

RL @ PicsArt

Day 4, part 1

Bandits, exploration, production hacks



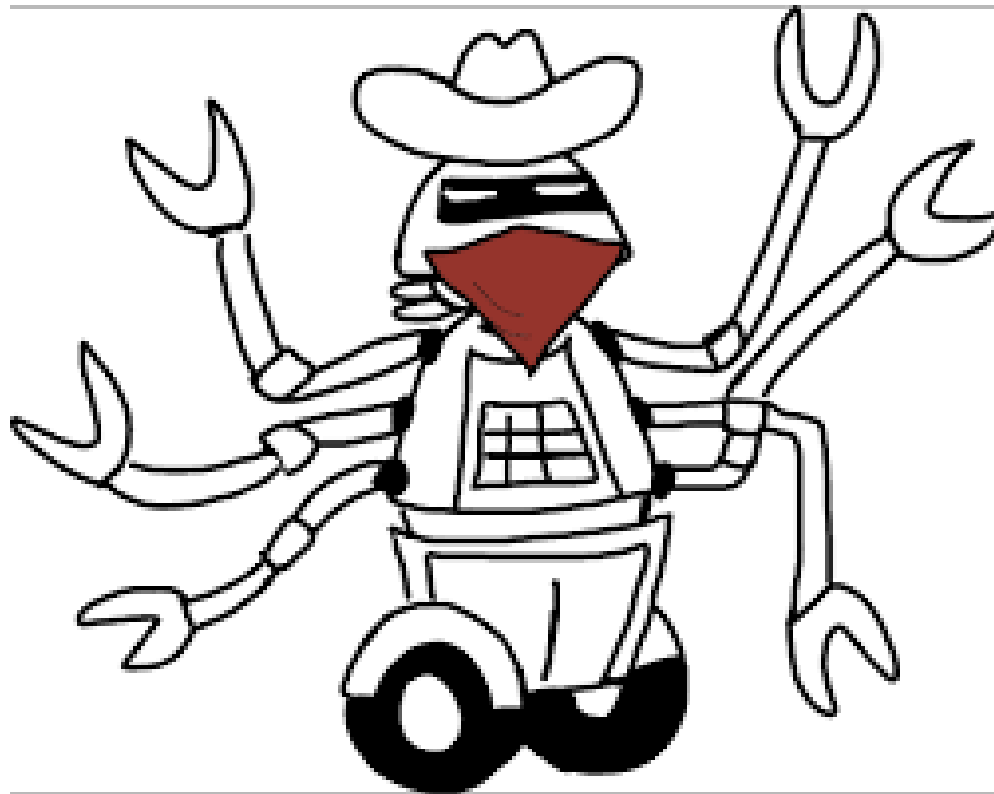
Yandex
Data Factory

LAMBDA 



**British Hedgehog
Preservation Society**

Multi-armed bandits



Multi-armed bandits



Multi-armed bandits

A simplified MDP with only one step



Why: it's simpler to explain exploration methods,
Formulae are shorter
(we can generalize to MDP if you wish)

What is: contextual bandit



Examples:

- banner ads (RTB)
- recommendations
- medical treatment

Basically 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

With convergence properties!

How to measure exploration

Bad idea: with convergence properties

Good idea: with \$\$\$ it brought/lost you

Regret of policy $\pi(a|s)$:

Consider an optimal policy, $\pi^*(a|s)$

Regret per tick = optimal – yours

$$\eta = \sum_t E_{s, a \sim \pi^{star}} r(s, a) - E_{s, a \sim \pi_t} r(s, a)$$

Finite horizon: $t < \max_t$ Infinite horizon: $t \rightarrow \inf$ ⁷

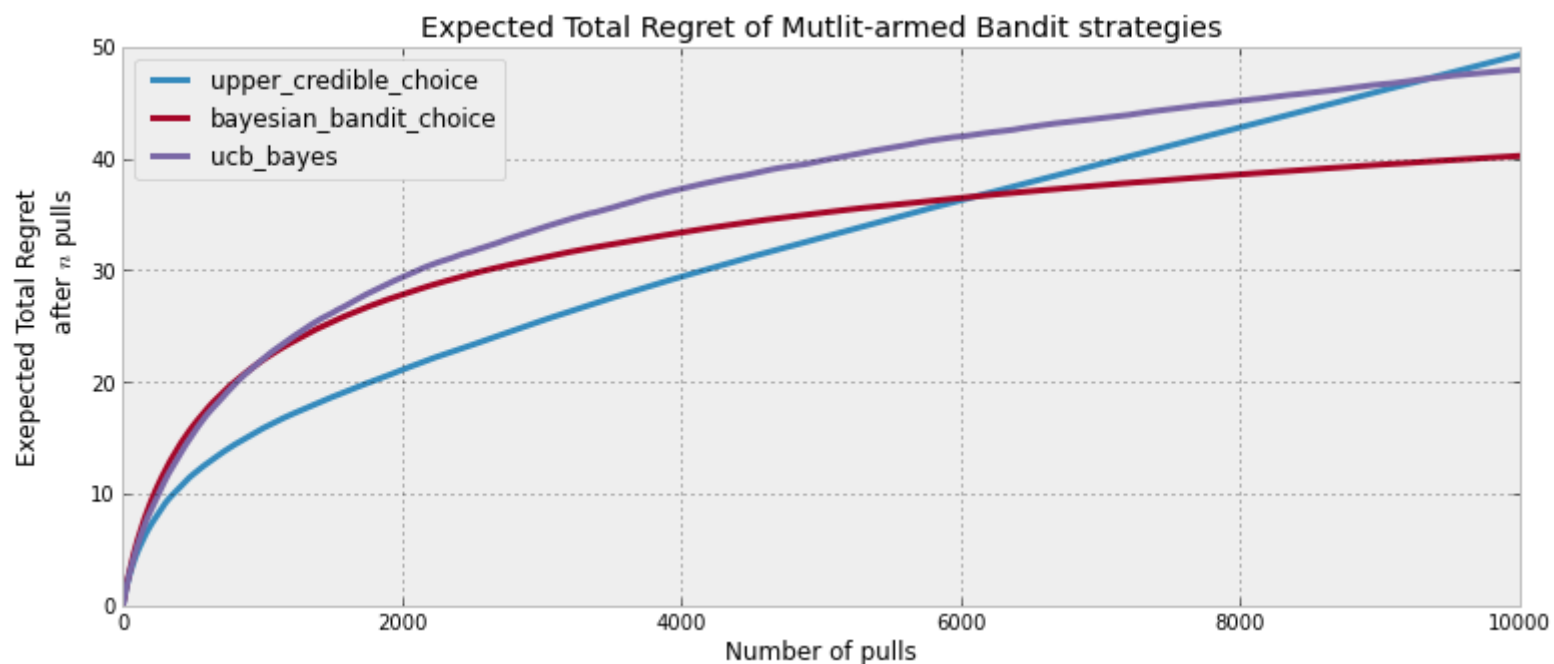
How to measure exploration

Bad idea: with convergence properties

Good idea: with \$\$\$ it brought/lost you

Regret of policy $\pi(a|s)$:

Regret per tick = optimal – yours



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

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

- Policy based: add entropy

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

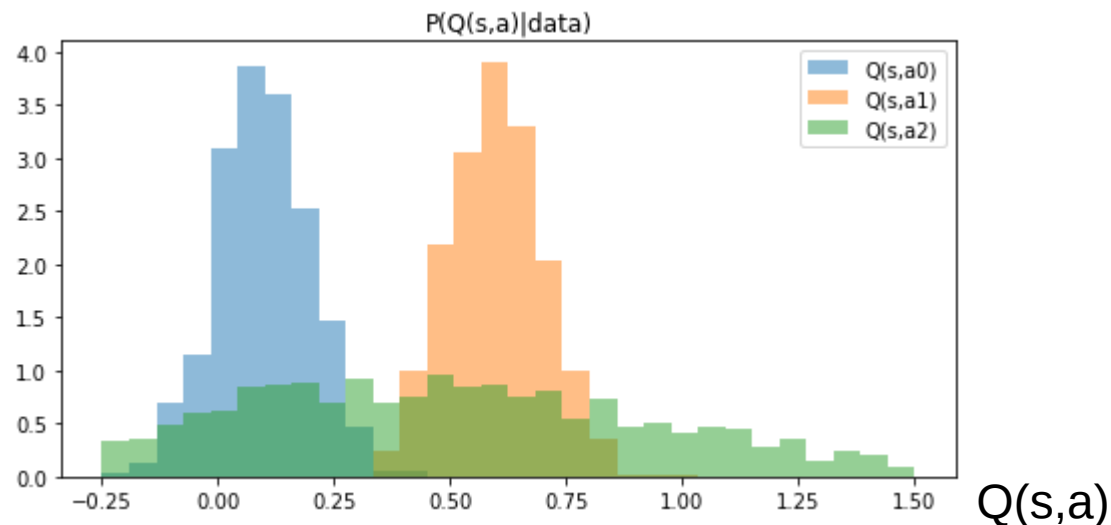
Whether you still can't fly by pulling your hair up



