A Scalable Algorithm for Higher-order Features Generation using MinHash

Pooja A
Amazon Development Centre
Bangalore, Karnataka
apooja@amazon.com

Naveen Nair
Amazon Development Centre
Seattle, Washington
nnair@amazon.com

Rajeev Rastogi
Amazon Development Centre
Bangalore, Karnataka
rastogi@amazon.com

ABSTRACT

Linear models have been widely used in the industry for their low computation time, small memory footprint and interpretability. However, linear models are not capable of leveraging non-linear feature interactions in predicting the target. This limits their performance. A classical approach to overcome this limitation is to use combinations of the original features, referred as higher-order features, to capture non-linearity. The number of higher-order features can be very large. Selecting the informative ones among them that are predictive of the target is essential for scalability. This is computationally expensive, requiring large memory footprint. In this paper, we propose a novel scalable MinHash based scheme to select informative higher-order features. Unlike typical use of MinHash for near-duplicate entity detection and association-rule mining, we use MinHash signature of features to approximate mutual information between higher-order features and target to enable their selection. By analyzing the running time and memory requirements, we show that our proposal is highly efficient in terms of running time and storage compared to existing alternatives. We demonstrate through experiments on multiple benchmark datasets that our proposed approach is not only scalable, but also able to identify the most important feature interactions resulting in improved model performance.

CCS CONCEPTS

• **Computing methodologies → Feature selection**;

KEYWORDS

Linear Models, Classification, Feature interactions

ACM Reference Format:
Pooja A, Naveen Nair, and Rajeev Rastogi. 2018. A Scalable Algorithm for Higher-order Features Generation using MinHash. In *The 27th ACM International Conference on Information and Knowledge Management (CIKM '18), October 22–26, 2018, Torino, Italy*. ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3269206.3271752

CIKM '18, October 22–26, 2018, Torino, Italy

ACM ISBN 978-1-4503-6014-2/18/10.
https://doi.org/10.1145/3269206.3271752

1 INTRODUCTION

Linear models are widely used in the industry for their low computation time, small memory footprint and interpretability. Unfortunately, linear models are not capable of leveraging non-linear feature interactions when predicting the target, which limits their performance. For example, linear model will not be able to predict income accurately when degree and field of specialization as a combination determine income, i.e., income is high if a person has a Masters degree in Business or Finance, but low if Masters is in a different field or the person does not have a Masters degree. This is because linear models predict the target as a weighted sum of features and not as weighted feature combinations. An alternative considered in the literature is to use models like Random Forests, Kernel Methods and Deep Neural Networks to capture non-linearity in mapping features to the target. Compared to Random Forests, linear models are more scalable, can be trained on large datasets that do not fit in memory, can handle millions of dimensions, and incur low computational overhead. Compared to Deep Neural Networks, linear models have smaller number of parameters and so can be trained even if data set sizes are not large, have fewer hyper-parameters that need to be tuned and so are easier to train, are easier to interpret, and incur lower computational overhead. These characteristics of the linear models make them the workhorse of the industry for incorporating artificial intelligence in business applications.

This leads us to another option explored in the literature, i.e., to enable linear models to learn non-linear mappings between features and target by incorporating feature interactions (aka. higher-order features) as additional features to capture non-linearity [19]. This approach retains all the benefits of linear models, but involves an additional step of identifying useful higher-order features to incorporate in the model. An exhaustive search for discovering informative higher-order features is computationally prohibitive and has huge memory requirements. It involves measuring the correlation between feature interactions and the target, which in turn requires estimation of their joint distribution. Assuming a small number of target values, the complexity of obtaining the joint distribution for higher-order features is exponential in the number of features and linear in the number of examples in the data. If a dataset has thousands of features and millions of examples, it can take trillions of time units to compute the correlation between the target and each possible feature pair (aka. second-order feature). Furthermore, millions of storage units will be required to store the co-occurrence counts associated with the joint distribution. In this paper, we propose to scale the discovery of informative higher-order features by leveraging the MinHash [2, 3] technique.

MinHash belongs to the family of Locality-sensitive Hashing (LSH) techniques and preserves Jaccard similarity between sets in the reduced hash space. Therefore, MinHash has been extensively used in large-scale similar entity detection [7], large-scale clustering [6] as well as in association-rule mining [8]. Unlike these applications of MinHash found in the literature, we propose to utilize the MinHash signature of features to approximate the mutual information between higher-order features and the target. Our proposal for discovering predictive higher-order features consists of two-steps. In the first step, we select candidate higher-order features based on mutual information with the target approximated using MinHash signature of the features. In the second step, we identify the most informative higher-order features to incorporate in the linear model from the candidate set based on mutual information with the target obtained using exact statistical computations. This two-step approach reduces the computation time as well as storage requirement of higher-order features selection significantly, while retaining most of the informative features in the selected set. The time and memory savings increase exponentially with increase in the order of feature interactions considered. In case of a dataset with thousands of features and millions of examples, the computation time for selecting second-order features with the proposed approach reduces by a factor of 1000 and storage requirement reduces by a factor of 10 as compared to exhaustive search. Experiments on multiple benchmark datasets demonstrate similar savings in time and memory, while reaffirming that linear models with higher-order features lead to better prediction performance. In summary the contributions of our paper are as follows:

- We propose a novel and scalable MinHash based scheme for automatic generation and selection of higher-order features.
- We show through extensive experimental evaluations that (1) Incorporation of higher-order features improves performance of linear models; (2) Our proposed MinHash based scheme accurately selects the most informative feature interactions; and (3) Our proposed MinHash based scheme is computationally efficient.
- We use available co-occurrence counts to identify and remove redundant features as well.

The rest of the paper is organized as follows: We introduce the notation used in the paper in Section 2 and formally define the higher-order features construction problem in Section 3. In Section 4, we describe the proposed MinHash based scheme for candidate features generation and compare with existing literature in Section 5. We empirically demonstrate the advantages of the proposed approach over exhaustive search in Section 6 and conclude the paper in Section 7 with possible future directions.

2 NOTATION

Before formally stating the higher-order features construction problem, we provide a brief overview of relevant concepts and introduce notation required for understanding the paper.

Information Theoretic based Feature Selection: Our proposal of higher-order features generation and selection relies on using correlation between feature interactions and the target to identify informative higher-order features. Correlation-based feature selection falls under a class of feature selection techniques called Information theoretic based techniques explained in the survey paper [14]. Information theoretic methods [4] have been used for almost two decades for filtering features based on variants of mutual information [13, 20, 26] between features and the target. These methods are capable of considering both feature relevance (how predictive is the feature of the target?) and feature redundancy (is feature reproducible as a weighted combination of other features?) while selecting features. As indicated in [14], feature selection under Information theoretic methods is independent of the learning algorithm and hence, is generalizable. However, these methods can only handle discrete data. As a result, continuous numerical variables require discretization as a preprocessing step. Further, these methods are applicable for supervised setting as they expect the target to be given. Given these requirements, we make the following assumptions.

