Coding Suggestions—Simple Perceptron

Introduction to Neural Networks

Mark Fleischer

Everyone has their own coding style and proclivities, but some basic, architectural concepts are very useful and important in order to develop, maintain and update a code-base. This is especially important when trying to take advantage of the object-oriented programming paradigm. This paper provides some basic information and guidelines for completing the programming assignments. Completing these assignments will be easier if you develop your initial code with an eye towards code-reuse and generalizations.

A simple Perceptron must be instantiated that can be trained using supervised learning based on the Perceptron Delta Function (or some other function). In using an object-oriented approach, it makes a lot of sense to define a *perceptron class* to allow you to instantiate/create a *perceptron* object with its own fields and functions. Below is a sort of *pseudo – code* description of the various fields and functions you may want to consider in defining a *perceptron* object:

Fields:

- *weights*: Each perceptron that you instantiate has its own set of weights that get modified. Thus, a vector or list or array object should be part of your *perceptron* object.
- *delta_weights*: You may find it useful for a variety of reasons to have a separate vector corresponding to the values of the change in weights based on computing the *Perceptron Delta Function.*
- *bias:* Each perceptron has its own bias value. This too may be updated, hence
- *delta_bias*: The value computed when applying the *Perceptron Delta Function.*
- *activity:* This field corresponds to the single, scalar value based on the weighted sum of the inputs plus the bias.
- *activation:* This field corresponds to the single, scalar value based on a function that computes the activation function value using the activity value as its argument.
- *delta:* This field corresponds to the *delta* value that is computed using the *Perceptron Delta Function.*

Functions/Methods:

- *calc_activity*(xxx): This function takes an input vector (or whatever data structure corresponds to input values) and computes the activity value and assigns it to the *activity* field.
- *calc_activation*(xxx): This function takes as an input, the activity value and outputs or assigns the activation function value to the *activation* field.
- *set_delta_weights*(xxx): This function takes the input vector and the eta value to calculate the *delta_weights.*
- *update_weights*(xxx): This function updates the *weights* using the values of the *set_delta_weights* function.

Constructor Functions:

The *perceptron* object is instantiated using a *constructor* function. The basic arguments to such a constructor function for each perceptron is its set of weights and bias. The constructor function must also take care to initialize certain of its fields.

The above elements are all that you really need to construct a useful Perceptron object. Naturally, there is going to be other code you will need to write to embellish and extend this objects utility and also to solve the programming assignment questions. You might want to consider testing your code using examples covered in the video lectures to make sure it is working correctly. You might also want to further develop code to perform the *Perceptron Delta Function* and the *feed-forward, back-propogation* (FFBP) algorithm for a specified number of iterations.