1. Feedforward neural networks

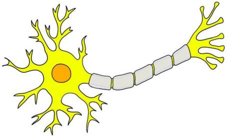
1.1. The perceptron

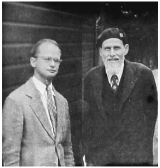
Manel Martínez-Ramón

Introduction

The concept of neuron

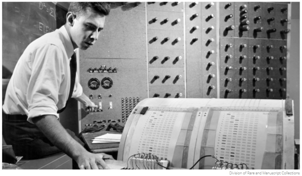
- Inspired in the structure of the nervous system (McCulloch and Pitts, 1943).
- Described as an element with two possible states (0, 1).

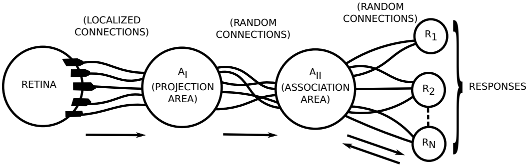




- If a linear combination of the inputs (dendrites) is above a threshold, the output (axon) will be activated.

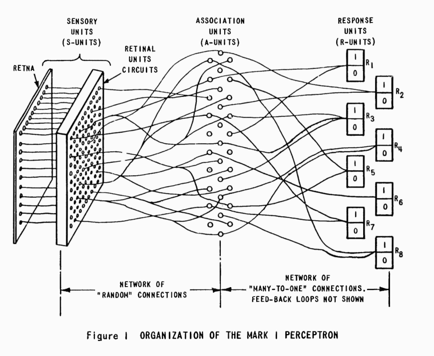
- The concept of artificial neural network was introduced by Rosemblatt (1958)





- First stage (retina): collects the observation.
- Second stage (projection): extracts the information.
- Third stage (association): maps the features into a set of responses.

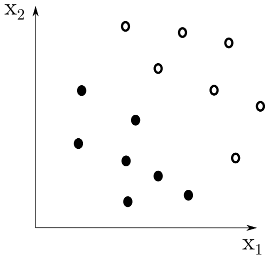
- Rosemblatt proved that his structure could learn from data.
- He developed the Mark 1 perceptron machine.
  - The Mark 1 was an electromechanical *learning machine*.
  - It had a camera of 400 pixels and attenuators driven by motors.
- The machine was able to classify *linearly separable* patterns.



Mark I Perceptron Operators' Manual, Cornell Aeronautical Laboratory (1960).

The perceptron

- A simplification of the perceptron is a binary classifier.
- Assume a set of observations in a column vector $\mathbf{x} = \{x_1 \cdots x_D\}^\top$, that can be arbitrarily labelled as "black" (-1) and "white" (+1) classes.



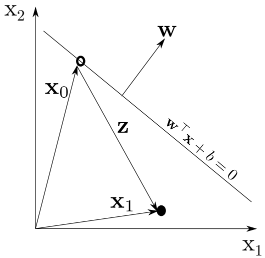
A classification function can be constructed from a *separating hyperplane* between both classes:

$$\mathbf{w}^\top \mathbf{x} + b = 0 \tag{1}$$

The classifier is defined as

$$f(\mathbf{x}) = \operatorname{sign}\left(\mathbf{w}^\top \mathbf{x} + b\right) \tag{2}$$

as we will prove next.



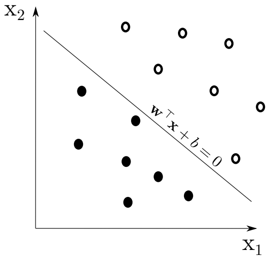
- Vector $\mathbf{w}$ is normal to the hyperplane defined by $\mathbf{w}\top\mathbf{x}+b=0$.
- Vector $\mathbf{x}_0$ is inside the plane, hence $\mathbf{w}^\top\mathbf{x}_0+b=0$
- Vector $\mathbf{x}_1$ can be written as $\mathbf{x}_1=\mathbf{x}_0+\mathbf{z}$
- Therefore, since $\angle\overline{\mathbf{x}_0\mathbf{z}}>90^o$:

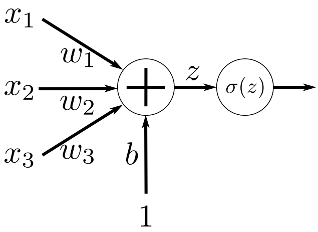
$$
\begin{aligned}
f(\mathbf{x}_1) &= \operatorname{sign}\left(\mathbf{w}^\top\mathbf{x}_1+b\right)=\operatorname{sign}\left(\mathbf{w}^\top(\mathbf{x}_0+\mathbf{z})+b\right)\\
&= \operatorname{sign}\left(\mathbf{w}^\top\mathbf{x}_0+b+\mathbf{w}^\top\mathbf{z}\right)=\operatorname{sign}\left(\mathbf{w}^\top\mathbf{z}\right)=-1
\end{aligned}
\tag{3}
$$

The binary classification function

Graphically, the classifier is a structure weights that represent the dot product and a bias, and a generic *activation function* σ:

$$f(\mathbf{x}) = \sigma\left(\mathbf{w}^\top \mathbf{x} + b\right) = \sigma\left(\sum_d w_d x_d + b\right)$$





The separating hyperplane in an example of 2 dimensions (left), and a representation of the classification function for the case of 3 dimensions.

- The perceptron rule is the first algorithm to train a learning machine, that was able to classify linearly separable patterns, and it was introduced by Rosenblatt in 1958.
- Let us start by defining the concept of error.

$$e_i = \frac{1}{2}\left(y_i - \text{sign}(\mathbf{w}^\top \mathbf{x}_i + b)\right) \tag{4}$$

  This is simply 1 if the sample has been misclassified, and zero otherwise.
- The training criterion is simply “If a training sample is misclassified, then slightly move the boundary so it is properly classified”.

Assume a sample $\mathbf{x}_k$ with label y_k and a classifier $f(\mathbf{x}) = \text{sign}(\mathbf{w}^\top \mathbf{x}_k + b) \neq y_k$ this is, the sample is misclassified.

- If the label is $y_k = 1$ and the classification is wrong, then

$$\mathbf{w}^{(k)^\top} \mathbf{x}_k + b^{(k)} < 0$$

- The training rule is

$$\begin{aligned} \mathbf{w}^{(k+1)} &= \mathbf{w}^{(k)} + y_k \mathbf{x}_k \\ b^{(k+1)} &= b^{(k)} + y_k \end{aligned} \tag{5}$$

and, after the update

$$\left(\mathbf{w}^{(k)} + \mathbf{x}_k\right)^\top \mathbf{x}_k + b + 1 = \mathbf{w}^{(k)^\top} \mathbf{x}_k + b^{(k)} + \|\mathbf{x}_k\|^2 + 1 > \mathbf{w}^{(k)^\top} \mathbf{x}_k + b^{(k)}$$

The perceptron rule

- If the label is $y_i = -1$ and the classification is wrong, then

$$\mathbf{w}^\top \mathbf{x}_i + b > 0$$

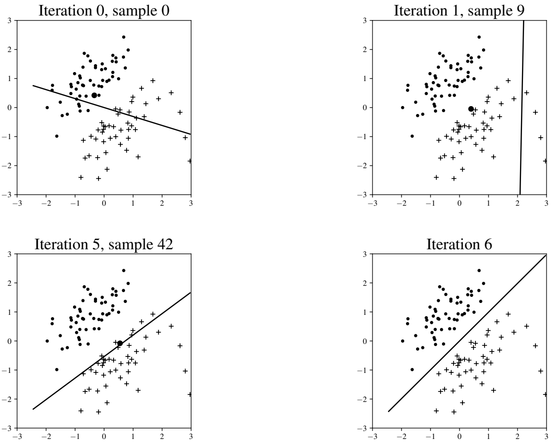
  and, after the update

$$(\mathbf{w} - \mathbf{x}_i)^\top \mathbf{x}_i + b - 1 < \mathbf{w}^\top \mathbf{x}_i + b$$

In both cases, the response of the classifier gets closer to the correct classification.

Theorem

If the training dataset is linearly separable, then the perceptron rule converges in a finite number of iterations (Novikoff, 1963).



Example of the application of the perceptron rule in a set of separable data in dimension 2.

The perceptron rule

- The perceptron can be generalized to multiclass classfication in a straightforward way to obtain the complete original algorithm.
- The algorithm presents two main limitations
    - The structure is purely linear.
    - The algorithm will not converge if the data is not linearly separable (i. e. if there are overlapping between classes.
- The second limitation can be overcome if the thresholded output (0, 1) is changed by a *soft* output that allows a more parsimonious update.
- Such an algorithm was not implemented in the Mark 1 because of the limitations of the electromechanical machine. For that purpose, an arithmetic unit is needed.

