Learning to Search Better Than Your Teacher
[Chang et al., 2015]

Leo Born

Hauptseminar: Imitation Learning
Artem Sokolov
Department of Computational Linguistics
Ruprecht-Karls-Universität Heidelberg

18.10.2018
1 Motivation

2 Locally Optimal Learning to Search (LOLS)

3 Performance in NLP Contexts
   - POS tagging
   - Dependency parsing
   - Timeline summarization

4 Conclusion
1 Motivation

2 Locally Optimal Learning to Search (LOLS)

3 Performance in NLP Contexts
   - POS tagging
   - Dependency parsing
   - Timeline summarization

4 Conclusion
Learning to Search (L2S) is nice when the reference policy is good.
Motivation

• Learning to Search (L2S) is nice when the reference policy is good
• But what happens when the reference policy is sub-optimal? Can’t we improve upon it?
Motivation

• Learning to Search (L2S) is nice when the reference policy is good
• But what happens when the reference policy is sub-optimal? Can’t we improve upon it?
  • “Yes, we can” [Obama, 2008]
Motivation

• Learning to Search (L2S) is nice when the reference policy is good
• But what happens when the reference policy is sub-optimal? Can’t we improve upon it?
  • “Yes, we can” [Obama, 2008]
• Locally Optimal L2S as one approach to be better than the teacher
Contents

1 Motivation

2 Locally Optimal Learning to Search (LOLS)

3 Performance in NLP Contexts
   - POS tagging
   - Dependency parsing
   - Timeline summarization

4 Conclusion
Learning to Search Better Than Your Teacher
[Chang et al., 2015]

- Roll-in with learned policy, mixture policy for roll-out
- If in one-step deviation there is a better solution than the expert’s, use this
- Guarantees locally optimal policy
- Outperforms SEARN [Daumé III et al., 2009] on POS tagging and dependency parsing tasks
- Useful for other NLP tasks as well
Algorithm 1 Locally Optimal Learning to Search (LOLS)

**Require:** Dataset \(\{x_i, y_i\}_{i=1}^N\) drawn from \(D\) and \(\beta \geq 0\): a mixture parameter for roll-out.

1: Initialize a policy \(\pi_0\).
2: **for all** \(i \in \{1, 2, \ldots, N\}\) (loop over each instance) **do**
3: Generate a reference policy \(\pi^{\text{ref}}\) based on \(y_i\).
4: Initialize \(\Gamma = \emptyset\).
5: **for all** \(t \in \{0, 1, 2, \ldots, T-1\}\) **do**
6: Roll-in by executing \(\pi_i^{\text{in}} = \hat{\pi}_i\) for \(t\) rounds and reach \(s_t\).
7: **for all** \(a \in A(s_t)\) **do**
8: Let \(\pi_i^{\text{out}} = \pi^{\text{ref}}\) with probability \(\beta\), otherwise \(\hat{\pi}_i\).
9: Evaluate cost \(c_{i,t}(a)\) by rolling-out with \(\pi_i^{\text{out}}\) for \(T - t - 1\) steps.
10: **end for**
11: Generate a feature vector \(\Phi(x_i, s_t)\).
12: Set \(\Gamma = \Gamma \cup \{\langle c_{i,t}, \Phi(x_i, s_t)\rangle\}\).
13: **end for**
14: \(\hat{\pi}_{i+1} \leftarrow\) Train(\(\hat{\pi}_i, \Gamma\)) (Update).
15: **end for**
16: Return the average policy across \(\hat{\pi}_0, \hat{\pi}_1, \ldots, \hat{\pi}_N\).
Analysis: What NOT to do

