Graph-Based Posterior Regularization for Semi-Supervised Structured Prediction

Luheng He **Jennifer Gillenwater**
Computer and Information Science
University of Pennsylvania
{luhe,jengi}@cis.upenn.edu

Ben Taskar
Computer Science and Engineering
University of Washington
taskar@cs.washington.edu

Abstract

We present a flexible formulation of semi-supervised learning for structured models, which seamlessly incorporates graph-based and more general supervision by extending the posterior regularization (PR) framework. Our extension allows for any regularizer that is a convex, differentiable function of the appropriate marginals. We show that surprisingly, non-linearity of such regularization does not increase the complexity of learning, provided we use multiplicative updates of the structured exponentiated gradient algorithm. We illustrate the extended framework by learning conditional random fields (CRFs) with quadratic penalties arising from a graph Laplacian. On sequential prediction tasks of handwriting recognition and part-of-speech (POS) tagging, our method makes significant gains over strong baselines.

1 Introduction

Recent success of graph-based semi-supervised learning builds on access to plentiful unsupervised data and accurate similarity measures between data examples (Zhu et al., 2003; Joachims, 2003; Belkin et al., 2005; Zhu and Lafferty, 2005; Altun et al., 2005; Zhu, 2005; Chapelle et al., 2006; Subramanya and Bilmes, 2009; Subramanya et al., 2010; Das and Petrov, 2011). Many approaches, such as Joachims (2003) and Subramanya and Bilmes (2009) use graph-based learning in the transductive setting, where unlabeled examples are classified without learning a parametric predictive model. While predicted labels can then be leveraged to learn such a model (e.g. a CRF), this pipelined approach misses out on the benefits of modeling sequential correlations *during* graph propagation. In this work we seek to better integrate graph propagation with estimation of a structured, parametric predictive model.

To do so, we build on the posterior regularization (PR) framework of Ganchev et al. (2010). PR is a principled means of providing weak supervision during structured model estimation. More concretely, PR introduces a penalty whenever the model's posteriors over latent variables contradict the specified weak supervision. Ganchev et al. (2010) show how to efficiently optimize a likelihood-plus-posterior-penalty type objective in the case where the penalty is linear in the model's marginals. Yet, there are many forms of supervision that cannot be expressed as a linear function of marginals. For example, graph Laplacian regularization. In this work, we extend PR to allow for penalties expressed as any convex, differentiable function of the marginals and derive an efficient optimization method for such penalties.

In our experiments, we explore graph Laplacian posterior regularizers for two applications: handwriting recognition and POS tagging. The methods of Altun et al. (2005), Subramanya et al. (2010), and Das and Petrov (2011) are the most closely related to this work. Altun et al. (2005) describes coupling a graph regularizer with a max-margin objective for pitch accent prediction and handwriting recognition tasks. Their method suffers from scalability issues though; it relies on optimization in the dual, which requires inversion of a matrix whose dimension grows with graph size.

The more recent work of Subramanya et al. (2010) tackles the POS tagging task and provides a more scalable method. Their method is a multi-step procedure that iterates two main steps, graph propagation and likelihood optimization, until convergence. Actually computing the optimum for the graph propagation step would require a matrix inversion similar to that used by Altun et al. (2005), but they skirt this issue by using an heuristic update rule. Unfortunately though, no

guarantees for the quality of this update are established. Das and Petrov (2011) proceed very similarly, adapting the iterative procedure to include supervision from bi-text data, but applying the same heuristic update rule.

The work we present here similarly avoids the complexity of a large matrix inversion and iterates steps related to graph propagation and likelihood optimization. But in contrast to Subramanya et al. (2010) and Das and Petrov (2011) it comes with guarantees for the optimality of each step and convergence of the overall procedure. Further, our approach is based on optimizing a joint objective, which affords easier analysis and extensions using other constraints or optimization methods. The key enabling insight is a surprising factorization of the non-linear regularizer, which can be exploited using multiplicative updates.

2 Posterior regularization

We focus on the semi-supervised setting, showing how to extend the discriminative, penalty-based version of PR for a linear chain CRF. Our results apply more generally though to the unsupervised setting, the constraint-based versions of PR, and other graphical models.

In the standard semi-supervised setting we are given n data instances, $\{\boldsymbol{x}^1, \ldots, \boldsymbol{x}^n\}$, and labels $\{\boldsymbol{y}^1, \ldots, \boldsymbol{y}^l\}$ for the first $l \ll n$ instances. For simplicity of notation, we'll assume each $\boldsymbol{x}^i$ has T components. Modeling this data with a linear chain CRF, the standard conditional log-likelihood objective with a Gaussian prior (variance $\propto \sigma^2$) is:

$$\mathcal{L}(\theta) = \sum_{i=1}^{l} \log p_\theta(\boldsymbol{y}^i \mid \boldsymbol{x}^i) - \frac{||\boldsymbol{\theta}||_2^2}{2\sigma^2} \,. \quad (1)$$

Note that this discriminative objective does not attempt to leverage the unlabeled data. Since p_θ decomposes according to the independence assumptions of a linear chain CRF, it can be expressed as:

$$p_\theta(\boldsymbol{y} \mid \boldsymbol{x}) = \frac{\exp\left[\sum_{t=1}^{T} \boldsymbol{\theta}^\top \boldsymbol{f}(y_t, y_{t-1}, \boldsymbol{x})\right]}{Z_p(\boldsymbol{x})} \quad (2)$$

where the $Z_p(\boldsymbol{x})$ is a normalizer:

$$Z_p(\boldsymbol{x}) = \sum_{\boldsymbol{y}'} \exp\left[\sum_{t=1}^{T} \boldsymbol{\theta}^\top \boldsymbol{f}(y'_t, y'_{t-1}, \boldsymbol{x})\right] \quad (3)$$

and the $\boldsymbol{f}$ are arbitrary feature functions. We assume $\boldsymbol{f}(y_1, y_0, \boldsymbol{x})$ receives a special "start" marker for y_0. In what follows, we refer to functions over the $(y_t, y_{t-1}, \boldsymbol{x})$ as local factors, or *p-factors*; $p_\theta(\boldsymbol{y} \mid \boldsymbol{x})$ decomposes as a product of p-factors.

