HW1: Linear Regression using Gradient Descent

In hw1, you need to implement linear regression by using only numpy, then train your implemented model by the provided dataset and test the performance with testing data

Please note that only **NUMPY** can be used to implement your model, you will get no points by simply calling sklearn.linear_model.LinearRegression

In [553...

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

Load data

In [554...

```
train_df = pd.read_csv("train_data.csv")
x_train, y_train = train_df['x_train'], train_df['y_train']
train_df.head()
```

Out[554...

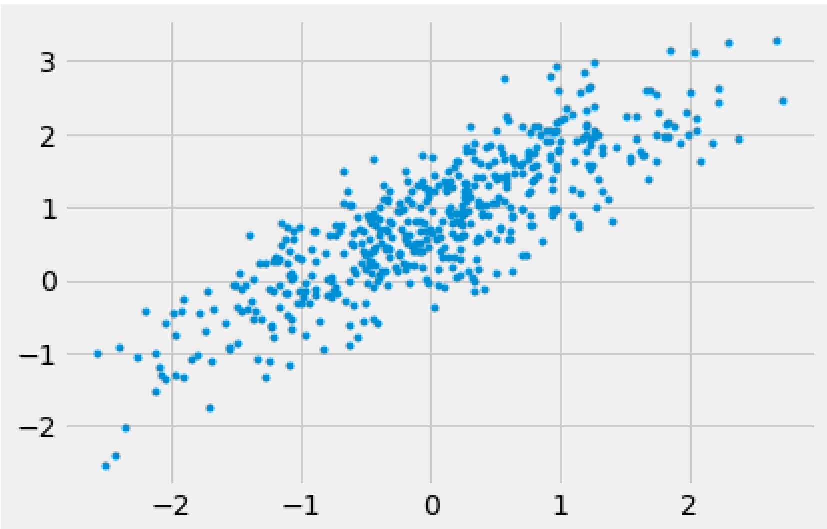
	x_train	y_train
0	-0.135800	0.204852
1	0.325663	1.298015
2	1.589894	1.949912
3	0.643482	1.486206
4	1.995714	2.573797

In [555...

```
plt.plot(x_train, y_train, '.')
```

Out[555...

```
[<matplotlib.lines.Line2D at 0x7f080d9a8e80>]
```



In [556...

```
def read_data(df):
    name = df.columns
    x, y = df[name[0]], df[name[1]]

    x = np.asarray(x).reshape(-1, 1)
```

```
    y = np.asarray(y).reshape(-1, 1)
    return x, y
```

In [557…

```
def my_mse(y_batch, y_pred):
    error = y_batch - y_pred
    mse =  (error ** 2).sum() / error.shape[0]
    return mse
```

In [558…

```
def gd_fit(x, y, learning_rate=1e-6, iteration=50, batch_size=None):
    x = np.hstack((np.ones(x.shape), x))
    n = x.shape[0]
    d = x.shape[1]
    beta = np.random.randn(d, 1)
    beta_log = [beta.copy()]
    mse = []

    for i in range(iteration):
        if batch_size is not None:
            idx = np.random.randint(0, n, batch_size)
#             idx.sort()
        else:
            idx = np.arange(n)
        x_batch = x[idx]
        y_batch = y[idx]
        y_pred = x_batch @ beta
        error = y_batch - y_pred
        beta -= learning_rate * -2 * (x_batch.T @ error) / n
        beta_log.append(beta.copy())
        mse.append(my_mse(y_batch, y_pred))

    return beta, beta_log, mse
```

In [559…

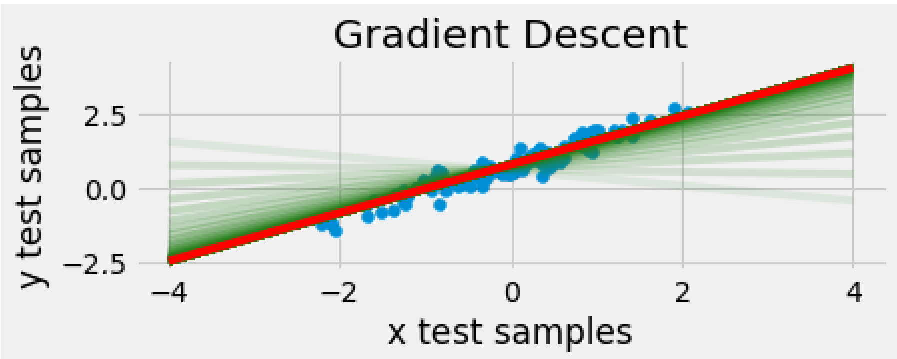
```
def predict(x_test, y_test, beta, verbose=None):
    x_test = np.hstack((np.ones(x_test.shape), x_test))
    y_pred = x_test @ beta
    error = my_mse(y_test, y_pred)

    if verbose is not None:
        print(f'weight: {beta[1][0]}\nintercept: {beta[0][0]}')
    else:
        pass

    return y_pred, error
```

