

# Minimum Risk Annealing for Training Log-Linear Models\*

David A. Smith and Jason Eisner

Department of Computer Science  
Center for Language and Speech Processing  
Johns Hopkins University  
Baltimore, MD 21218, USA  
{dasmith, eisner}@jhu.edu

## Abstract

When training the parameters for a natural language system, one would prefer to minimize 1-best *loss* (error) on an evaluation set. Since the error surface for many natural language problems is piecewise constant and riddled with local minima, many systems instead optimize log-likelihood, which is conveniently differentiable and convex. We propose training instead to minimize the *expected loss*, or *risk*. We define this expectation using a probability distribution over hypotheses that we gradually sharpen (anneal) to focus on the 1-best hypothesis. Besides the *linear* loss functions used in previous work, we also describe techniques for optimizing *nonlinear* functions such as precision or the BLEU metric. We present experiments training log-linear combinations of models for dependency parsing and for machine translation. In machine translation, annealed minimum risk training achieves significant improvements in BLEU over standard minimum error training. We also show improvements in labeled dependency parsing.

## 1 Direct Minimization of Error

Researchers in empirical natural language processing have expended substantial ink and effort in developing metrics to evaluate systems automatically against gold-standard corpora. The ongoing evaluation literature is perhaps most obvious in the machine translation community’s efforts to better BLEU (Papineni et al., 2002).

Despite this research, parsing or machine translation systems are often trained using the much simpler and harsher metric of maximum likelihood. One reason is that in supervised training, the log-likelihood objective function is generally convex, meaning that it has a single global maximum that can be easily found (indeed, for supervised generative models, the parameters at this maximum may even have a closed-form solution). In contrast to the likelihood surface, the *error* surface for discrete structured prediction is not only riddled with local minima, but piecewise constant

and not everywhere differentiable with respect to the model parameters (Figure 1). Despite these difficulties, some work has shown it worthwhile to minimize error directly (Och, 2003; Bahl et al., 1988).

We show improvements over previous work on error minimization by minimizing the **risk** or **expected error**—a continuous function that can be derived by combining the likelihood with any evaluation metric (§2). Seeking to avoid local minima, deterministic annealing (Rose, 1998) gradually changes the objective function from a convex entropy surface to the more complex risk surface (§3). We also discuss regularizing the objective function to prevent overfitting (§4). We explain how to compute expected loss under some evaluation metrics common in natural language tasks (§5). We then apply this machinery to training log-linear combinations of models for dependency parsing and for machine translation (§6). Finally, we note the connections of minimum risk *training* to max-margin training and minimum Bayes risk *decoding* (§7), and recapitulate our results (§8).

## 2 Training Log-Linear Models

In this work, we focus on rescoring with log-linear models. In particular, our experiments consider log-linear combinations of a relatively small number of features over entire complex structures, such as trees or translations, known in some previous work as *products of experts* (Hinton, 1999) or *logarithmic opinion pools* (Smith et al., 2005). A feature in the combined model might thus be a log probability from an entire submodel. Giving this feature a small or negative weight can discount a submodel that is foolishly structured, badly trained, or redundant with the other features.

For each sentence  $x_i$  in our training corpus  $S$ , we are given  $K_i$  possible analyses  $y_{i,1}, \dots, y_{i,K_i}$ . (These may be all of the possible translations or parse trees; or only the  $K_i$  most probable under

\*This work was supported by an NSF graduate research fellowship for the first author and by NSF ITR grant IIS-0313193 and ONR grant N00014-01-1-0685. The views expressed are not necessarily endorsed by the sponsors. We thank Sanjeev Khudanpur, Noah Smith, Markus Dreyer, and the reviewers for helpful discussions and comments.

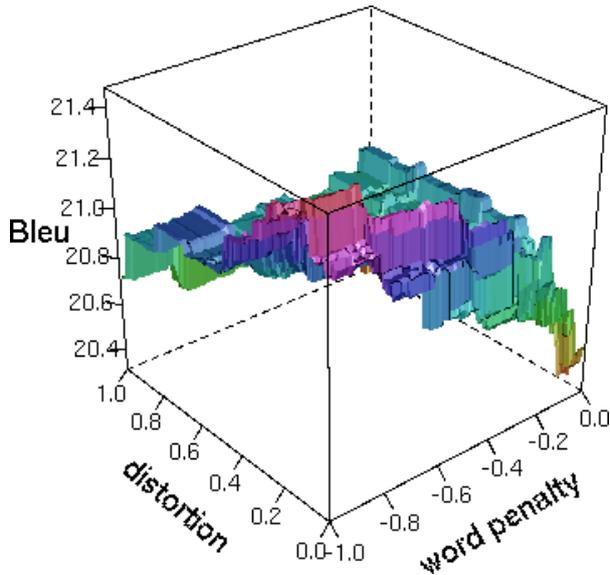


Figure 1: The loss surface for a machine translation system: while other parameters are held constant, we vary the weights on the distortion and word penalty features. Note the piecewise constant regions with several local maxima.

