Data Analytics Course - Lesson 07

Ths. Vu Duy Khuong

Agenda

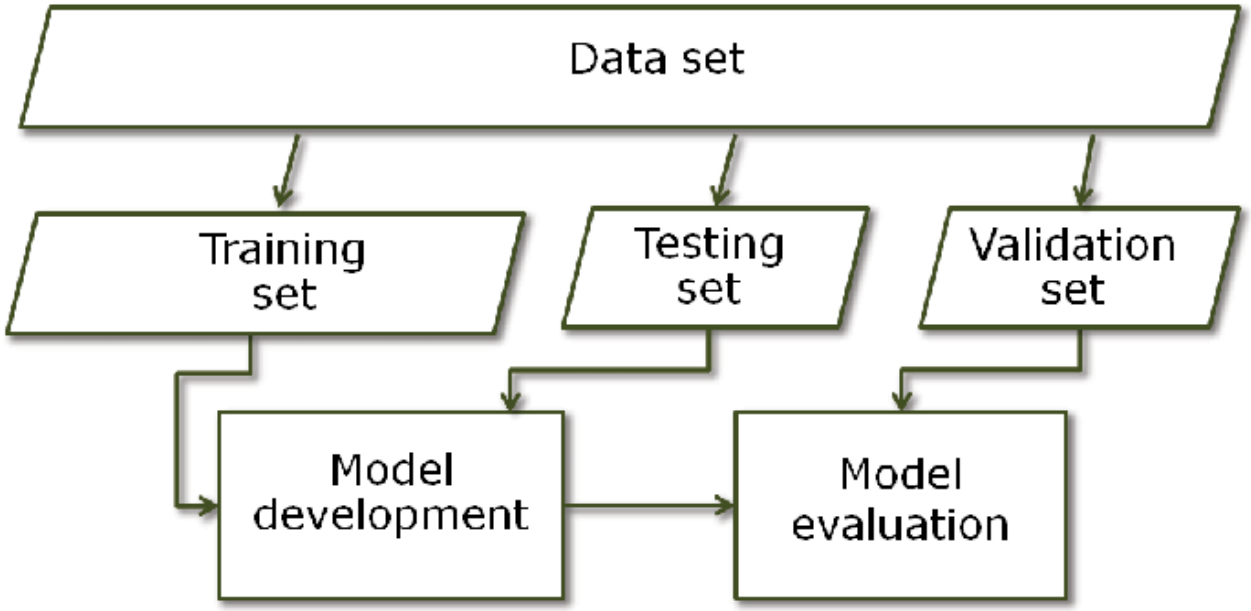
- I. Model Evaluation Metrics
- II. Model Evaluation Problems
- III. Model Evaluation Methods

I. Model Evaluation Metrics

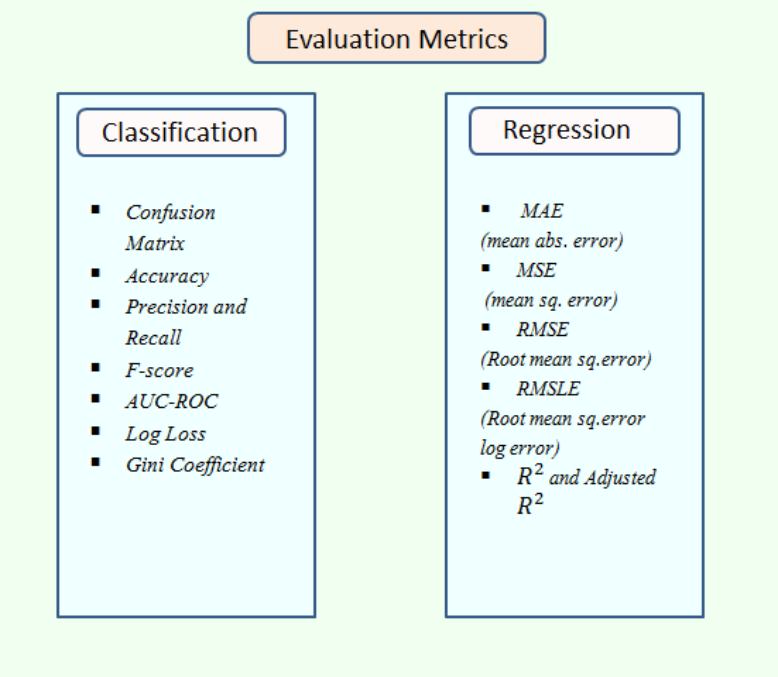
1. What is Model Evaluation

Model validation is the process of evaluating a trained model on test data set.

This provides the



I. Model Evaluation Metrics



I. Model Evaluation Metrics

2. Accuracy

- Accuracy is defined as the percentage of correct predictions for the test data.