Assumptions: We consider all features to be discretized. Thus, features are either categorical or binary. We bin numeric features using quantile binning and treat them as categorical features. We found performance improvements when discretized version of numerical features were used for building linear models instead of their continuous counterparts. Discretizing numeric features injects non-linearity and aids in obtaining higher predictive accuracy when using linear models. Therefore, we use discretized numerical features when reporting model performance as well. Text features, when present, are transformed to binary features by considering presence of each word as a feature. We discuss our techniques in a multi-class classification setting. Extending our proposal to a regression setting is a future direction of work.

Notation: In this paper, we represent the given feature set by $\mathcal{F} = \{f_1, f_2, \ldots, f_M\}$, where $f_1, f_2, \ldots, f_M$ are the M individual features. Each feature f_i can take values from $dom(f_i)$, the domain of f_i. Let T be the class or target variable which can take values from $dom(T)$, the domain of T. Therefore, an example instance e in the training set is a vector $\mathbf{e} =< v_1, ..., v_M, t >$, where $v_i \in dom(f_i), \forall i$ and $t \in dom(T)$. Since our features are assumed to be categorical, $dom(f)$ of a feature f is the set of categorical values that f takes. Without loss of generality, we will assume that the domains of features in $\mathcal{F}$ are disjoint and there is no overlap between the domains of different features. Further, we assume that the average size of the domain of features is $\bar{L}$. The order of $\bar{L}$ contributes to time and space complexity.

We represent a set of second-order features (aka. quadratic features) by $\mathcal{H}_2 = \{h_2 = f_{i_1} \bigotimes f_{i_2} |\ f_{i_1}, f_{i_2} \in \mathcal{F}\}$. Then, $dom(h_2) = dom(f_{i_1}) \times dom(f_{i_2})$. Each quadratic feature $f_{i_1} \bigotimes f_{i_2}$ can take values, $uv \in dom(f_{i_1}) \times dom(f_{i_2})$. Since quadratic features are constructed from individual categorical features whose domain is finite, the domain of a quadratic feature is also finite and the feature itself is categorical. Extending the same notation, we can represent the set of p^{th} order features by $\mathcal{H}_p = \{h_p = f_{i_1} \bigotimes ... \bigotimes f_{i_p} |\ f_{i_1}, ..., f_{i_p} \in \mathcal{F}\}$ with $dom(h_p) = dom(f_{i_1}) \times dom(f_{i_2}) \times ... \times dom(f_{i_p})$.

We represent the training set of examples by $\mathcal{S}$ and the number of examples in $\mathcal{S}$ by N. For a given feature value, $u \in f$, we denote the subset of examples that contain u by $\mathcal{S}_u$ and the size of the subset by N_u. Similarly the example subset containing a value pair $u \in f_{i_1}$ and $v \in f_{i_2}$ is denoted by $\mathcal{S}_{uv}$ and the size of the subset by N_{uv}. The example subset covered by a target class $t \in T$ is

denoted by $\mathcal{S}^t$ and its size by N^t. Therefore, the subset of examples in which the value pair ($u \in f_{i_1}, v \in f_{i_2}$) occurs and the target label is $t \in T$, can be denoted by $\mathcal{S}^t_{uv}$. Note that this is exactly the same set of examples covered by quadratic feature value $uv \in dom(h_2)$ and belonging to target $t \in T$. The number of examples in $\mathcal{S}^t_{uv}$ is denoted by N^t_{uv}. Extending the notation, each value tuple, $v-tuple$, of p^{th} order feature is associated with set $\mathcal{S}^t_{v-tuple}$ and its size is denoted by $N^t_{v-tuple}$.

Correlation Metric: We use the well known *symmetric information gain ratio* $U(f,T)$ [12, 26] between a feature f and the target T, as stated in Equation (1), to measure how informative the feature is about the target.

$$U(f,T) = 2\frac{H(f) + H(T) - H(f,T)}{H(f) + H(T)} \tag{1}$$

As indicated in Equation (1) , computing $U(f,T)$ for a given feature f and target T requires computation of three entropy values, $H(f), H(T)$ and $H(f,T)$. For feature $f, H(f) \stackrel{\Delta}{=} -\sum_{u \in dom(f)} P_u.logP_u$. Replacing f with T and u with t in the entropy definition of feature gives $H(T)$. The Entropy $H(f,T)$ for feature f and T can be computed as $H(f,T) = -\sum_{u \in dom(f)} \sum_{t \in dom(T)} P^t_u.logP^t_u$. Computing these entropy values require obtaining the below sets of probabilities, and in turn, obtaining counts N^t_u for all possible features and feature interactions.

(1) $P_u = Pr[f = u] = \frac{N_u}{N}, \forall u \in dom(f)$
(2) $P^t = Pr[T = t] = \frac{N^t}{N}, \forall t \in dom(T)$
(3) $P^t_u = Pr[f = u, T = t] = \frac{N^t_u}{N}, \forall u \in dom(f), t \in dom(T)$

Equation (1) can be extended to compute *symmetric information gain ratio* for p^{th}-order feature and target T by replacing f in the equation with h_p and u with $v - tuple$, respectively. It can be noted that $U(f,T)$'s value lies in $[0,1]$, with a higher value indicating that the feature f is more informative about the target. We apply a threshold τ for selecting the most informative higher-order features. We use $\mathcal{H}^{sel}_p$ to denote the selected set of p^{th}-order features and $\mathcal{H}^{sel}$ to denote the set of all selected higher-order features. Including only the most informative features helps to reduce noise in the data and improve prediction accuracy. In contrast, including all features can lead to overfitting apart from increasing storage as well as training and prediction times.

In the rest of the paper, we use the terms *symmetric information gain ratio* and *correlation measure* interchangeably.

3 PROBLEM STATEMENT AND SOLUTION OVERVIEW

With the notation defined, we state the higher-order features generation problem formally as below.

Higher-order Features Generation Problem: Given training set $\mathcal{S}$, with feature set $\mathcal{F} = \{f_1, f_2, \ldots, f_M\}$ and target T, identify the set $\mathcal{H}^{sel}_p$ such that *symmetric information gain ratio* in Equation (1) is greater than or equal to a threshold τ. ie,

$$\mathcal{H}^{sel} = \{\, h_p \mid U(h_p,T) \;\geq\; \tau,\; \forall h_p \in \mathcal{H}_p,\; p \in [2,M]\} \tag{2}$$

We hypothesize that data augmented with $\mathcal{H}^{sel}$, when used for training linear classification models, injects non-linearity into those models and improves performance (compared to the models built using the non-augmented data).

Determining the set $\mathcal{H}^{sel}$ requires computation of the correlation measure $U(h_p,T)$ for all possible feature combinations. This involves obtaining counts $N^t_{v-tuple}$, $N_{v-tuple}$ and N^t. The occurrence count $N_{v-tuple}$ can be derived from $N^t_{v-tuple}$ as $N_{v-tuple} = \sum_{t \in T} N^t_{v-tuple}$. Further, N^t for each target class $t \in T$ needs to be computed only once and can be obtained in time linear in the number of examples. The most expensive task in the higher-order feature selection process is obtaining $N^t_{v-tuple}$ for all possible tuples of features. Algorithm 1 encodes exhaustive search to select higher-order features. It considers all possible feature combinations as candidates[1] and uses Algorithm 2 for obtaining the exact co-occurrence counts required to compute correlation measures. Finally, only the feature combinations with correlation greater than or equal to τ with target are incorporated in the dataset. We refer to the higher-order feature set selected using exhaustive search as $\mathcal{H}^{selEx}$.

