APPROXIMATE CONSTRAINT GENERATION FOR EFFICIENT STRUCTURED BOOSTING

Guosheng Lin, Chunhua Shen, Anton van den Hengel

School of Computer Science, The University of Adelaide, Australia

ABSTRACT

We propose efficient training methods (SBoost) for totally-corrective boosting based structured learning. The optimization of boosting method for structured learning is more challenging than the structured support vector machine. Basically, we propose smooth and convex formulation for boosting based structured learning, and develop approximate constraint generation together with column generation to solve the optimization with large number of constraints and variables. Because of the convexity and smoothness, the optimization in each generation iteration can be solved efficiently. We demonstrate some structured learning applications in computer vision using SBoost, including invariance learning for digit recognition, object detection and hierarchical image classification.

1. INTRODUCTION

Structured learning has attracted extensive attention recently in machine learning and computer vision (e.g. image segmentation [1] and object detection [2]). Structured learning is to learn a prediction function that is able to return structured outputs which can be vectors, sequences, tree, etc. It is an extension of the traditional binary classification and regression which is for one dimensional output. For example, in object detection, the output of the detector can be naturally formulated as box coordinates. Structured support vector machines (SSVM) [3] generalize the multi-class SVM of [4] and [5] to a much broader problem of predicting interdependent and structured outputs. We propose efficient training methods for boosting based structured learning (SBoost), which is to learn a discriminant function that measures the compatibility of the input and output pair \((x, y)\). Analogous to SSVM, the general form of \(F\) is \(F(x, y; w) : \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}\) over input-output pairs and the prediction is achieved by maximizing \(F(x, y; w)\) over all possible \(y \in \mathcal{Y}\).

Boosting algorithms linearly combine a set of moderately accurate weak learners to form a highly accurate strong predictor. Boosting method for structured learning is to linearly combine a set of weak structured learner to generate a strong structured output predictor. A boosting method for structured learning can be developed using the general total corrective boosting framework of [6] with column generation technique. Recently, Shen and Hao [7] proposed a total corrective multi-class boosting methods using the loss functions of multi-class SVM [4, 5] and column generation technique, which can be regarded as a special case of boosting based structured learning algorithm.

The boosting method usually solves a more complex optimization problem than SVM. Given that the number of possible weak learners can be infinitely large, the number of weighting variables can be infinitely large. Moreover, structured learning usually involves a large (or infinite) number of constraints. We develop algorithms using column generation and approximate constraint generation to overcome these difficulties (Sec. 2). One issue arises that the optimization need to be solved in every generation iteration. Chapelle [8] proposed a smooth formulation of SVM which can be solved efficiently. Inspired by this work, we proposed smooth formulations of our boosting method to achieve fast optimization.

1.1. Main contributions

The main contributions of this paper are two-fold. First, we propose efficient training methods for boosting based structured learning with a large number of constraints and variables by using smooth formulation, column generation and approximate constraint generation. Second, we demonstrate several structured learning applications using SBoost, including invariance learning, object detection, and hierarchical image classification, which show the flexibility and usefulness of SBoost.

1.2. Related work

The two popular structured learning methods are conditional random field (CRF) [9] and SSVM [3]. Our SBoost is a boosting method for structured output prediction which builds upon the work of column generation boosting [6]. Recently many computer vision applications using structured learning have been proposed. The work of [2] formulated the object detector output as structured output problem and directly optimized the Pascal area overlap criterion. The works of [10, 11] proposed structured learning method for invariance learning and object detection.

2. STRUCTURED BOOSTING

To implement structured learning using boosting, we can formulate the following problem with a general convex risk function:

\[
\min_{w \geq 0, \gamma} \| w \|_1 + C \sum_{i} g(\gamma_i) \\
\text{s.t.} \quad w^T \left[ I(y_i | x_i) - I(x_i, y) \right] = \gamma_i \\
\quad \forall i = 1, \ldots, m; \quad \forall y \in \mathcal{Y} \setminus \mathcal{Y}_i.
\]

(1b)

