Machine Learning 4/M - Vectors, Matrices and Cross-validation

Dr. Simon Rogers

Aims

The aim of this laboratory is to get you familiar with working with vectors and matrices (in Matlab or Python) and use them to fit (and evaluate) linear models describing various order polynomials. You'll find some useful python numpy commands at the bottom.

Tasks

Regression

- Load the Olympic 100m data (you should have this from the previous lab)
- For the linear model, create the vector t and the matrix X. t should be a column matrix containing all of the winning times. X should be a matrix with the same number of rows as t. The first column should all be ones and the second should be the Olympic year (you might want to scale the Olympic year as we did in lectures: $x = (x - 1896)/4$)
- Compute the optimal weight vector w using the equation derived in the lecture
- Create some test data (maybe 100 points evenly spaced between the first and last olympic years (or go a bit further backwards and forwards if you like)). Create the X object (call it Xtest) for your test data (first column 1, second column x)
- Plot the predicted values along with the data
- Modify your code to do the same for any order of polynomial. Plot what you get. If the inversion gives you errors, re-scale the data as mentioned above

Cross-validation

- For a range of polynomial orders, compute the squared loss at the optimal value of w. Plot this as a function of polynomial order - what do you notice?
- Implement a cross-validation routine. This should partition your data randomly into K sets (of roughly the same size). You then train and test your model K times, each time using K-1 sets to train, and then computing the loss on the remaining set. Each time, you should hold out a different set for testing. Average the loss over the K train-test cycles (folds).
- Plot this loss, along with the loss on the training data - what do you notice now?

Useful numpy

Load the data and store them as objects. Note the [:,None] - we need to make this into a 2-dimensional object to make it easy to concatenate things together later on.

```
import numpy as np
data = np.loadtxt('data100m.csv',delimiter=',')
x = data[:,0][:,None]
t = data[:,1][:,None]
```

Make an X (linear)

```
X = np.ones_like(x)
X = np.hstack((X,x))
```

Make a test x (you then need to turn this into a test X)

```
x_test = np.linspace(min,max,number_points)[:,None]
```

Matrix multiplication (A*B) is done like:

```
C = np.dot(A,B)
```

and inverse:

```
Ci = np.linalg.inv(C)
```