some other model; or only a random sample of size  $K_i$ .) Each analysis has a vector of real-valued features (i.e., factors, or experts) denoted  $f_{i,k}$ . The score of the analysis  $y_{i,k}$  is  $\theta \cdot f_{i,k}$ , the dot product of its features with a parameter vector  $\theta$ . For each sentence, we obtain a normalized probability distribution over the  $K_i$  analyses as

$$p_\theta(y_{i,k} | x_i) = \frac{\exp \theta \cdot f_{i,k}}{\sum_{k'=1}^{K_i} \exp \theta \cdot f_{i,k'}} \quad (1)$$

We wish to adjust this model’s parameters  $\theta$  to minimize the severity of the errors we make when using it to choose among analyses. A **loss function**  $L_{y^*}(y)$  assesses a penalty for choosing  $y$  when  $y^*$  is correct. We will usually write this simply as  $L(y)$  since  $y^*$  is fixed and clear from context. For clearer exposition, we assume below that the total loss over some test corpus is the sum of the losses on individual sentences, although we will revisit that assumption in §5.

## 2.1 Minimizing Loss or Expected Loss

One training criterion directly mimics test conditions. It looks at the loss incurred if we choose the best analysis of each  $x_i$  according to the model:

$$\min_{\theta} \sum_i L(\operatorname{argmax}_{y_i} p_\theta(y_i | x_i)) \quad (2)$$

Since small changes in  $\theta$  either do not change the best analysis or else push a different analysis to the top, this objective function is piecewise

constant, hence not amenable to gradient descent. Och (2003) observed, however, that the piecewise-constant property could be exploited to characterize the function exhaustively along any line in parameter space, and hence to minimize it globally along that line. By calling this global line minimization as a subroutine of multidimensional optimization, he was able to minimize (2) well enough to improve over likelihood maximization for training factored machine translation systems.

Instead of considering only the best hypothesis for any  $\theta$ , we can minimize **risk**, i.e., the **expected loss** under  $p_\theta$  across all analyses  $y_i$ :

$$\min_{\theta} \mathbf{E}_{p_\theta} L(y_{i,k}) \stackrel{\text{def}}{=} \min_{\theta} \sum_i \sum_k L(y_{i,k}) p_\theta(y_{i,k} | x_i) \quad (3)$$

This “smoothed” objective is now continuous and differentiable. However, it no longer exactly mimics test conditions, and it typically remains non-convex, so that gradient descent is still not guaranteed to find a global minimum. Och (2003) found that such smoothing during training “gives almost identical results” on translation metrics.

The simplest possible loss function is 0/1 loss, where  $L(y)$  is 0 if  $y$  is the true analysis  $y_i^*$  and 1 otherwise. This loss function does not attempt to give partial credit. Even in this simple case, assuming  $P \neq NP$ , there exists no general polynomial-time algorithm for even approximating (2) to within any constant factor, even for  $K_i = 2$  (Hoffgen et al., 1995, from Theorem 4.10.4).<sup>1</sup> The same is true for (3), since for  $K_i = 2$  it can be easily shown that the min 0/1 risk is between 50% and 100% of the min 0/1 loss.

## 2.2 Maximizing Likelihood

Rather than minimizing a loss function suited to the task, many systems (especially for language modeling) choose simply to maximize the probability of the gold standard. The log of this **likelihood** is a convex function of the parameters  $\theta$ :

$$\max_{\theta} \sum_i \log p_\theta(y_i^* | x_i) \quad (4)$$

where  $y_i^*$  is the true analysis of sentence  $x_i$ . The only wrinkle is that  $p_\theta(y_i^* | x_i)$  may be left undefined by equation (1) if  $y_i^*$  is not in our set of  $K_i$  hypotheses. When maximizing likelihood, therefore, we will replace  $y_i^*$  with the min-loss analysis in the hypothesis set; if multiple analyses tie

<sup>1</sup>Known algorithms are exponential but only in the dimensionality of the feature space (Johnson and Preparata, 1978).

