Lecture 3: Multi-armed bandits

Hrvoje Stojic

April 29, 2020

The roadmap

The roadmap

The roadmap

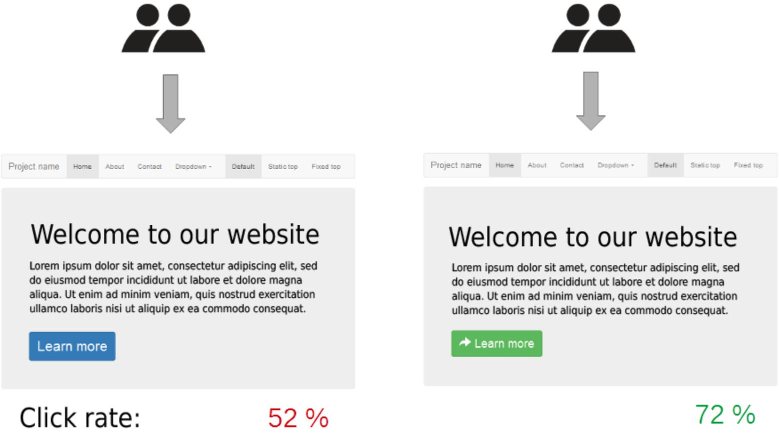
- Multi-armed bandits (MAB) - 2 lectures
  - Naive classics: ϵ-greedy, Softmax
  - Optimism in the face of uncertainty with Upper confidence bound (UCB)
  - Bayesian bandits with Thompson sampling
  - Overview of extensions
  - Application: A/B testing
  - Problem set 1

The roadmap

- Multi-armed bandits (MAB) - 2 lectures
  - Naive classics: ϵ-greedy, Softmax
  - Optimism in the face of uncertainty with Upper confidence bound (UCB)
  - Bayesian bandits with Thompson sampling
  - Overview of extensions
  - Application: A/B testing
  - Problem set 1

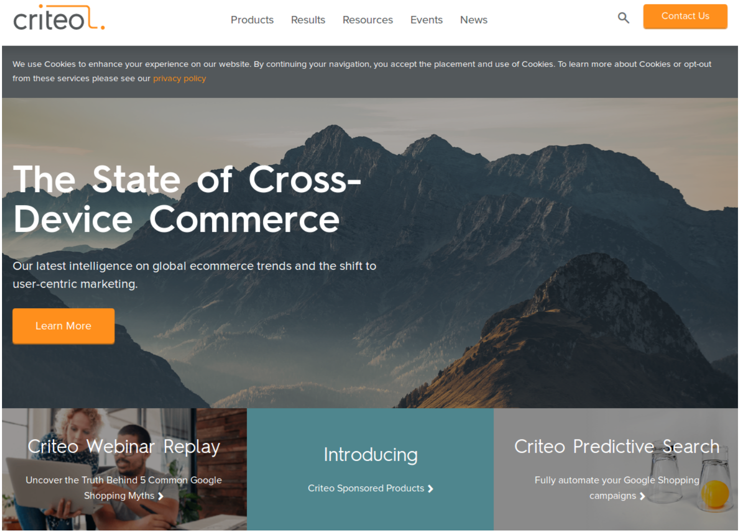
- Contextual bandits (CMAB) - 2 lectures
  - Linear models with UCB
  - Introduction to Gaussian processes (GP)
  - Bayesian optimization with GP-UCB
  - Overview of extensions
  - Application: Optimizing hyperparameters
  - Problem set 2

A/B testing



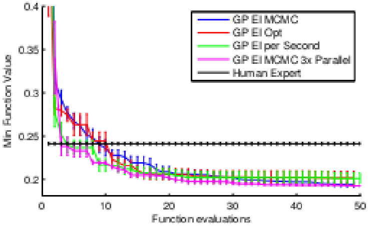
Source: Wikipedia

Recommender systems and ad placement



Source: Criteo webpage

AutoML and hyperparameter tuning

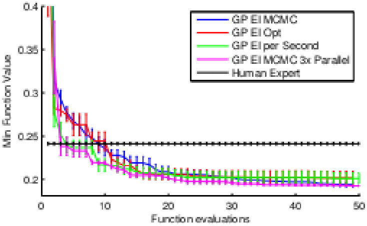


	convex	MRBI
TPE	**14.13** ±0.30 %	**44.55** ±0.44%
GP	16.70 ± 0.32%	47.08 ± 0.44%
Manual	18.63 ± 0.34%	47.39 ± 0.44%
Random	18.97 ± 0.34 %	50.52 ± 0.44%

Table 2: The test set classification error of the best model found by each search algorithm on each problem. Each search algorithm was allowed up to 200 trials. The manual searches used 82 trials for **convex** and 27 trials **MRBI**.

Source: Snoek et al 2012; Bergstra et al 2011

AutoML and hyperparameter tuning



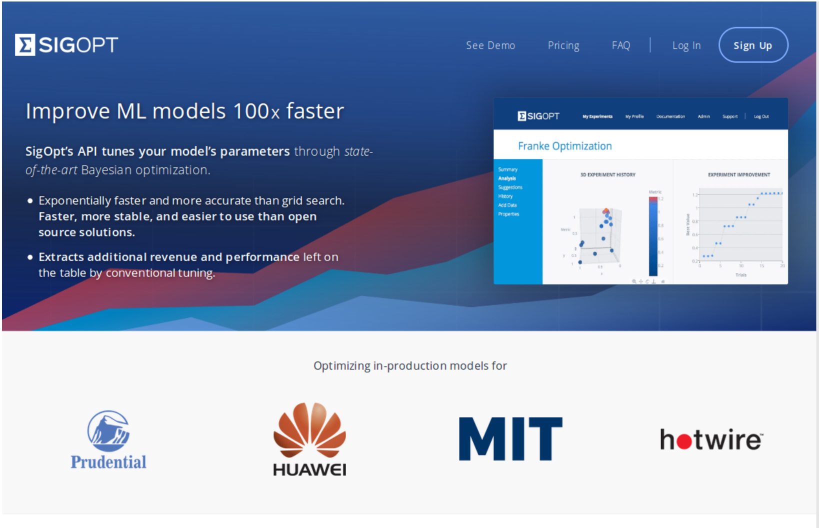
	convex	MRBI
TPE	**14.13** ± 0.30 %	**44.55** $\pm 0.44\%$
GP	$16.70 \pm 0.32\%$	$47.08 \pm 0.44\%$
Manual	$18.63 \pm 0.34\%$	$47.39 \pm 0.44\%$
Random	18.97 ± 0.34 %	$50.52 \pm 0.44\%$

Table 2: The test set classification error of the best model found by each search algorithm on each problem. Each search algorithm was allowed up to 200 trials. The manual searches used 82 trials for **convex** and 27 trials **MRBI**.

