ПРАКТИЧЕСКАЯ РАБОТА №1: ПОЛНОСВЯЗНАЯ НЕЙРОННАЯ СЕТЬ

Цель

Изучить процесс построения полносвязной нейронной сети, освоить базовые методы её обучения, визуализацию процесса обучения и оценку качества на различных наборах данных.

Задание

Часть 1: Общий пример на MNIST

6. Импортируйте необходимые библиотеки (TensorFlow/PyTorch, Matplotlib, NumPy).
7. Загрузите датасет MNIST (из библиотек keras.datasets или torchvision.datasets). Разделите его на обучающую и тестовую выборки.
8. Постройте полносвязную нейронную сеть с использованием следующих параметров:

- Входной слой, преобразующий изображение 28x28 в вектор длины 784.
- Один или два скрытых слоя (например, с 128 и 64 нейронами).
- Функция активации: ReLU в скрытых слоях и Softmax на выходе.
- Функция потерь: кросс-энтропия.
- Оптимизатор: SGD или Adam.

9. Обучите модель на обучающей выборке и оцените точность на

тестовых данных.

10. Постройте графики:

- Потери (loss) на обучающей и тестовой выборках.
- Точность классификации на обучении и тестировании.

11. Добавьте визуализацию ошибок классификации (например, изображения неверно классифицированных цифр).

Часть 2: Индивидуальные задания

1. Каждому студенту дается индивидуальный набор данных из Kaggle или OpenML (например, датасеты по классификации цветов, текстов или изображений).
2. Постройте полносвязную нейронную сеть для своей задачи классификации. Требования:

- Минимум 3 слоя (входной, один или два скрытых, выходной).
- Попробуйте разные размеры скрытых слоев (128, 256, 512 нейронов) и функции активации (ReLU, Sigmoid, Tanh).
- Подберите гиперпараметры: количество эпох, размер мини-выборки (batch size), метод оптимизации.

3. Визуализируйте процесс обучения с использованием Matplotlib или TensorBoard:

- Графики потерь и точности на обучении и тестировании.

4. Проведите анализ переобучения. Включите Dropout в архитектуру сети и сравните результаты с его использованием и без него.

Часть 3: Сравнительный анализ и защита

1. Сравните модели с разными функциями активации.

2. Выполните защиту результатов, представив:
- Основные метрики качества (точность, F1-мера).
- Влияние гиперпараметров на сходимость.

Что нужно вставить в отчет

1. Введение:
- Описание целей работы.
- Краткое описание датасета (размер, тип данных, задача классификации).
2. Описание архитектуры модели:
- Схема архитектуры нейронной сети: количество слоев, количество нейронов в каждом слое.
- Используемые функции активации, оптимизаторы, функция потерь.
3. Результаты обучения:
- Графики потерь (loss) на обучающей и тестовой выборках.
- Графики точности классификации.
- Визуализация ошибок классификации (например, неверно классифицированные изображения).
4. Сравнительный анализ:
- Влияние разных функций активации на точность классификации и скорость сходимости.
- Сравнение моделей с Dropout и без него (графики потерь, точности).
5. Анализ гиперпараметров:
- Подбор оптимального количества нейронов и скрытых слоев.
- Влияние параметров оптимизации (learning rate, batch size).
6. Заключение:
- Основные выводы о том, как разные параметры влияют на обучение сети.

- Оценка точности итоговой модели.
- Возможные проблемы (переобучение, медленная сходимость) и предложения по их решению.

ТЕОРЕТИЧЕСКИЙ МАТЕРИАЛ ПО 1 ПРАКТИЧЕСКОЙ

Теоретический материал по 1 части

Полносвязная нейронная сеть (Fully Connected Neural Network, FCNN) — это тип искусственной нейронной сети, в которой каждый нейрон одного слоя связан с каждым нейроном следующего слоя. Полносвязные сети широко используются для решения задач классификации и регрессии.

Ключевые компоненты полносвязной сети:

- Входной слой принимает данные в виде числовых векторов. Для изображений размером 28x28 (например, из датасета MNIST) входной слой содержит 784 нейрона, так как каждое изображение преобразуется в вектор длиной 28x 28 = 784.
- Скрытые слои представляют собой уровни, в которых происходит обработка данных. Количество слоев и нейронов в них — гиперпараметры, которые подбираются экспериментально.
- Выходной слой содержит столько нейронов, сколько классов в задаче классификации. В случае с MNIST выходной слой содержит 10 нейронов (по одному на каждую цифру от 0 до 9).

MNIST — это классический датасет рукописных цифр, состоящий из 60 000 обучающих изображений и 10 000 тестовых. Каждое изображение имеет размер 28 x 28 пикселей и принадлежит одному из 10 классов (цифры от 0 до 9).

