Unsupervised learning as supervised structured prediction

Hal Daumé III
School of Computing
University of Utah
me@hal3.name
Structured Prediction 101

- Learn a function mapping inputs to complex outputs:

\[ f : X \rightarrow Y \]

- Structure prediction
  - Sequence Labeling
  - Parsing
  - Coreference Resolution
  - Machine Translation

- Input Space
- Decoding
- Output Space
Reducing Structured Prediction

Key Assumption: *Optimal Policy for training data*

Given: input, true output and state;
Return: best successor state

Weak!
Theoretical Analysis

**Theorem:** For conservative $\beta$, after $2T^3 \ln T$ iterations, the loss of the learned policy is bounded as follows:

$$L(h) \leq L(h_0) + 2T \ln T l_{\text{avg}} + (1 + \ln T) \frac{C_{\text{max}}}{T}$$

- Loss of the optimal policy
- Average multiclass classification loss
- Worst case per-step loss
Dependency Parsing

Nivre-style shift-reduce algorithm

root John hit the ball with the bat
Dependency Parsing

Start in state \((S,i,A) = (\{\}, 1, \{\})\) and end when \(i=N\)

One of four actions:

- **LeftA**: \((t:S, i, A)\) to \((S, i, (i,t):A)\)
  so long as no \((?,t)\) in \(A\)

- **RightA**: \((t:s, i, A)\) to \((i:t:S, i+1, (t,i):A)\)
  so long as no \((?,i)\) in \(A\)

- **Reduce**: \((t:S, i, A)\) to \((S, i, A)\) so long as \((?,t)\) in \(A\)

- **Shift**: \((S, i, A)\) to \((i:S, i+1, A)\)
Unsupervised prediction

Key idea (not new!): a tree is good if it enables us to do a good job re-predicting the input

e.g., we should be able to predict "with" given it's parent "hit" and dependent "bat"
Algorithm

- Initialize with random trees
- Repeat:
  - Train a classifier A to predict sentences given trees, optimized for some loss function L
  - Train a classifier B to predict trees given sentences, optimized for L(A)
  - Predict trees using B
- In other words, a tree has low loss if it aids in the reprediction of the input
Mini attempt at an analysis

➢ Under two key assumptions, $L(A)$ will decrease with each iteration

➢ Assumption 1:

➢ Assumption 2:
Minor Aside

- In sequence labeling problems, this algorithm yields exactly the forward backward algorithm when:
  - The loss is Hamming loss
  - Naïve Bayes classifiers are used
  - Features are just tag/tag and tag/word pairs
  - Searn losses are computed using dynamic programming

<table>
<thead>
<tr>
<th>Model</th>
<th>States</th>
<th>Truth</th>
<th>EM</th>
<th>Searn-NB</th>
<th>Searn-LR</th>
</tr>
</thead>
<tbody>
<tr>
<td>HMM1</td>
<td>$K = 2$</td>
<td>$0.227 \pm 0.107$</td>
<td>$0.275 \pm 0.128$</td>
<td>$0.287 \pm 0.138$</td>
<td>$0.276 \pm 0.095$</td>
</tr>
<tr>
<td>HMM1</td>
<td>$K = 5$</td>
<td>$0.687 \pm 0.043$</td>
<td>$0.678 \pm 0.026$</td>
<td>$0.688 \pm 0.025$</td>
<td>$0.672 \pm 0.022$</td>
</tr>
<tr>
<td>HMM1</td>
<td>$K = 10$</td>
<td>$0.806 \pm 0.035$</td>
<td>$0.762 \pm 0.021$</td>
<td>$0.771 \pm 0.019$</td>
<td>$0.755 \pm 0.019$</td>
</tr>
<tr>
<td>HMM2</td>
<td>$K = 2$</td>
<td>$0.294 \pm 0.072$</td>
<td>$0.396 \pm 0.057$</td>
<td>$0.408 \pm 0.056$</td>
<td>$0.271 \pm 0.057$</td>
</tr>
<tr>
<td>HMM2</td>
<td>$K = 5$</td>
<td>$0.651 \pm 0.068$</td>
<td>$0.695 \pm 0.027$</td>
<td>$0.710 \pm 0.016$</td>
<td>$0.633 \pm 0.018$</td>
</tr>
<tr>
<td>HMM2</td>
<td>$K = 10$</td>
<td>$0.815 \pm 0.032$</td>
<td>$0.764 \pm 0.021$</td>
<td>$0.771 \pm 0.015$</td>
<td>$0.705 \pm 0.019$</td>
</tr>
</tbody>
</table>
Back to Parsing

<table>
<thead>
<tr>
<th>Model</th>
<th>Accuracy-Train</th>
<th>Accuracy-Test</th>
<th>Iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random-Generative</td>
<td>0.235 ± 0.009</td>
<td>0.235 ± 0.013</td>
<td></td>
</tr>
<tr>
<td>Random-SEARN</td>
<td>0.213 ± 0.002</td>
<td>0.210 ± 0.006</td>
<td></td>
</tr>
<tr>
<td>Klein+Manning:Random-Init</td>
<td>0.236 ± 0.038</td>
<td>0.236 ± 0.043</td>
<td>63.3 ± 9.2</td>
</tr>
<tr>
<td>Klein+Manning:Smart-Init</td>
<td>0.352 ± 0.066</td>
<td>0.352 ± 0.060</td>
<td>64.1 ± 11.1</td>
</tr>
<tr>
<td>Smith+Eisner:Length</td>
<td>0.338 ± 0.036</td>
<td>0.337 ± 0.059</td>
<td>173.1 ± 77.7</td>
</tr>
<tr>
<td>Smith+Eisner:Trans1</td>
<td>0.488 ± 0.009</td>
<td>0.490 ± 0.015</td>
<td>173.4 ± 71.0</td>
</tr>
<tr>
<td>Smith+Eisner:DelOrTrans1</td>
<td>0.473 ± 0.060</td>
<td>0.471 ± 0.059</td>
<td>132.2 ± 29.9</td>
</tr>
<tr>
<td>SEARN:Unsupervised</td>
<td>0.438 ± 0.016</td>
<td>0.434 ± 0.022</td>
<td>27.6 ± 3.7</td>
</tr>
<tr>
<td>Smith+Eisner:Supervised</td>
<td>0.799 ± 0.002</td>
<td>0.786 ± 0.008</td>
<td>350.5 ± 54.4</td>
</tr>
<tr>
<td>SEARN:Supervised</td>
<td>0.800 ± 0.003</td>
<td>0.806 ± 0.004</td>
<td>24.4 ± 2.6</td>
</tr>
</tbody>
</table>

*root* John hit the ball with the bat
Discussion

- Proposed algorithm that naturally extends Searn to the unsupervised setting

- Key (non-novel) idea: predict the input

- Comes with (somewhat trivial) guarantees
  - Algorithmically resembles Viterbi EM, but will converge
  - Assumptions are, occasionally, verifiable

- Can train unsupervised sequence labelers with SVMs or DTs whatever classifier you want with whatever feature spaces you want

- Can do the same for any structured prediction problem

- Want: better understanding of why it works