Reinforcement Learning and NLP

Kapil Thadani
kapil@cs.columbia.edu

YAHOO!
RESEARCH

Outline

- Model-free RL
  - Markov decision processes (MDPs)
  - Derivative-free optimization
  - Policy gradients
  - Variance reduction
  - Value functions
  - Actor-critic methods
- Policy gradients in NLP
  - Non-differentiable metrics
  - Latent structure

Reinforcement learning

"*Instead of trying to produce a programme to simulate the adult mind, why not rather try to produce one which simulates the child's? If this were then subjected to an appropriate course of education one would obtain the adult brain* ."

— Alan Turing
Computing Machinery and Intelligence (1950)

Reinforcement learning

Sequential decision making

- Learn to model behavior over time
- Rewards may be stochastic and delayed
- Trade off exploration vs exploitation

Generalization of supervised learning

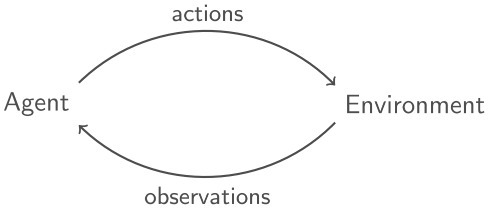
- No full access to function to optimize
- Stateful environment, input affected by previous actions
- Nonstationarity for samples (no i.i.d. assumption)

Reinforcement learning

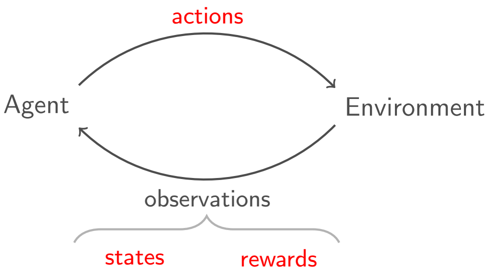
Agent

Environment

Reinforcement learning



Reinforcement learning



Reinforcement learning

Model-free

- Policy-based: learn how to take actions in each state
- Value-based: learn the value of actions in each state

Model-based

- Model environment to predict next states and rewards

Markov Decision Process (MDP)

Discrete-time stochastic control process defined by $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$

$\mathcal{S}$: set of states

$\mathcal{A}$: set of actions

$\mathcal{P}$: transition probability distribution

$$\mathcal{P}^a_{ss'} = p(s_{t+1} = s' | s_t = s, a_t = a)$$

$\mathcal{R}$: reward function

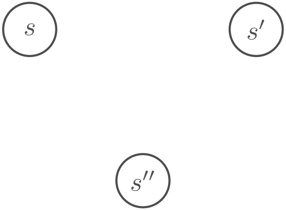
$$\mathcal{R}^a_{ss'} = \mathbb{E}\left[r_t | s_t = s, a_t = a\right]$$

$\gamma \in [0, 1)$: discount factor

Markov Decision Process (MDP)

Discrete-time stochastic control process defined by $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$

$\mathcal{S}$: set of states



$\mathcal{A}$: set of actions

$\mathcal{P}$: transition probability distribution

$$\mathcal{P}^a_{ss'} = p(s_{t+1} = s' | s_t = s, a_t = a)$$

$\mathcal{R}$: reward function

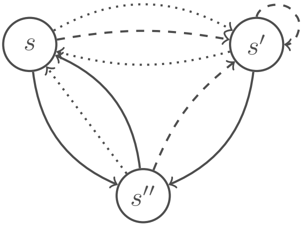
$$\mathcal{R}^a_{ss'} = \mathbb{E}\left[r_t | s_t = s, a_t = a \right]$$

$\gamma \in [0, 1)$: discount factor

Markov Decision Process (MDP)

Discrete-time stochastic control process defined by $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$

$\mathcal{S}$: set of states



$\mathcal{A}$: set of actions

$\mathcal{P}$: transition probability distribution

$$\mathcal{P}^a_{ss'} = p(s_{t+1} = s' | s_t = s, a_t = a)$$

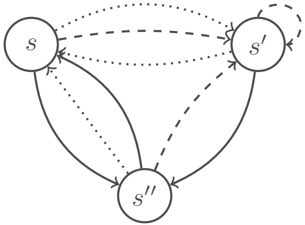
$\mathcal{R}$: reward function

$$\mathcal{R}^a_{ss'} = \mathbb{E}\left[r_t | s_t = s, a_t = a \right]$$

$\gamma \in [0, 1)$: discount factor

Markov Decision Process (MDP)

