~~Data and metadata.~~

Data is facts and statistics collected together for reference or analysis. It can take several forms such as a table, a signal, a text, a sequence, a map, an image or a video.

Metadata is information about the features and instances in data, such as their name, description and units of measurement.

Feature as a mapping.

In table data the columns are referred to as *features*. Features can be thought of as *mappings* of entities in data to sets of values. For example, they can map species of flowering plants to their family or their flower type, or companies to the sector they work in.

Feature as a probabilistic distribution.

In table data the columns are referred to as *features*. Features can thought of as *random distributions* with set parameters and the corresponding data as a sample from said distribution. Therefore, features can be described by their relative histogram (frequencies of different feature values in data) or, in case of quantitative features, density functions.

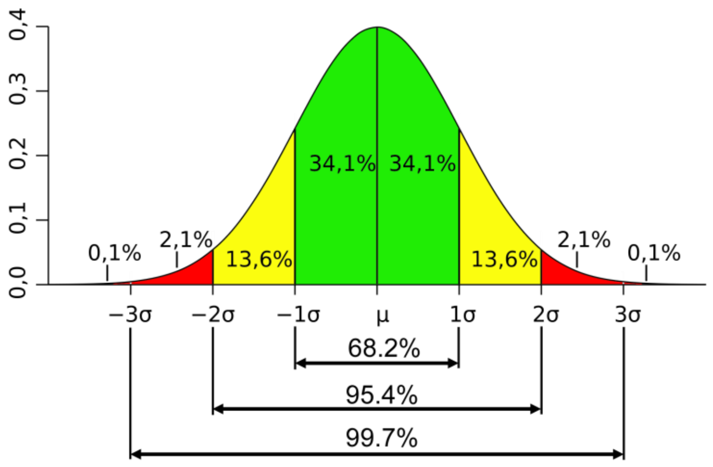
Popular density functions: Gaussian, Power Law, Uniform.

Gaussian or *normal* distribution is a continuous distribution given by density function

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right), x \in \mathbb{R},$$

where μ is *mean*, σ is *standard deviation* and σ^2 is *variance*. The most popular distribution in this class is standard normal distribution, where the mean is zero and the standard deviation is one.

Normal distribution works is such a way that 68% of all values are within σ of mean, 95% -- within 2σ and 99.7% within 3σ.

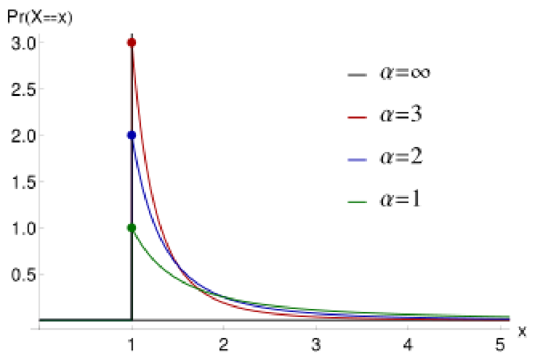


Power law is a continuous distribution given by following density:

$$f(x) = cx^{-\alpha}, c \in \mathbb{R},$$

where α is steepness of the curve. This distribution is invariable relative to scale c, since

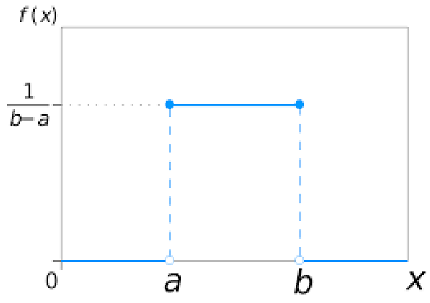
$$f(cx) = c^{1-\alpha}x^{-\alpha} \propto f(x).$$



Uniform distribution is a continuous distribution given by following density:

$$f(x) = \frac{1}{b-a}, x \in [a, b].$$

Its mean is $\frac{b+a}{2}$ and its variance is $\frac{(b-a)^2}{12}$.



What is a quantitative feature?

A feature is called quantitative if an arithmetic average of it makes sense.

Nominal feature.

A feature is called nominal if its values are disjunctive categories for which the only comparison that makes sense is “equal-not equal”.

Does it make sense to consider a binary feature as a quantitative one?

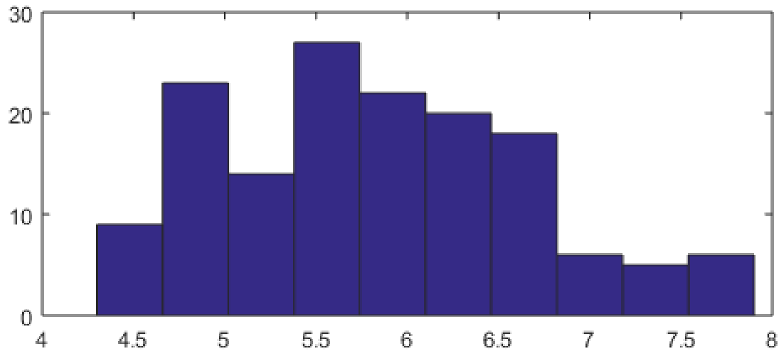
It does, since by encoding “true” or “yes” as 1 and “no” or “false” as 0 by computing the average of the encoded data one can find the proportion of entities for which the value of the binary feature is “true” or “yes”.

Enveloping a nominal feature into a set of binary features.

A nominal feature can be binarized by encoding it as a set of binary features corresponding to each unique value of given feature. For example: economy (agriculture, industry, services) translates to agriculture (yes/no), industry (yes/no), services (yes/no).

~~Histogram and bin size.~~

Histogram is a representation of data that divides feature's range of values into several bins, and shows the number of objects falling into bin as the bar height. A histogram with 10 bins is given on a figure below:



~~Categorization of quantitative features.~~

Mean, median, variance, standard deviation.

The *mean* or *sample mean* is given by

$$\bar{x} = \sum_{i=1}^{n} \frac{x_i}{n},$$

where x_i are data entries and n is the number of them. However, it can also be defined as mathematical expectation

$$E\xi = \int_{-\infty}^{+\infty} x f(x) dx$$

The *median* is the value that is bigger than or equal to 50% of all the feature values. Hence, it is also called 50% percentile. In case of finite data this is the middle element in sorted list.

The *standard deviation* is given by following expression

$$\sigma = \sqrt{E(\xi - E\xi)^2}.$$

Here the expression under the root sign is called *variance*.

Scatter plot.

A scatter plot is a graphical representation of data which uses values of a pair of features as a set of points in 2D Cartesian coordinates.

The problem of linear regression and its solution.

The linear regression is a model that estimates a relation between several variables as a linear one. Thus, the problem of linear regression can be stated as “how to find coefficients a and b which minimize the errors e_i in expression $y_i = a \cdot x_i + b + e_i$ for entities $i = 1, \dots, N$”. This problem can be mathematically represented as

$$L(a,b) = \sum_{i=1}^{N} (y_i - a \cdot x_i - b)^2 \to min.$$

This can be interpreted as minimization of squares of errors or least squares linear regression problem.