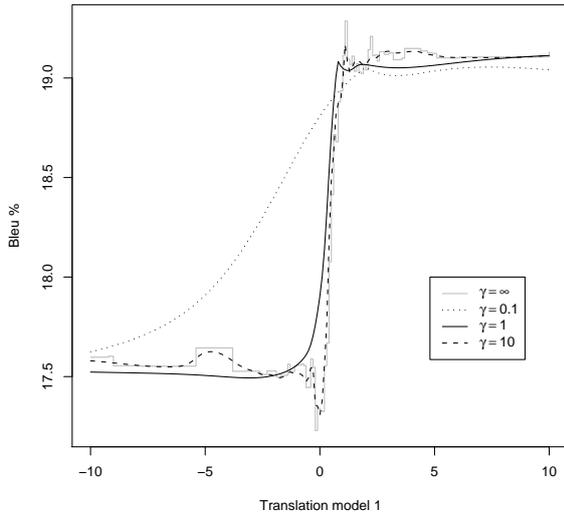


Figure 2: Loss and expected loss as one translation model’s weight varies: the gray line ( $\gamma = \infty$ ) shows true BLEU (to be optimized in equation (2)). The black lines show the expected BLEU as  $\gamma$  in equation (5) increases from 0.1 toward  $\infty$ .

for this honor, we follow Charniak and Johnson (2005) in summing their probabilities.<sup>2</sup>

Maximizing (4) is equivalent to minimizing an upper bound on the expected 0/1 loss  $\sum_i (1 - p_\theta(y_i^* | x_i))$ . Though the log makes it tractable, this remains a 0/1 objective that does not give partial credit to wrong answers, such as imperfect but useful translations. Most systems should be evaluated and preferably trained on less harsh metrics.

### 3 Deterministic Annealing

To balance the advantages of direct loss minimization, continuous risk minimization, and convex optimization, **deterministic annealing** attempts the solution of increasingly difficult optimization problems (Rose, 1998). Adding a scale hyperparameter  $\gamma$  to equation (1), we have the following family of distributions:

$$p_{\gamma,\theta}(y_{i,k} | x_i) = \frac{(\exp \theta \cdot f_{i,k})^\gamma}{\sum_{k'=1}^{K_i} (\exp \theta \cdot f_{i,k'})^\gamma} \quad (5)$$

When  $\gamma = 0$ , all  $y_{i,k}$  are equally likely, giving the uniform distribution; when  $\gamma = 1$ , we recover the model in equation (1); and as  $\gamma \rightarrow \infty$ , we approach the winner-take-all Viterbi function that assigns probability 1 to the top-scoring analysis.

For a fixed  $\gamma$ , deterministic annealing solves

$$\min_{\theta} \mathbf{E}_{p_{\gamma,\theta}}[L(y_{i,k})] \quad (6)$$

<sup>2</sup>An alternative would be to artificially add  $y_i^*$  (e.g., the reference translation(s)) to the hypothesis set during training.

We then increase  $\gamma$  according to some *schedule* and optimize  $\theta$  again. When  $\gamma$  is low, the smooth objective might allow us to pass over local minima that could open up at higher  $\gamma$ . Figure 3 shows how the smoothing is gradually weakened to reach the risk objective (3) as  $\gamma \rightarrow 1$  and approach the true error objective (2) as  $\gamma \rightarrow \infty$ .

Our risk minimization most resembles the work of Rao and Rose (2001), who trained an isolated-word speech recognition system for expected word-error rate. Deterministic annealing has also been used to tackle non-convex likelihood surfaces in unsupervised learning with EM (Ueda and Nakano, 1998; Smith and Eisner, 2004). Other work on “generalized probabilistic descent” minimizes a similar objective function but with  $\gamma$  held constant (Katagiri et al., 1998).

Although the entropy is generally higher at lower values of  $\gamma$ , it varies as the optimization changes  $\theta$ . In particular, a pure unregularized log-linear model such as (5) is really a function of  $\gamma \cdot \theta$ , so the optimizer could exactly compensate for increased  $\gamma$  by decreasing the  $\theta$  vector proportionately!<sup>3</sup> Most deterministic annealing procedures, therefore, express a direct preference on the entropy  $H$ , and choose  $\gamma$  and  $\theta$  accordingly:

$$\min_{\gamma,\theta} \mathbf{E}_{p_{\gamma,\theta}}[L(y_{i,k})] - T \cdot H(p_{\gamma,\theta}) \quad (7)$$

In place of a schedule for raising  $\gamma$ , we now use a **cooling** schedule to lower  $T$  from  $\infty$  to  $-\infty$ , thereby weakening the preference for high entropy. The Lagrange multiplier  $T$  on entropy is called “temperature” due to a satisfying connection to statistical mechanics. Once  $T$  is quite cool, it is common in practice to switch to raising  $\gamma$  directly and rapidly (**quenching**) until some convergence criterion is met (Rao and Rose, 2001).

### 4 Regularization

Informally, high temperature or  $\gamma < 1$  smooths our model during training toward higher-entropy conditional distributions that are not so peaked at the desired analyses  $y_i^*$ . Another reason for such smoothing is simply to prevent overfitting to these training examples.

