15) Vanishing / Exploding Gradients

a) Vanishing Gradients

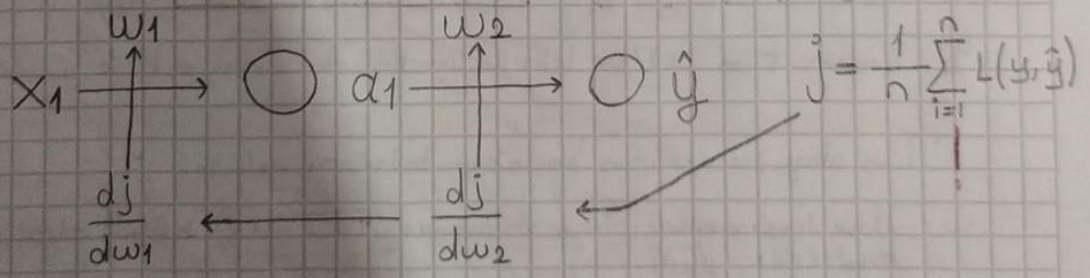
Problem :

Sigmoid function

↳ $\sigma(x) = \frac{1}{1+e^{-x}}$

* First derivatives of sigmoid function values can range from 0 to 0.25.

For the nodes with sigmoid activation functions, we know that the partial derivative of the sigmoid function reaches a max. value of 0.25. When there are more layers in the network, the value of the product of derivative decreases until at some point the partial derivative of the "Loss function" approaches a value close to zero, and the partial derivative vanishes. In a network with vanishing gradient, the weights can not be updated, so the network can not learn.



$$\frac{dj}{dw_2} = \frac{dj}{d\hat{y}} \cdot \frac{d\hat{y}}{dz_2} \cdot \frac{dz_2}{dw_2}$$

↳ $= \sigma(z_2)(1-\sigma(z_2)) = \frac{1}{1+e^{z_2}} \cdot \left(1 - \frac{1}{1+e^{z_2}}\right)$

Assume; $z_2 = 5$

↳ $\frac{d\hat{y}}{dz_2} \approx 0.0066$

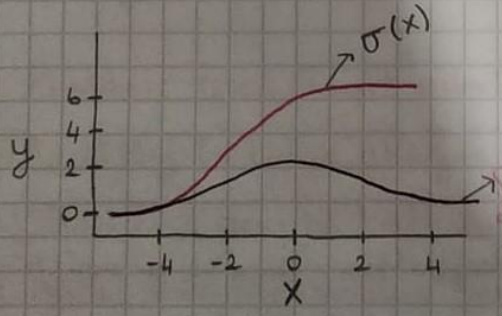
(2)

$$\frac{dj}{dw_1} = \frac{dj}{d\hat{y}} \cdot \underbrace{\frac{d\hat{y}}{dz_2}}_{\text{0.0066}} \cdot \frac{dz_2}{da_1} \cdot \underbrace{\frac{da_1}{dz_1}}_{\text{Sigmoid derivative}} \cdot \frac{dz_1}{dw_1}$$

! If we have several of these expressions in our chain of derivatives, the gradient quickly shrinks to a number so small.

2 nolu denklemden çok rahat görülebilir ki 2 aktivasyon fonksiyonun türevlerini değerlendirmemiz gerekiyor.

Özetle; → The "vanishing gradients" issue occurs when we use the Sigmoid or Tanh activation functions in the hidden Layer.



→ When the inputs become fairly small or fairly large, derivative get close to 0.

Bu problemi nasıl çözeceğiz?

1) Vanishing gradient problem is caused by the "Derivative of the activation function". The simplest solution to the problem is to replace the Activation function of the network.

Use "ReLU".

Derivative of a ReLU is 1 for inputs greater than 0.

" " " " is 0 " " that are negative.

→ ReLU'nun problemi, gradient 0 olduğunda ortaya çıkar. Bu gibi durumlarda node aslında ölüdür, çünkü yeni ve eski Weight değerleri aynı kalmıştır. (dying ReLu problem)

↳ Tabi, burada çözümü vardır. **Leaky ReLU** kullanmak.

