(Deep) Inverse Reinforcement Learning

Zheng Zhang

NYU Shanghai

zz@nyu.edu

March 14, 2018

Overview

Techniques Involved

A lot!

- Maximum entropy model and Max-Ent RL
- Energy based model (EBM) and soft optimal policy
- Dealing with partition function; log likelihood gradient
- (Biased) importance sampling
- Adversarial training (and GAN) and saddle point optimization

All in the name of a different way of doing supervised imitation learning!

RL beyond robots and games

RL beyond robots and games

Reinforcement gaol:

- Focuses on rewards over "trajectories": $\theta^* = \text{argmax}_\theta \mathbb{E}_{\tau \sim \pi(\theta)}[r(\tau)]$
- ... and the unbiased PG:
  $\nabla J(\theta) = \mathbb{E}_{p(a_t, s_t)}[\nabla \log \pi_\theta(a_t|s_t) \cdot \sum_t r(a_t, s_t)]$

- Disconnected to the rest of the more common ML/AI problems?

Computer vision

The usual setup: single object classification

- $D = \{x^i, y^i\}_{i=1}^N$, a parametrized model: $p_\theta(y|x) : R^{H \times W \times 3} \rightarrow R^{|C|}$
- Cross entropy loss of a multinomial distribution
- Minimize the loss: $\nabla J(\theta) \approx -\frac{1}{N} \sum_i \nabla \log p_\theta(y^i|x^i) \cdot \mathbb{I}(y^i = \hat{y}^i)$

Computer vision

The usual setup: single object classification

- $D = \{x^i, y^i\}_{i=1}^N$, a parametrized model: $p_\theta(y|x) : R^{H \times W \times 3} \rightarrow R^{|C|}$
- Cross entropy loss of a multinomial distribution
- Minimize the loss: $\nabla J(\theta) \approx -\frac{1}{N} \sum_i \nabla \log p_\theta(y^i|x^i) \cdot \mathbb{I}(y^i = \hat{y}^i)$

The RL perspective

- The trajectory length $T = 1$
- $r(y^i, x^i) = 1$ if "pick" the correct class and 0 otherwise
- Maximize the reward:
  $\nabla J(\theta) = \mathbb{E}[\nabla \log \pi(y^i|x^i) r(y^i, x^i)] \approx \frac{1}{N} \sum_i \nabla \log \pi(y^i|x^i) \cdot \mathbb{I}(y^i = \hat{y}^i)$

The model p_θ is the policy π: these two perspectives are equivalent.
The policy is *hard*: some classes are closer in ontology (not captured)

Computer vision (cont.)

- Multiple objects detection is to process regions of x *in parallel*.
- But understanding a real image is far more complex
  - Requires a trajectory of glimpses (to encode semantics)
  - ...more likely a structured set of glimpses (e.g. a tree):

Still far away from solving the vision problem!

Natural Language Processing

Setup

- $D = \{Y^i, X^i\}_{i=1}^N$, $Y^i = (y_1^i, \cdots, y_{T_i}^i)$, $y \in \mathcal{V}$, X^i is a "message" (an image, a source (foreign) sentence, a response to a question etc.)
- Learn a parametrized model: $\theta^* = \text{argmax}_\theta p(Y|X)$

