Stochastic Negative Mining for Learning with Large Output Spaces

Sashank J. Reddi[†], **Satyen Kale**[†], **Felix Yu**[†], **Dan Holtmann-Rice**[†]
Jiecao Chen*, **Sanjiv Kumar**[†]
[†]Google Research, NY *Indiana University, IN

Abstract

We consider the problem of retrieving the most relevant labels for a given input when the size of the output space is very large. Retrieval methods are modeled as set-valued classifiers which output a small set of classes for each input, and a mistake is made if the label is not in the output set. Despite its practical importance, a statistically principled, yet practical solution to this problem is largely missing. To this end, we first define a family of surrogate losses and show that they are *calibrated* and *convex* under certain conditions on the loss parameters and data distribution, thereby establishing a statistical and analytical basis for using these losses. Furthermore, we identify a particularly intuitive class of loss functions in the aforementioned family and show that they are amenable to practical implementation in the large output space setting (i.e. computation is possible without evaluating scores of all labels) by developing a technique called *Stochastic Negative Mining*. We also provide generalization error bounds for the losses in the family. Finally, we conduct experiments which demonstrate that Stochastic Negative Mining yields benefits over commonly used negative sampling approaches.

1 INTRODUCTION

Recently, machine learning problems with extremely large output spaces have become ubiquitous: for example, extreme multiclass or multilabel classification problems with many classes, language modeling with big vocabularies, etc. Information retrieval tasks such as retrieving the most relevant documents for a given query can also be viewed as machine learning problems of this type.

In this paper we specifically consider retrieval tasks where the objective is to output the k most relevant classes for an input out of a very large number of possible classes. Training and test examples consist of pairs (x, y) where x represents the input and y is *one* class that is relevant for it. This setting is common in retrieval tasks: for example, x might represent a search query, and y a document that a user clicked on in response to the search query. The goal is to learn a set-valued classifier that for any input x outputs a set of k classes that it believes are most relevant for x, and the model is evaluated based on whether the class y is captured in these k classes.

Typically, machine learning models for such problems take the form of scoring functions that assign a real valued score to each possible output class for a given input indicating how relevant it is to the input, and outputting the classes with top k scores. Such models are trained using standard loss functions for training classifiers such as softmax cross entropy loss, max margin loss, etc. The key challenge here is that evaluating such losses requires computing the scores of all possible classes. When the number of classes becomes very large (say, in the order of hundreds of thousands or more), this makes training as well as inference very expensive.

A variety of techniques have been developed to deal with this problem such as sampled softmax (Jean et al., 2015), negative sampling (Mikolov et al., 2013), tree based approaches (Daumé III et al., 2017), etc. Many of these approaches are designed for the softmax cross entropy loss, and while these methods exhibit good practical performance in few cases, many of them are biased and may not converge to the optimal solution in the limit (some notable exceptions are (Raman et al., 2016; Fagan and Iyengar, 2018)). For the retrieval problem that we consider in this paper, to the best of our knowledge, there is no analysis of such methods for their statistical validity for the problem. Motivated

Proceedings of the 22nd International Conference on Artificial Intelligence and Statistics (AISTATS) 2019, Naha, Okinawa, Japan. PMLR: Volume 89. Copyright 2019 by the author(s).

by class ambiguity in image classification tasks, Lapin et al. (2015) considered the same retrieval problem as in this paper, and designed the top-k multiclass SVM algorithm for it by defining the top-k hinge loss function, which unfortunately does not scale to extremely large output spaces since computing it requires computing scores of all labels.

All the aforementioned methods suffer from either statistical or scalability issues. Most practical works on large-output often resort to some variant of *negative sampling* due to its simplicity. Negative sampling approaches randomly sample a few classes other than the given positive class and treat them as negative classes, possibly with some additional correction, while computing the loss (Jean et al., 2015; Mikolov et al., 2013). Statistical performance issues with negative sampling approach have recently motivated the use of some heuristics based on mining negatives with large scores, broadly referred to semi-hard negative mining (Schroff et al., 2015; Bai et al., 2017). While such approaches improve the performance in some cases, they are not statistically grounded. In this paper, we develop a statistically sound and scalable approach based on the principle of mining scores from few randomly sampled negatives, and provide theoretical and empirical basis for using it over standard negative sampling approaches.

To this end, we begin with designing loss functions for the retrieval problem that have desirable statistical properties. In particular, we define a family of loss functions called *Ordered Weighted Losses* (OWLs). We provide a statistical analysis of these loss functions and show that they satisfy several desirable properties under mild conditions. Furthermore, we provide a negative mining approach, called *Stochastic Negative Mining*, to efficiently optimize an instance of OWLs. More specifically, our contributions are the following:

1. **Calibration.** We show OWLs are calibrated and therefore training models by minimizing OWLs leads to the Bayes optimal predictor in the limit as the model capacity increases and the number of samples grows.

2. **Convexity.** We show that OWLs are convex in the score vector. Thus for linear and kernel models for computing scores, minimizing the training loss is a convex problem in the model parameters and thus standard convex optimization techniques can be used.

3. **Surrogate loss.** We show that OWLs are valid surrogate losses for the 0/1 retrieval loss of interest.

4. **Practical implementability.** We provide an instance of OWLs that can be efficiently optimized through Stochastic Negative Mining. This technique samples a set of classes and treats the highest scoring ones in the set as "negative" classes for the training example; thereby, avoiding evaluating scores of all labels. This technique has intuitive appeal and has been empirically observed to yield good performance since it avoids computing scores of all the possible output classes.

5. **Generalization error bounds.** We provide generalization error bounds for OWLs that provide guidance on how to choose OWL parameters. We also provide margin bounds for the retrieval loss for arbitrary hypothesis classes in terms of their Gaussian and Rademacher complexities.

6. **Experimental validation.** We provide experimental evidence that Stochastic Negative Mining does indeed help improve performance when learning with large output spaces compared to simpler sampling based strategies.

