Statistical NLP
Winter 2008

Lecture 16: Unsupervised Learning I

Roger Levy

[thanks to Sharon Goldwater for many slides]
Supervised training

- Standard statistical systems use a supervised paradigm.

Training:

- Labeled training data → Statistics → Machine learning system → Prediction procedure
The real story

- Annotating labeled data is \textit{labor-intensive}!!!
The real story (II)

• This also means that moving to a new language, domain, or even genre can be difficult.
• But unlabeled data is cheap!
• It would be nice to use the unlabeled data directly to learn the labelings you want in your model.
• Today we’ll look at methods for doing exactly this.
Today’s plan

• We’ll illustrate unsupervised learning with the “laboratory” task of part-of-speech tagging
• We’ll start with MLE-based methods
• Then we’ll look at problems with MLE-based methods
• This will lead us to Bayesian methods for unsupervised learning
• We’ll look at two different ways to do Bayesian model learning in this case.
Learning structured models

- Most of the models we’ve looked at in this class have been **structured**
  - Tagging
  - Parsing
  - Role labeling
  - Coreference
- The structure is **latent**
- With raw data, we have to construct models that will be **rewarded for inferring that latent structure**
A very simple example

- Suppose that we observe the following counts:
  
<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>9</td>
</tr>
<tr>
<td>B</td>
<td>9</td>
</tr>
<tr>
<td>C</td>
<td>1</td>
</tr>
<tr>
<td>D</td>
<td>1</td>
</tr>
</tbody>
</table>

- Suppose we are told that these counts arose from tossing two coins, each with a different label on each side.

- Suppose further that we are told that the coins are not extremely unfair.

- There is an intuitive solution; how can we learn it?
Suppose we fully parameterize the model:

\[ \pi : \text{Probability of flipping coin 1} \]
\[ p_1 : \text{Probability of coin 1 coming up “heads”} \]
\[ p_2 : \text{Probability of coin 2 coming up “heads”} \]

The MLE of this solution is totally degenerate: it cannot distinguish which letters should be paired on a coin

• Convince yourself of this!

We need to specify more constraints on the model

• The general idea would be to place priors on the model parameters

• An extreme variant: force \( p_1 = p_2 = 0.5 \)
A very simple example (III)

- An extreme variant: force $p_1=p_2=0.5$
- This *forces structure into the model*
- It also makes it easy to visualize the log-likelihood as a function of the remaining free parameter
- The intuitive solution is found!
The EM algorithm

• In the two-coin example, we were able to explore the likelihood surface exhaustively:
  • Enumerating all possible model structures
  • Analytically deriving the MLE for each model structure
  • Picking the model structure with best MLE
• In general, however, latent structure often makes direct analysis of the likelihood surface intractable or impossible
• [mixture of Gaussians example?]
The EM algorithm

• In cases of an unanalyzable likelihood function, we want to use *hill-climbing* techniques to find good points on the likelihood surface

• Some of these fall under the category of iterative numerical optimization

• In our case, we’ll look at a general-purpose tool that is guaranteed “not to do bad things”: the Expectation-Maximization (EM) algorithm
We’ve already seen examples of using dynamic programming via a trellis for inference in HMMs:

\[
\begin{align*}
\delta_i(s) &= \max_{s_{i-1}} P(s_i | s_{i-1}) P(w_{i-1} | s_{i-1}) \delta_{i-1}(s_{i-1}) \\
\psi_i(s) &= \arg \max_{s_{i-1}} P(s_i | s_{i-1}) P(w_{i-1} | s_{i-1}) \delta_{i-1}(s_{i-1})
\end{align*}
\]
Category learning: EM for HMMs

- You want to estimate the parameters $\theta$
  - There are statistics you’d need to do this supervised
  - For HMMs, the # transitions & emissions of each type
- Suppose you have a starting estimate of $\theta$
- $E$: calculate the expectations over your statistics
  - Expected # of transitions between each state pair
  - Expected # of emissions from each state to each word
- $M$: re-estimate $\theta$ based on your expected statistics

$$\hat{\pi} = \text{expected frequency in state } i \text{ at time } t = 1$$

$$\hat{a}_{i,j} = \frac{\text{expected number of transitions from state } i \text{ to } j}{\text{expected number of transitions from state } i}$$

$$\hat{b}_{i,j,k} = \frac{\text{expected number of transitions from } i \text{ to } j \text{ with } k \text{ observed}}{\text{expected number of transitions from } i \text{ to } j}$$
EM for HMMs: example (M&S 1999)

- We have a crazy soft drink machine with two states
- We get the sequence <lemonade, iced tea, cola>
- Start with the parameters

```
Original
Π  CP  1.0
  IP  0.0

A CP  IP
CP  0.7  0.3
IP  0.5  0.5

B CP  ice_t  lem
cola  0.6  0.1  0.3
ice_t  0.1  0.7  0.2

Re-estimate!
```
Adding a Bayesian Prior

