Keywords

- *predictive model*: the goal is to find a model that performs well on new / unseen data
- *new / unseen data*: data which the model did not seen during the training
- *(low) generalization error*: if the predicitve model has low generalization error it performs well on new / unseen data
- *test-error / out-of-sample error*: test-error is an estimate of the generalization error (splitting data in training- and test set - calculate error on test set). can be splited in to bias / variance.
- *bias*: simple models -¿ high bias (low variance). There is a risk of undertiffnt
- *variance*: complex models -¿ small bias (high variance). There is a risk of overfitting
- *underfitti*:ng
- *overfitti*:ng
- *regularization*: is a method to control the model complexity. Using regularization we can find the optimal balance in the bias-varance trade-off
- *bias-variance trade-of*:f
- *L1 / L2 norm*: Penalty for model complexity

Machine Learning Types

- Unsupervised learning
- Supervised learning
- Reinforcement learning

NLP

Natural Language Processing is a sub-field of artificial intelligence. It aims to understand and generate human (natural) language. NLP is often used in Chats Bots.

Dialogflow

- Intent matching: Dialogflow Intents: Know what your users want
- Dialogflow Entities: Dialogflow Entities: Identify things your users mention
- Dialog Control: Dialogflow Dialog Control: Shape the flow of your conversation

The 4 Ingredients of Machine Learning

1. *Data*:
   - Sometimes you have to do some clean up or some normalization
   - we can not do better than what is in this data
2. *Cost-Function (loss)*:
   - a formal (mathematical) expression for "good" / "bad"
   - Mean Squared Error is commonly used
3. *Model*:
   - Something to map input to the output
   - two parameter linear / complicated neural network
4. *Optimization Procedure*:
   - An algorithm that changes the parameters of the model such that the cost function is minimized.

+3 Steps for better ML:

1. Performance optimization: Building efficient pipe-lines is difficult
2. Visualization and evaluation of the learning Process: Learning curves, Performance Measures
3. Cross-Validation & Regularization: goal to develop models that generalize

One-Hot Representation

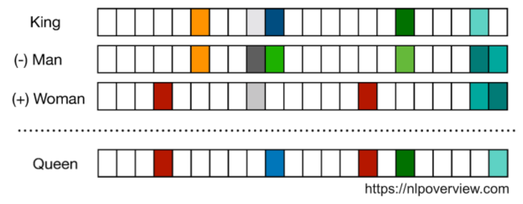
One-Hot is possible way to represent words as vectors. The number of diffrent words is the dimension of the vector. Each word has now only set one possition to 1.
Problems with this approach are:

- the vectors are very hight dimensional
- is not memory efficient, a single 1 and for example 99'9999 zeros
- no generalization possible. This representation can not store the meaning of a word

Index Representation

Index is a *dense equivalent* to One-Hot Representation. Each word is asigned an index. Indexing is often used as preprocessing step.

Distributed Representation



Word Embedding

A Word embedding is something like a function / black box which maps a word to a vector .

Cosine Similarity

The cosine similarity is just the formula for the Dot Product transformed:

$$\cos(\phi) = \frac{A \cdot B}{|A||B|}$$

Coseine distance:

$$\begin{aligned} \text{cosine distance} &= 1 - \text{cosine similarity} && (1)\\ &= 1 - \frac{A \cdot B}{|A||B|} && (2) \end{aligned}$$

Probability
The marginal probability is the general known probabiliy.
The Joint Probaility $p(x, y)$ means that the event x and the event y occurs. If x and y are independent the formula is:

$$p(x, y) = p(x) \cdot p(y)$$

If x and y are **not** independent then the formula is:

$$p(x, y) = p(x|y) \cdot p(y)$$

Conditional Probability

$$P(Y|X) = \frac{P(Y \cap X)}{P(Y)}$$

Bayes Rule

$$P(Y|X) = \frac{P(X|Y) \cdot P(Y)}{P(X)}$$

Regression
Linear Regression is a Supervised learning algorithm.

