Severin Grimm | Ostschweizer Fachhochschule

Contents

Week 1 - Introduction and NLP

Introduction

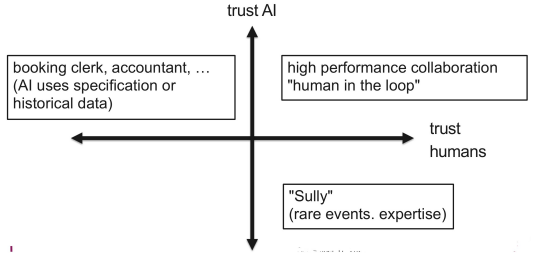
What is AI?

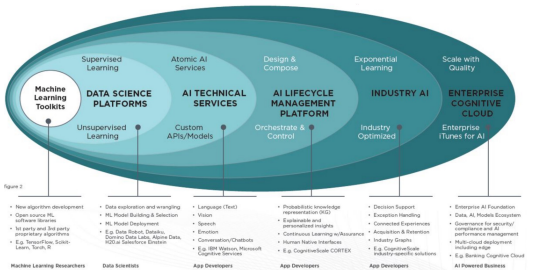
A broad concept with different interpretations. In this class: Where a computer **learns from data**. Often called **statistical machine learning.**
Ultimate research goal: AGI (artificial general intelligence). a hypothetical computer program that can perform intellectual tasks as well as, or better than, a human.

How to tell if a machine is intelligent

With the **Turing Test:**
If the behaviour is indistinguishable from a human, it is intelligent.
Problems: AI mus learn to lie, If a complex problem is too hard for humans.

Where to use it





NLP (Natural Language Processing)

NLP is the **automated processing** of human language. A Subfield of AU, aiming to understand and generate language. Long research history, took-off recently. Still difficult, but Big-Business.

Dialogflow

Intents

What does the user want? Give example sentences and Dialogflow will generate new sentences to match intent.

Entities

Extract information from sentence. Assign words to entities (variables). Again, define examples and dialogflow learns new words.

Dialog Control

Linear: Collect all Information, for e.g. booking an appointment
Non-Linear: More a real conversation with dependend answers based on changes in context. Context can be set as Input and Ouput.

Week 2 - NLP, Representation, Embeddings

Examples

Information Retrieval (Search);
Machine Translation;
Sentiment Analysis (Check if Text is positive or negative, used in marketing);
Information Extraction (Bring to Structured Text); Question Answering;

4 Ingredients of Machine-Learning

1-4 are basics. 5-7 are improvements for success

1. **Data:** Quality of the data is the limit for how good we can do. A few 2-dimensional points with, presumably, a linear relationship.
2. **Cost-Function (Loss):** formal mathematical expression for good and bad. e.g.: Mean Squared Error (MSE)
3. **Model:** from linear model with two parameter up to million-parameter neural network. Different task, different model.
4. **Optimization Procedure:** Algorithm that changes parameters of model to minimize cost-function. E.g.: SGD, ADAM, RMSProp.
5. **Performance optimization:** Good pipelines.
6. **Visualization and evaluation of the learning Process:** Learning boards.
7. **Cross-Validation & Regularization:** Not Covered

Example

Task	**Text Analysis, Sentiment Classification → Supervised learning. Binary classification**
Example	comment: 'This is the worst movie I've ever seen 🤢' label: 0 (=negative)
Link	https://www.tensorflow.org/text/guide/word_embeddings
Data & Preprocessing	labeled sentences (Words) from IMDB How to put words into numbers ? Embeddings !
Model	Multi-layer NN with ~160'000 parameters. Input Layer: sentences in an appropriate representation Output-Layer: single neuron. probability that the input belongs to class 1.
Loss	Binary Crossentropy
Optimizer	Adam

Representing words in a computer

Chat-bot must understand the request. You can use vectors to represent words based on their meaning. Also called **word embedding**. Word embeddings can be learned from data.

What is a word?

"You shall know a word by the company it keeps"
What is a *mukawibuu*? Look at that little furry *mukawibuu* with white paws climbing a tree.

How "similar" is "mukawibuu?" to each of the following words?
a) rat
b) cat
c) train

- A word can be "defined" by context
- Words with similar semantics share context (e.g. rat and cat are both animals)

1: One-hot representation

A **vertical** Vector with a single 1-Value and all other 0. Vector is as long as how many words: 100 Word = 100 Tokens. Dot-Product is always **0** for different words, 1 for same words.

