Marginal Weighted Maximum Log-likelihood for Efficient Learning of Perturb-and-Map models

Tatiana Shpakova, Francis Bach, Anton Osokin

Marginal Weighted Maximum Log-likelihood for Efficient Learning of Perturb-and-Map models

Tatiana Shpakova

INRIA - ENS
PSL Research University
Paris, France

Francis Bach

INRIA - ENS
PSL Research University
Paris, France

Anton Osokin

National Research University
Higher School of Economics
Moscow, Russia

Abstract

We consider the structured-output prediction problem through probabilistic approaches and generalize the "perturb-and-MAP" framework to more challenging weighted Hamming losses, which are crucial in applications. While in principle our approach is a straightforward marginalization, it requires solving many related MAP inference problems. We show that for log-supermodular pairwise models these operations can be performed efficiently using the machinery of dynamic graph cuts. We also propose to use *double* stochastic gradient descent, both on the data and on the perturbations, for efficient learning. Our framework can naturally take weak supervision (e.g., partial labels) into account. We conduct a set of experiments on medium-scale character recognition and image segmentation, showing the benefits of our algorithms.

1 INTRODUCTION

Structured-output prediction is an important and challenging problem in the field of machine learning. When outputs have a structure, often in terms of parts or elements (e.g., pixels, sentences or characters), methods that do take it into account typically outperform more naive methods that consider outputs as a set of independent elements. Structured-output methods based on optimization can be broadly separated in two main families: *max-margin methods*, such as structured support vector machines (SSVM) (Taskar et al., 2003; Tsochantaridis et al., 2005) and *probabilistic methods based on maximum likelihoods* such as conditional random fields (CRF) (Lafferty et al., 2001).

Structured-output prediction faces many challenges:

(1) on top of large input dimensions, problems also have large outputs, leading to scalability issues, in particular when prediction or learning depends on combinatorial optimization problems (which are often polynomial-time, but still slow given they are run many times); (2) it is often necessary to use losses which go beyond the traditional 0-1 loss to shape the behavior of the learned models towards the final evaluation metric; (3) having fully labelled data is either rare or expensive and thus, methods should be able to deal with weak supervision.

Max-margin methods can be used with predefined losses, and have been made scalable by several recent contributions (see, e.g., Lacoste-Julien et al., 2013, and references therein), but do not deal naturally with weak supervision. However, a few works (Yu and Joachims, 2009; Kumar et al., 2010; Girshick et al., 2011) incorporate weak supervision into the max-margin approach via the CCCP (Yuille and Rangarajan, 2003) algorithm.

The flexibility of probabilistic modeling naturally allows (a) taking into consideration weak supervision and (b) characterizing the uncertainty of predictions, but it comes with strong computational challenges as well as a non-natural way of dealing with predefined losses beyond the 0-1 loss. The main goal of this paper is to provide new tools for structured-output inference with probabilistic models, thus making them more widely applicable, while still being efficient. There are two main techniques to allow for scalable learning in CRFs: stochastic optimization (Vishwanathan et al., 2006) and piecewise training (Sutton and McCallum, 2005, 2007; Kolesnikov et al., 2014); note that the techniques above can also be used for weak supervision (and we reuse some of them in this work).

Learning and inference in probabilistic structured-output models recently received a lot of attention from the research community (Bakir et al., 2007; Nowozin and Lampert, 2011; Smith, 2011). In this paper we consider only models for which maximum-a posteriori (MAP)

inference is feasible (a step often referred to as decoding in max-margin formulations, and which typically makes them tractable). A lot of efforts were spent to explore MAP-solvers algorithms for various problems, leveraging various structures, e.g., graphs of low tree-width (Bishop, 2006; Wainwright and Jordan, 2008; Sontag et al., 2008; Komodakis et al., 2011) and function submodularity (Boros and Hammer, 2002; Kolmogorov and Zabih, 2004; Bach, 2013; Osokin and Vetrov, 2015).

Naturally, the existence of even an exact and efficient MAP-solver does not mean that the partition function (a key tool for probabilistic inference as shown below) is tractable to compute. Indeed, the partition function computation is known to be $\#P$-hard (Jerrum and Sinclair, 1993) in general. For example, MAP-inference is efficient for log-supermodular probabilistic models, while computation of their partition function is not (Djolonga and Krause, 2014).

For such problems where MAP-inference is efficient, but partition function computation is not, "perturb-and-MAP" ideas such as proposed by Papandreou and Yuille (2011); Hazan and Jaakkola (2012) are a very suitable treatment. By adding random perturbations, and then performing MAP-inference, they can lead to estimates of the partition function. In Section 2, we review the existing approaches to the partition function approximation, parameter learning and inference.

An attempt to learn parameters via "perturb-and-MAP" ideas was made by Hazan et al. (2013), where the authors have developed a PAC-Bayesian-flavoured approach for the non-decomposable loss functions. While the presented algorithm has something in common with ours (gradient descent optimization of the objective upper bound), it differs in the sense of the objective function and the problem setup, which is more general but that requires a different (potentially with higher variance) estimates of the gradients. Such estimates are usual in reinforcement learning, e.g., the log-derivative trick from the REINFORCE algorithm (Williams, 1992).

The goal of this paper is to make the "perturb-and-MAP" technique applicable to practical problems, in terms of (a) scale, by increasing the problem size significantly, and (b) losses, by treating structured losses such as the Hamming loss or its weighted version, which are crucial to obtaining good performance in practice.

Overall, we make the following contributions:

– In Section 3, we generalize the "perturb-and-MAP" framework to more challenging weighted Hamming losses which are commonly used in applications. In principle, this is a straightforward marginalization but this requires solving many related MAP inference prob-

lems. We show that for graph cuts (our main inference algorithm for image segmentation), this can be done particularly efficiently. Besides that, we propose to use a *double* stochastic gradient descent, both on the data and on the perturbations.

– In Section 4, we show how weak supervision (e.g., partial labels) can be naturally dealt with. Our method in this case relies on approximating marginal probabilities that can be done almost at the cost of the partition function approximation.

– In Section 5, we conduct a set of experiments on medium-scale character recognition and image segmentation, showing the benefits of our new algorithms.

