A Convex Alternative to IBM Model 2

Andrei Simion
Columbia University
IEOR Department
New York, NY, 10027
aas2148@columbia.edu

Michael Collins
Columbia University
Computer Science
New York, NY, 10027
mc3354@columbia.edu

Clifford Stein
Columbia University
IEOR Department
New York, NY, 10027
cs2035@columbia.edu

Abstract

The IBM translation models have been hugely influential in statistical machine translation; they are the basis of the alignment models used in modern translation systems. Excluding IBM Model 1, the IBM translation models, and practically all variants proposed in the literature, have relied on the optimization of likelihood functions or similar functions that are non-convex, and hence have multiple local optima. In this paper we introduce a convex relaxation of IBM Model 2, and describe an optimization algorithm for the relaxation based on a subgradient method combined with exponentiated-gradient updates. Our approach gives the same level of alignment accuracy as IBM Model 2.

1 Introduction

The IBM translation models (Brown et al., 1993) have been tremendously important in statistical machine translation (SMT). The IBM models were the first generation of SMT systems; in recent work, they play a central role in deriving alignments used within many modern SMT approaches, for example phrase-based translation models (Koehn, 2008) and syntax-based translation systems (e.g., (Chiang, 2005; Marcu et al., 2006)). Since the original IBM paper, there has been a large amount of research exploring the original IBM models and modern variants (e.g., (Moore, 2004; Liang et al., 2006; Toutanova and Galley, 2011; Riley and Gildea, 2012; Vogel et al., 1996)).

Excluding IBM Model 1, the IBM translation models, and practically all variants proposed in the literature, have relied on the optimization of likelihood functions or similar functions that are non-convex. Unfortunately, non-convex objective functions have multiple local optima, and finding a global optimum of a non-convex function is typically a computationally intractible problem. Typically, an EM algorithm is used, which often runs in a reasonable amount of time, but with no guarantees of finding a global optima (or for that matter, even a near-optimal solution).

In this paper we make the following contributions:

- We introduce a convex relaxation of IBM Model 2. At a very high level, the relaxation is derived by replacing the product $t(f_j|e_i) \times d(i|j)$ with a relaxation that is commonly used in the linear programming literature (e.g., see (Bertsimas, 1997; Bertsimas and Tsitsiklis, 1997; Martins et al., 2010)). (Here $t(f|e)$ are the translation parameters of the model, and $d(i|j)$ are the distortion parameters; the product is non-linear, effectively introducing non-convexity into the problem.)
- We describe an optimization algorithm for the relaxed objective, based on a combination of stochastic subgradient methods with the exponentiated-gradient (EG) algorithm (Kivinen and Warmuth, 1997; Beck and Teboulle, 2003).
- We describe experiments with the method on standard alignment datasets, showing that the EG algorithm converges in only a few passes over the data, and that our method achieves accuracies that are very similar to those of IBM Model 2.

Framing the unsupervised learning of alignment models as a convex optimization problem, with guaranteed convergence to a global optimum, has several clear advantages. First, the method is easier to analyze, as the objective function is being truly maximized. Second, there is no need for initialization heuristics with the approach, given that the method will always converge to a global optimum. Finally, we expect that our convexity-based approach may facilitate the further development of more convex models. There has been a rich

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 1574–1583,
Seattle, Washington, USA, 18-21 October 2013. ©2013 Association for Computational Linguistics

interplay between convex and non-convex methods in machine learning: as one example consider the literature on classification problems, with early work on the perceptron (linear/convex), then work on neural networks with back-propagation (non-linear/non-convex), then the introduction of support vector machines (non-linear/convex), and finally recent work on deep belief networks (non-linear/non-convex). In view of these developments, the lack of convex methods in translation alignment models has been noticeable, and we hope that our work will open up new directions and lead to further progress in this area.

Notation. Throughout this paper, for any integer N, we use $[N]$ to denote $\{1 \dots N\}$ and $[N]_0$ to denote $\{0 \dots N\}$.

2 Related Work

(Brown et al., 1993) introduced IBM Models 1 through 5, and optimization methods for these models based on the EM algorithm. While the models were originally introduced for full translation, they are now mainly used to derive alignments which are then used by phrase-based and other modern SMT systems. Since the original IBM models were introduced, many variants have been introduced in the literature. (Vogel et al., 1996) introduced a model, sometimes referred to as IBM 2.5, which uses a parameterization that is similar to a hidden Markov model, and which allows the value of each alignment variable to be conditioned on a previous alignment variable. (Liang et al., 2006) describe a method that explicitly incorporates agreement preferences during training. (Och and Ney, 2003) give a systematic comparison of several alignment models in the literature. (Moore, 2004) gives a detailed study of IBM Model 1, showing various steps that can be used to improve its performance. (Ganchev et al., 2010) describes a method based on posterior regularization that incorporates additional constraints within the EM algorithm for estimation of IBM models. All of these approaches are unsupervised, in that they do not require labeled alignment data; however several authors have considered supervised models (e.g., see (Lacoste-Julien et al., 2006; Taskar et al., 2005; Haghighi et al., 2009)). The focus of the current paper is on unsupervised learning; the unsupervised variants described above all make use of non-convex objective functions during training, with the usual problems with multiple local maxima.