Natural Language Processing

Setup

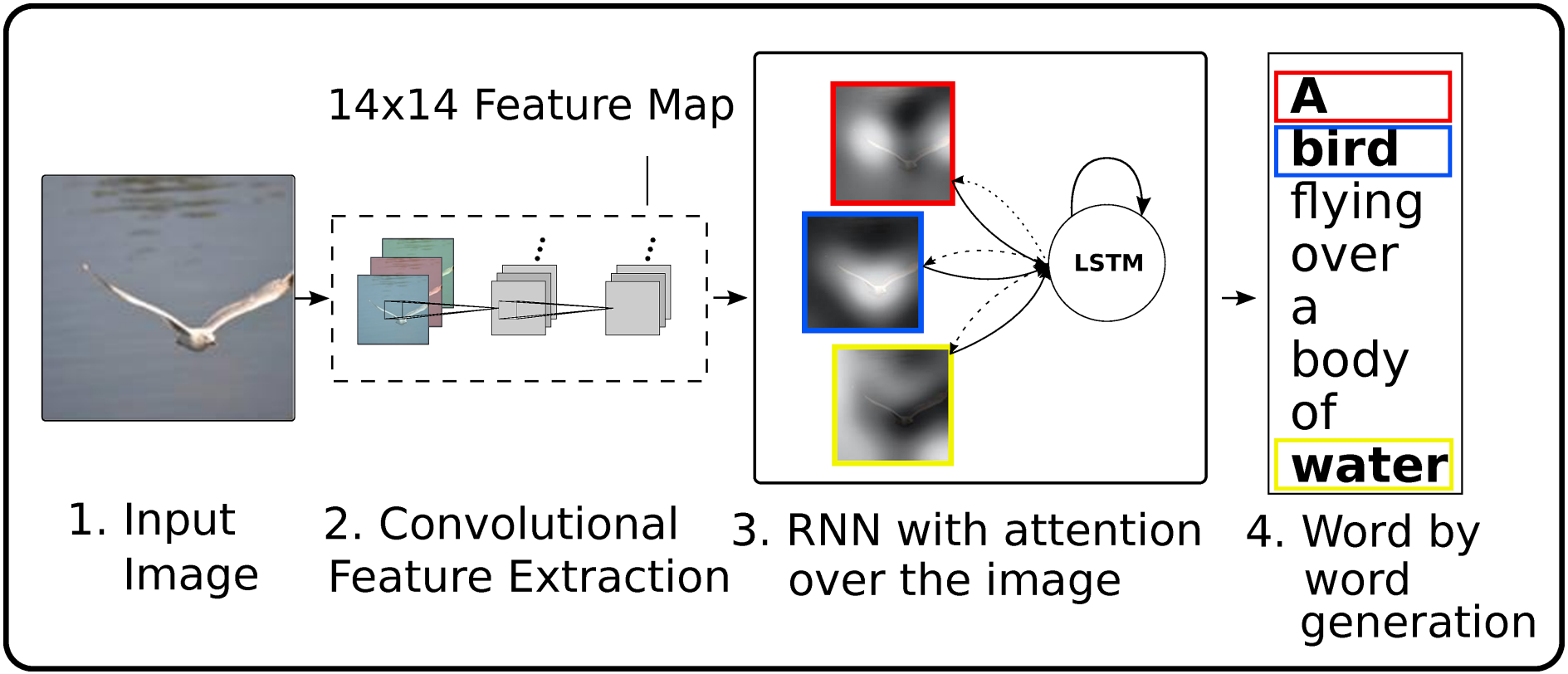
- $D = \{Y^i, X^i\}_{i=1}^N$, $Y^i = (y_1^i, \cdots, y_{T_i}^i)$, $y \in \mathcal{V}$, X^i is a "message" (an image, a source (foreign) sentence, a response to a question etc.)
- Learn a parametrized model: $\theta^* = \text{argmax}_\theta p(Y|X)$

The RL perspective: imitation learning

- The state $s_t = f(y_{<=t}, X)$: the message, the words uttered so far
- The action a_t is to pick the next work: y_{t+1}
- The policy: $\pi(a_t = y_{t+1}|s_t = y_{<=t}, X)$
- Implementation:
  - $s_t \equiv h_t = \text{RNN}(h_{t-1}, y_t, \text{Enc}(X))$
  - $a_t \equiv y_{t+1} \sim \text{Softmax}(\text{MLP}(h_t))$

CV/NLP example

Show, Attend and Tell: Neural Image Caption Generation with Visual Attention, Kevin Xu et al.

