Optimizing Average Precision using Weakly Supervised Data

Thesis submitted in partial fulfillment
of the requirements for the degree of

Master of Science (by Research)
in
Computer Science Engineering

by

Aseem Behl
201050058
aseem.behl@research.iiit.ac.in

International Institute of Information Technology
Hyderabad - 500 032, INDIA
July 2015

International Institute of Information Technology
Hyderabad, India

CERTIFICATE

It is certified that the work contained in this thesis, titled "Optimizing Average Precision using Weakly Supervised Data" by Aseem Behl, has been carried out under my supervision and is not submitted elsewhere for a degree.

_____ _____

Date Adviser: Prof. C.V. Jawahar

_____ _____

Date Adviser: Dr. M. Pawan Kumar

To my Family

Acknowledgments

I would like to thank my advisers Dr. M Pawan Kumar and Prof. C.V. Jawahar for all the guidance and support. They have been a constant source of inspiration and their guidance has been critical to my development as a budding researcher in machine learning. Their mentorship has not only helped in my research but has truly made a difference to my life. It was both a honour and a great privilege to work with them.

Working at CVIT was great fun. I was fortunate to have the stimulating company of many brilliant students and wonderful friends - Vijay, Praveen, Natraj, Pritish, Udit, Devender, Nagender, Viresh, Aniket.

Many thanks to Prof. P.J. Narayan, Prof. Jayanthi and Dr. Anoop Namboodiri for their encouraging presence and for providing an environment conducive to research of the finest quality. I am grateful to Mr. R. S. Satyanarayana for all the administrative support.

Finally, I would like to thank my family for supporting me always.

Abstract

Many tasks in computer vision, such as action classification and object detection, require us to rank a set of samples according to their relevance to a particular visual category. The performance of such tasks is often measured in terms of the average precision (AP). Yet it is common practice to employ the support vector machine (SVM) classifier, which optimizes a surrogate 0-1 loss. The popularity of SVM can be attributed to its empirical performance. Specifically, in fully supervised settings, SVM tends to provide similar accuracy to AP-SVM, which directly optimizes an AP-based loss. However, we hypothesize that in the significantly more challenging and practically useful setting of weakly supervised learning, it becomes crucial to optimize the right accuracy measure. In order to test this hypothesis, we propose a novel latent AP-SVM that minimizes a carefully designed upper bound on the AP-based loss function over weakly supervised samples. Using publicly available datasets, we demonstrate the advantage of our approach over standard loss-based learning frameworks on three challenging problems: action classification, character recognition and object detection.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Several problems in computer vision can be formulated as ranking tasks, that is, sorting a set of samples according to their relevance to a query. As a running example throughout this paper, we will consider the task of action classification, where the input is a set of images and the desired output is a ranking of the images. An accurate output is one that ranks an image containing a person performing an action of interest (such as 'jumping' or 'walking') higher than the images that do not contain a person performing the action of interest. Ranking is often reformulated as binary classification, for which there exist several learning frameworks. Among the most popular binary classifiers in computer vision is the support vector machine (SVM) [52]. The popularity of the support vector machine (SVM) [52] can be gauged by its numerous applications in computer vision including, image classification [37, 59], action classification [16, 39, 62] and object detection [15, 53]. During training, an SVM minimizes a convex regularized upper bound on the misclassification error over a fully supervised dataset, which consists of positive (that is, relevant) and negative (that is, non-relevant) samples. During testing, the samples are sorted in descending order of the scores provided by the SVM. The main advantages of SVM are its well-understood connections to statistical learning theory [52] and the availability of efficient algorithms to learn its parameters [27, 30, 46].

One of the disadvantages of SVM is that it optimizes the 0-1 loss instead of the average precision (AP) over the training dataset. As the most commonly used accuracy measure for ranking tasks in computer vision is the average precision (AP) [19], and not misclassification error, the choice of SVM may appear surprising. The case for its use appears even weaker when we consider that there already exists a related learning framework (henceforth referred to as AP-SVM) that optimizes an AP-based loss function (henceforth referred to as the AP loss) [64]. However, a closer look at the empirical evidence reveals the reasoning behind this choice: SVM can be trained more efficiently, and provides comparable accuracy to AP-SVM.

The above observation suggests that we should continue to collect fully supervised datasets and use simple loss functions. If the supervision involves labelling each sample with its class (positive or negative), then this task does not appear to be daunting. However, recent research has shown that the key to achieving high classification and ranking accuracy is to provide additional annotations for each

sample, which can guide the learner towards the correct output [16, 22, 39, 60, 62]. Going back to the example of action classification (Figure 1.1), it would be helpful to not only know the class information of each image but the exact location of the person in the image.

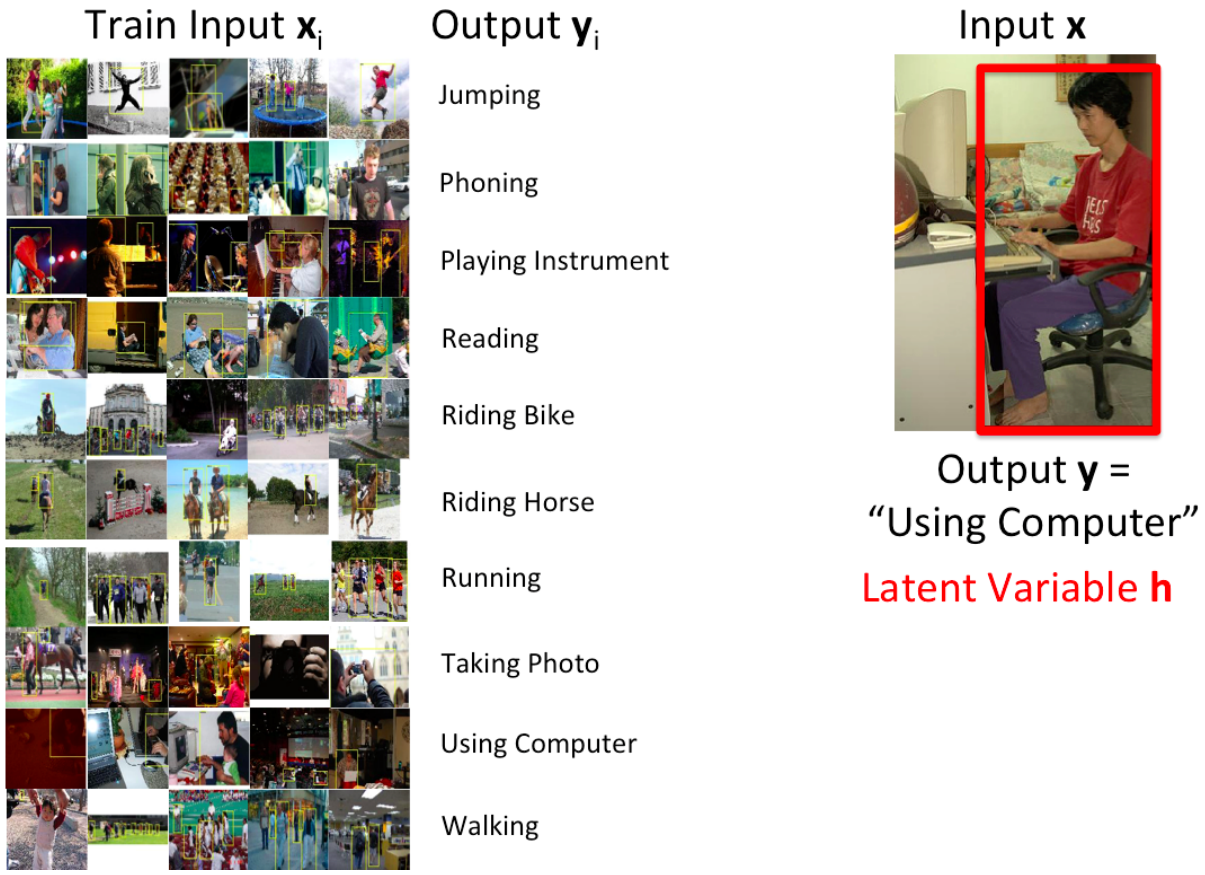

Figure 1.1: *Task of action classification as a weakly supervised learning problem. Although class information of each image is available, exact locations of the person in the image is missing. Weakly supervised learning methods work by modelling this missing information as latent variables.*

The need for complex additional annotations makes supervised learning impractical. To overcome this deficiency, researchers have started exploring weakly supervised learning [2, 18, 22, 32, 33, 40, 43, 42, 45, 54, 56], where the annotations of some or all the samples contain missing information. The 0-1 loss based latent SVM [22], which models missing annotations as latent variables is a popular weakly supervised learning algorithm. Latent SVM can be thought of as a special case of latent structured SVM (latent SSVM) [47, 63], which optimizes a general loss function. Latent SSVM has received considerable attention in the computer vision community [22, 33, 35, 54, 56, 57, 61], on tasks ranging from binary classification (such as object detection) to structured output prediction (such as semantic segmentation and indoor scene understanding). Not surprisingly, the convenience of using partial annotations comes at

the cost of a significantly more challenging machine learning problem. Specifically, weakly supervised learning typically requires us to solve a non-convex optimization problem, which makes it prone to converge to a bad local minimum.

1.1 Problem Statement and Contributions

Given the inherent difficulty of the weakly supervised learning problem, we hypothesize that the choice of the loss function becomes crucial in such settings. In order to provide empirical evidence for our hypothesis, we propose a novel latent AP-SVM framework that models the missing additional annotations using latent variables. Our formulation differs from the standard latent structured SVM (latent SSVM) [63] for general loss functions in three significant aspects. First, it uses a more intuitive two-step prediction criterion, where the first step consists of choosing the best latent variable for each sample and the second step consists of ranking the samples. This is in contrast to the latent SSVM formulation, which requires the joint optimization of the latent variables and the ranking. For example, in 'jumping' action classification, our latent AP-SVM formulation would first pick out the bounding box that is most likely to contain a 'jumping' person in each image, and then rank them. In contrast, the latent SSVM formulation would require us to simultaneously classify the samples as positive or negative, while picking out the best bounding box for the positive images (bounding box that is most likely to contain a 'jumping' person) and the worst bounding box for the negative images (bounding box that is least likely to contain a 'jumping' person). Second, using the above prediction criterion, the parameters of latent AP-SVM are learned by minimizing a tighter upper bound on the AP loss compared to latent SSVM. Third, unlike latent SSVM, latent AP-SVM lends itself to efficient optimization during learning, which is guaranteed to provide a local minimum or saddle point solution. While the first of the aforementioned differences makes our approach more intuitive, the latter two differences provide a sound theoretical justification for its superiority to latent SSVM. In order to demonstrate that the theoretical superiority also translates to better empirical results, we provide a thorough comparison of latent AP-SVM with the baseline methods for three challenging problems: action classification, character recognition and object detection. For the sake of clarity, we defer the details that are not essential for the understanding of the paper to the appendices. To facilitate the use of latent AP-SVM, we have made our code and data available online at `http://cvit.iiit.ac.in/projects/lapsvm/`.

1.2 Thesis Outline

In Chapter 2, we review the existing algorithms for binary classification in fully and weakly supervised learning setting. We then review the current methods for the learning to rank problem and motivate the importance of optimising alternate loss functions like average precision (AP). In this section, we also outline the current approaches in optimising average precision (AP) in fully and weakly supervised setting. The notation used in the thesis is also outlined in this chapter.

In Chapter 3, we show the theoretical benefit of our novel latent AP-SVM over the standard latent SSVM formulation, namely that it minimizes a tighter upper bound on the AP loss and allows for efficient inference, while using an intuitive prediction rule.

In Chapter 4, show that the theoretical benefits translate to improved empirical performance using three important and challenging problems in computer vision, namely, action classification, character recognition and object detection. We also describe the experimental setup and the properties of the datasets used for the evaluation of the proposed algorithm.

Chapter 5 summarizes the contributions of our work and outlines possible directions of future work.

Chapter 2

Background

2.1 Support Vector Machines

Given a data example with observed variables $\{\mathbf{x_1}, \mathbf{x_2}, \cdots, \mathbf{x_n}\} \in \mathbf{R^m}$, we wish to classify the example by assigning a label, $\mathbf{y} \in \{-\mathbf{1}, \mathbf{1}\}$. ($\mathbf{y} = \mathbf{1}$ denotes a "positive" example and $\mathbf{y} = -\mathbf{1}$, a "negative" example.) A linear classifier is a predictor of the form:

$$\mathbf{y_i^{pred}} = sign(\mathbf{w^T \Phi(x_i)})$$

Here, $\mathbf{w} \in \mathbf{R^d}$ are the parameters of the classifier and $\mathbf{\Phi} : \mathbf{R^m} \rightarrow \mathbf{R^d}$ is a feature map, which maps the observed variables in $\mathbf{R^m}$ to a feature vector in $\mathbf{R^d}$.

Learning is the process of fitting the parameter vector $\mathbf{w}$ to a set of training examples, by optimising an objective function defined over the training set. Let $\{\mathbf{x_i}, \mathbf{y_i}\}_{i=1}^n$ be a set of n training examples, each of which is comprised of an observed data vector $\mathbf{x_i}$ and a ground-truth label $\mathbf{y_i}$. The support vector machine (SVM) has a geometric interpretation of finding the separating hyperplane with maximum geometric margin between the sets of positive and negative training examples in the feature space. The separating hyperplane is $\{\mathbf{x} : \mathbf{w}^T \mathbf{\Phi}(\mathbf{x}) = 0\}$, and the geometric margin is the maximum distance that this hyperplane can be translated in the direction $\mathbf{w}$ or -$\mathbf{w}$ before it touches a positive or negative example. A support vector machine is a linear classifier whose training procedure is finding the minimum geometric margin problem formulated as

$$\begin{aligned} \min_{\mathbf{w}} \quad & \frac{1}{2}||\mathbf{w}||^2, \\ \text{s.t. } \forall i = 1, \cdots, n \quad & \mathbf{y_i}(\mathbf{w^T x_i}) \geq 1 \\ \forall i = 1, \cdots, n \quad & \xi_\mathbf{i} \geq 1 \end{aligned} \tag{2.1}$$

Note that this imposes the hard constraint that $\mathbf{w}$ must define a hyperplane that separates all positive examples from all negatives. This constraint can be relaxed to allow violations in exchange for incurring a penalty in the objective. This relaxation yields the following problem:

$$\begin{aligned} \min_{\mathbf{w}} \quad & \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{n}\xi_{\mathbf{i}}, \\ \text{s.t. } \forall i = 1, \cdots, n \quad & \mathbf{y_i}(\mathbf{w^T x_i}) \geq 1 - \xi_{\mathbf{i}} \\ \forall i = 1, \cdots, n \quad & \xi_{\mathbf{i}} \geq 1 \end{aligned} \tag{2.2}$$

Here, the first term is a quadratic regularisation term on $\mathbf{w}$ that prevents overfitting. The slack variables ξ's define the penalty incurred by each example for violating the margin constraint $\mathbf{y_i w^T x_i} \geq 1$. The first term is a quadratic regularisation term on $\mathbf{w}$ that prevents overfitting. This is known as the soft-margin formulation of the SVM. The support vector machine is a powerful algorithm used in many binary classification applications in computer vision and machine learning, including object detection, image segmentation, and spam filtering.

2.2 Structural SVM

The structural support vector machine (SSVM) is a generalisation of the standard binary support vector machine to structured output spaces, where the label, $\mathbf{y}$, is a vector whose elements can be, for instance, sequences, strings, labeled trees, lattices, or graphs. Such output spaces arise in a variety of applications, like multilabel classification, classification with class taxonomies, to label sequence learning, sequence alignment learning, and supervised grammar learning, to name just a few. 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.

