Journal of Machine Learning Research 9 (2008) 1775-1822 Submitted 9/07; Published 8/08

Exponentiated Gradient Algorithms for Conditional Random Fields and Max-Margin Markov Networks

Michael Collins* MCOLLINS@CSAIL.MIT.EDU
Amir Globerson* GAMIR@CSAIL.MIT.EDU
Terry Koo* MAESTRO@CSAIL.MIT.EDU
Xavier Carreras CARRERAS@CSAIL.MIT.EDU
Computer Science and Artificial Intelligence Laboratory
Massachusetts Institute of Technology
Cambridge, MA 02139, USA

Peter L. Bartlett BARTLETT@CS.BERKELEY.EDU
University of California, Berkeley
Division of Computer Science and Department of Statistics
Berkeley, CA 94720, USA

Editor: John Lafferty

Abstract

Log-linear and maximum-margin models are two commonly-used methods in supervised machine learning, and are frequently used in structured prediction problems. Efficient learning of parameters in these models is therefore an important problem, and becomes a key factor when learning from very large data sets. This paper describes exponentiated gradient (EG) algorithms for training such models, where EG updates are applied to the convex dual of either the log-linear or max-margin objective function; the dual in both the log-linear and max-margin cases corresponds to minimizing a convex function with simplex constraints. We study both batch and online variants of the algorithm, and provide rates of convergence for both cases. In the max-margin case, $O(\frac{1}{\varepsilon})$ EG updates are required to reach a given accuracy ε in the dual; in contrast, for log-linear models only $O(\log(\frac{1}{\varepsilon}))$ updates are required. For both the max-margin and log-linear cases, our bounds suggest that the online EG algorithm requires a factor of n less computation to reach a desired accuracy than the batch EG algorithm, where n is the number of training examples. Our experiments confirm that the online algorithms are much faster than the batch algorithms in practice. We describe how the EG updates factor in a convenient way for structured prediction problems, allowing the algorithms to be efficiently applied to problems such as sequence learning or natural language parsing. We perform extensive evaluation of the algorithms, comparing them to L-BFGS and stochastic gradient descent for log-linear models, and to SVM-Struct for max-margin models. The algorithms are applied to a multi-class problem as well as to a more complex large-scale parsing task. In all these settings, the EG algorithms presented here outperform the other methods.

Keywords: exponentiated gradient, log-linear models, maximum-margin models, structured prediction, conditional random fields

∗. These authors contributed equally.

1. Introduction

Structured prediction problems involve learning to map inputs x to labels y, where the labels have rich internal structure, and where the set of possible labels for a given input is typically exponential in size. Examples of structured prediction problems include sequence labeling and natural language parsing. Several models that implement learning in this scenario have been proposed over the last few years, including log-linear models such as conditional random fields (CRFs, Lafferty et al., 2001), and maximum-margin models such as maximum-margin Markov networks (Taskar et al., 2004a).

For both log-linear and max-margin models, learning is framed as minimization of a regularized loss function which is convex. In spite of the convexity of the objective function, finding the optimal parameters for these models can be computationally intensive, especially for very large data sets. This problem is exacerbated in structured prediction problems, where the large size of the set of possible labels adds an additional layer of complexity. The development of efficient optimization algorithms for learning in structured prediction problems is therefore an important problem.

In this paper we describe learning algorithms that exploit the structure of the dual optimization problems for log-linear and max-margin models. For both log-linear and max-margin models the dual problem corresponds to the minimization of a convex function Q subject to simplex constraints (Jaakkola and Haussler, 1999; Lebanon and Lafferty, 2002; Taskar et al., 2004a). More specifically, the goal is to find

$$\operatorname*{argmin}_{\forall i,\, \mathbf{u}_i \in \Delta} Q(\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_n) \;, \tag{1}$$

where n is the number of training examples, each $\mathbf{u}_i$ is a vector of dual variables for the i'th training example, and $Q(\mathbf{u})$ is a convex function.[1] The size of each vector $\mathbf{u}_i$ is $|\mathcal{Y}|$, where $\mathcal{Y}$ is the set of possible labels for any training example. Furthermore, $\mathbf{u}_i$ is constrained to belong to the simplex of distributions over $\mathcal{Y}$, defined as:

$$\Delta = \left\{ \mathbf{p} \in \mathbb{R}^{|\mathcal{Y}|} : p_y \geq 0 \;\;,\;\; \sum_{y \in \mathcal{Y}} p_y = 1 \right\} \;. \tag{2}$$

Thus each $\mathbf{u}_i$ is constrained to form a distribution over the set of possible labels. The max-margin and log-linear problems differ only in their definition of Q.

The algorithms in this paper make use of exponentiated gradient (EG) updates (Kivinen and Warmuth, 1997) in solving the problem in Eq. 1, in particular for the cases of log-linear or max-margin models. We focus on two classes of algorithms, which we call *batch* and *online*. In the batch case, the entire set of $\mathbf{u}_i$ variables is updated simultaneously at each iteration of the algorithm; in the online case, a single $\mathbf{u}_i$ variable is updated at each step. The "online" case essentially corresponds to coordinate-descent on the dual function Q, and is similar to the SMO algorithm (Platt, 1998) for training SVMs. The online algorithm has the advantage of updating the parameters after every sample point, rather than after making a full pass over the training examples; intuitively, this should lead to considerably faster rates of convergence when compared to the batch algorithm, and indeed our experimental and theoretical results support this intuition. A different class of online algorithms consists of stochastic gradient descent (SGD) and its variants (e.g., see LeCun et al., 1998; Vishwanathan et al., 2006). In contrast to SGD, however, the EG algorithm is guaranteed to

1. In what follows we use $\mathbf{u}$ to denote the variables $\mathbf{u}_1, \ldots, \mathbf{u}_n$.

improve the dual objective at each step, and this objective may be calculated after each example without performing a pass over the entire data set. This is particularly convenient when making a choice of learning rate in the updates.

We describe theoretical results concerning the convergence of the EG algorithms, as well as experiments. Our key results are as follows:

- For the max-margin case, we show that $O(\frac{1}{\varepsilon})$ time is required for both the online and batch algorithms to converge to within ε of the optimal value of $Q(\mathbf{u})$. This is qualitatively similar to recent results in the literature for max-margin approaches (e.g., see Shalev-Shwartz et al., 2007). For log-linear models, we show convergence rates of $O(\log(\frac{1}{\varepsilon}))$, a significant improvement over the max-margin case.

- For both the max-margin and log-linear cases, our bounds suggest that the online algorithm requires a factor of n less computation to reach a desired accuracy, where n is the number of training examples. Our experiments confirm that the online algorithms are much faster than the batch algorithms in practice.

- We describe how the EG algorithms can be efficiently applied to an important class of structured prediction problems where the set of labels $\mathcal{Y}$ is exponential in size. In this case the number of dual variables is also exponential in size, making algorithms which deal directly with the $\mathbf{u}_i$ variables intractable. Following Bartlett et al. (2005), we focus on a formulation where each label y is represented as a set of "parts", for example corresponding to labeled cliques in a max-margin network, or context-free rules in a parse tree. Under an assumption that part-based marginals can be calculated efficiently—for example using junction tree algorithms for CRFs, or the inside-outside algorithm for context-free parsing—the EG algorithms can be implemented efficiently for both max-margin and log-linear models.

- In our experiments we compare the online EG algorithm to various state-of-the-art algorithms. For log-linear models, we compare to the L-BFGS algorithm (Byrd et al., 1995) and to stochastic gradient descent. For max-margin models we compare to the SVM-Struct algorithm of Tsochantaridis et al. (2004). The methods are applied to a standard multi-class learning problem, as well as to a more complex natural language parsing problem. In both settings we show that the EG algorithm converges to the optimum much faster than the other algorithms.

- In addition to proving convergence results for the definition of $Q(\mathbf{u})$ used in max-margin and log-linear models, we give theorems which may be useful when optimizing other definitions of $Q(\mathbf{u})$ using EG updates. In particular, we give conditions for convergence which depend on bounds relating the Bregman divergence derived from $Q(\mathbf{u})$ to the Kullback-Leibler divergence. Depending on the form of these bounds for a particular $Q(\mathbf{u})$, either $O(\frac{1}{\varepsilon})$ or $O(\log(\frac{1}{\varepsilon}))$ rates of convergence can be derived.

The rest of this paper is organized as follows. In Section 2, we introduce the log-linear and max-margin learning problems, and describe their dual optimization problems. Section 3 describes the batch and online EG algorithms; in Section 4, we describe how the algorithms can be efficiently applied to structured prediction problems. Section 5 then gives convergence proofs for the batch and

online cases. Section 6 discusses related work. Sections 7 and 8 give experiments, and Section 9 discusses our results.

This work builds on previous work described by Bartlett et al. (2005) and Globerson et al. (2007). Bartlett et al. (2005) described the application of the EG algorithm to max-margin parameter estimation, and showed how the method can be applied efficiently to part-based formulations. Globerson et al. (2007) extended the approach to log-linear parameter estimation, and gave new convergence proofs for both max-margin and log-linear estimation. The work in the current paper gives several new results. We prove rates of convergence for a randomized version of the EG online algorithm; previous work on EG algorithms had not given convergence rates for the online case. We also report new experiments, including experiments with the randomized strategy. Finally, the $O(\log(\frac{1}{\varepsilon}))$ convergence rates for the log-linear case are new. The results in Globerson et al. (2007) gave $O(\frac{1}{\varepsilon})$ rates for the batch algorithm for log-linear models, and did not give any theoretical rates of convergence for the online case.

2. Primal and Dual Problems for Regularized Loss Minimization

In this section we present the log-linear and max-margin optimization problems for supervised learning. For each problem, we describe the equivalent dual optimization problem, which will form the core of our optimization approach.

2.1 The Primal Problems

Consider a supervised learning setting with objects $x \in \mathcal{X}$ and labels $y \in \mathcal{Y}$.[2] In the structured learning setting, the labels may be sequences, trees, or other high-dimensional data with internal structure. Assume we are given a function $\mathbf{f}(x,y) : \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}^d$ that maps (x,y) pairs to feature vectors. Our goal is to construct a linear prediction rule

$$h(x,\mathbf{w}) = \operatorname*{argmax}_{y \in \mathcal{Y}} \mathbf{w} \cdot \mathbf{f}(x,y) \ ,$$

with parameters $\mathbf{w} \in \mathbb{R}^d$, such that $h(x,\mathbf{w})$ is a good approximation of the *true* label of x. The parameters $\mathbf{w}$ are learned by minimizing a regularized loss

$$\mathcal{L}(\mathbf{w}; \{(x_i,y_i)\}_{i=1}^n, C) = \sum_{i=1}^n \ell(\mathbf{w},x_i,y_i) + \frac{C}{2}\|\mathbf{w}\|^2 \ ,$$

defined over a labeled training set $\{(x_i,y_i)\}_{i=1}^n$. Here $C > 0$ is a constant determining the amount of regularization. The function ℓ measures the loss incurred in using $\mathbf{w}$ to predict the label of x_i, given that the true label is y_i.

In this paper we will consider two definitions for $\ell(\mathbf{w},x_i,y_i)$. The first definition, originally introduced by Taskar et al. (2004a), is a variant of the hinge loss, and is defined as follows:

$$\ell_{\text{MM}}(\mathbf{w},x_i,y_i) = \max_{y \in \mathcal{Y}} \Big[e(x_i,y_i,y) - \mathbf{w} \cdot (\mathbf{f}(x_i,y_i) - \mathbf{f}(x_i,y)) \Big] \ . \tag{3}$$

2. In general the set of labels for a given example x may be a set $\mathcal{Y}(x)$ that depends on x; in fact, in our experiments on dependency parsing $\mathcal{Y}$ does depend on x. For simplicity, in this paper we use the fixed notation $\mathcal{Y}$ for all x; it is straightforward to extend our notation to the more general case.

Here $e(x_i,y_i,y)$ is some non-negative measure of the error incurred in predicting y instead of y_i as the label of x_i. We assume that $e(x_i,y_i,y_i) = 0$ for all i, so that no loss is incurred for correct prediction, and therefore $\ell_{\rm MM}(\mathbf{w},x_i,y_i)$ is always non-negative. This loss function corresponds to a maximum-margin approach, which explicitly penalizes training examples for which, for some $y \neq y_i$,

$$\mathbf{w} \cdot (\mathbf{f}(x_i,y_i) - \mathbf{f}(x_i,y)) < e(x_i,y_i,y) \ .$$

The second loss function that we will consider is based on log-linear models, and is commonly used in conditional random fields (CRFs, Lafferty et al., 2001). First define the conditional distribution

$$p(y \,|\, x;\mathbf{w}) = \frac{1}{Z_x} e^{\mathbf{w}\cdot\mathbf{f}(x,y)} \ ,$$

where $Z_x = \sum_y e^{\mathbf{w}\cdot\mathbf{f}(x,y)}$ is the partition function. The loss function is then the negative log-likelihood under the parameters $\mathbf{w}$:

$$\ell_{\rm LL}(\mathbf{w},x_i,y_i) = -\log p(y_i \,|\, x_i;\mathbf{w}) \ .$$

The function $\mathcal{L}$ is convex in $\mathbf{w}$ for both definitions $\ell_{\rm MM}$ and $\ell_{\rm LL}$. Furthermore, in both cases minimization of $\mathcal{L}$ can be re-cast as optimization of a dual convex problem. The dual problems in the two cases have a similar structure, as we describe in the next two sections.

2.2 The Log-Linear Dual

The problem of minimizing $\mathcal{L}$ with the loss function $\ell_{\rm LL}$ can be written as

$$\underline{\text{P-LL}}: \quad \mathbf{w}^* = \operatorname*{argmin}_{\mathbf{w}} \sum_i -\log p(y_i \,|\, x_i;\mathbf{w}) + \frac{C}{2}\|\mathbf{w}\|^2 \ .$$

This is a convex optimization problem, and has an equivalent convex dual which was derived by Lebanon and Lafferty (2002). Denote the dual variables by $u_{i,y}$ where $i = 1,\ldots,n$ and $y \in \mathcal{Y}$. We also use $\mathbf{u}$ to denote the set of all variables, and $\mathbf{u}_i$ the set of all variables corresponding to a given i. Thus $\mathbf{u} = [\mathbf{u}_1,\ldots,\mathbf{u}_n]$. We assume $\mathbf{u}$ is a column vector. Define the function $Q_{\rm LL}(\mathbf{u})$ as

$$Q_{\rm LL}(\mathbf{u}) = \sum_i \sum_y u_{i,y} \log u_{i,y} + \frac{1}{2C}\|\mathbf{w}(\mathbf{u})\|^2 \ ,$$

where

$$\mathbf{w}(\mathbf{u}) = \sum_i \sum_y u_{i,y} \mathbf{g}_{i,y} \ ,$$

and where $\mathbf{g}_{i,y} = \mathbf{f}(x_i,y_i) - \mathbf{f}(x_i,y)$. We shall find the following matrix notation convenient:

$$Q_{\rm LL}(\mathbf{u}) = \sum_i \sum_y u_{i,y} \log u_{i,y} + \frac{1}{2}\mathbf{u}^T A \mathbf{u} \ , \tag{4}$$

where A is a matrix of size $n|\mathcal{Y}| \times n|\mathcal{Y}|$ indexed by pairs (i,y), and $A_{(i,y),(j,z)} = \frac{1}{C}\mathbf{g}_{i,y} \cdot \mathbf{g}_{j,z}$.

In what follows we denote the set of distributions over $\mathcal{Y}$, that is, the $|\mathcal{Y}|$-dimensional probability simplex, by Δ, as in Eq. 2. The Cartesian product of n distributions over $\mathcal{Y}$ will be denoted by Δ^n. The dual optimization problem is then

$$\underline{\text{D-LL}}: \quad \mathbf{u}^* = \begin{array}{ll} \operatorname{argmin} & Q_{\rm LL}(\mathbf{u}) \\ s.t. & \mathbf{u} \in \Delta^n \ . \end{array}$$

The minimum of D-LL is equal to -1 times the minimum of P-LL. The duality between P-LL and D-LL implies that the primal and dual solutions satisfy $C\mathbf{w}^* = \mathbf{w}(\mathbf{u}^*)$.

2.3 The Max-Margin Dual

When the loss is defined using $\ell_{\text{MM}}(\mathbf{w}, x_i, y_i)$, the primal optimization problem is as follows:

$$\underline{\text{P-MM}}: \quad \mathbf{w}^* = \operatorname*{argmin}_{\mathbf{w}} \sum_i \max_y \Big[e(x_i, y_i, y) - \mathbf{w} \cdot (\mathbf{f}(x_i, y_i) - \mathbf{f}(x_i, y)) \Big] + \frac{C}{2} \|\mathbf{w}\|^2 \ .$$

The dual of this minimization problem was derived in Taskar et al. (2004a) (see also Bartlett et al., 2005). We first define the dual objective

$$Q_{\text{MM}}(\mathbf{u}) = -\mathbf{b}^T \mathbf{u} + \frac{1}{2} \mathbf{u}^T A \mathbf{u} \ . \tag{5}$$

Here, the matrix A is as defined above and $\mathbf{b} \in \mathbb{R}^{n|\mathcal{Y}|}$ is a vector defined as $b_{i,y} = e(x_i, y_i, y)$. The convex dual for the max-margin case is then given by

$$\underline{\text{D-MM}}: \quad \mathbf{u}^* = \begin{array}{ll} \operatorname{argmin} & Q_{\text{MM}}(\mathbf{u}) \\ s.t. & \mathbf{u} \in \Delta^n \ . \end{array}$$

The minimum of D-MM is equal to -1 times the minimum of P-MM. (Note that for D-MM the minimizer $\mathbf{u}^*$ may not be unique; in this case we take $\mathbf{u}^*$ to be any member of the set of minimizers of $Q_{\text{MM}}(\mathbf{u})$). The optimal primal parameters are again related to the optimal dual parameters, through $C\mathbf{w}^* = \mathbf{w}(\mathbf{u}^*)$. Here again the constraints are that $\mathbf{u}_i$ is a distribution over $\mathcal{Y}$ for all i.

It can be seen that the D-LL and D-MM problems have a similar structure, in that they both involve minimization of a convex function $Q(\mathbf{u})$ over the set Δ^n. This will allow us to describe algorithms for both problems using a common framework.

3. Exponentiated Gradient Algorithms

In this section we describe batch and online algorithms for minimizing a convex function $Q(\mathbf{u})$ subject to the constraints $\mathbf{u} \in \Delta^n$. The algorithms can be applied to both the D-LL and D-MM optimization problems that were introduced in the previous section. The algorithms we describe are based on exponentiated gradient (EG) updates, originally introduced by Kivinen and Warmuth (1997) in the context of online learning algorithms.[3]

The EG updates rely on the following operation. Given a sequence of distributions $\mathbf{u} \in \Delta^n$, a new sequence of distributions $\mathbf{u}'$ can be obtained as

$$u'_{i,y} = \frac{1}{Z_i} u_{i,y} e^{-\eta \nabla_{i,y}} \ ,$$

where $\nabla_{i,y} = \frac{\partial Q(\mathbf{u})}{\partial u_{i,y}}$, $Z_i = \sum_{\hat{y}} u_{i,\hat{y}} e^{-\eta \nabla_{i,\hat{y}}}$ is a partition function ensuring normalization of the distribution $\mathbf{u}'_i$, and the parameter $\eta > 0$ is a learning rate. We will also use the notation $u'_{i,y} \propto u_{i,y} e^{-\eta \nabla_{i,y}}$ where the partition function should be clear from the context.

3. Kivinen and Warmuth (1997) study the online setting, as opposed to a fixed data set which we study here. They are thus not interested in minimizing a fixed objective, but rather study regret type bounds. This leads to algorithms and theoretical analyses that are different from the ones considered in the current work.

Clearly $\mathbf{u}' \in \Delta^n$ by construction. For the dual function $Q_{\text{LL}}(\mathbf{u})$ the gradient is

$$\nabla_{i,y} = 1 + \log u_{i,y} + \frac{1}{C}\mathbf{w}(\mathbf{u}) \cdot \mathbf{g}_{i,y} \ ,$$

and for $Q_{\text{MM}}(\mathbf{u})$ the gradient is

$$\nabla_{i,y} = -b_{i,y} + \frac{1}{C}\mathbf{w}(\mathbf{u}) \cdot \mathbf{g}_{i,y} \ .$$

In this paper we will consider both parallel (batch), and sequential (online) applications of the EG updates, defined as follows:

- **Batch**: At every iteration the dual variables $\mathbf{u}_i$ are simultaneously updated for all $i = 1, \ldots, n$.
- **Online**: At each iteration a single example k is chosen uniformly at random from $\{1, \ldots, n\}$ and $\mathbf{u}_k$ is updated to give $\mathbf{u}'_k$. The dual variables $\mathbf{u}_i$ for $i \neq k$ are left unchanged.

Pseudo-code for the two schemes is given in Figures 1 and 2. From here on we will refer to the batch and online EG algorithms applied to the log-linear dual as LLEG-Batch, and LLEG-Online respectively. Similarly, when applied to the max-margin dual, they will be referred to as MMEG-Batch and MMEG-Online.

Note that another plausible online algorithm would be a "deterministic" algorithm that repeatedly cycles over the training examples in a fixed order. The motivation for the alternative, randomized, algorithm is two-fold. First, we are able to prove bounds on the rate of convergence of the randomized algorithm; we have not been able to prove similar bounds for the deterministic variant. Second, our experiments show that the randomized variant converges significantly faster than the deterministic algorithm.

