# **AI Foundations | AIFo**

Zusammenfassung

## Natural Language processing (NLP), embeddings

### The 4 Ingredients of Machine-Learning

* Data: Dataset including pre-processing
* Cost-Function (Loss): A formal expression for “good” and “bad” (MSE)
* Model: From linear model to a million-parameter Neural Network.
* Optimization Procedure: An algorithm that changes the parameters of the model such that the cost-function is minimized. (SGD)

### How to represent words in a computer

The meaning of words can be represented using vectors.

* + 1. One-hot representation

A one-hot vector is a vector with a single 1-value and all others set to 0. Count the number of different words. Then define one unique one-hot-vector per word.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **It** | rains | **.** | We | go | home | **.** | **It** | stops | raining | **.** |
|  |  |  |  |  |  |  |  |  |  |  |

* + - 1. Disadvantages
* High dimensional vector space: In an article with 100’000 Words, each word is represented by an 100’000-dimensional space.
* Sparse representation: The vectors are memory-inefficient, and algorithms cannot efficiently learn from such representations.
* No generalization: with one-hot vectors, all words are completely unrelated to each other. Therefore, we cannot generalize knowledge. One-hot representation does **not capture any aspect of the meaning** of a word.
  + 1. Indexing

List of all the words (optionally alphabetically sorted), the index represents each word.

|  |  |  |
| --- | --- | --- |
| cat | 0 | “The cat sat on the mat.”  This is the “dense equivalent” of one-hot encoding. Indexes are not more useful than one-hot vectors. But indexing is (often) used as a preprocessing step. These indices are then fed into a network which learns more useful representations. |
| mat | 1 |
| on | 2 |
| sat | 3 |
| the | 4 |
| . | 5 |

* + 1. Distributed representation

A word can be “defined” by context. Words with similar semantics share some context (e.g., rat and cat are both animals). Distributed representations can be learned. Example below: Similar traits are coloured similarly / at the same positions. We can subtract the properties of “Man” from “King” and add “Woman” to get the word “Queen”.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| King |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Man |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Woman |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Queen |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

Word to Vector

A mathematical function maps a word or the corresponding index (input) to a high dimensional vector (output). Machine-learning algorithms can learn this function. In neural networks this function is implemented in a embedding layer.

Ein Bild, das Text, Schrift, Reihe, Screenshot enthält.

Automatisch generierte BeschreibungAdvantages of vectors

Ideally, a “good” embedding maps similar/related words to similar regions of the vector-space. That is, nearby words have a semantic similarity. Once we have “good” vectors, we can do math. The dot-product (Skalarprodukt) is a measure of similarity. Vectors also can be added/subtracted.

For specific tasks, it is sufficient to have a notion of similarity between words. The question “how to understand words” becomes “how to calculate similarity between words”.

Basic properties of the dot product:

* The dot product between two vectors is maximal when they are both the same (going in the same direction). If the two vectors both have norm / length 1, then the maximal value is 1.
* The dot product between two vectors is zero iff (if and only if) they share no components (are perpendicular / 90° difference / orthogonality)
* Ein Bild, das Reihe, Diagramm enthält.

  Automatisch generierte BeschreibungThe dot product between two vectors is minimal (negative) when they are anti-parallel (pointing in opposite directions). If the two vectors both have norm 1, then this minimal value is -1.

Cosine-similarity or cosine-distance

and are vectors. Say represents “cat”, represents “dog”. The cosine-distance is a way to calculate how similar two words (vectors) are.

Calculation of cosine similarity

The **dot product** can be used to multiply two vectors of equal size. The result is a real number.

The **length / magnitude / norm** of a vector is calculated by squaring all the values of the vector, adding them together and then take the square root of the sum. The result is a real number.

**Cosine-similarity:**

The resulting similarity ranges from meaning exactly opposite, to meaning exactly the same vector/word, with indicating orthogonality (perpendicular/ 90°).  
There are two approaches to training a general language model: Download and use a predefined language model or the usage and optimization of an Embedding Layer. A known architecture for training embeddings is known under the term word2vec.

### Large Language Models (LLMs)

Large Language Models dominate the current success stories. LLMs are large Artificial Neural Networks that are used for translation, chat, Q&A, programming etc.  
Current LLMs are Transformer based. Transformers are computational units with a particular structure and a trainable “Attention mechanism”. Not further covered in AiFo  
Generative Language models produce sequences by calculating a probability distribution over the next word given the past text. They sample one word (or token) and repeat the process.

## Ein Bild, das Reihe, Diagramm, Schrift, Text enthält. Automatisch generierte BeschreibungStatistics

### Random Variables

A random variable is a variable that takes a numerical value which depends on a random experiment. This is a way to represent outcomes of a random experiment as numbers.

* Discrete: takes any of a finite set of values, e.g.
* Ein Bild, das Schrift, Reihe, Diagramm, Zahl enthält.

