Appeared in *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL 2004)*, Barcelona, July 2004.

Annealing Techniques for Unsupervised Statistical Language Learning

Noah A. Smith and **Jason Eisner**
Department of Computer Science / Center for Language and Speech Processing
Johns Hopkins University, Baltimore, MD 21218 USA
{nasmith,jason}@cs.jhu.edu

Abstract

Exploiting unannotated natural language data is hard largely because unsupervised parameter estimation is hard. We describe deterministic annealing (Rose et al., 1990) as an appealing alternative to the Expectation-Maximization algorithm (Dempster et al., 1977). Seeking to avoid search error, DA begins by globally maximizing an easy concave function and maintains a local maximum as it gradually morphs the function into the desired non-concave likelihood function. Applying DA to parsing and tagging models is shown to be straightforward; significant improvements over EM are shown on a part-of-speech tagging task. We describe a variant, skewed DA, which can incorporate a good initializer when it is available, and show significant improvements over EM on a grammar induction task.

1 Introduction

Unlabeled data remains a tantalizing potential resource for NLP researchers. Some tasks can thrive on a nearly pure diet of unlabeled data (Yarowsky, 1995; Collins and Singer, 1999; Cucerzan and Yarowsky, 2003). But for other tasks, such as machine translation (Brown et al., 1990), the chief merit of unlabeled data is simply that nothing else is available; unsupervised parameter estimation is notorious for achieving mediocre results.

The standard starting point is the Expectation-Maximization (EM) algorithm (Dempster et al., 1977). EM iteratively adjusts a model's parameters from an initial guess until it converges to a local maximum. Unfortunately, likelihood functions in practice are riddled with suboptimal local maxima (e.g., Charniak, 1993, ch. 7). Moreover, maximizing likelihood is not equivalent to maximizing task-defined accuracy (e.g., Merialdo, 1994).

Here we focus on the search error problem. Assume that one has a model for which improving likelihood really will improve accuracy (e.g., at predicting hidden part-of-speech (POS) tags or parse trees). Hence, we seek methods that tend to locate mountaintops rather than hilltops of the likelihood function. Alternatively, we might want methods that find hilltops with other desirable properties.[1]

In §2 we review deterministic annealing (DA) and show how it generalizes the EM algorithm. §3 shows how DA can be used for parameter estimation for models of language structure that use dynamic programming to compute posteriors over hidden structure, such as hidden Markov models (HMMs) and stochastic context-free grammars (SCFGs). In §4 we apply DA to the problem of learning a trigram POS tagger without labeled data. We then describe how one of the received strengths of DA—its robustness to the initializing model parameters—can be a shortcoming in situations where the initial parameters carry a helpful bias. We present a solution to this problem in the form of a new algorithm, *skewed* deterministic annealing (SDA; §5). Finally we apply SDA to a grammar induction model and demonstrate significantly improved performance over EM (§6). §7 highlights future directions for this work.

2 Deterministic annealing

Suppose our data consist of a pairs of random variables X and Y, where the value of X is observed and Y is hidden. For example, X might range over sentences in English and Y over POS tag sequences. We use $\mathcal{X}$ and $\mathcal{Y}$ to denote the sets of possible values of X and Y, respectively. We seek to build a model that assigns probabilities to each $(x, y) \in \mathcal{X} \times \mathcal{Y}$. Let $\vec{x} = \{x_1, x_2, ..., x_n\}$ be a corpus of unlabeled examples. Assume the class of models is fixed (for example, we might consider only first-order HMMs with s states, corresponding notionally to POS tags). Then the task is to find good parameters $\vec{\theta} \in \mathbb{R}^N$ for the model. The criterion most commonly used in building such models from unlabeled data is *maximum likelihood* (ML); we seek the parameters $\vec{\theta}^*$:

$$\operatorname*{argmax}_{\vec{\theta}} \Pr(\vec{x} \mid \vec{\theta}) = \operatorname*{argmax}_{\vec{\theta}} \prod_{i=1}^{n} \sum_{y \in \mathcal{Y}} \Pr(x_i, y \mid \vec{\theta}) \quad (1)$$

[1] Wang et al. (2003) suggest that one should seek a high-entropy hilltop. They argue that to account for partially-observed (unlabeled) data, one should choose the distribution with the highest Shannon entropy, subject to certain data-driven constraints. They show that this desirable distribution is one of the local maxima of likelihood. Whether high-entropy local maxima really predict test data better is an empirical question.

Input: $\vec{x}$, $\vec{\theta}_{(0)}$ **Output:** $\vec{\theta}^*$

$i \leftarrow 0$

do:

(E) $\tilde{p}(\vec{y}) \leftarrow \frac{\Pr(\vec{x},\vec{y}|\vec{\theta}_{(i)})}{\sum_{\vec{y}' \in \mathcal{Y}^n} \Pr(\vec{x},\vec{y}'|\vec{\theta}_{(i)})}, \forall \vec{y}$

(M) $\vec{\theta}_{(i+1)} \leftarrow \text{argmax}_{\vec{\theta}} \mathbf{E}_{\tilde{p}(\vec{Y})} \left[\log \Pr(\vec{x}, \vec{Y} \mid \vec{\theta})\right]$

$i \leftarrow i + 1$

until $\vec{\theta}_{(i)} \approx \vec{\theta}_{(i-1)}$

$\vec{\theta}^* \leftarrow \vec{\theta}_{(i)}$

Fig. 1: The EM algorithm.

