Big and Tall: Large Margin Learning with High Order Losses

Daniel Tarlow
University of Toronto
dtarlow@cs.toronto.edu

Richard Zemel
University of Toronto
zemel@cs.toronto.edu

Abstract

Graphical models are ubiquitous in computer vision, where they are typically used to represent an energy function that factorizes according to the graphical model structure. An exciting and active direction of research is to look for richer types of interaction, where energy functions can be made more realistic, but where inference can still be made to remain tractable.

In this work, we take the insights that are usually applied to modeling and inference*, and we apply them to* loss functions and learning*. Current standard approaches to learning in structured output models directly incorporate the loss function in the learning. Ideally this loss function is tailored to the task and accurately represents the aims for the model's output. Yet learning in these models typically does not optimize performance on the true task loss, due to computational complexity, instead resorting to surrogate simple decomposable loss functions. Here, we explicate a large class of non-decomposable (high order) loss functions and show that they can be readily and efficiently incorporated in a large-margin learning formulation.*

We demonstrate these loss functions in several settings. First, we train an image labeling model to optimize an evaluation metric of interest—the intersection-over-union score used to evaluate PASCAL VOC Challenge entries. Second, we propose two loss functions that are appropriate to use when only weak label information is available at training time, but when the model is evaluated based on full label information at test time. Experimental results show the high order loss functions to improve performance over baselines in all of these settings.

1. Introduction

Many computer vision problems, such as object segmentation, disparity estimation, boundary localization, and 3-D reconstruction, can be formulated as pixel or voxel labeling tasks. A challenge when formulating a learning objective (a *loss function*) is how to balance expressiveness with computational efficiency: we aim for application-specific loss

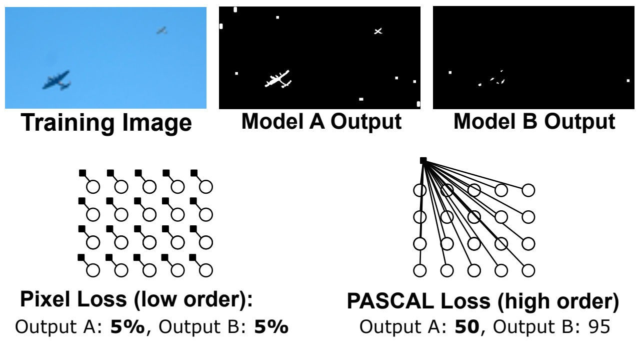

Figure 1. Which model produces a better segmentation? The standard Pixel Loss function does not distinguish between Outputs A and B, because the number of pixel errors is equal (5%). Here we describe methods that learn to optimize high order losses, such as (1 – the PASCAL intersection-over-union score), which prefers A.

functions that capture our desires for the model but that can be optimized efficiently. These issues are especially relevant in a structured output setting, where both the structure and loss are expressed over large sets of output variables.

Consider the problem of evaluating a simple foreground-background image labeling output (see Fig. 1). A common loss function to optimize is the number of per-pixel errors made relative to the ground truth. However, while convenient, this is often not an accurate representation of our desires for the model. In labeling problems where foreground objects are small, the labeling that assigns all pixels to the background class obtains a disproportionately low loss relative to what we likely believe to be a good evaluation metric. For this reason, more complex evaluation measures, have been introduced. In the PASCAL VOC Segmentation Challenge [2], for example, accuracy is judged as $\frac{\text{\# true positives}}{\text{\# true positives + \# false positives + \# false negatives}}$. In the domain of machine translation, the BLEU score, which assesses the precision of n-grams in candidate machine translations against reference human translations, involves interactions over large subsets of outputs [5]. Similarly, the F_β score used in natural language processing, the area under the ROC curve used to evaluate binary classifiers, and many other common evaluation measures involve such *high order* interactions. In vision, most important structural properties do not factorize according to individual variables. If we would like to evaluate whether outputs are e.g., connected, convex, or of a particular size, it will require a performance measure that takes into account joint configurations of large subsets of variables.