  Automatisch generierte BeschreibungContinuous: takes any value of an uncountable range, e.g. the real numbers in the interval .
  + 1. Probability Mass Function (Wahrscheinlichkeitsfunktion)

Function that provides the probability for each value of a discrete random variable . The tables below are PMFs, and the graph at the right-hand side.

* + 1. Example: Rolling a single dice

The discrete random variable is the number observed when rolling a fair dice. The possible values and the probabilities they take are:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Value of the random variable | 1 | 2 | 3 | 4 | 5 | 6 |
| can also be written as , or  (Probability that the random variable takes the value |  |  |  |  |  |  |

* + 1. Example 2: Sum of two dice

The discrete random variable is the sum of eyes of two dice. (Options, only 2 dice combinations can result in a 3 (1/2 and 2/1), while for a 9, there are 4 combinations (6/3, 5/4, 4/5 and 3/6)

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Value | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
|  |  |  |  |  |  |  |  |  |  |  |  |

Probability that the sum is between 4 and 7 (sum of probabilities):

In Example 2, we have two random variables. is the number of eyes on dice #1 and is the number of eyes of dice #2.

### Joint Probabilities

The joint properties of two random variables are defined by the Joint Probabilities Mass Function.

* + 1. Joint Probabilities with independent random variables

For independent random variables, the joint probability is the product of the individual probabilities. This is also true with more than two independent random variables.

**Example:** The first dice does not affect the probability of the second dice, so the probability for die #1 showing 5 and die #2 showing 4 is still :

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

* + 1. Joint Probabilities with dependent random variables

If the events are not independent, the variables are dependent or correlated. Example:

* : The event to observe clouds ( no clouds, small clouds, big clouds)
* : The event to observe that it rains ( no rain, light rain, moderate rain, heavy rain)

Given there are small clouds, what is the probability for ***moderate rain***? This value cannot be read directly from the table, because all the probabilities in the full table together are , in this case however we only look at given that .

* + 1. Marginal Probability

The probability of an event occurring, irrespective of the outcome of another random variable. For example, the probability of for all outcomes of . If the two variables are visible in a table, then the marginal probability of one variable would be the sum of probabilities for the other variable on the margin of the table. This is often used to “normalize” the values across a “row” or “column”.

In other words, the probability of regardless of is the sum of all probabilities of where appears.

Written as  *or*

Example: The probability of rain, regardless of cloud size

* + 1. Conditional Probability ( when given )

The probability that something will happen in relation to knowledge we already have about another correlating event. Joint probability and conditional probability are related in the following way:

Example: Probability of moderate rain (what we want to know) given small clouds (what we know)

### Two-step experiments

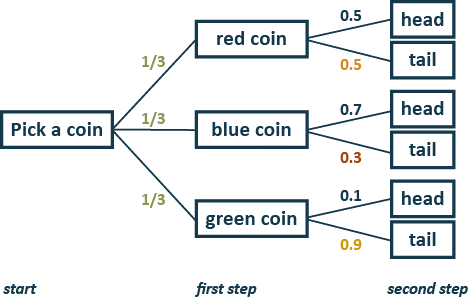
Example: Consider a box with three different coins, a red, a blue and a green one. The red coin is fair. The others have different probabilities for head/tail.

* Red coin :
* Blue coin :
* Green coin :

We now do a two-step experiment:

* Step 1: Pick a random coin from the box.
* Step 2: toss the coin and observe the outcome.

What is the probability to observe “tail”?

* + 1. Tree Diagram

Tree diagrams are a probabilistic model that explains how data is generated. They are also a structured visualization of the experiment.

To calculate the probabilities, you just need to multiply along the path.



The probability of observing tail is therefore:

To generalize: The first step can be written as and the second as . So, the full calculation can be written as

The same can be shown in a table of joint probabilities.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | red coin | blue coin | green coin | marginal |
| head |  |  |  |  |
| tail |  |  |  |  |
| marginal |  |  |  |  |

### Bayes Rule

Bayes theorem lets us “invert” the direction of the tree model. From observed outcomes, we can make probabilistic statements about the “hidden causes” of the observation. The main goal is to update the probability of an event based on prior knowledge. H = Hypothesis, E = Evidence

Example 1: If we observe tail, which coin was drawn in step 1?  
We can formulate this question in terms of probabilities.

**Prior:** , **Likelihood:** for red, for blue, for green

The result is a posterior distribution. It is the result of updating the prior distribution with the evidence / data / observation.

Example 2: If we flip the same coin 3 times and observe tail, head, head. Which coin was drawn?  
For this example, we need to repeatedly apply Bayes rule. We use the first observation to calculate the first Posterior Distribution. The Posterior then becomes the new Prior in the second application. To calculate the second posterior Distribution (getting tail & head), we also need to adjust the normalizer with the first posterior.

The first application is the same as in Example 1.

