Learning Structured Models with the AUC Loss and Its Generalizations

Nir Rosenfeld
Hebrew University
Jerusalem, Israel

Ofer Meshi
TTI Chicago

Danny Tarlow
Microsoft Research
Cambridge, UK

Amir Globerson
Hebrew University
Jerusalem, Israel

Abstract

Many problems involve the prediction of multiple, possibly dependent labels. The structured output prediction framework builds predictors that take these dependencies into account and use them to improve accuracy. In many such tasks, performance is evaluated by the Area Under the ROC Curve (AUC). While a framework for optimizing the AUC loss for unstructured models exists, it does not naturally extend to *structured* models. In this work, we propose a representation and learning formulation for optimizing structured models over the AUC loss, show how our approach generalizes the unstructured case, and provide algorithms for solving the resulting inference and learning problems. We also explore several new variants of the AUC measure which naturally arise from our formulation. Finally, we empirically show the utility of our approach in several domains.

1 Introduction

Many learning problems require the simultaneous prediction of multiple related variables. For example, labeling the pixels of an image with their corresponding objects or labeling words with their parts of speech. The framework of structured output prediction has proven very useful for such tasks (Bakir et al., 2007; Taskar et al., 2003; Tsochantaridis et al., 2004; Lafferty et al., 2001). The main insight of this approach is that it is better to jointly predict the labels, rather than to learn individual classifiers for each label.

Appearing in Proceedings of the 17^{th} International Conference on Artificial Intelligence and Statistics (AISTATS) 2014, Reykjavik, Iceland. JMLR: W&CP volume 33.

As with any learning problem, a key decision is which loss to minimize. A common choice is the Hamming loss, namely the number of labels which the classifier errs on. While the Hamming loss is sometimes appropriate, often it is chosen for computational simplicity rather than for its appropriateness for the task at hand.

For example, consider a link prediction task where one wishes to predict which links in a social network will be realized within some time window. Friendship networks are known to exhibit a high degree of clustering, so modeling inter-link dependencies with structured output models should be advantagous. What loss function is appropriate? In this scenario, it is less likely that we require knowing exactly which links will appear at some specific time point in the future. Rather, we might be satisfied if a model ranks potential links by the likelihood of their future formation. In this case, training under Hamming loss is inappropriate, and may significantly hurt performance. It will encourage the model to focus on aspects that are unimportant (what links will appear exactly) at the expense of aspects that we care about (the relative order).

Many other problems — such as document retrieval, multi-label categorization and medical diagnostics — are structured and exhibit the above characteristics. A natural objective for such tasks is the area under the receiver operator characteristic (ROC) curve, or AUC (Provost et al., 1997; Fawcett, 2006). The AUC measures to which degree a ranked list is consistent with the ground truth. It will be higher when variables that are highly ranked are within the ground truth set. While the AUC can be interpreted as a probabilistic measure for the accuracy of binary classifiers, it can also be thought of as an accuracy measure for predictors of multi-labeled instances. In this paper we focus on the latter and apply an Empirical Risk Minimization (ERM) methodology, where our goal is to minimize the AUC loss over a set of examples.

Ideally, we would like to be able to take the same model that we developed for learning under a Ham-

ming loss objective, replace the Hamming loss with AUC, then proceed to learn. This is possible with many other losses that have been studied recently for structured prediction (e.g., all of those discussed in Tarlow and Zemel (2012)). However, for the AUC loss, this is not possible with existing methods, and developing a method capable of this is the main goal of the current paper. As we show, it is not immediately clear how to naturally extend standard structured output predictors to incorporate the AUC loss because standard structured predictors return a single structured labeling, and not a ranked list. While there have been works on structured prediction for ranking, these have not addressed the AUC loss, nor do they use features which are natural in the context of the problems we consider. Furthermore, while methods for optimizing the AUC loss for unstructured models exist (e.g. Joachims, 2005), extending them to handle structured models is non-trivial.

Our approach relies on a representation of the space of rankings under which (a) the AUC can be effectively optimized, and (b) structured prediction models can be easily extended to handle rankings. This representation allows us to incorporate features which one would use in standard structured prediction, as opposed to features arising in the context of ranking. This is a desirable property since in most cases the available data is labeled rather than ranked. We then prove that our method is in fact a generalization of the AUC optimization framework for unstructured prediction previously proposed by Joachims (2005). Moreover, since the resulting inference problems are hard, we show how to approximate them, by applying the typical local linear programming (LP) relaxation to the problem. Finally, we consider several novel generalizations to the AUC which arise naturally under this representation.

We apply our approach to multiple problems: link prediction, document retrieval, and multi-label classification. In all these we observe performance gains with respect to unstructured prediction that optimizes the AUC loss.

2 Structured Output Prediction

We begin by presenting the standard prediction setting and in the next section extend the model to perform ranking.