MSE

$$\begin{aligned} E &= \frac{1}{2N} \sum_{i=1}^{N} e_i^2 && (3)\\ &= \frac{1}{2N} \sum_{i=1}^{N} (y_i - (a \cdot x_i + b))^2 && (4) \end{aligned}$$

Residual
The Residual e is the difference between the original y_i and the estimated $\hat{y}_i$.
Bias
The inability for a machine learning methods (e.g. Linear Regression) to capture the true relationship is called bias.
Variance
The difference in the fits between training and testing data.
Gradient Descent
Gradient Descent is an algorithm used for optimization problems. It works with an iterative approach using gradients. On each iteration the error function is derived.
SGD
The Stochastic Gradient Descent (SGD) is a variant of Gradient Descent. Instead of calculating the Update Rule (The Gradient Descent Update Rule) for all available data only one random chosen data is used. SGD is therefore a Gradient Descent with Batch Size 1 (The batch-size in SGD). Since you only use a subset, it is faster, but you might not find the global minimum.
Batch Size
The subset of the data which are used for Gradient Descent (Gradient Descent) is called **batch-size**. Increasing the batch-size will reduce the variance of the gradient estimation. There's no "best" batch-size, but often mini-batches of size $n = 32$ or $n = 64$ are used.

- *n = 1*: The Stochastic Gradient Descent
- *1 ¡ n ¡ N*: mini-batches (an approx. gradient is used)
- *n = N*: the real gradient is used

Update Rule

$$\begin{bmatrix} a \\ b \end{bmatrix}_{t+1} = \begin{bmatrix} a \\ b \end{bmatrix}_{t} - \alpha \begin{bmatrix} \frac{\partial E}{\partial a} \\ \frac{\partial E}{\partial b} \end{bmatrix} \Bigg|_{\begin{bmatrix} a \\ b \end{bmatrix}_{t}} \qquad (5)$$

Derived MSE

$$\begin{aligned} \text{E} &= \frac{1}{2N} \sum_{i=1}^{N} (y_i - (a \cdot x_i + b))^2 && (6)\\ \frac{\partial \text{E}}{\partial a} &= \frac{1}{N} \sum_{i=1}^{N} (y_i - (a \cdot x_i + b))(-x_i) && (7)\\ &\approx \frac{1}{n} \sum_{j \in J_n} (y_j - (a \cdot x_j + b))(-x_j) && (8)\\ \frac{\partial \text{E}}{\partial b} &= \frac{1}{N} \sum_{i=1}^{N} (y_i - (a \cdot x_i + b))(-1) && (9)\\ &\approx \frac{1}{n} \sum_{j \in J_n} (y_j - (a \cdot x_j + b))(-1) && (10)\\ & && (11) \end{aligned}$$

Annealed Learning
In the Update Rule is the learning rate α specified. At the beginning the learning rate α is high and over time the value is reduced.
In-sample error

The *in-sample error*, also called *training error*, is error which your model makes when predicting the sample data. It is possible to find a model which fits perfectly the data.

out-of-sample error

When you try to predict with a model *unseen* data the resulting error is called *out-of-sample error*. An other name is *generalization error* or *test error*.

Bias-Variance Trade-off

There is a trade-off: higher bias implies lower variance, lower bias implies higher variance. Normally we want reduce the variance (no surprise with out-of-sample data)

Regularization

Regularization is used to punish single parameters of the model to prevent overfitting. Normally the L1 and L2 Norms are used to measure the model complexity:

1. L1 - Norm $\sum_i^p |\beta_i|$
2. L2 - Norm $\sum_i^p |\beta_i^2|$

$$\sum_{i=1}^{n}(y_i - \sum_{j=1}^{p} x_{ij}\beta_j)^2 + \lambda \sum_{j=1}^{p} \beta_i^2 \tag{12}$$

Cross-Validation

To train your model you normally split your data in testing and training data. If you create this split only once you can not verify how well your model performs on new data. To solve this problem you can split your data in k times in k splits. k-fold Cross-Validation is an implementation of Cross-Validation

Hyper Parameters

By "parameter" or "model parameter" we usually mean the weights of a model. Parameters of the training procedure, (e.g. regularization parameter λ) are called hyper parameters.

