Analyzing the Errors of Unsupervised Learning

Percy Liang **Dan Klein**
Computer Science Division, EECS Department
University of California at Berkeley
Berkeley, CA 94720
{pliang,klein}@cs.berkeley.edu

Abstract

We identify four types of errors that unsupervised induction systems make and study each one in turn. Our contributions include (1) using a *meta-model* to analyze the incorrect biases of a model in a systematic way, (2) providing an efficient and robust method of measuring distance between two parameter settings of a model, and (3) showing that local optima issues which typically plague EM can be somewhat alleviated by increasing the number of training examples. We conduct our analyses on three models: the HMM, the PCFG, and a simple dependency model.

1 Introduction

The unsupervised induction of linguistic structure from raw text is an important problem both for understanding language acquisition and for building language processing systems such as parsers from limited resources. Early work on inducing grammars via EM encountered two serious obstacles: the inappropriateness of the likelihood objective and the tendency of EM to get stuck in local optima. Without additional constraints on bracketing (Pereira and Shabes, 1992) or on allowable rewrite rules (Carroll and Charniak, 1992), unsupervised grammar learning was ineffective.

Since then, there has been a large body of work addressing the flaws of the EM-based approach. Syntactic models empirically more learnable than PCFGs have been developed (Clark, 2001; Klein and Manning, 2004). Smith and Eisner (2005) proposed a new objective function; Smith and Eisner (2006) introduced a new training procedure. Bayesian approaches can also improve performance (Goldwater and Griffiths, 2007; Johnson, 2007; Kurihara and Sato, 2006).

Though these methods have improved induction accuracy, at the core they all still involve optimizing non-convex objective functions related to the likelihood of some model, and thus are not completely immune to the difficulties associated with early approaches. It is therefore important to better understand the behavior of unsupervised induction systems in general.

In this paper, we take a step back and present a more statistical view of unsupervised learning in the context of grammar induction. We identify four types of error that a system can make: *approximation*, *identifiability*, *estimation*, and *optimization* errors (see Figure 1). We try to isolate each one in turn and study its properties.

Approximation error is caused by a mis-match between the likelihood objective optimized by EM and the true relationship between sentences and their syntactic structures. Our key idea for understanding this mis-match is to "cheat" and initialize EM with the true relationship and then study the ways in which EM repurposes our desired syntactic structures to increase likelihood. We present a *meta-model* of the changes that EM makes and show how this tool can shed some light on the undesired biases of the HMM, the PCFG, and the dependency model with valence (Klein and Manning, 2004).

Identifiability error can be incurred when two distinct parameter settings yield the same probability distribution over sentences. One type of non-identifiability present in HMMs and PCFGs is label symmetry, which even makes computing a meaningful distance between parameters NP-hard. We present a method to obtain lower and upper bounds on such a distance.

Estimation error arises from having too few training examples, and optimization error stems from

EM getting stuck in local optima. While it is to be expected that estimation error should decrease as the amount of data increases, we show that optimization error can also decrease. We present striking experiments showing that if our data actually comes from the model family we are learning with, we can sometimes recover the true parameters by simply running EM without clever initialization. This result runs counter to the conventional attitude that EM is doomed to local optima; it suggests that increasing the amount of data might be an effective way to partially combat local optima.

2 Unsupervised models

Let $\mathbf{x}$ denote an input sentence and $\mathbf{y}$ denote the unobserved desired output (e.g., a parse tree). We consider a model family $\mathcal{P} = \{p_\theta(\mathbf{x}, \mathbf{y}) : \theta \in \Theta\}$. For example, if $\mathcal{P}$ is the set of all PCFGs, then the parameters θ would specify all the rule probabilities of a particular grammar. We sometimes use θ and p_θ interchangeably to simplify notation. In this paper, we analyze the following three model families:

In the **HMM**, the input $\mathbf{x}$ is a sequence of words and the output $\mathbf{y}$ is the corresponding sequence of part-of-speech tags.

In the **PCFG**, the input $\mathbf{x}$ is a sequence of POS tags and the output $\mathbf{y}$ is a binary parse tree with yield $\mathbf{x}$. We represent $\mathbf{y}$ as a multiset of binary rewrites of the form $(y \to y_1\, y_2)$, where y is a nonterminal and y_1, y_2 can be either nonterminals or terminals.

In the dependency model with valence (**DMV**) (Klein and Manning, 2004), the input $\mathbf{x} = (x_1, \ldots, x_m)$ is a sequence of POS tags and the output $\mathbf{y}$ specifies the directed links of a projective dependency tree. The generative model is as follows: for each head x_i, we generate an independent sequence of arguments to the left and to the right from a direction-dependent distribution over tags. At each point, we stop with a probability parametrized by the direction and whether any arguments have already been generated in that direction. See Klein and Manning (2004) for a formal description.

