From Multiarmed Bandits to Stochastic Optimization

Multiarmed Bandits Workshop
Rotterdam, NL

May 24, 2018

Warren B. Powell
Princeton University
Department of Operations Research
and Financial Engineering

© 2018 Warren B. Powell, Princeton University
Materials science

» Optimizing payloads: reactive species, biomolecules, fluorescent markers, …

» Controllers for robotic scientist for materials science experiments

» Optimizing nanoparticles to maximize photoconductivity
Learning problems

- Health sciences
  - Sequential design of experiments for drug discovery
  - Drug delivery – Optimizing the design of protective membranes to control drug release
  - Medical decision making – Optimal learning for medical treatments.
Drug discovery

- Optimizing the configuration of molecules

Design of effective policies can accelerate the search process for new drugs.
Optimal learning in diabetes

How do we find the best treatment for diabetes?

- The standard treatment is a medication called metformin, which works for about 70 percent of patients.
- What do we do when metformin does not work for a patient?
- There are about 20 other treatments, and it is a process of trial and error. Doctors need to get through this process as quickly as possible.
Truckload brokerages

- Now we have a logistic curve for each origin-destination pair \((i,j)\)

\[
P^y(p, a | \theta) = \frac{e^{\theta_i^0 + \theta_{ij} p + \theta_i^a a}}{1 + e^{\theta_i^0 + \theta_{ij} p + \theta_i^a a}}
\]

- Number of offers for each \((i,j)\) pair is relatively small.

- Need to generalize the learning across “traffic lanes.”

- Slides that follow are from senior thesis of Connor Werth ’2017
Ad-click optimization

- Optimizing bids for internet ads
  - In partnership with Roomsage.com
  - Developed Princeton ad-click game
  - Teams compete to find best policy

<table>
<thead>
<tr>
<th>Policy</th>
<th>Profit</th>
</tr>
</thead>
<tbody>
<tr>
<td>PresidentBidness_LA_1</td>
<td>10528</td>
</tr>
<tr>
<td>MaxBidder_LAPS_alpha</td>
<td>8439</td>
</tr>
<tr>
<td>PresidentBidness_PS_1</td>
<td>5553</td>
</tr>
<tr>
<td>Weebs_LA_EZPolicy</td>
<td>3458</td>
</tr>
<tr>
<td>MaxBidder_PS_alpha</td>
<td>2573</td>
</tr>
<tr>
<td>Weebs_LA_MetropolisHastings</td>
<td>1740</td>
</tr>
<tr>
<td>AKCB.LA.1</td>
<td>1471</td>
</tr>
<tr>
<td>pbchen_PS_s4real</td>
<td>790</td>
</tr>
<tr>
<td>BaoWang_PS_WeGo2</td>
<td>599</td>
</tr>
<tr>
<td>MnM_LAPS_M</td>
<td>219</td>
</tr>
<tr>
<td>MmegwaWagnerinterval_estimation</td>
<td>61</td>
</tr>
<tr>
<td>AKCB_PS_1</td>
<td>0</td>
</tr>
<tr>
<td>ohiustina_LA.3</td>
<td>0</td>
</tr>
</tbody>
</table>
Emergency storm response

Hurricane Sandy
» Once in 100 years?
» Rare convergence of events
» But, meteorologists did an amazing job of forecasting the storm.

The power grid
» Loss of power creates cascading failures (lack of fuel, inability to pump water)
» How to plan?
» How to react?
Emergency storm response
Emergency storm response
Emergency storm response
# The “bandit” vocabulary

<table>
<thead>
<tr>
<th>Bandit problem</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Multiarmed bandits</td>
<td>Basic problem with discrete alternatives, online (cumulative regret) learning, lookup table belief model with independent beliefs</td>
</tr>
<tr>
<td>Restless bandits</td>
<td>Truth evolves exogenously over time</td>
</tr>
<tr>
<td>Adversarial bandits</td>
<td>Distributions from which rewards are being sampled can be set by arbitrarily by an adversary</td>
</tr>
<tr>
<td>Continuum-armed bandits</td>
<td>Arms are continuous</td>
</tr>
<tr>
<td>X-armed bandits</td>
<td>Arms are a general topological space</td>
</tr>
<tr>
<td>Contextual bandits</td>
<td>Exogenous state is revealed which affects the distribution of rewards</td>
</tr>
<tr>
<td>Dueling bandits</td>
<td>The agent gets a relative feedback of the arms as opposed to absolute feedback</td>
</tr>
<tr>
<td>Arm-acquiring bandits</td>
<td>New machines arrive over time</td>
</tr>
<tr>
<td>Intermittent bandits</td>
<td>Arms are not always available</td>
</tr>
<tr>
<td>Response surface bandits</td>
<td>Belief model is a response surface (typically a linear model)</td>
</tr>
</tbody>
</table>
The “bandit” vocabulary

<table>
<thead>
<tr>
<th>Bandit problem</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear bandits</td>
<td>Belief is a linear model</td>
</tr>
<tr>
<td>Dependent bandits</td>
<td>A form of correlated beliefs</td>
</tr>
<tr>
<td>Finite horizon bandits</td>
<td>Finite-horizon form of the classical infinite horizon multi-armed bandit problem</td>
</tr>
<tr>
<td>Parametric bandits</td>
<td>Beliefs about arms are described by a parametric belief model</td>
</tr>
<tr>
<td>Nonparametric bandits</td>
<td>Bandits with nonparametric belief models</td>
</tr>
<tr>
<td>Graph-structured bandits</td>
<td>Feedback from neighbors on graph instead of single arm</td>
</tr>
<tr>
<td>Extreme bandits</td>
<td>Optimize the maximum of received rewards</td>
</tr>
<tr>
<td>Quantile-based bandits</td>
<td>The arms are evaluated in terms of a specified quantile</td>
</tr>
<tr>
<td>Preference-based bandits</td>
<td>Find the correct ordering of arms</td>
</tr>
<tr>
<td>Best-arm bandits</td>
<td>Identify the optimal arm with the largest confidence given a fixed budget</td>
</tr>
</tbody>
</table>
Arms…
... and bandits
Multiarmed bandit problems

What is a “bandit problem”?

