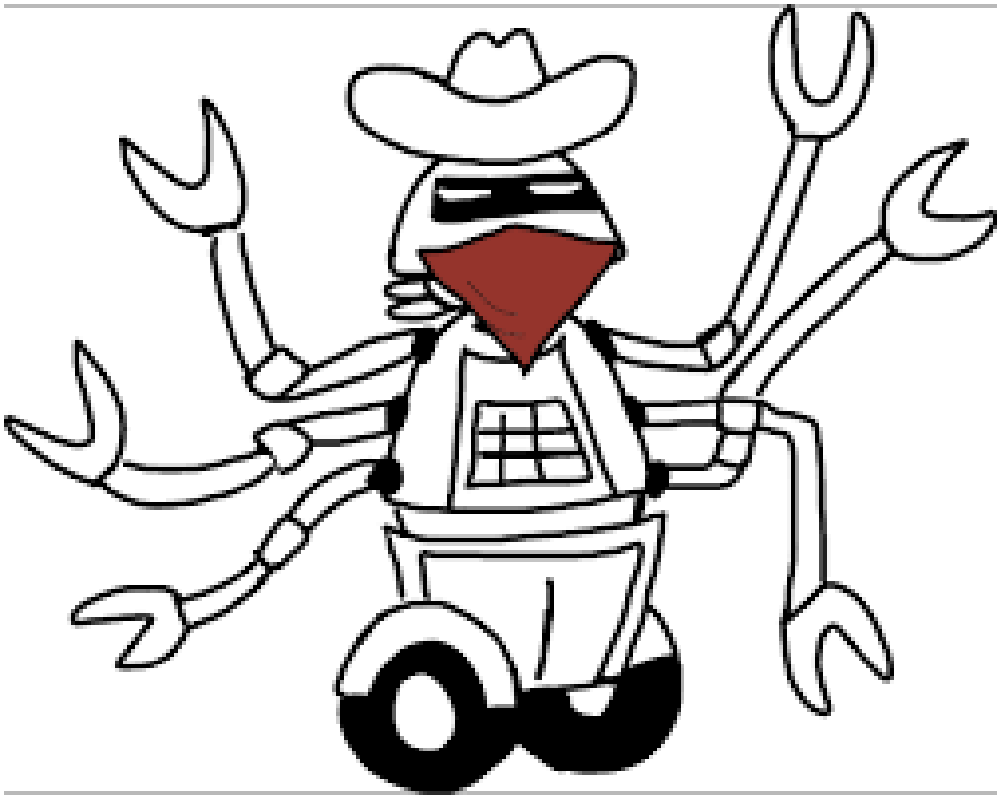
Practical RL

week 5, spring'18

Bandits, exploration, production hacks

Multi-armed bandits



Multi-armed bandits

Multi-armed bandits: simplest

Simple case: no different “states”, just N actions

Exploration: figure out which action is best overall; take as few bad actions as you can

Multi-armed bandits: contextual

A simplified MDP with only one step

Why bandits: it's simpler to explain math here,
Formulae are, like, 50% shorter
(we will generalize to MDP in the second half)

What is: contextual bandit

Examples:

- banner ads (RTB)
- recommendations
- medical treatment

Basicaly it's 1-step MDP where

- G(s,a) = r(s,a)
- Q(s,a) = E r(s,a)
- All formulae are 50% shorter

How to measure exploration

Ideas?

How to measure exploration

Bad idea: by the sound of the name
Good idea: by $$$ it brought/lost you

Regret of policy π(a|s)**:**
Consider an optimal policy, π*(a|s)

Regret = sum over training time [optimal – yours]

$$\eta = \sum_t \underset{s,a\sim\pi^*}{E} r(s,a) - \underset{s,a\sim\pi_t}{E} r(s,a)$$

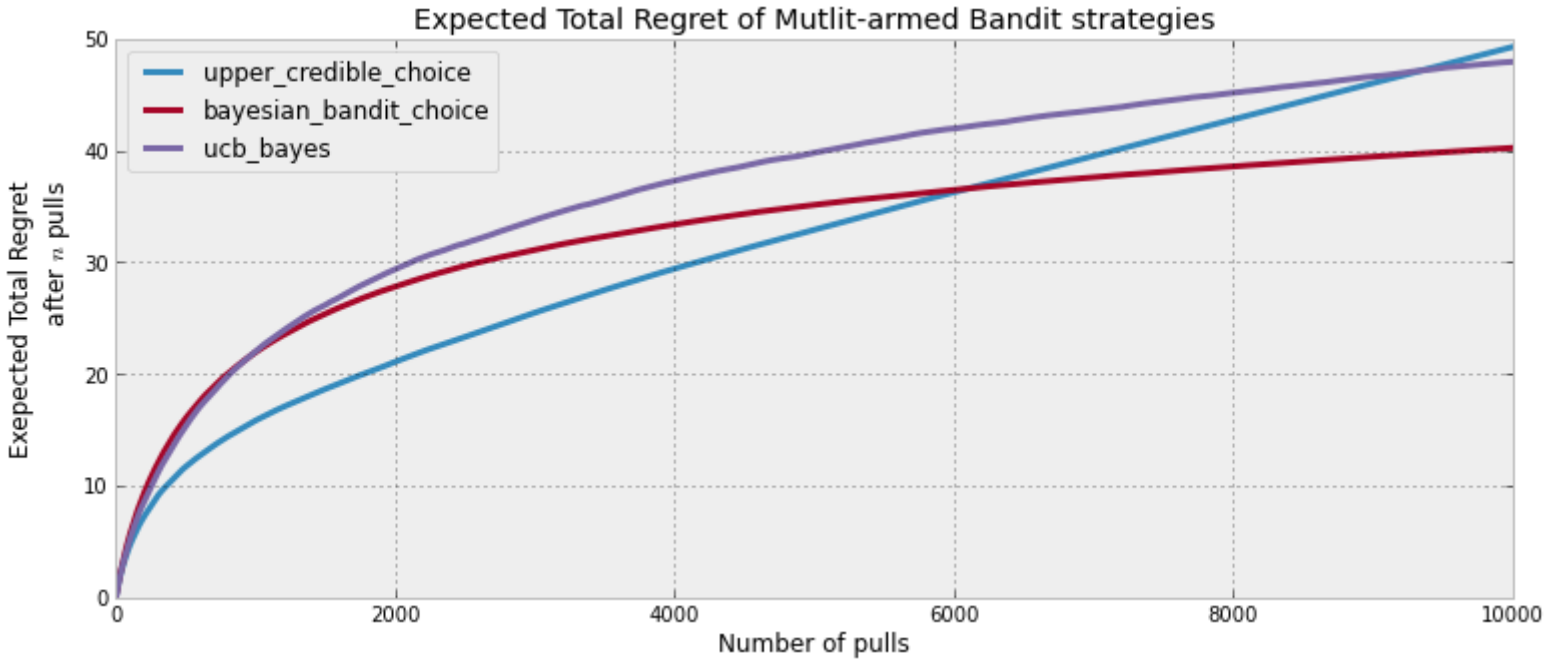
Finite horizon: t < max_t Infinite horizon: t → inf

How to measure exploration

Bad idea: by the sound of the name
Good idea: by $$$ it brought/lost you

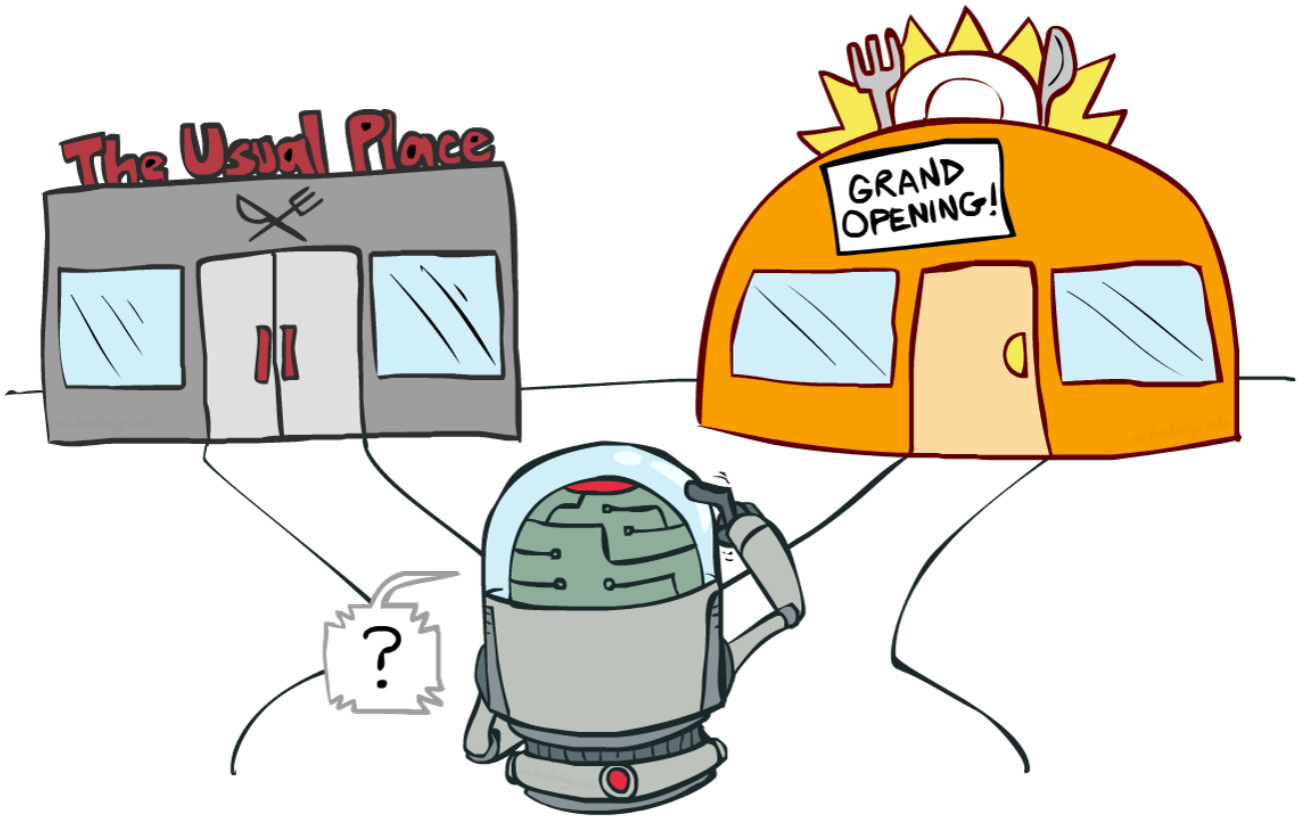
Regret of policy π(a|s)**:**

Regret per tick = optimal – yours



Exploration Vs Exploitation

What exploration strategies did we use before?



Exploration strategies so far...

Strategies:

- **ε-greedy**
  - With probability ε take a uniformly random action;
  - Otherwise take optimal action.
- **Boltzman**
  - Pick action proportionally to transformed Qvalues

$$P(a) = softmax\left(\frac{Q(s,a)}{std}\right)$$

- **Optimistic initialization**
  - start from high initial Q(s,a) for all states/actions
  - good for tabular algorithms, hard to approximate

Exploration strategies so far...

Strategies:

- **ε-greedy**
  - With probability ε take a uniformly random action;
  - Otherwise take optimal action.

Say, we use ε-greedy with const **ε = 0.25** on top of q-learning to play a videogame.

What can you say about regret?

$$\eta = \sum_t \underset{s,a\sim\pi^*}{E} r(s,a) - \underset{s,a\sim\pi_t}{E} r(s,a)$$

Exploration strategies so far...

Strategies:

- **ε-greedy**
  - With probability ε take a uniformly random action;
  - Otherwise take optimal action.

  Say, we use ε-greedy with const **ε = 0.25** on top of q-learning to play a videogame.

  Regret grows linearly over time!