We introduce the risk function \(g(\cdot)\) here to denote the general convex function on the margin variables \(\gamma_i\). \(m\) is the number of input example \(x_i\). The output variable \(y \in \mathcal{Y}\) are structured variables, e.g., vectors, strings, graphs. \(\mathcal{F}\) denotes a set of weak structured learners; the size of \(\mathcal{F}\) can be infinite. \(h\) denote a weak structured learner; \(h_j(x, y) \in \mathcal{F}, j = 1 \ldots n\). \(h(\cdot, \cdot)\) is a function that maps an input-output \((x, y)\) pair to \([-1, +1]\). Although our discussion works for the general case that \(h(\cdot, \cdot)\) can be any function with a real value output, we consider binary weak learners here. We define column vectors \(\mathbf{h}(x, y) = [h_1(x, y), h_2(x, y), \ldots]^T\) to be the outputs of all weak learners on the training datum \(x\) and label \(y\).

The discriminant function that we want to learn is then \(F:\mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}\) over input-output pairs: \(F(x, y; w) = w^T \mathbf{h}(x, y) = w^T h_1(x, y) + \cdots + w^T h_n(x, y)\).
\[ \sum_{i} w_{i} h_{i}(x, y) \text{ with } w \geq 0. \] Analogous to SSVM, the output prediction \( y^* \) for unknown example \( x \) is to maximize the joint compatibility function over the output \( y: y^* = \arg \max_{y} F(x, y; w) = \arg \max_{y} w^{\top} \ln(x, y). \) There are two major obstacles for solving the optimization of SBoost(1). First, as in conventional boosting, because the possibility of weak learners \( h(\cdot, \cdot) \) can be infinitely large, the dimension of \( w \) can be infinitely large. Second, same as in SSVM, the number of constraints (1b) can be infinitely large. So we are not able to directly solve for \( w. \) In our method, we use column generation together with constraint generation to overcome these difficulties. First we discuss a conventional definition of \( g \) following the large margin framework:

\[ g^{NS}(\gamma) = \frac{1}{n} \sum_{i} \max \{ \Delta(y, y) \max \{ 1 - \gamma i(i, y), 0 \} \}. \] (2)

Here \( \Delta(y, y) \) calculates the loss associated with a prediction \( y \) against the true label value \( y_i. \) With the definition in (2), the optimization (1) can be written as (3) which is similar to SSVM with slack rescaled. To simplify the notation, we introduce \( \delta_{H}(y) = \ln(x, y_i) - \ln(x, y) \) and slack variable \( \xi_i = \max_{y} \{ \Delta(y, y)(1 - \gamma i(i, y)) \}. \)

\[
\begin{align*}
\min_{w \geq 0, \xi \geq 0} & \|w\|_1 + \frac{c}{n} \sum_{i} \xi_i \\
\text{s.t.} & \quad w^{\top} \delta_{H}(y) \geq 1 - \frac{1}{n} \ln_{i}(y_i), \quad \forall i = 1, \ldots, m; \text{ and } \forall y \in \mathbb{Y}\setminus y_i. \quad (3a)
\end{align*}
\]

The optimization in (3) is a linear programming (LP) problem. However solving this LP problem in each generation iteration is usually slow. The definition of \( g^{NS} \) in (2) is not smooth because of the maximization over \( y \) and the hinge loss: \( \max \{ 1 - \gamma(i, y), 0 \} \). To make the optimization efficient, we replace the non-smooth function \( g^{NS} \) in (2) by smooth and convex function \( g^{S} \) as shown in (4), so that we can use much more efficient optimization method such as L-BFGS [12].

\[ g^{S}(\gamma) = \frac{1}{p} \sum_{i} \Delta(y, y) \max \{ 1 - \gamma i(i, y), 0 \}^2. \] (4)

Here \( p \) is the number of slack variable \( \xi(i, y). \) With the definition of \( g^{S} \) and slack variable \( \xi(i, y) = 1 - \gamma i(i, y), \) the optimization in (1) can be written as (5), denoted as SBoost.

\[
\begin{align*}
\min_{w \geq 0, \xi \geq 0} & \|w\|_1 + \frac{c}{p} \sum_{i} \Delta(y, y) \max \{ \xi(i, y), 0 \}^2 \\
\text{s.t.} & \quad 1 - \xi_{i} \delta_{H}(y) = \xi(i, y), \quad \forall i = 1, \ldots, m; \text{ and } \forall y \in \mathbb{Y}\setminus y_i. \quad (5a)
\end{align*}
\]

\[
\begin{align*}
\text{s.t.} & \quad \sum_{i} \frac{i \cdot y \cdot y_i}{\Delta(y, y)} \delta_{H}(y) \lesssim 1 \quad (6b)
\end{align*}
\]

\[ \lambda(i, y) \) is the Lagrange dual multiplier associated with the marginal constraints (5b). With the primal-dual pair of (5) and (6), our column generation is described in Algorithm 1.