Uncertainty in returns

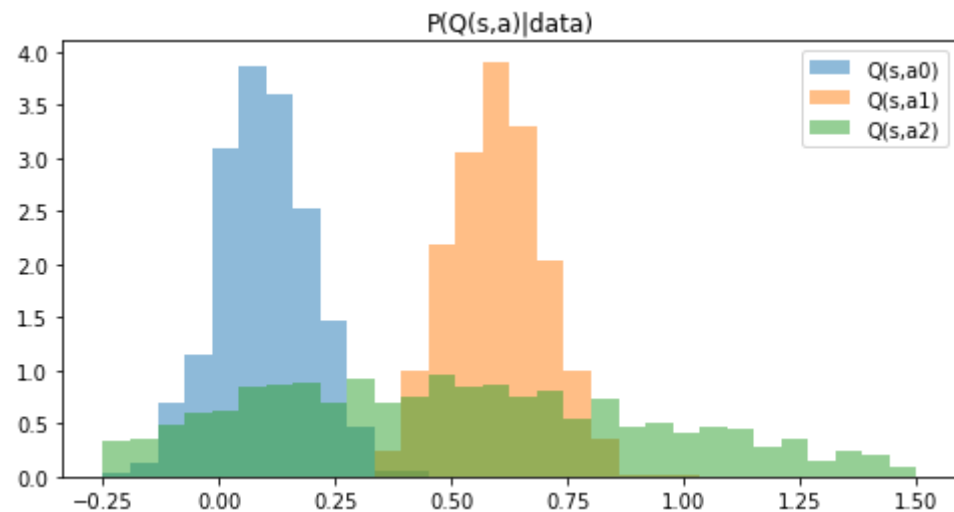
We want to try actions if we believe there's a chance they turn out optimal.

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?

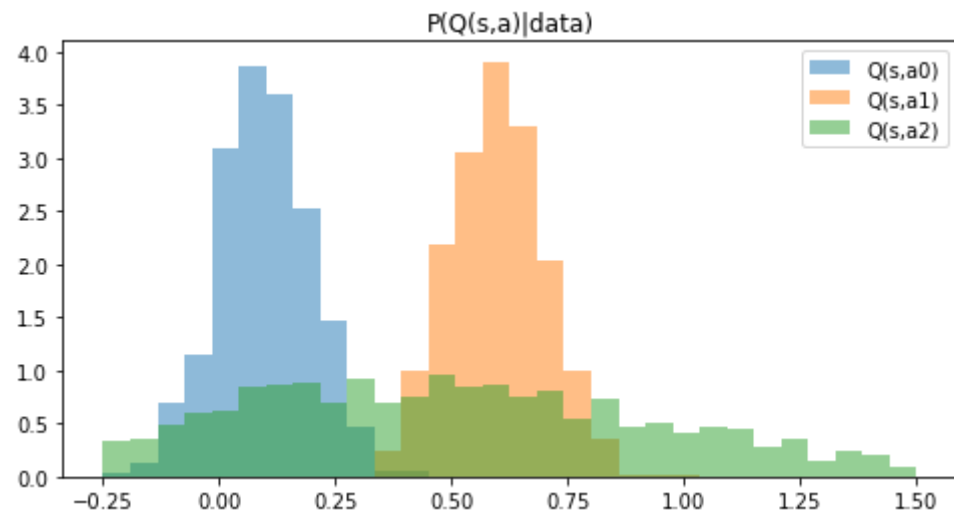


$Q(s,a)$

Thompson sampling

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

Takes a1 with $p \sim 0.65$, a2 with $p \sim 0.35$, a0 ~ never



$Q(s,a)$

Optimism in face of uncertainty

Idea:

Prioritize actions with uncertain outcomes!

