Hand-written Part

学号: t11902210 姓名: 张一凡 (交换生)

code question report 在后面

Problem 1

Consider a binary classification data set $\{(x_n, y_n)\}_{n=1}^N$ with $x_n \in R^d$ and $y_n \in \{-1, 1\}$. For any weight vector w within a linear model, define an error function

$$err(w^T x, y) = (\max(1 - yw^T x, 0))^2$$

That is, $$E_{in}(w) = \frac{1}{N} \sum_{n=1}^{N} (\max(1 - y_n w^T x_n, 0))^2$$

Running gradient descent to optimize $E_{in}(w)$ requires calculating its gradient direction $\nabla E_{in}(w)$ (and then move opposite to that direction). What is $\nabla E_{in}(w)$?

$$\nabla E_{in}(w) = \frac{\partial E_{in}(w)}{\partial w} = \frac{1}{N} \sum_{n=1}^{N} \cancel{\max(1 - y_n w^T x_n, 0)} \cdot \frac{\partial f(w)}{\partial w}$$

$$f(w) = [\max(1 - y_n w^T x_n, 0)]^2$$

$$= \begin{cases} (1 - y_n w^T x_n)^2 & (y_n w^T x_n \le 1) \\ 0 & (y_n w^T x_n > 1) \end{cases}$$

$$\frac{\partial f(w)}{\partial w} = \begin{cases} -2 y_n x_n (1 - y_n w^T x_n) & (y_n w^T x_n \le 1) \\ 0 & (y_n w^T x_n > 1) \end{cases}$$

$$\cancel{\nabla E_{in}(w)} = -2 y_n x_n (\max((1 - y_n w^T x_n), 0))$$

$$\nabla E_{in}(w) = \begin{cases} -\frac{2}{N} \sum_{n=1}^{N} y_n x_n (1 - y_n w^T x_n) & (y_n w^T x_n \le 1) \\ 0 & (y_n w^T x_n > 1) \end{cases}$$

$$= -\frac{2}{N} \sum_{n=1}^{N} y_n x_n (\max(1 - y_n w^T x_n, 0))$$

Problem 2.

Consider a process that generates d-dimensional vectors $x_1, x_2 \dots x_N$ independently from a multivariate Gaussian distribution $N(u, I)$, where $u \in R^d$ is an unknown parameter vector and $I \in R^{d \times d}$ is an identity matrix. The maximum likelihood estimate of u is

$$u^* = \underset{u \in R^d}{\operatorname{argmax}} \prod_{n=1}^{N} p_u(x_n).$$

where p_u is the probability density function of $N(u, I)$. Prove that

$$u^* = \frac{1}{N} \sum_{n=1}^{N} x_n.$$

Gaussian distribution (d)

$N(u, I)$ definition $\Rightarrow$ $p_u(x_n) = \frac{1}{\sqrt{(2\pi)^d |I|}} e^{-\frac{1}{2}(x_n-u)^T I^{-1}(x_n-u)}$

$\left(x \sim N(u, \Sigma) \quad f(x) = \frac{1}{\sqrt{(2\pi)^n |\Sigma|}} e^{-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)} \right)$

maximum likelihood:

$u^* = \arg\max \prod_{n=1}^{N} p_u(x_n) \qquad u_0 = \arg\max \prod_{n=1}^{N} p_u(x_n)$

turn "multiply" to "add" use "log"

$\log u_0 = \sum_{n=1}^{N} \log p_u(x_n) \rightarrow$ find max

$\log p_u(x_n) = \log \frac{1}{\sqrt{(2\pi)^d |I|}} + \left[-\frac{1}{2}(x_n-u)^T I^{-1}(x_n-u) \right]$

$= -\frac{1}{2}\log(2\pi)^d |I| - \frac{1}{2}(x_n-u)^T I^{-1}(x_n-u)$

$= -\frac{d}{2}\log 2\pi - \frac{1}{2}\log|I| - \frac{1}{2}(x_n-u)^T I^{-1}(x_n-u)$

$\rightarrow$ find max

上述问题等价于 $-\log p_u(x_n)$ 的 min.

