Лабораторная работа №9. Диагностика машинного обучения.

В данной работе вам необходимо выполнить диагностику модели линейной регрессии, построить кривые обучения и оценить наличие проблем дисперсии/смещения, а также подобрать параметры обучения по валидационной выборке и оценить обобщающую способность модели. Теоретический материал для данной лабораторной работы можно найти в разделе 6 учебного пособия.

В качестве тестовой задачи будем рассматривать линейную регрессию предсказания объема воды, перетекающей через дамбу водохранилища, в зависимости от изменения уровня воды ($n = 1$).

1 Загрузка и визуализация данных

На этом шаге загружается файл ex5data1.npy, который содержит наборы данных. Из файла загружаются следующие массивы:

X – обучающая выборка, вектор значений изменения уровня воды, 12 значений ($m = 12$);

y – значения потока воды, вытекающей за секунду, для заданных значений X;

Xval, yval – валидационная выборка, 21 значение ($m_{cv} = 21$);

Xtest, ytest – тестовая выборка, 21 значение ($m_{test} = 21$).

После загрузки данных необходимо отобрать обучающую выборку на плоскости. Найдите в программе функцию display_data. В ней оставлено два места для дополнения программы: для шага 1 и шага 4 лабораторной работы. На данный момент в разделе для шага 1 дополните функцию кодом для отображения обучающей выборки. Пример результата работы функции показан на рис. 1.

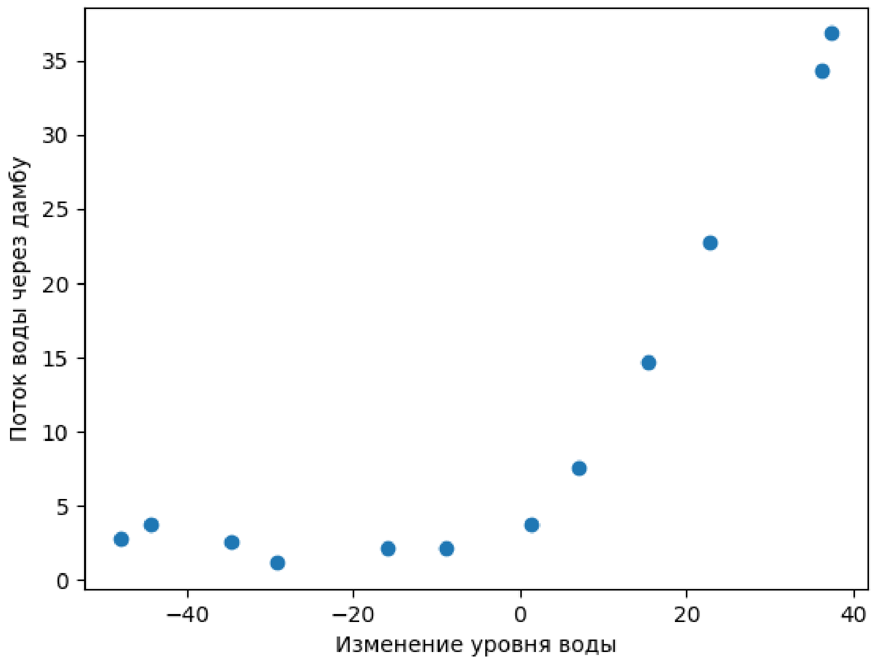

Рис. 1. Отображение обучающей выборки.

Удалите оператор return после первого шага в функции main, чтобы перейти ко второму шагу.

2 Функция стоимости логистической регрессии

На этом шаге вам необходимо запрограммировать вычисление функции стоимости регуляризованной логистической регрессии. Найдите в программе функцию cost_function и дополните ее кодом до рабочего состояния. Функция принимает на вход следующие переменные:

theta – вектор θ параметров регрессионной модели;

X, y – обучающая выборка;

lamb – параметр регуляризации.

Функция должна возвращать значение функции стоимости $J(\theta)$.