It	rains	.	We	go	home	.	It	stops	raining	.
1	0	0	0	0	0	0	1	0	0	0
0	1	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	1	0	0	0	1
0	0	0	1	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0	0	0
0	0	0	0	0	1	0	0	0	0	0
0	0	0	0	0	0	0	0	1	0	0
0	0	0	0	0	0	0	0	0	1	0

Disadvantage:
- High dimensional vector space. 10'000 Words each represented with a vector.
- **Sparse** representation: a single 1 and 9'999 zero.
- No generalization: Hummus is close to Chickpea, not zeus. **NO meaning of word is captured.**

Sparse: means "dünn besiedelt, spählich"

2: Indexing

cat	→	0
mat	→	1
on	→	2
sat	→	3
the	→	4
.	→	5

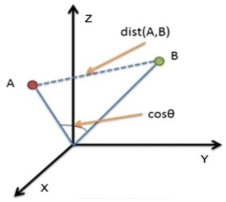
"The cat sat on the mat" → [4, 0, 3, 2, 4, 1, 5]

Dense equivalent of one-hot encoding, not more useful. Often used as preprocessing step. (Index fed as data to learn more useful representations)

3: Distributed representations

Similar words share similar representations. Distributed representations can be learned.
Input: Word Index -> mathematical function -> Output: high deminsional Vector.
Advantage: Close Vectors have semantic similarity. With good vectors, you can do math. Dot-product is a measure of similarity. Add/Subtract vectors.

Cosine-similarity or Cosine-distance



$$\cos(\theta) = \frac{\mathbf{A}\cdot\mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}}$$

Week 3 - Discreete Random Vars

Random Variable - Informal Definitions

A random variable X (or some other upper case letter) is a variable that takes a numerical value x (lower case), which depends on a random experiment. The values depend on outcomes of a random phenomenon.
Discreete: X takes any of a finite set of values, e.g. 1.5, 2.693, 5, 6.3, 10
Continuous: X takes any value of an uncountable range, e.g. the real numbers in the interval (2, 7)).
Before observing the outcome of a random experiment, the "best we can know" is a list of all possible values, and a list for each possible value how likely it will occur.

Other definitions

Algebra: variable value turns the equation into a true (not changeable)
Programming: variable value is determined (and changeable) by the assignment.

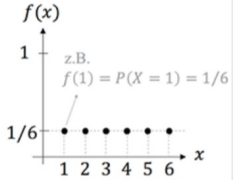
Severin Grimm | Ostschweizer Fachhochschule

Example

Value x of the random variable X	1	2	3	4	5	6
Pr(X=x)	1/6	1/6	1/6	1/6	1/6	1/6

Pr(X=x) is the probability that the random variable X takes the value x.

Probability Mass Function: A PMF specifies a (discrete) Probability Distribution.



Joint-Probability

This is usually written as **Pr(X=5, Y=4) = 1/36** or more compact as **p(5,4)=1/36** you'll also come across **PXY (5,4)=1/36** The komma is a logical **AND**.

	X=1	X=2	X=3	X=4	X=5	X=6
Y=1	1/36	1/36	1/36	1/36	1/36	1/36
Y=2	1/36	1/36	1/36	1/36	1/36	1/36
Y=3	1/36	1/36	1/36	1/36	1/36	1/36
Y=4	1/36	1/36	1/36	1/36	1/36	1/36
Y=5	1/36	1/36	1/36	1/36	1/36	1/36
Y=6	1/36	1/36	1/36	1/36	1/36	1/36

Marginal Probability

Marginal probability is the probability of an event irrespective of the outcome of another variable. Can be calculated from the Joint Probability. (Spalte von Joint Probability)

Independent random variables

If the first dice is a 6, there is no probability that the second dice is also a 6. If X and Y are independent:
Pr(X,Y) = Pr(X) * Pr(Y)

Correlated random variables

Events that are **not independent.**
Here given as joint probability:

	X=0	X=1	X=2
Y=0	0.35	0.21	0.03
Y=1	0.10	0.07	0.04
Y=2	0.00	0.05	0.05
Y=3	0.00	0.02	0.08

OST

X: The event to observe clouds (0= no clouds, 1= small clouds, 2= big clouds)

Y: The event that it rains (0=no rain, 1=light rain, 2=moderate rain, 3=heavy rain)

Conditional Probability

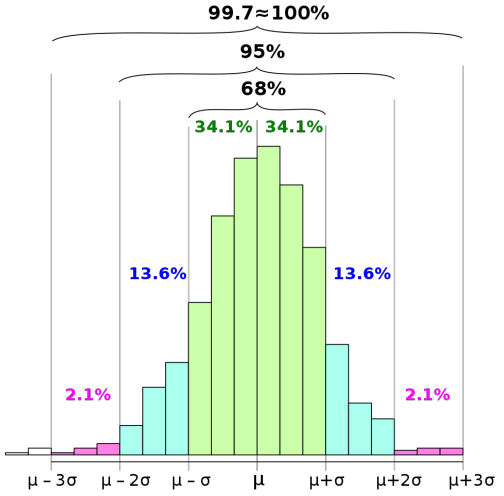
Consider: You observe small clouds. What is the probability for moderate rain? The observation X is no longer random. X is observed, its value is fixed. We say "X is given". To answer the question, we calculate the probabilities of Y given X with **Pr(Y|X)**.
Important: Do not just read the 0.05 in this example, because the total value of the Column is **not 1.**
P(Y|X) = P(X,Y) / P(X)
In this example: P(Y|X) = 0.05 / 0.35 = 0.1428 = 14.28%

Bayes Rule

$$P(X|Y)P(Y) = P(Y|X)P(X)$$

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

Empirical rule



Regression

Linear Regression is **Supervised Learning** and subclass **Regression**.

