DOI 10.37539/2949-1991.2023.4.4.034
УДК 004.855.5

Половников Владислав Олегович, студент,
Сахалинский государственный университет, Южно-Сахалинск
Polovnikov Vladislav Olegovich, Sakhalin State University,
Yuzhno-Sakhalinsk

Научный руководитель:
Осипов Геннадий Сергеевич, д.т.н., Зав. Кафедрой Информатики,
Сахалинский государственный университет, Южно-Сахалинск
Osipov Gennady Sergeevich, Sakhalin State University, Yuzhno-Sakhalinsk

МЕТОДОЛОГИЯ СИНТЕЗА НЕЙРОСЕТЕВЫХ КЛАССИФИКАТОРОВ НА ЯЗЫКЕ PYTHON
METHODOLOGY OF SYNTHESIS OF NEURAL NETWORK CLASSIFIERS IN PYTHON

Аннотация: Проведено исследование и предложена основополагающая методология синтеза нецифровых классификаторов с помощью многослойной искусственной нейронной сети. Предложена реализация проекта на базе современного языка решения задач искусственного интеллекта Python.

Abstract: A study has been conducted and a fundamental methodology for the synthesis of non-digital classifiers using a multilayer artificial neural network has been proposed. The implementation of the project based on the modern language for solving artificial intelligence problems Python is proposed.

Ключевые слова: машинное обучение, классификации объектов, нейронная сеть.

Keywords: machine learning, object classification, neural network.

Введение

В современном мире стремительно растет тенденция использования нейросети для решения задач искусственного интеллекта. Нейросети, основанные на принципе искусственного интеллекта, позволяют имитировать работу человеческого мозга, обрабатывать и анализировать большие объемы данных, распознавать предметы на картинках, предсказывать тренды и другие сложные задачи.

Для разработки нейросетей часто используют такой язык программирования как Python [1], так как он прост в изучении, а также имеет понятный и лаконичный синтаксис. Python имеет большое количество библиотек и документации для работы с нейросетью, а так же обширное сообщество разработчиков, которые могут помочь в решении различных задач.

Целью настоящего исследования является отработка методологии нейросетевого моделирования для решения задачи синтеза классификаторов объектов, характеризуемых показателями различных типов

Формальная постановка задачи

Объектом и предметом исследования является проблема построения оптимального классификатора вида f: $\boldsymbol{x}\rightarrow y$,

где $\boldsymbol{x}=(x_1, x_2, \ldots, x_n)$ – вектор числовых, символьных и лингвистических характеристик объекта;

y – лингвистический идентификатор класса к которому относится объект.

На основании экспертных заключений известна базовая классификация вида:

$$f^*: \boldsymbol{x}\rightarrow y.$$

Количество информации, содержащейся в классификаторе, обозначим через $I(f)$. Требуется построить нейросетевой классификатор, который обеспечивает минимальную потерю полезной информации, т.е.:

$$||I(f)-I(f^*)||\rightarrow \min.$$

Термин «количество информации» в классификаторе определим как процент правильно классифицированных им объектов достигнутом за период обучения модели на обучающей выборке (data set). Положим $I(f^*)=100\%$.

Исходные данные

Для отработки методологии синтеза оптимального нейросетевого классификатора используем базу данных об экспертных оценках автомобилей [2], которые представлены в таблице 1.

Таблица 1

Фрагмент исходных данных

x_1	x_2	x_3	x_4	x_5	x_6	y
низкая	очень высокая	>=5	4	малый	высокая	удовл
низкая	очень высокая	>=5	4	средний	низкая	неуд
низкая	очень высокая	>=5	4	средний	средняя	удовл
низкая	очень высокая	>=5	4	средний	высокая	удовл
низкая	очень высокая	>=5	4	большой	низкая	неуд
низкая	очень высокая	>=5	4	большой	средняя	удовл
…	…	…	…	…	…	…

Множество возможных значений характеристик объектов и идентификаторов классов представлены в таблице 2.

