INF 552 Assignment5 Neural Network Yi Ren, YimingLiu

1. Implementation

- **Data Structure**

```
Inputs: X (training set), y(desired output),
        weights, deltas(result of error function)
        alpha, epochs

Outputs: Prediction y, accuracy
```

- **Logic**

1. Load data, get Xtest,Ytest Xtrain,Ytrain;
2. Propagate the input forward - calculate the outputs of all units;
3. Propagate the error backward - update weights of all units by gradient descent with learning rate alpha (changed by a small step against the direction of the gradient of the error function to decrease the error quickly);
4. Repeat until the condition is satisfied.
5. The prediction is similar to the forward part in back- propagation except we don't need to keep all the values of the activations for each neuron, so we keep only the last one.

- **Output**
  The accuracy is about 80% with epoch 1000 or 5000.

```
prediction: [1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0,
0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0
, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,
 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0]
accuracy: 0.855421686747

prediction: [1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0,
0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1
, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,
 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0]
accuracy: 0.867469879518
```

- **Optimization**

For optimization, the first two layers have been added a bias unit which corresponds to the threshold value for the sigmoid function. The program can be optimized by adding overfitting solutions such as adding regularization and dropout. Also, it is helpful to improve the function by conducting momentum and batch normalization on it, and expanding the model with parameterized layer sizes.

---

2. Description of Library Function

The library function is from sklearn.neural_network.MLPClassifier [1] which is a Multi-layer Perceptron classifier.For optimizing the log-loss function, we can choose LBFGS or stochastic gradient descent. Fit(X, y) is for training data and predict(X) is for predicting the results on test data and getting the accuracy.

- **Usage**

```
from sklearn.neural_network import MLPClassifier

nn = MLPClassifier(solver='sgd',alpha=0,hidden_layer_sizes
=(100,), activation='logistic', learning_rate_init=0.1,max_iter=1000)

nn.fit(X, y)
with open('downgesture_test.list') as f:
    total += 1
    for test_image in f.readlines():
        test_image = test_image.strip()
        p = nn.predict([load_pgm_image(test_image),])[0]
        print nn.predict([load_pgm_image(test_image),])
        print('{}: {}'.format(test_image, p))
        if (p != 0) == ('down' in test_image):
            correct += 1
        pre.append(p)
print "prediction:" , pre
print('correct rate: {}'.format(correct / total))
```

- **Output**

```
prediction: [1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
correct rate: 66.00.972251568416
```

To compute the accuracy, we can also use score(X, y, sample_weight=None) in the library, which is defined as (1 - u/v), where u is the regression sum of squares ((y_true - y_pred) ^ 2).sum() and v is the residual sum of squares ((y_true - y_true.mean()) ^ 2).sum().

3.Related Applications

One of interesting applications of neural networks is getting rat thoughts to move robotic parts[2]. To achieve the goal, the first step is to train a rat to press a lever, which activates a robotic arm. Robotic arm delivers reward to rat. Then, by attaching a 16-probe array to the rat's brain,it can record the activity of 30 neurons at once. After training a neural network program to recognize brain-activity patterns during a lever press , the neural network can predict movement from the rat's brain activity alone, so when the rat's brain activity indicates that it is about to press the lever, robotic arm moves and rewards the rat - the rat does not need to press the lever, but merely needs to "think" about doing so.

Division of Work

Both members wrote the program to verify the accuracy of each others' code, and finish this report together.

Reference

[1]http://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html

[2]http://cs.stanford.edu/people/eroberts/courses/soco/projects/2000-01/neural-networks/Applications/index.html