Практическое задание № 5

РЕГУЛЯРИЗОВАННАЯ ЛИНЕЙНАЯ РЕГРЕССИЯ. НЕДООБУЧЕНИЕ И ПЕРЕОБУЧЕНИЕ

Цель работы: получить практику анализа статистических данных с использованием регуляризованной линейной регрессии при изучении проблем недообучения и переобучения моделей.

Содержание задания

1. Общие сведения

1. Ознакомиться с материалами лекций № 4 и № 5.

2. Установить необходимое программное обеспечение. При выполнении задания наверняка понадобятся **Python 3**, **NumPy**, **SciPy**, и **Matplotlib**.

3. Ознакомиться с содержимым папки с заданием, которая включает в себя файлы, представленные ниже.

`main.py` – «основной» модуль, необходимый для выполнения задания, который поможет выполнить его поэтапно. Настоящий программный код не требует какой-либо коррекции!

`data.mat` – база данных для выполнения задания.

`featureNormalize.py` – модуль, содержащий функцию `featureNormalize`, которая необходима для нормализации признаков. Данный модуль не требует коррекции!

`computeCost.py` – модуль, содержащий функцию `computeCost`, которая необходима для вычисления значения стоимостной функции регуляризованной линейной регрессии.

`gradientDescent.py` – модуль, содержащий функцию `gradientDescent`, которая необходима для выполнения градиентного спуска с целью поиска параметров модели регуляризованной линейной регрессии.

`learningCurve.py` – модуль, содержащий функцию `learningCurve`, которая необходима для вычисления ошибки предсказания на обучающем и тестовом множествах данных для обученной модели регрессии и разного объема тестовой выборки. Вычисленные ошибки позволяют выполнить построение кривых обучения.

`polyFeatures.py` – модуль, содержащий функцию `polyFeatures`, которая необходима для вычисления

дополнительных свойств, необходимых для построения модели на основе полиномиальной регрессии.

4. Поэтапно выполнить задание, связанное с реализацией и исследованием нейронной сети прямого распространения.

5. Ответить на вопросы, необходимые для составления отчета по данному практическому заданию. Отчет сдается на проверку в печатной или письменной форме в указанные сроки.

2. Регуляризованная линейная регрессия

При выполнении данного задания требуется заполнить пустые места программного кода в блоках с комментарием «Ваш код здесь». Данную процедуру необходимо выполнить для следующих функций: `computeCost`, `gradientDescent`, `learningCurve`, `polyFeatures`.

1. При решении любой задачи с использованием инструментов машинного обучения важным является понимание структуры анализируемых данных и их визуализация в случае возможности. В настоящем задании предлагается использовать базу данных из файла **`data.mat`**. Данные представляют собой множество объектов, описываемых одним признаком (изменение уровня воды в реке) и меткой (количество воды, сбрасываемое дамбой). Обратите внимание на то, что загружаемые данные разделены на три подмножества (обучающее, проверочное и тестовое). Блок кода для визуализации исследуемых данных находится в модуле **`main.py`**, который, как и в предыдущих практических заданиях, требуется выполнить при завершении отсутствующих блоков программного кода на этапе тестирования. Результат визуализации данных представлен на рис. 1.

2. Завершите программный код в модуле **`computeCost.py`**, который позволит вычислить значение стоимостной функции для регуляризованной линейной регрессии. Формулы, описывающие ее вычисление, представлены в лекции № 4 и № 5. При выполнении данной части задания могут понадобиться функции из библиотеки **NumPy**, представленные ниже.

`dot` – позволяет вычислить матричное произведение для двумерных массивов и скалярное произведение для одномерных массивов (без комплексного сопряжения).

`sum` – позволяет вычислить сумму элементов вдоль определенной размерности двумерного массива и сумму всех элементов для одномерного массива.

Также полезным может оказаться оператор поэлементное возведение компонентов двумерного и одномерного массивов в квадрат: `** 2`.