The EG online algorithm is essentially performing coordinate descent on the dual objective, and is similar to SVM algorithms such as SMO (Platt, 1998). For binary classification, the exact minimum of the dual objective with respect to a given coordinate can be found in closed form,[4] and more complicated algorithms such as the exponentiated-gradient method may be unnecessary. However for multi-class or structured problems, the exact minimum with respect to a coordinate $\mathbf{u}_i$ (i.e., a set of $|\mathcal{Y}|$ dual variables) cannot be found in closed form: this is a key motivation for the use of EG algorithms in this paper.

In Section 5 we give convergence proofs for the batch and online algorithms. The techniques used in the convergence proofs are quite general, and could potentially be useful in deriving EG algorithms for convex functions Q other than Q_{LL} and Q_{MM}. Before giving convergence results for the algorithms, we describe in the next section how the EG algorithms can be applied to structured problems.

4. Structured Prediction with the EG Algorithms

We now describe how the EG updates can be applied to structured prediction problems, for example parameter estimation in CRFs or natural language parsing. In structured problems the label set $\mathcal{Y}$ is typically very large, but labels can have useful internal structure. As one example, in CRFs each

4. This is true for the max-margin case. For log-linear models, minimization with respect to a single coordinate is a little more involved.

Inputs: A convex function $Q : \Delta^n \to \mathbb{R}$, a learning rate $\eta > 0$.

Initialization: Set $\mathbf{u}^1$ to a point in the interior of Δ^n.

Algorithm:

- For $t = 1, \ldots, T$, repeat:
  - For all i, y, calculate $\nabla_{i,y} = \frac{\partial Q(\mathbf{u}^t)}{\partial u_{i,y}}$
  - For all i, y, update $u_{i,y}^{t+1} \propto u_{i,y}^t e^{-\eta \nabla_{i,y}}$

Output: Final parameters $\mathbf{u}^{T+1}$.

Figure 1: A general batch EG Algorithm for minimizing $Q(\mathbf{u})$ subject to $\mathbf{u} \in \Delta^n$. We use $\mathbf{u}^t$ to denote the set of parameters after t iterations.

Inputs: A convex function $Q : \Delta^n \to \mathbb{R}$, a learning rate $\eta > 0$.

Initialization: Set $\mathbf{u}^1$ to a point in the interior of Δ^n.

Algorithm:

- For $t = 1, \ldots, T$, repeat:
  - Choose k_t uniformly at random from the set $\{1, 2, \ldots, n\}$
  - For all y, calculate: $\nabla_{k_t,y} = \frac{\partial Q(\mathbf{u}^t)}{\partial u_{k_t,y}}$
  - For all y, update $u_{k_t,y}^{t+1} \propto u_{k_t,y}^t e^{-\eta \nabla_{k_t,y}}$
  - For all $i \neq k_t$, set $\mathbf{u}_i^{t+1} = \mathbf{u}_i^t$

Output: Final parameters $\mathbf{u}^{T+1}$.

Figure 2: A general randomized online EG Algorithm for minimizing $Q(\mathbf{u})$ subject to $\mathbf{u} \in \Delta^n$.

label y is an m-dimensional vector specifying the labeling of all m vertices in a graph. In parsing each label y is an entire parse tree. In both of these cases, the number of labels typically grows exponentially quickly with respect to the size of the inputs x.

We follow the framework for structured problems described by Bartlett et al. (2005). Each label y is defined to be a set of *parts*. We use R to refer to the set of all possible parts.[5] We make the assumption that the feature vector for an entire label y decomposes into a sum over feature vectors for individual parts as follows:

$$\mathbf{f}(x, y) = \sum_{r \in y} \mathbf{f}(x, r) \ .$$

5. As with the label set $\mathcal{Y}$, the set of parts R may in general be a set $R(x)$ that depends on x. For simplicity, we assume that R is fixed.

Note that we have overloaded $\mathbf{f}$ to apply to either labels y or parts r.

As one example, consider a CRF which has an underlying graph with m nodes, and a maximum clique size of 2. Assume that each node can be labeled with one of two labels, 0 or 1. In this case the labeling of an entire graph is a vector $\mathbf{y} \in \{0,1\}^m$. Each possible input x is usually a vector in $\mathcal{X}^m$ for some set $\mathcal{X}$, although this does not have to be the case. Each part corresponds to a tuple (u,v,y_u,y_v) where (u,v) is an edge in the graph, and y_u, y_v are the labels for the two vertices u and v. The feature vector $\mathbf{f}(x,r)$ can then track any properties of the input x together with the labeled clique $r = (u,v,y_u,y_v)$. In CRFs with clique size greater than 2, each part corresponds to a labeled clique in the graph. In natural language parsing, each part can correspond to a context-free rule at a particular position in the sentence x (see Bartlett et al., 2005; Taskar et al., 2004b, for more details).

The label set $\mathcal{Y}$ can be extremely large in structured prediction problems. For example, in a CRF with an underlying graph with m nodes and k possible labels at each node, there are k^m possible labelings of the entire graph. The algorithms we have presented so far require direct manipulation of dual variables $u_{i,y}$ corresponding to each possible labeling of each training example; they will therefore be intractable in cases where there are an exponential number of possible labels. However, in this section we describe an approach that does allow an efficient implementation of the algorithms in several cases. The approach is based on the method originally described in Bartlett et al. (2005).

The key idea is as follows. Instead of manipulating the dual variables $\mathbf{u}_i$ for each i directly, we will make use of alternative data structures $\mathbf{s}_i$ for all i. Each $\mathbf{s}_i$ is a vector of real values $s_{i,r}$ for all $r \in R$. In general we will assume that there are a tractable (polynomial) number of possible parts, and therefore that the number of $s_{i,r}$ variables is also polynomial. For example, for a linear chain CRF with m nodes and k labels at every node, each part takes the form $r = (u,v,y_u,y_v)$, and there are $(m-1)k^2$ possible parts.

In the max-margin case, we follow Taskar et al. (2004a) and make the additional assumption that the error function decomposes into "local" error functions over parts:

$$e(x_i,y_i,y) = \sum_{r \in y} e(x_i,y_i,r) \ .$$

For example, when $\mathcal{Y}$ is a sequence of variables, the cost could be the Hamming distance between the correct sequence y_i and the predicted sequence y; it is straightforward to decompose the Hamming distance as a sum over parts as shown above. For brevity, in what follows we use $e_{i,r}$ instead of $e(x_i,y_i,r)$.

The $\mathbf{s}_i$ variables are used to implicitly define regular dual values $\mathbf{u}_i = \mathbf{p}(\mathbf{s}_i)$ where $\mathbf{p} : \mathbb{R}^{|R|} \to \Delta$ is defined as

$$p_y(\mathbf{s}) = \frac{\exp\left\{\sum_{r \in y} s_r\right\}}{\sum_{y'} \exp\left\{\sum_{r \in y'} s_r\right\}} \ .$$

To see how the $\mathbf{s}_i$ variables can be updated, consider again the EG updates on the dual $\mathbf{u}$ variables. The EG updates in all algorithms in this paper take the form

$$u'_{i,y} = \frac{u_{i,y} \exp\{-\eta \nabla_{i,y}\}}{\sum_{\hat{y}} u_{i,\hat{y}} \exp\{-\eta \nabla_{i,\hat{y}}\}} \ ,$$

where for Q_{LL}

$$\nabla_{i,y} = 1 + \log u_{i,y} + \frac{1}{C} \mathbf{w}(\mathbf{u}) \cdot (\mathbf{f}(x_i,y_i) - \mathbf{f}(x_i,y)) \ ,$$

and for Q_{MM},

$$\nabla_{i,y} = -b_{i,y} + \frac{1}{C}\mathbf{w}(\mathbf{u}) \cdot (\mathbf{f}(x_i, y_i) - \mathbf{f}(x_i, y)) \ ,$$

where $b_{i,y} = e(x_i, y_i, y)$ as in Section 2.3.

Notice that, for both objective functions, the gradients can be expressed as a sum over parts. For the Q_{LL} objective function, this follows from the fact that $\mathbf{u}_i = \mathbf{p}(\mathbf{s}_i)$ and from the assumption that the feature vector decomposes into parts. For the Q_{MM} objective, it follows from the latter, and the assumption that the loss decomposes into parts. The following lemma describes how EG updates on the $\mathbf{u}$ variables can be restated in terms of updates to the $\mathbf{s}$ variables, provided that the gradient decomposes into parts in this way.

Lemma 1 *For a given $\mathbf{u} \in \Delta^n$, and for a given $i \in [1 \ldots n]$, take $\mathbf{u}'_i$ to be the updated value for $\mathbf{u}_i$ derived using an EG step, that is,*

$$u'_{i,y} = \frac{u_{i,y} \exp\{-\eta \nabla_{i,y}\}}{\sum_{\hat{y}} u_{i,\hat{y}} \exp\{-\eta \nabla_{i,\hat{y}}\}} \ .$$

Suppose that, for some G_i and $g_{i,r}$, we can write $\nabla_{i,y} = G_i + \sum_{r \in y} g_{i,r}$ for all y. Then if $\mathbf{u}_i = \mathbf{p}(\mathbf{s}_i)$ for some $\mathbf{s}_i \in \mathcal{R}^{|R|}$, and for all r we define

$$s'_{i,r} = s_{i,r} - \eta g_{i,r} \ ,$$

it follows that $\mathbf{u}'_i = \mathbf{p}(\mathbf{s}'_i)$.

Proof: We show that, for $\mathbf{u}_i = \mathbf{p}(\mathbf{s}_i)$, updating the $s_{i,r}$ as described leads to $\mathbf{p}(\mathbf{s}'_i) = \mathbf{u}'_i$. For suitable partition functions Z_i, Z'_i, and Z''_i, we can write

$$\begin{aligned}
p_y(\mathbf{s}'_i) &= \frac{\exp\left\{\sum_{r \in y}(s_{i,r} - \eta g_{i,r})\right\}}{Z_i} \\
&= \frac{u_{i,y} \exp\left\{-\eta \sum_{r \in y} g_{i,r}\right\}}{Z'_i} \\
&= \frac{u_{i,y} \exp\{-\eta(\nabla_{i,y} - G_i)\}}{Z'_i} \\
&= \frac{u_{i,y} \exp\{-\eta \nabla_{i,y}\}}{Z''_i} \\
&= u'_{i,y} \ .
\end{aligned}$$

□

In the case of the Q_{LL} objective, a suitable update is

$$s'_{i,r} = s_{i,r} - \eta \left(s_{i,r} - \frac{1}{C}\mathbf{w}(\mathbf{u}) \cdot \mathbf{f}(x_i, r) \right) \ .$$

In the case of the Q_{MM} objective, a suitable update is

$$s'_{i,r} = s_{i,r} - \eta \left(-e_{i,r} - \frac{1}{C}\mathbf{w}(\mathbf{u}) \cdot \mathbf{f}(x_i, r) \right) \ .$$

Inputs: Examples $\{(x_i,y_i)\}_{i=1}^n$, learning rate $\eta > 0$.

Initialization: For each $i = 1 \ldots n$, set $\mathbf{s}_i^1$ to some (possibly different) point in $\mathbb{R}^{|R|}$.

Algorithm:

- Calculate
$$\mathbf{w}^1 = \sum_i \mathbf{f}(x_i,y_i) - \sum_{i,y} p_y(\mathbf{s}_i^1)\mathbf{f}(x_i,y)$$
- For $t = 1,\ldots,T$, repeat:
  - Choose k_t uniformly at random from the set $[1,2,\ldots,n]$
  - For all $r \in R$,
$$\textbf{If optimizing } Q_{\text{LL}}\textbf{:} \quad s_{k_t,r}^{t+1} = s_{k_t,r}^t - \eta\left(s_{k_t,r}^t - \frac{1}{C}\mathbf{w}^t \cdot \mathbf{f}(x_{k_t},r)\right)$$
$$\textbf{If optimizing } Q_{\text{MM}}\textbf{:} \quad s_{k_t,r}^{t+1} = s_{k_t,r}^t - \eta\left(-e_{k_t,r} - \frac{1}{C}\mathbf{w}^t \cdot \mathbf{f}(x_{k_t},r)\right)$$
  - For all $i \neq k_t$, for all r, set $s_{i,r}^{t+1} = s_{i,r}^t$
  - Calculate
$$\begin{aligned}\mathbf{w}^{t+1} &= \sum_i \mathbf{f}(x_i,y_i) - \sum_{i,y} p_y(\mathbf{s}_i^{t+1})\mathbf{f}(x_i,y) \\ &= \mathbf{w}^t + \sum_y p_y(\mathbf{s}_{k_t}^t)\mathbf{f}(x_{k_t},y) - \sum_y p_y(\mathbf{s}_{k_t}^{t+1})\mathbf{f}(x_{k_t},y)\end{aligned}$$

Output: Final dual parameters $\mathbf{s}^{T+1}$ or primal parameters $\frac{1}{C}\mathbf{w}^{T+1}$.

Figure 3: An implementation of the algorithm in Figure 2 using a part-based representation. The algorithm uses variables $\mathbf{s}_i$ for $i = 1 \ldots n$ as a replacement for the dual variables $\mathbf{u}_i$ in Figure 2.

Because of this result, all of the EG algorithms that we have presented can be restated in terms of the **s** variables: instead of maintaining a sequence $\mathbf{u}^t = \{\mathbf{u}_1^t, \mathbf{u}_2^t, \ldots, \mathbf{u}_n^t\}$ of dual variables, a sequence $\mathbf{s}^t = \{\mathbf{s}_1^t, \mathbf{s}_2^t, \ldots, \mathbf{s}_n^t\}$ is maintained and updated using the method described in the above lemmas.[6] To illustrate this, Figure 3 gives a version of the randomized algorithm in Figure 2 that makes use of **s** variables. The batch algorithm can be implemented in a similar way.

6. Note that in the max-margin case, the optimal **u** values may have zero probabilities which correspond to infinite **s** values. This does not pose a problem, since the algorithm will indeed converge to infinite **s** values at the limit, but $\mathbf{s}^t$ will not be infinite for any finite t. For the log-linear case, the optimal **u** will never have zero values, as shown in Globerson et al. (2007).

The main computational challenge in the new algorithms comes in computing the parameter vector $\mathbf{w}(\mathbf{p}(\mathbf{s}^t))$. The value for $\mathbf{w}(\mathbf{p}(\mathbf{s}^t))$ can be expressed as a function of the *marginal probabilities* of the part variables, as follows:

$$\begin{aligned}
\mathbf{w}(\mathbf{p}(\mathbf{s}^t)) &= \sum_i \sum_y u_{i,y} \left(\mathbf{f}(x_i, y_i) - \mathbf{f}(x_i, y)\right) \\
&= \sum_i \mathbf{f}(x_i, y_i) - \sum_{i,y} p_y(\mathbf{s}_i^t) \mathbf{f}(x_i, y) \\
&= \sum_i \mathbf{f}(x_i, y_i) - \sum_{i,y} \sum_{r \in y} p_y(\mathbf{s}_i^t) \mathbf{f}(x_i, r) \\
&= \sum_i \mathbf{f}(x_i, y_i) - \sum_i \sum_{r \in R} \mu_{i,r}(\mathbf{s}_i^t) \mathbf{f}(x_i, r) \ .
\end{aligned}$$

Here the $\mu_{i,r}$ terms correspond to marginals, defined as

$$\mu_{i,r}(\mathbf{s}_i^t) = \sum_{y : r \in y} p_y(\mathbf{s}_i^t) \ .$$

The mapping from parameters $\mathbf{s}_i^t$ to marginals $\mu_{i,r}(\mathbf{s}_i^t)$ can be computed efficiently in several important cases of structured models. For example, in CRFs belief propagation can be used to efficiently calculate the marginal values, assuming that the tree-width of the underlying graph is small. In weighted context-free grammars the inside-outside algorithm can be used to calculate marginals, assuming that the set of parts R corresponds to context-free rule productions. Once marginals are computed, it is straightforward to compute $\mathbf{w}(\mathbf{p}(\mathbf{s}^t))$ and thereby implement the part-based EG algorithms.

5. Convergence Results

In this section, we provide convergence results for the EG batch and online algorithms presented in Section 3. Section 5.1 provides the key results, and the following sections give the proofs and the technical details.

5.1 Main Convergence Results

Our convergence results give bounds on how quickly the error $|Q(\mathbf{u}) - Q(\mathbf{u}^*)|$ decreases with respect to the number of iterations, T, of the algorithms. In all cases we have $|Q(\mathbf{u}) - Q(\mathbf{u}^*)| \to 0$ as $T \to \infty$.

In what follows we use $D[\mathbf{p} \| \mathbf{q}]$ to denote the KL divergence between $\mathbf{p}, \mathbf{q} \in \Delta^n$ (see Section 5.2). We also use $|A|_\infty$ to denote the maximum magnitude element of A (i.e., $|A|_\infty = \max_{(i,y),(j,z)} |A_{(i,y),(j,z)}|$). The first theorem provides results for the EG-batch algorithms, and the second for the randomized online algorithms.

Theorem 1 *For the batch algorithm in Figure 1, for Q_{LL} and Q_{MM},*

$$Q(\mathbf{u}^*) \leq Q(\mathbf{u}^{T+1}) \leq Q(\mathbf{u}^*) + \frac{1}{\eta T} D[\mathbf{u}^* \| \mathbf{u}^1] \ , \tag{6}$$

assuming that the learning rate η satisfies $0 < \eta \leq \frac{1}{1+n|A|_\infty}$ for Q_{LL}, and $0 < \eta \leq \frac{1}{n|A|_\infty}$ for Q_{MM}. Furthermore, for Q_{LL},

$$Q(\mathbf{u}^*) \leq Q(\mathbf{u}^{T+1}) \leq Q(\mathbf{u}^*) + \frac{e^{-\eta T}}{\eta} D[\mathbf{u}^* \| \mathbf{u}^1] \ ,$$

	Batch Algorithm	Online Algorithm
Q_{MM}	$\frac{n^2}{\varepsilon}\|A\|_\infty D[\mathbf{u}^*\|\mathbf{u}^1]$	$\frac{n}{\varepsilon}\left(\|A\|_\infty D[\mathbf{u}^*\|\mathbf{u}^1] + Q(\mathbf{u}^1) - Q(\mathbf{u}^*)\right)$
Q_{LL}	$n(1+n\|A\|_\infty)\log(\frac{c_1}{\varepsilon})$	$n(1+\|A\|_\infty)\log(\frac{c_2}{\varepsilon})$

Table 1: Each entry shows the amount of computation (measured in terms of the number of training sample processed using the EG updates) required to obtain $|Q(\mathbf{u}) - Q(\mathbf{u}^*)| \leq \varepsilon$ for the batch algorithm, or $\mathbf{E}\left[|Q(\mathbf{u}) - Q(\mathbf{u}^*)|\right] \leq \varepsilon$ for the online algorithm, for a given $\varepsilon > 0$. The constants are $c_1 = (1+n|A|_\infty)D[\mathbf{u}^*\|\mathbf{u}^1]$, and $c_2 = \left[(1+|A|_\infty)D[\mathbf{u}^*\|\mathbf{u}^1] + Q(\mathbf{u}^1) - Q(\mathbf{u}^*))\right]$.

assuming again that $0 < \eta \leq \frac{1}{1+n|A|_\infty}$.

The randomized online algorithm will produce different results at every run, since different points will be processed on different runs. Our main result for this algorithm characterizes the mean value of the objective $Q(\mathbf{u}^{T+1})$ when averaged over all possible random orderings of points. The result implies that this mean will converge to the optimal value $Q(\mathbf{u}^*)$.

Theorem 2 *For the randomized algorithm in Figure 2, for* Q_{LL} *and* Q_{MM}*,*

$$Q(\mathbf{u}^*) \leq \mathbf{E}\left[Q(\mathbf{u}^{T+1})\right] \leq Q(\mathbf{u}^*) + \frac{n}{\eta T} D[\mathbf{u}^*\|\mathbf{u}^1] + \frac{n}{T}\left[Q(\mathbf{u}^1) - Q(\mathbf{u}^*)\right] , \tag{7}$$

assuming that the learning rate η *satisfies* $0 < \eta \leq \frac{1}{1+|A|_\infty}$ *for* Q_{LL}*, and* $0 < \eta \leq \frac{1}{|A|_\infty}$ *for* Q_{MM}*. Furthermore, for* Q_{LL}*, for the algorithm in Figure 2,*

$$Q(\mathbf{u}^*) \leq \mathbf{E}\left[Q(\mathbf{u}^{T+1})\right] \leq Q(\mathbf{u}^*) + e^{-\frac{\eta T}{n}}\left[\frac{1}{\eta} D[\mathbf{u}^*\|\mathbf{u}^1] + Q(\mathbf{u}^1) - Q(\mathbf{u}^*)\right] ,$$

assuming again that $0 < \eta \leq \frac{1}{1+|A|_\infty}$.

The above result characterizes the average behavior of the randomized algorithm, but does not provide guarantees for any specific run of the algorithm. However, by applying the standard approach of repeated sampling (see, for example, Mitzenmacher and Upfal, 2005; Shalev-Shwartz et al., 2007), one can obtain a solution that, with high probability, does not deviate by much from the average behavior. In what follows, we briefly outline this derivation.