In all our experiments, we used the Wall Street Journal (WSJ) portion of the Penn Treebank. We binarized the PCFG trees and created gold dependency trees according to the Collins head rules. We trained 45-state HMMs on all 49208 sentences, 11-state PCFGs on WSJ-10 (7424 sentences) and DMVs on WSJ-20 (25523 sentences) (Klein and Manning, 2004). We ran EM for 100 iterations with the parameters initialized uniformly (always plus a small amount of random noise). We evaluated the HMM and PCFG by mapping model states to Treebank tags to maximize accuracy.

3 Decomposition of errors

Now we will describe the four types of errors (Figure 1) more formally. Let $p^*(\mathbf{x}, \mathbf{y})$ denote the distribution which governs the true relationship between the input $\mathbf{x}$ and output $\mathbf{y}$. In general, p^* does not live in our model family $\mathcal{P}$. We are presented with a set of n unlabeled examples $\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(n)}$ drawn i.i.d. from the true p^*. In unsupervised induction, our goal is to approximate p^* by some model $p_\theta \in \mathcal{P}$ in terms of strong generative capacity. A standard approach is to use the EM algorithm to optimize the empirical likelihood $\hat{\mathbb{E}} \log p_\theta(\mathbf{x})$.[1] However, EM only finds a local maximum, which we denote $\hat{\theta}_{\text{EM}}$, so there is a *discrepancy* between what we get ($p_{\hat{\theta}_{\text{EM}}}$) and what we want ($p^*$).

We will define this discrepancy later, but for now, it suffices to remark that the discrepancy depends on the distribution over $\mathbf{y}$ whereas learning depends only on the distribution over $\mathbf{x}$. This is an important property that distinguishes unsupervised induction from more standard supervised learning or density estimation scenarios.

Now let us walk through the four types of error bottom up. First, $\hat{\theta}_{\text{EM}}$, the local maximum found by EM, is in general different from $\hat{\theta} \in \text{argmax}_\theta\, \hat{\mathbb{E}} \log p_\theta(\mathbf{x})$, any global maximum, which we could find given unlimited computational resources. *Optimization error* refers to the discrepancy between $\hat{\theta}$ and $\hat{\theta}_{\text{EM}}$.

Second, our training data is only a noisy sample from the true p^*. If we had infinite data, we would choose an optimal parameter setting under the model, $\theta_2^* \in \text{argmax}_\theta\, \mathbb{E} \log p_\theta(\mathbf{x})$, where now the expectation $\mathbb{E}$ is taken with respect to the true p^* instead of the training data. The discrepancy between θ_2^* and $\hat{\theta}$ is the *estimation error*.

Note that θ_2^* might not be unique. Let θ_1^* denote

[1] Here, the expectation $\hat{\mathbb{E}} f(\mathbf{x}) \stackrel{\text{def}}{=} \frac{1}{n} \sum_{i=1}^n f(\mathbf{x}^{(i)})$ denotes averaging some function f over the training data.

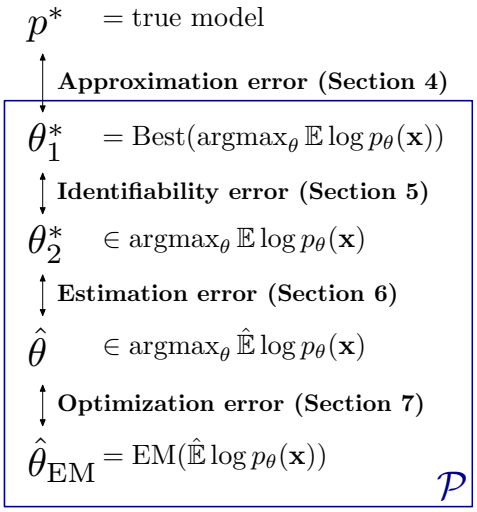


Figure 1: The discrepancy between what we get ($\hat{\theta}_{\text{EM}}$) and what we want (p^*) can be decomposed into four types of errors. The box represents our model family $\mathcal{P}$, which is the set of possible parametrized distributions we can represent. Best(S) returns the $\theta \in S$ which has the smallest discrepancy with p^*.

the maximizer of $\mathbb{E} \log p_\theta(\mathbf{x})$ that has the smallest discrepancy with p^*. Since θ_1^* and θ_2^* have the same value under the objective function, we would not be able to choose θ_1^* over θ_2^*, even with infinite data or unlimited computation. Identifiability error refers to the discrepancy between θ_1^* and θ_2^*.

Finally, the model family $\mathcal{P}$ has fundamental limitations. *Approximation error* refers to the discrepancy between p^* and $p_{\theta_1^*}$. Note that θ_1^* is not necessarily the best in $\mathcal{P}$. If we had labeled data, we could find a parameter setting in $\mathcal{P}$ which is closer to p^* by optimizing joint likelihood $\mathbb{E} \log p_\theta(\mathbf{x}, \mathbf{y})$ (generative training) or even conditional likelihood $\mathbb{E} \log p_\theta(\mathbf{y} \mid \mathbf{x})$ (discriminative training).

