Learning to search (L2S)

SEARN Hal Daumé's PhD thesis

\[ h_w : X \rightarrow Y \]

\[ h_w(x) = \arg \max_{y \in Y} s(x, y, w) \quad \text{[parameterized search]} \]

Special case of SEARN: learning to do greedy search.

Split \( Y \) in ordered list of decisions

\((y_1, y_2, \ldots, y_T)\)

Learn a classifier \( \hat{T}_w \) (feature \((\hat{y}_1, \ldots, \hat{y}_T; x)\)) = \( \hat{y}_T \)

L2S framework

\[ \text{from } (x^{(i)}, y^{(i)}) \quad \text{and } l(\cdot, \cdot) \]

Learn a good classifier/policy \( \Pi_w \) s.t.

\[ \hat{y}_T = \hat{T}_w (y_1, \ldots, y_{T-1}; x) \]

\[ h_w(x) = (\hat{y}_1, \ldots, \hat{y}_T) \quad \text{"greedy decoder"} \]

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

\[ \text{central idea: } \text{"reduction" where reduces structured pred. learning problem to problem of cost-sensitive classification learning of } \hat{T}_w \]

method: generate training data for classifier \( \hat{T}_w \)

\[ (\hat{y}_1, x^{(i)}, \text{cost}(y^{(i)}; x^{(i)})) \]

preprocess sequence to make next prediction

"roll in" policy \( \rightarrow \) determines how \( \hat{y}_{T-1} \) contact

"roll-out" policy \( \rightarrow \) \( \hat{y}_1, \ldots, \hat{y}_T \) "longest computation" to get cost

\[ \text{\( y_1 \leadsto \hat{a}_1, \hat{a}_2, \ldots, \hat{a}_T \rightarrow \text{cost; all } y^{(i)}(\hat{a}_1, \ldots, \hat{a}_T) \)
"reference policy", ideally, $T_{ref}((\hat{y}^1, \ldots, \hat{y}^{n-1}, y_{next}))$ is approximated as $\min_{y^{n-1}} \{ l(y^{n-1}, y_{next}, y_{endpoint}) \}$

intractable (NP hard) in general

in practice: use heuristics to approximate it

but sometimes can compute exactly

eg. $T_{ref}$ for Hamming loss is just copy ground truth

LOLS -> "locally optimal learning to search"

ICML 2015 "L2S better than your teacher"

\[ \begin{array}{c|c|c|c|c|c|c} & \text{approximate} & \text{learned that} & \text{learned that} & \text{inconsistent} & \text{inconsistent} & \text{inconsistent} \\ \hline \text{roll-out} & \text{reference policy} & \text{reference policy} & \text{reference policy} & \text{reference policy} & \text{reference policy} & \text{reference policy} \\ \hline \text{learned (use } T_{ref} \text{)} & \text{consistent not locally optimal} & \text{consistent and locally optimal} & \text{reinforcement learning} & \text{not using enough information} & \text{can learn a policy is optimal up to one step deviation} & \text{can learn a policy is optimal up to one step deviation} \\ \hline \end{array} \]

1000

\[ \begin{array}{c|c|c|c|c} & \text{is cannot do better}\text{ than reference} & \text{is not using enough information} & \text{can learn a policy is optimal up to one step deviation} & \text{is not using enough information} \\ \hline a \rightarrow c \rightarrow 10 & \text{is not using enough information} & \text{can learn a policy is optimal up to one step deviation} & \text{is not using enough information} & \text{can learn a policy is optimal up to one step deviation} \\ \hline b \rightarrow 10 & \text{can learn a policy is optimal up to one step deviation} & \text{is not using enough information} & \text{can learn a policy is optimal up to one step deviation} & \text{is not using enough information} \\ \hline \end{array} \]

Example of approximate $T_{ref}$: $l$ is taken score

(heuristic: translate)

Consider all possible suffixes from ground truth and pick the best/best score one.
**SEARN**: apply LAS to RNN training

[Lehman et al., ICLR 2015]  
\[ P(y_{t+1} | y_{1:t}, x) \sim \text{ANN cell} \]

\[ \log p(y_{t+1} | y_{1:t}, x) \text{ of a (decoder) RNN} \]

If you use \(-\log p(y_{t+1} | y_{1:t-1}, x)\) as a cost surrogate

and \(y_{t+1} = \text{ground truth}\)

then LAS with \(\text{ref roll-in}\) is standard MLE.

---

Cost-sensitive surrogate losses choice

1. Structured SVM: \[ \max_{y} \left[ A(y') + s(y') \right] - s(y_{target}) \]
   \[ \leq c(y') - c(y_{target}) \]
   
   \[ y_{target} = \arg\max \ c(y') \]

2. "Target log-loss" \(-\log p(y_{target} | y_{1:t-1}, x)\)

---

**Cost-sensitive surrogate losses**

MLE vs ground truth & MLE

**Structured predction anomaly rectifies (SPEN3)**

ICML 2016

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

- relax \(y \in \mathbb{R}^{13T} \rightarrow \mathbb{R}^{13T}\)
  
  \[ \text{hull}(x) = \text{few steps of projected GD on } E(x, y; w) \]

  w.r.t. \(y\)

  - rounding

2016 paper:

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

\[ \max_{y \in \mathbb{R}^{13T}} \left( l(y_{i}, \tilde{y}) - E(x^{i}, \tilde{y}; w) + E(x^{i}, y^{i}, y) \right) \]

- requires on extension
\[ y = w^T \theta \]

Approximate subgradient is:
\[ -\nabla_{\theta} E(x_i^t, y_i^t; \theta) + \nabla_{\theta} E(x_i^t, y_i^t; \theta) \]

E.g. See Clarke subdifferential for generalization on non-convex \( f(x) \).

2017 paper: end-to-end learning:

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

\[ y_i = y_{i-1} - \eta_i \frac{\partial}{\partial y} E(x, y_i; \theta) \]

\[ y_{i+1} = y_0 - \eta_{i+1} \frac{\partial}{\partial y} E(x, y; \theta) \]

Recursively defined.