Introduction to Machine Learning (Lecture Notes)

Perceptron

Lecturer: Barnabas Poczos

1 History of Artificial Neural Networks

The history of artificial neural networks is like a roller-coaster ride. There were times when it was popular(up), and there were times when it wasn't. We are now in one of its very big time.

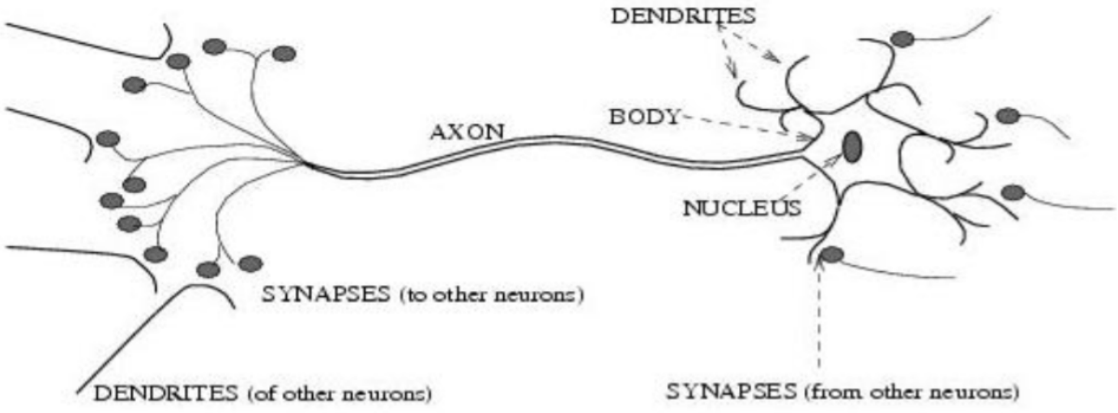
- Progression (1943-1960)
  - First Mathematical model of neurons
    - Pitts & McCulloch (1943) [MP43]
  - Beginning of artificial neural networks
  - Perceptron, Rosenblatt (1958) [R58]
    - A single neuron for classification
    - Perceptron learning rule
    - Perceptron convergence theorem [N62]
- Degression (1960-1980)
  - Perceptron can't even learn the XOR function [MP69]
  - We don't know how to train MLP
  - 1963 Backpropagation (Bryson *et al.*)
    - But not much attention
- Progression (1980-)
  - 1986 Backpropagation reinvented:
    - *Learning representations by back-propagation errors. Rumilhart et al.* **Nature** [RHW88]
  - Successful applications in
    - Character recognition, autonomous cars, ..., etc.
  - But there were still some **open questions** in
    - Overfitting? Network structure? Neuron number? Layer number? Bad local minimum points? When to stop training?
  - Hopfield nets (1982) [H82], Boltzmann machines [AHS85], ..., etc.
- Degression (1993-)

  - SVM: Support Vector Machine is developed by Vapnik *et al.*. [CV95] However, SVM is a **shallow** architecture.
  - Graphical models are becoming more and more popular
  - Great success of SVM and graphical models almost kills the ANN (Artificial Neural Network) research.
  - Training deeper networks consistently yields poor results.
  - However, Yann LeCun (1998) developed **deep convolutional neural networks** (a discriminative model). [LBBH98]

- Progression (2006-)
  Deep learning is a rebranding of ANN research.
  - **Deep Belief Networks (DBN)**
    * *A fast learning algorithm for deep belief nets. Hinton et al. Neural Computation.* [HOT06]
    * Generative graphical model
    * Based on restrictive Boltzmann machines
    * Can be trained efficiently
  - **Deep Autoencoder based networks**
    * *Greedy Layer-Wise Training of Deep Networks.Bengio et al. NIPS* [BLPL07]
  - **Convolutional neural networks running on GPUs**
    * Great success for NN since the massiave usage of GPUs.
    * *AlexNet (2012). Krizhevsky et al. NIPS* [KSH12]

To summarize, in the past, the popularity of artificial neural networks sometimes degraded due to bad performance or poor scalability. Recently, thanks to the advancement of computing power (GPUs), availability of big data, and the development of techniques for training deep neural networks on large dataset, artificial neural network and deep learning research have become very popular again.

