CONTEXTUAL BANDIT

Reinforcement learning without feedback

Diego Klabjan
Professor, Industrial Engineering and Management Sciences

Northwestern | McCORMICK SCHOOL OF ENGINEERING

Outline

- Multi-arm bandit
  - Expected improvement algorithm
  - Upper confidence bound algorithm
  - Exp3
  - Exp4
- Contextual bandit
  - Upper confidence bound algorithm
  - Exp3
  - Exp4
- Use in RL

MULTI-ARM BANDIT

Examples

- Exploration-exploitation dilemma in RL
  - Explore to learn new states
  - Exploit to get a good policy on already seen states
- Restaurants
  - Go to those that you like
  - Go to a new one because you might like it
- Ad placement
  - Keep showing same ‘provable’ ads
  - Display a new ad
    - Hope the user clicks on it

Marketing Campaign

- Book advertising space on multiple web sites
- Pay-per-impression model
- Maximize investment
  - Number of subscribers, profit, etc
- Limited budget
  - Campaign runs for a limited time
- Not all sites have same performance
  - Revenue number observed only after we book on a site

Marketing Campaign

- Each web site is an action (arm)
- Reward = Revenue (number of subscribers) – pay for impression
  - Observed only after ad placed on a site
- Action (pulling an arm)
  - On which site to place our next ad
- Modern A/B testing done based on multi-arm bandit
  - Precisely its extension contextual bandit

Bandits

- Does not have anything to do with bandits
- One-arm bandit is one slot machine

$\mathcal{A} = \{\text{pull arm}\}$

$r(\text{pull arm}) = ?$

$\mathcal{A} = \{\text{pull}_1, \text{pull}_2, \dots, \text{pull}_n\}$

$r(a_n) = ?$

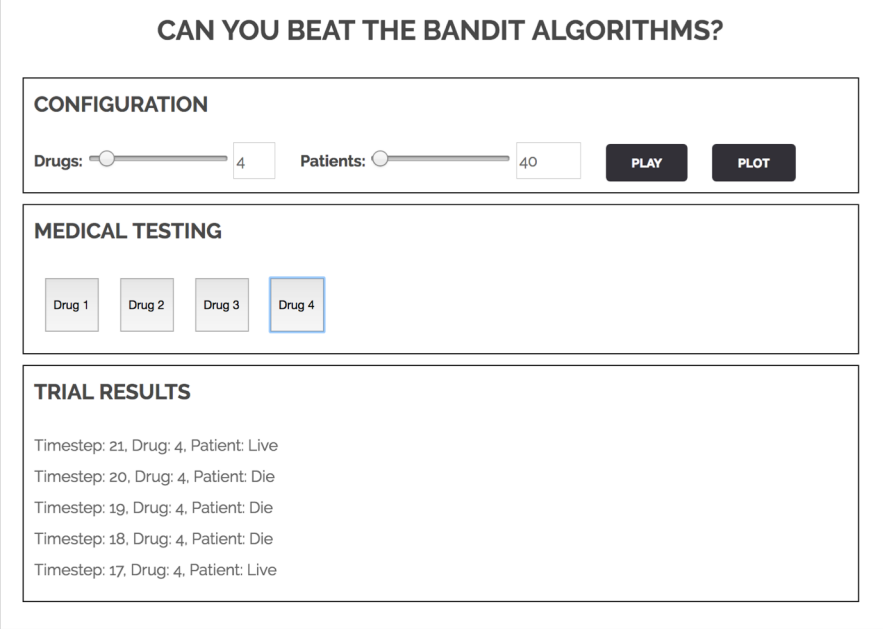
assume $r(a_n) \sim p(r|a_n)$

Unknown *per-action* reward distribution!

Challenge

- Obviously
  - Pick the arm with the highest expected reward
- Problem
  - Expected rewards are not given a priori
  - Have to learn them
- Reinforcement learning
  - One time step
  - No notion of state

Let's play!



