Computer Science 158

Neural Networks in Python

1 The Neural Network Class

`nn = neuralNet(nInput, nHidden, nOutput)` creates a network with the specified number of nodes in each layer. The initial weights are random values between -2.0 and 2.0.

`nn.evaluate(input)` returns the output of the neural network when it is presented with the given input. Remember that the input and output are *lists.*

`nn.train(trainingData)` carries out a training cycle. As specified earlier, the training data is a list of input-output pairs. There are four optional arguments to the `train` function:

`learningRate` defaults to 0.5.

`momentumFactor` defaults to 0.1. The idea of momentum is discussed in the next section. Set it to 0 to suppress the affect of the momentum in the calculation.

`iterations` defaults to 1000. It specifies the number of passes over the training data.

`printInterval` defaults to 100. The value of the error is displayed after `printInterval` passes over the data; we hope to see the value decreasing. Set the value to 0 if you do not want to see the error values.

You may specify some, or all, of the optional arguments in the following format.

```
nn.train(trainingData,
         learningRate=0.8,
         momentumFactor=0.0,
         iterations=100,
         printInterval=5)
```

`nn.test(testingData)` evaluates the network on a list of examples. As mentioned above, the testing data is presented in the same format as the training data. The result is a list of triples which can be used in further evaluation.

`nn.getIHWeights()` returns a list of lists representing the weights between the input and hidden layers. If there are t input nodes and u hidden nodes, the result will be a list containing t lists of length u.

`nn.getHOWeights()` returns a list of lists representing the weights between the input and hidden layers. If there are u hidden nodes and v output nodes, the result will be a list containing u lists of length v.

`nn.useTanh()` changes the activation function from the sigmoid function to a commonly used alternative, the hyperbolic tangent function.

2 Examples

These examples are used at the end of neural.py to demonstrate usage.

The standard XOR example.

inputs		output
0	0	0
0	1	1
1	0	1
1	1	0

Voter preferences. This is training information. Voters were asked to rate the importance of some issues and then to identify their party affiliation. The inputs are, from left to right, the importance of budget, defense, crime, environment, and social security. They are ranked on a scale of 0.0 to 1.0. The outputs are 0.0 for Democrat and 1.0 for Republican.

inputs					output
0.9	0.6	0.8	0.3	0.1	1.0
0.8	0.8	0.4	0.6	0.4	1.0
0.7	0.2	0.4	0.6	0.3	1.0
0.5	0.5	0.8	0.4	0.8	0.0
0.3	0.1	0.6	0.8	0.8	0.0
0.6	0.3	0.4	0.3	0.6	0.0