For discriminative learning, the model optimization should take the target performance measure into account. In a *margin-scaled* structural SVM, learning aims to maximize the margin between the ground truth and the most violating constraint, which is defined as the output that maximizes a sum of likelihood and loss. When the loss is decomposable, finding this most violating constraint is easy, as the loss can be folded directly into standard inference computations. However, for *high order losses (HOLs)*, standard methods for solving the relaxed problem are still intractable. In the standard cutting plane formulation, finding the most violating constraint is in general an intractable high-order inference problem; in the subgradient formulation, computing a subgradient requires solving the same optimization problem. Interestingly, there are certain classes of high order interaction where efficient optimization can be performed either exactly or approximately. Such cases have been the focus of much recent work on *maximum a posteriori (MAP)* inference in high order models (see e.g., [6]), but little work has applied the computational insights at the heart of this work to learning problems; that is our goal here.

Contributions: In this work, we show that the same routines and classes of interaction being explored in an inference setting can be utilized to define loss functions in a large margin learning setting. We extend the basic insights of these inference procedures to develop novel and useful loss functions, and show their utility in a variety of image labeling scenarios. A secondary emphasis of our work concerns its modularity and generality. The complex loss functions are expressed as factors that can be used generically in a factor graph formulation. These factors can be combined to form a range of loss functions and also be combined with any factor graph model, with no need for new formulations, derivations, or code.

2. Structural SVMs

Given an input image x with N pixels at test time, our goal is to produce an output image labeling $y \in \{0, \ldots, K-1\}^N$. We will assume $K = 2$ throughout, but the loss formulations apply equally well to problems with larger K. A structural SVM works by mapping x to y via maximizing an input-dependent (log) likelihood function: $y = \arg\max_{\hat{y}} L(\hat{y}; x) = \arg\max_{\hat{y}} w^T \phi(\hat{y}; x)$. The learning problem is to find a w that maximizes a margin between the ground truth $y^*_{(j)}$ and all other outputs $y_{(j)}$ for all training examples j.

The problem can be written as a quadratic program (QP):

$$\textbf{min.}_{w,\xi} \;\; w^T w + C \sum_n \xi_{(j)}$$
$$\textbf{s.t.} \;\; w^T \Big[\phi(y^*_{(j)}, x_{(j)}) - \phi(y_{(j)}, x_{(j)}) \Big] \geq 1 - \xi_{(j)} \qquad (1)$$

where the constraint is replicated for each example j and for all $y_{(j)} \neq y^*_{(j)}$. C is a regularization parameter. To incorporate a loss function Δ, the *margin-scaling* approach enforces a loss-dependent variable margin for different labelings, replacing (1) with:

$$\textbf{s.t.} \;\; w^T \Big[\phi(y^*_{(j)}, x_{(j)}) - \phi(y_{(j)}, x_{(j)}) \Big] \geq \Delta(y_{(j)}) - \xi_{(j)} \qquad (2)$$

Even for a single example, (2) represents an exponentially large number of constraints, so explicitly instantiating the QP is intractable. A common strategy is to start with an empty set of constraints, successively add the most violated constraint, re-solve the QP, and repeat. To find violated constraints, a *loss-augmented MAP* problem must be solved:

$$y^- = \max_{\hat{y}} \left[L(\hat{y}; x) + \Delta(\hat{y}) \right]. \qquad (3)$$

This procedure converges in polynomial time [8].

One Slack Formulation [3]: A more efficient formulation turns out to be equivalent. As usual, at iteration t, on example j, find a negative example $y^-_{(j)}$. Add one constraint in the form of (2) for a single $y_{(j)}$ to a constraint set $\mathcal{S}_t$. After finding most violated constraints for all instances, add the averaged constraints

$$w^T \sum_{(y^+_{(j)}, y^-_{(j)}) \in \mathcal{S}_t} \Big[\phi(y^+_{(j)}, x_{(j)}) - \phi(y^-_{(j)}, x_{(j)}) \Big] \geq \sum_{(y^+_{(j)}, y^-_{(j)}) \in \mathcal{S}_t} \Delta(y^-_{(j)}) - \xi \qquad (4)$$

to the quadratic program for each t. In this formulation, there is a single ξ, and the size of the QP grows with the number of iterations rather than number of iterations times number of training examples. Especially with large data sets, this is a significantly more efficient, yet surprisingly equivalent, formulation.

