Introduction to single-layer perceptron

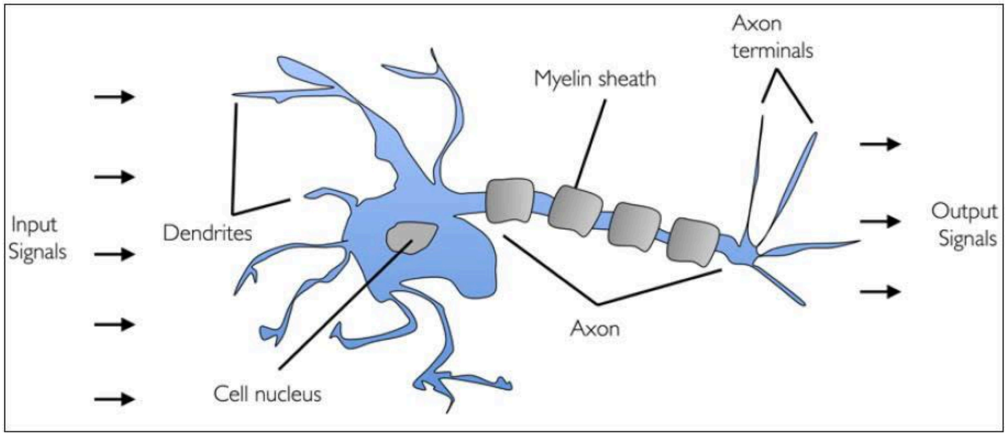
Yulia Newton, Ph.D.
CS156, Introduction to Artificial Intelligence
San Jose State University
Spring 2021

What are neural networks?

- Artificial Neural Networks (ANN) - computational predictive model inspired by neuronal structure of the mammalian cerebral cortex (brain)
- "...a computing system made up of a number of simple, highly interconnected processing elements, which process information by their dynamic state response to external inputs" - Dr. Robert Hecht-Nielsen
- In data science we are not concerned about recreating the biological structure of the brain, ANNs are simply inspired by the brain neurons
  - Typical mammalian brain has billions of neurons while a big neural network model might have hundreds thousands units

