AI Foundations | AIFo

Summary

CONTENTS

1. Machine Learning (ML) 2
2. Probability 2
2.1. Random Variables 2
2.2. Discrete random variables 2
2.2.1. Multiple Random Variables 3
2.3. Conditional Probability 3
2.4. Continuous random variables 4
2.5. Distributions 6
2.5.1. Univariate normal distribution 6
2.5.2. Bernoulli distribution 7
3. Tree Diagrams and sequential events 8
3.1. Vocab for human experiments 10
4. Linear Regression 10
4.1. Ingredients of Machine Learning 10
4.1.1. The 4 main ones 10
4.1.2. Others 10
4.2. Linear Regression 11
4.2.1. Residual sum of square 11
4.2.2. Model 12
4.2.3. Mean squared error 12
4.2.4. Correlation and causality 12
4.2.5. Multiple linear regression 13
5. Optimization 14
5.1. Partial derivative and gradient 14
5.2. Gradient Descent 15
5.2.1. Stochastic Gradient Descent (SGD) 16
6. Natural Language Processing (NLP) 18
6.1. Large Language Model (LLM) 19
6.1.1. How a LLM generates text 19
6.1.2. Where the LLM stores its “knowledge” 19
6.2. Representation of words 19
6.2.1. One-hot representation 19
6.2.2. Indexing 19
6.2.3. Distributed Representation (dense vectors) 19
7. Project 20

WARN:
This document is still under development and will be completed around 1.2.2027

1. MACHINE LEARNING (ML)

There are many different AI research fields and AI applications. Algorithms and applications where a computer learns from data are typically in the field of statistical machine learning (ML).

Computer: performs a mapping from input to output.

Processing: any sort of non-trivial "calculation" (mapping) or information processing

Input: any sort of data/information (sensory input, bits, mechanical configuration…)

Output: any sort of response (data, actions, new state of a system…)

Data Visualization: Gives intuitive understanding of structure in the data, helps in identifying patterns, detecting outliers and data quality. Raw data $\rightarrow$ Information

TODO:
plot examples

https://ourworldindata.org/

https://informationisbeautiful.net/

2. PROBABILITY

2.1. RANDOM VARIABLES

A random variable X is a function from the sample space to the real numbers.

$$X : S \rightarrow \mathbb{R}$$

Random Variables come in two flavours:

discrete: X takes any of a finite or countable set of values, e.g. $\{-8, 1.5, 1.5, 2.693, 5, 6.3, 10\}$

continuous: X takes any value of an uncountable range, e.g. the real numbers in the interval $(2, 7)$.

$\Pr(X = x)$ is the probability that the random variable X takes the value x. It's often written as $P(x)$ or $p(x)$ or $P_X(x)$ or $\mathbb{P}(x)$.

TODO:
MathFML explained this a lot better

Definition 1: **Range**

The range of a random variable X, shown by $\text{Range}(X)$ or R_X, is the set of possible values of X.

2.2. DISCRETE RANDOM VARIABLES

The Probability Mass Function (PMF) of a discrete random variable is a function $P(x)$ that provides the probability for each value x of a discrete random variable X.

TODO:

$$0 \leq P_X(x) \leq 1$$

$$\sum_{x \in R_X} P_X(x) = 1$$

$$\text{for any set } A \subset R_X,\ P(X \in A) = \sum_{x \in A} P_X(x)$$

***Example** 1:*

Dice rolls

Value x of the random Variable X	1	2	3	4	5	6
$P(X = x)$	$1/6$	$1/6$	$1/6$	$1/6$	$1/6$	$1/6$

From the PMF, we can calculate the expected value

$$E[X] = \sum_{i=1}^{\infty} P(x_i) \cdot x_i$$

2.2.1. Multiple Random Variables

The (joint) properties of multiple random variables are defined by the ***Joint Probability Mass Function***, usually simply called Joint Probability.

For **independent** random variables, the joint probability is simply the product of the individual probabilities

$$P(X \cap Y) = P(X) \cdot P(Y)$$

***Example** 2:* **Rolling two dice**

The random experiment of rolling die 1 and die 2 together has 36 possible outcomes (events). The Joint Probability is:

$$P(X = 5, Y = 4) = \frac{1}{6} \cdot \frac{1}{6} = \frac{1}{36}$$

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

2.3. CONDITIONAL PROBABILITY

For **dependent** random variables we use the conditional probability of X given Y defined as

$$P(X|Y) = \frac{P(X \cap Y)}{P(Y)}, \quad \text{when } P(Y) > 0$$

TODO:

$$P(X \cap Y) = P(X|Y) \cdot P(Y)$$

$$X \cap Y = \varnothing \Rightarrow P(X|Y) = 0$$
$$Y \subset X \Rightarrow P(X|Y) = 1$$
$$X \subset Y \Rightarrow P(X|Y) = \frac{P(X)}{P(Y)}$$
$$P(X \cap Y) = P(X)P(Y|X) = P(Y)P(X|Y)$$

Independence: Two events X and Y are independent iff $P(X \cap Y) = P(X)P(Y)$

Law of Total Probability: If $Y_1, Y_2, Y_3, ...$ is a partition of the sample space S, then for any event X we have

$$P(X) = \sum_i P(X \cap Y_i) = \sum_i P(X|Y_i)P(Y_i)$$

Bayes' Rule: For any two Events X and Y, where $P(X) \neq 0$, we have

$$P(Y|X) = \frac{P(X|Y) \cdot P(Y)}{P(X)}$$
$$\text{Interpretation:} \quad \text{Posterior} = \frac{\text{Likelihood} \times \text{Prior}}{\text{Normalizer}}$$

Conditional independence: Two events X and Y are conditionally independent given an event Z with $P(Z) > 0$ if

$$P(X \cap Y|Z) = P(X|Z)P(Y|Z)$$
$$\Rightarrow P(X|Y \cap Z) = P(X|Z)$$

***Example** 3:* **Chance of rain**

Let X be the event to observe clouds (0=no clouds, 1=small clouds, 2=big clouds) and Y the event that it rains (0=no rain, 1=light rain, 2=moderate rain, 3=heavy rain)

	$X=0$	$X=1$	$X=2$
$Y=0$	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

TODO:
Given a known joint distribution of two discrete random variables, say, X and Y, the marginal distribution of either variable – X for example – is the probability distribution of X when the values of Y are not taken into consideration. This can be calculated by summing the joint probability distribution over all values of Y.

$$P(X) = \sum_Y P(X, Y)$$
$$P(Y) = \sum_X P(X, Y)$$

2.4. CONTINUOUS RANDOM VARIABLES

For a continuous random variable X following holds:

$$P(-\infty < x < \infty) = 1$$

$$P(a < x < b) = \int_a^b f(x)\,\mathrm{d}x$$
$$P(x) = 0$$

The ***uniform probability distribution*** $\mathrm{unif}(0,1)$ characterizes an experiment in which one number is chosen from the sample space $\Omega = [0;1]$ in such a way, that all numbers of Ω have an equal chance to occur.