  Agent always acts suboptimally due to ε

Exploration over time

Idea:

If you want to converge to optimal policy,
you need to gradually reduce exploration

Example:

Initialize ε-greedy ε = 0.5, then gradually reduce it

- If $\varepsilon \rightarrow 0$, it's **greedy in the limit**
- Be careful with non-stationary environments

How many lucky random actions it takes to

- Apply medical treatment
- Control robots
- Invent efficient VAE training

Except humans can learn these in less than a lifetime

How many lucky random actions it takes to

- Apply medical treatment
- Control robots
- Invent efficient VAE training

We humans explore not with e-greedy policy!

BTW how humans explore?

Whether some new particles violate physics

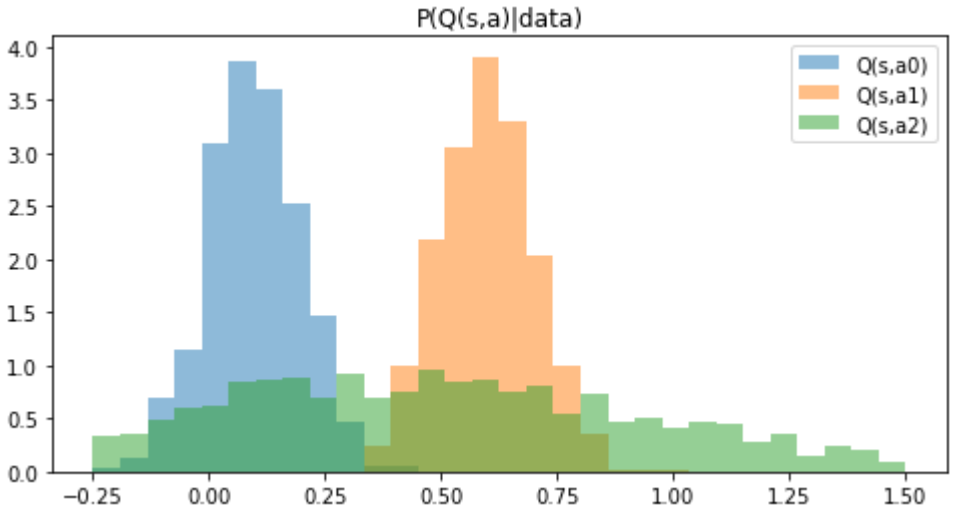
Vs

Whether you still can't fly by pulling your hair up

Uncertainty in returns

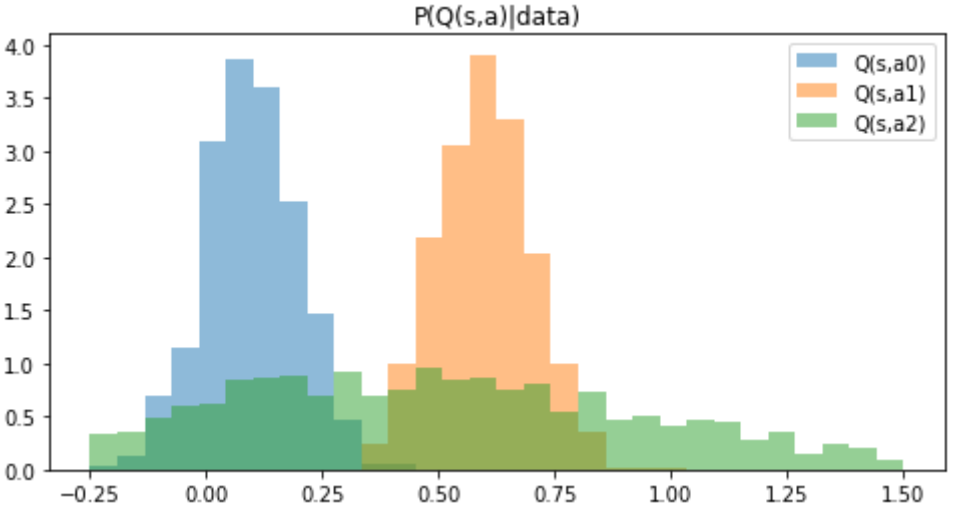
We want to try actions if we believe there's a chance they turn out optimal.

Idea: let's model how certain we are that Q(s,a) is what we predicted



Thompson sampling

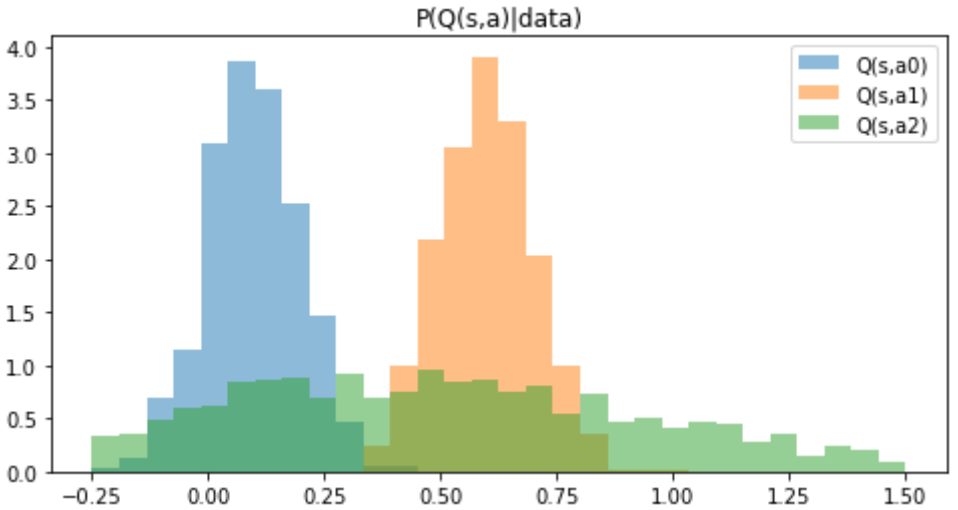
- Policy:
  - sample **once** from each Q distribution
  - take argmax over samples
  - which actions will be taken?



Thompson sampling

- Policy:
  - sample **once** from each Q distribution
  - take argmax over samples
  - which actions will be taken?

Takes a1 with p ~ 0.65, a2 with p ~ 0.35, a0 ~ never



Optimism in face of uncertainty

Idea:

Prioritize actions with uncertain outcomes!

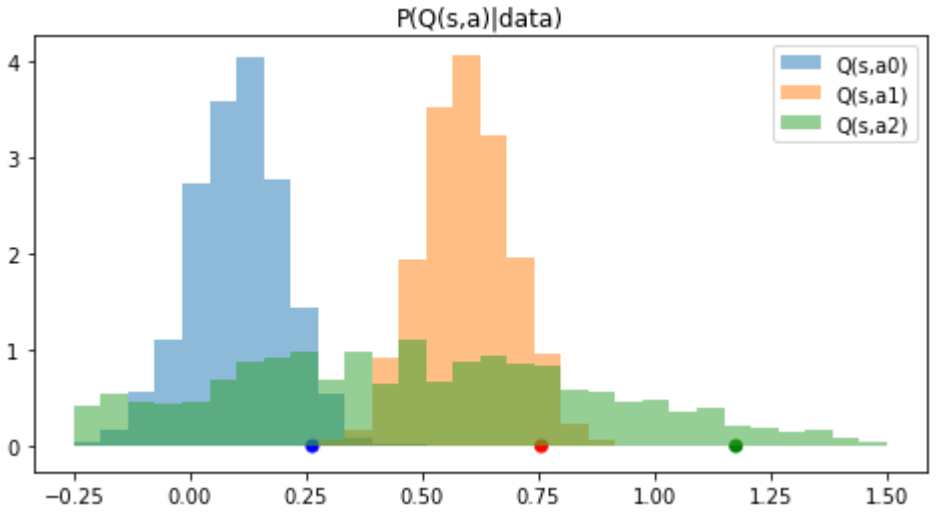
More uncertain = better.

Greater expected value = better

Math: try until upper confidence bound is small enough.

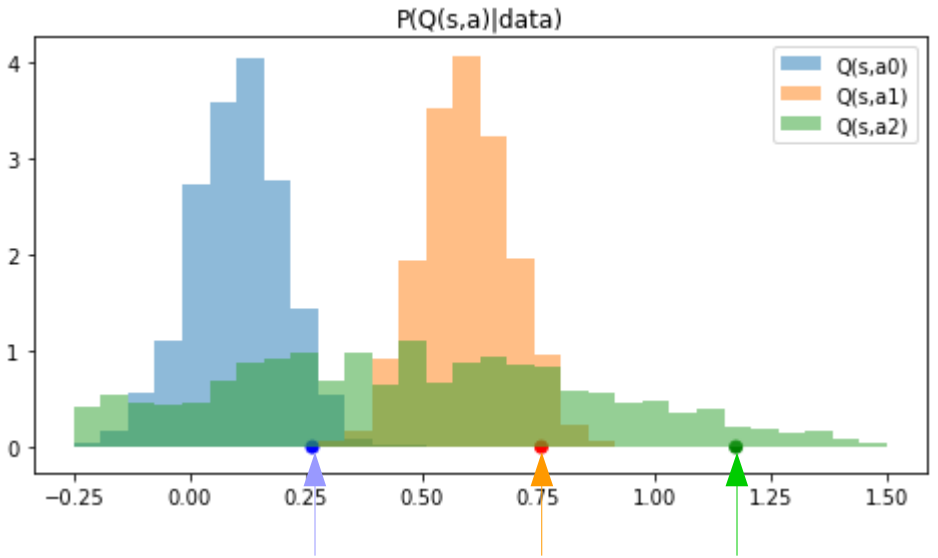
Optimism in face of uncertainty

- Policy:
  - Compute 95% upper confidence bound *for each a*
  - Take action with highest confidence bound
  - What can we tune here to explore more/less?



Optimism in face of uncertainty

- Policy:
  - Compute 95% upper confidence bound *for each a*
  - Take action with highest confidence bound
  - Adjust: change 95% to more/less



Frequentist approach

There's a number of inequalities that bound

P(x>t) < something

- E.g. Hoeffding inequality (arbitrary x in [0,1])

$$P\left(x - Ex \geq t\right) = e^{-2nt^2}$$

- Remember any others?

Frequentist approach

There's a number of inequalities that bound

P(x>t) < something

- E.g. Hoeffding inequality (arbitrary x in [0,1])

$$P\left(x - Ex \geq t\right) = e^{-2nt^2}$$

- Remember any others?

(Chernoff, Chebyshev, over9000)

Count-based exploration

UCB-1 for bandits

Take actions in in proportion to $\tilde{v}_a$

$$\tilde{v}_a = v_a + \sqrt{\frac{2 \log N}{n_a}}$$

- N number of time-steps so far
- n_a times action **a** is taken

Count-based exploration

UCB-1 for bandits

Take actions in in proportion to $\tilde{v}_a$

$$\tilde{v}_a = v_a + \sqrt{\frac{2 \log N}{n_a}}$$

Upper conf. bound for r in [0,1]

- N number of time-steps so far
- n_a times action **a** is taken

If not?

Count-based exploration

UCB-1 for bandits

Take actions in in proportion to $\tilde{v}_a$

$$\tilde{v}_a = v_a + \sqrt{\frac{2 \log N}{n_a}}$$

Upper conf. bound for r in [0,1]

- N number of time-steps so far
- n_a times action **a** is taken

If not – divide by r_max

Count-based exploration

UCB generalized for multiple states

$$\tilde{Q}(s,a) = Q(s,a) + \alpha \cdot \sqrt{\frac{2 \log N_s}{n_{s,a}}}$$

where

- N_s visits to state **s**
- $n_{s,a}$ times action **a** is taken from state **s**

Bayesian UCB

The usual way:

- Start from prior P(Q)
- Learn posterior P(Q|data)
- Take q-th percentile

What models can learn that?