- In order to change the hard decision function $\text{sign}(\cdot)$ by a soft one, we can take the simplest choice, which is the linear one:

$$f(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + b$$

- The training criterion is then (MMSE)

$$\min_{\mathbf{w},b} \mathbb{E}\left[e_i^2\right] = \min \mathbb{E}\left[\left(y_i - \mathbf{w}^\top \mathbf{x}_i - b\right)^2\right] \tag{6}$$

Of course, the actual expectation cannot be computed, because the probability density functions of the random variables are not available, so the expectation will be approximated by a sample average.

- Assuming that N labelled samples $\mathbf{x}_i, y_i$ are available, we introduce here matrix $\mathbf{X} = [\mathbf{x}_1, \cdots, \mathbf{x}_N]$ and vector $\mathbf{y} = [y_1, \cdots, y_N]$ containing all training samples and labels.
- By computing the gradients of the error with respect to $\mathbf{w}$ abd b and nulling them we obtain

$$\begin{aligned} \mathbf{w} &= \left(\mathbf{X}\mathbf{X}^\top\right)^{-1} \mathbf{X}\left(\mathbf{y} - b\mathbf{1}\right) \\ b &= \frac{1}{N}\sum_{i=1}^{N}\left(y_i - \mathbf{w}^\top \mathbf{x}_i\right) \end{aligned} \tag{7}$$

where $\mathbf{1}$ is a column of N ones. (Derivation as an exercise).

The above solution is too cumbersome, but if we extend both the input data and the weight vector, the solution is more compact. The extension is

$$\mathbf{x} \rightarrow \begin{bmatrix} \mathbf{x} \\ 1 \end{bmatrix}, \quad \mathbf{w} \rightarrow \begin{bmatrix} \mathbf{w} \\ b \end{bmatrix}$$

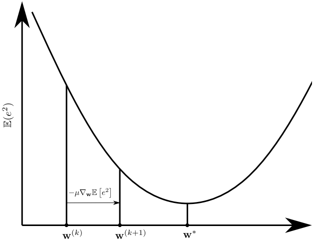
In this case, nulling the gradient gives

$$\mathbf{w} = \left(\mathbf{X}\mathbf{X}^\top\right)^{-1}\mathbf{X}\mathbf{y} \tag{8}$$

(derivation as an exercise) which is a compact solution, i.e. no iterations needed.

The Least Mean Square solution

Here we derive a recursive solution and we compare it to the Perceptron rule. The method is based on a *gradient descent* approach: compute the gradient of the error wrt $\mathbf{w}$ and move the weights in its opposite direction.



$$\mathbf{w}^{(k+1)} = \mathbf{w}^{(k)} - \mu\nabla_{\mathbf{w}}\mathbb{E}\left[e^2\right] \quad (9)$$

where

$$\nabla_{\mathbf{w}}\mathbb{E}\left[e^2\right] = \mathbf{X}\mathbf{X}^\top\mathbf{w} - \mathbf{X}\mathbf{y} \quad (10)$$

Now we approximate Eq. (10) by using a single sample:

$$\begin{aligned}\nabla_{\mathbf{w}}\mathbb{E}\left[e^2\right] &= \mathbf{X}\mathbf{X}^\top\mathbf{w} - \mathbf{X}\mathbf{y} \\ &\approx \mathbf{x}_k\mathbf{x}_k^\top\mathbf{w} - \mathbf{x}_k y_k \\ &= \mathbf{x}_k\left(\mathbf{x}_k^\top\mathbf{w} - y_k\right) \\ &= -e_k\mathbf{x}_k\end{aligned} \tag{11}$$

Where $e_k = y_k - \mathbf{x}_k^\top\mathbf{w}$. This leads to the following update rule

$$\mathbf{w}^{(k+1)} = \mathbf{w}^{(k)} + \mu e_k\mathbf{x}_k \tag{12}$$

Compare the LMS solution with the Perceptron rule in Eq. (5). The error as defined for LMS is $e_k = y_k - \mathbf{x}_k^\top \mathbf{w} = y_k - \mathbf{w}^\top \mathbf{x}_k$, but in the perceptron rule, the error is defined as