После того, как функция готова, запустите программу. Программа вычисляет функцию стоимости для предварительно заданных параметров. Полученное значение должно быть близко к ожидаемому.

Удалите оператор return, чтобы перейти к следующему шагу.

3 Функция градиента линейной регрессии

На этом шаге вам необходимо запрограммировать вычисление градиента функции стоимости регуляризованной логистической регрессии. Найдите в программе функцию gradient_function и дополните ее кодом до рабочего состояния. Функция принимает на вход такие же параметры, как и cost_function. Функция должна возвращать значение вектор градиента функции стоимости.

После того, как функция готова, запустите программу. Программа вычисляет градиент для предварительно заданных параметров. Полученные значения должны быть близки к ожидаемым. Удалите оператор return, чтобы перейти к следующему шагу.

4 Обучение линейной регрессии

На данном шаге выполняется обучение модели регуляризованной линейной регрессии с использованием написанных вами функций cost_function и gradient_function. Вам не нужно писать для этого свой код. В программе имеется функция train_model, которая выполняет все действия с использованием библиотечной функции minimize.

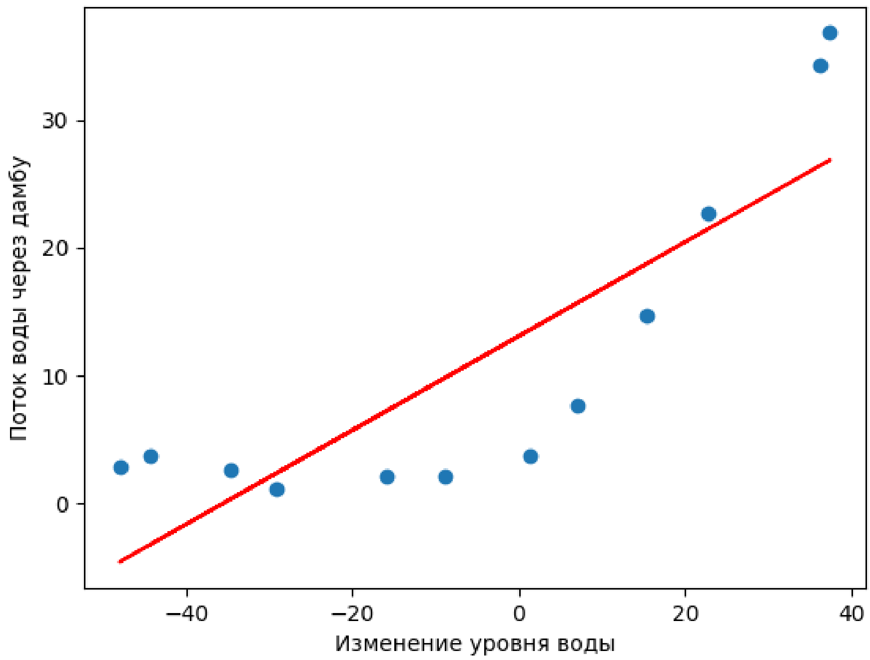

Рис. 2. Отображение графика модели на выборке

После обучения модели необходимо отобразить на обучающей выборке график регрессионной модели для наглядной визуализации. Вернитесь к функции display_data и в разделе для шага 4 дополните ее кодом для отображения модели.

Рекомендация: можно вычислить значения гипотезы для всех точек X обучающей выборки и отобразить поверх точек выборки линией другого цвета.

Пример графика показан на рис. 2.

Удалите оператор return, чтобы перейти к следующему шагу.

5 Построение кривых обучения

На данном шаге вам необходимо построить кривые обучения для полученной регрессионной модели и оценить возможные проблемы у модели со смещением или дисперсией.

Найдите в программе функцию learning_curves и дополните ее кодом для формирования кривых обучения модели. Функция принимает на вход переменные:

X, y – обучающая выборка;

Xval, yval – валидационная выборка;

lamb – параметр регуляризации.

