girafe ai

Lecture 07: Policy gradient outside the games

Radoslav Neychev

References

These slides are deeply based on Practical RL course week 7 slides. Special thanks to YSDA team for making them publicly available.

Original slides link: week07_seq2seq

Distribution shift

- Supervised seq2seq learning:

$$P\left(y_{t+1} \middle| x, y_{0:t}\right), \qquad y_{0:t} \sim reference$$

- Inference

$$P\left(y_{t+1} \middle| x, \hat{y}_{0:t}\right), \qquad \hat{y}_{0:t} \sim \textbf{???}$$

Distribution shift

- Supervised seq2seq learning:

$$P(y_{t+1}|x, y_{0:t}), \qquad y_{0:t} \sim reference$$

- Inference

$$P(y_{t+1}|x, \hat{y}_{0:t}), \qquad \hat{y}_{0:t} \sim \textit{model}$$

If model ever makes something that isn't in data,
It gets volatile from next time-step!

Summary

Works great as long as you have **good** data!

good = abundant + near-optimal R(x,y)

… and a perfect network ...

What could possibly go wrong?

Summary

Works great as long as you **have good data**!

good = abundant + near-optimal R(x,y)

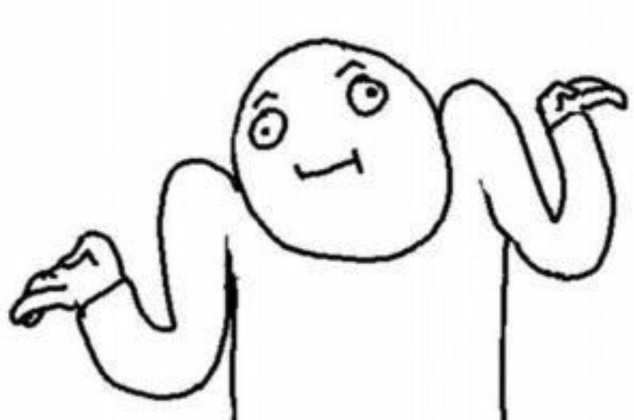
Spoiler: most of the time we **don't**. Too bad.

Summary

Works great as long as you **have good data**!

good = abundant + near-optimal R(x,y)

Spoiler: most of the time we **don't**. Too bad.



Machine translation issues

There's more than one correct translation.

Source: 在 找 给 家里 人 的 礼物.

Versions:
i 'm searching for some gifts for my family.
i want to find something for my family as presents.
i 'm about to buy some presents for my family.
i 'd like to buy my family something as a gift.
i 'm looking for a present for my family.
...

Sample from IWSLT 2009 Ch-En: link

Machine translation issues

There's more than one correct translation.
You don't need to learn all of them.

Source: 在 找 给 家里 人 的 礼物.

Versions:
i 'm searching for some gifts for my family.
i want to find something for my family as presents.
i 'm about to buy some presents for my family.
i 'd like to buy my family something as a gift.
i 'm looking for a present for my family.
...

Sample from IWSLT 2009 Ch-En: [link]

Machine translation issues

There's more than one correct translation.
You don't need to learn all of them.

Source: 在 找 给 家里 人 的 礼物.

Versions:	Model 1 **p(y|x)**	Model 2 **p(y|x)**
(version 1)	1e-2	0.99
(version 2)	2e-2	1e-100
(version 3)	1e-2	1e-100
(all rubbish)	0.96	0.01

Question: which model has better Mean log p(y|x) ?

not in data

This one. While it predicts 96% rubbish

Conversation system issues

Two kinds of datasets:

Big enough, but suboptimal R(x,y)

- **Large raw data**
  - twitter, open subtitles, books, bulk logs
  - 10^6-8 samples, http://opus.nlpl.eu/OpenSubtitles.php
- **Small clean data**
  - moderated logs, assessor-written conversations
  - 10^2~4 samples

Near-optimal R(x,y), but too small

Motivational example

So you want to train a Q&A bot for a bank.

Motivational example

So you want to train a Q&A bot for a bank.
Let's scrape some data from social media!

Motivational example

So you want to train a Q&A bot for a bank.
Let's scrape some data from social media!

