**Лабораторная работа №3: «Переобучение и регуляризация»**

Набор данных **ex3data1.mat** представляет собой файл формата *.mat (т.е. сохраненного из Matlab). Набор содержит две переменные X (изменения уровня воды) и y (объем воды, вытекающий из дамбы). По переменной X необходимо предсказать y. Данные разделены на три выборки: обучающая выборка (X, y), по которой определяются параметры модели; валидационная выборка (Xval, yval), на которой настраивается коэффициент регуляризации; контрольная выборка (Xtest, ytest), на которой оценивается качество построенной модели.

**Задание.**

1. Загрузите данные **ex3data1.mat** из файла.
2. Постройте график, где по осям откладываются X и y из обучающей выборки.
3. Реализуйте функцию стоимости потерь для линейной регрессии с L2-регуляризацией.
4. Реализуйте функцию градиентного спуска для линейной регрессии с L2-регуляризацией.
5. Постройте модель линейной регрессии с коэффициентом регуляризации 0 и постройте график полученной функции совместно с графиком из пункта 2. Почему регуляризация в данном случае не сработает?
6. Постройте график процесса обучения (learning curves) для обучающей и валидационной выборки. По оси абсцисс откладывается число элементов из обучающей выборки, а по оси ординат - ошибка (значение функции потерь) для обучающей выборки (первая кривая) и валидационной выборки (вторая кривая). Какой вывод можно сделать по построенному графику?
7. Реализуйте функцию добавления p - 1 новых признаков в обучающую выборку ($X^2$, $X^3$, $X^4$, …, $X^p$).
8. Поскольку в данной задаче будет использован полином высокой степени, то необходимо перед обучением произвести нормализацию признаков.
9. Обучите модель с коэффициентом регуляризации 0 и p = 8.
10. Постройте график модели, совмещенный с обучающей выборкой, а также график процесса обучения. Какой вывод можно сделать в данном случае?
11. Постройте графики из пункта 10 для моделей с коэффициентами регуляризации 1 и 100. Какие выводы можно сделать?

12. С помощью валидационной выборки подберите коэффиент регуляризации, который позволяет достичь наименьшей ошибки. Процесс подбора отразите с помощью графика (графиков).
13. Вычислите ошибку (потерю) на контрольной выборке.
14. Ответы на вопросы представьте в виде отчета.

**Реализация:**

In[1]:

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

**1.1 Загрузите набор данных ex3data1.mat из файла.**

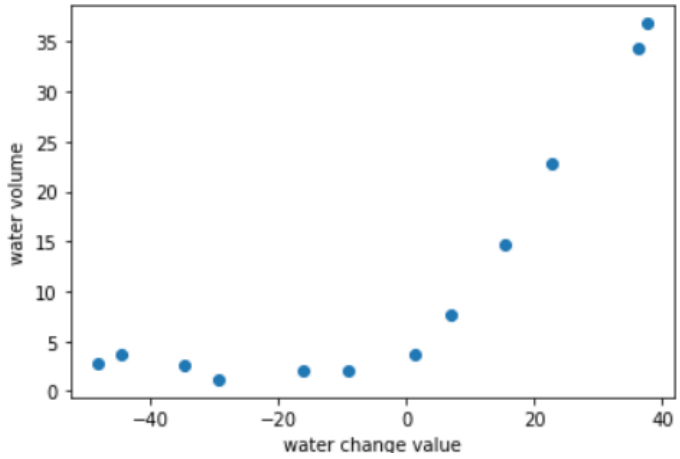
In[2]:

```
from scipy.io import loadmat

mat = loadmat('data/ex3data1.mat')
mat_y = mat['y']
x_train, y_train = mat['X'], mat_y.reshape(len(mat_y))

yval = mat['yval']
x_val, y_val = mat['Xval'], yval.reshape(len(yval))

ytest = mat['ytest']
x_test, y_test = mat['Xtest'], ytest.reshape(len(ytest))
```

