Non-Uniform Stochastic Average Gradient Method for Training Conditional Random Fields

Mark Schmidt
University of British Columbia

Ann Clifton
Simon Fraser University

Anoop Sarkar
Simon Fraser University

Abstract

This paper explores using a stochastic average gradient (SAG) algorithm for training conditional random fields (CRFs). The SAG algorithm is the first general stochastic gradient algorithm to have a linear convergence rate. However, despite its success on simple classification problems, when applied to CRFs the algorithm requires too much memory because it requires storing a previous gradient with respect to every training example. In this work we show that SAG algorithms can be tractably applied to large-scale CRFs by tracking the marginals over vertices and edges in the graphical model. We also incorporate a simple non-uniform adaptive sampling scheme that learns how often we should sample each training point. Our experimental results reveal that this method significantly outperforms existing methods.

1 Conditional Random Fields

Conditional random fields (CRFs) [9] are a ubiquitous tool in natural language processing. They are used for part-of-speech tagging [12], semantic role labeling [1], topic modeling [27], information extraction [16], shallow parsing [19], named-entity recognition [18], and a variety of applications in other fields such as computer vision [15]. Despite the widespread use of CRFs and major advances in fitting them, the time needed for numerical optimization in CRF models remains a bottleneck in many application.

Lafferty et al. [9] proposed training CRFs with an iterative scaling algorithm (2), but this proved to be inferior to generic optimization strategies like the limited-memory quasi-Newton algorithm L-BFGS [24, 19]. The bottleneck in quasi-Newton methods is that we must do inference on all training examples on every iteration. Due to the high cost of inference on even a single training example example, it is now common to train CRFs using stochastic gradient methods, which work online and only process one example at a time [23, 5]. However, stochastic gradient methods have sub-linear convergence rates. Collins et al. [2] proposed an online exponentiated gradient algorithm with a linear convergence rate in the dual problem, but the practical performance of this method degrades without strong regularization, see [2, Figures 5-6 and Table 3] and [8, Figure 1]. Hybrid stochastic/quasi-Newton methods have also been explored [6], but these methods eventually process every example on every iteration. In this work, we consider training CRFs with the stochastic average gradient (SAG) algorithm of Le Roux et al. [11], a general stochastic gradient method that achieves a linear convergence rate for finite training sets and has shown impressive empirical performance for binary classification. Beyond the faster convergence rate, the SAG method also has effective heuristics for issues that have traditionally frustrated users of stochastic gradient methods: *setting the step-size* and *deciding when to stop*. Our results indicate that the SAG algorithm with a simple adaptive step-size and non-uniform sampling strategy can outperform previous training methods by an order of magnitude.

2 Stochastic Average Gradient for Conditional Random Fields

CRFs predict a structured objective $y \in \mathcal{Y}$ (such as a sequence of labels) given an input $x \in \mathcal{X}$ (such as a sequence of words or characters), based on a vector of features $F(x, y)$ and parameters w using

$$p(y|x, w) = \frac{\exp(w^T F(x, y))}{Z}, \tag{1}$$

where Z is the normalizing constant. Given n pairs $\{x_i, y_i\}$ comprising our training set, the standard approach to training CRFs is to minimize the ℓ_2-regularized negative log-likelihood,

$$\min_w f(w) = \frac{1}{n}\sum_{i=1}^{n} -\log p(y_i|x_i, w) + \frac{\lambda}{2}\|w\|^2, \tag{2}$$

where $\lambda > 0$ is the regularization strength. Evaluating $p(y_i|x_i, w)$ is expensive due to the normalizing constant Z in (1) that sums over all assignments to y. E.g., in chain-structured models we compute $\log p(y_i|x_i, w)$ and its gradient by running the forward-backward algorithm.

The SAG algorithm has the form

$$w^{t+1} = w^t - \frac{\alpha_t}{n}\sum_{i=1}^{n} s_i^t, \tag{3}$$

where α_t is the step-size and at each iteration we set $s_i^t = -\nabla \log p(y_i|x_i, w^t) + \lambda w^t$ for one randomly chosen data point and keep the remaining s_i^t the same. The surprising aspect of the work of [11, 17] is that this algorithm achieves a similar convergence rate to the classic full gradient algorithm despite the iterations being n times faster.

