DOI: http://dx.doi.org/10.1109/TPAMI.2014.2315792

StructBoost: Boosting Methods for Predicting Structured Output Variables

Chunhua Shen, Guosheng Lin, Anton van den Hengel

Abstract—Boosting is a method for learning a single accurate predictor by linearly combining a set of less accurate weak learners. Recently, structured learning has found many applications in computer vision. Inspired by structured support vector machines (SSVM), here we propose a new boosting algorithm for structured output prediction, which we refer to as StructBoost. StructBoost supports nonlinear structured learning by combining a set of weak structured learners.

As SSVM generalizes SVM, our StructBoost generalizes standard boosting approaches such as AdaBoost, or LPBoost to structured learning. The resulting optimization problem of StructBoost is more challenging than SSVM in the sense that it may involve exponentially many variables and constraints. In contrast, for SSVM one usually has an exponential number of constraints and a cutting-plane method is used. In order to efficiently solve StructBoost, we formulate an equivalent 1-slack formulation and solve it using a combination of cutting planes and column generation. We show the versatility and usefulness of StructBoost on a range of problems such as optimizing the tree loss for hierarchical multi-class classification, optimizing the Pascal overlap criterion for robust visual tracking and learning conditional random field parameters for image segmentation.

Index Terms—Boosting, ensemble learning, AdaBoost, structured learning, conditional random field.

• *The authors are with School of Computer Science, The University of Adelaide, SA 5005, Australia. Correspondence should be addressed to C. Shen (e-mail: chunhua.shen@adelaie.edu.au).*

CONTENTS

1 Introduction

Structured learning has attracted considerable attention in machine learning and computer vision in recent years (see, for example [1]–[4]). Conventional supervised learning problems, such as classification and regression, aim to learn a function that predicts the best value for a response variable $y \in \mathbb{R}$ for an input vector $\boldsymbol{x} \in \mathbb{R}^d$ on the basis of a set of example input-output pairs. In many applications, however, the outputs are often complex and cannot be well represented by a single scalar because the classes may have inter-class dependencies, or the most appropriate outputs are objects (vectors, sequences, trees, etc.). Such problems are referred to as *structured output prediction*. Structured support vector machines (SSVM) [4] generalize the multi-class SVM of [5] and [6] to the much broader problem of predicting interdependent and structured outputs. SSVM uses discriminant functions that take advantage of the dependencies and structure of outputs. In SSVM, the general form of the learned discriminant function is $F(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{w}) : \mathcal{X} \times \mathcal{Y} \mapsto \mathbb{R}$ over input-output pairs and the prediction is achieved by maximizing $F(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{w})$ over all possible $\boldsymbol{y} \in \mathcal{Y}$. Note that to introduce non-linearity, the discriminant function can be defined by an *implicit* feature mapping function that is only accessible as a particular inner product in a reproducing kernel Hilbert space. This is the so-called kernel trick.

On the other hand, boosting algorithms linearly combine a set of moderately accurate weak learners to form a nonlinear strong predictor, whose prediction performance is usually highly accurate. Recently, Shen and Hao [7] proposed a direct formulation for multi-class boosting using the loss functions of multi-class SVMs [5], [6]. Inspired by the general boosting framework of Shen and Li [8], they implemented multi-class boosting using column generation. Here we go further by generalizing multi-class boosting of Shen and Hao to broad structured output prediction problems. StructBoost thus enables nonlinear structured learning by combining a set of weak structured learners.

The effectiveness of SSVM has been limited by the fact that only the linear kernel is typically used. This limitation arises largely as a result of the computational expense of training and applying kernelized SSVMs. Nonlinear kernels often deliver improved prediction accuracy over that of linear kernels, but at the cost of significantly higher memory requirements and computation time. This is particularly the case when the training size is large, because the number of support vectors is linearly proportional to the size of training data [9]. Boosting, however, learns models which are much faster to evaluate. Boosting can also select relevant features during the course of learning by using particular weak learners such as decision stumps or decision trees, while almost all nonlinear kernels are defined on the entire feature space. It thus remains difficult (if not impossible) to see how kernel methods can select/learn explicit features. For boosting, the learning procedure also selects or induces relevant features. The final model learned by boosting methods are thus often significantly simpler and computationally cheaper. In this sense, the proposed StructBoost possesses the advantages of both nonlinear SSVM and boosting methods.

1.1 Main contributions

The main contributions of this work are three-fold.

1. We propose StructBoost, a new fully-corrective boosting method that combines a set of weak structured learners for predicting a broad range of structured outputs. We also discuss special cases of this general structured learning framework, including multi-class classification, ordinal regression, optimization of complex measures such as the Pascal image overlap criterion and conditional random field (CRF) parameters learning for image segmentation.
2. To implement StructBoost, we adapt the efficient cutting-plane method—originally designed for efficient linear SVM training [10]—for our purpose. We equivalently reformulate the m-slack optimization to 1-slack optimization.
3. We apply the proposed StructBoost to a range of computer vision problems and show that StructBoost can indeed achieve state-of-the-art performance in some of the key problems in the field. In particular, we demonstrate a state-of-the-art object tracker trained by StructBoost. We also demonstrate an application for CRF and super-pixel based image segmentation, using StructBoost together with graph cuts for CRF parameter learning.

Since StructBoost builds upon the fully corrective boosting of Shen and Li [8], it inherits the desirable properties of column generation based boosting, such as a fast convergence rate and clear explanations from the primal-dual convex optimization perspective.

1.2 Related work

The two state-of-the-art structured learning methods are CRF [11] and SSVM [4], which capture the interdependency among output variables. Note that CRFs formulate global training for structured prediction as a convex optimization problem. SSVM also follows this path but employs a different loss function (hinge loss) and optimization methods. Our StructBoost is directly inspired by SSVM. StructBoost can be seen as an extension of boosting methods to structured prediction. It therefore builds upon the column generation approach to boosting from [8] and the direct formulation for multi-class boosting [7]. Indeed, we show that the multi-class boosting of [7] is a special case of the general framework presented here.

CRF and SSVM have been applied to various problems in machine learning and computer vision mainly because the learned models can easily integrate prior knowledge given a problem of interest. For example, the linear chain CRF has been widely used in natural language processing [11], [12]. SSVM takes the context into account using the joint feature maps over the input-output pairs, where features can be

represented equivalently as in CRF [10]. CRF is particularly of interest in computer vision for its success in semantic image segmentation [13]. A critical issue of semantic image segmentation is to integrate local and global features for the prediction of local pixel/segment labels. Semantical segmentation is achieved by exploiting the class information with a CRF model. SSVM can also be used for similar purposes as demonstrated in [14]. Blaschko and Lampert [3] trained SSVM models to predict the bounding box of objects in a given image, by optimizing the Pascal bounding box overlap score. The work in [1] introduced structured learning to real-time object detection and tracking, which also optimizes the Pascal box overlap score. SSVM has also been used to learn statistics that capture the spatial arrangements of various object classes in images [15]. The trained model can then simultaneously predict a structured labeling of the entire image. Based on the idea of large-margin learning in SSVM, Szummer et al. [16] learned optimal parameters of a CRF, avoiding tedious cross validation. The survey of [2] provided a comprehensive review of structured learning and its application in computer vision.

Dietterich et al. [17] learned the CRF energy functions using gradient tree boosting. There the functional gradient of the CRF *conditional likelihood* is calculated, such that a regression tree (weak learner) is induced as in gradient boosting. An ensemble of trees is produced by iterating this procedure. In contrast, we learn CRF within the *large-margin* framework, by generalizing the work of [16], [18] where CRF parameters are learned using SSVM. In our case, we do not require approximations such as pseudo-likelihood. Another relevant work is [19], where Munoz et al. used the functional gradient boosting methodology to discriminatively learn max-margin Markov networks (M3N), as proposed by Taskar et al. [20]. The random fields' potential functions are learned following gradient boosting [21].

There are a few structured boosting methods in the literature. As we discuss here, all of them are based on gradient boosting, which are not as general as that which we propose here. Ratliff et al. [22], [23] proposed a boosting-based approach for imitation learning based on structured prediction, called maximum margin planning (MMP). Their method is named as MMPBoost. To train MMPBoost, a demonstrated policy is provided as example behavior as the input, and the problem is to learn a function over features of the environment that produce policies with similar behavior. Although MMPBoost is structured learning in that the output is a vector, it differs from ours fundamentally. First, the optimization procedure of MMPBoost is not directly defined on the joint function $F(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{w})$. Second, MMPBoost is based on gradient descent boosting [21], and StructBoost is built upon fully corrective boosting of Shen and Li [8], [24].

Parker et al. [25] have also successfully applied gradient tree boosting to learning sequence alignment. Later, Parker [26] developed a margin-based structured perceptron update and showed that it can incorporate general notions of misclassification cost as well as kernels. In these methods, the objective function typically consists of an exponential number of terms that correspond to all possible pairs of $(\boldsymbol{y}, \boldsymbol{y}')$. Approximation is made to make the computation of gradient tractable [25]. Wang et al. [27] learned a local predictor using standard methods, e.g., SVM, but then achieved improved structured classification by exploiting the influence of misclassified components after structured prediction, and iteratively re-training the local predictor. This approach is heuristic and it is more like a post-processing procedure—it does not directly optimize the structured learning objective.

1.3 Notation

A bold lower-case letter ($\boldsymbol{u}$, $\boldsymbol{v}$) denotes a column vector. An element-wise inequality between two vectors or matrices such as $\boldsymbol{u} \geq \boldsymbol{v}$ means $u_i \geq v_i$ for all i. Let $\boldsymbol{x}$ be an input; $\boldsymbol{y}$ be an output and the input-output pair be $(\boldsymbol{x}, \boldsymbol{y}) \in \mathcal{X} \times \mathcal{Y}$, with $\mathcal{X} \subset \mathbb{R}^d$. Unlike classification ($\mathcal{Y} = \{1, 2, \ldots, k\}$) or regression ($\mathcal{Y} \subset \mathbb{R}$) problems, we are interested in the case where elements of $\mathcal{Y}$ are *structured variables* such as vectors, strings, or graphs. Recall that the proposed StructBoost is a structured boosting method, which combines a set of *weak structured learners* (or weak compatibility functions). We denote by $\mathcal{H}$ the set of weak structured learners. Note that $\mathcal{H}$ is typically very large, or even infinite. Each weak structured learner: $\psi(\cdot, \cdot) \in \mathcal{H}$, is a function that maps an input-output pair $(\boldsymbol{x}, \boldsymbol{y})$ to a scalar value which measures the compatibility of the input and output. We define column vector $\Psi(\boldsymbol{x}, \boldsymbol{y}) = [\psi_1(\boldsymbol{x}, \boldsymbol{y}), \cdots, \psi_n(\boldsymbol{x}, \boldsymbol{y})]^\top$ to be the outputs of all weak structured learners. Thus $\Psi(\boldsymbol{x}, \boldsymbol{y})$ plays the same sole as the joint mapping vector in SSVM, which relates input $\boldsymbol{x}$ and output $\boldsymbol{y}$. The form of a weak structured learner is task-dependent. We show some examples of $\psi(\cdot, \cdot)$ in Section 3. The discriminant function that we aim to learn is $F : \mathcal{X} \times \mathcal{Y} \mapsto \mathbb{R}$, which measures the compatibility over input-output pairs. It has the form of

$$F(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{w}) = \boldsymbol{w}^\top \Psi(\boldsymbol{x}, \boldsymbol{y}) = \textstyle\sum_j w_j \psi_j(\boldsymbol{x}, \boldsymbol{y}), \qquad (1)$$

with $\boldsymbol{w} \geq 0$. As in other structured learning models, the process for predicting a structured output (or inference) is to find an output $\boldsymbol{y}$ that maximizes the joint compatibility function:

$$\boldsymbol{y}^\star = \operatorname*{argmax}_{\boldsymbol{y}} F(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{w}) = \operatorname*{argmax}_{\boldsymbol{y}} \boldsymbol{w}^\top \Psi(\boldsymbol{x}, \boldsymbol{y}). \qquad (2)$$

We denote by $\mathbf{1}$ a column vector of all 1's, whose dimension shall be clear from the context.

We describe the StructBoost approach in Section 6.1, including how to efficiently solve the resulting optimization problem. We then highlight applications in various domains in Section 3. Experimental results are shown in Section 4 and we conclude the paper in the last section.

2 Structured Boosting

We first introduce the general structured boosting framework, and then apply it to a range of specific problems: classification, ordinal regression, optimizing special criteria

such as the area under the ROC curve and the Pascal image area overlap ratio, and learning CRF parameters.

To measure the accuracy of prediction we use a loss function, and as is the case with SSVM, we accept arbitrary loss functions $\Delta : \mathcal{Y} \times \mathcal{Y} \mapsto \mathbb{R}$. $\Delta(\boldsymbol{y}, \boldsymbol{y}')$ calculates the loss associated with a prediction $\boldsymbol{y}'$ against the true label value $\boldsymbol{y}$. Note that in general we assume that $\Delta(\boldsymbol{y}, \boldsymbol{y}) = 0$, $\Delta(\boldsymbol{y}, \boldsymbol{y}') > 0$ for any $\boldsymbol{y}' \neq \boldsymbol{y}$ and the loss is upper bounded.

The formulation of StructBoost can be written as (m-slack primal):

$$\min_{\boldsymbol{w}\geq 0, \boldsymbol{\xi}\geq 0} \ \mathbf{1}^\top \boldsymbol{w} + \tfrac{C}{m} \mathbf{1}^\top \boldsymbol{\xi} \tag{3a}$$

$$\text{s.t.:} \ \boldsymbol{w}^\top \Big[\Psi(\boldsymbol{x}_i, \boldsymbol{y}_i) - \Psi(\boldsymbol{x}_i, \boldsymbol{y})\Big] \geq \Delta(\boldsymbol{y}_i, \boldsymbol{y}) - \xi_i, \quad \forall i = 1, \ldots, m; \ \text{and} \ \forall \boldsymbol{y} \in \mathcal{Y}. \tag{3b}$$

Here we have used the ℓ_1 norm as the regularization function to control the complexity of the learned model. To simplify the notation, we introduce

$$\delta\psi_i(\boldsymbol{y}) = \psi(\boldsymbol{x}_i, \boldsymbol{y}_i) - \psi(\boldsymbol{x}_i, \boldsymbol{y}); \tag{4}$$

and,

$$\delta\Psi_i(\boldsymbol{y}) = \Psi(\boldsymbol{x}_i, \boldsymbol{y}_i) - \Psi(\boldsymbol{x}_i, \boldsymbol{y}); \tag{5}$$

then the constraints in (35) can be re-written as:

$$\boldsymbol{w}^\top \delta\Psi_i(\boldsymbol{y}) \geq \Delta(\boldsymbol{y}_i, \boldsymbol{y}) - \xi_i.$$

There are two major obstacles to solve problem (35). First, as in conventional boosting, because the set of weak structured learners $\psi(\cdot, \cdot)$ can be exponentially large or even infinite, the dimension of $\boldsymbol{w}$ can be exponentially large or infinite. Thus, in general, we are not able to directly solve for $\boldsymbol{w}$. Second, as in SSVM, the number of constraints (35b) can be extremely or infinitely large. For example, in the case of multi-label or multi-class classification, the label $\boldsymbol{y}$ can be represented as a binary vector (or string) and clearly the possible number of $\boldsymbol{y}$ such that $\boldsymbol{y}$ is exponential in the length of the vector, which is $2^{|\mathcal{Y}|}$. In other words, *problem (35) can have an extremely or infinitely large number of variables and constraints.* This is significantly more challenging than solving standard boosting or SSVM in terms of optimization. In standard boosting, one has a large number of variables while in SSVM, one has a large number of constraints.

