CS4442b/9542b: Artificial Intelligence II, Winter 2017
Assignment 1: Hand Written Digit Classification
Due: February 12

Instructions:

- Submit your assignment through OWL by 11:55pm on the due date.
- Include all matlab code you write and a soft copy of answers to questions (in pdf format). Do not need to include code/data that I give you. Do not zip your files, submit all files individually.
- Except for problem 10, you cannot use matlab built-in tools for machine learning/classification, you have to develop learning tools from scratch. You can use standard matlab functions for matrix manipulations. If in doubt about use of any matlab function, post a question on the discussion board. For problem 10, you can use matlab built in neural network tools as specified in the problem description.
- Make sure to use the function names, input/output parameters as I specify, including capitalization. If you write helper functions, you can give them any names you wish.
- This assignment has extra credit questions. Extra credit counts only towards the assignments, it does not transfer to quizzes.

Data: In this assignment you will develop and test several classifiers for hand written digit recognition using the reduced version of the famous MNIST dataset[1]. The images in the original dataset are of size 28 by 28 pixels. We rescaled images to size 16 by 16 pixels for computational efficiency. Each 2D image is stretched into a vector for training/testing. Thus each image is a 256 dimensional feature vector of raw pixel intensities.

Load data into matlab using command `load A1`. Matrix `X_train` (size 5000 by 256) contains 5000 samples to use for training. Samples are stored as rows. Thus the first training image is in `X_train(1,:)`. You can visualize any digit image with the provided matlab function `showImage`. For example, to see the 7th training image, use `showImage(X_train(7,:))`.

Column vector `Y_train` contains the true labels of samples in `X_train`. Thus its size is `5000 by 1`. Since matlab starts indexing at 1, it is convenient to denote digit `0` with label 1, digit `1` with label 2, ..., digit `9` with label 10. Thus `Y_train` contains integers from 1 to 10, corresponding to digits from 0 to 9. For example, the label of the first training image is `Y_train(1)` is 6, which means the first training image has digit 5. There is also matrix `X_test` containing 2000 samples to be used for testing, and column vector `Y_test` containing true labels of samples in `X_test`. All results in the report for the problems below should be computed on data in "A1.mat".

For debugging purposes, I also provide you with a small dataset in `A1_tiny.mat`. The format of the data in this file is the same as in `A1.mat`, except that each matrix has a suffix "tiny". If you want to experiment with the full version[2] of the MINST dataset, you can get it from file `A1_full.mat`. Again, all results should be reported for the data in `A1.mat`. However, for the

[1] `http://yann.lecun.com/exdb/mnist/`

[2] The images are still reduced to the size 16 by 16 pixels, but otherwise `A1_full.mat` contains the same data as the MINST dataset. Therefore you can compare your performance on the test data to what researchers were able to get previously on the MINST dataset, see `http://yann.lecun.com/exdb/mnist/`.

competition (problem 12), you can submit a classifier trained on the full version of the MINST dataset. In fact, for the competition you can train on both the training and test samples contained in `A1_full.mat` since the competition is based on different data (provided by students in the class).

You develop all functions in the assignment so that they can handle input data of any size. When I ask you for training error, test error, etc. on `X_test`, `X_train`, etc., I always refer to the data stored in file `A1.mat`.

Input/output samples: For each (non extra credit) problem, I provide sample input/output in files with corresponding names. For example, for problem 1, the input/output sample file is named `p1_sample`.

1. (5%) Some of the problems in the assignment deal with the 2-class classification problems. Therefore, it is useful to have a function that extracts data for 2-class problem from our multiclass data. Write a matlab function

   ```
   [X_out,Y_out] = p1(X,Y,l1,l2)
   ```

   that takes as input samples in matrix `X`, true labels in column vector `Y`, and two distinct integers specifying labels `l1, l2`. The program should extract from `X` only the samples whose true label is `l1` or `l2` and store them in output matrix `X_out`. It should also store the (renamed) corresponding labels in `Y_out`. In `Y_out`, the smaller of `l1, l2` should be renamed with `1`, and the larger with `2`.

   For example, for input $X = \begin{bmatrix} 1 & 2 \\ -1 & 4 \\ 3 & 2 \\ 1 & 7 \\ 3 & 5 \end{bmatrix}$, $Y = \begin{bmatrix} 4 \\ 1 \\ 3 \\ 2 \\ 3 \end{bmatrix}$, `l1 = 3, l2 = 4`,

   the output of `p1(X,Y,l1,l2)` is $\texttt{X\_out} = \begin{bmatrix} 1 & 2 \\ 3 & 2 \\ 3 & 5 \end{bmatrix}$, $\texttt{Y\_out} = \begin{bmatrix} 2 \\ 1 \\ 1 \end{bmatrix}$.

