Technical Report: Learning to Rank using High-Order Information

Puneet K. Dokania[1], Aseem Behl[2], C. V. Jawahar[2], and M. Pawan Kumar[1]

Ecole Centrale de Paris[1], INRIA Saclay[1], IIIT Hyderabad - India[2]

Abstract. The problem of ranking a set of visual samples according to their relevance to a query plays an important role in computer vision. The traditional approach for ranking is to train a binary classifier such as a support vector machine (SVM). Binary classifiers suffer from two main deficiencies: (i) they do not optimize a ranking-based loss function, for example, the average precision (AP) loss; and (ii) they cannot incorporate high-order information such as the *a priori* correlation between the relevance of two visual samples (for example, two persons in the same image tend to perform the same action). We propose two novel learning formulations that allow us to incorporate high-order information for ranking. The first framework, called *high-order binary* SVM (HOB-SVM), allows for a structured input. The parameters of HOB-SVM are learned by minimizing a convex upper bound on a surrogate 0-1 loss function. In order to obtain the ranking of the samples that form the structured input, HOB-SVM sorts the samples according to their *max-marginals*. The second framework, called *high-order average precision* SVM (HOAP-SVM), also allows for a structured input and uses the same ranking criterion. However, in contrast to HOB-SVM, the parameters of HOAP-SVM are learned by minimizing a difference-of-convex upper bound on the AP loss. Using a standard, publicly available dataset for the challenging problem of action classification, we show that both HOB-SVM and HOAP-SVM outperform the baselines that ignore high-order information.

1 Introduction

Many tasks in computer vision require the development of automatic methods that sort a given set of visual samples according to their relevance to a query. For example, consider the problem of action classification (or more precisely action ranking). The input is a set of samples corresponding to bounding boxes of persons, and an action such as ‘jumping’. The desired output is a ranking where a sample representing a jumping person is ranked higher than a sample representing a person performing a different action. Other related problems include image classification (sorting images according to their relevance to a user query) and object detection (sorting all the windows in a set of images according to their relevance to an object category). As the desired output of the aforementioned problems is a ranking, the accuracy of an approach is typically reported using a ranking-based measure such as the average precision (AP).

A popular way of solving a problem that requires us to rank a set of samples is to train a binary classifier. The positive class of the classifier corresponds to the relevant samples and the negative class corresponds to the non-relevant samples. Once a classifier is learned on a training set, a new set of samples is sorted according to the scores assigned to the samples by the classifier. Perhaps the most commonly used classifier is the support vector machine (SVM) [25]. However, the SVM framework has two main drawbacks. First, an SVM minimizes an upper bound on the 0-1 loss function (that is, the fraction of misclassifications) instead of a loss function that depends on the AP. Second, an SVM only uses first-order information to classify a sample, that is, the score of a sample depends only on itself and not on other samples in the dataset. In other words, an SVM does not explicitly incorporate *a priori* high-order information, which can be very useful in improving the accuracy of ranking. For example, in action classification, most of the persons present in the same image tend to perform the same action. In object detection, two objects of the same category tend to have the similar aspect ratio. In pose estimation, people in the same scene tend to have similar poses (sitting down to watch movie). In document retrieval, documents containing same or similar words are more likely to belong to the same class.

At first glance, the two drawbacks seem to be easily fixable using a generalization of the SVM framework, known as structured output support vector machines (SSVM) [22, 24]. Given a structured input, the SSVM framework provides a linear prediction rule to obtain a structured output. Specifically, the score of a putative output is the inner product of the parameters of an SSVM with the joint feature vector of the input and the output. The prediction requires us to maximize the score over all possible outputs for an input. Given a training dataset, the parameters of an SSVM are learned by minimizing a regularized convex upper bound on a user-specified loss function. In the past decade, several customized algorithms have been developed to solve the optimization problem that learns the SSVM parameters [7, 10, 15, 22, 24]. While the optimization algorithms for SSVM differ significantly in their details, they share the common characteristic of iteratively performing *loss-augmented inference*. In other words, given the current estimate of the parameters, they compute the output that jointly maximizes the sum of the score and the loss function. Loss-augmented inference can be viewed as the optimal cutting-plane or the subgradient of the learning objective, which exploits its central role in the optimization.

The SSVM framework places no restriction on the form of the loss function and on the structure of the input and the output. Thus, it appears the ideal framework to (i) optimize the AP loss; and (ii) incorporate high-order information. However, in order to successfully employ an SSVM, we need an efficient algorithm for loss-augmented inference. In that regard, the first drawback can be addressed using AP-SVM [26], which is a special form of SSVM. In the AP-SVM framework, the input is a set of samples and the output is a ranking. The loss value for a putative output is one minus the AP of the corresponding ranking with respect to the ground truth ranking. The joint feature vector of the input and the output is a weighted sum of the feature vectors for all samples, where the weights are governed by the ranking. Yue *et al.* [26] showed that, for this choice of joint feature vector, loss-augmented inference can be performed optimally using an efficient greedy algorithm. Furthermore, they showed that the prediction of AP-SVM is exactly the same as the prediction of the standard SVM, that is, to sort the samples according to their individual scores. Since the joint feature vector of AP-SVM depends only on the feature vectors of the individual samples, AP-SVM does not incorporate high-order information. A straightforward way to address this deficiency would be to modify the joint feature vector such that it depends on feature vectors of pairs of samples, or more generally, on feature vectors of subsets of samples. For example, Rosenfeld *et al.* [21] recently proposed a framework to optimize the area under curve (AUC) while considering the high-order information. However, similar approach cannot be used for optimizing AP based loss function since it does not decompose over single variable. Therefore, such a modification can not be introduced trivially into the AP-SVM formulation.

We present two alternate frameworks to incorporate high-order information for ranking. The first framework, which we call high-order binary SVM (HOB-SVM), takes its inspiration from the standard SVM. The input of HOB-SVM is a set of samples. The output is a binary label for each sample, where the label 1 indicates that the sample is relevant and 0 indicates that the sample is not relevant. The joint feature vector of HOB-SVM depends not only on the feature vectors of the individual samples, but also on the feature vectors of subsets of samples. In this work, we restrict the subsets to be of size two, but our frameworks can easily be generalized to other subset sizes. The loss function of HOB-SVM is a weighted 0-1 loss, which allows us to efficiently perform loss-augmented inference using graph cuts [14]. Practically speaking, the difficulty with employing HOB-SVM is that it provides a single score for the entire labeling of a dataset, whereas we need scores corresponding to each sample in order to find the ranking. To address this difficulty, we propose to rank the samples using the difference between the *max-marginal* for assigning a sample to the relevant class and the max-marginal for assigning it to the non-relevant class. Intuitively, difference of max-marginals measure the positivity of a particular sample while capturing high-order information. Empirically, we show that the difference of max-marginals provides an accurate ranking. The main advantage of HOB-SVM is that its parameters can be estimated efficiently by solving a convex optimization problem. However, its main disadvantage is that, similar to SVM, it optimizes a surrogate loss function instead of the AP loss.

The second framework, which we call high-order AP-SVM (HOAP-SVM), takes its inspiration from AP-SVM and HOB-SVM. Similar to AP-SVM, the input of HOAP-SVM is a set of samples, its output is a ranking of the samples, and its loss function is the AP loss. However, unlike AP-SVM, the score of a ranking is equal to the weighted sum of the difference of max-marginals of the individual samples. Since the max-marginals capture high-order information, and the loss function depends on the AP, HOAP-SVM addresses both the aforementioned deficiencies of traditional classifiers such as SVM. The main disadvantage of HOAP-SVM is that estimating its parameters requires solving a difference-of-convex program

[8]. While we cannot obtain an optimal set of parameters for HOAP-SVM, we show how a local optimum of the HOAP-SVM learning problem can be computed efficiently by the concave-convex procedure [27]. Using standard, publicly available datasets, we empirically demonstrate that HOAP-SVM outperforms the baselines by effectively utilizing high-order information while optimizing the correct loss function. For the sake of clarity, the proofs of all the propositions presented in the paper are given in the accompanying technical report. To facilitate the use of HOB-SVM and HOAP-SVM, we have made our code and data available online at `http://cvn.ecp.fr/projects/ranking-highorder`.

2 Preliminaries

2.1 Structured Output SVM

An SSVM, parameterized by $\mathbf{w}$, provides a linear prediction rule to obtain a structured output $\mathbf{y} \in \mathcal{Y}$ from a structured input $\mathbf{x} \in \mathcal{X}$. Formally, let $\Psi(\mathbf{x}, \mathbf{y})$ denote the joint feature vector of the input $\mathbf{x}$ and the output $\mathbf{y}$. The prediction for a given input $\mathbf{x}$ is obtained by maximizing the score over all possible outputs, that is, $\mathbf{y} = \text{argmax}_{\overline{\mathbf{y}} \in \mathcal{Y}} \mathbf{w}^\top \Psi(\mathbf{x}, \overline{\mathbf{y}})$.