For the moment, let us put aside the difficulty of the large number of constraints, and focus on how to iteratively solve for $\boldsymbol{w}$ using column generation as in boosting methods [8], [28], [29]. We derive the Lagrange dual of the optimization of (35) as:

$$\max_{\boldsymbol{\mu}\geq 0} \ \sum_{i,\boldsymbol{y}} \mu_{(i,\boldsymbol{y})} \Delta(\boldsymbol{y}_i, \boldsymbol{y}) \tag{6a}$$

$$\text{s.t.:} \ \textstyle\sum_{i,\boldsymbol{y}} \mu_{(i,\boldsymbol{y})} \delta\Psi_i(\boldsymbol{y}) \leq \mathbf{1}, \tag{6b}$$

$$0 \leq \textstyle\sum_{\boldsymbol{y}} \mu_{(i,\boldsymbol{y})} \leq \frac{C}{m}, \forall i = 1, \ldots, m. \tag{6c}$$

Here $\boldsymbol{\mu}$ are the Lagrange dual variables (Lagrange multipliers). We denote by $\mu_{(i,\boldsymbol{y})}$ the dual variable associated with the margin constraints (35b) for label $\boldsymbol{y}$ and training pair $(\boldsymbol{x}_i, \boldsymbol{y}_i)$.

The idea of column generation is to split the original primal problem in (35) into two problems: a master problem and a subproblem. The master problem is the original problem with only a subset of variables (or constraints for the dual form) being considered. The subproblem is to add new variables (or constraints for the dual form) into the master problem. With the primal-dual pair of (35) and (37) and following the general framework of column generation based boosting [8], [28], [29], we can obtain our StructBoost as follows:

Iterate the following two steps until converge :

1) Solve the following subproblem, which generates the best weak structured learner by finding the most violated constraint in the dual:

$$\psi^\star(\cdot, \cdot) = \operatorname*{argmax}_{\psi(\cdot,\cdot)} \sum_{i,\boldsymbol{y}} \mu_{(i,\boldsymbol{y})} \delta\psi_i(\boldsymbol{y}). \tag{7}$$

2) Add the selected structured weak learner $\psi^\star(\cdot, \cdot)$ into the master problem (either the primal form or the dual form) and re-solve for the primal solution $\boldsymbol{w}$ and dual solution $\boldsymbol{\mu}$.

The stopping criterion can be that no violated weak learner can be found. Formally, for the selected $\psi^\star(\cdot, \cdot)$ with (7) and a preset precision $\epsilon_{\text{cg}} > 0$, if the following relation holds:

$$\sum_{i,\boldsymbol{y}} \mu_{(i,\boldsymbol{y})} \delta\psi_i^\star(\boldsymbol{y}) \leq 1 - \epsilon_{\text{cg}}, \tag{8}$$

we terminate the iteration. Algorithm 1 presents the details of column generation for StructBoost. This approach, however, may not be practical because it is very expensive to solve the master problem (the reduced problem of (35)) at each column generation (boosting iteration), which still can have extremely many constraints due to the set of $\{\boldsymbol{y} \in \mathcal{Y}\}$. The direct formulation for multi-class boosting in [7] can be seen as a specific instance of this approach, which is in general very slow. We therefore propose to employ the 1-slack formulation for efficient optimization, which is described in the next section.

2.1 1-slack formulation for fast optimization

Inspired by the cutting-plane method for fast training of linear SVM [10] and SSVM [30], we rewrite the above problem into an equivalent "1-slack" form so that the efficient cutting-plane method can be employed to solve the optimization problem (35):

$$\min_{\boldsymbol{w}\geq 0, \xi\geq 0} \ \mathbf{1}^\top \boldsymbol{w} + C\xi \tag{9a}$$

$$\text{s.t.:} \ \frac{1}{m} \boldsymbol{w}^\top \Big[\sum_{i=1}^{m} c_i \cdot \delta\Psi_i(\boldsymbol{y})\Big] \geq \frac{1}{m} \sum_{i=1}^{m} c_i \Delta(\boldsymbol{y}_i, \boldsymbol{y}) - \xi, \quad \forall \boldsymbol{c} \in \{0, 1\}^m; \forall \boldsymbol{y} \in \mathcal{Y}, i = 1, \cdots, m. \tag{9b}$$

The following theorem shows the equivalence of problems (35) and (38).

Theorem 2.1. *A solution of problem* (38) *is also a solution of problem* (35) *and* vice versa. *The connections are:* $\boldsymbol{w}^\star_{(38)} = \boldsymbol{w}^\star_{(35)}$ *and* $\xi^\star_{(38)} = \frac{1}{m}\mathbf{1}^\top \boldsymbol{\xi}^\star_{(35)}$.

Proof: This proof adapts the proof in [10]. Given a fixed $\boldsymbol{w}$, the only variable $\boldsymbol{\xi}_{(35)}$ in (35) can be solved by

$$\xi_{i,(35)} = \max_{\boldsymbol{y}}\Big\{0, \Delta(\boldsymbol{y}_i, \boldsymbol{y}) - \boldsymbol{w}^\top \delta\Psi_i(\boldsymbol{y})\Big\}, \forall i.$$

For (38), the optimal $\xi_{(38)}$ given a $\boldsymbol{w}$ can be computed as:

$$\begin{aligned}
\xi_{(38)} &= \frac{1}{m}\max_{\boldsymbol{c},\boldsymbol{y}}\left\{\sum_{i=1}^m c_i\Delta(\boldsymbol{y}_i,\boldsymbol{y}) - \boldsymbol{w}^\top\Big[\sum_{i=1}^m c_i\delta\Psi_i(\boldsymbol{y})\Big]\right\} \\
&= \frac{1}{m}\sum_{i=1}^m\left\{\max_{c_i\in\{0,1\},\boldsymbol{y}} c_i\Delta(\boldsymbol{y}_i,\boldsymbol{y}) - c_i\boldsymbol{w}^\top\delta\Psi_i(\boldsymbol{y})\right\} \\
&= \frac{1}{m}\sum_{i=1}^m\max_{\boldsymbol{y}}\Big\{0, \Delta(\boldsymbol{y}_i,\boldsymbol{y}) - \boldsymbol{w}^\top\delta\Psi_i(\boldsymbol{y})\Big\} \\
&= \frac{1}{m}\mathbf{1}^\top\boldsymbol{\xi}_{(35)}.
\end{aligned}$$

Note that $\boldsymbol{c}\in\{0,1\}^m$ in the above equalities. Clearly the objective functions of both problems coincide for any fixed $\boldsymbol{w}$ and the optimal $\boldsymbol{\xi}_{(35)}$ and $\xi_{(38)}$. □

As demonstrated in [10] and SSVM [30], cutting-plane methods can be used to solve the 1-slack primal problem (38) efficiently. This 1-slack formulation has been used to train linear SVM in linear time. When solving for $\boldsymbol{w}$, (38) is similar to ℓ_1-norm regularized SVM—except the extra non-negativeness constraint on $\boldsymbol{w}$ in our case.

In order to utilize column generation for designing boosting methods, we need to derive the Lagrange dual of the 1-slack primal optimization problem, which can be written as follows:

$$\max_{\boldsymbol{\lambda}\geq 0} \quad \sum_{\boldsymbol{c},\boldsymbol{y}}\lambda_{(\boldsymbol{c},\boldsymbol{y})}\sum_{i=1}^m c_i\Delta(\boldsymbol{y}_i,\boldsymbol{y}) \tag{10a}$$

$$\text{s.t.:} \quad \frac{1}{m}\sum_{\boldsymbol{c},\boldsymbol{y}}\lambda_{(\boldsymbol{c},\boldsymbol{y})}\Big[\sum_{i=1}^m c_i\cdot\delta\Psi_i(\boldsymbol{y})\Big] \leq \mathbf{1}, \tag{10b}$$

$$0 \leq \textstyle\sum_{\boldsymbol{c},\boldsymbol{y}}\lambda_{(\boldsymbol{c},\boldsymbol{y})} \leq C. \tag{10c}$$

Here $\boldsymbol{c}$ enumerates all possible $\boldsymbol{c}\in\{0,1\}^m$. We denote by $\lambda_{(\boldsymbol{c},\boldsymbol{y})}$ the Lagrange dual variable (Lagrange multiplier) associated with the inequality constraint in (9b) for $\boldsymbol{c}\in\{0,1\}^m$ and label $\boldsymbol{y}$. The subproblem to find the most violated constraint in the dual form for generating weak structured learners is:

$$\begin{aligned}
\psi^\star(\cdot,\cdot) &= \operatorname*{argmax}_{\psi(\cdot,\cdot)}\sum_{\boldsymbol{c},\boldsymbol{y}}\lambda_{(\boldsymbol{c},\boldsymbol{y})}\sum_i c_i\delta\psi_i(\boldsymbol{y}) \\
&= \operatorname*{argmax}_{\psi(\cdot,\cdot)}\sum_{i,\boldsymbol{y}}\underbrace{\sum_{\boldsymbol{c}}\lambda_{(\boldsymbol{c},\boldsymbol{y})}c_i}_{:=\mu_{(i,\boldsymbol{y})}}\delta\psi_i(\boldsymbol{y}).
\end{aligned} \tag{11}$$

We have changed the order of summation in order to have a similar form as in the m-slack case.

Algorithm 1 Column generation for StructBoost

Input: training examples $(\boldsymbol{x}_1;\boldsymbol{y}_1),(\boldsymbol{x}_2;\boldsymbol{y}_2),\cdots$; parameter C; termination threshold ϵ_{cg}, and the maximum iteration number.
Initialize: for each i, $(i=1,\ldots,m)$, randomly pick any $\boldsymbol{y}_i^{(0)}\in\mathcal{Y}$, initialize $\mu_{(i,\boldsymbol{y})} = \frac{C}{m}$ for $\boldsymbol{y}=\boldsymbol{y}_i^{(0)}$, and $\mu_{(i,\boldsymbol{y})}=0$ for all $\boldsymbol{y}\in\mathcal{Y}\backslash\boldsymbol{y}_i^{(0)}$.
Repeat
  – Find and add a weak structured learner $\psi^\star(\cdot,\cdot)$ by solving the subproblem (7) or (11).
  – Call Algorithm 2 to obtain $\boldsymbol{w}$ and $\boldsymbol{\mu}$.
Until either (8) is met or the maximum number of iterations is reached.
Output: the discriminant function $F(\boldsymbol{x},\boldsymbol{y};\boldsymbol{w}) = \boldsymbol{w}^\top\Psi(\boldsymbol{x},\boldsymbol{y})$.

Algorithm 2 Cutting planes for solving the 1-slack primal

Input: the cutting-plane termination threshold ϵ_{cp}, and inputs from Algorithm 1.
Initialize: working set $\mathcal{W}\leftarrow\emptyset$; $c_i=1$, $\boldsymbol{y}'_i\leftarrow$ any element in $\mathcal{Y}$, for $i=1,\ldots,m$.
Repeat
  – $\mathcal{W}\leftarrow\mathcal{W}\cup\{(c_1,\ldots,c_m,\boldsymbol{y}'_1,\ldots,\boldsymbol{y}'_m)\}$.
  – Obtain primal and dual solutions $\boldsymbol{w},\xi$; $\boldsymbol{\lambda}$ by solving

$$\begin{aligned}
&\min_{\boldsymbol{w}\geq 0,\xi\geq 0}\ \mathbf{1}^\top\boldsymbol{w} + C\xi \\
&\text{s.t.:}\ \forall(c_1,\ldots,c_m,\boldsymbol{y}'_1,\ldots,\boldsymbol{y}'_m)\in\mathcal{W}: \\
&\qquad \frac{1}{m}\boldsymbol{w}^\top\Big[\sum_{i=1}^m c_i\cdot\delta\Psi_i(\boldsymbol{y}'_i)\Big] \geq \frac{1}{m}\sum_{i=1}^m c_i\Delta(\boldsymbol{y}_i,\boldsymbol{y}'_i) - \xi.
\end{aligned}$$

  – **For** $i=1,\ldots,m$
    $\boldsymbol{y}'_i = \operatorname{argmax}_{\boldsymbol{y}}\Delta(\boldsymbol{y}_i,\boldsymbol{y}) - \boldsymbol{w}^\top\delta\Psi_i(\boldsymbol{y})$;
    $c_i = \begin{cases} 1 & \Delta(\boldsymbol{y}_i,\boldsymbol{y}'_i) - \boldsymbol{w}^\top\delta\Psi_i(\boldsymbol{y}'_i) > 0; \\ 0 & \text{otherwise.}\end{cases}$
  End for
Until $\frac{1}{m}\boldsymbol{w}^\top\Big[\sum_{i=1}^m c_i\delta\Psi_i(\boldsymbol{y}'_i)\Big] \geq \frac{1}{m}\sum_{i=1}^m c_i\Delta(\boldsymbol{y}_i,\boldsymbol{y}'_i) - \xi - \epsilon_{\text{cp}}$.
  – Update $\mu_{(i,\boldsymbol{y})} = \sum_{\boldsymbol{c}}\lambda_{(\boldsymbol{c},\boldsymbol{y})}c_i$ for $\forall i=1,\ldots,m$; $\forall\boldsymbol{y}\in\mathcal{Y}\backslash\boldsymbol{y}_i$.
Output: $\boldsymbol{w}$ and $\boldsymbol{\mu}$.

2.2 Cutting-plane optimization for the 1-slack primal

Despite the extra nonnegative-ness constraint $\boldsymbol{w}\geq 0$ in our case, it is straightforward to apply the cutting-plane method in [10] for solving our problem (38). The cutting-plane algorithm for StructBoost is presented in Algorithm 2. A key step in Algorithm 2 is to solve the maximization for finding an output $\boldsymbol{y}'$ that corresponds to the most violated constraint for every $\boldsymbol{x}_i$ (inference step):

$$\boldsymbol{y}'_i = \operatorname*{argmax}_{\boldsymbol{y}}\ \Delta(\boldsymbol{y}_i,\boldsymbol{y}) + \boldsymbol{w}^\top\delta\Psi_i(\boldsymbol{y}). \tag{12}$$

The above maximization problem takes a similar form as the output prediction problem in (2). They only differ in the loss term $\Delta(\boldsymbol{y}_i,\boldsymbol{y})$. Typically these two problems can be solved using the same strategy. This inference step usually dominates the running time for a few applications, e.g., in the application of image segmentation. In the experiment section, we empirically show that solving (38) using cutting planes can be significantly faster than solving (35). Here improved cutting-plane methods such as [31] can also be adapted to solve our optimization problem at each column generation boosting iteration.

