11 October, 2016

Neural Networks

Course 2: The Perceptron Algorithm

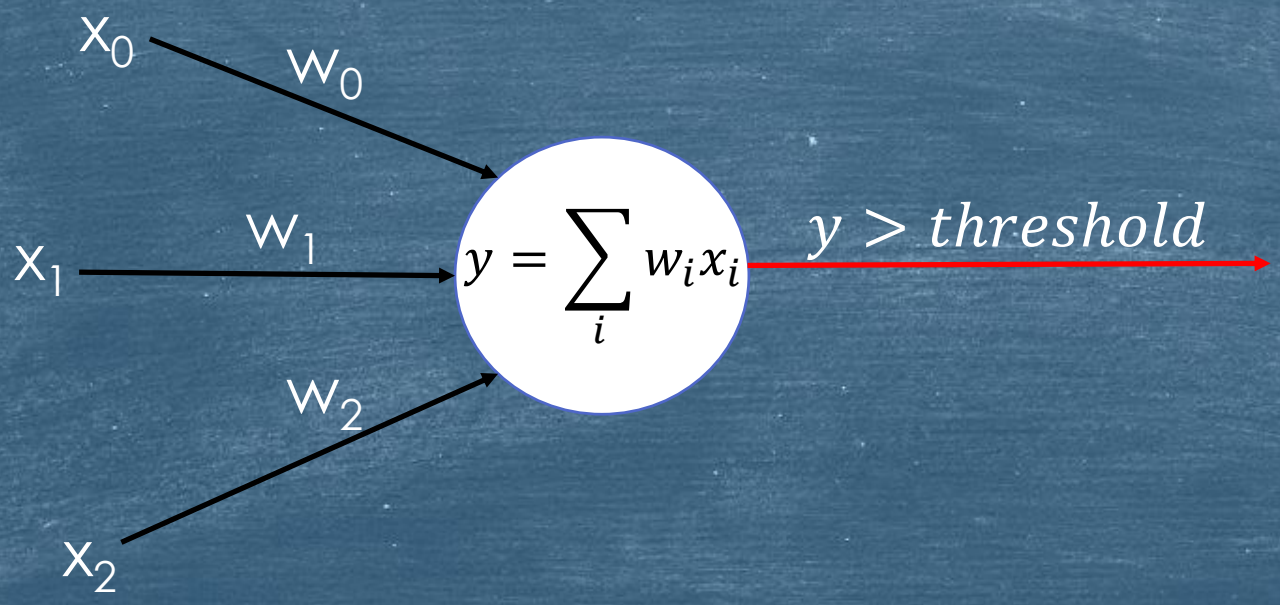
Overview

- Properties
- Online Training
- Batch Training
- Adaline Perceptron
- Conclusions

Perceptron Properties

Perceptron Properties

- Developed in 1957 by Frank Rosenblat
- Based on the model developed by Warren McCulloch and Walter Pits (1943)
- The idea is simple: a weighted sum over the inputs is compared to a threshold. If it greater than the threshold, than the neuron fires.



Perceptron Properties

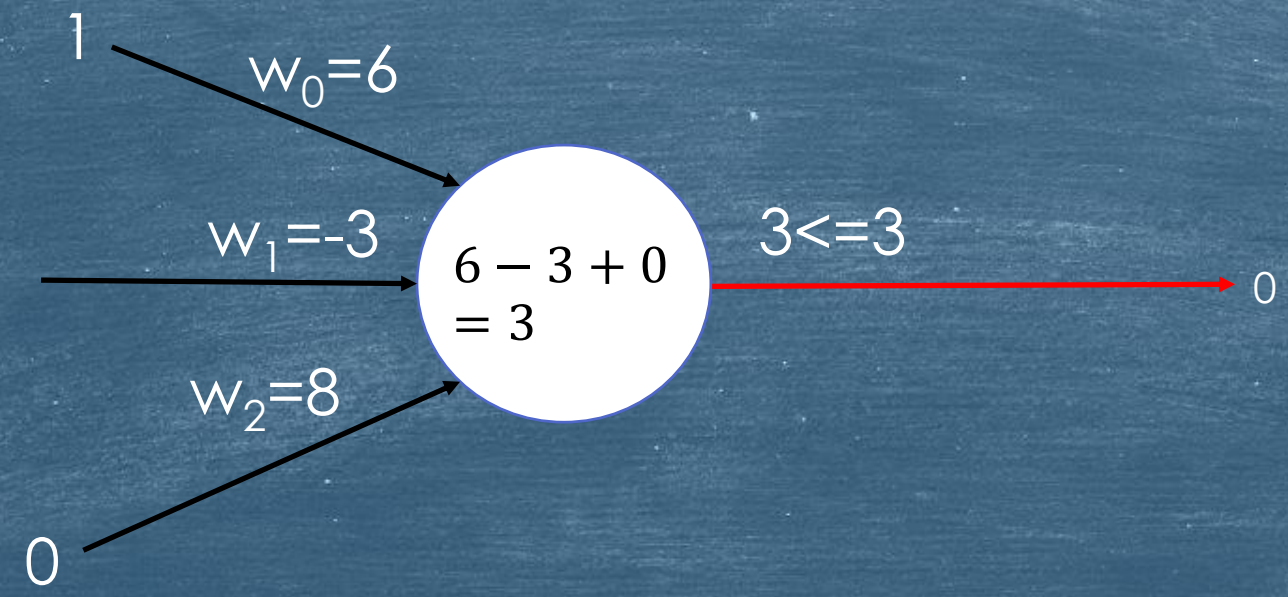
- Makes decisions by weighting up evidence

$$output = \begin{cases} 1, if \sum_{i=0}^{n} x_i w_i > threshold \\ 0, if \sum_{i=0}^{n} x_i w_i \leq threshold \end{cases}$$

