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.
Meeting variability with *portfolios* of generation with mixtures of *dispatchability*
Real-time logistics

- **Uber**
  - Provides real-time, on-demand transportation.
  - Drivers are encouraged to enter or leave the system using pricing signals and informational guidance.

- **Decisions:**
  - How to price to get the right balance of drivers relative to customers.
  - Assigning and routing drivers to manage Uber-created congestion.
  - Real-time management of drivers.
  - Pricing (trips, new services, …)
  - Policies (rules for managing drivers, customers, …)
Planning for a risky world

**Disaster response**
- Robust design of emergency response networks.
- Design of sensor networks and communication systems to manage responses to hurricanes, tsunamis, nuclear disasters and terrorist attacks.

**Disease**
- Management of medical personnel, equipment and vaccines to respond to a disease outbreak.
- Robust design of supply chains to mitigate the disruption of transportation systems.
Designing robust power grids

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?
Modeling

Before we can solve complex problems, we have to know how to think about them.

The biggest challenge when making decisions under uncertainty is modeling.
For deterministic problems, we speak the language of mathematical programming

» Linear programming:
\[
\min_x cx
\]
\[
Ax = b
\]
\[
x \geq 0
\]

» For time-staged problems
\[
\min_{x_0, \ldots, x_T} \sum_{t=0}^{T} c_t x_t
\]
\[
A_t x_t - B_{t-1} x_{t-1} = b_t
\]
\[
D_t x_t \leq u_t
\]
\[
x_t \geq 0
\]

Arguably Dantzig’s biggest contribution, more so than the simplex algorithm, was his articulation of optimization problems in a standard format, which has given algorithmic researchers a common language.
Modeling

For deterministic problems, we speak the language of mathematical programming

» Linear programming:

\[
\min_x \ c x
\]
\[
A x = b
\]
\[
x \geq 0
\]

» Optimal control:

\[
\min_{u_0, \ldots, u_T} \sum_{t=0}^{T} L(x_t, u_t) + J_T(x_T)
\]

» For time-staged problems

\[
\min_{x_0, \ldots, x_T} \sum_{t=0}^{T} c_t x_t
\]
\[
A_t x_t - B_{t-1} x_{t-1} = b_t
\]
\[
D_t x_t \leq u_t
\]
\[
x_t \geq 0
\]
Outline

- Canonical problems
- Problem classes
- Solution strategies for learning problems
- Elements of a dynamic model
- An energy storage illustration
- Modeling uncertainty
- Designing policies
- The four classes of policies
- From deterministic to stochastic optimization
Outline

- Canonical problems
- Problem classes
- Solution strategies for learning problems
- Elements of a dynamic model
- An energy storage illustration
- Modeling uncertainty
- Designing policies
- The four classes of policies
- From deterministic to stochastic optimization
Canonical problems

- Decision trees
Canonical problems

- Stochastic search (derivative based)
  
  » Basic problem:
  
  \[ \max_x \mathbb{E} F(x, W) \]

  \[
  \begin{aligned}
  \text{Manufacturing network (x=design)} \\
  \text{Unit commitment problem (x=day ahead decisions)} \\
  \text{Transformers (x=replacement policy)} \\
  \text{Inventory system (x=design, replenishment policy)} \\
  \text{Battery system (x=choice of material)} \\
  \text{Patient treatment cost (x=drug, treatments)} \\
  \text{Trucking company (x=fleet size and mix)}
  \end{aligned}
  \]

  » Stochastic gradient
  
  \[ x^{n+1} = x^n + \alpha_n \nabla_x F(x^n, W^{n+1}) \]

  » Convergence:
  
  \[ \lim_{n \to \infty} \mathbb{E} F(x^n, W) \to \mathbb{E} F(x^*, W) \]
Canonical problems

Ranking and selection (derivative free)

» Basic problem:

$$\max_{x_1, \ldots, x_M} \mathbb{E} F(x, W)$$

» We need to design a policy $X^\pi (S^n)$ that

$$\max_\pi \mathbb{E} F( X^{\pi,N} , W )$$
Canonical problems

Multi-armed bandit problems

- We do not know the expected winnings from each slot machine (“arm”).
- Collect information by playing a machine.
- We need to find a policy $X^\pi(S^n)$ for playing machine $x$ that maximizes:

$$\max_\pi \mathbb{E} \sum_{n=0}^{N-1} W_{x^n}^{n+1}$$

where

- $W^{n+1} =$ "winnings"
- $S^n =$ State of knowledge
- $x^n = X^\pi(S^n)$

New information

- What we know about each slot machine

Choose next “arm” to play
Canonical problems

- **Two-stage stochastic programming**
  - Make initial decision $x_0$
    - How many Christmas trees to plant
  - See information $W_1$
    - See the orders for Christmas trees from retailers
  - Make final decision $x_1$
    - Shipping Christmas trees to retailers

- **Optimization model**
  \[
  \min_{x_0} c_0 x_0 + \mathbb{E} Q_1(x_0, W_1)
  \]
  where
  \[
  Q_1(x_0, W_1(\omega)) = \min_{x_1(\omega) \in X_1(\omega)} c_1(\omega) x_1(\omega)
  \]
  This is often solved using
  \[
  \min c_0 x_0 + \sum_{\omega \in \Omega} p(\omega) \sum_{t=1}^T c_t(\omega) x_t(\omega)
  \]
Canonical problems

Multi-stage stochastic programming

» The stochastic programming community likes to write:

$$\min_{A_0 x_0 = b_0, x_0 \geq 0} \langle c_0, x_0 \rangle + \mathbb{E} \left[ \min_{B_0 x_0 + A_1 x_1 = b_1, x_1 \geq 0} \langle c_1, x_1 \rangle + \mathbb{E} \left[ \cdots + \mathbb{E} \left[ \min_{B_{T-1} x_{T-1} + A_T x_T = b_T, x_T \geq 0} \langle c_T, x_T \rangle \right] \cdots \right] \right]$$

» This is the same as:

$$\min_{x_0 \in \mathcal{X}_0(S_0)} C(S_0, x_0) + \mathbb{E} \left[ \min_{x_1 \in \mathcal{X}_1(S_1)} C(S_1, x_1) + \mathbb{E} \left[ \cdots + \mathbb{E} \left[ \min_{x_T \in \mathcal{X}_T(S_T)} C(S_T, x_T) \right] \cdots \right] \right]$$

» …which is the same as

$$\min_\pi E \left\{ \sum_{t=0}^{T} C(S_t, X_t^\pi(S_t)) \mid S_0 \right\}$$
Canonical problems

(Discrete) Markov decision processes

» Bellman’s optimality equation

\[ V_t(S_t) = \min_{a_t \in \mathcal{A}} \left( C(S_t, a_t) + \gamma \mathbb{E}\{ V_{t+1}(S_{t+1}) \mid S_t \} \right) \]

\[ = \min_{a_t \in \mathcal{A}} \left( C(S_t, a_t) + \gamma \sum_{s'} p(S_{t+1} = s' \mid S_t, a_t) V_{t+1}(S_{t+1}) \right) \]

» where

- \( S_t \) = Discrete state (node in network, items in inventory)
- \( a_t \) = Action (transition to node, purchases)
- \( W_t \) = Random information (demand, prices, wind, deposits)

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

» Solve starting at \( t=T \) with \( V_T(S_T) = 0 \) and step backward in time.
Canonical problems

**Linear quadratic regulation (LQR)**

» A popular optimal control problem in engineering involves solving:

\[
\min_{u_0,\ldots,u_T} \mathbb{E} \sum_{t=0}^{T} \left( (x_t)^T Q x_t + (u_t)^T R u_t \right)
\]

» where:

\[
x_t = \text{State at time } t
\]

\[
u_t = \text{Control at time } t \text{ (must be } F_t - \text{measurable)}
\]

\[
x_{t+1} = f(x_t, u_t) + w_t \quad (w_t \text{ is random at time } t)
\]

» Possible to show that the optimal policy looks like:

\[
U_t^\pi(x_t) = K_t x_t
\]

where \(K_t\) is a complicated function of \(Q\) and \(R\).
Canonical problems

Optimal stopping - Find the best time to stop and sell an asset

» Model:
  • Exogenous process:
    \[ \omega = (p_1, p_2, \ldots, p_T) = \text{Sequence of stock prices} \]
  • Decision:
    \[ X_t(\omega) = \begin{cases} 
      1 & \text{If we stop and sell at time } t \\
      0 & \text{Otherwise} 
    \end{cases} \]
  • Reward:
    \[ f(p_t) = \text{Reward received if we stop at time } t \text{ (e.g. } f(p_t) = p_t) \]

