bais-variance-trade-off_solution Übungen

August 15, 2022

1 Bias-variance decomposition for classification and regression losses

The goal for any machine learning model is to estimate a mapping function $f(x)$ based on the training data. Prediction errors can be divided into three types:

- Bias error
- Deviation error
- Irreducible error

We will address the bias and variance error. Because we can influence them.

1) Bias error
   Bias are the simplifying assumptions made by a model to make the objective function easier to learn.

2) Deviation error
   The variance is the amount by which the estimate of the objective function changes when different training data are used.

what does that mean? It means ML algorithms struggle to achieve low variance and low base with good estimate. But it turns out that's not possible. In machine learning, there is no getting around the relationship between bias and variance.

- Increasing the bias reduces the variance.
- Increasing the variance decreases the distortion.

we might say that "high variance" is proportional to overfitting, and "high bias" is proportional to underfitting.

Bias and variance provide a framework to understand the behavior of machine learning algorithms when striving for best performance.

Bias Variance Decomposition of a Decision Tree Classifier digits dataset

[ ]:
```
from mlxtend.evaluate import bias_variance_decomp
from sklearn.tree import DecisionTreeClassifier
from sklearn import datasets, metrics
from sklearn.model_selection import train_test_split
```

2 Exercise 1:

- 1.1 Load Sklearn's load_digits and perform the necessary preprocessing steps to perform a multiclass classification.

```
[ ]: # your code


     print(digits.data.shape)
```

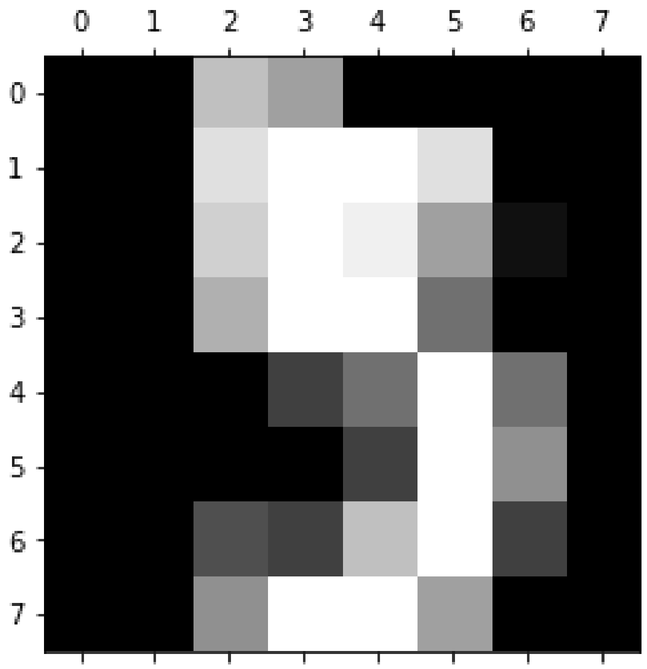
```
(1797, 64)
```

- 1.2 visualize the 5th image:

```
[ ]: #your code
```

```
[ ]: <matplotlib.image.AxesImage at 0x7f88331d7280>

<Figure size 432x288 with 0 Axes>
```



3 Exersice 2

- 2.1 Split your image dataset into random train and test subsets ()
- 2.2 Call DecisionTreeClassifier object to start classification
- 2.3 Use bias_variance_decomp to calculate average expected loss, average deviation, and average variance

```
[ ]: # Enter your code here


     # 2.1 test_size=0.5


     # 2.2 random_state=123



     print('Average expected loss: %.3f' % avg_expected_loss)
     print('Average bias: %.3f' % avg_bias)
     print('Average variance: %.3f' % avg_var)
```

```
Average expected loss: 0.272
Average bias: 0.119
Average variance: 0.240
```

4 Exersice 3

- 3.1) For comparison, the bias-variance decomposition of a AdaBoostClassifier classifier:

```
[ ]: # insert your code




     print('Average expected loss: %.3f' % avg_expected_loss)
     print('Average bias: %.3f' % avg_bias)
     print('Average variance: %.3f' % avg_var)
```

```
Average expected loss: 0.271
Average bias: 0.117
Average variance: 0.239
```

- 3.2) discuss your results:

4.1 Exersice 4

Bias Variance Decomposition of a Decision Tree Regressor Iris Dataset

- 4.1 Load the sklearn iris dataset and perform the necessary preprocessing steps to perform a multiclass classification.
- 4.2 Do the same straps as in the previous exercises (spliting and compute average expected loss, average deviation, and average variance)
- 4.3 For comparison, the bias-variance decomposition of a AdaBoostClassifier classifier and ddiscuss your results:

```
[ ]: from mlxtend.data import iris_data



     ##### insert the code ######




     print('Average expected loss: %.3f' % avg_expected_loss)
     print('Average bias: %.3f' % avg_bias)
     print('Average variance: %.3f' % avg_var)
```

```
Average expected loss: 0.062
Average bias: 0.022
Average variance: 0.040
```

For comparison, the bias-variance decomposition of a AdaBoostClassifier classifier, which should intuitively have a lower variance compared than a single decision tree:

```
[ ]: from sklearn.ensemble import AdaBoostClassifier

     #####   insert your code #####

     print('Average expected loss: %.3f' % avg_expected_loss)
     print('Average bias: %.3f' % avg_bias)
     print('Average variance: %.3f' % avg_var)
```

```
Average expected loss: 0.061
Average bias: 0.022
Average variance: 0.039
```