Функция должна возвращать два вектора:

Jtrain – вектор значений ошибки на обучении от величины обучающей выборки (длиной m);

Jval – вектор значений ошибки на валидации от величины обучающей выборки (длиной m).

Рекомендация к реализации: можно выбрать в отдельный срез подмножество обучающей выборки, например, X[:5] и y[:5] будут соответствовать первым пяти элементам выборки. С использованием функции train_model можно обучить модель и получить вектор значений theta для нее. Используя функцию cost_function можно вычислить величину функции стоимости.

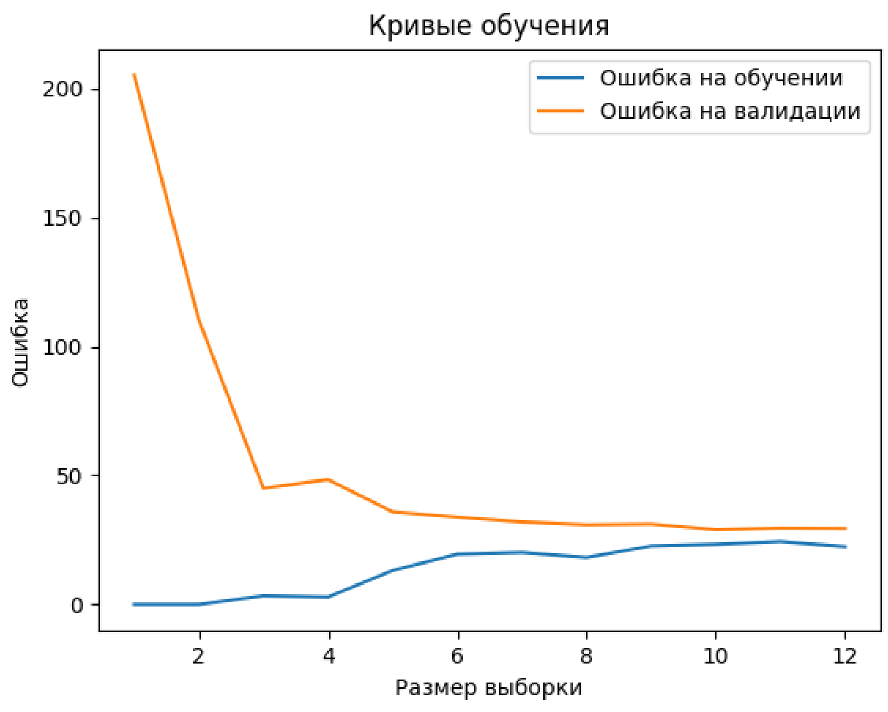

Рис. 3. Кривые обучения.

Обратите внимание:

1) Обучение модели выполняется с параметром регуляризации, тогда как вычисление Jtrain и Jval выполняется без регуляризации;
2) Вычисление Jtrain выполняется от тех же данных, на которых модель обучалась (часть обучающей выборки), тогда как Jval всегда вычисляется от всей валидационной выборки.

После того, как функция готова, запустите программу. Она вызывает написанную вами функцию для вычисления значений Jtrain и Jval, а затем отображает кривые обучения. Если все сделано верно, вы должны получить графики, подобные рис. 3. Оцените какие есть проблемы у этой модели.

Удалите оператор return, чтобы перейти к следующему шагу.