Whenever $\Omega \subset \mathbb{R}$, we can use $\mathbb{P}$ to define the ***cumulative distribution function*** (CDF)

$$F(\alpha) = \mathbb{P}((-\infty;\alpha] \cap \Omega)$$
$$\lim_{\alpha\to-\infty} F(\alpha) = 0$$
$$\lim_{\alpha\to\infty} F(\alpha) = 1$$

A function $f : \mathbb{R} \to \mathbb{R}^+$, such that

$$\int_{-\infty}^{\infty} f(t)\,\mathrm{d}t = 1$$

is called ***probability density function*** (PDF). With any PDF we can associate a CDF

$$F(\alpha) = \int_{-\infty}^{\alpha} f(t)\,\mathrm{d}t$$

and a probability measure $\mathbb{P}$ that associates the probability of the event $E \subset \mathbb{R}$ to occur with the area of all points underneath the function $f(t)$ whose t-values reside in E:

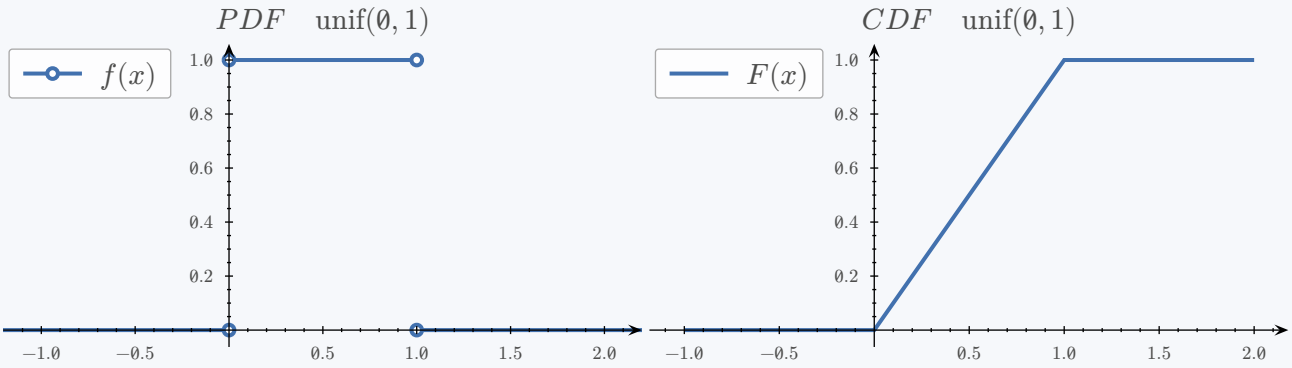
$$\mathbb{P}(E) = \int_{t\in E} f(t)\,\mathrm{d}t$$

Example 4: **Cumulative distribution function F and probability density function f for $\mathrm{unif}(a,b)$**

$$F : \begin{cases} \mathbb{R} \to [0;1] \\ x \mapsto \mathbb{1}_{[a;b]}(x) \cdot \frac{x-a}{b-a} + \mathbb{1}_{(b;\infty)}(x) \end{cases}$$

$$F'(x) = f(x) = \begin{cases} \frac{1}{b-a} & \text{if } a < x < b \\ 0 & \text{else} \end{cases} = \frac{\mathbb{1}_{(a;b)}(x)}{b-a} = \text{density for } X \sim \mathrm{unif}(a,b)$$

$$\mathbb{P}(X \le x) = \int_{-\infty}^{x} f(t)\,\mathrm{d}t = \int_{-\infty}^{x} \frac{\mathbb{1}_{(a;b)}(t)}{b-a}\,\mathrm{d}t \overset{a<x<b}{=} \int_a^x \frac{1}{a-b}\,\mathrm{d}t = \frac{x-a}{b-a}$$



2.5. DISTRIBUTIONS

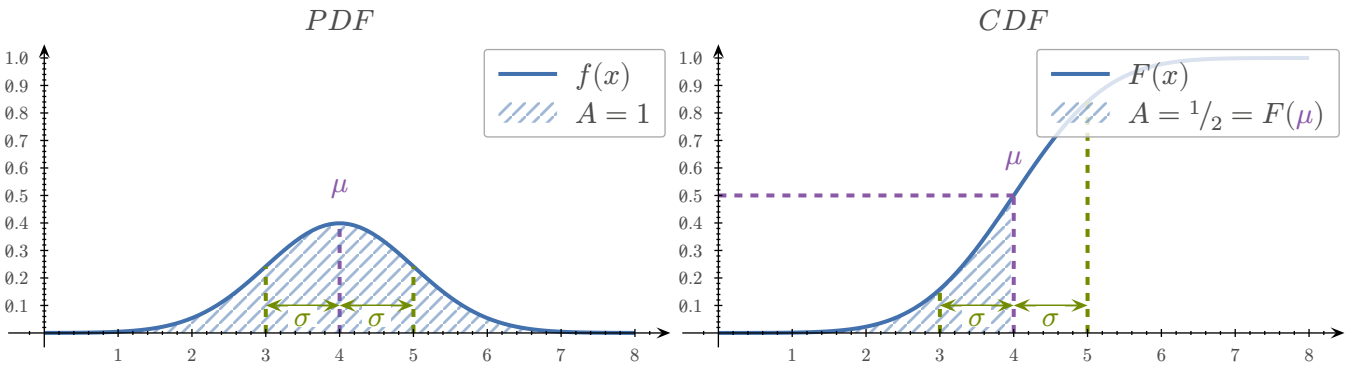
2.5.1. Univariate normal distribution

The probability density of the normal / Gaussian distribution is given by

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2\sigma^2}(x-\mu)^2}$$

in which μ and σ are parameters, denoted as mean value (μ) and standard deviation (σ).

$$\int_{\mu-\sigma}^{\mu+\sigma} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2\sigma^2}(x-\mu)^2} \, \mathrm{d}t \approx 0.68$$



The value of the CDF evaluated at x represents the area of the PDF from $-\infty$ to x, thus giving the probability of an experiment be in $(-\infty; x]$.

All normal distributions can be defined with the help of the so called ***standard normal distribution***

$$\varphi(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}x^2}$$

for which $\mu = 0$ and $\sigma = 1$. The general normal distribution can then be expressed in terms of the standard normal distribution using the formula

$$f(x|\mu, \sigma) = \frac{1}{\sigma}\varphi\left(\frac{x-\mu}{\sigma}\right)$$

The ***error function*** is defined through

$$\operatorname{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} \, \mathrm{d}t$$

or its Taylor series

$$\operatorname{erf}(x) = \frac{2}{\sqrt{\pi}} \sum_{k=0}^{\infty} \frac{(-1)^k}{k!(2k+1)} x^{2k+1}$$