3 The IBM Model 1 and Model 2 Optimization Problems

In this section we give a brief review of IBM Models 1 and 2, and the optimization problems arising from these models. The standard approach for optimization within these models is the EM algorithm.

Throughout this section, and the remainder of the paper, we assume that our set of training examples is $(e^{(k)}, f^{(k)})$ for $k = 1 \dots n$, where $e^{(k)}$ is the k'th English sentence and $f^{(k)}$ is the k'th French sentence. Following standard convention, we assume the task is to translate from *French* (the "source" language) into *English* (the "target" language). We use E to denote the English vocabulary (set of possible English words), and F to denote the French vocabulary. The k'th English sentence is a sequence of words $e_1^{(k)} \dots e_{l_k}^{(k)}$ where l_k is the length of the k'th English sentence, and each $e_i^{(k)} \in E$; similarly the k'th French sentence is a sequence $f_1^{(k)} \dots f_{m_k}^{(k)}$ where each $f_j^{(k)} \in F$. We define $e_0^{(k)}$ for $k = 1 \dots n$ to be a special NULL word (note that E contains the NULL word). Finally, we define $L = \max_{k=1}^n l_k$ and $M = \max_{k=1}^n m_k$.

For each English word $e \in E$, we will assume that $D(e)$ is a *dictionary* specifying the set of possible French words that can be translations of e. The set $D(e)$ is a subset of F. In practice, $D(e)$ can be derived in various ways; in our experiments we simply define $D(e)$ to include all French words f such that e and f are seen in a translation pair.

Given these definitions, the IBM model 2 optimization problem is given in Figure 1. The parameters in this problem are $t(f|e)$ and $d(i|j)$. The $t(f|e)$ parameters are *translation* parameters specifying the probability of English word e being translated as French word f. The *distortion* parameters $d(i|j)$ specify the probability of the j'th French word in a sentence being aligned to the i'th English word. We use a variant of IBM Model 2 where the distortion variables are shared across all sentence lengths (similar variants have been used in (Liang et al., 2006) and (Koehn, 2008)). The objective function is then

Input: Define E, F, L, M, $(e^{(k)}, f^{(k)}, l_k, m_k)$ for $k = 1 \ldots n$, $D(e)$ for $e \in E$ as in Section 3.

Parameters:
- A parameter $t(f|e)$ for each $e \in E, f \in D(e)$.
- A parameter $d(i|j)$ for each $i \in [L]_0, j \in [M]$.

Constraints:

$$\forall e \in E, f \in D(e), \qquad t(f|e) \geq 0 \tag{1}$$

$$\forall e \in E, \qquad \sum_{f \in D(e)} t(f|e) = 1 \tag{2}$$

$$\forall i \in [L]_0, j \in [M], \qquad d(i|j) \geq 0 \tag{3}$$

$$\forall j \in [M], \qquad \sum_{i \in [L]_0} d(i|j) = 1 \tag{4}$$

Objective: Maximize

$$\frac{1}{n} \sum_{k=1}^{n} \sum_{j=1}^{m_k} \log \sum_{i=0}^{l_k} t(f_j^{(k)}|e_i^{(k)}) d(i|j) \tag{5}$$

with respect to the $t(f|e)$ and $d(i|j)$ parameters.

Figure 1: The IBM Model 2 Optimization Problem.

the log-likelihood of the training data (see Eq. 5):

$$\frac{1}{n} \sum_{k=1}^{n} \sum_{j=1}^{m_k} \log p(f_j^{(k)}|e^{(k)}) \; ,$$

where

$$p(f_j^{(k)}|e^{(k)}) = \sum_{i=0}^{l_k} t(f_j^{(k)}|e_i^{(k)}) d(i|j) \; .$$

Crucially, while the constraints in the IBM Model 2 optimization problem are linear, the objective function in Eq. 5 is non-convex. Therefore, optimization methods for IBM Model 2, in particular the EM algorithm, are typically only guaranteed to reach a local maximum of the objective function.

For completeness, Figure 2 shows the optimization problem for IBM Model 1. In IBM Model 1 the distortion parameters $d(i|j)$ are all fixed to be the uniform distribution (i.e., $1/(L + 1)$). The objective function for IBM Model 1 is actually convex, so the EM algorithm will converge to a global maximum. However IBM Model 1 is much weaker than model 2, and typically gives far worse performance.

Input: Define E, F, L, M, $(e^{(k)}, f^{(k)}, l_k, m_k)$ for $k = 1 \ldots n$, $D(e)$ for $e \in E$ as in Section 3.

Parameters:
- A parameter $t(f|e)$ for each $e \in E, f \in D(e)$.

Constraints:

$$\forall e \in E, f \in D(e), \qquad t(f|e) \geq 0 \tag{6}$$

$$\forall e \in E, \qquad \sum_{f \in D(e)} t(f|e) = 1 \tag{7}$$

Objective: Maximize

$$\frac{1}{n} \sum_{k=1}^{n} \sum_{j=1}^{m_k} \log \sum_{i=0}^{l_k} \frac{t(f_j^{(k)}|e_i^{(k)})}{(L + 1)} \tag{8}$$

with respect to the $t(f|e)$ parameters.