**1.2 Постройте график, где по осям откладываются X и y из обучающей выборки.**

In[3]:

```
plt.plot(x_train, y_train, 'o')
plt.xlabel('water change value')
plt.ylabel('water volume')
plt.show()
```




**1.3 Реализуйте функцию стоимости потерь для линейной регрессии с L2-регуляризацией.**

In[4]:

```
def h(x, theta):
    if len(x.shape) > 1 and x.shape[1] < theta.shape[0]:
        x = np.column_stack((np.ones(x.shape[0]), x))

    return x.dot(theta)

def cost_func(x, y, theta, l2_penalty_value=0.1):
    err = (h(x, theta) - y) ** 2

    theta_ = theta[1:]
    total_cost = err.sum() + l2_penalty_value * np.dot(theta_.T, theta_)
    return total_cost / 2 / x.shape[0]
```

**1.4 Реализуйте функцию градиентного спуска для линейной регрессии с L2-регуляризацией.**

In[5]:

```
THRESHOLD = 1e-7

def gradient_descent(x, y, max_iters_count=300000, a=0.001, l2_pen-
alty_value=0.1, cost_func=cost_func):
    theta = np.zeros(x.shape[1])
    last_loss = cost_func(x, y, theta)
    logs = []

    for i in range(max_iters_count):
        diff = h(x, theta) - y
        gradient_first = np.dot(x.T[:1], diff)
        gradient_full = np.dot(x.T[1:], diff) + l2_penalty_value * theta[1:]
        gradient = np.insert(gradient_full, 0, gradient_first)
        gradient /= x.shape[0]
        gradient *= a
        theta -= gradient

        curr_los = cost_func(x, y, theta)
        logs.append(curr_los)
        if abs(curr_los - last_loss) < THRESHOLD:
            break

        last_loss = curr_los

    return theta, logs
```

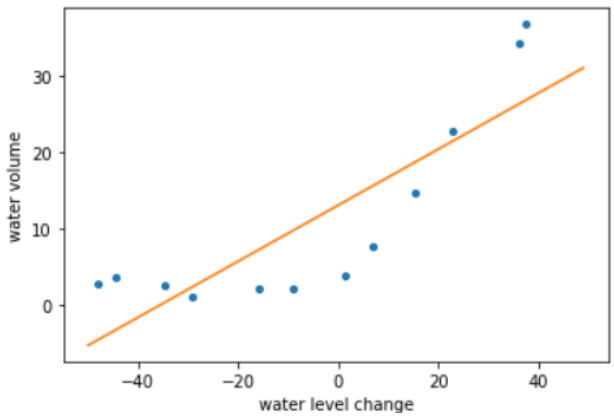
Out[5]:

```
Optimization terminated successfully.
         Current function value: 0.203498
         Iterations: 25
         Function evaluations: 33
         Gradient evaluations: 33
Optimization terminated successfully.
         Current function value: 0.203498
         Iterations: 198
         Function evaluations: 379


(array([-25.16133374,   0.20623172,   0.2014716 ]), [])
```

**1.5 Постройте модель линейной регрессии с коэффициентом регуляризации 0 и постройте график полученной функции совместно с графиком из пункта 2. Почему регуляризация в данном случае не сработает?**

In[6]:

```
def normalize_features(x):
    N = x.shape[1]
    copy_x = x.copy()
    for i in range(N):
        feature = x[:, i]
        mean = np.mean(feature)
        delta = np.max(feature) - np.min(feature)
        copy_x[:, i] -= mean
        copy_x[:, i] /= delta
    return copy_x

def fit(x, y, normalize=False, **kwargs):
    x = x.astype('float64')
    y = y.astype('float64')

    if normalize:
        x = normalize_features(X)

    x = np.column_stack((np.ones(x.shape[0]), x))

    return gradient_descent(x, y, **kwargs)

def predict(x, theta):
    x_extended = np.insert(x, 0, 1)
    return h(x_extended, theta)

theta, logs = fit(x_train, y_train, a=0.001, max_iters_count=1000000, l2_pen-
alty_value=0)

xi = list(range(-50, 50))
line = [predict(np.array(i), theta) for i in xi]
plt.plot(x_train, y_train, 'o', xi, line, markersize=4)
plt.xlabel('water level change')
plt.ylabel('water volume')
plt.show()
```