A typical way to control overfitting is to use a quadratic regularizing term,  $\|\theta\|^2$  or more generally  $\sum_d \theta_d^2 / 2\sigma_d^2$ . Keeping this small keeps weights

<sup>3</sup>For such models,  $\gamma$  merely aids the nonlinear optimizer in its search, by making it easier to scale all of  $\theta$  at once.

low and entropy high. We may add this regularizer to equation (6) or (7). In the maximum likelihood framework, we may subtract it from equation (4), which is equivalent to maximum *a posteriori* estimation with a diagonal Gaussian prior (Chen and Rosenfeld, 1999). The variance  $\sigma_d^2$  may reflect a prior belief about the potential usefulness of feature  $d$ , or may be tuned on heldout data.

Another simple regularization method is to stop cooling before  $T$  reaches 0 (cf. Elidan and Friedman (2005)). If loss on heldout data begins to increase, we may be starting to overfit. This technique can be used along with annealing or quadratic regularization and can achieve additional accuracy gains, which we report elsewhere (Dreyer et al., 2006).

## 5 Computing Expected Loss

At each temperature setting of deterministic annealing, we need to minimize the expected loss on the training corpus. We now discuss how this expectation is computed. When rescoring, we assume that we simply wish to combine, in some way, statistics of whole sentences<sup>4</sup> to arrive at the overall loss for the corpus. We consider evaluation metrics for natural language tasks from two broadly applicable classes: **linear** and **nonlinear**.

A **linear** metric is a sum (or other linear combination) of the loss or gain on individual sentences. Accuracy—in dependency parsing, part-of-speech tagging, and other labeling tasks—falls into this class, as do recall, word error rate in ASR, and the crossing-brackets metric in parsing. Thanks to the linearity of expectation, we can easily compute our *expected* loss in equation (6) by adding up the expected loss on each sentence.

Some other metrics involve **nonlinear** combinations over the sentences of the corpus. One common example is precision,  $P \stackrel{\text{def}}{=} \sum_i c_i / \sum_i a_i$ , where  $c_i$  is the number of correctly posited elements, and  $a_i$  is the total number of posited elements, in the decoding of sentence  $i$ . (Depending on the task, the elements may be words, bigrams, labeled constituents, etc.) Our goal is to maximize  $P$ , so during a step of deterministic annealing, we need to maximize the *expectation* of  $P$  when the sentences are decoded randomly according to equation (5). Although this expectation is continuous and differentiable as a function of

<sup>4</sup>Computing sentence  $x_i$ 's statistics usually involves iterating over hypotheses  $y_{i,1}, \dots, y_{i,K_i}$ . If these share substructure in a hypothesis lattice, dynamic programming may help.

$\theta$ , unfortunately it seems hard to compute for any given  $\theta$ . We observe however that an equivalent goal is to minimize  $-\log P$ . Taking that as our loss function instead, equation (6) now needs to minimize the expectation of  $-\log P$ ,<sup>5</sup> which decomposes somewhat more nicely:

$$\begin{aligned} \mathbf{E}[-\log P] &= \mathbf{E}[\log \sum_i a_i - \log \sum_i c_i] \\ &= \mathbf{E}[\log A] - \mathbf{E}[\log C] \end{aligned} \quad (8)$$

where the integer random variables  $A = \sum_i a_i$  and  $C = \sum_i c_i$  count the number of posited and correctly posited elements over the whole corpus.

To approximate  $\mathbf{E}[g(A)]$ , where  $g$  is any twice-differentiable function (here  $g = \log$ ), we can approximate  $g$  locally by a quadratic, given by the Taylor expansion of  $g$  about  $A$ 's mean  $\mu_A = \mathbf{E}[A]$ :

$$\begin{aligned} \mathbf{E}[g(A)] &\approx \mathbf{E}[g(\mu_A) + (A - \mu_A)g'(\mu_A) \\ &\quad + \frac{1}{2}(A - \mu_A)^2g''(\mu_A)] \\ &= g(\mu_A) + \mathbf{E}[A - \mu_A]g'(\mu_A) \\ &\quad + \frac{1}{2}\mathbf{E}[(A - \mu_A)^2]g''(\mu_A) \\ &= g(\mu_A) + \frac{1}{2}\sigma_A^2g''(\mu_A). \end{aligned}$$

Here  $\mu_A = \sum_i \mu_{a_i}$  and  $\sigma_A^2 = \sum_i \sigma_{a_i}^2$ , since  $A$  is a sum of *independent* random variables  $a_i$  (i.e., given the current model parameters  $\theta$ , our randomized decoder decodes each sentence independently). In other words, given our quadratic approximation to  $g$ ,  $\mathbf{E}[g(A)]$  depends on the (true) distribution of  $A$  only through the single-sentence means  $\mu_{a_i}$  and variances  $\sigma_{a_i}^2$ , which can be found by enumerating the  $K_i$  decodings of sentence  $i$ . The approximation becomes arbitrarily good as we anneal  $\gamma \rightarrow \infty$ , since then  $\sigma_A^2 \rightarrow 0$  and  $\mathbf{E}[g(A)]$  focuses on  $g$  near  $\mu_A$ . For equation (8),

$$\mathbf{E}[g(A)] = \mathbf{E}[\log A] \approx \log(\mu_A) - \frac{\sigma_A^2}{2\mu_A^2}$$

and  $\mathbf{E}[\log C]$  is found similarly.

