A DEEP DIVE INTO COUNT-MIN SKETCH FOR EXTREME CLASSIFICATION IN LOGARITHMIC MEMORY

Anonymous authors
Paper under double-blind review

ABSTRACT

Extreme Classification Methods have become of paramount importance, particularly for Information Retrieval (IR) problems, owing to the development of smart algorithms that are scalable to industry challenges. One of the prime class of models that aim to solve the memory and speed challenge of extreme multi-label learning is Group Testing. Multi-label Group Testing (MLGT) methods construct label groups by grouping original labels either randomly or based on some similarity and then train smaller classifiers to first predict the groups and then recover the original label vectors. Recently, a novel approach called MACH (Merged Average Classifiers via Hashing) was proposed which projects the huge label vectors to a small and manageable count-min sketch (CMS) matrix and then learns to predict this matrix to recover the original prediction probabilities. Thereby, the model memory scales $O(logK)$ for K classes. MACH is a simple algorithm which works exceptionally well in practice. Despite this simplicity of MACH, there is a big gap between the theoretical understanding of the trade-offs with MACH. In this paper we fill this gap. Leveraging the theory of count-min sketch we provide precise quantification of the memory-identifiablity tradeoffs. We extend the theory to the case of multi-label classification, where the dependencies make the estimators hard to calculate in closed forms. To mitigate this issue, we propose novel quadratic approximation using the Inclusion-Exclusion Principle. Our estimator has significantly lower reconstruction error than the typical CMS estimator across various values of number of classes K, label sparsity and compression ratio.

1 INTRODUCTION

Extreme Classification has taken center-stage of Data Mining and Information Retrieval research in the past few years (Zamani et al., 2018; Prabhu et al., 2018b; Jain et al., 2019; Choromanska & Langford, 2015). It refers to the vanilla multiclass and multilabel classification problems where the number of classes K is significantly large. A large number of classes K brings a new set of computational and memory challenges in training and deploying classifiers.

There have been several paradigms of models that tackle the scale challenge of Extreme Classification like 1-vs-all methods (Prabhu et al., 2018b; Jain et al., 2019; Babbar & Schölkopf, 2017), tree based methods (Prabhu et al., 2018a; Jain et al., 2016), embedding models (Nigam et al., 2019; Bhatia et al., 2015), etc. (as noted on the popular Extreme Classification Repository). One of the recent approaches proposed to alleviate the scale challenge of Multilabel Classification is Group Testing (Ubaru & Mazumdar, 2017; Ubaru et al., 2016; Vem et al., 2017). In this method, all labels are grouped randomly into m groups/clusters. Each label may go into more than one group. We first train a classifier that predicts which of these clusters the input belongs to (treating each cluster as a separate label in a multilabel setting). For any given input, we first predict the clusters into which the true labels of the input may have been pooled. We can then identify all the true labels by taking an intersection over the inverted clusters. This approach suffers from a critical problem that even tree based approaches have, i.e., hard assignment of clusters. Since the recovery of true labels depends solely on hard-prediction of clusters, a mistake in the cluster prediction can cost us dearly in the final label prediction. Also, since the labels are pooled randomly, each individual meta-classifier is a weak and noisy one.

In a recent development, Merged Average Classifiers via Hashing (MACH) (Medini et al., 2019) was proposed that alleviates the hard-prediction problem in Group Testing methods by identifying the best labels based on the sum of prediction probabilities of the respective groups for a given input. In the hindsight, MACH subtly learns to predict a count-min sketch (CMS) (Cormode & Muthukrishnan, 2005) matrix of the original probability vector. For the case of multiclass classification (every

input having just a single label unlike multilabel), MACH proposes an unbiased estimator to recover the original K dimensional probability vector from the predicted CMS matrix. Multiclass classification naturally fits into the count-min sketch setting as no two labels can appear simultaneously for a given input. But the proposed theory does not naturally extend to multilabel learning. Further, the variance and error bounds for multiclass classification rely heavily on the choice of number of hash tables and the size of each hash table. That aspect has not been explored in prior work.

Our Contributions: In this work we broadly make the following contributions: **1)** We revisit MACH with a thorough analysis of proposed reconstruction estimator for multiclass learning. In particular, we prove that the variance of estimation is inversely proportional to the product of product of number of hash tables and size of each hash table (in theorem 2). **2)** We also obtain a lower bound on hash table hyperparametrs given a tolerance to prediction error (in Theorems 4 and 5). **3)** We propose a novel reconstruction estimator for the case of multilabel learning using Inclusion-Exclusion principle (in theorem 6). This estimator comes out as a solution to a quadratic equation (hence we code-name it as 'quadratic estimator'). **4)** We simulate multilabel learning setting by generating K dimensional probability vectors and their proxy CMS measurements. We then reconstruct the probability vector using both the mean estimator and the quadratic estimator and show that the reconstruction Mean-Squared Error (MSE) is significantly lower for the new estimator.

