Efficient Optimization for Rank-based Loss Functions

Pritish Mohapatra*
IIIT Hyderabad

Michal Rolínek*
MPI Tübingen

C.V. Jawahar
IIIT Hyderabad

Vladimir Kolmogorov
IST Austria

M. Pawan Kumar
University of Oxford, Alan Turing Institute

Abstract

The accuracy of information retrieval systems is often measured using complex loss functions such as the average precision (AP) or the normalized discounted cumulative gain (NDCG). Given a set of positive and negative samples, the parameters of a retrieval system can be estimated by minimizing these loss functions. However, the non-differentiability and non-decomposability of these loss functions does not allow for simple gradient based optimization algorithms. This issue is generally circumvented by either optimizing a structured hinge-loss upper bound to the loss function or by using asymptotic methods like the direct-loss minimization framework. Yet, the high computational complexity of loss-augmented inference, which is necessary for both the frameworks, prohibits its use in large training data sets. To alleviate this deficiency, we present a novel quicksort flavored algorithm for a large class of non-decomposable loss functions. We provide a complete characterization of the loss functions that are amenable to our algorithm, and show that it includes both AP and NDCG based loss functions. Furthermore, we prove that no comparison based algorithm can improve upon the computational complexity of our approach asymptotically. We demonstrate the effectiveness of our approach in the context of optimizing the structured hinge loss upper bound of AP and NDCG loss for learning models for a variety of vision tasks. We show that our approach provides significantly better results than simpler decomposable loss functions, while requiring a comparable training time.

1. Introduction

Information retrieval systems require us to rank a set of samples according to their relevance to a query. The risk of the predicted ranking is measured by a user-specified loss function. Several intuitive loss functions have been proposed in the literature. These include simple decomposable losses (that is, loss functions that decompose over each training sample) such as 0-1 loss [16, 20] and the area under the ROC curve [1, 11], as well as the more complex non-decomposable losses (that is, loss functions that depend on the entire training data set) such as the average precision (AP) [5, 27] and the normalized discounted cumulative gain (NDCG) [6].

When learning a retrieval system, one can use a training objective that is agnostic to the risk, such as in the case of LambdaMART [4]. In this work, we focus on approaches that explicitly take into account the loss function used to measure the risk. Such approaches can use any one of the many machine learning frameworks such as structured support vector machines (SSVM) [24, 25], deep neural networks [23], decision forests [14], or boosting [21]. To estimate the parameters of the framework, they employ a training objective that is closely related to the empirical risk computed over a large training data set. Specifically, it is common practice to employ either a structured hinge upper bound to the loss function [6, 27], or an asymptotic alternative such as direct loss minimization [10, 22].

The feasibility of both the structured hinge loss and the direct loss minimization approach depends on the computational efficiency of the *loss-augmented inference* procedure. When the loss function is decomposable, the loss-augmented inference problem can be solved efficiently by independently considering each training sample. However, for non-decomposable loss functions, it presents a hard computational challenge. For example, given a training data set with P positive (relevant to the query) and N negative (not relevant to the query) samples, the best known algorithms for loss-augmented inference for AP and NDCG loss functions have a complexity of $O(PN + N \log N)$ [6, 27]. Since the number of negative samples N can be very large in practice, this prohibits their use on large data sets.

In order to address the computational challenge of non-decomposable loss functions such as those based on AP and

*The first two authors contributed equally and can be reached at *pritish.mohapatra@research.iiit.ac.in*, *michal.rolinek@tuebingen.mpg.de* respectively.

NDCG, we make three contributions. First, we characterize a large class of ranking based loss functions that are amenable to a novel quicksort flavored optimization algorithm for the corresponding loss-augmented inference problem. We refer to the class of loss functions as QS-*suitable*. Second, we show that the AP and the NDCG loss functions are QS-suitable, which allows us to reduce the complexity of the corresponding loss-augmented inference to $O(N \log P)$. Third, we prove that there cannot exist a comparison based method for loss-augmented inference that can provide a better asymptotic complexity than our quicksort flavored approach. It is worth noting that our work is complementary to previous algorithms that have been proposed for the AP based loss functions [18]. Specifically, while the method of [18] cannot improve the asymptotic complexity of our loss-augmented inference algorithm, it can be used to reduce the runtime of a subroutine.

For the sake of clarity, we limit our discussion to the structured hinge loss upper bound to the loss function. However, as our main contribution is to speed-up loss-augmented inference, it is equally applicable to direct loss minimization. We demonstrate the efficacy of our approach on the challenging problems of action recognition, object detection and image classification, using publicly available data sets. Rather surprisingly, we show that in case of some models, parameter learning by optimizing complex non-decomposable AP and NDCG loss functions can be carried out faster than by optimizing simple decomposable 0-1 loss. Specifically, while each loss-augmented inference call is more expensive for AP and NDCG loss functions, it can take fewer calls in practice to estimate the parameters of the corresponding model.

2. Background

We begin by providing a brief description of a general retrieval framework that employs a rank-based loss function, hereby referred to as the ranking framework. Note that this framework is the same as or generalizes the ones employed in previous works [6, 12, 18, 22, 27]. The two specific instantiations of the ranking framework that are of interest to us employ the average precision (AP) loss and the normalized discounted cumulative gain (NDCG) loss respectively. A detailed description of the two aforementioned loss functions is provided in the subsequent subsection.

2.1. The Ranking Framework

Input. The input to this framework is a set of n samples, which we denote by $\mathbf{X} = \{\mathbf{x}_i, i = 1, \ldots, n\}$. For example, each sample can represent an image and a bounding box of a person present in the image. In addition, we are also provided with a query, which in our example could represent an action such as 'jumping'. Each sample can either belong to the positive class (that is, the sample is relevant to the query) or the negative class (that is, the sample is not relevant to the query). For example, if the query represents the action 'jumping' then a sample is positive if the corresponding person is performing the jumping action, and negative otherwise. The set of positive and the negative samples are denoted by $\mathcal{P}$ and $\mathcal{N}$ respectively. which we assume are provided during training, but are not known during testing.