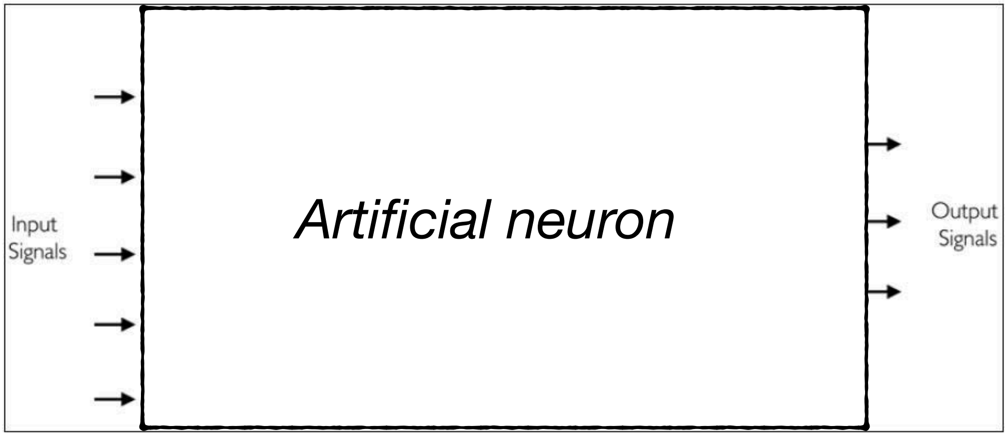
Motivation for neural networks

- Biological neurons



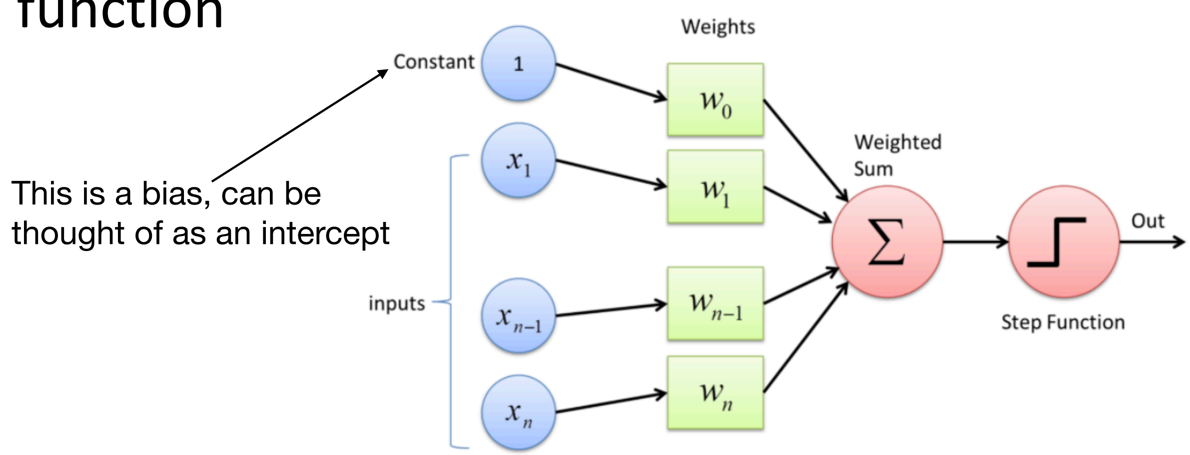
www.simplilearn.com

Can we imitate biological neurons?



Perceptron model

- Perceptron - single layer neural network
- Perceptron model is the simplest form of ANN
- Binary classifier
- Perceptrons consist of: input values, weights + bias, net sum, activation function



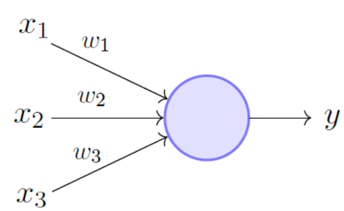
https://towardsdatascience.com

Terms perceptron vs. neural network

- “Perceptron” - single-layer ANN
- “Neural network” - multi-layer ANN
