Introduction to Machine Learning

Perceptrons

Mingchen Gao

September 28, 2022

Outline

Contents

1 Perceptrons

- Number of neurons 10^{10-11}
- Connections per neuron 10^{4-5}
- Switching time 0.001 seconds
- Scene recognition time 0.1 seconds
- Number of cycles per scene recognition 100

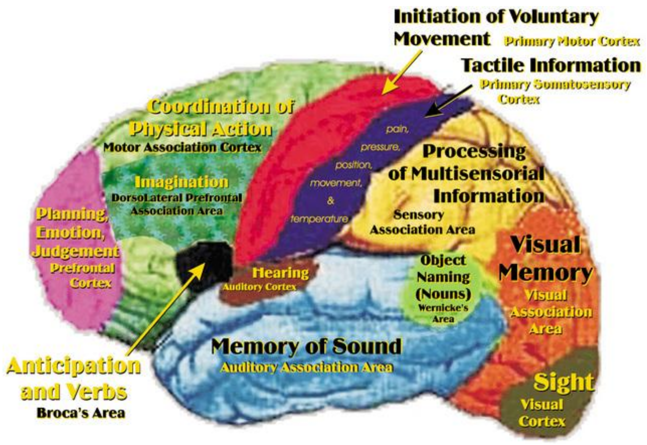


Figure 1: *Src: http://brainjackimage.blogspot.com/*

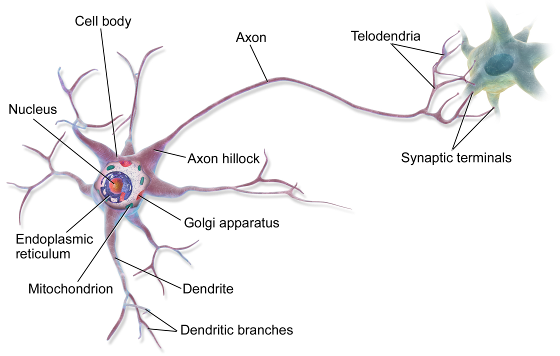
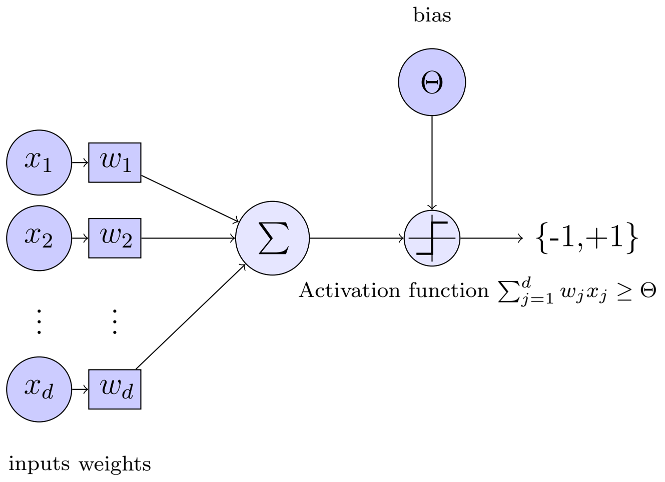


Figure 2: *Src: Wikipedia*

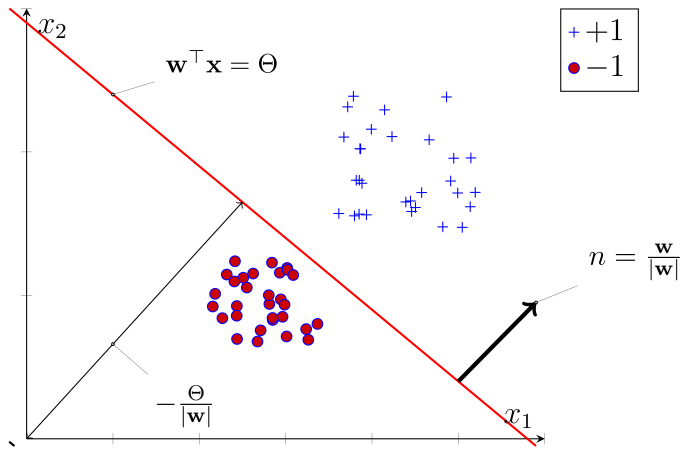
So far we have focused on learning concepts defined over d binary attributes. Now we generalize this to the case when the attributes are numeric, i.e., they can take infinite values. Thus the concept space becomes infinite. How does one search for the target concept in this space and how is their performance analyzed. We begin with the simplest, but very effective, algorithm for learning in such a setting, called **perceptrons**. Perceptron algorithm was developed by Frank Rosenblatt [3] at the Calspan company in Cheetowaga, NY. In fact, it is not only the first machine learning algorithm, but also the first to be implemented in hardware, by Rosenblatt. The motivation for perceptrons comes directly from the model by McCulloch-Pitts [1] for the artificial neurons. The neuron model states that multiple inputs enter the main neuron (*soma*) through multiple *dendrites* and the output is sent out through a single *axon*.



