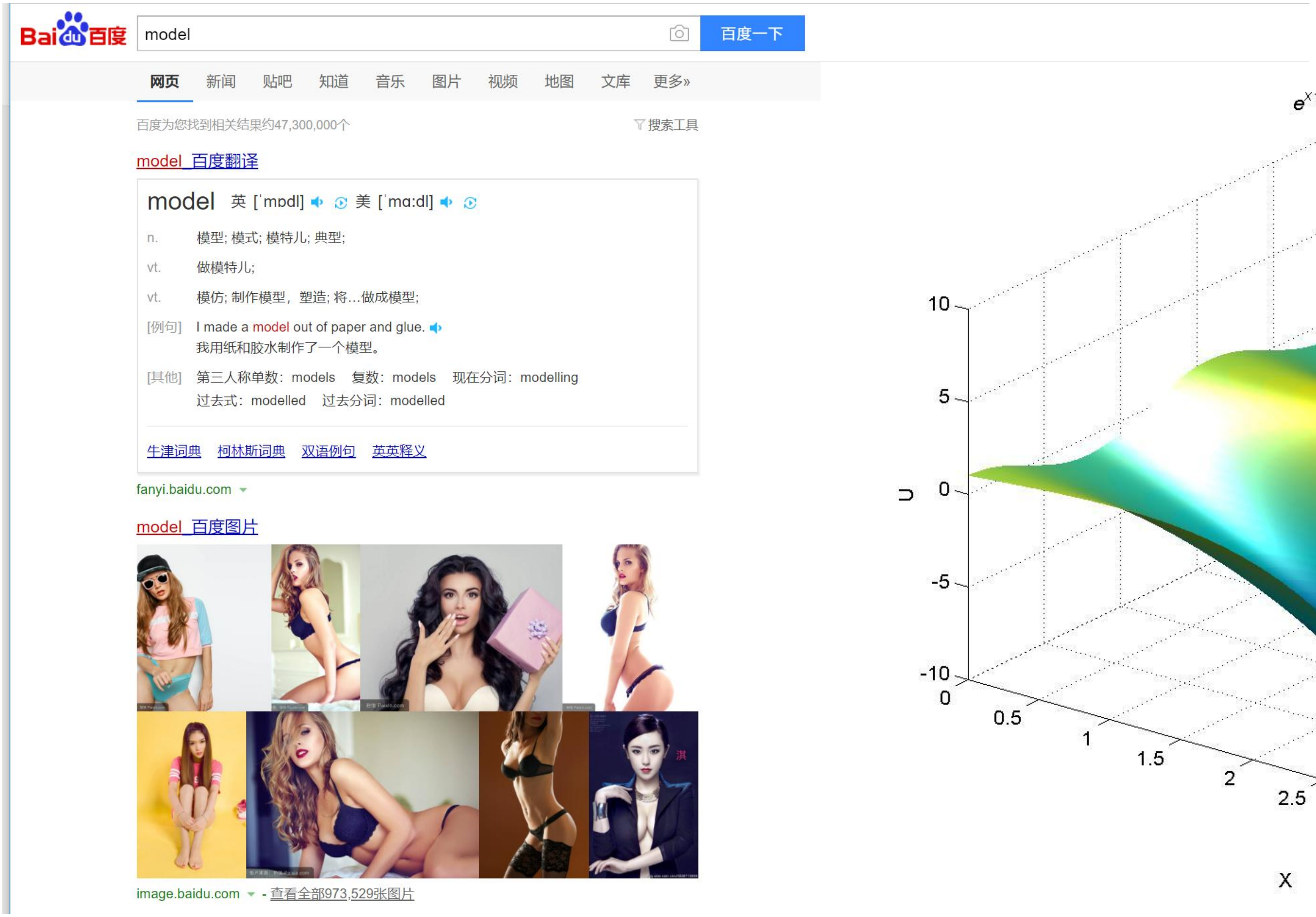
Machine Learning

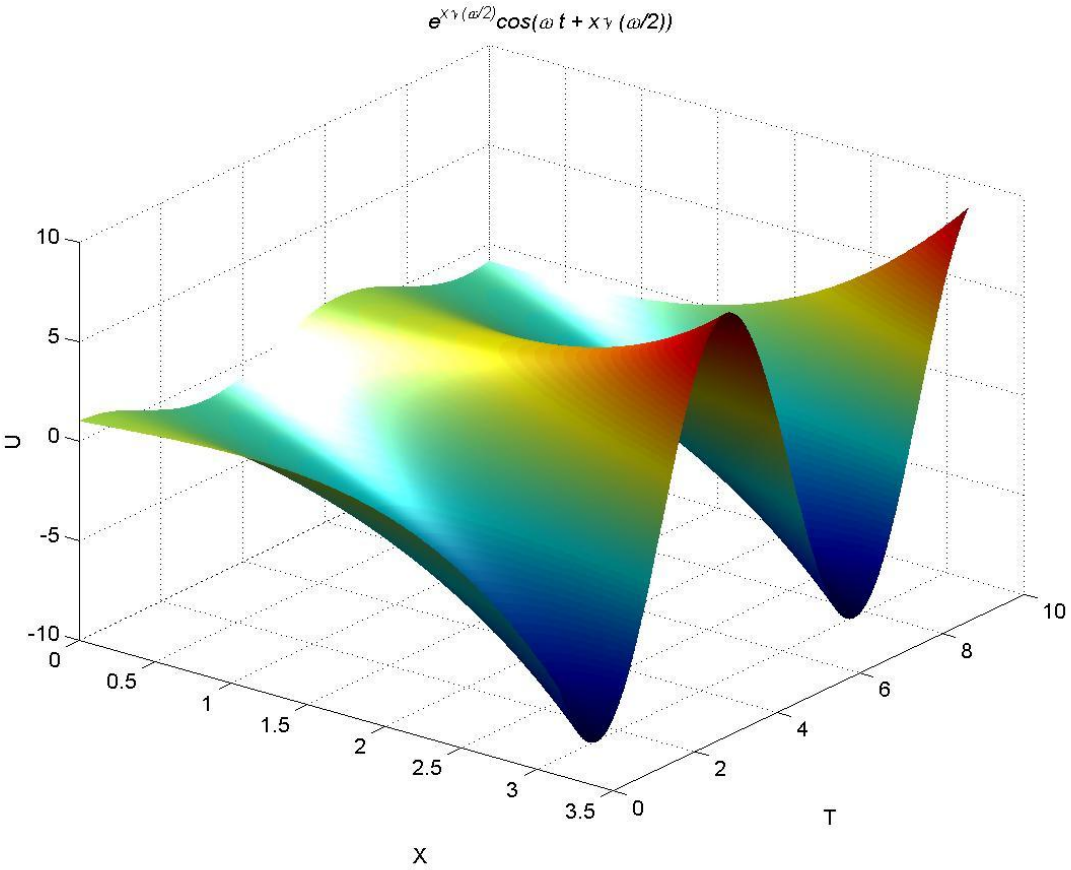
Lecture 2 – Classical Models

Zengchang Qin (PhD)
Intelligent Computing and Machine Learning Lab
Beihang University, Beijing, China
zengchang.qin@gmail.com

Linear Models

Model – Mathematical Models





Example of Linear Prediction

The neuron has a real valued output which is a weighted sum of its inputs

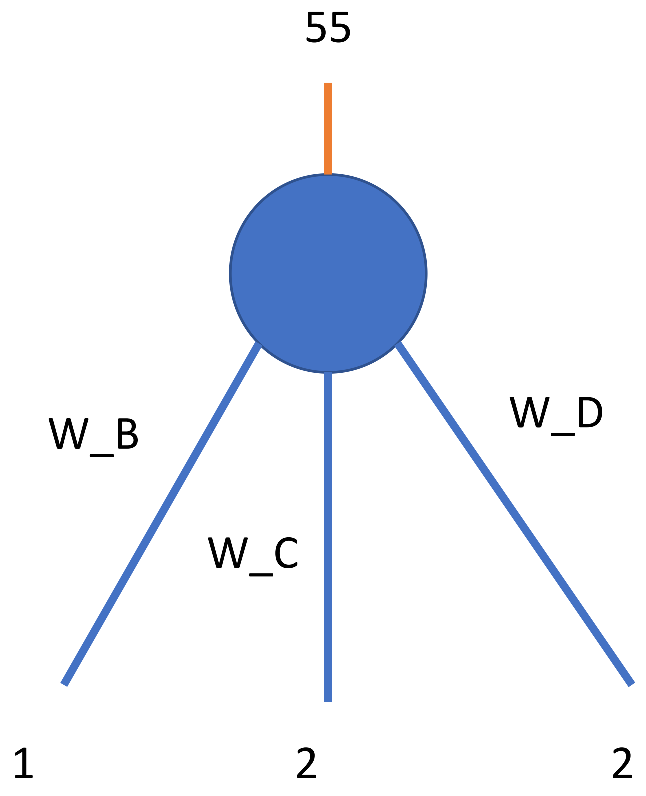
Price = X (Burger) W(Burger) + X(Chips) W(Chips) + X(Drink)W(Drink)

The obvious approach is just to solve a set of simultaneous linear equations, one equation per meal.

Burger	W_B	Chips	W_C	Drink	W_D	Total Price
1		2		2		55
2		1		1		50
0		2		2		40
3		1		1		65

Linear Classification

We may want a method that could be implemented in a learning model

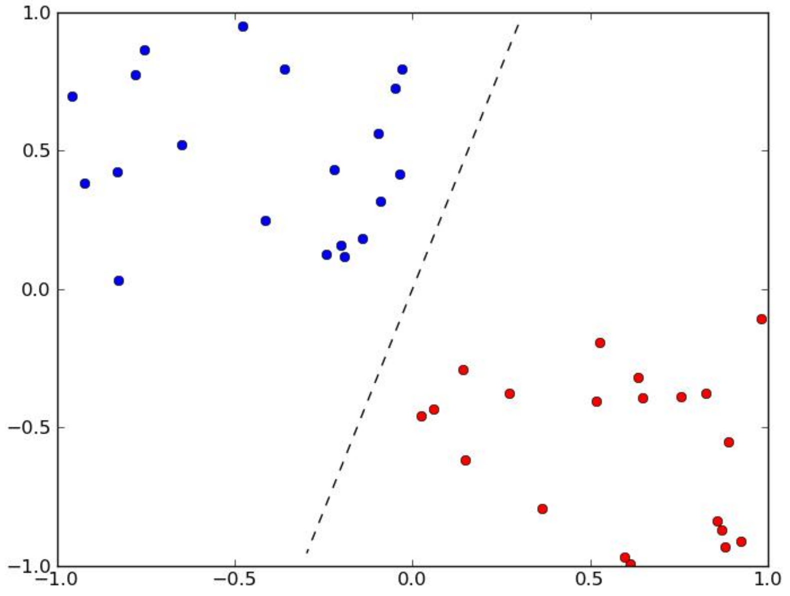


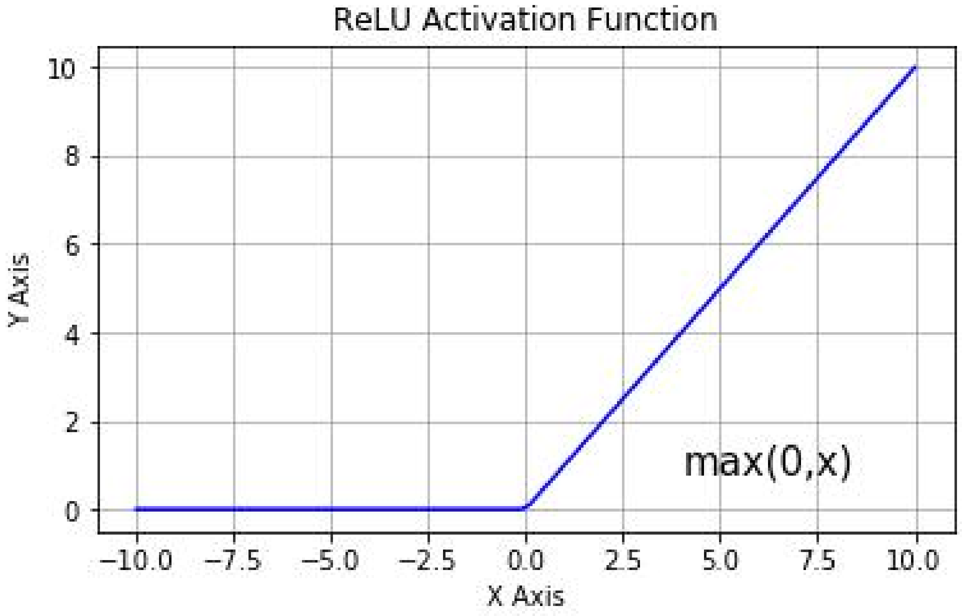
Date_or_Not = X (Handsome) W(Handsome) + X(Salary) W(Salary) + X(Education)W(Education)

Handsome	W_H	Salary	W_S	Education	W_E	Date_or_Not
6		6		9		Yes
5		8		3		No
9		2		8		No
4		8		7		Yes