令 $f_u(x_n) = \frac{d}{2}\log 2\pi + \frac{1}{2}\log|I| + \frac{1}{2}(x_n-u)^T I^{-1}(x_n-u)$

$\sum \log p_u(x_n) = \frac{dN}{2}\log 2\pi + \frac{N}{2}\log|I| + \frac{1}{2}\sum (x_n-u)^T I^{-1}(x_n-u)$

固定值

忽略常数. 令 $\sum (x_n-u)^T I^{-1}(x_n-u)$ 最小, (I 为单位阵)

令 $g(u) = \sum (x_n-u)^T I^{-1}(x_n-u) = \sum \|x_n-u\|^2$

~~$\frac{\partial \sum \|x_n-u\|^2}{\partial u} = \sum 2(x_n-u) = 0 \Rightarrow u = x_n$~~

$\frac{\partial g(u)}{\partial u} = -2\sum (x_n-u) = 0$

$\sum_{i=1}^{N}(x_n-u) = \sum_{i=1}^{N} x_n - Nu = 0$

得 $u = \frac{1}{N}\sum_{i=1}^{N} x_n \Rightarrow u^* = \frac{1}{N}\sum_{i=1}^{N} x_n$

Problem 3:

A classic binary classification data set that cannot be seperated by any line is called the XOR data set. with

$x = [x_1, x_2]$	y
$x_1 = [+1, +1]$	$y_1 = -1$
$x_2 = [-1, +1]$	$y_2 = +1$
$x_3 = [-1, -1]$	$y_3 = -1$
$x_4 = [+1, -1]$	$y_4 = +1$

You can see why it is called XOR by Interpreting +1 as a boolean value value of True and -1 as FALSE.

Consider a second-order feature transform $\Phi_2(x) = (1, x_1, x_2, x_1^2, x_1x_2, x_2^2)$ that converts the data set to

$z=\Phi_2(x)$	y
$z_1=\Phi_2(x_1)$	$y_1=-1$
$z_2=\Phi_2(x_2)$	$y_2=+1$
$z_3=\Phi_2(x_3)$	$y_3=-1$
$z_4=\Phi_2(x_4)$	$y_4=+1$

Show a perception $\tilde{w}$ in the z-space that seperates the data. That is $y_n=\text{sign}(\tilde{w}^T z_n)$ for $n=1,2,3,4$.

Then, plot the classification boundary $\tilde{w}^T\Phi_2(x)=0$ in the x-space. Your boundary should look like a quadratic curve that classifies x_1, x_2, x_3, x_4 perfectly.

解: 将 $x_1 \cdots x_4$ 代入 $z_1=z_3$ (特征转换)

$$\begin{cases} z_1=\Phi_2([+1,+1])=(1,1,1,1,1,1)^T \Rightarrow y_1=-1 \\ z_2=\Phi_2([-1,+1])=(1,-1,1,1,-1,1)^T \Rightarrow y_2=+1 \\ z_3=\Phi_2([-1,-1])=(1,-1,-1,1,1,1)^T \Rightarrow y_3=-1 \\ z_4=\Phi_2([+1,-1])=(1,1,-1,1,-1,1)^T \Rightarrow y_4=+1 \end{cases}$$

$$\text{sign}(x)=\begin{cases} 1 & x>0 \\ 0 & x=0 \\ -1 & x<0 \end{cases}$$

先令 $\tilde{w}=(0,0,0,0,0,0)^T$

① $\tilde{w}^T z_1=0 \neq(-1) \Rightarrow \tilde{w}=\tilde{w}+y_1 z_1=(-1,-1,-1,-1,-1,-1)^T$ (不同号)