2 BACKGROUND

Count-Min Sketch: Count-Min Sketch (CMS) (Cormode & Muthukrishnan, 2005) was proposed to solve the frequency counting problem in large streaming setting. Assume that we have an infinite stream of elements $e_1, e_2, e_3, ...$ coming in. Each of these elements can take any value between K distinct ones. Here, K is very large and we cannot afford to store an array of counts to store every element's frequency (limited memory setting). We need a sub-linear efficient data structure from which we can retrieve the frequency of every element.

In Count-Min Sketch (Cormode & Muthukrishnan, 2005), we basically assign $O(\log K)$ 'signatures' to each class using 2-universal hash functions. We use $O(\log K)$ different hash functions $H_1, H_2, H_3, ..., H_{O(\log K)}$, each mapping any class i to a small range of buckets $B << K$, i.e., $H_j(i) \in \{0, 1, 2, ..., B\}$. We maintain a counting-matrix C of order $O(\log K) * B$. If we encounter class i in the stream of classes, we increment the counts in cells $H_1(i), H_2(i)....., H_{O(\log K)}(i)$. It is easy to notice that there will be collisions of classes into these counting cells. Hence, the counts for a class in respective cells could be over-estimates of the true count.

	H1	H2	H3	H4
A	1	6	3	1
B	1	2	4	6
C	3	4	1	6
D	6	2	4	1

	0	1	2	3	4	5	6
H1	0	1+1+1+1=4		1+1 = 2	0	0	1
H2	0	0	1+1=2	0	1+1 = 2	0	1+1+1=3
H3	0	1+1 = 2	0	1+1+1=3	1+1 = 2	0	0
H4	0	1+1+1+1=4	0	0	0	0	1+1+1=3

Figure 1: Illustration of count-min sketch for a stream of letters ABCAACD. The hash codes for each letter for 4 different hash functions is shown on the left and the accumulated counts for each of the letter in the stream is shown on the right

During inference, we want to know the frequency of a particular element say a_1. We simply go to all the cells where a_1 is mapped to. Each cell gives and over-estimated value of the original frequency of a_1. To reduce the offset of estimation, the algorithm proposes to take the minimum of all the estimates as the approximate frequency, i.e., $n_{approx}(a_1) = min(C[1, H_1(i)], C[2, H_2(i), ..., C[\log K, H_{\log K}])$.

An example illustration of CMS is shown in figure 1.

Connecting CMS and Extreme Classification: Given a data instance x, a vanilla classifier outputs the probabilities p_i, $i \in \{1,\ 2, ...,\ K\}$. We want to essentially compress the information of these K numbers to $\log K$,i.e., we can only keep track of $\log K = BR$ measurements. Ideally, without any assumption, we cannot compress the information in K numbers to anything less than $O(K)$, if we want to retain all information. However, in classification, the most informative quantity is the identity of $\arg\max p_i$. If we can identify a scheme that can recover the high probability classes from smaller measurement vector, we can train a small-classifier to map an input to these measurements instead of the big classifier.

The foremost class of models to accomplish this task are Encoder and Decoder based models like Compressive Sensing (Baraniuk, 2007). The connection between compressed sensing and extreme classification was identified in prior works (Hsu et al., 2009; Dietterich & Bakiri, 1995). We provide an intuitive explanation of why compressed sensing or any other sketching algorithm does work like count-min sketch in the appendix A.

2.1 Merged Average Classifiers via Hashing (MACH)

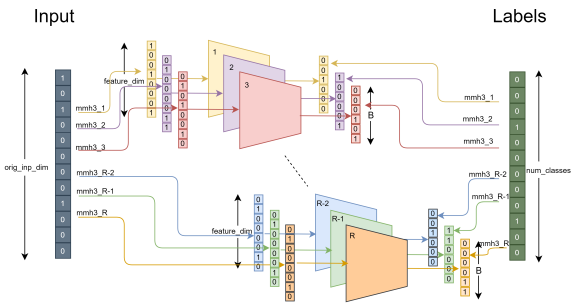


Figure 2: Schematic diagram of MACH. Both the input and the label vector are independently hashed R times (label vector is hashed from K to B, K being number of classes and B being number of buckets in each of the R hash tables). Small models are then trained in parallel.

MACH (Medini et al., 2019) is a new paradigm for extreme classification that uses universal hashing to reduce memory and computations. MACH randomly merges K classes into B meta-classes or buckets ($B \ll K$). We then runs any off-the shelf classifier (typically simple feed forward neural networks) to predict the meta classes. This process is repeated R number of times, changing the hash function each time (or by simply changing the random seed of the same hash function, to induce a different random pooling each time). During prediction, MACH aggregates the output from each of the R small meta classifiers to retrieve the best class.

