# STAT 339: HOMEWORK 1 (CLASSIFICATION AND REGRESSION)

DUE ON BLACKBOARD BY START OF CLASS ON MON. FEB. 6

**Instructions.** Turn in your writeup and code to Blackboard as an archive file (e.g., `.zip`, `.tar`, `.gz`) by the start of class on Friday, 2/17.

As always, you may use any language you like to do this assignment — the tasks are stated in a language-neutral way. You may also use any typesetting software to prepare your writeup, but the final document should be a PDF. LaTeXis encouraged; a reproducible research format in which code is embedded into the document (e.g., knitr, RMarkdown, Jupyter or IPython Notebook) is even more encouraged.

All data files referred to in the problems below can be found at

`http://colindawson.net/data/<filename>.csv`.

0. Create a project directory for this homework to keep your files organized, and set it to be your working directory. Your archive will be this directory.

1. **K-nearest-neighbors Classifier.** The files `S1train.csv` `S1test.csv`, `S2train.csv` and `S2test.csv` contain synthetic binary class data with 2 real-valued features, generated from one of two data models (S1 or S2). The data distributions for each class in scenario 1 (S1) are simple (single bivariate Gaussians), whereas the distributions in scenario 2 (S2) are more complex. The training sets for each scenario are plotted below:

   In each file, the first column is the class label (1 or -1), and the other two columns are the input features. Your task is to implement a $K$-nearest-neighbors classifier, compute the training, 10-fold cross-validation, and test misclassification rates for various choices of $K$ on each of the two scenarios (choose odd values of $K$ to avoid ties), plot the results, and comment on how the complexity of the class-conditional distributions affects the optimal choice of $K$. Some suggested steps are below, but

*Date*: Last Revised: February 2, 2017.

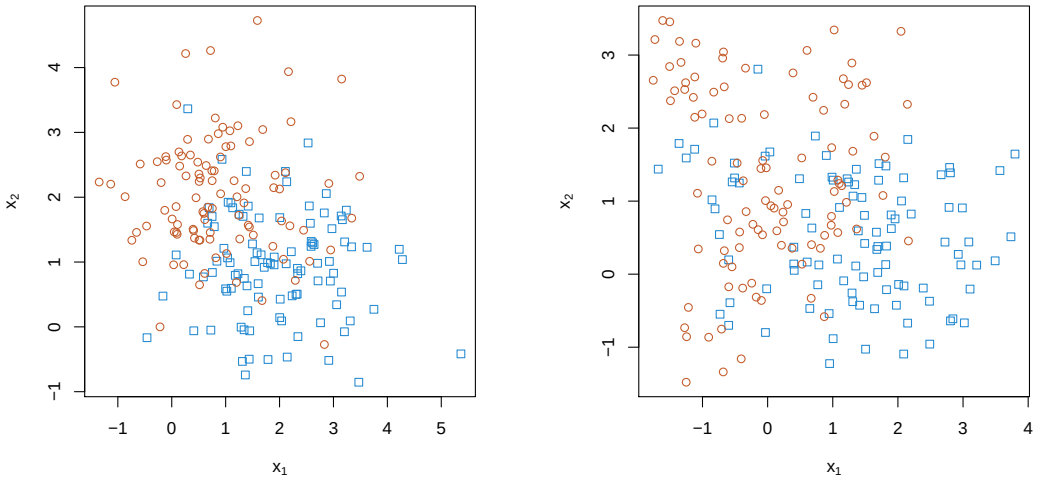


FIGURE 1. (left) Training data for S1; (right) Training data for S2

feel free to implement the above however you see fit. It is worth making some effort to avoid explicit `for` loops where possible, instead using built-in vector and matrix arithmetic methods, in the interest of run time. However, don't spend too much time optimizing for speed.

(a) Write a function that computes the Euclidean distance between two particular feature vectors (cases).

(b) Write a function that computes the Euclidean distance between a particular case and each case in a training set, returning the results as an array of distances.

(c) Write a function that finds the indices in the training set with the smallest $K$ distances to the case currently under consideration (where $K$ is provided in an argument to the function).

(d) Write a function that returns the "majority class" for the $K$ nearest neighbors (i.e., that classifies the case under consideration).

(e) Write a function that classifies all of the instances in one data set using a second dataset (possibly the same one) as the training data, returning a vector of class predictions.

(f) Write a function that takes a vector of classifier labels and a vector of ground truth labels and returns the misclassification rate.

(g) Write a cross-validation function that takes the number of folds (I am avoiding using $K$ for this given the context), the number of nearest neighbors to consider, and the training set as inputs, and returns the average validation set misclassification rate over all folds. Your function should randomize the order of the observations before selecting the folds (you may want to give the option to set the seed so you can reproduce your results).

(h) Calculate and plot training, cross-validation, and test misclassification rates for each of several values of $K$ (that is, the number of nearest neighbors), commenting on the difference between the two scenarios.