Since $L(a,b)$ is a second degree polynomial, first-order optimality conditions are given as:

$$\begin{cases} \dfrac{\partial L}{\partial a} = 2\displaystyle\sum_{i=1}^{N} (y_i - ax_i - b)(-x_i) = 0, \\ \dfrac{\partial L}{\partial b} = 2\displaystyle\sum_{i=1}^{N} (y_i - ax_i - b)(-1) = 0. \end{cases}$$

Solving the second equation for b we get that

$$b = \bar{y} - a \cdot \bar{x}.$$

Here, $\bar{x}, \bar{y}$ are means of x and y respectively. Since b is now a known value, we can solve the first equation for a and get

$$a = \frac{\sum_{i=1}^{N}(y_i - \bar{y})\, x_i}{\sum_{i=1}^{N}(x_i - \bar{x})x_i}.$$

And since

$$\sum_{i=1}^{N}(x_i - \bar{x}) = \sum_{i=1}^{N}(y_i - \bar{y}) = 0,$$

We can subtract $\sum_{i=1}^{N}(y_i - \bar{y})\,\bar{x}$ from nominator and $\sum_{i=1}^{N}(x_i - \bar{x})\,\bar{x}$ from denominator and divide them both by N to get

$$a = \frac{\frac{1}{N}\sum_{i=1}^{N}(y_i - \bar{y})\,(x_i - \bar{x})}{\frac{1}{N}\sum_{i=1}^{N}(x_i - \bar{x})(x_i - \bar{x})}.$$

Since $\rho_{xy} = \frac{\frac{1}{N}\sum_{i=1}^{N}(y_i - \bar{y})(x_i - \bar{x})}{\sigma_x \sigma_y}$, we finally get that

$$a = \rho_{xy}\frac{\sigma_y}{\sigma_x},$$

where ρ_{xy} is correlation coefficient between x and y.

After that, substituting a and b in $L(a,b)$ we get that

$$L(a,b) = \sum_{i=1}^{N}\left(y_i - \rho_{xy}\frac{\sigma_y}{\sigma_x}x_i - \bar{y} + \rho_{xy}\frac{\sigma_y}{\sigma_x}\bar{x}\right)^2 = \sum_{i=1}^{N}\left((y_i - \bar{y}) - \rho_{xy}\frac{\sigma_y}{\sigma_x}(x_i - \bar{x})\right)^2,$$

$$L(a,b) = \sum_{i=1}^{N}(y_i - \bar{y})^2 - 2\rho_{xy}\frac{\sigma_y}{\sigma_x}\sum_{i=1}^{N}(x_i - \bar{x})(y_i - \bar{y}) + \rho_{xy}^2\frac{\sigma_y^2}{\sigma_x^2}\sum_{i=1}^{N}(x_i - \bar{x})^2,$$

$$L(a,b) = N\sigma_y^2 + N\rho_{xy}^2\frac{\sigma_y^2}{\sigma_x^2}\sigma_x^2 - 2N\rho_{xy}^2\sigma_y^2 = N\sigma_y^2\left(1 - \rho_{xy}^2\right).$$

Relative error of linear regression in the perspectives of Data Analysis and Machine Learning.

Error is the difference between the observed value and its estimate, while *relative error* is the error related to the true value.

Different things can be considered true values: from ML or scientific perspective theoretical data is true, since the observed values may be subjects to noise, while from DA point of view the observed values are true because all other values are an assumption.

Therefore, the errors from ML and DA perspectives are given by

$$e_{ML} = \frac{|y - y_{pred}|}{|y_{pred}|}, e_{DA} = \frac{|y - y_{pred}|}{|y|}.$$

Correlation coefficient in the linear regression problem; its properties.

Correlation coefficient is given by

$$\rho_{xy} = \frac{\frac{1}{N}\sum_{i=1}^{N}(y_i - \bar{y})\,(x_i - \bar{x})}{\sigma_x\sigma_y},$$

where σ_x and σ_y are standard deviations of *x* and *y* respectively. It has following properties:

1° The determinacy coefficient ρ^2 is within an interval $[0{,}1]$, while the correlation coefficient ρ, is within $[-1, +1]$.

2° Coefficient ρ is 1 or -1 if and only if equation $y_i = ax_i + b$ is true for each $i = 1,2, \dots, N$ with no errors.

3° Coefficient ρ is 0 if and only if the slope $a = 0$, since $a = \rho_{xy}\frac{\sigma_y}{\sigma_x}$.

4° The sign of ρ is the sign of the slope a; therefore, *x* and *y* are related positively in case $\rho > 0$, and negatively, if $\rho < 0$.

Determinacy coefficient, its meaning.

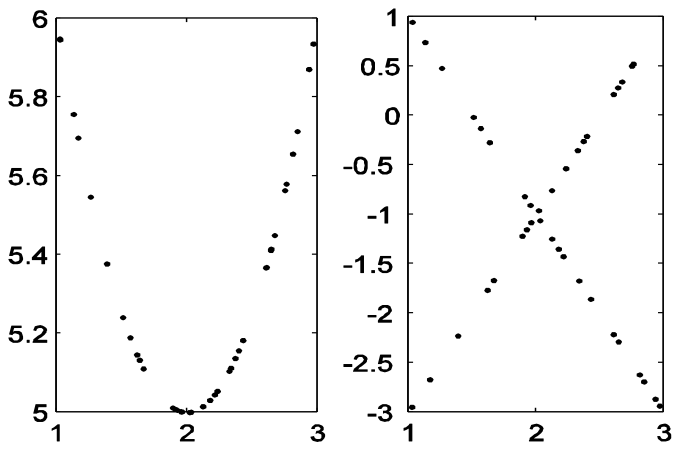
Determinacy or R^2*-coefficient* for linear regression is given by

$$R^2 = \rho^2 = \left(\frac{\frac{1}{N}\sum_{i=1}^{N}(x_i - \bar{x})(y_i - \bar{y})}{\sigma_x\sigma_y}\right)^2.$$

This coefficient represents the proportion of the variance σ_y^2 taken into account by the linear regression of y over x:

$$L(a,b) = N\sigma_y^2\left(1 - \rho_{xy}^2\right).$$

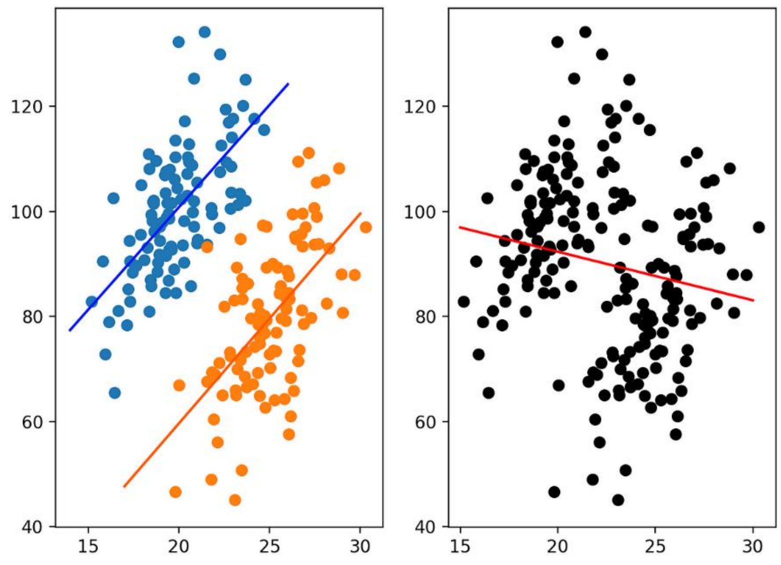
Give an example of interrelated features with 0 correlation coefficient.



Several cases of related features have zero correlation are given on this figure.

False correlation, Simpson’s paradox.

