**МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ**

Ордена Трудового Красного Знамени федеральное государственное бюджетное образовательное учреждение высшего образования «Московский технический университет связи и информатики»

Кафедра «Математическая кибернетика и информационные технологии»

**Лабораторная работа № 1**

**тематика**

**Многоклассовая классификация цветов**

**Москва 2021**

**Цель**

Реализовать классификацию сортов растения ирис (Iris Setosa - 0, Iris Versicolour - 1, Iris Virginica - 2) по четырем признакам: размерам пестиков и тычинок его цветков.

**Задачи**

∙ Ознакомиться с задачей классификации

∙ Загрузить данные

∙ Создать модель ИНС

∙ Настроить параметры обучения

∙ Обучить и оценить модель

**Выполнение работы**

Задача многоклассовой классификации является одним из основных видов задач, для решения которых применяются нейронные сети. В листинге 1 представлен пример данных.

Пример данных

| 5.1,3.5,1.4,0.2,Iris-setosa  4.9,3.0,1.4,0.2,Iris-setosa  4.7,3.2,1.3,0.2,Iris-setosa  4.6,3.1,1.5,0.2,Iris-setosa  5.0,3.6,1.4,0.2,Iris-setosa |
| --- |

Получить набор данных от преподавателя . Скачанный файл необходимо переименовать в “iris.csv” и поместить в директорию своего проекта.

Импортируем необходимые для работы классы и функции. Кроме Keras понадобится Pandas для загрузки данных и scikit-learn для подготовки данных и оценки модели.

Подключение модулей

| import pandas  from tensorflow.keras.layers import Dense  from tensorflow.keras.models import Sequential  from tensorflow.keras.utils import to\_categorical  from sklearn.preprocessing import LabelEncoder |
| --- |

Набор данных загружается напрямую с помощью pandas. Затем необходимо разделить атрибуты (столбцы) на входные данные(X) и выходные данные(Y).

Загрузка данных

| dataframe = pandas.read\_csv("iris.csv", header=None) dataset = dataframe.values  X = dataset[:,0:4].astype(float)  Y = dataset[:,4] |
| --- |

При решении задач многоклассовой классификации хорошей практикой является преобразование выходных атрибутов из вектора в матрицу к виду представленных ниже. Листинг 4 - Представление данных

| Iris-setosa, Iris-versicolor, Iris-virginica  1, 0, 0  0, 1, 0  0, 0, 1 |
| --- |

Для этого необходимо использовать функцию to\_categorical()

Переход от текстовых меток к категориальному вектору

| encoder = LabelEncoder()  encoder.fit(Y)  encoded\_Y = encoder.transform(Y)  dummy\_y = to\_categorical(encoded\_Y) |
| --- |

Теперь можно задать базовую архитектуру сети

| model = Sequential()  model.add(Dense(4, activation=’relu’))  model.add(Dense(3, activation=’softmax’)) |
| --- |

Основным строительным блоком нейронных сетей является слой (или уровень), модуль обработки данных, который можно рассматривать как фильтр для данных. Он принимает некоторые данные и выводит их в более полезной форме. В частности, слои извлекают представления из подаваемых в них данных, которые, как мы надеемся, будут иметь больше смысла для решаемой задачи. Фактически методика глубокого обучения заключается в объединении простых слоев, реализующих некоторую форму поэтапной очистки данных. Модель глубокого обучения можно сравнить с ситом, состоящим из последовательности фильтров все более тонкой очистки данных — слоев.

В данном случае наша сеть состоит из последовательности двух слоев Dense, которые являются тесно связанными (их еще называют полносвязными) нейронными слоями. Второй (и последний) слой — это 3-переменный слой потерь (softmax layer), возвращающий массив с 3 оценками вероятностей (в сумме дающих 1). Каждая оценка определяет вероятность принадлежности текущего изображения к одному из 3 классов цветов.

Чтобы подготовить сеть к обучению, нужно настроить еще три параметра для этапа компиляции:

1. функцию потерь, которая определяет, как сеть должна оценивать качество своей работы на обучающих данных и, соответственно, как корректировать ее в правильном направлении;

2. оптимизатор — механизм, с помощью которого сеть будет обновлять себя, опираясь на наблюдаемые данные и функцию потерь;

3. метрики для мониторинга на этапах обучения и тестирования — здесь нас будет интересовать только точность (доля правильно классифицированных изображений).

Инициализация параметров обучения

| model.compile(optimizer='adam',loss='categorical\_crossentropy', metrics=['accuracy']) |
| --- |

Теперь можно начинать обучение сети, для чего в случае использования библиотеки Keras достаточно вызвать метод fit сети — он пытается адаптировать (fit) модель под обучающие данные.

Обучение сети

| model.fit(X, dummy\_y, epochs=75, batch\_size=10,  validation\_split=0.1) |
| --- |

В процессе обучения отображаются четыре величины: потери сети на обучающих данных и точность сети на обучающих данных, а также потери и точность на данных, не участвовавших в обучении.

**Требования**

1. Изучить различные архитектуры ИНС (Разное кол-во слоев, разное кол-во нейронов на слоях)

2. Изучить обучение при различных параметрах обучения (параметры функции fit) 3. Построить графики ошибок и точности в ходе обучения

4. Выбрать наилучшую модель