Inhaltsverzeichnis

1. General topics 1
1.1. Fields 1
1.2. Disciplines 1
2. Data Quality 1
2.1. Data Quality Assessment 1
2.2. Statistical Refresher 1
2.3. Pearson Correlation 1
3. KNN 1
3.1. Euclidean Distance 1
3.2. Cosine Similarity 1
3.3. Manhattan Distance 1
3.4. Levenshtein / Edit Distance 1
3.5. Jaccard Similarity 1
3.6. k-NN 1
3.7. Normalization 1
4. Supervised Learning 1
4.1. Measuring Regression Errors 1
4.2. Regression Models 1
4.3. R-Squared 1
4.4. Fitting 1
4.5. ML Workflow 1
4.6. Brain as Back-Channel 1
4.7. K-Fold-Cross-Validation 1
5. Linear Regression 1
5.1. Terminology 1
5.2. Ordinary Least Squares (OLS) 1
5.3. Regression Performance (R^2) 1
5.4. Multiple Linear Regression 1
5.5. Regularization 1
6. Classification 1
6.1. Decision Bounddaries 1
7. Tree Models 2
7.1. Construction Rules 2
7.2. Splitting Criterion 2
7.3. Gini Impurity 2
7.4. Building a Random Forest 2
7.5. Ensembles 2
8. Neural Networks 2
8.1. Activation Functions 2
8.2. Loss Functions 2
8.3. Forward Propagation 2
9. Convolutional Neural Networks 2
9.1. Motivation 2
9.2. CNN 2
9.3. Filters 2
9.4. Number of parameters 2
9.5. Pooling 2
9.6. Behaviour of layers 2
10. Recurrent Neural Networks 2
10.1. Unsupervised NLP 2
10.2. Supervised NLP 2
11. Transformers 2
11.1. Overview 2
12. Transfer Learning 2
12.1. Model Repurposing 2
12.2. Model Fine-Tuning 2
12.3. Masked Language Modelling 2
12.4. Contrastive Learning 2
13. Clustering 2
13.1. k-Means Algorithm 2
13.2. Agglomerative Clustering 2
13.3. k-Means ≠ Agglom. Clust. 2
14. Association Rules 2
14.1. Market Basket 2
14.2. Apriori Algorithm 2
15. Recommender Systems 2
15.1. Non-Personalized 2
15.2. Personalized 2
15.3. Hybridization 2
15.4. Evaluation 2
16. How to get Fired 🔥 🤳 🚽 2

1. General topics

1.1. Fields

AI Imitates human Intelligence
ML Learns from Exerpeince
Deep Learning ML Model based on Deep Neural Net
Gen-AI Generation of content using deep-learning

1.2. Disciplines

Supervised Labeled data; prediction on unseen data
Unsupervised *unlabeled* data; uses inherent struct.
Semi-Supervised mixture of the above
Reinforcement Learning Guided by reward func

2. Data Quality

2.1. Data Quality Assessment

Resons for Low Quality Programming errors, Technical Issues, Outdated Data, Poorly extration, missing verification, Human Error, Conversion issues
DQA-Steps Identify sources, Interpret statistical figures, Visualize selected portions, Manually check ranges (for Example salaries can not be negative, not to many people should be 150 years old), Validate plausability of variable corelations (milage should be negatively corelated to a car price), Check anomalies in syntax or semantics, Explore Null-Values and duplicate records
Data Cleaning
- Identify duplicates, find plausible reasons and discard them
- Replace null values with meaningful values
- Standardize formats of dates, timezones, currencies
- Always document changes and track changes to datasets

Strategies for NULL Delete them, Fill missing values manually, Find a central tendency (mean or median of all values, mean if symetric, median for skewed data), Use measure for central tendency by class (for example if we have healthy and sick people)
Feature Engineering Extract fields from columns, date to day, mont year for example;
Vector Space Models Only contain numerical data
Dummy Variables Represent categorical data
- To be able to predict the missing value from all variables, one has to be deleted, just pick one and remove it, remove redundancy

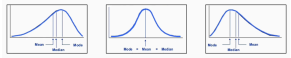
Numerical encoding of text
- Each word as row
- Each article / document as column
- Numerical word in intersection describing the relevance of the word for the article
- TF-IDF

2.2. Statistical Refresher

Central Tendency
mean average of a set of numerical observations $\mu_X = \frac{1}{n}\sum_{i=1}^{n} x_i$
mode value that occurs the most
median middlemost value of a sorted set of numerical observations

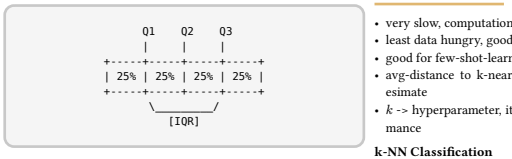
- For median, a list has to be sorted, for example with complexity of $n \cdot \log n$ when using quicksort
- Mean is sensitive to outliers, median is not

Skewness



left sqewed mean − mode < 0
right sqewed mean − mode > 0

Quartiles + IQR



Boxplot
- Represents the 5 key measures
- Outliers are 1.5 · IQR below or above 1st or 3rd Quantile
- Min value is the lowes value within 1.5 · IQR
- Max value is the highest value within 1.5 · IQR