- Roll-in and roll-out with learned policy
- Roll-in with reference policy

**Figure:** Effect of different roll-in/roll-out strategies [Chang et al., 2015].
Analysis: What to do instead

- Roll-in with learned policy, roll-out with mixture of learned and reference policy
- “a policy is locally optimal if changing any one decision it makes never improves its performance”

\[
\delta_N = \frac{1}{NT} \sum_{i=1}^{N} \sum_{t=1}^{T} \mathbb{E}_{s \sim d_{\pi_i}^t} \left[ Q_{\pi_i}^{\text{out}} (s, \hat{\pi}_i) - \left( \beta \min_{\alpha} Q_{\pi_{\text{ref}}} (s, a) 
+ (1 - \beta) \min_{\alpha} Q_{\pi_i} (s, a) \right) \right]
\]

\[
\beta (J(\pi) - J(\pi_{\text{ref}})) + (1 - \beta) \sum_{t=1}^{T} (J(\hat{\pi}) - \min_{\pi \in \Pi} \mathbb{E}_{s \sim d_t} [Q_{\pi} (s, \pi)]) 
\leq T \delta_N.
\]
Analysis: What to do instead

- Minimizes a combination of regret to reference policy and regret to its own one-step deviations
- When reference is optimal, first term is non-negative; competes with one-step deviations in this case
- When reference is sub-optimal, first term can be negative -> learned policy has improved upon the reference policy
## Analysis

<table>
<thead>
<tr>
<th>roll-out →</th>
<th>Reference</th>
<th>Mixture</th>
<th>Learned</th>
</tr>
</thead>
<tbody>
<tr>
<td>↓ roll-in</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reference</td>
<td>Inconsistent</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Learned</td>
<td>Not locally opt.</td>
<td>Good</td>
<td>RL</td>
</tr>
</tbody>
</table>

**Figure:** Effect of different roll-in/roll-out strategies [Chang et al., 2015].
Contents

1 Motivation

2 Locally Optimal Learning to Search (LOLS)

3 Performance in NLP Contexts
   - POS tagging
   - Dependency parsing
   - Timeline summarization

4 Conclusion
POS tagging

- Train on 38k sentences, test on 11k from Penn Treebank
- Best SEARN performance: 94.88 accuracy
- Performance of LOLS:

<table>
<thead>
<tr>
<th>roll-out</th>
<th>Reference</th>
<th>Mixture</th>
<th>Learned</th>
</tr>
</thead>
<tbody>
<tr>
<td>roll-in</td>
<td><strong>Reference is optimal</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reference</td>
<td>94.12</td>
<td>94.10</td>
</tr>
<tr>
<td></td>
<td>Learned</td>
<td>94.13</td>
<td>94.10</td>
</tr>
</tbody>
</table>
Dependency parsing

- Shift-reduce parser with three actions
- Three reference policies
  - *Optimal*: “non-deterministic oracle”
    [Goldberg and Nivre, 2013]
  - *Sub-optimal*: action that leads to good end state when obvious (?), else arbitrary
  - *Bad*: Arbitrary action
- Performance of SEARN:
  - 84.0
  - 81.1
  - 63.4
Dependency parsing

- Performance of LOLS:

<table>
<thead>
<tr>
<th>roll-out →</th>
<th>Reference</th>
<th>Mixture</th>
<th>Learned</th>
</tr>
</thead>
<tbody>
<tr>
<td>↓ roll-in</td>
<td>Reference</td>
<td>Mixture</td>
<td>Learned</td>
</tr>
<tr>
<td>Reference</td>
<td>87.2</td>
<td>89.7</td>
<td>88.2</td>
</tr>
<tr>
<td>Learned</td>
<td>90.7</td>
<td>90.5</td>
<td>86.9</td>
</tr>
</tbody>
</table>

Reference is optimal

<table>
<thead>
<tr>
<th>Reference</th>
<th>Mixture</th>
<th>Learned</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reference</td>
<td>83.3</td>
<td>87.2</td>
</tr>
<tr>
<td>Learned</td>
<td>87.1</td>
<td>90.2</td>
</tr>
</tbody>
</table>

Reference is suboptimal

<table>
<thead>
<tr>
<th>Reference</th>
<th>Mixture</th>
<th>Learned</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reference</td>
<td>68.7</td>
<td>66.7</td>
</tr>
<tr>
<td>Learned</td>
<td>75.8</td>
<td>87.5</td>
</tr>
</tbody>
</table>