Bayesian UCB

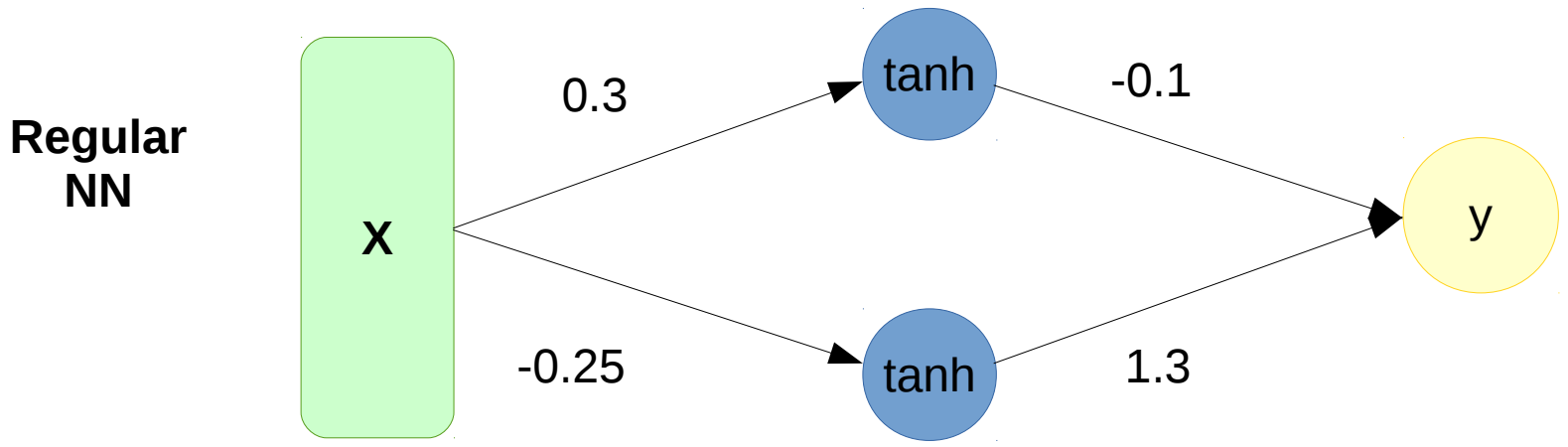
The usual way:

- Start from prior P(Q)
- Learn posterior P(Q|data)
- Take q-th percentile

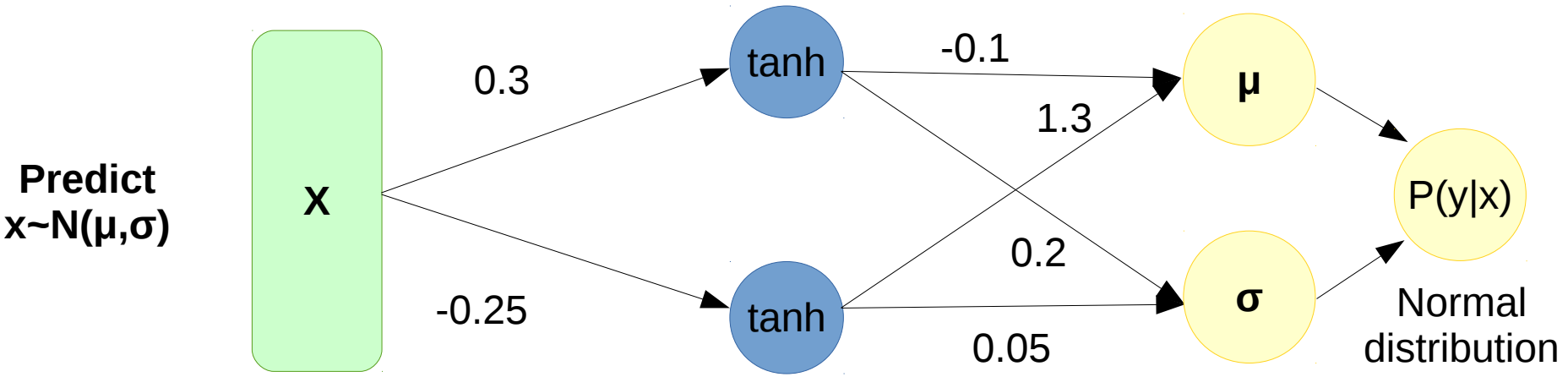
Approach 1: learn parametric P(Q), *e.g. normal*

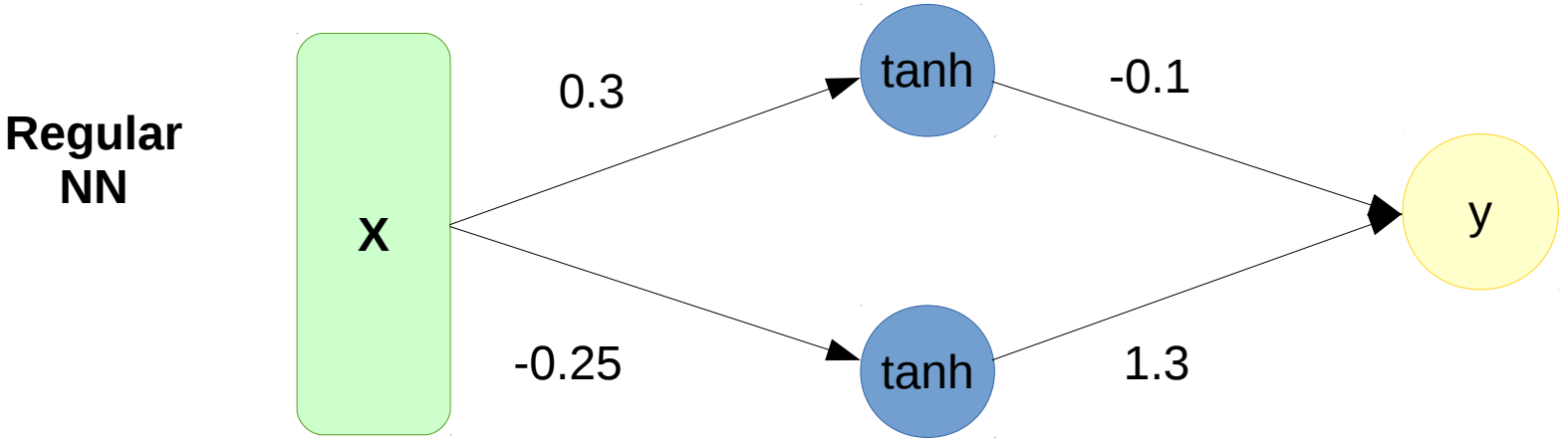
Approach 2: use bayesian neural networks

Parametric



Parametric

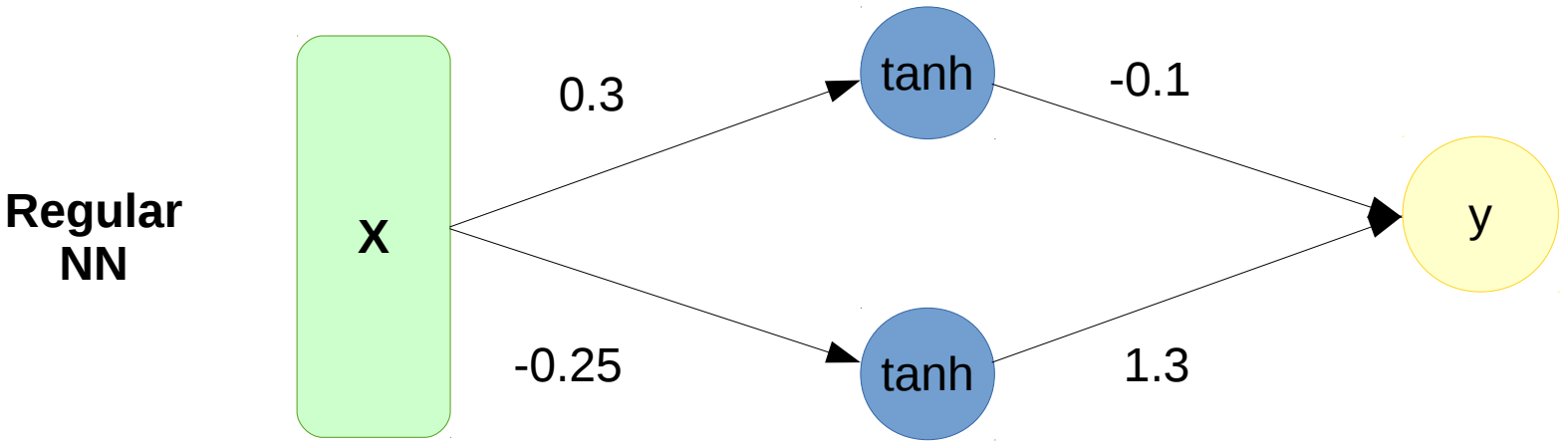

Predict
x~N(μ,σ)
x
0.3
-0.25
tanh
tanh
-0.1
1.3
0.2
0.05
μ
σ
P(y|x)
Normal
distribution


Regular
NN
x
0.3
-0.25
tanh
tanh
-0.1
1.3
y

BNNs

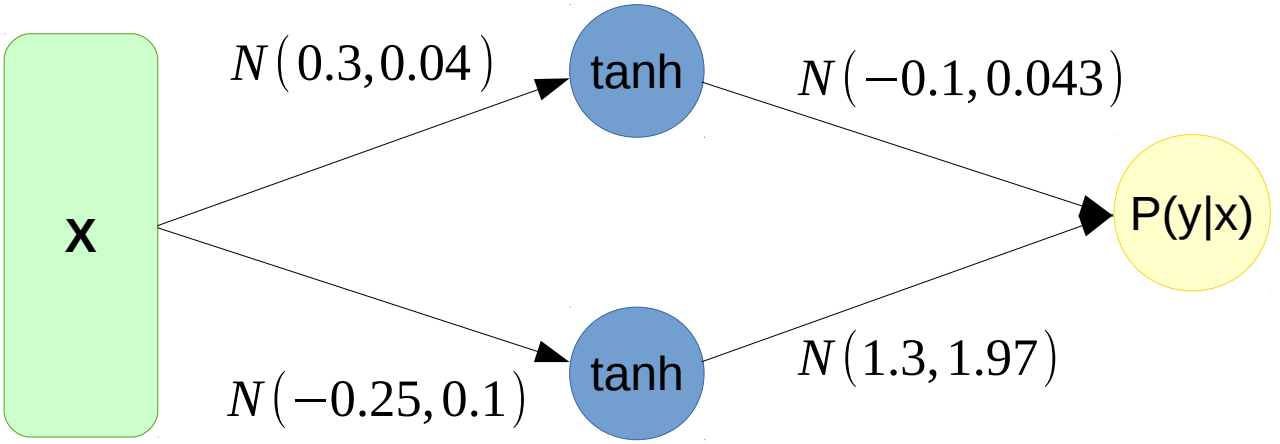
Disclaimer: this is a **hacker's guide** to BNNs!

It does not cover all the philosophy and general cases.