Neurons

Artifical Nuerons are the building blocks for an ANN. An Neuron receives an input vector ($[x_1\ x_2\ \ldots]$). Each input has a different *weight* ($[w_1\ \ w_2\ \ \ldots]$) and bias b (=intercept). The neuron iteself:

1. calculates the sum of the weighted input ($\vec{x} \cdot \vec{w}$)
2. adds a bias b
3. pass it through a nonlinear activation function (e.g. ReLU)

ANN

An ANN (Artifical Neural Network) consists of many Artifical Neurons. Basically the ANN is just a data-structure to define arbitrarily complex mathematical functions. The ANN has 3 types of layer:

- one input layer
- one or many input layers
- one output layer

Train ANN

1. initial each weight randomly
2. on optimizer reduces a cost-function (e.g. MSE)
    - for each weight the partial derived is calculated
    - Backpropagation is used to optimized the weigts (trains the ANN, used in Keras)

Classification

Using classification you try to classify given data (X).

sigmoid

$$sigmoid(y) = \frac{1}{1+e^{-y}} \tag{13}$$

$$y = w_1x_1 + w_2x_2 + \cdots + w_nx_n \tag{14}$$

$$sigmoid(y) = \frac{1}{1+e^{-y}} \tag{15}$$

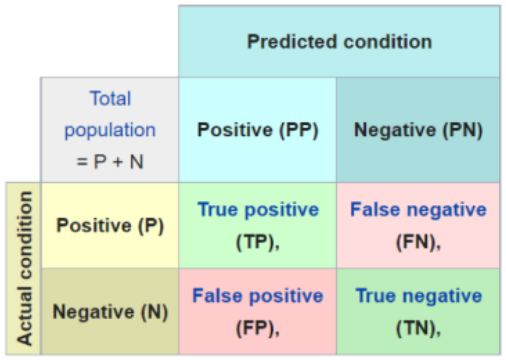
Train classification

You have to minimize the cost by adjusting the weights. This is called **Maximum Likelihood**. We want to minimize the equation 16. This could be done with Gradient Descent.

$$\frac{-1}{N}\sum_{i=1}^{N}(y_i \cdot \log(p_i) + (1-y_i) \cdot \log(1-p_i)) \tag{16}$$

The loss depends on prediction and reality:

- if I say it is very likely that it happens and it happens → small loss
- if I say it is very likely that it happens and but it does NOT happens → big loss



		Predicted condition	
	Total population = P + N	Positive (PP)	Negative (PN)
Actual condition	Positive (P)	True positive (TP),	False negative (FN),
	Negative (N)	False positive (FP),	True negative (TN),

Mean Accuracy:
How often is the classifier correct?

$$Accuracy = \frac{t_p + t_n}{n} \tag{17}$$

Mean Error:
How often is the classifier wrong?

$$Error = \frac{f_p + f_n}{n} \tag{18}$$

Precision:
When the prediction is 1 how often is it correct?

$$Precision = \frac{t_p}{t_p + f_p} \tag{19}$$

Sensitivity, Recall, True Positive Rate (TPR):
How often the prediction is 1 in comparison it should be 1.

$$Recall = \frac{t_p}{t_p + f_n} \tag{20}$$

Miss Rage, False Negative Rage (FNR)

$$MissRate = 1 - TPR \tag{21}$$

False Positive Rate (FPR):

$$FPR = \frac{f_p}{f_p + t_n} \tag{22}$$

Precision vs. Recall
Application 1 - classify videos safe for kids (p = safe):