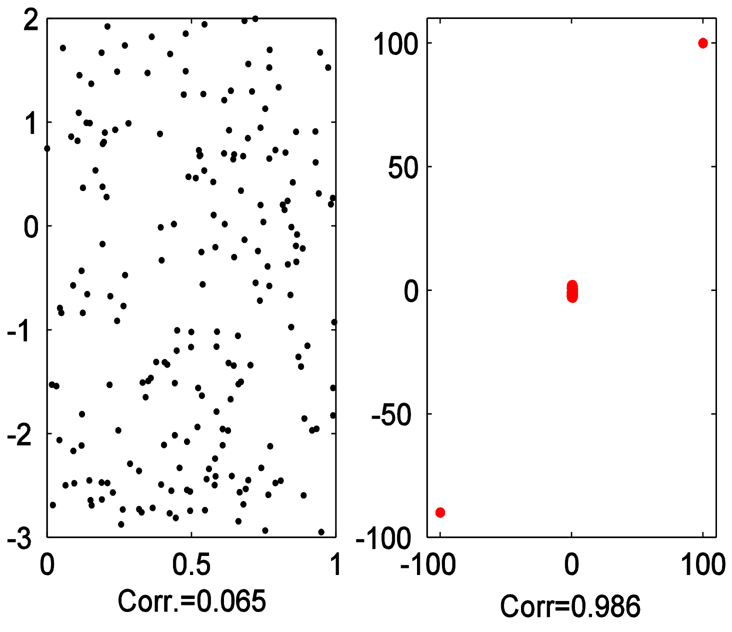
Simpson's paradox is a phenomenon in data analysis in which a trend appears in several groups of data but disappears or reverses when the groups are combined.



The figure above exemplifies the case with false correlation.

How can one increase the coefficient of correlation by adding just one or two objects?

We can manipulate the correlation coefficient by adding outliers to the data in the direction we want it to change to.



Correlation coefficient in the probabilistic perspective.

Correlation coefficient is a measure of extent of a linear relation between *x* and *y*. Moreover, since the density function bivariate normal distribution is

$$f(x,y,\mu_x,\mu_y,\Sigma) = \\ = \frac{1}{2\pi\sigma_x\sigma_y\sqrt{1-\rho^2}}\exp\left\{-\frac{1}{2(1-\rho^2)}\left(\left(\frac{x-\mu_x}{\sigma_x}\right)^2 - 2\rho\left(\frac{x-\mu_x}{\sigma_x}\right)\left(\frac{y-\mu_y}{\sigma_y}\right) + \left(\frac{y-\mu_y}{\sigma_y}\right)^2\right)\right\},$$

where μ_x, μ_y and σ_x, σ_y are means and standard deviations for *x* and y, the correlation coefficient is a sample-based estimate of the parameter ρ in the Gaussian density function under the assumption of independent random sampling.

Multivariate linear regression and the orthogonal projector.

Let us consider a model that approximates multivariate function $u(x) = f(x_1, x_2, \dots, x_p)$ with a hyperplane defined by function

$$\hat{u}\left(x_1, x_2, \dots, x_p, w\right) = w_0 \cdot x_0 + w_1 \cdot x_1 + w_2 \cdot x_2 + \cdots + w_p \cdot x_p,$$

where $x_i = (x_{i1}, x_{i2}, \dots, x_{iN})^T \in \mathbb{R}^N, i = 1,2, \dots N$ are vectors of all feature values for a set instance in data and $x_0 = (1,1, \dots ,1)^T \in \mathbb{R}^N$.

If data matrix is defined as $X = \left(x_0, x_1, x_2, \dots, x_p\right) \in \mathbb{R}^{(p+1)\times N}$, then the corresponding weights $\widehat{w}$ can be found as a solution for the least squares problem

$$||u - \hat{u}||^2 = ||u - X\widehat{w}||^2 \to min.$$

Using orthogonal projector $P_X = X(X^TX)^{-1}X^T \in \mathbb{R}^{N\times N}$ we get that

$$\hat{u} = P_X u = X(X^TX)^{-1}X^T u,$$

therefore, $\widehat{w} = (X^TX)^{-1}X^T u$.

~~Discriminant analysis.~~

Linear regression can be used as a classifier is some cases. This is called linear discriminant analysis and the regression line itself is called discriminant.

~~What is the Support vector machine?~~

Support vector machine is a model that constructs a hyperplane that acts a classifier for the data. This hyperplane is the solution for the following task:

Find hyperplane H: $(w, x) - b = 0$ for $x \in H$, and $w \perp H,\ ||w|| = 1$, that maximizes λ with following constraints:

$(w, x_i) - b \ \geq \lambda$ for x_i in “yes” class,

$(w, x_i) - \ b \ \leq -\lambda$ for x_i in “no” class.

Here, $(\cdot,\cdot)$ is scalar product.

Naïve Bayes approach at building a classifier.

Let us consider a problem of classification. We have some data that contains multiple entries $x_i = \left(x_{i,1}, x_{i,2}, \dots, x_{i,p}\right), i = 1,2,\dots,N$, that have p features and are divided into several distinct categories $C_k, k = 1,2,\dots,m$, and we also have an entry $x_0 = (x_{0,1}, x_{0,2}, \dots, x_{0,p})$ that does not have a category, thus we have to classify it.

Assume that each class has a corresponding probability density function that generates its entries

$$f_{C_k} = f_{C_k}\left(x_1, x_2, \dots, x_p\right), k = 1,2,\dots,m.$$

Let us also assume that the features in data are independent, hence

$$f_{C_k}\left(x_1, x_2, \dots, x_p\right) = f_{C_k1}(x_1) \cdot f_{C_k2}(x_2) \cdot \dots \cdot f_{C_kp}\left(x_p\right), k = 1,2,\dots,m.$$

If we assume that prior probabilities of classes are given by their frequencies in data $\mathbb{P}(C_k) = \frac{|C_k|}{\sum_{j=1}^{m}|C_j|}$ and conditional probabilities $\mathbb{P}(x_0|\, C_k)$ are given by the corresponding density function

$$\mathbb{P}(x_0|\, C_k) = f_{C_k1}\left(x_{0,1}\right) \cdot f_{C_k2}\left(x_{0,2}\right) \cdot \dots \cdot f_{C_kp}\left(x_{0,p}\right),$$

we can assign the entry x_0 to whichever class that has the greatest posterior probability $\mathbb{P}(C_k|\, x_0) = \mathbb{P}(C_k) \cdot \mathbb{P}(x_0|\, C_k)$. In case the probabilities $f_{C_kj}\left(x_{0,j}\right), j = 1,2,\dots,p$ have small values, the problem can be substituted with maximization of log-probabilities:

$$\ln\left(\mathbb{P}(C_k|\, x_0)\right) = \ln\left(\mathbb{P}(C_k)\right) + \sum_{j=1}^{p} \ln\left(f_{C_kj}\left(x_{0,j}\right)\right).$$

Bag-of-Words model and Laplace smoothing for estimating within-cluster probabilities.

Let us consider a case of text classification by the number of certain keywords contained in it. In some cases there are keywords that are present in each instance with different number of repetitions, or keywords that do not appear in any instance.

The *Bag-of-Words* model is helpful for solving the first issue. It assigns each keyword w_i following probability:

$$\mathbb{P}(w_i) = \frac{\#\text{ of uses of keyword } w_i}{\#\text{ uses of all keywords}}.$$

However, this approach is flawed, as the probabilities of words with zero repetitions are equal to zero. This issue can be solved by using a technique called *Laplace smoothing*. By adding one use to each keyword w_i we get that

$$\mathbb{P}(w_i) = \frac{\#\text{ of uses of keyword } w_i + 1}{\#\text{ uses of all keywords} + \#\text{ of keywords}}.$$

Thus, even the keywords that do not appear in any instance have a non-zero probability.