Algorithm 1 Exhaustive Search

Input: $\mathcal{S}$
Output: $\mathcal{H}^{selEx}$

$\mathcal{H}^{selCand} = \{h_p \mid h_p \in \mathcal{H}_p,\ \forall p \in [2,M]\}$
$\mathcal{N}$ = GetExactCoOccurCounts($\mathcal{S}, \mathcal{H}^{selCand}$)
$\mathcal{H}^{selEx} = \emptyset$
for $h \in \mathcal{H}^{selCand}$, **do**
  Compute $U(h,T)$ using relevant counts from $\mathcal{N}$ in Equation (1)
  if $U(h,T) > \tau$ **then**
    $\mathcal{H}^{selEx} = \mathcal{H}^{selEx} \cup h$
  end if
end for

Algorithm 2 Method to Compute Exact Co-occurrence Counts

GetExactCoOccurCounts
Input: $\mathcal{S}, \mathcal{H}^{selCand}$
Output: $\mathcal{N}$

$N^t_{v-tuple} = 0,\ \forall\ v - tuple \in dom(h_p),\ h_p \in \mathcal{H}^{selCand},\ t \in dom(T)$
for each example $e \in \mathcal{S}$, **do**
  for each higher-order feature $h_p \in \mathcal{H}^{selCand}$, **do**
    $N^t_{v-tuple} = N^t_{v-tuple} + 1$, where value of selected p features in e form $v - tuple$ and target takes value t in e.
  end for
end for
$\mathcal{N} = \{N^t_{v-tuple},\ \forall\ v - tuple \in dom(h_p),\ h_p \in \mathcal{H}^{selCand},\ t \in dom(T)\}$

[1]We use the term candidates here to make the algorithm consistent with the rest of the algorithms discussed in this paper.

Let us, now, analyze the computation time of Algorithm 1. We observe that we need to pass through the data once to obtain all occurrence counts associated with a higher-order feature. This is $O(N)$. Further, obtaining correlation measure for a p^{th}-order feature involves $O(\bar{L}^p)$ computations, where $\bar{L}$ is the average number of distinct values an individual feature takes. Since there exists $C(M,p)$ possible p^{th}-order features, the total computation time to obtain the correlation measure for all possible p^{th}-order features is $O(NM^p + \bar{L}^p M^p)$. In terms of storage, this scheme requires $O(M^p \bar{L}^p)$ memory to store occurrence counts of all possible feature combination values. If M is in thousands and N is in millions, obtaining $\mathcal{H}^{selEx}$ using Algorithm 1 becomes expensive, in terms of both time and memory. Even with $p = 2$ and $\bar{L} = 2$, the computation time is in trillions of units of time and storage required is in millions.

Therefore, our proposal is aimed at reducing the computation time and storage requirement in identifying higher-order features that satisfy the objective in Equation (2). We leverage the MinHash signature of features to approximate occurrence counts of higher-order feature values and in turn, approximate their correlation measures with target. Based on these approximate correlation measures, a relatively small candidate set of higher-order features are chosen and only for these, exact correlation measures are computed to determine the subset that satisfies the objective in Equation (2). Thus, our proposed strategy is to modify Step 1 in Algorithm 1 to initialize $\mathcal{H}^{selCand}$ with the one identified by MinHash based scheme rather than considering all possible feature combinations. This modification reduces the computation time by a factor of thousand and storage requirement by a factor of 10 for the case of quadratic binary features, i.e., when $p = 2$ and $\bar{L} = 2$. The order of reduction in time and storage is much larger for higher-order features as indicated in the next section.

4 CANDIDATE HIGHER-ORDER FEATURES GENERATION USING MINHASH

In this section, we propose an efficient, albeit approximate, way of obtaining the occurrence counts of higher-order feature values with reduced computation time and storage linear in M, the number of features. In this scheme, K hash functions, $h_1, \ldots, h_K$, are used to generate K-dimensional signature for each feature value-target combination. For a feature value $u \in f$ and target $t \in T$, the k^{th} MinHash [2, 3] signature value $h_k(u,t)$ is the minimum of row indices of example set S_u^t, permuted in a random order defined by the k^{th} hash function as indicated in Equation (3).

$$h_k(u,t) = min(index(e) : e \in \Pi_k(\mathcal{S})_u^t) \tag{3}$$

where, $index(e)$ is the index of example e in the random permutation and $\Pi_k(.)$ is the random permutation defined by the k^{th} hash function. The following proposition gives the probability that S_u^t corresponding to value u of f_{i_1} and S_v^t corresponding to v of f_{i_2} under target value t generate the same MinHash value for a given hash function h_k.

Proposition 1. *Given two sets, S_u^t and S_v^t, and a hash function, h_k, the probability of two features to generate the same MinHash value is given by [8]:*

$$Pr[h_k(u,t) = h_k(v,t)] = \frac{|S_u^t \cap S_v^t|}{|S_u^t \cup S_v^t|} \tag{4}$$

Proof Sketch: We observe that set S_u^t contains only examples with feature f_i taking value u. Similarly, S_v^t contain only examples with f_j taking value v. In this case, the probability of S_u^t and S_v^t to generate the same MinHash value is equal to the number of ways of picking an example taking values u for feature f_i and v for feature f_j out of the number of ways of picking examples taking value u for feature f_i or value v for feature f_j. This probability is nothing but Jaccard similarity between the sets S_u^t and S_v^t.

We observe that $|S_u^t \cap S_v^t|$ is the co-occurrence count N_{uv}^t and $|S_u^t \cup S_v^t| = N_u^t + N_v^t - N_{uv}^t$, where $N_u^t = |S_u^t|$ and $N_v^t = |S_v^t|$. Hence, Equation (4) can be used to obtain the co-occurrence counts if we know $Pr[h_k(u,t) = h_k(v,t)]$. With K-dimensional signature, this probability can be estimated as the fraction of matches in signatures, i.e., K_{uv}^t out of K. Thus, we can obtain the approximate co-occurrence counts, $\hat{N}_{uv}^t$, using Equation (5).

$$\begin{aligned} \frac{|S_u^t \cap S_v^t|}{|S_u^t \cup S_v^t|} &\approx \frac{K_{uv}^t}{K} \\ \frac{N_{uv}^t}{N_u^t + N_v^t - N_{uv}^t} &\approx \frac{K_{uv}^t}{K} \\ N_{uv}^t &\approx \frac{K_{uv}^t}{K + K_{uv}^t}(N_u^t + N_v^t) = \hat{N}_{uv}^t \end{aligned} \tag{5}$$