- CIFAR 10: state of the art was test error of 18%, they achieved 14.98%
- MNIST rotated background images

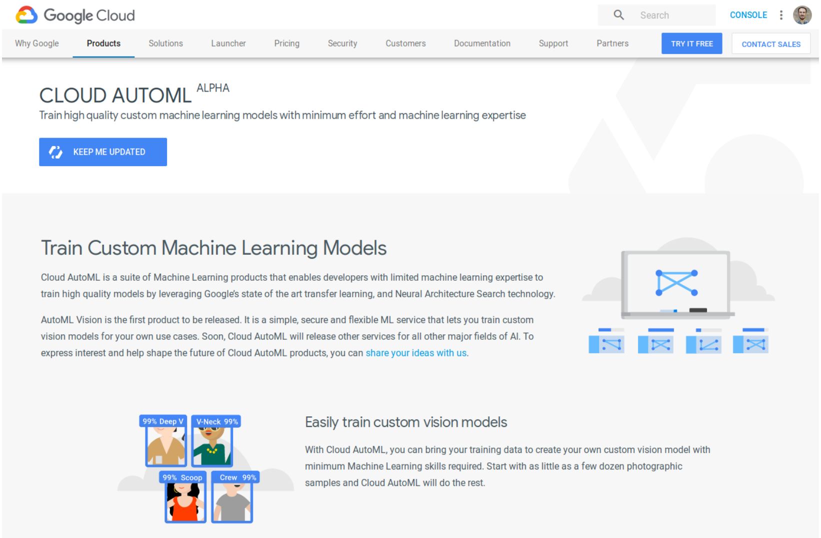
Source: Snoek et al 2012; Bergstra et al 2011

Bayesian optimization going mainstream



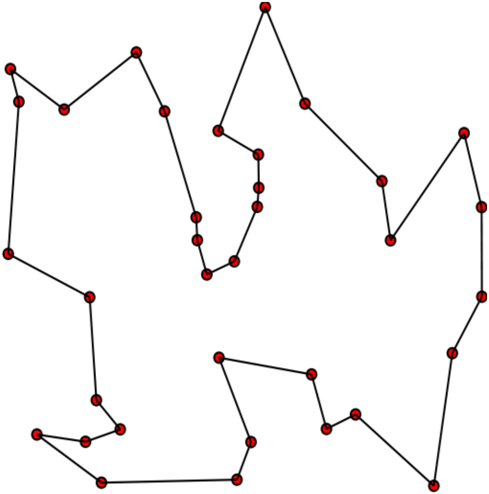
Source: SigOpt webpage

Google Cloud AutoML for computer vision



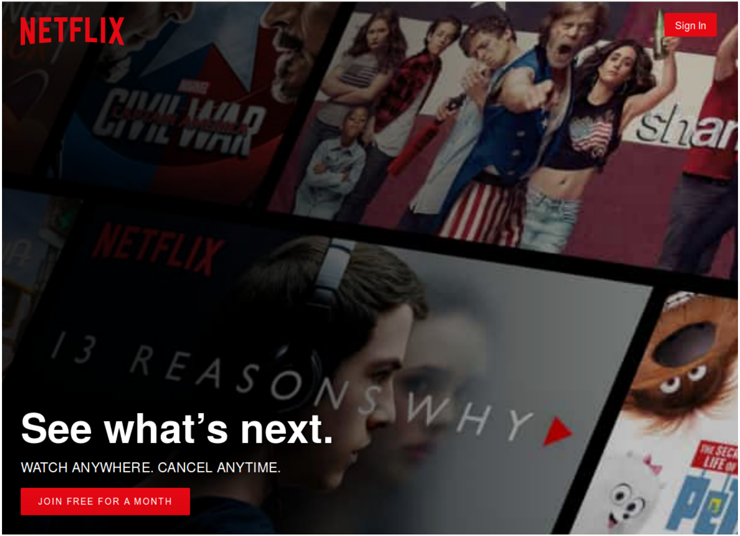
Source: Google AutoML webpage

Optimizing parameters of combinatorial optimization software



Source: Wikipedia

Preference learning and interactive user interfaces



Source: Netflix webpage

References

- Bandits

  - Sutton, R., & Barto, A. (2018). Introduction to Reinforcement Learning (book free of charge: www.incompleteideas.net/sutton/book/the-book.html)
  - Lattimore, T., & Szepesvári, C. (2020). Bandit algorithms. (book free of charge: banditalgs.com/)
  - Szepesvári, C. (2010). Algorithms for Reinforcement Learning.
  - D. Silver's lectures (videos and slides: www0.cs.ucl.ac.uk/staff/D.Silver/web/Teaching.html)

- Gaussian Processes

  - Rasmussen, C. E., & Williams, C. K. I. (2006). Gaussian processes for machine learning. MIT Press. (book free of charge: www.gaussianprocess.org/gpml/)
  - Carl Rasmussen's lectures (videos)
  - Nando De Freitas' lectures (videos and slides: www.youtube.com/user/ProfNandoDF/videos)

References

- Contextual bandits, Bayesian optimization

  - Auer, P., Cesa-Bianchi, N., & Fischer, P. (2002). Finite-time analysis of the multiarmed bandit problem. Machine learning, 47(2-3), 235-256
  - Li, L., Chu, W., Langford, J., & Schapire, R. E. (2010). A contextual-bandit approach to personalized news article recommendation. In Proceedings of the 19th international conference on World wide web (pp. 661-670).
  - Snoek, J., Larochelle, H., & Adams, R. P. (2012). Practical Bayesian Optimization of Machine Learning Algorithms. Advances in Neural Information Processing Systems, 2951-2959.
  - Shahriari, B., Swersky, K., Wang, Z., Adams, R. P., & de Freitas, N. (2016). Taking the Human Out of the Loop: A Review of Bayesian Optimization. Proceedings of the IEEE, 104(1), 148–175.

Software

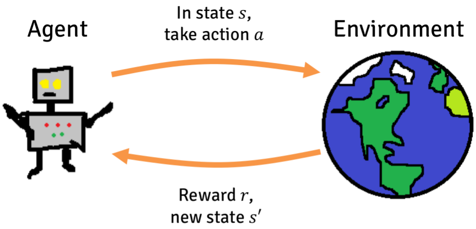
- Python libraries
  - RLlib
  - Tensorflow agents
  - scikit-learn, auto-sklearn
  - Hyperopt (Bergstra et al., 2011)
  - Spearmint (Snoek et al., 2014)
- R packages
  - GPfit, gptk, FastGP
  - rBayesianOptimization
  - DiceOptim (Roustant et al., 2012)
- Matlab
  - GPML (Rasmussen)
- C++
  - BayesOpt (Martinez-Cantin, 2014)
- Java
  - SMAC (Hutter et al., 2011)
  - AutoWEKA

Practicalities

- Contact:
  - hrvoje.stojic_youknowwhat_protonmail.com
  - Office hours by video calls
- Materials:
  - Videos and slides will be uploaded to Box
  - In addition, I will have the whole course posted on Github, where you will have access to the source code
- Evaluation:
  - No exam
  - Individual problem sets 50% and group projects 50%
  - We are still deciding on the exact form of the group projects, we will let you know the details soon!