Note that the random variable $Q(\mathbf{u}^{T+1}) - Q(\mathbf{u}^*)$ is nonnegative, and so by Markov's inequality, it satisfies

$$\Pr\left\{Q(\mathbf{u}^{T+1}) - Q(\mathbf{u}^*) \geq 2\left(\mathbf{E}\left[Q(\mathbf{u}^{T+1})\right] - Q(\mathbf{u}^*)\right)\right\} \leq \frac{1}{2} .$$

Given some $\delta > 0$, if we run the algorithm $k = \log_2(\frac{1}{\delta})$ times,[7] each time with T iterations, and choose the best $\hat{\mathbf{u}}$ of these k results, we see that

$$\Pr\left\{Q(\hat{\mathbf{u}}) - Q(\mathbf{u}^*) \geq 2\left(\mathbf{E}\left[Q(\mathbf{u}^{T+1})\right] - Q(\mathbf{u}^*)\right)\right\} \leq \delta .$$

7. Assume for simplicity that $\log_2(\frac{1}{\delta})$ is integral.

Thus, for any desired confidence $1-\delta$, we can obtain a solution that is within a factor of 2 of the bound for T iterations in Theorem 2 by using $T\log_2(\frac{1}{\delta})$ iterations. In our experiments, we found that repeated trials of the randomized algorithm did not yield significantly different results.

The first consequence of the two theorems above is that the batch and randomized online algorithms converge to a $\mathbf{u}$ with the optimal value $Q(\mathbf{u}^*)$. This follows since Equations 6 and 7 imply that as $T \to \infty$ the value of $Q(\mathbf{u}^{T+1})$ approaches $Q(\mathbf{u}^*)$.

The second consequence is that for a given $\varepsilon > 0$ we can find the number of iterations needed to reach a $\mathbf{u}$ such that $|Q(\mathbf{u}) - Q(\mathbf{u}^*)| \leq \varepsilon$ for the batch algorithm or $\mathbf{E}\left[|Q(\mathbf{u}) - Q(\mathbf{u}^*)|\right] \leq \varepsilon$ for the online algorithm. Table 1 shows the computation required by the different algorithms, where the computation is measured by the number of training examples that need to be processed using the EG updates.[8] The entries in the table assume that the maximum possible learning rates are used for each of the algorithms—that is, $\frac{1}{1+n|A|_\infty}$ for LLEG-Batch, $\frac{1}{1+|A|_\infty}$ for LLEG-Online, $\frac{1}{n|A|_\infty}$ for MMEG-batch, and $\frac{1}{|A|_\infty}$ for MMEG-Online.

Crucially, note that these rates suggest that the online algorithms are significantly more efficient than the batch algorithms; specifically, the bounds suggest that the online algorithms require a factor of n less computation in both the Q_{LL} and Q_{MM} cases. Thus these results suggest that the randomized online algorithm should converge much faster than the batch algorithm. Roughly speaking, this is a direct consequence of the learning rate η being a factor of n larger in the online case (see also Section 9). This prediction is confirmed in our empirical evaluations, which show that the online algorithm is far more efficient than the batch algorithm.

A second important point is that the rates for Q_{LL} lead to an $O(\log(\frac{1}{\varepsilon}))$ dependence on the desired accuracy ε, which is a significant improvement over Q_{MM}, which has an $O(\frac{1}{\varepsilon})$ dependence. Note that the $O(\frac{1}{\varepsilon})$ dependence for Q_{MM} is similar to several other max-margin algorithms in the literature (see Section 6 for more discussion).

To gain further intuition into the order of magnitude of iterations required, note that the factor $D[\mathbf{u}^*\|\mathbf{u}^1]$ which appears in the above expressions is at most $n\log|\mathcal{Y}|$, which can be achieved by setting $\mathbf{u}_i^1$ to be the uniform distribution over $\mathcal{Y}$ for all i. Also, the value of $|A|_\infty$ can easily be seen to be $\frac{1}{C}\max_{i,y}\|\mathbf{g}_{i,y}\|^2$.

In the remainder of this section we give proofs of the results in Theorems 1 and 2. In doing so, we also give theorems that apply to the optimization of general convex functions $Q : \Delta^n \to \mathbb{R}$.

5.2 Divergence Measures

Before providing convergence proofs, we define several divergence measures between distributions. Define the KL divergence between two distributions $\mathbf{u}_i, \mathbf{v}_i \in \Delta$ to be

$$D[\mathbf{u}_i\|\mathbf{v}_i] = \sum_y u_{i,y}\log\frac{u_{i,y}}{v_{i,y}} \ .$$

Given two sets of n distributions $\mathbf{u}, \mathbf{v} \in \Delta^n$ define their KL divergence as

$$D[\mathbf{u}\|\mathbf{v}] = \sum_i D[\mathbf{u}_i\|\mathbf{v}_i] \ .$$

8. Note that if we run the batch algorithm for T iterations (as in the figure), nT training examples are processed. In contrast, running the online algorithm for T iterations (again, as shown in the figure) only requires T training examples to be processed. It is important to take this into account when comparing the rates in Theorems 1 and 2; this is the motivation for measuring computation in terms of the number of examples that are processed.

Next, we consider a more general class of divergence measures, Bregman divergences (e.g., see Bregman, 1967; Censor and Zenios, 1997; Kivinen and Warmuth, 1997). Given a convex function $Q(\mathbf{u})$, the Bregman divergence between $\mathbf{u}$ and $\mathbf{v}$ is defined as

$$B_Q[\mathbf{u}\|\mathbf{v}] = Q(\mathbf{u}) - Q(\mathbf{v}) - \nabla Q(\mathbf{v}) \cdot (\mathbf{u} - \mathbf{v}) \ .$$

Convexity of Q implies $B_Q[\mathbf{u}\|\mathbf{v}] \geq 0$ for all $\mathbf{u}, \mathbf{v} \in \Delta^n$.

Note that the Bregman divergence with $Q(\mathbf{u}) = \sum_{i,y} u_{i,y} \log u_{i,y}$ is the KL divergence. We shall also be interested in the Mahalanobis distance

$$M_A[\mathbf{u}\|\mathbf{v}] = \frac{1}{2}(\mathbf{u} - \mathbf{v})^T A (\mathbf{u} - \mathbf{v}) \ ,$$

which is the Bregman divergence for $Q(\mathbf{u}) = \frac{1}{2}\mathbf{u}^T A \mathbf{u}$.

In what follows, we also use the L_p norm defined for $\mathbf{x} \in \mathbb{R}^m$ as $\|\mathbf{x}\|_p = \sqrt[p]{\sum_i |x_i|^p}$.

5.3 Dual Improvement and Bregman Divergence

In this section we provide a useful lemma that determines when the EG updates in the batch algorithm will result in monotone improvement of $Q(\mathbf{u})$. The lemma requires a condition on the relation between the Bregman and KL divergences which we define as follows (the second part of the definition will be used in the next section):

Definition 5.1 : A convex function $Q : \Delta^n \to \mathbb{R}$ is τ-upper-bounded for some $\tau > 0$ if for any $\mathbf{p}, \mathbf{q} \in \Delta^n$,

$$B_Q[\mathbf{p}\|\mathbf{q}] \leq \tau D[\mathbf{p}\|\mathbf{q}] \ .$$

In addition, we say $Q(\mathbf{u})$ is (μ, τ)-bounded for constants $0 < \mu < \tau$ if for any $\mathbf{p}, \mathbf{q} \in \Delta^n$,

$$\mu D[\mathbf{p}\|\mathbf{q}] \leq B_Q[\mathbf{p}\|\mathbf{q}] \leq \tau D[\mathbf{p}\|\mathbf{q}] \ .$$

The next lemma states that if $Q(\mathbf{u})$ is τ-upper-bounded, then the change in the objective as a result of an EG update can be related to the KL divergence between consecutive values of the dual variables.

Lemma 2 *If $Q(\mathbf{u})$ is τ-upper-bounded, then if η is chosen such that $0 < \eta \leq \frac{1}{\tau}$, it holds that for all t in the batch algorithm (Figure 1):*

$$Q(\mathbf{u}^t) - Q(\mathbf{u}^{t+1}) \geq \frac{1}{\eta} D[\mathbf{u}^t \| \mathbf{u}^{t+1}] \ .$$

Proof: Given a $\mathbf{u}^t$, the EG update is

$$u^{t+1}_{i,y} = \frac{1}{Z^t_i} u^t_{i,y} e^{-\eta \nabla^t_{i,y}} \ ,$$

where

$$\nabla^t_{i,y} = \frac{\partial Q(\mathbf{u}^t)}{\partial u_{i,y}} \ , \qquad Z^t_i = \sum_{\hat{y}} u^t_{i,\hat{y}} e^{-\eta \nabla^t_{i,y}} \ .$$

Simple algebra yields

$$\sum_i \left(D[\mathbf{u}_i^t \| \mathbf{u}_i^{t+1}] + D[\mathbf{u}_i^{t+1} \| \mathbf{u}_i^t] \right) = \eta \sum_{i,y} (u_{i,y}^t - u_{i,y}^{t+1}) \nabla_{i,y}^t \ .$$

Equivalently, using the notation for KL divergence between multiple distributions:

$$D[\mathbf{u}^t \| \mathbf{u}^{t+1}] + D[\mathbf{u}^{t+1} \| \mathbf{u}^t] = \eta (\mathbf{u}^t - \mathbf{u}^{t+1}) \cdot \nabla Q(\mathbf{u}^t) \ .$$

The definition of the Bregman divergence B_Q then implies

$$-\eta B_Q[\mathbf{u}^{t+1} \| \mathbf{u}^t] + D[\mathbf{u}^t \| \mathbf{u}^{t+1}] + D[\mathbf{u}^{t+1} \| \mathbf{u}^t] = \eta (Q(\mathbf{u}^t) - Q(\mathbf{u}^{t+1})) \ . \tag{8}$$

Since $Q(\mathbf{u})$ is τ-upper-bounded and $\eta \leq \frac{1}{\tau}$ it follows that $D[\mathbf{u}^{t+1} \| \mathbf{u}^t] \geq \eta B_Q[\mathbf{u}^{t+1} \| \mathbf{u}^t]$, and together with Eq. 8 we obtain the desired result $\eta(Q(\mathbf{u}^t) - Q(\mathbf{u}^{t+1})) \geq D[\mathbf{u}^t \| \mathbf{u}^{t+1}]$. □

Note that the condition in the lemma may be weakened to requiring that $\tau D[\mathbf{u}^t \| \mathbf{u}^{t+1}] \geq B_Q[\mathbf{u}^t \| \mathbf{u}^{t+1}]$ for all t. For simplicity, we require the condition for all $\mathbf{p}, \mathbf{q} \in \Delta^n$. Note also that $D[\mathbf{p} \| \mathbf{q}] \geq 0$ for all $\mathbf{p}, \mathbf{q} \in \Delta^n$, so the lemma implies that for an appropriately chosen η, the EG updates always decrease the objective $Q(\mathbf{u})$. We next show that the log-linear dual $Q_{\text{LL}}(\mathbf{u})$ is in fact τ-upper-bounded.

Lemma 3 *Define $|A|_\infty$ to be the maximum magnitude of any element of A, that is, $|A|_\infty = \max_{(i,y),(j,z)} |A_{(i,y),(j,z)}|$. Then $Q_{\text{LL}}(\mathbf{u})$ is τ_{LL}-upper-bounded with $\tau_{\text{LL}} = 1 + n|A|_\infty$.*

Proof: First notice that the Bregman divergence B_Q is linear in Q. Thus, for any $\mathbf{p}, \mathbf{q} \in \Delta^n$ we can write $B_{Q_{\text{LL}}}$ as a sum of two terms (see Eq. 4).

$$B_{Q_{\text{LL}}}[\mathbf{p} \| \mathbf{q}] = D[\mathbf{p} \| \mathbf{q}] + M_A[\mathbf{p} \| \mathbf{q}] \ .$$

We first bound $M_A[\mathbf{p} \| \mathbf{q}]$ in terms of squared L_1 distance between $\mathbf{p}$ and $\mathbf{q}$. Denote $\mathbf{r} = \mathbf{p} - \mathbf{q}$. Then:

$$M_A[\mathbf{p} \| \mathbf{q}] = \frac{1}{2} \sum_{i,y,j,z} r_{i,y} r_{j,z} A_{(i,y),(j,z)} \leq \frac{|A|_\infty}{2} \sum_{i,y,j,z} |r_{i,y}| |r_{j,z}| = \frac{|A|_\infty}{2} \|\mathbf{p} - \mathbf{q}\|_1^2 \ .$$

Next, we use the inequality $D[p_1 \| p_2] \geq \frac{1}{2} \|p_1 - p_2\|_1^2$ (also known as Pinsker's inequality, see Cover and Thomas, 1991, p. 300), which holds for any two distributions p_1 and p_2. Consider the two distributions $\hat{\mathbf{p}} = \frac{1}{n}\mathbf{p}$ and $\hat{\mathbf{q}} = \frac{1}{n}\mathbf{q}$, each defined over an alphabet of size $n|\mathcal{Y}|$. Then it follows that:[9]

$$\frac{|A|_\infty}{2} \|\mathbf{p} - \mathbf{q}\|_1^2 = \frac{n^2 |A|_\infty}{2} \|\hat{\mathbf{p}} - \hat{\mathbf{q}}\|_1^2 \leq n^2 |A|_\infty D[\hat{\mathbf{p}} \| \hat{\mathbf{q}}] = n|A|_\infty D[\mathbf{p} \| \mathbf{q}] \ ,$$

and thus $M_A[\mathbf{p} \| \mathbf{q}] \leq n|A|_\infty D[\mathbf{p} \| \mathbf{q}]$. So for the Bregman divergence of $Q_{\text{LL}}(\mathbf{u})$ we obtain

$$B_{Q_{\text{LL}}}[\mathbf{p} \| \mathbf{q}] \leq (1 + n|A|_\infty) D[\mathbf{p} \| \mathbf{q}] \ ,$$

yielding the desired result. □

The next lemma shows that a similar result can be obtained for the Q_{MM} objective.

9. Note that $D[\hat{\mathbf{p}} \| \hat{\mathbf{q}}]$ is a divergence between two distributions over $|\mathcal{Y}|n$ symbols and $D[\mathbf{p} \| \mathbf{q}]$ is a divergence between two sets of n distributions over $|\mathcal{Y}|$ symbols.

Lemma 4 *The function $Q_{\text{MM}}(\mathbf{u})$ is τ_{MM}-upper-bounded with $\tau_{\text{MM}} = n|A|_\infty$.*

Proof: For Q_{MM} defined in Eq. 5, we have

$$B_{Q_{\text{MM}}}[\mathbf{p}\|\mathbf{q}] = M_A[\mathbf{p}\|\mathbf{q}] \ .$$

We can then use a similar derivation to that of Lemma 3 to obtain the result. □

We thus have that the condition in Lemma 2 is satisfied for both the $Q_{\text{LL}}(\mathbf{u})$ and $Q_{\text{MM}}(\mathbf{u})$ objectives, implying that their EG updates result in monotone improvement of the objective, for a suitably chosen η:

Corollary 1 *The LLEG-Batch algorithm with $0 < \eta \leq \frac{1}{\tau_{\text{LL}}}$ satisfies for all t*

$$Q_{\text{LL}}(\mathbf{u}^t) - Q_{\text{LL}}(\mathbf{u}^{t+1}) \geq \frac{1}{\eta} D[\mathbf{u}^t \| \mathbf{u}^{t+1}] \ ,$$

and the MMEG-Batch algorithm with $0 < \eta \leq \frac{1}{\tau_{\text{MM}}}$ satisfies for all t

$$Q_{\text{MM}}(\mathbf{u}^t) - Q_{\text{MM}}(\mathbf{u}^{t+1}) \geq \frac{1}{\eta} D[\mathbf{u}^t \| \mathbf{u}^{t+1}] \ .$$

5.4 Convergence Rates for the EG Batch Algorithms

The previous section showed that for appropriate choices of the learning rate η, the batch EG updates are guaranteed to improve the Q_{LL} and Q_{MM} loss functions at each iteration. In this section we build directly on these results, and address the following question: how many iterations does the batch EG algorithm require so that the $|Q(\mathbf{u}^t) - Q(\mathbf{u})| \leq \varepsilon$ for a given $\varepsilon > 0$? We show that as long as $Q(\mathbf{u})$ is τ-upper-bounded, the number of iterations required is $O(\frac{1}{\varepsilon})$. This bound thus holds for both the log-linear and max-margin batch algorithms. Next, we show that if $Q(\mathbf{u})$ is (μ, τ)-bounded, the rate can be significantly improved to requiring $O(\log(\frac{1}{\varepsilon}))$ iterations. We conclude by showing that $Q_{\text{LL}}(\mathbf{u})$ is (μ, τ)-bounded, implying that the $O(\log(\frac{1}{\varepsilon}))$ rate holds for LLEG-Batch.

The following result gives an $O(\frac{1}{\varepsilon})$ rate for Q_{LL} and Q_{MM}:

Lemma 5 *If $Q(\mathbf{u})$ is τ-upper-bounded and $0 \leq \eta \leq \frac{1}{\tau}$, then after T iterations of the EG-Batch algorithm, for any $\mathbf{z} \in \Delta^n$ including $\mathbf{z} = \mathbf{u}^*$,*

$$Q(\mathbf{u}^{T+1}) - Q(\mathbf{z}) \leq \frac{1}{\eta T} D[\mathbf{z}\|\mathbf{u}^1] \ .$$

Proof: See Appendix A. □

The lemma implies that to get ε-close to the optimal objective value, $O(\frac{1}{\varepsilon})$ iterations are required—more precisely, if a choice of $\eta = \frac{1}{\tau}$ is made, then at most $\frac{\tau}{\varepsilon} D[\mathbf{z}\|\mathbf{u}^1]$ iterations are required. Since its conditions are satisfied by both $Q_{\text{LL}}(\mathbf{u})$ and $Q_{\text{MM}}(\mathbf{u})$ (given an appropriate choice of η) the result holds for both the LLEG-Batch and MMEG-Batch algorithms.

A much improved rate may be obtained if $Q(\mathbf{u})$ is not only τ-upper-bounded, but also (μ, τ)-bounded (see Definition 5.1).

Lemma 6 *If $Q(\mathbf{u})$ is (μ,τ)-bounded and $0 < \eta \leq \frac{1}{\tau}$ then after T iterations of the EG-Batch algorithm, for any $\mathbf{z} \in \Delta^n$ including $\mathbf{z} = \mathbf{u}^*$,*

$$Q(\mathbf{u}^{T+1}) - Q(\mathbf{z}) \leq \frac{e^{-\eta\mu T}}{\eta} D[\mathbf{z}||\mathbf{u}^1] \ .$$

Proof: See Appendix B. □

The lemma implies that an accuracy of ε may be achieved by using $O(\log(\frac{1}{\varepsilon}))$ iterations.

To see why $Q_{\text{LL}}(\mathbf{u})$ is (μ,τ)-bounded note that for any $\mathbf{p},\mathbf{q} \in \Delta^n$,

$$B_{Q_{\text{LL}}}[\mathbf{p}\|\mathbf{q}] = D[\mathbf{p}\|\mathbf{q}] + M_A[\mathbf{p}\|\mathbf{q}] \geq D[\mathbf{p}\|\mathbf{q}] \ ,$$

implying (together with Lemma 3) that $Q_{\text{LL}}(\mathbf{u})$ is $(1,\tau_{\text{LL}})$-bounded.

Finally, note that Lemmas 5 and 6, together with the facts that Q_{LL} is $(1,\tau_{\text{LL}})$-bounded and Q_{MM} is τ_{MM}-upper-bounded, imply Theorem 1 of Section 5.1.

5.5 Convergence Results for the Randomized Online Algorithm

This section analyzes the rate of convergence of the randomized online algorithm in Figure 2. Before stating the results, we need some definitions. We will use $Q_{\mathbf{u},i} : \Delta \rightarrow \mathbb{R}$ to be the function defined as

$$Q_{\mathbf{u},i}(\mathbf{v}) = Q(\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_{i-1}, \mathbf{v}, \mathbf{u}_{i+1}, \ldots, \mathbf{u}_n) \ ,$$

for any $\mathbf{v} \in \Delta$. We denote the Bregman divergence associated with $Q_{\mathbf{u},i}$ as $B_{Q_{\mathbf{u},i}}[\mathbf{x}\|\mathbf{y}]$. We then introduce the following definitions:

Definition 5.2: *A convex function $Q : \Delta^n \rightarrow \mathbb{R}$ is τ-online-upper-bounded for some $\tau > 0$ if for any $i \in 1 \ldots n$ and for any $\mathbf{p},\mathbf{q} \in \Delta$,*

$$B_{Q_{\mathbf{u},i}}[\mathbf{p}\|\mathbf{q}] \leq \tau D[\mathbf{p}\|\mathbf{q}] \ .$$

In addition, Q is (μ,τ)-online-bounded for $0 < \mu < \tau$ if Q is τ-online-upper-bounded, and in addition, for any $\mathbf{p},\mathbf{q} \in \Delta^n$,

$$\mu D[\mathbf{p}\|\mathbf{q}] \leq B_Q[\mathbf{p}\|\mathbf{q}] \ .$$

Note that the lower bound in the above definition refers to B_Q and not to $B_{Q_{\mathbf{u},i}}$. Also, note that if a function is (μ,τ)-online-bounded then it must also be τ-online-upper-bounded.

