Отлично! Давайте перейдем к третьему уроку.

Урок 3: Линейная Регрессия и Основы Математики в Машинном Обучении

Цель урока:

Понять, как работает линейная регрессия и какие математические основы необходимы для ее применения. Изучить концепции линейной алгебры и градиентного спуска, которые являются фундаментальными для многих алгоритмов машинного обучения.

Теоретическая часть (15 минут)

1. **Линейная регрессия:**
   - **Определение:** Линейная регрессия — это метод, используемый для прогнозирования непрерывных значений. Она ищет линейную зависимость между входными признаками (независимыми переменными) и выходным значением (зависимой переменной).
   - **Простая линейная регрессия:** Использует одну независимую переменную для предсказания зависимой переменной.

     $$y = \beta_0 + \beta_1 x + \epsilon$$

     где:
     - y — предсказанное значение.
     - x — независимая переменная.
     - β_0 — свободный член (пересечение с осью y).
     - β_1 — коэффициент наклона.
     - ϵ — ошибка (шум).
   - **Множественная линейная регрессия:** Использует несколько независимых переменных.

     $$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_n x_n + \epsilon$$

2. **Метод наименьших квадратов:**
   - **Цель:** Найти такие коэффициенты $\beta_0, \beta_1, \ldots, \beta_n$, которые минимизируют сумму квадратов отклонений предсказанных значений от истинных.
   - **Функция ошибки (ошибка суммарных квадратов):**

$$J(\beta) = \sum_{i=1}^{m}(y_i - \hat{y}_i)^2 = \sum_{i=1}^{m}(y_i - (\beta_0 + \beta_1 x_{i1} + \ldots + \beta_n x_{in}))^2$$

- **Решение:** Используя производные, можно получить аналитическое решение для коэффициентов:

$$\beta = (X^T X)^{-1} X^T y$$

3. **Градиентный спуск (Gradient Descent):**
   - **Определение:** Это метод оптимизации, используемый для нахождения минимального значения функции (например, функции ошибки).
   - **Идея:** Начать с произвольных значений коэффициентов и итеративно их обновлять в направлении, противоположном градиенту функции ошибки, чтобы минимизировать эту ошибку.
   - **Обновление коэффициентов:**

$$\beta_j = \beta_j - \alpha \frac{\partial J(\beta)}{\partial \beta_j}$$

     где:
     - α — скорость обучения (learning rate).
     - $\frac{\partial J(\beta)}{\partial \beta_j}$ — частная производная функции ошибки по β_j.
   - **Алгоритм:**
     a. Инициализировать коэффициенты случайными значениями.
     b. Вычислить предсказания и ошибку.
     c. Вычислить градиент функции ошибки.
     d. Обновить коэффициенты.
     e. Повторять шаги 2-4 до сходимости (когда изменение ошибки становится достаточно малым).

Практическая часть (15 минут)

Теперь давайте реализуем линейную регрессию с использованием метода наименьших квадратов и градиентного спуска в Python.

1. **Пример с использованием библиотеки `scikit-learn` (метод наименьших квадратов):**

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score

# Генерация синтетических данных
np.random.seed(0)
X = 2 * np.random.rand(100, 1)
y = 4 + 3 * X + np.random.randn(100, 1)

# Создание модели линейной регрессии
model = LinearRegression()
model.fit(X, y)

# Прогнозирование
y_pred = model.predict(X)

# Оценка модели
mse = mean_squared_error(y, y_pred)
r2 = r2_score(y, y_pred)

print("Mean squared error:", mse)
print("R2 score:", r2)

# Визуализация данных и предсказаний
plt.scatter(X, y, color='black', label='Данные')
plt.plot(X, y_pred, color='blue', linewidth=3, label='Линейная регрессия')
plt.xlabel('Признак X')
plt.ylabel('Значение Y')
plt.title('Линейная регрессия: пример сгенерированных данных')
plt.legend()
plt.show()
```

2. **Реализация линейной регрессии с использованием градиентного спуска:**