In terms of implementation of the cutting-plane algorithm, as mentioned in SSVM [30], a variety of design decisions can have substantial influence on the practical efficiency of the algorithm. We have considered some of

these design decisions in our implementation. In our case, we need to call the cutting-plane optimization at each column generation iteration. Consideration of warm-start initialization between two consecutive column generations can substantially speed up the training. We re-use the working set in the cutting-plane algorithm from previous column generation iterations. Finding a new weak learner in (11) is based on the dual solution $\boldsymbol{\mu}$. We need to ensure that the solution of cutting-plane is able to reach a sufficient precision, such that the generated weak learner is able to "make progress". Thus, we can adapt the stopping criterion parameter ϵ_{cp} (Line 10 of Algorithm 2) according to the cutting-plane precision in the last column generation iteration.

2.3 Discussion

Let us have a close look at the StructBoost algorithm in Algorithm 1. We can see that the training loop in Algorithm 1 is almost identical to other fully-corrective boosting methods (e.g., LPBoost [28] and Shen and Li [8]). Line 4 finds the most violated constraint in the dual form and add a new weak structured learner to the master problem. The dual solution $\mu_{(i,\boldsymbol{y})}$ defined in (11) plays the role as the example weight associated to each training example in conventional boosting methods such as AdaBoost and LPBoost [28]. Then Line 5 solves the master problem, which is the reduced problem of (35). Here we can see that, the cutting-plane in Algorithm 2 only serves as a solver for solving the master problem in Line 5 of Algorithm 1. This makes our StructBoost framework flexible—we are able to replace the cutting-plane optimization by other optimization methods. For example, the bundle methods in [32] may further speed up the computation.

For the convergence properties of the cutting-plane algorithm in Algorithm 2, readers may refer to [10] and [30] for details.

Our column generation algorithm in Algorithm 1 is a constraint generation algorithm for the dual problem in (37). We can adapt the analysis of the standard constraint generation algorithm for Algorithm 1. In general, for general column generation methods, the global convergence can be established but it remains unclear about the convergence rate if no particular assumptions are made. See the supplementary document[1] for details.

3 Examples of StructBoost

We consider a few applications of the proposed general structured boosting in this section, namely binary classification, ordinal regression, multi-class classification, optimization of Pascal overlap score, and CRF parameter learning. We show the particular setup for each application.

3.1 Binary classification

As the simplest example, the LPBoost of [28] for binary classification can be recoverd as follows. The label set is $\mathcal{Y} = \{+1, -1\}$; and $\Psi(\boldsymbol{x}, y) = \frac{1}{2} y \Phi(\boldsymbol{x})$. The label cost can be a simple constant; for exmaple, $\Delta(y, y') = 1$ for $y \neq y'$ and 0 for $y = y'$. Here we have introduced a column vector $\Phi(\boldsymbol{x})$:

$$\Phi(\boldsymbol{x}) = [\varphi_1(\boldsymbol{x}), ..., \varphi_n(\boldsymbol{x})]^\top, \tag{13}$$

which is the outputs of all weak classifiers $\varphi_1(\cdot)$, $\cdots$, $\varphi_n(\boldsymbol{x})$ on example $\boldsymbol{x}$. The output of a weak classifier, e.g., a decision stump or tree, usually is a binary value: $\varphi(\cdot) \in \{+1, -1\}$. In kernel methods, this feature mapping $\Phi(\cdot)$ is only known through the so-called kernel trick. Here we explicitly learn this feature mapping. Note that if $\Phi(\boldsymbol{x}) = \boldsymbol{x}$, we have the standard linear SVM.

3.2 Ordinal regression and AUC optimization

In ordinal regression, labels of the training data are ranks. Let us assume that the label $y \in \mathbb{R}$ indicates an ordinal scale, and pairs (i, j) in the set $\mathcal{S}$ has the relationship of example i being ranked higher than j, i.e., $y_i \succ y_j$. The primal can be written as

$$\min_{\boldsymbol{w} \geq 0, \boldsymbol{\xi} \geq 0} \ \mathbf{1}^\top \boldsymbol{w} + \tfrac{C}{m} \textstyle\sum_{(i,j) \in \mathcal{S}} \xi_{ij} \tag{14a}$$

$$\text{s.t.:} \ \ \boldsymbol{w}^\top \Big[\Phi(\boldsymbol{x}_i) - \Phi(\boldsymbol{x}_j) \Big] \geq 1 - \xi_{ij}, \forall (i, j) \in \mathcal{S}. \tag{14b}$$

Here $\Phi(\cdot)$ is defined in (13). Note that (14) also optimizes the area under the receiver operating characteristic (ROC) curve (AUC) criterion. As pointed out in [33], (14) is an instance of the multiclass classification problem. We discuss how the multiclass classification problem fits in our framework shortly.

Here, the number of constraints is quadratic in the number of training examples. Directly solving (14) can only solve problems with up to a few thousand training examples. We can reformulate (14) into an equivalent 1-slack problem, and apply the proposed StructBoost framework to solve the optimization more efficiently.

3.3 Multi-class boosting

The MultiBoost algorithm of Shen and Hao [7] can be implemented by the StructBoost framework as follows. Let $\mathcal{Y} = \{1, 2, \ldots, k\}$ and $\boldsymbol{w} = \boldsymbol{w}_1 \odot \cdots \odot \boldsymbol{w}_k$. Here $\odot$ stacks two vectors. As in [7], $\boldsymbol{w}_y$ is the model parameter associated with the y-th class. The multi-class discriminant function in [7] writes $F(\boldsymbol{x}, y; \boldsymbol{w}) = \boldsymbol{w}_y^\top \Phi(\boldsymbol{x})$. Now let us define the orthogonal label coding vector:

$$\Gamma(y) = [\mathbb{1}(y, 1), \cdots, \mathbb{1}(y, k)]^\top \in \{0, 1\}^k. \tag{15}$$

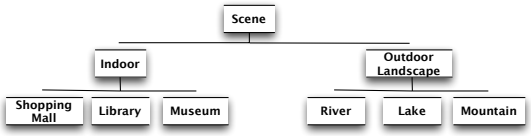
Here $\mathbb{1}(y, z)$ is the indicator function defined as:

$$\mathbb{1}(y, z) = \begin{cases} 1 & \text{if } y = z, \\ 0 & \text{if } y \neq z. \end{cases} \tag{16}$$

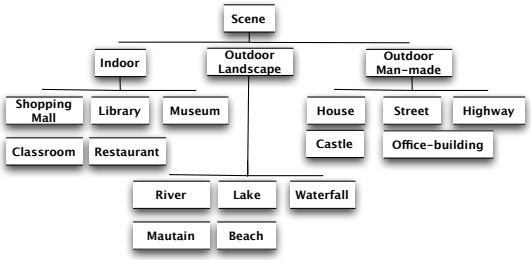
Then the following joint mapping function

$$\Psi(\boldsymbol{x}, y) = \Phi(\boldsymbol{x}) \otimes \Gamma(y)$$

1. Available at http://arxiv.org/abs/1302.3283



(a) Taxonomy of the 6-scene dataset



(b) Taxonomy of the 15-scene dataset

Fig. 1: The hierarchy structures of two selected subsets of the SUN dataset [34] used in our experiments for hierarchical image classification.

recovers the StructBoost formulation (35) for multi-class boosting. The operator $\otimes$ calculates the tensor product. The multi-class learning can be formulated as

$$\min_{\boldsymbol{w}\geq 0,\boldsymbol{\xi}\geq 0} \quad \mathbf{1}^\top \boldsymbol{w} + \tfrac{C}{m}\,\mathbf{1}^\top \boldsymbol{\xi} \tag{17a}$$

$$\text{s.t.:} \quad \boldsymbol{w}_{y_i}^\top \Phi(\boldsymbol{x}_i) - \boldsymbol{w}_y^\top \Phi(\boldsymbol{x}_i) \geq 1 - \xi_i, \quad \forall i = 1,\ldots,m; \text{ and } \forall y \in \{1,\ldots,k\}. \tag{17b}$$

A new weak classifier $\varphi(\cdot)$ is generated by solving the argmax problem defined in (11), which can be written as:

$$\varphi^\star(\cdot) = \operatorname*{argmax}_{\varphi(\cdot),y} \sum_{i,y} \mu_{(i,y)} \Big[\varphi(\boldsymbol{x}_i)\otimes\Gamma(y_i) - \varphi(\boldsymbol{x}_i)\otimes\Gamma(y))\Big]. \tag{18}$$

3.4 Hierarchical classification with taxonomies

In many applications such as object categorization and document classification [35], classes of objects are organized in taxonomies or hierarchies. For example, the ImageNet dataset has organized all the classes according to the tree structures of WordNet [36]. This problem is a classification example that the output space has interdependent structures. An example tree structure (taxonomy) of image categories is shown in Figure 1.

Now we consider the taxonomy to be a tree, with a partial order $\prec$, which indicates if a class is a predecessor of another class. We override the indicator function, which indicates if z is a predecessor of y in a label tree:

$$\mathbb{1}(y,z) = \begin{cases} 1 & y \prec z \text{ or } y = z, \\ 0 & \text{otherwise.} \end{cases} \tag{19}$$

The label coding vector has the same format as in the standard multi-class classification case:

$$\Gamma(y) = [\mathbb{1}(y,1),\cdots,\mathbb{1}(y,k)]^\top \in \{0,1\}^k. \tag{20}$$

Thus $\Gamma(y)^\top\Gamma(y')$ counts the number of common predecessors, while in the case of standard multi-class classification, $\Gamma(y)^\top\Gamma(y') = 0$ for $y \neq y'$.

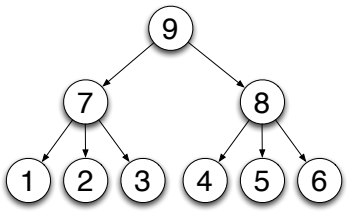


Fig. 2: Classification with taxonomies (tree loss), corresponding to the first example in Figure 1.

Figure 2 shows an example of the label coding vector for a given label hierarchy. In this case, for example, for class 3, $\Gamma(3) = [0,0,1,0,0,0,1,0,1]^\top$. The joint mapping function is $\Psi(\boldsymbol{x},y) = \Phi(\boldsymbol{x})\otimes\Gamma(y)$. The tree loss function $\Delta(y,y')$ is the height of the first common ancestor of the arguments y, y' in the tree. By re-defining $\Gamma(y)$ and $\Delta(y,y')$, classification with taxonomies can be immediately implemented using the standard multi-class classification shown in the last subsection.

Here we also consider an alternative approach. In [37], the authors show that one can train a multi-class boosting classifier by projecting data to a label-specific space and then learn a single model parameter $\boldsymbol{w}$. The main advantage might be that the optimization of $\boldsymbol{w}$ is simplified. Similar to [37] we define label-augmented data as $\boldsymbol{x}'_y = \boldsymbol{x}\otimes\Gamma(y)$. The max-margin classification can be written as

$$\min_{\boldsymbol{w}\geq 0,\boldsymbol{\xi}\geq 0} \quad \mathbf{1}^\top \boldsymbol{w} + \tfrac{C}{m}\mathbf{1}^\top \boldsymbol{\xi}$$

$$\text{s.t.:} \quad \boldsymbol{w}^\top \Big[\Phi(\boldsymbol{x}'_{i,y_i}) - \Phi(\boldsymbol{x}'_{i,y})\Big] \geq \Delta(y_i,y) - \xi_i, \quad \forall i = 1,\cdots,m; \text{ and } \forall y.$$

Compared with the first approach, now the model $\boldsymbol{w} \in \mathbb{R}^n$, which is independent of the number of classes.

3.5 Optimization of the Pascal image overlap criterion

Object detection/localization has used the image area overlap as the loss function [1]–[3], e.g, in the Pascal object detection challenge:

$$\Delta(\boldsymbol{y},\boldsymbol{y}') = 1 - \frac{\text{area}(\boldsymbol{y}\cap\boldsymbol{y}')}{\text{area}(\boldsymbol{y}\cup\boldsymbol{y}')}, \tag{21}$$

with $\boldsymbol{y},\boldsymbol{y}'$ being the bounding box coordinates. $\boldsymbol{y}\cap\boldsymbol{y}'$ and $\boldsymbol{y}\cup\boldsymbol{y}'$ are the box intersection and union. Let $\boldsymbol{x}_{\boldsymbol{y}}$ denote an image patch defined by a bounding box $\boldsymbol{y}$ on the image $\boldsymbol{x}$. To apply StructBoost, we define $\Psi(\boldsymbol{x},\boldsymbol{y}) = \Phi(\boldsymbol{x}_{\boldsymbol{y}})$. $\Phi(\cdot)$ is defined in (13). Weak learners such as classifiers or regressors $\varphi(\cdot)$ are trained on the image features extracted from image patches. For example, we can extract histograms of oriented gradients (HOG) from the image patch $\boldsymbol{x}_{\boldsymbol{y}}$ and train a decision stump with the extracted HOG features by solving the argmax in (11).

Note that in this case, solving (12), which is to find the most violated constraint in the training step as well as the inference for prediction (2), is in general difficult. In [3], a branch-and-bound search has been employed to find the

global optimum. Following the simple sampling strategy in [1], we simplify this problem by evaluating a certain number of sampled image patches to find the most violated constraint. It is also the case for prediction.

3.6 CRF parameter learning

CRF has found successful applications in many vision problems such as pixel labelling, stereo matching and image segmentation. Previous work often uses tedious cross-validation for setting the CRF parameters. This approach is only feasible for a small number of parameters. Recently, SSVM has been introduced to learn the parameters [16]. We demonstrate here how to employ the proposed StructBoost for CRF parameter learning in the image segmentation task. We demonstrate the effectiveness of our approach on the Graz-02 image segmentation dataset.

To speed up computation, super-pixels rather than pixels have been widely adopted in image segmentation. We define $\boldsymbol{x}$ as an image and $\boldsymbol{y}$ as the segmentation labels of all super-pixels in the image. We consider the energy E of an image $\boldsymbol{x}$ and segmentation labels $\boldsymbol{y}$ over the nodes $\mathcal{N}$ and edges $\mathcal{S}$, which takes the following form:

$$E(\boldsymbol{x},\boldsymbol{y};\boldsymbol{w}) = \sum_{p\in\mathcal{N}} \boldsymbol{w}^{(1)\top}\Phi^{(1)}\left(\mathbf{U}(y^p,\boldsymbol{x})\right) + \sum_{(p,q)\in\mathcal{S}} \boldsymbol{w}^{(2)\top}\Phi^{(2)}\left(\mathbf{V}(y^p,y^q,\boldsymbol{x})\right). \quad (22)$$

Recall that $\mathbb{1}(\cdot,\cdot)$ is the indicator function defined in (16). p and q are the super-pixel indices; y^p, y^q are the labels of the super-pixel p, q. $\mathbf{U}$ is a set of unary potential functions: $\mathbf{U} = [U_1, U_2, \dots]^\top$. $\mathbf{V}$ is a set of pairwise potential functions: $\mathbf{V} = [V_1, V_2, \dots]^\top$. When we learn the CRF parameters, the learning algorithm sees only $\mathbf{U}$ and $\mathbf{V}$. In other words $\mathbf{U}$ and $\mathbf{V}$ play the role as the input features. Details on how to construct $\mathbf{U}$ and $\mathbf{V}$ are described in the experiment part. $\boldsymbol{w}^{(1)}$ and $\boldsymbol{w}^{(2)}$ are the CRF potential weighting parameters that we aim to learn. $\Phi^{(1)}(\cdot)$ and $\Phi^{(2)}(\cdot)$ are two sets of weak learners (e.g., decision stumps) for the unary part and pairwise part respectively: $\Phi^{(1)}(\cdot) = [\varphi_1^{(1)}(\cdot), \varphi_2^{(1)}(\cdot), \dots]^\top$, $\Phi^{(2)}(\cdot) = [\varphi_1^{(2)}(\cdot), \varphi_2^{(2)}(\cdot), \dots]^\top$.