In structured output prediction we are given an input vector x, and the goal is to predict a discrete output vector y. Here we are interested in binary vectors $y \in \{0,1\}^n$. For example, in a link prediction task the input x may represent the network at time t, and the goal is to predict which of a set of potential links will form at time $t+1$. In this framework it is common to assume that inputs map to outputs via a linear discrimination rule: $y(x;w) = \operatorname{argmax}_{y'} w^\top \phi(x,y')$, where $\phi(x,y)$ is a function mapping input-output pairs to feature vectors, and w is the corresponding weight vector. Since the output space may be large, enumerating all possible outputs is usually prohibitive. Therefore, in many applications the score function is assumed to decompose over single variables. That is, $w^\top \phi(x,y) = \sum_i w_i^\top \phi_i(x,y_i)$. In this *fully-factored* model each output variable is predicted independently of the others. However, modeling dependencies between multiple variables increases expresiveness and may often improves performance (Bakir et al., 2007; Taskar et al., 2003; Tsochantaridis et al., 2004; Lafferty et al., 2001). For example, in many applications the model naturally decomposes over nodes and edges of a graph G:

$$\begin{aligned} w^\top \phi(x,y) = &\sum_{i \in V(G)} w_i^\top \phi_i(x,y_i) + \sum_{ij \in E(G)} w_{ij}^\top \phi_{ij}(x,y_i,y_j) \\ \equiv &s(y;x,w) \end{aligned} \tag{1}$$

Unfortunately, introducing such dependencies between variables makes the prediction task intractable in general (Shimony, 1994), so in practice approximations are often used (Finley and Joachims, 2008).

In the learning task for structured outputs we have a training set consisting of M labeled examples $\{(x^m, y^m)\}_{m=1}^M$, and we wish to learn w. In this work we use structured SVM, an elegant generalization of binary SVM to structured outputs (Taskar et al., 2003; Tsochantaridis et al., 2004). In particular, w is learned by minimizing the regularized structured hinge loss:

$$\begin{aligned} \min_w \frac{\lambda}{2}\|w\|^2 + \frac{1}{M}\sum_m \max_y \left[w^\top \phi(x^m,y) + \Delta(y,y^m)\right] \\ - w^\top \phi(x^m,y^m) \end{aligned} \tag{2}$$

where λ is the regularization constant, and $\Delta(y,y^m)$ is a label-loss function which determines the cost of predicting y when the true label is y^m. Notice that evaluating this objective involves solving a so called *loss-augmented* prediction problem for each training example.

3 Structured Ranking

Our focus in this work is on prediction of binary vectors $(y_1,\ldots,y_n)$ where it is hard (or unnecessary) to exactly predict which y_i-s have the value 1. Instead the goal is to rank the items $1,\ldots,n$ such that elements with $y_i = 1$ are ranked high.

To obtain a numerical measure for the *goodness* of a ranking, we need to understand how it will be used in

practice. One option is that a user considers the top K elements of the ranking, in which case the natural loss is the number of mistakes within this list (or other normalized measures such as precision at K). However, it is often more likely that users do not have a fixed K, and we cannot say in advance how many items a users will consider. As an example, in recommender systems, some users may sift through many recommendations, while others may consider just a few.

The above setting suggests that we should use a measure that spans all possible thresholds employed by users. This is precisely what is achieved by the AUC measure, as explained next. The ROC curve is a plot used for depicting the tradeoff between the false positive rate (FPR) and the true positive rate (TPR) (Metz, 1978). It can be obtained by gradually 'turning on' variables according to their *rank*. If the variable which has just been turned on corresponds to a true label, the TPR increases by one; if the corresponding label is false, the FPR increases by one. Therefore, the ROC is a non-decreasing step function. In the best case, all the true labels have been turned on first, and the curve quickly saturates. Alternatively, if the correct labels are turned on last the curve grows more slowly. This suggests that the quality of the curve can be measured using the area under it. This area, known as the AUC, is a common performance measure (Provost et al., 1997; Fawcett, 2006), and is routinely used in many machine learning applications. The AUC has several other intuitive interpretations. For example, it can be understood as the probability of correctly discriminating between randomly drawn positive and negative samples (e.g., see Joachims, 2005).

Our goal in this paper is to optimize the AUC loss for structured prediction. Ideally, we would like to be able to take a standard structured predictor for a given problem, and optimize its AUC. In other words, we would like to preserve a feature representation as in Eq. (1) and just change the loss. However, it turns out that this is not easy to do. The difficulty stems from the fact that the ground truth y is a boolean vector, whereas the actual output is a permutation π. There are thus two options. One is to represent the features in permutation space and *translate* the ground truth to that space. Such an approach has been suggested for the unstructured case in Chapelle et al. (2007) and extended to the structured case in Mensink et al. (2011) and Weston and Blitzer (2012). However, this requires defining features in permutation space (such as a penalty for a distance within the permutation) as well as several additional parameters that need to be set (the matrix D in Mensink et al. (2011) and the weights w in Weston and Blitzer (2012)), which are unnatural in our context.