- Learning is performed by varying the weights and the threshold

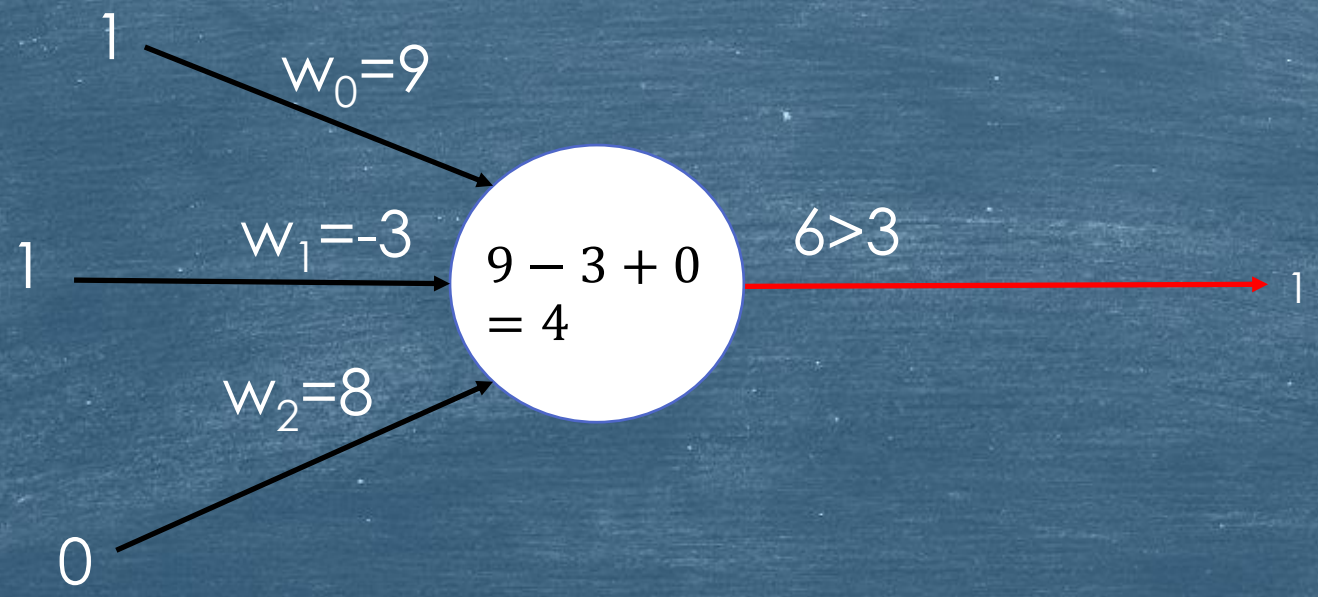
Perceptron Properties

- Example
  - Should I buy a new laptop?
    - X0: A new videogame that I like was just released
    - X1: The laptop is expensive
    - X2: The older one doesn't work anymore
    - Threshold = 3



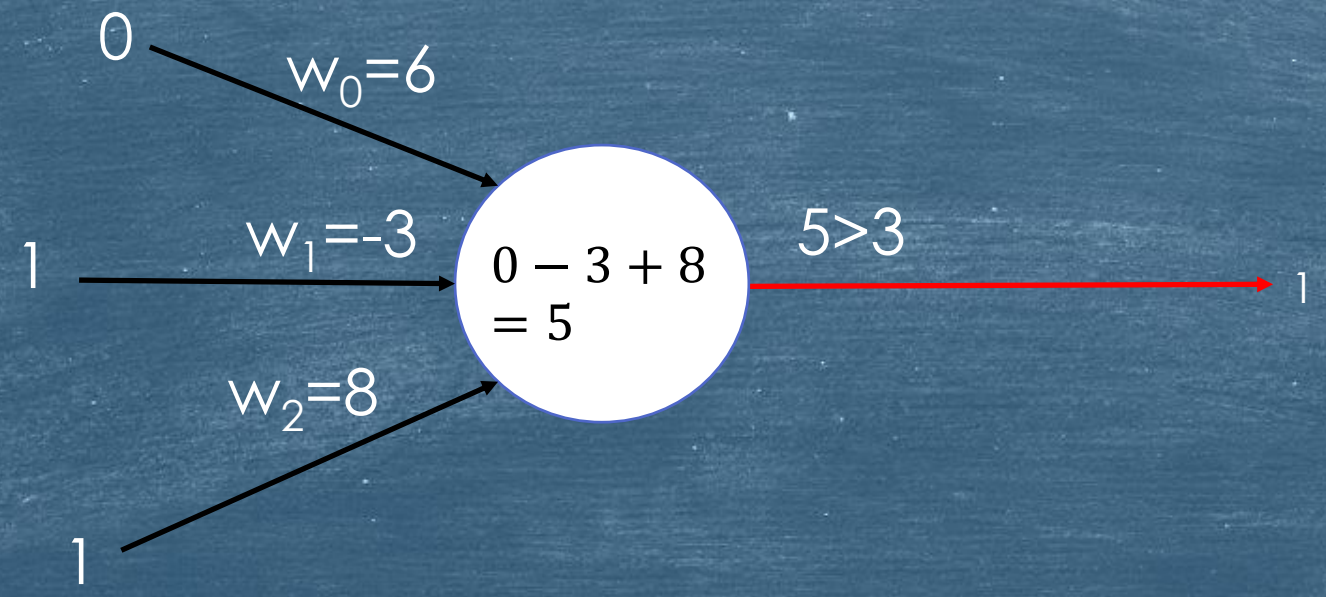
Perceptron Properties

- Example (the new game is very important)
  - Should I buy a new laptop?
    - X0: A new videogame that I like was just released
    - X1: The laptop is expensive
    - X2: The older one doesn't work anymore
    - Threshold = 3