Each parameter θ_j corresponds to the conditional probability of a single *model event*, e.g., a state transition in an HMM or a rewrite in a PCFG. Many NLP models make it easy to maximize the likelihood of *supervised* training data: simply count the model events in the observed (x_i, y_i) pairs, and set the conditional probabilities θ_i to be proportional to the counts. In our unsupervised setting, the y_i are unknown, but solving (1) is almost as easy provided that we can obtain the *posterior distribution* of Y given each x_i (that is, $\Pr(y \mid x_i)$ for each $y \in \mathcal{Y}$ and each x_i). The only difference is that we must now count the model events fractionally, using the *expected* number of occurrences of each (x_i, y) pair.

This intuition leads to the EM algorithm in Fig. 1. It is guaranteed that $\Pr(\vec{x} \mid \vec{\theta}_{(i+1)}) \geq \Pr(\vec{x} \mid \vec{\theta}_{(i)})$.

For language-structure models like HMMs and SCFGs, efficient dynamic programming algorithms (forward-backward, inside-outside) are available to compute the distribution $\tilde{p}$ at the E step of Fig. 1 and use it at the M step. These algorithms run in polynomial time and space by structure-sharing the possible y (tag sequences or parse trees) for each x_i, of which there may be exponentially many in the length of x_i. Even so, the majority of time spent by EM for such models is on the E steps. In this paper, we can fairly compare the runtime of EM and other training procedures by counting the number of E steps they take on a given training set and model.

2.1 Generalizing EM

Figure 2 shows the *deterministic annealing* (DA) algorithm derived from the framework of Rose et al. (1990). It is quite similar to EM.[2] However, DA adds an outer loop that iteratively increases a value β, and computation of the posterior in the E step is modified to involve this β.

[2]Other expositions of DA abound; we have couched ours in data-modeling language. Readers interested in the Lagrangian-based derivations and analogies to statistical physics (including phase transitions and the role of β as the inverse of temperature in free-energy minimization) are referred to Rose (1998) for a thorough discussion.

Input: $\vec{x}$, $\vec{\theta}_{(0)}$, $\beta_{\max} > \beta_{\min} > 0$, $\alpha > 1$ **Output:** $\vec{\theta}^*$

$i \leftarrow 0$; $\beta \leftarrow \beta_{\min}$

while $\beta \leq \beta_{\max}$**:**

do:

(E) $\tilde{p}(\vec{y}) \leftarrow \frac{\Pr(\vec{x},\vec{y}|\vec{\theta}_{(i)})^{\beta}}{\sum_{\vec{y}' \in \mathcal{Y}^n} \Pr(\vec{x},\vec{y}'|\vec{\theta}_{(i)})^{\beta}}, \forall \vec{y}$

(M) $\vec{\theta}_{(i+1)} \leftarrow \text{argmax}_{\vec{\theta}} \mathbf{E}_{\tilde{p}(\vec{Y})} \left[\log \Pr(\vec{x}, \vec{Y} \mid \vec{\theta})\right]$

$i \leftarrow i + 1$

until $\vec{\theta}_{(i)} \approx \vec{\theta}_{(i-1)}$

$\beta \leftarrow \alpha \cdot \beta$

end while

$\vec{\theta}^* \leftarrow \vec{\theta}_{(i)}$

Fig. 2: The DA algorithm: a generalization of EM.

When $\beta = 1$, DA's inner loop will behave exactly like EM, computing $\tilde{p}$ at the E step by the same formula that EM uses. When $\beta \approx 0$, $\tilde{p}$ will be close to a uniform distribution over the hidden variable $\vec{y}$, since each numerator $\Pr(\vec{x}, \vec{y} \mid \vec{\theta})^{\beta} \approx 1$. At such β-values, DA effectively *ignores* the current parameters θ when choosing the posterior $\tilde{p}$ and the new parameters. Finally, as $\beta \to +\infty$, $\tilde{p}$ tends to place nearly all of the probability mass on the single most likely $\vec{y}$. This winner-take-all situation is equivalent to the "Viterbi" variant of the EM algorithm.

2.2 Gradated difficulty

In both the EM and DA algorithms, the E step selects a posterior $\tilde{p}$ over the hidden variable $\vec{Y}$ and the M step selects parameters $\vec{\theta}$. Neal and Hinton (1998) show how the EM algorithm can be viewed as optimizing a single objective function over both $\vec{\theta}$ and $\tilde{p}$. DA can also be seen this way; DA's objective function at a given β is

$$\mathcal{F}\left(\vec{\theta}, \tilde{p}, \beta\right) = \frac{1}{\beta} H(\tilde{p}) + \mathbf{E}_{\tilde{p}(\vec{Y})} \left[\log \Pr(\vec{x}, \vec{Y} \mid \vec{\theta})\right] \quad (2)$$

The EM version simply sets $\beta = 1$. A complete derivation is not difficult but is too lengthy to give here; it is a straightforward extension of that given by Neal and Hinton for EM.

