Variational Information Maximizing Exploration

Bayesian reinforcement learning

A.Panin, K. Sidorov, LAMBDA lab

Yandex
Data Factory

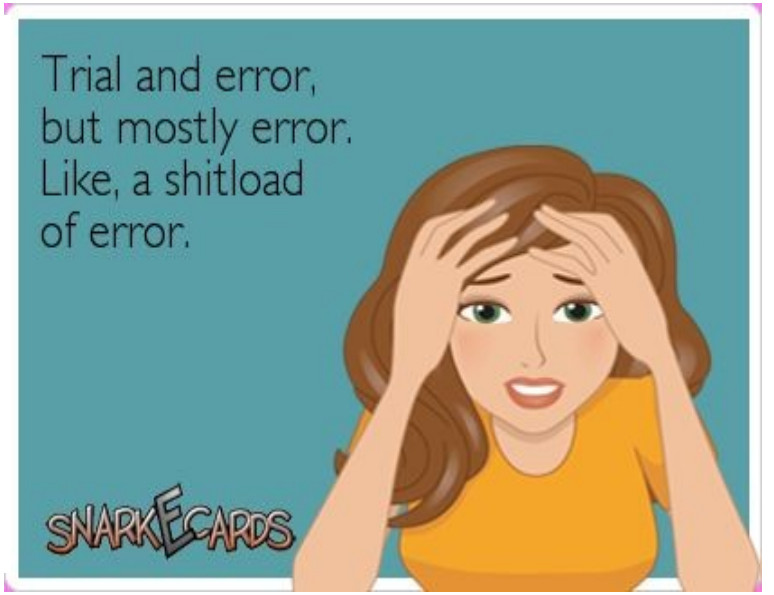
LAMBDA

British Hedgehog
Preservation Society

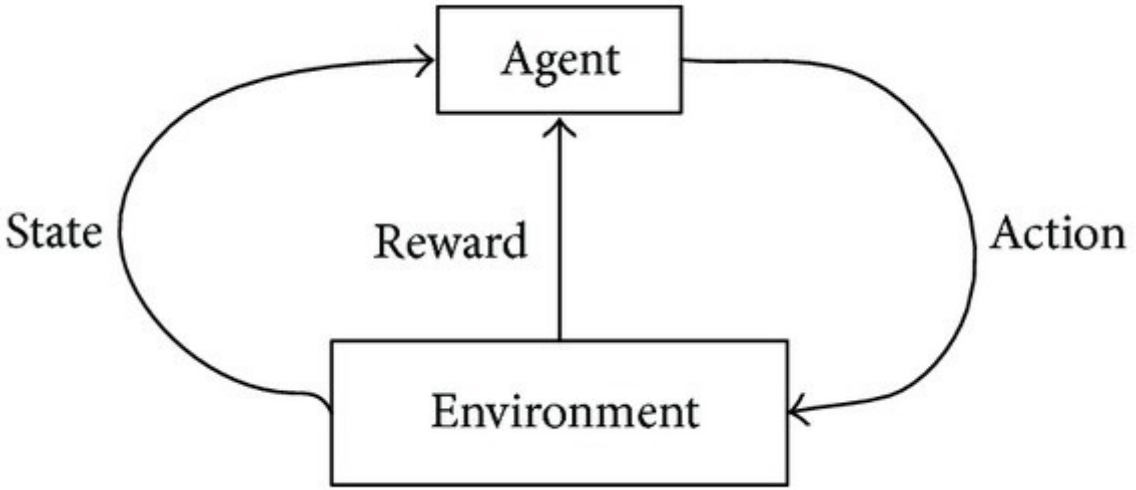
Reinforcement Learning

What-what learning?

- Learn optimal policy in a process
- Policy (situation) → (optimal action)
- Get data by trial and error and error and error and error



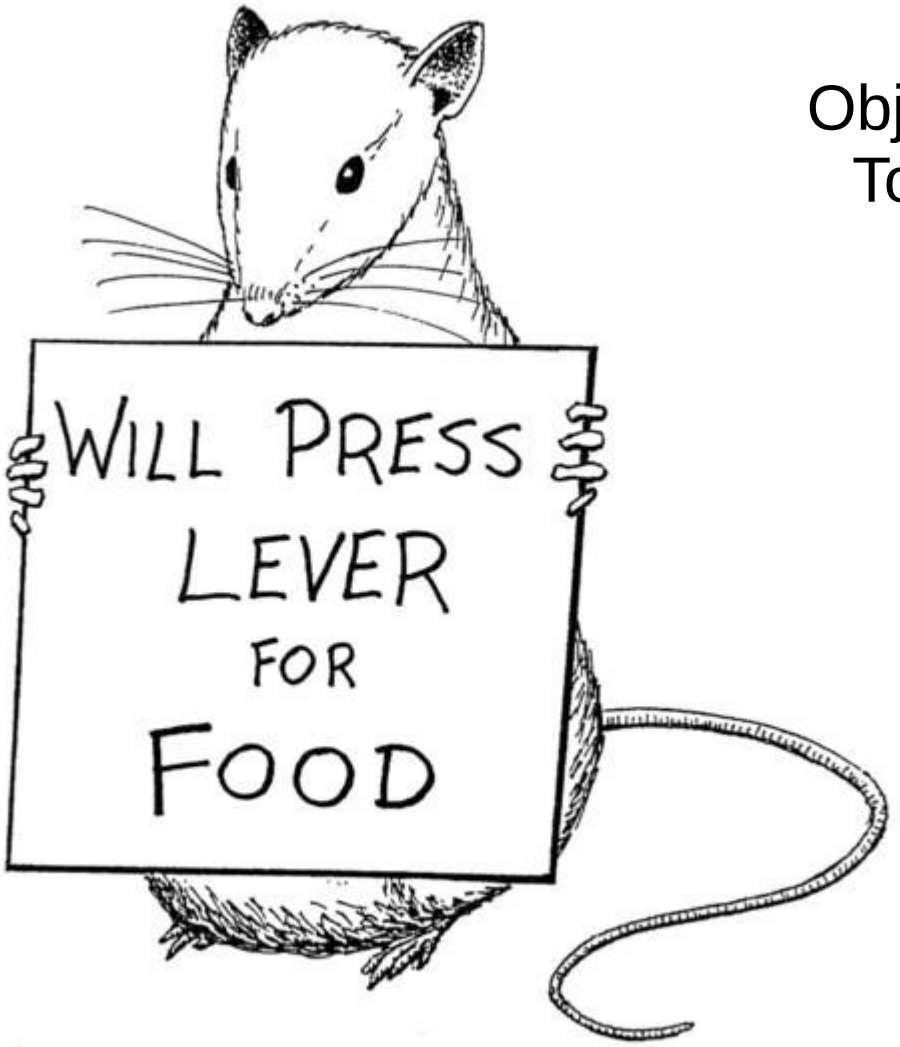
The MDP formalism



Classic MDP(Markov Decision Process)
Agent interacts with environment

- Environment states: $s \in S$
- Agent actions: $a \in A$
- State transition: $P(s_{t+1}|s_t, a_t)$
- Reward: $r_t = r(s_t, a_t)$

Optimal policy formalism



Objective:
Total reward

$$R_t = r_t + \gamma \cdot r_{t+1} + \gamma^2 \cdot r_{t+2} + \ldots + \gamma^n \cdot r_{t+n}$$

