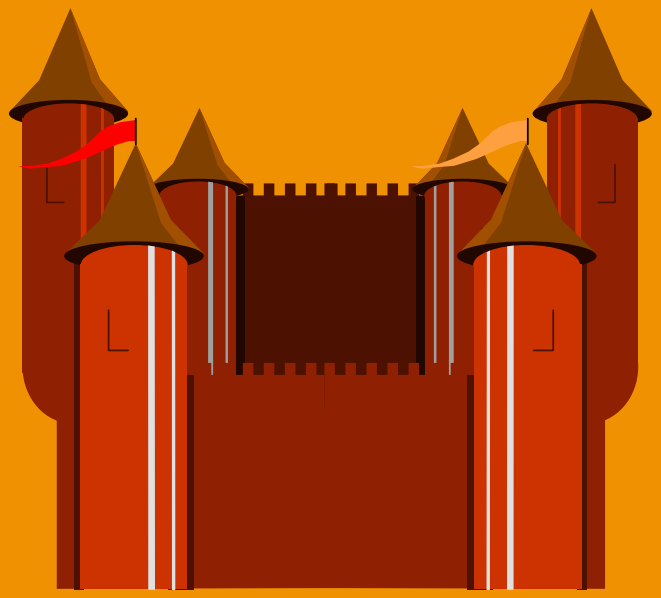
ORF 544

Stochastic Optimization and Learning

Spring, 2019



Warren Powell
Princeton University
http://www.castlelab.princeton.edu

Week 4

Chapter 7: Derivative-free stochastic search

Derivative-free stochastic search

- Notes:
  - » The material for this week and next will all be drawn from chapter 7.
  - » Chapter 7 is 60 pages, and desperately in need of a rewrite. I don't have time to do this, but the lectures will follow a new (and improved) outline.
  - » Derivative-free stochastic optimization is an extremely rich problem class. We will use this to illustrate four fundamental classes of policies, which can be organized along two core strategies:
    - Policy search – Here we will search within a class of policies to identify which work best over time/iterations.
    - Lookahead policies – These are policies that are constructed to optimize the value of an experiment plus the value of the downstream state.

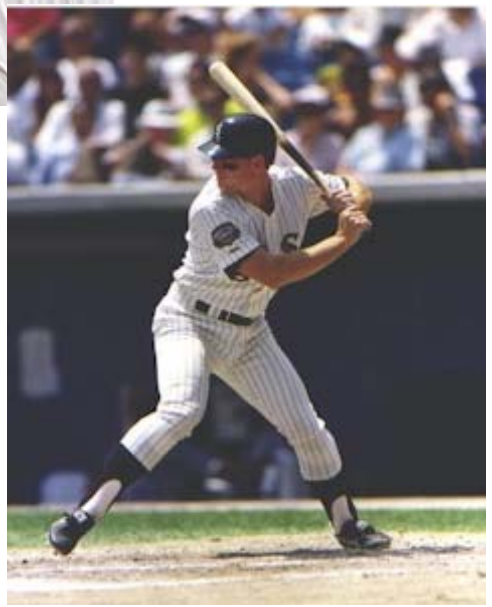
Derivative-free stochastic search

- Week 4 (this week): We will cover:
  - » Introduction to derivative-free stochastic optimization
  - » Introduction to two strategies for developing policies:
    - Policy search class
    - Lookahead class
  - » Then we will focus on the "policy search" class, which can be divided into two classes:
    - Policy function approximations (PFAs)
    - Cost function approximations (CFAs)
  - » We will do the more difficult lookahead classes next week.

Introduction to derivative-free stochastic search

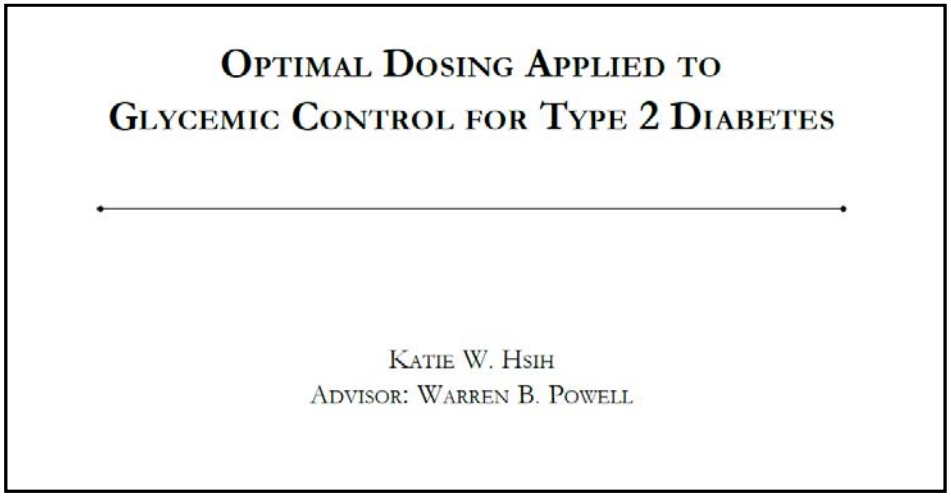
Sports

- Finding the best player
  - » We have a set of players from which to choose a team
  - » The effectiveness of a certain team is best revealed by playing a game
  - » We maximize the total number of games won in the season



Optimal learning in diabetes

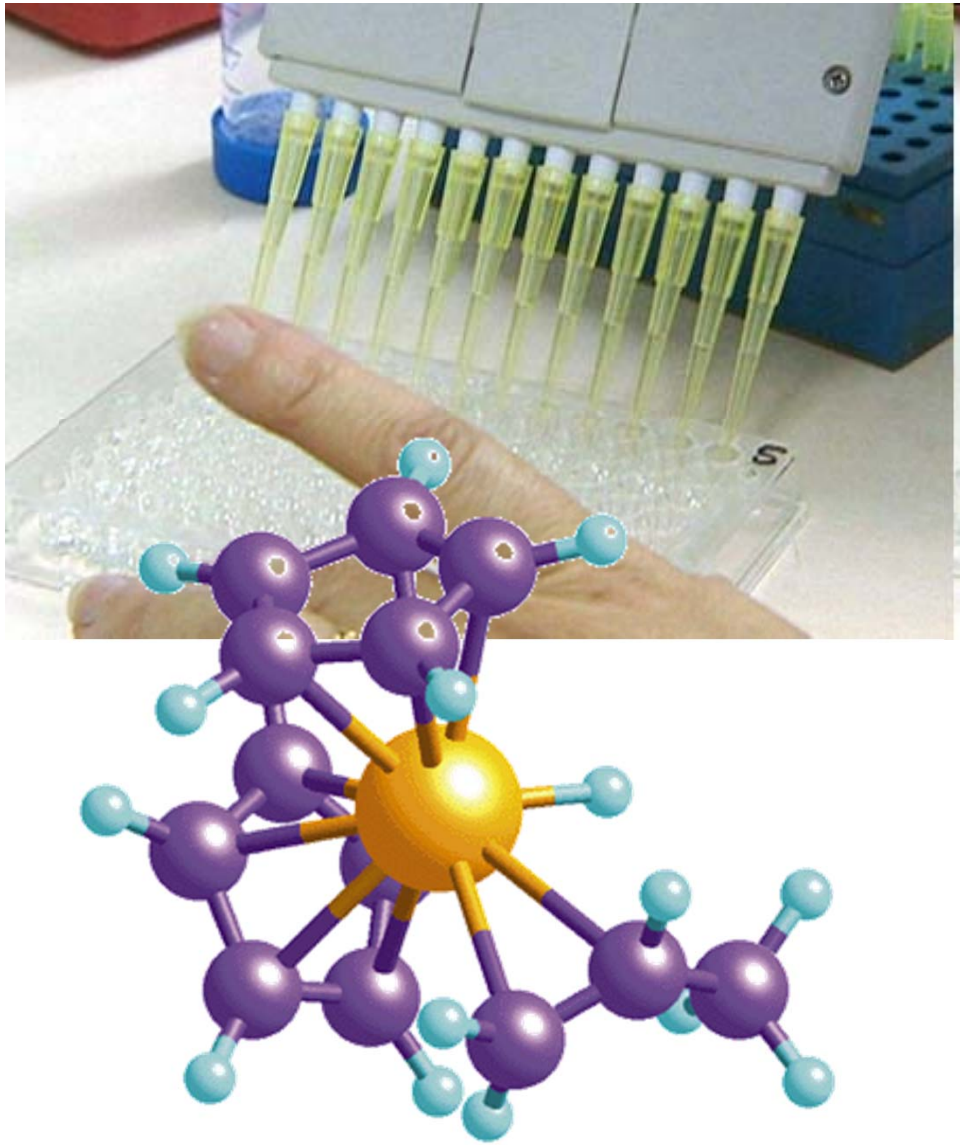
- How do we find the best treatment for diabetes?
  - » The standard treatment is a medication called metformin, which works for about 70 percent of patients.
  - » What do we do when metformin does not work for a patient?
  - » There are about 20 other treatments, and it is a process of trial and error.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


OPTIMAL DOSING APPLIED TO
GLYCEMIC CONTROL FOR TYPE 2 DIABETES

KATIE W. HSIH
ADVISOR: WARREN B. POWELL

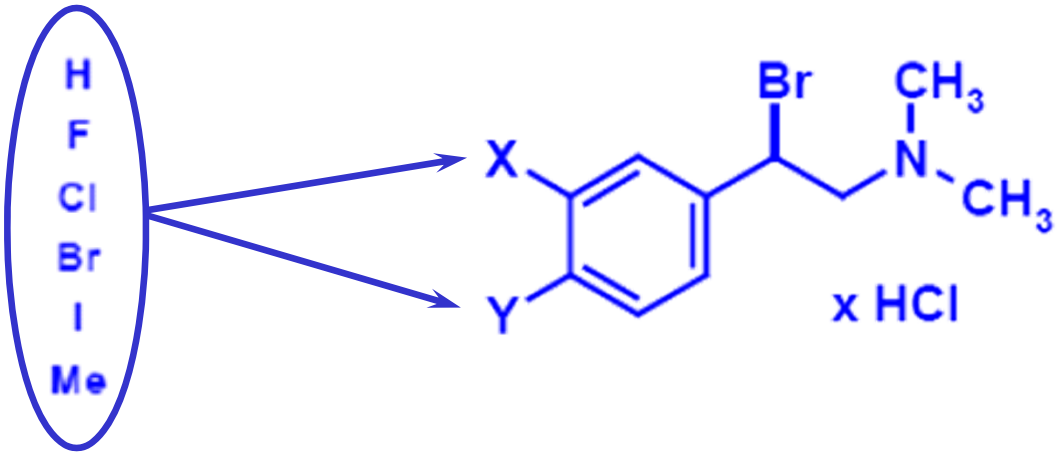
Drug discovery

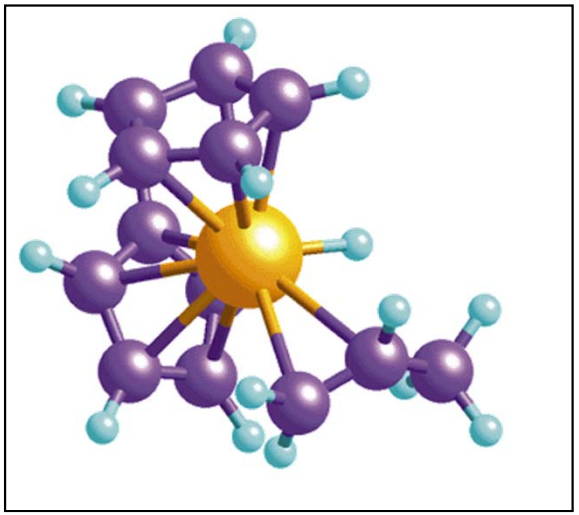
- Biomedical research
  - » How do we find the best drug to cure cancer?
  - » There are millions of combinations, with laboratory budgets that cannot test everything.
  - » We need a method for sequencing experiments.



Drug discovery

- Designing molecules

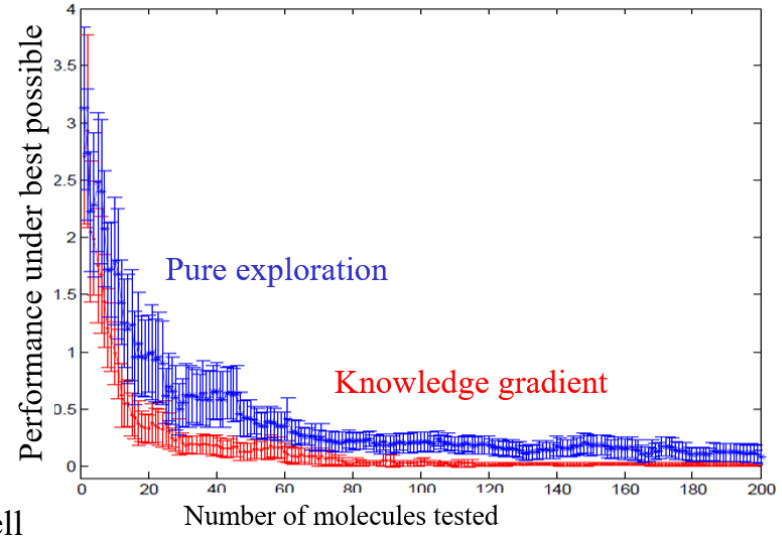




» X and Y are *sites* where we can hang *substituents* to change the behavior of the molecule. We approximate the performance using a linear belief model:

$$Y = \theta_0 + \sum_{\text{sites } i} \sum_{\text{substituents } j} \theta_{ij} X_{ij}$$

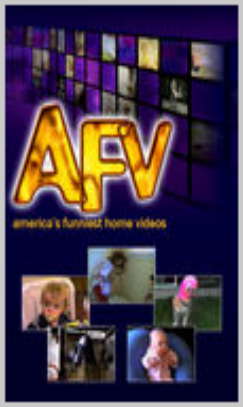
» How to sequence experiments to learn the best molecule as quickly as possible?



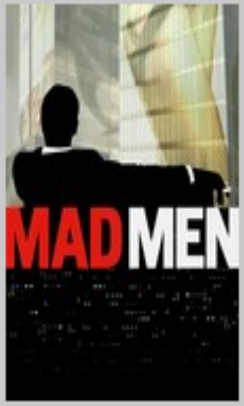
NETFLIX

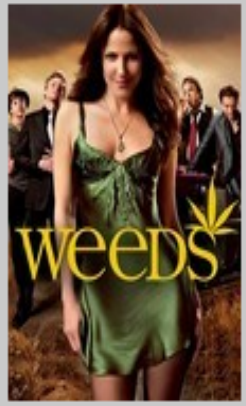
Member Sign In

New Arrivals in TV



TV Drama





TV Comedy

Designing menus

- What should you offer?
  - » No-one will like everything on a menu.
  - » The trick is to find choices that will satisfy people.
  - » You will probably need to observe the response to a particular menu over a period of time.
  - » So, observations are time

Derivative free stochastic search

- Settings

in problems where a function evaluation is relatively expensive. Examples of different costs of function evaluations are:

- An analytical function, such as $\sum_i c_i x_i$, which may take fractions of a second.
- Complex analytical functions, which require large sums or numerical integration, which may take seconds to minutes.
- Computer simulations (of physical or business processes) which take minutes to hours to days or more.
- Laboratory experiments - Testing a new chemical compound or material can take hours to days (or more).
- Field experiments - Evaluating the price of a product, testing a new drug or business process, can take days to weeks to months.

In addition, there are settings (especially with field experiments) where it makes sense to evaluate performance based on the cumulative rewards (equation (7.3)) as opposed to just considering the final design (equation (7.2).

The multiarmed bandit problem


$1,036.03
Triple
DOUBLE

Multiarmed bandit problems

- Bandit problems and objective functions
  - » The classical bandit problem is cumulative reward:

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

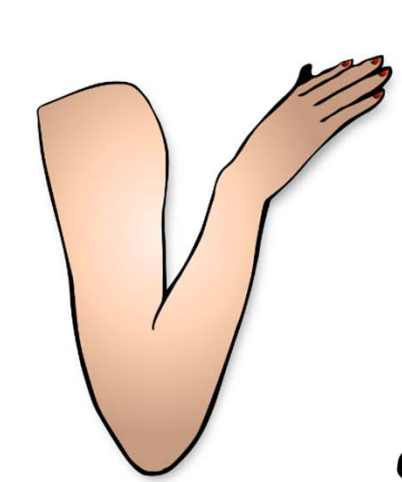
where

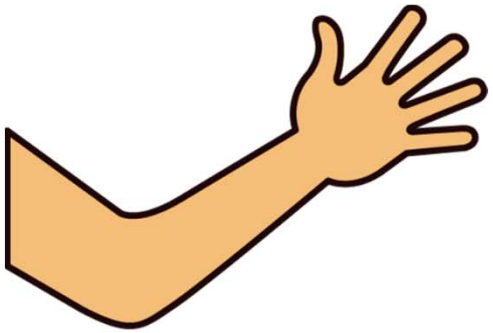
$W^{n+1} = \text{"winnings"}$	New information
$S^n = \text{State of knowledge}$	What we know about each slot machine
$x^n = X^{\pi}(S^n)$	Choose next "arm" to play

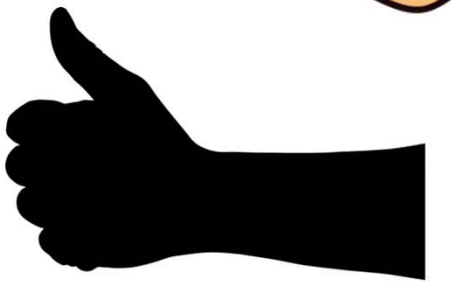
  - » The essential characteristic of bandit problems is the exploration vs. exploitation tradeoff. Do you try something that does not look as good, potentially incurrent a lower reward, so you can learn and make better decisions in the future.
  - » Then the bandit community discovered that the same tradeoff exists in final reward. These problems became known as "best arm" bandit problems in the bandit vocabulary.

Multiarmed bandit problems

- *Arms:*







Multiarmed bandit problems

- *Bandits:*

Multiarmed bandit problems

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

Multiarmed bandit problems

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

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

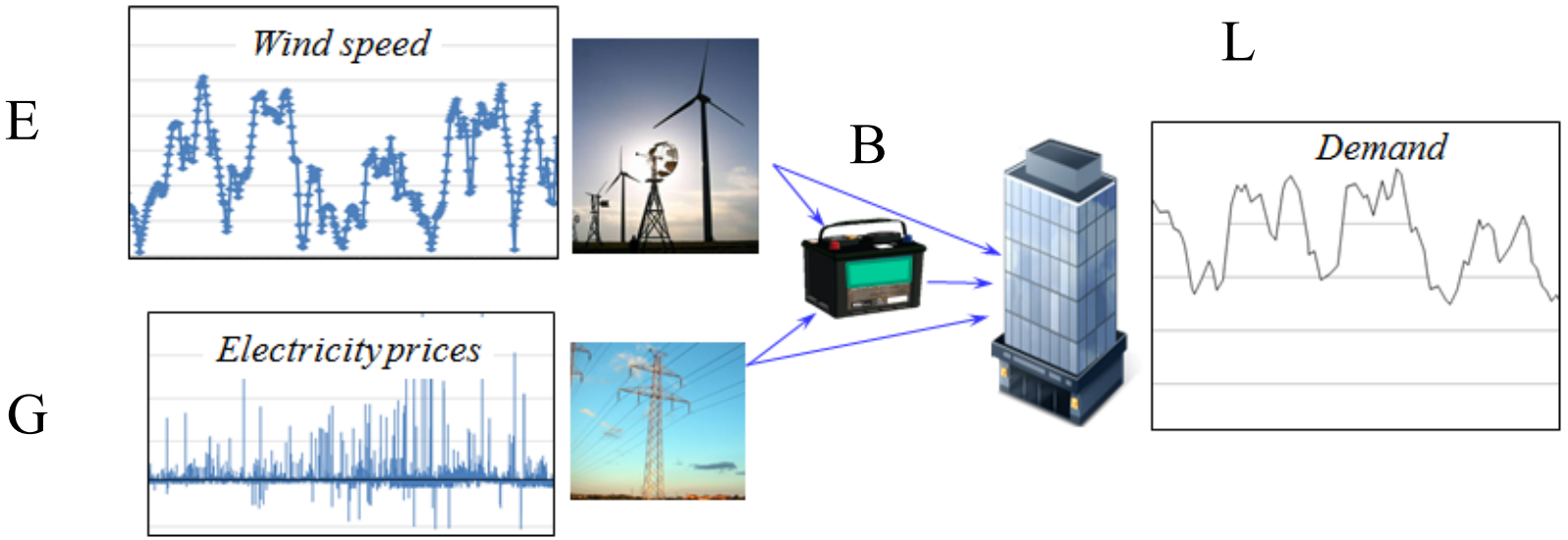
- Dimensions of a "bandit" problem:
  - » Belief models
    - Lookup tables (these are most common)
      - – Independent or correlated beliefs
    - Parametric models
      - – Linear or nonlinear in the parameters
    - Nonparametric models
      - – Locally linear
      - – Deep neural networks/SVM
    - 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

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 learning problem:
    - Maximizing cumulative rewards (finite or infinite horizon)
    - Maximizing the terminal reward with a finite budget.
  - » Fundamental elements of a "bandit" problem:
    - A belief model that describes what we know about the function
    - The ability to learn sequentially about the function.
  - » I prefer to distinguish three problem classes:
    - Active learning – These problems arise when our decisions affect the information that arrives. This opens the door to making decisions that balance current performance against learning to improve future performance.
    - Passive learning – Here we may learn as we go, but we do not directly influence the learning process through our decisions.
    - No learning – This would be for problems where there is no element of learning.

An energy storage problem

- Transition function



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

$$p_{t+1} = \theta_{t0} p_t + \theta_{t1} p_{t-1} + \theta_{t2} p_{t-2} + \varepsilon^p_{t+1}$$

$$D_{t+1} = f^D_{t,t+1} + \varepsilon^D_{t+1}$$

$$R^{battery}_{t+1} = R^{battery}_t + x_t$$

An energy storage problem

- Types of learning:
  - » No learning ($\theta's$ are known)

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

  - » Passive learning (learn θs from price data)

$$p_{t+1} = \bar{\theta}_{t0} p_t + \bar{\theta}_{t1} p_{t-1} + \bar{\theta}_{t2} p_{t-2} + \varepsilon^p_{t+1}$$

We have no control over the evolution of prices.

If we have to learn the parameters, then we have to introduce this learning process into the dynamics.