- It can be calculated easily by dividing the number of correct predictions by the number of total predictions. Given a dataset with m samples:

$$Accuracy = \frac{Correct\ Prediction}{All\ Prediction} = \frac{1}{m}\sum_{i=1}^{m}(f(x_i) = y_i)$$

I. Model Evaluation Metrics

2. Accuracy

Error rate:

$$Error\ Rate = \frac{Incorrect\ Prediction}{All\ Prediction} = \frac{1}{m}\sum_{i=1}^{m}(f(x_i) \neq y_i)$$

$$Accuracy = 1 - Error\ Rate$$

I. Model Evaluation Metrics

2. Accuracy

```
# Cross Validation Classification Accuracy
from pandas import read_csv
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
from sklearn.linear_model import LogisticRegression
filename = 'pima-indians-diabetes.data.csv'
names = ['preg', 'plas', 'pres', 'skin', 'test', 'mass', 'pedi', 'age', 'class']
dataframe = read_csv(filename, names=names)
array = dataframe.values
X = array[:,0:8]
Y = array[:,8]
kfold = KFold(n_splits=10, random_state=7)
model = LogisticRegression()
scoring = 'accuracy'
results = cross_val_score(model, X, Y, cv=kfold, scoring=scoring)
print("Accuracy: %.3f (%.3f)") % (results.mean(), results.std())
```

I. Model Evaluation Metrics

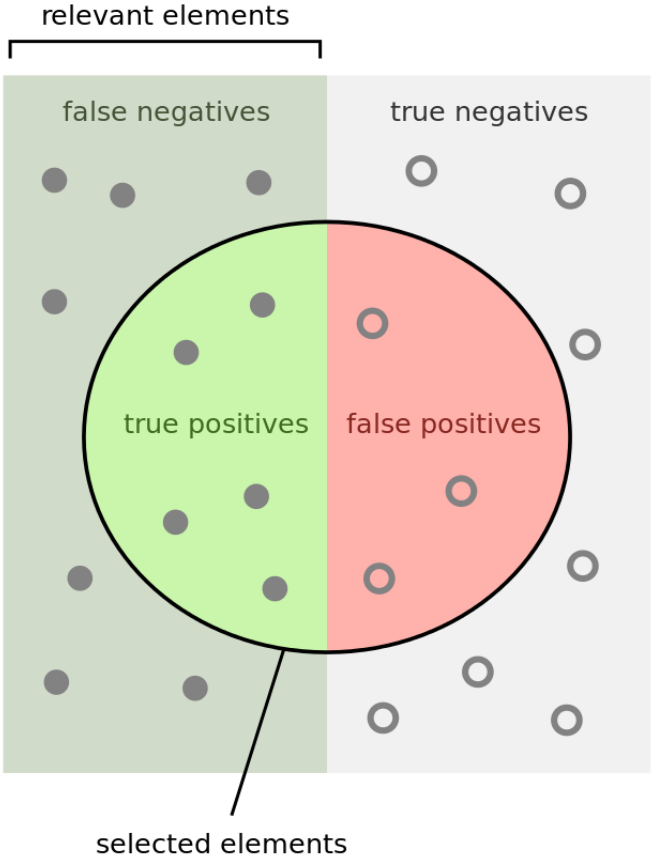
3. Precision, Recall, and F-Score

When performing classification predictions, there's four types of outcomes that could occur.

- **True positives** are when you predict an observation belongs to a class and it actually does belong to that class.
- **True negatives** are when you predict an observation does not belong to a class and it actually does not belong to that class.
- **False positives** occur when you predict an observation belongs to a class when in reality it does not.
- **False negatives** occur when you predict an observation does not belong to a class when in fact it does.

I. Model Evaluation Metrics

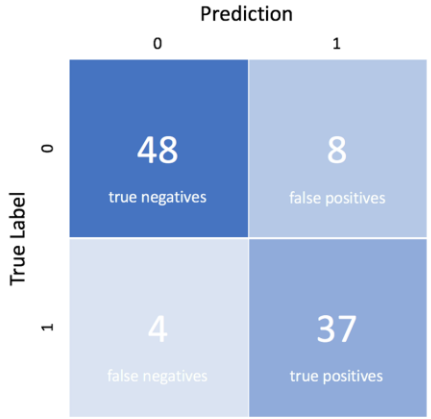
3. Precision, Recall, and F-Score



I. Model Evaluation Metrics

3. Precision, Recall, and F-Score

- These four outcomes are often plotted on a confusion matrix.
- The following confusion matrix is an example for the case of binary classification.
- You would generate this matrix after making predictions on your test data and then identifying each prediction as one of the four possible outcomes described above.

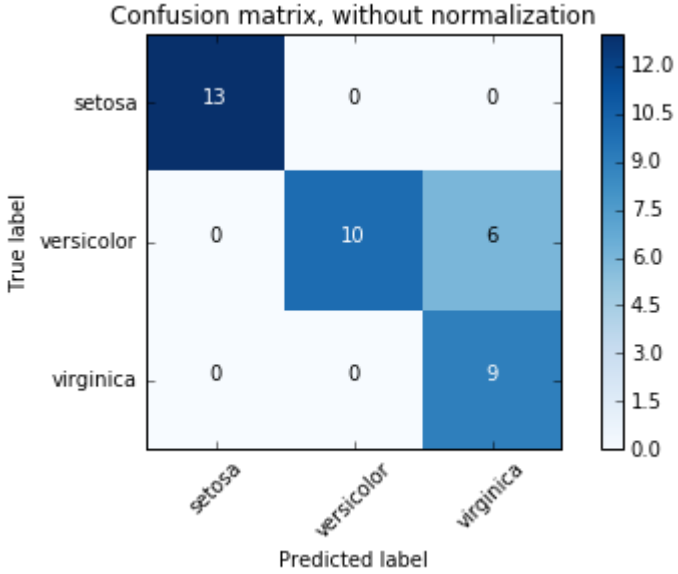


I. Model Evaluation Metrics

3. Precision, Recall, and F-Score

- You can also extend this confusion matrix to plot multi-class classification predictions.

