Neural Network Training

can code communities

Training

Neural networks don't inherently know how to perform a task. They learn through a process called training, which involves iteratively adjusting their internal parameters (weights and biases) to minimize error and improve accuracy on unseen data.

Forward Pass

Forward propagation (or forward pass)

- The calculation and storage of intermediate variables (including outputs) for a neural network in order from the input layer to the output layer.
    - The network takes an input and feeds it through its layers.
    - Each neuron performs calculations based on its weights, biases, and the input it receives from previous layers.
    - Activation functions determine whether a neuron "fires" and transmits its output to the next layer.
    - This process continues until the final output layer produces a prediction.

Cost

- We compare the network's prediction (output) to the desired output using a cost function.
- Different cost functions are used for different tasks (e.g., mean squared error for regression, cross-entropy for classification).
- Each function measures the discrepancy between the network's prediction and the desired output.
- Quantifies how "wrong" the network's prediction is.

Cost vs. Loss Function

The cost function and loss function are used semi interchangeably.

- The cost function is the average error of n-samples (typically the test dataset).
- The loss function represents the error calculated on only one data sample.

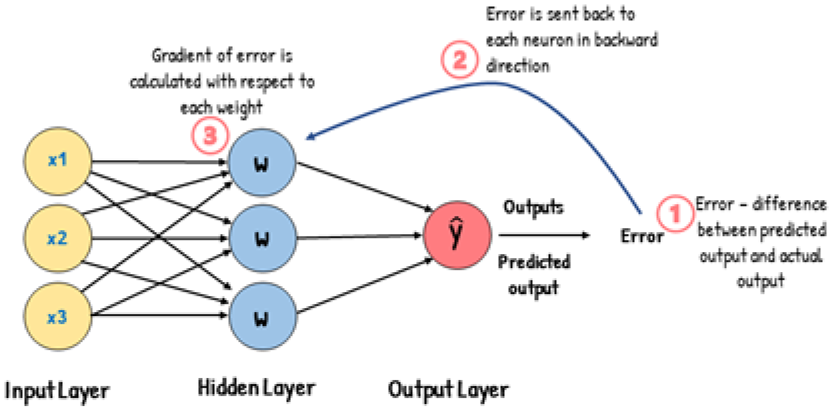
Exercise

Research and find the most appropriate cost/loss function for a sentiment analysis task. Pick one function and explain your reasoning for your choice.

Remember a sentiment analysis task outputs probabilities for three options (positive, negative, neutral). Here is a helpful link that explains different cost/loss functions in a simplified manner.

Backpropagation

- The network adjusts its weights and biases in a backward fashion based on the cost.
- The adjustments aim to minimize the overall cost
  - Essentially "teaching" the network how to improve its predictions for future examples.

