Sketching Methods for Extreme Classification in Log-Memory

Anonymous Author(s)
Affiliation
Address
email

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 primary 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. However, there are two strong assumptions that we take exception to: **1)** the proposed estimator to recover label probabilities assumes that the output layer gives softmax probabilities for the classes, which is not true for multilabel learning. To that effect, we propose a novel quadratic-approximation based estimator using Inclusion-Exclusion Principle, having significantly lower reconstruction error than the typical CMS estimator across various values of number of classes, label sparsity and compression ratio. **2)** MACH claims that there is no other sketch that can be trained as a classifier except CMS. We show that Count-Sketch (CS) can also be trained as a classifier and we achieve nearly the same precision as CMS on popular Extreme Classification datasets. Unlike CMS, CS is unbiased and could potentially be used to train large hidden layers.

1 Introduction

Extreme Classification has taken center-stage of Data Mining and Information Retrieval research in the past few years [22, 16, 11, 5]. In extreme classification, we solve the vanilla multiclass and multilabel classification problems but with the number of classes K being 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 [16, 11, 1], tree based methods [15, 12], embedding models [14, 3], 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 [17, 18, 20, 10]. 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

Submitted to 34th Conference on Neural Information Processing Systems (NeurIPS 2020). Do not distribute.

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) [13] 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) [6] matrix of the original probability vector. For the case of multiclass classification (where every input has just a single label unlike multilabel classification), 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. We give an upper bound on the variance of estimation and lower bounds and hash table parameters for a given level of error tolerance. **2)** 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'). 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. **3)** We show how to train with measurements of Count-Sketch instead of Count-Min Sketch. We compare both the popular sketching methods on popular Extreme Classification [19] datasets.

2 Background

Count-Min Sketch: Count-Min Sketch (CMS) [6] is one of the most popular algorithms 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 among 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 [6], we basically assign $O(\log K)$ 'signatures' to each class using universal hash functions. We use $R = O(\log K)$ different hash functions $H_1, H_2, H_3, ..., H_R$, 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 $R * B$. If we encounter class i in the stream of classes, we increment the counts in cells $H_1(i), H_2(i)....., H_R(i)$. 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.

	H_1	H_2	H_3	H_4
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
H_1	0	1+1+1+1 = 4	0	1+1 = 2	0	0	1
H_2	0	0	1+1 = 2	0	1+1 = 2	0	1+1+1 = 3
H_3	0	1+1=2	0	1+1+1 = 3	1+1 = 2	0	0
H_4	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 are on the left and the accumulated counts for each of the letter in the stream are 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., $n_{approx}(a_1) = min(C[1, H_1(i)], C[2, H_2(i), ..., C[R, H_R])$.

An example illustration of CMS is shown in figure 1.

Count-Sketch: Count-Sketch (CS) is a de-biasing variant of Count-Min Sketch. In addition to the main hash functions $H_1, H_2, ...H_R$, CS also uses auxiliary hash functions $G_1, G_2, ..., G_R$ each of

which map a class i to $\{-1, 1\}$. When we encounter a class i in the input stream, we increment the count in cell $H_j(i)$ if $G_j(i) = +1$ and decrement the count if $G_j(i) = -1$. During inference, the min-operation in CMS is replaced by mean ,i.e., $n_{approx}(a_1) = mean(G_1(i) * C[1, H_1(i)],\ G_2(i) * C[2, H_2(i),\ ....,\ G_R(i) * C[R, H_R])$

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 vectors, 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 [2]. The connection between compressive sensing and extreme classification was identified in prior works [9, 7].

Why not Compressive Sensing? MACH[13] argues that only when the compressed measurements correspond to a probability distribution, a classifier can be trained to predict them. Otherwise, only a regression model can be trained which is not a desirable scenario particularly at high dimensions. Compressive Sensing measurements are a few linear combinations of original probabilities. On the other hand, CMS measurements are probabilities of unions of classes. Hence, it is easier to train a classifier to predict CMS measurements.

Can we do Count-Sketch? The authors also argue that the auxiliary hash functions in Count-Sketch prevent the notion of union of classes and hence it cannot be used to train classifiers. However, thanks to the label sparsity in real Extreme Classification problems, the compressed measurements of label vectors only have few non-zeros from $\{-1, 1\}$ (barring very few collisions for the same input). Using the map $\{-1- > 0,\ 0- > 0.5,\ +1- > 1\}$, we train a binary cross-entropy and show that it achieves nearly the same performance as CMS (in section 4.2).