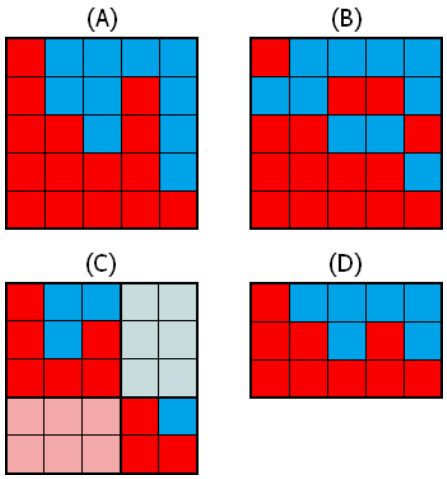

Figure 1: The ranking matrix z in different settings, where 1 is marked in red and 0 in blue. (A) In the basic setting, each row k has k elements on, and nestedness is enforced. The pictured matrix corresponds to the ranking (1,4,2,3,5). (B) In the general setting, nestedness is dropped, allowing for independent predictions at each rank. (C) A ranking matrix $\hat{z}$ consistent with an assignment $y = (11100)$. (D) A binned ranking matrix, with bins of sizes 1,3,5.

Another option is to use the function $s(y; x, w)$ in Eq. (1) to obtain a ranking on the parameters. For example one could set $s_i = \max_{y_{-i}} s(y; x, w)$, namely the max-marginal of y_i, and rank according to it. However, this seems difficult to optimize, and would be further complicated by the need to optimize the AUC of the resulting ranking.

We propose a different solution to the above problem. The idea is to represent the ranking in a structure that is very closely related to the original label space $y_1, \ldots, y_n$. We illustrate this with an example. Consider the ranking $[1, 4, 2, 3, 5]$. It is clearly equivalent to the following lists of subsets of increasing size: $\{1\}, \{1, 4\}, \{1, 4, 2\}, \{1, 4, 2, 3\}$ and $\{1, 4, 2, 3, 5\}$. Note that the subsets themselves are unordered, and the ranking can be recovered by just noting which new element appears between consecutive subsets.

Next, note that this subset representation is equivalent to a set of variables $z_{ki} \in \{0, 1\}$ where $k, i \in [n] = \{1, \ldots, n\}$ and $z_{ki} = 1$ means that the k^{th} subset contains the i^{th} item. This is illustrated graphically in Fig. 1A. In this representation a permutation is obtained by solving n *standard* structured prediction problems, where the outputs of these problems must be nested.

In order for z to faithfully represent a ranking, it must

satisfy two constraints:

- **Cardinality:** At the k^{th} level, exactly k items can be chosen. Namely: $\sum_i z_{ki} = k$ for all k. Note that this is a cardinality constraint over the variables in the k^{th} row (Tarlow et al., 2010).
- **Nestedness:** Adjacent rows must be consistent, in the sense that the $(k+1)^{th}$ row includes the k^{th} one. Formally: $z_{ki} \leq z_{k+1,i}$ for all $i \in [n], k \in [n-1]$.

We denote the set of assignments satisfying these constraints by $\mathcal{Z}$. By construction, each legal assignment $z \in \mathcal{Z}$ corresponds to a ranking π, and vice versa. We therefore sometimes refer to such assignments z as "rankings".

Each row $z_{k\cdot}$ corresponds to a prediction problem where one needs to set k variables out of $y_1, \ldots, y_n$ to one. Thus, $z_{k\cdot}$ have exactly the same meaning as the original y variables. This is precisely what we set out to achieve. In particular, it makes sense to apply the original features to the z variables. The score for a z assignment is then expressed as:

$$w^\top \varphi(x, z) = w^\top \left(\sum_{k=1}^{n} \phi(x, z_{k\cdot}) \right) \tag{3}$$

where ϕ is the same feature used in Eq. (1). Outputting a ranking under this model is done by predicting:

$$z(x; w) = \operatorname*{argmax}_{z' \in \mathcal{Z}} w^\top \varphi(x, z') \tag{4}$$

We notice that this is a generalization of the model in Joachims (2005) since when the model fully factors, then this reduces to: $\max_{z \in \mathcal{Z}} \sum_i \sum_k w_i^\top \phi_i(x, z_{ki})$, which can be calculated by sorting the individual scores. On the other hand, when the model has additional structure, this problem is no easier than the original prediction problem over assignments y, and we will have to resort to some approximation (see Sec. 4).

We next show how to formulate the learning objective to optimize for AUC in our representation. To do so, we replace the maximization over assignments y in the structured hinge loss of Eq. (2) with a maximization over rankings z. The resulting learning problem is therefore:

$$\min_w \frac{\lambda}{2}\|w\|^2 + \frac{1}{M}\sum_m h^m(w, z^m), \quad \text{where:} \tag{5}$$

$$h^m(w, z^m) = \max_{z \in \mathcal{Z}} \left[w^\top \varphi(x^m, z) + \Delta_{\text{AUC}}(z, y^m) \right] - w^\top \varphi(x^m, z^m)$$

where $\Delta_{\text{AUC}}(z, y^m)$ is the *AUC loss*, defined as 1 minus the AUC for the ranking z under the ground-truth assignment y^m. Another appealing property of the proposed representation is that Δ_{AUC} decomposes over single variables (see Sec. 4).

