Accelerated Training of Conditional Random Fields with Stochastic Gradient Methods

S.V. N. Vishwanathan SVN.VISHWANATHAN@NICTA.COM.AU
Nicol N. Schraudolph NIC.SCHRAUDOLPH@NICTA.COM.AU
Statistical Machine Learning, National ICT Australia, Locked Bag 8001, Canberra ACT 2601, Australia; and
Research School of Information Sciences & Engr., Australian National University, Canberra ACT 0200, Australia

Mark W. Schmidt SCHMIDTM@CS.UBC.CA
Kevin P. Murphy MURPHYK@CS.UBC.CA
Department of Computer Science, University of British Columbia, Canada

Abstract

We apply Stochastic Meta-Descent (SMD), a stochastic gradient optimization method with gain vector adaptation, to the training of Conditional Random Fields (CRFs). On several large data sets, the resulting optimizer converges to the same quality of solution over an order of magnitude faster than limited-memory BFGS, the leading method reported to date. We report results for both exact and inexact inference techniques.

1. Introduction

Conditional Random Fields (CRFs) have recently gained popularity in the machine learning community (Lafferty et al., 2001; Sha & Pereira, 2003; Kumar & Hebert, 2004). Current training methods for CRFs[1] include generalized iterative scaling (GIS), conjugate gradient (CG), and limited-memory BFGS. These are all batch-only algorithms that do not work well in an online setting, and require many passes through the training data to converge. This currently limits the scalability and applicability of CRFs to large real-world problems. In addition, for many graph structures with large treewidth, such as 2D lattices, computing the exact gradient is intractable. Various approximate inference methods can be employed, but these cause many optimizers to break.

[1] In this paper, "training" specifically means penalized maximum likelihood parameter estimation.

Appearing in *Proceedings of the 23rd International Conference on Machine Learning*, Pittsburgh, PA, 2006. Copyright 2006 by the author(s)/owner(s).

Stochastic gradient methods, on the other hand, are online and scale sub-linearly with the amount of training data, making them very attractive for large data sets; empirically we have also found them more resilient to errors made when approximating the gradient. Unfortunately their asymptotic convergence to the optimum is often painfully slow. Gain adaptation methods like Stochastic Meta-Descent (SMD) accelerate this process by using second-order information to adapt the gradient step sizes (Schraudolph, 1999, 2002). Key to SMD's efficiency is the implicit computation of fast Hessian-vector products (Pearlmutter, 1994; Griewank, 2000).

In this paper we marry the above two techniques and show how SMD can be used to significantly accelerate the training of CRFs. The rest of the paper is organized as follows: Section 2 gives a brief overview of CRFs while Section 3 introduces stochastic gradient methods. We present experimental results for 1D chain CRFs in Section 4, and 2D lattice CRFs in Section 5. We conclude with a discussion in Section 6.

2. Conditional Random Fields (CRFs)

CRFs are a probabilistic framework for labeling and segmenting data. Unlike Hidden Markov Models (HMMs) and Markov Random Fields (MRFs), which model the joint density $\mathbb{P}(X, Y)$ over inputs X and labels Y, CRFs directly model $\mathbb{P}(Y|\boldsymbol{x})$ for a given input sample $\boldsymbol{x}$. Furthermore, instead of maintaining a per-state normalization, which leads to the so-called *label bias problem* (Lafferty et al., 2001), CRFs utilize a global normalization which allows them to take long-range interactions into account.

We now introduce exponential families, and describe CRFs as conditional models in the exponential family.

2.1. Exponential Families

Given $\boldsymbol{x} \in \mathcal{X}$ and $\boldsymbol{y} \in \mathcal{Y}$ (where $\mathcal{Y}$ is a discrete space), a conditional exponential family distribution over $\mathcal{Y}$, parameterized by the natural parameter $\boldsymbol{\theta} \in \Theta$, can be written in its canonical form as

$$p(\boldsymbol{y}|\boldsymbol{x};\boldsymbol{\theta}) = \exp(\langle \phi(\boldsymbol{x},\boldsymbol{y}), \boldsymbol{\theta}\rangle - z(\boldsymbol{\theta}|\boldsymbol{x})). \quad (1)$$

Here $\phi(\boldsymbol{x},\boldsymbol{y})$ is called the sufficient statistics of the distribution, $\langle\cdot,\cdot\rangle$ denotes the inner product, and $z(\cdot)$ the log-partition function

$$z(\boldsymbol{\theta}|\boldsymbol{x}) := \ln \sum_{\boldsymbol{y}} \exp(\langle \phi(\boldsymbol{x},\boldsymbol{y}), \boldsymbol{\theta}\rangle). \quad (2)$$

It is well-known (Barndorff-Nielsen, 1978) that the log-partition function is a C^∞ convex function. Furthermore, it is also the cumulant generating function of the exponential family, *i.e.,*

$$\frac{\partial}{\partial\boldsymbol{\theta}} z(\boldsymbol{\theta}|\boldsymbol{x}) = \mathbb{E}_{p(\boldsymbol{y}|\boldsymbol{x};\boldsymbol{\theta})}[\phi(\boldsymbol{x},\boldsymbol{y})], \quad (3)$$

$$\frac{\partial^2}{(\partial\boldsymbol{\theta})^2} z(\boldsymbol{\theta}|\boldsymbol{x}) = \mathrm{Cov}_{p(\boldsymbol{y}|\boldsymbol{x};\boldsymbol{\theta})}[\phi(\boldsymbol{x},\boldsymbol{y})], \quad \textit{etc.} \quad (4)$$

The sufficient statistics $\phi(\boldsymbol{x},\boldsymbol{y})$ represent salient features of the data, and are typically chosen in an application-dependent manner as part of the CRF design for a given machine learning task.