Discrete-time stochastic control process defined by $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$



$\mathcal{S}$: set of states

$\mathcal{A}$: set of actions

$\mathcal{P}$: transition probability distribution

$$\mathcal{P}_{ss'}^{a} = p(s_{t+1} = s' | s_t = s, a_t = a)$$

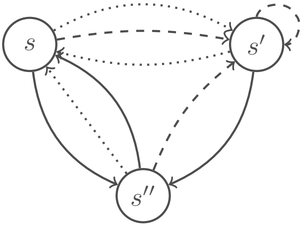
$\mathcal{R}$: reward function

$$\mathcal{R}_{ss'}^{a} = \mathbb{E}\left[r_t | s_t = s, a_t = a\right]$$

$\gamma \in [0, 1)$: discount factor

Markov Decision Process (MDP)

Discrete-time stochastic control process defined by $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$



$\mathcal{S}$: set of states

$\mathcal{A}$: set of actions

$\mathcal{P}$: transition probability distribution

$$\mathcal{P}_{ss'}^{a} = p(s_{t+1} = s' | s_t = s, a_t = a)$$

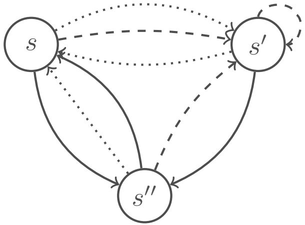
$\mathcal{R}$: reward function

$$\mathcal{R}_{ss'}^{a} = \mathbb{E}\left[r_t | s_t = s, a_t = a\right]$$

$\gamma \in [0, 1)$: discount factor

Markov Decision Process (MDP)

Discrete-time stochastic control process defined by $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$



$\mathcal{S}$: set of states

$\mathcal{A}$: set of actions

$\mathcal{P}$: transition probability distribution

$$\mathcal{P}^a_{ss'} = p(s_{t+1} = s' | s_t = s, a_t = a)$$

$\mathcal{R}$: reward function

$$\mathcal{R}^a_{ss'} = \mathbb{E}\left[r_t | s_t = s, a_t = a\right]$$

$\gamma \in [0, 1)$: discount factor

Markov Decision Process (MDP)

Goal: Take actions to maximize expected return over trajectories

Trajectory τ: path through state space up to a horizon

$$\tau = \langle s_0,\ a_0,\ r_0,\ s_1,\ a_1,\ r_1,\ s_2,\ a_2,\ \dots \rangle$$

Return R_t: cumulative future discounted rewards from s_t

$$\begin{aligned} R_t &= r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \cdots \\ &= \sum_{k=0}^{\infty} \gamma^k r_{t+k} \end{aligned}$$

Assuming infinite horizon:

- if $\gamma \geq 1$, R_t is unbounded
- if $0 \leq \gamma < 1$, R_t is well-defined and converges

Markov Decision Process (MDP)

Goal: Learn policy to maximize expected return over trajectories

Policy $\pi(s, a)$ represents action probabilities $p(a|s)$ in state s

- Deterministic policy: $a_t = \pi(s_t)$
- Stochastic policy: $a_t \sim \pi(a|s_t)$ $\leftarrow$ borrowing conditional prob notation

Episodic setting: Agent acts until a *terminal* state is reached

Learn parameterized policy π_θ with neural network weights θ

Network architecture follows action space $\mathcal{A}$:

- Discrete $\mathcal{A}$: softmax layer to represent $p(a|s_t)$
- Continuous $\mathcal{A}$: output μ and diagonal σ to sample $a_t \sim \mathcal{N}(\mu, \sigma)$

Derivative-free optimization

Objective: maximize $\mathbb{E}_{\tau \sim \theta}\left[R(\tau)\right]$

- Treat policy as a black box with parameters $\theta \in \mathbb{R}^d$
- Iteratively update θ to make good returns more likely

Cross-entropy method

Mannor et al (2003)

The Cross Entropy method for Fast Policy Search

- Initialize $\mu_0 \in \mathbb{R}^d$, $\sigma_0 \in \mathbb{R}^d$
- At each iteration i, draw L samples for $\theta^l \sim \mathcal{N}(\mu_i, \sigma_i^2)$
- Evaluate each trajectory τ^l using parameters θ^l
- Select the top ρ% of samples by $R(\tau^l)$ as the *elite set*
- Fit a new diagonal Gaussian to the elite set to obtain μ_{i+1}, σ_{i+1}
- At convergence, return final μ

