Учреждение образования

«Белорусский государственный университет

информатики и радиоэлектроники»

Кафедра информатики

ЛабоРАТОРНАЯ РАБОТА №3

«Переобучение и регуляризация»

Выполнил: Шпаковская Валерия

магистрант кафедры информатики

группа №858641

Проверил: доцент, кандидат технических наук Стержанов Максим Валерьевич

Минск 2019

ХОД РАБОТЫ

**Данные.**

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.

**Выполнение:**

1. Загрузите данные ex3data1.mat из файла:

file\_path = 'ex3data1.mat'

dataset = sio.loadmat(file\_path)

x\_train = dataset["X"]

x\_val = dataset["Xval"]

x\_test = dataset["Xtest"]

# squeeze the target variables into one-dimensional arrays

y\_train = dataset["y"].squeeze()

y\_val = dataset["yval"].squeeze()

y\_test = dataset["ytest"].squeeze()

2. Постройте график, где по осям откладываются X и y из обучающей выборки:

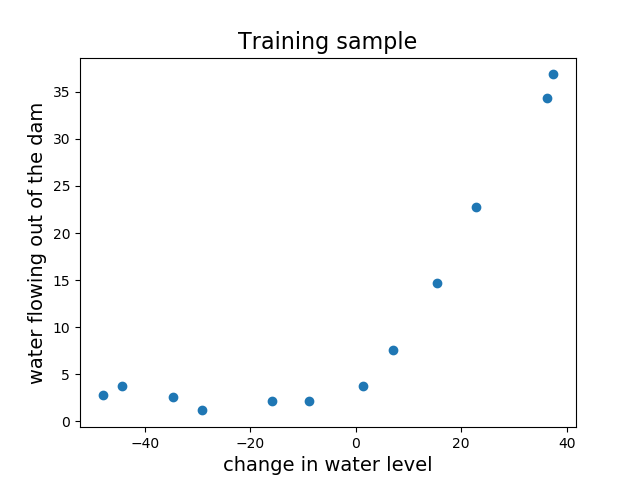
****

Рисунок 1 – график зависимости объёма воды, вытекающего из дамбы от изменения уровня воды

3. Реализуйте функцию стоимости потерь для линейной регрессии с L2-регуляризацией:

def cost\_l2(theta, X, y, lamb=0):

predictions = h0x(X, theta)

squared\_errors = np.sum(np.square(predictions - y))

regularization = np.sum(lamb \* np.square(theta[1:]))

return (squared\_errors + regularization) / (2 \* len(y))

4. Реализуйте функцию градиентного спуска для линейной регрессии с L2-регуляризацией:

def gradient\_l2(theta, X, y, lamb):

predictions = h0x(X, theta)

gradient = np.dot(X.transpose(), (predictions - y))

regularization = lamb \* theta

regularization[0] = 0 # because formula for 0 member is different

return (gradient + regularization) / len(y)

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?

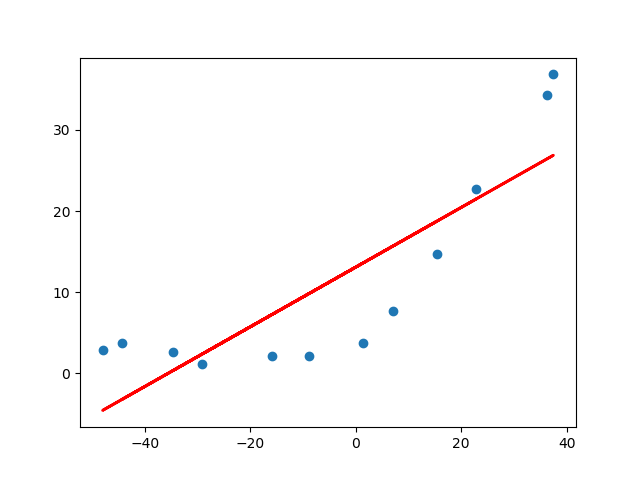


Рисунок 2 – график полученной функции

Регуляризация не помогает при high bias.

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:

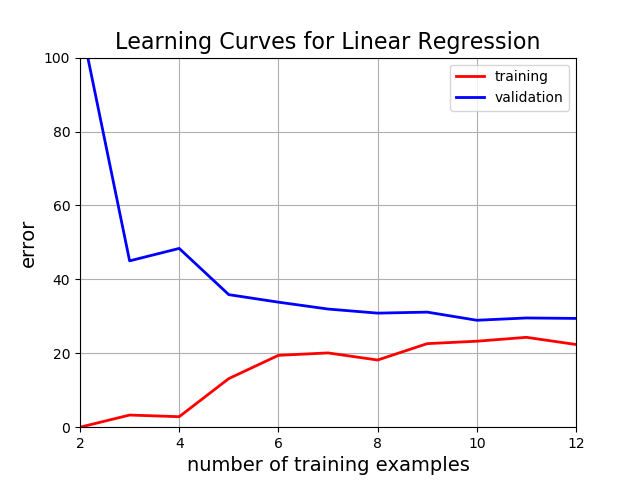


Рисунок 3 – график зависимости ошибки на валидационном и тренировочном сете в зависимости от количества примеров

На графике очевиден high bias problem (underfitting). Выбранная hypothesis (линейная функция) не может подойти данной выборке. Один из вариантов улучшения predict – усложнить функцию, добавив полиномы высшей степени.