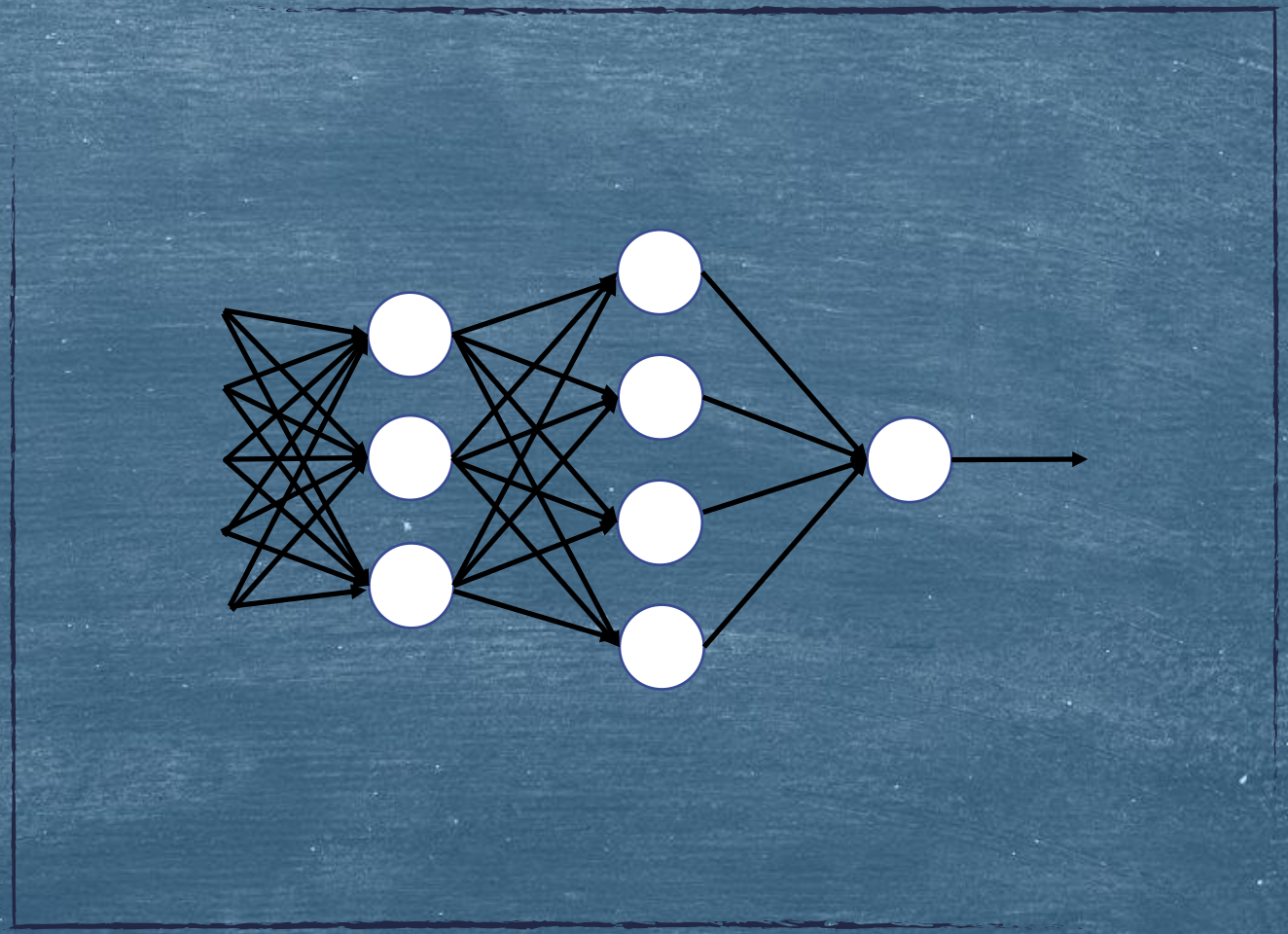
Perceptron Properties

- Example (the old laptop is broken)
  - Should I buy a new laptop?
    - X0: A new videogame that I like was just released
    - X1: The laptop is expensive
    - X2: The older one doesn't work anymore
    - Threshold = 3



Perceptron Properties

Many perceptrons can make up a network that is able to take complex decisions.