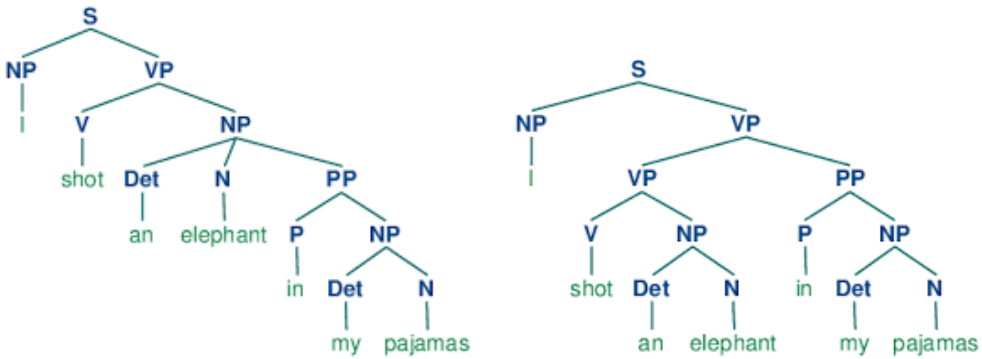


Natural Language Processing (cont.)

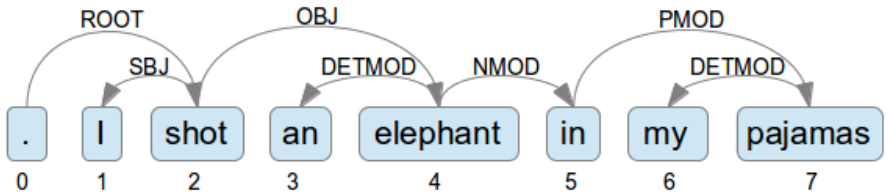
NLP is much more than just sequences; it's about structure:

- Generation includes generating a latent syntactic tree;
- Understanding recovers a tree
- Goes beyond the typical hierarchical RL (usually two levels).

Parse (constituency) tree



Dependency grammar



An exciting (new) research field!

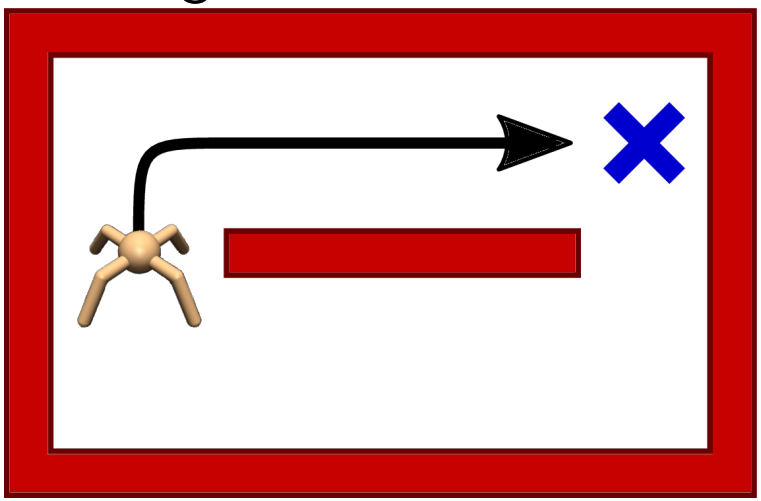
Max-Ent: applications and models

Maximum Entropy Model

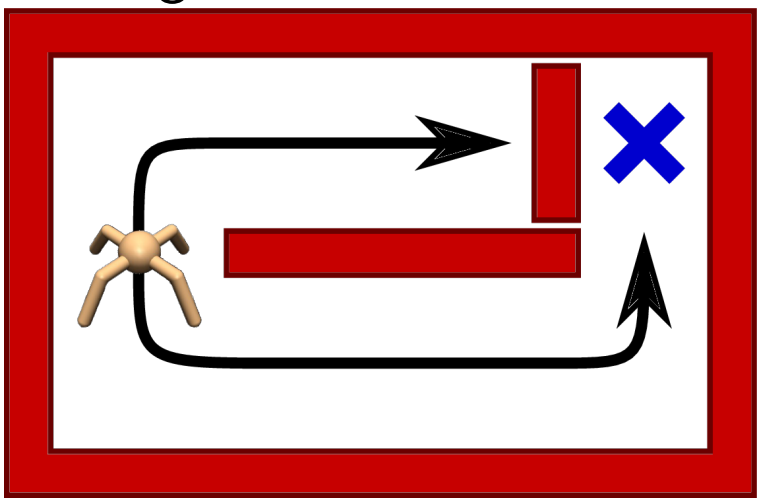
Ed. T. Jaynes (1957):
maximally non-committal with regard to missing information
Or, in plain English: *keep your options open!* (Occam's Razor, Bayes' Theorem)

See Learning Diverse Skills via Maximum Entropy Deep Reinforcement Learning

Training



Testing



More videos; relevant segment in CS294.

