Introduction to Machine Learning Algorithms

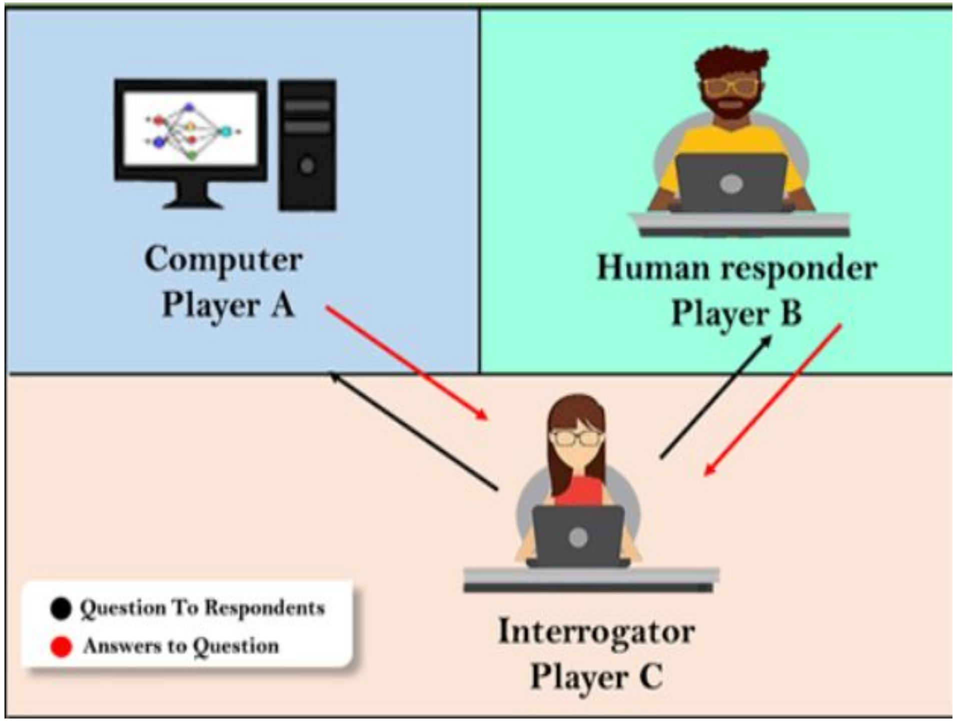
Samuel Edet

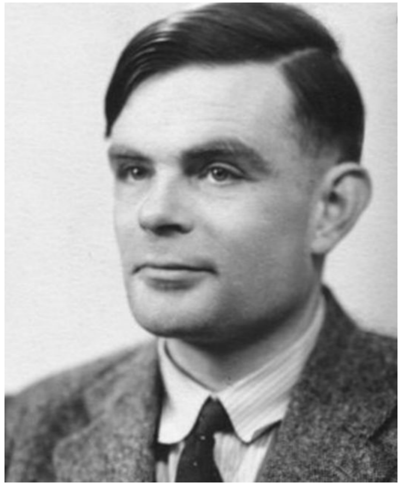
PhD Student, System Science and Business Economics

Agenda

Can Machine Think?

Discourse on the method published 1637.