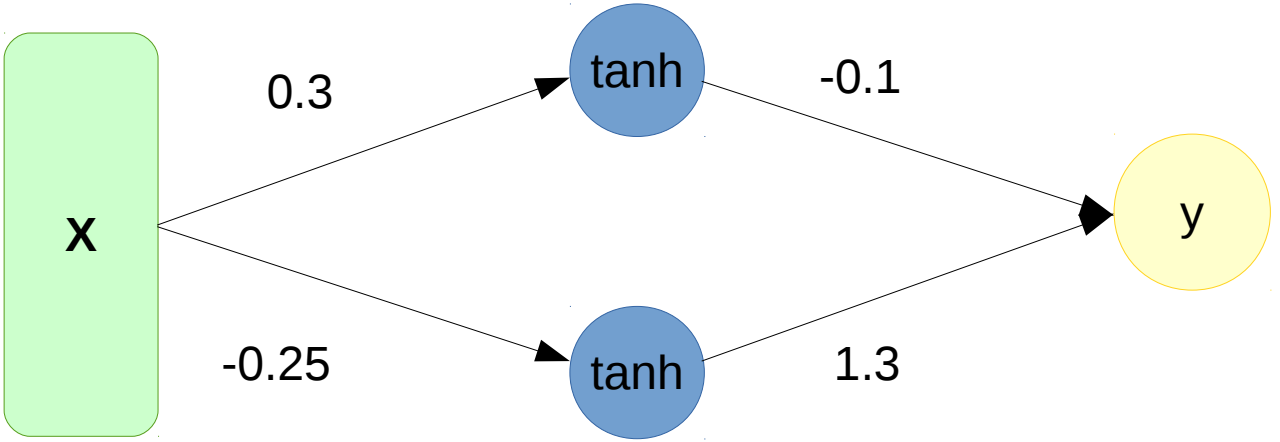


BNNs

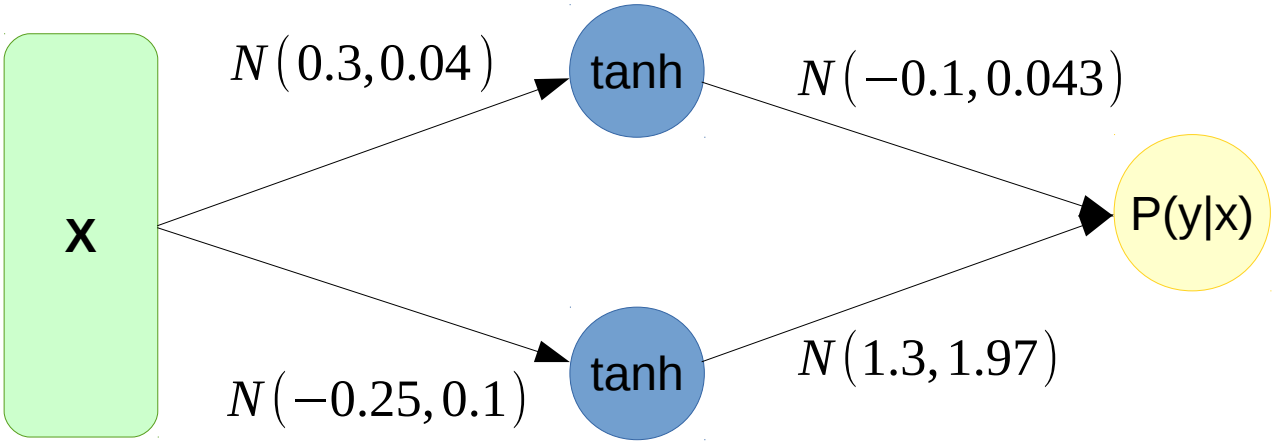
Bayesian NN



Regular NN



BNNs



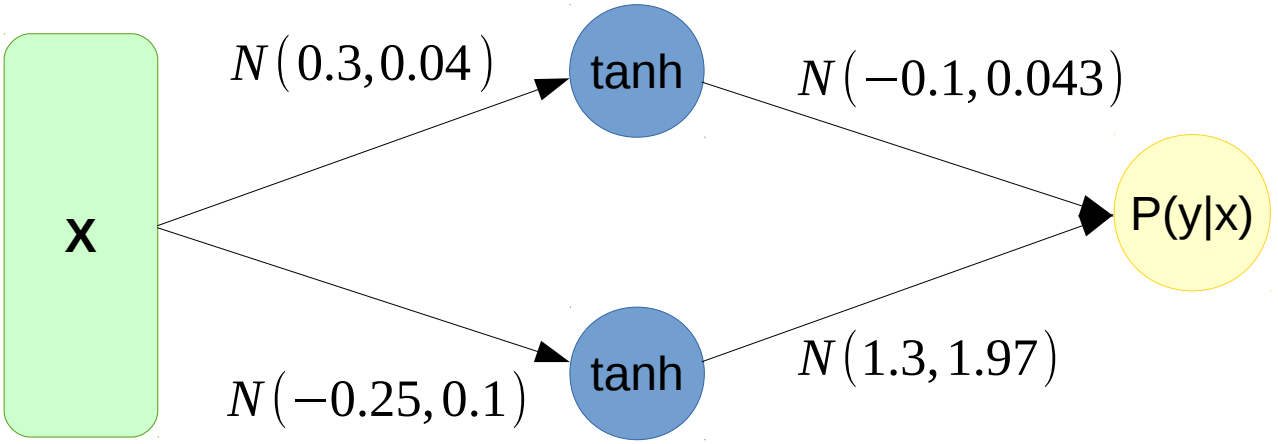
Idea:

- No explicit weights
  - Maintain parametric distribution on them instead!
    - Practical: fully-factorized normal or similar

$$q(\theta|\phi:[\mu,\sigma]) = \prod_i N(\theta_i|\mu_i,\sigma_i)$$

$$P(y|x) = E_{\theta\sim q(\theta|\phi)} P(y|x,\theta)$$

BNNs



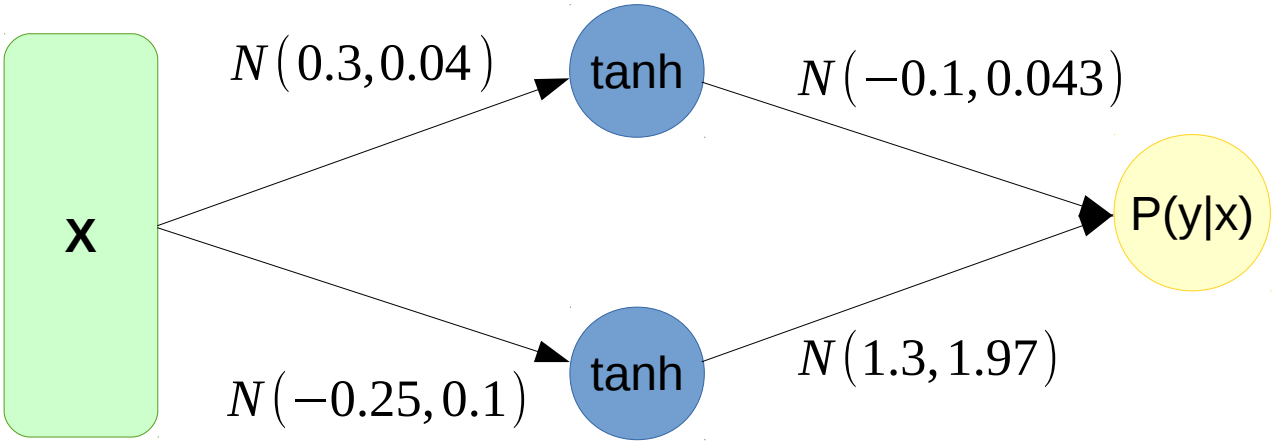
Idea:

- No explicit weights
  - Maintain parametric distribution on them instead!
    - Practical: fully-factorized normal or similar

$$q(\theta|\phi:[\mu,\sigma]) = \prod_i N(\theta_i|\mu_i,\sigma_i)$$

$$P(y|x) = E_{\theta\sim q(\theta|\phi)} P(y|x,\theta)$$

BNNs



Idea:

- No explicit weights
- Inference: sample from weight distributions, predict 1 point
- To get distribution, aggregate K samples (e.g. with histogram)
  - Yes, it means running network **multiple times per one X**

$$P(y|x) = E_{\theta \sim q(\theta|\phi)} P(y|x, \theta)$$

BNNs

Idea:

- No explicit weights
  - Maintain parametric distribution on them instead!
    - Practical: fully-factorized normal or similar

$$q\left(\theta|\phi:[\mu,\sigma]\right)=\prod_i N\left(\theta_i|\mu_i,\sigma_i\right)$$

$$P\left(y|x\right)=E_{\theta\sim q(\theta|\phi)}P\left(y|x,\theta\right)$$

- Learn parameters of that distribution (reparameterization trick)
  - Less variance: local reparameterization trick.

$$\hat{\phi}=argmax_{\phi}E_{x_i,y_i\sim d}E_{\theta\sim q(\theta|\phi)}P\left(y_i|x_i,\theta\right)$$

wanna explicit formulae? *d = dataset*

Lower bound

d = dataset

$$-KL(q(\theta|\phi)||p(\theta|d)) = -\int_{\theta} q(\theta|\phi)\cdot\log\frac{q(\theta|\phi)}{p(\theta|d)}$$

$$-\int_{\theta} q(\theta|\phi)\cdot\log\frac{q(\theta|\phi)}{\left[\frac{p(d|\theta)\cdot p(\theta)}{p(d)}\right]} = -\int_{\theta} q(\theta|\phi)\cdot\log\frac{q(\theta|\phi)\cdot p(d)}{p(d|\theta)\cdot p(\theta)}$$

$$-\int_{\theta} q(\theta|\phi)\cdot[\log\frac{q(\theta|\phi)}{p(\theta)} - \log p(d|\theta) + \log p(d)]$$

$$[E_{\theta\sim q(\theta|\phi)}\log p(d|\theta)] - KL(q(\theta|\phi)||p(\theta)) + \log p(d)$$

loglikelihood **-distance to prior** **+const**

Lower bound

$$\phi = \underset{\phi}{argmax}\left(-KL\left(q\left(\theta|\phi\right)||p\left(\theta|d\right)\right)\right)$$

$$\underset{\phi}{argmax}\left(\left[E_{\theta\sim q\left(\theta|\phi\right)}\log p\left(d|\theta\right)\right]-KL\left(q\left(\theta|\phi\right)||p\left(\theta\right)\right)\right)$$

Can we perform gradient ascent directly?

Reparameterization trick

$$\phi = \underset{\phi}{argmax}\left(-KL\left(q(\theta|\phi)\,||\,p(\theta|d)\right)\right)$$

$$\underset{\phi}{argmax}\left(\underline{\left[E_{\theta\sim q(\theta|\phi)}\log p(d|\theta)\right]} - \underline{KL\left(q(\theta|\phi)\,||\,p(\theta)\right)}\right)$$

Use reparameterization trick

simple formula (for normal q)

What does this log P(d|...) mean?

BNN likelihood

$$E_{\theta\sim N(\theta|\mu_\phi,\sigma_\phi)}\log p(d|\theta) = E_{\psi\sim N(0,1)}\log p\left(d|\left(\mu_\phi + \sigma_\phi\cdot\psi\right)\right)$$

Better: local reparameterization trick (google it)

Reparameterization trick

$$\phi = \underset{\phi}{argmax}\left(-KL\left(q\left(\theta|\phi\right)||p\left(\theta|d\right)\right)\right)$$

$$\underset{\phi}{argmax}\left(\left[E_{\theta\sim q(\theta|\phi)}\log p\left(d|\theta\right)\right]-KL\left(q\left(\theta|\phi\right)||p\left(\theta\right)\right)\right)$$

In other words,
$\Sigma_{x,y\sim d}$ log p(y|x,μ+σψ)

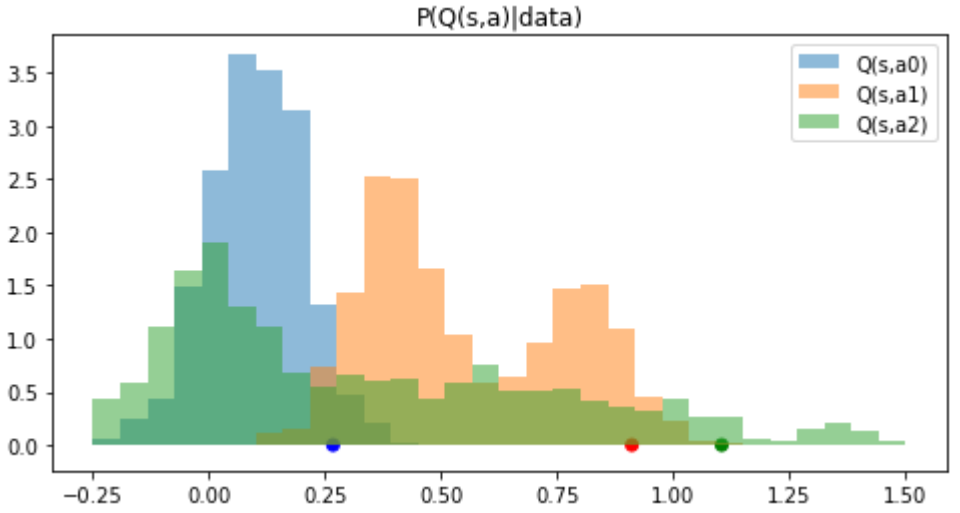
BNN likelihood

$$E_{\theta\sim N(\theta|\mu_\phi,\sigma_\phi)}\log p\left(d|\theta\right)=E_{\psi\sim N(0,1)}\log p\left(d|\left(\mu_\phi+\sigma_\phi\cdot\psi\right)\right)$$