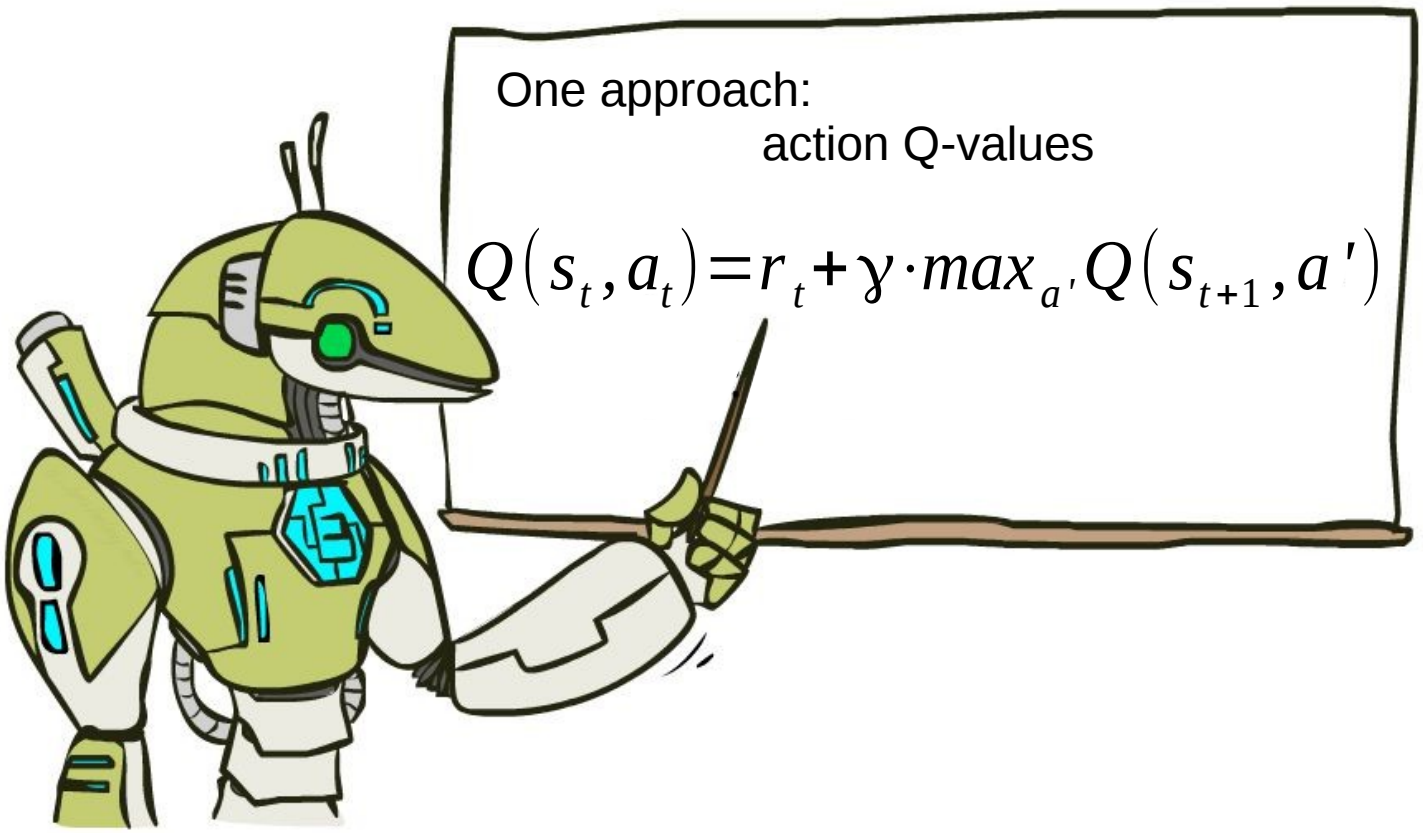
$$R_t = \sum_i \gamma^i \cdot r_{t+i} \qquad \gamma \in (0,1) const$$

γ ~ patience
Cake tomorrow is γ as good as now

Reinforcement learning:
- Find policy that maximizes the expected reward

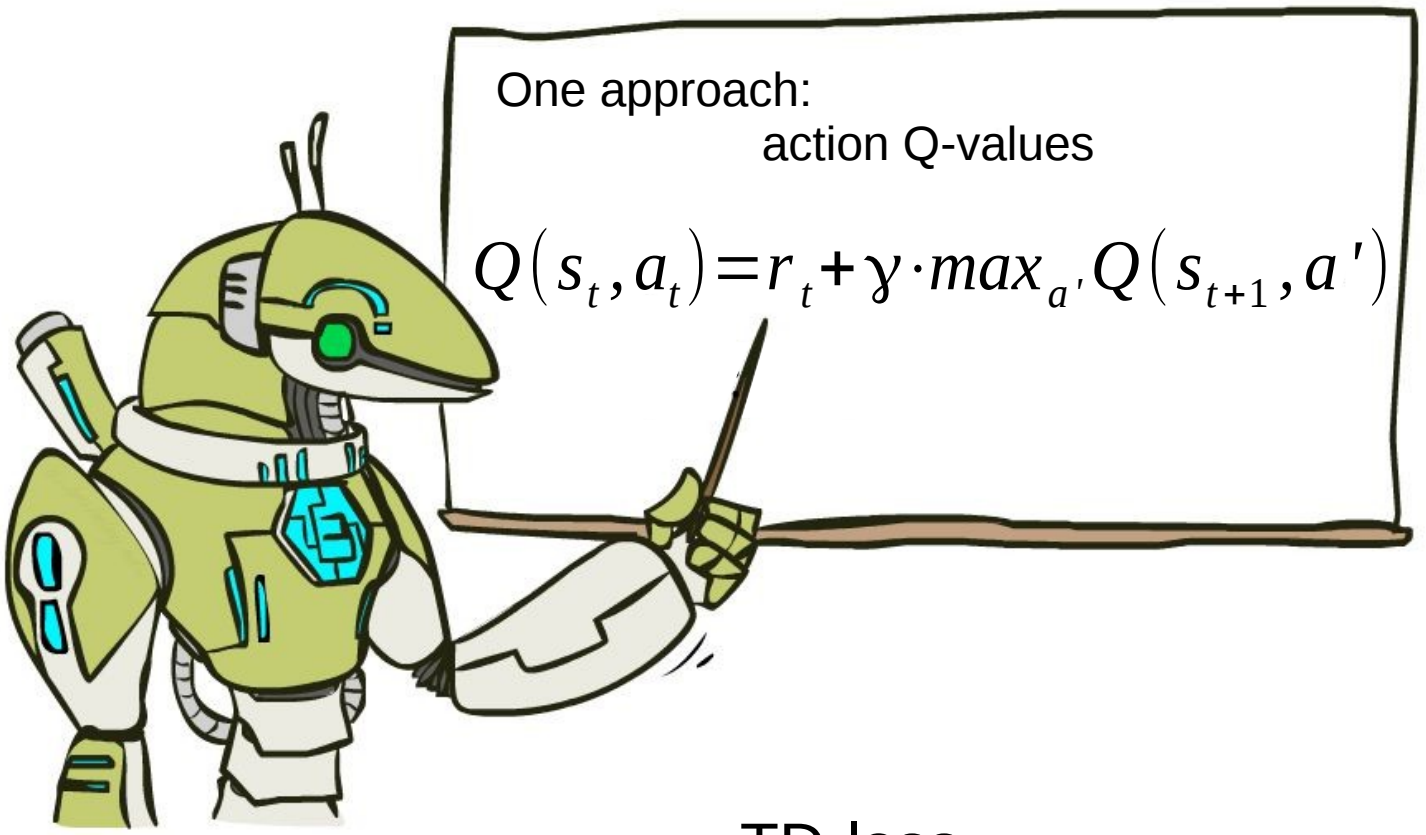
$$\pi = P(a|s) : E[R] \rightarrow max$$

Dat stupid qlearning again



Q(s,a) definition: an expected total reward R that agent gets from state **s** by taking action **a** and behaving optimally from next state.

Dat stupid qlearning again



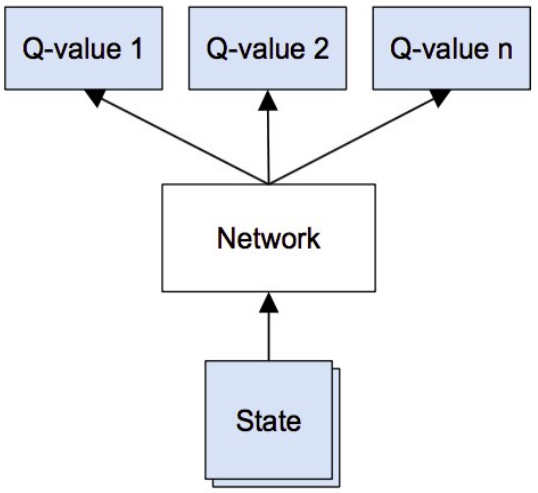
TD-loss

$$argmin_Q \left(Q(s_t, a_t) - [r_t + \gamma \cdot max_{a'} Q(s_{t+1}, a')] \right)^2$$

$$\pi(s) : argmax_a Q(s, a)$$

Dat stupid qlearning again

- Tabular RL
  - Maintain a table [s,a] → Q(s,a)
    - Impractical for large/continuous state spaces
- Approximate RL
  - Approximate Q(s,a)
    - Minimize the same loss
  - Linear regression
  - Neural networks
  - Could be anything



Disclaimer

Naively minimizing the TD loss on currently observed situations will lead to unstable training with poor global convergence or even divergence for nonlinear function approximations!