Similar techniques can be used to compute the expected logarithms of some other non-linear metrics, such as F-measure (the harmonic mean of precision and recall)<sup>6</sup> and Papineni et al. (2002)'s

<sup>5</sup>This changes the trajectory that DA takes through parameter space, but ultimately the objective is the same: as  $\gamma \rightarrow \infty$  over the course of DA, minimizing  $\mathbf{E}[-\log P]$  becomes indistinguishable from maximizing  $\mathbf{E}[P]$ .

<sup>6</sup> $R \stackrel{\text{def}}{=} C/B$ ; the count  $B$  of correct elements is *known*. So  $\log F \stackrel{\text{def}}{=} \log 2PR/(P+R) = \log 2R/(1+R/P) = \log 2C/B - \log(1+A/B)$ . Consider  $g(x) = \log 1+x/B$ .

BLEU translation metric (the geometric mean of several precisions). In particular, the expectation of log BLEU distributes over its  $N + 1$  summands:

$$\log \text{BLEU} = \min(1 - \frac{r}{A_1}, 0) + \sum_{n=1}^N w_n \log P_n$$

where  $P_n$  is the precision of the  $n$ -gram elements in the decoding.<sup>7</sup> As is standard in MT research, we take  $w_n = 1/N$  and  $N = 4$ . The first term in the BLEU score is the log *brevity penalty*, a continuous function of  $A_1$  (the total number of uni-gram tokens in the decoded corpus) that fires only if  $A_1 < r$  (the average word count of the reference corpus). We again use a Taylor series to approximate the expected log brevity penalty.

We mention an alternative way to compute (say) the expected precision  $C/A$ : integrate numerically over the *joint* density of  $C$  and  $A$ . How can we obtain this density? As  $(C, A) = \sum_i (c_i, a_i)$  is a sum of *independent* random length-2 vectors, its mean vector and  $2 \times 2$  covariance matrix can be respectively found by summing the means and covariance matrices of the  $(c_i, a_i)$ , each exactly computed from the distribution (5) over  $K_i$  hypotheses. We can easily approximate  $(C, A)$  by the (continuous) bivariate normal with that mean and covariance matrix<sup>8</sup>—or else accumulate an exact representation of its (discrete) probability mass function by a sequence of numerical convolutions.

## 6 Experiments

We tested the above training methods on two different tasks: dependency parsing and phrase-based machine translation. Since the basic setup was the same for both, we outline it here before describing the tasks in detail.

In both cases, we start with 8 to 10 models (the “experts”) already trained on separate training data. To find the optimal coefficients  $\theta$  for a log-linear combination of these experts, we use separate development data, using the following procedure due to Och (2003):

1. **Initialization:** Initialize  $\theta$  to the 0 vector. For each development sentence  $x_i$ , set its  $K_i$ -best list to  $\emptyset$  (thus  $K_i = 0$ ).

<sup>7</sup>BLEU is careful when measuring  $c_i$  on a particular decoding  $y_{i,k}$ . It only counts the first two copies of *the* (e.g.) as correct if *the* occurs at most twice in any reference translation of  $x_i$ . This “clipping” does not affect the rest of our method.

<sup>8</sup>Reasonable for a large corpus, by Lyapunov’s central limit theorem (allows non-identically distributed summands).

2. **Decoding:** For each development sentence  $x_i$ , use the current  $\theta$  to extract the 200 analyses  $y_{i,k}$  with the greatest scores  $\exp \theta \cdot f_{i,k}$ . Calculate each analysis’s loss statistics (e.g.,  $c_i$  and  $a_i$ ), and add it to the  $K_i$ -best list if it is not already there.
3. **Convergence:** If  $K_i$  has not increased for any development sentence, or if we have reached our limit of 20 iterations, stop: the search has converged.
4. **Optimization:** Adjust  $\theta$  to improve our objective function over the whole development corpus. Return to step 2.

Our experiments simply compare three procedures at step 4. We may either

- **maximize log-likelihood (4)**, a convex function, at a given level of quadratic regularization, by BFGS gradient descent;
- **minimize error (2)** by Och’s line search method, which *globally* optimizes each component of  $\theta$  while holding the others constant,<sup>9</sup> or
- minimize the same error (2) more effectively, by raising  $\gamma \rightarrow \infty$  while **minimizing the annealed risk (6)**, that is, cooling  $T \rightarrow -\infty$  (or  $\gamma \rightarrow \infty$ ) and at each value, locally minimizing equation (7) using BFGS.

Since these different optimization procedures will usually find different  $\theta$  at step 4, their  $K$ -best lists will diverge after the first iteration.