Learning in stochastic optimization

- Learning the pricing model:
  - » Let p_{t+1} be the new price and let

$$\bar{F}_t^{price}(\bar{p}_t \mid \bar{\theta}_t) = (\bar{\theta}_t)^T \bar{p}_t = \bar{\theta}_{t0} p_t + \bar{\theta}_{t1} p_{t-1} + \bar{\theta}_{t2} p_{t-2}$$

  - » We update our estimate $\bar{\theta}_t$ using our recursive least squares equations:

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

$$\varepsilon_{t+1} = \bar{F}_t^{price}(\bar{p}_t \mid \bar{\theta}_t) - p_{t+1},$$

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

$$\gamma_{t+1} = 1 + (\bar{p}_t)^T B_t \bar{p}_t$$

An energy storage problem

- Types of learning:
  - » No learning ($\theta's$ are known)

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

  - » Passive learning (learn θs from price data)

$$p_{t+1} = \bar{\theta}_{t0} p_t + \bar{\theta}_{t1} p_{t-1} + \bar{\theta}_{t2} p_{t-2} + \varepsilon^p_{t+1}$$

  - » Active learning ("bandit problems")

Buy/sell decisions

$$p_{t+1} = \bar{\theta}_{t0} p_t + \bar{\theta}_{t1} p_{t-1} + \bar{\theta}_{t2} p_{t-2} + \bar{\theta}_{t3} x^{GB}_t + \varepsilon^p_{t+1}$$

Our decisions influence the prices we observe, which helps with learning.

Classes of problems

Major problem classes

- Special structure
  - » There are special cases where we can solve

$$\max_x \mathbb{E}F(x,W)$$ Deterministic!

exactly. But not very many.

- Sampled problems (SAA, scenario trees)
  - » If the only problem is that we cannot compute the expectation, we might solve a sampled approximation

$$\max_x \hat{\mathbb{E}}F(x,W) = \frac{1}{N}\sum_{n=1}^{N} F(x,W^n)$$ Also deterministic!

- Adaptive learning algorithms
  - » This is what we have to turn to for most problems, and is the focus of this tutorial.

Major problem classes

- State independent problems
  - » The *problem* does not depend on the state of the system.

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

  - » The only state variable is what we know (or believe) about the unknown function $\mathbb{E}F(x,W)$, called the belief state B_t, so $S_t = B_t$.
- State dependent problems
  - » Now the *problem* may depend on what we know at time *t*:

$$\max_{0\le x\le R_t} \mathbb{E}C(S,x,W) = \mathbb{E}\left\{p_t\min(x,W) - cx\right\}$$

  - » Now the state is $S_t = (R_t, p_t, B_t)$

Major problem classes

- Offline (final reward)
  - » We can iteratively search for the best solution, but only care about the final answer.
  - » Asymptotic formulation:

$$\max_x \mathbb{E} F(x, W)$$

  - » Finite horizon formulation:

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

"ranking and selection" or "stochastic search"

- Online (cumulative reward)
  - » We have to learn as we go

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

Major problem classes

- There are entire fields of stochastic optimization built around "final reward" and "cumulative reward" objections:
  - » Final reward
    - Stochastic search (e.g. derivative-based stochastic optimization)
    - Ranking and selection (finding the best out of a set of choices)
  - » Cumulative reward
    - Multiarmed bandit problems
    - Classical dynamic programming, optimal control, stochastic programming, …
  - » Our presentation will not care whether we are optimizing final or cumulative reward – it is just an objective function.

Major problem classes

- "Offline" vs. "online" learning
  - » My view (or perhaps the view of stochastic optimization):
    - "offline learning" is learning in the computer. We do not care about making mistakes as long as we get a good answer in the end.
    - "online learning" would be learning in the field, where you have to live with your mistakes.
  - » The machine learning community uses these terms differently:
    - "offline learning" means batch – you have a batch dataset from which you fit a model.
    - "online learning" is sequential, as would happen if data is arriving in the field over time.
    - The problem is that there are many sequential algorithms that are used in "offline" (e.g. laboratory) settings, and the machine learning community calls these "online."

Major problem classes

- Notes:
  - » For chapter 7, we will focus on the top row, “state independent problems.”
  - » These are pure learning problems, where we are trying to learn a deterministic decision or policy.

Modeling

Modeling

- Any learning problem can be written

$$(S^0, x^0, W^1, S^1, x^1, W^2, \ldots$$

where:

» $S^n = (\bar{\mu}_x^n, \beta_x^n)_{x \in X}, \quad x \in X = \{x_1, \ldots, x_M\}$
= Belief about the value of μ_x.

» S^0 = Initial belief

» x^n = Decision made using information from first n experiments.

» x^0 = Initial decision based on information in $S\text{^}0$.

» W^n =Observation from nth experiment, $n = 1,2, \ldots, N$

- Decisions are made using a *policy*:

$$x^n = X^\pi(S^n)$$

Modeling

- All sequential decision problems can be described using:
  - » States
    - Belief states – what do we know about $\mathbb{E}F(x,W)$, or any other function we are learning (transition functions, value functions, policies)
    - Physical and informational states (we will get to these later).
  - » Decisions – What are our choices?
    - Pure information collection decisions (e.g. run an MRI, purchase a credit report)
    - Implementation decisions, but which may also affect what we learn.
  - » Exogenous information
    - Initial belief/prior
    - What we learn from an experiment
  - » Transition function
    - Updating beliefs (statistical estimation)
    - Possibly updating physical state.
  - » Objective function
    - Performance metrics
    - Finding the best policy

Modeling

- State variables
  - » For derivative-free, we often just have a belief state, capturing what we know about our function:
    - Lookup tables
    - Parametric models
    - Nonparametric models
  - » Physical state
    - What node we are at in a network
    - How much inventory do we have
    - We will get to this in the second half of the course
  - » Information state
    - Weather forecast, current price
    - Information state at time t may be independent of the state at t-1.
    - State at time t depends on state at t-1:
      - Information evolves exogenously
      - Our decisions influence the state (e.g. selling stock)

Modeling

- Decisions
  - » We have a finite set of choices:
    - $x \in X = \{x_1, x_2, \ldots, x_M\}$
  - » Examples:
    - Experimenting with different drugs to kill cancer in mice.
    - Running simulations to plan operations (e.g. for Amazon)
    - Evaluating different players for baseball during spring training.
    - Testing different materials to maximize the energy density of a battery.
    - Test marketing new products in specific regions.
  - » We only care about the performance at the end of a set of N experiments. We do not care about how well we do along the way.

Modeling

- Types 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

    $x = (a_1,...,a_I), \quad a_i$ is a category (e.g. red/green/blue)

There are entire fields dedicated to particular classes of decisions.

Modeling

- Exogenous information
  - » Simplest: let $W_{x^n}^{n+1}$ be the performance of alternative x^n.
  - » If we are using a lookup table representation with a Bayesian belief model, we would assume:

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

  - » If we are using a parametric representation, we would write

$$W_{x^n}^{n+1} = f(x^n|\theta) + \varepsilon^{n+1}$$

Modeling

- Transition function:
  - » Now use $W_{x^n}^{n+1}$ to update our belief model:
    - Frequentist?

$$\bar{\mu}_x^{n+1} = \begin{cases} \left(1 - \frac{1}{N_x^n + 1}\right)\bar{\mu}_x^n + \frac{1}{N_x^n + 1} W_x^{n+1} & \text{If } x^n = x \\ \bar{\mu}_x^n & \text{Otherwise} \end{cases}$$

    - Bayesian?

$$\bar{\mu}_x^{n+1} = \begin{cases} \frac{\beta_x^n \bar{\mu}_x^n + \beta^W W_x^{n+1}}{\beta_x^n + \beta^W} & \text{If } x^n = x \\ \bar{\mu}_x^n & \text{Otherwise} \end{cases}$$

  - » Use any of the recursive learning methods from chapter 3.

Modeling

- Objective functions for
  - » State independent and state dependent *problems.*
  - » Final reward and cumulative reward

	Offline Terminal reward	Online Cumulative reward
State independent problems	$\max_\pi \mathbb{E}\{F(x^{\pi,N}, W)\|S_0\}$ Stochastic search (1)	$\max_\pi \mathbb{E}\{\sum_{n=0}^{N-1} F(X^\pi(S^n), W^{n+1})\|S_0\}$ Multiarmed bandit problem (2)
State dependent problems	$\max_{\pi^{lrn}} \mathbb{E}\{C(S, X^{\pi^{impl}}(S\|\theta^{imp}), W)\|S_0\}$ Offline dynamic programming (4)	$\max_\pi \mathbb{E}\{\sum_{t=0}^{T} C(S_t, X^\pi(S_t), W_{t+1})\|S_0\}$ Online dynamic programming (3)

Modeling

- Simulating objective functions
  - » It is important to know how to simulate objective functions. We will get to this in chapter 9.

	Offline Terminal reward	Online Cumulative reward
State independent problems	$\max_\pi \mathbb{E}\{F(x^{\pi,N}, W)\|S_0\}$ Stochastic search (1)	$\max_\pi \mathbb{E}\{\sum_{n=0}^{N-1} F(X^\pi(S^n), W^{n+1})\|S_0\}$ Multiarmed bandit problem (2)
State dependent problems	$\max_{\pi^{lrn}} \mathbb{E}\{C(S, X^{\pi^{impl}}(S\|\theta^{imp}), W)\|S_0\}$ Offline dynamic programming (4)	$\max_\pi \mathbb{E}\{\sum_{t=0}^{T} C(S_t, X^\pi(S_t), W_{t+1})\|S_0\}$ Online dynamic programming (3)

$$\max_{\pi^{lrn}} \mathbb{E}_{S^0} \mathbb{E}^{\pi^{imp}}_{(W_t^n)_{t=0}^T, n=1,\ldots,N|S^0} \left(\mathbb{E}^{\pi^{imp}}_{(\widehat{W}_t)_{t=0}^T|S^0} \frac{1}{T} \sum_{t=0}^{T-1} C(S_t, X^{\pi^{imp}}(S_t|\theta^{imp}), \widehat{W}_{t+1}) \right)$$

Modeling

- Objective functions

	Offline Terminal reward	Online Cumulative reward
State independent problems	$\max_\pi \mathbb{E}\{F(x^{\pi,N}, W)\|S_0\}$ Stochastic search (1)	$\max_\pi \mathbb{E}\{\sum_{n=0}^{N-1} F(X^\pi(S^n), W^{n+1})\|S_0\}$ Multiarmed bandit problem (2)
State dependent problems	$\max_{\pi^{lrn}} \mathbb{E}\{C(S, X^{\pi^{impl}}(S\|\theta^{imp}), W)\|S_0\}$ Offline dynamic programming (4)	$\max_\pi \mathbb{E}\{\sum_{t=0}^{T} C(S_t, X^\pi(S_t), W_{t+1})\|S_0\}$ Online dynamic programming (3)

Learning policies:
- Approximate dynamic programming
- Q-learning
- SDDP
- …

We will focus on these in the second half of the course.

Modeling

- Objective functions

	Offline Terminal reward	Online Cumulative reward
State independent problems	$\max_\pi \mathbb{E}\{F(x^{\pi,N}, W)\|S_0\}$ Stochastic search (1)	$\max_\pi \mathbb{E}\{\sum_{n=0}^{N-1} F(X^\pi(S^n), W^{n+1})\|S_0\}$ Multiarmed bandit problem (2)
State dependent problems	$\max_{\pi^{lrn}} \mathbb{E}\{C(S, X^{\pi^{impl}}(S\|\theta^{imp}), W)\|S_0\}$ Offline dynamic programming (4)	$\max_\pi \mathbb{E}\{\sum_{t=0}^{T} C(S_t, X^\pi(S_t), W_{t+1})\|S_0\}$ Online dynamic programming (3)

"Online" (cumulative reward) dynamic programming is recognized as the "dynamic programming problem," but the entire literature on solving dynamic programs describes class (4) problems. Class (3) appears to be an open problem class.

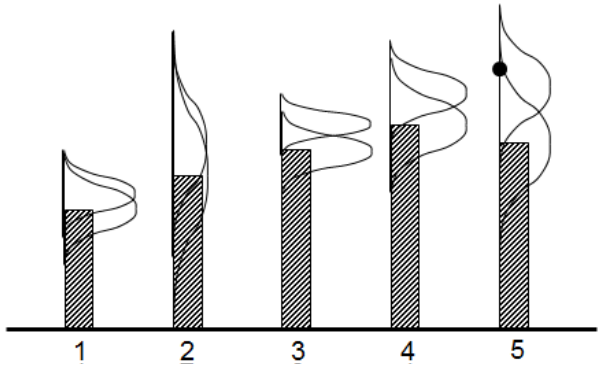
Belief models

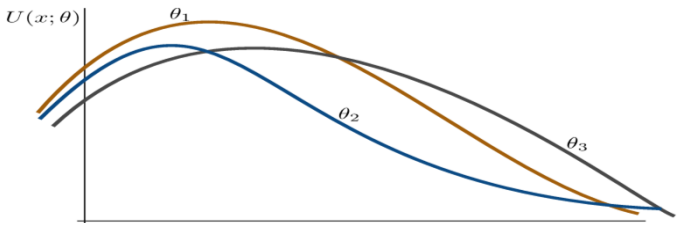
Belief models

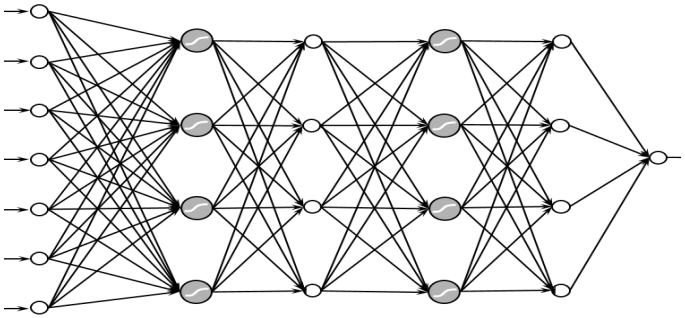
- Notes:
  - » With derivative-based search, we had gradients.
  - » With derivative-free, we have to form a belief model about $\mathbb{E}F(x,W)$. Classes of belief models are
    - Lookup table
      - – Independent beliefs
      - – Correlated beliefs
    - Parametric models
      - – Linear
      - – Nonlinear
        - » Logistic regression
        - » Step functions (sell if price is over some number)
        - » Neural network

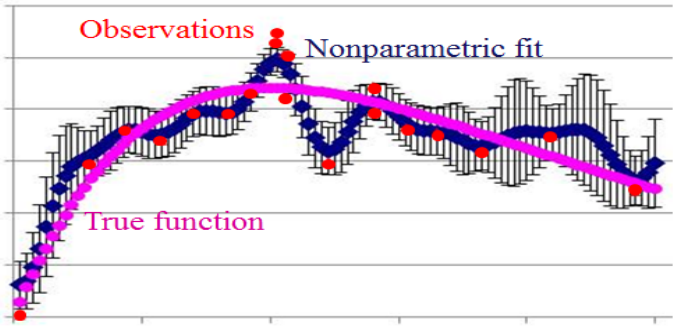
Approximation strategies

- 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
    - Kernel regression
    - Support vector machines
    - Deep neural networks





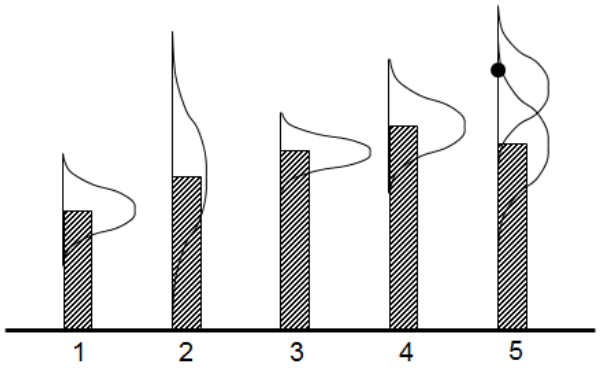




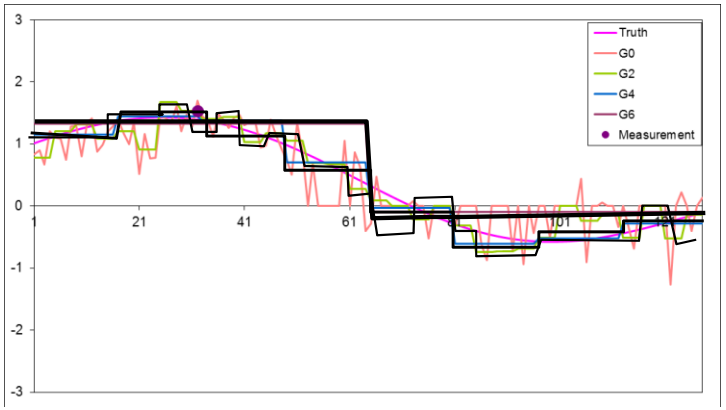
Approximating strategies

- Lookup tables
  - » Independent beliefs

$$\mu_x^n \approx \mathbb{E}F(x,W) \quad x \in \{x_1,...,x_M\}$$



  - » Correlated beliefs
    - A few dozen observations can teach us about thousands of points.
  - » Hierarchical models
    - Create beliefs at different levels of aggregation and then use weighted combinations



Approximating strategies

- Parametric models
  - » Linear models

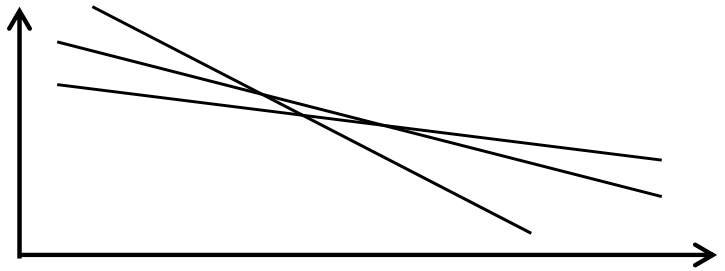
$$F(x|\theta) = \sum_{f \in F} \theta_f \phi_f(x)$$

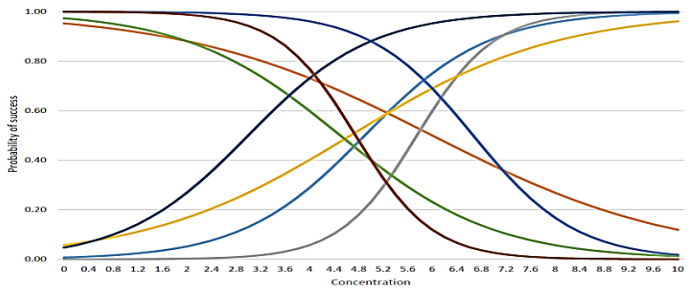
    - Might include sparse-additive, where many parameters are zero.
  - » Nonlinear models

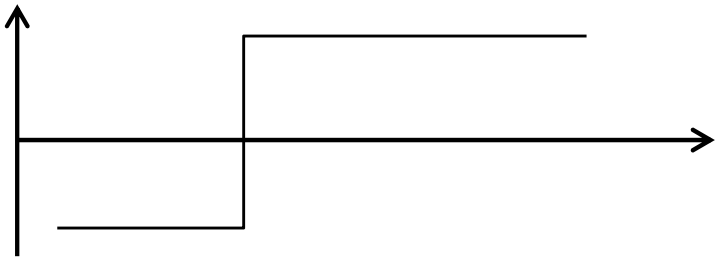
$$F(x|\theta) = \frac{e^{\theta_0 + \theta_1 \phi_1(x) + \ldots}}{1 + e^{\theta_0 + \theta_1 \phi_1(x) + \ldots}}$$

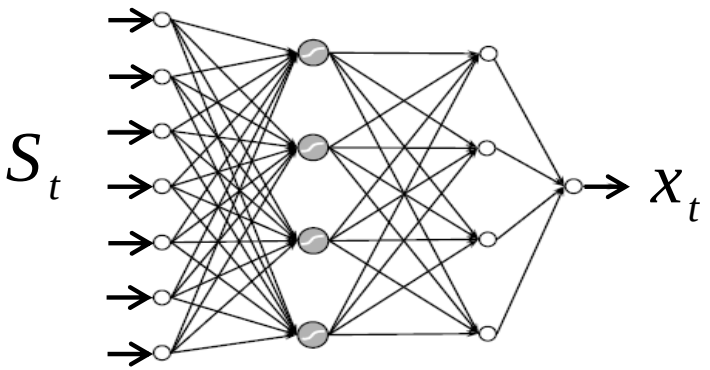
$$X^{\pi}(S_t|\theta) = \begin{cases} +1 & \text{if } p_t < \theta^{\text{charge}} \\ 0 & \text{if } \theta^{\text{charge}} < p_t < \theta^{\text{discharge}} \\ -1 & \text{if } p_t > \theta^{\text{charge}} \end{cases}$$

    - (Shallow) Neural networks



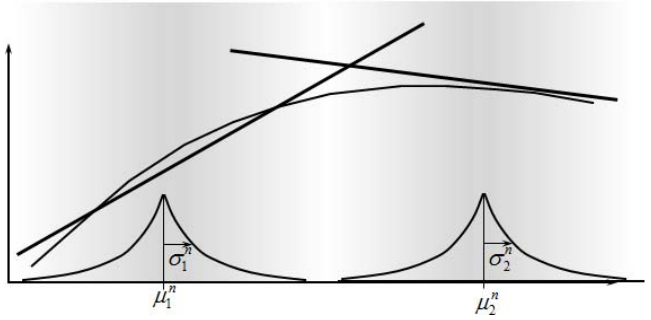


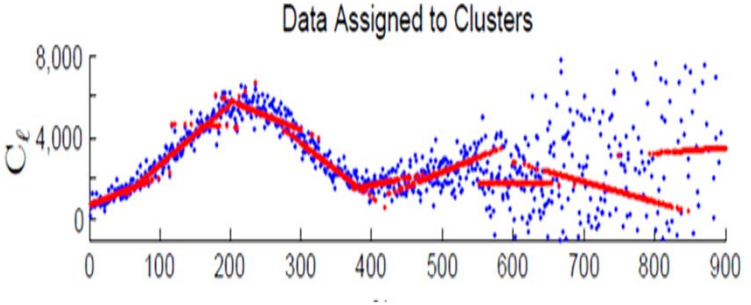


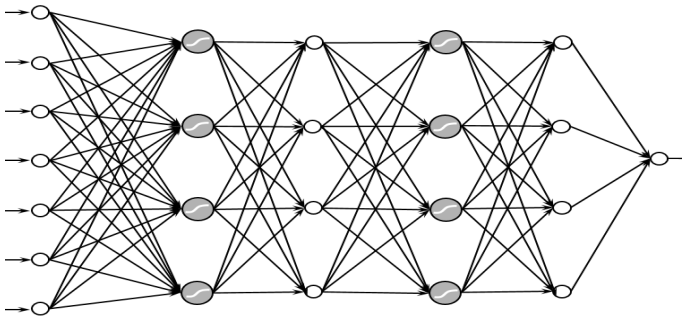


Approximating strategies

- Nonparametric models
  - » Kernel regression
    - Weighted average of neighboring points
    - Limited to low dimensional problems
  - » Locally linear methods
    - Dirichlet process mixtures
    - Radial basis functions
  - » Splines
  - » Support vector machines
  - » Deep neural networks