Given this decomposition, $\mathcal{L}$ and its gradient with respect to θ can be efficiently computed using the forward-backward algorithm for linear chains. This amounts to computing posterior marginals for each p-factor $(y_t, y_{t-1}, \boldsymbol{x})$. Following the gradient suffices to find the global optimum of $\mathcal{L}$, since likelihood is concave, and the Gaussian prior makes it strictly concave.

Penalty-based posterior regularization (PR) modifies the likelihood objective by adding a "penalty" term expressing prior knowledge about the posteriors (Ganchev et al., 2010). To allow for more efficient optimization, penalty terms are imposed on an auxiliary joint distribution q over the labels instead of directly on p_θ. Agreement between q and p_θ is encouraged by a KL term:

$$\mathbf{KL}(q \parallel p_\theta) = \sum_{i=1}^{n} \mathbf{KL}(q(\boldsymbol{Y} \mid \boldsymbol{x}^i) \parallel p_\theta(\boldsymbol{Y} \mid \boldsymbol{x}^i))$$

where $\boldsymbol{Y}$ is a random variable that can take on any possible labeling $\boldsymbol{y}$, and $q(\boldsymbol{Y}|\boldsymbol{x}^i)$ is a an arbitrary distribution over $\boldsymbol{Y}$ for each i[1]. The penalty term itself is restricted to be an essentially linear function of the p-factor marginals of $q(\boldsymbol{Y} \mid \boldsymbol{x}^i)$. To compactly express this, we first define some notation. Let $\boldsymbol{m}^i$ denote the p-factor marginals of $q(\boldsymbol{Y} \mid \boldsymbol{x}^i)$. For first-order linear chain models, if K is the total number of labels a y variable can take on, then $\boldsymbol{m}^i$ contains the marginals for $t \in \{1, \ldots, T\}$ and all K^2 possible (y_t, y_{t-1}) label pairs. That is, $\boldsymbol{m}^i$ is a length $O(TK^2)$ vector with entries:

$$m^i_{t,k,j} = \sum_{\boldsymbol{y}} \mathbb{1}(y_t = k,\ y_{t-1} = j) q(\boldsymbol{y} \mid \boldsymbol{x}^i) \,. \quad (4)$$

Stacking all these $\boldsymbol{m}^i$, we let $\boldsymbol{m}$ represent the $O(nTK^2)$ vector $[\boldsymbol{m}^1, \ldots, \boldsymbol{m}^n]$. We further define a matrix A of constraint features. The product $A\boldsymbol{m}$ is then the expectation of these features under q. Finally we have, with a vector $\boldsymbol{b}$ of limits, the following expression for the penalty term:

$$h_{lin}(\boldsymbol{m}) = ||\max(A\boldsymbol{m} - \boldsymbol{b}, \boldsymbol{0})||_\beta \quad (5)$$

where $||\cdot||_\beta$ denotes an arbitrary norm. This expression will be non-zero if the expected value of

[1] We use a notation that is slightly different than, but equivalent to, that of prior work, in order to facilitate our extensions later.

$A\boldsymbol{m}$ is larger than the limit $\boldsymbol{b}$. The full posterior regularizer is then:

$$\mathcal{R}(\theta, q) = \mathbf{KL}(q \parallel p_\theta) + \lambda h_{lin}(\boldsymbol{m}) \ , \quad (6)$$

where λ is a hyperparameter that controls the strength of the second term.

Running example: Consider the task of part-of-speech (POS) tagging, where the $\boldsymbol{y}$ are tags and the $\boldsymbol{x}$ are words. To encourage every sentence to contain at least one verb, we can penalize if the expected number of verbs under the q distribution is less than 1. Specifically, if "verb" is represented by tag number v, for sentence i we penalize unless:

$$1 \leq \sum_{t=1}^{T} \sum_{y_{t-1}=1}^{K} m^i_{t,v,y_{t-1}} \ . \quad (7)$$

In the notation of Equation (5), these penalties correspond to: an n-row A matrix, where row i has -1's to select exactly the portion of $\boldsymbol{m}$ from Equation (7), and a limit $\boldsymbol{b} = -\mathbf{1}$.

We briefly note here that generalized expectation (Mann and McCallum, 2007; Mann and McCallum, 2008) can be used to impose similar penalties, but without the auxiliary q distribution. Unfortunately though, this means the expectation of the A features is with respect to p_θ, so computing the gradient requires the covariance between the constraint features in A and the model features $\boldsymbol{f}$, under θ. For a linear chain CRF, this means the run time of forward-backward is squared, although some optimizations are possible. PR's use of the auxiliary q allows us to optimize more efficiently by splitting the problem into easier blocks.