Output. Given a query and a set of n samples $\mathbf{X}$, the desired output of the framework is a ranking of the samples according to their relevance to the query. This is often represented by a ranking matrix $\mathbf{R} \in \{-1, 0, 1\}^{n \times n}$ such that $\mathbf{R}_{\mathbf{x},\mathbf{y}} = 1$ if $\mathbf{x}$ is ranked higher than $\mathbf{y}$, -1 if $\mathbf{x}$ is ranked lower than $\mathbf{y}$ and 0 if $\mathbf{x}$ and $\mathbf{y}$ are ranked the same. In other words, the matrix $\mathbf{R}$ is an anti-symmetric that represents the relative ranking of a pair of samples.

Given the sets $\mathcal{P}$ and $\mathcal{N}$ during training, we construct a ground truth ranking matrix $\mathbf{R}^*$, which ranks each positive sample above all the negative samples. Formally, the ground truth ranking matrix $\mathbf{R}^*$ is defined such that $\mathbf{R}^*_{\mathbf{x},\mathbf{y}} = 1$ if $\mathbf{x} \in \mathcal{P}$ and $\mathbf{y} \in \mathcal{N}$, -1 if $\mathbf{x} \in \mathcal{N}$ and $\mathbf{y} \in \mathcal{P}$, and 0 if $\mathbf{x}, \mathbf{y} \in \mathcal{P}$ or $\mathbf{x}, \mathbf{y} \in \mathcal{N}$. Note that the ground truth ranking matrix only defines a partial ordering on the samples since $\mathbf{R}^*_{i,j} = 0$ for all pairs of positive and negative samples. We will refer to rankings where no two samples are ranked equally as *proper rankings*. Without loss of generality, we will treat all rankings other than the ground truth one as a proper ranking by breaking ties arbitrarily.

Discriminant Function. Given an input set of samples $\mathbf{X}$, the discriminant function $F(\mathbf{X}, \mathbf{R}; \mathbf{w})$ provides a score for any candidate ranking $\mathbf{R}$. Here, the term $\mathbf{w}$ refers to the parameters of the discriminant function. We assume that the discriminant function is piecewise differentiable with respect to its parameters $\mathbf{w}$. One popular example of the discriminant function used throughout the ranking literature is the following:

$$F(\mathbf{X}, \mathbf{R}; \mathbf{w}) = \frac{1}{|\mathcal{P}|\,|N|} \sum_{\mathbf{x} \in \mathcal{P}} \sum_{\mathbf{y} \in \mathcal{N}} \mathbf{R}_{\mathbf{x},\mathbf{y}} (\phi(\mathbf{x}; \mathbf{w}) - \phi(\mathbf{y}; \mathbf{w})). \tag{1}$$

Here, $\phi(\mathbf{x}; \mathbf{w})$ is the score of an individual sample, which can be provided by a structured SVM or a deep neural network with parameters $\mathbf{w}$.

Prediction. Given a discriminant function $F(\mathbf{X}, \mathbf{R}; \mathbf{w})$ with parameters $\mathbf{w}$, the ranking of an input set of samples $\mathbf{X}$ is predicted by maximizing the score, that is, by solving the following optimization problem:

$$\mathbf{R}(\mathbf{w}) = \operatorname*{argmax}_{\mathbf{R}} F(\mathbf{X}, \mathbf{R}; \mathbf{w}). \tag{2}$$

The special form of the discriminant function in equation (1) enables us to efficiently obtain the predicted ranking $\mathbf{R}(\mathbf{w})$ by sorting the samples in descending order of their

individual scores $\phi(\mathbf{x};\mathbf{w})$. We refer the reader to [12, 27] for details.

Parameter Estimation. We now turn towards estimating the parameters of our model given input samples $\mathbf{X}$, together with their classification into positive and negative sets $\mathcal{P}$ and $\mathcal{N}$ respectively. To this end, we minimize the risk of prediction computed using a user-specified loss function $\Delta(\mathbf{R}^*, \mathbf{R}(\mathbf{w}))$, where $\mathbf{R}^*$ is the ground truth ranking that is determined by $\mathcal{P}$ and $\mathcal{N}$ and $\mathbf{R}(\mathbf{w})$ is the predicted ranking as shown in equation (2). We estimate the parameters of our model as

$$\mathbf{w}^* = \min_{\mathbf{w}} \mathbb{E}[\Delta(\mathbf{R}^*, \mathbf{R}(\mathbf{w}))]. \tag{3}$$

In the above equation, the expectation is taken with respect to the data distribution.

Optimization for Parameter Estimation. For many intuitive rank based loss functions such as AP loss and NDCG loss, owing to their non-differentiability and non-decomposability, problem (3) can be difficult to solve using simple gradient based methods. One popular approach is to modify problem (3) to instead minimize a structured hinge loss upper bound to the user-specified loss. We refer the reader to [27] for further details about this approach.

Formally, the model parameters can now be obtained by solving the following problem:

$$\mathbf{w}^* = \min_{\mathbf{w}} \mathbb{E}[J(\mathbf{w})] \tag{4}$$
$$J(\mathbf{w}) = \max_{\mathbf{R}} \Delta(\mathbf{R}^*, \mathbf{R}) + F(\mathbf{X}, \mathbf{R}; \mathbf{w}) - F(\mathbf{X}, \mathbf{R}^*; \mathbf{w})$$

The function $J(\mathbf{w})$ in problem (4) is continuous and piecewise differentiable, and is amenable to gradient based optimization. The semi-gradient [1] of $J(\mathbf{w})$ takes the following form:

$$\nabla_{\mathbf{w}} J(\mathbf{w}) = \nabla_{\mathbf{w}} F(\mathbf{X}, \bar{\mathbf{R}}; \mathbf{w}) - \nabla_{\mathbf{w}} F(\mathbf{X}, \mathbf{R}^*; \mathbf{w}), \tag{5}$$
$$\text{with,} \quad \bar{\mathbf{R}} = \operatorname*{argmax}_{\mathbf{R}} \Delta(\mathbf{R}^*, \mathbf{R}) + F(\mathbf{X}, \mathbf{R}; \mathbf{w}). \tag{6}$$

Borrowing terminology from the structured prediction literature [13, 27], we call $\bar{\mathbf{R}}$ the *most violating ranking* and problem (6) as the *loss-augmented inference* problem. An efficient procedure for loss-augmented inference is key to solving problem (4).