6 Формирование полиномиальной модели

Полученная модель очень проста и явно подвержена большому смещению. Для усложнения модели добавим в нее полиномиальных признаков. Вместо модели:

$$h_\theta(x) = \theta_0 + \theta_1 x$$

будем использовать модель:

$$h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \cdots + \theta_p x^p.$$

Как описано в разделе 2.3 учебного пособия, мы не используем более полиномиальную функцию гипотезы. Вместо этого мы добавляем в обучающую выборку дополнительные столбцы с вычисленными значениями различных степеней от x, и используем множественную линейную регрессию. То есть, обучающая выборка вместо вида

$$X = \begin{bmatrix} x^{(1)} \\ x^{(2)} \\ \dots \\ x^{(m)} \end{bmatrix}$$

будет представляться матрицей

$$Xpoly = \begin{bmatrix} x^{(1)} & \left(x^{(1)}\right)^2 & \dots & \left(x^{(1)}\right)^p \\ x^{(2)} & \left(x^{(2)}\right)^2 & \dots & \left(x^{(2)}\right)^p \\ \dots & \dots & \dots & \dots \\ x^{(m)} & \left(x^{(m)}\right)^2 & \dots & \left(x^{(m)}\right)^p \end{bmatrix}.$$

Найдите в программе функцию poly_features и дополните ее кодом для формирования обучающей выборки с полиномиальными признаками заданной степени. Функция принимает на вход:

X – вектор исходных признаков модели (одномерный);

p – требуемая степень полиномиальных признаков.

Функция должна возвращать матрицу $Xpoly$.

Поскольку величины $x^{(i)}$ и $\left(x^{(i)}\right)^p$ могут очень сильно отличаться друг от друга по масштабу, для корректной работы модели необходимо выполнить нормализацию признаков (см. раздел 2.2 учебника).

Найдите в программе функцию feature_normalize и дополните ее кодом для нормализации полученной матрицы $Xpoly$.

Функция получает на вход матрицу X для нормализации, а возвращает нормализованную матрицу Xnorm и вектора нормализации: mu – вектор средних значений μ; sigma – вектор стандартных отклонений σ.

Пусть необходимы нормализовать матрицу признаков

$$X = \begin{bmatrix} x_{11} & \cdots & x_{1n} \\ \cdots & \cdots & \cdots \\ x_{m1} & \cdots & x_{mn} \end{bmatrix} \in \mathbb{R}^{m \times n}.$$

Нормализацию необходимо выполнить для каждого столбца матрицы X отдельно. Сначала вычисляется вектор средних значений матрицы X:

$$\mu = [\mu_1 \quad \cdots \quad \mu_n],$$

где μ_1 – среднее значение i-го столбца матрицы X. Далее вычисляется вектор стандартных отклонений матрицы X:

$$\sigma = [\sigma_1 \quad \cdots \quad \sigma_n],$$

где σ_1 – стандартное отклонение i-го столбца матрицы X.

Для вычисления среднего значения можно воспользоваться функцией **mean**, а для стандартного отклонения применить функцию **std** библиотеки NumPy (подробнее в руководстве «Практическая работа с NumPy»).

Нормализованная матрица признаков может быть вычислена в NumPy одним выражением как:

$$Xnorm = \frac{X - \mu}{\sigma}.$$

После того, как обе функции готовы, запустите программу. Программа оценивает размеры и статистические характеристики полученной функциями матрицы. Ваши ответы должны быть близки к ожидаемым.

Удалите оператор return, чтобы перейти к следующему шагу.

7 Кривые обучения полиномиальной модели

На данном шаге работы вам не нужно дополнять программу своим кодом, все необходимые функции уже готовы. Программа выполнить обучение построенной вами полиномиальной модели и отобразить график гипотезы на обучающем наборе. Вы должны получить модель, похожую на рис. 4.

