Министерство образования Республики Беларусь

Учреждение образования

БелорусскиЙ государственный университет

информатики и радиоэлектроники

Факультет компьютерных систем и сетей

Кафедра информатики

|  |
| --- |
|  |

**ЛабоРАТОРНАЯ РАБОТА №2**

«Реализация глубокой нейронной сети»

|  |  |  |
| --- | --- | --- |
| Студент |  | К. Д. Зюсько |
| Преподаватель |  | М. В. Стержанов |

Минск 2020

ХОД РАБОТЫ

**Задание.**

В работе предлагается использовать набор данных notMNIST, который состоит из изображений размерностью 28×28 первых 10 букв латинского алфавита (A … J, соответственно). Обучающая выборка содержит порядка 500 тыс. изображений, а тестовая – около 19 тыс.

Данные можно скачать по ссылке:

* https://commondatastorage.googleapis.com/books1000/notMNIST\_large.tar.gz (большой набор данных);
* https://commondatastorage.googleapis.com/books1000/notMNIST\_small.tar.gz (маленький набор данных).

Описание данных на английском языке доступно по ссылке:

http://yaroslavvb.blogspot.sg/2011/09/notmnist-dataset.html

1. Реализуйте полносвязную нейронную сеть с помощью библиотеки TensorFlow. В качестве алгоритма оптимизации можно использовать, например, стохастический градиент (Stochastic Gradient Descent, SGD). Определите количество скрытых слоев от 1 до 5, количество нейронов в каждом из слоев до нескольких сотен, а также их функции активации (кусочно-линейная, сигмоидная, гиперболический тангенс и т.д.).
2. Как улучшилась точность классификатора по сравнению с логистической регрессией?
3. Используйте регуляризацию и метод сброса нейронов (dropout) для борьбы с переобучением. Как улучшилось качество классификации?
4. Воспользуйтесь динамически изменяемой скоростью обучения (learning rate). Наилучшая точность, достигнутая с помощью данной модели составляет 97.1%. Какую точность демонстрирует Ваша реализованная модель?

**Результат выполнения:**

1. Код реализации:

# definition

model = Sequential()

# adding layers

model.add(Dense(200, kernel\_initializer='normal', activation='relu', name='hidden\_0'))

model.add(Dense(170, kernel\_initializer='normal', activation='relu', name='hidden\_1'))

model.add(Dense(130, kernel\_initializer='normal', activation='relu', name='hidden\_2'))

model.add(Dense(100, kernel\_initializer='normal', activation='relu', name='hidden\_3'))

model.add(Dense(10, kernel\_initializer='normal', activation='relu', name='hidden\_4'))

model.compile(loss='mse', optimizer='sgd', metrics=['accuracy'])

# train

model.fit(X\_train, np.array(pd.get\_dummies(y\_train)),

epochs=60,

verbose=1,

validation\_data=(X\_test, np.array(pd.get\_dummies(y\_test))))

В данном пункте предлагалось использовать в качестве фреймворка для машинного обучения TensorFlow, однако я выбрал Keras, который в качестве бекенда использует TensorFlow. Данный подход помогает сосредоточится только на написании модели нейронной сети и опустить некоторые рутинные операции, которые присущи TensorFlow, такие как создание сессии и пр.

2. В случае логистической регрессии точность составляла 88.67%. В случае нейронной сети (код реализации которой приведён выше) точность составила 93.09%.

3. Код реализации:

# definition

model = Sequential()

# adding layers

model.add(Dense(200, kernel\_initializer='normal', activation='relu', kernel\_regularizer=regularizers.l2(0.001), name='hidden\_0'))

model.add(Dropout(0.1, name='dropout\_0'))

model.add(Dense(170, kernel\_initializer='normal', activation='relu', kernel\_regularizer=regularizers.l2(0.001), name='hidden\_1'))

model.add(Dropout(0.1, name='dropout\_1'))

model.add(Dense(130, kernel\_initializer='normal', activation='relu', kernel\_regularizer=regularizers.l2(0.001), name='hidden\_2'))

model.add(Dropout(0.1, name='dropout\_2'))

model.add(Dense(100, kernel\_initializer='normal', activation='relu', kernel\_regularizer=regularizers.l2(0.001), name='hidden\_3'))

model.add(Dropout(0.1, name='dropout\_3'))

model.add(Dense(10, kernel\_initializer='normal', activation='relu', name='hidden\_4'))

model.compile(loss='mse', optimizer='sgd', metrics=['accuracy'])

# train

model.fit(X\_train, np.array(pd.get\_dummies(y\_train)),

epochs=60,

verbose=1,

validation\_data=(X\_test, np.array(pd.get\_dummies(y\_test))))

С данной конфигурацией нейронной сети я не столкнулся с проблемой переобучения, поэтому при добавлении методов борьбы с переобучением, точность модели, как и ожидаемо, немного ухудшилась и составила 90.47%.

4. Код реализации:

sgd = optimizers.SGD(lr=0.1, decay=1e-4, momentum=0.9, nesterov=True)

# compile model

model.compile(loss='mse', optimizer=sgd, metrics=['accuracy'])

Точность модели составила 92.42%.

**Программный код:**

from keras.models import Sequential

from keras.layers import Dense

from keras import regularizers

from keras.layers import Dropout

from keras import optimizers

import pandas as pd

import numpy as np

import tensorflow as tf

import warnings

warnings.filterwarnings("ignore", category=FutureWarning)

import sys, os

sys.path.append(os.path.abspath(os.path.join('..', 'loaders')))

from load\_not\_mnist import load

X\_train, y\_train, X\_test, y\_test = load()

with tf.device('/cpu:0'):

# definition

model = Sequential()

# adding layers

model.add(Dense(200, kernel\_initializer='normal', activation='relu', kernel\_regularizer=regularizers.l2(0.001), name='hidden\_0'))

model.add(Dropout(0.1, name='dropout\_0'))

model.add(Dense(170, kernel\_initializer='normal', activation='relu', kernel\_regularizer=regularizers.l2(0.001), name='hidden\_1'))

model.add(Dropout(0.1, name='dropout\_1'))

model.add(Dense(130, kernel\_initializer='normal', activation='relu', kernel\_regularizer=regularizers.l2(0.001), name='hidden\_2'))

model.add(Dropout(0.1, name='dropout\_2'))

model.add(Dense(100, kernel\_initializer='normal', activation='relu', kernel\_regularizer=regularizers.l2(0.001), name='hidden\_3'))

model.add(Dropout(0.1, name='dropout\_3'))

model.add(Dense(10, kernel\_initializer='normal', activation='relu', name='hidden\_4'))

sgd = optimizers.SGD(lr=0.1, decay=1e-4, momentum=0.9, nesterov=True)

# compile model

model.compile(loss='mse', optimizer=sgd, metrics=['accuracy'])

# train

model.fit(X\_train, np.array(pd.get\_dummies(y\_train)),

epochs=60,

verbose=1,

validation\_data=(X\_test, np.array(pd.get\_dummies(y\_test))))

score, acc = model.evaluate(X\_test, np.array(pd.get\_dummies(y\_test)), verbose=1)

print(score, acc)