MDL Assignment-1 Report

TEAM :
Nishant Sachdeva (2018111040)
Sai Tanmay Reddy Chakkera (2018101054)

Q1 and Q2: (solutions)

Algorithm Implementation:

The question asks us to perform polynomial regression (separately on degrees going from 1 to 9) on a given set of data and to thence plot and analyse the bias-variance values.

The algorithm starts so:

1. We extract the data from the pkl files and dump them into lists from python.
2. As per instructions, we divide the data into test set and data set in the ratio 9:1.
3. The training data is then further divided into 10 sub set of equal size.

```
xTrain, xTest, yTrain, yTest = train_test_split(xDict, yDict, test_size=0.1, random_state=57)
temp = np.concatenate((xTrain, yTrain), axis =1)
np.random.shuffle(temp)
xTrain, yTrain = np.hsplit(temp, 2)

dx , dy = distribute(xTrain, yTrain)
```

4. Thereafter we send the sets dx, dy, xTest, yTest into the train function from the train module. We expect to get as return values , the bias and variance values in tabulated form.
5. In the train function, for every degree in range 1 to 9, both inclusive, we iterate over each training data sets, making 10 different models(90 in total). For each such model created, we run the test set, and store the predicted values. Mean of the predicted values of models of the same degree for a given test x is subtracted from the actual test y, and such values are calculated for all x in the test set. The

mean of them gives the mean bias. Similarly, mean of the variance of the predicted values of models of the same degree for a given test x gives the mean variance.

```
  bi = []
  vare = []
  for degree in range(1,10):
      variance_wala_array = []
      bias_wala_array = np.zeros((np.shape(xTest)[0], 10))
      for i in range(0, 10):
          train_x = dx[i][:, np.newaxis]
          train_y = dy[i][:, np.newaxis]
          poly = PolynomialFeatures(degree = degree)
          X_poly = poly.fit_transform(train_x)
          lin2 = LinearRegression()
          lin2.fit(X_poly, train_y)
          pred_y = lin2.predict(poly.fit_transform(xTest))
          bias_wala_array[:, i] = pred_y.flatten()
          variance_wala_array.append(pred_y)
      bias, variance = calculate_bias_and_variance(bias_wala_a
rray, yTest)
      bi.append(bias)
      vare.append(variance)
```

6. The train function returns two lists, bi and vare. bi is the list of mean bias for a given degree. and vare is the list of the mean variance for a given degree.
7. Next, we put the bias, variance values together and print the data in tabular form.

Analysis :

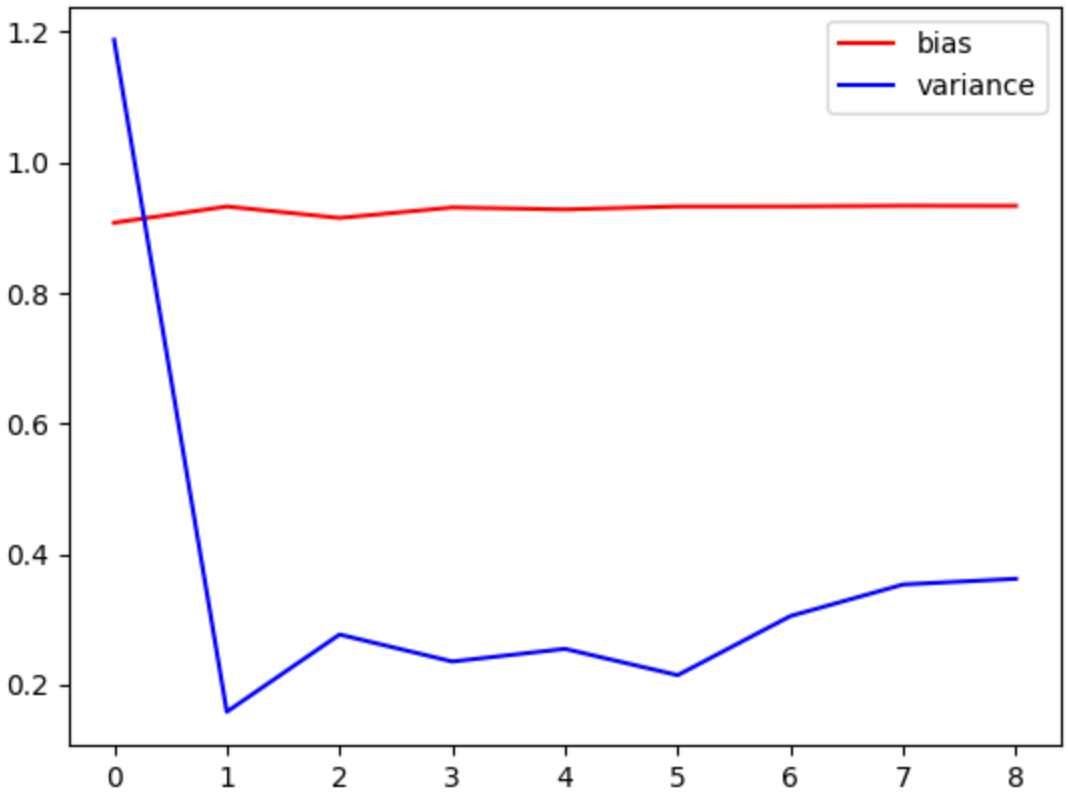
1. The results that we get vary with different running iterations of the same running code.

