122743dc32aba_img.jpg\)](#) SHARE



Outcome: ouch!

Python Practice

1. Download `fcasts_raw.csv` from Canvas and load it as `fcasts_raw` into your environment using "Voter" as your index.
2. Only keep those columns in `fcasts_raw` that are about binary questions.
3. Take a look at the data frame. Is any kind of data cleaning needed?
4. Only consider "yes" probabilities for `fcasts_raw`. (For the Trump question, interpret "3 or more" as "yes.") Further, transform the values in `fcasts_raw` currently in between 0 to 100 into probability values between 0 and 1. Convert this dataframe into a 2D-Numpy array called `fcasts`.
5. Create an outcome vector `outcome` with all the `x` values for the binary questions, i.e., 1 for outcomes that occurred and 0 else.
6. Create a function `qsr(y, x)` that computes the quadratic score for a single forecast and binary question.
7. Create a function `qsr_forecaster(i)` that computes a forecaster's average quadratic score over all questions.
8. Who was the most accurate by average quadratic score? Print the ranking with forecaster indices.
9. Create functions `lsr(y, x)` and `lsr_forecaster(i)` corresponding to the logarithmic score. Print the ranking again. Has it changed?
10. Compare the last two forecasters in the ranking according to the logarithmic score. What happened here?

Homework 2

1. You will participate in a machine learning competition on kaggle:
<https://www.kaggle.com/t/28e42293434246b5a7fa3df690b3280b>
2. Please sign up for kaggle with your FS email address. As before, you can work as a team. Make sure you join the same team on kaggle as on Canvas. As usual, a team consists of 3 members. You can make 30 submission per day (more than usual because I want you to try out a lot). The team with the most creative name might get 0.5 points extra (up to the maximal 20 points).
3. After you have signed up for kaggle and familiarized yourself with the setup, go through the following tutorial for this classic dataset and make sure you understand each of the steps:
<https://www.kaggle.com/code/alexisbcook/titanic-tutorial> Briefly explain what `get_dummies()` is doing and why it is used in the tutorial code. Also make a first submission using a random forest with the same features as given in the tutorial code.
4. Try to improve your score from the tutorial code submission. For kaggle, you can play around with whichever model you like but for your Canvas submission, I will only count models we have seen in class (so no neural networks, gradient boosted trees, ...). You should consider cleaning the data, playing around with the model parameters, and applying "feature engineering." You are encouraged to explore some of the approaches and observations discussed in the kaggle forum but you are not allowed to copy code from there. You are also not allowed to use any other data than the data provided in the competition. (I am sure you can find the true labels for this but a) using those is lame in a play competition and b) I'm not grading you on your leaderboard position anyway.)
5. In addition to your kaggle submissions, please submit the code of the model you think is best on Canvas as `hw2.py` with appropriate comments explaining what you are doing, why you are doing what you are doing, and what else you have tried. (Leave those working but less successful tries commented out.)

Homework 2 (cont.)

General notes:

- Only upload one ".py" file called `hw2.py` with your code and your textual answers in comments right after the respective code.
- Elegant and easier-to-follow code and plots are part of this (and following) homeworks and will get more points than naive code and ugly plots. Your code should also have meaningful comments. Typos lead to point reductions.
- All students in a group need to upload the **(identical!) group solution** before the deadline. Clearly state all **full(!) names** in the comments at the beginning of your code. Groups of 2 are only allowed if there would otherwise be a group of 1. Groups of 1 are not allowed. I would prefer it if you could stay in the same groups as in Homework 1.
- Submission deadline is 1.15pm on September 26. For fairness, late submissions will not be accepted.
- **Start early!** Depending on your familiarity with Python and your current understanding of the course material, this could take longer to finish than you expect!
- Reminder: All students in a group need to fully understand the submitted solutions and no collaborations outside of your group are allowed! We will discuss your submitted solutions and you will need to be able to answer questions about them.

Any question?

Credits

Some of the figures in this presentation are taken from the book "An Introduction to Statistical Learning" with permission from the authors.

Prof. Dr. Jens Witkowski

Assistant Professor of Computer Science and Management

Frankfurt School of Finance & Management
Adickesallee 32-34
60322 Frankfurt am Main
Germany

Email: j.witkowski@fs.de
Tel.: +49 (0) 69-154008-875
Fax: +49 (0) 69-154008-4875

Class 09

Probabilistic Predictions II



Computational Statistics

Prof. Dr. Jens Witkowski
September 29, 2022

Agenda

- Session 1
 - Recap
 - Value of Information
- Session 2
 - Aggregation

Recap: Proper Scoring Rules

Proper Scoring Rules: Truth-telling Incentives

Timeline:

1. Assume forecaster has true belief $p \in [0, 1]$. (We are agnostic as to how she formed it.)
2. Forecaster reports forecast $y \in [0, 1]$.
3. Event materializes to $x = 1$ ("event occurred") or $x = 0$ ("event did not occur").
4. Pay forecaster $R(y, x)$.

Objective:

Scoring rule $R(y, x)$ is *truthful* (or: *proper*) if reporting $y = p$ maximizes expected score, i.e.

$$p = \arg \max_y \mathbf{E}_{X \sim p} [R(y, X)]$$

Proper Scoring Rules: Accuracy

Timeline:

1. Forecaster reports forecast $y \in [0, 1]$.
2. Event materializes to $x = 1$ ("event occurred") or $x = 0$ ("event did not occur").
3. Give accuracy score $R(y, x)$ to forecaster.

Objective:

Scoring rule $R(y, x)$ is *proper* if reporting $y = \theta$ maximizes expected score, i.e.

$$\theta = \arg \max_y \mathbf{E}_{X \sim \theta} [R(y, X)]$$

Scoring Rules for Binary Events

- Linear scoring rule: $R_{\text{lin}}(y, x) = \begin{cases} y, & \text{if } x = 1 \\ 1 - y, & \text{if } x = 0 \end{cases}$

Linear rule is not proper!

- Quadratic ("Brier") Scoring Rule: $R_q(y, x) = 1 - (y - x)^2$

Quadratic Scoring Rule is proper!

- Logarithmic Scoring Rule: $R_l(y, x) = \begin{cases} \ln(y), & \text{if } x = 1 \\ \ln(1 - y), & \text{if } x = 0 \end{cases}$

Logarithmic Scoring Rule is proper!

Comparing Accuracy Scores of Imperfect Forecasts

Probability of heads for biased  ?

Assume: $\theta = 0.7$



$$y_M = 0.79$$



$$y_L = 0.6$$

Which forecaster should be considered better?

Quadratic rule: Mario has higher expected score since

$$0.7 \cdot (1 - (0.79 - 1)^2) + 0.3 \cdot (1 - 0.79^2) = 0.7819 > 0.78 = 0.7 \cdot (1 - (0.6 - 1)^2) + 0.3 \cdot (1 - 0.6^2)$$

Log rule: Luigi better since

$$0.7 \cdot \ln(0.6) + 0.3 \cdot \ln(0.4) = -0.632 >= -0.633 = 0.7 \cdot \ln(0.79) + 0.3 \cdot \ln(0.21).$$

Rankings not necessarily unambiguous: different proper scoring rules rank imperfect forecast(er)s differently!

Real-valued Outcomes

With real-valued outcomes, first need to decide what should be elicited, e.g.:

- Mean?
- Median?
- Variance?

Assume we know upper bound $\beta \in \mathbb{R}$ and lower bound $\alpha \in \mathbb{R}$ of the outcome $x \in [\alpha, \beta]$ and report $y \in [\alpha, \beta]$.

- Mean:

$$R_q(y, x) = (\beta - \alpha)^2 - (y - x)^2$$

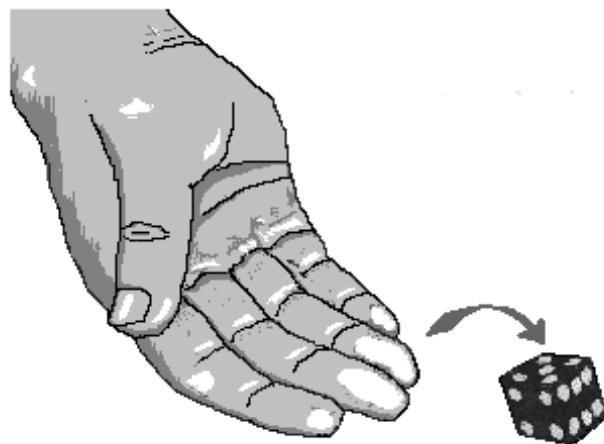
- Median:

$$R_q(y, x) = (\beta - \alpha) - |y - x|$$

- Variance: no proper scoring rule exists! But: can elicit mean and variance together! (beyond scope of lecture)

Value of Information

A Fair Die Example



Probability that "6" comes up?

You are paid according to quadratic scoring rule:

$$R_q(y, x) = 1 - (y - x)^2$$

What would you report? What is expected score of that report?