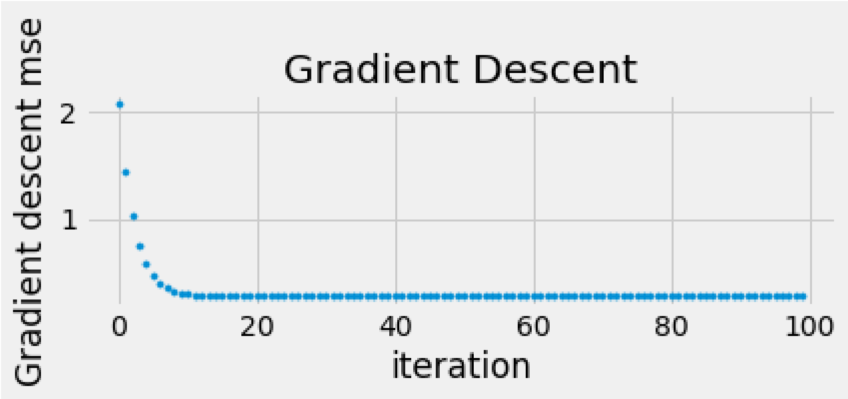
In [561…

```
if __name__ == '__main__':
    train_df = pd.read_csv("train_data.csv")
    x, y = read_data(train_df)

    test_data = pd.read_csv("test_data.csv")

    parameters={
            '0':{'learning_rate':1e-1,'iteration':100,'batch_size':None,'name':'Grad
            '1':{'learning_rate':1e-0,'iteration':100,'batch_size':20,'name':'Mini-B
            '2':{'learning_rate':1e-2,'iteration':100,'batch_size':1,'name':'Stochas
            }

    for key in parameters:
```

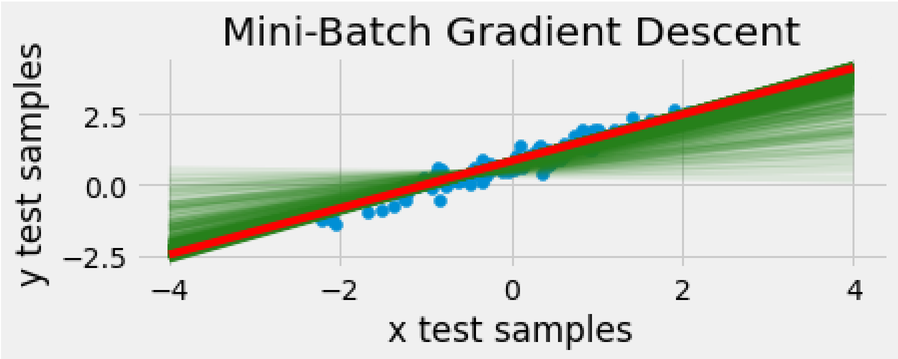
```
    print(f'Parameter(learning rate:{learning_rate}_iteration:{iteration}_batch
    learning_rate = parameters[key]['learning_rate']
    iteration = parameters[key]['iteration']
    batch_size = parameters[key]['batch_size']
    name = parameters[key]['name']

    ## training model
    beta, beta_log, mse = gd_fit(x, y, learning_rate=learning_rate, iteration=it

    upper_bound = np.ceil(np.max(x_test)) + 1
    lower_bound = np.floor(np.min(x_test)) - 1

    x_pred = np.linspace(lower_bound, upper_bound, len(x_test)).reshape(-1, 1)

    ## predict testing data
    y_pred, _ = predict(x_pred, y_test, beta, verbose=1)

    plt.figure()
    plt.subplot(211)
    plt.title(name)
    for i in range(len(beta_log)):
        y_pred, _ = predict(x_pred, y_test, beta_log[i])
        alpha = np.sqrt(i/len(beta_log))
        plt.plot(x_pred.ravel(), y_pred.ravel(), c=(0.15, 0.5, 0.1, alpha))


    plt.scatter(x_test, y_test)
    plt.plot(x_pred.ravel(), y_pred.ravel(), c='red')
    plt.xlabel('x test samples')
    plt.ylabel('y test samples')
    plt.show()

    plt.subplot(212)
    plt.title(name)
    plt.scatter(np.arange(len(mse)), mse, s=10)
    plt.xlabel('iteration')
    plt.ylabel('Gradient descent mse')
    plt.show()
```