To predict the segmentation labels $\boldsymbol{y}^\star$ of an unknown image $\boldsymbol{x}$ is to solve the energy minimization problem:

$$\boldsymbol{y}^\star = \operatorname*{argmin}_{\boldsymbol{y}} E(\boldsymbol{x},\boldsymbol{y};\boldsymbol{w}), \quad (23)$$

which can be solved efficiently by using graph cuts [16], [38].

Consider a segmentation problem with two classes (background versus foreground). It is desirable to keep the submodular property of the energy function in (22). Otherwise graph cuts cannot be directly applied to achieve globally optimal labelling. Let us examine the pairwise energy term:

$$\theta_{(p,q)}(y^p,y^q) = \boldsymbol{w}^{(2)\top}\Phi^{(2)}\left(\mathbf{V}(y^p,y^q,\boldsymbol{x})\right),$$

and a super-pixel label $y \in \{0,1\}$. It is well known that, if the following is satisfied for any pair $(p,q) \in \mathcal{S}$, the energy function in (22) is submodular:

$$\theta_{(p,q)}(0,0) + \theta_{(p,q)}(1,1) \le \theta_{(p,q)}(0,1) + \theta_{(p,q)}(1,0). \quad (24)$$

We want to keep the above property.

First, for a weak learner $\varphi^{(2)}(\cdot)$, we enforce it to output 0 when two labels are identical. This can always be achieved by multiplying $(1 - \mathbb{1}(y^p, y^q))$ to a conventional weak learner. Now we have $\theta_{(p,q)}(0,0) = \theta_{(p,q)}(1,1) = 0$.

Given that the nonnegative-ness of $\boldsymbol{w}$ is enforced in our model, now a sufficient condition is that the output of a weak learner $\varphi^{(2)}(\cdot)$ is always nonnegative, which can always be achieved. We can always use a weak learner $\varphi^{(2)}(\cdot)$ which takes a nonnegative output, e.g., a discrete decision stump or tree with outputs in $\{0,1\}$.

By applying weak learners on $\mathbf{U}$ and $\mathbf{V}$, our method introduces nonlinearity for the parameter learning, which is different from most linear CRF learning methods such as [16]. Until recently, Berteli et al. presented an image segmentation approach that uses nonlinear kernels for the unary energy term in the CRF model [14]. In our model, nonlinearity is introduced by applying weak learners on the potential functions' outputs. This is the same as the fact that an SVM introduces nonlinearity via the so-called kernel trick and boosting learns a nonlinear model with nonlinear weak learners. Nowozin et al. [39] introduced decision tree fields (DTF) to overcome the problem of overly-simplified modeling of pairwise potentials in most CRF models. In DTF, local interactions between multiple variables are determined by means of decision trees. In our StructBoost, if we use decision trees as the weak learners on the pairwise potentials, then StructBoost and DTF share similar characteristics in that both use decision trees for the same purpose. However, the starting points of these two methods as well as the training procedures are entirely different.

To apply StructBoost, the CRF parameter learning problem in a large-margin framework can then be written as:

$$\min_{\boldsymbol{w}\ge 0,\boldsymbol{\xi}\ge 0} \ \mathbf{1}^\top\boldsymbol{w} + \tfrac{C}{m}\mathbf{1}^\top\boldsymbol{\xi} \quad (25a)$$

$$\text{s.t.:} \ E(\boldsymbol{x}_i,\boldsymbol{y};\boldsymbol{w}) - E(\boldsymbol{x}_i,\boldsymbol{y}_i;\boldsymbol{w}) \ge \Delta(\boldsymbol{y}_i,\boldsymbol{y}) - \xi_i, \quad \forall i = 1,\dots,m; \ \text{and} \ \forall \boldsymbol{y} \in \mathcal{Y}. \quad (25b)$$

Here i indexes images. Intuitively, the optimization in (25) is to encourage the energy of the ground truth label $E(\boldsymbol{x}_i,\boldsymbol{y}_i)$ to be lower than any other *incorrect* labels $E(\boldsymbol{x}_i,\boldsymbol{y})$ by at least a margin $\Delta(\boldsymbol{y}_i,\boldsymbol{y})$, $\forall \boldsymbol{y}$. We simply define $\Delta(\boldsymbol{y}_i,\boldsymbol{y})$ using the Hamming loss, which is the sum of the differences between the ground truth label $\boldsymbol{y}_i$ and the label $\boldsymbol{y}$ over all super-pixels in an image:

$$\Delta(\boldsymbol{y}_i,\boldsymbol{y}) = \sum_p (1 - \mathbb{1}(y_i^p, y^p)). \quad (26)$$

We show the problem (25) is a special case of the general formulation of StructBoost (35) by defining

$$\boldsymbol{w} = -\boldsymbol{w}^{(1)} \odot \boldsymbol{w}^{(2)},$$

and,

$$\Psi(\boldsymbol{x},\boldsymbol{y}) = \sum_{p\in\mathcal{N}} \Phi^{(1)}\left(\mathbf{U}(y^p,\boldsymbol{x})\right) \odot \sum_{(p,q)\in\mathcal{S}} \Phi^{(2)}\left(\mathbf{V}(y^p,y^q,\boldsymbol{x})\right).$$

Recall that $\odot$ stacks two vectors. With this definition, we have the relation:

$$\boldsymbol{w}^\top \Psi(\boldsymbol{x},\boldsymbol{y}) = -E(\boldsymbol{x},\boldsymbol{y};\boldsymbol{w}).$$

The minus sign here is to inverse the order of subtraction in (35b). At each column generation iteration (Algorithm 1), two new weak learners $\varphi^{(1)}(\cdot)$ and $\varphi^{(2)}(\cdot)$ are added to the unary weak learner set and the pairwise weak learner set, respectively by solving the argmax problem defined in (11), which can be written as:

$$\begin{aligned}\varphi^{(1)\star}(\cdot) = &\operatorname*{argmax}_{\varphi(\cdot)} \sum_{i,\boldsymbol{y}} \mu_{(i,\boldsymbol{y})} \sum_{p\in\mathcal{N}} \Big[\varphi^{(1)}\left(\mathbf{U}(y^p,\boldsymbol{x}_i)\right) \\ &- \varphi^{(1)}\left(\mathbf{U}(y_i^p,\boldsymbol{x}_i)\right)\Big]; \qquad (27)\\ \varphi^{(2)\star}(\cdot) = &\operatorname*{argmax}_{\varphi(\cdot)} \sum_{i,\boldsymbol{y}} \mu_{(i,\boldsymbol{y})} \sum_{(p,q)\in\mathcal{S}} \Big[\varphi^{(2)}\left(\mathbf{V}(y^p,y^q,\boldsymbol{x}_i)\right) \\ &- \varphi^{(2)}\left(\mathbf{V}(y_i^p,y_i^q,\boldsymbol{x}_i)\right)\Big]. \qquad (28)\end{aligned}$$

The maximization problem to find the most violated constraint in (12) is to solve the inference:

$$\boldsymbol{y}_i' = \operatorname*{argmin}_{\boldsymbol{y}} E(\boldsymbol{x}_i,\boldsymbol{y}) - \Delta(\boldsymbol{y}_i,\boldsymbol{y}), \qquad (29)$$

which is similar to the label prediction inference in (23), and the only difference is that the labeling loss term: $\Delta(\boldsymbol{y}_i,\boldsymbol{y})$ is involved in (29). Recall that we use the Hamming loss $\Delta(\boldsymbol{y}_i,\boldsymbol{y})$ as defined in (26), the term $\Delta(\boldsymbol{y}_i,\boldsymbol{y})$ can be absorbed into the unary term of the energy function defined in (22) (such as in [16]). The inference in (29) can be written as:

$$\begin{aligned}\boldsymbol{y}_i' = \operatorname*{argmin}_{\boldsymbol{y}} &\sum_{p\in\mathcal{N}} \left[\boldsymbol{w}^{(1)\top}\Phi^{(1)}\left(\mathbf{U}(y^p,\boldsymbol{x}_i)\right) - \left(1-\mathbb{1}(y_i^p,y^p)\right)\right] \\ &+ \sum_{(p,q)\in\mathcal{S}} \boldsymbol{w}^{(2)\top}\Phi^{(2)}\left(\mathbf{V}(y^p,y^q,\boldsymbol{x}_i)\right). \qquad (30)\end{aligned}$$

The above minimization (30) can also be solved efficiently by using graph cuts.

4 Experiments

To evaluate our method, we run various experiments on applications including AUC maximization (ordinal regression), multi-class image classification, hierarchical image classification, visual tracking and image segmentation. We mainly compare with the most relevant method: Structured SVM (SSVM) and some other conventional methods (e.g., SVM, AdaBoost). If not otherwise specified, the cutting-plane stopping criteria (ϵ_{cp}) in our method is set to 0.01.

TABLE 1: AUC maximization. We compare the performance of m-slack and 1-slack formulations. "$-$" means that the method is not able to converge within a memory and time limit. We can see that 1-slack can achieve similar AUC results on training and testing data as m-slack while 1-slack is significantly faster than m-slack.

dataset	method	time (sec)	AUC training	AUC test
wine	m-slack	13±1	1.000±0.000	0.994±0.005
	1-slack	3±1	1.000±0.000	0.994±0.006
glass	m-slack	20±1	0.967±0.011	0.849±0.028
	1-slack	6±1	0.955±0.030	0.844±0.039
svmguide2	m-slack	332±6	0.988±0.003	0.905±0.036
	1-slack	106±8	0.988±0.003	0.905±0.036
svmguide4	m-slack	564±79	1.000±0.000	0.982±0.005
	1-slack	106±13	1.000±0.000	0.982±0.005
vowel	m-slack	4051±116	0.999±0.001	0.968±0.013
	1-slack	952±139	0.999±0.001	0.967±0.013
dna	m-slack	–	–	–
	1-slack	1598±643	0.998±0.000	0.992±0.003
segment	m-slack	–	–	–
	1-slack	475±42	1.000±0.000	0.999±0.001
satimage	m-slack	–	–	–
	1-slack	37769±6331	0.999±0.000	0.997±0.002

4.1 AUC optimization

In this experiment, we compare efficiency of the 1-slack (solving (38)) and m-slack (solving (35) or its dual) formulations of our method StructBoost. The details for AUC optimization are described in Section 3.2. We run the experiments on a few UCI multi-class datasets. To create imbalanced data, we use one class of the multi-class UCI datasets as positive data and the rest as negative data. The boosting (column generation) iteration is set to 200; the cutting-plane stopping criterion (ϵ_{cp}) is set to 0.001. Decision stumps are used as weak learners ($\Phi(\cdot)$ in (14)). For each data set, we randomly sample 50% for training, 25% for validation and the rest for testing. The regularization parameter C is chosen from 6 candidates ranging from 10 to 10^3. Experiments are repeated 5 times on each dataset and the mean and standard deviation are reported. Table 1 reports the results. We can see that the 1-slack formulation of StructBoost achieves similar AUC performance as the m-slack formulation, while being much faster than m-slack. The values of objective function and optimization time are shown in Figure 3 by varying column generation iterations. Again, it shows that 1-slack achieves similar objective values as m-slack with less running time.

Note that RankBoost may also be applied to this problem [40]. RankBoost has been designed for solving ranking problems.

4.2 Multi-class classification

Multi-class classification is a special case of structured learning. Details are described in Section 3.3. We carry out experiments on some UCI multi-class datasets and MNIST. We compare with Structured SVM (SSVM), conventional multi-class boosting methods (namely AdaBoost.ECC and AdaBoost.MH), and the one-vs-all SVM method. For each dataset, we randomly select 50% data for training, 25% data for validation and the rest for testing. The maximum number of boosting iterations is set to 200; the regularization parameter C is chosen from 6 candidates whose values

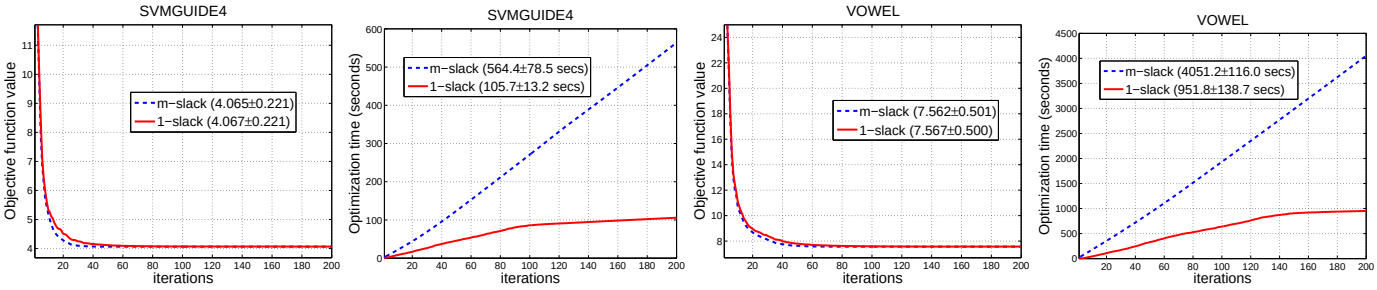


Fig. 3: AUC optimization on two UCI datasets. The objective values and optimization time are shown in the figure by varying boosting (or column generation) iterations. It shows that 1-slack achieves similar objective values as m-slack but needs less running time.