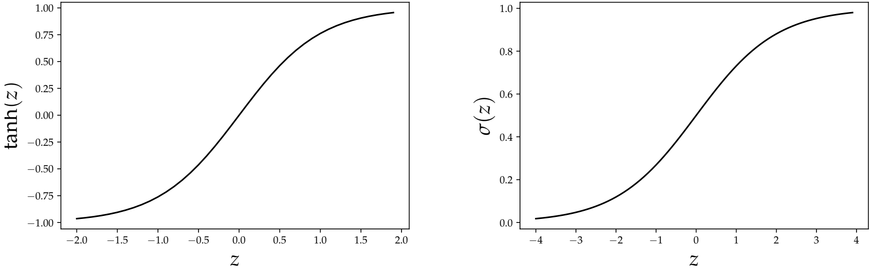
$$e_k = \frac{1}{2}\left(y_k - \text{sign}(\mathbf{w}^\top \mathbf{x}_k + b)\right) = \begin{cases} y_k, & \mathbf{x}_k \text{ misclassified} \\ 0, & \text{otherwise.} \end{cases}$$

If we use this error and $\mu = 1$ the LMS then becomes the perceptron rule

$$\mathbf{w}^{(k+1)} = \begin{cases} \mathbf{w}^{(k)} + y_i \mathbf{x}_i, & \mathbf{x}_k \text{ misclassified} \\ 0, & \text{otherwise.} \end{cases}$$

Soft activations

- The perceptron uses a hard output (sign detection) as an output, that translates the *affine* transformation $\mathbf{w}^\top \mathbf{x} + b$ into one of two states (-1, +1).
- In order to implement the LMS, we just remove the sign detection.
- A *sigmoid* function can be used to produce an output that can be interpreted as a soft state or a probability of a state.



Hyperbolic tangent function (left) and logistic function.

- The logistic function or the hyperbolic tangent are classic neural network activations, but nowadays, the logistic is used in combination with other functions that will be studied further.
- The hyperbolic tangent, the logistic function and their derivatives are:

$$\begin{aligned} \tanh(z) &= \tfrac{e^z - e^{-z}}{e^z + e^{-z}}, \quad & \tfrac{d}{dz}\tanh(z) &= 1 - \tanh^2(z) \\ \sigma(z) &= \tfrac{1}{1+e^{-z}}, & \tfrac{d}{dz}\sigma(z) &= \sigma(z)\left(1 - \sigma(z)\right) \end{aligned} \tag{13}$$

Now, if the output of the classifier is

$$f(\mathbf{x}) = \tanh\left(\mathbf{w}^\top \mathbf{x}_i + b\right) \tag{14}$$

then the criterion to optimize the parameters will be

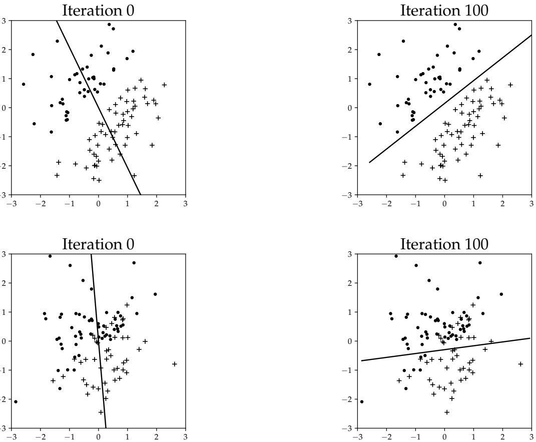
$$\min_{\mathbf{w},b} \mathbb{E}\left[e_i^2\right] \approx \min_{\mathbf{w},b} \sum_{i=1}^{N} \left(y_i - \tanh\left(\mathbf{w}^\top \mathbf{x}_i + b\right)\right)^2 \tag{15}$$

which leads to the update rule

$$\begin{aligned} \mathbf{w}^{(k+1)} &= \mathbf{w}^{(k)} + \mu \sum_{i=1}^{N} e_i \left(1 - f^2(\mathbf{x}_i)\right) \mathbf{x}_i \\ b^{(k+1)} &= b^{(k)} + \mu \sum_{i=1}^{N} e_i \left(1 - f^2(\mathbf{x}_i)\right) \end{aligned} \tag{16}$$

The proof is left as an exercise.

Example of the MMSE applied to a perceptron



Example of the application of the MMSE criterion to a perceptron with hyperbolic tangent activation. The first row corresponds to a separable case and the second one to a non separable one. In both cases the algorithm converges to a solution.