# Lecture 4: Multi-armed bandits extensions and applications

Hrvoje Stojic

May 8, 2020

# Optimal solutions for stationary bandit problem

# MDP formulation and dynamic programming

# MDP formulation and dynamic programming

- Bandit problem can be formulated as an MDP

# MDP formulation and dynamic programming

- Bandit problem can be formulated as an MDP
  - Considering agent's information as a part of state space

# MDP formulation and dynamic programming

- Bandit problem can be formulated as an MDP
  - Considering agent's information as a part of state space
  - Lookahead to determine how information helps in maximizing rewards

# MDP formulation and dynamic programming

- Bandit problem can be formulated as an MDP
  - Considering agent's information as a part of state space
  - Lookahead to determine how information helps in maximizing rewards
- Here is a sketch

# MDP formulation and dynamic programming

- Bandit problem can be formulated as an MDP
  - Considering agent's information as a part of state space
  - Lookahead to determine how information helps in maximizing rewards
- Here is a sketch
  - Agent uses Bayes theorem to update beliefs about parameters $\theta(a)$ that determine the reward distribution

# MDP formulation and dynamic programming

- Bandit problem can be formulated as an MDP
  - Considering agent's information as a part of state space
  - Lookahead to determine how information helps in maximizing rewards
- Here is a sketch
  - Agent uses Bayes theorem to update beliefs about parameters $\theta(a)$ that determine the reward distribution
  - Assume an information set $s_t(a)$ that summarizes agent's beliefs at time $t$ about $\theta(a)$

# MDP formulation and dynamic programming

- Bandit problem can be formulated as an MDP
  - Considering agent's information as a part of state space
  - Lookahead to determine how information helps in maximizing rewards
- Here is a sketch
  - Agent uses Bayes theorem to update beliefs about parameters $\theta(a)$ that determine the reward distribution
  - Assume an information set $s_t(a)$ that summarizes agent's beliefs at time $t$ about $\theta(a)$
  - Postulate a prior distribution at $t = 0$ that describes beliefs about $\theta(a)$, $B_0(a)(\theta(a); s_0(a))$

# MDP formulation and dynamic programming

- Bandit problem can be formulated as an MDP
  - Considering agent's information as a part of state space
  - Lookahead to determine how information helps in maximizing rewards
- Here is a sketch
  - Agent uses Bayes theorem to update beliefs about parameters $\theta(a)$ that determine the reward distribution
  - Assume an information set $s_t(a)$ that summarizes agent's beliefs at time $t$ about $\theta(a)$
  - Postulate a prior distribution at $t = 0$ that describes beliefs about $\theta(a)$, $B_0(a)(\theta(a); s_0(a))$
  - At trial $t$ agent's posterior can be summarised by $B_t(a)(\theta(a); s_t(a))$

# MDP formulation and dynamic programming

- Bandit problem can be formulated as an MDP
  - Considering agent's information as a part of state space
  - Lookahead to determine how information helps in maximizing rewards
- Here is a sketch
  - Agent uses Bayes theorem to update beliefs about parameters $\theta(a)$ that determine the reward distribution
  - Assume an information set $s_t(a)$ that summarizes agent's beliefs at time $t$ about $\theta(a)$
  - Postulate a prior distribution at $t = 0$ that describes beliefs about $\theta(a)$, $B_0(a)(\theta(a); s_0(a))$
  - At trial $t$ agent's posterior can be summarised by $B_t(a)(\theta(a); s_t(a))$
  - Which leads to a familiar Bellman equation: $V(s_t) = \max_{a \in A} E[r_t(a) + \delta V(s_{t+1}) | s_t]$

# MDP formulation and dynamic programming

- Bandit problem can be formulated as an MDP
  - Considering agent's information as a part of state space
  - Lookahead to determine how information helps in maximizing rewards
- Here is a sketch
  - Agent uses Bayes theorem to update beliefs about parameters $\theta(a)$ that determine the reward distribution
  - Assume an information set $s_t(a)$ that summarizes agent's beliefs at time $t$ about $\theta(a)$
  - Postulate a prior distribution at $t = 0$ that describes beliefs about $\theta(a)$, $B_0(a)(\theta(a); s_0(a))$
  - At trial $t$ agent's posterior can be summarised by $B_t(a)(\theta(a); s_t(a))$
  - Which leads to a familiar Bellman equation: $V(s_t) = \max_{a \in A} E[r_t(a) + \delta V(s_{t+1})|s_t]$
- Curse of dimensionality: state space of size $|S|^A$

# MDP formulation and dynamic programming

- Bandit problem can be formulated as an MDP
  - Considering agent's information as a part of state space
  - Lookahead to determine how information helps in maximizing rewards
- Here is a sketch
  - Agent uses Bayes theorem to update beliefs about parameters $\theta(a)$ that determine the reward distribution
  - Assume an information set $s_t(a)$ that summarizes agent's beliefs at time $t$ about $\theta(a)$
  - Postulate a prior distribution at $t = 0$ that describes beliefs about $\theta(a)$, $B_0(a)(\theta(a); s_0(a))$
  - At trial $t$ agent's posterior can be summarised by $B_t(a)(\theta(a); s_t(a))$
  - Which leads to a familiar Bellman equation: $V(s_t) = \max_{a \in A} E[r_t(a) + \delta V(s_{t+1}) | s_t]$
- Curse of dimensionality: state space of size $|S|^A$
- Tractable approximation with Bayesian Adaptive Monte Carlo Planning (Guez, Silver, Dayan, 2012; 2014)

# Gittins indices

# Gittins indices

- For certain problem formulations there are Gittins indices (Gittins & Jones, 1974; Whittle, 1980)

# Gittins indices

- For certain problem formulations there are Gittins indices (Gittins & Jones, 1974; Whittle, 1980)

- A sketch:

# Gittins indices

- For certain problem formulations there are Gittins indices (Gittins & Jones, 1974; Whittle, 1980)

- A sketch:
  - We use a Bayesian setting same as above

# Gittins indices

- For certain problem formulations there are Gittins indices (Gittins & Jones, 1974; Whittle, 1980)

- A sketch:
  - We use a Bayesian setting same as above
  - Compare arm $a$ with virtual arm with fixed reward $\lambda(a)$

# Gittins indices

- For certain problem formulations there are Gittins indices (Gittins & Jones, 1974; Whittle, 1980)

- A sketch:
  - We use a Bayesian setting same as above
  - Compare arm $a$ with virtual arm with fixed reward $\lambda(a)$
  - Bellman equation is then simplified: $V(s_t(a), \lambda(a)) = \max\{\lambda(a) + \delta V(s_t(a), \lambda(a)), E[r_t(a) + \delta V(s_{t+1}, \lambda(a))|s_t(a)]\}$

# Gittins indices

- For certain problem formulations there are Gittins indices (Gittins & Jones, 1974; Whittle, 1980)

- A sketch:
  - We use a Bayesian setting same as above
  - Compare arm $a$ with virtual arm with fixed reward $\lambda(a)$
  - Bellman equation is then simplified: $V(s_t(a), \lambda(a)) = \max\{\lambda(a) + \delta V(s_t(a), \lambda(a)), E[r_t(a) + \delta V(s_{t+1}, \lambda(a))|s_t(a)]\}$
  - Each subproblem only depends on the state evolution of a single arm $a$

# Gittins indices

- For certain problem formulations there are Gittins indices (Gittins & Jones, 1974; Whittle, 1980)

- A sketch:
  - We use a Bayesian setting same as above
  - Compare arm $a$ with virtual arm with fixed reward $\lambda(a)$
  - Bellman equation is then simplified: $V(s_t(a), \lambda(a)) = \max\{\lambda(a) + \delta V(s_t(a), \lambda(a)), E[r_t(a) + \delta V(s_{t+1}, \lambda(a))|s_t(a)]\}$
  - Each subproblem only depends on the state evolution of a single arm $a$
  - Gittins' index $G(s_t(a))$ is the smallest value of $\lambda(a)$ such that the agent at time $t$ is just indifferent between sampling $a$ and receiving the fixed reward

# Gittins indices

- For certain problem formulations there are Gittins indices (Gittins & Jones, 1974; Whittle, 1980)