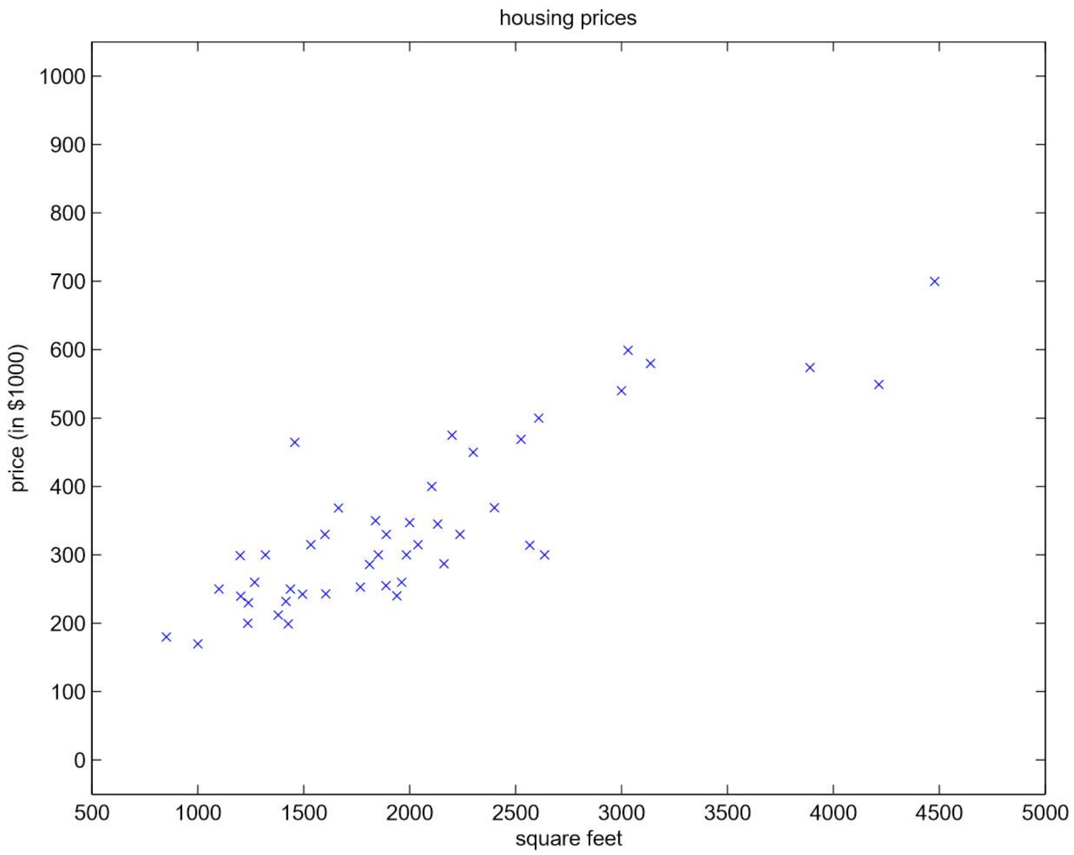
Weight

In short, the activation functions are used to map the input between the required values like (0, 1) or (-1, 1). Weights shows the strength of the particular node. A bias value allows you to shift the activation function curve up or down.





A Machine Learning Problem



https://www.bilibili.com/video/av35141747?from=search&seid=18345810337327930366

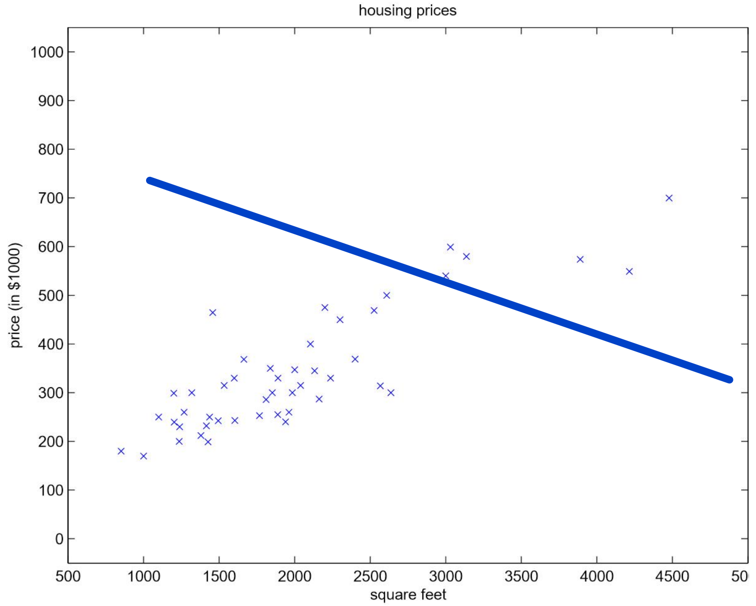
There is a dataset giving the living areas and prices of 47 houses from Portland, Oregon. We are looking for a function gives the pattern of inputs-outputs (A. Ng).

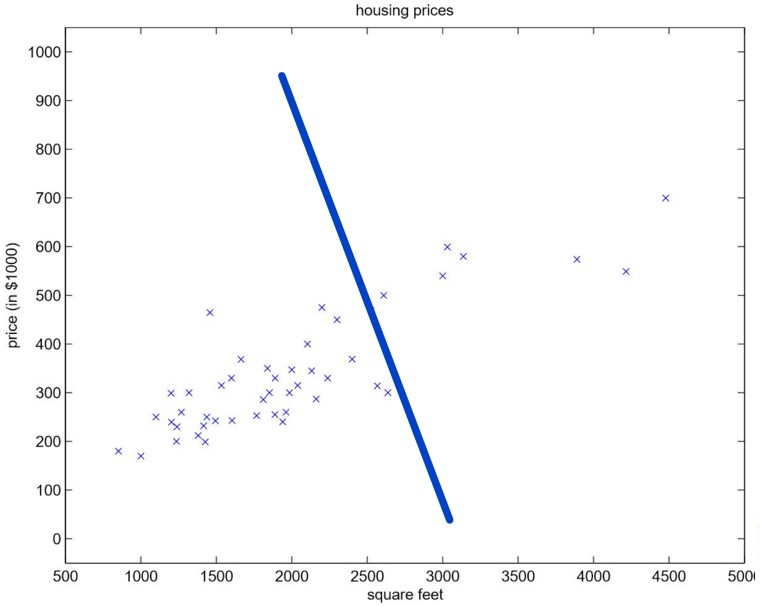
$$\boxed{x \sim F(x) = h_\theta(x)}$$

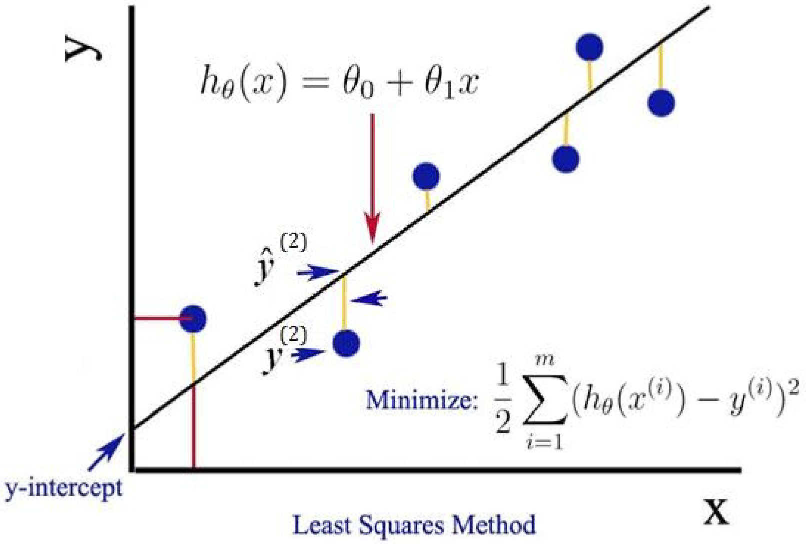
Living area (feet^2)	Price (1000\$s)
2104	400
1600	330
2400	369
1416	232
3000	540
⋮	⋮

$$h_\theta(x) = \theta_0 + \theta_1 x$$

A Classical Example of Linear Regression





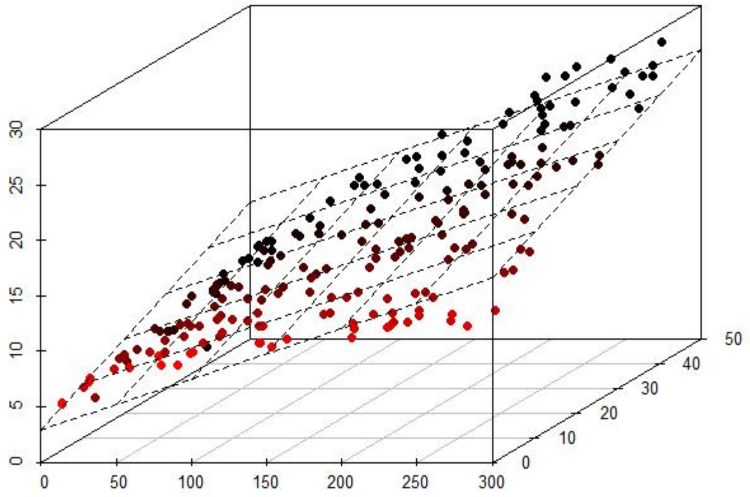


Machine learning is about searching in the hypothesis space! We aim to find the relation of parameter to *soundness of fitting*.

$x \sim h_\theta(x)$ $\Rightarrow$ $\theta \sim J(\theta, x)$

A Higher-Dimensional Case

Living area (feet^2)	#bedrooms	Price (1000$s)
2104	3	400
1600	3	330
2400	3	369
1416	2	232
3000	4	540
⋮	⋮	⋮



From one dimension, we can extend to 2 dimensional case, we can define the general loss function for linear models by:

$$J(\theta) = \frac{1}{2}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2$$

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2$$

$$h(x) = \sum_{i=0}^{n} \theta_i x_i = \theta^T x$$

Derivative of Functions

Consider the function $h : \mathbb{R} \to \mathbb{R}$, $h(t) = (f \circ g)(t)$ with

$$f : \mathbb{R}^2 \to \mathbb{R}$$
$$g : \mathbb{R} \to \mathbb{R}^2$$
$$f(\boldsymbol{x}) = \exp(x_1 x_2^2)\,,$$
$$\boldsymbol{x} = g(t) = \begin{bmatrix} t\cos t \\ t\sin t \end{bmatrix}$$

and compute the gradient of h with respect to t.

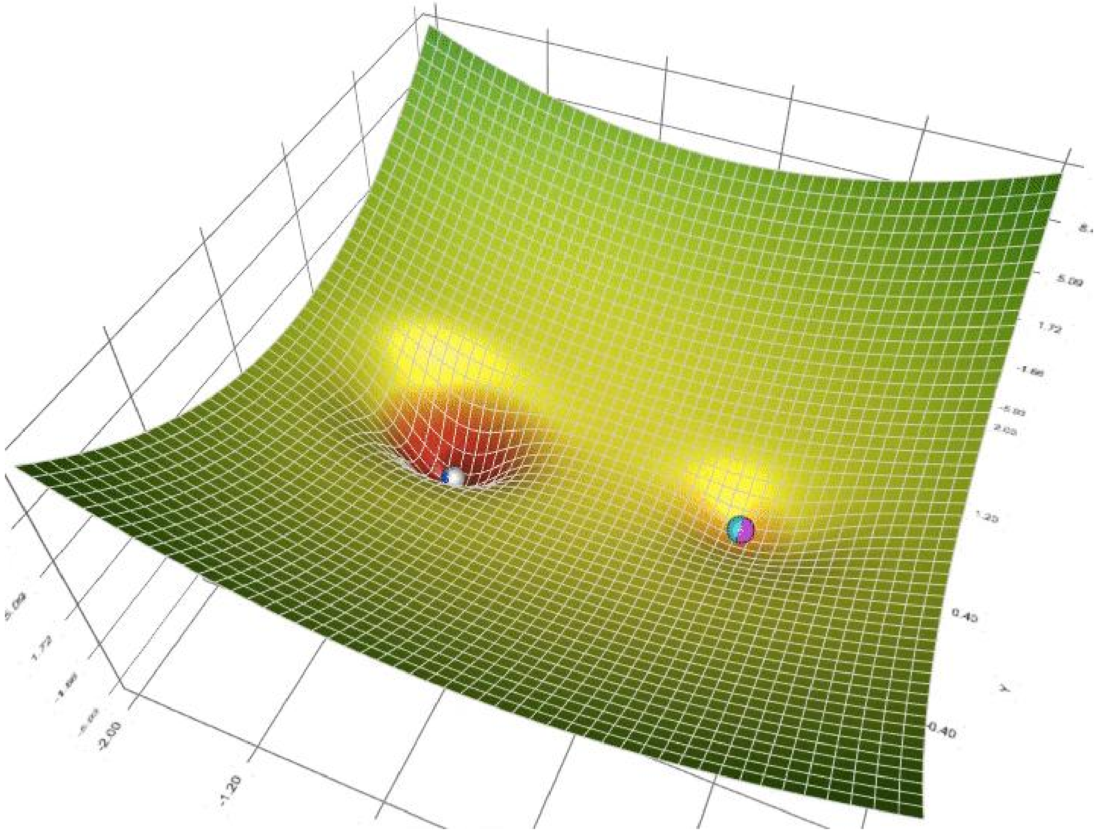
$$\begin{aligned}\frac{\mathrm{d}h}{\mathrm{d}t} &= \frac{\partial f}{\partial \boldsymbol{x}}\frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}t} \\ &= \begin{bmatrix} \frac{\partial f}{\partial x_1} & \frac{\partial f}{\partial x_2} \end{bmatrix} \begin{bmatrix} \frac{\mathrm{d}x_1}{\mathrm{d}t} \\ \frac{\mathrm{d}x_2}{\mathrm{d}t} \end{bmatrix} \\ &= \begin{bmatrix} \exp(x_1 x_2^2) x_2^2 & 2\exp(x_1 x_2^2) x_1 x_2 \end{bmatrix} \begin{bmatrix} \cos t - t\sin t \\ \sin t + t\cos t \end{bmatrix}\end{aligned}$$