- Drug prescription problem
- Bandit arm = drug (1 of 4)
- Reward
  - 1 if patient lives
  - 0 if patient dies
  - (stakes are high)

http://iosband.github.io/2015/07/28/Beat-the-bandit.html

Definition

- Assume $r(a_i) \sim p_{\theta_i}(r_i)$
  - $p(r_i = 1) = \theta_i$ and $p(r_i = 0) = 1 - \theta_i$
  - Goal to find θ_i
    - Optimal action then clear
- Prior $\theta_i \sim p(\theta)$
  - Otherwise unknown
- Formulate as RL
  - $s = [\theta_1, \ldots, \theta_n]$
- Caveat
  - Do not observe the state
  - Only estimates
  - Partially observed MDP
- Too much baggage
  - Much easier strategies work well

Regret

- Play game *T* times
- Optimal strategy to pull best arm a^* every single time
  - Have to learn the best arm
- Regret
  - $\mathrm{Reg}(T) = T \cdot E[r(a^*)] - \mathrm{E}[\sum_{t=1}^{T} r(a_t)]$

actual reward of action actually taken

Expected reward of best action

    - Minimize regret
  - Accumulated reward vs optimal reward

Regret

- Optimal exploration algorithm
  - Regret of order $\log T$
    - Can be shown cannot be done better
- Optimal algorithms do not work well in practice
- Different algorithms (non-optimal) work better in practice
- Multi-arm bandit
  - Action corresponds to each individual slot machine

ALGORITHMS

Exploitation Only: Expected Improvement

- Greedy approach
  - Keep track of accumulated observed average rewards
  - Denote as $\hat{\mu}_a$
    - Accumulated up to a certain point in time
  - Select action

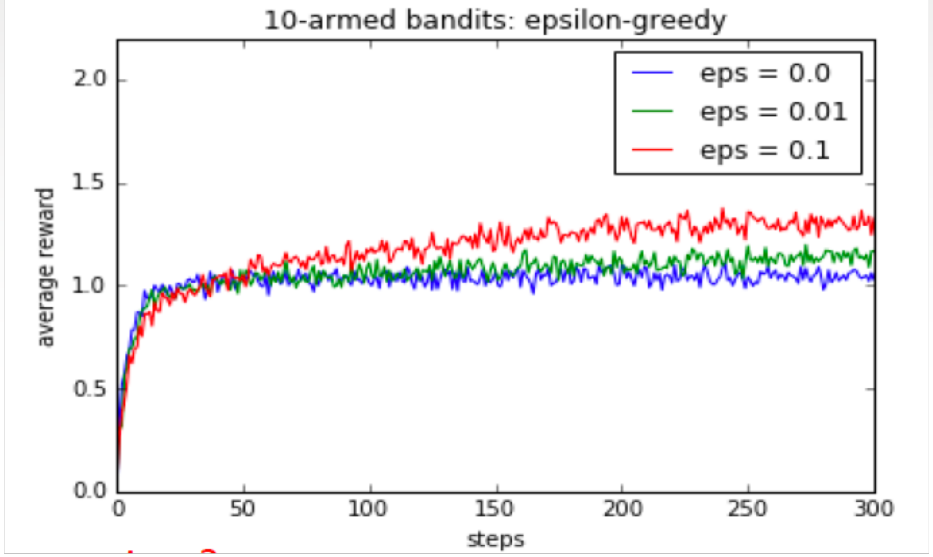
$$a^* = \text{argmax}\, \hat{\mu}_a$$

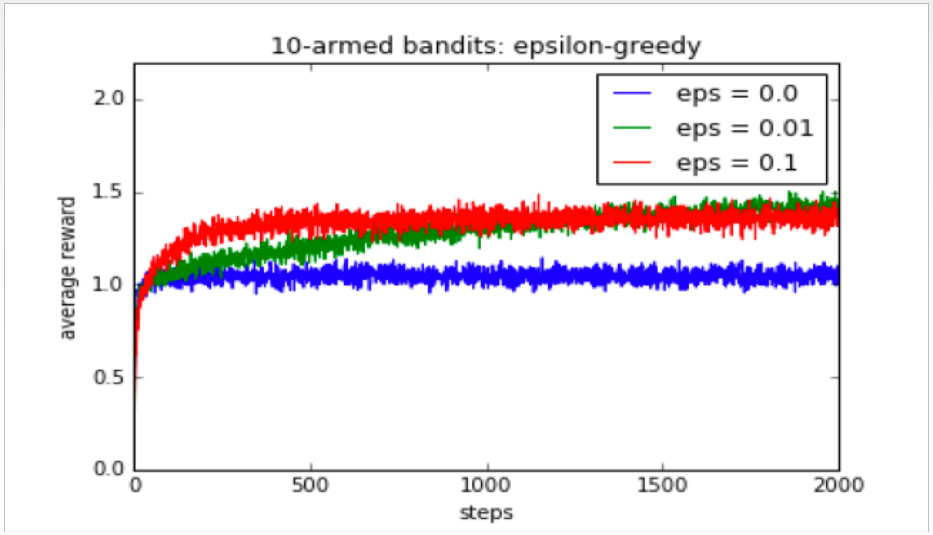
- No exploration
- Expected improvement algorithm
- Regret of order *T*

