Computational Communication Science 2
Week 8
Lab session

Roeland Dubèl
r.dubel@uva.nl

2025-05-20

Digital Society Minor
University of Amsterdam

Today

1. Exam
2. Weekly MC-questions
3. Recap on SML
4. Q&A
5. Weekly exercises

Exam

Exam

- 50% of final grade
- Individual in-class open book exam
  - Allowed to use all course materials (scripts, notes, e.g.) → download in advance in a separate desktop folder (e.g. "CCS2 Exam")
  - Not allowed to use the internet and/or AI
- Your understanding of concepts and computational coding will be tested
  - Predominantly about the 2nd part of the course → by Dr. Saurabh Khanna
  - About $^1/_{3_{rd}}$ will be coding → sometimes placeholder code given
  - About $^2/_{3_{rd}}$ will be open answer questions → analytical strategy & reflection
- 27th of May, 12:30 – 14:30, REC A2.10, 90 minutes

Weekly MC-questions

MC-questions Week 8

- Canvas → Modules → Week 8 → MC-questions
- 4 questions, 8 minutes (in silence)
- Afterwards we will discuss the questions

Recap SML

Introduction

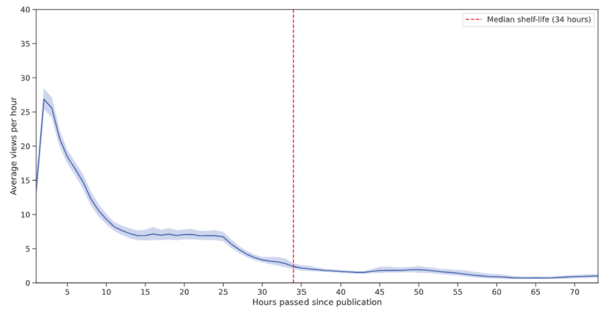
Supervised machine learning → learning from examples with answers

- X – Features/Inputs/X
- y – Labels/Outputs/Y (ground truth/human annotations)
- $f : X \rightarrow y$ – The model learns from the data to be able to predict unseen cases

Regression → predicting a number
Classification → predicting a category

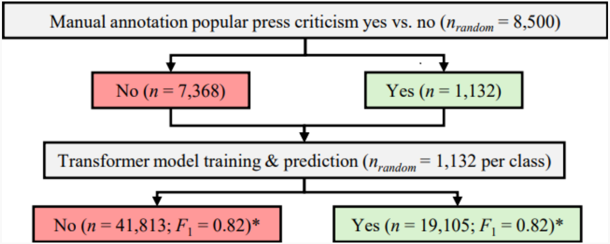
Example Regression

$f : X$[news outlet, author, day and time of publication, ...] $\rightarrow y$[hours of shelf life]



Example Classification

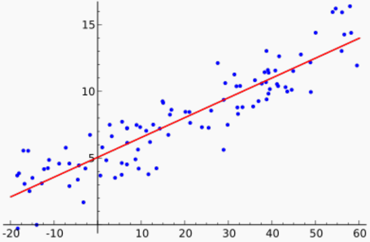
$f: X$[vectorized comments to news channels] $\rightarrow y$[press criticism yes vs. no]



Linear regression

Choosing the right model for our data

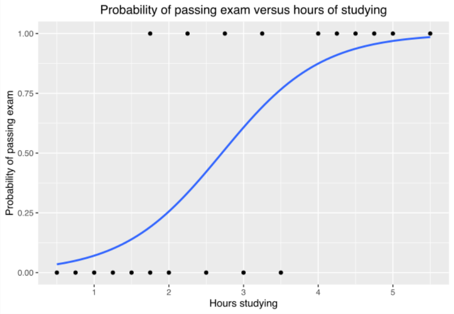
Draws best-fit straight line through data points (least-squares principle)



+ Predicting linear relations
− Predicting curved relations

Logistic regression

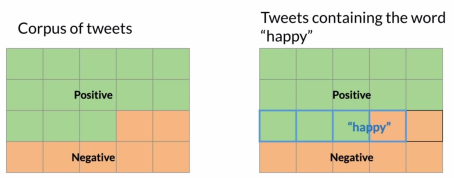
Fitting an S-shaped curve to estimate a probability (0 to 1)