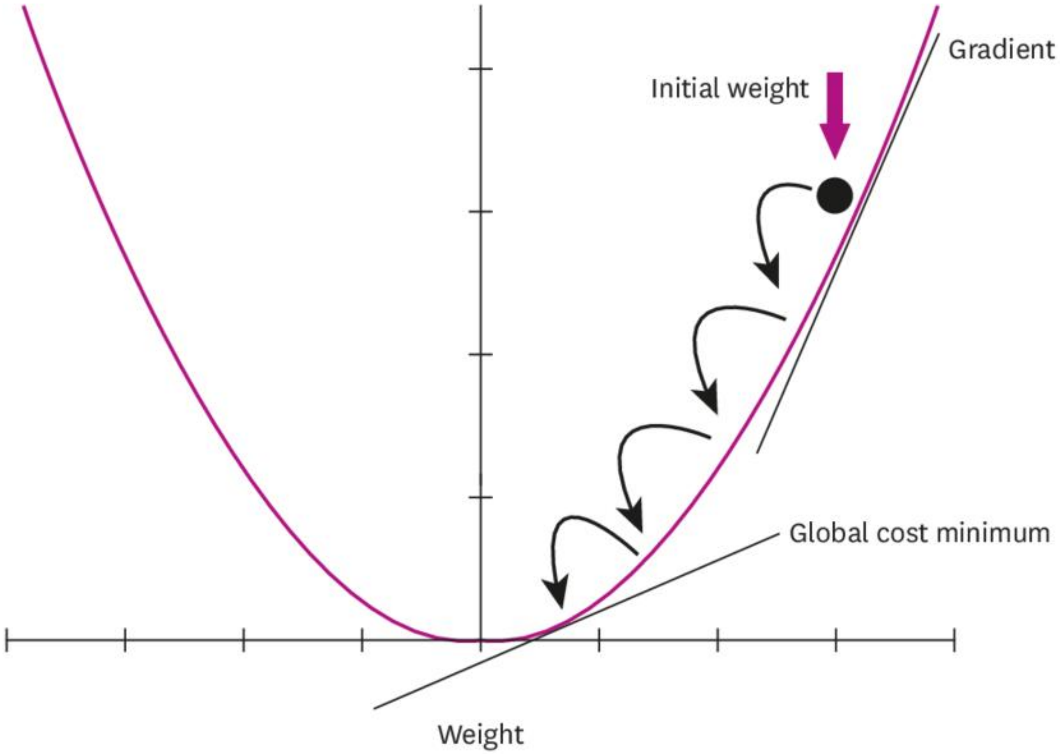
$$\frac{\partial}{\partial \boldsymbol{x}}\big(f(\boldsymbol{x})g(\boldsymbol{x})\big) = \frac{\partial f}{\partial \boldsymbol{x}} g(\boldsymbol{x}) + f(\boldsymbol{x})\frac{\partial g}{\partial \boldsymbol{x}}$$

$$\frac{\partial}{\partial \boldsymbol{x}}\big(f(\boldsymbol{x}) + g(\boldsymbol{x})\big) = \frac{\partial f}{\partial \boldsymbol{x}} + \frac{\partial g}{\partial \boldsymbol{x}}$$

$$\boxed{\frac{\partial}{\partial \boldsymbol{x}}(g \circ f)(\boldsymbol{x}) = \frac{\partial}{\partial \boldsymbol{x}}\big(g(f(\boldsymbol{x}))\big) = \frac{\partial g}{\partial f}\frac{\partial f}{\partial \boldsymbol{x}}}$$

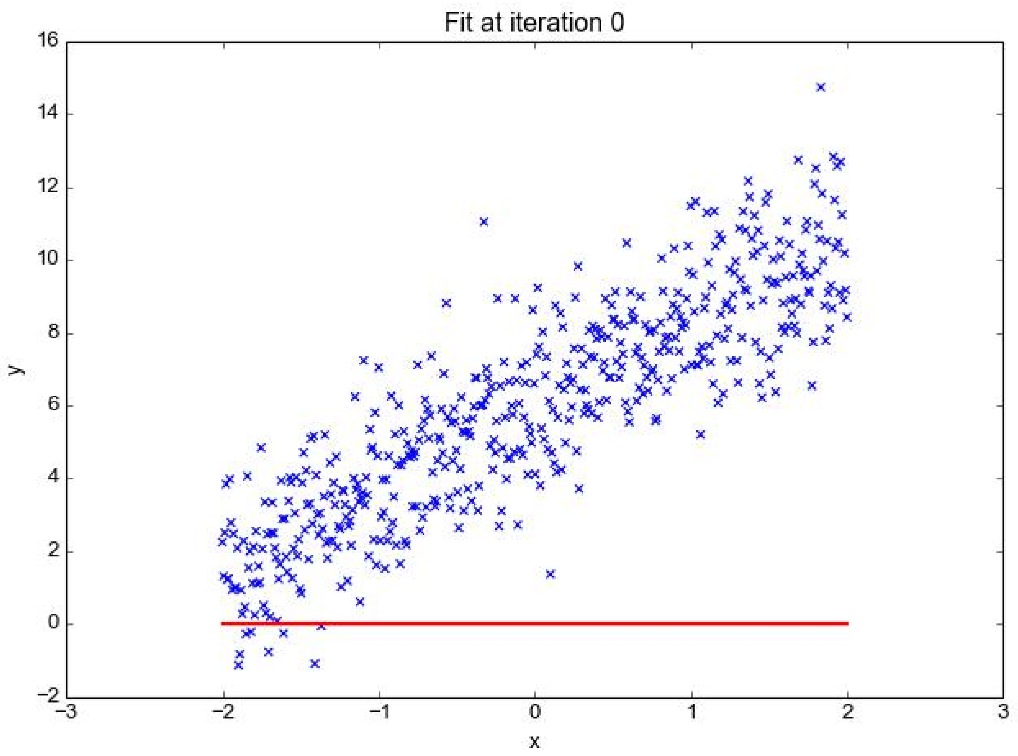
Gradient Descent

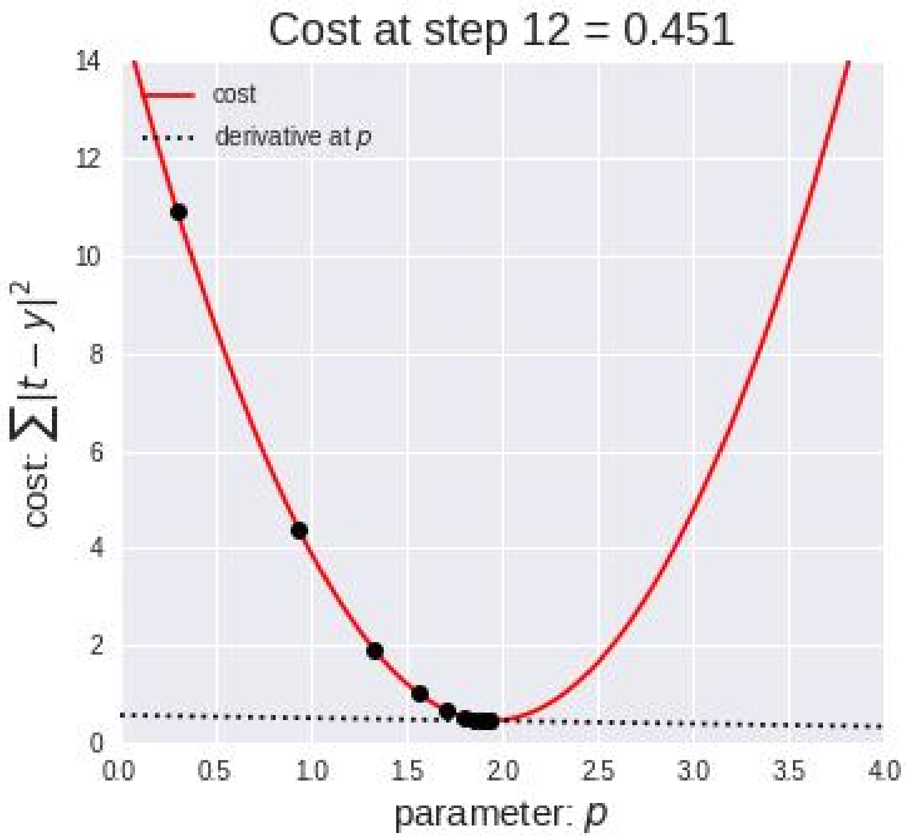




https://github.com/lilipads/gradient_descent_viz

Simulation of GD





A mathematical procedure for finding the best-fitting curve to a given set of points by minimizing the sum of the squares of the offsets.

Gradient

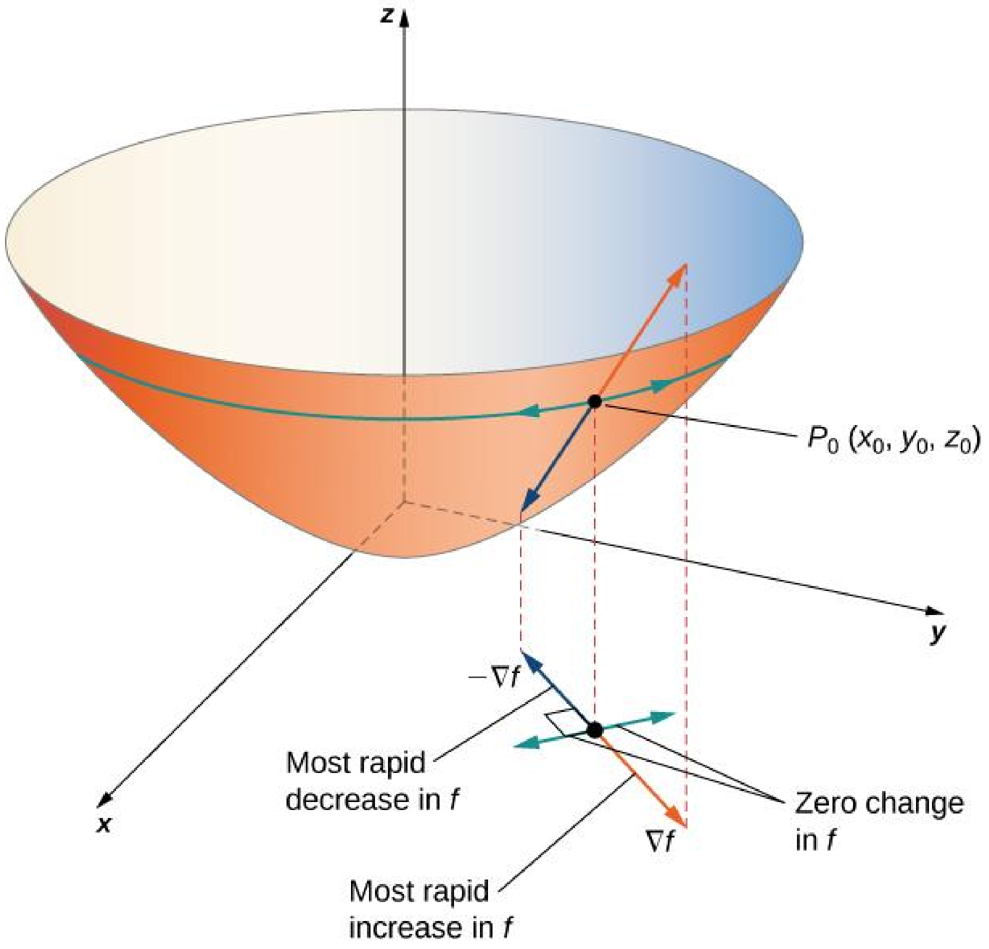
Consider the function $f(x, y) = x^2 + y^2$

We obtain the partial derivative $\partial f / \partial x$

$$\frac{\partial f(x,y)}{\partial x} = 2x \,.$$

Similarly, we obtain the partial derivative of f with respect to y

$$\frac{\partial f(x,y)}{\partial y} = 2y \,.$$



Least Mean Square

Residuals are the vertical distances between the data points and the corresponding predicted values.

$$S = \sum_{i=1}^{n} {r_i}^2 \qquad r_i = y_i - f(x_i, \boldsymbol{\beta}).$$

Solving the least squares problem:

$$\frac{\partial S}{\partial \beta_j} = 2\sum_i r_i \frac{\partial r_i}{\partial \beta_j} = 0,\ j = 1, \ldots, m,$$

$$= -2\sum_i r_i \frac{\partial f(x_i, \boldsymbol{\beta})}{\partial \beta_j} = 0,\ j = 1, \ldots, m.$$

Since $r_i = y_i - f(x_i, \boldsymbol{\beta})$

$$\sum_{j=1}^{n} X_{ij}\beta_j = y_i,\ (i = 1, 2, \ldots, m), \quad \mathbf{X}\boldsymbol{\beta} = \mathbf{y},$$

$$\mathbf{X} = \begin{bmatrix} X_{11} & X_{12} & \cdots & X_{1n} \\ X_{21} & X_{22} & \cdots & X_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ X_{m1} & X_{m2} & \cdots & X_{mn} \end{bmatrix}, \quad \boldsymbol{\beta} = \begin{bmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_n \end{bmatrix}, \quad \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix}$$

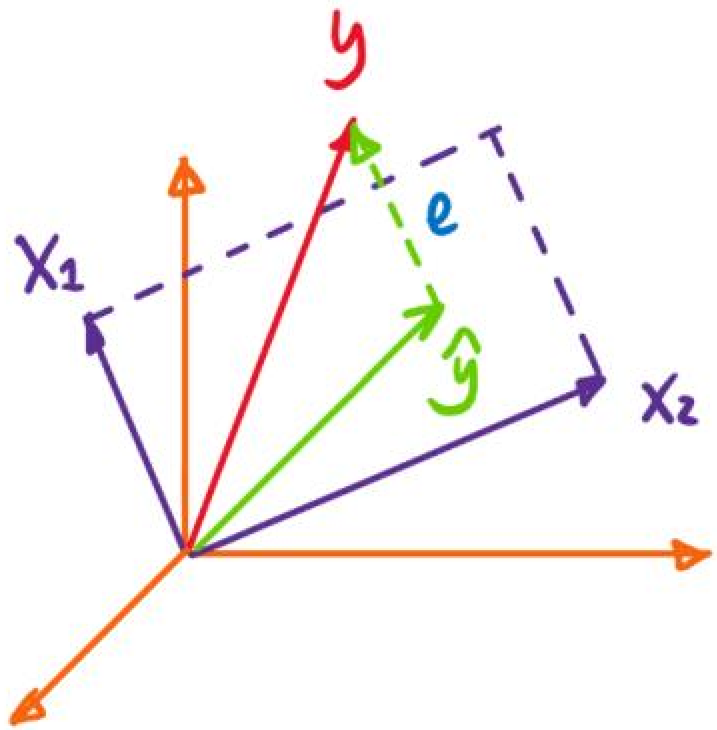
$$\hat{\boldsymbol{\beta}} = \underset{\boldsymbol{\beta}}{\arg\min}\, S(\boldsymbol{\beta}),$$

$$S(\boldsymbol{\beta}) = \sum_{i=1}^{m} \left| y_i - \sum_{j=1}^{n} X_{ij}\beta_j \right|^2 = \left\| \mathbf{y} - \mathbf{X}\boldsymbol{\beta} \right\|^2$$

$$(\mathbf{X}^{\mathrm{T}}\mathbf{X})\hat{\boldsymbol{\beta}} = \mathbf{X}^{\mathrm{T}}\mathbf{y}.$$

We can obtain: $\hat{\boldsymbol{\beta}} = (X^T X)^{-1} X^T \boldsymbol{y}.$

Geometrical View of OLS



See the Lecture Note:

Geometrical Interpretation of OLS

Bayesian Probabilistic Interpretation

- Let us assume that the target variables and the inputs are related via the equation.

