МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ

по лабораторной работе №1

по дисциплине «Искусственные нейронные сети»

Тема: Многоклассовая классификация цветов

Студентка гр. 7382 _____ Дерябина П.С.

Преподаватель _____ Жукова Н.А.

Санкт-Петербург

2020

Цель работы

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

Порядок выполнения работы

- Ознакомиться с задачей классификации
- Загрузить данные
- Создать модель ИНС в Keras
- Настроить параметры обучения
- Обучить и оценить модель

Требования к выполнению задания

- Изучить различные архитектуры ИНС (Разное кол-во слоев, разное кол-во нейронов на слоях)
- Изучить обучение при различных параметрах обучения (параметры функций fit)
- Построить графики ошибок и точности в ходе обучения
- Выбрать наилучшую модель

Основные теоретические положения

Задача многоклассовой классификации является одним из основных видов задач, для решения которых применяются нейронные сети. В данной работе использовали классический датасет iris.

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.

Чтобы подготовить сеть к обучению, нужно настроить 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).

Ход работы

Код программы, реализующий структуру ИНС, приведен в приложении А.

Посмотрим на графики функции потерь и точности модели в зависимости от эпохи для начальных данных на рис. 1 и рис. 2 соответственно.

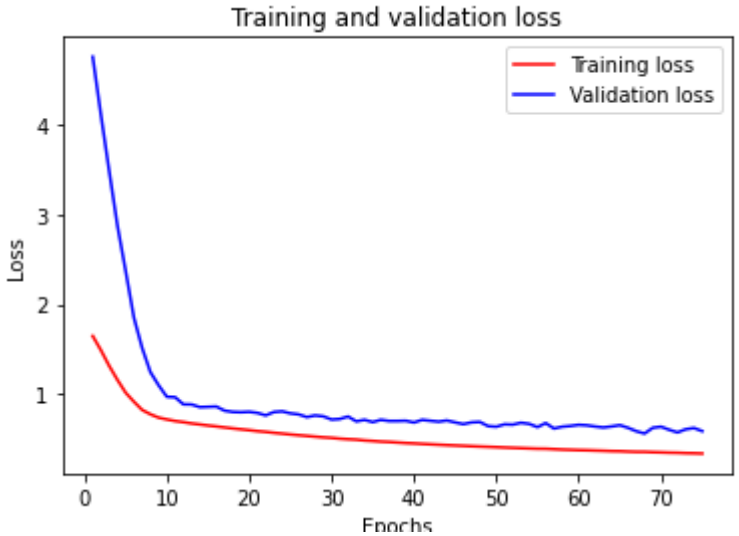

Рисунок 1 – Начальные параметры, график функции потерь

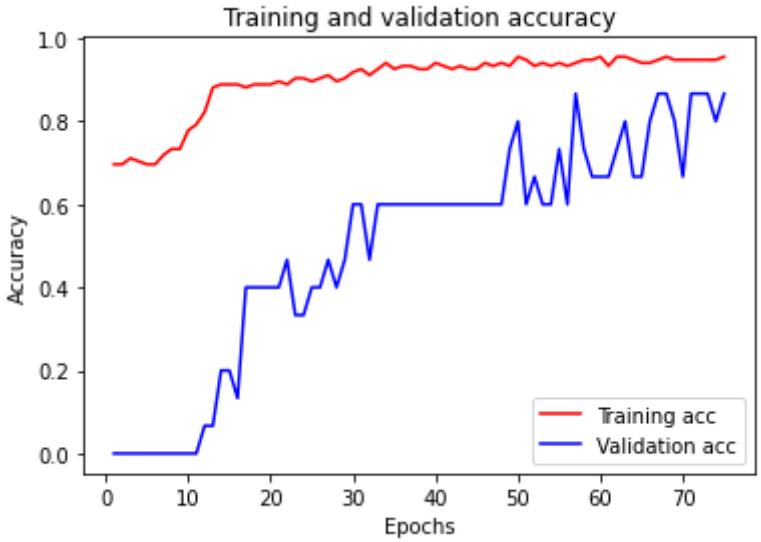

Рисунок 2 – Начальные параметры, график точности модели

Из рис. 1 видим, что происходит переобучение на 30 эпохе, из рис. 2 видим недостаточную точность.

Добавим еще один слой из четырех полносвязных нейронов и посмотрим, что получилось на рис. 3 и рис. 4.