2 PERTURB-AND-MAP

In this section, we introduce the notation and review the relevant background. We study the following probabilistic model (a.k.a. a Gibbs distribution) over a discrete product space $Y = Y_1 \times \cdots \times Y_D$,

$$P(y) = \frac{1}{Z(f)} e^{f(y)}, \qquad (1)$$

which is defined by a potential function $f : Y \to \mathbb{R}$. The constant $Z(f) = \sum_{y \in Y} e^{f(y)}$ is called the partition function and normalizes $P(y)$ to be a valid probability function, i.e., to sum to one. $Z(f)$ is in general intractable to compute as the direct computation requires summing over exponentially (in D) many elements.

Various partition function approximations methods have been used in parameter learning algorithms (Parise and Welling, 2005), e.g., mean-field (MF, Jordan et al., 1999), tree-reweighted belief propagation (TRW, Wainwright and Jordan, 2008) or loopy belief propagation (LBP, Weiss, 2001). We will work with the upper bound on the partition function proposed by Hazan and Jaakkola (2012) as it allows us to approximate the partition function via MAP-inference, calculate gradients efficiently, approximate marginal probabilities and guarantee tightness for some probabilistic models. We introduce this class of techniques below.

2.1 Gumbel perturbations

Recently, Hazan and Jaakkola (2012) provided a general-purpose upper bound on the *log-partition function* $A(f) = \log Z(f)$, based on the "perturb-and-MAP" idea (Papandreou and Yuille, 2011): maximize the potential function perturbed by Gumbel-distributed noise.[1]

---

[1] The Gumbel distribution on the real line has cumulative distribution function $F(z) = \exp(-\exp(-(z+c)))$, where c is the Euler constant.

Proposition 1 (Hazan and Jaakkola (2012), Corollary 1)**.** *For any function* $f : Y \to \mathbb{R}$, *we have* $A(f) \leqslant A_\mathrm{G}(f)$, *where*

$$A_\mathrm{G}(f) = \mathbb{E}_{z_1,\dots,z_D \sim \mathrm{Gumbel}} \Big[\max_{y \in Y} \Big(f(y) + \sum_{d=1}^{D} z_d(y_d) \Big) \Big]. \quad (2)$$

Gumbel *denotes the Gumbel distribution and* $\{z_d(y_d)\}_{y_d \in Y_d}^{d=1,\dots,D}$ *is a collection of independent Gumbel samples.*

The bound is tight when $f(y)$ is a separable function (i.e., a sum of functions of single variables), and the tightness of this bound was further studied by Shpakova and Bach (2016) for log-supermodular models (where f is supermodular). They have shown that the bound A_G is always lower (and thus provide a better bound) than the "L-field" bound proposed by Djolonga and Krause (2014, 2015), which is itself based on separable optimization on the base polytope of the associated supermodular function.

The partition function bound A_G can be approximated by replacing the expectation by an empirical average. That is, to approximate it we need to solve a large number (as many as the number of Gumbel samples used to approximate the expectation) of MAP-like problems (i.e., maximizing f plus a separable function) which are feasible by our assumption. Strictly speaking, the MAP-inference is NP-hard in general, but firstly, it is much easier than the partition function calculation, secondly, there are solvers for special cases, e.g., for log-supermodular models (which include functions f which are negatives of cuts (Kolmogorov and Zabih, 2004; Boykov and Kolmogorov, 2004)) and those solvers are often efficient enough in practice. In this paper, we will focus primarily on a subcase of supermodular potentials, namely negatives of graph cuts.

2.2 Parameter learning and Inference

In the standard supervised setting of structured prediction, we are given N pairs of observations $\mathcal{D} = \{(x^n, y^n)\}_{n=1}^{N}$, where x^n is a feature representation of the n-th object and $y^n \in Y = Y_1 \times \cdots \times Y_{D^n}$ is a structured vector of interest (e.g., a sequence of tags, a segmentation MAP or a document summarization representation). In the standard linear model, the potential function $f(y|x)$ is represented as a linear combination: $f(y|x) = w^T \Psi(x, y)$, where w is a vector of weights and the structured feature map $\Psi(x, y)$ contains the relevant information for the feature-label pair (x, y). To learn the parameters using the predefined probabilistic model, one can use the (regularized) maximum likelihood approach:

$$\max_{w} \frac{1}{N} \sum_{n=1}^{N} \log P(y^n | x^n, w) - \frac{\lambda}{2} \|w\|^2, \quad (3)$$

where $\lambda > 0$ is a regularization parameter and the likelihood $P(y|x, w)$ is defined as $\frac{\exp(f(y|x))}{Z(f,x)} = \exp(f(y|x) - A(f, x))$, where $A(f, x)$ is the log-partition function (that now depends on x, since we consider conditional models).

Hazan and Jaakkola (2012) proposed to learn parameters based on the Gumbel bound $A_G(f, x)$ instead of the intractable log-partition function:

$$\log P(y|x) = f(y|x) - A(f, x) \leq f(y|x) - A_G(f, x)$$

$$= f(y|x) - \mathbb{E}_z \Big[\max_{y \in Y} \Big\{ \sum_{d=1}^{D} z_d(y_d) + f(y) \Big\} \Big]$$

$$\approx f(y|x) - \frac{1}{M} \sum_{m=1}^{M} \max_{y^{(m)} \in Y} \Big\{ \sum_{d=1}^{D} z_d^{(m)}(y_d^{(m)}) + f(y^{(m)}) \Big\}.$$

Hazan and Jaakkola (2012) considered the fully-supervised setup where labels y^n were given for all data points x^n. Shpakova and Bach (2016) developed the approach, but also considered a setup with missing data (part of the labels y^n are unknown) for the small Weizmann Horse dataset (Borenstein et al., 2004). Leveraging the additional stochasticity present in the Gumbel samples, Shpakova and Bach (2016) extend the use of stochastic gradient descent, not on the data as usually done, but on the Gumbel randomization. It is equivalent to the choice of parameter $M = 1$ for every gradient computation (but with a new Gumbel sample at every iteration). In our work, we use the stochastic gradient descent in a regime stochastic w.r.t. *both* the data and the Gumbel perturbations. This allows us to apply the method to large-scale datasets.