  - Also referred to as multi-layer perceptron
  - As well as feedforward neural network

Brief history of perceptron algorithm

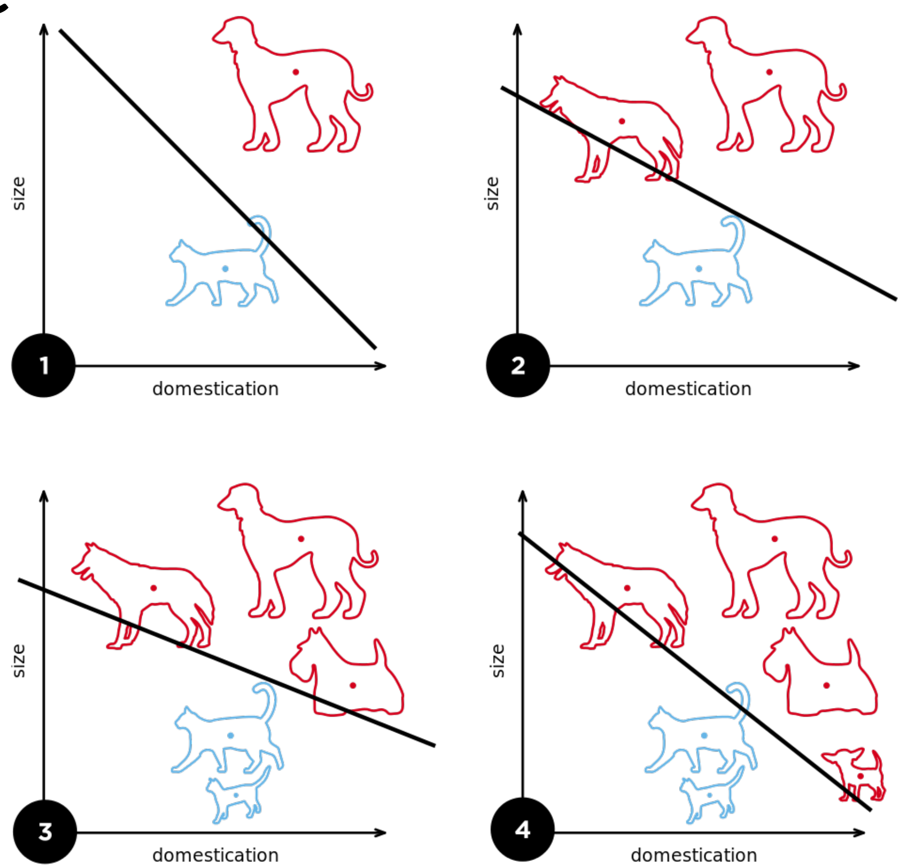
- Invented by Frank Rosenblatt at the Cornell Aeronautical Laboratory in 1958
  - Perceptron was initially a machine, not a program
  - Was intended to perform image recognition task
  - Its popularity diminished because of its limitations, until the rise of multi-layer perceptrons
- Only capable of separating linearly separable patterns
  - Linear classifier
- If you understand how a perceptron works, you will understand how neural networks work