It is clear that the value of β allows us to manipulate the relative importance of the two terms when maximizing $\mathcal{F}$. When β is close to 0, only the H term matters. The H term is the Shannon entropy of the posterior distribution $\tilde{p}$, which is known to be concave in $\tilde{p}$. Maximizing it is simple: set all x to be equiprobable (the uniform distribution). Therefore a sufficiently small β drives up the importance of H relative to the other term, and the entire problem becomes concave with a single global maximum to which we expect to converge.

In gradually increasing β from near 0 to 1, we start out by solving an easy concave maximization problem and use the result to initialize the next max-

imization problem, which is slightly more difficult (i.e., less concave). This continues, with the solution to each problem in the series being used to initialize the subsequent problem. When β reaches 1, DA behaves just like EM. Since the objective function is continuous in β where $\beta > 0$, we can visualize DA as gradually morphing the easy concave objective function into the one we really care about (likelihood); we hope to "ride the maximum" as β moves toward 1.

DA guarantees iterative improvement of the objective function (see Ueda and Nakano (1998) for proofs). But it does not guarantee convergence to a global maximum, or even to a better local maximum than EM will find, even with extremely slow β-raising. A new mountain on the surface of the objective function could arise at any stage that is preferable to the one that we will ultimately find.

To run DA, we must choose a few control parameters. In this paper we set $\beta_{\max} = 1$ so that DA will approach EM and finish at a local maximum of likelihood. $\beta_{\min}$ and the β-increase factor α can be set high for speed, but at a risk of introducing local maxima too quickly for DA to work as intended. (Note that a "fast" schedule that tries only a few β values is not as fast as one might expect, since it will generally take longer to converge at each β value.)

To conclude the theoretical discussion of DA, we review its desirable properties. DA is robust to initial parameters, since when β is close to 0 the objective hardly depends on $\vec{\theta}$. DA gradually increases the difficulty of search, which may lead to the avoidance of some local optima. By modifying the annealing schedule, we can change the runtime of the DA algorithm. DA is almost exactly like EM in implementation, requiring only a slight modification to the E step (see §3) and an additional outer loop.

2.3 Prior work

DA was originally described as an algorithm for clustering data in $\mathbb{R}^N$ (Rose et al., 1990). Its predecessor, *simulated annealing*, modifies the objective function during search by applying random perturbations of gradually decreasing size (Kirkpatrick et al., 1983). *Deterministic* annealing moves the randomness "inside" the objective function by taking expectations. DA has since been applied to many problems (Rose, 1998); we describe two key applications in language and speech processing.

Pereira, Tishby, and Lee (1993) used DA for soft hierarchical clustering of English nouns, based on the verbs that select them as direct objects. In their case, when β is close to 0, each noun is fuzzily placed in each cluster so that $\Pr(cluster \mid noun)$ is nearly uniform. On the M step, this results in clusters that are almost exactly identical; there is one *effective cluster*. As β is increased, it becomes increasingly attractive for the cluster centroids to move apart, or "split" into two groups (two effective clusters), and eventually they do so. Continuing to increase β yields a hierarchical clustering through repeated splits. Pereira et al. describe the tradeoff given through β as a control on the locality of influence of each noun on the cluster centroids, so that as β is raised, each noun exerts less influence on more distant centroids and more on the nearest centroids.

DA has also been applied in speech recognition. Rao and Rose (2001) used DA for supervised discriminative training of HMMs. Their goal was to optimize not likelihood but classification error rate, a difficult objective function that is piecewise-constant (hence not differentiable everywhere) and riddled with shallow local minima. Rao and Rose applied DA,[3] moving from training a nearly uniform classifier with a concave cost surface ($\beta \approx 0$) toward the desired deterministic classifier ($\beta \rightarrow +\infty$). They reported substantial gains in spoken letter recognition accuracy over both a ML-trained classifier and a localized error-rate optimizer.

Brown et al. (1990) gradually increased learning difficulty using a series of increasingly complex models for machine translation. Their training algorithm began by running an EM approximation on the simplest model, then used the result to initialize the next, more complex model (which had greater predictive power and many more parameters), and so on. Whereas DA provides gradated difficulty in parameter search, their learning method involves gradated difficulty among classes of models. The two are orthogonal and could be used together.

3 DA with dynamic programming

We turn now to the practical use of deterministic annealing in NLP. Readers familiar with the EM algorithm will note that, for typical stochastic models of language structure (e.g., HMMs and SCFGs), the bulk of the computational effort is required by the E step, which is accomplished by a two-pass dynamic programming (DP) algorithm (like the forward-backward algorithm). The M step for these models normalizes the posterior expected counts from the E step to get probabilities.[4]

[3] With an M step modified for their objective function: it improved expected accuracy under $\tilde{p}$, not expected log-likelihood.

[4] That is, assuming the usual generative parameterization of such models; if we generalize to Markov random fields (also known as log-linear or maximum entropy models) the M step, while still concave, might entail an auxiliary optimization routine such as iterative scaling or a gradient-based method.