Таблица 2

Множество значений переменных

x_1	x_2	x_3	x_4	x_5	x_6	y
очень высокая	очень высокая	2	2	малый	низкая	неуд
высокая	высокая	3	4	средний	средняя	удовл
средняя	средняя	4	>=5	большой	высокая	хор
низкая	низкая	>=5				отл

Таким образом $\boldsymbol{x}=(x_1, x_2, \ldots, x_6)$.

Модель нейросетевого классификатора

Структура нейросетевого классификатора выбрана с целью исследования влияния числа скрытых слоев и различных видов функции активации на качество решения задачи [3, 4].

На вход подается 6 переменных.

На выходе 4 значения, при этом выбирается максимальное значение, что и будет результатом вычисления нейросети.

Структура нейросети:

- 1 скрытый слой: 128 нейронов, способ активации: ReLU (rectified linear unit), для лучшей обучаемости и избегания “скачков” выходных результатов случайным образом присвоим нулевые значения 20% нейронам.
- 2 скрытый слой: 64 нейронов, способ активации: ReLU (rectified linear unit)
- 3 скрытый слой: 32 нейронов, способ активации: Sigmoid
- 4 скрытый слой: 16 нейронов, способ активации: ReLU (rectified linear unit)
- 5 скрытый слой: 8 нейронов, способ активации: ReLU (rectified linear unit)
- 6 слой: 4 выходных нейрона, над которыми мы используем функцию Softmax, для выбора максимального значения

Основные результаты

На рисунке 1 представлены операторы подключения библиотек, используемых для разработки приложений.

```
# Подключаем библиотеки
import pandas as pd
from sklearn.preprocessing import LabelEncoder
from tensorflow import keras
import tensorflow as tf
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
```

Рис. 1 Подключение библиотек

Обучающая выборка (data set) хранится в виде массива (см. таблицу 1) в файле Excel. последовательность ввода исходных данных приведена на рисунке 2.

```
# Считываем файл
data = pd.read_excel('cars.xlsx')

# Присваиваем переменным столбцы из файла
X = data.iloc[:, :6]
y = data.iloc[:, 6]

# Преобразовываем столбцы в массивы
X = X.values
y = y.values
```

Рис. 2 Ввод исходных данных

Для обеспечения функционирования нейросети – ее обучения, тестирования и использования для классификации все элементы массива исходных данных получают соответствующие метки (см. рисунок 3).

```
# Выдаем каждому элементу массива соответствующую ему метку
label_encoder = LabelEncoder()
X_encoded = label_encoder.fit_transform(X.flatten()).reshape(X.shape)
y_encoded = label_encoder.fit_transform(y)
```

Рис.3 Присвоение меток данным

Реализация структуры нейронной сети, используемой в качестве классификатора приведена на рисунке 4.

```
# Разделяем данные для обучения и для теста (80% для обучения, 20% для теста)
X_train, X_test, y_train, y_test = train_test_split(X_encoded, y_encoded,
test_size=0.2)

# Создаем структуру нейросети (6 скрытых слоев)
model = keras.models.Sequential()
model.add(keras.layers.Dense(128, input_dim=6, activation='ReLU'))
model.add(keras.layers.Dropout(0.2))
model.add(keras.layers.Dense(64, activation='ReLU'))
model.add(keras.layers.Dense(32, activation='sigmoid'))
model.add(keras.layers.Dense(16, activation='ReLU'))
model.add(keras.layers.Dense(8, activation='ReLU'))
model.add(keras.layers.Dense(4, activation='softmax'))
```

Рис. 4 Задание структуры нейронной сети

Здесь же осуществляется разбиение обучающей выборки на два множества – обучающее и тестовое.