Figure 2: The IBM Model 1 Optimization Problem.

A common heuristic is to initialize the $t(f|e)$ parameters in EM optimization of IBM Model 2 using the output from IBM Model 1. The intuition behind this heuristic is that the IBM Model 1 values for $t(f|e)$ will be a reasonable starting point, and the EM algorithm will climb to a “good” local optimum. We are not aware of any guarantees for this initialization heuristic, however.

4 A Convex Relaxation of IBM Model 2

We now introduce a convex optimization problem, the I2CR (IBM 2 Convex Relaxation) problem. As its name suggests, this optimization problem is closely related to IBM Model 2, but is convex. Because of this it will be relatively easy to derive an optimization algorithm that is guaranteed to converge to a global optimum. Our experiments show that the relaxation gives very similar performance to the original IBM 2 optimization problem, as described in the previous section.

We first describe an optimization problem, I2CR-1, that illustrates the basic idea behind the convex relaxation. We then describe a refined relaxation, I2CR-2, that introduces a couple of modifications, and which performs well in experiments.

Input: Define E, F, L, M, $(e^{(k)}, f^{(k)}, l_k, m_k)$ for $k = 1 \ldots n$, $D(e)$ for $e \in E$ as in Section 3.

Parameters:

- A parameter $t(f|e)$ for each $e \in E, f \in D(e)$.
- A parameter $d(i|j)$ for each $i \in [L]_0, j \in [M]$.
- A parameter $q(i,j,k)$ for each $k \in [n]$, $i \in [l_k]_0$, $j \in [m_k]$.

Constraints:

$$\forall e \in E, f \in D(e), \quad t(f|e) \geq 0 \quad (9)$$

$$\forall e \in E, \quad \sum_{f \in D(e)} t(f|e) = 1 \quad (10)$$

$$\forall i \in [L]_0, j \in [M], \quad d(i|j) \geq 0 \quad (11)$$

$$\forall j \in [M], \quad \sum_{i \in [L]_0} d(i|j) = 1 \quad (12)$$

$$\forall i, j, k, \qquad q(i,j,k) \geq 0 \quad (13)$$

$$\forall i, j, k, \qquad q(i,j,k) \leq d(i|j) \quad (14)$$

$$\forall i, j, k, \qquad q(i,j,k) \leq t(f_j^{(k)}|e_i^{(k)}) \quad (15)$$

Objective: Maximize

$$\frac{1}{n} \sum_{k=1}^{n} \sum_{j=1}^{m_k} \log \sum_{i=0}^{l_k} q(i,j,k) \quad (16)$$

with respect to the $q(i,j,k)$, $t(f|e)$ and $d(i|j)$ parameters.

Figure 3: The I2CR-1 (IBM 2 Convex Relaxation) Problem, version 1.

4.1 The I2CR-1 Problem

The I2CR-1 problem is shown in Figure 3. A first key idea is to introduce a new variable $q(i,j,k)$ for each $k \in [n]$, $i \in [l_k]_0$, $j \in [m_k]$: that is, a new variable for each triple (i,j,k) specifying a sentence pair, and a specific English and French position in that sentence. Each q variable must satisfy the constraints in Eqs. 13-15, repeated here for convenience:

$$\begin{aligned} \forall i, j, k, &\qquad q(i,j,k) \geq 0 \, , \\ \forall i, j, k, &\qquad q(i,j,k) \leq d(i|j) \, , \\ \forall i, j, k, &\qquad q(i,j,k) \leq t(f_j^{(k)}|e_i^{(k)}) \, . \end{aligned}$$

The objective function is

$$\frac{1}{n} \sum_{k=1}^{n} \sum_{j=1}^{m_k} \log \sum_{i=0}^{l_k} q(i,j,k)$$

which is similar to the objective function in Figure 1, but where $t(f_j^{(k)}|e_i^{(k)}) \times d(i|j)$ has been replaced by $q(i,j,k)$. The intuition behind the new problem is as follows. If, instead of the constraints in Eqs. 13-15, we had the constraint

$$q(i,j,k) = t(f_j^{(k)}|e_i^{(k)}) \times d(i|j) \, , \qquad (17)$$

then the I2CR-1 problem would clearly be identical to the IBM Model 2 optimization problem. We have used a standard relaxation of the non-linear constraint $x = y \times z$ where x, y, z are all variables in the range $[0, 1]$, namely

$$\begin{aligned} x &\leq y \, , \\ x &\leq z \, , \\ x &\geq y + z - 1 \, . \end{aligned}$$

These inequalites are a relaxation in the sense that any (x, y, z) triple that satisfies $x = y \times z$ also satisfies these constraints. Applying this relaxation to Eq. 17 gives

$$\begin{aligned} q(i,j,k) &\leq t(f_j^{(k)}|e_i^{(k)}) \, , \\ q(i,j,k) &\leq d(i|j) \, , \\ q(i,j,k) &\geq t(f_j^{(k)}|e_i^{(k)}) + d(i|j) - 1 \, . \quad (18) \end{aligned}$$

The final thing to note is that the constraint in Eq. 18 can be omitted in the I2CR-1 problem. This is because the task is to maximize the objective with respect to the q variables and the objective is strictly increasing as the q values increase—thus lower bounds on their values are redundant in the I2CR-1 problem.