Data Spread
variance how much values spread out in average; $\mathrm{Var}(X) = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \mu_X)^2$
- Population variance devides by n
- Sample variance devides by $n-1$

standard deviation preserves units of data; $\sigma = \sqrt{\mathrm{Var}(X)}$
covariance $\mathrm{Cov}(X,Y) = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \mu_X)(y_i - \mu_Y)$
- If two datapoints are both above or below their respective mean, the covariance is positive
- If one datapoint is above, the other one blow its mean, the covariance is negative, they show opposite tendency
- In case of independent distributions, positive and negative values cancel out in the sum, so that the covariance becomes zero

Covariance Matrix displays covariances pairwise
- Summarized all possible covariance values
- Matrix is symmetric
- Hard to interpret

2.3. Pearson Correlation

- Covariance is hard to interpret
- Divide out the std deviation to obtain *Pearson correlation*

$$p(X,Y) = \frac{\mathrm{Cov}(X,Y)}{\sigma_X \sigma_Y}$$

- Ranged between −1 and 1
- **−1** perfect anti correlation
- **0** Independent
- **1** perfect correlation
- Can be shown in a correlation matrix

3. KNN

Distance vs Similarity The closer two data points in the geometric space, the more similar they are

3.1. Euclidean Distance

- Generalization of Pythagoras for arbitarty num of dimensions
- aka L^2 norm, unbound range $[0, \infty[$

$$\mathrm{euclid}(X,Y) = \|X - Y\|_2$$
$$= \sqrt{\sum_{\{i=1\}}^{n}(x_i - y_i)^2} = \sqrt{\sum_{\{i=1\}}^{n} x_i^2 - 2x_i y_i + y_i^2}$$

3.2. Cosine Similarity

- Angle θ between vec x, y
- Dividing by L_2 norm, become points on the unit circle
- range is $[-1, 1]$
- Use abs value when negative values exist
- cosine distance = 1 − cosine similairty

$$\cos(X,Y) = \frac{< X, Y >}{\|X\|_2 \cdot \|Y\|_2} = \frac{\sum_{i=1}^{n} x_i y_i}{\sqrt{\sum_{i=1}^{n} x_i^2}\sqrt{\sum_{i=1}^{n} y_i^2}}$$

3.3. Manhattan Distance

$$\mathrm{manhattan}(X,Y) = \sum_{i=1}^{n} |x_i - y_i|$$

- Range ist $[0, \infty[$

3.4. Levenshtein / Edit Distance

- Count the min. number of changes to turn one string into the other one
  - +1 when deleting a character
  - +1 when adding a character
  - +2 when changing a character
- Efficient implementation is very difficult

3.5. Jaccard Similarity

$$\mathrm{jaccard}(X,Y) = \frac{|X \cap Y|}{|X \cup Y|}$$

- Compare the simmilarity of sets
- Range is $[0, 1]$
- # Items in both / # unique items

3.6. k-NN

- very slow, computation is deferred to classification time
- least data hungry, good for small datasets
- good for few-shot-learning
- avg-distance to k-nearest neigh can be seen as local density esimate
- k -> hyperparameter, its choice has big influence on the performance

k-NN Classification
- Data in Vector Space Model + suitable distance measure
- needs normalized data
- if $k = 1$, assign label of the nearest training sample
- if $k > 1$, majority voting among k nearest samples
- Optional: weight votes by $\frac{1}{d}$ depending on distance

k-NN Regression
- if $k = 1$, assign value of nearest training sample
- if $k > 1$, assign weighted mean of k nearest samples

3.7. Normalization

- Distance may be sensitive to Scale of Axes
- If the dimensions are not scaled, it can lead to errors

Min-Max-Normalization

$$x \mapsto \frac{x - \min x}{\max x - \min x}$$

- Transform data to interval $[0, 1]$, largest val becomes 1, smallest 0
- Can now be interpreted as %, no negative values anymore
- Unusable for supervised learning, since min and max are unknown

***Z*-Score Normalization**

$$x \mapsto \frac{x - \mu_x}{\sigma_x}$$

- σ_x: STD-Deviation, μ_x: mean
- Transforms your data to mean = 0 and std-dev = 1
- Works regardless of supervised / unsupervised learning
- Can not be interpreted directly, adopts negative values
- when data contains outliers, replace mean by median

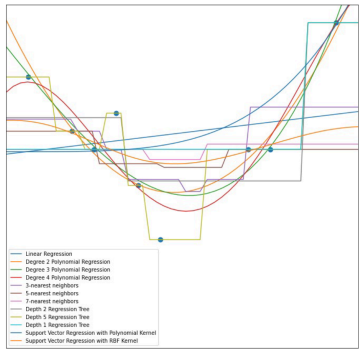
4. Supervised Learning

4.1. Measuring Regression Errors

Sum of errors $\frac{1}{m}\sum_{i=1}^{m}(y_i - f_i)$
- Does not make sense, since positive and negative errors cancel out

Mean Absolute Error (MAE) $\frac{1}{m}\sum_{i=1}^{m}|y_i - f_i|$, same scale
Mean Absolute Percentage Error (MAPE) $\frac{1}{m}\sum_{i=1}^{m}\left|\frac{y_i - f_i}{y_i}\right|$
Mean Squared Error (MSE) $\frac{1}{m}\sum_{i=1}^{m}(y_i - f_i)^2$
- f: Y-Coordinate fo the regression line

