Problem Set 1

Due Sunday, April 30, 2017 at 11:55pm

How to Submit

Create one .zip file (**not** .rar or something else) of your code and written answers and submit it via `ilearn.ucr.edu`. Your zip file should contain `runq1.m`, `cvknn.m`, `knntest.m`, and a file of your written answers for problem 1. Please submit your written answers in a pdf or ascii text file, not an MS Word document.

Each file should include at the top (in comments if necessary)

- Your name & UCR student ID number
- The date
- The course (CS 171) & assignment number (PS 2)

Problem 1. [10 pts] The supplied function `learnlogreg` implements logistic regression, including regularization (`lambda` specifies the degree of squared regularization). If `lambda` is non-zero, it assumes that the first feature is the special constant feature of 1.

Write a function called `runq1` (takes no arguments) that tries logistic regression on the data in the file `phishing.dat`. This data consists of a set of features about websites' URLs (the first 30 columns) and whether the website is a phishing attempt (last column). More information on the features can be found at `https://archive.ics.uci.edu/ml/datasets/Phishing+Websites` Because the features are discretized to 0 or 1 and are thus already standardized, they should not be normalized. Treat the entire dataset as training data.

Your code should load the data and try both linear and quadratic models (by augmenting the feature space). It should try different regularization strengths.

Have your *code* pick a particular set of features and regularization in some principled way.

Report which features and regularization strength your code selected. Explain why you decided to do it this way.

Problem 2. [15 pts] Two classification data sets are included with this problem set: `iris.dat`, and `vert.dat`. The former are measurements about irises (taken in the 1930s). There are four attributes (all in centimeters), each given by one of the first four columns. In order, they are (1) the sepal length, (2) the sepal width, (3) the petal length, and (4) the petal width. The last column is the class or target (0 for Setosa, 1 for Versicolor, and 2 for Virginica).

The second data set consists of measurements about the vertebral column. In particular, the first six columns are the attributes (pelvic incidence, pelvic tilt, lumbar, lordosis angle, sacral slope, pelvic radius, and grade of spondylolisthesis). The last column is the category of patient (0 for disk hernia, 1 for spondylolisthesis, and 2 for normal). Just as for the iris data set, the examples have been randomly ordered. Do not further scramble them (so that we can easily check your solutions). For both data sets, you should use all attributes.

The supplied function `runknn` does four things:

1. loads and splits a data set into training and testing;
2. z-normalizes the training (and applies the same normalization to the testing);
3. splits the training into training and validation and runs cross-validation for k-nearest neighbor; and
4. checks the accuracy of the result on the testing set.

The supplied function `runq2` executes `runknn` for both datasets.

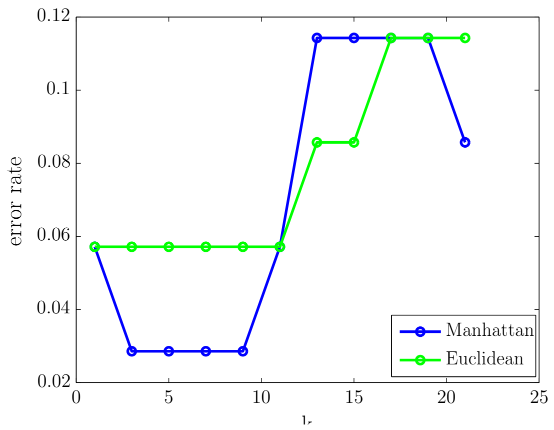
However, it is missing two critical pieces: `cvknn` and `knntest`. Your task is to supply these functions.

The former has signature `[k,lnorm] = cvknn(TrainX,TrainY,ValidX,ValidY,maxk)`. It should take in the training and validation data sets and the maximum k value to check. It should use cross-validation to select the best combination of k (among all of the odd values between 1 and maxk, inclusive) and distance metric (between either Euclidean or Manhattan distance). It should return the best pair. The returned `lnorm` should be 2 for Euclidean and 1 for Manhattan. *Additionally* your code should also draw a plot of the validation-set error rate versus k for both of the distance metrics (an example is shown below).

The latter function has signature `[err,C] = knntest(TrainX,TrainY,TestX,TestY,k,lnorm)` and takes in a training and testing set, along with k and the distance metric (encoded as above). It should return the testing error for k-nearest-neighbor, as well as a confusion matrix, C. A confusion matrix records how often a label of one type was classified as a different type. In particular, the i-j element is the fraction of time the label was reported to be i, when it was actually j.

As an example, on the Iris data set, the solutions report the text and plot below.

```
Iris:
chosen: k=3 and lnorm=1
testing error = 0.04
confusion matrix =
0.4600         0         0
0    0.2400         0
0    0.0400    0.2600
```


error rate
Manhattan
Euclidean