   Label `3` is renamed with `1` and label `4` with 2. You must avoid loops in this function in order to receive the full credit. Note that the order of samples in the output can be different from the order in the input, depending on your implementation. Useful matlab functions: `find`.

2. (10%) Assume multiclass setting for this problem. Write a matlab function

   ```
   [err,CONF] = p2(C,T)
   ```

   that takes as an input a column vector `C` of classification results, and a column vector `T` of true labels. Assume labels are from 1 to `m`, and compute `m` is the maximum integer contained in `C` and `T`. The output should be the error rate `err` and confusion matrix `CONF`. Here `err` is the fraction of misclassified examples, which is defined as the number of incorrectly classified examples divided by the total number of examples. `CONF` is an m by m matrix where `CONF(i,j)` is the number of examples of class `i` that are classified as class `j`. Thus diagonal entries in `CONF` are the correct classifications, and off-diagonal entries are errors.

For example, for input: $C = \begin{bmatrix} 3 \\ 1 \\ 1 \\ 3 \\ 2 \\ 3 \end{bmatrix}$, $T = \begin{bmatrix} 2 \\ 4 \\ 1 \\ 3 \\ 3 \\ 3 \end{bmatrix}$, output is: $CONF = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 2 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$, $err = \frac{3}{6}$. For example, `CONF(3,2) = 1` because there is exactly one example (5th one) of true class 3 that got classified as class 2.

If you write this function without loops, you will get `5%` extra credit. Useful matlab commands to avoid loops: `find, sub2ind, unique, hist, sum`.

3. (10%) Assume multiclass setting for this problem.

   (a) Write function

   ```
   C = p3(X_train, Y_train, X_test,k)
   ```

   that performs kNN classification. Here `X_train`, `Y_train`, and `X_test` are the training samples, the true class of training samples, and the test samples, respectively, and `k` is the parameter for kNN classifier. The output column vector `C` stores the classes assigned to the test samples.

   (b) Using the function you wrote in part (a), report the test errors for $k = 1, 3, 5, 7$. Do you notice any trend? It is convenient to use function `p2` you developed previously.

   (c) Using the function you wrote in part (a), examine the confusion matrix for $k = 5$. It is convenient to use the function `p2`. Which two digits `(i,j)` are most frequently confused with each other? Is the confusion between these 2 digits symmetric (i.e. is digit `i` mistaken for digit `j` almost as often as digit `j` mistaken for digit `i`)?

4. (5%) Assume 2-class setting for this problem. Write a function

   ```
   C = p4(w,X)
   ```

   that takes vector of weights `w` (size $(d+1) \times 1$), matrix `X` (size $n \times d$) of samples to classify, and performs linear classification. The output column vector `C` (size $n \times 1$) stores the class assigned to the samples in `X`. Assume the bias weight is stored in `w(1)`, and the rest of the weights in `w(2:end)`. Positive samples should be assigned label `1`, and negative label `2`.

   For example, for input $X = \begin{bmatrix} 1 & 2 \\ -1 & -4 \\ 3 & 2 \end{bmatrix}$, $w = \begin{bmatrix} 2 \\ -1 \\ -3 \end{bmatrix}$, the output is: $C = \begin{bmatrix} 2 \\ 1 \\ 2 \end{bmatrix}$.

   This is because for the first sample, we get `w(1)+X(1,1)*w(2)+X(1,2)*w(3) < 0`, for the second sample, we get `w(1)+X(2,1)*w(2)+X(2,2)*w(3) > 0`, and for the last sample, we get `w(1)+X(3,1)*w(2)+X(3,2)*w(3) < 0`.

5. (10%) Assume 2-class setting. The goal of this problem is to explore how well one can find weights `w` for a linear classifier with random search.

(a) Write a function

`w = p5(X_train,Y_train, iterNum)`

that takes as an input training samples `X_train` (size $n \times d$) , their true class `Y_train` (size $n \times 1$), and the number of iterations `iterNum` to run the algorithm for. At each iteration, you should sample a random set of weights `w` for linear classifier and use them to classify the training samples. The output should be the best weights `w` (size $(d+1) \times 1$) you find, i.e. the weights that lead to the smallest error on the training samples. Functions `p2` and `p4` you wrote previously are useful for this problem. Useful matlab commands: `randn`. Do not use `rand` as it only generates positive numbers.

(b) Extract training and test data for digits 4 (class 5) and digit 9 (class 10) using function `p1`. Train on training data using `p5` for the case of 100, 1000, and 10000 iterations. Report and discuss training and test error for each case. Functions `p2, p4` are useful.