It is easily verified that the constraints in the I2CR-1 problem are linear, and that the objective function is convex. In Section 5 of this paper we describe an optimization method for the problem.

Note that because the objective function is being maximized, and the objective increases monotonically as the q values increase, at the global optimum[1]

[1] More precisely, at *any* global optimum: the objective function may not be strictly convex, in which case there will be multiple global optima.

Input: Same as in I2CR-1 (Figure 4). **Parameters**: Same as in I2CR-1 (Figure 4).

Constraints: Same as in I2CR-1 (Figure 4).
Objective: Maximize

$$\frac{1}{2n}\sum_{k=1}^{n}\sum_{j=1}^{m_k}\log'\sum_{i=0}^{l_k} q(i,j,k)$$

$$+ \quad \frac{1}{2n}\sum_{k=1}^{n}\sum_{j=1}^{m_k}\log'\sum_{i=0}^{l_k} \frac{t(f_j^{(k)}|e_i^{(k)})}{(L+1)}$$

with respect to the $q(i,j,k)$, $t(f|e)$ and $d(i|j)$ parameters.

Figure 4: The I2CR-2 (IBM 2 Convex Relaxation) Problem, version 2. The problem is identical to the I2CR-1 problem, but it also includes a term in the objective function that is identical to the IBM Model 1 objective. We define $\log'(z) = \log(z + \lambda)$ where λ is a small positive constant.

we have

$$q(i,j,k) = \min\{t(f_j^{(k)}|e_i^{(k)}), d(i|j)\} ,$$

where $\min\{x, y\}$ returns the minimum of the two values x and y. Thus, we could actually eliminate the q variables and write an optimization problem that is identical to the IBM Model 2 optimization problem, but with the objective function

$$\frac{1}{n}\sum_{k=1}^{n}\sum_{j=1}^{m_k}\log\sum_{i=0}^{l_k}\min\{t(f_j^{(k)}|e_i^{(k)}), d(i|j)\} .$$

It will turn out that both views of the I2CR-1 problem—with and without the q variables—are helpful, so we have included both in this paper.

4.2 The I2CR-2 Problem

Figure 4 shows the refined optimization problem, which we call I2CR-2. The problem incorporates two modifications. First, we modify the objective function to be

$$\frac{1}{2n}\sum_{k=1}^{n}\sum_{j=1}^{m_k}\log'\sum_{i=0}^{l_k} q(i,j,k)$$

$$+ \quad \frac{1}{2n}\sum_{k=1}^{n}\sum_{j=1}^{m_k}\log'\sum_{i=0}^{l_k} \frac{t(f_j^{(k)}|e_i^{(k)})}{(L+1)} .$$

Thus the objective function includes a second term that is identical to the objective function for IBM Model 1 (see Figure 2). In preliminary experiments with the I2CR-1 optimization problem, we found that the I2CR-1 objective was not sufficiently dependent on the t parameters: intuitively, if the d parameters achieve the $\min$ on many training examples, the values for the t variables become unimportant. The addition of the IBM Model 1 objective fixed this problem by introducing a term that depends on the t values alone.

Second, we replace $\log$ by $\log'$, where $\log'(z) = \log(z + \lambda)$, and λ is a small positive constant (in our experiments we used $\lambda = 0.001$). Under this definition the derivatives of $\log'$ are upper-bounded by $1/\lambda$, in contrast to $\log$, where the derivatives can diverge to infinity. The optimization methods we use are gradient-based methods (or more precisely, *subgradient*-based methods), and we have found them to be considerably more stable when the values for gradients do not diverge to infinity.

The modified objective remains convex.

5 A Stochastic Exponentiated-Gradient Algorithm for Optimization

We now describe an algorithm for optimizing the I2CR-2 problem in Figure 4. The algorithm is closely related to stochastic gradient ascent, but with two modifications:

- First, because the $t(f|e)$ and $d(i|j)$ parameters have simplex constraints (see Figure 1), we use *exponentiated gradient* (EG) updates. EG algorithms are gradient-based methods that maintain simplex constraints; see for example: (Kivinen and Warmuth, 1997; Beck and Teboulle, 2003; Collins et al., 2008).

- Second, the objective function in the I2CR-2 problem is convex, but is not differentiable (the gradient may not exist at all points). For this reason we use *subgradients* in the place of gradients. In spite of the non-differentiability of the objective function, subgradient methods still have strong convergence guarantees when combined with EG updates (e.g., the convergence proofs in (Beck and Teboulle, 2003) go through with minor modifications; see also (Bertsekas, 1999)).

To derive the updates, recall that we are maximizing the following objective function:

$$\begin{aligned} & h(t,d) \\ = \; & \frac{1}{2|\mathcal{T}|}\sum_{k\in\mathcal{T}}\sum_{j=1}^{m_k}\log'\sum_{i=0}^{l_k}\min\left\{t(f_j^{(k)}|e_i^{(k)}),d(i|j)\right\} \\ + \; & \frac{1}{2|\mathcal{T}|}\sum_{k\in\mathcal{T}}\sum_{j=1}^{m_k}\log'\sum_{i=0}^{l_k}\frac{t(f_j^{(k)}|e_i^{(k)})}{(L+1)} \; . \end{aligned} \quad (19)$$

Here we use $\mathcal{T}$ to denote the set $\{1\ldots n\}$; we will see shortly why this notation is convenient. We use t and d to refer to the full set of t and d parameters respectively; $h(t,d)$ is the function to be maximized. Recall that $\log'(z) = \log(z+\lambda)$ where λ is a small positive parameter.