4.2. Regression Models



4.3. R-Squared

$$R^2 = \frac{\mathrm{MSE(mean)} - \mathrm{MSE(model)}}{\mathrm{MSE(mean)}} = 1 - \frac{\sum_{i=1}^{m}(y_i - f_i)^2}{\sum_{i=1}^{m}(y_i - \mu_y)^2}$$

- MSE(mean) coreresponds to data variance
- **How much of the data variance is being explained by the model**
- $R^2 = 1$, predictions perfectly fit the data
- $R^2 = 0.53$, 53% of the variance is being explained by the model
- $R^2 < 0$ also possible, is the case when the model fits the data worse than a horizonal hyperplane
- Undefined if data has variance = 0

4.4. Fitting

- Find balance between *under* and *overfitted* models

training error Errors observed when the model performs on training data
generalization error Errors observed when the model performs on unseen data

4.5. ML Workflow

Simplistic
1. Shuffle your data (unless timeseries)
2. Split into training and test set (typically 80/20)
3. Train a model on training set
4. Evaluate on unseen test data

this workflow only works if using a fixed set of hyper parameters

Model Selection
1. Split data into *train* (60%), *validation* (20%) and *test* (20%)
2. Loop over all hyperparameter combinations and models to test
3. train model with selected parameters on *training* set
4. Evaluate on *validation* set
5. Select model with best performance on *validation* set
6. Evaluate model **only once** on *test set* on unseen data for final performance estimate

4.6. Brain as Back-Channel

When one is looking at the model performance on unseen test data and after that tweaks some settings, data implicitly flows back from the test set into the training setup

4.7. K-Fold-Cross-Validation

- Can be used if there is not enough data for a 60 / 20 / 20 split
- Split data into 80% *training* and 20% *test* data
- Split set in as many parts as different models / configurations to test
- Use a different *test set* on each iteration
- Use the final set apart *test data* to do a final validation

5. Linear Regression

- We try to find relationships in data
- Used to predict values
- Can we get a house price based on the area of living?
- Supervised learning, labelled data

5.1. Terminology

X Independent Variable / Regressor
Y Dependent Variable (on X)

- Hypotheses / prediction: $h_\theta(x_j) = \theta_0 + \theta_1 x_j$
- θ_0 and θ_1 are regression parameters
- θ_0 is the y-intercept of teh fitted line
- θ_1 is the slope of the fitted line
- $\hat{Y} = \theta_0 + \theta_1 X$ is the regression line
- ε_j is the j-th residual, the error of said dot
- $y_j = \theta_0 + \theta_1 d_j + \varepsilon_j$ is the actual value of point y_j
- How to find the best values for θ_0, θ_1?
  - Pick values for them that minimize some measure of the total fitting error

5.2. Ordinary Least Squares (OLS)

- Measurement of regression errors

Mean Abs. Error (MAE) $\frac{1}{N}\sum_{j=1}^{N}|\varepsilon_j| = \frac{1}{N}\sum_{j=1}^{N}|y_i - \hat{y}_j|$
M. Abs. Percent Err. (MAPE) $\frac{1}{N}\sum_{j=1}^{N}\left|\frac{\varepsilon_j}{y_j}\right| = \frac{1}{N}\sum_{j=1}^{N}\left|\frac{y_i - \hat{y}_j}{y_j}\right|$
M. Squared Err. (MSE) $\frac{1}{N}\sum_{j=1}^{N}\varepsilon_j^2 = \frac{1}{N}\sum_{j=1}^{N}(y_i - \hat{y}_j)^2$

- MSE commonly used as cost function because it is convex (has minimum) and is differentiable
- Cost Func:

$$J(\theta_0, \theta_1) = \frac{1}{N}\sum_{j=1}^{N}\varepsilon_j^2 = \frac{1}{N}\sum_{j=1}^{N}(y_j - \hat{y}_j)^2 = \frac{1}{N}\sum_{j=1}^{N}(y_j - \theta_0 + \theta_1 x_j)^2$$

- Cost Func can be written in Matrix form:

$$J(\theta) = y^T y - 2\theta^T X^T X\theta$$

- This cost function is a convex function with a global minimum that we can obtain with Calculus

$$\theta_{\mathrm{opt}} = (X^T X)^{-1} X^T y$$

- OLS can also be calculated by Hand
1. Given this data for $x = [1, 2, 3, 4]$ and for $y = [2, 3, 5, 4]$
2. We try to fit the linear model $\hat{y} = \theta_0 + \theta_1 x$ with ols
3. Construct the matrices:

$$x = \begin{pmatrix} 1 & x_1 \\ 1 & x_2 \\ 1 & x_3 \\ 1 & x_4 \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 1 & 2 \\ 1 & 3 \\ 1 & 4 \end{pmatrix} \text{ and } y = \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{pmatrix} = \begin{pmatrix} 2 \\ 3 \\ 5 \\ 4 \end{pmatrix}$$