+ No gradients needed; only forward pass
+ Converges quickly
+ Remarkably effective on many problems, e.g., Tetris

Evolution strategies

Salimans et al (2017)

Evolution Strategies as a Scalable Alternative to Reinforcement Learning

- Initialize $\theta_0 \in \mathbb{R}^d$
- At each iteration i, sample Gaussian noise $\epsilon^1, \ldots, \epsilon^L \sim \mathcal{N}(0, I)$
- Perturb θ_i with each ϵ^l to get $\tilde{\theta}^l = \theta_i + \sigma\epsilon^l$
- Evaluate each trajectory τ^l using parameters $\tilde{\theta}^l$ and update

$$\theta_{i+1} = \theta_i + \frac{\eta}{\sigma L} \sum_{l=1}^{L} R(\tau^l) \cdot \epsilon^l$$

+ No gradients needed; only forward pass
+ Easy to parallelize with low communication overhead
+ Competitive on Atari, OpenAI benchmarks

– Requires 3-10x more data

Policy gradient

Learn to increase expected return by gradient ascent over θ

$$\theta_{i+1} = \theta_i + \eta \, \nabla_\theta \, \mathbb{E}_\tau \left[R(\tau) \right]$$

Policy gradient: REINFORCE

Williams (1992)

Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning

Learn to increase expected return by gradient ascent over θ

$$\theta_{i+1} = \theta_i + \eta \, \nabla_\theta \, \mathbb{E}_\tau \left[R(\tau) \right]$$

$$\begin{aligned}
\nabla_\theta \, \mathbb{E}_\tau \left[R(\tau) \right] &= \sum_\tau R(\tau) \cdot \nabla_\theta \, p(\tau|\theta) \\
&= \sum_\tau R(\tau) \cdot \nabla_\theta \, p(\tau|\theta) \cdot \frac{p(\tau|\theta)}{p(\tau|\theta)} \\
&= \sum_\tau R(\tau) \cdot \nabla_\theta \log p(\tau|\theta) \cdot p(\tau|\theta) \\
&= \mathbb{E}_\tau \left[R(\tau) \cdot \nabla_\theta \log p(\tau|\theta) \right]
\end{aligned}$$

computed using sample averages

$$\approx \frac{1}{L} \sum_{l=1}^{L} R(\tau^l) \cdot \nabla_\theta \log p(\tau^l|\theta)$$

Policy gradient: REINFORCE

Williams (1992)

Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning

Learn to increase expected return by gradient ascent over θ

$$\begin{aligned} \theta_{i+1} &= \theta_i + \eta \, \nabla_\theta \, \mathbb{E}_\tau \left[R(\tau) \right] \\ &= \theta_i + \eta \, \mathbb{E}_\tau \left[R(\tau) \cdot \nabla_\theta \log p(\tau|\theta) \right] \end{aligned}$$

i.e., increase logprob of τ proportional to return $R(\tau)$

+ Unbiased estimator of gradient
+ Valid even if R is discontinuous or unknown
+ Only need $p(\tau|\theta)$ to be differentiable
− High variance, particularly for long trajectories
− Assigns credit to whole trajectory rather than individual actions
− Large number of samples needed

Variance reduction: baseline

If $R(\tau) \geq 0 \quad \forall \tau$

- Estimator always modifies density
- θ_i doesn't stabilize with fixed η

Reduce variance with a baseline

$$\begin{aligned}\nabla_\theta \mathbb{E}_\tau \left[R(\tau)\right] &= \nabla_\theta \mathbb{E}_\tau \left[R(\tau) - b\right] \\ &= \mathbb{E}_\tau \left[(R(\tau) - b) \cdot \nabla_\theta \log p(\tau|\theta)\right]\end{aligned}$$

Estimator remains unbiased

$$\mathbb{E}_\tau \, b \nabla_\theta \log p(\tau|\theta) \propto b \nabla_\theta \sum_\tau p(\tau|\theta) = b \nabla_\theta 1 = 0$$

Using $b = \mathbb{E}\left[R(\tau)\right]$ is a near-optimal choice (must be estimated)
i.e., increase logprob of τ proportional to how much return $R(\tau)$ is better than expected