For final testing, we selected among several variants of each procedure using a separate small heldout set. Final results are reported for a larger, disjoint test set.

### 6.1 Machine Translation

For our machine translation experiments, we trained phrase-based alignment template models of Finnish-English, French-English, and German-English, as follows. For each language pair, we aligned 100,000 sentence pairs from European Parliament transcripts using GIZA++. We then used Philip Koehn’s phrase extraction software to merge the GIZA++ alignments and to extract

<sup>9</sup>The component whose optimization achieved the lowest loss is then updated. The process iterates until no lower loss can be found. In contrast, Papineni (1999) proposed a linear programming method that may search along diagonal lines.

and score the alignment template model’s phrases (Koehn et al., 2003).

The Pharaoh phrase-based decoder uses precisely the setup of this paper. It scores a candidate translation (including its phrasal alignment to the original text) as  $\theta \cdot f$ , where  $f$  is a vector of the following 8 features:

1. the probability of the source phrase given the target phrase
2. the probability of the target phrase given the source phrase
3. the weighted lexical probability of the source words given the target words
4. the weighted lexical probability of the target words given the source words
5. a phrase penalty that fires for each template in the translation
6. a distortion penalty that fires when phrases translate out of order
7. a word penalty that fires for each English word in the output
8. a trigram language model estimated on the English side of the bitext

Our goal was to train the weights  $\theta$  of these 8 features. We used the method described above, employing the Pharaoh decoder at step 2 to generate the 200-best translations according to the current  $\theta$ . As explained above, we compared three procedures at step 4: maximum log-likelihood by gradient ascent; minimum error using Och’s line-search method; and annealed minimum risk. As our development data for training  $\theta$ , we used 200 sentence pairs for each language pair.

Since our methods can be tuned with hyperparameters, we used performance on a separate 200-sentence held-out set to choose the best hyperparameter values. The hyperparameter levels for each method were

- **maximum likelihood:** a Gaussian prior with all  $\sigma_d^2$  at 0.25, 0.5, 1, or  $\infty$
- **minimum error:** 1, 5, or 10 different random starting points, drawn from a uniform

Optimization Procedure	Finnish-English	French-English	German-English
Max. like.	5.02	5.31	7.43
Min. error	10.27	26.16	20.94
Ann. min. risk	<b>16.43</b>	<b>27.31</b>	<b>21.30</b>

Table 1: BLEU 4n1 percentage on translating 2000-sentence test corpora, after training the 8 experts on 100,000 sentence pairs and fitting their weights  $\theta$  on 200 more, using settings tuned on a further 200. The current minimum risk annealing method achieved significant improvements over minimum error and maximum likelihood at or below the 0.001 level, using a permutation test with 1000 replications.

distribution on  $[-1, 1] \times [-1, 1] \times \dots$ , when optimizing  $\theta$  at an iteration of step 4.<sup>10</sup>

- **annealed minimum risk:** with explicit entropy constraints, starting temperature  $T \in \{100, 200, 1000\}$ ; stopping temperature  $T \in \{0.01, 0.001\}$ . The temperature was cooled by half at each step; then we quenched by doubling  $\gamma$  at each step. (We also ran experiments with quadratic regularization with all  $\sigma_d^2$  at 0.5, 1, or 2 (§4) in addition to the entropy constraint. Also, instead of the entropy constraint, we simply annealed on  $\gamma$  while adding a quadratic regularization term. None of these regularized models beat the best setting of standard deterministic annealing on heldout or test data.)

Final results on a separate 2000-sentence test set are shown in table 1. We evaluated translation using BLEU with one reference translation and  $n$ -grams up to 4. The minimum risk annealing procedure significantly outperformed maximum likelihood and minimum error training in all three language pairs ( $p < 0.001$ , paired-sample permutation test with 1000 replications).

Minimum risk annealing generally outperformed minimum error training on the held-out set, regardless of the starting temperature  $T$ . However, higher starting temperatures do give better performance and a more monotonic learning curve (Figure 3), a pattern that held up on test data. (In the same way, for minimum error training,

<sup>10</sup>That is, we run step 4 from several starting points, finishing at several different points; we pick the finishing point with lowest development error (2). This reduces the sensitivity of this method to the starting value of  $\theta$ . Maximum likelihood is not sensitive to the starting value of  $\theta$  because it has only a global optimum; annealed minimum risk is not sensitive to it either, because initially  $\gamma \approx 0$ , making equation (6) flat.

