Homework1

Course: Machine Learning

Name: Xiang Lei (雷翔)

Student ID: 2053932

Date: April 2024

Q1. Explain supervised and unsupervised learning, and list two representative algorithms respectively.

Ans. Whether supervised learning depends on whether there are labels of the data.

Supervised learning: The algorithms learn from the labeled data and each training sample consists of input data and the corresponding label. In the training stage, the algorithms aim to learn functions that can map inputs to outputs and further achieve great performance in out-of-sample.

eg. Linear Regression and Support Vector Machines (SVM).

Unsupervised learning: The algorithms learn from the unlabeled data, which means the outputs of the data are unknown. In the training stage, the algorithms aim to find a similar pattern or structure from the data.

eg. K-Means Clustering and Principal Component Analysis (PCA).

Q2. For K-fold cross-validation, if K = the number of samples, it is called the leave one out method. Suppose you have the following data: input and output have only one variable. Use a linear regression model ($y = wx + b$) to fit the data. So what is the mean squared error (MSE) obtained using Leave One Out cross validation?

Ans.

X(independent variable)	Y(dependent variable)
0	2
2	2
3	1

Use the least square method to fit the data:

$$w = \frac{\sum_{i=1}^{N}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{N}(x_i - \bar{x})^2} \tag{1a}$$

$$b = \bar{y} - w \cdot \bar{x} \tag{1b}$$

Case 1: training set: $\{(0, 2), (2, 2)\}$ and validation set: $\{(3, 1)\}$

$\bar{x} = 1$ and $\bar{y} = 2$

so $w_1 = 0$ and $b_1 = 2$, which means $y = 2$.

Therefore, $Loss^1_{MSE} = \frac{\sum(\hat{y}_i - y_i)^2}{n} = 1$.

Case 2: training set: $\{(0,2),(3,1)\}$ and validation set: $\{(2,2)\}$

$\bar{x} = 1.5$ and $\bar{y} = 1.5$

so $w_2 = -\frac{1}{3}$ and $b_2 = 2$, which means $y = -\frac{x}{3} + 2$.

Therefore, $Loss^2_{MSE} = \frac{\sum(\hat{y}_i - y_i)^2}{n} = \frac{4}{9}$.

Case 3: training set: $\{(2,2),(3,1)\}$ and validation set: $\{(0,2)\}$

$\bar{x} = 2.5$ and $\bar{y} = 1.5$

so $w_3 = -1$ and $b_3 = 4$, which means $y = -x + 4$.

Therefore, $Loss^3_{MSE} = \frac{\sum(\hat{y}_i - y_i)^2}{n} = 4$.

Average Loss: $\frac{1+\frac{4}{9}+4}{3} = \frac{49}{27}$

So the mean squared error obtained using Leave-One-Out cross-validation is $\frac{49}{27}$.

Q3. Let us take an example of actual demand and forecasted demand for a brand of ice creams in a shop in a year. In the table, M = Month, AD = Actual Demand, FD = Forecasted Demand.

Please calculate the MAE(mean absolute error) and MSE(mean square error).

Ans.

M	1	2	3	4	5	6	7	8	9	10	11	12
AD	42	45	49	55	57	60	62	58	54	50	44	40
FD	44	46	48	50	55	60	64	60	53	48	42	38

The MAE and MSE calculations are as follows:

$$Loss_{MAE} = \frac{\sum|\hat{y}_i - y_i|}{n} = \frac{\sum|FD - AD|}{12} \tag{2a}$$

$$Loss_{MSE} = \frac{\sum(\hat{y}_i - y_i)^2}{n} = \frac{\sum(FD - AD)^2}{12} \tag{2b}$$

$$Loss_{MAE} = \frac{2+1+1+5+2+0+2+2+1+2+2+2}{12} = \frac{11}{6}$$

$$Loss_{MSE} = \frac{4+1+1+25+4+0+4+4+1+4+4+4}{12} = \frac{14}{3}$$

```
import numpy as np


AD = np.array([42, 45, 49, 55, 57, 60, 62, 58, 54, 50, 44, 40])
FD = np.array([44, 46, 48, 50, 55, 60, 64, 60, 53, 48, 42, 38])
```

```
mae = np.mean(np.abs(AD - FD))
print("Mean Absolute Error (MAE):", mae)

mse = np.mean((AD - FD) ** 2)
print("Mean Squared Error (MSE):", mse)
```

