University of Skövde
School of Informatics
Master of Data science
Haftamu Hailu Tefera
al17hafte@student.his.e

Advanced Artificial Intelligence - IT714A

Evaluation of Machine Learning algorithms and Bias Variance Trade off analysis

I. Description of the experment

The objective of this assignment to evaluate machine learning algorithms on a given data set and analyzing the bias variance tradeoff as complexity of the model increases to fit the data using different polynomial regressions.

The selected dataset consists of different attributes of the plant called abalone. The main task of the experiment is predicting the age of the plant from different physical measurements like height, width, height and others. The data sets consist of 4177 instances and 8 features and one target variable called rings which is the age of the plant. The data is mainly collected for regression purpose.

The link for the data sets; - https://archive.ics.uci.edu/ml/datasets/abalone

I used Python as a programming language and polynomial regression and collection of functions and classes which are defined inside the sklearn machine learning library. Firstly, I modeled the relationship between the independent and response variable using simple linear mode and polynomial regression to fit the relationship between independent variables and the response variable

II. Design of the experiment

Firstly, I performed prepressing tasks to drop categorical variables (Sex) and some of features which are not used in the model such as Shell weight and Shucked weight. After preprocessing to handle the prediction task, I used cross validation with 5 folds. I used polynomial features to generate feature matrix consisting of all polynomial combinations from the linear repressor. To make it explicit, I selected three random rows from the data sets for computing bias and variance values and I tested the bias and variance using the records. Furthermore, to store the variance and bias values while predicting, we created a matrix of with number of test records and number of polynomials as main dimensions of the matrix.

To store the predicted values we created a matrix of results with test size and number of folds as dimensions of the matrix. These values are used for calculating the bias and variance. Using the selected number of cross validations, we predicted the value of each test data 5 times.

Example for the linear regression (polynomial degree 1) I created a variable called p1_results and the result looks like

	0	1	2	3	4
0	7.33368	7.53582	7.60998	7.21719	7.25984
1	9.62044	8.99238	9.75818	9.91887	9.61218
2	8.25333	8.33759	8.40019	8.50115	8.42326

Where the rows numbers are the test records and column number's the number of cross validations. Even if we used different polynomial regression with various degrees, but there is still error between the actual data points and predicted values. When I use simple modes, they do not represent the actual relationship between the response variable (ring) and the feature variable variables. To decide which model is perfect for my regression I performed bias variance analysis by varying complexities and on the next section we discuss the results and finding of the experiments.

III. Discussion of the results and findings from the experiment

After I trained the model on the train data, finally I evaluated the performance of the model on the unseen data and registered the following bias variance values for the selected test data. Where X0, X1 and X2 are the records at position 5,60 and 100 of the abalone data set.

Test Data	Length	Diameter	Height	Whole Weight	Viscera weight	Degree	Bias	Variance
X0	0.425	0.3	0.095	0.3515	0.0775	1	**0.371**	**0.0239**
X1	0.45	0.345	0.105	0.4115	0.1125	1	**6.659**	**0.099**
X2	0.36	0.265	0.095	0.2315	0.046	1	**1.913**	**0.007**
X0	0.425	0.3	0.095	0.3515	0.0775	2	**1.177**	**0.023**
X1	0.45	0.345	0.105	0.4115	0.1125	2	**8.241**	**0.155**
X2	0.36	0.265	0.095	0.2315	0.046	2	**1.297**	**0.067**
X0	0.425	0.3	0.095	0.3515	0.0775	3	**0.115**	**0.621**
X1	0.45	0.345	0.105	0.4115	0.1125	3	**17.556**	**0.593**
X2	0.36	0.265	0.095	0.2315	0.046	3	**0.489**	**0.063**

Table of Bias and Variance for selected test data's (row 5, 60,100)

Note that both bias and variance are calculated column wise (record wise calculation) taking into consideration all the values of the independent variables.

Bias variance graph for the selected test data.

