Cheatsheets / **Learn the Basics of Machine Learning**

codecademy

Perceptron

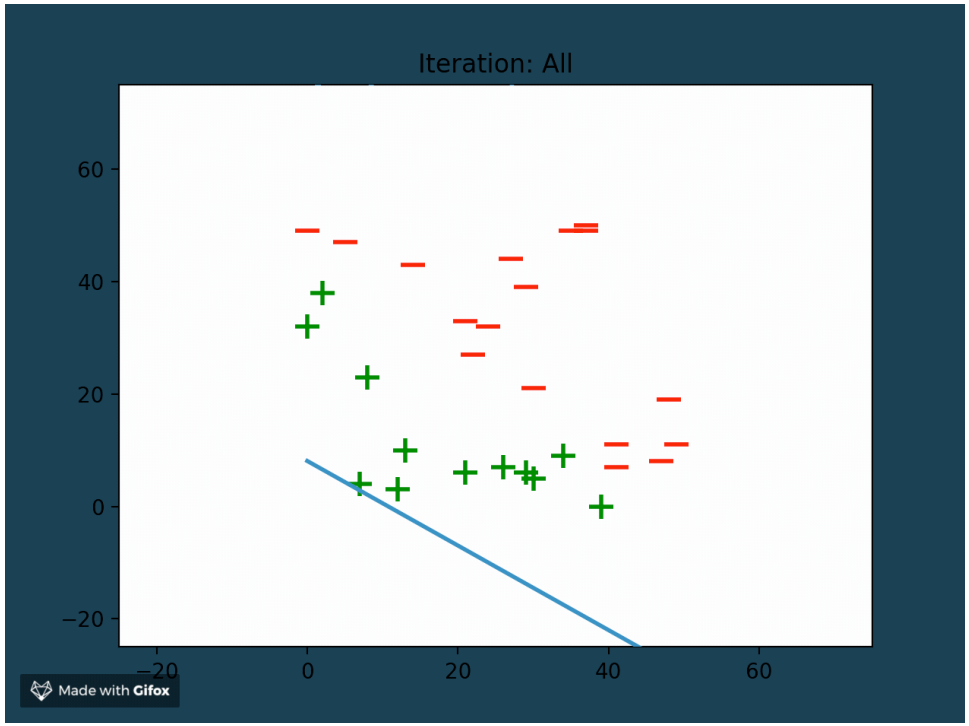
Perceptron Bias Term

The *bias* term is an adjustable, numerical term added to a perceptron's *weighted sum* of inputs and weights that can increase classification model accuracy. The addition of the bias term is helpful because it serves as another model parameter (in addition to weights) that can be tuned to make the model's performance on training data as good as possible.
The default input value for the *bias* weight is `1` and the weight value is adjustable.

```
weighted_sum = x1*w1 + x2*w2 + x3*w3 + 1*wbias
```

Perceptrons as Linear Classifiers

At the end of successful training, a *perceptron* is able to create a *linear classifier* between data samples (also called features). It finds this decision boundary by using the linear combination (or weighted sum) of all of the features. The perceptron separates the training data set into two distinct sets of features, bounded by the linear classifier.



Adjusting Perceptron Weights

The main goal of a *perceptron* is to make accurate classifications. To train a model to do this, perceptron weights must be optimizing for any specific classification task at hand.
The best weight values can be chosen by training a perceptron on labeled training data that assigns an appropriate label to each data sample (feature). This data is compared to the outputs of the perceptron and weight adjustments are made. Once this is done, a better classification model is created!

Perceptron Weighted Sum

The **first** step in the *perceptron* classification process is calculating the *weighted sum* of the perceptron's inputs and weights.
To do this, multiply each input value by its respective weight and then add all of these products together. This sum gives an appropriate representation of the inputs based on their importance.