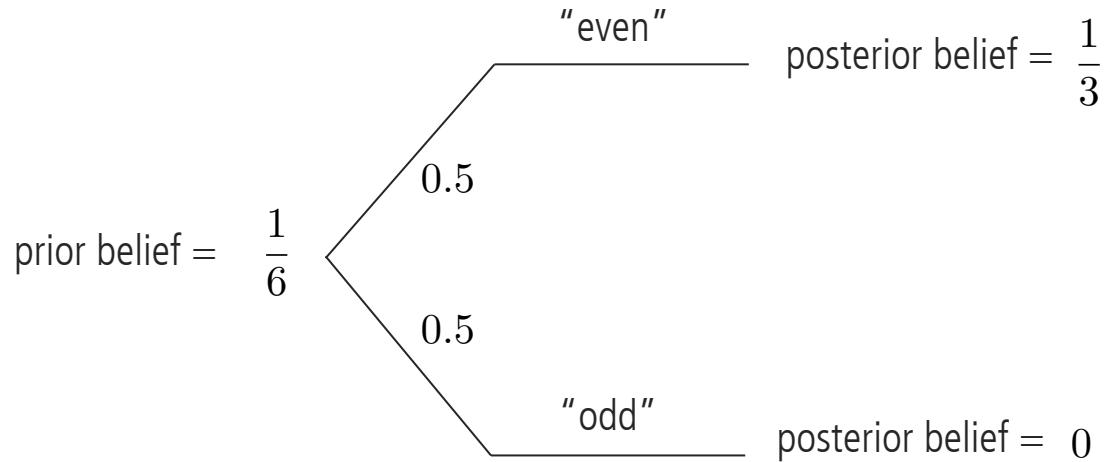
Expected score of truthful report:

$$\frac{1}{6} \cdot R_q\left(\frac{1}{6}, 1\right) + \frac{5}{6} \cdot R_q\left(\frac{1}{6}, 0\right) = \frac{31}{36} = 0.861111$$

Assume someone could tell you whether the number that came up was "odd" (1,3,5) or "even" (2,4,6). How much would that information be worth to you?

A Fair Die Example (cont.)

How is belief updated given new information?



Expected scores given new information?

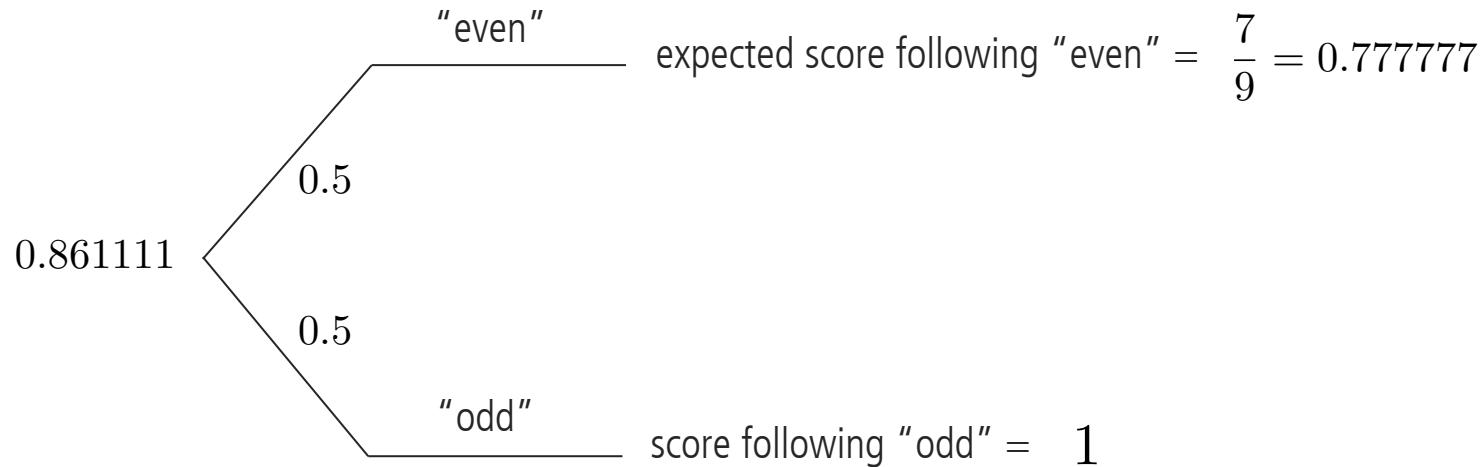
$$\text{"even:"} \quad \frac{1}{3} \cdot R_q\left(\frac{1}{3}, 1\right) + \frac{2}{3} \cdot R_q\left(\frac{1}{3}, 0\right) = \frac{7}{9} = 0.777777$$

$$\text{"odd:"} \quad 1$$

A Fair Die Example (cont.)

Expected score given new information?

Expected score at prior belief =



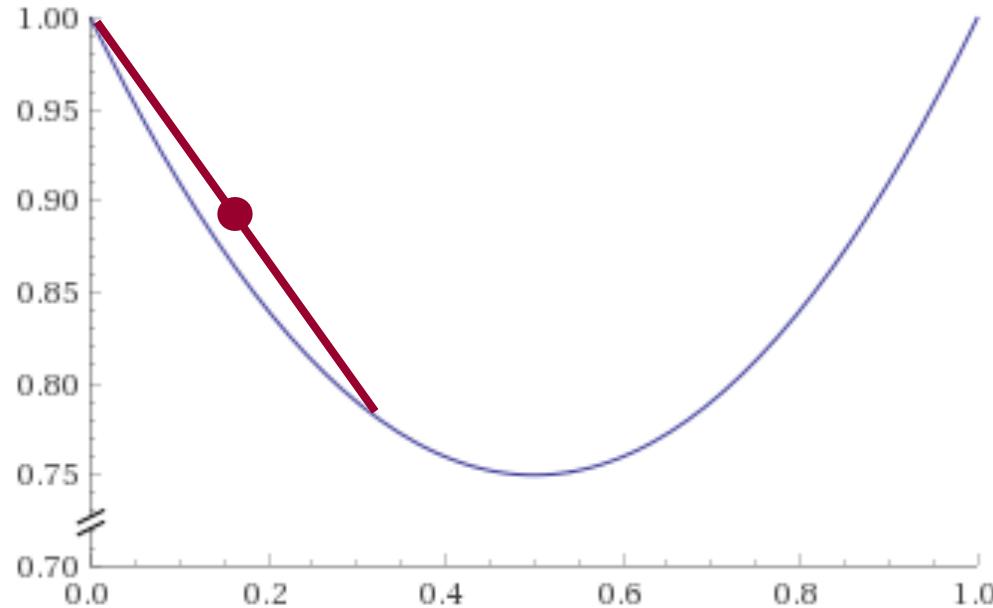
$$\text{Overall expected score after new information: } 0.5 \cdot \frac{7}{9} + 0.5 \cdot 1 = \frac{8}{9}$$

Information is worth $\frac{8}{9} - \frac{31}{36} = \frac{1}{36}$. Should be willing to invest up to that much for it.

Can think of information after learning "odd" or "even" as information after investing costly effort!

Intuition: Why is expected score after new information higher?

Expected score of truthful report at prior (in blue):



Expected score of new information (circle in dark red) always above expected score at prior. The dark red line shows the expected score for different probabilities of the two possible "signals" (here: signals "odd" and "even" equally likely).

Positive-affine Transformations

What if cost of effort higher than expected score difference between prior and posteriors? Can scale scores.

If $R(y, x)$ is a strictly proper scoring rule,

then $a \cdot R(y, x) + b$, where $a > 0$ and $b \in \mathbb{R}$ is also a strictly proper scoring rule

In particular:

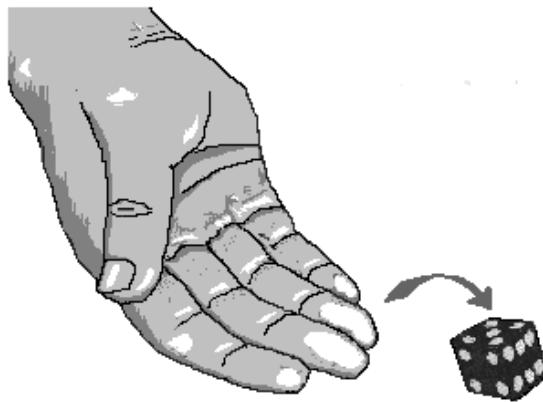
- Quadratic scoring rule (QSR) still proper after applying positive-affine transformation:

$$a \cdot R_q(y, x) + b = a \cdot (1 - (y - x)^2) + b$$

- Logarithmic scoring rule still proper after applying positive-affine transformation:

$$a \cdot R_l(y, x) + b = \begin{cases} a \cdot \ln(y) + b, & \text{if } x = 1 \\ a \cdot \ln(1 - y) + b, & \text{if } x = 0 \end{cases}$$

Die Example with QSR scaled by 2



Probability that "6" comes up?

You are now paid according to scaled quadratic scoring rule:

$$2 \cdot R_q(y, x) = 2(1 - (y - x)^2)$$

Expected score of truthful report:

$$\frac{1}{6} \cdot 2 \cdot R_q\left(\frac{1}{6}, 1\right) + \frac{5}{6} \cdot 2 \cdot R_q\left(\frac{1}{6}, 0\right) = 2 \cdot \frac{31}{36} = \frac{31}{18} = 1.722222$$

How much is learning "odd"/"even" information worth to you now, with scaled scoring rule?

Expected scores given new information?