+ Yes vs. no questions (i.e., binary tasks)

Naive Bayes

Learns how likely each feature (e.g., a word/n-gram) is to appear in each class (e.g, count/tf-idf), then adds the evidence from all features and selects the class with the highest total probability

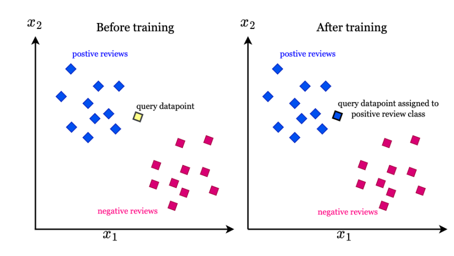


\+ Simple and fast, great for text data

− It's naive: Unrealistic assumption that all inputs are independent from each other given our target labels

k-Nearest Neighbors (K-NN)

Looks at the closest examples and predicts the same label as the majority

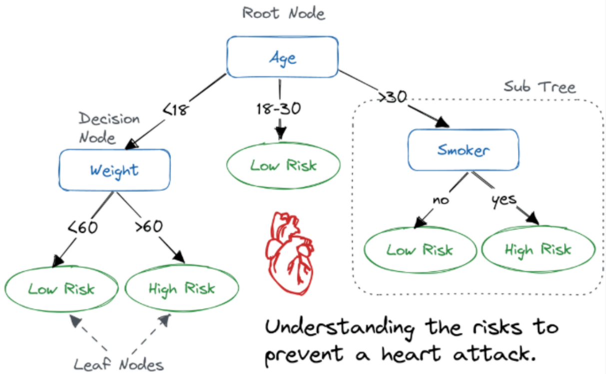


+ Quick start – no training needed (i.e., no parameter learning)

− Slows down if you have tons of examples, sensitive to irrelevant details (i.e., distances can be small)

Decision Trees

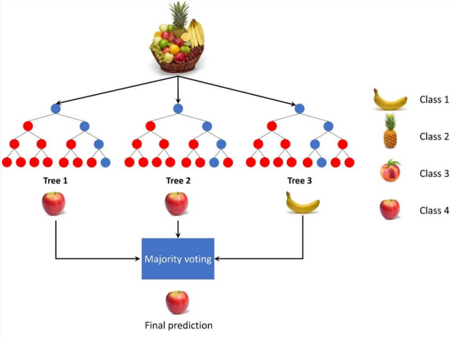
Splits data with yes/no questions in a flowchart



+ Clear rules you can draw and explain
– Chance of overfitting → no turning back

Random forest

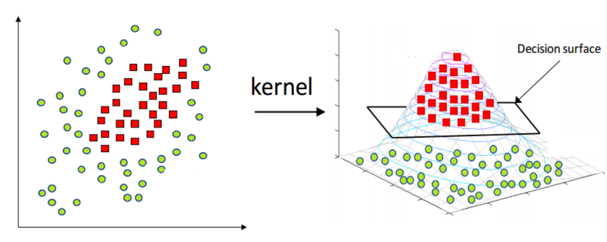
Builds lots of decision trees on different slices of data, then averages their answers (i.e., majority vote)



+ Stronger accuracy than a single tree
− Harder to explain all the little trees

Support Vector Machine (SVM)

Finds the line (or hyperplane; multidimensional line) that leaves the biggest gap between categories

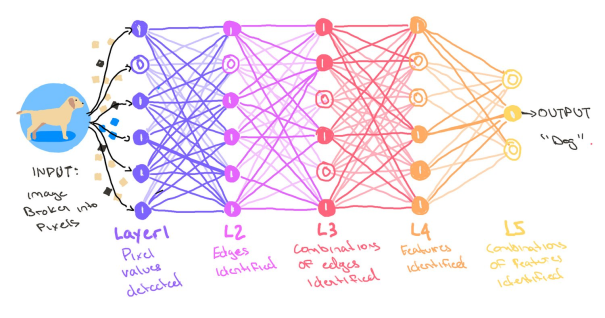


+ High-dimensional data (lots of features)

