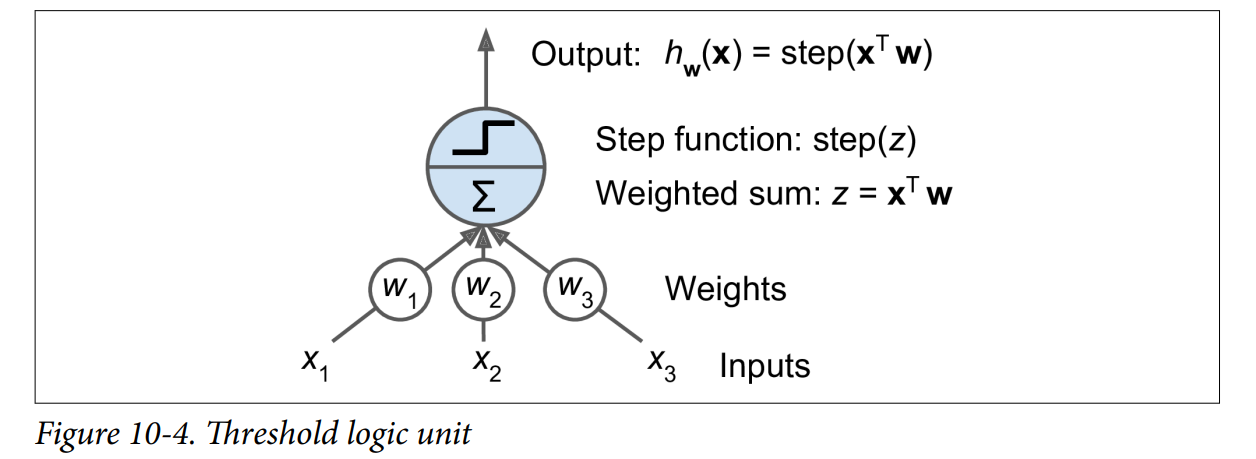
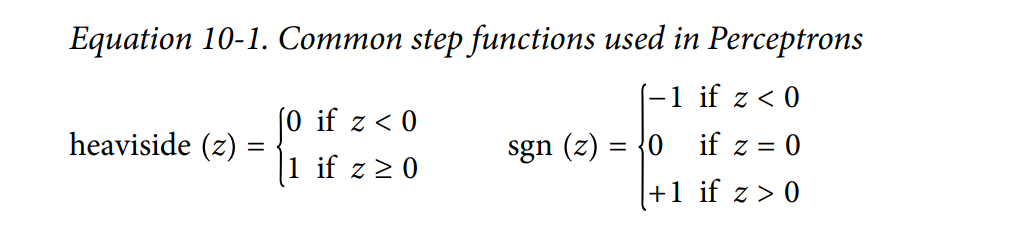
The Perceptron

* The Perceptron is one of the simplest ANN architectures.
* It is based on a slightly different artificial neuron called a threshold logic unit (TLU), or sometimes a linear threshold unit (LTU).
* The inputs and outputs are numbers and each input connection is associated with a weight.
* It computes a weighted sum of its inputs (z = w1 x1 + w2 x2 + ⋯ + wn xn ), then applies a step function to that sum and outputs the result.



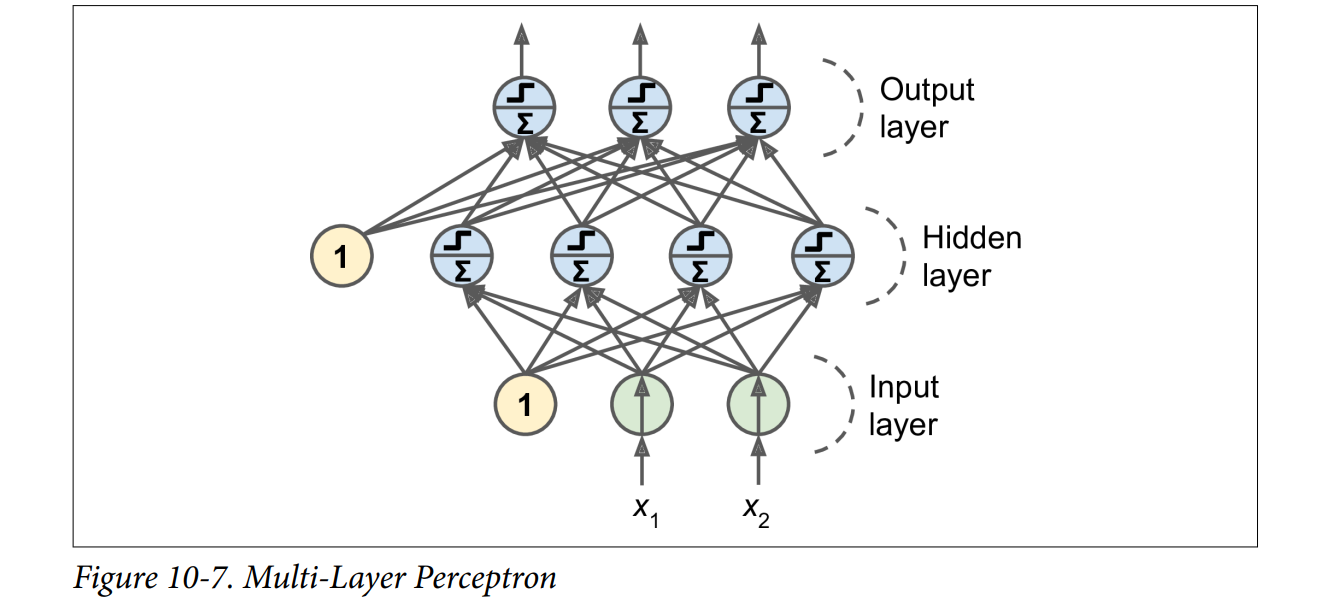
* The most common step function used in Perceptrons is,