3. Завершите программный код в модуле **gradientDescent.py**, который позволит выполнить алгоритм градиентного спуска с целью обучения параметров модели регуляризованной линейной регрессии. Формулы, описывающие реализацию градиентного спуска, представлены в лекции № 4. При выполнении данной части задания могут понадобиться следующие функции из библиотеки **NumPy**: `dot`, `transpose` и `concatenate`.

`transpose` – позволяет выполнить транспонирование массива. Для одномерного массива данная функция не оказывает никакого действия, а для двумерного массива использование функции соответствует обычному матричному транспонированию.

`concatenate` – выполняет объединение последовательности массивов вдоль определенной размерности.

После обучения параметров модели регуляризованной линейной регрессии с настройками градиентного спуска заданными по умолчанию в файле **main_one.py**, должен получиться результат, представленный на рис. 2.

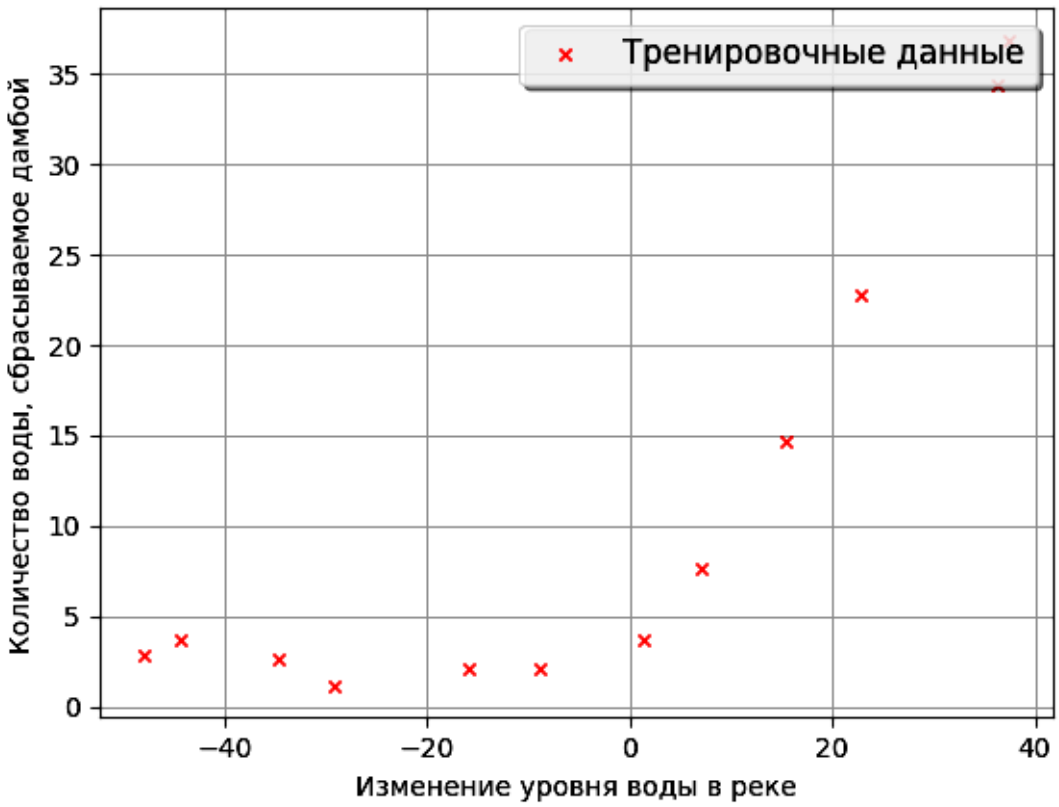


Рис. 1. Результат визуализации тренировочных данных

4. Завершите программный код в модуле **learningCurve.py**, который позволит выполнить вычисление ошибки предсказания