Algorithm 1 SAG algorithm for training CRFs

Require: $\{x_i, y_i\}$, λ, w, δ
$d = 0$, $L_g = 1$, $m = 0$, $g_i = 0$ for $i = 1, 2, \ldots, n$
while $m < n$ and $\|\frac{1}{n}d + \lambda w\|_\infty \geq \delta$ **do**
  Sample i from $\{1, 2, \ldots, n\}$
  $f = -\log p(y_i|x_i, w), \quad g = -\nabla \log p(y_i|x_i, w)$
  if this is the first time we sampled i **then**
    $m = m + 1$
  end if
  $d = d + g - g_i, \quad g_i = g$
  if $\|g_i\|^2 > 10^{-8}$ **then**
    L_g =lineSearch$(x_i, y_i, f, g_i, w, L_g)$
  end if
  $\alpha = 1/(L_g + \lambda), \quad w = (1 - \alpha\lambda)w - \frac{\alpha}{m}d, \quad L_g = L_g \cdot 2^{-1/n}$
end while

3 Implementation

Algorithm 1 outlines a practical variant of the SAG algorithm for training CRFs, that modifies the update (3) using suggestions in Le Roux et al. [11]. One of the core modifications is using the update

$$w^{t+1} = w^t - \alpha(\frac{1}{m}\sum_{i=1}^{n} g_i^t + \lambda w^t) = (1 - \alpha\lambda)w^t - \frac{\alpha}{m}d, \tag{4}$$

where g_i^t is the value of $-\nabla \log p(y_i|x_i, w^k)$ for the last iteration k where i was selected and d is the sum of the g_i^t over all i. Thus, this update uses the exact regularizer and only approximes the CRF log-likelihoods.This update also uses the number of examples seen, m, rather than the total number of examples n (since we initialize to $g_i^0 = 0$). We set the step-size to $\alpha = 1/L = 1/(L_g + \lambda)$, where L is the Lipschitz constant of f' and L_g is the constant for the gradient of the CRF log-likelihood. To avoid computing L_g we use the line-search from the code of Schmidt et al. [17]. We can use the magnitude of $(\frac{1}{n}d + \lambda w^t)$ to decide when to stop, since it converges to $\nabla f(w^t)$.

By using the structure in the CRF gradients, we can reduce the memory requirement of Algorithm 1 so that it only depends on the number possible labels and the number of vertices and edges in the

graphical model. This is best illustrated with an example. Consider the common case where we have a set of features x_j at position y_j in a sequence y of length V, and we define our conditional probability as

$$p(y|x,w) = \frac{\exp(F(y,x,w))}{Z(w)}, \quad \text{where } F(y,x,w) = \sum_{j=1}^{V} x_j^T w_{y_j} + \sum_{j=1}^{V-1} w_{y_j,y_{j+1}}.$$

Here we have a vector w_k used to combine the features x_j to give the potential for state y_j, and we have a set of transition weights $w_{k,k'}$ for each combination of transitions between k and k'. In this setting, the log-probability takes the form

$$\log p(y|x,w) = F(y,x,w) - \log(Z(w)).$$

If we take the gradient with respect to a particular vector w_k we obtain

$$\nabla_{w_k} \log\ p(y|x,w) = \sum_{j=1}^{V} x_j \left[\mathbb{I}(y_j = k) - p(y_j = k|x,w)\right],$$

where $p(y_j = k|x,w)$ is the marginal probability of $y_j = k$ given x and w,

$$p(y_j = k|x,w) = \sum_{\{y'|y_j=k\}} p(y'|x,w).$$

Notice that Algorithm 1 only depends on the old gradient through its difference with the new gradient (line 8), which in our example gives

$$\nabla_{w_k} \log p(y|x,w) - \nabla_{w_k} \log p(y|x,w_{\text{old}}) = \sum_{j=1}^{V} x_j [p(y_j = k|x,w_{\text{old}}) - p(y_j = k|x,w)],$$

where w is the current parameter vector and w_{old} is the old parameter vector. Thus, to perform this calculation the only thing we need to know about w_{old} is the unary marginals $p(y_j = k|x,w_{\text{old}})$, while to perform the full gradient difference we will also need the pairwise marginals $p(y_j = k, y_{k+1} = k'|x,w_{\text{old}})$ to update the transition parameters $w_{k,k'}$. For general pairwise graphical model structures, the memory requirements to store these marginals will be $O(VK + EK^2)$, where V is the number of vertices and E is the number of edges. Further, since computing these marginals is a by-product of computing the gradient, this potentially-enormous reduction in the memory requirements comes at no extra computational cost.

Recently, several works show that we can improve the convergence rates of randomized optimization algorithms by using non-uniform sampling (NUS) schemes. This incldes randomized Kaczmarz [21], randomized coordinate descent [14], and stochastic gradient methods [13]. Schmidt et al. [17] argue that faster convergence rates might be achieved with NUS for SAG since it allows a larger step size α. The key idea behind all of these NUS strategies is to *bias the sampling towards the Lipschitz constants*, so we sample more often things that change quickly and sample things that change slowly less often. We explored a simple NUS strategy where with probability 0.5 we sample an example uniformly, and with probability 0.5 we sample proportional to our approximations L_i of the Lipschitz constants of the gradients of the examples. We update these approximations L_i using the line-search as in the code of Schmidt et al. [17]. Unlike this code, we initialize each L_i with the current average of the L_i (rather than 1, which reduces the number of backtracking operations), and we only multiply L_i by 0.9 after each iteration (instead of 0.5, which also reduces the number of expensive backtracking steps needed), and we simply set the step-size to $\alpha = \frac{1}{2}\left(\frac{1}{L_{\text{max}}} + \frac{1}{L_{\text{mean}}}\right)$.

