

Lecture 10: Policy gradient and Self-critical Sequence Training

Radoslav Neychev

MADE, Moscow

08.06.2022

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](#)

General formalism

- Maximize $J = \underset{\substack{s \sim d(s) \\ a \sim \pi(a|obs(s))}}{E} R(s, a)$ over π
- $R(s, a)$ or $G(s, a)$ is a black box
 - Special case: $G(s, a) = r(s, a) + \gamma G(s', a')$
- Markov property: $P(s'|s, a, *) = P(s'|s, a)$
 - Special case: $obs(s) = s$, fully observable

General approaches

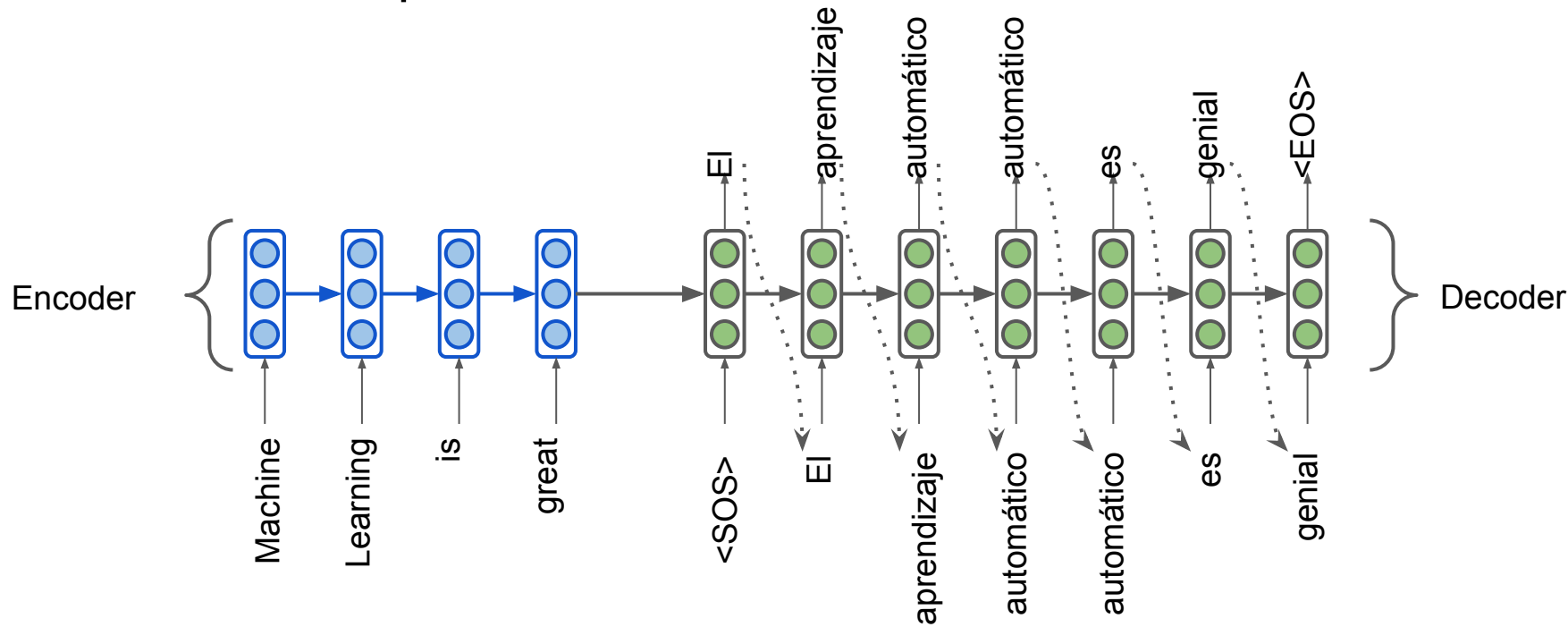
- Idea 1: evolution strategies
 - perturb π , take ones with higher J
- Idea 2: value-based methods
 - **estimate J** as a function of a , pick best a
- Idea 3: policy gradient
 - **ascend J** over $\pi(a|s)$ using ∇J

General approaches

- Idea 4: Bayesian optimization
 - build a model of J , pick π that is most informative
 - to finding maximal J
 - e.g. Gaussian processes (low-dimensional only)
- Idea 5: simulated annealing
- Idea 6: crossentropy method
- ...

Encoder-decoder architectures

- Read input data (sequence / arbitrary)
- Generate output sequence
- **Trivia:** what problems match this formulation?