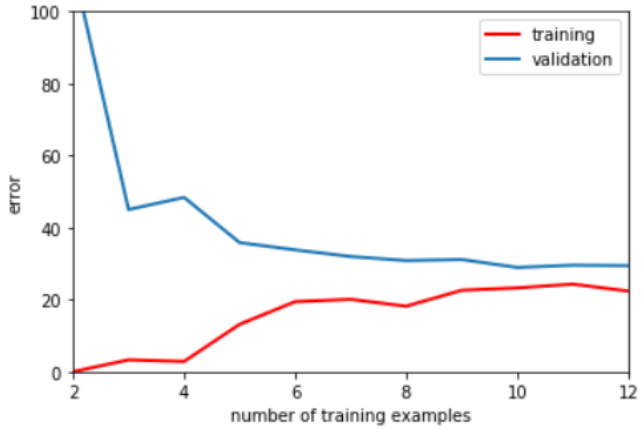

Когда коэффициент регуляризации равен 0, то мы получаем обычную линейную регрессию. Из графика выше видно, данная модель плохо

описывает зависимость между данными, таким образом, нужны дополнительные переменные и фукция зависимости должна быть нелинейной.

**1.6. Постройте график процесса обучения (learning curves) для обучающей и валидационной выборки. По оси абсцисс откладывается число элементов из обучающей выборки, а по оси ординат - ошибка (значение функции потерь) для обучающей выборки (первая кривая) и валидационной выборки (вторая кривая). Какой вывод можно сделать по построенному графику?**

In[7]:

```
def learn-
ing_curves(cost_func, x_train, y_train, x_val, y_val, max_axis=100, l2_pen-
alty_value=0, **kwargs):
    N = len(y_train)
    train_err = np.zeros(N)
    val_err = np.zeros(N)

    for i in range(1, N):
        theta, logs = fit(x_train[0:i + 1, :], y_train[0:i + 1], l2_pen-
alty_value=l2_penalty_value, **kwargs)
        train_err[i] = cost_func(x_train[0:i + 1, :], y_train[0:i + 1], theta, l2
_penalty_value=l2_penalty_value)
        val_err[i] = cost_func(x_val, y_val, theta, l2_penalty_value=l2_pen-
alty_value)

    plt.plot(range(2, N + 1), train_err[1:], c="r", linewidth=2)
    plt.plot(range(2, N + 1), val_err[1:], linewidth=2)
    plt.xlabel("number of training examples")
    plt.ylabel("error")
    plt.legend(["training", "validation"], loc="best")
    plt.axis([2, N, 0, max_axis])
    plt.show()

learning_curves(cost_func, x_train, y_train, X_val, y_val)
```




Исходя из графика можно сделать вывод, что модель недообучена. Так как даже на обучающей выборке у нас большая ошибка.

**1.7 Реализуйте функцию добавления p - 1 новых признаков в обучающую выборку (X2, X3, X4, …, Xp).**

In[8]:

```
def make_polynom_of_features(x, degree):
    x = x.reshape(x.shape[0])
    x_res = np.array(x)

    for i in range(2, degree + 1):
        x_res = np.column_stack((x_res, x ** i))

    return x_res
```

**1.8 Поскольку в данной задаче будет использован полином высокой степени, то необходимо перед обучением произвести нормализацию признаков**

Функция normalize_features выше.

**1.9 Обучите модель с коэффициентом регуляризации 0 и p = 8.**

In[9]:

```
x_train_poly = normalize_features(make_polynom_of_features(x_train, 8))
theta, _ = fit(x_train_poly, y_train, a=0.3, max_iters_count=500000, l2_penalty_v
alue=0)
```