Variance reduction

$$\nabla_\theta \log p(\tau|\theta) = \nabla_\theta \log \prod_{t=0}^{T-1} \pi_\theta(a_t|s_t)\mathcal{P}_{s_t s_{t+1}}^{a_t} = \nabla_\theta \sum_{t=0}^{T-1} \log \pi_\theta(a_t|s_t)$$

Rewards are distributed over trajectory

$$\nabla_\theta \, \mathbb{E}_\tau \left[R(\tau)\right] = \mathbb{E}_\tau \left[\left(\sum_{t=0}^{T-1} r_t \right) \sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t|s_t) \right]$$

Alternatively: use estimator for single reward and rewrite

$$\nabla_\theta \, \mathbb{E}_\tau \left[r_t\right] = \mathbb{E} \left[r_t \sum_{t'=0}^{t} \nabla_\theta \log \pi_\theta(a_{t'}|s_{t'}) \right]$$

$$\begin{aligned} \nabla_\theta \, \mathbb{E}_\tau \left[R(\tau)\right] &= \mathbb{E}_\tau \left[\sum_{t=0}^{T-1} r_t \sum_{t'=0}^{t} \nabla_\theta \log \pi_\theta(a_{t'}|s_{t'}) \right] \\ &= \mathbb{E}_\tau \left[\sum_{t'=0}^{T-1} \nabla_\theta \log \pi_\theta(a_{t'}|s_{t'}) \sum_{t=t'}^{T-1} r_t \right] \end{aligned}$$

Variance reduction

Can add baseline for each state

$$\nabla_\theta \mathbb{E}_\tau [R(\tau)] = \mathbb{E}_\tau \left[\sum_{t'=0}^{T-1} \nabla_\theta \log \pi_\theta(a_{t'}|s_{t'}) \left(\sum_{t=t'}^{T-1} r_t - b(s_{t'}) \right) \right]$$

Estimator remains unbiased as long as b doesn't depend on a_t

i.e., increase logprob of action $a_{t'}$ proportional to how much *future* return $R_{t'}(\tau) = \sum_{t=t'}^{T-1} r_t$ is better than expected

Value functions

State-value function $V(s)$:

Expected value of being in state s and following policy π

$$V^{\pi}(s) = \mathbb{E}_{\pi}\left[R_t | s_t = s\right]$$

State-action-value function $Q(s, a)$:

Expected value of taking action a from s and then following policy π

$$Q^{\pi}(s) = \mathbb{E}_{\pi}\left[R_t | s_t = s, a_t = a\right]$$

Advantage function $A(s, a)$:

Expected value of taking action a instead of following policy π

$$A^{\pi}(s, a) = Q^{\pi}(s, a) - V^{\pi}(s)$$

Value functions

$$\nabla_\theta \mathbb{E}_\tau [R(\tau)] = \mathbb{E}_\tau \left[\sum_{t'=0}^{T-1} \nabla_\theta \log \pi_\theta(a_{t'}|s_{t'}) \sum_{t=t'}^{T-1} r_t \right] + \text{baseline term}$$

$$= \sum_{t'=0}^{T-1} \mathbb{E}_{s_0 \dots a_t} \left[\nabla_\theta \log \pi_\theta(a_{t'}|s_{t'}) \, \mathbb{E}_{r_t \dots s_T} \left[\sum_{t=t'}^{T-1} r_t \right] \right]$$

$$Q^\pi(s_{t'}, a_{t'})$$

Substituting and reintroducing the baseline

$$\nabla_\theta \mathbb{E}_\tau [R(\tau)] = \mathbb{E}_\tau \left[\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t|s_t) \left(Q^\pi(s_t, a_t) - b(s_t) \right) \right]$$

Defining $b(s_t) = V^\pi(s_t)$ is near-optimal (Greenmith et al, 2004)

$$\nabla_\theta \mathbb{E}_\tau [R(\tau)] = \mathbb{E}_\tau \left[\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t|s_t) A^\pi(s_t, a_t) \right]$$

Value functions

$$\nabla_\theta \, \mathbb{E}_\tau \left[R(\tau) \right] = \mathbb{E}_\tau \left[\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t | s_t) A^\pi(s_t, a_t) \right]$$

Want $\mathbb{E}_\tau \left[A^\pi \right] = 0$ to keep variance low

- i.e., positive advantage for good actions, negative for bad actions

Don't know A^π, can use an *advantage estimator* $\hat{A}$, e.g.,

$$\hat{A}(s_t) = r_t + r_{t+1} + r_{t+2} + r_{t+3} + \ \dots \ - b(s_t)$$

+ Unbiased estimator
− High-variance single sample estimate
− Confounds the effect of actions $a_t, a_{t+1}, a_{t+2}, \dots$