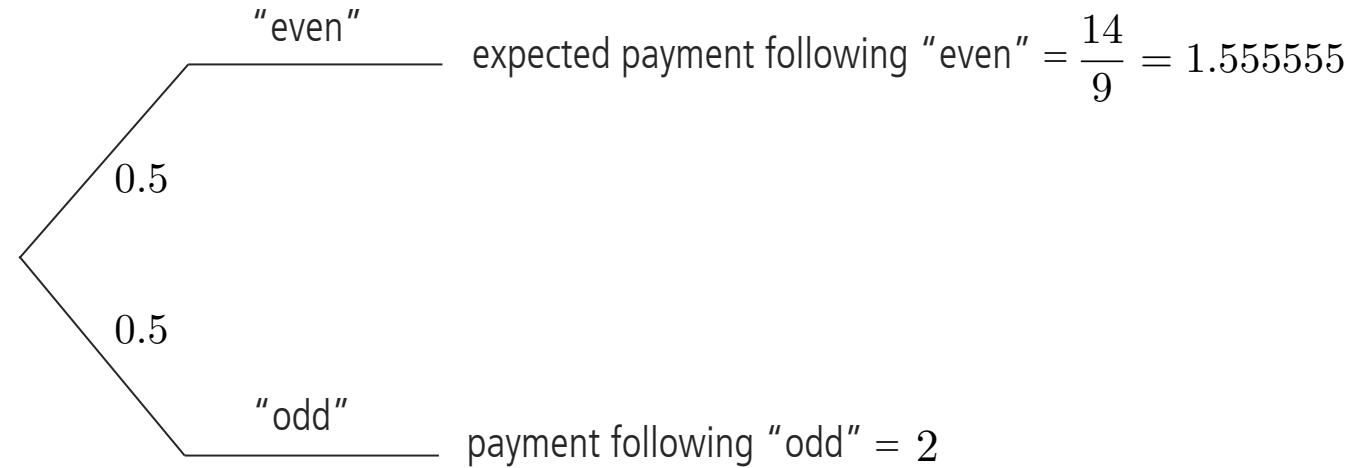
"even:" $\frac{1}{3} \cdot 2 \cdot R_q\left(\frac{1}{3}, 1\right) + \frac{2}{3} \cdot 2 \cdot R_q\left(\frac{1}{3}, 0\right) = 2 \cdot \frac{7}{9} = \frac{14}{9} = 1.555555$

"odd:" 2

Die Example with QSR scaled by 2 (cont.)

Expected payment
at prior belief =

$$1.722222$$



Overall expected score after new information: $0.5 \cdot \frac{14}{9} + 0.5 \cdot 2 = \frac{16}{9}$

Information is now worth $\frac{16}{9} - \frac{31}{18} = \frac{1}{18}$, and forecaster should thus be willing to invest double the effort as before.

Generally, incentivized effort is linear in scaling (e.g., scaling a proper scoring rule by 3 incentivizes 3 times the effort cost of the unscaled proper scoring rule)!

Aggregation

Forecast Aggregation

In crowd forecasting, not enough to incentivize and identify most accurate forecaster.
 Also want to aggregate forecasts from different forecasters into single forecast.



$$y_M = 0.9$$



$$y_L = 0.7$$



$$y_G = 0.2$$

Any ideas for aggregation?

Mean: $\bar{y} = 0.6$

Median: $\bar{y} = 0.7$

- Less susceptible to outliers than mean
- Typically mildly better empirical performance

If some questions already closed, use weighted versions: more weight on forecasters who were accurate on previous questions.

Extremizing

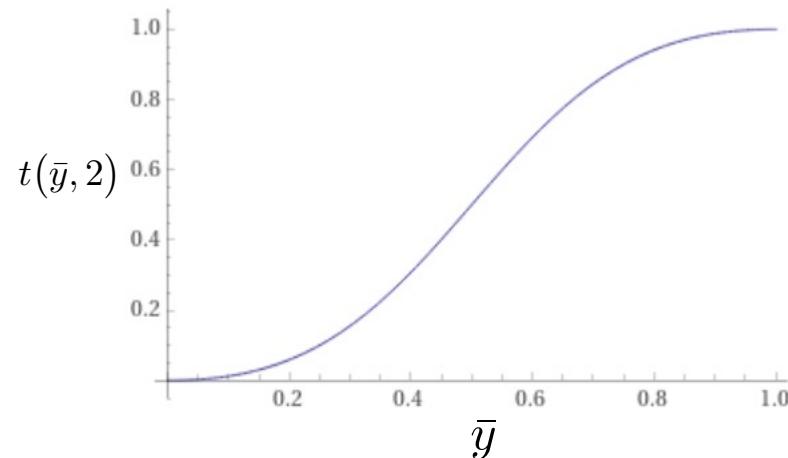
Empirically, the accuracy of both mean and median increases after “extremizing:”

If original value is larger than 0.5, push further towards 1.

If original value is smaller than 0.5, push further towards 0.

Formula:

$$t(\bar{y}, a) = \frac{\bar{y}^a}{\bar{y}^a + (1 - \bar{y})^a}$$



Parameter a controls curvature.

Typical values are around 1.5 to 2.5.

Example:

$$\text{Mean: } \bar{y} = 0.4$$

$$\text{Extremized Mean: } t(0.4, 2) = 0.31$$

$$\text{Median } \bar{y} = 0.7$$

$$\text{Extremized Median: } t(0.7, 2) = 0.84$$

Intuition for Extremizing

Probability of heads (H) for biased ?

Before observing flip: $p(H) = 0.5$

Two forecasters observe a flip and report:



$$p(H|H) = 0.7$$



$$p(H|H) = 0.7$$

Aggregated forecast:

Same coin flip $\Rightarrow p(H|H) = 0.7$

Same coin, different flips $\Rightarrow p(H|H, H) > 0.7$

Less information overlap \Rightarrow more extremizing!

Python Practice

1. Download the code from last class from Canvas and run it once.
2. Create a new vector `mean_fcasts` and use it to save the mean of all forecasts for the 7 questions.
3. How does the mean's accuracy (according to Brier score) compare to the average accuracy of individual forecasters?
4. Create a new vector `median_fcasts` and use it to save the median of all forecast for the 7 questions. How does the median's accuracy compare to individual forecasters?
5. Write a function `extremize(vec, a)` that takes a probability vector `vec` and an extremization parameter `a`, and returns the `a`-extremized probability vector.
6. Score the extremized `mean_fcasts` and the extremized `median_fcasts` with `a=2`, and interpret the results.
7. Write a for loop that prints out the scores of the extremized median with all `a` values between 1 and 10 (by 0.1). What do you see? What is the optimal value of `a`? What is a problem with the approach of simply picking the `a` that maximizes the average score?
8. Now randomly divide the questions into two sets of sizes $l=3$ and $7-l = 4$, resulting in `fcasts_train`, `fcasts_test`, `outcome_train`, `outcome_test` (use `train_test_split`).
9. Write a function `qsr_forecaster_sets(i, forecasts, outcomes)` that Brier-scores forecaster `i` on the given forecasts matrix and outcomes vector.
10. Adapt the code from last class that used the `qsr_forecaster` function to obtain the ranking of forecasters in the training set. Create a variable `k` and save the top `k` forecasters on the training set in `topk_forecasters_train`.
11. Compute the mean of forecasts from only those top `k` on the test set and Brier-score it. Compare it to the mean of all forecasters on the test set. What do you see for different values of `l` and `k`?

Prof. Dr. Jens Witkowski
Assistant Professor of Computer Science and Management

Frankfurt School of Finance & Management
Adickesallee 32-34
60322 Frankfurt am Main
Germany

Email: j.witkowski@fs.de
Tel.: +49 (0) 69-154008-875
Fax: +49 (0) 69-154008-4875

Class 10

Bayesian Models & Forecasting Competitions



Computational Statistics

Prof. Dr. Jens Witkowski
September 26, 2023

Agenda

- Session 1
 - Bayesian Models
- Session 2
 - Forecasting Competitions
 - Trivia Quiz

Joint Probability

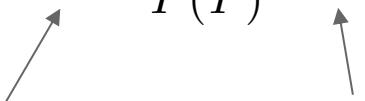
Let X_1, \dots, X_n be n random variables.

Joint distribution is denoted by $P(X_1, \dots, X_n)$ with atomic events being assignments x_1, \dots, x_n .

Two rules for joint distributions:

1. Sum rule (marginalization): $P(X_i) = \sum_{x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_n} P(x_1, \dots, x_{i-1}, X_i, x_{i+1}, \dots, x_n)$
2. Product rule (chain rule): $P(X_1, \dots, X_n) = P(X_1) \cdot P(X_2|X_1) \cdot \dots \cdot P(X_n|X_1, \dots, X_{n-1})$

$$\text{Bayes' rule: } P(X|Y) = \frac{P(X, Y)}{P(Y)} = \frac{P(Y|X)P(X)}{\sum_{X=x} P(Y|X=x)P(X=x)}$$

Def. (cond. probability) / product rule

Product and sum rule

Conditional Independence

RVs X_1, \dots, X_n are *independent* if and only if for all x_1, \dots, x_n :

$$P(X_1 = x_1, \dots, X_n = x_n) = P(x_1) \cdot P(x_2) \cdot \dots \cdot P(x_n)$$

RVs X and Y are *conditionally independent* given Z if and only if for all x, y, z :

$$P(X = x, Y = y | Z = z) = P(X = x | Z = z) \cdot P(Y = y | Z = z)$$

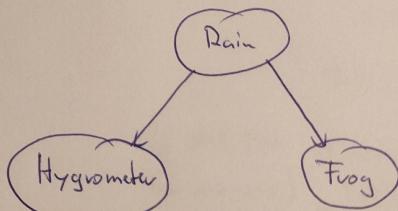
If $P(Y = y | Z = z) > 0$, this is equivalent to $P(X = x | Z = z, Y = y) = P(X = x | Z = z)$.

(condition needed since $P(X = x | Z = z, Y = y) = \frac{P(X = x, Z = z, Y = y)}{P(Z = z, Y = y)}$ otherwise undefined)

Naive Bayes Model: Multiple effects with a single cause (e.g., flu causes fever, runny nose, cough, ...). Effects (fever, runny nose, cough, ...) are conditionally independent given cause.