Within the SSVM framework, the prediction requires us to maximize the score over all possible outputs for an input, thus we have a linear predictor of the form:

$$\mathbf{y^{pred}} = \underset{\mathbf{y}}{\operatorname{argmax}}(\mathbf{w^T \Psi}(\mathbf{x}, \mathbf{y}))$$

As in the binary SVM case, $\mathbf{x}$ is the observed data vector. Note that the feature function, Ψ, is now a function of both $\mathbf{x}$ and $\mathbf{y}$, rather than of $\mathbf{x}$ alone. Given a training dataset, the parameters of an SSVM are learned by minimizing a regularized convex upper bound on a user-specified loss function.

$$\begin{aligned} \min_{\mathbf{w}} \quad & \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{n}\xi_{\mathbf{i}}, \\ \text{s.t.} \quad & \forall \hat{\mathbf{y}}_{\mathbf{i}} : \{\mathbf{w}^\top \Psi(\mathbf{x_i}, \mathbf{y_i}) - \mathbf{w}^\top \Psi(\mathbf{x_i}, \hat{\mathbf{y}}_{\mathbf{i}})\} \geq \Delta(\hat{\mathbf{y}}_{\mathbf{i}}, \mathbf{y_i}) - \xi_{\mathbf{i}} \end{aligned} \tag{2.3}$$

Here, $\Delta(\hat{\mathbf{y}}_{\mathbf{i}}, \mathbf{y_i})$ is a loss function, which captures the penalty associated with mislabeling $\mathbf{y_i}$ as $\hat{\mathbf{y}}_{\mathbf{i}}$. It must be nonnegative and satisfy the condition that the correct labeling yields a penalty of zero. As in the

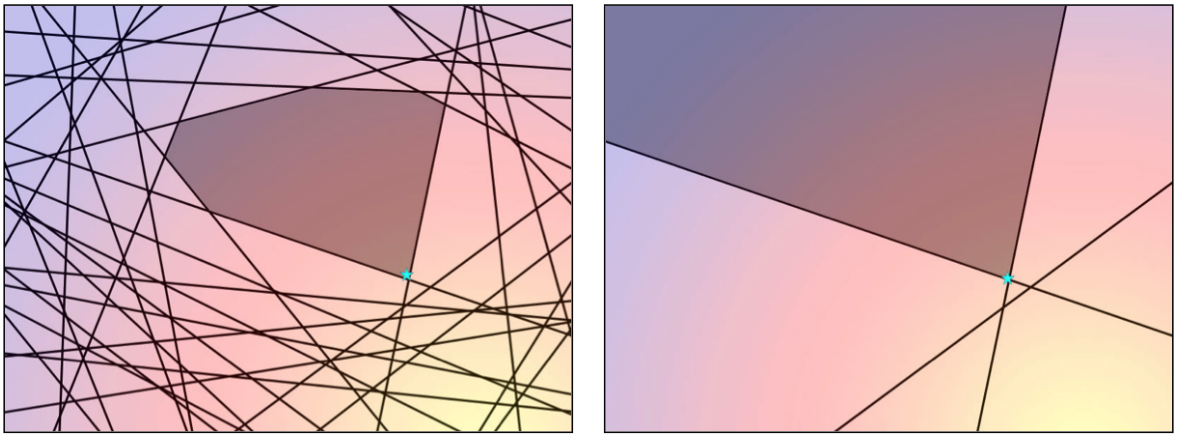

Figure 2.1: Illustrative example comparing SVM & SSVM learning respectively. SVM problem has exponential constraints dominated by small set of critical constraints. Whereas, SSVM repeatedly finds the next most violated constraint until set of constraints is a good approximation.

binary SVM case, the second term in the objective $\xi_\mathbf{i}$ is the margin constraint violation penalty incurred by each example in the training set. Clearly, there will always be at least one linear constraint for which this inequality is tight, otherwise, we could reduce the objective by lowering $\xi_\mathbf{i}$ by a small amount. The constraint for which the inequality is tight is also known as the "most violated constraint" associated with i^{th} example. Computing the most violated constraint is a crucial subroutine in learning the SSVM. For the benefit of the reader, we review the SSVM learning algorithm in the following subsection 2.2.1.

2.2.1 Structural SVM Learning

The formulation of the SSVM objective with linear constraints in problem (2.3) is a convex problem. However, the number of linear constraints can be intractably large, so instead the problem is typically treated using the cutting plane method described in Tsochantaridis, et al [50]. This cutting plane method exploits the sparsity and structure of the problem to reduce the number of constraints needed to be considered. This method requires us to specify how to compute the most violated constraint for each example, which can be done using a procedure known as "loss-augmented inference". 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. 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.

$$\hat{\mathbf{y}} = \underset{\mathbf{y}}{\operatorname{argmax}}(\mathbf{w}^\mathbf{T}\mathbf{\Psi}(\mathbf{x_i}, \mathbf{y})) + \Delta(\mathbf{y}, \mathbf{y_i})$$

Note that this is identical to the standard inference procedure, except that we add the loss into the inference objective. The method operates by finding a series of cutting-planes that successively reduce the set of potentially optimal values of $\mathbf{w}$ until this set is within ϵ of the solution. At each iteration, the set of most violated constraints yields a subgradient for the objective function at the current value of

$\mathbf{w}$. The subgradient, in turn, defines the cutting-plane for this iteration. Figure 2.1 shows an illustrative example comparing SVM & SSVM learning respectively. The cutting plane algorithm is outlined in Algorithm 1.

Algorithm 1 *Cutting plane algorithm for solving of* SSVM.

Require: $\{\mathbf{x_1}, \mathbf{x_2}, \cdots, \mathbf{x_n}\}, \{\mathbf{y_1}, \mathbf{y_2}, \cdots, \mathbf{y_n}\}, \mathbf{w}_0, \epsilon$
$\mathcal{W}_i \rightarrow \phi, \forall \mathbf{i} = 1, \cdots, \mathbf{n}$
repeat
  for $\mathbf{i} = 1, \cdots, \mathbf{n}$ **do**
    $\mathbf{H}(\mathbf{y}; \mathbf{w}) = \mathbf{\Delta}(\mathbf{y_i}, \mathbf{y}) + \mathbf{w^T}\mathbf{\Psi}(\mathbf{x_i}, \mathbf{y}) - \mathbf{w^T}\mathbf{\Psi}(\mathbf{x_i}, \mathbf{y_i})$
    compute $\hat{\mathbf{y}} = \text{argmax}_{\mathbf{y} \in \mathbf{Y}} \mathbf{H}(\mathbf{y}; \mathbf{w})$
    compute $\xi_\mathbf{i} = \max\{\mathbf{0}, \max_{\mathbf{y} \in \mathcal{W}_\mathbf{i}} \mathbf{H}(\mathbf{y}; \mathbf{w})\}$
    if $\mathbf{H}(\hat{\mathbf{y}}; \mathbf{w}) > \xi_\mathbf{i} + \epsilon$ **then**
      $\mathcal{W}_\mathbf{i} \rightarrow \mathcal{W}_\mathbf{i} \cup \{\hat{\mathbf{y}}\}$
      $\mathbf{w} \rightarrow$ optimize (2.3) over $\mathcal{W} = \bigcup_\mathbf{i} \mathcal{W}_\mathbf{i}$
    end if
  end for
until no $\mathcal{W}_i$ has changed during iteration tolerance ϵ

2.3 Weakly Supervised Learning

The ability to incorporate diverse datasets into the training regime is valuable, because it is often the case that only a small set of fully annotated data is available due to the time and expense of labelling such data. In contrast, there may be a much larger source of partially annotated data. We would like to be able to make use of both the limited supply of fully annotated data as well as the large amount of weak or noisy annotations. In order to use diverse data during training, researchers have started exploring weakly supervised learning [2, 18, 22, 32, 33, 40, 43, 42, 45, 54, 56], where the annotations of some or all the samples contain missing information. The convenience of using partial annotations comes at the cost of a significantly more challenging machine learning problem. Specifically, weakly supervised learning typically requires us to solve a non-convex optimization problem, which makes it prone to converge to a bad local minimum. The LSVM framework [22, 63] elegantly incorporates such diverse sources of data into a single, unified model.

2.3.1 Latent Structural SVM

Latent structural SVMs (LSVM) [22, 63] extend the SSVM framework to incorporate latent (or hidden) variables. These variables comprise part of the label $\mathbf{y}$ and may not be observed at training time. The use of hidden variables enables the use of diverse data in training. In the non-latent SSVM problem, it is assumed that each training instance is labeled with the same level of annotation, encompassing the entire label vector $\mathbf{y}$. In a training set of diverse data, some types of annotation will be provided for all

training examples, but other types of annotation will be available for only a subset of training examples. We can incorporate the sometimes-available annotations as latent variables within the LSVM framework. For each sample $\mathbf{x_i}$, the dataset can also provide additional annotations, which we denote by $\mathbf{h_i}$. For example, in action classification each sample represents an image and the additional annotation $\mathbf{h_i}$ can represent the bounding box of the person in the image. In the weakly supervised setting, we consider a setting where the additional annotations $\mathbf{h_i}$'s are unknown.

In the LSVM framework, the inference procedure remains the same as before. At training time, however, we decompose the label into two parts, $(\mathbf{y}, \mathbf{h})$. The annotation, $\mathbf{y}$, is the observed portion of the label at training time while the latent variable $\mathbf{h}$ is the hidden portion. The training problem is:

$$\begin{aligned} \min_{\mathbf{w}} \quad & \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{n}\xi_\mathbf{i}, \\ \text{s.t.} \quad & \forall \hat{\mathbf{y}}_\mathbf{i}, \hat{\mathbf{h}}_\mathbf{i} : \{\max_{\mathbf{h_i}} \mathbf{w}^\top \Psi(\mathbf{x_i}, \mathbf{y_i}, \mathbf{h_i}) - \mathbf{w}^\top \Psi(\mathbf{x_i}, \hat{\mathbf{y}}_\mathbf{i}, \hat{\mathbf{h}}_\mathbf{i})\} \geq \Delta(\mathbf{y_i}, \hat{\mathbf{y}}_\mathbf{i}) - \xi_\mathbf{i} \end{aligned} \tag{2.4}$$

As in the case of SSVMs, the LSVM training objective is a regularised upper bound on the empirical loss of the dataset.

LSVM Learning via CCCP

Unlike the training problems of the binary SVM and structural SVM, the LSVM training problem is not convex. Rather, the objective is the difference of two convex functions. Because it is not convex, it is intractable to solve exactly. We can, however, minimize a convex upper-bound on the objective by iteratively approximating the concave portion of the objective with an linear upper bound and then optimizing the resulting convex function. This method is known as the concave-convex procedure (CCCP). Concretely, the learning algorithm proceeds iteratively, with each iteration consisting of two stages: (i) the hidden variables are either imputed which corresponds to approximating the concave function by a linear upper bound; and (ii) updating the value of the parameter using the values of the hidden variables. Note that updating the parameters requires us to solve a convex SSVM learning problem (where the output $\mathbf{y_i}$ is now concatenated with the hidden variable $\mathbf{h_i^*}$) for which several efficient algorithms exist in the literature. To use this method, we therefore have to specify two subroutines. The first is the impute latent variable routine ($\mathbf{h_i^*} = \text{argmax}_{\mathbf{h}\in\mathbf{H}}\, \mathbf{w^T\Psi}(\mathbf{x_i}, \mathbf{y_i}, \mathbf{h_i}$, where $\mathbf{H} = \mathbf{dom}(\mathbf{h})$) and the second is the loss-augmented inference routine associated with solving the standard SSVM. It can be shown that CCCP converges to a local minimum. The CCCP algorithm is outlined in Algorithm 2.

2.4 Learning To Rank

Many tasks in computer vision and information retrieval systems require the development of automatic methods that sort a given set of samples according to their relevance to a query. For example,

Algorithm 2 *The* CCCP *algorithm for parameter estimation of* LSVM.

Require: $\{\mathbf{x_1}, \mathbf{x_2}, \cdots, \mathbf{x_n}\}, \{\mathbf{y_1}, \mathbf{y_2}, \cdots, \mathbf{y_n}\}, \mathbf{w}_0, \epsilon$

1: $t \Leftarrow 0$
2: **repeat**
3: Impute the latent variables by solving $\mathbf{h_i^*} = \text{argmax}_{\mathbf{h} \in \mathbf{H}} \, \mathbf{w^T \Psi}(\mathbf{x_i}, \mathbf{y_i}, \mathbf{h_i})$
4: Update $\mathbf{w}_{t+1}$ by fixing the latent variables for output $\mathbf{y_i}$ to $\mathbf{h_i^*}$ and solving the following convex problem,

$$\mathbf{w}_{t+1} = \text{argmin}_{\mathbf{w}} \, \tfrac{1}{2}||\mathbf{w}||^2 + C\xi,$$
$$\text{s.t. } \mathbf{w}^\top \Psi(\mathbf{x_i}, \mathbf{y_i}, \mathbf{h_i^*}) - \mathbf{w}^\top \Psi(\mathbf{x_i}, \hat{\mathbf{y}}_\mathbf{i}, \hat{\mathbf{h}}_\mathbf{i}) \geq \Delta(\mathbf{y_i}, \hat{\mathbf{y}}_\mathbf{i}) - \xi, \forall \hat{\mathbf{y}}_\mathbf{i}, \hat{\mathbf{h}}_i$$

5: $t \leftarrow t + 1$
6: **until** Objective function cannot be decreased below tolerance ϵ

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, learning to rank is an active research area where supervised learning is increasingly used [10, 28, 4, 3, 29, 64, 7, 49, 36]. Ranking models can be divided into three categories depending on the approach they are based on [38]: pointwise, pairwise, or listwise. The main difference consists in what form input and output data, as well as the loss function used.

The pointwise approach

In the pointwise approach, it is expected that each input-output data pair confronts with a certain numerical assessment. The input space of the pointwise approach contains the feature vector of each single sample. The output space contains the relevance degree of each single sample. The ground truth label for a sample is defined as the relevance degree of the sample. The loss function examines the accurate prediction of the ground truth label for each single sample. In different pointwise ranking algorithms, ranking is modelled as regression, classification, and ordinal regression. Therefore the corresponding regression loss, classification loss, and ordinal regression loss are used as the loss function. The disadvantage with pointwise model is that by its nature, ranking is more relative order prediction rather than a precise relevance degree of a given object. In other words, the pointwise approach does not consider the interdependency among samples, and thus the position of a sample in the final ranked list is invisible to its loss function. Thus, while the pointwise model is applied, a relative order between objects can not normally take part in the learning process. Example algorithms belonging to the pointwise approach include [12, 13, 14, 9, 48].

The pairwise approach

The pairwise approach does not focus on accurately predicting the relevance degree of each sample; instead, it cares about the relative order between two samples. Thus, the pairwise ranking reduces to building a binary classifier dealing with two input objects corresponding to the same query. The binary classifier is needed to determine which of them is more relevant. The input space of the pairwise approach contains a pair of samples, both represented as feature vectors. The output space contains the pairwise preference (which takes values from $\{1, -1\}$) between each pair of samples. The loss function measures the inconsistency between the hypothesis and the ground truth label. For example, in some algorithms, ranking is modelled as a pairwise classification, and the corresponding classification loss on a pair of samples is used as the loss function. The disadvantage with the pairwise approach is that the loss function used in the pairwise approach only considers the relative order between two samples. However, when one looks at only a pair of samples, the position of the samples in the final ranked list cannot be derived. Example algorithms belonging to the pairwise approach include [1, 6, 11, 23, 25].

The listwise approach

The main problem with the pointwise and pairwise approaches is that their loss functions are related to individual samples while most evaluation metrics of information retrieval measure the ranking quality for individual queries, not samples. This mismatch has motivated the so called listwise approaches for ranking, and is also the focus of this thesis. The listwise approach consists of building a model dealing with a set of all input samples corresponding. As a result it returns a ranked list of these samples. The input space of the listwise approach contains the entire group of samples associated with a query. The output space contains the ranked list of the samples. The loss function measures the difference between the ranked list given by the hypothesis and the ground truth list. As compared to the pointwise and pairwise approaches, the advantage of the listwise approach lies in that its loss function can naturally consider the positions of samples in the ranked list of all the samples associated with the same query. Example algorithms that belong to the listwise approach include [3, 7, 49, 64, 58, 44].