The new objective that combines likelihood with the PR penalty is: $\mathcal{J}(\theta, q) = \mathcal{L}(\theta) - \mathcal{R}(\theta, q)$. While optimizing $\mathcal{L}(\theta)$ is easy, finding $\max_{\theta,q} \mathcal{J}(\theta, q)$ is NP-hard even for the simplest models. To optimize $\mathcal{J}$, Ganchev et al. (2010) employ an expectation maximization (EM) based method. At iteration $t+1$, the algorithm updates q and θ as follows:

$$\mathbf{E} : q^{t+1} = \arg\min_q \ \mathcal{R}(\theta^t, q) \quad (8)$$

$$\mathbf{M} : \theta^{t+1} = \arg\max_\theta \ \mathcal{L}(\theta) + \quad (9)$$
$$\delta \sum_{i=l+1}^{n} \sum_{\boldsymbol{y}} q^{t+1}(\boldsymbol{y} \mid \boldsymbol{x}^i) \log p_\theta(\boldsymbol{y} \mid \boldsymbol{x}^i)$$

where δ here is a hyperparameter that trades off between the labeled and unlabeled data. Though not stated above, note that in the E-step minimization over $q(\boldsymbol{Y} \mid \boldsymbol{x}^i)$ is constrained to the probability simplex. Ganchev et al. (2010) show that this E-step can be efficiently implemented, via projected gradient descent on the dual. The M-step is similar to optimizing the original $\mathcal{L}$, but with a contribution from the unlabeled data that further encourages q and p_θ to agree. Thus, the M-step can be implemented via the same gradient ascent methods as used for $\mathcal{L}$. As with standard EM, this method monotonically increases $\mathcal{J}$ and thus is guaranteed to converge to a local optimum.

In this work, we contemplate what other types of posterior penalty terms besides $h_{lin}(\boldsymbol{m})$ are possible. In the subsequent section, we show that it is possible to extend the class of efficiently-optimizable PR penalties to encompass all convex, differentiable functions of the marginals.

3 Non-linear PR

Let $h(\boldsymbol{m})$ denote an arbitrary convex, differentiable function of the marginals of q. Replacing $\mathcal{R}$'s penalty term with h, we have:

$$\tilde{R}(\theta, q) = \mathbf{KL}(q \parallel p_\theta) + \lambda h(\boldsymbol{m}) \quad (10)$$

Let $\tilde{\mathcal{J}}$ represent the full objective with $\tilde{R}$. We show that $\tilde{\mathcal{J}}$ can be efficiently optimized.

Running example: Returning to our POS tagging example, let's consider one type of non-linear convex penalty that might be useful. Suppose our corpus has N unique trigrams, and we construct a graph $G = (V, E, W)$ where each vertex in V is a trigram and each edge $(a, b) \in E$ has a weight w_{ab} that indicates the similarity of trigrams a and b. To use the information from this graph to inform our CRF, we can use the graph Laplacian: $L = D - W$, where D is a diagonal degree matrix with $d_{aa} = \sum_{j=1}^{N} w_{aj}$. The form of L is such that for every vector $\boldsymbol{v} \in \mathbb{R}^N$:

$$\boldsymbol{v}^\top L \boldsymbol{v} = \frac{1}{2} \sum_{a=1}^{N} \sum_{b=1}^{N} w_{ab} (v_a - v_b)^2 \ . \quad (11)$$

The larger the disparity in $\boldsymbol{v}$ values of similar vertices, the larger the value of $\boldsymbol{v}^\top L \boldsymbol{v}$. The matrix L is positive semi-definite, so $\boldsymbol{v}^\top L \boldsymbol{v}$ is

convex in $\boldsymbol{v}$. If each entry v_a is a linear function of the vector of marginals $\boldsymbol{m}$ described above, then $\boldsymbol{v}(\boldsymbol{m})^\top L \boldsymbol{v}(\boldsymbol{m})$ is convex in $\boldsymbol{m}$. Thus, for any linear $\boldsymbol{v}(\boldsymbol{m})$, we can use this Laplacian expression as a PR penalty.

For example, we can define $\boldsymbol{v}(\boldsymbol{m})$ such that $h(\boldsymbol{m})$ applies a penalty if trigrams that are similar according to the graph have different expected taggings under the CRF model. To state this more formally, let's define a mapping $B : (\{1, \ldots, n\}, \{1, \ldots, T\}) \mapsto V$ from words in the corpus to vertices in the graph: $B(i, t) = a$ implies word x_t^i maps to vertex a. Then, for a given tag k, we have the following formula for the value of vertex a:

$$v_{a,k} = \tilde{m}_{a,k} = \frac{\sum_{i=1}^{n} \sum_{\substack{t=1 \\ B(i,t)=a}}^{T} \sum_{y_{t-1}=1}^{K} m^i_{t,k,y_{t-1}}}{\sum_{i=1}^{n} \sum_{t=1}^{T} \mathbb{1}(B(i,t) = a)}$$

There are several issues to overcome in showing that EM with these more general $h(\boldsymbol{m})$ can still be run efficiently and will still reach a local optimum. First, we have to show that the optimal q for the E-step minimization can still be compactly representable as a product of p-factors.