» The literature seems to characterize a “bandit problem” as any problem where a policy has to balance exploration vs. exploitation.

» But this means that a bandit “problem” is defined by how it is solved. E.g., if you use a pure exploration policy, is it a bandit problem?

My definition:

» Any sequential decision problem which involves learning, and where we have direct or indirect control over the information that is collected.
Multiarmed bandit problems

Dimensions of a “bandit” problem:

» The “arms” (decisions) may be
  • Binary (A/B testing, stopping problems)
  • Discrete alternatives (drug, catalyst, …)
  • Continuous choices (price)
  • Vector-valued (basketball team, products, movies, …)
  • Multiattribute (attributes of a movie, song, person)
  • Static vs. dynamic choice sets
  • Sequential vs. batch

» Information (what we observe)
  • Success-failure/discrete outcome
  • Exponential family (e.g. Gaussian, exponential, …)
  • Heavy-tailed (e.g. Cauchy)
  • Data-driven (distribution unknown)
  • Stationary vs. nonstationary processes
  • Lagged responses?
  • Adversarial?
Multiarmed bandit problems

Dimensions of a “bandit” problem:

» Belief models
  • Lookup tables (these are most common)
    – Independent or correlated beliefs
  • Parametric models
    – Linear or nonlinear in the parameters
  • Nonparametric models
    – Locally linear
    – Deep neural networks/SVM
  • Bayesian vs. frequentist

» Objective function
  • Expected performance (e.g. regret)
  • Offline (final reward) vs. online (cumulative reward)
    – Just interested in final design?
    – Or optimizing while learning?
  • Risk metrics
Outline

- Elements of a sequential decision model
- Mixed state problems
- Designing policies
- Searching for the best policy
Outline

- Elements of a sequential decision model
- Mixed state problems
- Designing policies
- Searching for the best policy
Modeling

Any sequential decision problem consists of five core elements:

» State variables
» Decision variables
» Exogenous information
» Transition function
» Objective function
Modeling dynamic problems

The state variable:

Controls community

\[ x_t = "\text{Information state}" \]

Operations research/MDP/Computer science

\[ S_t = (R_t, I_t, B_t) = \text{System state, where:} \]

\[ R_t = \text{Resource state (physical state)} \]

Location/status of truck/train/plane

Energy in storage

\[ I_t = \text{Information state} \]

Prices

Weather

\[ B_t = \text{Belief state ("state of knowledge")} \]

Belief about traffic delays

Belief about the status of equipment
Modeling dynamic problems

The state variable:

» The initial state $S^0$ contains:
  • All deterministic parameters
  • Initial values of dynamic parameters
  • Prior distribution of belief about unknown parameters

» The dynamic state $S^n$, $n > 0$, contains
  • All information that changes over time.
  • Physical state
    $R^{n+1} = R^n + x^n + \hat{R}^{n+1}$
  • Information state
    $p^{n+1} = p^n + \hat{p}^{n+1}$
  • Belief state (Bayesian updating):
    $\bar{\mu}_x^{n+1} = \frac{\beta^n \bar{\mu}_x^n + \beta^W W^{n+1}}{\beta^n + \beta^W}$
    $\beta_x^{n+1} = \beta_x^n + \beta^W$
Modeling dynamic problems

Decisions:

- Markov decision processes/Computer science
  - $a_t = \text{Discrete action}$
- Control theory
  - $u_t = \text{Low-dimensional continuous vector}$
- Operations research
  - $x_t = \text{Usually a discrete or continuous but high-dimensional vector of decisions.}$

At this point, we do not specify how to make a decision.
Instead, we define the function $X^\pi(s)$ (or $A^\pi(s)$ or $U^\pi(s)$), where $\pi$ specifies the type of policy. "\(\pi\)" carries information about the type of function $f$, and any tunable parameters $\theta \in \Theta^f$. 
The decision variables

- **Styles of decisions**
  - **Binary**
    $ x \in X = \{0, 1\}$
  - **Finite**
    $ x \in X = \{1, 2, \ldots, M\}$ ← Classic bandit model
  - **Continuous scalar**
    $ x \in X = [a, b]$  
  - **Continuous vector**
    $ x = (x_1, \ldots, x_K), \quad x_k \in \mathbb{R}$
  - **Discrete vector**
    $ x = (x_1, \ldots, x_K), \quad x_k \in \mathbb{Z}$
  - **Categorical**
    $ x = (a_1, \ldots, a_I), \quad a_i$ is a category (e.g. patient attributes)
Modeling dynamic problems

- Exogenous information:

\[ W_t = \text{New information that first became known at time } t \]
\[
= (\hat{R}_t, \hat{D}_t, \hat{p}_t, \hat{E}_t)
\]

- \( \hat{R}_t \) = Equipment failures, delays, new arrivals
  - New drivers being hired to the network

- \( \hat{D}_t \) = New customer demands

- \( \hat{p}_t \) = Changes in prices

- \( \hat{E}_t \) = Information about the environment (temperature, ...)

Note: Any variable indexed by \( t \) is known at time \( t \). This convention, which is not standard in control theory, dramatically simplifies the modeling of information.