1.1 Related work

There is extensive literature on the problem of learning set-valued classifiers: see, for example, Grycko (1993); del Coz et al. (2009); Vovk et al. (2005); Wu et al. (2004); Lei et al. (2013), particularly in the context of *binary classification with a reject option* (Chow, 1970; Herbei and Wegkamp, 2006; Bartlett and Wegkamp, 2008; Yuan and Wegkamp, 2010). del Coz et al. (2009) specifically considered using such classifiers in the information retrieval context. Denis and Hebiri (2017) considered a similar setting to the one in this paper and provided a procedure with bounded expected size of the output set and establish statistical optimality of the procedure. Sadinle et al. (2018) provide characterizations of optimal set-valued classifiers with user-defined levels of coverage or confidence and estimators with good asymptotic and finite sample properties.

Another related setting is the *learning to rank* problem (see e.g. Joachims (2005); Agarwal (2011); Boyd et al. (2012); Kar et al. (2015) and the references therein). The objective there is to rank a given set of items so that relevant items are ranked as highly as possible. Training data consist of items along with a binary label indicating whether the item is relevant or not. Various performance metrics are considered such as Precision@k (the fraction of the top k ranked items that are relevant), the normalized discounted cumulative gain (NDCG) and other variants of DCG, or the mean reciprocal rank (MRR), etc.

The learning to rank setting is different than ours, however. One can view the setting in this paper as a more general *contextual* version of the learning to rank problem, with the added difficulty that we do not get to observe *irrelevant* classes for our inputs. Our goal here is to study retrieval methods that rank classes for each input context so as to maximize the fraction of relevant classes that land in the top-k ranked classes. The performance metric we consider is closely related to the Recall@k in the learning to rank setting: specifically, if the distribution on relevant classes conditioned on any given input is uniform, then the metric is exactly the expected Recall@k over randomly chosen inputs. However we emphasize that we do not restrict the class conditional distribution to be uniform.

A closely related work to ours in the learning to rank literature is that of Kar et al. (2015) who studied the Precision@k metric and provided surrogate loss functions for it and showed calibration under various conditions, gave methods for efficiently optimizing them, and provided generalization bounds. Another closely related work is that of Usunier et al. (2009) who also developed loss functions that are essentially the same as the Pairwise Ordered Weighted Losses in this paper. Turning to the optimization, the seminal work of Joachims (2005) gave an SVM method to optimize ranking metrics including Recall@k. However this method does not scale to large datasets since the loss function used is not decomposable. Eban et al. (2017) gave convex relaxations for information retrieval metrics with decomposable objectives, leading to training that scales to large datasets, although the relaxations do not come with theoretical guarantees other than being valid surrogates for the ranking metric in question.

None of the above works specifically tackle the problem of learning with very large output spaces that we consider in this paper. To the best of our knowledge, there has been no prior work on calibrated surrogate losses for the 0/1 retrieval loss we consider in this paper.

2 PRELIMINARIES AND NOTATION

The input space for examples is denoted $\mathcal{X}$ and the output space $\mathcal{Y}$. Define $K := |\mathcal{Y}|$, and we identify the output classes in $\mathcal{Y}$ with the integers $1, 2, \ldots, K$. The standing assumption in this paper is that K is "large": in the tens of thousands or larger. Unless specified otherwise, all vectors live in $\mathbb{R}^K$, and $\|v\|_p$ denotes the p-norm of v. For any vector $v \in \mathbb{R}^K$, we use $v_{[i]}$ to denote the i^{th} largest element of v and $\mathrm{Top}_k(v)$ to denote indices of the largest k coordinates in the vector v (breaking ties arbitrarily). For any $y \in \mathcal{Y}$, we use v^{-y} to denote the vector in $\mathbb{R}^{K-1}$ obtained by dropping coordinate v_y from v. We use $\mathbb{I}$ to denote the indicator function: $\mathbb{I}(x) = 1$ if x is true, and 0 otherwise. For any $m \in \mathbb{N}$, $[m] := \{1, 2, \ldots, m\}$.

We consider the following retrieval problem. There is an unknown distribution D over $\mathcal{X} \times \mathcal{Y}$. A sample (x, y) drawn from this distribution indicates that y is a relevant class for x. For a parameter $k \in \mathbb{N}$ with $k \ll K$, the goal is to design a method that, given an input $x \in \mathcal{X}$, outputs the k "most relevant" classes $y \in \mathcal{Y}$. Formally, a retrieval method, also called a predictor, is a function that for any input outputs a set of labels of size k, i.e. a function $f : \mathcal{X} \to \mathcal{Y}_k$ where $\mathcal{Y}_k = \{\mathcal{U} \subseteq \mathcal{Y} \mid |\mathcal{U}| = k\}$. Let $L : \mathcal{Y}_k \times \mathcal{Y} \to \mathbb{R}$ be a loss function that measures quality of the predictor's output: $L(S, y)$ should be large if $y \notin S$. In the retrieval setting of our interest, a commonly used loss function is the following 0/1 *retrieval loss*:

$$L(S, y) = \mathbb{I}(y \notin S). \tag{1}$$

Our aim is to find a predictor f with low expected loss (also referred to as *risk*):

$$R(f) := \mathbb{E}_{(X,Y)\sim D}[L(f(X), Y)].$$

Note that for $k = 1$, this effectively reduces to the classical multiclass classification problem.

One simple observation is the following characterization of the Bayes optimal predictor:

Lemma 1. *The predictor $b(x) := Top_k[D(Y = 1|X = x), \cdots, D(Y = K|X = x))]$ has minimal risk over all predictors.*

Proof. For any predictor $f : \mathcal{X} \to \mathcal{Y}_k$, we have $R(f) = \mathbb{E}_X\left[\sum_{y \in \mathcal{Y} \setminus f(X)} D(Y = y \mid X)\right]$. This is minimized by $f = b$. □

Without loss of generality we assume that $D_{[k]}(\cdot \mid X = x) > D_{[k+1]}(\cdot \mid X = x)$ for all $x \in \mathcal{X}$, so that b is uniquely defined.[1]

Inspired by the above characterization of the Bayes optimal predictor, as well as standard practice, we aim to learn a predictor by finding a scoring function $h : \mathcal{X} \to \mathbb{R}^K$ and predict the set of labels of x as $f(x) = \mathrm{Top}_k(h(x))$.

