МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ

Ордена Трудового Красного Знамени

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Московский технический университет связи и информатики»

Кафедра «Математическая Кибернетика и Информационные технологии»

Исследование метода обратного распространения ошибки для обучения нейронной сети

Выполнил: Студент группы

БВТ2402

Юдин Владимир

Москва

2025

Описание задачи
Дан список входных значений и соответствующих им выходных значений.
В левом столбце указано значение температуры в градусах
Цельсия, а в правом столбце — в градусах Фаренгейта.
Известно, что все указанные в таблице точки лежат на одной
прямой.
Нужно найти правило, по которому можно будет переводить
градусы Цельсия в градусы Фаренгейта (с некоторой
точностью).

*Если проверяющему так будет удобней, весь мой код с комментариями есть в колабе по ссылке:
https://colab.research.google.com/drive/1Dqb--_V-HZAMopOH-nuiP4ZR-sJxghij?usp=sharing
*Также этот документ по ссылке:
https://docs.google.com/document/d/1iwAdgyD89sCXJwNDX58FkcYSADVn6AEHgsWtUoi_ick/edit?usp=sharing

Ход работы

Задание на два плюса
Выполните прямой и обратный проход с одним обучающим примером (одна строчка таблицы).
Обновление весов должно быть выполнено 1 раз

```python
import numpy as np

#Задание на 2 плюса
# Возьму вторую строку (Обучающая выборка)

X = np.array([0.0, 1.0, 10.0, 15.0, 22.0])        # Цельсий
Y = np.array([32.0, 33.8, 50.0, 59.0, 71.6])      # Фаренгейт

# Я по списку 31-ый, так что возьму вариант |31-28|, ну то есть 3-ий

W = 41.0   # начальный вес
b = 8.0    # смещение
lr = 0.01  # learning_rate

print(f"For train on 1 example\n"
    f"start  W={W:.4f}, b={b:.4f}")

epochs = 1
for epoch in range(1, epochs+1):

    s = W * X[1] + b  # взвешенная сумма
    y_pred = s        # функция активации и наше предсказание

    # функция потерь
    error = Y[1] - y_pred
    loss = 0.5 * np.sum(error**2)

    dE_dPred = y_pred-Y[1]
    dPred_dS = 1                  # Так как ds/ds = 1
    dS_dW = X[1]
    dS_db = 1

    dE_dW = dE_dPred * dPred_dS * dS_dW
    dE_db = dE_dPred * dPred_dS * dS_db


    # Обновление параметров
    W -= lr * dE_dW
    b -= lr * dE_db

    print(f"epoch {epoch}: W={W:.4f}, b={b:.4f}")
```

Результат:

```
For train on 1 example
start  W=41.0000, b=8.0000
epoch 1: W=40.8480, b=7.8480
```

Видим, что веса обновились, произошел полный цикл обучения

Также я заинтересовался результатом, который может продемонстрировать такая сеть после одной итерации. Сравним предсказание и реальное значение по пятой строчке таблицы
Используем только прямой проход для предсказания:

```
[131] # Пусть у нас пятая строка будет тестовой
      #Сделаем просто прямой проход чтобы увидеть результат

      s = W * X[4] + b
      y_pred = s
      print(f"Predicted: {y_pred}, True value: {Y[4]}")

      Predicted: 906.5039999999999, True value: 71.6
```

Задание на третий плюс
Завершите первую эпоху обучения (с ещё одной строчкой таблицы) и выполните вторую (с теми же
двумя строчками таблицы).

```
# Задание на третий плюс

X = np.array([0.0, 1.0, 10.0, 15.0, 22.0])        # Цельсий
Y = np.array([32.0, 33.8, 50.0, 59.0, 71.6])       # Фаренгейт

W = 41.0   # начальный вес
b = 8.0    # смещение
lr = 0.01  # learning_rate


print(f"For train on 2 examples\n"
    f"start  W={W:.4f}, b={b:.4f}")

epochs = 2
n_examples = 2
losses = []

for epoch in range(1, epochs+1):
  for n in range(0, n_examples):
      s = W * X[n] + b  # взвешенная сумма
      y_pred = s        # функция активации наше предсказание

      # функция потерь
      error = Y[n] - y_pred
      loss = 0.5 * np.sum(error**2)
      losses.append(loss)

      dE_dPred = y_pred-Y[n]
      dPred_dS = 1                  # Так как ds/ds = 1
      dS_dW = X[n]
      dS_db = 1

      dE_dW = dE_dPred * dPred_dS * dS_dW
      dE_db = dE_dPred * dPred_dS * dS_db


      # Обновление параметров
      W -= lr * dE_dW
      b -= lr * dE_db

      print(f"epoch {epoch}, example {n+1}: W={W:.4f}, b={b:.4f}")
```