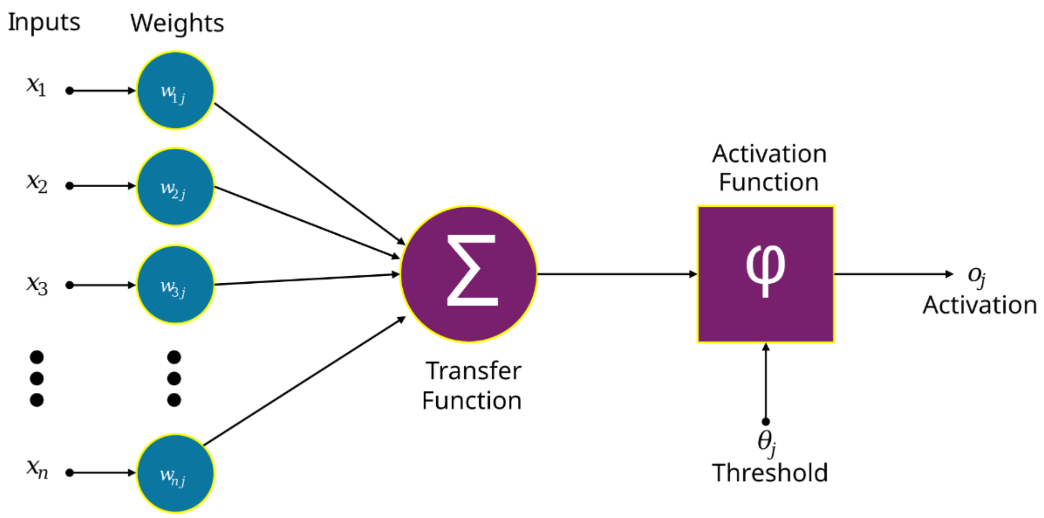
Accuracy metrics: Accuracy, precision, recall, F-measure.

Given that TP is the number of true positives (correctly classified positive instances), TN is the number of true negatives (correctly classified negative instances), FP is the number of false positives (incorrectly classified negative instances) and FN is the number of false negatives (incorrectly classified positive instances), here are several popular classification metrics:

- $\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN}$. This is the proportion of instances that got correctly classified.
- $\text{Precision} = \frac{TP}{TP+FP}$ is the proportion of true positives in all instances that were classified as positive.
- $\text{Recall} = \frac{TP}{TP+FN}$ is the proportion of positive instances that were correctly classified as such.
- $\text{F} - \text{measure} = \frac{2\,\text{precision}\cdot\text{recall}}{\text{precision}+\text{recall}}$. This is the harmonic mean of precision and recall.

Artificial neuron; activation function; Sigmoid, Tangent hyperbolic.

An artificial neuron is a mathematical function conceived as a model of a biological neuron in a neural network. It consists of several parts: inputs, their weights, an activation function and an output, -- as shown on the figure below.



A neuron is activated if weighted sum of all inputs is greater than a certain threshold:

$$\sum_{n=1}^{N} w_n x_n > a.$$

Furthermore, this threshold can be considered an additional input:

$$\sum_{n=1}^{N} w_n x_n + a \cdot (-1) > 0,$$

$$\sum_{n=1}^{N+1} w_n x_n > 0.$$

Then this sum is put through activation function to get the neuron’s output. There are several popular activation functions:

1. *sign* function, $\hat{u} = \operatorname{sign}(u)$,
2. *linear* function, $\hat{u} = au + b$,

3. *ReLU* (Rectified linear unit), $\hat{u} \ = \ \max(0, u)$,
4. *sigmoid* (logistic) activation function

$$\hat{u} = \sigma(u) = \frac{1}{1 + e^{-u}},$$

5. *tanh* activation function

$$\hat{u} \ = \tanh(u) = 2\sigma(2u) - \ 1.$$

This is a smoothed version of $\text{sign}(u)$ and a symmetric version of the sigmoid:

$$\tanh(x) = \tanh(-x).$$

The structure of a feed-forward neural network with a hidden layer.

Let us consider a neural network with inputs $u_i, i = 1,2, \dots, N$, outputs $\hat{y}_j, j = 1,2, \dots, m$, a hidden layer consisting of N neurons with activation function $\varphi = \varphi(z)$. The input vector of the *i*th neuron in hidden layer is given by a following formula:

$$z_i = \sum_{j=1}^{N} w_{ji} u_j,$$

where w_{ji} are input weights. Then, the output of the *i*th neuron in hidden layer is $y_i = \varphi(z_i)$. Finally, *j*th output is given by following equation:

$$\hat{y}_j = \sum_{k=1}^{N} v_{kj} \, \varphi(z_k) = \sum_{k=1}^{N} v_{kj} y_k,$$

where $\tanh(z_k)$ is the output of *k*th neuron in hidden layer and v_{kj} are weights.

The outputs can also be found in vector form. The inputs for hidden layer then are

$$z = u^T W,$$

where $u = (u_1, u_2, \dots, u_N)^T$ is the input vector and $W = \begin{pmatrix} w_{11} & \cdots & w_{1N} \\ \vdots & \ddots & \vdots \\ w_{N1} & \cdots & w_{NN} \end{pmatrix} \in \mathbb{R}^{N \times N}$ is the matrix of weights. Then, the output of hidden layer is $y = \varphi(z)$ and the output layer is given by $\hat{y} = \varphi(z)^T V = \varphi(u^T W)^T V$, where $V \in \mathbb{R}^{N\times m}$ is the matrix of hidden layer weights.

The problem of learning a neural network.

The output $\hat{y}$ of a neural network is given by formula

$$\hat{y} = \varphi(u^T W)^T V,$$

where both W and V are unknown matrices, u is input vector and $\varphi = \varphi(z)$ is activation function. The goal of learning in a neural network is estimating those matrices in such a way that they produce the least possible error. This can be done by solving the following optimization problem:

$$e(W,V) = ||y - \hat{y}||^2 = ||y - \varphi(u^T W)^T V||^2 \to min.$$

Steepest (gradient) descent method.

Gradient descent method for finding a minimum works by iteratively moving an argument x of a function $f = f(x)$ in the direction of its antigradient:

$$x^{t+1} = x^t - \lambda \cdot \overrightarrow{\text{grad}}(f)\Big|_{x=x^t}.$$

Here, $\overrightarrow{\text{grad}}(f)\Big|_{x=x^t} = \sum_{i=1}^{n} \frac{\partial f}{\partial x_i}\Big|_{x=x^t} \boldsymbol{e_i}$ is gradient of a function f at point x^t and λ is the learning rate.