For model \((w, t, \theta)\), try to find the optimal value for \(\theta\) using Bayes’ rule:

\[
P(\theta \mid w) \propto P(w \mid \theta)P(\theta)
\]

- posterior \hspace{1cm} likelihood \hspace{1cm} prior

Two standard objective functions are

- Maximum-likelihood estimation (MLE):
  \[
  \theta^* = \operatorname{argmax}_\theta P(w \mid \theta)
  \]

- Maximum a posteriori (MAP) estimation:
  \[
  \theta^* = \operatorname{argmax}_\theta P(w \mid \theta)P(\theta)
  \]
Dirichlet priors

- For multinomial distributions, the Dirichlet makes a natural prior.

A symmetric Dirichlet(\(\beta\)) prior over \(\theta = (\theta_1, \theta_2)\):

- \(\beta > 1\): prefer uniform distributions
- \(\beta = 1\): no preference
- \(\beta < 1\): prefer sparse (skewed) distributions
MAP estimation with EM

- We have already seen how to do ML estimation with the Expectation-Maximization Algorithm.
- We can also do MAP estimation with the appropriate type of prior.
- MAP estimation affects the M-step of EM.

  - For example, with a Dirichlet prior, the MAP estimate can be calculated by treating the prior parameters as “pseudo-counts.”

\[
A : \ a_{j,j'} \leftarrow \frac{(u_{j'}^{(A)} - 1) + \sum_{t=2}^{T} \langle s_{t-1,j} s_{t,j'} \rangle}{\sum_{j'=1}^{k} (u_{j'}^{(A)} - 1) + \sum_{t=2}^{T} \langle s_{t-1,j} \rangle} \]  

(Beal 2003)
More than just the MAP

- Why do we want to estimate $\theta$?
  - Prediction: estimate $P(w_{n+1}|\theta)$.
  - Structure recovery: estimate $P(t|\theta, w)$.

- To the true Bayesian, the model $\theta$ parameters should really be marginalized out:
  - Prediction: estimate $P(w_{n+1}|w) = \int P(w_{n+1}|\theta)P(\theta|w)d\theta$
  - Structure: estimate $P(t|w) = \int P(t|\theta, w)P(\theta|w)d\theta$

- We don’t want to choose model parameters if we can avoid it
Bayesian integration

• When we integrate over the parameters $\theta$, we gain
  • Robustness: values of hidden variables will have high probability over a range of $\theta$.
  • Flexibility: allows wider choice of priors, including priors favoring sparse solutions.
Integration example

Suppose we want to estimate $t \in \{0, 1\}$ where

- $P(\theta|w)$ is broad:

- $P(t = 1|\theta, w)$ is peaked:

Estimating $t$ based on fixed $\theta^*$ favors $t = 1$, but for many probable values of $\theta$, $t = 0$ is a better choice.
Sparse distributions

In language learning, sparse distributions are often preferable (e.g., HMM transition distributions).

• Problem: when $\beta < 1$, setting any $\theta_k = 0$ makes $P(\theta) \rightarrow \infty$ regardless of other $\theta_j$.

• Solution: instead of fixing $\theta$, integrate:

\[
P(w_{n+1} = w | w, \beta) = \int P(w | \theta)P(\theta | w, \beta) \, d\theta
\]

\[
= \frac{n_w + \beta}{n + W \beta}
\]
Integrating out $\theta$ in HMMs

- We want to integrate:

$$P(w_{n+1} = w \mid w, \beta) = \int P(w \mid \theta) P(\theta \mid w, \beta) \, d\theta$$

$$= \frac{n_w + \beta}{n + W\beta}$$

- Problem: this is intractable
- Solution: we can approximate the integral using sampling techniques.
Structure of the Bayesian HMM

- **Hyperparameters** $\alpha, \beta$ determine the model parameters $\tau, \omega$, and these influence the generation of structure.
The precise problem

- Unsupervised learning:
  - We know the hyperparameters* and the observations
  - We don’t really care about the parameters $\tau, \omega$
  - We want to infer the conditional distr on the labels!
Suppose that we knew all the latent structure but for one tag

We could then calculate the posterior distribution over this tag:

$$P(t_i | t_{-i}, w, \alpha, \beta)$$
Posterior inference w/ Gibbs Sampling

\[ P(t_i | t_{-i}, \mathbf{w}, \alpha, \beta) \]

- Really, even if we knew all but one label, we wouldn’t know the parameters \( \tau, \omega \).
- That turns out to be OK: we can integrate over them.