3. Learning with High Order Losses

At a high level, most structured output learning updates rely on an intelligent choice of *positive examples* y^+ and *negative examples* y^-. In the standard margin-scaling formulation, the positive example is chosen to be the ground truth y^*, and the negative example is chosen to be the loss-augmented MAP solution. In this section, we begin with a minor generalization of the standard structural SVM formulation (in Section 3.1). Then in Section 3.2, we describe the computational details of our main contributions, examples demonstrating how high order loss functions can be efficiently incorporated into large margin learning.

3.1. Finding Positive and Negative Examples

Positive Examples: Partially labeled training data do not provide full ground truths. Instead, there is a set of pixels with known labels (e.g., with bounding boxes, pixels outside the bounding box are background) and a set of pixels with unknown labels (e.g., pixels inside the bounding box may be either foreground or background). A loss function defined in terms of partial labels implicitly expresses properties that we desire in a positive example, but we need a single labeling y^+ to compute $\phi(y^+, x)$. To deal with this issue, we run inference to find the labeling that has zero loss and highest likelihood under the model. We call this the *0-Loss Constrained MAP* problem: $y^+ = \max_{\hat{y}:\Delta(\hat{y})=0} L(\hat{y}; x)$. Note that when full labelings are given as training data (such as in our first set of experiments below), there is typically only one labeling with zero loss, so this reduces to using the ground truth as the positive example.

Negative Examples: The loss-augmented MAP problem (3) is equivalent to a MAP inference problem in a factor graph for the model with an additional factor for the loss function. For a factor representing loss function Δ, the needed MPBP messages are,

$$m_{\Delta\to i}(y_i) = \max_{y_{-i}} \left[\Delta(y_i, y_{-i}) + \sum_{i' \neq i} m_{i'\to\Delta}(y_{i'}) \right], \quad (5)$$

where y_{-i} is the set of all variables excluding the ith. Note there is a separate message for each variable and each value it can take on, so naively there are $2N$ optimizations to perform. A theme in developing efficient message passing algorithms is to share the work between these different optimizations. In the following section, we show how to efficiently compute these messages for factors representing several loss functions of interest.

3.2. Factor Computations

PASCAL Challenge Loss: Let y^* be the ground truth labeling. The loss used to evaluate the PASCAL VOC Segmentation Challenge can be written as

$$\Delta^{\text{PASCAL}}_{y^*}(y) = 1 + \frac{\sum_i y^*_i(1-y_i) - \sum_i y^*_i}{\sum_i y^*_i + \sum_i y_i(1-y^*_i)}. \quad (6)$$

Let $N^+ = \sum_i y^*_i$, $N_0 = \sum_{i:y^*_i=1}(1-y_i)$, and $N_1 = \sum_{i:y^*_i=0} y_i$ be the number of ground truth pixels, false negatives, and false positives, respectively. We can rewrite the loss as $\Delta^{\text{PASCAL}}_{y^*}(y) = 1 + \frac{N_0 - N^+}{N^+ + N_1}$ and the objective inside the maximization in (5) as

$$f(N_0, N_1) = \frac{N_0 - N^+}{N^+ + N_1} + s_0(N_0) + s_1(N_1) + \kappa, \quad (7)$$

where $s_0(N_0)$ is the cumulative sum of the first N_0 sorted negative incoming message values from variables where $y^* = 1$ and $s_1(N_1)$ is the cumulative sum of the first N_1 sorted incoming message values from variables where $y^* = 0$, and κ is a constant.[1] After relaxing N_0 and N_1 to be real-valued, f is quasi-concave restricted to the domain of interest: $N_0, N_1 \geq 0$.

