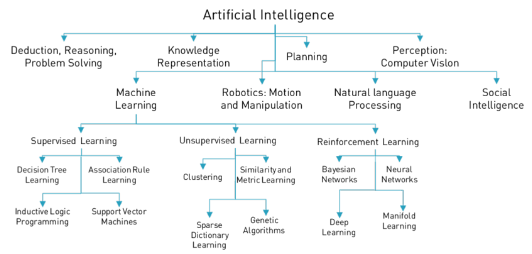


1 Problems ML can solve

1.1 Supervised learning

- Decision making processes (supervised learning)
- It's called supervised learning because a "teacher" provides the input and output

1.1.1 Examples

- Reading the handwritten postalcode from an envelope
- Detecting credit card fraud

1.2 Unsupervided learning

- only the input is given, not the output

1.2.1 Examples

- Identifying topic in a set of blog posts
- Segmenting customer in groups with similar preferences

1.3 Nomenclature

- rows → samples or data point
- columns → features

1.4 Knowing your Task and Data

- you need to understand what your data is and which algorithm work best for it

1.4.1 Keep in mind

- What question am I trying to answer? Can my Data answer it?
- What is the best way to phrase my question as a ML problem?
- Have I collected enough data to represent the problem I want to solve?
- How will I measure success in my application
- How will it interact with other parts of my research or business product?

2 First Example

- Classifying iris flowers → classification Problem
- The three different species of flowers are different **classes**
- for a specific data point/ sample, it's called a **label**

2.1 k-Nearest Neighbors

- We can choose k Neighbors that are closest to the new point and then choose the majority amongst those.

2.2 Interface of a supervised model

- **fit-methode**
- **predict-methode**
- **score-methode**

3 Supervised Learning

- when we have input/ ouput pairs
- requires human effort to build the training sets

3.1 Classification and Regression

- two major types of supervise learning

3.1.1 Classification

- Goals is to predict class labels from a predefined list of possibilities

binary Classification:

- When there are exactly two possibilities e.g. yes/no

multi classification:

- More than two possibilities

3.1.2 Regression

- The goal is to predict a continuous number.
- predicting someone's salary with their age, job, education and where they live
- If it doesn't matter if the value is a small margin of like $39'990 when we expected $40'000 it's a regression task.

3.1.3 Generalization, Overfitting and Underfitting

Generalization:

- When we build a Model on the training data that can accurately predict unseen data → it's able to **generalize**.
- When we over or undefit generalization is bad.

Overfitting:

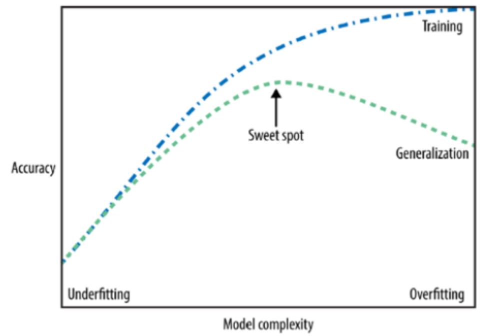
- Is when you fit the model to close to the particularities of the training set.

Underfitting:

- Is when you make the model to simple

Sweet Spot:

- Is between under- and overfitting. This is the Model we want to make



3.2 Relation of Model Complexity to Dataset Size

- The more variety of data you have, the more complex your model can be without overfitting.
- **Never underestimate the power of more data**

3.3 Supervised Machine Learning Algorithms

3.3.1 KNN

- A bigger k makes the prediction smoother
- But pay attention to not Overfitting
- KNN is not often used in reality
- KNN doesn't work well with **sparse datasets** (a lot of 0 in the table)
- It doesn't work well dataset that have a lot (hundreds of features)

3.4 Linear Regression

- $\hat{y}$ = prediction
- w = weights
- b = offset
- $w[0] \cdot x[0]$ = one feature

3.4.1 Pros

- linear regression is powerful for multiple features
- good if you have more features than samples

3.4.2 convention

- For training data always _ like coef_, intercept_

3.4.3 Coefficient of determination/ fitting

- A way R^2 to measure how good the model predicts.
- If R^2 for the training and test data are close together, we are likely underfitting.
- If there are a lot of features, there is a high possibility of overfitting.

3.5 Ridge regression

- basic principle is the same as in linear regression
- one addition is that we want the *weights* w to be as small as possible.
- it restricts coefficients to be **close to zero**

3.5.1 Regularization

- this addition (constraint) is the first example for *regularization*-
- regularization means restricting a model to avoid overfitting.

3.5.2 conclusion

- ridge regression is a trade-off the training-score is lower while the test-score is higher.
- for small datasets, ridge is far superior to linear regression
- the smaller the dataset, the more important is regularization

3.6 Lasso

- like ridge it restricts coefficients to be close
- some coefficients are *exactly zero* → some features are ignored
- Having *zeros* can make a model easier and reveal the important features.

3.6.1 conclusion

- When you have to choose between **Ridge** and **Lasso**, choose Ridge only if you have a lot of features and expect only a few of them to matter or you want to simplify choose Lasso.

3.7 ElasticNet

- **Combines Ridge and Lasso**
- **In Practice use this!!!**

3.8 Linear models for classification

3.8.1 Binary classification

- $\hat{y} = w[0] \cdot x[0] + w[1] \cdot x[1] + ... + w[p] \cdot x[p] + b > 0$
- If the function is smaller than zero, we predict the class –1 if it's bigger than zero, we predict the class +1
- It separates two class with a line, plan or hyperplane.