TABLE 2: Multi-class classification test errors (%) on several UCI and MNIST datasets. StructBoost-stump and StructBoost-per denote our StructBoost using decision stumps and perceptrons as weak learners, respectively. StructBoost outperforms SSVM in most cases and achieves competitive performance compared with other multi-class classifiers.

	glass	svmguide2	svmguide4	vowel	dna	segment	satimage	usps	pendigits	mnist
StructBoost-stump	35.8 ± 6.2	21.0 ± 3.9	20.1 ± 2.9	17.5 ± 2.2	**6.2 ± 0.7**	**2.9 ± 0.7**	12.1 ± 0.7	6.9 ± 0.6	3.9 ± 0.4	12.5 ± 0.4
StructBoost-per	37.3 ± 6.2	22.7 ± 4.8	53.4 ± 6.1	**6.8 ± 1.8**	6.6 ± 0.6	3.8 ± 0.7	**11.4 ± 1.1**	**4.1 ± 0.6**	**1.8 ± 0.3**	**6.5 ± 0.6**
Ada.ECC	32.7 ± 4.9	23.3 ± 4.0	**19.1 ± 2.3**	20.6 ± 1.5	7.6 ± 1.2	**2.9 ± 0.8**	12.8 ± 0.7	8.4 ± 0.7	8.4 ± 0.7	15.8 ± 0.3
Ada.MH	**32.3 ± 5.0**	21.9 ± 4.5	19.3 ± 3.0	18.8 ± 2.1	7.1 ± 0.6	3.7 ± 0.7	12.7 ± 0.9	7.4 ± 0.5	7.4 ± 0.5	13.4 ± 0.4
SSVM	38.8 ± 8.7	21.9 ± 5.9	45.7 ± 3.9	25.6 ± 2.5	6.9 ± 0.9	5.3 ± 1.0	14.9 ± 0.1	5.8 ± 0.3	5.2 ± 0.3	9.6 ± 0.2
1-vs-all SVM	40.8 ± 7.0	**17.7 ± 3.5**	47.0 ± 3.2	54.4 ± 2.2	6.3 ± 0.5	7.7 ± 0.8	17.5 ± 0.4	5.4 ± 0.5	8.1 ± 0.5	9.2 ± 0.2

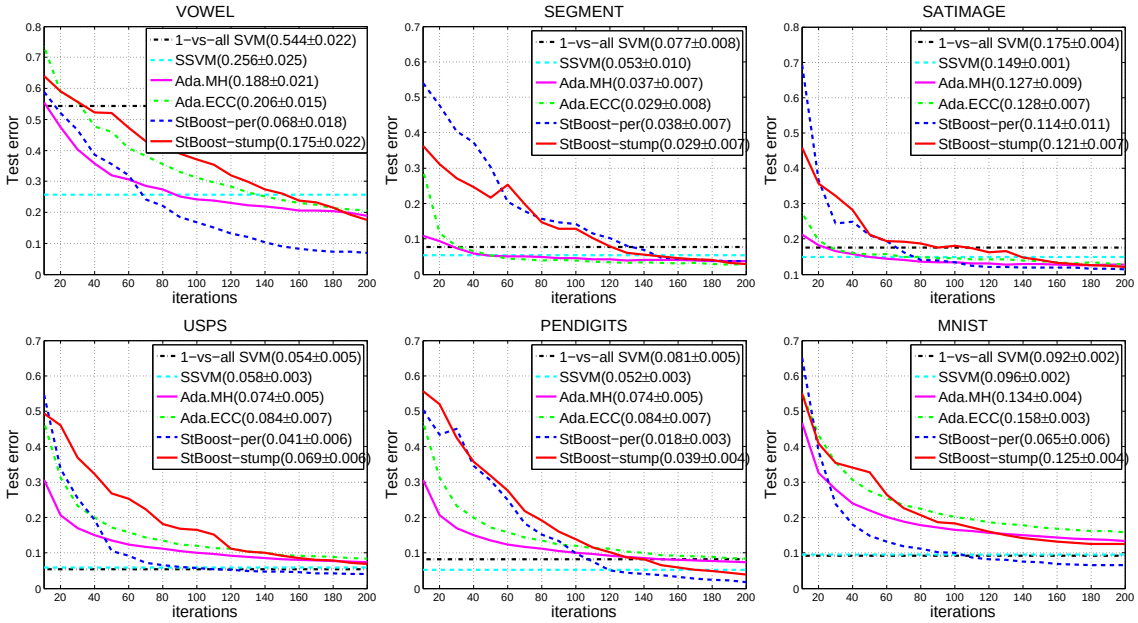


Fig. 4: Test performance versus the number of boosting iterations of multi-class classification. StBoost-stump and StBoost-per denote our StructBoost using decision stumps and perceptrons as weak learners, respectively. The results of SSVM and SVM are shown as straight lines in the plots. The values shown in the legend are the error rates of the final iteration for each method. Our methods perform better than SSVM in most cases.

range from 10 to 1000. The experiments are repeated 5 times for each dataset.

To demonstrate the flexibility of our method, we use decision stumps and linear perceptron functions as weak learners ($\Phi(\cdot)$ in (17)). The perceptron weak learner can be written as:

$$\varphi(\boldsymbol{x}) = \text{sign}(\boldsymbol{v}^\top \boldsymbol{x} + b). \tag{31}$$

We use a smooth sigmoid function to replace the step function so that gradient descent optimization can be applied. We solve the argmax problem in (18) by using the Quasi-Newton LBFGS [41] solver. We found that decision stumps often provide a good initialization for LBFGS learning of the perceptron. Compared with decision stumps, using the perceptron weak learner usually needs fewer boosting iterations to converge. Table 2 reports the error rates. Figure 4 shows test performance versus the number of boosting (column generation) iterations. The results demonstrate that our method outperforms SSVM, and achieves competitive performance compared with other conventional multi-class methods.

StructBoost performs better than SSVM on most datasets. This might be due to the introduction of non-linearity in StructBoost. Results also show that using the perceptron weak learner often achieves better performance than using

TABLE 3: Hierarchical classification. Results of the tree loss and the 1/0 loss (classification error rate) on subsets of the SUN dataset. StructBoost-tree uses the hierarchy class formulation with the tree loss, and StructBoost-flat uses the standard multi-class formulation. StructBoost-tree that minimizes the tree loss performs best.

datasets		StructBoost-tree	StructBoost-flat	Ada.ECC-SVM	Ada.ECC	Ada.MH
6 scenes	1/0 loss	0.322 ± 0.018	0.343 ± 0.028	0.350 ± 0.013	0.327 ± 0.002	**0.315 ± 0.015**
	tree loss	**0.337 ± 0.014**	0.380 ± 0.027	0.377 ± 0.018	0.352 ± 0.023	0.346 ± 0.018
15 scenes	1/0 loss	**0.394 ± 0.005**	0.396 ± 0.013	0.414 ± 0.012	0.444 ± 0.012	0.418 ± 0.010
	tree loss	**0.504 ± 0.007**	0.536 ± 0.009	0.565 ± 0.019	0.584 ± 0.017	0.551 ± 0.013

decision stumps on those large datasets.

4.3 Hierarchical multi-class classification

The details of hierarchical multi-class are described in Section 3.4. We have constructed two hierarchical image datasets from the SUN dataset [34] which contains 6 classes and 15 classes of scenes. The hierarchical tree structures of these two datasets are shown in the Figure 1. For each scene class, we use the first 200 images from the original SUN dataset. There are 1200 images in the first dataset and 3000 images in the second dataset. We have used the HOG features as described in [34]. For each dataset, 50% examples are randomly selected for training and the rest for testing. We run 5 times for each dataset. The regularization parameter is chosen from 6 candidates ranging from 1 to 10^3.

We use linear SVM as weak classifiers in our method. The linear SVM weak classifier has the same form as (31). At each boosting iteration, we solve the argmax problem by training a linear SVM model. The regularization parameter C in SVM is set to a large value (10^8/#examples). To alleviate the overfitting problem of using linear SVM as weak learners, we set 10% of the smallest non-zero dual solutions $\mu_{(i,\boldsymbol{y})}$ to zero.

We compare StructBoost using the hierarchical multi-class formulation (StructBoost-tree) and the standard multi-class formulation (StructBoost-flat). We run some other multi-class methods for further comparison: Ada.ECC, Ada.MH with decision stumps. We also run Ada.ECC using linear SVM as weak classifiers (labelled as Ada.ECC-SVM).

When using SVM as weak learners, the number of boosting iterations is set to 200, and for decision stump, it is set to 500. Table 3 shows the tree loss and the 1/0 loss (classification error rate). We observe that StructBoost-tree has the lowest tree loss among all compared methods, and it also improves its counterpart, StructBoost-flat, in terms of both classification error rates and the tree loss. Our StructBoost-tree makes use of the class hierarchy information and directly optimizes the tree loss. That might be the reason why StructBoost-tree achieves best performance.

4.4 Visual tracking

A visual tracking method, termed Struck [1], was introduced based on SSVM. The core idea is to train a tracker by optimizing the Pascal image overlap score using SSVM. Here we apply StructBoost to visual tracking, following the similar setting as in Struck [1]. More details can be found in Section 3.5.

Our experiment follows the on-line tracking setting. Here we use the first 3 labeled frames for initialization and training of our StructBoost tracker. Then the tracker is updated by re-training the model during the course of tracking. Specifically, in the i-th frame (represented by $\boldsymbol{x}_i$), we first perform a prediction step (solving (2)) to output the detection box ($\boldsymbol{y}_i$), then collect training data for tracker update. For solving the prediction inference in (2), we simply sample about 2000 bounding boxes around the prediction box of the last frame (represented by $\boldsymbol{y}_{i-1}$), and search for the most confident bounding box over all sampled boxes as the prediction. After the prediction, we collect training data by sampling about 200 bounding boxes around the current prediction $\boldsymbol{y}_i$. We use the training data in recent 60 frames to re-train the tracker for every 2 frames. Solving (12) for finding the most violated constraint is similar to the prediction inference.

For StructBoost, decision stumps are used as the weak classifiers; the number of boosting iterations is set to 300; the regularization parameter C is selected from $10^{0.5}$ to 10^2. We use the down-scaled gray-scale raw pixels and HOG [42] as image features.

For comparison, we also run a simple binary AdaBoost tracker using the same setting as our StructBoost tracker. The number of weak classifiers for AdaBoost is set to 500. When training the AdaBoost tracker, we collect positive boxes that significantly overlap (overlap score above 0.8) with the prediction box of the current frame, and negative boxes with small overlap scores (lower or equal to 0.3).

We also compare with a few state-of-the-art tracking methods, including Struck [1] (with a buffer size of 50), multi-instance tracking (MIL) [43], fragment tracking (Frag) [44], online AdaBoost tracking (OAB) [45], and visual tracking decomposition (VTD) [46]. Two different settings are used for OAB: one positive example per frame (OAB_1) and five positive examples per frame (OAB_5) for training. The test video sequences: "coke, tiger1, tiger2, david, girl and sylv" were used in [1]; "shaking, singer" are from [46] and the rest are from [47].

Table 4 reports the Pascal overlap scores of compared methods. *Our StructBoost tracker performs best on most sequences*. Compared with the simple binary AdaBoost tracker, StructBoost that optimizes the pascal overlap criterion perform significantly better. Note that here Struck uses Haar features. When Struck uses a Gaussian kernel defined on raw pixels, the performance is slightly different [1], and ours still outperforms Struck in most cases. This might be due to the fact that StructBoost selects relevant features (300 features selected here), while SSVM of Struck

[1] uses all the image patch information which may contain noises.

The center location errors (in pixels) of compared methods are shown in Table 5. We can see that optimizing the overlap score also helps to minimize the center location errors. Our method also achieves the best performance.

Figure 5 plots the Pascal overlap scores and central location errors frame by frame on several video sequences. Some tracking examples are shown in Figure 6.

4.5 CRF parameter learning for image segmentation

We evaluate our method on CRF parameter learning for image segmentation, following the work of [16]. The work of [16] applies SSVM to learn weighting parameters for different potentials (including multiple unary and pairwise potentials). The goal of applying StructBoost here is to learn a non-linear weighting for different potentials. Details are described in Section 3.6.

We extend the super-pixels based segmentation method [38] with CRF parameter learning. The Graz-02 dataset[2] is used here which contains 3 categories (bike, car and person). Following the setting as other methods [38], the first 300 labeled images in each category are used in the experiment. Images with the odd indices are used for training and the rest for testing. We generate super-pixels using the same setting as [38]. For each super-pixel, we generate 5 types of features: visual word histogram [38], color histograms, GIST [48], LBP[3] and HOG [42]. For constructing the visual word histogram, we follow [38] using a neighborhood size of 2; the code book size is set to 200. For GIST, LBP and HOG, we extract features from patches centered at the super-pixel with 4 increasing sizes: $4 \times 4, 8 \times 8, 12 \times 12$ and 16×16. The cell size for LBP and HOG is set to a quarter of the patch size. For GIST, we generate 512 dimensional features for each patch by using 4 scales and the number of orientations is set to 8. In total, we generate 14 groups of features (including features extracted on patches of different sizes). Using these super-pixel features, we construct 14 different unary potentials ($\mathbf{U} = [\,U_1, \ldots, U_{14}\,]^\top$) from AdaBoost classifiers, which are trained on the foreground and background super-pixels. The number of boosting iterations for AdaBoost is set to 1000. Specifically, we define $F'(\boldsymbol{x}^p)$ as the discriminant function of AdaBoost on the features of the p-th super-pixel. Then the unary potential function can be written as:

$$U(\boldsymbol{x}, y^p) = -y^p F'(\boldsymbol{x}^p). \tag{32}$$

We also construct 2 pairwise potentials ($\mathbf{V} = [\,V_1, V_2\,]^\top$): V_1 is constructed using color difference, and V_2 using shared boundary length [38] which is able to discourage small isolated segments. Recall that $\mathbb{1}(\cdot,\cdot)$ is an indicator function defined in (16). $\|\boldsymbol{x}^p - \boldsymbol{x}^q\|_2$ calculates the ℓ_2 norm of the color difference between two super-pixels in the LUV color-space; $\ell(\boldsymbol{x}^p, \boldsymbol{x}^q)$ is the shared boundary length between two super-pixels. Then V_1, V_2 can be written as:

$$V_1(y^p, y^q, \boldsymbol{x}) = \exp(-\|\boldsymbol{x}^p - \boldsymbol{x}^q\|_2)\big[1 - \mathbb{1}(y^p, y^q)\big], \tag{33}$$

$$V_2(y^p, y^q, \boldsymbol{x}) = \ell(\boldsymbol{x}^p, \boldsymbol{x}^q)\big[1 - \mathbb{1}(y^p, y^q)\big]. \tag{34}$$

We apply StructBoost here to learn non-linear weights for combining these potentials. We use decision stumps as weak learners ($\Phi(\cdot)$ in (22)) here. The number of boosting iterations for StructBoost is set to 50.

For comparison, we run SSVM to learn CRF weighting parameters on exactly the same potentials as our method. The regularization parameter C in SSVM and our StructBoost is chosen from 6 candidates with the value ranging from 0.1 to 10^3. We also run two simple binary super-pixel classifiers (linear SVM and AdaBoost) trained on visual word histogram features of foreground and background super-pixels. The regularization parameter C in SVM is chosen from 10^2 to 10^7. The number of boosting iterations for AdaBoost is set to 1000.

We use the intersection-union score, pixel accuracy (including the foreground and background) and $precision = recall$ value (as in [38]) for evaluation. Results are shown in Table 6. Some segmentation examples are shown in Figure 7. As shown in the results, both StructBoost and SSVM, which learn to combine different potential functions, are able to significantly outperform the simple binary models (AdaBoost and SVM). StructBoost outperforms SSVM since it learns a non-linear combination of potentials. Note that SSVM learns a linear weighting for different potentials. By employing nonlinear parameter learning, our method gains further performance improvement over SSVM.

