Министерство образования Республики Беларусь

Учреждение образования
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет Компьютерных Систем и Сетей

Кафедра Информатики

ОТЧЁТ

по лабораторной работе №3 по теме

Применение программных средств машинного обучения для обучения модели нейронной сети

Магистрант: Долматович А.С.

МИНСК 2019

Цель: обучить выбранную модель нейронной сети с помощью программных средств машинного обучения

Набор для обучения содержит 5000 изображений 20x20 в оттенках серого. Каждый пиксель представляет собой значение яркости (вещественное число). Каждое изображение сохранено в виде вектора из 400 элементов. Далее расположены метки классов изображений от 1 до 9 (соответствуют цифрам от 1 до 9), а также 10 соответствует цифре 0).

Топология модели (Персептрон)

- Вход - 400
- Скрытый слой - 25
- Выход - 10

Демонстрационные примеры реализованных моделей и результатов их работы:

Инициализируем веса небольшими случайными числами

```
def generateWeights(shape):
    weights = []
    for i in range(shape[0]):
        line = []
        for j in range(shape[1]):
            value = random.random()
            line.append(value)
        weights.append(line)
    return np.array(weights)

```

Реализуем функцию распространения с сигмоидом в качестве функции активации

```
def sigmoid(x):
    return 1 / (1 + np.exp(-x))

def getSigmoidData(ones, x, theta):
    if x.shape[1] != theta.shape[1]:
        x = np.hstack((ones, x))
    z = np.dot(x, theta.T)
    return sigmoid(z)

def h(theta1, theta2, x):
    m = len(x)
    ones = np.ones((m, 1))
    x = getSigmoidData(ones, x, theta1)
    return x, getSigmoidData(ones, x, theta2)
```

Реализуем функцию фычисления производной для функции активации

```
def sigmoidDerivate(x):
    return np.exp(-x) / ((1 + np.exp(-x)) ** 2)

```

Реализуем алгоритм обратного распространения ошибки

```
for j in xrange(100):
    l0 = x
    l1, l2 = h(weights1, weights2, l0)
    l2error = y - l2
    if (j% 100) == 0:
        print "Error:" + str(np.mean(np.abs(l2error)))
    l2delta = l2error*sigmoidDerivate(l2)
    l1error = l2delta.dot(weights2)
    l1delta = l1error * sigmoidDerivate(l1)
    weights1 += np.dot(l0.T, l1delta).T
    weights2 += np.dot(l1.T, l2delta).T

```

Значение параметра weights1, weights2:

```
[[-2.25623899e-02 -1.05624163e-08  2.19414684e-09 ... -1.30529929e-05
  -5.04175101e-06  2.80464449e-09]
 [-9.83811294e-02  7.66168682e-09 -9.75873689e-09 ... -5.60134007e-05
   2.00940969e-07  3.54422854e-09]
 [ 1.16156052e-01 -8.77654466e-09  8.16037764e-09 ... -1.20951657e-04
  -2.33669661e-06 -7.50668099e-09]
 ...
 [-1.83220638e-01 -8.89272060e-09 -9.81968100e-09 ...  2.35311186e-05
  -3.25484493e-06  9.02499060e-09]
 [-7.02096331e-01  3.05178374e-10  2.56061008e-09 ... -8.61759744e-04
   9.43449909e-05  3.83761998e-09]
 [-3.50933229e-01  8.85876862e-09 -6.57515140e-10 ... -1.80365926e-06
  -8.14464807e-06  8.79454531e-09]]
[[-0.76100352 -1.21244498 -0.10187131 -2.36850085 -1.05778129 -2.20823629
   0.56383834  1.21105294  2.21030997  0.44456156 -1.18244872  1.04289112
  -1.60558756  1.30419943  1.37175046  1.74825095 -0.23365648 -1.52014483
   1.15324176  0.10368082 -0.37207719 -0.61530019 -0.1256836  -2.27193038
  -0.71836208 -1.29690315]
 [-0.61785176  0.61559207 -1.26550639  1.85745418 -0.91853319 -0.05502589
  -0.38589806  1.29520853 -1.56843297 -0.97026419 -2.18334895 -2.85033578
  -2.07733086  1.63163164  0.3490229   1.82789117 -2.44174379 -0.8563034
  -0.2982564  -2.07947873 -1.2933238   0.89982032  0.28306578  2.31180525
  -2.46444086  1.45656548]
 [-0.68934072 -1.94538151  2.01360618 -3.12316188 -0.2361763   1.38680947
   0.90982429 -1.54774416 -0.79830896 -0.65599834  0.7353833  -2.58593294
   0.47210839  0.55349499  2.51255453 -2.4167454  -1.63898627  1.2027302
  -1.20245851 -1.83445959 -1.88013027 -0.34056098  0.23692483 -1.06137919
   1.02759232 -0.47690832]
 [-0.67832479  0.46299226  0.58492321 -0.1650184   1.93264192 -0.22965765
  -1.84731492  0.49011768  1.07146054 -3.31905643  1.54113507  0.37371947
  -0.86484681 -2.58273522  0.97062447 -0.51021867 -0.68427897 -1.64713607
   0.21153145 -0.27422442  1.72599755  1.32418658 -2.63984479 -0.08055871
  -2.03510803 -1.46123776]
 [-0.59664339 -2.04481799  2.05698407  1.95100909  0.17637699 -2.16141218
  -0.40394736  1.80157532 -1.56278739 -0.25253004  0.23586497  0.71656699
   1.07689092 -0.35457279 -1.67743058 -0.12939255 -0.67488849  1.14066535
   1.32431237  3.21158484 -2.15888898 -2.60164082 -3.2226466  -1.89612906
  -0.87488068  2.51038628]
 [-0.87794907  0.4344112  -0.93161049  0.18390778 -0.36078216  0.61958137
   0.38624948 -2.65150343  2.29710773 -2.08818098 -1.86382323  1.06057836
   0.77562146  2.1346861  -1.14973702 -0.52081426  0.99743429 -1.48309353
  -2.3139424   0.29517333 -0.38704879 -2.20607697  0.30702191 -1.17646114
  -1.63462966 -0.82467661]
 [-0.52746527  1.21564288 -1.50095981 -2.03195359 -1.52366734 -2.43732079
  -2.37570311 -1.39987277 -0.88735315 -0.63278873  1.50450176 -1.580763
   0.58599217 -0.77540416  0.94257331  2.10919653  0.54479132  0.43773612
  -1.28024228 -0.04360994  1.4774997  -1.13276949 -0.72846904  0.04734716
   1.6574566   1.68540944]
 [-0.7490154  -0.72249056 -3.15228173  0.36577778  0.19811362 -0.73059946
   1.65263918 -2.300357   -1.87468162  0.98095387 -1.58825159  1.35434142
   2.17895331 -1.99239762 -2.00371362 -0.388613   -2.33992976 -2.91719062
   0.99398645 -2.70476768 -1.27139772  1.86091461 -1.20519404 -0.38014194
   0.7087181  -2.11014003]
 [-0.6665468   0.53601845  1.30307573 -1.03372714 -4.03084753  0.58173469
  -2.65717902  0.80379994 -1.09241928  2.49910058  0.362008    0.66195337
  -0.92160534 -0.83123666 -2.00200952 -2.94897501  0.64564202 -1.10114694
   0.74510309  0.58506717 -1.99545251  0.62591105  1.80596103 -0.22309315
  -1.40442136 -2.1319153 ]
 [-0.46089119 -1.43944954 -1.21809509  0.71093011  0.45216919 -0.35953381
   0.62284954 -0.67005297 -0.7069138   0.06311351 -1.23199074 -1.74645233
  -2.71960897 -2.21437178 -1.69307505 -0.90927394  0.87852311  1.18664814
  -1.87041262  0.39796295  1.72113872 -1.36934055  0.8580668  -0.24779579
   1.28009118 -1.32752042]]
```