2.2. Clique Decomposition Theorem

The clique decomposition theorem essentially states that if the conditional density $p(\boldsymbol{y}|\boldsymbol{x};\boldsymbol{\theta})$ factorizes according to a graph G, then the sufficient statistics (or features) $\phi(\boldsymbol{x},\boldsymbol{y})$ decompose into terms over the maximal cliques $\{c_1,\ldots c_n\}$ of G: $\phi(\boldsymbol{x},\boldsymbol{y}) = (\{\phi_c(\boldsymbol{x},\boldsymbol{y}_c)\})$, where c indexes the maximal cliques, and $\boldsymbol{y}_c$ is the label configuration for nodes in clique c.

For ease of notation we will assume that all maximal cliques have size two, *i.e.,* each edge of the graph has a potential associated with it, denoted ϕ_{ij} for an edge between nodes i and j. We will also refer to single-node potentials ϕ_i as *local evidence.*

To reduce the amount of training data required, all cliques share the same parameters $\boldsymbol{\theta}$ (Lafferty et al., 2001; Sha & Pereira, 2003); this is the same parameter tying assumption as used in HMMs. This enables us to compute the sufficient statistics by simply summing the clique potentials over all nodes and edges:

$$\phi(\boldsymbol{x},\boldsymbol{y}) = \left(\sum_{ij\,\in\mathcal{E}} \phi_{ij}(\boldsymbol{x},y_i,y_j), \sum_{i\in\mathcal{N}} \phi_i(\boldsymbol{x},y_i)\right) \quad (5)$$

where $\mathcal{E}$ is the set of edges and $\mathcal{N}$ is the set of nodes.

2.3. Parameter Estimation

Let $X := \{\boldsymbol{x}_i \in \mathcal{X}\}_{i=1}^m$ be a set of m data points and $Y := \{\boldsymbol{y}_i \in \mathcal{Y}\}_{i=1}^m$ be the corresponding set of labels. We assume a conditional exponential family distribution over the labels, and also that they are i.i.d. given the training samples. Thus we can write

$$\mathbb{P}(Y|X;\boldsymbol{\theta}) = \prod_{i=1}^m p(\boldsymbol{y}_i|\boldsymbol{x}_i;\boldsymbol{\theta}) \quad (6)$$
$$= \exp(\sum_{i=1}^m [\langle\phi(\boldsymbol{x}_i,\boldsymbol{y}_i),\boldsymbol{\theta}\rangle - z(\boldsymbol{\theta}|\boldsymbol{x}_i)]).$$

Bayes' rule states that $\mathbb{P}(\boldsymbol{\theta}|X,Y) \propto \mathbb{P}(\boldsymbol{\theta})\,\mathbb{P}(Y|X;\boldsymbol{\theta})$. For computational convenience we assume an isotropic Gaussian prior over the parameters $\boldsymbol{\theta}$, *i.e.,* $\mathbb{P}(\boldsymbol{\theta}) \propto \exp(-\frac{1}{2\sigma^2}||\boldsymbol{\theta}||^2)$ for some fixed σ, and write the negative log-posterior of the parameters given the data and labels, up to a constant, as

$$\mathcal{L}(\boldsymbol{\theta}) := \frac{||\boldsymbol{\theta}||^2}{2\sigma^2} - \sum_{i=1}^m [\langle\phi(\boldsymbol{x}_i,\boldsymbol{y}_i),\boldsymbol{\theta}\rangle - z(\boldsymbol{\theta}|\boldsymbol{x}_i)] \quad (7)$$
$$= -\ln \mathbb{P}(\boldsymbol{\theta}|X,Y) + \text{const.}$$

Maximum a posteriori (MAP) estimation involves maximizing $\mathbb{P}(\boldsymbol{\theta}|X,Y)$, or equivalently minimizing $\mathcal{L}(\boldsymbol{\theta})$. Prediction then utilizes the plug-in estimate $p(\boldsymbol{y}|\boldsymbol{x};\boldsymbol{\theta}^*)$, where $\boldsymbol{\theta}^* = \mathrm{argmin}_{\boldsymbol{\theta}}\, \mathcal{L}(\boldsymbol{\theta})$.

2.4. Gradient and Expectation

As stated in Section 2.3, to perform MAP estimation we need to minimize $\mathcal{L}(\boldsymbol{\theta})$. For this purpose we compute its gradient $\boldsymbol{g}(\boldsymbol{\theta}) := \frac{\partial}{\partial\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta})$. Differentiating (7) with respect to $\boldsymbol{\theta}$ and substituting (3) yields

$$\boldsymbol{g}(\boldsymbol{\theta}) = \frac{\boldsymbol{\theta}}{\sigma^2} - \sum_{i=1}^m \left[\phi(\boldsymbol{x}_i,\boldsymbol{y}_i) - \mathbb{E}_{p(\boldsymbol{y}|\boldsymbol{x}_i;\boldsymbol{\theta})}[\phi(\boldsymbol{x}_i,\boldsymbol{y})]\right]. \quad (8)$$

which has the familiar form of features minus expected features. The expected feature vector for each clique,

$$\mathbb{E}_{p(\boldsymbol{y}|\boldsymbol{x};\boldsymbol{\theta})}\left[\phi(\boldsymbol{x},\boldsymbol{y})\right] = \sum_{\boldsymbol{y}\in\mathcal{Y}} p(\boldsymbol{y}|\boldsymbol{x};\boldsymbol{\theta})\phi(\boldsymbol{x},\boldsymbol{y}) \quad (9)$$

can be computed in $O(N|\mathcal{Y}|^w)$ time using dynamic programming, where N is the number of nodes and w is the *treewidth* of the graph, *i.e.,* the size of its largest clique after the graph has been optimally triangulated. For chains and (undirected) trees, $w = 2$, so this computation is usually fairly tractable, at least for small state spaces. For cases where this is intractable, we discuss various approximations in Section 2.6. Since we assume all the variables are fully observed during training, the objective function is convex, so we can find the global optimum.