Naive Bayes Model (from whiteboard)

$$P(R=0) = 0.5 \quad P(R=1) = 0.5$$



R	$P(H=0)$	$P(H=1)$
0	0.6	0.4
1	0.2	0.8

R	$P(F=0)$	$P(F=1)$
0	0.6	0.4
1	0.2	0.8

Reminder: Bayes' Rule

$$P(A|B) = \frac{P(B|A) P(A)}{P(B)}$$

Update following hygrometer signal observation:

$$P(R=1|H=1) = \frac{P(H=1|R=1) P(R=1)}{P(H=1)} = \frac{0.8 \cdot 0.5}{P(H=1)} = \frac{0.8 \cdot 0.5}{0.6} = \underline{\underline{0.6}}$$

$$P(H=1) = P(H=1|R=1) P(R=1) + P(H=1|R=0) P(R=0) = 0.8 \cdot 0.5 + 0.4 \cdot 0.5 = 0.6$$

$$P(R=0|H=1) = 1 - P(R=1|H=1) = \underline{\underline{0.4}} \quad P(H=0) = 1 - P(H=1) = 0.4$$

$$P(R=1|H=0) = \frac{P(H=0|R=1) P(R=1)}{P(H=0)} = \frac{0.2 \cdot 0.5}{0.4} = \underline{\underline{0.25}}$$

$$P(R=0|H=0) = 1 - P(R=1|H=0) = 0.75$$

Conditional Probability Table (CPT) of "Frog" identical to "Hygrometer", hence same numbers/probabilities.

Following observation of (either "Frog" or) "Hygrometer" signal, one of two possible beliefs: if $H=0$, belief that it will rain is 0.25;
if $H=1$, belief that it will rain is 0.6.

Update following observation of both hygrometer and frog signals:

$$P(R=1|H=1, F=1) = \frac{P(R=1, H=1, F=1)}{P(H=1, F=1)} = \frac{P(R=1) P(H=1|R=1) P(F=1|R=1)}{0.4}$$

$$\xrightarrow{P(H=1, F=1) = P(R=1) P(H=1|R=1) P(F=1|R=1) + P(R=0) P(H=1|R=0) P(F=1|R=0)}$$

$$= 0.5 \cdot 0.8 \cdot 0.8 + 0.5 \cdot 0.4 \cdot 0.4 = 0.4$$

$$P(R=1|H=0, F=0) = \frac{0.5 \cdot 0.2 \cdot 0.2}{0.5 \cdot 0.2 \cdot 0.2 + 0.5 \cdot 0.6 \cdot 0.6} = \underline{\underline{0.1}}$$

$$P(R=1|H=1, F=0) = P(R=1|H=0, F=1) = \frac{0.5 \cdot 0.2 \cdot 0.8}{0.5 \cdot 0.2 \cdot 0.8 + 0.5 \cdot 0.4 \cdot 0.6} = \underline{\underline{0.4}}$$

Forecasting Competitions

From Scores to Forecaster Rankings

- So far: assumption that a probabilistic forecaster gets paid her score.
- Could extend to several forecasters by paying each individually according to proper scoring rule.
- In practice, however, proper scores mostly used to rank forecasters.

Sports Forecasting

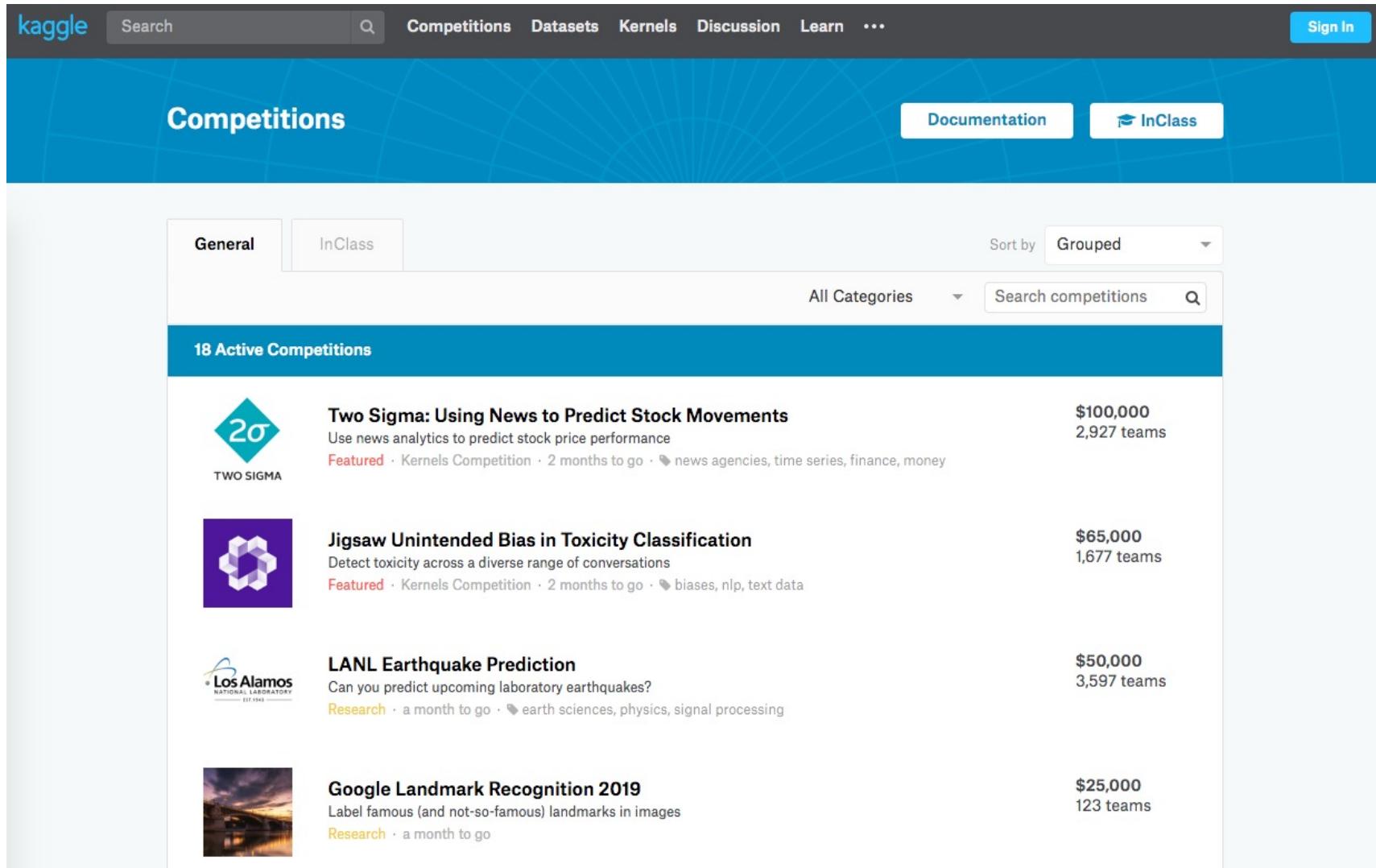
Season leaderboard

Entire season



RANK	NAME	POINTS	PERCENTILE
1	Neil Paine	+1,377.4	99 th
2	Dave Dexter	+1,352.6	99 th
3	Han Zhang	+1,333.4	99 th
4	Jeffrey Starns	+1,328.4	99 th
5	David Marshall	+1,321.0	99 th
6	Tom Skinner	+1,308.1	99 th
7	Dean Jens	+1,297.9	99 th
8	Scott Spitze	+1,294.9	99 th
10	Sean E.	+1,287.0	99 th
11	Peter W	+1,286.4	99 th
12	Noel Raine	+1,285.7	99 th
13	Don Leake	+1,278.2	99 th
14	Darren Symons	+1,268.3	99 th
15	Laurier Bear	+1,252.0	99 th

Machine Learning Competitions



The screenshot shows the Kaggle Competitions page with the following details:

- General** tab selected (highlighted in blue).
- InClass** tab available.
- Sort by**: Grouped (dropdown menu).
- All Categories** dropdown.
- Search competitions** search bar.
- 18 Active Competitions** (blue header bar).
- Two Sigma: Using News to Predict Stock Movements** (by Two Sigma):
 - Thumbnail: Two Sigma logo (blue diamond with '2σ').
 - Description: Use news analytics to predict stock price performance.
 - Status: Featured · Kernels Competition · 2 months to go · news agencies, time series, finance, money.
 - Prize: \$100,000.
 - Teams: 2,927 teams.
- Jigsaw Unintended Bias in Toxicity Classification** (by Jigsaw):
 - Thumbnail: Jigsaw logo (purple hexagonal pattern).
 - Description: Detect toxicity across a diverse range of conversations.
 - Status: Featured · Kernels Competition · 2 months to go · biases, nlp, text data.
 - Prize: \$65,000.
 - Teams: 1,677 teams.
- LANL Earthquake Prediction** (by Los Alamos National Laboratory):
 - Thumbnail: LANL logo (blue circular emblem).
 - Description: Can you predict upcoming laboratory earthquakes?
 - Status: Research · a month to go · earth sciences, physics, signal processing.
 - Prize: \$50,000.
 - Teams: 3,597 teams.
- Google Landmark Recognition 2019** (by Google):
 - Thumbnail: Landmark recognition image (city skyline at sunset).
 - Description: Label famous (and not-so-famous) landmarks in images.
 - Status: Research · a month to go.
 - Prize: \$25,000.
 - Teams: 123 teams.

Machine Learning Competitions (cont.)

Overview Data Kernels Discussion Leaderboard **Rules**

Rules

OFFICIAL COMPETITION RULES

COMPETITION TITLE: Mercari Price Suggestion Challenge

COMPETITION SPONSOR: Mercari, Inc.