In this case, given that error function is

$$e = (y - \varphi(u^T W)^T V, y - \varphi(u^T W)^T V), \text{or}$$

$$e = \sum_{i=1}^{m} \left(y_i - \sum_{j=1}^{N} v_{ji} \cdot \varphi \left(\sum_{k=1}^{N} w_{kj} u_k \right) \right)^2,$$

the weights w_{ij} and v_{ij} in a neural network can be found in a following way:

$$v_{ij}^{t+1} = v_{ij}^t - \lambda \cdot \frac{\partial e}{\partial v_{ij}}\Bigg|_{v_{ij}=v_{ij}^t},$$

$$w_{ij}^{t+1} = w_{ij}^{t} - \lambda \cdot \left.\frac{\partial e}{\partial w_{ij}}\right|_{w_{ij}=w_{ij}^{t}}.$$

Let us consider the gradients of the error function over w_{ij} and v_{ij}. They are given by following expressions:

$$\frac{\partial e}{\partial v_{ji}} = 2\left(y_i - \sum_{l=1}^{N} v_{li} \cdot \varphi\left(\sum_{k=1}^{N} w_{kl} u_k\right)\right) \cdot \left(-\varphi'\left(\sum_{k=1}^{N} w_{kj} u_k\right)\right),$$

$$\frac{\partial e}{\partial w_{kj}} = 2\sum_{i=1}^{m}\left(\left(y_i - \sum_{p=1}^{N} v_{pi} \cdot \varphi\left(\sum_{q=1}^{N} w_{qp} u_q\right)\right) \cdot \left(-\frac{\partial \sum_{p=1}^{N} v_{pi} \cdot \varphi\left(\sum_{q=1}^{N} w_{qp} u_q\right)}{\partial w_{kj}}\right)\right),$$

$$\frac{\partial e}{\partial w_{kj}} = 2\sum_{i=1}^{m}\left(\left(y_i - \sum_{p=1}^{N} v_{pi} \cdot \varphi\left(\sum_{q=1}^{N} w_{qp} u_q\right)\right) \cdot \left(-v_{ji} \cdot \varphi'\left(\sum_{q=1}^{N} w_{qj} u_q\right) \cdot u_k\right)\right).$$

The method of error back-propagation to minimize the squared error; epoch.

Error backpropagation in neural networks is defined as a following algorithm:

1. Specify NE, *number of epochs*. Standardize features to $[-1,1]$ interval. Generate W and V as matrices of random values with standard normal distribution.
2. **Epoch**. In a loop over objects (x, y).
   2.1. Forward computation: Find $\hat{y} = F(x)$ and error $e = ||y - \hat{y}||^2$.
   2.2. Error back-propagation: Estimate the derivatives $\frac{\partial e}{\partial v_{ji}}$ and $\frac{\partial e}{\partial w_{kj}}$ of error function e.
   2.3. Weights update: Update the weights V and W using gradient descent.
3. **Halting test:** If the number of epochs reached NE, output e, E, V, W and halt, otherwise randomize the order of objects and go to step 1.

Singular triplet of a rectangular matrix Y.

(σ, c, z) is called a singular triplet for matrix $Y \in \mathbb{R}^{n\times m}$ if and only if

$$\begin{cases} Yc = \sigma c, \\ Y^T z = \sigma c, \end{cases}$$

c is eigenvector of Y^TY, $\sigma^2 = \lambda$, where λ is eigenvalue of Y^TY and $z = Yc/\sigma$.

Properties of singular triplets:

1° The number of different singular triplets is equal to $\text{rank}(Y)$.

2° Different singular c's are mutually orthogonal, as are different singular z's.

Eigenvalues of YY$^{\mathbf{T}}$ and Y$^{\mathbf{T}}$Y matrices; relation to singular triplets of Y.

Let us consider a following problem for a matrix $Y \in \mathbb{R}^{n\times m}$: find c such that $c \,||\, Y^TYc$. For each c that solves the problem we get that

$$Y^TYc = \lambda c, \lambda \in \mathbb{R}^+.$$

Here, $\mathbb{R}^+ = [0, +\infty)$, and it is clear that solutions c must be eigenvectors of Y^TY. Furthermore, their corresponding eigenvalues λ are positive-semidefinite, since they are squared singular values of Y. Indeed, by substituting Y with its singular value decomposition $Y = U\Sigma V^T$ we get that

$$Y^TY = V\Sigma^T U^T U \Sigma V^T = V\Sigma^T\Sigma V^T$$

because matrices U and V are orthogonal.

By substituting eigenvalues λ with squares their singular counterparts σ we get that

$$Y^TYc = \sigma^2 c,$$

$$Y^T(Yc/\sigma) = \sigma c.$$

Denoting Yc/σ as z we get that $Y^Tz = \sigma c$. Tuple (σ, c, z) is called singular triplet for matrix Y.

Singular value decomposition of a rectangular matrix.

For matrix $Y \in \mathbb{R}^{n \times m}$ singular value decomposition (SVD) is factorization of the form $Y = U\Sigma V^T$, where $U \in \mathbb{R}^{n \times n}$ and $V \in \mathbb{R}^{m \times m}$ are orthogonal matrices and $\Sigma \in \mathbb{R}^{n \times m}$ is a rectangular diagonal matrix. Alternatively, given a set if singular triplets sorted by their singular value σ_i in descending order we get that SVD of a matrix Y is:

$$Y = \sigma_1 z_1 c_1^T + \sigma_2 z_2 c_2^T + \cdots + \sigma_r z_r c_r^T,$$

where $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r$, $r = \text{rank Y}$, $z_i \in \mathbb{R}^n$, $c_j \in \mathbb{R}^m$. Matrix elements can be found using following equation:

$$y_{ij} = \sigma_1 z_{i1} c_{j1} + \sigma_2 z_{i2} c_{j2} + \cdots + \sigma_r z_{ir} c_{jr}.$$

Approximation of a data table in a p-dimensional space.

Given data table $X \in \mathbb{R}^{n \times m}$, rank $X = r$, let us consider a following problem: find a matrix $Y \in \mathbb{R}^{n \times m}$, rank $Y = p < r$ such that

$$||X - Y||^2 = \sum_{i,j} \left(x_{ij} - y_{ij}\right)^2 \to min.$$

The solution for this problem is to suppose Y to be the sum of first p singular triplets of matrix X:

$$Y = \sigma_1 z_1 c_1^T + \sigma_2 z_2 c_2^T + \cdots + \sigma_p z_p c_p^T.$$

Principal component model and its relation to the singular value decomposition.

Given data table $X \in \mathbb{R}^{n \times m}$, rank $X = r$, let us consider a following problem: find $p < r$ hidden factors $z^* = (z_1^*, z_2^*, \dots, z_n^*)^T$ and loadings vector $c^* = (c_1^*, c_2^*, \dots, c_m^*)$ such that

$$x_{ij} = z_i^* \cdot c_j^* + e_{ij},$$

$$l = \sum_{i,j} \left(x_{ij} - z_i^* \cdot c_j^*\right)^2 \to min.$$

Since this model is multiplicative, it has multiple solutions. To avoid that let us convert it to a constrained optimization problem:

$$l = \sum_{i,j} \left(x_{ij} - z_i^* \cdot c_j^*\right)^2 \to min,$$

$$||z^*|| = ||c^*|| = 1.$$

The solution for this problem is the first singular triplet of matrix X:

$$x_{ij} = \sigma_1 z_{i1} c_{j1} + e_{ij},$$

thus, $z^* = \sqrt{\sigma_1} z_1, c^* = \sqrt{\sigma_1} c_1$.

Contribution of a principal component to the data scatter.

Data scatter is given by $\sum_{i,j} x_{ij}^2$. It can be decomposed in a following way:

$$\sum_{i,j} x_{ij}^2 = \sigma_1^2 + l,$$

where σ_1^2 is *contribution* of the first singular triplet of $X \in \mathbb{R}^{n\times m}$ and $l = \sum_{i,j}\left(x_{ij} - z_i^* \cdot c_j^*\right)^2$ is least squares criterion.

Covariance and correlation interpretation of Y^TY matrix.