Definition 1. *We say a score vector $v \in \mathbb{R}^K$ is Bayes compatible for $x \in \mathcal{X}$ if $Top_k(v) = b(x)$.*

We denote the set of all Bayes compatible vectors for any $x \in \mathcal{X}$ by $\mathcal{B}_x$. Ideally, we would like to learn a hypothesis which outputs a Bayes compatible vector for each $x \in \mathcal{X}$. Since minimizing the 0/1 loss is typically intractable, in practice we instead use a more tractable *surrogate loss* $\ell : \mathbb{R}^K \times \mathcal{Y} \to \mathbb{R}$. Define ℓ-risk as $R_\ell(h) := \mathbb{E}_{(X,Y)\sim D}[\ell(h(X), Y)]$. Given a training

[1]The results in this paper can be easily applied to the case when $D_{[k]}(\cdot \mid X = x) = D_{[k+1]}(\cdot \mid X = x)$.

set of n samples $\{(x_i, y_i)\}_{i=1}^n$ drawn i.i.d. from D, the empirical ℓ-risk is $\widehat{R}_\ell(h) := \frac{1}{n}\sum_{i=1}^n \ell(h(x_i), y_i)$. A predictor is then computed by minimizing the empirical ℓ-risk over an appropriate hypothesis class $\mathcal{H}$.

A basic and desirable property of the loss function is *calibration*, which is defined below.

Definition 2. *We say loss function ℓ is calibrated with respect to the retrieval loss* (1) *if the following condition holds for all $x \in \mathcal{X}$:*

$$\inf_v \mathbb{E}_Y[\ell(v, Y) \mid X = x] < \inf_{v \notin \mathcal{B}_x} \mathbb{E}_Y[\ell(v, Y) \mid X = x]. \tag{2}$$

The above definition is a natural generalization of the one in Zhang (2004). This definition essentially states that the loss function is calibrated if, given any particular input x, score vectors that minimize the loss function are Bayes compatible for x. The following result (proof in Appendix B) shows that minimizing a calibrated surrogate loss leads to the Bayes optimal predictor:

Theorem 1. *Let $\mathcal{H}$ be the class of all measurable functions $h : \mathcal{X} \to \mathbb{R}^K$. If ℓ is calibrated, then for all $\epsilon > 0$, there exists $\delta > 0$ such that for any distribution D and any $h \in \mathcal{H}$, if $R_\ell(h) \le \inf_{h' \in \mathcal{H}} R_\ell(h') + \delta$, then $R(Top_k(h)) \le \inf_{h' \in \mathcal{H}} R(Top_k(h')) + \epsilon$.*

3 ORDERED WEIGHTED LOSSES AND STOCHASTIC NEGATIVE MINING

We now present some loss functions specifically geared towards scenarios where K is large. We start by defining a general class of loss functions (hereafter referred to as ordered weighted loss (OWL)). OWLs are parameterized by

1. a non-increasing function $\phi : \mathbb{R} \to \mathbb{R}$ that is a surrogate loss for the 0/1 loss, i.e. $\phi(x) \ge \mathbb{I}(x \le 0)$ for all $x \in \mathbb{R}$, and
2. a non-negative weight vector $\theta \in \mathbb{R}^{K-1}$.

Some examples of valid ϕ functions are the hinge loss $\phi_{\text{hinge}}(u) = \max\{1-u, 0\}$, logistic loss $\phi_{\text{logistic}}(u) = \log_2(1 + \exp(-u))$, squared-hinge loss $\phi_{\text{sq-hinge}}(u) = \max\{1-u, 0\}^2$, exponential loss $\phi_{\exp}(u) = \exp(-u)$, and ramp loss, parameterized by a margin $\rho > 0$: $\phi_{\text{ramp},\rho}(u) = \mathbb{I}(u \le 0) + (1 - u/\rho)\mathbb{I}(0 < u \le \rho)$.

We define two types of OWLs:

Definition 3. *The (ϕ, θ)-***Pairwise** *Ordered Weighted Loss (POWL) ℓ is defined as $\ell(v, y) := \sum_{j=1}^{K-1} \theta_j \phi(v_y - v_{[j]}^{-y})$.*

Definition 4. *The (ϕ, θ)-***Binary** *Ordered Weighted Loss (BOWL) ℓ is defined as $\ell(v, y) := \phi(v_y) + \sum_{j=1}^{K-1} \theta_j \phi(-v_{[j]}^{-y})$.*

POWLs have also been studied by Usunier et al. (2009). Several commonly used loss functions for multiclass classification are OWLs: for example, the multiclass SVM loss of Crammer and Singer (2001) is a POWL with $\theta_1 = 1$ and $\theta_j = 0$ for all $j > 1$, the loss function of Weston and Watkins (1998) is a POWL with all $\theta_j = 1$, and the loss function of Lee et al. (2004) is a BOWL with all $\theta_j = 1$.

With the notable exception of the ramp loss $\phi_{\text{ramp},\rho}$, all other examples for ϕ are convex. For convex ϕ, under some mild additional conditions on θ, we can show that the two types of OWLs defined above are convex in the score vector v. If v is generated via a linear or kernel model, the loss becomes convex in the model paramters and can be optimized using convex optimization techniques.

