SeaRNN: training RNNs with global-local losses




Rémi Leblond*, Jean-Baptiste Alayrac*, Anton Osokin, Simon Lacoste-Julien

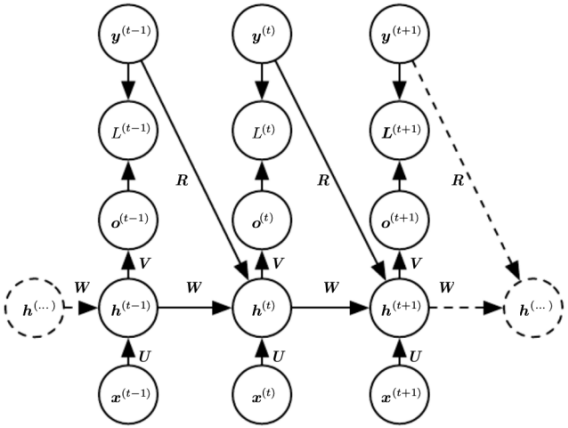
INRIA / Ecole Normale Supérieure MILA/DIRO UdeM

*equal contribution

RNNs: models for sequential data

Produce a sequence of hidden states by repeatedly applying a **cell** or **unit** on the input.

Can predict based on their **previous outputs**.

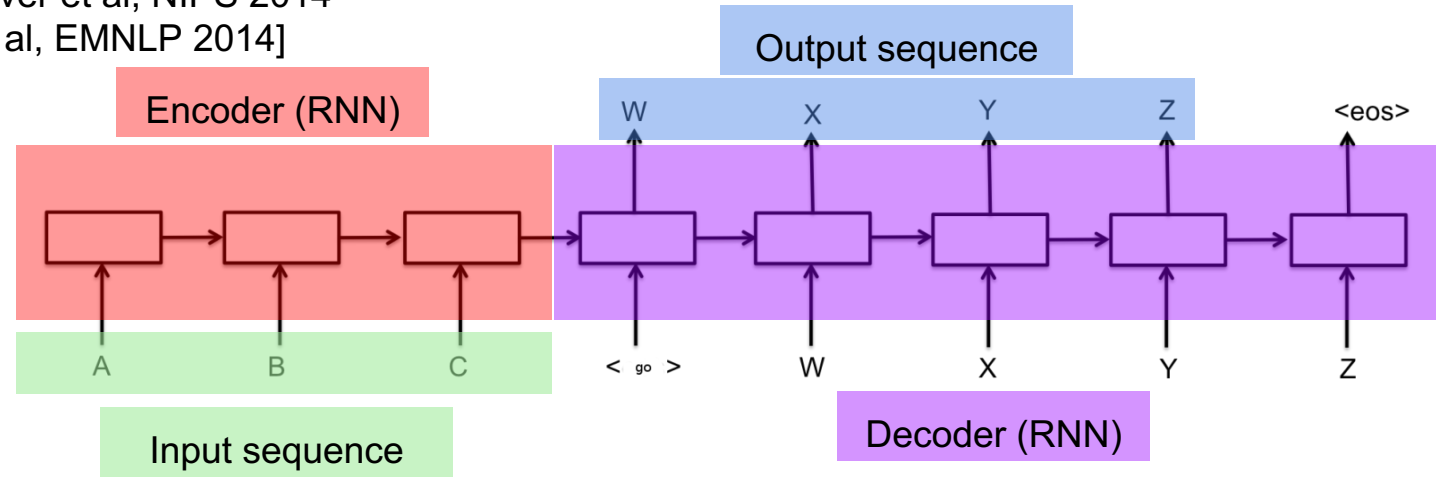


From Goodfellow et al, 2016

$$h_t = f(h_{t-1}, y_{t-1})$$
$$s_t = \text{proj}(h_t)$$
$$o_t = \text{softmax}(s_t)$$

Encoder-decoder architecture

[Sutskever et al, NIPS 2014
Cho et al, EMNLP 2014]



The **encoder** RNN maps the input sequence into a **compact representation** that is fed to the **decoder** RNN. The decoder outputs a sequence by taking **sequential decisions** given the past information.

