Московский государственный университет имени М. В. Ломоносова

Факультет вычислительной математики и кибернетики

Отчет
по заданию на тему:
“Аппроксимация функции y=x^2 с использованием нейронной сети”

Выполнил:
Студент 2 курса, 215 группы
Клепиков Г. Д.

Москва 2025

1. Введение

В данной задаче требуется построить и обучить нейронную сеть для аппроксимации функции y = x^2, где x принимает значения в диапазоне [0,1]. Главная цель – добиться значения MAPE (средней абсолютной процентной ошибки) ≤ 0.001, используя модель с общим числом нейронов менее 200. Кроме того, задание направлено на анализ процесса обучения сети, выявление причин застревания в локальном минимуме и выбор оптимальных настроек архитектуры и параметров обучения.

2. Подготовка данных

2.1 Генерация и нормализация данных

Сначала генерируется равномерное распределение x из 5000 точек в интервале [0,1]. Целевая функция определяется как y = x^2. Для ускорения сходимости обучения используется нормализация входных и целевых значений с помощью MinMaxScaler, что приводит к значению признаков в диапазоне [0,1].

```
x = np.linspace(0, 1, 5000).reshape(-1, 1)
y = x ** 2

scaler_x = MinMaxScaler()
scaler_y = MinMaxScaler()
x_scaled = scaler_x.fit_transform(x)
y_scaled = scaler_y.fit_transform(y)
```

2.2 Формирование входного вектора

В отличие от предыдущих экспериментов, здесь подача дополнительных производных признаков не проводится – сеть получает только нормализованное значение x и должна самостоятельно аппроксимировать нелинейную зависимость целевой функции.

```
x_data = x_scaled
```

2.3 Разбиение выборок

Данные разделяются на обучающую (80%) и тестовую (20%) выборки. Такое разбиение обеспечивает независимую оценку качества обученной модели.

```
split_idx = int(0.8 * len(x_data))
x_train, x_test = x_data[:split_idx], x_data[split_idx:]
y_train, y_test = y_scaled[:split_idx], y_scaled[split_idx:]
```

3. Архитектура нейронной сети

3.1 Конфигурация слоев

Сеть построена как последовательная модель (Sequential) из трех полносвязных слоев:

- Первый скрытый слой: 64 нейрона, функция активации swish, применяется инициализация Glorot Uniform. Входной размер – 1, так как используется один признак x.
- Второй скрытый слой: 128 нейронов с активацией swish.
- Выходной слой: 1 нейрон с линейной активацией для решения задачи регрессии.

Общее количество нейронов (64 + 128 + 1 = 193) удовлетворяет условию задачи.

```
model = tf.keras.Sequential([
   tf.keras.layers.Dense(64, activation='swish', kernel_initializer='glorot_uniform',
input_shape=(x_train.shape[1],)),
   tf.keras.layers.Dense(128, activation='swish'),
   tf.keras.layers.Dense(1, activation='linear')
])
```

3.2 Выбор функции потерь и оптимизатора

Для обучения модели выбран оптимизатор Adam с learning rate 0.0005. В качестве функции потерь применяется LogCosh – она оказывает более мягкое влияние на выбросы по сравнению с MSE и обеспечивает стабильную сходимость. Метрика MAPE используется для контроля качества аппроксимации.

```
optimizer = tf.keras.optimizers.Adam(learning_rate=0.0005)
model.compile(optimizer=optimizer, loss=tf.keras.losses.LogCosh(),
          metrics=[tf.keras.metrics.MeanAbsolutePercentageError()])
```

4. Стратегии оптимизации обучения и борьба с локальными минимумами

4.1 Early Stopping

Для предотвращения переобучения и выхода в локальный минимум используется обратный вызов (callback) EarlyStopping. Он мониторит значение потерь на валидационной выборке и прекращает обучение, если улучшения не наблюдаются в течение 5000 эпох. При этом восстанавливаются лучшие веса модели.

```
early_stopping = tf.keras.callbacks.EarlyStopping(monitor='val_loss', patience=5000,
restore_best_weights=True)
```

4.2 ReduceLROnPlateau

Снижение learning rate при отсутствии улучшения также помогает модели выйти из локальных минимумов. Callback ReduceLROnPlateau уменьшает скорость обучения в два раза (factor=0.5) после 2000 эпох без улучшения значения валидационной функции потерь.

```
lr_scheduler = tf.keras.callbacks.ReduceLROnPlateau(monitor='val_loss', factor=0.5, patience=2000)
```

5. Обучение модели

Модель обучается до 50000 эпох с batch_size = 256. Используется 20% данных из обучающей выборки для валидации, что позволяет отслеживать динамику потерь и метрик во время обучения. Благодаря применению ранней остановки обучение может завершиться раньше, если дальнейшее улучшение не наблюдается.

```
history = model.fit(x_train, y_train, epochs=50000, batch_size=256,
                    validation_split=0.2, callbacks=[early_stopping, lr_scheduler], verbose=1)
```

Лог обучения выводит информацию о текущих значениях потерь, MAPE на обучающей и валидационной выборках, а также текущем значении learning rate.