- The following is an example confusion matrix for classifying observations from the Iris flower dataset.



I. Model Evaluation Metrics

3. Precision, Recall, and F-Score

```
# Cross Validation Classification Confusion Matrix
from pandas import read_csv
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix
filename = 'pima-indians-diabetes.data.csv'
names = ['preg', 'plas', 'pres', 'skin', 'test', 'mass', 'pedi', 'age', 'class']
dataframe = read_csv(filename, names=names)
array = dataframe.values
X = array[:,0:8]
Y = array[:,8]
test_size = 0.33
```

```
seed = 7
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=test_size,
    random_state=seed)
model = LogisticRegression()
model.fit(X_train, Y_train)
predicted = model.predict(X_test)
matrix = confusion_matrix(Y_test, predicted)
print(matrix)
```

I. Model Evaluation Metrics

3. Precision, Recall, and F-Score

Precision is defined as the fraction of relevant examples (true positives) among all of the examples which were predicted to belong in a certain class.

$$Precision = \frac{True\ Positive}{True\ Positive + False\ Positive}$$

I. Model Evaluation Metrics

3. Precision, Recall, and F-Score

Recall is defined as the fraction of examples which were predicted to belong to a class with respect to all of the examples that truly belong in the class.

$$Recall = \frac{True\ Positive}{True\ Positive + False\ Negative}$$

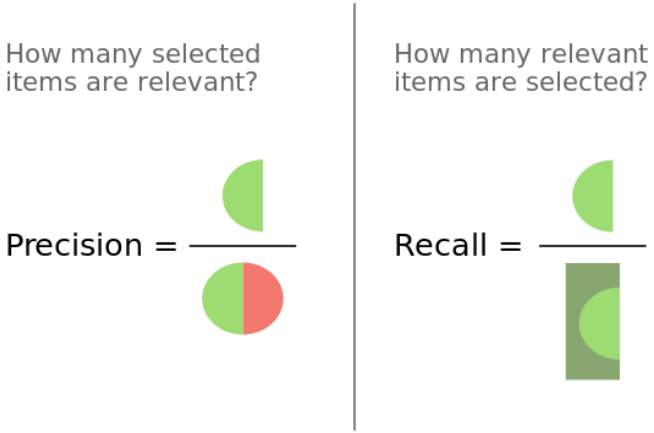
I. Model Evaluation Metrics

3. Precision, Recall, and F-Score

- Precision and recall are useful in cases where classes aren't evenly distributed.

- The common example is for developing a classification algorithm that predicts whether or not someone has a disease.

- If only a small percentage of the population (let's say 1%) has this disease, we could build a classifier that always predicts that the person does not have the disease, we would have built a model which is 99% accurate and 0% useful.



I. Model Evaluation Metrics

3. Precision, Recall, and F-Score

In case of need to evaluate both precision and recall, we can combine them into one metrics; the common approach for combining these metrics is known as the f-score.

$$F_\beta = (1 + \beta^2) \frac{Precision * Recall}{(\beta^2 * Presision) + Recall}$$

The β parameter allows us to control the tradeoff of importance between precision and recall. β<1 focuses more on precision while β>1 focuses more on recall.

I. Model Evaluation Metrics

4. Mean Squared Error (MSE)

- MSE is simply defined as the average of squared differences between the predicted output and the true output.

- Squared error is commonly used because it is agnostic to whether the prediction was too high or too low, it just reports that the prediction was incorrect.