Q4. Consider a triple classification problem that requires the recognition of three classes, A, B, and C. Given the confusion matrix, please complete the following calculations. (1) Calculate the precision and recall for each class respectively. (2) Use both Macro average and Weighted average to calculate the precision and recall of the whole classifier. If you are not sure about these concepts, use a search engine. Retain 4 decimal places (only for (2)).

Ans.

Actual \ Predicted	**A**	**B**	**C**
A	40	20	10
B	35	85	40
C	0	10	20

(1) The precision and recall can be calculated as follows:

$$Precision = \frac{TP}{TP + FP} \tag{3a}$$

$$Recall = \frac{TP}{TP + FN} \tag{3b}$$

For **Class A**: $TP = 40$, $FN = 30$, $FP = 35$, and $TN = 155$

So, $Precision_A = \frac{40}{40+35} = \frac{8}{15}$ and $Recall_A = \frac{40}{40+30} = \frac{4}{7}$.

For **Class B**: $TP = 85$, $FN = 75$, $FP = 30$ and $TN = 70$

So, $Precision_B = \frac{85}{85+30} = \frac{17}{23}$ and $Recall_B = \frac{85}{85+75} = \frac{17}{32}$.

For **Class C**: $TP = 20$, $FN = 10$, $FP = 50$, and $TN = 180$

So, $Precision_C = \frac{20}{20+50} = \frac{2}{7}$ and $Recall_C = \frac{20}{20+10} = \frac{2}{3}$.

(2) $Macro\_Precision = \frac{1}{n}\sum_{i=1}^{n} P_i = \frac{\frac{8}{15}+\frac{17}{23}+\frac{2}{7}}{3} = 0.5194$

$Macro\_Recall = \frac{1}{n}\sum_{i=1}^{n} R_i = \frac{\frac{4}{7}+\frac{17}{32}+\frac{2}{3}}{3} = 0.5898$

$Weighted\_Macro\_Precision = \sum_{i=1}^{n} w_i \times Precision_i = \frac{70\times\frac{8}{15}+160\times\frac{17}{23}+30\times\frac{2}{7}}{70+160+30} = 0.6314$

$Weighted\_Macro\_Recall = \sum_{i=1}^{n} w_i \times Recall_i = \frac{70\times\frac{4}{7}+160\times\frac{17}{32}+30\times\frac{2}{3}}{70+160+30} = 0.5577$

```python
import numpy as np

confusion_matrix = np.array([[40, 20, 10], [35, 85, 40], [0, 10, 20]])

TP = np.diag(confusion_matrix)
FN = np.sum(confusion_matrix, axis=1) - TP
FP = np.sum(confusion_matrix, axis=0) - TP
TN = np.sum(confusion_matrix) - (TP + FN + FP)

for i in range(len(TP)):
    print(f"For class {i+1}:")
    print("TP:", TP[i])
    print("FN:", FN[i])
    print("FP:", FP[i])
    print("TN:", TN[i])

precision = np.diag(confusion_matrix) / np.sum(confusion_matrix, axis=0)
recall = np.diag(confusion_matrix) / np.sum(confusion_matrix, axis=1)

print("Precision for each class:", precision)
print("Recall for each class:", recall)

macro_precision = np.mean(precision)
macro_recall = np.mean(recall)

weighted_precision = np.sum(
    precision * np.sum(confusion_matrix, axis=1) / np.sum(confusion_matrix)
)
weighted_recall = np.sum(
    recall * np.sum(confusion_matrix, axis=1) / np.sum(confusion_matrix)
)

print("Macro-average Precision:", round(macro_precision, 4))
print("Macro-average Recall:", round(macro_recall, 4))
print("Weighted-average Precision:", round(weighted_precision, 4))
print("Weighted-average Recall:", round(weighted_recall, 4))
```