There are good practical ways to improve on that but that's a story for another day.

Or just google them: experience replay, asynchronuous rl, target networks, double dqn, etc.

Yes we can!

Can solve RL efficiently for

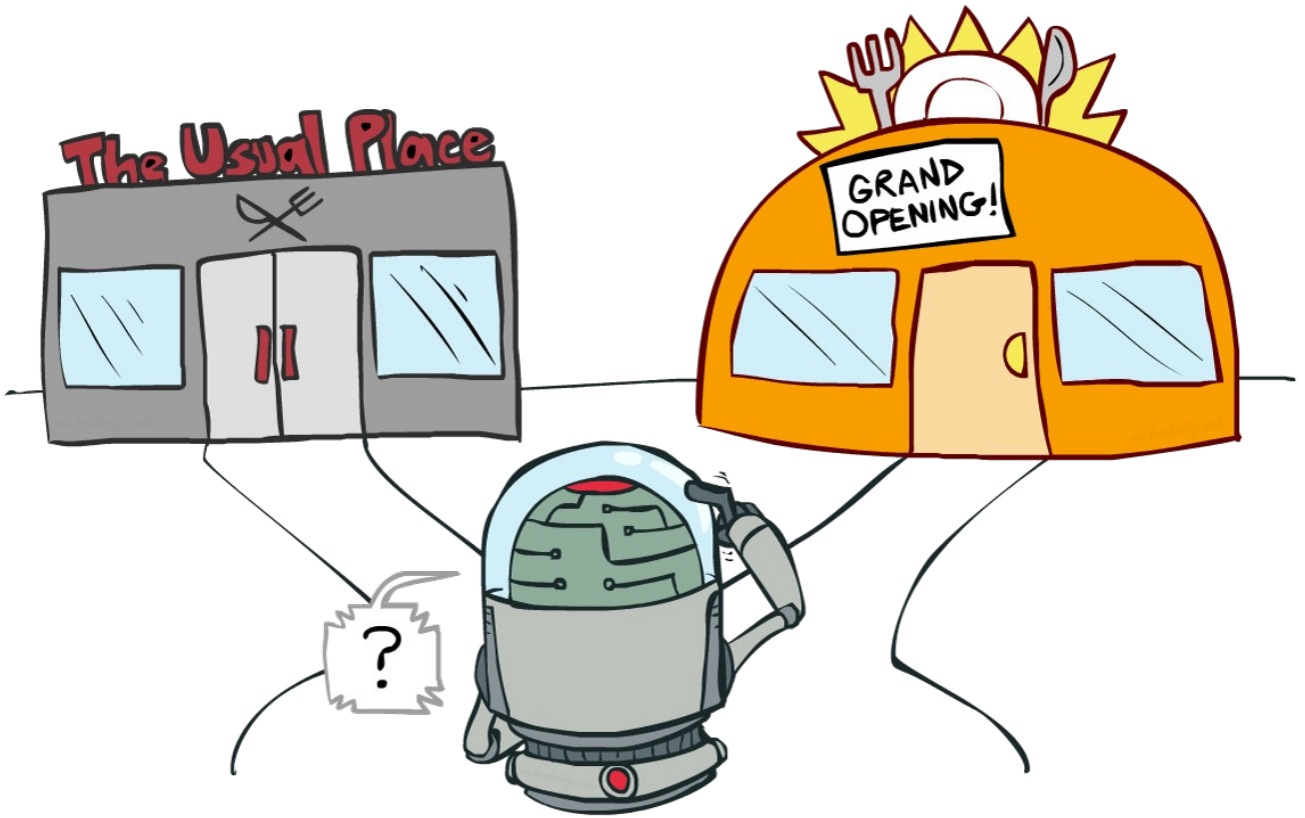
- Finite/Infinite MDP
- Full or partial observability
- Large or continuous state space
- Small OR structured OR continuous action space

Known limitations

- Frequent rewards
- Short delay between reward and its cause
- **Efficient exploration**

Exploration Vs Exploitation

Balance between using what you learned and trying to find something even better



Exploration strategies

Strategies:

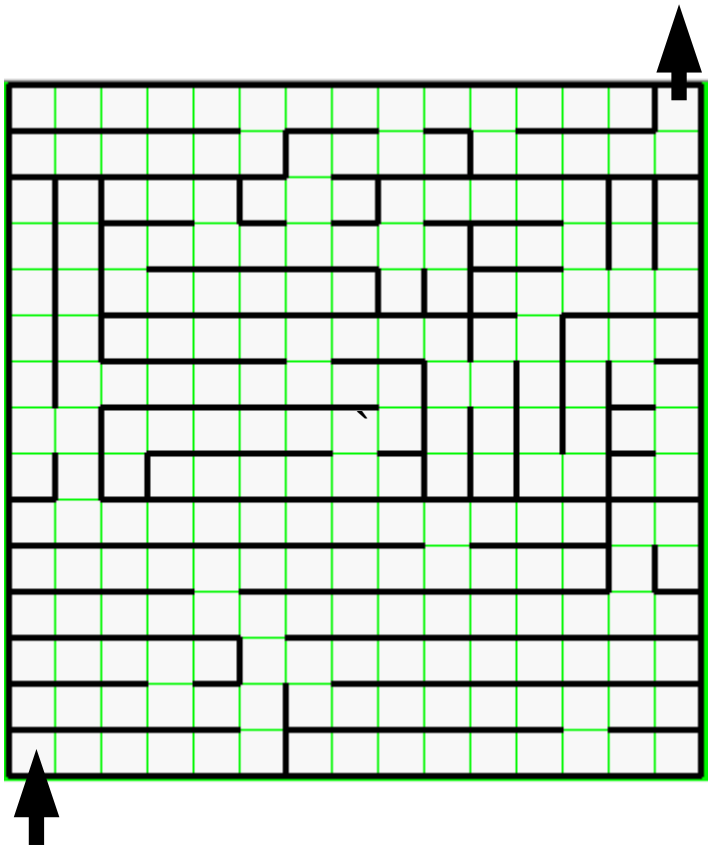
- **ε-greedy**
  - With probability ε take a uniformly random action;
  - Otherwise take optimal action.
- **Softmax**
  - Pick action proportionally to transformed Qvalues

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

- Some methods have a built-in exploration strategy (e.g. A2c)