3.8.2 Logistic regression (classification!)

- Is used to classify in a model.
- For data with fewer features it may look weak, but the more features there are the stronger it gets.
- Higer **C** → lower regularization → more overfitting.
- Multiclass classification possible

3.8.3 Liner models for multiclass classification

- Most linear regression models are for binary classification.
- It's possible to use a binary classification model for multiclass classification with the **one-vs.-rest** approach

3.8.3.1 How it works

- Multiple binary models comparing one class to all the others.
- Then to find the class every binary model is run. The one with the highest score on its single class is chosen.

classification confidence formula:

- $w[0] \cdot x[0] + w[1] \cdot x[1] + ... + w[p] \cdot x[p] + b$
- The one with the highest $w[0] \cdot x[0] + w[1] \cdot x[1] + ... + w[p] \cdot x[p] + b$ is chosen.

3.8.3.2 Linear SVC classifier

- A way to make multiclass classification.

3.8.4 Strengths, weaknesses and parameters

- Main parameter of linerar models is the **regularization** parameter either called **alpha** in regression models or **C** in the classification models.
- alpha ↑ or C ↓ means a simple model
- Usually alpha and C are searched for on a log-scale.
- The default is **L2-regularization**
- Only **a few** of your features are important or the **inpretability** of your model is important→ **L1-regularization**.
- **Fast** to train and predict
- They scale well for **large** datasets
- They work well with **spare** datasets
- **Very BIG** datasets use SGDClassifier, SGDRegressor

3.9 Naive Bayer Classifiers

- Similar to the linear models but **faster**.
- **Worse Generalization** than LogisticRegression or LinearSVC

3.9.1 How it works

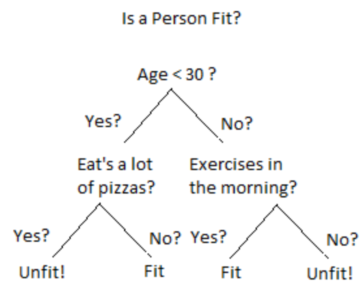
- The average and sometimes the standard deviation of each feature is calculated.
- The class where the statistics match the best is chosen.

3.10 Decision Trees

- Widely used models for classification and regression
- terminal nodes **(leafs)** at the end

3.10.1 How it works

- It's asking yes/no question like is *feature i larger than value a*.



- One Test splits the data along a line parallel to an axis.
- Multiple test build a decision tree.
- If the data points of a **leaf** share all the same target value it's calles ***pure***.
- The leafs are scanned till the right *box* is found. Then the majority is chosen as the class.

3.10.2 Controlling complexity

How to stop overfitting:

- There are to common ways, either pre-pruning or post-pruning

pre-pruning:

- Stopping the creation of the tree early

post-pruning:

- Building a complex tree and then removing or collapsing nodes with little information.
- The top split is the most important.

3.10.3 Summarizing a tree

feature importance:

- To understand which features are most important to predict the class.
- Sum is always 1
- If the **feature importance** is low, that doesn't mean the feature is uninformative, it just means it wasn't picked.
- In contrast to the linear models, feature importance are always positive and don't encode which class a feature is indicative. It just shows which are most important to come to the conclusion.

3.10.4 Short commings

- Decision trees **are not able to predict** data that is outside the training data range.

3.10.5 Strengths, weaknesses and parameters

Parameters:

- pre-pruning with max_depth, max_leaf_nodes or min samples_leaf

Strengths:

- Easy to visualize and understand
- The algorithms are invariant to scaling of data.
- Good if features are on completely different scales.

Weaknesses:

- They tend to **overfit.**
- That's why we use ensemble of decision tress etc.

3.11 Ensembles of decision trees

- **Ensembles** are methods that combine multiple ML models to create more powerful.

3.11.1 Random forest

- A way to reduce the overfitting of a normal decision tree.

3.11.1.1 How it works

- collection of decision trees (each is a little bit different)
- Each tree overfits in different ways, by averaging the results overfitting is reduced.
- Random forests get their name from injecting **randomness** into the tree build. So that each tree is different.

3.11.1.2 How to build random forests

- Decide how many trees you want to build.
- Create **bootstrap sample** by creating a data set with the same number of samples, but some are missing and some will be repeated.
- Creating the tree but not choosing the best test of all features, just the best of a random subset.
- For a high *max_features* the trees will be very similar, for a low one they will be very different.
- For **regression**, each tree makes a prediction, then the average is taken to make the decision.
- For **classification**, a *soft voting* strategy is used. Each algorithm gives a probability for each possible output. The highest average is chosen.

3.11.1.3 Strengths, weaknesses and parameters

Strengths:

- Among the **most used** ML methods.
- Precise

Weaknesses:

- high dimensional and sparse data eg. (text data)
- Needs more memory and time than linear models

Parameters:

- n_estimators, max_features, max_depth
- max_features = np.sqrt(n_features)

3.11.2 Gradient boosted regression trees (gradient boosting machines)

- Ensemble method
- Regression and classification is possible

3.11.2.1 How it works

- **First use random forest** if time is of essence or you really need to be accurate use gradient boosting.
- It builds trees in a serial manner, each tree tries to correct the mistake.
- Strong **pre-pruning** → shallow trees (1-5).
- The idea is to combine many **weak learners** (simple models)

3.11.2.2 Packages

- xgboost

3.11.2.3 Strengths, weaknesses and parameters

Strengths:
- Among most **powerful** supervised learning models

Weaknesses:
- Require careful parameter tuning
- high dimensional and sparse data eg. (text data)

Parameters:
- n_estimator, learning_rate
- learning_rate $\downarrow \rightarrow \downarrow$ correction
- A higher n_estimator is unlike with the random forest not always better it may lead to overfitting.