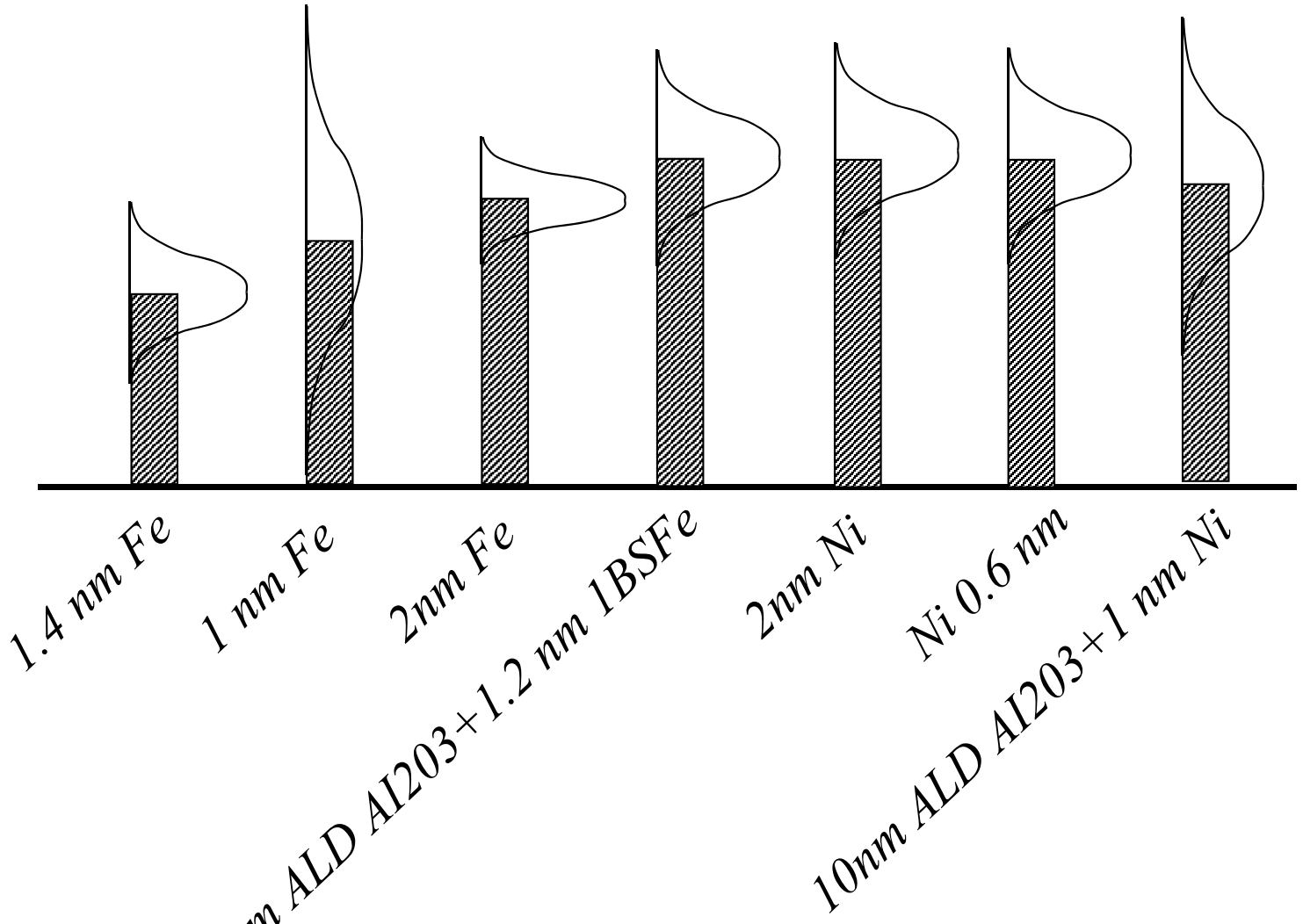
Belief models

- Lookup table
  - » We can organize potential catalysts into groups
  - » Scientists using domain knowledge can estimate correlations in experiments between similar catalysts.

	1.4 nm Fe	1 nm Fe	2nm Fe	10nm ALD Al2O3+1.2 nm IBS Fe	2 nm Ni	Ni 0.6 nm	10nm ALD Al2O3+1 nm Ni
1.4 nm Fe	1	0.7	0.7	0.6	0.4	0.4	0.2
1 nm Fe	0.7	1	0.7	0.6	0.4	0.4	0.2
2nm Fe	0.7	0.7	1	0.6	0.4	0.4	0.2
10nm ALD Al2O3+1.2 nm IBS Fe	0.6	0.6	0.6	1	1	0.3	0
2 nm Ni	0.4	0.4	0.4	1	1	0.7	0.6
Ni 0.6 nm	0.4	0.4	0.4	0.3	0.7	1	0.6
10nm ALD Al2O3+1 nm Ni	0.2	0.2	0.2	0	0.6	0.6	1

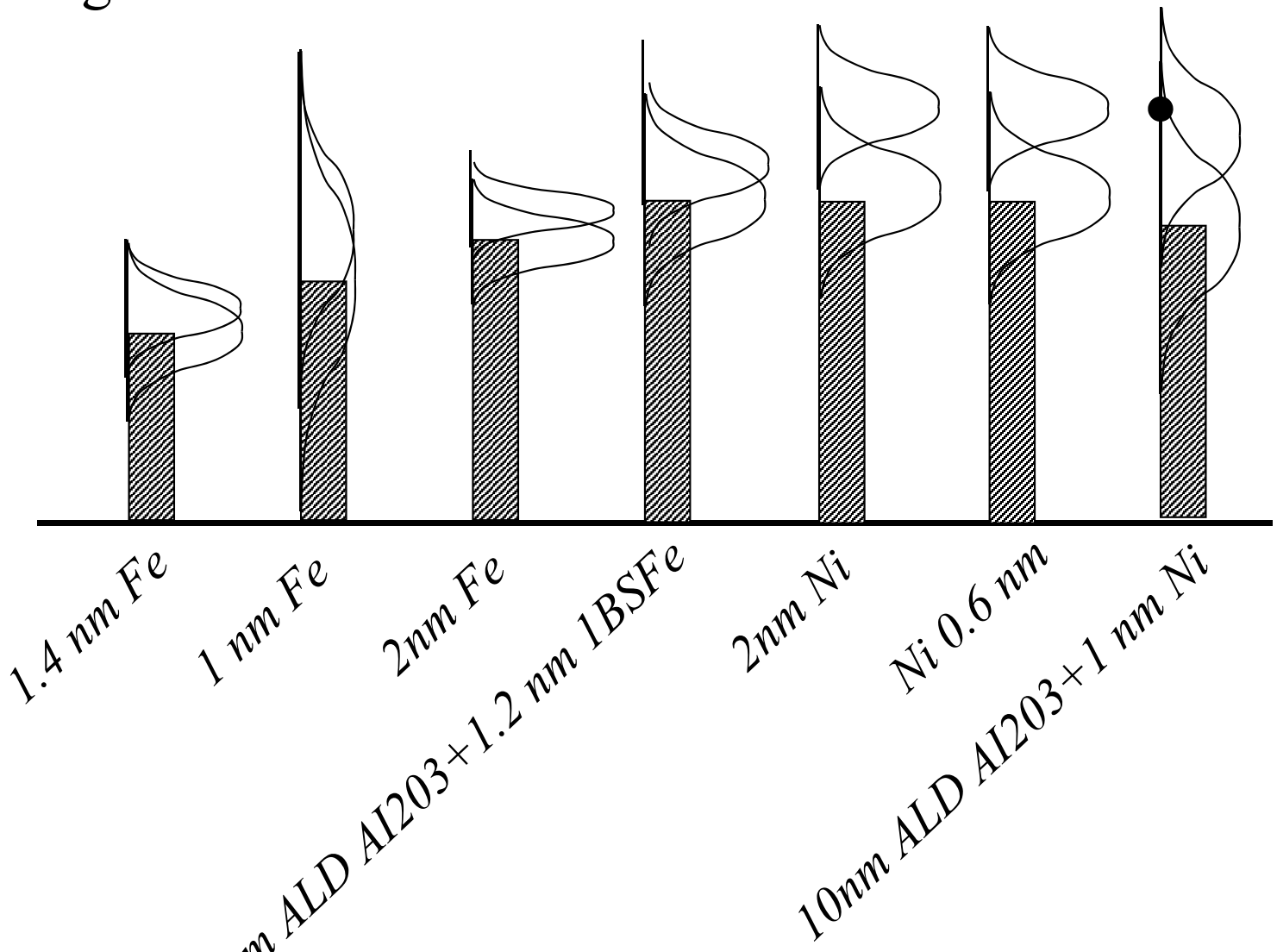
Belief models

- We start with a belief about each material (lookup table)



Belief models

- Testing one material teaches us about other materials



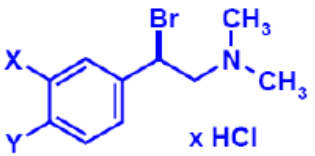
Belief models

- We express our belief using a linear, additive QSAR model
  - $X^m = \left(X_{ij}^m\right)_{ij}$ = Indicator variable for molecule *m*.
  - $Y = \theta_0 + \sum_{\text{sites } i} \sum_{\text{substituents } j} \theta_{ij} X_{ij}$

Hugo Kubinyi, www.kubinyi.de

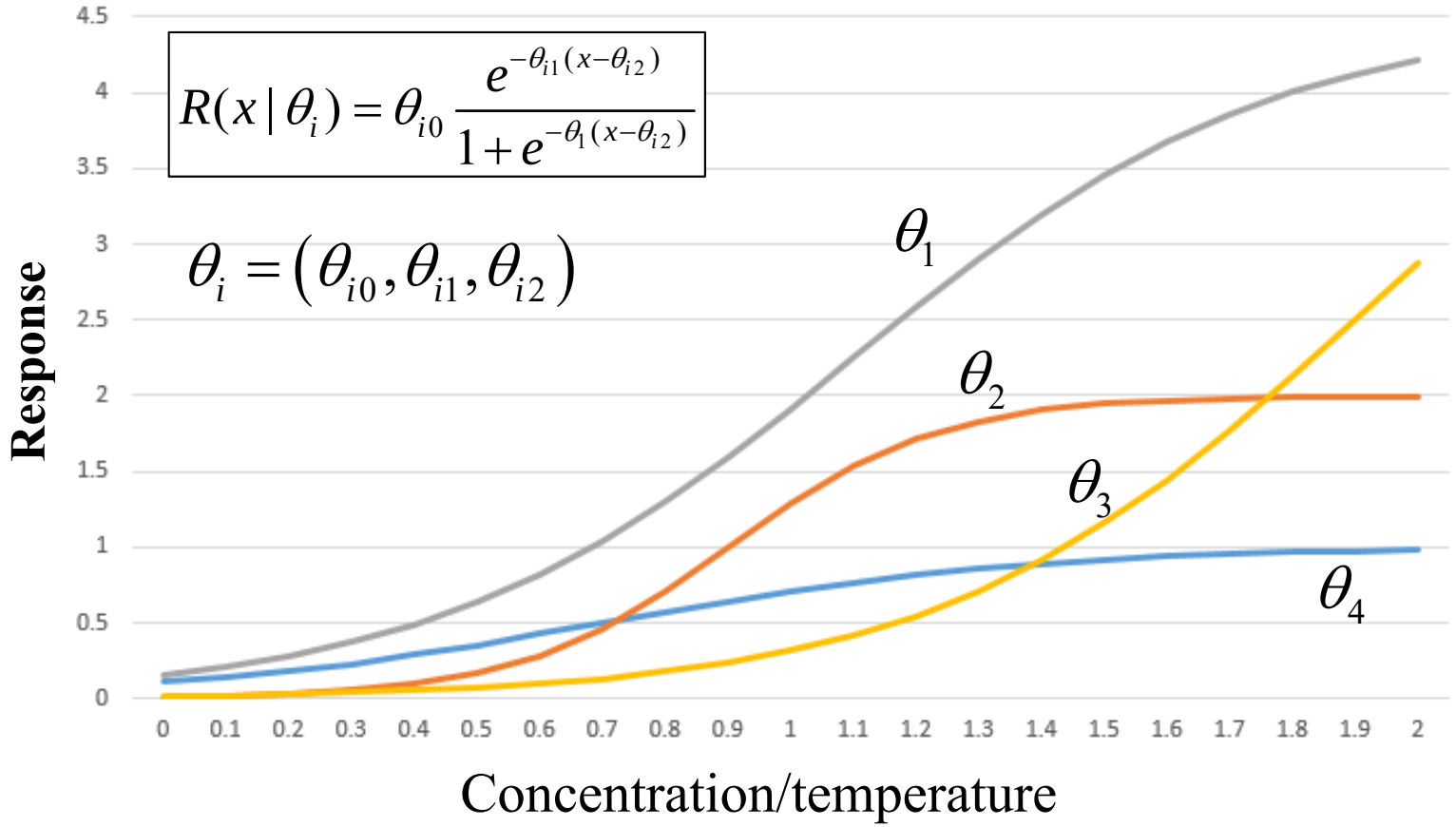
meta (X)	para (Y)	log 1/C obs.	meta- F	Cl	Br	I	Me	para- F	Cl	Br	I	Me	log 1/C calc.
H	H	7.46											7.82
H	F	8.16						1					8.16
H	Cl	8.68							1				8.59
H	Br	8.89								1			8.84
H	I	9.25									1		9.25
H	Me	9.30										1	9.08
F	H	7.52	1										7.52
Cl	H	8.16		1									8.03
Br	H	8.30			1								8.26
I	H	8.40				1							8.40
Me	H	8.46					1						8.28
Cl	F	8.19		1				1					8.37
Br	F	8.57			1			1					8.60
Me	F	8.82					1	1					8.62
Cl	Cl	8.89		1					1				8.80
Br	Cl	8.92			1				1				9.02
Me	Cl	8.96					1		1				9.04
Cl	Br	9.00		1						1			9.05
Br	Br	9.35			1					1			9.28
Me	Br	9.22					1			1			9.30
Me	Me	9.30					1					1	9.53
Br	Me	9.52			1							1	9.51

Matrix for Free Wilson Analysis



Belief models

- A sampled belief model

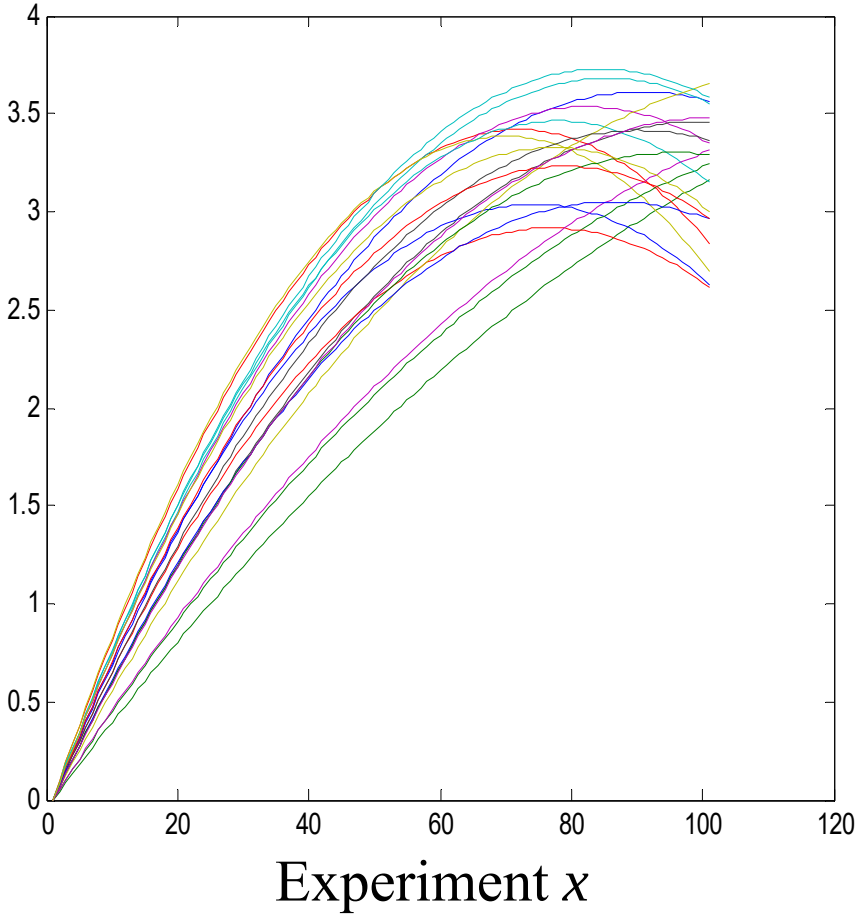


Thickness proportional to p_k^n

Belief models

- Parametric belief models
  - » Value of information (the KG) behaves in an unexpected way
  - » Graph to right shows the value of information while learning a quadratic approximation.
  - » We are using these insights to develop simple rules for where to run experiments that avoid the complexity of KG calculations.

$$Y = \theta_1 x + \theta_2 x^2$$

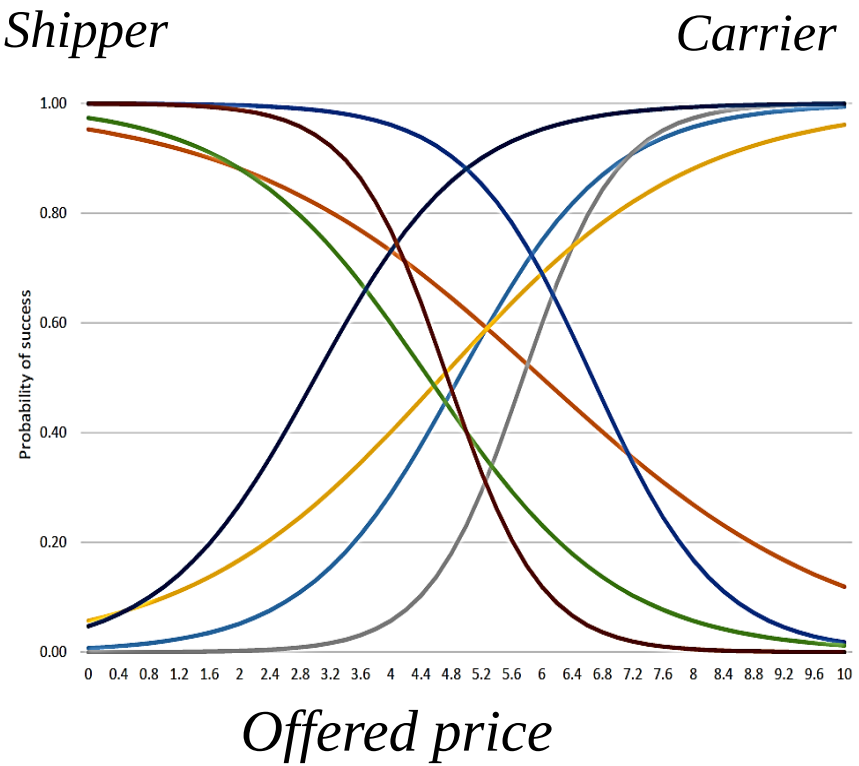


Belief models

- Truckload brokerages:
  - » Now we have a logistic curve for each origin-destination pair (i,j)

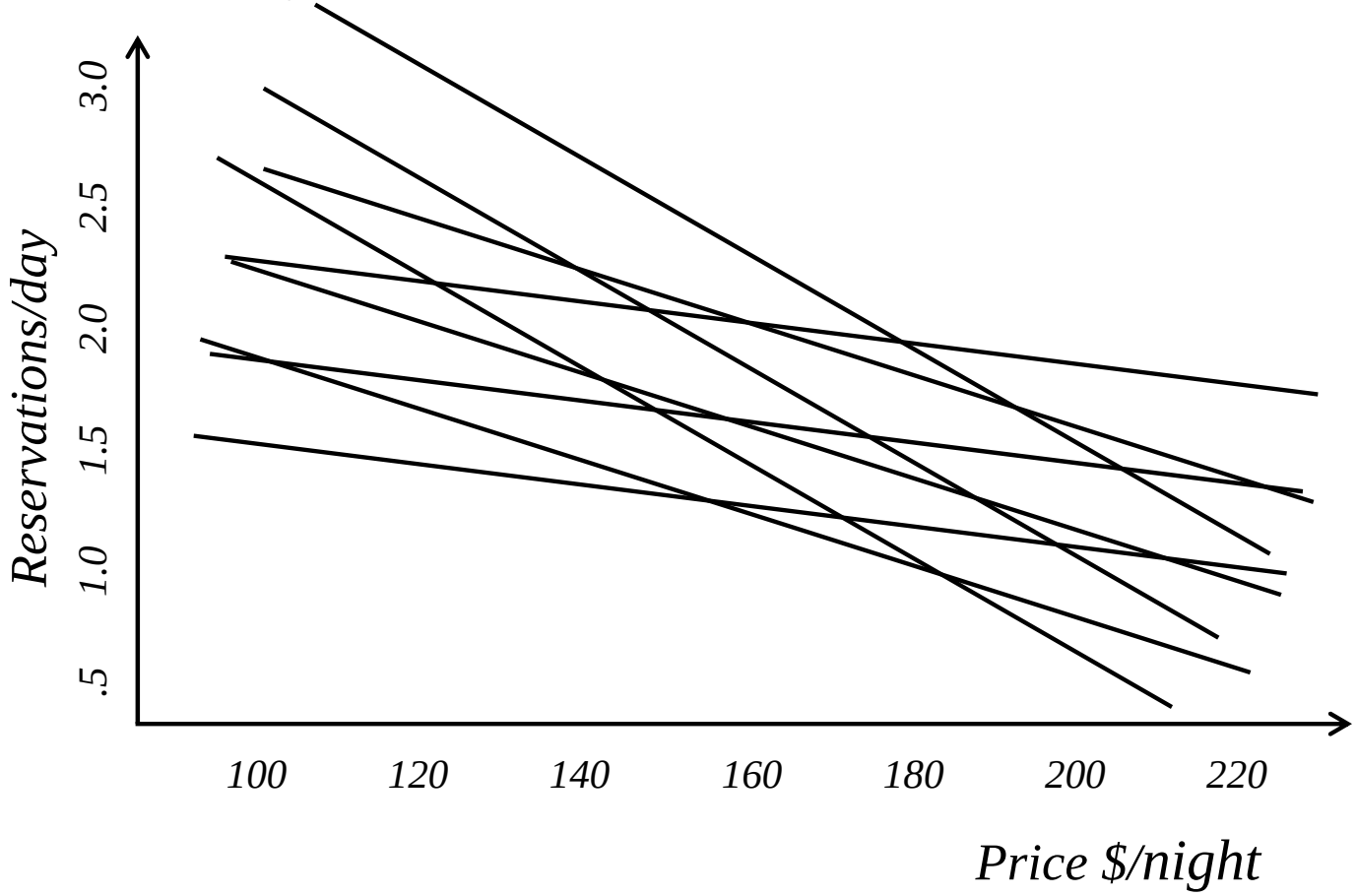
$$P^Y(p,a \mid \theta) = \frac{e^{\theta_{ij}^0 + \theta_{ij} p + \theta_{ij}^a a}}{1 + e^{\theta_{ij}^0 + \theta_{ij} p + \theta_{ij}^a a}}$$

  - » Number of offers for each (i,j) pair is relatively small.
  - » Need to generalize the learning across "traffic lanes."