The following lemma then gives results for the Q_{LL} and Q_{MM} functions:

Lemma 7 *The log-linear dual $Q_{\text{LL}}(\mathbf{u})$ is (μ,τ)-online-bounded for $\mu = 1$ and $\tau = 1 + |A|_\infty$. The max-margin dual $Q_{\text{MM}}(\mathbf{u})$ is τ-online-upper-bounded for $\tau = |A|_\infty$.*

Proof: See Appendix C. □

For any τ-online-upper-bounded Q, the online EG algorithm converges at an $O(\frac{1}{\varepsilon})$ rate, as shown by the following lemma.

Lemma 8 *Consider the algorithm in Figure 2 applied to a convex function $Q(\mathbf{u})$ that is τ-online-upper-bounded. If $\eta > 0$ is chosen such that $\eta \leq \frac{1}{\tau}$, then it follows that for all $\mathbf{z} \in \Delta^n$*

$$\mathbf{E}\left[Q(\mathbf{u}^{T+1})\right] \leq Q(\mathbf{z}) + \frac{n}{\eta T} D[\mathbf{z}\|\mathbf{u}^1] + \frac{n}{T}\left[Q(\mathbf{u}^1) - Q(\mathbf{u}^*)\right] ,$$

where $\mathbf{E}\left[Q(\mathbf{u}^{T+1})\right]$ is the expected value of $Q(\mathbf{u}^{T+1})$, and $\mathbf{u}^* = \operatorname{argmin}_{\mathbf{u}\in\Delta^n} Q(\mathbf{u})$.

Proof: See Appendix D. □

The previous lemma shows, in particular, that the online EG algorithm converges at an $O(\frac{1}{\varepsilon})$ rate for the function Q_{MM}. However, we can prove a faster rate of convergence for Q_{LL}, which is (μ,τ)-online-bounded. The following lemma shows that such functions exhibit an $O(\log(\frac{1}{\varepsilon}))$ rate of convergence.

Lemma 9 *Consider the algorithm in Figure 2 applied to a convex function $Q(\mathbf{u})$ that is (μ,τ)-online-bounded. If $\eta > 0$ is chosen such that $\eta \leq \frac{1}{\tau}$, then it follows that for all $\mathbf{z} \in \Delta^n$*

$$\mathbf{E}\left[Q(\mathbf{u}^{T+1})\right] \leq Q(\mathbf{z}) + e^{-\frac{\eta\mu T}{n}}\left[\frac{1}{\eta} D[\mathbf{z}\|\mathbf{u}^1] + Q(\mathbf{u}^1) - Q(\mathbf{u}^*)\right] ,$$

where $\mathbf{u}^* = \operatorname{argmin}_{\mathbf{u}\in\Delta^n} Q(\mathbf{u})$.

Proof: See Appendix E. □

Note that Lemmas 7, 9 and 8 complete the proof of Theorem 2 in Section 5.1.

6. Related Work

The idea of solving regularized loss-minimization problems via their convex duals has been addressed in several previous papers. Here we review those, specifically focusing on the log-linear and max-margin problems.

Zhang (2002) presented a study of convex duals of general regularized loss functions, and provided a methodology for deriving such duals. He also considered a general procedure for solving such duals by optimizing one coordinate at a time. However, it is not clear how to implement this procedure in the structured learning case (i.e., when $|\mathcal{Y}|$ is large), and convergence rates are not given.

In the specific context of log-linear models, several papers have addressed dual optimization. Earlier work (Jaakkola and Haussler, 1999; Keerthi et al., 2005; Zhu and Hastie, 2001) treated the logistic regression model, a simpler version of a CRF. In the binary logistic regression case, there is essentially one parameter u_i per example with the constraint $0 \leq u_i \leq 1$, and therefore simple line-search methods can be used for optimization. Minka (2003) presents empirical results which show that this approach performs similarly to conjugate gradient. The problem becomes much harder when $\mathbf{u}_i$ is constrained to be a distribution over many labels, as in the case discussed here. Recently, Memisevic (2006) addressed this setting, and suggests optimizing $\mathbf{u}_i$ by transferring probability mass between two labels y_1, y_2 while keeping the distribution normalized. This requires a strategy for choosing these two labels, and the author suggests one which seems to perform well.

While some previous work on log-linear models proved convergence of dual methods (e.g., Keerthi et al., 2005), we are not aware of rates of convergence that have been reported in this context. Convergence rates for related algorithms, in particular a generalization of EG, known as the Mirror-Descent algorithm, have been studied in a more general context in the optimization literature. For instance, Beck and Teboulle (2003) describe convergence results which apply to quite general definitions of $Q(\mathbf{u})$, but which have only $O(\frac{1}{\varepsilon^2})$ convergence rates, as compared to our results of $O(\frac{1}{\varepsilon})$ and $O(\log(\frac{1}{\varepsilon}))$ for the max-margin and log-linear cases respectively. Also, their work considers optimization over a single simplex, and does not consider online-like algorithms such as the one we have presented.

For max-margin models, numerous dual methods have been suggested, an earlier example being the SMO algorithm of Platt (1998). Such methods optimize subsets of the $\mathbf{u}$ parameters in the dual SVM formulation (see also Crammer and Singer, 2002). Analysis of a similar algorithm (Hush et al., 2006) results in an $O(\frac{1}{\varepsilon})$ rate, similar to the one we have here. Another algorithm for solving SVMs via the dual is the multiplicative update method of Sha et al. (2007). These updates are shown to converge to the optimum of the SVM dual, but convergence rate has not been analyzed, and extension to the structured case seems non-trivial. An application of EG to binary SVMs was previously studied by Cristianini et al. (1998). They show convergence rates of $O(\frac{1}{\varepsilon^2})$, that are slower than our $O(\frac{1}{\varepsilon})$, and no extension to structured learning (or multi-class) is discussed.

Recently, several new algorithms have been presented, along with a rate of convergence analysis (Joachims, 2006; Shalev-Shwartz et al., 2007; Teo et al., 2007; Tsochantaridis et al., 2004; Taskar et al., 2006). All of these algorithms are similar to ours in having a relatively low dependence on n in terms of memory and computation. Among these, Shalev-Shwartz et al. (2007), Teo et al. (2007) and Taskar et al. (2006) present an $O(\frac{1}{\varepsilon})$ rate, but where accuracy is measured in the primal or via the duality gap, and not in the dual as in our analysis. Thus, it seems that a rate of $O(\frac{1}{\varepsilon})$ is currently the best known result for algorithms that have a relatively low dependence on n (general QP solvers, which may have $O(\log(\frac{1}{\varepsilon}))$ behavior, generally have a larger dependence on n, both in time and space). Note that, as in our analysis, all these convergence rates depend on $|A|_\infty$.

Finally, we emphasize again that the EG algorithm is substantially different from stochastic gradient and stochastic subgradient approaches (LeCun et al., 1998; Nedic and Bertsekas, 2001; Shalev-Shwartz et al., 2007; Vishwanathan et al., 2006). EG and stochastic gradient methods are similar in that they both process a single training example at a time. However, EG corresponds to block-coordinate descent in the dual, and uses the *exact* gradient with respect to the block being updated. In contrast, stochastic gradient methods directly optimize the primal problem, and at each update use a single example to *approximate* the gradient (or subgradient) of the primal objective function.

7. Experiments on Regularized Log-Likelihood

In this section we analyze the performance of the EG algorithms for optimization of regularized log-likelihood. We describe experiments on two tasks: first, the MNIST digit classification task, which is a multiclass classification task; second, a log-linear model for a structured natural-language dependency-parsing task. In each case we first give results for the EG method, and then compare

its performance to L-BFGS (Byrd et al., 1995), which is a batch gradient descent method, and to stochastic gradient descent.[10]

We do not report results on LLEG-Batch, since we found it to converge much more slowly than the online algorithm. This is expected from our theoretical results, which anticipate a factor of n speed-up for the online algorithm. We also report experiments comparing the randomized online algorithm to a deterministic online EG algorithm, where samples are drawn in a fixed order (e.g., the algorithm first visits the first example, then the second, etc.).

Although EG is guaranteed to converge for an appropriately chosen η, it turns out to be beneficial to use an adaptive learning rate. Here we use the following simple strategy: we first consider only 10% of the data-set, and find a value of η that results in monotone improvement for at least 95% of the samples. Denote this value by η^{ini} (for the experiments in Section 7.1 we simply use $\eta^{\text{ini}} = 0.5$). For learning over the entire data-set, we keep a learning rate η_i for each sample i (where $i = 1,\ldots,n$), and initialize this rate to η^{ini} for all points. When sample i is visited, we halve η_i until an improvement in the objective is obtained. Finally, after the update, we multiply η_i by 1.05, so that it does not decrease monotonically.

It is important that when updating a single example using the online algorithms, the improvement (or decrease) in the dual can be easily evaluated, allowing the halving strategy described in the previous paragraph to be implemented efficiently. If the current dual parameters are $\mathbf{u}$, the i'th coordinate is selected, and the EG updates then map $\mathbf{u}_i$ to $\mathbf{u}'_i$, the change in the dual objective is

$$\sum_y u'_{i,y} \log u'_{i,y} + \frac{1}{2C} \left\| \mathbf{w}(\mathbf{u}) + \sum_y \left(u'_{i,y} - u_{i,y}\right) \mathbf{g}_{i,y} \right\|^2 - \sum_y u_{i,y} \log u_{i,y} - \frac{1}{2C} \|\mathbf{w}(\mathbf{u})\|^2 \ .$$

The primal parameters $\mathbf{w}(\mathbf{u})$ are maintained throughout the algorithm (see Figure 3), so that this change in the dual objective can be calculated efficiently. A similar method can be used to calculate the change in the dual objective in the max-margin case.

We measure the performance of each training algorithm (the EG algorithms, as well as the batch gradient and stochastic gradient methods) as a function of the amount of computation spent. Specifically, we measure computation in terms of the number of times each training example is visited. For EG, an example is considered to be visited for every value of η that is tested on it. For L-BFGS, all examples are visited for every evaluation performed by the line-search routine. We define the measure of *effective iterations* to be the number of examples visited, divided by n. In the following sections we compare the algorithms in terms of their performance as a function of the effective number of iterations. A comparison in terms of running time is provided in Appendix F; there is little difference between the timed comparisons and the results presented in this section.

7.1 Multi-class Classification

We first conducted multi-class classification experiments on the MNIST classification task. Examples in this data set are images of handwritten digits represented as 784-dimensional vectors. We used a training set of 59k examples, and a validation set of 10k examples.[11] Note that since we

10. We also experimented with conjugate gradient algorithms, but since these resulted in worse performance than L-BFGS, we do not report these results here.

11. In reporting results, we consider only *validation* error; that is, error computed during the training process on a validation set. This measure is often used in early-stopping of algorithms, and is therefore of interest in the current context. We do not report test error since our main focus is algorithmic.

only use a linear kernel, accuracy results are not competitive with state of the art classifiers which use non-linear kernels (e.g., see Cortes and Vapnik, 1995). We define a ten-class logistic-regression model where

$$p(y|x) \propto e^{\mathbf{x}\cdot\mathbf{w}_y} ,$$

and $\mathbf{x}, \mathbf{w}_y \in \mathbb{R}^{784}, y \in \{1, \ldots, 10\}$.

Models were trained for various values of the regularization parameter C: specifically, we tried values of C equal to 1000, 100, 10, 1, 0.1, and 0.01. Convergence of the EG algorithm for low values of C (i.e., 0.1 and 0.01) was found to be slow; we discuss this issue more in Section 7.1.1, arguing that it is not a serious problem.

Figure 4 shows plots of the validation error versus computation for C equal to 1000, 100, 10, and 1, when using the EG algorithm. For C equal to 10 or more, convergence is fast. For $C = 1$ convergence is somewhat slower. Note that there is little to choose between $C = 10$ and $C = 1$ in terms of validation error.

Figure 5 shows plots of the primal and dual objective functions for different values of C. To obtain the primal objective values, we used the EG weight vector $\frac{1}{C}\mathbf{w}(\mathbf{u}^t)$. Note that EG does not explicitly minimize the primal objective function, so the EG primal will not necessarily decrease at every iteration. Nevertheless, our experiments show that the EG primal decreases quite quickly. Figure 6 shows how the duality gap decreases with the amount of computation spent (the duality gap is the difference between the primal and dual values at each iteration). The log of the duality gap decreases more-or-less linearly with the amount of computation spent, as predicted by the $O(\log(\frac{1}{\varepsilon}))$ bounds on the rate of convergence.[12]

Finally, we compare the deterministic and randomized versions of the EG algorithm. Figure 7 shows the primal and dual objectives for both algorithms. It can be seen that the randomized algorithm is clearly much faster to converge. This is even more evident when plotting the duality gap, which converges much faster to zero in the case of the randomized algorithm. These results give empirical evidence that the randomized strategy is to be preferred over a fixed ordering of the training examples (note that we have been able to prove bounds on convergence rate for the randomized algorithm, but have not been able to prove similar bounds for the deterministic case).

7.1.1 CONVERGENCE FOR LOW VALUES OF C

As mentioned in the previous section, convergence of the EG algorithm for low values of C can be very slow. This is to be expected from the bounds on convergence, which predict that convergence time should scale linearly with $\frac{1}{C}$ (other algorithms, e.g., see Shalev-Shwartz et al., 2007, also require $O(\frac{1}{C})$ time for convergence). This is however, not a serious problem on the MNIST data, where validation error has reached a minimum point for around $C = 10$ or $C = 1$.

If convergence for small values of C is required, one strategy we have found effective is to start C at a higher value, then "anneal" it towards the target value. For example, see Figure 8 for results for $C = 1$ using one such annealing scheme. For this experiment, if we take t to be the number of iterations over the training set, where for any t we have processed $t \times n$ training examples, we set $C = 10$ for $t \leq 5$, and set $C = 1 + 9 \times 0.7^{t-5}$ for $t > 5$. Thus C starts at 10, then decays exponentially quickly towards the target value of 1. It can be seen that convergence is significantly faster for the annealed method. The intuition behind this method is that the solution to the dual problem for

12. The rate results presented in this paper are for dual accuracy, but it is straightforward to obtain an $O(\log(\frac{1}{\varepsilon}))$ for the duality gap in the log-linear case.

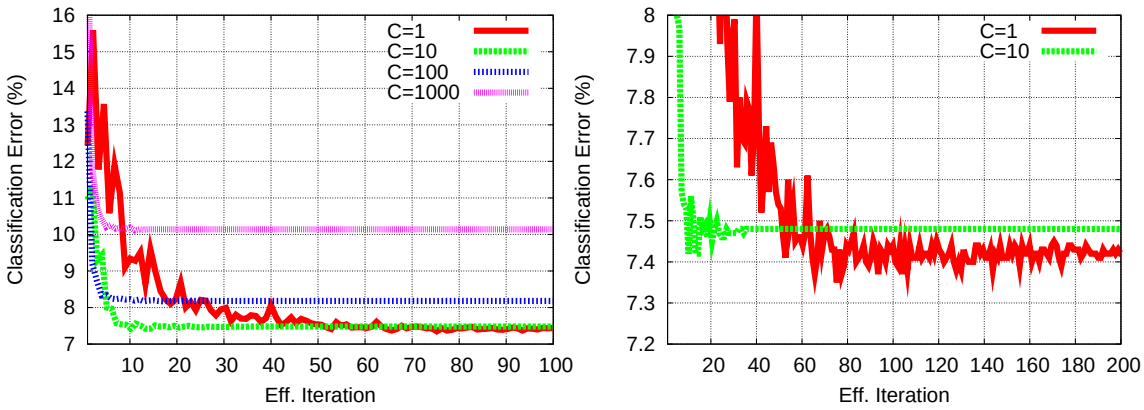

Figure 4: Validation error results on the MNIST learning task for log-linear models trained using the EG randomized online algorithm. The X axis shows the number of effective iterations over the entire data set. The Y axis shows validation error percentages. The left figure shows plots for values of C equal to 1, 10, 100, and 1000. The right figure shows plots for C equal to 1 and 10 at a larger scale.

$C = 10$ is a reasonable approximation to the solution for $C = 1$, and is considerably easier to solve; in the annealing strategy we start with an easier problem and then gradually move towards the harder problem of $C = 1$.

7.1.2 AN EFFICIENT METHOD FOR OPTIMIZING A RANGE OF C VALUES

In practice, when estimating parameters using either regularized log-likelihood or hinge-loss, a range of values for C are tested, with cross-validation or validation on a held-out set being used to choose the optimal value of C. In the previously described experiments, we independently optimized log-likelihood-based models for different values of C. In this section we describe a highly efficient method for training a sequence of models for a range of values of C.

The method is as follows. We pick some maximum value for C; as in our previous experiments, we will choose a maximum value of $C = 1000$. We also pick a tolerance value ε, and a parameter $0 < \nu < 1$. We then optimize C using the randomized online algorithm, until the duality gap is less than $\varepsilon \times p$, where p is the primal value. Once the duality gap has converged to within this ε tolerance, we reduce C by a factor of ν, and again optimize to within an ε tolerance. We continue this strategy—for each value of C optimizing to within a factor of ε, then reducing C by a factor of ν—until C has reached a low enough value. At the end of the sequence, this method recovers a series of models for different values of C, each optimized to within a tolerance of ε.

It is crucial that each time we decrease C, we take our initial dual values to be the final dual values resulting from optimization for the previous value of C. In practice, if C does not decrease too quickly, the previous dual values are a very good starting point for the new value of C; this corresponds to a "warm start" in optimizing values of C that are less than the maximum value. A

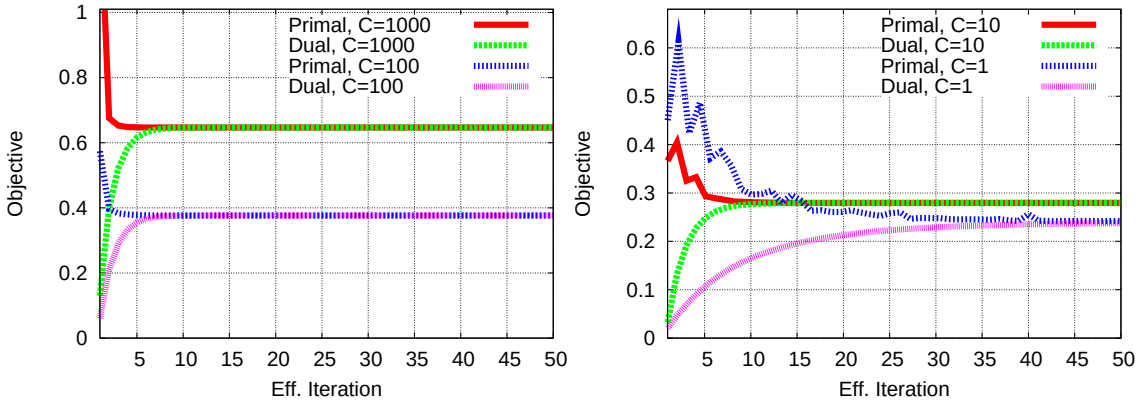

Figure 5: Primal and dual objective values on the MNIST learning task for log-linear models trained using the EG randomized online algorithm. The dual values have been negated so that the primal and dual problems have the same optimal value. The X axis shows the number of effective iterations over the entire data set. The Y axis shows the value of the primal or dual objective functions. The left figure shows plots for values of C equal to 1000 and 100; the right figure shows plots for C equal to 10, and 1. In all cases the primal and dual objectives converge to the same value, with faster convergence for larger values of C.

similar initialization method is used in Koh et al. (2007) in the context of ℓ_1 regularized logistic regression.

As one example of this approach, we trained models in this way with the starting (maximum) value of C set to 1000, ε set to 0.001 (i.e., 0.1%), and v set to 0.7. Table 2 shows the number of iterations of training required for each value of C. The benefits of using the previous dual values at each new value of C are clear: for $13.84 \leq C \leq 700$ at most 5 iterations are required for convergence; even for $C = 0.798$ only 15.24 iterations are required; a range of 25 different values of C between 1000 and 0.274 can be optimized with 211.17 effective iterations over the training set.

7.1.3 COMPARISONS TO STOCHASTIC GRADIENT DESCENT

This section compares performance of the EG algorithms to stochastic gradient descent (SGD) on the primal objective. In SGD the parameters $\mathbf{w}$ are initially set to be 0. At each step an example index i is chosen at random, and the following update is performed:

$$\mathbf{w} = \mathbf{w} - \eta \frac{\partial}{\partial \mathbf{w}} \left(-\log p(y_i \,|\, x_i; \mathbf{w}) + \frac{C}{2n} \|\mathbf{w}\|^2 \right) ,$$

where $\eta > 0$ is a learning rate. The term

$$\frac{\partial}{\partial \mathbf{w}} \left(-\log p(y_i \,|\, x_i; \mathbf{w}) + \frac{C}{2n} \|\mathbf{w}\|^2 \right) ,$$

C	Iterations	Total Iterations	Error
1000	11	11	0.1011
700	3	14	0.0968
490	4.01	18.01	0.0926
343	4.09	22.1	0.0895
240.1	4.24	26.34	0.0869
168.07	4.32	30.67	0.0846
117.649	4.3	34.97	0.0829
82.3543	4.29	39.27	0.0809
57.648	4.32	43.6	0.0803
40.3536	4.33	47.93	0.0775
28.2475	4.34	52.28	0.0768
19.7733	4.36	56.64	0.0758
13.8413	4.38	61.03	0.076
9.6889	5.47	66.51	0.0744
6.78223	5.49	72	0.0741
4.74756	5.51	77.52	0.0732
3.32329	6.6	84.12	0.0736
2.32631	7.69	91.82	0.0735
1.62841	8.78	100.61	0.0729
1.13989	12	112.62	0.074
0.797923	15.24	127.86	0.0747
0.558546	20.61	148.47	0.0749
0.390982	27.05	175.53	0.074
0.273687	35.63	211.17	0.0747

Table 2: Table showing number of effective iterations required to optimize a sequence of values for C for the MNIST task, using the method described in Section 7.1.2. The column C shows the sequence of decreasing regularizer constants. *Iterations* shows the number of effective iterations over the training set required to optimized each value of C. *Total iterations* shows the cumulative value of *Iterations*, and *Error* shows the validation error obtained for every C value. It can be seen that the optimal error is reached at $C = 1.62841$.