3.1 Decomposition

Theorem 1. *If $h(\boldsymbol{m})$ is a convex, differentiable function of q's p-factor marginals, $q^* = \arg\min_q \tilde{\mathcal{R}}(\theta, q)$ decomposes as a product of p-factors.*

Proof. Consider the E-step gradient of $\tilde{R}(\theta, q)$ with respect to q. Using the shorthand $q^i_{\boldsymbol{y}}$ for $q(\boldsymbol{y} \mid \boldsymbol{x}^i)$, the gradient is:

$$\frac{\partial \tilde{\mathcal{R}}}{\partial q^i_{\boldsymbol{y}}} = \log q^i_{\boldsymbol{y}} + 1 - \log p_\theta(\boldsymbol{y} \mid \boldsymbol{x}^i) + \lambda \frac{\partial h(\boldsymbol{m})}{\partial \boldsymbol{m}}^\top \frac{\partial \boldsymbol{m}}{\partial q^i_{\boldsymbol{y}}} . \quad (12)$$

Here, $\frac{\partial \boldsymbol{m}}{\partial q^i_{\boldsymbol{y}}}$ is just a 0-1 vector indicating which of the marginals from $\boldsymbol{m}$ apply to $q^i_{\boldsymbol{y}}$. For example, for $y_t = k$ and $y_{t-1} = j$, the marginal $m^i_{t,k,j}$ is relevant. We can more simply write:

$$\frac{\partial h(\boldsymbol{m})}{\partial \boldsymbol{m}}^\top \frac{\partial \boldsymbol{m}}{\partial q^i_{\boldsymbol{y}}} = \sum_{t=1}^{T} \frac{\partial h(\boldsymbol{m})}{\partial m^i_{t,y_t,y_{t-1}}} . \quad (13)$$

Setting the gradient equal to zero and solving for $q^i_{\boldsymbol{y}}$, we see that it must take the following form:

$$q^i_{\boldsymbol{y}} = \frac{p_\theta(\boldsymbol{y} \mid \boldsymbol{x}^i) \exp\left[-\lambda \sum_{t=1}^{T} \frac{\partial h(\boldsymbol{m})}{\partial m^i_{t,y_t,y_{t-1}}}\right]}{Z_q(\boldsymbol{x}^i)} . \quad (14)$$

From this expression, it is clear that $q^i_{\boldsymbol{y}}$ is proportional to a product of p-factors. □

Running example: Recall the graph Laplacian penalty, discussed above for a particular tag k. Summing over all tags, the penalty is:

$$h(\boldsymbol{m}) = \frac{1}{2} \sum_{k=1}^{K} \sum_{a=1}^{N} \sum_{b=1}^{N} w_{ab} (\tilde{m}_{a,k} - \tilde{m}_{b,k})^2 .$$

The derivative $\frac{\partial h(\boldsymbol{m})}{\partial m^i_{t,y_t,y_{t-1}}}$ is then:

$$2 \sum_{k=1}^{K} \sum_{a=1}^{N} w_{a,B(i,t)} (\tilde{m}_{B(i,t),k} - \tilde{m}_{a,k}) . \quad (15)$$

In words: for a given k, this gradient is positive if node $B(i, t)$ has larger probability of taking tag k than its close neighbors. Moving in the direction opposite the gradient encourages similar taggings for similar trigrams.

Theorem 1 confirms that the optimal q will decompose as desired, but does not address whether we can efficiently find this q. Previous PR work optimized the E-step in the dual. But while the dual is easy to compute in closed form for norms or linear functions, for arbitrary convex functions the dual is often non-trivial.

Running example: For the case of a graph Laplacian regularizer, in the primal the penalty takes the form of a quadratic program: $\boldsymbol{v}^\top L \boldsymbol{v}$. Unfortunately, the dual of a quadratic program contains a matrix inverse, L^{-1} (van de Panne and Whinston, 1964). Taking a matrix inverse is expensive, which makes optimization in the dual unattractive.

Since moving to the dual would be inefficient, optimizing $\tilde{R}$ will require some form of gradient descent on the $q^i_{\boldsymbol{y}}$. However, the standard gradient descent update:

$$q^i_{\boldsymbol{y}} \leftarrow q^i_{\boldsymbol{y}} - \eta \frac{\partial \tilde{\mathcal{R}}}{\partial q^i_{\boldsymbol{y}}} \quad (16)$$

where η is the step size, does not result in a feasible optimization scheme, for several reasons. First, it is possible for the updated q to be outside the probability simplex. To be sure it remains in the simplex would require a projection step on the full, exponential-size set of all $q^i_{\boldsymbol{y}}$, for each example $\boldsymbol{x}^i$. Second, the updated q may not be proportional to a product of p-factors. To be concrete, suppose the starting point is $q^i_{\boldsymbol{y}} = p_\theta(\boldsymbol{y} \mid \boldsymbol{x}^i)$, which does decompose as a product of p-factors. Then after the first gradient update, we have:

$$q^i_{\boldsymbol{y}} = p_\theta(\boldsymbol{y} \mid \boldsymbol{x}^i) - \eta \left(1 + \lambda \sum_{t=1}^{T} \frac{\partial h(\boldsymbol{m})}{\partial m^i_{t,y_t,y_{t-1}}}\right) .$$

Unfortunately, while $p_\theta(\boldsymbol{y} \mid \boldsymbol{x}^i)$ decomposes as a product of p-factors, the other term decomposes as a sum. Naturally, as we discuss in the following section, multiplicative updates are more suitable.

3.2 Exponentiated Gradient

The exponentiated gradient descent (EGD) algorithm was proposed by Kivinen and Warmuth (1995), who illustrate its application to linear prediction. More recently, Collins et al. (2005) and Collins et al. (2008) extended EGD to exploit factorization in structured models. The most important aspect of EGD for us is that a variable's update formula takes a multiplicative rather than an additive form. Specifically, the update for $q^i_{\boldsymbol{y}}$ is:

$$q^i_{\boldsymbol{y}} \leftarrow q^i_{\boldsymbol{y}} \exp\left[-\eta \frac{\partial \tilde{\mathcal{R}}}{\partial q^i_{\boldsymbol{y}}}\right] . \tag{17}$$

