Categorical Feature Compression via Submodular Optimization

MohammadHossein Bateni [* 1]   **Lin Chen** [* 1 2]   **Hossein Esfandiari** [* 1]   **Thomas Fu** [* 1]   **Vahab S. Mirrokni** [* 1]
Afshin Rostamizadeh [* 1]

Abstract

In the era of big data, learning from categorical features with very large vocabularies (e.g., 28 million for the Criteo click prediction dataset) has become a practical challenge for machine learning researchers and practitioners. We design a highly-scalable vocabulary compression algorithm that seeks to maximize the mutual information between the compressed categorical feature and the target binary labels and we furthermore show that its solution is guaranteed to be within a $1 - 1/e \approx 63\%$ factor of the global optimal solution. To achieve this, we introduce a novel reparametrization of the mutual information objective, which we prove is submodular, and design a data structure to query the submodular function in amortized $O(\log n)$ time (where n is the input vocabulary size). Our complete algorithm is shown to operate in $O(n \log n)$ time. Additionally, we design a distributed implementation in which the query data structure is decomposed across $O(k)$ machines such that each machine only requires $O(\frac{n}{k})$ space, while still preserving the approximation guarantee and using only logarithmic rounds of computation. We also provide analysis of simple alternative heuristic compression methods to demonstrate they cannot achieve any approximation guarantee. Using the large-scale Criteo learning task, we demonstrate better performance in retaining mutual information and also verify competitive learning performance compared to other baseline methods.

1. Introduction

In modern large scale machine learning tasks, the presence of categorical features with extremely large vocabularies

---
*Equal contribution [1]Google, New York, NY, USA [2]Department of Electrical Engineering, Yale University, New Haven, CT, USA. Correspondence to: Lin Chen <lin.chen@yale.edu>.

is a standard occurrence. For example, in tasks such as product recommendation and click-through rate prediction, categorical variables corresponding to inventory id, webpage identifier, or other such high cardinality values, can easily contain anywhere from hundreds of thousands to tens of millions of unique values. The size of machine learning models generally grows at least linearly with the vocabulary size and, thus, the memory required to serve the model, the training and inference cost, as well as the risk of overfitting become an issue with very large vocabularies. In the particular case of neural networks model, one generally uses an embedding layer to consume categorical inputs. The number of parameters in the embedding layer is $O(nh)$, where n is the size of the vocabulary and h is the number of units in the first hidden layer.

To give a concrete example, the Criteo click prediction benchmark has about 28 million categorical feature values (CriteoLabs, 2014), thus resulting in an embedding layer more than 1 billion parameters for a modestly sized first hidden layer. Note, this number dwarfs the number of parameters found in the remainder of the neural network. Again, to give a concrete example, even assuming a very deep fully connected network of depth 10^2 with hidden layers of size 10^3, we have $(10^3 \times 10^3)10^2 = 10^8$ parameters in the hidden network – still an order of magnitude smaller than the embedding layer alone. This motivates the problem of compressing the vocabulary into a smaller size while still retaining as much information as possible.

In this work, we model the compression task by considering the problem of maximizing the mutual information between the compressed version of the categorical features and the target label. We first observe a connection between this problem and the quantization problem for discrete memoryless channels, and note a polynomial-time algorithm for the problem (Kurkoski & Yagi, 2014; Iwata & Ozawa, 2014). The resulting algorithm, however, is based on solving a quadratic-time dynamic program, and is not scalable. Our main goal in this paper is to develop a scalable and distributed algorithm with a guaranteed approximation factor. We achieve this goal by developing a novel connection to submodular optimization. Although in some settings, entropy-based set functions are known to be submodular, this is not the case for the mutual information objective

we consider (mutual information with respect to the target labels). Our main insight is in proving the submodularity of a particular transformation of the mutual information objective, which still allows us to provide an approximation guarantee on the quality of the solution with respect to the original objective. We also provide a data structure that allows us to query this newly defined submodular function in amortized logarithmic time. This logarithmic-time implementation of the submodular oracle empowers us to incorporate the fastest known algorithm for submodular maximization (Mirzasoleiman et al., 2015), which leads us to a sequential quasi-linear-time $(1 - 1/e - \epsilon)$-approximation algorithm for binary vocabulary compression. Next, we provide a distributed implementation for binary vocabulary compression. Previous distributed algorithms for submodular maximization assume a direct access the query oracle on every machine (e.g., see (Barbosa et al., 2015; Mirrokni & Zadimoghaddam, 2015; Mirzasoleiman et al., 2013)). However, the query oracle itself requires $O(n)$ space, which may be restrictive in the large scale setting. In this work, we provide a truly distributed implementation of the submodular maximization algorithm of (Badanidiyuru & Vondrák, 2014) (or similarly (Kumar et al., 2015)) for our application by distributing the query oracle. In this distributed implementation we manage to decompose the query oracle across $O(k)$ machines such that each machine only requires $O(\frac{n}{k})$ space to store the partial query oracle. As a result, we successfully provide a distributed $(1 - 1/e - \epsilon)$-approximation algorithm for vocabulary compression in logarithmic rounds of computation. Our structural results for submodularity of this new set function is the main technical contribution of this paper, and can also be of independent interest in other settings that seek to maximize mutual information.

We also study a number of heuristic and baseline algorithms for the problem of maximizing mutual information, and show that they do not achieve a guaranteed approximation for the problem. Furthermore, we study the empirical performance of our algorithms on two fronts: First, we show the effectiveness of our greedy scalable approximation algorithm for maximizing mutual information. Our study confirms that this algorithm not only achieves a theoretical guarantee, but also it beats the heuristic algorithms for maximizing mutual information. Finally, we examine the performance of this algorithm on the vocabulary compression problem itself, and confirm the effectiveness of the algorithm in producing a high-quality solution for vocabulary compression large scale learning tasks.

Organization. In the remainder of this section we review related previous works and introduce the problem formally along with appropriate notation. Then in Section 2, we introduce the novel compression algorithm and corresponding theoretical guarantees as well as analysis of some basic heuristic baselines. In Section 3, we present our empirical

evaluation of optimizing the mutual information objective as well as an end-to-end learning task.

