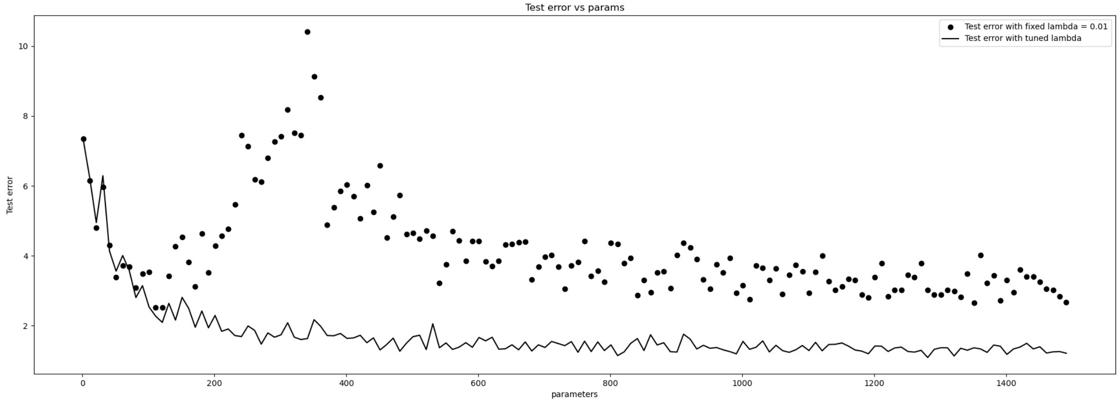
```
import matplotlib.pyplot as plt
import numpy as np


"""
Step 1 : Generate Toy data
"""

d = 35
n_train, n_val, n_test = 300, 60, 30
np.random.seed(0)
beta = np.random.randn(d)
beta_true = beta / np.linalg.norm(beta)
# Generate and fix training data
X_train = np.array([np.random.multivariate_normal(np.zeros(d), np.identity(d)) for _ in range(n_train)])
Y_train = X_train @ beta_true + np.random.normal(loc = 0.0, scale = 0.5, size = n_train)
# Generate and fix validation data (for tuning lambda).
X_val = np.array([np.random.multivariate_normal(np.zeros(d), np.identity(d)) for _ in range(n_val)])
Y_val = X_val @ beta_true
# Generate and fix test data
X_test = np.array([np.random.multivariate_normal(np.zeros(d), np.identity(d)) for _ in range(n_test)])
Y_test = X_test @ beta_true


"""
Step 2 : Solve the problem
"""



lambda_list = [2 ** i for i in range(-6, 6)]
num_params = np.arange(1,1501,10)

errors_opt_lambda = []
errors_fixed_lambda = []

def least_square(X, Y, lamda):
    return  np.linalg.inv(np.dot(X.T, X) + lamda * np.identity(X.shape[1])) @ X.T @ Y

def ReLU(x):
    return np.maximum(0, x)

def error(X, Y, theta) :
    return np.linalg.norm(X @ theta - Y)

def optimize_lambda(X_tilde, Y, X_tilde_val, Y_val) :
    lambda_list = [2 ** i for i in range(-6, 6)]
    val_errors = []
    for lamda in lambda_list :
        theta = least_square(X_tilde, Y, lamda)
        val_errors.append(error(X_tilde_val, Y_val, theta))
    return lambda_list[np.argmin(val_errors)]

for p in num_params :
```

```python
    weight = np.random.normal(loc = 0.0, scale = 1/p**0.5, size = (p, d))

    X_tilde = ReLU(X_train @ weight.T)
    X_tilde_val = ReLU(X_val @ weight.T)

    opt_lambda = optimize_lambda(X_tilde, Y_train, X_tilde_val, Y_val)
    theta_opt = least_square(X_tilde, Y_train, opt_lambda)

    theta_fixed = least_square(X_tilde, Y_train, 0.01)

    X_tilde_test = ReLU(X_test @ weight.T)

    errors_opt_lambda.append(error(X_tilde_test, Y_test, theta_opt))
    errors_fixed_lambda.append(error(X_tilde_test, Y_test, theta_fixed))




"""
Step 3 : Plot the results
"""

plt.figure(figsize = (24, 8))


plt.scatter(num_params, errors_fixed_lambda, color = 'black',
            label = "Test error with fixed lambda = 0.01",
            )
plt.legend()

plt.plot(num_params, errors_opt_lambda, 'k', label = "Test error with tuned lambda")
plt.legend()
plt.xlabel('parameters')
plt.ylabel('Test error')
plt.title('Test error vs params')

plt.savefig('double_descent.png')
plt.show()
```



Test error vs params
Test error with fixed lambda = 0.01
Test error with tuned lambda
Test error
parameters
10
8
6
4
2
0
200
400
600
800
1000
1200
1400