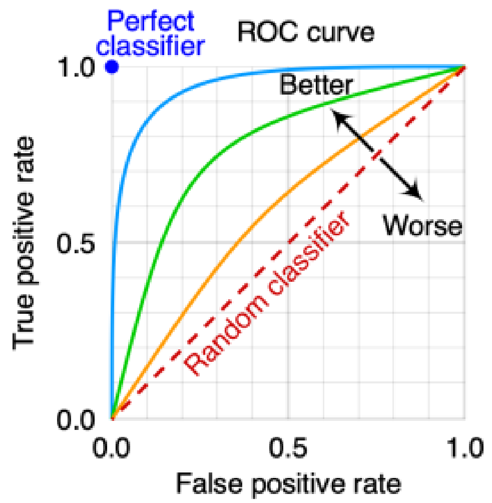
- Low recall (f_n is big) $\rightarrow$ many safe videos are rejected.
  High precision (f_p is low) $\rightarrow$ keeps only safe videos.
- High recall (f_n is low) $\rightarrow$ no safe videos are rejected.
  Low precision (f_p is big) $\rightarrow$ many unsafe videos are kept.

Application 2 - classify shoplifters from a camera:

- low precision but high recall $\rightarrow$ false negatives are OK

ROC
To determine an optimal threshold the ROC is a good way to find it. The FPR (x-axes) and TPR (y-axes) (How to determine the quality of a classifier?) are used to generate a coordinate system. The best classifier would be at the point (0/1). In general:

> the greater the area under the curve (AUC), the higher the ration of true positive to false positives



KNN
KNN is a classification algorithm. KNN calculates the k nearest neighbors of it and returns the most frequent class of the k neighbors.
KNN Algorithm

1. load the training data & test data
2. choose the value of k
3. For each test data points x_{test}
   (a) For all training data $\mathrm{x}_{\text{train}}$, calculate the $d(x_{test}, x_{train})$
      - distance metric: euclidean, Manhattan, cosine, …
   (b) Sort training data in the ascending order of the distance
   (c) Choose the first k data points from the sorted training data
   (d) Choose the most frequently occurring class from the k data points as the classification result.

KNN can be adjusted with the numbers of neighbors (k) and the distance metric.

Multiclass classification with linear regression

One-vs.-rest Train a single classifier for each class c. The samples of class c as positive and all other samples as negative. Apply all classifiers to an unseen sample x. The classifier reports the highest p is the class.

One-vs.-one Train classifiers to distinguish between each pair of classes. Apply all classifiers to an unseen sample x. Combine the result to produce final classification.

Unsupervised learning

Goal: identify hidden patterns among the data. A simple example of a structure in the data are clusters: i.e the data points which have some shared properties will fall into one cluster or one alike group. — Marko Lehmann

Clustering algorithm

1. Pick k random data points (class).
2. For each of the remaining data points calculate the similarity to the k clusters.
3. The data point is assigned to the most similar class.

Naive k-means clustering

k-means is a clustering algorithm.

1. Assume the number of clusters k_c
2. Initialize the value of the k clusters centers / mean (C_1, C_2, ..., C_{k_c})
3. Assignment:
   (a) Find the *squared euclidean distance* between the centers and all the data points.
   (b) Assign each data point to the cluster of *the nearest center*
4. Update: Each cluster now potentially has a new center.
   (a) New Centres (C'_1, C'_2, ..., C'_3) = Average of all the data points in the cluster.
5. If some stopping criterions met, Done
6. Else, go to Assignment / Step 3

Initialization:

- Run the algorithm multiple times with randomly chosen initial centers $\rightarrow$ are the clusters stable?

Possible Stopping Criterions:

- When the centers don't changed (time-consuming)
- The distance of data points from their centers ¡= threshold
- Fixed number of iterations

Good Solution

In ML you execute the same algorithm multiple times with different staring parameters. With this approach you get a better view about the whole situation. Additional you can be sure that your parameters result in a good model.

Scale value

It is important that all data are in the same range. If not the extremes (min / max) may dominate the algorithm.

Cluster quality

WCSS - Within-cluster sum-of-squre: For each cluster you calculate 23 and try to minimize it.