The key insight is that the optimal setting of y, and thus the full outgoing message optimization (5) can be easily computed from the choice of N_0 and N_1 that optimizes f. A simple (but possibly suboptimal) optimization approach that we found to work well is to perform local search using the neighborhood achieved by incrementing or decrementing either N_0 or N_1 by 1.

Empirically, by reusing intermediate solutions, solving the first optimization takes linear time, then solving the latter N problems takes constant time each (usually 0, 1, or 2 steps of local search). Thus, we expect the sort operation to dominate the complexity and runtime for computing *all* N outgoing messages from a factor to scale like $N \log N$ i.e., $\log N$ time amortized per message. Indeed, for an image with ten thousand pixels, like we use in our experiments, *exact* computation at the loss factor in each iteration takes .03 seconds. Empirical runtimes on larger problems indicate the scaling to be roughly $N \log N$ (10k pixels: .03s, 100k pixels: .32s, 1M pixels: 3.3s, 10M pixels: 34.5s).

Bounding Box Loss: Suppose we are given bounding box labels at training time and wish to optimize for performance on a per-pixel segmentation task. This situation may arise if we are training a segmentation model and would like to augment the set of detailed per-pixel ground truths with a set of easier-to-obtain, coarser bounding box ground truths. One assumption that we might make is that bounding boxes contain roughly a fraction R foreground pixels (and $1 - R$ background pixels). Let y^* be the coarse labeling where y^*_i indicates whether pixel i is within the bounding box. A reasonable loss function is

$$\Delta^{\text{BB}}_{y^*}(y) = \sum_{i:y^*_i=0} y_i + \left| \sum_{i:y^*_i=1} y^*_i y_i - R \sum_{i:y^*_i=1} y^*_i \right|. \quad (8)$$

This loss can be decomposed into low order parts over pixels outside the bounding box and a high order part over pixels inside the bounding box. The high order term can be written as a function of the number of pixels on inside the bounding box i.e., $f(\sum_{i:y^*_i=1} y_i)$, so the cardinality potential from [7] can be used without modification.

Local Border Convexity Loss: A second form of weak labeling that is easier for humans to produce than a full per-pixel labeling is a rough inner-bound on the true labeling plus a rough outline of the object to serve as an outer bound.

[1]This gives the basic idea but skips some steps and ignores some subtleties.

For example, a popular form of labeling in interactive image segmentation derives from a user drawing a few strokes to mark the internal skeleton of the object, and a crude circular stroke around the outside. We assume that the strokes define the inner core of the object, so if follow any ray extending out from the stroke towards the boundary of the object, the labeling along the ray will be *monotonic*—i.e., of the form $1^m 0^n$. To maximize the loss, we need to define a *not-monotonic* factor, where all N outgoing messages can be computed in $O(N)$ time, i.e., $O(1)$ amortized time per message.

4. Experimental Evaluation

Data: We took subsets of images from the PASCAL VOC Segmentation challenge data set containing a given object—Aeroplane, Car, Cow, and Dog. We then created ground truth labels by assigning a pixel to foreground if it was labeled $\{$Aeroplane, Car, Cow, Dog$\}$ in the VOC labels and background otherwise. We scaled the images so that the minimum dimension was 100 pixels.

Model: We use 84 per-pixel features that represent color and texture in the patch surrounding a pixel (the **unary** model). We use a standard pairwise 4-connected grid model, with 1 constant pairwise feature and 12 pairwise features that are based on boundary appearance (the **unary + pairwise** model). Weights on pairwise features are constrained to be positive, so that the resulting pairwise potentials are submodular.

Inference: We use the COMPOSE framework [1] to compute messages for the entire pairwise model using dynamic graph cuts [4]. With decomposable (per-pixel) losses, this guarantees that inference is exact, even though the grid graph contains loops. With high order losses, inference is not guaranteed to be exact, but we find this framework to work significantly better than standard max-product belief propagation with a static message passing schedule.