Running DA for such models is quite simple and requires no modifications to the usual DP algorithms. The only change to make is in the values of the parameters passed to the DP algorithm: simply replace each θ_j by θ_j^β. For a given x, the forward pass of the DP computes (in a dense representation) $\Pr(y \mid x, \vec{\theta})$ for all y. Each $\Pr(y \mid x, \vec{\theta})$ is a product of some of the θ_j (each θ_j is multiplied in once for each time its corresponding model event is present in (x, y)). Raising the θ_j to a power will also raise their product to that power, so the forward pass will compute $\Pr(y \mid x, \vec{\theta})^\beta$ when given $\vec{\theta}^\beta$ as parameter values. The backward pass normalizes to the sum; in this case it is the sum of the $\Pr(y \mid x, \vec{\theta})^\beta$, and we have the E step described in Figure 2. We therefore expect an EM iteration of DA to take the same amount of time as a normal EM iteration.[5]

4 Part-of-speech tagging

We turn now to the task of inducing a trigram POS tagging model (second-order HMM) from an unlabeled corpus. This experiment is inspired by the experiments in Merialdo (1994). As in that work, complete knowledge of the tagging dictionary is assumed. The task is to find the trigram transition probabilities $\Pr(tag_i \mid tag_{i-1}, tag_{i-2})$ and emission probabilities $\Pr(word_i \mid tag_i)$. Merialdo's key result:[6] If some labeled data were used to initialize the parameters (by taking the ML estimate), then it was *not* helpful to improve the model's likelihood through EM iterations, because this almost always hurt the accuracy of the model's Viterbi tagging on a held-out test set. If only a small amount of labeled data was used (200 sentences), then some accuracy improvement was possible using EM, but only for a few iterations. When no labeled data were used, EM was able to improve the accuracy of the tagger, and this improvement continued in the long term.

Our replication of Merialdo's experiment used the Wall Street Journal portion of the Penn Treebank corpus, reserving a randomly selected 2,000 sentences (48,526 words) for testing. The remaining 47,208 sentences (1,125,240 words) were used in training, without any tags. The tagging dictionary was constructed using the entire corpus (as done by Merialdo). To initialize, the conditional transition and emission distributions in the HMM were set to uniform with slight perturbation. Every distribution was smoothed using add-0.1 smoothing (at every M step). The criterion for convergence is that the relative increase in the objective function between two iterations fall below 10^{-9}.

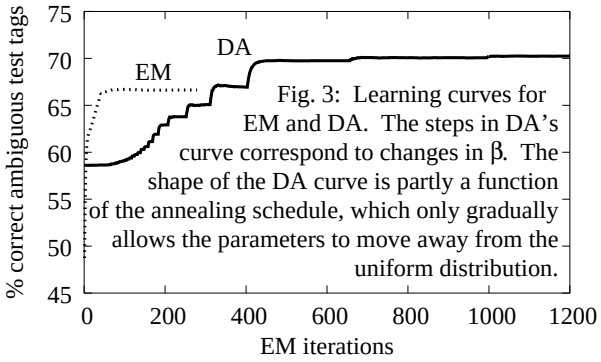

Fig. 3: Learning curves for EM and DA. The steps in DA's curve correspond to changes in β. The shape of the DA curve is partly a function of the annealing schedule, which only gradually allows the parameters to move away from the uniform distribution.

4.1 Experiment

In the DA condition, we set $\beta_{\min} = 0.0001$, $\beta_{\max} = 1$, and $\alpha = 1.2$. Results for the completely unsupervised condition (no labeled data) are shown in Figure 3 and Table 1. Accuracy was nearly monotonic: the final model is approximately the most accurate.

DA happily obtained a 10% reduction in tag error rate on training data, and an 11% reduction on test data. On the other hand, it did not manage to improve likelihood over EM. So was the accuracy gain mere luck? Perhaps not. DA may be more resistant to overfitting, because it may favor models whose posteriors $\tilde{p}$ have high entropy. At least in this experiment, its initial bias toward such models carried over to the final learned model.[7]

In other words, the higher-entropy local maximum found by DA, in this case, explained the observed data almost as well without overcommitting to particular tag sequences. The maximum entropy and latent maximum entropy principles (Wang et al., 2003, discussed in footnote 1) are best justified as ways to avoid overfitting.

For a supervised tagger, the maximum entropy principle prefers a conditional model $\Pr(\vec{y} \mid \vec{x})$ that is maximally unsure about what tag sequence $\vec{y}$ to apply to the training word sequence $\vec{x}$ (but expects the same feature counts as the true $\vec{y}$). Such a model is hoped to generalize better to unsupervised data.

We can make the same argument. The overfitting is not evident from Table 1, however, because in our setting the training/test split does *not* correspond to supervised vs. unsupervised data. Our supervised data are, roughly, the fragments of the training corpus that are unambiguously tagged thanks to the tag dictionary.[8] The EM model may overfit some

[5] With one caveat: less pruning may be appropriate because probability mass is spread more uniformly over different reconstructions of the hidden data. This paper uses no pruning.

[6] Similar results were found by Elworthy (1994).

[7] We computed the entropy over possible tags for each word in the test corpus, given the sentence the word occurs in. On average, the DA model had 0.082 bits per tag, while EM had only 0.057 bits per tag, a statistically significant difference ($p < 10^{-6}$) under a binomial sign test on word tokens.