State of the art for translation and other tasks.

Standard training

Probabilistic interpretation: $o_t = P(Y_t|X, Y_1, ..., Y_{t-1})$

Chain rule: $\prod_{t=1}^{T} o_t = P(Y_1, ..., Y_T|X)$

Training with MLE (teacher forcing): $\max_\theta \sum_{i=1}^{n} \log(P_\theta(Y = Y_X|X))$

Known problems of MLE:

* *different from the test loss*,
* *all-or-nothing loss* (bad for structured losses),
* exposure bias leading to compounding error.

Existing approaches: Bahdanau et al (ICLR 2017), Ranzato et al (ICLR 2016), Bengio et al (NIPS 2015), Norouzi et al (NIPS 2016)

Structured prediction

Goal: learn a prediction mapping f between inputs X and structured outputs Y, i.e. outputs that are made of interrelated parts often subject to constraints.

Examples: OCR, translation, tagging, segmentation...

Difficulty: there is an exponential number (with respect to the input size) of possible outputs (K^L possibilities if K is the alphabet size and L the number of letters).

Standard approaches: SVM struct, CRFs...

Learning to Search, a close relative?

[SEARN, Daumé et al 2009]

Makes predictions **one by one**: each Yi is predicted sequentially, conditioned on X and the previous Yj (instead of predicting Y in one shot).

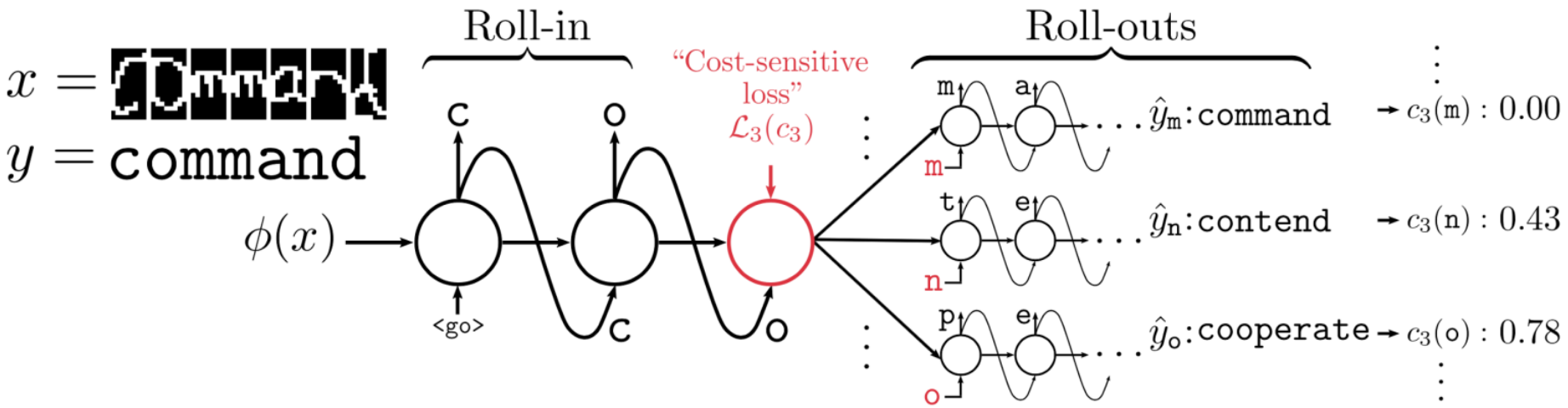
Enables **reduction:** instead of learning a global classifier for Y, we learn a **shared classifier** for the Yi.

Reduces SP down to a **cost-sensitive classification** problem, with **theoretical guarantees** on the solution quality.

