Lab 8: Model Assessment and Selection (I)

Ke Chen

NON-ASSESSED Lab Exercise

To carry out this work, you will need the Python notebook, `class8_code.ipynb`, which contains code to get you started with each of the problems, the data file, `class8_generated_data.npy`, and FTSE data, `class8_data_FTSE100.csv`. These are all available in a zipped file on BlackBoard alongside this sheet.

1 Polynomial models

Polynomial models are models of the form $f(x, \boldsymbol{\beta}) = \beta_0 + \beta_1 x + \beta_2 x^2 + \cdots + \beta_d x^d$ for some given d, known as the degree of the polynomial. The parameters of the model are given by the coefficients $\boldsymbol{\beta} = (\beta_0, \beta_1, \beta_2, \cdots, \beta_d)$ which can take on any real value. Although polynomial models can also be applied to multivariate settings in general, we shall focus only on the single-variable case in this class.

While fitting a polynomial model may seem like it would require a complicated non-linear closed-form solution, in fact we can turn the problem of fitting an d-degree polynomial model to a single variable into a problem of fitting a linear model to an d-dimensional vector. This can be done by treating each power of the input variable as a separate feature, i.e. an input variable x becomes the vector $\boldsymbol{x} = (1, x, x^2, \cdots, x^d)$, so then $f(x, \boldsymbol{\beta}) = \boldsymbol{\beta}^T \boldsymbol{x} = \beta_0 + \beta_1 x + \beta_2 x^2 + \cdots + \beta_d x^d$, which can be viewed as a special case of multivariate linear model. Thus, the *ordinary least square* (OLS) fitting for linear regression is applicable to the estimate of optimal coefficients, $\boldsymbol{\beta}$, based on a training data set of n examples $(n > d)$, $\big(X, \boldsymbol{y}\big) = \big\{(\boldsymbol{x}_i, y_i)\big\}_{i=1}^{n}$:

$$\hat{\boldsymbol{\beta}} = (X^T X)^{-1} X^T \boldsymbol{y}.$$

2 Bias, variance and their trade-off

The bias and variance trade-off lays a theoretic foundation for model assessment and selection. In this section, we are going to utilise a 1-D simulated data set and polynomial models of different degrees to understand the key concepts in statistical (machine) learning such as *underfitting* and *overfitting*.

2.1 Visualizing different polynomial fits

In this class, we are going to look into polynomial models via a visualisation of a polynomial fit to some simulated data given in the data file. Experiment with polynomial models of different degrees and different numbers of examples. Observe the behaviour of polynomial models of different degrees to find out on which degree the polynomial model suffer from the underfitting/overfitting.

2.2 Calculating bias and variance

This overfitting/underfitting is often described in terms of *Bias* and *Variance*, a decomposition of test errors. If the data is generated by a ground-truth function $f(x)$ plus some irreducible noise ϵ, and $\hat{f}(\cdot)$ is an estimator of our chosen model trained on a sample randomly drawn from a data set. Then, based on different estimators trained on different samples, Bias and Variance are defined as follows

$$Bias^2(x_i) = (\mathbb{E}[\hat{f}(x_i)] - f(x_i))^2,$$
$$Var(x_i) = \mathbb{E}[\hat{f}(x_i) - \mathbb{E}[\hat{f}(x_i)]]^2.$$

Recall that the average mean square error can be decomposed as follows:

$$\mathbb{E}[\mathrm{Err}(x_i)] = Bias^2(x_i) + Var(x_i) + \sigma^2,$$

where σ is the standard deviation of the irreducible (system) noise.

Calculating these values directly is intractable in most cases, however, if we know the ground-truth generator function, we can estimate these terms from a finite number of samples randomly drawn from our generated data. To do this we take k samples of data (a.k.a. k data subsets in the machine learning terminology) and fit our model to each of these samples, giving us k different functions $\hat{f}_i$. For any given datapoint x_i, we can then use the following estimates

$$Bias^2(x_i) \simeq (\frac{1}{k}\sum_{i=1}^{k}\hat{f}_i(x_i) - f(x_i))^2,$$

$$Var(x_i) \simeq (\frac{1}{k}\sum_{i=1}^{k}(\hat{f}_i(x_i) - \frac{1}{k}\sum_{i=1}^{k}\hat{f}_i(x_i)))^2.$$

We will consider some data generated by an unknown polynomial $f(x)$ plus added noise $\epsilon \sim \mathcal{N}(0,1)$. We wish to fit this data with a polynomial model. However, it is not immediately known what degree polynomial we should use.

Assignment 1: Run the code provided and look at the plots of polynomials of different degrees fit to multiple samples randomly drawn from the data set. With the experimental evidence and your justification, comment on (i) how the bias and variance relate to a degree of the polynomial and (ii) which polynomial degree has been used to generate the data set.

3 Cross Validation

In this section, we are going to compare two popular cross validation methods, *K*-fold CV, and Leave-One-Out CV (LOOCV), and investigate how we can use cross validation to guide model selection. In particular we will be looking at the FTSE 100, and how it has grown over the last 10 years. `ex8_data_FTSE.csv` contains historical data on the price of the FTSE 100 index on a monthly basis[1]. We would like to know if the growth of the FTSE has been linear (i.e. modelled by a polynomial of degree 1) or non-linear (which we will model by a polynomial of degree 3).

[1]Data was obtained from the following website https://uk.investing.com/indices/uk-100-historical-data

3.1 *K*-fold Cross Validation

In *K*-fold cross validation we divide the data into K 'folds'. For each of these folds we can get an estimate of the validation performance of our model by training on the other $K-1$ folds of the data set, and test on that one. Doing so leads to K different estimates of the validation performance which onr can use to estimate the mean validation performance of our model on the data set.

A sample implementation of *K*-fold cross validation with $K = 5$ folds is given in the example code. This will give 5 estimates of the validation performance of the given model.

3.2 Leave-One-Out Cross Validation (LOOCV)

LOOCV is a special case of *K*-fold cross validation where we set K to n, the number of examples in our dataset. This gives n different estimates of the validation performance of our model (although each of these estimates is on a test set of just a single example!).

Assignment 2: Based on the *K*-fold cross validation code provided in the notebook, complement the LOOCV code. Then, run an experiment with 5-fold and LOOCV based on two polynomial models ($d = 1$ or $d = 3$), respectively. For each of the two polynomial models, record the approximate time spent by using 5-fold cross validation and LOOCV. Based on the experimental results, comment on the difference between the two cross-validation methods in terms of computational efficiency.

Assignment 3: For each of the two polynomial models ($d = 1$ or $d = 3$), calculate the average, and standard-deviation, of the mean square error loss on the validation set using LOOCV and record your results.