4. Compute $X^T X$:

$$X^T X = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 2 & 3 & 4 \end{pmatrix}\begin{pmatrix} 1 & 1 \\ 1 & 2 \\ 1 & 3 \\ 1 & 4 \end{pmatrix} = \begin{pmatrix} 4 & 10 \\ 10 & 30 \end{pmatrix}$$

5. Compute $X^T y$:

$$X^T y = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 2 & 3 & 4 \end{pmatrix}\begin{pmatrix} 2 \\ 3 \\ 5 \\ 4 \end{pmatrix} = \begin{pmatrix} 14 \\ 39 \end{pmatrix}$$

6. Compute $(X^T X)^{-1}$
   - First, get the Determinant: $\det(X^T X) = \det\begin{pmatrix} 4 & 10 \\ 10 & 30 \end{pmatrix} = 4 * 30 - 10 * 10 = 120 - 100 = 20$
   - Then get the inverse:

$$(X^T X)^{-1} = \frac{1}{\det(X^T X)}\begin{pmatrix} 30 & -10 \\ -10 & 4 \end{pmatrix} = \begin{pmatrix} \frac{30}{20} & -\frac{10}{20} \\ -\frac{10}{20} & \frac{4}{20} \end{pmatrix} = \begin{pmatrix} 1.5 & -0.5 \\ -0.5 & 0.2 \end{pmatrix}$$

7. Now compute the OLS estmator: $\hat{\theta} = (X^T X)^{-1} X^T y = \begin{pmatrix} 1.5 & -0.5 \\ -0.5 & 0.2 \end{pmatrix}\begin{pmatrix} 14 \\ 39 \end{pmatrix} = \begin{pmatrix} 1.5 \\ 0.8 \end{pmatrix}$
8. This results in the linear regression model: $\hat{y}\theta_0 + \theta_1 \cdot x = 1.5 + 0.8x$

5.3. Regression Performance (R^2)

Total variability Variability of Y around its mean

$$\overline{Y} : \frac{1}{N}\sum_j \delta_j^2 = \frac{1}{N}\sum_j \left(y_j - \overline{Y}\right)^2$$

Regression Variability Variability of Y around its regression line $\hat{Y}$:

$$\frac{1}{N}\varepsilon_j^2 = \frac{1}{N}\sum_j \left(y_j - \hat{Y}\right)^2$$

$\boldsymbol{R^2}$ is the division of these two,

$$R^2 = 1 - \frac{\sum_j \varepsilon_j^2}{\sum_j \delta_j^2}$$

, ε: Residual; δ: See total variability $\overline{Y}$

- Make sure to visualize your Residuals (parts that are left when you subtract your regression line), what you should get is some white noise dots without any pattern recognizable
- Regression provides an approximate relationship between Y and X
- Changes in X can explain changes in Y to a certain extent, but not completely
- $R^2 = 1 - \left(\frac{\text{Regression Variability}}{\text{Total Variability}}\right) = \frac{\mathrm{MSE(mean)} - \mathrm{MSE(model)}}{\mathrm{MSE(mean)}}$
- R^2 (%) of the variability of the dependent variable is explained in the regression
- $1 - R^2$ (%) of the variability remains unexplained.
- R^2 is not a perfect performance metric: Visualize your data and your residuals
- Correlation does not imply causation!

5.4. Multiple Linear Regression

- Linear Regression against more than one variable
- We now fit a hyperplane to the data in N-dimensional space
- Our Regression-Matrix gets bigger

$$y(j) = \theta_0 + \theta_1 x_1(j) + \theta_2 x_2(j) + \ldots + \theta_M x_M(j) + \varepsilon(j)$$

$$\begin{pmatrix} y(1) \\ y(2) \\ \vdots \\ y(N) \end{pmatrix} = \begin{pmatrix} 1 & x_1(1) & x_2(1) & \cdots & x_M(1) \\ 1 & x_1(2) & x_2(2) & & x_M(2) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_1(N) & x_2(N) & & x_M(N) \end{pmatrix}\begin{pmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_M \end{pmatrix} + \begin{pmatrix} \varepsilon(1) \\ \varepsilon(2) \\ \vdots \\ \varepsilon(N) \end{pmatrix}$$

Non-Linear Regression
- polynomial regression can be used
- if our hypothesis is $h(\theta, X) = \theta_0 + \theta_1 X + \theta_2 X^2 + \theta_3 X^3$, we can let $X_1 = X, X_2 = X^2, X_3 = X^3$ and then have a linear hypothesis $h(\theta, X) = \theta_0 + \theta_1 X_1 + \theta_2 X_2 + \theta_3 X_3$

5.5. Regularization

- If more features are taken into account, the risk for overfitting also gets bigger
- Regularization helps to focus on the most explanatory features
- **Ridge** and **LASSO** are implementations of Regularization
- Always start with *OLS*, then add either the sum of squares (Ridge) or the sum of absolute values (LASSO)

Ridge $J(\theta) = \frac{1}{N}\sum_{j=1}^{N}\left(y_j - \hat{y}_j\right)^2 + \lambda\sum_{k=1}^{M}\theta_k^2$
- Forces irrelevant features to be small

Lasso[1] $J(\theta) = \frac{1}{N}\sum_{j=1}^{N}\left(y_j - \hat{y}_j\right)^2 + \lambda\sum_{k=1}^{M}|\theta_k|$
- Forces irrelevant features to zero
- leads to large parameters getting penalized when choosing parameters θ to minimize $J(\theta)$
- Only important features will be driving the regression
- Less relevant ones get supressed
- λ as regularization parameter, big → high penalty