What is a model?

In ML, we use the term **model** for any mathematical function that explains the data:
$y_i = f(x_i)$
$y_i = f(x_i) + \epsilon_i$
where ϵ_i is unexplained noise. It is often assumed that ϵ_i follows a normal distribution.
Instead of approximating y_i, we calculate an **estimate** $\hat{y_i}$ (y hat) of the usually unknown y_i:

$$\hat{y_i} = f(x)$$

Linear Regression

In ML, linear regression falls into the category of supervised learning. **Two categories:**
Interpretation: We want to understand if some input has an effect on the output. Example: Is there a relationship between smoking cigaretts and the risk of lung cancer?
Prediction: Given some sensor data like oil pressure, temperature etc. a model could predict (and thereby hopefully prevent) an engine failure. «

- Only considers a linear relationship between input and output
- In the simplest case, x and y are scalars and the linear model therefore has only two free parameters
- The goal is to identify a (slope) and b (intercept) for which the linear model best explains the data

Examples:
Linear: $\hat{y_i} = ax_i + b$
Squared: $\hat{y_i} = w_2 * x_i^2 + w_1 * x_i + w_0$
Cubic: $\hat{y_i} = w_3 * x_i^3 + w_2 * x_i^2 + w_1 * x_i + w_0$

Mean Squared Error (MSE)

- Loss we want to minimize
- Usually divided by 2

$$e_i = y_i - \hat{y_i}$$

The difference e_i, called **residual**.
(Value between linear function and datapoint)

$$MSE = \frac{1}{2N} * \sum_{i=1}^{N} e_i^2$$

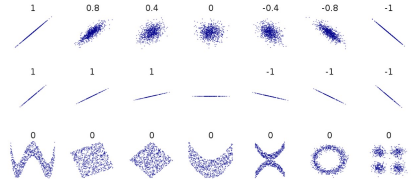
$$MSE = \frac{1}{2N} * \sum_{i=1}^{N} (y_i - \hat{y_i})^2$$

Einsetzten:

$$\hat{y_i} = ax_i + b$$

$$E = \frac{1}{2N} * \sum_{i=1}^{N} (y_i - (a * x_i + b))^2$$

Correlation and Causality



- Correlation is not causality
- Correlation refers to the degree to which a pair of variables are linearly related
- Linear regression is a tool to detect correlations between two or more variables
- Correlation can be quantified using the Pearson correlation coefficient
- Even if the correlation coefficient is 0, the data can still be highly structured

Complex linear models

We can have linear models that depend on more than one input variable. In this case, x is a vector with p features (=dimensions):

$$y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \beta_3 x_{i3} + \cdots + \beta_p x_{ip}$$

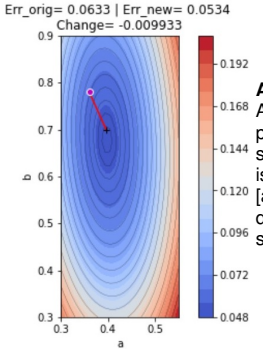
For many data (x_i, y_i) we can express the linear model in matrix notation:

$$X\beta = y$$

Optimization & SGD

MSE as Function

Gradient Descent



An iterative method.
At each iteration, the model parameters are updated such that the Loss (MSE) is reduced. At any location [a,b] move a step in the direction where the error shrinks the most.

Calculation

MSE Formula:

$$E = \frac{1}{2N} * \sum_{i=1}^{N} (\hat{y_i} - (a * x_i + b))^2$$

Gradient of E:

$$E = \begin{bmatrix} \frac{\partial E}{\partial a} \\ \frac{\partial E}{\partial b} \end{bmatrix} = \begin{bmatrix} \frac{1}{N} \sum_{i=1}^{N} (y_i - (a * x_i + b))(-x_i) \\ \frac{1}{N} \sum_{i=1}^{N} (y_i - (a * x_i + b))(-1) \end{bmatrix}$$

Update Rule

$$\begin{bmatrix} a \\ b \end{bmatrix}_{t+1} = \begin{bmatrix} a \\ b \end{bmatrix}_t - \alpha \begin{bmatrix} \frac{\partial E}{\partial a} \\ \frac{\partial E}{\partial b} \end{bmatrix} \Big|_{\begin{bmatrix} a \\ b \end{bmatrix}_t} \quad (1)$$

The algorithm starts at some initial position $[a, b]_{t0}$. Then, the update rule is applied repeatedly. The number of update steps needed to come "close enough" to the minimum depends on the problem.