Below, we let \( \omega \) represent a sequence of actual observations \( W_1, W_2, \ldots \). \( W_t(\omega) \) refers to a sample realization of the random variable \( W_t \).
Modeling dynamic problems

The transition function

\[ S_{t+1} = S^M(S_t, x_t, W_{t+1}) \]

\[ R_{t+1} = R_t + x_t + \hat{R}_{t+1} \] Inventories

\[ p_{t+1} = p_t + \hat{p}_{t+1} \] Spot prices

\[ D_{t+1} = D_t + \hat{D}_{t+1} \] Market demands

\[ \bar{\mu}_x^{n+1} = \beta^n \bar{\mu}_x^n + \beta^W W^{n+1} \]

Bayesian updating of belief

\[ \beta_x^{n+1} = \beta_x^n + \beta^W \]

Also known as the:

“System model”

“State transition model”

“Plant model”

“Plant equation”

“Transition law”

“Transfer function”

“Transformation function”

“Law of motion”

“Model”
Modeling stochastic, dynamic problems

The universal objective function

» Cumulative reward (classical bandit objective)

\[
\max_\pi \mathbb{E}\left\{ \sum_{t=0}^{T} C_t \left( S_t, X_t^\pi (S_t), W_{t+1} \right) \middle| S_0 \right\}
\]

» Final reward ("best arm" bandit objective)

\[
\max_\pi \mathbb{E}F(x^{\pi,N}, \hat{W})
\]

Given a system model (transition function)

\[
S_{t+1} = S^M \left( S_t, x_t, W_{t+1}(\omega) \right)
\]

and a stochastic process:

\[
\left( S_0, W_1, W_2, \ldots, W_T \right)
\]
Stochastic programming
Markov decision processes
Reinforcement learning
Optimal control
Model predictive control
Robust optimization
Approximate dynamic programming
Online computation
Simulation optimization
Stochastic search
Decision analysis
Stochastic control
Simulation optimization
Dynamic Programming and Optimal Control
Introducing Multi-Armed Bandit Allocation Indices
Introduction to Markov Decision Processes
Discrete Stochastic Dynamic Programming
Optimal Control
Online Computation and Competitive Analysis
Stochastic Simulation Optimization
Outline

- Elements of a sequential decision model
- Mixed state problems
- Designing policies
- Searching for the best policy
Modeling dynamic problems

Some major problem classes

» Pure physical state \( S^n = (R^n) \)
  - Inventory problems
  - Stochastic shortest path problems

» Physical plus information \( S^n = (R^n, I^n) \)
  - Inventory with exogenous prices, weather, …

» Pure belief states \( S^n = (B^n) \)
  - These are classical bandit problems
  - Different types of belief models

» Belief plus information \( S^n = (I^n, B^n) \)
  - Patient arriving to doctor’s office who then prescribes a drug.
  - “Contextual bandit problems”

» Everything: \( S^n = (R^n, I^n, B^n) \)
  - Revenue management
  - Clinical trials
Modeling dynamic problems

- Mixed state problems (physical and belief state)
  - Clinical trials
    - Learning the performance of a new drug (belief state)
    - Tracking the number of patients signed up (physical state)
  - Revenue management for hotels
    - Learning market response to price (belief state)
    - Tracking how many rooms have been reserved (physical state)
  - An energy storage problem…
An energy storage problem

Consider a basic energy storage problem:

» We have to manage the flows of energy (blue lines) while managing different sources of uncertainty.
An energy storage problem

- Transition function without learning

\[ E_{t+1} = E_t + \hat{E}_{t+1} \]

\[ p_{t+1} = \theta_0 p_t + \theta_1 p_{t-1} + \theta_2 p_{t-2} + \varepsilon^p_{t+1} \]

\[ D_{t+1} = f_{t,t+1}^D + \varepsilon^D_{t+1} \]

\[ R_{t+1}^{\text{battery}} = R_t^{\text{battery}} + x_t \]
An energy storage problem

Transition function with passive learning

\[ E_{t+1} = E_t + \hat{E}_{t+1} \]
\[ p_{t+1} = \theta_0 p_t + \theta_1 p_{t-1} + \theta_2 p_{t-2} + \epsilon_{t+1}^p \]
\[ D_{t+1} = f^{D}_{t,t+1} + \epsilon_{t+1}^p \]
\[ R^{battery}_{t+1} = R^{battery}_t + x_t \]
Learning in stochastic optimization

- Updating the demand parameter

  » Let $p_{t+1}$ be the new price and let

  $$F^n(x \mid \theta_t) = \theta_{t0} p_t + \theta_{t1} p_{t-1} + \theta_{t2} p_{t-2}$$

  » We update our estimate $\theta_t$ using our recursive least squares equations:

  $$\overline{\theta}_{t+1} = \overline{\theta}_t - \frac{1}{\gamma_{t+1}} B_t \phi_t \varepsilon_{t+1}$$

  $$\varepsilon_{t+1} = F_t(x_t \mid \theta_t) - p_{t+1},$$

  $$B_{t+1} = B_t - \frac{1}{\gamma_{t+1}} \left( B_t \phi(\phi)^T B_t \right)$$

  $$\gamma_{t+1} = 1 + (\phi)^T B_t \phi$$

  $$\phi_t = \begin{pmatrix} p_t \\ p_{t-1} \\ p_{t-2} \end{pmatrix}$$
An energy storage problem

Transition function with active learning

\[ E_{t+1} = E_t + \hat{E}_{t+1} \]
\[ p_{t+1} = \theta_{t0} p_t + \theta_{t1} p_{t-1} + \theta_{t2} p_{t-2} - \theta_{t3} x^{GB} + \varepsilon_{t+1} \]
\[ D_{t+1} = f^{D}_{t,t+1} + \varepsilon_{t+1} \]
\[ R_{t+1}^{battery} = R_t^{battery} + x_t \]
Outline

- Elements of a sequential decision model
- Mixed state problems
- Designing policies
- Searching for the best policy
Designing policies

- We have to start by describing what we mean by a policy.
  » Definition:

  \[ A \text{ policy is a mapping from a state to an action.} \]
  \[ ... \text{ any mapping.} \]

- How do we search over an arbitrary space of policies?
Designing policies

- Two fundamental strategies:

1) Policy search – Search over a class of functions for making decisions to optimize some metric.

\[
\max_{\pi = (f \in F, \theta \in \Theta_f)} \mathbb{E}\left\{ \sum_{t=0}^{T} C_t \left( S_t, X_t^\pi (S_t | \theta) \right) \mid S_0 \right\}
\]

2) Lookahead approximations – Approximate the impact of a decision now on the future.

\[
X_t^*(S_t) = \arg \max_{x_t} \left\{ C(S_t, x_t) + \mathbb{E}\left\{ \max_{\pi \in \Pi} \left\{ \mathbb{E}\left\{ \sum_{t'=t+1}^{T} C(S_t', X_{t'}^\pi (S_t')) \mid S_{t+1} \right\} \mid S_t, x_t \right\} \right\} \right\}
\]
Designing policies