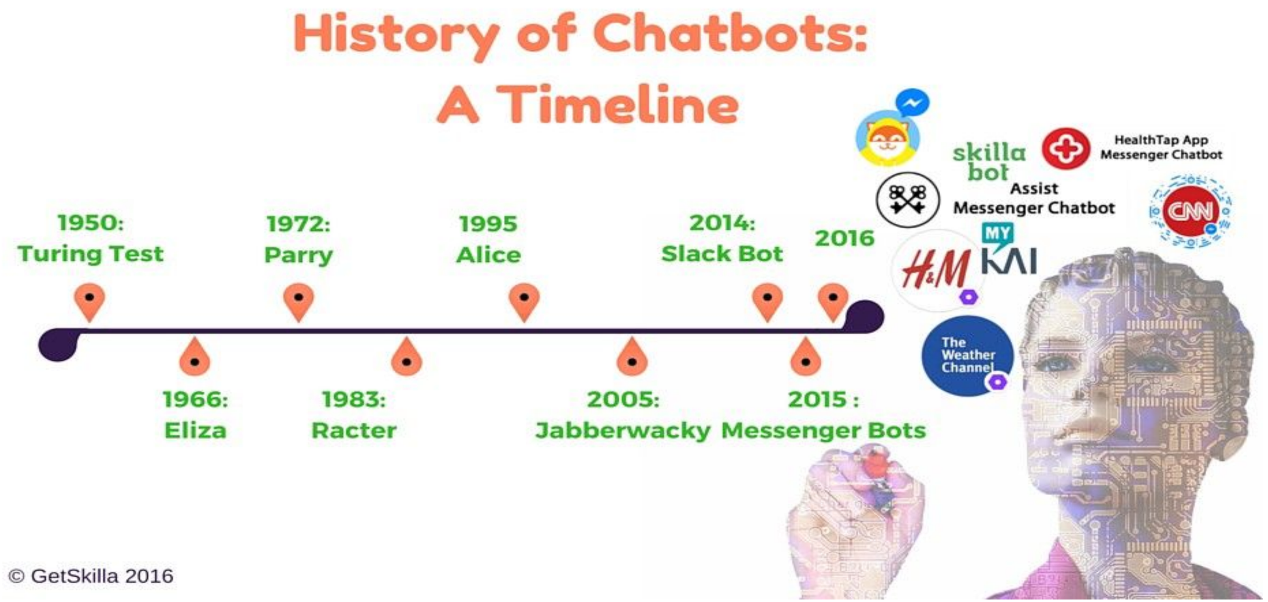


Computing Machinery and Intelligence published 1950

IMT SCUOLA ALTI STUDI LUCCA

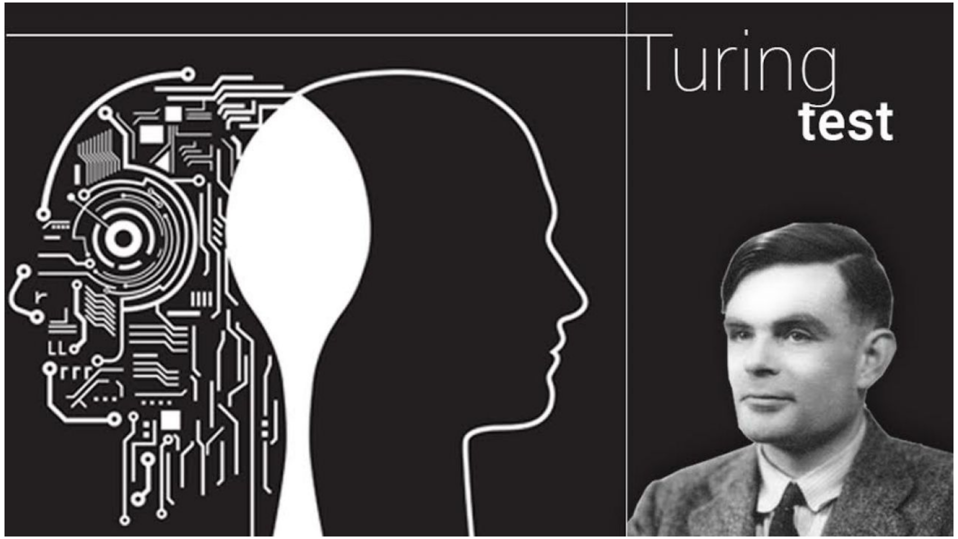
KU LEUVEN

Can Machine Think?



Can Machine Think?

WHAT IS THE TASK?
SUPERVISED OR
UNSUPERVISED



WHAT LEARNING
RULES CAN I APPLY?

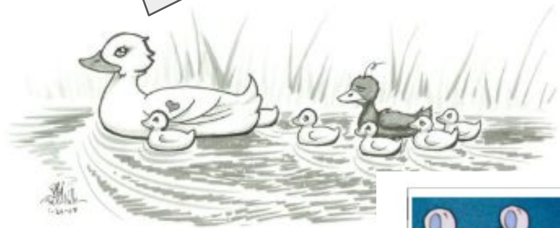
WHAT FEATURES WILL
I CONSIDER?

HOW CAN I
EVALUATE MY
PREDICTION?

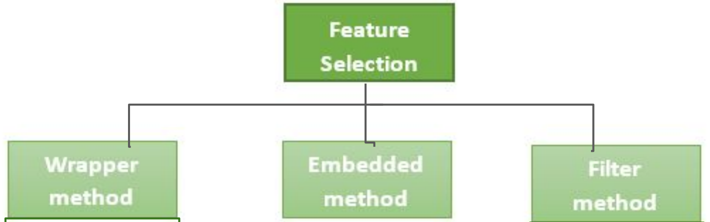
IMT SCUOLA ALTI STUDI LUCCA KU LEUVEN

Feature Selection

All differences are equal except we have some prior knowledge.



	A	B	C		
1	1	0	0	$F = F \wedge W$	first
2	1	1	0	$W = F \vee W$	white
3	0	0	0	$0 = F \wedge \neg W$	first and non-white
4	0	0	1	$\neg W = \neg F \wedge \neg W$	non-white
5	0	1	0	$\neg F \wedge W = F \oplus W$	non-first and white
6	0	1	1	$\neg F = \neg F \vee \neg W$	non-first
7	1	0	1	$F \vee \neg W = \neg F \oplus W$	first or non-white
8	1	1	1	$1 = \neg F \vee W$	non-first or white



Exhaustive search, SFSS, SBFS, Simulated annealing, Genetic algorithms

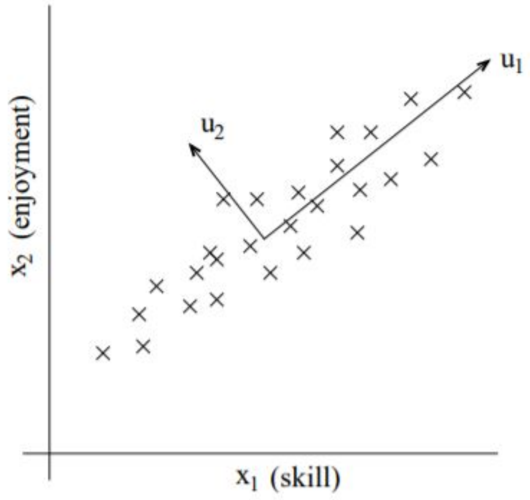
Information theoretic approach

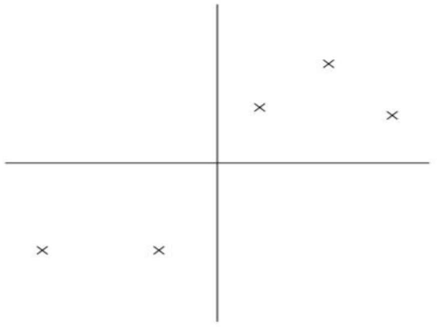
FEATURE REDUCTION:

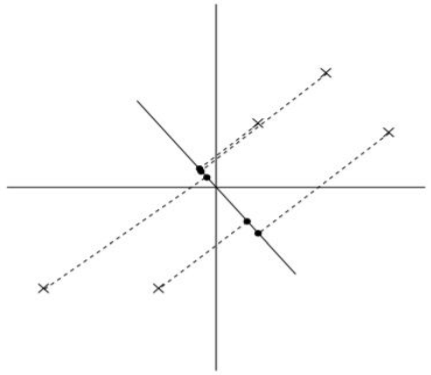
1. PCA (or Karhunen-Loeve Transform).
2. Fisher Discriminant Analysis.
3. Independent Component Analysis.