1.1. Related Work

Feature Clustering: The use of vocabulary compression has been studied previously, especially in text classification applications where it is commonly known as feature (or word) clustering. In particular, Baker & McCallum (1998) and Slonim & Tishby (2001) both propose agglomerative clustering algorithms, which start with singleton clusters that are iteratively merged using a Jenson-Shannon divergence based function to measure similarity between clusters, until the desired number of clusters is found. Both algorithms are greedy in nature and do not provide any guarantee with respect to a global objective. In Dhillon et al. (2003), the authors introduce an algorithm that empirically performs better than the aforementioned methods and that also seeks to optimize the same global mutual information objective that is analyzed in this work. Their proposed iterative algorithm is guaranteed to improve the objective at each iteration and arrive at a local minimum, however, no guarantee with respect to the global optimum is provided. Furthermore, each iteration of the algorithm requires $O(mn)$ time (where m is the size of the compressed vocabulary) and the number of iterations is only guaranteed to be finite (but potentially exponential). Later in this work, we compare the empirical performance of this algorithm with our proposed method.
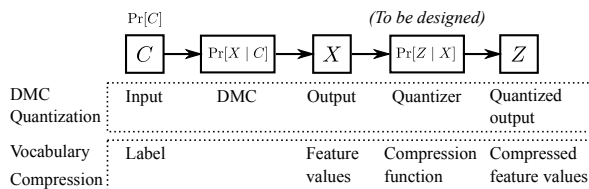

Figure 1. Translation of terminologies of the DMC quantizer design problem and the feature compression problem.

Compression in Discrete Memoryless Channels: An area from information theory that is closely related to our vocabulary compression problem, and which our algorithm draws inspiration from, is compression in a discrete memoryless channels (DMC) (Cicalese et al., 2018; Zhang & Kurkoski, 2016; Iwata & Ozawa, 2014). In this problem, we assume there is a DMC which (in machine learning terminology) receive a class label and produces a corresponding categorical feature value drawn according to a fixed underlying distribution. The goal is to design a *quantizer* that maps the space of categorical features in lower cardinatility set, while preserving as much of the mutual information between the class label and newly constructed vocabulary. In Figure 1, we present a diagram that illustrates the DMC quantization problem and vocabulary compression problem as well as the translation of terminologies of these two problems. The

results of Kurkoski & Yagi (2014) are of particular interest, as they show a cubic-time dynamic programming based algorithm is able to provide an optimal solution in the case of binary labels. Following this work, Iwata & Ozawa (2014) improve the computational complexity of this approach to quadratic time using the SMAWK algorithm (Aggarwal et al., 1987). Such algorithms are useful in the smaller scale regime, however, the use of a cubic- or even quadratic-time algorithm will be infeasible for our massive vocabulary size use cases. Finally, Mumey & Gedeon (2003) shows that in the general case of greater than two class labels, finding the optimal compression is NP-complete. In this work, we will be focusing on the binary label setting.

Feature Selection: A related method for dealing with very large vocabularies is to do feature selection, in which we simply select a subset of the vocabulary values and remove the rest (see Guyon & Elisseeff (2003) and the many references therein). One can view this approach as a special case of vocabulary compression, where we are restricted to only singleton "clusters". Restricting the problem by selecting a subset of the vocabulary may have some benefits, such as potentially simplifing the optimization problem and the use of a simple filter to transform data at inference time. However, the obvious downside to this restriction is the loss of information and potentially poorer learning performance (see introduction of Jiang et al. (2011)). In this work we focus on the more general vocabulary compression setting.

Other Feature Extraction Approaches: Clustering features in order to compress a vocabulary is only one approach to lower dimensional feature extraction. There are of course many classical approaches to feature extraction (see Chapter 15 of Mohri et al. (2018)), such as learning linear projections (e.g., Principle Component Analysis, Linear Discriminant Analysis) or non-linear transformations (e.g., Locally Linear Embeddings, ISOMAP, Laplacian Eigenmaps). However, these classical methods generally incur more than quasilinear computational cost, for both learning and the application the transformation, and are not feasible for our setting.

1.2. Notation

In the vocabulary compression problem we are given a correlated pair of random variables X (a categorical feature) and C (a label), where $X \in \{1, 2, \ldots, n\}$ and $C \in \{0, 1\}$. We aim to define a random variable $Z \in \{1, 2, \ldots, m\}$ as a function of X that maximizes the mutual information with the label C, i.e., $I(Z; C)$, where for general random variables A and B taking values in $\mathcal{A}$ and $\mathcal{B}$, respectively,

$$I(A; B) = \sum_{A \in \mathcal{A}} \sum_{B \in \mathcal{B}} \Pr[A, B] \log\left(\frac{\Pr[A, B]}{\Pr[A]\Pr[B]}\right). \quad (1)$$

Note that Z is a function of X and hence we have $I(Z; C) \leq I(X; C)$. If we let $m \geq n$, $Z = X$ maximizes the mutual

information $I(Z; C)$. We are interested in the nontrivial case of $m \ll n$. Intuitively, we are compressing the vocabulary of feature X from size n to a smaller size m, while preserving the maximum amount of information about C.

2. Algorithm and Analysis

In this section, we first show how to transform the original objective into a set function and then prove that this set function is in fact submodular. Next, we describe the components of a quasi-linear and parallelizable algorithm to optimize the objective. Finally, we consider a few simple intuitive baselines and show that they may create features that fail to capture any mutual information with the label.