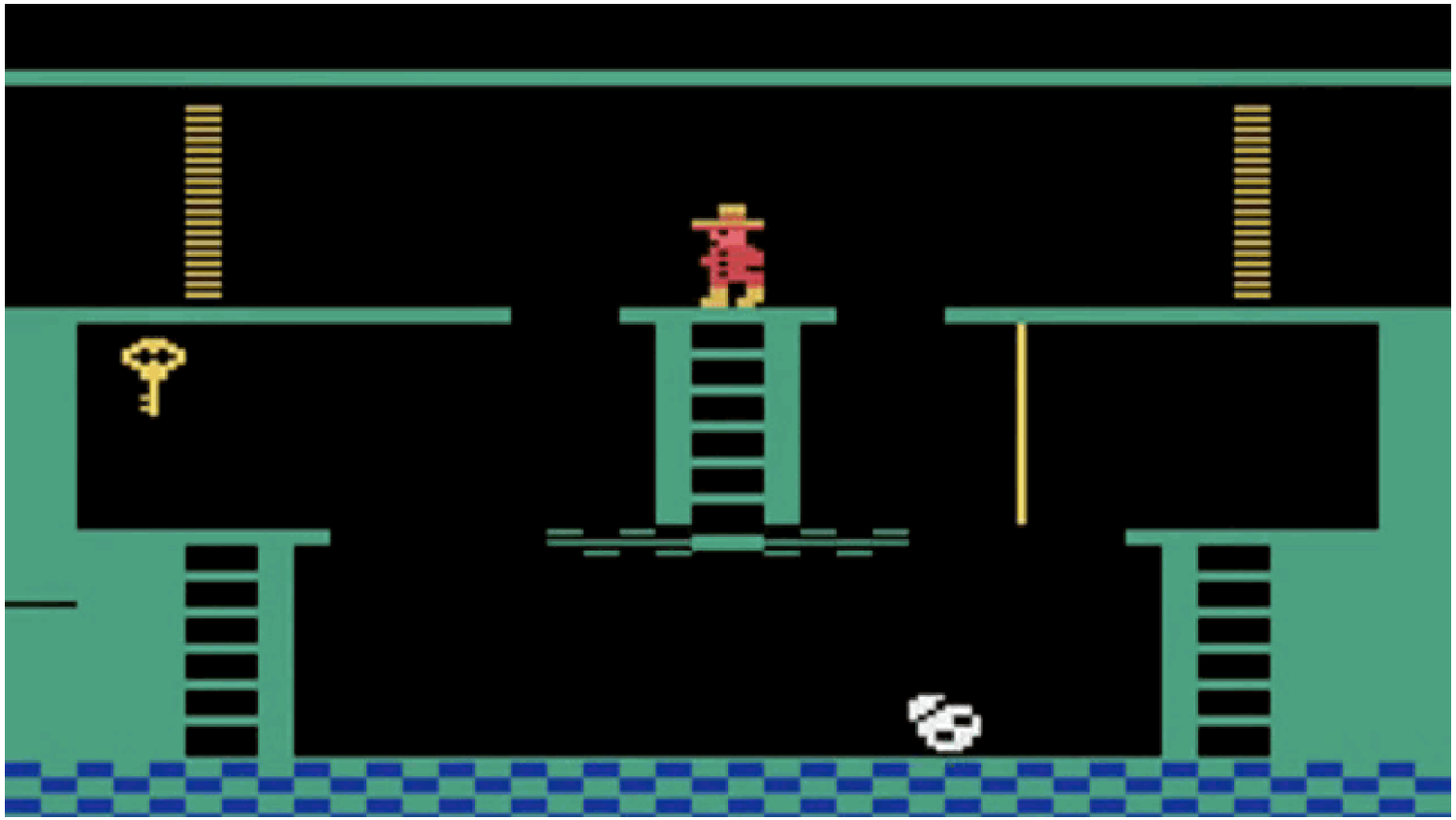
How many random actions does it take to exit this maze?

(you only get reward at the end)



Less than it takes to discover that

this game is solved by using the key



Less than it takes to learn how to

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

Except humans can learn these in less than a lifetime

Less than it takes to learn how to

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

We humans explore not with e-greedy policy!

BTW how humans explore?

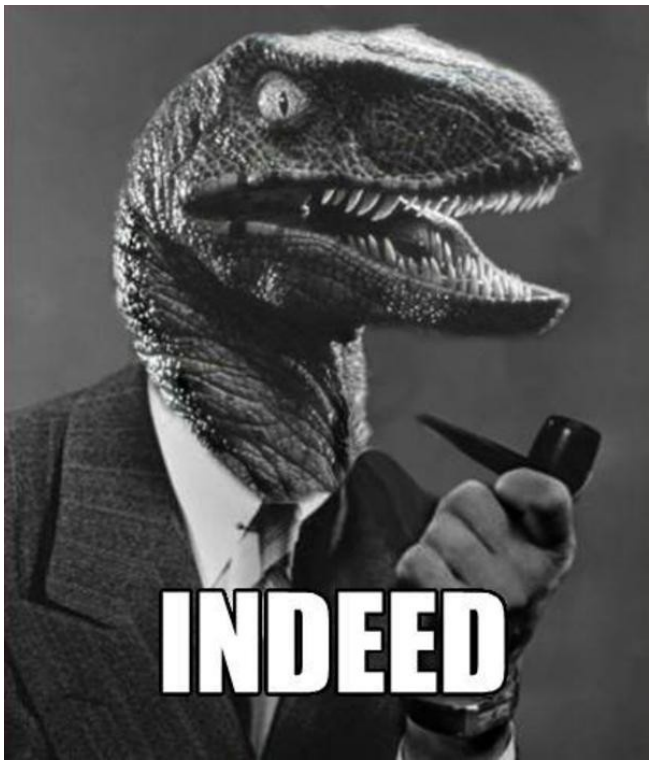
Whether some new particles violate physics

Vs

Whether you still can't fly by pulling yourself up

I got it!

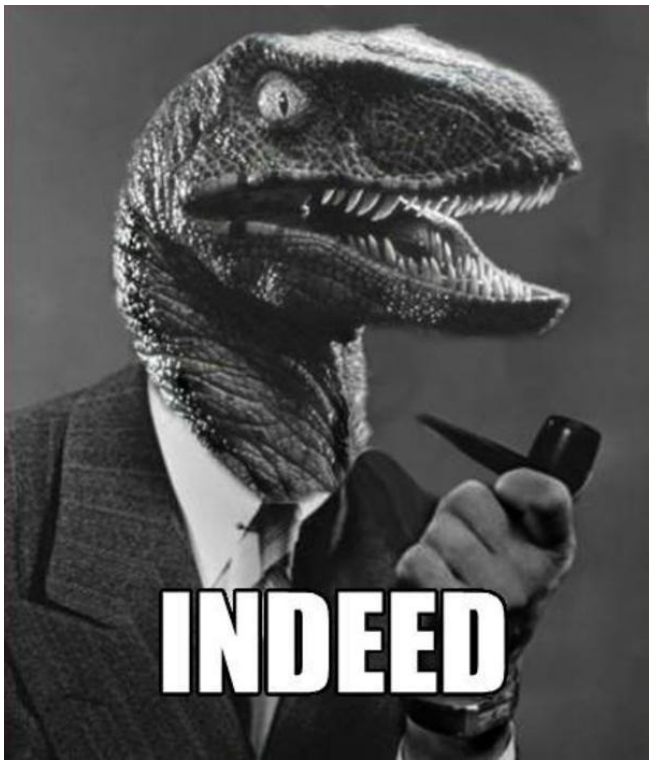
It's all about funding!



I got it!

It's all about funding!

(Just kidding)



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

Vime main idea

Add curiosity to the reward

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

Curiosity definition

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

Vime main idea

Environment model

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

Session

$$z_t=\langle s_{0,}a_{0,}s_{1,}a_{1,}...,s_t\rangle$$

Surrogate reward

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

curiosity

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

need proof for that last line?

Naive objective

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

where

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

Naive objective

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

Sample From MDP

Sample Somehow...

Model **Prior**

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

dunno

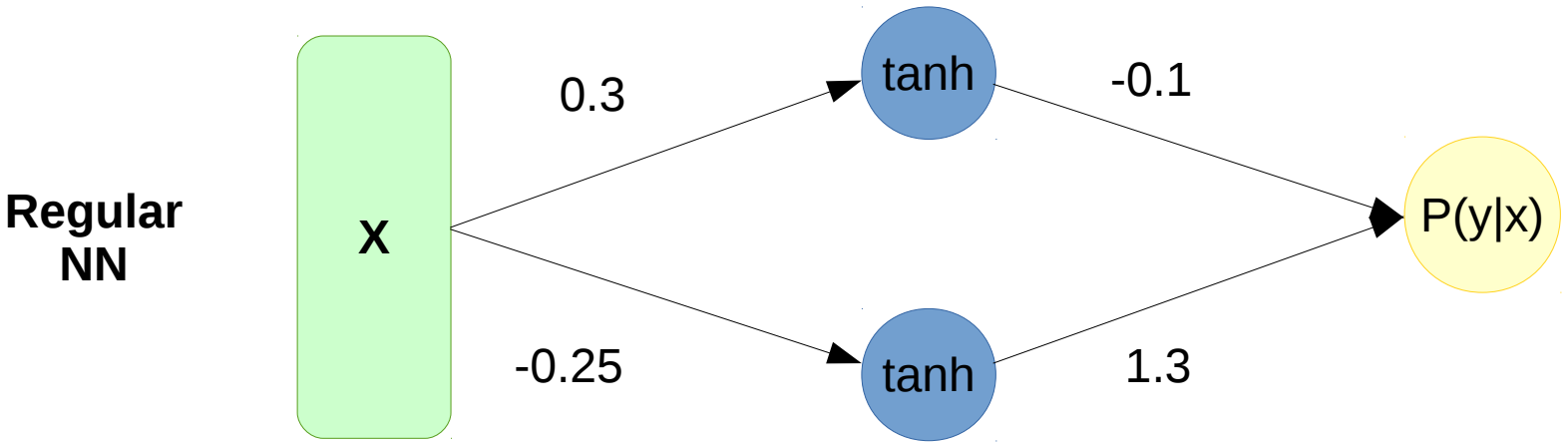
Better avoid computing P(theta|z) directly!