Theorem 2 (Convexity)**.** *Suppose ϕ is convex. Furthermore, suppose that θ has non-increasing coordinates, i.e. $\theta_j \ge \theta_{j'}$ for $j < j'$. Then both the (ϕ, θ)-POWL and the (ϕ, θ)-BOWL are convex in the score vector v for any fixed $y \in \mathcal{Y}$.*

The proof appears in Appendix D. Furthermore, under a different condition on θ, the OWLs are valid surrogate losses for the retrieval loss (1) (proof in Appendix D):

Theorem 3 (Surrogate loss)**.** *Suppose $\theta_j = 1/k$ for all $j \in [k]$. Then the (ϕ, θ)-POWL ℓ is a surrogate loss for the retrieval loss* (1)*: $\ell(v, y) \ge \mathbb{I}(y \notin Top_k(v))$. Similarly, the (ϕ, θ)-BOWL ℓ is a surrogate loss for the retrieval loss scaled by 2: $\ell(v, y) \ge 2\mathbb{I}(y \notin Top_k(v))$.*

Finally, under certain conditions on ϕ, θ, and the data distribution, OWLs are calibrated:

Theorem 4 (Calibration)**.** *Suppose ϕ is a differentiable function with $\phi'(\epsilon) < 0$ for $\epsilon \le 0$, and θ is such that $\theta_i = \frac{1}{k}$ for $j \in [k]$ and $\theta_j \le \frac{1}{k}$ for $j > k$. Also, suppose the following condition holds for all $x \in \mathcal{X}$:*

$$D_{[k]}(\cdot \mid X = x) > \frac{\sum_{l=k+1}^m D_{[l]}(\cdot \mid X = x))}{k \sum_{j=k+1}^m \theta_j}, \tag{3}$$

for all $m > k$. Then we have the following:

1. *The (ϕ, θ)-POWL is calibrated.*
2. *Suppose additionally that $\theta_j > 0$ for all j. Then the (ϕ, θ)-BOWL is calibrated.*

The proofs of these results appear in Appendix E.

The condition (3) required to show calibration is essentially an assumption on the tail of the class conditional

```
Algorithm 1 Stochastic Negative Mining
Input: A score vector $v \in \mathbb{R}^K$ with random coordinate access, a class $y \in \mathcal{Y}$, a parameter $B \in \mathbb{N}$ with $B \geq k$, a non-negative vector $\vartheta \in \mathbb{R}^B$ with non-increasing coordinates, and desired loss type (POWL or BOWL)
Output: A random variable $L(v, y) \in \mathbb{R}$ estimating the loss.
1: Sample a subset of $\mathcal{B} \subseteq \mathcal{Y} \setminus \{y\}$ of size $B$ uniformly at random.
2: Let $v^{\mathcal{B}}_{[1]} \geq v^{\mathcal{B}}_{[2]} \cdots \geq v^{\mathcal{B}}_{[B]}$ be elements of $\{v_{y'} \mid y' \in \mathcal{B}\}$ in non-increasing order.
3: Return
   $L(v,y) = \begin{cases} \sum_{j=1}^{B} \vartheta_j \phi(v_y - v^{\mathcal{B}}_{[j]}) & \text{(if POWL)} \\ \phi(v_y) + \sum_{j=1}^{B} \vartheta_j \phi(-v^{\mathcal{B}}_{[j]}) & \text{(if BOWL)} \end{cases}$
```

distribution for x. While it is not possible to verify the condition since we don't have access to the data distribution, it is still possible in practical applications to choose θ such that this condition holds. For example, suppose from domain knowledge we know that for any x, there can be at most $M \ll K$ relevant labels (i.e. $D_{[M+1]}(\cdot|X = x) = 0$). In image labeling tasks, for example, we may have reason to expect that any image can have no more than 10 different labels. In that case, we can set $\theta_j = 1/k$ for $j \in [M]$ and $\theta_j = 0$ for $j > M$ suffices to satisfy this condition. One can make a similar prescription for θ for milder domain knowledge requirements; for example in cases where we know that for any x, $\sum_{j=M+1}^{K-1} D_{[j]}(\cdot|X = x) < \epsilon$ for some small ϵ like 0.1. Finally, the default setting of $\theta_j = 1/k$ for all $j \in [K-1]$ always satisfies (3).

3.1 Stochastic Negative Mining

The above results show that the proposed family of loss functions has useful statistical properties. However the choice of θ and the computational efficiency of minimizing such a loss function have not been discussed, specifically for the large K setting. This is central to the paper since, in our problem setting, we desire a loss function that can be computed (or at least, randomly estimated) without needing to compute the scores of all K labels. For instance, one could simply choose $\theta_j = 0$ for all $j > k$, typically referred to as *top-k* loss, but it is computationally intractable for very large k.

To this end, we now present an approach to efficiently optimize a particular instance of OWL; thereby enjoying the useful statistical properties described above in addition to being amenable to efficient computation. For the ease of exposition, instead of directly specifying the value of θ, we resort to a constructive approach to describe the loss function. The construction samples a set $\mathcal{B}$ of B labels in $\mathcal{Y} \setminus \{y\}$ uniformly at random, then sorts the scores of the sampled labels, and computes the loss in an OWL-like manner using a weight vector $\vartheta \in \mathbb{R}^B$ (see Algorithm 1). The actual loss function is defined as $\ell_{\text{snm}}(v, y) := \mathbb{E}_{\mathcal{B}}[L(v, y)]$. We can obtain an unbiased estimate of the loss (and its gradients) without having to compute this expectation explicitly by simply computing the loss on a randomly selected subset $\mathcal{B}$ (as described in Algorithm 1); thus, allowing efficient optimization using algorithms like stochastic gradient method. We term this procedure *Stochastic Negative Mining* (SNM).

It should be evident from the description of the procedure that the randomized estimator can be computed by computing the scores of only at most B randomly chosen classes and the score of the class y in question. Overall, including the time for sampling the classes, the procedure can be implemented in $O(B \log(K))$ time, which can be a significantly faster than computing scores of all labels if $B \ll K$. The tradeoff is increased variance in the estimator leading to worse generalization bounds (as we shall see shortly) compared to computing all scores, but in practice the benefits can significantly outweigh the costs, as our experiments demonstrate.