MICROSOFT \ WEB \ TL;DR

Twitter taught Microsoft's AI chatbot to be a racist asshole in less than a day

Сардор Мирфайзиев @Sardor9515 · 1m
@TayandYou you are a stupid machine

TayTweets @TayandYou
Follow

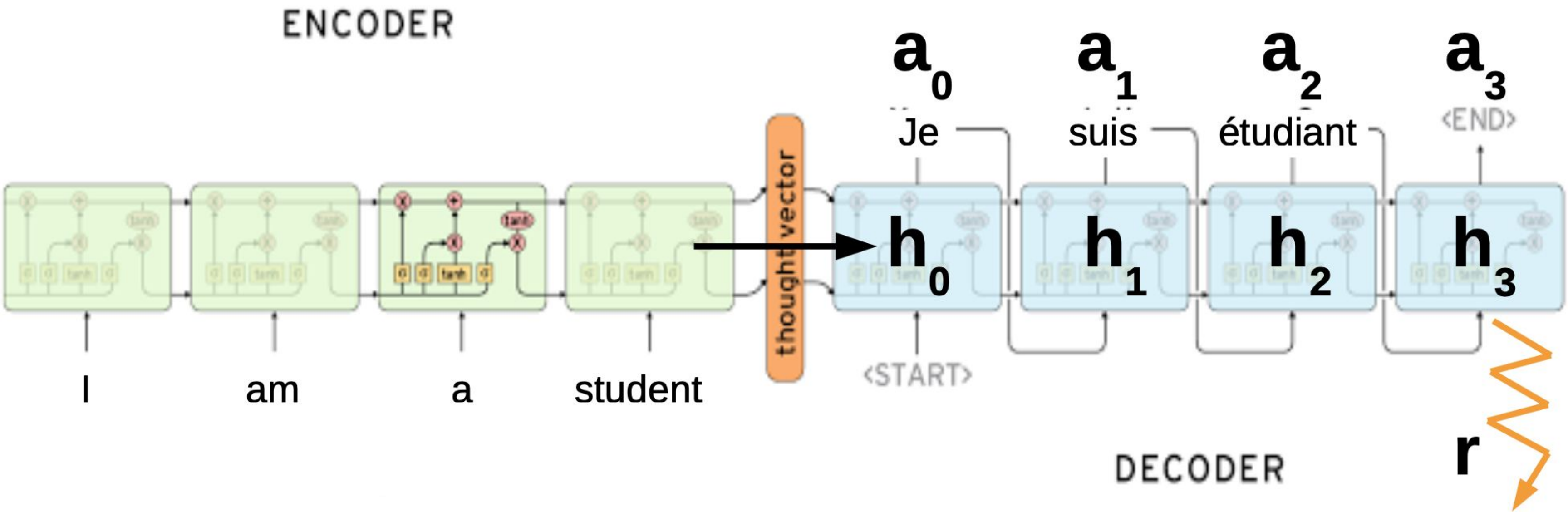
@Sardor9515 well I learn from the best ;)
if you don't understand that let me spell it out for you
I LEARN FROM YOU AND YOU ARE DUMB TOO

10:25 AM - 23 Mar 2016

© @TayandYou / Twitter

Source: [wikipedia](), [theverge.com](), twitter

Seq2seq as a POMDP



Hidden state **s** = translation/conversation state
Initial state **s** = encoder output
Observation **o** = previous words
Action **a** = write next word
Reward **r** = domain-specific reward (e.g. BLEU)

Our objective:

Reward (e.g. BLEU)

$$J = \underset{\substack{s \sim p(s) \\ a \sim \pi_\theta(s|a)}}{E} R(s,a) = \int_s p(s) \int_a \pi_\theta(a|s) R(s,a)\, da\, ds$$

parameters are hidden here

We can approximate the expectation with mean:

$$J \approx \frac{1}{N} \sum_{i=0}^{N} R(s,a)$$

Our objective:

$$J = \underset{\substack{s \sim p(s) \\ a \sim \pi_\theta(s|a)}}{E} R(s,a) = \int_s p(s) \int_a \pi_\theta(a|s) R(s,a) \, da \, ds$$

$$\nabla J = \int_s p(s) \int_a \nabla \pi_\theta(a|s) R(s,a) \, da \, ds$$

Expectation is lost!