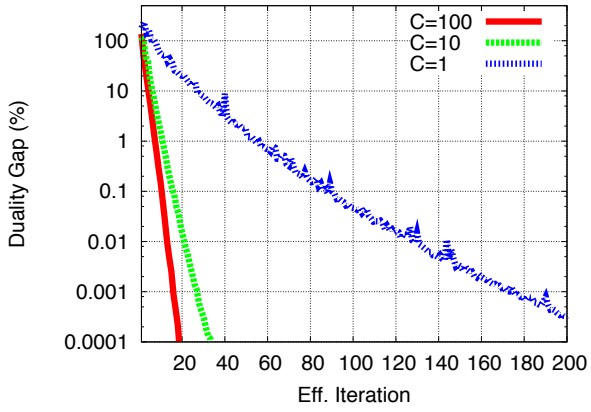

Figure 6: Graph showing the duality gap on the MNIST learning task for log-linear models trained using the EG randomized online algorithm. The X axis shows the number of effective iterations over the entire data set. The Y axis (with a log scale) shows the value of the duality gap, as a percentage of the final optimal value.

can be thought of as an estimate of the gradient of the primal objective function for the entire training set.

In our experiments, we chose the learning rate η to be

$$\eta = \frac{\eta_0}{1 + k/n} ,$$

where $\eta_0 > 0$ is a constant, n is the number of training examples, and k is the number of updates that have been performed up to this point. Thus the learning rate decays to 0 with the number of examples that are updated. This follows the approach described in LeCun et al. (1998); we have consistently found that it performs better than using a single, fixed learning rate.

We tested SGD for C values of 1000, 100, 10, 1, 0.1 and 0.01. In each case we chose the value of η_0 as follows. For each value of C we first tested values of η_0 equal to 1, 0.1, 0.01, 0.001, and 0.0001, and then chose the value of η_0 which led to the best validation error after a single iteration of SGD. This strategy resulted in a choice of $\eta_0 = 0.01$ for all values of C except $C = 1000$, where $\eta_0 = 0.001$ was chosen. We have found this strategy to be a robust method for choosing η_0 (note that we do not want to run SGD for more than one iteration with all (C, η_0) combinations, since each iteration is costly).

Figure 9 compares validation error rates for SGD and the randomized EG algorithm. For the initial (roughly 5) iterations of training, SGD has better validation error scores, but beyond this the EG algorithm is very competitive on this task. Note that the amount of computation for SGD does not include the iterations required to find the optimal value of η_0; if this computation was included the SGD curves would be shifted 5 iterations to the right.

Figure 10 shows graphs comparing the primal objective value for EG and SGD. For C equal to 1000, 100, and 10, the results are similar: SGD is initially better than EG, but after around 5

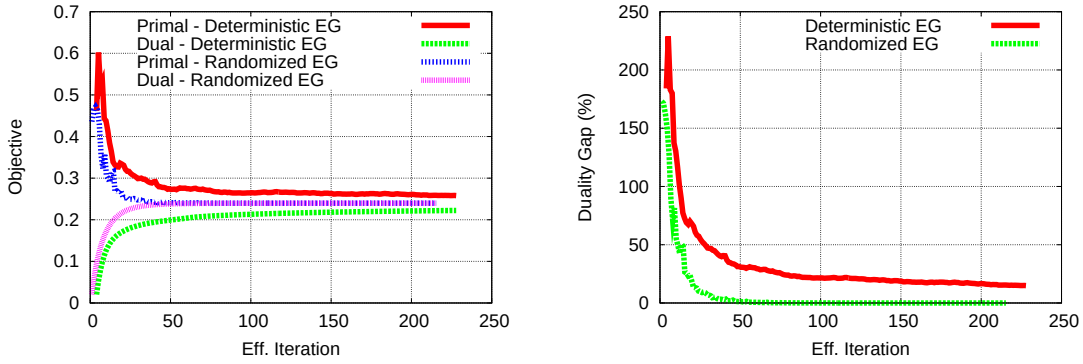

Figure 7: Results on the MNIST learning task, comparing the randomized and deterministic online EG algorithms, for $C = 1$. The left figure shows primal and dual objective values for both algorithms. The right figure shows the normalized value of the duality gap: $(\text{primal(t)} - \text{dual(t)})/\text{opt}$, where opt is the value of the joint optimum of the primal and dual problems, and t is the iteration number. The X axis counts the number of effective iterations over the entire data set.

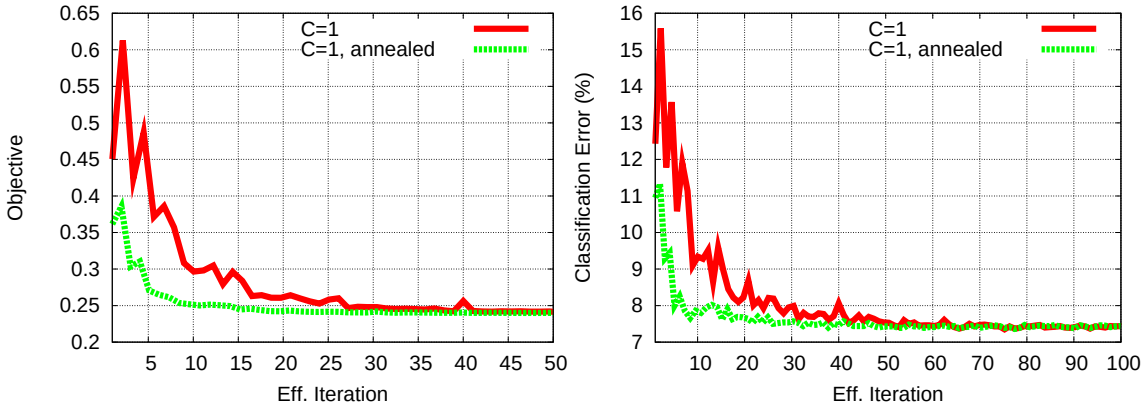

Figure 8: Results on the MNIST learning task, for $C = 1$, comparing the regular EG randomized algorithm with an annealed version of the algorithm (see Section 7.1.1). The left figure shows primal objective values calculated for $C = 1$; the right figure shows validation error. The annealed strategy gives significantly faster convergence.

iterations EG overtakes SGD, and converges much more quickly to the optimal point. The difference between EG and SGD appears to become more pronounced as C becomes smaller. For $C = 1$ our

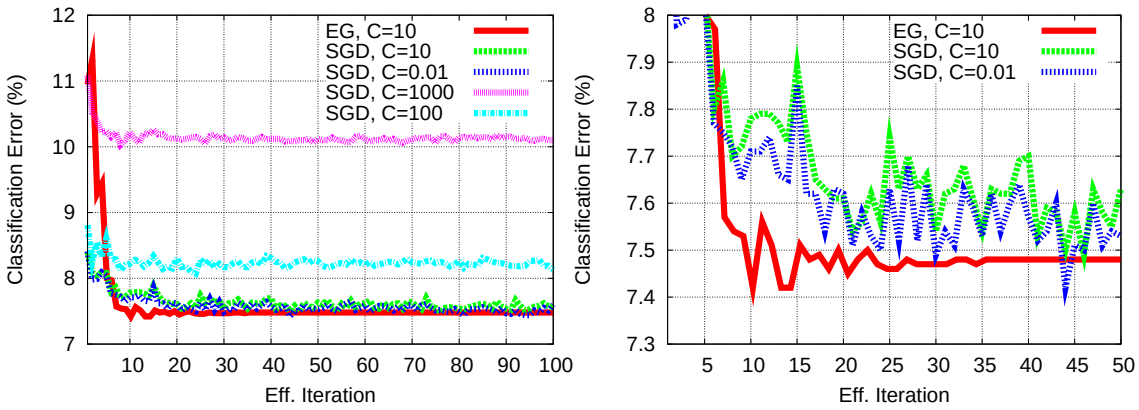

Figure 9: Graphs showing validation error results on the MNIST learning task, comparing the EG randomized algorithm to stochastic gradient descent (SGD). The X axis shows number of effective training iterations, the Y axis shows validation error in percent. The EG results are shown for $C = 10$; SGD results are shown for several values of C. For SGD for $C = 1$, $C = 0.1$, and $C = 0.01$ the curves were nearly identical, hence we omit the curves for $C = 1$ and $C = 0.1$. Note that the amount of computation for SGD does not include the iterations required to find the optimal value for the learning rate η_0.

strategy for choosing η_0 does not pick the optimal value for η_0 at least when evaluating the primal objective; see the caption to the figure for more discussion. EG again appears to out-perform SGD after the initial few iterations.

7.1.4 COMPARISONS TO L-BFGS

One of the standard approaches to training log-linear models is using the L-BFGS gradient-based algorithm (Sha and Pereira, 2003). L-BFGS is a batch algorithm, in the sense that its updates require evaluating the primal objective and gradient, which involves iterating over the entire data-set. To compare L-BFGS to EG, we used the implementation based on Byrd et al. (1995).[13]

For L-BFGS, a total of n training examples must be processed every time the gradient or objective function is evaluated; note that because L-BFGS uses a line search, each iteration may involve several such evaluations.[14]

13. Specifically, we used the code by Zhu, Byrd, Lu, and Nocedal (www.ece.northwestern.edu/∼nocedal/) with L. Stewart's wrapper (www.cs.toronto.edu/∼liam/). In all the experiments, we used 10 pairs of saved gradient vectors (see also Sha and Pereira, 2003).

14. The implementation of L-BFGS that we use requires both the gradient and objective when performing the line-search. In some line-search variants, it is possible to use only objective evaluations. In this case, the EG line search will be somewhat more costly, since the dual objective requires evaluations of both marginals and partition function, whereas the primal objective only requires the partition function. This will have an effect on running times only if the EG line search evaluates more than one point, which happened for less than 10%.

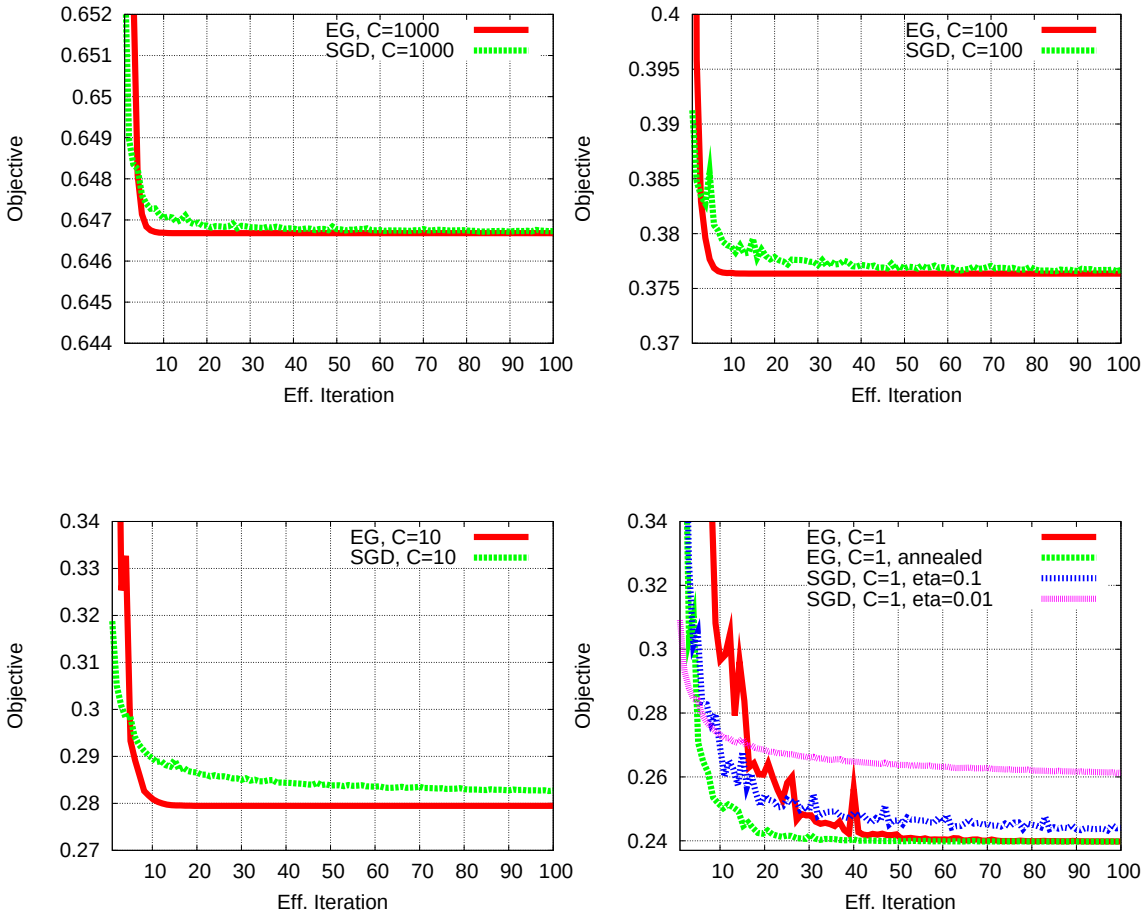

Figure 10: Graphs showing primal objective values on the MNIST learning task, comparing the EG randomized algorithm to stochastic gradient descent (SGD). The X axis shows number of effective training iterations, the Y axis shows primal objective. The graphs are for C equal to 1000, 100, 10, and 1. For $C = 1$ we show EG results with and without the annealed strategy described in Section 7.1.1. For $C = 1$ we also show two SGD curves, for learning rates 0.01 and 0.1: in this case $\eta_0 = 0.01$ was the best-performing learning rate after one iteration for both validation error and primal objective, however a post-hoc analysis shows that $\eta_0 = 0.1$ converges to a better value in the limit. Thus our strategy for choosing η_0 was not optimal in this case, although it is difficult to know how $\eta_0 = 0.1$ could be chosen without post-hoc analysis of the convergence for the different values of η_0. For other values of C our strategy for picking η_0 was more robust.

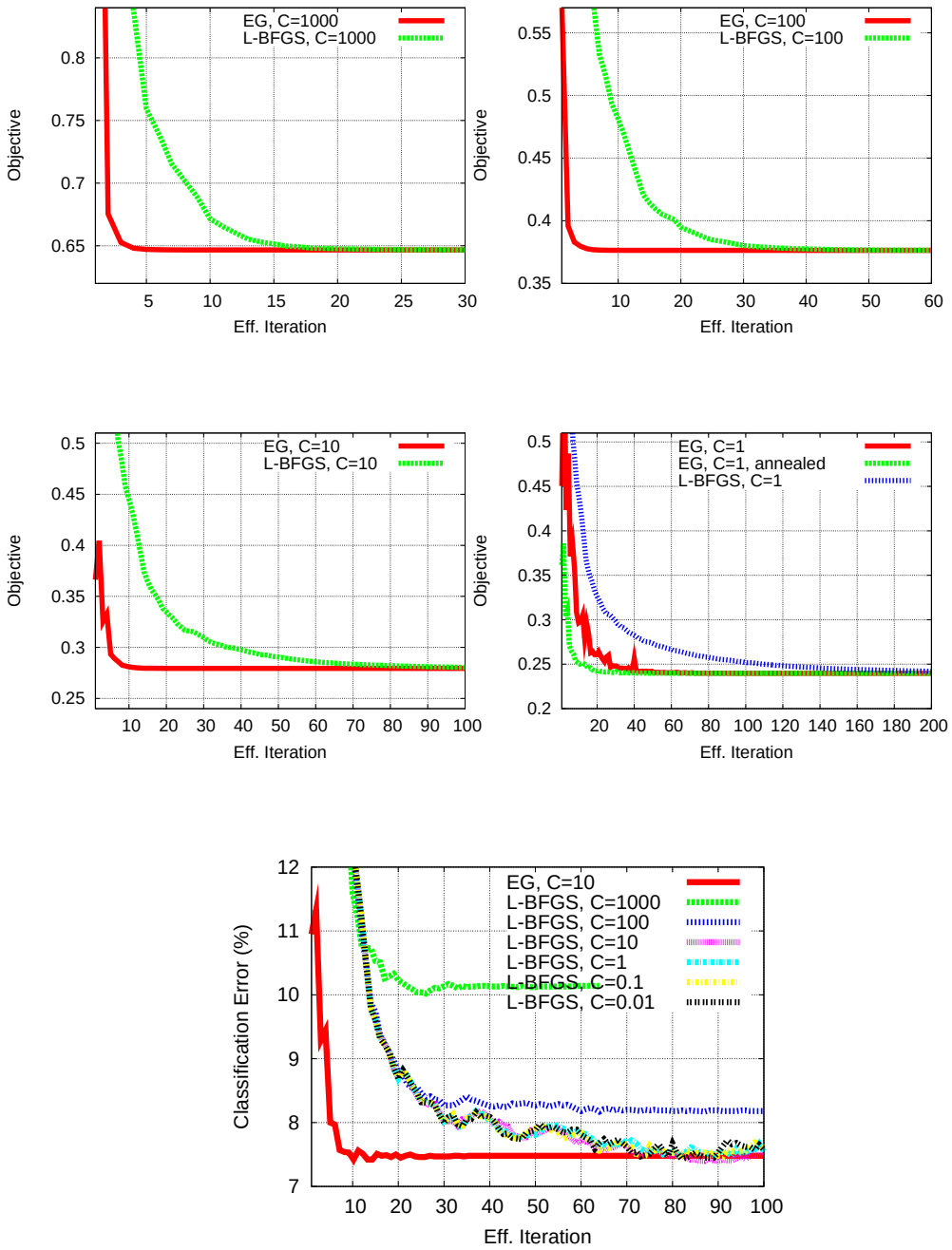

Figure 11: Results on the MNIST learning task, comparing the EG algorithm to L-BFGS. The figures on the first and second row show the primal objective for both algorithms, for various values of C. The bottom curve shows validation error for L-BFGS for various values of C and for EG with $C = 10$.

As in Section 7.1.3, we calculated primal values for EG. Figure 11 shows the primal objective for EG, and L-BFGS. It can be seen that the primal value for EG converges considerably faster than the L-BFGS one. Also shown is a curve of validation error for both algorithms. Here we show the results for EG with $C = 10$ and L-BFGS with various C values. It can be seen that L-BFGS does not outperform the EG curve for any value of C.

7.2 Structured learning - Dependency Parsing

Parsing of natural language sentences is a challenging structured learning task. Dependency parsing (McDonald et al., 2005) is a simplified form of parsing where the goal is to map sentences x into projective directed spanning trees over the set of words in x. Each label y is a set of directed arcs (dependencies) between pairs of words in the sentence. Each dependency is a pair (h,m) where h is the index of the head word of the dependency, and m is the index of the modifier word. Assuming we have a function $\mathbf{f}(x,h,m)$ that assigns a feature vector to dependencies (h,m), we can use a weight vector $\mathbf{w}$ to score a given tree y by $\mathbf{w}\cdot\sum_{(h,m)\in y}\mathbf{f}(x,h,m)$. Dependency parsing corresponds to a structured problem where the parts r are dependencies (h,m); the approach described in Section 4 can be applied efficiently to dependency structures. For projective dependency trees (e.g., see Koo et al., 2007), the required marginals can be computed efficiently using a variant of the inside-outside algorithm (Baker, 1979).

In the experiments below we use a feature set $\mathbf{f}(x,h,m)$ similar to that in McDonald et al. (2005) and Koo et al. (2007), resulting in $2,500,554$ features. We report results on the Spanish data-set which is part of the CoNLL-X Shared Task on multilingual dependency parsing (Buchholz and Marsi, 2006). The training data consists of $2,306$ sentences ($58,771$ tokens). To evaluate validation error, we use $1,000$ sentences ($30,563$ tokens) and report accuracy (rate of correct edges in a predicted parse tree) on these sentences.[15] Since we used only sentences from the training set, results are not directly comparable to the CoNLL-X shared task results. However, our previous work on this data set (Koo et al., 2007) shows that regularized max-margin and log-linear models typically outperform the averaged perceptron, which is not explicitly regularized.

As in the multi-class experiments, we compare to SGD and L-BFGS. The implementation of the algorithms is similar to that described in Section 7.1. The gradients for SGD and L-BFGS were obtained by calculating the relevant marginals of the model, using the inside-outside algorithm that was also used for EG. The learning rate for SGD was chosen as in the previous section; that is, we tested several learning rates ($\eta_0 = 1, 0.1, 0.001, 0.0001$) and chose the one that yielded the minimum validation error after one iteration.

