Учреждение образования

«Белорусский государственный университет

информатики и радиоэлектроники»

Кафедра информатики

Отчет по лабораторной работе #2

«Логистическая регрессия в качестве нейронной сети»

Карп Александр Игоревич

магистрант кафедры информатики

группа №858641

Минск 2019

**Задание 1.**

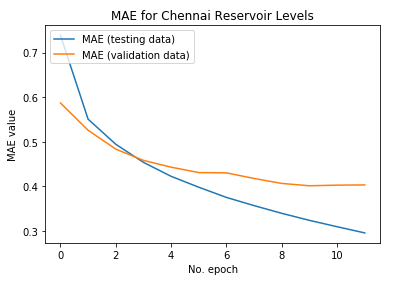
Реализуйте полносвязную нейронную сеть с помощью библиотеки Tensor Flow. В качестве алгоритма оптимизации можно использовать, например, стохастический градиент (Stochastic Gradient Descent, SGD). Определите количество скрытых слоев от 1 до 5, количество нейронов в каждом из слоев до нескольких сотен, а также их функции активации (кусочно-линейная, сигмоидная, гиперболический тангенс и т.д.).

Определим сеть с 3 скрытыми слоями с 100, 50 и 30 нейронами соотвественно. Входной слой будет иметь 300 нейронов, а выходной – по количеству классов, 10. В качестве функций активаций на всех слоях, кроме выходного, будет использоваться ReLU, а на выходном – softmax. Функция потерь будет считать по формуле

, где

В качестве алгоритма оптимизации используется стохастический градиентный спуск.

model = Sequential()  
model.add(Dense(units=300, activation='relu', input\_dim=784))  
model.add(Dense(units=100, activation='relu'))  
model.add(Dense(units=50, activation='relu'))  
model.add(Dense(units=30, activation='relu'))  
model.add(Dense(units=10, activation='softmax'))  
model.compile(loss='categorical\_crossentropy',  
 optimizer='sgd',  
 metrics=['accuracy'])  
  
history = model.fit(X, np\_utils.to\_categorical(y), batch\_size=32, epochs=12, validation\_split=0.2)  
y\_pred = np.argmax(model.predict(np.array(X\_test)), axis=1)  
  
print(accuracy\_score(y\_pred, y\_test))

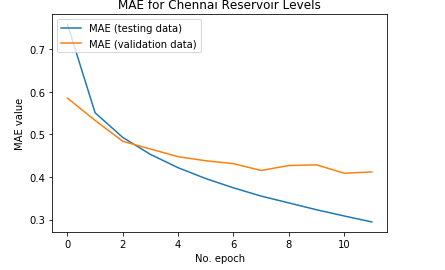


На графике можно заметить результаты переобучения сети – ошибка на тренировочных данных с какого-то момента продолжает уменьшаться, а на валидационных данных не меняется

**Задание 2.**

Как улучшилась точность классификатора по сравнению с логистической регрессией?

Точность составила 88% по сравнению с 82% в логистической регрессии.



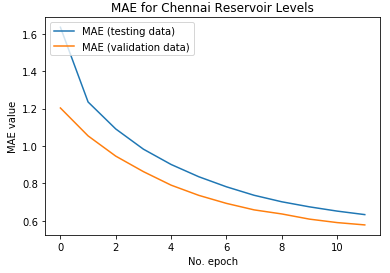
**Задание 3.**

Используйте регуляризацию и метод сброса нейронов (dropout) для борьбы с переобучением. Как улучшилось качество классификации?

from keras.models import Sequential  
from keras.layers import Dense  
from keras.layers import Dropout  
from keras import regularizers  
import numpy as np  
from keras.utils import np\_utils  
model = Sequential()  
model.add(Dense(units=300, activation='relu', input\_dim=784, kernel\_regularizer=regularizers.l2(0.001)))  
model.add(Dropout(0.2))  
model.add(Dense(units=100, activation='relu', kernel\_regularizer=regularizers.l2(0.001)))  
model.add(Dropout(0.2))  
model.add(Dense(units=50, activation='relu', kernel\_regularizer=regularizers.l2(0.001)))  
model.add(Dropout(0.2))  
model.add(Dense(units=30, activation='relu', kernel\_regularizer=regularizers.l2(0.001)))  
model.add(Dropout(0.2))  
model.add(Dense(units=10, activation='softmax'))  
model.compile(loss='categorical\_crossentropy',  
 optimizer='sgd',  
 metrics=['accuracy'])  
  
model.fit(X, np\_utils.to\_categorical(y), batch\_size=32, epochs=50)

Добавление регуляризации и дропаута уменьшило точность на тренировочном сете, но при этом увеличило на тестовом – 89%

Следовательно, данные изменения позволили избежать переобучения нейронной сети:



**Задание 4.**

Воспользуйтесь динамически изменяемой скоростью обучения (learning rate). Наилучшая точность, достигнутая с помощью данной модели составляет 97.1%. Какую точность демонстрирует Ваша реализованная модель?

Воспользуемся конструктором SGD из библиотеки keras для добавления шага, который будет уменьшать learning rate в зависимости от эпохи обучения:

opt = SGD(lr=1e-2, momentum=0.9, decay=1e-2/epochs)

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

Наилучший результат, который удалось достичь: 90.87%

Выводы

С помощью полносвязной нейронной сети удалось достичь большей точности по сравнению с логистической регрессией.