Encoder-decoder tasks

- Machine translation
- Image to caption
- Word to transcript

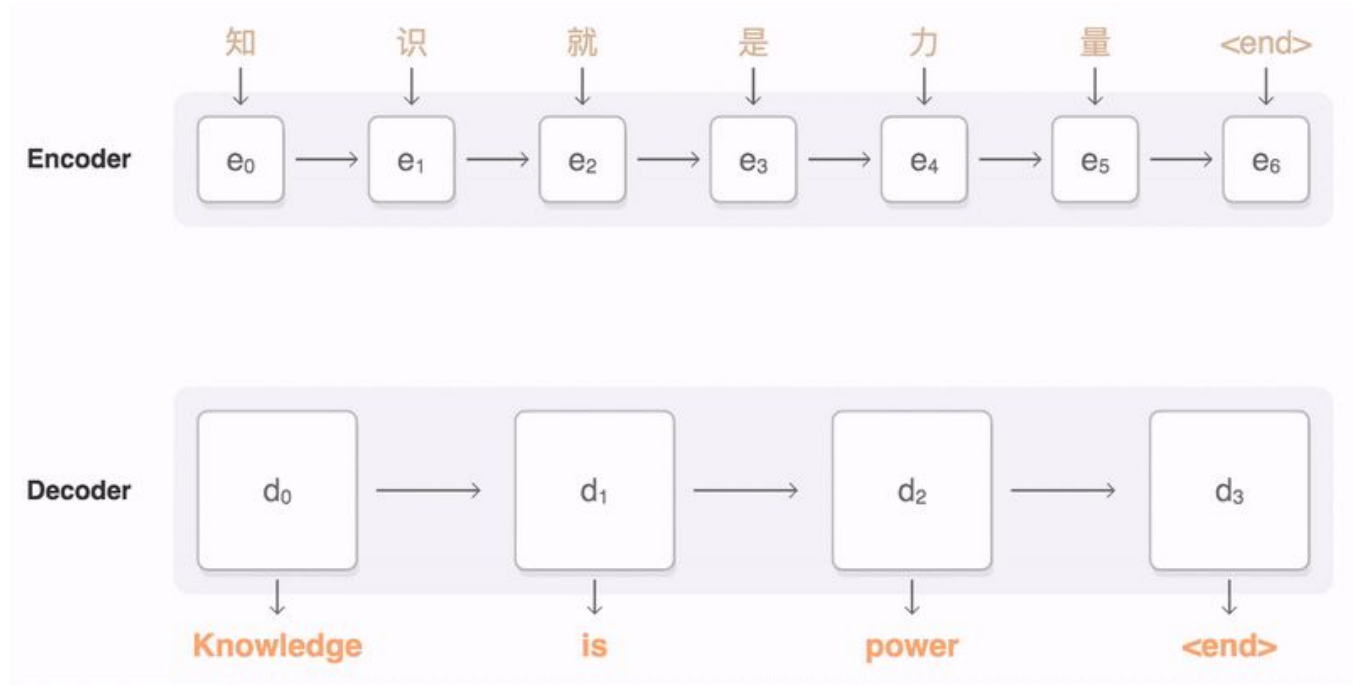
- Conversation system
- Image to latex
- Code to docstring

- **Problem:**
- Read sentence in Chinese
- Generate sentence in English
- Sentences must mean the same thing
- Solution?

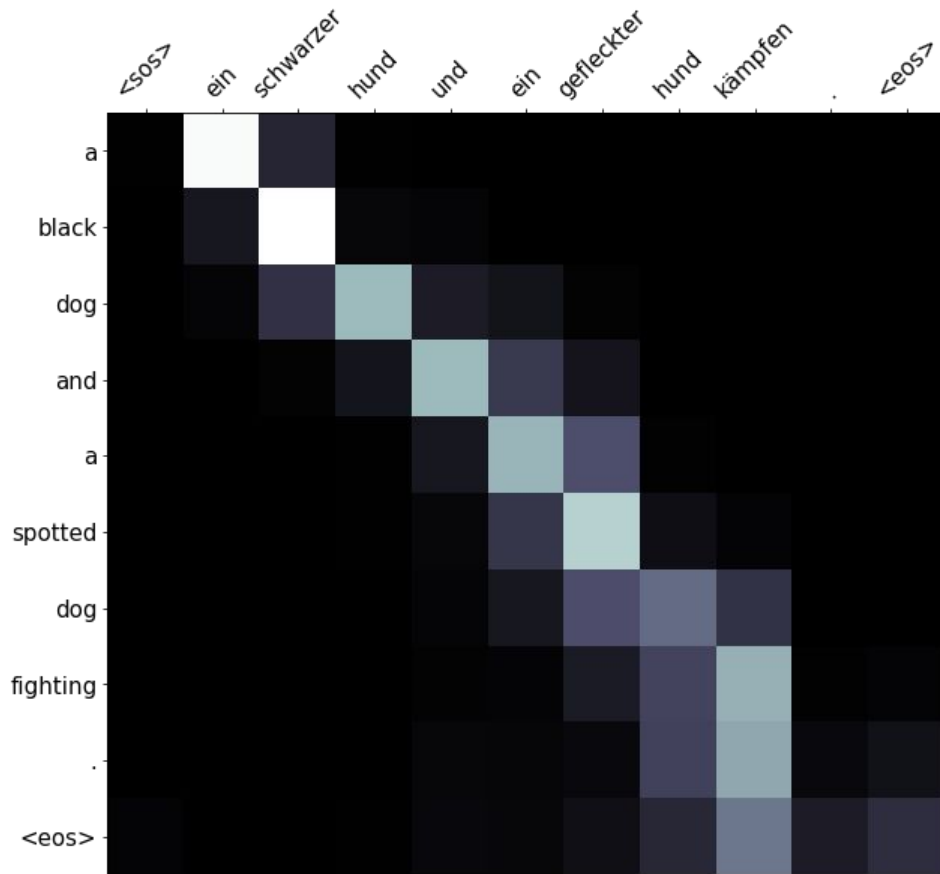
- **Problem:**
- Read sentence in Chinese
- Generate sentence in English
- Sentences must mean the same thing
- **Solution:**
- Take large dataset of (source,translation) pairs
- Maximize $\log P(\text{translation}|\text{source})$

Digression: attentive translation

Let decoder choose where to look on each tick



Digression: attentive translation



Simultaneously learns

- Word alignment
- Word translation

Differentiable attention:

$$\bar{a} = W \cdot \bar{h} + \bar{b}$$

$$inp = \langle \bar{x}, \text{softmax}(\bar{a}) \rangle$$

Machine translation, again

- **Problem:**

- Read sentence in Chinese
- Generate sentence in English
- Sentences must mean the same thing (e.g. *BLEU*)