MaxEnt in recognition task

Problem set up: $p(\mathcal{Y}|x)$, $\mathcal{Y} = (y_1, \cdots, y_{|\mathcal{Y}|})$, $y_i \in \mathcal{C}$

No predefined order, an item can appear multiple times.

Sean et al.: Loss Functions for **Multiset** Prediction

[dog, person]

[person, person, dog, person, person]

[cat, dog]

Free-set: $\mathcal{Y}_t \leftarrow \mathcal{Y}_{t-1} \backslash \{\hat{y}_t\}$; $\{\hat{y}_t\}$ is prediction made by π_θ.

Oracle policy: (maximum entropy)

$$\pi_*(y_t|\hat{y}_{<t}, x) = \begin{cases} \frac{1}{|\mathcal{Y}_t|}, & \text{if } y_t \in \mathcal{Y}_t \\ 0, & \text{otherwise} \end{cases}$$

Minimize $\mathrm{KL}(\pi_\theta || \pi_*)$.

MaxEnt RL

Vanilla RL: $J(\theta) = \mathbb{E}_{\tau \sim \pi(\theta)}[r(\tau)]$
Adding exploration: $J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta}[r(\tau)] + H(\pi_\theta)$
But, *why*, and *how*?

MaxEnt RL

Vanilla RL: $J(\theta) = \mathbb{E}_{\tau\sim\pi(\theta)}[r(\tau)]$
Adding exploration: $J(\theta) = \mathbb{E}_{\tau\sim\pi_\theta}[r(\tau)] + H(\pi_\theta)$
But, *why*, and *how*?

Assume:
the policy family takes form of energy function (*soft optimal policy*):

$$\pi(\tau) = \exp(r(\tau))/Z$$

Minimize:
KL divergence between the current policy and that of the oracle:

$$\begin{aligned} J(\theta) &= -\mathrm{KL}(\pi_\theta(\tau)||\pi^*(\tau)) && \text{(1a)} \\ &= -\mathbb{E}_{\tau\sim\pi_\theta}[\log \pi_\theta - \log \pi^*] && \text{(1b)} \\ &= H(\pi_\theta) + \mathbb{E}_{\tau\sim\pi_\theta}[r(\tau) - \log Z] && \text{(1c)} \\ &= \mathbb{E}_{\tau\sim\pi_\theta}[r(\tau)] + H(\pi_\theta) + C && \text{(1d)} \end{aligned}$$

Policy in energy function form leads to exploration!
Question: doesn't reverse KL drop mode still?

Reward revisted

The trouble of an explicit reward function $r(\tau)$:

- Often not clear what it should be (e.g. pouring water into a cup)
- $\text{Var}(\nabla J(\theta)) \propto ||r(\tau)||^2$: big impact on the variance of gradient estimation
- Often at the end of the trajectory – sparse!

Solution, learn it (parameter ψ)!

- $p_\psi(\tau) = \exp(r_\psi(\tau))/Z$
- ("better actions are exponentially more likely")
- Implies a maximum entropy policy: $J(\theta) = \mathbb{E}_{\tau\sim\pi_\theta}[r(\tau)] + H(\pi_\theta)$
- Can output at every time step too! (dense)

How: maximum likelihood estimation of expert demonstration $\pi^*(\tau)$
(Imitation learning, but using reward function as a proxy)