2.1 Merged Average Classifiers via Hashing (MACH)

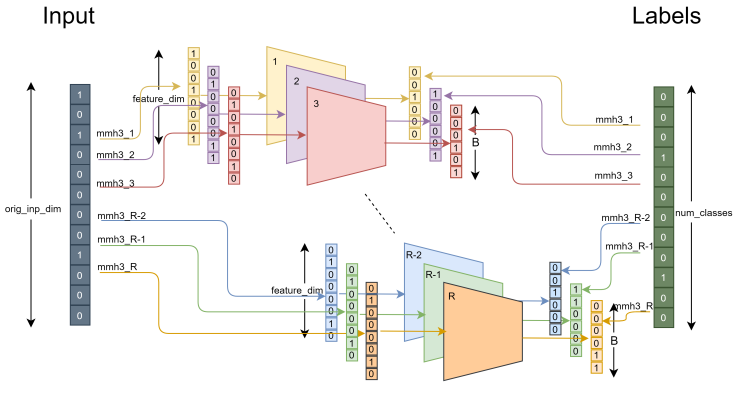

Figure 2: Schematic diagram of MACH. Both the input and the label vector are independently hashed R times (label vector is hashed from K to B, K being number of classes and B being number of buckets in each of the R hash tables). Small models are then trained in parallel.

MACH [13] 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 run 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 [21] 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.

3 Theoretical Analysis

In this section, we first formalize MACH and discuss the reconstruction estimator in detail. We provide various error bounds that help us choose optimal hyper-parameters. MACH assumes that that there is no dependence among the classes. However, labels in extreme classification usually have strong correlations. In section 3.2, we propose a new estimator (theorem 6) using Inclusion-Exclusion principle for probability of unions and show that it has lower reconstruction MSE than MACH's estimator. Our new estimator relaxes the necessity of having a probability distribution over the labels (which is intrinsically violated in multilabel learning).

Let there be K classes originally. We will 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

For a multiclass setting (with only one true label per input), 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. $\mathbb{E}\left[\frac{B}{B-1}\left[\frac{1}{R}\sum_{j=1}^{R} P^j_{h_j(i)} - \frac{1}{B}\right]\right] = Pr\left(y=i\middle|x\right) = p_i$

Proof: Proof for this theorem has been given in [13]. For coherence, we show the proof again here.

· For any j, we can always write $P^j_{h_j(i)} = p_i + \sum_{k\neq i} \mathbf{1}_{h_j(k)=h_j(i)}\ p_k$ 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 sums 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

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

Let's denote our new estimator for p_i as $\hat{p_i} = \left(\frac{B}{B-1}\right)\left[\frac{1}{R}\sum_{j=1}^{R} P^j_{h_j(i)} - \frac{1}{B}\right]$. The subsequent four theorems give bounds on variance and hyper-parameters (R, B) given a level of error tolerance. Due to space constraints, all the proofs are given in Appendix A.

Theorem 2. $Var(\hat{p_i}) \leq \left(\frac{B-1}{B}\right)^2 \frac{(1-p_i)}{RB}$

Theorem 2 says 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 and computational complexity of vanilla logistic regression model (or any linear classifier) is $O(Kd)$. 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.

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 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 opposed to when the difference is higher. Theorem 3 is completely agnostic to such considerations.

Hence, the next theorems quantify 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. $P(|\hat{p_i} - p_i| < \epsilon| > 1 - \delta_2 \implies RB \geq \frac{1-p_i}{\delta_2 \epsilon^2}$

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 $p_i > p_j$, then $|p_i - p_j| > 2\epsilon \implies P(\hat{p_i} > \hat{p_j}) > (1 - \delta)^2$*

Here, if p_i, p_j represent the top 2 classes, then $p_i - p_k > p_i - p_j \; \forall \; k \neq i, j$. Hence, $P(\hat{p_i} > \hat{p_k}) > (1 - \delta)^2 \; \forall \; k$.

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 equation 1 does not apply anymore. Hence, all the subsequent derivations do not apply in the case of multi-label classification. In the following theorem, we will 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.

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

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 the quadratic equation for p_i, we get our desired result

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

Let $(B - V - 1)$ be replaced by M for simplicity. Unfortunately, proposing an unbiased estimator using the above result is hard. The most intuitive estimator that can potentially work the best is

$$\hat{p_i} = \frac{-M + \sqrt{M^2 - 4V + \frac{4B}{R} \sum_{j=1}^{R} P^j_{h_j(i)}}}{2}$$

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

$$\frac{-M + \mathbb{E}\left[\sqrt{M^2 - 4V + \frac{4B}{R}\sum_{j=1}^{R} P^j_{h_j(i)}}\right]}{2} \leq \frac{-M + \sqrt{M^2 - 4V + 4B\mathbb{E}[P^j_{h_j(i)}]}}{2}$$

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

We split our experiments into two parts: 1) Comparison of reconstruction Mean-Squared Error (MSE) for MACH's mean estimator (theorem 1) vs our new quadratic estimator (theorem 6). 2) Comparison of precision on popular Extreme Classification datasets for original MACH, our quadratic estimator and Count-Sketch training.