Given a concave function $f(x)$ where $x \in \mathbb{R}^d$, a subgradient of $f(x)$ at x is any vector $g(x) \in \mathbb{R}^d$ such that for any $y \in \mathbb{R}^d$,

$$f(y) \le f(x) + g(x)\cdot(y-x) \; ,$$

where $u{\cdot}v$ is the inner product between vectors u and v. Subgradients are similar to gradients for differentiable concave functions, in that gradients satisfy the above property. Subgradients can be used in the place of gradients in many optimization algorithms (see for example (Bertsekas, 1999)).

The subgradients for the objective function in Eq. 19 take a simple form. First, define

$$\begin{aligned} R(j,k) &= \lambda + \sum_{i=0}^{l_k} t(f_j^{(k)}|e_i^{(k)}) \; , \\ Q(j,k) &= \lambda + \sum_{i=0}^{l_k} \min\{t(f_j^{(k)}|e_i^{(k)}), d(i|j)\} \; , \end{aligned}$$

and

$$I(i,j,k) = \begin{cases} 1 & \text{if } t(f_j^{(k)}|e_i^{(k)}) \le d(i|j) \\ 0 & \text{otherwise .} \end{cases}$$

Then the subgradients[2] are

$$\nabla t(f|e) = \frac{1}{2|\mathcal{T}|}\sum_{\substack{i,j,k: \\ f_j^{(k)}=f \\ e_i^{(k)}=e}}\left(\frac{1}{R(j,k)} + \frac{I(i,j,k)}{Q(j,k)}\right)$$

and

$$\nabla d(i|j) = \frac{1}{2|\mathcal{T}|}\sum_{k:i\le l_k, j\le m_k}\frac{1-I(i,j,k)}{Q(j,k)} \; .$$

Exponentiated-gradient updates then take the following form:

$$t(f|e) \leftarrow \frac{t(f|e)\times\exp\{\gamma\times\nabla t(f|e)\}}{\sum_f t(f|e)\times\exp\{\gamma\times\nabla t(f|e)\}} \quad (20)$$

and

$$d(i|j) \leftarrow \frac{d(i|j)\times\exp\{\gamma\times\nabla d(i|j)\}}{\sum_i d(i|j)\times\exp\{\gamma\times\nabla d(i|j)\}} \; , \quad (21)$$

where $\gamma > 0$ is a constant step size in the algorithm. Note that the EG updates make use of subgradients, but maintain the simplex constraints on the t and d variables.

The method just described is a *batch* gradient method, where the entire training set $\mathcal{T} = \{1\ldots n\}$ is used to derive the subgradients before the updates in Eqs. 20 and 21 are made. Many results in machine learning and NLP have shown that *stochastic gradient methods*, where a subset of the training examples is used before each gradient-based update, can converge much more quickly than batch gradient methods. In our notation, this simply involves replacing $\mathcal{T}$ by some subset $\mathcal{T}'$ of the training examples in the above definitions, where $|\mathcal{T}'|$ is typically much smaller than $|\mathcal{T}|$.

Figure 5 shows our final algorithm, a stochastic version of the exponentiated-gradient method. The method takes S passes over the data. For each pass, it randomly partitions the training set into mini-batches $\mathcal{T}_1\ldots\mathcal{T}_K$ of size B, where B is an integer specifying the size of each mini-batch (in our experiments we used $B = 125$ or $B = 250$). The algorithm then performs EG updates using each mini-batch $\mathcal{T}_1\ldots\mathcal{T}_K$ in turn. As can be seen in Table 3, our experiments show that the algorithm makes very significant progress in the first pass over the data, and takes very few iterations to converge to a good solution even though we initialized with uniform parameter values.

6 Experiments

In this section we describe experiments using the I2CR-2 optimization problem combined with the