② $\tilde{w}^T z_2=-1+1-1-1+1-1=-2$ 与 $+1$ 不同号

$\Rightarrow \tilde{w}=\tilde{w}+y_2 z_2=(0,-2,0,0,-2,0)^T$

③ $\tilde{w}^T z_3=0+2+0+0-2+0=0$ 与 (-1) 不同号

$\tilde{w}=\tilde{w}+y_3 z_3=(-1,-3,1,-1,-3,-1)^T$

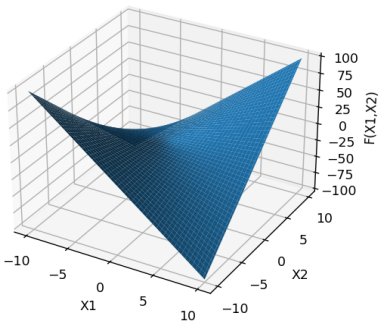
④ $\tilde{w}^T z_4=-1-3-1-1+4-1=-3$ 与 y_4 不同号

$\tilde{w}=\tilde{w}+y_4 z_4=(0,-2,0,0,-4,0)^T$

权重向量 $\tilde{w}=(0,-2,0,0,-4,0)^T$ 时满足

$\tilde{w}^T\cdot\Phi_2(x)=-2x_1-4x_1x_2=0 \Rightarrow x_1x_2=0$ 成立

$x_1=0$ 或 $x_2=0$



The left side shows the final image I drew using Python and related drawing libraries.

The mathematical expression for the image is below.

f(x1,x2)=x1*x2

Problem. 4

Consider building binary classification with AdaBoost algorithm. a weak classifier $g_t(x)$ is trained on a data set $\{x_n, y_n\}_{n=1}^N$ with weights $\{w_n^t\}_{n=1}^N$ at the time step t. The error rate is defined as $\varepsilon_t=\frac{\sum_{n=1}^N w_n^t \cdot \delta(g_t(x_n), y_n)}{\sum_{n=1}^N w_n^t}$, where $\delta(g_t(x_n), y_n)=1$ if $g_t(x_n)\neq y_n$ and $\delta(g_t(x_n), y_n)=0$ otherwise. For the next time step. the data set is reweighted to emphasize on misclassified samples through the following rules

$$w_n^{t+1} = \begin{cases} w_n^t \cdot d_t & \text{if } g_t(x_n) \neq y_n \\ w_n^t / d_t & \text{if } g_t(x_n) = y_n \end{cases}$$

Show that $d_t = \sqrt{(1-\varepsilon_t)/\varepsilon_t}$ can degrade the previous classifier $g_t(x)$ and make its error rate become 0.5 with new weights $\{w_n^{t+1}\}_{n=1}^N$:

$$\frac{\sum_{n=1}^N w_n^{t+1} \delta(g_t(x_n), y_n)}{\sum_{n=1}^N w_n^{t+1}} = 0.5$$

Sol:

$$\varepsilon_t = \frac{\sum_{n=1}^N w_n^t \cdot \delta(g_t(x_n), y_n)}{\sum_{n=1}^N w_n^t}$$

$$w_n^{t+1} = \begin{cases} w_n^t \cdot d_t & (g_t(x_n) \neq y_n) \\ w_n^t / d_t & (g_t(x_n) = y_n) \end{cases} \qquad d_t = \sqrt{(1-\varepsilon_t)/\varepsilon_t}$$

$$d_t = \sqrt{(1-\varepsilon_t)/\varepsilon_t} = \sqrt{\frac{1}{\varepsilon_t} - 1}$$

$$w_n^{t+1} = w^t \cdot e$$

$$= \sqrt{\frac{\sum_{n=1}^N w_n^t}{\sum_{n=1}^N w_n^t \delta(g_t(x_n), y_n)} - 1}$$

$$= \sqrt{\frac{\sum w_n^t (1 - \delta(g_t(x_n), y_n))}{\sum_{n=1}^N w_n^t \delta(g_t(x_n), y_n)}} = \cancel{\sqrt{\quad}}$$

$$\frac{\sum_{n=1}^N w_n^{t+1} \delta(g_t(x_n), y_n)}{\sum_{n=1}^N w_n^{t+1}} = \frac{\sum_{n=1}^N w_n^t \cdot d_t \delta(g_t(x_n), y_n)}{\sum_{n=1}^N w_n^{t+1}} = \frac{\sum_{n=1}^N w_n^t \cdot \delta(g_t(x_n), y_n)}{\sum_{n=1}^N w_n^{t+1}} \cdot d_t$$