Lemma 2. *EGD update Equation (17) preserves decomposition of q into p-factors.*

Proof. Applying the multiplicative EGD update formula to $q^i_{\boldsymbol{y}}$, we see that its new value equals the following product:

$$(q^i_{\boldsymbol{y}})^{1-\eta} p_\theta(\boldsymbol{y} \mid \boldsymbol{x}^i)^\eta \exp\left[-\eta\lambda \sum_{t=1}^{T} \frac{\partial h(\boldsymbol{m})}{\partial m^i_{t,y_t,y_{t-1}}}\right] ,$$

up to a normalization constant. Since $q^i_{\boldsymbol{y}}$ and $p_\theta(\boldsymbol{y} \mid \boldsymbol{x}^i)$ both decompose as a product of p-factors and since the update term is another product of p-factors, the updated expression is itself a product of p-factors (up to normalization). □

Note that normalization is not an issue with the EGD updates. Since q retains its decomposition, the normalization can be efficiently computed using forward-backward. Thus, Lemma 2 moves us much closer to the goal of running EM efficiently, though there remain several stumbling blocks. First and foremost, we cannot afford to actually apply EGD to each $q^i_{\boldsymbol{y}}$, as there are an exponential number of them. Thankfully, we can show these EGD updates are equivalent to following the gradient on a much smaller set of values. In particular, letting F represent the dimension of $\boldsymbol{m}$, which for example is $O(nTK^2)$ for linear chains, we have the following result.

Lemma 3. *Given the gradient vector $\frac{\partial h(\boldsymbol{m})}{\partial \boldsymbol{m}}$, one step of EGD on $\tilde{\mathcal{R}}(\theta, q)$ can be completed in time $O(F)$, where F is the dimension of $\boldsymbol{m}$.*

Proof. First, we re-express $q^i_{\boldsymbol{y}}$ in log-linear form. Applying Lemma 2, we know that $q^i_{\boldsymbol{y}}$ is proportional to a product of p-factors This means that there must exist some factors r such that $q^i_{\boldsymbol{y}}$ can be written:

$$q^i_{\boldsymbol{y}} = \frac{1}{Z_q(\boldsymbol{x}^i)} \exp\left[\sum_{t=1}^{T} r_{i,t}(y_t, y_{t-1})\right] . \tag{18}$$

Re-expressing $\frac{\partial \tilde{\mathcal{R}}}{\partial q^i_{\boldsymbol{y}}}$ given these r, we have:

$$\frac{\partial \tilde{\mathcal{R}}}{\partial q^i_{\boldsymbol{y}}} = C + \sum_{t=1}^{T} \Bigg[r_{i,t}(y_t, y_{t-1}) - \boldsymbol{\theta}^\top \boldsymbol{f}(y_t, y_{t-1}, \boldsymbol{x}^i) + \lambda \frac{\partial h(\boldsymbol{m})}{\partial m^i_{t,y_t,y_{t-1}}} \Bigg] , \tag{19}$$

where $C = 1 - \log Z_q(\boldsymbol{x}^i) + \log Z_p(\boldsymbol{x}^i)$ is constant with respect to $\boldsymbol{y}$. This means that we can just update the individual r factors as follows:

$$r_{i,t}(y_t, y_{t-1}) \leftarrow (1-\eta) r_{i,t}(y_t, y_{t-1}) + \eta \boldsymbol{\theta}^\top \boldsymbol{f}(y_t, y_{t-1}, \boldsymbol{x}^i) - \eta\lambda \frac{\partial h(\boldsymbol{m})}{\partial m^i_{t,y_t,y_{t-1}}} . \tag{20}$$

Note that if we start from $q^i_{\boldsymbol{y}} = p_\theta(\boldsymbol{y} \mid x^i)$, then the initial $r_{i,t}(y_t, y_{t-1})$ are just $\boldsymbol{\theta}^\top \boldsymbol{f}(y_t, y_{t-1}, \boldsymbol{x}^i)$. To conclude, since the number of r functions is equal to the dimension of $\boldsymbol{m}$, the overall update is linear in the number of marginals. □

At this point, just one small issue remains: how expensive is computing $\frac{\partial h(\boldsymbol{m})}{\partial \boldsymbol{m}}$? Work analyzing the reverse mode of automatic differentiation indicates that if computing a function h requires c operations, then computing its gradient vector requires no more than $O(c)$ operations (Griewank, 1988). Thus, as long as our penalty function is

itself efficiently computable, the gradient vector will be too. We conclude by observing that our efficient algorithm converges to a local optimum.

Theorem 4. *The above EGD-based EM algorithm for optimizing $\tilde{\mathcal{J}}(\theta, q)$ converges to a local optimum of this objective.*

Proof. The M-step remains unchanged from standard PR EM, and as such is strictly convex in θ. The E-step is strictly convex in q, since KL-divergence is strictly convex and $h(\boldsymbol{m})$ is convex. Applying EGD, we know that we can efficiently find the E-step optimum. Therefore, the EGD-based EM algorithm efficiently implements coordinate ascent on $\tilde{\mathcal{J}}(\theta, q)$, with each step monotonically increasing $\tilde{\mathcal{J}}$:

$$\tilde{\mathcal{J}}(\theta^t, q^t) \leq \tilde{\mathcal{J}}(\theta^t, q^{t+1}) \leq \tilde{\mathcal{J}}(\theta^{t+1}, q^{t+1}) \ .$$

□

Hence, we have shown that it is possible to efficiently use an arbitrary convex, differentiable function of the marginals, $h(\boldsymbol{m})$, as a PR penalty function. In the following section, we apply one such function — the graph Laplacian quadratic from the running example — to several tasks.

