SeaRNN: training RNNs with global-local losses

*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.

\[
h_t = f(h_{t-1}, y_{t-1})
\]

\[
s_t = \text{proj}(h_t)
\]

\[
o_t = \text{softmax}(s_t)
\]

From Goodfellow et al, 2016
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: \[ o_t = P(Y_t|X, Y_1, \ldots, Y_{t-1}) \]

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

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.

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

\[ x = \text{command} \]
\[ y = \text{command} \]

\[ \phi(x) \]

"Cost-sensitive loss" \( \mathcal{L}_3(c_3) \)

Roll-outs:
- \( \hat{y}_m: \text{command} \rightarrow c_3(m) : 0.00 \)
- \( \hat{y}_n: \text{contend} \rightarrow c_3(n) : 0.43 \)
- \( \hat{y}_o: \text{cooperate} \rightarrow c_3(o) : 0.78 \)
The devil in the details

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

**Roll-out:** reference? learned? mixed?

We can leverage L2S theoretical results!

**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?

---

<table>
<thead>
<tr>
<th>roll-out →</th>
<th>Reference</th>
<th>Mixed</th>
<th>Learned</th>
</tr>
</thead>
<tbody>
<tr>
<td>↓ roll-in</td>
<td>MLE (with TL)</td>
<td>Inconsistent</td>
<td></td>
</tr>
<tr>
<td>Reference</td>
<td>Not locally opt.</td>
<td>Good</td>
<td>RL</td>
</tr>
</tbody>
</table>

From Chang et al, 2015
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:

<table>
<thead>
<tr>
<th>Dataset</th>
<th>$A$</th>
<th>$T$</th>
<th>Cost</th>
<th>MLE</th>
<th>LL</th>
<th>LLCAS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OCR</td>
<td>26</td>
<td>15</td>
<td>Hamming</td>
<td>2.8</td>
<td>1.9</td>
<td>2.5</td>
</tr>
<tr>
<td>CoNLL</td>
<td>22</td>
<td>70</td>
<td>norm. Hamming</td>
<td>4.2</td>
<td>3.7</td>
<td>6.1</td>
</tr>
</tbody>
</table>
| Spelling | 0.3 | 43  | edit    | 19.6| 17.8| 19.5| 17.9
|          | 0.5 | 10  |         | 43.0| 37.3| 43.3| 37.5

Sampling results:

<table>
<thead>
<tr>
<th>Dataset</th>
<th>MLE</th>
<th>LL</th>
<th>sLL</th>
<th>sLLCAS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OCR</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Spelling</td>
<td>0.3</td>
<td>19.6</td>
<td>17.7</td>
<td>17.7</td>
</tr>
<tr>
<td></td>
<td>0.5</td>
<td>43.0</td>
<td>37.0</td>
<td>37.0</td>
</tr>
</tbody>
</table>


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