6. (15%) Assume 2-class setting for this problem.

(a) Write a function

`w = p6(X_Train,Y_Train, iterNum, wInit, alpha )`

that takes as an input training samples `X_Train`, their true class `Y_Train`, the number of iterations `iterNum`, initial weights for linear classifier `wInit`, and learning rate `alpha`. Your function should run logistic regression batch rule for `iterNum` iterations starting with `wInit`. It should output the final vector of weights `w`. The input/output matrix sizes are the same as in the previous problem. Although not required, for debugging it is useful to compute the loss function at each iteration. It should be decreasing (almost always) from one iteration to the next.

(b) Use the data from problem 5(b) but now train using `p6` with learning rate `alpha = 0.1`, starting with `w` set to a vector of `1`'s, and train for 30 iterations. Compute and compare the training and test errors to that in problem 5(b).

7. (5%) Assume multiclass setting for this problem. Write a function

`C = p7(W,X)`

that takes a matrix of weights `W` (size $m \times (d+1)$) for multiclass linear classifier, matrix of samples `X` (size $n \times d$) to classify, and performs linear classification. The output column vector `C` (size $n \times 1$) is the classes (from `1` to `m`) assigned to samples in `X`. As discussed in class, each row of W stores a discriminant function for one of the classes. The first entry in each row is the bias weight. Thus the first column of W stores all the biases.

8. (15%) Assume multiclass setting for this problem.

(a) Write a function

`W = p8(X_train, Y_train, iterNum, WInit,alpha)`

that takes as an input training samples `X_train`, their true class `Y_train`, the number of iterations `iterNum`, initial weights for linear classifier `WInit`, and fixed learning rate `alpha`. Your function should run multiclass Perceptron single sample rule for `iterNum` iterations starting with weights `WInit`. It should output the final matrix of weights `W`. Even though it is better to reshuffle training samples, do not reshuffle for ease of grading.

(b) Use multiclass digit training data `X_train` and `Y_train` to train using `p8` with learning rate `alpha = 0.01`. Start with `W` set to random, and run training for 100 iterations. Compute and report the training and test errors. Using the confusion matrix on the test data, find out which two digits are confused the most. Is this consistent with the results in Problem 3(c)?

9. (10%) Assume multiclass setting for this problem.

   (a) Write a function

   `W = p9(X_Train,Y_Train, iterNum, WInit, alpha )`

   that takes as an input training samples `X_Train`, their true class `Y_Train`, the number of iterations `iterNum`, initial weights for linear classifier `WInit`, and fixed learning rate `alpha`. Your function should run softmax single sample rule for `iterNum` iterations starting with `WInit`. The output is the final matrix of weights `W`. Do not reshuffle samples.

   (b) Use multiclass digit training data `X_train` and `Y_train` to train using `p9` with learning rate `alpha = 0.01`. Start with `W` set to random, and run training for 100 iterations. Compute and report the training and test errors. Compare the training and test errors with those in problem 8(b).

10. (15%) Assume multiclass setting for this problem.

    (a) Write a function

    `[net,valErr] = p10a(X_train,Y_train, H, regularizerWeight)`

    that takes as an input training samples `X_train`, their true class `Y_train`, number of hidden layers and units specified by row vector `H`, the weight of regularizer `regularizerWeight`. This function should trains a neural network using matlab function `patternnet`. The output is the trained network `net` and the validation error `valErr`. The number of columns in `H` is equal to the number of hidden layers, and the entries in `H` give the number of hidden units in the corresponding hidden layer. For example, if `H = [100,10]`, a neural network 100 units in the first hidden layer and 10 units in the second hidden layer should be trained. As in previous problems, `X_train` stores samples as rows and `Y_train` is a column vector of true labels.

    Brief tutorial on using matlab NN tools

    This summary should be sufficient, but you can also find multiple tutorials on the web.

    - Start with initializing NN as
      `net = patternnet(H);`
      Here `H` is exactly as described for `p10` function input.
    - Initialize the training, validation, and test folds ratio. For `p10` function, we do not need the test fold, only training and validation. Therefore, you should set training fold to 70%, validation fold to 30%, and test fold to 0%.
      ```
      net.divideParam.testRatio = 0;
      net.divideParam.valRatio = 0.3;
      net.divideParam.trainRatio = 0.7;
      ```
    - Set regularization strength with
      `net.performParam.regularization = regularizerWeight;`