The same approach can be extended to higher-order features. It can be shown that probability of MinHash of p feature values to match is given by Equation (6). Hence, we can obtain the approximate occurrence counts of higher-order feature values as well.

$$Pr[h_k(v_{i_1},t) = h_k(v_{i_2},t) = \ldots = h_k(v_{i_p},t)] = \frac{|S_{v_{i_1}}^t \cap S_{v_{i_2}}^t \cap \ldots \cap S_{v_{i_p}}^t|}{|S_{v_{i_1}}^t \cup S_{v_{i_1}}^t \cup \ldots \cup S_{v_{i_p}}^t|} \tag{6}$$

Thus, we can use MinHash signatures to obtain correlation measure of higher-order features in an efficient way and select the candidate set based on these approximated measures as depicted in Algorithm 3. In the first 14 lines of the algorithm, the K-dimensional MinHash signature is obtained for all feature value-target combinations in the dataset using the Fast MinHash approach [22]. In the rest of the algorithm, generated signatures are used to select the candidate features set. Since the candidate set is identified using approximate statistics, it has to be large enough to contain most of the informative features. Therefore, we decrease the threshold τ by multiplying it with a damping factor, $C < 1$ before applying it on the approximate correlation measure to select candidate set. An exact computation of co-occurrence counts is performed only on this candidate set to determine the final set of higher-order features.

When we analyze the computation time, we observe that obtaining K-dimensional signature for any feature value does not depend on other features. Hence, signatures for all feature value-target combinations can be obtained in $O(NM)$ time. From the signatures, the approximate co-occurrence counts and correlation measure can be computed in $O(KM^p\bar{L}^p)$ time, where $\bar{L}$ is the average number of distinct values individual feature takes. Let M_{cand} be the size of the candidate set of higher-order features. Then, the exact computation to select from candidate set takes $O(NM_{cand} + \bar{L}^p M_{cand})$ time,

Algorithm 3 MinHash scheme to get Candidate Features

GetCandidates

Input: $\mathcal{S}, \tau$

Output: $\mathcal{H}^{selCand}$

for each $t \in T$, **do**
  for each f_i, **do**
    for each $u \in f_i$, **do**
      set $h_k(u,t) = \infty$ for $k = 1, \ldots, K$.
    end for
  end for
end for
for each $k \in \{1, \ldots, K\}$, **do**
  for each $e \in \Pi_k(\mathcal{S})$, **do**
    for each f_i, **do**
      $h_k(u,t) = min(h_k(u,t), index(e))$, where $T = t$ and $f_i = u$ in e.
    end for
  end for
end for
$\mathcal{H}^{selCand} = \emptyset$
for each feature combination $h_p = (f_{i_1}, f_{i_2}, ..., f_{i_p})$, **do**
  $\hat{N} = 0$
  for each $v_{i_1} \in f_{i_1}, v_{i_2} \in f_{i_2}, ..., v_{i_p} \in f_{i_p}, t \in T$, **do**
    $K^t_{v-tuple} = |k : 1 \leq k \leq K \wedge h_k(v_{i_1}, t) = h_k(v_{i_2}, t) = ... = h_k(v_{i_p}, t)|$
    Compute $\hat{N}^t_{v-tuple}$ on similar lines as Equation (5)
    $\hat{N} = \hat{N} + \hat{N}^t_{v-tuple}$
  end for
  Compute correlation measure $\hat{U}(h_p, T)$ on similar lines as Equation (1) with Ns replaced by $\hat{N}$s
  if $\hat{U}(h_p, T) > C.\tau$ **then**
    $\mathcal{H}^{selCand} = \mathcal{H}^{selCand} \cup h_p$
  end if
end for

where $M_{cand} \ll M^p$. Thus, the total running time of Algorithm 1 with $\mathcal{H}^{selCand}$ in Step 1 initialized by the output of Algorithm 3 is $O(NM + KM^p\bar{L}^p + NM_{cand}\bar{L}^p)$. If Step 1 is initialized with all possible higher-order features, the running time is $O(NM^p + M^p\bar{L}^p)$. If we can set $K \ll N$ and $M_{cand} \ll M^p$, we expect a significant speed up in higher-order feature generation with proposed MinHash based candidate selection [2] . The reduction in time is by a factor of $N/K\bar{L}^{(p-1)}$. For $p = 2$, $\bar{L} = 2$, $N = 10^6$ and $M = 10^3$, running time of the proposed scheme is one-thousandth the running time of exhaustive search with $\mathcal{H}^{selCand}$ comprising all possible higher-order features. With higher p, N and M values, the reduction in running time is higher. Through experiments on multiple datasets, we empirically establish the running time improvements with the proposed approach in Section 6.

Further, the memory footprint of the proposed approach is less than that required for exhaustive search. The MinHash based algorithm requires $O(KM\bar{L})$ memory to store MinHash signatures of features along with $O(M_{cand}\bar{L}^p)$ memory to store occurrence counts of the feature values in the candidate set. Thus, the total memory required is $O(KM\bar{L} + M_{cand}\bar{L}^p)$. On the other hand, exhaustive search that considers all possible higher-order feature combinations as candidates require $O(M^p\bar{L}^p)$ memory, which is higher by a factor of $M^{(p-1)}/K$. Note that the memory requirement is independent of the number of examples, N, in either case. For $p = 2$, $\bar{L} = 2$ and $M = 10^3$, memory required by the proposed scheme is in few hundred thousands of storage units, which is less by a factor of 10 compared to the memory required by complete exhaustive search. While memory requirement of MinHash scheme increases linearly in the number of features M, it increases exponentially in M for complete exhaustive search. Thus, the reduction in memory requirement is higher with higher values of p and M.

Choice of C: Damping factor C determines the size of candidate set. When $C = 0$, all the higher-order features are included in the candidate set and it is equivalent to skipping candidate generation step. When C is close to 1, size of candidate set becomes closer to the size of final set of selected features. We set C between 0.9 and 1.

Choice of K: We need to choose K such that the candidate set generated contains most of the informative features and result in high feature recall. This requires the approximated Jaccard similarity $\hat{s}$ using K-dimensional MinHash signatures of feature

[2]We derive lower bound for K in the following sections and provide empirical evidence showing effectiveness of the proposed approach using small values for K.

values to be within ϵ ($\epsilon > 0$) of the true Jaccard similarity s, that is, $Pr[|\hat{s} - s| > \epsilon] < \delta$, for all feature interactions.

Theorem 4.1. *The dimension K of the MinHash signature required to bound the error in estimation of all Jaccard similarities involved in p^{th}-order features selection by ϵ with confidence $1 - \delta$ ($0 < \delta \leq 1$) must satisfy the following constraint:*

$$K > \frac{1}{2\epsilon^2} ln(\frac{2n}{\delta}) \tag{7}$$

where, $n = |T|M^p\bar{L}^p$ is the number of Jaccard similarities to be computed for selecting p^{th}-order features among M given features.

Proof. See Appendix A

We use Chernoff-Hoeffding's bound to prove the theorem. This bound is not tight as it is independent of the value of Jaccard similarity. A tighter bound can be obtained by considering variance of Jaccard similarities involved. In practice, we observed setting K to 100 to result in high feature recall.