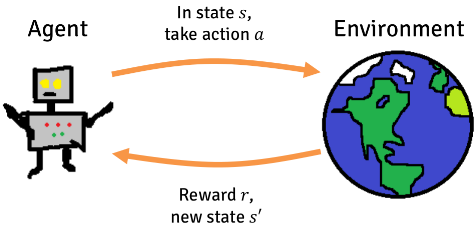
Multi-armed bandits

A subclass of reinforcement learning problems



Learning to
- maximize reward
- in a reactive environment
- under partial feedback

A subclass of reinforcement learning problems



Learning to
- maximize reward
- in a reactive environment
- under partial feedback

Two key challenges of RL:

1. Dealing with long-term effects of actions
2. Dealing with uncertainty due to partial feedback

Partial feedback -> Exploration exploitation problem

Partial feedback -> Exploration exploitation problem

- Acting involves a fundamental trade-off:
  - **Exploitation**: Make the best decision given current information
  - **Exploration**: Gather more information

Partial feedback -> Exploration exploitation problem

- Acting involves a fundamental trade-off:
  - **Exploitation**: Make the best decision given current information
  - **Exploration**: Gather more information

- The best long-term strategy may involve short-term sacrifices

Partial feedback -> Exploration exploitation problem

- Acting involves a fundamental trade-off:
  - **Exploitation**: Make the best decision given current information
  - **Exploration**: Gather more information

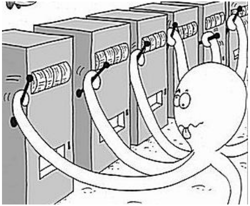
- The best long-term strategy may involve short-term sacrifices

- Main idea is to gather just enough information to make the best decisions, that is, accumulate as much rewards as possible

Partial feedback -> Exploration exploitation problem

- Acting involves a fundamental trade-off:
  - **Exploitation**: Make the best decision given current information
  - **Exploration**: Gather more information
- The best long-term strategy may involve short-term sacrifices
- Main idea is to gather just enough information to make the best decisions, that is, accumulate as much rewards as possible
- Examples:
  - Going to a favourite restaurant (**exploitation**), or try a new restaurant (**exploration**)
  - Show the most successful ad (**exploitation**), or show a new ad (**exploration**)

Formulation



- A tuple $\langle \mathcal{A}, \mathcal{R} \rangle$
- $\mathcal{A}$ is a (stationary) set of K actions/arms
- $\mathcal{R}^a(r) = P[r|a]$ is an unknown but stationary probability distribution over rewards
- At each step t the agent selects an action $a_t \in \mathcal{A}$
- The environment generates a reward $r_t \sim \mathcal{R}^{a_t}$
- The goal is to maximise cumulative reward $\sum_{t=1}^{\tau} r_t$

Regret: measure of performance

Regret: measure of performance

- The **action value** is the expected reward for action a, $Q(a) = E[r|a]$

Regret: measure of performance

- The **action value** is the expected reward for action a, $Q(a) = E[r|a]$
- The **optimal value** is $V^* = Q(a^*) = \max_{a \in \mathcal{A}} Q(a)$

Regret: measure of performance

- The **action value** is the expected reward for action a, $Q(a) = E[r|a]$

- The **optimal value** is $V^* = Q(a^*) = \max_{a \in \mathcal{A}} Q(a)$

- The **cumulative regret** is the total opportunity loss $L_t = E[\sum_{\tau=1}^{t} V^* - Q(a_\tau)]$

Regret: measure of performance

- The **action value** is the expected reward for action a, $Q(a) = E[r|a]$
- The **optimal value** is $V^* = Q(a^*) = \max_{a \in \mathcal{A}} Q(a)$
- The **cumulative regret** is the total opportunity loss $L_t = E[\sum_{\tau=1}^{t} V^* - Q(a_\tau)]$
- Regret can be expressed in terms of counts and gaps:
  - The count $N_t(a)$ is number of selections for action a
  - The gap Δ_a is the difference in value between action a and optimal action a^*, $\Delta_a = V^* - Q(a)$
  - The cumulative regret, stated differently:

$$L_t = \sum_{a \in \mathcal{A}} E[N_t(a)](V^* - Q(a)) = \sum_{a \in \mathcal{A}} E[N_t(a)]\Delta_a$$

Agents

1. Learning

Agents

1. Learning
   - Action values are initially unknown

Agents

1. Learning
   - Action values are initially unknown
   - Estimating the expected action value: $\hat{Q}_t(a)$

Agents

1. Learning
   - Action values are initially unknown
   - Estimating the expected action value: $\hat{Q}_t(a)$
   - In our stationary bandit problem, after taking an action and observing a reward, we could simply update our estimates by

$$\hat{Q}_t(a) = \frac{1}{N_t(a)} \sum_{t'=1}^{t} r_{t'} \mathbf{1}(a_{t'} = a)$$

or

$$\hat{Q}_t(a) = \hat{Q}_{t-1}(a) + \frac{1}{N_t(a)} (r_t - \hat{Q}_{t-1}(a))$$

Agents

1. Learning
   - Action values are initially unknown
   - Estimating the expected action value: $\hat{Q}_t(a)$
   - In our stationary bandit problem, after taking an action and observing a reward, we could simply update our estimates by

     $$\hat{Q}_t(a) = \frac{1}{N_t(a)} \sum_{t'=1}^{t} r_{t'} \mathbf{1}(a_{t'} = a)$$

     or

     $$\hat{Q}_t(a) = \hat{Q}_{t-1}(a) + \frac{1}{N_t(a)} (r_t - \hat{Q}_{t-1}(a))$$

   - Note that $\hat{Q}_t(a)$ have to be initialized in some way

Agents

1. Learning
   - Action values are initially unknown
   - Estimating the expected action value: $\hat{Q}_t(a)$
   - In our stationary bandit problem, after taking an action and observing a reward, we could simply update our estimates by

     $$\hat{Q}_t(a) = \frac{1}{N_t(a)} \sum_{t'=1}^{t} r_{t'} \mathbf{1}(a_{t'} = a)$$

     or

     $$\hat{Q}_t(a) = \hat{Q}_{t-1}(a) + \frac{1}{N_t(a)}(r_t - \hat{Q}_{t-1}(a))$$

   - Note that $\hat{Q}_t(a)$ have to be initialized in some way

2. Choice

Agents

1. Learning
   - Action values are initially unknown
   - Estimating the expected action value: $\hat{Q}_t(a)$
   - In our stationary bandit problem, after taking an action and observing a reward, we could simply update our estimates by

     $$\hat{Q}_t(a) = \frac{1}{N_t(a)} \sum_{t'=1}^{t} r_{t'} \mathbf{1}(a_{t'} = a)$$

     or

     $$\hat{Q}_t(a) = \hat{Q}_{t-1}(a) + \frac{1}{N_t(a)} (r_t - \hat{Q}_{t-1}(a))$$

   - Note that $\hat{Q}_t(a)$ have to be initialized in some way

2. Choice
   - Actions with greatest estimated value are called **greedy** actions

Agents

1. Learning
   - Action values are initially unknown
   - Estimating the expected action value: $\hat{Q}_t(a)$
   - In our stationary bandit problem, after taking an action and observing a reward, we could simply update our estimates by

$$\hat{Q}_t(a) = \frac{1}{N_t(a)} \sum_{t'=1}^{t} r_{t'} \mathbf{1}(a_{t'} = a)$$

   or

$$\hat{Q}_t(a) = \hat{Q}_{t-1}(a) + \frac{1}{N_t(a)}(r_t - \hat{Q}_{t-1}(a))$$

   - Note that $\hat{Q}_t(a)$ have to be initialized in some way

2. Choice
   - Actions with greatest estimated value are called **greedy** actions
   - If you select one of greedy actions we say you are **exploiting** your knowledge, otherwise you are **exploring**

Agents

1. Learning
   - Action values are initially unknown
   - Estimating the expected action value: $\hat{Q}_t(a)$
   - In our stationary bandit problem, after taking an action and observing a reward, we could simply update our estimates by

     $$\hat{Q}_t(a) = \frac{1}{N_t(a)} \sum_{t'=1}^{t} r_{t'} \mathbf{1}(a_{t'} = a)$$

     or

     $$\hat{Q}_t(a) = \hat{Q}_{t-1}(a) + \frac{1}{N_t(a)}(r_t - \hat{Q}_{t-1}(a))$$

   - Note that $\hat{Q}_t(a)$ have to be initialized in some way

2. Choice
   - Actions with greatest estimated value are called **greedy** actions
   - If you select one of greedy actions we say you are **exploiting** your knowledge, otherwise you are **exploring**
   - A policy, $\pi(a)$, given estimates of action values, $\hat{Q}_t(a)$, gudies actions

Can we do better with random exploration?

