Overview

In this assignment, you will experiment with three classifiers - kNN, logistic regression, and a neural network with one hidden layer. You will also extend the software and experiment with these extensions. Note that you should understand the code first instead of using it as a black box. You will be working with the following datasets:

1. **Fruit**: The first dataset consists of height and width measurements of oranges and lemons. The goal is to classify fruit just from these measurements. The two files are `fruit_train.mat` and `fruit_test.mat`.

2. **Spirals**: The second dataset is called the “two-spirals” problem. Each data example belongs to one of two inter-woven spirals, and is represented by two coordinates. The goal is to learn to classify a two-dimensional input as belonging to one or the either spiral. The four files are `spiral1_train.mat`, `spiral1_test.mat`, `spiral2_train.mat` and `spiral2_test.mat`.

3. **Digits**: The last dataset consists of hand-written digits, 4s and 9s, represented as 28x28 pixel arrays. The two files are `mnist_train.mat` and `mnist_test.mat`.

Some code for visualizing the datasets has been included (`plotFruit.m`, `plotDigits.m`, `plotSpiral.m`). You may find these functions useful.

1. Simple Classifiers

   a) **k-Nearest neighbors [4 points]**

   The k-NN implementation you saw in the tutorial is included in the code. Run the k-NN classifier on the fruit data. Try different values of k and report your findings.
b) Logistic regression [6 points]

Look through LRtemplate.m and logistic.m. Complete the implementation of logistic regression by providing the missing part of logistic.m. Use checkgrad.m to make sure that your gradients are correct. Run the code on the fruit dataset and report your findings.

c) Penalized logistic regression [6 points]

Implement penalized logistic regression from Lecture 6 by modifying logistic.m to include a regularizer. Use checkgrad.m to verify the gradients. Run the code on the fruit dataset. Try different values of the weight regularization parameter and report your findings.

2. Neural Networks

Code for training a neural network with logistic output units and a squared error function is included. The main components are:

- \texttt{initBp.m}: initializes the weights and loads the training and testing patterns
- \texttt{runBp.m}: runs numEpochs of backprop learning (using the following two functions)
- \texttt{calcGrad.m}: uses backprop to calculate weight gradients
- \texttt{updateWts.m}: updates the weights using gradient descent with momentum

a) Basic generalization [6 points]

Train a neural network with 10 hidden units on the fruit dataset. You should first use \texttt{initBp} to initialize the net, and then execute \texttt{runBp} repeatedly (more than 5 times). Note that \texttt{runBp} runs 100 epochs each time and will output the statistics and plot the error curves. Examine the statistics and plots of training error and test error (generalization). How does the network’s performance differ on the training set versus the test set during learning?

b) Classification error [4 points]

You should add a second performance measure to the mean-squared error, the mean classification error. This applies to both the training and test set. You can consider the output
correct if the correct label is given a higher probability than the incorrect label. You should then count up the total number of examples that are classified incorrectly according to this criterion for training and testing respectively, and maintain this statistic at the end of each epoch. Plot the results.

c) Learning rate [4 points]

Try different values of the learning rate \( \eta \) (“eta”) defined in \texttt{initBp.m}. You should reduce it to 0.001, and increase it to 0.02 and 0.05. What happens to the convergence properties of the algorithm (looking at both MSE and %Correct)? How would you choose the best value of this parameter?

d) Number of hidden units [4 points]

Set the learning rate \( \eta \) (“eta”) to 0.002 and try different numbers of hidden units on this problem. You should use two values \{2, 5\}, which are smaller than the original and two other \{30, 100\}, which are larger. Describe the effect of this modification on the convergence properties, and the generalization of the network.

e) Cross-entropy and softmax [6 points]

Modify the code so that the cross-entropy is used as the objective function and the softmax activation is used for the outputs, based on the notes from Lecture 5. Run the modified code on the fruit data and describe the effect on convergence and generalization (for a fixed number of hidden units 10, and learning rate \( \eta = 0.01 \)).

3. Spirals and Digits

In this section you will run all three classifiers (k-NN, logistic regression, and neural net) on the remaining datasets. You are asked to experiment with different parameters (the parameters from parts 1 and 2 are a good starting point) and report your findings. Which classifiers worked well and which did not? Look for signs of overfitting and underfitting (large error on the training data). Most importantly, try to explain why you think the results turned out the way they did. Try to include only relevant supporting information. For example, do not include learning curves for hundreds of epochs if you are comparing only the final values.
a) Spirals1 [6 points]

Run all three classifiers on the first spirals dataset (spirall_train.mat and spirall_test.mat). Experiment with different values of the parameters for all three classifiers. Report your findings and include supporting plots.

b) Spirals2 [6 points]

Run all three classifiers on the second spirals dataset (spiral2_train.mat and spiral2_test.mat). Experiment with different values of the parameters for all three classifiers. How do the results differ from the results on the first spirals dataset? Report your findings and include supporting plots.

c) Digits [6 points]

Run all three classifiers on the digits dataset. Experiment with different values of the parameters for all three classifiers. Report your findings and include supporting plots.

Write up

Hand in answers to all the questions in the parts above. The goal of your write-up is to document the experiments you’ve done and your main findings. So choose the plots appropriately and explain the results.

Submit your code and write-up in a zip file by email to csc411ta@cs.toronto.edu, and turn in a hard copy of the write-up before class on October 18 as well.