Machine Learning Group

Ovidiu Calin, September 29, 2017

A Few Introductory Problems

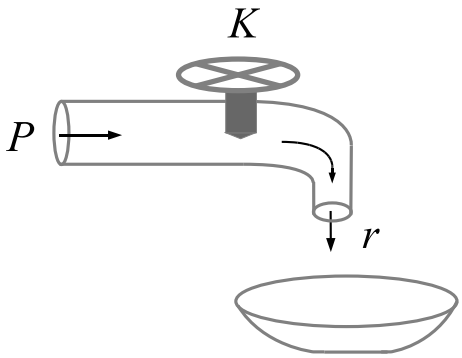
• This section introduces a few daily life problems which lead towards the concept of abstract neuron.

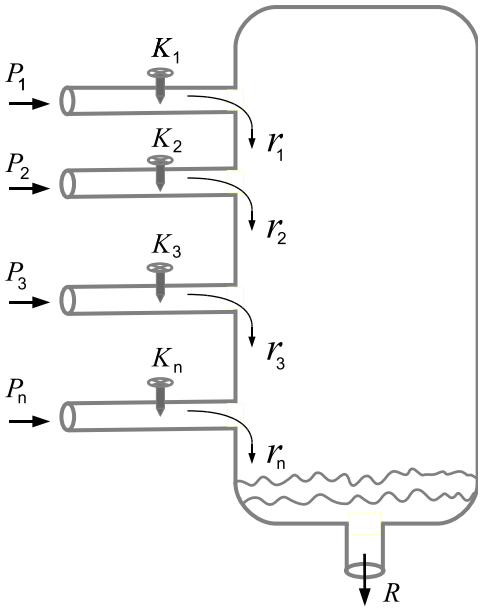
• They are all based on the process of adjusting a rate, a flow or a current that feeds a given unit (tank, cell, fund, transistor, etc.), which triggers a certain activation function.

• At the end of the section we shall provide some conclusions, which will pave the path to the abstract neuron definition.

Water in a sink

A pipe is supplied with water at a given pressure P. The knob K can adjust the water pressure, providing a variable outgoing water flow at rate r. The knob influences the rate by introducing a non-negative control w such that $P = r/w$.




K_1
P_1
r_1
K_2
P_2
r_2
K_3
P_3
r_3
K_n
P_n
r_n
R

A certain number of pipes of this type are used to pour water into a tank. The tank is drained at a rate R.

Given the water pressure supplies $P_1, \cdots, P_n$, how can one adjust the knobs $K_1, \cdots, K_n$ such that after an apriori fixed interval of time t there is a predetermined volume V of water in the tank?

The resulting water amount is

$$V = \begin{cases} 0, & \text{if } R > \sum_{i=1}^{n} r_i \\ (\sum_{i=1}^{n} r_i - R)t, & \text{otherwise.} \end{cases}$$

Let w_i denote the control provided by the knob K_i, so the ith pipe supplies water at a rate $r_i = P_i w_i$.

Consider the *activation function*

$$\varphi_t(x) = \begin{cases} 0, & \text{if } x < 0 \\ xt, & \text{otherwise.} \end{cases}$$

The volume of water, V, can be written in terms of the pressures P_i, controls w_i and function φ_t as

$$V = \varphi_t\Big(\sum_{i=1}^{n} P_i w_i - R\Big). \tag{0.1}$$

The rate R is called *bias*.

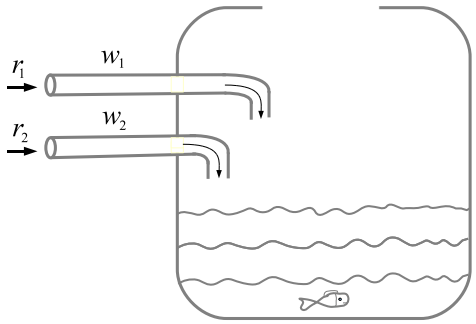
The problem is: find a solution of equation (0.1). In practice it would suffice to obtain an approximation of the solution by choosing the controls w_i and the bias R such that the following proximity function

$$\frac{1}{2}\left(\varphi_t\Big(\sum_{i=1}^{n} P_i w_i - R\Big) - V\right)^2$$

is minimized.