COMPETITION WEBSITE: <https://www.kaggle.com/c/mercari-price-suggestion-challenge/>

PRIZES

In addition to job opportunities at Mercari, three cash prizes will be awarded to the top teams.

First Prize: \$60,000

Second Prize: \$30,000

Third Prize: \$10,000

Forecasting Competitions

In practice, forecasters are competing with others:

- Netflix prize: \$1,000,000 to best model.
- Kaggle: prizes and companies hire best-performing developers.
- Good Judgment Project: top 2% ("superforecaster" status).

Does winner-takes-all competition affect truthfulness?

Naive Approach: Winner-takes-all Selection

Select forecaster with highest $R_q(y, x) = 1 - (y - x)^2$.

Not truthful!

2-Forecaster Example



Both truthful \Rightarrow same selection probability of 0.5.

Misreport $y_1 > 0.8 \Rightarrow$ forecaster 1 has highest score iff $x = 1$, occurring with probability 0.8 \Rightarrow advantageous misreport.

Real World: Gaming Kaggle Competitions

Gaming of forecasting/ML competitions does happen in real life and can be successful:

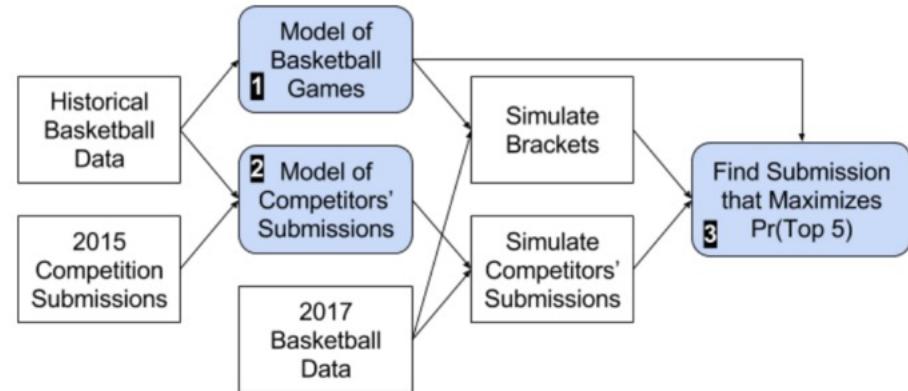
March Machine Learning Mania, 1st
Place Winner's Interview: Andrew
Landgraf

 Kaggle Team [Follow](#)
May 19, 2017 · 5 min read

[Twitter](#) [LinkedIn](#) [Facebook](#) [Bookmark](#)



Kaggle's [2017 March Machine Learning Mania competition](#) challenged Kagglers to do what millions of sports fans do every year—try to predict the winners and losers of the US men's college basketball tournament. In this winner's interview, 1st place winner, Andrew Landgraf, describes how he cleverly analyzed his competition to optimize his luck.



Truthful Forecasting Competition for n=2 Forecasters

Select forecaster i with probability f_i :

$$\begin{aligned} f_i &= 0.5 \cdot R_q(y_i, x) - 0.5 \cdot R_q(y_j, x) + 0.5 \\ &= 0.5 + 0.5 \left(R_q(y_i, x) - R_q(y_j, x) \right) \end{aligned}$$

Truthful!

Confirm that f_i form valid distribution: $f_1, f_2 \geq 0$ and $f_1 + f_2 = 1$

Alternative interpretation of mechanism:

- ① Give forecasters 0.5 probability to start.
- ② Adjust up or down depending on performance compared with others.

Truthful Forecasting Competition for n Forecasters

Select forecaster i with probability f_i :

$$f_i = \frac{1}{n} + \frac{1}{n} \left(R_q(y_i, x) - \frac{1}{n-1} \sum_{j \neq i} R_q(y_j, x) \right).$$

1. Truthful!
2. Selects most accurate forecaster with highest probability!
3. Generalizes to multiple events when using scores averaged over events.

Python Practice

1. Load `forecast-aggregation.py` (last class) and run it once.
2. Create a function `selection_prob(fcast_vec, single_outcome)` that computes the vector of selection probabilities from the truthful forecasting competition mechanism and takes as inputs (1) a vector of different forecasters' forecasts corresponding to the "yes" outcome of a single question (as 1D array) and (2) the outcome (0 or 1).
3. Apply `selection_prob` to the first question of `fcasts`. What do you notice regarding the probabilities?
4. For settings with several independent questions, a generalized mechanism can be used: for each question, select a winner using the single-event mechanism and then select the forecaster having won the most events as the overall winner (ties broken uniformly at random).
Write a function `overall_winner(fcasts_matrix, outcomes)` that takes a forecast matrix (e.g., `fcasts`) and an outcome vector as input and outputs a the winner of the described multi-question mechanism.
5. Write a function `overall_winner_ranking(fcasts_matrix, outcomes)` that samples 10,000 winners from `overall_winner` to estimate the probability of selection for different forecasters. Output the ranking of forecasters by estimated selection probability. How does this compare to the order of average Brier scores?

Prof. Dr. Jens Witkowski
Assistant Professor of Computer Science and Management

Frankfurt School of Finance & Management
Adickesallee 32-34
60322 Frankfurt am Main
Germany

Email: j.witkowski@fs.de
Tel.: +49 (0) 69-154008-875
Fax: +49 (0) 69-154008-4875

Class 11

Peer Prediction & Conclusion



Computational Statistics

Prof. Dr. Jens Witkowski
October 2, 2023

Agenda

- Session 1
 - Course Evaluations
 - Homework 1 Solutions
 - Peer Prediction
 - Trivia Quiz Analysis
- Session 2
 - Exam Preparation

Homework 1

1. Download `hw1.csv` and load it as `df` into your Python environment. You want to predict the data in column 1 from the data in column 0.
 - a) Plot the data with column 0 on the x axis and column 1 on the y axis.
 - b) Fit polynomial models with degree from 1 to 20 and implement 5-fold cross validation. Which of these 20 model classes has lowest validation MSE? Briefly discuss.
 - c) Set aside 10% of the data in a test set and then reuse your code from part b) to run 5-fold cross validation on the remaining data. Which of the 20 model classes has lowest validation MSE and what is that model's expected generalization error?
 - d) Train the model class you found to be best on the entire data set and provide the resulting model as `best_model` in your code.

(High-level) Solution: Optimal degree is 4.

Homework 1 (cont.)

2. Extend the code in `linear_regression.py` (from lecture) with functions that take a feature matrix and return a feature matrix with new, transformed (typically: non-linear) features.
 - a) Implement a short function `transformed_feature_matrix_quadratic(X)` that adds a quadratic feature to the feature space. For example, for the current dataset, calling `transformed_feature_matrix_quadratic(X)` should return:

```
>>> array([[ 1. ,  1. ],
       [ 2. ,  4. ],
       [ 3. ,  9. ],
       [ 4. , 16. ],
       [ 5. , 25. ],
       [ 6. , 36. ],
       [ 7. , 49. ],
       [ 2.5 ,  6.25]])
```

Apply this function to fit the `LinearRegression` object using these transformed features and to plot the model's predictions. Do not create new variables for these transformed features and do not add any additional libraries to `linear_regression.py`.

Solution:

```
def transformed_feature_matrix_quadratic(X):
    return np.append(X, X**2, axis=1)

...
polyreg.fit(transform_feature_matrix_quadratic(X), Y)
plt.plot(X_seq, polyreg.predict(transformed_feature_matrix_quadratic(X_seq)))
```

Homework 1 (cont.)

2. Extend the code in `linear_regression.py` (from lecture) with functions that take a feature matrix and return a feature matrix with new, transformed (typically: non-linear) features.
 - b) Similarly, implement a short function `transformed_feature_matrix_singlecube(X)` that returns a feature matrix with a single feature that has been cubed. (Again, also fit and plot the model as in a)). For the current dataset, the function would return:

```
>>> array([[ 1.    ],
       [ 8.    ],
       [ 27.   ],
       [ 64.   ],
       [125.   ],
       [216.   ],
       [343.   ],
       [ 15.625]])
```

Solution:

```
def transformed_feature_matrix_singlecube(X):
    return(X**3)
```

(remainder analogous to a)

Homework 1 (cont.)

2. Extend the code in `linear_regression.py` (from lecture) with functions that take a feature matrix and return a feature matrix with new, transformed (typically: non-linear) features.
 - c) Can you find a single feature transformation that achieves a (training) MSE smaller than 0.06 on the full dataset with that single feature? (You do not need to code any optimization but are allowed to just try out different transformations.) Implement this as function `transformed_feature_matrix_singlemin(x)`. Also fit and plot the model as before and provide a short discussion of your result: How did you find it? What is the intuition behind it? What might be problematic about this from an applied data science perspective?

Solution: Shift original feature by 4 and square.

```
def transformed_feature_matrix_singlemin(x):
    return (x-4)**2
```

Can find this by either looking at the plot and making an educated guess or by simply using the polynomial of degree 2 that we found in a) as the new, single feature (using the coefficients we found there). The former is more difficult in the usual setting with higher-dimensional data (but people do look at plots when engineering features!). The latter requires you to have found a good-fitting function already. Of course, this feature is very much hand-tailored to the dataset at hand and will not work well on other datasets.

Aggregating Categorical Information

Categorical Information

So far: elicitation and aggregation of forecasts:

The Good Judgment Project™



Often want to aggregate answers to categorical questions:



"Should we outsource our IT?"

"Do you prefer Coca Cola over Pepsi?"

"Is Riga the capital of Lithuania?"

Majority Vote

"Is Riga the capital of Lithuania?"



Example:

10 people say "yes"

5 people say "no."

Majority says "yes" => Aggregate to "yes."