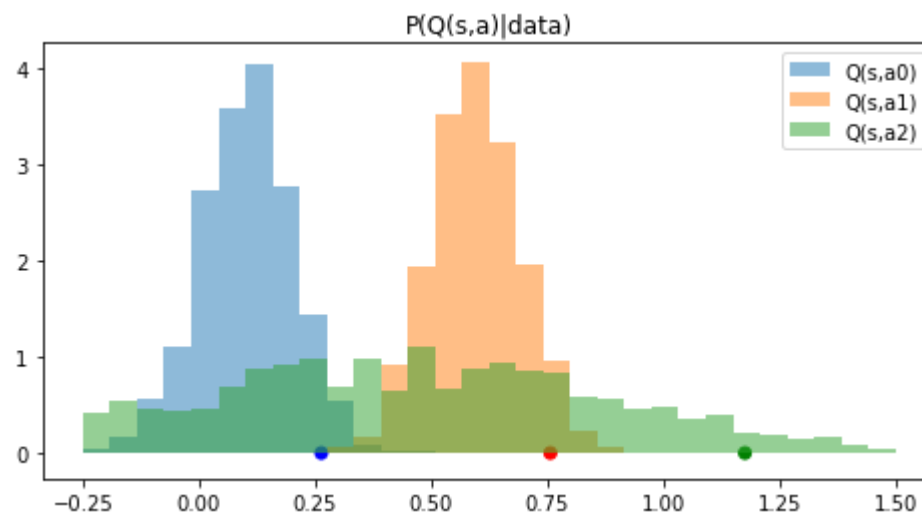
More uncertain = better.

Greater expected value = better

Math: try until upper confidence bound is small enough.

Optimism in face of uncertainty

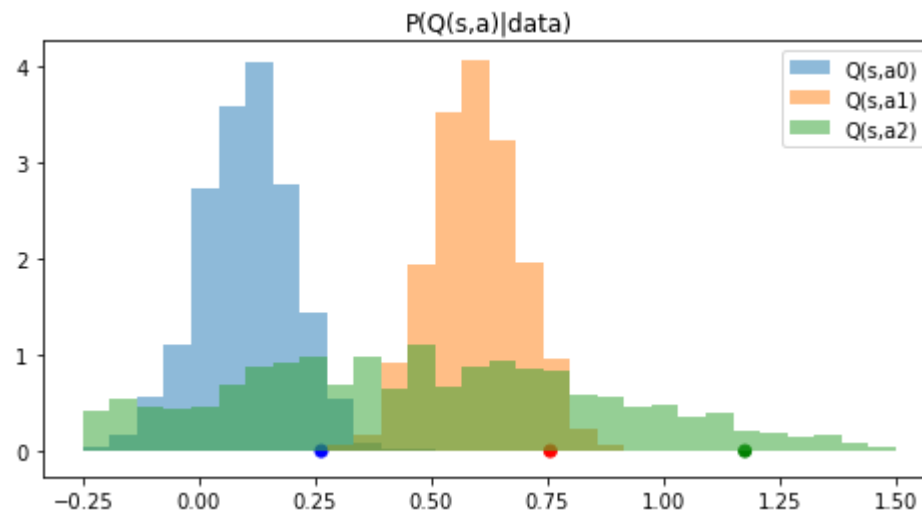
- Policy:
 - Compute 95% upper confidence bound *for each a*
 - Take action with highest confidence bound
 - What can we tune here to explore more/less?



$Q(s,a)$

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



$Q(s,a)$

Frequentist approach

There's a number of inequalities that bound

$$P(x > t) < \text{something}$$

- E.g. Hoeffding inequality

$$\mathbb{P}(S_n - \mathbb{E}[S_n] \geq t) \leq \exp\left(-\frac{2t^2}{\sum_{i=1}^n (b_i - a_i)^2}\right),$$

- Remember any others?

Frequentist approach

There's a number of inequalities that bound

$$P(x > t) < \text{something}$$

- E.g. Hoeffding inequality

$$\mathbb{P}(S_n - \mathbb{E}[S_n] \geq t) \leq \exp\left(-\frac{2t^2}{\sum_{i=1}^n (b_i - a_i)^2}\right),$$

- Remember any others?

(Chernoff, Chebyshev, over 9000)

Count-based exploration

UCB-1 for bandits

Take actions 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

Count-based exploration

UCB-1 for bandits

Take actions 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 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**

Count-based exploration

UCB-1 for bandits

Take actions in proportion to \tilde{v}_a

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

Upper bound
For bernoulli r

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

Count-based exploration

UCB-1 for bandits

Take actions 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 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 $P(Q|\text{data})$
- Take q -th percentile

What models can learn that?

