Lecture 23 - L2S; SeaRNN; SPENs

Thursday, April 11, 2024 9:39 AM

today : • learning to search
• SeaRNN
• SPENs

Learning to search (L2S)

SEARN Hal Daumé's PhD thesis

$h_w : \mathcal{X} \to \mathcal{Y}$

$h_w(x) = \underset{y \in \mathcal{Y}}{\operatorname{argmax}}\ s(x, y; w)$] parameterized search procedure

today : special case of SEARN : learning to do greedy search

split y in ordered list of decisions

$(y_1, y_2, \dots, y_T)$

learning a classifier $\pi_w(\text{feature}(\hat{y}_1, \hat{y}_2, \dots, \hat{y}_{t-1}; x)) = \hat{y}_t$

π_w classifier "decoding policy"

L2S framework

from $(x^{(i)}, y^{(i)})_{i=1}^n$ and $\ell(\cdot, \cdot)$

learn a good classifier/policy π_w s.t. $\hat{y}_t = \pi_w(\varphi(\hat{y}_{1:t-1}; x))$

$h_{\hat{w}}(x) \triangleq (\hat{y}_1, \dots, \hat{y}_T)$ "greedy decoder"

s.t. $\ell(y^{(i)}, h_{\hat{w}}(x^{(i)}))$ is small

(*) central idea : "reduction" where reduces structured prediction problem to problem of cost sensitive classification learning of π_w

method : generate training data for classifier π_w

$(\hat{y}^{(j)}_{\text{context}}, x^{(j)}, \text{cost}(y_{\text{next}}))$ ← cost function of y_{next}

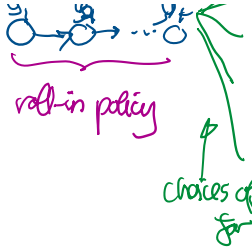
$\hat{y}^{(j)}_{\text{context}}$: prefix sequence to make next prediction

"roll in" policy → determines how $\hat{y}_{1:t-1}$ context

"roll-out" policy → " " $y_{t+1}, \dots, y_T$ "target completion" to get

$\text{cost}(y_{\text{next}}) \to \ell(y^{(i)}, (\hat{y}_{1:t-1}, y_{\text{next}}, \hat{y}_{t+1:T}))$

from roll-in from roll-



cost(y_{next}) → ... $\ell(\hat{y}_{1:t-1}, y_{next}, y_{t+1:T})$

from roll-in | from roll-out

↳ then idea is to define surrogate loss for π_w as function of cost(y_{next})

ex of roll-out:
"reference policy", ideally, $\pi_{ref}(\hat{y}_{1:t-1}, y_{next})$

$= \underset{y_{completion}}{\text{argmin}}\ \ell(y^{(i)}, (\hat{y}_{1:t-1}, y_{next}, y_{completion}))$