Epsilon-greedy Expected Improvement

- Algorithm
  - In a greedy manner find optimal action
  - With certain probability either pick this action or a random different action
- Same regret bound
- Works very well in practice
  - Very simple to implement and maintain

Performance





- Random action does not mean promising action
- Does not consider confidence intervals

Top-two Expected Improvement

- Idea
  - Use best arm with certain probability
  - Use second best arm otherwise
- Definition of second best
  - Second best with respect to empirical mean
    - Does not work well
  - Capture improvement from best arm
    - Disregard when worse than best arm
    - Account when better than best arm

- $a^* = \text{argmax}\, \hat{\mu}_a$
- Normal distribution fitting over all rewards of a given action
  - $r_t(a) \sim N(\mu_a, \sigma_a)$
- $v_{a,t} = E[(r_t(a) - r_t(a^*))^+]$
- Closed form expression exists
- Second best arm
  - $argmax_{a:a \neq a^*} v_{a,t}$

C. Qin, D. Russo, and D. Klabjan, Improving the Expected Improvement Algorithm. NIPS, Long Beach, CA 2017

Top-two Expected Improvement

	TTEI-1/2	EI
$[5, 4, 1, 1, 1]$	14.60	238.50
$[5, 4, 3, 2, 1]$	16.72	384.73
$[2, .8, .6, .4, .2]$	24.39	1525.42

Confidence 95% from optimal

	TTEI-1/2	aTTEI	TTEI-β^*	TTTS-β^*	RSO	TO	KG
$[5, 4, 1, 1, 1]$	61.97	61.98	**61.59**	62.86	97.04	77.76	75.55
$[5, 4, 3, 2, 1]$	66.56	**65.54**	65.55	66.53	103.43	88.02	81.49
$[2, .8, .6, .4, .2]$	76.21	72.94	**71.62**	73.02	101.97	96.90	86.98

Confidence 99% from optimal: TTTS = Thompson sampling, KG = knowledge gradient

Exploration-Exploitation

- Idea
  - Explore high reward actions
  - But also those that we have low confidence in
    - High variance