In the schema shown in figure 2, the input is assumed to be a large dimensional sparse vector. In order to reduce model size from both ends (input and output), the sparse input can also be feature hashed (Weinberger et al., 2009) to a manageable dimension. Please note that the theoretical analysis of MACH is agnostic to the input feature hashing. We are only concerned with retrieving the most relevant labels from the meta-class predictions.

The subsequent sections formalize the algorithm and quantify the mean, variance, error bounds and hyper-parameter bounds.

3 Theoretical Analysis

We begin with emphasizing that MACH does not assume any dependence among the classes. This is a fairly strong assumption because often in extreme classification, the labels have strong correlations. More so, this assumption is intrinsically violated in the case of multilabel learning. Nevertheless, MACH works extremely well in practice, particularly at industry scale challenges.

Let there be K classes originally. We'll hash them to B meta-classes using a universal hash function. We repeat this process R times each with a different hash function (can be obtained by simply changing the random seed each time). We only have an $R*B$ matrix that holds all information about the original probability vector of K dimensions ($R * B \ll K$). Typical classification algorithms model the probability $Pr(y = i|x) = p_i$ where $i \in \{0, 1, 2...K - 1\}$. With MACH, we bypass the hassle of training a huge last layer by instead modelling $Pr(y = b|x) = P_b^j$ for every hash function h_j, where $b \in \{0, 1, 2, ..., B - 1\}$ and $j \in \{0, 1, 2, ..., R - 1\}$. During prediction, we sought to recover the K vector from P_b^j matrix using an unbiased estimator as shown in subsequent sections.

$P^j_{h_j(i)}$ stands for the probability of the bin (meta-class) that i^{th} class is hashed into in j^{th} repetition. Our goal is to obtain an unbiased estimator of p_i in terms of $\{P^1_{h_1(i)}, P^2_{h_2(i)}, ..., P^R_{h_R(i)}\}$. From here on, the analysis diverges between Multiclass and Multilabel classification problems.

3.1 Multiclass Classification

We have

$$P_b^j = \sum_{i:h_j(i)=b} p_i; \forall j \quad and \quad 1 = \sum_{i=1}^{K} p_i = \sum_{b\in[B]} P_b^j \tag{1}$$

With the above equations, given the R classifier models, an unbiased estimator of p_i is:

Theorem 1.

$$\mathbb{E}\Big[\frac{B}{B-1}\Big[\frac{1}{R}\sum_{j=1}^{R}P^j_{h_j(i)}-\frac{1}{B}\Big]\Big]=Pr\Big(y=i\Big|x\Big)=p_i \tag{2}$$

Proof: Proof for this theorem has been given in (Medini et al., 2019). For clarity and coherence, we show the proof again here.

For any j, we can always write

$$P^j_{h_j(i)}=p_i+\sum_{k\neq i}\mathbf{1}_{h_j(k)=h_j(i)}p_k \tag{3}$$

where $\mathbf{1}_{h_j(k)=h_j(i)}$ is an indicator random variable (generically denoted by I_k from here on) suggesting whether class k has been hashed into the same bin as class i using hash function j. Since the hash function is universal, the expected value of the indicator is $\frac{1}{B}$ (each class will uniformly be binned into one of the B buckets). Thus

$$E(P^j_{h(i)})=p_i+\frac{1}{B}\sum_{k\neq i}p_k=p_i+(1-p_i)\frac{1}{B}$$

This is because the expression $\sum_{k\neq i}p_k=1-p_i$ as the total probability sum up to one. Simplifying, we get $p_i=\frac{B}{B-1}(E(P^j_{h_j(i)}-\frac{1}{B})$. Using linearity of expectation and the fact that $E(P^j_{h_j(i)})=E(P^k_{h_j(i)})$ for any $j\neq k$, it is not difficult to see that this value is also equal to

$$\mathbb{E}\Big[\frac{B}{B-1}\Big[\frac{1}{R}\sum_{j=1}^{R}P^j_{h_j(i)}-\frac{1}{B}\Big]\Big]$$

Let's denote our new estimator for p_i as $\hat{p_i}=\Big(\frac{B}{B-1}\Big)\Big[\frac{1}{R}\sum_{j=1}^{R}P^j_{h_j(i)}-\frac{1}{B}\Big]$

Theorem 2.

$$Var(\hat{p_i})\leq\Big(\frac{B-1}{B}\Big)^2\frac{(1-p_i)}{RB} \tag{4}$$

Proof: Using the known result $Var(aX+b)=a^2Var(X)$ and the fact that variance accumulates over sum of i.i.d random variables, we can write

$$Var(\hat{p_i})=\Big(\frac{B-1}{B}\Big)^2*\frac{1}{R^2}*R*Var(P^j_{h_j(i)}) \tag{5}$$