Partition function and the log-likelihood gradient

M.L.E $\log p(x;\psi) = \log \frac{\tilde{p}(x;\psi)}{Z(\psi)}$, and we only have access to $\tilde{p}(x,\psi)$
Then: $\nabla_\psi \log p(x;\psi) = \nabla_\psi \log \tilde{p}(x;\psi) - \nabla_\psi \log Z(\psi)$
We have: $\nabla_\psi \log Z(\psi) = \frac{\nabla_\psi Z}{Z}$

$$\begin{aligned}
\nabla_\psi Z &= \nabla_\psi \sum \tilde{p}(x) \\
&= \sum \nabla_\psi \exp(\log \tilde{p}(x)) \\
&= \sum \exp(\log \tilde{p}(x)) \nabla_\psi \log \tilde{p}(x) \\
&= \sum \tilde{p}(x) \nabla_\psi \log \tilde{p}(x)
\end{aligned}$$

Therefore: $\nabla_\psi \log Z(\psi) = \frac{\nabla_\psi Z}{Z} = \sum \frac{\tilde{p}(x)}{Z} \nabla_\psi \log \tilde{p}(x) = \mathbb{E}[\nabla_\psi \log \tilde{p}(x)]$
Thus:

$$\nabla_\psi \log p(x;\psi) = \nabla_\psi \log \tilde{p}(x;\psi) - \mathbb{E}[\nabla_\psi \log \tilde{p}(x)]$$

Plug into reward function

Previously: $\nabla_\psi \log p(x;\psi) = \nabla_\psi \log \tilde{p}(x;\psi) - \mathbb{E}[\nabla_\psi \log \tilde{p}(x)]$
And, in our case, $\tilde{p}(x;\psi) \triangleq \exp(r_\psi(\tau))$

We can learn by:

$$\nabla_\psi \mathcal{L} = \mathbb{E}_{\tau\sim\pi^*}[\nabla_\psi r_\psi(\tau)] - \mathbb{E}_{\tau\sim\pi_\theta}[\nabla_\psi r_\psi(\tau)]$$

Can be estimated by:

$$\nabla_\psi \mathcal{L} \approx \frac{1}{N}\sum_{\tau_i\sim\pi^*} \nabla_\psi r_\psi(\tau_i) - \frac{1}{M}\sum_{\tau_j\sim\pi_\theta} \nabla_\psi r_\psi(\tau_j)$$

The 1st term: samples from *expert demonstration*
The 2nd term: samples from *current soft optimal policy*

The full IRL algorithm

Iterate:

- Fix r_ψ, learn π_θ: $J(\theta) = \mathbb{E}_{\tau\sim\pi_\theta}[r_\psi(\tau)] + H(\pi_\theta)$
- Fix π_θ, learn r_ψ: $\nabla_\psi \mathcal{L} \approx \frac{1}{N}\sum_{\tau_i\sim\pi^*} \nabla_\psi r_\psi(\tau_i) - \frac{1}{M}\sum_{\tau_j\sim\pi_\theta} \nabla_\psi r_\psi(\tau_j)$

Letting the policy settle can be expensive!
What if we *improve* π_θ for a step and go directly to estimate ψ?

Problem:
$\frac{1}{M}\sum_{\tau_j\sim\pi_\theta} \nabla_\psi r_\psi(\tau_j)$ is biased, π_θ has not converged yet...

Solution: (biased) importance sampling.

Biased Importance Sampling

Importance sampling:
$\mathbb{E}_p[f(x)] = \mathbb{E}_q[\frac{p(x)f(x)}{q(x)}]$

Biased importance sampling:
$p(x) \propto \tilde{p}(x)$, $q(x) \propto \tilde{q}(x)$, and $w = \frac{\tilde{p}}{\tilde{q}}$:
$\mathbb{E}_p[f(x)] = \mathbb{E}_q[\frac{p(x)f(x)}{q(x)}] \approx \frac{1}{\sum_j w_j} \sum_j w_j f(x_j)$ is asymptotically unbiased

In our setting:

- $x_j \triangleq \tau_j$
- $f(x_j) \triangleq \nabla r_\psi(\tau_j)$
- $\tilde{p}(x_j) \triangleq \exp(r_\psi(\tau_j))$
- $\tilde{q}(x_j) \equiv q(x_j) \triangleq \pi_\theta(\tau_j)$
- $w_j = \frac{\exp(r_\psi(\tau_j))}{\pi_(\tau_j)}$

