RL @ PicsArt

Day 4, part 1

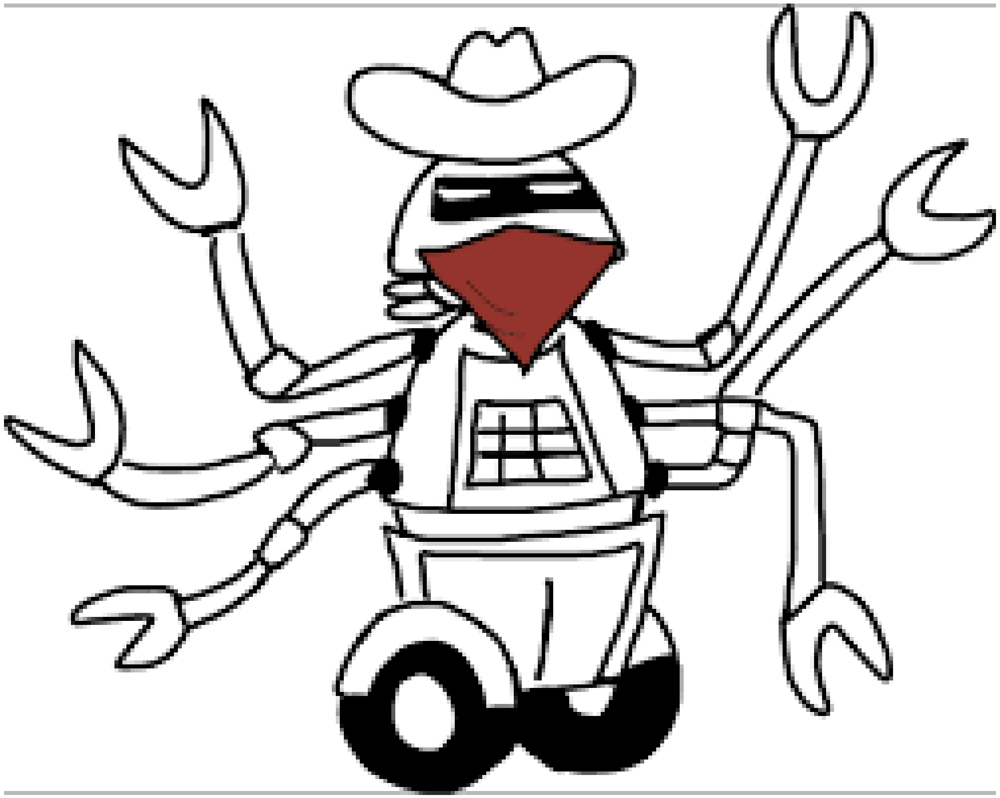
Bandits, exploration, production hacks

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

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

With convergence properties!

How to measure exploration

Bad idea: with convergence properties
Good idea: with $$$ it brought/lost you

Regret of policy π(a|s)**:**

Consider an optimal policy, π*(a|s)

Regret per tick = optimal – yours

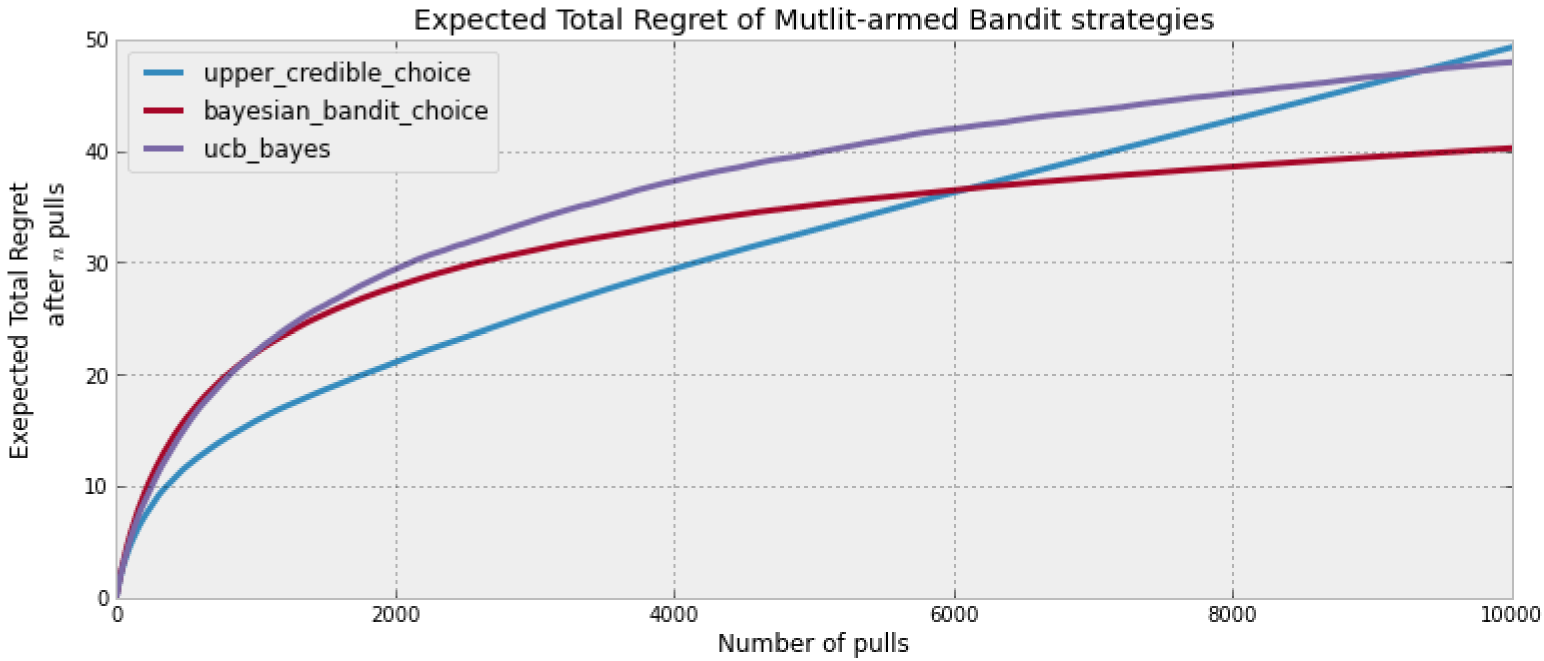
$$\eta = \sum_t \underset{s,a \sim \pi^{star}}{E} r(s,a) - \underset{s,a \sim \pi_t}{E} r(s,a)$$