$$a^* = \text{argmax}\, \hat{\mu}_a + C\, \sigma_a$$

Variance estimate
Empirical variance

  - Try each arm until we have confidence
- Upper confidence bound algorithm

UCB

- Simple version

$$a^* = \text{argmax}\, \hat{\mu}_a + \sqrt{\frac{2 \log T}{N(a)}}$$

Number of times we observed a

- Achieves optimal regret

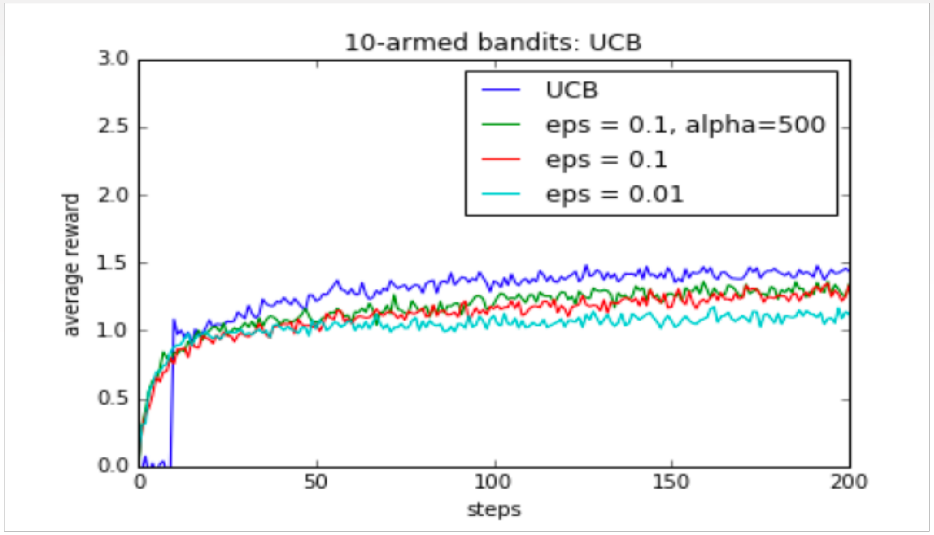
Derivation

- Hoeffding's Inequality
  - Let $X_1, \cdots, X_t$ be iid with $\bar{X}_t$ sample mean
  - $P[E[X] > \bar{X}_t + u] \leq e^{-2tu^2}$
- Bandit
  - Apply for action a with $X_i = r_i$
  - $\bar{X}_t = \mu_a$
  - t corresponds to $N(a)$
- Let p probability of true value exceeding UCB
  - $p = e^{-2tu^2}$
  - Solve for u

$$\sqrt{\frac{-\log p}{N(a)}}$$

- Want $p \to 0$ as $t \to \infty$
  - Pick $p = t^{-4}$

Performance



Decaying $\epsilon = \epsilon \cdot \alpha/(\alpha + t)$

Thompson Sampling

Assume $r(a_i) \sim p_{\theta_i}(r_i)$

POMDP with $s = [\theta_1, \ldots, \theta_K]$

Belief state is $\hat{p}(\theta_1, \ldots, \theta_K)$

This is a *model* of our bandit

- Hard to analyze theoretically
- Can work very well empirically

Sample $\theta_1, \ldots, \theta_K \sim \hat{p}(\theta_1, \ldots, \theta_K)$

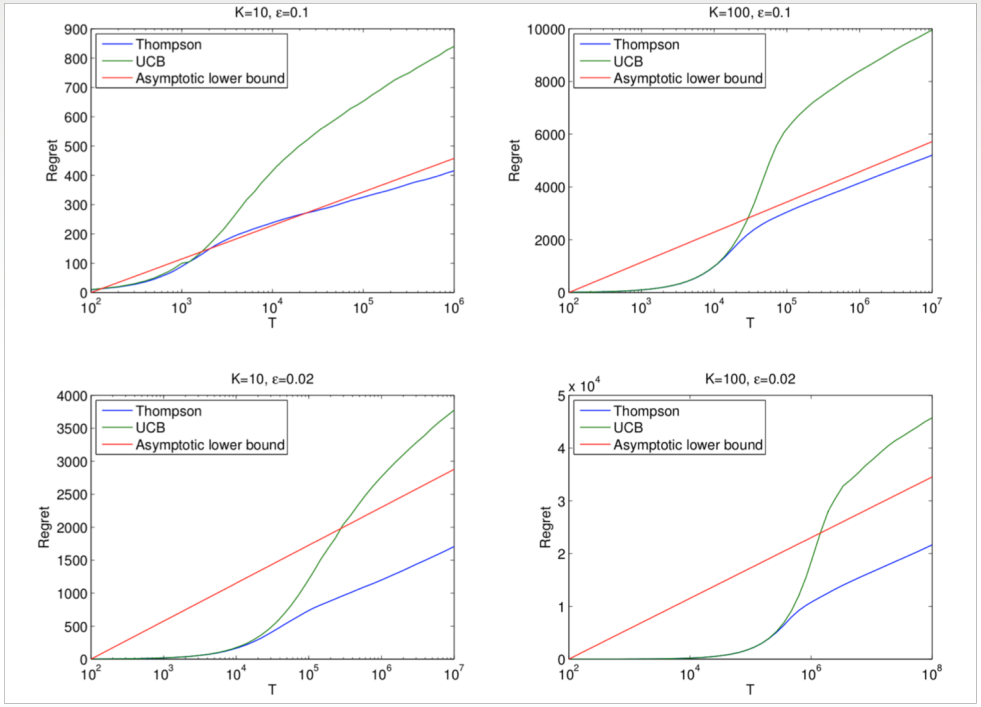
Pretend the model $\theta_1, \ldots, \theta_K$ is correct

Take the optimal action

Update the model

- Update the model step
  - Typically beta distribution used
  - Perform distribution fitting after observing the new sample

Thompson Sampling



Chapelle and Li, "An Empirical Evaluation of Thompson Sampling."