and can be approximated through hyperbolic functions

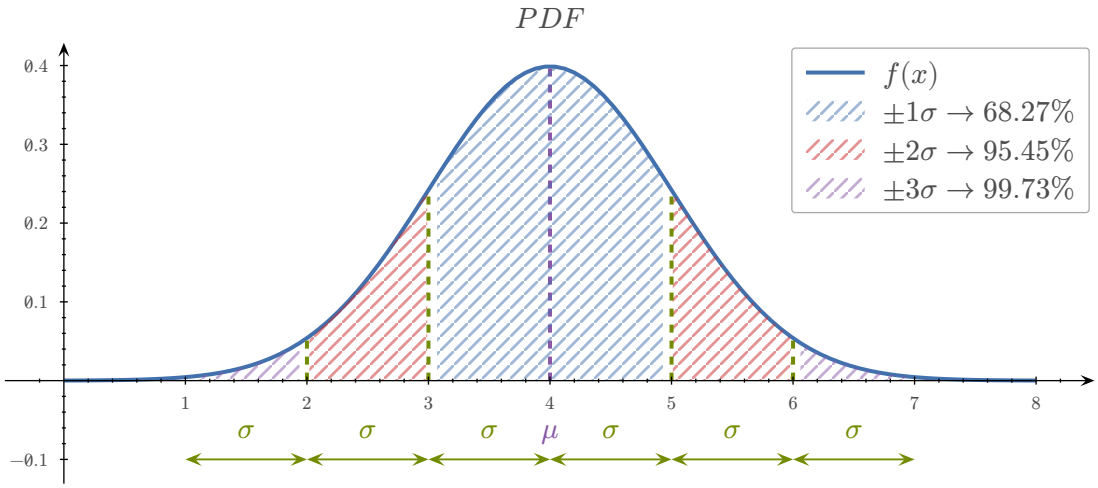
$$\operatorname{erf}(x) \approx \tanh\left(\frac{2}{\sqrt{\pi}}\left(x + \frac{11}{123}x^3\right)\right)$$

Given this function, one can show, that the CDF of the normal distribution is given by

$$F(x|\mu,\sigma) = \frac{1}{2}\left(1 + \operatorname{erf}\left(\frac{x-\mu}{\sqrt{2\sigma^2}}\right)\right)$$

The **68–95–99.7 rule**, also known as the empirical rule, for a normal distribution is a shorthand used to remember the percentage of values that lie within an interval estimate in a normal distribution: approximately 68%, 95%, and 99.7% of the values lie within one, two, and three standard deviations of the mean, respectively.

$$\mathbb{P}(\mu - 1\sigma \leq X \leq \mu + 1\sigma) \approx 68.27\%$$
$$\mathbb{P}(\mu - 2\sigma \leq X \leq \mu + 2\sigma) \approx 95.45\%$$
$$\mathbb{P}(\mu - 3\sigma \leq X \leq \mu + 3\sigma) \approx 99.73\%$$



,

2.5.2. Bernoulli distribution

A random variable X is said to be a Bernoulli random variable with parameter p, shown as $X \sim \text{Bernoulli}(p)$, if its PMF is given by

$$P_X(x) = \begin{cases} p & \text{for } x = 1 \\ 1-p & \text{for } x = 0 \\ 0 & \text{otherwise} \end{cases}$$

where $0 < p < 1$

A Bernoulli random variable is associated with a certain event A. If event A occurs, then $X = 1$; otherwise $X = 0$. For this reason the Bernoulli random variable is also called the indicator random variable. The indicator random variable I_A for an event A is defined by

$$I_A = \begin{cases} 1 & \text{if } A \text{ occurs} \\ 0 & \text{otherwise} \end{cases}$$

The indicator random variable for an event A has Bernoulli distribution with parameter $p = P(A)$, so we can write $I_A \sim \text{Bernoulli}(P(A))$.

TODO:
https://www.probabilitycourse.com/chapter3/3_1_5_special_discrete_distr.php

3. TREE DIAGRAMS AND SEQUENTIAL EVENTS

A ***probabilistic model*** that explains how data is generated (***generative model***). Generative Models connect domain knowledge with data.

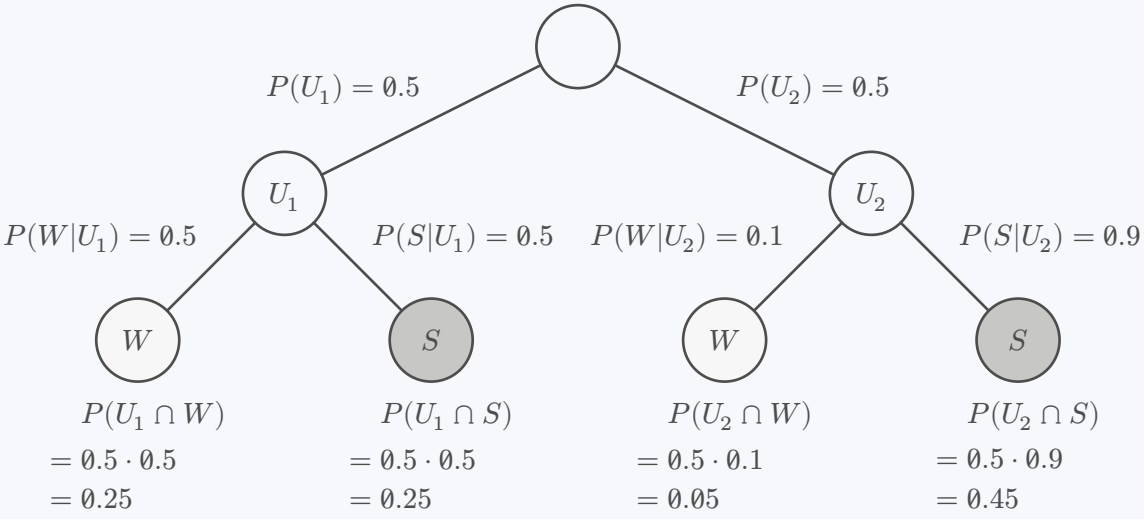
The 2-step model captures the causality of observable events.

– Step 1 lists all our ***hypotheses*** (mutually exclusive, exhaustive) that could have caused an outcome. (Example: U_1, U_2)
– Step 2 is an enumeration of all possible observable ***consequences*** (Example: W, S)

***Example** 5:* **Urns: Tree Diagram**

There are two urns, U_1 and U_2. Urn U_1 contains 5 silver and 5 white balls, and urn U_2 contains 9 silver and 1 white ball. The probability of drawing a ball from U_1 or U_2 is equal (50/50). What is the probability of drawing a white ball?

We can visualize the probabilities using a ***tree diagram***:



$$\begin{aligned} P(W|U_1) \wedge P(U_1) \vee P(W|U_2) \wedge P(U_2) &= P(W|U_1) \cdot P(U_1) + P(W|U_2) \cdot P(U_2) \\ &= 0.5 \cdot 0.5 + 0.5 \cdot 0.1 \\ &= 0.25 + 0.05 \\ &= 0.3 \\ &= P(W) \end{aligned}$$

The probability of drawing a white ball is therefore 30%, assuming that each urn is chosen with equal probability.

We can use Bayes' theorem to "go backwards" and start with the observation and deduce the cause.

$$P(\text{Hypothesis}|\text{Evidence}) = \frac{P(\text{Evidence}|\text{Hypothesis}) \cdot P(\text{Hypothesis})}{P(\text{Evidence})}$$

Bayesian inference is the statistical method that uses Bayes' theorem to update the probability of a hypothesis after seeing new data. It starts with a ***prior probability*** (belief before the evidence) and combines it with a likelihood to produce a ***posterior probability*** (updated belief).

***Example** 6:* **Urns: Bayesian inference**

We have seen that the choice of urns yield following probabilities (prior):

$$P(U_1) = 0.5$$
$$P(U_2) = 0.5$$

If we observe the drawing of a white ball, we can deduce which urn was more likely to be chosen from (posterior):

$$P(U_1|W) = \frac{P(W|U_1) \cdot P(U_1)}{P(W)} = \frac{0.5 \cdot 0.5}{0.3} = 0.8\overline{3}$$
$$P(U_2|W) = \frac{P(W|U_2) \cdot P(U_2)}{P(W)} = \frac{0.1 \cdot 0.5}{0.3} = 0.1\overline{6}$$

Thus the ***prior distribution*** of equally distributed probabilities changes to the ***posterior distribution*** favoring U_1 when provided with the observed data.