Better: local reparameterization trick (google it)

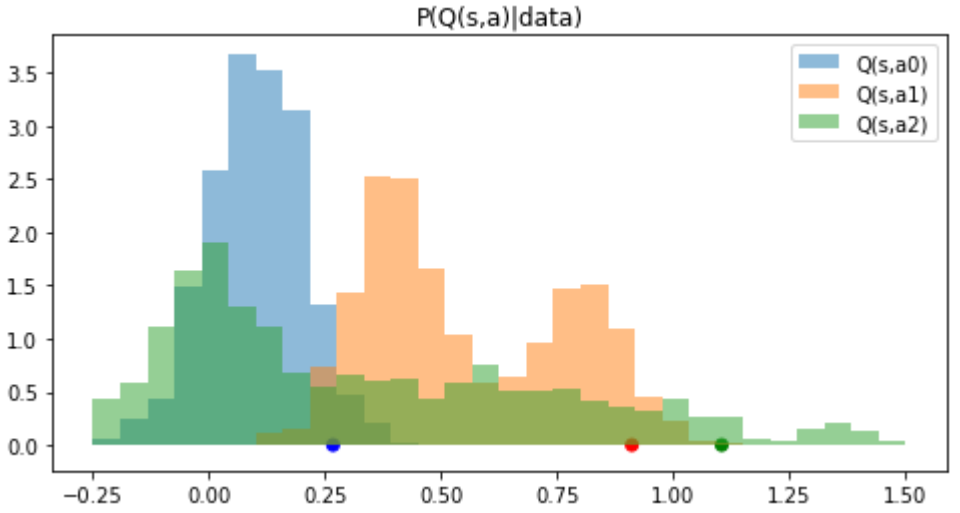
Using BNNs

- If you sample from BNNs
  - Can learn ~arbitrary distribution (e.g. multimodal)
  - But it takes running network many times
  - Use empirical percentiles for exploration priority
    - Again, 3 points on horizontal axis are percentiles



Using BNNs

- If you sample from BNNs
  - Can learn ~arbitrary distribution (e.g. multimodal)
  - **But it takes running network many times**
  - Use empirical percentiles for exploration priority
    - Again, 3 points on horizontal axis are percentiles



Practical stuff

- Approximate exploration policy

  with something cheaper

- Bayesian UCB:
  - Prior can make or break it
  - Sometimes parametric guys win (vs bnn)

- Of course, neural nets aren't

  always the best model

Markov Decision Processes

- Naive approach:
  - Infer posterior distribution on Q(s,a)
  - Do UCB or Thompson Sampling on those Q-values
  - **Anything wrong with this?**

Markov Decision Processes

- Naive approach:
  - Infer posterior distribution on Q(s,a)
  - Do UCB or Thompson Sampling on those Q-values
  - Agent is “greedy” w.r.t. exploration

    It would prefer taking one uncertain action now than make several steps to end up in unexplored regions

Markov Decision Processes

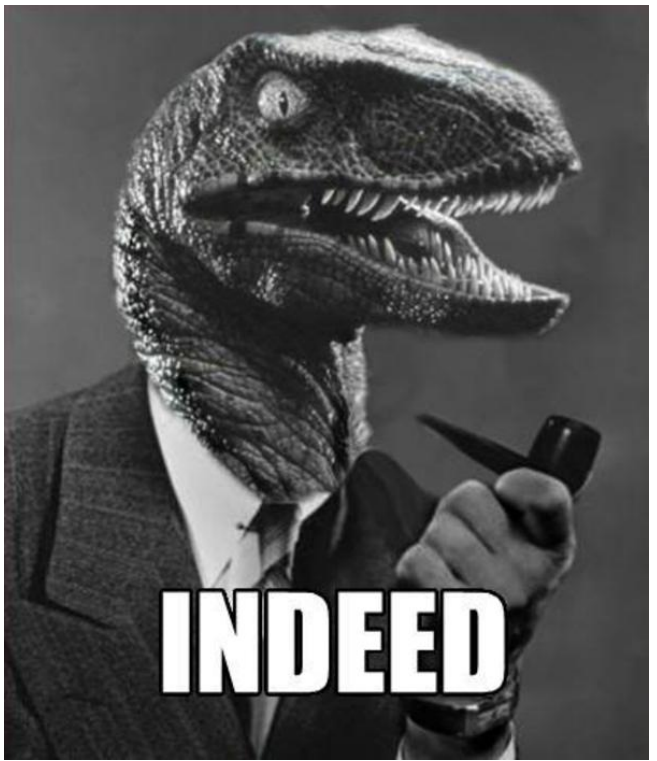
- Naive approach:
  - Infer posterior distribution on Q(s,a)
  - Do UCB or Thompson Sampling on those Q-values
  - Agent is “greedy” w.r.t. exploration

- Reward augmentation
  - Devise a surrogate “reward” for exploration
    - We “pay” our agent for exploring
  - Maximize this reward with a (separate) RL agent

I got it!

It's all about funding!

(Just kidding)



Reward augmentation

Let's “pay” agent for exploration!

$$\tilde{r}(z,a,s') = r(s,a,s') + r_{exploration}(s,a,s')$$

Reward augmentation

Let's “pay” agent for exploration!

$$\tilde{r}(z,a,s') = r(s,a,s') + r_{exploration}(s,a,s')$$

Q: Any suggestions on surrogate **r** for atari?

UNREAL main idea

- Auxilary objectives:
  - *Pixel control:* maximize pixel change in NxN grid over image
  - *Feature control:* maximize activation of some neuron deep inside neural network
  - *Reward prediction:* predict future rewards given history

article: arxiv.org/abs/1611.05397

blog post: bit.ly/2g9Yv2A

UNREAL main idea

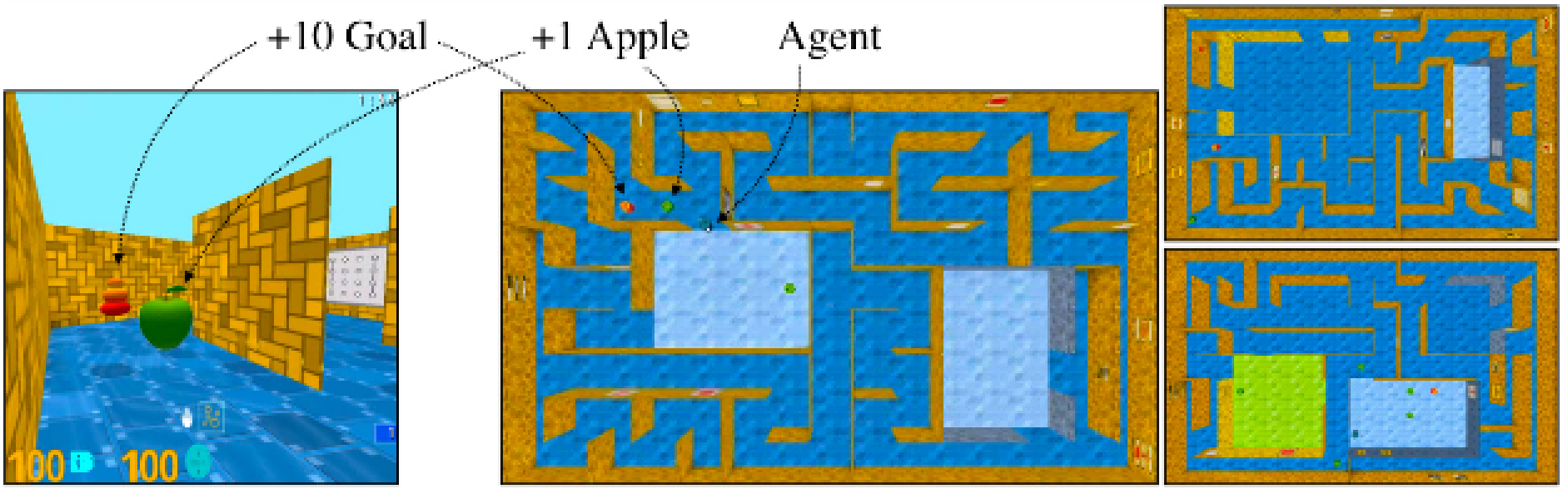
- Auxilary objectives:
  - *Pixel co*… …N grid over ima…
  - *Feature* … …me neuron …
  - *Reward* … …given history