Policy search:

1a) Policy function approximations (PFAs) \( x_t = X^{PFA}(S_t | \theta) \)
   - Lookup tables
     \( \text{“when in this state, take this action”} \)
   - Parametric functions
     - Order-up-to policies: if inventory is less than \( s \), order up to \( S \).
     - Affine policies - \( x_t = X^{PFA}(S_t | \theta) = \sum_{f \in F} \theta_f \phi_f(S_t) \)
     - Neural networks
   - Locally/semi/non parametric
     - Requires optimizing over local regions

1b) Cost function approximations (CFAs)
   - Optimizing a deterministic model modified to handle uncertainty (buffer stocks, schedule slack)
   \[
   X^{CFA}(S_t | \theta) = \arg \max_{x_t} \left( \mu_{tx} + \theta \sigma_{tx} \right)
   \]
Designing policies

Lookahead policies

2a) Value function approximations

We approximate the impact of a decision on the future

\[
X^*_t(S_t) = \arg \max_{x_t} \left\{ C(S_t, x_t) + \mathbb{E} \left[ \max_{\pi \in \Pi} \left\{ \mathbb{E} \sum_{t'=t+1}^{T} C(S_{t'}, X^\pi_{t'}(S_{t'})) \mid S_{t+1} \right\} \mid S_t, x_t \right]\right. \\
\]

Approximating the value of being in a downstream state using machine learning ("value function approximations")

\[
X^*_t(S_t) = \arg \max_{x_t} \left( C(S_t, x_t) + \mathbb{E} \left\{ V_{t+1}(S_{t+1}) \mid S_t, x_t \right\} \right)
\]

\[
X^{VFA}_t(S_t) = \arg \max_{x_t} \left( C(S_t, x_t) + \mathbb{E} \left\{ \tilde{V}_{t+1}(S_{t+1}) \mid S_t, x_t \right\} \right) = \arg \max_{x_t} \left( C(S_t, x_t) + \tilde{V}_t^x(S_t^x) \right)
\]
Designing policies

Lookahead policies

2a) Value function approximations

We approximate the impact of a decision on the future

\[ X_t^* (S_t) = \arg \max_{x_t} \left( C(S_t, x_t) + \mathbb{E} \left\{ \max_{\pi \in \Pi} \left\{ \mathbb{E} \sum_{t'=t+1}^{T} C(S_{t'}, X_{t'}^\pi (S_{t'})) \mid S_{t+1} \right\} \mid S_t, x_t \right\} \right) \]

Approximating the value of being in a downstream state using machine learning ("value function approximations")

\[ X_t^* (S_t) = \arg \max_{x_t} \left( C(S_t, x_t) + \mathbb{E} \left\{ V_{t+1} (S_{t+1}) \mid S_t, x_t \right\} \right) \]

\[ X_t^{VFA} (S_t) = \arg \max_{x_t} \left( C(S_t, x_t) + \mathbb{E} \left\{ \bar{V}_{t+1} (S_{t+1}) \mid S_t, x_t \right\} \right) \]

\[ = \arg \max_{x_t} \left( C(S_t, x_t) + \bar{V}_t^x (S_t^x) \right) \]
Designing policies

- Lookahead policies

2a) Value function approximations

We approximate the impact of a decision on the future

\[ X_t^* (S_t) = \arg \max_{x_t} \left( C(S_t, x_t) + \mathbb{E} \left\{ \max_{\pi \in \Pi} \left\{ \mathbb{E} \sum_{t'=t+1}^{T} C(S_{t'}, X_{t'}^{\pi}(S_{t'})) \mid S_{t+1} \right\} \mid S_t, x_t \right\} \right) \]

Approximating the value of being in a downstream state using machine learning ("value function approximations")

\[ X_t^* (S_t) = \arg \max_{x_t} \left( C(S_t, x_t) + \mathbb{E} \left\{ \hat{V}_{t+1}(S_{t+1}) \mid S_t, x_t \right\} \right) \]

\[ X_t^{VFA} (S_t) = \arg \max_{x_t} \left( C(S_t, x_t) + \mathbb{E} \left\{ \hat{V}_{t+1}^{x}(S_{t+1}) \mid S_t, x_t \right\} \right) \]

= \arg \max_{x_t} \left( C(S_t, x_t) + \hat{V}_t^{x} (S_t^{x}) \right) \]
Designing policies

2b) Direct lookahead policies

\[ X_t^*(S_t) = \arg \max_{x_t} \left( C(S_t, x_t) + \mathbb{E}\left[ \max_{\pi \in \Pi} \mathbb{E}\left[ \sum_{t'\neq t}^{T} C(S_{t'}, X_{t'}(S_{t'})) \mid S_{t+1}, S_t, x_t \right] \right] \right) \]
Designing policies

2b) Direct lookahead policies

» We replace the exact lookahead...

\[ X_t^*(S_t) = \arg \max_{x_t} \left( C(S_t, x_t) + \mathbb{E} \left\{ \max_{\pi \in \Pi} \left\{ \mathbb{E} \sum_{t'=t+1}^{T} C(S_{t'}, X_{t'}^\pi(S_{t'})) \mid S_{t+1} \right\} \mid S_t, x_t \right\} \right) \]

... with an approximation called the lookahead model:

\[ X_t^*(S_t) = \arg \max_{x_t} \left( C(S_t, x_t) + \mathbb{E} \left\{ \max_{\tilde{\pi} \in \tilde{\Pi}} \left\{ \mathbb{E} \sum_{t'=t+1}^{t+H} C(\tilde{S}_{t'}, \tilde{X}_{t'}^{\tilde{\pi}}(\tilde{S}_{t'})) \mid \tilde{S}_{t+1} \right\} \mid \tilde{S}_t, x_t \right\} \right) \]

» A lookahead policy works by approximating the lookahead model.
Designing policies