− Choosing how to draw that fence (i.e., the line) can be tricky → if the data isn't nicely separated, you need to specify how to transform your data (i.e., choose a kernel)

Neural Networks

Stacks many neurons (tiny decision-makers) in layers to learn complex patterns



+ Huge datasets – images, audio, text
– Needs a lot of data and can feel like a black box

Training vs. Test Set

Training the model on our data

- Training Set: The portion to train/fit/learn our data
- Test Set: An unseen portion to evaluate the performance

Overfitting vs. Underfitting

- 1% of our data = Training set: Simple patterns are found as too little variance is given → Underfitting
- 100% of our data = Training set: The model memorizes our data instead of recognizing patterns → Overfitting

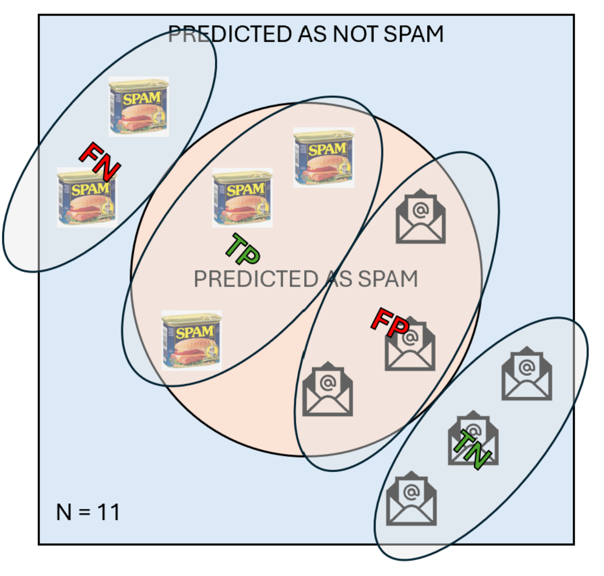
Confusion matrix

Testing the performance of our trained model
→ Counts are needed of whether the predictions align with our ground truth (e.g., human annotations)

	Actual: Spam	**Actual: Not Spam**
Predicted: Spam	True positives (TP)	False positives (FP)
Predicted: Not Spam	False negatives (FN)	True negatives (TN)

- True positives (TP) → spam correctly predicted as spam
- False positives (FP) → not spam incorrectly predicted as spam
- True negatives (TN) → not spam correctly predicted as not spam
- False negatives (FN) → spam incorrectly predicted as not spam

Spam example



Performance metrics

Choosing a performance metric for our model

- Roughly balanced classes of equal importance $\rightarrow$

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{FN} + \text{TN}}$$

- Minimize false positives (i.e., certainty that the prediction is correct) $\rightarrow$

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

- Minimize false negatives (i.e., certainty that positives are not missed) $\rightarrow$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

- Balance precision and recall, useful for imbalanced datasets $\rightarrow$

$$\text{F1-score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

Code example

Putting the steps together in code

```
df = pd.read_csv("data/spam_yes_or_no.csv") # We read in the data
```

```
tweets = df.tweets.to_list() # We seperate the features from the labels
labels = df.labels.to_list()
```

```
plt.hist(labels) # We inspect the balance of the classes
```

```
tweets_train, tweets_test, y_train, y_test = train_test_split(tweets, labels, test_size=0.2, random_state=1) # We split the data
```

```
vectorizer = CountVectorizer() # We define a vectorizer
X_train = countvectorizer.fit_transform(tweets_train) # We transform the data
X_test = countvectorizer.transform(tweets_test) # We transform the data
```

```
nb = MultinomialNB() # We choose a model
nb.fit(X_train, y_train) # We fit the data
y_pred = nb.predict(X_test) # We predict the labels for our test set
```

```
print(classification_report(y_test, y_pred)) # We print the performance metrics
```

Weekly exercises

Weekly exercises: Week 8

- Sit together with your group assignment members
- Go through the weekly exercises (GitHub → week08 → tutorial-exercises.ipynb)
- Reflect on the reproducibility of your group assignment code

Course evaluation

Course evaluation

- Please fill in the course evaluation!

Next week

Next week

- Any questions left?
  - Consultation hours this Friday (23rd of May, 10:00 – 12:00)
- Good luck with the exam!