» Optimization problem:

\[ \max_\tau \mathbb{E} X_\tau f(p_\tau) \]

where \( \tau \) is a “stopping time” (or "\( F_\tau \) – measurable function")
3.1.3 - The Optimization Problem

We assume that we are given an $\mathcal{F}_T$-measurable random variable representing the terminal cost. It is assumed to be square integrable. Most often, it will be of the form $g(X_T)$, where $g : \Omega \times \mathbb{R}^d \to \mathbb{R}$ is $\mathcal{F}_T \times \mathcal{B}(\mathbb{R}^d)$-measurable, and of polynomial growth in $x \in \mathbb{R}^d$ uniformly in $\omega \in \Omega$. We also assume that the cost includes a running cost given by a function $f : [0, T] \times \Omega \times \mathbb{R}^d \times A \to \mathbb{R}$ satisfying the same assumptions (S1) and (S2) as the drift $b$. Finally, we define the cost functional $J$ by

$$J(\alpha) = \mathbb{E} \left[ \int_0^T f(s, X_s, \alpha_s) ds + g(X_T) \right], \quad \alpha \in \mathbb{A}. \quad (3.4)$$

As explained earlier, the goal of a stochastic control problem is to find an admissible control $\alpha \in \mathbb{A}$ which minimizes the cost functional $J(\alpha)$. The cost functional $J$ is often called the objective, or objective functional.

» This is “MCCM” - mathematically correct, computationally meaningless.
Canonical problems

Engineers like to write

$$\max_{x_0,\ldots,x_T} \mathbb{E} \sum_{t=0}^{T} C(S_t, x_t)$$

» This way of modeling is astonishingly common in the engineering literature, but it is simply incorrect - $x_t$ is a random variable. This does not model the flow of information.

Mathematicians like to write

$$\max_{x_0,\ldots,x_T} \mathbb{E} \sum_{t=0}^{T} C(S_t, x_t)$$

where $x_t$ is $F_t$ - measurable.

» This is mathematically correct, but with no path to computation.
Better:

» Maximize over policies:

$$\max_{\pi} \mathbb{E} \left\{ \sum_{t=0}^{T} C(S_t, X_t^{\pi}(S_t)) \mid S_0 \right\}$$

where $X_t^{\pi}(S_t)$ is a function of the state $S_t$.

» Now we just have to show how to search over policies. This is likely to look like:

$$\max_{f \in F, \theta \in \Theta^f} \mathbb{E} \left\{ \sum_{t=0}^{T} C(S_t, X_t^{\pi}(S_t)) \mid S_0 \right\}$$

where $f \in F$ is function classes, $\theta \in \Theta^f$ is tunable parameters.

» In this tutorial, we are going to show that all of these canonical problems can be modeled this way.
Outline

- Canonical problems
- Problem classes
- Solution strategies for learning problems
- Elements of a dynamic model
- An energy storage illustration
- Modeling uncertainty
- Designing policies
- The four classes of policies
- From deterministic to stochastic optimization
Problem classes

- Staging of information and decisions
  - Static stochastic optimization:
    - Decision, information.
  - Two-stage stochastic programming (vector $x$):
    - Decision, information, decision.
  - Multistage stochastic programming (vector $x$):
    - Decision, information, decision, information, …, decision.
  - Finite horizon Markov decision process (finite actions):
    - Decision, information, decision, information, …, decision.
  - Asymptotic stochastic search:
    - Decision, information, decision, information, …
  - Infinite horizon Markov decision process (finite actions):
    - Decision, information, decision, information, decision, …

- Contextual information
  - Each problem above starts with initial information from an exogenous source (the “context”).
Problem classes

- Learning problems (state independent)
  - Arises when we are trying to optimize an unknown function (black box simulation, lab experiment, newsvendor with unknown distribution):
    \[
    \max_x \mathbb{E} F(x,W) = \mathbb{E} \left\{ p \min(x,W) - cx \right\}
    \]
  - The state variable is
    \[
    S_t = K_t = \text{Our state of knowledge about } \mathbb{E} F(x,W)
    \]
  - Transition:
    \[
    x_t = X^\pi(S_t) \rightarrow \hat{F}_{t+1} = F(x_t,W_{t+1}) \rightarrow K_{t+1}
    \]
    \[
    S_{t+1} = K_{t+1}
    \]
    \[
    (S_0 = K_0, x_0 = X^\pi(S_0), W_1, \hat{F}_1 = F(x_0,W_1), S_1 = K_1, x_1 = X^\pi(S_1), \ldots)
    \]
Problem classes

- **Parametric belief model**
  - We have a nonlinear model $f(x \mid \theta)$ with uncertain $\theta$
  - Knowledge is $p^n = \left( p^n_k \right)$, $p^n_k = \text{Prob}[\theta = \theta_k]$
Problem classes

- State-dependent problems
  - Dynamic information process
  - Imagine price $p_t$ is revealed before making a decision:

\[
\max_x \mathbb{E} F(x, W) = \mathbb{E}\left\{ p_t \min(x, W) - cx \right\}
\]

- State variable is now $S_t = (p_t, K_t)$
- Transitions:

\[
x_t = X^\pi(K_t) \rightarrow \hat{F}_{t+1} = F(x_t, W_{t+1}) \rightarrow K_{t+1}
\]

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

\[
S_{t+1} = (p_{t+1}, K_{t+1})
\]

\[
(S_0 = (p_0, K_0), x_0 = X^\pi(S_0), W_1 = (\hat{p}_1, \hat{F}_1), S_1 = (p_1, K_1), x_1 = X^\pi(S_1), ...)
\]
Problem classes

- State-dependent problems
  - Dynamic resource process – Excess inventory held over:
    \[ R_t = \text{Inventory available at time } t \]
  - Optimization problem is now
    \[ \max_{0 \leq x \leq R_t} \mathbb{E} F(x, W) = \mathbb{E} \left\{ p_t \min(x, \hat{R}) - cx \right\} \]
  - State variable is now \( S_t = (R_t, p_t, K_t) \)
  - Transitions:
    \[
    \begin{align*}
    x_t &= X^\pi(K_t) \\
    \hat{F}_{t+1} &= F(x_t, W_{t+1}) \\
    R_{t+1} &= \max(0, R_t + x_t - \hat{R}_{t+1}) \\
    p_{t+1} &= p_t + \hat{p}_{t+1} \\
    S_{t+1} &= (R_{t+1}, p_{t+1}, K_{t+1})
    \end{align*}
    \]
\( (S_0 = (R_0, p_0, K_0), x_0 = X^\pi(S_0), W_1 = (\hat{R}_1, \hat{p}_1, \hat{F}_1), S_1 = (R_1, p_1, K_1), x_1 = X^\pi(S_1), ...) \)
Problem classes

- **Offline (final reward)**
  - We can iteratively search for the best solution.
  - We only care about the final solution.
  - Asymptotic formulation:
    \[
    \max_x \mathbb{E} F(x, W)
    \]
  - Finite horizon formulation:
    \[
    \max_\pi \mathbb{E} F(x^{\pi,N}, W)
    \]

- **Online (cumulative reward)**
  - We have to learn as we go
  - \[
  \max_\pi \mathbb{E} \sum_{n=0}^{N-1} F(X^{\pi}(S^n), W^{n+1})
  \]
Problem classes

Our most general formulation that covers all of these problems is

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

$$\max_\pi E^\pi \left\{ F(X_T^\pi, W) | S_0 \right\}$$

where

$$S_{t+1} = S^M(S_t, X_t^\pi(S_t), W_{t+1})$$

So, how do we design policies?
Outline

- Canonical problems
- Problem classes
- Solution strategies for learning problems
- Elements of a dynamic model
- An energy storage illustration
- Modeling uncertainty
- Designing policies
- The four classes of policies
- From deterministic to stochastic optimization
Solution strategies

- Special structure
  - Where expectations can be computed, turning the problem into a deterministic problem
  - Implies we can solve $\max_x \mathbb{E}F(x,W)$ exactly as a deterministic problem (this is what we are doing when we use Bellman’s equation).