We first need to get $Var(P^j_{h_j(i)})$. From eqn. 3,

$$(P^j_{h_j(i)})^2=p_i^2+\sum_{k\neq i}I_kp_k^2+\sum_{k1\neq i}\sum_{k2\neq i}p_{k1}p_{k2}I_{k1}I_{k2}+2*p_i*\sum_{k\neq i}I_kp_k$$

Hence,

$$\mathbb{E}\Big[(P^j_{h_j(i)})^2\Big]=p_i^2+\frac{\sum_{k\neq i}p_k^2}{B}+\frac{\sum_{k1\neq i}\sum_{k2\neq i}p_{k1}p_{k2}}{B^2}+2*p_i*\frac{1-p_i}{B}\implies$$

$$\mathbb{E}\Big[(P^j_{h_j(i)})^2\Big]=p_i^2+\frac{\sum_{k\neq i}p_k^2}{B}+\frac{(1-p_i)(1-p_i)}{B^2}+2*p_i*\frac{1-p_i}{B}$$

and

$$\mathbb{E}\Big[P^j_{h_j(i)}\Big]^2=\Big(p_i+\frac{1-p_i}{B}\Big)^2=p_i^2+\frac{(1-p_i)^2}{B^2}+2*p_i*\frac{1-p_i}{B}$$

Therefore,

$$Var(P^j_{h_j(i)})=\mathbb{E}\Big[(P^j_{h_j(i)})^2\Big]-\mathbb{E}\Big[P^j_{h_j(i)}\Big]^2=\frac{\sum_{k\neq i}p_k^2}{B} \tag{6}$$

It's easy to see that $\sum_k p_k^2 \leq \sum_k p_k = 1 - p_i \implies \sum_{k \neq i} p_k^2 \leq 1 - p_i$. Hence, by merging eqns. 5 and 6, we get

$$Var(\hat{p}_i) \leq \Big(\frac{B-1}{B}\Big)^2 \frac{(1-p_i)}{RB}$$

We can observe that larger the original probability p_i, lower the variance of estimation which suggests that the higher probabilities are retained with high certainty and the lower probabilities are prone to noise. Since we only care for the correct prediction of the best class, we can offset the noise by increasing R.

For a d dimensional dataset (or d non-zeros for sparse data), the memory required by a vanilla logistic regression model (or any linear classifier) is $O(Kd)$. $O(Kd)$ is also the computational complexity of prediction. With MACH, the memory complexity is $O(BRd)$ and the computational complexity is $O(BRd + KR)$ (including inference). To obtain significant savings, we want BR to be significantly smaller than K. We next show that $BR \approx O(\log K)$ is sufficient for uniquely identifying the final class with high probability. Also, we need to tune the two knobs R and B for optimal performance on recovering the original probabilities. The subsequent theorems facilitate the prior knowledge of reasonable values of R and B based on our reconstruction error tolerance.

In (Medini et al., 2019), the following theorem has been proven

Theorem 3. *For any B, $R = \frac{\log \frac{K(K-1)}{2\delta_1}}{\log B}$, guarantees that all pairs of classes c_i and c_j are distinguishable from each other with probability greater than $1 - \delta_1$.*

The above theorem specifies a bound such that no two pair of classes end up in the same bucket on all R hash functions. While this is simple and intuitive, it does not take into account the ease of classification. To be precise, when the difference between the probability of best class and the 2^{nd} best class is low (predictions are spurious), it is much harder to identify the best class as oppposed to when the difference is higher. Theorem 3 is completely agnostic to such considerations.

Hence, the next theorems quantifies the requirements on R, B based on our tolerance to recovery error between p_i and $\hat{p}_i$ and also the ease of prediction (given by the difference between the p_i and p_j where i and j are the two best classes respectively).

Theorem 4. $P(|\hat{p}_i - p_i| < \epsilon| > 1 - \delta_2 \implies RB \geq \frac{1-p_i}{\delta_2 \epsilon^2}$

Proof: Chebyshev's inequality states that $P(|X - \mathbb{E}[X]| \geq \epsilon) \leq \frac{Var(X)}{\epsilon^2}$ for any random variable X. For our proposed unbiased estimator in theorem 1, we have

$$P(|p_i - \hat{p}_i| \geq \epsilon) \leq \frac{Var(X)}{\epsilon^2}$$

Using theorem 2, $P(|p_i - \hat{p}_i| \geq \epsilon) \leq \Big(\frac{B-1}{B}\Big)^2 \frac{(1-p_i)}{RB\epsilon^2} \implies P(|p_i - \hat{p}_i| < \epsilon) \geq 1 - \Big(\frac{B-1}{B}\Big)^2 \frac{(1-p_i)}{RB\epsilon^2}$

Hence, $P(|\hat{p}_i - p_i| < \epsilon| > 1 - \delta_2 \implies \Big(\frac{B-1}{B}\Big)^2 \frac{(1-p_i)}{RB\epsilon^2} < \delta_2$. For a large enough B, $\frac{B-1}{B} \approx 1$. Hence, we get the desired result

$$RB > \frac{1-p_i}{\delta_2 \epsilon^2}$$

If the best class i^* has $p_{i^*} > \alpha$ and we primarily care for recovering p_{i*} with high probability, then we have $RB > \frac{1-\alpha}{\delta_2 \epsilon^2}$

The next and final theorem (in multiclass learning) introduces the notion of 'Identifiability'.