Optimizing Perceptron Weights

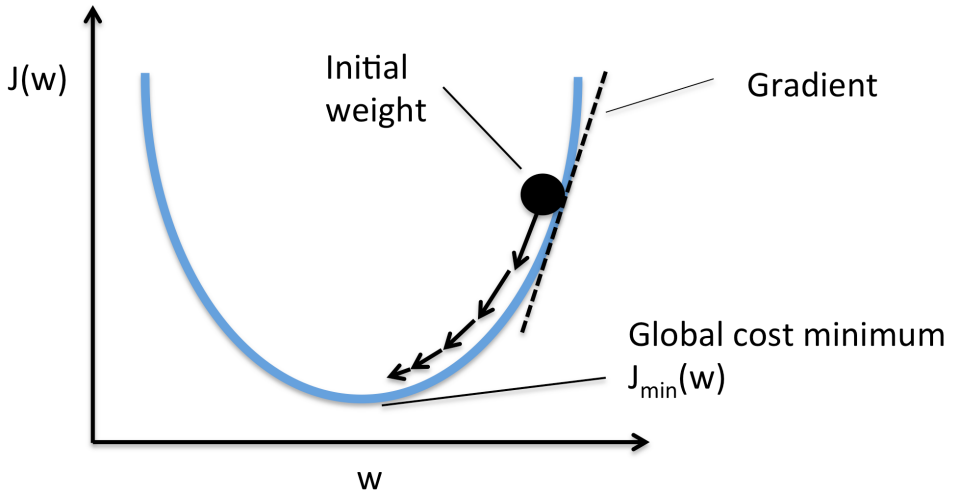
To increase the accuracy of a *perceptron's* classifications, its weights need to be slightly adjusted in the direction of a **decreasing** training error. This will eventually lead to minimized training error and therefore optimized weight values.
Each weight is appropriately updated with this formula:

$$weight = weight + (error * input)$$

codecademy

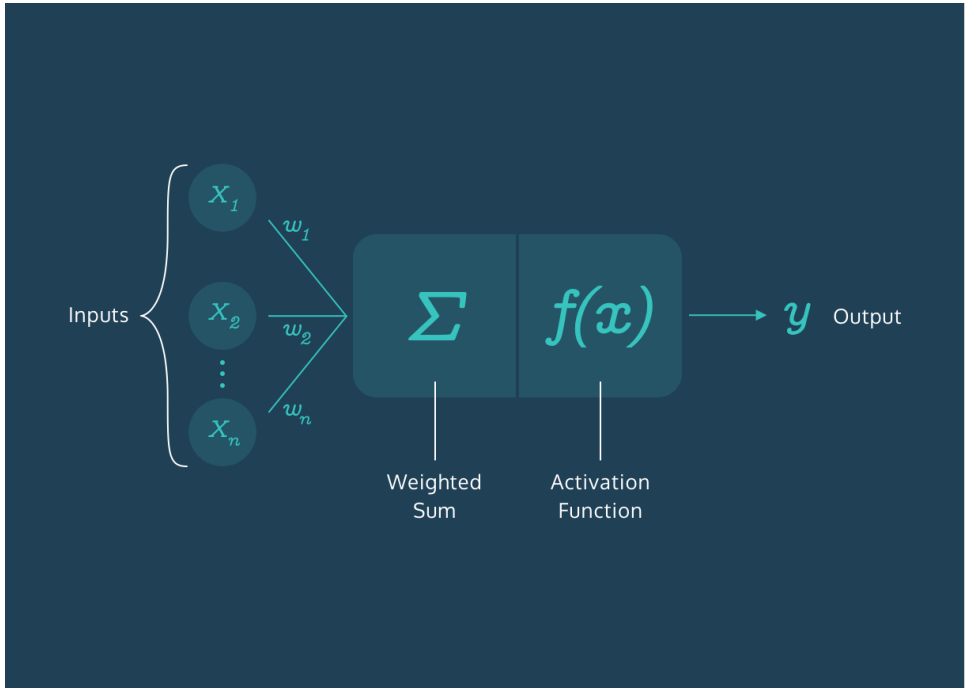
```
training_set = {(18, 49): -1, (2, 17): 1, (24, 35): -1,
(14, 26): 1, (17, 34): -1}
```

```
inputs = [x1,x2,x3]
weights = [w1,w2,w3]
weighted_sum = x1*w1 + x2*w2 + x3*w3
```



Introduction to Perceptrons

Perceptrons are the building blocks of *neural networks*. They are artificial models of biological neurons that simulate the task of decision-making. Perceptrons aim to solve binary classification problems given their input. The basis of the idea of the perceptron is rooted in the words **perception** (the ability to sense something) and **neurons** (nerve cells in the human brain that turn sensory input into meaningful information).