### Algorithm 1: Column generation for SBoost

1. **Input:** training examples \((x_1, y_1), (x_2, y_2), \ldots, p; \) parameter \( C; \) termination threshold and the maximum iteration number.
2. **Initialize:** weak learner set \( T = \emptyset; \) for each \( i, (i = 1, \ldots, m), \) randomly choose \( y^{(0)}(i) \in \mathbb{Y}\setminus y_i, \) initialize \( \lambda_i(y^{(0)}(i)) = 1 \).
3. **Repeat**
4. - Find a new weak learner \( h^{(i)}(\cdot, \cdot) \) and add to weak learner set \( T \) by solving: \( h^{(i)}(\cdot, \cdot) = \arg \max_{h(\cdot, \cdot)} \sum_{i} \xi(i, y) \lambda(i, y)(h(x, y) - h(x_i, y)). \)
5. - Solve (5) on weak learner set: \( h \in T \) using L-BFGS [12] to obtain \( w. \)
6. - Update dual variables \( \lambda \) by the primal solution \( w \) using KKT conditions: \( \lambda_i(y) = \frac{1}{p} \| \Delta(y, y) \max \{ 1 - w^{\top} \delta_{H}(y), 0 \} \)
7. **Until** converge or the maximum iteration is reached.
8. **Output:** the discriminant function \( F(x, y; w) = w^{\top} \ln(x, y). \)

### Table 1: Multi-class classification

We compare our method SBoost against MultiBoost with hinge loss [7]. The speed-up is significant. \( "-" \) means no results obtained after running 2 hours. \( "\text{>}" \) means that the method is not converged after running 2 hours.

<table>
<thead>
<tr>
<th>dataset (instances, classes)</th>
<th>method</th>
<th>CPU time (s)</th>
<th>training error %</th>
<th>test error %</th>
</tr>
</thead>
<tbody>
<tr>
<td>wine(178, 3)</td>
<td>SBoost</td>
<td>6.1±1</td>
<td>0.0±10.0</td>
<td>2.2±10.0</td>
</tr>
<tr>
<td>vowel(1452, 11)</td>
<td>SBoost</td>
<td>2.0±1</td>
<td>0.0±10.0</td>
<td>1.3±11.2</td>
</tr>
<tr>
<td>segment(2310, 7)</td>
<td>SBoost</td>
<td>2.0±1</td>
<td>0.0±10.0</td>
<td>1.3±11.2</td>
</tr>
<tr>
<td>usps(8298, 10)</td>
<td>SBoost</td>
<td>2.0±1</td>
<td>0.0±10.0</td>
<td>1.3±11.2</td>
</tr>
<tr>
<td>pendigits(10992, 10)</td>
<td>SBoost</td>
<td>2.0±1</td>
<td>0.0±10.0</td>
<td>1.3±11.2</td>
</tr>
</tbody>
</table>

### 2.2. Constraint generation

We develop an approximate constraint generation algorithm to overcome the difficulty of infinitely many constraints. The approximate constraint generation and column generation algorithm is shown in Algorithm 2. The inference algorithm for finding most violated constraints depends on applications, e.g. for multi-class classification, we can simply perform exhaustive search by enumerating all possible value of \( y \) for image segmentation, graph cut or message passing algorithms can be applied [1]. There are two points in the algorithm that need attention. First, we put the constraint generation before the column generation as the most outer loop to reduce inference calls significantly. Second, reusing the weak learner set \( T \). The weak learner set \( T^{(i)} \) in the constraint generation iteration \( t \) is initialized by \( T^{(i-1)} \) in the last iteration. To prevent the number of weak learners growing too large, we only reuse those weak learners with top-\( K \) largest weighting values in last iteration.