- Sampled problems (SAA, scenario trees)
  - Uncontrolled sampling
  - Controlled sampling

- Adaptive learning algorithms
  - Works with full probability space
  - This is our focus.
Solution strategies

Sampled problems

» Sample average approximation (stochastic optimization)
\[
\min_x \frac{1}{N} \sum_{n=1}^{N} F(x, W^n)
\]

» Probabilistic learning of a sampled model
\[
\min_x \sum_{k=1}^{K} F(x, \theta_k) p^n_k
\]

» Statistical learning
\[
\min_{\theta} \frac{1}{N} \sum_{n=1}^{N} \left( y_n - f(x_n | \theta) \right)^2
\]
Solution strategies

Sampling

» Uncontrolled – This is what is implied when we are given the data.
  • Batch dataset (“big data”)
  • No control over the arrival process:
    – e.g. Patients arriving to a hospital

» Direct control
  • Creating samples that accurately represent the underlying stochastic process
    – Quantization/epi-splines
    – Voronoi quantization
    – K-L divergence (more generally phi-divergence)

» Indirect control
  • Decision influences distribution (e.g. selling stock influences price)
Solution strategies

- Adaptive learning algorithms
  - We seek methods that are trying to solve the original problem (not the sampled approximation)
  - We are interested in:
    - Asymptotic optimality
    - Rapid finite-time convergence

- Strategies:
  - Derivative-based stochastic search
    - Asymptotic analysis (Robbins-Monro etc)
    - Finite-time analysis
  - Derivative-free stochastic search
    - Requires iterative learning
Solution strategies

Derivative-based stochastic search – infinite horizon

» Stochastic gradient algorithm:

\[
\max_x \mathbb{E} F(x, W) = \mathbb{E} \left( p \min(x, W) - cx \right)
\]

\[
x^{n+1} = x^n + \alpha_n \nabla_x F(x^n, W^{n+1})
\]

\[
\nabla_x F(x^n, W^{n+1}) = \begin{cases} 
p - c & x < W \\
-c & x \geq W
\end{cases}
\]

\[
\lim_{n \to \infty} \mathbb{E} F(x^n, W) \to \mathbb{E} F(x^*, W)
\]

» Asymptotic analysis produces a deterministic solution \(x^*\)
  • This is deterministic thinking on a stochastic problem

» What happens when we want the best solution in \(N\) iterations?
Solution strategies

Derivative-based stochastic search – finite horizon

» We want a method (an algorithm) that produces the best solution by time \( N \):

\[
x^{n+1} = x^n + \alpha_n \nabla_x F(x^n, W^{n+1})
\]

The “transition function”

The “state”

\[ x^{n+1} = x^n + \alpha_n \nabla_x F(x^n, W^{n+1}) \]

The “policy”

\[
\alpha_n = \frac{\theta}{\theta + N^n}
\]

where \( N^n = \) number of times the solution has not improved.

» Assume that our stepsize rule is

After \( n \) iterations, our “state” is

\[
S^n = (x^n, N^n)
\]

\[
S^{n+1} = S^M (S^n, \alpha_n, W^{n+1})
\]

Given the state \( S^n \) and the parameter \( \theta \), we can determine (after sampling \( W^{n+1} \)) the next state \( S^{n+1} \).
Solution strategies

Testing different stepsize rules (“policies”)

- We want to optimize the rate of convergence:
  - Different stepsize rules
  - Different ways of computing the gradient
Solution strategies

Derivative-based stochastic search – finite horizon

» If $X^\pi(S^n | \theta)$ is our algorithm (policy), we follow a sample path $W^1, \ldots, W^N$ to obtain a final solution $x^{\pi,N}$, which is a random variable.

» Our optimization problem is to find the best policy (algorithm) $X^\pi(S^n | \theta)$, which requires taking an expectation over the samples $W^1, \ldots, W^N$

$$\max_{\pi} \mathbb{E}_\pi \mathbb{E}_{W | x^{\pi,N}} \left\{ F(x^{\pi,N}, W) \mid x^{\pi,N} \right\}$$
Solution strategies

Derivative-free stochastic search

» Start by assuming that our set of possible decisions is finite:

\[ x \in X = \{x_1, x_2, \ldots, x_M\} \]

» Assume we have some belief about our function (say, lookup table). Using a Bayesian model, we assume we have a distribution of belief about \( f(x) = \mathbb{E}F(x, W) \) given by

\[ \mathbb{E}F(x, W) = \mu_x \sim N(\mu^0, \beta^0) \]

where \( \beta^0 \) is the precision where \( \beta^0 = 1/\sigma^{2,0} \).

» We refer to \( S^0 = K^0 = N(\mu^0, \beta^0) \) as our prior state of knowledge.
Derivative-free, finite horizon

Belief state for ranking and selection

» S is our “state of knowledge”

\[ S_n = \mathcal{N}(\mu_5^n, \sigma_5^n) \]

\[ S^n = (S_1^n, \ldots, S_5^n) \]
Derivative-free, finite horizon

● Updating beliefs

» After \( n \) experiments, our belief is \( \mu_x \sim N(\mu_x^n, \beta_x^n) \)

» Assume that based on this belief, we choose 
\[
x = x^n = X^\pi (S^n)
\]
to run for our next experiment (experiment \( n+1 \)):

\[
W_{x^n}^{n+1} = \mu_{x^n} + \varepsilon_{n+1}^n
\]

» We update our beliefs using

\[
\begin{align*}
\mu_x^{n+1} &= \frac{\beta^n \mu_x^n + \beta^W W_x^{n+1}}{\beta^n + \beta^W} \\
\beta_x^{n+1} &= \beta_x^n + \beta^W
\end{align*}
\]

Transition function:
\[
S^{n+1} = S^M (S^n, x^n, W^{n+1})
\]
Derivative-free, finite horizon

Designing a policy

» We need a rule for picking which decision to try next. We call this rule our policy $X^\pi (S^n | \theta)$. Some examples are:

- Interval estimation:
  
  \[ X^{IE} (S^n, \theta^{IE}) = \arg \max_x \left( \mu_x^n + \theta^{IE} \sigma_x^n \right) \quad \sigma_x^n = \text{Std. dev. of } \mu_x^n \]

- Upper confidence bounding
  
  \[ X^{UCB} (S^n, \theta^{UCB}) = \arg \max_x \left( \mu_x^n + \theta^{UCB} \sqrt{\frac{\log n}{N_x^n}} \right) \quad N_x^n = \text{No. of times } x \text{ is tested.} \]

- Thompson sampling:
  
  \[ X^{TS} (S^n) = \arg \max_x \hat{\mu}_x^n \quad \hat{\mu}_x^n \sim N(\mu_x^n, \beta_x^n) \]

- Knowledge gradient (expected value of information):
  
  \[ \nu^{KG,n} (x) = E \left\{ \max_y F (y, K^{n+1} (x)) \right\} - \max_y F (y, K^n) \]
Testing policies

» We have three sources of randomness:
  • The true function $\mu_x \sim N(\mu^0, \beta^0)$ (Bayesian belief model)
  • The samples $W^1, \ldots, W^N$ generated from truth $\mu_x$

$$W_{x^n}^{n+1} = \mu_{x^n} + \epsilon^{n+1}$$

while following policy $\pi$
  • Finally, the uncertainty $W$ needed to evaluate the final design $x^{\pi,N}$

» We choose our policy by solving:

$$\max_{\pi} \mathbb{E}_{\mu} \mathbb{E}_W \mathbb{E}_{\pi} \left\{ F(x^{\pi,N}, W) \mid x^{\pi,N} \right\}$$

$$\max_{\pi} \mathbb{E}_W F(x^{\pi,N}, W)$$
Outline

- Canonical problems
- Problem classes
- Solution strategies for learning problems
- Elements of a dynamic model
- An energy storage illustration
- Modeling uncertainty
- Designing policies
- The four classes of policies
- From deterministic to stochastic optimization
We lack a standard language for modeling sequential, stochastic decision problems. In the slides that follow, we propose to model problems along five fundamental dimensions:

- State variables
- Decision variables
- Exogenous information
- Transition function
- Objective function

This framework draws heavily from Markov decision processes and the control theory communities, but it is not the standard form used anywhere.
Modeling dynamic problems

- The system state:
  - Controls community

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

- Operations research/MDP/Computer science

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

\[ R_t = \text{Resource state (physical state)} \]
- Location/status of truck/train/plane
- Energy in storage

\[ I_t = \text{Information state} \]
- Prices
- Weather

\[ K_t = \text{Knowledge state ("belief state")} \]
- Belief about traffic delays
- Belief about the status of equipment
The state variable

- Classes of state variables

- Resource/physical state
- Information state
- Knowledge/belief state
The state variables

What is a state variable?

» Bellman’s classic text on dynamic programming (1957) describes the state variable with:
  • “… we have a physical system characterized at any stage by a small set of parameters, the state variables.”