2.5. Hessian and Hessian-Vector Product

In addition to the gradient, second-order methods based on Newton steps also require computation and inversion of the Hessian $\boldsymbol{H}(\boldsymbol{\theta}) := \frac{\partial^2}{(\partial \boldsymbol{\theta})^2} \mathcal{L}(\boldsymbol{\theta})$. Taking the gradient of (8) *wrt.* $\boldsymbol{\theta}$ and substituting (4) yields

$$\boldsymbol{H}(\boldsymbol{\theta}) = \frac{\mathrm{I}}{\sigma^2} + \sum_{i=1}^{m} \mathrm{Cov}_{p(\boldsymbol{y}|\boldsymbol{x}_i;\boldsymbol{\theta})} \phi(\boldsymbol{x}_i, \boldsymbol{y}). \tag{10}$$

Explicitly computing the full Hessian (let alone inverting it) costs $O(n^2)$ time per iteration, where n is the number of features (sufficient statistics). In our 1-D chain CRFs (Section 4) $n > 10^5$, making this approach prohibitively expensive. Our 2-D grid CRFs (Section 5) have few features, but computing the Hessian there requires the pairwise marginals $\mathbb{P}(\boldsymbol{x}_i, \boldsymbol{x}_j|\boldsymbol{y})\ \forall i,j$, which is $O(|\mathcal{Y}|^{2k})$ for an $k \times k$ grid, again infeasible for the problems we are looking at.

Our SMD optimizer (given below) instead makes use of the differential

$$\mathrm{d}\boldsymbol{g}(\boldsymbol{\theta}) = \boldsymbol{H}(\boldsymbol{\theta})\, \mathrm{d}\boldsymbol{\theta} \tag{11}$$

to efficiently compute the product of the Hessian with a chosen vector $\boldsymbol{v} =: \mathrm{d}\boldsymbol{\theta}$ by forward-mode algorithmic differentiation (Pearlmutter, 1994; Griewank, 2000). Such Hessian-vector products are implicit — *i.e.,* they never calculate the Hessian itself — and can be computed along with the gradient at only 2–3 times the cost of the gradient computation alone.

In fact the similarity between differential and complex arithmetic (*i.e.,* addition and multiplication) implies

$$\boldsymbol{g}(\boldsymbol{\theta} + i\,\epsilon\, \mathrm{d}\boldsymbol{\theta}) = \boldsymbol{g}(\boldsymbol{\theta}) + O(\epsilon^2) + i\,\epsilon\, \mathrm{d}\boldsymbol{g}(\boldsymbol{\theta}), \tag{12}$$

so for suitably small ϵ (say, 10^{-150}) we can effectively compute the Hessian-vector product in the imaginary part of the gradient function extended to the complex plane (Pearlmutter, personal communication). We use this technique in the experiments reported below.

2.6. Approximate Inference and Learning

Since we assume that all the variables are observed in the training set, we can find the global optimum of the objective function, so long as we can compute the gradient exactly. Unfortunately for many CRFs the treewidth is too large for exact inference (and hence exact gradient computation) to be tractable. The treewidth of an $N = k \times k$ grid, for instance, is $w = O(2k)$ (Lipton & Tarjan, 1979), so exact inference takes $O(|\mathcal{Y}|^{2k})$ time. Various approximate inference methods have been used in parameter learning algorithms (Parise & Welling, 2005). Here we consider two of the simplest: mean field (MF) and loopy belief propagation (LBP) (Weiss, 2001; Yedidia et al., 2003). The MF free energy is a lower bound on the log-likelihood, and hence an upper bound on our negative log-likelihood objective. The Bethe free energy minimized by LBP is not a bound, but has been found empirically to often better approximate the log-likelihood than the MF free energy (Weiss, 2001). Although LBP can sometimes oscillate, convergent versions have been developed (*e.g.,* Kolmogorov, 2004).

For some kinds of potentials, one can use graph cuts (Boykov et al., 2001) to find an approximate MAP estimate of the labels, which can be used inside a Viterbi training procedure. However, this produces a very discontinuous estimate of the gradient (though one could presumably use methods similar to Collins' (2002) voted perceptron to smoothe this out). For the same reason, we use the sum-product version of LBP rather than max-product.

An alternative to trying to approximate the conditional likelihood (CL) is to change the objective function. The pseudo-likelihood (PL) proposed by Besag (1986) has the significant advantage that it only requires normalizing over the possible labels at *one* node:

$$\hat{\theta}_{PL} = \arg\max_{\theta} \sum_{m} \sum_{i} \ln p(y_i^m | \boldsymbol{y}_{N_i}^m, \boldsymbol{x}^m, \theta), \tag{13}$$

where N_i are the neighbors of node i, and

$$p(y_i^m | \boldsymbol{y}_{N_i}^m, \boldsymbol{x}^m, \theta) = \frac{\phi_i(y_i^m)}{z_i(\boldsymbol{x}^m, \theta)} \prod_{j \in N_i} \phi_{ij}(y_i^m, y_j^m), \tag{14}$$

$$z_i(\boldsymbol{x}^m, \theta) = \sum_{y_i} \phi_i(y_i) \prod_{j \in N_i} \phi_{ij}(y_i^m, y_j^m). \tag{15}$$

Here y_i^m is the observed label for node i in the m'th training case, and z_i sums over all possible labels for node i. We have dropped the conditioning on $\boldsymbol{x}^m$ in the potentials for notational simplicity.