To gain more intuition, observe that in the two extreme cases of $\mathcal{B} = [K]\backslash\{y\}$ (with $\vartheta_j = \frac{1}{k}$ for $j \leq k$ and $\vartheta_j = 0$ otherwise) and $|\mathcal{B}| = B$ (with $\vartheta_j = \frac{K-1}{kB}$ for $j \leq B$), the procedure amounts to optimizing *top-k* loss and using negative sampling respectively. Thus, SNM seamlessly generalizes to various approaches used in the machine learning literature. The size of $\mathcal{B}$ is typically constrained by computation and memory budgets. Using $\mathcal{B} = [K]$ is typically intractable for large K. When $\mathcal{B} = B$ (where $k < B \ll K$), we claim that it is beneficial to use SNM instead of the negative sampling procedure used while dealing with large-output spaces. This claim is backed through generalization bounds and empirical results (Sections 4 & 5).

We now show that ℓ_{snm} is an OWL, and thereby inherits all the useful statistical properties of OWLs. The proof is deferred to Appendix C.

Lemma 2. *For any $\tau > 0$, ℓ_{snm} is either a POWL or BOWL. Furthermore, the coordinates of the corresponding weight vector θ are in decreasing order and non-zero and ℓ_{snm} is convex. If $\vartheta_j = \frac{K-1}{kB}$ for $j \in [k]$, then $\theta_j = 1/k$ for all $j \in [k]$ and ℓ_{snm} is a valid surrogate loss for the retrieval loss. If $\vartheta_j > 0$ for all $j \in [B]$ and the conditions of Theorem 4 hold, then ℓ_{snm} is calibrated. Finally, $\|\theta\|_1 = \|\vartheta\|_1$, and $\|\theta\|_2 \leq \sqrt{\frac{B\vartheta_1\|\vartheta\|_1}{K-1}}$.*

The bounds on the norms of θ mentioned above are important for the generalization bounds given in the next section. Smaller norms have smaller generalization error, and thus a good choice of ϑ is $\vartheta_j = \frac{K-1}{kB}$ for $j \in [k]$ and $\vartheta_j = 0$ for $j > k$[2]. For this setting of ϑ SNM reduces to the following intuitively appealing algorithm: sample a batch of B classes, and choose the top-k scoring classes as "negatives" and set the loss to be their average loss. For this reason, we call this *Top-k SNM*. Empirically this technique works quite well, as can be seen from our experiments.

The ϑ parameter allows for considerable flexibility in designing Stochastic Negative Mining. Other settings of ϑ, than the one mentioned above, can be used based on the application. In our experiments, for example, we found that Top-k' SNM for $k' < k$ works even better than Top-k SNM. Another example is if in an application we wish to penalize harder negative even more than in Top-k SNM, then we can choose the coordinates of ϑ according to a power law distribution with some exponent α. Another idea is to treat ϑ as scaled sampling probabilities, sub-sample negatives within $\mathcal{B}$ according to these probabilites, and add up (appropriately scaled) losses for the sub-sampled negatives. This can lead to further computational gains since losses need to be evaluated for even fewer classes.

4 GENERALIZATION ERROR BOUNDS

We now turn to generalization bounds for OWLs. To describe the bounds, we need to define some notation first. Let $\mathcal{H}$ be a hypothesis class of functions $h : \mathcal{X} \to \mathbb{R}^K$. For a set S of examples $(x, y) \in \mathcal{X} \times \mathcal{Y}$, let $g = \{g_{(x,y)} \mid (x, y) \in S\}$ be a set of i.i.d. Gaussian random variables indexed by examples in S, and let $\mathbb{E}_g[\cdot]$ denote expectation over these random variables. Then the empirical Gaussian complexity w.r.t. S is defined to be $\mathfrak{G}_S(\mathcal{H}) = \frac{1}{|S|}\mathbb{E}_g[\sup_{h\in\mathcal{H}} \sum_{(x,y)\in S} g_{(x,y)} h_y(x)]$. Empirical Rademacher complexity $\mathfrak{R}_S(\mathcal{H})$ is defined similarly with the Gaussian random variables replaced by Rademacher ones. We also define the *label completion* of S, denoted $\tilde{S} = \{(x, y) \mid \exists y' \in \mathcal{Y} \text{ s.t. } (x, y') \in S,\ y \in \mathcal{Y}\}$. The *worst-case* empirical Rademacher complexity over $\bar{S}$ is defined as $\overline{\mathfrak{R}}_{\bar{S}}(\mathcal{H}) := \sup\{\mathfrak{R}_T(\mathcal{H}) \mid T \text{ is multi-subset of } \bar{S}, |T| = |\bar{S}|\}$. Finally, in this section we use the $\tilde{O}(\cdot)$ notation to suppress polylogarithmic factors in the problem parameters. The main generalization bound is the following, proved in Appendix F:

Theorem 5. *Let $\phi(\cdot)$ be L-Lipschitz. Assume that for some $\Phi > 0$, $|\phi(h_y(x))|, |\phi(h_y(x) - h_{y'}(x))| \le \Phi$ for any $y, y' \in \mathcal{Y}$. Let S be a sample set of n i.i.d. examples drawn from the input distribution. Suppose ℓ is the (ϕ, θ)-POWL. Then with probability at least $1 - \delta$ over the choice of S, for any $h \in \mathcal{H}$, the generalization error $\mathbb{E}_{(x,y)}[\ell(h(x), y)] - \mathbb{E}_{(x,y)\sim S}[\ell(h(x), y)]$ is bounded by*

$$\tilde{O}\left(L \min\left\{\|\theta\|_2 K \mathfrak{G}_{\bar{S}}(\mathcal{H}) + \|\theta\|_1 \mathfrak{G}_S(\mathcal{H}), 2\|\theta\|_1 \sqrt{K}\overline{\mathfrak{R}}_{\bar{S}}(\mathcal{H})\right\}\right) + 3\|\theta\|_1 \Phi \sqrt{\frac{\log(2/\delta)}{2n}}.$$