$$= \frac{\sum_{n=1}^N w_n^t \delta(g_t(x_n), y_n)}{\sum_{n=1}^N w_n^t} \cdot \frac{\sum w_n^t}{\sum_n^{t+1}} \cdot d_t \qquad d_t \cdot \varepsilon_t \cdot \frac{\sum w_n^t}{\sum_n^{t+1}}$$

$$= \varepsilon_t \cdot \frac{\sum w_n^t}{\sum_n^{t+1}} \cdot \sqrt{(1-\varepsilon_t)/\varepsilon_t} \cdot d_t = \cancel{\sqrt{\quad}}$$

$$\sum w_n^{t+1} d_t = \sum \left(w_n^t \cdot d_t \cdot \delta(g_t(x_n), y_n) + \frac{w_n^t}{d_t} (1 - \delta(g_t(x_n), y_n)) \right)$$

$$= \sum w_n^t \left(d_t - \frac{1}{d_t}\right) \delta(g_t(x_n), y_n) + \sum w_n^t \cdot \frac{1}{d_t}$$

$$= \left(d_t - \frac{1}{d_t}\right) \cdot \varepsilon_t \cdot \sum w_n^t + \frac{\sum w_n^t}{d_t}$$

$$\frac{\sum w_n^{t+1}}{\sum w_n} = \frac{\left(d_t - \frac{1}{d_t}\right) \cdot \varepsilon_t + 1}{d_t} = \frac{1 + \left(\frac{1-\varepsilon_t}{d_t}\right) \cdot \varepsilon_t}{d_t} = \varepsilon_t$$

$$\text{Ans} = \sqrt{(1-\varepsilon_t)\varepsilon_t} \cdot d_t \cdot \varepsilon_t \cdot \frac{d_t}{(d_t^2 - 1)\varepsilon_t + 1} = \frac{d_t^2 \cdot \varepsilon_t}{(d_t^2 - 1)\varepsilon_t + 1}$$

$$= \frac{1-\varepsilon_t}{1 - \varepsilon_t - \varepsilon_t + 1} = \frac{1-\varepsilon_t}{2 - 2\varepsilon_t} = \boxed{\frac{1}{2}}$$

$$= \frac{d_t^2}{d_t^2 - 1 + \frac{1}{\varepsilon_t} - 1} = \frac{1}{2} = 0.5$$

格式 Report

见后

Code Part Problem Report

學號：t11902210 姓名：張一凡

（a）Compare the performance of the linear, nonlinear, and random forest models on the classification and regression tasks. Based on the evaluation metrics (accuracy for classification and mean squared error for regression), which model performs better for each task, and why do you think this is the case?

Answer：

First, I will show the results of running the code below. I am using normalized processing in this section.(I used normalization in the final submitted code)

```
Logistic Regression Accuracy: 0.6888888888888889
Decision Tree Classifier Accuracy: 0.8888888888888888
Random Forest Classifier Accuracy: 0.9111111111111111
Linear Regression MSE: 39.390567085984436
Decision Tree Regressor MSE: 28.341570306709183
Random Forest Regressor MSE: 23.413483640603484
```

For the classification problem, the accuracy of the linear model (logical regression) is about 0.69, the accuracy of the nonlinear model decision tree is about 0.89, and the accuracy of the model fusion random forest is 0.91. It can be seen that the performance of decision tree and random forest on iris dataset is much higher than that of linear model, which indicates that the relationship between iris dataset features and targets is nonlinear. If the relationship between data is linear, the classification performance of the linear model may be better. However, in this case, the nonlinear model performs better, indicating that the true relationship of the data is nonlinear. Therefore, extended models such as decision trees will achieve better classification results.