Problem: The majority is typically wrong here. Most people will say it is Riga but it is Vilnius.

Want an aggregation algorithm that can rediscover ground truth even when the majority is wrong!

Surprisingly Popular Algorithm (Binary)

Idea: ask for additional report regarding belief of signal responses in population.

Information report $x_i \in \{0, 1\}$: "Is Riga the capital of Lithuania?"

Prediction report $y_i \in [0, 1]$: "What do you think is the fraction of other respondents saying that it is?"

Surprisingly Popular (SP) Algorithm:

1. Compute average of information reports over all respondents.
2. Compute average of prediction reports over all respondents.
3. Select the answer option that has been reported more often than predicted (i.e., is "surprisingly popular").

Example:

10 people say "yes" and predict that 80% will say yes and 20% will say no.

5 people say "no" and predict that 60% will say yes and 40% will say no.

(They "no" voters know it's Vilnius but are aware most other people won't know!)

Average information report: 2/3

Average prediction report: 0.73 because $(10*0.8+5*0.6)/15 = 11/15 = 0.73$.

Majority says "yes" but "no" answer is "surprisingly popular" because $2/3 < 11/15 \Rightarrow$ Aggregate to "no."

Surprisingly Popular Algorithm (Categorical)

Can also be used for more than two answer options (with some caveats).

Let **x** be the empirical frequency of all information reports.

Let **y** be average of all prediction reports.

Select as best guess for ground truth answer the dimension with the largest value in **x-y**.

Example with 5 agents and 3 possible answers:

Information report: "What is the capital of Lithuania?"

A: "Vilnius"

B: "Riga"

C: "Tallinn"

Prediction report: "What do you think is the fraction of people reporting A, B, and C?"

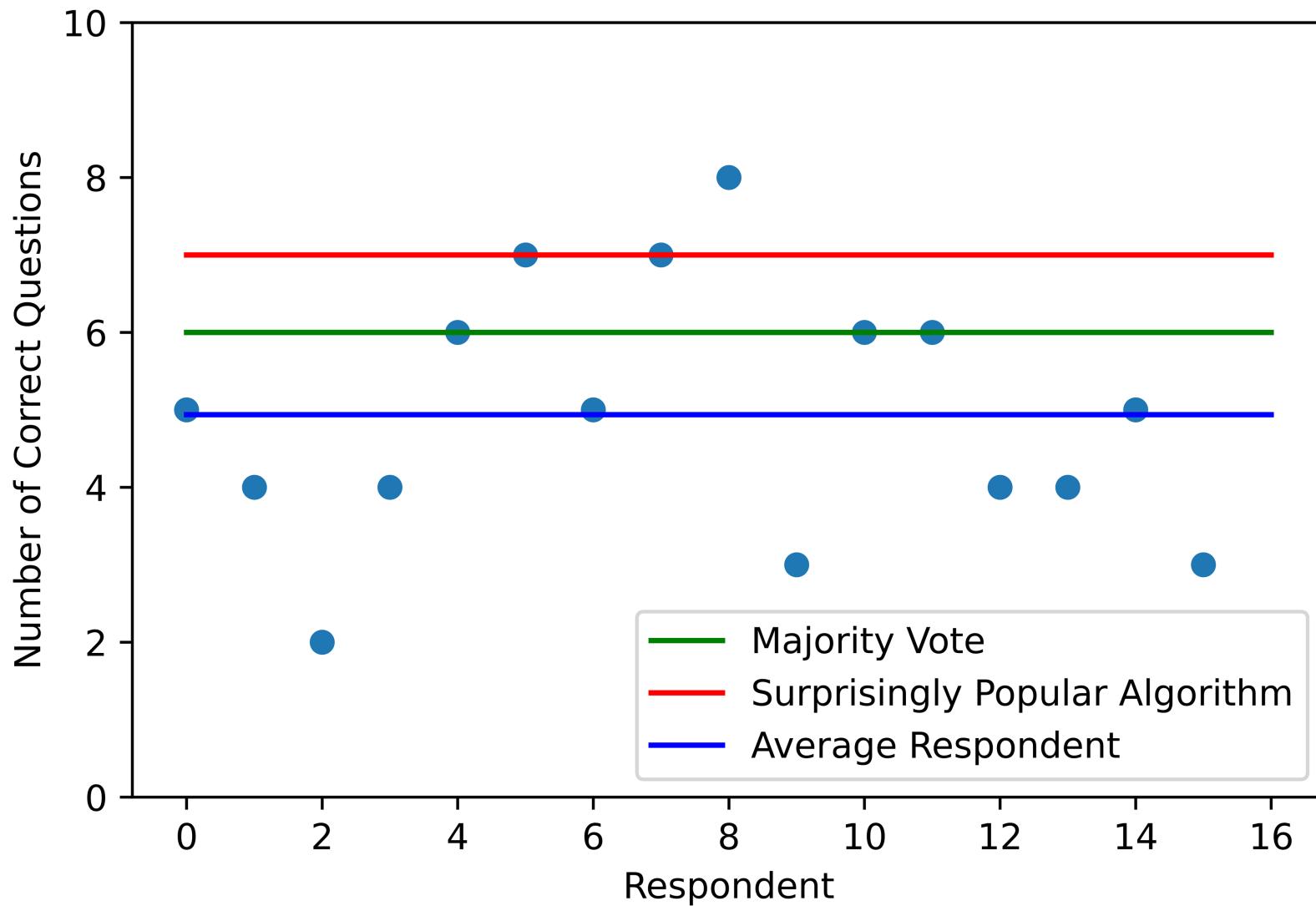
2 agents report A, 3 agents report B, no one reports C. The prediction reports of the agents reporting A are (0.4, 0.5, 0.1), and those of the agents reporting B are (0.15, 0.5, 0.35) .

Which answer should be selected?

$\mathbf{x-y} = (0.4, 0.6, 0.0) - (0.25, 0.5, 0.25) = (0.15, 0.1, -0.25)$, so A is answer with largest value.

**Answer A is most "surprisingly popular" (in absolute terms):
largest difference between fraction who reported it and average of its prediction!**

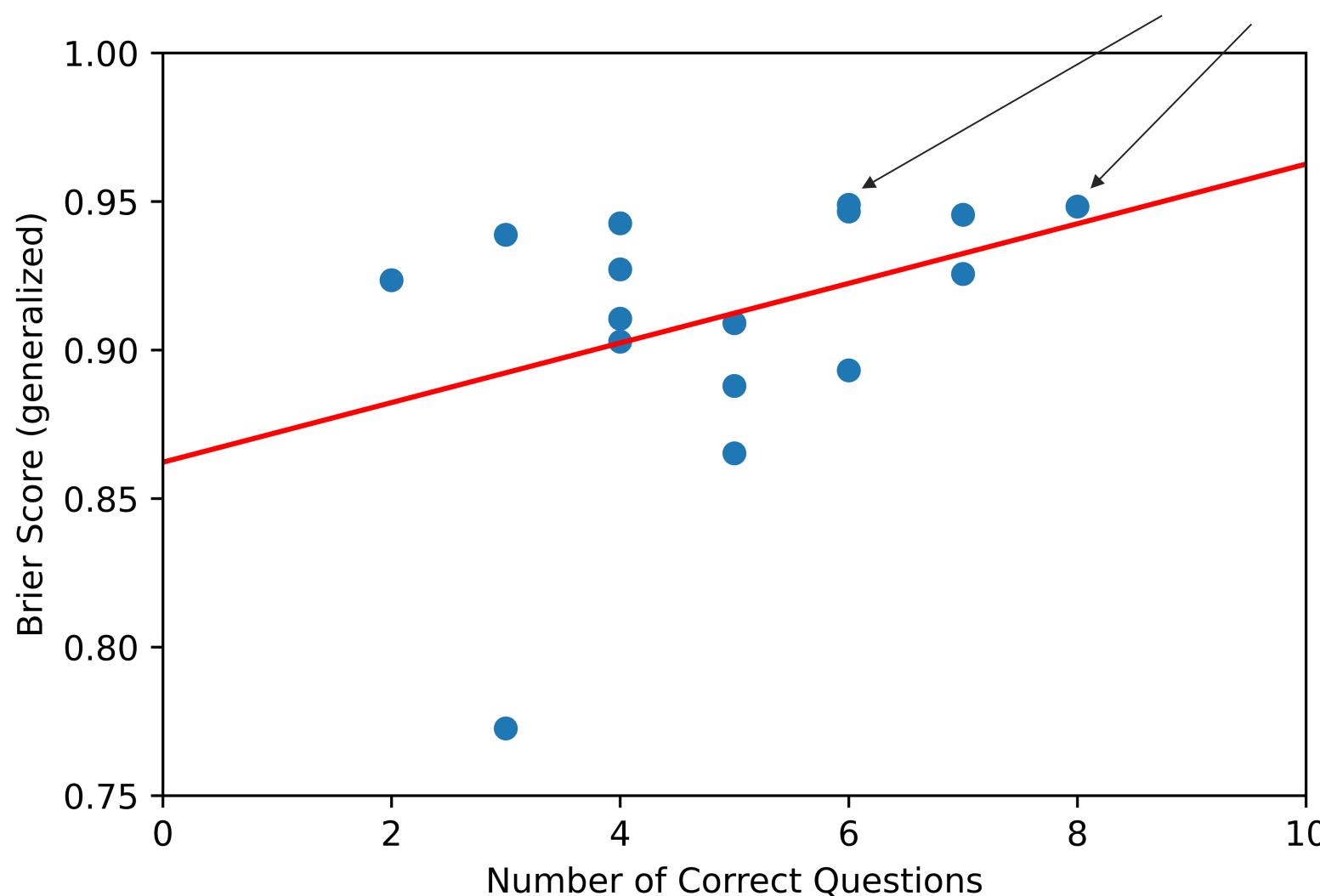
Trivia Data



Trivia Data (cont.)