However, the formulation in Eq. (5) is misleading, since we do not have access to a correct, ground truth, ranking z^m. Instead, we can replace it with a ranking $\hat{z}$ consistent with y^m. By consistent, we mean a ranking where elements with $y_i^m = 1$ are ranked higher than elements with $y_i^m = 0$ (see Fig. 1C). We denote this consistency by $\hat{z} \sim y^m$. For any fixed choice of consistent $\hat{z}$ the resulting objective is a convex upper bound on Δ_{AUC}. However, it makes sense to seek the tightest consistent upper bound, which yields the objective:

$$\min_w \frac{\lambda}{2}\|w\|^2 + \frac{1}{M}\sum_m \max_{z \in \mathcal{Z}} \left[w^\top \varphi(x^m, z) + \Delta_{\text{AUC}}(z, y^m) \right] - \max_{\substack{\hat{z} \in \mathcal{Z} \\ \hat{z} \sim y^m}} w^\top \varphi(x^m, \hat{z}) \tag{6}$$

Unfortunately, this objective is no longer convex, but a local optimum can be found using the DC algorithm (Tao and An, 1997). The algorithm is summarized in Algorithm 1. It proceeds by calculating a linear upper bound on the concave part of the objective (line 4), and then solving for the convex part plus linearization (line 6). This is repeated until convergence. In this work we solve the convex problems (line 6) using the stochastic Frank-Wolfe algorithm recently proposed by (Lacoste-Julien et al., 2013). Algorithm 1 assumes that the inference problems (calculating $\hat{z}^m$ and $h^m(w, \hat{z}^m)$) can be solved efficiently, which is not true in general. Instead, we approximate these optimization problems with an LP relaxation. In the next section we show how to solve the LP relaxations that arise in prediction and learning.

Algorithm 1

1: Initialize w
2: **repeat**
3: **for** $m = 1, \ldots, M$ **do**
4: $\hat{z}^m \leftarrow \operatorname{argmax}_{\substack{\hat{z} \in \mathcal{Z} \\ \hat{z} \sim y^m}} w^\top \varphi(x^m, \hat{z})$
5: **end for**
6: $w \leftarrow \operatorname{argmin}_w \frac{\lambda}{2}\|w\|^2 + \frac{1}{M}\sum_m h^m(w, \hat{z}^m)$
7: **until** convergence

The non-convexity of the above problem may seem particularly troubling when recalling that Joachims (2005) developed a convex objective for the unstructured case. Surprisingly, it turns out that in the unstructured case, the above objective is in fact convex, as the following proposition states (see supplementary material for a proof):

Proposition 3.1 *Eq. (6) generalizes the learning objective of Joachims (2005) to structured models.*

One may note that a tighter upper bound can be attained by dropping consistency, resulting in the structured ramp-loss objective (Chapelle et al., 2008). However, this formulation leads to an even 'less' convex objective, and is no longer a generalization of the unstructured case. Our choice of the maximal consistent ranking can be seen as a middle point in the tradeoff between tightness and convexity. The performance of the ramp loss was comparable in our experiments to that of the consistent loss.

4 Inference

Recall that within the structured learning framework presented, we need to solve the following optimization problems:

$$\begin{aligned} &\text{(a)} \ \max_{z\in\mathcal{Z}} w^\top \varphi(x,z), \\ &\text{(b)} \ \max_{z\in\mathcal{Z}} w^\top \varphi(x,z) + \Delta_{\text{AUC}}(z,y), \\ &\text{(c)} \ \max_{\substack{z\in\mathcal{Z}\\ z\sim y}} w^\top \varphi(x,z) \end{aligned} \tag{7}$$

Below we show that Eq. (7b) and Eq. (7a) are equivalently difficult, since the loss term $\Delta_{\text{AUC}}(z,y)$ factorizes over single variables, so it can be absorbed into the model score. We note that when the model is unstructured the problems in Eq. (7) can be efficiently solved by sorting elements according to their individual scores. However, when additional structure is introduced, this results in a combinatorial problem which is generally hard to solve. In such cases, it makes sense to seek an approximate solution that can be computed efficiently. In the sequel we show how to perform such approximate inference.

Factorizing the AUC loss: Seemingly, $\Delta_{\text{AUC}}(z,y)$ is quite an elaborate function of the ranking z. However, it turns out that in our representation it can actually be rewritten as a simple sum over scores corresponding to single variables. To see this, consider a given ranking $z \in \mathcal{Z}$ and a binary labeling y, and let $pos = \{i : y_i = 1\}$ and $neg = \{i : y_i = 0\}$. As shown in Hand and Till (2001), the AUC can be written as a linear function of z, namely:

$$\Delta_{\text{AUC}}(z,y) = 1 - \frac{1}{|pos|\cdot|neg|} \sum_{i\in pos} \sum_{k} z_{ki} + c \tag{8}$$

where $c = \frac{|pos|+1}{2|neg|}$ is constant. We can rewrite the above as $\sum_{i,k=1}^{n} A_{ki} z_{ki} + \bar{c}$, where we define $A_{ki} = -1/|pos|\cdot|neg|$ if $i \in pos$ and 0 otherwise. Thus, in this representation the AUC loss factorizes and can be absorbed into the singleton scores of the model.