» The most popular book on dynamic programming (Puterman, 2005, p.18) “defines” a state variable with the following sentence:
  • “At each decision epoch, the system occupies a state.”

» Wikipedia:
  • “State commonly refers to either the present condition of a system or entity” or….
  • A state variable is one of the set of variables that are used to describe the mathematical ‘state’ of a dynamical system
The state variable

A proposed definition of a state variable:

Definition 9.3.1 A state variable is:

a) The minimally dimensioned function of history that, combined with the exogenous information (and a policy) is necessary and sufficient to compute the decision function (the policy), the transition function, and the cost/contribution function.

b) The minimally dimensioned function of history that is necessary and sufficient to compute the cost/contribution function, the constraints, and the transition function.

» The first depends on a policy. The second depends only on the problem.

» Using either definition, all properly modeled problems are Markovian!
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
    \[ x \in X = \{1, 2, ..., M\} \]
  - Continuous scalar
    \[ x \in X = [a, b] \]
  - Continuous vector
    \[ x = (x_1, ..., x_K), \quad x_k \in \mathbb{R} \]
  - Discrete vector
    \[ x = (x_1, ..., x_K), \quad x_k \in \mathbb{Z} \]
  - Categorical
    \[ x = (a_1, ..., a_I), \quad a_i \text{ is a category (e.g. red/green/blue)} \]
Modeling dynamic problems

- Exogenous information:

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

\[ \hat{R}_t = \text{Equipment failures, delays, new arrivals} \]
New drivers being hired to the network

\[ \hat{D}_t = \text{New customer demands} \]

\[ \hat{p}_t = \text{Changes in prices} \]

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

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

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

$$R_{t+1} = R_t + x_t + \hat{R}_{t+1} \quad \text{Inventories}$$

$$p_{t+1} = p_t + \hat{p}_{t+1} \quad \text{Spot prices}$$

$$D_{t+1} = D_t + \hat{D}_{t+1} \quad \text{Market demands}$$

Also known as the:
- “System model”
- “State transition model”
- “Plant model”
- “Plant equation”
- “Transition law”

“Transfer function”
“Transformation function”
“Law of motion”
“Model”

For many applications, these equations are unknown. This is known as “model-free” dynamic programming.
Modeling dynamic problems

The objective function

Dimensions of objective functions

» Performance metrics
» Properties (convexity, monotonicity, continuity, unimodularity, …)
» Final reward vs. cumulative reward
» Time to compute (fractions of seconds to minutes, to hours, to days or months)
» Expectation or risk measures
Objective functions

Performance metrics

» Costs, profits, revenues, contributions (business)
» Gains, losses (engineering)
» Strength, conductivity, diffusivity (materials science)
» Tolerance, toxicity, effectiveness (health)
» Stability, robustness (engineering)
» Risk, volatility (finance)
» Utility (economics)
Objective functions

» Deterministic costs

\[ c^T x = \text{Deterministic linear costs} \]

» State-independent

\[ F(x^n, W^{n+1}) = p \max(x^n, W^{n+1}) - cx^n \]

» State-dependent

\[ C(S_t, x_t) = p_t x_t \quad (p_t \text{ is in the state variable}) \]
\[ C(S_t, x_t, W_{t+1}) = p_t \max(x_t, W_{t+1}) - c_t x_t \]
\[ C(S_t, x_t, S_{t+1}) = -S_t^T QS_t + S_{t+1}^T RS_{t+1} \]
Objective functions

Characteristics of the objective function

» Analytical behavior
  • Concave/convex, unimodal, monotone, smooth,…

» Computational cost:
  • Fractions of a second – Analytical functions
  • Minutes – Computer simulations
  • Hours – Laboratory experiments/computer simulations
  • Days (or longer) – Laboratory/field experiments
  • Weeks to months – Field experiments

» Startup/switching costs
  • What is involved to observe function for different inputs? Is there a cost to switch to different inputs?

» Risk operators
  • Expectations
  • Risk measures
  • Robust/worst case
Modeling stochastic, dynamic problems

Objective functions

» Offline (asymptotic) stochastic search
\[ \max_x \mathbb{E}F(x, W) \]

» Two-stage stochastic programming
\[ \max_{x_0} \left( c_0 x_0 + \mathbb{E}Q(x_0, W_1) \right) \]

» Offline (finite time) stochastic search
\[ \max_{\pi} \mathbb{E}F(X^{\pi, N}, W) \]

» Multiarmed bandit problem
\[ \max_{\pi} \mathbb{E} \sum_{n=0}^{N-1} F(X^{\pi}(S^n), W^{n+1}) \]

» Contextual bandit problem
\[ \max_{\pi} \mathbb{E} \left\{ \sum_{n=0}^{N-1} F(X^{\pi}(S^n), W^{n+1}) \mid S_0 \right\} \]

» Full dynamic programming
\[ \max_{\pi} \mathbb{E} \left\{ \sum_{t=0}^{T} C(S_t, X^{\pi}(S_t)) \mid S_0 \right\} \]

» Offline dynamic programming
\[ \max_{\pi^{\text{learn}}} \mathbb{E} \sum_{t=0}^{T} C(S_t, X^{\pi^{\text{imp}}}(S_t)) \]
Problem classes

- **The circle of stochastic optimization**

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

\[
\max_{\pi_{\text{learn}}} \mathbb{E}\sum_{t=0}^{T} C(S_t, X_t^{\pi_{\text{imp}}}(S_t))
\]

\[
\max_{\pi} \mathbb{E}\left\{ \sum_{t=0}^{T} C(S_t, X_t^\pi(S_t)) \mid S_0 \right\}
\]

\[
\max_{\pi} \mathbb{E}\sum_{n=0}^{N-1} F(X^\pi(S^n), W^{n+1})
\]

\[
\max_x \left( c_0 x_0 + \mathbb{E} Q(x_0, W_1) \right)
\]

\[
\max_{\pi} \mathbb{E} F(X^\pi, W)
\]

\[
\max_{\pi} \mathbb{E}\sum_{n=0}^{N-1} F(X^\pi(S^n), W^{n+1})
\]
Modeling stochastic, dynamic problems

The universal objective function

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

- Expectation over all random outcomes
- Contribution function
- Decision function (policy)
- State variable
- Initial state variable
- New information

Finding the best policy

Given a system model (transition function)

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

Now we just have to find the best policy.
## Problem classes

### Major problem classes

<table>
<thead>
<tr>
<th>State independent</th>
<th>Offline (terminal reward)</th>
<th>Online (cumulative reward)</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \max_x \mathbb{E}F(x, W) )</td>
<td>( \max_\pi \mathbb{E}F(X^{\pi,N}, W) )</td>
<td>( \max_\pi \mathbb{E}\left{ \sum_{n=0}^{N-1} F(X^{\pi}(S^n), W^{n+1}) \mid S_0 \right} )</td>
</tr>
<tr>
<td>( \max_\pi \mathbb{E}F(X^{\pi,N}, W) )</td>
<td>“stochastic search”</td>
<td>“multiarmed bandit problems”</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>State dependent</th>
<th>( \max_{\pi_{\text{learn}}} \mathbb{E} \sum_{t=0}^{T-1} C(S_t, X_t^{\pi_{\text{imp}}} (S_t)) )</th>
<th>( \max_\pi \mathbb{E}\left{ \sum_{t=0}^{T-1} C(S_t, X_t^{\pi}(S_t)) \mid S_0 \right} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( S_{t+1} = S^M (S_t, X_t^{\pi_{\text{imp}}} (S_t), W_{t+1}) )</td>
<td>“dynamic programming”</td>
<td>( S_{t+1} = S^M (S_t, X_t^{\pi}(S_t), W_{t+1}) )</td>
</tr>
</tbody>
</table>

- **Stochastic search**
- **Multiarmed bandit problems**
- **Dynamic programming**
Outline

- Canonical problems
- Problem classes
- Solution strategies for learning problems
- Elements of a dynamic model
- An energy storage illustration
- Modeling uncertainty
- Designing policies
- The four classes of policies
- From deterministic to stochastic optimization
An energy storage problem

Consider a basic energy storage problem:

- We are going to show that with minor variations in the characteristics of this problem, we can make each class of policy work best.
An energy storage problem

A model of our problem

» State variables

» Decision variables

» Exogenous information

» Transition function

» Objective function
An energy storage problem

- State variables

- We will present the full model, accumulating the information we need in the state variable.
- We will highlight information we need as we proceed. This information will make up our state variable.
An energy storage problem

Decision variables

\[ x_t = \left( x_t^{EL}, x_t^{EB}, x_t^{GL}, x_t^{GB}, x_t^{BL}, \right) \]