Exp3

- Exponential-weight algorithm for Exploration and Exploitation
- Idea
  - Each arm has a positive probability
    - It changes after each time
  - Draw an arm based on the probability
  - Observe reward
  - Turn to expected reward
    - Use probabilities
  - Update probability of the selected arm
    - Probability proportional to expected reward

Exp3

- Hard to deal with probabilities directly
  - Hard to impose that adjust probability is indeed probability
  - Keep track of weights
    - Turn them to probabilities
- Parameter γ, number of actions K
- Initial weights w_a^1 =1
  - Probabilities $\frac{w_a^1}{\sum_{\bar{a}} w_{\bar{a}}^1}$
  - Problem that some probabilities might become very small
    - For exploration we want them away from zero

Exp3

- For each time t
  - $p_a^t = (1-\gamma)\frac{w_a^t}{\sum_{\bar{a}} w_{\bar{a}}^t} + \frac{\gamma}{K}$
  - Draw action $\bar{a}$
  - Observe reward $r = r(\bar{a})$
  - Estimated reward $r = \bar{a}/p_{\bar{a}}^t$
  - Update weight
    - $w_{\bar{a}}^{t+1} = w_{\bar{a}}^t e^{-\eta r}$

- Interpretation
  - Have an expert that gives us advice on probabilities
  - Provide observed reward to expert
  - Expert updates probabilities
    - Beliefs

Regret

- $Reg(T) \leq \frac{logK}{\eta} + \eta TK$
- If $\eta = \sqrt{\frac{2\log K}{TK}}$
  - $Reg(T) \leq \sqrt{2TKlogK} = O(\sqrt{T})$

Exp4

- No reason to have a single expert
  - Ensemble often works better than a single model
- Each expert reveals distribution over arms
- Each expert has weight
  - Provides single distribution over arms
- Randomly select an arm
  - Compute reward as in Exp3
- Each expert computes its own expected reward of each arm in this round
  - Update expected reward across all rounds
- Update weights of each expert

Exp4

- For each round
  - Get expert advise $\pi_1, \cdots, \pi_M$
  - For each arm compute $p_a = \sum_m q_m \pi_{m,a}$
  - Draw arm $\bar{a}$ based on distribution p
  - Observe reward $r = r(\bar{a})$ and compute $r_{m,a} = \bar{a}/\pi_{m,\bar{a}}$
    - This defines reward vector $\bar{r}_m$
      - 0 for other arms
  - For each expert: $y_m = y_m + \bar{r}_m \cdot q_m$
  - $q_m = \frac{\exp(-\eta y_m)}{\sum_{\overline{m}} \exp(-\eta y_{\overline{m}})}$

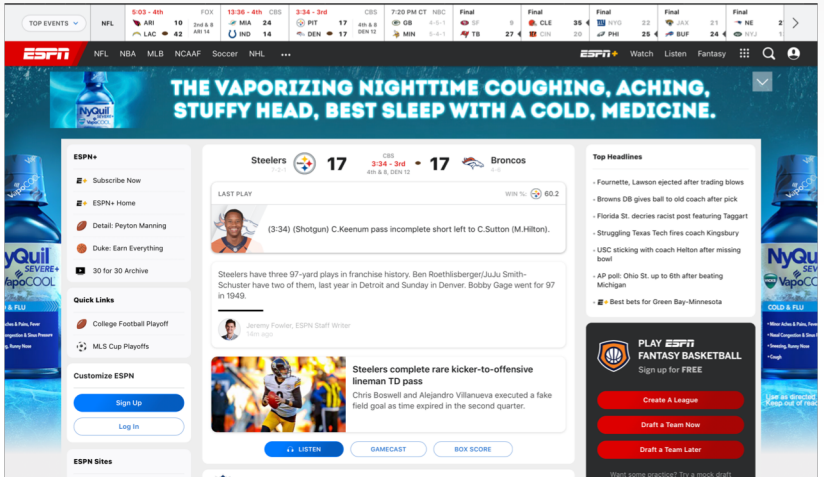
Exp4

- If $\eta = \sqrt{\frac{2\log(M)}{TK}}$
  - $\mathrm{Reg(T)} = \mathrm{O}(\sqrt{T})$
- Proof not that long
  - Uses elementary mathematics

