Artificial Neural Networks
Lecture Notes

Prof. Dr. Sen Cheng

Winter Semester 2019/20

8 Perceptron

8.1 Photos

See slides for historical photos of hardware implementation and news coverage.

8.2 Model class

The *perceptron* was invented by Frank Rosenblatt (Rosenblatt, 1958). It is a very simple neural network with n input units, and $m = 1$ output unit. The goal of the perceptron is to decide which of two classes $\{-1, 1\}$ the input belongs to. These class labels are chosen here for notational convenience. It is quite common to use 0 for the out-of-class label instead, but that doesn't change the way the perceptron works. The perceptron function maps a high-dimensional input to a binary variable and is therefore useful for solving classification problems. Like logistic regression, the perceptron uses a *linear decision boundary* (Fig. 1).

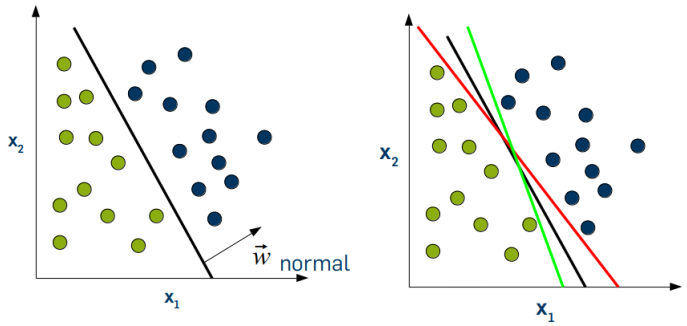

Figure 1: The perceptron implements a linear decision boundary (left). The boundary is given by the equation $w_0 + w_1 x_1 + w_2 x_2 = 0$ or $w^T x = 0$ in vector notation. Solutions found by the perceptron are not unique (right).

$$\hat{y} = f(x) = \begin{cases} -1 & w^T x \leq 0 \\ +1 & w^T x > 0 \end{cases} \tag{1}$$

The parameters of the perceptron are therefore the vector w, which is orthogonal to the decision boundary (Fig. 1). Note that in this formulation w has $n+1$ components, so that w_0 is the bias term and $x_0 = 1$ has to be set in the input.

x_1	x_2	output
0	0	-1
0	1	-1
1	0	-1
1	1	1

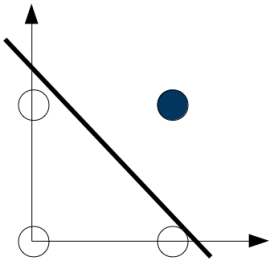

Figure 2: The AND function and a linear decision boundary.

8.2.1 Example: Boolean AND function

An an exercise, we manually determine a perceptron solution that implements the AND function (Fig. 2). We can define a decision boundary by

$$x_2 = 1.5 - x_1 \tag{2a}$$
$$0 = 1.5 - x_1 - x_2 \tag{2b}$$
$$= \begin{bmatrix} 1.5 & -1 & -1 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \end{bmatrix} \tag{2c}$$

Note that we can multiply the weight vector by any scalar and the equation would still hold, i.e., there is an infinite number of solutions to describe the boundary in Fig. 1. However, not all weight vector that generate the right boundary also yields a correct solution. For instance, the one above does not, because for $(1,1)$ it predicts the incorrect output -1. The solution is therefore to multiply the weight with a negative scalar:

$$w = \begin{bmatrix} -1.5 \\ 1 \\ 1 \end{bmatrix} \tag{2d}$$

Note that this solution is not unique. Any

$$w = \begin{bmatrix} z \\ 1 \\ 1 \end{bmatrix} \tag{3}$$

where $-2 < z \leq -1$ is also a solution. In addition, αw with $\alpha > 0$ is also a solution.

8.3 The Perceptron algorithm

The perceptron training algorithm (Alg. 8.1) was the first one suggested for any neural network.

Algorithm 8.1 Perceptron Algorithm (Rosenblatt, 1958)

Require: X, Y

initialize parameters w with 0 or small random values
repeat
  for all data pairs i **do**
    Calculate the perceptron output: $\hat{y}_i(x_i)$
    Update weights using the rule: $w \leftarrow w + \eta\,(y_i - \hat{y}_i)\,x_i$
  end for
until $\frac{1}{N}\sum_{i=1}^{N} |y_i - \hat{y}_i| \leq \mu$ or predetermined number of iterations is reached

If classification is already correct, $y_i = \hat{y}_i$ and the weight update term vanished. In other words, the weight are only updated if there is an error. This property is sometimes called error-driven learning.