While we focus on using loss-augmented inference for estimating the semi-gradient, it can also be used as the cutting plane [13] and the conditional gradient of the dual of problem (4). In addition to this, loss-augmented inference is also required for solving problem (3) using the direct loss minimization framework [22].

[1] For a continuous function $f(x)$ defined on a domain of any generic dimension, we can define semi-gradient $\nabla_s f(x)$ to be a random picking from the set $\{\nabla f(t) : ||x - t|| < \epsilon\}$, for a sufficiently small ϵ.

2.2. Loss Functions

While solving problem (6) is non-trivial, especially for non-decomposable loss functions, the method we propose in this paper allows for an efficient loss-augmented inference procedure for such complex loss functions. For our discussion, we focus on two specific non-decomposable loss functions. The first is the average precision (AP) loss, which is very popular in the computer vision community as evidenced by its use in the various challenges of PASCAL VOC [8]. The second is the normalized discounted cumulative gain (NDCG) loss, which is very popular in the information retrieval community [6].

Notation. In order to specify the loss functions, and our efficient algorithms for problem (4), it would be helpful to introduce some additional notation. We define $ind(\mathbf{x})$ to be the index of a sample $\mathbf{x}$ according to the ranking $\mathbf{R}$. Note that the notation does not explicitly depend on $\mathbf{R}$ as the ranking will always be clear from context. If $\mathbf{x} \in \mathcal{P}$ (that is, for a positive sample), we define $ind^+(\mathbf{x})$ as the index of $\mathbf{x}$ in the total order of positive samples induced by $\mathbf{R}$. For example, if $\mathbf{x}$ is the highest ranked positive sample then $ind^+(\mathbf{x}) = 1$ even though $ind(\mathbf{x})$ need not necessarily be 1 (in the case where some negative samples are ranked higher than $\mathbf{x}$). For a negative sample $\mathbf{x} \in \mathcal{N}$, we define $ind^-(\mathbf{x})$ analogously: $ind^-(\mathbf{x})$ is the index of $\mathbf{x}$ in the total order of negative samples induced by $\mathbf{R}$.

AP Loss. Using the above notation, we can now concisely define the average precision (AP) loss of a proper ranking $\mathbf{R}$ given the ground truth ranking $\mathbf{R}^*$ as follows:

$$\Delta_{AP}(\mathbf{R}^*, \mathbf{R}) = 1 - \frac{1}{|\mathcal{P}|} \sum_{\mathbf{x} \in \mathcal{P}} \frac{ind^+(\mathbf{x})}{ind(\mathbf{x})}.$$

For example, consider an input $\mathbf{X} = \{\mathbf{x}_1, \cdots, \mathbf{x}_8\}$ where $\mathbf{x}_i \in \mathcal{P}$ for $1 \le i \le 4$, and $\mathbf{x}_i \in \mathcal{N}$ for $5 \le i \le 8$, that is, the first 4 samples are positive while the last 4 samples are negative. If the proper ranking $\mathbf{R}$ induces the order

$$(\mathbf{x}_1, \mathbf{x}_3, \mathbf{x}_8, \mathbf{x}_4, \mathbf{x}_5, \mathbf{x}_2, \mathbf{x}_6, \mathbf{x}_7), \tag{7}$$

$$\text{then,} \quad \Delta_{AP}(\mathbf{R}^*, \mathbf{R}) = 1 - \frac{1}{4}\left(\frac{1}{1} + \frac{2}{2} + \frac{3}{4} + \frac{4}{6}\right) \approx 0.146.$$

NDCG Loss. We define a discount [2] $D(i) = 1/\log_2(1+i)$ for all $i = 1, \cdots, |\mathcal{N}| + |\mathcal{P}|$. This allows us to obtain a loss function based on the normalized discounted cumulative gain as

$$\Delta_{NDCG}(\mathbf{R}^*, \mathbf{R}) = 1 - \frac{\sum_{\mathbf{x} \in \mathcal{P}} D(ind(\mathbf{x}))}{\sum_{i=1}^{|\mathcal{P}|} D(i)}.$$

For example, consider the aforementioned input where the first four samples are positive and the last four samples are

[2] Chakrabarti *et al.* [6] use a slightly modified definition of the discount $D(\cdot)$. For a detailed discussion about it the reader can refer to the Appendix (Supplementary).

negative. For the ranking $\mathbf{R}$ that induces the order (7), we can compute

$$\Delta_{NDCG}(\hat{\mathbf{R}}, \mathbf{R}) = 1 - \frac{1 + \log_2^{-1} 3 + \log_2^{-1} 5 + \log_2^{-1} 7}{1 + \log_2^{-1} 3 + \log_2^{-1} 4 + \log_2^{-1} 5} \approx 0.056.$$

Both AP loss and NDCG loss are functions of the entire dataset and are not decomposable onto individual samples.

3. Quicksort Flavored Optimization

In order to estimate the parameters $\mathbf{w}$ in the ranking framework by solving problem (4), we need to compute the semi-gradient of $J(\mathbf{w})$. To this end, given the current estimate of parameters $\mathbf{w}$, as well as a set of samples $\mathbf{X}$, we are interested in obtaining the most violated ranking by solving problem (6). At first glance, the problem seems to require us to obtain a ranking matrix $\bar{\mathbf{R}}$. However, it turns out that we do not explicitly require a ranking matrix.