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

def polynom(x, degree):

X\_poly = np.zeros(shape=(len(x), degree))

for i in range(0, degree):

X\_poly[:, i] = x.squeeze() \*\* (i + 1);

return X\_poly

x\_train\_poly = polynom(x\_train, 8)

x\_val\_poly = polynom(x\_val, 8)

x\_test\_poly = polynom(x\_test, 8)

8. Поскольку в данной задаче будет использован полином высокой степени, то необходимо перед обучением произвести нормализацию признаков:

train\_means = x\_train\_poly.mean(axis=0)

train\_std = np.std(x\_train\_poly, axis=0, ddof=1)

x\_train\_poly = (x\_train\_poly - train\_means) / train\_std

x\_val\_poly = (x\_val\_poly - train\_means) / train\_std

x\_test\_poly = (x\_test\_poly - train\_means) / train\_std

X\_train\_poly = np.hstack((np.ones((len(x\_train\_poly), 1)), x\_train\_poly))

X\_val\_poly = np.hstack((np.ones((len(x\_val\_poly), 1)), x\_val\_poly))

X\_test\_poly = np.hstack((np.ones((len(x\_test\_poly), 1)), x\_test\_poly))

9. Обучите модель с коэффициентом регуляризации 0 и p = 8:

theta = opt.fmin\_cg(cost\_l2, np.zeros(X\_train\_poly.shape[1]), gradient\_l2, (X\_train\_poly, y\_train, 0), disp=False)

10. Постройте график модели, совмещенный с обучающей выборкой, а также график процесса обучения. Какой вывод можно сделать в данном случае?

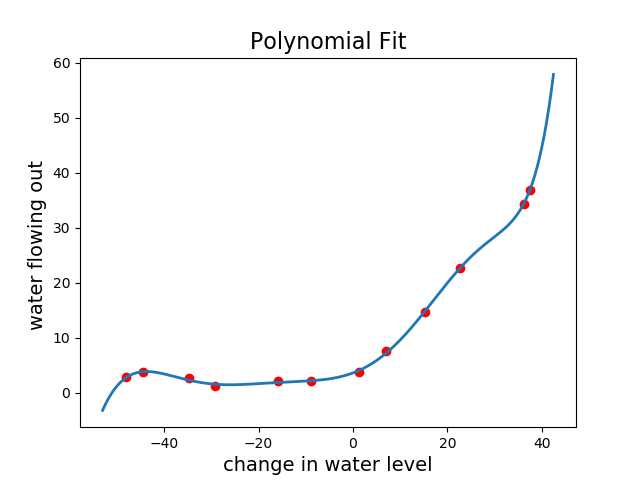


Рисунок 4 – график полученной функции совмещенный с исходными данными (lambda=0)



Рисунок 5 – график зависимости ошибки на валидационном и тренировочном сете в зависимости от количества примеров (lambda=0)

Очевиден hogh variance problem (overfitting). В общем случае, эта проблема решается увеличением коэффициента регуляризации и добавление дополнительных данных в тренировочный список. Альтернативный вариант – уменьшить полиномиальную степень hypothesis, что уменьшит вычисления.

11. Постройте графики из пункта 10 для моделей с коэффициентами регуляризации 1 и 100. Какие выводы можно сделать?

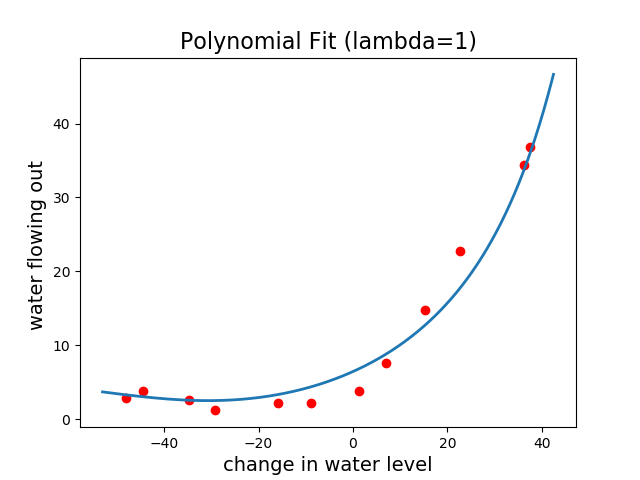


Рисунок 6 – график полученной функции совмещенный с исходными данными (lambda=1)