For linear models, we have $f(y|x) = w^T \Psi(x, y)$ and $\Psi(x, y)$ is usually given or takes zero effort to compute. We assume that the gradient calculation $\nabla_w f(y|x) = \Psi(x, y)$ does not add complexity to the optimization algorithm. The gradient of $\log P(y|x)$ is equal to $\nabla_w f(y|x) - \nabla_w \max_{y \in Y} \Big\{ \sum_{d=1}^{D} z_d(y_d) + f(y|x) \Big\} = \nabla_w f(y|x) - \nabla_w f(y^*|x)$, where y^* lies in $\arg\max$ of the perturbed optimization problem. The gradient of $\langle \log P(y|x) \rangle$ (the average over a subsample of data, typically a mini-batch) has the form $\langle \nabla_w f(y|x) \rangle - \langle \nabla_w f(y^*|x) \rangle = \langle \Psi(x, y) \rangle_{emp.} - \langle \Psi(x, y^*) \rangle$, where $\langle \Psi(x, y) \rangle_{emp.}$ denotes the empirical average over the data. Algorithm 1 contains this double stochastic gradient descent (SGD) with stochasticity w.r.t. both sampled data and Gumbel samples. The choice of the stepsize $\gamma_h = \frac{1}{\lambda h}$ is standard for strongly-convex problems (Shalev-Shwartz et al., 2011).

Note, that the classic log-likelihood formulation (3) is implicitly considering a "0-1 loss" $l_{0\text{-}1}(y, \hat{y}) = [y \neq \hat{y}]$ as it takes probability of the entire output object y^n conditioned on the observed feature representation x^n.

However, in many structured-output problems 0-1 loss

Algorithm 1 Double SGD: stochasticity w.r.t. data and Gumbel samples

---

Input: dataset $\mathcal{D} = \{(x^n, y^n)\}_{n=1}^N$, number of iterations H, size of the mini-batch T, stepsize sequence $\{\gamma_h\}_{h=1}^H$, regularization parameter λ
Output: model parameters w

1: **Initialization**: $w = 0$
2: **for** $h = 1$ **to** H **do**
3:    Sample data mini-batch of small size T (that is, T pairs of observations)
4:    Calculate sufficient statistics $\langle \Psi(x, y) \rangle_{emp.}$ from the mini-batch
5:    **for** t=1 **to** T **do**
6:        Sample $z_d(y_d)$ as independent Gumbels for all $y_d \in Y_d$ and for all d
7:        Find $y^* \in \arg\max_{y \in Y} \left\{ \sum_{d=1}^D z_d(y_d) + f(y) \right\}$
8:    **end for**
9:    Make a gradient step:
$$w_{h+1} \to w_h + \gamma_h \left(\langle \Psi(x, y) \rangle_{emp.} - \langle \Psi(x, y^*) \rangle - \lambda w_h \right)$$
10: **end for**

---

evaluation is not an adequate performance measure. The Hamming or weighted Hamming losses that sum mistakes across the D elements of the outputs, are more in demand as they count misclassification per element.

2.3 Marginal probability estimation

Either at testing time (to provide an estimate of the uncertainty of the model) or at training time (in the case of weak supervision, see Section 4), we need to compute marginal probabilities for a single variable y_d out of the d ones, that is,

$$P(y_d|x) = \sum_{y_{-d}} P(y_{-d}, y_d|x),$$

where y_{-d} is a sub-vector of y obtained by elimination of the variable y_d. Following Hazan and Jaakkola (2012) and Shpakova and Bach (2016), this can be obtained by taking m Gumbel samples and the associated maximizers $y^m \in Y = Y_1 \times \cdots \times Y_D$, and, for any particular d, counting the number of occurrences in each possible value in all the d-th components y_d^m of the maximizers y^m.

While this provides an estimate of the marginal probability, this is not an easy expression to optimize at it depends on several maximizers of potentially complex optimization problems. In the next section, we show how we can compute a different (and new) approximation which is easily differentiable and on which we can apply stochastic gradient descent.

3 MARGINAL LIKELIHOOD

In this section, we demonstrate the learning procedure for the element-decoupled losses. We consider the regularized empirical risk minimization problem in a general form:

$$\max_w \frac{1}{N} \sum_{n=1}^N \ell(w, x^n, y^n) - \frac{\lambda}{2} \|w\|^2, \tag{4}$$

where $\ell(w, x, y)$ can take various forms from Table 1 and λ is the regularization parameter. The choice of the likelihood form is based on the problem setting such as presence of missing data and the considered test-time evaluation function.

3.1 Hamming loss

The Hamming loss is a loss function that counts misclassification per dimension: $l_h(y, \hat{y}) = \frac{1}{D} \sum_{d=1}^D [y_d \neq \hat{y}_d]$. For this type of loss instead of the classic log-likelihood objective it is more reasonable to consider the following decoupling representation from Table 1:

$$\ell(w, x, y) = \sum_{d=1}^D \log P(y_d|x, w), \tag{5}$$

where

$$
\begin{aligned}
P(y_d|x, w) &= \sum_{y_{-d}} \exp(f(w, y_{-d}|y_d, x) - A(f, x)) \\
&= \exp(B(f, y_d, x) - A(f, x))
\end{aligned}
$$

is the marginal probability of the single element y_d given the entire input x, and $B(f, y_d) = \log \sum_{y_{-d}} \exp(f(w, y_{-d}|y_d))$, where $y_{-d} \in Y_1 \times \ldots Y_{d-1} \times Y_{d+1} \times \cdots \times Y_D$. Thus, the log-marginal probability may be obtained from the difference of two log-partition functions (which we will approximate below with Gumbel samples).