Given a dataset that consists of n samples, that is, $\mathcal{D} = \{(\mathbf{x}_i, \mathbf{y}_i^*), i = 1, \cdots, n\}$, the parameters of an SSVM are estimated by minimizing a regularized upper bound on the empirical risk. The risk is measured using a user-specified loss function $\Delta(\cdot, \cdot)$. In more detail, the parameters are estimated by solving the following convex optimization problem:

$$\min_{\mathbf{w}} \frac{1}{2}||\mathbf{w}||^2 + \frac{C}{n}\sum_{i=1}^{n} \xi_i, \tag{1}$$
$$\mathbf{w}^\top \Psi(\mathbf{x}_i, \mathbf{y}_i^*) - \mathbf{w}^\top \Psi(\mathbf{x}_i, \overline{\mathbf{y}}) \geq \Delta(\mathbf{y}_i^*, \overline{\mathbf{y}}) - \xi_i, \forall \overline{\mathbf{y}} \in \mathcal{Y}, \forall i \in \{1, \cdots, n\}.$$

Intuitively, the above problem encourages a margin (proportional to $\Delta(\mathbf{y}_i^*, \overline{\mathbf{y}})$) between the score of the ground-truth output $\mathbf{y}_i^*$ and all other outputs $\overline{\mathbf{y}}$. The hyperparameter C controls the trade-off between the training error and the model complexity. Inspite very large number of constraints, it has been shown that the above problem can be optimized efficiently using cutting-plane algorithm [10] which requires iteratively solving the loss-augmented inference problem (to find the most-violated constraint), that is, $\hat{\mathbf{y}}_i = \text{argmax}_{\overline{\mathbf{y}}} \mathbf{w}^\top \Psi(\mathbf{x}_i, \overline{\mathbf{y}}) + \Delta(\mathbf{y}_i^*, \overline{\mathbf{y}})$.

2.2 AP-SVM

The AP-SVM classifier is a special case of SSVM. The input of an AP-SVM is a set of n samples, which we denote by $\mathbf{X} = \{\mathbf{x}_i, i = 1, \cdots, n\}$. Each sample can either belong to the positive class (that is, the sample is relevant) or the negative class (that is, the sample is not relevant). The indices for the positive and negative samples are denoted by $\mathcal{P}$ and $\mathcal{N}$ respectively. In other words, if $i \in \mathcal{P}$ and $j \in \mathcal{N}$ then $\mathbf{x}_i$ belongs to positive class and $\mathbf{x}_j$ belongs to the negative class.

The desired output is a ranking matrix $\mathbf{R}$ of size $n \times n$, such that (i) $\mathbf{R}_{ij} = 1$ if $\mathbf{x}_i$ is ranked higher than $\mathbf{x}_j$; (ii) $\mathbf{R}_{ij} = -1$ if $\mathbf{x}_i$ is ranked lower than $\mathbf{x}_j$; and (iii) $\mathbf{R}_{ij} = 0$ if $\mathbf{x}_i$ and $\mathbf{x}_j$ are assigned the same rank. During training, the ground-truth ranking matrix $\mathbf{R}^*$ is defined as: (i) $\mathbf{R}^*_{ij} = 1$ and $\mathbf{R}^*_{ji} = -1$ for all $i \in \mathcal{P}$ and $j \in \mathcal{N}$; (ii) $\mathbf{R}^*_{ii'} = 0$ and $\mathbf{R}^*_{jj'} = 0$ for all $i, i' \in \mathcal{P}$ and $j, j' \in \mathcal{N}$.

Joint Feature Vector. For a sample $\mathbf{x}_i$, let $\psi(\mathbf{x}_i)$ denote its feature vector. For example, in action classification, $\psi(\mathbf{x}_i)$ can represent poselet [2] or bag-of-visual-words [4]. Similar to [26], we specify a joint feature vector as

$$\Psi(\mathbf{X}, \mathbf{R}) = \gamma \sum_{i \in \mathcal{P}} \sum_{j \in \mathcal{N}} \mathbf{R}_{ij} (\psi(\mathbf{x}_i) - \psi(\mathbf{x}_j)), \gamma = \frac{1}{|\mathcal{P}||\mathcal{N}|} \tag{2}$$

In other words, the joint feature vector is the scaled sum of the difference between the features of all pairs of samples having different classes.

Parameters and Prediction. The parameter vector of the classifier is denoted by $\mathbf{w}$. Given the parameters $\mathbf{w}$, the ranking of an input $\mathbf{X}$ is predicted by maximizing the score, that is, $\mathbf{R} = \text{argmax}_{\overline{\mathbf{R}}} \mathbf{w}^\top \Psi(\mathbf{X}, \overline{\mathbf{R}})$. Yue *et al.* [26] showed that the above optimization can be performed efficiently by sorting the samples $\mathbf{x}_k$ in descending order of the score $\mathbf{w}^\top \psi(\mathbf{x}_k)$.

Loss Function. Given a training dataset, our aim is to learn a classifier that provides a high AP measure. Let $\text{AP}(\mathbf{R}^*, \mathbf{R})$ denote the AP of the ranking matrix $\mathbf{R}$ with respect to the ground truth ranking $\mathbf{R}^*$. The $\text{AP}(\mathbf{R}^*, \mathbf{R})$ is defined as: $\text{AP}(\mathbf{R}, \mathbf{R}^*) = \frac{1}{|\mathcal{P}|} \sum_k \textit{Prec}(k) \delta(\textit{Rec}(k))$, where $|\mathcal{P}|$ is the number of positive samples in the ground truth $\mathbf{R}^*$, $\textit{Prec}(k)$ is the precision upto top k samples given by $\mathbf{R}$, and $\delta(\textit{Rec}(k))$ is the change in recall when moving from $(k-1)^{th}$ to k^{th} sample. The value of the $\text{AP}(\cdot, \cdot)$ lies between 0 and 1, where 0 corresponds to a completely incorrect ranking $-\mathbf{R}^*$ and 1 corresponds to the correct ranking $\mathbf{R}^*$. In order to maximize the AP, we will minimize a loss function defined as $\Delta(\mathbf{R}^*, \mathbf{R}) = 1 - \text{AP}(\mathbf{R}^*, \mathbf{R})$.

Parameter Estimation. Given the input $\mathbf{X}$ and the ground-truth ranking matrix $\mathbf{R}^*$, we would like to learn the parameters of the classifier such that regularized upper bound on the empirical AP loss is minimized. Specifically, the model parameters are obtained by solving the following convex optimization problem:

$$\begin{aligned} &\min_{\mathbf{w}} \ \frac{1}{2}||\mathbf{w}||^2 + C\xi, \\ &\mathbf{w}^\top \Psi(\mathbf{X}, \mathbf{R}^*) - \mathbf{w}^\top \Psi(\mathbf{X}, \mathbf{R}) \geq \Delta(\mathbf{R}^*, \mathbf{R}) - \xi, \forall \mathbf{R} \end{aligned} \tag{3}$$

Problem (3) is specified over an exponential number of $\mathbf{R}$. Nonetheless, Yue *et al.* [26] showed that it can be optimized efficiently by providing an optimal greedy algorithm to solve the corresponding loss-augmented inference problem, that is, $\hat{\mathbf{R}} = \text{argmax}_{\overline{\mathbf{R}}} \mathbf{w}^\top \Psi(\mathbf{X}, \overline{\mathbf{R}}) + \Delta(\mathbf{R}^*, \overline{\mathbf{R}})$.

3 High-Order Binary SVM (HOB-SVM)

We now describe our two frameworks for ranking while incorporating high-order information. As mentioned earlier, we will restrict our description to second-order information. However, extending our frameworks to general high-order information is trivial. We start with the simpler framework, which we call *High-Order Binary* SVM (HOB-SVM). This will allow us to define the terminology necessary to develop a more principled framework (HOAP-SVM) in the next section.

The input of a HOB-SVM is a set of n samples $\mathbf{X} = \{\mathbf{x}_i, i = 1, \cdots, n\}$. Similar to the AP-SVM, a sample can either belong to the positive class or a negative class. However, the output of HOB-SVM is not a ranking, but an assignment of a class for each sample. In other words, the output is a vector $\mathbf{Y} = \{y_i, i = 1, \cdots, n\}$ where $y_i \in \{0, 1\}$. The label '0' implies that the sample has been assigned to the negative class, whereas the label '1' implies that the sample has been assigned to the positive class. During training, the ground-truth output $\mathbf{Y}^*$ assigns all relevant samples to the positive class and all non-relevant samples to the negative class. Once again, given $\mathbf{Y}^*$, we denote the indices of the positive and the negative samples as $\mathcal{P}$ and $\mathcal{N}$ respectively.

Joint Feature Vector. The joint feature vector of the input $\mathbf{X}$ and the output $\mathbf{Y}$ consists of two parts. The first part $\Psi_1(\mathbf{X}, \mathbf{Y})$ captures first-order information, and is henceforth referred to as the unary joint feature vector. The second part $\Psi_2(\mathbf{X}, \mathbf{Y})$ captures second-order information, and is henceforth referred to as the pairwise joint feature vector. In more detail, let $\psi(\mathbf{x}_i) \in \mathbb{R}^d$ denote the feature vector of the sample $\mathbf{x}_i$. The unary joint feature vector is defined as follows:

$$\Psi_1(\mathbf{X}, \mathbf{Y}) = \begin{pmatrix} \sum_{i, y_i = 1} \psi(\mathbf{x}_i) \\ \sum_{i, y_i = 0} \psi(\mathbf{x}_i) \end{pmatrix}. \tag{4}$$

The unary joint feature vector is of dimensionality $2d$. The first d dimensions correspond to the sum of the feature vectors of the samples belonging to the positive class. The last d dimensions correspond to the sum of the feature vectors of the samples belonging to the negative class. Clearly, $\Psi_1(\mathbf{X}, \mathbf{Y})$ only captures the first-order information.