TABLE 6: Image segmentation results on the Graz-02 dataset. The results show the the pixel accuracy, intersection-union score (including the foreground and background) and $precision = recall$ value (as in [38]). Our method StructBoost for nonlinear parameter learning performs better than SSVM and other methods.

	bike	car	people
	intersection/union (foreground, background) (%)		
AdaBoost	69.2 (57.6, 80.7)	72.2 (51.7, 92.7)	68.9 (51.2, 86.5)
SVM	65.2 (53.0, 77.4)	68.6 (45.0, 92.3)	62.9 (41.0, 84.8)
SSVM	74.5 (64.4, 84.6)	80.2 (64.9, 95.4)	74.3 (58.8, 89.7)
StructBoost	**76.5** (66.3, 86.7)	**80.8** (66.1, 95.6)	**75.7** (61.0, 90.4)
	pixel accuracy (foreground, background) (%)		
AdaBoost	84.4 (83.8, 85.1)	82.9 (69.8, 96.0)	81.0 (70.0, 92.1)
SVM	81.9 (81.8, 82.1)	77.0 (57.2, 96.9)	73.5 (53.8, 93.2)
SSVM	**87.9** (87.9, 88.0)	86.9 (75.8, 98.1)	83.5 (71.8, 95.2)
StructBoost	87.4 (83.3, 91.5)	**87.6** (77.0, 98.1)	**84.6** (73.6, 95.6)
	$precision = recall$ (%)		
M. & S. [49]	61.8	53.8	44.1
F. et al. [38]	72.2	72.2	66.3
AdaBoost	72.7	67.8	67.0
SVM	68.3	63.4	61.2
SSVM	77.3	78.3	74.4
StructBoost	**78.9**	**79.3**	**75.9**

5 Conclusion

we have presented a structured boosting method, which combines a set of weak structured learners for nonlinear structured output leaning, as an alternative to SSVM [4] and CRF [11]. Analogous to SSVM, where the discriminant function is learned over a joint feature space of inputs

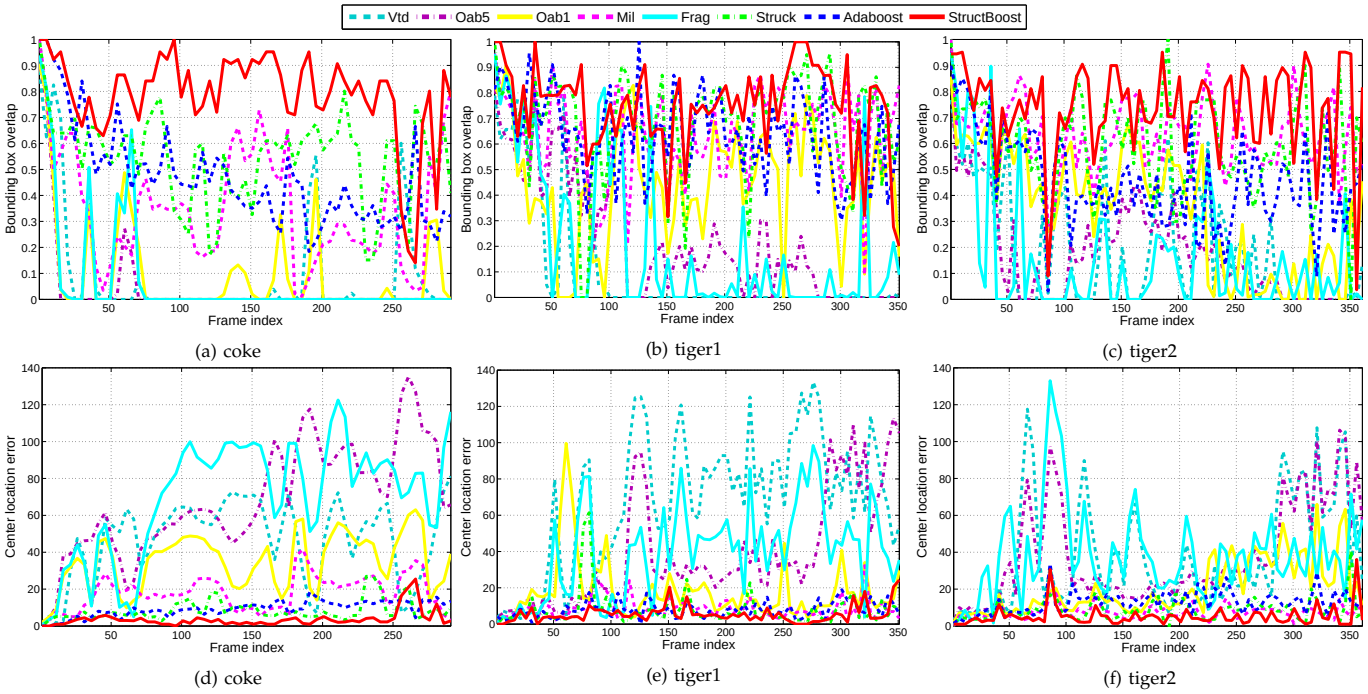
2. http://www.emt.tugraz.at/~pinz/
3. http://www.vlfeat.org/

TABLE 4: Average bounding box overlap scores on benchmark videos. Struck_{50} [1] is structured SVM tracking with a buffer size of 50. Our StructBoost performs the best in most cases. Struck performs the second best, which confirms the usefulness of structured output learning.

	StructBoost	AdaBoost	Struck_{50}	Frag	MIL	OAB_1	OAB_5	VTD
coke	**0.79 ± 0.17**	0.47 ± 0.19	0.55 ± 0.18	0.07 ± 0.21	0.36 ± 0.23	0.10 ± 0.20	0.04 ± 0.16	0.10 ± 0.23
tiger1	**0.75 ± 0.17**	0.64 ± 0.16	0.68 ± 0.21	0.21 ± 0.30	0.64 ± 0.18	0.44 ± 0.23	0.23 ± 0.24	0.11 ± 0.24
tiger2	**0.74 ± 0.18**	0.46 ± 0.18	0.59 ± 0.19	0.16 ± 0.24	0.63 ± 0.14	0.35 ± 0.23	0.18 ± 0.19	0.19 ± 0.22
david	**0.86 ± 0.07**	0.34 ± 0.23	0.82 ± 0.11	0.18 ± 0.24	0.59 ± 0.13	0.28 ± 0.23	0.21 ± 0.22	0.29 ± 0.27
girl	0.74 ± 0.12	0.41 ± 0.26	**0.80 ± 0.10**	0.65 ± 0.19	0.56 ± 0.21	0.43 ± 0.18	0.28 ± 0.26	0.63 ± 0.12
sylv	0.66 ± 0.16	0.52 ± 0.18	**0.69 ± 0.14**	0.61 ± 0.23	0.66 ± 0.18	0.47 ± 0.38	0.05 ± 0.12	0.58 ± 0.30
bird	**0.79 ± 0.11**	0.67 ± 0.14	0.60 ± 0.26	0.34 ± 0.32	0.58 ± 0.32	0.57 ± 0.29	0.59 ± 0.30	0.11 ± 0.26
walk	**0.74 ± 0.19**	0.56 ± 0.14	0.59 ± 0.39	0.09 ± 0.25	0.51 ± 0.34	0.54 ± 0.36	0.49 ± 0.34	0.08 ± 0.23
shaking	**0.72 ± 0.13**	0.49 ± 0.22	0.08 ± 0.19	0.33 ± 0.28	0.61 ± 0.26	0.57 ± 0.28	0.51 ± 0.21	0.69 ± 0.14
singer	0.69 ± 0.10	**0.74 ± 0.10**	0.34 ± 0.37	0.14 ± 0.30	0.20 ± 0.34	0.20 ± 0.33	0.07 ± 0.18	0.50 ± 0.20
iceball	**0.58 ± 0.17**	0.05 ± 0.16	0.51 ± 0.33	0.51 ± 0.31	0.35 ± 0.29	0.08 ± 0.23	0.38 ± 0.30	0.57 ± 0.29

TABLE 5: Average center errors on benchmark videos. Struck_{50} [1] is structured SVM tracking with a buffer size of 50. We observe similar results as in Table 4: Our StructBoost outperforms others on most sequences, and Struck is the second best.

	StructBoost	AdaBoost	Struck_{50}	Frag	MIL	OAB_1	OAB_5	VTD
coke	**3.7 ± 4.5**	9.3 ± 4.2	8.3 ± 5.6	69.5 ± 32.0	17.8 ± 9.6	34.7 ± 15.5	68.1 ± 30.3	46.8 ± 21.8
tiger1	**5.4 ± 4.9**	7.8 ± 4.4	7.8 ± 9.9	39.6 ± 25.7	8.4 ± 5.9	17.8 ± 16.4	38.9 ± 31.1	68.8 ± 36.4
tiger2	**5.2 ± 5.6**	12.7 ± 6.3	8.7 ± 6.1	38.5 ± 24.9	7.5 ± 3.6	20.5 ± 14.9	38.3 ± 26.9	38.0 ± 29.6
david	**5.2 ± 2.8**	43.0 ± 28.2	7.7 ± 5.7	73.8 ± 36.7	19.6 ± 8.2	51.0 ± 30.9	64.4 ± 33.5	66.1 ± 56.3
girl	14.3 ± 7.8	47.1 ± 29.5	**10.1 ± 5.5**	23.0 ± 22.5	31.6 ± 28.2	43.3 ± 17.8	67.8 ± 32.5	18.4 ± 11.4
sylv	9.1 ± 5.8	14.7 ± 7.8	**8.4 ± 5.3**	12.2 ± 11.8	9.4 ± 6.5	32.9 ± 36.5	76.4 ± 35.4	21.6 ± 35.7
bird	**6.7 ± 3.8**	12.7 ± 9.5	17.9 ± 13.9	50.0 ± 43.3	49.0 ± 85.3	47.9 ± 87.7	48.5 ± 86.3	143.9 ± 79.3
walk	**8.4 ± 10.3**	13.5 ± 5.4	33.9 ± 49.5	102.8 ± 46.3	35.0 ± 47.5	35.7 ± 49.2	38.0 ± 48.7	100.9 ± 47.1
shaking	**9.5 ± 5.4**	21.6 ± 12.0	123.9 ± 54.5	47.2 ± 40.6	37.8 ± 75.6	26.9 ± 49.3	29.1 ± 48.7	10.5 ± 6.8
singer	5.8 ± 2.2	**4.8 ± 2.1**	29.5 ± 23.8	172.8 ± 95.2	188.3 ± 120.8	189.9 ± 115.2	158.5 ± 68.6	10.1 ± 7.6
iceball	**8.0 ± 4.1**	107.9 ± 66.4	15.6 ± 22.1	39.8 ± 72.9	61.6 ± 85.6	97.7 ± 53.5	58.7 ± 84.0	13.5 ± 26.0



(a) coke (b) tiger1 (c) tiger2

(d) coke (e) tiger1 (f) tiger2

Fig. 5: Bounding box overlap (first row) and center location error (second row) in frames of several video sequences. Our StructBoost often achieves higher scores of box overlap and lower center location errors compared with other trackers.

and outputs, the discriminant function of the proposed StructBoost is a linear combination of weak structured learners defined over a joint space of input-output pairs.

To efficiently solve the resulting optimization problems, we have introduced a cutting-plane method, which was originally proposed for fast training of linear SVM. Our extensive experiments demonstrate that indeed the proposed algorithm is computationally tractable.

StructBoost is flexible in the sense that it can be used to optimize a wide variety of loss functions. We have exemplified the application of StructBoost by applying to multi-class classification, hierarchical multi-class classification by optimizing the tree loss, visual tracking that optimizes the Pascal overlap criterion, and learning CRF parameters for image segmentation. We show that StructBoost at least is comparable or sometimes exceeds conventional approaches

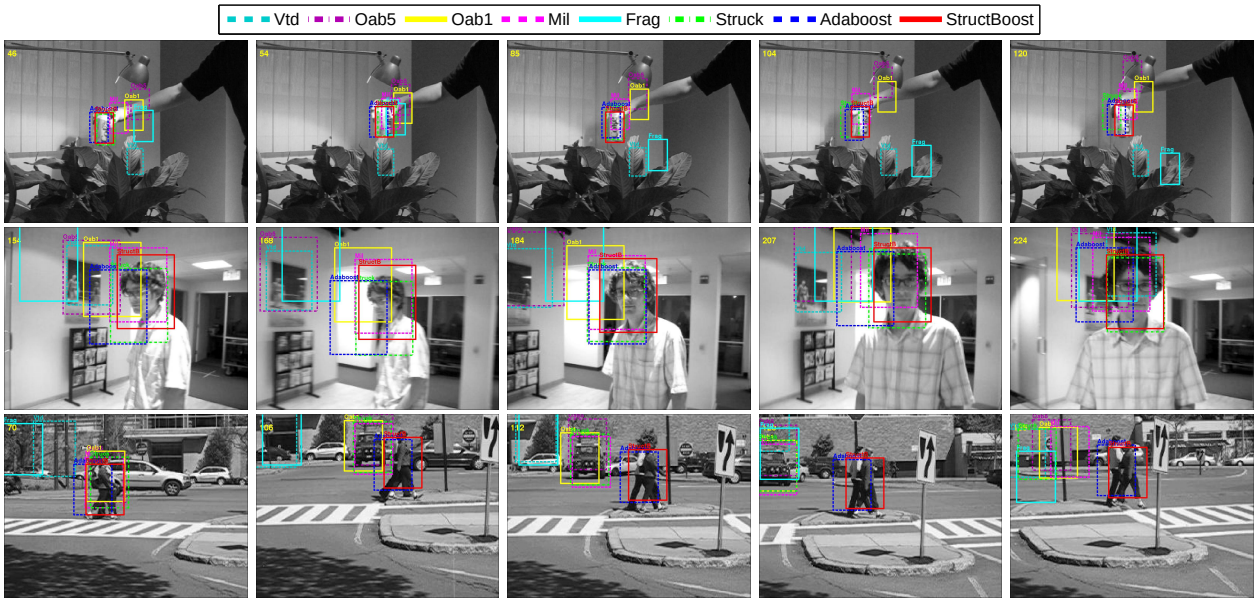
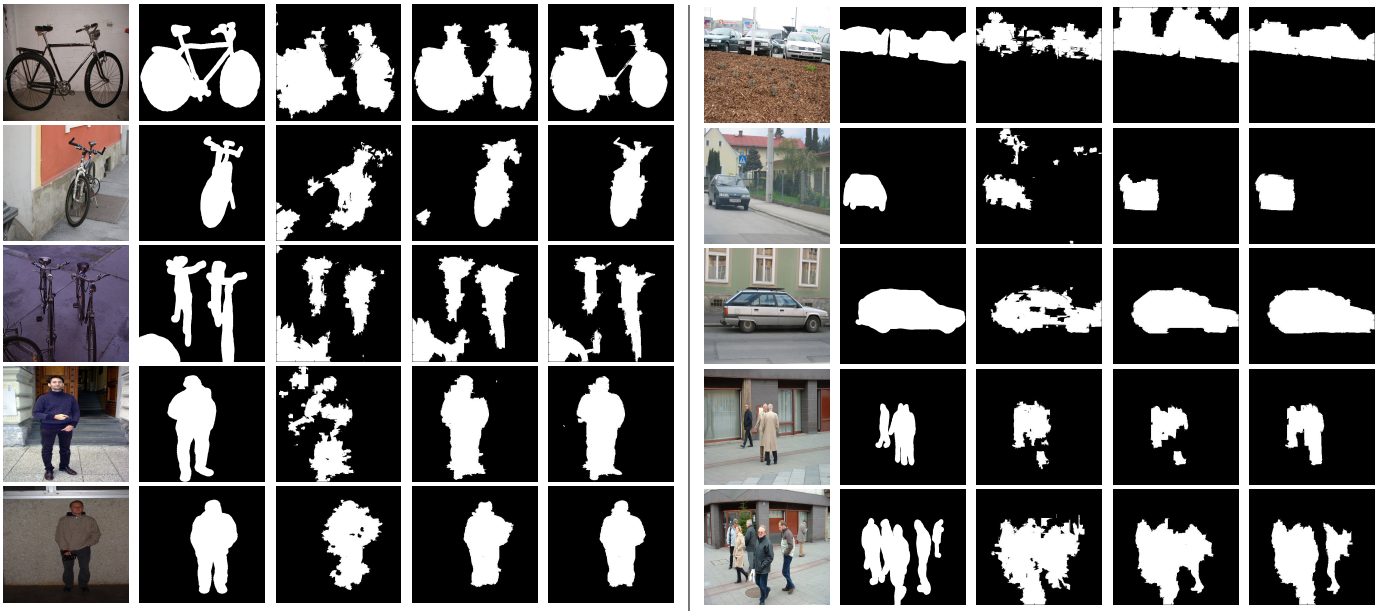


Fig. 6: Some tracking examples of several video sequences: "coke", "david", and "walk" (best viewed on screen). The output bounding boxes of our StructBoost better overlap against the ground truth than the compared methods.