*Second application*

**Prior:** for red, for blue, for green, **Likelihood:** for red, for blue, for green



*Third application*

**Prior:** for red, for blue, for green, **Likelihood:** for red, for blue, for green



This means, it is most likely that the blue coin was drawn (52%)

## Linear Regression

Linear Models are the simplest model to explain a relationship between “Input” and “Output”. Standard method to find an optimal linear model.  
**Interpretation:** Understand if some input has an effect on the output.   
Example: Is there a relationship between smoking cigarettes and the risk of lung cancer?  
**Prediction:** Given a new , use the model to predict / estimate the .   
Example: x is smoking rate, y is death rate

Linear regression belongs to supervised learning: The algorithm learns a linear relationship between and , both are given.

### Model

A model is a mathematical function that “explains the data”.

is “unexplained noise”. It is assumed that follows a normal distribution (Bell Curve/Glockenkurve). The function can be simple or very complicated. The goal of ML is to find the model which explains the data (as good as possible). Also, instead of approximating , we calculate an estimate of the usually unknown .

In linear regression, we only consider a linear relationship between the input and output. There are therefore only two free parameters, and . The goal is to identify and for which the linear model “best explains the data”. is usually called slope, the intercept.

Ein Bild, das Reihe, Diagramm, Steigung, Quittung enthält.

Automatisch generierte BeschreibungHow to find out if a model is good or bad?

### Mean Squared Error (MSE)

This is the loss we want to minimize.

( = the residual Difference between our estimate and the actual value)

Ein Bild, das Zeichnung, Entwurf, Muster enthält.

Automatisch generierte Beschreibung (Error) is the Sum of the areas of the residual Squares (red) divided by two times the number of squares.

* + 1. Pearson Correlation Coefficient ()

Most common way of measuring a linear correlation. It is a number between and that measures the strength and direction of the relationship between two variables.

|  |  |  |
| --- | --- | --- |
|  | Correlation type | Interpretation |
|  | **Positive** Correlation | Both variables change in the **same direction**. Positive Steigung |
|  | **No** correlation | There is **no linear relationship** between the variables. |
|  | **Negative** Correlation | The 2. variable changes in the **opposite direction.** Negative Steigung |

### Multiple Linear Regression

Add more “explaining factors” to the model, since most of the time, multiple factors contribute to the result to different degrees:   
Same concept, but different notation / indexing:

Idea: a single “dependent” variable is explained by multiple independent variables . To be able to change the importance of each variable, we also add a weight . Example: is an observed/measured quantity. Example: blood pressure. are “factors” like age, weight, sex, … . are weights. How much does each factor explain the outcome ?

A variant of multiple linear regression is polynomial linear regression, where each variable is an exponent

* + 1. Matrix Notation

Dataset: points where is a vector with features (=dimensions).  
The model: can be written much more compactly if we use matrix notation.  
 where = Datapoints, = Weights, = Estimates:

## Gradient Descent

Gradient Descent is a fundamental optimization algorithm. When an AI is “training” or “learning”, this means that an algorithm is performing some sort of optimization, like minimizing the loss function. It only works if we can express the loss function as a differentiable function, this is not always the case.

The gradient of a function is the collection of all its partial derivatives organized in a vector.

,

Example:

It will always point in the direction where there is the greatest increase in the function, in our case the loss. Since we want to minimize our loss, we need to invert our gradient (descending the gradient). Gradient Descent is an iterative operation, so we run GD again until the result converges (no/very small change). We can also set a convergence threshold, if the last move is smaller than this, we also end the GD.

The gradient descent follows these steps:

1. Ein Bild, das Text, Schrift, weiß, Diagramm enthält.

   Automatisch generierte BeschreibungPick a random point in the function, this is the starting point. This point is represented by the row vector
2. While the gradient hasn’t converged (iterative part of the algorithm)
   1. Compute the negative gradient at to all other data points and pick the one with the greatest descent.
   2. Move the location by the result of 2a.
3. Repeat until you have found the minimum or reached the convergence threshold. If, compared to the previous iteration, the new gradient of point has not changed more than the convergence threshold, the algorithm has converged.

Learning rate

The learning rate alpha is the size of the step Gradient Descent takes all the way until it reaches a minimum, and it directly impacts the performance of the algorithm. When it’s too big, you’re taking big steps, so you may step over the minimum and never reach it. When the learning rate is too small, the algorithm might take a long time to find the minimum.

Limitations of Gradient Descent

* Calculating derivatives for the entire dataset is time consuming.
* Memory required is proportional to the size of the dataset.

### Stochastic Gradient Descent (SGD)

It is a probabilistic approximation of Gradient descent. It is an approximation because, at each step, the algorithm calculates the gradient for one data point picked at random, instead of calculating the gradient for the entire dataset. This represents a significant performance improvement. But because the gradient is not computed for the entire dataset, and only for one random point on each iteration, the updates have a higher variance. This makes the cost function fluctuate more on each iteration, making it harder for the algorithm to converge.

* + 1. Batch-Gradient-Descent

Often, batch-gradient-descent is used which uses random subsets (or batches) instead of one random point. This is more efficient. Typical batch sizes: 32, 64, …, 1024 samples.

* + 1. **Annealed Stochastic Gradient Descent**

The learning rate gets adapted. Starts the algorithm with a large learning rate and then reduces it over time for example by multiplying it at each iteration by a decay\_factor like . Typically, there’s a lower bound where the learning rate doesn’t decrease any further.

