Neural Networks & Machine Learning: Worksheet 4

Aim

This worksheet introduces supervised learning and also R. As a result of completing the tasks below, you will

- gain practical experience in running and interpreting *k* Nearest Neighbour analyses,
- get experience of rearranging data as required by the functions you intend to use,
- practice the use of very simple toy data for developing code and for testing it, before applying it to process more complex data,
- have R installed on your computer,
- gain experience with using R's help system and documentation and using that to develop code,
- understand some essentials of R's basic data types and R's plotting facilities.

Please remember to develop your work in files (scripts and functions), and make sure you save these where you can access them in the future.

This worksheet is not marked and no submission is required.

Tasks

The key overall goal of the tasks below is to use apply K Nearest Neighbour (KNN) on the sparrows dataset, and to investigate its predictive performance. This requires some tools, and some tasks below guide you to developing these. Please follow the specifications in the tasks precisely, as that will ensure that the functions you develop will enable you to smoothly and flexibly explore KNN and its performance in the final task. Specifically, use the portions of code given as part of tasks exactly as they are provided, without making changes. If you do not fully understand any piece of the provided code, please ask.

This work requires a working Netlab installation and the sparrows dataset. Please refer to worksheets 2 for installing Netlab and and worksheet 3 for the sparrows dataset. As a brief recap, you need to tell Matlab where to search for the files containing libraries (toolboxes) and data using the `addpath` command. If you write a small script, containing the relevant `addpath` statements, you can run that at the start of a new Matlab session to quickly and reproducibly set up your working environment.

1. **Introduction to Netlab's KNN System.** Read the documentation of the `knn` function for constructing KNN models, and the `knnfwd` function for using such models to predict outputs (i.e. classes) for new inputs.

   Use the `knn` function to construct KNN models for the following toy dataset:

x	*y*	class
0	0	0
2	2	1
4	0	0
3	3	1

   The input matrix (`tr_in`) is comprised of the first two columns of the table above. The outputs (`tr_targets`) need to be provided in the following matrix form:

$$\texttt{toy\_targets} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \\ 0 & 1 \end{pmatrix}$$

This is the "One-Hot" encoding of categorical variables (called "1 of N" in the Netlab documentation). It is defined as a matrix with one column for each class and one row for each data item. In each row, exactly one element is 1 while all others are 0. The column containing the 1 corresponds to the class that the data item belongs to. The toy dataset has two classes, 0 and 1, and the first column in the matrix above corresponds to class 0, and the second column corresponds to class 1.

The other parameters are `k`, the (eponymous) number of nearest neighbours to consider, `nin`, the dimension of the inputs (2, as the toy dataset has two input columns, x and y), and `nout`, the number of classes (2, as the toy dataset has two classes, 0 and 1)

Construct models with $k = 1$ and $k = 3$, respectively. Then, use the `knnfwd` function to predict the class for points $p_1 = (0,1)$ and $p_2 = (1,0)$ using both models. Notice that `knnfwd` returns the classes as 1 and 2, so you need to subtract 1 from the values returned in `l`.

Do both models predict the same class for these points? Also, predict the class for the input data. Which model predicts all inputs correctly? Do you think the model predicting all inputs correctly is necessarily the better one?

2. **KNN Analysis of the Sparrow Data.** Use the methods introduced above to construct models for the sparrows dataset. You will need to build a "One-Hot" matrix from the single column in `sparrows_labels`. Recall or review the Matlab functions `not` (or, equivalently, the unary operator `~`) and `logical` which are useful for this purpose. Also remember that the `nin` parameter must be set to the input dimension, i.e. the number of columns in the `sparrows` matrix. Check the number of correct predictions with $k = 1$ and $k = 3$, and reflect on the results, by discussing them with other participants in the lab if you like.