\[
P(t_i | t_{-i}, \mathbf{w}, \alpha, \beta) \propto \frac{n(t_i, w_i) + \beta}{n_{t_i} + W_{t_i} \beta} \cdot \frac{n(t_{i-2}, t_{i-1}, t_i) + \alpha}{n(t_{i-2}, t_{i-1}) + T\alpha} \cdot \frac{n(t_{i-1}, t_i, t_{i+1}) + I(t_{i-2} = t_i-1 = t_i = t_{i+1}) + \alpha}{n(t_{i-1}, t_i) + I(t_{i-2} = t_i-1 = t_i) + T\alpha} \cdot \frac{n(t_i, t_{i+1}, t_{i+2}) + I(t_{i-2} = t_i = t_{i+2}, t_{i-1} = t_{i+1}) + I(t_{i-1} = t_i = t_{i+1} = t_{i+2}) + \alpha}{n(t_{i-1}, t_i, t_{i+1}) + I(t_{i-2} = t_i, t_{i-1} = t_{i+1}) + I(t_{i-1} = t_i = t_{i+1}) + T\alpha}
\]
The theory of *Markov Chain Monte Carlo* sampling says that if we do this type of resampling for a long time, we will converge to the true posterior distribution over labels:

- Initialize the tag sequence however you want
- Iterate through the sequence many times, each time sampling over

\[ P(t_i \mid t_{-i}, w, \alpha, \beta) \]
Experiments of Goldwater & Griffiths 2006

- Vary $\alpha$, $\beta$ using standard “unsupervised” POS tagging methodology:
  - Tag dictionary lists possible tags for each word (based on ~1m words of Wall Street Journal corpus).
  - Train and test on unlabeled corpus (24,000 words of WSJ).
    - 53.6% of word tokens have multiple possible tags.
    - Average number of tags per token = 2.3.
- Compare tagging accuracy to other methods.
  - HMM with maximum-likelihood estimation using EM (MLHMM).
  - Conditional Random Field with contrastive estimation (CRF/CE) (Smith & Eisner, 2005).
Results

- Transition hyperparameter $\alpha$ has more effect than output hyperparameter $\beta$.
  - Smaller $\alpha$ enforces sparse transition matrix, improves scores.
  - Less effect of $\beta$ due to more varying output distributions?
- Even uniform priors outperform MLHMM (due to integration).

<table>
<thead>
<tr>
<th>Model</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>MLHMM</td>
<td>74.7</td>
</tr>
<tr>
<td>BHMM ($\alpha = 1, \beta = 1$)</td>
<td>83.9</td>
</tr>
<tr>
<td>BHMM (best: $\alpha = 0.003, \beta = 1$)</td>
<td>86.8</td>
</tr>
<tr>
<td>CRF/CE (best)</td>
<td>90.1</td>
</tr>
</tbody>
</table>
Hyperparameter inference

- Selecting hyperparameters based on performance is problematic.
  - Violates unsupervised assumption.
  - Time-consuming.
- Bayesian framework allows us to infer values automatically.
  - Add uniform priors over the hyperparameters.
  - Resample each hyperparameter after each Gibbs iteration.
- Results: slightly worse than oracle (84.4% vs. 86.8%), but still well above MLHMM (74.7%).
Reducing lexical resources

Experiments inspired by Smith & Eisner (2005):

• Collapse 45 treebank tags onto smaller set of 17.

• Create several dictionaries of varying quality.
  • Words appearing at least \(d\) times in 24k training corpus are listed in dictionary (\(d = 1, 2, 3, 5, 10, \infty\)).
  • Words appearing fewer than \(d\) times can belong to any class.

• Since standard accuracy measure requires labeled classes, we measure using best many-to-one matching of classes.
Results

- BHMM outperforms MLHMM for all dictionary levels, more so with smaller dictionaries:

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>5</th>
<th>10</th>
<th>∞</th>
</tr>
</thead>
<tbody>
<tr>
<td>MLHMM</td>
<td>90.6</td>
<td>78.2</td>
<td>74.7</td>
<td>70.5</td>
<td>65.4</td>
<td>34.7</td>
</tr>
<tr>
<td>BHMM</td>
<td>91.7</td>
<td>83.7</td>
<td>80.0</td>
<td>77.1</td>
<td>72.8</td>
<td>63.3</td>
</tr>
</tbody>
</table>

- (results are using inference on hyperparameters).
Clustering results

- MLHMM groups tokens of the same lexical item together.
- BHMM clusters are more coherent, more variable in size. Errors are often sensible (e.g. separating common nouns/proper nouns, confusing determiners/adjectives, prepositions/participles).
Summary

• Using Bayesian techniques with a standard model dramatically improves unsupervised POS tagging.
  • Integration over parameters adds robustness to estimates of hidden variables.
  • Use of priors allows preference for sparse distributions typical of natural language.
  • Especially helpful when learning is less constrained.