4 Experiments

We evaluate the effect of a graph Laplacian PR penalty on two different sequence prediction tasks: part-of-speech (POS) tagging and handwriting recognition. Our experiments are conducted in a semi-supervised setting, where only a small number, l, of labeled sequences are available during training. Both the l labeled sequences and the remainder of the dataset (instances $l+1$ through n) are used to construct a graph Laplacian[2]. We train a second-order CRF using the methods described in Section 3 and report results for a test set consisting of instances $l+1$ through n.

4.1 Graph construction

For each task we define a symmetric similarity function on the task's vertices V, $\text{sim} : V \times V \mapsto \mathbb{R}$, and build the graph based on its values. Specifically, denoting the k nearest neighbors (NN) of node u by $\mathcal{N}_k(u)$, we use the following mutual k-NN criterion to decide which edges to include:

$$(u, v) \in E \iff u \in \mathcal{N}_k(v) \ \wedge \ v \in \mathcal{N}_k(u) \ .$$

Entries in the final edge weight matrix are: $w_{uv} = \mathbb{1}[(u, v) \in E]\text{sim}(u, v)$.

4.2 Part-of-speech tagging

We experiment on ten languages. Our English (EN) data is from the Penn Treebank (Marcus et al., 1993), Italian (IT) and Greek (EL) are from CoNLL-2007 (Nivre et al., 2007), and the remaining languages in Figure 1 (a): German (DE), Spanish (ES), Portuguese (PT), Danish (DA), Slovene (SL), Swedish (SV), and Dutch (NL) are from CoNLL-X (Buchholz and Marsi, 2006). We use a universal tag set (Das et al., 2012) throughout.

For each language, we first construct a mutual 60-NN graph[3] on trigram types, excluding trigrams whose center word is punctuation. Our smallest graph (Slovene) contains 25,198 nodes while the largest (English) has 611,730.

For the similarity function $\text{sim}(u, v)$, we follow the method used in (Subramanya et al., 2010) and (Das and Petrov, 2011), but with a somewhat modified feature set. For instance, while (Subramanya et al., 2010) uses suffixes of the trigram's center word, we find this type of feature is too easy for unrelated trigrams to match, leading to a noisy graph. Let a trigram and its left/right context be denoted by the 5-gram $(w_0, w_1, w_2, w_3, w_4)$. Then the features we use to build the graph are:

- Trigram features: w_{12}, w_{13}, w_{23}, w_2, $\text{suffix}(w_3)w_2$, $\text{suffix}(w_1)w_2$
- Context features: w_{0134}, w_{012}, w_{023}, w_{024}, w_{124}, w_{234}, w_{01}, w_{02}, w_{24}, w_{34}

where suffix indicates common suffixes collected from Wiktionary data. For a given feature f and trigram type t, the value of the feature is determined by pointwise mutual information (PMI): $\log \frac{\#(f \wedge t)}{\#(f)\#(t)}$. Then, for each pair of trigram types, $\text{sim}(u, v)$ is given by the cosine similarity of the trigrams' feature vectors.

For the second-order CRF, we use a fairly standard set of features:

- Emission features: $\mathbb{1}(y_t = k \ \wedge \ f(x_{t'}))$, where k can be any POS tag and $t' \in \{t, t-1, t+1\}$. The $f(x_{t'})$ takes the form of a function from the following set: one indicator for each word, lowercased word, and suf-

[2] While these particular experiments are transductive, our method can easily be applied inductively as well.

[3] In preliminary experiments we tested graphs with 20, 40, 60, 80, and 100 NNs and found that beyond 60 NNs additional performance gains are small.

fix, and also is-capitalized, is-punctuation, is-digit, contains-hyphen, and contains-period.

- Transition features: For any POS tags k_1, k_2, k_3, we have a feature $\mathbb{1}(y_t = k_1, y_{t-1} = k_2, y_{t+1} = k_3)$ and its backoffs (indicators for one or two matching tags).

4.3 Handwriting recognition

The handwriting dataset we use was collected by Kassel (1995) and filtered to 6,877 words (Taskar et al., 2003). For each word, the first letter is removed so that every remaining letter is one of the English language's 26 lowercase letters.

Again, we first build a mutual NN graph. In this case, we use 20-NN, since our graph has fewer nodes and a larger set of possible node identities (26 letters instead of 12 tags). Each node in this graph is one letter from the dataset, for a total of 52,152 nodes. As a first step, we compute cosine similarity on the pixels of each pair of nodes, and then consider only pairs with a similarity greater than 0.3. Next, we apply the Fast Earth Mover's distance $\widehat{\text{EMD}}(u, v)$ (Pele and Werman, 2009) with default parameters to compute the dissimilarity of each pair of images. We convert these into similarities via:

$$s(u, v) = \exp\left\{\frac{-\widehat{\text{EMD}}(u, v)}{\sigma^2_{EMD}}\right\} \qquad (21)$$

where we set the variance $\sigma_{EMD} = 10$. The final similarity function $\text{sim}(u, v)$ is the weighted combination of the similarity of the nodes (u, v) and their left neighbors (u_l, v_l) and right neighbors (u_r, v_r) from their respective words:

$$\text{sim}(u, v) = \alpha s(u, v) + (1-\alpha)(s(u_l, v_l) + s(u_r, v_r))$$

where we fix $\alpha = 0.8$.

For the second-order CRF, the transition features are same as for POS tagging, but with tags replaced by the English alphabet. The emission features take a similar form, but with different meanings for the $f(x_{t'})$ indicator functions. Specifically, there is one indicator for each pixel location, with value 1 if the pixel is turned on. As there are many more emission than transition features, we count the number of fired emission and transition features, say f_e and f_t, then discount all emission features, multiplying them by $\frac{f_t}{f_e}$ to balance the amount of supervision.