As mentioned earlier, our aim is to use second-order information to improve ranking. In other words, if we know *a priori* that two samples $\mathbf{x}_i$ and $\mathbf{x}_j$ are more likely to belong to the same class (henceforth referred to as similar samples), then we would like to encourage them to either both be labeled as relevant or as non-relevant. Let $\mathcal{E}$ denote the set of all pairs of similar samples. In other words, if samples $\mathbf{x}_i$ and $\mathbf{x}_j$ are similar, then $(i, j) \in \mathcal{E}$. We define the pairwise joint feature vector as follows:

$$\Psi_2(\mathbf{X}, \mathbf{Y}) = \eta \left(\sum_{(i,j) \in \mathcal{E}, y_i \neq y_j} \Phi(\psi(\mathbf{x}_i), \psi(\mathbf{x}_j)) \right) \tag{5}$$

where $\Phi(\psi(\mathbf{x}_i), \psi(\mathbf{x}_j))$ is a vector such that each of its elements is inversely proportional to the difference between the corresponding elements of its two input vectors and η controls the trade-off between the first-order and high-order information. In our work, we define $\Phi(\mathbf{z}_i, \mathbf{z}_j) = \exp(-(\mathbf{z}_i - \mathbf{z}_j)^2)$. All the operations are performed in an element-wise manner. In other words, $\Psi_2(\mathbf{X}, \mathbf{Y})$ is a d dimensional vector that is the sum of pairwise feature vectors over all pairs of similar samples having different classes.

Parameters and Prediction. Similar to the joint feature vector, the parameters of a HOB-SVM consist of two parts: the unary parameters $\mathbf{w}_1 \in \mathbb{R}^{2d}$ and the pairwise parameters $\mathbf{w}_2 \in \mathbb{R}^d$. Given an input $\mathbf{X}$, the output $\mathbf{Y}$ is predicted by maximizing the score, that is,

$$\mathbf{Y} = \operatorname*{argmax}_{\mathbf{Y}} \mathbf{w}^\top \Psi(\mathbf{X}, \mathbf{Y}), \mathbf{w} = \begin{pmatrix} \mathbf{w}_1 \\ \mathbf{w}_2 \end{pmatrix}, \Psi(\mathbf{X}, \mathbf{Y}) = \begin{pmatrix} \Psi_1(\mathbf{X}, \mathbf{Y}) \\ \Psi_2(\mathbf{X}, \mathbf{Y}) \end{pmatrix}. \tag{6}$$

Note that, in general, the above problem is NP-hard. However, when $\mathbf{w}_2 \leq 0$, it can be optimized efficiently using graph cuts [14]. This follows from the fact that each element of the pairwise joint feature vector is non-negative (see equation (5)), and hence the score of an output $\mathbf{Y}$ is a supermodular function of $\mathbf{Y}$. In what follows, we will always estimate the parameters of a HOB-SVM under the constraint that $\mathbf{w}_2 \leq 0$. Moreover, our approaches can also be used without this constraint by employing approximate inference algorithms such as [13, 16, 20].

Loss Function. Although we would ideally like to optimize the AP loss, as mentioned earlier, this results in a difficult loss-augmented inference problem when the joint feature vector captures high-order information. Hence, inspired by the success of SVM for ranking, we use a surrogate loss function defined as follows:

$$\Delta(\mathbf{Y}^*, \mathbf{Y}) = \frac{J \sum_{i, y_i^*=1} \delta(y_i = 0) + \sum_{j, y_j^*=0} \delta(y_j = 1)}{J|\mathcal{P}| + |\mathcal{N}|}, \tag{7}$$

where $\delta(\cdot)$ is 1 if its argument is true and 0 otherwise. The terms $|\mathcal{P}|$ and $|\mathcal{N}|$ are the total number of positive and negative samples (as specified by the ground-truth assignment $\mathbf{Y}^*$) respectively. The hyperparameter J is set to $|\mathcal{N}|/|\mathcal{P}|$. In other words, $\Delta(\mathbf{Y}^*, \mathbf{Y})$ is the weighted fraction of misclassifications.

Parameter Estimation. Given the dataset $(\mathbf{X}, \mathbf{Y}^*)$, the parameters of HOB-SVM are obtained by solving the following convex optimization problem:

$$\begin{aligned} &\min_{\mathbf{w}} \ \frac{1}{2}||\mathbf{w}||^2 + C\xi, \\ &\quad \mathbf{w}^\top \Psi(\mathbf{X}, \mathbf{Y}) - \mathbf{w}^\top \Psi(\mathbf{X}, \overline{\mathbf{Y}}) \geq \Delta(\mathbf{Y}^*, \overline{\mathbf{Y}}) - \xi, \forall \overline{\mathbf{Y}}, \mathbf{w}_2 \leq 0. \end{aligned} \tag{8}$$

Even though the number of constraints in the above problem are exponential in the number of samples n, it can be optimized efficiently by iteratively solving the loss-augmented inference problem, that is, $\hat{\mathbf{Y}} = \operatorname{argmax}_{\overline{\mathbf{Y}}}(\mathbf{w}^\top \Psi(\mathbf{X}, \overline{\mathbf{Y}}) + \Delta(\mathbf{Y}^*, \overline{\mathbf{Y}}))$. The restriction $\mathbf{w}_2 \leq 0$ allows us to solve the above problem efficiently using graph cuts [14]. The problem (8) is similar to training graphical models with approximate inference [6, 7, 17, 23].

Using HOB-SVM for Ranking. From a theoretical point of view, the main disadvantage of HOB-SVM is that it optimizes a surrogate loss function instead of the AP loss. In the next section, we will describe a novel framework that addresses this disadvantage. From a practical point of view, the main disadvantage of HOB-SVM is that it provides a single score for the entire assignment $\mathbf{Y}$. In other words, instead of assigning an individual score for each sample, it assigns one score $\mathbf{w}^\top \Psi(\mathbf{X}, \mathbf{Y})$ for all the samples taken together. This prevents us from specifying a ranking of the samples. To address this issue, we propose a simple yet intuitive solution: (i) compute the difference between the max-marginal of a sample being assigned to the positive class and the max-marginal of it being assigned to the negative class; and (ii) sort the samples according to the difference in max-marginals. Max-marginal captures high-order information and the difference of max-marginals measures our confidence on a particular sample belonging to the positive class. Formally, we define the max-marginal of a sample $\mathbf{x}_i$ belonging to the positive class $m_i^+(\mathbf{w})$ and negative class $m_i^-(\mathbf{w})$ as:

$$m_i^+(\mathbf{w}) = \mathbf{w}^\top \Psi(\mathbf{X}, \mathbf{Y}_i^+), \mathbf{Y}_i^+ = \underset{\mathbf{Y}, y_i = 1}{\operatorname{argmax}} \mathbf{w}^\top \Psi(\mathbf{X}, \mathbf{Y}). \tag{9}$$

$$m_i^-(\mathbf{w}) = \mathbf{w}^\top \Psi(\mathbf{X}, \mathbf{Y}_i^-), \mathbf{Y}_i^- = \underset{\mathbf{Y}, y_i = 0}{\operatorname{argmax}} \mathbf{w}^\top \Psi(\mathbf{X}, \mathbf{Y}). \tag{10}$$

The max-marginals for all the samples can be computed efficiently using the *dynamic graph cuts* algorithm [11, 12]. Given the max-marginals $m_i^+(\mathbf{w})$ and $m_i^-(\mathbf{w})$, the score of a sample $\mathbf{x}_i$ is defined as

$$s_i(\mathbf{w}) = m_i^+(\mathbf{w}) - m_i^-(\mathbf{w}). \tag{11}$$

Note that, if the two labelings $\mathbf{Y}_i^+$ and $\mathbf{Y}_i^-$ defined in equations (9)-(10) respectively differ only in the label assigned to the sample $\mathbf{x}_i$, this implies that the sample $\mathbf{x}_i$ has no influence in determining the labels of the other samples in the dataset. In this case, the difference in max-marginals does not depend on the feature vectors of any other samples except the sample $\mathbf{x}_i$. However, if the sample $\mathbf{x}_i$ does influence the labels of the other samples (that is, $\mathbf{Y}_i^+$ and $\mathbf{Y}_i^-$ differ significantly), then the difference in the max-marginals depends on several samples in the dataset. The ranking is obtained by sorting the samples in descending order of their scores $s_i(\mathbf{w})$. As will be seen in section 5, this intuitive way of scoring a sample provides an improved ranking over the baselines.