Основные шаги работы с датасетом:

- Загрузка данных с использованием библиотек TensorFlow или PyTorch.
- Нормализация значений пикселей, чтобы они находились в

диапазоне от 0 до 1.

- Разделение данных на обучающую и тестовую выборки.

Для классификации изображений из MNIST предлагается следующая базовая архитектура:

1. Входной слой: 784 нейрона (развёрнутое изображение).
2. Скрытые слои: один или два слоя с 128 и 64 нейронами.
3. Функции активации:
    - В скрытых слоях используется ReLU (Rectified Linear Unit), которая задается формулой 1:

$$f(x) = \max(0, x) \tag{1}$$

ReLU помогает избежать проблемы затухающих градиентов и ускоряет обучение сети.

- В выходном слое используется Softmax, которая преобразует выходы в вероятности для каждого класса (формула 2):

$$Softmax(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{n} e^{z_j}} \tag{2}$$

Основные этапы обучения:

1. Прямое распространение (forward pass):

Входные данные передаются через сеть от слоя к слою, и на выходе сети получаются предсказания.

2. Вычисление функции потерь:

Для задач классификации обычно используется кросс-энтропийная функция потерь (формула 3):

$$cross\ entropy = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{K} y_{ij} \log\left(\hat{y}_{ij}\right) \tag{3}$$

где:

- y_{ij} — истинная метка класса,
- $\hat{y}_{ij}$ — предсказанная вероятность принадлежности к классу.

3. Обратное распространение (backpropagation):

Сеть вычисляет градиенты функции потерь относительно параметров сети (весов) и обновляет их в направлении уменьшения ошибки.

4. Оптимизация:

Для обновления весов используется метод оптимизации, например SGD (стохастический градиентный спуск) или Adam.

Формула для обновления весов при использовании SGD представлена под номером 4:

$$\omega = \omega - \eta \times \nabla \text{cross entropy} \quad (4)$$

где η — скорость обучения (learning rate), а $\nabla \text{cross entropy}$ — градиент функции потерь.

Для понимания того, как сеть обучается, строятся графики потерь (loss) и точности на обучающей и тестовой выборках:

- График потерь показывает, как уменьшается ошибка модели по мере обучения.
- График точности показывает, насколько хорошо сеть классифицирует изображения на каждом этапе.

Также полезно визуализировать ошибки классификации, чтобы понять, какие цифры сеть классифицирует неверно и почему.

Возможные проблемы при обучении:

1. Переобучение:

Возникает, если сеть слишком сложна для данного набора данных. Визуально это можно увидеть, если ошибка на обучающей выборке уменьшается, а на тестовой остается высокой.

Решение:

- Использование Dropout — техники регуляризации, которая случайно “отключает” нейроны во время обучения.
- Уменьшение сложности сети (уменьшение числа слоев или нейронов).

2. Медленная сходимость:

Если сеть обучается слишком медленно, это может быть связано с плохим выбором гиперпараметров, таких как learning rate.

Решение:

- Использование адаптивных методов оптимизации, таких как Adam или RMSprop.

Итоговые шаги реализации:

1. Импорт библиотек: TensorFlow/PyTorch, NumPy, Matplotlib.
2. Загрузка и нормализация данных из MNIST.
3. Построение архитектуры полносвязной сети.
4. Обучение модели с использованием функции потерь и метода оптимизации.
5. Визуализация потерь и точности.
6. Анализ ошибок классификации.

Теоретический материал по 2 части

На этапе индивидуальных заданий каждый студент выбирает уникальный набор данных из Kaggle, OpenML или других источников. Работа с разными данными позволяет углубить знания и адаптировать архитектуру нейронной сети к специфике задачи. Основные типы данных, которые могут быть предоставлены:

- Изображения (например, CIFAR-10, Fashion MNIST, цветы Ирисов).
- Текстовые данные (например, классификация отзывов или новостей).
- Табличные данные (например, датасет по предсказанию кредитного риска).

Нейронная сеть должна быть адаптирована к специфике входных данных и задачи классификации. В ходе выполнения задания студентам предстоит поэкспериментировать с:

- Числом скрытых слоев: оптимальный вариант зависит от сложности данных. Например, для изображений может быть достаточно 2-3 слоев.
- Количество нейронов в каждом слое: эксперименты с 128, 256, 512 нейронами позволяют выявить оптимальный размер.
- Функциями активации:
    - ReLU используется для ускорения обучения.
    - Sigmoid и Tanh полезны в задачах, где важна нелинейность.
    - Softmax на выходном слое преобразует выходы в вероятности классов.