Stochastic Gradient Descent (SGD)

- At each iteration, the gradient is calculated on a (randomly selected) subset of the data
- For a fixed learning rate, SGD does not converge

Annealed SGD

- The learning rate alpha is reduced over time
- This is called (simulated) annealing
- There are different options (called schedules) how to reduce alpha over time

General remarks on SGD

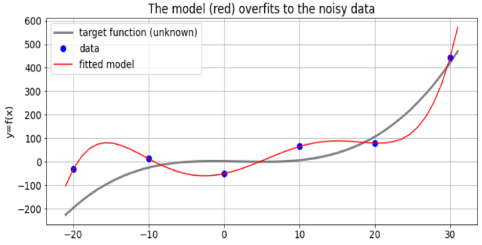
- Gradient-based methods only work if we can express a Loss function as a differentiable function
- SGD is dealing woth only a single datum at each iteration. This is very inefficient and rarely used.
- Batch- or mini-batch gradient-descent is usually used

Generalization & Regularization

Overfitting

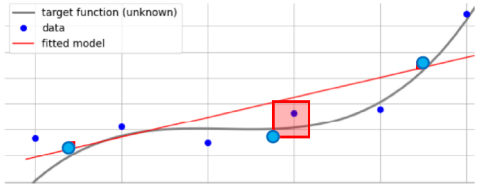
- A model that perfectly fits the data does not have to be perfect
- In-Sample Error (Trainig error) was minimized (MSE = 0)
- Out-of-sample Error (Generalization Error, Test Error) is the MSE of new Data
- A good model has a low Generalization Error
- Overfitting happens if the MSE of Training Error is small thanks to a complex model but the Generalization Error is large

Severin Grimm | Ostschweizer Fachhochschule



Underfitting

- Using a too simple model
- In-Sample Error is large
- Generalization Error is large



Training-Set, Test-Set, Model Evaluation

- The Generalization Error can't be calculated
- But Estimated
- Split the data into 2 sets
  - Training-Set (80% of data)
  - Test-Set (20% of data)

Training:

- Fit the model to the training set
- This minimizes the in-sample error

Evaluating

- Using the Test-Set
- Produces the Test-Error
- This is an estimate of the Generalization Error

Trade-off

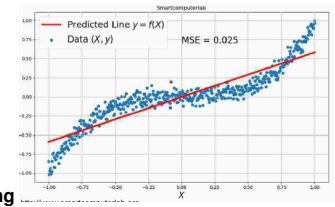
High Bias - Low Variance

High Bias

- A too simple model for the given data

Low Variance

- The model is relatively stable
- Very simular model if trained with new data



Risk is Underfitting

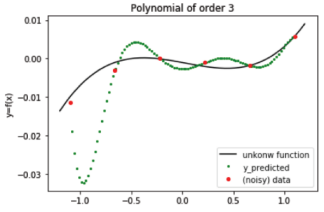
Low Bias - High Variance

Low Bias

- A more complex model can better explain the data

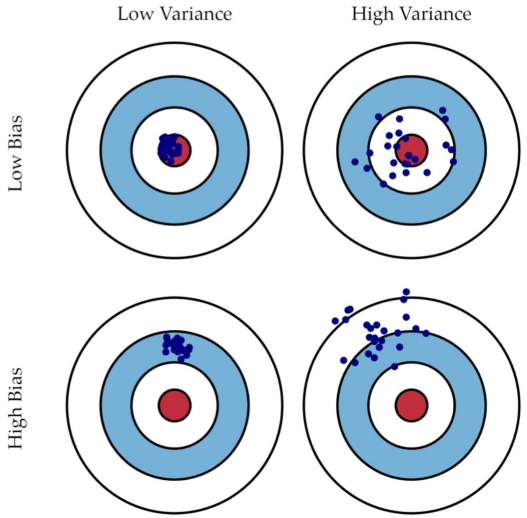
High Variance

- Given a new datapoint, the MSE can be very large
- For a different set with more datapoints, the model may be very different



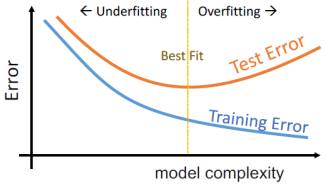
Risk is Overfitting

As Picture



Trade-off Goal

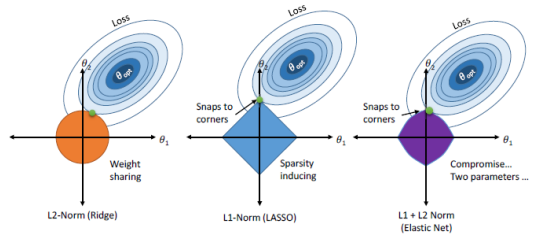
- Higher bias implies lower variance
- Lower bias implies higher variance
- In practice, all we want is low variance
- The model can only be as complex as the data permits
- You have to find an optimal balance between bias and variance



Regularization

- Technique to control the model complexity
  - Add a penalty term to the Loss
  - More complex models get a higher penalty
  - Add a constrain to the optimization process
  - *regularized loss = MSE + λ model-complexity*