Although the pseudo-likelihood is not necessarily a good approximation to the likelihood, as the amount of training data (or the size of the lattice, when using tied parameters) tends to infinity, its maximum coincides with that of the likelihood (Winkler, 1995).

Note that pseudo-likelihood estimates the parameters conditional on i's neighbors being observed. As a consequence, PL tends to place too much emphasis on the edge potentials, and not enough on the local evidence. For image denoising problems, this is often evident as "oversmoothing". The "frailty" of pseudo-likelihood in learning to segment images was also noted by Blake et al. (2004). Regularizing the edge parameters does help, but as we show in Section 5, it is often better to try to optimize the correct objective function.

3. Stochastic Gradient Methods

In this section we describe stochastic gradient descent and discuss how its convergence can be improved by gain vector adaptation via the Stochastic Meta-Descent (SMD) algorithm (Schraudolph, 1999, 2002).

3.1. Stochastic Approximation of Gradients

Since the log-likelihood (7) is summed over a potentially large number m of data points, we approximate it by subsampling *batches* of $b \ll m$ points:

$$\mathcal{L}(\boldsymbol{\theta}) \approx \sum_{t=0}^{\frac{m}{b}-1} \mathcal{L}_b(\boldsymbol{\theta}, t), \quad \text{where} \tag{16}$$

$$\mathcal{L}_b(\boldsymbol{\theta}, t) = \frac{b\|\boldsymbol{\theta}_t\|^2}{2m\sigma^2} - \sum_{i=1}^{b} [\langle \phi(\boldsymbol{x}_{bt+i}, \boldsymbol{y}_{bt+i}), \boldsymbol{\theta}_t \rangle - z(\boldsymbol{\theta}_t | \boldsymbol{x}_{bt+i})]. \tag{17}$$

Note that for $\boldsymbol{\theta}_t =$ const. (16) would be exact. We will, however, interleave an optimization step that modifies $\boldsymbol{\theta}$ with each evaluation of $\mathcal{L}_b(\boldsymbol{\theta}, t)$, *resp.* its gradient

$$\boldsymbol{g}_t := \tfrac{\partial}{\partial \boldsymbol{\theta}} \mathcal{L}_b(\boldsymbol{\theta}, t). \tag{18}$$

The batch size b controls the stochasticity of the approximation. At one extreme, $b = m$ recovers the conventional deterministic algorithm; at the other, $b = 1$ adapts $\boldsymbol{\theta}$ fully online, based on individual data samples. Typically small batches of data ($5 \leq b \leq 20$) are found to be computationally most efficient.

Unfortunately most advanced gradient methods do not tolerate the sampling noise inherent in stochastic approximation: it collapses conjugate search directions (Schraudolph & Graepel, 2003) and confuses the line searches that both conjugate gradient and quasi-Newton methods depend upon. Full second-order methods are unattractive here because the computational cost of inverting the Hessian is better amortized over a large data set.

This leaves plain first-order gradient descent. Though this can be very slow to converge, the speed-up gained by stochastic approximation dominates on large, redundant data sets, making this strategy more efficient overall than even sophisticated deterministic methods. The convergence of stochastic gradient descent can be further improved by *gain vector adaptation.*

3.2. SMD Gain Vector Adaptation

Consider a stochastic gradient descent where each coordinate of $\boldsymbol{\theta}$ has its own positive gain:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \boldsymbol{\eta}_t \cdot \boldsymbol{g}_t, \tag{19}$$

where $\boldsymbol{\eta}_t \in \mathbb{R}^n_+$, and $\cdot$ denotes component-wise (Hadamard) multiplication. The gain vector $\boldsymbol{\eta}$ serves as a diagonal conditioner; it is simultaneously adapted via a multiplicative update with meta-gain μ:

$$\boldsymbol{\eta}_{t+1} = \boldsymbol{\eta}_t \cdot \max(\tfrac{1}{2}, 1 - \mu\, \boldsymbol{g}_{t+1} \cdot \boldsymbol{v}_{t+1}), \tag{20}$$

where the vector $\boldsymbol{v} \in \Theta$ characterizes the long-term dependence of the system parameters on gain history over a time scale governed by the decay factor $0 \leq \lambda \leq 1$. It is computed by the simple iterative update

$$\boldsymbol{v}_{t+1} = \lambda \boldsymbol{v}_t - \boldsymbol{\eta}_t \cdot (\boldsymbol{g}_t + \lambda \boldsymbol{H}_t \boldsymbol{v}_t), \tag{21}$$

where $\boldsymbol{H}_t \boldsymbol{v}_t$ is calculated efficiently via (11). Since $\boldsymbol{\theta}_0$ does not depend on any gains, $\boldsymbol{v}_0 = 0$. SMD thus introduces two scalar tuning parameters, with typical values (for stationary problems) $\mu = 0.1$ and $\lambda = 1$; see Vishwanathan et al. (2006) for a detailed derivation.

4. Experiments on 1D Chain CRFs

We have applied SMD as described in Section 3, comprising Equations (19), (20), and (21), to the training of CRFs as described in Section 2, using the stochastic gradient (18). The Hessian-vector product $\boldsymbol{H}_t \boldsymbol{v}_t$ in (21) is computed efficiently alongside the gradient by forward-mode algorithmic differentiation using the differential (11) with $d\boldsymbol{\theta} := \boldsymbol{v}_t$.