Гиперпараметры нейронной сети и их настройка

Гиперпараметры — это параметры, которые задаются до начала обучения и существенно влияют на его результат.

Learning rate (скорость обучения): определяет, насколько сильно обновляются веса после каждой итерации.

- Маленький learning rate приводит к медленной сходимости, но снижает риск пропуска оптимума.
- Слишком большой может привести к неустойчивому обучению.

Batch size (размер мини-выборки): количество примеров, используемых для одного шага обновления параметров сети.

- Меньшие значения обеспечивают быстрое обновление весов, но могут давать шумные оценки градиента.
- Большие значения уменьшают шум, но замедляют процесс обучения.

Количество эпох: это число полных проходов по обучающей выборке. Важно выбрать баланс, чтобы избежать переобучения.

Пример экспериментов:

- Learning rate: 0.001, 0.01, 0.1.
- Batch size: 16, 32, 64.
- Количество эпох: 10, 20, 30.

Регуляризация: борьба с переобучением

Переобучение (overfitting) возникает, когда модель показывает отличные результаты на обучающих данных, но низкую точность на тестовых данных.

Для борьбы с этим применяются следующие методы:

Dropout:

- Во время обучения случайным образом “отключает” нейроны с заданной вероятностью, предотвращая избыточное запоминание данных.
- На этапе тестирования все нейроны снова включаются.

Формально: если p — вероятность отключения нейрона, то на каждом шаге обновления сеть обучается с отключенными нейронами в пропорции $1 - p$.

L1 и L2 регуляризация:

- L1 регуляризация: добавляет штраф за сумму абсолютных значений весов. Это может привести к занулению ненужных весов.
- L2 регуляризация: добавляет штраф за сумму квадратов весов, что помогает избежать слишком больших весов.

Визуализация процесса обучения

Для анализа эффективности обучения используются инструменты визуализации, такие как Matplotlib и TensorBoard. Основные графики:

- Потери (loss): показывает, как изменяется ошибка модели по мере обучения.
- Точность: отображает долю правильных предсказаний на обучающей и тестовой выборках.

Пример кода для построения графиков с использованием Matplotlib представлен в листинге 1:

Листинг 1

```
import matplotlib.pyplot as plt
# Предположим, что есть списки значений потерь и точности
epochs = range(1, len(train_loss) + 1)
```

```
plt.figure(figsize=(12, 5))
# График потерь
plt.subplot(1, 2, 1)
plt.plot(epochs, train_loss, 'b', label='Training loss')
plt.plot(epochs, val_loss, 'r', label='Validation loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.title('Потери на обучении и валидации')
# График точности
plt.subplot(1, 2, 2)
plt.plot(epochs, train_accuracy, 'b', label='Training accuracy')
plt.plot(epochs, val_accuracy, 'r', label='Validation accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.title('Точность на обучении и валидации')

plt.show()
```

Анализ переобучения

Признаки переобучения:

- Потери на обучении уменьшаются, а на тестировании растут.
- Точность на обучении растет, но на тестировании остается низкой.

Шаги анализа:

1. Постройте графики потерь и точности.
2. Сравните модели с Dropout и без него.
3. Подберите оптимальные гиперпараметры, которые минимизируют переобучение.

Итоговые шаги выполнения задания:

1. Загрузите выделенный набор данных из Kaggle или OpenML.
2. Постройте полносвязную нейронную сеть с индивидуальными параметрами (слои, функции активации).
3. Настройте гиперпараметры: learning rate, batch size, количество эпох.
4. Обучите модель, визуализируйте потери и точность.
5. Включите Dropout и сравните результаты с его использованием и без него.
6. Проведите анализ переобучения и подберите оптимальные

настройки.

Теоретический материал по 3 части

Сравнительный анализ необходим для понимания того, как различные архитектурные решения, функции активации и гиперпараметры влияют на производительность полносвязной нейронной сети. Это позволяет выбрать оптимальные настройки для конкретной задачи и оценить устойчивость модели к переобучению.

На этапе защиты результатов студент должен представить:

- Основные метрики производительности модели.
- Влияние различных параметров на сходимость.
- Выводы по итогам выполнения задания.

Метрики качества нейронной сети

Для анализа качества работы нейронной сети используются следующие основные метрики:

1. Точность (accuracy):

Доля правильно классифицированных примеров представлена в формуле 5:

$$Accuracy = \frac{\text{Количество правильных предсказаний}}{\text{Общее количество примеров}} \quad (5)$$

2. Precision (точность для положительного класса):

Отношение количества правильно предсказанных положительных примеров к количеству всех предсказанных положительных представлено в формуле 6:

$$Precision = \frac{TP}{TP + FP} \quad (6)$$

Где: TP — true positive, FP — false positive.