CONTEXTUAL BANDIT

Ad Displaying

- E-commerce site with users
- Which ads to display to each user on each page
- Actions = ads at disposal
- Same as multi-arm bandit but
  - Users have attributes
  - Pages have attributes
  - Ad to display depends on user and page
- Context
- Reward depends on action and context



Problem Statement

- Actions as before
- Context vector s_t
- Process
  - Observe context s_t
  - Decide action $a_t = a_t(s_t)$
  - Receive reward $r_t = r_t(s_t, a_t)$
- Maximize total reward $E[\sum_{t=1}^{T} r_t]$

- No notion of state transitions
- Identical to one time period RL
  - Environment generates context and reward
  - Does not imply that RL algorithms are amenable

Expected Improvement

- Disregard context – does not work well
- Empirical mean estimation for each different context
  - $\mu_a(s_t)$
  - Works only if low number of different context vectors
- Not used in practice

Linear Regression

- $Q(s,a) = E(r|s,a)$
- Approximate Q
  - Given context s

$$\max_a Q(s,a)$$

- Approximating Q by linear function
  - $Q_\theta(s,a) = \phi(s,a)^T\theta$
- Easy to select best action if ϕ also linear
  - Largest coordinate of θ pertaining to a

Linear Regression

- At time t
  - Have predictors $\phi(s_i, a_i)$ for $i = 1,2,\cdots,t-1$
  - Have dependent variables $r_i = r_i(s_i, a_i)$ for $i = 1,2,\cdots,t-1$
  - Regression gives θ_t
- Regression gives mean estimates at context-action pair
- Can we generalize to UCB?

UCB

- Regression also gives
  - Estimate of the variance of context-action pair
- $\sigma_\theta^2(s,a)$
  - Uncertainty due to parameter estimation error
  - A^{-1} parameter covariance
  - $\sigma_\theta^2(s,a) = \phi(s,a)^T A^{-1} \phi(s,a)$
- Select action based on
  - $\max_a \phi(s_t,a)^T\theta \; + C \; \sigma_\theta^2(s_t,a)$ ← Quadratic problem Convex
  - C hyper-parameter

Ext3

- Consider all different occurrences of a context to be multi-arm bandit
  - S set of all different context vectors
- $R(\mathrm{T}) = \sum_{s \in S} \max_{a_t} \sum_{t:s_t=s} r(s_t, a_t)$
- Equivalent
  - Mapping $\phi: S \rightarrow A$ (set of all actions)
  - $R(T) = \max_{\phi} \sum_t r(s_t, \phi(s_t))$
- Regret defined in the same way
  - Reg(T)=E[R(T)- $\sum_t r(s_t, \bar{\phi}(s_t))$]

Exp3

- *Q(s,T)*= number of times context *s* appears in *T* rounds
- *R(s,n)*= regret of context *s* if appeared *n* times
  - Within the *T* rounds
- $Reg(T) = \sum_{s \in S} R(s, Q(s,T))$
- *Q(s,T)*
  - Can be uneven
  - Extreme cases
    - |S|/T
    - 1 (one context)

Exp3

- Only flexibility in Exp3
  - $\eta = \eta_s$
- Select
  - $\eta_s = \sqrt{\frac{2\log K}{Q(s,T)\cdot K}}$
- Caveat
  - What is *Q(s,T)?*
- Approximation for round *t*
  - $\eta_{t,s} = \sqrt{\frac{2\log K}{t(s)\cdot K}}$ // *t(s)* = number of times context *s* appeared before *t*

$$\eta = \sqrt{\frac{2\log K}{TK}}$$

Exp3

- Can be shown
  - $R(s,n) = 2\sqrt{nKlog(K)}$
- $Reg(T) \leq 2\sqrt{Klog(K)} \cdot \sum_{s \in S} \sqrt{Q(s,T)}$
- Best case one context
  - $Reg(T) \leq 2\sqrt{Klog(K)} \cdot \sqrt{T}$
- Worst case equally distributed
  - $Reg(T) \leq 2\sqrt{K \cdot log(K) \cdot |S| \cdot T}$