In the remaining sections, we try to study each of the four errors in isolation. In practice, since it is difficult to work with some of the parameter settings that participate in the error decomposition, we use computationally feasible surrogates so that the error under study remains the dominant effect.

4 Approximation error

We start by analyzing approximation error, the discrepancy between p^* and $p_{\theta_1^*}$ (the model found by optimizing likelihood), a point which has been discussed by many authors (Merialdo, 1994; Smith and Eisner, 2005; Haghighi and Klein, 2006).[2]

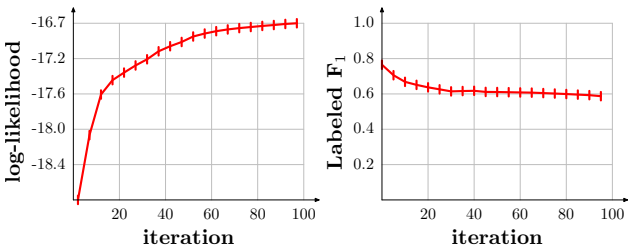


Figure 2: For the PCFG, when we initialize EM with the supervised estimate $\hat{\theta}_{\text{gen}}$, the likelihood increases but the accuracy decreases.

To confront the question of specifically how the likelihood diverges from prediction accuracy, we perform the following experiment: we initialize EM with the supervised estimate[3] $\hat{\theta}_{\text{gen}} = \text{argmax}_\theta \hat{\mathbb{E}} \log p_\theta(\mathbf{x}, \mathbf{y})$, which acts as a surrogate for p^*. As we run EM, the likelihood increases but the accuracy decreases (Figure 2 shows this trend for the PCFG; the HMM and DMV models behave similarly). We believe that the initial iterations of EM contain valuable information about the incorrect biases of these models. However, EM is changing hundreds of thousands of parameters at once in a non-trivial way, so we need a way of characterizing the important changes.

One broad observation we can make is that the first iteration of EM reinforces the systematic mistakes of the supervised initializer. In the first E-step, the posterior counts that are computed summarize the predictions of the supervised system. If these match the empirical counts, then the M-step does not change the parameters. But if the supervised system predicts too many JJs, for example, then the M-step will update the parameters to reinforce this bias.

4.1 A meta-model for analyzing EM

We would like to go further and characterize the specific changes EM makes. An initial approach is to find the parameters that changed the most during the first iteration (weighted by the correspond-

[2]Here, we think of discrepancy between p and p' as the error incurred when using p' for prediction on examples generated from p; in symbols, $\mathbb{E}_{(\mathbf{x},\mathbf{y})\sim p}\text{loss}(\mathbf{y}, \text{argmax}_{\mathbf{y}'} p'(\mathbf{y}' \mid \mathbf{x}))$.

[3]For all our models, the supervised estimate is solved in closed form by taking ratios of counts.

ing expected counts computed in the E-step). For the HMM, the three most changed parameters are the transitions 2:DT→8:JJ, START→0:NNP, and 8:JJ→3:NN.[4] If we delve deeper, we can see that 2:DT→3:NN (the parameter with the 10th largest change) fell and 2:DT→8:JJ rose. After checking with a few examples, we can then deduce that some nouns were retagged as adjectives. Unfortunately, this type of ad-hoc reasoning requires considerable manual effort and is rather subjective.

Instead, we propose using a general *meta-model* to analyze the changes EM makes in an automatic and objective way. Instead of treating parameters as the primary object of study, we look at predictions made by the model and study how they change over time. While a model is a distribution over sentences, a meta-model is a distribution over how the predictions of the model change.

Let $R(\mathbf{y})$ denote the set of *parts* of a prediction $\mathbf{y}$ that we are interested in tracking. Each part $(c, l) \in R(\mathbf{y})$ consists of a *configuration* c and a *location* l. For a PCFG, we define a configuration to be a rewrite rule (e.g., $c =$ PP→IN NP), and a location $l = [i, k, j]$ to be a span $[i, j]$ split at k, where the rewrite c is applied.

In this work, each configuration is associated with a parameter of the model, but in general, a configuration could be a larger unit such as a subtree, allowing one to track more complex changes. The size of a configuration governs how much the meta-model generalizes from individual examples.

Let $\mathbf{y}^{(i,t)}$ denote the model prediction on the i-th training example after t iterations of EM. To simplify notation, we write $R_t = R(\mathbf{y}^{(i,t)})$. The meta-model explains how R_t became R_{t+1}.[5]

In general, we expect a part in R_{t+1} to be explained by a part in R_t that has a similar location and furthermore, we expect the locations of the two parts to be related in some consistent way. The meta-model uses two notions to formalize this idea: a *distance* $d(l, l')$ and a *relation* $r(l, l')$. For the PCFG, $d(l, l')$ is the number of positions among i,j,k that are the same as the corresponding ones in l', and $r((i, k, j), (i', k', j')) = (\text{sign}(i - i'), \text{sign}(j - j'), \text{sign}(k - k'))$ is one of 3^3 values. We define a *migration* as a triple $(c, c', r(l, l'))$; this is the unit of change we want to extract from the meta-model.