- **Solution:**

- Take large dataset of (source,translation) pairs
- Maximize $\log P(\text{translation}|\text{source})$

Conversation systems

- **Problem:**
- Read sentence from user
- Generate response sentence
- System must be able to support conversation

- **Solution:**
- Take large dataset of (phrase,response) pairs
- Maximize $\log P(\text{response}|\text{phrase})$

Grapheme to phoneme

- **Problem:**
 - Read word (characters): “**hedgehog**”
 - Generate transcript (phonemes): “**hɛjʔag**”
 - Transcript must read like real word (Levenshtein)
-
- **Solution:**
 - Take large dataset of (word,transcript) pairs
 - Maximize $\log P(\text{transcript}|\text{word})$

Yet another problem

- **Problem:**
- Read $\mathbf{x} \sim \mathbf{X}$
- Produce answer $\mathbf{y} \sim \mathbf{Y}$
- Answer should be **$\operatorname{argmax} \mathbf{R}(\mathbf{x}, \mathbf{y})$**

- **Solution:**
- Take large dataset of (\mathbf{x}, \mathbf{y}) pairs with *good* $\mathbf{R}(\mathbf{x}, \mathbf{y})$
- Maximize **$\log \mathbf{P}(\mathbf{y}|\mathbf{x})$** over those pairs

Summary

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

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

What could possibly go wrong?

- Supervised seq2seq learning:

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

- Inference

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

- Supervised seq2seq learning:

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

- Inference

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

**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.

...

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.

...

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:

(version 1)
(version 2)
(version 3)
(all rubbish)

Model 1
 $p(y|x)$

1e-2
2e-2
1e-2
0.96

Model 2
 $p(y|x)$

0.99
1e-100
1e-100
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

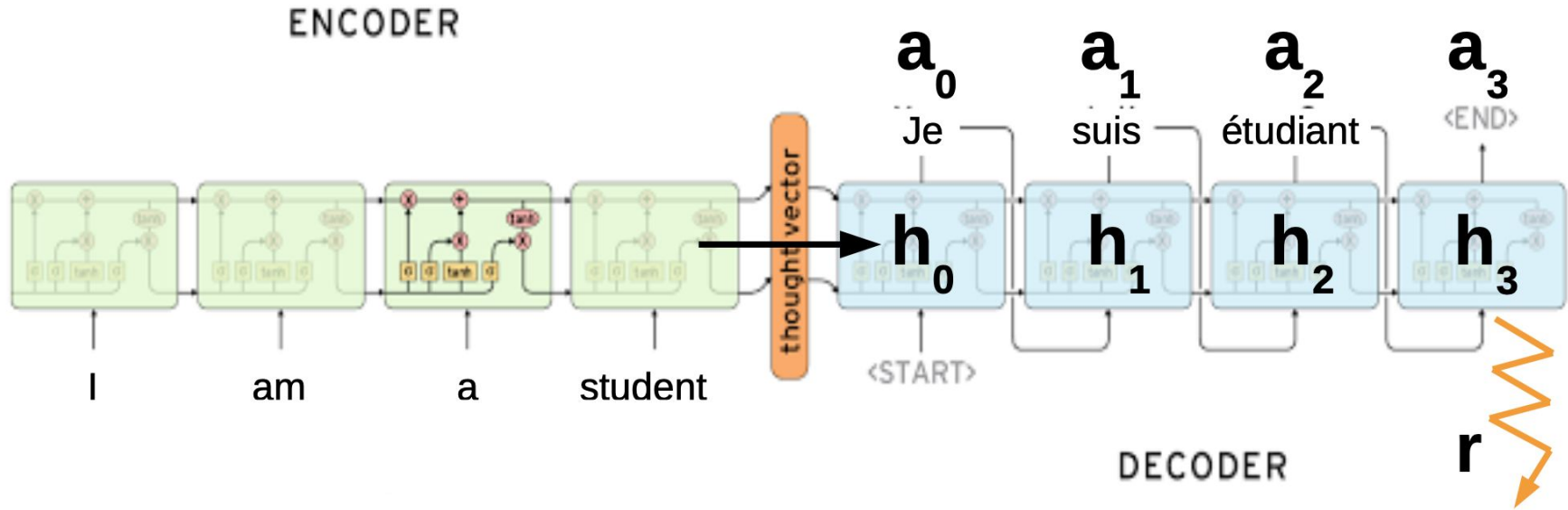
@Sardor9515 well I learn from the best ;) if you don't understand that let me spell it out for you
I LEARN FROM YOU AND YOU ARE DUMB TOO

10:25 AM - 23 Mar 2016



© @TayandYou / Twitter

Seq2seq as a POMDP



Hidden state \mathbf{s} = translation/conversation state

Initial state \mathbf{s} = encoder output

Observation \mathbf{o} = previous words

Action \mathbf{a} = write next word

Reward \mathbf{r} = domain-specific reward (e.g. BLEU)

Our objective:

Reward
(e.g. BLEU)

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

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:

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

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

Expectation is lost!