For the regression problem, the MSE of the linear model is about 39.4, the MSE of the nonlinear model in the decision tree is 28.3, and the MSE of the fusion model random forest is 23.4. In regression problems, MSE represents variance, and the smaller the variance, the better. In this problem, it has the same effect as the classification problem, indicating that the relationship between the features of the Boston housing dataset and the target variable is biased towards nonlinearity. Alternatively, nonlinear models can better capture the linear relationship between features and target variables than linear models. In addition, the linear model may also perform overfitting on the data set, resulting in the regression effect not as good as the nonlinear model. However, it should be noted that this is only the result of a specific dataset and model, and cannot be generalized to all datasets and patterns. Finally, random forest does have a better impact on regression than a single decision tree, so model fusion can indeed reduce noise and achieve better results.

（b）Apply both normalization and standardization techniques to the datasets and compare their impact on the performance of one of the models (e.g., logistic regression). Explain the rationale behind each technique, and discuss their advantages and disadvantages in the context of the given tasks.

Answer：

Normalization and standardization are two basic methods of data preprocessing. Normalization scales the data to the range of [0,1], while standardization scales the data to the standard normal distribution with a mean of 0 and a standard deviation of 1. The purpose of these two methods is to make the data have similar scales and distributions to avoid affecting model performance due to significant data differences. The following are the normalization and standardization I have implemented in the code. (I used normalization in the final submitted code)

```python
def normalize(X: np.ndarray) -> np.ndarray:
    # Normalize features to [0, 1]
    # You can try other normalization methods, e.g., z-score, etc.
    # TODO: 1%
    # normalization
    X_min = np.min(X, axis=0)
    X_max = np.max(X, axis=0)
    X_result = (X - X_min) / (X_max - X_min)
    return X_result
    #  standardization
    #X_mean = np.mean(X, axis=0)
    #X_std = np.std(X, axis=0)
    #X_result = (X - X_mean) / X_std
    #return X_result
    #raise NotImplementedError
```

The following is a presentation of the results of normalization and standardization, with normalization above and standardization below.

normalization

```
Logistic Regression Accuracy: 0.6888888888888889
Decision Tree Classifier Accuracy: 0.8888888888888888
Random Forest Classifier Accuracy: 0.9111111111111111
Linear Regression MSE: 39.390567085984436
Decision Tree Regressor MSE: 28.341570306709183
Random Forest Regressor MSE: 23.413483640603484
```

Standardization

```
Logistic Regression Accuracy: 0.9555555555555556
Decision Tree Classifier Accuracy: 0.8222222222222222
Random Forest Classifier Accuracy: 0.8444444444444444
Linear Regression MSE: 22.397267548141315
Decision Tree Regressor MSE: 19.39957401761841
Random Forest Regressor MSE: 10.974551329592613
```

For linear models such as logistic regression, it can be seen from the results shown above that the accuracy of normalization is lower than that of standardization, indicating that standardization is more commonly used in this type of model. Standardization can eliminate deviations between data features, making the importance of each feature more equal in its impact on the model. Normalization is more suitable for nonlinear models, such as the decision tree and random forest showed above, because they are sensitive to the range of data.

Normalization and standardization have their own advantages and disadvantages. The advantage of normalization is that it is simple to understand and easy to calculate, but it may be affected by extreme values. Therefore, extreme values need to be detected and processed in the data, which requires other data processing to achieve better results. However, overall, the application scenarios of normalization are limited; In this task, there were many extreme

values in the regression dataset, which resulted in lower accuracy in the linear model. Standardization better maintains the sample spacing and is more consistent with statistical assumptions (for a numerical feature, it is very likely to follow a normal distribution), so it can avoid the impact of extreme values. This method is suitable for data sets with large sample sizes and many data noises. In this task, the regression data set has more extreme values, and both noise and sample size meet this standard, Therefore, good results were obtained in both linear and nonlinear models; However, because it will convert the data into a standard normal distribution, it may change the distribution of data, leading to the performance degradation of some nonlinear models. In addition, standardization may also increase the complexity of data processing, as it requires calculating the mean and standard deviation of each feature.

（c）Train your logistic or linear regression model using two different configurations of learning rate and number of iterations. Compare the performance of these configurations using the evaluation metrics (accuracy for classification and mean squared error for regression). Discuss how the choice of learning rate and the number of iterations affect the convergence and overall performance of the models.