Perceptron Model (Minsky-Papert in 1969)

https://towardsdatascience.com

We use perceptron when our data is linearly separable



https://en.wikipedia.org

Steps of perceptron classification

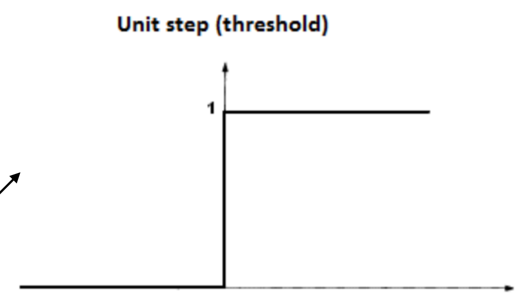
- Perceptron algorithm:
  - All input values are multiplied by the corresponding weights
  - Add up all the results from the multiplication step (we call it "weighted sum")
  - Use the weighted sum as the input into the activation function
    - Remember activation functions from the regression lecture?
      - Activation function transforms input into a different space

Activation function for a perceptron

- For linear perceptron we use an activation function called "unit step activation"

$$f(\mathbf{x}) = \begin{cases} 1 & \text{if } \mathbf{w} \cdot \mathbf{x} + b > 0, \\ 0 & \text{otherwise} \end{cases}$$

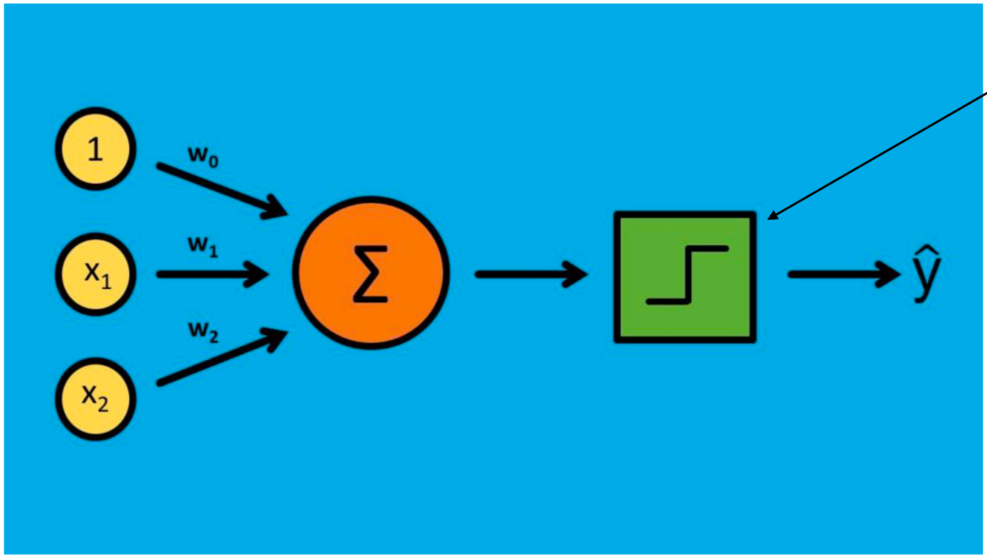
https://en.wikipedia.org



Maps values to 0 or 1

https://towardsdatascience.com

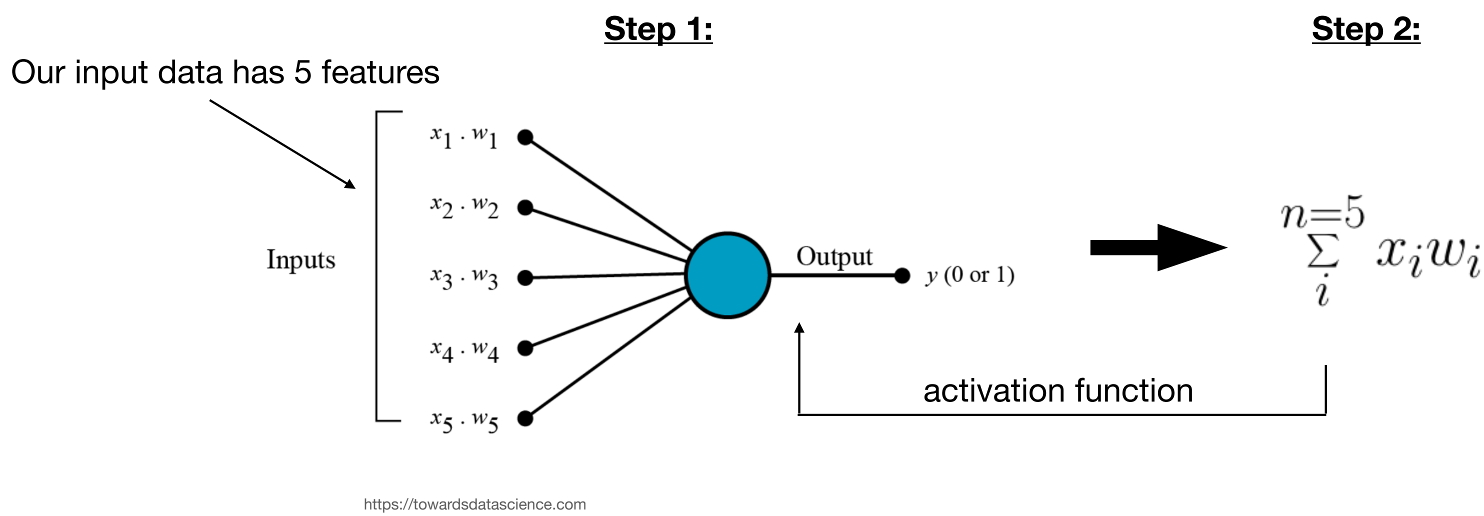
Activation function for a perceptron (cont'd)



