

Perceptron

1 Perceptron

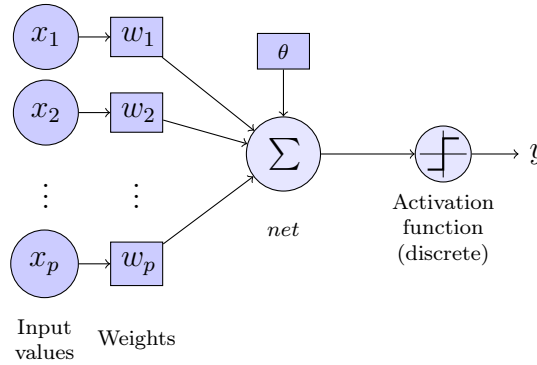


Figure 1: Perceptron with a discrete activation function.

A perceptron consists of the following:

- input vector $\mathbf{x} = (x_1 \ x_2 \ \dots \ x_p)^T$,
- weight vector $\mathbf{w} = (w_1 \ w_2 \ \dots \ w_p)^T$,
- bias θ ,
- activation function f ,
- output y .

1.1 Output value

The *net* value is the scalar product of the weight vector and input vector minus the bias:

$$net = \mathbf{w}^T \mathbf{x} - \theta = w_1 \cdot x_1 + w_2 \cdot x_2 + \dots + w_p \cdot x_p - \theta.$$

The output value is:

$$y = f(net) = \begin{cases} 1 & \text{if } net \geq 0 \\ 0 & \text{if } net < 0 \end{cases}$$

2 Geometric interpretation

A perceptron creates a hyperplane, which divides the decision space into two parts. The weight vector is the normal vector of this hyperplane. The perceptron's output value is equal to 1 if the input vector is on the same side of the hyperplane as the weight vector and 0 otherwise. The equation of the hyperplane created by a perceptron with the weight vector (w_1, w_2, \dots, w_p) and the bias θ is:

$$w_1 x_1 + w_2 x_2 + \dots + w_p x_p - \theta = 0.$$

3 Delta rule

Weights and biases are modified according to the following rule:

$$\begin{aligned}\mathbf{w}' &= \mathbf{w} + \alpha(d - y)\mathbf{x}, \\ \theta' &= \theta - \alpha(d - y),\end{aligned}$$

where:

- \mathbf{w}' is the new weight vector,
- \mathbf{w} is the old weight vector,
- θ' is the new bias,
- θ is the old bias,
- α is the learning rate (usually between 0 and 1),
- d is the expected output,
- y is the output,
- \mathbf{x} is the input vector.

4 Perceptron training algorithm

The input is the training set consisting of input vectors \mathbf{x}_i and expected outputs d_i :

$$D = \{(\mathbf{x}_1, d_1), (\mathbf{x}_2, d_2), \dots, (\mathbf{x}_n, d_n)\}.$$

1. Pick learning rate α and generate random initial weights.
2. Repeat for a set number of iterations ('epochs') or while the iteration error exceeds a certain threshold ($E > E_{max}$)
 - (a) For each training set vector \mathbf{x}_i :
 - Calculate output y_i .
 - Update weights and bias according to delta rule.
 - (b) Calculate iteration error:

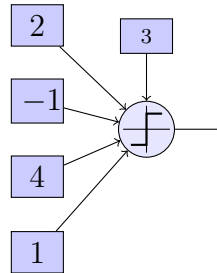
$$E = \frac{1}{n} \sum_{i=1}^n (d_i - y_i)^2.$$

Questions

Question 1.

Given the following perceptron with the weight vector $\mathbf{w} = (2, -1, 4, 1)$, bias $\theta = 3$, and a discrete activation function, calculate the output value for the following inputs.

- $\mathbf{x} = (7, -2, -5, 4)$
- $\mathbf{x} = (2, 0, 2, 8)$



Question 2.

What is the hyperplane equation of the perceptron in Question 1.

Question 3.

Design a perceptron to implement the following logical functions: AND, OR, XOR.

Question 4.

Modify the weights and bias of the perceptron from Question 1 using the delta rule for the following inputs ($\alpha = 0.5$).

- $\mathbf{x} = (7, -2, -5, 4)$, $d = 1$
- $\mathbf{x} = (5, 8, -1, -2)$, $d = 1$
- $\mathbf{x} = (0, 1, 2, 1)$, $d = 0$
- $\mathbf{x} = (2, 0, 2, 8)$, $d = 0$
- $\mathbf{x} = (3, 1, 9, -3)$, $d = 0$

Mini-project: Perceptron

The training set in `perceptron.data` contains the *Iris* dataset limited to `Iris-versicolor` and `Iris-virginica`. `perceptron.test.data` contains the test set. Implement the perceptron and train it to classify the two species. Test with the test set and output the accuracy. The program should have the following capabilities:

- Loading any dataset in csv format, where the last column is the class. The number of weights should be adjusted to the dataset. (!!)
- Picking the learning rate.
- Simple UI to manually input vectors to classify.

Hints:

- For best results pick a small learning rate (e.g. 0.01) and repeat the learning procedure for a larger number of iterations.
- Pick the initial weights and bias randomly from the range $[0, 1]$.
- `Iris-versicolor` and `Iris-virginica` are not linearly separable (the iteration error will never be 0), but you can still classify the test set with high accuracy.