$$MSE(y_{true}, y_{pred}) = \frac{1}{n_{samples}} \sum (y_{true} - y_{pred})^2$$

I. Model Evaluation Metrics

5. R^2

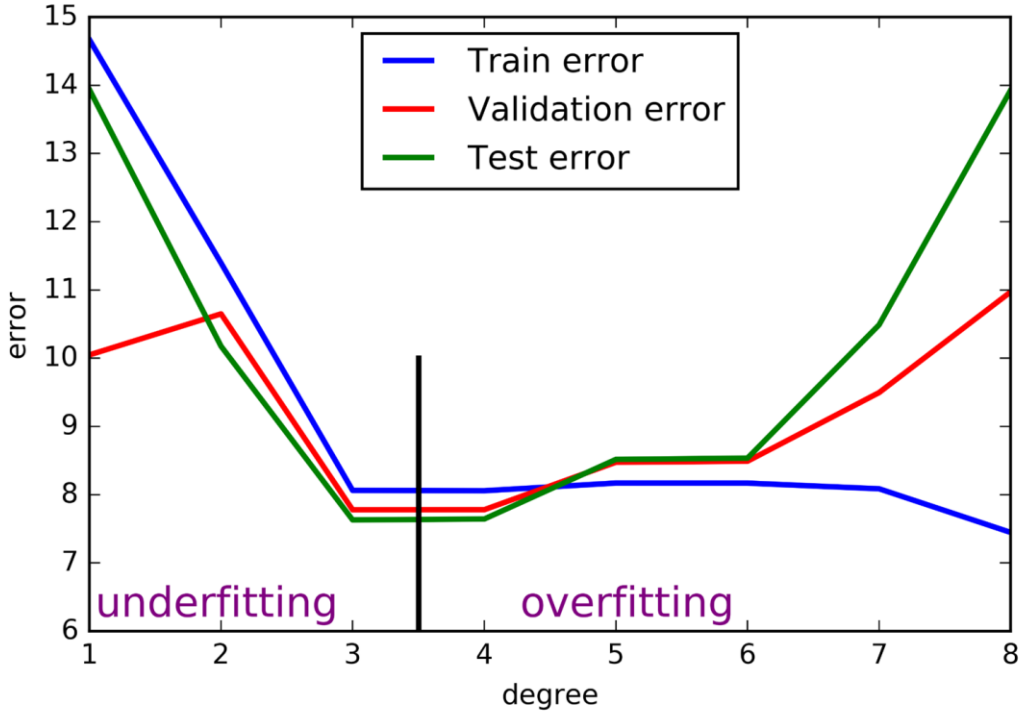
The **R^2 coefficient** represents the proportion of variance in the outcome that our model is capable of predicting based on its features.

$$R^2\left(y_{true}, y_{pred}\right) = 1 - \frac{\sum\left(y_{true} - y_{pred}\right)^2}{\sum\left(y_{true} - \overline{y}\right)^2}$$

$$\overline{y} = \frac{1}{n_{samples}} \sum y_{true}$$

II. Model Evaluation Problems

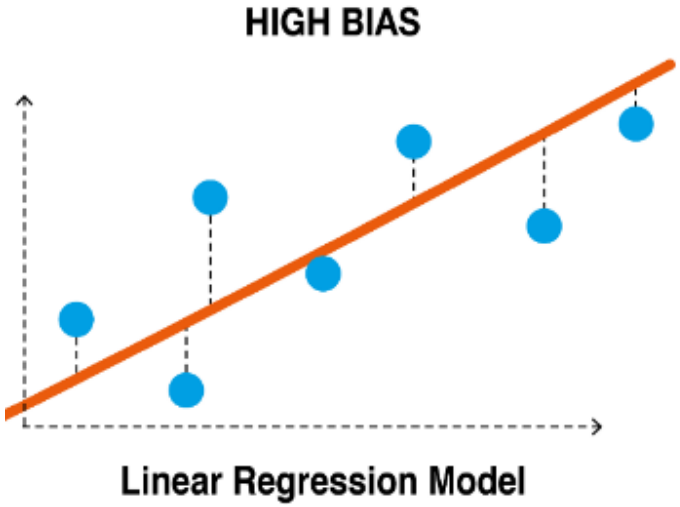
- Underfitting
- Overfitting

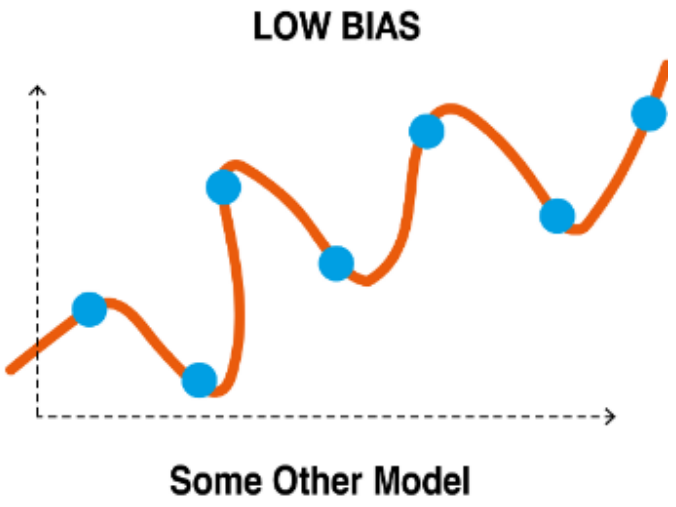


II. Model Evaluation Problems

1. Bias

- Bias is the difference between the value our model predicts and the real value (predict - ground truth).
- Model with low bias, the difference between predicted value and true value is small => Good model.
- Model with high bias, the difference between predict and ground truth is large => Model has high error on both training and testing set => Underfitting



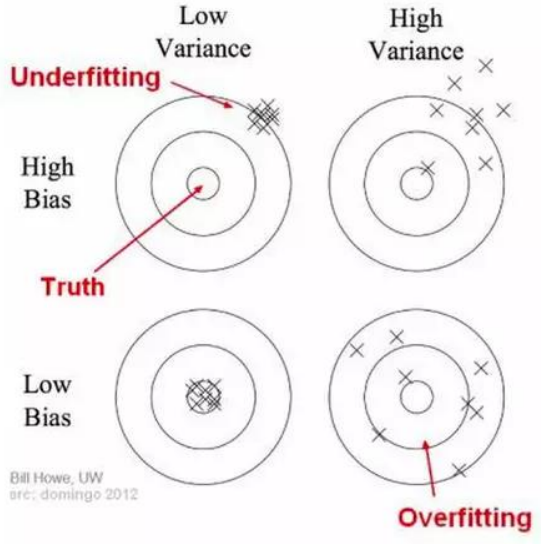


II. Model Evaluation Problems

2. Variance

- Variance represents the dispersion of the data.