[8] Without the tag dictionary, our learners would treat the tag

	E steps	final training cross-entropy (bits/word)	final test cross-entropy (bits/word)	% correct training tags (all)	(ambiguous)	% correct test tags (all)	(ambiguous)
EM	279	**9.136**	**9.321**	82.04	66.61	82.08	66.63
DA	1200	9.138	9.325	**83.85**	**70.02**	**84.00**	**70.25**

Table 1: EM vs. DA on unsupervised trigram POS tagging, using a tag dictionary. Each of the accuracy results is significant when accuracy is compared at either the word-level or sentence-level. (Significance at $p < 10^{-6}$ under a binomial sign test in each case. E.g., on the test set, the DA model correctly tagged 1,652 words that EM's model missed while EM correctly tagged 726 words that DA missed. Similarly, the DA model had higher accuracy on 850 sentences, while EM had higher accuracy on only 287. These differences are extremely unlikely to occur due to chance.) The differences in cross-entropy, compared by sentence, were significant in the training set but not the test set ($p < 0.01$ under a binomial sign test). Recall that lower cross entropy means higher likelihood.

parameters to these fragments. The higher-entropy DA model may be less likely to overfit, allowing it to do better on the unsupervised data—i.e., the parts of the training *and* test corpora with uncertain tags.

In summary, DA has settled on a local maximum of the likelihood function that (unsurprisingly) corresponds well with the entropy criterion, and perhaps as a result, does better on accuracy.

4.2 Significance

Seeking to determine how well this result generalized, we randomly split the corpus into ten equally-sized, nonoverlapping parts. EM and DA were run on each portion;[9] the results were inconclusive. DA achieved better test accuracy than EM on three of ten trials, better training likelihood on five trials, and better test likelihood on all ten trials.[10] Certainly decreasing the amount of data by an order of magnitude results in increased variance of the performance of any algorithm—so ten small corpora were not enough to determine whether to expect an improvement from DA more often than not.

4.3 Mixing labeled and unlabeled data (I)

In the other conditions described by Merialdo, varying amounts of labeled data (ranging from 100 sentences to nearly half of the corpus) were used to initialize the parameters $\vec{\theta}$, which were then trained using EM on the remaining unlabeled data. Only in the case where 100 labeled examples were used, and only for a few iterations, did EM improve the accuracy of this model. We replicated these experiments and compared EM with DA; DA damaged the models even more than EM. This is unsurprising; as noted before, DA effectively ignores the initial parameters $\vec{\theta}_{(0)}$. Therefore, even if initializing with a model trained on small amounts of labeled data had helped EM, DA would have missed out on this benefit. In the next section we address this issue.

5 Skewed deterministic annealing

The EM algorithm is quite sensitive to the initial parameters $\vec{\theta}_{(0)}$. We touted DA's insensitivity to those parameters as an advantage, but in scenarios where well-chosen initial parameters can be provided (as in §4.3), we wish for DA to be able exploit them.

In particular, there are at least two cases where "good" initializers might be known. One is the case explored by Merialdo, where some labeled data were available to build an initial model. The other is a situation where a good distribution is known over the labels y; we will see an example of this in §6.

We wish to find a way to incorporate an initializer into DA and still reap the benefit of gradated difficulty. To see how this will come about, consider again the E step for DA, which for all y:

$$\tilde{p}(y) \leftarrow \frac{\Pr(x, y \mid \vec{\theta})^{\beta}}{Z'(\vec{\theta}, \beta)} = \frac{\Pr(x, y \mid \vec{\theta})^{\beta} u(y)^{1-\beta}}{Z(\vec{\theta}, \beta)}$$

where u is the uniform distribution over $\mathcal{Y}$ and $Z'(\vec{\theta}, \beta)$ and $Z(\vec{\theta}, \beta) = Z'(\vec{\theta}, \beta) \cdot u(y)^{1-\beta}$ are normalizing terms. (Note that $Z(\vec{\theta}, \beta)$ does not depend on y because $u(y)$ is constant with respect to y.) Of course, when β is close to 0, DA chooses the uniform posterior because it has the highest entropy.

Seen this way, DA is *interpolating* in the log domain between two posteriors: the one given by y and $\vec{\theta}$ and the uniform one u; the interpolation coefficient is β. To generalize DA, we will replace the uniform u with another posterior, the "skew" posterior $\acute{p}$, which is an input to the algorithm. This posterior might be specified directly, as it will be in §6, or it might be computed using an M step from some good initial $\vec{\theta}_{(0)}$.

names as interchangeable and could not reasonably be evaluated on gold-standard accuracy.

[9] The smoothing parameters were scaled down so as to be proportional to the corpus size.

[10] It is also worth noting that runtimes were longer with the 10%-sized corpora than the full corpus (EM took 1.5 times as many E steps; DA, 1.3 times). Perhaps the algorithms traveled farther to find a local maximum. We know of no study of the effect of unlabeled training set size on the likelihood surface, but suggest two issues for future exploration. Larger datasets contain more idiosyncrasies but provide a stronger overall signal. Hence, we might expect them to yield a bumpier likelihood surface whose local maxima are more numerous but also differ more noticeably in height. Both these tendencies of larger datasets would in theory increase DA's advantage over EM.