2.1. Objective Transformation

Without loss of generality assume $\Pr[C = 0 | X = i]$ for $i \in \{1, \ldots, n\}$ is sorted in increasing order. Once the feature values are sorted in this order, Lemma 3 of Kurkoski & Yagi (2014) crucially shows that in the optimum solution each value of Z corresponds to a consecutive subsequence of $\{1, \ldots, n\}$ — this is a significant insight that we take from the quantization for DMC literature. Thus, we will cluster consecutive feature values into m clusters, with each cluster corresponding to a value in the compressed vocabulary of Z. Formally, define a function $F(S) : 2^{\{1, \ldots, n-1\}} \rightarrow \mathbb{R}$ as follows: Let $S = \{s_1, \ldots, s_{m-1}\}$, and assume $s_1 < s_2 < \cdots < s_{m-1}$. For simplicity, and without any loss in quality, we set $s_0 = 0$ and $s_m = n$. Let Z be a random variable constructed from X that has value i, if and only if $s_{i-1} < X \leq s_i$. We define $F(S) = I(Z; C)$. Notice that we have

$$\max_{S \subseteq \{2 \ldots n-1\} : |S| = m-1} F(S) = \max_{Z} I(Z; C),$$

where Z is a function of X with vocabulary size m. The non-negativity of mutual information implies that the function $F(S) = I(Z; C)$ is always non-negative (Cover & Thomas, 2006, p. 28). The monotonicity is equivalent to $I(Z_1; C) \leq I(Z_2; C)$ for any $S_1 \subseteq S_2 \subseteq \{1, \ldots, n-1\}$, where Z_1 and Z_2 are the random variables constructed from S_1 and S_2, respectively. Since S_2 represents a subdivision of S_1, Z_1 is a function of Z_2. By the data-processing inequality, we have $I(Z_1; C) \leq I(Z_2; C)$ (Cover & Thomas, 2006, p. 34). In the following section, we show that the function $F(S)$ is in fact submodular.

2.2. Submodularity of $F(S)$

For a set $S \subseteq \{1, \ldots, n-1\}$ and a number $s \in \{1, \ldots, n-1\} \setminus S$ we define $\Delta_s F(S) = F(S \cup \{s\}) - F(S)$. Let s' be the item right before s when we sort $S \cup \{s\}$. Note that, the items that are mapped to s' by $F(S)$ are either mapped to s' or s by $F(S \cup \{s\})$. We first observe the following useful

technical lemma (the proof of all lemmas can be found in the supplement).

Lemma 1. *Define the quantities* $p = \Pr[Z = s']$, $q = \Pr[Z = s]$, $\alpha = \Pr[C = 0|Z = s']$ *and* $\beta = \Pr[C = 0|Z = s]$, *then the following equality holds*

$$\Delta_s F(S) = pf(\alpha) + qf(\beta) - (p+q)f\left(\frac{p\alpha + q\beta}{p+q}\right), \quad (2)$$

where $f(\cdot)$ *the following convex function over* $(0,1)$:

$$f(t) = t \log \frac{t}{\Pr[C = 0]} + (1 - t) \log \frac{1 - t}{\Pr[C = 1]}. \quad (3)$$

Next, we provide several inequalities that precisely analyze expressions of the same form as (2) with various values of α, β, p and q.

Lemma 2. *Pick* $\alpha \leq \beta \leq \gamma \in \mathbb{R}$, *and* $p \in [0,1]$. *Let* $q = 1 - p$ *and let* f *be an arbitrary convex function. We have*

$$pf(\alpha) + qf(\beta) - f(p\alpha + q\beta) \leq pf(\alpha) + qf(\gamma) - f(p\alpha + q\gamma).$$

Replacing p and q in Lemma 2 with $\frac{p}{p+q}$ and $\frac{q}{p+q}$ and multiplying both sides by $p + q$ implies the following corollary.

Corollary 3. *Pick* $\alpha \leq \beta \leq \gamma \in \mathbb{R}$, *and* $p, q \in \mathbb{R}^+$. *Let* f *be an arbitrary convex function. We have*

$$pf(\alpha) + qf(\beta) - (p+q)f\left(\frac{p\alpha + q\beta}{p+q}\right)$$
$$\leq pf(\alpha) + qf(\gamma) - (p+q)f\left(\frac{p\alpha + q\gamma}{p+q}\right).$$

Similarly, we have the following corollary (simply by looking at $f(-x)$ instead of $f(x)$).

Corollary 4. *Pick* $\gamma \leq \alpha \leq \beta \in \mathbb{R}$, *and* $p, q \in \mathbb{R}^+$. *Let* f *be an arbitrary convex function. We have*

$$pf(\alpha) + qf(\beta) - (p+q)f\left(\frac{p\alpha + q\beta}{p+q}\right)$$
$$\leq pf(\gamma) + qf(\beta) - (p+q)f\left(\frac{p\gamma + q\beta}{p+q}\right).$$

We require one final lemma before proceeding to the main theorem.

Lemma 5. *Pick* $\alpha, \beta \in \mathbb{R}$, *and* $p, q, q' \in (0,1]$ *such that* $q < q'$. *Let* f *be an arbitrary convex function. We have*

$$pf(\alpha) + qf(\beta) - (p+q)f\left(\frac{p\alpha + q\beta}{p+q}\right)$$
$$\leq pf(\alpha) + q'f(\beta) - (p+q')f\left(\frac{p\alpha + q'\beta}{p+q'}\right).$$

$$\begin{array}{c}\overline{Z_1 = s'_1} \qquad \overline{Z_1 = s}\end{array}$$
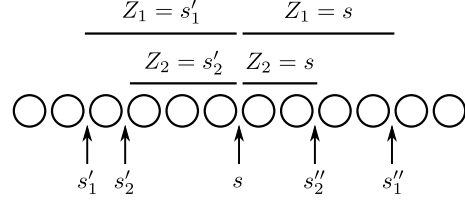

Figure 2. Illustration of boundaries used in proof of Theorem 6.

Theorem 6 (Submodularity). *For any pair of sets* $S_1 \subseteq S_2 \subseteq \{1, \ldots, n-1\}$ *and any* $s \in \{1, \ldots, n-1\} \setminus S_2$ *we have* $\Delta_s F(S_1) \geq \Delta_s F(S_2)$.