Our meta-model provides the following generative story of how R_t becomes R_{t+1}: each new part $(c', l') \in R_{t+1}$ chooses an old part $(c, l) \in R_t$ with some probability that depends on (1) the distance between the locations l and l' and (2) the likelihood of the particular migration. Formally:

$$p^{\text{meta}}(R_{t+1} \mid R_t) = \prod_{(c',l') \in R_{t+1}} \sum_{(c,l) \in R_t} Z_{l'}^{-1} e^{-\alpha d(l,l')} p(c' \mid c, r(l, l')),$$

where $Z_l = \sum_{(c,l) \in R_t} e^{-\alpha d(l,l')}$ is a normalization constant, and α is a hyperparameter controlling the possibility of distant migrations (set to 3 in our experiments).

We learn the parameters of the meta-model with an EM algorithm similar to the one for IBM model 1. Fortunately, the likelihood objective is convex, so we need not worry about local optima.

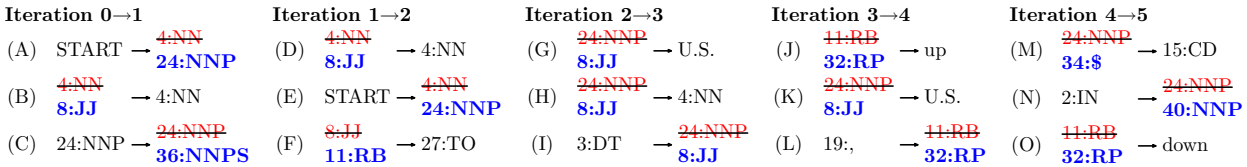
4.2 Results of the meta-model

We used our meta-model to analyze the approximation errors of the HMM, DMV, and PCFG. For these models, we initialized EM with the supervised estimate $\hat{\theta}_{\text{gen}}$ and collected the model predictions as EM ran. We then trained the meta-model on the predictions between successive iterations. The meta-model gives us an expected count for each migration. Figure 3 lists the migrations with the highest expected counts.

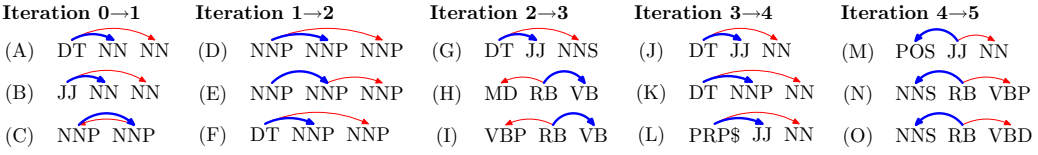
From these migrations, we can see that EM tries to explain $\mathbf{x}$ better by making the corresponding $\mathbf{y}$ more *regular*. In fact, many of the HMM migrations on the first iteration attempt to resolve inconsistencies in gold tags. For example, noun adjuncts (e.g., *stock-index*), tagged as both nouns and adjectives in the Treebank, tend to become consolidated under adjectives, as captured by migration (B). EM also re-purposes under-utilized states to better capture distributional similarities. For example, state 24 has migrated to state 40 (N), both of which are now dominated by proper nouns. State 40 initially contained only #, but was quickly overrun with distributionally similar proper nouns such as *Oct.* and *Chapter*, which also precede numbers, just as # does.

[4] Here 2:DT means state 2 of the HMM, which was greedily mapped to DT.

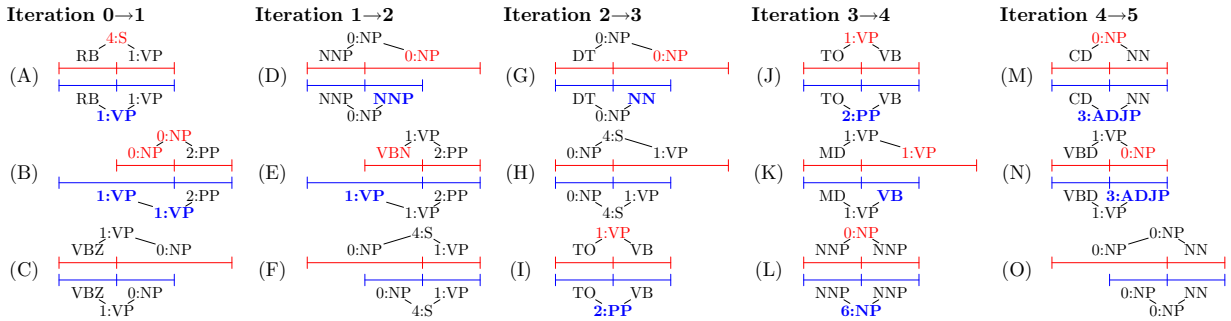
[5] If the same part appears in both R_t and R_{t+1}, we remove it from both sets.