* A single Perceptron can be used for simple linear binary classification. It computes a linear combination of the inputs and if the result exceeds a threshold, it outputs the positive class or else outputs the negative class.
* When all the neurons in a layer are connected to every neuron in the previous layer, it is called a **fully connected layer** or **dense layer.**
* The decision boundary of each output neuron is linear, so Perceptrons are incapable of learning complex patterns. However, if the training instances are linearly separable, this algorithm would converge to a solution. This is called the **Perceptron convergence theorem**.
* Perceptron learning algorithm strongly resembles Stochastic Gradient Descent.
* Scikit-Learn’s Perceptron class is equivalent to using an SGDClassifier with the following hyperparameters : loss="perceptron", learning\_rate="constant".

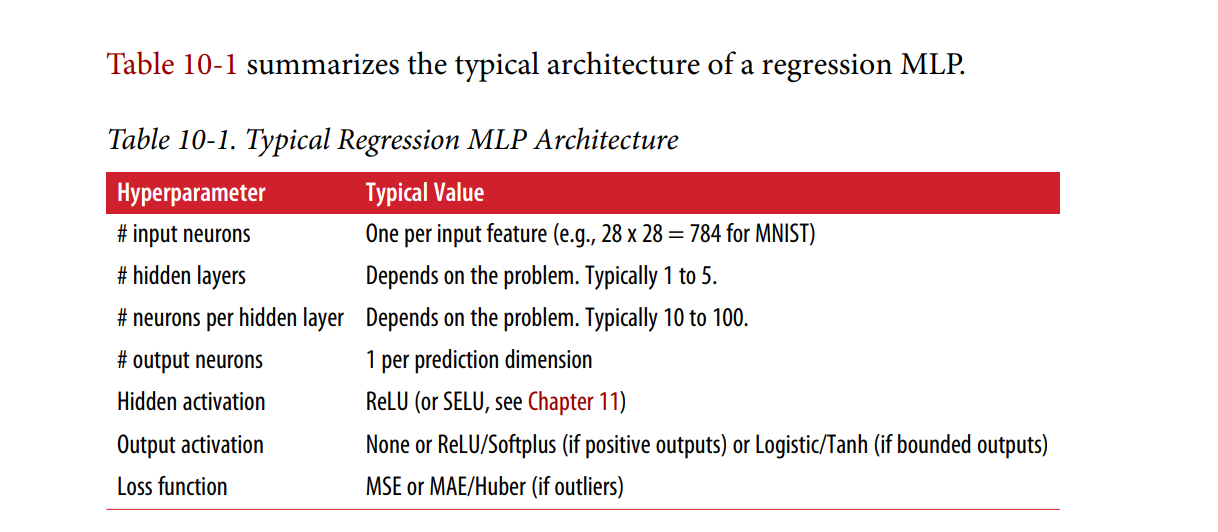
Multi-Layer Perceptron and Backpropagation

* An MLP is composed of one input layer, one or more layers of TLUs, called hidden layers, and one final layer of TLUs called the output layer.
* Every layer except the output layer includes a bias neuron and is fully connected to the next layer.
* It is important to initialize all the hidden layers’ connection weights randomly, or else training will fail. For example, if you initialize all weights and biases to zero, then all neurons in a given layer will be perfectly identical, and thus backpropagation will affect them in exactly the same way, so they will remain identical.



Regression MLPs

* First, MLPs can be used for regression tasks. If you want to predict a single value (e.g., the price of a house given many of its features), then you just need a single output neuron: its output is the predicted value.



Classification MLPs

* MLPs can also be used for classification tasks. For a binary classification problem, you just need a single output neuron using the logistic activation function: the output will be a number between 0 and 1.
* MLPs can also easily handle multilabel binary classification tasks.
* In multilabel classification problem, you need to have one output neuron per class, and you should use the softmax activation function for the whole output layer. The softmax function will ensure that all the estimated probabilities are between 0 and 1 and that they add up to one (which is required if the classes are exclusive).