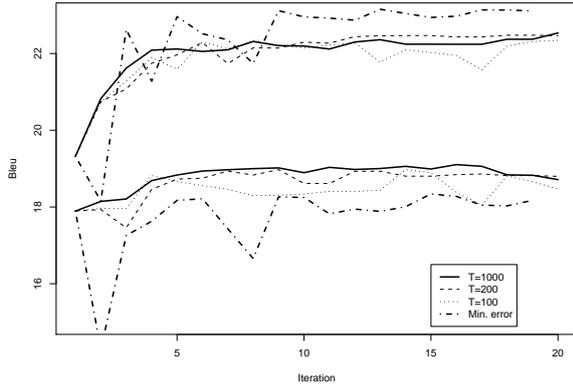


Figure 3: Iterative change in BLEU on German-English development (upper) and held-out (lower), under annealed minimum risk training with different *starting* temperatures, versus minimum error training with 10 random restarts.

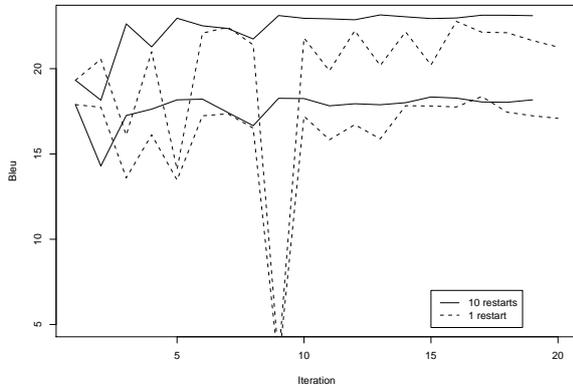


Figure 4: Iterative change in BLEU on German-English development (upper) and held-out (lower), using 10 random restarts vs. only 1.

more random restarts give better performance and a more monotonic learning curve—see Figure 4.)

Minimum risk annealing did *not* always win on the *training* set, suggesting that its advantage is not superior minimization but rather superior generalization: under the risk criterion, *multiple* low-loss hypotheses per sentence can help guide the learner to the right part of parameter space.

Although the components of the translation and language models interact in complex ways, the improvement on Finnish-English may be due in part to the higher weight that minimum risk annealing found for the word penalty. That system is therefore more likely to produce shorter output like *i have taken note of your remarks and i also agree with that* . than like this longer output from the minimum-error-trained system: *i have taken note of your remarks and i shall also agree with all that the union* .

We annealed using our novel expected-BLEU

approximation from §5. We found this to perform significantly better on BLEU evaluation than if we trained with a “linearized” BLEU that summed per-sentence BLEU scores (as used in minimum Bayes risk decoding by Kumar and Byrne (2004)).

## 6.2 Dependency Parsing

We trained dependency parsers for three different languages: Bulgarian, Dutch, and Slovenian.<sup>11</sup> Input sentences to the parser were already tagged for parts of speech. Each parser employed 10 experts, each parameterized as a globally normalized log-linear model (Lafferty et al., 2001). For example, the 9<sup>th</sup> component of the feature vector  $f_{i,k}$  (which described the  $k^{\text{th}}$  parse of the  $i^{\text{th}}$  sentence) was the log of that parse’s normalized probability according to the 9<sup>th</sup> expert.

Each expert was trained separately to maximize the conditional probability of the correct parse given the sentence. We used 10 iterations of gradient ascent. To speed training, for each of the first 9 iterations, the gradient was estimated on a (different) sample of only 1000 training sentences.

We then trained the vector  $\theta$ , used to combine the experts, to minimize the number of labeled dependency attachment errors on a 200-sentence development set. Optimization proceeded over lists of the 200-best parses of each sentence produced by a joint decoder using the 10 experts.

Evaluating on labeled dependency accuracy on 200 test sentences for each language, we see that minimum error and annealed minimum risk training are much closer than for MT. For Bulgarian and Dutch, they are statistically indistinguishable using a paired-sample permutations test with 1000 replications. Indeed, on Dutch, all three optimization procedures produce indistinguishable results. On Slovenian, annealed minimum risk training does show a significant improvement over the other two methods. Overall, however, the results for this task are mediocre. We are still working on improving the underlying experts.

## 7 Related Work

We have seen that annealed minimum risk training provides a useful alternative to maximum likelihood and minimum error training. In our experiments, it never performed significantly worse

<sup>11</sup>For information on these corpora, see the CoNLL-X shared task on multilingual dependency parsing: <http://nextens.uvt.nl/~conll/>.

Optimization Procedure	labeled dependency acc. [%]		
	Slovenian	Bulgarian	Dutch
Max. like.	27.78	47.23	<b>36.78</b>
Min. error	22.52	<b>54.72</b>	<b>36.78</b>
Ann. min. risk	<b>31.16</b>	54.66	36.71

Table 2: Labeled dependency accuracy on parsing 200-sentence test corpora, after training 10 experts on 1000 sentences and fitting their weights  $\theta$  on 200 more. For Slovenian, minimum risk annealing is significantly better than the other training methods, while minimum error is significantly worse. For Bulgarian, both minimum error and annealed minimum risk training achieve significant gains over maximum likelihood, but are indistinguishable from each other. For Dutch, the three methods are indistinguishable.

than either and in some cases significantly helped. Note, however, that annealed minimum risk training results in a deterministic classifier just as these other training procedures do. The orthogonal technique of **minimum Bayes risk decoding** has achieved gains on parsing (Goodman, 1996) and machine translation (Kumar and Byrne, 2004). In speech recognition, researchers have improved decoding by smoothing probability estimates numerically on heldout data in a manner reminiscent of annealing (Goel and Byrne, 2000). We are interested in applying our techniques for approximating nonlinear loss functions to MBR by performing the risk minimization inside the dynamic programming or other decoder.