Proof. Let s'_1 and s''_1 be the items right before and right after s when we sort $S_1 \cup \{s\}$. Also, let Z_1 and Z'_1 be the random variables corresponding to $F(S_1 \cup \{s\})$ and $F(S_1)$ respectively. Similarly let s'_2 and s''_2 be items right before and right after s when we sort $S_2 \cup \{s\}$, and let Z_2 and Z'_2 be the random variables corresponding to $F(S_2 \cup \{s\})$ and $F(S_2)$ respectively.

Let us set $p_1 = \Pr[Z_1 = s'_1]$, $q_1 = \Pr[Z_1 = s]$, $\alpha_1 = \Pr[C = 0|Z_1 = s'_1]$ and $\beta_1 = \Pr[C = 0|Z_1 = s]$. Similarly let us set $p_2 = \Pr[Z_2 = s'_2]$, $q_2 = \Pr[Z_2 = s]$, $\alpha_2 = \Pr[C = 0|Z_2 = s'_2]$ and $\beta_2 = \Pr[C = 0|Z_2 = s]$. Note that since $S_1 \subseteq S_2$, we have $s'_1, s''_1 \in S_2$ and hence we have $s'_2 \geq s'_1$ and $s''_2 \leq s''_1$ (see Figure 2). Therefore, we have following set of inequalities

$$p_2 = \Pr[Z_2 = s'_2] \leq \Pr[Z_1 = s'_1] = p_1, \quad (4)$$
$$q_2 = \Pr[Z_2 = s] \leq \Pr[Z_1 = s] = q_1. \quad (5)$$

Since in the definition of $F(\cdot)$ the elements are ordered by $\Pr[C = 0|X = x]$, we have the following set of inequalities

$$\alpha_1 = \Pr[C=0|Z_1=s'_1] \leq \Pr[C=0|Z_1=s] = \beta_1, \quad (6)$$
$$\alpha_2 = \Pr[C=0|Z_2=s'_2] \leq \Pr[C=0|Z_2=s] = \beta_2, \quad (7)$$
$$\alpha_1 = \Pr[C=0|Z_1=s'_1] \leq \Pr[C=0|Z_2=s'_2] = \alpha_2, \quad (8)$$
$$\beta_2 = \Pr[C=0|Z_2=s] \leq \Pr[C=0|Z_1=s] = \beta_1. \quad (9)$$

Finally, we have

$$\Delta_s F(S_2)$$
$$\overset{(a)}{=} p_2 f(\alpha_2) + q_2 f(\beta_2) - (p_2 + q_2)f\left(\frac{p_2\alpha_2 + q_2\beta_2}{p_2 + q_2}\right)$$
$$\overset{(b)}{\leq} p_2 f(\alpha_2) + q_2 f(\beta_1) - (p_2 + q_2)f\left(\frac{p_2\alpha_2 + q_2\beta_1}{p_2 + q_2}\right)$$
$$\overset{(c)}{\leq} p_2 f(\alpha_1) + q_2 f(\beta_1) - (p_2 + q_2)f\left(\frac{p_2\alpha_1 + q_2\beta_1}{p_2 + q_2}\right)$$
$$\overset{(d)}{\leq} p_1 f(\alpha_1) + q_2 f(\beta_1) - (p_1 + q_2)f\left(\frac{p_1\alpha_1 + q_2\beta_1}{p_1 + q_2}\right)$$
$$\overset{(e)}{\leq} p_1 f(\alpha_1) + q_1 f(\beta_1) - (p_1 + q_1)f\left(\frac{p_1\alpha_1 + q_1\beta_1}{p_1 + q_1}\right)$$
$$\overset{(f)}{=} \Delta_s F(S_1),$$

where (a) and (f) follow from equality 2, (b) follows from Corollary 3 and inequalities (9) and (7), (c) follows from Corollary 4 and inequalities (8) and (6), (d) follows from Lemma 5 and inequality (4), and (e) follows from Lemma 5 and inequality (5). This completes the proof. $\square$

2.3. Submodular Optimization Algorithms

Given that we have shown $F(\cdot)$ is submodular, we now show two approaches to optimization: a single machine algorithm that runs in time $O(n \log n)$ as well as an algorithm which allows the input to be processed in a distributed fashion, at the cost of an additional logarithmic factor in running time.

Single Machine Algorithm: We will make use of a $1 - 1/e - \epsilon$ approximation algorithm for submodular maximization that makes only $O(n)$ queries to $\Delta_s F(S)$ (Mirzasoleiman et al., 2015). First, fix an arbitrary small constant ϵ (this appears as a loss in the approximation factor as well as in the running time). The algorithm starts with an empty solution set and then proceeds in m iterations where, in each iteration, we sample a set of $\frac{n \log 1/\epsilon}{m}$ elements uniformly at random from the elements that have not been added to the solution so far and then add the sampled element with maximum marginal increase to the solution.

In general, we may expect that computing $\Delta_s F(S)$ requires at least $\Omega(|S|)$ time, which might be as large as m. However, we note that the algorithm of (Mirzasoleiman et al., 2015) (similar to most other algorithms for submodular maximization) only queries $\Delta_s F(S)$ for incrementally growing subsets of the final solution S. In that case, we can compute each incremental value of $\Delta_s F(S)$ in logarithmic time using a data structure that costs $O(n \log n)$ time to construct (see Algorithm 1). By using this query oracle, we do not require communicating the whole set S for every query. Moreover, we use a red-black tree to maintain S, and hence we can search for neighbors (s' and s'') in logarithmic time. Thus, combining the submodular maximization algorithm that requires only a linear number of queries with the logarithmic time query oracle implies the following theorem.

Theorem 7. *For any arbitrary small $\epsilon > 0$, there exists a $(1 - 1/e - \epsilon)$-approximation algorithm for vocabulary compression that runs in $O(n \log n)$ time.*

Distributed Algorithm: Again, fix an arbitrary small number $\epsilon > 0$ (for simplicity assume ϵk and $\frac{n}{\epsilon k}$ are integers). In this distributed implementation we use ϵk machines, requires $O(\frac{n}{\epsilon k})$ space per machine, and uses a logarithmic number of rounds of computation.

