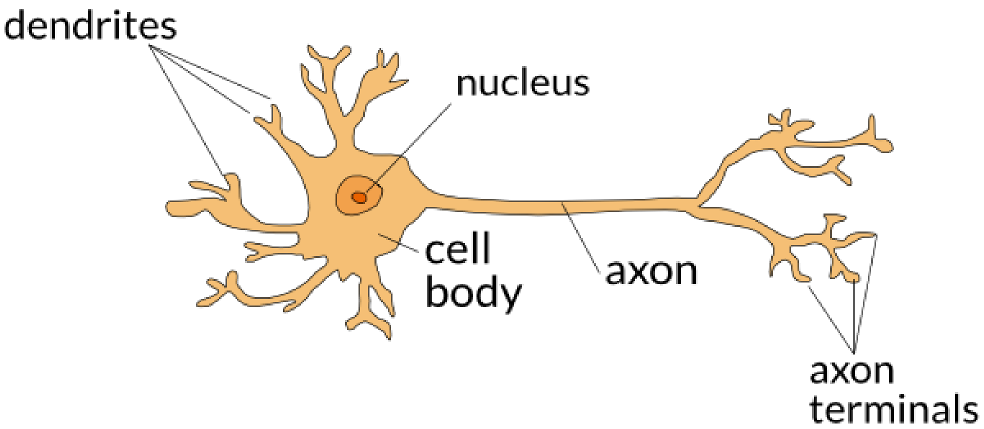
Introduction to Machine Learning

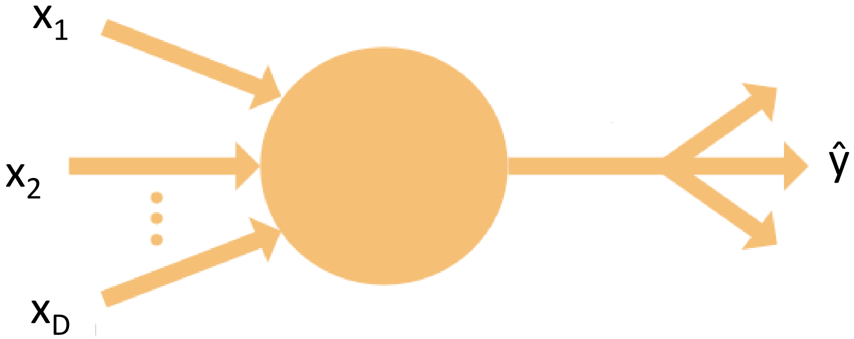
The Perceptron

Perceptron

- Supervised learning algorithm
- Regression or classification
- Allows us to weight features

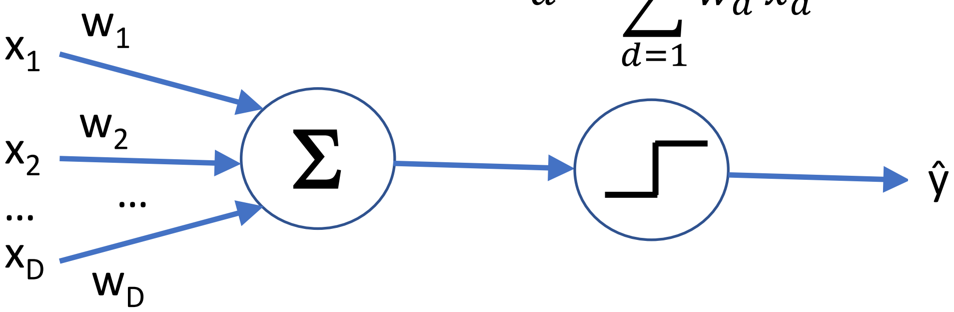
Biological inspiration





Perceptron neuron

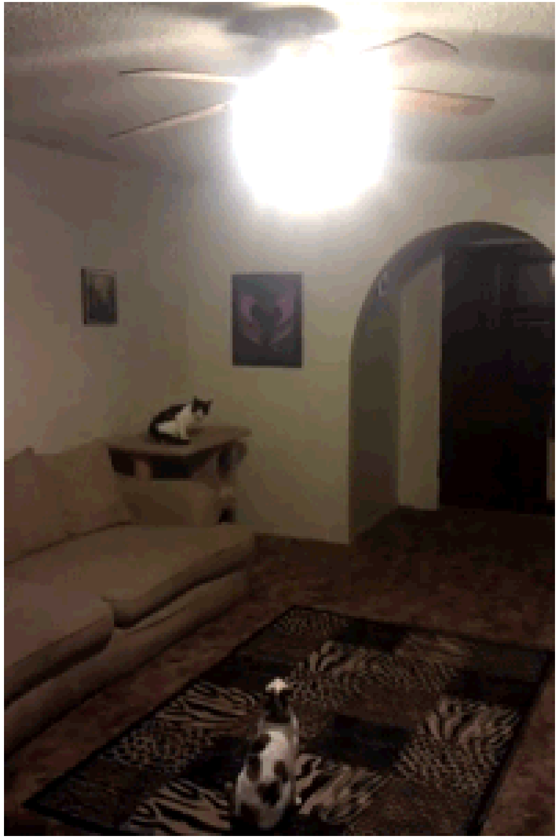
$$a = \sum_{d=1}^{D} w_d \, x_d$$



Input to neuron

- Impact of weights

$$a = \sum_{d=1}^{D} w_d \, x_d$$



Activation function

- If a > 0 then output 1 (positive example)
- Else output -1 (negative example)
- Use non-zero threshold
  - If $a > \theta$ output 1, else output -1
  - Can accomplish the same thing through bias term

$$a = \sum_{d=1}^{D} w_d \, x_d$$

$$a = \sum_{d=1}^{D} w_d \, x_d + b$$

Class labels

- Binary classifier
- Classes are + and –
- Denote by y=+1 and y=-1
- Once activation is computed, output is sign of a

$$a = \sum_{d=1}^{D} w_d \, x_d + b$$

Training a Perceptron

- Intuition
  - If output -1 but should have output +1, need to increase weights
  - If output +1 but should have output -1, need to decrease weights

$$a = \sum_{d=1}^{D} w_d \, x_d + b$$

Algorithm 5 PERCEPTRONTRAIN($\mathbf{D}$, *MaxIter*)

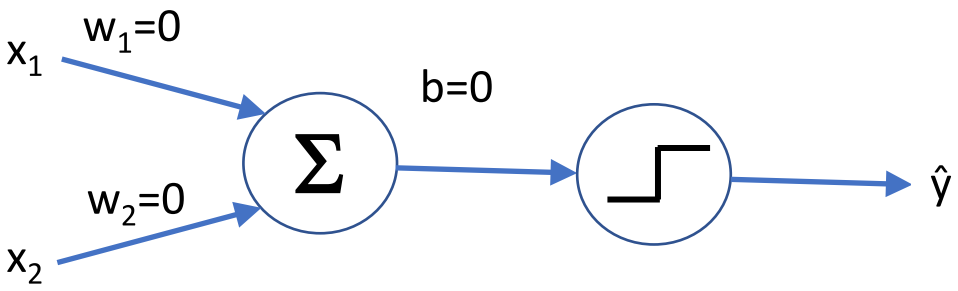
```
w_d ← 0, for all d = 1 ... D                          // initialize weights
b ← 0                                                 // initialize bias
for iter = 1 ... MaxIter do
    for all (x,y) ∈ D do
        a ← Σ_{d=1}^{D} w_d x_d + b                   // compute activation for this example
        if ya ≤ 0 then
            w_d ← w_d + y x_d, for all d = 1 ... D    // update weights
            b ← b + y                                 // update bias
        end if
    end for
end for
return w_0, w_1, ..., w_D, b
```

Algorithm 6 PERCEPTRONTEST($w_0, w_1, \dots, w_D, b, \hat{x}$)

```
a ← Σ_{d=1}^{D} w_d x̂_d + b                           // compute activation for the test example
return SIGN(a)
```

Example: Logical AND

y	x1	x2
-1	0	0
-1	0	1
-1	1	0
+1	1	1



if $ya \le 0$ **then**

$w_d \leftarrow w_d + y x_d$

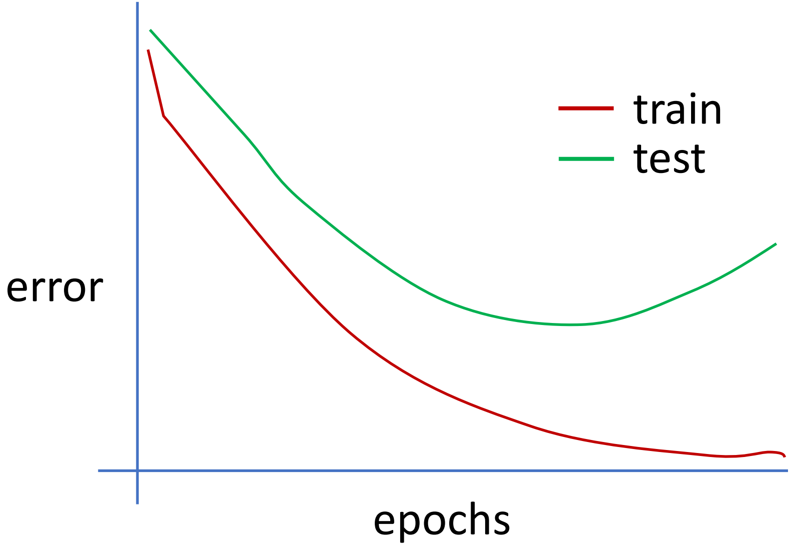
$b \leftarrow b + y$

$$
\begin{aligned}
a' &= \sum_{d=1}^{D} w'_d x_d + b' \\
&= \sum_{d=1}^{D} (w_d + x_d) x_d + (b + 1) \\
&= \sum_{d=1}^{D} w_d x_d + b + \sum_{d=1}^{D} x_d x_d + 1 \\
&= a + \sum_{d=1}^{D} x_d^2 + 1 \quad > \quad a
\end{aligned}
$$

Let's try this out

Number of iterations

- Too many, overfit
- Too little, underfit



Decision boundary

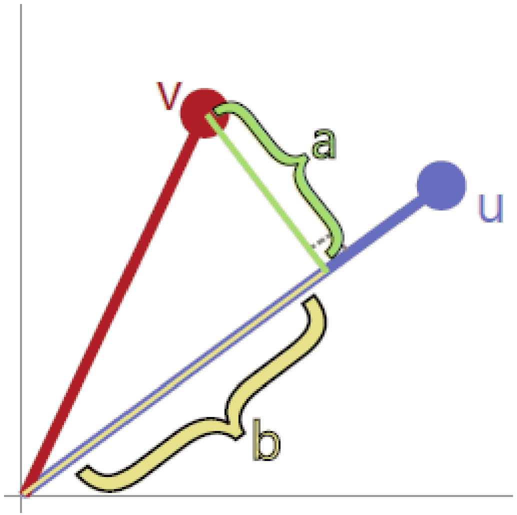
- Decision boundary is where sign(a) changes from -1 to +1

Let's try this out

Decision boundary

- Decision boundary is where sign(a) changes from -1 to +1

Dot products



Decision boundary

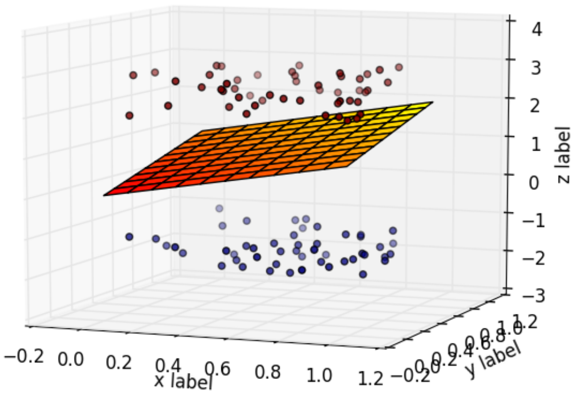
- Plane perpendicular to w

Role of bias

How to interpret weights?

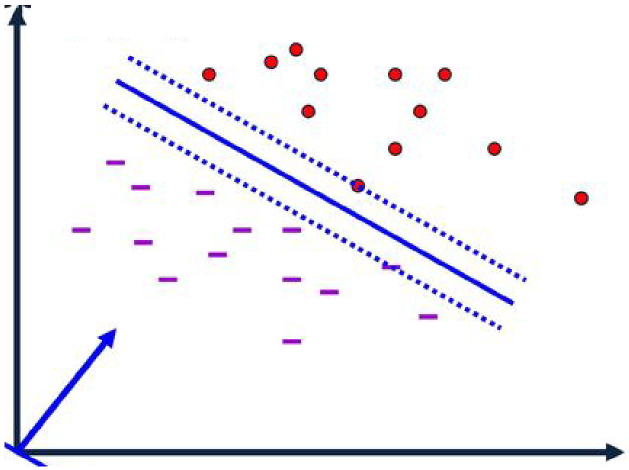
Linearly separable

- If the classes can be separated by a hyperplane, then they are linearly separable
- Perceptron can learn any linearly separable function



Margin

- Large margin -> easy
- Small margin -> hard



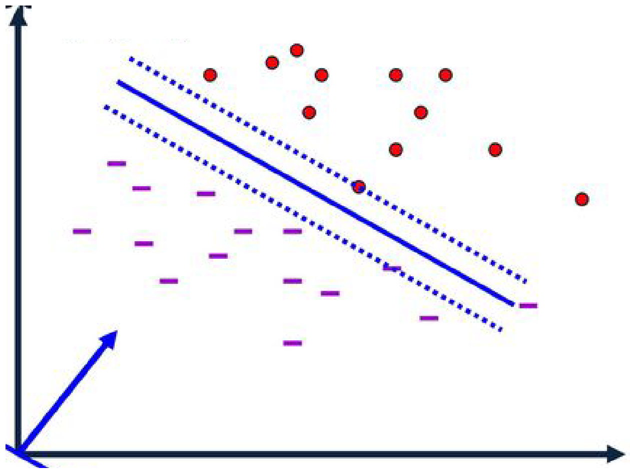
Margin

- Margin of w, b on D

$$margin(\mathbf{D}, \boldsymbol{w}, b) = \begin{cases} \min_{(x,y)\in\mathbf{D}} y(\boldsymbol{w} \cdot \boldsymbol{x} + b) & \text{if } \boldsymbol{w} \text{ separates } \mathbf{D} \\ -\infty & \text{otherwise} \end{cases}$$

- Margin of a dataset

$$margin(\mathbf{D}) = \sup_{\boldsymbol{w},b} margin(\mathbf{D}, \boldsymbol{w}, b)$$



Averaged Perceptron

- Voting
- Average

Algorithm 7 AveragedPerceptronTrain(**D**, *MaxIter*)

```
w ← ⟨0,0,...0⟩ , b ← 0                      // initialize weights and bias
u ← ⟨0,0,...0⟩ , β ← 0                      // initialize cached weights and bias
c ← 1                                        // initialize example counter to one
for iter = 1 ... MaxIter do
    for all (x,y) ∈ D do
        if y(w · x + b) ≤ 0 then
            w ← w + y x                      // update weights
            b ← b + y                        // update bias
            u ← u + y c x                    // update cached weights
            β ← β + y c                      // update cached bias
        end if
        c ← c + 1                            // increment counter regardless of update
    end for
end for
return w - (1/c) u, b - (1/c) β              // return averaged weights and bias
```

Perceptron Pros and Cons

XOR

y	x1	x2
-1	0	0
+1	0	1
+1	1	0
-1	1	1

XOR decision boundary