3. Recall (полнота):

Доля правильно предсказанных положительных примеров относительно

всех фактических положительных представлено в формуле 7:

$$Recall = \frac{TP}{TP + FN} \quad (7)$$

где FN — false negative.

4. F1-мера:

Гармоническое среднее между precision и recall, применяемое в случае несбалансированных данных представлено в формуле 8:

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall} \quad (8)$$

5. Функция потерь (Loss):

Показывает ошибку модели на каждом этапе обучения. Обычно используется кросс-энтропийная функция потерь, которая представлена в формуле 9:

$$Cross\ entropy = -\sum_{i=1}^{N} y_i \log(\hat{y}_i) \quad (9)$$

Влияние функций активации на сходимость

Различные функции активации могут влиять на то, насколько быстро и стабильно сходит нейронная сеть.

1. ReLU (Rectified Linear Unit) (представлено в формуле 10):

$$f(x) = \max(0, x) \quad (10)$$

- Быстро сходится в большинстве задач, особенно для глубоких сетей.
- Однако может страдать от проблемы “мертвых нейронов”, когда часть нейронов перестает обновляться.

2. Sigmoid (представлено в формуле 11):

$$f(x) = \frac{1}{1 + e^{-x}} \quad (11)$$

- Подходит для задач с бинарной классификацией.
- Может вызывать проблему затухающих градиентов, что замедляет обучение.

3. Tanh (представлено в формуле 12):

$$f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \quad (12)$$

- Обеспечивает лучшее центрирование данных (значения от -1 до 1) по сравнению с Sigmoid.
- Также может страдать от затухающих градиентов.

Влияние гиперпараметров на сходимость

Гиперпараметры напрямую влияют на скорость и стабильность обучения. Ниже приводится краткий анализ основных гиперпараметров:

1. Learning rate (скорость обучения):
- Маленький learning rate замедляет сходимость, но делает обучение более стабильным.
- Слишком большой learning rate может привести к неустойчивости и невозможности найти минимум функции потерь.
2. Batch size (размер мини-выборки):
- Маленький размер мини-выборки ускоряет обновление весов, но увеличивает шум в оценках градиента.
- Большой batch size уменьшает шум, но требует больше вычислительных ресурсов.
3. Количество эпох:
- Слишком малое количество эпох может привести к недообучению, а слишком большое — к переобучению.
4. Dropout:
- Dropout уменьшает вероятность переобучения, “выключая” случайные нейроны в процессе обучения.

Сравнение моделей с Dropout и без него

Dropout является одним из эффективных методов борьбы с переобучением. В процессе сравнения необходимо:

- Построить графики потерь и точности для моделей с Dropout и без него.
- Сравнить точность на тестовой выборке.

- Оценить, как Dropout влияет на сходимость и финальные результаты модели.

Визуализация результатов

На этапе защиты важно представить визуальные результаты, чтобы подтвердить сделанные выводы. Основные визуализации:

- Графики потерь: показывают динамику ошибки модели на обучающей и тестовой выборках.
- Графики точности: демонстрируют, как изменяется точность классификации в процессе обучения.
- Визуализация ошибок: примеры неверно классифицированных объектов, чтобы понять, какие данные сеть обрабатывает неправильно.

Пример кода для построения графиков с использованием Matplotlib представлен в листинге 2:

Листинг 2

```
import matplotlib.pyplot as plt
epochs = range(1, len(train_loss) + 1)
plt.figure(figsize=(12, 5))
# График потерь
plt.subplot(1, 2, 1)
plt.plot(epochs, train_loss, 'b', label='Training loss')
plt.plot(epochs, val_loss, 'r', label='Validation loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.title('Потери на обучении и валидации')
# График точности
plt.subplot(1, 2, 2)
plt.plot(epochs, train_accuracy, 'b', label='Training accuracy')
plt.plot(epochs, val_accuracy, 'r', label='Validation accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.title('Точность на обучении и валидации')
plt.show()
```

Подготовка к защите

На этапе защиты студент должен:

1. Представить краткое описание архитектуры модели и набора

данных.

2. Показать основные метрики (точность, F1-мера) для различных параметров сети.
3. Показать сравнительные графики потерь и точности.
4. Обосновать выбор архитектуры, гиперпараметров и методов регуляризации.

Итоговые выводы

На основе сравнительного анализа студент должен сделать выводы о:

- Влиянии функций активации на скорость обучения и точность.
- Влиянии гиперпараметров на сходимость.
- Роли Dropout в предотвращении переобучения.
- Оптимальных настройках модели для конкретного набора данных.