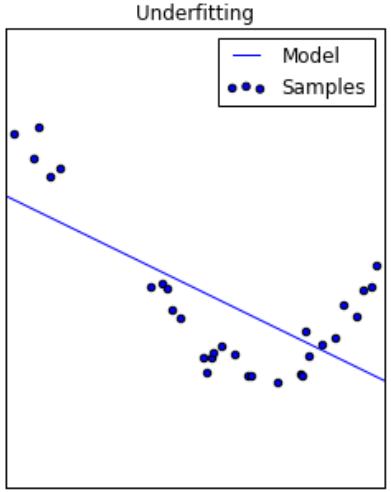
- High variance shows high dispersion, focusing mainly on training data but not giving generality on never-before-seen data => The model is very good on the training data set but the results are very bad on the set testing => Overfitting



II. Model Evaluation Problems

3. Underfitting

- Underfitting is a phenomenon when the built model does not have high accuracy in the training data set as well as generalizing to the overall data.
- When underfitting occurs, the model will not do well with any of the datasets in question.
- When Underfitting occurs, we can fix it by changing the algorithm or adding more input data.

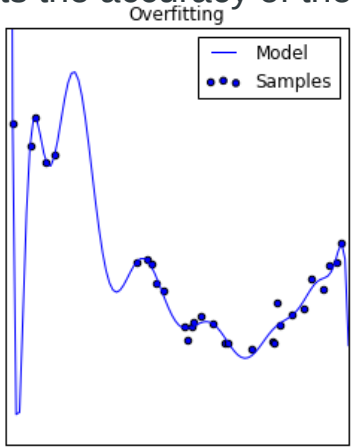


II. Model Evaluation Problems

4. Overfitting

- Overfitting is a phenomenon when the built model shows the details of the training dataset. This means that both noisy data, or outliers in the training set are selected and learned to generate the model rule.

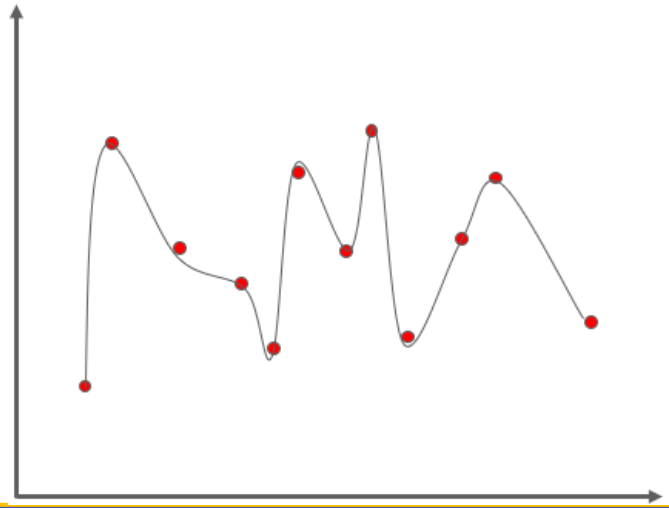
- These rules will not make much sense when applied to new datasets that may have a different type of noise data. Then, it negatively affects the accuracy of the model in general.



II. Model Evaluation Problems

5. Cause of Overfitting

- The cause of overfitting can be from a model that is too complex or the data that is not general enough.
- Overly complex models often have a large representation space and easily fit complex dividing boundaries. This seems to be good for the forecasting process but turns out to be bad because complex models tend to learn details instead of learning general rules.
- Data-driven overfitting is also quite common. When the data is not broad and generalizable enough, the model cannot make good predictions on the training set that has not been learned.