The *skewed* DA (SDA) E step is given by:

$$\tilde{p}(y) \leftarrow \frac{1}{Z(\beta)} \Pr(x, y \mid \theta)^{\beta} \acute{p}(y)^{1-\beta} \quad (3)$$

When β is close to 0, the E step will choose $\tilde{p}$ to be very close to $\acute{p}$. With small β, SDA is a "cautious" EM variant that is wary of moving too far from the initializing posterior $\acute{p}$ (or, equivalently, the initial parameters $\vec{\theta}_{(0)}$). As β approaches 1, the effect of $\acute{p}$ will diminish, and when $\beta = 1$, the algorithm becomes identical to EM. The overall objective (matching (2) except for the boxed term) is:

$$\mathcal{F}'\left(\vec{\theta}, \tilde{p}, \beta\right) = \frac{1}{\beta} H(\tilde{p}) + \mathbf{E}_{\tilde{p}(\vec{Y})}\left[\log \Pr\left(\vec{x}, \vec{Y} \mid \vec{\theta}\right)\right] + \boxed{\frac{1-\beta}{\beta} \mathbf{E}_{\tilde{p}(\vec{Y})}\left[\log \acute{p}\left(\vec{Y}\right)\right]} \quad (4)$$

Mixing labeled and unlabeled data (II) Returning to Merialdo's mixed conditions (§4.3), we found that SDA repaired the damage done by DA but did not offer any benefit over EM. Its behavior in the 100-labeled sentence condition was similar to that of EM's, with a slightly but not significantly higher peak in training set accuracy. In the other conditions, SDA behaved like EM, with steady degradation of accuracy as training proceeded. It ultimately damaged performance only as much as EM did or did slightly better than EM (but still hurt).

This is unsurprising: Merialdo's result demonstrated that ML and maximizing accuracy are generally not the same; the EM algorithm consistently degraded the accuracy of his supervised models. SDA is simply another search algorithm with the same criterion as EM. SDA *did* do what it was expected to do—it used the initializer, repairing DA damage.

6 Grammar induction

We turn next to the problem of statistical grammar induction: inducing parse trees over unlabeled text. An excellent recent result is by Klein and Manning (2002). The constituent-context model (CCM) they present is a generative, deficient channel model of POS tag strings given binary tree bracketings. We first review the model and describe a small modification that reduces the deficiency, then compare both models under EM and DA.

6.1 Constituent-context model

Let (x, y) be a (tag sequence, binary tree) pair. x_i^j denotes the subsequence of x from the ith to the jth word. Let $y_{i,j}$ be 1 if the yield from i to j is a constituent in the tree y and 0 if it is not. The CCM gives to a pair (x, y) the following probability:

$$\Pr(x, y) = \Pr(y) \cdot \prod_{1 \le i \le j \le |x|} \left[\psi\left(x_i^j \middle| y_{i,j}\right) \cdot \chi\left(x_{i-1}, x_{j+1} \middle| y_{i,j}\right) \right]$$

where ψ is a conditional distribution over possible tag-sequence yields (given whether the yield is a constituent or not) and χ is a conditional distribution over possible contexts of one tag on either side of the yield (given whether the yield is a constituent or not). There are therefore four distributions to be estimated; $\Pr(y)$ is taken to be uniform.

The model is initialized using expected counts of the constituent and context features given that all the trees are generated according to a random-split model.[11]

The CCM generates each tag not once but $O(n^2)$ times, once by every constituent or non-constituent span that dominates it. We suggest the following modification to alleviate some of the deficiency:

$$\Pr(x, y) = \Pr(y) \cdot \prod_{1 \le i \le j \le |x|} \left[\psi\left(x_i^j \middle| y_{i,j}, \boxed{j - i + 1}\right) \cdot \chi\left(x_{i-1}, x_{j+1} \middle| y_{i,j}\right) \right]$$

The change is to condition the yield feature ψ on the *length* of the yield. This decreases deficiency by disallowing, for example, a constituent over a four-tag yield to generate a seven-tag sequence. It also decreases inter-parameter dependence by breaking the constituent (and non-constituent) distributions into a separate bin for each possible constituent length. We will refer to Klein and Manning's CCM and our version as models 1 and 2, respectively.

6.2 Experiment

We ran experiments using both CCM models on the tag sequences of length ten or less in the Wall Street Journal Penn Treebank corpus, after extracting punctuation. This corpus consists of 7,519 sentences (52,837 tag tokens, 38 types). We report PARSEVAL scores averaged by constituent (rather than by sentence), and do not give the learner credit for getting full sentences or single tags as constituents.[12] Because the E step for this model is computationally intensive, we set the DA parameters at $\beta_{\min} = 0.01, \alpha = 1.5$ so that fewer E steps would be necessary.[13] The convergence criterion was relative improvement $< 10^{-9}$ in the objective.

The results are shown in Table 2. The first point to notice is that a uniform initializer is a bad idea,

[11] We refer readers to Klein and Manning (2002) or Cover and Thomas (1991, p. 72) for details; computing expected counts for a sentence is a closed form operation. Klein and Manning's argument for this initialization step is that it is less biased toward balanced trees than the uniform model used during learning; we also found that it works far better in practice.