Approximate Inference: We next review a general framework of approximate inference for structured outputs, and show how it can be applied in our setting. Specifically, we use *Linear Programming (LP) relaxations* (Wainwright and Jordan, 2008). To simplify the presentation[1], we assume that the model score decomposes as in Eq. (1). Therefore, using Eq. (3), the problem in (7a) becomes:

$$\max_{z\in\mathcal{Z}} \sum_k \left(\sum_i \theta_i(z_{ki}) + \sum_{ij} \theta_{ij}(z_{ki}, z_{kj}) \right)$$

where, $\theta_i(z_{ki}) \equiv w_i^\top \phi_i(x, z_{ki})$ and $\theta_{ij}(z_{ki}, z_{kj}) \equiv w_{ij}^\top \phi_{ij}(x, z_{ki}, z_{kj})$. In the LP relaxation approach indicator variables μ are introduced. The optimization problem is first cast as an integer LP in those variables. Then, the integrality constraints are relaxed in order to obtain a tractable LP. As we show in the supplementary, in our case this results in the following LP:

$$\begin{aligned} \max_{\mu\in\mathcal{L}(G)} & \sum_{k,i} \sum_{z_{ki}} \mu_{ki}(z_{ki})\theta_i(z_{ki}) + \\ & \sum_{k,ij} \sum_{z_{ki},z_{kj}} \mu_{k,ij}(z_{ki},z_{kj})\theta_{ij}(z_{ki},z_{kj}) \\ \text{s.t.} \quad & \sum_i \mu_{ki}(1) = k \qquad \forall k \in [n] \\ & \mu_{k+1,i}(1) \geq \mu_{ki}(1) \qquad \forall k \in [n-1], i \in [n] \end{aligned} \tag{9}$$

where $\mathcal{L}(G)$ is known as the local marginal polytope, defined as:

$$\begin{aligned} \mathcal{L}(G) = \Big\{ \mu \geq 0 \mid & \sum_{z_{kj}} \mu_{k,ij}(z_{ki}, z_{kj}) = \mu_{ki}(z_{ki}), \\ & \sum_{z_{ki}} \mu_{k,ij}(z_{ki}, z_{kj}) = \mu_{kj}(z_{kj}), \\ & \sum_{z_{ki}} \mu_{ki}(z_{ki}) = 1 \Big\} \end{aligned}$$

Notice that the constraints in $\mathcal{L}(G)$ apply separately to each level k. Since the objective and constraints are linear in μ, the above optimization problem can be solved by standard LP solvers.

To conclude this section, we show how to handle the optimization problem in Eq. (7c). Notice that this problem is similar to the one in Eq. (7a), but has additional constraints $z \sim y$. We observe that in our representation these constraints can be enforced by fixing the assignments of some of the variables. Specifically, we set $z_{ki} = 1$ for all $i \in pos$, $k \in neg$, and $z_{ki} = 0$ for all $i \in neg$, $k \in pos$ (see Fig. 1C). Thus, we are left

[1] Our approach is easily applicable to high-order scores as well.

with a modified optimization problem over the rest of the variables in z. This is equivalent to solving problem (7a) with evidence on some of the variables. The solution can be approximated, as before, with LP relaxation, where the model scores are "collapsed" to reflect the evidence.

5 AUC Generalizations

The AUC can also be viewed as the (normalized) count of swapped positive-negative label pairs:

$$\text{AUC} = \frac{|\{(i,j) : i \in pos, j \in neg, \pi(i) > \pi(j)\}|}{|pos| \cdot |neg|}$$

for a given ranking π. In this sense, the AUC treats all pairs and all ranks uniformly. However, in some ranking tasks this is not a desired trait.

In this section we present several variants to the AUC which address such tasks. These variants arise as natural generalizations to our representation of the AUC over ranking matrices. In section 6 we demonstrate their application. Moreover, by reducing the number of variables and/or constraints, these extensions significantly improve computational complexity.

Binned AUC: In some settings, instead of a full ranking, items are placed into ranked *bins* $b_1 \prec \cdots \prec b_K$. For instance, in web-search query, results are binned into sequential pages. In this setting, the relative rank of two results on the same low-ranked page is of little importance. Hence, the Binned AUC can be defined as the number of pairs swapped only *between* bins. To handle this setting, we replace the notion of ranks with bins. We duplicate the original structured prediction model K times instead of n times, and define it over *partial* ranking matrices z by constraining each row k in z to be of cardinality $\sum_{k' \leq k} |b_{k'}|$, as well as enforcing nestedness (see figure 1D). This can greatly reduce the number of variables, factors and constraints in the model. For instance, if $K \ll n$, then the number of variables and constraints is $O(n)$ instead of $O(n^2)$, and the resulting LP is much smaller.