Figure 12 shows results for EG and L-BFGS on the parsing task. We experiment with values of C in the set $\{0.1, 1, 10, 100, 1000\}$. Of these, the value that results in optimal validation error was $C = 10$. The performance of L-BFGS, SGD and EG is demonstrated in terms of the primal objective for a subset of the C values. L-BFGS and EG both converge to the optimal value, and EG is significantly faster. On the other hand, SGD does not converge to the optimum for all C values (e.g., for $C = 1, 10$), and when it does converge to the optimum, it is slower than EG.

Figure 12 also shows the validation error for EG at the optimal C value, compared to validation error for L-BFGS and SGD at various C values. Again, it can be seen that EG significantly outperforms L-BFGS. For SGD, performance is comparable to EG. However, as mentioned earlier, SGD

15. All $3,306$ sentences were obtained from the training data section of the CoNLL-X Spanish data-set (Civit and Martí, 2002).

C	Iterations	Total Iterations	Accuracy
1000	8	8	72.44
700	3.01	11.01	73.42
490	4.76	15.77	74.35
343	4.9	20.67	75.29
240.1	4.91	25.58	76.13
168.07	6.1	31.68	77.13
117.649	6.06	37.74	77.82
82.354	6.08	43.82	78.74
57.648	7.23	51.05	79.41
40.353	7.23	58.28	79.99
28.247	8.33	66.61	80.38
19.773	9.4	76.01	80.60
13.841	12.6	88.61	80.77
9.688	13.71	102.32	80.72
6.782	19.03	121.35	80.67
4.747	23.33	144.68	80.61
3.323	30.82	175.5	80.31
2.326	37.22	212.72	80.18
1.628	45.8	258.52	79.98
1.139	57.53	316.05	79.63
0.797	73.6	389.65	79.36

Table 3: Table showing number of effective iterations required to optimize a sequence of values for C for the parsing task, using the method described in Section 7.1.2. The column C shows the sequence of decreasing regularizer constants. *Iterations* shows the number of effective iterations over the training set required to optimize each value of C. *Total iterations* shows the cumulative value of *Iterations*, and *Accuracy* shows the validation accuracy obtained for every C value. It can be seen that the optimal accuracy is reached at $C = 13.841$.

in fact does not successfully optimize the primal objective for low values of C, and for higher values of C the SGD primal objective is slower to converge.

As in the multi-class experiments (see Figure 10), it is possible to find learning rates for SGD such that it converges to the primal optimum for $C = 1, 10$. However, the optimality of these rates only becomes evident after 10 iterations or more (results not shown). Thus, to find a learning rate for SGD that actually solves the optimization problem would typically require an additional few tens of iterations, making it significantly slower than EG.

Finally, it is possible to use EG to efficiently optimize over a set of regularization constants, as in Section 7.1.2. Table 3 shows results for a sequence of regularization constants.

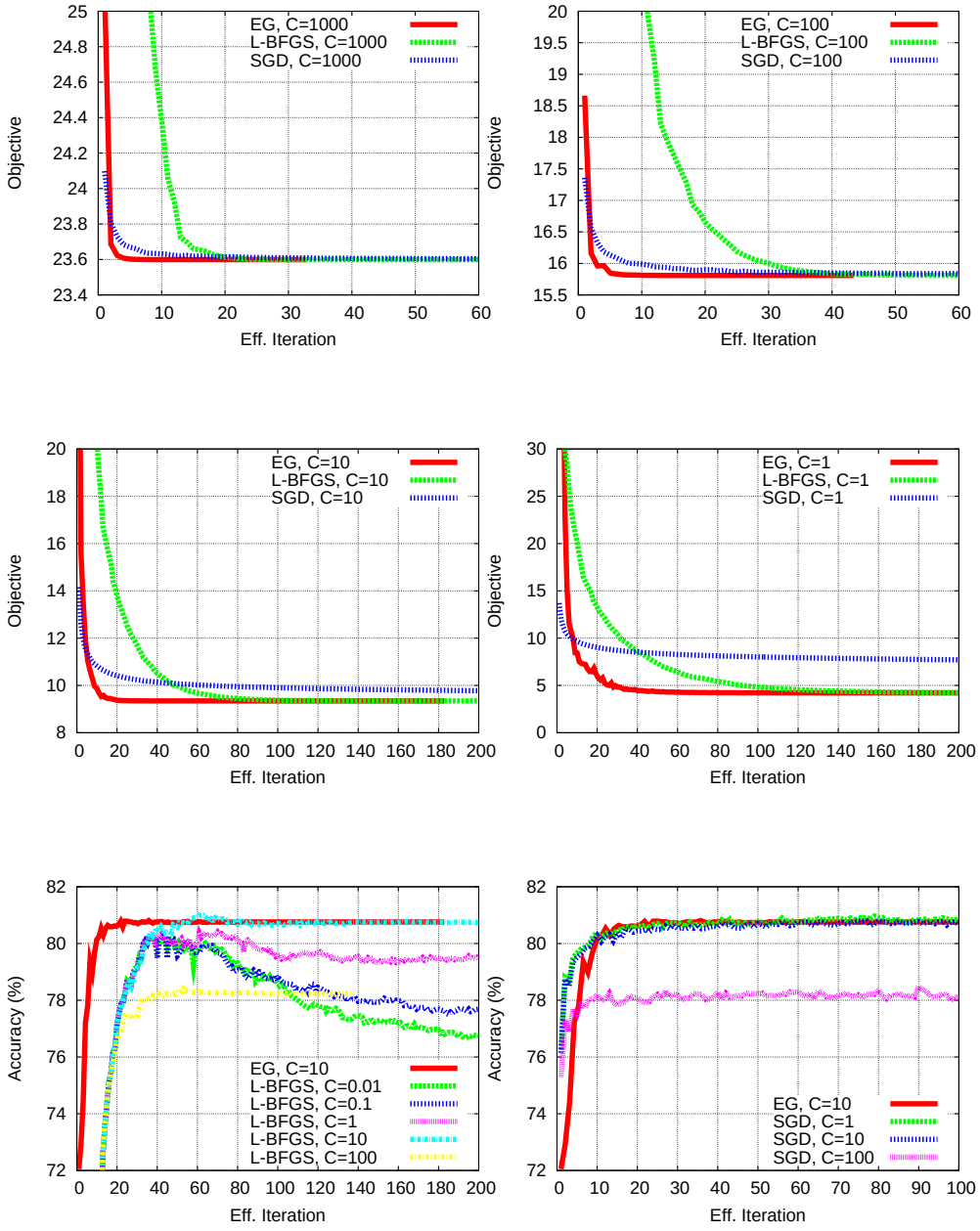

Figure 12: Results on the dependency-parsing task, comparing the EG algorithm to L-BFGS and SGD. All algorithms are trained on the log-linear objective function. The figures on the first and second rows show the primal objective for the three algorithms, for various values of C. The left bottom plot shows validation accuracy (measured as the fraction of correctly predicted edges) for L-BFGS for various values of C and for EG with $C = 10$. The right bottom plot show validation accuracy for EG (with $C = 10$) and SGD.

8. Experiments on Max-Margin Models

The max-margin loss (Eq. 3) has a discontinuity in its derivative. This makes optimization of max-margin models somewhat more involved than log-linear ones, since gradient algorithms such as L-BFGS cannot be used. This difficulty is exacerbated in the case of structured prediction models, since maximization in Eq. 3 is potentially over an exponentially large set.

In this section, we apply the EG algorithm to the max-margin problem, and compare its performance to the SVM-Struct algorithm presented in Tsochantaridis et al. (2004).[16] SVM-Struct is based on a cutting-plane algorithm that operates on the dual max-margin problem (D-MM) and results in monotone improvement in this dual. In this sense, it is similar to our EG algorithm. In order to facilitate a fair comparison, we report the performance of the two algorithms as a function of time. We do not report results by iteration since EG and SVM-struct involve different computation per iteration (e.g., SVM-Struct solves a QP per iteration).

We applied SVM-Struct and EG to the dependency parsing problem described in Section 7.2. To apply SVM-Struct to this problem, we supply it with a routine that finds the $y \in \mathcal{Y}$ which attains the maximum of the hinge-loss in Eq. 3. This maximum can be found using a Viterbi-style algorithm. For the value of C we experimented with $C \in \{1, 10, 100, 1000, 10000\}$. The optimal value in terms of validation error was $C = 100$.

Figure 13 shows results in terms of primal and dual objective and in terms of accuracy. It can be seen that EG is considerably faster than SVM-Struct for most C values. The performance is comparable only for $C = 1$, where convergence is slow for both algorithms.

9. Conclusion

We have presented novel algorithms for large-scale learning of log-linear and max-margin models, which provably converge to the optimal value of the respective loss functions. Although the algorithms have both batch and online variants, the online version turns out to be much more effective, both in theory and in practice. Our theoretical results (see Section 5.1) suggest that the online algorithm requires a factor of n less iterations to achieve a desired accuracy ε in the dual objective. This factor results from the fact that the online algorithm can use a learning rate η that is n times larger than the batch case to obtain updates that decrease the dual objective. Intuitively, this difference is associated with the fact that the batch algorithm updates all $\mathbf{u}$ values simultaneously. The dual objective has a term $\mathbf{u}^T A\mathbf{u}$ which involves all the $\mathbf{u}_i$ variables and second order interactions between them. It turns out that for batch updates only a relatively small change in the $\mathbf{u}_i$ is allowed, if one still requires an improvement in the dual objective after the update. It is possible that our bounds for the batch convergence rate are more conservative than those for the online case. However, we have observed in practice that the batch algorithm is much slower to converge. Furthermore, we also observed that other batch-based algorithms such as L-BFGS and conjugate gradient converge more slowly than the online EG algorithm.

Our results provide an $O(\log(\frac{1}{\varepsilon}))$ rate for the log-linear model, as opposed to $O(\frac{1}{\varepsilon})$ for max-margin. If these bounds are tight, they would imply that log-linear models have an advantage over max-margin ones in terms of training efficiency. However, it is possible that the analysis is not tight, and that improved rates may also be obtained for the max-margin model. In any case, this raises

16. The code is available from svmlight.joachims.org/svm_struct.html.

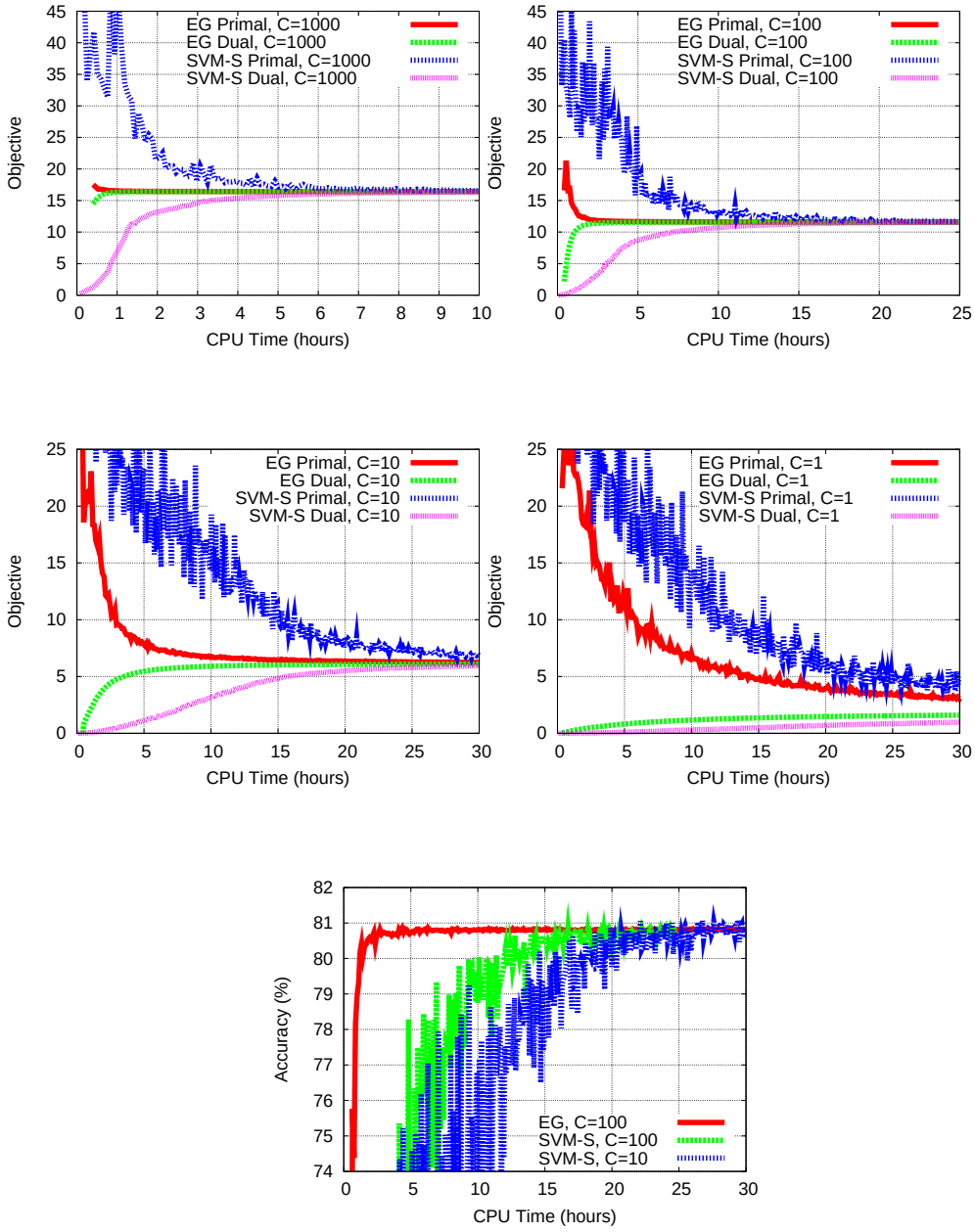

Figure 13: Results on the dependency-parsing task, comparing the EG algorithm to SVM-Struct. Both algorithms are trained on a max-margin model. The figures on the first and second rows show the primal objective for both algorithms, for various values of C. The bottom curve shows validation accuracy (measured as the fraction of correctly predicted edges) for SVM-Struct for various values of C and for EG with $C = 100$ (the value that yielded the highest validation accuracy). The X axis on all curves is running time in hours.

the interesting question of comparing classification models not only in terms of accuracy but also in terms of optimization efficiency.

Our convergence rates are with respect to accuracy in the dual objective. Some previous work (e.g., Shalev-Shwartz et al., 2007) has considered the accuracy with respect to the primal objective. It is relatively easy to show that in order to obtain ε accuracy in the primal, the EG algorithms require $O(\log(\frac{1}{\varepsilon}))$ updates for the log-linear problem and $O(\frac{1}{\varepsilon^2})$ for the max-margin case. It is possible that a more refined analysis of the max-margin case will result in $O(\frac{1}{\varepsilon})$ (e.g., see List et al., 2007), but we leave this for further study.

Most of our proofs rely on a relation between B_Q and the KL divergence. This relation holds for max-margin learning as well, a fact that simplifies previous results in this setting (Bartlett et al., 2005). We expect a similar analysis to hold for other functions Q.

An interesting extension of our method is to using second order derivative information, or its approximations, as in L-BFGS (Byrd et al., 1995). Such information may be used to obtain more accurate minimization for each $\mathbf{u}_i$ and may speed up convergence. Another possible improvement is to the line search method. In the experiments reported here we use a crude mechanism for adapting the learning rate, and it is possible that a more careful procedure will improve convergence rates in practice.

Parallelization is becoming increasingly relevant as multi-core CPUs become available. For the batch EG algorithm, it is straightforward to distribute the computation among k processors. One method for distributing the online EG algorithm would be to update k examples in parallel on k different processors. It should be possible to analyze this setting in a similar way to our proofs for the online case, but we leave this to future work.

Finally, our results show that the EG algorithms are highly competitive with state-of-the-art methods for training log-linear and max-margin models. We thus expect them to become useful as learning algorithms, particularly in the structured prediction setting.

Acknowledgments

The authors gratefully acknowledge the following sources of support. Amir Globerson was supported by a fellowship from the Rothschild Foundation - Yad Hanadiv. Terry Koo was funded by a grant from the NSF (DMS-0434222) and a grant from NTT, Agmt. Dtd. 6/21/1998. Xavier Carreras was supported by the Catalan Ministry of Innovation, Universities and Enterprise, and by a grant from NTT, Agmt. Dtd. 6/21/1998. Michael Collins was funded by NSF grants 0347631 and DMS-0434222. Peter Bartlett was funded by a grant from the NSF (DMS-0434383).

Appendix A. $O(\frac{1}{\varepsilon})$ Rate for Batch Algorithms - Proof of Lemma 5

We use a similar proof technique to that of Kivinen and Warmuth (2001). In particular, we need the following Lemma, which is very similar to results used by Kivinen and Warmuth (2001):

Lemma 10 *(See Kivinen and Warmuth (2001), Proof of Lemma 4) For any convex function $Q(\mathbf{u})$ over Δ^n, for any $\mathbf{z} \in \Delta^n$, and any $\mathbf{u}^t$ in the interior of Δ^n, if $\mathbf{u}^{t+1}$ is derived from $\mathbf{u}^t$ using the EG updates with a learning rate η, then*

$$\eta Q(\mathbf{u}^t) - \eta Q(\mathbf{z}) = D[\mathbf{z}\|\mathbf{u}^t] - D[\mathbf{z}\|\mathbf{u}^{t+1}] + D[\mathbf{u}^t\|\mathbf{u}^{t+1}] - \eta B_Q[\mathbf{z}\|\mathbf{u}^t] \; . \tag{9}$$

Proof: By the definition of Bregman divergence, we have

$$\eta Q(\mathbf{u}^t) - \eta Q(\mathbf{z}) = -\eta \nabla Q(\mathbf{u}^t) \cdot (\mathbf{z} - \mathbf{u}^t) - \eta B_Q[\mathbf{z} \| \mathbf{u}^t] \ . \tag{10}$$

Given that $\mathbf{u}^{t+1}$ is derived from $\mathbf{u}^t$ using EG updates,

$$u_{i,y}^{t+1} = \frac{u_{i,y}^t e^{-\eta \nabla_{i,y}}}{Z_i^t} \ ,$$

where Z_i^t is a normalization constant, and $\nabla_{i,y} = \frac{\partial Q(\mathbf{u}^t)}{\partial u_{i,y}}$. Simple algebra then shows that:

$$\begin{aligned} D[\mathbf{z}\|\mathbf{u}^t] - D[\mathbf{z}\|\mathbf{u}^{t+1}] + D[\mathbf{u}^t\|\mathbf{u}^{t+1}] &= \sum_{i,y} \left(z_{i,y} \log \frac{z_{i,y}}{u_{i,y}^t} - z_{i,y} \log \frac{z_{i,y}}{u_{i,y}^{t+1}} + u_{i,y}^t \log \frac{u_{i,y}^t}{u_{i,y}^{t+1}} \right) \\ &= \sum_{i,y} (z_{i,y} - u_{i,y}^t) \log \frac{u_{i,y}^{t+1}}{u_{i,y}^t} \\ &= \sum_{i,y} (z_{i,y} - u_{i,y}^t)(-\eta \nabla_{i,y} - \log Z_i^t) \\ &= \sum_{i,y} (z_{i,y} - u_{i,y}^t)(-\eta \nabla_{i,y}) \\ &= -\eta \nabla Q(\mathbf{u}^t) \cdot (\mathbf{z} - \mathbf{u}^t) \ . \end{aligned} \tag{11}$$

Note that we have used $\sum_{i,y}(z_{i,y} - u_{i,y}^t) \log Z_i^t = 0$, which follows because $\sum_{i,y} z_{i,y} \log Z_i^t = \sum_{i,y} u_{i,y}^t \log Z_i^t$ $= \sum_i \log Z_i^t$.

Combining Eq. 10 and Eq. 11 gives Eq. 9, thus proving the lemma. □

We can now prove Lemma 5:

Proof of Lemma 5: Using $-\eta B_Q[\mathbf{z}\|\mathbf{u}^t] \leq 0$, Lemma 10 implies that for all t

$$\eta Q(\mathbf{u}^t) - \eta Q(\mathbf{z}) \leq D[\mathbf{z}\|\mathbf{u}^t] - D[\mathbf{z}\|\mathbf{u}^{t+1}] + D[\mathbf{u}^t\|\mathbf{u}^{t+1}] \ . \tag{12}$$

By the assumptions of Lemma 5, $Q(\mathbf{u})$ is τ-upper-bounded, and $0 \leq \eta \leq \frac{1}{\tau}$, hence by Lemma 3 we have

$$Q(\mathbf{u}^t) - Q(\mathbf{u}^{t+1}) \geq \frac{1}{\eta} D[\mathbf{u}^t\|\mathbf{u}^{t+1}] \ . \tag{13}$$

Combining Eqs. 12 and 13 gives for all t

$$\eta Q(\mathbf{u}^{t+1}) - \eta Q(\mathbf{z}) \leq D[\mathbf{z}\|\mathbf{u}^t] - D[\mathbf{z}\|\mathbf{u}^{t+1}] \ .$$

Summing this over $t = 1, \ldots, T$ gives (the sum on the RHS telescopes)

$$\eta \sum_{t=1}^{T} Q(\mathbf{u}^{t+1}) - \eta T Q(\mathbf{z}) \leq D[\mathbf{z}\|\mathbf{u}^1] - D[\mathbf{z}\|\mathbf{u}^{T+1}] \ .$$

Because $Q(\mathbf{u}^t)$ is monotone decreasing (by Eq. 13), we have $T Q(\mathbf{u}^{T+1}) \leq \sum_{t=1}^{T} Q(\mathbf{u}^{t+1})$ and simple algebra gives

$$Q(\mathbf{u}^{T+1}) \leq Q(\mathbf{z}) + \frac{D[\mathbf{z}\|\mathbf{u}^1] - D[\mathbf{z}\|\mathbf{u}^{T+1}]}{\eta T} \ .$$

Dropping the term $D[\mathbf{z}\|\mathbf{u}^{T+1}]$ (because $-D[\mathbf{z}\|\mathbf{u}^{T+1}] \leq 0$) we obtain

$$Q(\mathbf{u}^{T+1}) \leq Q(\mathbf{z}) + \frac{D[\mathbf{z}\|\mathbf{u}^1]}{\eta T} ,$$

as required.