Variance reduction

Use discounted return when calculating advantage

$$\hat{A}(s_t) = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \gamma^3 r_{t+3} + \ \dots \ - b(s_t)$$

$\gamma < 1$ to discount the effect of actions that are far in the future

To keep $\mathbb{E}_\tau\left[A^\pi\right] = 0$, also use discounted return when fitting baseline to $V^\pi(s_t)$

+ Use near-term rewards to evaluate actions
+ Lower variance
– Biased estimator

Variance reduction

Use V^{π} to estimate future rewards

$$
\begin{aligned}
\hat{A}(s_t) &= r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \gamma^3 r_{t+3} + \ \dots \ - b(s_t) \\
&= r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \gamma^3 V^{\pi}(s_{t+3}) - b(s_t) \\
&= r_t + \gamma r_{t+1} + \gamma^2 V^{\pi}(s_{t+2}) - b(s_t) \\
&= r_t + \gamma V^{\pi}(s_{t+1}) - b(s_t)
\end{aligned}
$$

Can share estimator of discounted V^{π} for future rewards and baseline, e.g.,

$$\hat{A}(s_t) = r_t + \gamma \hat{V}_t(s_{t+1}) - \hat{V}_t(s_t)$$

+ Explicit bias-variance tradeoff by choosing when to cut off
  - earlier: lower variance, higher bias
  - later: higher variance, lower bias

Actor-critic methods

$$\hat{A}(s_t) = r_t + \gamma \hat{V}(s_{t+1}) - \hat{V}(s_t)$$

Actor: policy $\pi_\theta(a_t|s_t)$
Critic: value function $\hat{V}(s_t)$

Simplified algorithm:

- Evaluate policy π_{θ_i} to collect samples $\langle s_t, R_t \rangle$
- Fit $\hat{V}$ by minimizing $\sum_n ||\hat{V}(s_n) - R_n||^2$
- Update policy parameters

$$\theta_{i+1} = \theta_i + \eta \, \mathbb{E}_\tau \left[\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t|s_t) \left(r_t + \gamma \hat{V}(s_{t+1}) - \hat{V}(s_t) \right) \right]$$

Resources

- *Reinforcement Learning: An Introduction* (Sutton & Barto, 2017)
  http://incompleteideas.net/book/bookdraft2017nov5.pdf
- OpenAI Spinning Up: code, environment, papers
  https://spinningup.openai.com/
- Berkeley RL course materials
  http://rail.eecs.berkeley.edu/deeprlcourse/

Policy gradients in NLP

- Directly optimize against non-differentiable metrics
  - e.g., text generation metrics like BLEU, ROUGE, CIDEr etc
  - Learned measures, e.g., neural teachers
- Jointly learn latent structure
  - Sample from policy over latent variables
  - Back-propagate loss to policy network

Non-differentiable metrics

Rennie et al (2016)

Self-critical Sequence Training for Image Captioning

Optimize against CIDEr, BLEU, ROUGE, etc using REINFORCE

Use score of greedy decoding as a baseline

i.e., encourage *exploration* of new words that improve rewards

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.

Non-differentiable metrics

Rennie et al (2016)

Self-critical Sequence Training for Image Captioning

Optimize against CIDEr, BLEU, ROUGE, etc using REINFORCE

Use score of greedy decoding as a baseline
i.e., encourage *exploration* of new words that improve rewards

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**

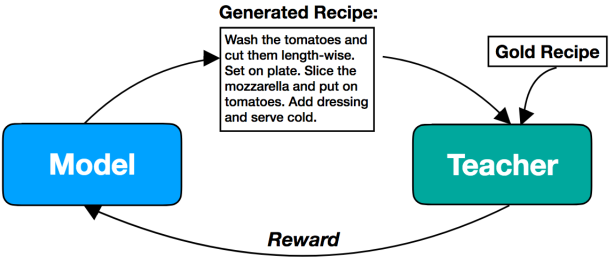
Non-differentiable metrics

Bosselut et al (2018)