- A sketch:
  - We use a Bayesian setting same as above
  - Compare arm $a$ with virtual arm with fixed reward $\lambda(a)$
  - Bellman equation is then simplified: $V(s_t(a), \lambda(a)) = \max\{\lambda(a) + \delta V(s_t(a), \lambda(a)), E[r_t(a) + \delta V(s_{t+1}, \lambda(a))|s_t(a)]\}$
  - Each subproblem only depends on the state evolution of a single arm $a$
  - Gittins' index $G(s_t(a))$ is the smallest value of $\lambda(a)$ such that the agent at time $t$ is just indifferent between sampling $a$ and receiving the fixed reward
  - Repeat this for every arm

# Gittins indices

- For certain problem formulations there are Gittins indices (Gittins & Jones, 1974; Whittle, 1980)

- A sketch:
  - We use a Bayesian setting same as above
  - Compare arm $a$ with virtual arm with fixed reward $\lambda(a)$
  - Bellman equation is then simplified: $V(s_t(a), \lambda(a)) = \max\{\lambda(a) + \delta V(s_t(a), \lambda(a)), E[r_t(a) + \delta V(s_{t+1}, \lambda(a))|s_t(a)]\}$
  - Each subproblem only depends on the state evolution of a single arm $a$
  - Gittins' index $G(s_t(a))$ is the smallest value of $\lambda(a)$ such that the agent at time $t$ is just indifferent between sampling $a$ and receiving the fixed reward
  - Repeat this for every arm
  - Choose an arm with largest index in each trial: $\pi(s_t) = \text{argmax}_{a \in A} G(s_t(a))$

# Gittins indices

Source: Lin, Zhang, Hauser (2015)

# Gittins indices

- In summary, we develop an index for each arm by solving a subproblem that involves only that arm

Source: Lin, Zhang, Hauser (2015)

# Gittins indices

- In summary, we develop an index for each arm by solving a subproblem that involves only that arm
- Instead of a problem that is exponential in number of arms, we have a problem that scales linearly with number of arms

Source: Lin, Zhang, Hauser (2015)

# Gittins indices

- In summary, we develop an index for each arm by solving a subproblem that involves only that arm
- Instead of a problem that is exponential in number of arms, we have a problem that scales linearly with number of arms
- However, it's still computationally intensive, $\mathcal{O}(|S|^4)$

Source: Lin, Zhang, Hauser (2015)

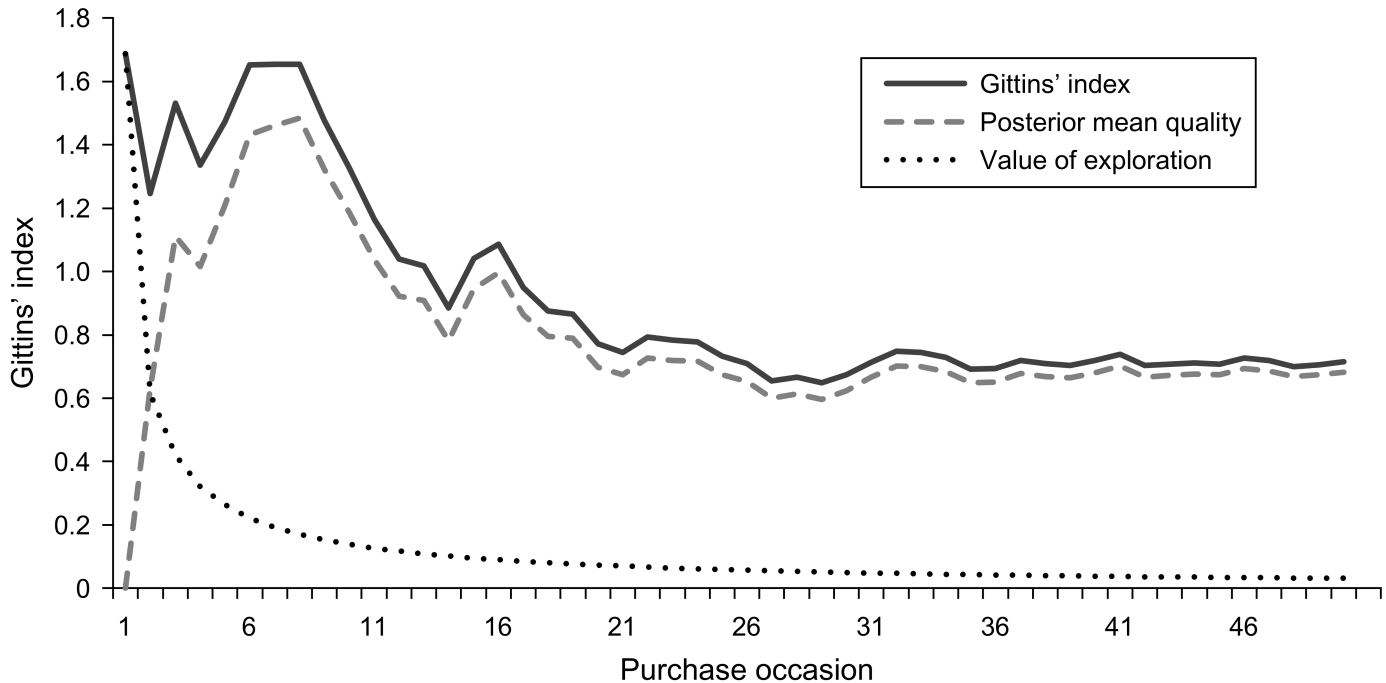
# Gittins indices

- In summary, we develop an index for each arm by solving a subproblem that involves only that arm
- Instead of a problem that is exponential in number of arms, we have a problem that scales linearly with number of arms
- However, it's still computationally intensive, $\mathcal{O}(|S|^4)$

Source: Lin, Zhang, Hauser (2015)

# Gittins indices

- In summary, we develop an index for each arm by solving a subproblem that involves only that arm
- Instead of a problem that is exponential in number of arms, we have a problem that scales linearly with number of arms
- However, it's still computationally intensive, $\mathcal{O}(|S|^4)$




Source: Lin, Zhang, Hauser (2015)

# Restless bandits

# Formulation

- World is often non-stationary!

# Formulation

- World is often non-stationary!
  - For example, restaurants change chefs, service staff, quality of ingredients varies depending on the season, owners change, and the list goes on.

# Formulation

- World is often non-stationary!
  - For example, restaurants change chefs, service staff, quality of ingredients varies depending on the season, owners change, and the list goes on.
- There are many ways to formulate a non-stationary process

# Formulation

- World is often non-stationary!
  - For example, restaurants change chefs, service staff, quality of ingredients varies depending on the season, owners change, and the list goes on.
- There are many ways to formulate a non-stationary process
- One popular way is to assume that the process is a random walk, e.g. some form of a Gaussian process:

$$r_t(a) = Q_t(a) + \epsilon_t(a) \qquad \epsilon_t(a) \sim \mathcal{N}(0, \sigma_\epsilon)$$

$$Q_t(a) = \lambda Q_{t-1}(a) + (1-\lambda)\kappa + \zeta_t(a) \qquad \zeta_t(a) \sim \mathcal{N}(0, \sigma_\zeta)$$

# Formulation

- World is often non-stationary!
  - For example, restaurants change chefs, service staff, quality of ingredients varies depending on the season, owners change, and the list goes on.
- There are many ways to formulate a non-stationary process
- One popular way is to assume that the process is a random walk, e.g. some form of a Gaussian process:

$$r_t(a) = Q_t(a) + \epsilon_t(a) \qquad \epsilon_t(a) \sim \mathcal{N}(0, \sigma_\epsilon)$$

$$Q_t(a) = \lambda Q_{t-1}(a) + (1 - \lambda)\kappa + \zeta_t(a) \qquad \zeta_t(a) \sim \mathcal{N}(0, \sigma_\zeta)$$