на обучающем и тестовом множествах данных для разного объема тестовой выборки. Формулы, описывающие вычисление этих ошибок, представлены в лекции № 5. Вычисленные ошибки позволяют выполнить построение кривых обучения. При выполнении данного задания фактически требуется выполнить обучение модели регрессии с использованием одного, двух и т.д. тренировочных примеров. А затем вычислить ошибки предсказания для одного, двух и т.п. тренировочных примеров и для всех проверочных примеров в каждом случае. Результат выполнения данной процедуры позволит выполнить построение кривых обучения, пример которых представлен на рис. 3 для случая регуляризованной линейной регрессии с параметром регуляризации равным 0. При выполнении данного пункта будут необходимы функции `computeCost` и `gradientDescent`, реализованные в модулях **`computeCost.py`** и **`gradientDescent.py`**, которые находятся в папке с заданием.

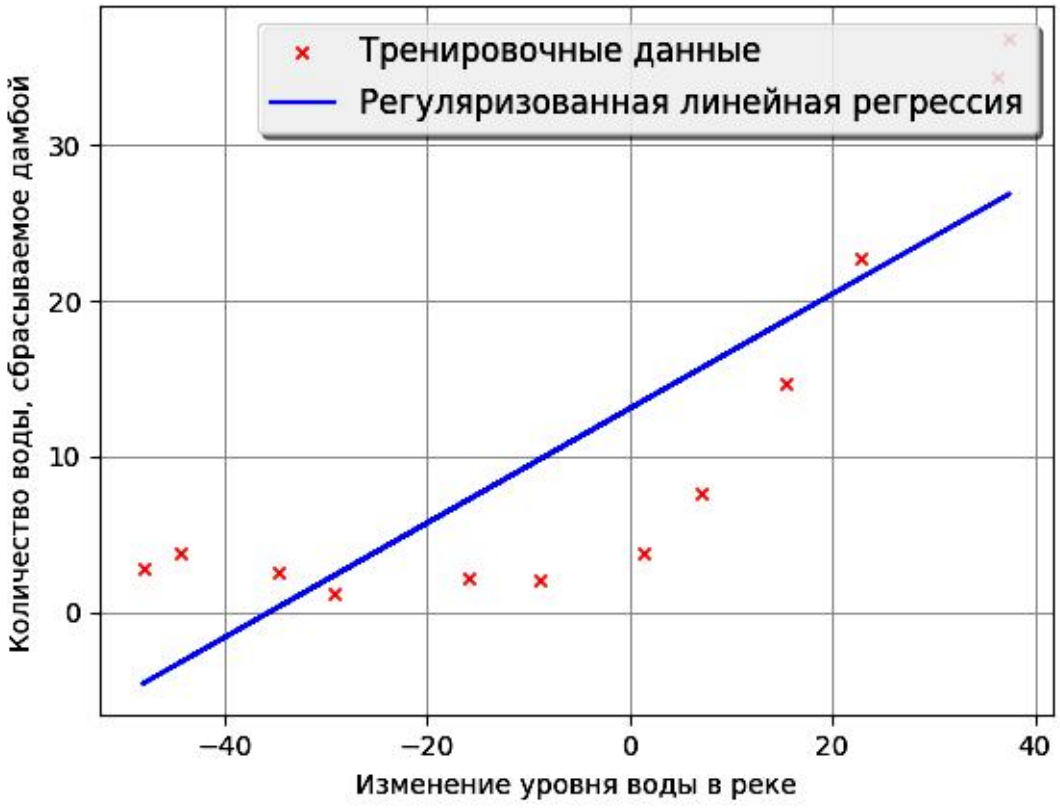


Рис. 2. Результат визуализации тренировочных данных и линии регрессии (гипотезы для регуляризованной линейной регрессии)

5. Завершите программный код в модуле **`polyFeatures.py`**, который позволит выполнить вычисление дополнительных свойств для построения модели на основе полиномиальной регрессия. Функция `polyFeatures` должна выполнять преобразование признака x в вектор-

признаков x, x^2, x^3, …, x^p. Теоретическое описание полиномиальной регрессии представлено в лекции № 2. При выполнении данного пункта полезным может оказаться оператор поэлементное возведение компонентов двумерного и одномерного массивов в степень n: `** n`.