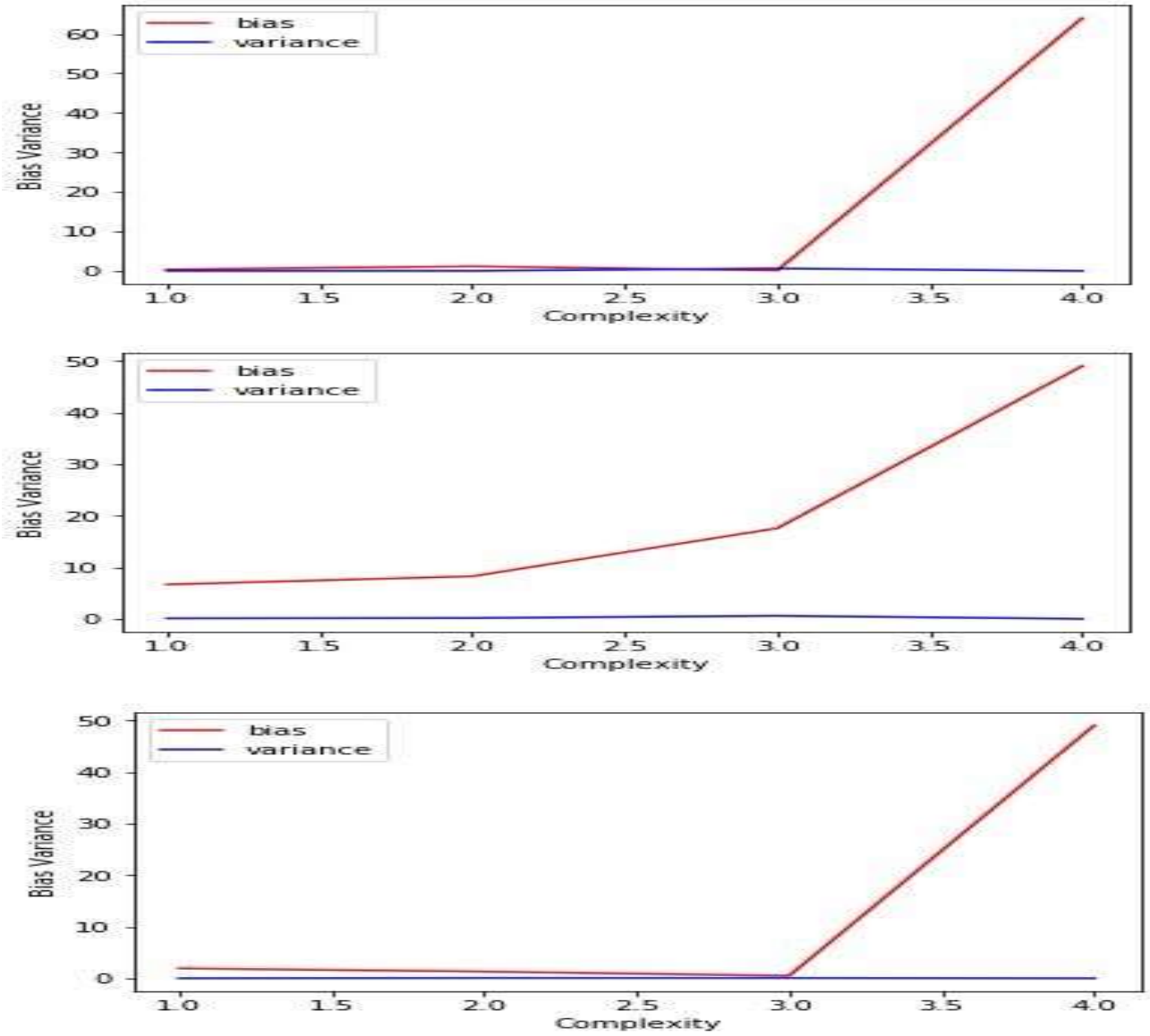

Fig 1. Bias and Variance graphs

From the above three graphs the variance is almost constant but the bias changes rapidly after the third complexity (Here the actual polynomial degree is 6 because I used polynomials with degree 1, 5, 6, and 7). For the first and third graph according the Occam's razor principle the optimum value for bias and variance is attained at the point 3 on the graph. As we stated earlier the actual degree represented by 3 above is polynomial degree 6. For the second graph the optimum complexity is polynomial degree 1.

