# Linear Algebra in Neural Network

CHEN Ming-yu 2017200506032
FENG Cheng-lin 2017200506035
YU Hong-ze 2017200506034

June 10 2018

## 1 Abstract

Today, one of the researches on aritificial intelligence is neural network. Sometimes people will find it hard to decide some situations, in other words, many decisions of human beings do not have quantified standards. Nevertheless, neural network can fulfill this job by training to get stabilized infrastructure.

## 2 Introduction

Our project aims to use fully connected neural network to find out what kinds of stature ladies prefer. Firstly, using questionaire to collect enough data. And then training the network to obtain a distinct standard. Specifically speaking, according to the comparison between its output and the correct answer, the machine is able to adjust the weights on the neural nodes automatically. Through enough training, the weights will converge to specific values and it shows a success on creating a clear standard as a reference of ladies' preference. If the weights fail to converge, it means the relationship between chosen factors and ladies preference is poor. Eventually, by employing the well-trained network, we are capable of deciding whether certain type of figure is loved by female.

# 3 Algorithm

Fully connected neural network with proper activated function is a non-linear algorithm which is capable of solving classification problems for the items with multiple features. Unlike a function in traditional concept, a fully connected neural network propagates the inputs-the features of items waiting to be classified-through several layers of neuros and obtains the final outcomes in the last layer, and each neuro must represent a non-linear transformation. The propagation, in detail, is to take the outputs of the previous layer as the inputs of the current. For one neuro, the independent parameter of the function is the sum of the products of each neuro in previous layer and its corresponding adjustable weight. The advantage of fully connected neural network is the absence of deduction of expression. Once the network was constructed,the structure is optimized by iteration.

# 4 Mathmatic Expression

In terms of the mathematic expression of this algorithm, we discuss a simpler version, two features with one layer of four neuros and an output, compared with the structure we used.

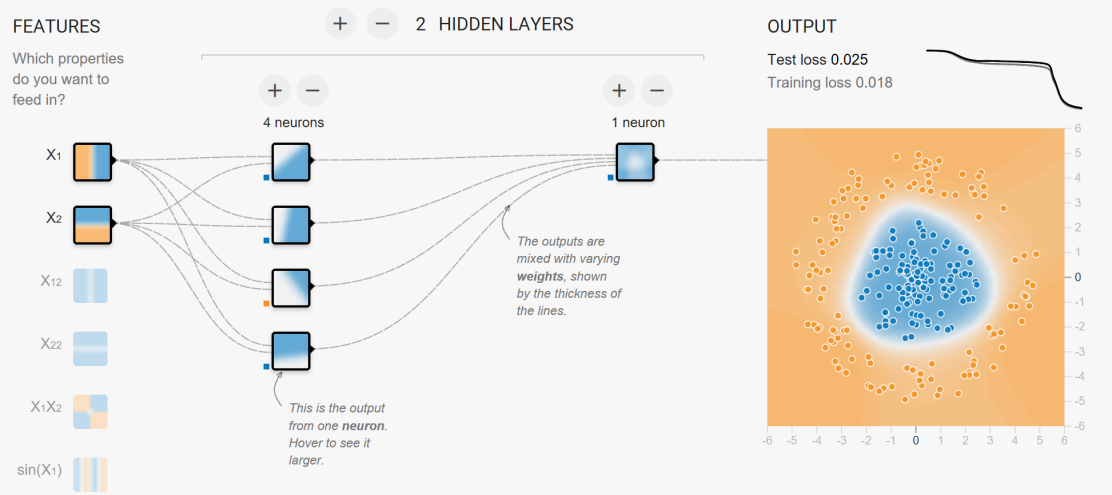


Figure 1: A simplified version[1]

The sum of the products of outputs and weights is converted into matrix multiplication by introducing the property of inner product in linear algebra. The matirx (1) represents the features of the each single item that we study, and matrix (2) represents the weights on the lines connecting the nodes in the first layer and the second, which is showed in Figure 1.