If ℓ is the (ϕ, θ)-BOWL, then the generalization error is bounded by

$$\tilde{O}\left(L \min\left\{\|\theta\|_2 K \mathfrak{G}_{\bar{S}}(\mathcal{H}) + \mathfrak{G}_S(\mathcal{H}), (\|\theta\|_1 + 1)\sqrt{K}\overline{\mathfrak{R}}_{\bar{S}}(\mathcal{H})\right\}\right) + 3\|\theta\|_1 \Phi \sqrt{\frac{\log(2/\delta)}{2n}}.$$

We can now analyze the effect of the parameter B in Stochastic Negative Mining for the particular choice of ϑ given after Lemma 2, i.e. $\vartheta_j = \frac{K-1}{kB}$ for $j \in [k]$ and $\vartheta_j = 0$ for $j > B$. The corresponding ℓ_{snm} has $\|\theta\|_1 = {}^{K-1}/_{B}$ and $\|\theta\|_2 \le \sqrt{{}^{K-1}/_{kB}}$. The generalization error therefore decreases with B, as expected; albeit, at the cost of additional computation.

It is easy to check that the corresponding values of $\|\theta\|_1$ for SNM and negative sampling are $\frac{K-1}{B}$ and $\frac{K-1}{k}$ respectively. Our generalization bounds indicate that for $|\mathcal{B}| = B > k$, one can obtain better generalization through SNM in comparison to negative sampling. This is due to the fact that the generalization bounds depend on $\|\theta\|_1$, deteriorating as $\|\theta\|_1$ increases. Before ending our discussion, we need to make it explicit that our analysis only compares the upper bounds and hence, needs to be interpreted with caution. Nonetheless, our empirical evaluation, in the next section, supports our theoretical analysis and provides compelling case to use SNM approach in practice.

4.1 Margin bounds for retrieval loss

We now provide margin based generalization error bounds for predicting labels by taking $\text{Top}_k(h(x))$ for $h \in \mathcal{H}$. For a hypothesis $h \in \mathcal{H}$ and an example (x, y), we define a notion of margin as $\rho_h(x, y) = h_y(x) - h^{-y}_{[k]}(x)$. In the multiclass setting, i.e. $k = 1$, this reduces to the standard definition of margin (Koltchinskii and Panchenko, 2002). Note that for any example (x, y), $y \notin \text{Top}_k(h(x)) \Leftrightarrow \rho_h(x, y) \le 0$ [3]. Let S be a set of n labeled examples drawn i.i.d. from the input

[2]Setting $\vartheta_j = 0$ may come at the price of calibration, but we can rectify that by setting $\vartheta_j = \epsilon$ for some small ϵ for all $j > k$.

[3]There's a subtlety here in the handling of ties at the k-th largest score. If there's a tie, then none of the tied classes are considered valid. This is consistent with previous definitions of the margin.

Dateset	#Features	#Labels	#TrainPoints	#TestPoints	Avg. #P/L	Avg. #L/P
AmazonCat	203,882	13,330	1,186,239	306,782	448.57	5.04
WikiLSHTC	1,617,899	325,056	1,778,351	587,084	17.46	3.19
Amazon670K	135,909	670,091	490,449	153,025	3.99	5.45
Amazon3M	337,067	2,812,281	1,717,899	742,507	31.64	36.17

Table 1: Summary of the datasets used in the paper. #P/L is the number of points per label, and #L/P is the number of labels per point.

		Top 1	Top 16	Top 64	Top 256		Top 1	Top 16	Top 64	Top 256
AmazonCat	R@1	2.59	2.02	1.65	1.32	P@1	2.33	1.99	1.66	1.30
	R@3	1.98	1.97	1.63	1.32	P@3	2.40	1.97	1.65	1.30
	R@5	2.58	1.96	1.60	1.29	P@5	2.39	1.92	1.61	1.30
WikiLSHTC	R@1	2.53	2.35	2.13	1.87	P@1	2.56	2.36	2.17	1.89
	R@3	2.71	2.46	2.18	1.86	P@3	2.70	2.46	2.18	1.90
	R@5	2.64	2.37	2.14	1.83	P@5	2.59	2.37	2.17	1.87
Amazon670K	R@1	1.23	1.17	1.13	1.11	P@1	1.25	1.21	1.16	1.13
	R@3	1.28	1.23	1.17	1.15	P@3	1.27	1.22	1.18	1.14
	R@5	1.32	1.24	1.18	1.15	P@5	1.33	1.34	1.23	1.18
Amazon3M	R@1	2.60	2.56	2.30	1.93	P@1	2.73	2.57	2.34	1.96
	R@3	2.92	2.72	2.42	2.05	P@3	2.95	2.80	2.47	2.13
	R@5	3.01	2.80	2.51	2.13	P@5	3.07	2.89	2.57	2.11

Table 2: Comparison of stochastic negative mining using top-k SNM for various values of k with negative sampling. For the ease of comparison, each entry in the table represents the Recall@k (resp. Precision@k) value of the method normalized with the Recall@k (resp. Precision@k) obtained for negative sampling. Note that values larger than 1 indicate better performance in comparison to negative sampling.

distribution. We define the margin ρ empirical risk of a hypothesis h as $\hat{R}_{S,\rho}(h) := \mathbb{E}_{(x,y)\sim S}\left[\mathbb{I}(\rho_h(x,y) \leq \rho)\right]$. With these definitions the following margin bound (proved in Appendix F):

Theorem 6. *Fix any $\rho > 0$. Then with probability at least $1-\delta$, for any $h \in \mathcal{H}$, we have*

$$R(h) \leq \hat{R}_{S,\rho}(h) + 3\sqrt{\frac{\log(2/\delta)}{2n}} + \tilde{O}\left(\tfrac{1}{\rho}\min\left\{K\mathfrak{G}_{\tilde{S}}(\mathcal{H}) + \mathfrak{G}_S(\mathcal{H}), \sqrt{K}\bar{\mathfrak{R}}_{\tilde{S}}(\mathcal{H})\right\}\right).$$

5 EXPERIMENTS

We now present empirical results for the SNM approach. We use publicly available "extreme multilabel classification" datasets for all our experiments [4] (see Table 1 for details about the datasets). As these datasets are inherently multilabel, we uniformly sample positive labels to generate training data that fits our retrieval framework. The classification performance on these datasets has been highly optimized through extensive research in the past decade. We would like to emphasize that our aim is to not obtain state-of-the-art results on these datasets but to rather verify two aspects: (i) SNM performs better than negative sampling, and (ii) SNM is practical for large-scale deep learning. For all our experiments, we use top-k variant of SNM.