4 Experiments

We compared a wide variety of approaches on four CRF training tasks: an optical character recognition (OCR) dataset [22], the CoNLL-2000 shallow parse chunking dataset,[1] the CoNLL-2002 Dutch named-entity recognition dataset,[2] and POS-tagging task using the Penn Treebank Wall Street

[1] http://www.cnts.ua.ac.be/conll2000/chunking

[2] http://www.cnts.ua.ac.be/conll2002/ner

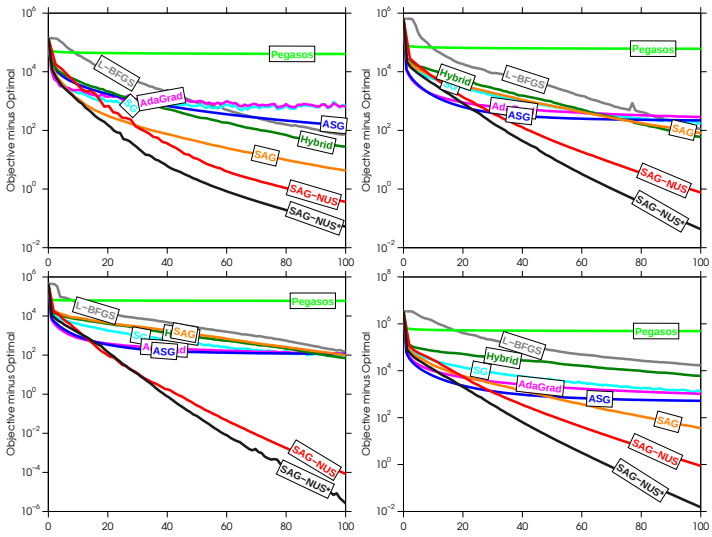

Figure 1: Objective minus optimal objective value against effective number of passes for different deterministic, stochastic, and semi-stochastic optimization strategies. Top-left: OCR, Top-right: CoNLL-2000, bottom-left: CoNLL-2002, bottom-right: POS-WSJ.

Journal data (POS-WSJ). We compared four classic stochastic gradient methods: *Pegasos* which is a standard stochastic gradient method with a step-size of $\alpha = \eta/\lambda t$ on iteration t [20], a basic stochastic gradient (*SG*) method where we use a constant $\alpha = \eta$, an averaged stochastic gradient (*ASG*) method where we use a constant step-size $\alpha = \eta$ and average the iterations, and *AdaGrad* where we use the per-variable $\alpha_j = \eta/(\delta + \sqrt{\sum_{i=1}^{t} \nabla_j \log p(y_i|x_i, w^i)^2})$ and the proximal-step with respect to the ℓ_2-regularizer [4]. Since setting the step-size is a notoriously hard problem when applying stochastic gradient methods, we let these classic stochastic gradient cheat by choosing the η which gives the best performance among powers of 10 on the training data (and we set $\delta = 1$). Our comparisons also included a deterministic *L-BFGS* algorithm and the *Hybrid* L-BFGS/stochastic algorithm of [6]. Finally, we included the *SAG* algorithm using our proposed strategy for controlling the memory, the *SAG-NUS* variant of [17], and finally *SAG-NUS** which uses the modified NUS strategy from the previous seciton.

Figure 1 shows the result of our experiments on the training objective. Here we measure the number of 'effective passes', meaning $(1/n)$ times the number of times we performed the bottleneck operation of computing $\log p(y_i|x_i, w)$ and its gradient (this dominates the runtime of all algorithms). We observe that *SG* outperformed *Pegasos*, *ASG* outperformed *AdaGrad*, and *Hybrid* outperformed *L-BFGS*. None of the three competitive algorithms *ASG*/*Hybrid*/*SAG* dominated the others, but both *SAG-NUS* methods outperform all other methods by a substantial margin based on the training objective. We omit the test error plot due to lack of space, but we note that *SAG-NUS** always performed as well and in one case much better than the best alternative method in terms of test error.