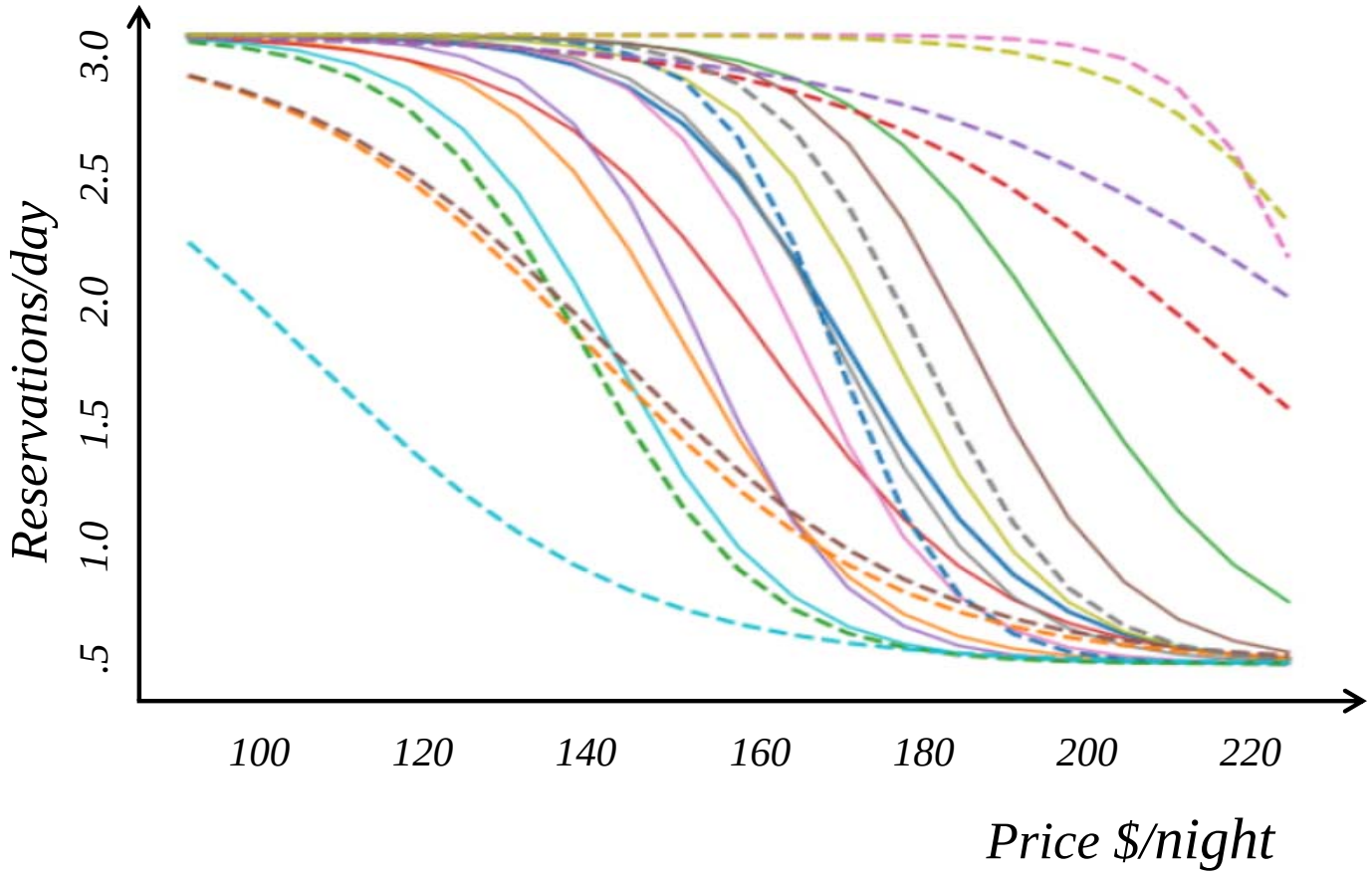
Belief models

- Hotel revenue management
  - » Locally linear curves



Belief models

- Hotel revenue management
  - » Guessing the right logistics curve



Designing policies

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

- Two fundamental strategies:

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

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

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

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

Designing policies

- Policy search:

1a) Policy function approximations (PFAs) $x_t = X^{PFA}(S_t \mid \theta)$

- 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 - $x_t = X^{PFA}(S_t \mid \theta) = \sum_{f \in F} \theta_f \phi_f(S_t)$
  - 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)

$$X^{CFA}(S_t \mid \theta) = \arg\max_{x_t \in \bar{X}_t^{\pi}(\theta)} \bar{C}^{\pi}(S_t, x_t \mid \theta)$$

Designing policies

- Lookahead approximations – Approximate the impact of a decision now on the future:
  - » An optimal policy (based on looking ahead):

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

2a) 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 approximations – Approximate the impact of a decision now on the future:
  - » An optimal policy (based on looking ahead):

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

2a) 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 approximations – Approximate the impact of a decision now on the future:
  - » An optimal policy (based on looking ahead):

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

2a) Approximating the value of being in a downstream state using machine learning ("value function approximations")

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

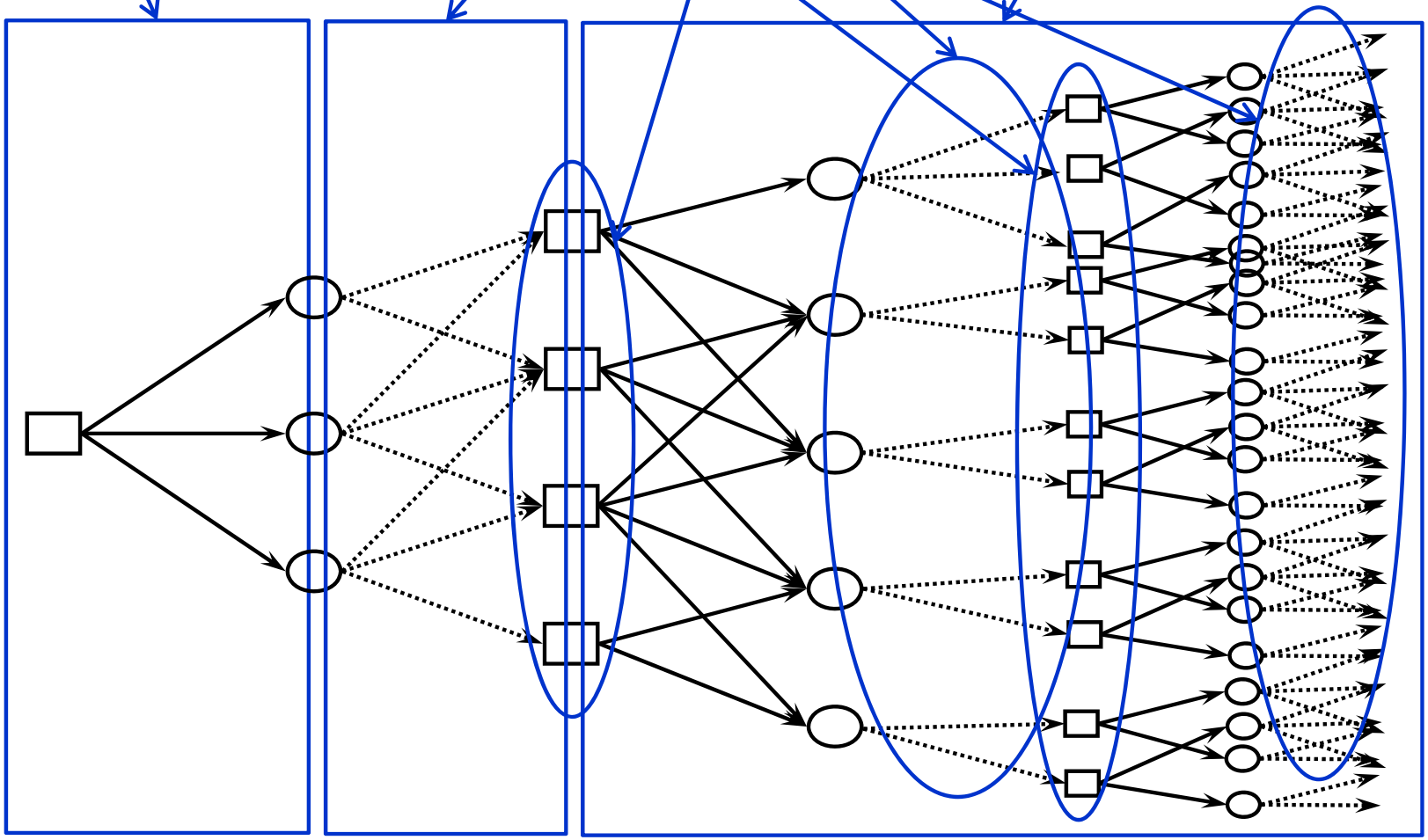
$$X_t^{VFA}(S_t) = \arg\max_{x_t} \left(C(S_t, x_t) + \mathbb{E}\left\{ \bar{V}_{t+1}(S_{t+1}) \mid S_t, x_t \right\} \right)$$

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

Designing policies

- The ultimate lookahead policy is optimal

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



Designing policies

- The ultimate lookahead policy is optimal

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

Maximization that we cannot compute

Expectations that we cannot compute

Designing policies

- The ultimate lookahead policy is optimal

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

» 2b) Instead, we have to solve an approximation called the *lookahead model:*

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

» *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
    - Expected improvement
    - 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

Policy search

1) **Policy function approximations (PFAs)**

» Lookup tables, rules, parametric/nonparametric functions

2) **Cost function approximation (CFAs)**

» $X^{CFA}(S_t \mid \theta) = \arg\max_{x_t \in \bar{X}_t^{\pi}(\theta)} \bar{C}^{\pi}(S_t, x_t \mid \theta)$

Lookahead approximations

3) **Policies based on value function approximations (VFAs)**

» $X_t^{VFA}(S_t) = \arg\max_{x_t} \left(C(S_t, x_t) + \bar{V}_t^{x} \left(S_t^{x}(S_t, x_t) \right) \right)$

4) **Direct lookahead policies (DLAs)**

» *Deterministic lookahead/rolling horizon proc./model predictive control*

$$X_t^{LA-D}(S_t) = \arg\max_{\tilde{x}_{tt}, \ldots, \tilde{x}_{t,t+H}} C(\tilde{S}_{tt}, \tilde{x}_{tt}) + \sum_{t'=t+1}^{T} C(\tilde{S}_{tt'}, \tilde{x}_{tt'})$$

» *Chance constrained programming*

$$P[A_t x_t \le f(W)] \le 1 - \delta$$

» *Stochastic lookahead /stochastic prog/Monte Carlo tree search*

$$X_t^{LA-S}(S_t) = \arg\max_{\tilde{x}_{tt}, \tilde{x}_{t,t+1}, \ldots, \tilde{x}_{t,t+T}} C(\tilde{S}_{tt}, \tilde{x}_{tt}) + \sum_{\tilde{\omega} \in \tilde{\Omega}_t} p(\tilde{\omega}) \sum_{t'=t+1}^{T} C(\tilde{S}_{tt'}(\tilde{\omega}), \tilde{x}_{tt'}(\tilde{\omega}))$$

» *"Robust optimization"*

$$X_t^{LA-RO}(S_t) = \arg\max_{\tilde{x}_{tt}, \ldots, \tilde{x}_{t,t+H}} \min_{w \in W_t(\theta)} C(\tilde{S}_{tt}, \tilde{x}_{tt}) + \sum_{t'=t+1}^{T} C(\tilde{S}_{tt'}(w), \tilde{x}_{tt'}(w))$$

Four (meta)classes of policies

Function approx.

1) **Policy function approximations (PFAs)**
   - » Lookup tables, rules, parametric/nonparametric functions
2) **Cost function approximation (CFAs)**
   - » $X^{CFA}(S_t \mid \theta) = \arg\max_{x_t \in \bar{X}_t^{\pi}(\theta)} \bar{C}^{\pi}(S_t, x_t \mid \theta)$
3) **Policies based on value function approximations (VFAs)**
   - » $X_t^{VFA}(S_t) = \arg\max_{x_t} \left(C(S_t, x_t) + \bar{V}_t^{x} \left(S_t^{x}(S_t, x_t) \right) \right)$
4) **Direct lookahead policies (DLAs)**
   - » *Deterministic lookahead/rolling horizon proc./model predictive control*

$$X_t^{LA-D}(S_t) = \arg\max_{\tilde{x}_{tt}, \ldots, \tilde{x}_{t,t+H}} C(\tilde{S}_{tt}, \tilde{x}_{tt}) + \sum_{t'=t+1}^{T} C(\tilde{S}_{tt'}, \tilde{x}_{tt'})$$

   - » *Chance constrained programming*

$$P[A_t x_t \le f(W)] \le 1 - \delta$$

   - » *Stochastic lookahead /stochastic prog/Monte Carlo tree search*

$$X_t^{LA-S}(S_t) = \arg\max_{\tilde{x}_{tt}, \tilde{x}_{t,t+1}, \ldots, \tilde{x}_{t,t+T}} C(\tilde{S}_{tt}, \tilde{x}_{tt}) + \sum_{\tilde{\omega} \in \tilde{\Omega}_t} p(\tilde{\omega}) \sum_{t'=t+1}^{T} C(\tilde{S}_{tt'}(\tilde{\omega}), \tilde{x}_{tt'}(\tilde{\omega}))$$

   - » *"Robust optimization"*

$$X_t^{LA-RO}(S_t) = \arg\max_{\tilde{x}_{tt}, \ldots, \tilde{x}_{t,t+H}} \min_{w \in W_t(\theta)} C(\tilde{S}_{tt}, \tilde{x}_{tt}) + \sum_{t'=t+1}^{T} C(\tilde{S}_{tt'}(w), \tilde{x}_{tt'}(w))$$

Four (meta)classes of policies

1) **Policy function approximations (PFAs)**

» Lookup tables, rules, parametric/nonparametric functions

Imbedded optimization

2) **Cost function approximation (CFAs)**

» $X^{CFA}(S_t \mid \theta) = \arg\max_{x_t \in \bar{X}_t^{\pi}(\theta)} \bar{C}^{\pi}(S_t, x_t \mid \theta)$

3) **Policies based on value function approximations (VFAs)**

» $X_t^{VFA}(S_t) = \arg\max_{x_t} \left(C(S_t, x_t) + \bar{V}_t^{x} \left(S_t^{x}(S_t, x_t) \right) \right)$

4) **Direct lookahead policies (DLAs)**

» *Deterministic lookahead/rolling horizon proc./model predictive control*

$$X_t^{LA-D}(S_t) = \arg\max_{\tilde{x}_{tt}, \ldots, \tilde{x}_{t,t+H}} C(\tilde{S}_{tt}, \tilde{x}_{tt}) + \sum_{t'=t+1}^{T} C(\tilde{S}_{tt'}, \tilde{x}_{tt'})$$

» *Chance constrained programming*

$$P[A_t x_t \le f(W)] \le 1 - \delta$$

» *Stochastic lookahead /stochastic prog/Monte Carlo tree search*

$$X_t^{LA-S}(S_t) = \arg\max_{\tilde{x}_{tt}, \tilde{x}_{t,t+1}, \ldots, \tilde{x}_{t,t+T}} C(\tilde{S}_{tt}, \tilde{x}_{tt}) + \sum_{\tilde{\omega} \in \tilde{\Omega}_t} p(\tilde{\omega}) \sum_{t'=t+1}^{T} C(\tilde{S}_{tt'}(\tilde{\omega}), \tilde{x}_{tt'}(\tilde{\omega}))$$

» *"Robust optimization"*

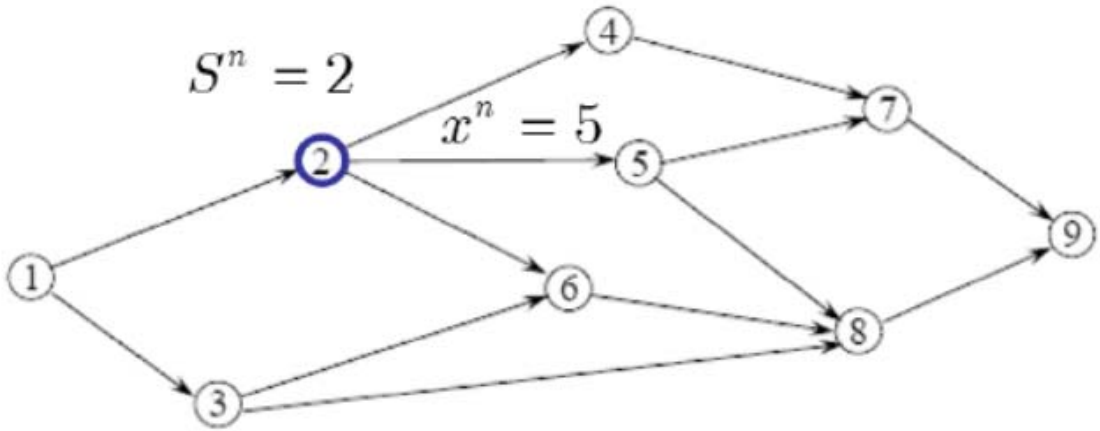
$$X_t^{LA-RO}(S_t) = \arg\max_{\tilde{x}_{tt}, \ldots, \tilde{x}_{t,t+H}} \min_{w \in W_t(\theta)} C(\tilde{S}_{tt}, \tilde{x}_{tt}) + \sum_{t'=t+1}^{T} C(\tilde{S}_{tt'}(w), \tilde{x}_{tt'}(w))$$

Optimal policies

Online vs. offline learning

Optimal policies

- Earlier we expressed an optimal policy using Bellman's equation:
  - » Recall that we used a graph:

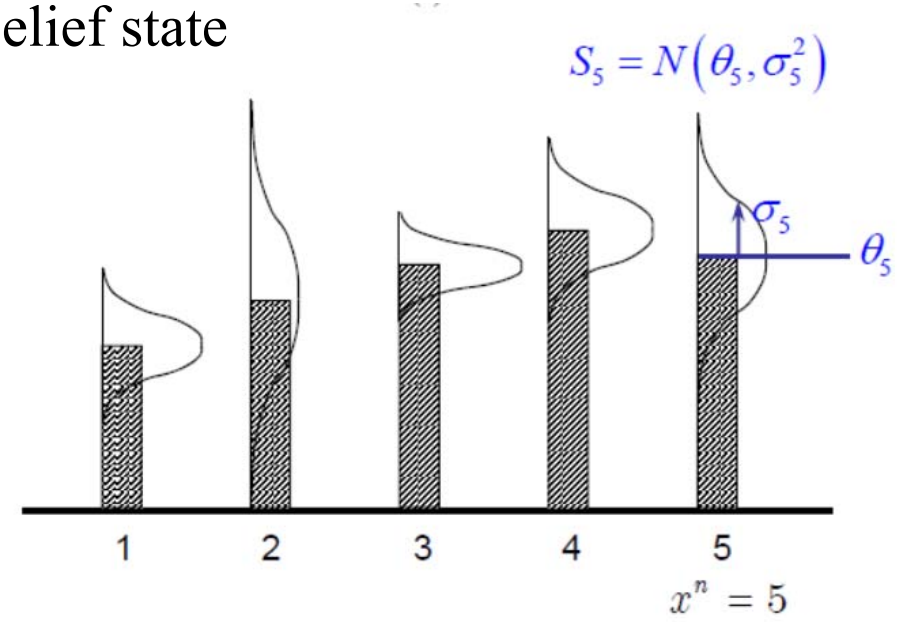


$$V(S) = \max_x \left(C(S,x) + V(S^M(S,x))\right).$$

  - » Bellman equation (state = node)

Optimal policies

- Bellman's equation for a learning problem:
  - » A belief state



$$V(S) = \max_x \left(C(S,x) + \mathbb{E}_W V(S^M(S,x,W))\right). \quad (3.7)$$

$$X^*(S) = \arg\max_x \left(C(S,x) + \mathbb{E}_W V(S^M(S,x,W))\right). \quad (3.8)$$

  - » Can only solve this in very special cases (and not with normally distributed beliefs). 5 alternatives -> 10-dimensional continuous state.

Optimal policies

- Online (cumulative reward) vs. offline (final reward)

  » Bellman's equation for online (cumulative reward) learning:

$$V^n(S^n) = \max_x (C(S^n, x^n) + \mathbb{E}\{V^{n+1}(S^{n+1})|S^n\})$$

  » Bellman's equation for offline (final reward) learning:

$$V^n(S^n) = \max_x (\quad 0 \quad + \mathbb{E}\{V^{n+1}(S^{n+1})|S^n\})$$

  » If we only care about the final reward, we do not care

Policy function approximation

Policy function approximation

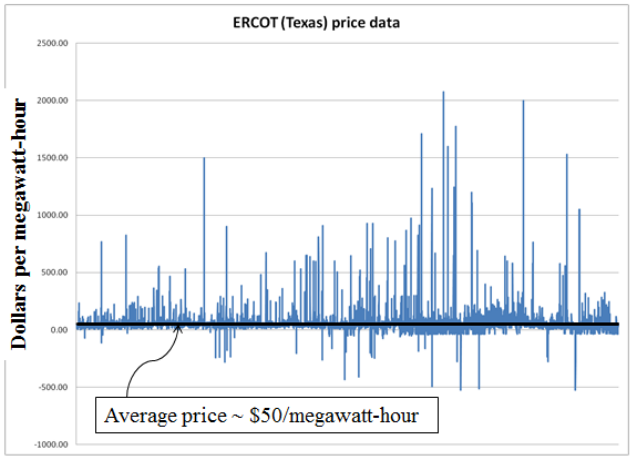
- Examples:

  » Optimizing buying-selling energy from the grid

  » Finding the best selling price for my book on Amazon.