Appendix B. $O(\log(\frac{1}{\varepsilon}))$ Rate for Batch Algorithms - Proof of Lemma 6

By the assumptions of Lemma 6, $Q(\mathbf{u})$ is τ-upper-bounded, and $0 \leq \eta \leq \frac{1}{\tau}$, hence by Lemma 3 we have for all t

$$Q(\mathbf{u}^t) - Q(\mathbf{u}^{t+1}) \geq \frac{1}{\eta} D[\mathbf{u}^t\|\mathbf{u}^{t+1}] .$$

Combining this result with Lemma 10 gives

$$\eta Q(\mathbf{u}^{t+1}) - \eta Q(\mathbf{z}) \leq D[\mathbf{z}\|\mathbf{u}^t] - D[\mathbf{z}\|\mathbf{u}^{t+1}] - \eta B_Q[\mathbf{z}\|\mathbf{u}^t] .$$

We can now make use of the assumption that $Q(\mathbf{u})$ is (μ,τ)-bounded, and hence $\eta B_Q[\mathbf{z}\|\mathbf{u}^t] \geq \eta\mu D[\mathbf{z}\|\mathbf{u}^t]$, to obtain

$$\begin{aligned} \eta Q(\mathbf{u}^{t+1}) - \eta Q(\mathbf{z}) &\leq D[\mathbf{z}\|\mathbf{u}^t] - D[\mathbf{z}\|\mathbf{u}^{t+1}] - \eta\mu D[\mathbf{z}\|\mathbf{u}^t] \\ &= (1-\eta\mu) D[\mathbf{z}\|\mathbf{u}^t] - D[\mathbf{z}\|\mathbf{u}^{t+1}] \\ &\leq (1-\eta\mu) D[\mathbf{z}\|\mathbf{u}^t] . \end{aligned} \tag{14}$$

If there exists a $t \leq T$ such that $Q(\mathbf{u}^{t+1}) - Q(\mathbf{z}) \leq 0$ then because $Q(\mathbf{u}^t)$ decreases monotonically with t we have $Q(\mathbf{u}^{T+1}) \leq Q(\mathbf{u}^{t+1}) \leq Q(\mathbf{z})$ and the lemma trivially holds. Otherwise, it must be the case that $Q(\mathbf{u}^{t+1}) - Q(\mathbf{z}) \geq 0$ for all $t \leq T$, and thus for all $t \leq T$

$$D[\mathbf{z}\|\mathbf{u}^{t+1}] \leq (1-\eta\mu) D[\mathbf{z}\|\mathbf{u}^t] .$$

Using this inequality recursively for $t = 1,\ldots,T$ we get

$$D[\mathbf{z}\|\mathbf{u}^{T+1}] \leq (1-\eta\mu)^T D[\mathbf{z}\|\mathbf{u}^1] .$$

Substituting back into Eq. 14 we obtain

$$Q(\mathbf{u}^{T+1}) - Q(\mathbf{z}) \leq \frac{(1-\eta\mu)^T}{\eta} D[\mathbf{z}\|\mathbf{u}^1] \leq \frac{e^{-\eta\mu T}}{\eta} D[\mathbf{z}\|\mathbf{u}^1] ,$$

where we have used $\log(1-x) \leq -x$.

Appendix C. Proof of Lemma 7

For the regularized log-likelihood dual, for any $\mathbf{v} \in \Delta$

$$Q_{\mathbf{u},i}(\mathbf{v}) = \sum_y v_y \log v_y + \frac{1}{2}\mathbf{v}^T A(i,i)\mathbf{v} + \sum_{j\neq i}\sum_y u_{j,y}\log u_{j,y} + \frac{1}{2}\sum_{j\neq i}\sum_{k\neq i}\mathbf{u}_j^T A(j,k)\mathbf{u}_k + \sum_{j\neq i}\mathbf{u}_j^T A(j,i)\mathbf{v} ,$$

where $A(j,k)$ is the $|\mathcal{Y}| \times |\mathcal{Y}|$ sub-matrix of A defined as $A_{y,z}(j,k) = A_{(j,y),(k,z)}$. To obtain the Bregman divergence $B_{Q_{\mathbf{u},i}}[\mathbf{p}\|\mathbf{q}]$, note that the last three terms in $Q_{\mathbf{u},i}(\mathbf{v})$ are either constant or linear in $\mathbf{v}$ and thus do not contribute to $B_{Q_{\mathbf{u},i}}[\mathbf{p}\|\mathbf{q}]$. It follows that

$$B_{Q_{\mathbf{u},i}}[\mathbf{p}\|\mathbf{q}] = D[\mathbf{p}\|\mathbf{q}] + M_{A(i,i)}[\mathbf{p}\|\mathbf{q}] \ .$$

By a similar argument to the proof of Lemma 3, it follows that $B_{Q_{\mathbf{u},i}}[\mathbf{p}\|\mathbf{q}] \leq (1+|A(i,i)|_\infty)D[\mathbf{p}\|\mathbf{q}]$. Because $A(i,i)$ is a sub-matrix of A we have $|A(i,i)|_\infty \leq |A|_\infty$, and the first part of the lemma follows. For the max-margin dual, a similar argument shows that

$$B_{Q_{\mathbf{u},i}}[\mathbf{p}\|\mathbf{q}] = M_{A(i,i)}[\mathbf{p}\|\mathbf{q}] \ ,$$

so we have $B_{Q_{\mathbf{u},i}}[\mathbf{p}\|\mathbf{q}] \leq |A(i,i)|_\infty D[\mathbf{p}\|\mathbf{q}] \leq |A|_\infty D[\mathbf{p}\|\mathbf{q}]$. □

Appendix D. Proof of Lemma 8

For the proof we will need some additional notation, which makes explicit the relationship between the sequence of dual variables $\mathbf{u}^1, \mathbf{u}^2, \ldots, \mathbf{u}^{T+1}$ and the sequence of indices $k_1, k_2, \ldots, k_T$ used in the algorithm in Figure 2. We will use the following definitions:

- We use k_1^t to denote a sequence of indices $k_1, k_2, \ldots, k_t$. We take k_1^0 to be the empty sequence.
- We write $\mathbf{r} : \Delta^n \times [1 \ldots n] \to \Delta^n$ to denote the function that corresponds to an EG update on a single example. More specifically, we have

$$\begin{aligned} \mathbf{r}_i(\mathbf{u},k) &= \mathbf{u}_i \ \text{ for } i \neq k \\ r_{i,y}(\mathbf{u},k) &\propto u_{i,y} \exp\{-\eta \nabla_{i,y}\} \ \text{ where } \nabla_{i,y} = \tfrac{\partial Q(\mathbf{u})}{\partial u_{i,y}} \text{ for } i = k, \text{ for all } y \ . \end{aligned}$$

- Finally, for any choice of index sequence k_1^T we will define a sequence of dual variables using the following iterative definition:

$$\begin{aligned} \mathbf{u}(k_1^0) &= \mathbf{u}^1 \\ \mathbf{u}(k_1^t) &= \mathbf{r}(\mathbf{u}(k_1^{t-1}), k_t) \ \text{ for } t \geq 1 \ . \end{aligned}$$

  Here $\mathbf{u}^1$ is the initial setting of the dual variables, as shown in the algorithm in Figure 2.

From these definitions it follows that if k_1^T is the sequence of indices chosen during a run of the algorithm in Figure 2, then the sequence of dual variables $\mathbf{u}^1, \mathbf{u}^2, \ldots, \mathbf{u}^{T+1}$ is such that $\mathbf{u}^{t+1} = \mathbf{u}(k_1^t)$ for $t = 0 \ldots T$. We can now give the proof.

Proof of Lemma 8. First, we have for any $\mathbf{u}, \mathbf{z} \in \Delta^n$

$$\begin{aligned} Q(\mathbf{u}) &\leq Q(\mathbf{z}) + (\mathbf{u}-\mathbf{z}) \cdot \nabla Q(\mathbf{u}) \\ &= Q(\mathbf{z}) + \frac{1}{\eta} \sum_{i=1}^n \Big[D[\mathbf{z}_i \| \mathbf{u}_i] - D[\mathbf{z}_i \| \mathbf{r}_i(\mathbf{u},i)] + D[\mathbf{u}_i \| \mathbf{r}_i(\mathbf{u},i)] \Big] \ . \end{aligned} \tag{15}$$

The second line follows by similar arguments to those in the proof of Lemma 10.

Next consider the terms on the right-hand-side of the inequality. For any i, we have

$$\begin{aligned} & D[\mathbf{z}_i \| \mathbf{u}_i] - D[\mathbf{z}_i \| \mathbf{r}_i(\mathbf{u}, i)] \\ &= D[\mathbf{z}_i \| \mathbf{u}_i] - D[\mathbf{z}_i \| \mathbf{r}_i(\mathbf{u}, i)] + \sum_{j=1\ldots n, j\neq i} D[\mathbf{z}_j \| \mathbf{u}_j] - \sum_{j=1\ldots n, j\neq i} D[\mathbf{z}_j \| \mathbf{u}_j] \quad (16) \\ &= D[\mathbf{z}_i \| \mathbf{u}_i] - D[\mathbf{z}_i \| \mathbf{r}_i(\mathbf{u}, i)] + \sum_{j=1\ldots n, j\neq i} D[\mathbf{z}_j \| \mathbf{u}_j] - \sum_{j=1\ldots n, j\neq i} D[\mathbf{z}_j \| \mathbf{r}_j(\mathbf{u}, i)] \quad (17) \\ &= D[\mathbf{z} \| \mathbf{u}] - D[\mathbf{z} \| \mathbf{r}(\mathbf{u}, i)] \ . \quad (18) \end{aligned}$$

Here Eq. 17 follows from Eq. 16 because $\mathbf{r}_j(\mathbf{u}, i) = \mathbf{u}_j$ for $j \neq i$. In addition, for any i we have

$$\frac{1}{\eta} D[\mathbf{u}_i \| \mathbf{r}_i(\mathbf{u}, i)] \leq Q(\mathbf{u}) - Q(\mathbf{r}(\mathbf{u}, i)) \ . \quad (19)$$

This follows because by the assumption in the lemma, $Q(\mathbf{u})$ is τ-online-upper-bounded, so we have $B_{Q_{\mathbf{u},i}}[\mathbf{r}_i(\mathbf{u}, i) \| \mathbf{u}_i] \leq \tau D[\mathbf{r}_i(\mathbf{u}, i) \| \mathbf{u}_i]$. By an application of Lemma 2 to the convex function $Q_{\mathbf{u},i}$, noting that by assumption $\eta \leq 1/\tau$, it follows that

$$\frac{1}{\eta} D[\mathbf{u}_i \| \mathbf{r}_i(\mathbf{u}, i)] \leq Q_{\mathbf{u},i}(\mathbf{u}_i) - Q_{\mathbf{u},i}(\mathbf{r}_i(\mathbf{u}, i)) \ .$$

Finally, note that $Q_{\mathbf{u},i}(\mathbf{u}_i) = Q(\mathbf{u})$, and $Q_{\mathbf{u},i}(\mathbf{r}_i(\mathbf{u}, i)) = Q(\mathbf{r}(\mathbf{u}, i))$, giving the result in Eq. 19.

Combining Equations 15, 18 and 19 gives for any $\mathbf{u}$,

$$Q(\mathbf{u}) \leq Q(\mathbf{z}) + \frac{1}{\eta} \sum_{i=1}^{n} \Big[D[\mathbf{z} \| \mathbf{u}] - D[\mathbf{z} \| \mathbf{r}(\mathbf{u}, i)] \Big] + \sum_{i=1}^{n} \Big[Q(\mathbf{u}) - Q(\mathbf{r}(\mathbf{u}, i)) \Big] \ . \quad (20)$$

Because Eq. 20 holds for any value of $\mathbf{u}$, we have for all $t = 1 \ldots T$, for all $k_1^{t-1} \in [1 \ldots n]^{t-1}$,

$$\begin{aligned} Q(\mathbf{u}(k_1^{t-1})) \leq Q(\mathbf{z}) &+ \frac{1}{\eta} \sum_{i=1}^{n} \Big[D[\mathbf{z} \| \mathbf{u}(k_1^{t-1})] - D[\mathbf{z} \| \mathbf{r}(\mathbf{u}(k_1^{t-1}), i)] \Big] \\ &+ \sum_{i=1}^{n} \Big[Q(\mathbf{u}(k_1^{t-1})) - Q(\mathbf{r}(\mathbf{u}(k_1^{t-1}), i)) \Big] \ . \quad (21) \end{aligned}$$

We can now take an expectation of both sides of the inequality in Eq. 21. For any function $f(k_1^t)$, we use the notation $\mathbf{E}_t[f(k_1^t)]$ to denote the expected value of $f(k_1^t)$ when k_1^t is drawn uniformly at random from $[1, 2, \ldots, n]^t$; more precisely

$$\mathbf{E}_t[f(k_1^t)] = \frac{1}{n^t} \sum_{k_1^t \in [1,2,\ldots,n]^t} f(k_1^t) \ .$$

We apply the operator $\mathbf{E}_{t-1}$ to both sides of Eq. 21. We consider the different terms in turn. First,

$$\mathbf{E}_{t-1}[Q(\mathbf{u}(k_1^{t-1}))] = \mathbf{E}_T[Q(\mathbf{u}(k_1^{t-1}))] \ .$$

This follows because $Q(\mathbf{u}(k_1^{t-1}))$ does not depend on the values for $k_t, \ldots, k_T$. Clearly, $\mathbf{E}_{t-1}[Q(\mathbf{z})] = Q(\mathbf{z})$. Next,

$$\mathbf{E}_{t-1} \left[\frac{1}{\eta} \sum_{i=1}^{n} D[\mathbf{z} \| \mathbf{u}(k_1^{t-1})] - D[\mathbf{z} \| \mathbf{r}(\mathbf{u}(k_1^{t-1}), i)] \right]$$

$$
\begin{aligned}
&= \mathbf{E}_{t-1}\left[\frac{n}{\eta}D[\mathbf{z}\|\mathbf{u}(k_1^{t-1})]\right] - \mathbf{E}_{t-1}\left[\frac{n}{\eta}\sum_{i=1}^{n}\frac{1}{n}D[\mathbf{z}\|\mathbf{r}(\mathbf{u}(k_1^{t-1}),i)]\right] \\
&= \mathbf{E}_{t-1}\left[\frac{n}{\eta}D[\mathbf{z}\|\mathbf{u}(k_1^{t-1})]\right] - \mathbf{E}_{t}\left[\frac{n}{\eta}D[\mathbf{z}\|\mathbf{u}(k_1^{t})]\right] \\
&= \mathbf{E}_{T}\left[\frac{n}{\eta}D[\mathbf{z}\|\mathbf{u}(k_1^{t-1})]\right] - \mathbf{E}_{T}\left[\frac{n}{\eta}D[\mathbf{z}\|\mathbf{u}(k_1^{t})]\right] .
\end{aligned}
$$

Finally, we consider the last term:

$$
\begin{aligned}
&\mathbf{E}_{t-1}\left[\sum_{i=1}^{n} Q(\mathbf{u}(k_1^{t-1})) - Q(\mathbf{r}(\mathbf{u}(k_1^{t-1}),i))\right] \\
&\quad= \mathbf{E}_{t-1}\left[nQ(\mathbf{u}(k_1^{t-1})) - n\sum_{i=1}^{n}\frac{1}{n}Q(\mathbf{r}(\mathbf{u}(k_1^{t-1}),i))\right] \\
&\quad= \mathbf{E}_{t-1}\left[nQ(\mathbf{u}(k_1^{t-1}))\right] - \mathbf{E}_{t}\left[nQ(\mathbf{u}(k_1^{t}))\right] \\
&\quad= \mathbf{E}_{T}\left[nQ(\mathbf{u}(k_1^{t-1}))\right] - \mathbf{E}_{T}\left[nQ(\mathbf{u}(k_1^{t}))\right] .
\end{aligned}
$$

Combining these results with Eq. 21 gives

$$
\begin{aligned}
\mathbf{E}_T[Q(\mathbf{u}(k_1^{t-1}))] \;\leq\; & Q(\mathbf{z}) + \mathbf{E}_T\left[\frac{n}{\eta}D[\mathbf{z}\|\mathbf{u}(k_1^{t-1})]\right] - \mathbf{E}_T\left[\frac{n}{\eta}D[\mathbf{z}\|\mathbf{u}(k_1^{t})]\right] \\
& + \mathbf{E}_T\left[nQ(\mathbf{u}(k_1^{t-1}))\right] - \mathbf{E}_T\left[nQ(\mathbf{u}(k_1^{t}))\right] . \qquad (22)
\end{aligned}
$$

Summing Eq. 22 over $t = 1 \ldots T$ gives

$$
\begin{aligned}
\sum_{t=1}^{T}\mathbf{E}_T[Q(\mathbf{u}(k_1^{t-1}))] \;\leq\; & TQ(\mathbf{z}) + \mathbf{E}_T\left[\frac{n}{\eta}D[\mathbf{z}\|\mathbf{u}(k_1^{0})]\right] - \mathbf{E}_T\left[\frac{n}{\eta}D[\mathbf{z}\|\mathbf{u}(k_1^{T})]\right] \\
& + \mathbf{E}_T\left[nQ(\mathbf{u}(k_1^{0}))\right] - \mathbf{E}_T\left[nQ(\mathbf{u}(k_1^{T}))\right] \\
\leq\; & TQ(\mathbf{z}) + \frac{n}{\eta}D[\mathbf{z}\|\mathbf{u}(k_1^{0})] + n\left[Q(\mathbf{u}(k_1^{0})) - Q(\mathbf{u}^*)\right] \\
=\; & TQ(\mathbf{z}) + \frac{n}{\eta}D[\mathbf{z}\|\mathbf{u}^1] + n\left[Q(\mathbf{u}^1) - Q(\mathbf{u}^*)\right] , \qquad (23)
\end{aligned}
$$

where $\mathbf{u}^* = \operatorname{argmin}_{\mathbf{u}\in\Delta^n} Q(\mathbf{u})$. Finally, note that for any value of k_1^T we have $Q(\mathbf{u}(k_1^t)) \leq Q(\mathbf{u}(k_1^{t-1}))$ for $t = 1 \ldots T$. Thus

$$
\mathbf{E}_T[Q(\mathbf{u}(k_1^t))] \leq \mathbf{E}_T[Q(\mathbf{u}(k_1^{t-1}))] ,
$$

and

$$
T\mathbf{E}_T[Q(\mathbf{u}(k_1^T)] \leq \sum_{t=1}^{T}\mathbf{E}_T[Q(\mathbf{u}(k_1^{t-1}))] .
$$

Combining this with Eq. 23 gives

$$
T\mathbf{E}_T[Q(\mathbf{u}(k_1^T)] \leq TQ(\mathbf{z}) + \frac{n}{\eta}D[\mathbf{z}\|\mathbf{u}^1] + n\left[Q(\mathbf{u}^1) - Q(\mathbf{u}^*)\right] ,
$$

thus proving the lemma. □

Appendix E. Proof of Lemma 9

(Note: this proof builds on notation and techniques given in the proof of Lemma 8, see Appendix D.)

We begin with the following identity

$$B_Q[\mathbf{z}||\mathbf{u}] \;=\; Q(\mathbf{z}) - Q(\mathbf{u}) - \nabla Q(\mathbf{u}) \cdot (\mathbf{z} - \mathbf{u}) \;.$$

Rearranging yields for any $\mathbf{z}, \mathbf{u} \in \Delta^n$,

$$\begin{aligned} Q(\mathbf{u}) - Q(\mathbf{z}) &= \nabla Q(\mathbf{u}) \cdot (\mathbf{u} - \mathbf{z}) - B_Q[\mathbf{z}||\mathbf{u}] \\ &= \frac{1}{\eta} \sum_{i=1}^{n} \Big[D[\mathbf{z}_i||\mathbf{u}_i] - D[\mathbf{z}_i||\mathbf{r}_i(\mathbf{u},i)] + D[\mathbf{u}_i||\mathbf{r}_i(\mathbf{u},i)] \Big] - B_Q[\mathbf{z}||\mathbf{u}] \;, \end{aligned}$$

where the second line follows by a similar argument to the proof of Lemma 10.