BAYES
TO SANTA ~~CLAUS~~

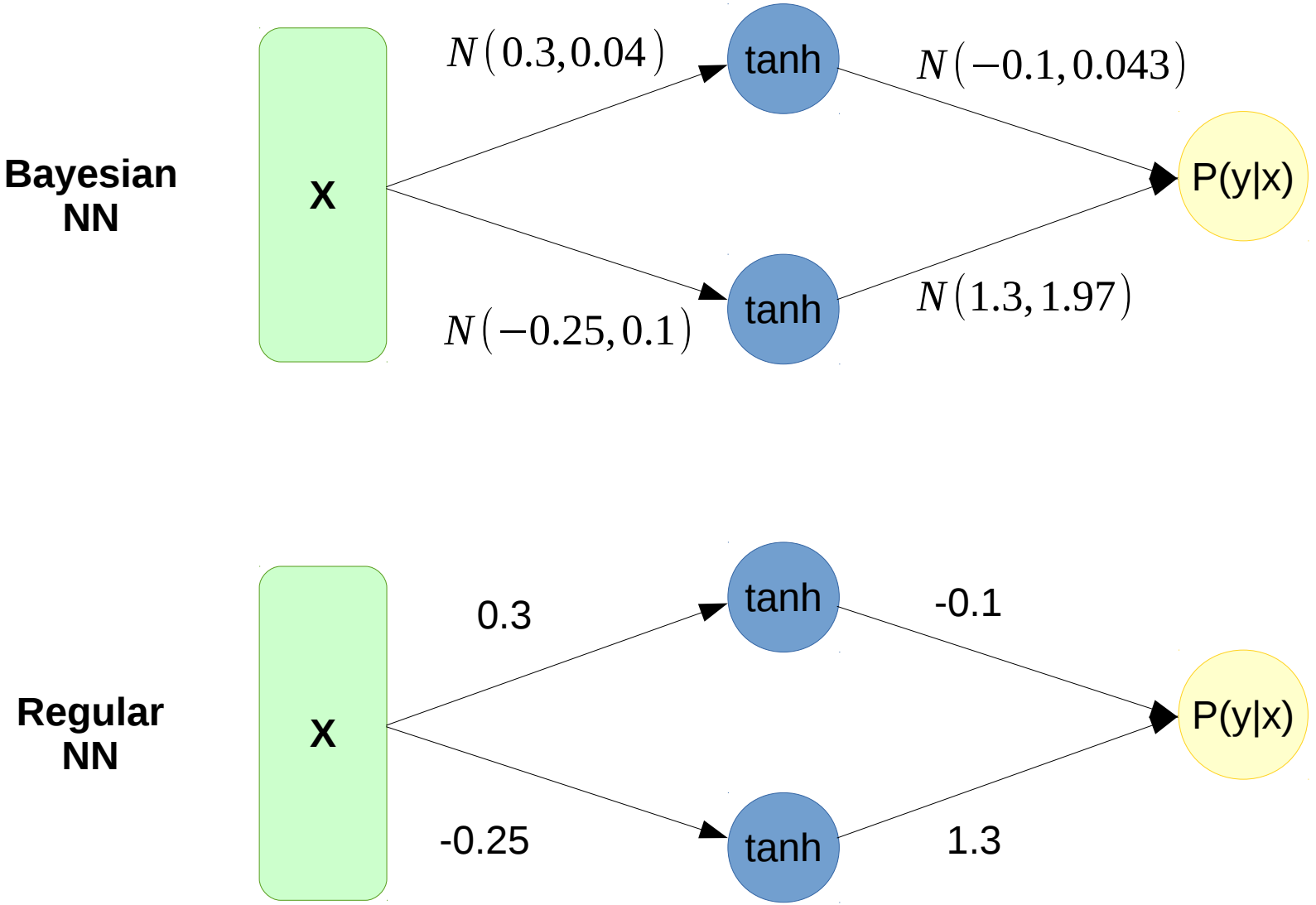
We want a model that

- predicts P(s'|z,a,theta)
- allows to estimate P(theta|D)
- we can sample from it

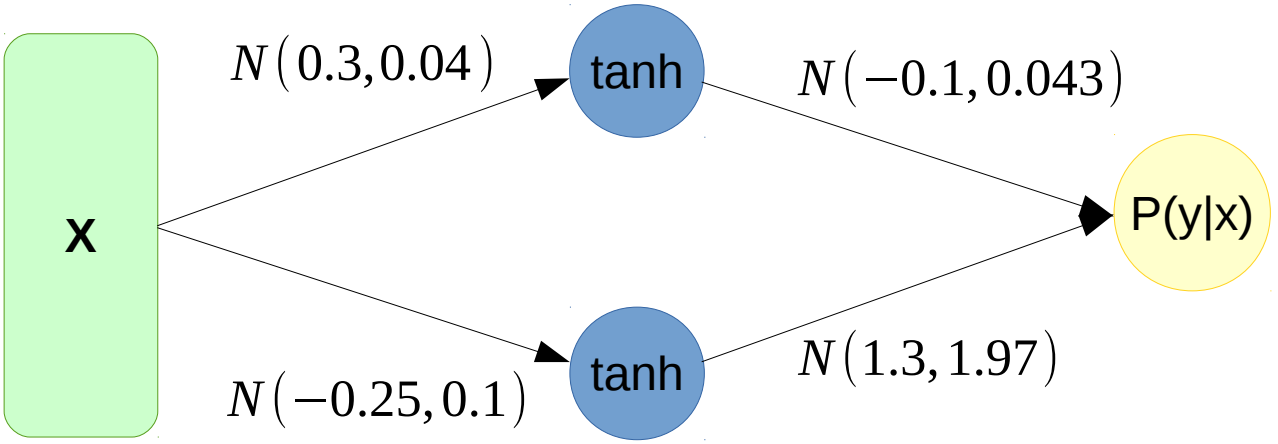
BNNs



BNNs


Bayesian NN
x
$N(0.3, 0.04)$
tanh
$N(-0.1, 0.043)$
P(y|x)
$N(-0.25, 0.1)$
tanh
$N(1.3, 1.97)$
Regular NN
x
0.3
tanh
-0.1
P(y|x)
-0.25
tanh
1.3

BNNs



Idea:

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

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

$$P(s'|s,a) = E_{\theta\sim q(\theta|\varphi)} P(s'|s,a,\theta)$$

BNNs

Idea:

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

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

$$P\left(s'|s,a\right)=E_{\theta\sim q(\theta|\varphi)}P\left(s'|s,a,\theta\right)$$

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

$$\mathring{\varphi}=argmax_{\varphi}E_{\theta\sim q(\theta|\varphi)}P\left(s'|s,a,\theta\right)$$

wanna explicit formulae?

Lower bound

$$\varphi_t = \underset{\varphi}{argmax}\left(-KL\left(q\left(\theta|\varphi\right)||p\left(\theta|z_t\right)\right)\right)$$

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

Lower bound

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

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

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

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

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

Lower bound

$$\varphi_t = \underset{\varphi}{argmax}\left(-KL\left(q\left(\theta|\varphi\right)||p\left(\theta|z_t\right)\right)\right)$$

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

Can we perform gradient ascent directly?