2 The Neuron

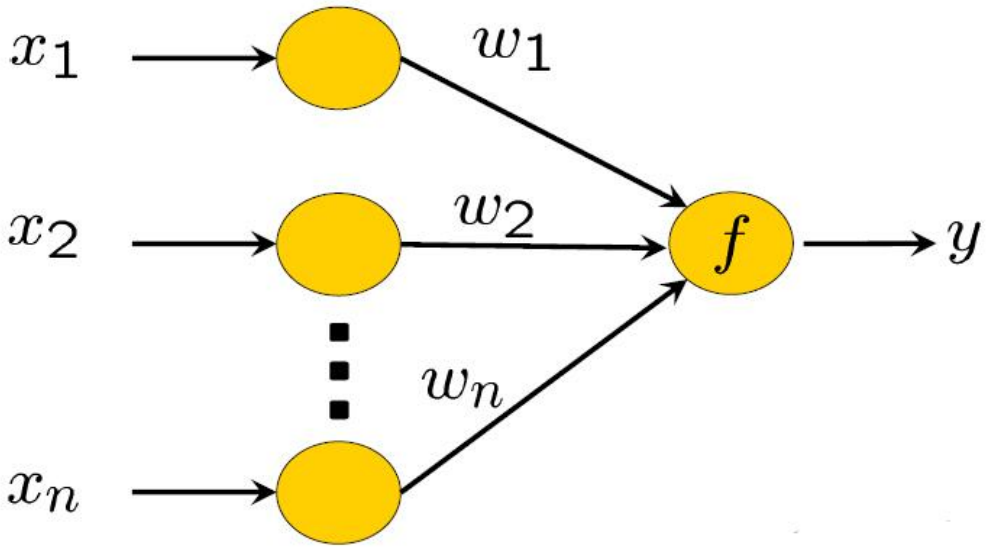


- Each neuron has a body, axon, and many dendrites.

- A neuron can fire or rest.
- If the sum of weighted inputs is larger than a threshold, then the neuron fires.
- Synapses: The gap between the axon and other neuron dendrites. It determines the weights in the sum.

2.1 Mathematical model of a neuron

The below figure illustrates the mathematical model of a neuron. Although inspired by biological neurons, this model has a number of simplifications and some of its design may not correspond to the actual functionality of biological neurons. However, due to its simplicity in computation and its effectiveness in practice, this is one of the most widely-used neuron models nowadays.

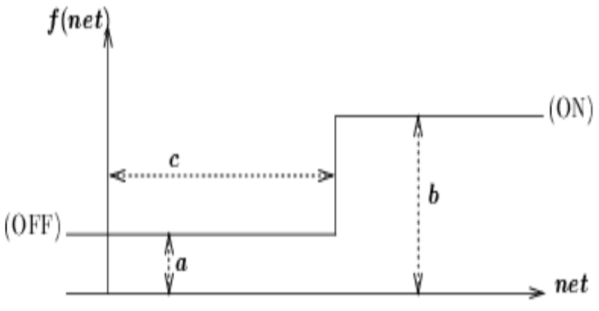


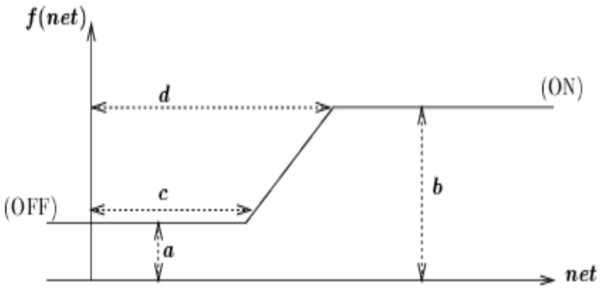
The output of a neuron $y = f(\omega_1 x_1 + \ldots + \omega_n x_n)$, is computed by applying a transformation f over the weighted sum of the input signals $x_1, x_2, \ldots, x_n$. The transformation f is called the activation function. The activation function is usually chosen to be non-linear. This allows neural networks to learn complex non-linear transformations over the input signal. Below is a list of different fs that are common.