$\nabla_\psi \mathcal{L} \approx \frac{1}{N} \sum_{\tau_i \sim \pi^*} \nabla_\psi r_\psi(\tau_i) - \frac{1}{\sum_j w_j} \sum_{\tau_j \sim \pi_\theta} w_j \nabla_\psi r_\psi(\tau_j)$

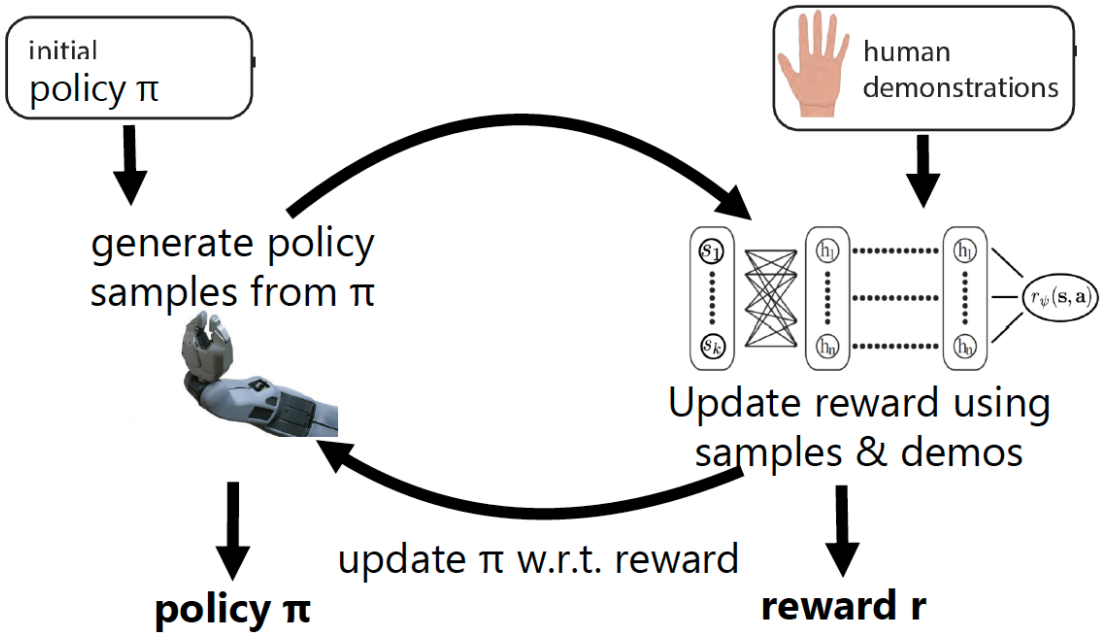
Guided cost learning

Iterate –

- Fix r_ψ, *improve* π_θ: $J(\theta) = \mathbb{E}_{\tau\sim\pi_\theta}[r_\psi(\tau)] + H(\pi_\theta)$
- Fix π_θ, learn r_ψ: $w_j = \frac{\exp(r_\psi(\tau_j))}{\pi_(\tau_j)}$

  $\nabla_\psi \mathcal{L} \approx \frac{1}{N}\sum_{\tau_i\sim\pi^*} \nabla_\psi r_\psi(\tau_i) - \frac{1}{\sum_j w_j}\sum_{\tau_j\sim\pi_\theta} w_j \nabla_\psi r_\psi(\tau_j)$