2. **K-nearest-neighbors Classifier for Digits.** (Adapted from BRML Ex. 14.1) The files `Train5.csv`, `Train9.csv`, `Test5.csv` and `Test9.csv` contain pixel brightness values for $28 \times 28$ pixel grayscale images of handwritten digits (5s and 9s). Each row of each data file represents an image, with the 784 columns representing the brightness (0-255) of a particular pixel in the corresponding $28 \times 28$ image. (If you are using MATLAB/Octave, the data is provided as a single `.mat` file accompanying the BRML book — the data will be read in as a struct with 4 fields, one correpsonding to each digit by train/test set. Note, however, that in that dataset, the matrices are transposed so that the rows represent pixels and the columns represent images)

(a) Reshape one or two images from each set into a $28 \times 28$ matrix and visualize it to verify that you have the format correct. (When implementing KNN, however, you will want to keep each image as a flat vector.)

(b) Apply your $KNN$-classifier and 10-fold cross-validation procedure to this data, again plotting the training, CV, and test set misclassification rate for various (odd) values of $K$. Be sure to randomize the order of the observations before selecting folds.

3. **Implementing a Regression Solver From Scratch.**

(a) (Adapted from FCML Ex. 1.2) Write a function that takes a dataset consisting of ordered pairs of the form $(x_n, t_n)$, $n = 1, \ldots, N$ (you can decide on an appropriate data structure representation for the input; one standard possibility would be an $N \times 2$ matrix/array) and returns the OLS coefficients $(w_0, w_1)$ for a linear model of the form $t_n = w_0 + w_1 \cdot x_n$. Don't forget a docstring!

(b) (Adapted from FCML Ex. 1.6) Verify that your function works using the data in `womens100.csv` containing winning race times from the Olympic women's 100m event in various years. Plot the data and the resulting line and compare the coefficients returned by your model to the coefficients returned by a pre-packaged regression solver (see also Sec. 1.2.1 of the textbook).

(c) The data ends with the 2008 olympics. Use your model to "predict" the winning times for the 2012 and 2016 olympics. Now look up the actual times; how does the model do?

(d) Modify your solver function to allow multiple input variables (e.g., columns). Include an option to allow the user to extend the data provided to a polynomial basis of any desired order $d$ (i.e., to append columns to the input matrix consisting of $x^2, ..., x^d$) before fitting the model.

(e) (Adapted from FCML Ex. 1.9) Apply your function to the data in `synthdata2016.csv` using $d = 3$ (the data file as it is consists simply of $(x, t)$ pairs). Plot the data and the fitted cubic curve.

(f) Modify your solver to allow the user to specify a regularization parameter, $\lambda$, to do ridge regression. Compare results for various choices of $\lambda$ using a cubic model on the `synthdata2016.csv` data.

4. **Cross-Validation With Regression.**

(a) Write a function that performs $K$-fold cross-validation (let the value of $K$ be specified by the user) for the OLS linear regression model based on a given target vector, input matrix, and regularization parameter $\lambda$, returning the mean and standard deviation of the mean squared error metric applied to both the training and validation sets, where the mean and standard deviation are taken across folds. Be sure to randomize the order of the observations before breaking up the data into folds (being careful to keep input variables and target variable together), in case the provided data is sorted in some way. You can do the fitting of the model either with the function you wrote above or with a pre-written function; your choice.

(b) Write another function that takes a dataset with one input variable and one target variable and performs $K$-fold cross validation (where $K$ is an input to the function) using polynomial regression models ranging from order 0 (constant) to $D$ (where $D$ is specified by the user). The function should return squared errors on the training and validation sets, with mean and standard

deviation taken across the $K$ folds. You will presumably want to call the function you wrote in 4a.

(c) Apply the function you wrote in 4b to both the `synthdata2016.csv` and the `womens100.csv` datasets to identify the optimal polynomial order. Try both $K = 10$ and $K = N$ (i.e., leave-one-out) cross-validation. Plot the mean training and validation errors for each polynomial order (optional: include standard deviation error bars) and comment on what the plots show.

(d) For the `womens100.csv` dataset, we can use the 2012 and 2016 times as a true test set. Compute squared prediction errors for each of the polynomial degree models (fit on the full dataset) for 2012 and 2016. Did cross validation give the best model on this metric?

(e) Now use 10-fold cross-validation and a grid search to find a good combination of values of $\lambda$ and $D$ for a ridge regression model. How does generalization performance (on the true test set) for the chosen model and regularization parameter compare to that for the best OLS regression fit?

5. **Deriving a Weighted Least Squares Regression Fit.** (adapted from FCML Ex. 1.11) (No coding here) Consider a generalization of the OLS loss function that applies (known) weights to each observation, given by:

$$\mathcal{L}(\mathbf{w}; \mathbf{x}, \mathbf{t}) = \frac{1}{N} \sum_{n=1}^{N} \alpha_n (t_n - \mathbf{w}^\mathsf{T} \mathbf{x}_n)^2$$

(or, equivalently, by

$$\mathcal{L}(\mathbf{w}; \mathbf{x}, \mathbf{t}) = \frac{1}{N} (\mathbf{t} - \mathbf{X}\mathbf{w})^\mathsf{T} \mathbf{A} (\mathbf{t} - \mathbf{X}\mathbf{w})$$

where $\mathbf{X}$ is the matrix whose $n$th column consists of $\mathbf{x}_n$, and $\mathbf{A}_{nn} = \alpha_n$, $n = 1, \ldots, N$ and $\mathbf{A}_{mn} = 0, m \neq n$.)

Derive an expression for the estimated weight vector $\hat{\mathbf{w}}$ that minimizes this loss.