In more detail, our algorithm uses an intermediate representation of the ranking using the notion of interleaving ranks. Given a ranking $\mathbf{R}$ and a negative sample $\mathbf{x}$, the interleaving rank $rank(\mathbf{x})$ is defined as one plus the number of positive samples preceding $\mathbf{x}$ in $\mathbf{R}$. Note that, similar to our notation for $ind(\cdot)$, $ind^+(\cdot)$ and $ind^-(\cdot)$, we have dropped the dependency of $rank(\cdot)$ on $\mathbf{R}$ as the ranking matrix would be clear from context. The interleaving rank of all the samples does not specify the total ordering of all the samples according to $\mathbf{R}$ as it ignores the relative ranking of the positive samples among themselves, and the relative ranking of the negative samples among themselves. However, as will be seen shortly, for a large class of ranking based loss functions, interleaving ranks corresponding to the most violating ranking are sufficient to compute the semi-gradient as in equation (5).

In the rest of the section, we discuss the class of loss functions that are amenable to a quicksort flavored algorithm, which we call QS-suitable loss functions. We then describe and analyze our quicksort flavored approach for finding the interleaving rank in some detail. For brevity and simplicity of exposition, in the following sub-section, we restrict our discussion to the properties of QS-suitable loss functions that are necessary for an intuitive explanation of our algorithm. For a thorough discussion on the characterization and properties of QS-suitable loss functions, we refer the interested reader to the full version of the paper [19].

3.1. QS-Suitable Loss Functions

As discussed earlier, many popular rank-based loss functions happen to be non-decomposable. That is, they can not be additively decomposed onto individual samples. However, it turns out that a wide class of such non-decomposable loss functions can be instead additively decomposed onto the negative samples. Formally, for some functions $\delta_j : \{1, \ldots, |\mathcal{P}|+1\} \to \mathbb{R}$ for $j = 1, \ldots, |\mathcal{N}|$, for a proper ranking $\mathbf{R}$ one can write

$$\Delta(\mathbf{R}^*, \mathbf{R}) = \sum_{\mathbf{x} \in \mathcal{N}} \delta_{ind^-(\mathbf{x})}(rank(\mathbf{x})).$$

We will call this the *negative-decomposability* property.

Further, many of those rank-based loss functions do not depend on the relative order of positive or negative samples among themselves. Rather, the loss for a ranking $\mathbf{R}$, $\Delta(\mathbf{R}^*, \mathbf{R})$, depends only on the interleaving rank of positive and negative samples corresponding to $\mathbf{R}$. We will call this the *interleaving-dependence* property.

As will be evident later in the section, the above properties in a loss function allows for an efficient quicksort flavored divide and conquer algorithm to solve the loss augmented problem. We formally define the class of loss functions that allow for such a quicksort flavored algorithm as QS-*suitable* loss functions. The following proposition establishes the usefulness for such a characterization.

Proposition 1 *Both Δ_{AP} and Δ_{NDCG} are QS-suitable.*

The proof of the above proposition is provided in Appendix (supplementary). Having established that both the AP and the NDCG loss are QS-suitable, the rest of the section will deal with a general QS-suitable loss function. A reader who is interested in employing another loss function need only check whether the required conditions are satisfied in order to use our approach.

3.2. Key Observations for QS-Suitable Loss

Before describing our algorithm in detail, we first provide some key observations which enable efficient optimization for QS-suitable loss functions. To this end, let us define an array $\{s_i^+\}_{i=1}^{|\mathcal{P}|}$ of positive sample scores and an array $\{s_i^-\}_{i=1}^{|\mathcal{N}|}$ of negative sample scores. Furthermore, for purely notational purposes, let $\{s_i^*\}$ be the array $\{s_i^-\}$ sorted in descending order. For $j \in \{1, \ldots |\mathcal{N}|\}$ we denote the index of s_j^- in $\{s_i^*\}$ as j^*.

With the above notation, we describe some key observations regarding QS-suitable loss functions. Their proofs are for most part straightforward generalizations of results that appeared in [18, 27] in the context of the AP loss and can be found in Appendix (Supplementary). Using the interleaving-dependence property of QS-suitable loss functions and structure of the discriminant function as defined in equation (1), we can make the following observation.

Observation 1 *An optimal solution $\bar{\mathbf{R}}$ of problem (6) would have positive samples appearing in the descending order of their scores s_i^+ and also the negative samples appearing in descending order of their scores s_i^-.*

Now, in order to find the optimal ranking $\bar{\mathbf{R}}$, it would seem natural to sort the arrays $\{s_i^+\}$ and $\{s_i^-\}$ in descending order and then find the optimal interleaving ranks $rank(\mathbf{x})$

for all $\mathbf{x} \in \mathcal{N}$. However, we are aiming for complexity below $O(|\mathcal{N}| \log |\mathcal{N}|)$, therefore we can not afford to sort the negative scores. On the other hand, since $|\mathcal{P}| << |\mathcal{N}|$, we are allowed to sort the array of positive scores $\{s_i^+\}$.

Let opt_i be the optimal interleaving rank for the negative sample with the i^{th} rank in the sorted list $\{s_i^*\}$ and $\mathbf{opt} = \{opt_i | j = 1, \ldots, |\mathcal{N}|\}$ be the optimal interleaving rank vector. A certain subtle monotonicity property QS-suitable loss functions (see Supplementary) and the structure of the discriminant function given in (1) gives us the opportunity to compute the interleaving rank for each negative sample independently. However, we actually need not do this computation for all the $|\mathcal{N}|$ negative samples. This is because, since the interleaving rank for any negative sample can only belong to $[1, |\mathcal{P}| + 1]$ and $|\mathcal{P}| << |\mathcal{N}|$, many of the negative samples would have the same interleaving rank. This fact can be leveraged to improve the efficiency of the algorithm for finding $\mathbf{opt}$ by making use of the following observation.

Observation 2 *If $i < j$, then $opt_i \leq opt_j$.*

Knowing that $opt_i = opt_j$ for some $i < j$, we can conclude that $opt_i = opt_k = opt_j$ for each $i < k < j$. This provides a cheap way to compute some parts of the vector $\mathbf{opt}$ if an appropriate sequence is followed for computing the interleaving ranks. Even without access to the fully sorted set $\{s_j^*\}$, we can still find s_j^*, the j-highest element in $\{s_i^-\}$, for a fixed j, in $O(|\mathcal{N}|)$ time. This would lead to an $O(|\mathcal{P}|\,|\mathcal{N}|)$ algorithm but we may at each step modify $\{s_i^-\}$ slowly introducing the correct order. This will make the future searches for s_j^* more efficient.