(a) Top HMM migrations. Example: migration (D) means a NN→NN transition is replaced by JJ→NN.



(b) Top DMV migrations. Example: migration (A) means a DT attaches to the closer NN.



(c) Top PCFG migrations. Example: migration (D) means a NP→NNP NP rewrite is replaced by NP→NNP NNP, where the new NNP right child spans less than the old NP right child.

Figure 3: We show the prominent migrations that occur during the first 5 iterations of EM for the HMM, DMV, and PCFG, as recovered by our meta-model. We sort the migrations across each iteration by their expected counts under the meta-model and show the top 3. Iteration 0 corresponds to the correct outputs. Blue indicates the new iteration, red indicates the old.

DMV migrations also try to regularize model predictions, but in a different way—in terms of the number of arguments. Because the stop probability is different for adjacent and non-adjacent arguments, it is statistically much cheaper to generate one argument rather than two or more. For example, if we train a DMV on only DT JJ NN, it can fit the data perfectly by using a chain of single arguments, but perfect fit is not possible if NN generates both DT and JJ (which is the desired structure); this explains migration (J). Indeed, we observed that the variance of the number of arguments decreases with more EM iterations (for NN, from 1.38 to 0.41).

In general, low-entropy conditional distributions are preferred. Migration (H) explains how adverbs now consistently attach to verbs rather than modals. After a few iterations, the modal has committed itself to generating exactly one verb to the right, which is statistically advantageous because there must be a verb after a modal, while the adverb is optional. This leaves the verb to generate the adverb.

The PCFG migrations regularize categories in a manner similar to the HMM, but with the added complexity of changing bracketing structures. For example, sentential adverbs are re-analyzed as VP adverbs (A). Sometimes, multiple migrations explain the same phenomenon.[6] For example, migrations (B) and (C) indicate that PPs that previously attached to NPs are now raised to the verbal level. Tree rotation is another common phenomenon, leading to many left-branching structures (D,G,H). The migrations that happen during one iteration can also trigger additional migrations in the next. For example, the raising of the PP (B,C) inspires more of the

[6] We could consolidate these migrations by using larger configurations, but at the risk of decreased generalization.

same raising (E). As another example, migration (I) regularizes TO VB infinitival clauses into PPs, and this momentum carries over to the next iteration with even greater force (J).

In summary, the meta-model facilitates our analyses by automatically identifying the broad trends. We believe that the central idea of modeling the errors of a system is a powerful one which can be used to analyze a wide range of models, both supervised and unsupervised.

5 Identifiability error

While approximation error is incurred when likelihood diverges from accuracy, identifiability error is concerned with the case where likelihood is indifferent to accuracy.

We say a set of parameters S is *identifiable* (in terms of $\mathbf{x}$) if $p_\theta(\mathbf{x}) \neq p_{\theta'}(\mathbf{x})$ for every $\theta, \theta' \in S$ where $\theta \neq \theta'$.[7] In general, identifiability error is incurred when the set of maximizers of $\mathbb{E} \log p_\theta(\mathbf{x})$ is non-identifiable.[8]

Label symmetry is perhaps the most familiar example of non-identifiability and is intrinsic to models with hidden labels (HMM and PCFG, but not DMV). We can permute the hidden labels without changing the objective function or even the nature of the solution, so there is no reason to prefer one permutation over another. While seemingly benign, this symmetry actually presents a serious challenge in measuring discrepancy (Section 5.1).

Grenager et al. (2005) augments an HMM to allow emission from a generic stopword distribution at any position with probability q. Their model would definitely not be identifiable if q were a free parameter, since we can set q to 0 and just mix in the stopword distribution with each of the other emission distributions to obtain a different parameter setting yielding the same overall distribution. This is a case where our notion of desired structure is absent in the likelihood, and a prior over parameters could help break ties.

[7] For our three model families, θ is identifiable in terms of $(\mathbf{x}, \mathbf{y})$, but not in terms of $\mathbf{x}$ alone.

[8] We emphasize that non-identifiability is in terms of $\mathbf{x}$, so two parameter settings could still induce the same marginal distribution on $\mathbf{x}$ (weak generative capacity) while having different joint distributions on $(\mathbf{x}, \mathbf{y})$ (strong generative capacity). Recall that discrepancy depends on the latter.

The above non-identifiabilities apply to all parameter settings, but another type of non-identifiability concerns only the maximizers of $\mathbb{E} \log p_\theta(\mathbf{x})$. Suppose the true data comes from a K-state HMM. If we attempt to fit an HMM with $K + 1$ states, we can split any one of the K states and maintain the same distribution on $\mathbf{x}$. Or, if we learn a PCFG on the same HMM data, then we can choose either the left- or right-branching chain structures, which both mimic the true HMM equally well.