The main challenge in adapting supervised learning for listless approach to ranking problems, is that the evaluation criterion or "loss function" is usually defined over the ranking induced by the scoring function over all input instances. Therefore, the loss function is not decomposable over instances as in regular Support Vector Machines (SVMs). In the following section 2.5, we review the existing methods to optimise such non decombosible loss functions using the SSVM learning framework.

2.5 Optimising Alternate Loss Functions

In real-world applications, different application requirements often employ different domain specific performance measures to evaluate the success of learning algorithms. For example, F1-score and Precision-Recall Breakeven Point (PREBP) are usually employed in text classification; Precision and Re-

call are often used in information retrieval; Area Under the ROC Curve (AUC) and Average Precision (AP) are important to ranking. Ideally, to achieve good prediction performance, learning algorithms should train classifiers by optimizing the concerned performance measures. However, this is usually not easy due to the nonlinear and nonsmooth nature of many performance measures like F1-score and PREBP. During the past decade, many algorithms have been developed to optimize frequently used performance measures, and they have shown better performance than conventional methods [3, 64, 8, 29, 5, 34]. Listwise structured learning has been applied recently to optimize important non-decomposable ranking criteria like AUC (area under ROC curve) [29] and AP (average precision) [64]. The advantage of the structured learning approach [50] is that it fits a model to minimize the loss of a whole permutation, not individual or pairs of items. Thus, it helps break down the difficult non-convex ranking loss optimization problem into a convex quadratic program and a combinatorial optimization problem which, is often tractable and simple for ranking applications.

In [29] Joachims et al. present a Support Vector Method for optimizing multivariate nonlinear performance measures like the F1-score. They show that by taking a multivariate prediction approach, multivariate SVMs can be trained in polynomial time for large classes of potentially non-linear performance measures, in particular ROCArea, Precision/Recall Breakeven Point (PREBP), Precision at k (Prec@k), and all measures that can be computed from the contingency table.

In [64], Yue et al. present a general approach for learning ranking functions that maximize AP performance. Specifically, an SVM algorithm that globally optimizes a hinge-loss relaxation of AP. This approach simplifies the process of obtaining ranking functions with high AP performance by avoiding additional intermediate steps and heuristics. The new algorithm also makes it conceptually just as easy to optimize SVMs for AP as was previously possible only for accuracy and ROCArea.

In [8], Chakrabarti et al. further augment the class of nonsmooth ranking loss functions that can be directly and efficiently optimized for listwise structured learning. Specifically, they propose new, almost linear-time algorithms, for widely used search system evaluation measures like NDCG (normalized discounted cumulative gain) [55] and for MRR (mean reciprocal rank) [26].

Out of the aforementioned ranking measures, average precision (AP) is of particular interest to us namely because it is widely used as an evaluation measure in computer vision and machine learning tasks.

2.6 Optimising Average Precision

As the desired output of the aforementioned ranking problems is a ranking, the accuracy of an approach is typically reported using a ranking-based measure such as the average precision (AP). However, most current approaches do not optimize for the most commonly used evaluation measure, namely 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). However, the SVM framework has a major drawback, namely, 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. This drawback can aaddressed by using AP-SVM [64] which is a special form of the structured output support vector machines framework (SSVM) outlined in section 2.2.

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. [64] 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. For the benefit of the reader, we provide an overview of the AP-SVM framework in the following subsection 2.6.1

2.6.1 Supervised AP-SVM.

Notation. We use a similar notation to [64]. The training dataset consists of n samples $\mathbf{X} = \{\mathbf{x}_i, i = 1, \cdots, n\}$ together with their class information. 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 the positive class and $\mathbf{x}_j$ belongs to the negative class. Furthermore, for each sample $\mathbf{x}$, the dataset can also provide additional annotations, which we denote by $\mathbf{h}$. For example, in action classification each sample represents an image and the additional annotation $\mathbf{h}$ can represent the bounding box of the person in the image. To simplify the discussion in this section, we will assume that the additional annotations $\mathbf{h}$ are known for all samples. In the next section, we will describe the setting where the additional annotations are latent. We denote the set of all additional annotations for the positive and negative samples by $\mathbf{H}_P = \{\mathbf{h}_i, i \in \mathcal{P}\}$ and $\mathbf{H}_N = \{\mathbf{h}_j, j \in \mathcal{N}\}$ respectively.

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

AP Loss. Given a training dataset, our aim is to learn a ranking framework that provides a high AP measure. Let $\mathrm{AP}(\mathbf{Y}, \mathbf{Y}^*)$ denote the AP of the ranking matrix $\mathbf{Y}$ with respect to the true ranking $\mathbf{Y}^*$. The value of the $\mathrm{AP}(\cdot, \cdot)$ lies between 0 and 1, where 0 corresponds to a completely incorrect ranking $-\mathbf{Y}^*$ and 1 corresponds to the correct ranking $\mathbf{Y}^*$. In order to maximize the AP, we will minimize a loss function defined as $\Delta(\mathbf{Y}, \mathbf{Y}^*) = 1 - \mathrm{AP}(\mathbf{Y}, \mathbf{Y}^*)$.

Joint Feature Vector. For positive samples, the feature vector of the input $\mathbf{x}_i$ and additional annotation $\mathbf{h}_i$ is denoted by $\Phi_i(\mathbf{h}_i)$. Similarly, for negative samples, the feature vector of the input $\mathbf{x}_j$ and additional annotation $\mathbf{h}_j$ is denoted by $\Phi_j(\mathbf{h}_j)$. For example, in action classification, $\Phi_i(\mathbf{h}_i)$ can represent poselet [39] or bag-of-visual-words [16] features extracted from an image $\mathbf{x}_i$ using the pixels specified by the bounding box $\mathbf{h}_i$. Similar to [64], we specify a joint feature vector of the input $\mathbf{X}$, output $\mathbf{Y}$, and additional annotations $\{\mathbf{H}_P, \mathbf{H}_N\}$ as

$$\Psi(\mathbf{X}, \mathbf{Y}, \{\mathbf{H}_P, \mathbf{H}_N\}) = \frac{1}{|\mathcal{P}||\mathcal{N}|} \sum_{i \in \mathcal{P}} \sum_{j \in \mathcal{N}} \mathbf{Y}_{ij}(\Phi_i(\mathbf{h}_i) - \Phi_j(\mathbf{h}_j)).$$

In other words, the joint feature vector is the scaled sum of the difference between the features of all pairs of samples where one sample is positive and the other is negative.

Parameters. The parameter vector of the ranking framework is denoted by $\mathbf{w}$, and is of the same size as the joint feature vector. Given the parameters $\mathbf{w}$, the ranking of an input $\mathbf{X}$ is defined as the one that maximizes the score, that is,

$$\mathbf{Y}_{opt} = \operatorname*{argmax}_{\mathbf{Y}} \mathbf{w}^\top \Psi(\mathbf{X}, \mathbf{Y}, \mathbf{H}), \tag{2.5}$$

where $\mathbf{H}$ is the set of all the given additional annotations. Yue *et al.* [64] showed that the above optimization can be performed efficiently by sorting the samples $(\mathbf{x}_k, \mathbf{h}_k)$ in descending order of the score $\mathbf{w}^\top \Phi_k(\mathbf{h}_k)$.

Given the input $\mathbf{X}$, ranking matrix $\mathbf{Y}$, and additional annotations $\mathbf{H}_P$ and $\mathbf{H}_N$, we would like to learn the parameters $\mathbf{w}$ such that the AP loss over the training dataset is minimized. However, the AP loss is highly non-convex in $\mathbf{w}$, and minimizing it directly can result in a bad local minimum solution. To avoid this undesirable outcome, Yue *et al.* [64] proposed the AP-SVM formulation, which minimizes a regularized upper bound on the AP loss. Specifically, the model parameters are obtained by solving the following convex optimization problem:

$$\begin{aligned} \min_{\mathbf{w}} \quad & \frac{1}{2}||\mathbf{w}||^2 + C\xi, \\ \text{s.t.} \quad & \forall \mathbf{Y} : \{\mathbf{w}^\top \Psi(\mathbf{X}, \mathbf{Y}^*, \{\mathbf{H}_P, \mathbf{H}_N\}) \\ & -\mathbf{w}^\top \Psi(\mathbf{X}, \mathbf{Y}, \{\mathbf{H}_P, \mathbf{H}_N\})\} \geq \Delta(\mathbf{Y}^*, \mathbf{Y}) - \xi. \end{aligned} \tag{2.6}$$

Intuitively, the above problem introduces a margin between the score of the correct ranking and all incorrect rankings. The desired margin is proportional to the difference in their AP values. The hyper-parameter C controls the trade-off between the training error and the model complexity.

Problem (2.6) is specified over all possible rankings $\mathbf{Y}$, which is exponential in the number of training samples. Nonetheless, it can be solved efficiently using a cutting-plane method [64] described in Algorithm 3.

Briefly, the algorithm starts by specifying no constraints (step 1 of Algorithm 3: $\mathcal{W}$ is initialized to the null set). At each iteration, it adds a single constraint, which corresponds to the most violated

Algorithm 3 *Cutting plane algorithm for solving* AP-SVM.

Require: $\mathbf{X}, \mathbf{Y}^*, \epsilon$

Initialize the set of active constraints $\mathcal{W} \Leftarrow \emptyset$

$t \Leftarrow 0$

repeat

$t \Leftarrow t + 1$

Learn parameters $\mathbf{w_t}, \xi_\mathbf{t}$ by solving the following convex problem over the set of active constraints $\mathcal{W}$,

$$\begin{aligned} &\underset{\mathbf{w},\xi}{\operatorname{argmin}} \frac{1}{2}||\mathbf{w}||^2 + C\xi \\ &\text{s.t. } \forall \mathbf{Y} \in \mathcal{W} : \{\mathbf{w}^\top \Psi(\mathbf{X}, \mathbf{Y}^*, \{\mathbf{H}_P, \mathbf{H}_N\}) \\ &-\mathbf{w}^\top \Psi(\mathbf{X}, \mathbf{Y}, \{\mathbf{H}_P, \mathbf{H}_N\})\} \geq \Delta(\mathbf{Y}^*, \mathbf{Y}) - \xi. \end{aligned}$$

Find the most violated constraint by solving the following problem,

$$\hat{\mathbf{Y}} = \underset{\mathbf{Y}}{\operatorname{argmax}} \{\Delta(\mathbf{Y}^*, \mathbf{Y}) + \mathbf{w}^\top \Psi(\mathbf{X}, \mathbf{Y}, \{\mathbf{H}_P, \mathbf{H}_N\})\} \tag{2.7}$$

Add the most violated constraint to set of active constraints $\mathcal{W}$.

until

$$\begin{aligned} &\Delta(\mathbf{Y}^*, \hat{\mathbf{Y}}) + \{\mathbf{w_t}^\top \Psi(\mathbf{X}, \hat{\mathbf{Y}}, \{\mathbf{H}_P, \mathbf{H}_N\}) \\ &-\mathbf{w_t}^\top \Psi(\mathbf{X}, \mathbf{Y}^*, \{\mathbf{H}_P, \mathbf{H}_N\})\} \leq \xi_t + \epsilon. \end{aligned}$$

ranking (step 6 of Algorithm 3: solving problem (2.7)). Intuitively, problem (2.7) finds a ranking that differs significantly from the ground-truth ranking in terms of AP but has a high score. Having added the most violated constraint, the cutting plane algorithm updates the parameters by solving a convex quadratic program (step 5 of Algorithm 3). The algorithm stops once no constraint can be found that is violated by more than the desired precision ϵ.

The feasibility of the cutting plane algorithm relies on solving problem (2.7) efficiently. For fixed values of $\mathbf{H}_P$ and $\mathbf{H}_N$ (which is indeed the case for supervised AP-SVM), this can be achieved using the greedy algorithm of Yue *et al.* [64] outlined in Algorithm 4.

Algorithm 4 *Finding the most violated constraint for* AP *loss with known values of additional annotations.*

Require: $\mathbf{X}, \mathbf{Y}^*, \mathbf{H}_P, \mathbf{H}_N, \mathbf{w}$

1: Sort positive inputs $\mathbf{x}_i \in \mathcal{P}$ and negative inputs $\mathbf{x}_j \in \mathcal{N}$ in descending order of the score $\mathbf{w}^\top \Phi_i(\mathbf{h}_i)$ and $\mathbf{w}^\top \Phi_j(\mathbf{h}_j)$ respectively.
2: Initialize ranking $\mathbf{Y}$, s.t. all positive samples are ranked higher than negative samples.
3: **for** $j = 1 \rightarrow |\mathcal{N}|$ **do**
4: Find the position of j^{th} ranked negative which maximizes loss-augmented score, $\Delta(\mathbf{Y}^*, \mathbf{Y}) + \mathbf{w}^\top \Psi(\mathbf{X}, \mathbf{Y}, \{\mathbf{H}_P, \mathbf{H}_N\})$.
5: Update $\mathbf{Y}$ with the best position of the j^{th} ranked negative.
6: **end for**
7: return $\mathbf{Y}$.

Briefly, it starts with the ground-truth ranking, where each positive sample is ranked higher than all the negative samples (step 2 of Algorithm 4). Next, it finds the ranking for each negative sample independently such that the loss-augmented score is maximized (step 3-6 of Algorithm 4). The overall complexity of the above algorithm is $O(n^2)$ (where n is the number of samples). It can be shown to provide the optimal ranking $\mathbf{Y}$, that is, one that maximizes the AP loss augmented score. We refer the interested reader to [64] for details.

2.7 Optimizing Average Precision with Weak Supervision

The main deficiency of supervised learning is that it involves the onerous task of collecting detailed annotations for each training sample. Since detailed annotations are also very expensive, such an approach quickly becomes financially infeasible as the size of the datasets grow. In this work, we consider a more pragmatic setting where the additional annotations $\mathbf{H}_P$ and $\mathbf{H}_N$ are unknown. For example, consider 'jumping' action classification, where each input represents an image that can belong to the positive class or the negative class. In order to learn a ranking framework that can distinguish between 'jumping' and 'not jumping' images, we only require image-level annotations instead of the bounding box of the person in each image.

The convenience of not specifying additional annotations comes at the cost of a more complex machine learning problem. Specifically, we need to deal with two confounding factors: (i) since the best value of the additional annotation $\mathbf{h}_i$ for each positive sample $i \in \mathcal{P}$ is unknown, it needs to be imputed automatically; (ii) since a negative sample remains negative regardless of the value of the additional annotation $\mathbf{h}_j$, we need to consider all possible values of $\mathbf{H}_N$ during parameter estimation. In the 'jumping' action classification example, this implies that (i) we have to identify the bounding box of the jumping person in all the positive images, and (ii) ensure that the scores of the identified jumping person bounding boxes are higher than the scores of all possible bounding boxes in the negative images. In the following subsection, we describe how the standard latent SSVM attempts to resolve these confounding factors in order to optimize the AP loss. This will allow us to identify its shortcomings and correct them with our novel formulation in chapter 3.

2.7.1 Latent SSVM Formulation

Given an input $\mathbf{X}$, the prediction rule of a latent SSVM requires us to maximize the score jointly over the output $\mathbf{Y}$ and the additional annotations $\mathbf{H}$, that is,

$$(\mathbf{Y}_{opt}, \mathbf{H}_{opt}) = \underset{(\mathbf{Y},\mathbf{H})}{\operatorname{argmax}}\, \mathbf{w}^\top \Psi(\mathbf{X}, \mathbf{Y}, \mathbf{H}). \tag{2.8}$$

The parameters $\mathbf{w}$ of a latent SSVM are learned by minimizing a regularized upper bound on the training loss. Specifically, the parameters are obtained by solving the following optimization problem:

$$\begin{aligned} \min_{\mathbf{w}} \quad & \frac{1}{2}||\mathbf{w}||^2 + C\xi, \\ \text{s.t.} \quad & \forall \mathbf{Y}, \mathbf{H} : \max_{\hat{\mathbf{H}}} \{\mathbf{w}^\top \Psi(\mathbf{X}, \mathbf{Y}^*, \hat{\mathbf{H}})\} \\ & -\mathbf{w}^\top \Psi(\mathbf{X}, \mathbf{Y}, \mathbf{H}) \geq \Delta(\mathbf{Y}^*, \mathbf{Y}) - \xi. \end{aligned} \tag{2.9}$$

Intuitively, the above problem introduces a margin between the maximum score corresponding to the ground-truth output and all other pairs of output and additional annotations. Similar to the supervised setting, the desired margin is proportional to the AP loss.