Основные операторы настройки и выполнения процесса обучения нейросетевого классификатора приведены на рисунке 5. Контроль качества обучения осуществляется подсчетом точности обучения на тестовом множестве.

```
# Задаем настройки для обучения нейросети
model.compile(loss='sparse_categorical_crossentropy', optimizer='adam',
metrics=['accuracy'])

# Создаем callback-переменную для записи истории обучения
history = tf.keras.callbacks.History()

# Запускаем обучение
model.fit(X_train, y_train, epochs=350, batch_size=16, callbacks=history)

# Выводим сколько процентов точности у модели
_, accuracy = model.evaluate(X_test, y_test)
print('Accuracy: %.2f' % (accuracy * 100))


# Получаем историю изменений потерь и точности у нейросети
loss = history.history['loss']
accuracy = history.history['accuracy']
```

Рис. 5 Организация обучения нейронной сети

Настройка параметров визуализации процесса синтеза классификатора приведена на рисунке 6.

```
# Созданием объект окна для графиков
fig, (ax1, ax2) = plt.subplots(2, 1, figsize=(8, 6))

# График изменения потерь
ax1.plot(loss, label='Loss')
ax1.set_xlabel('Epochs')
ax1.set_ylabel('Loss')
ax1.legend()

# График изменения точности
ax2.plot(accuracy, label='Accuracy')
ax2.set_xlabel('Epochs')
ax2.set_ylabel('Accuracy')
ax2.legend()

# Выводим окно с графиками
plt.tight_layout()
plt.show()
```

Рис. 6 Формирование графиков отчета о процессе обучения

Графики потерь и точности обучения приведены на рисунке 7.

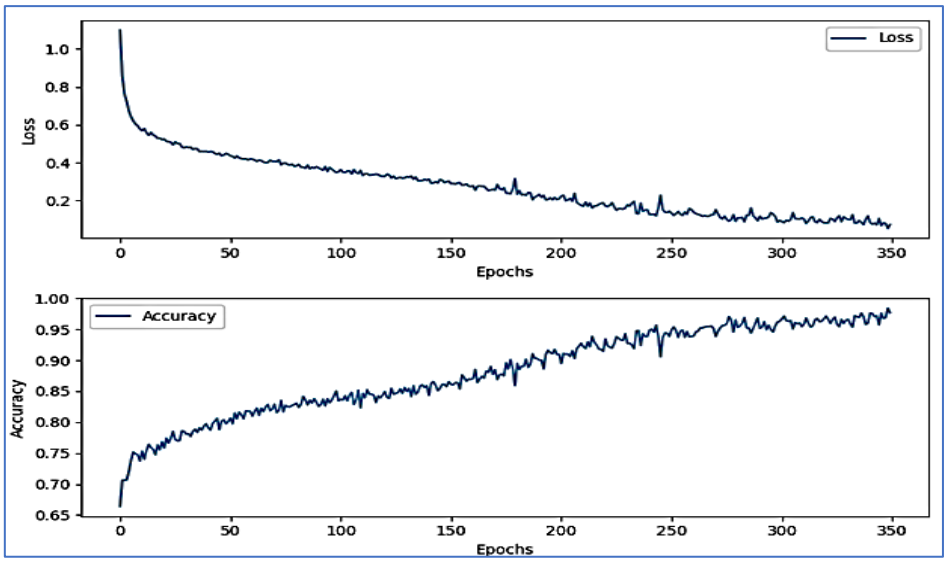

Рис. 7 Графики кривых обучения

Точность классификации на тестовом множестве составляет 92-99%

Выводы и заключение

Результатом исследования является отработка базовой методологии решения задачи построения интеллектуального классификатора объектов с нецифровыми характеристиками с помощью нейросетевой модели обучения.

Разработано программное обеспечения задачи, реализованное на языке Python. Произведена практическая апробация предложенной методологии на экспертной системе априорной классификации автомобилей. За счет выбора оптимальной структуры нейронной сети достигнута удовлетворительная точность классификации.

Список литературы:

1. Keras. Simple. Flexible. Powerful. https://keras.io/ (дата обращения 27.05.2023).

2. Car Evaluation Data Set. URL: https://archive.ics.uci.edu/ml/datasets/car+evaluation (дата обращения 30.05.2023).

3. Нейронная сеть с использованием TensorFlow: классификация изображений. https://habr.com/ru/articles/426797/, (дата обращения 30.05.2023).

4. API документация библиотеки TenserFlow. https://www.tensorflow.org/api_docs, (дата обращения 30.05.2023).