19CST303 - NNDL

Vanishing Gradient Problem

Vanishing gradient problem is a phenomenon that occurs during the training of deep neural networks, where the gradients that are used to update the network become extremely small or "vanish" as they are backpropogated from the output layers to the earlier layers.

During the training process of the neural network, the goal is to minimize a loss function by adjusting the weights of the network.

The backpropogation algorithm calculates these gradients by propogating the error from the output layer to the input layer.

The consequences of vanishing gradient problem include slow convergence, network getting stuck in low minima, and impaired learning of deep representations.

What is the importance of gradient descent in training neural networks?

1. Weight optimization
2. Backpropogation
3. Optimization landscape exploration
4. Efficiency & scalability
5. Generalization & learning

What is the impact of the vanishing gradient problem on deep learning models?

After understanding what is vanishing gradient problem, here's how it impacts Deep learning models:

1. Limited learning capacity
2. Difficulty in capturing the long-term dependencies
3. Slow convergence & training instability
4. Preferential learning in shallow layers
5. Architectural design considerations

How do you know if your model is suffering from the vanishing gradient problem?

Here are some signs that are indicators of your problem suffering from the vanishing gradient problem:

- The parameters of the higher layers change to a great extent, while the parameters of lower layers barely change (or, do not change at all).
- The model weights could become 0 during training.
- The model learns at a particularly slow pace and the training could stagnate at a very early phase after only a few iterations.

What causes the vanishing gradient problem?

The vanishing gradient problem is caused by the fact that while the process of backpropagation goes on, the gradient of the early layers (the layers that are nearest to the input layer are derived by multiplying the gradients of the later layers (the layers that are near the output layer). Therefore, if the gradients of later layers are less than one, their multiplication vanishes at a particularly rapid pace.

How to calculate gradients in neural networks?

1. Forward pass - Input is propogated through the network, layer by layer, for computing the output predictions. It has 2 layers - linear transformation & activation function.

2. Loss calculation - After forward passing is done, the output predictions are compared to true labels to compute the loss.

3. Backpropogation - Used to calculate the gradients of the loss function wrt the weights of the network. It has 2 steps - backward pass & gradient calculation.

4. Weight update - After calculating the gradients, the total weights of the network are updated to minimize the loss function.

Why is the vanishing gradient problem significant?

The vanishing gradient problem causes the gradients to shrink. But, if a gradient is small, it won't be possible to effectively update the weights and biases of the initial layers with each training session.

These initial layers are vital for recognizing the core elements of the input data, so, if their weights and biases are not properly updated, it is possible that the entire network could be inaccurate.

What are activation functions?

Activation functions have a direct impact on the occurence of vanishing gradient problems in neural networks. Here are a couple of activation functions:

1. Sigmoid & Tanh activation function
2. ReLU activation function
3. Leaky ReLU and Parametric ReLU activation function
4. Exponential linear units
5. Scaled exponential units
6. Swish activation functions

How do you overcome the vanishing gradient problem?

Here are some methods that are proposed to overcome the vanishing gradient problem:

1. Residual neural networks (ResNets)

2. Multi-level hierarchy

3. Long short term memory (LSTM)

4. ReLU

5. Batch normalization

6. Weight initialization

7. Gradient clipping

8. Skip connections

9. Gated architectures

10. Reduced network depth

11. Faster hardware

1. Residual neural networks (ResNets)

Here's a better understanding of how does ResNet solve vanishing gradient problem by the use of skip connections to learn the residual mapping, enabling easier gradient flow & efficient training of deep neural networks.

2. Rectified linear unit (ReLU)

ReLU avoids the saturation issues of functions like sigmoid tangent or hyperbolic tangent, that can cause the gradients to vanish.

ReLU allows for better gradient flow, promoting more effective training of deep networks.

3. Multi-level hierarchy

Multi-level hierarchy involves pre-training a single layer at a time and then performing backpropagation for fine-tuning.

4. Long Short Term Memory

LSTMs are designed to capture the long-term dependencies by allowing information to persist over many time steps by incorporating the memory cells & gating mechanisms.

This enables the effective training of deep recurrent networks & the modeling of complex temporal dependencies in sequential data.

5. Batch normalizaltion

It is a technique that normalizes the activations within each min-batch during training.

It helps to reduce the internal shifts & ensures more stable gradient propagation.

6. Weight initialization

Initializing the weights with appropriate values minimizes the possibility of the gradients becoming too small or too large.

Techniques like Xavier/Glorot initialization set the initial weights based on the network's inputs & outputs, ensuring a more balanced & stable initialization.

7. Gradient Clipping

It is a technique where the gradients are rescaled to the maximum threshold during backpropogation.

If the gradients exceed the thresholds, they are scaled down to prevent them from exploding.

8. Skip connections

It provides direct connections between layers, allowing the gradients to bypass multiple layers during backpropogation.

9. Gated architectures

Architectures like Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) incorporate gating mechanisms that selectively control the flow of information within the network.

These architectures allow the network to retain & propogate relevant information over long sequences.

GRUs are designed to overcome the limitations of traditional RNNs by providing efficient solutions for modeling the sequential data.

They achieve this by incorporating gating mechanisms (similar to LSTM networks) but with a simpler architecture.

10. Reduced network depth

Shallower networks have shorter paths for propogating gradients, reducing the likelihood of the gradients vanishing or exploding.

What is exploding gradient problem?

It's a counterpart to the vanishing gradient problem that occurs when the gradients in a neural network grow exponentially during the training phase.

This can lead to unstable training dynamics, making it challenging for the network to converge into an optimal solution.