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

How to measure exploration

Bad idea: with convergence properties
Good idea: with $$$ it brought/lost you

Regret of policy π(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

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

- 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

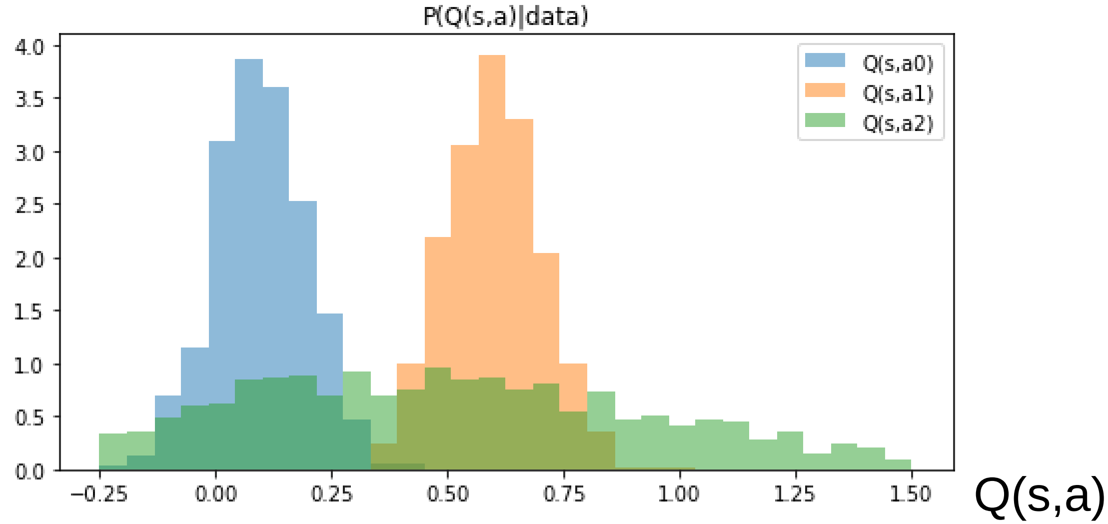
Vs

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

Uncertainty in returns

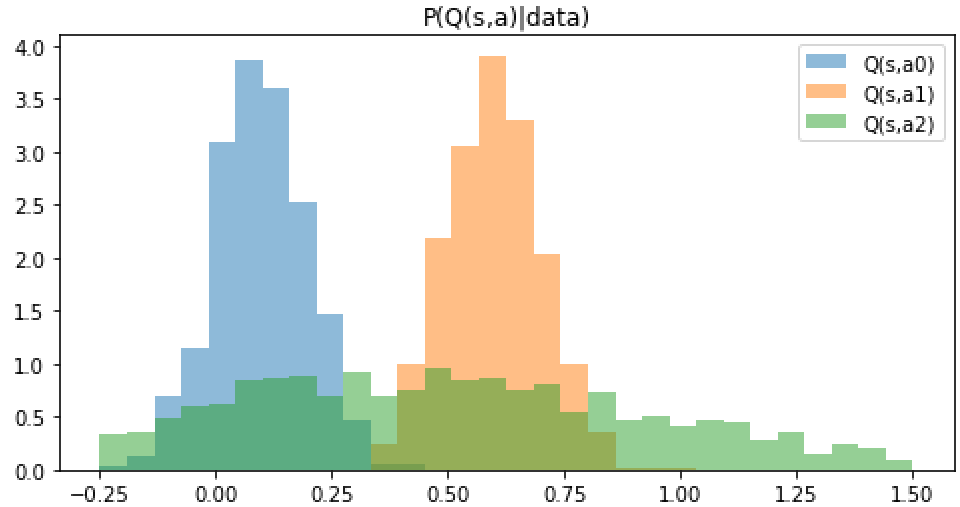
We want to try actions if we believe there's a chance they turn out optimal.