6. Classification

6.1. Decision Bounddaries

Probability / Odds Given a horse race with 6 horses 🏇
Probability $p = \frac{1}{6}$, 1 of 6 will win, ratio of something compared to everything that could happen
Odds $o = \frac{p}{1-p}$, here 1 to 5, 1 horse will winn, 5 will lose, probability of sth. happening to it not

Logit
- Logarithm of the odds
- Positively and negatively unbounded
- Logit $= \log\left(\frac{p}{1-p}\right)$
- Maps probabilities $(0, 1)$ to $(-\infty, +\infty)$
- Modeled as linear function
- Inverse link is the sigmoid $p = \frac{1}{1+e^{-s}}$

Logistic Regression
- sigmoid function is applied to the hypothesis of linear regression
- The logits are estimated by linear regression
- The sigmoid transforms the estimated logit into a probability value
- We use **cross-entropy** loss function => convex

Training of Logistic Regression
- For classification, the MSE is **not** a suitable training objective
- The MSE with a sigmoid is no more convex => local min

True- and False Positives

Accuracy $\frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{Total}}$
- how often the classifier was right

Error Rate $\frac{\mathrm{FP} + \mathrm{FN}}{\mathrm{Total}} = 1 - \mathrm{Accuracy}$
Sensitivity / **Recall** $\frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} = p(\mathrm{Classifier} = \mathrm{YES} | \mathrm{Real} = \mathrm{YES})$
Precision $\frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}} = p(\mathrm{Real} = \mathrm{YES} | \mathrm{Classifier} = \mathrm{YES})$

n=165	Predicted: NO	Predicted: YES	
Actual: NO	TN = 50	FP = 10	60
Actual: YES	FN = 5	TP = 100	105
	55	110	

F1
- harmonic mean between *precision* and *recall*
- Use whenever you have a classification problem with skewed data
- F1 $= \frac{2 * \mathrm{precision} * \mathrm{recall}}{\mathrm{precision} + \mathrm{recall}}$

Classification Metrics
- Binary classifier output probability that data point belongs to positive class
- We have to find a threshold to what probability to take
- We Plot TPR and FPR in a plot, we want high TPR and low FPR
- When plotted, we can look at the area below the ROC-Curve
  - Value between 0 and 1, AUROC = 0.5 → random guessing

[1] Least absolute shrinkage and selection operator

- Problematic with unbalanced data

7. Tree Models

Pro
- Simple to interpret
- numeric + categorical
- little data preperation
- perform well

Contra
- not the most accurate
- It's easy to create to overly-complexe trees
- very sensitive to data quality

7.1. Construction Rules

- If only positive or negative instances are left, stop branching and assign the corresponding decision as leaf node
- If some positive and some negative examples remain, choose another feature for branching or decide according to label frequency
- If no instances are left, such a combination of feature values does not occur in the training set. In this case, look at the parent node and decide according to the more frequent label
- If there are still instances but no features anymore, the training data is contradictory, noisy, non-deterministic or contains hidden features. In this case, decide according to the more frequent labe

7.2. Splitting Criterion

- No criterion separates data perfectly, both are impure
  - Choose the one with the lower Gini Impurity
- Choose feature with highes information gain

7.3. Gini Impurity

- measures the likelihood of an incorrect classification

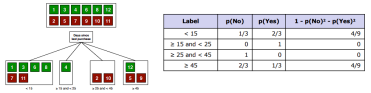
$$\text{Gini Impurity} = 1 - \sum_{i=1}^{J} p_i^2$$

- When dealing with *continuous variables*, 1. sort table acording to feature variable, 2. calculate avg of adjacent values, 3. take averages as splitting criteria

	shelf-warmer	normal	fast	#
black	$\frac{2}{6}$	—	$\frac{4}{6}$	**6**
red	$\frac{1}{4}$	$\frac{3}{4}$	—	**4**
silver	$\frac{1}{4}$	$\frac{1}{4}$	$\frac{2}{4}$	**4**
white	$\frac{2}{6}$	$\frac{2}{6}$	$\frac{2}{6}$	**6**

the table has **one row per feature** value and **one column per target** variable value

$$\frac{6}{20} * \left[1 - \left(\frac{2}{6}\right)^2 - \left(\frac{4}{6}\right)^2\right] + \frac{4}{20} * \left[1 - \left(\frac{1}{4}\right)^2 - \left(\frac{3}{4}\right)^2\right]$$
$$+\frac{4}{20} * \left[1 - \left(\frac{1}{4}\right)^2 - \left(\frac{1}{4}\right)^2 - \left(\frac{2}{4}\right)^2\right]$$
$$+\frac{6}{20} * \left[1 - \left(\frac{2}{6}\right)^2 - \left(\frac{2}{6}\right)^2 - \left(\frac{2}{6}\right)^2\right]$$
$$= \frac{8}{15} = 0.5\overline{3}$$



Label	p(No)	p(Yes)	1 - p(No)² - p(Yes)²
< 15	1/3	2/3	4/9
≥ 15 and < 25	0	1	0
≥ 25 and < 45	1	0	0
≥ 45	2/3	1/3	4/9

