University of L'Aquila

Department of Information Engineering, Computer Science and Mathematicst

Hyper Heursitic Cryptography with Mixed Adversarial Nets

Author :
Aly SHMAHELL

Supervisor :
Prof. Giovanni DE GASPERIS

June 5, 2018

ABSTRACT

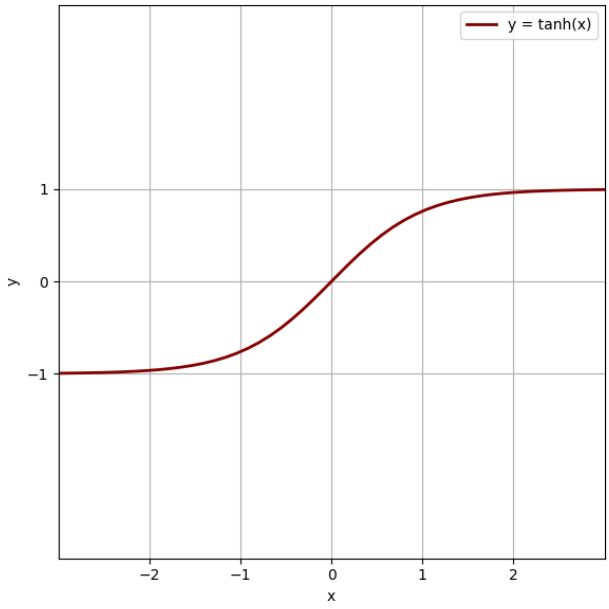
Abstract

1 INTRODUCTION

1.1 XAVIER INITIALIZATION [1]

When we talk about initialization, we mean the initial value which should be given to a certain neuron, this initial value is important for the neuron and for the network in general.

Lemma 1 *Suppose our net uses the hyperbolic tangent activation function for its neurons:*

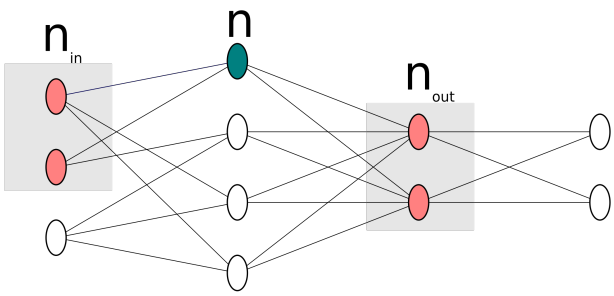


- *If the weights start too small, then the signal shrinks as it passes through each layer until it vanishes, then as it passes deeper in the network with its small values, the layers it enter will become linear, because the output of the hyperbolic tangent is linear with small input value, this means the deeper layers of the net will loose non-linearity.*
- *If the weights start too large, then the signal grows as it passes through each layer until it becomes too large, then as it passes deeper in the network with its large values, the layers it enter will become saturated, as the output of the hyperbolic tangent is flat with large input values, and this flatness will cause the gradient to become zero, and we will get the vanishing gradient problem.*

Lemma 2 *Having a pre-defined net graph: for each neuron we know the number of inputs and the number of outputs, therefor we can calculate a reasonable weight for the neuron in question based on a normal distribution of a zero mean and a 1/n variance.*

Lemma 3 *To achieve initialization while avoiding the two obstacles in Lemma 1, we want the variance to remain the same with each passing layer.*

Suppose we have an input X from a previous layer with n components and a linear neuron with random weights W in the current layer that spits out the same output Y to some neurons in the next layer.



The output of the neuron will have the following equation:

$$Y = W_1X_1 + W_2X_2 + ... + W_nX_n \tag{1}$$

To calculate the variance of each component:

$$Var(W_iX_i) = E[X_i]^2Var(W_i) + E[W_i]^2Var(X_i) + Var(W_i)Var(X_i) \tag{2}$$

Since our inputs and weights come from a normal distribution of zero mean (from Lemma 2):

$$E[X_i]^2Var(W_i) + E[W_i]^2Var(X_i) = 0 \implies Var(W_iX_i) = Var(W_i)Var(X_i) \tag{3}$$

Since the neurons in the same previous layer are all independent, we assume that both X_i and W_i are independent and also identically distributed:

$$Var(Y) = Var(W_1X_1 + W_2X_2 + ... + W_nX_n) = nVar(W_i)Var(X_i) \tag{4}$$

In the last equation, we have the variance of the inputs, the variance of the output and the variance of the weights, now we can calculate the variance of the weights from Lemma 3:

$$Var(Y) = Var(X_i) \implies Var(W_i) = \frac{1}{n_{in}} \implies Var(W_i) * n_{in} = 1 \tag{5}$$

Now if we go through the same derivation for back-propagation, we get:

$$Var(W_i) = \frac{1}{n_{out}} \implies Var(W_i) * n_{out} = 1 \tag{6}$$

To keep the variance of the input gradient & the output gradient the same, we combine (5) & (6) and we get:

$$(n_{out} + n_{in}) * Var(W_i) = 2 \implies Var(W_i) = \frac{2}{n_{in} + n_{out}} \tag{7}$$

2 DESIGN

3 IMPLEMENTATION

4 CONCLUSION

REFERENCES

[1] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2010, Chia Laguna Resort, Sardinia, Italy, May 13-15, 2010*, pages 249–256, 2010.

APPENDIX