Bayesian UCB

The usual way:

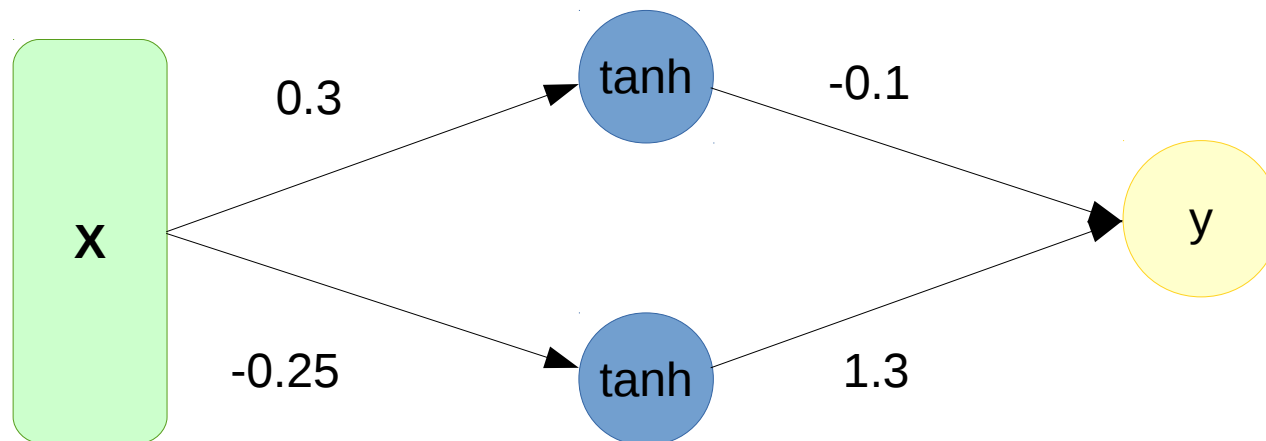
- Start from prior $P(Q)$
- Learn posterior $P(Q|\text{data})$
- Take q -th percentile

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

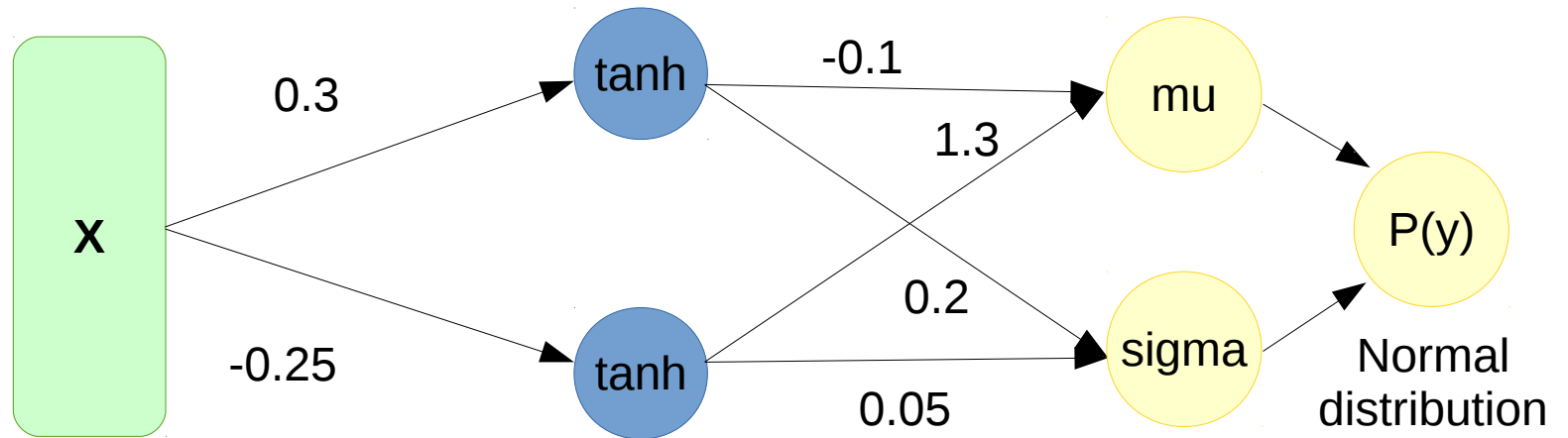
Approach 2: use bayesian neural networks