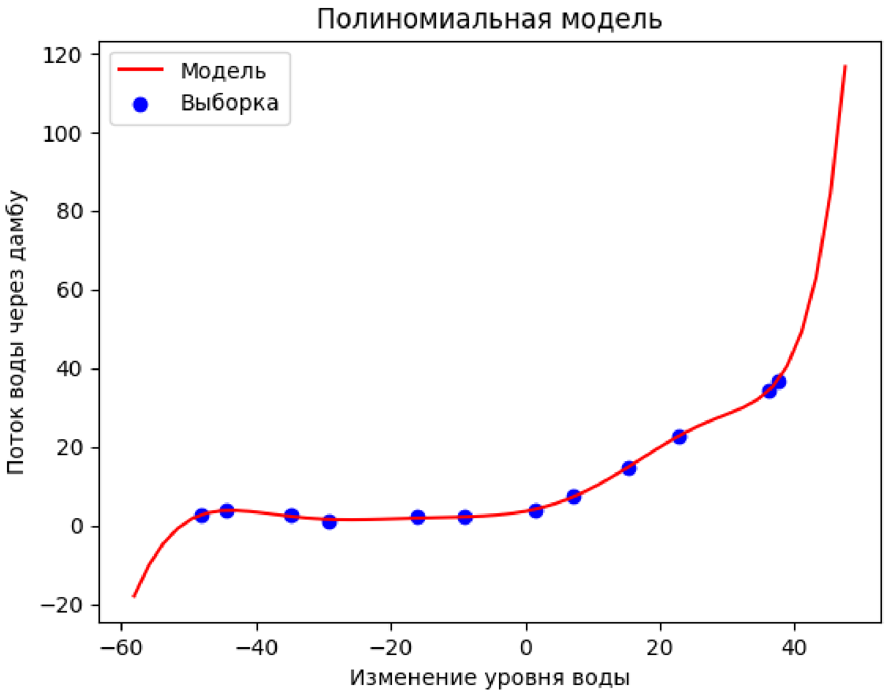

Рис. 4. Гипотеза полиномиальной модели

Далее для этой модели строятся кривые обучения с использованием написанной вами ранее функции learning_curves. Оцените кривые обучения и сделайте вывод о наличии у полиномиальной модели проблем смещение или дисперсии.

Удалите оператор return, чтобы перейти к следующему шагу.

8 Подбор параметра регуляризации

На данном шаге необходимо с использованием валидационной выборки подобрать оптимальное значение параметра регуляризации для полиномиальной модели, чтобы обеспечить ее хорошее качество (без дисперсии или смещения), как описано в разделе 6.5 учебного пособия.

Найдите в программе функцию validation_curves и дополните ее кодом так, чтобы получить данные для построения кривых ошибки на обучении и на валидации.

Функция принимает переменные:

X, y – обучающая выборка;

Xval, yval – валидационная выборка;

lambda_vec – вектор значений параметра регуляризации, в программе предлагается использовать значения: 0, 0.001, 0.003, 0.01, 0.03, 0.1, 0.3, 1, 3, 10 (примечание: вы можете предложить свои).

Функция должна вернуть два вектора, каждый равны по длине

Jtrain – вектор ошибок на обучении при использовании каждого из значений lambda_vec;

Jval – вектор ошибок на валидации при использовании каждого из значений lambda_vec.

Оба вектора Jtrain и Jval по длине совпадают с вектором lambda_vec.

Вы можете использовать функцию train_model для обучения модели регрессии с заданным значением параметра регуляризации и получения вектора theta, а затем использовать функцию cost_function, чтобы вычислить величину ошибки при полученном векторе параметров theta.

Замечание: обратите внимание, что для вычисления значения ошибки на обучении и на валидации регуляризация не используется ($\lambda = 0$).

После того, как функция готова, запустите программу. Подсчитанные функцией validation_curves ошибки выводятся на график. Определите по графику оптимальное значение λ (подсказка: самым лучшим значением является то, при котором ошибка на валидации имеет наименьшее значение).

Удалите оператор return, чтобы перейти к следующему шагу.

9 Оценка обобщающей способности модели

Полученное на предыдущем шаге значение λ будем использовать для построения оптимальной регрессионной модели и оценки ее обобщающей способности.

Вместо lamb = 0 установите оптимальное значение, определенное на прошлом шаге. Запустите программу. Она выполняет обучение модели с заданным λ и оценивает значение ошибки на тестовой выборке. Если все сделано правильно, это значение должно быть близко к ожидаемому.

На этом выполнение лабораторной работы №9 завершается.