5.1 Permutation-invariant distance

KL-divergence is a natural measure of discrepancy between two distributions, but it is somewhat nontrivial to compute—for our three recursive models, it requires solving fixed point equations, and becomes completely intractable in face of label symmetry. Thus we propose a more manageable alternative:

$$d_\mu(\theta \,||\, \theta') \stackrel{\text{def}}{=} \frac{\sum_j \mu_j |\theta_j - \theta'_j|}{\sum_j \mu_j}, \tag{1}$$

where we weight the difference between the j-th component of the parameter vectors by μ_j, the j-th expected sufficient statistic with respect to p_θ (the expected counts computed in the E-step).[9] Unlike KL, our distance d_μ is only defined on distributions in the model family and is not invariant to reparametrization. Like KL, d_μ is asymmetric, with the first argument holding the status of being the "true" parameter setting. In our case, the parameters are conditional probabilities, so $0 \leq d_\mu(\theta \,||\, \theta') \leq 1$, so we can interpret d_μ as an expected difference between these probabilities.

Unfortunately, label symmetry can wreak havoc on our distance measure d_μ. Suppose we want to measure the distance between θ and θ'. If θ' is simply θ with the labels permuted, then $d_\mu(\theta \,||\, \theta')$ would be substantial even though the distance ought to be zero. We define a revised distance to correct for this by taking the minimum distance over all label permutations:

$$D_\mu(\theta \,||\, \theta') = \min_\pi \, d_\mu(\theta \,||\, \pi(\theta')), \tag{2}$$

[9] Without this factor, rarely used components could contribute to the sum as much as frequently used ones, thus, making the distance overly pessimistic.

where $\pi(\theta')$ denotes the parameter setting resulting from permuting the labels according to π. (The DMV has no label symmetries, so just d_μ works.)

For mixture models, we can compute $D_\mu(\theta \,||\, \theta')$ efficiently as follows. Note that each term in the summation of (1) is associated with one of the K labels. We can form a $K \times K$ matrix M, where each entry M_{ij} is the distance between the parameters involving label i of θ and label j of θ'. $D_\mu(\theta \,||\, \theta')$ can then be computed by finding a maximum weighted bipartite matching on M using the $O(K^3)$ Hungarian algorithm (Kuhn, 1955).

For models such as the HMM and PCFG, computing D_μ is NP-hard, since the summation in d_μ (1) contains both first-order terms which depend on one label (e.g., emission parameters) and higher-order terms which depend on more than one label (e.g., transitions or rewrites). We cannot capture these problematic higher-order dependencies in M.

However, we can bound $D_\mu(\theta \,||\, \theta')$ as follows. We create M using only first-order terms and find the best matching (permutation) to obtain a lower bound $\underline{D}_\mu$ and an associated permutation π_0 achieving it. Since $D_\mu(\theta \,||\, \theta')$ takes the minimum over all permutations, $d_\mu(\theta \,||\, \pi(\theta'))$ is an upper bound for any π, in particular for $\pi = \pi_0$. We then use a local search procedure that changes π to further tighten the upper bound. Let $\overline{D}_\mu$ denote the final value.

6 Estimation error

Thus far, we have considered approximation and identifiability errors, which have to do with flaws of the model. The remaining errors have to do with how well we can fit the model. To focus on these errors, we consider the case where the true model is in our family ($p^* \in \mathcal{P}$). To keep the setting as realistic as possible, we do supervised learning on real labeled data to obtain $\theta^* = \text{argmax}_\theta \hat{\mathbb{E}} \log p(\mathbf{x}, \mathbf{y})$. We then throw away our real data and let $p^* = p_{\theta^*}$. Now we start anew: sample new artificial data from θ^*, learn a model using this artificial data, and see how close we get to recovering θ^*.

In order to compute estimation error, we need to compare θ^* with $\hat{\theta}$, the global maximizer of the likelihood on our generated data. However, we cannot compute $\hat{\theta}$ exactly. Let us therefore first consider the simpler supervised scenario. Here, $\hat{\theta}_{\text{gen}}$ has a closed form solution, so there is no optimization error. Using our distance D_μ (defined in Section 5.1) to quantify estimation error, we see that, for the HMM, $\hat{\theta}_{\text{gen}}$ quickly approaches θ^* as we increase the amount of data (Table 1).

# examples	500	5K	50K	500K
$\underline{D}_\mu(\theta^* \,\|\|\, \hat{\theta}_{\text{gen}})$	0.003	6.3e-4	2.7e-4	8.5e-5
$\overline{D}_\mu(\theta^* \,\|\|\, \hat{\theta}_{\text{gen}})$	0.005	0.001	5.2e-4	1.7e-4
$\underline{D}_\mu(\theta^* \,\|\|\, \hat{\theta}_{\text{gen-EM}})$	0.022	0.018	0.008	0.002
$\overline{D}_\mu(\theta^* \,\|\|\, \hat{\theta}_{\text{gen-EM}})$	0.049	0.039	0.016	0.004

Table 1: Lower and upper bounds on the distance from the true θ^* for the HMM as we increase the number of examples.