- Main idea is to add some noise to a greedy policy

Can we do better with random exploration?

- Main idea is to add some noise to a greedy policy

- Popular examples

Can we do better with random exploration?

- Main idea is to add some noise to a greedy policy
- Popular examples
  - ϵ-**greedy**: With probability $1-\epsilon$ select $a_t^* = \mathrm{argmax}_{a \in \mathcal{A}} \hat{Q}_t(a)$, with probability ϵ select a random action

Can we do better with random exploration?

- Main idea is to add some noise to a greedy policy
- Popular examples
  - ϵ-**greedy**: With probability $1-\epsilon$ select $a_t^* = \text{argmax}_{a\in\mathcal{A}}\hat{Q}_t(a)$, with probability ϵ select a random action
  - **Softmax**: $P(a_t = a) = \frac{\exp(\hat{Q}_t(a)/\tau)}{\sum_{a'=1}^{K}\exp(\hat{Q}_t(a')/\tau)}$

Can we do better with random exploration?

- Main idea is to add some noise to a greedy policy
- Popular examples
  - ϵ-**greedy**: With probability $1-\epsilon$ select $a_t^* = \text{argmax}_{a\in\mathcal{A}}\hat{Q}_t(a)$, with probability ϵ select a random action
  - **Softmax**: $P(a_t = a) = \frac{\exp(\hat{Q}_t(a)/\tau)}{\sum_{a'=1}^{K}\exp(\hat{Q}_t(a')/\tau)}$
- How would these policies perform?

Can we do better with random exploration?

- Main idea is to add some noise to a greedy policy
- Popular examples
  - ϵ-**greedy**: With probability $1-\epsilon$ select $a_t^* = \text{argmax}_{a \in \mathcal{A}} \hat{Q}_t(a)$, with probability ϵ select a random action
  - **Softmax**: $P(a_t = a) = \frac{\exp(\hat{Q}_t(a)/\tau)}{\sum_{a'=1}^{K} \exp(\hat{Q}_t(a')/\tau)}$
- How would these policies perform?
  - In the limit all options would be visited infinitely many times

Can we do better with random exploration?

- Main idea is to add some noise to a greedy policy
- Popular examples
  - ϵ-**greedy**: With probability $1-\epsilon$ select $a_t^* = \text{argmax}_{a \in \mathcal{A}} \hat{Q}_t(a)$, with probability ϵ select a random action
  - **Softmax**: $P(a_t = a) = \frac{\exp(\hat{Q}_t(a)/\tau)}{\sum_{a'=1}^{K} \exp(\hat{Q}_t(a')/\tau)}$
- How would these policies perform?
  - In the limit all options would be visited infinitely many times
  - By the law of large numbers, $\hat{Q}_t(a) \approx Q(a)$

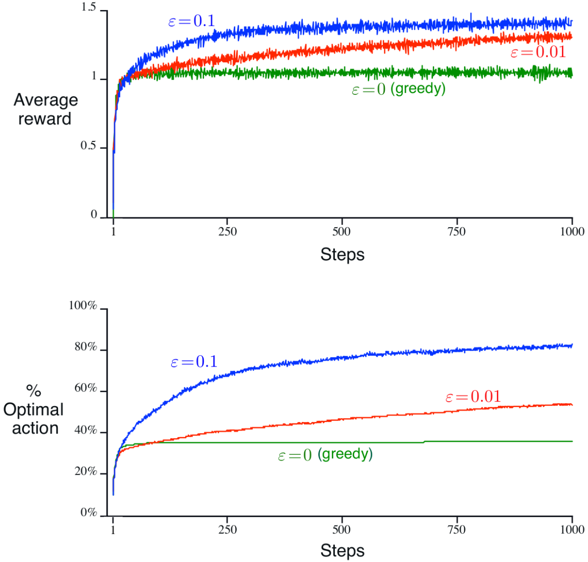
Can we do better with random exploration?

- Main idea is to add some noise to a greedy policy
- Popular examples
  - ϵ-**greedy**: With probability $1-\epsilon$ select $a_t^* = \mathrm{argmax}_{a \in \mathcal{A}} \hat{Q}_t(a)$, with probability ϵ select a random action
  - **Softmax**: $P(a_t = a) = \frac{\exp(\hat{Q}_t(a)/\tau)}{\sum_{a'=1}^{K} \exp(\hat{Q}_t(a')/\tau)}$
- How would these policies perform?
  - In the limit all options would be visited infinitely many times
  - By the law of large numbers, $\hat{Q}_t(a) \approx Q(a)$
  - I.e agents would identify the optimal action value, V^*

Rewards and % optimal actions

Source: Sutton & Barto 2018

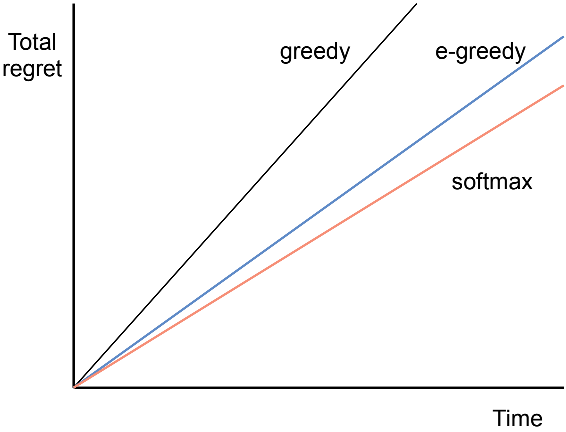
Rewards and % optimal actions



Source: Sutton & Barto 2018

Linear regret of random exploration

Linear regret of random exploration



Sublinear regret

Sublinear regret

- Is it possible with random exploration?