Types of lookahead approximations

- One-step lookahead – Widely used in pure learning policies:
  - Bayes greedy/naïve Bayes
  - Thompson sampling
  - Value of information (knowledge gradient)

- Multi-step lookahead
  - Deterministic lookahead, also known as model predictive control, rolling horizon procedure
  - Stochastic lookahead:
    - Two-stage (widely used in stochastic linear programming)
    - Multistage
      - Monte carlo tree search (MCTS) for discrete action spaces
      - Multistage scenario trees (stochastic linear programming) – typically not tractable.
### Four (meta)classes of policies

<table>
<thead>
<tr>
<th>Policy search</th>
<th>Lookahead approximations</th>
</tr>
</thead>
<tbody>
<tr>
<td>1) Policy function approximations (PFAs)</td>
<td>» Lookup tables, rules, parametric/nonparametric functions</td>
</tr>
<tr>
<td>2) Cost function approximation (CFAs)</td>
<td>» (X^{CFA}(S_t</td>
</tr>
<tr>
<td>3) Policies based on value function approximations (VFAs)</td>
<td>» (X^{VFA}<em>t(S_t) = \arg \max</em>{x_t} \left( C(S_t, x_t) + \overline{V}_t^x \left( S_t^x(S_t, x_t) \right) \right))</td>
</tr>
<tr>
<td>4) Direct lookahead policies (DLAs)</td>
<td>» Deterministic lookahead rolling horizon proc./model predictive control (X^{LA-D}<em>t(S_t) = \arg \max</em>{\tilde{x}<em>t, \ldots, \tilde{x}</em>{t+H}} C(\tilde{S}<em>t, \tilde{x}<em>t) + \sum</em>{t'=t+1}^{T} C(\tilde{S}</em>{t'}, \tilde{x}_{t'}))</td>
</tr>
<tr>
<td></td>
<td>» Chance constrained programming (P[A_t x_t \leq f(W)] \leq 1 - \delta)</td>
</tr>
<tr>
<td></td>
<td>» Stochastic lookahead /stochastic prog/Monte Carlo tree search (X^{LA-S}<em>t(S_t) = \arg \max</em>{\tilde{x}<em>t, \ldots, \tilde{x}</em>{t+T}} C(\tilde{S}<em>t, \tilde{x}<em>t) + \sum</em>{t'=t+1}^{T} p(\tilde{\omega}) \sum</em>{\tilde{\omega} \in \tilde{\Omega}<em>t} C(\tilde{S}</em>{t'}, (\tilde{\omega}), \tilde{x}_{t'}(\tilde{\omega})))</td>
</tr>
<tr>
<td></td>
<td>» “Robust optimization” (X^{LA-RO}<em>t(S_t) = \arg \max</em>{\tilde{x}<em>t, \ldots, \tilde{x}</em>{t+H}} \min_{w \in W_t(\theta)} C(\tilde{S}<em>t, \tilde{x}<em>t) + \sum</em>{t'=t+1}^{T} C(\tilde{S}</em>{t'}, (w), \tilde{x}_{t'}(w)))</td>
</tr>
</tbody>
</table>
Four (meta)classes of policies

1) Policy function approximations (PFAs)
   » Lookup tables, rules, parametric/nonparametric functions

2) Cost function approximation (CFAs)
   » \[ X^{CFA}(S_t \mid \theta) = \arg \max_{x_t \in \mathbb{X}_t^\pi(\theta)} \mathbb{C}^\pi(S_t, x_t \mid \theta) \]

3) Policies based on value function approximations (VFAs)
   » \[ X^{VFA}(S_t) = \arg \max_{x_t} \left( C(S_t, x_t) + \mathbb{V}_t^x(S_t^x(S_t, x_t)) \right) \]

4) Direct lookahead policies (DLAs)
   » Deterministic lookahead/rolling horizon proc./model predictive control
     \[ X_t^{LA-D}(S_t) = \arg \max_{\tilde{x}_t, \ldots, \tilde{x}_{t+H}} C(\tilde{S}_{tt}, \tilde{x}_{tt}) + \sum_{t'=t+1} C(\tilde{S}_{tt'}, \tilde{x}_{tt'}) \]
   » Chance constrained programming
     \[ P[A_t x_t \leq f(W)] \leq 1 - \delta \]
   » Stochastic lookahead/stochastic prog/Monte Carlo tree search
     \[ X_t^{LA-S}(S_t) = \arg \max_{\tilde{x}_t, \ldots, \tilde{x}_{t+T}} C(\tilde{S}_{tt}, \tilde{x}_{tt}) + \sum_{t'=t+1} p(\tilde{\omega}) \sum_{t'=t+1} C(\tilde{S}_{tt'}, (\tilde{\omega}), \tilde{x}_{tt'}(\tilde{\omega})) \]
   » “Robust optimization”
     \[ X_t^{LA-RO}(S_t) = \arg \max_{\tilde{x}_t, \ldots, \tilde{x}_{t+H}} \min_{w \in W_t(\theta)} C(\tilde{S}_{tt}, \tilde{x}_{tt}) + \sum_{t'=t+1} C(\tilde{S}_{tt'}, (w), \tilde{x}_{tt'}(w)) \]
Four (meta)classes of policies

1) Policy function approximations (PFAs)
   » Lookup tables, rules, parametric/nonparametric functions

2) Cost function approximation (CFAs)
   » \( X^{\text{CFA}}(S_t | \theta) = \arg \max_{x_t \in \hat{x}_t^\pi(\theta)} \bar{C}^\pi(S_t, x_t | \theta) \)

3) Policies based on value function approximations (VFAs)
   » \( X_t^{\text{VFA}}(S_t) = \arg \max_{x_t} \left( C(S_t, x_t) + \bar{V}_t^x(S_t^x(S_t, x_t)) \right) \)