3.3. Divide and Conquer

Algorithm 1 describes the main steps of our approach. Briefly, we begin by detecting $s^*_{|\mathcal{N}|/2}$ that is the median score among the negative samples. We use this to compute $opt_{|\mathcal{N}|/2}$. Given $opt_{|\mathcal{N}|/2}$, we know that for all $j < |\mathcal{N}|/2$, $opt_j \in [1, opt_{|\mathcal{N}|/2}]$ and for all $j > |\mathcal{N}|/2$, $opt_j \in [opt_{|\mathcal{N}|/2}, |\mathcal{P}| + 1]$. This observation allows us to employ a divide-and-conquer recursive approach.

In more detail, we use two classical linear time array manipulating procedures MEDIAN and SELECT. The first one outputs the index of the median element. The second one takes as its input an index of a particular element x. It rearranges the array such that x separates higher-ranked elements from lower-ranked elements (in some total order). For example, if array s^- contains six scores $[a\ b\ 4.5\ 6\ 1\ c]$ then `Median`(3, 5) would return 3 (the index of score 4.5), while calling `Select`(3, 3, 5) would rearrange the array to $[a\ b\ 1\ 4.5\ 6\ c]$ and return 4 (the new index of 4.5). The SELECT procedure is a subroutine of the classical QUICKSORT algorithm.

Using the two aforementioned procedures in conjunction with the divide-and-conquer strategy allows us to compute the entire interleaving rank vector $\mathbf{opt}$ and this in turn allows us to compute the semi-gradient $\nabla_{\mathbf{w}} J(\mathbf{w})$, as in equation (5), efficiently.

Algorithm 1: Recursive procedure for finding all interleaving ranks.

Description: The function finds optimal interleaving rank for all $i \in [\ell^-, r^-]$ given that
(i) array s^- is partially sorted, namely
$\texttt{MAX}(s^-[1 \ldots \ell^- - 1]) \leq \texttt{MIN}(s^-[\ell^- \ldots r^-])$ and
$\texttt{MAX}(s^-[\ell^- \ldots r^-]) \leq \texttt{MIN}(s^-[r^- + 1 \ldots |\mathcal{N}|])$;
(ii) optimal interleaving ranks for $i \in [\ell^-, r^-]$ lie in the interval $[\ell^+, r^+]$.

1 **function** `OptRanks`(int ℓ^-, int r^-, int ℓ^+, int r^+)
2 **if** $\ell^+ = r^+$ **then**
3 set $opt_i = \ell^+$ for each $i \in [\ell^-, r^-]$ and return
4 **end**
5 $m = \texttt{Median}(\ell^-, r^-)$ ▷ *gives the index of the median score in a subarray of s^-*
6 $m = \texttt{Select}(m, \ell^-, r^-)$ ▷ *splits the subarray by $s = s^-[m]$, returns the new index of s*
7 Find opt_m by trying all options in $[\ell^+, r^+]$
8 **if** $\ell^- < m$ **then** `OptRanks`(ℓ^-, $m-1$, ℓ^+, opt_m)
9 **if** $m < r^-$ **then** `OptRanks`($m+1$, r^-, opt_m, r^+)

Figure 1 provides an illustrative example of our divided-and-conquer strategy. Here, $|\mathcal{N}| = 11$ and $|\mathcal{P}| = 2$. We assume that the optimal interleaving rank vector $\mathbf{opt}$ is $[1, 2, 2, 2, 2, 2, 2, 2, 2, 3, 3]$. Let us now go through the procedure in which Algorithm 1 computes this optimal interleaving rank vector. Before starting the recursive procedure, we only sort the positive samples according to their scores and do not sort the negative samples. To start with, we call `OptRanks`$(1, 11, 1, 3)$. We find the negative sample with the median score (6^{th} highest in this case) and compute its optimal interleaving rank opt_6 to be 2. In the next step of the recursion, we make the following calls: `OptRanks`$(1, 5, 1, 2)$ and `OptRanks`$(7, 11, 2, 3)$. These calls compute opt_3 and opt_9 to be 2. In the next set of recursion calls however, the calls `OptRanks`$(4, 5, 2, 2)$ and `OptRanks`$(7, 8, 2, 2)$, get terminated in step 4 of Algorithm 1 and opt_j for $j = 4, 5, 7, 8$ are assigned without any additional computation. We then continue this procedure recursively for progressively smaller intervals as described in Algorithm 1. Leveraging the fact stated in observation 2, our algorithm has to explicitly compute the interleaving rank for only 6 (shown in square brackets) out of the 11 negative samples. In a typical real data set, which is skewed more in favor of the negative samples, the expected number of negative samples for which is the interleaving rank has to be explicitly computed is far less than $|\mathcal{N}|$. In contrast, the algorithm proposed by Yue *et al.* in

-	-	-	-	-	-	-	-	-	-	-
-	-	-	-	-	[2]	-	-	-	-	-
-	-	[2]	-	-	[2]	-	-	[2]	-	-
-	[2]	[2]	2	2	[2]	2	2	[2]	[3]	-
[1]	[2]	[2]	2	2	[2]	2	2	[2]	[3]	3

Figure 1. *Example illustrating the path followed by the quick sort flavored recursive algorithm while computing the interleaving rank vector* $\mathbf{opt}$. *Row correspond to the status of* $\mathbf{opt}$ *at selected time steps.*

[27] first sorts the entire negative set in descending order of their scores and explicitly computes the interleaving rank for each of the $|\mathcal{N}|$ negative samples.

3.4. Computational Complexity

The computational complexity of the divide-and-conquer strategy to estimate the output of problem (6), is given by the following theorem.