Label distribution:

p(< 15) = 6/12, p(≥ 15 and < 25) = 1/12, p(≥ 25 and < 45) = 2/12, p(≥ 45) = 3/12

Gini impurity for entire feature:

6/12 * 4/9 + 1/12 * 0 + 2/12 * 0 + 3/12 * 4/9 = 1/3

7.4. Building a Random Forest

Training Phase:

1. Choose a random subset D^* of your training set D
2. Choose a random subset A^* of attributes
3. Build a decision or regression tree from A^* and D^*
4. Repeat step 1 – 3 for the number of trees you want

Decision Phase:

1. Get an separate decision from each tree in the random forest
2. Combine the result of each tree to obtain the final decision

7.5. Ensembles

complex machine learning models built from simpler models techniques for building:

1. **bagging** weighted average of votes from simple models
2. **boosting** improves a simple model by correcting its errors

8. Neural Networks

single layer network **cannot** implement **non-linear** models

8.1. Activation Functions

- **Sigmoid (Logistic Activation)** $\sigma(z) = \frac{1}{1+e^{-z}}$
  - Praktisch ist der Sigmoid ab $-5 \approx 0$ und ab $5 \approx 1$
- **ReLU** $\text{ReLu}(z) = \max\{0, z\}$
  - alleviates vanishing gradient problem
- **Leaky ReLU** $\text{Leaky ReLu}(z) = \max\{0.01z, z\}$
  - alleviates vanishing gradient and dying unit problem
- **Softmax** $\text{softmax}_j(z) = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}}$

Recommendations
- **ReLU** for hidden layers
- **Sigmoid** for binary classification

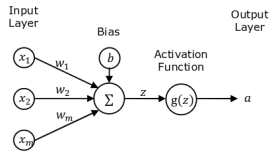
last layer
- **Softmax** for multi-class classification in last layer

8.2. Loss Functions

Choosing **lossfunction** depends on what the problem is:
- **Regression**: MSE, MAE, MAPE
- **Binary Classification**: Cross Entropy, Hinge Loss, Squared Hinge Loss
- **Multi Classification**: Multilabel Cross Entropy, Kullback Leibler Divergence

Recommendations
- MSE for linear regression
- Cross Entropy for binary classification
- Multilabel Cross Entropy for multi-class classification

8.3. Forward Propagation



$$z = b + x_1 w_1 + ... + x_m w_m$$
$$z = b + W_m$$

- Weights are matrizes with rows for each hidden node in the current layer and columns for each node in the prev. layer
- Bias Term is just a vector with biases for each node

9. Convolutional Neural Networks

9.1. Motivation

Processing of Images

150 mio weights in first layer only: Network would be huge and nearly impossible to train. Manual feature engineering problem: Performance only as good as the human selected filtes

9.2. CNN

CNN are feature extractors. Layers in CNN share parameters.

Convolution Apply many filters in parallel

9.3. Filters

Filter Dimension of target image depends on input image size, *padding* and *stride*

Stride How big steps to take when moving over

Padding How much lines of `0` to add at edges

9.4. Number of parameters

Conv2D $\left(\underbrace{\text{height} \times \text{width}}_{\text{filter size}} \times \underbrace{\text{channels or depth} + 1}_{\text{from layer before}}\right) \times \text{filters}$

Flatten Layer 0

Dense Layer $(\text{inputs} + 1) \times \text{units}$

9.5. Pooling

Pooling Neighbouring pixels tend to have similar values, removes redundancy by using min, max or avg to a pixel neighberhood; neight-size is pool size param

Pooling Layers Divide width and height by pool size, so $26 \times 26 \times 8 \rightarrow 13 \times 13 \times 8$

9.6. Behaviour of layers

Convolutional layers extract and combine features. Pooling layers zoom out to enable detection of macro features. Softmax results in a probability distribution and quantifies how sure the network is

10. Recurrent Neural Networks

10.1. Unsupervised NLP

Syntax Spelling of words is compared

- Word similarity with **Levenshtein distance**.
- 3 Ideas of Relatedness
  1. Related if appering in same document, low relatedness of synonyms
  2. Related if relating to same topic, for example in Wikipedia, each article can be thought of as topic; → **TF-IDF**
     - $\text{TF-IDF}(x, y) = \text{TF}(x, y) * \log\left(\frac{N}{\text{DF}(x)}\right)$
     - N is the number of documents
     - Term Frequency (TF) counts the number of occurrences of x in y
     - Document Frequency (DF) is the number of documents that contain x
  3. Related if in same context; use context to train a Neural Net to predict gaps

10.2. Supervised NLP

Types

Sequence to Vector Sentiment analysis, input is sequence, output is vector

Vector to Sequence Image Captioning, input fixed size, output sequence

Sequence to Sequence Text Summarization, Translation, input and output sequence

RNNs
- RNN are made for the supervised analysis of arbitrary time-series data
- Next hidden state is calculated based on prev hidden state and input
- If input length != output length → encoder-decoder architecture

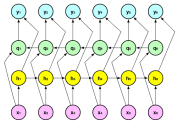
$$h_t = \varphi(W_{\text{xh}} x_t + W_{\text{hh}} h_{t-1} + b_h)$$
$$y_t = \varphi(W_{\text{hy}} h_t + b_y)$$