4.1 Reconstruction MSE: MACH Estimator vs Quadratic Estimator

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

1) Initialize a K dimensional vector $\overline{p\_orig} = (p_1, p_2, ..., p_K)$ with all zeros. We need to implant values to $\overline{p\_orig}$ such that $\sum p_i = V$.

2) Choose a value $base\_prob \in (0, 1]$ which governs how spread out the values are in $\overline{p\_orig}$. The higher $base\_prob$ is, the fewer non-zeros $\overline{p\_orig}$ has.

3) Implant the value $base\_prob$ in $int(\frac{V}{base\_prob})$ number of random locations. $\overline{p\_orig}$ now has the desired property $\sum p_i = V$.

3) 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 $\overline{p\_orig}$.

4) 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 $j \in 1, 2, ..., R$.

5) For each of R repetitions, calculate the mean of the respective B dimensional labels to get $P^j = (P^j_1, P^j_2, ..., P^j_B)$

6) Reconstruct $\overline{p\_approx}$ using theorem 6 and $\{P^j : j = 1, 2, .., R\}$.

7) Calculate L2-norm of $\overline{p\_orig} - \overline{p\_approx}$.

8) 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).

Results: Figure 3 shows the comparison of our Quadratic Estimator (in green) against the plain mean estimator (in magenta) for various values of K,V and B (more comparisons in Appendix B). We can observe the consistent pattern of Quadratic Estimator having lower MSE than the mean estimator. Additionally, we can observe that MSE decreases as B increases, which is expected due to fewer collisions. At smaller B, we see a larger gap between the two estimators which further add strength to our estimator.

4.2 Extreme Classification Datasets

We now validate our proposal on three large popular Extreme Classification datasets Wiki10-31K, Delicious-200K and AmazonTitles-670K [19]. We train $R = 16$ feed forward networks for each dataset with varying values of B. We use Tensorflow v1.14 and train with 2 V-100 GPUs. We compare the precision@1/3/5 for all the datasets for both Mean and Quadratic Estimator. In addition, we also train to predict Count-Sketches instead of Count-Min Sketch. In order to facilitate training a classifier with Count-Sketch, we perform the following modifications to the label vectors:

1) Project the label vectors to a $B * R$ Count-Sketch matrix (similar to CMS as shown in Figure 2). Unlike CMS, the CS matrix will have negative values (due to the auxiliary hash functions). However, most values will still be 0 with only a few being $+1$ and few others -1.

2) Add $+1$ to all the elements of CS matrix and divide by 2.

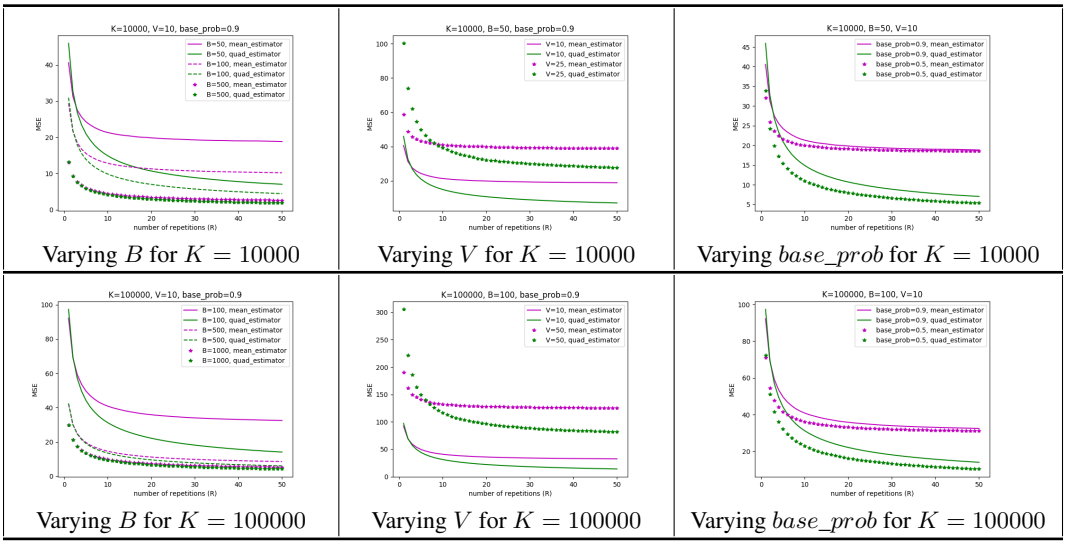

Figure 3: Reconstruction Error (MSE) comparison between 1) vanilla mean estimator (plotted in magenta) and 2) proposed square-root estimator (plotted in green); for various configurations of K,B and V. In all cases, we notice that the square-root estimator is consistently and significantly lower in MSE than the corresponding mean estimator.