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



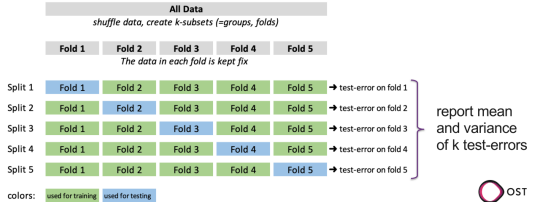
Keras

```
model = tf.keras.models.Sequential()
inputs = keras.Input(shape=(poly_order,))
model.add(inputs)
output_layer = layers.Dense(1, activation=None,
      use_bias=True)
model.add(output_layer)
model.compile(
    # Optimizer
    optimizer =
        keras.optimizers.Adam(learning_rate=0.05),
    # Loss function to minimize
    loss=keras.losses.MSE,
    # List of metrics to monitor
    metrics=[keras.metrics.MSE]
)
history_object = model.fit(X_scaled, Y_Data,
      batch_size=5, epochs=500, verbose=False)
x_ftr = pipe.transform([[1.5]])
y_hat = model.predict(x_ftr)
```

Cross-Validation

k-fold Cross-Validation

Without Crossvalidation: Train 1 Model with 80% of the Data and Test with 20%.
With **k-fold:**
Repeat the split-train-test procedure for k times.



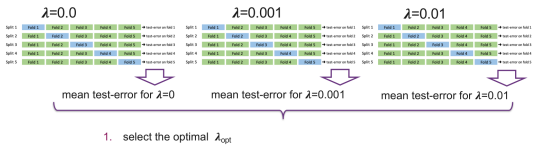
E.g.: Split into k = 5 folds. A Datapoint is fixed in the fold, so no shuffling afterwards. Then test k times, each time with a different fold as test data. Every split is used as test data once.

Comments

Typical values are 5, 10 or N.
k = N is good for small datasets. Every datapoint is in a single fold, and every datapoint is at least once a test set as single datapoint. Bad for huge datasets, as it has to run N times.
Preprocessing: Do not preprocess the whole dataset, but for each split (Calculation mean and variance).

k-fold - Use Case 2

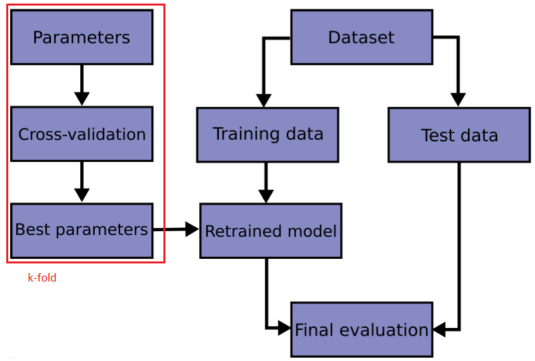
model parameter: Usually weights of the model, like param a and b.
hyper parameters: Params for training procedure (regularization parameter λ)



Lambda is often selected on logaritmic basis and very small.

scikit-learn: Cross-Validation

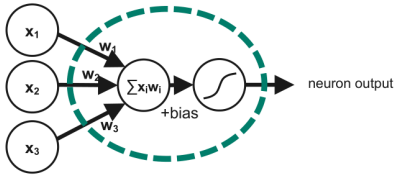
Take a dataset at the beginning, and do not use it **at all** until the end. This helps for a final evaluation. Do useful for really small datasets, like 10 patients with rare diseases, because you cannot just leave two away.



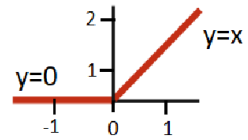
ANN (Artificial Neural Networks)

Artifical Neurons

The artificial neuron receives an input vector multiplied with a weight.



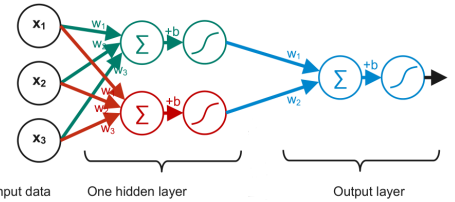
The neuron calculates the sum adds a bias **b** and passes it thorugh a **nonlinear** activation function.



example of a popular non-linear activation function:
The Rectified Linear Unit (ReLU) cuts-off negative values (y=0). Positive-values are passed through (y=x).

Bias here is not the same as in the Trade-Off. Here it is the height on the x scale (a*x + **b**).

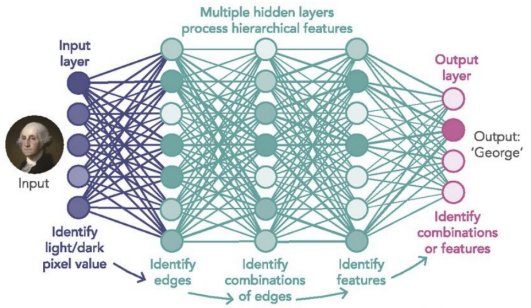
A simple ANN



This ANN has: 2 Layers, 3 Neurons, 11 trainable params.
Parameters: The Weights and the Bias of each neuron.

Severin Grimm | Ostschweizer Fachhochschule

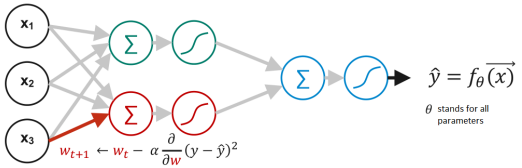
A complex ANN



ANN with multiple hidden layers are called **deep** neural networks.