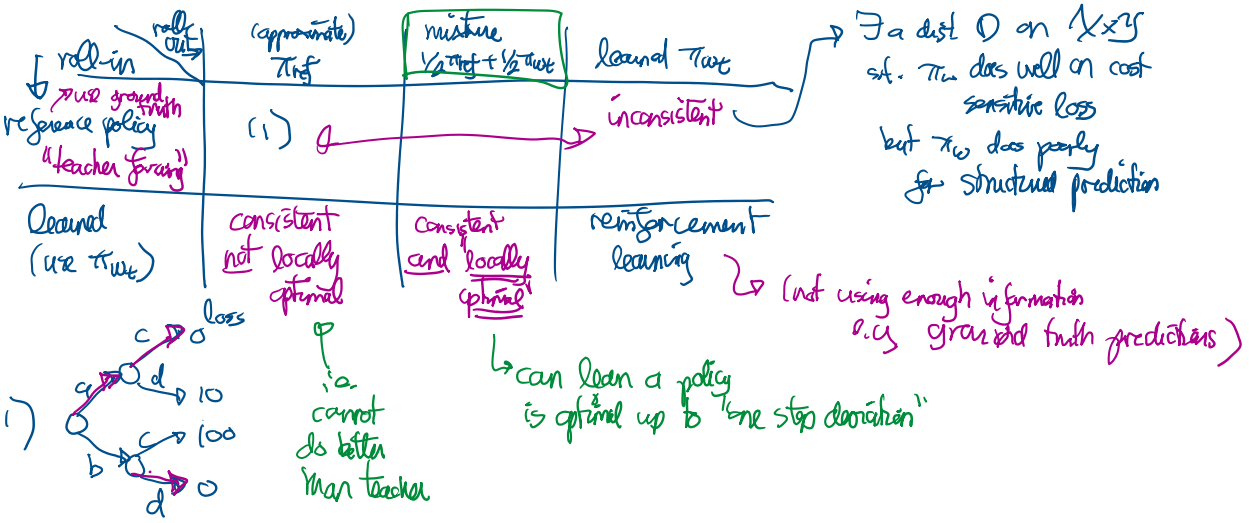
intractable (NP hard) in general

in practice: use heuristic to approximate π_{ref}

but sometimes, can compute exactly

e.g. π_{ref} for Hamming loss: is just copying ground truth

LOLS ↳ "locally optimal learning to search"
ICML 2015 "LOLS better than your teacher"



example of approximate π_{ref}: ℓ is bleu score (in machine translation)
consider all possible suffixes of ground truth and pick best bleu score

10:33

SEARNN: apply LOLS to RNN training

[Leblond & al. ICLR 2018] is $\pi_w(y_{1:t-1}, x)$ ~ RNN cell

$p(y_{next} \mid y_{1:t-1}, x)$ of a (decoder) RNN

if you use $-\log p(\hat{y}_{target} \mid y_{1:t-1}, x)$ as a cost surrogate

if you use $-\log p(\hat{y}_{\text{target}} \mid y_{1:t-1}, x)$ as a cost surrogate

and $\hat{y}_{\text{target}}$ = ground truth

then Las with π_{ref} roll-in is standard MLE

(*) cost-sensitive surrogate losses choices

a) structured SVM : $\max_{y'} [\underbrace{\Delta(y')}_{\triangleq c(y') - c(y_{\text{target}})} + s(y')] - s(y_{\text{target}})$

$\triangleq \text{argmin}_{y'} c(y')$

y_{target}

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

↳ differences with MLE :
- addres exposure bias using learned roll-in
- make use of structured loss $\ell(\cdot)$ to predict y_{target} vs. ground truth in MLE

Structured prediction energy networks (SPENs)

ICML 2016

$E(x, y; w)$ $\quad (-s(x, y; w))$

- relax $y \in \{0,1\}^T \rightsquigarrow [0,1]^T$

$h_w(x) =$ a few steps of projected G.D. on $E(x,y;w)$ w.r. to y (approximate prediction procedure)

+ rounding

2016 paper :

SSVM loss $\rightsquigarrow$ "subgradient" method on w

hinge loss $\ell(y^{(i)}, \bar{y}) - E(x^{(i)}, \bar{y}; w) + E(x^{(i)}, y^{(i)}; w)$

$\bar{y} \approx y^*(w)$ $\quad \bar{y} \in \arg\max_{\bar{y} \in [0,1]^T}$ (requires extension)

approximate "subgradient" is $-\nabla_w E(x^{(i)}, \bar{y}^*(w_t); w) + \nabla_w E(x^{(i)}, y^{(i)}; w)$ (through PGD)

because $\bar{y}^*$ is approximate max

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

2017 paper : end-to-end training

$h_w(x) = y_0 - \sum_{t=1}^{T} \eta_t \frac{d}{dy} E(x, y_t; w)$ ("unrolled optimization")

(y_t recursively defined)

$$y_1 = y_0 - \eta_1 \frac{d}{dy} E(x, y_0, w)$$

$$y_2 = y_1 - \eta_2 \frac{d}{dy} E(x, y_1, w) \text{ etc...}$$