Fish in a tank

Salty water is poured through two pipes in an empty tank at rates r_1 and r_2, having concentrations w_1 and w_2, respectively. The tank is prepared for transporting ocean fish and the proper concentration of salt is required to keep the fish alive.



Given the ocean salt concentration, θ, and the water inflow rates, r_1, r_2, we need to find the concentrations of salt w_1 and w_2 in each of the pipes such that after a fixed time interval, t, the salty water reaches an acceptable salinity.

By *acceptable* we mean that fish have, let's say, a 95% chance of survival during transportation. In this problem the input is the pair of concentrations (w_1, w_2) and the output is a chance, modeled by a number between 0 and 1.

Let t be fixed. The amount of salt cumulated in the tank during this time is $A(t) = (r_1 w_1 + r_2 w_2)t$.
The amount of water poured in the tank is $V(t) = (r_1 + r_2)t$.
The salt concentration at time t is

$$c(t) = \frac{A(t)}{V(t)} = \frac{r_1 w_1 + r_2 w_2}{r_1 + r_2}$$

$$c = \widetilde{r}_1 w_1 + \widetilde{r}_2 w_2,$$

where $\widetilde{r}_i = \dfrac{r_i}{r_1 + r_2}$ is the percentage rate for each pipe.

The inflow rates r_i are fixed and the concentrations w_i are adjustable. If the final concentration c is very close to the critical concentration θ, then the chances of survival are maximal. If c is too low, $c << \theta$, or too large, $c >> \theta$, then the survival chance is small.

It makes sense to model the survival chance using the activation function

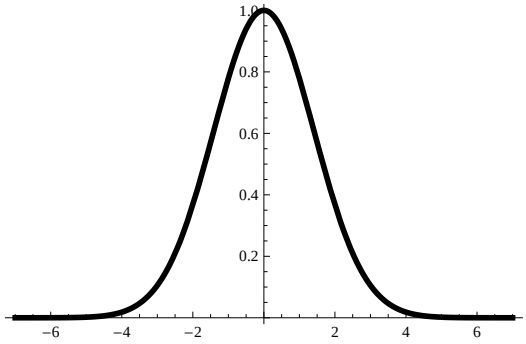
$$\varphi_\theta(x) = e^{\frac{-(x-\theta)^2}{2s^2}},$$

where the parameter $s > 0$ describes the spread of the bump.

The chance of survival is described by

$$\varphi_\theta(c) = \varphi_\theta(\widetilde{r}_1 w_1 + \widetilde{r}_2 w_2) = \varphi_0(\widetilde{r}_1 w_1 + \widetilde{r}_2 w_2 - \theta),$$

where $\varphi_0(x)$ is the Gaussian function centered at zero.



Ideally, the 100% chance of survival is reached for $c = \theta$, since

$$\varphi_\theta(c) = \varphi_\theta(\theta) = 1.$$

In practice we would be happy to adjust w_i just to make c to get in a proximity of θ. This fact is achieved by minimizing, for instance, the following distance function

$$d(w_1, w_2) = \frac{1}{2}\Big(\varphi_0(\widetilde{r}_1 w_1 + \widetilde{r}_2 w_2 - \theta) - 1\Big)^2.$$

Bacteria in a pool

Water is poured through two pipes in an empty pool at rates r_1 and r_2, having chlorine concentrations w_1 and w_2. The goal is to obtain a concentration of chlorine high enough such that survival bacteria are no more harmful for humans. Assume that this occurs at the critical concentration θ.

The input is the pair of concentrations (w_1, w_2) and the output is a number between 0 and 1. The number 0 corresponds to the case when bacteria is not affected at all, while 1 corresponds to the scenario when all bacteria have died.

The chlorine concentration at time t is

$$c(t) = \frac{(r_1 w_1 + r_2 w_2)t}{(r_1 + r_2)t} = \widetilde{r}_1 w_1 + \widetilde{r}_2 w_2.$$

Possible assumption:

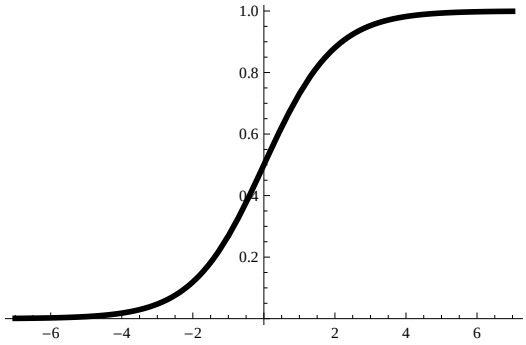
• if $c < \theta$ then the survival chance of bacteria is lower than 50%;

• if $c = \theta$ then the survival chance of bacteria is exactly 50%;

• if $c > \theta$ then the survival chance of bacteria lower than 50%.

It makes sense to choose the activation function

$$\varphi_\theta(x) = \frac{1}{1 + e^{-(x-\theta)}}.$$



The mortality of bacteria is modeled by

$$\varphi_\theta(c) = \varphi_\theta(\widetilde{r}_1 w_1 + \widetilde{r}_2 w_2) = \varphi_0(\widetilde{r}_1 w_1 + \widetilde{r}_2 w_2 - \theta),$$

where $\varphi_0(x)$ is the sigmoid function centered at zero.

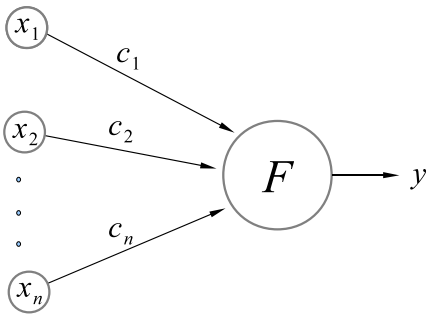
We have $\varphi_\theta(c) \in (0, 1)$ and the difference $1 - \varphi_\theta(c)$ represents the survival function of bacteria at concentration c.

If, for instance, a value of at most 5% for the survival function is considered "safe", then the concentrations w_i have to be adjusted such that $\varphi_\theta(c)$ has a value close to 0.95. This can be achieved by minimizing, for instance, the following distance function

$$\rho(w_1, w_2) = \frac{1}{2}\Big(\varphi_0(\widetilde{r}_1 w_1 + \widetilde{r}_2 w_2 - \theta) - 0.95\Big)^2.$$

Factory suppliers

A factory F has n suppliers that produce quantities $x_1, \cdots, x_n$ per day. The factory is connected with suppliers by a system of roads, which can be used at variable capacities $c_1, \cdots, c_n$. The factory F is supplied daily the amount $x = c_1x_1 + \cdots + c_nx_n$.

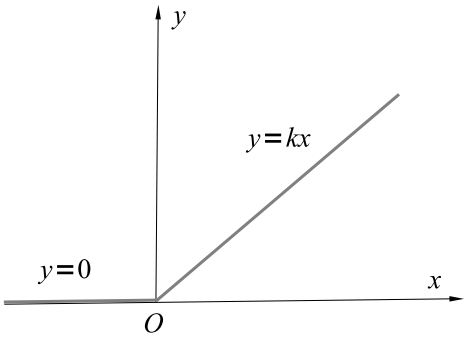


The factory production process starts when the supply reaches the critical daily level b. Denote the daily factory revenue by y.

- If $x < b$, then the factory does not produce anything, so $y = 0$;
- If $x \geq b$, then the revenue is $y = k(x - b)$, where k is a positive constant related to the cost of production.

Consider the following activation function:

$$\varphi(x) = \begin{cases} 0, & \text{if } x < 0 \\ kx, & \text{otherwise.} \end{cases}$$



Then the revenue can be modeled by

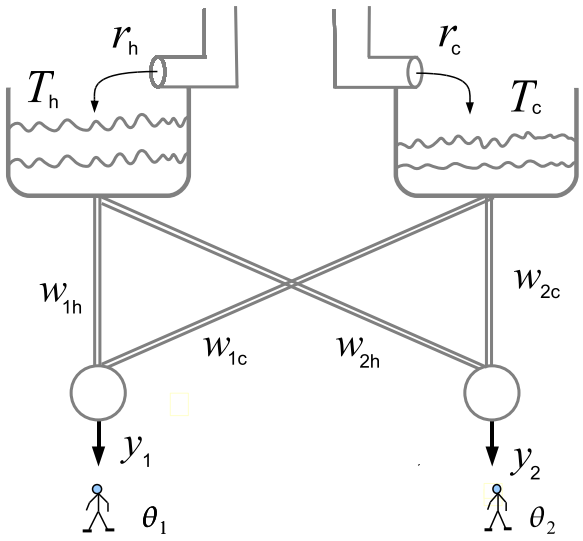
$$y = \varphi(x - b) = \varphi\Big(\sum_{i=1}^{n} c_i x_i - b\Big).$$

The emerging learning problem is the following: *Given the amounts x_i, what are the values of the road capacities c_i to meet or be very close to a given revenue value y?* The error function in this case can be considered to be $\frac{1}{2}(y - \varphi(x - b))^2$.

The two-shower problem

Two floor neighbors share the same tank of hot and cold water. If they take a shower in the same time, then it often happens that they are short of hot water. They need an algorithm of learning how to trade the hot water more efficiently.

The hot water tank is supplied by water of temperature T_h at a rate r_h. Similarly, the cold water tank is supplied by water at a rate r_c having temperature T_c. The two neighbors like to take showers at temperatures θ_1 and θ_2, respectively.


r_h
r_c
T_h
T_c
w_{1h}
w_{1c}
w_{2h}
w_{2c}
y_1
y_2
θ_1
θ_2

Rates constraints:

$$0 \leq w_{1c} + w_{2c} \leq r_c \tag{0.2}$$

$$0 \leq w_{1h} + w_{2h} \leq r_h. \tag{0.3}$$

This shower network has two outcomes: y_1 and y_2, the temperatures of shower water for each of the neighbors. These are

$$\begin{aligned} y_1 &= w_{1h}T_h + w_{1c}T_c \\ y_2 &= w_{2h}T_h + w_{2c}T_c. \end{aligned}$$

The rates w_{ih} and w_{ic} have to be adjusted such that the outgoing temperatures y_i are in the proximity of the preferred temperatures θ_i. This can be obtained considering the optimization of the function

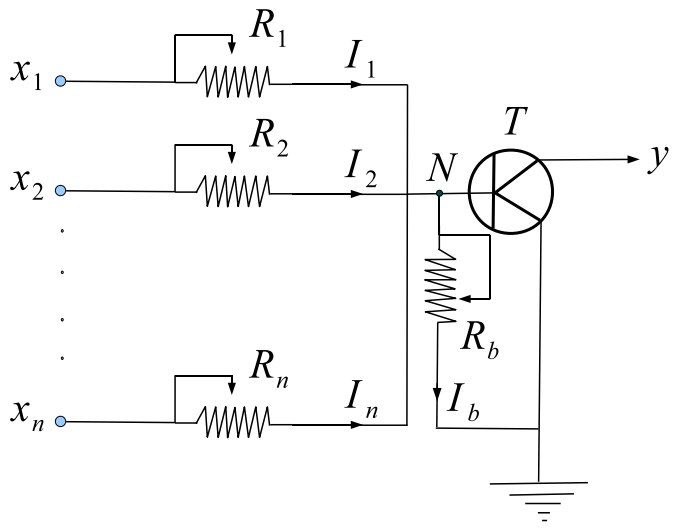
$$f(w_{1h}, w_{1c}, w_{2h}, w_{2c}) = \frac{1}{2}(y_1 - \theta_1)^2 + \frac{1}{2}(y_2 - \theta_2)^2$$

subject to the constraints (0.2)-(0.3).

Translation into electronics language:

Fluids	pressure	rate	knob
Electronics	voltage	intensity	resistivity

An electronic circuit



The neuron implemented as an electronic circuit

Voltages $x_1, \cdots, x_n$ applied at the input lines are transmitted through the variable resistors $R_1, \cdots, R_n$. The emerging currents I_k are given by the Ohm's law

$$I_k = \frac{x_k}{R_k}, \qquad k = 1, \cdots, n.$$

For simplicity reasons, define the weights $w_k = \dfrac{1}{R_k}$ as the inverse of resistances. Then the previous relation becomes $I_k = x_k w_k$.

The n currents converge at the node N. Another variable resistor, R_b, connects the node N with the base. The current through this wire is denoted by I_b. By the Kirchhoff second law, the sum of the incoming currents at the node N is equal to the sum of the outgoing currents at the same node. Therefore, the current I that enters the transistor T is given by

$$I = \sum_{k=1}^{n} I_k - I_b = \sum_{k=1}^{n} x_k w_k - I_b.$$

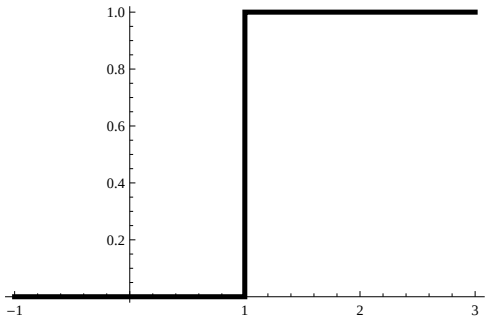
The transistor is an electronic component that acts as a switch: if the ingoing current I is lower than the transistor's threshold θ, then no current gets through. If the current I exceeds the threshold θ, the current gets through. The output current y of the transistor T can be modeled as

$$y = \begin{cases} 0, & \text{if } I < \theta \\ k, & \text{if } I \geq \theta, \end{cases}$$

for some constant $k > 0$.

The activation function is the step function

$$\varphi_\theta(x) = \begin{cases} 0, & \text{if } x < \theta \\ 1, & \text{if } x \geq \theta. \end{cases} \tag{0.4}$$



The output can be written as

$$\begin{aligned} y &= k\varphi_\theta(I) = k\varphi_0(I-\theta) = k\varphi_0\Big(\sum_{k=1}^{n} I_k - I_b - \theta\Big) \\ &= k\varphi_0\Big(\sum_{k=1}^{n} x_k w_k - \beta\Big), \end{aligned}$$

where $\beta = I_b + \theta$ is considered as the bias.

Note that $\varphi_0(x)$ is the Heaviside function centered at the origin.

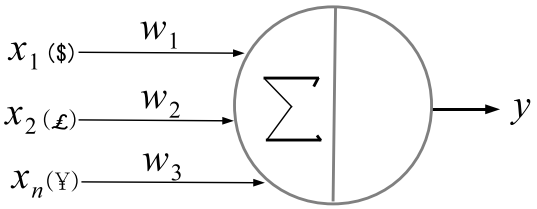
The learning problem is:

Given a current $z = z(x_1, \cdots, x_n)$ *depending on the input voltages* $x_1, \cdots, x_n$*, adjust the resistances* $R_1, \cdots, R_n$*,* R_b *such that the output* y *approximates* z *in the least squares way, i.e. such that* $\frac{1}{2}(z-y)^2$ *is minimum.*

Money in a fund

n financial institutions, each having a wealth x_i, deposit amounts of money in a fund, at some adjustable rates of deposit w_i. Then the money in the fund is given by

$$x = x_1 w_1 + \cdots + x_n w_n.$$



The fund functions as in the following:
As long as the fund has less than a certain reserve fund M, the fund manager does not invest. Only the money exceeding the reserve fund M is invested.

Let $k = e^{rt}$, where r and t denote the investment rate of return and time of investment, respectively. Then the output is given by

$$y = \begin{cases} 0, & \text{if } x \leq M \\ k(x - M), & \text{if } x > M. \end{cases}$$

Consider the activation function

$$\varphi(x) = \begin{cases} 0, & \text{if } x \leq 0 \\ kx, & \text{if } x > 0 \end{cases}$$

Then output becomes

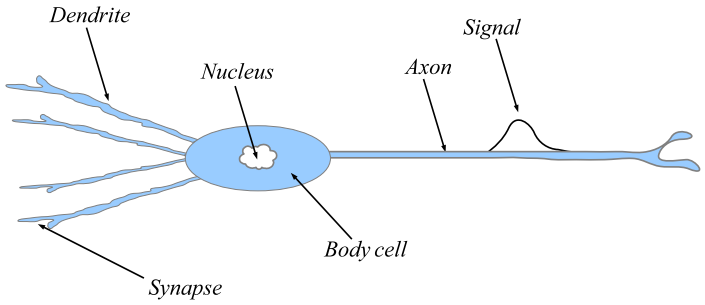
$$y = \varphi(x - M) = \varphi(x_1 w_1 + \cdots + x_n w_n - M).$$

The learning problem can be stated as:

Adjust the rates w_i such that an apriori planned profit z on the fund is obtained.

Biological neuron

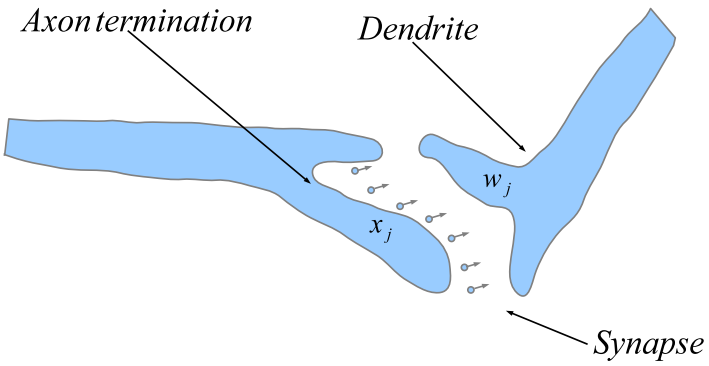
A neuron is a cell which consists in the following parts: *dendrites*, *axon* and *body-cell*.



• Dendrites are transmission channels that collect information from the axons of other neurons.

• The signal traveling through an axon reaches its terminal end and produces some chemicals x_i which are liberated in the synaptic gap.

• These chemicals are acting on the dendrites of the next neuron either in a strong or a weak way. The connection strength is described by the weights system w_i.

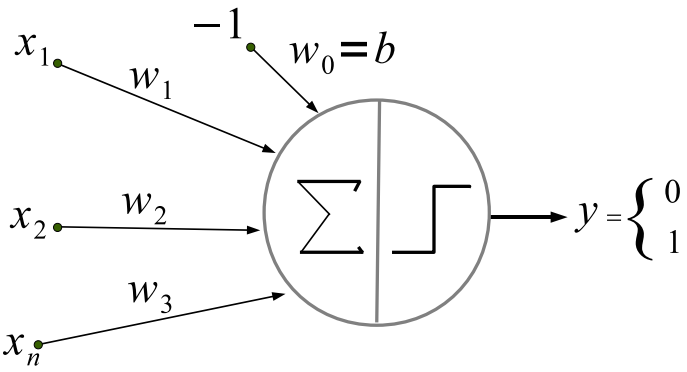


The *synapse* is the connection between the axon of one neuron and the dendrite of another.

• The body-cell collects all signals from dendrites. Here the dendrites activity adds up into a total potential and if a certain threshold is reached, the neuron fires a signal through the axon. The threshold depends on the sensitivity of the neuron and measures how easy is to get the neuron to fire.

• The axon is the channel for signal propagation. The signal consists in the movement of ions from the body-cell towards the end of the axon. The signal is transmitted electro-chemically to the dendrites of the next neuron.

Schematic model of the neuron



$y = \varphi_b\Big(\sum_{i=1}^{n} x_i w_i\Big) = \varphi_0\Big(\sum_{i=1}^{n} x_i w_i - b\Big)$,
where the activation function φ_b is the unit step function centered at b, and φ_0 is the Heaviside function.

Bias elimination

In order to get rid of the bias, a smart trick is used. The idea is to introduce an extra weight, $w_0 = b$, and an extra constant signal, $x_0 = -1$. Then the previous outcome can be written as

$$y = \varphi_0\Big(\sum_{i=0}^{n} x_i w_i\Big).$$

Learning process

The neuron learns an output function by updating the synaptic weights w_i. Only one neuron cannot learn complicated functions, but it can learn a simple function like, for instance, the piece-wise function

$$z(x_1, x_2) = \begin{cases} 0, & \text{if } x_2 \leq 0 \\ 1, & \text{if } x_2 > 0, \end{cases}$$

which takes the value 1 on the upper-half plane and 0 otherwise. This function can be learned by choosing the weights $w_1 = 0$, $w_2 = 1$ and bias $b = 0$. The neuron fires when the inequality $x_1 w_1 + x_2 w_2 > b$ is satisfied. This is equivalent to $x_2 > 0$, which is the equation of the upper-half plane.

One neuron can learn the upper-half plane. By this we mean that the neuron can distinguish whether a point (x_1, x_2) belongs to the upper-half plane. Similarly, one can show that a neuron can learn any half-plane.

However, the neuron cannot learn a more complicated function, such as

$$\zeta(x_1, x_2) = \begin{cases} 0, & \text{if } x_1^2 + x_2^2 \geq 1 \\ 1, & \text{if } x_1^2 + x_2^2 < 1, \end{cases}$$

i.e. it cannot learn to decide whether a point belongs to the interior of the unit circle.

Conclusions

• All examples start with some incoming information provided by the input variables $x_1, \cdots, x_n$. They can be deterministic variables, i.e. one can measure clearly their values, and these do not change from one observation to another. The analysis also makes sense in the case when the input variables x_i are random, i.e. they have different values for distinct observations, and these values are distributed according to a certain density law.

• Another common feature is that the inputs x_i are multiplied by a weight w_i and then the products are added up to form the sum $\sum_{i=1}^{n} x_i w_i$, which is the inner product between the input vector $\mathbf{x}$ and weight vector $\mathbf{w}$. We note that in most examples a bias b is subtracted from the aforementioned inner product. If a new weight $w_0 = b$ and a new input $x_0 = -1$ are introduced, this leads to $\sum_{i=1}^{n} x_i w_i - b = \sum_{i=0}^{n} x_i w_i$, which is still the inner product of two vectors that extend the previous input and weights vectors.

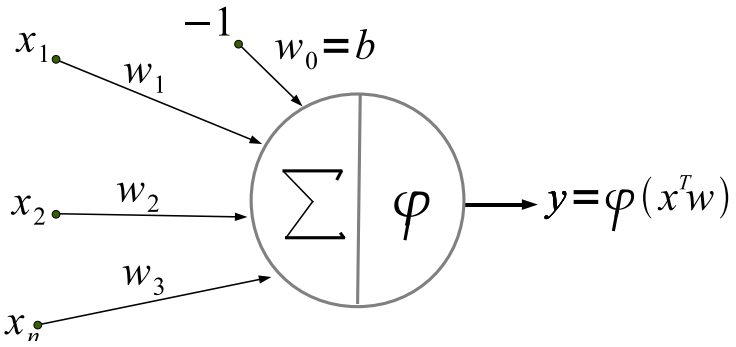
• Another common ingredient is the activation function, denoted by $\varphi(x)$. We have seen in the previous sections several examples of activation functions: unit step functions, sigmoid, Gaussian, and piecewise-linear; each of them describe a specific mode of action of the neuronal unit and suits a certain type of problem.

• The output function y is obtained applying the activation function φ on the previous inner product, leading to the expression $y = \varphi\Big(\sum_{i=0}^{n} x_i w_i\Big)$. If the input variables x_i are deterministic, then the output y is also deterministic. However, if the inputs are random variables, the output is also a random variable.

• The goal of all the presented examples is to adjust the system of weights $\{w_i\}_{i=0,n}$ such that the outcome y approximates a certain given function as good as possible. Even if this can be sometimes an exact match, in most cases it is just an approximation. This is achieved by choosing the weights w_i such that a certain proximity function is minimized. The proximity function is a distance-like function between the realized output and the desired output. For the sake of simplicity we had considered only Euclidean distances, but there are also other possibilities.

Abstract neurons

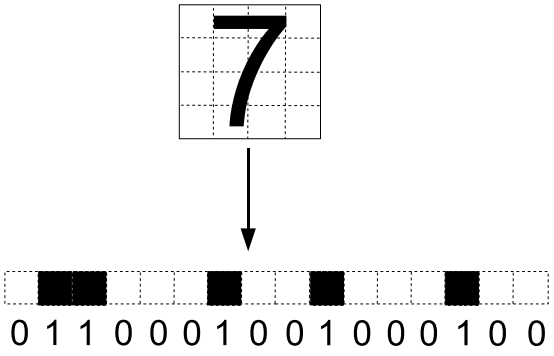
An **abstract neuron** is a quadruple $(\mathbf{x}, \mathbf{w}, \varphi, y)$, where $\mathbf{x}^T = (x_0, x_1, \cdots, x_n)$ is the input vector, $\mathbf{w}^T = (w_0, w_1, \cdots, w_n)$ is the weights vector, with $x_0 = -1$ and $w_0 = b$, the bias, and φ is an activation function that defines the outcome function $y = \varphi(x^T w) = \varphi(\sum_{i=0}^{n} w_i x_i)$.


x_1
-1
$w_0 = b$
w_1
x_2
w_2
w_3
x_n
$\sum$
φ
$y = \varphi(x^T w)$

Examples of input data

- binary if $x_i \in \{0, 1\}$ for $i = 1, n$.
- signed if $x_i \in \{-1, 1\}$ for $i = 1, n$;
- digital if $x_i \in \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$ for $i = 1, n$;
- arbitrary real numbers if $x_i \in (-\infty, \infty)$ for $i = 1, n$;
- an interval of real numbers if $x_i \in [0, 1]$ for $i = 1, n$.

Transformation of a 4×4 pixels character into a sequence of 0s and 1s.



Input efficiency

Which type of input is more efficient? Equivalently, how many switching states do optimize the transmitted information from a certain implementation cost point of view?

Let β be the number of input signal states (for instance $\beta = 2$ for the binary signals). Let n denote the number of channel inputs. The implementation cost is $F = cn\beta$, where $c > 0$ is a proportionality constant.

Using n channels with β states, one can represent β^n numbers. Assume the cost F fixed and optimize the number of transmitted numbers as a function of β. Consider the constant $k = \frac{F}{c}$. Then $n = \frac{F}{c\beta} = \frac{k}{\beta}$. Therefore $f(\beta) = \beta^{\frac{k}{\beta}}$ numbers can be represented. This function reaches its maximum at the same point as its logarithm, $g(\beta) = \ln f(\beta) = \frac{k}{\beta} \ln \beta$. Since its derivative is $g'(\beta) = \frac{k}{\beta}(1 - \ln \beta)$, it follows that the maximum is achieved for $\beta = e \approx 2.718$. Taking the closest integer value, it follows that $\beta = 3$ is the optimal number of states for the input signals.

Examples of abstract neurons

A *perceptron* is a neuron with the inputs either zeros or ones, i.e., $x_i \in \{0, 1\}$, and with the activation function given by the Heaviside function

$$\varphi(x) = \begin{cases} 0, & \text{if } x < 0 \\ 1, & \text{if } x \geq 0. \end{cases}$$

The output of the perceptron is given as a *threshold gate*

$$y = \varphi(x^T w) = \begin{cases} 0, & \text{if } \sum_{i=1}^{n} w_i x_i < b \\ 1, & \text{if } \sum_{i=1}^{n} w_i x_i \geq b. \end{cases}$$

A *sigmoid neuron* is a computing unit with the inputs $x_1, \ldots, x_n \in [0, 1]$, and with the activation function given by the logistic function

$$\sigma(z) = \frac{1}{1 + e^{-z}}. \tag{0.5}$$

The output of the sigmoid neuron is a number in $(0, 1)$, which is given by

$$y = \sigma(x^T w - b) = \frac{1}{1 + e^{-w^T x + b}}.$$

Advantages:

(i) it approximates perceptrons;

(ii) the output function is continuous with respect to weights, i.e. small changes in the weights and threshold correspond to small changes in the output; this does not hold in the case of the perceptron, whose output has a jump discontinuity.

Adaline (ADAptive LInear NEuron) is a neuron with a linear activation function and random inputs. Its learning algorithm uses the Least Mean Squares. It was introduced and actually implemented as a physical device by Widrow and Hoff in about 1959. Adaline is used to recognize patterns, data filtering and to approximate linear functions.