Keep calm!
we'll get more theoretically sound models in a few slides

…: arxiv.org/abs/1611.05397

…blog post: bit.ly/2g9Yv2A

Environment: Labyrinth

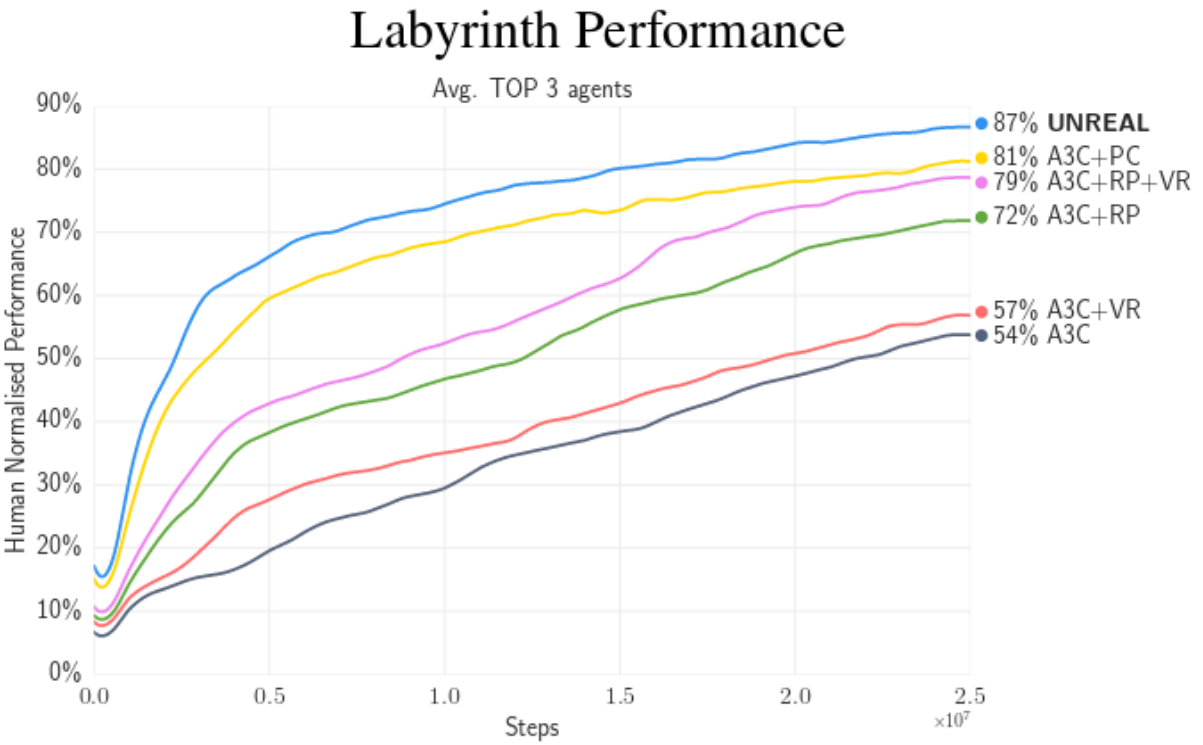


Agent Input

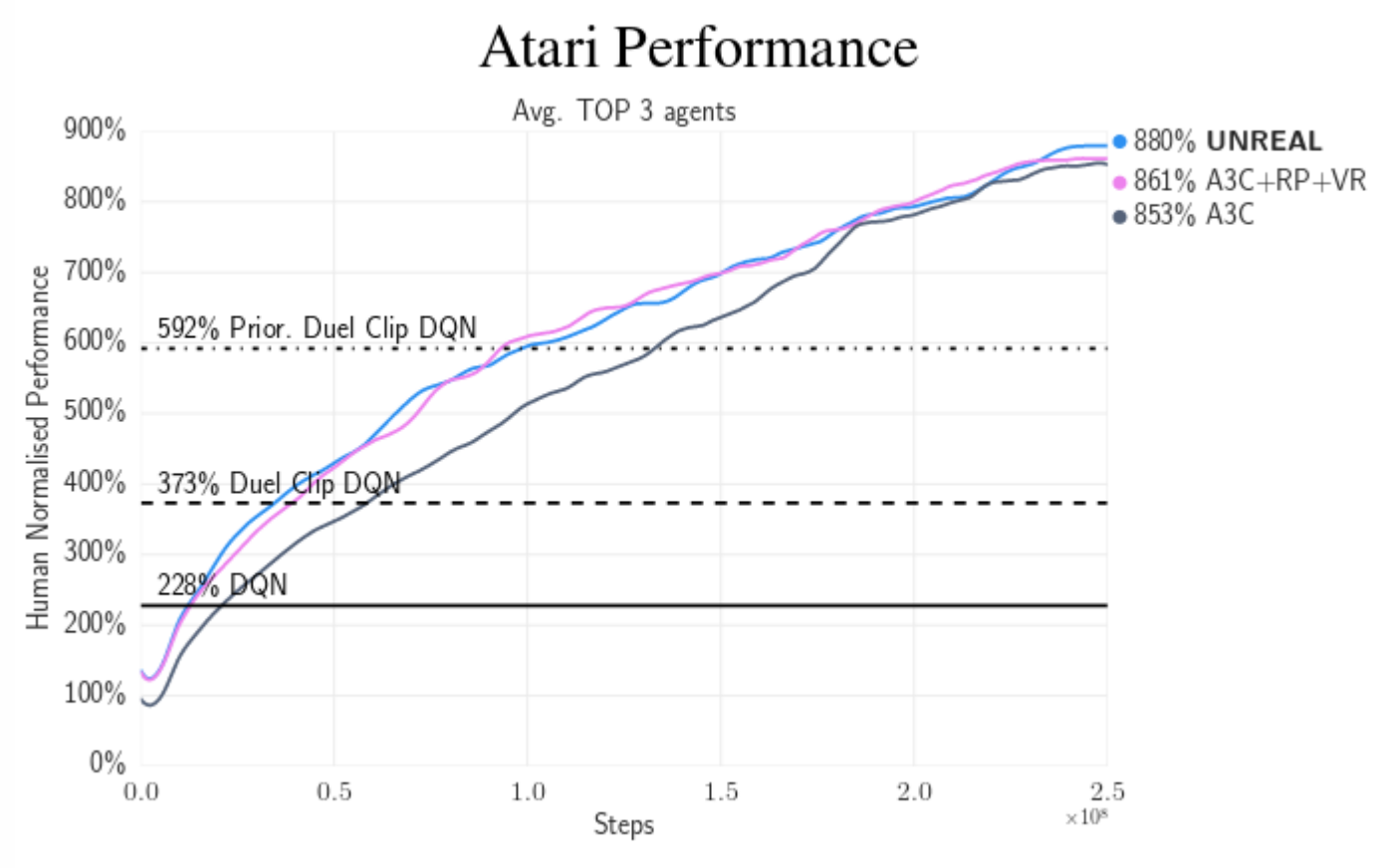
nav_maze_all_random_02 samples

- Maze with rewards
- Partially observable
  - Used a2c + LSTM + experience replay

Results: labyrinth



Results: Atari



Count-based models

TL;DR encourage visiting underexplored states

- Use approximate density model, N(s)

N(s) ~ how many times agent visited s over training time

Count-based models

TL;DR encourage visiting underexplored states

- Use approximate density model, N(s)

  N(s) ~ how many times agent visited s over training time

- Reward for visiting low-N states, e.g. N(s)^-0.5

Examples: arxiv:1606.01868, arxiv:1703.01310

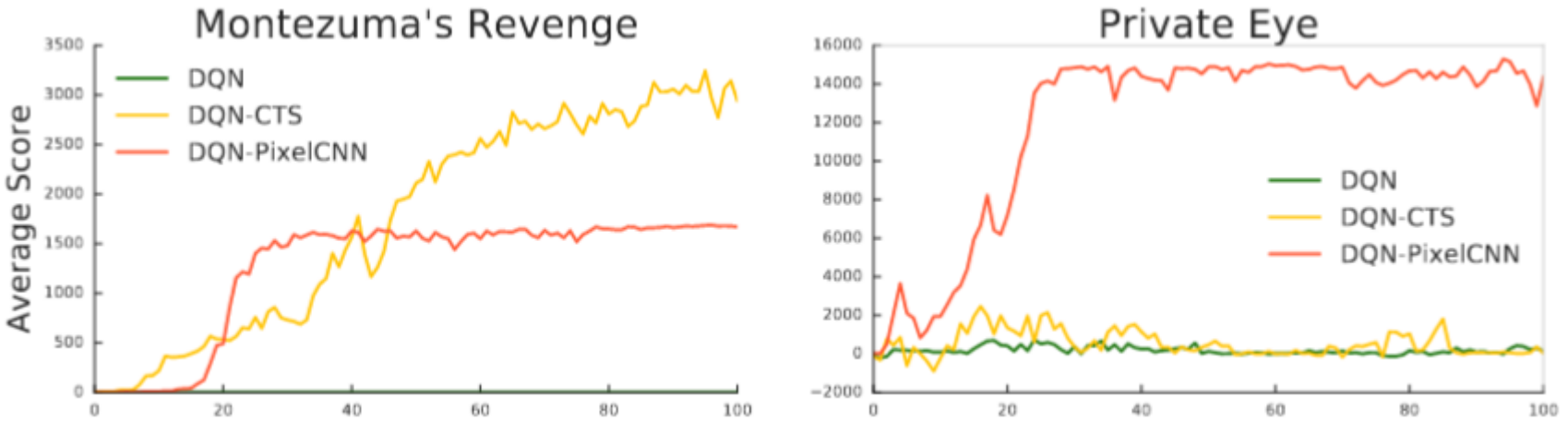
Count-based models

On-policy: start training with surrogate rewards, gradually reduce their weight over time

Off-policy: train a separate agent to maximize exploration via surrogate rewards, learn Q(s,a) for original reward from that with off-policy algo

Estimating counts

We need some way to estimate N(s)



Task-specific: e.g. image density models

Generic: e.g. VAE / alphaGAN, expected log p(x|z)

Density ratio: estimate p(s) / q(s) for baseline q

Density ratio trick

We need some way to estimate N(s)

$$N(s) \sim d(s)$$

count = state density (probability)
up to a coefficient of n_steps

Density ratio trick

We need some way to estimate N(s)

$$N(s) \sim d(s)$$

count = state density (probability)
up to a coefficient of n_steps

To estimate **d(s)**, we introduce any known distribution **q(s)**, e.g. uniform

$$d(s) = q(s) \cdot \frac{d(s)}{q(s)}$$

Density ratio trick

We need some way to estimate N(s)

$$N(s) \sim d(s)$$

count = state density (probability)
up to a coefficient of n_steps

To estimate **d(s)**, we introduce any known distribution **q(s)**, e.g. uniform

$$d(s) = q(s) \cdot \frac{d(s)}{q(s)}$$

Train another model to discriminate between $s \sim d(s)$ and $s \sim q(s)$

$$p(s \in d(s)) = ???$$

Q: What is P(s ∈ d(s)) under optimal model?

Density ratio trick

Train another model to discriminate between $s \sim d(s)$ and $s \sim q(s)$

Perfect discriminator: $$p(s \in d) = \frac{d(s)}{d(s) + q(s)}$$

Density ratio trick

Train another model to discriminate between $s \sim d(s)$ and $s \sim q(s)$

Perfect discriminator: $$p(s \in d) = \frac{d(s)}{d(s) + q(s)}$$

Q: Can you express $\frac{d(s)}{q(s)}$ in terms of $p(s \in d(s))$ **?**

Density ratio trick

Train another model to discriminate between $s \sim d(s)$ and $s \sim q(s)$

Perfect discriminator: $$p(s \in d) = \frac{d(s)}{d(s) + q(s)}$$

$$\frac{1}{p(s \in d)} = \frac{d(s) + q(s)}{d(s)} = 1 + \frac{q(s)}{d(s)}$$

$$\frac{q(s)}{d(s)} = \frac{1}{p(s \in d(s))} - 1 = \frac{1 - p(s \in d(s))}{p(s \in d(s))}$$

Density ratio trick

Train another model to discriminate between $s \sim d(s)$ and $s \sim q(s)$

Perfect discriminator: $$p(s \in d) = \frac{d(s)}{d(s) + q(s)}$$

$$\frac{d(s)}{q(s)} = \frac{p(s \in d(s))}{1 - p(s \in d(s))} \approx \frac{Discriminator(s)}{1 - Discriminator(s \in d(s))}$$

$$d(s) = q(s) \cdot \frac{d(s)}{q(s)} \approx q(s) \cdot \frac{Discriminator(s)}{1 - Discriminator(s \in d(s))}$$

Density ratio trick

We need some way to estimate N(s)

$$N(s) \sim d(s)$$

count = state density (probability)
up to a coefficient of n_steps

Train neural network to discriminate between visited states $s \sim d(s)$ and arbitrary known $s \sim q(s)$

$$d(s) = q(s) \cdot \frac{d(s)}{q(s)} \approx q(s) \cdot \frac{Discriminator(s)}{1 - Discriminator(s \in d(s))}$$

Uniform q(s) = simple math, high variance d(s),
Task-specific q(s) = possibly smaller variance

Variational Information-Maximizing Exploration

I hope I don’t have enough time for this…

arxiv:1605.09674

Variational Information-Maximizing Exploration

Curiosity

Taking actions that increase your knowledge about the world (a.k.a. the environment)

Knowledge about the world

Whatever allows you to predict how world works depending on your behavious

arxiv:1605.09674

Vime main idea

Add curiosity to the reward

$$\tilde{r}(z,a,s') = r(s,a,s') + \beta r_{vime}(\tau,a,s')$$

Curiosity definition

$$r_{vime}(z,a,s') = I(\theta; s' | \tau, a)$$

Vime main idea

Environment model

$$P(s'|s,a,\theta)$$

Session

$$\tau_t = \langle s_{0,} a_{0,} s_{1,} a_{1,} \ldots, s_t \rangle$$

Surrogate reward

$$\tilde{r}(\tau,a,s') = r(s,a,s') + \beta r_{vime}(\tau,a,s') = r(s,a,s') + \beta I(\theta; s'|\tau,a)$$

curiosity

$$I(\theta; s'|\tau,a) = H(\theta|\tau,a) - H(\theta|\tau,a,s') = E_{s_{t+1} \sim P(s_{t+1}|s,a)} KL[P(\theta|\tau,a,s') \| P(\theta|\tau)]$$

need proof for that last line?