## Regularization

Too complex models generalize badly. Too simple models may miss information and perform sub-optimally. Those two observations are related by the bias-variance trade-off (aka bias-variance dilemma). With regularization, we can constrain the learning process.

### Model Testing

The model must perform well on new, unseen inputs which means it must generalize well to new data.

In-sample Error (aka Training Error): Difference between the predictions and the actual results on the same data the model was trained on. It is possible to find a model which perfectly fits the data. When a model has an MSE of 0, the data was probably overfitted. Overfitted models perform great on the training data, but badly on new data. So ideally, the training error should be minimal, but not zero.  
  
Out-of-Sample Error (aka Test Error): Difference between the predictions and the actual results on different data the model wasn’t trained on. Difference between the predictions and the actual results on different data the model wasn’t trained on. If we use our model on new data sample to test how well it predicts for new data, we can calculate the error of the prediction using . This is the out-of-sample error.

The goal is to learn a model from data that generalizes well to new data. A “good” model has a low generalization error. So both errors should be as low as possible.

* + 1. Splitting Technique

We can’t calculate the generalization error, because we do not have “new data” to test our model. We can only estimate it by splitting the data we are given into two sets: The training and test set. A common split ratio is , so we have most of the data in the training set, while still having enough data to test with. The data in the test-set does not get used during fitting.

* Training Phase: Fit the model to the training set. This minimizes the in-sample error.
* Evaluation: Evaluate the model using the test-set. This gives us an estimate of the generalization error.
  + 1. Ein Bild, das Kreis, Screenshot enthält.

       Automatisch generierte BeschreibungBias-Variance Trade-Off

By analyzing the prediction error mathematically, one can decompose it into two terms: bias (average difference between predicted and actual values) and variance (difference between different runs of a model). The expression “high bias” is used in the sense of “a too simple model for the given data”

* High Variance: Not Precise Estimates are spread out
* Low Variance: Precise Estimates are clustered together
* High Bias: Not Accurate, high training error Model missed relevant relations between features and outputs
* Low Bias: Accurate Estimates are close to the correct result

Too Simple Model: High Bias, Low Variance (underfitting)

A very simple model (i.e. simple linear regression) cannot do better than finding a line, no matter how much data we give it to learn. Such a model imposes a high bias. It fails to learn the underlying structure of the data. By imposing a high bias, we assume a “simple world” and reduce the space of what can be learned.

Ein Bild, das Text, Reihe, Diagramm, parallel enthält.

Automatisch generierte BeschreibungThe flip-side of a too simple model is that it is relatively stable. With another test-sample, the model would find a very similar line. So for a change in data, we would fit almost the same model. This is the meaning of low variance.

Too Complex Model: Low Bias, High Variance (overfitting)

Low bias: A more complex model is less restrictive. It can better “explain” the data. A high variance means that for a different set of data points, the model could be very different.

The trade-off

Higher bias implies lower variance, lower bias implies higher variance. In practice, we do not directly care about the bias, we just want a low variance (reliable predictions). But we can only build a model as complex as the data permits. We therefore have to find an optimal balance between bias and variance.

### Regularization

With regularization, we reduce the number of polynomial degree to avoid overfitting or increase the degree to avoid underfitting. It does this by decreasing variance at the cost of increasing bias. This in turn decreases the training accuracy, but increases generalizability. Regularization adds a Constraint to the model, rather, its Optimizer, to achieve this.

* Measure of performance: regression error (MSE) how well the model predicts data
* Measure of complexity: regularization term control the complexity of the model

We want to Minimize the regression error + regularization term. It is common to have two separate functions: An optimization function for the optimizer (Gradient Descent) and a performance evaluation function to evaluate the error.

* + 1. How to express Model Complexity

The complexity of a model can be expressed by multiple parameters: degree of polynomial, number of features and size of coefficients. There are two different ways to express the complexity:

* L2-Norm (Euclidean Norm, Sum of weights):
* L1-Norm (Manhattan distance / Taxicab norm, Sum of absolute weights, «Häuschen zählen»):

We add one of these constraints to the optimizer to find the best weights for our model.

* + 1. Ridge

Ridge uses the L2-Norm (Euclidean Norm). Minimize: **MSE** + **Hyperparameter L2-Norm**

Example Calculation of L2-Norm: Point 1 is at and Point 2 is at . So, Lambda gets multiplied by .

L2 regularization is not robust to outliers. The squared terms will blow up the differences in the error of the outliers. The regularization would then attempt to fix this by penalizing the weights.

* + 1. Lasso (least absolute shrinkage and selection operator)

Lasso uses the L1-Norm (Manhattan distance). Minimize:

**MSE** + **Hyperparameter** \* **L1-Norm**

Lasso can force the weights to 0, unlike Ridge. It enables us to perform feature selection, making certain weights 0. means that is not relevant.

When we have highly correlated features i.e. number of rooms and house area size, the L1 norm would select only 1 of the features from the group of correlated features in an arbitrary nature, which is something that we might not want.