4) Direct lookahead policies (DLAs)
   » Deterministic lookahead rolling horizon proc./model predictive control
     \( X_t^{\text{LA-D}}(S_t) = \arg \max_{\tilde{x}_t, \ldots, \tilde{x}_{t+H}} C(\tilde{S}_t', \tilde{x}_t') + \sum_{t'=t+1} C(\tilde{S}_{t'}, \tilde{x}_{t'}) \)
   » Chance constrained programming
     \( P[A_t x_t \leq f(W)] \leq 1 - \delta \)
   » Stochastic lookahead/stochastic prog/Monte Carlo tree search
     \( X_t^{\text{LA-S}}(S_t) = \arg \max_{\tilde{x}_t, \ldots, \tilde{x}_{t+T}} C(\tilde{S}_t', \tilde{x}_t') + \sum_{\tilde{\omega} \in \tilde{\Omega}_t} \sum_{t'=t+1} \rho(\tilde{\omega}) C(\tilde{S}_{t'}, (\tilde{\omega}), \tilde{x}_{t'}(\tilde{\omega})) \)
   » “Robust optimization”
     \( X_t^{\text{LA-RO}}(S_t) = \arg \max_{\tilde{x}_t, \ldots, \tilde{x}_{t+H}} \min_{w \in W_t(\theta)} C(\tilde{S}_t', \tilde{x}_t') + \sum_{t'=t+1} C(\tilde{S}_{t'}, (w), \tilde{x}_{t'}'(w)) \)
Policies for pure learning problems

1) Policy function approximation (PFA)

- Revenue maximization problem
  - Demand function
    \[ D(p | \bar{\theta}^n) = \bar{\theta}_1^n - \bar{\theta}_2^n p \]
  - Revenue
    \[ R(p | \bar{\theta}^n) = pD(p) = \bar{\theta}_1^n p - \bar{\theta}_2^n p^2 \]
  - PFA policy – pure exploitation
    \[ p^n = \frac{\bar{\theta}_1^n}{2\bar{\theta}_2^n} \]
  - PFA policy with active exploration ("excitation policy")
    \[ p^n = \frac{\bar{\theta}_1^n}{2\bar{\theta}_2^n} + \varepsilon^n \quad \varepsilon^n \sim N(0, \sigma^\varepsilon) \]
  - Need to tune \( \sigma^\varepsilon \)
Policies for pure learning problems

1) Policy function approximation (PFA)
   » Linear decision rules ("affine policies")

\[ X^{PFA}(S^n | \theta) = \theta_0 + \theta_1 \phi_1(S^n) + \theta_2 \phi_2(S^n) + \ldots + \theta_F \phi_F(S^n) \]

» Neural networks
Policies for pure learning problems

2) Cost function approximations (CFA)
   » Upper confidence bounding

\[
X^{UCB} (S^n | \theta^{UCB}) = \arg \max_x \left( \mu_x^n + \theta^{UCB} \sqrt{\frac{\log n}{N_x^n}} \right)
\]

» Interval estimation

\[
X^{IE} (S^n | \theta^{IE}) = \arg \max_x \left( \mu_x^n + \theta^{IE} \bar{\sigma}_x^n \right)
\]

» Boltzmann exploration ("soft max")
   • Choose \( x \) with probability:

\[
P^n_x (\theta) = \frac{e^{\theta \bar{\mu}_x^n}}{\sum_{x'} e^{\theta \bar{\mu}_{x'}^n}}
\]

\[
X^{Boltz} (S^n | \theta) = \arg \max_x \{ x | P^n_x (\theta) \leq U \}.
\]
Policies for pure learning problems

- A learning problem with correlated beliefs
Policies for pure learning problems

- Picking $\theta^{IE} = 0$ means we are evaluating each choice at the mean.
Policies for pure learning problems

- Picking $\theta^{I_E} = 2$ means we are evaluating each choice at the 95th percentile.
Policies for pure learning problems

- PFAs and CFAs have to be tuned

  » Final reward (“offline learning”)

  \[
  \max_{\theta^{IE}} \mathbb{E} F(x^{\pi,N}, \hat{W}) = \mathbb{E}_\mu \mathbb{E}_{W^1,\ldots,W^N|\mu} \mathbb{E}_{\hat{W}} (x^{\pi,N}(\theta^{IE}), \hat{W})
  \]

  » Cumulative reward (“online learning”)

  \[
  \max_{\theta^{IE}} \mathbb{E}^\pi \left\{ \sum_{t=0}^{T} C_t \left( S_t, X^\pi_t(S_t | \theta^{IE}), W_{t+1} \right) | S_0 \right\}
  \]

  » Both require searching over tunable parameters.

  • Offline tuning is classical stochastic search
  • Online tuning is a relatively open research area
Cost function approximations

- Tuning the interval estimation policy

\[ X_{IE}^{n} (S) = \arg \max_{x} \bar{\mu}_{x}^{n} + \theta_{IE}^{n} \bar{\sigma}_{x}^{n} \]
3) Policies based on value function approximations
   » VFAs using a physical state problem

\[ V^n(S^n) = \max_x \left( C(S^n, x) + E \left\{ V^{n+1}(S^{n+1} | S^n) \right\} \right) \]

Current node (e.g. node 2)
Policies for pure learning problems

3) Policies based on value function approximations
   » VFAs using a physical state problem

\[
V^n(S^n) = \max_x \left( C(S^n, x) + E \left\{ V^{n+1}(S^{n+1}) \mid S^n \right\} \right)
\]

Decision to go to a node (e.g. 5)

Downstream node (e.g. 5)
Policies for pure learning problems

3) Policies based on value function approximations

» VFAs using a learning problem

\[ V^n(S^n) = \max_x \left( C(S^n, x) + E\{V^{n+1}(S^{n+1}) \mid S^n\} \right) \]

\[ S_5 = N(\mu_5, \sigma_5^2) \]

\[ S^n = (S_1^n, \ldots, S_5^n) \]
Policies for pure learning problems

3) Policies based on value function approximations

» Illustration: finding the best drug in the set \( \mathcal{X} \in \{x_1, x_2, \ldots, x_M\} \).