Theorem 2 *If Δ is QS-suitable, then the task (6) can be solved in time $O(|\mathcal{N}|\log|\mathcal{P}| + |\mathcal{P}|\log|\mathcal{P}| + |\mathcal{P}|\log|\mathcal{N}|)$, which in the most common case $|\mathcal{N}| > |\mathcal{P}|$ reduces to $O(|\mathcal{N}|\log|\mathcal{P}|)$ and any comparison-based algorithm would require $\Omega(|\mathcal{N}|\log|\mathcal{P}|)$ operations.*

Proof. Please refer to Appendix (Supplementary). □

Note that the above theorem not only establishes the superior runtime of our approach ($O(|\mathcal{N}|\log|\mathcal{P}|)$ compared to $O(|\mathcal{N}|\log|\mathcal{N}|)$ of [27] and [18]), it also provides an asymptotic lower bound for comparison based algorithms. However, it does not rule out the possibility of improving the constants hidden within the asymptotic notation for a given loss function. For example, as mentioned earlier, one can exploit the additional structure of the AP loss, as presented in [18], to further speed-up our algorithm.

4. Experiments

We demonstrate the efficacy of our approach on three vision tasks with increasing level of complexity. First, we use the simple experimental setup of doing action classification on the PASCAL VOC 2011 data set using a shallow model. This experimental set up allow us to thoroughly analyze the performance of our method as well as the baselines by varying the sample set sizes. Second, we apply our method to a large scale experiment of doing object detection on the PASCAL VOC 2007 data set using a shallow model. This demonstrates that our approach can be used in conjunction with a large data set consisting of millions of samples. Finally, we demonstrate the effectiveness of our method for layer wise training of a deep network on the task of image classification using the CIFAR-10 data set.

4.1. Action Classification

Data set. We use the PASCAL VOC 2011 [8] action classification data set for our experiments. This data set consists of 4846 images, which include 10 different action classes. The data set is divided into two parts: 3347 'trainval' person bounding boxes and 3363 'test' person bounding boxes. We use the 'trainval' bounding boxes for training since their ground-truth action classes are known. We evaluate the accuracy of the different models on the 'test' bounding boxes using the PASCAL evaluation server.

Model. We use structured SVM models as discriminant functions and use the standard poselet [17] activation features to define the sample feature for each person bounding box. The feature vector consists of 2400 action poselet activations and 4 object detection scores. We refer the reader to [17] for details regarding the feature vector.

Methods. We show the effectiveness of our method in optimizing both AP loss and NDCG loss to learn the model parameters. Specifically, we report the computational time for the loss-augmented inference evaluations. For AP loss, we compare our method (referred to as AP_QS) with the loss-augmented inference procedure described in [27] (referred to as AP). For NDCG loss, we compare our method (referred to as NDCG_QS) with the loss-augmented inference procedure described in [6] (referred to as NDCG). We also report results for loss-augmented inference evaluations when using the simple decomposable 0-1 loss function (referred to as 0-1). The hyperparameters involved are fixed using 5-fold cross-validation on the 'trainval' set.

Results. When we minimize AP loss on the training set to learn the model parameters, we get a mean AP of 51.196 on the test set. In comparison, minimizing 0-1 loss to learn model parameters leads to a mean AP value of 47.934 on the test set. Similarly, minimizing NDCG loss for parameter learning gives a superior mean NDCG value of 85.521 on the test set, compared to that of 84.3823 when using 0-1 loss. The AP and NDCG values obtained on the test set for individual action classes can be found in the supplementary material. This clearly demonstrates the usefulness of directly using rank based loss functions like AP loss and NDCG loss for learning model parameters, instead of using simple decomposable loss functions like 0-1 loss as surrogates.

The time required for the loss augmented inference eval-

0-1	AP	AP_QS	NDCG	NDCG_QS
0.0694	0.7154	0.0625	6.8019	0.0473

Table 1. *Total computation time (in seconds) when using the different methods, for multiple calls to loss augmented inference during model training. The reported time is averaged over the training for all the action classes.*

0-1	AP	AP_QS	NDCG	NDCG_QS
0.48±0.03	16.29±0.18	1.48±0.39	71.07±1.57	0.55±0.11

Table 2. *Mean computation time (in milli-seconds) when using the different methods, for single call to loss augmented inference. The reported time is averaged over all training iterations and over all the action classes.*

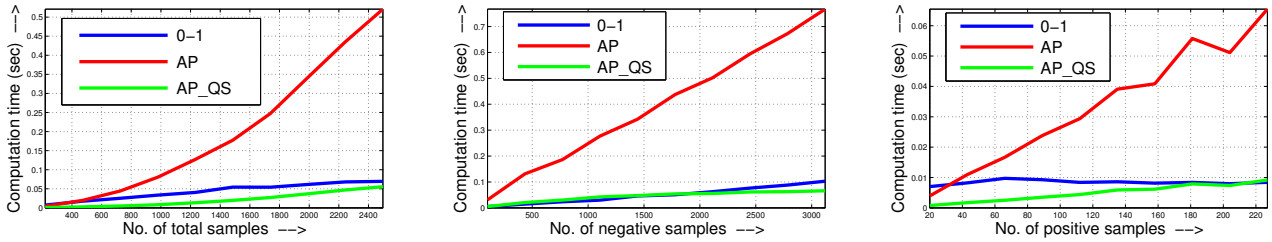

Figure 2. *Total computation time for multiple calls to loss augmented inference during model training, when the number of total, negative and positive samples are varied. Here, 0-1,* AP *and* AP_QS *correspond to loss augmented inference procedures for 0-1 loss, for* AP *loss using [27] and for* AP *loss using our method respectively. It can be seen that our method scales really well with respect to sample set sizes and takes computational time that is comparable to what is required for simpler 0-1 decomposable loss.*