To define our distributed algorithm we start with the (non-distributed) submodular maximization algorithm of (Badanidiyuru & Vondrák, 2014), which provides a $1 - 1/e - \epsilon$ approximate solution using $O(n \log n)$ queries to the submodular function oracle. The algorithm works by defining a

---

Algorithm 1 Data structure to compute $\Delta_s F(S)$

Procedure: Initialization
Input: Sorted list of probabilities $\Pr[C = 0 | X = x_i]$ and probabilities $\Pr[X = x_i]$.

1: Initiate a red-black tree data structure S.
2: Insert 0 and n into S.
3: $p_{<0} \leftarrow 0$
4: $p_{C=0|<0} \leftarrow 0$
5: **for** $i = 1$ to n **do**
6:      $p_{<i} \leftarrow p_{<i-1} + \Pr[x = x_i]$
7:      $p_{C=0|<i} \leftarrow \frac{p_{C=0|<i-1} \times p_{<i-1} + \Pr[C=0|X=x_i] \times \Pr[x=x_i]}{p_{<i}}$.
8: **end for**

Procedure: Query $\Delta_s F(S)$
Input: A number $s \in \{1, \ldots, n-1\} \backslash S$

1: $s' \leftarrow$ largest element smaller than s in S.
2: $s'' \leftarrow$ smallest element larger than s in S.
3: $p = p_{<s} - p_{<s'}$
4: $q = p_{<s''} - p_{<s}$
5: $\alpha = \frac{p_{C=0|<s} \times p_{<s} - p_{C=0|<s'} \times p_{<s'}}{p}$
6: $\beta = \frac{p_{C=0|<s'} \times p_{<s''} - p_{C=0|<s} \times p_{<s}}{q}$
7: Return $pf(\alpha) + qf(\beta) - (p+q)f\left(\frac{p\alpha + q\beta}{p+q}\right)$ where $f(\cdot)$ is defined by equation (3).

Procedure: Insert s **to** S
Input: A number $s \in \{1, \ldots, n-1\} \backslash S$

1: Insert s into S.

---

decreasing sequence of thresholds $w_0, w_1, \ldots, w_{\log_{\frac{1}{1-\epsilon}}(n)}$, where w_0 is the maximum marginal increase of a single element, and $w_i = (1 - \epsilon)^i w_0$. The algorithm proceeds in $\log_{\frac{1}{1-\epsilon}}(n)$ rounds, where in round i the algorithm iterates over all elements and inserts an element s into the solution set S if $\Delta_s F(S) \geq w_i$. The algorithm stops once it has selected k elements or if it finishes $\log_{\frac{1}{1-\epsilon}}(n)$ rounds, whichever comes first. As usual, this algorithm only queries $\Delta_s F(S)$ for incrementally growing subsets of the final solution S, and hence we can use Algorithm 1 to solve vocabulary compression in $O(n \log^2 n)$ time.

Now, we show how to distribute this computation across multiple machines. First, for all $j \in \{1, \ldots, \epsilon k - 1\}$, we select the $(j \frac{n}{\epsilon k})$-th element and add it to the solution set S. This decomposes the elements into ϵk continuous subsets of elements, each of size $\frac{n}{\epsilon k}$, and each of which we assign to one machine. Note that $\Delta_s F(S)$ only depends on the previous item and next item of s in S and, due to the way that we created the initial solution set S and decomposed the input elements, the previous item and next item of s are always both assigned to the same machine as s. Hence each machine can compute $\Delta_s F(S)$ locally. However, we assigned the first $\epsilon k - 1$ to the solution set blindly and their marginal gain may be very small. Intuitively, we are

potentially throwing away some of our selection budget for the ease of computation. Next we show that by forcing these elements into the solution we do not lose more than ϵ on the approximation factor.

First of all, notice that if we force a subset of the element to be included in the solution, the objective function is still submodular over the remaining elements. That is, the marginal impact of an element s (i.e., $\Delta_s F(S)$) shrinks as S grows. Next we show that if we force $\epsilon k - 1$ elements into the solution, it does not decrease the value of the optimum solution by more than a $(1 - \epsilon)$ factor. This means that if we provide a $(1 - 1/e - \epsilon)$-approximation to the new optimum solution, it is a $(1 - \epsilon) \times (1 - 1/e - \epsilon) \leq (1 - 1/e - 2\epsilon)$ approximate solution to the original optimum.

Let S^* be a solution of size k that maximizes $F(\cdot)$. Decompose S^* into $\frac{1}{\epsilon}$ subsets of size ϵk. Note that by submodularity the value of $F(S^*)$ is more than the sum of the marginal impacts of each $\frac{1}{\epsilon}$ subset (given the remainder of the subsets). Therefore, by the pigeonhole principle, the marginal impact of one of these subsets of S^* is at most $\epsilon F(S^*)$. If we remove this subset from S^* and add the $\epsilon k - 1$ forced elements, we find a solution of size (at most) k that contains all of the forced elements and has value at least $(1 - \epsilon)F(S^*)$ as desired. Hence, by forcing these initial $\epsilon k - 1$ elements to be in the solution we lose only an ϵ fraction on the approximation factor.

Now, to implement the algorithm of (Badanidiyuru & Vondrák, 2014), in iteration i, each machine independently finds and inserts all of its elements with marginal increase more than w_i. If the number of selected elements exceeds k, we remove the last few elements to have exactly k elements in the solution. This implies the following theorem.

Theorem 8. *For any arbitrary small $\epsilon > 0$, there exists a $(1 - 1/e - \epsilon)$-approximation $(\log n)$-round distributed algorithm for vocabulary compression with $O(\frac{n}{k})$ space per machine and $O(n \log^2 n)$ total work.*

2.4. Heuristic Algorithms

In this subsection we review a couple of heuristics that can serve as simple alternatives to the algorithm we suggest and show that they can, in fact, fail entirely for some inputs. We also provide an empirical comparison to these, as well as the algorithm of Dhillon et al. (2003), in Section 3.