In the next section, we provide a survey of existing work that is closely related to our proposal in terms of approach and/or objective.

5 RELATED WORK

We note that our proposal is a filter type feature selection technique [11] and uses one of the widely used correlation measure, that is, *symmetric information gain ratio* [12, 26] to select informative higher-order features. In filter method, features are selected independent of the model used for predicting the target. Hence, they are more scalable in terms of running time and memory requirement than wrapper and ensemble methods that are dependent on the learning method [12]. Though we use *symmetric information gain ratio* in the paper, MinHash based scheme can be used for any feature selection method that relies on co-occurrence counts to identify informative features like Chi-Square Score [18] and CFS [12].

We observe that proposed MinHash based scheme is similar in motivation to Deep Crossing model [24] and FAST [19]. Deep Crossing model is a deep neural network that automatically combines features to produce superior models. It extends the success of Deep Neural Networks (DNNs) in learning from structured data like images and text to learning from general categorical and numeric features. Though it outperforms the production model, it has same disadvantages as DNNs, that is, high computation cost, large memory footprint and difficult to interpret. Hence, this method is not comparable with MinHash based scheme for generation of higher-order features. FAST, on the other hand, is a feature selection method like MinHash scheme and hence, is comparable. It is developed to efficiently identify informative pairwise feature interactions to incorporate in Generalized Additive Models (GAMs). Unlike our approach, it is a wrapper method of feature selection and uses strength of pairwise interactions for greedy forward selection of features. The strength of the pairwise interaction is measured using reduction in the residual error of the model. Similar to our approach, they discretize the features to reduce memory requirement. However, FAST is efficient only when the values of the features are comparable and hence, sortable. Our proposal does not impose such limitations on features and can handle all types of features. Further, we observe that the order of time complexity of FAST is similar to that of our proposed scheme. However, the space complexity of FAST is higher as it grows exponentially with number of features and distinct feature values. On the contrary, space complexity of our method increases linearly in both parameters. As a result, MinHash based scheme have lower memory footprint than FAST.

In general, hashing techniques have been used in the past to scale machine learning algorithms and our approach is inspired by their success. While [23] used Random Fourier and Binning features to scale kernel methods, [25] utilize Randomized Hashing to reduce computation cost of forward and back propagation. Random Hashing, in general, has been used extensively for large scale information retrieval [1] and clustering [9]. Similarly, MinHash has been used for large-scale similar entity detection [7], large-scale clustering [6] as well as for association-rule mining [8]. Both MinHash and Random Hashing belong to the family of Locality Sensitive Hashing (LSH) techniques. While Random Hashing preserves relative distance between points in the reduced hash space, MinHash preserves Jaccard similarity between sets. Since we are interested in estimating the co-occurrence counts of higher-order-features with the target, preserving Jaccard similarity is more relevant for our use case. Therefore, we generate MinHash signatures of features and use them to generate candidate set of predictive higher-order features. Instead of complete MinHash, we can use b-bit MinHash[15] to further reduce memory requirement.

Table 1: Summary of Datasets

DATASET	# FEATURES	TRAIN		TEST	
		Size	Pos%	Size	Pos%
Magic	10	15,216	64.60	3,804	65.80
Letter	16	16,000	Multi	4,000	Multi
Spambase	57	3,681	39.40	920	39.40
Physics	78	40,000	49.70	10,000	49.82
Gisette	5,000	6,000	50.00	1,000	50.00
Dataset1	20	140,001	4.27	60,009	4.28
Dataset2	69	1,339,604	10.32	334,459	10.20
Dataset3	528	860,176	1.75	358,851	1.69
Dataset4	5379	566,119	16.26	818,520	0.48

Pos% indicates the percentage of positive class in the training/test set

6 EXPERIMENTAL RESULTS

In this section, we evaluate the proposed algorithm in terms of model performance improvements and reduction in running times. We use two sets of data. The first set includes datasets used by [19] for benchmarking FAST. The second set includes Amazon-specific datasets created for different business use cases. While datasets in the first set are mostly class-balanced, datasets in the second set are from real-world applications and have high class imbalance. Using both the datasets, we show that: (1) MinHash based scheme is able to consistently retrieve truly informative higher-order features, typically using just 100-dimension signatures, (2) Model performance

significantly improves with the incorporation of higher-order features and the change in performance is negligible when approximate MinHash based scheme is used for identifying higher-order features, (3) MinHash based scheme speeds up higher-order features detection by a factor of almost 10, and (4) MinHash based scheme enables identification and removal of redundant features.

6.1 Dataset Description

We use both publicly-available benchmark datasets and real-world datasets from Amazon for experimentation. All datasets, except Letter dataset, correspond to binary classification problem and are summarized in Table 1. In the table, Pos% indicates percentage of positive examples in the training or test set. The first set of datasets, referred as **Benchmark datasets**, are publicly-available and same as those used by [19] to benchmark their technique for feature selection. Spambase, Magic and Letter datasets are from the UCI Repository[16]. On the other hand, Physics dataset is from the KDD Cup 2004[5]. We split them in 80:20 ratio as in [19] to obtain train and test sets. While these datasets are for benchmarking model performance, the fifth dataset i.e. the Gisette dataset is from the NIPS feature selection challenge[10]. Out of 5,000 features, 2,500 features in the Gisette dataset are non-informative. Since we are interested in higher-order feature generation, we run an initial feature selection step to remove noisy features and use only top 2,000 predictive features for our experimentation. Unlike these datasets, real-world datasets are often highly skewed. Hence, we use four Amazon-specific datasets (**Amazon datasets**) built for different business use cases to benchmark utility of the proposed algorithm for real-world applications. Among them, Dataset4 has the highest class imbalance. Hence, we down-sample the negative class while training the model. All the four datasets have numeric as well as categorical features. Any numeric feature is binned before building the model or generating higher-order features. On these datasets, we evaluated the proposed approach on the following three aspects:

(1) Ability of MinHash scheme to identify the informative candidate higher-order features as compared to exhaustive search.
(2) Ability of MinHash scheme to maintain model performance as that achieved by incorporation of truly informative higher-order features.
(3) Reduction in running time due to the use of MinHash scheme for candidate higher-order features generation compared to exhaustive search.

Experiment Setup: We used logistic regression to create linear models. For fair comparison, we fixed the learning parameters to their default values. Tuning of the parameters could potentially result in improved numbers. For benchmark datasets, we included top 100 higher-order features identified using either complete exhaustive search or MinHash based scheme followed by exhaustive search. Higher-order features explored include pairs as well as triplets of features. For Amazon datasets, exhaustive search does not scale beyond second-order features. Hence, we only explored second-order features (aka. quadratic features) for experimentation. For Amazon datasets, we choose the threshold used for candidate generation, τ, from $\{0.01\%, 0.05\%, 0.1\%, 0.5\%, 1\%, 2\%\}$ based on the model performance on held-out data.