» Constraints;

\[ x_t^{EL} + x_t^{EB} \leq E_t, \]

\[ (x_t^{GL} + x_t^{EL} + x_t^{BL}) = L_t, \]

\[ x_t^{BL} \leq R_t, \]
An energy storage problem

Exogenous information

\[ W_t = \begin{cases} 
\hat{E}_t &= \text{Change in energy from wind between } t-1 \text{ and } t \\
\epsilon^p_t &= \text{Noise in the price process between } t-1 \text{ and } t \\
\hat{D}_t &= \text{Change in load between } t-1 \text{ and } t \\
\hat{f}^L_{tt'} &= \text{Forecast of load } D^\text{load}_{tt'} \text{ provided by vendor at time } t \\
\hat{f}^L_t &= \left( \hat{f}^L_{tt'} \right)_{t' > t} \text{ Provided exogenously}
\end{cases} \]
An energy storage problem

Transition function

\[ E_{t+1} = E_t + \hat{E}_{t+1} \]
\[ p_{t+1} = \theta_0 p_t + \theta_1 p_{t-1} + \theta_2 p_{t-2} + \varepsilon_{t+1} \]
\[ D_{t+1} = D_t + \hat{D}_{t+1} \]
\[ R_{t+1}^{\text{battery}} = R_t^{\text{battery}} + x_t \]
An energy storage problem

Objective function

\[ C(S_t, x_t) = p_t(x_t^{GB} + x_t^{GL}) \]

\[
\min_{\pi} \mathbb{E} \left\{ \sum_{t=0}^{T} C(S_t, X_t^\pi(S_t)) \mid S_0 \right\}
\]

Expectation depends on forecasts \( f_t \).
An energy storage problem

- **State variables**
  - Cost function: \( p_t \) = Price of electricity
  - Decision function

- **Constraints:**
  - Transition function:
    \[
    x_t^{EL} + x_t^{EB} \leq E_t,
    \]
    \[
    (x_t^{GL} + x_t^{EL} + x_t^{BL}) = L_t,
    \]
    \[
    x_t^{BL} \leq R_t
    \]

- **Needed to compute probability model:** \( f_t^L \)

- **Transition function:**
  \[
  p_{t+1} = \theta_0 p_t + \theta_1 p_{t-1} + \theta_2 p_{t-2} + \varepsilon_{t+1}^p
  \]
There is a common misunderstanding that state variables have to be simple (they don’t).

There is also a tendency to refer to problems that depend on prior information (such as $p_{t-1}$ and $p_{t-2}$) as “history dependent.” But this is information known at time $t$ (who cares when it first became known).

All properly modeled problems are Markovian!

Understanding state variables is very important in dynamic systems, because it forces you to understand what you know at time $t$, and what you don’t.
Outline

- Canonical problems
- Problem classes
- Solution strategies for learning problems
- Elements of a dynamic model
- An energy storage illustration
- Modeling uncertainty
- Designing policies
- The four classes of policies
- From deterministic to stochastic optimization
Modeling uncertainty

There are two information processes that drive the system:

» Decisions \( x_t \) – This is the *endogenously* controllable information process.

» Exogenous information - This comes from the initial state \( S_0 \), and the exogenous information process \( W_t \).

To figure out how to make good decisions, you need:

» The system model \( S_{t+1} = S^M (S_t, x_t, W_{t+1}) \)

» The initial state \( S_0 \) and the exogenous information process \( W_t \).

» The contribution function \( C(S_t, x_t, W_{t+1}) \)
Modeling uncertainty

The initial state $S_0$. This contains:

- All deterministic parameters needed by the system. This is static data, so it is not modeled as part of the dynamic state $S_t$, $t > 0$.

- “State of knowledge” – probabilistic information about uncertain parameters. This information is always represented as a probability distribution of some form.
Modeling uncertainty

The exogenous information process $W_t$ which might include:

» Passive information – This is information that arrives regardless of any actions we may take. Examples:
  • Purely exogenous – Information that is not influenced by the state of the system or any actions we take. Examples:
    – Rainfall, stock prices (if we are a small player).
  • Exogenous distributions may influenced by states and/or actions (stock prices if we are a large player).

» Active information – This is information we choose to collect
  • Running a laboratory experiment
  • Purchasing a report
Modeling uncertainty

Types of uncertainty

- Observational uncertainty – Errors in our observations of the state of the system:
  - What is the CO2 content of the atmosphere?
  - What is inventory of oil in the U.S.?

- Prognostic uncertainty – Uncertainty in the forecast of a future event.
  - Forecasting demands
  - Forecasting the weather
Modeling uncertainty

Types of uncertainty

» Experimental noise – This is the variability that arises when running repeated experiments (either in a lab or in the field)
  • Testing the impact of a new flu drug.
  • Testing the effect of a new material on battery lifetimes

» Transitional uncertainty – We have a model of how a (presumably) deterministic system evolves, but there is still noise:

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

  • Modeling the location of an aircraft moving at a certain speed from a known location.
  • Predicting the time of arrival of a car at a downstream node
Modeling uncertainty

Types of uncertainty

» Inferential uncertainty
  • Uncertainty in parameters estimated from observational data
  • Sometimes known as *diagnostic uncertainty* which might arise in the context of estimating a condition such as disease or the reason for a malfunction (in an engine). Such an assessment would an inference based on indirect observations.

» Model uncertainty – This is uncertainty about the model itself, which comes in two forms:
  • Uncertainty about the structure of the model:
    – Linear approximation of a nonlinear model
    – Different sets of equations describing the climate
  • Parameters characterizing the model
Modeling uncertainty

Types of uncertainty

» Systematic exogenous uncertainty - Errors in the model of exogenous information that occur on long time scales:
  - Modeling the effect of long-term drops in oil consumption due to conservation
  - Modeling the effect of increased cloud cover due to climate change

\[ W_t = \mu_t + \epsilon_t + \psi_t \]

- Base signal (forecasted)
- High frequency noise
- Low frequency noise ("scenarios")
Types of uncertainty

» Control uncertainty
  • You ask for \( x_t \) but you get \( x_t + \varepsilon_t \)
  • Wiley sets a wholesale price of $80, but Amazon sells at some random price above that (limits Wiley’s ability to set prices).

» Algorithmic uncertainty
  • Run the same algorithm twice, and you may get different answers (depends on the algorithm and the nature of the compute environment).
Modeling uncertainty

Bayesian vs. frequentist uncertainty

» Bayesian uncertainty is captured by a distribution of belief derived from prior information:
  • Expert judgment
  • Information collected from different settings
  • Past experience
  • Bayesian uncertainty is always communicated through $S_0$

» Frequentist uncertainty
  • This is uncertainty derived from statistical analysis of the variability inherent in the exogenous information $W_t$
Types of distributions

» Probability distributions come in different forms:
  • Classical “thin tailed” distributions –
    – Exponential family
      » Normal, exponential, gamma
      » Uniform
    – Discrete variants
  • Heavy-tailed distributions
    – Cauchy distribution (may have infinite variance)
    – “Jump diffusion” – Sum of low-variance normally distributed error, plus a high-variance error that occurs with low probability
  • Spikes
  • Bursts
  • Rare events
Modeling uncertainty

» How can you claim you have an optimal policy if you have not modeled the problem properly?

» Uncertainty is easily the most subtle and overlooked aspect of modeling.

» It is not enough to include uncertainty – you have to capture uncertainty in a way that represents reality. This issue universally pervades applications of stochastic optimization to real problems.

© 2016 W.B. Powell
Outline

- Canonical problems
- Problem classes
- Solution strategies for learning problems
- Elements of a dynamic model
- An energy storage illustration
- Modeling uncertainty
- Designing policies
- The four classes of policies
- From deterministic to stochastic optimization
Designing policies

We have to start by describing what we mean by a policy.

» Definition:

A policy is a mapping from a state to an action.
... any mapping.