(a) Testing (b) Truth (c) AdaBoost (d) SSVM (e) StructBoost (f) Testing (g) Truth (h) AdaBoost (i) SSVM (j) StructBoost

Fig. 7: Some segmentation results on the Graz-02 dataset (bicycle, car and person). Compared with AdaBoost, structured output learning methods (StructBoost and SSVM) present sharper segmentation boundaries, and better spatial regularization. Compared with SSVM, our StructBoost with non-linear parameter learning performs better, demonstrating more accurate foreground object boundaries and cleaner backgrounds.

for a wide range of applications. We also observe that StructBoost has improved performance over linear SSVM, demonstrating the usefulness of our nonlinear structured learning method. Future work will focus on more applications of this general StructBoost framework.

REFERENCES

[1] S. Hare, A. Saffari, and P. Torr, "Struck: Structured output tracking with kernels," in *Proc. IEEE Int. Conf. Comp. Vis.*, 2011.

[2] S. Nowozin and C. H. Lampert, "Structured learning and prediction in computer vision," *Foundations & Trends in Computer Graphics & Vision*, 2011.

[3] M. B. Blaschko and C. H. Lampert, "Learning to localize objects with structured output regression," in *Proc. Eur. Conf. Comp. Vis.*, 2008, pp. 2–15.

[4] I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun, "Support vector machine learning for interdependent and structured output spaces," in *Proc. Int. Conf. Mach. Learn.*, 2004, pp. 104–111.

[5] J. Weston and C. Watkins, "Multi-class support vector machines," in *Proc. Euro. Symp. Artificial Neural Networks*, 1999.

[6] K. Crammer and Y. Singer, "On the algorithmic implementation of multiclass kernel-based vector mchines," *J. Mach. Learn. Res.*, vol. 2, pp. 265–292, 2001.

[7] C. Shen and Z. Hao, "A direct formulation for totally-corrective multi-class boosting," in *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2011.

[8] C. Shen and H. Li, "On the dual formulation of boosting algo-

rithms," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 12, pp. 2216–2231, 2010.
[9] I. Steinwart, "Sparseness of support vector machines," *J. Mach. Learn. Res.*, 2003.
[10] T. Joachims, "Training linear SVMs in linear time," in *Proc. ACM SIGKDD Int. Conf. Knowledge discovery & data mining*, 2006, pp. 217–226.
[11] J. Lafferty, A. McCallum, and F. Pereira, "Conditional random fields: Probabilistic models for segmenting and labeling sequence data," in *Proc. Int. Conf. Mach. Learn.*, 2001, pp. 282–289.
[12] C. Sutton and A. McCallum, "An introduction to conditional random fields," *Foundations and Trends in Machine Learning*, 2012. [Online]. Available: http://arxiv.org/abs/1011.4088
[13] N. Plath, M. Toussaint, and S. Nakajima, "Multi-class image segmentation using conditional random fields and global classification," in *Proc. Int. Conf. Mach. Learn.*, 2009.
[14] L. Bertelli, T. Yu, D. Vu, and B. Gokturk, "Kernelized structural SVM learning for supervised object segmentation," in *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.* IEEE, 2011, pp. 2153–2160.
[15] C. Desai, D. Ramanan, and C. C. Fowlkes, "Discriminative models for multi-class object layout," *Int. J. Comp. Vis.*, vol. 95, no. 1, pp. 1–12, 2011.
[16] M. Szummer, P. Kohli, and D. Hoiem, "Learning CRFs using graph cuts," in *Proc. Eur. Conf. Comp. Vis.*, 2008, pp. 582–595.
[17] T. G. Dietterich, A. Ashenfelter, and Y. Bulatov, "Training conditional random fields via gradient tree boosting," in *Proc. Int. Conf. Mach. Learn.*, 2004. [Online]. Available: http://doi.acm.org/10.1145/1015330.1015428
[18] S. Nowozin, P. V. Gehler, and C. H. Lampert, "On parameter learning in CRF-based approaches to object class image segmentation," in *Proc. Eur. Conf. Comp. Vis.*, 2010, pp. 98–111.
[19] D. Munoz, J. A. Bagnell, N. Vandapel, and M. Hebert, "Contextual classification with functional max-margin markov networks," in *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2009, pp. 975–982.
[20] B. Taskar, C. Guestrin, and D. Koller, "Max-margin Markov networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2003.
[21] L. Mason, J. Baxter, P. L. Bartlett, and M. R. Frean, "Boosting algorithms as gradient descent," in *Proc. Adv. Neural Inf. Process. Syst.*, 1999, pp. 512–518.
[22] N. Ratliff, D. Bradley, J. A. Bagnell, and J. Chestnutt, "Boosting structured prediction for imitation learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2007.
[23] N. Ratliff, D. Silver, and J. A. Bagnell, "Learning to search: Functional gradient techniques for imitation learning," *Autonomous Robots*, vol. 27, no. 1, pp. 25–53, 2009.
[24] C. Shen, H. Li, and A. van den Hengel, "Fully corrective boosting with arbitrary loss and regularization," *Neural Networks*, vol. 48, pp. 44–58, 2013. [Online]. Available: http://hdl.handle.net/2440/78929
[25] C. Parker, A. Fern, and P. Tadepalli, "Gradient boosting for sequence alignment," in *Proc. National Conf. Artificial Intelligence*, 2006, pp. 452–457.
[26] C. Parker, "Structured gradient boosting," 2007, PhD thesis, Oregon State University. [Online]. Available: http://hdl.handle.net/1957/6490
[27] Q. Wang, D. Lin, and D. Schuurmans, "Simple training of dependency parsers via structured boosting," in *Proc. Int. Joint Conf. Artificial Intell.*, 2007, pp. 1756–1762.
[28] A. Demiriz, K. P. Bennett, and J. Shawe-Taylor, "Linear programming boosting via column generation," *Mach. Learn.*, vol. 46, no. 1-3, pp. 225–254, 2002.
[29] C. Shen, H. Li, and A. van den Hengel, "Fully corrective boosting with arbitrary loss and regularization," *Neural Networks*, 2013. [Online]. Available: http://dx.doi.org/10.1016/j.neunet.2013.07.006
[30] T. Joachims, T. Finley, and C.-N. J. Yu, "Cutting-plane training of structural svms," *Mach. Learn.*, 2009.
[31] V. Franc and S. Sonnenburg, "Optimized cutting plane algorithm for support vector machines," in *Proc. Int. Conf. Mach. Learn.*, New York, NY, USA, 2008, pp. 320–327. [Online]. Available: http://doi.acm.org/10.1145/1390156.1390197
[32] C. H. Teo, S. V. N. Vishwanthan, A. J. Smola, and Q. V. Le, "Bundle methods for regularized risk minimization," *J. Mach. Learn. Res.*, vol. 11, pp. 311–365, 2010.
[33] T. Joachims, "A support vector method for multivariate performance measures," in *Proc. Int. Conf. Machine Learning*, 2005, pp. 377–384.
[34] J. Xiao, J. Hays, K. Ehinger, A. Oliva, and A. Torralba, "SUN database: Large-scale scene recognition from abbey to zoo," in *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2010.
[35] L. Cai and T. Hofmann, "Hierarchical document categorization with support vector machines," in *Proc. ACM Int. Conf. Information & knowledge management*, 2004, pp. 78–87.
[36] C. Fellbaum, *WordNet: An Electronic Lexical Database*. Bradford Books, 1998.
[37] S. Paisitkriangkrai, C. Shen, Q. Shi, and A. van den Hengel, "RandomBoost: Simplified multi-class boosting through randomization," *IEEE Trans. Neural Networks and Learning Systems*, 2013. [Online]. Available: http://arxiv.org/abs/1302.0963
[38] B. Fulkerson, A. Vedaldi, and S. Soatto, "Class segmentation and object localization with superpixel neighborhoods," in *Proc. Int. Conf. Comp. Vis.*, 2009.
[39] S. Nowozin, C. Rother, S. Bagon, T. Sharp, B. Yao, and P. Kohli, "Decision tree fields," in *Proc. Int. Conf. Comp. Vis.*, 2011.
[40] Y. Freund, R. Iyer, R. E. Schapire, and Y. Singer, "An efficient boosting algorithm for combining preferences," *J. Mach. Learn. Res.*, vol. 4, pp. 933–969, 2003.
[41] C. Zhu, R. H. Byrd, P. Lu, and J. Nocedal, "Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound-constrained optimization," *ACM T. Math. Softw.*, 1997.
[42] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part-based models," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 9, pp. 1627–1645, September 2010.
[43] B. Babenko, M.-H. Yang, and S. Belongie, "Visual tracking with online multiple instance learning," in *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2009.
[44] A. Adam, E. Rivlin, and I. Shimshoni, "Robust fragments-based tracking using the integral histogram," in *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2006, pp. 798–805.
[45] H. Grabner, M. Grabner, and H. Bischof, "Real-time tracking via on-line boosting," in *Proc. British Mach. Vis. Conf.*, 2006, pp. 47–56.
[46] J. Kwon and K. M. Lee, "Visual tracking decomposition," in *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2010, pp. 1269–1276.
[47] S. Wang, H. Lu, F. Yang, and M.-H. Yang, "Superpixel tracking," in *Proc. Int. Conf. Comp. Vis.*, 2011, pp. 1323–1330.
[48] A. Oliva and A. Torralba, "Modeling the shape of the scene: A holistic representation of the spatial envelope," *Int. J. Comput. Vision*, 2001.
[49] M. Marszatek and C. Schmid, "Accurate object localization with shape masks," in *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2007.

6 SUPPLEMENTARY—STRUCTBOOST: BOOSTING METHODS FOR PREDICTING STRUCTURED OUTPUT VARIABLES

6.1 Dual formulation of m-slack

The formulation of StructBoost can be written as (m-slack primal):

$$\min_{\boldsymbol{w}\geq 0,\boldsymbol{\xi}\geq 0} \quad \mathbf{1}^\top \boldsymbol{w} + \tfrac{C}{m}\mathbf{1}^\top \boldsymbol{\xi} \tag{35a}$$

$$\text{s.t.:} \quad \boldsymbol{w}^\top \delta\Psi_i(\boldsymbol{y}) \geq \Delta(\boldsymbol{y}_i, \boldsymbol{y}) - \xi_i, \quad \forall i = 1,\ldots,m; \text{ and } \forall \boldsymbol{y} \in \mathcal{Y}. \tag{35b}$$

The Lagrangian of the m-slack primal problem can be written as:

$$L = \mathbf{1}^\top \boldsymbol{w} + \frac{C}{m}\mathbf{1}^\top \boldsymbol{\xi} - \sum_{i,\boldsymbol{y}} \mu_{(i,\boldsymbol{y})} \cdot \left[\boldsymbol{w}^\top \delta\Psi_i(\boldsymbol{y}) - \Delta(\boldsymbol{y}_i, \boldsymbol{y}) + \xi_i\right] - \boldsymbol{\nu}^\top \boldsymbol{w} - \boldsymbol{\beta}^\top \boldsymbol{\xi}, \tag{36}$$

where $\boldsymbol{\mu}, \boldsymbol{\nu}, \boldsymbol{\beta}$ are Lagrange multipliers: $\boldsymbol{\mu} \geq 0, \boldsymbol{\nu} \geq 0, \boldsymbol{\beta} \geq 0$. We denote by $\mu_{(i,\boldsymbol{y})}$ the Lagrange dual multiplier associated with the margin constraints (35b) for label $\boldsymbol{y}$ and training pair $(\boldsymbol{x}_i, \boldsymbol{y}_i)$. At optimum, the first derivative of the Lagrangian w.r.t. the primal variables must vanish,

$$\frac{\partial L}{\partial \xi_i} = 0 \implies \frac{C}{m} - \sum_{\boldsymbol{y}} \mu_{(i,\boldsymbol{y})} - \beta_i = 0$$

$$\implies 0 \leq \sum_{\boldsymbol{y}} \mu_{(i,\boldsymbol{y})} \leq \frac{C}{m};$$

and,

$$\frac{\partial L}{\partial \boldsymbol{w}} = 0 \implies \mathbf{1} - \sum_{i,\boldsymbol{y}} \mu_{(i,\boldsymbol{y})} \delta\Psi_i(\boldsymbol{y}) - \boldsymbol{\nu} = 0$$

$$\implies \sum_{i,\boldsymbol{y}} \mu_{(i,\boldsymbol{y})} \delta\Psi_i(\boldsymbol{y}) \leq \mathbf{1}.$$

By putting them back into the Lagrangian (36) and we can obtain the dual problem of the m-slack formulation in (35):

$$\max_{\boldsymbol{\mu}\geq 0} \quad \sum_{i,\boldsymbol{y}} \mu_{(i,\boldsymbol{y})} \Delta(\boldsymbol{y}_i, \boldsymbol{y}) \tag{37a}$$

$$\text{s.t.:} \quad \textstyle\sum_{i,\boldsymbol{y}} \mu_{(i,\boldsymbol{y})} \delta\Psi_i(\boldsymbol{y}) \leq \mathbf{1}, \tag{37b}$$

$$0 \leq \textstyle\sum_{\boldsymbol{y}} \mu_{(i,\boldsymbol{y})} \leq \frac{C}{m}, \forall i = 1,\ldots,m. \tag{37c}$$

6.2 Dual formulation of 1-slack

The 1-slack formulation of StructBoost can be written as:

$$\min_{\boldsymbol{w}\geq 0,\xi\geq 0} \quad \mathbf{1}^\top \boldsymbol{w} + C\xi \tag{38a}$$

$$\text{s.t.:} \quad \tfrac{1}{m}\boldsymbol{w}^\top \left[\sum_{i=1}^m c_i \cdot \delta\Psi_i(\boldsymbol{y})\right] \geq \frac{1}{m}\sum_{i=1}^m c_i \Delta(\boldsymbol{y}_i, \boldsymbol{y}) - \xi, \quad \forall \boldsymbol{c} \in \{0,1\}^m; \forall \boldsymbol{y} \in \mathcal{Y}, i = 1, \cdots, m. \tag{38b}$$