6.2 Identification of Informative Higher-order Features

In this subsection, we compare the performance of the model built using each of the following feature sets:

- **BASELINE**: $\mathcal{F}^{sel} = \{ f_i \mid U(f_i, T) \geq \tau , i \in [1, M]\}$
- **EXHAUSTIVE**: $\mathcal{F}^{sel} \cup \mathcal{H}^{selEx}$, where $\mathcal{H}^{selEx}$ is obtained by considering all possible higher-order features as candidates.
- **MINHASH BASED**: $\mathcal{F}^{sel} \cup \mathcal{H}^{selMH}$, where candidate higher-order features set is obtained using MinHash scheme (Algorithm 3). The MinHash signature length, K, is set to 100.

Performance Metrics: For benchmark datasets, we use accuracy in prediction as performance metric. For Amazon datasets, we use recall at indicated lift. Lift is defined as the ratio of precision to Pos% in the test data. It indicates the factor by which model is more predictive than a random prediction and hence, captures the utility of the model even under high class imbalance.

Table 2 notes the accuracy of different techniques of feature selection on benchmark datasets. For FAST and Random Forest, we convert the mean error rate reported in [19] to accuracy and include in the table for comparison. On the other hand, Table 3 notes recall at indicated lift for different Amazon datasets achieved by different proposed feature selection techniques. From Table 2 and Table 3, we observe that the addition of selected higher-order features improves the model performance for most of the datasets. The improvement is typically in the range of 3-8%. Logistic regression model built with incorporation of MinHash based higher-order features selection (**MINHASH BASED**) has performance competitive with that of **FAST** and **Random Forest**.

Among the benchmark datasets, we observe significant accuracy improvement in case of Spambase dataset. This dataset corresponds to identifying spam emails based on words and characters that occur in them. Since occurrence of multiple spam-indicative words make the email more likely to be spam, inclusion of higher-order feature interactions (effectively, n-grams) improve model performance in detecting spam emails. Among Amazon datasets, we observe an improvement of around 21% in case of Dataset2. This dataset corresponds to matching similar entities. The features encode similarity between entities in different aspects. Higher-order features, in this case, capture similarity in multiple aspects. Hence, inclusion of higher-order features boosts model performance. On the other hand, we do not observe any performance improvement with the inclusion of higher-order features for Gisette and Dataset4. In case of Gisette, the feature set already includes higher-order feature interactions as indicated on UCI repository page[3]. Therefore, duplication of these features does not result in model performance improvement. In case of Dataset4, features include values aggregated over different overlapping time periods at different time granularities. Aggregation at higher time granularity can be considered as a combination of aggregation at lower time granularity. Therefore, inclusion of higher-order features again leads to zero improvement. Irrespective of the dataset, we observe that the performance of the model trained using **EXHAUSTIVE** feature set is very close[4]

[3] Gisette Dataset page: https://archive.ics.uci.edu/ml/datasets/Gisette
[4] Small variations in performance is due to use of SGD and presence of correlated features leading to slightly different converged weight vectors.

Table 2: Accuracy% for Baseline, Exhaustive search and MinHash based scheme for Benchmark datasets

DATASET	BASELINE	RANDOM FOREST [19]	FAST [19]	EXHAUSTIVE	MINHASH BASED	IMPROVE% OVER BASELINE
Magic	85.04	**87.55**	86.12	85.80	85.59	0.65
Letter	86.65	**93.84**	91.38	93.58	**93.83**	8.29
Spambase	94.02	95.24	95.22	**99.57**	**99.57**	5.90
Physics	70.85	71.52	**71.80**	71.10	71.10	0.35
Gisette	**97.90**	96.75	97.09	97.70	**97.90**	0

Table 3: Recall % at set Lift for Baseline, Exhaustive search and MinHash based scheme for Amazon datasets

DATASET	LIFT %	BASELINE	EXHAUSTIVE	MINHASH BASED	IMPROVE% OVER BASELINE
Dataset1	300	70.89	74.09	74.09	4.5
Dataset2	900	52.07	62.24	62.87	20.74
Dataset3	1900	55.94	57.66	57.66	3.07
Dataset4	400	28.00	Exception	28.00	0

Table 4: Running time in minutes and Feature recall% for Exhaustive search and MinHash Based Scheme

DATASET	EXHAUSTIVE (in minutes)	MINHASH (in minutes)	FEATURE RECALL%
Gisette	53	63	-
Dataset1	≤ 1	≤ 1	100± 0
Dataset2	15	13	92.13± 3.60
Dataset3	1,286	133	89.56±5.43
Dataset4	Exception	932	-

to the performance of model trained using **MINHASH BASED** feature set. This ascertains that the candidate higher-order features selection using MinHash based scheme is able to retain most of the informative features, while filtering out less-informative ones. Further, the proposed scheme is faster than exhaustive search as established in the next subsection.

6.3 Reduction in Running time

In this paper, we proposed MinHash based scheme to mainly reduce the running time of Exhaustive search (Algorithm 1) for higher-order features generation. To demonstrate the time efficiency of the proposed approach, we compare the running time of Exhaustive search with and without candidate initialization using MinHash based scheme on Amazon Datasets. From Table 4, it can be observed that with the use of MinHash scheme, we can achieve significant reduction in running time. Running time for Dataset3 is observed to be reduced by almost a factor of 10. Further, MinHash based scheme helped scale exhaustive search for Dataset4. In case of benchmark datasets excluding Gisette, we observe that the running time with and without MinHash based candidate set initialization is less than a minute owing to their small data size. Hence, we do not include these datasets in Table 4. In case of Gisette, MinHash based scheme is observed to take more time than Exhaustive search. This is because the additional steps of generating MinHash signatures take more time than straight-forward exhaustive search when the number of examples is comparable to the number of features and signature length. Our proposed approach leads to reduction in running time when MinHash signature length is much smaller than the number of examples in the dataset, i.e., when $K \ll N$.

Further, we note that MinHash scheme is probabilistic and can lead to variations in the generated candidate set. To demonstrate that the variations in candidate set does not adversely impact performance, we run the scheme multiple times and report the mean and standard deviation of feature recall with MinHash based candidate generation. Feature recall is the fraction of truly informative features that were selected by MinHash based scheme and is given by the ratio, $|\mathcal{H}^{selMH} \cap \mathcal{H}^{selEx}|/|\mathcal{H}^{selEx}|$. From Table 4, we observe that the proposed scheme is stable and is able to retain more than 90% of the truly informative higher-order features set. The drop in some of the features, typically, does not affect model performance as observed in Table 2 and Table 3.