» After a test we observe success or failure:

\[
W_{x}^{n+1} = \begin{cases} 
1 & \text{Success} \\
0 & \text{Failure}
\end{cases} \quad \text{If } x^n = x
\]

» Let \( \rho_x \) = Probability that drug \( x \) is successful. We assume that

\[
\rho_x | S^n \sim \text{Beta}(\alpha_x^n, \beta_x^n)
\]

where \( S^n = (\alpha^n, \beta^n) \) is our belief state, with updating equations:

\[
\alpha_x^{n+1} = \alpha_x^n + W_x^{n+1}, \quad \beta_x^{n+1} = \beta_x^n + (1 - W_x^{n+1})
\]
Policies for pure learning problems

3) Policies based on value function approximations

- Bellman’s equation:

\[ V^n(\alpha^n, \beta^n) = \max_x \mathbb{E}_x [W_{x}^{n+1} + \gamma V^{n+1}(\alpha^n + W^{n+1}, \beta^n + 1 - W^{n+1}) | S^n] \]

- This can be solved for a stopping problem to determine when to stop testing a single drug.

- Problematic if \( \alpha^n \) and \( \beta^n \) are vectors. Gittins developed a novel decomposition that allows us to solve this problem for one drug (“arm”) at a time.
Policies for pure learning problems

3) Policies based on value function approximations

» For normally distributed rewards, Gittins (1974) showed that we can solve dynamic programs for each alternative.

» Produces a policy that looks like

$$X^{Gitt} (S^n) = \arg \max_x \left( \bar{\mu}^n_x + \sigma^n W \Gamma \left( \frac{\sigma^n_x}{\sigma^n_w}, \gamma \right) \right)$$

where $$\Gamma \left( \frac{\sigma^n_x}{\sigma^n_w}, \gamma \right)$$ is the “Gittins index” obtained by solving a dynamic program for whether to continue or stop testing a single drug.

» Considered a computational breakthrough, but computing Gittins indices is still a challenge, and only applies to special cases.
Policies for pure learning problems

4) Policies based on direct lookaheads (DLA)

» The knowledge gradient for offline (final reward):

\[ \nu_{x}^{KG,n} = E\{ \max_{y} F(y, B^{n+1}|x) \} = \max_{y} F(y, B^{n}) \]

- Finding the new design with our new belief (but without knowing the outcome of the experiment)
- Choosing the best design given what we know now.
- Averaging over the possible outcomes of the experiment (and our different beliefs about parameters)
- Updated parameter estimates after running experiment with density \( x \).
- Proposed experiment

Current belief state
The knowledge gradient

4) Policies based on direct lookaheads (DLA)

» The knowledge gradient computes the expected improvement from a single experiment

Change in estimate of value of option 5 due to measurement.

Change which produces a change in the decision.
The knowledge gradient

- Estimated value of alternative
- Standard deviation
- Knowledge gradient
The knowledge gradient
The knowledge gradient
The knowledge gradient

Some properties of the knowledge gradient for offline (final reward) problems.

- \( \nu_x^{KG,n} \geq 0 \)
- Asymptotically optimal (finds best \( x \) in the limit)
- Optimal (by construction) if budget = 1.
- Optimal for all \( n \) if number of alternatives = 2 (e.g. A/B testing).
- Only stationary policy that is both myopically and asymptotically optimal.

For online problems

- Asymptotically optimal (finds best \( x \) in the limit) as \( \gamma \rightarrow 1 \)
FINITE-TIME ANALYSIS FOR THE KNOWLEDGE-GRADIENT POLICY

YINGFEI WANG† AND WARREN B. POWELL‡

Abstract. We consider sequential decision problems in which we adaptively choose one of finitely many alternatives and observe a stochastic reward. We offer a new perspective on interpreting Bayesian ranking and selection problems as adaptive stochastic multiset maximization problems and derive the first finite-time bound of the knowledge-gradient policy for adaptive submodular objective functions. In addition, we introduce the concept of prior-optimality and provide another insight into the performance of the knowledge-gradient policy based on the submodular assumption on the value of information. We demonstrate submodularity for the two-alternative case and provide other conditions for more general problems, bringing out the issue and importance of submodularity in learning problems. Empirical experiments are conducted to further illustrate the finite-time behavior of the knowledge-gradient policy.

Key words. ranking and selection, sequential decision analysis, stochastic control

AMS subject classifications. 62F07, 62F15, 62L05, 93E35, 68W40, 68T05

DOI. 10.1137/16M1073388

1. Introduction. We consider sequential decision problems in which at each time step, we choose one of finitely many alternatives and observe a random reward. The rewards are independent of each other and follow some unknown probability distribution. One goal can be to identify the alternative with the best expected performance within a limited measurement budget, which is the objective of Bayesian ranking and selection problems. Ranking and selection problems are exam-
The knowledge gradient

Different belief models

- Lookup tables
  - Independent beliefs
  - Correlated beliefs

- Linear parametric models
  - Linear models
  - Sparse-linear
  - Tree regression

- Nonlinear parametric models
  - Logistic regression
  - Neural networks

- Nonparametric models
  - Gaussian process regression
  - Kernel regression
  - Support vector machines
  - Deep neural networks
The knowledge gradient

- The marginal value of information
  - Repeatedly sampling the same alternative

Value of information

Number of times we sample the same alternative
The knowledge gradient

- The marginal value of information
  - The value of information may be concave if an experiment is noisy

![Graph showing the relationship between the number of times we sample the same alternative and the value of information.](image-url)
The knowledge gradient

- The marginal value of information
  - The value of information may be concave if an experiment is noisy

\[ n_x^* = \arg\max_{n_x > 0} \frac{v_x(n_x)}{n_x} \]
The knowledge gradient

From offline to online learning

» The knowledge gradient computes the value of information for a terminal reward objective:

$$v_{x}^{KG,n} = E \left\{ \max_y F(y, B^{n+1}(x)) \right\} - \max_y F(y, B^n)$$

» Imagine that we have a budget of $N$ experiments, and that we are summing rewards over this horizon. The value of information from a single experiment is now

$$v_{x}^{KG-OL,n} = \bar{\mu}_x^n + (N - n) v_{x}^{KG,n}$$

Expected reward

Remaining horizon

Offline KG
The knowledge gradient

Knowledge gradient for offline and online learning

Offline learning

\[ \nu_{x}^{KG,n} \]