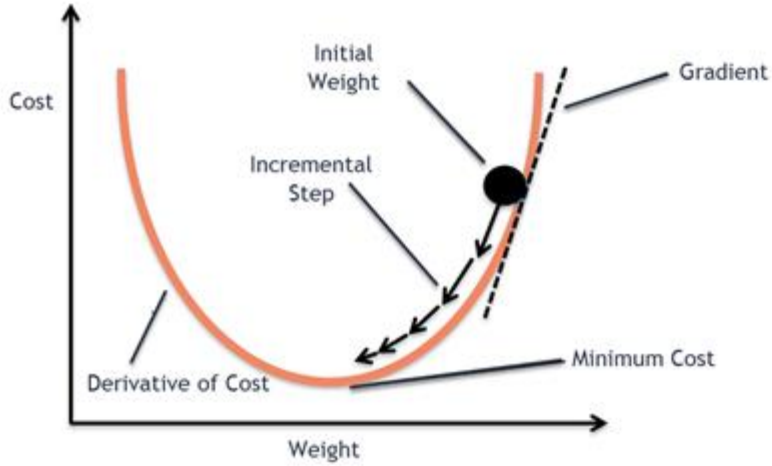


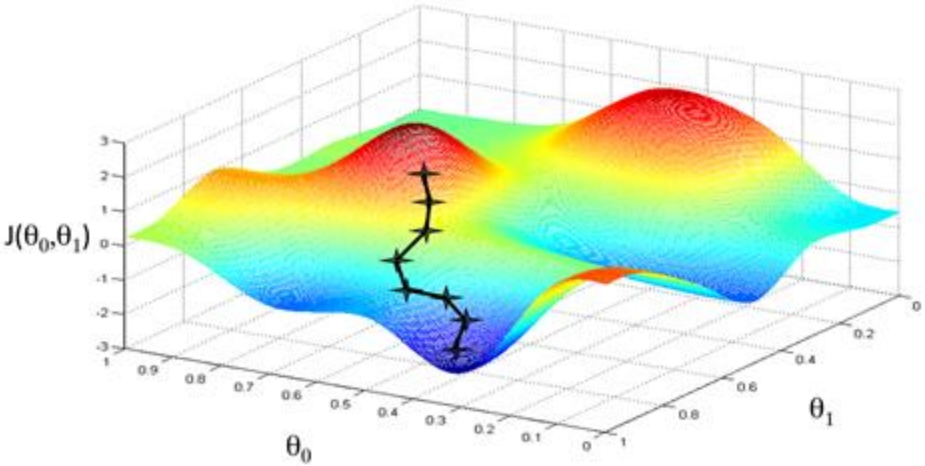
Gradient Descent

- An optimization algorithm that we use to lower the cost value as close to zero as possible.
    - Imagine you're lost in a mountain range and want to find the lowest valley (minimum elevation). Gradient descent helps you get there!
    - The cost function acts like the landscape in our analogy.
    - The valley (minimum point) which represents the best performance for the network.
- The gradient tells you the direction of the steepest descent in the cost function landscape.
    - It indicates how much and in which direction you should adjust the network's parameters to move closer to the minimum point.

Gradient Descent (continued)

- Gradient descent iteratively updates the network's parameters based on the calculated gradient.
  - You can think of it as taking small steps downhill based on the direction provided by the gradient.





Learning Rate

A critical factor is the learning rate, which controls the step size.

- A large learning rate might lead to large jumps that could miss the minimum or even jump you right past it.
- A small learning rate might make the process too slow.
- A fixed learning rate might not be optimal throughout training.
- Scheduling the learning rate (e.g., gradually reducing it) can help the network converge better.

Batch Size

Batch size refers to the number of data points processed by the network during a single training update. Common choices include batch sizes of 1, 32, 64, or 128.

- Larger batches can lead to faster training as the gradients are averaged over more data points, potentially leading to smoother optimization.
    - However, they might not capture the nuances of individual data points as well.
- Smaller batches provide more frequent updates to the network parameters, potentially leading to better convergence in some cases.
    - It can be more sensitive to noise in the data.

Batch Size

- SGD (Stochastic Gradient Descent) is the most basic training style, where the network updates its parameters after processing each individual data point (batch size of 1).
- Mini-batch SGD is a compromise between SGD and large batches.
    - It processes a small group of data points (mini-batch) at a time, balancing computational efficiency with capturing some of the detail from individual data points.
    - Popular mini-batch sizes include 32, 64, and 128 samples.

Epoch

- An epoch represents one complete pass through the entire training dataset.
    - During an epoch, the network processes all the data points in the training set once.
- The number of epochs required for training depends on the complexity of the network, the size and quality of the training data, and the chosen learning rate.
- Training often continues for multiple epochs until the network achieves a desired level of performance or reaches a point where further improvement is minimal.

Architecture of Neural Networks

In our exploration of neural networks, we've seen how they learn and improve. Now, let's delve into the various network architectures designed to tackle specific types of problems.

Feed Forward

Feedforward Neural Networks (FNNs), the simplest type, serve as the foundation for many other architectures.

- They consist of stacked layers of neurons, where information flows in a single, forward direction from input to output.
- Applications include:
  - Image recognition (early techniques)
  - Spam filtering
  - Sentiment analysis (basic architectures)

Example

Walk through creating a feed forward neural network structure using the MNIST dataset and data preprocessing and the training process.

FNN Limitations

Feedforward Neural Networks (FFNs) are powerful tools, but they have limitations that other models were designed to address.

- Difficulties with Sequential Data
    - FNNs struggle with sequential data like text or time series because they lack internal memory. They process information layer by layer, making it challenging to capture long-term dependencies or relationships within a sequence.
- Limited Capability in Feature Extraction
    - While FNNs can learn features to some extent, they might not be as efficient for tasks requiring specific feature extraction, especially for complex data like images.
- Scalability Issues
    - Training FNNs on large datasets with many layers can be computationally expensive and prone to vanishing or exploding gradients, hindering performance.

Take Home Exercise

Look into stochastic gradient descent with momentum and write up an explanation of how momentum makes this optimization function more efficient than normal stochastic gradient descent.