Reparameterization trick

$$\varphi_t = \underset{\varphi}{argmax}\left(-KL\left(q\left(\theta|\varphi\right)||p\left(\theta|z_t\right)\right)\right)$$

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

Use reparameterization trick

Using BNN

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

Better: local reparameterization trick (google it)

Vime objective

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

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

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

sample from env

sample from BNN

BNN

BNN last tick

Algorithm

Forever:

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

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

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

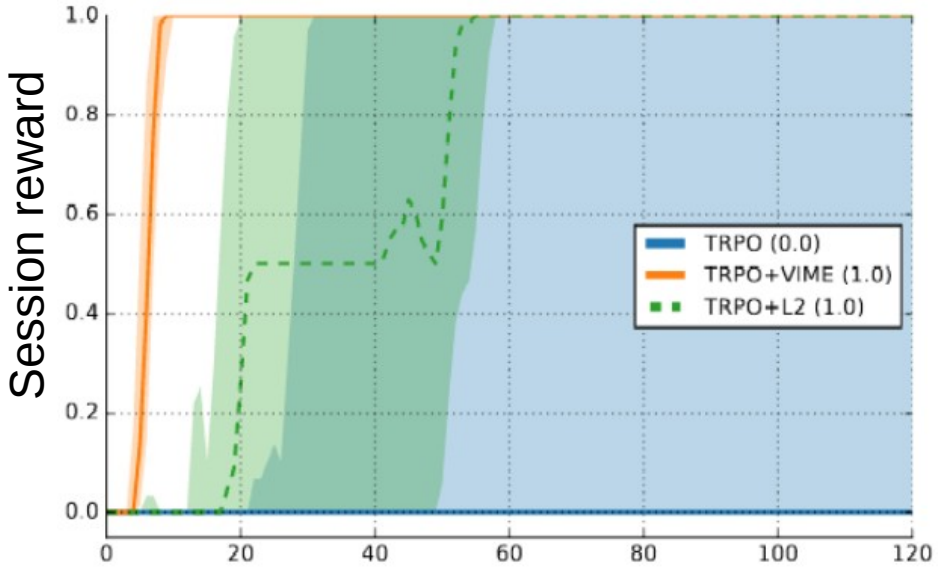
Dirty hacks

- **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

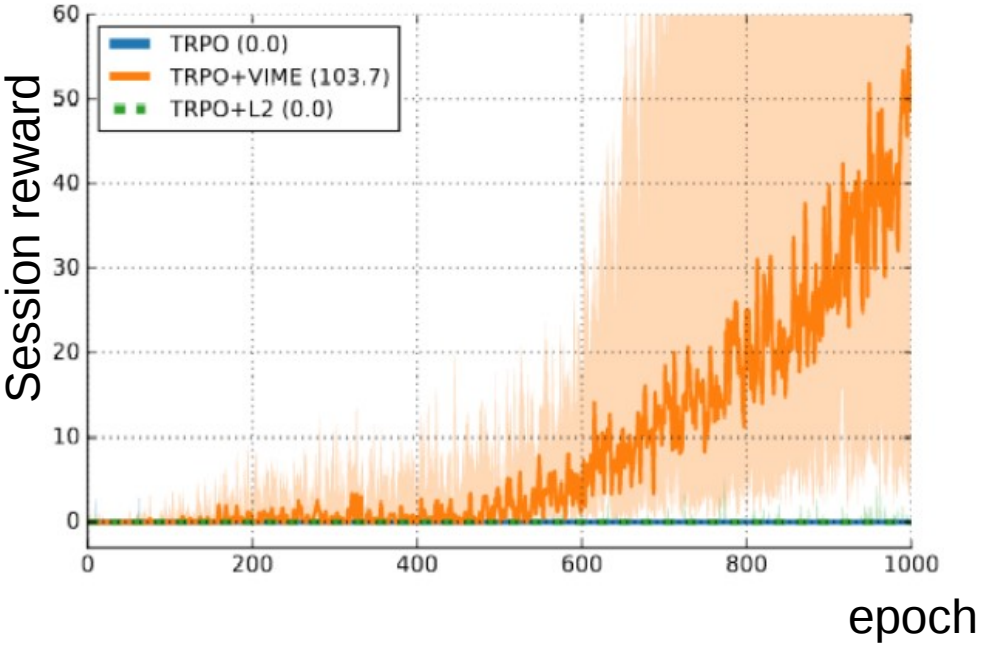
$$KL[q(\theta|\varphi')\|q(\theta|\varphi)] = \frac{1}{2}\sum_{i<|\theta|}[\left(\frac{\sigma_i'}{\sigma_i}\right)^2 + 2\log\sigma_i - 2\log\sigma_i' + \frac{(\mu_i' - \mu_i)^2}{\sigma_i^2}]$$

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

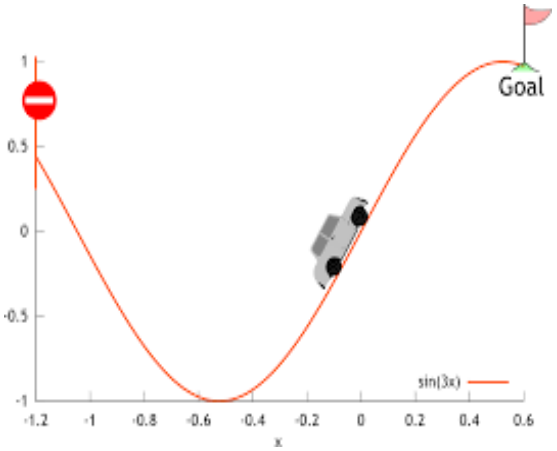
Results

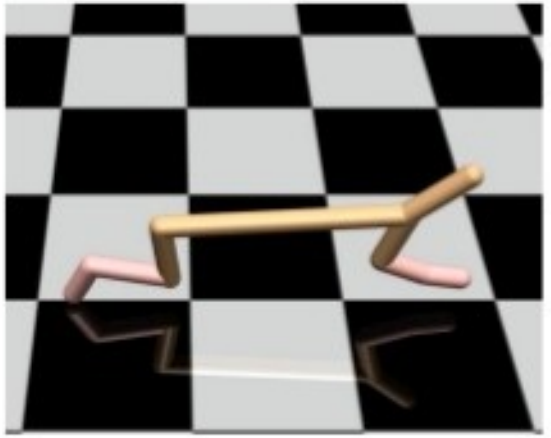


(a) MountainCar



(c) HalfCheetah





Results

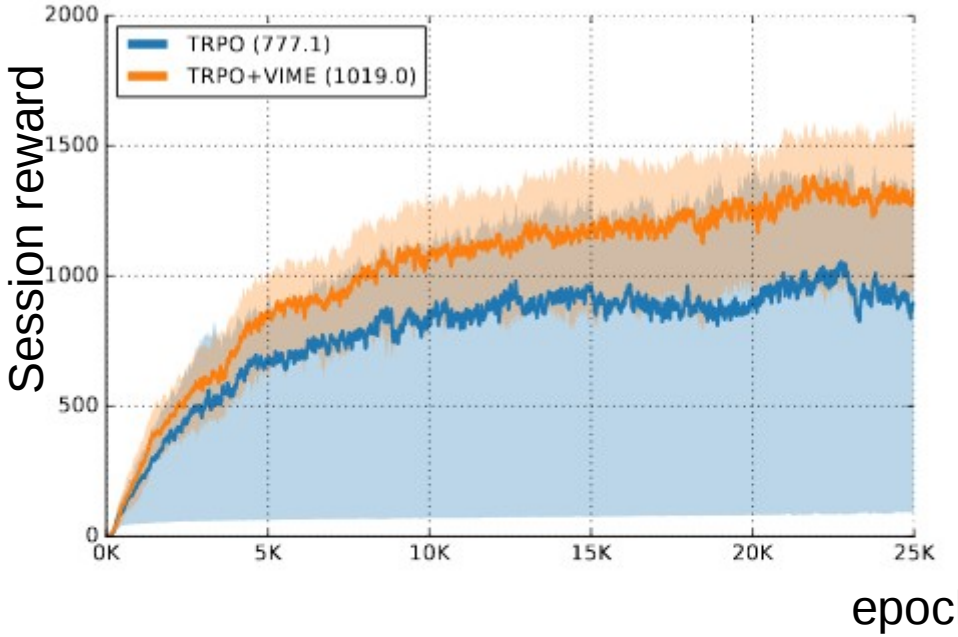
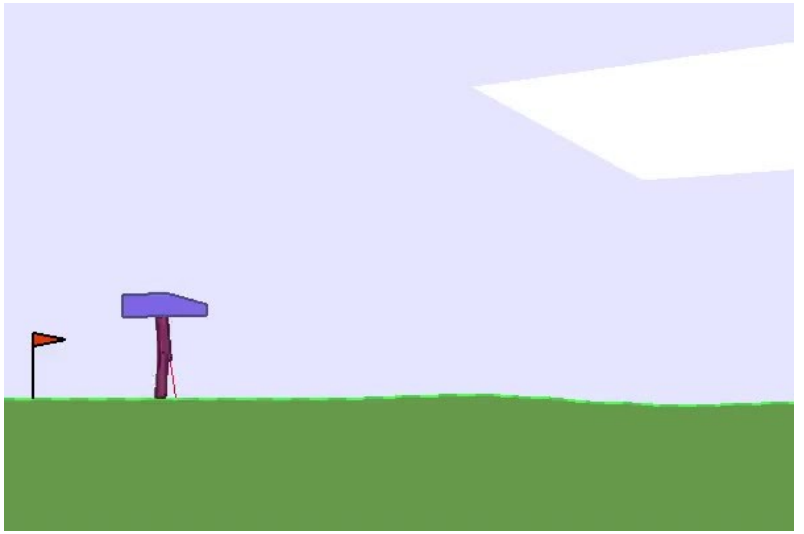
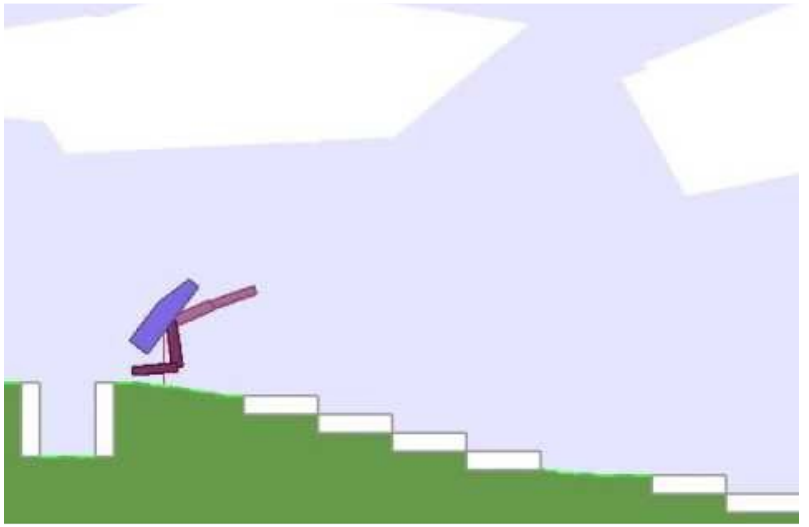