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?

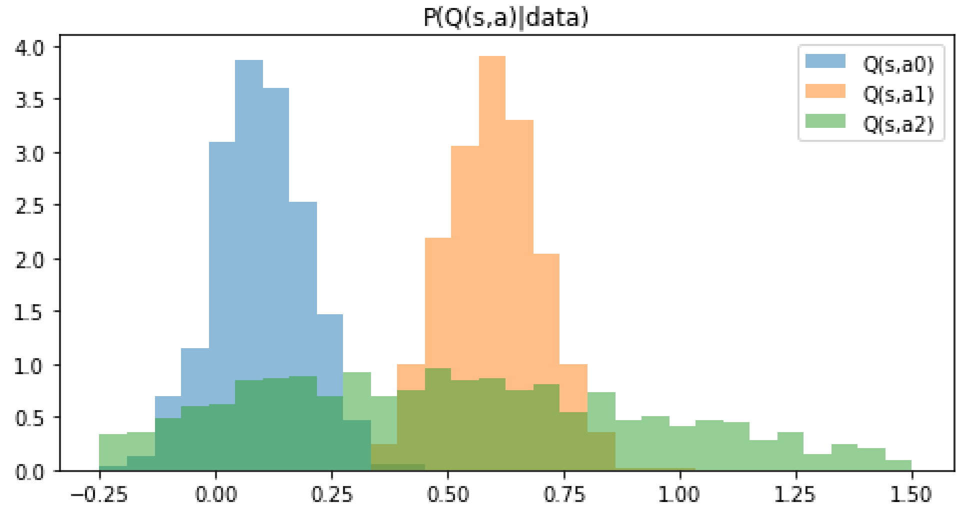


Q(s,a)

Thompson sampling

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

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



Q(s,a)

Optimism in face of uncertainty

Idea:

Prioritize actions with uncertain outcomes!

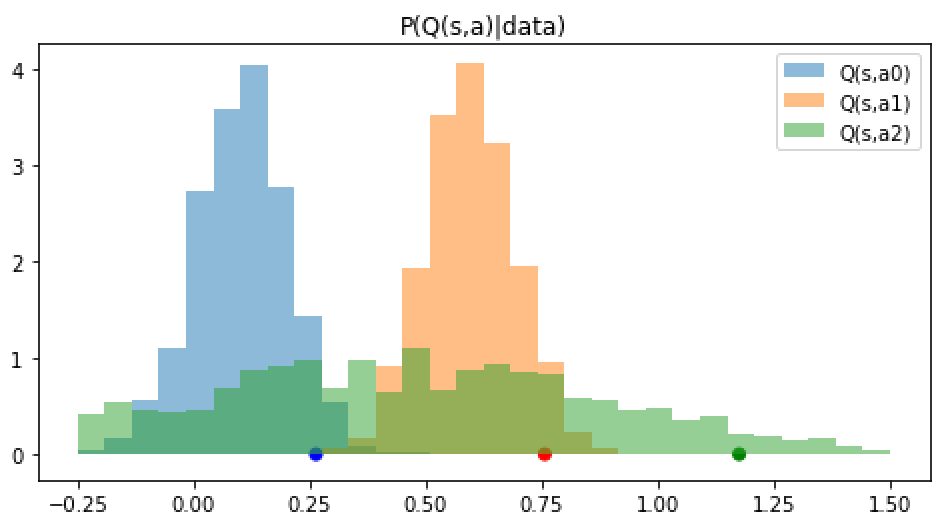
More uncertain = better.

Greater expected value = better

Math: try until upper confidence bound is small enough.

