Paper Summary

David Miller
CIS 5930: Social Network Mining

January 22, 2018

Deep Learning is a machine learning method based on learning data representations. The main goal of most deep learning systems, or neural networks, is to identify expert neurons and use them for training. Current implementations of neural networks require large matrix multiplications to determine the energy of each neuron, where higher energy can be thought of as an expert neuron. Even when matrices are sparse and clever numerical methods are implemented to compute matrix multiplications, the amount of FLOPS still remains computationally expensive. The paper aims at using a hashing approach to identify expert neurons.

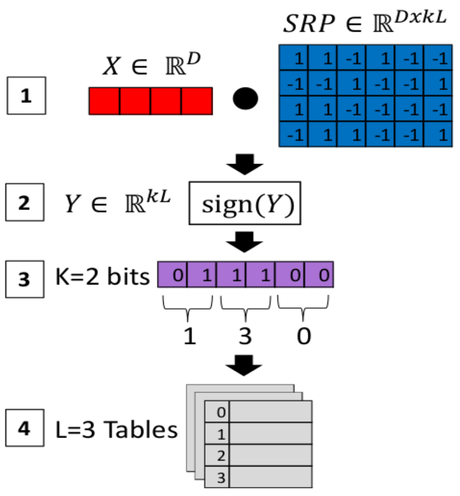

Figure 1

The first step is to take each element $x \in X \subset \mathbb{R}^D$ and take the inner product with a random signed matrix $SRP \subset \mathbb{R}^{D \times kL}$. From $x \cdot SRP = Y \subset \mathbb{R}^{kL}$ we can perform $\text{Hash}_i(Y)$ for $i = 1, ..., k$ to obtains a collection of fingerprints. These fingerprints are used to index into HashTable_j for $j = 1, ..., L$. Figure 1 illustrates the inner product between a data vector and a random SRP and the result being hashed on by the signum function to index into three HashTables. Each entry in a HashTable is called a bucket since they hold onto multiple neurons. The final step is to then union a bucket from each HashTable, where these buckets will be the expert neurons. Therefore we get that the output at each layer is $\text{Bucket}_1 \cup \cdots \cup \text{Bucket}_L$ where $\text{Bucket}_j \in \text{HashTable}_j$. However keeping multiple HashTables can be taxing on the memory so the paper uses multi-probe Local Sensitive Hashing (LSH). Rather than have L HashTables and take a single bucket from each, multi-probe LSH constructs only $L' < L$ HashTables and takes multiple buckets per HashTable. The other buckets are ones that are neighbors to that of the primary bucket, that is the buckets have neurons that are close to the energy of the primary bucket. Therefore the output now becomes $\text{Bucket}_1^1 \cup ... \cup \text{Bucket}_1^n \cup ... \cup \text{Bucket}_{L'}^1 \cup ... \cup \text{Bucket}_{L'}^n$ where Bucket_j^m is the m-th Bucket for HashTable_j. Figure 2 illustrates five inputs (data vectors that have taken the inner product with some SRP) being hashed on by some arbitrary hash function H1 and indexed into the first HashTable. At the same time, the same input data is hashed on by another arbitrary

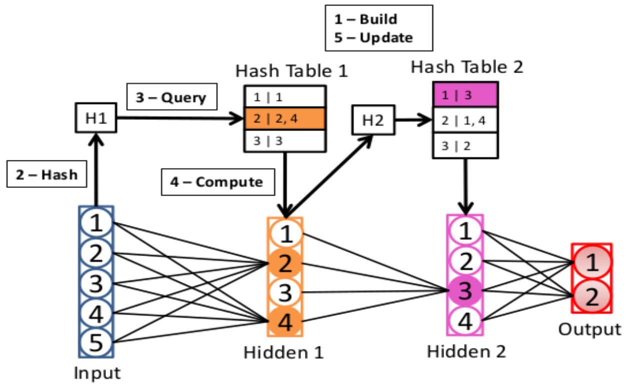

Figure 2

hash function H2 and indexed into the second hash table. The output is the union of buckets of expert neurons, in this case it is Bucket_2 from the first HashTable and Bucket_1 from the second HashTable. In terms of future work, the section provided in the paper discusses the most used technology in today's world: mobile processing. Due to the efficiency and lightweight framework of the method presented in this paper, optimization for mobile processors should not present to big of a problem. Since essentially everyone owns a phone, each person will act as a contributor to a huge mobile processor cluster. Anotehr direction that can be taken in the future is warehouse scale computing. The savings in computational time allows warehouse scale computing to perform an enormous amount of parallel deep learning tasks.

In terms of strengths of the paper, there are three that stand out to me

1. The method presented in the paper achieves an average of 99% accuracy compared to classical deep learning methods while using significantly less neurons.
2. The method presented in the paper is a lightweight framework due to it uses pointers to its data rather than storing it in memory.
3. The time complexity of the method presented in the paper is at worst $\mathcal{O}(n)$ at each layer due to the efficiency of hash tables.

There are not three strong flaws in the paper I could find but there were definitely ones that can be further analyzed

1. Due to the nature of Hash Tables there is no efficient way to identify neighbors of the primary bucket, therefore multi-probe LSH might present some computational complexity problems.
2. Statistically there is a small chance that most of the neurons can be put into the same bucket and therefore the output will have almost the same amount of neurons that was present initially.
3. Since the method is heavily reliant on statistical methods, a large data set is required to achieve accurate results.

Questions for the presenter

1. How are k, L, and buckets per table determined?
2. Is there an analysis that can be done on the choice of k, L, and buckets per table that can tell us how "good" the results will be?
3. Is there a way for the user to assign a matrix rather than some SRP, and if so does it have to be a matrix such that each entry is in $[-1, 1]$?