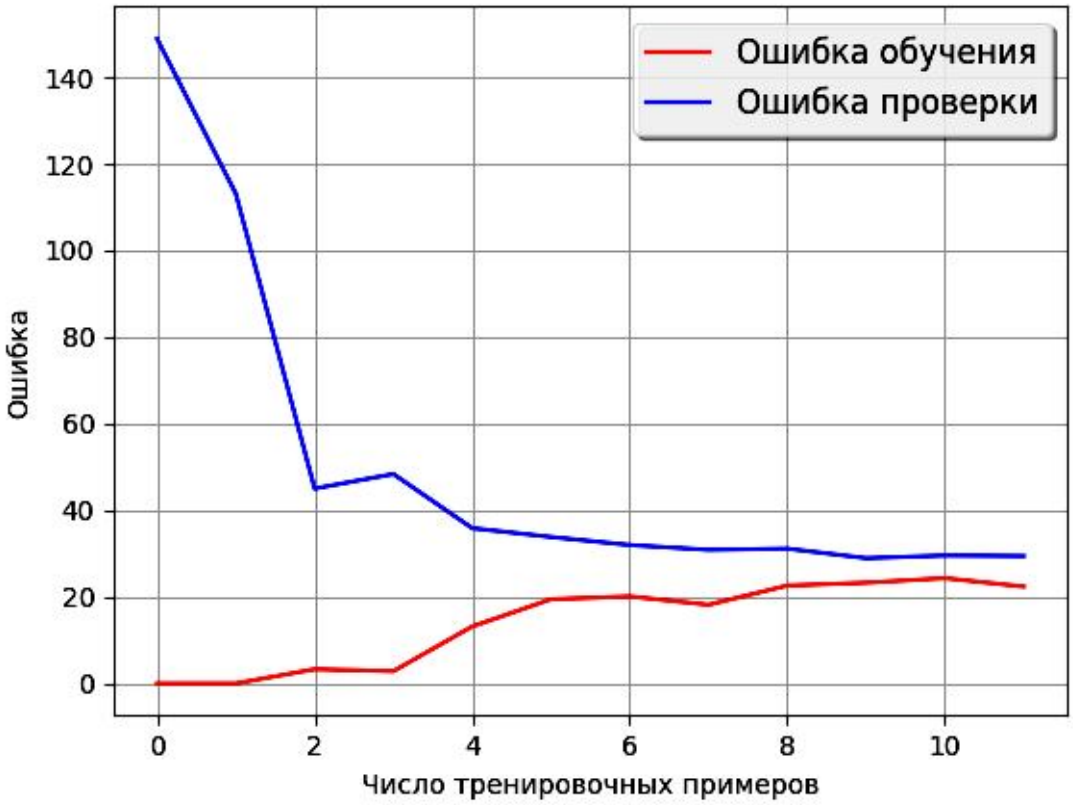