  - where $\lambda$ is a decay parameter and $\kappa$ a convergence point

# Formulation

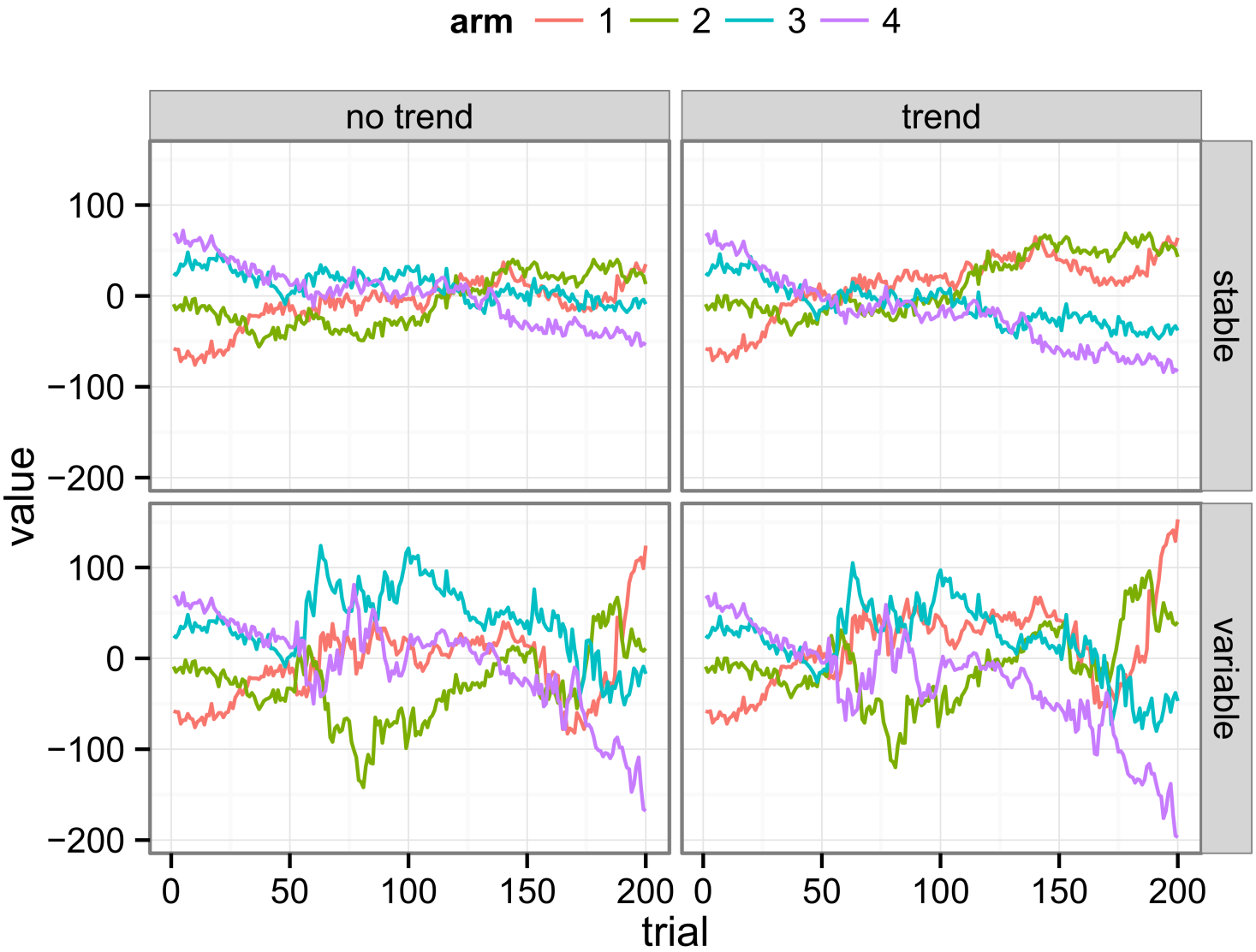
- World is often non-stationary!
  - For example, restaurants change chefs, service staff, quality of ingredients varies depending on the season, owners change, and the list goes on.
- There are many ways to formulate a non-stationary process
- One popular way is to assume that the process is a random walk, e.g. some form of a Gaussian process:

$$r_t(a) = Q_t(a) + \epsilon_t(a) \qquad \epsilon_t(a) \sim \mathcal{N}(0, \sigma_\epsilon)$$

$$Q_t(a) = \lambda Q_{t-1}(a) + (1 - \lambda)\kappa + \zeta_t(a) \qquad \zeta_t(a) \sim \mathcal{N}(0, \sigma_\zeta)$$

  - where $\lambda$ is a decay parameter and $\kappa$ a convergence point
- Solutions for stationary bandits will likely not work well here

# Ornstein-Uhlenbeck process




Source: Speekenbrink & Konstantinidis (2015)

# Learning gets more complex

- It is essential to capture the nature of the process well

# Learning gets more complex

- It is essential to capture the nature of the process well
- For this particular process there is an optimal Bayesian inference algorithm called Kalman filter (Kalman, 1960):

$$\hat{Q}_{t+1}(a) = \hat{Q}_t(a) + \delta_t(a) K_t(a) [r_t - \hat{Q}_t(a)]$$

# Learning gets more complex

- It is essential to capture the nature of the process well
- For this particular process there is an optimal Bayesian inference algorithm called Kalman filter (Kalman, 1960):

$$\hat{Q}_{t+1}(a) = \hat{Q}_t(a) + \delta_t(a) K_t(a) [r_t - \hat{Q}_t(a)]$$

  - where $\delta_t(a)$ is an indicator that arm $a$ was chosen, while "Kalman gain" term $K_t(a)$ acts as a learning rate

$$K_t(a) = \frac{U_t(a) + \sigma_\zeta^2}{U_t(a) + \sigma_\zeta^2 + \sigma_\epsilon^2}$$

# Learning gets more complex

- It is essential to capture the nature of the process well
- For this particular process there is an optimal Bayesian inference algorithm called Kalman filter (Kalman, 1960):

$$\hat{Q}_{t+1}(a) = \hat{Q}_t(a) + \delta_t(a) K_t(a)[r_t - \hat{Q}_t(a)]$$

  - where $\delta_t(a)$ is an indicator that arm $a$ was chosen, while "Kalman gain" term $K_t(a)$ acts as a learning rate

$$K_t(a) = \frac{U_t(a) + \sigma_\zeta^2}{U_t(a) + \sigma_\zeta^2 + \sigma_\epsilon^2}$$

  - $U_t(a)$ is the variance of the posterior distribution of action value, updated as $U_{t+1}(a) = [1 - \delta_t(a) K_t(a)][U_t(a) + \sigma_\zeta^2]$

# Learning gets more complex

- It is essential to capture the nature of the process well
- For this particular process there is an optimal Bayesian inference algorithm called Kalman filter (Kalman, 1960):

$$\hat{Q}_{t+1}(a) = \hat{Q}_t(a) + \delta_t(a) K_t(a)[r_t - \hat{Q}_t(a)]$$

  - where $\delta_t(a)$ is an indicator that arm $a$ was chosen, while "Kalman gain" term $K_t(a)$ acts as a learning rate

$$K_t(a) = \frac{U_t(a) + \sigma_\zeta^2}{U_t(a) + \sigma_\zeta^2 + \sigma_\epsilon^2}$$

  - $U_t(a)$ is the variance of the posterior distribution of action value, updated as $U_{t+1}(a) = [1 - \delta_t(a) K_t(a)][U_t(a) + \sigma_\zeta^2]$
  - $\hat{Q}_0(a)$ and $U_0(a)$ have to be initialized

# Learning gets more complex

- It is essential to capture the nature of the process well
- For this particular process there is an optimal Bayesian inference algorithm called Kalman filter (Kalman, 1960):

$$\hat{Q}_{t+1}(a) = \hat{Q}_t(a) + \delta_t(a) K_t(a)[r_t - \hat{Q}_t(a)]$$

  - where $\delta_t(a)$ is an indicator that arm $a$ was chosen, while "Kalman gain" term $K_t(a)$ acts as a learning rate

$$K_t(a) = \frac{U_t(a) + \sigma_\zeta^2}{U_t(a) + \sigma_\zeta^2 + \sigma_\epsilon^2}$$

  - $U_t(a)$ is the variance of the posterior distribution of action value, updated as $U_{t+1}(a) = [1 - \delta_t(a) K_t(a)][U_t(a) + \sigma_\zeta^2]$
  - $\hat{Q}_0(a)$ and $U_0(a)$ have to be initialized
- Easily combined with UCB or TS, e.g. for UCB: $a_t = \text{argmax}_{a \in \mathcal{A}} \hat{Q}_t(a) + \beta \sqrt{U_t(a)}$

# Learning gets more complex

- It is essential to capture the nature of the process well
- For this particular process there is an optimal Bayesian inference algorithm called Kalman filter (Kalman, 1960):

$$\hat{Q}_{t+1}(a) = \hat{Q}_t(a) + \delta_t(a) K_t(a)[r_t - \hat{Q}_t(a)]$$

  - where $\delta_t(a)$ is an indicator that arm $a$ was chosen, while "Kalman gain" term $K_t(a)$ acts as a learning rate

$$K_t(a) = \frac{U_t(a) + \sigma_\zeta^2}{U_t(a) + \sigma_\zeta^2 + \sigma_\epsilon^2}$$