Another training approach that incorporates arbitrary loss functions is found in the *structured prediction* literature in the margin-based-learning community (Taskar et al., 2004; Crammer et al., 2004). Like other max-margin techniques, these attempt to make the best hypothesis far away from the inferior ones. The distinction is in using a loss function to calculate the required margins.

## 8 Conclusions

Despite the challenging shape of the error surface, we have seen that it is practical to optimize task-specific error measures rather than optimizing likelihood—it produces lower-error systems. Different methods can be used to attempt this global, non-convex optimization. We showed that for MT, and sometimes for dependency parsing, an annealed minimum risk approach to optimization performs significantly better than a previous line-search method that does not smooth the error surface. It never does significantly worse. With such improved methods for minimizing error, we can hope to make better use of task-specific

training criteria in NLP.

## References

- L. R. Bahl, P. F. Brown, P. V. de Souza, and R. L. Mercer. 1988. A new algorithm for the estimation of hidden Markov model parameters. In *ICASSP*, pages 493–496.
- E. Charniak and M. Johnson. 2005. Coarse-to-fine n-best parsing and maxent discriminative reranking. In *ACL*, pages 173–180.
- S. F. Chen and R. Rosenfeld. 1999. A gaussian prior for smoothing maximum entropy models. Technical report, CS Dept., Carnegie Mellon University.
- K. Crammer, R. McDonald, and F. Pereira. 2004. New large margin algorithms for structured prediction. In *Learning with Structured Outputs (NIPS)*.
- M. Dreyer, D. A. Smith, and N. A. Smith. 2006. Vine parsing and minimum risk reranking for speed and precision. In *CoNLL*.
- G. Elidan and N. Friedman. 2005. Learning hidden variable networks: The information bottleneck approach. *JMLR*, 6:81–127.
- V. Goel and W. J. Byrne. 2000. Minimum Bayes-Risk automatic speech recognition. *Computer Speech and Language*, 14(2):115–135.
- J. T. Goodman. 1996. Parsing algorithms and metrics. In *ACL*, pages 177–183.
- G. Hinton. 1999. Products of experts. In *Proc. of ICANN*, volume 1, pages 1–6.
- K.-U. Hoffgen, H.-U. Simon, and K. S. Van Horn. 1995. Robust trainability of single neurons. *J. of Computer and System Sciences*, 50(1):114–125.
- D. S. Johnson and F. P. Preparata. 1978. The densest hemisphere problem. *Theoretical Comp. Sci.*, 6(93–107).
- S. Katagiri, B.-H. Juang, and C.-H. Lee. 1998. Pattern recognition using a family of design algorithms based upon the generalized probabilistic descent method. *Proc. IEEE*, 86(11):2345–2373, November.
- P. Koehn, F. J. Och, and D. Marcu. 2003. Statistical phrase-based translation. In *HLT-NAACL*, pages 48–54.
- S. Kumar and W. Byrne. 2004. Minimum bayes-risk decoding for statistical machine translation. In *HLT-NAACL*.
- J. Lafferty, A. McCallum, and F. C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*.
- F. J. Och. 2003. Minimum error rate training in statistical machine translation. In *ACL*, pages 160–167.
- K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu. 2002. BLEU: A method for automatic evaluation of machine translation. In *ACL*, pages 311–318.
- K. A. Papineni. 1999. Discriminative training via linear programming. In *ICASSP*.
- A. Rao and K. Rose. 2001. Deterministically annealed design of Hidden Markov Model speech recognizers. *IEEE Trans. on Speech and Audio Processing*, 9(2):111–126.
- K. Rose. 1998. Deterministic annealing for clustering, compression, classification, regression, and related optimization problems. *Proc. IEEE*, 86(11):2210–2239.
- N. A. Smith and J. Eisner. 2004. Annealing techniques for unsupervised statistical language learning. In *ACL*, pages 486–493.
- A. Smith, T. Cohn, and M. Osborne. 2005. Logarithmic opinion pools for conditional random fields. In *ACL*, pages 18–25.
- B. Taskar, D. Klein, M. Collins, D. Koller, and C. Manning. 2004. Max-margin parsing. In *EMNLP*, pages 1–8.
- N. Ueda and R. Nakano. 1998. Deterministic annealing EM algorithm. *Neural Networks*, 11(2):271–282.