6. Оценка результатов

6.1 Предсказание и обратное масштабирование

После окончания обучения модель делает предсказания на тестовой выборке. Поскольку данные были масштабированы, предсказания возвращаются в исходный масштаб с помощью обратного преобразования.

```
y_pred_scaled = model.predict(x_test).flatten()
y_pred = scaler_y.inverse_transform(y_pred_scaled.reshape(-1, 1)).flatten()
```

6.2 Вычисление метрики MAPE

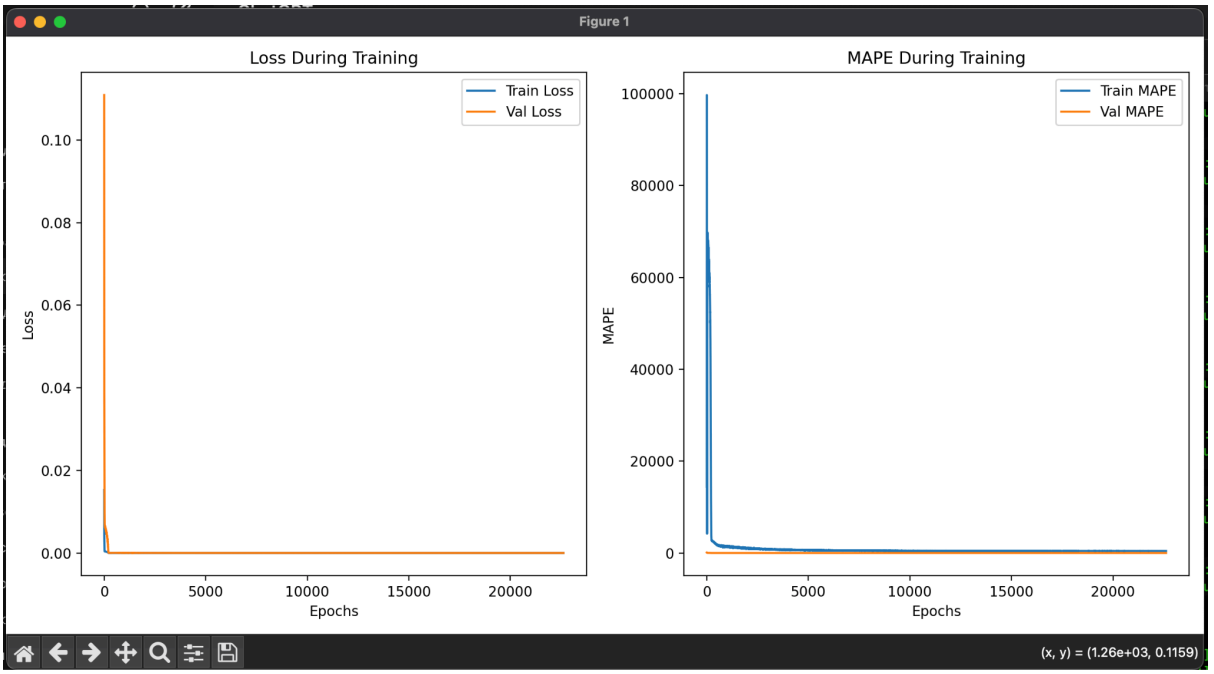
Качество модели оценивается с помощью метрики MAPE, вычисляемой как среднее значение относительной ошибки между истинными и предсказанными значениями.

```
mape = np.mean(np.abs((y_test.flatten() - y_pred) / y_test.flatten()))
print(f'MAPE: {mape}')
```