2. We also varied the the extent of degrees to which we were willing to perform regression on the data.

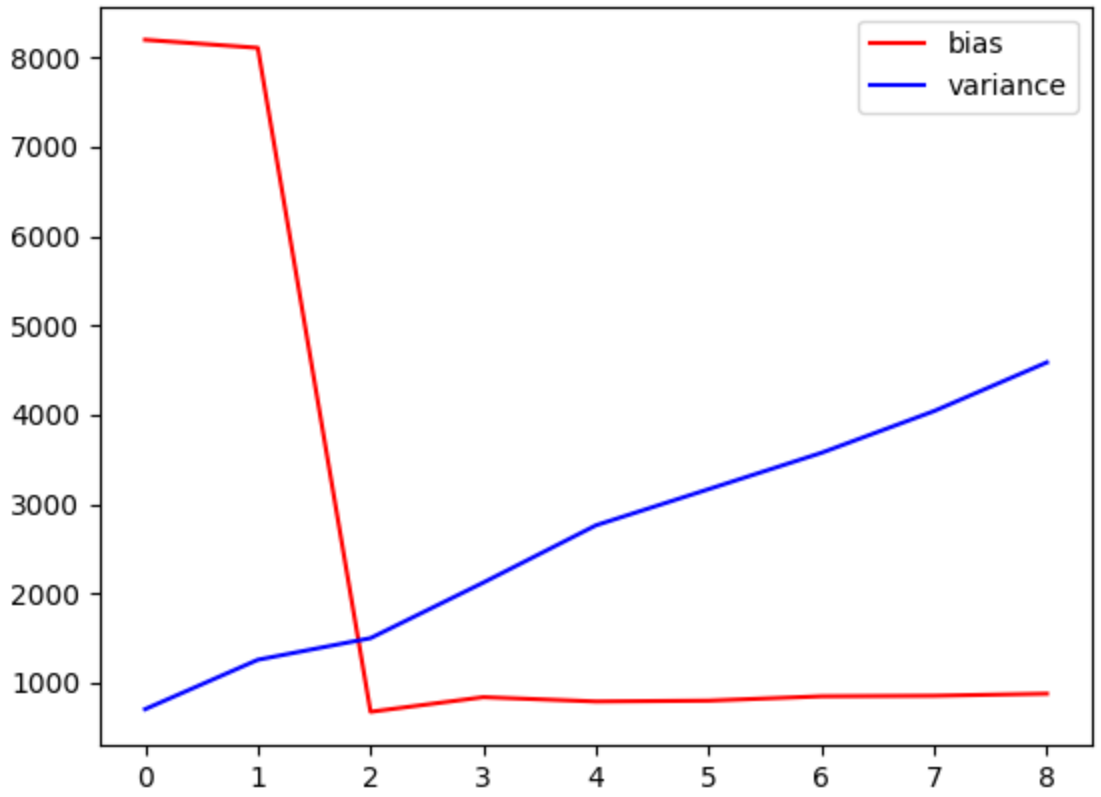
Below are a few examples of the trends that were observed with the various iterations:

Degree	Bias	Variance
1	9.0778	0.140452
2	9.32836	0.0194094
3	9.15524	0.0388045
4	9.30994	0.0283588
5	9.27929	0.0327247
6	9.32503	0.0347779
7	9.3252	0.0419026
8	9.33436	0.0445009
9	9.33196	0.0506098

Regular run of the test bench: Q1



Re-run of the test bench with modified and scaled parameters: Q1



Run of the test bench for Q2 (Note : The only real difference between both the questions is that the data sets in the first question have to be manually partitioned into sub sets that have random distribution whereas in the second question, we are already provided with well partitioned data sets which we can directly work on.

Note : Although the data sets are supposed to be random, by looking at the data plots, we can tell that the data sets are very uniformly partitioned. This, we say because the bias on the “test” data goes down even with increasing model complexity (which is a clear indicator as to the extent to which the test data mirrors the train data) .

Hence, at least on this data set, we cannot tell if the model is over-fitting or not because the “test” data is not exactly foreign.

Thanks

and Regards

Sai Tanmay

Chakkera

Nishant

Sachdeva