INFO 7375 - Neural Networks & AI

Homework to Chapter - 11

Submitted By:

Abdul Haseeb Khan
NUID: 002844724
khan.abdulh@northeastern.edu

What is normalization and why is it needed?

Normalization makes information consistent and brings data together in a similar format so that it is easier to interpret and use, with the goal to reduce redundancy and dependency within the stored information while ensuring integrity and eliminating anomalies. It is needed because different components of a dataset may have different scales that create artificial importance and training disbalance, and normalizing helps gradient descent work adequately for all components by making gradients similar in value so the learning rate impacts each component equally.

Different components of the original dataset can be on very different scales, which creates artificial importance for some components and can cause disbalance during artificial neural network training. For example, a price measured in cents looks higher than the same price in dollars, and if a feature vector has $x_1 = 1{,}000{,}000$ and $x_2 = 1$, the impact on the result from x_2 will be negligibly small compared to the impact from x_1. Through data normalization, information is made consistent and errors are removed and brought together in a similar format so it is easier to interpret and use, and its goal is to reduce redundancy and dependency within the stored information, ensuring integrity and eliminating anomalies. The method described normalizes the input dataset X over the entire batch of training samples in vector form $x_1(p), x_2(p), \ldots, x_{N_x}(p)$ where μ_k and σ_k^2 are the mean and variance across the x_k-th component of X in the entire batch. Normalization is further motivated because gradient descent works adequately for all components of the data when the gradients are similar by value, allowing the learning rate to equally impact changes by each component.

What are vanishing and exploding gradients?

Exploding and vanishing gradients may be caused by the initial conditions of the weights $W^{[s]}$, and for a deep network with $a^{[L]} = W^{[L]}W^{[L-1]} \ldots WWX = W^L X$, gradients explode when w>1 and vanish when w<1. Vanishing gradients can also arise from activation functions, since with sigmoid or tanh activation the gradients a are very low at reasonably high z even for a single neuron, which leads to very small gradient values during backpropagation.

What Adam algorithm and why is it needed?

Adam, short for Adaptive Moment Estimation, is an adaptive learning rate algorithm designed to improve training speeds in deep neural networks and reach convergence quickly. It is needed because it adjusts learning rates for each parameter individually using moving averages of gradient moments, which helps optimization proceed efficiently toward a minimum, although its adaptivity can be sensitive to noise for sparse gradients.

Adam is described as an adaptive learning rate method whose results are generally better than every other optimization algorithm while aiming to reach convergence

quickly in deep networks. It combines Momentum and RMSP by calculating the exponential moving average of gradients and squared gradients, with decay rates controlled by parameters β1 and β2 to scale learning rates per-parameter adaptively.

Adam adjusts the learning rates for each parameter individually by using the first-order moment (mean of gradients) and the second-order moment (uncentered variance of gradients), which enables faster and more stable progress during training compared to a single global learning rate. It is designed to improve training speeds and reach convergence quickly, though its adaptive learning rate can be sensitive to noisy gradient estimates with sparse data, potentially leading to suboptimal convergence or divergence in some cases.

How to choose hyperparameters?

The recommended hyperparameters are given as learning rate r that needs to be tuned, $\beta 1 \approx 0.9$ for the gradients, $\beta 2 \approx 0.9$ for the squared gradients, and $\epsilon \approx 10^{-8}$. In this setup, β1 governs the exponential moving average of the first moment and β2 governs the exponential moving average of the second moment, with ϵ providing numerical stability in the adaptive scaling.