  - $U_t(a)$ is the variance of the posterior distribution of action value, updated as $U_{t+1}(a) = [1 - \delta_t(a) K_t(a)][U_t(a) + \sigma_\zeta^2]$
  - $\hat{Q}_0(a)$ and $U_0(a)$ have to be initialized
- Easily combined with UCB or TS, e.g. for UCB: $a_t = \text{argmax}_{a \in \mathcal{A}} \hat{Q}_t(a) + \beta \sqrt{U_t(a)}$
  - where $\beta$ is a free parameter, a weight on uncertainty

# Learning gets more complex

- It is essential to capture the nature of the process well
- For this particular process there is an optimal Bayesian inference algorithm called Kalman filter (Kalman, 1960):

$$\hat{Q}_{t+1}(a) = \hat{Q}_t(a) + \delta_t(a) K_t(a)[r_t - \hat{Q}_t(a)]$$

  - where $\delta_t(a)$ is an indicator that arm $a$ was chosen, while "Kalman gain" term $K_t(a)$ acts as a learning rate

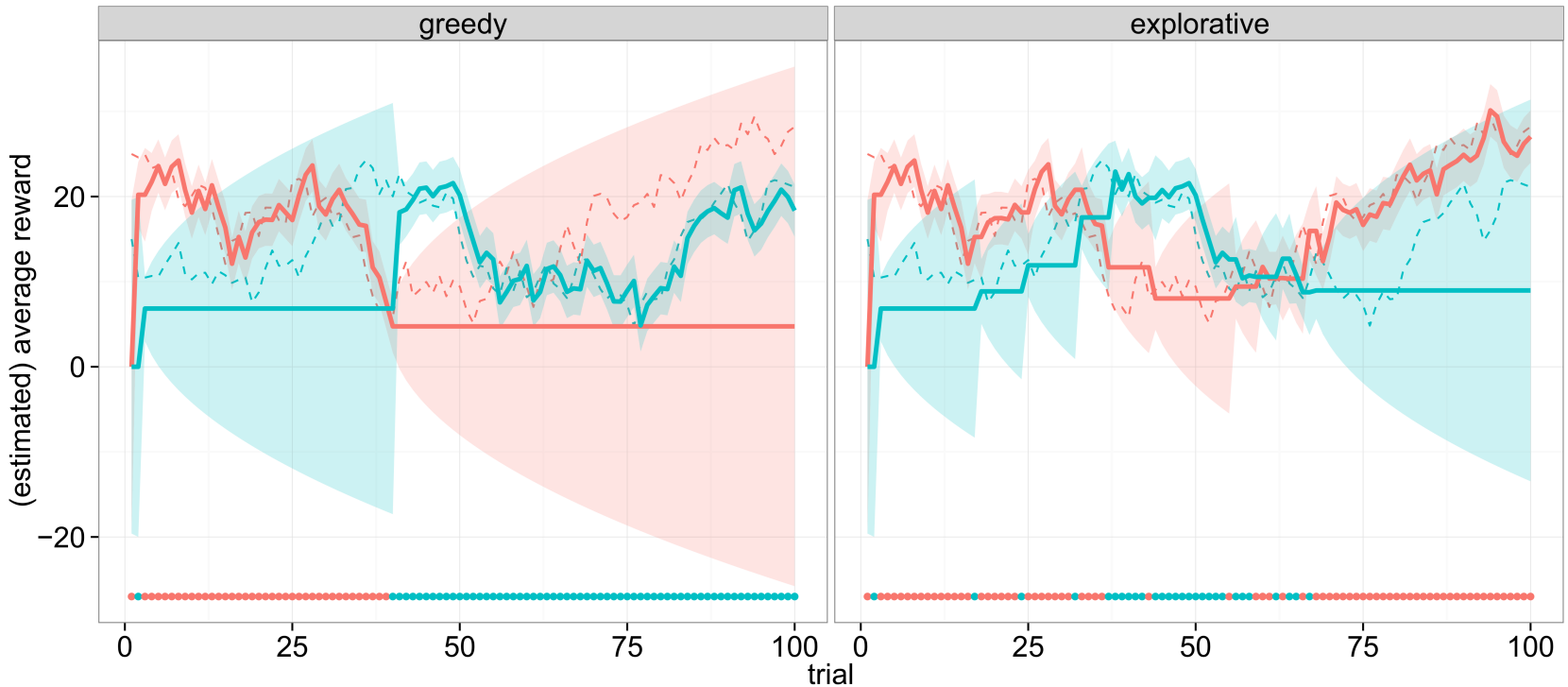
$$K_t(a) = \frac{U_t(a) + \sigma_\zeta^2}{U_t(a) + \sigma_\zeta^2 + \sigma_\epsilon^2}$$

  - $U_t(a)$ is the variance of the posterior distribution of action value, updated as $U_{t+1}(a) = [1 - \delta_t(a) K_t(a)][U_t(a) + \sigma_\zeta^2]$
  - $\hat{Q}_0(a)$ and $U_0(a)$ have to be initialized
- Easily combined with UCB or TS, e.g. for UCB: $a_t = \text{argmax}_{a \in \mathcal{A}} \hat{Q}_t(a) + \beta \sqrt{U_t(a)}$
  - where $\beta$ is a free parameter, a weight on uncertainty
- Other options: sliding-window or discounting UCB (Kocsis & Szepesvári, 2006; Garivier & Moulines, 2011; Auer et al., 2019)

# Tracking uncertainty (correctly) is paramount

Source: Speekenbrink & Konstantinidis (2015)

# Tracking uncertainty (correctly) is paramount




Source: Speekenbrink & Konstantinidis (2015)

# Remarks

# Remarks

- These are more realistic, but theoretically challenging problems

# Remarks

- These are more realistic, but theoretically challenging problems
  - Even when you know the process perfectly

# Remarks

- These are more realistic, but theoretically challenging problems
  - Even when you know the process perfectly
  - Difficult to derive meaningful regret guarantees (Ortner et al., 2012)

# Remarks

- These are more realistic, but theoretically challenging problems
  - Even when you know the process perfectly
  - Difficult to derive meaningful regret guarantees (Ortner et al., 2012)
- There are some optimal solutions

# Remarks

- These are more realistic, but theoretically challenging problems
  - Even when you know the process perfectly
  - Difficult to derive meaningful regret guarantees (Ortner et al., 2012)
- There are some optimal solutions
  - E.g. Whittle, 1988; Papadimitriou & Tsitsiklis, 1999

# Remarks

- These are more realistic, but theoretically challenging problems
  - Even when you know the process perfectly
  - Difficult to derive meaningful regret guarantees (Ortner et al., 2012)
- There are some optimal solutions
  - E.g. Whittle, 1988; Papadimitriou & Tsitsiklis, 1999
  - But they tend to have stringent assumptions

# Remarks

- These are more realistic, but theoretically challenging problems
  - Even when you know the process perfectly
  - Difficult to derive meaningful regret guarantees (Ortner et al., 2012)
- There are some optimal solutions
  - E.g. Whittle, 1988; Papadimitriou & Tsitsiklis, 1999
  - But they tend to have stringent assumptions
  - E.g. change is allowed only for the arm you are observing

# Remarks

- These are more realistic, but theoretically challenging problems
  - Even when you know the process perfectly
  - Difficult to derive meaningful regret guarantees (Ortner et al., 2012)
- There are some optimal solutions
  - E.g. Whittle, 1988; Papadimitriou & Tsitsiklis, 1999
  - But they tend to have stringent assumptions
  - E.g. change is allowed only for the arm you are observing
- If environment is indeed non-stationary

# Remarks

- These are more realistic, but theoretically challenging problems
  - Even when you know the process perfectly
  - Difficult to derive meaningful regret guarantees (Ortner et al., 2012)
- There are some optimal solutions
  - E.g. Whittle, 1988; Papadimitriou & Tsitsiklis, 1999
  - But they tend to have stringent assumptions
  - E.g. change is allowed only for the arm you are observing
- If environment is indeed non-stationary
  - It will usually be difficult to model it correctly

# Remarks

- These are more realistic, but theoretically challenging problems
  - Even when you know the process perfectly
  - Difficult to derive meaningful regret guarantees (Ortner et al., 2012)
- There are some optimal solutions
  - E.g. Whittle, 1988; Papadimitriou & Tsitsiklis, 1999
  - But they tend to have stringent assumptions
  - E.g. change is allowed only for the arm you are observing
- If environment is indeed non-stationary
  - It will usually be difficult to model it correctly
  - And algorithms are not easy to tune

# Remarks

- These are more realistic, but theoretically challenging problems
  - Even when you know the process perfectly
  - Difficult to derive meaningful regret guarantees (Ortner et al., 2012)
- There are some optimal solutions
  - E.g. Whittle, 1988; Papadimitriou & Tsitsiklis, 1999
  - But they tend to have stringent assumptions
  - E.g. change is allowed only for the arm you are observing
- If environment is indeed non-stationary
  - It will usually be difficult to model it correctly
  - And algorithms are not easy to tune
- If you can potentially gather observations that underlie the process, you might be better off switching to a contextual bandit formulation

# Adversarial bandits

# Formulation

- In “real-world” problems . . .

