P1

```
import numpy as np
from sklearn import linear_model
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
from numpy import linalg as LA
import math

# d = np.load("./data/dataset1_dim9_Ntr5.npz")
# d = np.load("./data/dataset2_dim9_Ntr50.npz")
# d = np.load("./data/dataset3_dim9_Ntr500.npz")
# d = np.load("./data/dataset4_dim2_Ntr5.npz")
# d = np.load("./data/dataset5_dim2_Ntr15.npz")
# d = np.load("./data/dataset6_dim2_Ntr50.npz")
# d = np.load("./data/dataset7_dim2_Ntr5.npz")
# d = np.load("./data/dataset8_dim2_Ntr15.npz")
d = np.load("./data/dataset9_dim2_Ntr50.npz")

X_train = d['X_train']
y_train = d['y_train']
X_test = d['X_test']
y_test = d['y_test']

X_train = np.hstack(([[1] for _ in range(len(X_train))], X_train))
X_test = np.hstack(([[1] for _ in range(len(X_test))], X_test))

X_train_len = len(X_train)
cross_validation_size = int(X_train_len / 5)

print("==================")
print("Least Square")
print("==================")

least_square = LinearRegression(fit_intercept=False).fit(X_train, y_train)
print("MSE on train: ", mean_squared_error(y_train,
least_square.predict(X_train)))
print("MSE on test: ", mean_squared_error(y_test,
least_square.predict(X_test)))
w = np.round(least_square.coef_, 3)
print("w: ", w)

l1_norm = 0
l2_norm = 0
spars = 0
for i in range(len(w)):
    l1_norm += abs(w[i])
    l2_norm += w[i] ** 2
    if w[i] == 0:
        spars += 1
```

```
l2_norm = math.sqrt(l2_norm)

print("l1(w): ", l1_norm)
print("l2(w): ", l2_norm)
print("spars: ", spars)


lasso_min_mean_mse = 9999999999
lasso_best_lambda = 100
lasso_final_std_mse = 0

ridge_min_mean_mse = 9999999999
ridge_best_lambda = 100
ridge_final_std_mse = 0

for log_lambda in range(-100, 105, 5):
    log_lambda = log_lambda / 10
    lambda_value = 2 ** log_lambda

    lasso_mse_list = []
    ridge_mse_list = []

    for i in range(5):
        X_validation = X_train[i * cross_validation_size : (i + 1) *
cross_validation_size]
        y_validation = y_train[i * cross_validation_size : (i + 1) *
cross_validation_size]
        X_train_cur = np.copy(X_train)
        X_train_cur = np.delete(X_train_cur, np.s_[i *
cross_validation_size : (i + 1) * cross_validation_size], axis = 0)
        y_train_cur = np.copy(y_train)
        y_train_cur = np.delete(y_train, np.s_[i * cross_validation_size :
(i + 1) * cross_validation_size], axis = 0)

        lasso = linear_model.Lasso(alpha=lambda_value, fit_intercept=False)
        lasso.fit(X_train_cur, y_train_cur)
        lasso_mse_list.append(mean_squared_error(y_validation,
lasso.predict(X_validation)))

        ridge = linear_model.Ridge(alpha=lambda_value, fit_intercept=False)
        ridge.fit(X_train_cur, y_train_cur)
        ridge_mse_list.append(mean_squared_error(y_validation,
ridge.predict(X_validation)))

    lasso_mean_mse = sum(lasso_mse_list) / len(lasso_mse_list)
    lasso_variance_mse = sum([((x - lasso_mean_mse) ** 2) for x in
lasso_mse_list]) / len(lasso_mse_list)
    lasso_std_mse = lasso_variance_mse ** 0.5

    ridge_mean_mse = sum(ridge_mse_list) / len(ridge_mse_list)
    ridge_variance_mse = sum([((x - ridge_mean_mse) ** 2) for x in
ridge_mse_list]) / len(ridge_mse_list)
    ridge_std_mse = ridge_variance_mse ** 0.5
```

```
        if lasso_mean_mse < lasso_min_mean_mse:
            lasso_min_mean_mse = lasso_mean_mse
            lasso_best_lambda = lambda_value
            lasso_final_std_mse = lasso_std_mse

        if ridge_mean_mse < ridge_min_mean_mse:
            ridge_min_mean_mse = ridge_mean_mse
            ridge_best_lambda = lambda_value
            ridge_final_std_mse = ridge_std_mse

    print("==================")
    print("Lasso")
    print("==================")

    lasso_final = linear_model.Lasso(alpha=lasso_best_lambda,
    fit_intercept=False)
    lasso_final.fit(X_train, y_train)
    print("mean of MSE: ", lasso_min_mean_mse)
    print("std of MSE: ", lasso_final_std_mse)
    print("MSE on train: ", mean_squared_error(y_train,
    lasso_final.predict(X_train)))
    print("MSE on test: ", mean_squared_error(y_test,
    lasso_final.predict(X_test)))
    print("Lambda: ", math.log2(lasso_best_lambda))
    w = np.round(lasso_final.coef_, 3)
    print("w: ", w)

    l1_norm = 0
    l2_norm = 0
    spars = 0
    for i in range(len(w)):
        l1_norm += abs(w[i])
        l2_norm += w[i] ** 2
        if w[i] == 0:
            spars += 1
    l2_norm = math.sqrt(l2_norm)

    print("l1(w): ", l1_norm)
    print("l2(w): ", l2_norm)
    print("spars: ", spars)

    print("==================")
    print("Ridge")
    print("==================")

    ridge_final = linear_model.Ridge(alpha=ridge_best_lambda,
    fit_intercept=False)
    ridge_final.fit(X_train, y_train)
    print("mean of MSE: ", ridge_min_mean_mse)
    print("std of MSE: ", ridge_final_std_mse)
    print("MSE on train: ", mean_squared_error(y_train,
    ridge_final.predict(X_train)))
    print("MSE on test: ", mean_squared_error(y_test,
    ridge_final.predict(X_test)))
```

```
print("Lambda: ", math.log2(ridge_best_lambda))
w = np.round(ridge_final.coef_, 3)
print("w: ", w)

l1_norm = 0
l2_norm = 0
spars = 0
for i in range(len(w)):
    l1_norm += abs(w[i])
    l2_norm += w[i] ** 2
    if w[i] == 0:
        spars += 1
l2_norm = math.sqrt(l2_norm)

print("l1(w): ", l1_norm)
print("l2(w): ", l2_norm)
print("spars: ", spars)
```

P4

```
import numpy as np
import math
import matplotlib.pyplot as plt

def plot1(x):
    y_value = []
    for s in x:
        y_value.append(math.log(1 + math.exp(-s)))
    y = np.array(y_value)
    plt.plot(x, y)
    plt.show()

def plot2(x):
    y_value = []
    for s in x:
        if s <= 0:
            y_value.append(-s)
        else:
             y_value.append(0)
    y = np.array(y_value)
    plt.plot(x, y)
    plt.show()

def plot3(x):
    y_value = []
    for s in x:
        y_value.append((1/2) * -2 * s)
    y = np.array(y_value)
    plt.plot(x, y)
```

```
    plt.show()


x = np.linspace(-10, 10, 100)
plot1(x)
x = np.linspace(-2, 2, 100)
plot1(x)

x = np.linspace(-10, 10, 100)
plot2(x)
x = np.linspace(-2, 2, 100)
plot2(x)

x = np.linspace(-10, 10, 100)
plot3(x)
x = np.linspace(-2, 2, 100)
plot3(x)
```