Given multiple observed results we can use ***Recursive Bayesian estimation*** (also called ***Bayesian filtering***), which is a general probabilistic approach for estimating an unknown PDF recursively.

$$p(x_0, ..., x_k, z_1, ..., z_k) = p(x_0) \prod_{i=1}^{k} p(z_i|x_i)p(x_i|x_{i-1})$$

***Example** 7:* **Urns: Recursive Bayesian estimation**

Given the observed outcomes of white, silver, white ($\{W, S, W\}$), we can construct our posterior distribution recursively. We start with the prior $P(U_1) = 0.5$ and $P(U_2) = 0.5$.

1) After the first white ball (W):

$$P(U_1|W) \approx 0.833$$
$$P(U_2|W) \approx 0.167$$

2) Using these as the new priors, we observe a silver ball (S):

$$P(U_1|W, S) = \frac{P(S|U_1) \cdot 0.833}{P(S)} = \frac{0.5 \cdot 0.833}{0.5 \cdot 0.833 + 0.9 \cdot 0.167} \approx 0.73$$
$$P(U_2|W, S) = \frac{P(S|U_2) \cdot 0.167}{P(S)} = \frac{0.9 \cdot 0.167}{0.5 \cdot 0.833 + 0.9 \cdot 0.167} \approx 0.27$$

3) Using these as the new priors, we observe another white ball (W):

$$P(U_1|W, S, W) = \frac{0.5 \cdot 0.73}{0.5 \cdot 0.73 + 0.1 \cdot 0.27} \approx 0.93$$
$$P(U_2|W, S, W) = \frac{0.1 \cdot 0.27}{0.5 \cdot 0.73 + 0.1 \cdot 0.27} \approx 0.07$$

The evidence of two white balls strongly overwhelms the single silver ball, significantly increasing our confidence that the balls have been drawn from U_1.

TODO:
Kahneman & Tversky

3.1. VOCAB FOR HUMAN EXPERIMENTS

Prevalence: Proportion of a population who have a specific characteristic in a given time period.

Specificity: Percentage of people who test negative for a specific condition among a group of people who do not have the condition. No test is 100% specific because some people who do not have the condition will test positive for it (false positive).

Sensitivity: How well a test can detect a specific condition in people who actually have condition. No test has 100% sensitivity because some people who have the condition will not be identified by the test (false-negative test result).

4. LINEAR REGRESSION

Linear Models are the most simple model to explain a relationship between “Input” and “Output”. Linear Regression is a standard method to find an optimal linear model

4.1. INGREDIENTS OF MACHINE LEARNING

4.1.1. The 4 main ones

Data: The dataset we are given plus the pre-processing pipe-line including cleansing, feature-engineering, data- augmentation etc.

Cost-Function (Loss): A formal (mathematical) expression for “good” and “bad”. Mean Squared Error (MSE) is commonly used.

Model: As simple as a two parameter, linear model $\hat{y}_i = mx_i + b$ or as complicated as a million-parameter Neural Network. Different tasks require different models (e.g. regression, decision-tree, …)

Optimization Procedure: An algorithm that changes the parameters of the model such that the cost-function is minimized. (e.g. Stochastic Gradient Descent (SGD), ADAM, …)

4.1.2. Others

Sounds like generic slop ngl

DevOps/MLOps Versioning. Reproducability: Building efficient pipe-lines is difficult. Following tool-specific recommendations and reference- implementations helps.

Visualization and evaluation of the learning Process: Learning curves, Performance measures, Tensorboard

Cross-Validation & Regularization: The goal is to train models that generalize well to unseen data (and get an estimate of the generalization error)

Baselines and Performance Monitoring: Implement a trivial baseline model. Every advanced model you train should beat it. Define what “better” means, and keep track of the performance evaluations. Deployed models need monitoring.

TODO:
CySec confusion matrix, ground truth, model prediction

TODO:
W4 S7 Different Machine Learning Techniques and applications

4.2. LINEAR REGRESSION

Linear Regression is a simple method to analyse data and quantify a relationship between variables. It has countless applications, most of them fall in one of two categories:

Interpretation: We want to understand if some input has an effect on the output. E.g. Is there a relationship between smoking cigaretts and the risk of lung cancer?

Prediction: Given a new "x" (e.g. Smoking rate), use the model to predict (estimate) the "y" (e.g Death rate)

In ML, linear regression falls into the category of ***supervised learning***: we are given both input data (X) and labels (Y). The ML algorithm learns a linear relationship between X and Y.

4.2.1. Residual sum of square

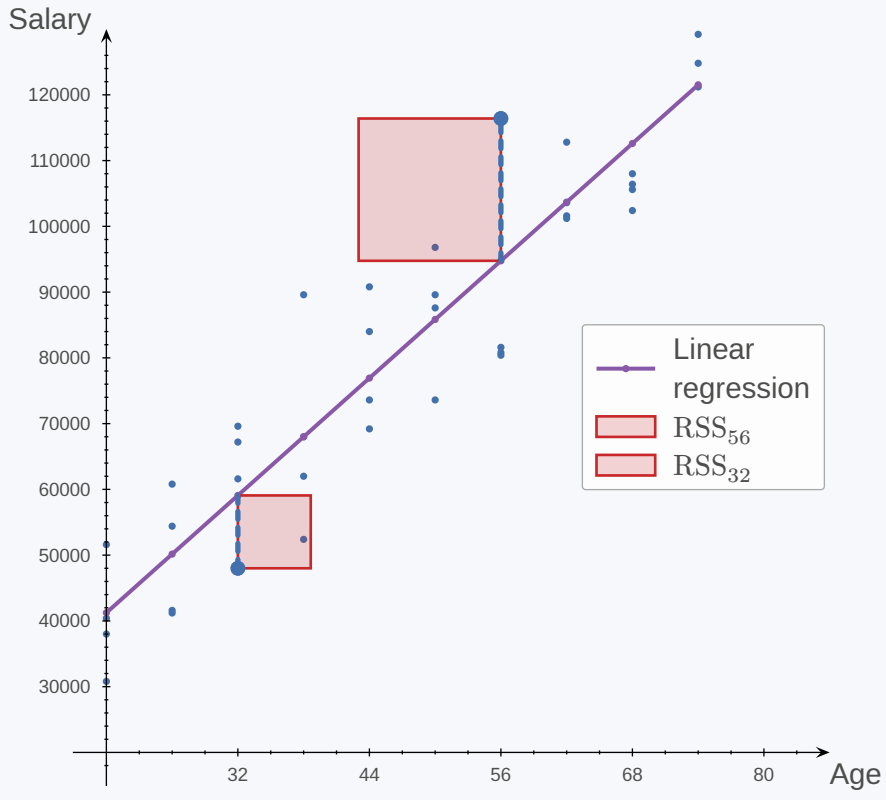
Residual sum of square (RSS) is a statistical method that helps identify the level of discrepancy in a dataset not predicted by a regression model. Thus, it measures the variance in the value of the observed data when compared to its predicted value as per the regression model.

$$RSS = \sum_{i=1}^{N} \underbrace{(y_i - f(x_i))^2}_{\text{Represented as red squares in the example}}$$

$$RSS(m,b) = \sum_{i=1}^{N} (y_i - (mx_i + b))^2 \geq 0, RSS : \mathbb{R}^2 \to \mathbb{R}$$

***Example** 8:* **Linear regression of salaries by age**

$$\mathrm{RSS}(m,b) = \sum_{i=1}^{N} ((mx_i + b) - y_i)^2 \geq 0, \mathrm{RSS} : \mathbb{R}^2 \to \mathbb{R}$$



4.2.2. Model

In ML, we use the term ***model*** for any mathematical function that "explains the data". More formally:

$$y_i \approx f(x_i)$$
$$y_i = f(x_i) + \varepsilon_i$$

Where ε_i is "unexplained noise". It is often assumed that ε_i follows a normal distribution.