Таблица с весами:

```
For train on 2 examples
start  W=41.0000, b=8.0000
epoch 1, example 1: W=41.0000, b=8.2400
epoch 1, example 2: W=40.8456, b=8.0856
epoch 2, example 1: W=40.8456, b=8.3247
epoch 2, example 2: W=40.6919, b=8.1710
```

Аналогично посмотрим на результат:

```
[133] s = W * X[4] + b
      y_pred = s
      print(f"Predicted: {y_pred}, True value: {Y[4]}")

Predicted: 903.39276488, True value: 71.6
```

Дополнительно

Я решил поставить чуть больше эпох и прогнать модель по четырем примерам из выборки

Аналогичный код

```python
import numpy as np

# Задание на четвертый плюс

X = np.array([0.0, 1.0, 10.0, 15.0, 22.0])        # Цельсий
Y = np.array([32.0, 33.8, 50.0, 59.0, 71.6])       # Фаренгейт

W = 41.0   # начальный вес
b = 8.0    # смещение
lr = 0.01  # learning_rate
epochs = 5

print(f"For train on 4 examples\n"
    f"start  W={W:.4f}, b={b:.4f}")

n_examples = 4
losses = []
for epoch in range(1, epochs+1):
  for n in range(0, n_examples):
      s = W * X[n] + b  # взвешенная сумма
      y_pred = s        # функция активации наше предсказание

      # функция потерь
      error = Y[n] - y_pred
      loss = 0.5 * np.sum(error**2)
      losses.append(loss)

      dE_dPred = y_pred-Y[n]
      dPred_dS = 1                  # Так как ds/ds = 1
      dS_dW = X[n]
      dS_db = 1

      dE_dW = dE_dPred * dPred_dS * dS_dW
      dE_db = dE_dPred * dPred_dS * dS_db


      # Обновление параметров
      W -= lr * dE_dW
      b -= lr * dE_db

      print(f"epoch {epoch}, example {n+1}: W={W:.4f}, b={b:.4f}")
```

Таблица с весами:

```
For train on 4 examples
start  W=41.0000, b=8.0000
epoch 1, example 1: W=41.0000, b=8.2400
epoch 1, example 2: W=40.8456, b=8.0856
epoch 1, example 3: W=4.1914, b=4.4202
epoch 1, example 4: W=2.9477, b=4.3373
epoch 2, example 1: W=2.9477, b=4.6139
epoch 2, example 2: W=3.2101, b=4.8763
epoch 2, example 3: W=4.5124, b=5.0065
epoch 2, example 4: W=2.4586, b=4.8696
epoch 3, example 1: W=2.4586, b=5.1409
epoch 3, example 2: W=2.7206, b=5.4029
epoch 3, example 3: W=4.4597, b=5.5768
epoch 3, example 4: W=2.4388, b=5.4421
epoch 4, example 1: W=2.4388, b=5.7077
epoch 4, example 2: W=2.6954, b=5.9642
epoch 4, example 3: W=4.4036, b=6.1350
epoch 4, example 4: W=2.4253, b=6.0031
epoch 5, example 1: W=2.4253, b=6.2631
epoch 5, example 2: W=2.6764, b=6.5142
epoch 5, example 3: W=4.3486, b=6.6814
epoch 5, example 4: W=2.4121, b=6.5523
```

Результат гораздо ближе к истине

```
s = W * X[4] + b
y_pred = s
print(f"Predicted: {y_pred}, True value: {Y[4]}")
```

```
Predicted: 59.617694351951506, True value: 71.6
```