The Linear Algebra Apllied in Neural Network

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

June 10 2018

1 Algorithm

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

2 Mathmatic Expression

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

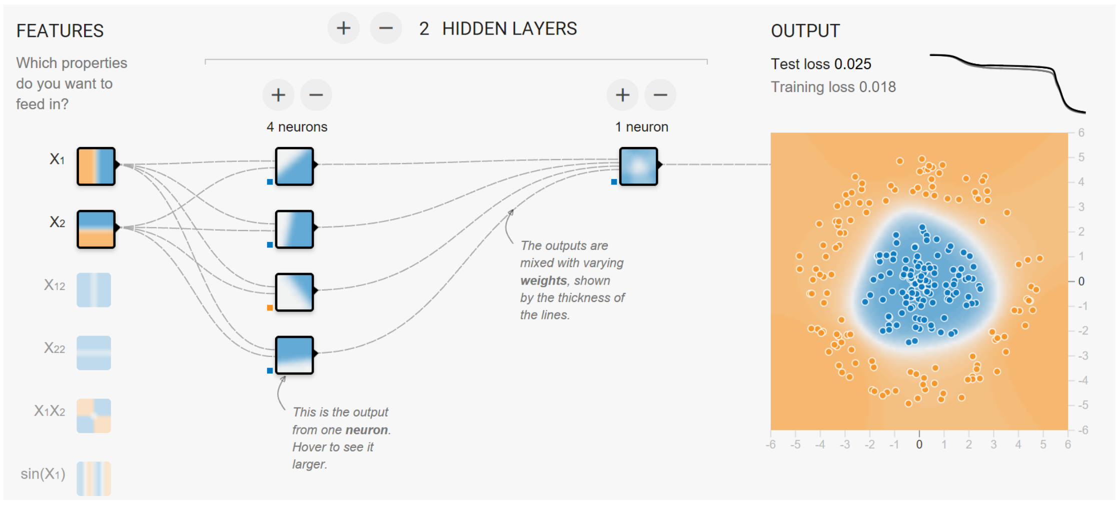
FEATURES
Which properties do you want to feed in?
X1
X2
X12
X22
X1X2
sin(X1)
+ − 2 HIDDEN LAYERS
+ −
4 neurons
+ −
1 neuron
The outputs are mixed with varying weights, shown by the thickness of the lines.
This is the output from one neuron. Hover to see it larger.
OUTPUT
Test loss 0.025
Training loss 0.018

Figure 1: A simplified version[1]

The sum of the product of outputs and weights is converted into matrix product by introducing the property of inner product in linear algebra. The matirx (1) represents the features of the each single items that we study, and matrix (2) represents the weights on the lines connecting the nods in first layger and second, which is showen in figure above.

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

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

By implementing dot multiple, the value of features propagate to the second layger, the matrix (3). The same approach are used to calculate final output. (NB: sigmoid funtcion is uesd to cancel linearity.)

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

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

However, the initial outcome may not be correct indicating the network needs improvement. The formula we used to reflect the distance between the outcome and the labeled correct answer is the cross entropy, which is capable to reflect the distance between two probability distribution.

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

The problem we study 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. Then update each laments in weights matrixes by subtracting the gradients of cross entropy (the loss function).

3 XXX

References

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