This idea of considering the marginal likelihood was proposed by Kakade et al. (2002). Our contribution is to consider the approximation by "perturb-and-MAP" techniques. We thus have a new objective function $\ell(w, x, y)$: $\ell(w, x, y) = \sum_{d=1}^D [(B(f, x, y_d) - A(f, x))]$, and now the following approximation could be applied:

$$A(f) \approx A_G(f) = \mathbb{E}_z \left\{ \max_{y \in Y} \sum_{d=1}^D z_d(y_d) + f(y) \right\},$$

$$
\begin{aligned}
B(f|y_d) &\approx B_G(f|y_d) \\
&= \mathbb{E}_z \left\{ \max_{y_{-d} \in Y_{-d}} \sum_{s=1: s \neq d}^D z_s(y_s) + f(y_{-d}|y_d) \right\}.
\end{aligned}
$$

It is worth noting, that the approximation is not anymore an upper bound of the marginal likelihood; moreover it is a difference of convex functions. Remarkably, the objective function exactly matches the log-likelihood in the

Table 1: Variants of the Objective Loss $\ell(w, x, y)$ Function. $\{\theta_d(y_d)\}_{d=1}^D$ are the weights of the weighted Hamming loss, $\{q_d(y_d)\}_{d=1}^D$ are the marginal probabilities $P(y_d|x)$.

Loss	Labelled Data	Unlabelled Data
0-1	$\log P(w, y|x)$	$\log \sum_{y \in Y} P(w, y, x)$
Hamming	$\sum_{d=1}^D \log P(w, y_d|x_d)$	$\sum_{d=1}^D \sum_{y_d \in Y_d} q_d(y_d) \log P(w, y_d|x_d)$
Weighted Hamming	$\sum_{d=1}^D \theta_d(y_d) \log P(w, y_d|x_d)$	$\sum_{d=1}^D \sum_{y_d \in Y_d} q_d(y_d) \theta_d(y_d) \log P(w, y_d|x_d)$

case of unary potentials (separable potential function) as the log-likelihood function becomes the sum of marginal likelihoods.

As noted, the objective $\ell(w, x, y)$ is not convex anymore, but it is presented as the difference of two convex functions. We can still try to approximate with stochastic gradient descent (which then only converges to a stationary point, typically a local minimum). Algorithm 2 describes the implementation details.

Algorithm 2 Double SGD for Marginal Likelihood

Input: dataset $\mathcal{D} = \{(x^n, y^n)\}_{n=1}^N$, number of iterations H, size of the mini-batch T, stepsize sequence $\{\gamma_h\}_{h=1}^H$, regularization parameter λ
Output: model parameters w
1: **Initialization**: $w = 0$
2: **for** $h = 1$ **to** H **do**
3:     Sample data mini-batch of small size T (that is, T pairs of observations)
4:     **for** $t=1$ **to** T **do**
5:         Sample $z_d(y_d)$ as independent Gumbels for all $y_d \in Y_d$ and for all d
6:         Find $y_A^* \in \arg\max_{y \in Y} \left\{ \sum_{d=1}^D z_d(y_d) + f(y) \right\}$
7:         **for** $d=1$ **to** D **do**
8:           Find $y_B^* \in \arg\max_{y_{-d} \in Y_{-d}} \left\{ \sum_{s=1:s \neq d}^D z_s(y_s) + f(y_{-d}|y_d) \right\}$
9:         **end for**
10:    **end for**
11:    Make a gradient step:
    $w_{h+1} \rightarrow w_h + \gamma_h \left(\langle \langle \Psi(x, y_B^*) \rangle - \Psi(x, y_A^*) \rangle - \lambda w_h \right)$
12: **end for**

Acceleration trick. Interestingly we can use the same Gumbel perturbation realizations for approximating $A_G(f)$ and $B_G(f|y_d)$ through an empirical average. On the one hand, this restriction should not influence on the result as with a sufficient large averaging number M, $A_G(f)$ and $B_G(f|y_d)$ converges to their expectations. This is the same for stochastic gradients: on every itera-

tion, we use a different Gumbel perturbation, but we share this one for the estimation of the gradients of $A_G(f)$ and $B_G(f|y_d)$. This allows us to save some computations as shown below, while preserving convergence (the extra correlation added by using the same samples for the two gradients does not change the unbiasedness of our estimates).

Moreover, if y_A^* has the same label value y_d as the ground truth, then the MAP inference problem for y_A^* exactly matches the one for y_B^* (with the element y_d fixed from the ground truth). Then $y_A^* = y_B^*$ and the corresponding difference of gradients gives zero impact into the gradient step. This fact allows us to reduce the number of MAP-inference problems. We should thus calculate y_B^* only for those indices d that leads to a mismatch between d-th label of y_A^* and the ground truth one. Remarkably, during the convergence to the optimal value, the reduction will occur more often and decrease the execution time with the number of iteration increase. Besides that, in the experiments with graph cuts in Section 5 we use dynamic graph cut algorithm for solving several optimization problems of similar structure (here D marginal probabilities calculation). We describe it in more details in Section 3.3.

3.2 Weighted Hamming loss

The weighted Hamming loss is used for performance evaluation in the models, where each dimension has its own level of importance, e.g., in an image segmentation with superpixels, proportional to the size of superpixels. It differs from the usual Hamming loss in this way: $l_h(y, \hat{y}) = \frac{1}{D} \sum_{d=1}^D \theta_d(y_d)[y_d \neq \hat{y}_d]$.

Thus we consider a dimension-weighted model as it can be adjusted for the problem of interest that gives the model more flexibility. The optimization problem of interest is transformed from the previous case by weighted multiplication:

$$\ell(w, x, y) = \sum_{d=1}^D \theta_d(y_d) \left[(B(f, x, y_d) - A(f, x)) \right]. \quad (6)$$

To justify this objective function, we notice that in the case of unit weights, the weighted loss and objective

function (6) match the loss and the objective from the previous section. Furthermore, y_d with a large weight $\theta_d(y_d)$ puts more importance towards making the right prediction for this y_d, and that is why we put more weight on the d-th marginal likelihood. This corresponds to the usual rebalancing used in binary classification (see, e.g., Bach et al., 2006, and references therein). Then, the algorithm for this case duplicated the one for the usual Hamming loss and the acceleration trick can be used as well.

3.3 Scalable algorithms for graph cuts

As a classical efficient MAP-solver for pairwise potentials problem we will use graph cut algorithms from Boykov and Kolmogorov (2004). The function $f(y|x)$ should then be supermodular, i.e., with pairwise potentials, all pairwise weights of w should remain negative.