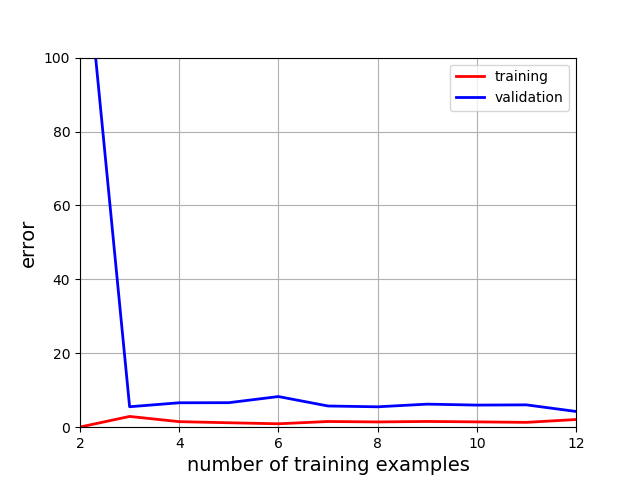


Рисунок 7 – график зависимости ошибки на валидационном и тренировочном сете в зависимости от количества примеров (lambda=1)

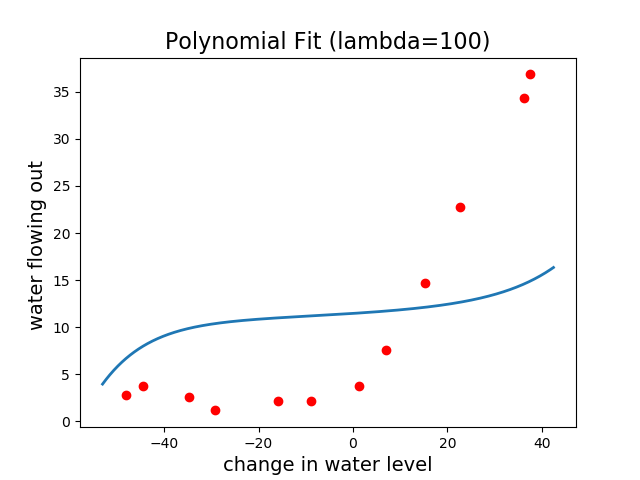


Рисунок 8 – график полученной функции совмещенный с исходными данными (lambda=100)

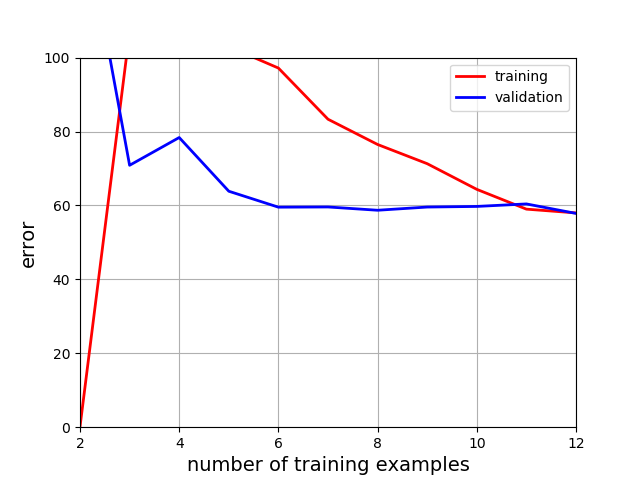


Рисунок 9 – график зависимости ошибки на валидационном и тренировочном сете в зависимости от количества примеров (lambda=100)

Коэффициент регуляризации = 1 – Just Right.

Коэффициент регуляризации = 100 – High Bias.

12. С помощью валидационной выборки подберите коэффиент регуляризации, который позволяет достичь наименьшей ошибки. Процесс подбора отразите с помощью графика (графиков):

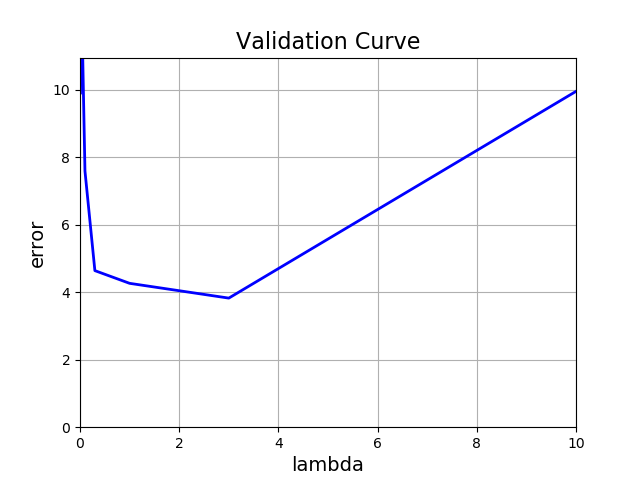


Рисунок 10 – график зависимости ошибки на валидационном сете от параметра lambda

13. Вычислите ошибку (потерю) на контрольной выборке:

theta = opt.fmin\_cg(cost\_l2, np.zeros(X\_train\_poly.shape[1]), gradient\_l2, (X\_train\_poly, y\_train, 3), disp=False)

test\_error = cost\_l2(theta, X\_test\_poly, y\_test)

print("Test Error: ", test\_error, "| Regularized Polynomial (lambda=3))")

Реультат выполнения:

'Test Error: ', 3.8598878210299112, '| Regularized Polynomial (lambda=3))'