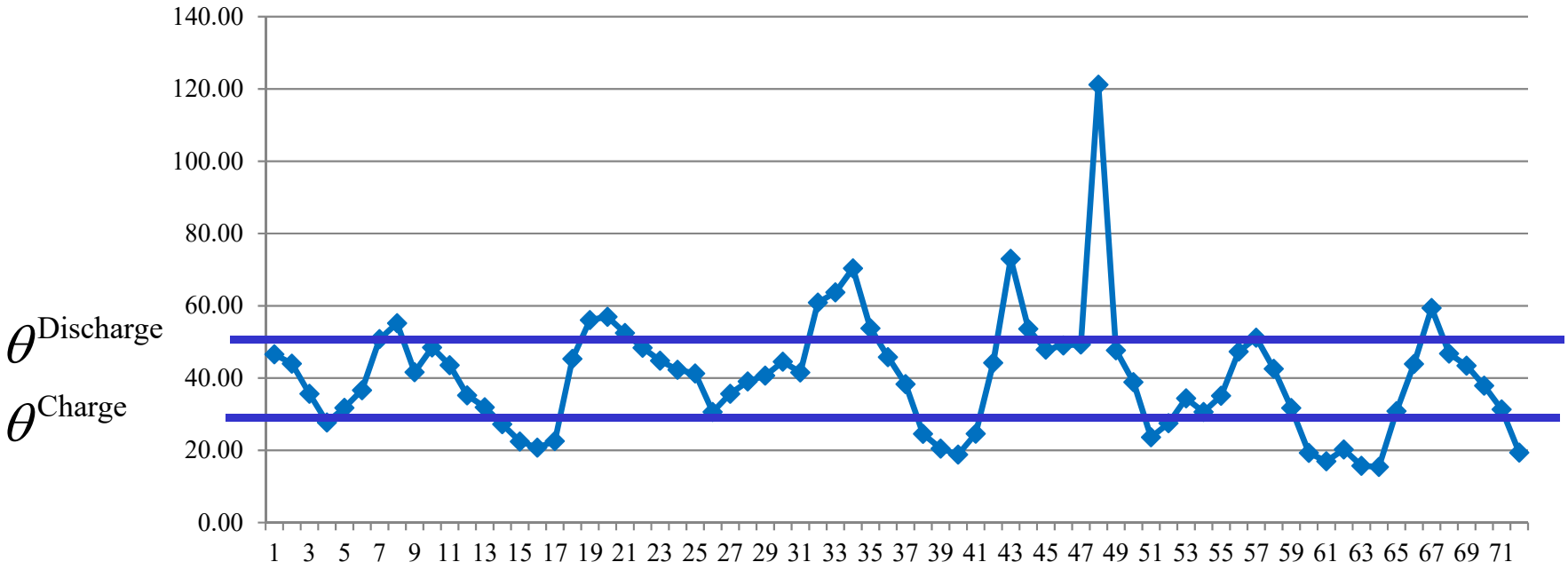
Policy function approximations

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



Policy function approximations

- Grid operators require that batteries bid charge and discharge prices, an hour in advance.

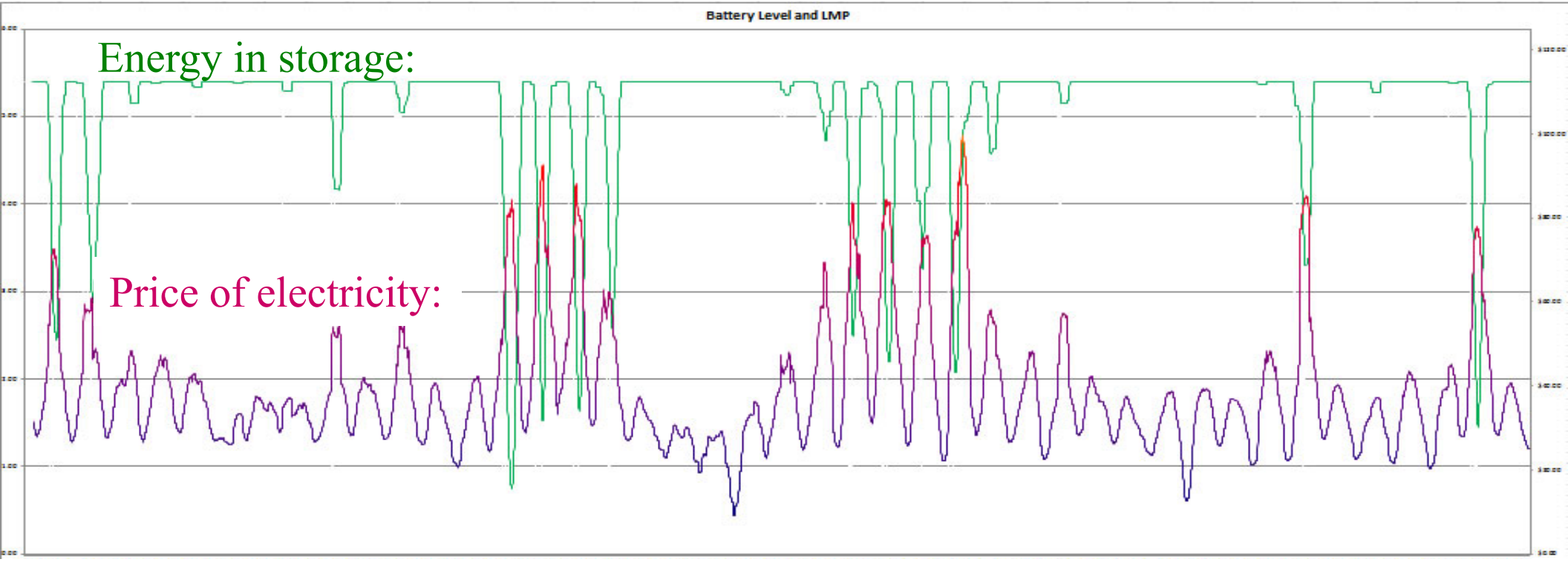


- We have to search for the best values for the policy parameters θ^{Charge} and $\theta^{\text{Discharge}}$.

Policy function approximations

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

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

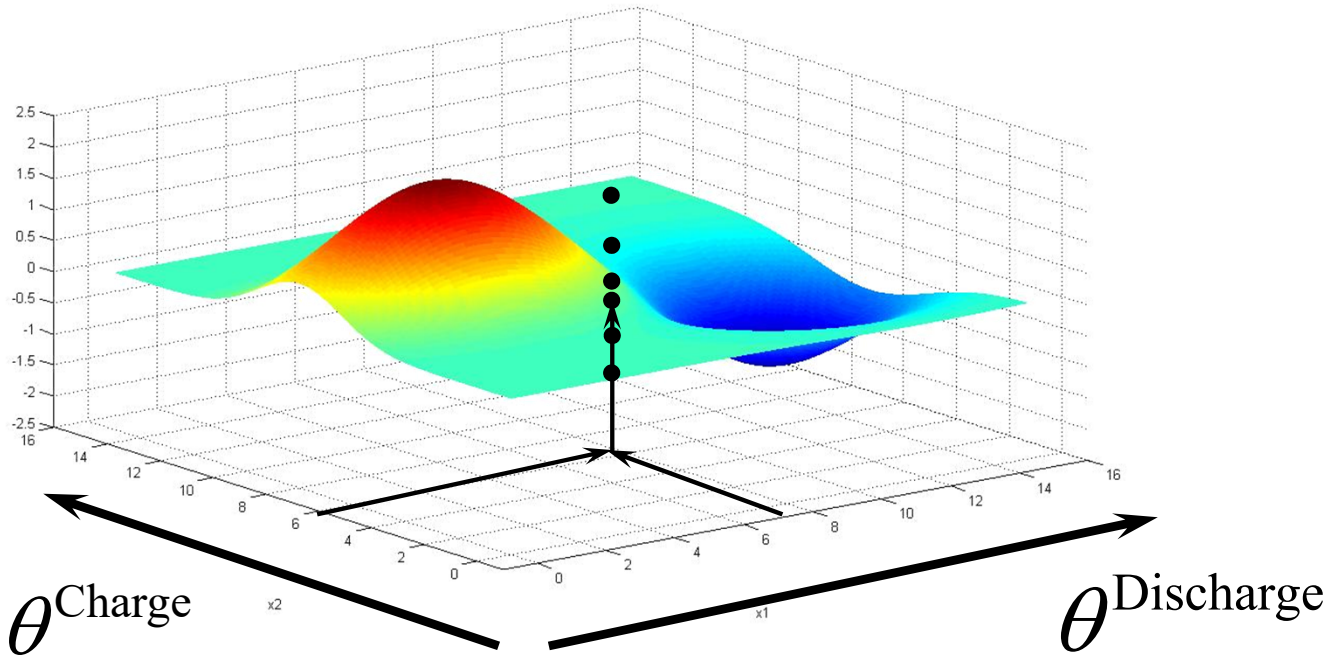


Policy function approximations

- Finding the best policy
  - » We need to maximize

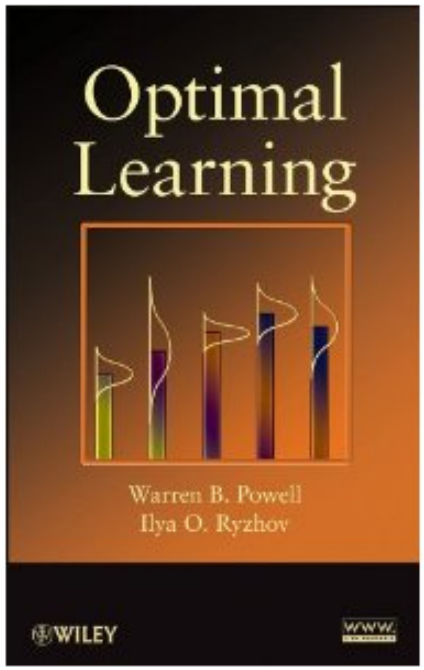
$$\max_\theta F(\theta) = \mathbb{E}\sum_{t=0}^{T} \gamma^t C\left(S_t, X_t^\pi(S_t \mid \theta)\right)$$

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



Policy function approximation

- Optimal pricing



See larger image

Publisher: learn how customers can search inside this book.

Tell the Publisher!
I'd like to read this book on Kindle

Don't have a Kindle? Get your Kindle here, or download a **FREE** Kindle Reading App.

Optimal Learning (Wiley Series in Probability and Statistics) [Hardcover]

Warren B. Powell (Author), Ilya O. Ryzhov (Author)

Like (0)

List Price: ~~$115.00~~

Price: $101.16 & this item ships for **FREE with Super Saver Shipping.** Details

You Save: $13.84 (12%)

Pre-order Price Guarantee. Learn more.

This title has not yet been released.
You may pre-order it now and we will deliver it to you when it arrives.
Ships from and sold by **Amazon.com**. Gift-wrap available.

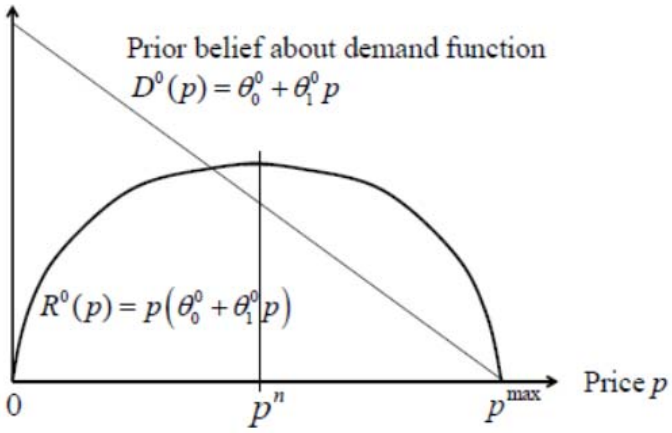
FREE Two-Day Shipping for students on millions of items. Learn more

Policy function approximation

- Dynamic pricing
  - » After n sales, our estimate of the demand function is
  - » $\mathrm{D}(p|\bar{\theta}^n) = \bar{\theta}_0^n - \bar{\theta}_1^n p$
  - » Revenue is

$$\mathrm{R}(p|\bar{\theta}^n) = p\mathrm{D}(p|\bar{\theta}^n)$$
$$= \bar{\theta}_0^n p - \bar{\theta}_1^n p^2$$

  - » The optimal price given our estimates would be:
  - » $p^n = \frac{\bar{\theta}_0^n}{2\bar{\theta}_1^n}$



Revenue management

- Earning vs. learning
  - » You earn the most with prices near the middle.
  - » You learn the most with extreme prices.
  - » Challenge is to strike a balance.

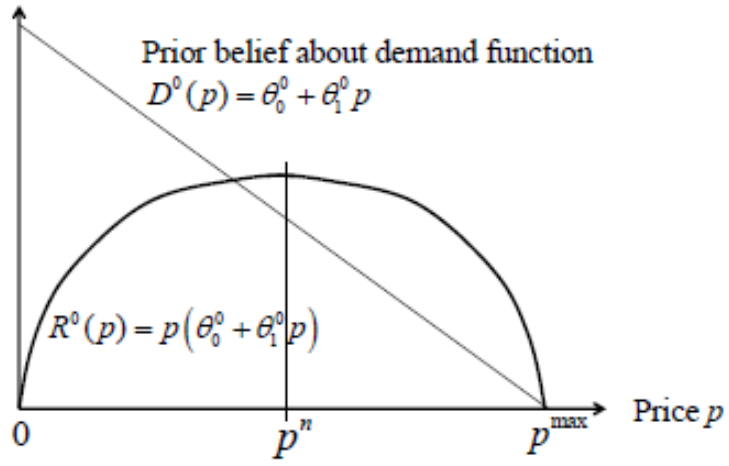


Figure 8.5 Prior demand function and revenue curve.

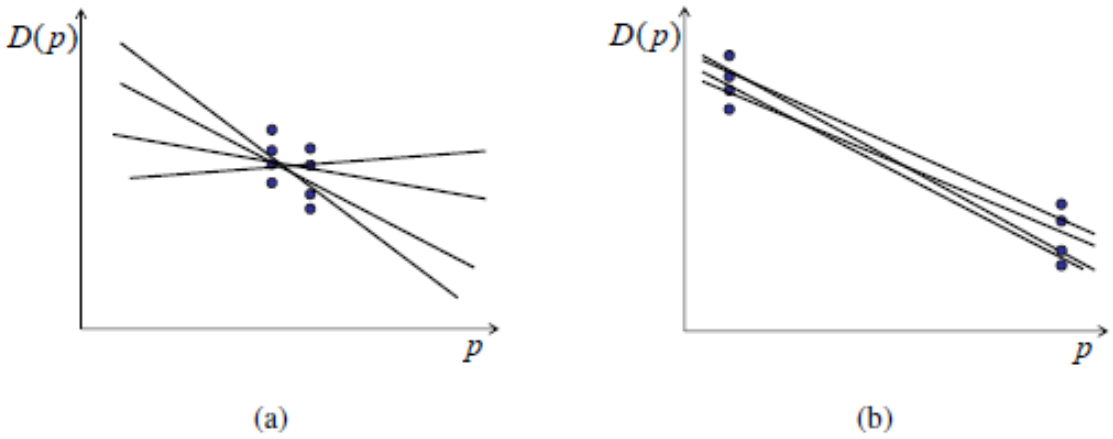


Figure 8.6 Estimating the demand function using (a) observations near the middle and (b) observations near the endpoints.

Policy function approximation

- Designing a policy:
  - » Just doing what appears to be best will encourage prices too close to the middle.
  - » We can try an *excitation* policy by adding noise:

$$p^n = P^\pi(S^n|\rho) = \frac{\bar{\theta}_0^n}{2\bar{\theta}_1^n} + \varepsilon$$

where $\varepsilon \sim N(0, \rho^2)$, where ρ is a tunable parameter.

  - » After charging price p^n, we observe revenue:

$$R^{n+1} = p^n\left(\theta_0 - \theta_1 p^n + \varepsilon^{demand,n+1}\right)$$

where $\varepsilon^{demand,n+1} \sim N(0, \sigma^{W,2})$ is the noise when observing demand.

Policy function approximation

- Designing a policy:
  - » ρ is a tunable parameter that needs to solve:

$$\max_{\rho} \mathbb{E}_{\theta} \mathbb{E}_{D^1, D^2, \ldots, D^N | \theta} \left\{ \sum_{n=1}^{N} R(P^{\pi}(S^n | \rho)) | S^0 \right\}$$

  - » The best value of ρ should strike a balance between doing a better job of exploration, without overdoing it.

Policy function approximation

- Other examples of PFAs:
  - » A basic linear decision rule

$$X^{\pi}(S^n|\rho) = \sum_{f \in F} \rho_f \phi_f(S^n)$$

  - » Where the features $\phi_f(S)$ for $f \in F$ have to be designed by hand.

Cost function approximation

Materials science example

Materials science application

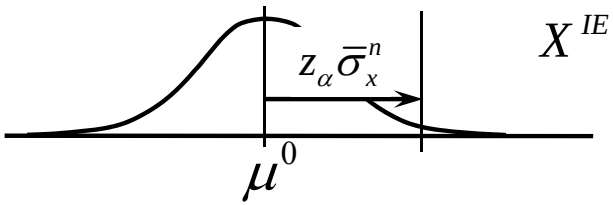
- Heuristic measurement policies
  - » **Boltzmann exploration**
    - Explore choice *x* with probability $P_x^n(\theta) = \frac{e^{\theta \bar{\mu}_x^n}}{\sum_{x'} e^{\theta \bar{\mu}_{x'}^n}}$
    - $X^{Boltz}(S^n|\theta) = \arg\max_x \{x | P_x^n(\theta) \le U\}.\ U \sim [0,1]$
  - » **Upper confidence bounding**

$$X^{UCB}(S^n \mid \theta^{UCB}) = \arg\max_x \left(\bar{\mu}_x^n + \theta^{UCB} \sqrt{\frac{\log n}{N_x^n}} \right)$$

  - » **Thompson sampling**

$$X^{TS}(S^n) = \arg\max_x \hat{\mu}_x^{n+1} \quad \text{where } \hat{\mu}_x^n \sim N(\bar{\mu}_x^n, \beta_x^n)$$

  - » **Interval estimation** (or upper confidence bounding)
    - Choose *x* which maximizes



$$X^{IE}(S^n \mid \theta^{IE}) = \arg\max_x \bar{\mu}_x^n + \theta^{IE} \bar{\sigma}_x^n$$

Materials science application

- Notes:
  - » Each one of these policies has an imbedded "max" (or "min") operator.
  - » But each one still has a tunable parameter – this is the distinguishing feature of "policy search" policies.
  - » For these policies, the max involves a simple search over a set of discrete choices, so this is not too difficult.
  - » Another example is solving a shortest path into New York. The tunable parameter might be how much time you add to the trip to deal with traffic.
  - » The max/min could also be a large optimization problem, such as what is solved to plan energy generation for tomorrow.

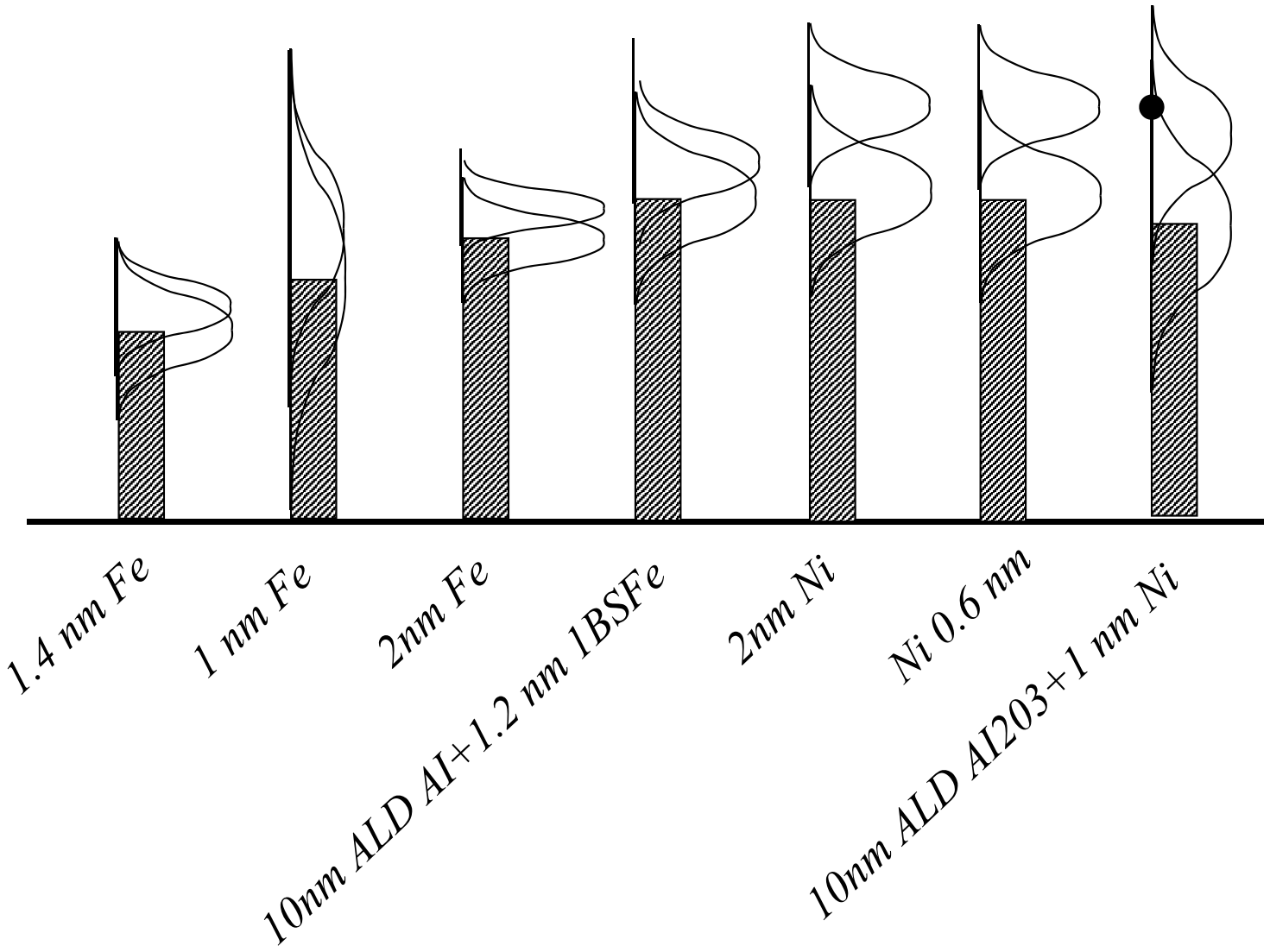
Materials science application

- Lookup table
  - » We can organize potential catalysts into groups
  - » Scientists using domain knowledge can estimate correlations in experiments between similar catalysts.