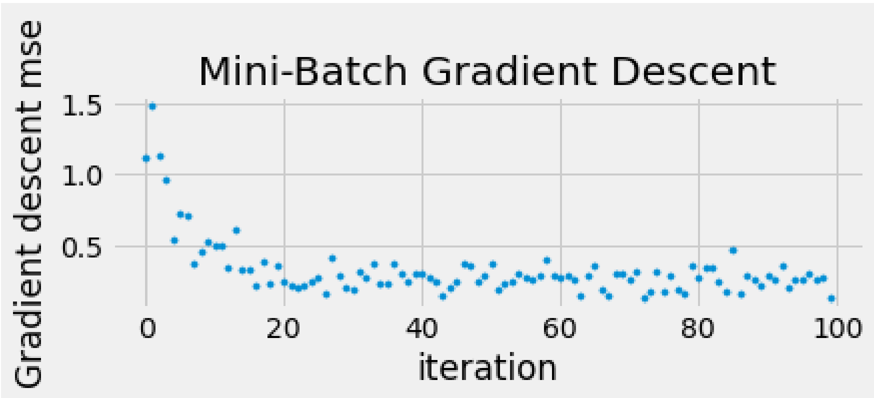
```
Parameter(learning rate:0.01_iteration:100_batch size:1)
weight: 0.8179703765425401
intercept: 0.7845650820481608
```



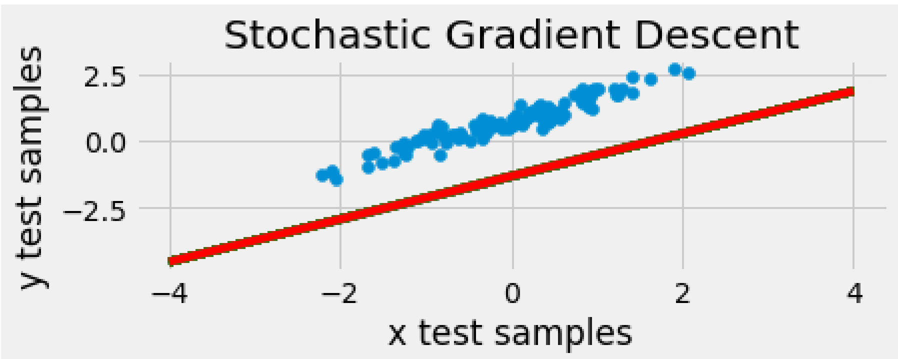


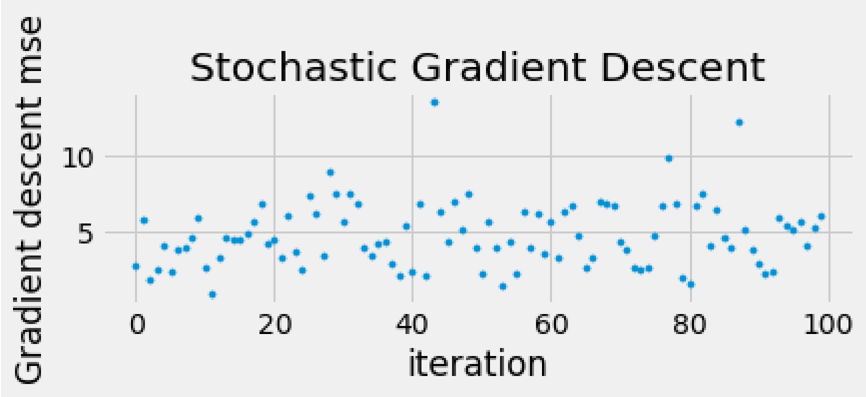

```
Parameter(learning rate:0.1_iteration:100_batch size:None)
weight: 0.8230547915929283
intercept: 0.8281726192998335
```






```
Parameter(learning rate:1.0_iteration:100_batch size:20)
weight: 0.7954157009539082
intercept: -1.332569722600517
```





What’s the difference between Gradient Descent, Mini-Batch Gradient Descent, and Stochastic Gradient Descent?

1. Gradient Descent: The gradient descent is to minimise a given function, which is a first-order iterative optimization algorithom for finding the minimum values. While in Gradient descent, you have to run through all the samples in your training set to do a single update for a parameter in a particular iteration.
2. Mini-Batch Gradient Descent: In large-scale dataset, the training data can have on order of millions of examples. It seems wastely to compute mean square error for entire training dataset. Therefore, The Mini-batch do not use the full training dataset, but we do not use the single data point. We use a randomly selected set of data from our traing dataset. In this way, we can reduce the calculation cost and achieve a lower varaience.
3. Stochastic Gradient Descent: The inclusion of the word stochastic simply means the random samples from the training data are chosen in each run to update parameter during optimisation, within the framework of gradient descent.