We implemented this by modifying the CRF++ software[2] developed by Taku Kudo. We compare the convergence of SMD to three control methods:

- Simple stochastic gradient descent (SGD) with a fixed gain $\boldsymbol{\eta}_0$,
- the batch-only limited-memory BFGS algorithm as supplied with CRF++, storing 5 BFGS corrections, and
- Collins' (2002) perceptron (CP), a fully online update ($b = 1$) that optimizes a different objective.

Except for CP — which in our implementation required far more time per iteration than the other methods — we repeated each experiment several times, obtaining for different random permutations of the data substantially identical results to those reported below.

4.1. CoNLL-2000 Base NP Chunking Task

Our first experiment uses the well-known CoNLL-2000 Base NP chunking task (Sang & Buchholz, 2000). Text

[2] Available under LGPL from http://chasen.org/~taku/software/CRF++/. Our modified code, as well as the data sets, configuration files, and results for all experiments reported here will be available for download from http://sml.nicta.com.au/code/crfsmd/.

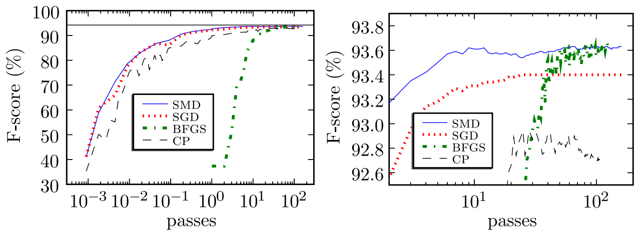

Figure 1. Left: F-scores on the CoNLL-2000 shared task, against passes through the training set, for SMD (solid), SGD (dotted), BFGS (dash-dotted), and CP (dashed). Horizontal line: F-score reported by Sha & Pereira (2003). Right: Enlargement of the final portion of the figure.

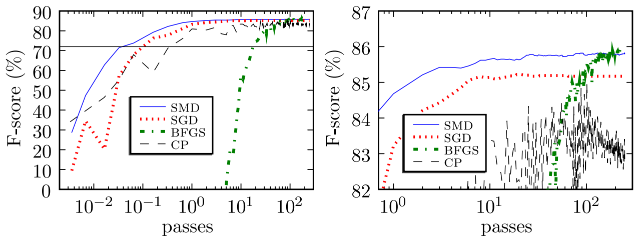

Figure 2. Left: F-scores on the BioNLP/NLPBA-2004 shared task, against passes through the training set. Horizontal line: best F-score reported by Settles (2004). Right: Enlargement of the final portion of the figure.

chunking, an intermediate step towards full parsing, consists of dividing a text into syntactically correlated parts of words. The training set consists of 8936 sentences, each word annotated automatically with part-of-speech (POS) tags. The task is to label each word with a label indicating whether the word is outside a chunk, starts a chunk, or continues a chunk. The standard evaluation metrics for this task are the precision p (fraction of output chunks which match the reference chunks), recall r (fraction of reference chunks returned), and their harmonic mean, the F-score given by $F = 2pr/(p + r)$, on a test set of 2012 sentences.

Sha & Pereira (2003) found BFGS to converge faster than CG and GIS methods on this task. We follow them in using binary-valued features which depend on the words, POS tags, and labels in the neighborhood of a given word, taking into account only those 330731 features which occur at least once in the training data. The main difference between our setup and theirs is that they assume a second-order Markov dependency between chunk tags, which we do not model.

We used $\sigma = 1$ and $b = 8$, and tuned $\boldsymbol{\eta}_0 = 0.1$ for best performance of SGD on this task. SMD then used the same $\boldsymbol{\eta}_0$ and the default values $\mu = 0.1$ and $\lambda = 1$. We evaluated the F-score on the test set after every batch during the first pass through the training data, after every iteration through the data thereafter. The result is plotted in Figure 1 as a function of the number of iterations through the data on a logarithmic scale.

Figure 1 shows the F-score obtained on the test set by the three algorithms as a function of the number of passes through the training set, on a logarithmic scale. The online methods show progress orders of magnitude earlier, simply because unlike batch methods they start optimizing long before having seen the entire training set even once.

Enlarging the final portion of the data reveals differences in asymptotic convergence between the online methods: While SMD and BFGS both attain the same F-score of 93.6% — compared to Sha & Pereira's (2003) 94.2% for a richer model — SMD does so almost an order of magnitude faster than BFGS. SGD levels out at around 93.4%, while CP declines to 92.7% from a peak of 92.9% reached earlier.

4.2. BioNLP/NLPBA-2004 Shared Task

Our second experiment uses the BioNLP/NLPBA-2004 shared task of biomedical named-entity recognition on the GENIA corpus (Kim et al., 2004). Named-entity recognition aims to identify and classify technical terms in a given domain (here: molecular biology) that refer to concepts of interest to domain experts (Kim et al., 2004). Following Settles (2004) we use binary orthographic features (AlphaNumeric, HasDash, RomanNumeral, etc.) based on regular expressions, though ours differ somewhat from those used by Settles (2004). We also use neighboring words to model context, and add features to capture correlations between the current and previous label, for a total of 106583 features that occur in the training data.

We permuted the 18546 sentences of the training data set so as to destroy any correlations across sentences, used the parameters $\sigma = 1$ and $b = 6$, and tuned $\boldsymbol{\eta}_0 = 0.1$ for best performance of SGD. SMD then used the same $\boldsymbol{\eta}_0$, $\mu = 0.02$ (moderately tuned), and $\lambda = 1$ (default value). Figure 2 plots the F-score, evaluated on the 3856 sentences of the test set, against number of passes through the training data.

Settles (2004) trained a CRF on this data and report a best F-score of 72.0%. Our asymptotic F-scores are far better; we attribute this to our use of different regular expressions, and a richer set of features. Again SMD converges much faster to the same solution as BFGS (85.8%), significantly outperforming SGD (85.2%) and CP, whose oscillations are settling around 83%.