```
# Градиентный спуск для простой линейной регрессии
def gradient_descent(X, y, learning_rate=0.1, n_iterations=1000):
    m = len(y)
    X_b = np.c_[np.ones((m, 1)), X]  # Добавление вектора единиц для свободного члена (bias terr
    theta = np.random.randn(2, 1)  # Инициализация случайных значений для коэффициентов (theta)

    for iteration in range(n_iterations):
        gradients = 2/m * X_b.T.dot(X_b.dot(theta) - y)  # Вычисление градиента
        theta = theta - learning_rate * gradients  # Обновление коэффициентов

    return theta

# Выполнение градиентного спуска
theta_final = gradient_descent(X, y)

# Прогнозирование с использованием полученных коэффициентов
X_new = np.array([[0], [2]])
X_new_b = np.c_[np.ones((2, 1)), X_new]  # Добавление вектора единиц для прогнозирования
y_predict = X_new_b.dot(theta_final)

# Визуализация
plt.scatter(X, y, color='black', label='Данные')
plt.plot(X_new, y_predict, color='red', label='Градиентный спуск')
plt.xlabel('Признак X')
plt.ylabel('Значение Y')
plt.title('Линейная регрессия с использованием градиентного спуска')
plt.legend()
plt.show()

print("Коэффициенты после градиентного спуска:", theta_final.ravel())
```

Пояснение коду:

- **Линейная регрессия с использованием `scikit-learn`:** Мы используем библиотеку `scikit-learn` для быстрого создания модели линейной регрессии, которая находит наилучшие коэффициенты β методом наименьших квадратов.
- **Градиентный спуск:** Реализация метода градиентного спуска “с нуля” для иллюстрации того, как алгоритм оптимизации работает шаг за шагом, чтобы найти минимальное значение функции ошибки.

Что дальше?

Попробуйте выполнить приведенные примеры на вашей машине и поэкспериментировать с параметрами (например, скоростью обучения или числом итераций для градиентного спуска). Если возникнут вопросы по линейной регрессии, методу наименьших квадратов или градиентному спуску, или если хотите углубиться в конкретную тему, дайте знать! Мы также можем перейти к следующему уроку, если вы готовы.

Конечно, давайте углубимся в математическую часть и подробно разберем формулы, которые используются в линейной регрессии и градиентном спуске. Я постараюсь объяснить их более доступно, чтобы освежить знания и помочь понять основные концепции.

1. Линейная регрессия: Основные понятия и формулы

Простая линейная регрессия

Задача линейной регрессии — найти линейную зависимость между входными признаками и целевой переменной. В самой простой форме (с одним признаком) линейная регрессия выражается как прямая линия:

$$y = \beta_0 + \beta_1 x + \epsilon$$

- y — предсказанное значение (целевая переменная).
- x — независимая переменная (признак).
- β_0 — свободный член (intercept), значение y, когда $x = 0$. Это точка, где линия пересекает ось y.
- β_1 — коэффициент наклона (slope), который показывает, насколько изменится y при изменении x на единицу.
- ϵ — ошибка, представляющая разницу между предсказанным и истинным значением. В идеале, мы хотим, чтобы эта ошибка была как можно меньше.

Множественная линейная регрессия

Когда у нас несколько признаков ($x_1, x_2, \ldots, x_n$), модель становится:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_n x_n + \epsilon$$

Здесь:

- $x_1, x_2, \ldots, x_n$ — независимые переменные (признаки).
- $\beta_1, \beta_2, \ldots, \beta_n$ — коэффициенты, которые модель должна найти.

2. Метод наименьших квадратов (Ordinary Least Squares, OLS)

Цель линейной регрессии — найти такие значения $\beta_0, \beta_1, \ldots, \beta_n$, которые минимизируют разницу между предсказанными значениями и истинными значениями. Эта разница измеряется с помощью **функции ошибки**.

Функция ошибки (ошибка суммарных квадратов):