Online learning

\[ \nu_{x}^{KG-OL,n} = \bar{\mu}_{x}^{n} + (N - n)\nu_{x}^{KG,n} \]

This bridges what have historically been fundamentally different fields.
Outline

- Elements of a sequential decision model
- Mixed state problems
- Designing policies
- Searching for the best policy
Designing policies

Finding the best policy

» We have to first articulate our classes of policies

\[ f \in \mathcal{F} = \{PFAs, CFAs, VFAs, DLAs\} \]

\[ \theta \in \Theta^f = \text{Parameters that characterize each family.} \]

» So minimizing over \( \pi \in \Pi \) means:

\[ \Pi = \{ f \in \mathcal{F}, \theta \in \Theta^f \} \]

» We then have to pick an objective such as

\[
\max_{\pi} \mathbb{E}\left\{ \sum_{t=0}^{T} C_t \left( S_t, X^\pi(S_t | \theta) \right) | S_0 \right\}
\]

or

\[
\max_{\pi} \mathbb{E}\left\{ F(X^\pi_T, W) | S_0 \right\}
\]
Multiarmed bandit problems

Policy search class
  » Policies tend to be relatively simple and easy to compute
  » Well suited to rapid (e.g. internet speed) learning applications needing fast computation.
  » Tuning is important, and typically requires a realistic simulator.

Lookahead class
  » Policies can be relatively complex to compute.
  » Well suited to problems with expensive experiments.
  » Typically avoids tuning, but may require a prior.
Multiarmed bandit problems

Notes:

» *Any* of the four classes of policies may be appropriate depending on the characteristics of the problem.

» Active learning arises in many applications, but is often overlooked.

» The “bandit” culture of coming up with problem variations should be inherited by other communities.

» Bandit researchers often focus on good but not optimal policies (e.g. UCB policies) with good characteristics (e.g. robust across a wide range of distributions).
MOLTE

- Modular, optimal learning testing environment
  - Matlab-based environment with modular library of problems and algorithms, each in its own .m file.
  - User specifies in a spreadsheet which algorithms are run on which problems

<table>
<thead>
<tr>
<th>Problem class</th>
<th>Prior</th>
<th>Measurement Budget</th>
<th>Belief Model</th>
<th>Offline/Online</th>
<th>Number of Policies</th>
</tr>
</thead>
<tbody>
<tr>
<td>PayloadD</td>
<td>MLE</td>
<td>0.2</td>
<td>independent</td>
<td>Offline</td>
<td>4 kriging, EXPL, IE(1.7), Thompson Sampling</td>
</tr>
<tr>
<td>Branin</td>
<td>MLE</td>
<td>10</td>
<td>correlated</td>
<td>Online</td>
<td>4 OLkgcb, UCBEcB(*)</td>
</tr>
<tr>
<td>Bubeck4</td>
<td>uninformative</td>
<td>5</td>
<td>independent</td>
<td>Online</td>
<td>4 OLKG, UCB, SR, UCBV</td>
</tr>
<tr>
<td>GPR</td>
<td>Default</td>
<td>0.3</td>
<td>correlated</td>
<td>Offline</td>
<td>4 kriging, kcb, IE(*)</td>
</tr>
</tbody>
</table>

http://www.castlelab.princeton.edu/software/
MOLTE

- Comparison on library problems
Princeton ad-click game

In collaboration with Roomsage.com
Princeton ad-click game

- Learning the bid-response curve

- Varies by hour of week
- Response depends on location, age, gender, device
Princeton ad-click game

The ad-click game:
» Learn the best policy for bidding for ads
» Bids compete in a simulated auction following the rules used by Google

<table>
<thead>
<tr>
<th>Policy</th>
<th>profit</th>
</tr>
</thead>
<tbody>
<tr>
<td>PresidentBidness_LA_1</td>
<td>10528</td>
</tr>
<tr>
<td>MaxBidder_LAPS_alpha</td>
<td>8439</td>
</tr>
<tr>
<td>PresidentBidness_PS_1</td>
<td>5553</td>
</tr>
<tr>
<td>Weebs_LA_EZPolicy</td>
<td>3458</td>
</tr>
<tr>
<td>MaxBidder_PS_alpha</td>
<td>2573</td>
</tr>
<tr>
<td>Weebs_LA_MetropolisHastings</td>
<td>1740</td>
</tr>
<tr>
<td>AKCB_LA_1</td>
<td>1471</td>
</tr>
<tr>
<td>pbchen_PS_s4real</td>
<td>790</td>
</tr>
<tr>
<td>BaoWang_PS_WeGo2</td>
<td>599</td>
</tr>
<tr>
<td>MnM_LAPS_M</td>
<td>219</td>
</tr>
<tr>
<td>MmegwaWagnerinterval_estimation</td>
<td>61</td>
</tr>
<tr>
<td>AKCB_PS_1</td>
<td>0</td>
</tr>
<tr>
<td>ohiustina_LA_3</td>
<td>0</td>
</tr>
<tr>
<td>ohiustina_PS_3</td>
<td>0</td>
</tr>
<tr>
<td>TnT_PS_M</td>
<td>0</td>
</tr>
<tr>
<td>ConnorDozie_PS</td>
<td>-7</td>
</tr>
<tr>
<td>pbchen_LA_s4real</td>
<td>-42</td>
</tr>
<tr>
<td>BaoWang_PS_WeGo</td>
<td>-54</td>
</tr>
<tr>
<td>ConnorDozie_LAPS</td>
<td>-1007</td>
</tr>
<tr>
<td>BreyerJohnson_LA_3</td>
<td>-1242</td>
</tr>
<tr>
<td>BreyerJohnson_PS_3</td>
<td>-7132</td>
</tr>
<tr>
<td>WagnerM_Megwa_LAPS</td>
<td>-13344</td>
</tr>
<tr>
<td>tw5_PS</td>
<td>-27302</td>
</tr>
</tbody>
</table>
Thank you!

For more information, please visit:

http://www.castlelab.Princeton.edu

See “Courses” or the “jungle” webpages.