3) Clip all values ≤ 0 to 0 and all values ≥ 1 to 1.

4) Train a binary cross-entropy loss function in the same fashion as with CMS

5) During inference, assign an aggregate score for class i by summing the probability values weighted by the auxiliary hash functions., i.e, $score_i = \sum_r G_r(i) * P^j_{h_j(i)}$.

Please note that we only need the order of predictions to be preserved irrespective of the reconstruction MSE for precision@1/3/5 to be high.

Results: Table 1 shows the precision performance on the three real classification tasks. We observe that in all cases, Quadratic Estimator has consistent superiority over the original Mean-Estimator. The last column shows the results for Count-Sketch training. While Count Sketch performs well for the 2 larger datasets, P@5 on Wiki10-31K is on the lower side. However, Count-Sketch outperforms Count-Min Sketch for Delicious-200K making it a valid and viable sketch for training large neural networks.

	Mean Estimator			Quadratic Estimator			CS+Weighted Estimator		
	P@1	P@3	P@5	P@1	P@3	P@5	P@1	P@3	P@5
Wiki10-31K (B=1K)	81.48	60.73	50.14	**81.89**	**65.18**	**55.44**	80.77	60.12	46.23
Delicious-200K (B=5K)	37.61	35.39	33.70	**38.81**	36.67	35.06	38.66	**36.75**	**35.52**
AmazonTitles-670K (B=10K)	33.79	30.49	28.57	**34.86**	**32.25**	**30.94**	33.93	31.67	28.93

Table 1: Comparison of various estimators on 3 public extreme classification datasets. We see that in all cases, Quadratic Estimator outperforms the 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 a new approximate 'Quadratic 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. When evaluated on 3 large Extreme Classification datasets, our Quadratic Estimator has better precision performance. Additionally, we also show that it is possible to train with Count-Sketches as a binary classifier and achieve the same performance as Count-Min Sketch.

Broader Impact

This paper contributes a fundamental strategy of using sketching algorithms to compress huge output layers in classification tasks. In particular, we show that an unbiased sketching algorithm like Count-Sketch can be as good as Count-Min Sketch to be trained as a classifier. In the last two years, several novel techniques have been proposed to reduce the computations in the hidden layers of a neural network like Lottery-Ticket Hypothesis [8] and SLIDE [4]. Our proposal of using Count-Sketches can be extended to end-to-end training with hidden layer compression.

Lottery-ticket Hypothesis obtains a network with lower memory footprint while SLIDE adaptively samples active neurons specific to each input in constant time. We believe applying sketching to compress hidden layers has both the advantages and our work can be a foundation for further research in network compression.

References

[1] 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*, pages 721–729. ACM, 2017.

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

[3] 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*, pages 730–738, 2015.

[4] Beidi Chen, Tharun Medini, James Farwell, Sameh Gobriel, Charlie Tai, and Anshumali Shrivastava. Slide : In defense of smart algorithms over hardware acceleration for large-scale deep learning systems. In *Proceedings of Machine Learning and Systems 2020*, pages 291–306. 2020.

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

[6] 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.

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

[8] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv preprint arXiv:1803.03635*, 2018.

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

[10] Qixuan Huang, Tharun Medini, Yiqiu Wang, Vijai Mohan, and Anshumali Shrivastava. Extreme classification in log memory. *arXiv preprint arXiv:1910.13830*, 2019.

[11] 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*, pages 528–536. ACM, 2019.

[12] 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*, pages 935–944. ACM, 2016.

[13] Tharun Kumar Reddy Medini, Qixuan Huang, Yiqiu Wang, Vijai Mohan, and Anshumali Shrivastava. Extreme classification in log memory using count-min sketch: A case study of amazon search with 50m products. In *Advances in Neural Information Processing Systems*, pages 13244–13254, 2019.

[14] 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*, pages 2876–2885. ACM, 2019.

[15] 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*, pages 441–449. ACM, 2018.

[16] 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*, pages 993–1002. International World Wide Web Conferences Steering Committee, 2018.

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

[18] 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)*, pages 2863–2867. IEEE, 2016.

[19] Manik Varma. Extreme Classification Repository. `http://manikvarma.org/downloads/XC/XMLRepository.html`, 2014.

[20] 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.

[21] 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*, pages 1113–1120. ACM, 2009.

[22] 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*, pages 497–506. ACM, 2018.