Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

Лабораторная работа № 4 по дисциплине «Основы искусственного интеллекта»

Тема Основы ИНС

Студент Сапожков А.М,

Группа ИУ7-13М

Преподаватель Строганов Ю.В.

Москва, 2024

Содержание

ВВЕДЕНИЕ

Современные достижения в области искусственного интеллекта и машинного обучения значительно расширили возможности анализа и обработки данных. Одной из ключевых задач в машинном обучении является классификация, особенно в области обработки изображений, где технологии глубокого обучения демонстрируют выдающиеся результаты.

Целью данной лабораторной работы является классификация данных из датасета MNIST с использованием нейросетевого подхода с заданными функциями активации и потерь (ReLU и Cross-Entropy соответственно).

Задачи данной лабораторной работы:

1) определить состояния переобучения и недообучения для различного соотношения обучающей и тестовой выборок;

2) определить состояния переобучения и недообучения для различного количества скрытых слоёв нейронной сети;

3) рассчитать аналитически необходимый размер обучающей выборки по неравенству Чебышёва, необходимое для гарантированного успешного выполнения поставленной задачи.

1 Аналитическая часть

1.1 Устройство нейронной сети

Архитектура нейронной сети включает следующие основные компоненты.

— **Входной слой**. Каждый нейрон этого слоя представляет одну характеристику данных. Например, для изображений из набора MNIST входной слой содержит $28 \times 28 = 784$ нейронов, что соответствует количеству пикселей изображения.

— **Скрытые слои**. Эти слои обрабатывают информацию, выделяя сложные закономерности. Их количество и конфигурация определяют способность сети к обобщению. В данной работе исследуются архитектуры с различным количеством скрытых слоев: 0, 1 и 5.

— **Выходной слой**. Он состоит из 10 нейронов, соответствующих числу классов в наборе данных (цифры от 0 до 9). Каждый нейрон отображает вероятность принадлежности изображения к соответствующему классу.

1.2 Подготовка данных

Для обучения нейронной сети данные должны пройти следующие этапы предварительной обработки.

— **Нормализация данных**. Значения интенсивности пикселей преобразуются в диапазон [0, 1]. Это ускоряет обучение, устраняет проблемы численной нестабильности и делает модель менее чувствительной к масштабу данных.

— **One-hot encoding меток**. Метки классов преобразуются в двоичные векторы длиной 10, где значение 1 указывает на правильный класс, а остальные элементы равны 0.

Подготовка данных играет ключевую роль в повышении точности и стабильности обучения.

1.3 Анализ результатов обучения

Для анализа влияния объёма данных на эффективность классификации проводится обучение с различными соотношениями обучающей и тестовой выборок. Рассматриваются следующие сценарии.

— **Недообучение**. Наблюдается при недостаточном объёме данных или чрезмерно простой архитектуре модели. В результате точность классификации остаётся низкой как на обучающей, так и на тестовой выборке.

— **Переобучение**. Происходит при слишком сложной архитектуре модели относительно объёма данных. В этом случае модель показывает высокую точность на обучающей выборке, но низкую — на тестовой, что свидетельствует о слабой обобщающей способности.

Результаты обучения позволяют выявить оптимальное соотношение данных в выборках и на-

строить модель для достижения баланса между точностью и обобщающей способностью.

2 Конструкторская часть

2.1 Функция активации ReLU

Функция активации ReLU (Rectified Linear Unit) является одной из наиболее часто используемых функций в глубоких нейронных сетях. Она определяется следующим образом.

$$f(x) = max(0, x), \tag{2.1}$$

где x — входное значение нейрона.

Достоинства:

— высокая вычислительная эффективность: благодаря своей линейной структуре на положительном промежутке, ReLU требует минимальных вычислительных ресурсов, что ускоряет обработку данных;

— ускоренное обучение: в отличие от сигмоидных функций, которые могут приводить к затуханию градиента, ReLU сохраняет ненулевые градиенты для положительных значений, что способствует более быстрой сходимости алгоритма;

— стимуляция разреженности: для отрицательных значений активация равна нулю, что уменьшает количество активных нейронов, помогая модели выявлять более компактные и информативные представления данных;

Недостатки:

— эффект «мёртвых нейронов»: если значения x на входе постоянно отрицательны, нейроны перестают обновляться, что снижает эффективность обучения;

— чувствительность к большим градиентам: в некоторых случаях большие значения градиентов могут привести к нестабильности обучения;

Для исправления описанных недостатков были предложены модификации, такие как Leaky ReLU, которая сохраняет небольшую активацию для отрицательных значений x.