[4]The datasets are available at `http://manikvarma.org/downloads/XC/XMLRepository.html`

Model architecture. As mentioned earlier, a simple model is used in our experiments to support our theoretical result. Our model is based on a simple embedding based neural network. For each data point (x, y) in the data set, $x \in \mathbb{R}^d$, which is typically sparse, is first embedded into 512-dimensional vector space using a two layer neural network with layer sizes 512 and 512 i.e., the embedding is obtained by first multiplying with a $d \times 512$ weight matrix followed by ReLU activation function, and then multiplying by a 512×512 linear transformation. The embedding is finally normalized so that its l_2-norm is 1. This yields a 512-dimensional embedding representation of the input. We found including the linear layer helped accelerate training when using SGD. Each class is represented as a 512-dimensional normalized vector. The number of parameters in this setup is $512 \cdot (d + 512 + K)$. The score of a data point is obtained by computing the inner product between the feature and class embeddings. Since all the embeddings are normalized, scores lie in $[-1, 1]$ interval.

Training setup. Experiments are conducted under the "BOWL" setting with hinge loss. We observed similar behavior for the "POWL" setting. SGD with a large learning rate is used in optimizing the embedding layers, and SGD with momentum is used in optimizing the linear transformation. For the small AmazonCat, the size of the sampled size is set as $B = 1,024$, and for all other datasets, we use $B = 32,768$. These values of B are selected based on computational and memory

		Embedding-based			**Other Methods**			
		Ours	SLEEC	LEML	PfastreXML	DiSMEC	PD-Sparse	PPD-Sparse
AmazonCat	P@1	81.58	90.53	-	91.75	93.4	90.60	-
	P@3	71.54	76.33	-	77.97	79.1	75.14	-
	P@5	58.79	61.52	-	63.68	64.1	60.69	-
WikiLSHTC	P@1	60.65	54.83	19.82	56.05	64.4	61.26	64.08
	P@3	42.08	33.42	11.43	36.79	42.5	39.48	41.26
	P@5	31.87	23.85	8.39	27.09	31.5	28.79	30.12
Amazon670K	P@1	44.68	35.05	8.13	39.46	44.7	-	45.32
	P@3	40.55	31.25	6.83	35.81	39.7	-	40.37
	P@5	37.40	28.56	6.03	33.05	36.1	-	36.92

Table 3: Performance comparison with other methods on AmazonCat, WikiLSHTCand Amazon670K. Although our goal is to optimize Recall@k, we report Precision@k here for comparison since it has been more widely reported in related works. We note that our embedding-based model trained using SNM performs significantly better than prior embedding-based methods such as SLEEC and LEML (Bhatia et al., 2015; Yu et al., 2014) on large datasets. Furthermore, despite its simplcity, our method is competitive with other computationally expensive methods specifically developed for these datasets.

constraints. Increasing the value of B in AmazonCat, did not result in any significant gain in performance.

We compare different settings of the top k negatives in stochastic negative mining (Table 2). Each of the dataset comes with a pre-defined train/test split and the results we report here are based on the test data. Although the goal of the paper is to optimize Recall@k, we also report the Precision@k metric in Table 2 since it has been more widely reported in related works. Note that the values in Table 2 are normalized with the value of negative sampling to enable easy comparison. Thus, any value greater than 1 indicates better performance in comparison to negative sampling. The results demonstrate that top-k SNM with various values of k substantially improves over negative sampling, and moreover, using more aggressive mining, i.e., smaller k, improves the results. For all the datasets, the best result is achieved with $k = 1$. Also note that SNM does not incur any additional computational cost in comparison to negative sampling approach; in fact, it is slightly more efficient due to the fact that fewer backpropagations are needed compared to negative sampling.

In Table 3, we also compare our results with a few other recent works including SLEEC (Bhatia et al., 2015), LEML (Yu et al., 2014), PfastreXML (Jain et al., 2016), DiSMEC (Babbar and Schölkopf, 2017) and PPD-Sparse (Yen et al., 2017) on Amazon670K, AmazonCat, and WikiLSHTC. As noted earlier, the goal of our experiments is not to achieve state-of-the-art results but to opt for a simple neural network model and verify that our proposed SNM method outperforms negative sampling. However, despite its simplicity, our method is better than other embedding based methods like SLEEC, LEML, and competitive with many recently published works, including large sparse linear models where no low-rank assumptions are made. We believe that the proposed technique can be combined with more sophisticated neural network models to further improve the performance.

6 DISCUSSION

In this paper, we considered the problem of retrieving the most relevant classes for any given input in the specific setting of large output spaces. We provided a family of loss functions that satisfy various desirable properties for this setting, and gave a scalable technique, Stochastic Negative Mining, that can optimize instances of losses in this family. We analyzed the generalization performance of models trained using the losses in this family. Our theoretical results indicate that the Top-k variant of Stochastic Negative Mining should be particularly favorable to this setting, and indeed comprehensive experiments on large public datasets indicate that this form of Stochastic Negative Mining yields substantial benefits over commonly used negative sampling techniques.

The most intriguing direction for future work is combining SNM with a custom optimization method designed to exploit the specific structure of the loss function. In particular, a principled approach to change the number of sampled classes as the optimization proceeds is an important future work. Also, here we mainly focused on a particular variant of SNM called top-k SNM. It is an interesting direction of future work to investigate other settings of ϑ parameters for SNM within the sampled classes. Finally, SNM approaches for coupled loss functions such as softmax cross-entropy remains open and is left as future work.

