CSE 5523: Problem Set 5

Due via Carmen dropbox at 11:59 PM 11/8.

1. PROGRAMMING:

   (In general for programming problems you should hand in the following as separate files:

   - a copy of each program file you write,
   - a representative sample of each input file you use,
   - a representative sample of each output you produce.

   Unless otherwise noted, your programs should be as short as possible, and may be based on linear algebra and data analysis functions used in the lecture notes, but should not use higher-level packages or functions.)

   Download the 'Indoor User Movement Prediction' time-series data:

   https://archive.ics.uci.edu/ml/machine-learning-databases/00348/MovementAAL.zip

   This consists of 314 sequences of RSS (location) measurements and a final target indicator for each sequence. You should use the even numbered series of this dataset as a training partition and the odd numbered series as a test partition.

   (a) [15 pts.] Adapt the example tensorflow recurrent neural network from the lecture notes to predict the 'MovementAAL_target.csv' data of this dataset (for this problem, you are allowed to use tensorflow, but not higher-level RNN functions). Note that there is a single target (y value) for each entire series, not for each time step as is assumed in the sample code in the lecture notes. Your program should be named 'aal.py' and may be hard-coded to use the 'MovementAAL_RSS...csv' and 'MovementAAL_target.csv' files in the downloaded 'dataset' folder, assuming it is in the same directory as your program. Your program should output a csv file containing one dimension of predicted values for the target data.

   HINT: You should get about 65% accuracy on the training set using 20 hidden units after about 10 epochs with the keras Adam optimizer and a learning rate of 1.0.

   (b) [5 pts.] Train your classifier on your training partition, then run your classifier on your test partition and evaluate its accuracy. You may use the code at the end of this problem set for credible interval determination.

2. PROGRAMMING:

   You are told that the below function makes a good kernel function in an SVM:

   $$K(\mathbf{x}, \mathbf{x}') = (\mathbf{x}^\top \mathbf{x}' + 1)^3$$

   (a) [15 pts.] Write a program called 'svm-kern.py' that implements a support vector machine classifier using the above kernel. Your program should take the following as command-line arguments in the following order:

   i. a filename of a csv file containing variable dimensions of training data points,
   ii. a filename of a csv file containing the same dimensions of test data points minus the first column,

   and output a csv file containing one dimension (the first column of the input) of predicted values. Your program may be based on the example support vector machine code in the lecture notes.

   HINT: Your classifier should perfectly separate the training data. You may need to rescale the data to avoid overflow errors (these may unfortunately manifest as strange numbered errors from the optimizer).

(b) [5 pts.] Run your classifier and the simple inner-product SVM code from the lecture notes on the 'mileage-train.csv' and 'mileage-test.csv' data on the course web page. Evaluate the accuracy of your program on the 'mileage-test.csv' data as compared with inner-product SVM code in the lecture notes. Do you notice a statistically significant improvement for using the kernel? You may use the code at the end of this problem set for credible interval determination and code from the lecture notes for significance testing.

3. PROJECT:

   As noted before, your project may be in any area (vision, natural language, etc), may use any programming language you choose, and may use any data, even data used in previous projects or from data repositories or shared task challenges (e.g. https://archive.ics.uci.edu/ml/datasets.php), but should involve new code that you write to implement techniques you did not use previously and should therefore produce new results. You may work in groups, but in this case you should specify what portion of the project code will be produced by you and the other group members.

   (a) [6 pts.] Perform a trial run of your simplest machine learning technique (or a simpler technique from the lecture notes or problem sets if none of your project models are ready) on your dataset and report how long it takes on your currently available computational resources. If your available computer resources are not adequate to complete testing runs of your project in the allotted time, please scale back your project and describe the rescaled project below.

   (b) If your available computer resources are adequate for your project, and you were not asked to modify your project in feedback on the previous problem set, you may skip this sub-question (please report that you are skipping it for this reason). If your compute resources are **not** adequate for your project, or you **were** asked to modify your project in feedback on the previous problem set, please resubmit the following (as in the previous problem set):

      i. [3 pts.] Describe in just a few sentences what *problem* your project will solve.
      ii. [3 pts.] Describe in just a few sentences what *data* your project will use, including how many items (N) the data include, how many variables (K) are involved, and where your data will come from.

iii. [3 pts.] Describe in just a few sentences what *machine learning techniques* your part of your project will use.

NOTE FOR PROGRAMMING PROBLEMS:

You may use the following code for credible interval estimation:

```
import sys
import numpy
import pandas

numsamples = 1000

X = pandas.read_csv(sys.argv[1])                             ## read 0/1 scores

S = pandas.Series( numpy.random.beta( len(X[ X[X.columns[0]]==1 ]) + 1,
                                      len(X[ X[X.columns[0]]==0 ]) + 1,
                                      numsamples ) )         ## get 1000 samples
m = S.mean()                                                 ## get sample mean

for a in S:                                                  ## for each sample
  n = len( S[ abs(S-m) <= abs(a-m) ] )                       ## test as boundary
  if n==numsamples*.95: print( 'mean: ' + str(m) + ' +/-' + str(abs(a-m)) )
```