8.4 Geometric interpretation

To further analyze the perceptron learning algorithm, it is convenient to implement the algorithm a little differently. First, $y_i\hat{y}_i = 1$ if the classification by the perceptron is correct, and $y_i\hat{y}_i = -1$ otherwise. Second, if weights need updating, $y_i - \hat{y}_i = 2y_i$. We can therefore rewrite the perceptron algorithm in the following form:

Algorithm 8.2 Perceptron Algorithm - Variant

Require: X, Y
1: initialize parameters w with 0 or small random values
2: initialize errors=0
3: **repeat**
4: **for all** data pairs i **do**
5: **if** $y_i\hat{y}_i = -1$ **then**
6: Update weights using the rule: $w \leftarrow w + 2\eta y_i x_i$
7: errors $\leftarrow$ errors +1
8: **end if**
9: **end for**
10: **until** errors$/N \leq \mu$ or predetermined number of iterations is reached

The corrections term always points in the direction of either x_i (if $y_i = 1$), or $-x_i$ (if $y_i = -1$). To illustrate how this helps the algorithm find the category boundary consider the following example in Fig. 3. Since the vector w is

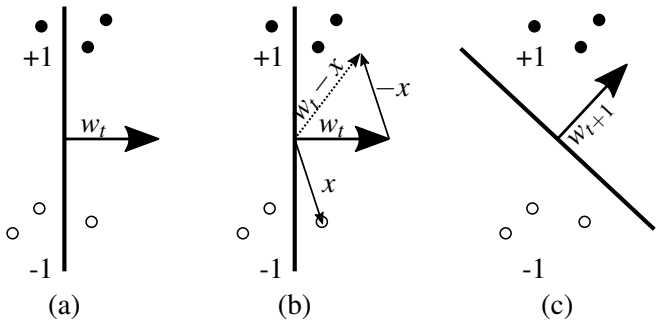

Figure 3: One update in the perceptron algorithm.

normal to the decision boundary, it should point in the direction of the items in the $+1$ class (Fig. 3, right). If this is not the case, then it needs to be rotated towards the class. If the item under analysis x_i is a member of the class, rotating w towards it is a pretty good bet. If the item under analysis x_i is not a member of the class ($y_t = -1$), w is rotated away from the item, i.e. towards $-x_i$.

8.5 Convergence – first steps

Since the perceptron algorithm updates the weights incrementally, an important question is whether the algorithm ever converges, i.e., do the weight updates stop at some point by themselves, if we didn't stop it? That would occur if and only if all items are classified correctly.

Before we proof convergence in general, we make a small first step. Given that a certain item x is misclassified, how many times in a row could the perceptron misclassify the same item, if testing and updating are applied repeatedly to the same item?

For simplicity, we denote the initial weight vector as w_0 and assume that $w_0 \neq 0$. $x \neq 0$ because of the bias component. If the item is misclassified, the weights are updated such that $w_1 = w_0 + 2\eta yx$, then $w_2 = w_1 + 2\eta yx = w_0 + 4\eta yx$, and so on. After k updates, the weight vector $w_k = w_0 + 2k\eta yx$.

If $w \neq 0$ and $yw^T x \geq 0$, then the item is classified correctly, because the sign of $w^T x_i$ determines the classification

$\hat{y}$. So, as long as x is misclassified,

$$yw_k^T x = y\left(w_0 + 2k\eta yx\right)^T x \tag{4a}$$

$$= yw_0^T x + 2k\eta \underbrace{y^2}_{1} x^T x < 0 \tag{4b}$$

Solving for k

$$k < -\frac{yw_0^T x}{2\eta x^T x} \tag{5}$$

This proofs that a given feature vector can be misclassied by the perceptron only a finite number of times. So, if we had only one feature vector in our training set, the perceptron would converge for sure.

8.6 Convergence proof

Minsky & Papert (1969) proofed that the Perceptron will find a solution after a finite number of updates (converge), if the data are linearly separable. The number of updates depends on the data set.