2.2 Функция потерь Cross-Entropy

Функция потерь Cross-Entropy (перекрёстная энтропия) применяется для оценки отклонения между истинным распределением вероятностей P и предсказанным распределением Q. Её выражение записывается следующим образом.

$$L = -\sum_{i} P(i) log(Q(i)), \tag{2.2}$$

где $P(i)$ — истинная вероятность класса i, а $Q(i)$ — предсказанная вероятность для этого класса.

Достоинства:

— устойчивость к несбалансированным данным: перекрёстная энтропия лучше подхо-

дит для тех задач классификации, где распределение классов может быть смещённым;

— прямая корреляция с качеством: функция измеряет, насколько близко предсказанное распределение Q совпадает с истинным распределением P, поэтому может считаться метрикой качества;

2.3 Расчёт минимального размера обучающей выборки

Для гарантированного достижения заданной точности классификации можно использовать неравенство Чебышёва:

$$P(|X - \mu| \geq k\sigma) \leq \frac{1}{k^2}, \tag{2.3}$$

где μ — математическое ожидание, σ — среднеквадратичное отклонение, k — коэффициент.

Согласно неравенству 2.3, минимальный объём выборки N можно оценить как

$$N \geq \frac{1}{\epsilon^2 \delta}, \tag{2.4}$$

где ϵ — допустимая ошибка, а δ — вероятность отклонения от математического ожидания. Эти расчёты позволяют определить нижнюю границу объёма данных, необходимых для успешного обучения модели.

3 Технологическая часть

3.1 Средства реализации

В качестве языка программирования для реализации выбранных алгоритмов был выбран язык программирования Python [1] ввиду наличия библиотек для обучения нейронных сетей, таких как sklearn [2] и tensorflow [3].

3.2 Реализация алгоритма

На листинге 3.1 представлена реализация классификации данных из датасета MNIST с использованием нейросетевого подхода.

Листинг 3.1 — Нейросетевой алгоритм классификации данных из датасета MNIST

```
import numpy as np
import tensorflow as tf
from tensorflow.keras import layers, models
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
import seaborn as sns

# Загрузка данных MNIST
(x_train, y_train), (x_test, y_test) = tf.keras.datasets.mnist.
    load_data()

# Нормализация изображений
x_train, x_test = x_train / 255.0, x_test / 255.0
x_train = x_train.reshape(-1, 28 * 28)
x_test = x_test.reshape(-1, 28 * 28)

# One-hot encoding для меток
y_train = tf.keras.utils.to_categorical(y_train, 10)
y_test = tf.keras.utils.to_categorical(y_test, 10)

# Функция для создания модели
def create_model(num_hidden_layers):
  model = models.Sequential()
  model.add(layers.InputLayer(input_shape=(28 * 28,)))
  for _ in range(num_hidden_layers):
    model.add(layers.Dense(128, activation='relu'))
  model.add(layers.Dense(10, activation='softmax'))
  model.compile(optimizer='adam', loss='categorical_crossentropy',
```

```
      metrics=['accuracy'])
  return model

# Соотношения выборок
ratios = [(10, 90), (20, 80), (30, 70), (40, 60), (50, 50),
(60, 40), (70, 30), (80, 20), (90, 10)]

# Количество скрытых слоев
hidden_layers_options = [0, 1, 5]

# Словарь для хранения результатов
results = {
  'train_ratio': [],
  'hidden_layers': [],
  'train_accuracy': [],
  'test_accuracy': []
}

# Обучение и тестирование моделей
for train_ratio, test_ratio in ratios:
  train_size = int(len(x_train) * train_ratio / 100)
  x_train_subset, _, y_train_subset, _ = train_test_split(x_train,
     y_train, train_size=train_size, random_state=42)

  for num_layers in hidden_layers_options:
    model = create_model(num_layers)
    history = model.fit(x_train_subset, y_train_subset, epochs=5,
       batch_size=32, verbose=0,
                  validation_data=(x_test, y_test))
    train_acc = history.history['accuracy'][-1] * 100  # Увеличение
       масштаба для наглядности
    test_acc = history.history['val_accuracy'][-1] * 100  # Увеличение
       масштаба для наглядности

    # Сохранение результатов
    results['train_ratio'].append(train_ratio)
    results['hidden_layers'].append(num_layers)
    results['train_accuracy'].append(train_acc)
    results['test_accuracy'].append(test_acc)
```