4 High-Order Average Precision SVM (HOAP-SVM)

While HOB-SVM allows us to incorporate high-order information via the pairwise joint feature vector, it suffers from the deficiency of using a surrogate loss function. Specifically, instead of optimizing the AP loss in order to estimate the parameters, it optimizes a weighted 0-1 loss. However, the way that HOB-SVM obtains a ranking points us to the direction of resolving this deficiency. We begin by presenting the high-level overview of our approach. We observe that the score of a ranking according to an AP-SVM is the weighted sum of the scores of the individual samples. The reason why AP-SVM fails to capture high-order information is that the score of the individual sample depends on no other sample in the dataset. This is in contrast to the score employed by HOB-SVM (see equation (11)). Hence, it would be desirable to extend AP-SVM such that the score of the ranking is the weighted sum of the difference of max-marginals of individual samples. This is precisely our next learning framework, which we call *High-Order* AP-SVM (HOAP-SVM). In what follows, we describe HOAP-SVM in detail.

The input of HOAP-SVM is a set of n samples $\mathbf{X} = \{\mathbf{x}_i, i = 1, \cdots, n\}$. Similar to AP-SVM, a sample can belong to the positive class or the negative class. The output of HOAP-SVM is a ranking matrix $\mathbf{R}$, defined in a similar manner to AP-SVM. During training, the ground-truth ranking matrix $\mathbf{R}^*$ assigns each positive sample to a higher rank than all negative samples. Once again, the indices of positive and negative samples is represented as $\mathcal{P}$ and $\mathcal{N}$ respectively.

Score of a Ranking. The parameters of HOAP-SVM are denoted by $\mathbf{w}$. Given an input $\mathbf{X}$ and a ranking $\mathbf{R}$, the score for the ranking specified by HOAP-SVM is defined as follows:

$$S(\mathbf{X}, \mathbf{R}; \mathbf{w}) = \gamma \sum_{i \in \mathcal{P}} \sum_{j \in \mathcal{N}} \mathbf{R}_{ij} (s_i(\mathbf{w}) - s_j(\mathbf{w})), \tag{12}$$

where $s_i(\mathbf{w})$ is as specified in equation (11). In other words, the score of a ranking is the weighted sum of the difference of max-marginals for each sample, where the weights are specified by the ranking.

Prediction. Given an input $\mathbf{X}$, the ranking $\mathbf{R}$ is predicted by maximizing the score over all possible rankings, that is,

$$\mathbf{R} = \operatorname*{argmax}_{\overline{\mathbf{R}}} S(\mathbf{X}, \mathbf{R}; \mathbf{w}). \tag{13}$$

Proposition 1. *Problem (13) can be solved efficiently by sorting the samples in descending order of their scores $s_i(\mathbf{w})$ (proof given in Appendix A.1).*

In other words, the prediction for HOAP-SVM is the same as the prediction for HOB-SVM. Recall that the score $s_i(\mathbf{w})$ can be computed efficiently using dynamic graph cuts [11, 12].

Parameter Estimation. Given the input $\mathbf{X}$ and the ground-truth ranking $\mathbf{R}^*$, the parameters of HOAP-SVM are learned by optimizing the AP loss. To this end, we propose to estimate $\mathbf{w}$ by solving the following optimization problem:

$$\begin{aligned} \min_{\mathbf{w}} \;\; & \frac{1}{2}||\mathbf{w}||^2 + C\xi, \\ & S(\mathbf{X}, \mathbf{R}^*; \mathbf{w}) - S(\mathbf{X}, \mathbf{R}; \mathbf{w}) \geq \Delta(\mathbf{R}^*, \mathbf{R}) - \xi, \forall \mathbf{R}, \mathbf{w}_2 \leq 0. \end{aligned} \tag{14}$$

Here, $\Delta(\mathbf{R}^*, \mathbf{R})$ is the AP loss, that is, one minus the AP of the ranking $\mathbf{R}$ with respect to $\mathbf{R}^*$. The following proposition establishes the suitability of the above problem for learning an HOAP-SVM.

Proposition 2. *Problem (14) minimizes a regularized upper bound on the* AP *loss of the predicted ranking (proof given in Appendix A.2).*

Algorithm 1 The CCCP algorithm for learning HOAP-SVM parameters.

```
input Samples $\mathbf{X}$, ranking $\mathbf{R}^*$, tolerance $\epsilon$, initial parameters $\mathbf{w}_0$.
$t \leftarrow 0$.
repeat
    For all $\mathbf{R}$, find a linear lower bound $l(\mathbf{w};\mathbf{R})$ tight at $\mathbf{w}_t$ using proposition 4.
    Update the parameters by solving the following convex optimization problem:
        $\mathbf{w}_{t+1} = \operatorname*{argmin}_{\mathbf{w}} \frac{1}{2}||\mathbf{w}||^2 + C\xi,$    (17)
        $\xi \geq \Delta(\mathbf{R}^*,\mathbf{R}) + f(\mathbf{w};\mathbf{R}) - l(\mathbf{w};\mathbf{R}), \forall \mathbf{R}.$
    The loss-augmented inference can be solved efficiently using proposition 5.
    $t \leftarrow t + 1$.
until Objective of problem (25) does not decrease more than $\epsilon$.
```

Optimization. While problem (14) provides a valid upper bound on the AP loss, it is not a convex program. Hence, it cannot be optimized efficiently to obtain an optimal set of parameters for HOAP-SVM. However, in what follows, we show that problem (14) is a difference-of-convex program. By identifying the convex and the concave part of problem (14), we show how a locally optimal set of parameters can be obtained efficiently using the concave-convex procedure (CCCP) [27].

We begin by specifying the following shorthand notation that will be useful in simplifying problem (14). Given a ranking $\mathbf{R}$ we define functions $f(\mathbf{w};\mathbf{R})$ and $g(\mathbf{w};\mathbf{R})$ of the parameters $\mathbf{w}$ as

$$f(\mathbf{w};\mathbf{R}) = \gamma\sum_{i\in\mathcal{P}} m_i^-(\mathbf{w})\left(\sum_{j\in\mathcal{N}}(\mathbf{R}^*_{ij} - \mathbf{R}_{ij})\right) + \gamma\sum_{j\in\mathcal{N}} m_j^+(\mathbf{w})\left(\sum_{i\in\mathcal{P}}(\mathbf{R}^*_{ij} - \mathbf{R}_{ij})\right),$$

$$g(\mathbf{w};\mathbf{R}) = \gamma\sum_{i\in\mathcal{P}} m_i^+(\mathbf{w})\left(\sum_{j\in\mathcal{N}}(\mathbf{R}^*_{ij} - \mathbf{R}_{ij})\right) + \gamma\sum_{j\in\mathcal{N}} m_j^-(\mathbf{w})\left(\sum_{i\in\mathcal{P}}(\mathbf{R}^*_{ij} - \mathbf{R}_{ij})\right) \tag{15}$$

Proposition 3. *For any valid ranking matrix* $\mathbf{R}$*, the functions* $f(\mathbf{w};\mathbf{R})$ *and* $g(\mathbf{w};\mathbf{R})$ *are convex in* $\mathbf{w}$ *(proof given in Appendix A.3).*

Using our shorthand notation problem (14) can be rewritten as follows:

$$\min_{\mathbf{w}} \frac{1}{2}||\mathbf{w}||^2 + C\xi, \tag{16}$$
$$\xi \geq \Delta(\mathbf{R}^*,\mathbf{R}) + f(\mathbf{w};\mathbf{R}) - g(\mathbf{w};\mathbf{R}), \forall \mathbf{R}.$$

The above problem is obtained by substituting the value of the score of the ranking, defined in equation (12), into problem (14). Using proposition 3, it follows that problem (25) is a difference-of-convex program. This allows us to obtain a locally optimal set of parameters for the HOAP-SVM formulation using the CCCP approach outlined in Algorithm 1. The CCCP algorithm consists of two steps. In the first step, given the current set of parameters $\mathbf{w}_t$, we obtain a linear approximation $l(\mathbf{w};\mathbf{R})$ of the function $g(\mathbf{w};\mathbf{R})$ such that $l(\mathbf{w}_t;\mathbf{R}) = g(\mathbf{w}_t;\mathbf{R}), l(\mathbf{w};\mathbf{R}) \leq g(\mathbf{w};\mathbf{R}), \forall \mathbf{w}$. In other words, the linear function $l(\mathbf{w};\mathbf{R})$ is a lower bound on the function $g(\mathbf{w};\mathbf{R})$ such that the lower bound is tight at the current parameters $\mathbf{w}_t$. While at first sight the problem of obtaining the linear approximation for each ranking matrix $\mathbf{R}$ may appear to be highly expensive, the following proposition shows how this step can be performed in a computationally efficient manner.