[12] This is why the CCM 1 performance reported here differs from Klein and Manning's; our implementation of the EM condition gave virtually identical results under either evaluation scheme (D. Klein, personal communication).

[13] A pilot study got very similar results for $\beta_{\min} = 10^{-6}$.

		E steps	cross-entropy (bits/tag)	UR	UP	F	CB
CCM 1	EM (uniform)	146	103.1654	61.20	45.62	52.27	1.69
	DA	403	**103.1542**	55.13	41.10	47.09	1.91
	EM (split)	124	103.1951	78.14	58.24	66.74	0.98
	SDA (split)	339	103.1651	62.71	46.75	53.57	1.62
CCM 2	EM (uniform)	26	84.8106	57.60	42.94	49.20	1.86
	DA	331	**84.7899**	40.81	30.42	34.86	2.66
	EM (split)	44	84.8049	78.56	58.56	67.10	0.98
	SDA (split)	290	84.7940	**79.64**	**59.37**	**68.03**	**0.93**

Table 2: The two CCM models, trained with two unsupervised algorithms, each with two initializers. Note that DA is equivalent to SDA initialized with a uniform distribution. The third line corresponds to the setup reported by Klein and Manning (2002). UR is unlabeled recall, UP is unlabeled precision, F is their harmonic mean, and CB is the average number of crossing brackets per sentence. All evaluation is on the same data used for unsupervised learning (i.e., there is no training/test split). The high cross-entropy values arise from the deficiency of models 1 and 2, and are not comparable across models.

as Klein and Manning predicted. All conditions but one find better structure when initialized with Klein and Manning's random-split model. (The exception is SDA on model 1; possibly the high deficiency of model 1 interacts poorly with SDA's search in some way.)

Next we note that with the random-split initializer, our model 2 is a bit better than model 1 on PARSEVAL measures and converges more quickly.

Every instance of DA or SDA achieved higher log-likelihood than the corresponding EM condition. This is what we hoped to gain from annealing: better local maxima. In the case of model 2 with the random-split initializer, SDA significantly outperformed EM (comparing both matches and crossing brackets per sentence under a binomial sign test, $p < 10^{-6}$); we see a $> 5\%$ reduction in average crossing brackets per sentence. Thus, our strategy of using DA but modifying it to accept an initializer worked as desired in this case, yielding our best overall performance.

The systematic results we describe next suggest that these patterns persist across different training sets in this domain.

6.3 Significance

The difficulty we experienced in finding generalization to small datasets, discussed in §4.2, was apparent here as well. For 10-way and 3-way random, nonoverlapping splits of the dataset, we did not have consistent results in favor of either EM or SDA. Interestingly, we found that training model 2 (using EM or SDA) on 10% of the corpus resulted on average in models that performed nearly as well on their respective training sets as the full corpus condition did on its training set; see Table 3. In addition, SDA sometimes performed as well as EM under model 1. For a random two-way split, EM and SDA converged to almost identical solutions on one of the sub-corpora, and SDA outperformed EM significantly on the other (on model 2).

In order to get multiple points of comparison of EM and SDA on this task with a larger amount of data, we jack-knifed the WSJ-10 corpus by splitting it randomly into ten equally-sized nonoverlapping parts then training models on the corpus with each of the ten sub-corpora excluded.[14] These trials are not independent of each other; any two of the sub-corpora have $\frac{8}{9}$ of their training data in common. Aggregate results are shown in Table 3. Using model 2, SDA always outperformed EM, and in 8 of 10 cases the difference was significant when comparing matching constituents per sentence (7 of 10 when comparing crossing constituents).[15] The variance of SDA was far less than that of EM; SDA not only always performed better with model 2, but its performance was more consistent over the trials.

We conclude this experimental discussion by cautioning that both CCM models are highly deficient models, and it is unknown how well they generalize to corpora of longer sentences, other languages, or corpora of words (rather than POS tags).

7 Future work

There are a number of interesting directions for future work. Noting the simplicity of the DA algorithm, we hope that current devotees of EM will run comparisons of their models with DA (or SDA).

[14] Note that this is *not* a cross-validation experiment; results are reported on the unlabeled training set, and the excluded sub-corpus remains unused.

[15] Binomial sign test, with significance defined as $p < 0.05$, though all significant results had $p < 0.001$.

		10% corpus		90% corpus	
		μ_F	σ_F	μ_F	σ_F
CCM 1	EM	65.00	1.091	66.12	0.6643
	SDA	63.00	4.689	53.53	0.2135
CCM 2	EM	66.74	1.402	67.24	0.7077
	SDA	**66.77**	**1.034**	**68.07**	**0.1193**

Table 3: The mean μ and standard deviation σ of F-measure performance for 10 trials using 10% of the corpus and 10 jack-knifed trials using 90% of the corpus.

Not only might this improve performance of existing systems, it will contribute to the general understanding of the likelihood surface for a variety of problems (e.g., this paper has raised the question of how factors like dataset size and model deficiency affect the likelihood surface).