# Formulation

- In “real-world” problems . . .
  - hard to argue rewards are randomly generated

# Formulation

- In "real-world" problems . . .
  - hard to argue rewards are randomly generated
  - sequences of rewards might be correlated

# Formulation

- In "real-world" problems . . .
  - hard to argue rewards are randomly generated
  - sequences of rewards might be correlated
  - in extreme, rewards could be generated by adversaries

# Formulation

- In "real-world" problems . . .
  - hard to argue rewards are randomly generated
  - sequences of rewards might be correlated
  - in extreme, rewards could be generated by adversaries
- Adversarial bandit

# Formulation

- In "real-world" problems . . .
  - hard to argue rewards are randomly generated
  - sequences of rewards might be correlated
  - in extreme, rewards could be generated by adversaries
- Adversarial bandit
  - Assumes almost nothing about what generates rewards

# Formulation

- In "real-world" problems . . .
  - hard to argue rewards are randomly generated
  - sequences of rewards might be correlated
  - in extreme, rewards could be generated by adversaries

- Adversarial bandit
  - Assumes almost nothing about what generates rewards
  - But keeps a (stationary) idea of a single best action

# Formulation

- In "real-world" problems . . .
  - hard to argue rewards are randomly generated
  - sequences of rewards might be correlated
  - in extreme, rewards could be generated by adversaries
- Adversarial bandit
  - Assumes almost nothing about what generates rewards
  - But keeps a (stationary) idea of a single best action
- Arbitrary sequence of rewards for each arm are generated before the game begins, with a constraint that they are from a bounded real interval, $r_1, r_2, ..., r_t \in [0, 1]^A$

# Formulation

- In "real-world" problems . . .
  - hard to argue rewards are randomly generated
  - sequences of rewards might be correlated
  - in extreme, rewards could be generated by adversaries

- Adversarial bandit
  - Assumes almost nothing about what generates rewards
  - But keeps a (stationary) idea of a single best action

- Arbitrary sequence of rewards for each arm are generated before the game begins, with a constraint that they are from a bounded real interval, $r_1, r_2, ..., r_t \in [0, 1]^A$
- Goal is to design a policy $\pi$ that keeps the regret small no matter the sequence of rewards

# Formulation

- In "real-world" problems . . .
  - hard to argue rewards are randomly generated
  - sequences of rewards might be correlated
  - in extreme, rewards could be generated by adversaries

- Adversarial bandit
  - Assumes almost nothing about what generates rewards
  - But keeps a (stationary) idea of a single best action

- Arbitrary sequence of rewards for each arm are generated before the game begins, with a constraint that they are from a bounded real interval, $r_1, r_2, ..., r_t \in [0, 1]^A$
- Goal is to design a policy $\pi$ that keeps the regret small no matter the sequence of rewards

- Performance measure with a *weak regret*:

$$L_T(\pi) = \max_a \sum_{t=1}^{T} r_t(a) - E[\sum_{t=1}^{T} r_t(a_t)]$$

# Exponential-weight algorithm for Exploration and Exploitation (Exp3)

# Exponential-weight algorithm for Exploration and Exploitation (Exp3)

- Originally proposed by Auer, Cesa-Bianchi, Freund & Schapire (2001)

# Exponential-weight algorithm for Exploration and Exploitation (Exp3)

- Originally proposed by Auer, Cesa-Bianchi, Freund & Schapire (2001)

- Given $\gamma \in [0, 1]$, initialize the weights $w_1(a) = 1$ for $a = 1, \ldots, K$.

# Exponential-weight algorithm for Exploration and Exploitation (Exp3)

- Originally proposed by Auer, Cesa-Bianchi, Freund & Schapire (2001)

- Given $\gamma \in [0, 1]$, initialize the weights $w_1(a) = 1$ for $a = 1, \ldots, K$.
- In each round $t = 1, 2, \ldots$

# Exponential-weight algorithm for Exploration and Exploitation (Exp3)

- Originally proposed by Auer, Cesa-Bianchi, Freund & Schapire (2001)

- Given $\gamma \in [0, 1]$, initialize the weights $w_1(a) = 1$ for $a = 1, \ldots, K$.
- In each round $t = 1, 2, \ldots$
  1. Set $p_t(a) = (1 - \gamma) \frac{w_t(a)}{\sum_{a'=1}^{K} w_t(a')} + \frac{\gamma}{K}$ for each $a$

# Exponential-weight algorithm for Exploration and Exploitation (Exp3)

- Originally proposed by Auer, Cesa-Bianchi, Freund & Schapire (2001)

- Given $\gamma \in [0, 1]$, initialize the weights $w_1(a) = 1$ for $a = 1, \ldots, K$.
- In each round $t = 1, 2, \ldots$
  1. Set $p_t(a) = (1 - \gamma) \frac{w_t(a)}{\sum_{a'=1}^{K} w_t(a')} + \frac{\gamma}{K}$ for each $a$
  2. Sample action $a_t \sim p_t(a)$ and observe reward $r_t(a_t) \in [0, 1]$

# Exponential-weight algorithm for Exploration and Exploitation (Exp3)

- Originally proposed by Auer, Cesa-Bianchi, Freund & Schapire (2001)

- Given $\gamma \in [0, 1]$, initialize the weights $w_1(a) = 1$ for $a = 1, \ldots, K$.
- In each round $t = 1, 2, \ldots$
  1. Set $p_t(a) = (1 - \gamma) \frac{w_t(a)}{\sum_{a'=1}^{K} w_t(a')} + \frac{\gamma}{K}$ for each $a$
  2. Sample action $a_t \sim p_t(a)$ and observe reward $r_t(a_t) \in [0, 1]$
  3. For $a = 1, \ldots, K$ set:

# Exponential-weight algorithm for Exploration and Exploitation (Exp3)

- Originally proposed by Auer, Cesa-Bianchi, Freund & Schapire (2001)

- Given $\gamma \in [0, 1]$, initialize the weights $w_1(a) = 1$ for $a = 1, \ldots, K$.
- In each round $t = 1, 2, \ldots$
  1. Set $p_t(a) = (1 - \gamma) \frac{w_t(a)}{\sum_{a'=1}^{K} w_t(a')} + \frac{\gamma}{K}$ for each $a$
  2. Sample action $a_t \sim p_t(a)$ and observe reward $r_t(a_t) \in [0, 1]$
  3. For $a = 1, \ldots, K$ set:
     - estimated action values to $\hat{Q}_t(a) = r_t(a)/p_t(a)$ if $a = a_t$, 0 otherwise

# Exponential-weight algorithm for Exploration and Exploitation (Exp3)

- Originally proposed by Auer, Cesa-Bianchi, Freund & Schapire (2001)

- Given $\gamma \in [0, 1]$, initialize the weights $w_1(a) = 1$ for $a = 1, \ldots, K$.
- In each round $t = 1, 2, \ldots$
  1. Set $p_t(a) = (1 - \gamma) \frac{w_t(a)}{\sum_{a'=1}^{K} w_t(a')} + \frac{\gamma}{K}$ for each $a$
  2. Sample action $a_t \sim p_t(a)$ and observe reward $r_t(a_t) \in [0, 1]$
  3. For $a = 1, \ldots, K$ set:
     - estimated action values to $\hat{Q}_t(a) = r_t(a)/p_t(a)$ if $a = a_t$, 0 otherwise
     - and weights to $w_{t+1}(a) = w_t(a) e^{\gamma \hat{Q}_t(a)/K}$

# Exponential-weight algorithm for Exploration and Exploitation (Exp3)

- Originally proposed by Auer, Cesa-Bianchi, Freund & Schapire (2001)