Proposition 4. *Given the current set of parameters $\mathbf{w}_t$, let*

$$\bar{\mathbf{Y}}_i^+ = \operatorname*{argmax}_{\mathbf{Y}, y_i = 1} \mathbf{w}_t^\top \Psi(\mathbf{X}, \mathbf{Y}), \bar{\mathbf{Y}}_j^- = \operatorname*{argmax}_{\mathbf{Y}, y_j = -1} \mathbf{w}_t^\top \Psi(\mathbf{X}, \mathbf{Y}). \tag{18}$$

The following linear function is a lower bound on $g(\mathbf{w};\mathbf{R})$ that is tight at $\mathbf{w}_t$:

$$l(\mathbf{w};\mathbf{R}) = \gamma \sum_{i \in \mathcal{P}} \mathbf{w}^\top \Psi(\mathbf{X}, \bar{\mathbf{Y}}_i^+) \left(\sum_{j \in \mathcal{N}} (\mathbf{R}_{ij}^* - \mathbf{R}_{ij}) \right) + \gamma \sum_{j \in \mathcal{N}} \mathbf{w}^\top \Psi(\mathbf{X}, \bar{\mathbf{Y}}_j^-) \left(\sum_{i \in \mathcal{P}} (\mathbf{R}_{ij}^* - \mathbf{R}_{ij}) \right)$$

(proof given in Appendix A.4)

An upshot of the above proposition is that the linear lower bound of $g(\mathbf{w};\mathbf{R})$ can be computed efficiently for any $\mathbf{R}$ by pre-computing the labelings $\bar{\mathbf{Y}}_i^+$ and $\bar{\mathbf{Y}}_j^-$, which are independent of $\mathbf{R}$. The labelings $\bar{\mathbf{Y}}_i^+$ and $\bar{\mathbf{Y}}_j^-$ can be obtained efficiently using dynamic graph cuts [11, 12].

In the second step of CCCP, we obtain a convex optimization problem by substituting the linear approximation $l(\mathbf{w};\mathbf{R})$ in place of the convex function $g(\mathbf{w};\mathbf{R})$. We update the parameters by solving the resulting convex optimization problem. To this end, we use the cutting-plane algorithm [10] in order to handle exponentially many constraints. Cutting-plane algorithm requires us to iteratively find the most-violated ranking $\hat{\mathbf{R}}$. The following proposition makes the cutting-plane algorithm efficient.

Proposition 5. *Given the upperbounded scores $\bar{m}_i^+(\mathbf{w}) = \mathbf{w}^\top \Psi(\mathbf{X}, \bar{\mathbf{Y}}_i^+)$, $\bar{m}_j^-(\mathbf{w}) = \mathbf{w}^\top \Psi(\mathbf{X}, \bar{\mathbf{Y}}_j^-)$, and the scores for the current parameters $m_i^+(\mathbf{w}) = \mathbf{w}^\top \Psi(\mathbf{X}, \mathbf{Y}_i^+)$, $m_j^-(\mathbf{w}) = \mathbf{w}^\top \Psi(\mathbf{X}, \mathbf{Y}_j^-)$, the following problem gives the most violated ranking.*

$$\hat{\mathbf{R}} \leftarrow \operatorname*{argmax}_{\mathbf{R}} \left\{ \eta \sum_{i \in \mathcal{P}, j \in \mathcal{N}} \mathbf{R}_{ij} (\bar{s}_i(\mathbf{w}) - \bar{s}_j(\mathbf{w})) + \Delta(\mathbf{R}, \mathbf{R}^*) - \eta \sum_{i \in \mathcal{P}, j \in \mathcal{N}} \mathbf{R}_{ij}^* (\bar{s}_i(\mathbf{w}) - \bar{s}_j(\mathbf{w})) \right\}$$

where, $\bar{s}_i(\mathbf{w}) = (\bar{m}_i^+(\mathbf{w}) - m_i^-(\mathbf{w}))$ and $\bar{s}_j(\mathbf{w}) = (m_j^+(\mathbf{w}) - \bar{m}_j^-(\mathbf{w}))$. The greedy algorithm of [26] can be used to find the $\hat{\mathbf{R}}$ efficiently (proof given in Appendix A.5).

Upon convergence, the CCCP algorithm provides a locally optimal set of parameters for the HOAP-SVM framework.

Table 1. *The AP over five folds for the best setting of the hyperparameters obtained using the cross-validation. Our frameworks outperforms* SVM *and* AP-SVM *in all the 10 action classes. Note that* HOAP-SVM *is initialized with* HOB-SVM.

Actions/ Methods	Jump	Phone	Play inst	Read	Ride bike	Run	Take photo	Use comp	Walk	Ride horse	Average
SVM	56.0	35.5	42.6	33.8	81.9	78.4	33.9	37.2	61.7	85.9	54.7
AP-SVM	57.5	34.4	46.3	35.5	83.0	79.3	33.3	42.7	63.1	86.6	56.2
HOB-SVM	60.9	**36.1**	48.1	35.7	84.1	**81.5**	35.1	45.8	63.0	**87.9**	57.8
HOAP-SVM	**63.4**	34.5	**48.8**	**38.3**	**84.3**	81.0	**36.5**	**48.7**	**65.3**	87.7	**58.9**

Table 2. *The AP of all the four methods. The training is performed over the entire 'trainval' dataset of PASCAL VOC 2011 using the best hyperparameters obtained during 5-fold cross-validation. The testing is performed on the 'test' dataset and evaluated on the PASCAL VOC server. Note that* HOAP-SVM *is initialized using* HOB-SVM.

Actions/ Methods	Jump	Phone	Play inst	Read	Ride bike	Run	Take photo	Use comp	Walk	Ride horse	Average
SVM	51.1	29.7	40.5	20.6	81.1	76.7	20.0	27.7	56.7	84.2	48.82
AP-SVM	54.0	**33.8**	42.3	26.5	82.5	76.7	23.7	32.8	**57.7**	84.2	51.42
HOB-SVM	56.3	**33.8**	42.8	24.3	82.5	80.5	**27.7**	32.8	53.6	84.5	51.88
HOAP-SVM	**59.5**	**33.8**	**47.5**	**27.2**	**84.0**	**82.6**	26.1	**36.4**	55.1	**85.3**	**53.75**

5 Experiments

We now demonstrate the efficacy of our learning frameworks on the challenging problem of action classification [4, 18]. The input for action classification is an action class such as 'jumping' or 'running' and a set of samples $\mathbf{X} = \{\mathbf{x}_i = (\mathbf{I}_i, \mathbf{b}_i), i = 1, \cdots, n\}$. Here, $\mathbf{I}_i$ is the image corresponding to the i-th sample, and $\mathbf{b}_i$ is a tight bounding box around a person present in the image. The desired output is a ranking of the samples according to their relevance to the action. Recall that our main hypothesis is that high-order information can help improve the ranking accuracy. To test our hypothesis, we require a set of similar samples such that samples $\mathbf{x}_i$ and $\mathbf{x}_j$ are more likely to belong to the same class (relevant or non-relevant) if $(i, j) \in \mathcal{E}$. *In the action classification experiments, we define* $\mathcal{E} = \{(i, j), \mathbf{I}_i = \mathbf{I}_j\}$*, that is, the set of all pairs of bounding boxes that are present in the same image.* Note that one could use any other similarity criterion in the proposed frameworks. Below we describe our experimental setup in detail.

Dataset. We use PASCAL VOC 2011 [5] action classification dataset, which consists of 4846 images depicting 10 action classes. The dataset is divided into two subsets: 2424 'trainval' images for which we are provided the bounding boxes of the person in the image together with their action classes; and 2422 'test' images for which we are only provided with the person bounding boxes.

Features. Given a sample $\mathbf{x}_i = (\mathbf{I}_i, \mathbf{b}_i)$, we use the concatenation of standard poselet-based feature vector [2] of the bounding box $\mathbf{b}_i$ and GIST feature vector [19] of the image $\mathbf{I}_i$ to specify the sample features $\psi(\mathbf{x}_i)$. The poselet feature consists of 2400 activation scores of action-specific poselets and 4 object activation scores. The GIST feature is a 512 dimensional feature vector that captures the overall scene depicted in the image. This results in a sample feature of size 2916. The sample features used to specify the unary and the pairwise joint feature vectors are shown in equations (4) and (5). As each pair of similar samples comes from the same image, we defined the joint pairwise feature vector using only the poselet features. The size of the joint feature vector is therefore 8748.

Methods. We compare our proposed approaches, namely HOB-SVM and HOAP-SVM, with the standard binary SVM (obtained by setting $\mathbf{w}_2 = 0$ in HOB-SVM) and AP-SVM (obtained by setting $\mathbf{w}_2 = 0$ in HOAP-SVM) that ignore high-order information. The baselines, SVM and AP-SVM requires one hyperparameter C, and HOB-SVM and HOAP-SVM requires two hyperparameters C and η. The common hyperparameter C is the trade-off between the regularization and the empirical loss, and η is the trade-off between the first order information and the high-order information. Note that the 'test' dataset was not used for cross-validation. We obtained the best setting of the hyperparameters for each method independently via a 5-fold cross-validation on the entire 'trainval' dataset. We consider the following putative values: $C \in \{10^{-1}, 10^0, \dots, 10^4\}$ and $\eta \in \{10^{-4}, 10^0, \dots, 10^4\}$. The J parameter in (7) is fixed to $|\mathcal{N}|/|\mathcal{P}|$.

Results. Table 1 shows the average AP over all the five folds for the best hyperparameter setting. By incorporating high-order information HOB-SVM provides an improvement in the ranking compared to the commonly used SVM classifier for all 10 action classes. Furthermore, even though HOB-SVM employs a surrogate loss function, it provides more accurate rankings compared to AP-SVM for 9 action classes. By optimizing the AP loss function, while incorpoating high-order information, HOAP-SVM outperforms SVM in 9 action classes, AP-SVM in all 10 action classes, and HOB-SVM in 7 action classes. Table 2 shows the AP values obtained for the 'test' set when the methods are trained using the best hyperparameter setting over the entire 'trainval' set. Table 2 clearly shows that HOB-SVM outperforms SVM classifier in 9 action classes and AP-SVM in 5 along with 3 ties. On the other hand, HOAP-SVM outperforms SVM classifier in all the 10 classes, AP-SVM in 8 along with 1 tie, and HOB-SVM in 8 along with 1 tie.