How to train an ANN

The ANN is initialized with random weights and produces an output. Then, an optimizer (e.g.: SGD) reduces the cost-function (e.g.: MSE). Aat every iteration, for every weight and bias, the partial derivative $\frac{\partial}{\partial w}(y-\hat{y})^2$ is calculated. This is done with the Algo **Backpropagation**



ANN in Computer Science

An ANN is a data-structure calles expression-tree. Each node can be evaluated. Finding the optimal weigths is only possible because of the Backpropagation Algo.
Math perspective: Backpropagation is basically the chain rule.
CS perspective, Backpropagation is a recursive, Dynamic-Programming algorithm.

Logistic Regression

Logistic Regression is **Supervised Learning** and subclass **Classification**. Logistic Regression is Classification prediction with features.

Scenario Admission Decision

Question is: Will Person be admitted for Masters.

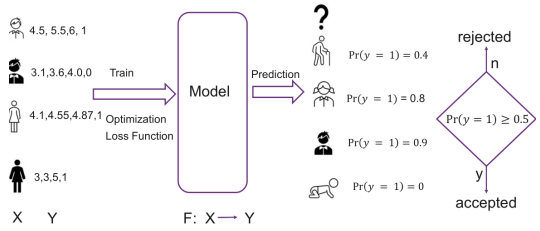
Data

X1	X2	X3	Y=
Secondary School Grade	**BS Grade**	**Years of Work experience**	**Acceptance**
4.5	5.5	4	0
5	6	10	1
. . .			
3	4	6	1

Joint Probability

$P(y=1|x_1=4.5, x_2=5, x_3=5.5)$

Whole Model



Notation

x_{n_1} = *years of Work experience of applicant n*
x_{n_2} = *BS grade of applicant n*
x_{n_3} = *Sec. School grade of applicant n*

$\mathbf{x}_n = (x_{n_1}, x_{n_2}, x_{n_3})$
$X = \{(x_{1_1}, x_{1_2}, x_{1_3}), (x_{2_1}, x_{2_2}, x_{2_3}), \ldots, (x_{n_1}, x_{n_2}, x_{n_3})\}$

$y_n = \begin{cases} 1 & \text{application n accepted} \\ 0 & \text{application n NOT acccpted} \end{cases}$

$Y = \{y_1, y_2, \ldots, y_n\}$

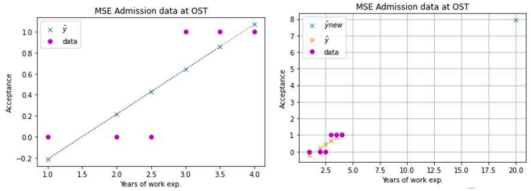
N in this case is participant. Every participant has 3 features and one result y.

Example Probability

What is the probability that applicant # 44 is accepted?
$Pr(y_{44} = 1|(x_{44,1}, x_{44,2}, x_{44,3}))$
$= Pr(y_{44} = 1|x_{44})$
$= Pr(y = 1|x)$
As soon as the context is set, the notation is minimized.

Using Linear Regression

Not good: Values higher and lower as 1 and 0. Outside of probability.

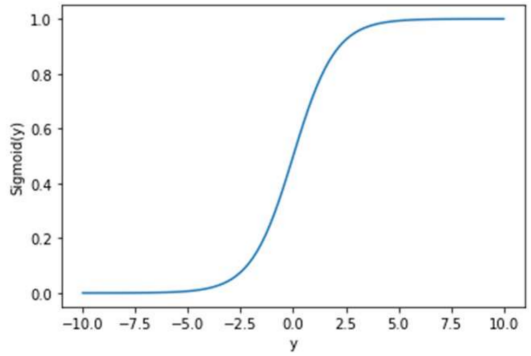


LR does not work for classification. MSE forces as wrong solution.

Solution

The sigmoid function (Model)

$$p = sigmoid(\hat{y}) = \frac{1}{1+e^{-\hat{y}}}$$



$$\hat{y} = w_1x_1 + w_2x_2 + w_3x_3 \ldots$$

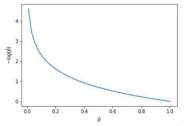
w is unknown and needs to be learned.

Maximum Likelihood (Cost-Function (Loss))

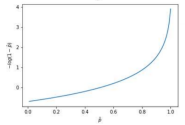
Maximize the likelihood by minimizing cost:
$MinimizeCost(W) =$

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

❑ $-\log(p)$ $\begin{cases} large\ when\ p = 0 \text{ and } y = 1 \\ small\ when\ p = 1 \text{ and } y = 1 \end{cases}$



❑ $-\log(1-p)$ $\begin{cases} large\ when\ p = 1 \text{ and } y = 0 \\ small\ when\ p = 0 \text{ and } y = 0 \end{cases}$



Classifier Evaluation

Confusion Matrix (Model Evaluation)

		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?
$Accuracy = (t_p + t_n)/n$
Mean Error:
How often is the classifier wrong?
$Error = (f_p + f_n/n)$
Precision:
When the prediction is 1, how often is it correct?
$Precision = t_p/(t_p + f_p)$
Sensitivity, Recall, True Positiv Rate (TPR):
How often the prediction is 1, when it's actually 1?
$Recall = t_p/(t_p + f_n)$
Miss Rate, False Negative Rate (FNR):
$MissRate = 1 - TPR$

Mean Accuracy

Can be misleading. If you guess 100% of Applicants are denied, and actually 90% are really denied, accuracy is 90%. But the precision and recall are bad.