Naive objective

$$E_{s_{t+1}\sim P(s_{t+1}|s,a)}\,KL\left[P(\theta|\tau,a,s')\,\|\,P(\theta|\tau)\right]=\int_{s'}P(s'|s,a)\cdot\int_{\theta}P(\theta|\tau,a,s')\cdot\log\frac{P(\theta|\tau,a,s')}{P(\theta|\tau)}\,d\theta\,ds'$$

where

$$P(\theta|\tau)=\frac{P(z|\theta)\cdot P(\theta)}{P(\tau)}=\frac{\prod_{t}P(s_{t+1}|s_t,a_t,\theta)\cdot P(\theta)}{\int_{\theta}P(\tau|\theta)\cdot P(\theta)\,d\theta}$$

Naive objective

$$E_{s_{t+1}\sim P(s_{t+1}|s,a)} KL\left[P(\theta|\tau,a,s')\|P(\theta|\tau)\right] = \int_{s'} P(s'|s,a)\cdot \int_{\theta} P(\theta|\tau,a,s')\cdot \log \frac{P(\theta|\tau,a,s')}{P(\theta|\tau)} d\theta\, ds'$$

Sample From MDP

Sample Somehow...

Model **Prior**

$$P(\theta|\tau) = \frac{P(z|\theta)\cdot P(\theta)}{P(\tau)} = \frac{\prod_t P(s_{t+1}|s_t,a_t,\theta)\cdot P(\theta)}{\int_{\theta} P(\tau|\theta)\cdot P(\theta) d\theta}$$

dunno

Better avoid computing P(theta|tau) directly!


BAYES
TO SANTA ~~CLAUS~~

We want a model that

- predicts P(s'|s,a,s,…,a, theta)
- we can estimate P(theta | D)
- we can sample from P(theta | D)

We want a model that

- predicts P(s'|s,a,s,…,a, theta)
- we can estimate P(theta | D)
- we can sample from P(theta | D)
- **Bayesian Neural Networks!**

Vime objective

$$E_{s_{t+1}\sim P(s_{t+1}|s,a)}\,KL\left[P(\theta|\tau,a,s')\,\|\,P(\theta|\tau)\right]=\int_{s'}P(s'|s,a)\cdot\int_{\theta}P(\theta|\tau,a,s')\cdot\log\frac{P(\theta|\tau,a,s')}{P(\theta|\tau)}\,d\theta\,ds'$$

$$KL\left[P(\theta|\tau,a,s')\,\|\,P(\theta|\tau)\right]\approx KL\left[q(\theta|\tau,a,s')\,\|\,q(\theta|\tau)\right]\equiv KL\left[q(\theta|\phi')\,\|\,q(\theta|\phi)\right]$$

$$E_{s_{t+1}\sim P(s_{t+1}|s,a)}\,KL\left[P(\theta|\tau,a,s')\,\|\,P(\theta|\tau)\right]\approx\int_{s'}P(s'|s,a)\cdot\int_{\theta}q(\theta|\tau,a,s')\cdot\log\frac{q(\theta|\tau,a,s')}{q(\theta|\tau)}\,d\theta\,ds'$$

BNN

sample from env

sample from BNN

BNN last tick

Algorithm

Forever:

1. Interact with environment, get <s,a,r,s'>
2. Compute curiosity reward

$$\tilde{r}(z,a,s') = r(s,a,s') + \beta KL[q(\theta|\phi') \| q(\theta|\phi)]$$

3. train_agent(s,a,$\tilde{r}$,s') //with any RL algorithm
4. train_BNN(s,a,s') //maximize lower bound

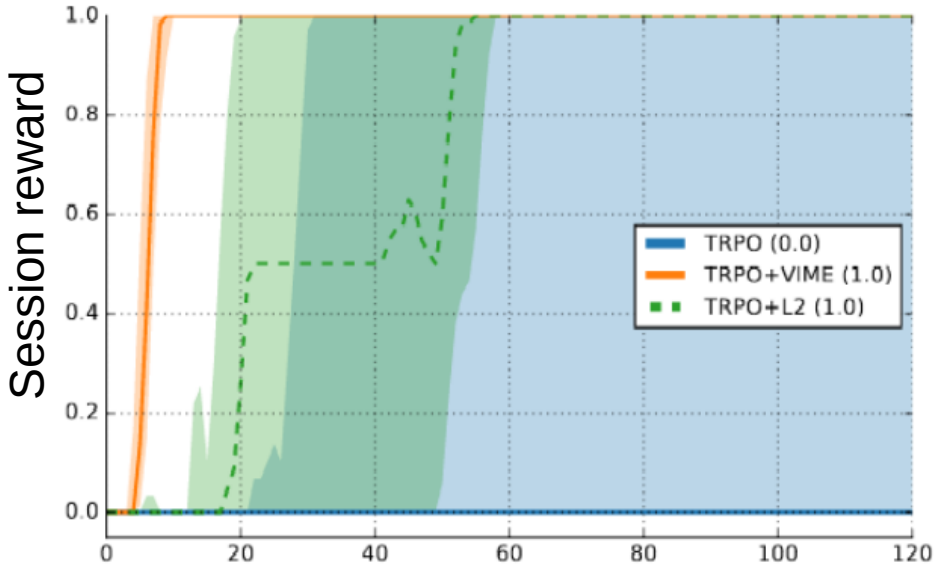
Dirty hacks

- **Use batches** of many <s,a,r,s'>
  - for CPU/GPU efficiency
  - greatly improves RL stability
- **Simple formula for KL**
  - Assuming fully-factorized normal distribution

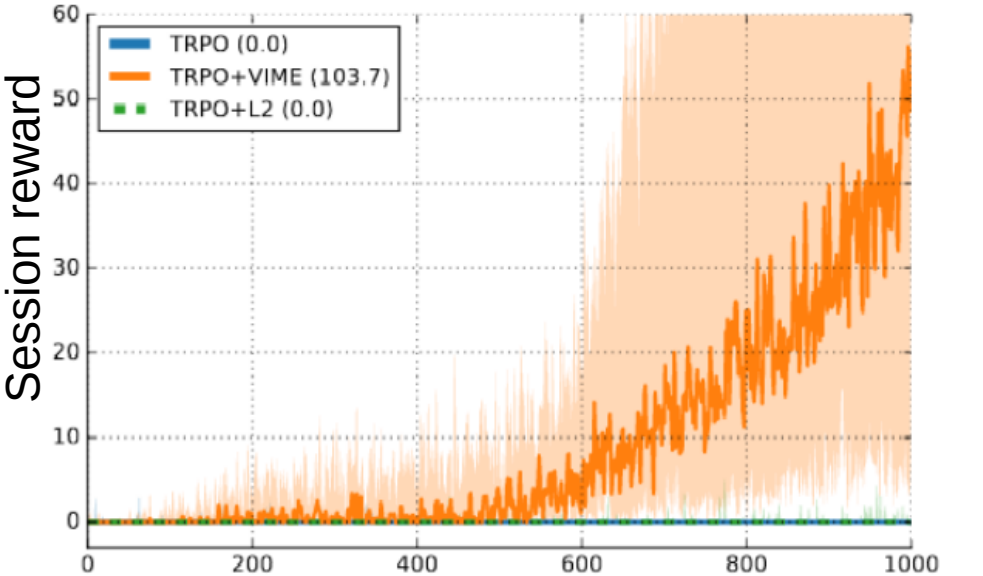
$$KL[q(\theta|\phi')||q(\theta|\phi)] = \frac{1}{2}\sum_{i<|\theta|}[\left(\frac{\sigma_i'}{\sigma_i}\right)^2 + 2\log\sigma_i - 2\log\sigma_i' + \frac{(\mu_i' - \mu_i)^2}{\sigma_i^2}]$$

  - Even simpler: second order Taylor approximation
- **Divide KL by its running average over past iterations**

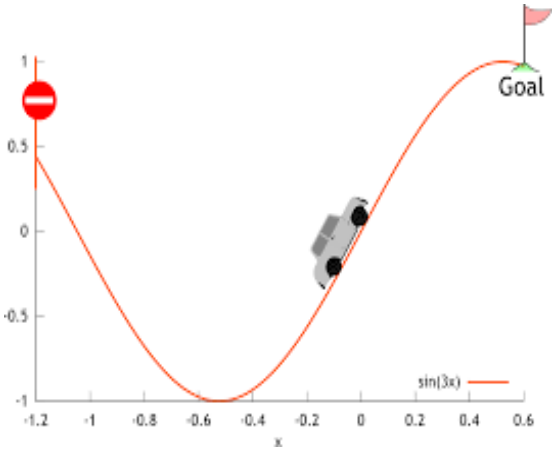
Results

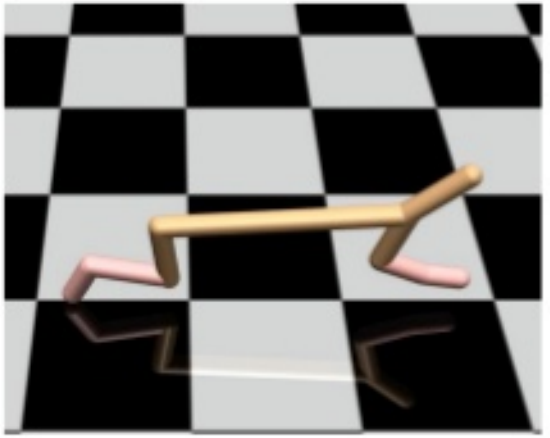


(a) MountainCar



(c) HalfCheetah





Results

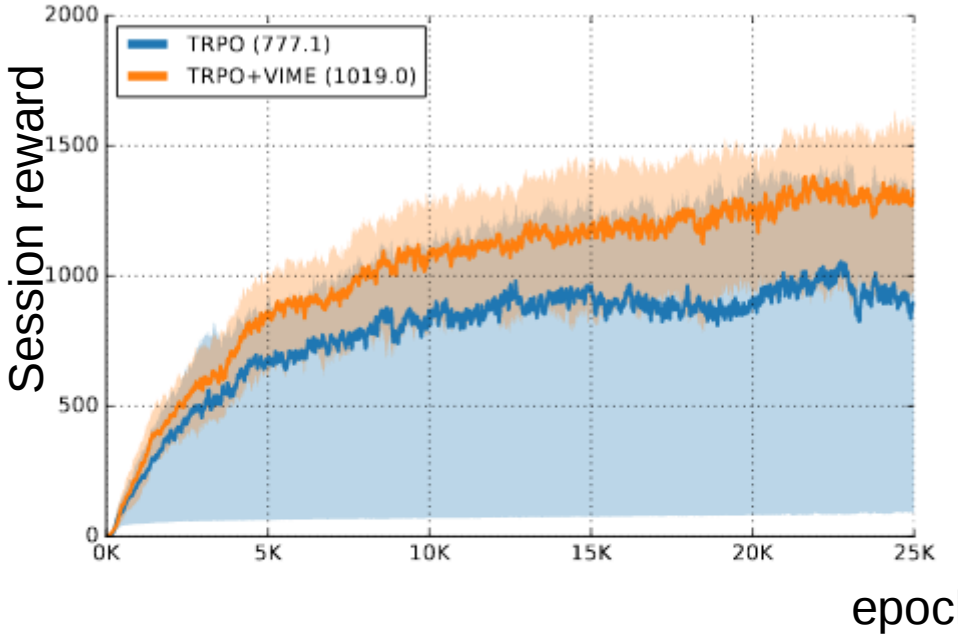
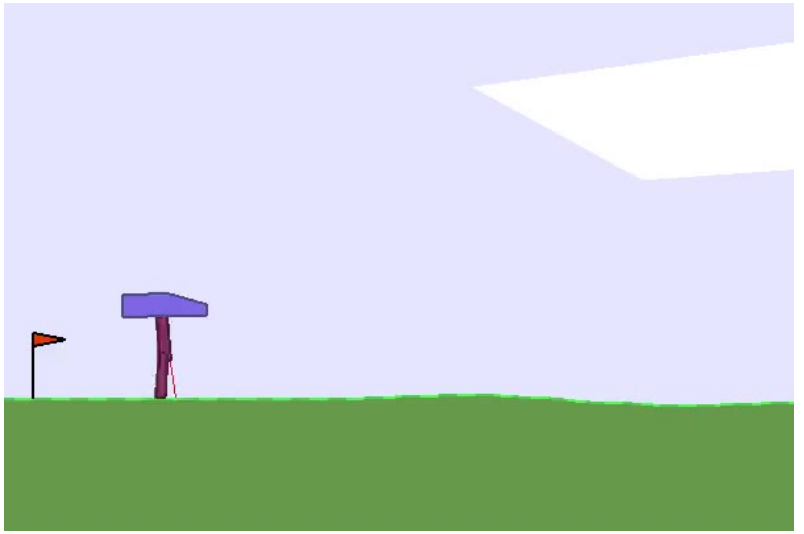
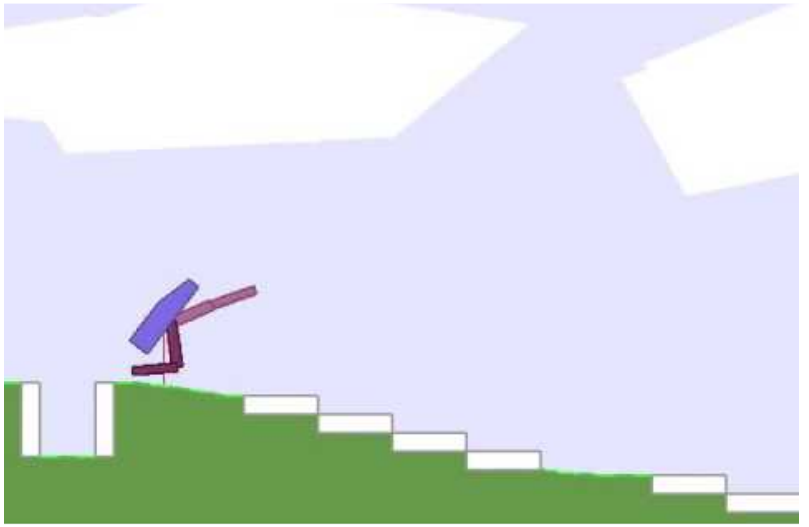