We deliberately chose a large value for the initial step size $\boldsymbol{\eta}_0$ in order to demonstrate the merits of step size adaptation. In other experiments (not reported here) SGD with a smaller value of $\boldsymbol{\eta}_0$ converged to the same quality of solution as SMD, albeit at a far slower rate.

We also obtained comparable results (not reported here) with a similar setup on the first BioCreAtivE (Critical Assessment of Information Extraction in Biology) challenge task 1A (Hirschman et al., 2005).

5. Experiments on 2D Lattice CRFs

For the 2D CRF experiments we compare four optimization algorithms: SGD, SMD, BFGS as implemented in Matlab's `fminunc` function (with 'largeScale' set to 'off'), and stochastic gradient with scalar gain annealed as $\eta_t = \eta_0/t$ (ASG). Note that full BFGS converges at least as fast as a limited-memory approximation. While the stochastic methods use only the gradient (8), `fminunc` also needs the value of the objective, and hence must compute the log-partition function (2), with the attendant computational cost. We also briefly experimented with conjugate gradient optimization (as implemented in Carl Rasmussen's `minimize` function), but found this to be slower and give worse results than `fminunc`.

We use these algorithms to optimize the conditional likelihood (CL) as approximated by loopy belief propagation (LBP) or mean field (MF), and the pseudo-likelihood (PL) which can be computed exactly. We apply this to the two data sets used by Kumar & Hebert (2004), using our own matlab/C code.[3]

For all experiments we plot the training objective (negative log-likelihood) and test error (pixel misclassification rate) against the number of passes through the data. The test error is computed by using the learned parameters to estimate the MAP node labels given a test image. In particular, we run sum-product belief propagation until convergence, and then compute the max marginals, $y_i^* = \arg\max_{y_i} p(y_i|\boldsymbol{x}, \boldsymbol{\theta})$. We also limit the number of loopy BP iterations (parallel node updates) to 200; LBP converged before this limit most of the time, but not always.

For the local evidence potentials, we follow Kumar & Hebert (2004) in using $\phi_{ij}(y_i, y_j) = \exp(y_i y_j \boldsymbol{\theta}_{\mathcal{E}}^\top \boldsymbol{h}_{ij})$, where $y_i = \pm 1$ is node i's label, and $\boldsymbol{h}_{ij}$ the feature vector of edge ij. The node potentials were likewise set to $\phi_i(y_i) = \exp(y_i \boldsymbol{\theta}_{\mathcal{N}}^\top \boldsymbol{h}_i)$. We initialize node potentials by logistic regression, and edge potentials to $\boldsymbol{\theta}_{\mathcal{E}} = 0.5$.

[3] The code will be made available at `http://www.cs.ubc.ca/~murphyk/Software/CRFs.html`.

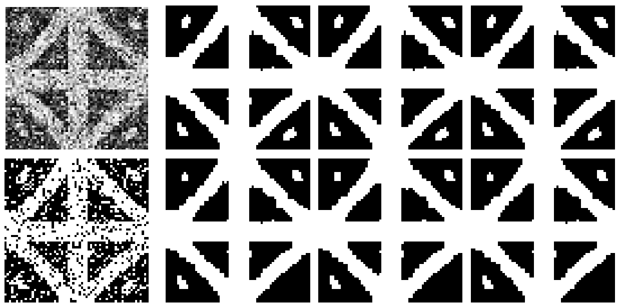

Figure 3. A noisy binary image (top left) and various attempts to denoise it. Bottom left: logistic regression; columns 2–4: BFGS, SGD and SMD using LBP (top row) *vs.* MF (bottom row) approximation. PL did not work.

Note that due to the small number of features here, the $O(m)$ calls to inference for each pass through the data dominate the runtime of all algorithms.

After some parameter tuning, we set $\sigma = 1$, $\lambda = 0.9$, and $\mu = 10\eta_0$, where the initial gain η_0 is set as high as possible for each method while maintaining stability: 0.0001 for LBP, 0.001 for MF, and 0.04 for PL.

5.1. Binary Image Denoising

This experiment uses 64×64 binary images of hand-drawn shapes, with artificial Gaussian noise added; see Figure 3 for an example chosen at random. The task is to denoise the image, *i.e.,* to recover the underlying binary image. The node features are $\boldsymbol{h_i} = [1, s_i]$, where s_i is the pixel intensity at location i; for edge features we use $\boldsymbol{h}_{ij} = [1, |s_i - s_j|]$. Hence in total there are 2 parameters per node and edge. We use 40 images for online ($b = 1$) training and 10 for testing.

Figure 3 shows that the CL criterion outperforms logistic regression (which does not enforce spatial smoothness), and that the LBP approximation to CL gives better results than MF. PL oversmoothed the entire image to black.

In Figure 4 we plot training objective and test error percentage against the number of passes through the data. With LBP (top row), SMD and SGD converge faster than BFGS, while the annealed stochastic gradient (ASG) is slower. Eventually all achieve the same performance, both in training objective and test error. Generalization performance worsens slightly under the MF approximation to CL (middle row) but breaks down completely under the PL criterion (bottom row), with a test error above 50%. This is probably because most of the pixels are black, and since pseudolikelihood tends to overweight its neighbors, black pixels get propagated across the entire image.