Insight of Worst Case

- $\mathrm{E}[\sum_t r(s_t, \bar{\phi}(s_t))] \geq R(T) - 2\sqrt{K \cdot log(K) \cdot |S| \cdot T}$
- In early rounds left-hand side negative
  - Total reward vacuous
- When |S| large
  - Better to consider several independent multi-arm bandit problems

Practical Consideration

- Flexibility in selecting ϕ
- Cluster context beforehand
  - Requires knowledge of all context up front
  - ϕ maps to the same action all context within the same cluster
- Employ an online clustering algorithm
- More general
  - Define similarity measure on context
  - If new sample has similarity within threshold from a previous sample
    - Consider the two to be mapped in the same set

Leverage Supervised Learning

- Create a sample of different context
  - Find optimal clustering/sampling
    - For each clustering run Exp3
  - Gives mapping
  - Train a multi-class classifier
- In true online setting
  - Use the classifier to map a newly observed context

Exp4

- Same treatment applies

$$\eta_{t,s} = \sqrt{\frac{2\log(M)}{t(s) \cdot K}}$$

$$\eta = \sqrt{\frac{2\log(M)}{TK}}$$

- Number of experts
  - $\log(M) = |S| \cdot \log(K)$
- Regret
  - $Reg(T) \leq \sqrt{2K \cdot log(K) \cdot |S| \cdot T}$
  - Same as Exp3 up to a constant

USE IN RL

Optimistic Exploration

- UCB: $a = \text{argmax}\, \mu_a + \underbrace{\sqrt{\frac{2 \ln T}{N(a)}}}_{\text{“exploration bonus”}}$

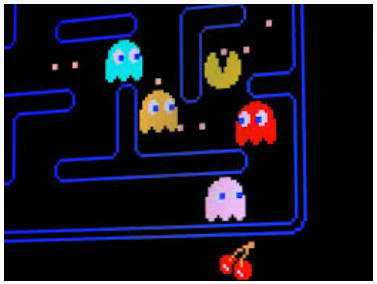
  Many functions work, so long as they decrease with $N(a)$

- How to incorporate in RL?
  - Count-based exploration
    - Use $N(s,a)$ or $N(s)$ to add exploration bonus
- Modify $r^+(s,a) = r(s,a) + \mathcal{B}\big(N(s)\big)$

  bonus that decreases with $N(s)$
- Use $r^+(s,a)$ instead of $r(s,a)$ with any model-free algorithm
  - Cons: simple addition to any RL algorithm
  - Pros: need to tune bonus weight

Counts in RL

- Use $r^{+}(s,a) = r(s,a) + \mathcal{B}\big(N(s)\big)$
- What is a count?



- We never see the same thing twice.
- Some states are more similar than others
  - Base count on similarity/partition of states

Fit Generative Model

- Fit density model $p_\theta(s)$ (or $p_\theta(s, a)$)
- $p_\theta(s)$ might be high even for a new s
  - Continuous distribution
  - If s is similar to previously seen states