Discourse-Aware Neural Rewards for Coherent Text Generation

Train 'teacher' network to score how well-ordered a text is

Incorporate teacher scores as a reward for text generation



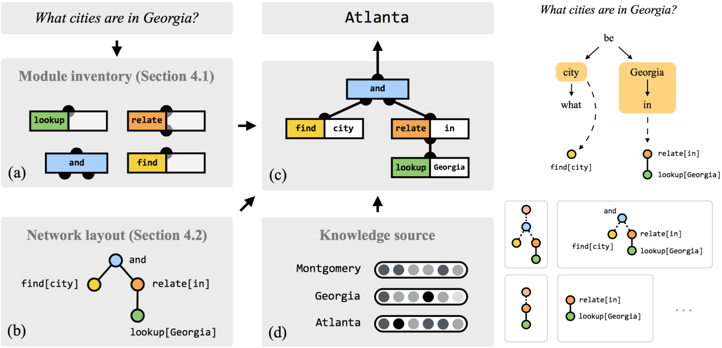
Latent structure

Andreas et al (2016)

Learning to Compose Neural Networks for Question Answering

Dynamically assemble a network for each question from modules

Sample module layout from policy and backpropagate answer errors with REINFORCE



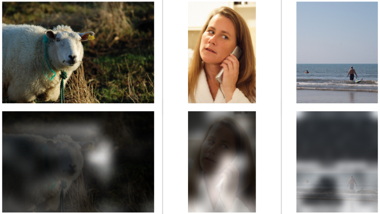
Latent structure

Andreas et al (2016)

Learning to Compose Neural Networks for Question Answering

Dynamically assemble a network for each question from modules

Sample module layout from policy and backpropagate answer errors with REINFORCE



What is in the sheep's ear?	*What color is she wearing?*	*What is the man dragging?*
`(describe[what] (and find[sheep] find[ear]))`	`(describe[color] find[wear])`	`(describe[what] find[man])`
tag	**white**	**boat** (board)

Is Key Largo an island?

`(exists (and lookup[key-largo] find[island]))`

yes: correct

What national parks are in Florida?

`(and find[park] (relate[in] lookup[florida]))`

everglades: correct

What are some beaches in Florida?

`(exists (and lookup[beach] (relate[in] lookup[florida])))`

yes (daytona-beach): wrong parse

What beach city is there in Florida?

`(and lookup[beach] lookup[city] (relate[in] lookup[florida]))`

[none] (daytona-beach): wrong module behavior

Latent structure

Lei et al (2016)

Rationalizing Neural Predictions

Define network to propose sequences of words from input text as rationales for classification

Sample rationales during inference and convey rewards with REINFORCE

Review

the beer was n't what i expected, and i'm not sure it's "true to style", but i thought it was delicious. **a very pleasant ruby red-amber color** with a relatively brilliant finish, but a limited amount of carbonation, from the look of it. aroma is what i think an amber ale should be - a nice blend of caramel and happiness bound together.

Ratings

Look: 5 stars

Smell: 4 stars

Latent structure

Lei et al (2016)

Rationalizing Neural Predictions

Define network to propose sequences of words from input text as rationales for classification

Sample rationales during inference and convey rewards with REINFORCE

a beer that is not sold in my neck of the woods , but managed to get while on a roadtrip . poured into an imperial pint glass with a generous head that sustained life throughout . nothing out of the ordinary here , but a good brew still . body was kind of heavy , but not thick . the hop smell was excellent and enticing . very drinkable

---

very dark beer . pours a nice finger and a half of creamy foam and stays throughout the beer . smells of coffee and roasted malt . has a major coffee-like taste with hints of chocolate . if you like black coffee , you will love this porter . creamy smooth mouthfeel and definitely gets smoother on the palate once it warms . it 's an ok porter but i feel there are much better one 's out there .

---

i really did not like this . it just seemed extremely watery . i dont ' think this had any carbonation whatsoever . maybe it was flat , who knows ? but even if i got a bad brew i do n't see how this would possibly be something i 'd get time and time again . i could taste the hops towards the middle , but the beer got pretty nasty towards the bottom . i would never drink this again , unless it was free . i 'm kind of upset i bought this .

---

a : poured a nice dark brown with a tan colored head about half an inch thick , nice red/garnet accents when held to the light . little clumps of lacing all around the glass , not too shabby . not terribly impressive though s : smells like a more guinness-y guinness really , there are some roasted malts there , signature guinness smells , less burnt though , a little bit of chocolate … … m : relatively thick , it is n't an export stout or imperial stout , but still is pretty hefty in the mouth , very smooth , not much carbonation . not too shabby d : not quite as drinkable as the draught , but still not too bad . i could easily see drinking a few of these .