CSC 120 (R Section, L0201), Spring 2015 — Assignment #2

Worth 10% of the course grade. Due by the start of class on March 17, by email (see end of this handout). This assignment may be handed in late, with a 20% penalty, by start of class on March 20. Assignments will not usually be accepted after that. Contact the instructor as soon as possible if you have a legitimate excuse (eg, documented illness) for handing in the assignment late.

This assignment is to be done by each student individually. You may discuss it in general terms with other students, but the work you hand in should be your own. In particular, you shouldn't leave a discussion with someone else with any written notes (either paper or electronic).

In this assignment, you will implement a "nearest neighbor" classifier, and apply it to two data sets — the first an artificial data set that I generated, and the second a data set on heart disease that I adapted from one at the UC Irvine repository of machine learning data sets (found at `https://archive.ics.uci.edu/ml/datasets/Heart+Disease`).

A classifier is a method for predicting the "class" of an item from the values of a set of "input" variables. For example, one might try to predict whether or not someone plays basketball from their age and height. The classifier has available a set of "training cases" in which both the inputs and the class are known. Its task is to predict the class in "test cases" where only the inputs are known. A nearest neighbor classifier predicts by guessing that the test case has the same class as the "nearest" training case.

For this assignment, you will be provided with the actual classes of each test case, as well as of the training cases, but you should look at the classes for test cases only at the end, to see how well the nearest neighbor method worked.

The programming aspects of this assignment include reading "data frames" and then modifying them, and more practice in how programs can be organized as several functions that do parts of the overall task.

The data for this assignment is on the course web pages, at the following URLs:

data set 1 training inputs	`http://www.cs.utoronto.ca/~radford/csc120/a2trainx1`
data set 1 training classes	`http://www.cs.utoronto.ca/~radford/csc120/a2trainy1`
data set 1 test inputs	`http://www.cs.utoronto.ca/~radford/csc120/a2testx1`
data set 1 test classes	`http://www.cs.utoronto.ca/~radford/csc120/a2testy1`

data set 2 training inputs	`http://www.cs.utoronto.ca/~radford/csc120/a2trainx2`
data set 2 training classes	`http://www.cs.utoronto.ca/~radford/csc120/a2trainy2`
data set 2 test inputs	`http://www.cs.utoronto.ca/~radford/csc120/a2testx2`
data set 2 test classes	`http://www.cs.utoronto.ca/~radford/csc120/a2testy2`

You should read the files of training and test inputs using `read.table`, with the `header=TRUE` option, which will give you a data frame. You should read the files of training and test classes (which have just one value for each case) with `scan`. For the first data set, the classes are 0 or 1, and can be read as numbers, giving `scan` just one argument that is the name of the file to read. For the second data set, the classes are "present" or "absent", indicating whether or not the person with the corresponding inputs did or did not have heart disease. These non-numeric classes can be read by calling `scan` with a second argument of `character()`, to indicate that the items are strings rather than numbers.

A nearest neighbor classifier requires a “distance” function, that determines how “far apart” the inputs for two cases are. For this assignment, the inputs are allowed to either be numbers, or non-numeric indicators (such as sex, which is “M” or “F”). The inputs for a single case will be represented by an R list, which can hold both numbers and other data. We could define “distance” in various ways, but for this asignment, we will define the distance between two cases (represented by two lists) as the sum of contributions to the distance from each pair of corresponding elements from the two lists. For numerical elements, the contribution to the distance will be the absolute value of their difference. For non-numeric elements, the contribution to the distance will be zero if they are equal and one if they are not equal. For example, the distance between `list(5,7,"abc")` and `list(4,10,"xyz")` is $1 + 3 + 1 = 5$ and the distance between `list("abc",1.3)` and `list("abc",1.5)` is $0 + 0.2 = 0.2$.

You should define a function called `distance` that computes the distance between two items, represented by two lists.

To classify a test case, a nearest neighbor classifier first finds the training case that has the smallest distance to the test case. You should define a function called `nearest_neighbor` that takes two arguemnts, a data frame of training case inputs, and a list with the input values for a single test case, and returns the index of the training case in the data frame that is closest to the test case. (In case of a tie, pick the earlier training case.)

Finally, you should define a function called `classify` that has as its arguments a data frame of training case inputs, a vector of training case classes, and a data frame of test case inputs, and which returns a vector of guessed values for the classes in the test cases, found by guessing that the test case has the same class as the nearest training case.

You should also write two R scripts, each of which starts by using `source` to read in the definitions of the functions mentioned above. The two scripts should apply these functions to the two data sets. There are some differences in what you should do for each data set, however.

The first data set, which has 50 training cases and 30 test cases, with three input variables, is small enough that you can look at what your classifier is doing, to see if it is doing the right thing. To help with this, you should make a scatterplot of the inputs for the training cases. There are two numerical inputs, which you can use for horizontal and vertical coordinates. The third input is “A” or “B”, and can be used as the plotting symbol for the scatterplot (which you can set by giving these values as the `pch` option to `plot`). You can show the class of the training case by the colour of the symbol (set with the `col` option to `plot`). You should use this plot to help debug your program, and also hand it in (see below).

For this dataset, you should also print out the class that the classifier guessed for each test case, and the actual class. Finally, you should print the fraction of test cases for which this guess was correct.

The second data set has 150 training cases and 120 test cases, with 13 input variables. You should print the fraction of test cases for which the nearest neighbor classifier correctly guesses the class, using the data as read.

You should also try rescaling some of the input variables, since some of them have much greater variation than others, which means that the variables with small variation are almost ignored when determining which is the nearest neighbor. In particular, you should try rescaling `age` by dividing by 10, and rescalsing `BP`, `chol`, and `rate` by dividing by 100. You should report the fraction of guesses that are correct using these rescaled variables. Be sure to recale the inputs for both the training cases and the test cases! To help with this, you may want to write a function for rescaling, which you can define in the script file for this data set (since it is specific to how to handle this data set, not generally useful).

Some R notes: The `abs` function in R returns the absolute value of its argument. You can check whether something is numeric with the `is.numeric` function. Logical values can be converted to numbers (with `FALSE`= 0 and `TRUE`= 1) using `as.numeric` (though this is often done automatically anyway). If you have extracted one row of a data frame, you can convert it to a list of the variable values with `as.list`.

To you submit your assignment, send an email to `radford@cdf.utoronto.ca`, with subject line "A2 your-family-name, your-given-name". The body of the email can be blank (but you can include a note if you like). You should attach three files. The first file should be the .R file containing **only** the definitions of functions for nearest-neighbor classification that are not specific to any data set, with suitable documentation on what the functions do. This file **must not** contain any references to any specific data set. The second file should be the .html file created by `knitr::spin` when you ran your script to read the artificial data set, plot it, apply nearest-neighbor classification, and display the results and classification accuracy. The third file should be the .html file created by `knitr::spin` when you ran your script to read the heart disease data set, apply nearest-neighbor classification to it, both without and with rescaling of some inputs, and display classification accuracy without and with rescaling. Your two script files should contain brief comments where appropriate that describe what they do.