Bucketing Algorithm: This algorithm splits the range of probabilities $[0, 1]$ into k equal size intervals $[0, 1/k), [1/k, 2/k), \ldots, [(k-1)/k, 1]$. Then it uses these intervals (or buckets) to form the compressed vocabulary. Specifically, each interval represents all elements i such that $\Pr[C = 0|X = i] \in [(j-1)/k, j/k)$. Note that there exists a set S^b that such that $F(S^b)$ correspond to the mutual information of the outcome of the bucketing algorithm and the la-

bels. First we show that it is possible to give an upper bound on the mutual information loss, i.e., $I(X; C) - F(S^b)$.

Theorem 9. *Let Z be the random variable provided by the bucketing algorithm. The total mutual information loss of the bucketing algorithm is bounded as follows: $I(X; C) - I(Z; C) \leq \Delta_{\max}$, where $\Delta_{\max} = \max_j \left(\max_{r \in [(j-1)/k, j/k)} f(r) - \min_{r \in [\frac{j-1}{k}, \frac{j}{k})} f(r) \right)$ and $f(\cdot)$ is defined in equation (3).*

Proof. Note that as we showed in Subsection 2.2 we have

$$
\begin{aligned}
F(S) &= \sum_{z \sim Z} \Pr[Z = z] f(\Pr[C = 0|Z = z]) \\
&= \sum_{z \sim Z} \Pr[Z = z] f\Big(E_{x \in z}\big[\Pr[C = 0|X = x] \big] \Big).
\end{aligned}
$$
(10)

On the other hand we have

$$
\begin{aligned}
I(X; C) &= \sum_{x \sim X} \Pr[X = x] f(\Pr[C = 0|X = x]) \\
&= \sum_{z \sim Z} \sum_{x \in z} \Pr[X = x] f(\Pr[C = 0|X = x]) \\
&= \sum_{z \sim Z} \Pr[Z = z] \sum_{x \in z} \frac{\Pr[X = x]}{\Pr[Z = z]} f(\Pr[C = 0|X = x]) \\
&= \sum_{z \sim Z} \Pr[Z = z] E_{x \in z}\Big[f\big(\Pr[C = 0|X = x] \big) \Big].
\end{aligned}
$$
(11)

Let j be the index of the interval corresponding to z. Then, by convexity of $f(\cdot)$, we have $E_{x \in z}\big[f\big(\Pr[C = 0|X = x] \big) \big] \leq \max_{r \in [\frac{j-1}{k}, \frac{j}{k})} f(r)$ and $f\big(E_{x \in z}\big[\Pr[C = 0|X = x] \big] \big) \geq \min_{r \in [\frac{j-1}{k}, \frac{j}{k})} f(r)$. Therefore we have

$$
\begin{aligned}
&E_{x \in z}\Big[f\big(\Pr[C = 0|X = x] \big) \Big] - f\Big(E_{x \in z}\big[\Pr[C = 0|X = x] \big] \Big) \\
&\leq \max_{r \in [(j-1)/k, j/k)} f(r) - \min_{r \in [(j-1)/k, j/k)} f(r) \leq \Delta_{\max}.
\end{aligned}
$$

This together with Equations (10) and (11) show that $I(X; C) - F(S) \leq \sum_{z \sim Z} \Pr[Z = z] \Delta_{\max} = \Delta_{\max}$ and completes the theorem. $\qquad\square$

The above theorem states that the information loss of the bucketing algorithm is no more than how much $f(\cdot)$ changes within one interval of size $1/k$. Note that this is an absolute loss and is not comparable to the approximation guarantee that we provide submodular maximization. The main problem with the bucketing algorithm is that it is to some extent oblivious to the input data and, thus, will fail badly for certain inputs as shown in the following proposition.

Proposition 10. *There is an input X to the bucketing algorithm such that $I(X; C) > 0$ and $I(Z; C) = 0$, where Z is the output of the bucketing algorithm.*

Proof. Fix a number j. In this example for half of the items we have $\Pr[C = 0|X = x] = \frac{j+1/3}{k}$ and for the other half we have $\Pr[C = 0|X = x] = \frac{j+2/3}{k}$. We also set the probability of all values of X to be the same, and hence $\Pr[C = 0] = \frac{j+0.5}{k}$. The mutual information of X with the label is non-zero since $\Pr[C = 0] \neq \Pr[C = 0|X = x]$. However, the bucketing algorithm merges all of the elements in the range $[\frac{j}{k}, \frac{j+1}{k})$, thereby merging all values together giving us $I(Z; C) = 0$ and completing the proof. $\square$

Note, we can further strengthen the previous example by giving a tiny mass to all buckets, so that all values do not collapse into a single bucket. However, still in this case, the bucketing method can only hope to capture a tiny fraction of mutual information since the vast majority of mass falls into a single bucket.

Frequency Based Filtering: This is very simple compression method (more precisely, a feature selection method) that is popular in practice. Given a vocabulary budget, we compute a frequency threshold τ which we use to remove any vocabulary value that appears in fewer than τ instances of our dataset and which results in a vocabulary of the desired size. Even though the frequency based algorithm is not entirely oblivious to the input, it is oblivious to the label and hence oblivious to conditional distributions. Similar to the bucketing algorithm with a simple example in the following theorem we show that the frequency based algorithm fails to provide any approximation gaurantee.

Proposition 11. *There is an input X to the frequency based algorithm such that $I(X; C) > 0$ and $I(Z; C) = 0$, where Z is the outcome of the frequency based algorithm.*

Proof. Assume we have $n = 3k$ values for X, namely $x_1, \ldots, x_n$. For all $i \in \{1, \ldots, k\}$ define $\Pr[X = x_i] = 2/n$, and for all $i \in \{k+1, \ldots, n\}$ we have $\Pr[X = x_i] = 0.5/n$. Note that the first k values are the most frequent values, however, we are going to define them such that they are independent of the label.