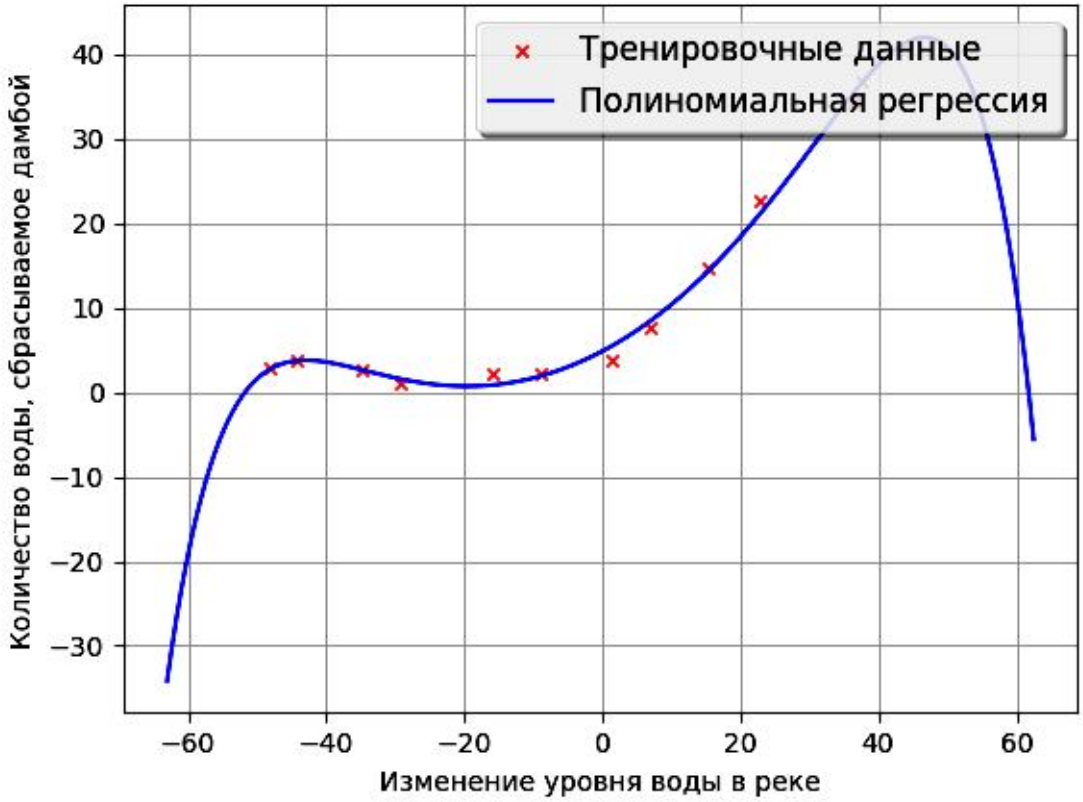