The paired t-test shows that: (a) HOB-SVM is statistically better than SVM for 6 action classes, (b) HOB-SVM is not statistically better than AP-SVM, (c) HOAP-SVM is statistically better than SVM for 6 action classes, and (d) HOAP-SVM is statistically better than AP-SVM for 4 action classes.

The effects of incorporating high-order information is illustrated in Fig. 1. While high-order information can introduce errors in the ranking, in general it provides boost in the overall performance.

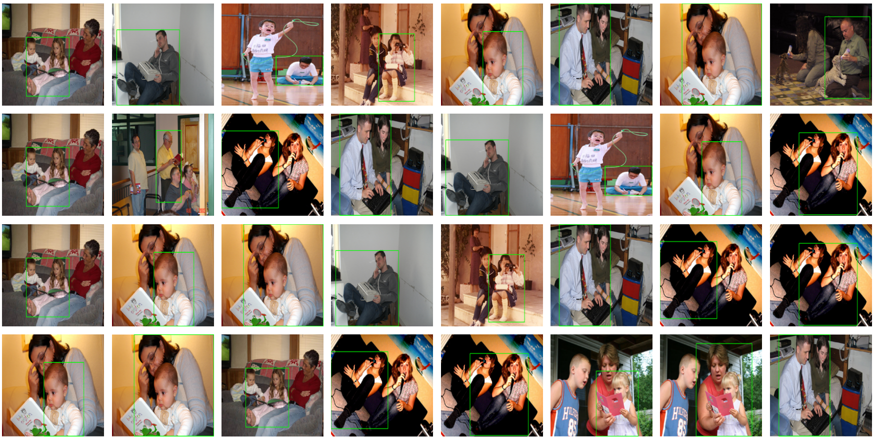

Fig. 1. *Top 8 samples ranked by all the four methods for 'reading' action class. First row –* SVM*, Second row –* AP-SVM*, Third row –* HOB-SVM*, and Fourth row –* HOAP-SVM*. Note that, the first false positive is ranked 2nd in case of* SVM *(first row) and 3rd in case of* AP-SVM *(second row), this shows the importance of optimizing the* AP *loss. On the other hand, in case of* HOB-SVM *(third row), the first false positive is ranked 4th and the 'similar samples' (2nd and 3rd) are assigned similar scores, this illustrates the importance of using high-order information. Furthermore,* HOAP-SVM *(fourth row) has the best* AP *among all the four methods, this shows the importance of using high-order information and optimizing the correct loss. Note that, in case of* HOAP-SVM*, the 4th and 5th ranked samples are false positives (underlying action is close to reading) and they both belong to the same image (our similarity criterion). This indicates that high-order information sometimes may lead to poor test* AP *in case of confusing classes (such as 'playinginstrument' vs 'usingcomputer') by assigning all the connected samples to the wrong label. Same effect can be seen in* HOB-SVM *for 7th and 8th ranked samples.*

6 Discussion

We proposed two new learning frameworks that incorporate high-order information to improve the accuracy of ranking. The first framework, HOB-SVM, uses a surrogate loss function, which allows us to compute its parameters by solving a convex optimization problem. The second framework, HOAP-SVM, minimizes the AP loss, which results in a difference-of-convex optimization problem. Both HOB-SVM and HOAP-SVM outperform baseline methods that do not make use of high-order information. By minimizing the correct loss function, HOAP-SVM outperforms HOB-SVM. An interesting direction for future work would be to allow for weakly supervised learning by extending the recently proposed *latent* AP-SVM [1] formulation to use high-order information. While such a learning formulation can be easily obtained with the introduction of latent variables, it is not clear whether the resulting optimization problem can be solved efficiently.

7 Acknowledgements

This work is partially funded by the European Research Council under the European Community's Seventh Framework Programme (FP7/2007-2013)/ERC Grant agreement number 259112, and the Ministere de l'education nationale, de l'enseignement superieure et de la recherche.

A Proofs

A.1 Proposition 1

Proof. Problem (13) can be solved by sorting the samples in descending order of their scores $s_i(\mathbf{w})$. Predicting ranking is basically picking each $\mathbf{R}_{ij}$ to maximize $S(\mathbf{X}, \mathbf{R}; \mathbf{w})$. Since $\mathbf{R}_{ij} = sign(s_i(\mathbf{w}) - s_j(\mathbf{w}))$, therefore, maximizing $S(\mathbf{X}, \mathbf{R}; \mathbf{w})$ w.r.t. $\mathbf{R}$ is basically reshuffling the scores such that $sign(s_i(\mathbf{w}) - s_j(\mathbf{w})) > 0$, which can be achieved by sorting the scores. Refer to [9] for more details.

A.2 Proposition 2

Proof. Note that, for a given $\mathbf{w}$ and predicted ranking $\mathbf{R}(\mathbf{w})$ (see equation 13), the AP loss can be upperbounded as follows:

$$\begin{aligned} \mathit{\Delta}(\mathbf{R}(\mathbf{w}), \mathbf{R}^*) &= \mathit{\Delta}(\mathbf{R}(\mathbf{w}), \mathbf{R}^*) + S(\mathbf{X}, \mathbf{R}(\mathbf{w}); \mathbf{w}) - S(\mathbf{X}, \mathbf{R}(\mathbf{w}); \mathbf{w}) \quad (19) \\ &\leq \mathit{\Delta}(\mathbf{R}(\mathbf{w}), \mathbf{R}^*) + S(\mathbf{X}, \mathbf{R}(\mathbf{w}); \mathbf{w}) - S(\mathbf{X}, \mathbf{R}^*; \mathbf{w}) \quad (20) \\ &\leq \max_R (\mathit{\Delta}(\mathbf{R}, \mathbf{R}^*) + S(\mathbf{X}, \mathbf{R}; \mathbf{w})) - S(\mathbf{X}, \mathbf{R}^*; \mathbf{w}) \quad (21) \end{aligned}$$

In expression (19) we add and subtract the same term, therefore, it is valid. Expression (20) follows from the fact that the score for the ground truth ranking $\mathbf{R}^*$ must be less than or equal to the score corresponding to the predicted ranking $\mathbf{R}(\mathbf{w})$ (see equation (13) for clarification). Expression (21) follows from the fact that the first two terms of the expression (21) are always greater than or equal to the first two terms of the expression (20). Using this upperbound on the AP loss (21), the regularized objective function of HOAP-SVM can be written as:

$$\min_{\mathbf{w}} \ \frac{1}{2}||\mathbf{w}||^2 + C\{\max_R (S(\mathbf{X}, \mathbf{R}; \mathbf{w}) + \mathit{\Delta}(\mathbf{R}^*, \mathbf{R})) - S(\mathbf{X}, \mathbf{R}; \mathbf{w})\}, \mathbf{w}_2 \leq 0. (22)$$

Note that the objective function (14) is the constrained form of the objective function (22), therefore, they represent exactly the same optimization problem.

A.3 Proposition 3

Proof. In order to prove that HOAP-SVM optimization problem is a difference of convex functions, let's substitute different terms (e.g. $m_i^+(\mathbf{w})$, $m_j^-(\mathbf{w})$) in the objective function (14):

$$\begin{aligned} \min_{\mathbf{w}} \quad & \frac{1}{2}||\mathbf{w}||^2 + C\xi, \\ s.t. \quad & \frac{1}{|\mathcal{P}||\mathcal{N}|} \sum_{i \in \mathcal{P}} \sum_{j \in \mathcal{N}} \mathbf{R}^*_{ij} \{(m_i^+(\mathbf{w}) - m_i^-(\mathbf{w})) - (m_j^+(\mathbf{w}) - m_j^-(\mathbf{w}))\} \\ & - \frac{1}{|\mathcal{P}||\mathcal{N}|} \sum_{i \in \mathcal{P}} \sum_{j \in \mathcal{N}} \mathbf{R}_{ij} \{(m_i^+(\mathbf{w}) - m_i^-(\mathbf{w})) - (m_j^+(\mathbf{w}) - m_j^-(\mathbf{w}))\} \\ & \geq \mathit{\Delta}(\mathbf{R}, \mathbf{R}^*) - \xi, \ \xi \geq 0, \forall R. \end{aligned} \quad (23)$$

Note that the objective function of the HOAP-SVM optimization problem (23) is convex but the constraints are non-convex functions. Each max-marginal, for example, $\max_{\mathbf{Y},y_i=+1} \mathbf{w}^\top \Psi(\mathbf{x},\mathbf{y})$, is a max over linear functions, therefore, is a convex function. Each constraint is the summation and substraction over the max-marginals, therefore, are non-convex. Below we prove that the constraints of (23) represents a very specific family of non-convex functions known as the difference of convex functions [8, 27].

