SeaRNN

M. Bacher

ICL HD

18th of October 2018

Outline

1. Intro
2. SeaRNN
   - 1st Experiment
3. Performance
   - 2nd experiment
4. Neural Machine Translation
5. Conclusion | Critique

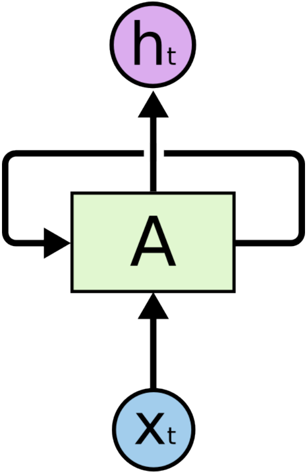
Intro

idea:

- improve RNN training
- integrate L2S into RNNs

RNNs

- neural network
- works on sequences
- used for tagging, machine translation



Motivation

problems of RNN training:

- maximum likelihood estimation
- teacher forcing
- local loss

MLE

- comes close to 0/1 loss
- only rewards reference
- not close to reference

Teacher Forcing

- training on ground truth
- testing on predictions
- leads to compounding errors

Local Level Loss

- loss is “local”
- loss function doesn’t consider the full sequence

Searn

- roll-ins
- roll-outs

Searn

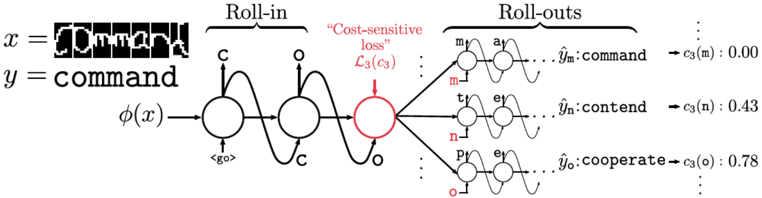


Figure 1: Leblond et al.

SeaRNN

- apply Searn to RNNs
- use rollouts
- rollouts enable new losses

SeaRNN Basic Idea

- roll-in the RNN
- at timestep t
- roll-out all actions
- calculate cost
- use cost for gradient update

SeaRNN

Algorithm 1 SEARNN algorithm (for a simple encoder-decoder network)

Initialize the weights ω of the RNN network.
for i in 1 to N **do**
    Sample B ground truth input/output structured pairs $\{(x^1, y^1), \cdots, (x^B, y^B)\}$
    # Perform the roll-in/roll-outs to get the costs. This step can be heavily parallelized.
    for b in 1 to B **do**
        Compute input features $\phi(x^b)$
        # Roll-in.
        Run the RNN until t^{th} cell with $\phi(x^b)$ as initial state by following the roll-in policy (see Appendix A.2 for details in the case of reference roll-in policy)
        Store the sequence of hidden states in order to perform several roll-outs
        for t in 1 to T **do**
            # Roll-outs for all actions in order to collect the cost vector at the t^{th} cell.
            for a in 1 to A **do**
                Pick a decoding method (e.g. greedy or beam search)
                Run the RNN from the t^{th} cell to the end by first enforcing action a at cell t, and then following the decoding method.
                Collect the cost $c_t^b(a)$ by comparing the obtained output sequence $\hat{y}_t^b(a)$ to y^b
            end for
        end for
    end for
    Derive a loss for each cell from the collected costs
    Update the parameters of the network ω by doing a single gradient step
end for

Figure 2: Leblond et al.

Policy choice

- follow LOLS [Chang et al.]

roll-out → ↓ roll-in	**Reference**	**Mixture**	**Learned**
Reference	Inconsistent		
Learned	Not locally opt.	Good	RL

Figure 3: Chang et al.