	1.4 nm Fe	1 nm Fe	2nm Fe	10nm ALD Al2O3+1.2 nm IBS Fe	2 nm Ni	Ni 0.6 nm	10nm ALD Al2O3+1 nm Ni
1.4 nm Fe	1	0.7	0.7	0.6	0.4	0.4	0.2
1 nm Fe	0.7	1	0.7	0.6	0.4	0.4	0.2
2nm Fe	0.7	0.7	1	0.6	0.4	0.4	0.2
10nm ALD Al2O3+1.2 nm IBS Fe	0.6	0.6	0.6	1	1	0.3	0
2 nm Ni	0.4	0.4	0.4	1	1	0.7	0.6
Ni 0.6 nm	0.4	0.4	0.4	0.3	0.7	1	0.6
10nm ALD Al2O3+1 nm Ni	0.2	0.2	0.2	0	0.6	0.6	1

Materials science application

- Correlated beliefs: Testing one material teaches us about other materials



Materials science application

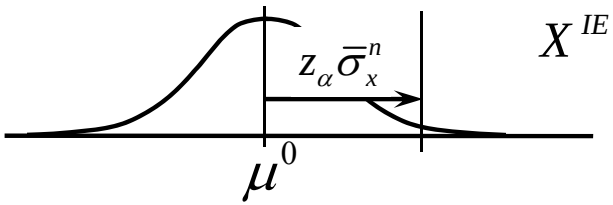
- Heuristic measurement policies
  - » **Boltzmann exploration**
    - Explore choice x with probability $P_x^n(\theta) = \frac{e^{\theta \bar{\mu}_x^n}}{\sum_{x'} e^{\theta \bar{\mu}_{x'}^n}}$
    - $X^{Boltz}(S^n|\theta) = \arg\max_x \{x | P_x^n(\theta) \leq U\}.\ U \sim [0,1]$
  - » **Upper confidence bounding**

$$X^{UCB}(S^n \mid \theta^{UCB}) = \arg\max_x \left(\bar{\mu}_x^n + \theta^{UCB} \sqrt{\frac{\log n}{N_x^n}} \right)$$

  - » **Thompson sampling**

$$X^{TS}(S^n) = \arg\max_x \hat{\mu}_x^{n+1} \quad \text{where } \hat{\mu}_x^n \sim N(\bar{\mu}_x^n, \beta_x^n)$$

  - » **Interval estimation**
    - Choose x which maximizes



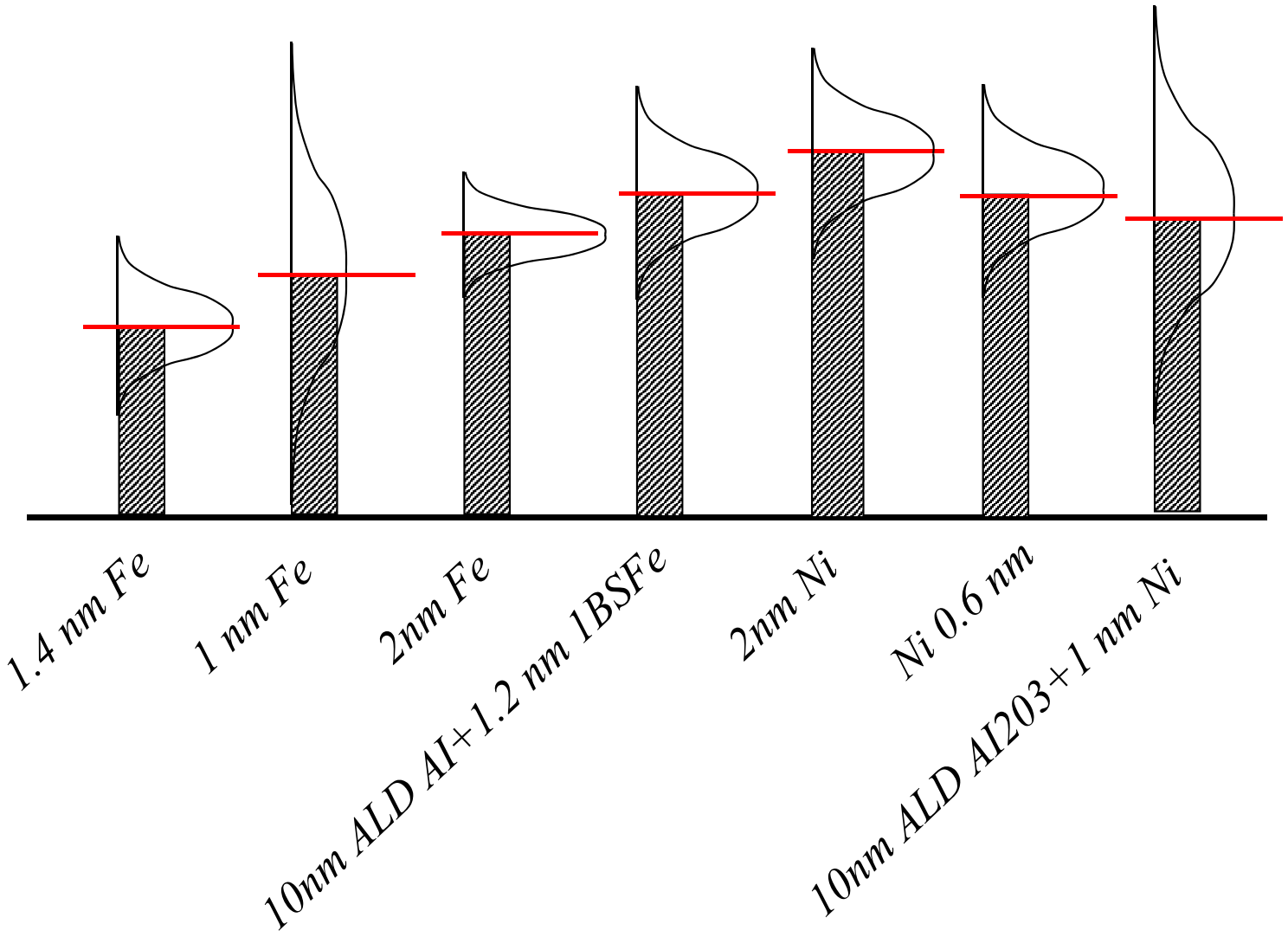
$$X^{IE}(S^n \mid \theta^{IE}) = \arg\max_x \bar{\mu}_x^n + \theta^{IE} \bar{\sigma}_x^n$$

Materials science application

- Notes:
  - » Upper confidence bounding and Thompson sampling are very popular in the computer science community.
  - » These have been shown to have provable regret bounds, which the CS community then uses to claim that they must be very good.
  - » My own experimental work has not supported this claim, but we have found that a properly tuned version of interval estimation tends to work surprisingly well.
  - » … The problem is the tuning.

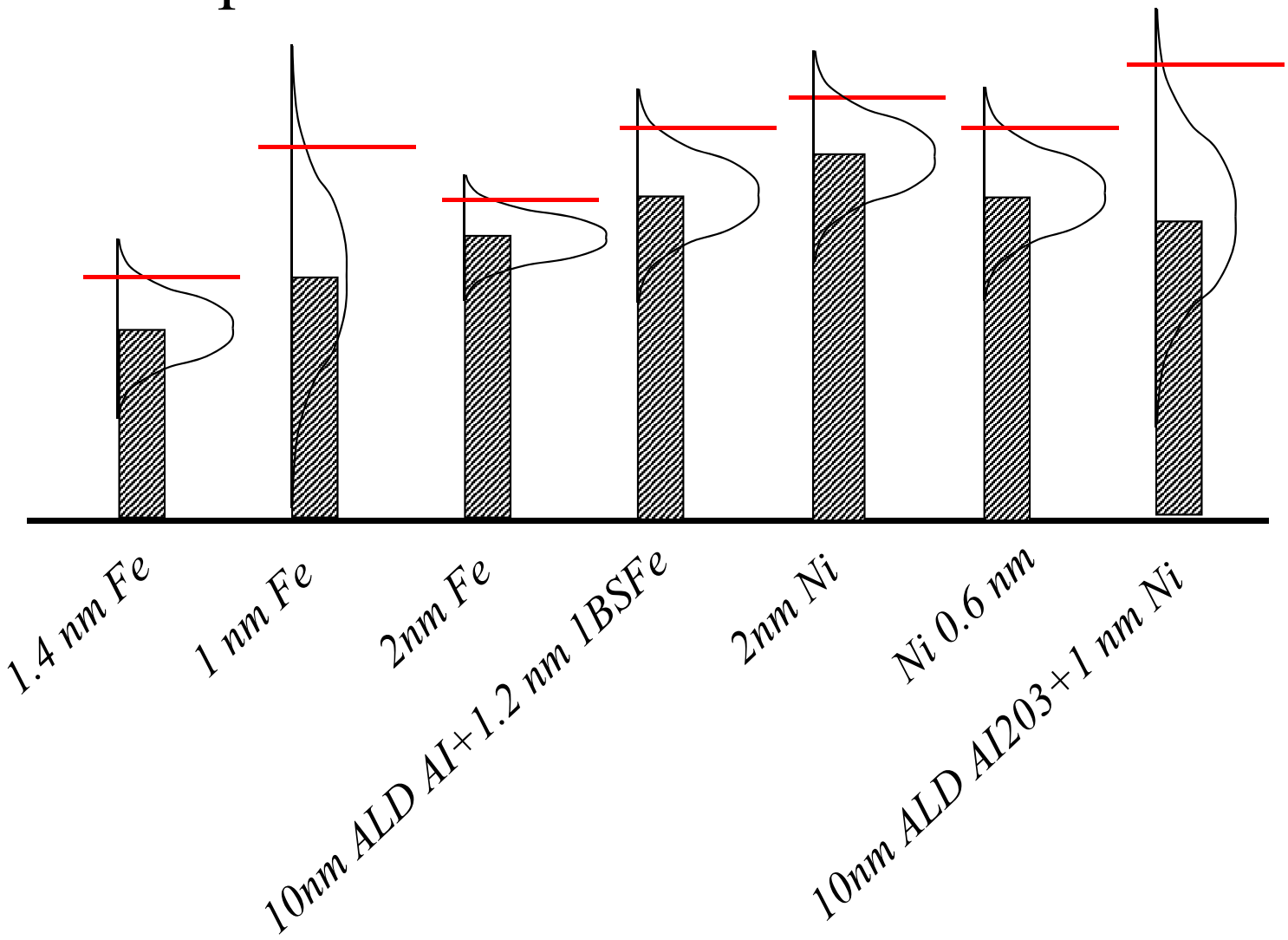
Materials science application

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



Materials science application

- Picking $\theta^{IE} = 2$ means we are evaluating each choice at the 95th percentile.



Materials science application

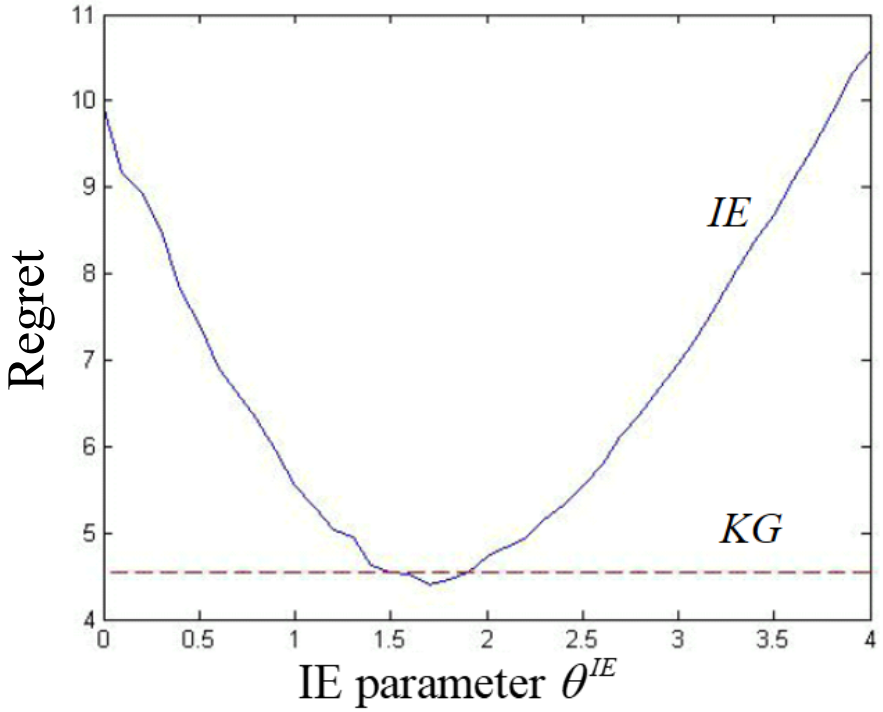
- Optimizing the policy
  - » We optimize θ^{IE} to maximize:

$$\max_{\theta^{IE}} F(\theta^{IE}) = \mathbb{E}F\left(x^{\pi,N}, W\right)$$

where

$$x^n = X^{IE}(S^n \mid \theta^{IE}) = \arg\max_x \left(\bar{\mu}_x^n + \theta^{IE}\bar{\sigma}_x^n\right) \qquad S^n = (\bar{\mu}_x^n, \bar{\sigma}_x^n)$$

- Notes:
  - » This can handle any belief model, including correlated beliefs, nonlinear belief models.
  - » All we require is that we be able to simulate a policy.



Materials science application

- Other applications
  - » Airlines optimizing schedules with schedule slack to handle weather uncertainty.
  - » Manufacturers using buffer stocks to hedge against production delays and quality problems.
  - » Grid operators scheduling extra generation capacity in case of outages.
  - » Adding time to a trip planned by Google maps to account for uncertain congestion.

Materials science application

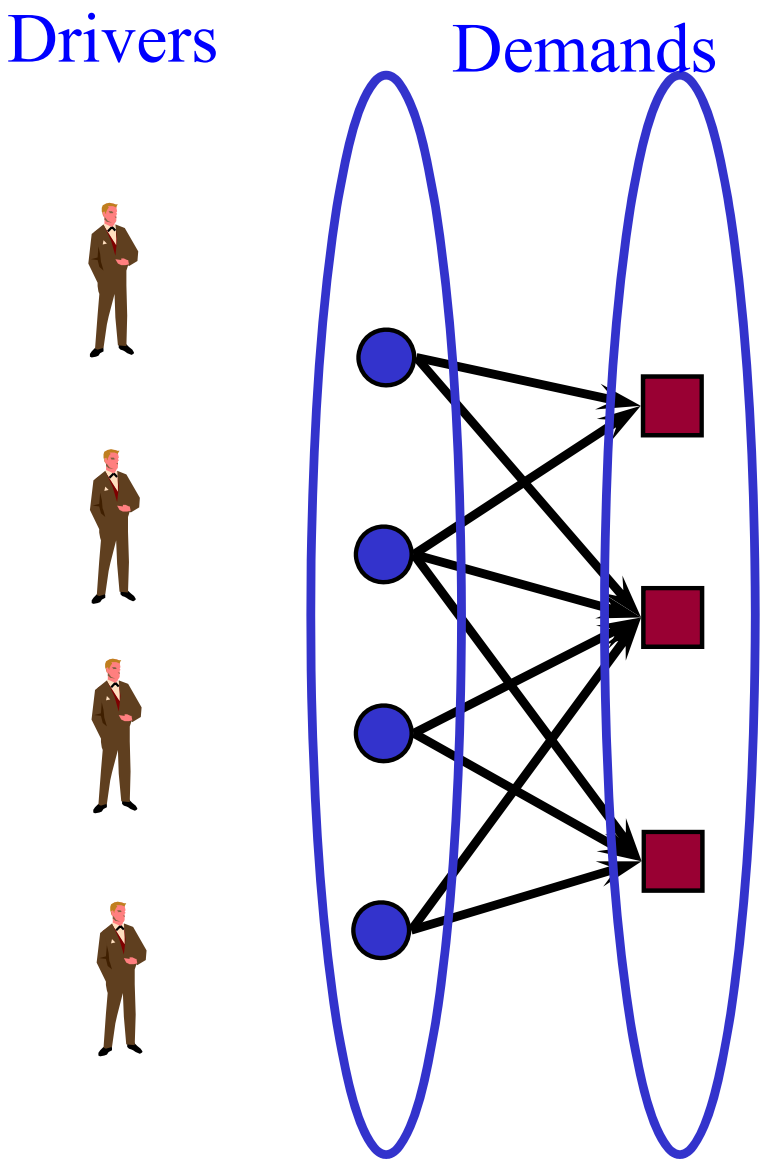
- Notes:
  - » CFA policies are exceptionally popular in internet applications:
  - » Choosing ads, news articles, products, … that maximize ad-clicks.
  - » The ease of computation of the policies is very attractive.
  - » Representative from google once noted: “We can use any policy that can be computed in under 50 milliseconds.”

Cost function approximation

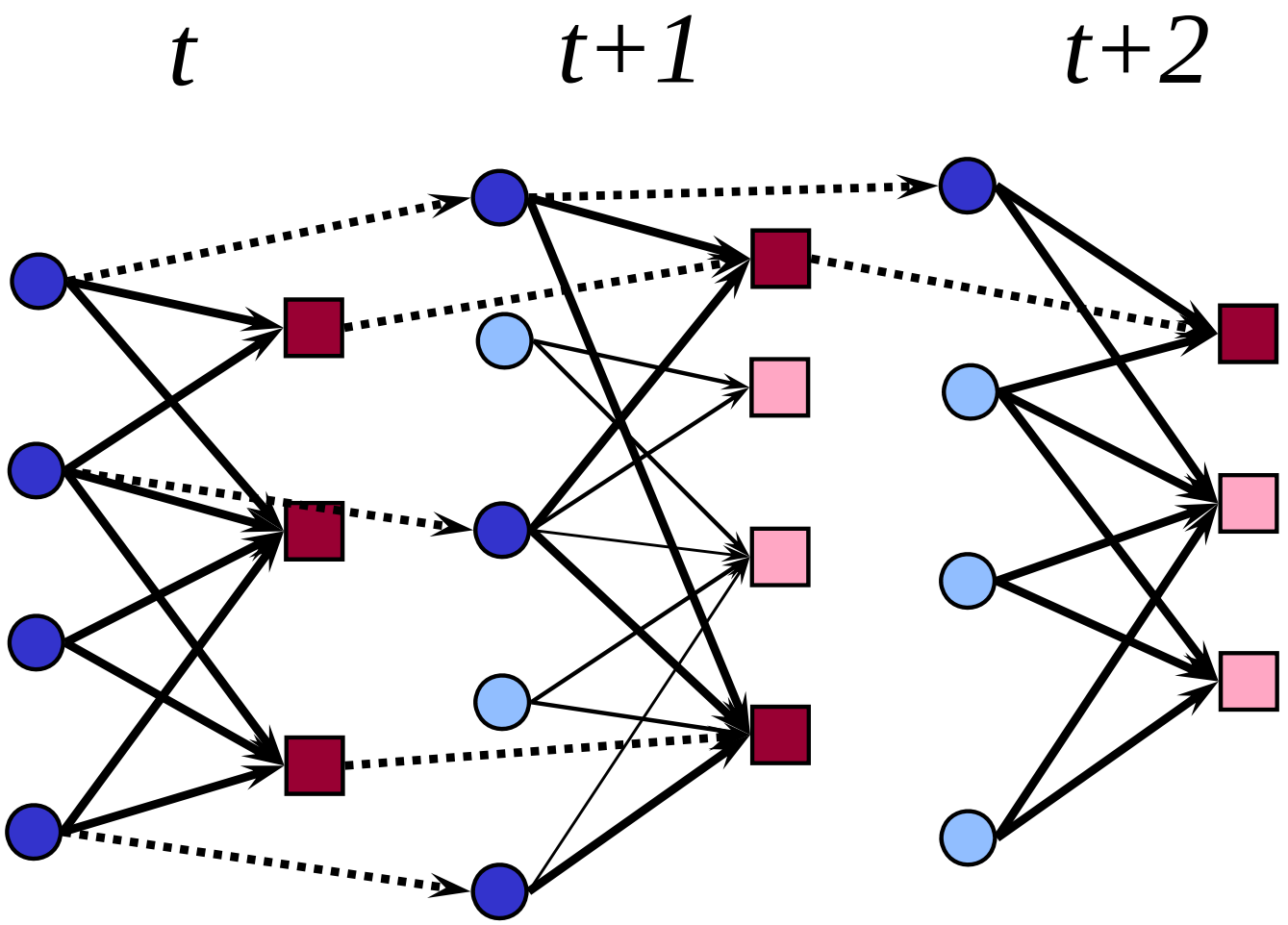
Fleet management example

Schneider National

Fleet management application



Fleet management application



The assignment of drivers to loads evolves over time, with new loads being called in, along with updates to the status of a driver.

Fleet management application

- A purely myopic policy would solve this problem using

$$\min_x \sum_d \sum_l c_{tdl} x_{tdl}$$

where

$$x_{tdl} = \begin{cases} 1 & \text{If we assign driver } d \text{ to load } l \\ 0 & \text{Otherwise} \end{cases}$$

$$c_{tdl} = \text{Cost of assigning driver } d \text{ to load } l \text{ at time } t$$

What if a load it not assigned to any driver, and has been delayed for a while? This model ignores the fact that we eventually have to assign someone to the load.

Fleet management application

- We can minimize delayed loads by solving a modified objective function:

$$\min_x \sum_d \sum_l \left(c_{tdl} - \theta \tau_{tl} \right) x_{tdl}$$

where

τ_{tl} = How long load l has been delayed by time t

θ = Bonus for moving a delayed load

We refer to our modified objective function as a *cost function approximation*.

Fleet management application

- We now have to tune our policy, which we define as:

$$X^{\pi}(S_t \mid \theta) = \arg\min_x \underbrace{\sum_d \sum_l \left(c_{tdl} - \theta\tau_{tl}\right) x_{tdl}}_{\bar{C}^{\pi}(S_t, x_t \mid \theta)}$$

We can now optimize θ, another form of *policy search*, by solving

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

Cost function approximation

Logistics example

Logistics application

- 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

Logistics application

- 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

$$X_t^{\pi}(S_t) = \arg\max_{x_t} \sum_{p \in P} c_p x_{tp}$$

subject to

$$\left.\begin{aligned} \sum_{p \in P} x_{tp} &\geq D_t \\ x_{tp} &\leq u_p \\ x_{tp} &\geq 0 \end{aligned}\right\} \mathcal{X}_t$$

» This assumes our demand forecast D_t is accurate.