$$\begin{bmatrix} x_{inputs} \end{bmatrix} = \begin{bmatrix} x_1 & x_2 \end{bmatrix} \tag{1}$$

$$W^{(1)} = \begin{bmatrix} W_{1,1}^{(1)} & W_{1,2}^{(1)} & W_{1,3}^{(1)} & W_{1,4}^{(1)} \\ W_{2,1}^{(1)} & W_{2,2}^{(1)} & W_{2,3}^{(1)} & W_{2,4}^{(1)} \end{bmatrix} \tag{2}$$

By implementing dot product, the values of features propagate to the second layer, the matrix (3). The same approach is used to calculate the final output. (NB: sigmoid funtcion is uesd to

cancel linearity.)

$$a_{layer} = \begin{bmatrix} a_1 & a_2 & a_3 & a_4 \end{bmatrix} = f_{sigmoid}(x_{inputs}W^{(1)}) = f(\begin{bmatrix} x_1 & x_2 \end{bmatrix} \begin{bmatrix} W_{1,1}^{(1)} & W_{1,2}^{(1)} & W_{1,3}^{(1)} & W_{1,4}^{(1)} \\ W_{2,1}^{(1)} & W_{2,2} & W_{2,3}^{(1)} & W_{2,4} \end{bmatrix})$$

$$= \begin{bmatrix} f(W_{1,1}^{(1)}x_1 + W_{2,1}^{(1)}x_2) & f(W_{1,2}^{(1)}x_1 + W_{2,2}^{(1)}x_2) & f(W_{1,3}^{(1)}x_1 + W_{2,3}^{(1)}x_2) & f(W_{1,4}^{(1)}x_1 + W_{2,4}^{(1)}x_2) \end{bmatrix} \tag{3}$$

$$\begin{bmatrix} y_{output} \end{bmatrix} = a_{layer}W^{(2)} = \begin{bmatrix} a_1 & a_2 & a_3 & a_4 \end{bmatrix} \begin{bmatrix} W_{1,1}^{(2)} \\ W_{2,1}^{(2)} \\ W_{3,1}^{(2)} \\ W_{4,1}^{(2)} \end{bmatrix}$$

$$= \begin{bmatrix} W_{1,1}^{(2)}a_1 + W_{2,1}^{(2)}a_2 + W_{3,1}^{(2)}a_3 + W_{4,1}^{(2)}a_4 \end{bmatrix}$$

However, the initial outcome may not be correct which indicates that improvements are necessary. The formula used to reflect the distance between the outcome and the labeled correct answer is the cross entropy, which is capable of reflecting the distance between two probability distributions.

$$H(p_{labeled}, q_{outcome}) = \sum_i p_i \times log_2(\frac{1}{q_i})$$

The problem we studied was converted into binary distribution with labeled wanted item 1 and unwanted item 0. Thus, after mapping the results within 0 to 1, the propagation outcome naturally becomes a binary digit. Then update each element in weight matrices by subtracting the gradients of cross entropy (the loss function). According to the figure above, the performance is acceptable after roughly 3000 steps.

# 5 Algorithmic Realization

To simplify the coding complexity, a python module TesnorFlow[2] is introduced. The algorithm are supposed to be realized in python environment with Jupyter[3]. We conducted a survey enquiring 5 different factors, height, weight, chest, waist and hip sizes, to determine wheter a male is charming or not.