In both Sections 3.1 and 3.2 we can apply the dynamic graphcut algorithm proposed by Kohli and Torr (2007), which is a modification of the Boykov-Kolmogorov graph-cut algorithms. It is dedicated to situations when a sequence of graphcut problems with slightly different unary potentials need to be solved. Then, instead of solving them separately, we can use the dynamic procedure and find the solutions for slightly different problems with less costs. This makes graphcut scalable for a special class of problems.

It can easily be seen that our sequence of problems $y_B^* \in \arg\max\limits_{y_{-d}\in Y_{-d}} \{ \sum\limits_{s=1:s\neq d} z_s(y_s) + f(y_{-d}|y_d) \}$ for $d = 1, \ldots, D$ is a perfect application for the dynamic graph cut algorithm. At each iteration we solve T sets of graph cut problems, each of set contains $1 + a_t$ problems solvable by the same dynamic cut, where a_t is the number of not matched pixels between y_A^* and ground truth y^n. Finally, using acceleration trick and dynamic cuts we reduce the gradient descent iteration complexity from $\sum_{t=1}^{T} (1 + D_t)$ graphcut problems to T dynamic graph cut problems. We make the approach scalable and can apply it for large datasets.

4 PARAMETER LEARNING IN THE SEMISUPERVISED SETUP

In this section we assume the presence of objects with unknown labels in the train dataset. We can separate the given data in two parts: fully annotated data $\mathcal{D}_1 = \{(x^n, y^n)\}_{n=1}^{N}$ as in the supervised case and unlabeled data $\mathcal{D}_2 = \{x^l\}_{l=1}^{L}$. Then, the optimal model parameter w is a solution of the following optimization problem:

$$\max_{w} L_1(w) + \kappa L_2(w) - \frac{\lambda}{2}\|w\|^2, \qquad (7)$$

where $L_1(w) = \sum_{n=1}^{N} \ell_1(w, x^n, y^n)$, $L_2(w) =$

$\sum_{l=1}^{L} \ell_2(w, x^l)$ and the parameter κ governs the importance of the unlabeled data. $\ell_1(w, x^n, y^n)$ can have a form from the left column of the Table 1, and $\ell_2(w, x^l)$ from the right one.

Marginal calculations. It is worth reminding from Section 2.3, that we can approximate marginal probabilities $q(y)$ of holding $y_d = k$ along with the partition function approximation almost for free. This can be obtained by taking m Gumbel samples and the associated maximizers $y^m \in Y = Y_1 \times \cdots \times Y_D$, and, for any particular d, counting the number of occurrences in each possible value in all the d-th components y_d^m of the maximizers y^m. The approximation accuracy depends on number of samples M. To calculate this we already need to have a trained weight vector w which we can obtained from the fully annotated dataset $\mathcal{D} = \{(x^n, y^n\}_{n=1}^{N})$. We will calculate $q(y)$ for the unlabelled data $\mathcal{D}_2 = \{x^l\}_{l=1}^{l}$. Those marginal probabilities contain much more information than MAP inference for the new data as can be seen on the example in Figure 1. We believe that proper use of the marginal probabilities will help to gain better result than using labels from the MAP inference (which we observe in experiments).

It is worth noting that for the inference and learning phases we use a different number of Gumbel samples. During the learning phase, we incorporate the double stochastic procedure and use 1 sample per 1 iteration and 1 label. For the marginal calculation (inference) we should use large number of samples (e.g. 100 samples) to get accurate approximation.

Algorithm 3 Sketch for the semisupervised algorithm.

Input: fully annotated dataset $\mathcal{D}_1 = \{(x^n, y^n)\}_{n=1}^{N}$, number of iterations H, size of the mini-batch T, stepsize sequence $\{\gamma_h\}_{h=1}^{H}$, regularization param. λ
Output: model parameters w_1
 1: **Initialization**: $w_1 = 0$
 2: **Find w_1 via Algorithm 2**
Input: fully annotated dataset $\mathcal{D}_1 = \{(x^n, y^n)\}_{n=1}^{N}$, unlabeled dataset $\mathcal{D}_2 = \{x^l\}_{l=1}^{L}$, number of iterations H, size of the mini-batch T, stepsize sequence $\{\gamma_h\}_{h=1}^{H}$, regularization parameter λ
Output: model parameters $w_{1,2}$
 3: **Initialization**: $w_{1,2} = w_1$
 4: **Calculate**: $q(y)$ for unlabeled data via w_1
 5: **Find $w_{1,2}$ via mixture of Algorithms 2 and 4**

We provide the sketch of the proposed optimization algorithm in Algorithm 3. The optimization of L_1 is fully supervised and this can be done with tools of the previous section. The optimization of L_2 requires the specification of $\ell_2(w, x)$, which we take as

$$\ell_2(w,x) = \sum_{d=1}^{D} \sum_{y_d \in \{0,\dots,K\}} q_d(y_d) \log P(w, y_d | x_d) =$$

$\sum_{d=1}^{D} \sum_{y_d \in \{0,\dots,K\}} q_d(y_d) B(f|y_d) - DA(f)$, that is, the average of the fully supervised cost function with labels generated from the model q. The term L_2 corresponds to the common way of treating unlabeled data through the marginal likelihood. The sub-algorithm for the optimization of ℓ_2 is presented as Algorithm 4.

Algorithm 4 Double SGD for Unsupervised Learning

Input: unlabeled dataset $\mathcal{D}_2 = \{x^l\}_{l=1}^{L}$, parameter estimate w_1, number of iterations H, size of the mini-batch T, stepsize sequence $\{\gamma_h\}_{h=1}^{H}$, regularization parameter λ

Output: model parameters $w_{1,2}$

1: **Initialization**: $w_{1,2} = w_1$
2: **for** $h = 1$ **to** H **do**
3:     Sample data mini-batch of small size T (that is, T pairs of observations)
4:     **for** $t=1$ **to** T **do**
5:         Sample $z_d(y_d)$ as independent Gumbels for all $y_d \in Y_d$ and for all d
6:         Find $y_A^* \in \arg\max_{y \in Y} \left\{ \sum_{d=1}^{D} z_d(y_d) + f(y) \right\}$
7:         **for** $d=1$ **to** D **and** $k=0$ **to** K **do**
8:             Find $y_{B,d,k}^* \in \arg\max_{y_{-d} \in Y_{-d}} \left\{ \sum_{s=1:s\neq d}^{D} z_s(y_s) + f(y_{-d}|y_d = k) \right\}$
9:         **end for**
10:    **end for**
11:    Make a gradient step:
    $w_{h+1} \rightarrow w_h + \gamma_h \Big(\langle \langle \sum_{k=0}^{K} q_d(k) \Psi(x, y_{B,d,k}^*) \rangle - \Psi(x, y_A^*) \rangle - \lambda w_h \Big)$
12: **end for**