	Prediction (accepted)	**Prediction (not-accepted)**
Actual(Accepted)	0 (tp)	10(fn)
Actual(not-accepted)	0(fp)	90(tn)

$Precision = \frac{0}{(0+0)} = \frac{0}{0}$
$Recall = \frac{0}{(0+10)} = 0$

Precision vs. Recall Tradeoff

Increasing precision reduces recall and vice versa.
Example 1 - classify safe videos for kids:

- Low recall (rejects many safe videos), i.e., high precision (keeps only safe ones)
- High recall (rejects few safe videos), i.e., low precision (keeps unsafe videos)

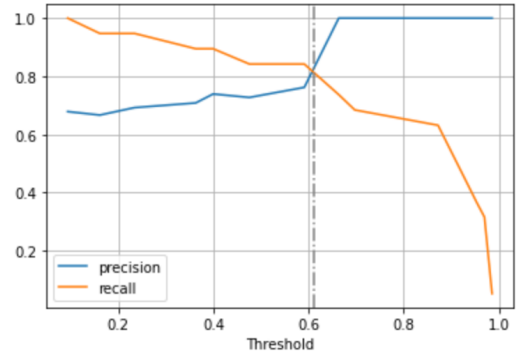
Example 2 - classify shoplifters:

- Low precision but High recall (false alerts are ok)

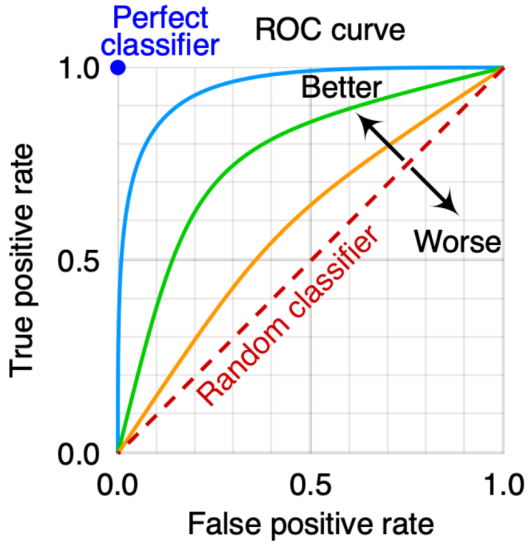
Where to set the Threshold?

It's a business decision. Rather have high precision or high recall.

Precision vs. Recall Curve



Receiver Operating Characteristics (ROC)

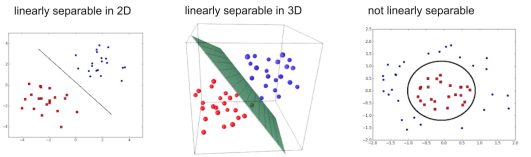


Depicts relative trade-offs between true positive (benefits) and false positive (costs).
AUC: Area under (ROC) curve. The greater the area under the curve, shows the higher quality of the model. Model has to be trained **multiple** times, to create a curve. One time is only a point.

Severin Grimm | Ostschweizer Fachhochschule

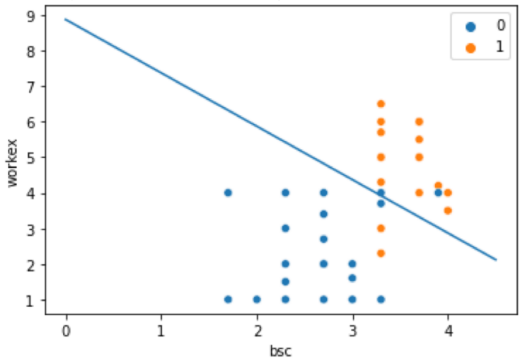
K-Nearest-Neighbours (KNN)

Linearly spearable data



Decision Boundary

Boundary with two **features** (bsc grades and workex):



Use the logistic regression (sigmoid) for calculation of boundary. If a simple line perfectly separates the classes, then the classes are said to be linearly separable!

Multi-class classification

LR can be extended to multiple classes in some obvious ways.