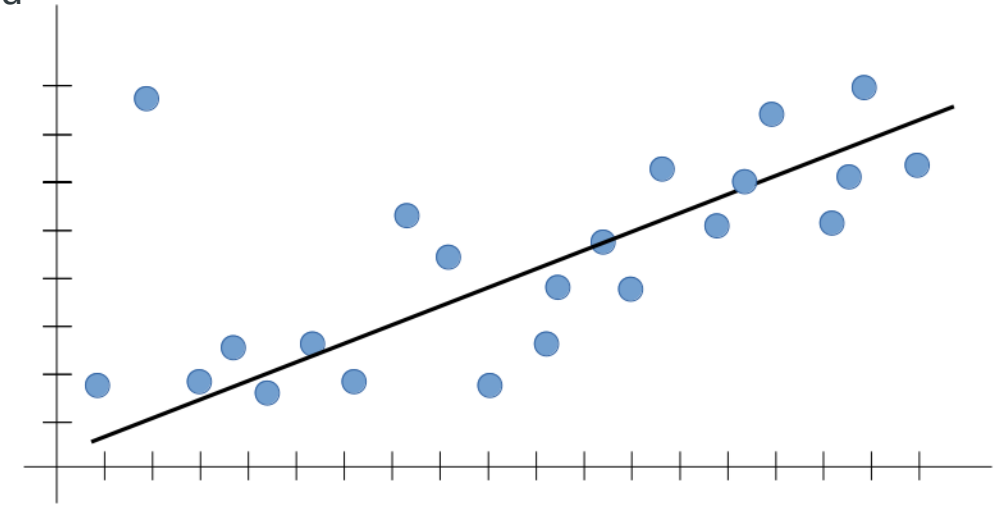


II. Model Evaluation Problems

5. Cause of Underfitting

- The underfitting phenomenon can also come from the model side or from the data side. For large datasets, using a model that is too small will not be able to represent the data well. Then we need to move on to more complex model classes.

- Insufficient data diversity is also the cause of arthroplasty

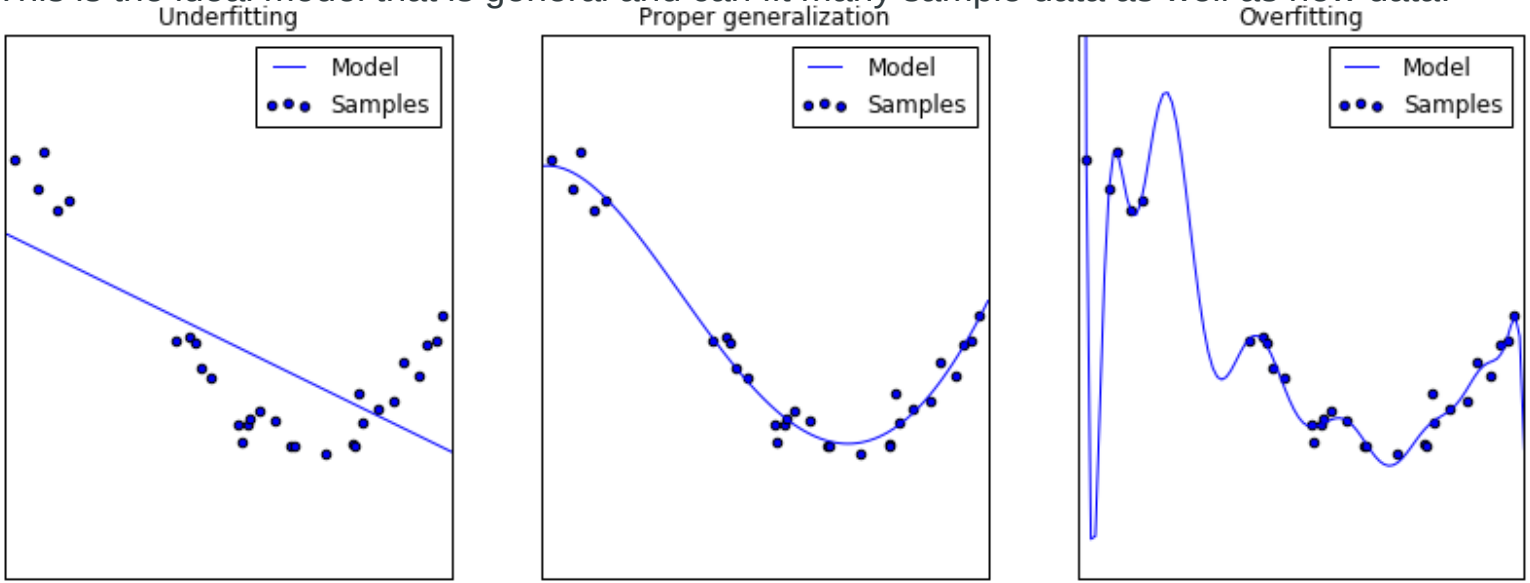


II. Model Evaluation Problems

6. Good Fitting

- Good Fitting is located between Underfitting and Overfitting. The model gives reasonable results with both the training dataset and the new data sets.

- This is the ideal model that is general and can fit many sample data as well as new data.



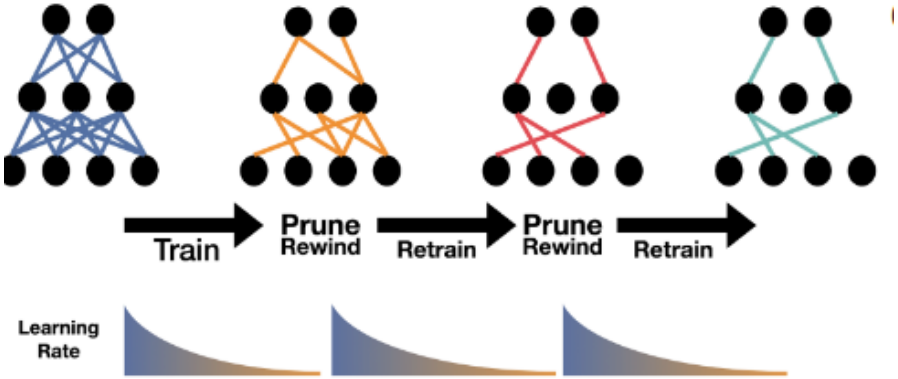
II. Model Evaluation Problems

7. Way to avoid overfitting

a, Simple model

- One of the reasons why your model becomes overfitting is: your model is too deep, complex (such as many layers, nodes) while having only a little bit of data.