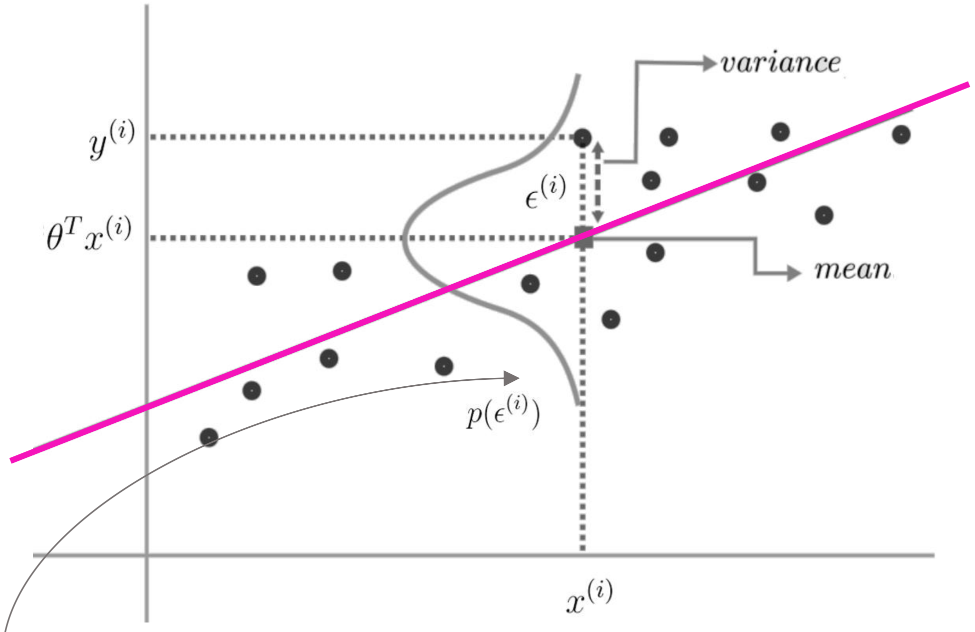
$$\boxed{y^{(i)} = \theta^T x^{(i)} + \epsilon^{(i)}}$$

where the error can be regarded as noise that follows a Gaussian distribution.

$$\boxed{p(\epsilon^{(i)}) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(\epsilon^{(i)})^2}{2\sigma^2}\right)}$$

Easily, we can pose the model like:

$$\mu_i = b_0 + b_1 x_i$$
$$y_i \sim \mathcal{N}(\mu_i, \varepsilon)$$



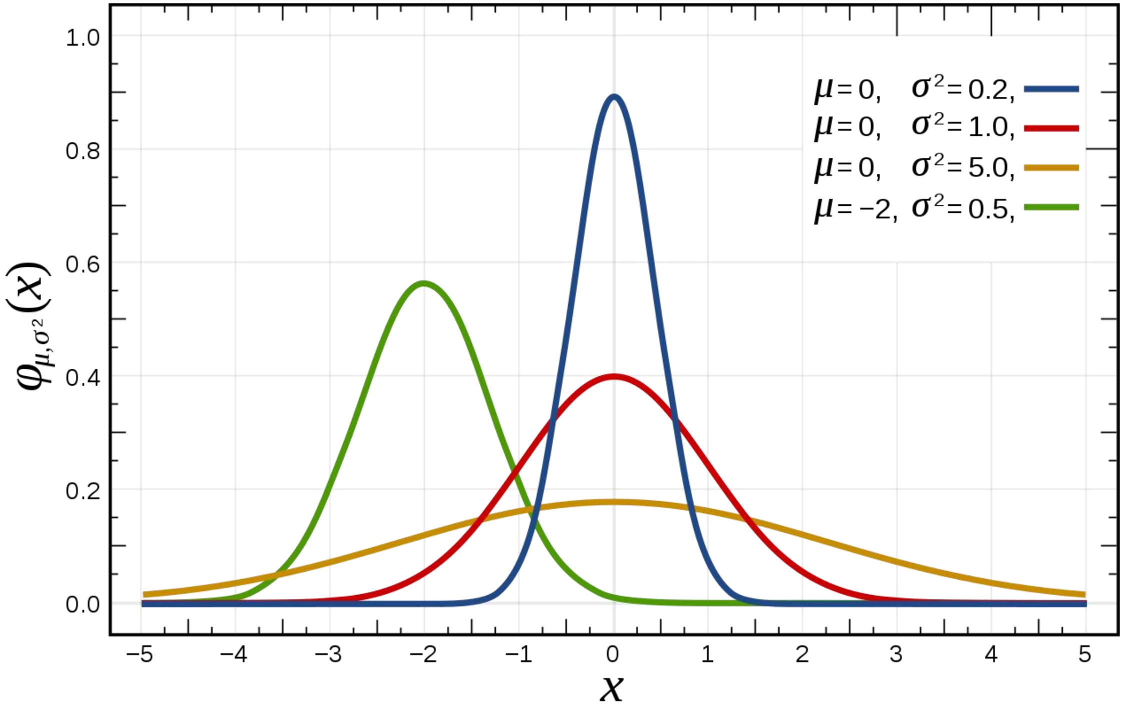
$$\ln \mathcal{L}(\theta\,; x_1, \ldots, x_n) = \sum_{i=1}^{n} \ln f(x_i \mid \theta),$$

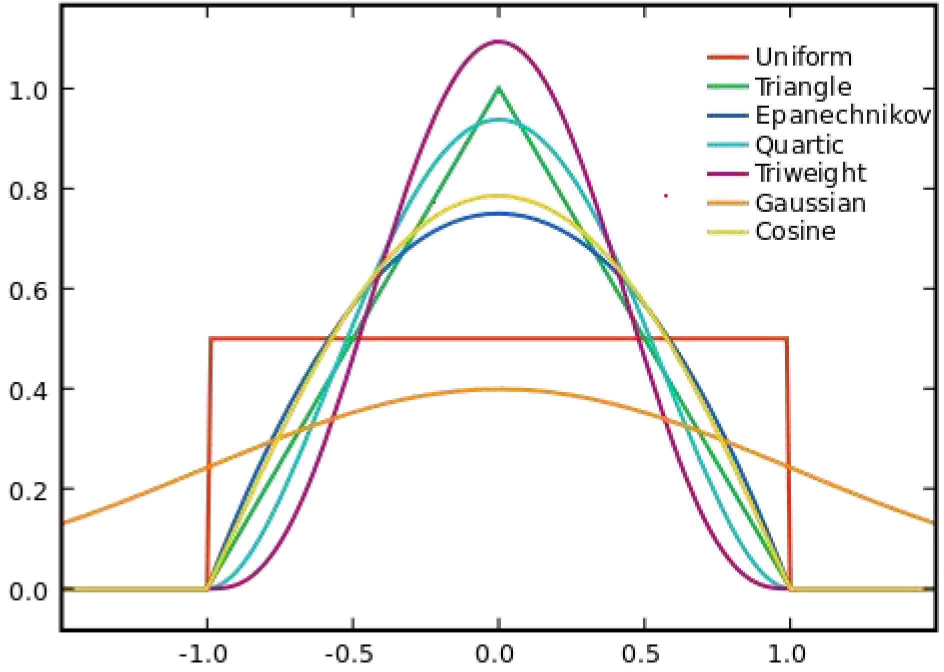
Parametric Model

Parametric models assume some finite set of parameters θ. Given the parameters, future predictions, x, are independent of the observed data, $\mathcal{D}$:

$$P(x|\theta, \mathcal{D}) = P(x|\theta)$$

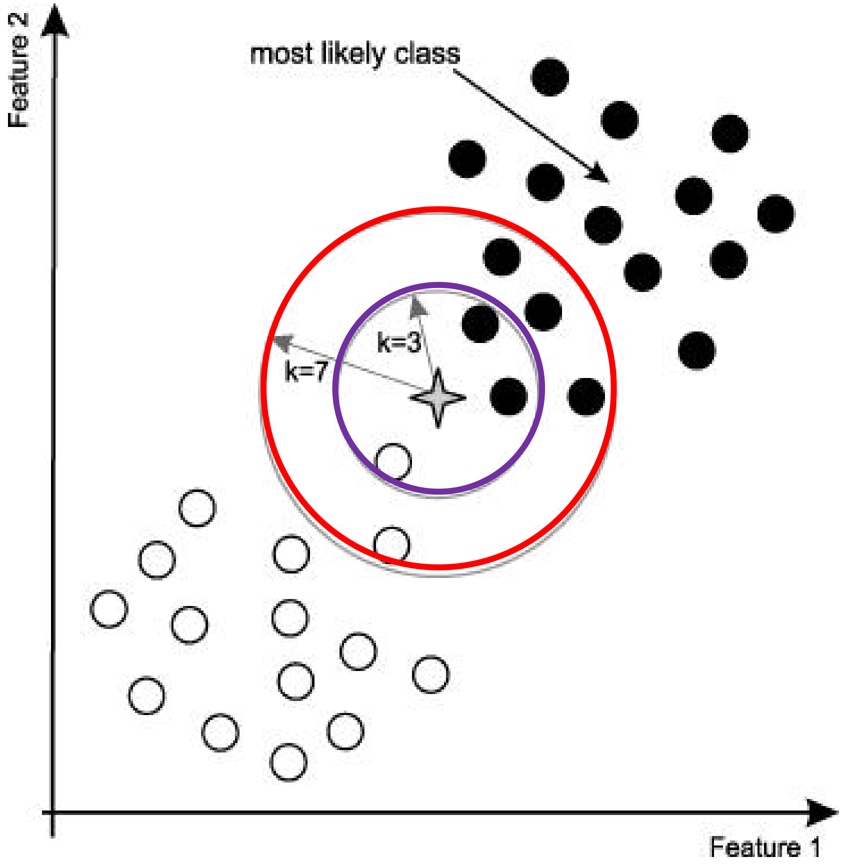
therefore θ capture everything there is to know about the data.