For all $i \in \{1, \ldots, k\}$ let $\Pr[X = x_i|C = 0] = 1/2$, and for all $i \in \{k+1, \ldots, 2k\}$ let $\Pr[X = x_i|C = 0] = 0$, and for all $i \in \{2k+1, \ldots, 3k\}$ let $\Pr[X = x_i|C = 0] = 1$. Note that we have $\Pr[C = 0] = \frac{1}{2}$. Therefore the mutual information of the k most frequent values with the label is zero, which implies for a certain vocabulary budget, and thereby frequency threshold, $I(Z; C) = 0$. Observe that even if we merge the last $2k$ values and use it as a new value (as opposed to ignoring them), the label corresponding the the merged value is 0 with probability half, and hence has no mutual information with the label. However, we have $I(X; C) = \sum_{x \sim X} \Pr[X = x] f(\Pr[C = 0|X = x]) = \sum_{i=k+1}^{2k} \frac{0.5}{n} = \frac{0.5 \times 2k}{3k} = \frac{1}{3} > 0$, which completes the proof. $\square$

3. Empirical Evaluation

In this section we report our empirical evaluation of the optimization the submodular function $F(S)$ described in the previous section. All the experiments are performed using the Criteo click prediction dataset (CriteoLabs, 2014), which consists of 37 million instances for training and 4.4 million held-out points.[1] In addition to 13 numerical features, this dataset contains 26 categorical features with a combined total vocabulary of more than 28 million values. These features have varying vocabulary sizes, from a handful up to millions of values. Five features, in particular, have more than a million distinct feature values each.

In order to execute a mutual information based algorithm, we require estimates of the conditional probabilities $\Pr[C = 0|X = x_i]$ and marginal probabilities $\Pr[X = x_i]$. Here, we simply use the maximum likelihood estimate based on the empirical count, i.e. given a sample of feature value and label pairs $((\hat{x}_1, \hat{c}_1), \ldots, (\hat{x}_k, \hat{c}_k))$, we have $\widehat{\Pr}[X = x_i] = \frac{1}{k} \sum_{j=1}^{k} \mathbf{1}\{\hat{x}_j = x_i\}$ and $\widehat{\Pr}[C = 0|X = x_i] = \frac{\frac{1}{k} \sum_{j=1}^{k} \mathbf{1}\{\hat{c}_j = 0 \wedge \hat{x}_j = x_i\}}{\widehat{\Pr}[X = x_i]}$. We note that such estimates may sometimes be poor, especially when certain feature values appear very rarely. Evaluating more robust estimates is outside the scope of the current study, but an interesting direction for future work.

3.1. Mutual information evaluation

We first evaluate the ability of our algorithm to maximize the mutual information retained by the compressed vocabulary and compare it to other baseline methods.

In particular, we compare our algorithm to the iterative divisive clustering algorithm (Dhillon et al., 2003), as well as the frequency filtering and bucketing heuristics introduced in the previous section. The divisive clustering algorithm resembles a version of the k-means algorithm where k is set to be the vocabulary size and distances between points and clusters are defined in terms of the KL divergence between the conditional distribution of the label variable given a feature value and the conditional distribution of the label variable given a cluster center. Notice that due to the large size of the dataset, we cannot run the dynamic programming algorithm introduced by Kurkoski & Yagi (2014) which would find the theoretically optimal solution. For ease of reference, we call our algorithm SUBMODULAR, and the other algorithms DIVISIVE, BUCKETING and FREQUENCY.

Note that our algorithm, as well as previous algorithms, seek to maximize the mutual information between a *single* categorical variable and the label, while in the Criteo dataset

---

[1]Note, we use the labeled training file from this challenge and chronologically partitioned it into train/hold-out sets.

we have several categorical variables that we wish to apply a global vocabulary size budget to. In the case of the FREQUENCY heuristic, this issue is addressed by sorting the counts of feature values across all categorical variables and applying a global threshold. In the case of SUBMODULAR, we run the almost linear-time algorithm for each categorical variable to obtain a sorted list of feature values and their marginal contributions to the global objective. Then we sort these marginal values and pick the top-score feature values to obtain the desired target vocabulary size. Thus, both SUBMODULAR and FREQUENCY are able to naturally employ *global* strategies in order to allocate the total vocabulary budget across different categorical features.

For the DIVISIVE and BUCKETING algorithms, a natural global allocation policy is not available, as one needs to define an allocation of the vocabulary budget to each categorical feature a priori. In this study, we evaluate two natural allocation mechanisms. The *uniform allocation* assigns a uniform budget across all categorical features, whereas the *MI allocation* assigns a budget that is proportional to the mutual information of the particular categorical feature.

The original vocabulary of over 28 million values is compressed by a factor of up to 2000. Using the methods mentioned above, we obtain vocabularies of size 10K, 20K, 40K, 80K, 120K and 160K. Then we compute the loss in average mutual information, which is defined as follows: let X_i denote the mutual information of uncompressed categorical feature i with the label and Z_i denote mutual information of the corresponding compressed feature, then the average mutual information loss is equal to $(\sum_i X_i - Z_i)/(\sum_j X_j)$.
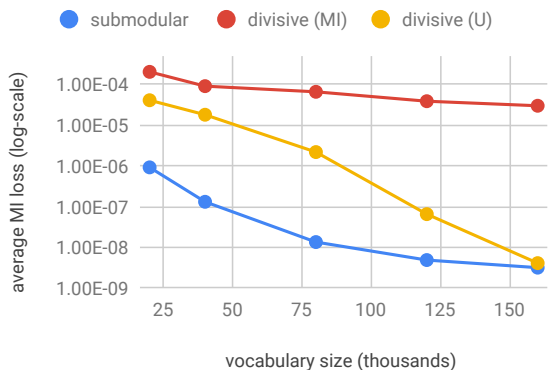

Figure 3. The average mutual information loss of several compression methods measured on to the Criteo dataset.