5 Discussion

An alternate way to reduce the memory requirements would be to group examples into mini-batches, or to use memory-free linearly-convergent stochastic gradient methods [7, 26]. This algorithm has been analyzed with NUS [25], but we found it less effective when combined with NUS in this setting, in addition to doubling the computational cost per iteration. The SAG algorithm could be modified to use multi-threaded computation as in the algorithm of [10], and indeed might be well-suited to distributed parallel implementations. We have also explored using the SAGA algorithm [3] and found it gave similar performance, and we are working on an analysis of this algorithm with NUS. Our experiments show that our algorithm converges faster than other competing methods and reaches the optimal test error at the same speed or faster than optimallty-tuned stochastic gradient methods, but without the actual tuning that often frustrates practitioners.

References

[1] T. Cohn and P. Blunsom. Semantic role labelling with tree conditional random fields. In *In Proceedings of CoNLL-2005*, pages 169–172, 2005.

[2] M. Collins, A. Globerson, T. Koo, X. Carreras, and P. Bartlett. Exponentiated gradient algorithms for conditional random fields and max-margin Markov networks. *The Journal of Machine Learning Research*, 9:1775–1822, 2008.

[3] A. Defazio, F. Bach, and S. Lacoste-Julien. Saga: A fast incremental gradient method with support for non-strongly convex composite objectives. *Advances in Neural Information Processing Systems*, 2014.

[4] J. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12:2121–2159, 2011.

[5] J. R. Finkel, A. Kleeman, and C. D. Manning. Efficient, feature-based, conditional random field parsing. In *Proceedings of ACL-08: HLT*, pages 959–967, 2008.

[6] M. P. Friedlander and M. Schmidt. Hybrid deterministic-stochastic methods for data fitting. *SIAM Journal of Scientific Computing*, 34(3):A1351–A1379, 2012.

[7] R. Johnson and T. Zhang. Accelerating stochastic gradient descent using predictive variance reduction. *Advances in Neural Information Processing Systems*, 2013.

[8] S. Lacoste-Julien, M. Jaggi, M. Schmidt, and P. Pletscher. Block-coordinate frank-wolfe optimization for structural svms. *International Conference on Machine Learning*, 2013.

[9] J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. *International Conference on Machine Learning*, 2001.

[10] T. Lavergne, O. Cappé, and F. Yvon. Practical very large scale CRFs. In *Proceedings the 48th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 504–513, 2010.

[11] N. Le Roux, M. Schmidt, and F. Bach. A stochastic gradient method with an exponential convergence rate for strongly-convex optimization with finite training sets. *Advances in Neural Information Processing Systems*, 2012.

[12] A. McCallum, K. Rohanimanesh, and C. Sutton. Dynamic conditional random fields for jointly labeling multiple sequences. In *NIPS Workshop on Syntax, Semantics, Statistics*, 2003.

[13] D. Needell, N. Srebro, and R. Ward. Stochastic gradient descent and the randomized Kaczmarz algorithm. *arXiv preprint*, 2013.

[14] Y. Nesterov. Efficiency of coordinate descent methods on huge-scale optimization problems. *CORE Discussion Paper*, 2010.

[15] S. Nowozin and C. H. Lampert. Structured learning and prediction in computer vision. *Foundation and Trends in Computer Vision*, 6, 2011.

[16] F. Peng and A. McCallum. Information extraction from research papers using conditional random fields. *Information Processing & Management*, 42(4):963–979, 2006.

[17] M. Schmidt, N. Le Roux, and F. Bach. Minimizing finite sums with the stochastic average gradient. *arXiv preprint*, 2013.

[18] B. Settles. Biomedical named entity recognition using conditional random fields and rich feature sets. In *Proceedings of the International Joint Workshop on Natural Language Processing in Biomedicine and its Applications*, pages 104–107, 2004.

[19] F. Sha and F. Pereira. Shallow parsing with conditional random fields. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 134–141. Association for Computational Linguistics, 2003.

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

[21] T. Strohmer and R. Vershynin. A randomized Kaczmarz algorithm with exponential convergence. *Journal of Fourier Analysis and Applications*, 15(2):262–278, 2009.

[22] B. Taskar, C. Guestrin, and D. Koller. Max-margin Markov networks. *Advances in Neural Information Processing Systems*, 2003.

[23] S. Vishwanathan, N. N. Schraudolph, M. W. Schmidt, and K. P. Murphy. Accelerated training of conditional random fields with stochastic gradient methods. *Iinternational conference on Machine learning*, pages 969–976, 2006.

[24] H. Wallach. Efficient training of conditional random fields. Master's thesis, University of Edinburgh, 2002.

[25] L. Xiao and T. Zhang. A proximal stochastic gradient method with progressive variance reduction. *arXiv preprint arXiv:1403.4699*, 2014.

[26] L. Zhang, M. Mahdavi, and R. Jin. Linear convergence with condition number independent access of full gradients. *Advances in Neural Information Processing Systems*, 2013.

[27] J. Zhu and E. Xing. Conditional Topic Random Fields. In *International Conference on Machine Learning*, 2010.