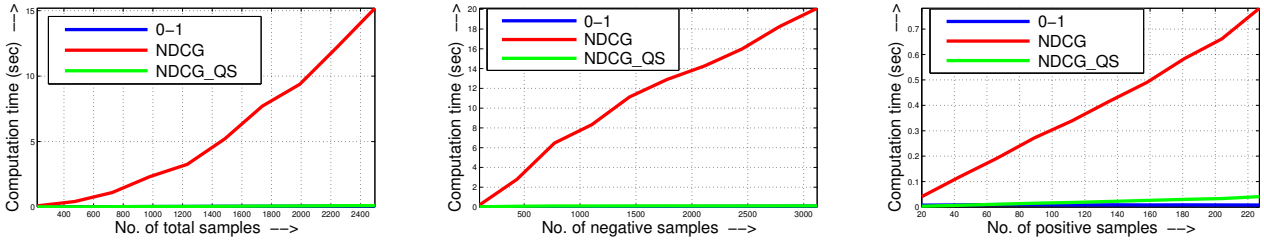

Figure 3. *Total computation time for multiple calls to loss augmented inference during model training, when the number of total, negative and positive samples are varied. Here, 0-1,* NDCG *and* NDCG_QS *correspond to loss augmented inference procedures for 0-1 loss, for* NDCG *loss using [6] and for* NDCG *loss using our method respectively. As can be seen, our approach scales elegantly with respect to sample set sizes and is comparable to the simpler 0-1 decomposable loss in terms of computation time.*

uations, while optimizing the different loss functions for learning model parameters, are shown in Table 1. It can be seen that using our method (AP_QS, NDCG_QS) leads to reduction in computational time by a factor of more than 10, when compared to the methods proposed in [27] and [6] for AP loss and NDCG loss respectively. For AP loss, the method proposed in [18] takes computational time of 0.0985 sec for loss augmented inference. Note that, this method is specific to AP loss, but our more general method is still around 3 times faster. It can also be observed that although the computational time for each call to loss-augmented inference for 0-1 loss is slightly less than that for AP loss and NDCG loss (Table 2), in some cases we observe that we required more calls to optimize the 0-1 loss. As a result, in those cases training using 0-1 loss is slower than training using AP or NDCG loss with our proposed method.

In order to understand the effect of the size and composition of the data set on our approaches, we perform 3 experiments with variable number of samples for the action class phoning. First, we vary the total number of samples while fixing the positive to negative ratio to $1 : 10$. Second, we vary the number of negative samples while fixing the number of positive samples to 227. Third, we vary the number of positive samples while fixing the number of negative samples to 200. As can be seen in Fig. 2 and Fig. 3, the time required for loss-augmented inference is significantly lower using our approach for both AP and NDCG loss.

4.2. Object Detection

Data set. We use the PASCAL VOC 2007 [8] object detection data set, which consists of a total of 9963 images. The data set is divided into a 'trainval' set of 5011 images and a 'test' set of 4952 images. All the images are labeled to indicate the presence or absence of the instances of 20 different object categories. In addition, we are also provided with tight bounding boxes around the object instances, which we ignore during training and testing. Instead, we treat the location of the objects as a latent variable. In order to reduce the latent variable space, we use the selective-search algorithm [26] in its fast mode, which generates an average of 2000 candidate windows per image. This results in a training set size of approximately 10 million bounding boxes.

Model. For each candidate window, we use a feature representation that is extracted from a trained Convolutional Neural Network (CNN). Specifically, we pass the image as input to the CNN and use the activation vector of the penultimate layer of the CNN as the feature vector. Inspired by the R-CNN pipeline of Girshick *et al.* [9], we use the CNN that is trained on the ImageNet data set [7], by rescaling each candidate window to a fixed size of 224×224. The length of the resulting feature vector is 4096. However, in contrast to [9], we do not assume ground-truth bounding boxes to be available for training images. We instead optimize AP loss in a weakly supervised framework to learn the parameters of

the SVM based object detectors for the 20 object categories.

Methods. We use our approach to learn the parameters of latent AP-SVMs [2] for each object category. In our experiments, we fix the hyperparameters using 5-fold cross-validation. During testing, we evaluate each candidate window generated by selective search and use non-maxima suppression to prune highly overlapping detections.

Results. For this task of weakly supervised object detection, using AP loss for learning model parameters leads to a mean test AP of 36.616 which is significantly better than the 29.4995 obtained using 0-1 loss. The AP values obtained on the test set by the detectors for each object class can be found in the supplementary material. These results establish the usefulness of optimizing AP loss for learning the object detectors. On the other hand, optimizing AP loss for this task places high computational demands due to the size of the data set (5011 'trainval' images) as well as the latent space (2000 candidate windows per image) amounting to around 10 million bounding boxes. We show that using our method for loss-augmented inference (LAI) leads to significant saving in computational time. During training, the total time taken for LAI, averaged over all the 20 classes, was 0.5214 sec for our method which is an order of magnitude better than the 7.623 sec taken by the algorithm proposed in [27]. Thus, using our efficient quicksort flavored algorithm can be critical when optimizing non-decomposable loss functions like AP loss for large scale data sets.

4.3. Image Classification

Data set. We use the CIFAR-10 data set [15], which consists of a total of 60,000 images of size 32×32 pixels. Each image belongs to one of 10 specified classes. The data set is divided into a 'trainval' set of 50,000 images and a 'test' set of 10,000 images. From the 50,000 'trainval' images, we use 45,000 for training and 5,000 for validation. For our experiments, all the images are centered and normalized.

Model. We use a deep neural network as our classification model. Specifically, we use a piecewise linear convolutional neural network (PL-CNN) as proposed in [3]. We follow the same framework as [3] for experiments on the CIFAR-10 data set and use a PL-CNN architecture comprising 6 convolutional layers and an SVM last layer. For all our experiments, we use a network that is pre-trained using softmax and cross-entropy loss.

Methods. We learn the weights of the PL-CNN by optimizing AP loss and NDCG loss for the training data set. For comparison, we also report results for parameter learning using the simple decomposable 0-1 loss. We use the layerwise optimization algorithm called LW-SVM, proposed in [3], for optimizing the different loss functions with respect to the network weights. Following the training regime used in [3], we warm start the optimization with a few epochs of Adadelta [28] before running the layer wise optimization. The LW-SVM algorithm involves solving a structured SVM problem for one layer at a time. This requires tens of thousands of calls to loss augmented inference and having a efficient procedure is therefore critical for scalability. We compare our method for loss-augmented inference with the methods described in [27] and [6], for AP loss and NDCG loss respectively.