AUC@k: In other settings, only the highly ranked items are of interest. This scenario has been recently addressed by Partial AUC, which takes into account the area under the ROC curve for a given false-positive range (Narasimhan and Agarwal, 2013). However, in many settings, such as recommendation services, the number of items of interest is a small fixed constant. Since a given false positive value can be reached at any item in the ranking, the above measure is not suitable for these settings. We define the AUC@k measure by discarding swapped pairs ranked $k+1, \ldots, n$. Thereby this measure focuses on ordering true items in the top k ranks. To handle this setting, we simply truncate ranks $k+1, \ldots, n$ from our model.

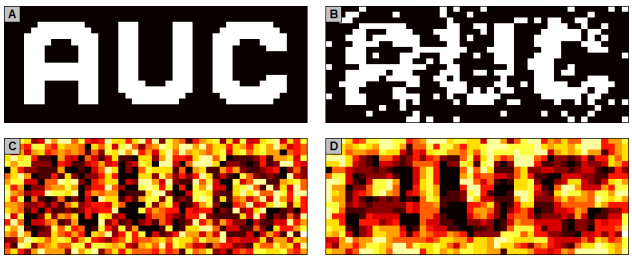

Figure 2: Synthetic image segmentation task. (A) original image, (B) noisy input, (C) binned ranking of factored model (binned AUC=88.7), (D) binned ranking of pairwise model (binned AUC=97.5). Darker colors correspond to higher rank position.

Unnested AUC: Recall that one of the motivations for constructing predictors which attain high AUC is that an end user may use it to predict any number of items $k \in [n]$. The natural approach usually taken is to rank the items and select the top k. However, outputing a ranking is not necessary for producing subsets of size k. In fact, in the general setting, the output can be *any* subset of k elements, where there is no dependency between sets of different sizes. This is suitable for tasks such as contour detection, where as the number of contour edges grows, different objects might be discriminated. The score of such an assignment can be computed by $\sum_{ki} A_{ki} z_{ki} + c$, just as in the standard AUC (see Sec. 4). In our framework, allowing for such outputs simply reduces to omitting the nestedness constraints from Eq. (9), resulting in improved runtime. Hence, this measure can also be thought of as an unnested generalization to AUC (see figure 1B for an illustration.

6 Experiments

To test our approach, we apply it to synthetic and real-world data. We first demonstrate performance on a synthetic image segmentation task, and then report results on the real tasks of link prediction and document retrieval. In all settings we compare the performance of a pairwise model trained over the AUC loss using our proposed method to a logistic regression model and to SVM$_{rank}$ (Joachims, 2005). Due to runtime constraints, the pairwise models were trained using binning. Test-time inference was performed over binned (link prediction) or unbinned (document retrieval) models. Note that since w is shared across

bins, training and inference can be done on different binnings. In all experiments the regularization constant λ was chosen via cross-validation.

Image segmentation: In the binary image segmentation task, noisy images are given as input, and the goal is to predict for each pixel whether it belongs to the foreground or background. We used noisy 20×17 images of the letter A for training. Noise was added by flipping each pixel with a fixed probability $p = 0.15$. We then evaluated performance by predicting a binned ranking over the pixels of three-letter 20×45 images and computing the AUC. We compared a factored model to a pairwise model with interactions between adjacent pixels. In both models 10 equally sized bins were used. As can be seen in Fig. 2, the pairwise factors encourage adjacent pixels to be closely ranked, leading to a higher binned AUC score.

Link prediction: In this task, a graph representing the network at time t is given as input, and the goal is to predict the new links created up to some time $t' > t$ (Liben-Nowell and Kleinberg, 2007). Due to the high sparsity of labels, AUC has become a standard measure of performance for this task. Nonetheless, most link prediction methods do not optimize directly for AUC (an exception is Menon and Elkan (2011)).

We test our method on a collaboration network of NIPS authors in the years 1987-2013. This data was parsed from the DBLP[2] repository and extends the currently available data which spans the years 1987-2003[3]. We take the years 1998-2002, 2003-2007, and 2008-2012 for train, validation, and test sets, respectively (we discard the first years since they are highly disconnected). This results in a roughly 25-25-50 split. Our supervised setting follows the line of Lichtenwalter et al. (2010); Backstrom and Leskovec (2011), where a sample is created for each focal node i, and a prediction consists of a ranking over nodes j at distance two from i. We discard trivial cases by considering only nodes with at least two neighbors at distance two at time t and form at least one new link at t'. A prediction is evaluated by averaging the AUC over all nodes.

As in other supervised methods for link prediction (Al Hasan et al., 2006; Lichtenwalter et al., 2010; Backstrom and Leskovec, 2011), we use well known network-related proximity measures as features on node pairs (i, j) with corresponding links y_{ij}. Specifically, we use the Adamic-Adar (AA), Katz (KZ), Jaccard (JC), and common neighbors (CN) measures (detailed in Liben-Nowell and Kleinberg, 2007). These measures are also used as unsupervised baselines to which we compare our results. In the pairwise model, a feature vector $\phi^i_{jk}(y_{ij}, y_{jk})$ for links $(i, j), (i, k)$ was added if the link (j, k) existed in the network at time t. This vector consisted of an indicator feature and the above proximity measures for the node pair (j, k). In all models the weight vector w was shared across links.