Logistics application

- 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

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

subject to

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

$$x_{tp} \leq u_p$$

$$x_{tp} \geq \theta^{\text{buffer}}$$

$\mathcal{X}_t^\pi(\theta)$

Reserve

Buffer stock

» This is a "parametric cost function approximation"

Logistics application

- An even more general CFA model:
  - » Define our policy:

$$X_t^\pi(\theta) = \arg\max_x \bar{C}^\pi(S_t, x_t \mid \theta)$$

Parametrically modified costs

subject to

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

Parametrically modified constraints

  - » We tune θ by optimizing:

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

Cost function approximation

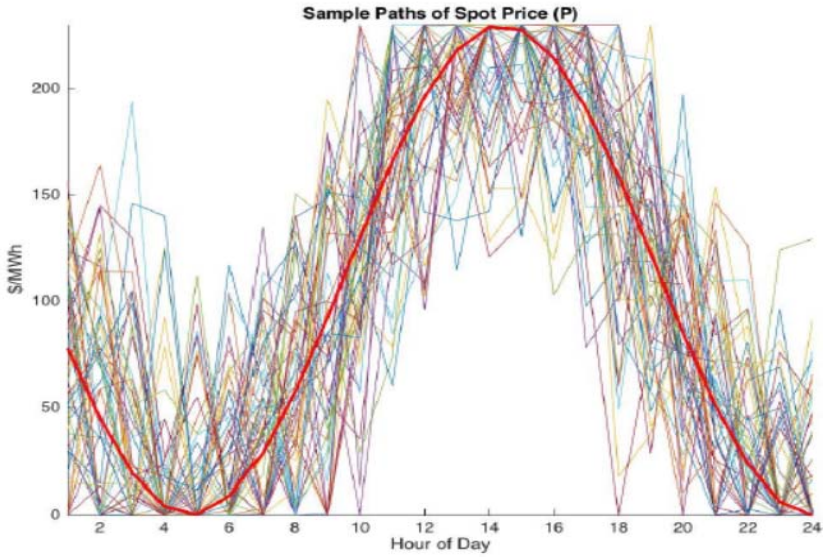
Energy storage example

Parametric cost function approximation

- Notes:
  - » In this set of slides, we are going to illustrate the use of a parametrically modified lookahead policy.
  - » This is designed to handle a nonstationary problem with rolling forecasts. This is just what you do when you plan your path with google maps.
  - » The lookahead policy is modified with a set of parameters that factor the forecasts. These parameters have to be tuned using the basic methods of policy search.

Parametric cost function approximation

- An energy storage problem:



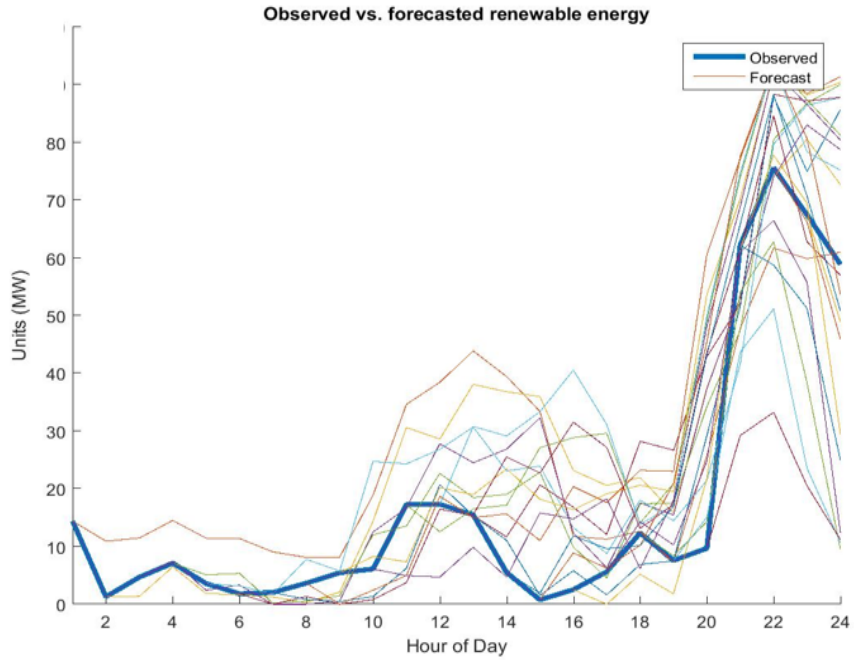
The state of the system can be represented by

r,

$$S_t = (R_t, E_t, P_t, D_t, G_t)$$

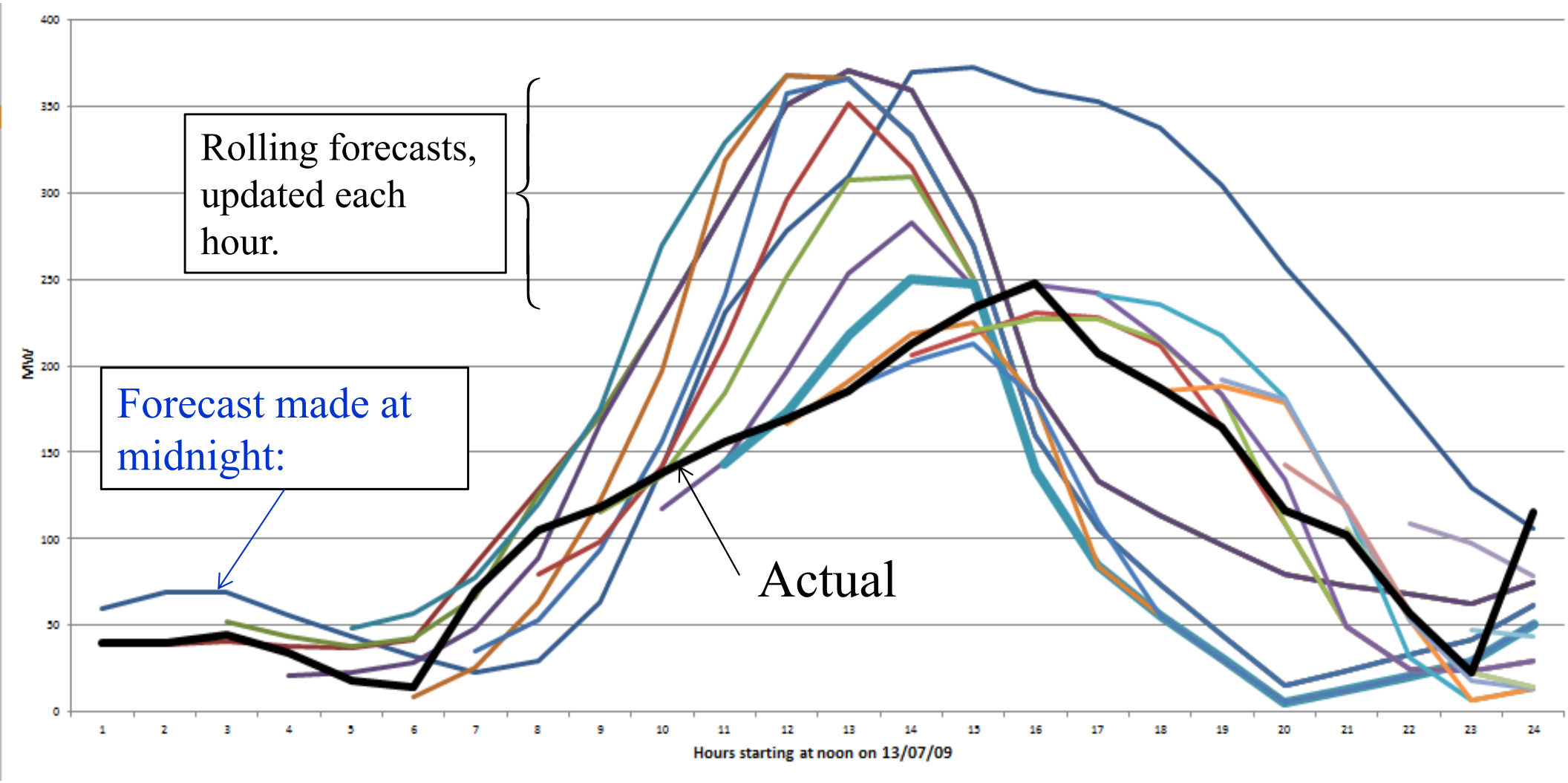
where

- $R_t \in [0, R_{\max}]$ is the level of energy in storage at time t
- E_t is the amount of energy available from wind
- P_t is the spot price of electricity
- D_t is the power demand
- G_t is the energy available from the grid



Parametric cost function approximation

- Forecasts evolve over time as new information arrives:



Parametric cost function approximation

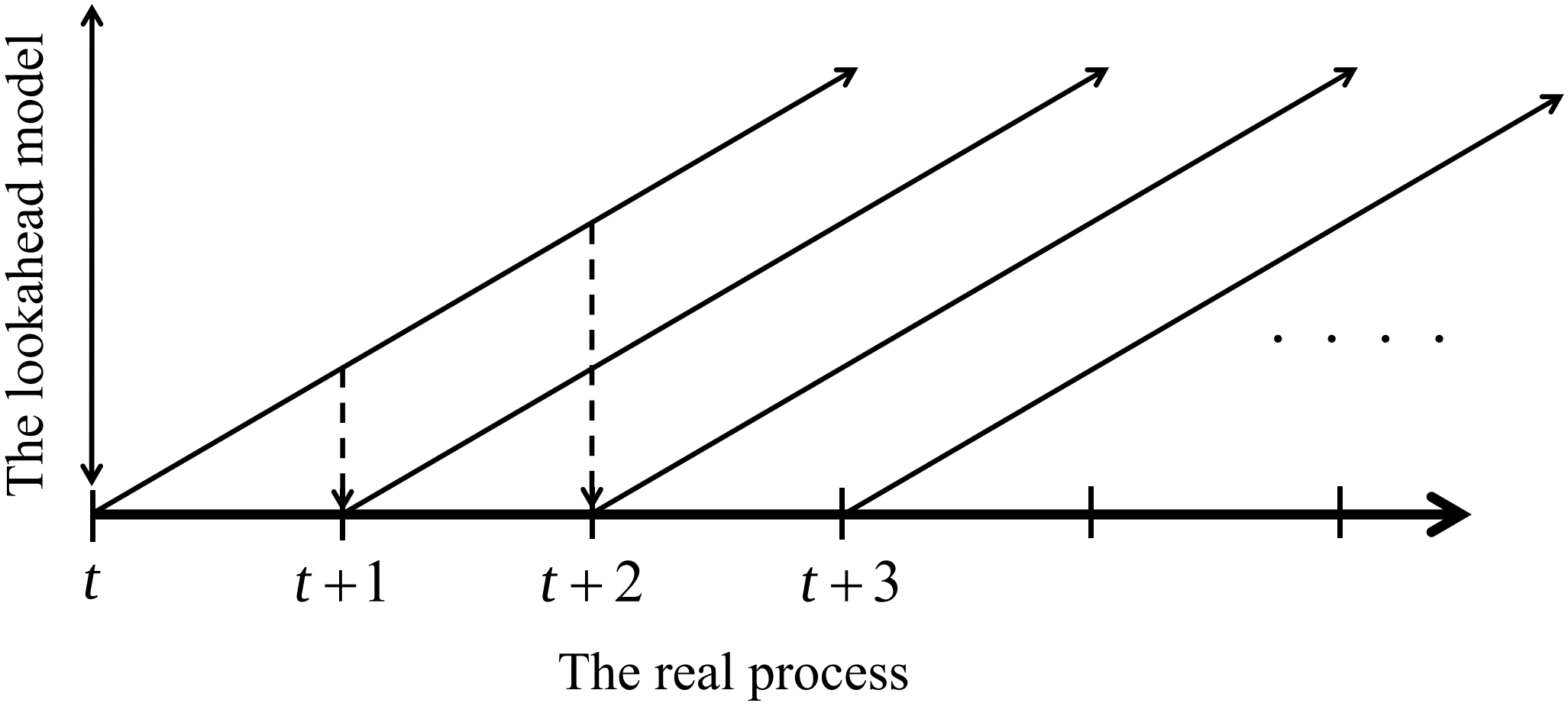
- Benchmark policy – Deterministic lookahead

$$X_t^{\text{D-LA}}(S_t) = \underset{x_t,(\tilde{x}_{tt'},t'=t+1,\ldots,t+H)}{\text{argmin}} \left(C(S_t,x_t) + \left[\sum_{t'=t+1}^{t+H} \tilde{c}_{tt'}\tilde{x}_{tt'} \right] \right)$$

$$\begin{aligned}
\tilde{x}_{tt'}^{wd} + \beta\tilde{x}_{tt'}^{rd} + \tilde{x}_{tt'}^{gd} &\le f_{tt'}^{D} \\
\tilde{x}_{tt'}^{gd} + \tilde{x}_{tt'}^{gr} &\le f_{tt'}^{G} \\
\tilde{x}_{tt'}^{rd} + \tilde{x}_{tt'}^{rg} &\le \tilde{R}_{tt'} \\
\tilde{x}_{tt'}^{wr} + \tilde{x}_{tt'}^{gr} &\le R^{\max} - \tilde{R}_{tt'} \\
\tilde{x}_{tt'}^{wr} + \tilde{x}_{tt'}^{wd} &\le f_{tt'}^{E} \\
\tilde{x}_{tt'}^{wr} + \tilde{x}_{tt'}^{gr} &\le \gamma^{charge} \\
\tilde{x}_{tt'}^{rd} + \tilde{x}_{tt'}^{rg} &\le \gamma^{discharge}
\end{aligned}$$

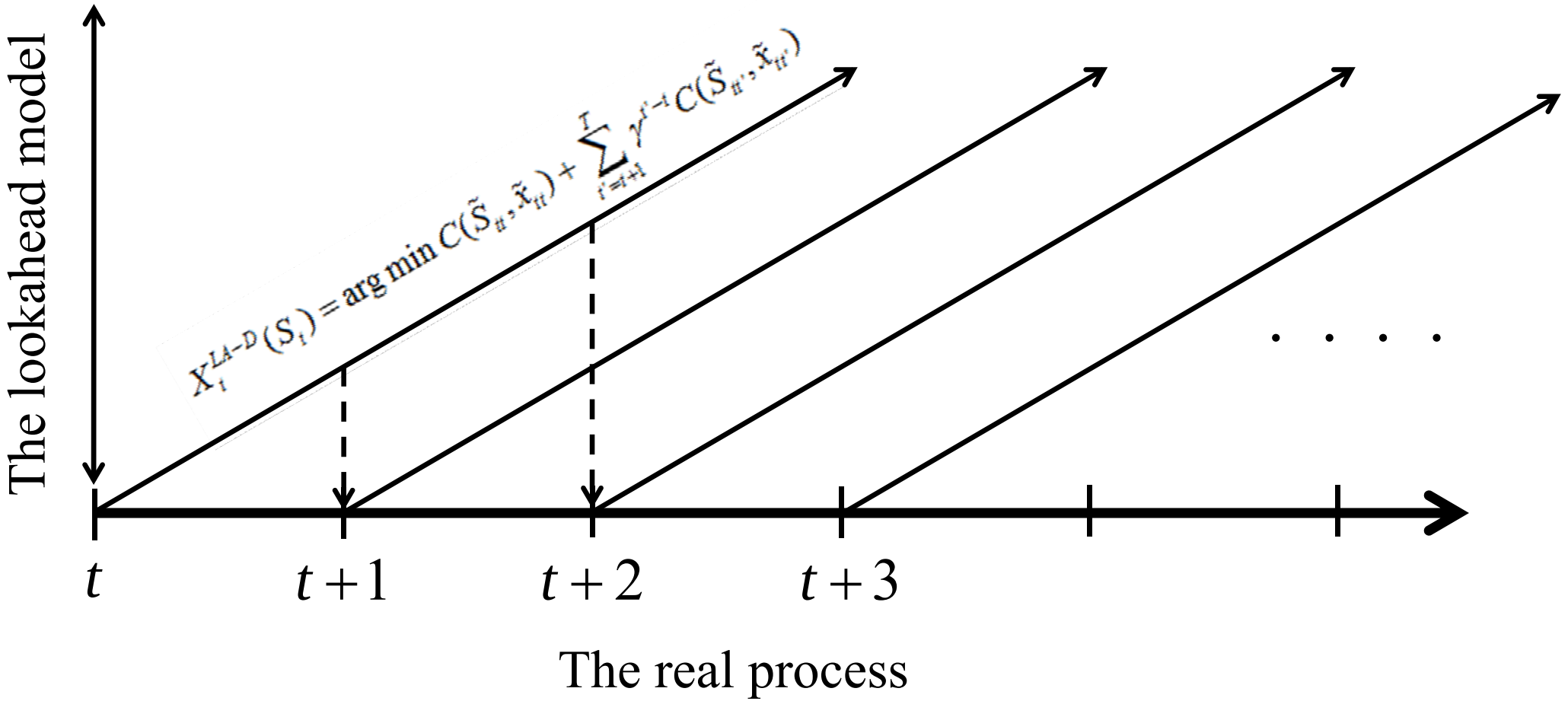
Lookahead policies

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



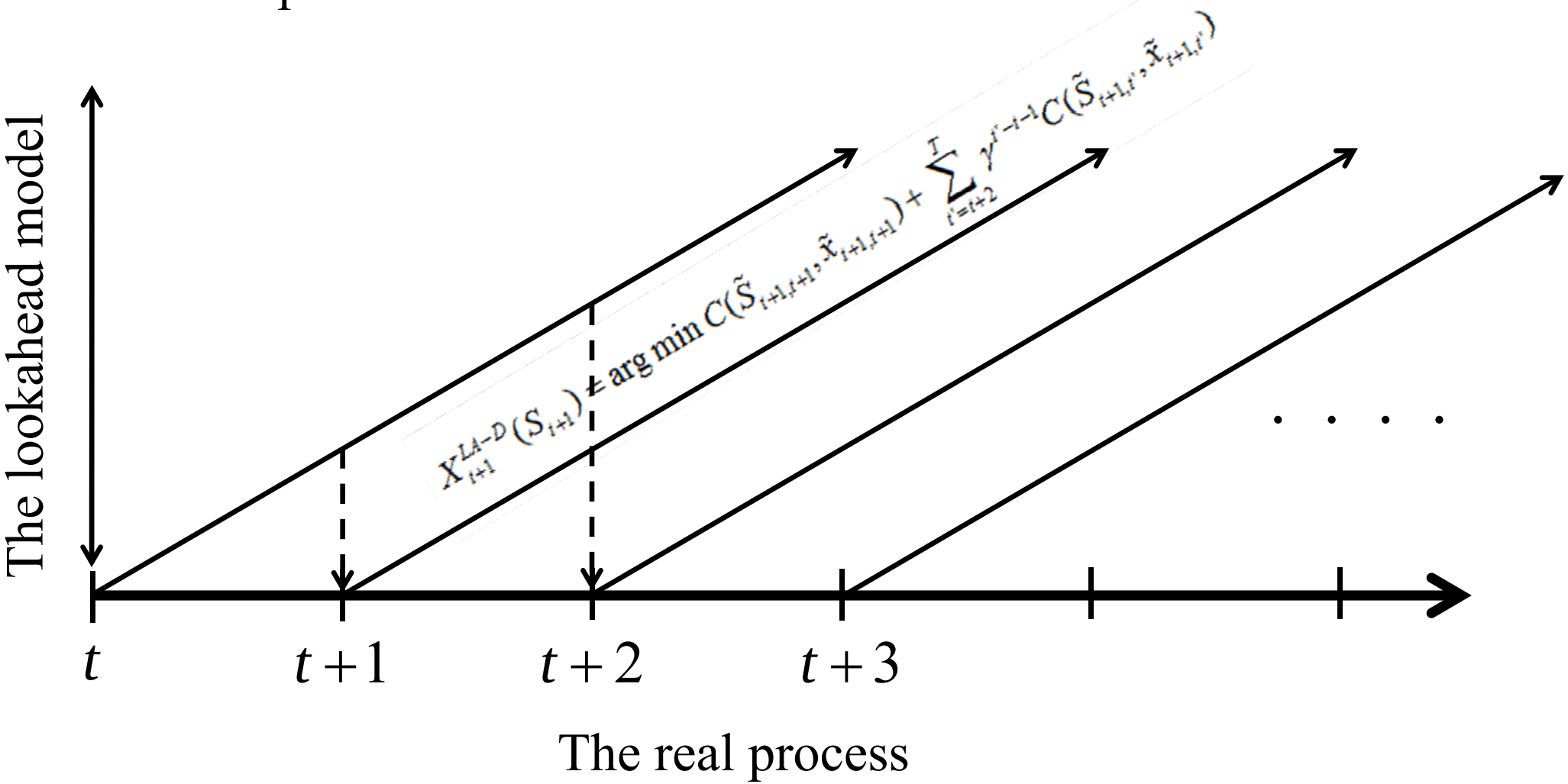
Lookahead policies

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



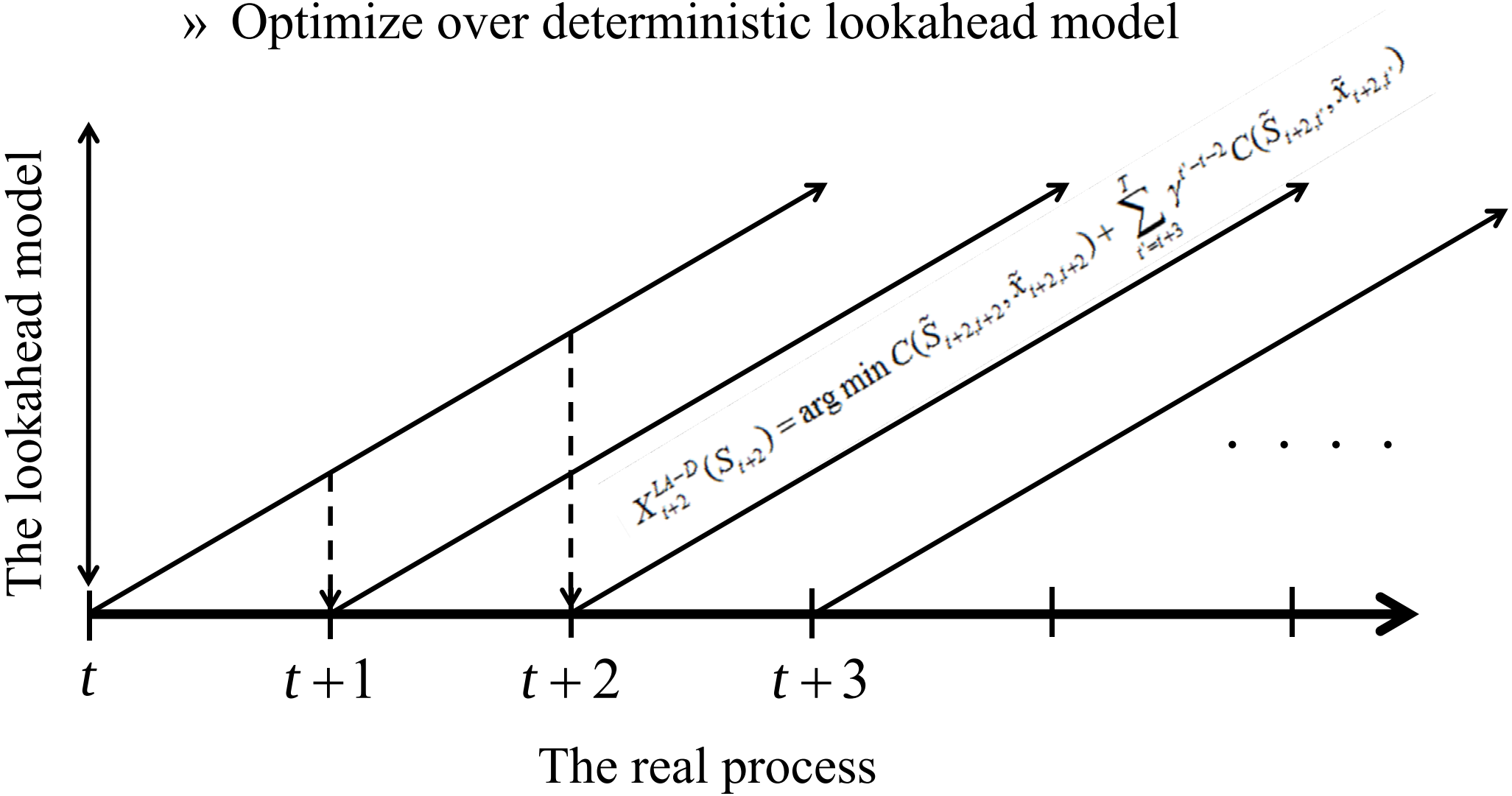
Lookahead policies

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



Lookahead policies

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



Parametric cost function approximation

- Benchmark policy – Deterministic lookahead

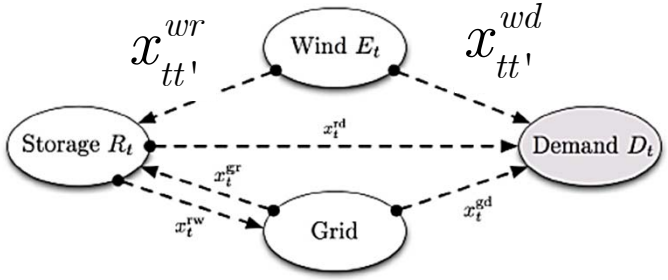
$$X_t^{\text{D-LA}}(S_t) = \underset{x_t,(\tilde{x}_{tt'},t'=t+1,\ldots,t+H)}{\text{argmin}} \left(C(S_t,x_t) + \left[\sum_{t'=t+1}^{t+H} \tilde{c}_{tt'}\tilde{x}_{tt'} \right] \right)$$

$$\begin{aligned}
\tilde{x}_{tt'}^{wd} + \beta\tilde{x}_{tt'}^{rd} + \tilde{x}_{tt'}^{gd} &\le f_{tt'}^{D} \\
\tilde{x}_{tt'}^{gd} + \tilde{x}_{tt'}^{gr} &\le f_{tt'}^{G} \\
\tilde{x}_{tt'}^{rd} + \tilde{x}_{tt'}^{rg} &\le \tilde{R}_{tt'} \\
\tilde{x}_{tt'}^{wr} + \tilde{x}_{tt'}^{gr} &\le R^{\max} - \tilde{R}_{tt'} \\
\tilde{x}_{tt'}^{wr} + \tilde{x}_{tt'}^{wd} &\le f_{tt'}^{E} \\
\tilde{x}_{tt'}^{wr} + \tilde{x}_{tt'}^{gr} &\le \gamma^{charge} \\
\tilde{x}_{tt'}^{rd} + \tilde{x}_{tt'}^{rg} &\le \gamma^{discharge}
\end{aligned}$$