- Call method `train` for training the network
  `[net,tr] = train(net,X,Y);`
  Matlab function `X` expects samples to be stored as columns, not rows, as in the rest of the assignment. Also, `Y` should be a matrix of targets, i.e. mutli-dimensional representation of the correct class, as in the lecture notes. Targets for each sample are stored as columns. For each sample, all entries in its target column are zero except there is a `1` in the row corresponding to the correct class. The output is the trained network `net` and the model parameters `tr` that you will need to use to find samples network used for validation.
- To classify any samples, use
  `Y = net(X)`
  Matrix `X` has samples stored as columns. The classificaiton result for sample stored in column `i` of `X` is stored in column `i` of `Y`. The row of the largest entry in the `i`th column of `Y` is the class the network assigns to the `i`th sample.
- To compute validation error, use `tr` output by the `train` function. Vector `tr.valInd` gives the indexes of data used for validation by the network. Put these samples through the network, and compute classification error.

(b) Write a function

`[err,CONF] = p10b(X_test,Y_test, net)`

that takes as an input samples `X_test`, their true class `Y_test`, trained neural network `net` and outputs the error rate and confusion matrix this network has on the input samples.

(c) Use multiclass digit training data `X_train` and `Y_train` to train a network using `p10a`. Use one hidden layer with `100` units. Use regularization strength 0.8. Report validation error. Using `p10b`, apply the trained network to the test data in `X_test` and `Y_test` and report test error. Compare the test error to that in problems 8(b) and 9(b).

(d) Use validation error returned by `p10` as a guide to search for the best network. Experiment with different number of units in the first and second hidden layer (you can go to more layers if you wish) and different regularization strengths. Choose as the best the network that gives the smallest validation error. Report the best network parameters you found (number of hidden layers, number of hidden units in each layer, regularization strength), and also the validation error. Now apply your best network to the test data and report the test error. Compare validation/test errors to that in part (b).

11. (Extra credit, 10%) Use provided matlab GUI `brushStrokes_gui` for data collection. Draw and save 5 images of each of the 10 digits (0,1,...,9). Images for digit `0` should be saved as `0_1.bmp`, `0_2.bmp`,...,`0_5.bmp`. Name convention for other digits is the same. A sample of how your digits should be sized and positioned are in `digitSamples.png`. Images should be saved in one directory. Submit the zipped directory through OWL by January 22.

12. (Extra Credit, 10%) Participate in competition on digit recognition dataset collected by students in this course. Our class competition is hosted by Kaggle `https://inclass.kaggle.com/c/digit-recognition`. From the above URL, load the data (`competitionDataX.mat`) and the function `fromMatrixtoCVS.m` that converts your matlab output into CVS format that Kaggle requires. You can choose to submit the results of any classifier you develop for

this assignment, including the extra credit problem 13. You can train your classifier on the full dataset (including the data in `X_test_full` and `X_train_full`).

In the written report, briefly explain what is the classifier you chose for competition submission. For example, you can simply say "I submit knn classifier on the full MINST data with k = 9". Also let us know which name you used for competition on the Kaggle web site.

Submit the matlab function for the classifier you used in the competition. It should be named

`C = p12(X_test)`,

Where `X_test` are the samples to classify stored as rows, and `C` is the vector of class labels. This function should run in a reasonable amount of time to be feasible to use in the provided gui `brushStrokes_gui.m`. If you need any data inside your `p12` function, you can load it inside the function with `load myFile.mat`. Submit any data files your function `p12` needs so that we can run it.

Optional: plug your classifier into function `recognize_digit` so that it can be used in the provided gui `brushStrokes_gui.m`.

13. (Extra Credit, 10%) Try to improve the multiclass digit classifier developed in this assignment. For example, you can add features, or use an ensemble of classifiers to make a decision. Describe your improvements and the resulting test error on samples in `X_test`. You can also experiment on the full dataset available in `A1_full.mat` and in this case report the test result on `X_test_full`. You cannot use any tools other than those already allowed in the other portions of this assignment. Namely, you are not allowed to use machine learning libraries other than `patternnet` allowed for problem 10. Your improved classifier should be invoked as

    `C = p13(X_test)`,

    where `X_test` are the samples to classify stored as rows, as usual, and `C` is a column vector of classes for samples in `X_test`. To load your data inside `p13`, use matlab `load myFile.mat` command. Submit any data files your function `p13` needs so that we can run it.

14. (Extra Credit, 20%) Instead of matlab buit-in function `patternnet`, write functions `p14a` and `p14b` that are exactly the same in functionality and input/output parameters as `p10a` and `p10b` but use your own implementation of neural networks. You can limit your network to one specific activation function and one specific loss function. The number of layers and hidden units in each layer should be unlimited. Compare the performance of your network to that in problem 10.