Sublinear regret

- Is it possible with random exploration?
- Decaying ϵ_t-greedy (Auer, Cesa-Bianchi, Fischer, 2002)

Sublinear regret

- Is it possible with random exploration?
- Decaying ϵ_t-greedy (Auer, Cesa-Bianchi, Fischer, 2002)
  - Pick a decay schedule for ϵ_1, ϵ_2, ... e.g. $\epsilon_t = \min\{1, \frac{c|\mathcal{A}|}{\min_a \Delta_a t}\}$

Sublinear regret

- Is it possible with random exploration?
- Decaying ϵ_t-greedy (Auer, Cesa-Bianchi, Fischer, 2002)
  - Pick a decay schedule for $\epsilon_1, \epsilon_2, \ldots$ e.g. $\epsilon_t = \min\{1, \frac{c|\mathcal{A}|}{\min_a \Delta_a t}\}$
  - Has logarithmic asymptotic total regret, but assumes knowing the gap

Sublinear regret

- Is it possible with random exploration?
- Decaying ϵ_t-greedy (Auer, Cesa-Bianchi, Fischer, 2002)
  - Pick a decay schedule for ϵ_1, ϵ_2, ... e.g. $\epsilon_t = \min\{1, \frac{c|\mathcal{A}|}{\min_a \Delta_a t}\}$
  - Has logarithmic asymptotic total regret, but assumes knowing the gap
- There are similar formulations for decaying temperature in Softmax

Sublinear regret

- Is it possible with random exploration?
- Decaying ϵ_t-greedy (Auer, Cesa-Bianchi, Fischer, 2002)
  - Pick a decay schedule for ϵ_1, ϵ_2, ... e.g. $\epsilon_t = \min\{1, \frac{c|\mathcal{A}|}{\min_a \Delta_a t}\}$
  - Has logarithmic asymptotic total regret, but assumes knowing the gap
- There are similar formulations for decaying temperature in Softmax
- These algorithms can achieve very good performance

Sublinear regret

- Is it possible with random exploration?
- Decaying ϵ_t-greedy (Auer, Cesa-Bianchi, Fischer, 2002)
  - Pick a decay schedule for ϵ_1, ϵ_2, ... e.g. $\epsilon_t = \min\{1, \frac{c|\mathcal{A}|}{\min_a \Delta_a t}\}$
  - Has logarithmic asymptotic total regret, but assumes knowing the gap
- There are similar formulations for decaying temperature in Softmax
- These algorithms can achieve very good performance
- But these are heuristic approaches and it is usually difficult to tune the decay

Brief diversion into gradient bandit algorithms

- Using action value estimates to guide actions is not the only game in town

Brief diversion into gradient bandit algorithms

- Using action value estimates to guide actions is not the only game in town
- We can formulate **preference** for an action, $H_t(a)$, which has no direct relation to expected reward

Brief diversion into gradient bandit algorithms

- Using action value estimates to guide actions is not the only game in town
- We can formulate **preference** for an action, $H_t(a)$, which has no direct relation to expected reward
- Actions are selected according to a softmax distribution with respect to relative differences in preferences

$$\pi_t(a) = P(a_t = a) = \frac{\exp(H_t(a))}{\sum_{a'=1}^{K} \exp(H_t(a'))}$$

Brief diversion into gradient bandit algorithms

- Using action value estimates to guide actions is not the only game in town
- We can formulate **preference** for an action, $H_t(a)$, which has no direct relation to expected reward

- Actions are selected according to a softmax distribution with respect to relative differences in preferences

$$\pi_t(a) = P(a_t = a) = \frac{\exp(H_t(a))}{\sum_{a'=1}^{K} \exp(H_t(a'))}$$

- Preferences are updated as follows

$$H_{t+1}(a_t) = H_t(a_t) + \alpha(r_t - \bar{r}_t)(1 - \pi_t(a_t))$$

and

$$H_{t+1}(a) = H_t(a) - \alpha(r_t - \bar{r}_t)\pi_t(a) \; \forall a \neq a_t$$

Brief diversion into gradient bandit algorithms

- Using action value estimates to guide actions is not the only game in town
- We can formulate **preference** for an action, $H_t(a)$, which has no direct relation to expected reward
- Actions are selected according to a softmax distribution with respect to relative differences in preferences

$$\pi_t(a) = P(a_t = a) = \frac{\exp(H_t(a))}{\sum_{a'=1}^{K} \exp(H_t(a'))}$$

- Preferences are updated as follows

$$H_{t+1}(a_t) = H_t(a_t) + \alpha(r_t - \bar{r}_t)(1 - \pi_t(a_t))$$

and

$$H_{t+1}(a) = H_t(a) - \alpha(r_t - \bar{r}_t)\pi_t(a) \; \forall a \neq a_t$$

  - where $\alpha > 0$ and $\bar{r}_t \in R$ is average over all the rewards

Brief diversion into gradient bandit algorithms

- Using action value estimates to guide actions is not the only game in town
- We can formulate **preference** for an action, $H_t(a)$, which has no direct relation to expected reward

- Actions are selected according to a softmax distribution with respect to relative differences in preferences

$$\pi_t(a) = P(a_t = a) = \frac{\exp(H_t(a))}{\sum_{a'=1}^{K} \exp(H_t(a'))}$$

- Preferences are updated as follows

$$H_{t+1}(a_t) = H_t(a_t) + \alpha(r_t - \bar{r}_t)(1 - \pi_t(a_t))$$

and

$$H_{t+1}(a) = H_t(a) - \alpha(r_t - \bar{r}_t)\pi_t(a) \; \forall a \neq a_t$$

  - where $\alpha > 0$ and $\bar{r}_t \in R$ is average over all the rewards
  - and initial preferences set to $H_1(a) = 0$

Brief diversion into gradient bandit algorithms

- Using action value estimates to guide actions is not the only game in town
- We can formulate **preference** for an action, $H_t(a)$, which has no direct relation to expected reward
- Actions are selected according to a softmax distribution with respect to relative differences in preferences

$$\pi_t(a) = P(a_t = a) = \frac{\exp(H_t(a))}{\sum_{a'=1}^{K} \exp(H_t(a'))}$$

- Preferences are updated as follows

$$H_{t+1}(a_t) = H_t(a_t) + \alpha(r_t - \bar{r}_t)(1 - \pi_t(a_t))$$

and

$$H_{t+1}(a) = H_t(a) - \alpha(r_t - \bar{r}_t)\pi_t(a) \; \forall a \neq a_t$$

  - where $\alpha > 0$ and $\bar{r}_t \in R$ is average over all the rewards
  - and initial preferences set to $H_1(a) = 0$

Brief diversion into gradient bandit algorithms

- Using action value estimates to guide actions is not the only game in town
- We can formulate **preference** for an action, $H_t(a)$, which has no direct relation to expected reward
- Actions are selected according to a softmax distribution with respect to relative differences in preferences

$$\pi_t(a) = P(a_t = a) = \frac{\exp(H_t(a))}{\sum_{a'=1}^{K} \exp(H_t(a'))}$$

- Preferences are updated as follows

$$H_{t+1}(a_t) = H_t(a_t) + \alpha(r_t - \bar{r}_t)(1 - \pi_t(a_t))$$

and

$$H_{t+1}(a) = H_t(a) - \alpha(r_t - \bar{r}_t)\pi_t(a) \; \forall a \neq a_t$$

  - where $\alpha > 0$ and $\bar{r}_t \in R$ is average over all the rewards
  - and initial preferences set to $H_1(a) = 0$

Brief diversion into gradient bandit algorithms

- Using action value estimates to guide actions is not the only game in town
- We can formulate **preference** for an action, $H_t(a)$, which has no direct relation to expected reward

