# Machine Learning Final Exam

**Department of Computer Science, University of Copenhagen**

Dhruv Chauhan

January 19, 2017

## 1 In a galaxy far, far away

### 1.1

The variance of the red-shifts in the spectroscopic training data was calculated to be:

$$0.0106$$

(where from now on, unless specified, values are shown to 3 significant places).

The MSE on the test SDSS predictions was calculated to be:

$$0.000812$$

This shows that the predictions were quite accurate.

### 1.2

The linear regression was done in Python, using the `sklearn` linear regression package. This performs an ordinary least squares linear regression. The error function is a Mean Squared Error.

The parameters of the model were calculated to be:

[ -2.82898070e+11, 6.79638352e+11, -7.30280682e+11, 3.84379194e+11,
-5.08387940e+10, -2.44829466e+11, 6.20014394e+10, 4.45567121e+11,
-3.89498568e+11, 1.26759474e+11, 2.82898070e+11, -3.96740282e+11,
3.33540400e+11, -5.08387940e+10, 2.44829466e+11, 1.82828027e+11,
-2.62739094e+11, 1.26759474e+11]

The error on the training data was calculated to be 0.00187, and on the test data was 0.00187 also. The errors noramlised by the variance, $\sigma^2_{red}$ were equal to 0.176 for both the test and the training data.

This normalised error falling below one signifies that...

## 1.3

For the non-linear regression, I chose to apply the K-nearest neighbours (KNN) algorithm. I chose this method for its simplicity (following Occam's razor), and therefore its ease of understanding. The simplicity of the algorithm is also reflected in the single hyperparameter, $k$, which means that there is less computation in tuning the hyperparameter.

I utilised the `neighbours` library from the `sklearn` package.

The KNN algorithm uses a distance metric to calculate the distance between a (set of) training point(s) and the other points. I used the Euclidian distance, given by $||\mathbf{x} - \mathbf{x}'||$, or $\sqrt{\mathbf{x}^T\mathbf{x}'}$. The algorithm works by calculating the distances from a test point to the other points, and then finding the nearest K points to that point. In a regression task, the test point is assigned the value of the mean of the nearest K neighbours.

My method involved using model selection methods such as cross-validation and grid search. Since this is model selection, cross validation was needed as we only use the training data during model selection. This gave us a better way to prevent overfitting of the training data. I used the `GridSearchCV` package from `sklearn`. The range of possible $k$ values was given as the odd numbers between 1 and 29. This performed a 5-fold cross validation on each of the possible values of $k$, averaging out the resulting error (i.e. splitting the data into 5 equal chunks, using 4 as the training set, and 1 as the validation set, and then cycling through all possible 5 validation sets). The error function used in the algorithm was the mean squared error.

This method resulted in the optimum hyperparameter as $k = 7$, with a MSE of 0.00118 on the test data, and a MSE of 0.000870. Clearly, the KNN Regressor worked better on the training data, which is to be expected as even with cross-validation, since the model is so simple, the training points are bound to have a low MSE. In comparison to the training data in the linear regressor, which performed the same as the test data. The difference in this would be due to the nature of a linear regressor, which would 'average out' the regression line over the points, thus leading to a small MSE on both the training and test data. On the test data, the KNN Regressor did perform a bit better than the linear regressor - perhaps due to the non-linear nature of KNN capturing the underlying nature of the data slightly more accurately. Overall, I believe the KNN method worked well as we had a relatively low variance on the data - KNN can be skewed by big outliers in the data. The cross-validation helped to prevent over-fitting on the data, which may have also caused the error on the test data to drop in comparison to the linear regressor.