Reference is bad
Results

- Roll-in with reference is bad
- When reference is optimal, doing roll-outs with reference is a good idea
- When reference is sub-optimal or bad, mixture rollouts perform better
- LOLS also significantly outperforms SEARN on all tasks
Real-time web scale event summarization using sequential decision making [Kedzie et al., 2016]

- Implementation of a *streaming summarization system* on sentence level
- Uses LOLS
- Evaluation on TREC TS data with TREC metrics
- Best F1 score, particularly in combination with sentence similarity filter
Timeline summarization

- Input: query (“eruption of Eyjafjallajökull’”), category (“natural disaster”), sentence stream:
  - For every new sentence, decision whether to include it in summary or ignore it must be made
- Loss function is complement of Dice coefficient (F1 score)
- Greedy reference policy
- SGD classifier for updating $\tilde{\pi}$
Timeline summarization

- Data from TREC TS task [Aslam et al., 2013, Aslam et al., 2014]
  - Web-crawled between 10/2011 and 02/2013, 1.2 billion docs
  - Only considered news section: 7.6 million docs
  - 44 categorized events, 73.35 *nuggets* per event on average
- Stream size 100
- 5 events dev, 39 evaluated using leave-one-out
## Timeline summarization

<table>
<thead>
<tr>
<th>Method</th>
<th>exp. gain</th>
<th>comp.</th>
<th>$F_1$</th>
<th>exp. gain</th>
<th>comp.</th>
<th>$F_1$</th>
<th>num. updates</th>
</tr>
</thead>
<tbody>
<tr>
<td>APSAL</td>
<td>0.119$^c$</td>
<td>0.09</td>
<td>0.094</td>
<td>0.105</td>
<td>0.088</td>
<td>0.088</td>
<td>8.333</td>
</tr>
<tr>
<td>Cos</td>
<td>0.075</td>
<td>0.176$^s$</td>
<td>0.099</td>
<td>0.095</td>
<td>0.236$^s$</td>
<td>0.128$^s$</td>
<td>145.615$^{s,f}$</td>
</tr>
<tr>
<td>Ls</td>
<td>0.097</td>
<td><strong>0.207$^{s,f}$</strong></td>
<td>0.112</td>
<td>0.136$^c$</td>
<td><strong>0.306$^{s,c,f}$</strong></td>
<td>0.162$^s$</td>
<td>89.872$^{s,f}$</td>
</tr>
<tr>
<td>LsCos</td>
<td>0.115$^{c,l}$</td>
<td>0.189$^s$</td>
<td><strong>0.127$^{s,c,l}$</strong></td>
<td><strong>0.162$^{s,c,l}$</strong></td>
<td>0.276$^s$</td>
<td><strong>0.184$^{s,c,l}$</strong></td>
<td>29.231$^{s,c}$</td>
</tr>
</tbody>
</table>

**Figure:** Average performance and number of updates (i.e. summary length) [Kedzie et al., 2016].
Contents

1 Motivation

2 Locally Optimal Learning to Search (LOLS)

3 Performance in NLP Contexts
   - POS tagging
   - Dependency parsing
   - Timeline summarization

4 Conclusion
Conclusion

- **SEARN**: batch learning, **LOLS**: online learning
- During training, **LOLS** needs just last policy for roll-in
- **Distinguishing feature** is mixture roll-out policy
- Even when the reference is bad, **LOLS** improves upon it
Conclusion

- **SEARN**: batch learning, **LOLS**: online learning
- During training, **LOLS** needs just last policy for roll-in
- **Distinguishing feature** is mixture roll-out policy
- Even when the reference is bad, **LOLS** improves upon it
  - Or does it? Cf. [Sharaf and Daumé III, 2017]
Bibliography I

TREC 2014 Temporal Summarization Track Overview.

TREC 2013 Temporal Summarization.

Learning to Search Better Than Your Teacher.

Search-based structured prediction.
Machine Learning, 75(3):297–325.
Bibliography II