Example Calculation of L1-Norm: Point 1 is at and Point 2 is at .

Ein Bild, das Kreis, Diagramm, Screenshot, Farbigkeit enthält.

Automatisch generierte BeschreibungTo test if you have too many features, you can use lasso regression to see if it eliminates any features.

Lambda ()

is a hyperparameter, it does not belong to the optimization process as such. It is varied to find the best fit. When it is zero, the is just the normal MSE. As gets larger, we are enforcing the weights to be smaller by constraining the squared sum of weights more and more. Increasing makes the model simpler, increases bias and reduces variance.

## Cross Validation

**Hyperparameter**: Specifies details of the learning process such as parameters of the optimizer(learning rate, type of gradient descent, regularization parameters (L1, L2, Lambda) etc.)

Ein Bild, das Screenshot, Reihe, Farbigkeit, Rechteck enthält.

Automatisch generierte Beschreibung**3-way holdout:** Split data in training-data, validation-data and test-data.

* Train the model with the training-data, multiple models with different hyperparameters
* Validate the trained models with the ***validation-data***. Take the hyperparameters with the best score.
* Train another model with the best hyperparameters and the training + ***validation data***. Test this model with the ***test-data***.
* Optional: Train the best model again with all the data.

Problems

* Training error is too optimistic about generalization (overfitting).
* Test error is unbiased but can be too pessimistic. The generalization error is calculated only on certain 20% set, which could lead to the model only being optimized for those 20% of the data.
* Test and training data may not be representative of the general/overall dataset.

### K-fold Cross validation

Ein Bild, das Text, Screenshot, Schrift, Zahl enthält.

Automatisch generierte BeschreibungCross validation is a technique to address these problems. It is an extension of the holdout method. With the k-fold cross-validation, the data is split into k folds. Then the train/validation process is repeated k-times. Each fold participates in k-1 training phases and is used once for validation. The folds can be overlapping.

We can use cross-validation to obtain a better estimate of the generalization error. This is also known as model evaluation. After k-fold cv, we can train the model on the complete data using the fixed hyperparameters and deploy that model. If , the Model is split into training data and test data. If # of data in the dataset, only one value is used for testing on every split (LOOCV – Leave one out cross validation). Typical values for are or . It is better to apply the preprocessing pipe-line (e.g. standardization) to each split, not only once in the beginning for the whole dataset. Otherwise, the results may be distorted. We can also find the best hyperparameter by running KFCV for each set of hyperparameters and pick the set with the smallest mean validation error.

## Feature Scaling

Ein Bild, das Text, Screenshot, Zahl, Schrift enthält.

Automatisch generierte BeschreibungFeature scaling is a method to normalize the range of independent variables of data. If for example, you have multiple independent variables like age, salary and height with ranges 18-100 years, 25’000 – 75’000 and 1-2 meters, feature scaling transforms them all to be in the same range. If the range differences are too big, small changes in the weights of large features have a huge impact on the MSE, while weights of small features need huge changes to affect the MSE.

Regularization penalizes larger coefficients more than the smaller ones. Standardization puts all the features on equal footing.

### Sklearn standardscaler

Rescales a dataset to have a mean of 0 and a standard deviation of 1.

,

***Example:*** Data Points : . . Sample variance,   
Sample Std. Deviations . Standardization of

## Classification and Logistic Regression

* Binary Classification: Only two classes true/false. Example: Epileptic seizure or healthy state?
* Multi-Class Classification: More than two classes. Example: Match is won, it’s a tie, Match is lost

### Logistic Regression

Used for binary classification (Yes/No, Spam/no spam). Linear regression is not usable for binary classification, because it is linear. Even with a threshold, the function does not work well with only two outputs, it outputs continuous values. If we input outlier data, the whole model gets distorted. The MSE does not work. So we need a probabilistic function like the sigmoid. We want a cost function over the probability that the data point belongs to a certain class.

* + 1. Multi class logistic regression

One vs rest

Train a single classifier for each class with the samples of it as positive and all other as negative (Sunny vs not sunny, cloudy vs not cloudy). Applied to an unseen sample, the classifier with the highest is the class.

One vs one

Train classifiers to distinguish between each pair of classes (sunny, cloudy), (sunny, rainy), (sunny, snowy)… Applied to an unseen sample, combine all results to produce the final classification.

* + 1. Ein Bild, das Text, Screenshot, Diagramm, Reihe enthält.

       Automatisch generierte BeschreibungSigmoid function

This is where our features (data x) enters the calculation. The ’s are unknown. That’s what needs to be learned.

Why sigmoid? Because of the odds ratio how many times more likely x will be accepted vs. against x will be rejected:

We can write the estimated probability as (formula for exercises)

* + 1. Maximum Likelihood (Cost function of logistic regression)

Given all the data points , we want to maximize the probability that all the predictions are correct. The objective of training is to set the coefficients so that (the prediction) is close to when the actual data and close to when . This can be calculated using gradient descent. The cost of is small on wrong and large on correct predictions.