Parametric

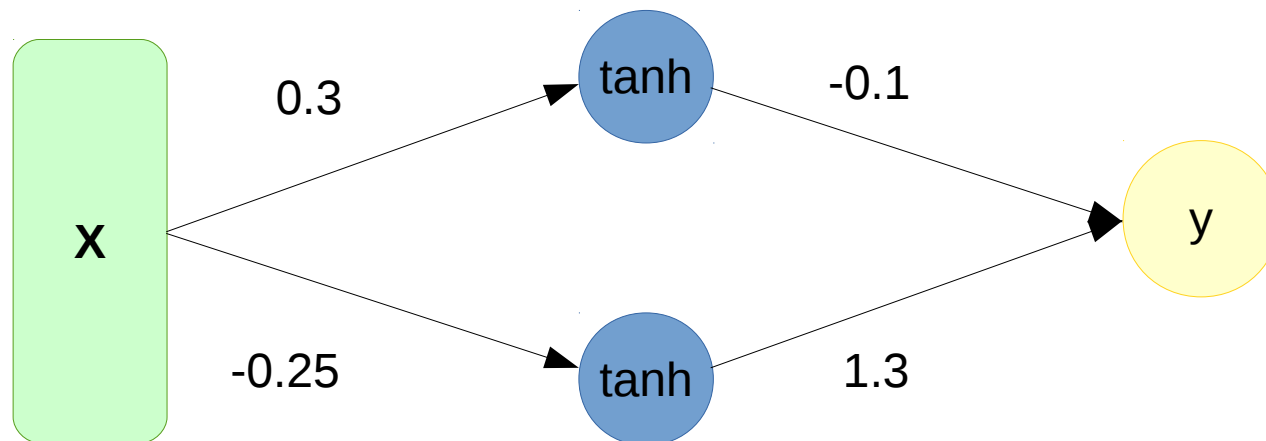
Regular
NN



Parametric

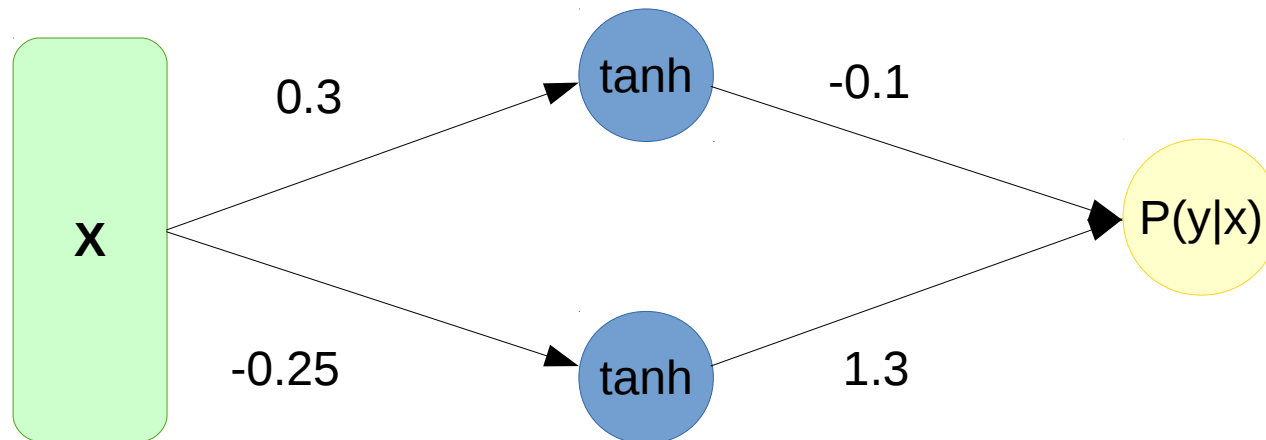


Regular
NN



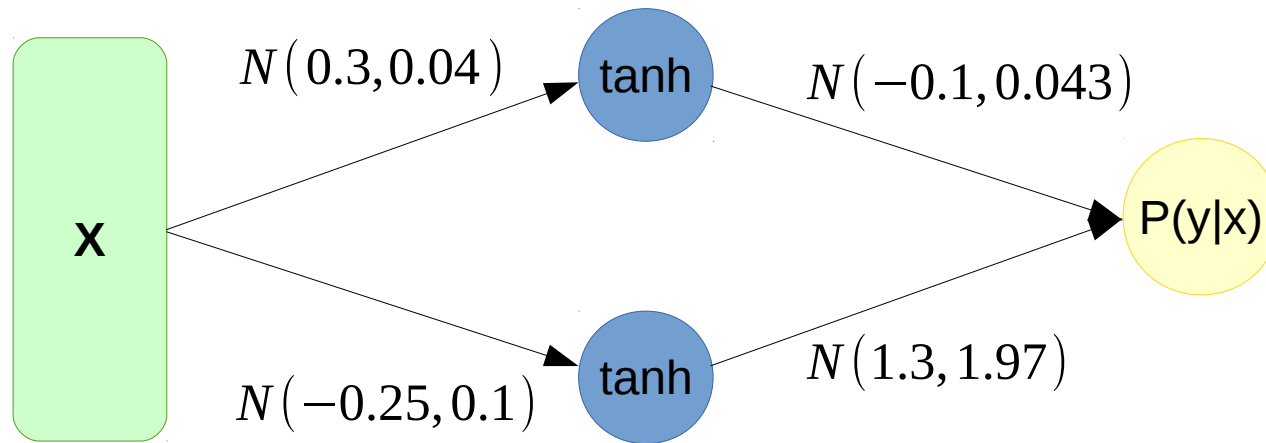
BNNs

Regular
NN

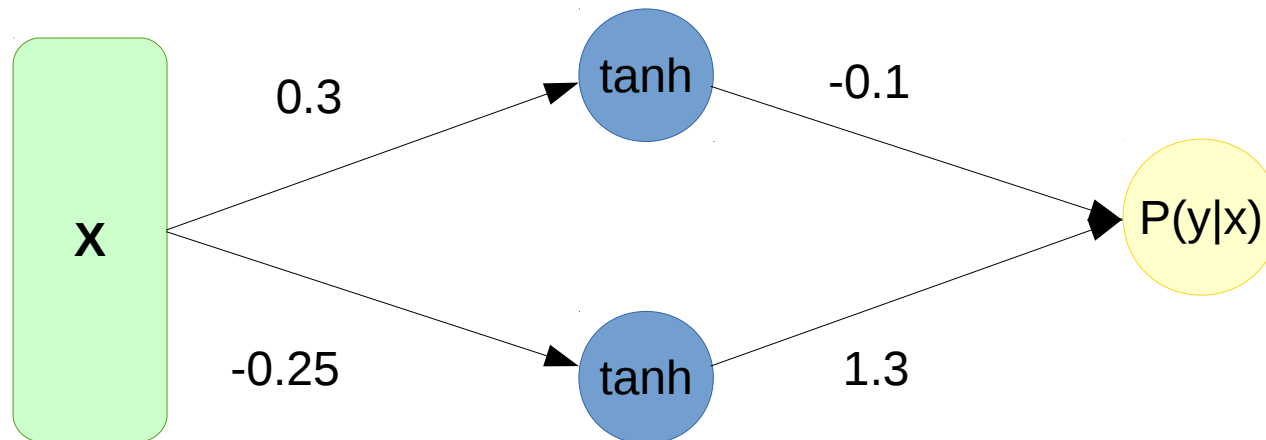


BNNs (for hackers)

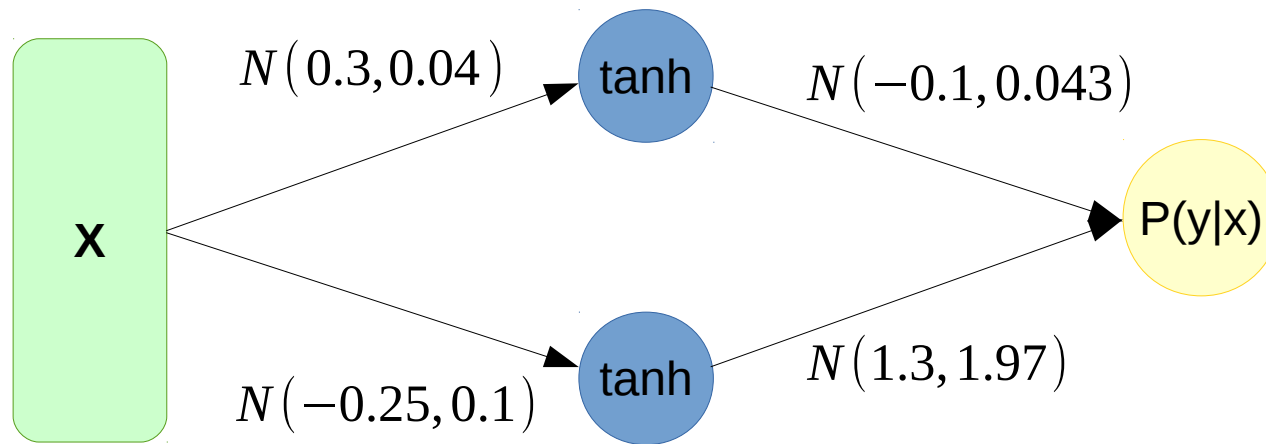
Bayesian
NN



Regular
NN



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

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

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

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

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

wanna explicit formulae?