Acceleration trick. Suppose, that y_d can take values in the range $\{0,\dots,K\}$. Again we use the same Gumbel perturbation for estimating $A_G(f)$ and $B_{dkG}(f|y_d = k)$ for all $k \in \{0,\dots,K\}$. The consequence of using the same perturbations is that if the d-th label y_d of y_A^* takes value k, than the corresponding d-th gradient will cancel out with one of the y_{Bk}^*. Thus, we will calculate only K (instead of $K+1$ labels) structured labels $y_{Bl}^*(l \neq k)$ and reduce the number of optimization problems to be solved. Dynamic graph cuts are applied here as well.

Finally in Table 1 we see the relationships between the proposed objective functions. Firstly, the known labels y^n in the supervised case are equivalent to the binary marginal probabilities $q(y^n) \in \{0,1\}^{D^n}$. Secondly, the unit weights $\theta_d(y_d) = 1$ in the weighted Hamming loss are equivalent to the basic Hamming loss.

Partial labels. Another case that we would like to mention is annotation with partial labels, e.g., in an image segmentation application, the bounding boxes of the images are given. Then denote y^{given} as the set of given labels. In this setup the marginal probabilities become conditional ones $q(y_d|y^{given})$ and to approximate this we need to solve several conditional MAP-inference problems. The objective function $\ell_2(w) = \sum_{d} \sum_{y_d \in \{0,\dots,K\}} q_d(y_d|y^{given}) \log P(w, y_d|x_d, y^{given})$ remains feasible to optimize.

5 EXPERIMENTS

The experimental evaluation consists of two parts: Section 5.1 is dedicated to the chain model problem, where we compare the different algorithms for supervised learning; Section 5.2 is focused on evaluating our approach for the pairwise model on a weakly-supervised problem.

5.1 OCR dataset

The given OCR dataset from Taskar et al. (2003) consists of handwritten words which are separated in letters in a chain manner. The OCR dataset contains 10 folds of ~ 6000 words overall. The average length of the word is ~ 9 characters. Two traditional setups of these datasets are considered: 1) "small" dataset when one fold is considered as a training data and the rest is for test, 2) "large" dataset when 9 folds of 10 compose the train data and the rest is the test data. We perform cross-validation over both setups and present results in Table 2.

As the MAP oracle we use the dynamic programming algorithm of Viterbi (1967). The chain structure also allows us to calculate the partition function and marginal probabilities exactly. Thus, the CRF approach can be applied. We compare its performance with the structured SVM from Osokin et al. (2016), perturb-and-MAP (Hazan and Jaakkola, 2012) and the one we propose for marginal perturb-and-MAP (as Hamming loss is used for evaluation).

The goal of this experiment is to demonstrate that the CRF approach with exact marginals shows a slightly worse performance as the proposed one with approximated marginals but correct Hamming loss.

Table 2: OCR Dataset. Performance Comparison.

method	small dataset	large dataset
CRF	19.5 ± 0.4	13.1 ± 0.8
S-SVM+BCFW	19.5 ± 0.4	12.0 ± 1.1
perturb&MAP	19.1 ± 0.3	12.5 ± 1.1
marg. perturb&MAP	19.1 ± 0.3	12.8 ± 1.2

For the OCR dataset, we performed 10-fold cross-validation and the numbers of Table 2 correspond to the averaged loss function (Hamming loss) values over the 10 folds. As we can see from the result in Table 2, the approximate probabilistic approaches slightly outperforms the CRF on both datasets. The Gumbel approximation (with or without marginal likelihoods) does lead to a better estimation for the Hamming loss. Note that S-SVM performs better in the case of a larger dataset, which might be explained by stronger effects of model misspecification that hurts probabilistic models more than S-SVM (Pletscher et al., 2011).

5.2 HorseSeg dataset

The problem of interest is foreground/background superpixel segmentation. We consider a training set of images $\{x^n\}_{n=1...N}$ that contain different numbers of superpixels. A hard segmentation of the image is expressed by an array $y^n \in \{0,1\}^{D^n}$, where D^n is the number of superpixels for the n-th image.

The HorseSeg dataset was created by Kolesnikov et al. (2014) and contains horse images. The "small" dataset has images with manually annotated labels and contains 147 images. The second "medium" dataset is partially annotated (only bounding boxes are given) and contains 5974 images. The remaining "large" one has 19317 images with no annotations at all. A fully annotated hold out dataset was used for the test stage. It consists of 241 images.

The graphical model is a pairwise model with loops. We consider log-supermodular distribution and thus, the max oracle is available as the graph cut algorithm by Boykov and Kolmogorov (2004). Note that CRFs with exact inference cannot be used here.

Following Kolesnikov et al. (2014), for the performance evaluation the weighted Hamming loss is used, where the weight is governed by the superpixel size and foreground/background ratio in the particular image.

That is, $l_h(y, \hat{y}) = \frac{1}{D} \sum_{d=1}^{D} \theta_d(y_d)[y_d \neq \hat{y}_d]$, where

$$\theta_d(y_d) = \begin{cases} \dfrac{V_d}{2V_{foreground}}, & \text{if } y_d = 1. \\ \dfrac{V_d}{2V_{background}}, & \text{if } y_d = 0. \end{cases}$$

V_d is the size of superpixel d, $V_{background}$ and $V_{foreground}$ are the sizes of the background and the foreground respectively. In this way smaller object sizes have more penalized mistakes.

Since we incorporate $\theta(y)$ into the learning process and for its evaluation we need to know the background and foreground sizes of the image, this formulation is only applicable for the supervised case, where y_d is given for

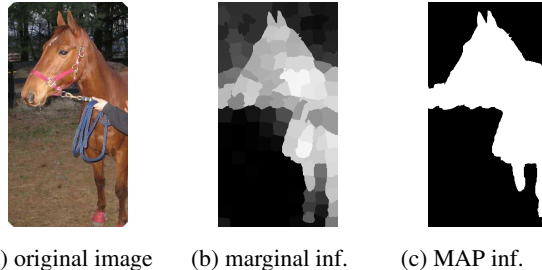(a) original image   (b) marginal inf.   (c) MAP inf.