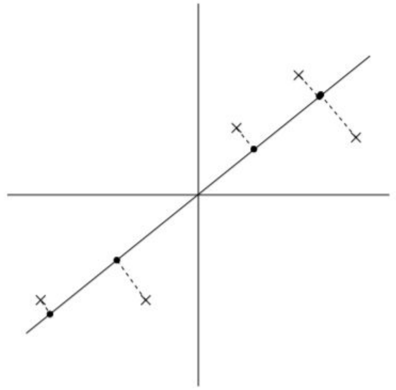
Principal Component Analysis

Reasoning:









Maximize:

$$\frac{1}{n}\sum_{i=1}^{n}({x^{(i)}}^T u)^2 = \frac{1}{n}\sum_{i=1}^{n} u^T x^{(i)} {x^{(i)}}^T u$$

$$= u^T \left(\frac{1}{n}\sum_{i=1}^{n} x^{(i)} {x^{(i)}}^T \right) u.$$

Subject to: $\|u\|_2 = 1$

This gives the principal eigen vector which is an empirical covariance matrix of the data

$$\Sigma = \frac{1}{n}\sum_{i=1}^{n} x^{(i)} {x^{(i)}}^T$$

Normalization:

$$x_j^{(i)} \leftarrow \frac{x_j^{(i)} - \mu_j}{\sigma_j}$$

Independent Component Analysis

Formulation: The acoustic reading observed is given as: **x = As,** where **s** is the source and **A** is the mixing matrix.

The goal is to recover s = Wx, where W is the unmixing matrix

ICA Ambiguities:

1. Cannot distinguish between permutations of mixing matrix.
2. Cannot determine scaling of mixing matrix.

Suppose the sources are Gaussian

Let **s** be distributed **N(0,I).** The distribution of **x is N(0, AA^T).** Let **R** be arbitrary orthogonal rotational matrix such that: **RR^T = I and A^{-1} = AR.** We have that **X^{-1}** is distributed as **N(0, AA^T).**

$$\mathbb{E}_{s\sim\mathcal{N}(0,I)}[x'(x')^T] = \mathbb{E}[A'ss^T(A')^T] = \mathbb{E}[ARss^T(AR)^T]$$
$$= ARR^TA^T = AA^T.$$

So long as the data is not Gaussian, it is possible given enough data to recover the independent sources.

Independent Component Analysis

The joint distribution of the sources given as:

$$p(s) = \prod_{j=1}^{d} p_s(s_j).$$

The density on the observed data for a source is

$$p_x(x) = p_s(Wx) \cdot |W|,$$

For all sources, we have:

$$p(x) = \prod_{j=1}^{d} p_s(w_j^T x) \cdot |W|.$$

Specify the cdf of the source as sigmoid function which is non-Gaussian i.e.

$g(s) = 1/(1+e^{-s})$ Hence, $p_s(s) = g'(s)$.

Given a training set: $\{x^{(i)}; i=1,..,n\}$. The ICA algorithm maximizes the log likelihood below in terms of **W.**

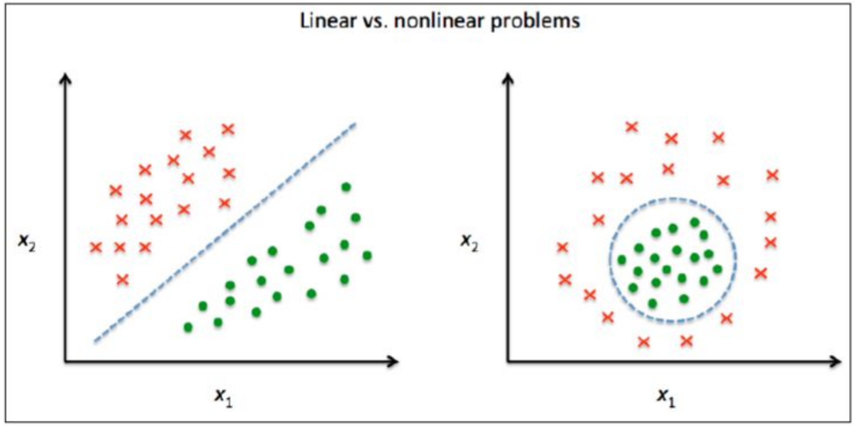
$$\ell(W) = \sum_{i=1}^{n} \left(\sum_{j=1}^{d} \log g'(w_j^T x^{(i)}) + \log |W| \right).$$

The weight is updated until the algorithm converges:

$$W := W + \alpha \left(\begin{bmatrix} 1 - 2g(w_1^T x^{(i)}) \\ 1 - 2g(w_2^T x^{(i)}) \\ \vdots \\ 1 - 2g(w_d^T x^{(i)}) \end{bmatrix} x^{(i)T} + (W^T)^{-1} \right),$$

$$s^{(i)} = Wx^{(i)}$$

Kernel PCA



Recall the covariance matrix from PCA:

$$\Sigma = \frac{1}{n}\sum_{i=1}^{n} x^{(i)} x^{(i)T} \Rightarrow \Sigma = \frac{1}{n}\sum_{i=1}^{n} \phi\left(x^{(i)}\right)\phi\left(x^{(i)}\right)^T$$

$$\Sigma v = \lambda v$$

$$\Rightarrow \frac{1}{n}\sum_{i=1}^{n} \phi\left(x^{(i)}\right)\phi\left(x^{(i)}\right)^T v = \lambda v$$

$$\Rightarrow v = \frac{1}{n\lambda}\sum_{i=1}^{n} \phi\left(x^{(i)}\right)\phi\left(x^{(i)}\right)^T v = \frac{1}{n}\sum_{i=1}^{n} a^{(i)}\phi\left(x^{(i)}\right)$$

In matrix notation form, we can show that:

$$\frac{1}{n}\phi(X)\phi(X)^T a = \lambda a \Rightarrow \frac{1}{n} K a = \lambda a$$

Where **K** is the (similarity) kernel matrix.