Theorem 5. ***Identifiability:*** *If $RB \geq \frac{1-min(p_i,p_j)}{2\delta\epsilon^2}$ and classes i and j are the first and second best respectively ($p_i > p_j > p_k$ for every $k \neq i, j$), then $|p_i - p_j| > 2\epsilon \implies P(\hat{p}_i > \hat{p}_j) > (1-\delta)^2$*

Proof: We have

$$P(\hat{p}_i > \hat{p}_j) > P(\hat{p}_i > p_i - \epsilon) * P(\hat{p}_j < p_j + \epsilon)$$

But

$$P(\hat{p_i} > p_i - \epsilon) = P(|\hat{p_i} - p_i| < \epsilon) + 0.5 * P(|\hat{p_i} - p_i| \geq \epsilon) \implies$$
$$P(\hat{p_i} > p_i - \epsilon) = 0.5 * (1 + P(|\hat{p_i} - p_i| < \epsilon)) > 1 - \delta$$

Similarly $P(\hat{p_j} < p_j + \epsilon) > 1 - \delta$. Therefore, $P(\hat{p_i} > \hat{p_j}) > (1 - \delta)^2$.

Hence, based on the previous two theorems, we can get a reasonable estimate of what bucket size B should we choose and how many models that we need to train in parallel.

3.2 MULTILABEL CLASSIFICATION

The major difference between multi-class and multi-label classification from an analysis perspective is that eqn. 1 does not apply anymore. Hence, all the subsequent derivations do not apply in the case of multi-label classification. In the following theorems, we'll derive an approximate estimator using inclusion-exclusion principle to recover original probability vectors from MACH measurements for the case of multi-label classification.

Each p_i independently takes a value in $[0, 1]$. If we do not assume any relation between p_i, it would be very difficult to derive an estimator. The most realistic assumption on the probability vectors is sparsity. Most real datasets have only few labels per sample even when the number of classes K is huge. For the purpose of analysis, we will assume that $\sum_{i=1}^{K} p_i = V$ where V is the average of number of active labels per input.

Theorem 6.

$$\frac{(V + 1 - B) + \sqrt{(B - (V + 1))^2 - 4V + 4B\mathbb{E}[P^j_{h_j(i)}]}}{2} \approx Pr\left(y = i \middle| x\right) = p_i \tag{7}$$

Proof: $P^j_{h_j(i)}$ is the probability of union of all classes that have been hashed to bin $h_j(i)$ in j^{th} hash function. Hence, using inclusion-exclusion principle, it can be written as

$$P^j_{h_j(i)} = \sum_k p_k I_k - \sum_{k1<k2} \sum p_{k1 \cap k2} I_{k1} I_{k2} + \sum_{k1<k2<k3} \sum \sum p_{k1 \cap k2 \cap k3} I_{k1} I_{k2} I_{k3} - ...$$

Since all classes are independent of each other, we have

$$P^j_{h_j(i)} = \sum_k p_k I_k - \sum_{k1<k2} \sum p_{k1} p_{k2} I_{k1} I_{k2} + \sum_{k1<k2<k3} \sum \sum p_{k1} p_{k2} p_{k3} I_{k1} I_{k2} I_{k3} - ...$$

Since $I_i = 1$ $w.p.$ 1, we have

$$P^j_{h_j(i)} = p_i + \sum_{k \neq i} p_k I_k - p_i \sum_{k \neq i} p_k I_k - \sum_{k1<k2;k1 \neq k2 \neq i} \sum p_{k1} p_{k2} I_{k1} I_{k2} +$$
$$p_i \sum_{k1<k2;k1 \neq k2 \neq i} \sum p_{k1} p_{k2} I_{k1} I_{k2} + \sum_{k1<k2<k3;k1 \neq k2 \neq k3 \neq i} \sum \sum p_{k1} p_{k2} p_{k3} I_{k1} I_{k2} I_{k3} - ...$$

Aggregating similar terms, we get

$$P^j_{h_j(i)} = 1 - (1 - p_i) + (1 - p_i) \sum_{k \neq i} p_k I_k \; - \; (1 - p_i) \sum_{k1<k2;k1 \neq k2 \neq i} \sum p_{k1} p_{k2} I_{k1} I_{k2} +$$
$$(1 - p_i) \sum_{k1<k2<k3;k1 \neq k2 \neq k3 \neq i} \sum \sum p_{k1} p_{k2} p_{k3} I_{k1} I_{k2} I_{k3} - ... \implies$$

$$P^j_{h_j(i)} = 1 - (1 - p_i)\left[1 - \sum_{k \neq i} p_k I_k + \sum_{k1<k2;k1 \neq k2 \neq i} \sum p_{k1} p_{k2} I_{k1} I_{k2} - ...\right]$$