Simply put, a perceptron computes a weighted sum of the input attributes and adds a bias term. The sum is compared to a threshold to classify the input as $+1$ or -1.

The bias term Θ signifies that the weighted sum of the actual attributes should be greater than Θ to become greater than 0.

1.1 Geometric Interpretation



x_2
$\mathbf{w}^\top \mathbf{x} = \Theta$
+ $+1$
● -1
$n = \frac{\mathbf{w}}{|\mathbf{w}|}$
$-\frac{\Theta}{|\mathbf{w}|}$
x_1

1. The hyperplane, characterized by $\mathbf{w}$ is also known as the decision boundary (or surface).
2. The decision boundary divides the input space into two *half-spaces*.
3. Changing $\mathbf{w}$ *rotates* the hyperplane
4. Changing Θ *translates* the hyperplane
5. Hyperplane passes through the origin if $\Theta = 0$

1.2 Perceptron Training

- Add another attribute $x_{d+1} = 1$.
- w_{d+1} is $-\Theta$
- Desired hyperplane goes through origin in $(d + 1)$ space
- **Assumption**: $\exists \mathbf{w} \in \mathbb{R}^{d+1}$ such that $\mathbf{w}$ can *strictly* classify all examples correctly.
- *Hypothesis space*: Set of all hyperplanes defined in the $(d+1)$-dimensional space passing through origin
  - The target hypothesis is also called **decision surface** or **decision boundary**.

```
1:  $\mathbf{w} \leftarrow (0, 0, \ldots, 0)_{d+1}$
2:  for $i$=1, 2, ... do
3:      if $\mathbf{w}^\top \mathbf{x}^{(i)} > 0$ then
4:          $c(\mathbf{x}^{(i)}) = +1$
5:      else
6:          $c(\mathbf{x}^{(i)}) = -1$
7:      end if
8:      if $c(\mathbf{x}^{(i)}) \neq c_*(\mathbf{x}^{(i)})$ then
9:          $\mathbf{w} \leftarrow \mathbf{w} + c_*(\mathbf{x}^{(i)})\mathbf{x}^{(i)}$
10:     end if
11: end for
```

- Every mistake *tweaks* the hyperplane
  - Rotation in $(d + 1)$ space
  - Accomodate the offending point
- Stopping Criterion:
  - Exhaust all training examples, or
  - No further updates

To demonstrate the working of perceptron training algorithm, let us consider a simple 2D example in which all data points are located on a unit circle.

To get an intuitive sense of why this training procedure should work, we should note the following: Every mistake makes $\mathbf{w}^\top \mathbf{x}$ become more positive (if $c_*(x) = 1$) or more negative (if $c_*(x) = -1$).

Let $c_*(x) = 1$. The new $\mathbf{w}' = \mathbf{w} + \mathbf{x}$. Hence, by substitution, $\mathbf{w}'^\top \mathbf{x} = (\mathbf{w} + \mathbf{x})^\top \mathbf{x} = \mathbf{w}^\top \mathbf{x} + \mathbf{x}^\top \mathbf{x}$. The latter quantity is always positive, thereby making $\mathbf{w}^\top \mathbf{x}$ more positive.

Thus whenever a mistake is made, the surface is "moved" to accomodate that mistake by increasing or decreasing $\mathbf{w}^\top \mathbf{x}$.

Sometimes a "learning rate" (η) is used to update the weights.

2 Perceptron Convergence

1. Linearly separable examples

2. No errors
3. $|\mathbf{x}| = 1$
4. A positive δ gap exists that "contains" the target concept (hyperplane)
    - $(\exists\delta)(\exists\mathbf{v})$ such that $(\forall\mathbf{x})\mathbf{v}^\top\mathbf{x} > c_*(\mathbf{x})\delta$.

The last assumptions "relaxes" the requirement that w converges to the target hyperplane exactly.