6.4 Choice of MinHash Signature Length, K

In this subsection, we analyze the impact of choosing the dimension, K, of the MinHash signature of features using one of the datasets, i.e., Letter dataset. We choose K from $\{10, 20, 50, 100, 500, 1000\}$ and plot corresponding running time, feature recall and model accuracy in Figure 1. From the plots, we observe that running time increases almost linearly with K and feature recall improves as K increases. However, the model accuracy remains almost same at around 93.60%. We observe similar trends for other datasets. Thus, we conclude that MinHash based scheme is not very sensitive to

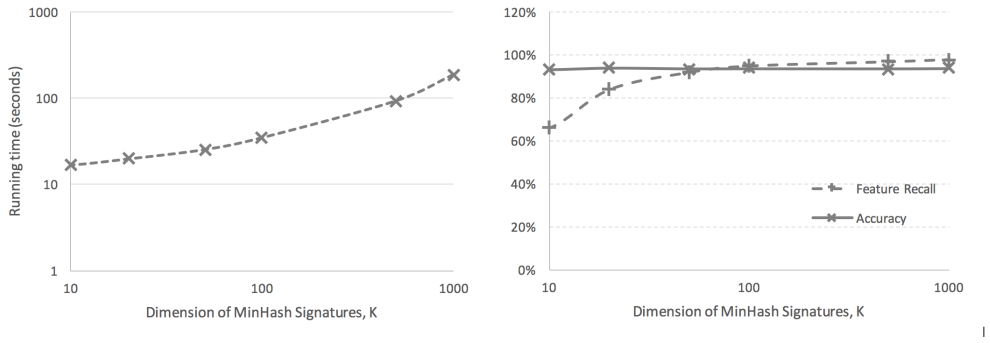

Figure 1: Change in running-time, feature recall and model accuracy with different K values.

the choice of K. In our experiments, we found $K = 100$ to be sufficient. This is much smaller than the lower bound on K derived in Appendix A.

6.5 Extension to Redundant Features Removal

In addition to identification of informative higher-order features, approximated co-occurrence counts using MinHash can be used to identify redundant features. The approximate co-occurrence counts of second-order feature-values $\hat{N}_{uv}$ can be obtained by aggregating co-occurrence counts over all target values t, i.e, $\hat{N}_{uv} = \sum_{t \in dom(T)} \hat{N}^t_{uv}$. With the co-occurrence counts known between all possible feature pairs, we can approximate the correlation between them. When the correlation between any two features is high, we can drop the one with lower correlation with the target without affecting model performance. We applied this step when generating higher-order candidates. This step adds minimal overhead on the computation. In case of Spambase dataset, we observed words "lab" and "labs" to be highly correlated. Words "cs", "project" and "conference" were found to be highly correlated with each other. Around 3% of all the possible features pairs in Gisette dataset were highly-correlated. Only the most predictive features from each redundant sets were used to train the linear models.

In summary, we observe through evaluation on multiple datasets that the performance of linear models can be improved with the addition of higher-order features and this can be accomplished efficiently with the use of MinHash based scheme for candidate generation.

7 CONCLUSION

In this paper, we noted that linear models are scalable and interpretable and hence, are widely used in the industry. We observed that the performance of linear models is limited by their inability to leverage non-linear interactions between features. We posed the problem of incorporating non-linear information in linear models as discovering informative feature interactions (aka. higher-order features). We proposed a novel MinHash based scheme for addressing this problem. In this scheme, the MinHash signature of features were used to approximate the mutual information between higher-order features and the target to enable selection of predictive feature interactions at scale. By analyzing the running time and memory requirements, we established that the proposed scheme leads to a significant reduction in time and memory requirement compared to alternatives like FAST, Random Forest, Deep Crossing model and exhaustive search. We demonstrated through experiments on multiple datasets that our proposed MinHash based scheme is able to consistently retrieve truly informative higher-order features in significantly lower computation time, while resulting in model performance improvements similar to that achieved with exhaustive search.

While we explored one approach for redundant features removal during feature selection, there are formulations available in the literature like minimal-redundancy-maximal-relevance criterion (mRMR) [21] and Conditional Infomax [17] that unify feature selection and feature redundancy removal. We plan to explore these formulations in our proposed MinHash based feature selection scheme. Further, we plan to extend our proposed approach to linear regression setting.

ACKNOWLEDGMENTS

We thank Vineet Chaoji, whose work (at Amazon) on using hashing techniques to discover redundant features inspired us to pursue this research path. We also thank Prakash Mandayam Comar and the anonymous referees for their valuable comments and helpful suggestions.

REFERENCES

[1] Ella Bingham and Heikki Mannila. 2001. Random projection in dimensionality reduction: applications to image and text data. In *Proceedings of the 7th ACM SIGKDD international conference on Knowledge discovery and data mining*. KDD, 245–250.
[2] Andrei Z Broder. 1997. On the resemblance and containment of documents. In *Compression and Complexity of Sequences 1997. Proceedings*. IEEE, 21–29.
[3] Andrei Z Broder, Steven C Glassman, Mark S Manasse, and Geoffrey Zweig. 1997. Syntactic clustering of the web. *Computer Networks and ISDN Systems* 29, 8-13 (1997), 1157–1166.
[4] Gavin Brown, Adam Pocock, Ming-Jie Zhao, and Mikel Luján. 2012. Conditional Likelihood Maximisation: A Unifying Framework for Information Theoretic Feature Selection. *J. Mach. Learn. Res.* 13 (Jan. 2012), 27–66. http://dl.acm.org/citation.cfm?id=2188385.2188387
[5] Rich Caruana and Thorsten Joachims. 2004. KDD Cup. (2004). http://osmot.cs.cornell.edu/kddcup/
[6] Ondrej Chum et al. 2010. Large-scale discovery of spatially related images. *IEEE transactions on pattern analysis and machine intelligence* 32, 2 (2010), 371–377.
[7] Ondrej Chum, James Philbin, Andrew Zisserman, et al. 2008. Near Duplicate Image Detection: min-Hash and tf-idf Weighting.. In *BMVC*, Vol. 810. 812–815.
[8] Edith Cohen, Mayur Datar, Shinji Fujiwara, Aristides Gionis, Piotr Indyk, Rajeev Motwani, Jeffrey D Ullman, and Cheng Yang. 2001. Finding interesting associations without support pruning. *IEEE Transactions on Knowledge and Data*