Therefore, $\mathbb{E}[P^j_{h_j(i)}] = 1 - (1 - p_i)\left[1 - \frac{\sum_{k \neq i} p_k}{1!\ B} + \frac{\sum_{k1 \neq k2 \neq i} \sum p_{k1} p_{k2}}{2!\ B^2} - ...\right]$

In typical multilabel dataset, K runs into the order of millions where B is a few thousands. If we ignore all terms with B in denominator, we essentially end up with a plain mean estimator ($\hat{p_i} = \frac{1}{R}\sum_{j=1}^{R} P^j_{h_j(i)}$). We ideally want to use all terms but it is very cumbersome to analyze the summation (please note that the summation doesn't simplify to exponential as we have the clause $k_j \neq k_l$ in each summation). In our case, we empirically show later on that even by limiting the expression to first order summation (ignore all terms B^2 or higher powers of B in denominator), we get a much better estimator for true probability.

We can simplify the above expression into $\mathbb{E}[P^j_{h_j(i)}] = 1 - (1 - p_i)\left[1 - \frac{V-p_i}{B}\right]$.

Solving for p_i, we get our desired result $p_i = \frac{(V+1-B)+\sqrt{(B-V-1)^2-4V+4B\mathbb{E}[P^j_{h_j(i)}]}}{2}$.

Unfortunately, proposing an unbiased estimator using the above result is hard. One intuitive estimator that can potentially work is $\hat{p_i} = \frac{(V+1-B)+\sqrt{(B-V-1)^2-4V+\frac{4B}{R}\sum_{j=1}^{R} P^j_{h_j(i)}}}{2}$.

Using Jensen's inequality (specifically, $\mathbb{E}[\sqrt{X}] \leq \sqrt{\mathbb{E}[X]}$),

$$\mathbb{E}[\hat{p_i}] \leq \frac{(V+1-B)+\sqrt{(B-V-1)^2-4V+4B\mathbb{E}[P^j_{h_j(i)}]}}{2} = p_i$$

Hence, $\mathbb{E}[\hat{p_i}] \leq p_i$ and we do not have an unbiased estimator. Nevertheless, the next section details simulation experiments that corroborate that our proposed estimator for multilabel classification has much lower mean-squared-error (MSE) than a plain mean estimator.

4 EXPERIMENTS

To simulate the setup for multi-label MACH, we perform the following steps:

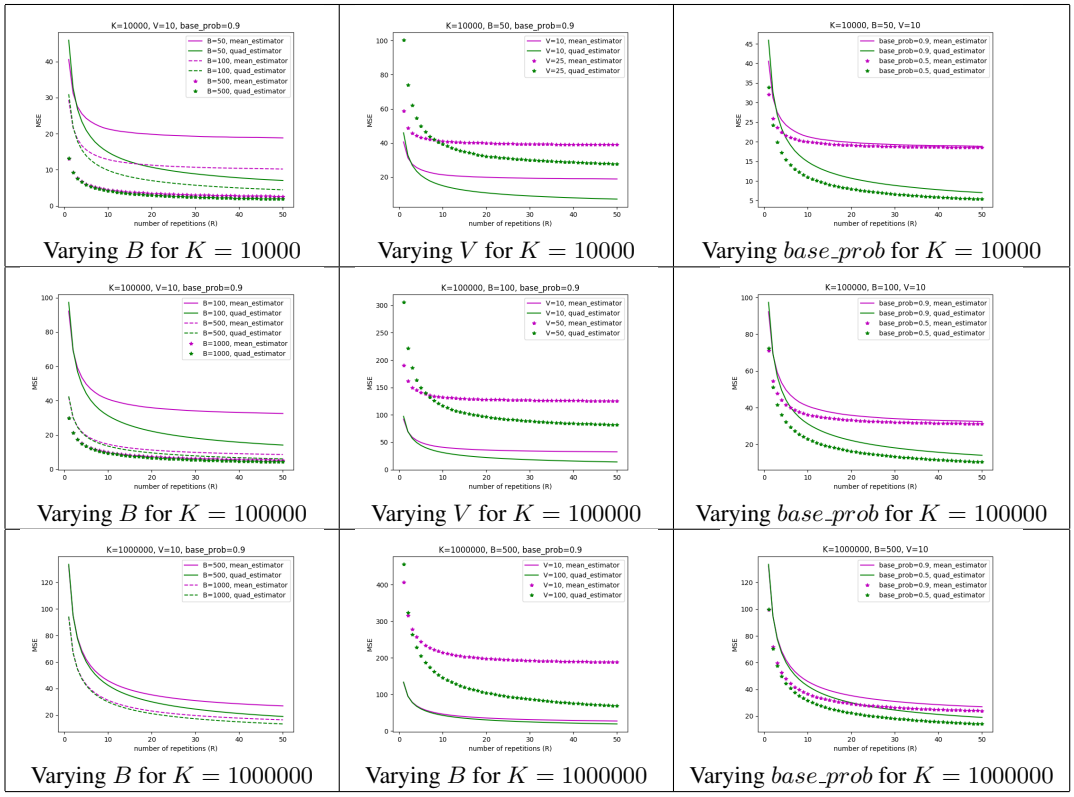
- Choose a $base\_prob \in (0, 1]$ which says how confident the prediction in the original probability vector is.
- Initialize a K dimensional vector $\overline{p\_orig} = (p_1, p_2, ..., p_K)$ with all zeros. We then implant the value $base\_prob$ in $int(\frac{V}{base\_prob})$ number of random locations. We now have a vector $\overline{p\_orig}$ which obeys $\sum p_i = V$.
- Generate 1000 samples of K dimensional label vectors where each dimension i is a Bernoulli random variable with probability p_i. These sample labels are realizations of $\overline{p\_orig}$.
- Merge each sample label vector into B dimensional binary labels where a bucket b is an OR over the constituent classes $\{i : h_j(i) = b\}$. We repeat this step for R different hash functions ,i.e., for all $j \in 1, 2, ..., R$.
- For each of R repetitions, calculate the mean of the respective B dimensional labels to get $P^j = (P^j_1, P^j_2, ..., P^j_B)$
- Reconstruct $\overline{p\_approx}$ using theorem 6 and $\{P^j : j = 1, 2, .., R\}$.
- Calculate L2-norm of $\overline{p\_orig} - \overline{p\_approx}$
- Repeat all above steps for 10000 times (generating a different $\overline{p\_orig}$ each time) and report the average L2-norm from the last step (it serves as the reconstruction MSE, lower the better).