The function f can be anything from a constant up to a multi-million parameter deep neural network. The goal of ML is to find the model which explains the data as good as possible. Instead of approximating y_i, we calculate an estimate $\hat{y}_i$ of the (usually unknown) y_i

$$\hat{y}_i \approx f(x_i)$$

In linear regression, we decide to only consider a linear relationship between the inputs and outputs. That is, before starting any learning algorithm, we constrain the model space to the family of linear functions. Such a design decision imposes an inductive bias on the model

In the simplest case, m and b are scalars and the linear model therefore has only two free parameters . The goal is to identify m and b for which the linear model "best explains the data"

$$\hat{y}_i = mx_i + b$$

m is usually called ***slope***, b the ***intercept*** (aka ***bias***). Often w for "weights" is used: $\hat{y}_i = w_1 x_i + w_\emptyset$

TODO:
Is y a linear function of x in this model?

$$y = ax^2 + bx + c$$

No. But the model is linear in the parameters a, b, c . Linear regression only needs this second property.

We can apply linear regression to arbitrarily complex models, as long as the model is linear in the unknowns.

4.2.3. Mean squared error

Mean squared error (MSE) is the average of the squares of the errors. It is usually divided by 2, which does not change the location of the minimum. It cancels the factor 2 that appears when we take the derivative of the square. This simplifies the gradient.

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

where

$$e_i = y_i - \hat{y}_i$$
$$\hat{y}_i = mx_i + b$$

and e_i (the difference) is called the ***residual***.

4.2.4. Correlation and causality

Correlation refers to the degree to which a pair of variables are linearly related. It can be quantified using the ***Pearson correlation coefficient***. Showing correlation is not hard, convincingly identifying causality on the other hand is.

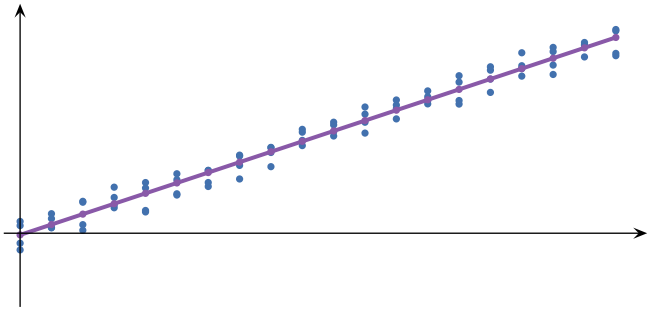
4.2.4.1. Pearson correlation coefficient

The Pearson correlation coefficient (PCC) is a correlation coefficient that measures linear correlation between two sets of data. It is the ratio between the covariance of two variables and the product of their standard deviations. Thus, it is essentially a normalized measurement of the covariance, such that the result always has a value between -1 and 1.

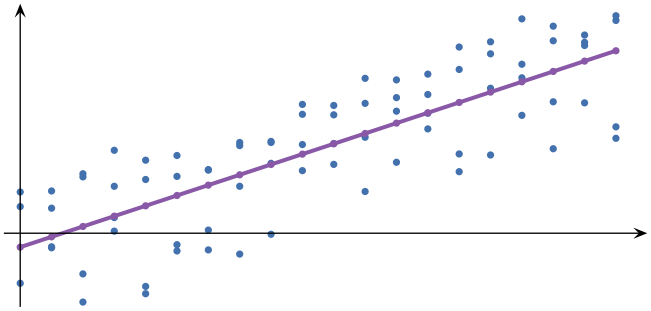
$$\mathrm{PCC}_{X,Y} = \frac{\sum_{i=1}^{N}(x_i - \overline{x})(y_i - \overline{y})}{\sqrt{\sum_{i=1}^{N}(x_i - \overline{x})^2}\sqrt{\sum_{i=1}^{N}(y_i - \overline{y})^2}}$$

- 1 shows a perfect positive correlation where both variables increase / decrease together at a constant rate
- -1 shows a perfect negative correlation where one variable increases as the other decreases proportionally
- 0 shows no linear relationship, meaning changes in one variable do not predict changes in the other

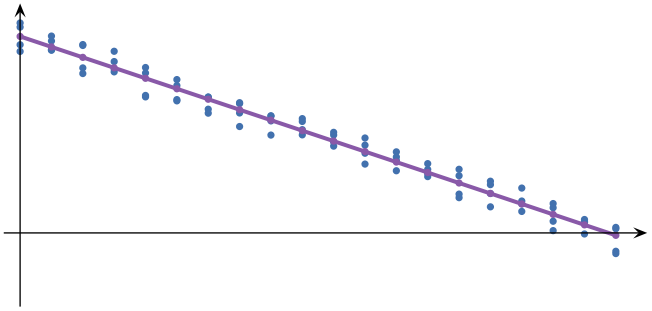
Strong positive correlation $r \approx 1$



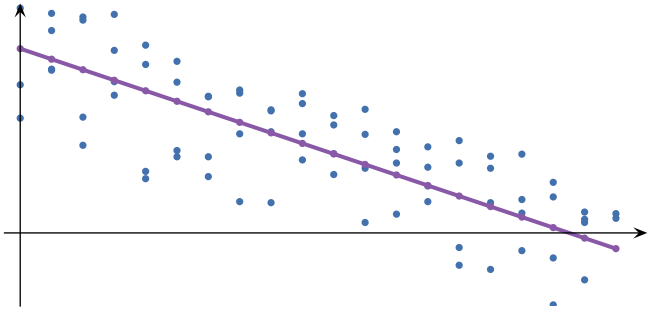
Weak positive correlation $r \approx 0.5$



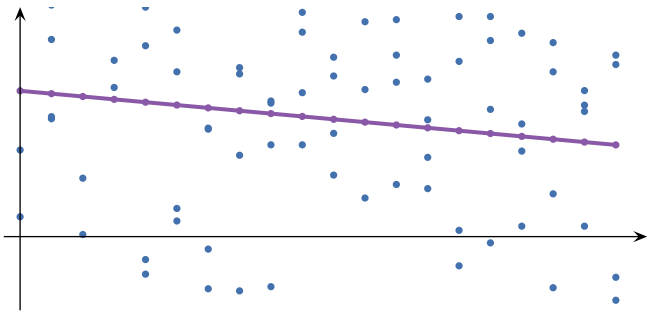
Strong negative correlation $r \approx -1$



Weak negative correlation $r \approx -0.5$



Weak / no correlation $r \approx 0$



4.2.5. Multiple linear regression

Same concept but with more factors and weights:

$$\hat{y}_i = \beta_1 x_{1i} + \beta_2 x_{2i} + ... + \beta_N x_{Ni} + \alpha_i$$

where

- y_i is an observed/measured quantity. Example: blood pressure
- $x_{1i}...x_{Ni}$ are "factors". Examples: $x_{1i} : \mathrm{age}, x_{2i} : \mathrm{weight}, x_{3i} : \mathrm{sex}, x_{4i} : \mathrm{drug_dosage}, ...$
- $\beta_1...\beta_N$ are weights. How much does each factor x explain the outcome y?