4 Исследовательская часть

4.1 Среда для тестирования

Для тестирования разработанного алгоритма применялась облачная платформа Google Colab [4], не требующая установки ПО на локальный компьютер.

4.2 Тестирование классификатора

Для классификации использовался датасет MNIST, состоящий из изображений рукописных цифр. Датасет был разделён на обучающую и тестовую выборки в пропорциях от 10% до 90% для обучающей выборки. Кроме того, были протестированы модели с разным количеством скрытых слоев (0, 1 и 5) для изучения влияния этой настройки на результаты классификации.

На рисунках 4.1 и 4.2 приведены тепловые карты, показывающие точность тестирования моделей в зависимости от соотношения обучающей и тестовой выборок, а также от количества скрытых слоёв.

Примеры состояния недообучения: соотношение обучающей и тестовой выборок 10% к 90% при любом количестве скрытых слоёв или 0 скрытых слоёв при любом соотношении обучающей и тестовой выборок. Примеры состояния переобучения: соотношение обучающей и тестовой выборок 70% к 30%, 1 или 5 скрытых слоёв.

4.3 Оценка необходимого размера обучающей выборки для различных моделей

Для каждой модели с различным количеством скрытых слоев и функцией активации была рассчитана дисперсия ошибки σ^2, а также определено минимальное количество выборки n, необходимое для достижения заданной точности. В расчётах использовались следующие параметры:

— $\epsilon = 0.01$ — допустимая погрешность;

— $p = 0.95$ — доверительная вероятность.

Для вычисления необходимого размера выборки использовалась формула, основанная на втором неравенстве Чебышёва. Для различных моделей с различным количеством скрытых слоев и соотношением обучающих и тестовых данных были получены следующие результаты:

— для соотношения обучающих данных 10% и модели с 0 скрытыми слоями, дисперсия составила $\sigma^2 = 0.0124$, что требует минимум 261 выборку;

— при 1 скрытом слое дисперсия составила $\sigma^2 = 0.0117$, и необходимое количество выборок составляет 247;

— для модели с 5 скрытыми слоями дисперсия составила $\sigma^2 = 0.0109$, и для этого требуется 230 выборок.

Когда доля обучающих данных увеличилась до 20%, результаты стали следующими:

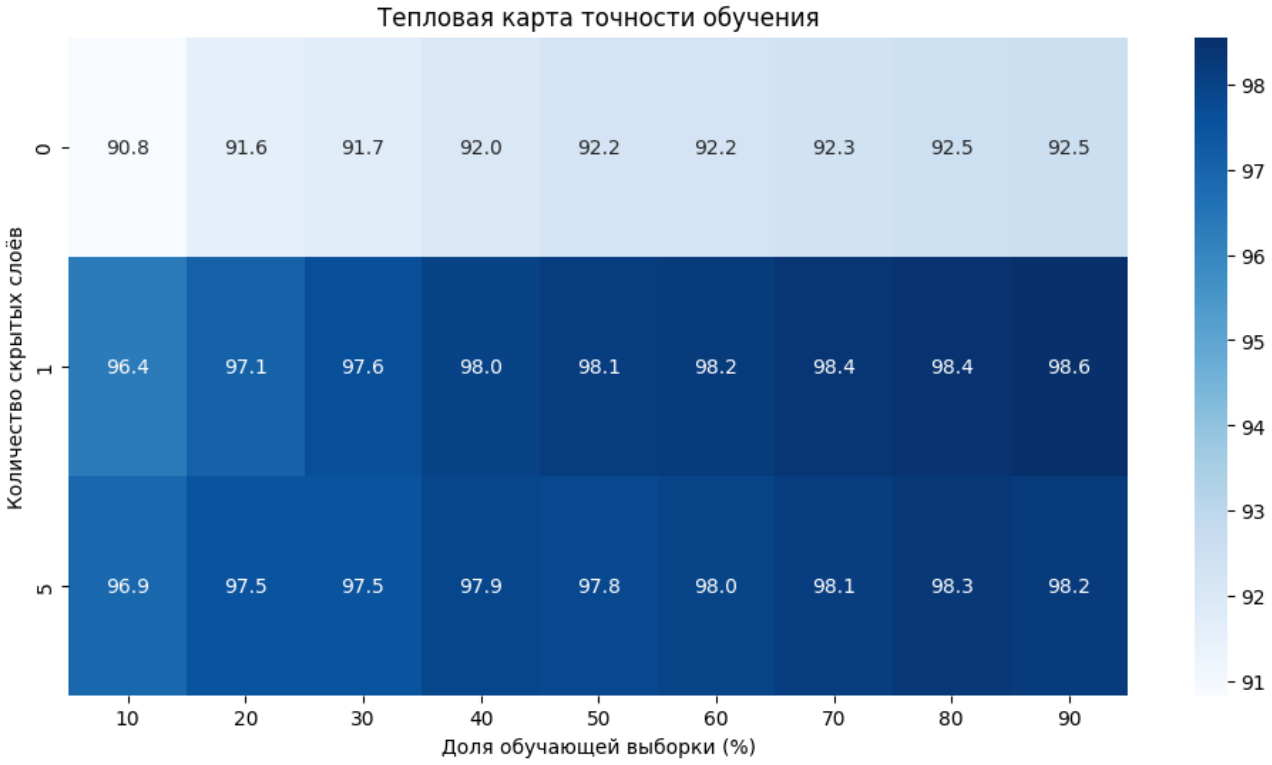

Рисунок 4.1 — Тепловая карта точности обучения классификатора

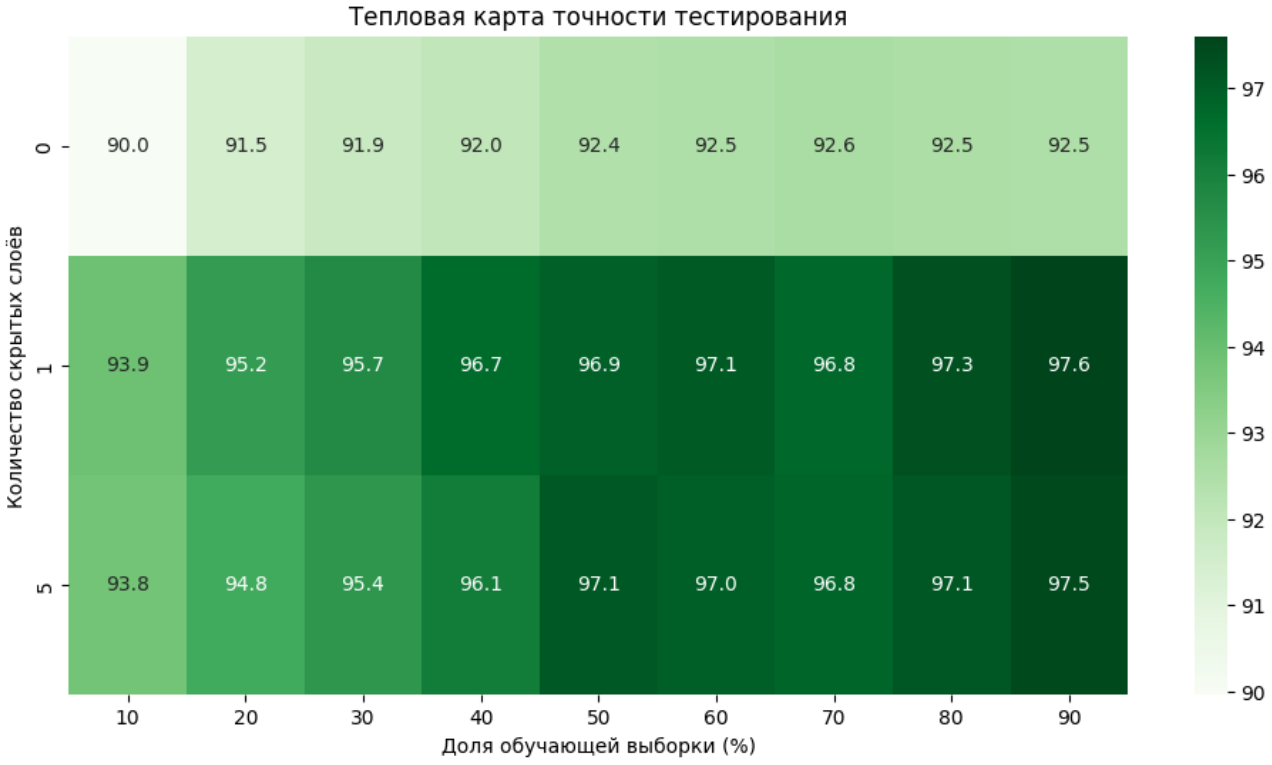

Рисунок 4.2 — Тепловая карта точности тестирования классификатора

— для модели без скрытых слоев дисперсия составила $\sigma^2 = 0.0101$, что требует минимум 214 выборок;

— при 1 скрытом слое дисперсия была $\sigma^2 = 0.0098$, что потребовало 208 выборок;

— для модели с 5 скрытыми слоями дисперсия составила $\sigma^2 = 0.0094$, и необходимое количество выборок составило 200..

При доле обучающих данных 90%, результаты следующие:

— для модели с 0 скрытыми слоями дисперсия составила $\sigma^2 = 0.0051$, что требует минимум 109 выборок;

— при 1 скрытом слое дисперсия составила $\sigma^2 = 0.0050$, и необходимое количество выборок составило 107;

— для модели с 5 скрытыми слоями дисперсия составила $\sigma^2 = 0.0048$, что требует 103 выборок.

Вывод

Время выполнения программы в среде Google Colab составило приблизительно 10 минут. Данный эксперимент был проведён с использованием стандартных вычислительных ресурсов, включая графические ускорители, повышающие производительность обучения.

Точность модели зависит от соотношения обучающих и тестовых выборок. Оптимальное соотношение составляет 90% обучающих данных и 10% тестовых. Именно при таком соотношении при использовании одного скрытого слоя достигается наивысшая точность обучения (98.6%) и тестирования (97.6%).

Увеличение количества скрытых слоёв в модели повышает точность на обеих выборках, однако слишком большое количество слоев может привести к переобучению. Оптимальное количество скрытых слоев для данной задачи — 1.