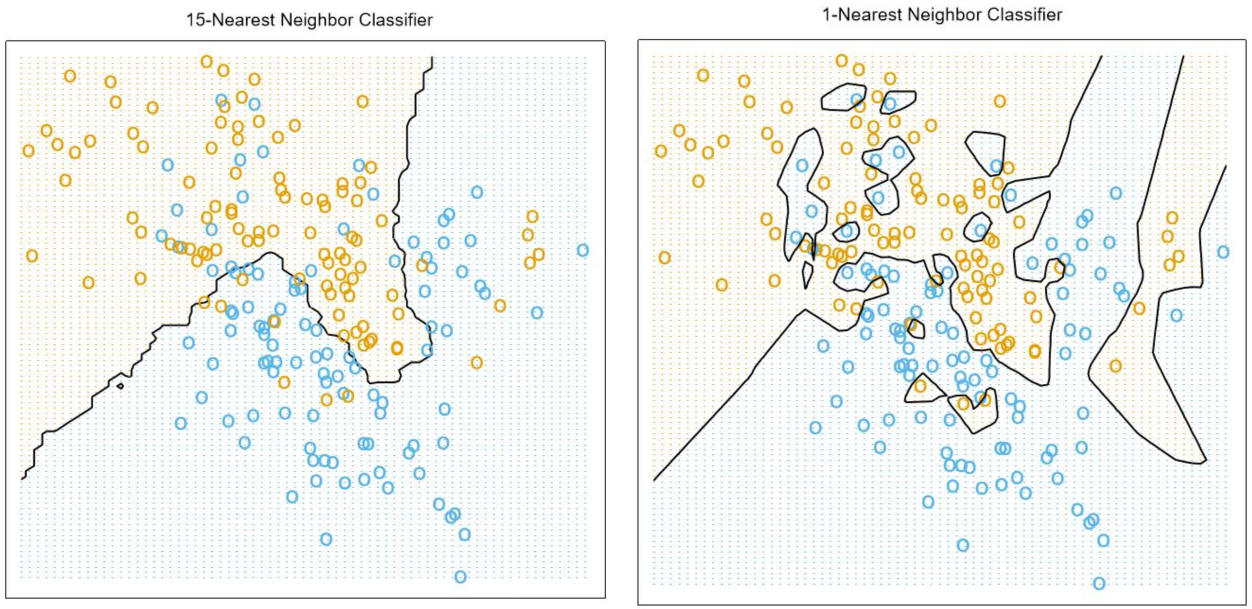
K-NN and K-Means

K-Nearest Neighbors

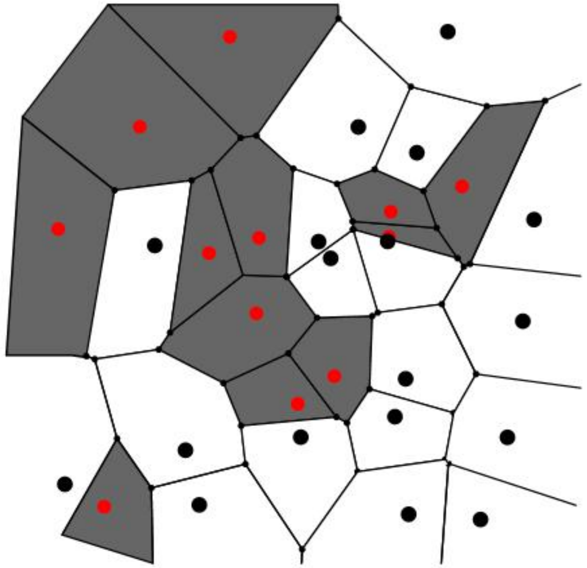


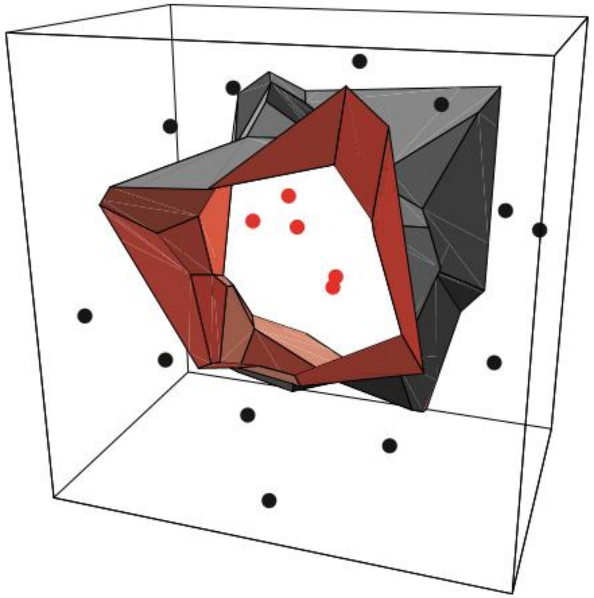
k*-nearest neighbors algorithm** (k*-NN**) is a non-parametric method used for classification and regression. The input consists of the *k* closest training examples in the feature space.

Q: *k* has to be an odd number?



Decision Boundary of KNN

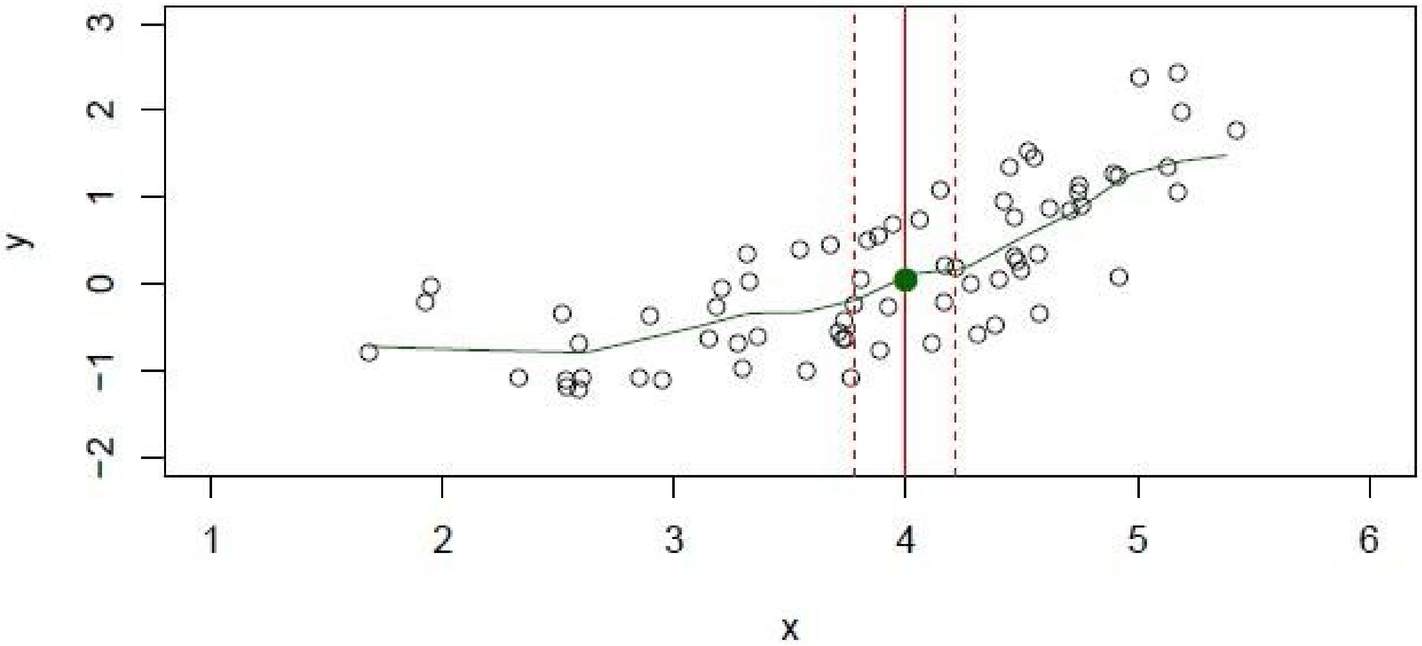




Q： What if K becomes very large?

In two dimensions, the nearest-neighbor algorithm leads to a partitioning of the input space into Voronoi cells, each label led by the category of the training point it contains. In three dimensions, the cells are three-dimensional, and the decision boundary resembles the surface of a crystal.

Neighbors for Prediction?

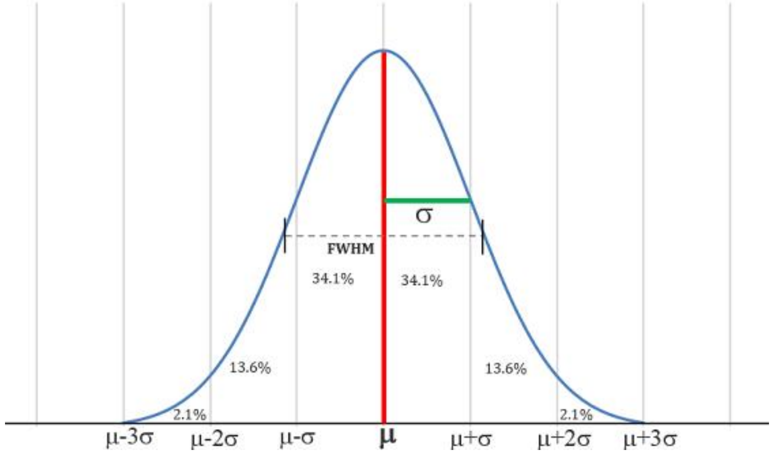


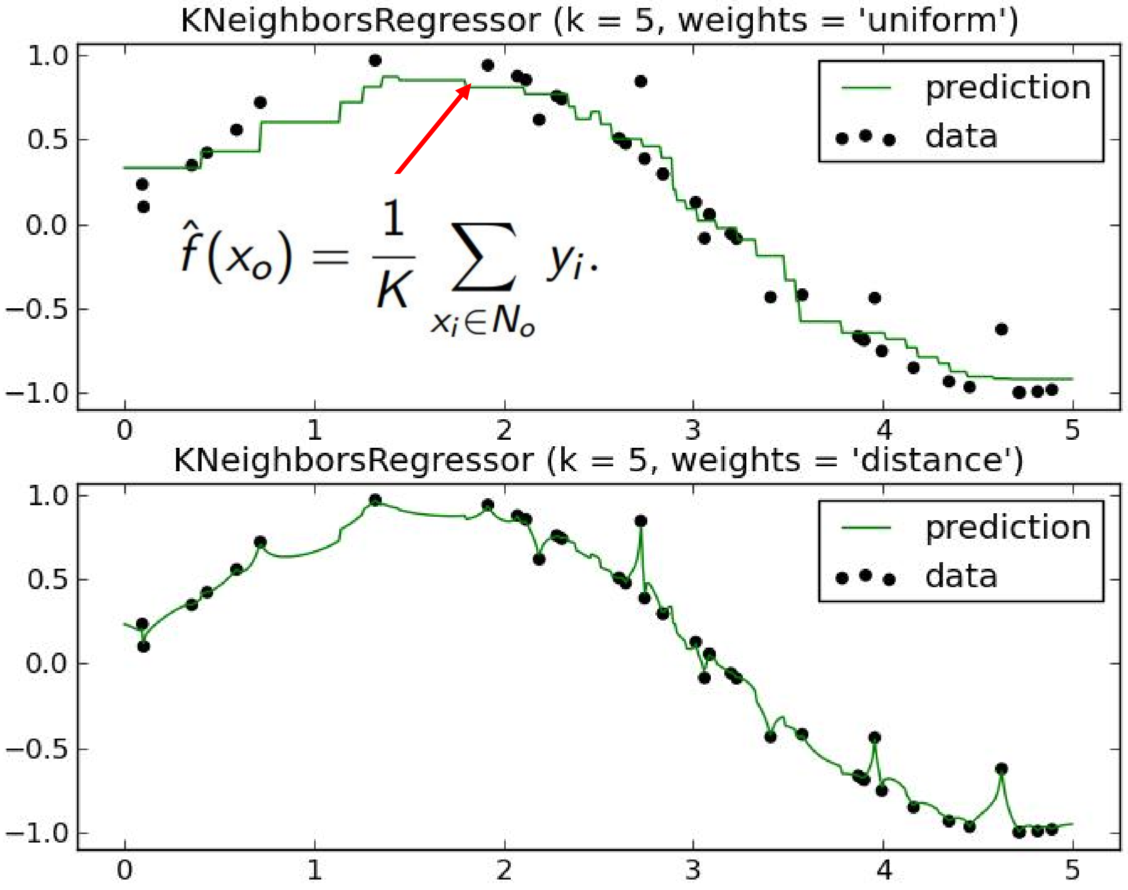
1. Assume a value for the number of nearest neighbors K and a prediction point *x*.
2. KNN identifies the training observations N closest to the prediction point *x*.
3. KNN estimates *f* (*x*) using the average of all the responses in N neighboring points

KNN Regression with Kernels

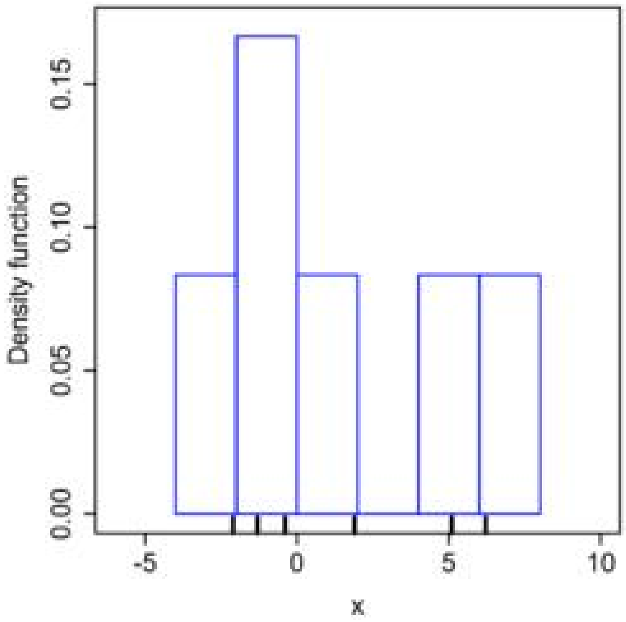
Q: Any better non-parametric model, do we need to adjust the weights?

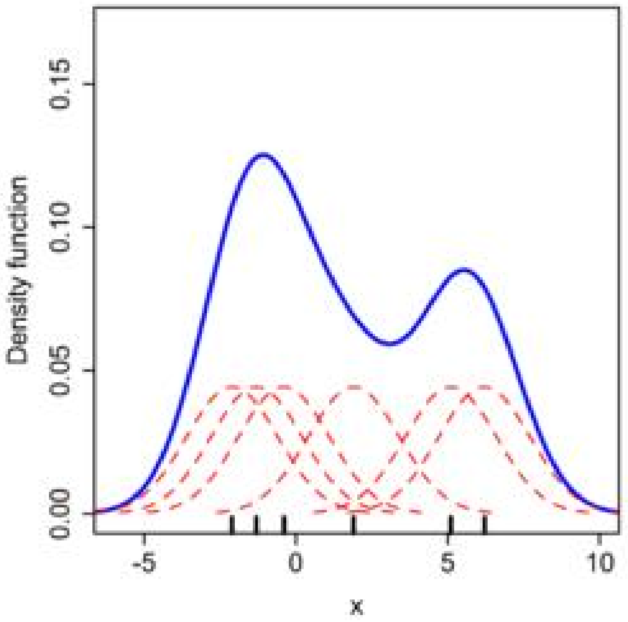
$$k(x,y) = \exp\left(-\frac{\|x-y\|^2}{2\sigma^2}\right)$$