Обучим нейронную сеть с помощью метода градиентного спуска

```
y = data["y"]
x = data["X"]
trainWeights1 = generateWeights((25, 401))
trainWeights2 = generateWeights((10, 26))
trainWeights = gradient([trainWeights1, trainWeights2], x, 0.0000001)
trainWeights1, trainWeights2 = trainWeights[0], trainWeights[1]
l1, l2 = h(trainWeights1, trainWeights2, x)
```

Вычислим процент правильных классификаций на обучающей выборке (87.52)

```
def predictionPercentValue(resultLayer, y):
    predictions = np.argmax(resultLayer, axis=1) + 1
    predictionsCount = 0
    for predictionValue, realValue in zip(predictions, y):
        if predictionValue == realValue:
            predictionsCount += 1
    percentValue =  float(predictionsCount) / len(y) * 100
    return percentValue
```

Ответы на вопросы.

Как были инициализированы весовые коэффициенты и влияют ли их начальные знаичения на результат обучения?

Весовые коэффициенты инициализированы случайными значениями близкими к нулю.

Как влияют параметры обучения на результат?

В зависимости от распределения случайной величины качество варьируется на 20-30%

Как зависит функция потерь от количества итераций?

С количеством итераций, функция потерь уменьшается

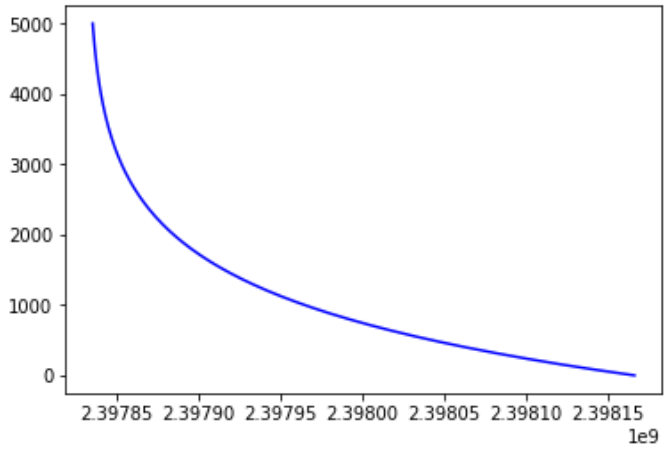

Рисунок 1 График функции потерь. Как видно с количеством итераций функция убывает

Каковы достигнутые полнота и точность классификации\распознавания или какова ошибка прогнозирования?

Используя алгоритм обратного распространения ошибки удалось достигнуть ошибки, равной 0.65

Заключение:

В лабораторной работе я качественно обучила нейронную сеть с помощью программных средств машинного обучения.

Список источников:

[1]Nils J. Nilsson Introduction To Machine Learning // Stanford University, 1998.

[2] Hal Daume A Course in Machine Learning // 2015.

[3] D. Michie, D.J. Spiegelhalter, C.C. Taylor Machine Learning, Neural and Statistical Classication // 1994.

[4] Max Kuhn, Kjell Johnson. Applied Predictive Modeling // Springer, 2013.

[7] Andrew Ng http://www.mlyearning.org/