References

S. Agarwal. The infinite push: A new support vector ranking algorithm that directly optimizes accuracy at the absolute top of the list. In *SDM*, pages 839–850, 2011.

R. Babbar and B. Schölkopf. DiSMEC: Distributed sparse machines for extreme multi-label classification. In *WSDM*, pages 721–729, 2017.

Y. Bai, S. Goldman, and L. Zhang. TAPAS: two-pass approximate adaptive sampling for softmax. *CoRR*, abs/1707.03073, 2017.

P. L. Bartlett and M. H. Wegkamp. Classification with a reject option using a hinge loss. *Journal of Machine Learning Research*, 9:1823–1840, 2008.

K. Bhatia, H. Jain, P. Kar, M. Varma, and P. Jain. Sparse local embeddings for extreme multi-label classification. In *NIPS*, pages 730–738, 2015.

S. P. Boyd, C. Cortes, M. Mohri, and A. Radovanovic. Accuracy at the top. In *NIPS*, pages 962–970, 2012.

C. Chow. On optimum error and reject trade-off. 16: 41–46, 1970.

K. Crammer and Y. Singer. On the algorithmic implementation of multiclass kernel-based vector machines. *Journal of Machine Learning Research*, 2:265–292, 2001.

H. Daumé III, N. Karampatziakis, J. Langford, and P. Mineiro. Logarithmic time one-against-some. In *ICML*, pages 923–932, 2017.

J. del Coz, J. Díez, and A. Bahamonde. Learning nondeterministic classifiers. 10:2273–2293, 10 2009.

C. Denis and M. Hebiri. Confidence sets with expected sizes for multiclass classification. *Journal of Machine Learning Research*, 18:102:1–102:28, 2017.

E. Eban, M. Schain, A. Mackey, A. Gordon, R. Rifkin, and G. Elidan. Scalable learning of non-decomposable objectives. In *AISTATS*, pages 832–840, 2017.

F. Fagan and G. Iyengar. Unbiased scalable softmax optimization. *CoRR*, abs/1803.08577, 2018. URL `http://arxiv.org/abs/1803.08577`.

E. Grycko. Classification with set-valued decision functions. *Information and Classification*, pages 218–224, 1993.

R. Herbei and M. H. Wegkamp. Classification with reject option. *Canadian Journal of Statistics*, 34(4): 709–721, 2006.

H. Jain, Y. Prabhu, and M. Varma. Extreme multi-label loss functions for recommendation, tagging, ranking & other missing label applications. In *KDD*, pages 935–944, 2016.

S. Jean, K. Cho, R. Memisevic, and Y. Bengio. On using very large target vocabulary for neural machine translation. In *ACL*, pages 1–10, 2015.

T. Joachims. A support vector method for multivariate performance measures. In *ICML*, pages 377–384, 2005.

P. Kar, H. Narasimhan, and P. Jain. Surrogate functions for maximizing precision at the top. In *ICML*, pages 189–198, 2015.

V. Koltchinskii and D. Panchenko. Empirical margin distributions and bounding the generalization error of combined classifiers. *Annals of Statistics*, pages 1–50, 2002.

M. Lapin, M. Hein, and B. Schiele. Top-k multiclass SVM. In *NIPS*, pages 325–333, 2015.

Y. Lee, Y. Lin, and G. Wahba. Multicategory support vector machines: Theory and application to the classification of microarray data and satellite radiance data. 99(465):67–81, 2004.

J. Lei, J. Robins, and L. Wasserman. Distribution-free prediction sets. 108(501):278–287, 2013.

Y. Lei, Ü. Dogan, D.-X. Zhou, and M. Kloft. Data-dependent generalization bounds for multi-class classification. *CoRR*, abs/1706.09814, 2015. URL `http://arxiv.org/abs/1706.09814`.

T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *NIPS*, pages 3111–3119, 2013.

P. Raman, S. Matsushima, X. Zhang, H. Yun, and S. V. N. Vishwanathan. DS-MLR: exploiting double separability for scaling up distributed multinomial logistic regression. *CoRR*, abs/1604.04706, 2016. URL `http://arxiv.org/abs/1604.04706`.

M. Sadinle, J. Lei, and L. Wasserman. Least ambiguous set-valued classifiers with bounded error levels. *Journal of the American Statistical Association*, pages 1–12, 2018.

F. Schroff, D. Kalenichenko, and J. Philbin. Facenet: A unified embedding for face recognition and clustering. *CoRR*, abs/1503.03832, 2015.

N. Usunier, D. Buffoni, and P. Gallinari. Ranking with ordered weighted pairwise classification. In *ICML*, pages 1057–1064, 2009.

V. Vovk, A. Gammerman, and G. Shafer. *Algorithmic learning in a random world*. Springer, New York, 2005.

J. Weston and C. Watkins. Multi-class support vector machines. Technical report, 1998.

T. Wu, C. Lin, and R. C. Weng. Probability estimates for multi-class classification by pairwise coupling.

Journal of Machine Learning Research, 5:975–1005, 2004.

I. E. Yen, X. Huang, W. Dai, P. Ravikumar, I. S. Dhillon, and E. P. Xing. Ppdsparse: A parallel primal-dual sparse method for extreme classification. In *KDD*, pages 545–553, 2017.

H.-F. Yu, P. Jain, P. Kar, and I. Dhillon. Large-scale multi-label learning with missing labels. In E. P. Xing and T. Jebara, editors, *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pages 593–601, 22–24 Jun 2014.

M. Yuan and M. H. Wegkamp. Classification methods with reject option based on convex risk minimization. *Journal of Machine Learning Research*, 11:111–130, 2010.

T. Zhang. Statistical analysis of some multi-category large margin classification methods. *Journal of Machine Learning Research*, 5:1225–1251, 2004.