There are three main drawbacks of the standard latent SSVM formulation in the case of AP loss optimization. The first drawback is the prediction rule. While existing latent SVMs for binary loss [22] first obtain the score of each sample by maximising over the additional annotations and then ranking them according to their scores, this does not hold true for existing latent SVMs optimising a general structured loss function such as AP loss. This is specified by problem (2.8), which requires us to simultaneously label the samples as positive or negative (optimize over $\mathbf{Y}$) and find the highest scoring additional annotations for the positive samples and the lowest scoring additional annotations for the negative samples (optimize over $\mathbf{H}$) in order to maximize the score. This is in stark contrast to the prediction rule of existing weakly supervised binary classifiers, which first obtain the score of each sample by maximizing over the additional annotations (regardless of whether they will be labelled as positive or negative), and

then ranking them according to their scores. For example, in action classification, we rank the images according to the highest scoring bounding box of a person in each image. In other words, we never compare the scores of particular choice of additional annotations with a different set of additional annotations. The second drawback is the learning formulation. This is specified by problem (2.9), which provides a very loose upper bound on the AP loss. The third drawback is the optimization. Specifically, to the best of our knowledge, the local optimum solution of problem (2.9) cannot be found efficiently due to the lack of an appropriate cutting plane algorithm. For the details on the difficulty of optimization of latent SSVM, as well as an approximate algorithm used in our experiments, we refer the reader to Appendix A.

Chapter 3

Latent AP-SVM

We now describe a novel latent AP-SVM formulation that overcomes the three drawbacks of the standard latent SSVM framework discussed in the previous section. Specifically, latent AP-SVM uses an intuitive prediction rule, provides a tighter upper bound on the AP loss, and lends itself to efficient optimization.

3.1 Intuitive Prediction

We use a two-step prediction rule. In the first step, we obtain the value of the additional annotations for each sample by maximizing the score, that is,

$$\mathbf{h}_{opt} = \underset{\mathbf{h}}{\operatorname{argmax}}\, \mathbf{w}^{\top}\Phi(\mathbf{h}). \tag{3.1}$$

Next, we obtain the optimal ranking $\mathbf{Y}_{opt}$ for the additional annotations $\mathbf{H}_{opt}$, that is,

$$\mathbf{Y}_{opt} = \underset{\mathbf{Y}}{\operatorname{argmax}}\, \mathbf{w}^{\top}\Psi(\mathbf{X}, \mathbf{Y}, \mathbf{H}_{opt}), \tag{3.2}$$

where $\mathbf{H}_{opt}$ is the set of all the additional annotations obtained by solving problem (3.1) for all samples. Similar to the supervised setting, the optimal ranking is computed by sorting the samples in descending order of their scores. Note that our prediction rule is the same as the ones used in conjunction with the current weakly supervised binary classifiers [2, 22].

3.2 Tighter Bound on the AP Loss

We learn the parameters of latent AP-SVM by solving the following optimization problem:

$$\begin{aligned} \min_{\mathbf{w}} \quad & \frac{1}{2}||\mathbf{w}||^2 + C\xi, \\ \text{s.t.} \quad & \forall \mathbf{Y}, \mathbf{H}_N : \max_{\mathbf{H}_P} \{\mathbf{w}^{\top}\Psi(\mathbf{X}, \mathbf{Y}^*, \{\mathbf{H}_P, \mathbf{H}_N\}) \\ & -\mathbf{w}^{\top}\Psi(\mathbf{X}, \mathbf{Y}, \{\mathbf{H}_P, \mathbf{H}_N\})\} \geq \Delta(\mathbf{Y}^*, \mathbf{Y}) - \xi. \end{aligned} \tag{3.3}$$

Intuitively, the above problem finds the best assignment of values for the additional annotations $\mathbf{H}_P$ of the positive samples such that the score for the correct ranking (which places all the positive samples above the negative samples) is higher than the score for an incorrect ranking, regardless of the choice of the additional annotations $\mathbf{H}_N$ of the negative samples.

It is worth noting the significant difference between the optimization corresponding to latent AP-SVM and the standard latent SSVM. Specifically, in the constraints of problem (2.9), the values of the additional annotations for a correct and incorrect ranking are independent of each other. In contrast, the constraints of problem (3.3) are specified using the same values of the additional annotations. The following proposition provides a sound theoretical justification for preferring problem (3.3) over problem (2.9).

Proposition 1. *The latent* AP-SVM *formulation provides a tighter upper bound on the* AP *loss compared to the standard latent* SSVM *formulation*

Before deriving the proof for our proposition that the latent AP-SVM formulation provides a tighter upper bound on the AP loss compared to the standard latent SSVM formulation, it would be helpful to prove the following lemma.

Lemma 1. *For a given value of the ranking matrix* $\mathbf{Y}$ *and the additional annotations of the negative samples* $\mathbf{H}_N$*, consider the following optimization problem:*

$$\mathbf{H}_P^* = \operatorname*{argmin}_{\mathbf{H}_P} \mathbf{w}^\top \Psi(\mathbf{X}, \mathbf{Y}, \{\mathbf{H}_P, \mathbf{H}_N\}) - \mathbf{w}^\top \Psi(\mathbf{X}, \mathbf{Y}^*, \{\mathbf{H}_P, \mathbf{H}_N\}), \tag{3.4}$$

where $\mathbf{Y}^*$ *is the optimal ranking matrix. The above optimization problem can be solved optimally as follows regardless of the choice of* $\mathbf{Y}$ *and* $\mathbf{H}_N$*:*

$$\mathbf{h}_i^* = \operatorname*{argmax}_{\mathbf{h}_i} \mathbf{w}^\top \Phi_i(\mathbf{h}_i). \tag{3.5}$$

Proof. We use the following shorthand:

$$s_i(\mathbf{h}_i) = \frac{1}{|\mathcal{P}||\mathcal{N}|} \mathbf{w}^\top \Phi_j(\mathbf{h}_j),$$
$$s_j(\mathbf{h}_j) = \frac{1}{|\mathcal{P}||\mathcal{N}|} \mathbf{w}^\top \Phi_j(\mathbf{h}_j). \tag{3.6}$$

Using the above shorthand, we can rewrite the expression in the RHS of problem (3.4) as follows:

$$\begin{aligned} & \mathbf{w}^\top \Psi(\mathbf{X}, \mathbf{Y}, \{\mathbf{H}_P, \mathbf{H}_N\}) - \\ & \mathbf{w}^\top \Psi(\mathbf{X}, \mathbf{Y}^*, \{\mathbf{H}_P, \mathbf{H}_N\}) \\ = \quad & \sum_{i \in \mathcal{P}} s_i(\mathbf{h}_i) \left(\sum_{j \in \mathcal{N}} (Y_{ij} - Y^*_{ij}) \right) + \\ & \sum_{j \in \mathcal{N}} s_j(\mathbf{h}_j) \left(\sum_{i \in \mathcal{P}} (Y^*_{ij} - Y_{ij}) \right). \end{aligned} \tag{3.7}$$

By definition $\sum_{j \in \mathcal{N}} Y^*_{ij} \geq \sum_{j \in \mathcal{N}} Y_{ij}$ since $Y^*_{ij} = 1$ for all $i \in \mathcal{P}$ and $j \in \mathcal{N}$. Thus, the coefficient of the score term $s_i(\mathbf{h}_i)$ is negative for all positive samples. Therefore, in order to minimize problem (3.4) over all possible choices of $\mathbf{H}_P$, we should maximize the score $s_i(\mathbf{h}_i)$ for each positive sample. This is exactly the solution proposed in equation (3.5), which completes the proof. □

We are now ready to prove the following proposition 1.

Proof. We compare the optimization problems corresponding to the standard latent SSVM formulation and the latent AP-SVM formulation. The parameters of latent SSVM are estimated by solving the following problem:

$$\begin{aligned} & \min_{\mathbf{w}} \frac{1}{2} ||\mathbf{w}||^2 + C\xi, \\ & \text{s.t. } \forall \mathbf{Y}, \hat{\mathbf{H}}_N, \mathbf{H}_P, \mathbf{H}_N : \\ & \max_{\hat{\mathbf{H}}_P} \{\mathbf{w}^\top \Psi(\mathbf{X}, \mathbf{Y}^*, \hat{\mathbf{H}}_P, \hat{\mathbf{H}}_N)\} \\ & -\mathbf{w}^\top \Psi(\mathbf{X}, \mathbf{Y}, \mathbf{H}_P, \mathbf{H}_N) \geq \Delta(\mathbf{Y}^*, \mathbf{Y}) - \xi. \end{aligned} \tag{3.8}$$

The parameters of latent AP-SVM are estimated by solving the following problem:

$$\begin{aligned} \min_{\mathbf{w}} \quad & \frac{1}{2} ||\mathbf{w}||^2 + C\xi, \\ \text{s.t.} \quad & \forall \mathbf{Y}, \mathbf{H}_N : \max_{\mathbf{H}_P} \{\mathbf{w}^\top \Psi(\mathbf{X}, \mathbf{Y}^*, \{\mathbf{H}_P, \mathbf{H}_N\}) \\ & -\mathbf{w}^\top \Psi(\mathbf{X}, \mathbf{Y}, \{\mathbf{H}_P, \mathbf{H}_N\})\} \geq \Delta(\mathbf{Y}^*, \mathbf{Y}) - \xi. \end{aligned} \tag{3.9}$$

Note that the constraints specified in problem (3.9) are a subset of the constraints specified in problem (3.8). In other words, the feasible region of problem (3.9) is a superset of the feasible region of problem (3.8). It follows that the optimal objective function value of problem (3.9) is guaranteed to be less than or equal to the optimal objective function value of problem (3.8). In other words, minimizing

problem (3.9) provides a quantity that is guaranteed to be less than or equal to the regularized upper bound on the AP loss that is optimized by the standard latent SSVM. Thus, in order to prove proposition 1 it is sufficient to show that problem (3.9) minimizes a valid regularized upper bound on the AP loss.

In order to show that problem (3.9) minimizes a regularized upper bound on the AP loss, we introduce the following notation. Given a set of parameters $\mathbf{w}$ for the latent AP-SVM formulation, we denote the predicted additional annotations by $H(\mathbf{w}) = (\mathbf{H}_P(\mathbf{w}), \mathbf{H}_N(\mathbf{w}))$. Recall that, for each sample $\mathbf{x}_i$, the additional annotation $\mathbf{h}_i(\mathbf{w})$ is predicted using the following rule:

$$\mathbf{h}_i(\mathbf{w}) = \operatorname*{argmax}_{\mathbf{h}_i} \mathbf{w}^\top \phi_i(\mathbf{h}_i). \tag{3.10}$$

Similarly, we denote the predicted output (that is, the ranking matrix) as $\mathbf{Y}(\mathbf{w})$, which is obtained using the following rule:

$$\mathbf{Y}(\mathbf{w}) = \operatorname*{argmax}_{\mathbf{Y}} \mathbf{w}^\top \Psi(\mathbf{X}, \mathbf{Y}, \mathbf{H}(\mathbf{w})). \tag{3.11}$$

Using the above notation, the AP loss over the training dataset $\Delta(\mathbf{Y}^*, \mathbf{Y}(\mathbf{w}))$ can be upper bounded as follows:

$$\begin{aligned}
& \Delta(\mathbf{Y}^*, \mathbf{Y}(\mathbf{w})) \\
= \quad & \mathbf{w}^\top \Psi(\mathbf{X}, \mathbf{Y}(\mathbf{w}), \{\mathbf{H}_P(\mathbf{w}), \mathbf{H}_N(\mathbf{w})\}) \\
& + \Delta(\mathbf{Y}^*, \mathbf{Y}(\mathbf{w})) \\
& - \mathbf{w}^\top \Psi(\mathbf{X}, \mathbf{Y}(\mathbf{w}), \{\mathbf{H}_P(\mathbf{w}), \mathbf{H}_N(\mathbf{w})\}) && (3.12) \\
\leq \quad & \mathbf{w}^\top \Psi(\mathbf{X}, \mathbf{Y}(\mathbf{w}), \{\mathbf{H}_P(\mathbf{w}), \mathbf{H}_N(\mathbf{w})\}) \\
& + \Delta(\mathbf{Y}^*, \mathbf{Y}(\mathbf{w})) \\
& - \mathbf{w}^\top \Psi(\mathbf{X}, \mathbf{Y}^*, \{\mathbf{H}_P(\mathbf{w}), \mathbf{H}_N(\mathbf{w})\}) && (3.13) \\
\leq \quad & \max_{\mathbf{Y}, \mathbf{H}_N} \Big\{ \mathbf{w}^\top \Psi(\mathbf{X}, \mathbf{Y}, \{\mathbf{H}_P(\mathbf{w}), \mathbf{H}_N\}) \\
& + \Delta(\mathbf{Y}^*, \mathbf{Y}) \\
& - \mathbf{w}^\top \Psi(\mathbf{X}, \mathbf{Y}^*, \{\mathbf{H}_P(\mathbf{w}), \mathbf{H}_N\}) \Big\}. && (3.14)
\end{aligned}$$

Expression (3.13) follows from the fact that the score for optimal ranking $\mathbf{Y}^*$ must be less than or equal to the score for the predicted ranking $\mathbf{Y}(\mathbf{w})$ (since the predicted ranking is obtained by maximizing the score as shown in equation (3.11)). Expression (3.14) follows from the fact that instead of using the prediction, we maximize over all possible rankings and additional annotations of the negative samples.

Note that expression (3.14) still contains the predicted value of the additional annotations of the positive samples. In order to further simplify the upper bound, we make use of lemma (1), which implies that for any value of $\mathbf{Y}$ and $\mathbf{H}_N$, the following holds true:

$$\mathbf{H}_P(\mathbf{w}) = \underset{\mathbf{H}_P}{\operatorname{argmin}}\, \mathbf{w}^\top \Psi(\mathbf{X}, \mathbf{Y}, \{\mathbf{H}_P, \mathbf{H}_N\}) - \mathbf{w}^\top \Psi(\mathbf{X}, \mathbf{Y}^*, \{\mathbf{H}_P, \mathbf{H}_N\}), \tag{3.15}$$

since the predicted value of $\mathbf{H}_P(\mathbf{w})$ is exactly equal to that specified by equation (3.5). Therefore, it follows that

$$\mathbf{H}_P(\mathbf{w}) = \underset{\mathbf{H}_P}{\operatorname{argmin}}\, \mathbf{w}^\top \Psi(\mathbf{X}, \mathbf{Y}, \{\mathbf{H}_P, \mathbf{H}_N\}) - \mathbf{w}^\top \Psi(\mathbf{X}, \mathbf{Y}^*, \{\mathbf{H}_P, \mathbf{H}_N\}) + \Delta(\mathbf{Y}^*, \mathbf{Y}), \tag{3.16}$$

since $\Delta(\mathbf{Y}^*, \mathbf{Y})$ is independent of $\mathbf{H}_P$. For example, consider the action classification task, where $\mathbf{Y}$ denotes the ranking of the images and $\mathbf{H}_P$ is the bounding boxes in the positive samples. Regardless of the choice of the positive samples, the loss is computed based on the ranking $\mathbf{Y}$ and not the bounding boxes $\mathbf{H}_P$. Using the fact that the above equation holds true for any choice of $\mathbf{Y}$ and $\mathbf{H}_N$, expression (3.14) can be simplified as

$$\min_{\mathbf{H}_P} \max_{\mathbf{Y}, \mathbf{H}_N} \Big\{ \mathbf{w}^\top \Psi(\mathbf{X}, \mathbf{Y}, \{\mathbf{H}_P, \mathbf{H}_N\}) + \Delta(\mathbf{Y}^*, \mathbf{Y}) - \mathbf{w}^\top \Psi(\mathbf{X}, \mathbf{Y}^*, \{\mathbf{H}_P, \mathbf{H}_N\}) \Big\}. \tag{3.17}$$

We note that this is exactly the value of the slack variable ξ in problem (3.9). This proves that problem (3.9) minimizes a valid regularized upper bound of the AP loss, which in turn proves the proposition. □

3.3 Efficient Optimization

The local minimum or saddle point solution of problem (3.3) can be obtained using the CCCP algorithm [65], as described in Algorithm 5. The algorithm involves two main steps. In the first step (step 3 of Algorithm 5), it imputes the best additional annotations $\mathbf{H}_P$ of the positive samples given the current estimate of the parameters. In the second step (step 4 of Algorithm 5), given the imputed values of $\mathbf{H}_P$, CCCP updates the parameters by solving the resulting convex optimization problem. We discuss both these steps in detail below.