How good are you at predicting everybody's answers?

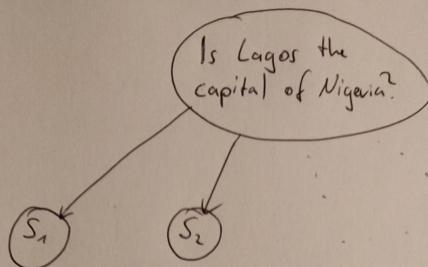
Want to know who this is?



Intuition for SP Algorithm (from whiteboard)

Model

$$P(L=0) = 0.3 \quad P(L=1) = 0.7$$



Conditional Probability Table (same for all respondents):

L	$P(S=0)$	$P(S=1)$
0	0.4	0.6
1	0.1	0.9

Belief update on Lagos being the capital based on signal observation:

$$P(L=1 | S=0) = \frac{P(S=0 | L=1) P(L=1)}{P(S=0)} = \frac{0.1 \cdot 0.7}{0.1 \cdot 0.7 + 0.4 \cdot 0.3} = \frac{7}{19} = 0.37$$

$$P(L=1 | S=1) = \frac{P(S=1 | L=1) P(L=1)}{P(S=1)} = \frac{0.9 \cdot 0.7}{0.9 \cdot 0.7 + 0.6 \cdot 0.3} = \frac{7}{9} = 0.77$$

Belief about others:

$$P(S_2=1 | S_1=1) = P(S_2=1 | L=1) \cdot P(L=1 | S_1=1) + P(S_2=1 | L=0) \cdot P(L=0 | S_1=1) \\ = 0.9 \cdot \frac{7}{9} + 0.6 \cdot \frac{2}{9} = \frac{5}{6} = 0.83$$

$$P(S_2=1 | S_1=0) = P(S_2=1 | L=1) \cdot P(L=1 | S_1=0) + P(S_2=1 | L=0) \cdot P(L=0 | S_1=0) \\ = 0.9 \cdot \frac{7}{19} + 0.6 \cdot \frac{12}{19} = \frac{27}{38} = 0.71$$

Long-run average (w/ many respondents):

If $L=0$: Answers (roughly) 60% "yes" and 40% "no."

NB: Majority is wrong!

What about SP algorithm?

Average prediction report: $0.6 \cdot \frac{5}{6} + 0.4 \cdot \frac{27}{38} = \frac{149}{190} = 0.78$

SP algorithm: "We expected 78% of 'yes' votes and 22% of 'no' votes, but got 60% of 'yes' votes and 40% of 'no' votes."

\Rightarrow SP selects "no" because $0.4 > 0.22$.

If $L=1$: Answers (roughly) 90% "yes" and 10% "no."

Majority correct.

What about SP algorithm?

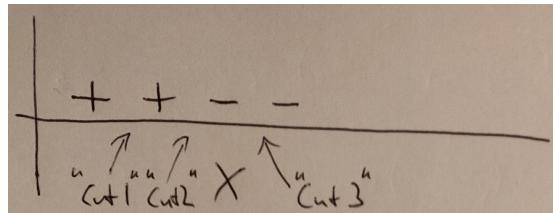
Average prediction report: $0.9 \cdot \frac{5}{6} + 0.1 \cdot \frac{27}{38} = \frac{78}{95} = 0.82$

\Rightarrow SP selects "yes" because $0.9 > 0.82$.

Exam Preparation

Question 1 (from whiteboard)

You are given the following training dataset with 4 points and a single feature:



Provide the mean loss for the entire dataset and each possible cut/split using:

a) 0/1 loss:

$$\begin{aligned} \text{Before any cut/split: } & \frac{1}{2} \\ \text{After cut 1: } & \frac{1}{4} \cdot 0 + \frac{3}{4} \cdot \frac{1}{3} = \frac{1}{4} \\ \text{After cut 2: } & \frac{1}{2} \cdot 0 + \frac{1}{2} \cdot 0 = 0 \\ \text{After cut 3: } & \frac{3}{4} \cdot \frac{1}{3} + \frac{1}{4} \cdot 0 = \frac{1}{4} \end{aligned}$$

b) Entropy loss:

$$\begin{aligned} \text{Before any cut/split: } & -0.5 \log(0.5) - 0.5 \log(0.5) = 0.693 \\ \text{After cut 1: } & \frac{1}{4} \cdot (0) + \frac{3}{4} \cdot \left(-\frac{1}{3} \log\left(\frac{1}{3}\right) - \frac{2}{3} \log\left(\frac{2}{3}\right) \right) = 0.477 \\ \text{After cut 2: } & \frac{1}{2} \cdot (0) + \frac{1}{2} \cdot (0) = 0 \\ \text{After cut 3: } & 0.477 \end{aligned}$$

Connection to Value of Information

Dataset A contains 4 data points, 2 of which have label "+" with the other 2 having label "-". You are asked to predict the probability that a random data point drawn uniformly at random from A has label "+" and you will be rewarded using the logarithmic scoring rule.

Case 1: You have no further information.

In that case, your belief is 0.5 and your expected score reporting truthfully is $0.5 \log(0.5) + 0.5 \log(0.5) = -0.693$

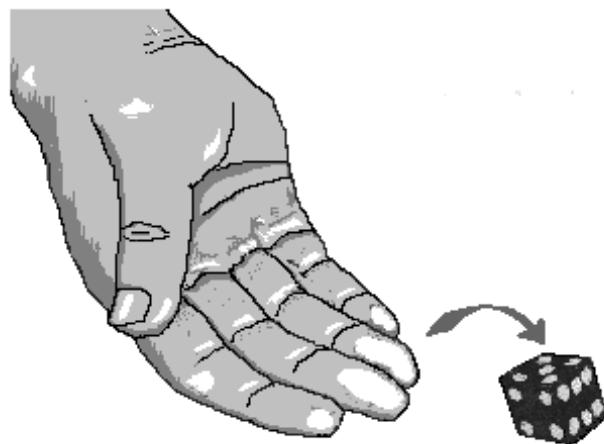
Case 2: An informant tells you that the dataset has been split into 2 subsets, one containing only 1 data point with label "+" and the other containing the 3 remaining data points (1 with label "+" and 2 with label "-"). The informant also offers to tell you the subset/region in which the data point you are asked to predict the label for lies. How much would you be willing to pay for this information?

Expected score knowing subset: $\frac{1}{4} * 0 + \frac{3}{4} * (1/3 \log(1/3) + 2/3 \log(2/3)) = -0.477$.

Expected value of information: $-0.477 - (-0.693) = 0.216$. Hence, the information provided by the informant is worth up to 0.216 to you.

The only difference to the entropy loss from the previous question is that the logarithmic scoring rule is a reward function (higher is better) and the entropy loss is a loss function (lower is better). You can interpret the entropy loss splitting criterion as asking "How much does my information improve by this split?" ("information gain")

Connection to Value of Information (cont.)



Probability that "6" comes up?

You are paid according to quadratic scoring rule:

$$R_q(y, x) = 1 - (y - x)^2$$

What would you report? What is expected score of that report?

Expected score of truthful report:

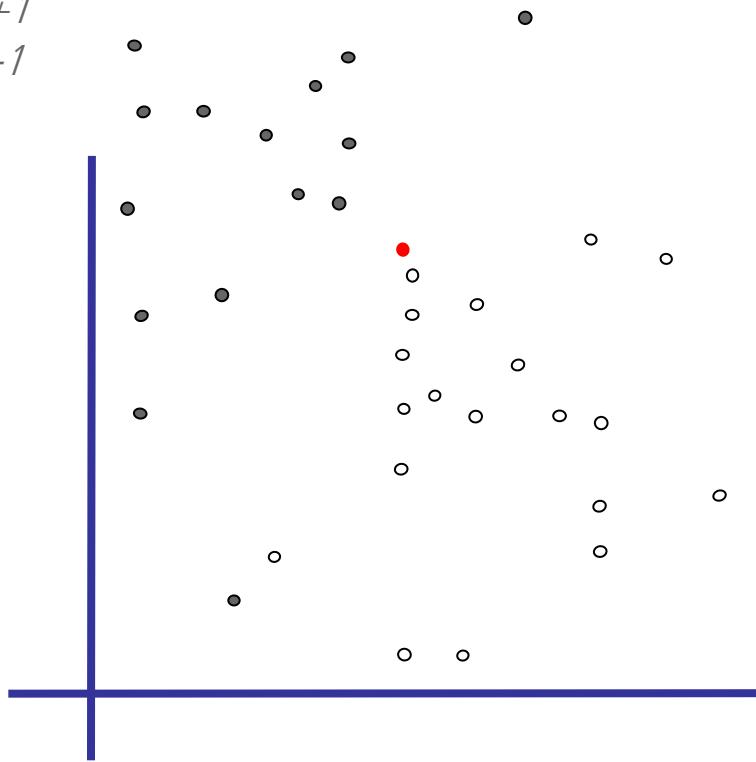
$$\frac{1}{6} \cdot R_q\left(\frac{1}{6}, 1\right) + \frac{5}{6} \cdot R_q\left(\frac{1}{6}, 0\right) = \frac{31}{36} = 0.861111$$

Assume someone could tell you whether the number that came up was "less than 5" (1,2,3,4) before you need to report a probability. How much would that information be worth to you?

Same idea as in decision tree learning with Gini index or entropy: "Which split on which feature provides the most information (i.e., reduces uncertainty the most)?"