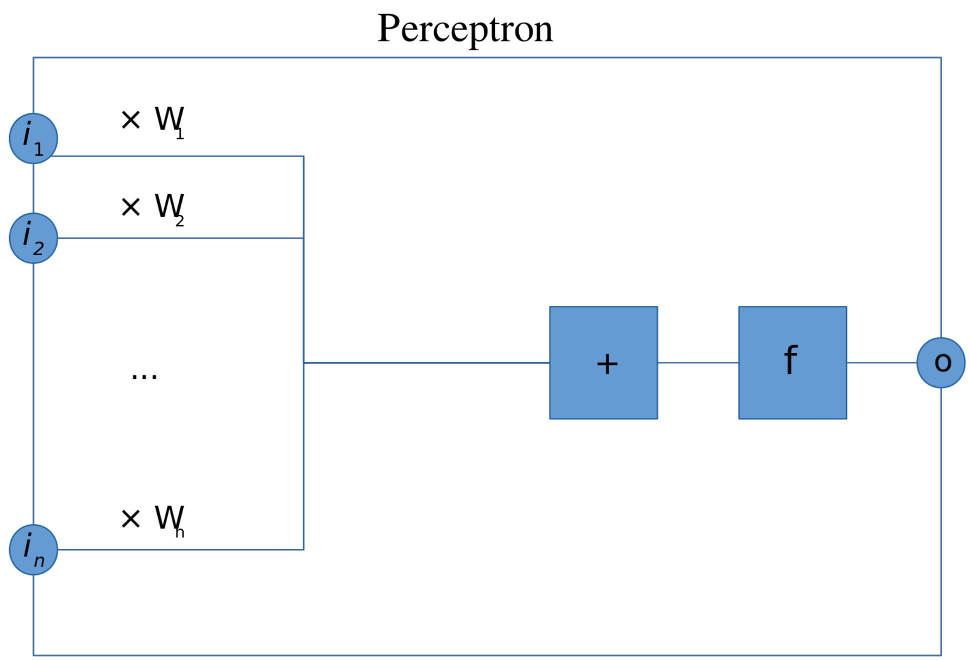
https://towardsdatascience.com

$$f(\mathbf{x}) = \begin{cases} 1 & \text{if } \mathbf{w} \cdot \mathbf{x} + b > 0, \\ 0 & \text{otherwise} \end{cases}$$

Let’s visualize the steps for the perceptron algorithm



Putting it all together



$$o = f\left(\sum_{k=1}^{n} i_k \cdot W_k\right)$$

Sometimes you will see this equation written with a bias and sometimes without a bias, but bias is a part of the perceptron model

https://en.wikipedia.org

Perceptron classification in matrix form

- Weighted sum is just a dot product of input values and weights

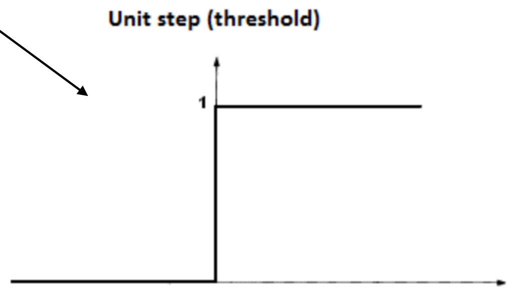
$$w^T x = w_0 + w_1 x_1 + w_2 x_2 + \ldots + w_n x_n$$

$$\Downarrow$$

$$f(\mathbf{x}) = \begin{cases} 1 & \text{if } \mathbf{w} \cdot \mathbf{x} + b > 0, \\ 0 & \text{otherwise} \end{cases}$$

Interpretation of weights and bias in ANN

- Weights show the strength and direction of each node's contribution
- Bias gives a shift to the activation function (intercept)
  - Where the function cuts the x-axis



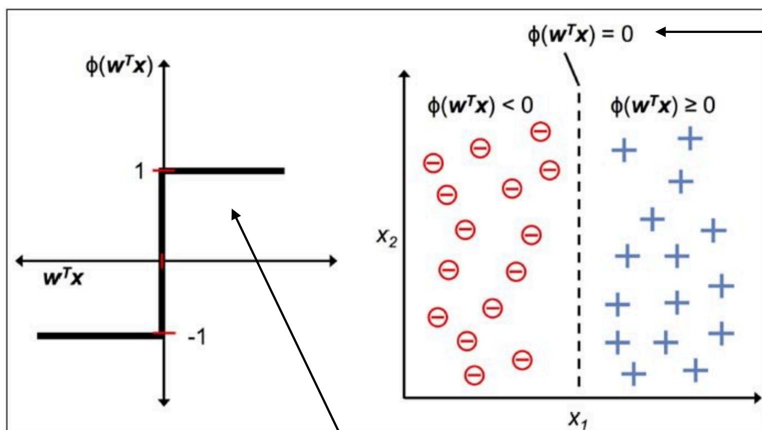
https://towardsdatascience.com

Why do we need an activation function?

- Activation function maps the output to [0,1] space
- Remember that the input values and weights are unbounded numeric values, potentially weighted sum value can belong to (-inf, +inf) interval
- Activation function allows us to have a reliable output space

How do we learn perceptron weights?