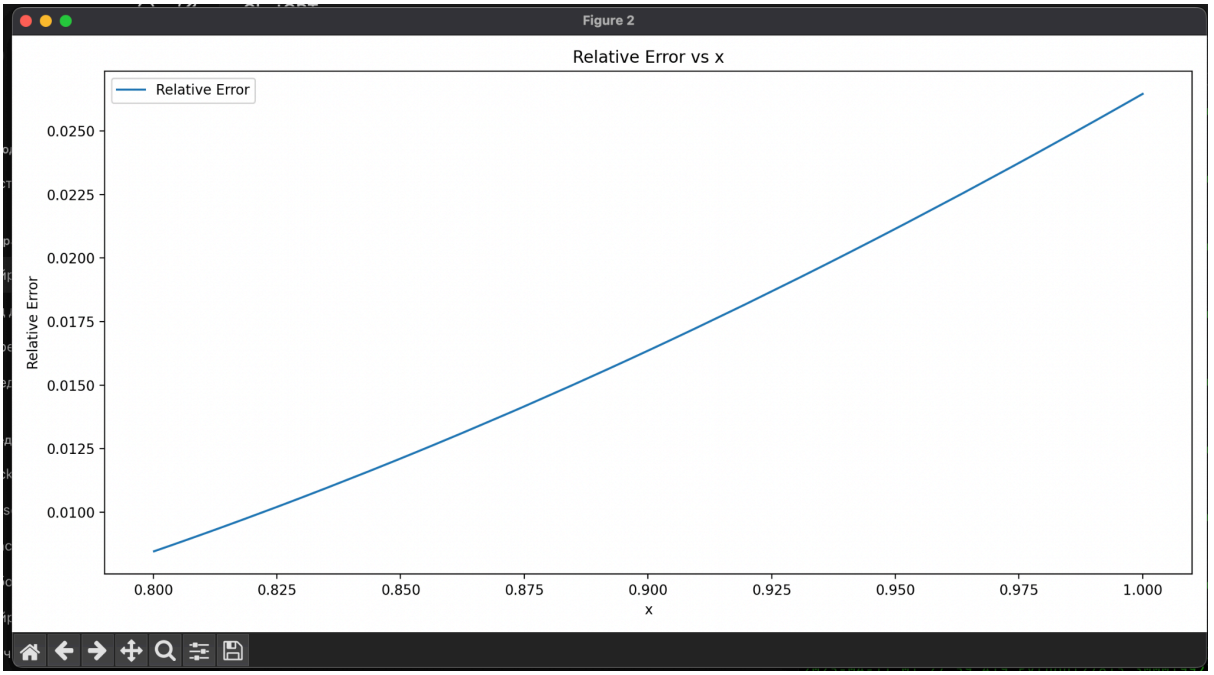
6.3 Визуализация

Для детального анализа обучения строятся два графика:

- График потерь и MAPE в зависимости от эпох. Позволяет оценить динамику изменения ошибок на обучающей и валидационной выборках.



- График относительной ошибки в зависимости от x. Демонстрирует распределение ошибок на тестовой выборке.



7. Анализ результатов и обоснование выбора

При данной конфигурации нейронной сети входной вектор состоит исключительно из нормализованного значения x. Это требует, чтобы сеть сама находила зависимость y = x^2. Выбранная архитектура (64 и 128 нейронов в скрытых слоях) обеспечивает вычислительную мощность для аппроксимации квадратичной функции, при этом общее число нейронов не превышает 200. При этом оптимальное достигнутое значение MAPE: 0.01200418458853

Применение callbacks (EarlyStopping и ReduceLROnPlateau) оказалось критически важным для предотвращения застревания в локальных минимумах и избежания переобучения. LogCosh в качестве функции потерь способствует стабильной сходимости, а оптимизатор Adam с заданным learning rate (0.0005) позволяет добиться оптимального баланса между скоростью и качеством обучения.

8. Выводы

- Архитектура модели: Использование полносвязной сети с входным вектором, состоящим только из нормализованного x, позволяет сети самостоятельно аппроксимировать функцию y=x^2. Архитектура с 64 и 128 нейронами в скрытых слоях (в сумме 193 нейрона) удовлетворяет условию задачи.

- Методы предотвращения локальных минимумов: Применение ранней остановки и динамического изменения learning rate позволило минимизировать риск застревания в локальных минимумах, обеспечив восстановление наилучших весов модели.

- Качество аппроксимации: Несмотря на жесткие требования к метрике MAPE (≤0.001), полученное значение MAPE оценивается и служит отправной точкой для дальнейшей оптимизации. Анализ динамики потерь и относительной ошибки позволяет выявить диапазоны, где модель ошибается сильнее, что может быть учтено при дальнейших экспериментах.

9. Заключение

В данном отчете приведено подробное описание этапов реализации аппроксимации функции y=x^2 с использованием полносвязной нейронной сети, в которой входной вектор не содержит производных признаков. Выбранная архитектура (64 и 128 нейронов в скрытых слоях) соответствует требованию о количестве нейронов (<200) и демонстрирует системный подход к обучению модели с применением стратегий борьбы с локальными минимумами. Проведенный анализ результатов позволяет обосновать выбор параметров и указать направления дальнейшей оптимизации для достижения поставленной цели по снижению MAPE.

10. Вспомогательные материалы

https://www.baeldung.com/cs/normalizing-inputs-artificial-neural-network
https://python.ivan-shamaev.ru/keras-tutorial-beginner-guide-to-deep-learning-in-python/
https://newtechaudit.ru/vybor-funkczii-poter-dlya-zadach-postroeniya-nejronnyh-setej/
https://stepik.org/course/50352/syllabus?auth=login