### 3. EXPERIMENTS

We carry out experiments on a few applications including multi-class classification, handwritten digit recognition, car detection in aerial images and hierarchical image classification.
Fig. 4: Some aerial car detection results of our method SBoost, which optimizes the Pascal box overlap criteria. The color of bounding boxes shows the confidence of prediction. Yellow color indicates highest confidence, while black color indicates lowest confidence.

Fig. 1: Random transformation examples of digit 3 and 8. The first column is the original digit image, the rest are some examples transformed by randomly scaling, rotating, translating and shearing.

Fig. 2: Invariance Learning on MNIST. The left figure is the error rate against the sample number per digit. The right figure is the number of valid examples selected by the constraint generation of SBoost. Our SBoost performs the best, and SBoost is able to select a relative small subset of examples.

Fig. 3: The precision and recall curve of aerial car detection [14]. Our method SBoost performs the best.

3.1. Multi-class classification

To evaluate the training time of our method SBoost, we run experiments on some UCI multi-class datasets which is shown in Table 1. We use decision stumps as weak learners in all of the experiments. In this experiment, we use all constraints for our method. We compare with a closely related method: MultiBoost [7] (using the hinge loss). We randomly sample 75% data for training and the rest for testing. All results are averaged by 5 runs. The maximum iteration is set to 500. The maximum training time is set to 2 hours. We can see that our method is significantly faster.

3.2. Invariance learning for digit recognition

Invariance is usually a desired property of many computer vision applications. For instance, in digit image recognition, if a digit image is slightly transformed, such as small range of rotation, scaling and other affine transformation, it still can be recognized. Recently some SVM based structured learning methods [11, 10] are proposed for invariance learning, which basically make use of the most violated constraint inference to discover useful transformed examples. To apply our method, we define the output with transformation: \( y = (t, y') \). \( t \) denotes the transformation pair term applied to the example \( x \) and its output \( y' \). \( t = (t_x, t_y) \in \mathcal{T} \). The output loss \( \Delta \) and weak learners \( h \) now involve the transformations \( t \) in the optimization step. It is not possible to consider all transformations at the same time. We use constraint generation to select useful constraints. In this experiment we use MNIST dataset. We choose two digits as a simple example: 3 and 8 which are easy to be confused. Transformations are important when the training data is limited. We randomly sample a few images of these two digits to generate a small training set (range from 10 to 50), and test on the whole test set. Transformations of the training image include 1 pixel to 3 pixels translation on 8 directions, scaling up to ±2.0, shearing on x-axis and y-axis up to ±0.15 unit, rotation up to ±15 degrees and random combination of the above transformations. No transformation is defined on \( y \). We
Algorithm 2 Approximate constraint generation and column generation

1: Input: training samples \((x_i; y_i), \ldots\); parameter \(C; K, Q\).
2: Initialize: \(t = 0\); initialize constraint set \(W\); weak learner set \(J(t) = \emptyset\).
3: Repeat (Constraint generation loop)
   4: \ - Update \(t = t + 1\); Initialize \(\lambda_t\) by \(\lambda(t-1)\)
   5: \ - Initialize \(J(t)\) by \(J(t-1)\) with top-K largest weights in \(w\).
   6: \ - Repeat (Column generation loop)
   7: \ - Find weak learner \(h^*(\cdot)\) and add to \(J(t)\) by solving:
   \[ h^*(\cdot) = \arg\max_{h(\cdot)} \sum_{(i,y) \in W} \lambda_{i,y} \left[ h(x_i, y) - (h(x_i, y) \right] \]
   8: \ - Solve the optimization (5) on the constraint set: \((i, y) \in W\) and weak learner set: \(h \in J(t)\) using L-BFGS to obtain \(w_{t,\lambda}\).
   9: \ - Update \(\lambda\) by \(w\) (using KKT): \((i,y) \in W, h \in J(t)\) :
   \[ \lambda_{i,y} = \frac{2C}{T} \Delta(y, y^\prime) \max\{1 - w_{t,\lambda}^\top \delta h_i(y, y^\prime), 0\} \]
10: \ - Until max column generation iteration is reached or converge.
11: \ - While the number of newly added constraints \(\leq Q\).
12: \ - Randomly pick \(i\) from \(1, \ldots, m\) without replacement;
13: \ - Find a constraint: \((i, y_i^\prime)\) by solving (inference step):
   \[ y_i^\prime = \arg\max_y \left\{ [1 - w_{t,\lambda}^\top \delta h_i(y)] \sqrt{\Delta(y_i, y_i^\prime)} \right\} \]
14: \ - If \( [1 - w_{t,\lambda}^\top \delta h_i(y_i^\prime)] \sqrt{\Delta(y_i, y_i^\prime)} > 0 \) then add \((i, y_i^\prime)\) into \(W\) if not exist.
15: \ - End While
16: Until max constraint generation iteration is reached or converge.
17: Output: the discriminant function \(F(x, y; w) = w_{t,\lambda}^\top h(x, y)\).