Optimism in face of uncertainty

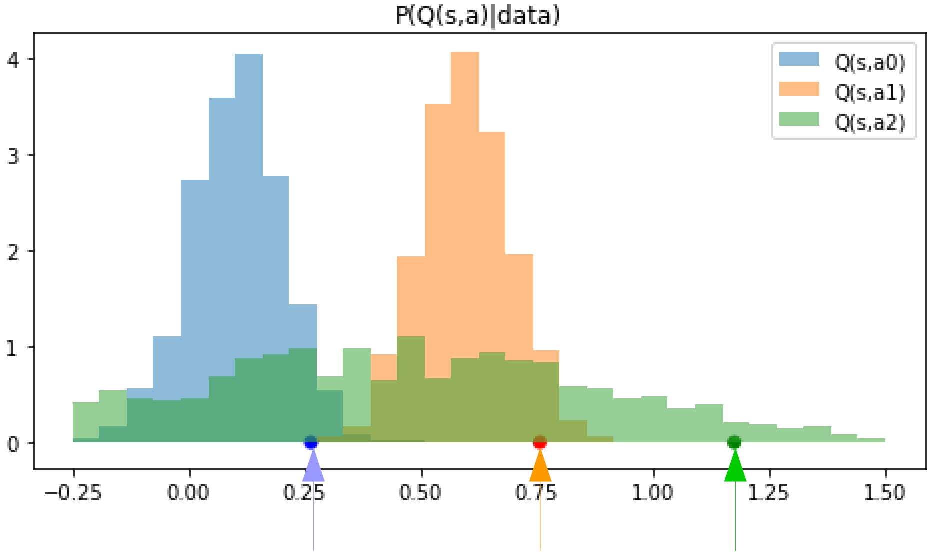
- Policy:
  - Compute 95% upper confidence bound *for each a*
  - Take action with highest confidence bound
  - What can we tune here to explore more/less?



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



points = 95% percentiles

Q(s,a)

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]

If not – divide
by r_max

- 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|data)
- Take q-th percentile

What models can learn that?

Bayesian UCB

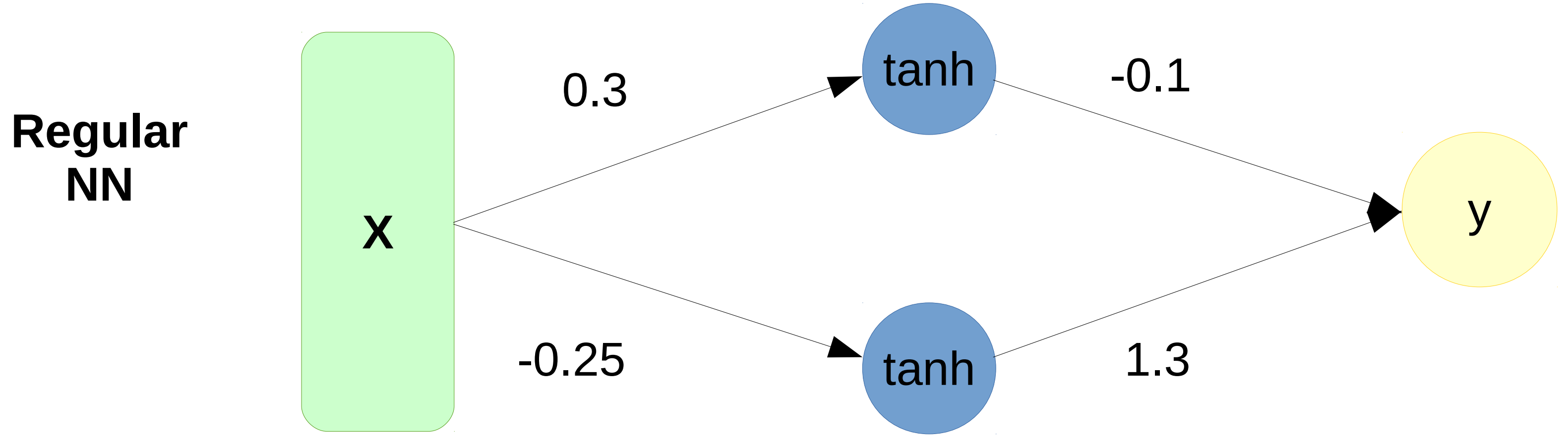
The usual way:

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

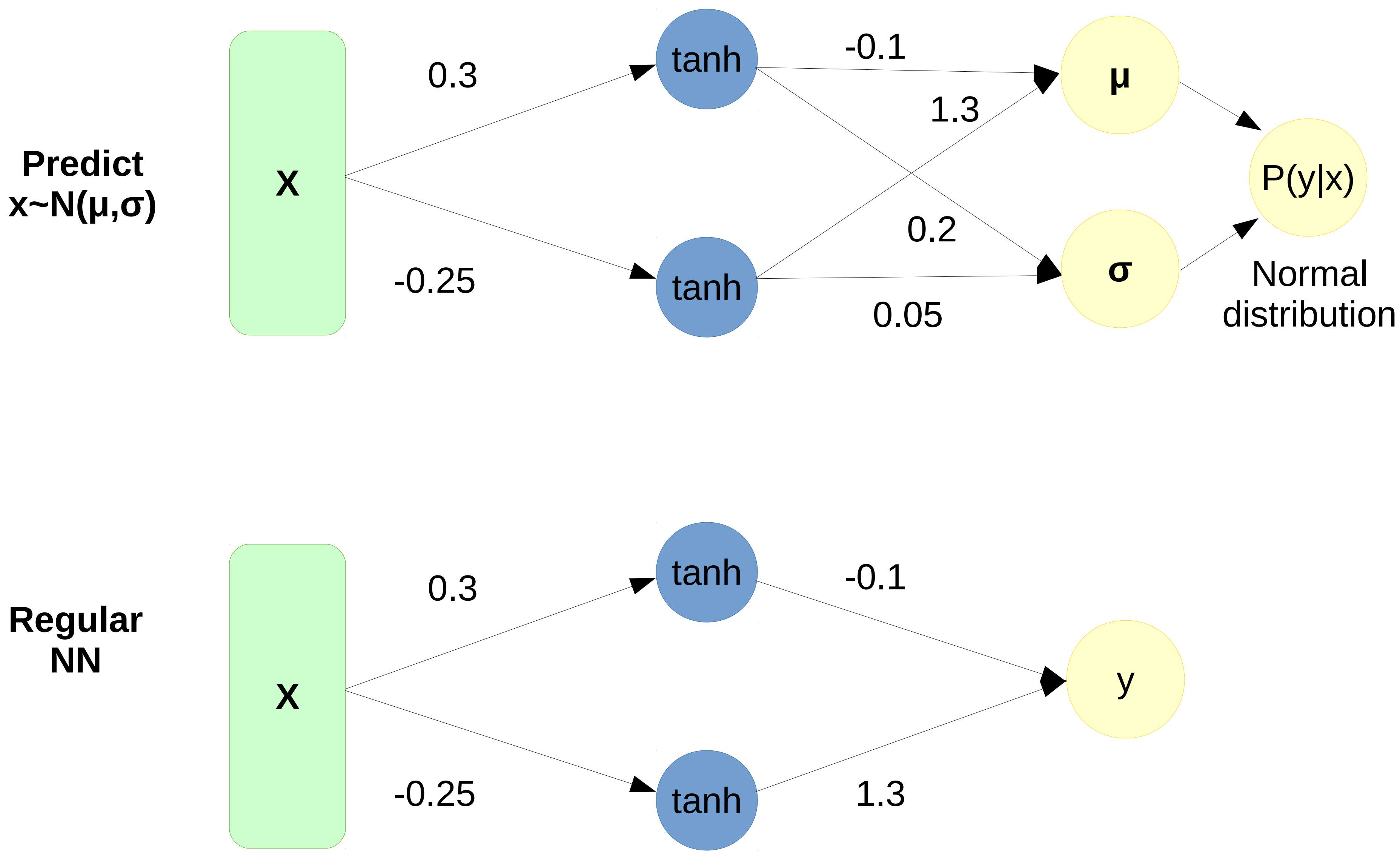
Approach 1: learn parametric P(Q), *e.g. normal*