4.2.5.1. Matrix notation

The dataset can be written as N points (x, y), where x is a vector with p features.

$$\hat{y} = X\beta + \alpha$$

where

$$X = \begin{pmatrix} x_{11} & x_{12} & \dots & x_{1p} \\ x_{21} & x_{22} & \dots & x_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ x_{N1} & x_{N2} & \dots & x_{Np} \end{pmatrix}, \quad \beta = \begin{pmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_N \end{pmatrix}, \quad y = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{pmatrix}$$

TODO:
W4 S18,20

5. OPTIMIZATION

Optimization is the problem of finding a set of inputs to an objective function that results in a maximum or minimum function evaluation. In our applications the objective is to minimize the loss function.

5.1. PARTIAL DERIVATIVE AND GRADIENT

Let $f : D \to \mathbb{R}$ be an arbitrary real valued function and $x \in D \subset \mathbb{R}^n$ be an arbitrary vector in the domain of f. We consider the domain D of f to be sufficiently large for doing calculus at x, if for every vector $v \in \mathbb{R}^n$ there is a small interval $(-\varepsilon; \varepsilon), \varepsilon > \emptyset$, such that for all $t \in (-\varepsilon; \varepsilon)$ the vector $x + t \cdot v \in D$. This condition guarantees that any curve that hits the domain of f, remains there for at least a short period of time, and is one of the properties that needs to be satisfied in order to denote f as a hyper-surface. The other property that is required for f to be a hyper-surface, is that f is continuous at any point of its domain.

Definition 2: **Partial derivative**

The ***partial derivative*** of f with respect to the i-th coordinate function is defined through

$$\frac{\partial}{\partial x_i} f(x) = \lim_{t \to \emptyset} \frac{f(x + t \cdot e_i) - f(x)}{t}$$

Definition 3: **Gradient**

The ***gradient*** of f is an n-dimensional row vector, whose components correspond to the various **partial derivatives**.

$$\nabla f(x) = \left(\frac{\partial}{\partial x_1} f(x), \frac{\partial}{\partial x_2} f(x), ..., \frac{\partial}{\partial x_n} f(x) \right)$$

Example 9: $f : \begin{cases} \mathbb{R} \times \mathbb{R} \to \mathbb{R} \\ (x, y) \mapsto x^2 y^3 \end{cases}$

$$f'_x(x,y) = \frac{\partial f}{\partial x} = 2x \cdot y^3$$
$$f'_y(x,y) = \frac{\partial f}{\partial y} = x^2 \cdot 3y^2$$
$$\nabla f(x,y) = \left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right) = (2x \cdot y^3, x^2 \cdot 3y^2)$$

5.2. GRADIENT DESCENT

Goal: $\nabla f(x) \approx 0$, where $f(x)$ in ML is usually the loss function.

Iterative approach:

$$x_{i+1} = x_i - \gamma \cdot \nabla f(x_i)$$

where $\gamma > 0$ is the ***step size*** or ***learning rate*** of gradient descent. Large values of γ allow us to move quickly at the price that we may jump over local minimum points with out notice.

We also need a ***termination criterion*** through which we control the precision $\varepsilon > 0$ of the approximation:

$$\left|x_{i+1} - x_i\right| < \varepsilon$$

or equivalently

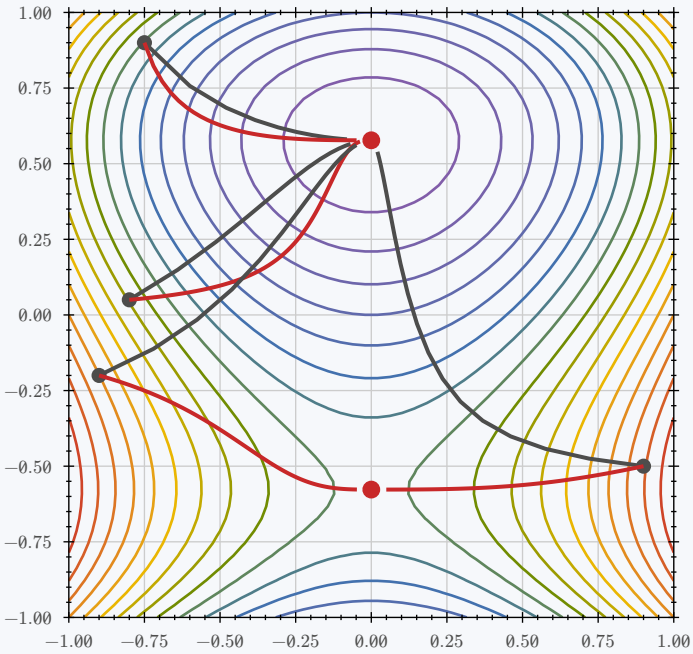
$$|\nabla f(x_i)| < \frac{\varepsilon}{\gamma}$$

leading to gradient descent terminating at a point where $\nabla f(x) \approx 0$. The algorithm is guaranteed to stop at a stationary point, but there is **no guarantee that the stationary point is in fact a local minimum**.

Comparison to Newton's method:

$$x_{i+1} = x_i - \gamma \cdot \underbrace{\left(H_{f(x_i)}\right)^{-1}}_{\text{Too expensive in ML!}} \cdot \nabla f(x_i)$$

***Example** 10:*



$$f(x,y) = y^3 - y + x^2 - \frac{1}{2}$$

Black lines starting from the black points lead to the local minimum, calculated using gradient descent. Red lines are calculated using Newton's method.

The parameters chosen are as follows:

Newton's method: $\gamma = 0.01, \varepsilon = 0.001$
Gradient descent: $\gamma = 0.1, \varepsilon = 0.01$

Procedures for finding good values for ε, γ can include:
– Conjugate gradient method
– Stochastic gradient descent

The standard gradient descent is also called ***batch gradient descent***.