Algorithm 5 *The* CCCP *algorithm for parameter estimation of latent* AP-SVM.

Require: $\mathbf{X}, \mathbf{Y}^*, \mathbf{w}_0, \epsilon$

$t \Leftarrow 0$

repeat

  For the current set of parameters $\mathbf{w}_t$, obtain the value of the latent variables $\mathbf{H}_P^*$ that minimizes the objective function value of problem (3.3).

  Update $\mathbf{w}_{t+1}$ by fixing the latent variables to $\mathbf{H}_P^*$ and solving the resulting convex problem.

  $t \leftarrow t + 1$

until Objective function cannot be decreased below tolerance ϵ

Imputing the Additional Annotations.

For a given parameter $\mathbf{w}$, we need to obtain the values of the additional annotations $\mathbf{H}_P$ for the positive samples such that it minimizes the objective function of problem (3.3). Since $\mathbf{w}$ is fixed, the first term of the objective function (that is, the squared ℓ_2 norm of $\mathbf{w}$) cannot be modified. Instead, we need to minimize the slack ξ, which is equivalent to solving the following problem:

$$\min_{\mathbf{H}_P} \max_{\mathbf{Y}, \mathbf{H}_N} \{\Delta(\mathbf{Y}^*, \mathbf{Y}) - \mathbf{w}^\top \Psi(\mathbf{X}, \mathbf{Y}^*, \{\mathbf{H}_P, \mathbf{H}_N\}) + \mathbf{w}^\top \Psi(\mathbf{X}, \mathbf{Y}, \{\mathbf{H}_P, \mathbf{H}_N\})\}. \tag{3.18}$$

We refer to the above problem as output-consistent inference (since it fills in the missing information under the constraint that it is consistent with the output, that is, the optimal ranking). Although problem (3.18) contains $\mathbf{Y}$ and $\mathbf{H}_N$, the following proposition shows that it can be optimized easily with respect to $\mathbf{H}_P$.

Proposition 2. *Problem (3.18) can be solved efficiently by independently choosing the latent variable for each positive sample using the following criterion:*

$$\mathbf{h}_i^* = \operatorname*{argmax}_{\mathbf{h}_i} \mathbf{w}^\top \Phi_i(\mathbf{h}_i), \forall i \in \mathcal{P} \tag{3.19}$$

Proof. For any given value of $\mathbf{Y}$ and $\mathbf{H}_N$, lemma (1) proves the following:

$$\mathbf{H}_P^* = \operatorname*{argmin}_{\mathbf{H}_P} \mathbf{w}^\top \Psi(\mathbf{X}, \mathbf{Y}, \{\mathbf{H}_P, \mathbf{H}_N\}) - \mathbf{w}^\top \Psi(\mathbf{X}, \mathbf{Y}^*, \{\mathbf{H}_P, \mathbf{H}_N\}), \tag{3.20}$$

where $\mathbf{H}_P^*$ is computed as suggested in the above proposition, that is, by maximizing the scores of positive samples independently over their choice of additional annotations. Therefore, it follows that

$$\mathbf{H}_P^* = \operatorname*{argmin}_{\mathbf{H}_P} \mathbf{w}^\top \Psi(\mathbf{X}, \mathbf{Y}, \{\mathbf{H}_P, \mathbf{H}_N\}) - \mathbf{w}^\top \Psi(\mathbf{X}, \mathbf{Y}^*, \{\mathbf{H}_P, \mathbf{H}_N\}) + \Delta(\mathbf{Y}^*, \mathbf{Y}), \tag{3.21}$$

since $\Delta(\mathbf{Y}^*, \mathbf{Y})$ is independent of $\mathbf{H}_P$. For example, consider the action classification task, where $\mathbf{Y}$ denotes the ranking of the images and $\mathbf{H}_P$ is the bounding boxes in the positive samples. Regardless of the choice of the positive samples, the loss is computed based on the ranking $\mathbf{Y}$ and not the bounding boxes $\mathbf{H}_P$. Using the fact that the above equation holds true for any choice of $\mathbf{Y}$ and $\mathbf{H}_N$, it follows that

$$\begin{aligned}\mathbf{H}_P^* = \quad & \underset{\mathbf{H}_P}{\operatorname{argmin}} \max_{\mathbf{Y},\mathbf{H}_N} \Big\{ \mathbf{w}^\top \Psi(\mathbf{X}, \mathbf{Y}, \{\mathbf{H}_P, \mathbf{H}_N\}) \\ & + \Delta(\mathbf{Y}^*, \mathbf{Y}) \\ & - \mathbf{w}^\top \Psi(\mathbf{X}, \mathbf{Y}^*, \{\mathbf{H}_P, \mathbf{H}_N\}) \Big\}. \end{aligned} \tag{3.22}$$

This proves the proposition. □

Updating the Parameters.

Given the imputed latent variables $\mathbf{H}_P^*$, the parameters are updated by solving the following convex problem:

$$\begin{aligned} & \min_{\mathbf{w}} \frac{1}{2}||\mathbf{w}||^2 + C\xi, \\ & \mathbf{w}^\top \Psi(\mathbf{X}, \mathbf{Y}^*, \{\mathbf{H}_P^*, \mathbf{H}_N\}) - \mathbf{w}^\top \Psi(\mathbf{X}, \mathbf{Y}, \{\mathbf{H}_P^*, \mathbf{H}_N\}) \\ & \geq \Delta(\mathbf{Y}^*, \mathbf{Y}) - \xi, \forall \mathbf{Y}, \mathbf{H}_N. \end{aligned} \tag{3.23}$$

Similar to supervised AP-SVM, the above problem can be solved using a cutting plane algorithm. The computational feasibility of the cutting plane algorithm relies on being able to efficiently compute the most violated constraint. In our case, the most violated constraint is found by solving the following problem:

$$\begin{aligned} \hat{\mathbf{Y}}, \hat{\mathbf{H}}_N = \underset{\mathbf{Y},\mathbf{H}_N}{\operatorname{argmax}} \{ & \mathbf{w}^\top \Psi(\mathbf{X}, \mathbf{Y}, \{\mathbf{H}_P^*, \mathbf{H}_N\}) \\ & - \mathbf{w}^\top \Psi(\mathbf{X}, \mathbf{Y}^*, \{\mathbf{H}_P^*, \mathbf{H}_N\}) + \Delta(\mathbf{Y}^*, \mathbf{Y}) \}. \end{aligned} \tag{3.24}$$

We refer to the above problem as loss-augmented inference (since it augments the score of the ranking with its AP loss). Note that, in contrast to supervised AP-SVM, we not only need to optimize over the ranking $\mathbf{Y}$, but also the variables $\mathbf{H}_N$. The following proposition allows us to perform the joint optimization efficiently. Here, we use the following shorthand:

$$s_j(\mathbf{h}_j) = \frac{1}{|\mathcal{P}||\mathcal{N}|} \mathbf{w}^\top \Phi_j(\mathbf{h}_j). \tag{3.25}$$

Proposition 3. *Problem (3.24) can be solved by first maximizing over* $\mathbf{H}_N$ *using the criterion,* $\mathbf{h}_j = \operatorname{argmax}_{\mathbf{h}_j} s_j(\mathbf{h}_j)$.

Proof. Problem (3.24) can be written as

$$\underset{\mathbf{Y},\mathbf{H}_N}{\operatorname{argmax}}\{\sum_{i\in\mathcal{P}}\sum_{j\in\mathcal{N}}(\mathbf{Y}^*_{ij}-\mathbf{Y}_{ij})s_j(\mathbf{h}_j) + \sum_{i\in\mathcal{P}}\sum_{j\in\mathcal{N}}(\mathbf{Y}_{ij}-\mathbf{Y}^*_{ij})s_i(\mathbf{h}^*_i)+\Delta(\mathbf{Y}^*,\mathbf{Y})\}. \quad (3.26)$$

By definition, $\mathbf{Y}^*_{ij}=1$ for all $i\in\mathcal{P}, j\in\mathcal{N}$. Thus, the coefficient of the score term $s_j(\mathbf{h}_j)$ is non-negative for all negative samples regardless of the value of $\mathbf{Y}$. Therefore, problem (3.24) can be maximized first over $\mathbf{H}_N$ by maximizing the score of each negative sample independently. This proves the proposition. □

Using Proposition 3, problem (3.24) can be solved in two steps. In the first step we maximize the loss-augmented score over $\mathbf{H}_N$ by maximizing the score of each negative sample independently. The second step is to maximize the loss-augmented score over $\mathbf{Y}$, which is achieved using the optimal greedy algorithm described in Algorithm 4.

Chapter 4

Experiments

The previous section shows the theoretical benefit of latent AP-SVM over the standard latent SSVM formulation, namely that it minimizes a tighter upper bound on the AP loss and allows for efficient inference, while using an intuitive prediction rule. We now show that the theoretical benefits translate to improved empirical performance using three important and challenging problems in computer vision.

4.0.1 Action Classification

Dataset. We use the PASCAL VOC 2011 [19] 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 persons in the image together with their action class; and 2422 'test' images for which we are only provided with the person bounding boxes.

Recall that our main hypothesis is that the challenging nature of weakly supervised learning makes it essential to use the right loss function during training. In order to test this hypothesis, we use the 'trainval' images to create five types of datasets that vary in their level of supervision. Specifically, each type of dataset provides the ground-truth additional annotations[1] for S percent of the positive and the negative samples, where $S \in \{0, 25, 50, 75, 100\}$. The additional annotations for the remaining $100 - S$ percent of the samples are treated as latent variables. The putative values of each latent variable are restricted to the top $T = 20$ boxes obtained by a standard person detector [21]. During testing, we use the learned parameters to rank the given person bounding boxes in the 'test' dataset. The performance is measured by submitting the scores of all the bounding boxes to the PASCAL VOC evaluation server.

Features. Given a bounding box $\mathbf{h}_i$ of the image $\mathbf{x}_i$, we use the standard poselet-based feature vector [39] to specify $\Phi_i(\mathbf{h}_i)$. It consists of 2400 activation scores of action-specific poselets and 4 object activation scores. In addition, we use the score of the person detector [21], which results in a 2405 dimensional feature vector.

[1] Additional annotation provided is the bounding-box obtained by a standard person detector overlapping most with the ground-truth bounding box in PASCAL VOC.

Methods. We compare our latent AP-SVM formulation with the baseline latent SVM that is commonly used in computer vision. Latent SVM consists of two hyperparameters: (i) C, the trade-off between the regularization and the loss; and (ii) J, the relative weight of the positive samples. In order to further strengthen the baseline, we add robustness to outliers using a further hyperparameter c. Specifically, we prevent the classifier from considering the most confusing $c\%$ bounding boxes in the negative samples under the constraint that at least one bounding box is used per negative image. We obtain the best settings of the hyperparameters via a 5-fold cross validation, where the 'trainval' set is split into 1940 training images and 484 validation images. We consider the following putative values: $C \in \{10^{-3}, 10^{-2}, \ldots, 10^{4}\}$, $J \in \frac{|\mathcal{P}|+|\mathcal{N}|}{|\mathcal{P}|} \times \{10^{-4}, 10^{-3}, \ldots, 10^{1}\}$ and $c \in \{0, 0.1, \ldots, 0.9\}$ (note that, when $c = 0$, the resulting baseline is the standard latent SVM without robustness). In addition, we also compare the performance of our latent AP-SVM with latent SSVM. For the latent AP-SVM and latent SSVM, we only need to specify a single hyperparameter C, whose value is also obtained via 5-fold cross-validation. In order to mitigate the effects of initialization, we use 5 random seeds and choose the one that provides the minimum objective value for each method independently.

Complexity. The running time of weakly supervised learning algorithms is dominated by computation of the most violated constraint. Empirically, we found that computation of most violated constraint in latent AP-SVM is around 5 times slower and 100 times faster compared to latent SVM and latent SSVM respectively. However, latent AP-SVM does not require an extra hyperparameter J (the relative weight of the positive samples). Hence, the time taken for crossvalidation by both latent SVM and latent AP-SVM is comparable.

Results. Figure 4.1 shows the best mean AP value over all 10 action classes obtained during 5-fold cross validation. Note that as the amount of supervision decreases, the gap between our method and the two baselines steadily increases. In the fully supervised setting, that is, $S = 100$, latent AP-SVM provides statistically significant improvements over latent SVM for only 4 out of 10 classes (using paired t-test with p-value less than 0.05), with an overall improvement of less than 3%. Note that, for fully supervised datasets, both latent AP-SVM and latent SSVM are equivalent to the AP-SVM, and hence provide the same results. However, in the more interesting weakly supervised setting, that is, $S = 0$, latent AP-SVM provides statistically significant improvements over latent SVM for 6 out of 10 classes, and an overall improvement of more than 5%. By cross-validating c, instead of choosing the default value of $c = 0$ (standard latent binary SVM), we improve the performance of the baseline by 2.4%. Latent AP-SVM also provides statistically significant improvements over latent SSVM for 7 out of 10 classes and an overall improvement of more than 4%.