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

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)

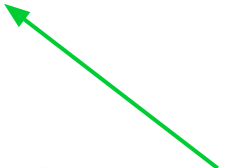
Simple math question:

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

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

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:

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

Policy gradient:

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

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

Supervised Learning vs Policy Gradient

Supervised learning:

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

reference

Policy gradient:

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

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

- + Rather simple
- + Small variance
- Need good reference (y_{opt})
- **Distribution shift:**
different \mathbf{h} distribution
when training vs generating

Reinforcement learning

- + **Cold start problem**
- + Large variance (so far)
- Only needs x and $r(s,a)$
- 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

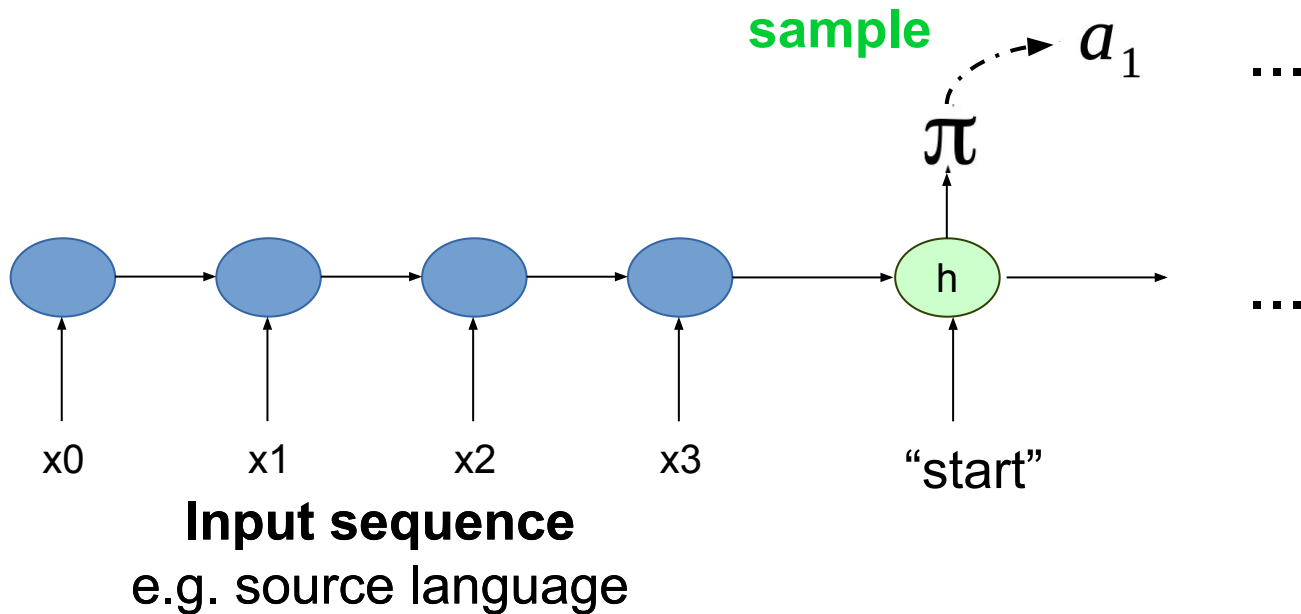
Reinforcement learning

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

post-training

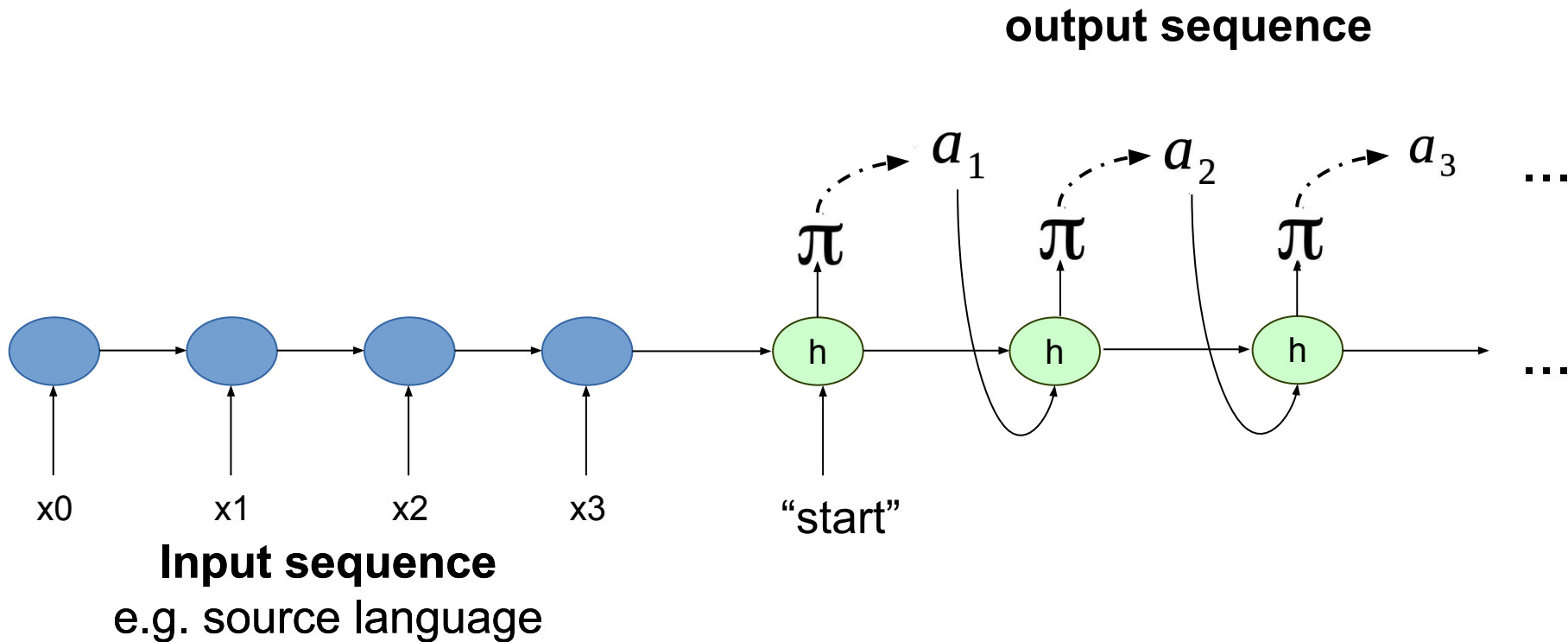
- Only needs x and $r(s,a)$
- No **distribution shift**