Lower bound

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

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

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

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

loglikelihood

-distance to prior

+const

Lower bound

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

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

Can we perform gradient ascent directly?

Reparameterization trick

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

$$\underset{\varphi}{\operatorname{argmax}} \left(\underbrace{[E_{\theta \sim q(\theta|\varphi)} \log p(z_t|\theta)]}_{\text{Use reparameterization trick}} - KL(q(\theta|\varphi) \| p(\theta)) \right)$$

Use reparameterization trick

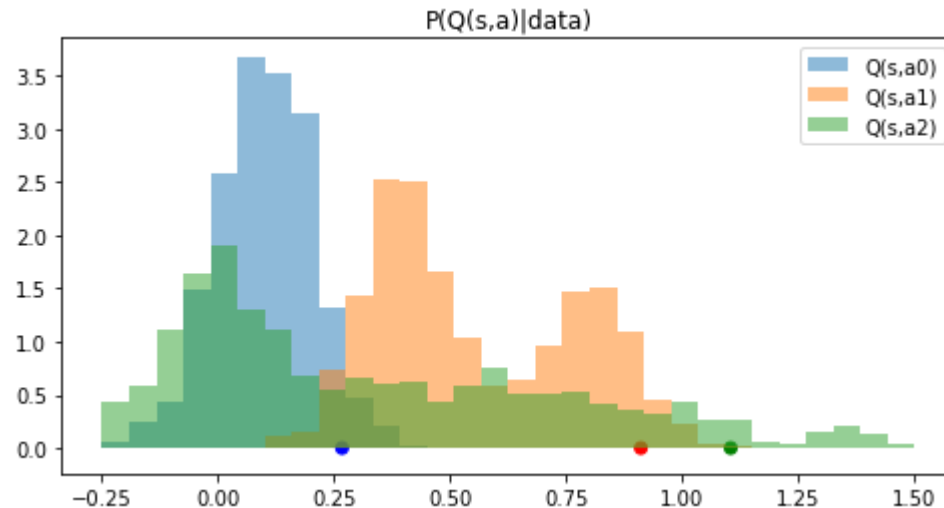
Using BNN

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

Better: local reparameterization trick (google it)

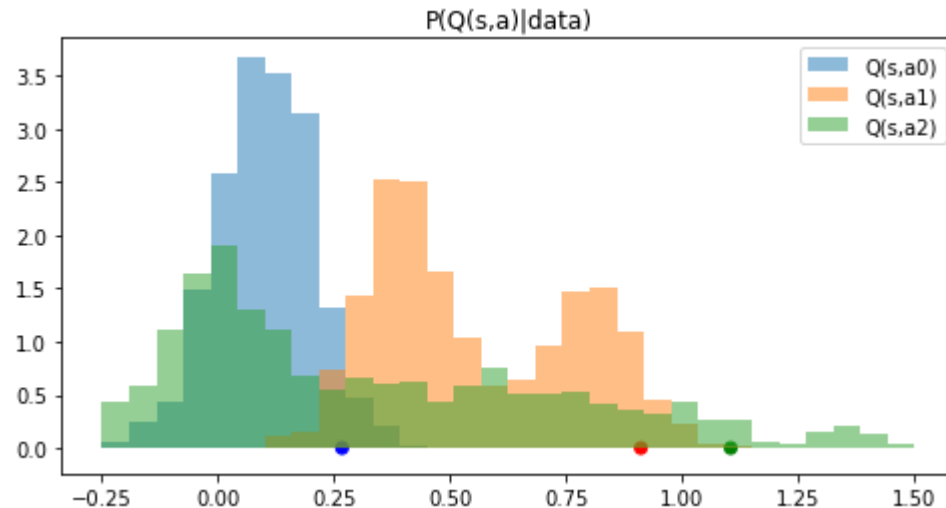
Using BNNs

- If you sample from BNNs
 - Can learn nonparametric/multimodal distribution
 - But it takes running network many times
 - Use empirical percentiles



Using BNNs

- If you sample from BNNs
 - Can learn nonparametric/multimodal distribution
 - **But it takes running network many times**
 - Use empirical 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



<us talking>