Table 4.1 shows the comparison of our latent AP-SVM with latent SVM and latent SSVM on the test set. Note that we use 5 different random seeds for each method. The hyperparameters are set using 5-fold cross-validation. Latent AP-SVM performs better than latent SVM for all 10 classes with significant increase in performance for 4 classes. Overall, we get an improvement of 5.1% on the test performance

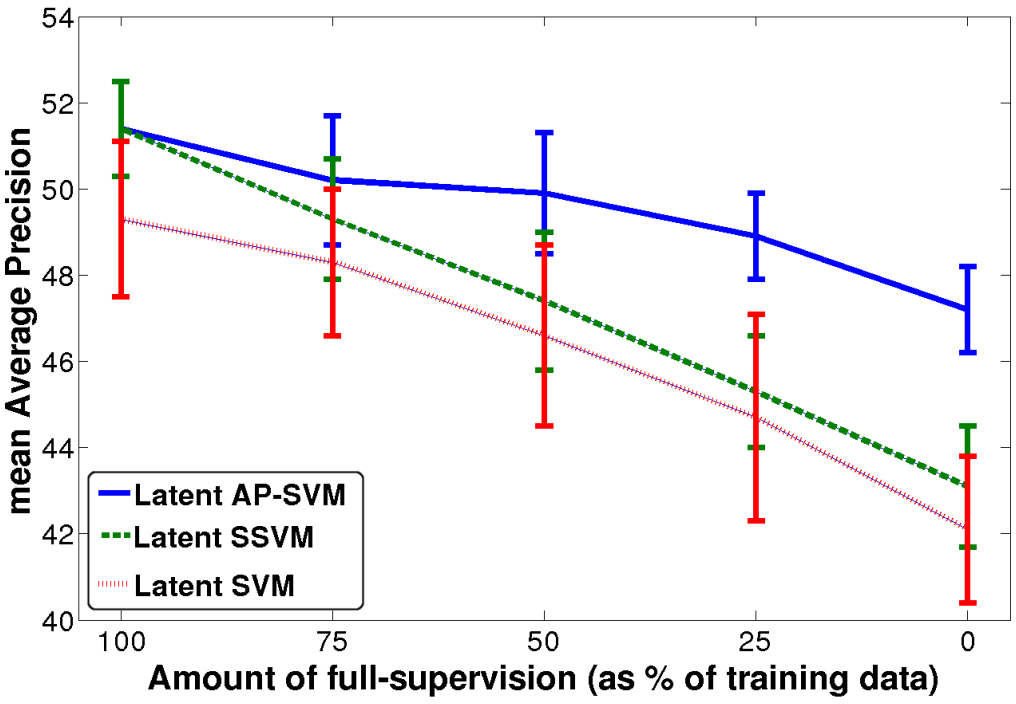

Figure 4.1: *The best mean average precision over all 10 action classes obtained during 5-fold cross validation. The x-axis corresponds to the amount of supervision provided. The y axis corresponds to the mean average precision. As the amount of supervision decreases, the gap in the performance of latent* AP-SVM *and the baseline methods increases, thereby illustrating the importance of using the correct loss function and the correct learning formulation for weakly supervised learning.*

compared to latent SVM. Similarly, latent AP-SVM performs better than latent SSVM for 8 out of 10 classes. Overall, we get an improvement of 3.7% on the test performance compared to latent SSVM.

Method	Jump	Use phone	Play inst.	Read	Ride bike	Ride horse	Run	Take photo	Use comp.	Walk	Overall
LAP-SVM	45.7	30.5	34.0	21.1	75.5	74.9	76.0	15.7	24.6	47.5	44.6
LSSVM	37.6	26.5	33.9	22.5	71.2	66.7	66.8	17.4	21.9	44.8	40.9
LSVM	36.9	28.0	32.2	20.6	65.3	68.2	63.5	13.4	21.6	45.7	39.5

Table 4.1: *The average precision of latent* AP-SVM *and the baseline latent* SVM *and latent* SSVM *methods under weak supervision. The training is performed over the entire 'trainval' dataset with $S = 0$ 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. The last column ('Overall') shows the mean average precision over all ten action classes.*

4.0.2 Character Recognition in Natural Images

Dataset. We use the IIIT 5K-WORD [41] scene text dataset, which consists of 5000 cropped word images from scene texts and born-digital images, which are divided into 2000 'trainval' images and 3000 'test' images. Each image is annotated with the corresponding word, that is, a string where each character is an upper case letter ('A' to 'Z'), a lower case letter ('a' to 'z'), or a number ('0' to '9'). In addition, the dataset also provides the bounding boxes for each character of the word, which we discard during learning. Instead, we treat the bounding box of the characters as latent variables whose putative values are restricted to $T = 20$ boxes obtained by a standard character detector [15]. Using this dataset, we perform ranking for the 22 classes that contain at least 150 samples in the 'trainval' dataset.

Features. Given a character bounding box $\mathbf{h}_i$ of the word image $\mathbf{x}_i$, we use the histogram of oriented gradients (HOG) [15] features to specify $\Phi_i(\mathbf{h}_i)$. The HOG features are computed by resizing the bounding box to 48×48 pixels.

Methods. We compare our latent AP-SVM formulation with the baseline latent SVM and latent SSVM with AP loss. Similar to the action classification experiments, we set the hyperparameters of all the methods using 5-fold crossvalidation by splitting the 'trainval' dataset into 80%/20% folds. In order to avoid errors due to initialization, we use 3 different random seeds for each method and pick the one corresponding to the minimum objective value.

Results. Figure 4.2 and 4.4 show the best AP values for all the classes where the performance of latent AP-SVM is statistically different from that of latent SVM and latent SSVM respectively (using paired t-test with p-value less than 0.05). Latent SVM and latent SSVM provides statistically significant improvements

over latent AP-SVM for only 1 class. In contrast, latent AP-SVM improves the performance for 4 and 3 classes over latent SVM and latent SSVM respectively. In terms of the mean AP value, latent AP-SVM provides an improvement of 3.2% and 2.7% over latent SVM and latent SSVM respectively.

Figure 4.3 and 4.5 show the AP values for the statistically significant characters on the 'test' set. Similar to the cross-validation results, latent AP-SVM outperforms latent SVM and and latent SSVM for 4 and 3 classes respectively. In terms of the mean AP value on the 'test' set, latent AP-SVM provides an improvement of 2.7% and 2.5% over latent SVM and latent SSVM respectively.

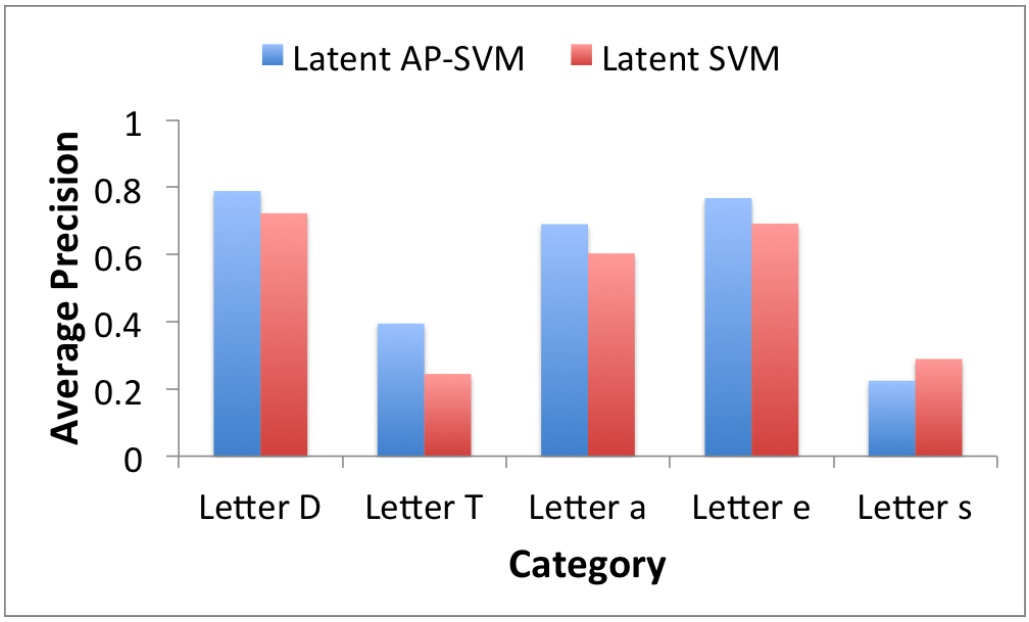

Figure 4.2: *The best average precision values for the* 5 *statistically significant character classes obtained during 5-fold cross validation on the 'trainval' set of the* IIIT 5K-WORD *dataset. The* x*-axis corresponds to the characters. The* y *axis corresponds to the average precision. Latent* AP-SVM *provides statistically significant improvements over latent* SVM *for* 4 *out of the* 5 *characters.*

The detailed results over all 22 classes during cross validation and testing are provided in Figure 4.6 and Figure 4.7 respectively.

4.0.3 Object Detection

Dataset. We use the PASCAL VOC 2007 [20] object detection dataset, which consists of a total of 9963 images. The dataset is divided into a 'trainval' set of 5011 images and a 'test' set of 4952 images. All the images are labeled to indicate the presence or absence of the instances of 20 different object categories. In addition, we are also provided with tight bounding boxes around the object, which we ignore during training and testing. Instead, we treat the location of the objects as a latent variable. In order to reduce

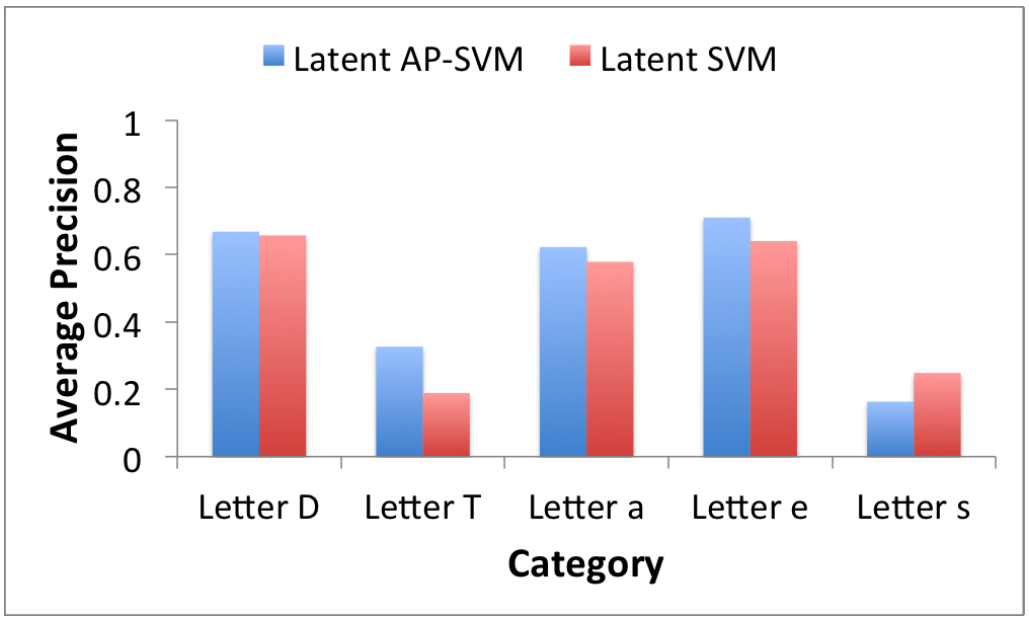

Figure 4.3: *The average precision values for the* 5 *statistically significant characters obtained on the 'test' set of the* IIIT 5K-WORD *dataset. The* x*-axis corresponds to the character categories. The* y *axis corresponds to the average precision.*

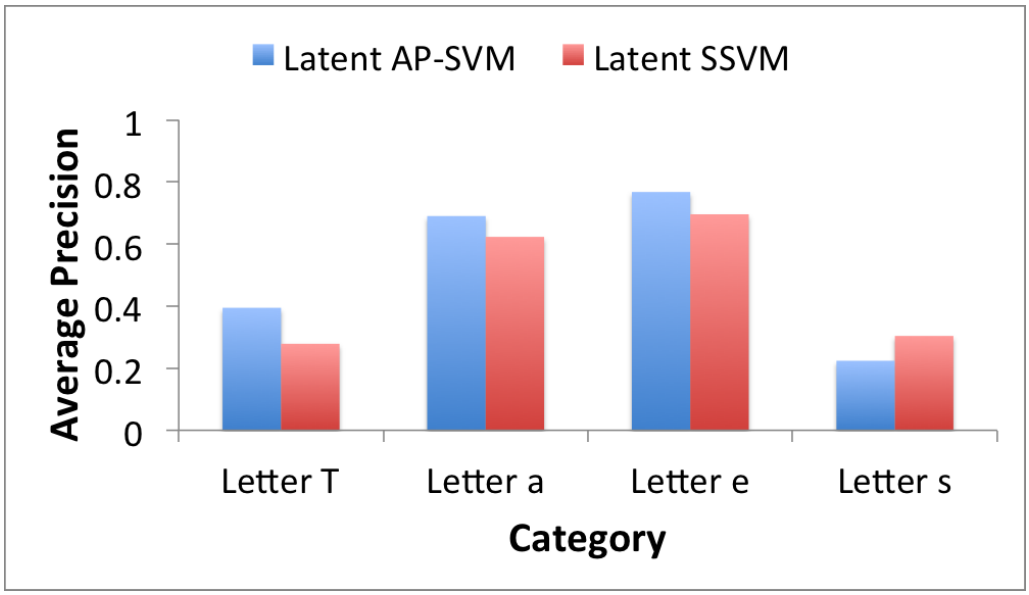

Figure 4.4: *The best average precision values for the* 4 *statistically significant character classes obtained during 5-fold cross validation on the 'trainval' set of the* IIIT 5K-WORD *dataset. The* x*-axis corresponds to the characters. The* y *axis corresponds to the average precision. Latent* AP-SVM *provides statistically significant improvements over latent* SSVM *for* 3 *out of the* 4 *characters.*

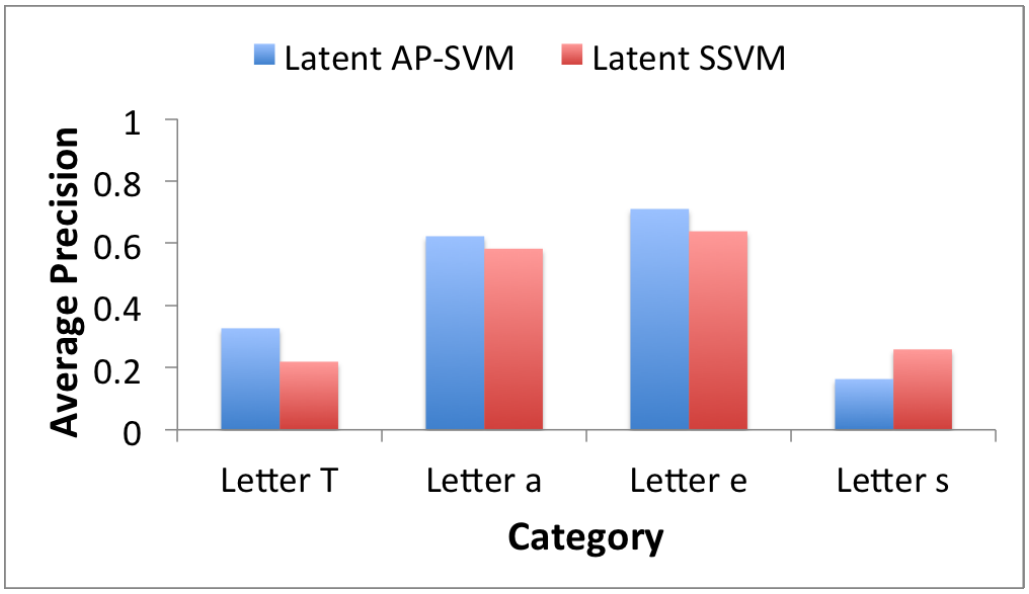

Figure 4.5: *The average precision values for the* 4 *statistically significant characters obtained on the 'test' set of the* IIIT 5K-WORD *dataset. The* x*-axis corresponds to the character categories. The* y *axis corresponds to the average precision.*

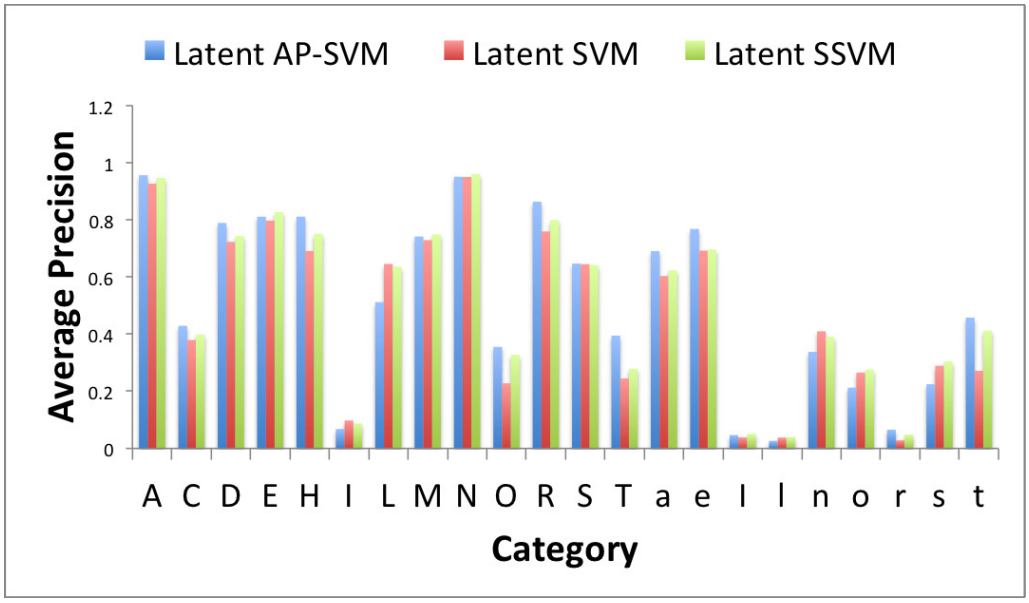

Figure 4.6: *The best average precision values for all* 22 *character classes obtained during 5-fold cross validation on the 'trainval' set of the* IIIT 5K-WORD *dataset. The* x*-axis corresponds to the characters. The* y *axis corresponds to the average precision.*