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

“Policies” and the English language

<table>
<thead>
<tr>
<th>Behavior</th>
<th>Manner</th>
<th>Ritual</th>
</tr>
</thead>
<tbody>
<tr>
<td>Belief</td>
<td>Method</td>
<td>Rule</td>
</tr>
<tr>
<td>Bias</td>
<td>Mode</td>
<td>Style</td>
</tr>
<tr>
<td>Commandment</td>
<td>Mores</td>
<td>Technique</td>
</tr>
<tr>
<td>Conduct</td>
<td>Patterns</td>
<td>Tenet</td>
</tr>
<tr>
<td>Convention</td>
<td>Plans</td>
<td>Tradition</td>
</tr>
<tr>
<td>Culture</td>
<td>Policies</td>
<td>Way of life</td>
</tr>
<tr>
<td>Customs</td>
<td>Practice</td>
<td></td>
</tr>
<tr>
<td>Dogma</td>
<td>Prejudice</td>
<td></td>
</tr>
<tr>
<td>Etiquette</td>
<td>Principle</td>
<td></td>
</tr>
<tr>
<td>Fashion</td>
<td>Procedure</td>
<td></td>
</tr>
<tr>
<td>Formula</td>
<td>Process</td>
<td></td>
</tr>
<tr>
<td>Habit</td>
<td>Protocols</td>
<td></td>
</tr>
<tr>
<td>Laws/bylaws</td>
<td>Recipe</td>
<td></td>
</tr>
</tbody>
</table>
Designing policies

Two fundamental strategies:

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

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

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

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

Policy search:

1a) Analytical functions that directly map states to actions ("policy function approximations" or PFAs) $x_t = X^{PFA}(S_t | \theta)$
   - Lookup tables
     - "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) Maximizing analytical approximations of costs and/or constraints ("cost function approximations" or CFAs)
   - Optimizing a deterministic model modified to handle uncertainty (buffer stocks, schedule slack)
   \[
   X^{CFA}(S_t | \theta) = \arg \max_{x_t \in \mathcal{X}_t^\pi(\theta)} \bar{C}^\pi(S_t, x_t | \theta)
   \]
Designing policies

- **Policy search:**
  » Typically involves searching within a parameterized family…
  » … but may involve comparisons across classes of functional approximations.
  » May be done offline (terminal reward) or online (cumulative reward).

- **Styles**
  » Active learning – Experiment with new policies with the hope of finding improvements, but risks spending time using less effective policies.
  » Passive learning – Using the policy you believe is best, do updating based on samples that work well.
Designing policies

- Lookahead approximations – Approximate the impact of a decision now on the future:
  - An optimal policy (based on looking ahead):
    \[ X_t^* (S_t) = \arg \max_{x_t} \left( C(S_t, x_t) + \mathbb{E} \max_{\pi \in \Pi} \left\{ \mathbb{E} \sum_{t'=t+1}^{T} C(S_{t'}, X_{t'}^\pi (S_{t'})) \mid S_{t+1} \right\} \mid S_t, x_t \right) \]
Designing policies

Policies based on value function approximations

- This is the foundation of all solution strategies that depend on Bellman (or Hamilton-Jacobi) optimality equations.

- Exact value functions are rare:
  - Discrete states and actions, with a computable one-step transition matrix.
  - Analytical solutions for special functions (e.g. LQR)

- Approximate value functions are generally based on:
  - Approximate value iteration
  - Approximate policy iteration
Designing policies

The ultimate lookahead policy is optimal

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

Maximization that we cannot compute

Expectations that we cannot compute
Designing policies

- The ultimate lookahead policy is optimal

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

- Instead, we have to solve an approximation called the lookahead model:

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

» 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

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

2) **Cost function approximation (CFAs)**
   - \[ X^{CFA}(S_t | \theta) = \arg \max_{x_t \in \bar{X}_t(\theta)} \bar{C}^\pi(S_t, x_t | \theta) \]

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

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

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

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

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

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

» 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
  • Hierarchical aggregation
Designing policies

Finding the best policy

» We have to first articulate our classes of policies

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

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

» 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

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

or

\[
\max_\pi \mathbb{E} C\left(S_T, X^\pi_T\right) = \mathbb{E} F\left(X^\pi_T, W\right)
\]
Designing policies

Notes:

» Three of the four classes of policies involve some form of function approximation (PFA, CFA, VFA)

» Lookahead models require approximating the lookahead model, which requires (among other approximations) replacing the full probability space with a sampled approximation (can be hard to solve).

» Searching for the best parameterized policy is just like solving a stochastic search problem:
  • May be derivative-based or derivative-free.
  • May be solved offline or online.

» VFAs have to be estimated using biased observations.
Outline

- Canonical problems
- Problem classes
- Solution strategies for learning problems
- Elements of a dynamic model
- An energy storage illustration
- Modeling uncertainty
- Designing policies
- The four classes of policies
- From deterministic to stochastic optimization
Outline

The four classes of policies

» Policy function approximations (PFAs)
» Cost function approximations (CFAs)
» Value function approximations (VFAs)
» Direct lookahead policies (DLAs)
Outline

The four classes of policies

» Policy function approximations (PFAs)
» Cost function approximations (CFAs)
» Value function approximations (VFAs)
» Direct lookahead policies (DLAs)
Policy function approximations

Battery arbitrage – When to charge, when to discharge, given volatile LMPs

[Graph showing ERCOT (Texas) price data with average price around $50/megawatt-hour]

[Images of power lines and a battery storage facility]
Grid operators require that batteries bid charge and discharge prices, an hour in advance.

We have to search for the best values for the policy parameters $\theta^{\text{Charge}}$ and $\theta^{\text{Discharge}}$. 
Policy function approximations

- Our policy function might be the parametric model (this is nonlinear in the parameters):

\[
X^\pi (S_t | \theta) = \begin{cases} 
+1 & \text{if } p_t < \theta^{\text{charge}} \\
0 & \text{if } \theta^{\text{charge}} < p_t < \theta^{\text{discharge}} \\
-1 & \text{if } p_t > \theta^{\text{charge}} 
\end{cases}
\]

Energy in storage:

Price of electricity:
Policy function approximations

Finding the best policy

» We need to maximize

$$\max_{\theta} F(\theta) = \mathbb{E} \sum_{t=0}^{T} \gamma^t C(S_t, X_t^\pi(S_t | \theta))$$

» We cannot compute the expectation, so we run simulations:
Outline

The four classes of policies

» Policy function approximations (PFAs)
» Cost function approximations (CFAs)
» Value function approximations (VFAs)
» Direct lookahead policies (DLAs)
Robust cost function approximation

Inventory management

» How much product should I order to anticipate future demands?

» Need to accommodate different sources of uncertainty.
  • Market behavior
  • Transit times
  • Supplier uncertainty
  • Product quality
Robust cost function approximations

Imagine that we want to purchase parts from different suppliers. Let $x_{tp}$ be the amount of product we purchase at time $t$ from supplier $p$ to meet forecasted demand $D_t$. We would solve

$$X^\pi_t(S_t) = \arg \min_{x_t} \sum_{p \in P} c_p x_{tp}$$

subject to

$$\sum_{p \in P} x_{tp} \geq D_t$$

$$x_{tp} \leq u_p$$

$$x_{tp} \geq 0$$

» This assumes our demand forecast $D_t$ is accurate.
Imagine that we want to purchase parts from different suppliers. Let $x_{tp}$ be the amount of product we purchase at time $t$ from supplier $p$ to meet forecasted demand $D_t$. We would solve

$$X_t^{\pi}(S_t \mid \theta) = \arg \min_{x_t \in \mathcal{X}^{\pi}(\theta)} \sum_{p \in P} c_p x_{tp}$$

subject to

$$\sum_{p \in P} x_{tp} \geq \theta^{\text{Reserve}} D_t$$
$$x_{tp} \leq u_p$$
$$x_{tp} \geq \theta^{\text{buffer}}$$

This is a “parametric cost function approximation”
Cost function approximations

A general way of creating CFAs:

» Define our policy:

$$X^\pi_t (\theta) = \arg\min_x \left( C(S_t, x_t) + \sum_{f \in F} \theta_f \phi_f (S_t, x_t) \right)$$

Cost correction term.

