Stochastic Optimization for CRF Autoencoders

Waleed Ammar & Fan Yang
Carnegie Mellon University

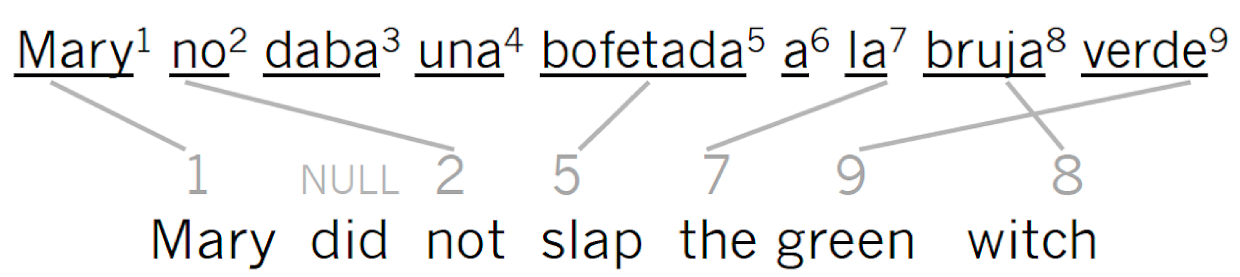
INTRODUCTION

Goal: Implement existing stochastic optimization methods in the context of conditional random field (CRF) autoencoders.

Model: CRF autoencoders are a class of probabilistic models which was designed to address unsupervised and semi-supervised problems in natural language processing. For concreteness, we will focus on a particular instantiation of CRF autoencoders for the classic problem of bitext *word alignment*.

Problem description: Given an observed sentence pair $(\mathbf{s}, \mathbf{t})$, we model the alignment variables $(\mathbf{a})$ and a reconstruction of the target sentence $(\hat{\mathbf{t}})$ as follows:

$$p(\mathbf{a}, \hat{\mathbf{t}} \mid \mathbf{s}, \mathbf{t}) = p_\lambda(\mathbf{a} \mid \mathbf{s}, \mathbf{t}) p_\theta(\hat{\mathbf{t}} \mid \mathbf{s}, \mathbf{a})$$
$$= \frac{\exp \lambda^\top \sum_{i=1}^{n_t} f(a_i, a_{i-1}, \mathbf{s}, \mathbf{t})}{\sum_{\mathbf{a}'} \exp \lambda^\top \sum_{i=1}^{n_t} f(a'_i, a'_{i-1}, \mathbf{s}, \mathbf{t})} \times \prod_{i=1}^{n_t} \theta_{\hat{t}_i \mid s_{a_i}}$$



Objective: We optimize the parameters of the CRF autoencoder model by maximizing the conditional log-likelihood of generating the correct reconstruction of target sentences $(\hat{\mathbf{t}})$, given a pair of source and target sentences $(\mathbf{s}, \mathbf{t})$, marginalizing out the word alignment variables $(\mathbf{a})$, as follows:

$$\text{maximize}_{\lambda,\theta} \quad \sum_{\langle \mathbf{s},\mathbf{t},\hat{\mathbf{t}}\rangle} \log \sum_{\mathbf{a}} p_{\lambda,\theta}(\mathbf{a}, \hat{\mathbf{t}} \mid \mathbf{s}, \mathbf{t})$$
$$\text{subject to} \quad \sum_{t \in \mathcal{V}_t} \theta_{t|s} = 1, 0 \le \theta_{.|s} \le 1, \forall s \in \mathcal{V}_s$$

OPTIMIZATION

In theory: the problem is **non-convex**.
In practice: locally-optimal solutions have been found to be useful, provided that we start with a good initialization for model parameters.

Optimizing λ using L-BFGS vs. SGD

The sufficient statistics needed for one iteration of L-BFGS require expensive computations and abundant memory since the training sets for word alignments tend to be large. Since we use L-BFGS to solve for the optimal λ inside an outer loop of block-coordinate descent, we cannot afford to spend too much time optimizing λ. Instead, we proposed to use stochastic gradient descent (SGD) and update λ according to an approximation of the gradient based on a few sentence pairs.

Intuition: one epoch (i.e., full pass over the training set) of SGD constitutes **many updates**, and incurs the same runtime cost as **one update** of L-BFGS.

Optimizing θ using batch vs. online EM

Expectation Maximization (EM) is a popular method for optimizing parameters of models with latent variables. In each iteration of batch EM, we update θ by solving: $\min_\theta E_{\theta^{old}}[\log p_\theta(\mathbf{a}, \hat{\mathbf{t}} \mid \mathbf{s}, \mathbf{t})]$ subject to the multinomial distribution constraints on θ. In order to update θ more frequently, we use online EM (Cappe and Moulines, 2009). The three algorithms are outlined below, reproduced from (Liang and Klein, 2009):

Batch EM:

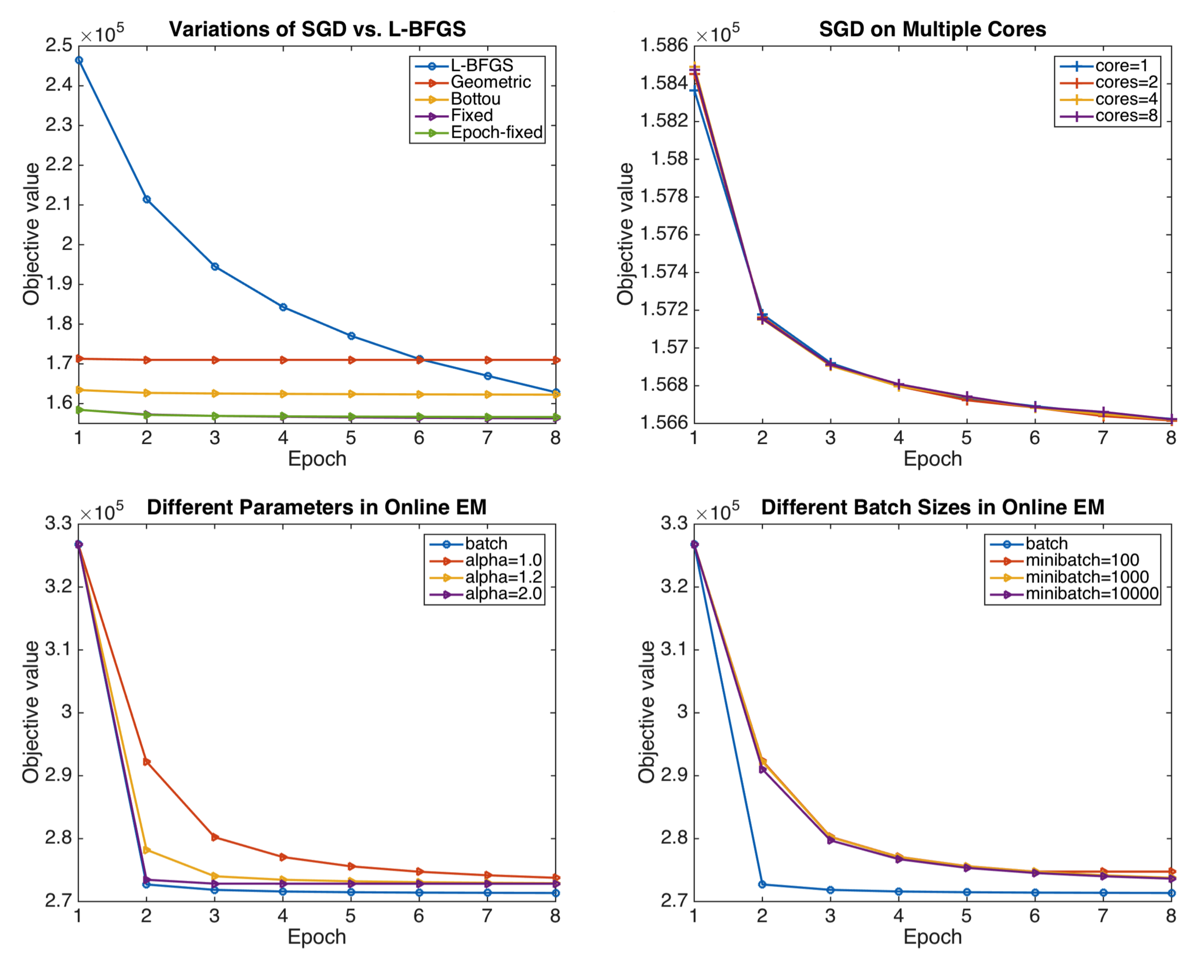
$\mu :=$ initialize
for each EM iteration $t = 1, \dots, T$:
– $\mu' := 0$
– for each example $i : \langle \mathbf{s}, \mathbf{t}, \hat{\mathbf{t}} \rangle$
—- $m'_i := \sum_{\mathbf{a}} p(\mathbf{a} \mid \mathbf{s}, \mathbf{t}, \hat{\mathbf{t}}; \theta(\mu)) \phi(\mathbf{a}, \mathbf{s}, \mathbf{t}, \hat{\mathbf{t}})$[inference]
—- $\mu' := \mu' + m'_i$ [accumulate new]
– $\mu := \mu'$ [replace old with new]

Online EM:

$\mu :=$ initialize, $k := 0$
for each EM iteration $t = 1, \dots, T$:
– for each example $i : \langle \mathbf{s}, \mathbf{t}, \hat{\mathbf{t}} \rangle$ in random order
—- $m'_i := \sum_{\mathbf{a}} p(\mathbf{a} \mid \mathbf{s}, \mathbf{t}, \hat{\mathbf{t}}; \theta(\mu)) \phi(\mathbf{a}, \mathbf{s}, \mathbf{t}, \hat{\mathbf{t}})$ [inference]
—- $\mu := (1 - \eta_k)\mu + \eta_k m'_i; k := k + 1$ [interpolate]

In the algorithm, μ is a vector of expected counts for each element in θ, ϕ is a function that maps a sentence pair and its alignment to a vector of sufficient statistics, and m'_i are the expected counts for a given sentence pair.

RESULTS



CONCLUSIONS

- Convergence to approximate solution with SGD is much faster than with L-BFGS.
- Using epoch-fixed learning rate in SGD has the best performance, similar to L-BFGS after many iterations.
- SGD can be scaled to multiple processors (asynchronous updates) with little loss of accuracy.
- Batch EM converges much faster than online EM.