In [ ]:

```

```

學號：410551027
姓名：江衍涵

1. Condition

Box R : 3 apples, 4 oranges, 3 guavas , $P(R) = 0.2$

Box B : 2 apples, 0 oranges, 2 guavas , $P(B) = 0.4$

Box G : 12 apples, 4 oranges, 4 guavas , $P(G) = 0.4$

The probability of guava $P(g)$:

$$P(g) = P(g|R)\,P(R) + P(g|B)\cdot P(B) + P(g|G)\,P(G)$$

$$\Rightarrow \frac{3}{10}\cdot 0.2 + \frac{1}{2}\cdot 0.4 + \frac{1}{5}\cdot 0.4$$

$$\Rightarrow 0.34$$

Based on Bayes' theorem, the probability of an selected guava coming from blue box $P(g|B)$

$$P(B|g) = \frac{P(g|B)\cdot P(B)}{P(g)} = \frac{\frac{1}{2}\cdot 0.4}{0.36} = 0.55$$

Ans: 0.55 #

2. The first question is rather simple:

$$(ab)^{\frac{1}{2}} - a = a^{\frac{1}{2}}(b^{\frac{1}{2}} - a^{\frac{1}{2}}) \geq 0$$

Where we have taken advantage of $b \geq a \geq 0$,

base on

$$p(\text{mistake}) = p(x \in R_1, C_2) + p(x \in R_2, C_1)$$

$$= \int_{R_1} p(x, C_2)\, dx + \int_{R_2} p(x, C_1)\, dx$$

if $p(x, C_1) > p(x, C_2)$, for a given value of x, we will assign that x to class C_1.

it should statisfied $p(x, C_1) > p(x, C_2)$

Therefore, using what we have proved, we can obtain:

$$\int_{R} p(x, C_2)\, dx \leq \int_{R_1} \{p(x, C_1)\, p(x, C_2)\}^{\frac{1}{2}}\, dx$$

It is the same for decision area R_2.

Therefore we can obtain:

$$p(\text{mistake}) \leq \int \{p(x, C_1)\, p(x, C_2)\}^{\frac{1}{2}}\, dx$$

3.

We solved it base on definition

$$E_y[E_x[x|y]] = \int E_x[x|y]\, p(y)\, dx$$

$$= \int \left(\int x p(x|y)\, dx \right) p(y)\, dy$$

$$= \iint x\, p(x|y)\, p(y)\, dx\, dy$$

$$= \iint x \cdot p(x,y)\, dx \cdot dy$$

$$= \int x\, p(x)\, dx = E[x]$$

$$E_y[var_x[x|y]] = \int var_x[x|y]\, p(y)\, dy$$

$$= \int \left(\int x - E_x[x|y])^2\, p(x|y)\, dx \right) p(y)\, dy$$

$$= \int \int (x - E_x[x|y])^2\, p(x,y)\, dx \cdot dy$$

$$= \int \int (x^2 - 2x E_x[x|y] + E_x[x|y]^2)\, p(x,y) \cdot dx \cdot dy$$

$$\Rightarrow \iint x^2 p(x)\, dx - \underline{\iint 2x E_x[x|y]\, p(x,y)\, dx\, dy} + \iint (E_x[x|y]^2)\, p(y)\, dy$$

Simplify the second term:

$$\underline{\iint 2x E_x[x|y]\, p(x,y)\, dx\, dy} = 2 \int E_x[x|y] \left(\int x p(x,y)\, dx \right) dy$$

$$= 2 \int E_x[x|y]\, p(y) \left(\int x p(x|y)\, dx \right) dy$$

$$= 2 \int E_x[x|y]^2 p(y)\, dy$$

Therefore, we obtain the first term on the right side

$$E_y[Var_x[x|y]] = \int\int x^2 p(x)\,dx - \int\int E_x[x|y]^2 p(y)\,dy \quad \text{— (1)}$$

Then we process for the second term

$$var_y[E_x[x|y]] = \int \left(E_x[x|y] - E_y[E_x[x|y]]\right)^2 p(y)\,dx$$

$$= \int (E_x[x|y] - E[x])^2 p(y)\,dy$$

$$= \int E_x[x|y]^2 p(y)\,dy - 2\int E[x]\, E_x[x|y]\, p(y)\,dy + \int E[x]^2 p(y)\,dy$$

$$= \int E_x[x|y]^2 p(y)\,dy \underline{- 2E[x]\int E_x[x|y]\,p(y)\,dy} + E[x]^2$$

Then following the same procedure

$$\underline{2E[x]\cdot\int E_x[x|y]\,p(y)\,dy} = 2E[x]\cdot E_y[E_x[x|y]] = 2E[x]^2$$

Then, we simple the second term:

$$var_y[E_x[x|y]] = \int E_x[x|y]^2 p(y)\,dy - E[x]^2 \quad \text{— (2)}$$

Finally, we add (1) and (2), we will obtain:

$$E_y[var_x[x|y]] + var_y[E_x[x|y]] = E[x^2] - E[x]^2 = var[x]$$