» This has been confused with approximate dynamic programming, but the correction term is not a value function.
An even more general CFA model:

- Define our policy:

$$X_t^\pi(\theta) = \arg\min_x \tilde{C}^\pi(S_t, x_t | \theta)$$

subject to

$$Ax = \tilde{b}^\pi(\theta)$$

- We tune $\theta$ by optimizing:

$$\min_\theta F^\pi(\theta) = \mathbb{E}\sum_{t=0}^{T} C(S_t, X_t^\pi(\theta))$$
Outline

The four classes of policies

» Policy function approximations (PFAs)
» Cost function approximations (CFAs)
» Value function approximations (VFAs)
» Direct lookahead policies (DLAs)
Value function approximation

Pre-decision state: we see the demands

$$S_t = \left( \begin{array}{c} TX \\ t \end{array} \right), \hat{D}_t$$
We use initial value function approximations…

\[ S_t = \left( \frac{TX}{t}, \hat{D}_t \right) \]

\[ \bar{V}^0(MN) = 0 \]

\[ \bar{V}^0(CO) = 0 \]

\[ \bar{V}^0(CA) = 0 \]

\[ \bar{V}^0(NY) = 0 \]

\[ \text{Value function approximation} \]

\[ \hat{V}(t,t) \]
Value function approximation

... and make our first choice: $x^1$

$$S^x_t = \begin{pmatrix} NY \\ t+1 \end{pmatrix}$$
Value function approximation

Update the value of being in Texas.

$$V^0(CO) = 0$$

$$V^0(MN) = 0$$

$$V^0(NY) = 0$$

$$V^1(TX) = 450$$

$$S_t^x = \left( \begin{array}{c} NY \\ t+1 \end{array} \right)$$
Now move to the next state, sample new demands and make a new decision.

\[ S_{t+1} = \left( \frac{NY}{t+1}, \hat{D}_{t+1} \right) \]
Value function approximation

Update value of being in NY

\[ V^0(MN) = 0 \]
\[ V^0(CO) = 0 \]
\[ V^0(CA) = 0 \]
\[ V^0(NY) = 600 \]
\[ V^1(TX) = 450 \]

\[ S_{t+1}^x = \begin{pmatrix} CA \\ t + 2 \end{pmatrix} \]
Value function approximation

- Move to California.

\[ S_{t+2} = \left( \frac{CA}{t + 2}, \hat{D}_{t+2} \right) \]
Value function approximation

Make decision to return to TX and update value of being in CA

\[ S_{t+2} = \left( \frac{CA}{t+2}, \hat{D}_{t+2} \right) \]
Value function approximation

Updating the value function:

Old value:
\[ \tilde{V}^1(TX) = $450 \]

New estimate:
\[ \hat{v}^2(TX) = $800 \]

How do we merge old with new?
\[
\tilde{V}^2(TX) = (1 - \alpha)\tilde{V}^1(TX) + (\alpha)\hat{v}^2(TX)
\]
\[
= (0.90)\tilde{V}^1(TX) + (0.10)\hat{v}^2(TX)
\]
\[
= (0.90)\tilde{V}^1(TX) + (0.10)\hat{v}^2(TX)
\]
\[
= (0.90)$450 + (0.10)$800
\]
\[
= $485
\]
Value function approximation

- An updated value of being in TX

$$S_{t+3} = \left( \frac{TX}{t+3}, \hat{D}_{t+3} \right)$$
Approximate value iteration

Step 1: Start with a pre-decision state $S^n_t$

Step 2: Solve the deterministic optimization using an approximate value function:

$$\hat{v}_t^n = \min_x \left( C_t(S^n_t, x_t) + \bar{V}^{n-1}_t(S^{M,x}(S^n_t, x_t)) \right)$$

to obtain $x^n$.

Step 3: Update the value function approximation

$$\bar{V}^{n}_t(S^{x,n}_{t-1}) = (1 - \alpha_{n-1})\bar{V}^{n-1}_{t-1}(S^{x,n}_{t-1}) + \alpha_{n-1}\hat{v}_t^n$$

Step 4: Obtain Monte Carlo sample of $W_t(\omega^n)$ and compute the next pre-decision state:

$$S^n_{t+1} = S^M(S^n_t, x^n_t, W_{t+1}(\omega^n))$$

Step 5: Return to step 1.

“on policy learning”
Approximate value iteration

Step 1: Start with a pre-decision state $S^n_t$

Step 2: Solve the deterministic optimization using an approximate value function:

$$\hat{v}_t^n = \min_x \left( C_t(S^n_t, x_t) + \bar{V}_{t-1}^{n-1}(S^{M,x}(S^n_t, x_t)) \right)$$

to obtain $x^n$.

Step 3: Update the value function approximation

$$\bar{V}_{t-1}^n(S^{x,n}_{t-1}) = (1 - \alpha_{n-1}) \bar{V}_{t-1}^{n-1}(S^{x,n}_{t-1}) + \alpha_{n-1} \hat{v}_t^n$$

Step 4: Obtain Monte Carlo sample of $W_t(\omega^n)$ and compute the next pre-decision state:

$$S^n_{t+1} = S^M(S^n_t, x^n_t, W_{t+1}(\omega^n))$$

Step 5: Return to step 1.
Approximate dynamic programming

... a typical performance graph.
Outline

The four classes of policies

» Policy function approximations (PFAs)
» Cost function approximations (CFAs)
» Value function approximations (VFAs)
» Direct lookahead policies (DLAs)
Lookahead policies

Planning your next chess move:

» You put your finger on the piece while you think about moves into the future. This is a lookahead policy, illustrated for a problem with discrete actions.
Lookahead policies

- Decision trees:
Lookahead policies

Modeling lookahead policies

» Lookahead policies solve a lookahead model, which is an approximation of the future.

» It is important to understand the difference between the:

  • Base model – this is the model we are trying to solve by finding the best policy. This is usually some form of simulator.

  • The lookahead model, which is our approximation of the future to help us make better decisions now.

» The base model is typically a simulator, or it might be the real world.
Lookahead policies

- Lookahead models use five classes of approximations:
  - Horizon truncation – Replacing a longer horizon problem with a shorter horizon.
  - Stage aggregation – Replacing multistage problems with two-stage approximation.
  - Outcome aggregation/sampling – Simplifying the exogenous information process.
  - Discretization – Of time, states and decisions.
  - Dimensionality reduction – We may ignore some variables (such as forecasts) in the lookahead model that we capture in the base model (these become latent variables in the lookahead model).
Lookahead policies

Notes:

» The academic literature at the moment does not distinguish between lookahead models and base models.

» When uncertainty is involved, base models are almost always simulators that simulate different policies (which might include a lookahead policy).

» When we use a deterministic lookahead policy to solve a stochastic problem, we understand that the model being solved by the lookahead policy is just an approximation.

» When we use a stochastic lookahead model, then things tend to get confusing.
Lookahead policies

Lookahead policies are the trickiest to model:

» We create “tilde variables” for the lookahead model:
  \[ \tilde{S}_{t,t'} = \text{Approximated state variable (e.g coarse discretization)} \]
  \[ \tilde{x}_{t,t'} = \text{Decision we plan on implementing at time } t' \text{ when we are planning at time } t, \ t' = t, t+1, ..., t+H \]
  \[ \tilde{x}_t = (\tilde{x}_{t,t}, \tilde{x}_{t,t+1}, ..., \tilde{x}_{t,t+H}) \]
  \[ \tilde{W}_{t,t'} = \text{Approximation of information process} \]
  \[ \tilde{c}_{t,t'} = \text{Forecast of costs at time } t' \text{ made at time } t \]
  \[ \tilde{b}_{t,t'} = \text{Forecast of right hand sides for time } t' \text{ made at time } t \]

» All variables are indexed by \( t \) (when the lookahead model is being generated) and \( t' \) (the time within the lookahead model).
Lookahead policies

We can use this notation to create a policy based on our *lookahead model*:

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

- Simplest lookahead is deterministic.

» Simplest lookahead is deterministic.
Lookahead policies

- Deterministic lookahead

\[ X_t^{LA-D}(S_t) = \arg \min_{\tilde{x}_{tt}, \tilde{x}_{t,t+1}, \ldots, \tilde{x}_{t,t+T}} C(\tilde{S}_{tt}, \tilde{x}_{tt}) + \sum_{t'=t+1}^{T} \gamma^{t'-t} C(\tilde{S}_{tt'}, \tilde{x}_{tt'}) \]

- Stochastic lookahead (with two-stage approximation)

\[ X_t^{LA-S}(S_t) = \arg \min_{\tilde{x}_{tt}, \tilde{x}_{t,t+1}, \ldots, \tilde{x}_{t,t+T}} C(\tilde{S}_{tt}, \tilde{x}_{tt}) + \sum_{\tilde{w} \in \tilde{\Omega}_t} p(\tilde{w}) \sum_{t'=t+1}^{T} \gamma^{t'-t} C(\tilde{S}_{tt'}(\tilde{w}), \tilde{x}_{tt'}(\tilde{w})) \]
Lookahead policies

- Lookahead policies peek into the future
  » Optimize over deterministic lookahead model
Lookahead policies

- Lookahead policies peek into the future
  - Optimize over deterministic lookahead model
Lookahead policies

- Lookahead policies peek into the future
  » Optimize over deterministic lookahead model

\[ X_{t+1}^{LA-D}(S_{t+1}) = \arg \min C(\tilde{S}_{t+1}, \tilde{x}_{t+1}) + \sum_{\ell=0}^{\gamma-1} \gamma^{\ell} C(\tilde{S}_{t+1+\ell}, \tilde{x}_{t+1+\ell}) \]
Lookahead policies

- Lookahead policies peek into the future
  » Optimize over deterministic lookahead model

The lookahead model

\[
Y_{t+2}^{\text{LA-D}}(S_{t+2}) = \arg\min C(\bar{S}_{t+2}, \bar{x}_{t+2+2}) + \sum_{i=t+3}^{t+2} y_{t+2} C(\bar{S}_{t+i}, \bar{x}_{t+i+2})
\]
Lookahead policies

There are two strategies for formulating and solving stochastic lookahead models:

» Solve a *sampled model* of the future over a set of scenarios. We have two options:
  • A full multistage tree – Usually impossible to solve
  • A two-stage approximation – We break the problem into:
    – Initial decision
    – See all future information
    – Make all remaining decisions
    – These may still be very hard to solve.