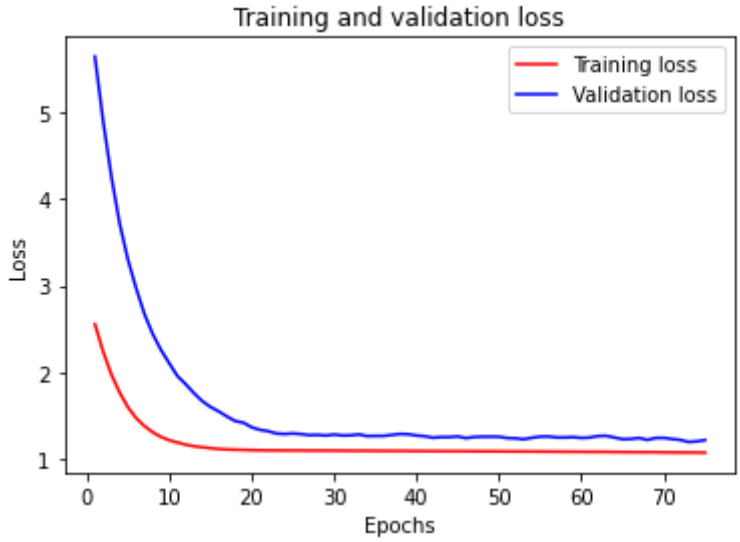

Рисунок 3 – Добавили слой, график функции потерь

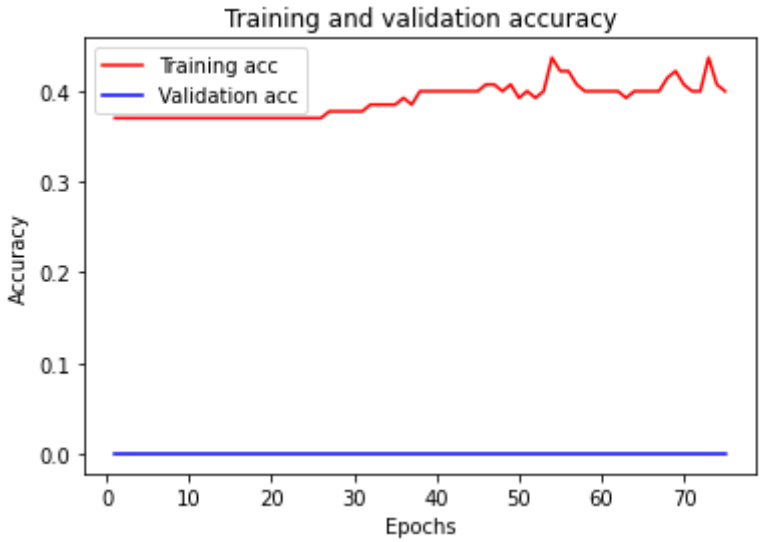

Рисунок 4 – Добавили слой, график точности модели

Видим, что особо лучше не стало, поэтому стратегию добавления слоев отложим - попробуем увеличть количетсво нейронов в первом слое с 4 до 16 (см. рис 5 и рис. 6).

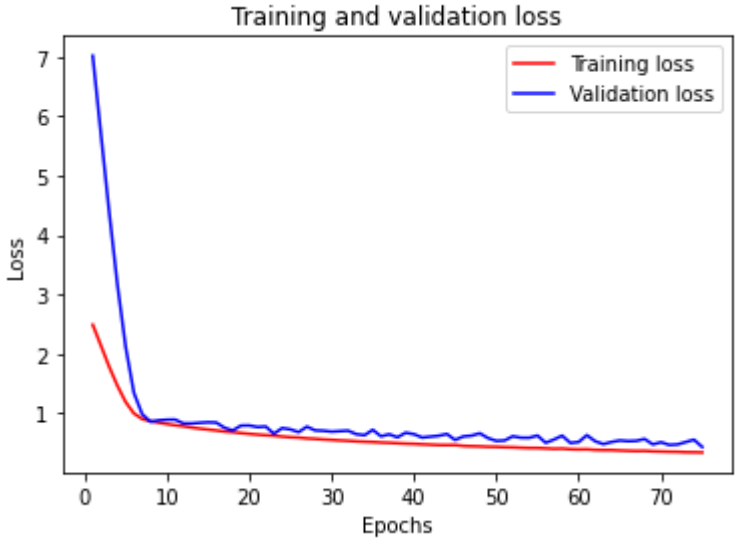

Рисунок 5 – Добавили нейронов, график функции потерь

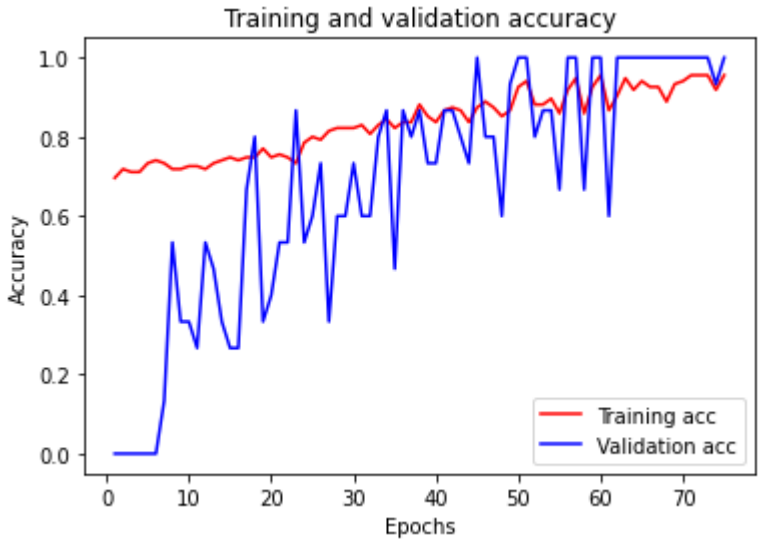

Рисунок 6 – Добавили нейронов, график точности

Из рис. 5-6 видим улучшение по обоим показателям – функции потерь и точности модели, поэтому оставим 16 нейронов в первом слое и увеличим количество эпох до 500 (см. рис. 7-8).