Рис. 3. Кривые обучения для регуляризованной линейной регрессии с параметром регуляризации равным 0

6. После выполнения предыдущего пункта осуществить обучение модели на основе полиномиальной регрессии для настроек градиентного спуска, заданных по умолчанию. В ходе выполнения данного пункта рекомендуется выбрать различные значения параметра регуляризации в интервале от 0 до 30 и пронаблюдать изменение формы гипотезы, а также кривых обучения. Пример полученной гипотезы для полиномиальной регрессии и кривых обучения представлен на рис. 4. Степень полинома выбрана равной 8. Параметр регуляризации выбран равным 0. Анализ рис. 4 и 5 показывает, что обученная гипотеза достаточно хорошо подгоняет тренировочные данные, обеспечивая низкую ошибку обучения. При этом поведение гипотезы вне интервала, на котором представлены данные, является неудовлетворительным, указывая на возможное переобучение модели. Примеры качественно обученной и недообученой моделей приведены на рис. 5.



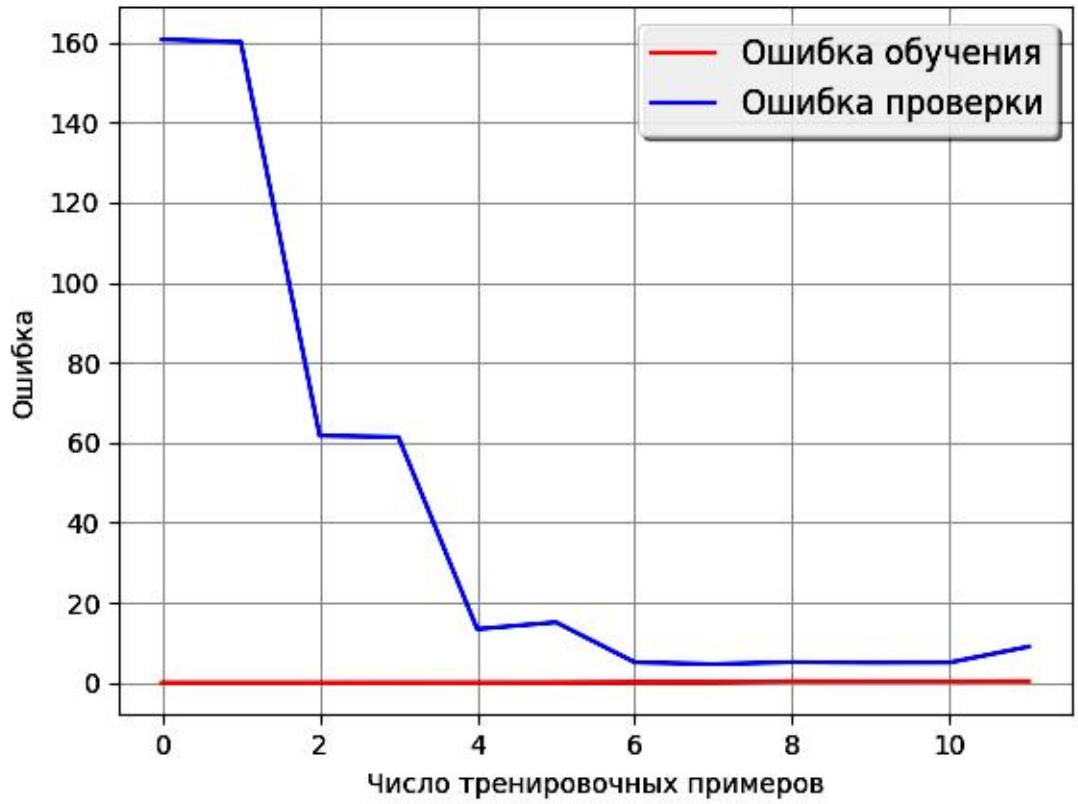


Рис. 4. Обученная модель полиномиальной регрессии для параметра регуляризации равного 0 (сверху) и соответствующие кривые обучения (снизу)