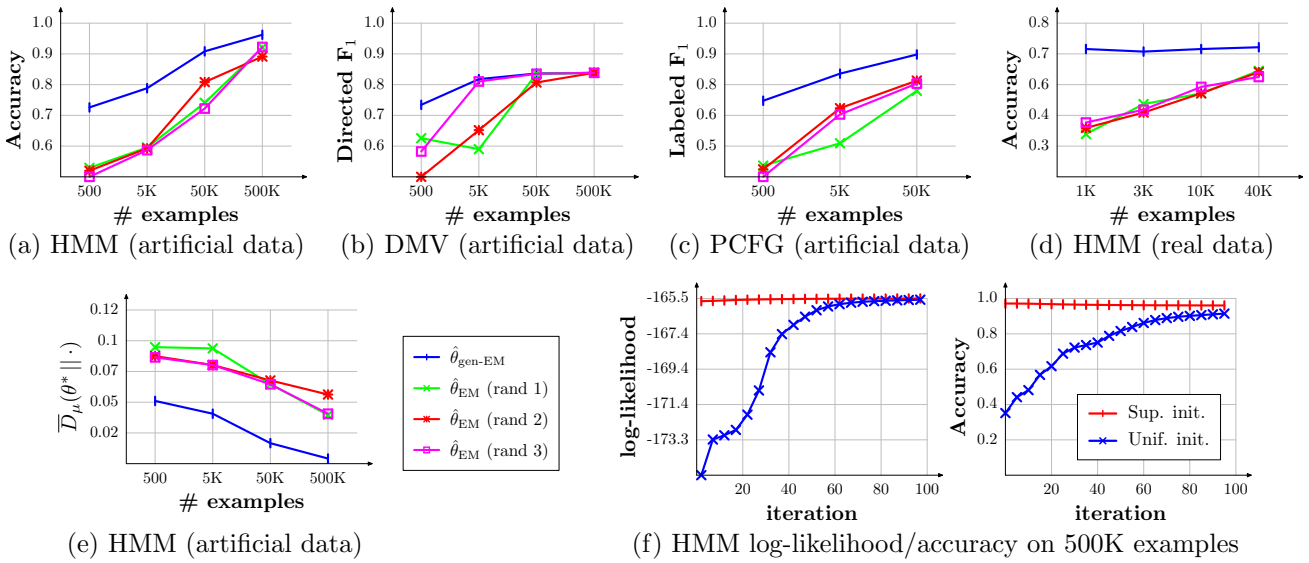
In the unsupervised case, we use the following procedure to obtain a surrogate for $\hat{\theta}$: initialize EM with the supervised estimate $\hat{\theta}_{\text{gen}}$ and run EM for 100 iterations. Let $\hat{\theta}_{\text{gen-EM}}$ denote the final parameters, which should be representative of $\hat{\theta}$. Table 1 shows that the estimation error of $\hat{\theta}_{\text{gen-EM}}$ is an order of magnitude higher than that of $\hat{\theta}_{\text{gen}}$, which is to expected since $\hat{\theta}_{\text{gen-EM}}$ does not have access to labeled data. However, this error can also be driven down given a moderate number of examples.

7 Optimization error

Finally, we study optimization error, which is the discrepancy between the global maximizer $\hat{\theta}$ and $\hat{\theta}_{\text{EM}}$, the result of running EM starting from a uniform initialization (plus some small noise). As before, we cannot compute $\hat{\theta}$, so we use $\hat{\theta}_{\text{gen-EM}}$ as a surrogate. Also, instead of comparing $\hat{\theta}_{\text{gen-EM}}$ and $\hat{\theta}$ with each other, we compare each of their discrepancies with respect to θ^*.

Let us first consider optimization error in terms of prediction error. The first observation is that there is a gap between the prediction accuracies of $\hat{\theta}_{\text{gen-EM}}$ and $\hat{\theta}_{\text{EM}}$, but this gap shrinks considerably as we increase the number of examples. Figures 4(a,b,c) support this for all three model families: for the HMM, both $\hat{\theta}_{\text{gen-EM}}$ and $\hat{\theta}_{\text{EM}}$ eventually achieve around 90% accuracy; for the DMV, 85%. For the PCFG, $\hat{\theta}_{\text{EM}}$ still lags $\hat{\theta}_{\text{gen-EM}}$ by 10%, but we believe that more data can further reduce this gap.

Figure 4(d) shows that these trends are not particular to artificial data. On real WSJ data, the gap



(a) HMM (artificial data) (b) DMV (artificial data) (c) PCFG (artificial data) (d) HMM (real data)

(e) HMM (artificial data) (f) HMM log-likelihood/accuracy on 500K examples

Figure 4: Compares the performance of $\hat{\theta}_{\text{EM}}$ (EM with a uniform initialization) against $\hat{\theta}_{\text{gen-EM}}$ (EM initialized with the supervised estimate) on (a–c) various models, (d) real data. (e) measures distance instead of accuracy and (f) shows a sample EM run.

between $\hat{\theta}_{\text{gen-EM}}$ and $\hat{\theta}_{\text{EM}}$ also diminishes for the HMM. To reaffirm the trends, we also measure distance D_μ. Figure 4(e) shows that the distance from $\hat{\theta}_{\text{EM}}$ to the true parameters θ^* decreases, but the gap between $\hat{\theta}_{\text{gen-EM}}$ and $\hat{\theta}_{\text{EM}}$ does not close as decisively as it did for prediction error.