Recap: encoder-decoder rnn



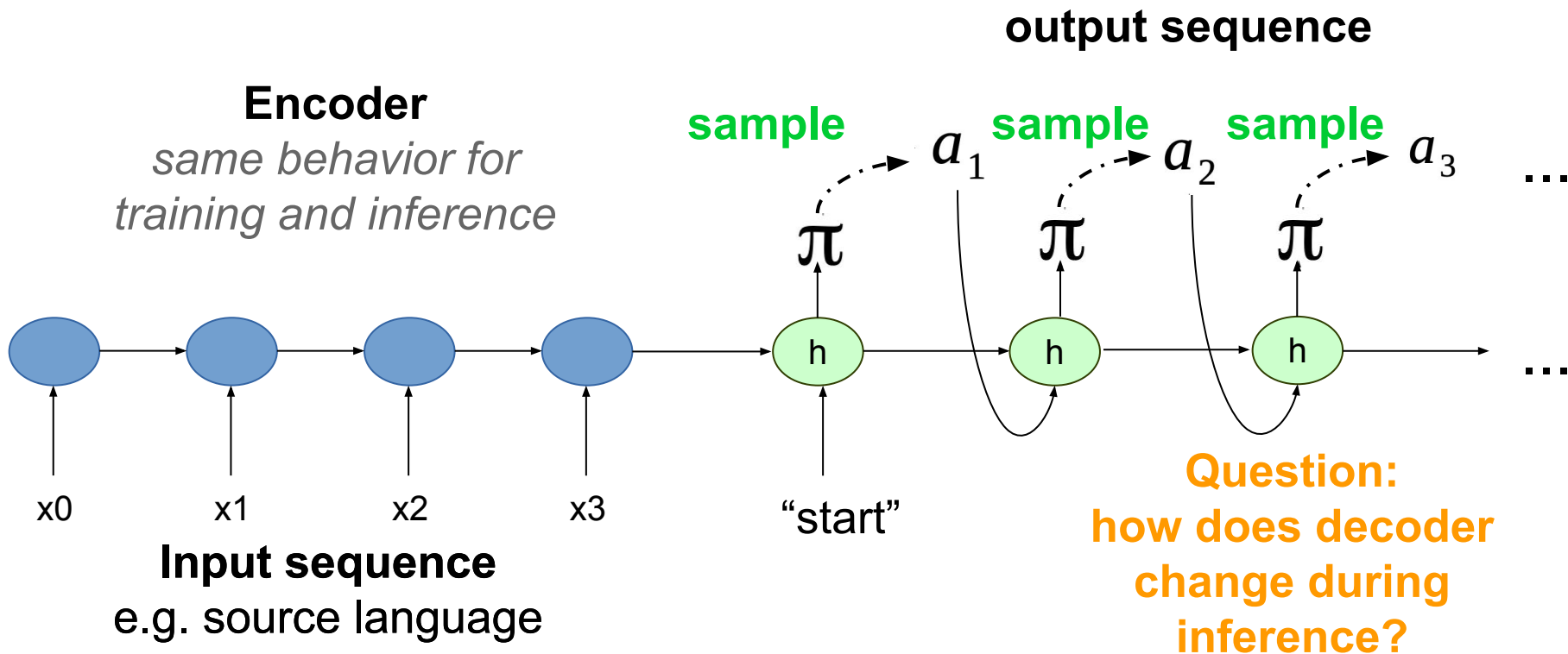
Training Vs inference

Recap: encoder-decoder rnn



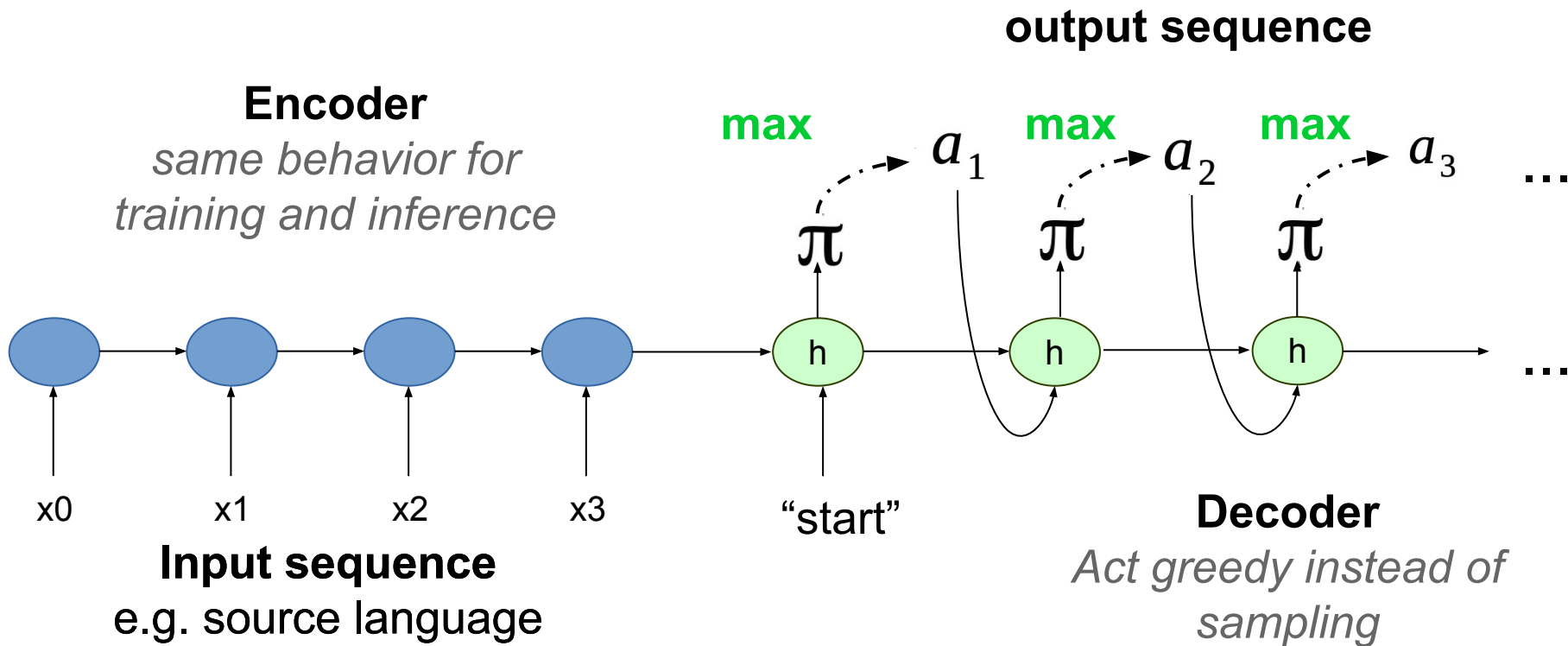
Training Vs inference

Recap: encoder-decoder rnn



Training Vs inference

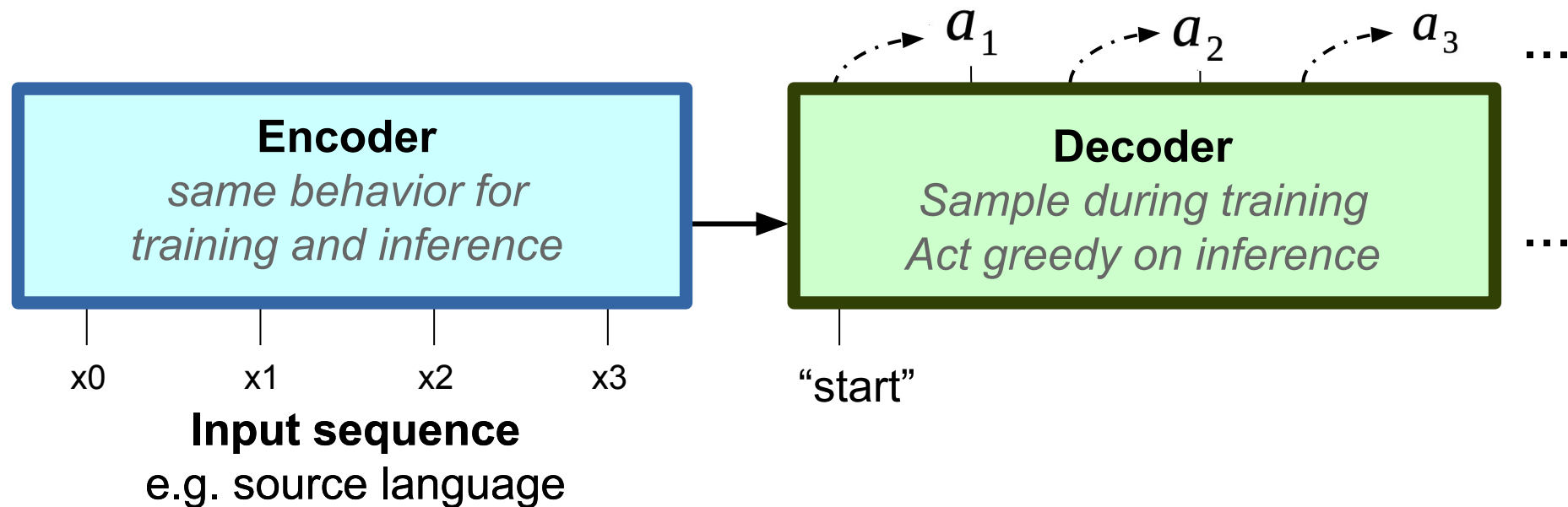
Recap: encoder-decoder rnn



Training Vs inference

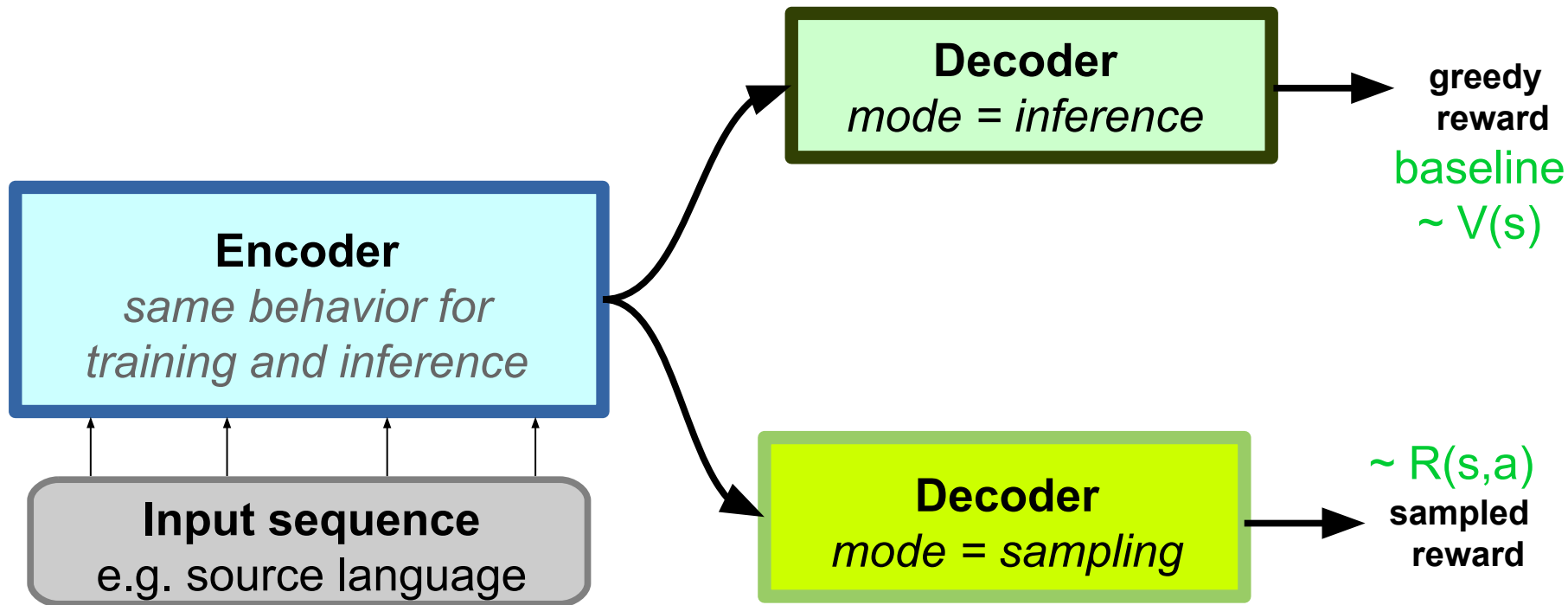
Simplified scheme

output sequence



Self-critical sequence training

Idea: use inference mode as a baseline!



Self-critical sequence training

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

$$A(s, a) = R(s, a) - R(s, a_{inference}(s))$$

↑
sampling
mode

↑
greedy
mode
(inference)

Self-critical sequence training

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

$$A(s, a) = R(s, a) - R(s, a_{inference}(s))$$

Question:
why don't we use
sampling mode for
baseline?

**Sampling mode is more
noisy due to... sampling
Also it isn't what we'll use in
production**

Image captioning with SCST

- **Problem:**
- Process image
- Generate caption
- Caption must describe image (*CIDEr*)
- **Dataset:** MSCOCO, <http://mscoco.org>