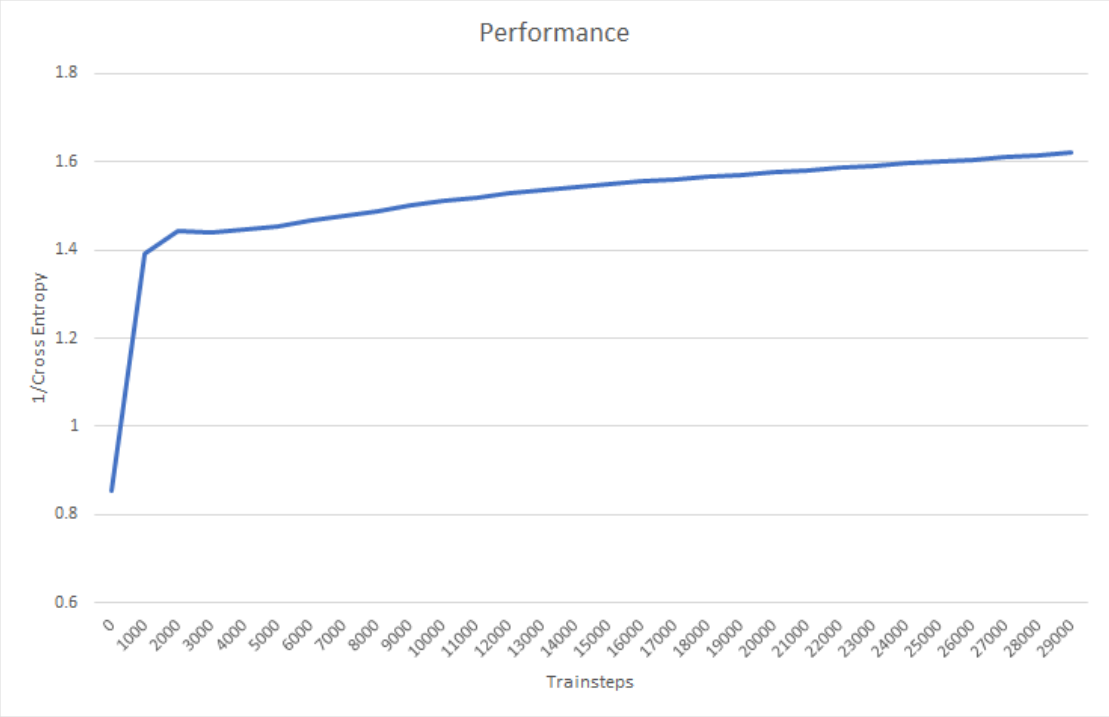


Figure 2: Performance through out 30000 training stpes.

For each step of training, the network picks out eight five-dimentional feature vectors whoes elements are the data for one male's reported information. Then follow the process mentioned above to implement gradient descend. The reciprocal of cross entropy is used to indicate the perofrmance (or accuracy).

# 6 Conclusion

Since the reciprocal of of cross entropy shows the differentiation level of two subspaces in this five dimensional vectorspace, and as the value goes bigger, two subspaces distinguish more apparenly. After approximately 3000 steps of training, the reciprocal stabilizes at a level of 1.5, which means that the subspace of qualified male figure and that of unqualified are clearly differentiated. To be more specific, the neural network has fulfilled the duty of deciding the different figures to a satisfying extent. However, there is one thing worth mentioning that although the reciprocal of cross entropy keeps enlarging, we have not controlled the complexity of the model, for which the algorithm is just compelled to memorize the "right answer".

# 7 Appendix

**Training Sets**

| Y\N | Height | Weight | Chest | Waist | Hip |
|---|---|---|---|---|---|
| 1 | 177 | 73 | 96 | 80 | 94 |
| 1 | 175 | 71 | 103 | 84 | 97 |
| 1 | 182 | 72 | 102 | 81 | 102 |
| 1 | 174 | 67 | 102 | 81 | 97 |
| 1 | 183 | 69 | 103 | 80 | 94 |
| 1 | 167 | 70 | 95 | 88 | 96 |
| 1 | 168 | 73 | 101 | 81 | 99 |
| 1 | 167 | 70 | 99 | 81 | 98 |
| 1 | 176 | 68 | 103 | 83 | 96 |
| 1 | 176 | 72 | 97 | 80 | 102 |
| 1 | 173 | 70 | 97 | 83 | 99 |