4.1. PASCAL Loss

We first examine how training on different loss functions affects test performance. We look at three loss functions: **0-1 Loss** is the constant-margin structural SVM (1), while **Pixel Loss** and **PASCAL Loss** are loss-augmented structural SVM training with the respective loss functions. We pair each of these loss functions with different evaluations at test time, pixel and PASCAL accuracy.

We also evaluate the tradeoff associated with using pairwise potentials in the model. On one hand, we know that objects in images are smooth, so introducing pairwise interactions should make the model more realistic. On the other hand, when paired with high order losses, loss-augmented MAP inference is no longer guaranteed to be exact, which could possibly hurt learning performance.

See Fig. 2 for Aeroplane results and Fig. 3 for other objects. We show qualitative results in Fig. 4. Visually, the PASCAL-trained model labels more pixels foreground, correcting the overly conservative pixel-trained model.

We will show more results, including those for the other losses at the poster.

Train \ Evaluate	Pixel Acc.	PASCAL Acc.
0-1 Loss	82.1%	28.6
Pixel Loss	**91.2%**	47.5
PASCAL Loss	88.5%	**51.6**

(a) Unary only model

Train \ Evaluate	Pixel Acc.	PASCAL Acc.
0-1 Loss	79.0%	28.8
Pixel Loss	**92.7%**	54.1
PASCAL Loss	90.0%	**58.4**

(b) Unary + pairwise model.

Figure 2. Test accuracies for training-test loss function combinations on Aeroplane. A labeling of all background gives 86.6% pixel accuracy, but 0 PASCAL accuracy.

	Train \ Evaluate	Pixel Acc.	PASCAL Acc.
Car	Pixel Loss	**80.4%**	6.7
	PASCAL Loss	72.9%	**37.0**
Cow	Pixel Loss	**80.3%**	23.3
	PASCAL Loss	79.4%	**48.1**
Dog	Pixel Loss	**81.5%**	16.6
	PASCAL Loss	75.9%	**38.3**

Figure 3. Test accuracies for training-test loss function combinations on other objects. All models are unary + pairwise. A labeling of all background gives pixel accuracies of 79.8%, 78.9%, and 80.2%, on Car, Cow, and Dog respectively.

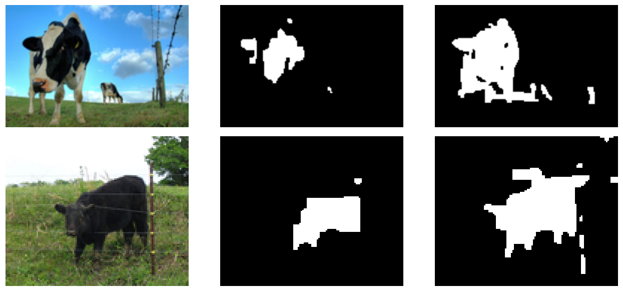

Figure 4. Example test results on Cow dataset. From left to right: (Left) Input. (Middle) Pixel Loss. (Right) PASCAL Loss.

References

[1] J. Duchi, D. Tarlow, G. Elidan, and D. Koller. Using combinatorial optimization within max-product belief propagation. In *NIPS*, 2007. 4

[2] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL visual object classes (VOC) challenge. *IJCV*, 88(2):303–338, June 2010. 1

[3] T. Joachims, T. Finley, and C.-N. Yu. Cutting-plane training of structural SVMs. *Machine Learning*, 77(1):27–59, 2009. 2

[4] P. Kohli and P. H. S. Torr. Dynamic graph cuts for efficient inference in markov random fields. *PAMI*, 29(12):2079–2088, 2007. 4

[5] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu. BLEU: a method for automatic evaluation of machine translation. In *ACL*, 2002. 1

[6] C. Rother and S. Nowozin. *Tutorial at CVPR 2010:* Higher-order models in computer vision, 2010. 2

[7] D. Tarlow, I. Givoni, and R. Zemel. HOP-MAP: Efficient message passing for high order potentials. In *AISTATS*, 2010. 3

[8] I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun. Large margin methods for structured and interdependent output variables. 2