DA provides a very natural way to gradually introduce complexity to clustering models (Rose et al., 1990; Pereira et al., 1993). This comes about by manipulating the β parameter; as it rises, the number of effective clusters is allowed to increase. An open question is whether the analogues of "clusters" in tagging and parsing models—tag symbols and grammatical categories, respectively—might be treated in a similar manner under DA. For instance, we might begin with the CCM, the original formulation of which posits only one distinction about constituency (whether a span is a constituent or not) and gradually allow splits in constituent-label space, resulting in multiple grammatical categories that, we hope, arise naturally from the data.

In this paper, we used $\beta_{\max} = 1$. It would be interesting to explore the effect on accuracy of "quenching," a phase at the end of optimization that rapidly raises β from 1 to the winner-take-all (Viterbi) variant at $\beta = +\infty$.

Finally, certain practical speedups may be possible. For instance, increasing $\beta_{\min}$ and α, as noted in §2.2, will vary the number of E steps required for convergence. We suggested that the change might result in slower or faster convergence; optimizing the schedule using an online algorithm (or determining precisely how these parameters affect the schedule in practice) may prove beneficial. Another possibility is to relax the convergence criterion for earlier β values, requiring fewer E steps before increasing β, or even raising β slightly after every E step (collapsing the outer and inner loops).

8 Conclusion

We have reviewed the DA algorithm, describing it as a generalization of EM with certain desirable properties, most notably the gradual increase of difficulty of learning and the ease of implementation for NLP models. We have shown how DA can be used to improve the accuracy of a trigram POS tagger learned from an unlabeled corpus. We described a potential shortcoming of DA for NLP applications—its failure to exploit good initializers—and then described a novel algorithm, *skewed* DA, that solves this problem. Finally, we reported significant improvements to a state-of-the-art grammar induction model using SDA and a slight modification to the parameterization of that model. These results support the case that annealing techniques in some cases offer performance gains over the standard EM approach to learning from unlabeled corpora, particularly with large corpora.

Acknowledgements

This work was supported by a fellowship to the first author from the Fannie and John Hertz Foundation, and by NSF ITR grant IIS-0313193 to the second author. The views expressed are not necessarily endorsed by the sponsors. The authors thank Shankar Kumar, Charles Schafer, David Smith, and Roy Tromble for helpful comments and discussions; three ACL reviewers for advice that improved the paper; Eric Goldlust for keeping the Dyna compiler (Eisner et al., 2004) up to date with the demands made by this work; and Dan Klein for sharing details of his CCM implementation.

References

P. F. Brown, J. Cocke, S. A. Della Pietra, V. J. Della Pietra, F. Jelinek, J. D. Lafferty, R. L. Mercer, and P. S. Roossin. 1990. A statistical approach to machine translation. *Computational Linguistics*, 16(2):79–85.

E. Charniak. 1993. *Statistical Language Learning*. MIT Press.

M. Collins and Y. Singer. 1999. Unsupervised models for named-entity classification. In *Proc. of EMNLP*.

T. M. Cover and J. A. Thomas. 1991. *Elements of Information Theory*. John Wiley and Sons.

S. Cucerzan and D. Yarowsky. 2003. Minimally supervised induction of grammatical gender. In *Proc. of HLT/NAACL*.

A. Dempster, N. Laird, and D. Rubin. 1977. Maximum likelihood estimation from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society B*, 39:1–38.

J. Eisner, E. Goldlust, and N. A. Smith. 2004. Dyna: A declarative language for implementing dynamic programs. In *Proc. of ACL* (companion volume).

D. Elworthy. 1994. Does Baum-Welch re-estimation help taggers? In *Proc. of ANLP*.

S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. 1983. Optimization by simulated annealing. *Science*, 220:671–680.

D. Klein and C. D. Manning. 2002. A generative constituent-context model for grammar induction. In *Proc. of ACL*.

B. Merialdo. 1994. Tagging English text with a probabilistic model. *Computational Linguistics*, 20(2):155–72.

R. Neal and G. Hinton. 1998. A view of the EM algorithm that justifies incremental, sparse, and other variants. In M. I. Jordan, editor, *Learning in Graphical Models*. Kluwer.

F. C. N. Pereira, N. Tishby, and L. Lee. 1993. Distributional clustering of English words. In *Proc. of ACL*.

A. Rao and K. Rose. 2001. Deterministically annealed design of Hidden Markov Model speech recognizers. *IEEE Transactions on Speech and Audio Processing*, 9(2):111–126.

K. Rose, E. Gurewitz, and G. C. Fox. 1990. Statistical mechanics and phase transitions in clustering. *Physical Review Letters*, 65(8):945–948.

K. Rose. 1998. Deterministic annealing for clustering, compression, classification, regression, and related optimization problems. *Proc. of the IEEE*, 86(11):2210–2239.

N. Ueda and R. Nakano. 1998. Deterministic annealing EM algorithm. *Neural Networks*, 11(2):271–282.

S. Wang, D. Schuurmans, and Y. Zhao. 2003. The latent maximum entropy principle. In review.

D. Yarowsky. 1995. Unsupervised word sense disambiguation rivaling supervised methods. In *Proc. of ACL*.