Optimizing sequential Information Bottleneck

Assaf Toledo
assaf.toledo@ibm.com

Elad Venezian
eladv@il.ibm.com

July 2020

Draft

1 Introduction

The sequential Information Bottleneck (sIB, [3]) is a text clustering algorithm published about 20 years ago. While sIB has shown strong results on standard benchmark datasets, such as [1], compared to the more commonly known Lloyd's K-Means [2], the run-time of the algorithm could be much slower than K-Means. Consequently, despite the better clustering analysis that it provides, sIB might have been the less preferred choice in many circumstances, especially when speed is a high priority.

This work presents an improvement to sIB run-time by calculating sIB's partition optimization efficiently in the most common sparse use-case. This reduces the computational complexity of the algorithm and effectively makes sIB much more practical for real-world use-cases.

2 Definitions and Assumptions

In the *Bag-of-Words* (BoW) model, texts are represented using vectors over a vocabulary of terms, such as unigrams. The vocabulary, sometimes referred to as *lexicon* or *dictionary*, is a corpus-level concept – an ordered list that aims to include the terms that are assumed to be meaningful for comparing texts in this corpus. When representing a text in the BoW model, a vector is used to specify the frequency of each vocabulary item in that text.

Typically, the number of unique terms found in a specific text is smaller than the vocabulary size by a large magnitude. Therefore, it is more efficient to represent texts using sparse vector representations, both in terms of memory usage and computation workload.[1] In the sparse representation, it is sufficient to hold the list of ids of vocabulary items found in the text and their frequency, rather than an array of the size of the full vocabulary in which most of the values are zero.

sIB is a centroid-based clustering algorithm, and as such it represents centroids using vectors. Since the centroid of every cluster is constructed from the set of vectors that are associated with that cluster, centroid vectors are typically dense.

Formally, Let V be the vocabulary and let $d = |V|$ be the vocabulary size. Each centroid is a dense vector of size d. Each sparse text vector v is associated with a list of ids of vocabulary items, v_{ind}, that it contains. Thus, for every text represented by a sparse vector v:

- $\forall i \in v_{ind}.v[i] > 0$
- $\forall i \notin v_{ind}.v[i] = 0$

In addition, let m indicate the average size of v_{ind}, let n be the number of samples for clustering, and let k be the number of clusters to be created by the algorithm.[2]

[1] In general, texts can exploit even all of the vocabulary items but this a rarity and we assume that on average each text uses only a small subset of the vocabulary.

[2] In practice, sparse vectors are encoded using two lists: (a) for the ids of the vocabulary items being used in the text the vector represents, and the second for the frequencies of these items in that text. Under the assumptions described above, a sparse vector is represent by $2m$ entries on average, and $2m << d$.

3 Partition Optimization

Pseudo-code of the algorithm main-loop is given in Figure 1 of [3] and outlines the sequential workflow in which sIB optimizes a given partition of the samples. Here we will focus on the calculation of the most computational heavy statement of the pseudo-coded:

$$t^{new}(x) = argmin_t d_F(x,t) \tag{1}$$

To maximize the information between the clustering analysis and the vocabulary ($I < T; Y >$), sIB employs the following distance function:

$$d_F(x,t) = (p(x) + p(t)) \cdot JS(p(y|x), p(y|t)) \tag{2}$$

where JS is a weighted *Jensen-Shannon divergence* defined with weights $pi1$ and $pi2$ as follows:

$$pi1_{x,t} = \frac{p(x)}{p(x) + p(t)} \tag{3}$$

$$pi2_{x,t} = \frac{p(t)}{p(x) + p(t)} \tag{4}$$

$$\begin{aligned}
JS(p(y|x), p(y|t)) : \quad & && (5)\\
& average = pi1 \cdot p(y|x) + pi2 \cdot p(y|t)) && (6)\\
& kl1 = KL(p(y|x), average) && (7)\\
& kl2 = KL(p(y|t)), average) && (8)\\
& \text{return } pi1 \cdot kl1 + pi2 \cdot kl2 && (9)
\end{aligned}$$

and KL is the *Kullback-Leibler divergence* defined as:

$$KL(u,v) = \sum_i u[i] \cdot log(\frac{u[i]}{v[i]}) \tag{10}$$

This means that when assigning the sample x to a new cluster, t^{new}, the new cluster is the one whose centroid, $p(y|t)$, is the closest to the vector representing x, $p(y|x)$, using the weighted Jensen-Shannon divergence. Notice that $p(y|t)$ and $p(y|x)$ are probability vectors, thus L_1 normalized. Also recall that we represent the centroid $p(y|t)$ as a dense vector and each sample $p(y|x)$ as a sparse vector. For convenience, we will use c for the vector of a *centroid*, and s for the vector of a *sample*. It follows that:

$$\forall i \notin s_{ind}. s[i] = 0 \tag{11}$$

$$\sum_{i=0}^{d} c[i] = 1 \tag{12}$$

Computation of JS

Development of kl1

$$\begin{aligned}
kl1 &= KL(s, pi1 \cdot s + pi2 \cdot c)\\
&= \sum_i s[i] \cdot log(\frac{s[i]}{(pi1 \cdot s[i] + pi2 \cdot c[i])})\\
&= \sum_{i \in s_{ind}} s[i] \cdot log(\frac{s[i]}{(pi1 \cdot s[i] + pi2 \cdot c[i])}) + \sum_{i \notin s_{ind}} s[i] \cdot log(\frac{s[i]}{(pi1 \cdot s[i] + pi2 \cdot c[i])})\\
&= \sum_{i \in s_{ind}} s[i] \cdot log(\frac{s[i]}{(pi1 \cdot s[i] + pi2 \cdot c[i])}) && \text{by (11)}
\end{aligned}$$

Development of kl2

$$
\begin{aligned}
kl2 &= KL(c, pi1 \cdot s + pi2 \cdot c)\\
&= \sum_{i=0}^{d} c[i] \cdot log(\frac{c[i]}{(pi1 \cdot s[i] + pi2 \cdot c[i])})\\
&= \sum_{i \in s_{ind}} c[i] \cdot log(\frac{c[i]}{(pi1 \cdot s[i] + pi2 \cdot c[i])}) + \sum_{i \notin s_{ind}} c[i] \cdot log(\frac{c[i]}{(pi1 \cdot s[i] + pi2 \cdot c[i])})\\
&= \sum_{i \in s_{ind}} c[i] \cdot log(\frac{c[i]}{(pi1 \cdot s[i] + pi2 \cdot c[i])}) + \sum_{i \notin s_{ind}} c[i] \cdot log(\frac{c[i]}{pi2 \cdot c[i]}) && \text{by (11)}\\
&= \sum_{i \in s_{ind}} c[i] \cdot log(\frac{c[i]}{(pi1 \cdot s[i] + pi2 \cdot c[i])}) + \sum_{i \notin s_{ind}} c[i] \cdot log(\frac{1}{pi2})\\
&= \sum_{i \in s_{ind}} c[i] \cdot log(\frac{c[i]}{(pi1 \cdot s[i] + pi2 \cdot c[i])}) + log(\frac{1}{pi2}) \cdot \sum_{i \notin s_{ind}} c[i]\\
&= \sum_{i \in s_{ind}} c[i] \cdot log(\frac{c[i]}{(pi1 \cdot s[i] + pi2 \cdot c[i])}) + log(\frac{1}{pi2}) \cdot (\sum_{i=0}^{d} c[i] - \sum_{i \in s_{ind}} c[i])\\
&= \sum_{i \in s_{ind}} c[i] \cdot log(\frac{c[i]}{(pi1 \cdot s[i] + pi2 \cdot c[i])}) + log(\frac{1}{pi2}) \cdot (1 - \sum_{i \in s_{ind}} c[i]) && \text{by (12)}
\end{aligned}
$$

3.1 Computation of $\mathbf{d}_F$

$$
\begin{aligned}
d_F(x,t) &= (p(x) + p(t)) \cdot JS(p(y|x), p(y|t))\\
&= (p(x) + p(t)) \cdot JS(p, c)\\
&= (p(x) + p(t))(pi1 \cdot kl1 + pi2 \cdot kl2)\\
&= (p(x) + p(t))(\frac{p(x)}{p(x) + p(t)} \cdot kl1 + \frac{p(t)}{p(x) + p(t)} \cdot kl2)\\
&= p(x) \cdot kl1 + p(t) \cdot kl2\\
&= p(x) \cdot \sum_{i \in s_{ind}} s[i] \cdot log(\frac{s[i]}{(pi1 \cdot s[i] + pi2 \cdot c[i])}) +\\
&\quad p(t) \cdot (\sum_{i \in s_{ind}} c[i] \cdot log(\frac{c[i]}{(pi1 \cdot s[i] + pi2 \cdot c[i])}) + log(\frac{1}{pi2}) \cdot (1 - \sum_{i \in s_{ind}} c[i]))
\end{aligned}
$$