[2] We set $\nabla t(f|e)$ and $\nabla d(i|j)$ as the subgradients for the objective function in Eq. 19 with respect to $t(f|e)$ and $d(i|j)$ respectively.

```
Input: Define $E$, $F$, $L$, $M$, $(e^{(k)}, f^{(k)}, l_k, m_k)$ for $k = 1 \dots n$, $D(e)$ for $e \in E$ as in Section 3. An integer $B$ specifying the batch size. An integer $S$ specifying the number of passes over the data. A step size $\gamma > 0$. A parameter $\lambda > 0$ used in the definition of $\log'$.
Parameters:
  • A parameter $t(f|e)$ for each $e \in E, f \in D(e)$.
  • A parameter $d(i|j)$ for each $i \in [L]_0, j \in [M]$.
Definitions:
  $R(j,k) = \lambda + \sum_{i=0}^{l_k} t(f_j^{(k)}|e_i^{(k)})$
  $Q(j,k) = \lambda + \sum_{i=0}^{l_k} \min\{t(f_j^{(k)}|e_i^{(k)}), d(i|j)\}$
Initialization:
  • $\forall e \in E,\ f \in D(e),\ t(f|e) = 1/|D(e)|$
  • $\forall j \in [M],\ i \in [L]_0,\ d(i|j) = 1/(L+1)$
Algorithm:
for all $s = 1$ to $S$ do
    Randomly partition $[n]$ into subsets $\mathcal{T}_1 \dots \mathcal{T}_K$ of size $B$ where $K = n/B$.
    for all $b = 1$ to $K$ do
        $\forall e \in E,\ f \in D(e),\ \alpha(e,f) = 0$
        $\forall j \in [M],\ i \in [L]_0,\ \beta(i,j) = 0$
        for all $k \in \mathcal{T}_b$ do
            for all $j = 1$ to $m_k$ do
                for all $i = 0$ to $l_k$ do
                    $\alpha(e_i^{(k)}, f_j^{(k)})$ += $1/(2R(j,k))$
                    if $t(f_j^{(k)}|e_i^{(k)}) \leq d(i|j)$ then
                        $\alpha(e_i^{(k)}, f_j^{(k)})$ += $1/(2Q(j,k))$
                    else
                        $\beta(i,j)$ += $1/(2Q(j,k))$
        $\forall e, f,\ t(f|e) = t(f|e)\exp(\gamma \times \alpha(e,f)/B)$
        $\forall i, j, d(i|j) = d(i|j)\exp(\gamma \times \beta(i,j)/B)$
        Renormalize $t$ and $d$ parameters to satisfy $\sum_f t(f|e) = 1$ and $\sum_i d(i|j) = 1$.
Output: $t$ and $d$ parameters.
```

Figure 5: The stochastic exponentiated-gradient algorithm for optimization of I2CR-2.

stochastic EG algorithm for parameter estimation. We first describe the data sets we use, and then describe experiments with the method, comparing our approach to results from IBM Model 2. We compare the various algorithms in terms of their accuracy in recovering alignments, using metrics such as F-measure and AER.

6.1 Data Sets

We use data from the bilingual word alignment workshop held at HLT-NAACL 2003 (Michalcea and Pederson, 2003). As a first dataset, we use the Canadian Hansards bilingual corpus, with 247,878 English-French sentence pairs as training data, 37 sentences of development data, and 447 sentences of test data (note that we use a randomly chosen subset of the original training set of 1.1 million sentences, similar to the setting used in (Moore, 2004)). The development and test data have been manually aligned at the word level, annotating alignments between source and target words in the corpus as either "sure" (S) or "possible" (P) alignments, as described in (Och and Ney, 2003).

As a second data set, we used the Romanian-English data from the HLT-NAACL 2003 workshop. This consisted of a training set of 48,706 Romanian-English sentence-pairs, a development set of 17 sentence pairs, and a test set of 248 sentence pairs.

6.2 Methodology

For each of the models—IBM Model 1, IBM Model 2, and I2CR-2—we follow convention in applying the following methodology: first, we estimate the t and d parameters using models in both source-target and target-source directions; second, we find the most likely alignment for each development or test data sentence in each direction; third, we take the intersection of the two alignments as the final output from the model.

For the EG algorithm we use a batch size $B = 250$ and step size $\gamma = 0.5$ on the Hansards data, and $B = 125$ and $\gamma = 0.5$ for the Romanian-English data.

We report the performance of the models in terms of *Precision*, *Recall*, *AER*, and *F-Measure* as defined by (Och and Ney, 2003). If A is the set of alignments produced by an algorithm, S is the set of sure alignments as annotated in test data, and P is the set of possible alignments, then these quantities are defined as

$$\text{Recall} = \frac{|A \cap S|}{|S|},$$

$$\text{Precision} = \frac{|A \cap S|}{|A|} ,$$

$$\text{AER} = 1 - \frac{|A \cap S| + |A \cap P|}{|A| + |S|} ,$$

$$\text{F-Measure} = \frac{1}{\frac{.5}{\text{Recall}} + \frac{.5}{\text{Precision}}} .$$

Note that we report results in both AER and F-measure; however there is evidence (Fraser and Marcu, 2004) that F-measure is better correlated with translation quality when the alignments are used in a full system.

In training IBM Model 1 we follow (Moore, 2004) in running EM for 15 iterations. In training IBM Model 2 we first train IBM Model 1 for 15 iterations to initialize the t parameters, then train IBM Model 2 for a further 10 iterations. For the EG algorithm, we use 10 iterations over the training data for the Hansards data, and 15 iterations on the Romanian-English data (on the latter dataset results on the trial data showed that the method took slightly longer to converge). We report F-measure and AER results for each of the iterations under the IBM Model 2 and I2CR-2 models. See Table 1 for the results on the Hansards data, and Table 2 for the results on the English-Romanian dataset. It can be seen that both I2CR-2 and IBM Model 2 converge to a fairly stable result after 2-3 iterations. The two models give very similar levels of performance, for example after 10 iterations on the Hansard data IBM Model 2 gives 14.22 AER and 0.7516 F-Measure versus 14.60 AER and 0.7506 F-Measure for I2CR-2.

