Learning to search (L2S)

**L2S framework**

From \((x^{(i)}, y^{(i)})_{i=1}^{n} \) and \(l^{(i)} \):

Learn a good classifier/policy \(\gamma_{\omega}\) s.t.

\[
\hat{y}_{t} = \gamma_{\omega}(y_{1}, \ldots, y_{t-1}; x)
\]

s.t. \(l(y_{i}^{(i)}; h_{\omega}(z^{(i)})) \) is small

Central idea: "reduction" that reduces the structured prediction problem to sequence of cost-sensitive classification learning of \(\gamma_{\omega}\)

Method: generate training data for classifier \(\gamma_{\omega}\)

\((\hat{y}_{1}^{(i)}, x^{(i)}), (\text{cost}(y_{1}^{(i)}))\)

"Roll-in" policy - determines how \(\hat{y}_{t-1}^{(i)} = \text{context}\)
"roll-out" policy $\rightarrow \tilde{y}_1^*, \ldots, \tilde{y}_T^*$ "target evaluation" to get $\text{cost} \left( l(y_l) \right)$ from return

Values for $T_w$

"reference policy", ideally, $\Pi_{\text{ref}} \left( \tilde{y}_1, \ldots, \tilde{y}_T, Y_{\text{ref}} \right)$

$\Rightarrow \text{argmin} \ l(y_l), \ (\tilde{y}_1, \ldots, \tilde{y}_T, Y_{\text{ref}}), \ Y_{\text{sample}}$

intractable (NP hard) in general

in practice: use heuristics to approximate $\Pi_{\text{ref}}$

but sometimes, can compute exactly

E.g. $\Pi_{\text{ref}}$ for Hamming loss: is the copy of ground truth

ICML 2015 "L2S better than your teacher"

LOLS $\rightarrow$ "locally optimal learning to search"

<table>
<thead>
<tr>
<th>$l(y_l)$</th>
<th>$T_w$</th>
<th>$\text{cost} \left( l(y_l) \right)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\rightarrow$</td>
<td>$\left{ \text{consistent}, \ \text{I.C.} \right}$</td>
<td>$\left{ \text{consistent}, \ \text{I.C.} \right}$</td>
</tr>
<tr>
<td>$\rightarrow$</td>
<td>$\left{ \text{not locally optimal}, \ \text{and locally optimal} \right}$</td>
<td>$\left{ \text{not locally optimal}, \ \text{and locally optimal} \right}$</td>
</tr>
</tbody>
</table>

"LOLS does well on cost-sensitive loss but $\Pi_{\text{ref}}$ does poorly on structured prediction"

(Not using exactly informative e.g. ground truth vs.

Example of approximate $\Pi_{\text{ref}}$: $l$ is bleu score

in machine translation"
15h30

**SEARCHN**: apply LAS to RNN training

\[ L \log p_Y (y_{target} \mid y_{1:t-1}, x) \rightarrow \text{RNN cell} \]

\[ P(y_{target} \mid y_{1:t-1}, x) \quad \text{of a (decoder) RNN} \]

If you use \(-\log p_y (y_{target} \mid y_{1:t-1}, x)\) as a cost surrogate

and \(y_{target} = \text{ground truth}\)

then LAS with Tref roll-in is standard MLE

- **Cost-sensitive surrogate losses classes**
  a) **structured SVM**: \( \max \left[ \Delta(h(\hat{y})) + SL(h(\hat{y})) \right] - s(y_{target}) \)

  \[ \hat{y} \leq \arg\min \{ y \} \quad c(h(\hat{y})) - c(h(y_{target})) \]

  \(y_{target} = \arg\min \{ y \} \)

  b) "target log-loss": \(-\log p_y (y_{target} \mid y_{1:t-1}, x)\)

- **differences with MLE**: 
  - address exposure bias using \(h_{W,z}(x)\)
  - make use of structured loss \(L(-,-)\) to predict \(y_{target}\) vs. \(y\) ground truth for MLE

**Structured prediction energy networks (SPENs)**

ICML 2016

\[ E(x, y; w) = -S(x, y; w) \]

- relax \(y \in \mathbb{R}^T \rightarrow \mathbb{R}^{11T} \)

\[ h_{W,z}(x) = \text{a few steps of projected GD on } E(p, y; w) \text{ w.r.t. } y \]

(approximate prediction procedure)

+ rounding

in 2016 paper: SSVM loss \(\rightarrow \) "subgradient" method on \(w\)
**Hinge Loss**: \[
\max_{y \in \{0,1\} \setminus \{y_i\}} \left( l(y^{(i)}, \hat{y}_i) - E(x^{(i)}, \hat{y}_i; w) \right) + E(x^{(i)}, y^{(i)}; w)
\]

- Requires extending \( y \) to fractional \( y \)'s

- Approximate "subgradient" is

\[
\frac{\nabla_w E(x^{(i)}, \hat{y}_i; w_t) + \nabla_w E(x^{(i)}, y^{(i)}; w)}{2}
\]

because use \( \hat{y}_i \) as approximate \( y \).

- e.g. see Clarke subdifferential for generalization on non-convex f.

2017 paper: end-to-end learning

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

"unrolled optimization"

\[
y_1 = y_0 - \eta_1 \frac{\partial}{\partial y} E(x, y_1; w)
\]

\[
y_2 = y_1 - \eta_2 \frac{\partial}{\partial y} E(x, y_2; w)
\]

\vdots

Recursively defined