- Given $\gamma \in [0, 1]$, initialize the weights $w_1(a) = 1$ for $a = 1, \ldots, K$.
- In each round $t = 1, 2, \ldots$
  1. Set $p_t(a) = (1 - \gamma) \frac{w_t(a)}{\sum_{a'=1}^{K} w_t(a')} + \frac{\gamma}{K}$ for each $a$
  2. Sample action $a_t \sim p_t(a)$ and observe reward $r_t(a_t) \in [0, 1]$
  3. For $a = 1, \ldots, K$ set:
     - estimated action values to $\hat{Q}_t(a) = r_t(a)/p_t(a)$ if $a = a_t$, 0 otherwise
     - and weights to $w_{t+1}(a) = w_t(a) e^{\gamma \hat{Q}_t(a)/K}$

- Upper bound on the weak regret: $L_T(\text{Exp3}) \leq (e - 1)\gamma G_{\max}(T) + \frac{K \log K}{\gamma}$

# Performance illustrations

Source: Seldin & Slivkins (2014)

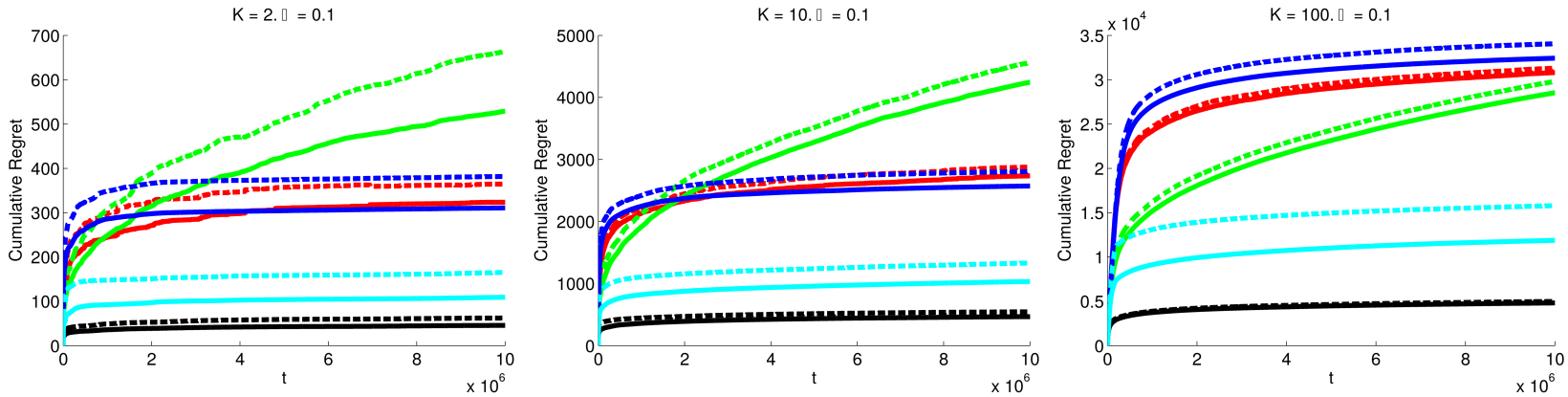
# Performance illustrations

- For an example on real world trading data, see a blog post: UCB1 fares better than Exp3

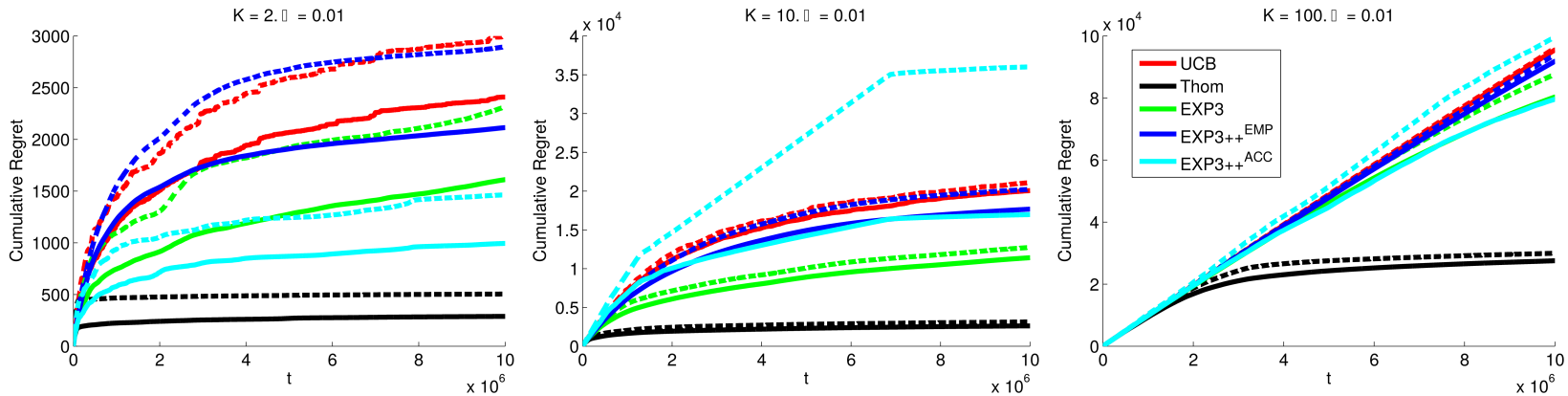
Source: Seldin & Slivkins (2014)

# Performance illustrations

- For an example on real world trading data, see a blog post: UCB1 fares better than Exp3




(a) $K = 2, \Delta = 0.1$ (b) $K = 10, \Delta = 0.1$ (c) $K = 100, \Delta = 0.1$




(d) $K = 2, \Delta = 0.01$ (e) $K = 10, \Delta = 0.01$ (f) $K = 100, \Delta = 0.01$

Source: Seldin & Slivkins (2014)

# Remarks

# Remarks

- Game theory view:

# Remarks

- Game theory view:
  - It's fundamentally a game with two players - the *bandit* and *god*

# Remarks

- Game theory view:
  - It's fundamentally a game with two players - the *bandit* and *god*
  - Best hope is to reduce the advantage that god's intelligence provides

# Remarks

- Game theory view:
  - It's fundamentally a game with two players - the *bandit* and *god*
  - Best hope is to reduce the advantage that god's intelligence provides
  - Hence, randomize - not even omniscient opponents can predict that

# Remarks

- Game theory view:
  - It's fundamentally a game with two players - the *bandit* and *god*
  - Best hope is to reduce the advantage that god's intelligence provides
  - Hence, randomize - not even omniscient opponents can predict that

- A bit too extreme, should be called *paranoid* bandit

# Remarks

- Game theory view:
  - It's fundamentally a game with two players - the *bandit* and *god*
  - Best hope is to reduce the advantage that god's intelligence provides
  - Hence, randomize - not even omniscient opponents can predict that
- A bit too extreme, should be called *paranoid* bandit
  - Real world payoffs are almost never entirely adversarial