Different activation functions:

Identify function: $f(\mathbf{x}) = \mathbf{x}$

Threshold function (perceptron):





Ramp function:

Logistic function: $f(x) = (1 + e^{-x})^{-1}$

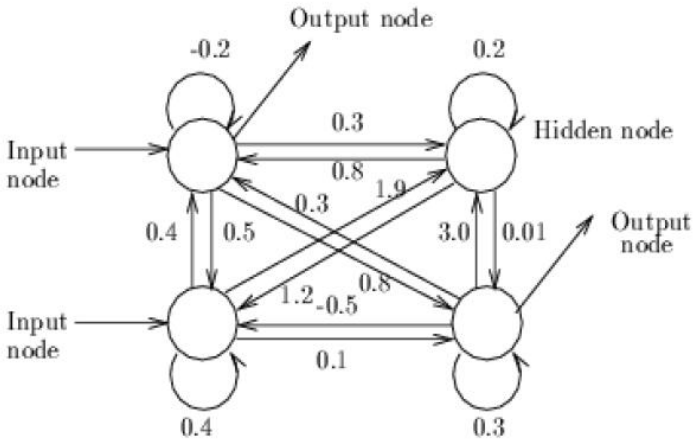
Hyperbolic tangent function: $f(x) = \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} = \frac{e^{2x}-1}{e^{2x}+1} = \frac{1-e^{-2x}}{1+e^{-2x}}$

3 Structure of Neural Networks

Neural networks are typically consists of neuron(nodes) and connections(edges). The structure of a neural network refers to the configuration of its neurons and connections.

3.1 An example: Fully connected neural network

The network is called fully connected if it has connections between every pair of nodes. It contains input neurons, hidden neurons and output neurons. Each connection has an associated weight as illustrated by the numbers on the edges.

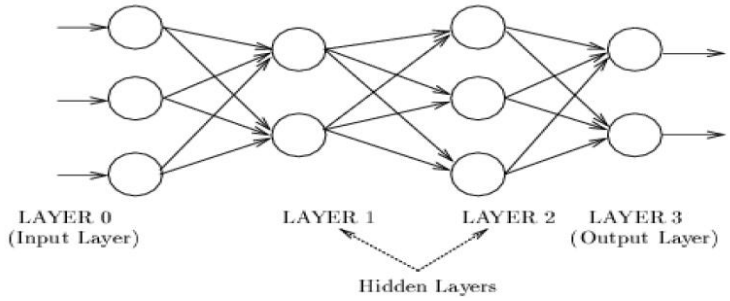


There are any models for updating the weights, e.g., asynchronous model.

3.2 Feedforward neural networks

Feedforward neural networks are networks that don't contain backward connections. In other words, messages in feedforward networks are passed layer by layer without feedbacks to previous ones. The connections

between the neurons do not form a cycle. The input layer is conventionally called Layer 0.



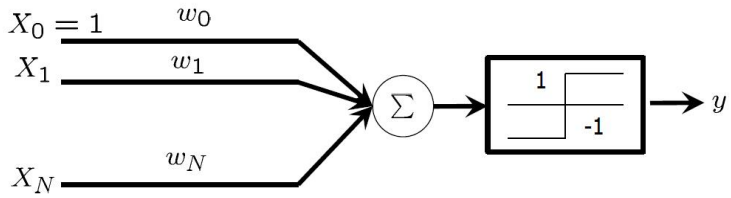
- There are no connections within the layers
- No backward connections
- An important special case is the **Multilayer perceptron**, where there are connections only between Layer i and Layer $i+1$
- Most popular architecture
- Recurrent NN: there are connections backward. A

Matlab demo nnd11nf can visualize these structures.

4 The Perceptron

The perceptron is a simple neuron with *sgn* (threshold) activation function. It can be used to solve simple binary classification problems. Formally it can be described with the following equation:

$y = sgn(w^T x) \in \{-1, 1\}$.



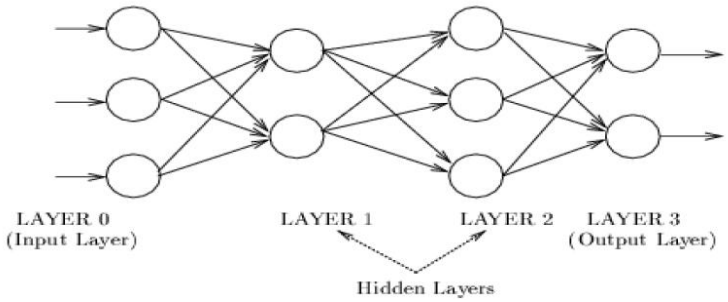
Training set:

- $X^+ = \{x_k | x_k \in Class+\}$.
- $X^- = \{x_k | x_k \in Class-\}$.
- Assume they are linearly separable.

w^* is the normal vector of the separating hyperplane through the origin, i.e.,

$$w^{*T}x > 0 \text{ if } x \in X^+,$$
$$w^{*T}x < 0 \text{ if } x \in X^-.$$

Note If we want to relax the assumption that w^* goes through the origin, we could add a bias term b in the vector w^*, and add a 1 in the input vecotr X.



