Data Science

Marc Tommasi

November 21, 2023

Outline

Outline

Cybernetics: 50's

- Bio-inspiration
- McCullloch-Pitts 43: neuron model
- Rosenblatt 58/62: learning weights
- Widrow-Hoff 60: Adaline as an instance of stochastic gradient descent
- Minsky-Papert 69: limitation by XOR example

Connectionism: 1980s

- a complex task is the result of the composition of simple tasks.
- gradient backpropagation: Rumelhart 86 / Lecun 87
- use of unlabeled data to facilitate deep network learning
- no longer too closely linked to neuroscience (little is known about the brain, so getting closer to it has yielded nothing).

Deep learning: since the mid-2000s

- Lots of (labeled) data
- Powerful machines (GPUs, etc.)
- Explosion in network size (1000 billion parameters?)
- Resolutely inspired by statistics, probability, optimization, numerical calculation, etc.

From Goodfellow's book

The choice of the functions used to compute these representations is also loosely guided by neuroscientific observations about the functions that biological neurons compute. Modern neural network research, however, is guided by many mathematical and engineering disciplines, and the goal of neural networks is not to perfectly model the brain. It is best to think of feedforward networks as function approximation machines that are designed to achieve statistical generalization, occasionally drawing some insights from what we know about the brain, rather than as models of brain function

Evolution I

Evolution of model types

- Linear model, single neuron (perceptron)
- Multi-layer models,
- Convolution networks,
- Recurrent models,
- Models with attention.

Evolution of data sets

- from a few hundred or thousand examples
- to millions and billions today
- 10 million labeled examples for near-human performance in many cases...

Evolution II

Performance evolution

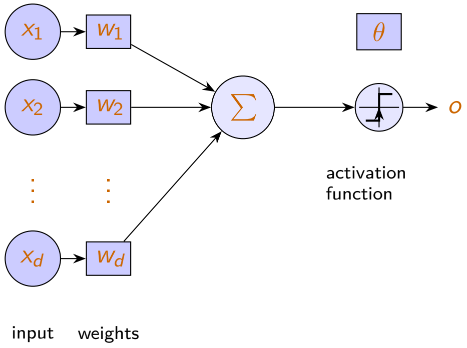
- Impressive impact
  - in vision
  - speech recognition
  - natural language processing

Outline

Supervised classification

- Data represented by d features: $\boldsymbol{x} \in \mathbb{R}^d$
- Binary classification: y is in $\{-1, 1\}$
- A labeled data set: $\{(\boldsymbol{x}_1, y_1) \ldots, (\boldsymbol{x}_m, y_m)\}$

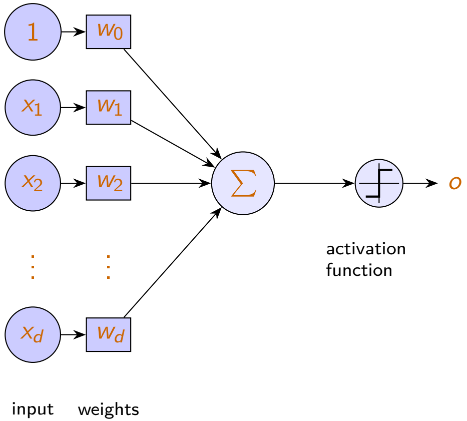
Perceptron linéaire à seuil



- d inputs $x_1, \ldots x_d$; one output: o is a binary value
- definition: d (synaptic) coefficients $w_1, \ldots, w_d$, bias θ ,

$$o = \begin{cases} 1 & \text{if } \boldsymbol{w}^\top \boldsymbol{x} > \theta \\ -1 & \text{otherwise} \end{cases}$$

Perceptron linéaire à seuil



- d inputs $x_1, \dots x_d$; one output: o is a binary value
- definition: $d+1$ (synaptic) coefficients $w_0, \dots, w_d$, additional entry $x_0 = 1$,

$$o = \begin{cases} 1 & \text{if } \boldsymbol{w}^\top \boldsymbol{x} > 0 \\ -1 & \text{otherwise} \end{cases}$$

Interpretation

Example (Exercises)

- Propose a perceptron that calculates a logical OR between n binary variables (taking the values 0 or 1).
- Give a graphical representation of what you've proposed in the case of two variables: represent the possible values and the decision boundary for the output (considering this time that the x_i values can be real) by points of two colors.
- Now represent the points corresponding to an XOR function.
- Observe that a linear threshold perceptron cannot separate the points corresponding to the XOR.

Learning algorithm: Intuition I

Intuition

- let's consider values for $\boldsymbol{w}$, how can we adapt them to an example $(\boldsymbol{x}, y)$?
- Recall that the weights w_i associated with inputs x_i that are 0 are not involved in the decision.
- Let's assume $y = 1$.
  - If $\boldsymbol{w}^\top \boldsymbol{x} > 0$ then we have nothing to do.
  - If $\boldsymbol{w}^\top \boldsymbol{x} <= 0$ then we need to increase the weights associated with inputs x_i which are 1.
- Let's assume $y = -1$.
  - If $\boldsymbol{w}^\top \boldsymbol{x} <= 0$ then we have nothing to do.
  - If $\boldsymbol{w}^\top \boldsymbol{x} > 0$ then we must decrease the weights associated with the x_i entries which are 1.