## Ein Bild, das Text, Screenshot, Schrift, Zahl enthält. Automatisch generierte BeschreibungClassifier Evaluation

How to calculate accuracy and error from the confusion matrix?

* Accuracy: How often is the classifier correct:
* Error: How often is the classifier wrong:

These metrics can be misleading, as a model that always predicts that a patient is healthy can still have a very high accuracy, even if all sick patients have been misclassified. Example: 10/1000 patients are sick, everyone is classified as healthy, accuracy is still 99%

It depends on the objective if a False Positive or a False Negative is worse. (sickness: false negative, Spam: false positive)

### Recall/Sensitivity (True Positive Rate)

Useful when false negatives are worse. Among the positive ground truth samples (), how many did we correctly classify? If you have no false negatives because you have no negatives, you can fool recall.

### Precision

Useful when false positives are worse. Among the predicted positives (), how many were correctly classified?

### F-Score

Combining precision and recall. In real life, false negatives and false positives are bad. So we need the harmonic mean of Precision and Recall . If both and are high, the score is high. If one of them is low, the score is also low.

There is also a combined metric . acts as a dial to decide the emphasis between precision and recall. (Both equally important), (Recall not important), (Precision not important).

### Threshold

There is no universal solution, different goals require different thresholds.

1. Train your machine learning model
2. Use the trained model to make predictions on your test set, so that each example has a classification probability between 0 and 1.
3. Using a variety of threshold values, convert the predicted probabilities to predicted classes. Calculate True positive rate () and False positive rate ( ). Different thresholds result in different TPR and FPR.
4. Ein Bild, das Text, Reihe, Diagramm, Schrift enthält.

   Automatisch generierte BeschreibungPlot a curve of TPR vs FPR for the different thresholds.
   * 1. Receiver Operating Characteristics (ROC)

A ROC space is defined by FPR and TPR as and axes, respectively, which depicts relative trade-offs between true positive and false positive.

Area under the curve (AUC) shows how well the TPR and FPR is looking in the aggregate. The greater the area under the curve, the greater the quality of the model. If the AUC is < 0.5, the model is useless, because it is worse than just randomly assigning classes.

## K-Nearest-Neighbours (KNN)

If a simple line perfectly separates the classes, then the classes are said to be linearly separable. However, what to do when the classes are not linearly separable? Logistic regression is possible, but very inconvenient. That’s where KNN comes into play. Basic idea: “A datapoint is known by the company it keeps”.

Ein Bild, das Kreis, Screenshot, Diagramm, Grafiken enthält.

Automatisch generierte BeschreibungGiven a test data point, KNN computes nearest neighbours of it and returns the most frequent class of the neighbours.

Example in image:

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
| **Sample 1** | ***green*** | ***green*** | ***green*** |
| **Sample 2** | ***blue*** | ***blue*** | ***blue*** |
| **Sample 3** | ***red*** | ***green*** | ***red or green*** |

### KNN Details

* Load the training and test data
* Chose value of (the number of nearest neighbours to consider for classification) Should be tuned based on validation error. results in overfitting, a increase in leads to smoother boundaries, results in underfitting
* Chose a distance metric (Euclidean, Manhattan, cosine, Minkowski, …)
* For each test data points :
  + For all training data , calculate the distance with your distance metric
  + Sort training data in the ascending order of the distance
  + Choose the first data points from the sorted training data
  + Choos the most frequently occurring class from the data points as the classification result.

Advantages of KNN: Easy and simple machine learning model. Few hyperparameters to tune k & distance metric.  
Disadvantages: k should be wisely selected, Large computation cost during runtime if dataset is large. Not efficient for high dimensional datasets, proper scaling should be provided.

### Distance MetricS

Given and . Example: , (hyperparameter used for minkowski distance, normally 1 or 2)

Cosine distance  
,

Manhattan distance  
,

Euclidean distance  
,

Minkowski distance:  
,

## Naïve Bayes Classifier

Naïve Bayes is a generative method for classification (it generates something) based on Bayes’ Theorem (See page 4).). It assumes that all the features that predict the target value are independent. It describes the probability of an event based on a prior knowledge of conditions.

Naïve Bayes is good when the dataset is small and there is no training phase. It is used extensively when data contains categorical features, but it’s not used much in numerical features. However, we can use binning to create categories.

|  |  |  |
| --- | --- | --- |
| nr. | email header | spam |
| 1 | ***Hurry*** Sale Tomorrow | 1 |
| 2 | Rain tomorrow | 0 |
| 3 | Sale price tomorrow | 1 |
| 4 | Tomorrow workshop rain | 0 |
| 5 | ***Hurry*** sale | ? |

Assume we have a bunch of emails we want to classify as spam or not spam. How to calculate

We can simply take each word as a separate feature.  
All the words: ***Hurry***, ***Sale***, ***Tomorrow***, ***Rain***, ***Price***, ***Workshop***

So “Hurry Sale Tomorrow” can be encoded as 1 = spam, 0 = not spam:

Now when an email contains “hurry”, would it be classified as spam or ham? We can find out by calculating the probability that the email is spam given that it contains “hurry”.