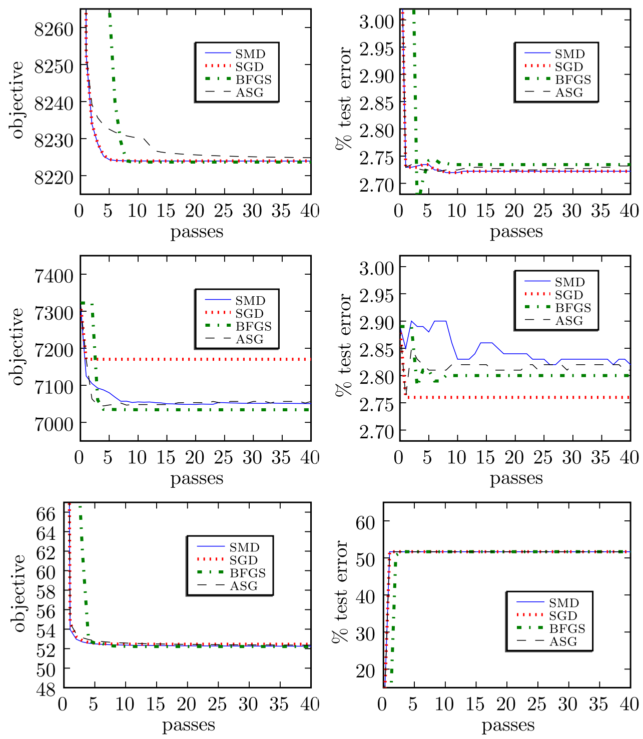

Figure 4. Training objective (left) and percent test error (right) against passes through the data, for binary image denoising with (rows, top to bottom) LBP, MF, and PL.

5.2. Classifying Image Patches

This dataset consists of real images of size 256×384 from the Corel database, divided into 24×16 patches. The task is to classify each patch as containing "man-made structure" or background. For the node features we took a 5-dimensional feature vector computed from the edge orientation histogram (EOH), and performed a quadratic kernel expansion, resulting in a 21-dimensional $\boldsymbol{h}_i$ vector. For the edge features we used $\boldsymbol{h}_{ij} = [1, |\boldsymbol{m}_i - \boldsymbol{m}_j|]$, where $\boldsymbol{m}_i$ is a 14-dimensional multi-scale EOH feature vector associated with patch i; see Kumar & Hebert (2003) for further details on the features. Hence the total number of parameters is 21 per node, and 15 per edge. We use a batchsize of $b = 3$, with 129 images for training and 129 for testing.

In Figure 6, we see that SGD and SMD are initially faster than BFGS, but eventually the latter catches up. We also see that ASG's $\eta_t = \eta_0/t$ annealing schedule does not work well for this particular problem, while its fixed gain $\eta_t = \eta_0$ prevents SGD from reaching the global optimum in the PL case (bottom row). One advantage of SMD is that its annealing schedule is adaptive. The LBP and MF approximations to CL perform slightly better than PL on the test set; all optimizers achieve similar final generalization performance here.

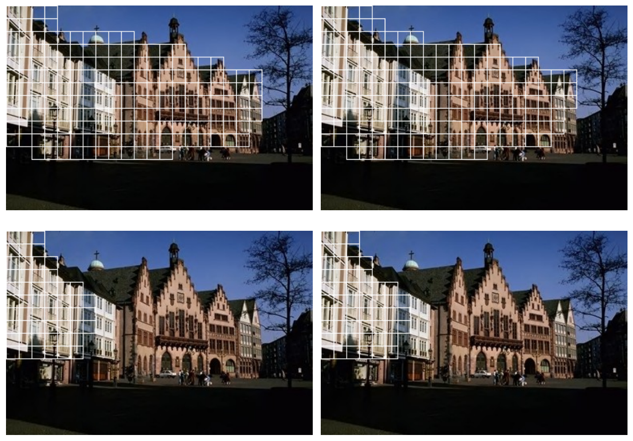

Figure 5. A natural image (chosen at random from the test set) with patches of man-made structure highlighted, as classified via LBP (top) *vs.* PL (bottom) objectives optimized by BFGS (left) *vs.* SMD (right).

6. Outlook and Discussion

In the cases where exact inference is possible (1D CRFs and PL objective for 2D CRFs), we have shown that stochastic gradient methods in general, and SMD in particular, are considerably more efficient than BFGS, which is generally considered the method of choice for training CRFs. When exact inference cannot be performed, stochastic gradient methods appear sensitive to appropriate scheduling of the gain parameter(s); SMD does this automatically. The magnitude of the performance gap between the stochastic methods and BFGS is largely a function of the training set size; we thus expect the scaling advantage of stochastic gradient methods to dominate in our 2D experiments as well as we scale them up.

The idea of stochastic training is not new; for instance, it has been widely used to train neural networks. It does not seem popular in the CRF community, however, perhaps because of the need to carefully adapt gains — the simple annealing schedule we tried did not always work. By providing automatic gain adaptation, the SMD algorithm can make stochastic gradient methods easier to use and more widely applicable.

Acknowledgments

National ICT Australia is funded by the Australian Government's Department of Communications, Information Technology and the Arts and the Australian Research Council through Backing Australia's Ability and the ICT Center of Excellence program. This work is also supported by the IST Program of the European Community, under the Pascal Network of Excellence, IST-2002-506778, and an NSERC Discovery Grant.