- Actions are selected according to a softmax distribution with respect to relative differences in preferences

$$\pi_t(a) = P(a_t = a) = \frac{\exp(H_t(a))}{\sum_{a'=1}^{K} \exp(H_t(a'))}$$

- Preferences are updated as follows

$$H_{t+1}(a_t) = H_t(a_t) + \alpha(r_t - \bar{r}_t)(1 - \pi_t(a_t))$$

and

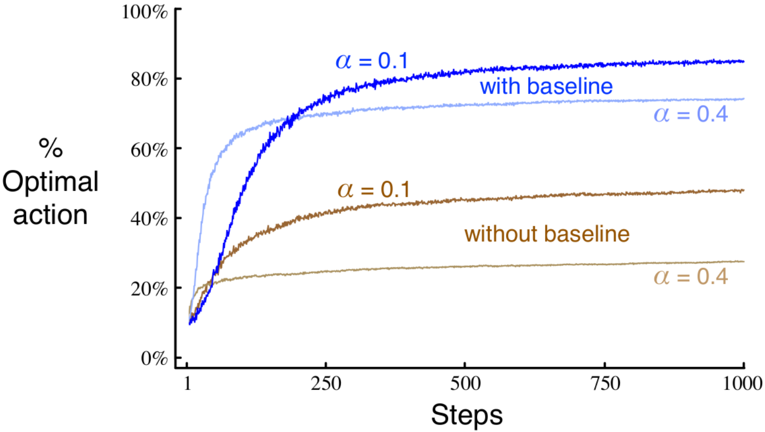
$$H_{t+1}(a) = H_t(a) - \alpha(r_t - \bar{r}_t)\pi_t(a) \; \forall a \neq a_t$$

  - where $\alpha > 0$ and $\bar{r}_t \in R$ is average over all the rewards
  - and initial preferences set to $H_1(a) = 0$

Gradient bandit algorithm performance

Source: Sutton & Barto (2018)

Gradient bandit algorithm performance



Source: Sutton & Barto (2018)

Lower bound on regret

Lower bound on regret

- Asymptotic total regret is at least logarithmic in number of steps (Lai & Robbins, 1985)

$$\lim_{t\to\infty} \geq \log t \sum_{a|\Delta_a>0} \frac{\Delta_a}{KL(\mathcal{R}^a \parallel \mathcal{R}^{a^*})}$$

Lower bound on regret

- Asymptotic total regret is at least logarithmic in number of steps (Lai & Robbins, 1985)

$$\lim_{t\to\infty} \geq \log t \sum_{a|\Delta_a>0} \frac{\Delta_a}{KL(\mathcal{R}^a \parallel \mathcal{R}^{a^*})}$$

- $\log t$ is the important bit, the second term is a constant, roughly task difficulty

Lower bound on regret

- Asymptotic total regret is at least logarithmic in number of steps (Lai & Robbins, 1985)

$$\lim_{t\to\infty} \geq \log t \sum_{a|\Delta_a>0} \frac{\Delta_a}{KL(\mathcal{R}^a \parallel \mathcal{R}^{a^*})}$$

- $\log t$ is the important bit, the second term is a constant, roughly task difficulty
  - KL divergence says how similar the reward distributions of two arms are

Lower bound on regret

- Asymptotic total regret is at least logarithmic in number of steps (Lai & Robbins, 1985)

$$\lim_{t\to\infty} \geq \log t \sum_{a|\Delta_a>0} \frac{\Delta_a}{KL(\mathcal{R}^a \parallel \mathcal{R}^{a^*})}$$

- $\log t$ is the important bit, the second term is a constant, roughly task difficulty
  - KL divergence says how similar the reward distributions of two arms are
  - The difference in expected rewards between the arms is described by the gap, Δ_a

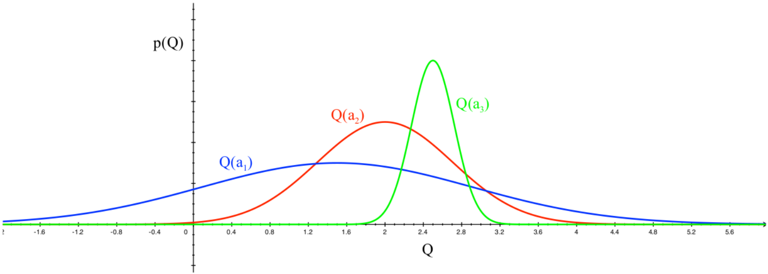
Optimism in the face of uncertainty

Optimism in the face of uncertainty

- Exploration is needed because there is always uncertainty about the accuracy of the action value estimates.
- This suggests we could exploit information about uncertainty!

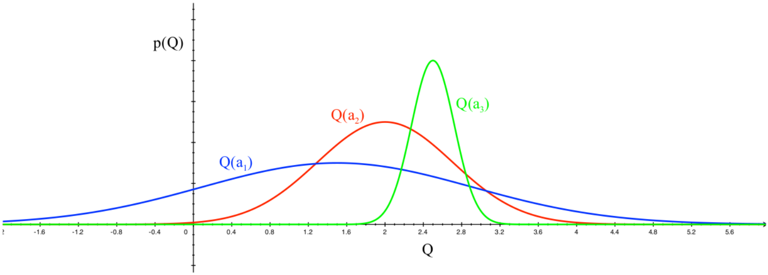
Optimism in the face of uncertainty

- Exploration is needed because there is always uncertainty about the accuracy of the action value estimates.
- This suggests we could exploit information about uncertainty!



Optimism in the face of uncertainty

- Exploration is needed because there is always uncertainty about the accuracy of the action value estimates.
- This suggests we could exploit information about uncertainty!



- Optimistic initialization and fixed uncertainty bonus approaches are based on the same principle

Upper Confidence Bounds (UCB)

Upper Confidence Bounds (UCB)

- The main principle

```
+ Each arm is pulled once to initialize action values,
```

Upper Confidence Bounds (UCB)

- The main principle
  - Estimate an upper confidence $\hat{U}_t(a)$ for each action value, such that $Q(a) \leq \hat{Q}_t(a) + \hat{U}_t(a)$ with high probability

+ Each arm is pulled once to initialize action values,

Upper Confidence Bounds (UCB)

- The main principle
  - Estimate an upper confidence $\hat{U}_t(a)$ for each action value, such that $Q(a) \leq \hat{Q}_t(a) + \hat{U}_t(a)$ with high probability
  - Select action maximising Upper Confidence Bound (UCB)

$$a_t = \mathrm{argmax} \hat{Q}_t(a) + \hat{U}_t(a)$$

+ Each arm is pulled once to initialize action values,

Upper Confidence Bounds (UCB)

- The main principle
  - Estimate an upper confidence $\hat{U}_t(a)$ for each action value, such that $Q(a) \leq \hat{Q}_t(a) + \hat{U}_t(a)$ with high probability
  - Select action maximising Upper Confidence Bound (UCB)

$$a_t = \text{argmax} \hat{Q}_t(a) + \hat{U}_t(a)$$

- UCB1 algorithm (Auer et al, 2002)

$$a_t = \text{argmax}_{a \in \mathcal{A}} \hat{Q}_t(a) + \sqrt{\frac{2 \text{log} t}{N_t(a)}}$$

`+ Each arm is pulled once to initialize action values,`

UCB1 derivation

UCB1 derivation

- Hoeffding's Inequality

UCB1 derivation

- Hoeffding's Inequality
  - Let $X_1, ..., X_t$ be IID random variables in $[0, 1]$, and let $\bar{X}_t = \frac{1}{t}\sum_{\tau=1}^{t} X_\tau$ be the sample mean. Then

$$P[E[X] > \bar{X}_t + u] \leq e^{-2tu^2}$$

UCB1 derivation

- Hoeffding's Inequality
  - Let $X_1, ..., X_t$ be IID random variables in $[0, 1]$, and let $\bar{X}_t = \frac{1}{t}\sum_{\tau=1}^{t} X_\tau$ be the sample mean. Then

$$P[E[X] > \bar{X}_t + u] \leq e^{-2tu^2}$$

- When applied to bandit setting, conditioned on arm a, $P[Q(a) > \hat{Q}_t(a) + U_t(a)] \leq e^{-2N_t(a)U_t(a)^2}$