» Use value function approximations:
  • Benders cuts (SDDP)
  • Other strategies for approximating value functions.
Multistage lookahead models

Stochastic lookahead

» Here, we approximate the information model by using a Monte Carlo sample to create a scenario tree:

1am   2am   3am   4am   5am   …..

Change in wind speed

Change in wind speed

Change in wind speed
Multistage lookahead models

We can then simulate this *lookahead policy* over time:

The lookahead model

The base model

\[ \text{The lookahead model} \quad \text{The base model} \]
Multistage lookahead models

- We can then simulate this *lookahead policy* over time:
Multistage lookahead models

- We can then simulate this *lookahead policy* over time:

The lookahead model

The base model
Multistage lookahead models

We can then simulate this lookahead policy over time:

The lookahead model

The base model
Lookahead policies

Notes:

» Solving stochastic lookahead policies can be hard!

» … but this is still just a lookahead policy which is a class of rolling horizon heuristic.

» Even if solving the lookahead model is hard, an optimal solution of a lookahead model (even a stochastic one) is (with rare exceptions) not an optimal policy.
There are two ways of evaluating a policy

» Offline learning using a simulator:
\[
F^\pi (\omega) = \sum_{t=0}^{T-1} C(S_t(\omega), X_t^\pi (S_t(\omega)))
\]

where
\[
S_{t+1}(\omega) = S^M (S_t(\omega), X^\pi (S_t(\omega)), W_{t+1}(\omega))
\]

» Online learning (in the field)
  • Implement the policy and observe how well it works.
  • Make adjustments as necessary.
Lookahead policies

**Offline learning**
- Can test policies fairly quickly
- Can test conditions that have not actually happened.
- Requires making assumptions about dynamics and random events

**Online learning**
- Takes a day to observe a day's performance
- Have to live with real world events
- No assumptions required
Lookahead policies

Notes:

» Lookahead policies are the only class that does not require using any form of statistical function approximation.

» Lookahead policies do not require tuning, but you may want to test parameter settings (horizon, number of scenarios).

» The price of these features tends to be policies that are much harder to compute.

» Approximation strategies can only be evaluated in the controlled setting of a simulator.
An energy storage problem

Consider a basic energy storage problem:

» We are going to show that with minor variations in the characteristics of this problem, we can make each class of policy work best.
An energy storage problem

We can create distinct flavors of this problem:

» Problem class 1 – Best for PFAs
  • Highly stochastic (heavy tailed) electricity prices
  • Stationary data

» Problem class 2 – Best for CFAs
  • Stochastic prices and wind (but not heavy tailed)
  • Stationary data

» Problem class 3 - Best for VFAs
  • Stochastic wind and prices (but not too random)
  • Time varying loads, but inaccurate wind forecasts

» Problem class 4 – Best for deterministic lookaheads
  • Relatively low noise problem with accurate forecasts

» Problem class 5 – A hybrid policy worked best here
  • Stochastic prices and wind, nonstationary data, noisy forecasts.

© 2016 W.B.Powell
An energy storage problem

- The policies

  - The PFA:
    - Charge battery when price is below p1
    - Discharge when price is above p2

  - The CFA
    - Optimize over a horizon H; maintain upper and lower bounds (u, l) for every time period except the first (note that this is a hybrid with a lookahead).

  - The VFA
    - Piecewise linear, concave value function in terms of energy, indexed by time.

  - The lookahead (deterministic)
    - Optimize over a horizon H (only tunable parameter) using forecasts of demand, prices and wind energy

  - The lookahead CFA
    - Use a lookahead policy (deterministic), but with a tunable parameter that improves robustness.
An energy storage problem

- Each policy is best on certain problems
  - Results are percent of posterior optimal solution

<table>
<thead>
<tr>
<th>Problem</th>
<th>Problem description</th>
<th>PFA</th>
<th>CFA Error correction</th>
<th>VFA</th>
<th>Deterministic Lookahead</th>
<th>CFA Lookahead</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>A stationary problem with heavy-tailed prices, relatively low noise, moderately accurate forecasts.</td>
<td>0.959</td>
<td>0.839</td>
<td>0.936</td>
<td>0.887</td>
<td>0.887</td>
</tr>
<tr>
<td>B</td>
<td>A time-dependent problem with daily load patterns, no seasonalities in energy and price, relatively low noise, less accurate forecasts.</td>
<td>0.714</td>
<td>0.752</td>
<td>0.712</td>
<td>0.746</td>
<td>0.746</td>
</tr>
<tr>
<td>C</td>
<td>A time-dependent problem with daily load, energy and price patterns, relatively high noise, forecast errors increase over horizon.</td>
<td>0.865</td>
<td>0.590</td>
<td>0.914</td>
<td>0.886</td>
<td>0.886</td>
</tr>
<tr>
<td>D</td>
<td>A time-dependent problem, relatively low noise, very accurate forecasts.</td>
<td>0.962</td>
<td>0.749</td>
<td>0.971</td>
<td>0.997</td>
<td>0.997</td>
</tr>
<tr>
<td>E</td>
<td>Same as (C), but the forecast errors are stationary over the planning horizon.</td>
<td>0.865</td>
<td>0.590</td>
<td>0.914</td>
<td>0.922</td>
<td>0.934</td>
</tr>
</tbody>
</table>

... any policy might be best depending on the data.

Joint research with Prof. Stephan Meisel, University of Muenster, Germany.
Outline

- Canonical problems
- Problem classes
- Solution strategies for learning problems
- Elements of a dynamic model
- An energy storage illustration
- Modeling uncertainty
- Designing policies
- The four classes of policies
- From deterministic to stochastic optimization
From deterministic to stochastic

Imagine that you would like to solve the time-dependent linear program:

\[
\min_{x_0, \ldots, x_T} \sum_{t=0}^{T} c_t x_t
\]

subject to

\[
A_0 x_0 = b_0
\]

\[
A_t x_t - B_{t-1} x_{t-1} = b_t, \quad t \geq 1.
\]

We can convert this to a proper stochastic model by replacing \( x_t \) with \( X_t^\pi (S_t) \):

\[
\min_{\pi} \sum_{t=0}^{T} c_t X_t^\pi (S_t)
\]

The policy \( X_t^\pi (S_t) \) has to satisfy \( A_t x_t = R_t \) with transition function:

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

**Deterministic**

» Objective function

\[
\min \sum_{t=0}^{T} c_t x_t
\]

» Decision variables:

\[
(x_0, \ldots, x_T)
\]

» Constraints:
- at time \(t\)
  \[
  A_t x_t = R_t \\
  x_t \geq 0
  \]
- Transition function
  \[
  R_{t+1} = b_{t+1} + B_t x_t
  \]

**Stochastic**

» Objective function

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

» Policy

\[
X^\pi : S \rightarrow \mathcal{X}
\]

» Constraints at time \(t\)

\[
x_t = X_t^\pi (S_t) \in \mathcal{X}_t
\]

» Transition function

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

» Exogenous information

\[
(W_1, W_2, \ldots, W_T)
\]
From deterministic to stochastic

**Stochastic problems**
- Modeling is the most important, and hardest, aspect of stochastic optimization.
- Searching for policies is important, but less critical.
- Modeling uncertainty is often overlooked, but is of central importance.
- Evaluating a policy is important, and difficult.

**Deterministic problems**
- Modeling is important, but not central.
- Algorithms are the most important, and hardest part.
- Huh?
- Just add up the costs!!
Modeling stochastic, dynamic problems

The universal objective function

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

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

Now search for policies:

» Policy search:
  • PFAs, CFAs

» Lookahead policies:
  • VFAs, DLAs