On the right, Table 3 shows the values of the objective function at each iteration when using the EG algorithm to optimize the I2CR-2 objective. The method makes a large amount of progress on the first iteration and then continues to improve. Finally, we note that the memory requirements for I2CR-2 and IBM2 are about the same, but that the time for one iteration of I2CR-2 on the Hansards data is approximately one hour, while the time for one iteration of IBM2 was approximately 10 minutes.

7 Conclusions and Future Work

We have introduced the first convex model for unsupervised learning of alignments in statistical machine translation with performance comparable to

Iteration	IBM2 AER	I2CR-2 AER	IBM2 F-Measure	I2CR-2 F-Measure
		Test Set Statistics		
1	0.1491	0.1556	0.7530	0.7369
2	0.1477	0.1489	0.7519	0.7456
3	0.1451	0.1476	0.7527	0.7467
4	0.1426	0.1488	0.7536	0.7449
5	0.1422	0.1495	0.7535	0.7472
6	0.1431	0.1476	0.7511	0.7478
7	0.1434	0.1506	0.7506	0.7456
8	0.1437	0.1495	0.7501	0.7470
9	0.1434	0.1494	0.7501	0.7468
10	0.1422	0.1460	0.7516	0.7506
		Development Set Statistics		
1	0.1871	0.1971	0.6823	.6676
2	0.1896	0.1760	0.6758	.6827
3	0.1964	0.1860	0.6648	.6739
4	0.1912	0.1835	0.6713	.6775
5	0.1884	0.1813	0.6740	.06773
6	0.1836	0.1851	0.6767	0.6811
7	0.1831	0.1806	0.6749	0.6765
8	0.1842	0.1843	0.6739	0.6775
9	0.1864	0.1928	0.6694	0.6640
10	0.1845	0.1829	0.6703	.6721

Table 1: Results on the Hansards data for IBM Model 2 and the I2CR-2 method.

Iteration	IBM2 AER	I2CR-2 AER	IBM2 F-Measure	I2CR-2 F-Measure
		Test Set Statistics		
1	0.4041	0.5354	0.5959	0.4646
2	0.4010	0.4764	0.5990	0.5256
3	0.4020	0.4543	0.5980	0.5457
4	0.4012	0.4384	0.5988	0.5617
5	0.4003	0.4277	0.5997	0.5723
6	0.3990	0.4266	0.6010	0.5834
7	0.4000	0.4162	0.6000	0.5838
8	0.4023	0.4114	0.5977	0.5886
9	0.4022	0.4081	0.5978	0.5919
10	0.4027	0.4043	0.5973	0.5957
11	0.4031	0.4040	0.5969	0.5960
12	0.4042	0.4027	0.5958	0.5973
13	0.4043	0.4021	0.5957	0.5979
14	0.4062	0.4007	0.5938	0.5993
15	0.4057	0.4014	0.5943	0.5986
		Development Set Statistics		
1	0.4074	0.5841	0.5926	0.4159
2	0.3911	0.4938	0.6089	0.5062
3	0.3888	0.4673	0.6112	0.5327
4	0.3904	0.4596	0.6096	0.5404
5	0.3881	0.4463	0.6119	0.5537
6	0.3904	0.4306	0.6096	0.5694
7	0.3936	0.4175	0.6094	0.5826
8	0.3897	0.4060	0.6103	0.5940
9	0.3961	0.4014	0.6039	0.5986
10	0.3970	0.4072	0.6030	0.5928
11	0.4018	0.3956	0.5982	0.6044
12	0.4035	0.3931	0.5965	0.6069
13	0.4035	0.3862	0.5965	0.6138
14	0.4014	0.3908	0.5986	0.6092
15	0.4063	0.3858	0.5937	0.6142

Table 2: Results on the English-Romanian data for IBM Model 2 and the I2CR-2 method.

Iteration	EF Objective	FE Objective
0	-99.6053	-79.5566
1	-32.4528	-27.4925
2	-31.1641	-26.262
3	-30.6311	-25.7093
4	-30.3367	-25.3714
5	-30.1428	-25.1456
6	-30.0000	-24.992
7	-29.8736	-24.8605
8	-29.8093	-24.7551
9	-29.7326	-24.684
10	-29.6771	-24.6099

Table 3: Objective values for the EG algorithm optimization of I2CR-2 at each iteration. "EF Objective" corresponds to training a model with $t(e|f)$ parameters, "FE Objective" corresponds to the reverse direction, with $t(f|e)$ parameters. Iteration 0 corresponds to the objective value under the initial, uniform parameter values.

the commonly-used IBM Model 2. We believe that introducing convexity without sacrificing performance will open the door to further improvements in this area. Future work will consider ways to speed up our algorithm and extensions of the method to more complex alignment models.

Acknowledgments

Michael Collins is partly supported by NSF grant IIS-1161814. Cliff Stein is partly supported by NSF grant CCF-0915681. The authors thank Sasha Rush for his help with implementation questions. We also thank the anonymous reviewers for many useful comments; we hope to pursue the comments we were not able to address in a followup paper.