4.1 DISCUSSION OF RESULTS:

Following the above steps, we show the comparison of our proposed quadratic estimator in theorem 6 against the plain mean estimator by varying the values of K, B, V and $base\_prob$ in figure 3. We can infer the following insights from the plots :

- As K increases, the MSE grows. This is expected because the reconstructed vector has a small non-zero probability for many of the K classes and this induces noise and hence MSE grows. But the top classes are still retrieved with high certainty.

- For any $K, V, base\_prob$, the MSE decreases when B increases which is expected (fewer collisions of classes and hence less noisier predictions). As the MSE gets lower, the gains from the square-root estimator are also low. This is good because in scenarios where B and R are small, we can do much better recovery using the proposed estimator.
- For any $K, B, base\_prob$ the MSE increases with V. This is again natural because larger V induces more 'true' class collisions and hence the retrieval becomes fuzzy.
- For any K, B, V the MSE decreases with $base\_prob$, albeit with much little difference than previous cases. This is interesting because a high $base\_prob$ means that we have few but highly confident 'true' classes among K. On the other hand, lower $base\_prob$ indicates that 'true' classes are scattered among a larger subset among K classes. Yet, MACH recovers the original probabilities with commendably low MSE.



Varying B for $K = 10000$ | Varying V for $K = 10000$ | Varying $base\_prob$ for $K = 10000$

Varying B for $K = 100000$ | Varying V for $K = 100000$ | Varying $base\_prob$ for $K = 100000$

Varying B for $K = 1000000$ | Varying B for $K = 1000000$ | Varying $base\_prob$ for $K = 1000000$

Figure 3: Reconstruction Error (MSE) comparison between 1) vanilla mean estimator (plotted in magenta) and 2) proposed square-root estimator (plotted in green); for various configurations of K,B and V. The value of K varies as 10000, 100000, 1000000 for the 1^{st}, 2^{nd} and 3^{rd} rows respectively. In each row, the first plot fixes $V, base\_prob$ and compares various values of B. The 2^{nd} plot fixes $B, base\_prob$ and compares different values of B. The 3^{rd} one fixes B, V and compares different values of $base\_prob$. In all cases, we notice that the square-root estimator is consistently and significantly lower in MSE than the corresponding mean estimator.

5 Conclusion

We perform a rigorous theoretical analysis of using Count-Min-Sketch for Extreme Classification and come up with error bounds and hyper-parameter constraints. We identify a critical shortcoming of reconstruction estimators proposed in prior research. We overcome the shortcoming by treating each bucket in a hash table as a union of merged original classes. Using inclusion-exclusion principle and a controlled label sparsity assumption, we come up with an approximate estimator to reconstruct original probability vector from the predicted Count-Min Sketch measurements. Our new estimator has significantly lower reconstruction MSE than the prior estimator.

REFERENCES

Rohit Babbar and Bernhard Schölkopf. Dismec: Distributed sparse machines for extreme multi-label classification. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*, pp. 721–729. ACM, 2017.