4.4 Baselines

We compare our posterior regularization (PR) results with three baselines. We also include results for the first EM iteration of our PR method (PR_1), to show there is still significant optimization occurring after the first iteration.

The first baseline is graph propagation (GP). Specifically, we start from uniform posteriors for all the unlabeled nodes in the graph, then for each tag/letter k and each node v we apply the gradient update:

$$q_{k,v} \leftarrow q_{k,v} - \eta \sum_{u \in \mathcal{N}_k(v)} w^k_{uv}(q_{k,v} - q_{k,u}) \qquad (22)$$

until convergence. We then select the tag/letter with the largest probability as the prediction for a node. If multiple tokens are mapped to a node, then all receive the same prediction.

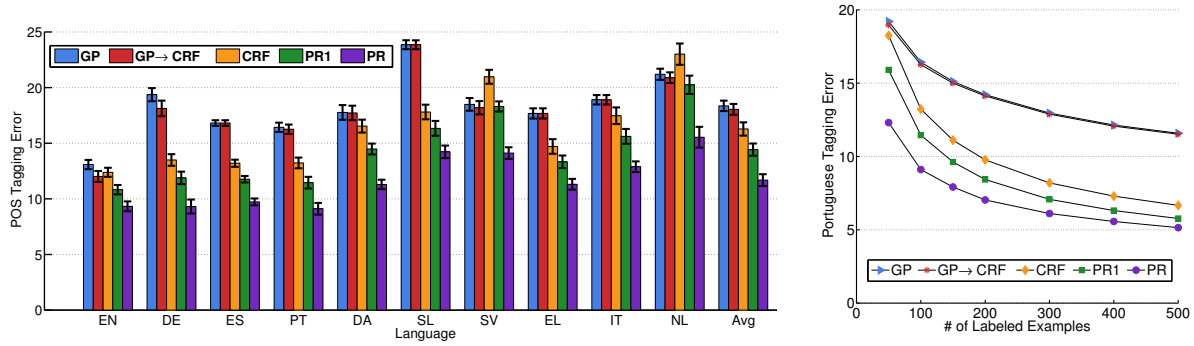
The second baseline incorporates both graph propagation and sequence information. As a first step, we run the GP baseline, then use the decoding as additional labeled data to train a second-order CRF (see GP→CRF results). The third baseline is simply a second-order CRF, trained on the l labeled examples.

4.5 Training details

For optimizing the CRF, we use L-BFGS (Bertsekas, 2004) and a Gaussian prior with $\sigma = 100$ (chosen by cross-validation on the labeled training examples). The final predictions are obtained via posterior decoding. For PR, we run EM for at most 20 iterations, which is enough for convergence of the combined objective $\tilde{\mathcal{J}}(\theta, q)$. We cross-validate the constraint strength parameter λ over the following values: $\{0.1, 0.5, 1.0, 2.0\}$, ultimately selecting $\lambda = 1$ for the POS tagging task and $\lambda = 0.1$ for the handwriting recognition task.

4.6 Results and analysis

POS tagging. For each language, we randomly sample 1000 labeled examples and split them into 10 non-overlapping training sets of size $l = 100$. Figure 1 (a) shows the average error and its standard deviation for these training sets. If for each language we take the difference between the average error of PR and that of the best of the three baselines, the min, average, and max improvements are: 2.69%, 4.06%, and 5.35%. When analyzing the results, we observed that one region where PR makes substantial gains over the



(a) (b)

Figure 1: (a): POS results for 10 languages. Each bar in each group corresponds to the average POS tagging error of one method; the left-to-right order of the methods is the same as in the legend. Whiskers indicate standard deviations. The final set of bars is an average across all languages. See supplement for a table with the exact numbers. (b): POS results on one language for a range of l.

CRF baseline is on unseen words (words that do not occur in the set of l labeled examples). If we measure performance only on such words, the gain of PR over CRF is 6.7%. We also test with $l = \{50, 100, 150, 200, 300, 400, 500\}$ on one language to illustrate how PR performs with different amounts of supervision. Figure 1 (b) shows that even when $l = 500$ our PR method is still able to provide improvement over the best baseline.

Handwriting recognition. For this task, the overall dataset contains 55 distinct word types. Thus, we set $l = 110$ and sample 10 training sets such that each contains 2 examples of each of word. Note that due to the well-balanced training sets, baselines are fairly high here compared to other similar work with this dataset. Table 1 shows there is an average improvement of 4.93% over the best of the three baselines.

	GP	GP→CRF	CRF	PR_1	PR
Mean	17.57	15.07	9.82	6.03	**4.89**
StdDev	0.30	0.35	0.48	0.20	0.42

Table 1: Handwriting recognition errors.

Even in a simpler setting closer to that of POS tagging, where we just draw $l = 100$ samples randomly, there are many cases where PR beats the baselines. Figure 2 shows predictions from such a setting and provides general intuition as to why PR does well on handwriting recognition. For the word 'Wobble' (with the first letter removed), the CRF predicts 'obble' as 'ovely', because of it relies heavily on sequential information; in our small training set, bigrams 'ov' (2 times) and 'ly' (12 times) are more frequent than 'ob' (1 time) and 'le' (7 times). GP correctly predicts these letters because the graph connects them to good neighbors. However, GP mislabels 'l' as 'i', since most of this letter's neighbors are i's. The coupling of GP and CRF via PR links the neighbor information with bigram information — 'bl' (5 times) is more frequent than 'bi' in the training set — to yield the correct labeling.

