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

\[
\max_{\lambda, \theta} \sum_{(s, t, \hat{t})} \log \sum_a p_s(a \mid s, t) p_t(\hat{t} \mid s, a)
\]

subject to

\[
\sum_{i \in V_s} \theta_{i,s} = 1, 0 \leq \theta_{i,s} \leq 1, \forall s \in V_s
\]

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

\[
p(a, \hat{t} \mid s, t) = p_s(a \mid s, t)p_t(\hat{t} \mid s, a)
= \exp \lambda \sum_{n} f(a, a_{-n}, s, t) \times \prod_{n} \theta_{n,s}
\]

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 \((s, t)\), we model the alignment variables \((a)\) and a reconstruction of the target sentence \((\hat{t})\) as follows:

\[
p(a, \hat{t} \mid s, t) = p_s(a \mid s, t)p_t(\hat{t} \mid s, a)
= \exp \lambda \sum_{n} f(a, a_{-n}, s, t) \times \prod_{n} \theta_{n,s}
\]

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 \(\lambda\) 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 \(\lambda\) inside an outer loop of block-coordinate descent, we cannot afford to spend too much time optimizing \(\lambda\). Instead, we proposed to use stochastic gradient descent (SGD) and update \(\lambda\) 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 \(\theta\) 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 \(\theta\) by solving: \(\min_{\theta} E_{p(a)}[\log p_s(a \mid s, t)]\) subject to the multinomial distribution constraints on \(\theta\). In order to update \(\theta\) more frequently, we use online EM (Cappe and Moulines, 2009). The three algorithms are outlined below, reproduced from (Liang and Klein, 2009):

Batch EM:

\[
\mu := \text{initialize}, \quad k := 0
\]

for each EM iteration \(t = 1, \ldots, T:\)

\[
- \mu^t := 0
- \text{for each example } i : (s, t, \hat{t})\text{ in random order}
- m_i^t := \sum_a p[a \mid s, t, \hat{t} | \theta(\mu)] \phi(a, s, t, \hat{t}) \text{[inference]}
- \mu^t := \mu^t + m_i^t \text{[accumulate new]}
- \mu := (1 - \eta) \mu + \eta \mu^t, \quad k := k + 1 \text{ [interpolate]}
\]

Online EM:

\[
\mu := \text{initialize}, \quad k := 0
\]

for each EM iteration \(t = 1, \ldots, T:\)

\[
- \mu^t := 0
- \text{for each example } i : (s, t, \hat{t})\text{ in random order}
- m_i^t := \sum_a p[a \mid s, t, \hat{t} | \theta(\mu)] \phi(a, s, t, \hat{t}) \text{[inference]}
- \mu := (1 - \eta) \mu + \eta m_i^t, \quad k := k + 1 \text{ [interpolate]}
\]

In the algorithm, \(\mu\) is a vector of expected counts for each element in \(\theta\), \(\phi\) is a function that maps a sentence pair and its alignment to a vector of sufficient statistics, and \(m_i^t\) are the expected counts for a given sentence pair.

RESULTS

- Batch EM converges much faster than online EM.
- 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.

CONCLUSIONS