References

Peter L. Bartlett, Ben Taskar, Michael Collins and David Mcallester. 2004. Exponentiated Gradient Algorithms for Large-Margin Structured Classification. *In Proceedings of NIPS.*

Amir Beck and Marc Teboulle. 2003. Mirror Descent and Nonlinear Projected Subgradient Methods for Convex Optimization. *Operations Research Letters*, 31:167-175.

Dimitris Bertsimas and John N. Tsitsiklis. 1997. Introduction to Linear Programming. Athena Scientific.

Dimitris Bertsimas. 2005. Optimization Over Integers. Dynamic Ideas.

Dimitri P. Bertsekas. 1999. Nonlinear Optimization. Athena Press.

Steven Boyd and Lieven Vandenberghe. 2004. Convex Optimization. Cambridge University Press.

Peter F. Brown, Vincent J. Della Pietra, Stephen A. Della Pietra, and Robert. L. Mercer. 1993. The Mathematics of Statistical Machine Translation: Parameter Estimation. *Computational Linguistics*, 19:263-311.

David Chiang. 2005. A Hierarchical Phrase-Based Model for Statistical Machine Translation. *In Proceedings of the ACL.*

Michael Collins, Amir Globerson, Terry Koo, Xavier Carreras and Peter L. Bartlett. 2008. Exponentiated Gradient Algorithms for Conditional Random Fields and Max-Margin Markov Networks. *Journal Machine Learning,* 9(Aug): 1775-1822.

A. P. Dempster, N. M. Laird, and D. B. Rubin. 1977. Maximum Likelihood From Incomplete Data via the EM Algorithm. *Journal of the royal statistical society, series B*, 39(1):1-38.

Alexander Fraser and Daniel Marcu. 2007. Measuring Word Alignment Quality for Statistical Machine Translation. *Journal Computational Linguistics*, 33(3): 293-303.

Kuzman Ganchev, Joao V. Graca, Jennifer Gillenwater, Ben Taskar. 2010. Posterior Regularization for Structured Latent Variable Models. *Journal of Machine Learning*, 11(July): 2001-2049.

Joao V. Graca, Kuzman Ganchev and Ben Taskar. 2007. Expectation Maximization and Posterior Constraints. *In Proceedings of NIPS.*

Aria Haghighi, John Blitzer, John DeNero and Dan Klein. 2009. Better Word Alignments with Supervised ITG Models. *In Proceedings of the ACL.*

Darcey Riley and Daniel Gildea. 2012. Improving the IBM Alignment Models Using Variational Bayes. *In Proceedings of the ACL.*

Yuhong Guo and Dale Schuurmans. 2007. Convex Relaxations of Latent Variable Training. *In NIPS.*

Simon Lacoste-Julien, Ben Taskar, Dan Klein, and Michael Jordan. 2008. Word Alignment via Quadratic Assignment. *In Proceedings of the HLT-NAACL.*

Phillip Koehn. 2008. Statistical Machine Translation. Cambridge University Press.

Kivinen, J., Warmuth, M. 1997. Exponentiated Gradient Versus Gradient Descent for Linear Predictors. *Information and Computation*, 132, 1-63.

Percy Liang, Ben Taskar and Dan Klein. 2006. Alignment by Agreement. *In Proceedings of NAACL.*

Daniel Marcu, Wei Wang, Abdessamad Echihabi, and Kevin Knight. 2006. SPMT: Statistical Machine Translation with Syntactified Target Language Phrases. *In Proceedings of the EMNLP.*

Andre F. T. Martins, Noah A. Smith and Eric P. Xing. 2010. Turbo Parsers: Dependency Parsing by Approximate Variational Inference. *In Proceedings of the EMNLP*.

Rada Michalcea and Ted Pederson. 2003. An Evaluation Exercise in Word Alignment. *HLT-NAACL 2003: Workshop in building and using Parallel Texts: Data Driven Machine Translation and Beyond*.

Robert C. Moore. 2004. Improving IBM Word-Alignment Model 1. *In Proceedings of the ACL.*

Stephan Vogel, Hermann Ney and Christoph Tillman. 1996. HMM-Based Word Alignment in Statistical Translation. *In Proceedings of COLING*.

Franz Och and Hermann Ney. 2003. A Systematic Comparison of Various Statistical Alignment Models. *Computational-Linguistics*, 29(1): 19-52.

Libin Shen, Jinxi Xu and Ralph Weischedel. 2008. A New String-to-Dependency Machine Translation Algorithm with a Target Dependency Language Model. *In Proceedings of the ACL-HLT*.

Ben Taskar, Simon Lacoste-Julien and Dan Klein. 2005. A Discriminative Matching Approach to Word Alignment. *In Proceedings of the EMNLP*.

Kristina Toutanova and Michel Galley. 2011. Why Initialization Matters for IBM Model 1: Multiple Optima and Non-Strict Convexity. *In Proceedings of the ACL.*

Kenji Yamada and Kevin Knight. 2001. A Syntax-Based Statistical Translation Model. *In Proceedings of the ACL.*

Kenji Yamada and Kevin Knight. 2002. A Decoder for Syntax-Based Statistical Machine Translation. *In Proceedings of the ACL.*

Ashish Vaswani, Liang Huang and David Chiang. 2012. Smaller Alignment Models for Better Translations: Unsupervised Word Alignment with the $L0$-norm. *In Proceedings of the ACL.*