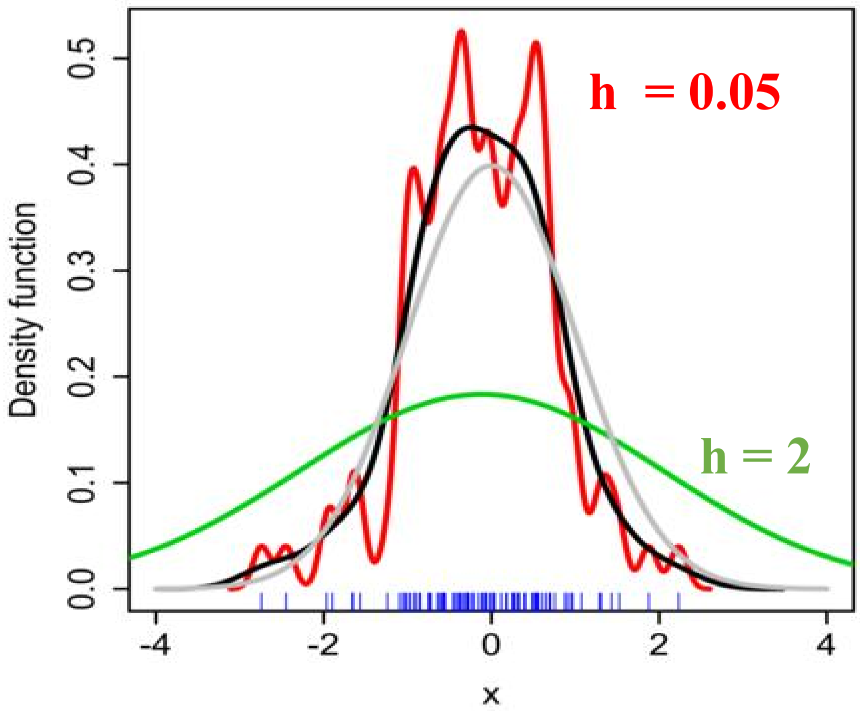


Kernel Density Estimation





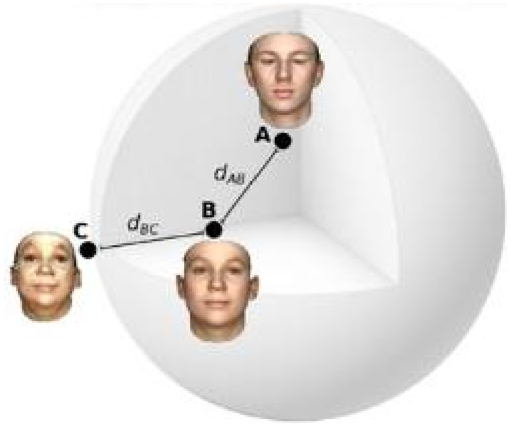
$$\widehat{f}_h(x) = \frac{1}{n}\sum_{i=1}^{n} K_h(x - x_i) = \frac{1}{nh}\sum_{i=1}^{n} K\Big(\frac{x - x_i}{h}\Big),$$



The bandwidth of the kernel is a free parameter which exhibits a strong influence on the resulting estimate.

https://www.wikiwand.com/en/Kernel_density_estimation

Distance from Dissimilarity



The similarity measure is usually expressed as a numerical value: It gets higher when the data samples are more alike. It is often expressed as a number between zero and one by conversion. The quantitative representation of objects (or events) decides how the distances are evaluated.

Distance Measure

When we find a way of representing objects, we can always find a proper the distance measure.

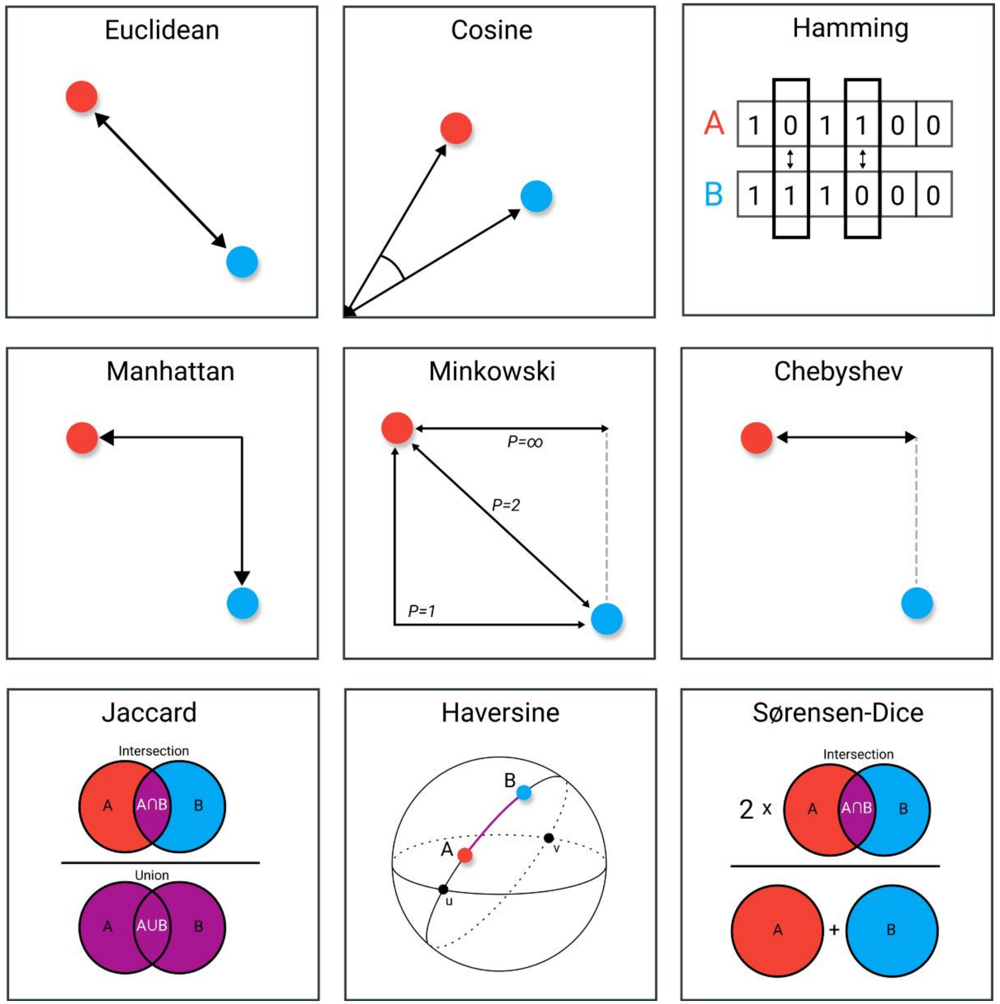
$d(A,B) = d(B,A)$ Symmetry

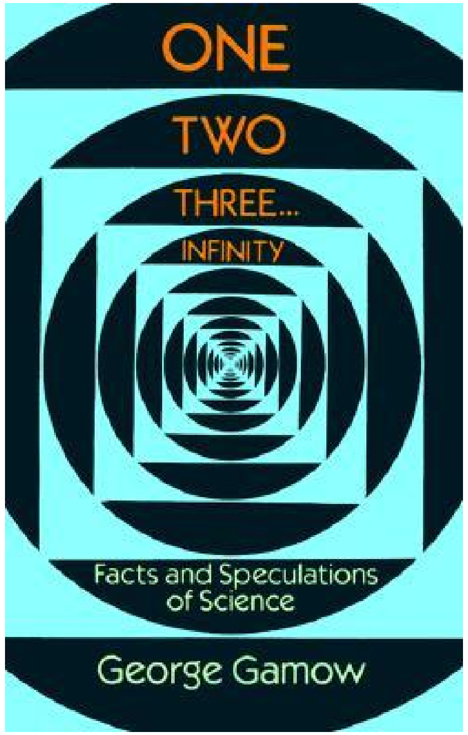
$d(A,A) = 0$ Constancy of Self-Similarity

$d(A,B) = 0$ iff $A=B$ Positivity Separation

$d(A,B) \leq d(A,C)+d(B,C)$ Triangular Inequality

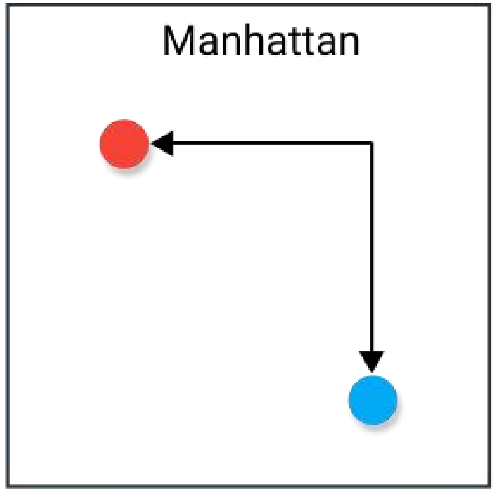
Learning

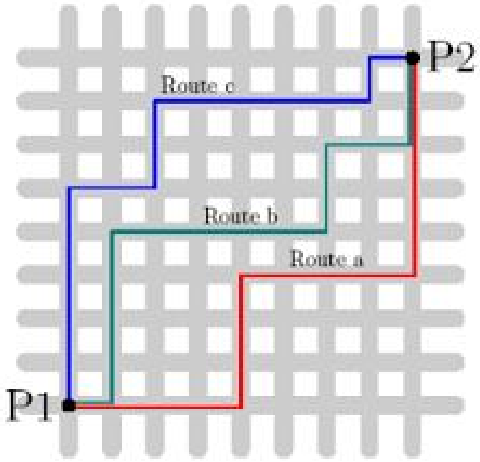


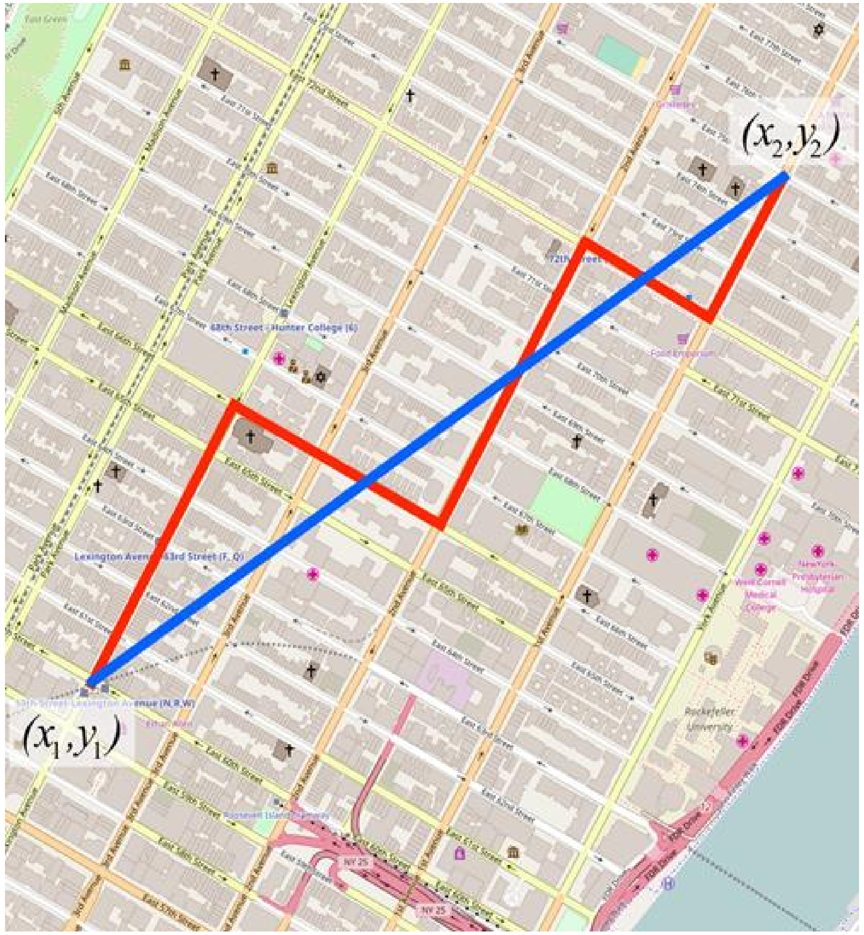


George Gamow (1904 –1968)