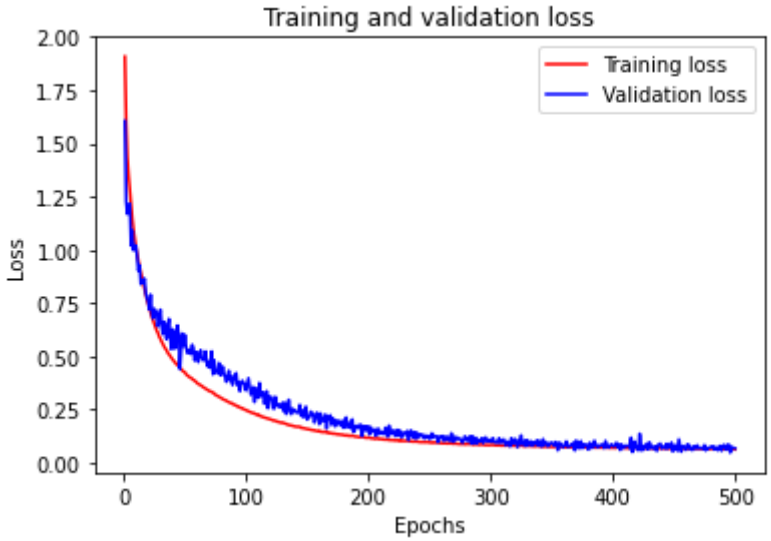

Рисунок 7– Добавили эпох, график функции потерь

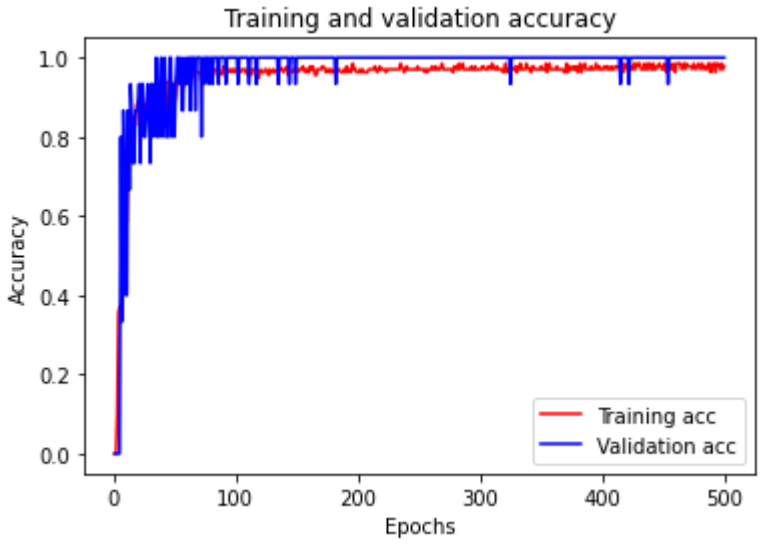

Рисунок 8 – Добавили эпох, график точности модели

Видим, что добились существенного улучшения. Последующее увеличение количества нейронов не дает особого улучшения, поэтому модель с 16 нейронами в первом слое типа relu, 3 нейронами типа softmax в последнем слое и 500 эпохами будем считать лучшей.

Вывод

Изучили основы построения нейронных сетей. Проверили различные архитектуры нейронных сетей на классическом датасете ирисов, выбрали наилучшую модель, основываясь на таких показателях, как функция потерь и точность модели.

ПРИЛОЖЕНИ А
ИСХОДНЫЙ КОД

```
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
import matplotlib.pyplot as plt

#1
dataframe = pandas.read_csv('./iris.data', header=None)
dataset = dataframe.values
X = dataset[:,0:4].astype(float)
Y = dataset[:,4]

#2
encoder = LabelEncoder()
encoder.fit(Y)
encoded_Y = encoder.transform(Y)
dummy_y = to_categorical(encoded_Y)

#3
model = Sequential()
model.add(Dense(16, activation='relu'))
model.add(Dense(3, activation='softmax'))
#4
model.compile(optimizer='adam',loss='categorical_crossentropy', metrics=['ac
curacy'])

#5
h = model.fit(X, dummy_y, epochs=500, verbose=0, batch_size=10, validation_s
plit=0.1)
history = h.history

# Получение ошибки и точности в процессе обучения
loss = history['loss']
val_loss = history['val_loss']
acc = history['accuracy']
val_acc = history['val_accuracy']
epochs = range(1, len(loss) + 1)

# Построение графика ошибки
plt.plot(epochs, loss, 'r', label='Training loss')
plt.plot(epochs, val_loss, 'b', label='Validation loss')
plt.title('Training and validation loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()
```

```
# Построение графика точности
#plt.clf()
plt.plot(epochs, acc, 'r', label='Training acc')
plt.plot(epochs, val_acc, 'b', label='Validation acc')
plt.title('Training and validation accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```