can see that the number of all possible transformations is very large.

Fig. 1 shows some transformation examples. We compare with LP
boost [13] and Adaboost. The results are shown in Fig. 2. All results
are averaged by 5 runs. With transformations our method performs
the best and is able to select a small set of useful constraints.

3.3. Car detection in aerial images

Some recently proposed methods apply structured learning to object
detection, which are based on bounding box ranking and optimize
the box overlap criteria [2, 16]. The output loss function can be
defined as:
\[ \Delta(y, y') = 1 - \frac{area(y \cap y')}{area(y \cup y')} \]
with \(y, y'\) being the bounding box coordinates. \(y \cap y'\) and \(y \cup y'\)
are the box intersection and union. Here \(y_i\) is the labelled ground truth box, and \(y\) is any prediction
box in the image \(x_i\). Weak learners \(h_i(x, y)\) represent weak
classifiers trained with the image features (HOG) extracted from
the image patch defined by \(y\). We use the aerial car detection dataset
proposed in [14]. The dataset consists of 30 aerial images and more
than 1000 cars are annotated manually. The bounding box size is set
to 45 \times 45 which is approximately the average size of labelled boxes.
Standard sliding window setting is used here. The cell size of HOG
feature is set to 5 \times 5. For comparison, we also train detectors using
linear SVM and Adaboost. We divide the dataset into 5 folds, with
each fold containing 6 images. In each run, 4 folds are for training,
and the rest 1 for testing. We run 5 times and report the averaged
Pascal precision-recall curves in Fig. 3. The result shows SBoost
performs the best. Some detection examples of SBoost are shown in
Fig. 4.

3.4. Hierarchical image classification

Classes of images are usually organized in taxonomies or hierar-
chy. For example, the ImageNet dataset has organized all the
classes according to the tree structures of WordNet. Hierarchical
image classification is to predict hierarchical labels of an image
that match the taxonomy. The prediction performance is evaluated
by tree loss, which penalizes the false prediction according to the
taxonomy. Generally, considering the class taxonomy also helps to
improve the multi-class classification accuracy that evaluated by
the conventional 1/0 loss. Our SBoost for hierarchical image classi-
fication follows the way in [3]. The output \(y\) of an input image \(x_i\)
is hierarchical labels which can be defined as an \(L\) dimensional vector
\(y^{(1)}, \ldots, y^{(L)}\), where \(y^{(i)}\) is the class label in the \(i\)-th level of
the taxonomy \(T\). We use the tree loss \(\Delta_{tree}(y, y')\) as the output loss
function \(\Delta\). The tree loss \(\Delta_{tree}(y, y')\) is the height of the first
common node of \(y\) and \(y'\) in the tree taxonomy from the bottom
to the top. In this experiment, we construct a dataset that contains
6 classes of scene images with 3 category levels. The taxonomy
is shown in Fig. 5. For each scene class, we use the top first 200
images from the SUN dataset [15]. In each run, we randomly select
75\% examples for training, and the rest for testing. We report the
averaged results of 5 runs in Table 2. We also run the multi-class
SBoost (SBoost-multi), the MultiBoost [7] with exponential loss,
and two conventional multi-class boosting methods: AdaBoost.ECC
[17] and AdaBoost.MH [18]. The max iteration is set to 1000. We
use the HOG features described in [15]. SBoost-tree performs the
best in terms of the classification accuracy and the tree loss.

4. CONCLUSION

We have proposed an efficient training method for boosting based
structured learning, which typically has a large number of constraints
and variables. Our experiments show the flexibility and usefulness
of the proposed SBoost method. Future work will focus on more
applications in computer vision and image processing.

REFERENCES