- The solution here: one is to increase the data as above, 2 to reduce the complexity of the model by removing some layers. , node.



II. Model Evaluation Problems

7. Way to avoid overfitting

b, Use Regularization

The regularization technique is to add another quantity to the slicing function (loss).

$$J_{reg}(\theta) = J(\theta) + \lambda R(\theta)$$

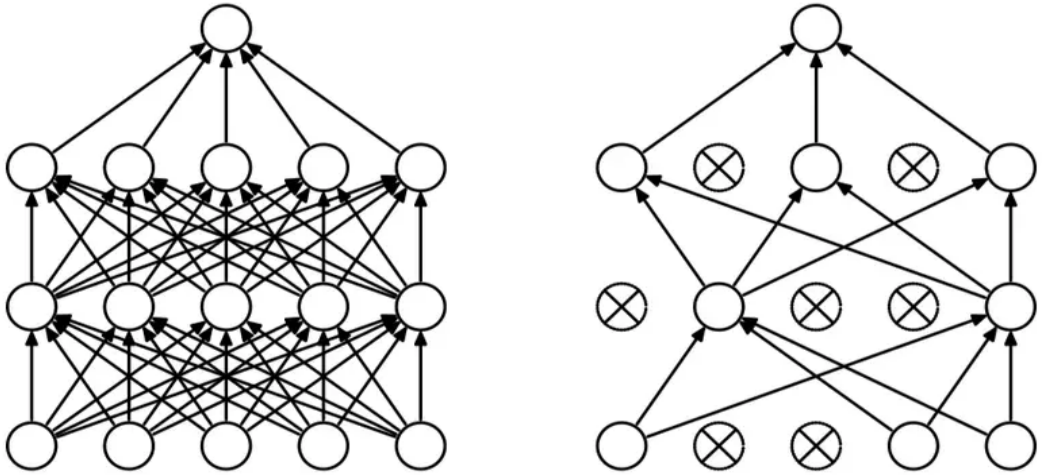
This quantity will affect the loss function. Specifically: if the lamda is large, the effect of the added quantity on the loss function is also large and vice versa if the lamda is small, its effect on the loss function is also small. But the lamda should not be too large because if it is too large, the added quantity will overwhelm the loss => the construction model will be wrong (underfitting).

II. Model Evaluation Problems

7. Way to avoid overfitting

c, Use Dropout

Dropout is a technique to help avoid overfitting similar to regularization by removing the random p% node of the layer => making the model less complex (p belongs to [0.2, 0.5]).



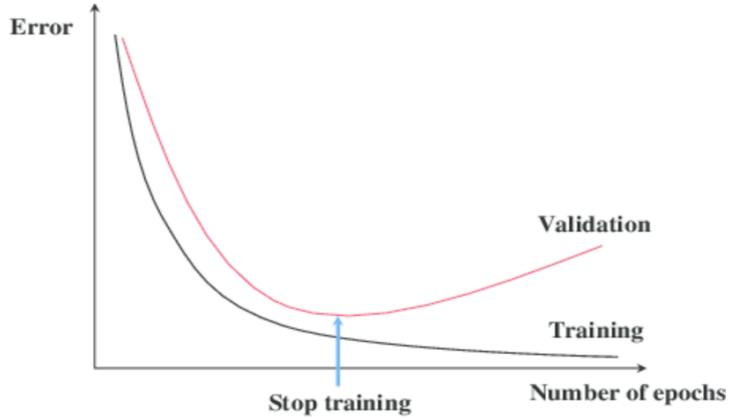
(a) Standard Neural Net (b) After applying dropout.

II. Model Evaluation Problems

7. Way to avoid overfitting

d, Use Early Stopping

When training the model, the loss of the training set and the test set is not always reduced, at a certain epoch, the loss of the training set will continue to decrease, but the loss of the test set will not decrease but increase again. => That is the phenomenon of overfitting. So to prevent it, at that moment people will stop the training (because letting the program continue training will not improve anything but consume resources).



III. Model Evaluation Methods

1. Phương pháp Hold-out

- The Hold-out method divides the data set into two independent sets. For example, training set 2/3, testing set 1/3.

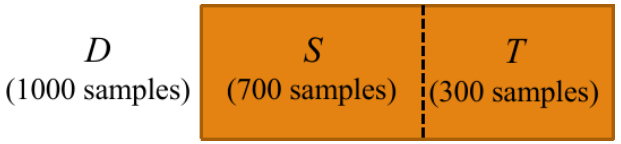
- This method is suitable for small data sets. However, the samples may not be representative of the entire data (missing classes in the test set).