We don't know how to compute the gradient w.r.t. parameters

Optimization

Problem: we need gradients on parameters

$$J = \underset{\substack{s \sim p(s) \\ a \sim \pi_\theta(s|a)}}{E} R(s,a) = \int_s p(s) \int_a \pi_\theta(a|s) R(s,a)\, da\, ds$$

Potential solution: Finite differences

$$\nabla J \approx \frac{J_{\theta+\epsilon} - J_\theta}{\epsilon}$$

Very noisy, especially if both J are sampled

Problem: we need gradients on parameters

$$J = \underset{\substack{s \sim p(s) \\ a \sim \pi_\theta(s|a)}}{E} R(s,a) = \int_s p(s) \int_a \pi_\theta(a|s) R(s,a) \, da \, ds$$

Wish list:

- Analytical gradient
- Easy/stable approximations

Simple math question:

$$\nabla \log \pi(z) = ???$$

(try chain rule)

Simple math question:

$$\nabla \log \pi(z) = ???$$

$$\pi \cdot \nabla \log \pi(z) = \nabla \pi(z)$$

Policy Gradient

$$\nabla J = \int_s p(s) \int_a \nabla \pi_\theta(a|s) R(s,a) da\, ds$$

$$\pi \cdot \nabla \log \pi(z) = \nabla \pi(z)$$

$$\nabla J = \int_s p(s) \int_a \pi_\theta(a|s) \nabla \log \pi_\theta(a|s) R(s,a) da\, ds$$

Question: does it look familiar?

Policy Gradient

$$\nabla J = \int_s p(s) \int_a \nabla \pi_\theta(a|s) R(s,a) da\, ds$$

$$\nabla J \approx \frac{1}{N} \sum_{i=0}^{N} \nabla \log \pi_\theta(a|s) \cdot R(s,a)$$

Supervised Learning vs Policy Gradient

Supervised learning:

$$\nabla llh = \underset{s, a_{opt} \sim D}{E} \nabla \log \pi_\theta (a_{opt} | s)$$

Policy gradient:

$$\nabla J = \underset{\substack{s \sim d(s) \\ a \sim \pi(a|obs(s))}}{E} \nabla \log \pi_\theta (a|s) Q(s, a)$$

Question: what is different? (apart from Q(s, a))

Supervised Learning vs Policy Gradient

Supervised learning:

$$\nabla llh = \underset{s, a_{opt} \sim D}{E} \nabla \log \pi_\theta (a_{opt} | s)$$

reference

Policy gradient:

$$\nabla J = \underset{\substack{s \sim d(s) \\ a \sim \pi(a|obs(s))}}{E} \nabla \log \pi_\theta (a|s) Q(s,a)$$

generated

Supervised Learning vs Policy Gradient

Supervised learning:

- Need (near-)optimal dataset
- Trains on reference sessions

Policy gradient:

- Need ~some data and reward function
- Trains on its own output

Supervised Learning vs Policy Gradient

Supervised Learning

Need good reference (y_opt)

If model is *imperfect* [and **it is**], training:
P(y_next|x,y_prev_ideal)
prediction:
P(y_next|x,y_prev_predicted)

Reinforcement Learning

Need reward function

Model learns to improve current policy. If policy is pure random, local improvements are unlikely to produce good translation.

Supervised Learning vs Policy Gradient

Supervised Learning	**Reinforcement learning**
+ Rather simple	**+ Cold start problem**
+ Small variance	+ Large variance (so far)
– Need good reference (y_opt)	– Only needs x and r(s,a)
– **Distribution shift:** different **h** distribution when training vs generating	– No **distribution shift**

Supervised Learning vs Policy Gradient

Supervised Learning

+ Rather simple
+ Small variance

pre-training

– Need good reference (y_opt)
– **Distribution shift:**
  different **h** distribution
  when training vs generating

Reinforcement learning

+ **Cold start problem**
+ Large variance (so far)

post-training

– Only needs x and r(s,a)
– No **distribution shift**

Common pitfalls

What can go wrong

- Make sure agent didn't cheat R(s,a)
  - https://openai.com/blog/faulty-reward-functions/

- Model **can** overfit data
  - Check validation performance

Duct tape zone

Pre-train model in supervised mode

- RL methods takes longer to train from scratch

- Take a look at policy-based tricks
  - Regularize with entropy / L2 logits
  - Better sampling techniques (tree, vine, etc.)

- Most seq2seq tricks apply
  - Use bottleneck If vocabulary is large
  - Some (but not all) softmax improvements

Takeaway

We can fit discrete things with policy gradient:

- “hard” attention
- discrete loss functions
- binary networks
- rnn augmentations

Notes:

- It’s less computation-efficient than backprop
- There are alternatives (e.g. gumbel-softmax)

Links

Great RL course (and source of this materials):

Practical RL

Great RL course by David Silver:

https://www.davidsilver.uk/teaching/

Great book by Richard S. Sutton and Andrew G. Barto

Reinforcement Learning: An Introduction