Results. We get a better mean AP of 85.28 on the test set when we directly optimize AP loss for learning network weights compared to that of 84.22 for 0-1 loss. Similarly, directly optimizing NDCG loss leads to a better mean NDCG of 96.14 on the test set compared to 95.31 for 0-1 loss. This establishes the usefulness of optimizing non-decomposable loss functions like the AP loss and NDCG loss. The LW-SVM algorithm involves very high number of calls to the loss augmented inference procedure. In light of this, the efficient method for loss augmented inference proposed in this paper leads to significant reduction in total training time. When optimizing the AP loss, using our method leads to a total training time of 1.589 hrs compared to that of 1.974 hrs for the algorithm proposed in [27]. Similarly, when optimizing NDCG loss, our method leads to a total training time of 1.632 hrs, which is significantly better than the 2.217 hrs taken for training when using the method proposed in [6]. This indicates that using our method helps the layerwise training procedure scale much better.

5. Discussion

We provided a characterization of ranking based loss functions that are amenable to a quicksort based optimization algorithm for the loss augmented inference problem. We proved that the our algorithm provides a better computational complexity than the state of the art methods for AP and NDCG loss functions and also established that the complexity of our algorithm cannot be improved upon asymptotically by any comparison based method. We empirically demonstrated the efficacy of our approach on challenging real world vision problems. In future, we would like to explore extending our approach to other ranking based non-decomposable loss functions like those based on the F-measure or the mean reciprocal rank.

Acknowledgement

This work is partially funded by the EPSRC grants EP/P020658/1 and TU/B/000048 and a CEFIPRA grant. MR and VK were supported by the European Research Council under the European Unions Seventh Framework Programme (FP7/2007-2013)/ERC grant agreement no 616160. PM was partially supported by a Microsoft Research Travel Grant for attending this conference.

References

[1] B. Bartell, G. Cottrell, and R. Belew. Automatic combination of multiple ranked retrieval systems. In *Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*, 1994. 1

[2] A. Behl, C. V. Jawahar, and M. P. Kumar. Optimizing average precision using weakly supervised data. In *CVPR*, 2014. 8

[3] L. Berrada, A. Zisserman, and M. P. Kumar. Trusting SVM for piecewise linear CNNs. In *International Conference on Learning Representations*, 2017. 8

[4] C. Burges, R. Ragno, and Q. V. Le. Learning to rank with nonsmooth cost functions. In *Advances in neural information processing systems*, 2007. 1

[5] R. Caruana, A. Niculescu-Mizil, G. Crew, and A. Ksikes. Ensemble selection from libraries of models. In *Proceedings of the twenty-first international conference on Machine learning*. ACM, 2004. 1

[6] S. Chakrabarti, R. Khanna, U. Sawant, and C. Bhattacharyya. Structured learning for non-smooth ranking losses. In *KDD*, 2008. 1, 2, 3, 6, 7, 8

[7] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009. 7

[8] M. Everingham, L. Van Gool, C. Williams, J. Winn, and A. Zisserman. The PASCAL visual object classes (VOC) challenge. *IJCV*, 2010. 3, 6, 7

[9] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014. 7

[10] T. Hazan, J. Keshet, and D. McAllester. Direct loss minimization for structured prediction. In *Advances in Neural Information Processing Systems*, 2010. 1

[11] A. Herschtal and B. Raskutti. Optimising area under the ROC curve using gradient descent. In *Proceedings of the twenty-first international conference on Machine learning*. ACM, 2004. 1

[12] T. Joachims. A support vector method for multivariate performance measures. In *ICML*, 2005. 2, 3

[13] T. Joachims, T. Finley, and C. Yu. Cutting-plane training for structural SVMs. *JMLR*, 2009. 3

[14] D. Kim. Minimizing structural risk on decision tree classification. In *Multi-Objective Machine Learning*. Springer, 2006. 1

[15] A. Krizhevsky. Learning multiple layers of features from tiny images. *Technical report, University of Toronto*. 8

[16] Y. Lin, Y. Lee, and G. Wahba. Support vector machines for classification in nonstandard situations. *Machine learning*, 2002. 1

[17] S. Maji, L. Bourdev, and J. Malik. Action recognition from a distributed representation of pose and appearance. In *CVPR*, 2011. 6

[18] P. Mohapatra, C. V. Jawahar, and M. P. Kumar. Efficient optimization for average precision SVM. In *NIPS*, 2014. 2, 4, 6, 7

[19] P. Mohapatra, M. Rolinek, C. Jawahar, V. Kolmogorov, and M. Kumar. Efficient optimization for rank-based loss functions. *arXiv preprint arXiv:1604.08269*. 4

[20] K. Morik, P. Brockhausen, and T. Joachims. Combining statistical learning with a knowledge-based approach: a case study in intensive care monitoring. Technical report, Technical Report, SFB 475: Komplexitätsreduktion in Multivariaten Datenstrukturen, Universität Dortmund, 1999. 1

[21] C. Shen, H. Li, and N. Barnes. Totally corrective boosting for regularized risk minimization. *arXiv preprint arXiv:1008.5188*, 2010. 1

[22] Y. Song, A. Schwing, R. Zemel, and R. Urtasun. Training deep neural networks via direct loss minimization. In *International Conference on Machine Learning*, 2016. 1, 2, 3

[23] C. Szegedy, A. Toshev, and D. Erhan. Deep neural networks for object detection. In *NIPS*, 2013. 1

[24] B. Taskar, C. Guestrin, and D. Koller. Max-margin Markov networks. In *NIPS*, 2003. 1

[25] I. Tsochantaridis, T. Hofmann, Y. Altun, and T. Joachims. Support vector machine learning for interdependent and structured output spaces. In *ICML*, 2004. 1

[26] J. Uijlings, K. van de Sande, T. Gevers, and A. Smeulders. Selective search for object recognition. *IJCV*, 2013. 7

[27] Y. Yue, T. Finley, F. Radlinski, and T. Joachims. A support vector method for optimizing average precision. In *SIGIR*, 2007. 1, 2, 3, 4, 6, 7, 8

[28] M. Zeiler. ADADELTA: an adaptive learning rate method. In *CoRR*, 2012. 8