Our goal is to find such w^*.

5 The Perceptron Algorithm

Notations

Let $\hat{y}(k)$ denote the predicted label of data $x(k)$, which is computed by

$$\hat{y}(k) = sgn(w(k-1)^T x(k)$$

Let $w(k)$ denote the weight of the k-th iteration. The update rule is given by

$$w(k) = w(k-1) + \mu(y(k) - \hat{y}(k))x(k)$$

The Algorithm

1. Let $\mu > 0$ arbitrary step size parameter. Let $w(0) = 0$ (it could also be arbitrary)
2. Let $k = 1$.
3. Let $x(k) \in X^+ \cup X^-$ be a training point misclassified by $w(k-1)$
4. If there is no such vector $\Rightarrow$ 6
5. If $\exists$ a misclassified vector, then
   $\alpha(k) = \mu\epsilon(k) = \mu(y(k) - \hat{y}(k))$
   $w(k) = w(k-1) + \alpha(k)x(k)$
   $k = k + 1$
   Back to 3

6. End

Observation:

$$\text{If } y(k) = 1,\ \hat{y}(k) = -1, \text{ then } \alpha(k) > 0.$$
$$\text{If } y(k) = -1,\ \hat{y}(k) = 1, \text{ then } \alpha(k) < 0.$$

How to remember: Gradient descent on $\frac{1}{2}(y(k) - \hat{y}(k))^2$ with learning rate μ:

$$w(k) = w(k-1) - \mu \frac{\partial \frac{1}{2}(y(k) - \hat{y}(k))^2}{\partial w(k-1)}$$

An interesting property: we do not require the learning rate to go to zero. (Many learning algorithms requires learning rate to degrade at some point in order to converge)

Why the algorithm works?

- Each input x_i determines a hyperplane orthogonal to x_i.
- On the + side of the hyperplane for each $w \in \mathbb{R}^n : w^T x_i > 0$, $sgn(w^T x_i) = 1$.
- On the − side of the hyperplane for each $w \in \mathbb{R}^n : w^T x_i < 0$, $sgn(w^T x_i) = -1$.
- We need to update the weights, if $\exists x_i$ in the training set, such that $sgn(w^T x_i) \neq y_i$, where $y_i = class(x_i) \in \{-1, 1\}$
- Then update w such that $\hat{y}_i$ gets closer to $y_i \in \{-1, 1\}$

Convergence of the Perceptron Algorithm

Theorem 1 *If the samples are linearly separable, then the perceptron algorithm finds a separating hyperplane in finite steps.*

Theorem 2 *The running time does not depend on the sample size n.*

Proof

Lemma 3 *Let*

$$\bar{X} = X^+ \cup \{-X^-\}$$

Then $\exists b > 0$, *such that* $\forall \bar{x} \in \bar{X}$ *we have* $w^{*T} \bar{x} \geq b > 0$.

Proof:

By definition we have that

$$w^{*T} x > 0 \text{ if } x \in X^+$$
$$w^{*T} x < 0 \text{ if } x \in X^-.$$

Therefore $\exists b > 0$, such that $\forall \bar{x} \in \bar{X}$ we have $w^{*T}\bar{x} \geq b > 0$. ■

We need an update step at iteration $k-1$, if $\exists \bar{x} \in \bar{X}$ such that $w(k-1)^T\bar{x} \leq 0$. Let this $\bar{x}$ be denoted by $\bar{x}(k)$.

According to the perceptron update,

$$w(k) = w(k-1) + \alpha\bar{x}(k) = \alpha\sum_{i=1}^{k}\bar{x}(i)$$

where $\alpha > 0$ is an arbitrary constant.