Bonus: it addresses the problem mentioned before with MLE!

L2S, roll-in/roll-out

Trained with an **iterative procedure**: we create **intermediate datasets** for our shared cost sensitive classifier using **roll-in/roll-out** strategies.



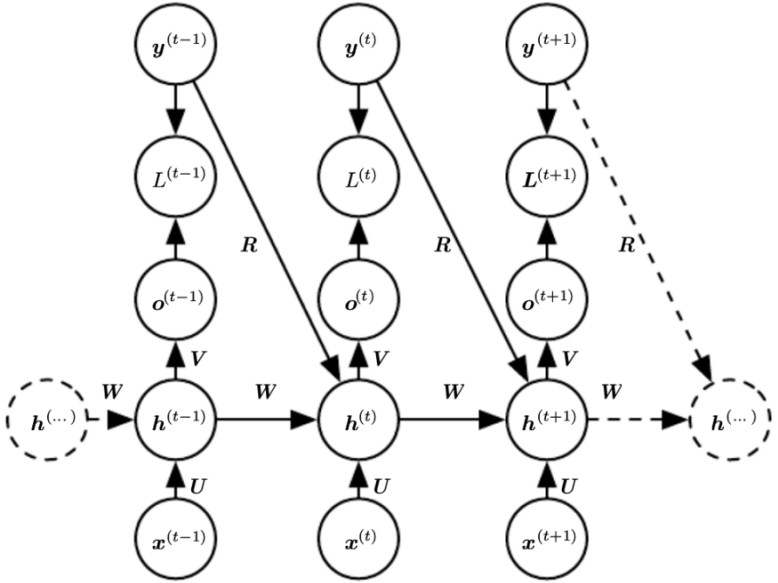
Links to RNNs

Both rely on decomposing structured tasks into **sequential predictions**, conditioned on the past.

Both use a **unique shared classifier** for every decision, using previous decisions.

What ideas can we share between the two?

While RNNs have built-in roll-ins, they don't have roll-outs. Can we train RNNs using the **iterative procedure** of learning to search?



From Goodfellow et al, 2016

Our approach: SeaRNN

Idea: use concepts from **learning to search** in order to train the **decoder** RNN.

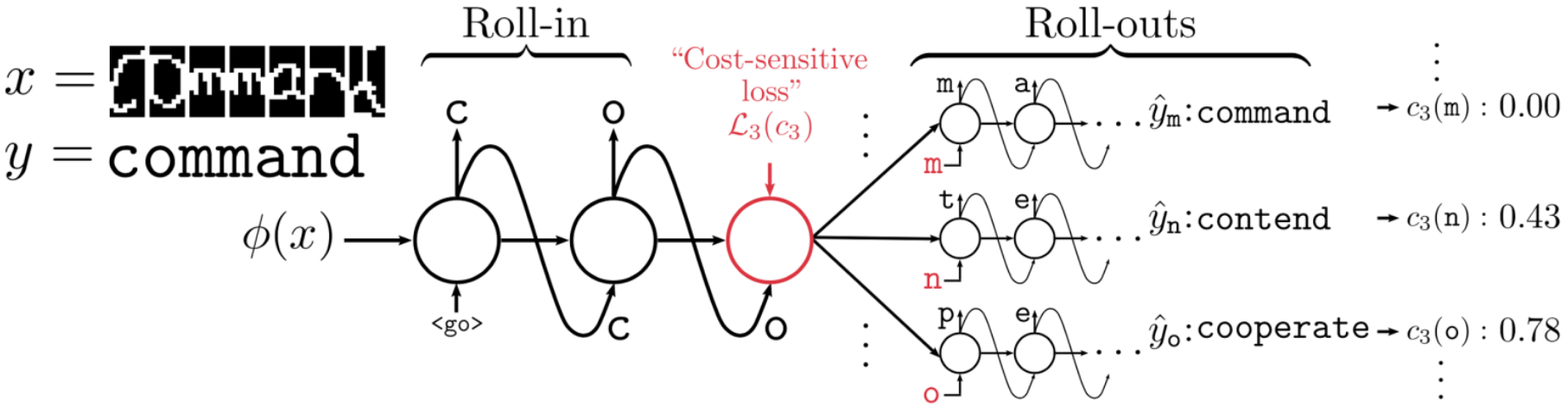
Integrate roll-outs in the decoder to compute the cost of every possible action at every step.