Learning algorithm: Intuition II

Rosenblatt's algorithm, 57

```
Input: a sample $\{(\boldsymbol{x}_1, y_1) \dots, (\boldsymbol{x}_m, y_m)\}$
$t \leftarrow 0$
Initialize $\boldsymbol{w}^{(0)}$ with 0 values
Repeat
    get a new example $(\boldsymbol{x}_k, y_k)$ from $S$.
    $\hat{y} \leftarrow \mathrm{sign}(\boldsymbol{w}^{(t)\top}\boldsymbol{x}_k)$
    $t \leftarrow t + 1$
    if $(\hat{y} \neq y_k)$ then $\boldsymbol{w}^{(t)} \leftarrow \boldsymbol{w}^{(t-1)} + y_k\boldsymbol{x}_k$
until convergence
output: $\boldsymbol{w}^{(t)}$
```

Learning algorithm: Intuition III

Example with OR

- Start with $w_0 = 0$; $w_1 = 1$ et $w_2 = -1$.
- data: $S = \{((1,0,0),-1),((1,0,1),1),((1,1,0),1),((1,1,1),1)\}$.
- run the algorithm!

Property I

If the sample is linearly separable, if the examples are presented fairly, the algorithm stops and calculates a linear separator for S.

Theorem (Convergence (Novikoff, 62))

Let $\boldsymbol{x}_1, \dots, \boldsymbol{x}_m \in \mathbb{R}^d$ such that there exists $r > 0$ with $\|\boldsymbol{x}_k\| \leq r$ for all $k \in [1, m]$. If there exists $\rho > 0$ and $v \in \mathbb{R}^d$, $\|\boldsymbol{v}\| = 1$, such that for all $k \in [1, m]$, $\rho \leq y_k(\boldsymbol{v}^\top \boldsymbol{x}_k)$, then the number of iterations of the perceptron algorithm with $\boldsymbol{x}_1, \dots, \boldsymbol{x}_m$ as input is bounded by r^2/ρ^2.

- r: is the maximum norm of the data
- $\boldsymbol{v}$: is the separator hyperplane
- ρ: is the margin

Property II

Sketch of proof, lower bound

Let's assume that at step n, the algorithm has already performed M updates at times t_1, t_2, t_M.

$$\begin{aligned}\|\boldsymbol{w}_n\|^2 &= \|\boldsymbol{w}_{t_M} + y_{t_M}\boldsymbol{x}_{t_M}\|^2 \\ &= \|\boldsymbol{w}_{t_M}\|^2 + \|\boldsymbol{x}_{t_M}\|^2 + 2y_{t_M}\boldsymbol{w}_{t_M}^\top\boldsymbol{x}_{t_M}\end{aligned}$$

From the algorithm, $y_{t_M}\boldsymbol{w}_{t_M}^\top\boldsymbol{x}_{t_M} < 0$ on a

$$\|\boldsymbol{w}_n\|^2 \leq \|\boldsymbol{w}_{t_M}\|^2 + \|\boldsymbol{x}_{t_M}\|^2$$

.

By induction

$$\|\boldsymbol{w}_n\|^2 \leq \|\boldsymbol{w}_0\|^2 + \sum_{i=1}^{M}\|\boldsymbol{x}_{t_i}\|^2 \leq r^2 M$$

Property III

Sketch of proof, upper bound

We have $\mathbf{v}$ which is unitary (of norm 1) so, for any vector $\mathbf{w}$, the norm of $\mathbf{w}$ is greater than the scalar product $\mathbf{v}^\top \mathbf{w}$. Here,

$$\begin{aligned}\|\mathbf{w}_n\| &\geq \mathbf{v}^\top \mathbf{w}_n \\ &\geq \mathbf{v}^\top(\mathbf{w}_{t_M} + y_{t_M}\mathbf{x}_{t_M}) \\ &\geq \mathbf{v}^\top \mathbf{w}_{t_M} + y_{t_M}\mathbf{v}^\top \mathbf{x}_{t_M} \\ &\geq \mathbf{v}^\top \mathbf{w}_{t_M} + y_{t_M}\rho\end{aligned}$$

By induction (assuming we initialize $\mathbf{w}_0$ to 0)

$$\|\mathbf{w}_n\| \geq M\rho$$

In conclusion, we have $\|\mathbf{w}_n\|^2 \geq M^2\rho^2$ and $\|\mathbf{w}_n\|^2 \leq r^2 M$. So

$$M \leq r^2/\rho^2$$

Properties, limitations

- The smaller the margin, the longer it may take the algorithm to converge.
- Note that the vector $\boldsymbol{w}$ is calculated with combinations of the examples $\boldsymbol{x}$.
- Weights are integers
- No guarantee of convergence in the non-separable case
- Many efforts to approximate and tolerate errors
- The algorithm can be seen as a stochastic gradient descent. . .

Stochastic Gradient Descent

Algorithm

```
• Input: $S = \{(\boldsymbol{x}_1, y_1) \ldots, (\boldsymbol{x}_m, y_m)\}$, a loss function $f$
• Set weights $\boldsymbol{w}^{(0)}$ to random values
• For $t$ from 1 to $T$
    ▸ Choose $\boldsymbol{x}, y$ in $S$
    ▸ $\boldsymbol{w}^{(t)} \leftarrow \boldsymbol{w}^{(t-1)} - \eta \nabla_{\boldsymbol{w}^{(t)}} f(\boldsymbol{w}, \boldsymbol{x}, y)$
• Output: $\boldsymbol{w}^{(T)}$
```

- Numerous loss functions can be used depending on the case (MSE (linear regression etc...), logistic (logistic regression), exponential (in Adaboost), Hinge loss
- Hinge Loss : $\max(0, 1 - y_k \boldsymbol{w}^\top x_k)$
  - Derivative: $-y_k x_k$ if signs of y_k and $\boldsymbol{w}^\top x_k$ differ
  - We find the algorithm by error correction with $\eta = 1$.