Midterm Project

Due 4/09/2021

The objective of this assignment is for you to become familiar with the basic concepts of Neural Networks. The recommended platform is jupyter lab.

To refresh your understanding of neural networks, refer to this link:

https://towardsdatascience.com/understanding-neural-networks-19020b758230

For backpropagation explanation, refer to this link:

https://towardsdatascience.com/understanding-backpropagation-algorithm-7bb3aa2f95fd

Use the notebook at:
https://cse.sc.edu/~yiannisr/590/2021/Midterm-Question.ipynb

In it you will find code providing the skeleton for creating a Neural Network using fully connected Layers. In particular you will find the following parts:

Some imports
class Layer: abstract class, implemented.
class FCLayer(Layer): Fully connected layer, you need to implement:
- def **forward_propagation**(self, data):
- def **backward_propagation**(self, error, lr):

class ActivationLayer(Layer): Activation layer, implemented.
functions for mean square error and tanh activation function, implemented.
class Network: The Neural Network class, you need to implement:
- def **predict**(self, data): This method is used to predict the labels with a trained network. You need feed all the samples to the network.
- def **fit**(self, x_train, y_train, epochs, learning_rate): This method is used to predict a label, then use the prediction to calculate the error, which is then backpropagated to adjust the weights.

Testing code:
some very basic data for prototyping made for fast execution.
```
x_train = np.array([[[0,0]], [[0,1]], [[1,0]], [[1,1]]])
y_train = np.array([[[0]], [[1]], [[1]], [[0]]])
```

The result of testing should have the form of
```
epoch 1/1000   error=0.549278
...
epoch 1000/1000   error=0.000278
[array([[0.00093899]]), array([[0.97740932]]), array([[0.97557041]]), array([[-0.00183729]])]
```

See next page.

Testing code for the MNIST dataset.
(x_train, y_train), (x_test, y_test) = mnist.load_data() Proper separation between train and test data.

A network is implemented with 3 fully connected layers.

Training:
net.fit(x_train[0:1000], y_train[0:1000], epochs=200, learning_rate=0.1) Use only the first 1000 samples for efficiency and only for 200 epochs.

```
epoch 1/200   error=0.239762
…
epoch 200/200   error=0.001406
```

Testing on the first five samples:
out = net.predict(x_test[0:5])

The results will be something like:

```
predicted values :
[1 2 2 0 0 1 1 0 1 0]
true values :
[0 2 1 0 0 0 0 0 0 0]
```

Clearly the accuracy of the system is not very good. Not surprising when you have a simple network, no convolutional layers, only 1000 samples and training for 200 epochs.

3. (Grad Credit)
Try to improve the performance of the network by making it deeper, that is add a few more layers (fully connected followed by activation). Remember the size of the output of a layer should be equal to the size of the input of the next layer. Report on the improvement.

Important: **Write your own code**. I will use plagiarism tools to detect copied code from the internet.

What to Submit

Please together with your code in the form of a jupyter notebook provide a report discussing your implementation together with some illustrative examples (screen shots). Ensure that is well written and formatted. Document your work in images and use captions to describe what each image displays. Use of LaTeX is encouraged.