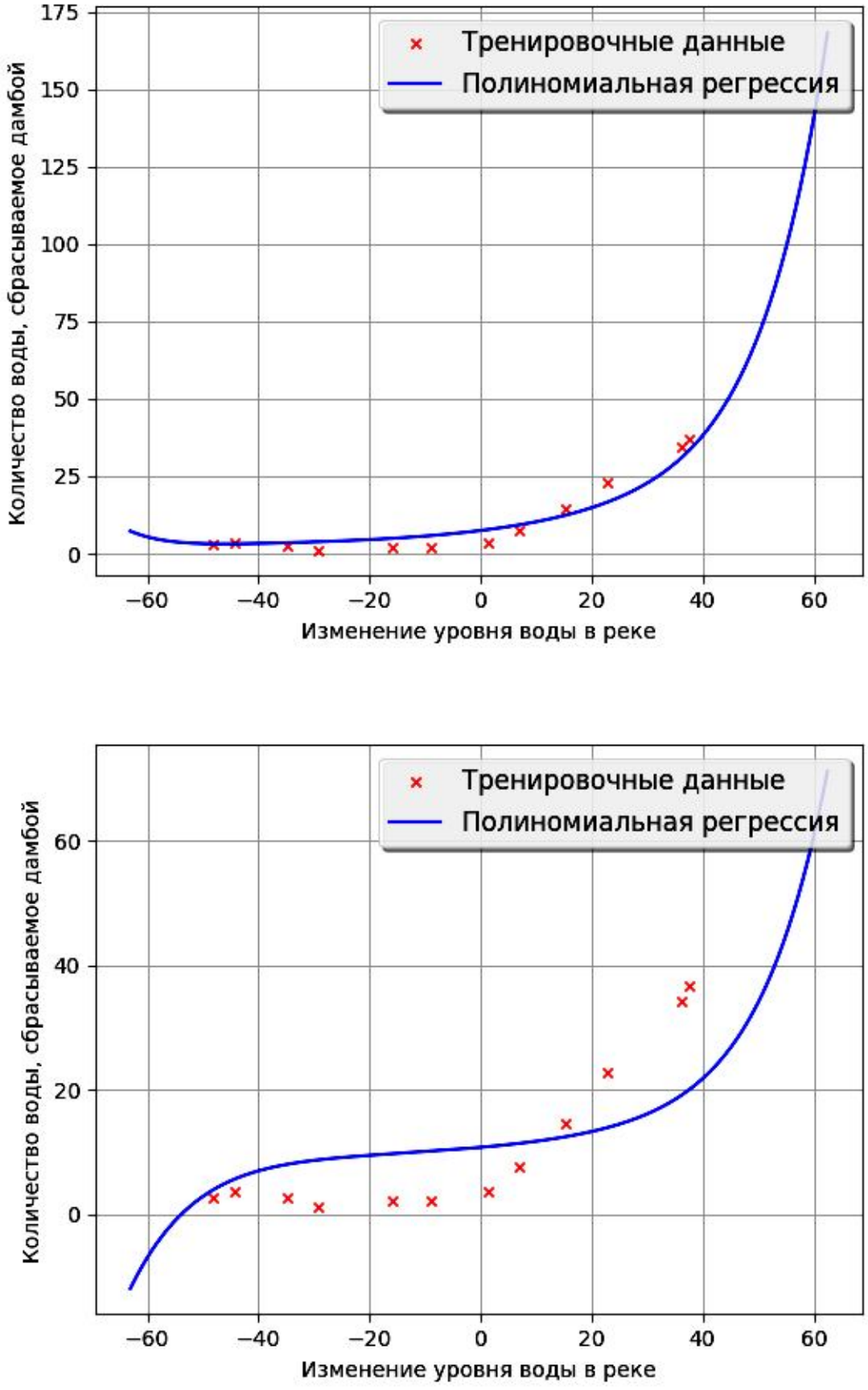


Рис. 5. Пример качественно обученной (сверху) и недообученой (снизу) моделей на основе полиномиальной регрессии

7. Из набора чисел 0, 0.001, 0.003, 0.01, 0.03, 0.1, 0.3, 1, 3, 10 выполнить выбор наилучшего параметра регуляризации, который позволит построить гипотезу на основе полиномиальной регрессии (с максимальной степенью полиномиального признака равной 8) подобную той, что представлена на рис. 5 (сверху). Настройки градиентного спуска оставить заданными по умолчанию.

3. Вопросы для составления отчета

1. Чему равно значение регуляризованной стоимостной функции для случая, когда все параметры модели, а также параметр регуляризации равны единице (**25 баллов**)?
2. Чему равны значения параметров обученной модели регуляризованной линейной регрессии для случая, когда параметр сходимости равен 0.05, число итераций градиентного спуска равно 1500, а параметр регуляризации равен 0 (**25 баллов**)?
3. Чему равно значение наилучшего параметра регуляризации из набора чисел 0, 0.001, 0.003, 0.01, 0.03, 0.1, 0.3, 1, 3, 10 для обученной модели полиномиальной регрессии с максимальной степенью полиномиального признака равной 8, когда параметр сходимости равен 0.05, а число итераций градиентного спуска равно 1500 (**25 баллов**)?
4. Чему равны значения параметров обученной модели полиномиальной регрессии из пункта 3 для наилучшего подобранного параметра регуляризации (**25 баллов**)?