$$P(s) = \frac{N(s)}{n}$$

count

probability/density

total states visited

After observing s

$$P^{new}(s) = \frac{N(s) + 1}{n + 1}$$

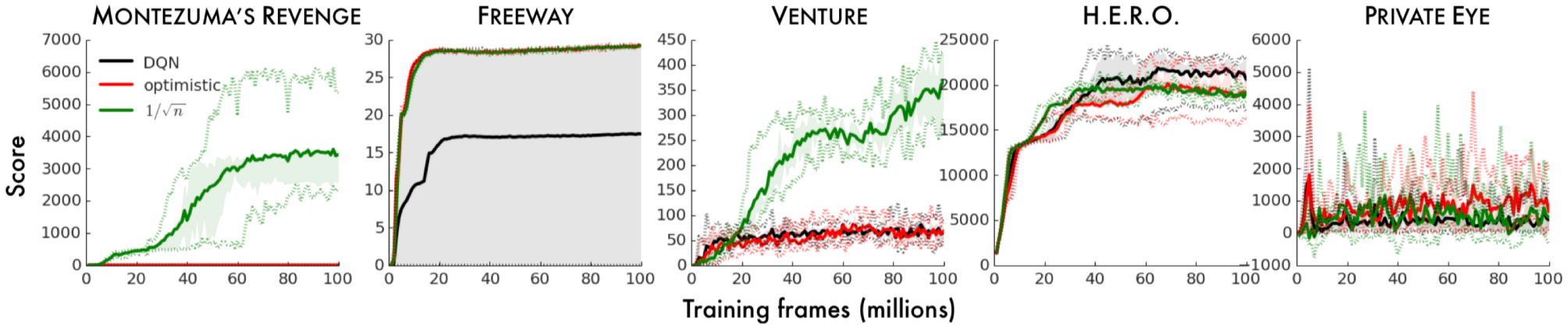
Two equations and two unknowns!

Pseudo-counts

- $p_\theta(s_i) = \frac{\hat{N}(s_i)}{\hat{n}}$ $\qquad p_{\theta'}(s_i) = \frac{\hat{N}(s_i)+1}{\hat{n}+1}$
- Solve
  - $\hat{N}(s_i) = \hat{n} p_\theta(s_i) \qquad \hat{n} = \frac{1-p_{\theta'}(s_i)}{p_{\theta'}(s_i)-p_\theta(s_i)} p_\theta(s_i)$

"pseudo-count"

fit model $p_\theta(s)$ to all states $\mathcal{D}$ seen so far
take step i and observe s_i
fit new model $p_{\theta'}(s)$ to $\mathcal{D} \cup s_i$
use $p_\theta(s_i)$ and $p_{\theta'}(s_i)$ to estimate $\hat{N}(s)$
set $r_i^+ = r_i + \mathcal{B}\left(\hat{N}(s)\right)$



Bellemare et al, Unifying Count-Based Exploration and Intrinsic Motivation

Pseudo-counts

- $p_\theta(s)$
  - Need to have distribution (output density)
  - No need to sample
- Compare new state to other states
  - Novel state if easy to distinguish from all previous states

- Fit binary classifier
  - Majority class all samples in *D*
  - Minority class the new sample

$$p_\theta(s) = \frac{1 - D_s(s)}{D_s(s)}$$

$D_s(s)$ probability of classifier that *s* is positive

Pseudo-counts

- Key observation
  - State *s* might have been observed before and thus several occurrences in *D* – as negative class
- $D_s(s) = \frac{1}{1+p(s)}$
- Heavily unbalanced classification problem
  - Regularize
  - Downsample
- One network across all states
  - Apply gradient steps on new state

Thompson Sampling

- Maintain distribution over parameters
  - Sample
  - Draw action
- What are parameters in RL?
  - Reward
    - Not informative enough
    - Q-factor good candidate
- Distribution over Q-factor functions
  - Not over Q-factor values

Sample $\theta_1, \ldots, \theta_K \sim \hat{p}(\theta_1, \ldots, \theta_K)$

Pretend the model $\theta_1, \ldots, \theta_K$ is correct

Take the optimal action

Update the model

Thompson Sampling

- Q-functions
  - Distribution over weights
    - Standard models expected value of weight
    - Add variance and assume normal distributed
  - Better to have an ensemble of Q-functions
- Ensemble in DQN
  - Sample several datasets from reply buffer
    - Train Q-function network on each dataset
  - Bootstrap

Sample Q-function Q from $p(Q)$

Act according to Q for one episode

Update $p(Q)$

Since off-policy do not care which Q used to collect data

Thompson Sampling

- From replay buffer *D* create samples $D_1, \cdots, D_K$ with replacement
- Train Q-network f_{θ_i} based on data D_i
- Select a random network f_{θ_i} and use it to sample an episode
  - Add episode to *D*
- In practice
  - One dataset
  - Random seeds in training