159.372 Artificial Intelligence
Lab 2: Dealing With Data

Stephen Marsland

1 Introduction

Computers are designed to deal with data, and this course in particular is about precisely that. However, actually doing stuff with data is often a bit of a pain. In this lab we are going to use a prepared dataset, and even that is a bit annoying; real data tends to be even worse, since it often has non-numeric values, missing data, and parts that are just plan wrong.

Before using machine learning algorithms, it is almost always a good idea to play with the data a bit yourself, compute a few things about it, and make some plots. And that is what we are going to do today. You did a little bit of this last week with the Iris dataset.

2 The UCI Machine Learning Repository

As people like to compare the results of different algorithms on standard datasets, there is a need for various datasets to be held and made available. The University of California at Irving (UCI) has made their repository of data available as a service for researchers in machine learning: the UCI Machine Learning Repository at `http://archive.ics.uci.edu/ml/`. This website holds lots of datasets that can be downloaded and used for experimenting with different machine learning algorithms and seeing how well they work. By using these test datasets for experimenting with the algorithms, we do not have to worry about getting hold of suitable data and preprocessing it into a suitable form for learning by removing the errors, etc.

3 The Pima Indian Dataset

The `Pima` dataset from the UCI repository provides eight measurements of a group of American Pima Indians living in Arizona in the USA, with a classification concerning whether or not each person had diabetes. Download the data, and note that there is a file inside the folder giving details of what the different variables mean.

Read about the data. What do the variables tell you about the people?

Once you have downloaded it, import the relevant modules (NumPy to use the array methods, and PyLab to plot the data) and then load the data into Python. This requires something like the following:

```
import os
import pylab as pl
import numpy as np

>>> os.chdir('/Users/srmarsla/Book/Datasets/pima')
>>> pima = np.loadtxt('pima-indians-diabetes.data',delimiter=',')
>>> np.shape(pima)
(768, 9)
```

where the path in the `os.chdir` line will obviously need to be changed to wherever you have saved the dataset. In the `np.loadtxt()` command the `delimiter` specifies which character is used to separate out the datapoints. Start by checking out the size of the data (using the `np.shape()` method), which tells us that there are 768 datapoints, arranged as rows of the file, with each row containing nine numbers. These are the eight dimensions of data, with the class being the ninth element of each line (indexed as 8 since Python is zero-indexed). This arrangement, with each line of a file (or row of an array) being a datapoint is the standard one in machine learning.

You should have a look at the dataset. Obviously, you can't plot the whole thing at once, since that would require being able to visualise eight dimensions. But you can plot any two-dimensional subset of the data. Have a look at a few of them. In order to see the two different classes in the data in your plot, you will have to work out how to use the `np.where` command as you did last week. Once you have worked that out, you will be able to plot them with different shapes and colours. Remember that the `pl.ion()` command ensures that the data is actually plotted, while the `pl.show()` command is only required if you are using Eclipse, and ensures that the graph does not vanish when the program terminates.

Can you find a linear separation between these two classes with these features? What about if you consider combinations of the features?

Now split the data into different sets. We could do this using training, testing, and validation sets. *In fact, do this to make sure you can.*

And then split it into 10 subsets, and use the idea of cross-validation to provide a variety of different training, testing, and validation sets. *Talk to me at this point so that I can see how you are going with it.*

4 Using a Machine Learning Algorithm

We can go a bit further by actually stuffing the data into a machine learning algorithm, namely the Perceptron, which we will see in a couple of weeks. Download the code for the Perceptron from my website (`http://stephenmonika.net`) and `import` it into Python (`import pcn`), and then use it with code like (note that this version uses the even numbers for training and the odd numbers for testing; the use of 0.25 and 100 will become clearer later in the course):

```
trainin = pima[::2,:8]
testin = pima[1::2,:8]
traintgt = pima[::2,8:9]
testtgt = pima[1::2,8:9]

p = pcn.pcn(trainin,traintgt)
p.pcntrain(trainin,traintgt,0.25,100)
p.confmat(testin,testtgt)
```

You will need to put some thought into this in order to use your training, testing, and validation sets. Note that the results aren't very good (*talk to me about what the results are showing!*).

Last week you looked at data normalisation by dividing by the mean, etc. *Add this to your code. Where should it be done? At the start, or after partitioning the data into sets?*