It is quite surprising that by simply running EM with a neutral initialization, we can accurately learn a complex model with thousands of parameters. Figures 4(f,g) show how both likelihood and accuracy, which both start quite low, improve substantially over time for the HMM on artificial data.

Carroll and Charniak (1992) report that EM fared poorly with local optima. We do not claim that there are no local optima, but only that the likelihood surface that EM is optimizing can become smoother with more examples. With more examples, there is less noise in the aggregate statistics, so it might be easier for EM to pick out the salient patterns.

Srebro et al. (2006) made a similar observation in the context of learning Gaussian mixtures. They characterized three regimes: one where EM was successful in recovering the true clusters (given lots of data), another where EM failed but the global optimum was successful, and the last where both failed (without much data).

There is also a rich body of theoretical work on learning latent-variable models. Specialized algorithms can provably learn certain constrained discrete hidden-variable models, some in terms of weak generative capacity (Ron et al., 1998; Clark and Thollard, 2005; Adriaans, 1999), others in term of strong generative capacity (Dasgupta, 1999; Feldman et al., 2005). But with the exception of Dasgupta and Schulman (2007), there is little theoretical understanding of EM, let alone on complex model families such as the HMM, PCFG, and DMV.

8 Conclusion

In recent years, many methods have improved unsupervised induction, but these methods must still deal with the four types of errors we have identified in this paper. One of our main contributions of this paper is the idea of using the meta-model to diagnose the approximation error. Using this tool, we can better understand model biases and hopefully correct for them. We also introduced a method for measuring distances in face of label symmetry and ran experiments exploring the effectiveness of EM as a function of the amount of data. Finally, we hope that setting up the general framework to understand the errors of unsupervised induction systems will aid the development of better methods and further analyses.

References

P. W. Adriaans. 1999. Learning shallow context-free languages under simple distributions. Technical report, Stanford University.

G. Carroll and E. Charniak. 1992. Two experiments on learning probabilistic dependency grammars from corpora. In *Workshop Notes for Statistically-Based NLP Techniques*, pages 1–13.

A. Clark and F. Thollard. 2005. PAC-learnability of probabilistic deterministic finite state automata. *JMLR*, 5:473–497.

A. Clark. 2001. Unsupervised induction of stochastic context free grammars with distributional clustering. In *CoNLL*.

S. Dasgupta and L. Schulman. 2007. A probabilistic analysis of EM for mixtures of separated, spherical Gaussians. *JMLR*, 8.

S. Dasgupta. 1999. Learning mixtures of Gaussians. In *FOCS*.

J. Feldman, R. O'Donnell, and R. A. Servedio. 2005. Learning mixtures of product distributions over discrete domains. In *FOCS*, pages 501–510.

S. Goldwater and T. Griffiths. 2007. A fully Bayesian approach to unsupervised part-of-speech tagging. In *ACL*.

T. Grenager, D. Klein, and C. D. Manning. 2005. Unsupervised learning of field segmentation models for information extraction. In *ACL*.

A. Haghighi and D. Klein. 2006. Prototype-based grammar induction. In *ACL*.

M. Johnson. 2007. Why doesn't EM find good HMM POS-taggers? In *EMNLP/CoNLL*.

D. Klein and C. D. Manning. 2004. Corpus-based induction of syntactic structure: Models of dependency and constituency. In *ACL*.

H. W. Kuhn. 1955. The Hungarian method for the assignment problem. *Naval Research Logistic Quarterly*, 2:83–97.

K. Kurihara and T. Sato. 2006. Variational Bayesian grammar induction for natural language. In *International Colloquium on Grammatical Inference*.

B. Merialdo. 1994. Tagging English text with a probabilistic model. *Computational Linguistics*, 20:155–171.

F. Pereira and Y. Shabes. 1992. Inside-outside reestimation from partially bracketed corpora. In *ACL*.

D. Ron, Y. Singer, and N. Tishby. 1998. On the learnability and usage of acyclic probabilistic finite automata. *Journal of Computer and System Sciences*, 56:133–152.

N. Smith and J. Eisner. 2005. Contrastive estimation: Training log-linear models on unlabeled data. In *ACL*.

N. Smith and J. Eisner. 2006. Annealing structural bias in multilingual weighted grammar induction. In *ACL*.

N. Srebro, G. Shakhnarovich, and S. Roweis. 2006. An investigation of computational and informational limits in Gaussian mixture clustering. In *ICML*, pages 865–872.