INFO 7375 - Neural Networks & AI

Homework to Chapter - 12

Submitted By:

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

What is learning rate decay and why is it needed?

Learning rate decay is a de facto technique that begins with an initially large learning rate and then reduces it by a certain factor after predefined epochs to avoid oscillations and ensure the descent of gradient-based training while maintaining fast progress early on [Slide 4].Gradient descent may come halfway and then start oscillating, so to avoid such an effect one first uses higher steps, then reduces them to ensure the descent [Slide 4]. Using a low learning rate at the beginning will slow down the descent process, while using a high learning rate facilitates the descent but may overshoot the minimum to cause oscillations, so gradual reduction makes the process fast without overshooting the minimum. A practical schedule is to start with the initial learning rate r to go through epoch 1, then reduce the learning rate to go through epoch 2, and so on, where ro is the initial learning rate, γ is the decay rate, and e is the epoch sequential number. Many methods can manage decay, including exponential decay r=α ero with α<1, as well as step down decay and manual decay. The optimal learning rate depends on the topology of the loss landscape determined by the model and dataset, where an optimal rate yields a steep drop in the loss, decreasing it below the optimum reduces the loss only very shallowly, and increasing it beyond the optimum causes the loss to bounce about the minima, which motivates starting higher and decaying over time.

What are saddle and plateau problems?

Saddle is a local optima problem where the gradient descent is shaped like a saddle making the algorithm think it is making progress, while the observation is that it is most likely stuck in the local optima.
Plateu are also a local optima problem where the training takes a lot of time.

Why should we avoid grid approach in hyperparameter choice?

Avoid a grid approach in hyperparameter choice because choosing randomly scattered rather than regularly distributed parameters better explores the set of possible values and helps avoid artificially infused period dependencies. A mini batch is a subset of the training samples, and training proceeds sequentially through mini batches with cost fluctuating per mini batch but decreasing over time while the added noise makes the process more robust and stable.
A hyperparameter is external to the model, cannot be learned within the estimator, and has no analytical formula to calculate an appropriate value, which motivates searching rather than computing it directly. The guidance is to try random values and avoid grids by choosing randomly scattered rather than regularly distributed parameters to better explore the set of possible values and to avoid artificially infused period dependencies. It is also advised to find the right scale by trying a coarse search first and then going down to a fine scale to pick hyperparameters.

What is mini batch and how is it used?

Batch includes all training samples used in training, whereas mini batches are subsets of the training samples and a series of mini batches include the entire sample set. A mini batch may consist of a single sample if it was decided to do so, and mini batches are denoted as {1}, {2}, and so on in the superscript, with notation distinguishing layers [s], samples (p), and mini batches {t}. For example, if the batch size M=5,000,000 then there are 5,000 mini batches of 1,000 samples each, and mini batch t is denoted X{t} Y{t}Training with mini batch gradient descent breaks the entire set of M samples into mini batches of T samples each, yielding about B=M/T mini batches processed sequentially one mini batch at a time. A representative loop performs forward propagation on X{t} for all T samples in mini batch t, computes the mini batch cost J{t}, backpropagates errors, and updates W[s] and b[s] with learning rate r before moving to the next mini batch. In mini batch gradient descent the loss fluctuates over training mini batches and does not necessarily decrease for each mini batch, but in the long run the cost decreases with fluctuations and the added variations (noise) make the process more stable