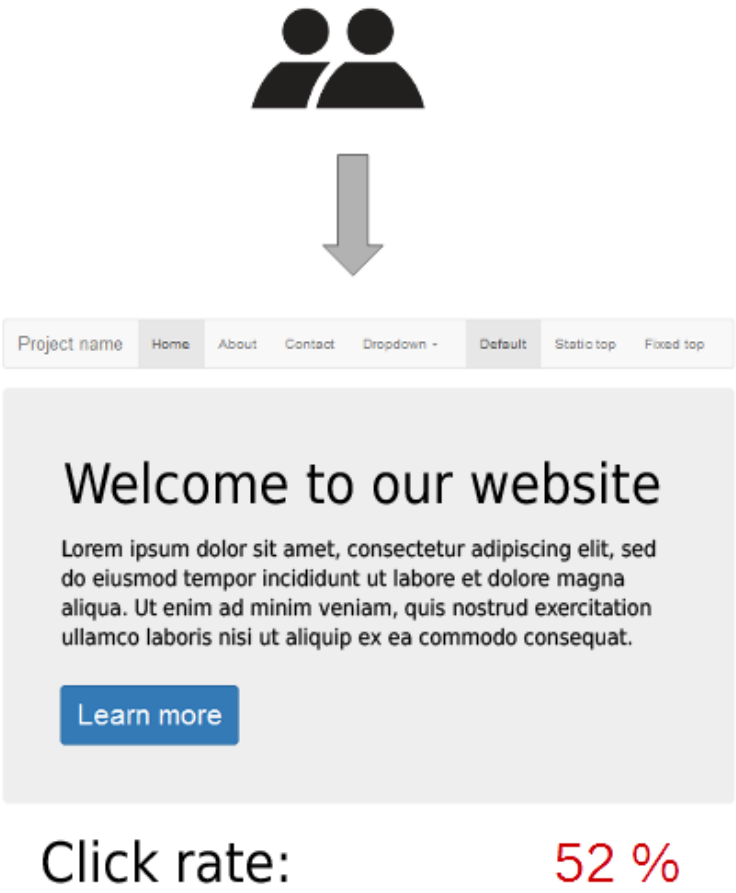
# Remarks

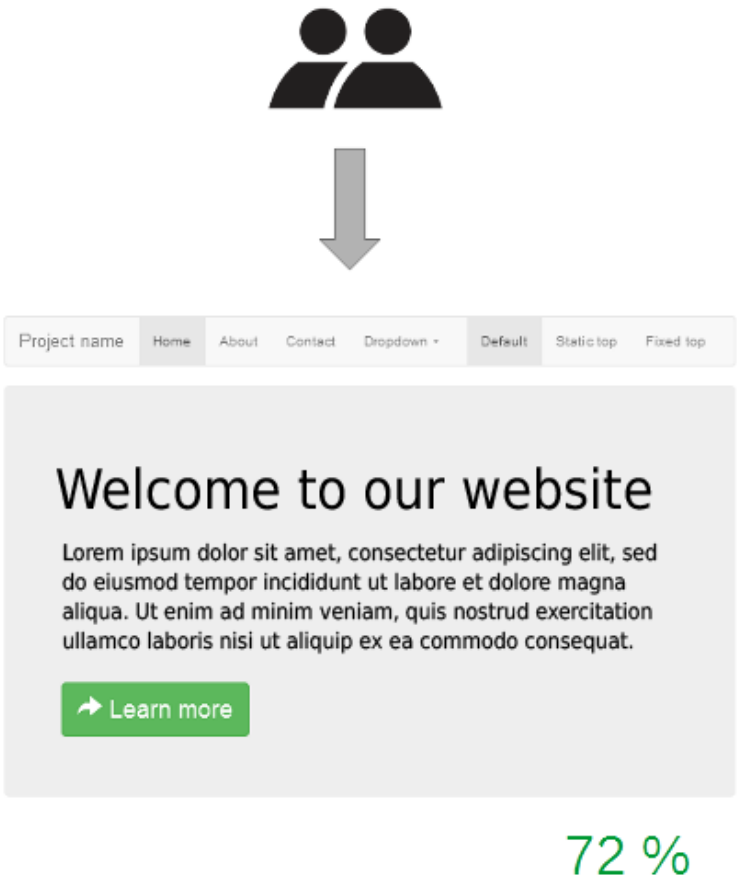
- Game theory view:
  - It's fundamentally a game with two players - the *bandit* and *god*
  - Best hope is to reduce the advantage that god's intelligence provides
  - Hence, randomize - not even omniscient opponents can predict that

- A bit too extreme, should be called *paranoid* bandit
  - Real world payoffs are almost never entirely adversarial
  - But guarantees and performance show that we don't lose much by still using the Exp3

# Applications: A/B testing

# Version of a website that max interaction




Source: Wikipedia

# Displaying ads most likely to be clicked



Source: Me, buying a keyboard to alleviate my wrist pain :)

# Choosing frontpage stories most likely to be read


Source: theguardian.com, May 6, 2020

# Choosing a political ad that attracts the voters (and $)



Source: Optimizely Blog, retrieved May 6, 2020

# Classical hypothesis testing approach

# Classical hypothesis testing approach

- Randomized controlled trial aka experiment:
  - Assign $N$ users randomly to conditions A and B (e.g. default and new website design)
  - Measure clicks and perform a statistical test (e.g. for $\text{CTR}_B - \text{CTR}_A > 0$, perform a Fisher's exact test)
  - If $p < 0.05$, then switch to B
  - Ideally: do a proper power analysis to determine $N$ and no peeking before the end of experiment

# Classical hypothesis testing approach

- Randomized controlled trial aka experiment:
  - Assign $N$ users randomly to conditions A and B (e.g. default and new website design)
  - Measure clicks and perform a statistical test (e.g. for $\text{CTR}_B - \text{CTR}_A > 0$, perform a Fisher's exact test)
  - If $p < 0.05$, then switch to B
  - Ideally: do a proper power analysis to determine $N$ and no peeking before the end of experiment

- Web-based companies run these all the time
  - Various measures: time spent on a page, click-through rates, conversion to sale etc
  - On a small portion of the traffic, users unaware
  - In tools like Google Analytics, Google Website optimizer

# Classical hypothesis testing approach

- Randomized controlled trial aka experiment:
  - Assign $N$ users randomly to conditions A and B (e.g. default and new website design)
  - Measure clicks and perform a statistical test (e.g. for $\text{CTR}_B - \text{CTR}_A > 0$, perform a Fisher's exact test)
  - If $p < 0.05$, then switch to B
  - Ideally: do a proper power analysis to determine $N$ and no peeking before the end of experiment

- Web-based companies run these all the time
  - Various measures: time spent on a page, click-through rates, conversion to sale etc
  - On a small portion of the traffic, users unaware
  - In tools like Google Analytics, Google Website optimizer

- Limitations:
  - You have to expose users to potentially bad outcomes
  - Difficult to scale to many conditions
  - Power analysis is tricky

# Classical hypothesis testing approach

- Randomized controlled trial aka experiment:
  - Assign $N$ users randomly to conditions A and B (e.g. default and new website design)
  - Measure clicks and perform a statistical test (e.g. for $\text{CTR}_B - \text{CTR}_A > 0$, perform a Fisher's exact test)
  - If $p < 0.05$, then switch to B
  - Ideally: do a proper power analysis to determine $N$ and no peeking before the end of experiment

- Web-based companies run these all the time
  - Various measures: time spent on a page, click-through rates, conversion to sale etc
  - On a small portion of the traffic, users unaware
  - In tools like Google Analytics, Google Website optimizer

- Limitations:
  - You have to expose users to potentially bad outcomes
  - Difficult to scale to many conditions
  - Power analysis is tricky

# Multi-armed bandit approach

# Multi-armed bandit approach

- Conditions, i.e. looks, buttons, ads, news articles - arms of the bandit
- Clicks on buttons, ads, news articles can be thought of as rewards
- Since $r_t \in \{0, 1\}$, this is a Bernoulli bandit
- Each condition has true probability of being clicked $\theta(a)$

# Multi-armed bandit approach

- Conditions, i.e. looks, buttons, ads, news articles - arms of the bandit
- Clicks on buttons, ads, news articles can be thought of as rewards
- Since $r_t \in \{0, 1\}$, this is a Bernoulli bandit
- Each condition has true probability of being clicked $\theta(a)$

- Instead of assigning users randomly to experimental conditions, bandit algorithm dynamically adapts assignment of users
- Main idea is that bandit approach will result in less time spent in bad conditions and scale better

# Multi-armed bandit approach

- Conditions, i.e. looks, buttons, ads, news articles - arms of the bandit
- Clicks on buttons, ads, news articles can be thought of as rewards
- Since $r_t \in \{0, 1\}$, this is a Bernoulli bandit
- Each condition has true probability of being clicked $\theta(a)$

- Instead of assigning users randomly to experimental conditions, bandit algorithm dynamically adapts assignment of users
- Main idea is that bandit approach will result in less time spent in bad conditions and scale better

- This can have big business impact, clicks missed out due to more users in bad conditions can result in a loss of revenue

# Which bandit algorithm?

# Which bandit algorithm?

- Choice will depend a bit on exact setting

# Which bandit algorithm?

- Choice will depend a bit on exact setting
- Overall, setup fits very well Bayesian bandits

# Which bandit algorithm?

- Choice will depend a bit on exact setting
- Overall, setup fits very well Bayesian bandits
- Optimal Bayesian learning with Beta-Bernoulli model

# Which bandit algorithm?

- Choice will depend a bit on exact setting
- Overall, setup fits very well Bayesian bandits
- Optimal Bayesian learning with Beta-Bernoulli model

- This calls for a Thompson sampling algorithm

# Which bandit algorithm?

- Choice will depend a bit on exact setting
- Overall, setup fits very well Bayesian bandits
- Optimal Bayesian learning with Beta-Bernoulli model

- This calls for a Thompson sampling algorithm
- But there is also a UCB tuned for Bernoulli bandits (Langford, 2005):

$$\frac{k}{m} + \sqrt{\frac{2\frac{k}{m}\log\frac{1}{\delta}}{m} + \frac{2\log\frac{1}{\delta}}{m}}, \quad \delta = \sqrt{\frac{1}{t}}$$