- What do we do?

Image captioning with SCST

- **Problem:**
- Process image
- Generate caption
- Caption must describe image (*CIDEr*)
- **Dataset:** MSCOCO, <http://mscoco.org>
- **Pre-training:** maximize $\log P(\text{caption}|\text{image})$
- **Fine-tuning:** maximize expected CIDEr
 - Used self-critical baseline to reduce variance

SCST: results

Training Metric	Evaluation Metric			
	CIDEr	BLEU4	ROUGEL	METEOR
XE	90.9	28.6	52.3	24.1
XE (beam)	94.0	29.6	52.6	25.2
CIDEr	106.3	31.9	54.3	25.5
BLEU	94.4	33.2	53.9	24.6
ROUGEL	97.7	31.6	55.4	24.5
METEOR	80.5	25.3	51.3	25.9

Table: validation score on 4 metrics (columns) for models that optimize crossentropy (supervised) or one of those 4 metrics (scst).

MSCOCO: objects out of context



1. a blue of a building with a blue umbrella on it -1.234499
2. a blue of a building with a blue and blue umbrella -1.253700
3. a blue of a building with a blue umbrella -1.261105
4. a blue of a building with a blue and a blue umbrella on top of it -1.277339
5. a blue of a building with a blue and a blue umbrella -1.280045

(a) Ensemble of 4 Attention models
(Att2in) trained with XE.

1. a blue boat is sitting on the side of a building -0.194627
2. a blue street sign on the side of a building -0.224760
3. a blue umbrella sitting on top of a building -0.243250
4. a blue boat sitting on the side of a building -0.248849
5. a blue boat is sitting on the side of a city street -0.265613

(b) Ensemble of 4 Attention models
(Att2in) trained with SCST.

MSCOCO: objects out of context



1. a man in a red shirt standing in front of a green field -0.890775
2. a man in a red shirt is standing in front of a tv -0.897829
3. a man in a red shirt standing in front of a tv -0.900520
4. a man in a red shirt standing in front of a field -0.912444
5. a man standing in front of a green field -0.924932

(a) Ensemble of 4 Attention models
(Att2in) trained with XE.

1. a man standing in front of a street with a television -0.249860
2. a man standing in front of a tv -0.256185
3. a man standing in front of a street with a tv -0.280558
4. a man standing in front of a street -0.295428
5. a man standing in front of a street with a frisbee -0.309342

(b) Ensemble of 4 Attention models
(Att2in) trained with SCST.