Preliminaries: Assume that data is *linearly separable*, i.e., there exist a solution $\hat{w}$ such that $y_i \hat{w}^T x_i \geq 0 \ \forall i$. Without loss of generality, assume $\|\hat{w}\| = 1$, $\|x_i\| \leq 1$, and $\mu = 0$. Define the *margin* as the shortest distance between the decision boundary and any feature vector, i.e.,

$$\gamma = \min_i |\hat{w}^T x_i| \tag{6}$$

That is, $\gamma \leq |\hat{w}^T x_i| \ \forall i$. Due to Cauchy-Schwarz inequality

$$|\hat{w}^T x_i| \leq \underbrace{\|\hat{w}\|}_{1} \underbrace{\|x_i\|}_{\leq 1} \leq 1 \tag{7}$$

Hence $\gamma \leq |\hat{w}^T x_i| \leq 1 \ \forall i$. We also know that $|\hat{w}^T x_i| = y_i \hat{w}^T x_i$ and $\gamma \geq 0$ because of linear separability. Hence, altogether

$$0 \leq \gamma \leq y_i \hat{w}^T x_i \leq 1 \ \forall i \tag{8}$$

Step 1: Show that with each update w become more similar to $\hat{w}$, i.e., $w^T \hat{w}$ is monotonically increasing with each update. Limit analysis to cases when an error occurs. If no error ocurrs, the algorithm is done.

$$w^T \hat{w} \leftarrow (w + 2\eta y_i x_i)^T \hat{w} = w^T \hat{w} + 2\eta \underbrace{y_i x_i^T \hat{w}}_{\geq \gamma} \geq w^T \hat{w} + 2\eta\gamma \tag{9}$$

Step 2: The previous could be accomplished trivially by increasing $\|w\|$ without getting closer to $\hat{w}$. So, we need to look at how $w^T w$ is changing in each update.

$$w^T w \leftarrow (w + 2\eta y_i x_i)^T (w + 2\eta y_i x_i) \quad = \quad w^T w + 2\eta \underbrace{y_i w^T x_i}_{\leq 0} + 2\eta \underbrace{y_i x_i^T w}_{= y_i w^T x_i} + 4\eta^2 \underbrace{y_i^2}_{=1} \underbrace{x_i^T x_i}_{\leq 1} \tag{10}$$

$$\leq \quad w^T w + 4\eta^2 \tag{11}$$

Step 3: Starting with $w = 0$, after a finite number of updates M, in each of which a misclassified element is encountered:

$$w^T \hat{w} \quad \geq \quad 2\eta\gamma M \tag{12}$$

$$w^T w \quad \leq \quad 4\eta^2 M \tag{13}$$

Step 4: Derive relationship between the two constraints.

$$2\eta\gamma M \leq w^T \hat{w} \leq \underbrace{|w^T \hat{w}| \leq \|w\| \, \|\hat{w}\|}_{\text{Cauchy-Schwarz inequality}} = \|w\| = \sqrt{w^T w} \leq \sqrt{4\eta^2 M} \tag{14}$$

Hence,

$$M \leq \frac{1}{\gamma^2} \tag{15}$$

This result indicates that convergence is faster if the margin is larger.

8.7 Limitations of the perceptron

- Solutions found by the perceptron are not unique (Fig. 1).
- If the data is not linearly separable, convergence is not guaranteed. Counterexamples are simple functions such as the Boolean XOR function. It is not linearly separable (Fig. 4) and the perceptron algorithm will not converge, if applied to this data. This has become known as the *XOR problem*. If the data is not linearly separable, there will be oscillation (which could be detected automatically).

x_1	x_2	output
0	0	-1
0	1	1
1	0	1
1	1	-1

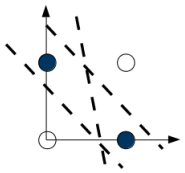

Figure 4: The XOR function (left) is not linearly separable (right) and cannot be learned by the perceptron.

References

Minsky, M. & Papert, S. (1969). *Perceptrons*. Perceptrons. Oxford, England: M.I.T. Press.

Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. *Psychol. Rev.*, *65*(6), 386–408. tex.mendeley-tags= theory tex.pmid= 13602029 tex.type= Journal article.