3. **Computing the Elements of a Confusion Matrix.** Write a function that takes the true outputs and the predicted outputs as parameters, and returns the numbers of true positives (TP), true negatives (TN), false positives (FP) and false negatives (FN).

```
function [tp, tn, fp, fn] = confusionAnalysis(trueOutputs, predictedOutputs)
    % insert code here to place the correct values in the
    % output variables tp, tn, fp and fn.
end
```

4. **Accuracy.** Write a function that computes accuracy given true and predicted outputs. You should use your `confusionAnalysis` as follows:

```
function a = accuracy(trueOutputs, predictedOutputs)
    % use confusionAnalysis to get numbers of true positives,
    % true negatives, false positives and false negatives.
    [tp, tn, fp, fn] = confusionAnalysis(trueOutput, predictedOutputs);
    % use these numbers to assign the accuracy value to the
    % variable a below.

end
```

5. **Test Your Functions for Characterising Performance.** Create two variables holding the following true and predicted labels:

```
yTrue = [1, 0, 0, 1, 0, 1, 0, 1, 1, 1];
yPredicted = [1, 0, 1, 1, 1, 1, 0, 1, 1, 0];
```

Use paper and pen to find the confusion matrix, by counting the numbers of true positives, true negatives, false positives and false negatives, and calculate the accuracy based on these counts. Check that the functions that you wrote produce the correct values.

Jan T Kim, j.t.kim@herts.ac.uk 22 March 2020

We will start to use R in the next worksheet, and the tasks below provide guidance for setting up R and familiarising yourself with it. I recommend that you install and explore R in your own time, i.e. not during a lab, and that you discuss your experiences in the next lab session.

1. **Set up R on your computer.** Visit the R project website `https://www.r-project.org/`. Take a look around on the start page and if you're curious, explore some of the resources provided or linked to there. Read the "Getting Started" section and the "Download and Install R" instructions on a Comprehensive R Archive Network (CRAN) mirror, e.g. `https://cran.ma.imperial.ac.uk/`.

   Consider the following before starting to install R:

   - **Use general good practice with software installation**, e.g. before installing, save any valuable work, close / finish running other programs, make sure your computer has mains power or is well charged, and that you are not too exposed to distractions or disruptions.
   - **If you already have a version of R installed, it is recommended that you use your existing installation for now.** If you consider upgrading, be mindful that this may impact R code you have already written. Your code may not work with the new R version without modifications.

   R is installed and used by a large user base, and as a result, the installation process is generally very safe. This advice is to make sure you proceed with your installation based on a suitably informed decision. If you have any questions or concerns, please do not hesitate to raise these before proceeding.

   You may also use RStudio Desktop, available from `https://rstudio.com/`.

2. **Overview of R.** Start R and run the `help.start()` function. This should bring up a browser page showing a directory of documentation. Familiarise yourself with

   - the R Language Definition,
   - the Introduction to R,
   - the Packages page, and specifically the packages "base", "graphics", "utils" and "stats",
   - The R Data Import/Export documentation, specifically the "Spreadsheet-like data" section.

   Scan through the documents, read some paragraphs that attract your interest, and try out a few R functions when checking out the packages. The aim here is to get a first impression of R and its documentation, and to begin building a mental map that will enable you to identify and retrieve the information you need about R increasingly quickly.

3. **Vector operations.** Given vectors $\boldsymbol{x} = (1,2,3)^T$ and $\boldsymbol{y} = (2,1,-3)^T$, write R code to compute the dot product

   $$\boldsymbol{x} \cdot \boldsymbol{y} = \sum_{i=1}^{3} x_i y_i.$$

   Review the documentation of the `sum` function, and chapter 2 of the Introduction to R for this task. Have you done something similar, e.g. in another language? How does that compare to R?

4. **Matrices.** Study the documentation of the `matrix` function and construct two matrices

   $$\texttt{m1} = \begin{pmatrix} 1 & 2 \\ 4 & 5 \end{pmatrix}, \quad \texttt{m2} = \begin{pmatrix} 1 & 0 \\ 0 & 2 \end{pmatrix}$$

   Consult the documentation of the function `t` and infix operator `%*%` in the base package, and use these to compute their transposes, their sum, and their product. Verify that $\texttt{m1} \cdot \texttt{m2}$ is different from $\texttt{m2} \cdot \texttt{m1}$). Can you use transposition to compensate for switching the order in which you multiply the matrices?

   Also try to use the `%*%` operator to compute the inner product of two vectors, e.g. as requested in Task 3.

5. **Plotting.** Review the documentation for the functions `plot` and `points`. Use these functions to plot a graph of the function

$$f(x) = x^2 - 2x.$$

Plot the graph as a line, and to use the `points` function to highlight the minimum of the function with a red dot.

6. **Reading and plotting data.** Download the `disastersim01.csv` file from Canvas. Consult the documentation for the `read.table` function and its variants. Use the appropriate function to load the data into a data frame. Try to generate some plots resembling those shown for the “Parachuted Water Supplies” example in the introductory lecture.