Vanishing Gradients
- Information that occured some time ago is lost
- *GRU* and *LSTM* help with that

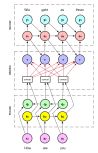
GRU Instead of h_t a *candidate* $\tilde{h}_t$ is calculated and mixed with old value

LSTM Long Short-Term Memory
- Alternative to GRU
- Additional hidden memory state
- Slower than GRU, but better performance

Bidirectional RNNs
- Hidden layer can be RNN, GRUs, LSTM, ...
- No cycles
- References in back direction via q_n



Attention
- Mimics possibility to look up information in another place
- Decoders have access to all encoder states
- Attention determine, how much info to extract from encoder state, $a^{<7,2>}$ tells how much attention the 7th output word should pay at 2nd output
- How important is each input for the next output



Parallelization
- RNNs do not parallelize (not enough)
- every layer assigned to different GPU
- Model parallelism, Data parallelism
- → Transformers: works without recurrencies

11. Transformers

11.1. Overview

Transformers
- Basis for BERT/GPT -style models
- positional and semantical encodings
- token representation is a weighted average of other token representations
- Memory requirement scales poorly to inference on long sequences
- words are projected into embedding space to capture semnatics

RNN
- suffer from exploding or vanishing gradients
- only last hidden state required in order to calculate the new hidden state
- Uses back propagation through time
- Basis for LSTM and GRU

BERT-style encoder only

GPT-style decoder only

Single-Head Attention $\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right) \cdot V$, where d_k is the number of columns in K

Multi-Head Attention

$$\text{head}_i = \text{Attention}\left(QW_i^Q, KW_i^K, VW_i^V\right)$$
$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}1, ..., \text{head}_h)W^O$$

12. Transfer Learning

- base network is trained on a base dataset and task
- repurpose the learned features or transfer them to a second target network
- process will tend to work if the features are general, meaning suitable to both base and target tasks

12.1. Model Repurposing

1. Choose a pre-trained model for object detection
2. Chop off the last layer (object localization layer)
3. Add a new object localization layer with e.g. two categories
4. Train only the last layer with the domain-specific dataset

Pros and Cons
- super efficient (train only a single layer)
- least data hungry training technique
- backbone layers are frozen
- Backpropagation of gradients is stopped at frozen layers

12.2. Model Fine-Tuning

- Early backbone layers frozen
- retrain the subsequent backbone layers that detect complex structures
- For unfrozen layers training starts off from the pre-trained weight values
- This is the transfer learning aspect in model fine-tuning
- we can unfreeze all backbone layers

Problem Data is abundant, but gathering labels does not scale

12.3. Masked Language Modelling

- huge number of unlabeled sentences → randomly mask some words
- Pre-train language model on masked work prediction
- is currently used for pre-training BERT

12.4. Contrastive Learning

- huge number of unlabeled images
- Randomly sample two tiles A and B from the same image
- Add tile B along with many tiles from other images to a batch
- Neural net must output high similarity between A and B and low similarity otherwise

13. Clustering

13.1. k-Means Algorithm

1. Choose the number of clusters
2. (randomly) throw in cluster centers
3. search nearest cluster for each data Point
   1. calculate mean of all points currently assigned to the cluster
   2. Update each cluster center to mean of assigned data points
- Choose cluster with smallest distortion

Clustering Distortion total distortion $= \sum_{i=1}^{n} ||x_i - \text{p}_{\text{ci}}||^2$

average distortion $= \frac{1}{n} \sum_{i=1}^{n} ||x_i - \text{p}_{\text{ci}}||^2$

- Optimal clustering minimizes total distortion, which is NP-hard even for k=2
- k-Means therefore only approximates the optimal solution
- k-Means always converges but not necessarily to a global minimum
- execute k-Means many times and take the result with minimal distortion

13.2. Agglomerative Clustering

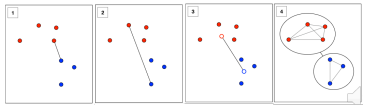
1. Consider each point a separate Cluster
2. While stop-cond not met
   - Calculate distance between all pairs of clusters
   - Merge the two closest clusters into a new one
- Configs for this are

Distance how is distance / similarity determined

Linkage Which endpoints are used for distance measuring

Stop criterion Whe nto stop, cluster density, number of clusters, ...

Endpoint Distance Measurement
1. **Simple linkage** shortest distance between any two members
2. **Complete linkage** Longest distance between any two cluster members
3. **Average linkage** Distance between cluster centroids
4. **Ward Linkage** Minimize variance inside each cluster while maximizing variance between clusters



13.3. k-Means ≠ Agglom. Clust.