Perceptron Properties

- Easier notation

$$\sum_{i=0}^{n} x_i w_i = x \cdot w \quad \text{(Dot product)}$$

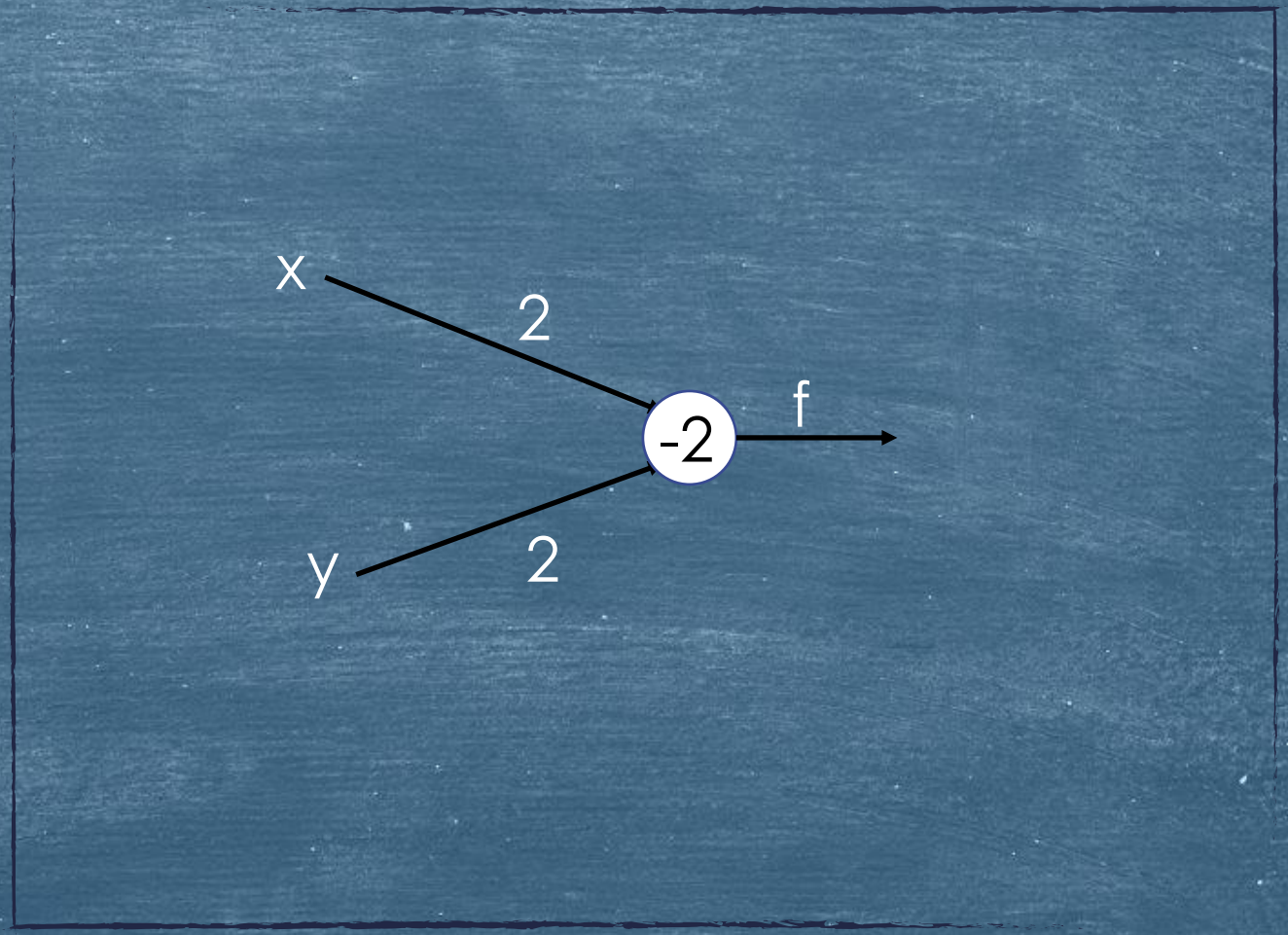
$b = -threshold$ (bias)

$$output = \begin{cases} 1, if\ w \cdot x + b > 0 \\ 0, if\ w \cdot x + b \leq 0 \end{cases}$$

Perceptron Properties

For example. AND operator

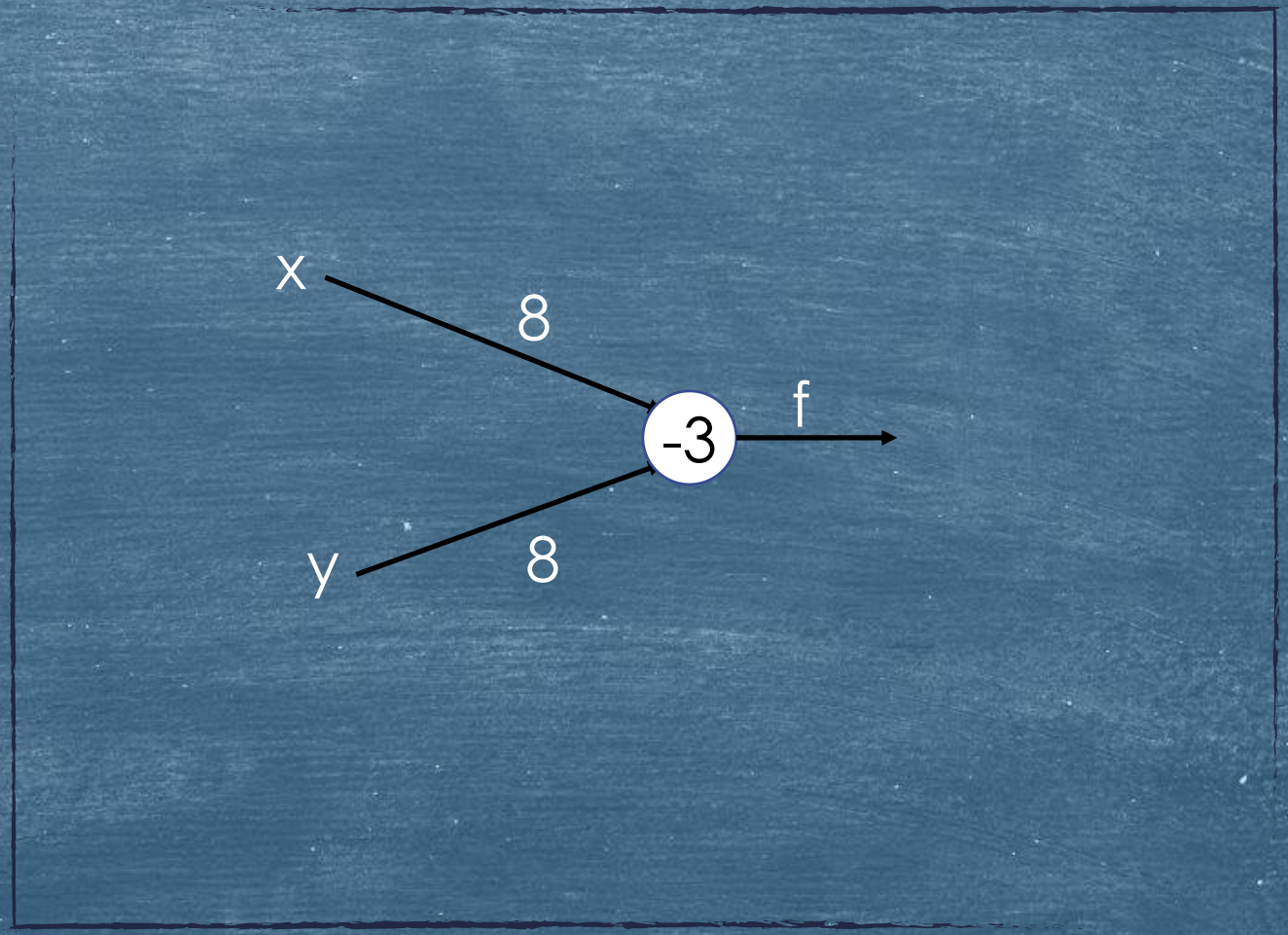
x	y	f	output
0	0	-2	0
0	1	0	0
1	0	0	0
1	1	2	1



Perceptron Properties

For example. OR operator

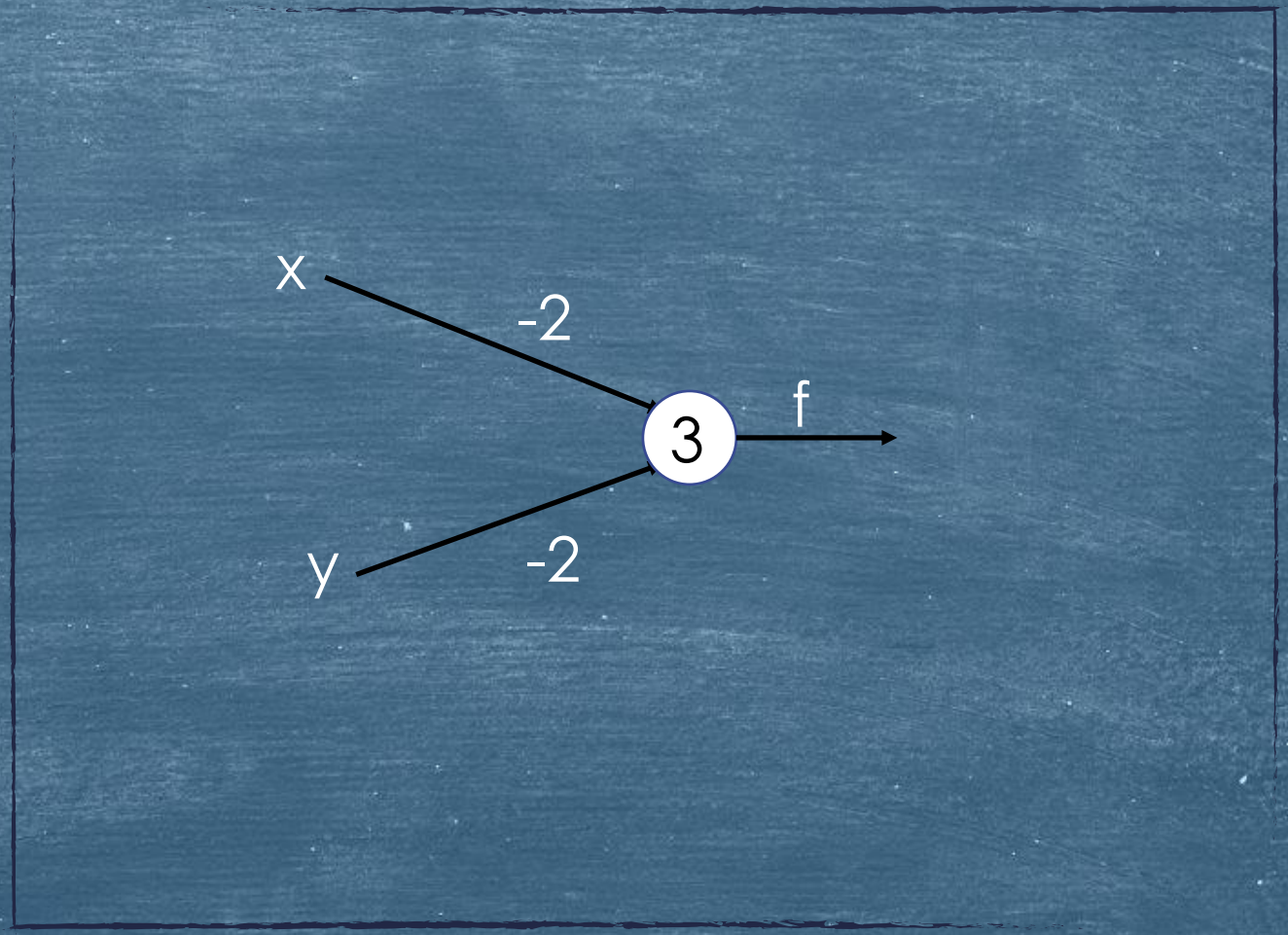
x	y	f	output
0	0	-3	0
0	1	5	1
1	0	5	1
1	1	15	1



Perceptron Properties

For example. NAND operator

x	y	f	output
0	0	3	1
0	1	1	1
1	0	1	1
1	1	-1	0



Perceptron Properties

- NAND is generator over the Boolean set. That means that every Boolean function can be obtained just by using NAND gates.

$$\bar{x} = \overline{x \cdot x}$$

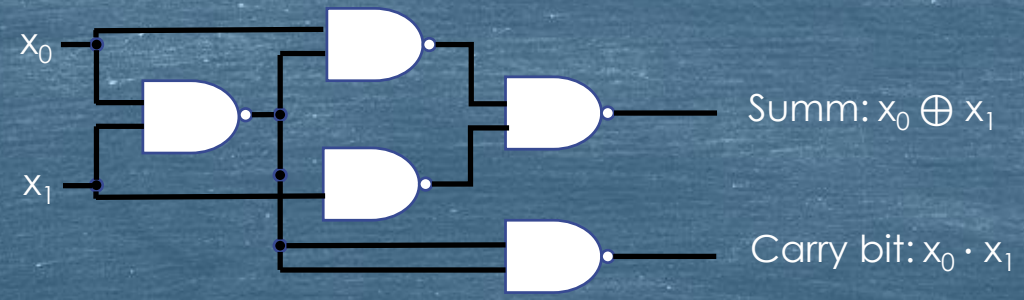
$$x \cdot y = \overline{\overline{x \cdot y} \cdot \overline{x \cdot y}}$$

$$x + y = \overline{\overline{x \cdot x} \cdot \overline{y \cdot y}}$$

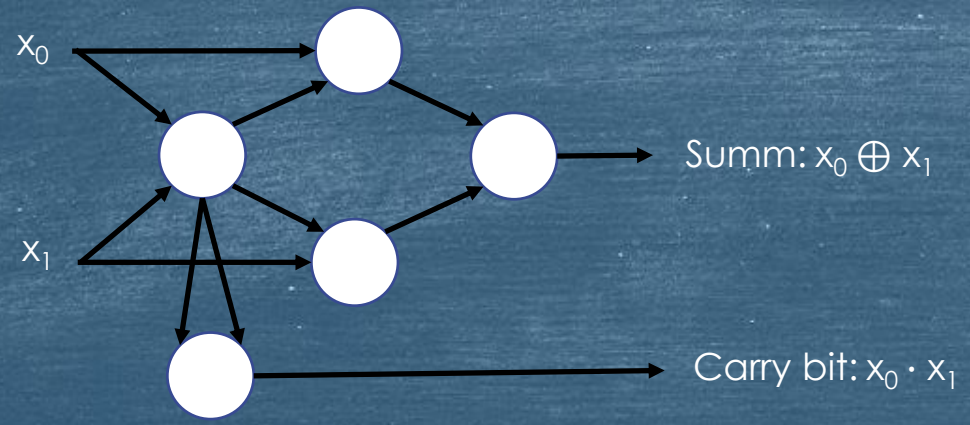
$$x \oplus \text{y} = \overline{\overline{x \cdot \overline{x \cdot y}} \cdot \overline{\overline{x \cdot y} \cdot y)}}$$

Perceptron Properties

- Implementing the sum function with perceptrons
  - Standard sum function implemented using NAND gates



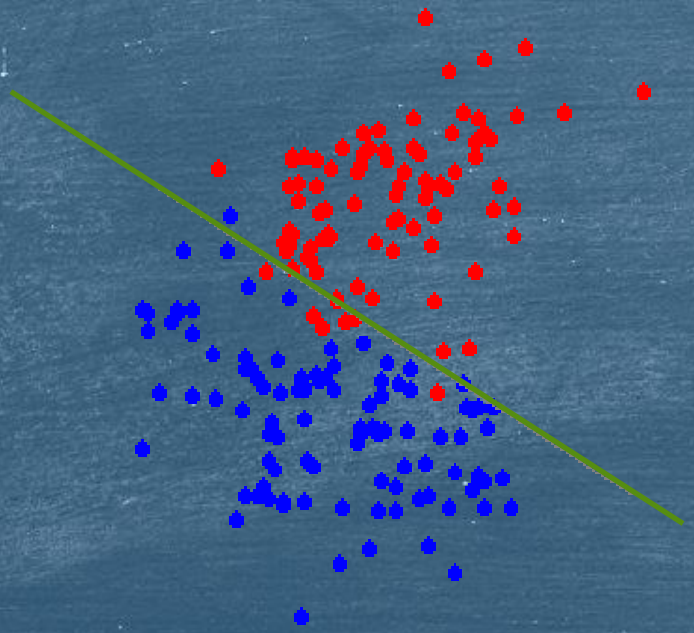
- Same summation using perceptrons

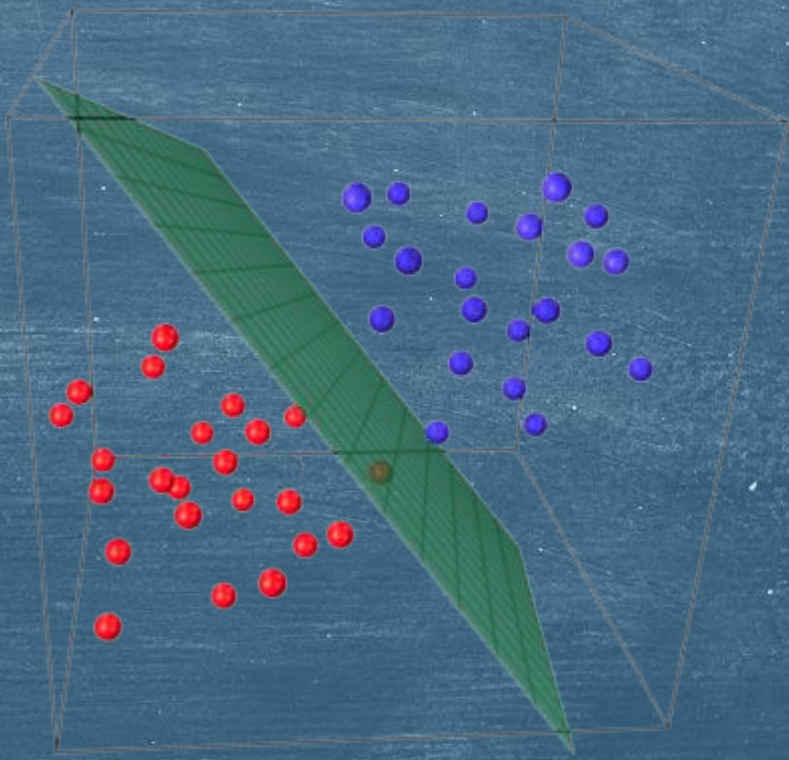


Online Training

Online Training

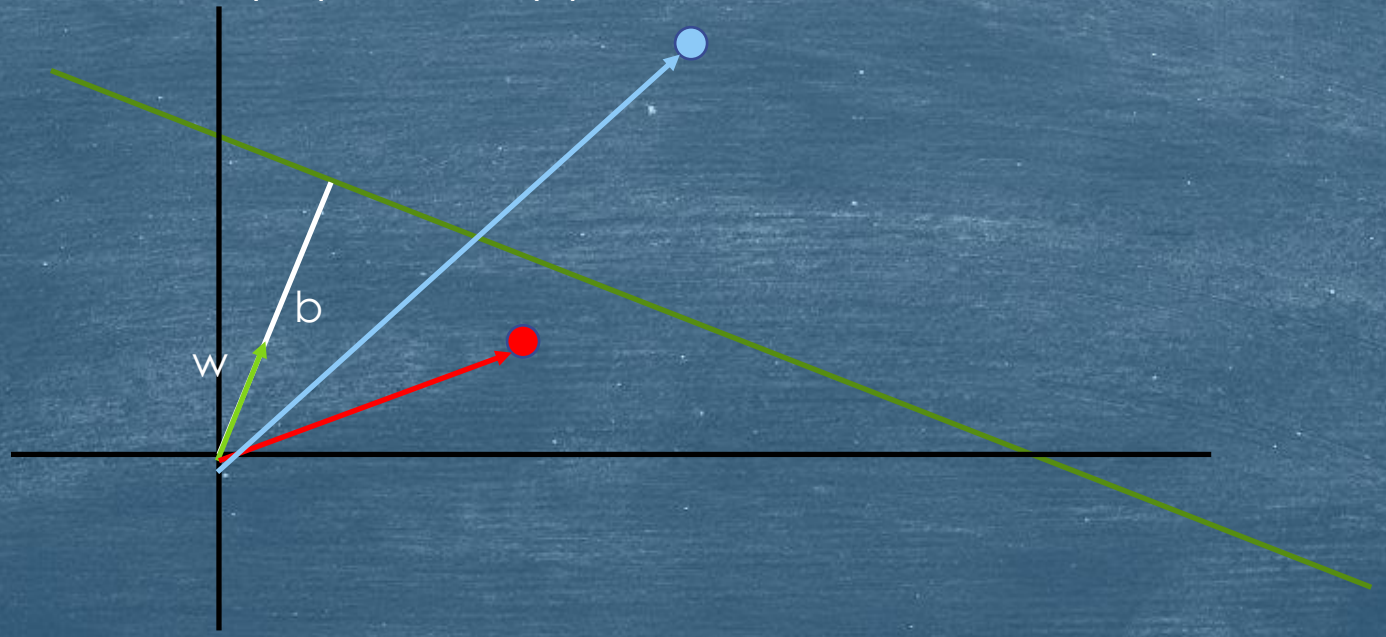
- The perceptron is also called a linear classifier since it is defined by a linear equation.
- In a 2 dimensional space, the perceptron is described as a line. In a space with 3 or more dimensions it is described as a hyperplane





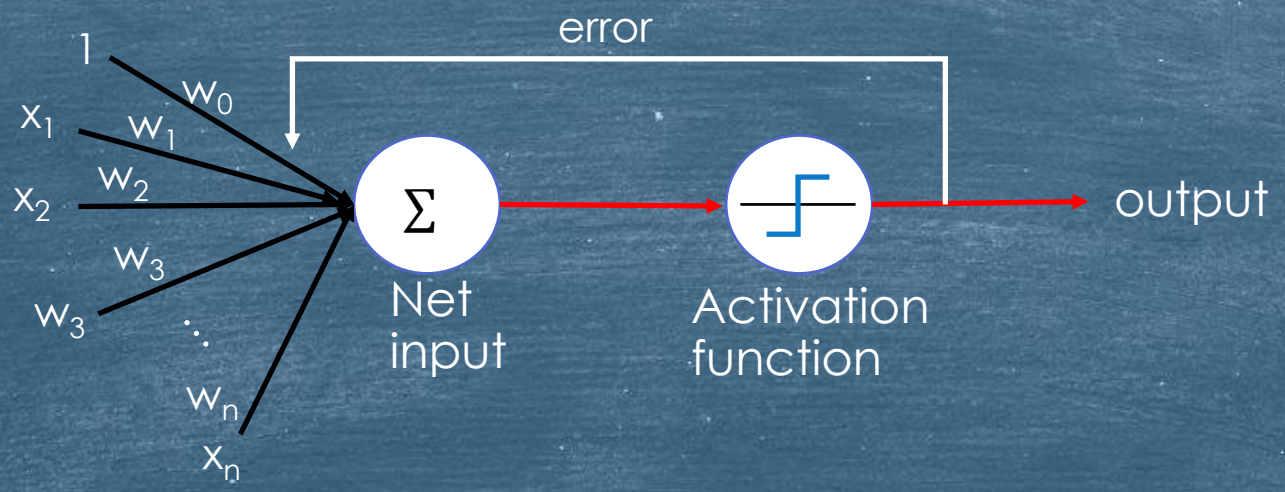
Online Training

- The coefficients (weights) define a normal vector, that we can consider a unit vector (length 1)
- The bias defines the distance from the origin to the plane
- The dot product $x \cdot w$ will be equal to |b|, for every point located on the plane
  - Will be less than |b| for every point located under the plane
  - Will be grater than |b| for every point located above b



Online Training

- The training algorithm is simple:
  1) Test if each element is correctly classified
  2) If it is correct, than do nothing
  3) If it is incorrect, adjust the hyperplane in the desired direction
  4) Repeat the steps above until stopping condition is met (all elements are correctly classified or a number of iterations have passed)



Online Training

In order to make the training smoother, a learning rate with the value less than 1 is used.

```
int activation(input)
      if (input > 0) return 1
      return 0

While(!allClassified and nrIterations >0)
      allClassified = True
      For each x,t in TrainingSet:
            z= w · x + b                          //compute net input
            output = activation(z)                //classify the sample
            w = w + (t − output)x ∗ λ             //adjust the weights
            b = b + (t − output) ∗ λ              //adjust the bias
            if output!=t
                  allClassified = False
      nrIterations-=1
```

Mini-Batch Training

Mini-Batch Training

- Same idea as online training (adjust the weights if an error is made), but this time, instead of updating the weights of the hyperplane, update an intermediary term (delta) that will later be used to update the hyperplane.

- So, delta will contain all the adjustments needed for all the elements in the batch

```
While(nrIterations >0)
    for i = 1,nr_batches
        For each x,t in batch_i:
            z = w · x + b
            y = activation(z)
            Δ_i = Δ_i + (t − y)x ∗ λ
            β_i = β_i + (t − y) ∗ λ
    for i = 1,nr_batches
        w = w + Δ_i
        b = b + β_i
    nrIterations-=1
```

Mini-Batch Training

- Mini-Batch Training vs Online Training:
  - In online training learning is performed with one element at a time, while in mini-batch multiple elements are considered on one weight adjustment
  - The order of elements in online Training is important, while in mini-batch it is not. Shuffling should be performed in Online Training
  - Mini-Batch training can be performed in parallel.
  - It is also very useful to compute the update for the weight using a mini batch instead of online learning. This is because the weights adjustments in mini-batch are performed more rarely and are more orientated towards the values that minimize the error for all the samples.

Adaline Perceptron

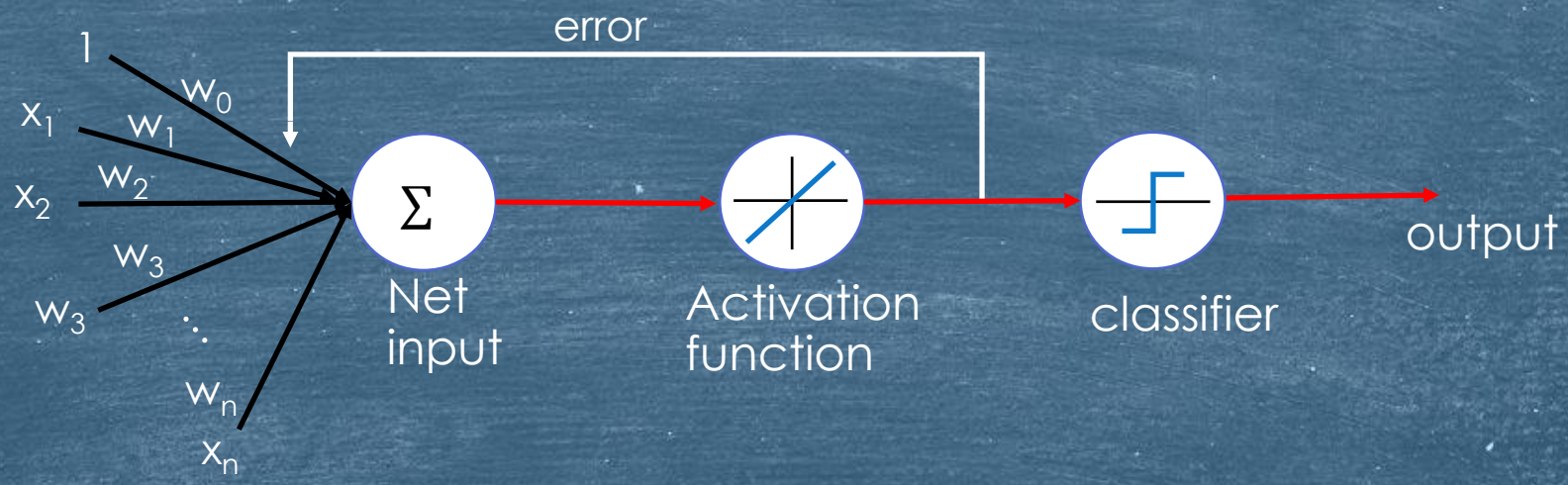
Adaline Perceptron

- The standard perceptron algorithm no longer adapts once all the elements are correctly classified. This makes the resulted hyperplane to not generalize well on the trained elements
- If the elements are not linear separable, the perceptron doesn't stop updating the weights, thus not stabilizing at a good decision boundary

Adaline Perceptron

- A slight modification of the perceptron algorithm:

  When adjusting the weights, take into account how bad a sample is misclassified (instead of just considering it misclassified)



Adaline activation model

Adaline Perceptron

The label of the samples, must be changed from {0,1} to {-1, 1} since we do not want to consider that the negative elements are located on the hyperplane.

Note: for the perceptron, the labels were only used more as a convention. Here, the labels are considered target values

Adaline Perceptron

The label of the samples, must be changed from {0,1} to {-1, 1}

The algorithm:

```
While(!allClassified and nrIterations >0)
     allClassified = True
     For each x,t in TrainingSet:
          z= w · x + b                          //compute net input
          output = activation(z)                //classify the sample
          w = w + (t − z)x ∗ λ                  //adjust the weights
          b = b + (t − z) ∗ λ                   //adjust the bias

     //other stopping condition. For example, error doesn't improve
     nrIterations-=1
```

Conclusions

- Useful for classifying linear separable items. However, most of the data is not linear separable
- It is a very fast algorithm that requires a very little amount of memory
- In certain cases other variation of the perceptron algorithm must be used, such as Adaline Perceptron
- It achieves best results when combined with other algorithms capable of creating features

Bibliography

- http://neuralnetworksanddeeplearning.com/ by Michael Nielsen
- Perceptrons: an introduction to computational geometry, by Marvin Minsky and Seymour Papert, 1969
- http://blog.sairahul.com/2014/01/linear-separability.html, by Sai Raul, 17 January 2014
- http://sebastianraschka.com/Articles/2015_singlelayer_neurons.html, by Sebastian Raschka, 24 March 2015

Questions & Discussion