Table 1: AUC scores in different settings. TOP: Document retrieval on the OHSUMED dataset. BOTTOM: Link prediction on the NIPS dataset. Numbers in parentheses indicate the difference in AUC score to column on left.[5]

OHSUMED	SVM_{rank}	Pairwise
Fold 1	63.1	65.2 (+2.1)
Fold 2	64.9	67.5 (+2.6)
Fold 3	65.1	68.3 (+3.2)
Fold 4	60.9	61.0 (+0.1)
Fold 5	61.6	62.0 (+0.4)
Average	63.1	64.8 (+1.7)

Link prediction - NIPS			
Log.Reg.	SVM_{rank}	Pairwise	
58.7	59.0	62.1 (+3.1)	
AA	CN	JC	KZ
58.4	54.5	54.5	55.2

Training and test-time inference were performed over bins of increasing size. Table 1 (bottom) displays the standard AUC of the pairwise model (*'Pairwise'*) as well as that of SVM_{rank} and the above unsupervised methods. As can be seen, the pairwise model shows superior performance.

Document Retrieval: In the task of document retrieval, samples consist of queries and related documents. Given a query, the goal is to rank the related documents in the order of relevance. The ground truth consists of a classification of the documents into relevant and irrelevant, making the AUC a natural performance measure. We test our method on the *OHSUMED* dataset.[6] The dataset consists of 106 medical related queries. Each query is associated with between 35 and 320 (mean 153) medical publications from the MEDLINE database. Each document is as-

[2] `http://www.informatik.uni-trier.de/~ley/db/conf/nips/`

[3] `http://ai.stanford.edu/~gal/data.html`

[5] Logistic Regression on the OHSUMED data performed worse than chance, and therefore is not displayed. We also considered binned factored models, but they attained inferior performance, most likely due to binning.

[6] `http://research.microsoft.com/~letor/`

sociated with 45 standard features, most of which express similarity between query and document. Since documents are labeled as either definitely, possibly, or not relevant, we simply treated both 'possibly' and 'definitely' as relevant. The online package also includes a similarity relation matrix between documents of the same query. For the pairwise model we selected the 100 most similar pairs, and computed 20 similarity features (similar to the above document-query features) over document pairs.

Training was performed with bins of size 20, and inference with unbinned models. As can be seen in Table 1 (top), the pairwise model outperformed SVM_{rank}.

AUC Generalizations: Finally, we conduct experiments to evaluate the performance of the generalized AUC measures in Sec. 5. For lack of space we give the results in the supplementary material. Our results show that the generalized measures are higher than the standard AUC, which is expected as some of the errors are not accounted for. Surprisingly, sometimes training with generalized AUC measure achieves better standard AUC at test time.

7 Related Work

While structured in itself, the AUC has been previously used as an error measure for *unstructured models*. In this case both prediction and learning can be performed efficiently within the SVM framework (Joachims, 2005), as well as with decision trees (Ferri et al., 2002) and neural networks (Herschtal and Raskutti, 2004) among others.

Our goal in this paper is to extend the use of the AUC loss to structured models, where variables have additional direct dependencies. We show in Sec. 6 that these richer models often achieve better performance than the unstructured ones.

In Chapelle et al. (2007) the authors show that unstructured models can be trained to optimize various other error measures for ranking. They suggest to define a prediction problem over the space of permutations, and show that this reduces to a linear assignment problem (or a matching problem), which can be solved in polynomial time. In contrast to our work, they do not handle structured models, and AUC is not one of the measures considered. That work was later extended by Mensink et al. (2011) to handle structured models that output rankings. They show that the prediction problem is hard in the structured case and focus on specific type of models which can be solved efficiently. Our work differs from theirs in several aspects. First, we specifically target the AUC loss. Second, instead of defining the model in the space of permutations, we propose an alternative representation in which it is more natural to incorporate the features used for prediction. Finally, we do not restrict ourselves to tractable instances, but instead propose an approximation scheme for the general case, where prediction is hard.

Weston and Blitzer (2012) use a similar approach to Mensink et al. (2011), and also discusses a variant of the AUC loss. However, their AUC loss is essentially the same as Joachims' unstructured variant and does not apply to complete rankings as we have here.[7] Furthermore, their feature construction is restricted in the same way as Mensink et al. (2011).

Another recent work (Dembczynski et al., 2012) focuses on the rank loss (a generalization of the AUC) in multilabel ranking. The authors show that certain univariate surrogate losses defined over unstructured models are consistent with regard to the AUC, and propose that each label can be learned independently. However, since in general we do not have access to the real distribution, the expressiveness of structured models often leads to superior performance (see Sec. 6). Moreover, this framework assumes that the labels are bound to some fixed label set, which may be true for multilabel ranking, but not for other applications such as link prediction and document retrieval.