$$J(\beta) = \sum_{i=1}^{m} (y_i - \hat{y}_i)^2$$

- $J(\beta)$ — функция ошибки, которую мы хотим минимизировать.
- m — количество наблюдений (примеров) в наборе данных.
- y_i — истинное значение для i-го наблюдения.
- $\hat{y}_i$ — предсказанное значение для i-го наблюдения, которое вычисляется как:

$$\hat{y}_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \ldots + \beta_n x_{in}$$

Метод наименьших квадратов находит такие коэффициенты β, которые минимизируют $J(\beta)$. Решение этой задачи можно получить аналитически:

$$\beta = (X^T X)^{-1} X^T y$$

- X — матрица признаков, где каждая строка — это один пример, а каждый столбец — один признак.
- X^T — транспонированная матрица X.
- y — вектор целевых значений.
- $(X^T X)^{-1}$ — обратная матрица для произведения $X^T X$.

Этот метод позволяет найти оптимальные коэффициенты β без использования итеративных методов, таких как градиентный спуск.

3. Градиентный спуск (Gradient Descent)

Градиентный спуск — это итеративный метод оптимизации, который используется для минимизации функции ошибки, особенно когда размерность данных слишком велика для метода наименьших квадратов или нет аналитического решения.

Идея градиентного спуска:

- **Начальная точка:** Начинаем с произвольных значений коэффициентов (например, случайных).
- **Градиент:** Вычисляем градиент функции ошибки по отношению к коэффициентам. Градиент указывает направление наибольшего роста функции. Чтобы минимизировать функцию, мы должны двигаться в противоположном направлении.
- **Обновление коэффициентов:** Обновляем значения коэффициентов, перемещаясь в направлении, противоположном градиенту, с определенной скоростью (шагом).
- **Итерации:** Повторяем процесс до тех пор, пока изменение в функции ошибки не станет незначительным.

Формула обновления коэффициентов:

$$\beta_j = \beta_j - \alpha \frac{\partial J(\beta)}{\partial \beta_j}$$

- β_j — коэффициент, который обновляется.
- α — скорость обучения (learning rate), определяющая, на сколько обновляется коэффициент на каждом шаге.
- $\frac{\partial J(\beta)}{\partial \beta_j}$ — частная производная функции ошибки по коэффициенту β_j, определяющая направление и величину изменения.

Обновление всех коэффициентов:

Для всех коэффициентов (включая свободный член):

$$\beta_0 = \beta_0 - \alpha \sum_{i=1}^{m} (\hat{y}_i - y_i)$$

$$\beta_1 = \beta_1 - \alpha \sum_{i=1}^{m} (\hat{y}_i - y_i) x_{i1}$$

$$\beta_2 = \beta_2 - \alpha \sum_{i=1}^{m} (\hat{y}_i - y_i) x_{i2}$$

$$\vdots$$

$$\beta_n = \beta_n - \alpha \sum_{i=1}^{m} (\hat{y}_i - y_i) x_{in}$$

Здесь:

- $\hat{y}_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \ldots + \beta_n x_{in}$ — предсказанное значение.

4. Как это связано с линейной регрессией?

Линейная регрессия пытается найти такую прямую (или гиперплоскость в случае нескольких признаков), которая минимизирует суммарное расстояние (в виде квадратов) между точками данных и самой линией. Это позволяет нам делать прогнозы для новых данных на основе этой линии. Метод наименьших квадратов и градиентный спуск — это два способа найти оптимальные параметры этой линии.

Итог

- **Линейная регрессия** — пытается найти линейную зависимость между признаками и целевым значением.
- **Метод наименьших квадратов** — аналитический метод, который позволяет найти оптимальные коэффициенты, минимизируя сумму квадратов отклонений.
- **Градиентный спуск** — итеративный метод оптимизации, который используется, когда размерность данных слишком велика или нет аналитического решения.

Если что-то из этих объяснений все еще остается неясным, или у вас есть дополнительные вопросы, пожалуйста, дайте знать! Мы можем разобрать конкретные части еще более подробно.