UCB1 derivation

- Hoeffding's Inequality
  - Let $X_1, ..., X_t$ be IID random variables in $[0, 1]$, and let $\bar{X}_t = \frac{1}{t}\sum_{\tau=1}^{t} X_\tau$ be the sample mean. Then

$$P[E[X] > \bar{X}_t + u] \leq e^{-2tu^2}$$

- When applied to bandit setting, conditioned on arm a, $P[Q(a) > \hat{Q}_t(a) + U_t(a)] \leq e^{-2N_t(a)U_t(a)^2}$

- Solving for $U_t(a)$, $U_t(a) = \sqrt{\frac{-\log p}{2N_t(a)}}$

UCB1 derivation

- Hoeffding's Inequality
  - Let $X_1, ..., X_t$ be IID random variables in $[0, 1]$, and let $\bar{X}_t = \frac{1}{t}\sum_{\tau=1}^{t} X_\tau$ be the sample mean. Then

$$P[E[X] > \bar{X}_t + u] \leq e^{-2tu^2}$$

- When applied to bandit setting, conditioned on arm a, $P[Q(a) > \hat{Q}_t(a) + U_t(a)] \leq e^{-2N_t(a)U_t(a)^2}$

- Solving for $U_t(a)$, $U_t(a) = \sqrt{\frac{-\log p}{2N_t(a)}}$
  - As $t \to \infty$ we want a tendency to select the optimal action, so we reduce p as a function of time, e.g. $p = t^{-4}$

UCB1 derivation

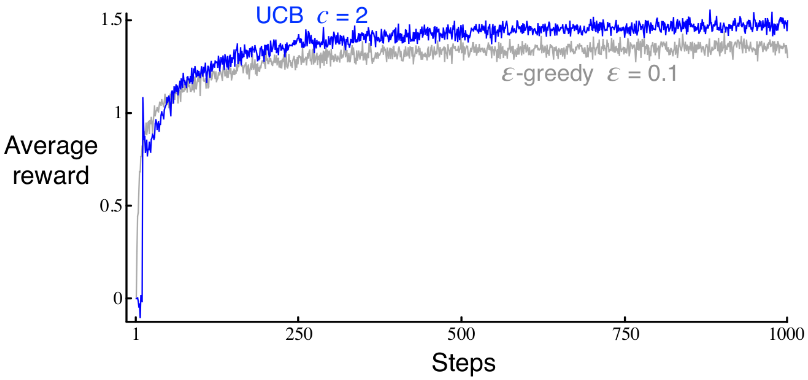
- Hoeffding's Inequality
  - Let $X_1, ..., X_t$ be IID random variables in $[0, 1]$, and let $\bar{X}_t = \frac{1}{t}\sum_{\tau=1}^{t} X_\tau$ be the sample mean. Then

$$P[E[X] > \bar{X}_t + u] \leq e^{-2tu^2}$$

- When applied to bandit setting, conditioned on arm a, $P[Q(a) > \hat{Q}_t(a) + U_t(a)] \leq e^{-2N_t(a)U_t(a)^2}$
- Solving for $U_t(a)$, $U_t(a) = \sqrt{\frac{-\log p}{2N_t(a)}}$
  - As $t \rightarrow \infty$ we want a tendency to select the optimal action, so we reduce p as a function of time, e.g. $p = t^{-4}$
  - We arrive at

$$U_t(a) = \sqrt{\frac{2\log t}{N_t(a)}}$$

UCB performance



Source: Sutton & Barto (2018)

UCB performance

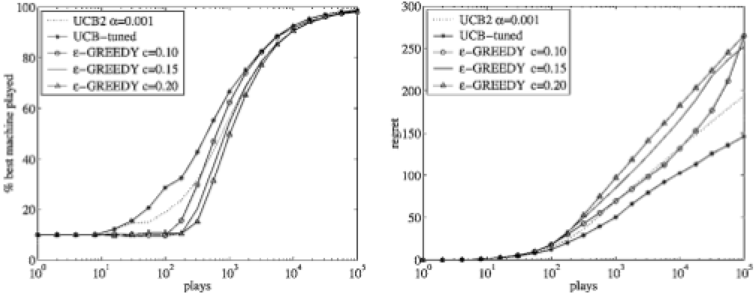


Figure 10. Comparison on distribution 12 (10 machines with parameters 0.9, 0.8, 0.8, 0.8, 0.7, 0.7, 0.7, 0.6, 0.6, 0.6).

Source: Auer, P., Cesa-Bianchi, N., & Fischer, P. (2002). Finite-time analysis of the multiarmed bandit problem. Machine Learning, 47, 235-256.

Bayesian bandits

Bayesian bandits

- So far we have made very few assumptions about the reward distribution R

Bayesian bandits

- So far we have made very few assumptions about the reward distribution R

- With Bayesian approach
  - We can exploit our prior knowledge of rewards, $P[R]$
  - We get full posterior distributions of rewards $P[R|h_t]$

Bayesian bandits

- So far we have made very few assumptions about the reward distribution R
- With Bayesian approach
  - We can exploit our prior knowledge of rewards, $P[R]$
  - We get full posterior distributions of rewards $P[R|h_t]$
- Use posterior instead of counts to guide exploration
  - Bayesian UCB, $a_t = \mathrm{argmax} \mu_a + \beta\sigma_a$
  - Probability matching, selects action a according to probability that a is the optimal action,

$$\pi(a|h_t) = P[Q(a) > Q(a'), \forall a' \neq a|h_t]$$

Bayesian bandits

- So far we have made very few assumptions about the reward distribution R
- With Bayesian approach
  - We can exploit our prior knowledge of rewards, $P[R]$
  - We get full posterior distributions of rewards $P[R|h_t]$
- Use posterior instead of counts to guide exploration
  - Bayesian UCB, $a_t = \text{argmax} \mu_a + \beta \sigma_a$
  - Probability matching, selects action a according to probability that a is the optimal action,

$$\pi(a|h_t) = P[Q(a) > Q(a'), \forall a' \neq a|h_t]$$

- Wrong distribution assumption and priors might cause issues.