Perceptron Activation Functions

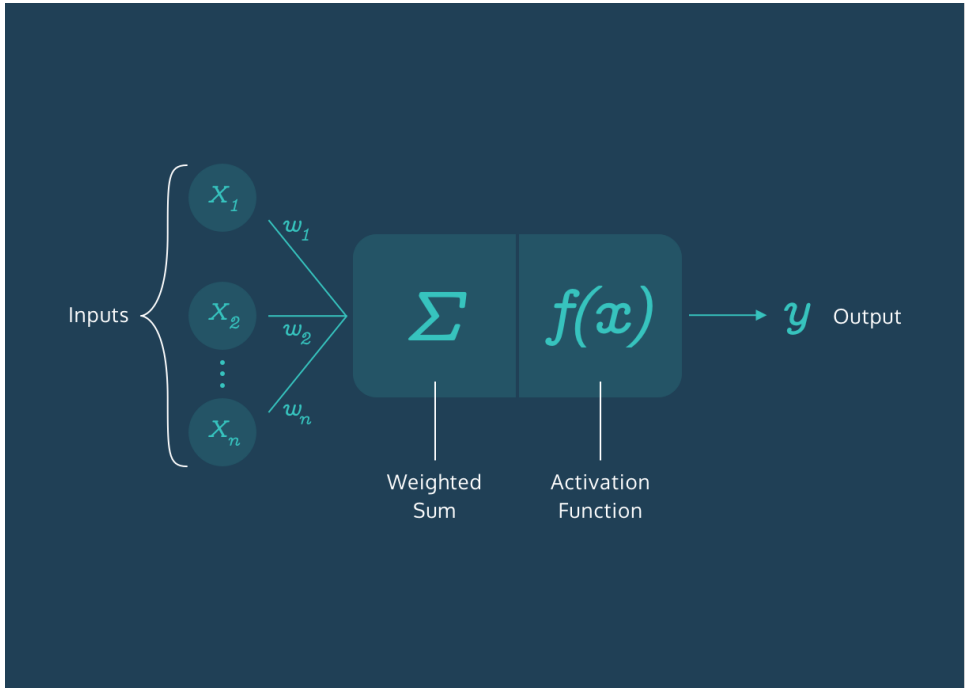
codecademy

The **second** step of the *perceptron* classification process involves an *activation function*. One of these special functions is applied to the *weighted sum* of inputs and weights to constrain perceptron output to a value in a certain range, depending on the problem.

Some example ranges are `[0,1]`, `[-1,1]`, `[0,100]`.

The *sign activation function* is a common activation function that contains the perceptron output to be either `1` or `-1`:

- If *weighted sum* > 0, return `1`.
- If *weighted sum* < 0, return `-1`.



Perceptron Training Error

Training error is the measure of how accurate a *perceptron's* classification is for a specific training data sample. It essentially measures "how bad" the perceptron is performing and helps determine what adjustments need to be made to the weights of that sample to increase classification accuracy.

$$training_error = actual_label - predicted_label$$

The **goal** of a perceptron is to have a training error of 0; this indicates that a perceptron is performing well on a data sample.

Actual Label	Predicted Label	Training Error
+1	+1	0
+1	-1	2
-1	-1	0
-1	+1	-2

Perceptron Main Components

Perceptrons use three main components for classification:

- *Input*: Numerical input values correspond to features. i.e. [22, 130] could represent a person's age & weight features.
- *Weights*: Each feature has a *weight* assigned; this determines the feature's importance. i.e. In a class, homework might be weighted 30%, but a final exam 50%. Therefore the final is more important to the overall grade (*output*).
- *Output*: This is computed using inputs and weights. *Output* is either *binary* (1,0) or a value in a continuous range (70-90).