5.2.1. Stochastic Gradient Descent (SGD)

Batch Gradient Descent computes the gradient of the cost function using the entire training dataset for each iteration. This approach ensures that the computed gradient is precise, but it can be computationally expensive when dealing with very large datasets ($O(n)$).

$$\begin{aligned} x_{i+1} &= x_i - \gamma \cdot \nabla f(x_i) \\ &= x_i - \gamma \frac{1}{n} \sum_{j=1}^{n} \nabla f_j(x_i) \end{aligned}$$

The idea behind SGD is that we do not need the exact gradient to find a trajectory toward the minimum. Instead, at each iteration we can randomly pick a single training example at j and use it to calculate an approximation of the gradient:

$$x_{i+1} = x_i - \gamma \cdot \nabla f_{j_i}(x_i)$$

Where j_i is the index of the randomly chosen example at iteration i. This brings our complexity down to $O(1)$. Due to the estimate being noisy (as it's based on one sample) we can average m samples to get $O(m)$ with a less noisy result. This method is called ***mini-batch SGD***.

$$x_{i+1} = x_i - \gamma \frac{1}{m} \sum_{j \in m_i} \nabla f_j(x_i)$$

TODO:
batch diagrams

***Example** 11:* **Linear regression**

Let

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

be our loss function. We can now optimize a, b using gradient descent:

$$\begin{aligned}
\begin{pmatrix} a \\ b \end{pmatrix}_{i+1} &= \begin{pmatrix} a \\ b \end{pmatrix}_i - \gamma \cdot \left.\begin{pmatrix} \frac{\partial l}{\partial a} \\ \frac{\partial l}{\partial b} \end{pmatrix}\right|_{\begin{pmatrix} a \\ b \end{pmatrix}_i} \\
&= \begin{pmatrix} a \\ b \end{pmatrix}_i - \gamma \cdot \left.\begin{pmatrix} \frac{1}{N}\sum_{j=1}^{N}(y_j - (a\cdot x_j + b))(-x_j) \\ \frac{1}{N}\sum_{j=1}^{N}(y_j - (a\cdot x_j + b))(-1) \end{pmatrix}\right|_{\begin{pmatrix} a \\ b \end{pmatrix}_i} \\
&= \begin{pmatrix} a \\ b \end{pmatrix}_i + \gamma \cdot \frac{1}{N}\sum_{j=1}^{N}(y_j - (a_i \cdot x_j + b_i))\begin{pmatrix} x_j \\ 1 \end{pmatrix} \\
&= w_i + \frac{\gamma}{N}\sum_{j=1}^{N}(y_j - w_i^\top \tilde{x}_j)\tilde{x}_j
\end{aligned}$$

where

$$w_i = \begin{pmatrix} a_i \\ b_i \end{pmatrix} \quad \tilde{x}_j = \begin{pmatrix} x_j \\ 1 \end{pmatrix}$$

For the stochastic gradient descent we just choose a random $j \in \{1, 2, ..., N\}$ for each i:

$$w_{i+1} = w_i + (y_j - w_i^\top \tilde{x}_j)\tilde{x}_j$$

5.2.1.1. Annealing

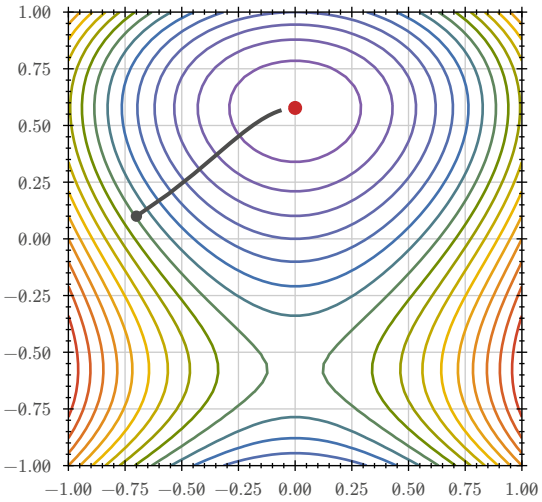
For a fixed learning rate γ, SGD does not converge. Instead, the algorithm keeps fluctuating around the minimum. One can set γ to a very small value, in which case the fluctuations are smaller and SGD stays "close enough" to the minimum during the "late iterations".

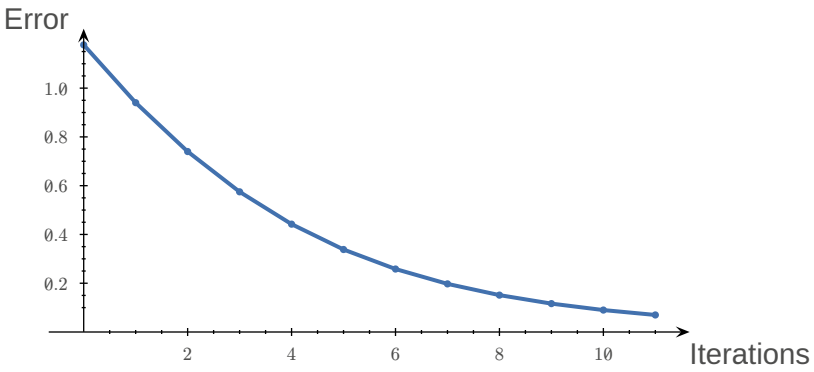
During the "early iterations", the situation is different: There is no benefit staying close to the optimal trajectory. As long as SGD moves roughly in the direction toward the minimum, fluctuations (even relatively large ones) around the "optimal" trajectory are acceptable. One would therefore choose a large γ in order to improve the speed of the algorithm.

Annealing solves this apparent contradiction by adapting the learning rate. It starts with a large γ and reduces it over time. There are many options (so called ***schedules***) how one can reduce the learning rate over time. Here we decay the learning rate by multiplying it at each iteration by an annealing factor.

Without annealing:

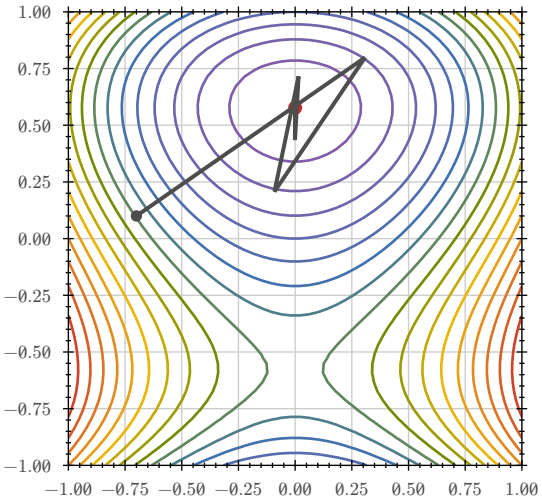
$$\gamma = 0.1$$

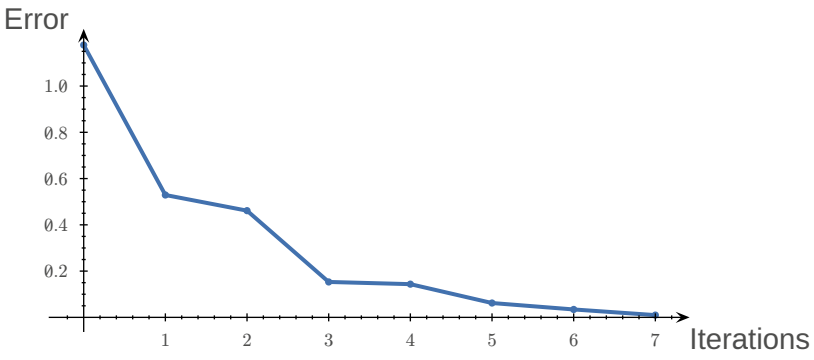




With annealing:

$$\gamma = 0.8, \quad a(\gamma) = \gamma \cdot \frac{9}{10}$$





6. NATURAL LANGUAGE PROCESSING (NLP)

***Example** 12:* **NLP task, four ingredients**

Task: Text Analysis, Sentiment Classification → Supervised learning. Binary classification

Example (training data): “This is the worst movie I’ve ever seen” → label: 0 (=negative)

Ingredient	*Description*
Data & Preprocessing	Labeled sentences (text) from IMDB
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 (a variant of SGD)

6.1. LARGE LANGUAGE MODEL (LLM)

6.1.1. How a LLM generates text

The output is generated sequentially: the output is generated word-by-word (or token-by-token). Sometimes multiple sequences of next-tokens are generated (beam-search). The output is a probability distribution over a dictionary, according to which the next word is sampled. Each output is added to the input-context (this is called ***auto-regressive***).