IMT SCUOLA ALTI STUDI LUCCA KU LEUVEN

Kernel PCA

What is obtained after the Kernel PCA is sample already projected to the component. We do not construct a transformation matrix as the case of standard PCA

Some Kernel functions include:

1. RBF or Gaussian kernel
2. Polynomial kernel
3. Hyperbolic tangent kernel etc.

Summary of Implementing Kernel PCA

1. Compute Kernel similarity **K**

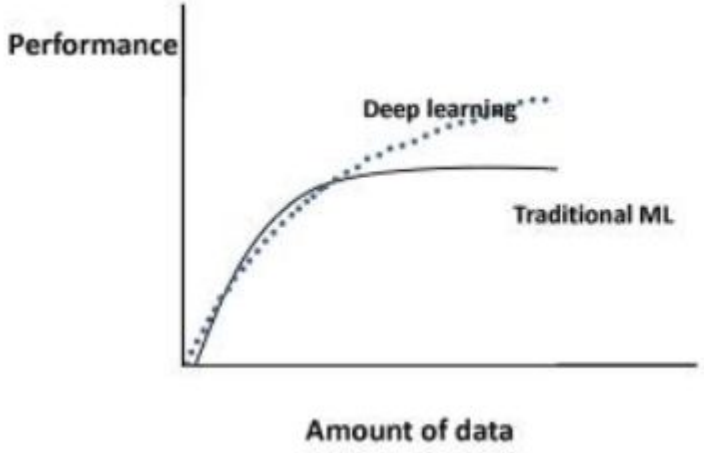
$$k\left(\boldsymbol{x}^{(i)}, \boldsymbol{x}^{(j)}\right) = \exp\left(-\gamma \left\|\boldsymbol{x}^{(i)} - \boldsymbol{x}^{(j)}\right\|^2\right)$$

2. Centralize the kernel matrix

$$\boldsymbol{K}' = \boldsymbol{K} - 1_n \boldsymbol{K} - \boldsymbol{K} 1_n + 1_n \boldsymbol{K} 1_n$$

3. Select the top k eigenvectors of the centered kernel matrix based on their corresponding eigenvalues, which are ranked by decreasing magnitude.

Clustering Algorithm



Top down (partitional) — Example: AGHC

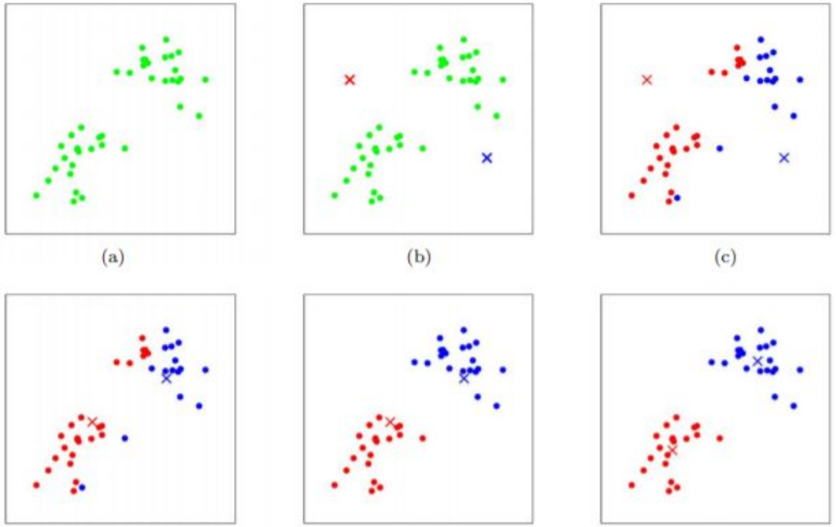
Bottom up (Hierarchical) — Example: K-means, Fuzzy K-means, Kohonen maps

K-Means

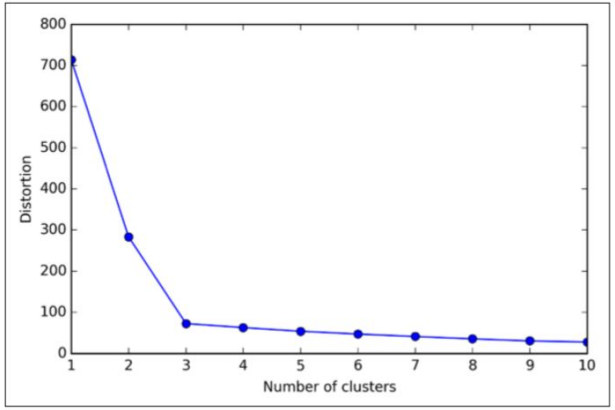
The k-means clustering algorithm is as follows:

1. Initialize **cluster centroids** $\mu_1, \mu_2, \ldots, \mu_k \in \mathbb{R}^n$ randomly.
2. Repeat until convergence: {

    For every i, set
$$c^{(i)} := \arg\min_j ||x^{(i)} - \mu_j||^2.$$
    For each j, set
$$\mu_j := \frac{\sum_{i=1}^m 1\{c^{(i)} = j\}x^{(i)}}{\sum_{i=1}^m 1\{c^{(i)} = j\}}.$$
}



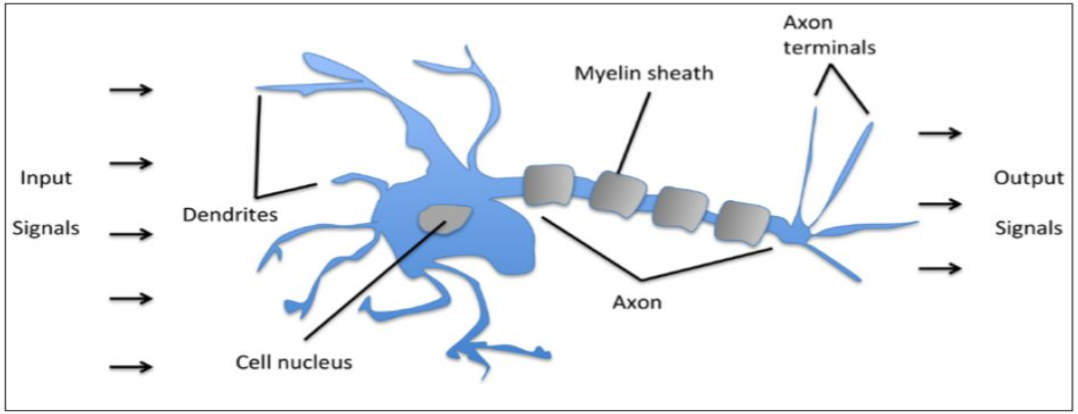
DRAWBACK: Number of clusters specified apriori. So what is the optimal number to specify?

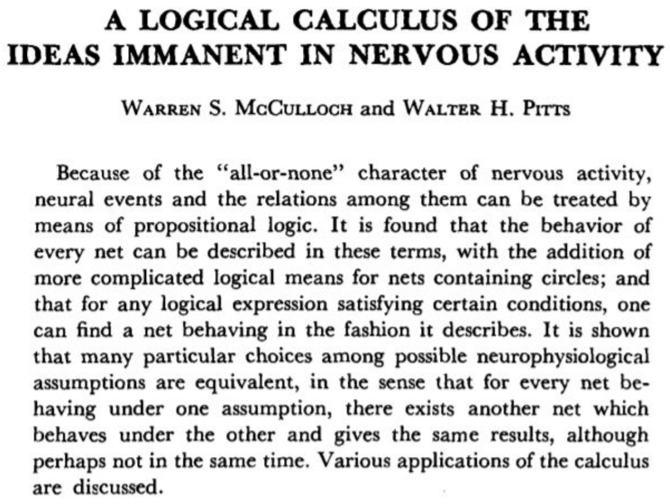


Quantifying optimality and quality of clusters

(a) Elbow method
(b) Silhouette analysis

Supervise Learning - Single Layer Network

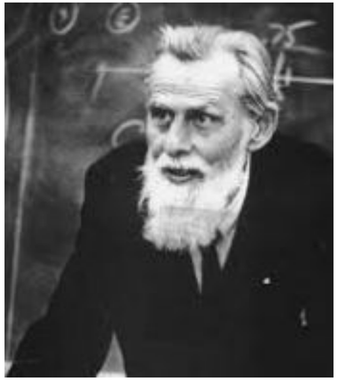


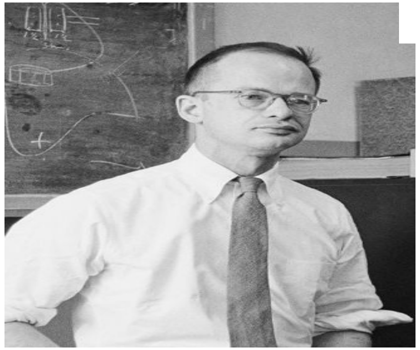


A LOGICAL CALCULUS OF THE IDEAS IMMANENT IN NERVOUS ACTIVITY

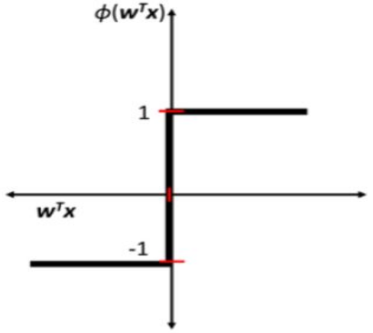
WARREN S. McCULLOCH and WALTER H. PITTS

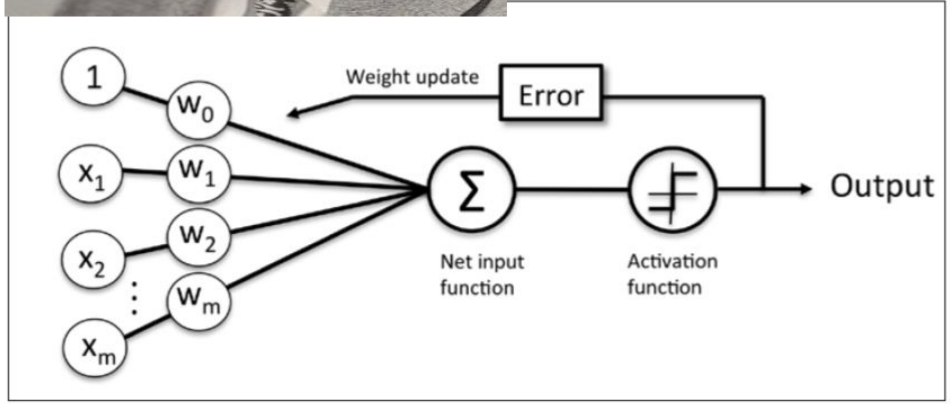
Because of the "all-or-none" character of nervous activity, neural events and the relations among them can be treated by means of propositional logic. It is found that the behavior of every net can be described in these terms, with the addition of more complicated logical means for nets containing circles; and that for any logical expression satisfying certain conditions, one can find a net behaving in the fashion it describes. It is shown that many particular choices among possible neurophysiological assumptions are equivalent, in the sense that for every net behaving under one assumption, there exists another net which behaves under the other and gives the same results, although perhaps not in the same time. Various applications of the calculus are discussed.





Perceptron





$$z = w_0 x_0 + w_1 x_1 + \ldots + w_m x_m = \mathbf{w}^T \mathbf{x}$$

$$\phi(z) = \begin{cases} 1 & \text{if } z \geq \theta \\ -1 & \text{otherwise} \end{cases}$$

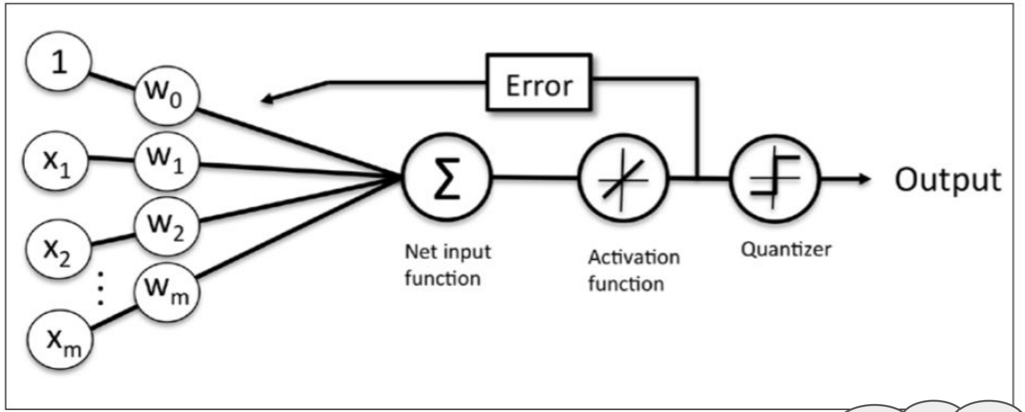
Perceptron rule:

$$\Delta w_j = \eta \left(y^{(i)} - \hat{y}^{(i)} \right) x_j^{(i)}$$

$$w_j := w_j + \Delta w_j.$$

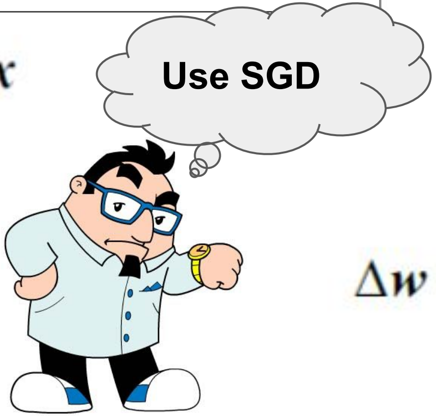
The discontinuous loss function is difficult to minimize because its gradient is zero almost everywhere

Adaptive Linear Neurons (Adaline)



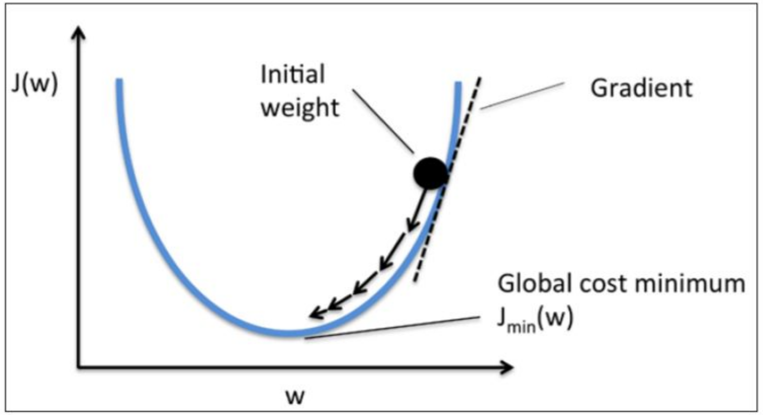
$$z = w_0 x_0 + w_1 x_1 + \ldots + w_m x_m = \boldsymbol{w}^T \boldsymbol{x}$$

$$\phi\left(\boldsymbol{w}^T \boldsymbol{x}\right) = \boldsymbol{w}^T \boldsymbol{x}$$



Batch Gradient descent

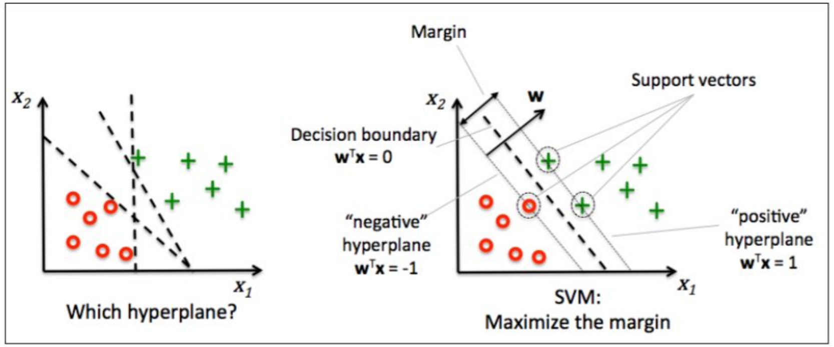
$$J(\boldsymbol{w}) = \frac{1}{2}\sum_i \left(y^{(i)} - \phi\left(z^{(i)}\right)\right)^2$$



$$\Delta \boldsymbol{w} = -\eta \Delta \mathrm{J}(\boldsymbol{w}) \qquad \boldsymbol{w} := \boldsymbol{w} + \Delta \boldsymbol{w}$$

IMT SCUOLA ALTI STUDI LUCCA KU LEUVEN

Support Vector Machine



$w_0 + \boldsymbol{w}^T \boldsymbol{x}_{pos} = 1$ and $w_0 + \boldsymbol{w}^T \boldsymbol{x}_{neg} = -1 \Rightarrow \boldsymbol{w}^T \left(\boldsymbol{x}_{pos} - \boldsymbol{x}_{neg} \right) = 2$

$$\frac{\boldsymbol{w}^T \left(\boldsymbol{x}_{pos} - \boldsymbol{x}_{neg} \right)}{\|\boldsymbol{w}\|} = \frac{2}{\|\boldsymbol{w}\|}$$

Constructing an hyperplane to maximize the margin implies:

minimize $\frac{1}{2}\|\boldsymbol{w}\|^2,$

Subject to $y^{(i)}\left(w_0 + \boldsymbol{w}^T \boldsymbol{x}^{(i)} \right) \geq 1 \ \forall_i$

Steps to Solving SVM Optimization problem:

1. Lagrange formulation of the problem:

$$L(x,a) = f(x) + \sum_i a_i g_i(x)$$

2. One relation that must be satisfied after taking the partial derivative of the Lagrange:

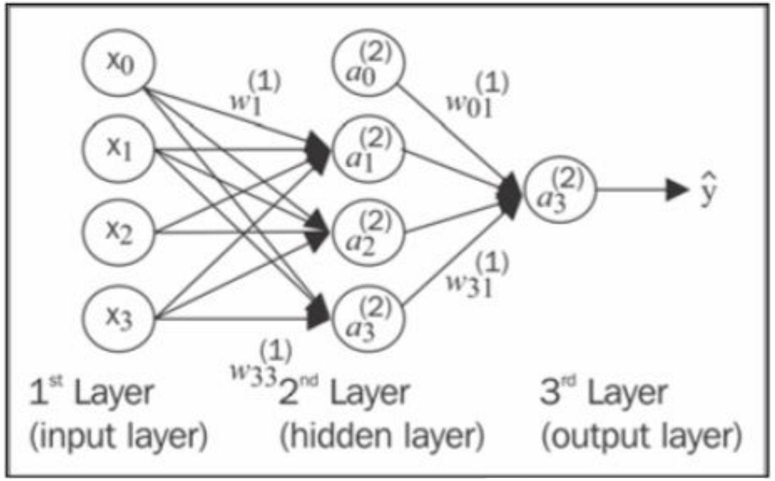
$$\mathbf{w} = \sum_{i=1}^{l} a_i y_i \mathbf{x}_i$$

3. Next substitute **w into** the original objective function to construct a **dual problem** that will be maximized. This gives an expression in terms of **a,** we can take the partial derivative of the new objective function wrt **a. To obtain the value of a.**

Check the full proof here

IMT SCUOLA ALTI STUDI LUCCA KU LEUVEN

Feedforward Neural Networks



$$z^{[\ell]} = W^{[\ell]}a^{[\ell-1]} + b^{[\ell]} \qquad a^{[\ell]} = g^{[\ell]}(z^{[\ell]})$$

For *l=1,....N.* Note that the activation function $g^{[N]}$ differ based on the task.

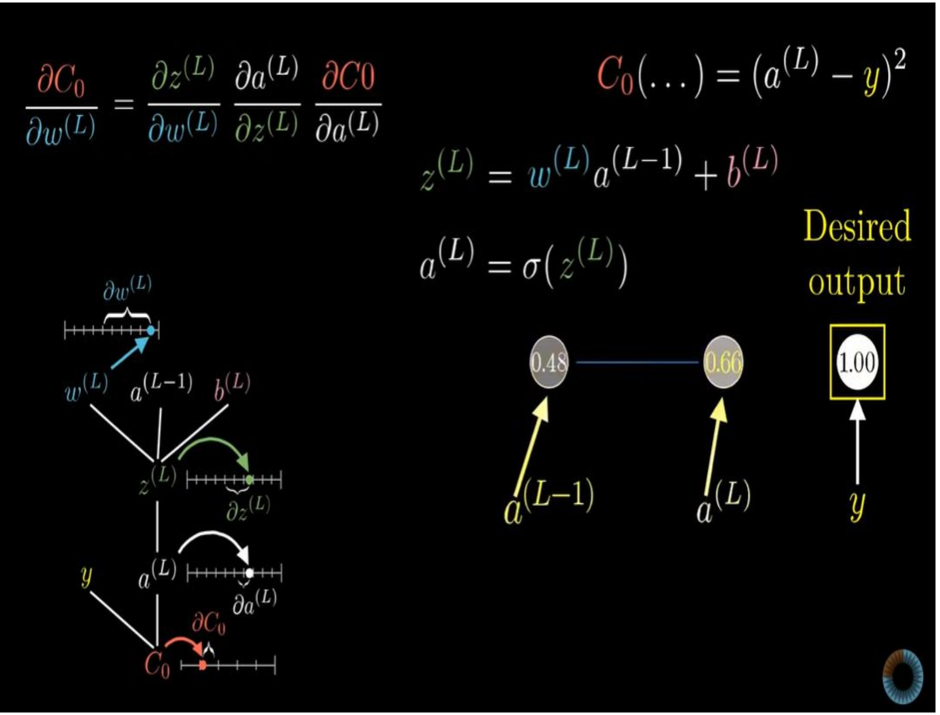
1.Starting at the input layer, we forward propagate the patterns of the training data through the network to generate an output.

2.Based on the network's output, we calculate the error that we want to minimize using a cost function.

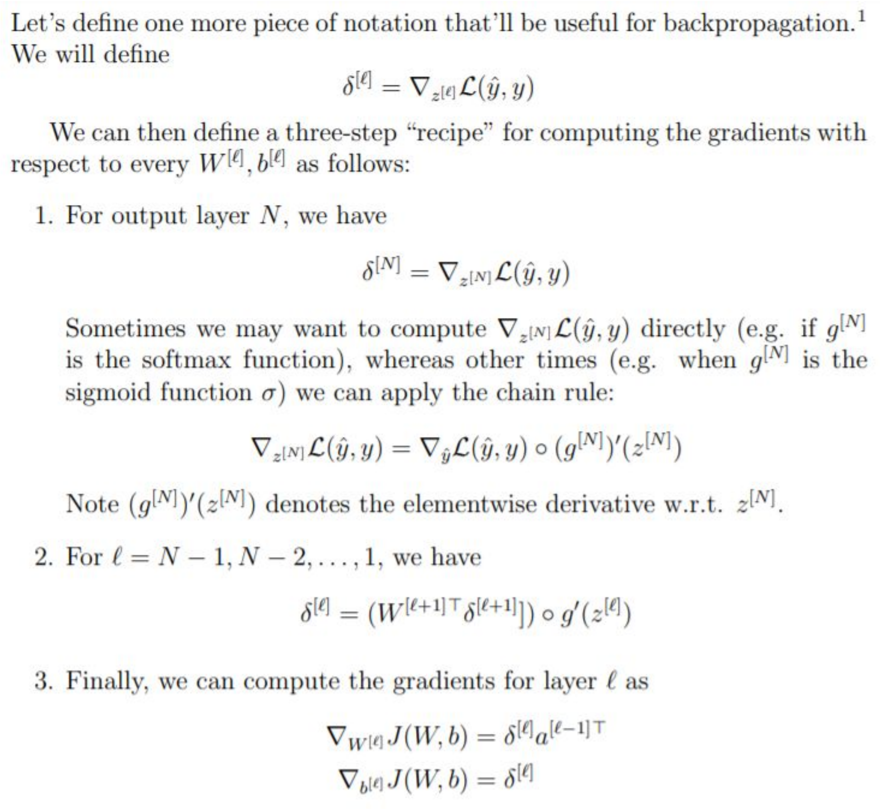
3.We backpropagate the error, find its derivative with respect to each weight in the network, and update the model.

This is because in the output layer we may be doing **regression** [hence we might use g(x) = x] or **binary classification** [g(x) = sigmoid(x)] or **multiclass classification** [g(x) = softmax(x)]

Backpropagation Algorithm



$\frac{\partial C_0}{\partial w^{(L)}} = \frac{\partial z^{(L)}}{\partial w^{(L)}} \frac{\partial a^{(L)}}{\partial z^{(L)}} \frac{\partial C0}{\partial a^{(L)}}$
$C_0(\dots) = (a^{(L)} - y)^2$
$z^{(L)} = w^{(L)} a^{(L-1)} + b^{(L)}$
$a^{(L)} = \sigma(z^{(L)})$
Desired output
$\partial w^{(L)}$
$w^{(L)}$ $a^{(L-1)}$ $b^{(L)}$
$z^{(L)}$
$\partial z^{(L)}$
y $a^{(L)}$
$\partial a^{(L)}$
∂C_0
C_0
0.48
0.66
1.00
$a^{(L-1)}$
$a^{(L)}$
y



Let's define one more piece of notation that'll be useful for backpropagation.[1] We will define

$$\delta^{[\ell]} = \nabla_{z^{[\ell]}} \mathcal{L}(\hat{y}, y)$$

We can then define a three-step "recipe" for computing the gradients with respect to every $W^{[\ell]}, b^{[\ell]}$ as follows:

1. For output layer N, we have

$$\delta^{[N]} = \nabla_{z^{[N]}} \mathcal{L}(\hat{y}, y)$$

   Sometimes we may want to compute $\nabla_{z^{[N]}} \mathcal{L}(\hat{y}, y)$ directly (e.g. if $g^{[N]}$ is the softmax function), whereas other times (e.g. when $g^{[N]}$ is the sigmoid function σ) we can apply the chain rule:

$$\nabla_{z^{[N]}} \mathcal{L}(\hat{y}, y) = \nabla_{\hat{y}} \mathcal{L}(\hat{y}, y) \circ (g^{[N]})'(z^{[N]})$$

   Note $(g^{[N]})'(z^{[N]})$ denotes the elementwise derivative w.r.t. $z^{[N]}$.

2. For $\ell = N-1, N-2, \dots, 1$, we have

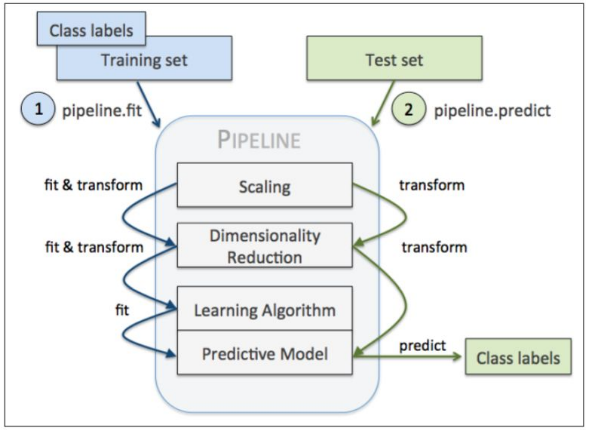
$$\delta^{[\ell]} = (W^{[\ell+1]\top} \delta^{[\ell+1]}) \circ g'(z^{[\ell]})$$

3. Finally, we can compute the gradients for layer ℓ as

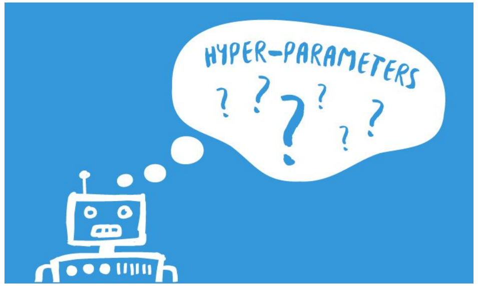
$$\nabla_{W^{[\ell]}} J(W, b) = \delta^{[\ell]} a^{[\ell-1]\top}$$
$$\nabla_{b^{[\ell]}} J(W, b) = \delta^{[\ell]}$$

Machine Learning as an art!

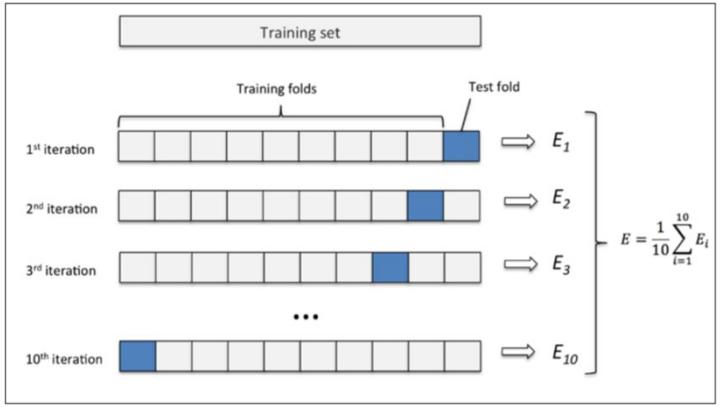
Combining transformers and estimators as pipeline



Grid search

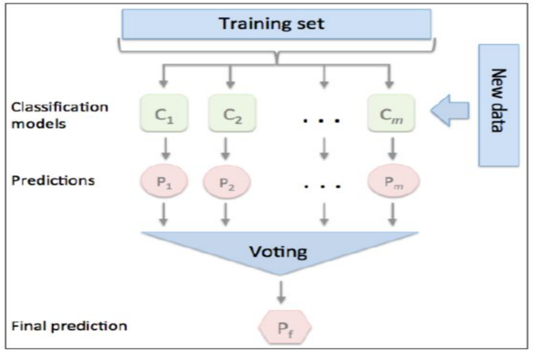


K-Fold cross validation



$$\hat{y} = mode\{C_1(x), C_2(x), \ldots, C_m(x)\}$$

Ensemble method



Instructions for Lab Session - Movie Review project

1. Download the folder for the lab session from this git repository: https://github.com/samueledet/datascience-nigeria. You can git clone or simply click the download button to have the folder on your local machine.

2. The goal of the project is to perform a sentiment analysis on movie reviews. S

3. Work in group of 4. Your task involves the following:
    i. Study the algorithms implemented for the movie review project.
    ii. Use a different algorithm for the estimator
    iii. The project was designed to be deployed on web. Can you improve the CSS and HTML code?
    iv. Can you try working on deploying this project?
    v. Think about how you can build a start-up around this. For example a review aggregation website for films and television like the American based company ROTTEN TOMATOES

References

[1] CS229 Machine Learning Course, Stanford University, https://see.stanford.edu/Course/CS229

[2] O. Bousquet et al.: Advanced Lectures on Machine Learning, Springer-Verlag, Berlin Heidelberg, 2004.

[3] Sebastian Raschka: Python Machine Learning, Packt Publishing, 2015.

[4] The Turing test: Stanford Encyclopedia of Philosophy