By applying similar arguments to those leading to Eq. 20 in Appendix D, we get for any $\mathbf{z}, \mathbf{u} \in \Delta^n$,

$$\begin{aligned} Q(\mathbf{u}) - Q(\mathbf{z}) &\leq \frac{1}{\eta} \sum_{i=1}^{n} \Big[D[\mathbf{z}\|\mathbf{u}] - D[\mathbf{z}\|\mathbf{r}(\mathbf{u},i)] \Big] + \sum_{i=1}^{n} \Big[Q(\mathbf{u}) - Q(\mathbf{r}(\mathbf{u},i)) \Big] - B_Q[\mathbf{z}||\mathbf{u}] \\ &= \frac{n}{\eta} D[\mathbf{z}\|\mathbf{u}] - \frac{n}{\eta} \sum_{i=1}^{n} \frac{1}{n} D[\mathbf{z}\|\mathbf{r}(\mathbf{u},i)] + \sum_{i=1}^{n} \Big[Q(\mathbf{u}) - Q(\mathbf{r}(\mathbf{u},i)) \Big] - B_Q[\mathbf{z}||\mathbf{u}] \\ &\leq \left(\frac{n}{\eta} - \mu \right) D[\mathbf{z}\|\mathbf{u}] - \frac{n}{\eta} \sum_{i=1}^{n} \frac{1}{n} D[\mathbf{z}\|\mathbf{r}(\mathbf{u},i)] + \sum_{i=1}^{n} \Big[Q(\mathbf{u}) - Q(\mathbf{r}(\mathbf{u},i)) \Big] \;, \end{aligned}$$

where in the third line, we have used the assumption from the lemma that $Q(\mathbf{u})$ is (μ, τ)-online-bounded, and hence $B_Q[\mathbf{z}||\mathbf{u}] \geq \mu D[\mathbf{z}\|\mathbf{u}]$. The inequality above holds for all $\mathbf{u}, \mathbf{z} \in \Delta^n$, so we can take $\mathbf{u} = \mathbf{u}(k_1^{t-1})$ for any sequence k_1^{t-1}. Taking expectations of both sides, and using similar arguments to those leading to Eq. 22 in Appendix D, we get

$$\begin{aligned} \mathbf{E}_T \Big[Q(\mathbf{u}(k_1^{t-1})) \Big] - Q(\mathbf{z}) \;\leq\; & \left(\frac{n}{\eta} - \mu \right) \mathbf{E}_T \Big[D[\mathbf{z}\|\mathbf{u}(k_1^{t-1})] \Big] - \frac{n}{\eta} \mathbf{E}_T \Big[D[\mathbf{z}\|\mathbf{u}(k_1^{t})] \Big] \\ & + n \mathbf{E}_T \Big[Q(\mathbf{u}(k_1^{t-1})) \Big] - n \mathbf{E}_T \Big[Q(\mathbf{u}(k_1^{t})) \Big] \;, \end{aligned} \tag{24}$$

where $\mathbf{E}_T$ is again an expectation with respect to the sequence k_1^T being drawn from the uniform distribution over $[1 \ldots n]^T$. For convenience, define

$$\tilde{\mathcal{Q}}^t \equiv \mathbf{E}_T \Big[Q(\mathbf{u}(k_1^t)) \Big] - Q(\mathbf{z}) \qquad \text{and} \qquad \tilde{\mathcal{D}}^t \equiv \frac{1}{\eta} \mathbf{E}_T \Big[D[\mathbf{z}||\mathbf{u}(k_1^t)] \Big] \;.$$

We may assume that $\tilde{\mathcal{Q}}^t \geq 0$ for all $t \leq T$ since if this is not true the lemma trivially holds.[17] Eq. 24 can be rearranged to give

$$\begin{aligned} \tilde{\mathcal{Q}}^{t-1} &\leq (n - \eta\mu) \tilde{\mathcal{D}}^{t-1} - n \tilde{\mathcal{D}}^t + n \tilde{\mathcal{Q}}^{t-1} - n \tilde{\mathcal{Q}}^t \\ n \tilde{\mathcal{Q}}^t + n \tilde{\mathcal{D}}^t &\leq (n-1) \tilde{\mathcal{Q}}^{t-1} + (n - \eta\mu) \tilde{\mathcal{D}}^{t-1} \;\leq\; (n - \eta\mu) \left(\tilde{\mathcal{Q}}^{t-1} + \tilde{\mathcal{D}}^{t-1} \right) \\ \tilde{\mathcal{Q}}^t + \tilde{\mathcal{D}}^t &\leq \left(1 - \frac{\eta\mu}{n} \right) \left(\tilde{\mathcal{Q}}^{t-1} + \tilde{\mathcal{D}}^{t-1} \right) \;, \end{aligned} \tag{25}$$

17. Note that $\mathbf{E}_T \left[Q(\mathbf{u}(k_1^t)) \right]$ is monotone decreasing since every random sequence of updates results in monotone improvement. The lemma then holds by an argument similar to Appendix B.

where Eq. 25 uses the observation that $n-\eta\mu \geq n-1$ because $\eta\mu \leq 1$ (this follows because $\eta \leq 1/\tau$ and $\mu < \tau$ for some $\tau > 0$). By iterating this result, it follows that

$$\begin{aligned}
\tilde{\mathcal{Q}}^T + \tilde{\mathcal{D}}^T &\leq \left(1-\frac{\eta\mu}{n}\right)^T \left(\tilde{\mathcal{Q}}^0 + \tilde{\mathcal{D}}^0\right) \\
\mathbf{E}_T\left[Q(\mathbf{u}(k_1^T))\right] &\leq Q(\mathbf{z}) + \left(1-\frac{\eta\mu}{n}\right)^T \left(Q(\mathbf{u}^1) - Q(\mathbf{z}) + \frac{1}{\eta} D[\mathbf{z}||\mathbf{u}^1]\right) \\
&\leq Q(\mathbf{z}) + e^{-\eta\mu T/n} \left(Q(\mathbf{u}^1) - Q(\mathbf{z}) + \frac{1}{\eta} D[\mathbf{z}||\mathbf{u}^1]\right) ,
\end{aligned}$$

thus proving the lemma. □

Appendix F. Empirical Comparisons in Terms of Running Time

In this section we compare EG to SGD and L-BFGS in terms of running time. The experiments in the main text provide comparison in terms of "effective" iterations, which do not take into account the computational cost of processing a single example. Here we show that EG maintains its advantages over the other learning algorithms when running time is used as a performance measure, with similar relative improvements to those reported in the main text.

Clearly, any timed comparison depends on the quality of the implementations being compared. Data processing and gradient and objective calculations were performed using the same C++ code for all three algorithms: EG, SGD, and L-BFGS. For L-BFGS, we used the implementation based on Byrd et al. (1995).[18] This code is available online and is written in Fortran. The SGD update is straightforward and we implemented it ourselves in our C++ package. All the timing experiments were performed on a 1.8GHz AMD Opteron™ CPU.

We focus on the log-linear case here, since timing results for the max-margin case were provided in Section 8.

Figures 14 and 15 show results for the MNIST multi-class (see Section 7.1), and the parsing tasks (see Section 7.2) respectively. As in the results in the main text, it can be seen that the EG objective converges faster than the two other algorithms. Also, as in the main text, SGD converges quickly in terms of accuracy, but its objective converges very slowly to the optimum.

Note that the timing of the EG experiments includes the time required to convert the dual parameters to the primal representation. We have found that the EG algorithm is quite fast in practice; in the MNIST task, for example, the EG algorithm requires on average only 10% more time per iteration (including the step-size search) than SGD and L-BFGS. To help explain why EG is able to run almost as fast as SGD, Figure 16 presents pseudocode for the SGD and online EG algorithms. Both SGD and EG share the following operations: (a) inner products between the feature vectors and the primal vector, (b) computation of part-wise marginals, and (c) addition of scaled feature vectors to the primal vector. In the EG algorithm, we require two additional loops over $R(x_i)$ in order to update the dual variables and compute the dual entropy term. In practice, however, the cost of the two additional loops is dominated by the three shared operations mentioned above. Thus, processing a single example takes roughly the same time for EG and SGD. Similar arguments can be used to explain why EG can run almost as fast as L-BFGS.

18. Specifically, we used the code by Zhu, Byrd, Lu, and Nocedal (www.ece.northwestern.edu/∼nocedal/).

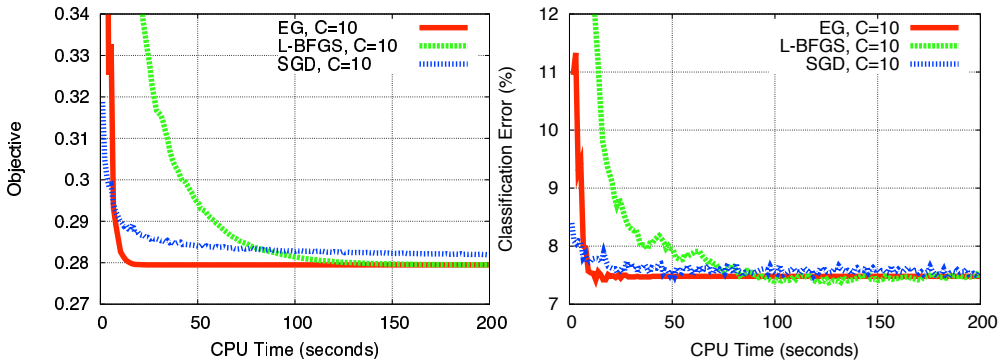

Figure 14: Timing results on the MNIST task, comparing the EG algorithm to L-BFGS and SGD. All algorithms are trained on the log-linear objective function with $C = 10$. The left figure shows objective values and the right figure shows classification error (see Figure 12). The results roughly correspond to 200 effective iterations.

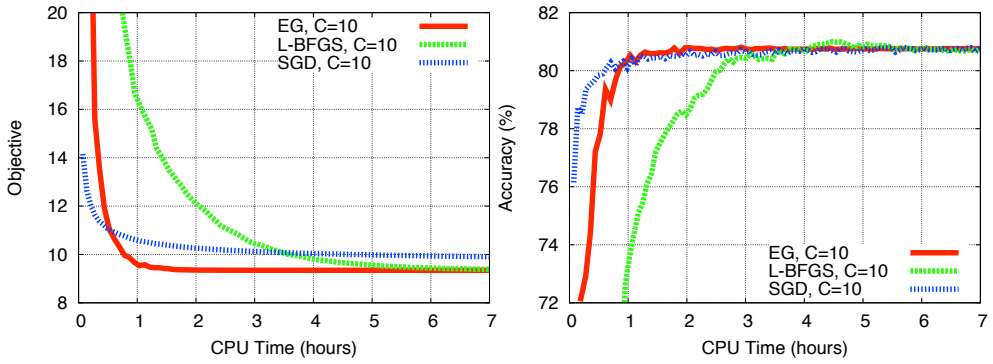

Figure 15: Timing results on the dependency-parsing task, comparing the EG algorithm to L-BFGS and SGD. All algorithms are trained on the log-linear objective function with $C = 10$. The left figure shows objective values and the right figure shows accuracy (see Figure 12). The results roughly correspond to 100 effective iterations.

SGD Update:

◁ *compute part-wise inner products*
1. $\mathbf{q} = \mathbf{0}$
2. **for** $r \in R(x_i)$
3. $\quad q_r = \mathbf{w}^t \cdot \mathbf{f}(x_i, r)$
4. **endfor**

◁ *compute marginals*
5. $\mathbf{m} = \text{MARGINALS}(\mathbf{q})$

◁ *apply gradient of regularizer*
6. $\mathbf{w}^{t+1} = (1 - \eta C)\mathbf{w}^t$

◁ *apply gradient of log-loss*
7. **for** $r \in R(x_i)$
8. $\quad$ **if** $r \in y_i$

◁ *empirical contribution*
9. $\quad\quad \mathbf{w}^{t+1} = \mathbf{w}^{t+1} + \eta \mathbf{f}(x_i, r)$
10. $\quad$ **endif**

◁ *expected contribution*
11. $\quad \mathbf{w}^{t+1} = \mathbf{w}^{t+1} - \eta m_r \mathbf{f}(x_i, r)$
12. **endfor**

Online EG Update:

◁ *compute part-wise inner products*
1. $\mathbf{q} = \mathbf{0}$
2. **for** $r \in R(x_i)$
3. $\quad q_r = \frac{1}{C}\mathbf{w}^t \cdot \mathbf{f}(x_i, r)$
4. **endfor**

◁ *update part-wise duals*
5. **for** $r \in R(x_i)$
6. $\quad s_{i,r}^{t+1} = (1 - \eta) s_{i,r}^t + \eta q_r$
7. **endfor**

◁ *compute marginals and partition function*
8. $(\mu_i^{t+1}, Z) = \text{MARGINALS}(\mathbf{s}_i^{t+1})$

◁ *compute new dual entropy*
9. $H_i^{t+1} = \log Z$
10. **for** $r \in R(x_i)$
11. $\quad H_i^{t+1} = H_i^{t+1} - \mu_{i,r}^{t+1} s_{i,r}^{t+1}$
12. **endfor**

◁ *update primals*
13. $\mathbf{w}^{t+1} = \mathbf{w}^t$
14. **for** $r \in R(x_i)$
15. $\quad \mathbf{w}^{t+1} = \mathbf{w}^{t+1} + (\mu_{i,r}^t - \mu_{i,r}^{t+1}) \mathbf{f}(x_i, r)$
16. **endfor**

◁ *compute change in dual objective*
17. $\delta = \frac{1}{2C}\|\mathbf{w}^{t+1}\|^2 - H_i^{t+1} - (\frac{1}{2C}\|\mathbf{w}^t\|^2 - H_i^t)$

Figure 16: Pseudocode for the updates performed in SGD and online EG for structured log-linear models (note that ◁ denotes a comment). In EG, we maintain dual vectors $\mathbf{s}_i^t$, marginals μ_i^t, entropy values H_i^t, and a vector $\mathbf{w}^t = \mathbf{w}(\mathbf{u}^t)$. Note that line-search techniques can be implemented based on the δ value computed in line 17 of the EG update. A vector scaling operation is required in line 6 of SGD, and vector norm operations are required in line 17 of EG; these can be performed in $O(1)$ time using an appropriate representation (e.g., see Shalev-Shwartz et al., 2007).

References

J. Baker. Trainable grammars for speech recognition. In J.J. Wolf and D.H. Klatt, editors, *Proceedings of the 97th meeting of the Acoustical Society of America*, pages 547–550. Acoustical Society of America, New York, NY, 1979.

P. L. Bartlett, M. Collins, B. Taskar, and D. McAllester. Exponentiated gradient algorithms for large–margin structured classification. In L. K. Saul, Y. Weiss, and L. Bottou, editors, *Advances in Neural Information Processing Systems 17*, pages 113–120, Cambridge, MA, 2005. MIT Press.

A. Beck and M. Teboulle. Mirror descent and nonlinear projected subgradient methods for convex optimization. *Operations Research Letters*, 31:167–175, 2003.

L.M. Bregman. The relaxation method of finding the common point of convex sets and its application to the solution of problems in convex programming. *U.S.S.R. Computational Mathematics and Mathematical Physics*, 7:200–217, 1967.

S. Buchholz and E. Marsi. CoNLL-X shared task on multilingual dependency parsing. In *Proceedings of the 10th Conference on Computational Natural Language Learning*, pages 149–164, New York City, 2006. Association for Computational Linguistics.

R.H. Byrd, P. Lu, and J. Nocedal. A limited memory algorithm for bound constrained optimization. *SIAM Journal on Scientific and Statistical Computing*, 16(5):1190–1208, 1995.

Y. Censor and S.A. Zenios. *Parallel Optimization*. Oxford University Press, 1997.

M. Civit and M. Antònia Martí. Design principles for a Spanish treebank. In *Proceedings of the 1st Workshop on Treebanks and Linguistic Theories*, pages 61–77, 2002.

C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.

T.M. Cover and J.A Thomas. *Elements of Information Theory*. Wiley, 1991.

K. Crammer and Y. Singer. On the algorithmic implementation of multiclass kernel-based vector machines. *Journal of Machine Learning Research*, 2:265–292, 2002.

N. Cristianini, C. Campbell, and J. Shawe-Taylor. Multiplicative updatings for support-vector learning. Technical report, NC-TR-98-016, Neuro COLT, Royal Holloway College, 1998.

A. Globerson, T. Koo, X. Carreras, and M. Collins. Exponentiated gradient algorithms for log-linear structured prediction. In Z. Ghahramani, editor, *Proceedings of the 24th International Conference on Machine Learning*, pages 305–312. ACM Press, New York, NY, 2007.

D. Hush, P. Kelly, C. Scovel, and I. Steinwart. QP algorithms with guaranteed accuracy and run time for support vector machines. *Journal of Machine Learning Research*, 7:733–769, 2006.

T. Jaakkola and D. Haussler. Probabilistic kernel regression models. In D. Heckerman and J. Whittaker, editors, *Proceedings of 7th Workshop on Artificial Intelligence and Statistics*. Morgan Kaufmann, San Francisco, CA, 1999.

T. Joachims. Training linear SVMs in linear time. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 217–226. ACM Press, New York, NY, 2006.

S.S. Keerthi, K.B. Duan, S.K. Shevade, and A. N. Poo. A fast dual algorithm for kernel logistic regression. *Machine Learning*, 61:151–165, 2005.

J. Kivinen and M. Warmuth. Exponentiated gradient versus gradient descent for linear predictors. *Information and Computation*, 132(1):1–63, 1997.

J. Kivinen and M. Warmuth. Relative loss bounds for multidimensional regression problems. *Machine Learning*, 45(3):301–329, 2001.

K. Koh, S.J. Kim, and S. Boyd. An interior point method for large scale l_1-regularized logistic regression. *Journal of Machine Learning Research*, 8:1519–1555, 2007.

T. Koo, A. Globerson, X. Carreras, and M. Collins. Structured prediction models via the matrix-tree theorem. In *Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 141–150. Association for Computational Linguistics, 2007.

J. Lafferty, A. McCallum, and F. Pereira. Conditonal random fields: Probabilistic models for segmenting and labeling sequence data. In C.E. Brodley and A.P. Danyluk, editors, *Proceedings of the 18th International Conference on Machine Learning*, pages 282–289, San Francisco, CA, 2001. Morgan Kaufmann.

G. Lebanon and J. Lafferty. Boosting and maximum likelihood for exponential models. In T.G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems 14*, pages 447–454. MIT Press, Cambridge, MA, 2002.

Y. LeCun, L. Bottou, Y. Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

N. List, D. Hush, C. Scovel, and I. Steinwart. Gaps in support vector optimization. In *Proceedings of the 20th Conference on Learning Theory*, pages 336–348, 2007.

R. McDonald, K. Crammer, and F. Pereira. Online large-margin training of dependency parsers. In *Proceedings of the 43rd Annual Meeting of the ACL*, pages 91–98. Association for Computational Linguistics, 2005.

R. Memisevic. Dual optimization of conditional probability models. Technical report, University of Toronto, 2006.

T. Minka. A comparison of numerical optimizers for logistic regression. Technical report, Carnegie Mellon University, 2003.

M. Mitzenmacher and E. Upfal. *Probability and Computing: Randomized Algorithms and Probabilistic Analysis*. Cambridge University Press, 2005.

A. Nedic and D. P. Bertsekas. Incremental subgradient methods for nondifferentiable optimization. *SIAM Journal on Optimization*, 12(1):109–138, 2001.

J. Platt. Fast training of support vector machines using sequential minimal optimization. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*, pages 41–64. MIT Press, 1998.

F. Sha and F. Pereira. Shallow parsing with conditional random fields. In *Proceedings of the conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, pages 134–141. Association for Computational Linguistics, 2003.

F. Sha, Y. Lin, L.K. Saul, and D.D. Lee. Multiplicative updates for nonnegative quadratic programming. *Neural Computation*, 19(8):2004–2031, 2007.

S. Shalev-Shwartz, Y. Singer, and N. Srebro. Pegasos: Primal estimated sub-gradient solver for SVM. In Z. Ghahramani, editor, *Proceedings of the 24th International Conference on Machine Learning*, pages 807–814. ACM Press, New York, NY, 2007.

B. Taskar, C. Guestrin, and D. Koller. Max margin Markov networks. In S. Thrun, L. Saul, and B. Schölkopf, editors, *Advances in Neural Information Processing Systems 16*, pages 25–32. MIT Press, Cambridge, MA, 2004a.

B. Taskar, D. Klein, M. Collins, D. Koller, and C. Manning. Max-margin parsing. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1–8. Association for Computational Linguistics, 2004b.

B. Taskar, S. Lacoste-Julien, and M. Jordan. Structured prediction, dual extragradient and Bregman projections. *Journal of Machine Learning Research*, pages 1627–1653, 2006.

C.H. Teo, Q. Le, A. Smola, and S.V.N. Vishwanathan. A scalable modular convex solver for regularized risk minimization. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 727–736. ACM Press, New York, NY, USA, 2007.

I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun. Support vector machine learning for interdependent and structured output spaces. In C.E. Brodley, editor, *Proceedings of the 21st International Conference on Machine Learning*, pages 823–830. ACM, New York, NY, 2004.

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 W.W. Cohen and A. Moore, editors, *Proceedings of the 23rd International Conference on Machine Learning*, pages 969–976. ACM Press, New York, NY, 2006.

T. Zhang. On the dual formulation of regularized linear systems with convex risks. *Machine Learning*, 46:91–129, 2002.

J. Zhu and T. Hastie. Kernel logistic regression and the import vector machine. In T.G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems 14*, pages 1081–1088. MIT Press, Cambridge, MA, 2001.