Manhattan Distance



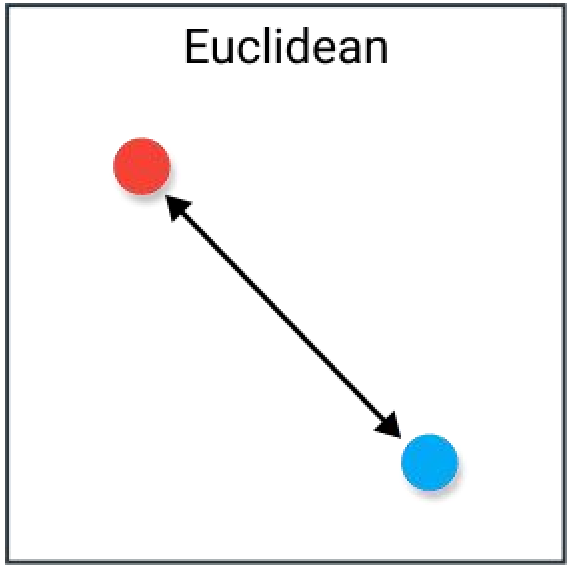


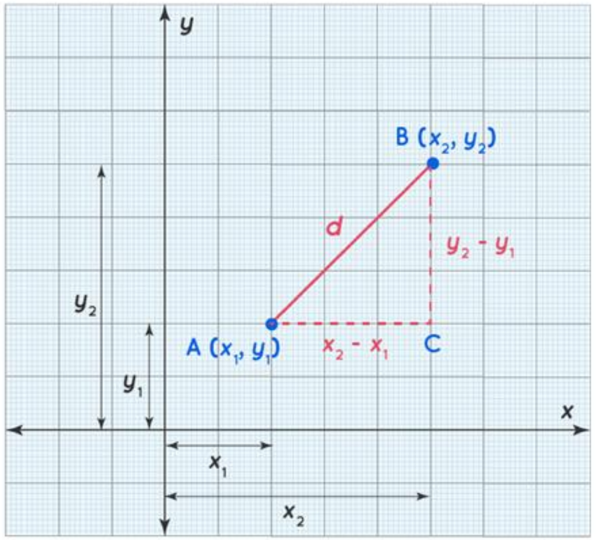


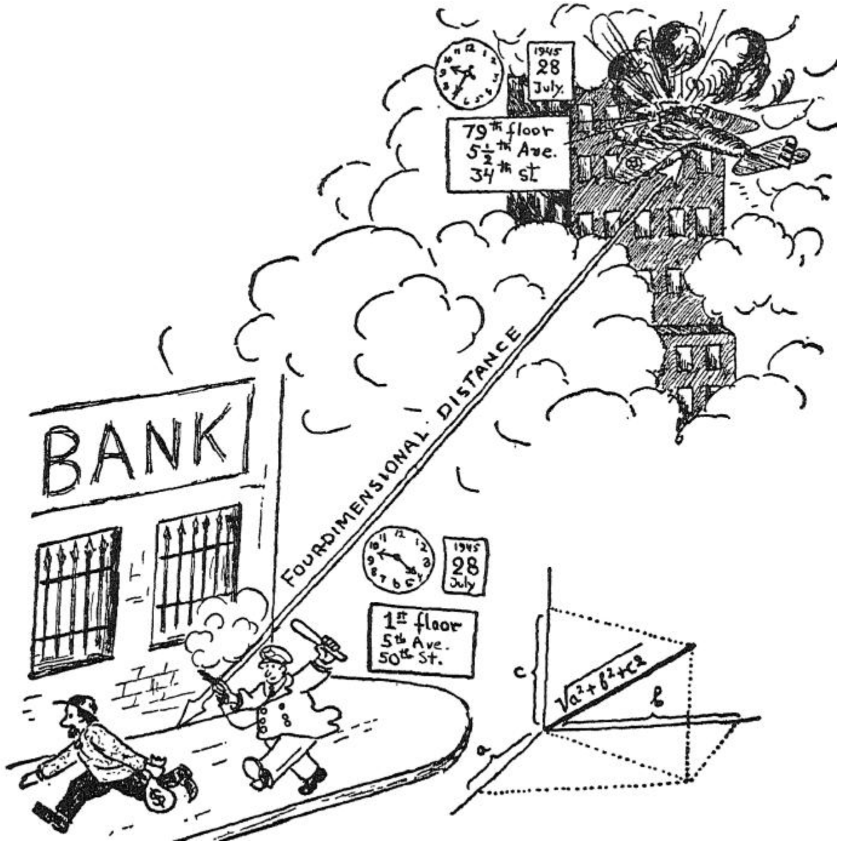
Manhattan distance is **a distance metric between two points in a *N* dimensional vector space**. It is the sum of the lengths of the projections of the line segment between the points onto the coordinate axes.

$$D(x, y) = \sum_{i=1}^{k} |x_i - y_i|$$

Euclidean Distance





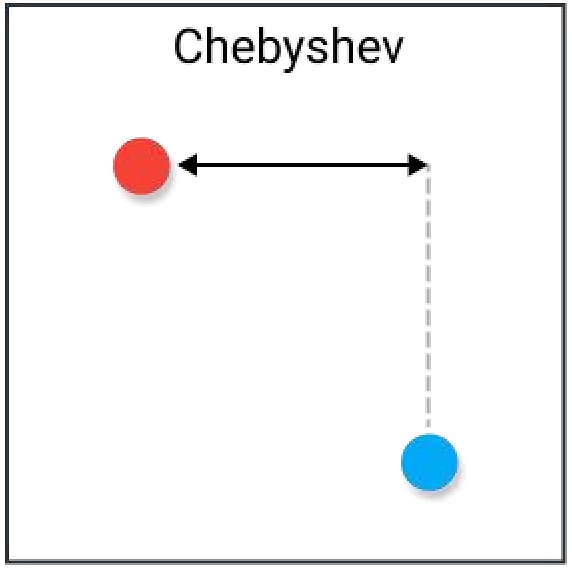


How to calculate 4th dimensional distance of two events?

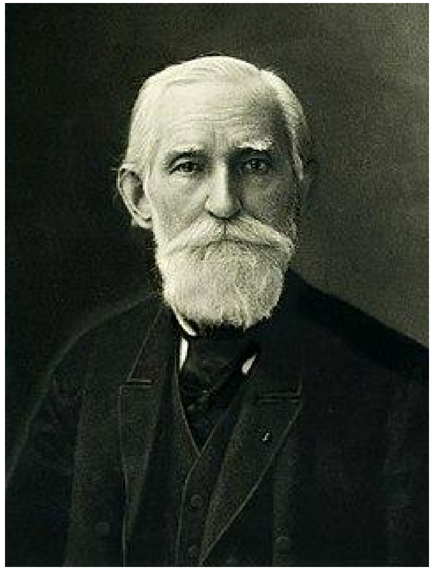
$$D(x, y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2)}$$

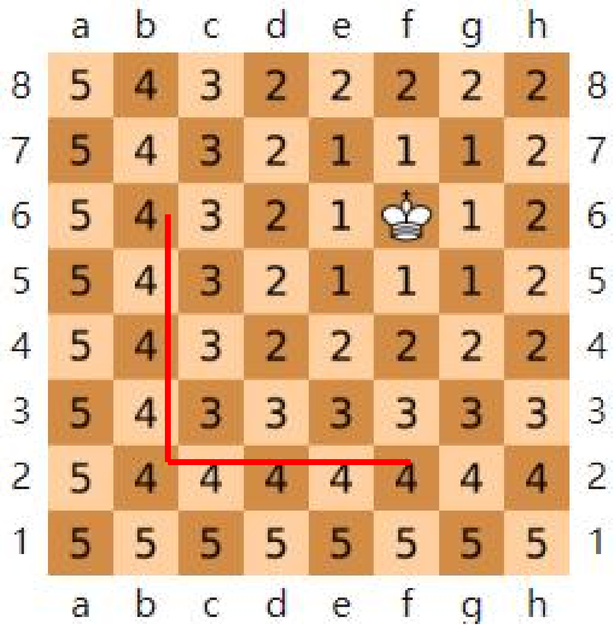
https://towardsdatascience.com/9-distance-measures-in-data-science-918109d069fa

Chebyshev Distance



Pafnuty Lvovich (1821-1894) Chebyshev was considered to be the founding father of Russian mathematics.

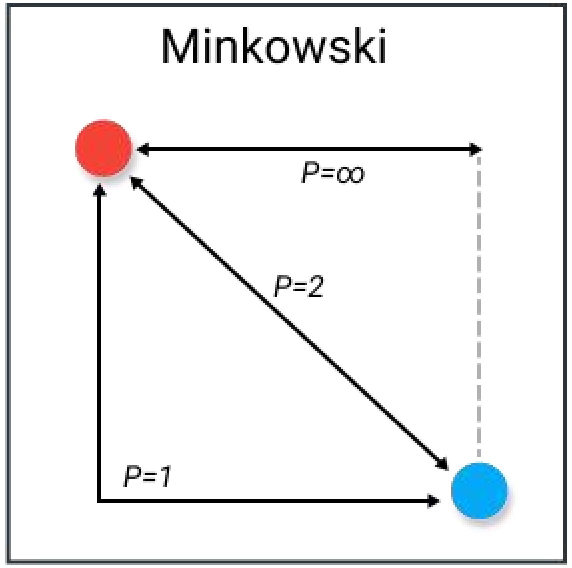




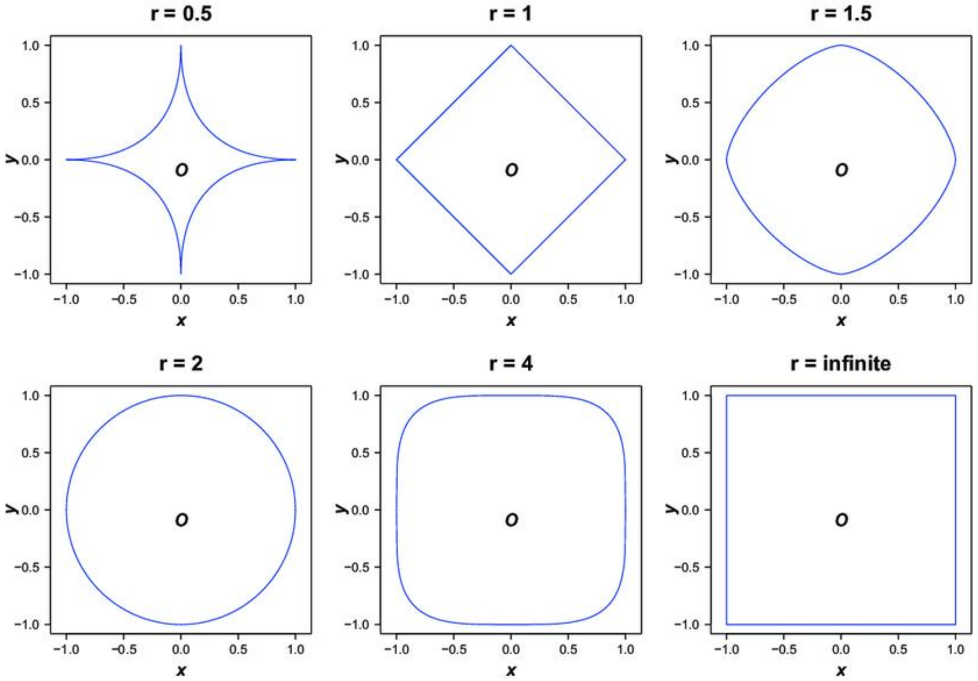
$$D(x, y) = \max_i (|x_i - y_i|)$$

The discrete Chebyshev distance between two spaces on a chess board gives the minimum number of moves a king requires to move between them. This is because a king can move diagonally, so that the jumps to cover the larger.

Minkowski Distance



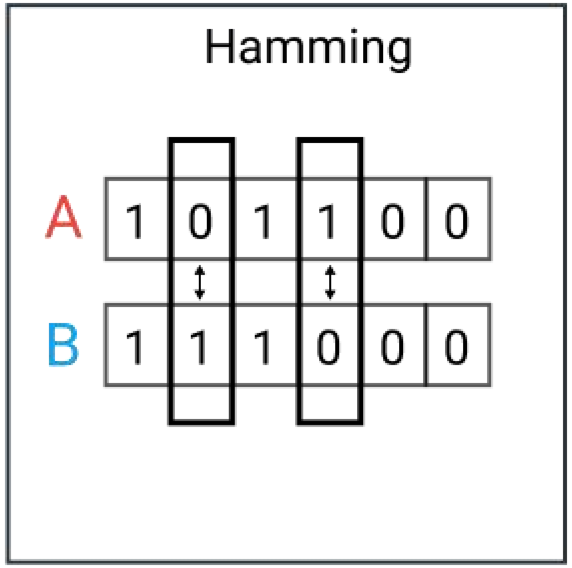
Hermann Minkowski

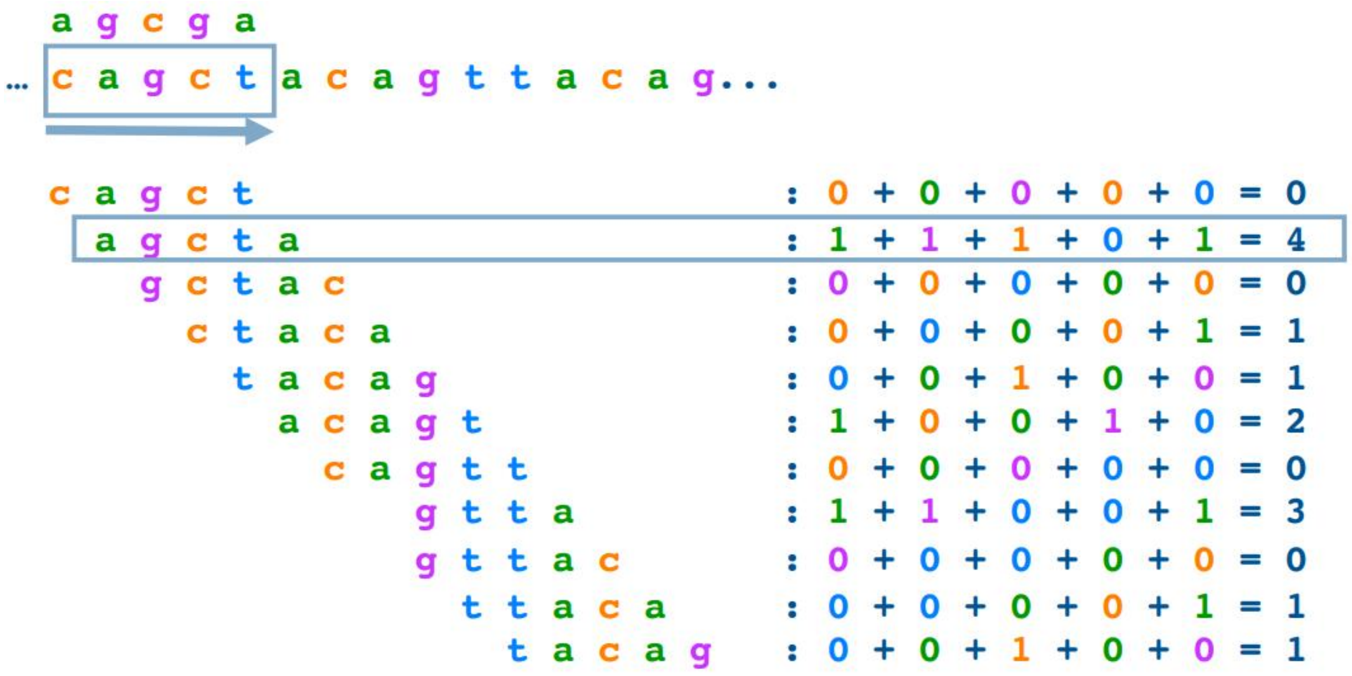


$$D(x,y) = \left(\sum_{i=1}^{n} |x_i - y_i|^p \right)^{\frac{1}{p}}$$

Introduced by his former student Albert Einstein in 1905, Minkowski realized that the special theory of relativity could best be understood in a four-dimensional "Minkowski spacetime", in which time and space are not separated entities but intermingled in a four-dimensional space.