Question 2

- denotes +1
- denotes -1



The red data point is a new point for whose label we would like to predict.

- a) Which label would be assigned by a linear support vector classifier with 0 misclassifications on the training data?
- +1
 - 1
 - Either class is possible.

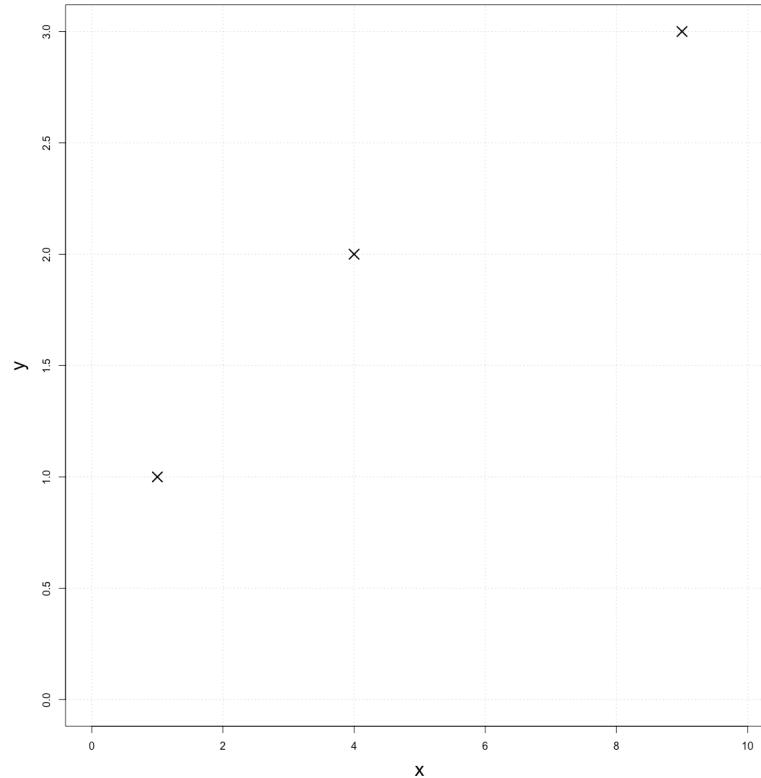
Answer: i.

- b) Briefly explain why classifiers with large margins are preferred over classifiers with small margins.

Answer. A larger margin gives us more robustness to individual observations and (theoretically and empirically) provides better generalization error on unseen data points. Classifiers with small margins risk overfitting to the training data.

Question 3

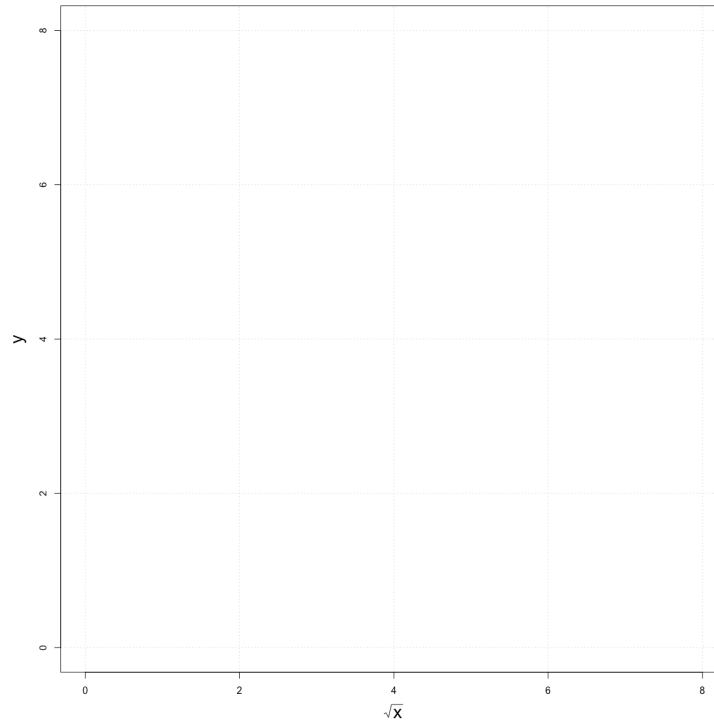
The data frame `df` contains 3 points: $(x_1, y_1) = (1, 1)$, $(x_2, y_2) = (4, 2)$, and $(x_3, y_3) = (9, 3)$ as shown here:



You want to fit a linear regression model to predict y from x with a single transformed feature $\text{sqrt}(x)$.

Question 3 (cont.)

a) Provide all 3 data points in the transformed feature space with the square root of x instead of x :



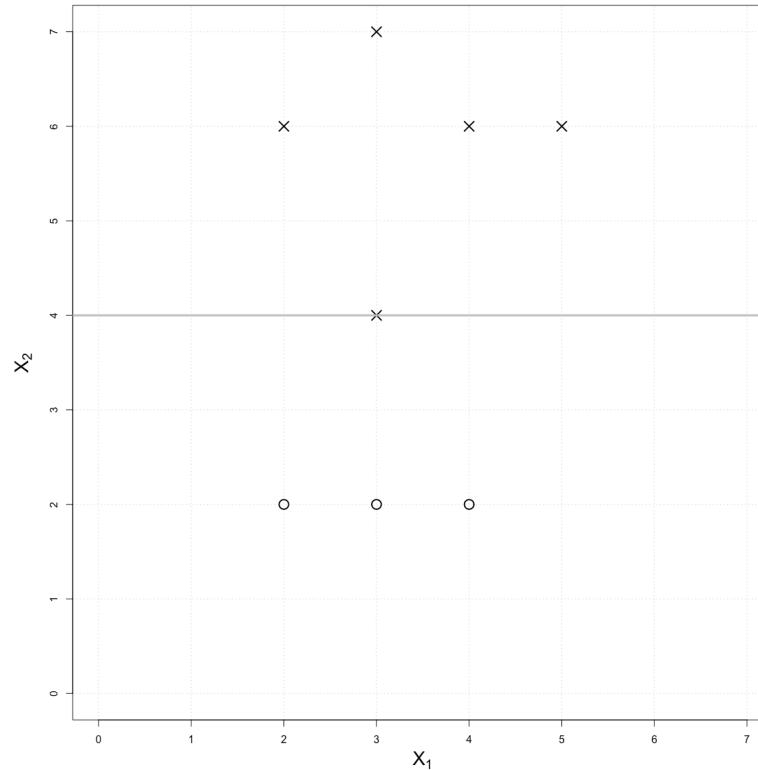
Solution: The points are at $(1,1)$, $(2,2)$, and $(3,3)$

b) Assume you fit a linear regression model using this transformed feature. What is the model's MSE on the three points? What is the leave-one-out cross validation MSE?

Solution: 0 and 0

Question 4

You are presented with the following data set with two classes (crosses for "+1", circles for "-1"), and you want to classify the points using the support vector classifier with soft margin. For your first choice of budget C, you obtain the following hyperplane (in gray) with a margin width of 4:



Draw the margin using dashed lines ("---"), and circle all support vectors.

Solution: The margin planes are at $X_2 = 2$ and $X_2 = 6$. Every data point is a support vector except for (3, 7).

Question 5

A large biochemical company needs accurate demand predictions for their fertilizer in order to decide how much to ship. You have done market analysis and believe demand will be at 3 million tons with probability 1/3 and at 12 million tons with probability 2/3. The company asks you to report your estimate and will pay you $90 - (y - x)^2$ dollars, where y is your report in million tons and x is the actual demand in million tons that eventually materialized.

- a) What payment do you expect when you report y ?

Solution:

$$\begin{aligned}
 & \frac{1}{3} (90 - (3 - y)^2) + \frac{2}{3} (90 - (12 - y)^2) \\
 &= \frac{1}{3} (90 - (9 + y^2 - 6y)) + \frac{2}{3} (90 - (144 + y^2 - 24y)) \\
 &= \frac{1}{3} (81 - y^2 + 6y) + \frac{2}{3} (-54 - y^2 + 24y) \\
 &= 27 - \frac{1}{3}y^2 + 2y - 36 - \frac{2}{3}y^2 + 16y \\
 &= -9 - y^2 + 18y
 \end{aligned}$$

Question 5 (cont.)

- b) What is the report maximizing your expected payment? What is your expected payment at that optimal report?

Solution:

Take derivative of expected payment and set to 0:

$$-2y + 18 := 0 \Rightarrow y = 9$$

The expected payment for $y=9$ is:

$$-9 - 9^2 + 18 \cdot 9 = -9 - 81 + 162 = 72$$

Question 6

Consider the quadratic scoring rule $R_q(y, x) = 1 - (y - x)^2$ and let $\theta = 0.25$ be the true probability that $x = 1$.

What is the expected score of a forecaster reporting $y = 0.2$?

Solution:

$$\begin{aligned}
 & 0.25 \cdot (1 - (0.2 - 1)^2) + 0.75 \cdot (1 - (0.2 - 0)^2) \\
 &= 0.25 \cdot (1 - 0.64) + 0.75 \cdot 0.96 \\
 &= \frac{1}{4} \cdot \frac{36}{100} + \frac{3}{4} \cdot \frac{96}{100} = \frac{9}{100} + \frac{72}{100} = \frac{81}{100} = 0.81
 \end{aligned}$$

Prof. Dr. Jens Witkowski

Assistant Professor of Computer Science and Management

Frankfurt School of Finance & Management
Adickesallee 32-34
60322 Frankfurt am Main
Germany

Email: j.witkowski@fs.de
Tel.: +49 (0) 69-154008-875
Fax: +49 (0) 69-154008-4875