Given a data matrix $X \in \mathbb{R}^{n\times m}$, we can compute its centred version $Y \in \mathbb{R}^{n\times m}$ and the feature covariance matrix B:

1. Centre matrix X by finding, for each feature, its mean and subtracting it from all the feature values: $Y = X - \overline{X}$.
2. Compute square matrix Y^TY and divide it by n or $n-1$ depending on sample variance choice: $B = Y^TY/n$. Dividing B by square root of variances of corresponding features we get the correlation matrix.

Conventional formulation of the PCA method; its relation to that SVD based; similarities and differences.

Conventional definition of the PCA method is following: given a data matrix $X \in \mathbb{R}^{n\times m}$ and its centered version $Y \in \mathbb{R}^{n\times m}$, first principal component (PC) is a weighted combination z of features of X after centering, that is,

$$z = Yc,$$

that has the maximum variance with respect to all normed c. The first principal component itself is calculated in a following way:

1. Compute centred version Y of X and the feature covariance matrix $B = Y^T Y/n$.
2. Find the first eigenvalue λ_1 and corresponding normed eigenvector c_1 so that $Bc_1 \; = \; \lambda_1 c_1$;
3. Compute the principal component

$$z = \frac{Yc_1}{\sqrt{n\lambda_1}}.$$

The 2nd PC is computed in the same way based on a residual covariance matrix $\dot{B} = B - \; \lambda_1 c_1 c_1^T$, the rest of components are found in a similar way.

Using PCA for measuring a hidden variable (ranking).

Hidden factor can be derived in a following way: first, select base features and convert them into the same scale to form matrix $X \in \mathbb{R}^{n\times m}$, then apply PCA to the data matrix X to find the first singular triplet (σ, c, z). After that, rescale z to range [0,100] using following equation:

$$z = \alpha z = \alpha \cdot Xc.$$

Here, $\alpha \in \mathbb{R}$ is such that

$$100 = s^T c \cdot \alpha,$$

where $s = (100,100, \dots ,100)^T \in \mathbb{R}^m$.

Data visualization by using PCA.

Data visualization using PCA can be done with a following procedure:

1. Calculate the centred data matrix Y by subtracting the column means from the columns of X; normalize if needed.
2. Compute two first singular triplets $(\sigma_i, c_i, z_i), i = 1,2$.
3. Rescale them according to following equations:

$$\begin{cases} z_i^* = z_i \cdot \sqrt{\sigma_i}, \\ c_i^* = c_i^* \cdot \sqrt{\sigma_i}, \end{cases} i = 1,2.$$

4. Determine the proportion p of the "explained" variance using formula

$$p = 100\% \cdot \frac{\sigma_1^2 + \sigma_2^2}{\sum_{i,j} x_{ij}^2}.$$

5. Visualize the data as a scatterplot.
6. Interpret the axes by looking at the loadings c_i^* and their signs.

Should there be any difference in data standardization at PCA applied: (a) for ranking, (b) for visualization?

PageRank method.

K-Means clustering method.

Consider the task of finding $K \in \mathbb{N}$ non-overlapping cluster structures in data. Let each cluster be represented by its centre $C_k \in \mathbb{R}^n$ and set of points $S_k, k = 1,2,\dots,K$. Then, K-Means clustering method can be defined by the following algorithm:

1. Provide initial cluster centres $C_k^0, k = 1,2,\dots,K$. Assign each data point to the cluster corresponding to the closest centre.
2. Compute new cluster centres $C_k^t, k = 1,2,\dots,K$ as means of all data points in corresponding cluster.
3. Compute new sets of points S_k^t for each cluster $k = 1,2,\dots,K$ by assigning each data point to the cluster corresponding to the closest centre.
4. Repeat this process until either the new cluster centres coincide with the previous ones.

Quadratic Error Criterion (QEC) for partitioning; its formulation in terms of Euclidean distances.

QEC is as defined as follows: find partition $S = S_1, S_2, \dots, S_K$ and centres $C = C_1, C_2, \dots, C_K$ which minimize inertia $D(S,C)$

$$D(S,C) = \sum_{k=1}^{K} \sum_{i \in S_k} d(i, C_k) = \sum_{k=1}^{K} \sum_{i \in S_k} \sum_{v \in V} (y_{iv} - C_{kv})^2,$$

where $d(i, C_k) = (i - C_k, i - C_k) = ||i - C_k||^2$ is Euclidean distance between centre $C_k = (C_{k1}, C_{k2}, \dots, C_{km})^T$ and entity i from partition S_k, v is a feature from set of all features V, y_{iv} is the value of feature v at i, $k = 1,2,\dots K$ and K is the number of clusters.

QED is equivalent to the least-squares criterion for model

$$y_{iv} = \sum_{k=1}^{K} z_{ik} c_{kv} + e_{iv},$$

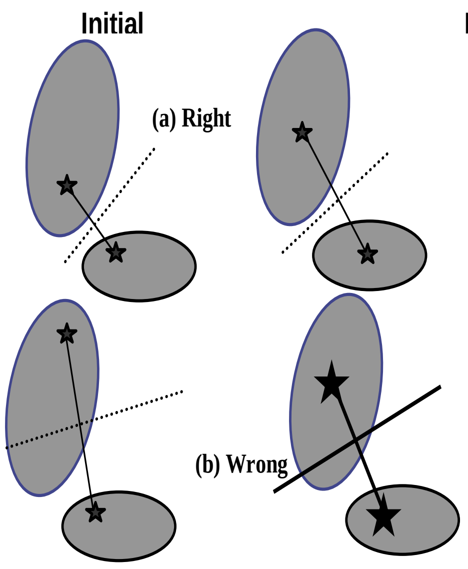
where $z_{ik} = 1$ for $i \in S_k$, and $z_{ik} = 0$ for $i \notin S_k$.

Alternating minimization method for QEC and its relation to K-Means.

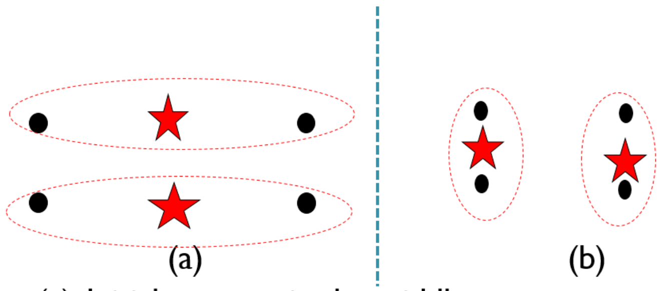
QEC can be implemented using alternating minimization of the inertia coefficient $D(S,C)$. First, we compute $\min_S D(S,C)$ and update clusters accordingly, then we compute $\min_C D(S,C)$ and update their centres. With this algorithm inertia decreases at each step, thus K-Means converges.

Initialization of K-Means: May this affect the result?

The results heavily depend on the initial clusters, as shown on the picture:



Moreover, there may be several different stable partitions of data:



(a): Initial centers in the middle
(b) Initial centers are right

Decomposition of the data scatter in the explained part and the quadratic error criterion.

Data scatter for data matrix $X \in \mathbb{R}^{n\times m}$ can be decomposed in a following way:

$$\sum_{i,j} x_{ij}^2 = D(S,C) + W(S,C),$$

where $D(S,C)$ is inertia index, $W(S,C) = \sum_{k=1}^{K} |S_k|(C_k, C_k)$, $K \in \mathbb{N}$ is the number of clusters, C_k – their centres, S_k – set of all data points in *k^th^* cluster and $(\cdot,\cdot)$ is scalar product. Since the data scatter is constant, the task of inertia minimization is equivalent to maximization of $W(S,C)$.

Formulaic expression of the explained part and its geometric meaning.