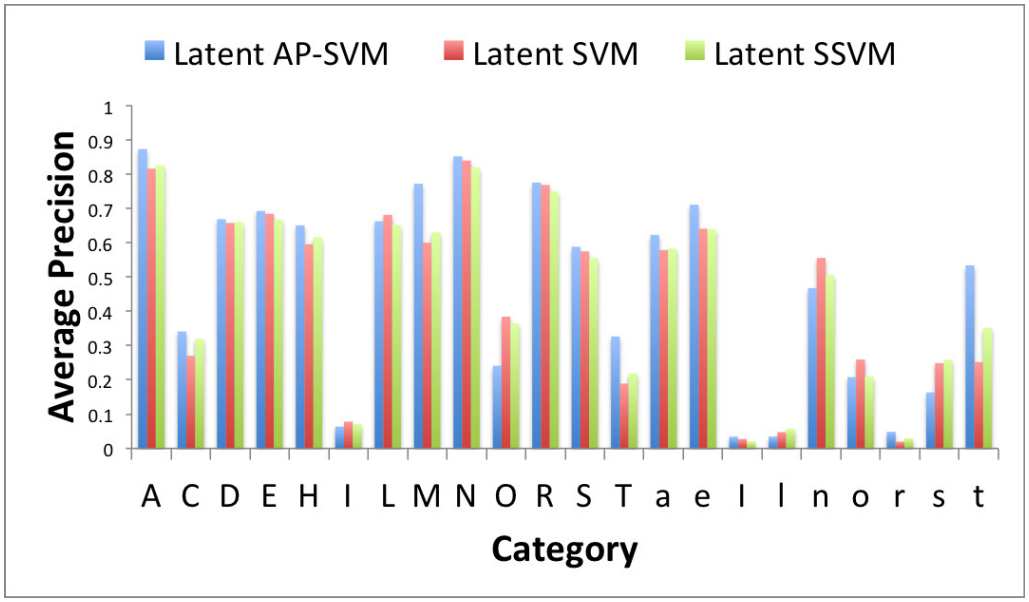

Figure 4.7: *The average precision values for all* 22 *characters obtained on the 'test' set of the* IIIT 5K-WORD *dataset. The* x*-axis corresponds to the character categories. The* y *axis corresponds to the average precision.*

the latent variable space, we use the selective-search algorithm [51] in its fast mode, which generates an average of 2000 candidate windows per image.

Features. For each of the candidate windows, we use a feature representation that is extracted from a trained Convolutional Neural Network (CNN). Specifically, we pass the image as input to the CNN and use the activation vector of the penultimate layer of the CNN as the feature vector. Inspired by the work of Girshick *et al.* [24], we use the CNN that is trained on the ImageNet dataset [17], by rescaling each candidate window to a fixed size of 224×224. The length of the resulting feature vector is 4096.

Methods. We compare our latent AP-SVM formulation with latent SVM for 20 detection tasks, corresponding to the 20 object categories. Note that this experiment places high computational demands due to the size of the dataset (5011 'trainval' images), as well as the size of the latent space (2000 candidate windows per image). Hence, we were unable to run the expensive latent SSVM baseline. For both latent AP-SVM and latent SVM, we determine the value of the hyperparameters using 5-fold cross-validation. During testing, we evaluate each candidate window generated by selective search, and use non-maxima suppression to prune highly overlapping detections.

Object category	latent SVM	latent AP-SVM
Aeroplane	46.60	48.18
Bicycle	48.53	61.45
Bird	33.31	36.73
Boat	15.23	19.66
Bottle	6.10	1.01
Bus	37.01	49.51
Car	61.28	66.78
Cat	38.12	40.77
Chair	2.71	3.23
Cow	21.06	38.52
Dining-table	14.20	39.53
Dog	33.55	36.25
Horse	46.14	53.86
Motorbike	29.97	34.81
Person	29.58	30.41
Potted-plant	21.27	23.03
Sheep	11.65	32.20
Sofa	36.66	42.03
Train	29.71	37.10
TV-monitor	27.31	37.26

Table 4.2: Object category wise detection AP (%) on Pascal VOC2007 test set.

Results. We report the detection AP for all the 20 object categories obtained by latent SVM and latent AP-SVM. For all object categories other than 'bottle', latent AP-SVM does better than latent SVM on the test set. For 15 of the 20 object categories, we get statistically significant improvement with latent AP-SVM over latent SVM (using paired t-test with p-value less than 0.05). While latent AP-SVM gives an overall improvement of 7.12% compared to latent SVM, for 5 classes it gives an improvement of more than 10%. The bottom 2 classes with the least improvement obtained by latent AP-SVM, 'chair' and 'bottle' seem to be difficult object categories to detect, with detectors registering very low detection APs. The superior performance of latent AP-SVM compared to latent SVM can be partially attributed to the better localization of objects by latent AP-SVM during training. Figure 4.8 shows this difference in localization performance of the two methods.

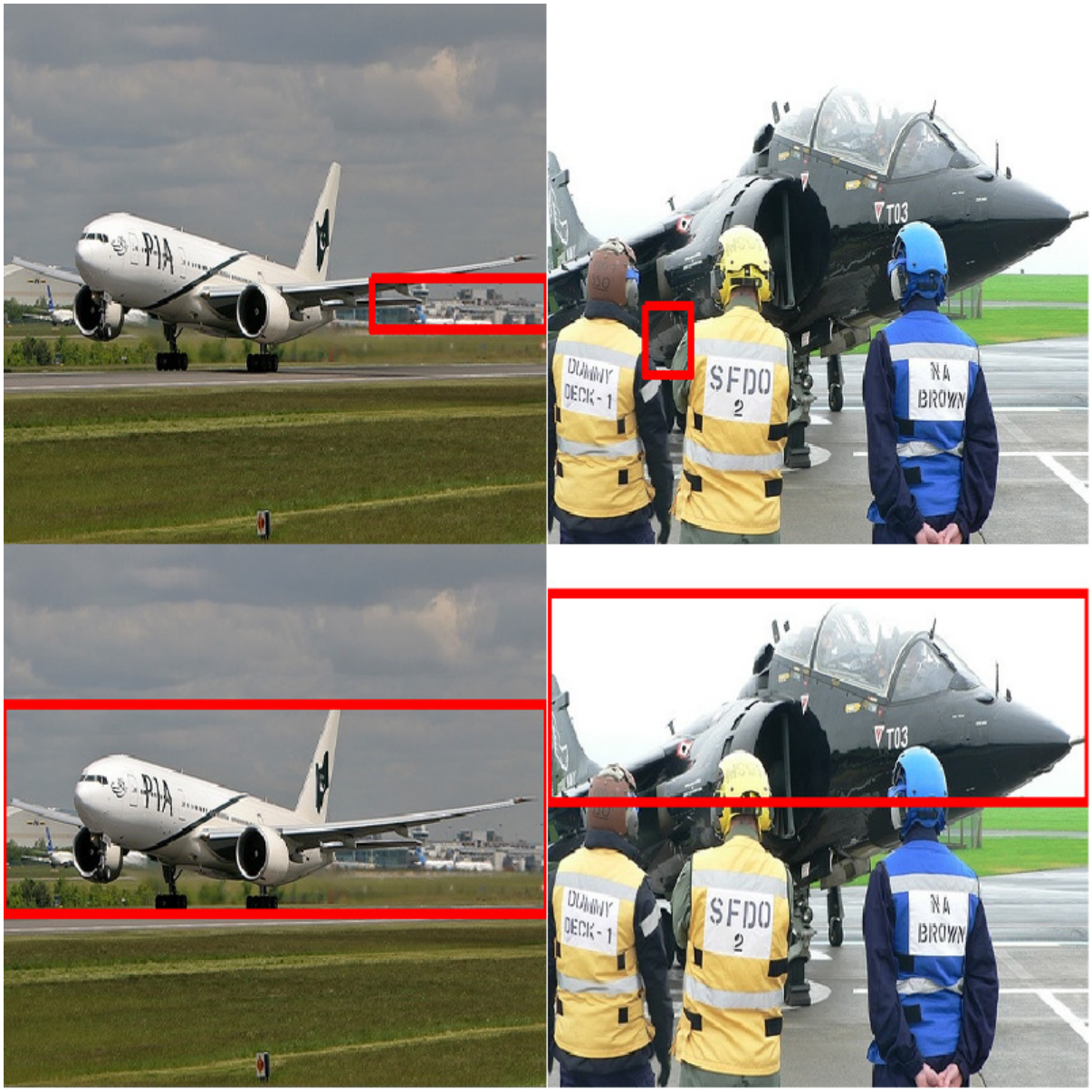

Figure 4.8: *Localization results for some training set images. First row corresponds to latent* SVM *and the second row corresponds to latent* AP-SVM.

Chapter 5

Conclusion and Future Work

In this thesis, we have explored the problem of optimising average precision (AP) in a weakly supervised setting and presented the solutions to solve the problem. We are motivated by the drawbacks in the existing approaches to optimise AP using standard latent SSVM formulation, namely using a non-intuitive prediction rule, while optimising a very loose upper bound on AP loss and lack of an efficient solution to the latent SSVM learning problem. We proposed a novel latent AP-SVM formulation that obtains accurate ranking by minimizing a carefully designed difference-of-convex upper bound on the AP loss. We showed the theoretical benefit of our novel latent AP-SVM over the standard latent SSVM formulation, namely that it minimizes a tighter upper bound on the AP loss and allows for efficient inference, while using an intuitive prediction rule. Finally, we showed the advantage of our approach over latent SVM and the standard latent SSVM for action classification, character recognition and object detection on standard, publicly available PASCAL VOC 2011 [19] action classification dataset, IIIT 5K-WORD [41] scene text dataset and PASCAL VOC 2007 [20] object detection dataset respectively.

5.1 Future Work

We list down some of the possible extensions of the work presented in this thesis:

- **Extension to handle noisy labels**

  An interesting direction of future research would be to extend the latent AP-SVM formulation to learn from images that have been labelled by noisy tags. This will allow us to exploit the large, freely available datasets provided by photo-sharing websites (for example, Flickr or Google images).

- **Improved efficiency**

  The large size of weakly supervised and noisy tagged datasets would also make it necessary to improve the efficiency of our method. Therefore, the efficiency of CCCP algorithm for latent AP-SVM can be improved.

- **Weakly supervised learning for other loss functions**

  Another interesting direction of future work would be to extend our method to handle other ranking based loss functions like AUC (area under the ROC curve).

Appendix A

Optimization for Latent SSVM

The parameters $\mathbf{w}$ of a latent SSVM are learned by minimizing a regularized upper bound on the training loss. Specifically, the parameters are obtained by solving the following optimization problem:

$$\begin{aligned} &\min_{\mathbf{w}} \frac{1}{2}||\mathbf{w}||^2 + C\xi, \\ &\text{s.t. } \forall \mathbf{Y}, \hat{\mathbf{H}}_N, \mathbf{H}_P, \mathbf{H}_N : \\ &\max_{\hat{\mathbf{H}}_P} \{\mathbf{w}^\top \Psi(\mathbf{X}, \mathbf{Y}^*, \hat{\mathbf{H}}_P, \hat{\mathbf{H}}_N)\} - \mathbf{w}^\top \Psi(\mathbf{X}, \mathbf{Y}, \mathbf{H}_P, \mathbf{H}_N) \geq \Delta(\mathbf{Y}^*, \mathbf{Y}) - \xi. \end{aligned} \tag{A.1}$$

The above problem belongs to a special class of non-convex optimization problems called difference-of-convex programs. Specifically, its feasible region can be viewed as the difference of two convex sets. Thus, we can employ a concave-convex procedure (CCCP) [65] to obtain an approximate solution, as described in Algorithm 6. Briefly, the CCCP algorithm starts with an initial set of parameters and iterates over two steps until convergence. In the first step, it fixes the parameters and finds the best set of additional annotations $\hat{\mathbf{H}}_P^*$ of the positives samples for the ground-truth output $\mathbf{Y}^*$. In the second step, it fixes the additional annotations $\hat{\mathbf{H}}_P^*$ and updates the parameters by solving the resulting convex optimization problem.

In more detail, the imputation of the additional annotations for the positive samples requires us to minimize the slack ξ, which is carried out by solving the following problem:

$$\operatorname*{argmin}_{\hat{\mathbf{H}}_P} \{\Delta(\mathbf{Y}^*, \mathbf{Y}) - \mathbf{w}^\top \Psi(\mathbf{X}, \mathbf{Y}^*, \{\hat{\mathbf{H}}_P, \hat{\mathbf{H}}_N\}) + \mathbf{w}^\top \Psi(\mathbf{X}, \mathbf{Y}, \{\mathbf{H}_P, \mathbf{H}_N\})\}.$$

As the loss term $\Delta(\mathbf{Y}^*, \mathbf{Y})$ and score for incorrect ranking $\Psi(\mathbf{X}, \mathbf{Y}, \{\mathbf{H}_P, \mathbf{H}_N\})$ are independent of $\hat{\mathbf{H}}_P$, problem (A.2) reduces to the following:

$$\hat{\mathbf{H}}_P^* = \operatorname*{argmax}_{\hat{\mathbf{H}}_P} \mathbf{w}^\top \Psi(\mathbf{X}, \mathbf{Y}^*, \{\hat{\mathbf{H}}_P, \hat{\mathbf{H}}_N\}). \tag{A.2}$$

We refer to the above problem as output-consistent inference (since it fills in the missing information under the constraint that it is consistent with the output, that is, the optimal ranking).

Given the imputed latent variables $\hat{\mathbf{H}}_P^*$, the parameter can be updated using the cutting plane algorithm. The computational feasibility of the cutting plane algorithm relies on being able to efficiently compute the most violated constraint. In our case, the most violated constraint is found by solving the following problem:

$$\hat{\mathbf{Y}}, \hat{\mathbf{H}}_N, \mathbf{H}_P, \mathbf{H}_N = \underset{\mathbf{Y}, \hat{\mathbf{H}}_N, \mathbf{H}_P, \mathbf{H}_N}{\operatorname{argmax}} \{\mathbf{w}^\top \Psi(\mathbf{X}, \mathbf{Y}, \{\mathbf{H}_P, \mathbf{H}_N\}) - \mathbf{w}^\top \Psi(\mathbf{X}, \mathbf{Y}^*, \{\hat{\mathbf{H}}_P^*, \hat{\mathbf{H}}_N\}) + \Delta(\mathbf{Y}^*, \mathbf{Y})\}. \tag{A.3}$$

We refer to the above problem as loss-augmented inference (since it augments the score of the ranking with its AP loss).

Algorithm 6 *The* CCCP *algorithm for parameter estimation of latent* SSVM.

Require: $\mathbf{X}, \mathbf{Y}^*, \mathbf{w}_0, \epsilon$

$t \Leftarrow 0$

repeat

Impute the latent variables by solving problem (A.2).