Approach 2: use bayesian neural networks

Parametric



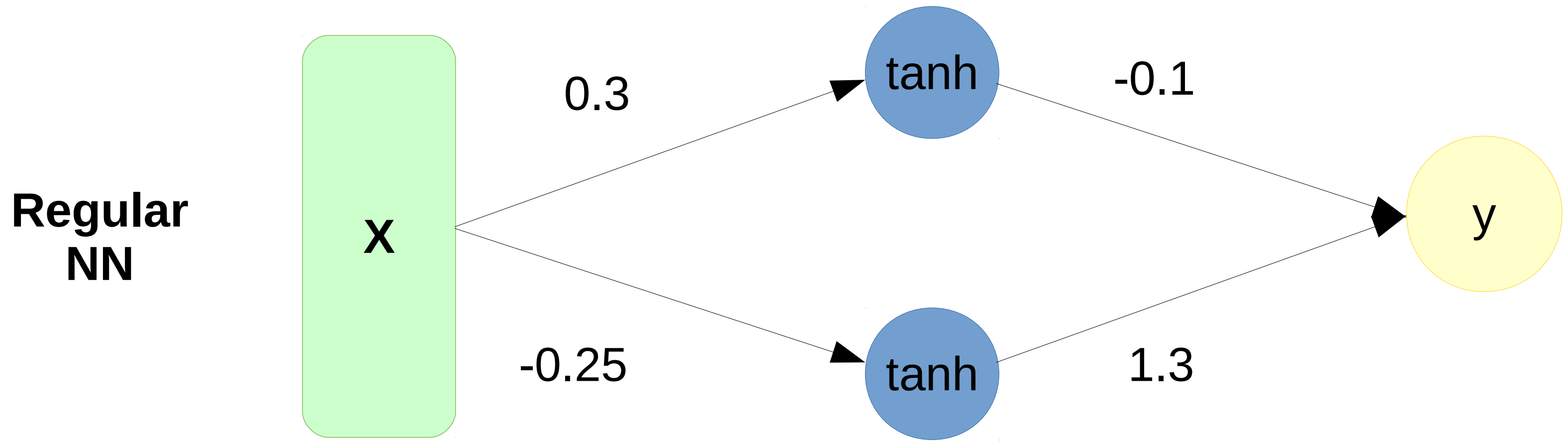
Parametric


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

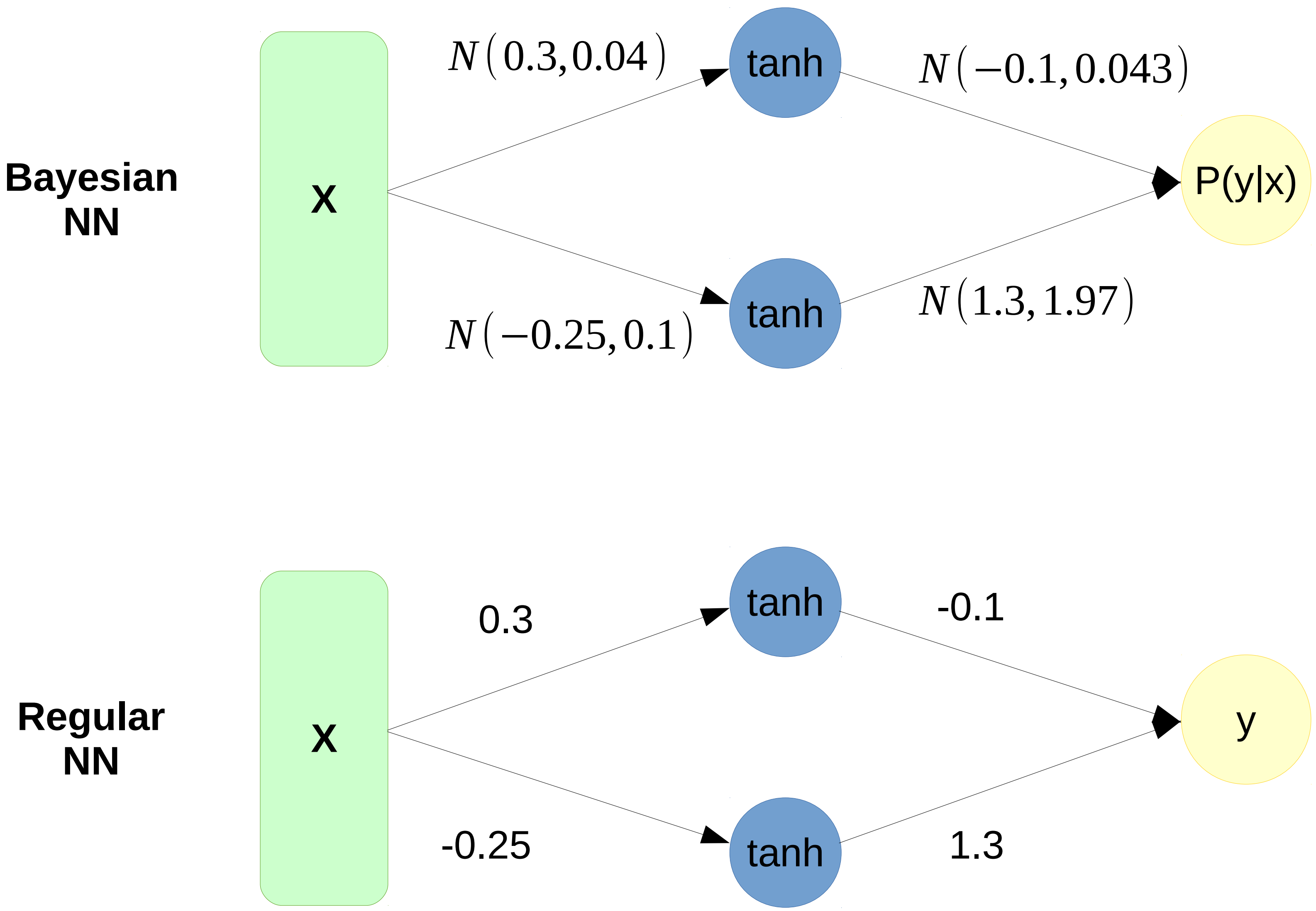
BNNs

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

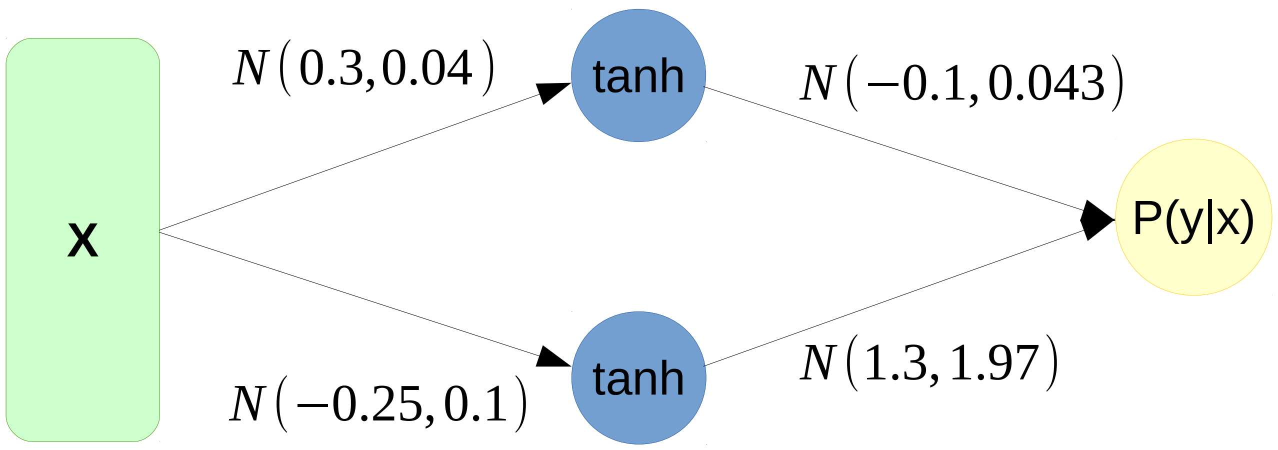
It does not cover all the philosophy and general cases.



