Considering a 3 layer neural network (1 hidden layer)

1 Algorithm

1. We will maintain L hash tables, each hash table will include K LSH hash functions of size R. For each input-output value (xi,yi) in our training data, we will hash these tuples into Q (predetermined) buckets in all L hash tables. This is a one time cost.

2. Send the first randomly selected batch of size N for training. Calculate the gradients for each sample in the batch and find the L2 norm of the gradients. Store the (input, L2 norm) in a min(max) heap of size K. Update the weights using the gradients.

3. Send in the second training batch (also randomly selected) and start processing the K inputs from previous step.

Processing
1.Hash K elements from the min(max)-heap and select the inputs for the next batch from each of the buckets that the K elements hash to .i.e hash one item from the (max)minheap, take union of all the L buckets hashed to from the different hash tables. Similarly, hash all the remaining elements from the minheap and take union of their respective buckets.

2. The input which has the higher gradient norm should be given a higher weight during sampling. This is done using importance sampling. To curate the elements for our next batch, we So the bucket of the element with the higher gradient will choose the highest number of elements i.e x*n, here x is the weight for that input. The remaining elements will be selected in a similar way from the remaining TopK inputs.

4. Use the newly created batch for training and keep repeating.

2 Issues

1. We assumed that the if the inputs are similar then the gradients would also follow the same behavior. If the function changes rapidly, then this assumption fails.

2. After performing the forward pass and getting the gradients, the model is already updated. We are selecting the next batch according to the inputs which have already changed the model. This may or may not be very effective.