Update $\mathbf{w}_{t+1}$ by fixing the latent variables to $\hat{\mathbf{H}}_P^*$ and solving the following convex problem,

$$\mathbf{w}_{t+1} = \operatorname{argmin}_{\mathbf{w}} \tfrac{1}{2}||\mathbf{w}||^2 + C\xi,$$

$$\text{s.t. } \mathbf{w}^\top \Psi(\mathbf{X}, \mathbf{Y}^*, \{\hat{\mathbf{H}}_P^*, \hat{\mathbf{H}}_N\}) - \mathbf{w}^\top \Psi(\mathbf{X}, \mathbf{Y}, \{\mathbf{H}_P, \mathbf{H}_N\}) \geq \Delta(\mathbf{Y}^*, \mathbf{Y}) - \xi, \forall \mathbf{Y}, \hat{\mathbf{H}}_N, \mathbf{H}_P, \mathbf{H}_N$$

$t \leftarrow t + 1$

until Objective function cannot be decreased below tolerance ϵ

To summarize, CCCP requires us to solve two problems: problem (A.2), that is, output-consistent inference and problem (A.3), that is, loss-augmented inference. We now describe how both the problems can be solved efficiently to obtain an accurate set of parameters for latent SSVM.

Output-Consistent Inference.

Problem (A.2) can be solved efficiently by independently choosing the additional annotation for each positive sample using the following criterion:

$$\mathbf{h}_i^* = \underset{\mathbf{h}_i}{\operatorname{argmax}}\, s_i(\mathbf{h}_i). \tag{A.4}$$

Loss-Augmented Inference.

To the best of our knowledge, the second step of the CCCP algorithm cannot be solved optimally due to the lack of an efficient algorithm that computes the most violating constraint for the corresponding cutting plane algorithm. We now provide details of an approximate optimization algorithm for problem (A.3) that was used in our experiments.

An approximate solution of problem (A.3) can be obtained in two steps. In the first step we minimize the positive score $\mathbf{w}^\top \Psi(\mathbf{X}, \mathbf{Y}^*, \{\hat{\mathbf{H}}_P^*, \hat{\mathbf{H}}_N\})$ over $\hat{\mathbf{H}}_N$. Since, the positive score is independent of the loss, it decomposes over all negative samples and thus can be solved independently by choosing the additional annotation $\mathbf{h}_j$ for each negative sample using the following criterion:

$$\hat{\mathbf{h}}_j = \operatorname*{argmax}_{\mathbf{h}_j} s_j(\mathbf{h}_j). \tag{A.5}$$

The second step is to maximize the loss-augmented negative score. Note that loss-augmented inference requires us to not only obtain the ranking of the samples, but also the additional annotations of each sample. Let us denote the set of all the ranks that are occupied by a positive sample by $\mathcal{R}^+$. Similarly, the set of all the ranks that are occupied by a negative sample is denoted by $\mathcal{R}^-$. We first focus on the problem of finding the additional annotations of each sample for a given ranking, that is, for fixed sets $\mathcal{R}^+$ and $\mathcal{R}^-$. We will later describe how we approximately optimize over the rankings. Let us consider the task of obtaining the additional annotations for the positive samples. To solve this task, we construct a $|\mathcal{P}| \times |\mathcal{P}|$ matrix $\mathbf{S}_P$, such that $\mathbf{S}_P(i, a)$ is the score of assigning the sample $i \in \mathcal{P}$ to the rank $a \in \mathcal{R}^+$. Formally,

$$\mathbf{S}_P(i, a) = \max_{\mathbf{h}_i} c_a s_i(\mathbf{h}_i), \tag{A.6}$$

where

$$c_a = \max_{b \in \mathcal{R}^-} \delta(b < a) - \delta(b > a). \tag{A.7}$$

$\delta(.)$ returns the number of positive, negative example pairs satisfying the condition in the argument. The best assignment of additional annotations for the positive samples can be found efficiently by applying the dynamic Hungarian algorithm [31] to the matrix $\mathbf{S}_P$. Similarly, to solve the task of obtaining the additional annotations for the negative samples, we construct a $|\mathcal{N}| \times |\mathcal{N}|$ matrix $\mathbf{S}_N$, such that $\mathbf{S}_N(j, a)$ is the score of assigning the sample $j \in \mathcal{N}$ to the rank $a \in \mathcal{R}^-$, that is,

$$\mathbf{S}_N(j, a) = \max_{\mathbf{h}_j} c_a s_j(\mathbf{h}_j), \tag{A.8}$$

where

$$c_a = \max_{b \in \mathcal{R}^+} \delta(b > a) - \delta(b < a). \tag{A.9}$$

Once again, the best assignment of additional annotations for the negative samples can be found efficiently by applying the dynamic Hungarian algorithm [31] to the matrix $\mathbf{S}_N$.

The above argument shows that, for a given ranking, the optimal assignment of additional annotations for both the positive and the negative samples can be obtained in a computationally feasible manner. However, the optimization over the ranking itself poses a difficult problem. In order to obtain an approximate solution, we employ the following greedy strategy, which is a natural extension of the algorithm proposed by Yue *et al.* [64] for the supervised learning case. We start with the perfect ranking, where all the positive samples are ranked higher than all the negative samples. Next, we consider shifting the highest negative rank up the ranking while keeping all other ranks fixed. For each such

ranking, we compute the value of the loss. Furthermore, we also compute the assignment of additional annotation to the ranks such that the score is maximized. We pick the ranking that maximizes the loss augmented score among all such rankings. Next, we consider shifting the second highest negative rank up the ranking, and find the ranking that maximizes the loss augmented negative score. We continue this procedure until we have considered shifting the lowest negative rank up the ranking.

Non-Optimality of Loss-Augmented Inference

The above algorithm can be shown to be non-optimal using a counter-example. For example, consider table A.1 with two positive and negative samples each. Each sample has two possible values for additional annotation. The first row of the table represents the identifier of each sample. The second and third rows represent the score of the sample upon choosing first or second value of additional annotation respectively. The bottom row contains the label of each sample.

Sample ID	0	1	2	3
Score_0	4	7	1	0
Score_1	8	6	9	7
Label	-1	1	-1	1

Table A.1: *Counter-example with two positive and negative samples each. Each sample has two possible values for additional annotation. The first row of the table represents the identifier of each sample. The second and third rows represent the score of the sample upon choosing first or second value of additional annotation respectively. The bottom row contains the label of each sample.*

We represent the ranking as a list of ordered pairs (a, b). The first entry of the ordered pair is the Sample-ID and second entry is the index of additional annotation selected:

The global optimum for the example in Table A.1 is the following ranking,

$$(0, 1); (2, 1); (3, 0); (1, 1),$$

which yields a loss augmented score of is 22.6. In contrast, the greedy algorithm described above provides the following ranking,

$$(1, 0); (3, 1); (2, 0); (0, 0),$$

which yields a loss augmented score of 18.

Publications

- Aseem Behl, C. V. Jawahar and M. Pawan Kumar: **Optimizing Average Precision using Weakly Supervised Data**, Computer Vision and Pattern Recognition (CVPR), 2014
- Puneet Kumar Dokania, Aseem Behl, C. V. Jawahar, M. Pawan Kumar: **Learning to Rank Using High-Order Information**, European Conference on Computer Vision (ECCV), 2014
- Aseem Behl, Pritish Mohapatra, C.V. Jawahar and M. Pawan Kumar: **Optimizing Average Precision using Weakly Supervised Data**, In Pattern Analysis and Machine Ingelligence (PAMI), 2015

Bibliography

[1] B. Bartell, E. Britannica, R. Belew, G. Cottrell, and R. Belew. Learning to retrieve information. In *Proceedings of the Swedish Conference on Connectionism*, 1995.

[2] M. Blaschko, A. Vedaldi, and A. Zisserman. Simultaneous object detection and ranking with weak supervision. In *NIPS*, 2010.

[3] C. J. C. Burges, R. Ragno, and Q. V. Le. Learning to rank with nonsmooth cost functions. In *NIPS*, 2006.

[4] C. J. C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. N. Hullender. Learning to rank using gradient descent. In *ICML*, 2005.

[5] Y. Cao, J. X. 0001, T.-Y. Liu, H. Li, Y. Huang, and H.-W. Hon. Adapting ranking svm to document retrieval. In *SIGIR*, 2006.

[6] Y. Cao, J. Xu, T. yan Liu, H. Li, Y. Huang, and H. wuen Hon. Adapting ranking svm to document retrieval. In *In Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2006.

[7] Z. Cao, T. Qin, T. Liu, M. Tsai, and H. Li. Learning to rank: from pairwise approach to listwise approach. In *ICML*, 2007.

[8] S. Chakrabarti, R. Khanna, U. Sawant, and C. Bhattacharyya. Structured learning for non-smooth ranking losses. In *SIGKDD*, 2008.

[9] W. Chu and S. S. Keerthi. New approaches to support vector ordinal regression. In *ICML*, 2005.

[10] W. Chu and S. S. Keerthi. Support vector ordinal regression. *Neural Computation*, 2007.

[11] W. W. Cohen, R. E. Schapire, and Y. Singer. Learning to order things. *J. Artif. Intell. Res. (JAIR)*, 1999.

[12] W. S. Cooper, F. C. Gey, and D. P. Dabney. Probabilistic retrieval based on staged logistic regression. In *SIGIR*, 1992.

[13] D. Cossock and T. Zhang. Subset ranking using regression. In *COLT*, pages 605–619, 2006.

[14] K. Crammer and Y. Singer. Pranking with ranking. In *Advances in Neural Information Processing Systems 14*, 2001.

[15] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *CVPR*, 2005.

[16] V. Delaitre, I. Laptev, and J. Sivic. Recognizing human actions in still images: a study of bag-of-features and part-based representations. In *BMVC*, 2010.

[17] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

[18] T. Deselaers, B. Alexe, and V. Ferrari. Localizing objects while learning their appearance. In *ECCV*, 2010.

[19] M. Everingham, L. Van Gool, C. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *IJCV*, 2010.

[20] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results. http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html.

[21] P. Felzenszwalb, R. Girshick, and D. McAllester. Discriminatively trained deformable part models, release 4. http://people.cs.uchicago.edu/ pff/latent-release4/.

[22] P. Felzenszwalb, R. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *PAMI*, 2010.

[23] Y. Freund, R. D. Iyer, R. E. Schapire, and Y. Singer. An efficient boosting algorithm for combining preferences. *Journal of Machine Learning Research*, 2003.

[24] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. *arXiv preprint arXiv:1311.2524*, 2013.

[25] R. Herbrich, T. Graepel, and K. Obermayer. Large margin rank boundaries for ordinal regression. In *Advances in Large Margin Classifiers*, 2000.

[26] K. Järvelin and J. Kekäläinen. IR evaluation methods for retrieving highly relevant documents. In *SIGIR*, 2000.

[27] T. Joachims. Making large-scale SVM learning practical. In B. Scholkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods*. MIT Press, 1999.

[28] T. Joachims. Optimizing search engines using clickthrough data. In *SIGKDD*, 2002.

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

[30] T. Joachims, T. Finley, and C.-N. J. Yu. Cutting-plane training of structural svms. *Machine Learning*, 2009.

[31] G. A. Korsah, A. T. Stentz, and M. B. Dias. The dynamic hungarian algorithm for the assignment problem with changing costs. Technical report, Robotics Institute, 2007.

[32] M. P. Kumar, B. Packer, and D. Koller. Modeling latent variable uncertainty for loss-based learning. In *ICML*, 2012.

[33] M. P. Kumar, H. Turki, D. Preston, and D. Koller. Learning specific-class segmentation from diverse data. In *ICCV*, 2011.

[34] J. D. Lafferty and C. Zhai. Document language models, query models, and risk minimization for information retrieval. In *SIGIR*, 2001.

[35] T. Lan, Y. Wang, W. Yang, S. Robinovitch, and G. Mori. Discriminative latent models for recognizing contextual group activities. *PAMI*, 2012.

[36] P. Li, C. J. C. Burges, and Q. Wu. Mcrank: Learning to rank using multiple classification and gradient boosting. In *NIPS*, 2007.

[37] Y. Lin, F. Lv, S. Zhu, M. Yang, T. Cour, K. Yu, L. Cao, and T. Huang. Large-scale image classification: Fast feature extraction and svm training. In *CVPR*, 2011.

[38] T. Liu. *Learning to Rank for Information Retrieval*. Springer, 2011.

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

[40] K. Miller, M. P. Kumar, B. Packer, D. Goodman, and D. Koller. Max-margin min-entropy models. In *AISTATS*, 2012.

[41] A. Mishra, K. Alahari, and C. V. Jawahar. Scene text recognition using higher order language priors. In *BMVC*, 2012.

[42] M. Pandey and S. Lazebnik. Scene recognition and weakly supervised object localization with deformable part-based models. In *ICCV*, 2011.

[43] A. Prest, C. Schmid, and V. Ferrari. Weakly supervised learning of interactions between humans and objects. *PAMI*, 2012.

[44] T. Qin, X.-D. Zhang, M.-F. Tsai, D.-S. Wang, T.-Y. Liu, and H. Li. Query-level loss functions for information retrieval. In *Information Processing and Management*, 2008.

[45] O. Russakovsky, Y. Lin, K. Yu, and L. Fei-Fei. Object-centric spatial pooling for image classification. In *ECCV*, 2012.

[46] S. Shalev-Shwartz, Y. Singer, and N. Srebro. Pegasos: Primal estimated sub-gradient solver for SVM. In *ICML*, 2009.

[47] A. Smola, S. Vishwanathan, and T. Hofmann. Kernel methods for missing variables. In *AISTATS*, 2005.

[48] P. K. Srijith, S. K. Shevade, and S. Sundararajan. Semi-supervised gaussian process ordinal regression. In *ECML*, 2013.

[49] M. J. Taylor, J. Guiver, S. Robertson, and T. Minka. Softrank: optimizing non-smooth rank metrics. In *WSDM*, 2008.

[50] I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun. Large margin methods for structured and interdependent output variables. *JMLR*, 2005.

[51] J. Uijlings, K. van de Sande, T. Gevers, and A. Smeulders. Selective search for object recognition. *International journal of computer vision*, 104(2), 2013.

[52] V. Vapnik. *Statistical learning theory*. Wiley, 1998.

[53] A. Vedaldi, V. Gulshan, M. Varma, and A. Zisserman. Multiple kernels for object detection. In *ICCV*, 2009.

[54] A. Vezhnevets, J. Buhmann, and V. Ferrari. Weakly supervised structured output learning for semantic segmentation. In *CVPR*, 2012.

[55] E. M. Voorhees. Overview of the TREC 2001 question answering track. In *TREC*, 2001.

[56] H. Wang, S. Gould, and D. Koller. Discriminative learning with latent variables for cluttered indoor scene understanding. In *ECCV*, 2010.

[57] Y. Wang and G. Mori. A discriminative latent model of object classes and attributes. In *ECCV*, 2010.

[58] T. yan Liu, J. Wang, W. Zhang, and H. Li. Listwise approach to learning to rank - theory and algorithm. In *ICML*, 2008.

[59] J. Yang, K. Yu, Y. Gong, and T. Huang. Linear spatial pyramid matching using sparse coding for image classification. In *CVPR*, 2009.

[60] L. Yang, R. Jin, R. Sukthankar, and F. Jurie. Unifying discriminative visual codebook generation with classifier training for object category recognition. In *CVPR*, 2008.

[61] W. Yang, Y. Wang, and G. Mori. Recognizing human actions from still images with latent poses. In *CVPR*, 2010.

[62] B. Yao, X. Jiang, A. Khosla, A. Lin, L. Guibas, and L. Fei-Fei. Human action recognition by learning bases of action attributes and parts. In *ICCV*, 2011.

[63] C.-N. Yu and T. Joachims. Learning structural svms with latent variables. In *ICML*, 2009.

[64] Y. Yue, T. Finley, F. Radlinski, and T. Joachims. A support vector method for optimizing average precision. In *SIGIR*, 2007.

[65] A. Yuille and A. Rangarajan. The concave-convex procedure. *Neural Computation*, 2003.