

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

ОТЧЕТ
по лабораторной работе №1
по дисциплине «Искусственные нейронные сети»
Тема: Многоклассовая классификация цветов

Студентка гр. 7382

Еременко А.

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.

1. Ознакомиться с задачей классификации;
2. Загрузить данные;
3. Создать модель ИНС в Keras;
4. Настроить параметры обучения;
5. Обучить и оценить модель.

Требования к выполнению задания.

1. Изучить различные архитектуры ИНС (Разное количество слоев, разное количество нейронов на слоях);
2. Изучить обучение при различных параметрах обучения (параметры функций fit);
3. Построить графики ошибок и точности в ходе обучения;
4. Выбрать наилучшую модель.

Основные теоретические положения.

Задача классификации – задача, в которой имеется множество объектов (ситуаций), разделённых некоторым образом на классы. Задано конечное множество объектов, для которых известно, к каким классам они относятся. Это множество называется выборкой. Классовая принадлежность остальных объектов неизвестна.

Требуется построить алгоритм, способный классифицировать произвольный объект из исходного множества.

Классифицировать объект – значит, указать номер (или наименование) класса, к которому относится данный объект.

Классификация объекта – номер или наименование класса, выдаваемый алгоритмом классификации в результате его применения к данному конкретному объекту.

Функция потерь (Loss function) – измеряет точность модели во время обучения. Мы хотим минимизировать эту функцию чтоб "направить" модель в верном направлении.

Оптимизатор (Optimizer) – показывает каким образом обновляется модель на основе входных данных и функции потерь.

Метрики (Metrics) – используются для мониторинга тренировки и тестирования модели. Наш пример использует метрику ассигасу равную доле правильно классифицированных изображений.

Ход работы.

Для изучения различной структуры ИНС была разработана и использована программа из приложения А.

При изначальных данных график точности модели предоставлен на рис. 1.

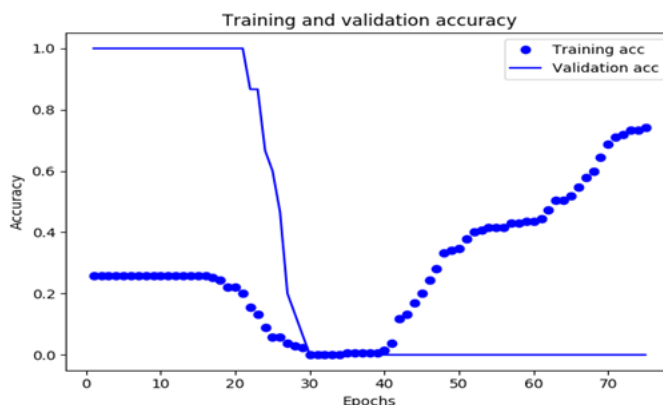


Рисунок 1 – Точность модели при изначальных данных

Посмотрим, как будет вести себя модель при большом количестве эпох. Изменим количество проходов с 75 до 4000, точность и потери модели изменились. Результат предоставлен на рис. 2 и 3 соответственно.