- We can improve by using sampling method so that each class is evenly distributed in both training and testing data sets. Or random sampling: do holdout k times and accuracy acc(M) = average of k exact values.

$$D = (x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$$

Given a Dataset:
D is divided into two mutually exclusive sets: a training set S, and a testing set T, i.e:

$$D = S \cup T, S \cap T = \oslash$$



III. Model Evaluation Methods

1. Phương pháp Hold-out

- It’s far from enough to fix the proportions of different classes in different sets.
- Different cut of D produces different training set and testing set.
- To evaluate the performance of a learner, we should randomly cut D for example 100 time and test the learner on 100 different testing sets (the size of 100 testing sets should be the same).
- The performance of a learner is the average performance on 100 testing set.

III. Model Evaluation Methods

1. Phương pháp Hold-out

Disadvantage

- If S is too big, the learned model will be more close to the model trained using D. But because T is too small, the evaluation results may be not so accurate.
- If S is small, the learned model may not be fully trained. Then the fidelity of evaluation results will be low.
- Usually 2/3~4/5 samples are used in the training process, and the left ones are used to test.

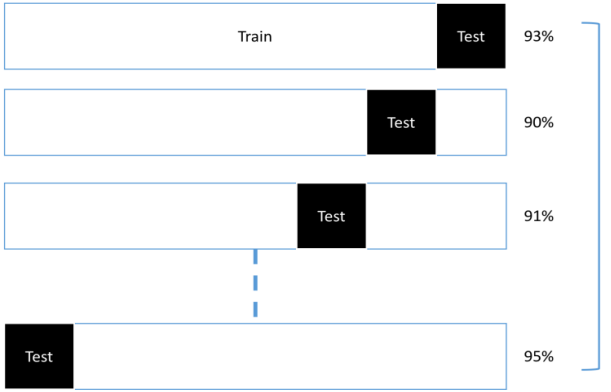
III. Model Evaluation Methods

2. Phương pháp Cross validation

- Also known as k-fold Cross validation. This method divides the data into k subsets of the same size.

- At each loop use a subset of test and the remaining subsets are training sets. The value of k is usually = 10. We can use one of two ways:

  - Leave-one-out : k=number of samples in the data (for small datasets)
  - Stratified cross-validation: use sampling method so that the classes in each subset are distributed as over the entire data.



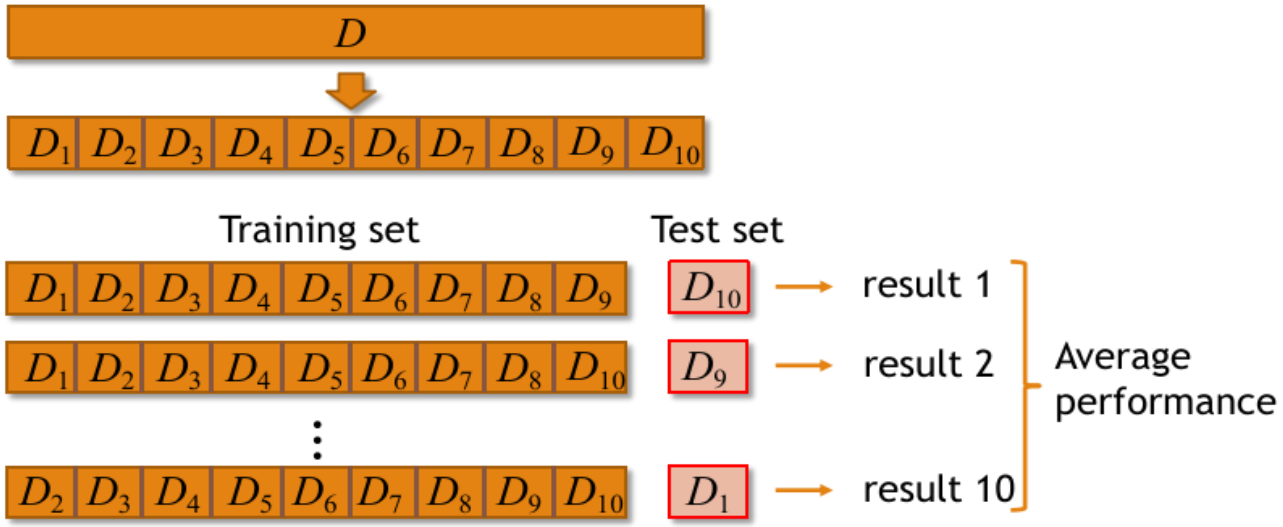
$$Finall\ Accuracy = \frac{1}{n}\sum_{i=1} Accuracy_i$$

III. Model Evaluation Methods

2. Phương pháp Cross validation

Given a Dataset: $D = (x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$

D is divided into k mutually exclusive subset, i.e. $D = D_1 \cup D_2 \cup D_3 \ldots \cup D_n, D_i \cap D_j = \oslash$



IV. Reference

Book:

Evaluating-machine-learning-models.

Q & A