Seminar Week 2: Introduction

In this first lab class we will be looking at the ingredients of machine learning and thinking about classification. You should work through the exercises below in your own time. If you don't manage to complete them all in the lab class itself, make sure that you work through them all as best you can before next week's class. Feel free to discuss the questions with anyone else in your lab group or work with someone to complete them.

You may want to begin by briefly reviewing the first two lectures, to remind yourself of what we covered there.

1. Have a go at **a short quiz**. You can have as many goes as you like to get the answers right, but it is important to make sure that you understand the key ideas here.

2. In the first lecture we looked at the **task mapping** encoded in a model. A machine learning model is understood as encoding a mapping from a space of instances to a space of labels. An instance represents an object of interest, whilst a label represents a possible output. In the following systems, what are the objects of interest and what kinds of label are used as output?

   - an email spam filter
   - a program that learns to tell whether a photograph contains a face
   - a program that learns to tell whether a photograph is of a particular person
   - a system for predicting the price of houses
   - a system for targeting supermarket customers with information about products that they might wish to buy

3. One particular issue that we looked at was that of **assessing classifier performance** (classification and evaluation of classifiers quiz goes back over some of this). We saw that there are various measures of performance that we might use to evaluate a binary classifier. Two such measures that are widely quoted in the literature are precision and recall. The two measures tell you rather different things about how good you are at predicting the + instances. Consider the two confusion matrices given below and imagine that they represent the performance of two classifiers on the same test set.

	Predicted +	*Predicted –*	
Actual +	25	0	25
Actual –	25	50	75
	50	50	100

	Predicted +	*Predicted –*	
Actual +	1	24	25
Actual –	0	75	75
	1	99	100

(a) What is the overall accuracy of each classifier? Do they differ much?

(b) How do they differ in terms of precision and recall? Calculate both measures for each contingency table to see.

(c) Can you think of a practical application where you would prefer the first classifier to the second? What about the other way around?

4. The second lecture looks at the **(binary) classification task**. As an introduction to some of the issues involved, you should go to the following site (caution: this online teaching aid site is a work-in-progress): http://users.sussex.ac.uk/~nq28/WebMLTeachingAid/pages/perceptron.html.

   - The figure in the middle panel shows 80 instances (data points). What do you see? Make sure that you understand what is displayed. Ask if it is not clear.
   - Press the "Run" button. This shows a process of learning a decision boundary to separate red instances from blue instances.
   - The decision boundary depends on a set of weights. These weights effectively draws the straight line through the two dimensional space that you see displayed, dividing it in two.
   - The process of learning a decision boundary that is shown here is based on a single layer perceptron algorithm. We will cover the single layer perceptron and how it works later in this module.
   - You can add more blue or red data points and see how the decision boundary changes. You can also *upload a new dataset* in a comma separated value with the heading: x1,x2,class. For example:

x1,	x2,	class
0.7,	0.6,	1
-0.7,	0.6,	-1
-0.7,	-0.6,	1
0.7,	-0.6,	-1

What do you observe with regards to the learning process?

5. Start running a perceptron classifier **with no setup** (unfortunately, it requires you to have *a Google account*). The following notebook (https://colab.research.google.com/drive/1_Omq2PzvXW9Tk_mM_s31GMZizUzvr-A-#scrollTo=SPaKc8F8il6F&forceEdit=true&offline=true&sandboxMode=true) run right from your browser (Chrome and Firefox are recommended), thanks to Colaboratory. It is a Jupyter notebook environment that requires no setup to use and runs entirely in the cloud. This notebook calls a single layer perceptron algorithm, which learns to classify the instances. The classifier code is contained in the **slperceptron** function. Take a look, but don't worry if you don't completely understand the code; as said, we will cover the single layer perceptron and how it works later in this module. **Note**: *You can trust the author of the notebook.*

6. Machine learning **toolboxes**. You will be using a toolbox for your assignments, which toolbox is up to you. It is good to start getting familiar with one of them:

   - Perceptron in Python (**Scikit-learn** Toolbox): http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Perceptron.html
   - Perceptron in Matlab (**Neural Network** Toolbox): https://www.mathworks.com/examples/matlab/community/22553-perceptron-learning-rule
   - Perceptron in Julia (**educational** Toolbox): https://github.com/kirnap/Machine-Learning-in-Julia
   - Perceptron in C++ (**Shogun** Toolbox): http://www.shogun-toolbox.org/api/latest/Perceptron_8cpp_source.html
   - Perceptron in Java (**Weka** Toolbox): http://weka.sourceforge.net/doc.dev/weka/classifiers/functions/MultilayerPerceptron.html
   - ...