Figure 1: Example of the marginal and MAP inference for an image from the HorseSeg database Kolesnikov et al. (2014).

Table 3: HorseSeg Dataset. Performance Comparison.

method	"small"	"medium"	"large"
S-SVM+BCFW	12.3	10.9	10.9
perturb&MAP	20.9	21.0	20.9
w.m. perturb&MAP	11.6	10.9	10.9

all superpixels. However, in this dataset we have plenty of images with partial or zero annotation. For these set of images $\mathcal{D}_2 = \{x^l\}_{l=1}^{L}$, we handle approximate marginal probabilities q_d^l associated to the unknown labels. Using them we can approximate the foreground and background volumes: $V_{foreground}^l \approx \sum_{d=1}^{D^l} q_d^l$ and $V_{background}^l \approx \sum_{d=1}^{D^l} (1 - q_d^l)$.

We provide an example of the marginal and MAP inference in Figure 1. The difference of the information compression between these two approaches is visually comparable. We believe that the smoother and accurate marginal approach should have a positive impact on the result, as the uncertainty about the prediction is well propagated.

As an example of the max-margin approaches we take S-SVM+BCFW from the paper of Osokin et al. (2016) which is well adapted to large-scale problems. For S-SVM+BCFW and perturb-and-MAP methods we use MAP-inference for labelling unlabelled data using w_1 (see Section 4).

For the HorseSeg dataset (Table 3), the numbers correspond to the averaged loss function (weighted Hamming loss) values over the hold out test dataset. The results of the experiment in Table 3 demonstrate that the approaches taking into account the weights of the loss $\theta_d(\cdot)$ (S-SVM+BCFW and w.m. perturb&MAP) give a much better accuracy than the regular perturb&MAP. S-SVM+BCFW uses loss-augmented inference and thereby augments the weighted loss structure into the learning phase. Weighted marginal perturb-and-MAP plugs the weights of the weighted Hamming loss inside the objective log-likelihood function. Basic perturb-and-MAP does

Table 4: Reduced HorseSeg Dataset. Performance Comparison.

method	10% of "small"	"medium" with bbox	"medium" w/o bbox
S-SVM+BCFW	17.3	14.0	16.1
perturb&MAP	23.2	23.4	23.0
w.m. p.&MAP	18.1	13.7	14.4

method \ it	100	10^3	10^4	10^5	10^6
basic	0.9	9.2	89.5	900	8993
DC	0.7	6.9	69.0	696	7171
GR	0.3	2.1	15.5	133.4	1186
DC+GR	0.2	1.5	10.9	83.5	727

Table 5: Execution time comparison in seconds. HorseSeg small dataset.

not use the weights $\theta_d(\cdot)$ and loses a lot of accuracy. This shows us that the predefined loss for performance evaluation has a significant influence on the result and should be taken into account.

Small dataset size influence. We now investigate the effect of the reduced "small" dataset. We preserve the setup from the previous section and the only thing that we change is N, the size of the "small" fully-annotated dataset $\mathcal{D}_1 = \{(x^n, y^n)\}_{n=1}^N$. The new "small" dataset is 10% the size of the initial one, i.e., only 14 images. By taking a small labelled dataset, we test the limit of supervised learning when few labels are present.

For the HorseSeg dataset (Table 4), the numbers correspond to the averaged loss function (weighted Hamming loss) values over the hold out test dataset. The results of this experiments are presented in Table 4. In this setup, the probabilistic approach "weighted marginal perturb-and-MAP" gains more than max-margin "S-SVM+BCFW". This could happen because of very limited fully supervised data. The learned parameter w_1 gives a noisy model and this noisy model produces a lot of noisy labels for the unlabeled data, while weighted perturb-and-MAP is more cautious as it uses probabilities that contain more information (see Figure 1).

5.2.1 Acceleration trick impact

We now compare the execution time of the algorithm with and without our acceleration techniques (namely Dynamic Cuts [DC] and Gumbel Reduction[GR]) to get an idea on how helpful they are. Table 5 shows the execution time for calculating all y_B^* (Algorithm 2) for different numbers of iterations on the HorseSeg small dataset. We conclude that the impact of DC does not depend on the total number of iterations always leading to acceleration around 1.3. For GR, acceleration goes from 3.5 for 100 iterations to 7.6 for one million iterations. Overall, we get acceleration of factor around 10 for one million iterations.

5.3 Experiments analysis

The experiments results mainly show that not taking into account the right loss in the learning procedure is detrimental to probabilistic technique such as CRFs, while taking it into account (our novelty) improves results. Also, Tables 2 and 3 show that the proposed methods achieves (and sometimes surpasses) the level performance of the max-margin approach (with loss-augmented inference).

Further, we observed that the size of the training set influences the SSVM and perturb-and-MAP approaches differently. For smaller datasets, the max-margin approaches tend to lose information due to usage of the hard estimates for the unlabelled data (e.g. in Table 4: 16.1 against 14.4 for "medium" dataset without bounding boxes labeling).

Table 4 reports an experiment about using weakly-labeled data at the training stage (the results on the partially annotated "medium" dataset). This experiment studied the impact on the final prediction quality of the training set of "medium" size on top of the reduced "small" fully-labelled set. The results of Table 4 mean that the usage of our approach adopted to the correct test measure outperforms the default perturb-and-MAP by a large margin. Our approach also significantly outperformed the comparable baseline of SSVM due to reduced size of the "small" fully-labelled set.

6 CONCLUSION

In this paper, we have proposed an approximate learning technique for problems with non-trivial losses. We were able to make marginal weighted log-likelihood for perturb-and-MAP tractable. Moreover, we used it for semi-supervised and weakly-supervised learning. Finally, we have successfully demonstrated good performance of the marginal-based and weighted-marginal-based approaches on the middle-scale experiments. As a future direction, we can go beyond the graph cuts and image segmentation application and consider other combinatorial problems with feasible MAP-inference, e.g., matching.