The Lagrangian of the 1-slack primal problem can be written as:

$$L = \mathbf{1}^\top \boldsymbol{w} + C\xi - \sum_{\boldsymbol{c},\boldsymbol{y}} \lambda_{(\boldsymbol{c},\boldsymbol{y})} \cdot \left\{\frac{1}{m}\boldsymbol{w}^\top \left[\sum_{i=1}^m c_i \cdot \delta\Psi_i(\boldsymbol{y})\right] - \frac{1}{m}\sum_{i=1}^m c_i \Delta(\boldsymbol{y}_i, \boldsymbol{y}) + \xi\right\} - \boldsymbol{\nu}^\top \boldsymbol{w} - \beta\xi, \tag{39}$$

where $\boldsymbol{\lambda}, \boldsymbol{\nu}, \beta$ are Lagrange multipliers: $\boldsymbol{\lambda} \geq 0, \boldsymbol{\nu} \geq 0, \beta \geq 0$. We denote by $\lambda_{(\boldsymbol{c},\boldsymbol{y})}$ the Lagrange multiplier associated with the inequality constraints for $\boldsymbol{c} \in \{0,1\}^m$ and label $\boldsymbol{y}$. At optimum, the first derivative of the Lagrangian w.r.t. the primal variables must be zeros,

$$\frac{\partial L}{\partial \xi} = 0 \implies C - \sum_{\boldsymbol{c},\boldsymbol{y}} \lambda_{(\boldsymbol{c},\boldsymbol{y})} - \beta = 0$$

$$\implies 0 \leq \sum_{\boldsymbol{c},\boldsymbol{y}} \lambda_{(\boldsymbol{c},\boldsymbol{y})} \leq C;$$

and,

$$\frac{\partial L}{\partial \boldsymbol{w}} = 0 \implies \mathbf{1} - \frac{1}{m}\sum_{\boldsymbol{c},\boldsymbol{y}} \lambda_{(\boldsymbol{c},\boldsymbol{y})} \cdot \left[\sum_{i=1}^{m} c_i \cdot \delta\Psi_i(\boldsymbol{y})\right] = \boldsymbol{\nu}.$$
$$\implies \frac{1}{m}\sum_{\boldsymbol{c},\boldsymbol{y}} \lambda_{(\boldsymbol{c},\boldsymbol{y})} \cdot \left[\sum_{i=1}^{m} c_i \cdot \delta\Psi_i(\boldsymbol{y})\right] \leq \mathbf{1}. \tag{40}$$

The dual problem of (38) can be written as:

$$\max_{\boldsymbol{\lambda}\geq 0} \quad \sum_{\boldsymbol{c},\boldsymbol{y}} \lambda_{(\boldsymbol{c},\boldsymbol{y})} \sum_{i=1}^{m} c_i \Delta(\boldsymbol{y}_i, \boldsymbol{y}) \tag{41a}$$
$$\text{s.t.:} \quad \frac{1}{m}\sum_{\boldsymbol{c},\boldsymbol{y}} \lambda_{(\boldsymbol{c},\boldsymbol{y})} \left[\sum_{i=1}^{m} c_i \cdot \delta\Psi_i(\boldsymbol{y})\right] \leq \mathbf{1}, \tag{41b}$$
$$0 \leq \textstyle\sum_{\boldsymbol{c},\boldsymbol{y}} \lambda_{(\boldsymbol{c},\boldsymbol{y})} \leq C. \tag{41c}$$

6.3 Convergence analysis of StructBoost

The following result shows the convergence property of Algorithm 1.

Proposition 6.1. *Algorithm 1 makes progress at each column generation iteration; i.e., the objective value decreases at each iteration.*

Proof: Let us assume that the current solution is a finite subset of weak learners and their corresponding coefficients are $\boldsymbol{w}$. When we add a weak learner that is not in the current subset and resolve the problem and the corresponding $\hat{w}$ is zero, then the objective value and the solution keep unchanged. In this case, we can draw a conclusion that the current selected weak learner and the solution $\boldsymbol{w}$ are optimal.

Now let us assume that the optimality condition is violated. We want to show that we can find a weak learner $\hat{\psi}(\cdot,\cdot)$ that is not in the current set of weak learners, such that its corresponding coefficient $\hat{w} > 0$ holds. Assume that $\hat{\psi}(\cdot,\cdot)$ is found by solving the weak learner generation subproblem and the convergence condition $\frac{1}{m}\sum_{\boldsymbol{c},\boldsymbol{y}\neq\boldsymbol{y}_i} \lambda_{(\boldsymbol{c},\boldsymbol{y})}\left[\sum_{i=1}^{m} c_i \cdot \delta\hat{\psi}_i(\boldsymbol{y})\right] \leq 1$ does not hold. In other words, we have $\frac{1}{m}\sum_{\boldsymbol{c},\boldsymbol{y}\neq\boldsymbol{y}_i} \lambda_{(\boldsymbol{c},\boldsymbol{y})}\left[\sum_{i=1}^{m} c_i \cdot \delta\hat{\psi}_i(\boldsymbol{y})\right] > 1$.

Now if this $\hat{\psi}(\cdot,\cdot)$ is added into the master problem and the primal solution is not changed; i.e., $\hat{w} = 0$, then we know that in (40), $\nu = 1 - \frac{1}{m}\sum_{\boldsymbol{c},\boldsymbol{y}\neq\boldsymbol{y}_i} \lambda_{(\boldsymbol{c},\boldsymbol{y})}\left[\sum_{i=1}^{m} c_i \cdot \delta\hat{\psi}_i(\boldsymbol{y})\right] < 0$. This contradicts the fact that the Lagrange multiplier ν must be nonnegative.

Therefore, after this weak learner is added into the master problem, its corresponding coefficient $\hat{w}$ must be a non-zero positive value. It means that one more free variable is added into the master problem and re-solving the it must reduce the objective value. That means, a strict decrease in the objective is assured. Hence Algorithm 1 makes progress at each iteration. □

At the t-th column generation iteration, the objective of the primal in (35) can be written as:

$$f(\boldsymbol{w}^{(t)}) = \sum_{j=1}^{t} w_j^{(t)} + \frac{C}{m}\sum_{i=1}^{m} \max_{\boldsymbol{y}}\left\{\Delta(\boldsymbol{y}_i, \boldsymbol{y}) - \sum_{j=1}^{t} w_j^{(t)}\left[\psi_j(\boldsymbol{x}_i, \boldsymbol{y}_i) - \psi_j(\boldsymbol{x}_i, \boldsymbol{y})\right]\right\}. \tag{42}$$

Notice that the constraints: $\boldsymbol{\xi} \geq 0$ can be removed in the optimization (35) because it is implicitly enforced by the first set of constraints.

Following the analysis of the boosting algorithm in **[?]**, we have the following proposition which describes the progress of reducing the objective at each boosting (column generation) iteration.

Proposition 6.2. *The decrease of objective value between boosting iterations $(t-1)$ and t is lower bounded as:*

$$f\left(\boldsymbol{w}^{(t-1)}\right) - f\left(\boldsymbol{w}^{(t)}\right) \geq \max_{\alpha\geq 0}\left\{-\alpha + \frac{\alpha C}{m}\sum_{i=1}^{m}\left[\psi_t\left(\boldsymbol{x}_i, \boldsymbol{y}_i\right) - \psi_t\left(\boldsymbol{x}_i, \boldsymbol{y}_i^{(t)*}(\alpha)\right)\right]\right\}, \tag{43}$$

in which,

$$\boldsymbol{y}_i^{(t)*}(\alpha) = \operatorname*{argmax}_{\boldsymbol{y}}\left\{\Delta(\boldsymbol{y}_i, \boldsymbol{y}) + \sum_{j=1}^{t-1} w_j^{(t-1)}\psi_j(\boldsymbol{x}_i, \boldsymbol{y}) + \alpha\psi_t(\boldsymbol{x}_i, \boldsymbol{y})\right\}. \tag{44}$$

Proof: We define the maximization solution in (42) as:

$$\boldsymbol{y}_i^{(t)*} = \operatorname*{argmax}_{\boldsymbol{y}}\left\{\Delta(\boldsymbol{y}_i, \boldsymbol{y}) + \sum_{j=1}^{t} w_j^{(t)}\psi_j(\boldsymbol{x}_i, \boldsymbol{y})\right\}. \tag{45}$$

The objective of the t-th interation in (42) can be written as:

$$f\left(\boldsymbol{w}^{(t)}, \boldsymbol{y}^{(t)*}\right) = \sum_{j=1}^{t} w_j^{(t)} + \frac{C}{m}\sum_{i=1}^{m}\left\{\Delta\left(\boldsymbol{y}_i, \boldsymbol{y}_i^{(t)*}\right) - \sum_{j=1}^{t} w_j^{(t)}\left[\psi_j\left(\boldsymbol{x}_i, \boldsymbol{y}_i\right) - \psi_j\left(\boldsymbol{x}_i, \boldsymbol{y}_i^{(t)*}\right)\right]\right\}. \tag{46}$$

Clearly, $\boldsymbol{y}^{(t)*}$ is a sub-optimal maximazation solution for the $(t-1)$-th iteration. The decrease of the objective value between iterations $(t-1)$ and t can be lower bounded as:

$$f\left(\boldsymbol{w}^{(t-1)}\right) - f\left(\boldsymbol{w}^{(t)}\right) = f\left(\boldsymbol{w}^{(t-1)}, \boldsymbol{y}^{(t-1)*}\right) - f\left(\boldsymbol{w}^{(t)}, \boldsymbol{y}^{(t)*}\right) \geq f\left(\boldsymbol{w}^{(t-1)}, \boldsymbol{y}^{(t)*}\right) - f\left(\boldsymbol{w}^{(t)}, \boldsymbol{y}^{(t)*}\right). \tag{47}$$

Here we construct a sub-optimal solution (denoted as $\boldsymbol{w}^{(t)'}$) for the t-th iteration, which takes the following form:

$$\boldsymbol{w}^{(t)'} = \begin{bmatrix} \boldsymbol{w}^{(t-1)} \\ \alpha \end{bmatrix}, \tag{48}$$

in which $\alpha \geq 0$. With this sub-optimial solution, the maximization solution is:

$$\boldsymbol{y}_i^{(t)*}(\alpha) = \underset{\boldsymbol{y}}{\operatorname{argmax}}\left\{\Delta(\boldsymbol{y}_i, \boldsymbol{y}) + \sum_{j=1}^{t-1} w_j^{(t-1)}\psi_j(\boldsymbol{x}_i, \boldsymbol{y}) + \alpha\psi_t(\boldsymbol{x}_i, \boldsymbol{y})\right\}. \tag{49}$$

Then the objective decrease between iterations $(t-1)$ and t can be further lower bounded as:

$$\begin{aligned} f\left(\boldsymbol{w}^{(t-1)}, \boldsymbol{y}^{(t-1)*}\right) - f\left(\boldsymbol{w}^{(t)}, \boldsymbol{y}^{(t)*}\right) &\geq f\left(\boldsymbol{w}^{(t-1)}, \boldsymbol{y}^{(t)*}\right) - f\left(\boldsymbol{w}^{(t)}, \boldsymbol{y}^{(t)*}\right) \\ &\geq f\left(\boldsymbol{w}^{(t-1)}, \boldsymbol{y}^{(t)*}(\alpha)\right) - f\left(\boldsymbol{w}^{(t)'}, \boldsymbol{y}^{(t)*}(\alpha)\right). \end{aligned}$$

Substituting it into the objective function, the left part: $f\left(\boldsymbol{w}^{(t-1)}, \boldsymbol{y}^{(t)*}(\alpha)\right)$ can be written as:

$$f\left(\boldsymbol{w}^{(t-1)}, \boldsymbol{y}^{(t)*}(\alpha)\right) = \sum_{j=1}^{t} w_j^{(t-1)} + \frac{C}{m}\sum_{i=1}^{m}\left\{\Delta\left(\boldsymbol{y}_i, \boldsymbol{y}_i^{(t)*}(\alpha)\right) - \sum_{j=1}^{t} w_j^{(t-1)}\left[\psi_j\left(\boldsymbol{x}_i, \boldsymbol{y}_i\right) - \psi_j\left(\boldsymbol{x}_i, \boldsymbol{y}_i^{(t)*}(\alpha)\right)\right]\right\}.$$

With the definition in (48), the right part: $f\left(\boldsymbol{w}^{(t)'}, \boldsymbol{y}^{(t)*}(\alpha)\right)$ is written as:

$$\begin{aligned} f\left(\boldsymbol{w}^{(t)'}, \boldsymbol{y}^{(t)*}(\alpha)\right) =& \sum_{j=1}^{t} w_j^{(t)'} + \frac{C}{m}\sum_{i=1}^{m}\left\{\Delta\left(\boldsymbol{y}_i, \boldsymbol{y}_i^{(t)*}(\alpha)\right) - \sum_{j=1}^{t} w_j^{(t)'}\left[\psi_j\left(\boldsymbol{x}_i, \boldsymbol{y}_i\right) - \psi_j\left(\boldsymbol{x}_i, \boldsymbol{y}_i^{(t)*}(\alpha)\right)\right]\right\} \\ =& \sum_{j=1}^{t-1} w_j^{(t-1)} + \alpha + \frac{C}{m}\sum_{i=1}^{m}\left\{\Delta\left(\boldsymbol{y}_i, \boldsymbol{y}_i^{(t)*}(\alpha)\right)\right. \\ & \left. - \sum_{j=1}^{t-1} w_j^{(t-1)}\left[\psi_j\left(\boldsymbol{x}_i, \boldsymbol{y}_i\right) - \psi_j\left(\boldsymbol{x}_i, \boldsymbol{y}_i^{(t)*}(\alpha)\right)\right] - \alpha\left[\psi_t\left(\boldsymbol{x}_i, \boldsymbol{y}_i\right) - \psi_t\left(\boldsymbol{x}_i, \boldsymbol{y}_i^{(t)*}(\alpha)\right)\right]\right\} \\ =& f\left(w^{(t-1)}, \boldsymbol{y}^{(t)*}(\alpha)\right) + \alpha - \frac{\alpha C}{m}\sum_{i=1}^{m}\left\{\left[\psi_t\left(\boldsymbol{x}_i, \boldsymbol{y}_i\right) - \psi_t\left(\boldsymbol{x}_i, \boldsymbol{y}_i^{(t)*}(\alpha)\right)\right]\right\}. \end{aligned}$$

From the above, the lower bound in (50) can be wrriten as:

$$f\left(w^{(t-1)}, \boldsymbol{y}^{(t)*}(\alpha)\right) - f\left(\boldsymbol{w}^{(t)'}, \boldsymbol{y}^{(t)*}(\alpha)\right) \geq -\alpha + \frac{\alpha C}{m}\sum_{i=1}^{m}\left[\psi_t\left(\boldsymbol{x}_i, \boldsymbol{y}_i\right) - \psi_t\left(\boldsymbol{x}_i, \boldsymbol{y}_i^{(t)*}(\alpha)\right)\right]. \tag{50}$$

Finally, the objective decrease is lower bounded as:

$$f\left(\boldsymbol{w}^{(t-1)}\right) - f\left(\boldsymbol{w}^{(t)}\right) \geq \max_{\alpha \geq 0}\left\{-\alpha + \frac{\alpha C}{m}\sum_{i=1}^{m}\left[\psi_t\left(\boldsymbol{x}_i, \boldsymbol{y}_i\right) - \psi_t\left(\boldsymbol{x}_i, \boldsymbol{y}_i^{(t)*}(\alpha)\right)\right]\right\}, \tag{51}$$

in which,

$$\boldsymbol{y}_i^{(t)*}(\alpha) = \underset{\boldsymbol{y}}{\operatorname{argmax}}\left\{\Delta(\boldsymbol{y}_i, \boldsymbol{y}) + \sum_{j=1}^{t-1} w_j^{(t-1)}\psi_j(\boldsymbol{x}_i, \boldsymbol{y}) + \alpha\psi_t(\boldsymbol{x}_i, \boldsymbol{y})\right\}. \tag{52}$$

This concludes the proof. □