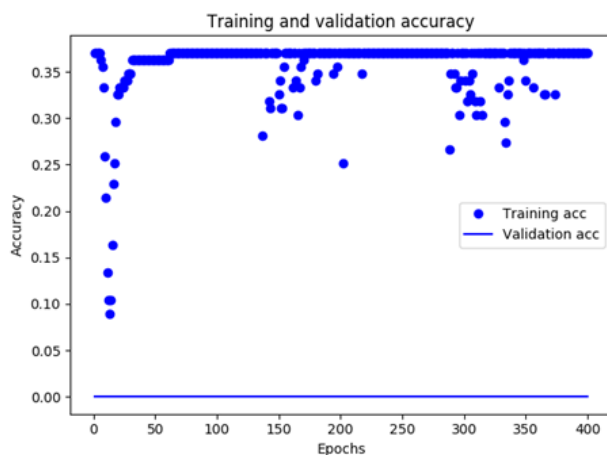


Рисунок 2 – Точность модели при 4000 эпох

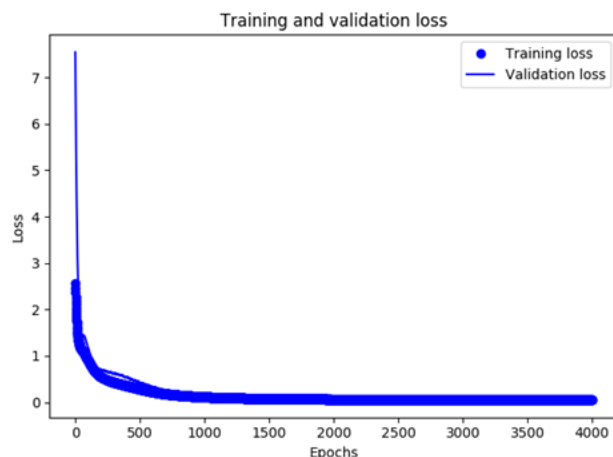


Рисунок 3 – График потерь при 4000 эпох

Уменьшим количество эпох до 400 и увеличим количество нейронов в 1 слое до 16. Как видно из рис. 4 и 5, модель смогла достичь максимальной точности примерно за 50 поколений, а потери были минимизированы за 100 поколений. При увеличении количество слоев результат не становился лучше, поэтому дальнейшее обучение смысла не имеет. Данный результат является наилучшим (см. рис. 4 и 5).

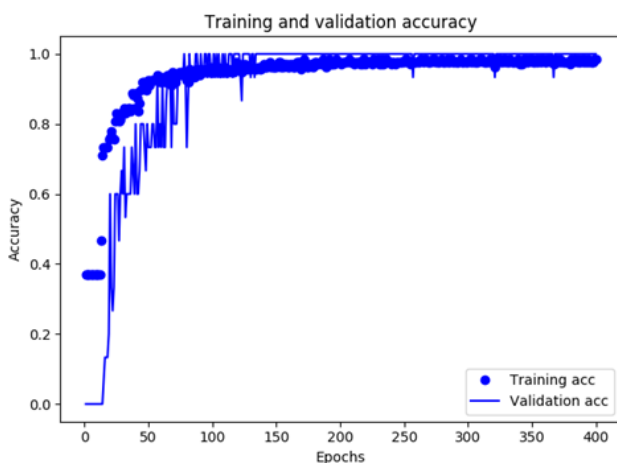


Рисунок 4 – Точность модели при 400 эпохах

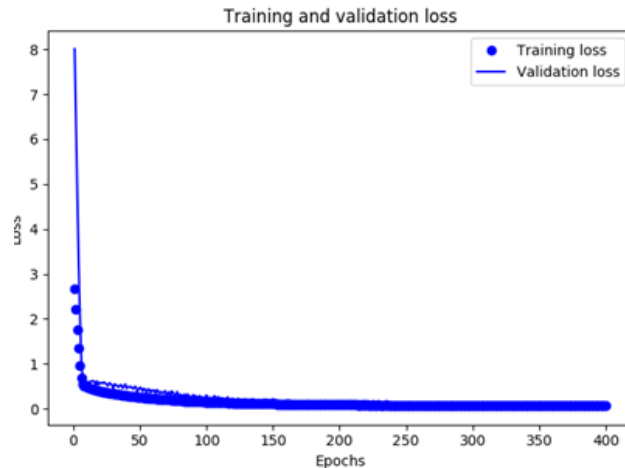


Рисунок 5 – Точность модели при 400 эпохах

Выводы.

Были изучены основы работы с искусственными нейронными сетями. Наблюдали за тем, как меняется результат в ходе изменений различных параметров, как в ходе построения модели, так и в ходе ее обучения.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

```
import pandas
from keras.layers import Dense
from keras.models import Sequential
from keras.utils import to_categorical
from sklearn.preprocessing import LabelEncoder
import matplotlib.pyplot as plt

dataframe = pandas.read_csv("iris.csv", header=None)
dataset = dataframe.values
X = dataset[:,0:4].astype(float)
Y = dataset[:,4]

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

model = Sequential()
model.add(Dense(4, activation='relu'))
model.add(Dense(3, activation='softmax'))

model.compile(optimizer='adam', loss='categorical_crossentropy',
metrics=['accuracy'])

H = model.fit(X, dummy_y, epochs=1000, batch_size=10,
validation_split=0.1)

loss = H.history['loss']
val_loss = H.history['val_loss']
acc = H.history['accuracy']
val_acc = H.history['val_accuracy']
epochs = range(1, len(loss) + 1)

# Построение графика ошибки

plt.plot(epochs, loss, 'bo', 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, 'bo', 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()
```