( means spam)



Final calculation: email no. 5, “hurry sale” will be classified as spam.

## unsupervised learning - Clustering

When we are given data without labels classifier of the data, can we still learn something from the data? Yes. Often, the data has some structure. The goal of unsupervised learning is to self-discover patterns from the data without any training.

A simple example of a structure in the data are clusters. i.e., the data points which have some shared properties will group together into a cluster.

### Clustering

The goal of clustering is to group data points into number of clusters. How do we do that? Similar principle as KNN

* + 1. Naïve K-means

1. Let us assume we know the number of clusters we want to group the data in.
2. Initialize the value of cluster centres (aka means, centroids) .Usually randomly initialized.
3. Find the squared Euclidean distance between the centres and all the data points. Assign each data point to the cluster of the nearest center.
4. Each cluster now potentially has a new centre (mean). Update the centre for each cluster. The new center is the average of all the data points in the cluster.
5. If some stopping criterion met, done (like centres do not change anymore, the distance of datapoints to the centre is bigger than a set threshold or a fixed number of iterations has been reached). Else, go to step 3.
   * 1. Evaluate Cluster quality

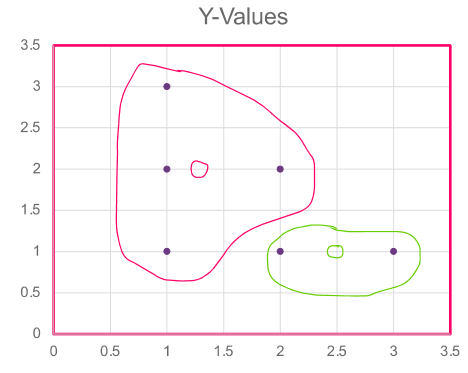
The number of clusters is a hyperparameter. You need at least two clusters, but less than the amount of data points. How can one evaluate the cluster quality?

Goal of good clustering: Create clusters so that for each cluster the distance of each cluster member from its center is minimized. There are two approaches to find the optimum.

**Inertia or within-cluster sum-of-squares (WCSS)**

Sum of squared distances of samples to their closest cluster centre (How far away the points within a cluster are). A small inertia is desired.

Ein Bild, das Reihe, Diagramm, Steigung enthält.

Automatisch generierte BeschreibungExample:

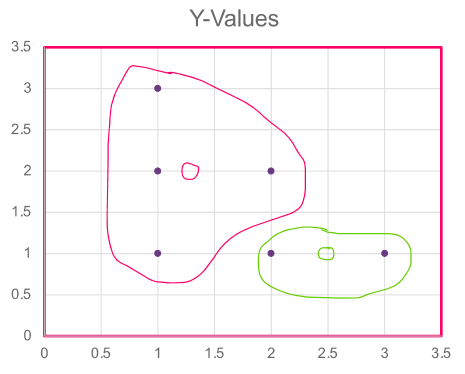
|  |  |  |
| --- | --- | --- |
|  | Squared Euclidean distance from red centre | Squared Euclidean distance from green centre |
|  |  |  |
|  |  | - |
|  |  | - |
|  |  | - |
| 2,1 | - |  |
|  |  | - |
|  | - |  |
| **WCSS** | wide cluster | tight cluster |

Do this calculation as you increase the amount of clusters. Then draw a plot vs inertia. The more clusters we have, the closer the points are to their centers.

The optimal amount of clusters is found at the “elbow” of the graph. In this example 3 or 4

**Silhouette Score**

The silhouette score considers both the cohesion(how far apart the points in a cluster are) and separation (how far apart the clusters are). It provides a value between and for each data point, with higher values indicating better-defined clusters. The overall Silhouette Score for a clustering solution is the average of these individual scores:

Example: calculation of value **(average of all distances)**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Distance from other points in the cluster | | | | | |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
| Distance from other points in the cluster | | | | | |
|  |  | |  | |  |
|  |  | |  | |  |
|  |  | |  | |  |

Calculation of value **(average of all distances)**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Distance from points in the RED cluster | | | |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

Ein Bild, das Text, Diagramm, Reihe, Screenshot enthält.

Automatisch generierte BeschreibungSo the Silhouette Score of .

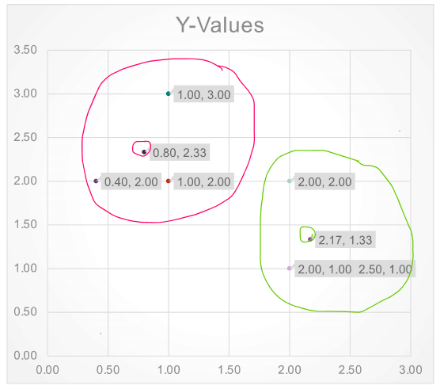
We can now decide on an amount of clusters in which the WCSS and the Silhouette Score look good, in this example around 3-5.