Parametric Bayesian approach: Beta-Bernoulli bandit

Parametric Bayesian approach: Beta-Bernoulli bandit

- A generic probabilistic model parametrized by $\mathbf{w}$, with $\mathcal{D}$ denoting the data

Parametric Bayesian approach: Beta-Bernoulli bandit

- A generic probabilistic model parametrized by $\mathbf{w}$, with $\mathcal{D}$ denoting the data
- We can express our prior beliefs about the parameter values through $P[\mathbf{w}]$

Parametric Bayesian approach: Beta-Bernoulli bandit

- A generic probabilistic model parametrized by $\mathbf{w}$, with $\mathcal{D}$ denoting the data
- We can express our prior beliefs about the parameter values through $P[\mathbf{w}]$
- Posterior is then obtained by applying the Bayes rule,

$$P[\mathbf{w}|\mathcal{D}] = \frac{P[\mathcal{D}|\mathbf{w}]P[\mathbf{w}]}{P[\mathcal{D}]}$$

Parametric Bayesian approach: Beta-Bernoulli bandit

- A generic probabilistic model parametrized by $\mathbf{w}$, with $\mathcal{D}$ denoting the data
- We can express our prior beliefs about the parameter values through $P[\mathbf{w}]$
- Posterior is then obtained by applying the Bayes rule,

$$P[\mathbf{w}|\mathcal{D}] = \frac{P[\mathcal{D}|\mathbf{w}]P[\mathbf{w}]}{P[\mathcal{D}]}$$

- Consider the MAB version where reward distribution of each arm follows Bernoulli distribution with unknown parameter $p \in (0, 1)$ with rewards, $r \in 0, 1$

Parametric Bayesian approach: Beta-Bernoulli bandit

- A generic probabilistic model parametrized by $\mathbf{w}$, with $\mathcal{D}$ denoting the data
- We can express our prior beliefs about the parameter values through $P[\mathbf{w}]$
- Posterior is then obtained by applying the Bayes rule,

$$P[\mathbf{w}|\mathcal{D}] = \frac{P[\mathcal{D}|\mathbf{w}]P[\mathbf{w}]}{P[\mathcal{D}]}$$

- Consider the MAB version where reward distribution of each arm follows Bernoulli distribution with unknown parameter $p \in (0, 1)$ with rewards, $r \in 0, 1$
- Reward of each arm is determined by function f that takes index of an arm $a \in 1, ..., K$ and returns parameter p_a

Parametric Bayesian approach: Beta-Bernoulli bandit

- A generic probabilistic model parametrized by $\mathbf{w}$, with $\mathcal{D}$ denoting the data
- We can express our prior beliefs about the parameter values through $P[\mathbf{w}]$
- Posterior is then obtained by applying the Bayes rule,

$$P[\mathbf{w}|\mathcal{D}] = \frac{P[\mathcal{D}|\mathbf{w}]P[\mathbf{w}]}{P[\mathcal{D}]}$$

- Consider the MAB version where reward distribution of each arm follows Bernoulli distribution with unknown parameter $p \in (0,1)$ with rewards, $r \in 0, 1$
- Reward of each arm is determined by function f that takes index of an arm $a \in 1, ..., K$ and returns parameter p_a
- We can fully describe f with parameter $\mathbf{w} \in (0,1)^K$ so that $f_{\mathbf{w}}(a) = w_a$

Parametric Bayesian approach: Beta-Bernoulli bandit

- A generic probabilistic model parametrized by $\mathbf{w}$, with $\mathcal{D}$ denoting the data
- We can express our prior beliefs about the parameter values through $P[\mathbf{w}]$
- Posterior is then obtained by applying the Bayes rule,

$$P[\mathbf{w}|\mathcal{D}] = \frac{P[\mathcal{D}|\mathbf{w}]P[\mathbf{w}]}{P[\mathcal{D}]}$$

- Consider the MAB version where reward distribution of each arm follows Bernoulli distribution with unknown parameter $p \in (0, 1)$ with rewards, $r \in 0, 1$
- Reward of each arm is determined by function f that takes index of an arm $a \in 1, ..., K$ and returns parameter p_a
- We can fully describe f with parameter $\mathbf{w} \in (0, 1)^K$ so that $f_{\mathbf{w}}(a) = w_a$
- Observations are collected in $\mathcal{D}_t = \{(a_\tau, r_\tau)\}_\tau^t$ as a set of tuples, where a_τ identifies the arm and r_τ is the reward

Thompson Sampling for Beta-Bernoulli MAB problem

Thompson Sampling for Beta-Bernoulli MAB problem

- Classical choice for the prior is a conjugate to the Bernoulli likelihood, Beta distribution

$$P[\mathbf{w}|\alpha, \beta] = \prod_{a=1}^{K} \mathrm{Beta}(w_a|\alpha, \beta)$$

Thompson Sampling for Beta-Bernoulli MAB problem

- Classical choice for the prior is a conjugate to the Bernoulli likelihood, Beta distribution

$$P[\mathbf{w}|\alpha, \beta] = \prod_{a=1}^{K} \mathrm{Beta}(w_a|\alpha, \beta)$$

- With such conjugate prior we can efficiently compute the posterior,

$$P[\mathbf{w}|\mathcal{D}] = \prod_{a=1}^{K} \mathrm{Beta}(w_a|\alpha + n_{,1}, \beta + n_{a,0})$$

  - $n_{,1}$ is a count of 1 outcomes whenever for arm a
  - $n_{a,0}$ is a count of 0 outcomes whenever for arm a

Thompson Sampling for Beta-Bernoulli MAB problem

- Classical choice for the prior is a conjugate to the Bernoulli likelihood, Beta distribution

$$P[\mathbf{w}|\alpha, \beta] = \prod_{a=1}^{K} \mathrm{Beta}(w_a|\alpha, \beta)$$

- With such conjugate prior we can efficiently compute the posterior,

$$P[\mathbf{w}|\mathcal{D}] = \prod_{a=1}^{K} \mathrm{Beta}(w_a|\alpha + n_{,1}, \beta + n_{a,0})$$

  - $n_{,1}$ is a count of 1 outcomes whenever for arm a
  - $n_{a,0}$ is a count of 0 outcomes whenever for arm a

- Thompson sampling (Thompson, 1933; Chapelle, Li, 2010)
  - Sample $\mathbf{w}'$ from each posterior and then maximize,

$$a_{t+1} = \mathrm{argmax}_a f_{\mathbf{w}'}(a), \text{where } \mathbf{w}' \sim P[\mathbf{w}|\mathcal{D}_t]$$

  - Thompson sampling achieves Lai and Robbins lower bound!

Algorithm & Example

Algorithm 2: Thompson Sampling for Beta-Bernoulli Bandit

Require: α, β: hyperparameters of the beta prior
Initialize $n_{a,0} = n_{a,1} = i = 0$ for all a
repeat
  for $a = 1, \ldots, K$ **do**
    $\tilde{w}_a \sim \text{beta}(\alpha + n_{a,1}, \beta + n_{a,0})$
  end for
  $a_i = \arg\max_a \tilde{w}_a$
  Observe y_i by pulling arm a_i
  if $y_i = 0$ **then**
    $n_{a_i,0} = n_{a_i,0} + 1$
  else
    $n_{a_i,1} = n_{a_i,1} + 1$
  end if
  $i = i + 1$
until stopping criterion reached

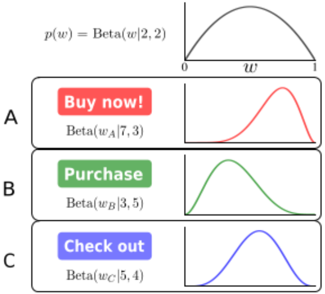


Fig. 2. *Example of the beta-Bernoulli model for A/B testing. Three different buttons are being tested with various colors and text. Each option is given two successes (click-throughs) and two failures as a prior (top). As data are observed, each option updates its posterior over w. Option A is the current best with five successes and only one observed failure.*

Source: Shahriari, B., Swersky, K., Wang, Z., Adams, R. P., & de Freitas, N. (2016).
Taking the Human Out of the Loop: A Review of Bayesian Optimization.
Proceedings of the IEEE, 104(1), 148-175.

Thompson sampling performance