| | | | | | |
|---|---|---|---|---|---|
| 1 | 173 | 71 | 96 | 84 | 99 |
| 1 | 172 | 72 | 100 | 88 | 99 |
| 1 | 174 | 71 | 97 | 88 | 100 |
| 1 | 172 | 73 | 97 | 88 | 98 |
| 1 | 183 | 70 | 97 | 85 | 98 |
| 1 | 171 | 70 | 99 | 85 | 98 |
| 1 | 177 | 68 | 101 | 84 | 97 |
| 1 | 179 | 68 | 95 | 81 | 99 |
| 1 | 169 | 73 | 99 | 85 | 102 |
| 1 | 179 | 69 | 95 | 86 | 101 |
| 1 | 174 | 68 | 97 | 86 | 99 |
| 1 | 177 | 68 | 99 | 80 | 99 |
| 1 | 178 | 73 | 100 | 84 | 96 |
| 1 | 171 | 71 | 102 | 82 | 98 |
| 1 | 177 | 72 | 99 | 80 | 100 |
| 1 | 170 | 68 | 102 | 88 | 98 |
| 1 | 173 | 72 | 96 | 83 | 97 |
| 1 | 168 | 71 | 95 | 81 | 94 |
| 1 | 169 | 69 | 97 | 86 | 98 |
| 1 | 181 | 69 | 100 | 86 | 100 |
| 1 | 172 | 70 | 96 | 83 | 98 |
| 1 | 177 | 68 | 102 | 81 | 100 |
| 1 | 169 | 67 | 97 | 85 | 95 |
| 1 | 175 | 67 | 95 | 82 | 102 |

| | | | | | |
|---|---|---|---|---|---|
| 1 | 169 | 67 | 98 | 83 | 101 |
| 1 | 177 | 69 | 97 | 82 | 96 |
| 1 | 174 | 73 | 100 | 81 | 97 |
| 1 | 177 | 69 | 103 | 82 | 96 |
| 1 | 171 | 72 | 95 | 82 | 97 |
| 1 | 181 | 68 | 99 | 81 | 97 |
| 1 | 173 | 71 | 99 | 84 | 99 |
| 1 | 181 | 70 | 97 | 80 | 97 |
| 1 | 168 | 70 | 98 | 81 | 99 |
| 1 | 177 | 71 | 98 | 85 | 97 |
| 1 | 173 | 68 | 97 | 86 | 97 |
| 1 | 175 | 67 | 102 | 86 | 99 |
| 1 | 180 | 68 | 100 | 81 | 95 |
| 0 | 153 | 60 | 78 | 65 | 82 |
| 0 | 142 | 62 | 81 | 75 | 76 |
| 0 | 142 | 55 | 86 | 61 | 77 |
| 0 | 144 | 51 | 88 | 60 | 73 |
| 0 | 146 | 49 | 85 | 63 | 73 |
| 0 | 157 | 50 | 82 | 74 | 80 |
| 0 | 154 | 53 | 76 | 69 | 85 |
| 0 | 154 | 52 | 85 | 64 | 76 |
| 0 | 157 | 50 | 74 | 66 | 88 |
| 0 | 155 | 59 | 80 | 60 | 73 |
| 0 | 148 | 49 | 77 | 63 | 72 |

| | | | | | |
|---|---|---|---|---|---|
| 0 | 154 | 54 | 73 | 73 | 87 |
| 0 | 148 | 52 | 89 | 61 | 72 |
| 0 | 141 | 50 | 83 | 61 | 79 |
| 0 | 149 | 50 | 76 | 59 | 79 |
| 0 | 143 | 57 | 87 | 65 | 70 |
| 0 | 147 | 52 | 77 | 66 | 81 |
| 0 | 150 | 55 | 89 | 65 | 87 |
| 0 | 154 | 61 | 79 | 66 | 83 |
| 0 | 146 | 52 | 77 | 74 | 86 |
| 0 | 143 | 53 | 72 | 70 | 83 |
| 0 | 148 | 52 | 76 | 63 | 81 |
| 0 | 141 | 55 | 86 | 71 | 78 |
| 0 | 157 | 55 | 77 | 62 | 76 |
| 0 | 141 | 59 | 81 | 74 | 80 |
| 0 | 202 | 81 | 120 | 117 | 115 |
| 0 | 200 | 87 | 120 | 106 | 137 |
| 0 | 208 | 77 | 110 | 111 | 115 |
| 0 | 193 | 77 | 137 | 112 | 130 |
| 0 | 209 | 96 | 124 | 97 | 118 |
| 0 | 195 | 92 | 115 | 117 | 114 |
| 0 | 200 | 92 | 111 | 99 | 118 |
| 0 | 206 | 86 | 131 | 108 | 111 |
| 0 | 200 | 80 | 134 | 107 | 108 |
| 0 | 208 | 89 | 126 | 95 | 108 |