8 Conclusion

We have presented a natural generalization of structured prediction models to ranking in general and AUC optimization in particular. Previous approaches directly represented the ranking via a permutation matrix, and were thus limited in the features they could use. Here we represent ranking as a nested sequence of standard labelings of the output variables.This allows us to use natural features for the domain, and does not require hand coded parameters as in permutation based representations.

Our empirical results show that it is indeed worthwhile to optimize AUC in these models, and significant performance gains are achieved compared to unstructured models in a variety of domains.

Acknowledgements

This research is funded by the ISF Centers of Excellence grant 1789/11. We thank Jacob Goldenberg for many useful discussions and suggestions.

[7] Specifically, note that in their Eq. 10 and algorithm, the score function is applied to single documents, and is not consistent with their ranking score in Eq. 3. This is a result of using a greedy algorithm that considers just pairs of documents at each step.

References

M. Al Hasan, V. Chaoji, S. Salem, and M. Zaki. Link prediction using supervised learning. In *SDM06: Workshop on Link Analysis, Counter-terrorism and Security*, 2006.

L. Backstrom and J. Leskovec. Supervised random walks: predicting and recommending links in social networks. In *WSDM*, pages 635–644. ACM, 2011.

G. H. Bakir, T. Hofmann, B. Schölkopf, A. J. Smola, B. Taskar, and S. Vishwanathan. *Predicting structured data*. MIT press, 2007.

O. Chapelle, Q. V. Le, and A. Smola. Large margin optimization of ranking measures. In *NIPS Workshop on Machine Learning for Web Search*, 2007.

O. Chapelle, C. B. Do, C. H. Teo, Q. V. Le, and A. J. Smola. Tighter bounds for structured estimation. In *NIPS*, pages 281–288, 2008.

K. Dembczynski, W. Kotlowski, and E. Huellermeier. Consistent multilabel ranking through univariate losses. *arXiv preprint arXiv:1206.6401*, 2012.

T. Fawcett. An introduction to roc analysis. *Pattern recognition letters*, 27(8):861–874, 2006.

C. Ferri, P. Flach, and J. Hernández-Orallo. Learning decision trees using the area under the roc curve. In *ICML*, volume 2, pages 139–146, 2002.

T. Finley and T. Joachims. Training structural svms when exact inference is intractable. In *ICML*, pages 304–311. ACM, 2008.

D. J. Hand and R. J. Till. A simple generalisation of the area under the roc curve for multiple class classification problems. *Machine Learning*, 45(2):171–186, 2001.

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*, page 49. ACM, 2004.

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

S. Lacoste-Julien, M. Jaggi, M. Schmidt, and P. Pletscher. Block-coordinate Frank-Wolfe optimization for structural SVMs. In *ICML*, pages 53–61, 2013.

J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*, pages 282–289, 2001.

D. Liben-Nowell and J. Kleinberg. The link-prediction problem for social networks. *Journal of the American society for information science and technology*, 58(7): 1019–1031, 2007.

R. N. Lichtenwalter, J. T. Lussier, and N. V. Chawla. New perspectives and methods in link prediction. In *KDD*, pages 243–252. ACM, 2010.

A. Menon and C. Elkan. Link prediction via matrix factorization. *Machine Learning and Knowledge Discovery in Databases*, pages 437–452, 2011.

T. Mensink, J. Verbeek, T. Caetano, et al. Learning to rank and quadratic assignment. In *NIPS Workshop on Discrete Optimization in Machine Learning*, 2011.

C. E. Metz. Basic principles of roc analysis. In *Seminars in nuclear medicine*, volume 8, pages 283–298. Elsevier, 1978.

H. Narasimhan and S. Agarwal. A structural SVM based approach for optimizing partial auc. In *ICML*, pages 516–524, 2013.

F. Provost, T. Fawcett, et al. Analysis and visualization of classifier performance: Comparison under imprecise class and cost distributions. In *KDD*, pages 43–48, 1997.

Y. Shimony. Finding the MAPs for belief networks is NP-hard. *Art. Intell.*, 68(2):399–410, 1994.

P. D. Tao and L. T. H. An. Convex analysis approach to dc programming: theory, algorithms and applications. *Acta Mathematica Vietnamica*, 22(1):289–355, 1997.

D. Tarlow and R. Zemel. Structured output learning with high order loss functions. In *AISTATS*, 2012.

D. Tarlow, I. Givoni, and R. Zemel. Hop-map: Efficient message passing with high order potentials. In *AISTATS*, 2010.

B. Taskar, C. Guestrin, and D. Koller. Max-margin Markov networks. In *NIPS*. MIT Press, 2003.

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

M. J. Wainwright and M. I. Jordan. Graphical models, exponential families, and variational inference. *Foundations and Trends in Machine Learning*, 1(1-2):1–305, 2008.

J. Weston and J. Blitzer. Latent structured ranking. In *UAI*, pages 903–913, 2012.