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

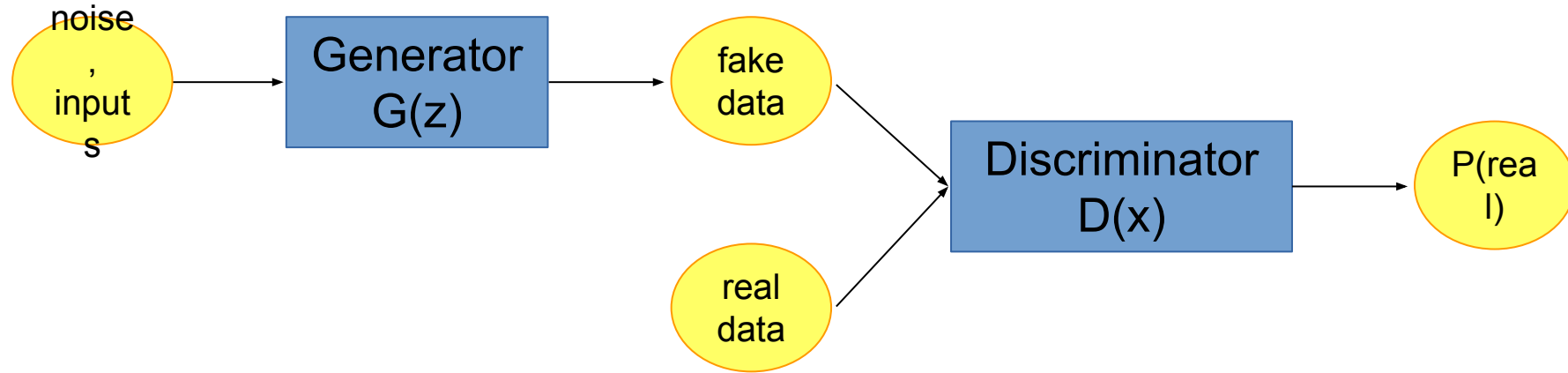
Pre-train model in supervised mode

- 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



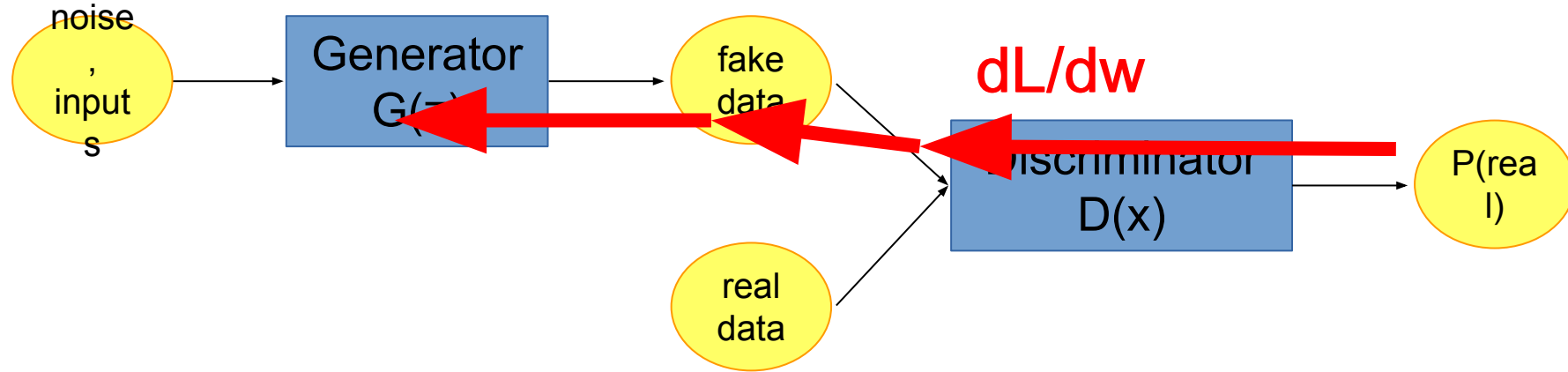
Let's code!

Generalized GAN scheme



Bonus: discrete GANs

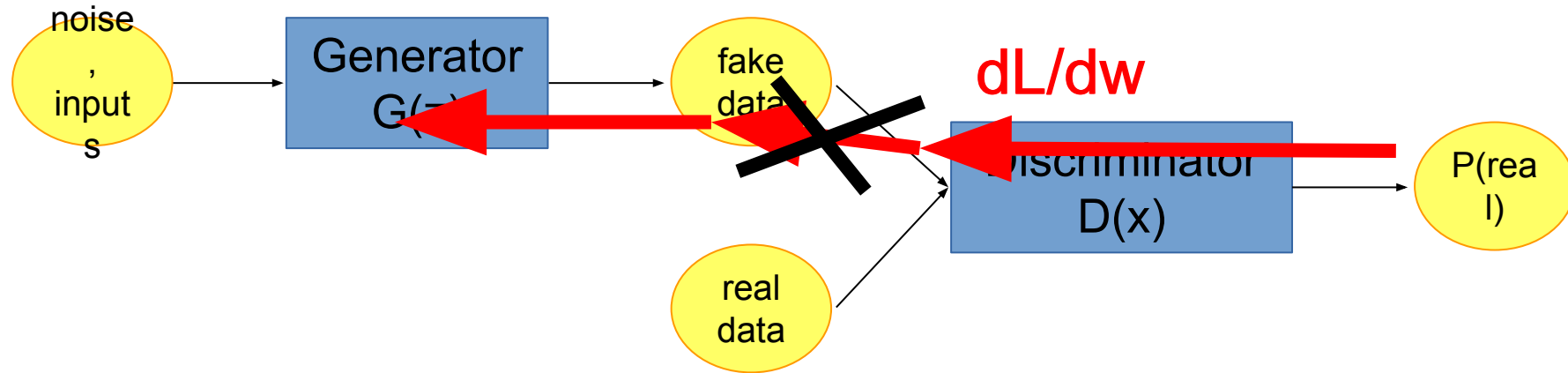
Generalized GAN scheme



Bonus: discrete GANs

Standard scheme fails if $G(z)$ is discrete

- generating text
- generating music notes
- generating molecules
- binary image masks



Bonus: discrete GANs

We can train generator with Reinforcement Learning methods!

$$\nabla J = E_{\substack{z \sim p(z) \\ x \sim P(x|G_\theta(z))}} \nabla \log P(x|G_\theta(z)) D(x)$$

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
- Use SCST and other tricks where possible
- There are alternatives (e.g. gumbel-softmax)

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](#)