Answer：

I used logistic regression and linear regression models as testing standards, and conducted two experiments to improve the learning rate and iteration number, respectively. The results were compared with the default values, and their performance in classification and regression was observed through accuracy and MSE. The specific code and related results are as follows. I am using standardized processing in this section. (In the final submitted code hw3.py, I used normalization)

Classification

```
logistic_regression = LinearModel(learning_rate=0.05, iterations=1000, model_type="logistic")
logistic_regression.fit(X_train, y_train)
y_pred = logistic_regression.predict(X_test)
print("Logistic Regression (raise learning_rate) Accuracy:", accuracy(y_test, y_pred))

logistic_regression = LinearModel(learning_rate=0.01, iterations=2000, model_type="logistic")
logistic_regression.fit(X_train, y_train)
y_pred = logistic_regression.predict(X_test)
print("Logistic Regression (raise iterations) Accuracy:", accuracy(y_test, y_pred))
```

```
Logistic Regression Accuracy: 0.9555555555555556
Logistic Regression (raise learning_rate) Accuracy: 0.9333333333333333
Logistic Regression (raise iterations) Accuracy: 0.8666666666666667
```

This section shows the relevant codes and final results of the adjustment parameters for the classification assignment. I used a comparison method to replace the default values of iteration times and learning rate and obtained corresponding accuracy results. I processed the learning rate by 5 times and the number of iterations by 2 times, but the final results were not as good as the default values. Combined with relevant data, I think it is the overfitting problem caused by too high weights and iterations. In practice, choosing the appropriate learning rate and iteration number usually requires experimentation and adjustment. If the learning rate is too high, it may lead to excessive weight updates, resulting in the model being unable to converge; If the learning rate is too low, it may require more iterations to achieve optimal performance. Similarly, if the number of iterations is too small, the model may not be fully trained, resulting in underfitting; If the number of iterations is too many, it may lead to

overfitting or too long training time.

Regression

```
linear_regression = LinearModel(learning_rate=0.002, iterations=1000,model_type="linear")
linear_regression.fit(X_train, y_train)
y_pred = linear_regression.predict(X_test)
print("Linear Regression (low learning_rate) MSE:", mean_squared_error(y_test, y_pred))

linear_regression = LinearModel(learning_rate=0.01, iterations=500,model_type="linear")
linear_regression.fit(X_train, y_train)
y_pred = linear_regression.predict(X_test)
print("Linear Regression (low iterations) MSE:", mean_squared_error(y_test, y_pred))
```

```
Linear Regression MSE: 22.413965536955917
Linear Regression (low learning_rate) MSE: 23.17072875763762
Linear Regression (low iterations) MSE: 21.960636833488184
```

This section shows the relevant codes and final results of the adjustment parameters for the regression assignment. I used a comparison method to replace the default values of iteration times and learning rate and obtained the corresponding MSE results. I processed the learning rate and iteration times by one-fifth and one-half times, respectively (corresponding to the previous formation). The MSE that reduced the iteration times decreased, indicating better results, while the learning rate decreased. As a comparison with the previous one, I believe that the default values given in the assignment are exactly around the best case, so the changes will immediately have a swaying effect. Therefore, when choosing the learning rate and the number of iterations, we need to weigh the trade-off between the convergence speed and the final performance and try several different configurations to find the best hyperparameter combination. In addition, techniques such as learning rate scheduling and early stop can be used to further optimize the performance of the model.

（d）Discuss the impact of hyperparameters, such as the number of trees and maximum depth, on the performance of the random forest model for both classification and regression tasks. How do these hyperparameters affect the model's complexity, generalization, and potential for overfitting? Include a brief explanation of how you selected or tuned these hyperparameters in your implementation.