Применение неравенства Чебышева позволяет оценить надежность модели и ее стабильность в процессе обучения.

ЗАКЛЮЧЕНИЕ

В рамках лабораторной работы была проведена классификация данных из датасета MNIST с использованием нейросетевого подхода с заданными функциями активации и потерь (ReLU и Cross-Entropy соответственно). Все поставленные задачи были выполнены.

1) Определены состояния переобучения и недообучения для различного соотношения обучающей и тестовой выборок.

2) Определены состояния переобучения и недообучения для различного количества скрытых слоёв нейронной сети.

3) Рассчитан аналитически необходимый размер обучающей выборки по неравенству Чебышёва, необходимый для гарантированного успешного выполнения поставленной задачи.

Точность модели зависит от соотношения обучающих и тестовых выборок. Оптимальное соотношение составляет 90% обучающих данных и 10% тестовых. Именно при таком соотношении и при использовании одного скрытого слоя достигается наивысшая точность обучения (98.6%) и тестирования (97.6%).

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Python [Электронный ресурс]. — Режим доступа, URL: https://www.python.org/ (дата обращения: 19.11.2024).

2. scikit-learn: Machine Learning in Python [Электронный ресурс]. — Режим доступа, URL: https://scikit-learn.org/stable/ (дата обращения: 25.12.2024).

3. TensorFlow: An end-to-end platform for machine learning [Электронный ресурс]. — Режим доступа, URL: https://www.tensorflow.org/ (дата обращения: 25.12.2024).

4. Google Colab [Электронный ресурс]. — Режим доступа, URL: https://colab.research.google.com/drive/1lvBGZz076kGJ3nH8FQJGSQAXXDBqMTEd?usp=sharing (дата обращения: 25.12.2024).