| | | | | | |
|---|---|---|---|---|---|
| 0 | 209 | 91 | 137 | 114 | 127 |
| 0 | 196 | 98 | 132 | 105 | 109 |
| 0 | 199 | 80 | 121 | 104 | 137 |
| 0 | 206 | 78 | 112 | 116 | 135 |
| 0 | 197 | 82 | 128 | 113 | 136 |
| 0 | 199 | 85 | 129 | 98 | 122 |
| 0 | 196 | 94 | 124 | 94 | 136 |
| 0 | 205 | 85 | 123 | 105 | 129 |
| 0 | 193 | 81 | 127 | 95 | 119 |
| 0 | 210 | 78 | 118 | 113 | 122 |
| 0 | 195 | 98 | 113 | 100 | 111 |
| 0 | 201 | 85 | 118 | 105 | 122 |
| 0 | 205 | 97 | 124 | 98 | 127 |
| 0 | 203 | 98 | 128 | 94 | 133 |
| 0 | 204 | 86 | 136 | 107 | 110 |

Code

```
import xlrd
import tensorflow as tf
from numpy.random import RandomState

batch_size = 8
w1= tf.Variable(tf.random_normal([5, 7],name='matrix1', stddev=1))
w2= tf.Variable(tf.random_normal([7, 7],name='matrix2', stddev=1))
w3= tf.Variable(tf.random_normal([7, 1],name='matrix3', stddev=1))

x = tf.placeholder(tf.float32, shape=(None, 5), name="x-input")
```

```
y_= tf.placeholder(tf.float32, shape=(None, 1), name='y-input')


with tf.name_scope('graph') as scope:
    a1= tf.matmul(x, w1,name='product1')
    a2=tf.matmul(tf.nn.sigmoid(a1),w2,name='product2')
    y=tf.matmul(tf.nn.sigmoid(a2),w3,name='producr3')


    y=tf.nn.sigmoid(y)
    cross_entropy = -tf.reduce_mean(y_ * tf.log_2(tf.clip_by_value(y, 1e-10, 1.0))
    + (1 - y_) * tf.log(tf.clip_by_value(1 - y, 1e-10, 1.0)))
    train_step = tf.train.AdamOptimizer(0.001).minimize(cross_entropy)


book=xlrd.open_workbook('RANDOM.xlsx')
sheet0=book.sheet_by_index(0)
rows_number=sheet0.nrows


matrix_X=[]
for i in range(rows_number-1):
    temp=sheet0.row_values(i+1)
    del temp[0]
    matrix_X.append(temp)


matrix_Y=[]
for i in range(rows_number-1):
    temp=sheet0.row_values(i+1)
    del temp[1:6]
    matrix_Y.append(temp)


X = matrix_X
Y = matrix_Y
```

```
import xlwt
bookresult=xlwt.Workbook(encoding='uft-8',style_compression=0)
sheetresult=bookresult.add_sheet('sheet1',cell_overwrite_ok=True)

with tf.Session() as sess:
    writer = tf.summary.FileWriter("logs/", sess.graph)
    init_op = tf.global_variables_initializer()
    sess.run(init_op)

    print(sess.run(w1))
    print(sess.run(w2))
    print(sess.run(w3))
    print("\n")

    j=0
    STEPS = 30000
    for i in range(STEPS):
        start = (i*batch_size) % (rows_number-1)
        end = (i*batch_size) % (rows_number-1) + batch_size
        sess.run([train_step, y, y_], feed_dict={x: X[start:end], y_: Y[start:end]
                                                 })
        if i % 1000== 0:
            total_cross_entropy = sess.run(cross_entropy, feed_dict={x: X, y_: Y})
            print("After %d training step(s), cross entropy on all data is %g" % (
                                                      i, total_cross_entropy))
            sheetresult.write(j,0,i)
            sheetresult.write(j,1,1/total_cross_entropy)
            j=j+1

    bookresult.save(r'RESULT.xls')
    print("\n")
```

```
print(sess.run(w1))
print(sess.run(w2))
print(sess.run(w3))
```

## References


[1] Snipped from TensorFlow Playground. http://playground.tensorflow.org/

[2] TensorFlow is an open-source software library for dataflow programming across a range of tasks. it was developed by the Google Brain team for internal Google use.

[3] The Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text.