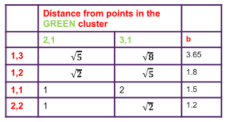
$$\sum_{i=1}^{n} d(C_k, x)^2 \tag{23}$$

Silhouette Score

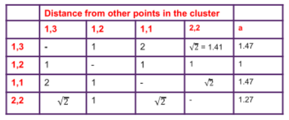
- a: average intra-cluster distance
- b: average inter-cluster distance

$$\text{Silhouette Score of a point} = \frac{b-a}{\max(a,b)} \tag{24}$$

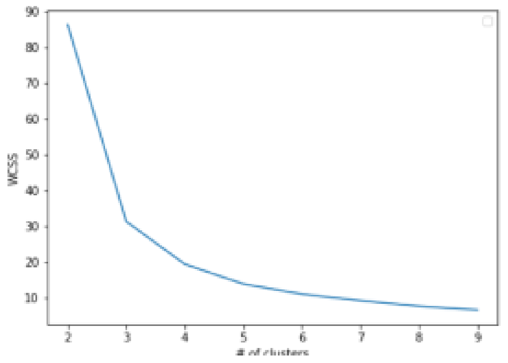
Each point has multiple a and b. For a data point X the average distance to each other data point in the cluster is a. For a data point X the average distance to each other data point in the other cluster is b.



	Distance from points in the GREEN cluster		
	2,1	3,1	b
1,3	$\sqrt{5}$	$\sqrt{8}$	3.65
1,2	$\sqrt{2}$	$\sqrt{5}$	1.8
1,1	1	2	1.5
2,2	1	$\sqrt{2}$	1.2



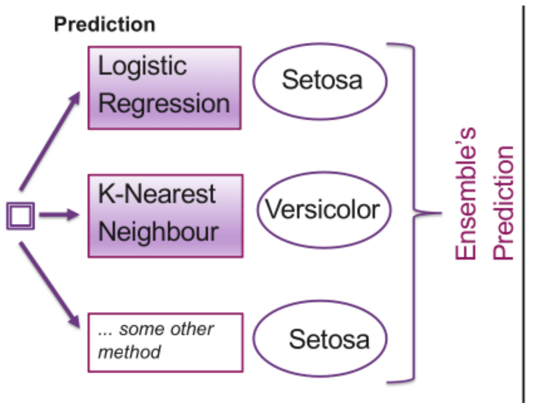
	Distance from other points in the cluster				
	1,3	1,2	1,1	2,2	a
1,3	-	1	2	$\sqrt{2}$ = 1.41	1.47
1,2	1	-	1	1	1
1,1	2	1	-	$\sqrt{2}$	1.47
2,2	$\sqrt{2}$	1	$\sqrt{2}$	-	1.27



Wisdom of Crowd
Suppose you have a very difficult question to answer. If you don't know the answer you can search for the best suited expert, who answers your question. But sometimes it is better to ask many random people and aggregate the answers. This might produce the better result. This is called the *Wisdom of Crowd*.
Ensemble
A group of predictors is called Ensemble. In ML you can aggregate several weak models. If you aggregate predictors of regressors or classifiers you might get a better predictor than the best predictor! This is a type of Wisdom of crowd.
Enseble Learning
Ensemble Learning use different models. A model can be different in the following dimensions:

- algorithms
- hyperparameters
- training data

Different model predict a solution. And in a vote the final prediction is chosen.



Hard / Soft Voting
Hard and soft voting are used to decide which prediction should be used in Ensemble Learning.

- *Hard voting*: the prediction with the most votes is used
- *Soft voting*: the prediction with the highest class probability

Bagging / Pasting
Sampling with replacement is called *Bagging* (Bootstrap Aggregating). Sampling without replacement is called *Pasting*.
No free luch theorem

> No free lunch theorem (informal): no single machine learning algorithm is universally the best-performing algorithm for all problems – Marko Lehmann

Out-Of-Bag
In Bagging some data points may be used several times. Other data points may not be used at all. This not seen data are called *out-of-bag* (oob) data and can be used for evaluation.