The rearrangement of the different terms in (23) leads to following form:

$$\begin{aligned}
\min_{\mathbf{w}} \quad & \frac{1}{2}||\mathbf{w}||^2 + C\xi, \\
\xi \geq & \frac{1}{|\mathcal{P}||\mathcal{N}|}\sum_{i\in\mathcal{P}}\sum_{j\in\mathcal{N}}(\mathbf{R}^*_{ij} - \mathbf{R}_{ij})(m^-_i(\mathbf{w}) + m^+_j(\mathbf{w})) + \Delta(\mathbf{R},\mathbf{R}^*) \\
& - \frac{1}{|\mathcal{P}||\mathcal{N}|}\sum_{i\in\mathcal{P}}\sum_{j\in\mathcal{N}}(\mathbf{R}^*_{ij} - \mathbf{R}_{ij})(m^+_i(\mathbf{w}) + m^-_j(\mathbf{w})), \forall R. \qquad (24)
\end{aligned}$$

Objective function (24) can be further written as:

$$\begin{aligned}
\min_{\mathbf{w}} \; & \frac{1}{2}||\mathbf{w}||^2 + C\xi, \qquad (25) \\
& \xi \geq \Delta(\mathbf{R}^*,\mathbf{R}) + f(\mathbf{w};\mathbf{R}) - g(\mathbf{w};\mathbf{R}), \forall \mathbf{R}.
\end{aligned}$$

where,

$$\begin{aligned}
f(\mathbf{w};\mathbf{R}) = & \frac{1}{|\mathcal{P}||\mathcal{N}|}\sum_{i\in\mathcal{P}} m^-_i(\mathbf{w}) \left(\sum_{j\in\mathcal{N}}(\mathbf{R}^*_{ij} - \mathbf{R}_{ij})\right) \\
& + \frac{1}{|\mathcal{P}||\mathcal{N}|}\sum_{j\in\mathcal{N}} m^+_j(\mathbf{w}) \left(\sum_{i\in\mathcal{P}}(\mathbf{R}^*_{ij} - \mathbf{R}_{ij})\right), \qquad (26) \\
g(\mathbf{w};\mathbf{R}) = & \frac{1}{|\mathcal{P}||\mathcal{N}|}\sum_{i\in\mathcal{P}} m^+_i(\mathbf{w}) \left(\sum_{j\in\mathcal{N}}(\mathbf{R}^*_{ij} - \mathbf{R}_{ij})\right) \\
& + \frac{1}{|\mathcal{P}||\mathcal{N}|}\sum_{j\in\mathcal{N}} m^-_j(\mathbf{w}) \left(\sum_{i\in\mathcal{P}}(\mathbf{R}^*_{ij} - \mathbf{R}_{ij})\right). \qquad (27)
\end{aligned}$$

Since max-marginals in itself are convex and the summation of convex functions with positive coefficients is also convex, therefore, the convexity or concavity of the terms in (26) and (27) depends on the sign of $\sum_i(\mathbf{R}^*_{ij} - \mathbf{R}_{ij})$ and $\sum_j(\mathbf{R}^*_{ij} - \mathbf{R}_{ij})$. We can observe that for the fixed values of i, $0 \leq \sum_j \mathbf{R}_{ij} \leq |\mathcal{N}|$, similarly, $0 \leq \sum_i \mathbf{R}_{ij} \leq |\mathcal{P}|$. Also, since $\mathbf{R}^*$ is the true ranking matrix, therefore, $\sum_j \mathbf{R}^*_{ij} = |\mathcal{N}|$ and $\sum_i \mathbf{R}^*_{ij} = |\mathcal{P}|$. Hence, we can conclude that $\sum_j(\mathbf{R}^*_{ij} - \mathbf{R}_{ij}) \geq 0$ and $\sum_i(\mathbf{R}^*_{ij} - \mathbf{R}_{ij}) \geq 0$. Therefore, for given $\mathbf{R}$, the constraints of the optimization problem (25) clearly represents a difference-of-convex functions with $f(\mathbf{w};\mathbf{R})$ and $-g(\mathbf{w};\mathbf{R})$ as the the convex and concave functions respectively.

A.4 Proposition 4

Proof. Upperbounding the concave function $-g$ is equivalent to lower bounding the convex function g. Given the current set of parameters $\mathbf{w}_t$, we obtain a linear approximation $l(\mathbf{w};\mathbf{R})$ of the function $g(\mathbf{w};\mathbf{R})$ such that

$$l(\mathbf{w}_t;\mathbf{R}) = g(\mathbf{w}_t;\mathbf{R}), l(\mathbf{w};\mathbf{R}) \leq g(\mathbf{w};\mathbf{R}), \forall \mathbf{w}. \tag{28}$$

In other words, the linear function $l(\mathbf{w};\mathbf{R})$ is a lower bound on the function $g(\mathbf{w};\mathbf{R})$ such that the lower bound is tight at the current parameters $\mathbf{w}_t$. Since $g(\mathbf{w};\mathbf{R})$ is a convex function, therefore, the linear approximation of g at a given $\mathbf{w}_t$ lowerbounds g. Mathematically,

$$\left.\frac{\partial g(\mathbf{w};\mathbf{R})}{\partial \mathbf{w}}\right|_{\mathbf{w}_t} \leq g(\mathbf{w};\mathbf{R}), \forall \mathbf{w} \tag{29}$$

$$(\mathbf{w} - \mathbf{w}_t)^\top \frac{\partial g(\mathbf{w};\mathbf{R})}{\partial \mathbf{w}_t} + g(\mathbf{w}_t;\mathbf{R}) \leq g(\mathbf{w};\mathbf{R}), \forall \mathbf{w} \tag{30}$$

Note that, $g(\mathbf{w};\mathbf{R})$ is the linear combination of max-marginals with positive coefficients. Also, max-marginals are the functions of $\mathbf{w}$, therefore, finding gradient of $g(\mathbf{w};\mathbf{R})$ is equivalent to finding gradients of max-marginals independently and summing them up with their respective coefficients. Gradients for the max-marginals are not defined as max-marginals are the *max* over affine functions, therefore, non-smooth. Instead, we calculate the subgradients. The subgradient of the max-marginal at $\mathbf{w}_t$ can be defined as:

$$\begin{aligned}\frac{\partial(m_i^+(\mathbf{w}))}{\partial \mathbf{w}_t} &= \frac{\partial(\text{argmax}_{\mathbf{Y},y_i=1}\, \mathbf{w}^\top \Psi(\mathbf{X},\mathbf{Y}))}{\partial \mathbf{w}_t}, \\ &= \Psi(\mathbf{X},\bar{\mathbf{Y}}_i^+), \bar{\mathbf{Y}}_i^+ = \underset{\mathbf{Y},y_i=1}{\text{argmax}}\, \mathbf{w}_t^\top \Psi(\mathbf{X},\mathbf{Y}). \end{aligned} \tag{31}$$

Similarly, subgradient of $m_j^-(\mathbf{w})$ can be defined. From equations (27) and (31) it is clear that:

$$g(\mathbf{w}_t;\mathbf{R}) = (\mathbf{w}_t)^\top \frac{\partial g(\mathbf{w};\mathbf{R})}{\partial \mathbf{w}_t} \tag{32}$$

Combining equations (32) and (30), the lowercound can be defined as:

$$\begin{aligned} l(\mathbf{w}_t;\mathbf{R}) &= \mathbf{w}^\top \frac{\partial g(\mathbf{w};\mathbf{R})}{\partial \mathbf{w}_t} & (33)\\ &\leq g(\mathbf{w};\mathbf{R}), \forall \mathbf{w}. & (34)\end{aligned}$$

Also from equation (32) it's clear that $l(\mathbf{w}_t;\mathbf{R}) = g(\mathbf{w}_t;\mathbf{R})$. Therefore, combining equations (33), (31) and (27) leads to the following form of $l(\mathbf{w};\mathbf{R})$:

$$\begin{aligned} l(\mathbf{w};\mathbf{R}) = &\frac{1}{|\mathcal{P}||\mathcal{N}|} \sum_{i\in\mathcal{P}} \mathbf{w}^\top \Psi(\mathbf{X},\bar{\mathbf{Y}}_i^+) \left(\sum_{j\in\mathcal{N}} (\mathbf{R}_{ij}^* - \mathbf{R}_{ij}) \right) \\ &+ \frac{1}{|\mathcal{P}||\mathcal{N}|} \sum_{j\in\mathcal{N}} \mathbf{w}^\top \Psi(\mathbf{X},\bar{\mathbf{Y}}_j^-) \left(\sum_{i\in\mathcal{P}} (\mathbf{R}_{ij}^* - \mathbf{R}_{ij}) \right). \end{aligned} \tag{35}$$