Figure 3: Performance of TRPO with and without VIME on the high-dimensional Walker2D locomotion task.

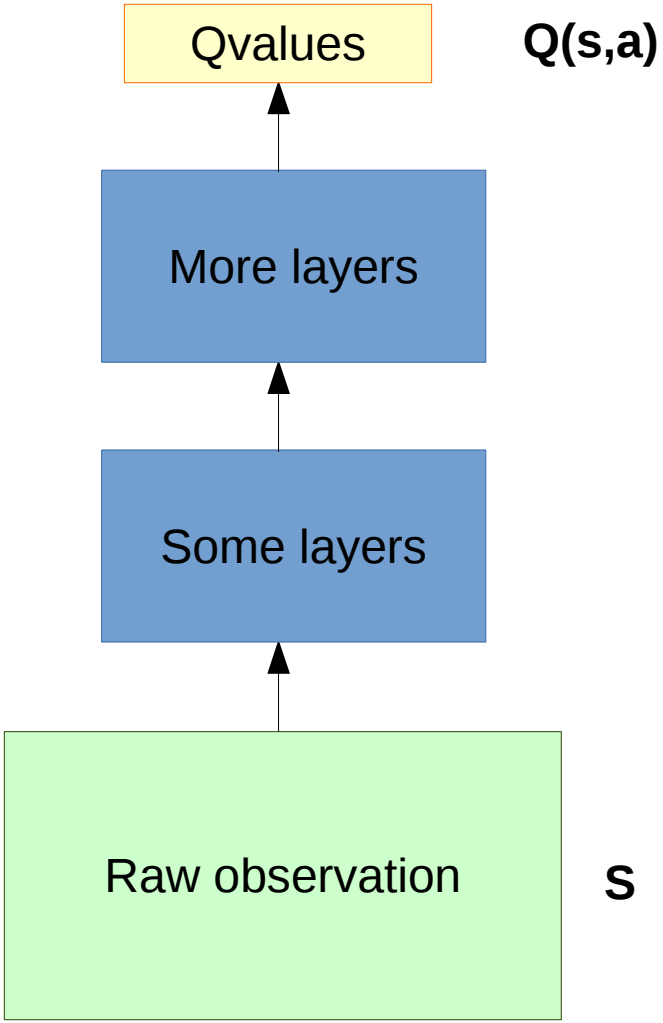




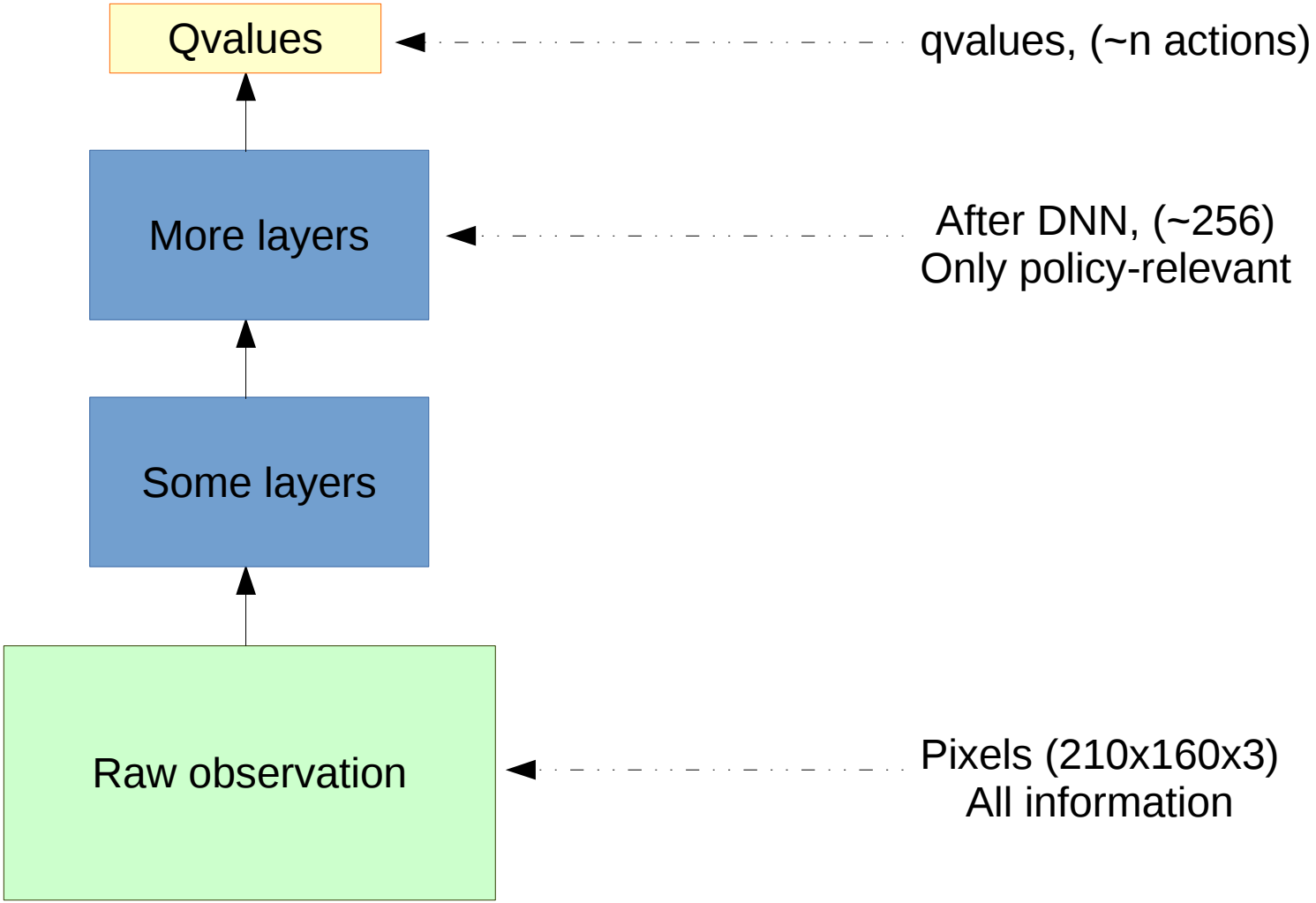
Pitfalls

- It's curious about irrelevant things
- Predicting (210x160x3) images is hard
- We don't observe full states (POMDP)

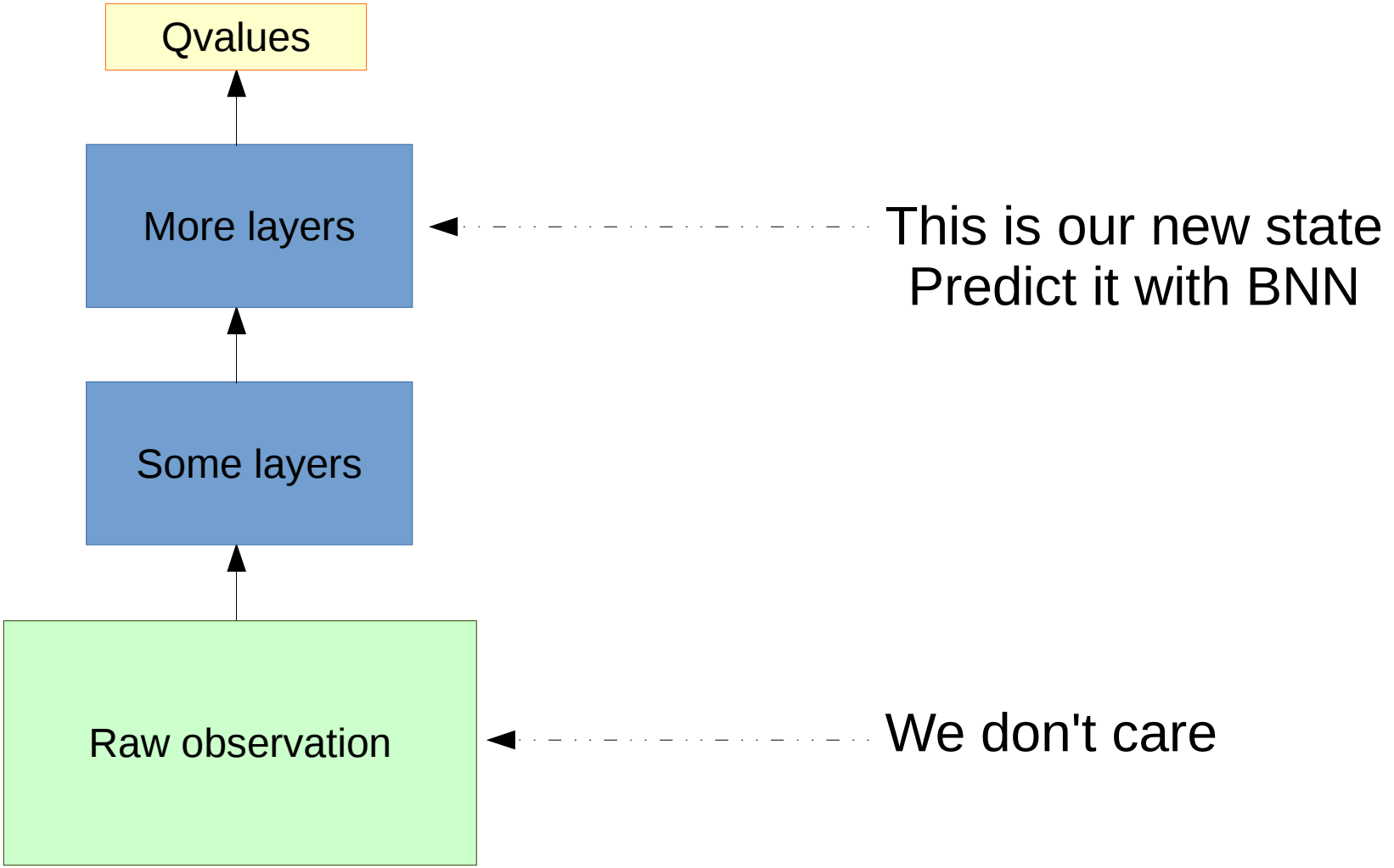
State = hidden NN activation



State = hidden NN activation



State = hidden NN activation



A case for POMDP