3.2 Computation of $t^{new}(x)$

$$
\begin{aligned}
t^{new}(x) &= argmin_t d_F(x,t)\\
&= argmin_t (p(x) \cdot \sum_{i \in s_{ind}} s[i] \cdot log(\frac{s[i]}{(pi1 \cdot s[i] + pi2 \cdot c[i])}) +\\
&\qquad p(t) \cdot (\sum_{i \in s_{ind}} c[i] \cdot log(\frac{c[i]}{(pi1 \cdot s[i] + pi2 \cdot c[i])}) + log(\frac{1}{pi2}) \cdot (1 - \sum_{i \in s_{ind}} c[i])))\\
&= argmin_t ((p(x) \cdot \sum_{i \in s_{ind}} s[i] \cdot log(s[i])) + (p(x) \cdot \sum_{i \in s_{ind}} s[i] \cdot log(\frac{1}{(pi1 \cdot s[i] + pi2 \cdot c[i])})) +\\
&\qquad p(t) \cdot (\sum_{i \in s_{ind}} c[i] \cdot log(\frac{c[i]}{(pi1 \cdot s[i] + pi2 \cdot c[i])}) + log(\frac{1}{pi2}) \cdot (1 - \sum_{i \in s_{ind}} c[i])))
\end{aligned}
$$

Since the expression:

$$p(x) \cdot \sum_{i \in s_{ind}} s[i] \cdot log(s[i]) \tag{13}$$

is independent of t, it does not affect the computation of $argmin_t$. Thus, we get:

$$\begin{aligned}
t^{new}(x) = argmin_t(p(x) \cdot \sum_{i \in s_{ind}} s[i] \cdot log(\frac{1}{(pi1 \cdot s[i] + pi2 \cdot c[i])}) + \\
p(t) \cdot (\sum_{i \in s_{ind}} c[i] \cdot log(\frac{c[i]}{(pi1 \cdot s[i] + pi2 \cdot c[i])}) + log(\frac{1}{pi2}) \cdot (1 - \sum_{i \in s_{ind}} c[i]))) \\
= argmin_t(p(x) \cdot \sum_{i \in s_{ind}} s[i] \cdot log(\frac{1}{(pi1 \cdot s[i] + pi2 \cdot c[i])}) + \\
p(t) \cdot (\sum_{i \in s_{ind}} c[i] \cdot (log(c[i]) + log(\frac{1}{(pi1 \cdot s[i] + pi2 \cdot c[i])})) + log(\frac{1}{pi2}) \cdot (1 - \sum_{i \in s_{ind}} c[i]))) \\
= argmin_t(\sum_{i \in s_{ind}} (p(x) \cdot s[i] + p(t) \cdot c[i]) \cdot log(\frac{1}{(pi1 \cdot s[i] + pi2 \cdot c[i])}) + \\
p(t) \cdot (\sum_{i \in s_{ind}} c[i] \cdot log(c[i]) + log(\frac{1}{pi2}) \cdot (1 - \sum_{i \in s_{ind}} c[i]))) \\
= argmin_t(\sum_{i \in s_{ind}} (p(x) \cdot s[i] + p(t) \cdot c[i]) \cdot log(\frac{1}{(\frac{p(x)}{p(x)+p(t)} \cdot s[i] + \frac{p(t)}{p(x)+p(t)} \cdot c[i])}) + \\
p(t) \cdot (\sum_{i \in s_{ind}} c[i] \cdot log(c[i]) + log(\frac{1}{\frac{p(t)}{p(x)+p(t)}}) \cdot (1 - \sum_{i \in s_{ind}} c[i]))) \\
= argmin_t(\sum_{i \in s_{ind}} (p(x) \cdot s[i] + p(t) \cdot c[i]) \cdot log(\frac{p(x) + p(t)}{(p(x) \cdot s[i] + p(t) \cdot c[i])}) + \\
p(t) \cdot (\sum_{i \in s_{ind}} c[i] \cdot log(c[i]) + log(\frac{p(x) + p(t)}{p(t)}) \cdot (1 - \sum_{i \in s_{ind}} c[i])))
\end{aligned}$$

Computational Complexity

We get that the assignment of a sample to a centroid can be calculated in $O(m)$ operations. Given that there are k centroids, the overall complexity of $t^{new}(x)$ is $O(km)$

4 Summary

We have shown an optimization to the assignment of a sample to a new centroid. This is achieved by taking advantage of the sparse vector representation of samples. Since $m << d$, the obtained complexity of $O(km)$, is substantially better than the complexity of $O(kd)$ that is obtainable with dense representations.

References

[1] Ken Lang. Newsweeder: Learning to filter netnews. In *in Proceedings of the 12th International Machine Learning Conference (ML95)*, 1995.

[2] Stuart P. Lloyd. Least squares quantization in pcm. *IEEE Trans. Inf. Theor.*, 28(2):129–137, September 1982.

[3] Noam Slonim, Nir Friedman, and Naftali Tishby. Unsupervised document classification using sequential information maximization. In *Proceedings of the 25th Annual International ACM SIGIR Conference on*

Research and Development in Information Retrieval, SIGIR '02, page 129–136, New York, NY, USA, 2002. Association for Computing Machinery.