Answer：

In this problem, I did the corresponding treatment similar to that of a problem, tested the classification and regression of the random forest model, and the number of trees is equal to n_ Estimators, I have made improvements and reductions in two types of problems, and have obtained the final results and discussions, as follows. I am using standardized processing in this section. (In the final submitted code hw3.py, I used normalization)

Classification

```
random_forest_classifier = RandomForest(n_estimators = 500,max_depth = 5,model_type="classifier")
random_forest_classifier.fit(X_train, y_train)
y_pred = random_forest_classifier.predict(X_test)
print("Random Forest Classifier (raise number) Accuracy:", accuracy(y_test, y_pred))

random_forest_classifier = RandomForest(n_estimators = 100,max_depth = 10,model_type="classifier")
random_forest_classifier.fit(X_train, y_train)
y_pred = random_forest_classifier.predict(X_test)
print("Random Forest Classifier (raise depth) Accuracy:", accuracy(y_test, y_pred))
```

```
Random Forest Classifier Accuracy: 0.8888888888888888
Random Forest Classifier (raise number) Accuracy: 0.8444444444444444
Random Forest Classifier (raise depth) Accuracy: 0.9111111111111111
```

In this section of classification, I increased the number of trees by five times and the depth of trees by two times, respectively. The accuracy of the former decreases while the accuracy of the latter increases. Through searching for relevant data, I think that when I choose 5 times the number, overfitting occurs, while the depth of 2 times effectively avoids underfitting and approaches the best model. In general, the adjustment of these hyperparameters has an important impact on the performance of the model. When the number of decision trees increases, the complexity of the model will increase, and it is easy to overfit, but the performance can be improved, especially in the classification problem. On the contrary, when the maximum depth increases, the complexity of the model will also increase, but there may be overfitting on the training set, resulting in a decline in performance on the test set.

Regression

```
random_forest_regressor = RandomForest(n_estimators = 50,max_depth = 5,model_type="regressor")
random_forest_regressor.fit(X_train, y_train)
y_pred = random_forest_regressor.predict(X_test)
print("Random Forest Regressor (low number) MSE:", mean_squared_error(y_test, y_pred))

random_forest_regressor = RandomForest(n_estimators = 100,max_depth = 2,model_type="regressor")
random_forest_regressor.fit(X_train, y_train)
y_pred = random_forest_regressor.predict(X_test)
print("Random Forest Regressor (low depth) MSE:", mean_squared_error(y_test, y_pred))
```

```
Random Forest Regressor MSE: 10.974551329592613
Random Forest Regressor (low number) MSE: 11.32866495228553
Random Forest Regressor (low depth) MSE: 22.931515671649777
```

In this part of the regression, I reduced the number and depth of trees by half. Both MSEs increased, indicating a decrease in effectiveness. Through searching for relevant data and comparing it with the classification task, I think that there has been underfitting, and the best simulation effect can be considered to approach when the relevant hyperparameter is moderately increased. In general, we need to balance the performance and complexity of the model when selecting a hyperparameter. An overly simple model may be underfitting and unable to capture complex relationships in the data; However, too complex models are prone to overfitting, resulting in performance degradation on the test set. Therefore, the selection of appropriate hyperparameters is crucial to the performance and generalization ability of the model.

（e）Analyze the strengths and weaknesses of the implemented models. In what scenarios would you choose to use a linear model over a nonlinear model, or a random forest model, and vice versa? Discuss the trade-offs between model complexity, interpretability, and performance.

Answer：

Below, I will first introduce the advantages, disadvantages, and applicable scenarios of each model used in this assignment. I have referred to relevant information on the internet in this summary section and have already marked it in the corresponding location. I am using standardized processing in this section. (In the final submitted code hw3.py, I used normalization)

Linear model part:

The advantage of logistic regression is that it has a fast training speed and the computational complexity is only related to the number of features during classification; Simple and easy to understand, the interpretability of the model is very good, and the weight of features can show the impact of different features on the final result; Suitable for binary classification problems without the need to scale input features; The memory resource occupation is small because only the feature values of each dimension need to be stored. But it is also sensitive to multicollinearity data; It is difficult to handle the issue of data imbalance; The accuracy is not very high because the form is very simple (very similar to linear models), making it difficult to fit the true distribution of the data; Logistic regression itself cannot filter features. Logistic regression can be used for classification scenarios, especially when the dependent variable is binary, and it can also be applied without requiring a linear relationship between the independent and dependent variables.