6.1.2. Where the LLM stores its "knowledge"

In the weights (aka "parametric memory"), data is not stored explicitly (no "database"). This is a key limitation of LLMs, because one can't add data easily, thus being never up to date.

6.2. REPRESENTATION OF WORDS

One can use vectors to represent words based on their meaning.

One-hot representation: Does not incorporate meaning

Distributed representation: vectors that capture (at least partially) the semantics of a word

6.2.1. One-hot representation

A one-hot vector is a vector with a single 1-value and all others set to 0. One can count the number of different words and define one unique one-hot-vector per word. This bears the following disadvantages:

Very high dimensional vector space: n words have to be represented with n-dimensional vectors.

Sparse represesntation: The $n-1$ zeroes are memory-inefficient and ML algorithms cannot efficiently extract meaning from them.

No generalization: Using one-hot vectors, all words are completely unrelated to each other (because the vectors share no entries), and therefore we cannot generalize knowledge.

6.2.2. Indexing

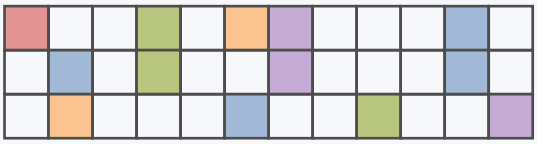
Make a list of words and use the index to represent each word. Indexing is the "dense equivalent" of one-hot encoding. Indexes are not more useful than one-hot vectors, but it's often used as a preprocessing step. In LLMs, this index is called "token ID".

6.2.3. Distributed Representation (dense vectors)

According to the distributional hypothesis, words that occur in similar contexts (with the same neighboring words), tend to have similar meanings. Distributed representations can be learned.

***Example** 13:*

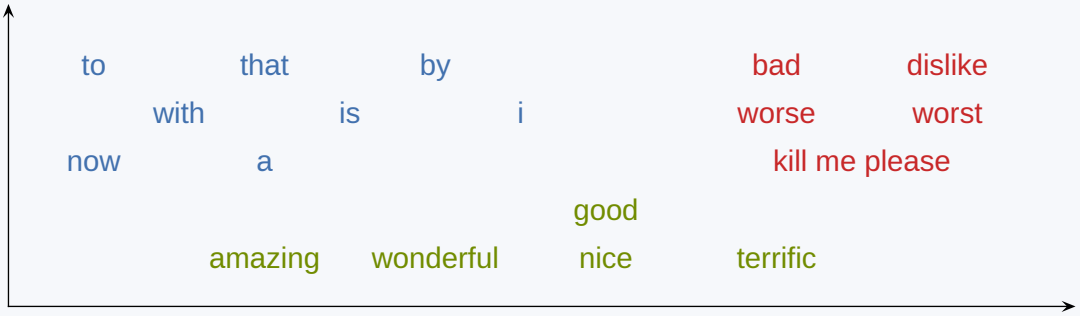
King
$-$ Man
$+$ Woman



= Queen 

Word2Vec is a popular NLP algorithm to “embed” words (or longer strings of characters) in a vector space. A mathematical function maps words to output vectors, where similar semantics are represented by vectors in similar regions of the vector-space. i

***Example** 14:*



The output vectors have fewer dimensions (but still 100s to 1000s) than the corresponding one-hot vectors. ML algorithms can learn this function. In neural networks this function is implemented in a so called ***Embedding Layer***.

This representation provides benefits such as the dot-product being a measure of similarity and the ability to add/subtract vectors.

TODO:
dot-product W6 S22,23

Cosine-distance:

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

- $\cos(\theta) = 1$ when the angle $\theta = 0°$ (the vectors point in exactly the same direction)
- $\cos(\theta) = -1$ when the angle $\theta = 180°$ (the vectors point in opposite directions)
- $\cos(\theta) = 0$ when the angle $\theta = 90°$ (the vectors are orthogonal to each other)

TODO:
watch 3b1b and self-study because some professors are too lazy to provide the materials themselves ig

7. PROJECT

TODO:
put into appropriate sections

Power law: A relationship where one quantity grows as a power of another. E.g.

$$q = a \cdot x^b \cdot y^c$$
$$\log q = \log a + b \log x + c \log y$$

Extrapolation: A model extrapolates when it predicts for inputs outside the range it saw during training.

Random forest model: An ensemble model: it builds many decision trees and combines their predictions. It can capture nonlinear relationships.

> ***TODO:***
> research

Standardizing features: Transforming each feature so it has mean 0 and standard deviation 1. This makes feature weights easier to compare. If features are standardized, "one unit" means one standard deviation.

Folds: Data partitions

Cross-validation: Repeatedly evaluates a model on held-out data to estimate how well it generalizes to unseen data.

Log-loss: Measures how wrong the probabilities are. It grows when probabilities are badly calibrated. kinda wrong/too specific

K-fold cross-validated RMSE: Square-root of the average squared prediction error obtained when a model is evaluated using k-fold cross-validation.

1) Split the data into k approximately equal "folds"
2) Iterate k times: each iteration treats one fold as the test set and the remaining k – 1 folds as the training set
3) Fit the model on the training folds, generate predictions for the held-out test fold, and compute the MSE (mean squared error) for that fold
4) Average the MSEs across all k folds
5) Take the square root to obtain the cross-validated RMSE

The resulting value estimates how well the model is expected to perform on unseen data, while reducing variance compared to a single train-test split.

Mean log-loss over 5 cross-validation folds: Split the data into 5 folds, train on 4 folds and test on the remaining fold, repeat 5 times, compute log-loss for each fold, then average them. This gives a more reliable estimate of model performance than a single train/test split.

Calibrating probabilities: A model is calibrated if its predicted probabilities match actual frequencies. For example, among data predicted to have $p \approx 0.3$, about 30% should actually have that feature

TPR: True Positive Rate, also called ***sensitivity*** or ***recall***. "How many actual positives did the model correctly identify?"

$$\mathrm{TPR} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}$$

FPR: False Positive Rate, also called ***fallout***. "How many actual negatives did the model incorrectly label as positive?"

$$\mathrm{FPR} = \frac{\mathrm{FP}}{\mathrm{FP} + \mathrm{TN}}$$

ROC: Receiver Operating Characteristic. A ROC curve plots TPR against FPR and shows how well a model separates two classes as you vary the decision threshold. A perfect classifier has a curve going straight up and then across: $\mathrm{TPR} = 1 \land \mathrm{FPR} = 0$

AUC: Area Under the (ROC) Curve. It measures ranking ability: "Does the model put positive cases above negative cases?"

$\mathrm{AUC} = 1$: perfect separation
$\mathrm{AUC} = 0.5$: no better than random guessing
$\mathrm{AUC} < 0.5$: worse than random

AUC tells you how well the model ranks risk. Calibration tells you whether the predicted probabilities are numerically correct.