Engineering 13, 1 (2001), 64–78.
[9] Xiaoli Z Fern and Carla E Brodley. 2003. Random projection for high dimensional data clustering: A cluster ensemble approach. In *Proceedings of the 20th International Conference on Machine Learning*. ICML, 186–193.
[10] Isabelle Guyon. 2003. NIPS Feature Selection Challenge. (2003). http://clopinet.com/isabelle/Projects/NIPS2003/#challenge
[11] Isabelle Guyon and André Elisseeff. 2003. An introduction to variable and feature selection. *Journal of machine learning research* 3, Mar (2003), 1157–1182.
[12] Mark A. Hall and Lloyd A. Smith. 1999. Feature Selection for Machine Learning: Comparing a Correlation-Based Filter Approach to the Wrapper. In *Proceedings of the Twelfth International Florida Artificial Intelligence Research Society Conference*. AAAI Press, 235–239. http://dl.acm.org/citation.cfm?id=646812.707499
[13] David D. Lewis. 1992. Feature Selection and Feature Extraction for Text Categorization. In *Proceedings of the Workshop on Speech and Natural Language (HLT '91)*. Association for Computational Linguistics, Stroudsburg, PA, USA, 212–217. https://doi.org/10.3115/1075527.1075574
[14] Jundong Li, Kewei Cheng, Suhang Wang, Fred Morstatter, Robert P. Trevino, Jiliang Tang, and Huan Liu. 2016. Feature Selection: A Data Perspective. *CoRR* abs/1601.07996 (2016). arXiv:1601.07996 http://arxiv.org/abs/1601.07996
[15] Ping Li and Arnd Christian König. 2010. Theory and Applications of b-bit Minwise Hashing. In *Proceedings of the Annual Conference on Neural Information Processing Systems*. NIPS, 1387–1395.
[16] M. Lichman. 2013. UCI Machine Learning Repository. (2013). http://archive.ics.uci.edu/ml
[17] Dahua Lin and Xiaoou Tang. 2006. Conditional Infomax Learning: An Integrated Framework for Feature Extraction and Fusion. In *Computer Vision – ECCV 2006*, Aleš Leonardis, Horst Bischof, and Axel Pinz (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 68–82.
[18] Huan Liu and Rudy Setiono. 1995. Chi2: Feature Selection and Discretization of Numeric Attributes. In *Proceedings of the Seventh International Conference on Tools with Artificial Intelligence (TAI '95)*. IEEE Computer Society, Washington, DC, USA, 88–. http://dl.acm.org/citation.cfm?id=832245.832359
[19] Yin Lou, Rich Caruana, Johannes Gehrke, and Giles Hooker. 2013. Accurate intelligible models with pairwise interactions. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*. KDD, 623–631.
[20] P. E. Meyer, C. Schretter, and G. Bontempi. 2008. Information-Theoretic Feature Selection in Microarray Data Using Variable Complementarity. *IEEE Journal of Selected Topics in Signal Processing* 2 (June 2008), 261–274. https://doi.org/10.1109/JSTSP.2008.923858
[21] Hanchuan Peng, Fuhui Long, and Chris Ding. 2005. Feature Selection Based on Mutual Information: Criteria of Max-Dependency, Max-Relevance, and Min-Redundancy. *IEEE Trans. Pattern Anal. Mach. Intell.* 27, 8 (Aug. 2005), 1226–1238. https://doi.org/10.1109/TPAMI.2005.159
[22] Jeff Phillips. 2013. Lecture on MinHashing. (2013). https://www.cs.utah.edu/~jeffp/teaching/cs5955/L5-Minhash.pdf
[23] Ali Rahimi and Benjamin Recht. 2008. Random features for large-scale kernel machines. In *Advances in neural information processing systems*. 1177–1184.
[24] Ying Shan, T Ryan Hoens, Jian Jiao, Haijing Wang, Dong Yu, and JC Mao. 2016. Deep crossing: Web-scale modeling without manually crafted combinatorial features. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD, 255–262.
[25] Ryan Spring and Anshumali Shrivastava. 2017. Scalable and sustainable deep learning via randomized hashing. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD, 445–454.
[26] Lei Yu and Huan Liu. 2003. Feature Selection for High-Dimensional Data: A Fast Correlation-Based Filter Solution. In *Proceedings, Twentieth International Conference on Machine Learning*, T. Fawcett and N. Mishra (Eds.), Vol. 2. 856–863.

A LOWER BOUND ON MINHASH SIGNATURE LENGTH, K

We proposed MinHash based scheme for approximating mutual information between higher-order features and the target and in turn, scale their feature selection in terms of computation time and memory requirement. In this section, we derive lower bound on K, the dimensionality of the MinHash signature. Then, we compare the lower bound on K for MinHash scheme with empirical observations.

Proof. of Theorem 4.1 Let us, first, consider the simple case of estimating Jaccard similarity between two binary vectors, v_1 and v_2, using K MinHash functions, $\{h_1 ... h_K\}$.

To help us derive lower bound on K like in [22], we introduce the random variable, X_k, defined as below.

$$X_k = \begin{cases} 1, & \text{if } h_k(v_1) = h_k(v_2) \\ 0, & \text{otherwise.} \end{cases}$$

Then, $\hat{s} = \frac{1}{K}\sum_{k=1}^{K} X_k$ is the estimate of Jaccard similarity, s, between vectors v_1 and v_2. We note that $E[\hat{s}] = s$ and thus, $\hat{s}$ is an unbiased estimate. To be confident about our estimate of Jaccard similarity, we require K to be chosen such that error in our estimate $|\hat{s} - s|$ rarely exceeds ϵ, that is, $Pr[|\hat{s} - s| > \epsilon] < \delta$, where, δ is the probability of error exceeding ϵ with $0 < \epsilon < 1$. Using Chernoff-Hoeffding's bound, we get $Pr[|\hat{s} - s| > \epsilon] < 2e^{-2K\epsilon^2}$, that is, the probability of error exceeding the limit decreases exponentially with increase in the size of the sample, K. Putting the two limits together, we get

$$K > \frac{1}{2\epsilon^2} \ln(\frac{2}{\delta}) \tag{8}$$

In the paper, MinHash signature of length K is used to approximate mutual information of all possible higher-order features. Therefore, we need to determine K such that all the Jaccard similarities involved are estimated within $\pm\epsilon$ with a high probability, $(1 - \delta)$. Therefore, we need to choose K such that the estimates of all $n = |T|\dot{M}^p\dot{L}^p$ Jaccard similarities are within the bound. This leads to

$$1 - \Pi_{q=1}^{n}(1 - Pr[|\hat{s_q} - s_q| > \epsilon]) < \delta$$

When K is large, we can apply the additive bound on LHS, that is, $1 - \Pi_{q=1}^{n}(1 - Pr[|\hat{s_q} - s_q| > \epsilon]) < \sum_{q=1}^{n} Pr[|\hat{s_q} - s_q| > \epsilon]$. Finally, applying Chernoff-Hoeffding's bound, we get

$$\begin{aligned} \sum_{q=1}^{n} Pr[|\hat{s_q} - s_q| > \epsilon] &< \delta \\ ne^{-2K\epsilon^2} &< \delta \\ K &> \frac{1}{2\epsilon^2} ln(\frac{2n}{\delta}) \end{aligned} \tag{9}$$

□

Discussion For $\epsilon = 0.1$ and $\delta = 0.01$, we get $K > 265$ from Equation 8. This implies we need signature of length $K > 265$ to estimate Jaccard similarity between two binary vectors within an error of ± 0.1 with 99% confidence. For selecting second-order features ($p = 2$) among thousands of features ($M = 10^3$) with average feature domain size of $\bar{L} = 10$ within an error of ± 0.1 with 99% confidence, we require $K > 1,221$ as per Equation 9. In practice, however, we observe that a relatively small value of K led to high recall of informative higher-order features. This is because the bound is not tight. In addition, we use MinHash signatures to estimate mutual information between higher-order features and the target. When higher-order feature and target take $\bar{L}^p$ and $|T|$ unique values, approximation of mutual information involves estimation of $\bar{L}^p\dot{T}$ Jaccard similarities. Even when Jaccard similarity of some of the feature-value tuples are erroneous, it might not affect the estimated mutual information significantly.