BNNs


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

BNNs



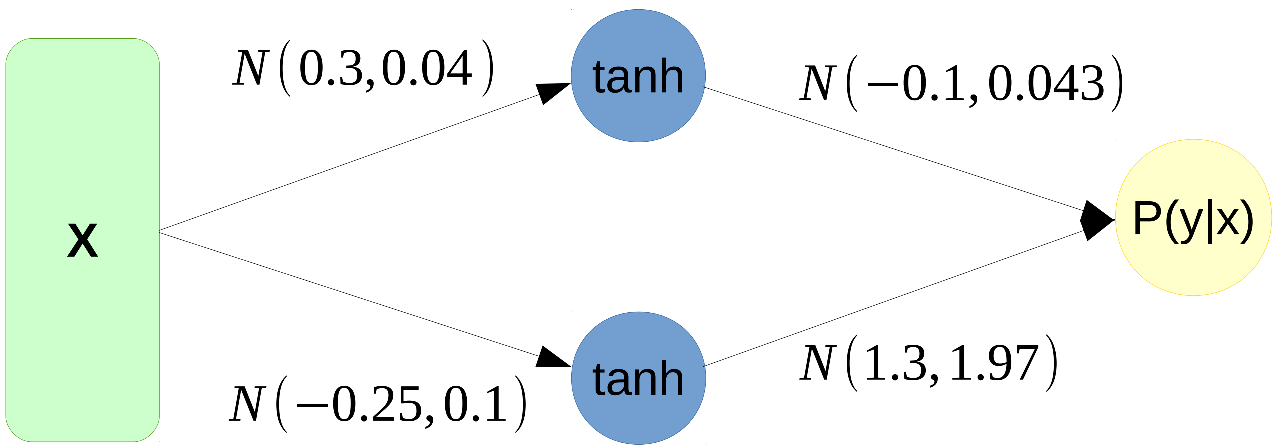
Idea:

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

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

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

BNNs



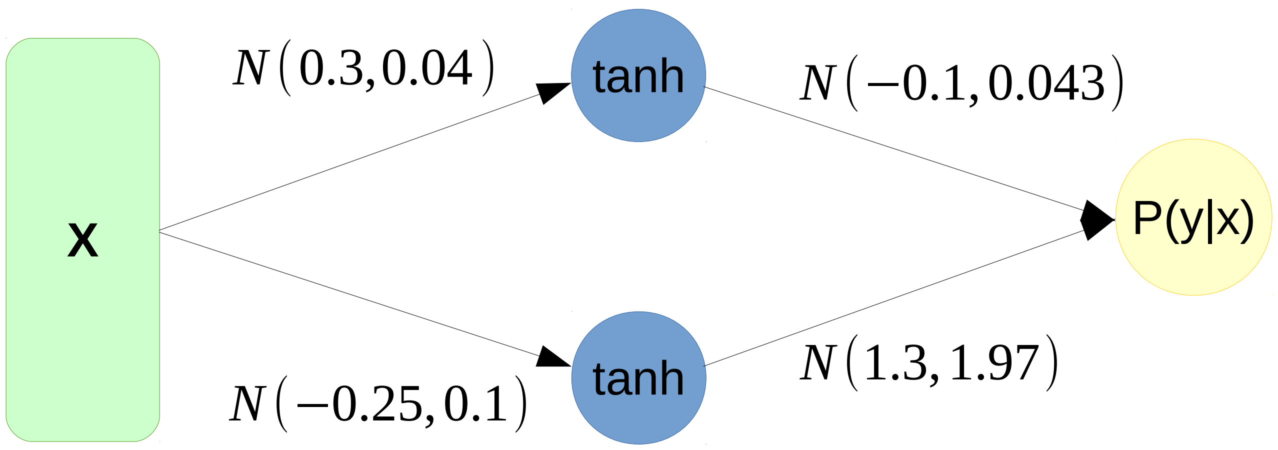
Idea:

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

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

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

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**

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

BNNs

Idea:

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

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

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

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

$$\mathring{\varphi}=argmax_{\varphi}E_{x_i,y_i\sim d}E_{\theta\sim q(\theta|\varphi)}P(y_i|x_i,\theta)$$

wanna explicit formulae? *d = dataset*

Lower bound

d = dataset

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

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

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

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

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

Lower bound

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

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

Can we perform gradient ascent directly?

Reparameterization trick

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

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

BNN likelihood

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

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

Better: local reparameterization trick (google it)

Reparameterization trick

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

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

In other words,
$\Sigma_{x,y\sim d} \log p(y|x,\mu+\sigma\psi)$

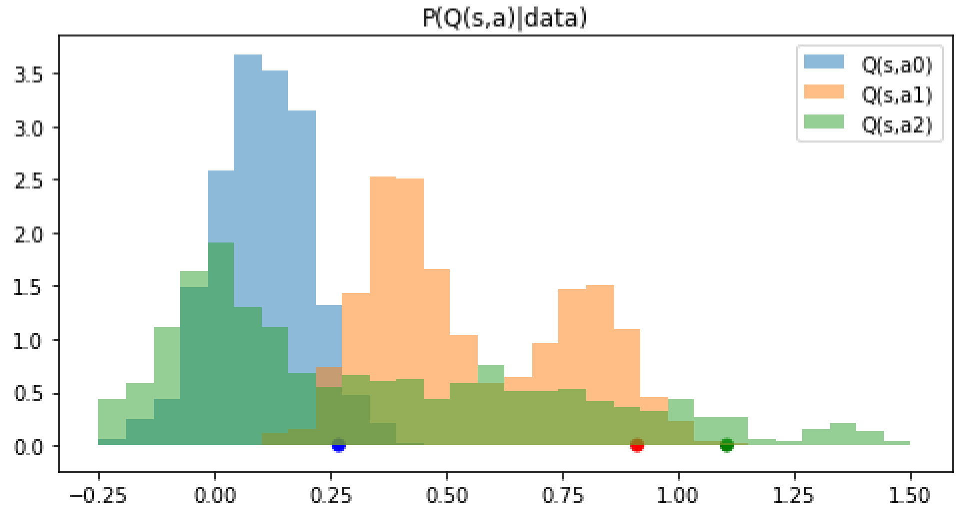
BNN likelihood

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

Better: local reparameterization trick (google it)

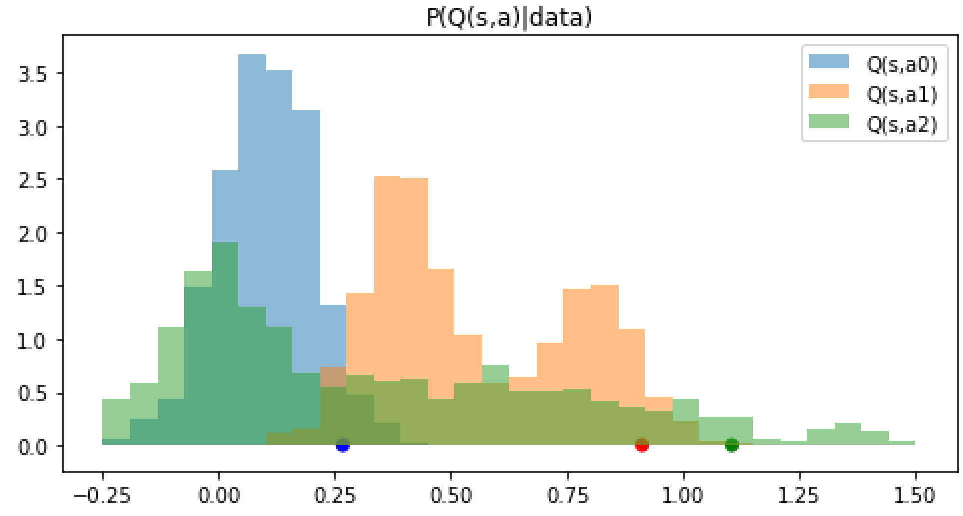
Using BNNs

- 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

<us talking>