IV. Code for the experiment

1. **Bias Function Code**

```
def compute_bias(predicted_values, actual):
  #compute  expectation
  mean_val=np.mean(predicted_values,axis=1)
  #axis =1 is column wise addition
  mean_val = np.reshape(mean_val, (len(mean_val), 1))
  actual = np.reshape(actual, (len(mean_val), 1))

  bias = mean_val - actual;
  bias  = np.square(bias)
  bias  = np.reshape(bias, (len(actual), 1))
    return bias
```

2. **Variance Function Code**

```
def compute_variance(predicted_values):
  # compute expectation
  mean_val = np.mean(predicted_values, axis=1)
  mean_val = np.reshape(mean_val, (len(mean_val), 1))
  diff_sqr = np.square(predicted_values - mean_val )
  #compute variance
  var=np.mean(diff_sqr,axis=1)
  var = np.reshape(var, (len(mean_val), 1))
    return var
```

3. Code computing prediction and storing the value of bias and variance

```
def main():
  """Following code is for calculating the age of the abalone plant"""

  Model = LinearRegression
  i = 0
  for train_indices, test_indices in kf:
    #split data
    X_train_ = X_train[train_indices, :]; Y_train_ = Y[train_indices]

    # Testing out on the linear reg_modelression
    for d,degree in enumerate(poly_degree):
      reg_model = Model()
      if degree == 1:
        X_tr = X_train_
        X_tst = X_test
```

```
            else:
                # PolynomialFeatures (prepreprocessing)
                poly = PolynomialFeatures(degree=degree)
                X_tr = poly.fit_transform(X_train_)
                X_tst = poly.fit_transform(X_test)

            # fit regressor
            reg_model.fit(X_tr, Y_train_)

            # make prediction
            predictions = reg_model.predict(X_tst)
            predictions = np.reshape(predictions, (len(predictions), 1))

            # store result
            if d==1:
                P1_result[:,[i]] = predictions
            elif d==2:
                P2_result[:,[i]] = predictions
            elif d == 3:
                P3_result[:,[i]] = predictions
            elif d == 4:
                P4_result[:,[i]] = predictions
        i += 1
    #%% compute bias and variance for each complexity
    bias[:,[0]]  = compute_bias (P1_result, Y_test)
    variance[:,[0]]  = compute_variance(P1_result)

    bias[:,[1]]  = compute_bias (P2_result, Y_test)
    variance[:,[1]]  = compute_variance(P2_result)

    bias[:,[2]]  = compute_bias (P3_result, Y_test)
    variance[:,[2]]  = compute_variance(P3_result)

    bias[:,[3]]  = compute_bias (P4_result, Y_test)
    variance[:,[3]]  = compute_variance(P4_result)
```

4. Code for plotting bias and variance

```
def plot_bias_var(bias=None, var=None):
#    x  = range(1,len(poly_degree)+1)
    x = np.linspace(1, len(poly_degree), len(poly_degree), endpoint=True)

    # plot bias and varinave for each data point
    for data_num in range(bias.shape[0]):
        y_bias  = bias[data_num,:]
        y_var  = var[data_num,:]
        plt.plot(x,y_bias, 'r', label='bias')
        plt.plot(x, y_var, 'b', label='variance')
        plt.legend()
        plt.show()
```

Reference

1. Class lecture notes on machine learning
2. http://scikit-learn.org/stable/
3. http://www.ultravioletanalytics.com/2014/12/12/kaggle-titanic-competition-part-ix-bias-variance-and-learning-curves/
4. http://scott.fortmann-roe.com/docs/BiasVariance.html
5. https://www.learnopencv.com/bias-variance-tradeoff-in-machine-learning/