Özetle;

- Gradient = 1 when the output of function is >0
- Gradient = 0 " " " " " " <0

Yani sadece 2 seçenek var. 1 veya 0. Bu durumda Weight update is nothing or takes contributions from the other weights and biases.

ÖR:

$$\frac{dj}{dw_1} = \frac{dj}{d\hat{y}} \underbrace{\frac{d\hat{y}}{dz_2}}_{\text{sigmoid tercih edilir output layer.}} \frac{dz_2}{da_1} \underbrace{\frac{da_1}{dz_1}}_{\text{0 or 1 if ReLU used.}} \frac{dz_1}{dw_1}$$

2) Multi-level Hierarchy

3) Weight Initialization

Problemi Nasıl Tespit Ederiz?

1) By analizing kernel weight distribution. There is a vanishing gradient problem if the weights are falling regularly near zero.

1) Very slow in training network.

3) Parameters in higher layers change great, but parameters of lower layers less change.

4) Birkaç iterasyon sonrası modelin hiç iyileşmediğini görmek.

b) Exploding Gradient

Problem:

It occurs when the derivatives or slope get larger and larger as we go backward with every layer during Backpropagation.

This problem happens because of **Weights, not because of the Activation Function.** Yüksek Weight değerlerinden ötürü türevlerde yüksek olur; böylece Yeni ağırlık eskiye göre çok değişir. As a result; gradient never converge.

↳ It may result in oscillating around Minima and never come to a global minima point.

Vanishing Gradient → Occurs due to the Sigmoid and Tanh Activation.

Exploding Gradient → Occurs due to Large Weights.

Problemi nasıl Tespit ederiz?

1) Resulting in Large changes in loss from update to update.
2) Model Loss goes to NaN during training.
3) Model weights quickly become very large during Training.
4) Model weights go to NaN values during Training
5) Error gradient values are consistently above 1 for each node and Layer during Training.

Problemi nasıl Çözeriz?

1) Re-Design Network.
   a) Have fewer Layers
   b) Have smaller batch size

2) Use Gradient Clipping (Kırpma)

Specifically, the values of the error gradient are checked against a threshold value and clipped or set to that threshold value if the error gradient exceeds threshold. (Keras'ta var)

3) Use Weight Regularization

→ Using an L_1 or L_2 penalty on the recurrent weights can help with Exploding Gradient.

* Implement penalty to the network's loss function for big weight values if exploding gradients are happening.

4) Proper Weight Initializer

Özetle;

Exploding gradients are a problem when large error gradients accumulate and result in very large updates to neural network model weight during training.

Weight Initialization:

→ Too large initialization leads to Exploding Gradients.

→ Too small initialization leads to Vanishing Gradients.

To large:

Consider the case where every weight is initialized slightly larger than the identity matrix.

$$W^{[1]} = W^{[2]} = \dots = W^{[L-1]} = \begin{bmatrix} 1.5 & 0 \\ 0 & 1.5 \end{bmatrix}$$

$$\hat{y} = W^{[L]} \cdot 1,5^{L-1} \cdot X$$

bu yüzden the values of $a^{[L]}$ increase exponentially with l.

When these activations are used in backpropagation, this leads to the exploding gradient problem.

Since, gradients of the cost with respect to the parameters are too big. This leads the cost to oscillate around its minimum value.

To small:

let, $W^{[1]} = W^{[2]} = \dots = W^{[L-1]} = \begin{bmatrix} 0.5 & 0 \\ 0 & 0.5 \end{bmatrix}$

$$\hat{y} = W^{[L]} \cdot 0,5^{L-1} \cdot X$$

bu yüzden values of activation ($a^{[l]}$) decrease exponentially with l.

That leads to vanishing gradient problem. The gradients of the cost with respect to the parameters are too small, leading to convergence of the cost before it has reached the minimum value.

Özetle;

↳ Initializing weights with inappropriate values will lead to divergence or a slow-down in the training of your neural network.

Main weight Initialization Methods:

a) Uniform Distribution
b) Xavier Initialization
c) He Initialization

} Kullanılan aktivasyon fonksiyonuna bağlı hangisinin iyi olduğuna karar verilir.