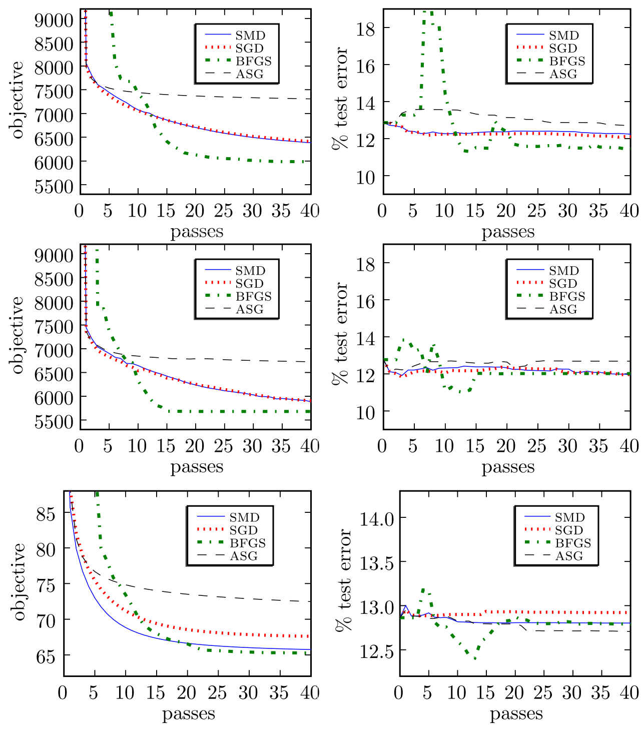

Figure 6. Training objective (left) and percent test error (right) *vs.* passes through the data, for classifying image patches with (rows, top to bottom) LBP, MF, and PL.

References

Barndorff-Nielsen, O. E. (1978). *Information and Exponential Families in Statistical Theory.* New York: John Wiley and Sons.

Besag, J. (1986). On the statistical analysis of dirty pictures. *Journal of the Royal Statistical Society B*, *48*(3), 259–302.

Blake, A., Rother, C., Brown, M., Perez, P., & Torr, P. (2004). Interactive image segmentation using an adaptive GMMRF model. In *Proc. European Conf. on Computer Vision.*

Boykov, Y., Veksler, O., & Zabih, R. (2001). Fast approximate energy minimization via graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *23*(11), 1222–1239.

Collins, M. (2002). Discriminative training methods for hidden Markov models. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing.*

Griewank, A. (2000). *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation.* Frontiers in Applied Mathematics. Philadelphia: SIAM.

Hirschman, L., Yeh, A., Blaschke, C., & Valencia, A. (2005). Overview of BioCreAtivE: Critical assessment of information extraction for biology. *BMC Bioinformatics*, *6*(Suppl 1).

Kim, J.-D., Ohta, T., Tsuruoka, Y., Tateisi, Y., & Collier, N. (2004). Introduction to the bio-entity recognition task at JNLPBA. In *Proc. Intl. Joint Workshop on Natural Language Processing in Biomedicine and its Applications (NLPBA)*, 70–75. Geneva, Switzerland.

Kolmogorov, V. (2004). Convergent tree-reweighted message passing for energy minimization. Tech. Rep. MSR-TR-2004-90, Microsoft Research, Cambridge, UK.

Kumar, S., & Hebert, M. (2003). Man-made structure detection in natural images using a causal multiscale random field. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition.*

Kumar, S., & Hebert, M. (2004). Discriminative fields for modeling spatial dependencies in natural images. In S. Thrun, L. Saul, & B. Schölkopf, eds., *Advances in Neural Information Processing Systems 16.*

Lafferty, J. D., McCallum, A., & Pereira, F. (2001). Conditional random fields: Probabilistic modeling for segmenting and labeling sequence data. In *Proc. Intl. Conf. Machine Learning*, vol. 18, 282–289. San Francisco, CA: Morgan Kaufmann.

Lipton, R. J., & Tarjan, R. E. (1979). A separator theorem for planar graphs. *SIAM Journal of Applied Mathematics*, *36*, 177–189.

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

Pearlmutter, B. A. (1994). Fast exact multiplication by the Hessian. *Neural Computation*, *6*(1), 147–160.

Sang, E. F. T. K., & Buchholz, S. (2000). Introduction to the CoNLL-2000 shared task: Chunking. In *Proc. Conf. Computational Natural Language Learning*, 127–132. Lisbon, Portugal.

Schraudolph, N. N. (1999). Local gain adaptation in stochastic gradient descent. In *Proc. Intl. Conf. Artificial Neural Networks*, 569–574. Edinburgh, Scotland: IEE, London.

Schraudolph, N. N. (2002). Fast curvature matrix-vector products for second-order gradient descent. *Neural Computation*, *14*(7), 1723–1738.

Schraudolph, N. N., & Graepel, T. (2003). Combining conjugate direction methods with stochastic approximation of gradients. In C. M. Bishop, & B. J. Frey, eds., *Proc. 9th Intl. Workshop Artificial Intelligence and Statistics*, 7–13. Key West, Florida. ISBN 0-9727358-0-1.

Settles, B. (2004). Biomedical named intity recognition using conditional random fields and rich feature sets. In *Proceedings of COLING 2004, International Joint Workshop On Natural Language Processing in Biomedicine and its Applications (NLPBA).* Geneva, Switzerland.

Sha, F., & Pereira, F. (2003). Shallow parsing with conditional random fields. In *Proceedings of HLT-NAACL*, 213–220. Edmonton, Canada: Association for Computational Linguistics.

Vishwanathan, S. V. N., Schraudolph, N. N., & Smola, A. J. (2006). Step size adaptation in reproducing kernel Hilbert space. *J. Mach. Learn. Res.*, *7*, 1107–1133.

Weiss, Y. (2001). Comparing the mean field method and belief propagation for approximate inference in MRFs. In D. Saad, & M. Opper, eds., *Advanced Mean Field Methods.* MIT Press.

Winkler, G. (1995). *Image Analysis, Random Fields and Dynamic Monte Carlo Methods.* Springer Verlag.

Yedidia, J., Freeman, W., & Weiss, Y. (2003). Understanding belief propagation and its generalizations. In *Exploring Artificial Intelligence in the New Millennium*, chap. 8, 239–236. Science & Technology Books.