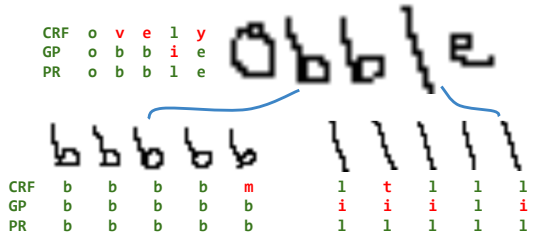


Figure 2: Predictions on the word 'Wobble' and the 5-NNs of its first 'b' and 'l'.

5 Conclusion

We have presented an efficient extension of the posterior regularization (PR) framework to a more general class of penalty functions. Encouraging results using a graph Laplacian penalty suggest potential applications to a much larger class of weakly supervised problems.

Acknowledgements

J. Gillenwater was supported by a National Science Foundation Graduate Research Fellowship. L. He and B. Taskar were partially supported by ONR Young Investigator Award N000141010746.

References

[Altun et al.2005] Y. Altun, D. McAllester, and M. Belkin. 2005. Maximum Margin Semi-Supervised Learning for Structured Variables. In *Proc. NIPS*.

[Belkin et al.2005] M. Belkin, P. Niyogi, and V. Sindhwani. 2005. On Manifold Regularization. In *Proc. AISTATS*.

[Bertsekas2004] D. Bertsekas. 2004. *Nonlinear Programming*.

[Buchholz and Marsi2006] S. Buchholz and E. Marsi. 2006. CoNLL-X Shared Task on Multilingual Dependency Parsing. In *Proc. CoNLL*.

[Chapelle et al.2006] O. Chapelle, B. Schölkopf, and A. Zien, editors. 2006. *Semi-Supervised Learning*.

[Collins et al.2005] M. Collins, P. Bartlett, D. McAllester, and B. Taskar. 2005. Exponentiated Gradient Algorithms for Large-Margin Structured Classification. In *Proc. NIPS*.

[Collins et al.2008] M. Collins, A. Globerson, T. Koo, and X. Carreras. 2008. Exponentiated Gradient Algorithms for Conditional Random Fields and Max-Margin Markov Networks. *JMLR*.

[Das and Petrov2011] D. Das and S. Petrov. 2011. Unsupervised Part-of-Speech Tagging with Bilingual Graph-Based Projections. In *Proc. ACL*.

[Das et al.2012] D. Das, S. Petrov, and R. McDonald. 2012. A Universal Part-of-Speech Tagset. In *Proc. LREC*.

[Ganchev et al.2010] K. Ganchev, J. Graça, J. Gillenwater, and B. Taskar. 2010. Posterior Regularization for Structured Latent Variable Models. *JMLR*.

[Griewank1988] A. Griewank. 1988. On Automatic Differentiation. Technical report, Argonne National Laboratory.

[Joachims2003] T. Joachims. 2003. Transductive Learning via Spectral Graph Partitioning. In *Proc. ICML*.

[Kassel1995] R. Kassel. 1995. *A Comparison of Approaches to On-line Handwritten Character Recognition*. Ph.D. thesis, Massachusetts Institute of Technology.

[Kivinen and Warmuth1995] J. Kivinen and M. Warmuth. 1995. Additive Versus Exponentiated Gradient Updates for Linear Prediction. In *Proc. STOC*.

[Mann and McCallum2007] G. Mann and A. McCallum. 2007. Simple, Robust, Scalable Semi-Supervised Learning via Expectation Regularization. In *Proc. ICML*.

[Mann and McCallum2008] G. Mann and A. McCallum. 2008. Generalized Expectation Criteria for Semi-Supervised Learning of Conditional Random Fields. In *Proc. ACL*.

[Marcus et al.1993] M. Marcus, M. Marcinkiewicz, and B. Santorini. 1993. Building a Large Annotated Corpus of English: Then Penn Treebank. *Computational Linguistics*.

[Nivre et al.2007] J. Nivre, J. Hall, S. Kübler, R. McDonald, J. Nilsson, S. Riedel, and D. Yuret. 2007. The CoNLL 2007 Shared Task on Dependency Parsing. In *Proc. CoNLL*.

[Pele and Werman2009] O. Pele and M. Werman. 2009. Fast and Robust Earth Mover's Distances. In *Proc. ICCV*.

[Subramanya and Bilmes2009] A. Subramanya and J. Bilmes. 2009. Entropic Graph Regularization in Non-Parametric Semi-Supervised Classification. In *Proc. NIPS*.

[Subramanya et al.2010] A. Subramanya, S. Petrov, and F. Pereira. 2010. Efficient Graph-Based Semi-Supervised Learning of Structured Tagging Models. In *Proc. EMNLP*.

[Taskar et al.2003] B. Taskar, C. Guestrin, and D. Koller. 2003. Max Margin Markov Networks. In *Proc. NIPS*.

[van de Panne and Whinston1964] C. van de Panne and A. Whinston. 1964. The Simplex and the Dual Method for Quadratic Programming. *Operational Research Quarterly*.

[Zhu and Lafferty2005] X. Zhu and J. Lafferty. 2005. Harmonic Mixtures: Combining Mixture Models and Graph-Based Methods for Inductive and Scalable Semi-Supervised Learning. In *Proc. ICML*.

[Zhu et al.2003] X. Zhu, Z. Ghahramani, and J. Lafferty. 2003. Semi-Supervised Learning Using Gaussian Fields and Harmonic Functions. In *Proc. ICML*.

[Zhu2005] X. Zhu. 2005. Semi-Supervised Learning Literature Survey. Technical report, University of Wisconsin-Madison.