Richard G Baraniuk. Compressive sensing [lecture notes]. *IEEE signal processing magazine*, 24 (4):118–121, 2007.

Kush Bhatia, Himanshu Jain, Purushottam Kar, Manik Varma, and Prateek Jain. Sparse local embeddings for extreme multi-label classification. In *Advances in neural information processing systems*, pp. 730–738, 2015.

Anna E Choromanska and John Langford. Logarithmic time online multiclass prediction. In *Advances in Neural Information Processing Systems*, pp. 55–63, 2015.

Graham Cormode and Shan Muthukrishnan. An improved data stream summary: the count-min sketch and its applications. *Journal of Algorithms*, 55(1):58–75, 2005.

Thomas G Dietterich and Ghulum Bakiri. Solving multiclass learning problems via error-correcting output codes. *Journal of artificial intelligence research*, 2:263–286, 1995.

Daniel J Hsu, Sham M Kakade, John Langford, and Tong Zhang. Multi-label prediction via compressed sensing. In *Advances in neural information processing systems*, pp. 772–780, 2009.

Himanshu Jain, Yashoteja Prabhu, and Manik Varma. Extreme multi-label loss functions for recommendation, tagging, ranking & other missing label applications. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 935–944. ACM, 2016.

Himanshu Jain, Venkatesh Balasubramanian, Bhanu Chunduri, and Manik Varma. Slice: Scalable linear extreme classifiers trained on 100 million labels for related searches. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, pp. 528–536. ACM, 2019.

Tharun Medini, Qixuan Huang, Yiqiu Wang, Vijai Mohan, and Anshumali Shrivastava. Simultaneous matching and ranking as end-to-end deep classification: A case study of information retrieval with 50m documents. In *Advances in Neural Information Processing Systems*, pp. 55–63, 2019.

Priyanka Nigam, Yiwei Song, Vijai Mohan, Lakshman Vihan, Ding Weitan, Shingavi Ankit, Choon Hui Teo, Hao Gu, and Bing Yin. Semantic product search. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 2876–2885. ACM, 2019.

Yashoteja Prabhu, Anil Kag, Shilpa Gopinath, Kunal Dahiya, Shrutendra Harsola, Rahul Agrawal, and Manik Varma. Extreme multi-label learning with label features for warm-start tagging, ranking & recommendation. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, pp. 441–449. ACM, 2018a.

Yashoteja Prabhu, Anil Kag, Shrutendra Harsola, Rahul Agrawal, and Manik Varma. Parabel: Partitioned label trees for extreme classification with application to dynamic search advertising. In *Proceedings of the 2018 World Wide Web Conference*, pp. 993–1002. International World Wide Web Conferences Steering Committee, 2018b.

Shashanka Ubaru and Arya Mazumdar. Multilabel classification with group testing and codes. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 3492–3501. JMLR. org, 2017.

Shashanka Ubaru, Arya Mazumdar, and Alexander Barg. Group testing schemes from low-weight codewords of bch codes. In *2016 IEEE International Symposium on Information Theory (ISIT)*, pp. 2863–2867. IEEE, 2016.

Avinash Vem, Nagaraj T Janakiraman, and Krishna R Narayanan. Group testing using left-and-right-regular sparse-graph codes. *arXiv preprint arXiv:1701.07477*, 2017.

Kilian Weinberger, Anirban Dasgupta, John Langford, Alex Smola, and Josh Attenberg. Feature hashing for large scale multitask learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pp. 1113–1120. ACM, 2009.

Hamed Zamani, Mostafa Dehghani, W Bruce Croft, Erik Learned-Miller, and Jaap Kamps. From neural re-ranking to neural ranking: Learning a sparse representation for inverted indexing. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, pp. 497–506. ACM, 2018.

A APPENDIX

Why not Compressive Sensing or Count-Sketch? The measurements in Compressive Sensing are not a probability distribution but rather a few linear combinations of original probabilities. Imagine a set of classes $\{cats,\ dogs,\ cars,\ trucks\}$. Suppose we want to train a classifier that predicts a compressed distribution of classes like $\{0.6 * cars + 0.4 * cats,\ 0.5 * dogs + 0.5 * trucks\}$. There is no intuitive sense to these classes and we cannot train a model using softmax-loss which has been proven to work the best for classification. We can only attempt to train a regression model to minimize the norm(like L_1-norm or L_2-norm) between the projections of true K-vector and the predicted K-vectors(like in the case of (Hsu et al., 2009)). This severely hampers the learnability of the model as classification is more structured than regression. On the other hand, imagine two union-classes $\{[cars\ and\ trucks], [cats\ and\ dogs]\}$. It is easier for a model to learn how to predict whether a data point belongs to '$cars\ and\ trucks$' because unions are well defined. Count-Min Sketch facilitates exactly this concept of union of classes.