COMP 135 – Machine Learning – Fall 2017

# Empirical/Programming Assignment 4

**Due date:** Wednesday, 12/6 (by the beginning of class, both paper and electronic)

In this assignment you will the Backpropagation algorithm for neural networks and evaluate its performance with changing network architecture.

## 1 Data

In this assignment we use two datasets which you can find on the course web page. The first captures the 8-3-8 network example data from [M] which was shown in lecture. The second is a digit recognition dataset[1] For this assignment we have already split the data into train and test portions. The files are in arff format so you should be able to use your reader from previous assignments to read the data files. To be clear you can reuse your own code from a previous assignment but please do not use weka's code or code from any other source for that purpose.

## 2 Network structure and the backpropagation algorithm

In this assignment we apply a neural network to classification problems with more than two labels. To implement this we use multiple output units with "one hot" coding. For example, for the digit dataset, since there are 10 possible labels we will have 10 output units. The encoding on label=2 is given by assigning 0100000000 to the corresponding output units. Your code should work on any arff dataset with numerical features. Therefore it should read the dataset to figure out the number of labels and hence the corresponding number of output units to be used.

During training we use the binary labels as the required outputs of the corresponding units. During testing each output unit calculates a score for the example. This is given by the value $x_i$ in the forward pass of the backpropagation algorithm in the slides. We then predict the label which has the highest score.

We will work with networks with $d$ (depth) hidden layers each of width $w$. As in the slides there is an edge (and weight) between nodes in consecutive layers. The first hidden layer is connected to all inputs (features in the examples) and the output layer is connected to the last hidden layer. In the case where $d = 0$ the output layer is directly connected to the input layer, i.e., there are no hidden nodes. In this case each output node is a "perceptron-like" unit - it calculates a linear threshold decision boundary over the inputs, but it differs from perceptron in that it uses the sigmoid function and has a different update rule. In this way, by varying $d$ and $w$ we can explore the power of large networks over single linear units, and the ability of our algorithm to successfully learn such networks.

The (stochastic gradient descent variant of the) backpropagation algorithm is exactly as given in lecture slides. The pseudocode given there is directly applicable to multiple output units. As in that code, we go through the dataset sequentially and update the weights with each example. You should implement this algorithm repeating the linear scan over examples `Iter` times (number specified below). In addition for reporting we record the number of mistakes during each training iteration and evaluate the network on the test set after each iteration so that the overall code has this structure as in the pseudocode on the next page. The weights should be initialized with independent random numbers uniformly sampled in the range $[-0.1, 0.1]$. Please make sure to seed the random number generator so that your results are reproducible. The learning rate should be set to $\eta = 0.1$.

[1]For more information on this dataset please see `https://archive.ics.uci.edu/ml/machine-learning-databases/optdigits/`

```
learn(w,d,traindata,testdata)
Construct network with w,d and Initialize weights
Repeat Iter times
   For each example in traindata
       Update weights using backpropagation formulas
   End For
   Calculate the train error rate (number of mistakes during training / number of examples)
   For each example in testdata
      Calculate scores of output units and evaluate prediction (correct/incorrect)
   End For
   Calculate test set error rate
End Repeat
```

## 3 Implementation and Evaluation

You should implement your code in a modular manner and provide an interface so that we can evaluate it with the arguments $w, d$ and data files as in the pseudo code above. You may assume that the data files reside in the same directory as the code. This will allow us to test your code on different data without modification.
Implement your code in a modular manner so that it can be tested with different values for $w, d$. You may assume that the data files reside in the same directory as the code.
You should write your own code for all portions of the assignment. If this helps in your implementation it is fine to use a library for matrix and vector operations but otherwise please do not use any special libraries or code from any source.
Recall that the sigmoid function $1/(1+e^{-a})$ requires calculation of $e^{-a}$ where $a$ is its argument. If you encounter numerical issues due to this you may replace $a$ with $\max\{-50, a\}$ (or use any similar thresholding).
First run your code on the 838 dataset using 3000 iterations. Here we have 8 examples and the labels are 1 to 8 so that a 1-hot representation yields the desired structure. Plot the training set error as a function of iterations. In addition, your code should print the representation of the hidden unit representation after the last iteration as illustrated in [M] and the lecture slides. Discuss the results: What can you observe about the learning curve? and is the "binary representation" discovered by your code?
Next evaluate the algorithm on the optdigit dataset. Use 200 iterations in this part. To explore a range of structures evaluate the algorithm with (1) $d = 3$ and $w = 5, 10, 15, 20, 30, 40$ and then with (2) $w = 10$ and $d = 0, 1, 2, 3, 4, 5$. Plot the training and test error as a function of iterations for each of these runs. In addition for (1) plot the test error after the last iteration as a function of $w$ and similarly for (2) plot the error as a function of $d$. What can you conclude from the learning curves and additional plots? How does the performance vary as a function of $d$ and $w$? What are the best $w, d$ settings among the ones you tested?

## 4 Submitting your assignment

### 4.1 What to submit

You should submit the following items both electronically and in hardcopy:

- All your code for data processing, learning algorithms, experiments, test programs, etc. Please write clear code and document it as needed.

  Please make sure that your code runs on `homework.eecs.tufts.edu`. Include a README file with instructions how to compile and run your code to reproduce the results of experiments. If this is nontrivial please include a script to compile and run your code.

  In addition, your submission should be done so that we can directly run your implementation of backpropagation (without any modifications) with arguments: w,d,traindata,testdata. Please package an interface to your code that enables this and document how to use it in the README file.

Your submitted code should assume that the data files are in the same directory as the program. There is no need to submit the original data files. There is no need to submit any intermediate files or results. These should be generated when your program is run.

- A short report with the results and plots as requested and a discussion with your observations from these plots. Please make sure that your discussion responds to questions posed in the assignment text.

## 4.2 When and How to Submit

- **Please submit electronically using provide by 12:00 noon of the due date.** Put all the files mentioned in the previous subsection into a zip or tar archive (no RAR please). For example call it `myfile.zip`. Then submit using: `provide comp135 pp4 myfile.zip`
- **Please submit a printed copy (of everything) by the beginning of class (4:30pm).**

## 4.3 Grading

Your assignment will be graded based on the **code**, its **clarity**, **documentation and correctness**, the **presentation** of the results/plots, and their **discussion**.