Finn et al. ICML'16. CS 294 video segment



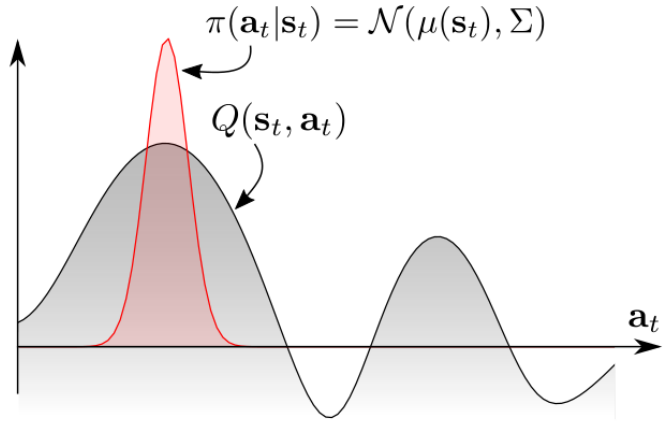
Review

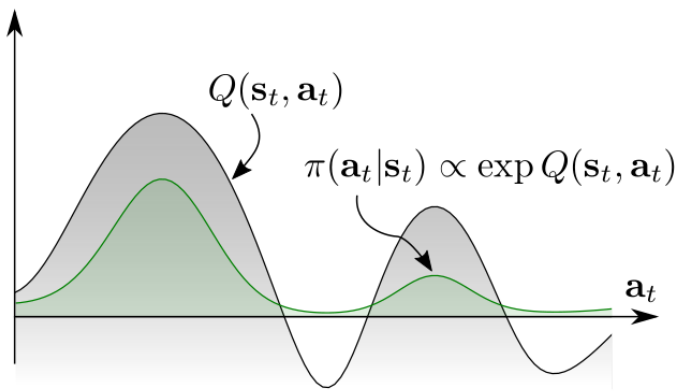
Ignore explicit rewards, rely on a collection of expert demonstrations $D \sim \pi^*$

Not to imitate *directly*. Instead:

- $p(\tau) = \exp(r_\psi(\tau))/Z$, energy based model: M.L.E for π^*
- Alternate:
  - Improve a max-ent policy π_θ
  - Learn reward function r_ψ (use log-likelihood gradient for Z)
- Use importance sampling to deal with distribution bias

All these hassles for what? Extrapolating with an energy envelop:





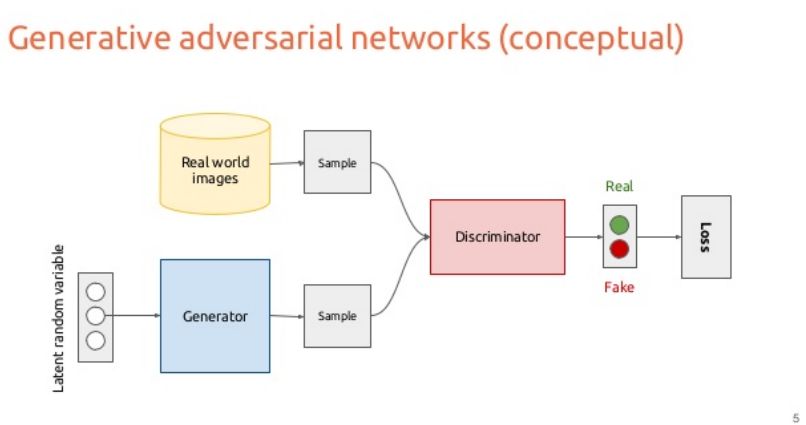
Connection to adversarial training

Generative Adversarial Network

GAN is *hot*!
Strongly recommend Ian Goodfellow's NIPS'16 tutorial video and paper

Min-Max game:

$$J^{(D)}(\theta^D, \theta^{(G)}) = -\frac{1}{2}\mathbb{E}_x \log D(x) - \frac{1}{2}\mathbb{E}_z \log(1 - D(G(z)))$$
$$J^{(G)}(\theta^{(D)}, \theta^{(G)}) = -J^{(D)}(\theta^{(D)}, \theta^{(G)})$$