New Losses

- use sequence information
- reward good, non-ground truth results

Log Loss

$$L_t(s_t; c_t) = -log(e^{s_t(a^*)} / \sum_{i=1}^{A} e^{s_t(i)}) \tag{1}$$

$$a^* = argmin_{a \in A} c_t(a) \tag{2}$$

Log Loss

- very similar to MLE
- maximize lowest cost action
- not ground truth action

Kullback Leibler

- see costs as probability distribution
- apply softmax to cost distribution
- P_C = Cost distribution over actions
- P_M = Model probability over actions
- similarity P_C and P_M

Kullback Leibler

$$L_t(s_t; c_t) = -\sum_{a=1}^{A}(P_C(a) * log(P_M(a))) \qquad (3)$$

Kullback Leibler

- provides more information

Optimization

- use online variant of Searn
- taken from Chang et al.
- works on mini-batches

First Experiment

- two datasets:
- OCR
- Corrupted Text Correction

OCR

- English words
- sequence of hand written characters

Corrupted Text

- 10 character sequence
- some characters randomly replaced
- random replacement of 30% or 50%

Results

Dataset		MLE		LL			KL		
			roll-in	learned	reference	learned	learned	reference	learned
			roll-out	mixed	learned	learned	mixed	learned	learned
OCR		2.8		1.9	2.5	1.8	**1.0**	1.4	1.1
Spelling	0.3	19.3		**17.7**	19.5	17.8	**17.7**	19.5	**17.7**
	0.5	41.9		**37.1**	43.2	37.6	38.1	43.2	**37.1**

Figure 4: Leblond et al.

Performance

- obvious problem
- for one gradient step
- roll-out over all actions
- OCR roll-out -> 26 actions

Performance

- machine translation -> 100 000 actions

Scaling Up

- don't roll-out all actions
- sample actions
- 3 sampling techniques

Scaling Up

- stochastic policy sampling
- biased policy sampling
- top-k sampling

Adapted Losses

- LL and KL applied to the sampled tokens

2nd experiment

- use sampling techniques
- on OCR and Spelling

Results

- 5x computation time speedup

Dataset		MLE	LL	KL	sLL				sKL			
					uni.	pol.	bias.	top-k	uni.	pol.	bias.	top-k
OCR		2.8	1.9	1.0	1.7	1.8	1.8	1.5	1.2	1.2	**0.9**	1.4
Spelling	0.3	19.3	17.7	17.7	17.6	17.7	17.7	**17.6**	18.4	17.7	17.7	18.2
Spelling	0.5	41.9	37.1	38.1	37.0	37.1	36.6	**36.6**	37.8	37.6	37.1	38.0

Figure 5: Leblond et al.

NMT

- neural Machine Translation
- German - English

NMT

- don't use one of the sampling methods
- use "custom" sampling
- mixing of top-k and ground truth neighbors

Policy Choice

"we use a reference roll-in and a mixed roll-out."

roll-out → / ↓ roll-in	**Reference**	**Mixture**	**Learned**
Reference	Inconsistent		
Learned	Not locally opt.	Good	RL

Figure 6: Chang et al.

Policy Choice

"One potential factor is that our reference policy is not good enough to yield valuable signal when starting from a poor roll-in. Another possibility is that the underlying optimization problem becomes harder when using a learned rather than a reference roll-in."
[Leblond et al.]

Results

MLE*	MIXER*	SEARNN (conv)	MLE†	BSO†	MLE'	AC'	MLE	SEARNN	MLE (dropout)	SEARNN (dropout)
17.7	20.7	20.5	22.5	23.8	25.8	27.5	24.8	26.8	27.4	**28.2**

Figure 7: Leblond et al.

Conclusion

- "SEARNN is a full integration of the L2S ideas to RNN training, whereas previous methods cannot be used for this purpose directly. "[Leblond et al.]
- clear improvement over MLE
- mostly consistent results with LOLS

Critique

Computational performance?

Improvement by sampling?

Performance comparison to other systems?

NMT Policy Choice

Choice of Reference Roll In?

Chang literally says "Roll-in with π^{ref} is bad."

Is this somewhat teacher forcing?

Would this work with better learned policy?

Sources

Remi Leblond, Jean-Baptiste Alayrac, Anton Osokin, and Simon Lacoste-Julien. SEARNN: Training RNNs with global-local losses

Kai-Wei Chang, Akshay Krishnamurthy, Alekh Agarwal, Hal Daumé, III, and John Langford. Learning to search better than your teacher. In ICML, 2015.

Hal Daumé, III, John Langford, and Daniel Marcu. Search-based structured prediction. Machine Learning, 2009.