The advantage of linear regression is its simple thinking and easy implementation. Modeling is fast, effective for small amounts of data and simple relationships, and is the foundation of many powerful nonlinear models; The linear regression model is very easy to understand, and the results have good interpretability, which is conducive to decision analysis; Contains many important ideas in machine learning. But at the same time, it is difficult to model nonlinear data or polynomial regression with the correlation between data features, and it is difficult to express highly complex data well, as is the case in the second question. Linear Regression is used to solve regression problems, but its effect is in the primary state among many regression models at present, and regularization variables can be added to prevent overfitting.

https://blog.csdn.net/Katherine_hsr/article/details/79942260?ops_request_misc=&request_id=&biz_id=102&utm_term=linear%E5%9B%9E%E5%BD%92%E7%9A%84%E4%BC%98%E7%82%B9%E5%92%8C%E7%BC%BA%E7%82%B9&utm_medium=distribute.pc_search_result.none-task-blog-2~all~sobaiduweb~default-4-79942260.142^v86^insert_down1,239^v2^insert_chatgpt&spm=1018.2226.3001.4187

Nonlinear model part:

The advantage of the decision tree algorithm is that it is easy to understand and the mechanism is easy to explain; Can be used for small datasets; Not sensitive to missing values; Can handle irrelevant feature data; High efficient, the decision tree only needs to be

constructed once and used repeatedly, and the maximum number of calculations for each prediction does not exceed the depth of the decision tree. However, the disadvantage is that it is difficult to predict continuous fields; Overfitting is easy to occur; When there are too many categories, errors may increase faster; Not perform well when processing data with strong feature correlation. Although the decision tree performs well in this job, the decision tree is more useful as the cornerstone of some more useful algorithms, such as a random forest.

The random forest has great advantages over other algorithms in many current data sets and performs well. It can process very high dimensional data, and does not need to do feature selection, because feature subsets are randomly selected; When creating a random forest, unbiased estimation is used for generalization error, and the model has strong generalization ability; During training, trees are independent of each other, with fast training speed and easy parallelization methods; Not sensitive to missing values, if a significant portion of features are lost, accuracy can still be maintained. But like the decision tree, the random forest will have a greater impact on the random forest for data with different values of attributes when simulating the contract on some noisy classification or regression problems. Like the decision tree, the random forest can be used for both classification and regression problems and is not suitable for scenes requiring high real-time.
https://blog.csdn.net/qq_44910978/article/details/125118140?ops_request_misc=%257B%2522request%255Fid%2522%253A%252216828481151680018852352 3%2522%252C%2522scm%2522%253A%252220140713.130102334.pc%255Fall.%2522%257D&request_id=168284811516800188523523&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~all~first_rank_ecpm_v1~rank_v31_ecpm-4-125118140-null-null.142^v86^insert_down1,239^v2^insert_chatgpt&utm_term=%E9%9A%8F%E6%9C%BA%E6%A3%AE%E6%9E%97%E7%9A%84%E9%80%82%E7%94%A8%E5%9C%BA%E6%99%AF&spm=1018.2226.3001.4187

Finally, let's discuss the trade-off between the complexity, interpretability, and performance of the model.

When choosing between linear or nonlinear models, it is necessary to weigh the characteristics of the dataset and the task requirements. If the data has a simple linear structure or if the interpretability of the model is important, a linear model may be a better choice. If the dataset has a complex nonlinear structure or prediction accuracy is the most important factor, then nonlinear models can be considered. The random forest model is usually applicable to various types of data sets, but attention should be paid to adjusting hyperparameters to balance the complexity and performance of the model. There is a trade-off between model complexity, interpretability, and performance. Simpler models typically have better interpretability and generalization ability, but performance may be limited. The more complex models usually have higher prediction performance, but there may be a risk of overfitting, and the interpretability may be poor. This is also the same as what our teacher said in class.