Probably given below?..

Anomalous cluster algorithm.

While solving the following optimization problem:

$$W(S,C) = \sum_{k=1}^{K} |S_k|(C_k, C_k) = \sum_{i,j} x_{ij}^2 - D(S,C) \to max,$$

if the data $X \in \mathbb{R}^{n\times m}$ is pre-processed by centring it so that the grand mean is $\vec{\mathbf{0}} \in \mathbb{R}^m$, or origin of the space, we get that anomalous populated clusters maximize the function $W(S,C)$. Clusters are called *anomalous*, if their centres are removed far from the grand mean.

An anomalous cluster can be found using the following algorithm:

1. With data centred at the grand mean, select the furthest point from it as initial cluster centre C.

2. Assign all the points that are closer to C than to grand mean to the corresponding set of points S.
3. Find centroid C' of S, terminate the process if $C = C'$, otherwise got to step 2.

Initialization at the intelligent K-Means.

Intelligent K-Means initialization uses centres of K most populated anomalous clusters for its initialization. It goes as follows:

1. Centre the data so that the grand mean coincides with origin of the space. Then normalize it if necessary.
2. Find all anomalous clusters by iteratively finding and removing them from the data.
3. Pick the K largest of them by number of entities, or, in case K is difficult to specify, pick those that go over some threshold.
4. Initialize K-Means at centres of chosen clusters.

Interpretation of clusters using relative differences.

Given a cluster k and a quantitative feature $v \in V$, the *relative difference* is computed using following equation:

$$d_{kv} = \left(\frac{c_{kv} - c_v}{c_v}\right) \cdot 100\% = \left(\frac{c_{kv}}{c_v} - 1\right) \cdot 100\%.$$

Here c_{kv} is within-cluster mean of v, and c_v is grand mean of v. However, relative difference is not defined for categorical feature. For them the *Quetelet's index* can be used instead. Given a cluster k and a category v in feature $F \in V$, Quetelet's index is:

$$q_{kv} = \left(\frac{p_{kv}}{p_k p_v} - 1\right) \cdot 100\%.$$

Here p_{kv} is the proportion of entities falling in both cluster k and category v, p_v is proportion of category v in the dataset, p_k is proportion of cluster k in the dataset.

With that, we can interpret clusters using the following algorithm:

1. Given a cluster k, pick up those features and categories $v \in F$ for which values of d_{kv} or q_{kv} are far from 0, say, greater than 30-35%, forming set V_k^+, or smaller than 30-35%, forming set V_k^-.
2. Describe cluster k as that characterized by features from V_k^+ as "much greater than average" and features from V_k^- as "much smaller than average" (for larger deviations, you may use modifier "very much").

Advantages and drawbacks of K-means.

Advantages of K-Means:

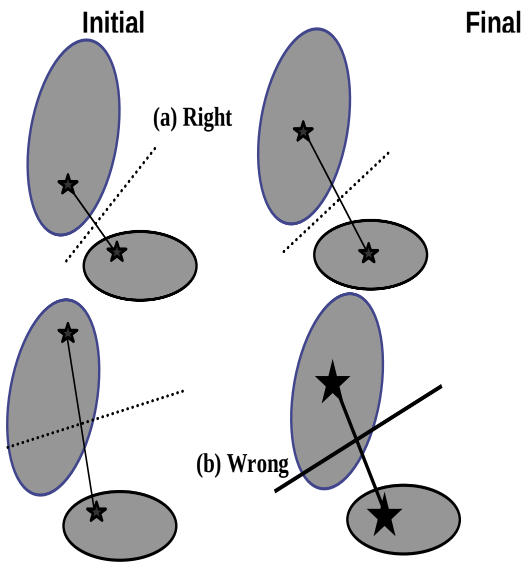
- K-Means computations mimic typology making;
- computation is intuitive;
- computation is fast and requires no additional memory;
- computation is easy to parallelize.

Drawbacks of K-Means:

- K-Means may converge to the answer really slowly;
- results heavily depend on the initialization;
- choosing the number of clusters K and location of initial centres is its own issue.

Initialization issue; K-means++.

The results heavily depend on the initial clusters, as shown on the picture:



This issue can be solved by using K-Means++ algorithm that goes as follows:

1. Choose a random object as the first centre.
2. Choose each next centre as a random object with the probability proportional to the distance from the nearest out of already chosen centres.

Number of clusters: Elbow method, Hartigan Index, Calinski-Harabasz index, Silhouette Width

In many cases the number of clusters in data is not known, hence there is a need for an estimate of the optimal number of clusters. This can be done by solving an optimization problem

$$J_K \to max \text{ or } J_K \to min.$$

for K, where K is the number of clusters. First, suppose that this is a minimization problem for $J_K = \min_{|S|=K} D(S,C) = D(K)$. $D(K)$ is monotonically decreasing function:

$$D(K) > D(K+1),$$

therefore, it has no finite minima and its optimum is infinity. However, here are several other J_K options:

- *Elbow index*. $J_K \to max$, where

$$J_K = \frac{D(K-1) - D(K)}{D(K) - D(K+1)} = E(K).$$

- *Calinski-Harabasz index*. $J_K \to max$, where

$$J_K = \frac{\frac{1}{K-1}\big(D(1) - D(K)\big)}{\frac{1}{N-K} D(K)} = CH(K).$$

- *Hartigan index*. $H(K)$ applies starting at $K = 1,2,\ldots$; picks the very first K at which $H(K)$ decreases to 10. $H(K)$ is given by following equation:

$$J_K = \left(\frac{D(K)}{D(K+1)} - 1\right) \cdot (N - K - 1) = H(K).$$

- *Silhouette width index*. $J_K = \sum_{i \in I} s(i)$, $J_K \to max$, where I is the set of all entries and $s(i)$ is given by

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}},$$

  where

$$a(i) = \frac{1}{|G_i| - 1} \sum_{j \in G_i, j \neq i} d(y_i, y),$$

$$b(i) = \min_{G_k:\, i \notin G_k} \frac{1}{|G_k|} \sum_{j \in G_k} d(y_i, y_j),$$

  and G_i is the cluster that contains entry i.

Matrix factorization; examples.

Data standardization; goal of centering, goal of rescaling.

Given data matrix $X \in \mathbb{R}^{n\times m}$, data standardization is done in the following way:

- the origin of linear space is shifted to a point $a = (a_1, a_2, \dots, a_m)^T \in \mathbb{R}^m$ is a process called *centring*;
- the data is rescaled by dividing columns elementwise by a set of coefficients $b = (b_1, b_2, \dots, b_m)^T \in \mathbb{R}^m$. This part of the process is called *rescaling*.

The elements of standardized data matrix are given by following expression:

$$Y_{iv} = \frac{X_{iv} - a_v}{b_v}.$$

Here, a typically is the mean and b is either standard deviation or range of data in X. The goal of centering is to compare the data to a certain reference point, while the goal of rescaling to make features comparable to each other (?).

Bootstrapping to validate the mean: pivotal and non-pivotal versions.

Bootstrapping is the procedure for estimating the distribution of statistics over data. It works by drawing the data with replacement and computing the statistic for sampled data multiple times in order to construct the distribution.

This can be used to validate an estimate of mean of data distribution by checking whether it falls into 95% confidence interval. Given the number of samples N we draw a portion of data with replacement N times and calculate the mean over each sample. Then we construct the histogram of means and use it to find the 95% confidence interval for the mean by either:

- assuming that means have normal distribution and applying two-sigma rule to determine the interval's boundaries;
- or taking 2.5% and 97.5% percentiles of data.