5.1 Lower bound on $\|w(k)\|^2$

Let us see how the weights change on set $\bar{X}$ while we run the perceptron algorithm. We can observe that

$$w^T(k)w^* = \alpha\sum_{i=1}^{k}\bar{x}(i)^Tw^* \geq \alpha kb.$$

From Cauchy-Schwartz

$$\|w(k)\|^2\|w^*\| \geq (w^T(k)w^*)^2 \geq \alpha^2k^2b^2.$$

Therefore,

$$\|w(k)\|^2 \geq \frac{\alpha^2k^2b^2}{\|w^*\|^2}$$

and thus $\|w(k)\|^2$ is at least quadratic in k.

5.2 Uppper bound

Let $w(0) = 0$, and let $M > \max_{\bar{x}(i)\in\bar{X}}\|\bar{x}(i)\|^2$.

According to the perceptron update,

$$w(k) = w(k-1) + \alpha\bar{x}(k),$$

and thus,

$$\|w(k)\|^2 = \|w(k-1)\|^2 + 2\alpha w^T(k-1)\bar{x}(k) + \alpha^2\|\hat{x}(k)\|^2$$

We know that $w^T(k-1)\hat{x}(k) \leq 0$ since we had to make an update step. From this it immediately follows that $\|w(k)\|^2 - \|w(k-1)\|^2 \leq \hat{\alpha}^2\|\hat{x}(k)\|^2$.

Therefore,

$$\begin{aligned}
\|w(k)\|^2 - \|w(k-1)\|^2 &\leq \alpha^2\|\hat{x}(k)\|^2 \\
\|w(k-1)\|^2 - \|w(k-2)\|^2 &\leq \alpha^2\|\hat{x}(k-1)\|^2 \\
&\vdots \\
\|w(1)\|^2 - \|w(0)\|^2 &\leq \alpha^2\|\hat{x}(1)\|^2
\end{aligned}$$

Thus,

$$\Rightarrow \|w(k)\|^2 \leq \alpha^2 \sum_{i=1}^{k} \|\hat{x}(i)\|^2$$
$$\Rightarrow \|w(k)\|^2 \leq \alpha^2 kM$$

$\Rightarrow \|w(k)\|^2$ does not grow faster than a linear function in k.

However, we already proved that it is at least quadratic in k. The only way to resolve this 'contradiction' is if k is finite. Therefore the perceptron algorithm terminates in finite steps.

References

[MP43] McCulloch, Warren S. and Pitts, Walter, "A logical calculus of the ideas immanent in nervous activity," *The bulletin of mathematical biophysics* 5.4 (1943): 115-133.

[R58] Rosenblatt, Frank, "The perceptron: a probabilistic model for information storage and organization in the brain," *Psychological review* 65.6 (1958): 386.

[N62] Novikoff, A. B., "On convergence proofs on perceptrons," *Symposium on the Mathematical Theory of Automata* 12 (1962), 615-622.

[MP69] Minsky, Marvin and Papert, Seymour, "Perceptrons: An Introduction to Computational Geometry," (1969)

[RHW88] Rumelhart, David E., Hinton, Geoffrey E. and Williams, Ronald J., "Learning representations by back-propagating errors," *Cognitive modeling* 5.3 (1988): 1.

[H82] Hopfield, John J., "Neural networks and physical systems with emergent collective computational abilities," *Proceedings of the national academy of sciences* 79.8 (1982): 2554-2558.

[AHS85] Ackley, David H., Hinton, Geoffrey E. and Sejnowski, Terrence J., "A learning algorithm for Boltzmann machines," *Cognitive science* 9.1 (1985): 147-169.

[CV95] Cortes, Corinna and Vapnik, Vladimir, "Support-vector networks," *Machine learning* 20.3 (1995): 273-297.

[LBBH98] LeCun, Yann, Bottou, Lon, Bengio, Yoshua and Haffner, Patrick, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE 86* no. 11 (1998): 2278-2324.

[HOT06] Hinton, Geoffrey E., Osindero, Simon and Teh, Yee-Whye, "A fast learning algorithm for deep belief nets," *Neural computation* 18.7 (2006): 1527-1554.

[BLPL07] Bengio, Yoshua, Lamblin, Pascal, Popovici, Dan and Larochelle, Hugo, "Greedy layer-wise training of deep networks," *Advances in neural information processing systems* 19 (2007): 153.

[KSH12] Krizhevsky, Alex, Sutskever, Ilya and Hinton, Geoffrey E., "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems* pp. 1097-1105. 2012.