* + 1. Performance

The performance depends on the random initialization of the seeds for the centroids. Some seeds result in a poor convergence rate, some can converge to suboptimal clustering. If the initial centers are very close together, it would take a lot of iterations for the algorithm to converge. The best way is to initialize randomly and run multiple times. If the clusters are stable, the clustering is optimal.

Features with large values may dominate the distance value. Features with small values will have no impact in the clustering. That’s why you should always employ feature scaling (normalize values).

* + 1. Example calculating cluster centre

* Center of red cluster: ,
* Center of green cluster: ,

Will a new point be assigned to the red or the green cluster?   
Squared Euclidean distance from red cluster:

Squared Euclidean distance from green cluster:

Now that a new point is added to the red cluster, the center needs to be recalculated.

## Ensemble

The combining of multiple weak models and the aggregation of their results is called ensemble learning. Aggregating results of many weak predictors for a better prediction. Techniques: Voting, Bagging, Boosting.

Ensemble works best, when:

* The weak models are better than random.
* The models are independent from one another and make uncorrelated errors.
* There is a sufficient number of weak learners.
* The models are not trained on the same data otherwise likely to make the same error.

Different learners use different Algorithms (KNN, Logistic Regression), Different Hyperparameters and different training data (Cross validation, feature engineering, feature selection).

### Hard Voting

Pick the class with the most votes. There are 5 classifiers to check if an email is spam or ham. For a particular data, the prediction of the classifiers are [spam, spam, ham, ham, spam]. The final prediction of the ensemble is spam, because 3 of the 5 models voted for spam.

* + 1. Hard voting with weights

There are 3 classifiers to predict class spam (1) and ham (0). The predictions from these classifiers have weights defined as [0.1,0.3, 0.6]. For one email, the predictions are [spam, spam, ham]. For spam, we calculate the sum of weights from all classes: . For ham, we calculate the same but with ***ham***: . The final prediction of the ensemble is ham, because the ***weighted sum of ham was bigger than the sum of spam***.

### Soft Voting

Predict the class with the highest class probability, averaged over all classifiers. Only possible if predictions are probabilities. Example: There are 3 classifiers “”. For a prediction, the classifiers return the following probabilities for each class.   
. The average for each class (“”) is the following:  
, , . Class 3 has the highest class probability and wins.

* + 1. Soft voting with weights

There are 3 classifiers and a 3-class classification problem where we assign equal weights to all classifiers. The weighted average probabilities for a sample would then be calculated as follows:

|  |  |  |  |
| --- | --- | --- | --- |
| Classifier | Class 1 | Class 2 | Class 3 |
| Classifier 1 |  |  |  |
| Classifier 2 |  |  |  |
| Classifier 3 |  |  |  |

In this example, the predicted class label is 2 since it has the highest average probability.

### Bagging (Bootstrap Aggregating)

Bagging methods form a class of algorithms which build several models on random subsets called Bootstraps of the original training set and then aggregate their individual predictions to form a final prediction. There are two ways of bagging: Sampling with replacement (Putting the cookie back in the bowl after taking it out) is called Bagging, sampling without replacement (eating the cookie after taking it out of the bowl) is called pasting.

Only bagging allows data points to be used several times for the same predictor. The individual models have a relatively low bias and high variance. Bagging (reuse of data) reduces the variance. This provides a way to reduce overfitting. Bagging works best with strong and complex models.

Random Subspaces: Samples are drawn as random subsets of the features. i.e. for the housing data train one model only with area and nr. of bedrooms and one with area and age of the house  
Random Patches: Samples are drawn as random subsets of both samples and features.

* + 1. Out of Bag (oob) Evaluation

If the data points are random, it is possible that some data points never get chosen. Those points are called oob-points and can be used as test-data. Since the models are only trained on a subset of data (bootstrap or “in-the-bag” data), the other data can be used as test data, the out-of-bag samples. We test every model with its oob samples and then take the majority vote / highest average probability for each oob sample to determine the accuracy with the oob Error (similar to cross validation).

### Boosting

Boosting is an ensemble method to train predictors sequentially. Each predictor tries to correct its predecessor. It tries to reduce the bias of all the combined estimators – the training error reduces.

* + 1. Ein Bild, das Text, Screenshot, Diagramm enthält.

       Automatisch generierte BeschreibungAdaBoost (Adaptive Boosting)

AdaBoost assigns equal weights to each training sample in the beginning. Then it trains a model to fit the given data. After that, it increases the weight of the misclassified samples so they will make up a larger part of the next classifier training set, so the next classifier will perform better on them. When we stop training, we compute the weight of each predictor with logs odds ratio: , means accuracy, higher is better. To use even the bad classifiers, the results of classifiers are flipped (“do the opposite”).

## Ein Bild, das weiß, Reihe enthält. Automatisch generierte BeschreibungArtificial Neural Networks

Example: **2**-dimensional input, first hidden layer with neurons, second hidden layer with neurons, output layer with **1** neuron and without activation function. Trainable params:

Ein Bild, das Reihe, Diagramm, Steigung, Design enthält.

Automatisch generierte BeschreibungReLU Activation Function: