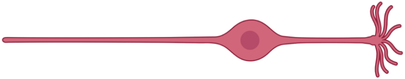
Neural Networks: Perceptrons

Spring 2021

The artificial neuron

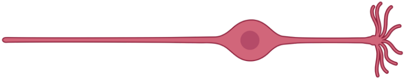
A neuron connects a series on inputs (dendrites) to an output (axon).

output: y  $\left.\begin{matrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{matrix}\right\}$ inputs

The artificial neuron

A neuron connects a series on inputs (dendrites) to an output (axon).

output: y

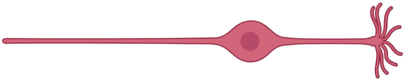


$$\left.\begin{matrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{matrix}\right\} \text{inputs}$$

Our model needs to include two processes:

1. The cell body (soma) combines all of the n inputs.
2. If the combined input exceeds a threshold, the output fires.

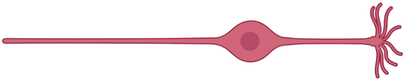
Modeling the artificial neuron

output: y  $\left.\begin{matrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{matrix}\right\}$ inputs

Let's model the combined input z as a linear combination of the inputs.

$$z = w_1x_1 + w_2x_2 + \cdots + w_nx_n = \mathbf{w} \cdot \mathbf{x}$$

Modeling the artificial neuron



Let's model the combined input z as a linear combination of the inputs.

$$z = w_1 x_1 + w_2 x_2 + \cdots + w_n x_n = \mathbf{w} \cdot \mathbf{x}$$

For now, let's assume the neuron "fires" based on the sign of z:

$$y = \begin{cases} +1, & \mathbf{w} \cdot \mathbf{x} > 0 \\ -1, & \mathbf{w} \cdot \mathbf{x} < 0 \end{cases}$$

Learning to classify

Let's assume we have n training points $(\mathbf{x}_i, y_i)$. How can we use these data to find the weights $\mathbf{w}$ so the neuron fires only when $y_i = +1$?

Learning to classify

Let's assume we have n training points $(\mathbf{x}_i, y_i)$. How can we use these data to find the weights $\mathbf{w}$ so the neuron fires only when $y_i = +1$?

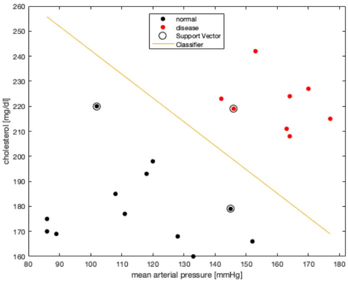
The Perceptron Update Algorithm

1. Choose a random vector of initial weights $\mathbf{w}^{(0)}$ and a step size α.
2. For each training point $(\mathbf{x}_i, y_i)$, update the weights by

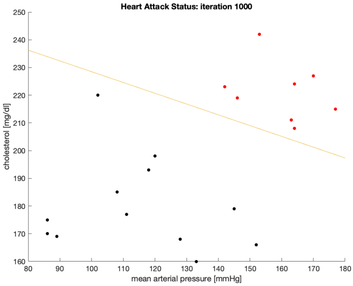
$$\mathbf{w}^{(k+1)} = \begin{cases} \mathbf{w}^{(k)} + \alpha y_i \mathbf{x}_i, & \text{if } \mathbf{x}_i \text{ is misclassified} \\ \mathbf{w}^{(k)}, & \text{otherwise} \end{cases}$$

3. Repeat until all points are classified correctly.

Does it work?

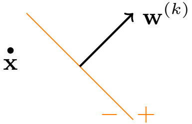


Heart attack classification by SVM.



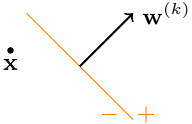
Heart attack classification by a perceptron.

Why does it work?



Point $\mathbf{x}$ is $+1$ and is therefore misclassified by $\mathbf{w}^{(k)}$.

Why does it work?



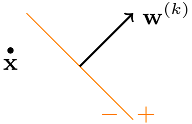
Point $\mathbf{x}$ is $+1$ and is therefore misclassified by $\mathbf{w}^{(k)}$.

The weight is updated by

$$\mathbf{w}^{(k+1)} = \mathbf{w}^{(k)} + \alpha\mathbf{x},$$

tilting $\mathbf{w}$ toward $\mathbf{x}$.

Why does it work?

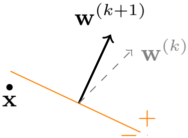


Point $\mathbf{x}$ is $+1$ and is therefore misclassified by $\mathbf{w}^{(k)}$.

The weight is updated by

$$\mathbf{w}^{(k+1)} = \mathbf{w}^{(k)} + \alpha\mathbf{x},$$

tilting $\mathbf{w}$ toward $\mathbf{x}$.



The new hyperplane is rotated toward $\mathbf{x}$ so the point will eventually be on the correct side.

Wait, what happened to the intercept?

The SVM problem used a linear classifier

$$y = \begin{cases} +1, & \mathbf{a} \cdot \mathbf{x} \geq b \\ -1, & \mathbf{a} \cdot \mathbf{x} \leq b \end{cases}$$

but our perceptron fires using the rule

$$y = \begin{cases} +1, & \mathbf{w} \cdot \mathbf{x} > 0 \\ -1, & \mathbf{w} \cdot \mathbf{x} < 0 \end{cases}$$

Does this mean the perceptron hyperplane always passes through the origin?

Wait, what happened to the intercept?

The SVM problem used a linear classifier

$$y = \begin{cases} +1, & \mathbf{a} \cdot \mathbf{x} \geq b \\ -1, & \mathbf{a} \cdot \mathbf{x} \leq b \end{cases}$$

but our perceptron fires using the rule

$$y = \begin{cases} +1, & \mathbf{w} \cdot \mathbf{x} > 0 \\ -1, & \mathbf{w} \cdot \mathbf{x} < 0 \end{cases}$$

Does this mean the perceptron hyperplane always passes through the origin?

No. We use a common ML trick to move the *bias* (intercept) into the weight vector and expand $\mathbf{x}$ with a dummy dimension containing 1.

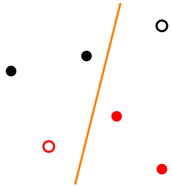
$$\mathbf{w} \cdot \mathbf{x} = b \Leftrightarrow \begin{pmatrix} w_1 \\ w_2 \\ -b \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ 1 \end{pmatrix} = 0$$

Example: Training a 2D Perceptron

x_1	x_2	y
1	0.9	$+1$
-1	0.3	$+1$
0	0.5	$+1$
1	-1	-1
-0.5	-0.7	-1
0.4	-0.3	-1

$k = 0, \quad \text{accuracy} = 4/6$

$\mathbf{w}^{(0)} = (-0.2, 0.05, 0.03)^{\mathsf{T}}$



Example: Training a 2D Perceptron

x_1	x_2	y	
1	0.9	$+1$	$\leftarrow$
-1	0.3	$+1$	
0	0.5	$+1$	
1	-1	-1	
-0.5	-0.7	-1	
0.4	-0.3	-1	

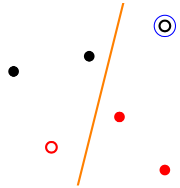
$$k = 0, \quad \text{accuracy} = 4/6$$

$$\mathbf{w}^{(0)} = (-0.2, 0.05, 0.03)^{\mathsf{T}}$$

$$\mathbf{w}^{(0)} \cdot \mathbf{x} = -0.125 \quad \text{(mismatch)}$$

$$\mathbf{w}^{(1)} = \mathbf{w}^{(0)} + 0.1(+1)\mathbf{x}$$

$$= (-0.1, 0.14, 0.13)^{\mathsf{T}}$$



Example: Training a 2D Perceptron

x_1	x_2	y	
1	0.9	$+1$	$\leftarrow$
-1	0.3	$+1$	
0	0.5	$+1$	
1	-1	-1	
-0.5	-0.7	-1	
0.4	-0.3	-1	

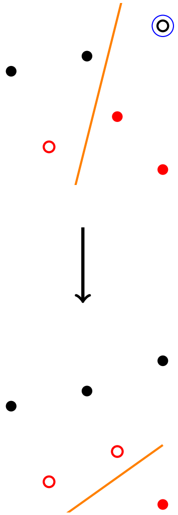
$$k = 0, \quad \text{accuracy} = 4/6$$

$$\mathbf{w}^{(0)} = (-0.2, 0.05, 0.03)^\mathsf{T}$$

$$\mathbf{w}^{(0)} \cdot \mathbf{x} = -0.125 \quad \text{(mismatch)}$$

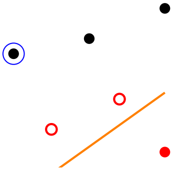
$$\mathbf{w}^{(1)} = \mathbf{w}^{(0)} + 0.1(+1)\mathbf{x}$$

$$= (-0.1, 0.14, 0.13)^\mathsf{T}$$



Example: Training a 2D Perceptron

x_1	x_2	y	
1	0.9	+1	
−1	0.3	+1	←
0	0.5	+1	
1	−1	−1	
−0.5	−0.7	−1	
0.4	−0.3	−1	



$$k = 1, \quad \text{accuracy} = 4/6$$

$$\mathbf{w}^{(1)} = (-0.1, 0.14, 0.13)^{\mathsf{T}}$$

$$\mathbf{w}^{(1)} \cdot \mathbf{x} = 0.272 \quad \text{(match)}$$

$$\mathbf{w}^{(2)} = \mathbf{w}^{(1)}$$

Example: Training a 2D Perceptron

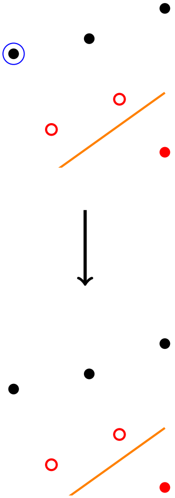
x_1	x_2	y	
1	0.9	$+1$	
-1	0.3	$+1$	$\leftarrow$
0	0.5	$+1$	
1	-1	-1	
-0.5	-0.7	-1	
0.4	-0.3	-1	

$$k = 1, \quad \text{accuracy} = 4/6$$

$$\mathbf{w}^{(1)} = (-0.1, 0.14, 0.13)^{\mathsf{T}}$$

$$\mathbf{w}^{(1)} \cdot \mathbf{x} = 0.272 \quad \text{(match)}$$

$$\mathbf{w}^{(2)} = \mathbf{w}^{(1)}$$



Example: Training a 2D Perceptron

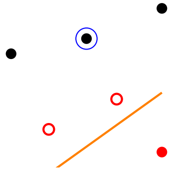
x_1	x_2	y	
1	0.9	$+1$	
-1	0.3	$+1$	
0	0.5	$+1$	$\leftarrow$
1	-1	-1	
-0.5	-0.7	-1	
0.4	-0.3	-1	

$$k = 2, \quad \text{accuracy} = 4/6$$

$$\mathbf{w}^{(2)} = (-0.1, 0.14, 0.13)^{\mathsf{T}}$$

$$\mathbf{w}^{(2)} \cdot \mathbf{x} = 0.2 \quad \text{(match)}$$

$$\mathbf{w}^{(3)} = \mathbf{w}^{(2)}$$



Example: Training a 2D Perceptron

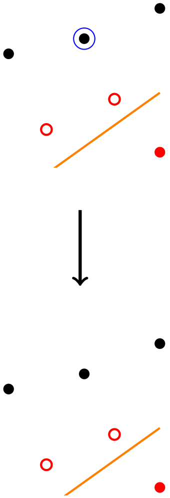
x_1	x_2	y	
1	0.9	$+1$	
-1	0.3	$+1$	
0	0.5	$+1$	$\leftarrow$
1	-1	-1	
-0.5	-0.7	-1	
0.4	-0.3	-1	

$$k = 2, \quad \text{accuracy} = 4/6$$

$$\mathbf{w}^{(2)} = (-0.1, 0.14, 0.13)^\mathsf{T}$$

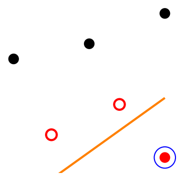
$$\mathbf{w}^{(2)} \cdot \mathbf{x} = 0.2 \quad \text{(match)}$$

$$\mathbf{w}^{(3)} = \mathbf{w}^{(2)}$$



Example: Training a 2D Perceptron

x_1	x_2	y	
1	0.9	$+1$	
-1	0.3	$+1$	
0	0.5	$+1$	
1	-1	-1	$\leftarrow$
-0.5	-0.7	-1	
0.4	-0.3	-1	



$$k = 3, \quad \text{accuracy} = 4/6$$

$$\mathbf{w}^{(3)} = (-0.1, 0.14, 0.13)^\mathsf{T}$$

$$\mathbf{w}^{(3)} \cdot \mathbf{x} = -0.11 \quad \text{(match)}$$

$$\mathbf{w}^{(4)} = \mathbf{w}^{(3)}$$

Example: Training a 2D Perceptron

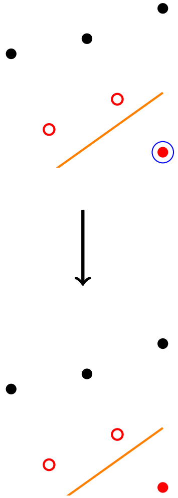
x_1	x_2	y	
1	0.9	$+1$	
-1	0.3	$+1$	
0	0.5	$+1$	
1	-1	-1	$\leftarrow$
-0.5	-0.7	-1	
0.4	-0.3	-1	

$$k = 3, \quad \text{accuracy} = 4/6$$

$$\mathbf{w}^{(3)} = (-0.1, 0.14, 0.13)^{\mathsf{T}}$$

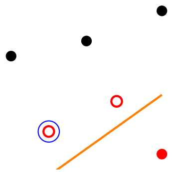
$$\mathbf{w}^{(3)} \cdot \mathbf{x} = -0.11 \quad \text{(match)}$$

$$\mathbf{w}^{(4)} = \mathbf{w}^{(3)}$$



Example: Training a 2D Perceptron

x_1	x_2	y	
1	0.9	$+1$	
-1	0.3	$+1$	
0	0.5	$+1$	
1	-1	-1	
-0.5	-0.7	-1	$\leftarrow$
0.4	-0.3	-1	



$$k = 4, \quad \text{accuracy} = 4/6$$

$$\mathbf{w}^{(4)} = (-0.1, 0.14, 0.13)^{\mathsf{T}}$$

$$\mathbf{w}^{(4)} \cdot \mathbf{x} = 0.082 \quad \text{(mismatch)}$$

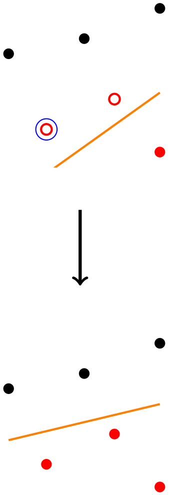
$$\mathbf{w}^{(5)} = \mathbf{w}^{(4)} + 0.1(-1)\mathbf{x}$$

$$= (-0.05, 0.21, 0.03)^{\mathsf{T}}$$

Example: Training a 2D Perceptron

x_1	x_2	y	
1	0.9	$+1$	
-1	0.3	$+1$	
0	0.5	$+1$	
1	-1	-1	
-0.5	-0.7	-1	$\leftarrow$
0.4	-0.3	-1	

$$
\begin{aligned}
k &= 4, \quad \text{accuracy} = 4/6 \\
\mathbf{w}^{(4)} &= (-0.1, 0.14, 0.13)^{\mathsf{T}} \\
\mathbf{w}^{(4)} \cdot \mathbf{x} &= 0.082 \quad \text{(mismatch)} \\
\mathbf{w}^{(5)} &= \mathbf{w}^{(4)} + 0.1(-1)\mathbf{x} \\
&= (-0.05, 0.21, 0.03)^{\mathsf{T}}
\end{aligned}
$$



Example: Training a 2D Perceptron

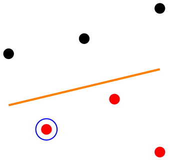
x_1	x_2	y	
1	0.9	$+1$	
-1	0.3	$+1$	
0	0.5	$+1$	
1	-1	-1	
-0.5	-0.7	-1	
0.4	-0.3	-1	$\leftarrow$

$$k = 5, \quad \text{accuracy} = 6/6$$

$$\mathbf{w}^{(5)} = (-0.05, 0.21, 0.03)^{\mathsf{T}}$$

$$\mathbf{w}^{(5)} \cdot \mathbf{x} = -0.05 \quad \text{(match)}$$

$$\mathbf{w}^{(6)} = \mathbf{w}^{(5)}$$



Example: Training a 2D Perceptron

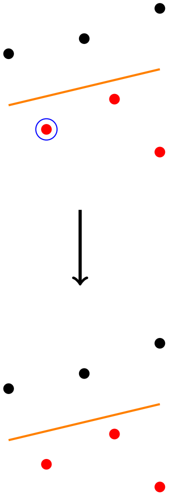
x_1	x_2	y	
1	0.9	$+1$	
-1	0.3	$+1$	
0	0.5	$+1$	
1	-1	-1	
-0.5	-0.7	-1	
0.4	-0.3	-1	$\leftarrow$

$$k = 5, \quad \text{accuracy} = 6/6$$

$$\mathbf{w}^{(5)} = (-0.05, 0.21, 0.03)^\mathsf{T}$$

$$\mathbf{w}^{(5)} \cdot \mathbf{x} = -0.05 \quad \text{(match)}$$

$$\mathbf{w}^{(6)} = \mathbf{w}^{(5)}$$



Summary

- A perceptron is a simplistic model of a single neuron.
- A perceptron can learn to perform simple classification tasks using an update rule.

Summary

- A perceptron is a simplistic model of a single neuron.
- A perceptron can learn to perform simple classification tasks using an update rule.
- **Next time:** Imagine what a network of millions of perceptrons can learn!