Leverage these costs to enable better training losses.

Algorithm:

1) Compute costs with roll-in/outs
2) Derive a loss from the costs
3) Use the loss to take a gradient step
4) Rinse and repeat

Roll-outs in RNNs



The devil in the details

Roll-in: reference (teacher forcing)? learned?

Roll-out: reference? learned? mixed?

We can leverage L2S theoretical results!

roll-out → / ↓ roll-in	**Reference**	**Mixed**	**Learned**
Reference	MLE (with TL)	Inconsistent	
Learned	Not locally opt.	Good	RL

From Chang et al, 2015

Cost sensitive losses: since RNNs are tuned to be trained with MLE, can we find a structurally similar loss that leverages our cost information?

Scaling: compared to MLE, our approach is very costly. Can we use subsampling to mitigate this? What sampling strategy should we use?

Expected benefits

Make **direct use** of the test error.

Leverage **structured information** by comparing costs, contrary to MLE.

Global-local losses, with **global** information at each **local** cell, whereas alternatives either use local information (MLE) or only work at the global level (RL approaches).

Sampling: **reduced computational costs** while maintaining improvements.

Experimental results

SeaRNN (full algorithm) on OCR, text chunking and spelling correction:

Dataset		A	T	Cost	MLE		LL			LLCAS		
						roll-in	learned	reference	learned	learned	reference	learned
						roll-out	mixed	learned	learned	mixed	learned	learned
OCR		26	15	Hamming	2.8		1.9	2.5	**1.8**	1.9	2.4	1.9
CoNLL		22	70	norm. Hamming	4.2		**3.7**	6.1	5.6	5.8	5.3	5.1
Spelling	0.3	43	10	edit	19.6		17.8	19.5	17.9	**17.7**	19.6	**17.7**
	0.5				43.0		37.3	43.3	37.5	**37.1**	43.3	38.2

Sampling results:

Dataset		MLE	LL					sLL					sLLCAS				
			uni.	stat.	pol.	top-k	bias.	uni.	stat.	pol.	top-k	bias.	uni.	stat.	pol.	top-k	bias.
OCR		2.84	1.94	**1.50**	1.96	2.13	1.84	1.82	1.91	1.86	2.69	2.25	2.03	2.33	**1.50**	1.94	2.37
Spelling	0.3	19.6	**17.7**	17.8	17.9	17.8	18.0	18.8	18.9	18.3	18.4	18.39	18.8	18.7	**17.7**	18.2	**17.7**
	0.5	43.0	37.0	36.9	37.3	**36.6**	37.0	37.4	37.5	37.3	41.7	37.6	37.6	37.7	37.0	40.5	37.8

Experimental takeaways

Significant improvements over MLE on **all 3 tasks**.

The **harder** the task, the **bigger** the improvement.

Learned/mixed is the best performing strategy for roll-in/out.

The best performing losses are those **structurally close** to MLE.

No need for **warm start**.

Sampling works, maintaining improvements at a **fraction of the cost**.

Future work

Large vocabulary problems (e.g. machine translation)

Smarter sampling strategies

- hierarchical sampling
- curriculum sampling
- trainable sampling?

Cheaper approximation of costs: actor-critic model?

Thank you! Questions?

Come to our poster to discuss!

See our paper on arxiv: https://arxiv.org/abs/1706.04499