Figure 3: Performance of TRPO with and without VIME on the high-dimensional Walker2D locomotion task.

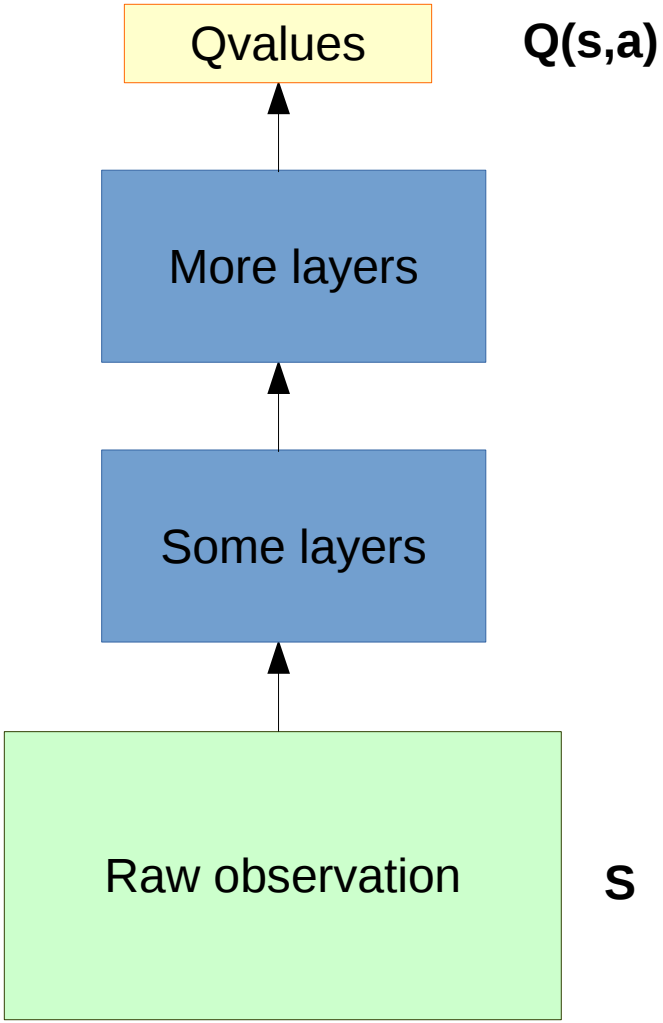




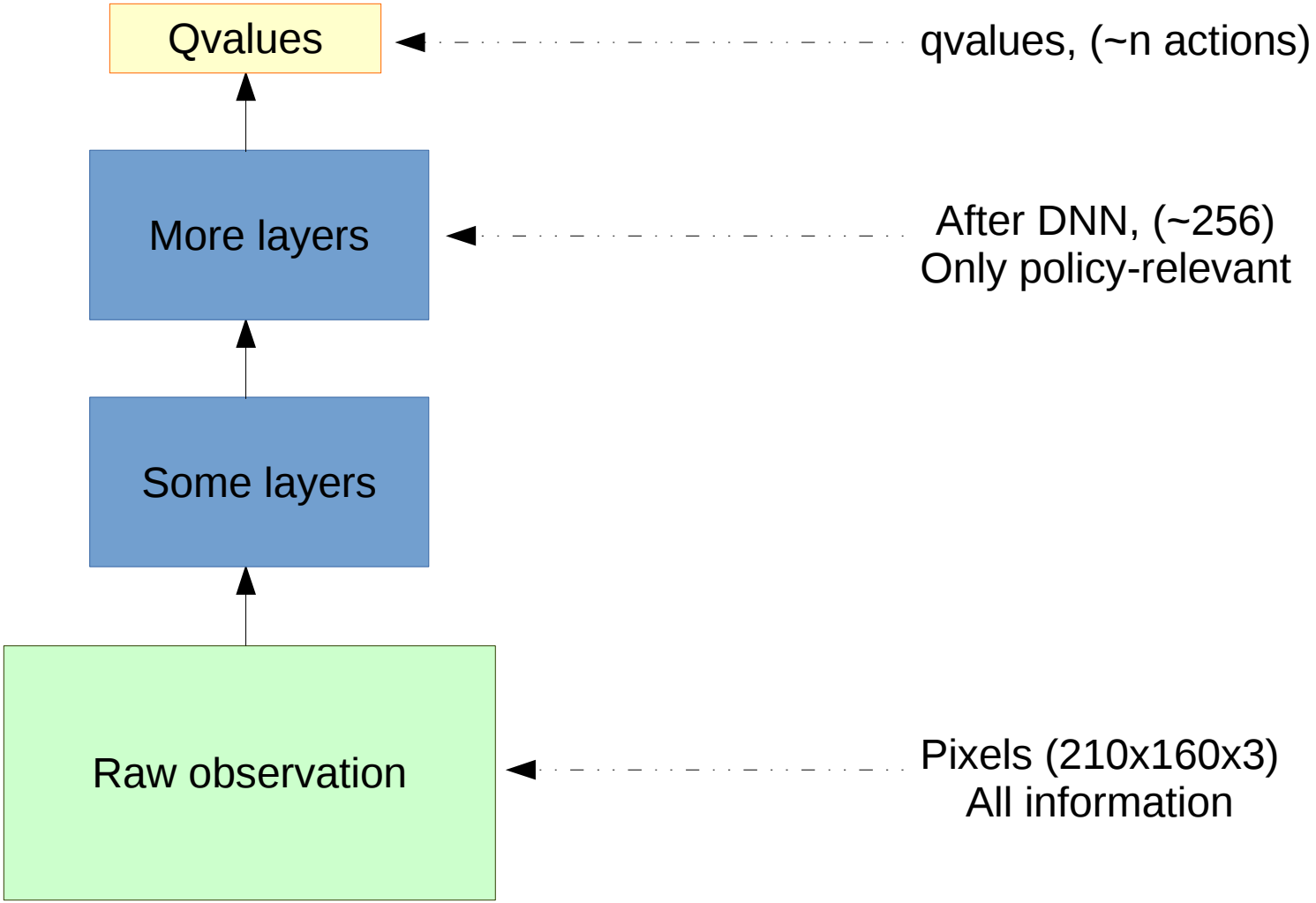
Pitfalls

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

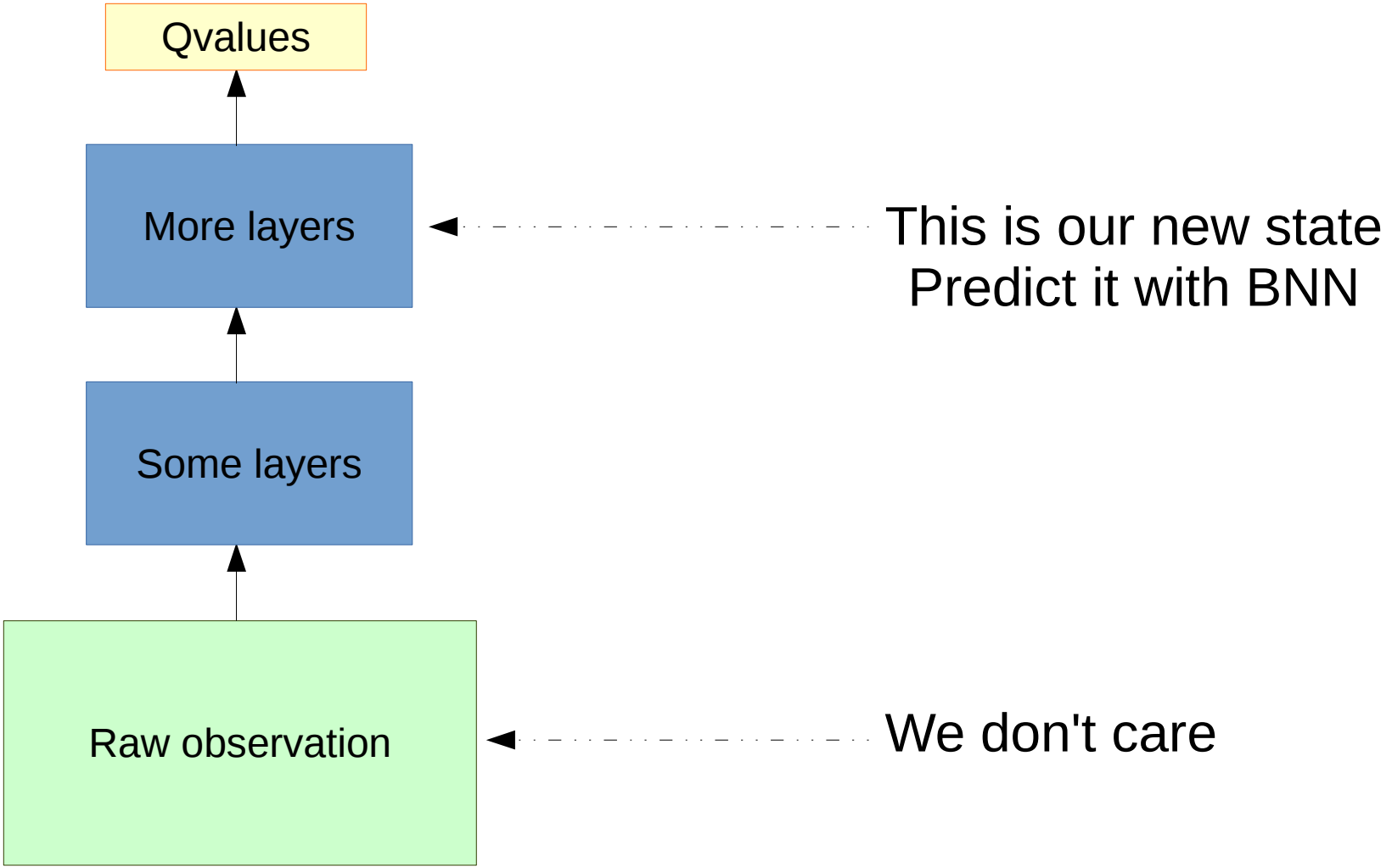
State = hidden NN activation



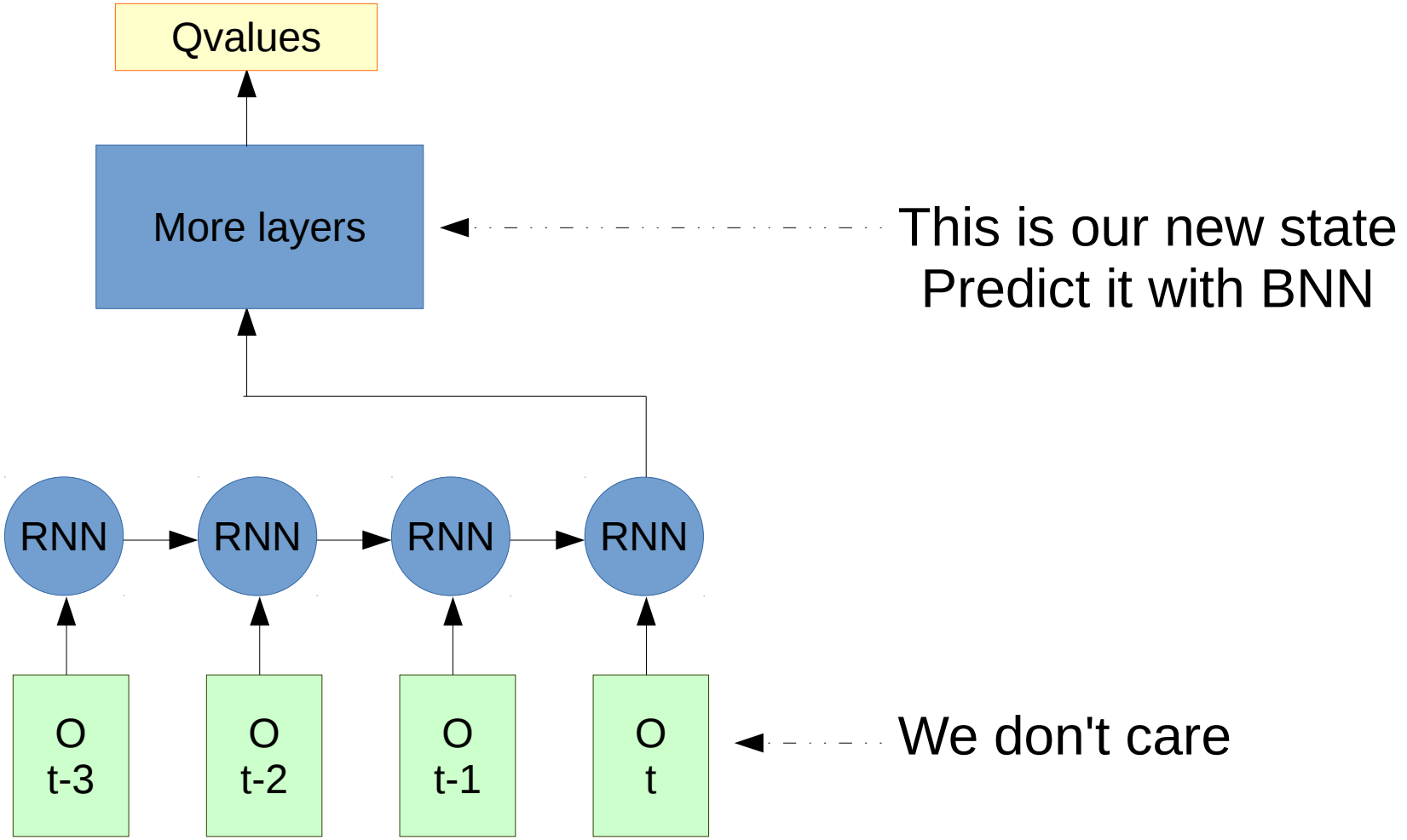
State = hidden NN activation



State = hidden NN activation



A case for POMDP



links

- https://arxiv.org/pdf/1605.09674v3.pdf
- http://mybinder.org/repo/justheuristic/vime
  - Will add pacman once it converges :P

Special thanks to

- **Arseniy Ashukha** – for reviewing (and covering my arse at HSE DL this very moment)
- **Maxim Kochurov** (ferrine@github) – for simple BNN @lasagne