For the heuristic FREQUENCY algorithm, the measured loss ranges from 0.520 (for budget of 160K) to 0.654 (for budget of 10K), while for BUCKETING the loss ranges from 5×10^{-6} to 5×10^{-3}. As expected, the mutual information based methods perform significantly better, in particular, the loss for SUBMODULAR ranges from 9×10^{-7} to 3×10^{-9} and consistently outperforms the DIVISIVE algorithm

(regardless of allocation strategy). Figure 3 provides a closer look at the mutual information based methods. Thus, we find that not only is our method fast, scalable and exhibits a theoretical $1 - 1/e$ lower bound on the performance, but that in practice it maintains almost all the mutual information between data points and the labels.
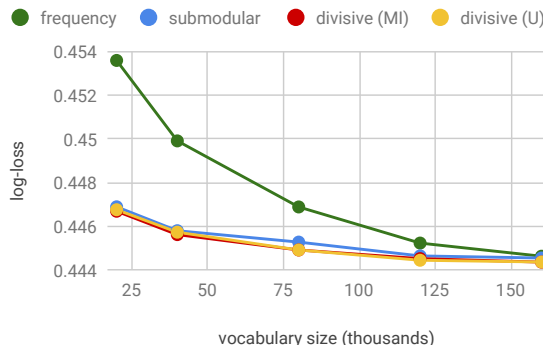

Figure 4. The log-loss of a neural network model trained with compressed vocabularies of several sizes and using several different compression methods.

3.2. Learning evaluation

Our focus thus far has been in optimizing the mutual information objective. In this section we also evaluate the compressed vocabularies in an end-to-end task to demonstrate its application in a learning scenario. Given a compressed vocabulary we train a neural network model on the training split and measure the log-loss on the hold out set (futher details in supplement Section A.2).[2] In Figure 4 we see that the mutual information based methods perform comparably to each other and significantly outperform popular heuristic method FREQUENCY. We observe that our scalable quasi-linear compression algorithm with provable approximation guarantees also performs competitively in end-to-end learning.

4. Conclusion

In this work we have shown the first scalable quasi-linear compression algorithm for maximizing mutual information with the label that also exhibits and $1 - 1/e$ factor approximation guarantee. The algorithm, as well as our insights into constructing a submodular objective function, might be of interest in other applications as well (for example, quantization in DMC). One future line of work is extending this work to the multiclass (non-binary) setting.

Acknowledgements

---

[2]In order to alleviate the potential issue of poor conditional/marginal distribution estimates we initially start with only features values that appear in at least 100 instances.

References

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., et al. Tensorflow: a system for large-scale machine learning. In *OSDI*, volume 16, pp. 265–283, 2016.

Aggarwal, A., Klawe, M. M., Moran, S., Shor, P., and Wilber, R. Geometric applications of a matrix-searching algorithm. *Algorithmica*, 2(1-4):195–208, 1987.

Badanidiyuru, A. and Vondrák, J. Fast algorithms for maximizing submodular functions. In *Proceedings of the twenty-fifth annual ACM-SIAM symposium on Discrete algorithms*, pp. 1497–1514. SIAM, 2014.

Baker, L. D. and McCallum, A. K. Distributional clustering of words for text classification. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 96–103. ACM, 1998.

Barbosa, R., Ene, A., Nguyen, H., and Ward, J. The power of randomization: Distributed submodular maximization on massive datasets. In *International Conference on Machine Learning*, pp. 1236–1244, 2015.

Cicalese, F., Gargano, L., and Vaccaro, U. Bounds on the entropy of a function of a random variable and their applications. *IEEE Transactions on Information Theory*, 64(4):2220–2230, 2018.

Cover, T. M. and Thomas, J. A. *Elements of Information Theory (Wiley Series in Telecommunications and Signal Processing)*. Wiley-Interscience, New York, NY, USA, 2006.

CriteoLabs. *Display Advertising Challenge*, 2014. URL https://www.kaggle.com/c/criteo-display-ad-challenge.

Dhillon, I. S., Mallela, S., and Kumar, R. A divisive information-theoretic feature clustering algorithm for text classification. *Journal of machine learning research*, 3 (Mar):1265–1287, 2003.

Guyon, I. and Elisseeff, A. An introduction to variable and feature selection. *Journal of machine learning research*, 3(Mar):1157–1182, 2003.

Iwata, K.-i. and Ozawa, S.-y. Quantizer design for outputs of binary-input discrete memoryless channels using smawk algorithm. In *Information Theory (ISIT), 2014 IEEE International Symposium on*, pp. 191–195. IEEE, 2014.

Jiang, J.-Y., Liou, R.-J., and Lee, S.-J. A fuzzy self-constructing feature clustering algorithm for text classification. *IEEE transactions on knowledge and data engineering*, 23(3):335–349, 2011.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Kumar, R., Moseley, B., Vassilvitskii, S., and Vattani, A. Fast greedy algorithms in mapreduce and streaming. *ACM Transactions on Parallel Computing (TOPC)*, 2(3):14, 2015.

Kurkoski, B. M. and Yagi, H. Quantization of binary-input discrete memoryless channels. *IEEE Transactions on Information Theory*, 60(8):4544–4552, 2014.

Mirrokni, V. and Zadimoghaddam, M. Randomized composable core-sets for distributed submodular maximization. In *Proceedings of the forty-seventh annual ACM symposium on Theory of computing*, pp. 153–162. ACM, 2015.

Mirzasoleiman, B., Karbasi, A., Sarkar, R., and Krause, A. Distributed submodular maximization: Identifying representative elements in massive data. In *Advances in Neural Information Processing Systems*, pp. 2049–2057, 2013.

Mirzasoleiman, B., Badanidiyuru, A., Karbasi, A., Vondrák, J., and Krause, A. Lazier than lazy greedy. In *AAAI*, pp. 1812–1818, 2015.

Mohri, M., Rostamizadeh, A., and Talwalkar, A. *Foundations of machine learning*. MIT press, 2018.

Mumey, B. and Gedeon, T. Optimal mutual information quantization is np-complete. In *Proc. Neural Inf. Coding (NIC) Workshop*, 2003.

Slonim, N. and Tishby, N. The power of word clusters for text classification. In *23rd European Colloquium on Information Retrieval Research*, volume 1, pp. 200, 2001.

Zhang, J. A. and Kurkoski, B. M. Low-complexity quantization of discrete memoryless channels. In *Information Theory and Its Applications (ISITA), 2016 International Symposium on*, pp. 448–452. IEEE, 2016.