neural networks regression using pybrain

I need to solve a regression problem with a feed forward network and I've been trying to use PyBrain to do it. Since there are no examples of regression on pybrain's reference, I tried to adapt it's classification example for regression instead, but with no success (The classification example can be found here: http://pybrain.org/docs/tutorial/fnn.html). Following is my code:

This first function converts my data in numpy array form to a pybrain SupervisedDataset. I use the SupervisedDataset because according to pybrain's reference it is the dataset to use when the problem is regression. The parameters are an array with the feature vectors (data) and their expected output (values):

```
def convertDataNeuralNetwork(data, values):

fulldata = SupervisedDataSet(data.shape[1], 1)

for d, v in zip(data, values):

    fulldata.addSample(d, v)

return fulldata
```

Next, is the function to run the regression. train_data and train_values are the train feature vectors and their expected output, test_data and test_values are the test feature vectors and their expected output:

```
regressionTrain = convertDataNeuralNetwork(train_data, train_values)

regressionTest = convertDataNeuralNetwork(test_data, test_values)

fnn = FeedForwardNetwork()

inLayer = LinearLayer(regressionTrain.indim)
hiddenLayer = LinearLayer(5)
outLayer = GaussianLayer(regressionTrain.outdim)

fnn.addInputModule(inLayer)
fnn.addModule(hiddenLayer)
fnn.addOutputModule(outLayer)

in_to_hidden = FullConnection(inLayer, hiddenLayer)
hidden_to_out = FullConnection(hiddenLayer, outLayer)

fnn.addConnection(in_to_hidden)
fnn.addConnection(hidden_to_out)

fnn.sortModules()

trainer = BackpropTrainer(fnn, dataset=regressionTrain, momentum=0.1, verbose=True

for i in range(10):

    trainer.trainEpochs(5)

    res = trainer.testOnClassData(dataset=regressionTest )

    print res
```

when I print res, all it's values are 0. I've tried to use the buildNetwork function as a shortcut to build the network, but it didn't work as well. I've also tried different kinds of layers and different number of nodes in the hidden layer, with no luck.

Does somebody have any idea of what I am doing wrong? Also, some pybrain regression examples would really help! I couldn't find any when I looked.

Thanks in advance

python | machine-learning | neural-network | regression | pybrain

asked Jun 2 '13 at 5:01

Alberto A
203 2 13

If you are interested in neural networks, you may consider joining the machine-learning site: area51.stackexchange.com/proposals/41738/machine-learning – trav1s Jun 6 '13 at 5:30

1 pretty sure you want the output layer to be linear for regression---you probably also want to use sigmoidal/tanh hidden units – Ben Allison Jun 6 '13 at 14:30

add comment

2 Answers

`pybrain.tools.neuralnets.NNregression` is a tool which

> Learns to numerically predict the targets of a set of data, with optional online progress plots.

so it seems like something well suited for constructing a neural network for your regression task.

answered Aug 29 '13 at 16:28

piokuc
11.7k 1 12 36

add comment

I think there could be a couple of things going on here.

First, I'd recommend using a different configuration of layer activations than what you're using. In particular, for starters, try to use sigmoidal nonlinearities for the hidden layers in your network, and linear activations for the output layer. This is by far the most common setup for a typical supervised network and should help you get started.

The second thing that caught my eye is that you have a relatively large value for the `weightDecay` parameter in your trainer (though what constitutes "relatively large" depends on the natural scale of your input and output values). I would remove that parameter for starters, or set its value to 0. The weight decay is a regularizer that will help prevent your network from overfitting, but if you increase the value of that parameter too much, your network weights will all go to 0 very quickly (and then your network's gradient will be basically 0, so learning will halt). Only set `weightDecay` to a nonzero value if your performance on a validation dataset starts to decrease during training.

answered Sep 19 '13 at 5:37

lmjohns3
1,907 2 4 17

add comment

Not the answer you're looking for? Browse other questions tagged python

machine-learning | neural-network | regression | pybrain **or ask your own question.**