Algorithm 2 The Cutting Plane algorithm for optimizing the resulting convex function of HOAP-SVM.

```
input Samples $\mathbf{X}$, ranking $\mathbf{R}^*$, tolerance $\epsilon$, initial parameters $\mathbf{w}_0$.
$t \leftarrow 0$.
Start with empty constraint set: $\mathcal{C} \leftarrow \{\}$.
repeat
    Find the most violated constraint using following equation:
        $\hat{R} \leftarrow \operatorname*{argmax}_{R} \Big\{ \eta \sum_{i \in \mathcal{P}} \sum_{j \in \mathcal{N}} R_{ij}(\bar{s}_i(\mathbf{w}) - \bar{s}_j(\mathbf{w})) + \Delta(R, R^*) - \eta \sum_{i \in \mathcal{P}} \sum_{j \in \mathcal{N}} R^*_{ij}(\bar{s}_i(\mathbf{w}) - \bar{s}_j(\mathbf{w})) \Big\}$   (36)
    Update the constraint set with the new most violated constraint.
    Update the parameters $\mathbf{w}$ by solving the following convex optimization problem
    with all the constraints in the constraint set $\mathcal{C}$:
        $\mathbf{w}_{t+1} = \operatorname*{argmin}_{\mathbf{w}} \ \frac{1}{2}||\mathbf{w}||^2 + C\xi,$   (37)
until No additional constraint can be added.
```

While at first sight the problem of obtaining the linear approximation for each ranking matrix $\mathbf{R}$ may appear to be highly expensive, but a little insight to the function $g(\mathbf{w}; \mathbf{R})$ reveals that each constraint corresponding to each $\mathbf{R}$ has exactly the same convex terms, $m_i^+(\mathbf{w})$ and $m_j^-(\mathbf{w})$, independent of $\mathbf{R}$. Therefore, we don't have to lowerbound g again and again for each $\mathbf{R}$. Also, lowerbounding $g(\mathbf{w}; \mathbf{R})$ at a given $\mathbf{w}_t$ is equivalent to finding the max-marginal labelings of the entire dataset at $\mathbf{w}_t$ such that each sample is marginalized to get it's ground truth label (see equation 31)). Each labeling problem can be solved using graph cut [3]. Solving graph cuts n times naively would be computationally expensive, therefore, in order to improve the efficiency, we use dynamic graph cuts [12]. In the case of max-marginals, the structure of the graph remains the same for all the samples, and only unary potentials change. Therefore, the time complexity of the dynamic graph cuts is mainly dominated by the time complexity of the graph creation and the first cut in the graph, which is computationally efficient [11].

A.5 Proposition 5

Proof. In the second step of CCCP, we obtain a convex optimization problem by substituting the linear approximation $l(\mathbf{w}; \mathbf{R})$ in place of the convex function $g(\mathbf{w}; \mathbf{R})$. We update the parameters by solving the convex optimization problem. Since the space of $\mathbf{R}$ is very large, therefore, in order to handle exponentially many constraints we use cutting plane algorithm (Algorithm 2) to optimize the resulting convex objective function [10] . The algorithm iteratively introduces *most violated constraints* until we have solved the original problem within a desired tolerance ϵ. In case of bounded joint feature map and bounded loss function, the number of iterations to solve this convex optimization is independent of the number of training samples and is linear in the desired precision (ϵ) and the regularization parameter (C) [10]. Let's define some short hand notations in order to derive the final formulation for finding the most violated constraint. For given $\bar{\mathbf{Y}}_i^+$ and $\bar{\mathbf{Y}}_j^-$ (see equation 31), the scores for the lower bounded functions corresponding to the current $\mathbf{w}$ can be defined as:

$$\bar{m}_i^+(\mathbf{w}) = \mathbf{w}^\top \varPsi(\mathbf{X}, \bar{\mathbf{Y}}_i^+), \bar{m}_j^-(\mathbf{w}) = \mathbf{w}^\top \varPsi(\mathbf{X}, \bar{\mathbf{Y}}_j^-). \tag{38}$$

Substituting (38) into (24) leads to the following objective function:

$$\begin{aligned} \min_{\mathbf{w}} \quad & \frac{1}{2}||\mathbf{w}||^2 + C\xi, \\ s.t. \quad & \eta \sum_{i \in \mathcal{P}} \sum_{j \in \mathcal{N}} \mathbf{R}^*_{ij} \{(\bar{m}_i^+(\mathbf{w}) - m_i^-(\mathbf{w})) - (m_j^+(\mathbf{w}) - \bar{m}_j^-(\mathbf{w}))\} \\ & - \eta \sum_{i \in \mathcal{P}} \sum_{j \in \mathcal{N}} \mathbf{R}_{ij} \{(\bar{m}_i^+(\mathbf{w}) - m_i^-(\mathbf{w})) - (m_j^+(\mathbf{w}) - \bar{m}_j^-(\mathbf{w}))\} \\ & \geq \varDelta(\mathbf{R}, \mathbf{R}^*) - \xi, \ \xi \geq 0, \forall R. \end{aligned} \tag{39}$$

Therefore, *finding the most violated constraint* problem can be defined as:

$$\begin{aligned} \hat{R} \leftarrow \operatorname*{argmax}_{R} & \Big\{ \eta \sum_{i \in \mathcal{P}} \sum_{j \in \mathcal{N}} R_{ij} (\bar{s}_i(\mathbf{w}) - \bar{s}_j(\mathbf{w})) + \varDelta(R, R^*) \\ & - \eta \sum_{i \in \mathcal{P}} \sum_{j \in \mathcal{N}} R^*_{ij} (\bar{s}_i(\mathbf{w}) - \bar{s}_j(\mathbf{w})) \Big\} \end{aligned} \tag{40}$$

where, $\bar{s}_i(\mathbf{w}) = (\bar{m}_i^+(\mathbf{w}) - m_i^-(\mathbf{w}))$ and $\bar{s}_j(\mathbf{w}) = (m_j^+(\mathbf{w}) - \bar{m}_j^-(\mathbf{w}))$. Note that the problem (40) is equivalent to the standard problem of finding the most violated constraint as defined in [26], therefore, the most violated constraint $\hat{R}$ can be computed using the optimal greedy procedure proposed in [26]. This step is mainly dominated by the sorting of the scores $s_i(\mathbf{w})$ and $s_j(\mathbf{w})$ (40), which has the time complexity of $O(nlogn)$.

References

1. Behl, A., Jawahar, C.V., Kumar, M.P.: Optimizing average precision using weakly supervised data. In: CVPR (2014)
2. Bourdev, L., Malik, J.: Poselets: Body part detectors trained using 3D human pose annotations. In: IJCV (2009)
3. Boykov, Y., Veksler, O., Zabih, R.: Fast approximate energy minimization via graph cuts. In: PAMI (2001)
4. Delaitre, V., Laptev, I., Sivic, J.: Recognizing human actions in still images: a study of bag-of-features and part-based representations. In: BMVC (2010)
5. Everingham, M., Gool, L., Williams, C., Winn, J., Zisserman, A.: The pascal visual object classes (voc) challenge. In: IJCV (2010)
6. Finley, T., Joachims, T.: Training structural SVMs when exact inference is intractable. In: ICML (2008)
7. Franc, V., Savchynskyy, B.: Discriminative learning of max-sum classifiers. In: JMLR (2008)
8. Horst, R., Thoai, N.: DC programming overview. Journal of Optimization Theory and Applications (1999)
9. Joachims, T.: A support vector method for multivariate performance measures. In: ICML (2005)
10. Joachims, T., Finley, T., Yu, C.: Cutting-plane training of structural SVMs. Machine Learning (2009)
11. Kohli, P., Torr, P.: Measuring uncertainty in graph cut solutions efficiently computing min-marginal energies using dynamic graph cuts. In: ECCV (2006)
12. Kohli, P., Torr, P.: Dynamic graph cuts for efficient inference in Markov random fields. In: PAMI (2007)
13. Kolmogorov, V.: Convergent tree-reweighted message passing for energy minimization. In: PAMI (2006)
14. Kolmogorov, V., Zabih, R.: What energy functions can be minimized via graph cuts. In: PAMI (2004)
15. Komodakis, N.: Efficient training for pairwise or higher order crfs via dual decomposition. In: CVPR (2011)
16. Komodakis, N., Paragios, N., Tziritas, G.: MRF energy minimization and beyond via dual decomposition. In: PAMI (2011)
17. Kulesza, A., Pereira, F.: Structured learning with approximate inference. In: NIPS (2007)
18. Maji, S., Bourdev, L., Malik, J.: Action recognition from a distributed representation of pose and appearance. In: CVPR (2011)
19. Oliva, A., Torralba, A.: Modeling the shape of the scene: A holistic representation of the spatial envelope. In: IJCV (2001)
20. Pearl, J.: Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference. Morgan Kaufmann (1988)
21. Rosenfeld, N., Meshi, O., Globerson, A., Tarlow, D.: Learning structured models with the AUC loss and its generalizations. In: AISTAT (2014)
22. Taskar, B., Guestrin, C., Koller, D.: Max-margin Markov networks. In: NIPS (2003)
23. Taskar, B., Lacoste-Julien, S., Jordan, M.I.: Structured prediction, dual extragradient and bregman projections. In: JMLR (2006)
24. Tsochantaridis, I., Hofmann, T., Joachims, T., Altun, Y.: Support vector machine learning for interdependent and structured output spaces. In: ICML (2004)
25. Vapnik, V.: Statistical learning theory. Wiley (1998)

26. Yue, Y., Finley, T., Radlinski, F., Joachims, T.: A support vector method for optimizing average precision. In: SIGIR (2007)
27. Yuille, A., Rangarajan, A.: The concave-convex procedure. Neural Computation (2003)