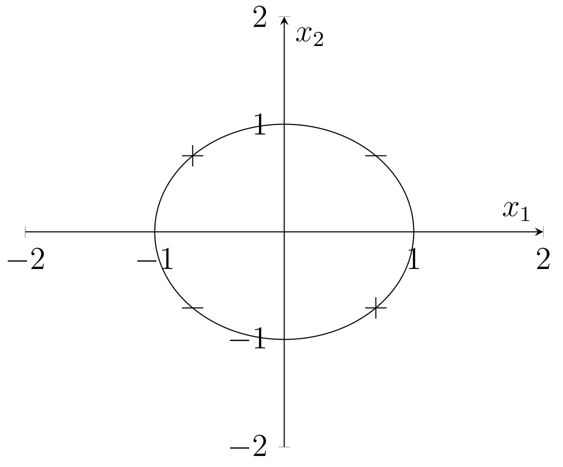
Theorem 1. *For a set of unit length and linearly separable examples, the perceptron learning algorithm will converge after a finite number of mistakes (at most $\frac{1}{\delta^2}$).*

Proof discussed in Minsky's book [2]. Note that while this theorem guarantees convergence for the algorithm, its dependence on δ makes it inefficient for *hard* classes. In his book, *Perceptrons* [2] Minsky criticized perceptrons for the same reason. If, for a certain class of problem, δ is very small, the number of mistakes will be very high.

Even for classes over boolean variables, there exist cases where the gap δ is exponentially small, $\delta \approx 2^{-kd}$. The convergence theorem states that there could be $\approx 2^{kd}$ mistakes before the algorithm converges. Note that for d binary attributes, one can have $2^\Theta(d^2)$ possible linear threshold hyperplanes. Using the halving algorithm, one can learn the hyperplane with $O(\log_2 2^\Theta(d^2)) = O(d^2)$ mistakes, instead of $O(\frac{1}{\delta^2})$ mistakes made by the perceptron learning algorithm. But the latter is implementable and works better in practice.

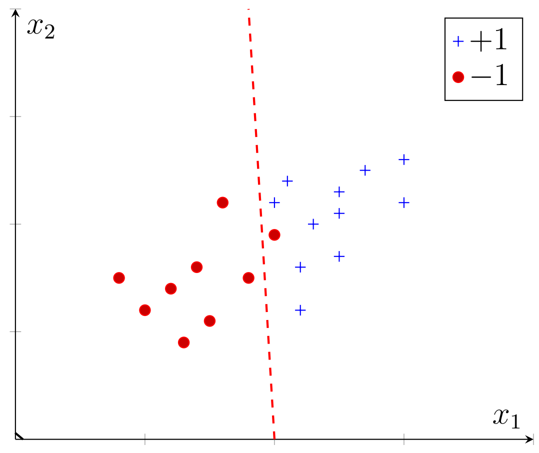
3 Perceptron Learning in Non-separable Case

- Expand $\mathcal{H}$?
- Lower expectations
    - *Principle of good enough*



Expanding $\mathcal{H}$ so that it includes the target concept is tempting, but it also makes the search more challenging. Moreover, one should always keep in mind that $\mathcal{C}$ is mostly unknown, so it is not even clear how much $\mathcal{H}$ should be expanded to.

Another way to address the XOR problem is to use different input attributes for the examples. For example, in this case, using the polar attributes, $\langle r, \theta \rangle$ can easily result in a simple threshold to discriminate between the positive and negative examples.



The example above clearly appears to be non-separable. Geometrically, one can always prove if two sets of points are separable by a straight line or not. If the convex hulls for each set intersect, the points are not linearly separable, otherwise they are.

To learn a linear decision boundary, one needs to "tolerate" mistakes. The question then becomes, which would be the best possible hyperplane, allowing for mistakes on the training data. Note that at this point we have moved from *online learning* to *batch learning*, where a batch of training examples is used to learn the best possible linear decision boundary.

In terms of the hypothesis search, instead of finding the target concept in the hypothesis space, we relax the assumption that the target concept belongs to the hypothesis space. Instead, we focus on finding the *most probable* hypothesis.

4 Gradient Descent and Delta Rule

- Which hyperplane to choose?
- Gives **best performance** on training data
  - Pose as an optimization problem
  - Objective function?
  - Optimization procedure?

4.1 Objective Function for Perceptron Learning

- An unthresholded perceptron (a linear unit)
- Training Examples: $\langle \mathbf{x}_i, y_i \rangle$
- Weight: $\mathbf{w}$

$$E(\mathbf{w}) = \frac{1}{2}\sum_i (y_i - \mathbf{w}^\top \mathbf{x}_i)^2$$

Note that we are denoting the weight vector as a vector in the coordinate space (denoted by w). The output y_i is a binary output $(0, 1)$.

References

References

[1] W. Mcculloch and W. Pitts. A logical calculus of ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5:127–147, 1943.

[2] M. L. Minsky and S. Papert. *Perceptrons: An Introduction to Computational Geometry*. MIT Press, 1969.

[3] F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386–408, 1958.