- In k-means, two data points can meet in a cluster and separate again in the next step
- In agglomerative clustering the data points would stay together forever

k-Means
1. number of clusters is the only stop condition
2. scales to large volumes of data
3. can produce different clustering depending on the initialization of centers
4. can be hard to interpret

Agglomerative Clustering
1. can terminate based on cluster density, distance, etc.
2. can only be used for smaller amounts of data
3. is completely deterministic
4. protocol each step in a dendrogram

14. Association Rules

An association rule is an implication $X \rightarrow Y$, where X and Y are disjoint item sets

14.1. Market Basket

Support
- measures how frequently items are bought together
- A good association rule has a high support
- **Support = Interestingness**

$$\text{support}(\{i_1, ..., i_n\}) = \frac{\#\text{purchases of}\{i_1, ..., i_n\}}{\#\text{transactions}}$$
$$\text{support}(X \rightarrow Y) = \text{support}(Y \cup X)$$
$$\text{support}(X \rightarrow Y) = \text{support}(Y \rightarrow X)$$

Confidence
- measures how frequently items in Y appear in transactions that contain X
- A good association rule has high confidence
- **Confidence = Trustworthiness**

$$\text{confidence}(X \rightarrow Y) = \frac{\text{support}(Y \cup X)}{\text{support}(X)}$$

Lift
- how many times more often X and Y show up together than expected if they were statistically independent
- the larger the lift value, the stronger the association between X and Y
- A good association rule has a high lift
- Lift = Association Strength
- **lift = 1** X and Y are statistically independent
- **lift < 1** X and Y appear less often together than expected (anti-correlated)
- **lift > 1** X and Y appear more often together than expected (correlated)

$$\text{lift}(X \rightarrow Y) = \frac{\text{support}(X \cup Y)}{\text{support}(X) \cdot \text{support}(Y)}$$

14.2. Apriori Algorithm

- An association rule is good if its support and confidence are above the thresholds
- Apriori algorithm finds good association rules in two steps

(item sets in lattice of subsets)

1. Generate frequent item sets satisfying the support threshold
   - If an item set is frequent all subsets must be frequent too
   - If an item set is infrequent all super-sets must be infrequent too (efficient pruning)
2. Extract rules from frequent item sets satisfying the confidence threshold
   - Let $X \cup Y$ be a frequent item set found in step 1
   - When a rule $[X \rightarrow Y]$ violates the confidence threshold, then any rule $[X - \{c\} \rightarrow Y \cup \{c\}]$ violates the confidence threshold as well, with c being any item

15. Recommender Systems

15.1. Non-Personalized

- all users get **same recommendations**

Pros
- No new user cold start problem
  - Content-Based recommenders not even item cold start problem
- Computationally, pre-processing possible

Scoring, Ranking
1. Collect rating from e.g. {0, ..., 5}
2. Calculate mean and round off
3. Instead of ratings use e.g., sales figures

Scoring **lacks Context** → compute associations

Associations
- Easy, fast and cheap
- Associations are context aware
- Associations can be tailored to specific businesses
- Non-Personalized Recommendations

Content-Based Recommenders homogeneous platform into vector space model and measure similarity

15.2. Personalized

implicit or explicit user data

Content-Based Recommendations

$$P_{\text{uj}} = \frac{\sum_{i \in N_j} \text{sim}(j, i) * r_{\text{ui}}}{\sum_{i \in N_j} \text{sim}(j, i)}$$

P_{uj} How much does user u like item j

$\sum_{i \in N_j}$ Neighbors of item j

$\text{sim}(j, i)$ How similar is item j to item i

r_{ui} Rating that user u gave to item i

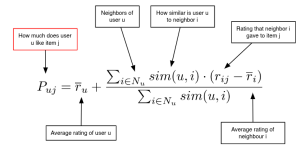
- Neighbors N_j refer to the (at most) N rated items most similar to item j, N being a hyperparameter
- Select only rated items with similarity > 0 (choose less than N if you cannot find enough candidates)

Pros
- User Independence & Transparency
- No New Item Cold Start Problem

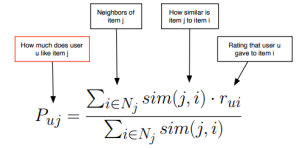
Cons
- Limited Content Analysis
- Overspecialization
- Severe New User Cold Start Problem

Collaborative Filtering

User-to-user



Item-to-Item



main difference is that we compare row vectors for neighbor search instead of column vectors

Pros
- Cross-Category Recommendations

Cons
- Many people must participate
- New User Problem aka. Cold Start Problem
- New Item Problem aka. Cold Start Problem
- Users with unusual taste –> no accurate recommendations

15.3. Hybridization

- Combine algorithms so that advantages of one compensates disadvantages
- Mitigate cold start problem by combining content-based and collaborative recommenders

common hybridization strategies
- Weighted Approach
- Mixed Approach
- Cascade Approach
- Switching Approach

15.4. Evaluation

- `precision@k` for offline evaluation on validation and test set
- Among the k recommended items how many are relevant (purchased, viewed, ...)
  - semmnatic problems (bought in different shops, not came across this product, ...)
  - Recommenders low scores
- Afterwards, online evaluation by A-B testing

16. How to get Fired 🔥 🤳 🗜️

- Hyperparameter is choosen based on test set
- Parameters for a scaler are extracted from all data
- Clustering with unnormalized data
- Usage of accuracy with skewed data
- Split inside loops makes evaluation incomparable
- Test data have their own normalization parameters
- Target value used in input
- Evaluation on training data
- Hyperparameter tuning on test set
- Cross-validation without re-initializing model for each fold
- Randomzied train/test split for time-series data wrong
- Leakage of data from test set into the training set