One-vs.-rest

- train a single classifier for each class “c”, with the samples of class “c” as positive samples and all other samples as negatives.
- Apply all classifiers to an unseen sample x
- the classifier reports the highest p is the class

One-vs.-one

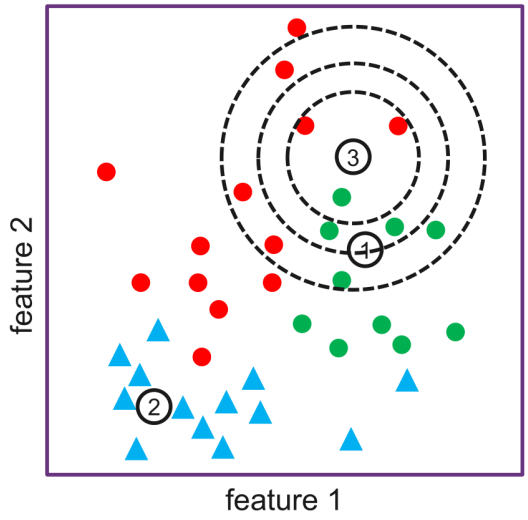
- Train classifiers to distinguish between each pair of classes (c1,c2), (c1, c3), (c1,c4), (c2, c3), (c2, c4), (c3,c4).
- Apply all classifiers to an unseen sample x.
- Combine the results to produce final classification.

KNN classification

A datapoint is known by the company it keeps.
Given a test data point, KNN:

- computes k nearest neighbours of it,
- returns the most frequent class of the k neighbours

For this, **all** distances to all points have to be calculated.



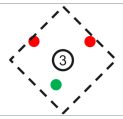
KNN Details

1. Load the training as well as test data.
2. Choose the value of k
3. For each test data points x_{test}
   a) For all training data x_{train}, calculate the $d(x_{test}, x_{train})$
   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

Distance Metric

Cosine Distance: See Week 2

Manhattan Distance: $d_M = \sum_{i=1}^{n} |X_{1,n} - X_{2,n}|$



Euclidean Distance: $d_E = \sqrt{\sum_{i=1}^{n} (X_{1,n} - X_{2,n})^2}$

Clustering

Clustering belongs to Unsupervised Learning with the Subcategory Clustering.

In General

Data often has structure. But the structure can be hidden by noise. The goal is to identify hidden patterns among the data.

Applications:

- Social network analysis
- Astronomical data
- market segmentation
- recommendation systems

Same concept as KNN, but taken further. We try to identify and group data points whose features are similar (close together). We partition the data into so called “clusters”.

Naive K-means

1. Let us assume we know the number of clusters k_c
2. Initialize the value of k cluster centres $(C_1, C_2, \dots C_{k_c})$
3. Assignment:
   a) Find the squared Euclidean distance between the centres and all the data points.
   b) Assign each data point to the cluster of the nearest centre.
4. Update: Each cluster now potentially has a new centre (mean). Update the centre for each cluster
   a) New Centres $(C'_1, C'_2, \dots C'_{k_c})$ = Average of all the data points in the cluster $(1, 2, \dots K_c)$
5. If some stopping criterion met, Done
6. Else, go to Assignment step 3

Stopping Criterion

- When centres don’t change (time consuming)
- The datapoints assigned to specific cluster remains the same (takes too much time)
- The distance of datapoints from their centres $>=$ treshold we have set
- Fixed number of iterations have reached (choose wisely)

Initialization

- Performance depends on the random initialization
- Some seeds can result in a poor convergence rate
- Some seeds can converge to suboptimal clustering
- If centres are very close, it takes a lot of iterations to converge
- Initialize randomly, run multiple times

Standardization of data

- Features with large values may dominate the distance value
- Features over small values will have no impact
- Normalize values!

Sklean k-means

Initialization

- Init = K-means++
- Only initialization of the centroids will change
- Chosen centroids should be far from each other

max_iter:

- Number of iterations before stopping

n_init:

- Number of time the k-means algorithm will be run with different centroid seeds

Evaluating Cluster Quality

- Make clusters so that for each cluster the distance of each cluster member from its center is minimizes

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

- Sum of squared distances to center
- As small as possible

Silhouette Score

- How far the datapoints in one cluster are from the datapoints in another cluster
- SS of a point: $\frac{b-a}{max(a,b)}$
- a: average intra-cluster distance (distance between each point within)
- b: average inter-cluster distance (distance between a cluster and its nearest neighbour)

Ensamble Methods

Wisdom of Crowd

- Suppose you have a difficult question
- Ask many people and aggregate the answer
- This might work very well instead of finding the best suited person

Ensamble

- Wisdom of Crowd can be applied to ML
- Instead of finding the best model, aggregate the results of weak models
- Aggregate predictions of regressors or classifiers
- Might get better accuracy than the best predictor
- Ensamble: group of predictors

Ensamble Method

- Suppose we have many different weak models (better than random)
- Get prediction from all of them and take a vote
- Class with most votes is the predicted class
- Commonly used towards the end of a project
- **Requirement**: enough models / diverse models

Bagging and Pasting

Bagging (Bootstrap Aggregating)

- Sampling with replacement
- Allows data points to be used several times

Pasting

- Sampling without replacement

No free lunch theorem

No single machine learning algorithm is universally the best-performing algorithm for all problems

Out of Bag (oob) Evaluation

- Using Bagging
- Some Data Points may not be used at all
- Use them for evaluation