- Perceptron learning rule
  - Computed output is compared to the true output
  - The error (difference between predicted and true outputs) is propagated back and weights are adjusted



Perceptron decision function (defines the decision hyperplane)

simplilearn.com

We say “the neuron fired” if the result is in this interval

Training a perceptron model

- Steps:
  - Initialize the weights and the bias
  - Compute the classification result (see slide 9 “Steps of perceptron classification”)
  - Compare the prediction results to the true labels
  - Update the weights
  - Repeat until the convergence and predefined number of iterations

Updating perceptron weights

- “Eta” is a learning rate of the update step
- We update the weights by the difference between the predicted and true labels weighted by input value and eta parameter

$$w^{n+1} = w_i^n + \eta(y_i - \hat{y}_i x_i)$$

eta - learning rate
y_i - true label for observation i
hat(y_i) - predicted label (output of perceptron) for observation i
x_i - input values for misclassified observation

Let's go step by step

weight vector

$$\begin{bmatrix} w_0 \\ w_1 \\ w_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

features

$$\begin{array}{c} x_0 \quad\; x_1 \quad\; x_2 \\ \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix} \end{array}$$

$$\Sigma w_i \cdot x_i = (w_0 \cdot x_0) + (w_1 \cdot x_1) + (w_2 \cdot x_2)$$

$$\Sigma w_i \cdot x_i = (0 \cdot 0) + (0 \cdot 1) + (0 \cdot 0)$$

$$f = 0$$

threshold

$z = 0$

if $f > z$, $\hat{y} = 1$

otherwise, $\hat{y} = 0$

$f = 0$

$\hat{y}_0 = 0$

our estimate

$\hat{y}_0 = 0$

$y_0 = 1$

actual output

outputs

$$y_i = \begin{bmatrix} y_0 \\ y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} \boxed{1} \\ 1 \\ 0 \end{bmatrix}$$

https://towardsdatascience.com

Let’s go step by step (cont’d)

current weights

$n=1$

$$\begin{bmatrix} w_0 \\ w_1 \\ w_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

features

$$\begin{array}{ccc} x_0 & x_1 & x_2 \\ \end{array}$$
$$\begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}$$

estimate $\hat{y} = 0$

actual $y = 1$

learning rate $\eta = 0.1$

$$w_0^2 = 0 + 0.1(1 - 0)0 = 0$$
$$w_1^2 = 0 + 0.1(1 - 0)1 = 0.1$$
$$w_2^2 = 0 + 0.1(1 - 0)0 = 0$$

new weights

$n=2$

$$\begin{bmatrix} w_0 \\ w_1 \\ w_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0.1 \\ 0 \end{bmatrix}$$

learning rate (“eta”)

$$w_i^{n+1} = w_i^n + \eta(y_i - \hat{y}_i)x_i$$

new weight current weight

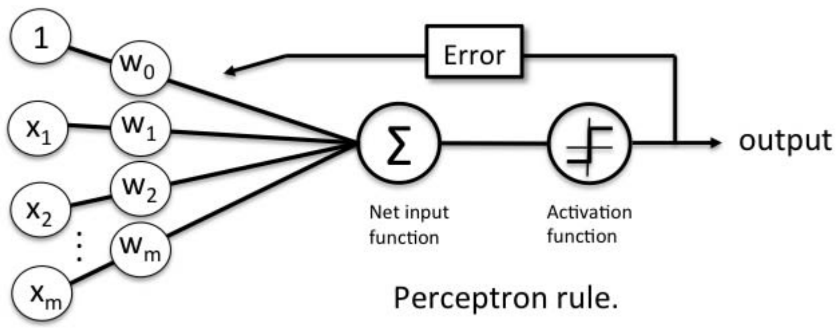
$\eta = 0.1$

https://towardsdatascience.com

Iterate until optimal weights are learned

- We continue through every observation in the dataset
- Then, continue to iterate over the dataset until
  - Convergence to acceptable prediction accuracy
  - Predetermined number of iterations is reached
  - Other termination criteria can be used
- Epoch - complete sweep through the dataset
  - In the previous toy example it takes 3 iterations to complete 1 epoch
- Number of iterations or epochs is a hyperparameter you can tune
- Perceptron algorithm will converge for a linearly separable data

Perceptron training overview



simplilearn.com

Let’s look at some code

- Classifying breast cancer using perceptron
  - *Perceptron.Breast.ipynb*

- Also, this is a great explanation of how to implement logic gates with perceptron (we will not be going over these in class)
  - https://towardsdatascience.com/perceptrons-logical-functions-and-the-xor-problem-37ca5025790a