An Unknown Signal Report

George Herbert
cj19328@bristol.ac.uk

April 1, 2021

Abstract

This report demonstrates my understanding of the methods I have used, the results results I have obtained and my understanding of issues such as overfitting for the 'An Unknown Signal' coursework.

1 Equations for linear regression

For a set of points that lie along a line with Gaussian noise $\mathbf{y} = \mathbf{X}\mathbf{w} + \epsilon$ where $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$, the maximum likelihood esimation is equivalent to the least square error estimation and is given by the equation:

$$\hat{\mathbf{w}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}.$$

I've implemented this equation in my code as the following function:

```
def regressionNormalEquation(self, X, y):
    return np.linalg.inv(X.T @ X) @ X.T @ y
```

2 Choice of polynomial order

3 Choice of unknown function

4 Overfitting

Overfitting occurs when a machine learning algorithm produces a model that has learnt the noise in the data as if it represents the structure of the underlying model. [1]

When producing a fit for a given line segment, overfitting is most likely to occur the algorithm produces a fit that has learnt the noise in the data as if it represents the structure of the underlying model.

In the case of linear regeression, overfitting typically occurs when the model produced contains too complex a function class, such that it would fail to predict future observations.

5 Model selection

To prevent overfitting each line segment, I have used a type of cross-validation known as k-fold cross-validation—in particular, I have used 5-fold cross-validation. I opted to use k-fold cross-validation due to the small number of datapoints in each segment.

To begin with, k-fold cross-validation involves shuffling and splitting each 20-point segment into k evenly-sized subsamples, as implemented in the `getKFold()` method. Then, each subsample is used exactly once as validation data, whilst the other $k-1$ paritions are used as training data. The cross-validation error for each model is calculated as follows [2]:

$$CV(\hat{f}) = \frac{1}{N}\sum_{i=1}^{N} SSE(y_i, \hat{f}_i(x_i))$$

where $\hat{f}$ is a given model; $\hat{f}_i$ is the fitted function, trained with the i-th subsample removed; N is the number of folds (in this case 5); SSE is the sum squared error function; y_i is the y datapoints of subsample i; and x_i is the x datapoints of subsample i.

The model with the lowest cross-validation error is then selected.

6 Testing

7 Optimisations and improvements

1. Using `np.linalg.solve`
2. Not shuffling the data before leave-one-out cross-validation
3. Not finding the average cross-validation error

References

[1] Burnham, K. P. and Anderson, D. R. (2002) *Model Selection and Multimodel Inference*. 2nd ed. Springer-Verlag.

[2] Hastie, T., Tibshirani, R. and Friedman, J. (2001) *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. 2nd ed. New York: Springer.