Hamming Distance

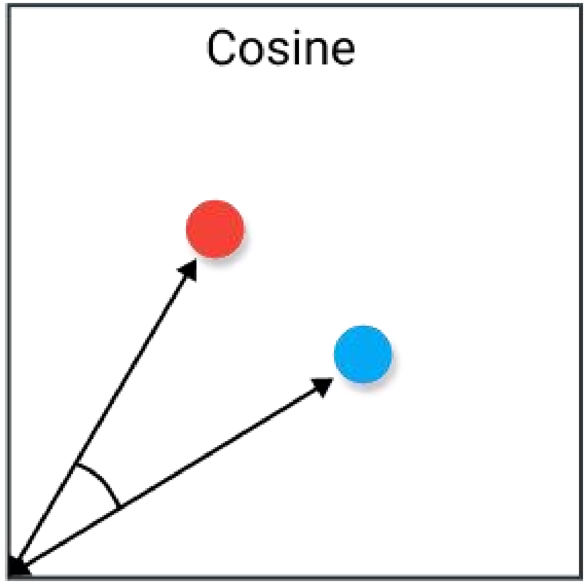


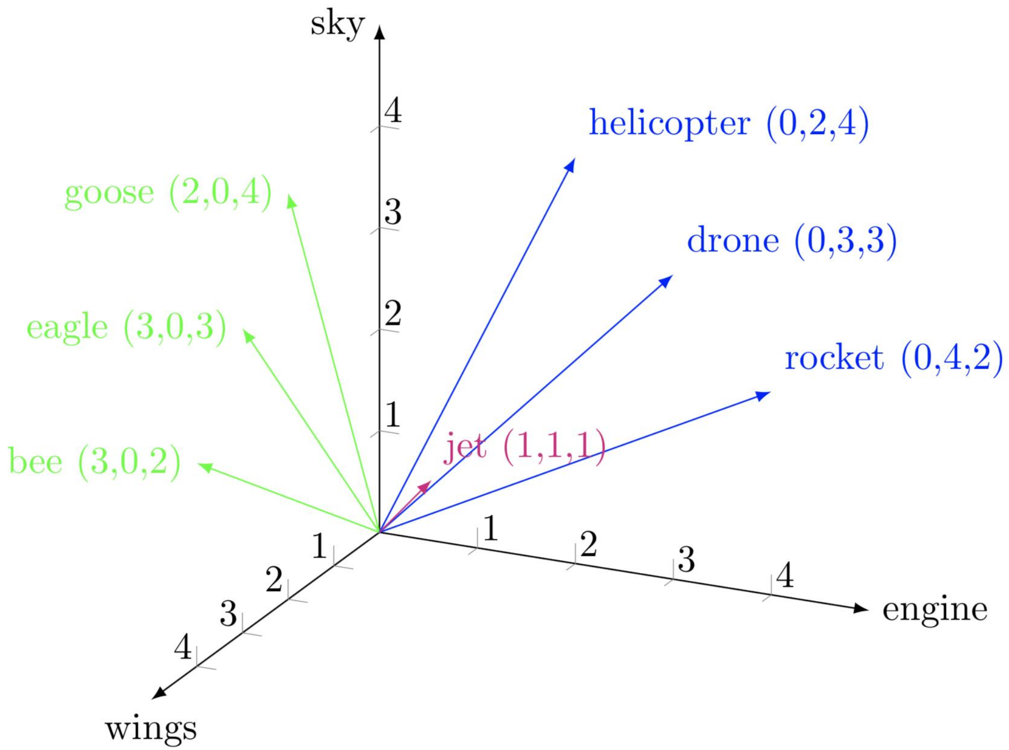


It is typically used to compare two binary strings of equal length.

Hamming distance can be generalized to any discrete values, e.g., the example above is about the 'negative Hamming distance applied of genes: *score = 5-HammingDis(x,y)*

Cosine Distance



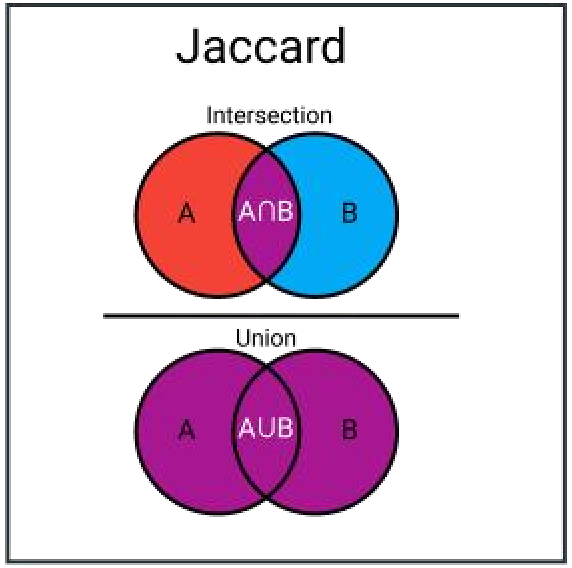


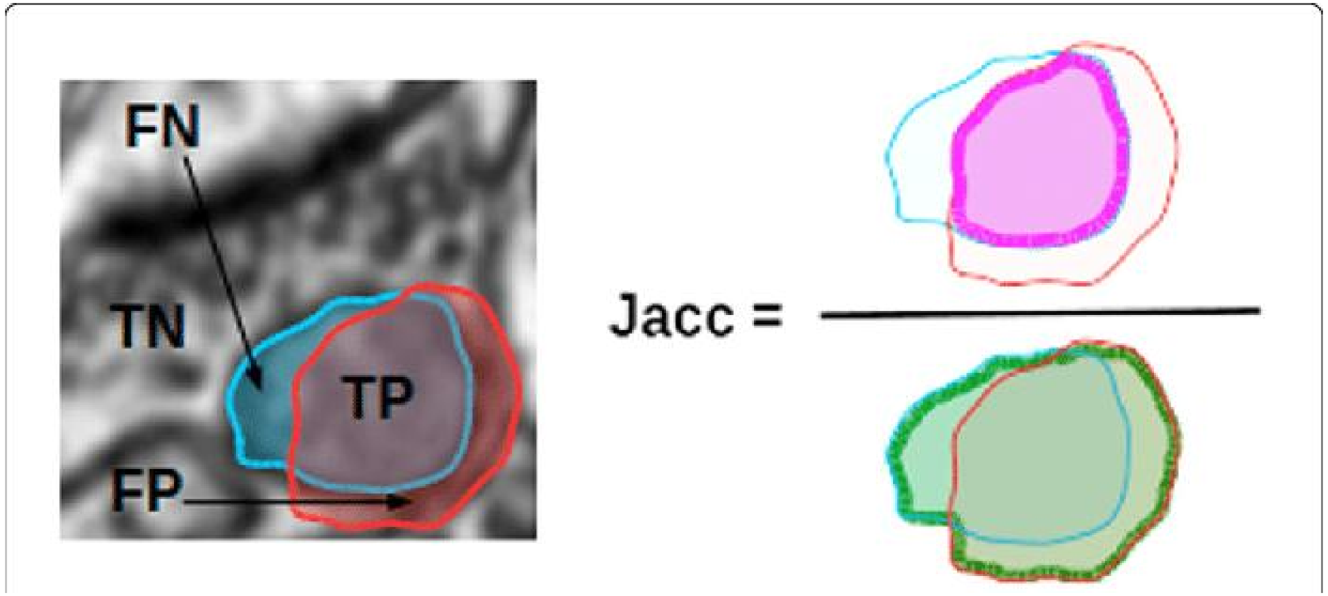
$$D(x, y) = cos(\theta) = \frac{x \cdot y}{\|x\| \, \|y\|}$$

Word embeddings formulate a semantic space in which calculatable cosine distance can represent the semantic difference between words.

https://corpling.hypotheses.org/495

Jaccard Index

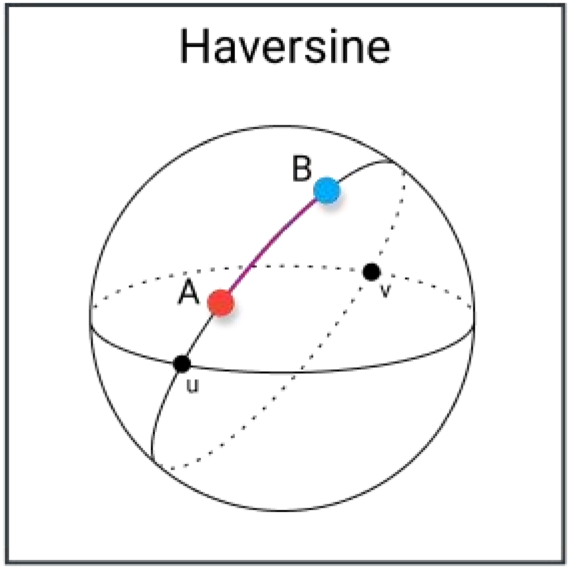




$$D(x, y) = 1 - \frac{|x \cap y|}{|y \cup x|}$$

When you have a deep learning model predicting segments of an image, for instance, a car, the Jaccard index can then be used to calculate how accurate that predicted segment given true labels.

Haversine



$$d = 2r \arcsin\left(\sqrt{\sin^2\left(\frac{\phi_2 - \phi_1}{2}\right) + \cos(\phi_1)\cos(\phi_2)\sin^2\left(\frac{\lambda_2 - \lambda_1}{2}\right)}\right)$$

As you might have expected, Haversine distance is often used in navigation. For example, you can use it to calculate the distance between two countries when flying between them.

https://towardsdatascience.com/17-types-of-similarity-and-dissimilarity-measures-used-in-data-science-3eb914d2681

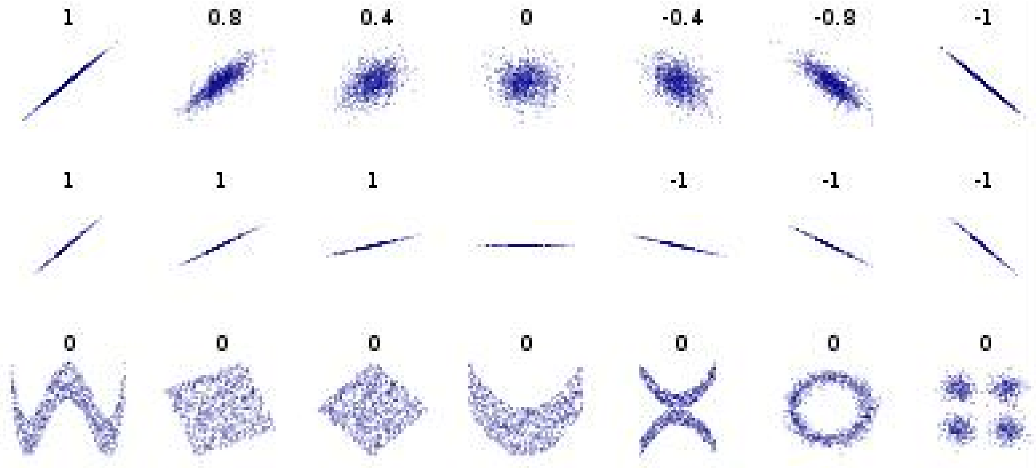
Distance between Uncertain Concepts

Pearson's correlation coefficient:

$$r_{xy} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}}$$

Kullback-Leibler Divergence:

$$D_{KL}(p||q) = \int_x p(x) log \frac{p(x)}{q(x)} dx$$

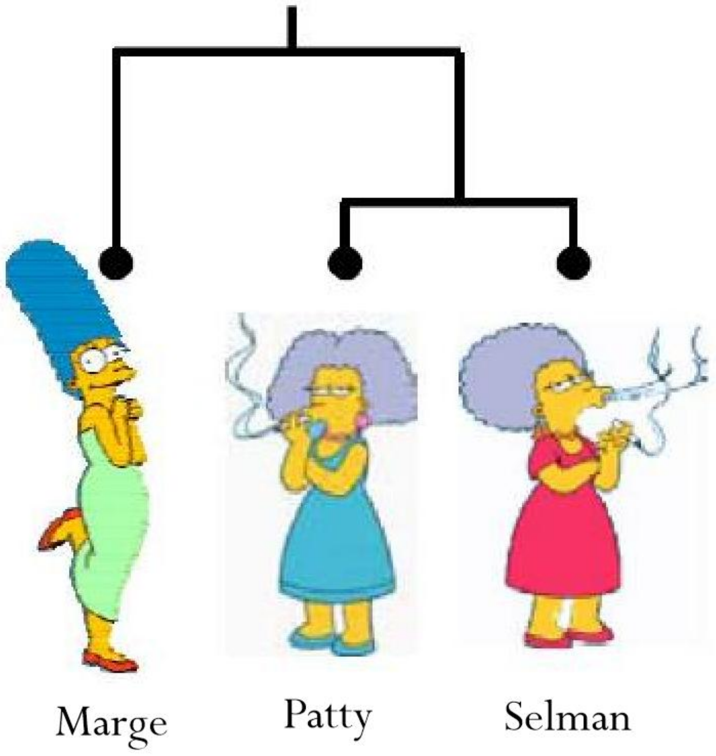


Cross Entropy:

$$L = -\frac{1}{m}\sum_{i=1}^{m} y_i \cdot \log(\hat{y}_i)$$

Edited Distance

To measure the similarity between two objects, transform one into the other, and measure how much effort it took. The measure of effort becomes the distance measure.



The distance between Marge and Selma

Change dress color, 1 point
Add earrings, 1 point
Decrease height, 1 point
Take up smoking, 1 point
Loss weight, 1 point

D(Marge, Selma) = 5

The distance between Patty and Selma.

Change dress color, 1 point
Change earring shape, 1 point
Change hair part, 1 point

D(Patty,Selma) = 3

References

[1]https://med.libretexts.org/Bookshelves/Pharmacology_and_Neuroscience/Book%3A_Computational_Cognitive_Neuroscience_(O%27Reilly_and_Munakata)/6%3A_Preception_and_Attention/6.3%3A_Oriented_Edge_Detectors_in_Primary_Visual_Cortex

[2] https://people.eecs.berkeley.edu/~malik/papers/SM-ncut.pdf