References

F. Bach. Learning with submodular functions: a convex optimization perspective. *Foundations and Trends in Machine Learning*, 6(2-3):145 – 373, 2013.

F. Bach, D. Heckerman, and E. Horvitz. Considering cost asymmetry in learning classifiers. *Journal of Machine Learning Research (JMLR)*, 7:1713–1741, 2006.

G. Bakir, T. Hofmann, B. Schölkopf, A. J. Smola, B. Taskar, and S. V. N. Vishwanathan. *Predicting Structured Data*. The MIT Press, 2007.

C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

E. Borenstein, E. Sharon, and S. Ullman. Combining Top-down and Bottom-up Segmentation. In *Proc. ECCV*, 2004.

E. Boros and P. L. Hammer. Pseudo-boolean optimization. *Discrete Applied Mathematics*, 123(1):155–225, 2002.

Y. Boykov and V. Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 26(9):1124–1137, 2004.

J. Djolonga and A. Krause. From MAP to Marginals: Variational Inference in Bayesian Submodular Models. In *Adv. NIPS*, 2014.

J. Djolonga and A. Krause. Scalable Variational Inference in Log-supermodular Models. In *Proc. ICML*, 2015.

R. B Girshick, P. F Felzenszwalb, and D. A Mcallester. Object detection with grammar models. In *Adv. NIPS*, 2011.

T. Hazan and T. Jaakkola. On the partition function and random maximum a-posteriori perturbations. In *Proc. ICML*, 2012.

T. Hazan, S. Maji, J. Keshet, and T. Jaakkola. Learning efficient random maximum a-posteriori predictors with non-decomposable loss functions. In *Adv. NIPS*, 2013.

M. Jerrum and A. Sinclair. Polynomial-time approximation algorithms for the Ising model. *SIAM Journal on Computing*, 22(5):1087–1116, 1993.

M. I. Jordan, Z. Ghahramani, T. S. Jaakkola, and L. K. Saul. An introduction to variational methods for graphical models. *Machine Learning*, 37(2):183–233, 1999.

S. Kakade, Y. W. Teh, and S. T. Roweis. An alternate objective function for markovian fields. In *Proc. ICML*, 2002.

P. Kohli and P. H. Torr. Dynamic graph cuts for efficient inference in markov random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 29(12):2079–2088, 2007.

A. Kolesnikov, M. Guillaumin, V. Ferrari, and C. H. Lampert. Closed-form training of conditional random fields for large scale image segmentation. In *Proc. ECCV*, 2014.

V. Kolmogorov and R. Zabih. What energy functions can be minimized via graph cuts? *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 26(2):147–159, 2004.

N. Komodakis, N. Paragios, and G. Tziritas. MRF energy minimization and beyond via dual decomposition. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 33(3):531–552, 2011.

M. P. Kumar, B. Packer, and D. Koller. Self-paced learning for latent variable models. In *Adv. NIPS*, 2010.

S. Lacoste-Julien, M. Jaggi, M. Schmidt, and P. Pletscher. Block-coordinate Frank-Wolfe optimization for structural SVMs. In *Proc. ICML*, 2013.

J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. ICML*, 2001.

S. Nowozin and C. H. Lampert. Structured learning and prediction in computer vision. *Foundations and Trends in Computer Graphics and Vision*, 6(3–4):185–365, 2011.

A. Osokin and D. P. Vetrov. Submodular relaxation for inference in Markov random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TRAMI)*, 37(7):1347–1359, 2015.

A. Osokin, J.-B. Alayrac, I. Lukasewitz, P. K. Dokania, and S. Lacoste-Julien. Minding the gaps for block Frank-Wolfe optimization of structured SVMs. In *Proc. ICML*, 2016.

G. Papandreou and A. Yuille. Perturb-and-MAP random fields: Using discrete optimization to learn and sample from energy models. In *Proc. ICCV*, 2011.

S. Parise and M. Welling. Learning in Markov random fields: an empirical study. In *Joint Statistical Meeting (JSM)*, 2005.

P. Pletscher, S. Nowozin, P. Kohli, and C. Rother. Putting MAP back on the map. In *33rd Annual Symposium of the German Association for Pattern Recognition*, 2011.

S. Shalev-Shwartz, Y. Singer, N. Srebro, and A. Cotter. Pegasos: Primal estimated sub-gradient solver for svm. *Mathematical Programming*, 127(1):3–30, 2011.

T. Shpakova and F. Bach. Parameter learning for log-supermodular distributions. In *Adv. NIPS*, 2016.

N. A. Smith. Linguistic structure prediction. *Synthesis Lectures on Human Language Technologies*, 4(2):1–274, 2011.

D. Sontag, T. Meltzer, A. Globerson, Y. Weiss, and T. Jaakkola. Tightening LP relaxations for MAP using message-passing. In *Proc. UAI*, 2008.

C. Sutton and A. McCallum. Piecewise training of undirected models. In *Proc. UAI*, 2005.

C. Sutton and A. McCallum. Piecewise pseudolikelihood for efficient training of conditional random fields. In *Proc. ICML*, 2007.

B. Taskar, C. Guestrin, and D. Koller. Max-margin Markov networks. In *Adv. NIPS*, 2003.

I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun. Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research (JMLR)*, 6:1453–1484, 2005.

S.V.N. Vishwanathan, N. N. Schraudolph, M. W. Schmidt, and K. P. Murphy. Accelerated training of conditional random fields with stochastic gradient methods. In *Proc. ICML*, 2006.

A. J. Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, IT-13(2):260–269, 1967.

M. J. Wainwright and M. I. Jordan. Graphical models, exponential families, and variational inference. *Foundations and Trends in Machine Learning*, 1(1-2):1–305, 2008.

Y. Weiss. Comparing the mean field method and belief propagation for approximate inference in MRFs. *Advanced Mean Field Methods: Theory and Practice*, pages 229–240, 2001.

R. J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8:229–256, 1992.

C.-N. J. Yu and T. Joachims. Learning structural svms with latent variables. In *Proc. ICML*, 2009.

A. L. Yuille and A. Rangarajan. The concave-convex procedure. In *Adv. NIPS*, 2003.