today:
  *Learning to search
  *SEARN
  *OPEN

Learning to search (L2S)

SEARN Hal Daume's Ph.D. thesis

\[ h_w : X \rightarrow Y \]

\[ h_w (x) = \arg \max_y s(x, y; w) \]

special case of SEARN: learning to do greedy search

split \( Y \) in ordered list of decisions

\[ (y_1, \ldots, y_T) \]

learn a classifier \( \pi_0 \) (feature \( (y_1, \ldots, y_{t-1}, x) \)) = \( y_t \)

classifier "decoding policy"

L2S framework

\[
\text{from } (x^{(i)}, y^{(i)})_{i=1}^n \text{ and } h^*(\cdot, \cdot) \\
\text{learn a good classifier/policy } \pi_w \text{ s.t. } \hat{y}_w = \pi_w (\Phi(y_1, \ldots, y_{t-1}, x)) \\
h_w (x) \triangleq (\hat{y}_1, \ldots, \hat{y}_T) \quad \text{"greedy decoding"}
\]

s.t. \( l(y^{(i)}, h_w(x^{(i)})) \) is good

\[ \text{central idea: "reduction" where reduces structured prediction learning} \]

\[ \text{problem of cost-sensitive classification learning for } \pi_w \]

methods generate training data for classifier \( \pi_w \)

\[ (\hat{y}_w^{(p)}, x^{(i)}, \text{cost}(y^{(i)})) \]

prefer sequence
"roll-in" policy → determines how \( f_{i:t} \) to get\( f_{t+1} \)

"roll-out" policy → "target completion" to get cost from roll-in possible (goal)

Using roll-out policy

Cost of roll-out \( c(f_{1:t}, f_{t+1}, y_{t+1}) \)

Reference policy: Ideally, \( R_{opt}(y_{1:t}, y_{t+1}) \)

\[ = \arg \min_{f_{1:t}} \{ l(y_{1:t}, f_{1:t}, y_{t+1}, y_{opt}) \} \]

Intractable (NP hard) in general

In practice: Use heuristic to approximate it

But sometimes can compute exactly

E.g. \( R_{opt} \): Hamming loss \( \L_1 \) is just copy ground-truth

LOLS → "locally optimal learning to search"

ICML 2015 "Las better than your teacher"

<table>
<thead>
<tr>
<th>roll-in</th>
<th>roll out</th>
<th>(approximate reference)</th>
<th>mature ( \frac{1}{2} ) Ref ( \frac{1}{2} ) Time</th>
<th>learned ( \frac{1}{2} ) Time</th>
<th>inconsistent</th>
</tr>
</thead>
<tbody>
<tr>
<td>learned (use ( \lambda_{ref} ))</td>
<td>consistent</td>
<td>consistent ( \Rightarrow ) not locally optimal</td>
<td>( \Rightarrow ) refinement learning</td>
<td>(not using enough information)</td>
<td>with ground truth</td>
</tr>
</tbody>
</table>

Example of approximate \( R_{opt} \): \( \lambda \) is bleu score

(On machine translation)

Consider all possible suffixes and pick
Kushal Arora: How (not) to Train your Generative Model: Scheduled Sampling, Likelihood, Adversary? by Huszar answers this question why SS might not work well. This is what I observed as well.

\[
\text{SEARNN} \quad \text{vs apply L2S to RNN training}
\]

\[
\text{[Lehnd, d. TICL 2016]} \quad \text{i.e. } \mathcal{P}(y_{t+1} | y_{t-1}, x) \sim \text{RNN cell}
\]

\[
p(y_{t+1} | y_{t-1}, x) \quad \text{of a RNN}
\]

\[
\text{if use } -\log p(y_{\text{target}} | y_{t-1}, x) \quad \text{as cost surrogate}
\]

\[
\text{and } y_{\text{target}} = \text{ground truth}
\]

\[
\text{then L2S with ref roll-in is standard MLE}
\]

3. Cost-sensitive surrogate loss choices:

   a) Structural SVM: \[\max \{\Delta(y') + s(y') \Delta(y_{\text{target}}) \}
   \]

   \[\hat{y}' \quad \Delta(y') - \Delta(y_{\text{target}}) \]

   \[y_{\text{target}} \in \text{argmin} \Delta(y') \]

   b) "Target log-loss": \[-\log p(y_{\text{target}} | y_{t-1}, x) \]

   \[L_0 \text{ differences with MLE:} \]

   \* address exposure bias using learned roll-in

   \* make use of structural loss \(L(\ldots)\) to predict \(y_{\text{target}} \) vs. MLE

Kushal Arora: How (not) to Train your Generative Model: Scheduled Sampling, Likelihood, Adversary? by Huszar answers this question why SS might not work well. This is what I observed as well.

\[
\text{Structured prediction energy networks (SPEaNs)}
\]

\[
\text{ICML 2016}
\]

\[
E(x, y, w) \quad (-s(x, y, w))
\]

\[
\text{relax } y \in \mathbb{R}^{d_T} \rightarrow \Delta[0,1] \]

\[
h_w(x) = \text{a few steps of projected GD on } E(x, y, w) \quad \text{/appropriate prediction}
\]
\[ h_w(x) = \text{a few steps of projected O.D. on } E(x, y; w) / \text{approximation of prediction procedure.} \]

2016 paper:

SSVM loss → "subgradient" method on \( w \)

\[
\text{huge loss: } \max_{y \in \{0,1\}} (l(y, w) - E(x,y;w)) + E(x,y;w)
\]

\[\text{approximate "subgradient" is } -\nabla_w \tilde{E}(x,y,w) + \nabla_w E(x,y;w)\]

\( y \) is approx max,

eg, see Clarke subdifferential

2017 paper: end-to-end learning:

\[ h_w(x) = y_0 - \sum_{t=1}^{T} M_t \frac{\partial}{\partial y} E(x,y_t;w) \]

"unrolled composition"