# Which bandit algorithm?

- Choice will depend a bit on exact setting
- Overall, setup fits very well Bayesian bandits
- Optimal Bayesian learning with Beta-Bernoulli model

- This calls for a Thompson sampling algorithm
- But there is also a UCB tuned for Bernoulli bandits (Langford, 2005):

$$\frac{k}{m} + \sqrt{\frac{2\frac{k}{m}\log\frac{1}{\delta}}{m} + \frac{2\log\frac{1}{\delta}}{m}}, \quad \delta = \sqrt{\frac{1}{t}}$$

  - $k$ is total reward (number of clicks), $m$ number of times the arm has been selected

# Which bandit algorithm?

- Choice will depend a bit on exact setting
- Overall, setup fits very well Bayesian bandits
- Optimal Bayesian learning with Beta-Bernoulli model

- This calls for a Thompson sampling algorithm
- But there is also a UCB tuned for Bernoulli bandits (Langford, 2005):

$$\frac{k}{m} + \sqrt{\frac{2\frac{k}{m}\log\frac{1}{\delta}}{m} + \frac{2\log\frac{1}{\delta}}{m}}, \quad \delta = \sqrt{\frac{1}{t}}$$

  - $k$ is total reward (number of clicks), $m$ number of times the arm has been selected

- Practical considerations:

# Which bandit algorithm?

- Choice will depend a bit on exact setting
- Overall, setup fits very well Bayesian bandits
- Optimal Bayesian learning with Beta-Bernoulli model

- This calls for a Thompson sampling algorithm
- But there is also a UCB tuned for Bernoulli bandits (Langford, 2005):

$$\frac{k}{m} + \sqrt{\frac{2\frac{k}{m}\log\frac{1}{\delta}}{m} + \frac{2\log\frac{1}{\delta}}{m}}, \quad \delta = \sqrt{\frac{1}{t}}$$

  - $k$ is total reward (number of clicks), $m$ number of times the arm has been selected

- Practical considerations:
  - Expensive to constantly train the models

# Which bandit algorithm?

- Choice will depend a bit on exact setting
- Overall, setup fits very well Bayesian bandits
- Optimal Bayesian learning with Beta-Bernoulli model

- This calls for a Thompson sampling algorithm
- But there is also a UCB tuned for Bernoulli bandits (Langford, 2005):

$$\frac{k}{m} + \sqrt{\frac{2\frac{k}{m}\log\frac{1}{\delta}}{m} + \frac{2\log\frac{1}{\delta}}{m}}, \quad \delta = \sqrt{\frac{1}{t}}$$

  - $k$ is total reward (number of clicks), $m$ number of times the arm has been selected

- Practical considerations:
  - Expensive to constantly train the models
  - Instead, make bunch of choices, collect rewards and re-train after a while

# Which bandit algorithm?

- Choice will depend a bit on exact setting
- Overall, setup fits very well Bayesian bandits
- Optimal Bayesian learning with Beta-Bernoulli model

- This calls for a Thompson sampling algorithm
- But there is also a UCB tuned for Bernoulli bandits (Langford, 2005):

$$\frac{k}{m} + \sqrt{\frac{2\frac{k}{m}\log\frac{1}{\delta}}{m} + \frac{2\log\frac{1}{\delta}}{m}}, \quad \delta = \sqrt{\frac{1}{t}}$$

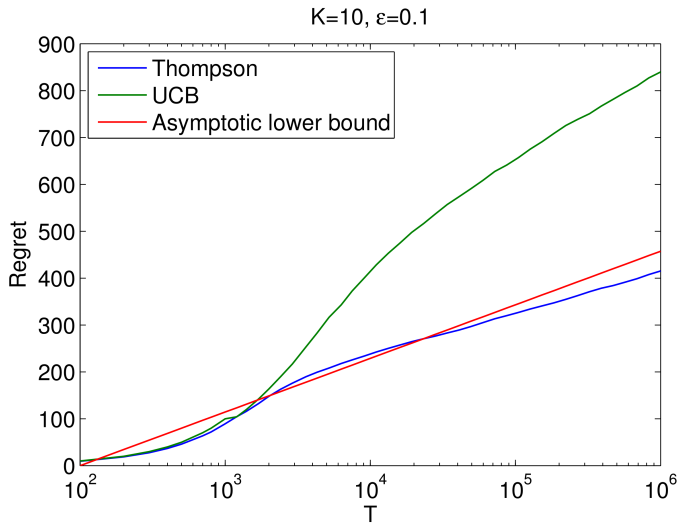
  - $k$ is total reward (number of clicks), $m$ number of times the arm has been selected

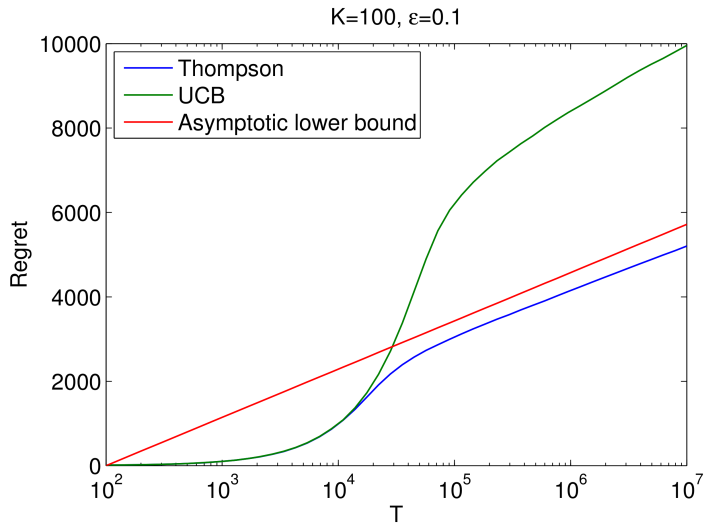
- Practical considerations:
  - Expensive to constantly train the models
  - Instead, make bunch of choices, collect rewards and re-train after a while
  - Deterministic models are not a good idea in this setting

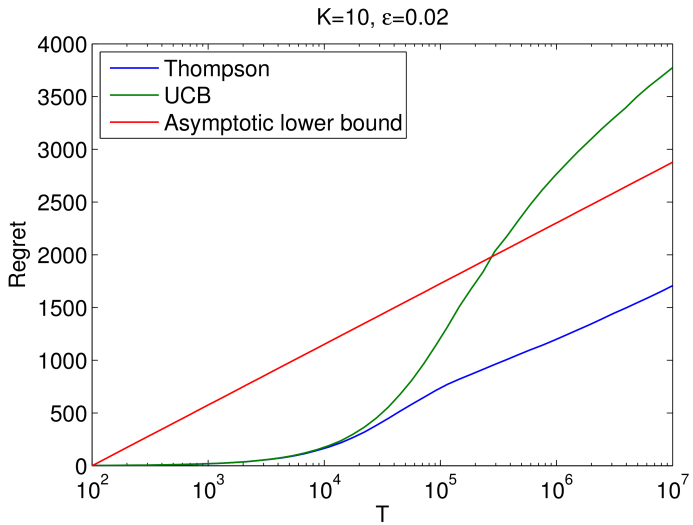
# What can research tell us which algo to choose?

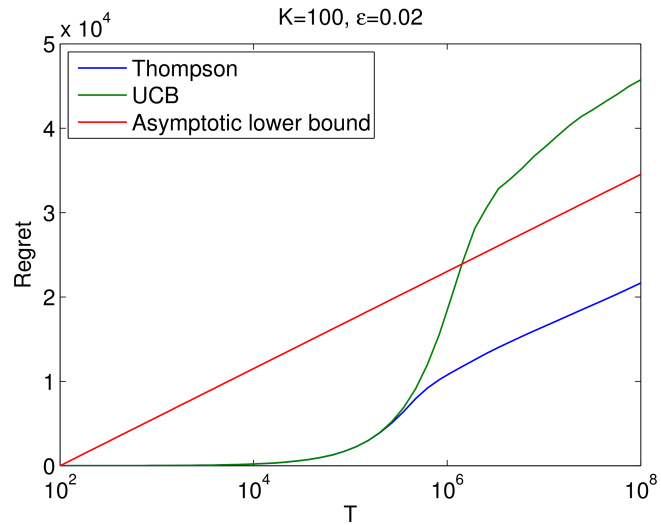
Source: Chapelle & Li (2011)

# What can research tell us which algo to choose?




Source: Chapelle & Li (2011)

# Some ethical dilemmas as well


Source: Dilbert