The first method is referred to as *pivotal method* and the latter is referred to as *non-pivotal method*.

Bootstrapping to compare the means: pivotal and non-pivotal versions.

Idk, probably a part of the above?..

Contingency table, its marginal row and column.

Given a non-empty set of entities and two sets of categories on them

$$G_K = \{1,2, \dots, K\}, K \in \mathbb{N} \setminus \{1\}, \text{and } H_L = \{1,2, \dots, L\}, L \in \mathbb{N} \setminus \{1\},$$

we can cross-classify those categories by constructing a table where $k \in G_K$ are the rows, $l \in H_L$ are the columns, values in the cells are either regular frequencies N_{kv} or relative frequencies p_{kv} of entries containing both corresponding categories k and l. Marginal row of this table is the sum of data over all the previous rows and represents overall frequencies of different categories in l in data. Dually, marginal column represents overall frequencies of different categories in k in data.

Conditional probability and statistical independence.

Given a contingency table constructed over two sets of categories

$$G_K = \{1,2, \dots, K\}, K \in \mathbb{N} \setminus \{1\}, \text{and } H_L = \{1,2, \dots, L\}, L \in \mathbb{N} \setminus \{1\},$$

we can substitute the values in cells with conditional ones given by

$$p(v|k) = \frac{p_{kv}}{p_k} = \frac{N_{kv}}{N_k},$$

where p_k – probability of category $k \in G_K$, which the sum of p_{kv} over $v \in H_L$, and N_k is frequency of category k given by the sum of N_{kv} over v.

It is known that features G_K and H_L are statistically independent if

$$\mathbb{P}(k \cap v\ ) = \mathbb{P}(k) \cdot \mathbb{P}(v), \forall k \in G_K, \forall v \in H_L.$$

Thus, by checking for each pair categories that $p_{kv} = \sum_i p_{iv} \cdot \sum_j p_{kj}$ we can find out whether G_K and H_L are statistically independent.

Quetelet index, its meaning, relation to rules for interpretation of clusters.

Given category $v \in V$ at cluster S_k, Quetelet index (in percent) is given by following formula:

$$q_{kv} = \left(\frac{p(v|k)}{p(k)p(v)} - 1\right) \cdot 100\%.$$

Shows the relative change of probability of *v* under condition of S_k (from the average).

Averaged Quetelet index Q: definition and meaning.

Given categories G_K and H_L, average Quetelet index is is given by following formula:

$$Q(G_K|H_L) = \sum_{k \in G_K} \sum_{v \in H_L} p(k \cap v) q(k|v).$$

It is an inner product of the two matrices: relative contingency table and Quetelet index table for categories G_K and H_L.

Pearson chi-squared coefficient and its meaning.

Pearson χ^2-coefficient is given by following equation:

$$X^2 = \sum_{k \in G_K} \sum_{v \in H_L} \frac{\left(p(k \cap v) - p(k) \cdot p(v)\right)^2}{p(k) \cdot p(v)}.$$

It is a sum of quadratic differences between the observed bivariate distribution and its form under assumption of statistical independence. It is used for testing the independence of features, since for a pair of independent features

$$X^2 = 0.$$

Testing the independence hypothesis with Pearson chi-squared.

Same as above?..

Relation between the chi-squared coefficient and the averaged Quetelet index.

Pearson χ^2-coefficient X^2 and averaged Quetelet index are given by

$$X^2 = \sum_{k \in G_K} \sum_{v \in H_L} \frac{\big(p(k \cap v) - p(k) \cdot p(v)\big)^2}{p(k) \cdot p(v)},$$

$$Q = \sum_{k \in G_K} \sum_{v \in H_L} p(k \cap v) q(k|v) = \sum_{k \in G_K} \sum_{v \in H_L} \frac{p^2(k \cap v)}{p(k)p(v)} - 1.$$

Therefore,

$$X^2 = Q + 1 - 2 \sum_{k \in G_K} \sum_{v \in H_L} \frac{p(k \cap v)}{p(k) \cdot p(v)} + \sum_{k \in G_K} \sum_{v \in H_L} p(k) \cdot p(v).$$

$$X^2 = Q.$$

Cluster hierarchy as a binary rooted tree with a height function, whose leaves are one-to-one labelled by dataset entities.

Given some dataset and division into clusters, we can express the cluster hierarchy using a binary tree. Here the dataset itself would be the set of leaves, clusters over it would be interior nodes and the height function being such that

$$h(t) = 0, t \text{ is a leaf},$$

$$h(t) < h(s), \text{if } t \subset s.$$

Agglomerative clustering algorithm.

Given a dataset $X = \{x_1, x_2, \dots, x_n\}$, where x_i are individual observations, and a between-cluster similarity/dissimilarity function $D(s,t)$, hierarchical clustering can be done using *agglomerative* algorithms, which are as follows:

1. Initialize the algorithms by assigning each entity its own cluster resulting in a partition $S = \big\{\{x_1\}, \{x_2\}, \dots, \{x_n\}\big\}$.
2. Given a partition $S = \{S_1, S_2, \dots, S_m\}$ of data into m clusters:

   2.1. find s^*, t^* optimizing $D(s,t)$ (thus maximizing similarity);

   2.2. merge clusters S_{s^*} and S_{t^*} to form $S_{s^*t^*} = S_{s^*} \cup S_{t^*}$;

   2.3. compute (dis)similarity between $S_{s^*t^*}$ and every other cluster S_i in $S \setminus \{S_{t^*}, S_{s^*}\}$ to form a new (dis)similarity matrix

   2.4. define and compute height $h(S_{s^*t^*})$.
3. Check where the stopping condition is satisfied. Terminate the algorithm if yes, otherwise let $m' = m - 1$, find new partition $S' = \{S'_1, S'_2, \dots, S'_{m'}\}$ and return to step 2.

Distance between clusters; Ward distance, Nearest Neighbour distance.

The main way agglomerative clustering algorithms differ from each other is in the way they measure distance between clusters. Let us consider the following two metrics:

- *Ward distance*. Given clusters S_k and S_l, their respective centres C_k and C_l and $N_k = |S_k|, N_l = |S_l|$, the ward distance $wd(S_k, S_l)$ is given by:

  $$wd(S_k, S_l) = \frac{N_k \cdot N_l}{N_k + N_l} d(C_k, C_l),$$

  where $d(\cdot,\cdot)$ is Euclidean distance. This metric combines distance between cluster centers and a factor depending on the distribution of objects between the clusters: the smaller the difference in sizes N_k and N_l, the larger the value of the factor.

- *Nearest Neighbour distance*.

Ward distance as the increment of the K-means square error criterion.

Ward distance is given by

$$Wd(S_k, S_l) = W\left(S_k \cup S_l, C(S_k \cup S_l)\right) - W(S, C),$$

where $C_{k\cup l}$ is the centre of cluster $S_k \cup S_l$ and $W(S, C)$ is K-Means squared distance criterion. Since

Type equation here.

Max/Min Spanning Tree and Prim's algorithm.

NN hierarchical clustering with MST: agglomerative and divisive.

Similarity data and sources of them.

Summary average clustering (SAC) criterion inherited from k-means.

Agglomerative clustering with SAC.

Summary criteria: Min-cut and within-cluster.

Modularity and Constant shift transformation.

Agglomerative clustering and Louvain algorithm.

Spectral clustering: Normalized cut, Laplacian transformation, Spectral methods.

LaPIn (Laplacian Pseudo-Inverse transformation).