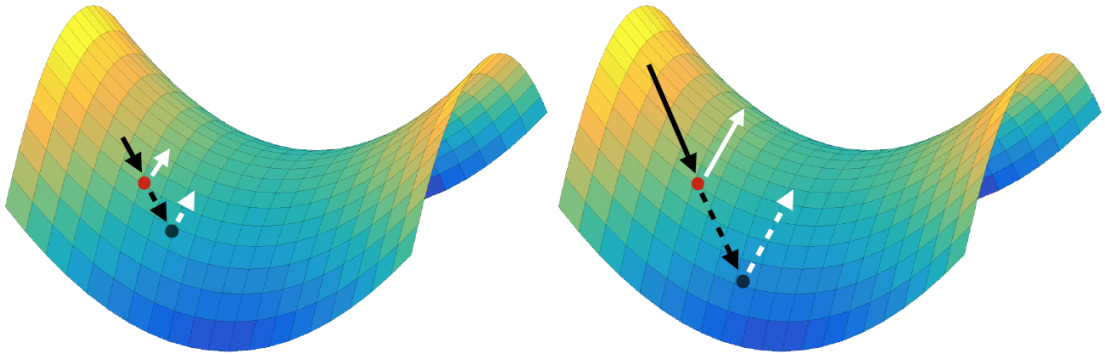


GAN and IRL compared

GAN	IRL
G: $p(G(z))$ close to $p(x)$	π_θ: π_θ close to π^*
D: discriminate $G(z)$ against x	r_ψ: promote likelihood of $\tau \sim \pi^*$ against π_θ
$D^*(x) = \frac{p_{data}(x)}{p_{data}(x)+p_{model}(x)}$	$D^*(\tau) = \frac{\frac{1}{Z}\exp(r_\psi)}{\frac{1}{Z}\exp(r_\psi)+\pi_\theta}$

Any improvement of training GAN will also improve IRL

c.f. Stabilizing Adversarial Nets With Prediction Methods, Abhay et. al.



(a) (b)

Figure 2: A schematic depiction of the prediction method. When the minimization step is powerful and moves the iterates a long distance, the prediction step (dotted black arrow) causes the maximization update to be calculated further down the loss surface, resulting in a more dramatic maximization update. In this way, prediction methods prevent the maximization step from getting overpowered by the minimization update.

Misc.

Model implementation:

- Use RNN (e.g. GRU, LSTM) to summarize past states into $h_t = \text{RNN}(h_{t-1}, s_t; \theta_{\text{RNN}})$
- Map h_t into action a_t, e.g. $\pi_\theta(a_t, s_t) = \text{Softmax}(\text{MLP}(h_t); \theta_{\text{MLP}})$
- Map h_t into $r(t)$, e.g. $r_\psi(t) = \tanh(\text{MLP}(h_t; \psi))$
- Alternate between optimizing $\{\theta_{\text{RNN}}, \theta_{\text{MLP}}\}$ and ψ

More to be explored:

- Directly use GAN with RL exploits richer label information (c.f. SeqGAN: Sequence Generative Adversarial Nets with Policy Gradient)
- Complementing IRL?
- Applying new saddle point optimization techniques?
- Most interesting problems require the use of expert demonstrations *and* figuring out latent structures (trajectories) (e.g. language/image understanding)

An aside: optimal point of GAN

See proof here

$$
\begin{aligned}
J(D) &= \mathbb{E}_x[\log D(x)] + \mathbb{E}_z[\log(1 - D(G(z)))] \\
&= \int_x p_{data}(x) \log D(x) dx + \int_z p(z) \log(1 - D(G(z))) dx \\
&= \int_x p_{data}(x) \log D(x) dx + \int_x p_{model}(x) \log(1 - D(x)) dz \\
&= \int_x p_{data}(x) \log D(x) + p_{model}(x) \log(1 - D(x)) dx
\end{aligned}
$$

For $f(y) = a \log y + b \log(1 - y)$, and $a, b \in (0, 1)$,
The maximum is $\frac{a}{a+b}$

$D^*(x) = \frac{p_{data}(x)}{p_{data}(x) + p_{model}(x)}$
When $p_{data}(x) = p_{model}(x)$ this is $1/2$, meaning D is perfectly confused.

Remarks

Techniques Involved:

- Maximum entropy model and Max-Ent RL
- Energy based model (EBM) and soft optimal policy
- Dealing with partition function; log likelihood gradient
- (Biased) importance sampling
- Adversarial training (and GAN) and saddle point optimization

Discussion: labels vs. rewards

The End