Parametric cost function approximation

- Parametric cost function approximations
  - » Replace the constraint

$$\tilde{x}^{wr}_{tt'} + \tilde{x}^{wd}_{tt'} \leq f^{E}_{tt'}$$



  with:

  - » Lookup table modified forecasts (one adjustment term for each time $\tau = t' - t$ in the future):

$$x^{wr}_{tt'} + x^{wd}_{tt'} \leq \theta_{t'-t} f^{E}_{tt'}$$

  - » We can simulate the performance of a parameterized policy

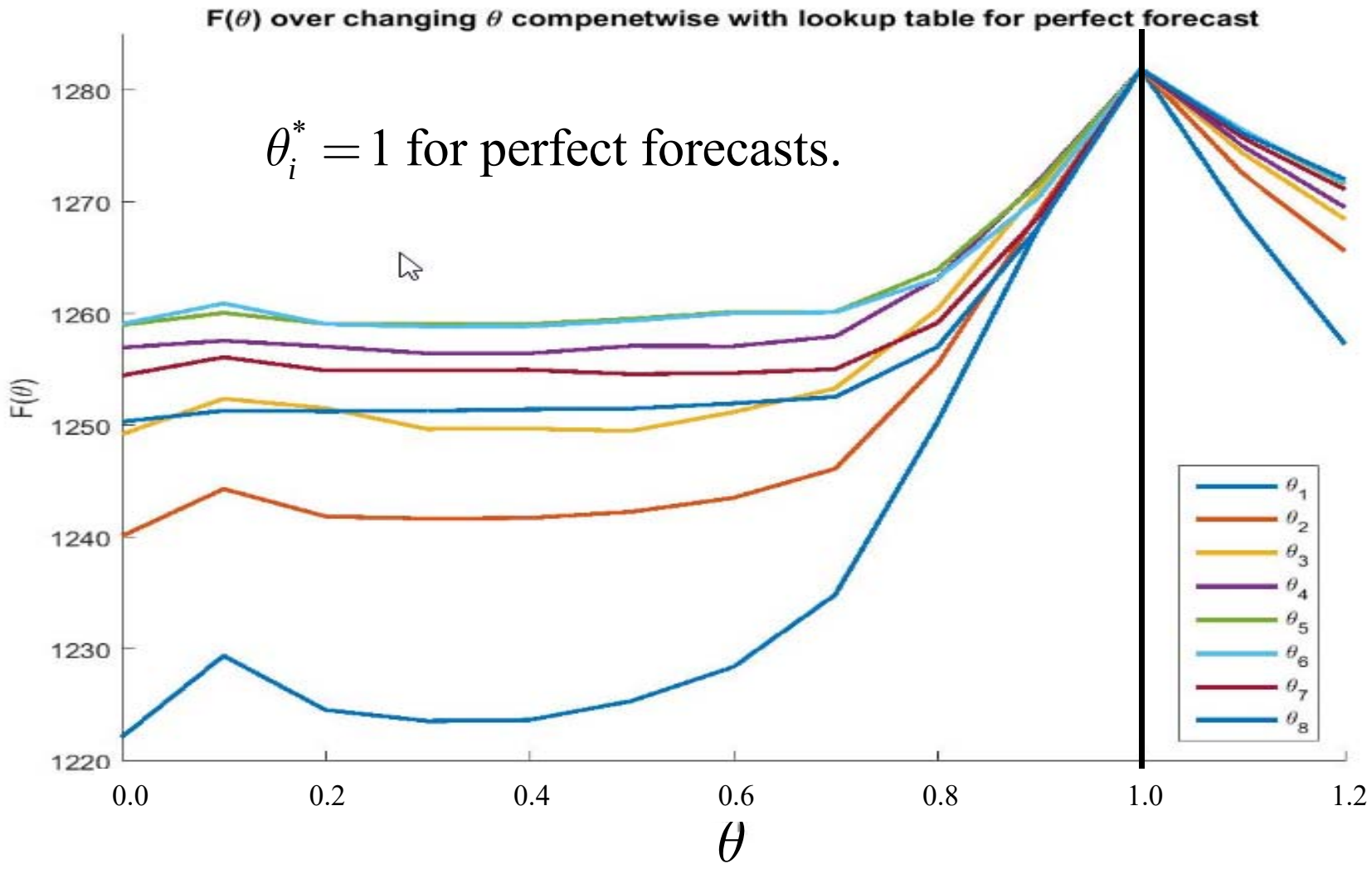
$$\bar{F}(\theta, \omega) = \sum_{t=0}^{T} C\left(S_t(\omega), X^{\pi}_t(S_t(\omega) \mid \theta)\right)$$

  - » The challenge is to optimize the parameters:

$$\min_{\theta} \mathbb{E} \sum_{t=0}^{T} C\left(S_t, X^{\pi}_t(S_t \mid \theta)\right)$$

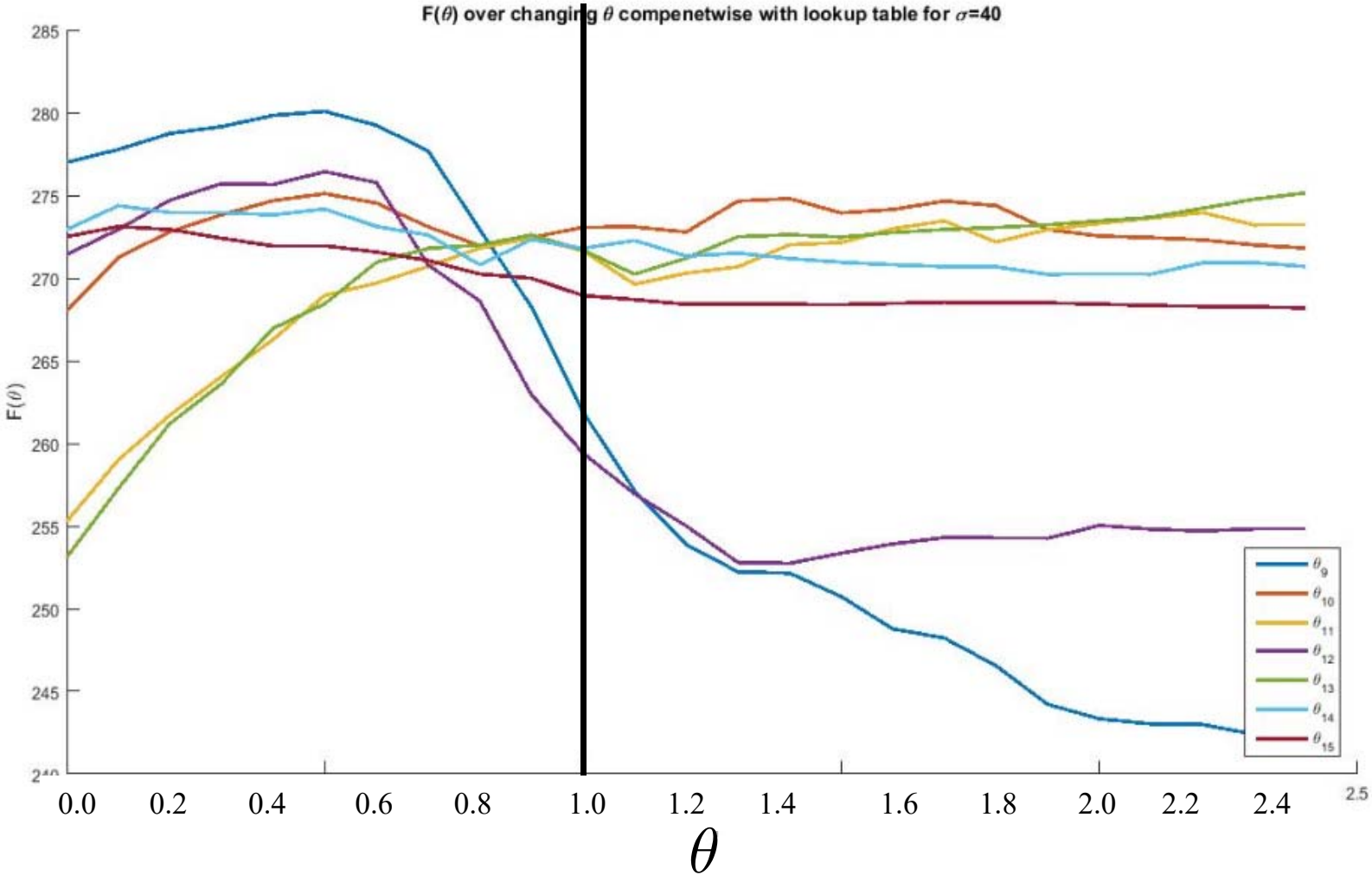
Parametric cost function approximation

- One-dimensional contour plots – perfect forecast
  - » θ_i for i=1,…, 8 hours into the future.

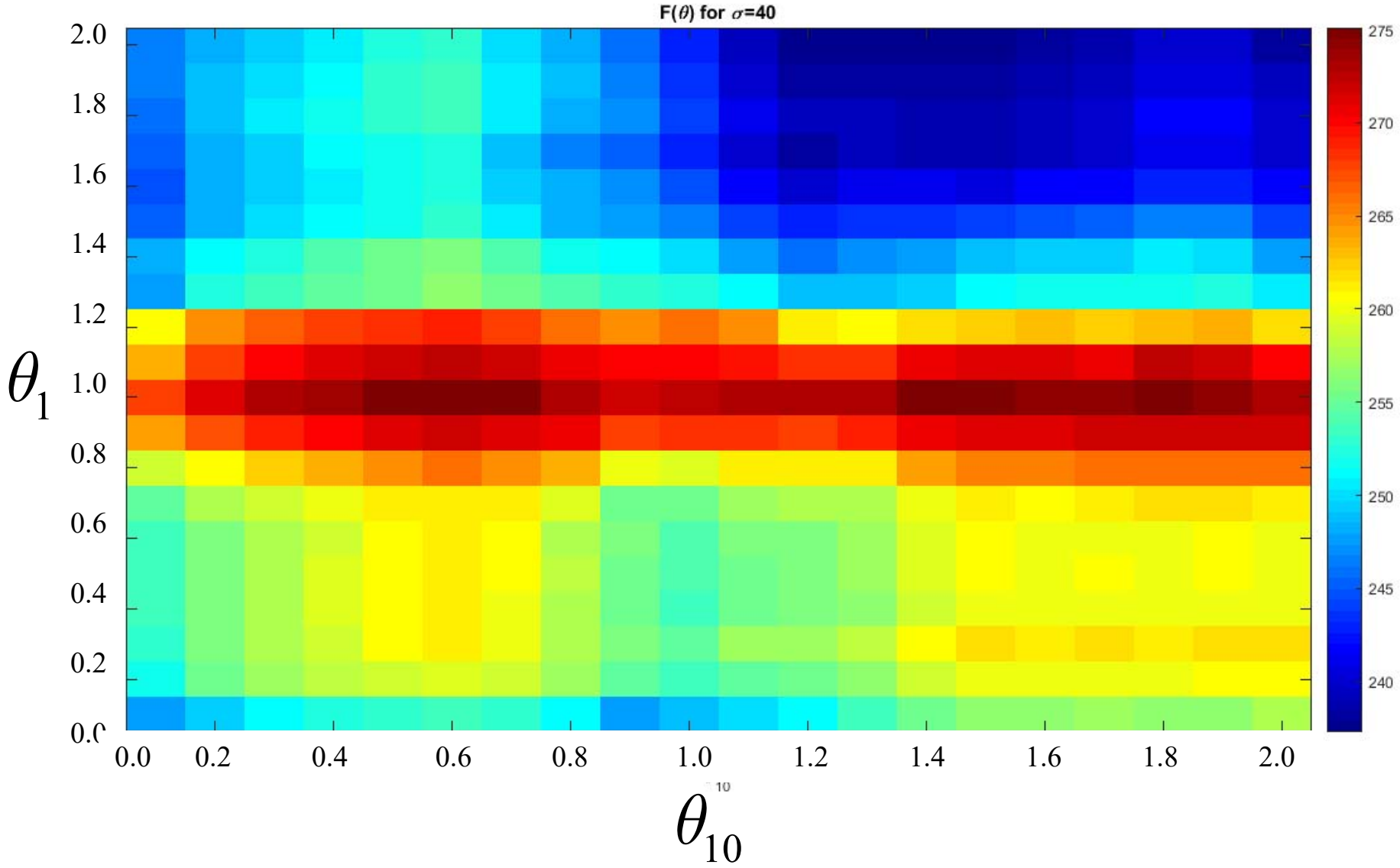


Parametric cost function approximation

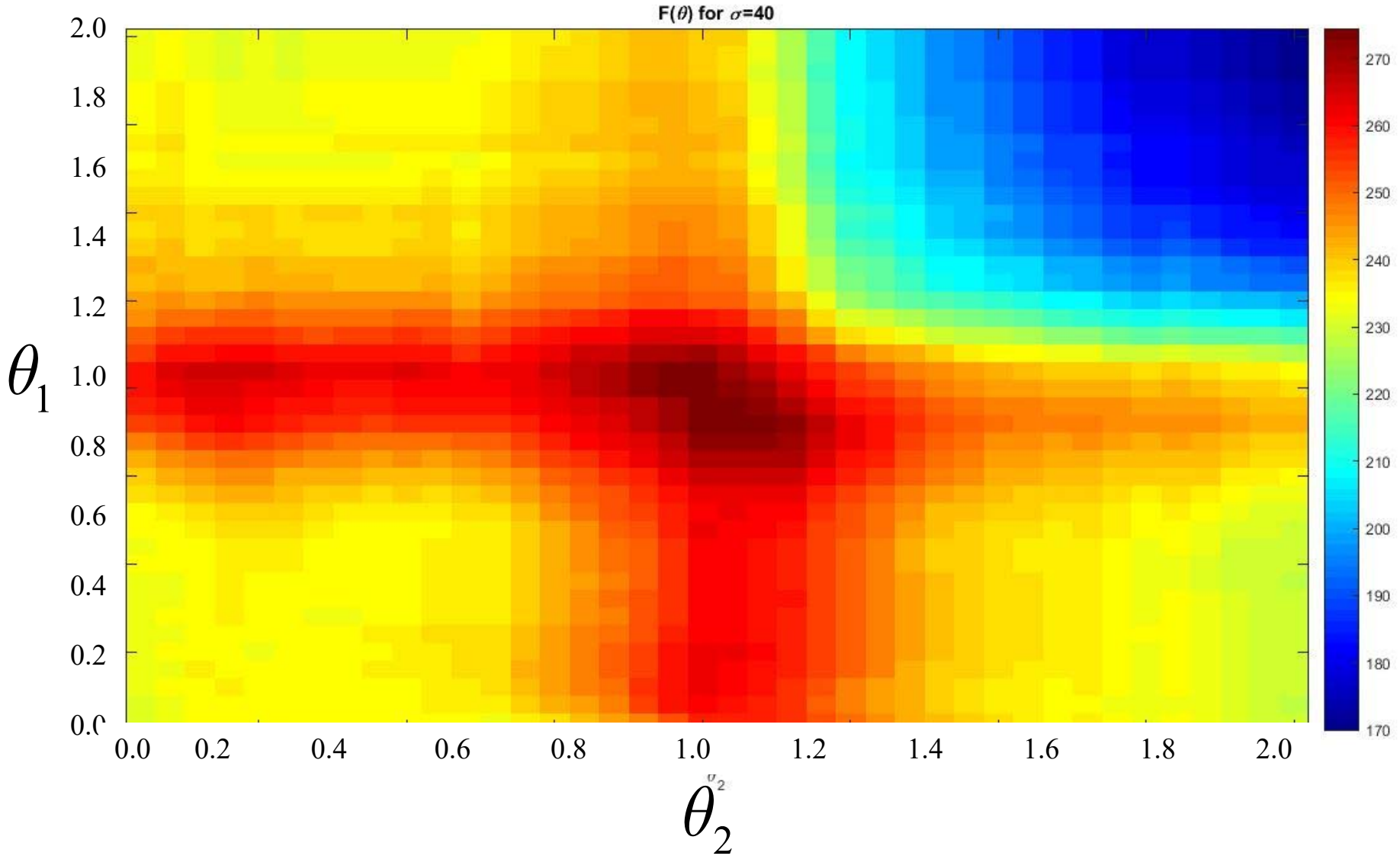
- One-dimensional contour plots-uncertain forecast
  - » θ_i for i=9,…, 15 hours into the future



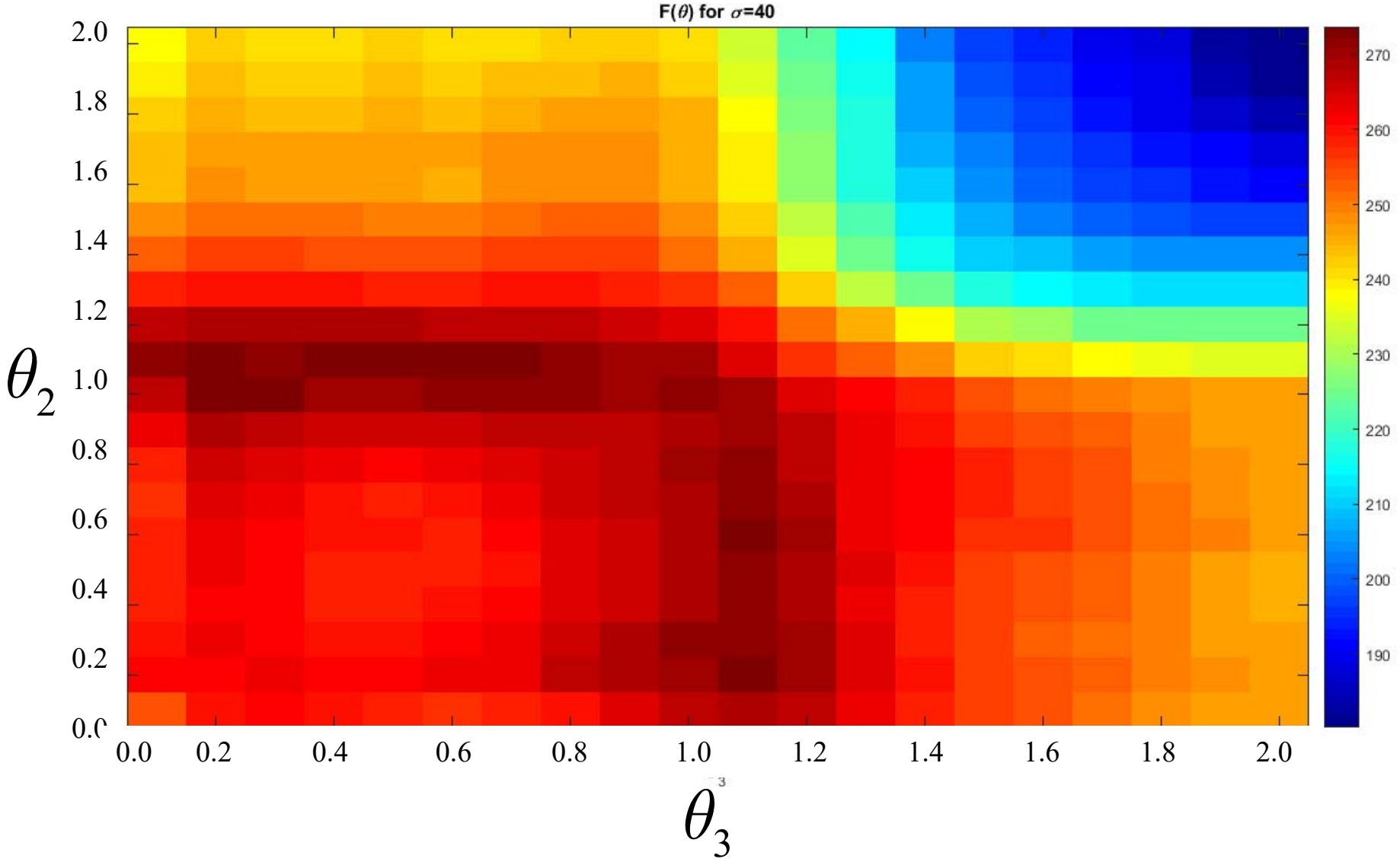
Parametric cost function approximation



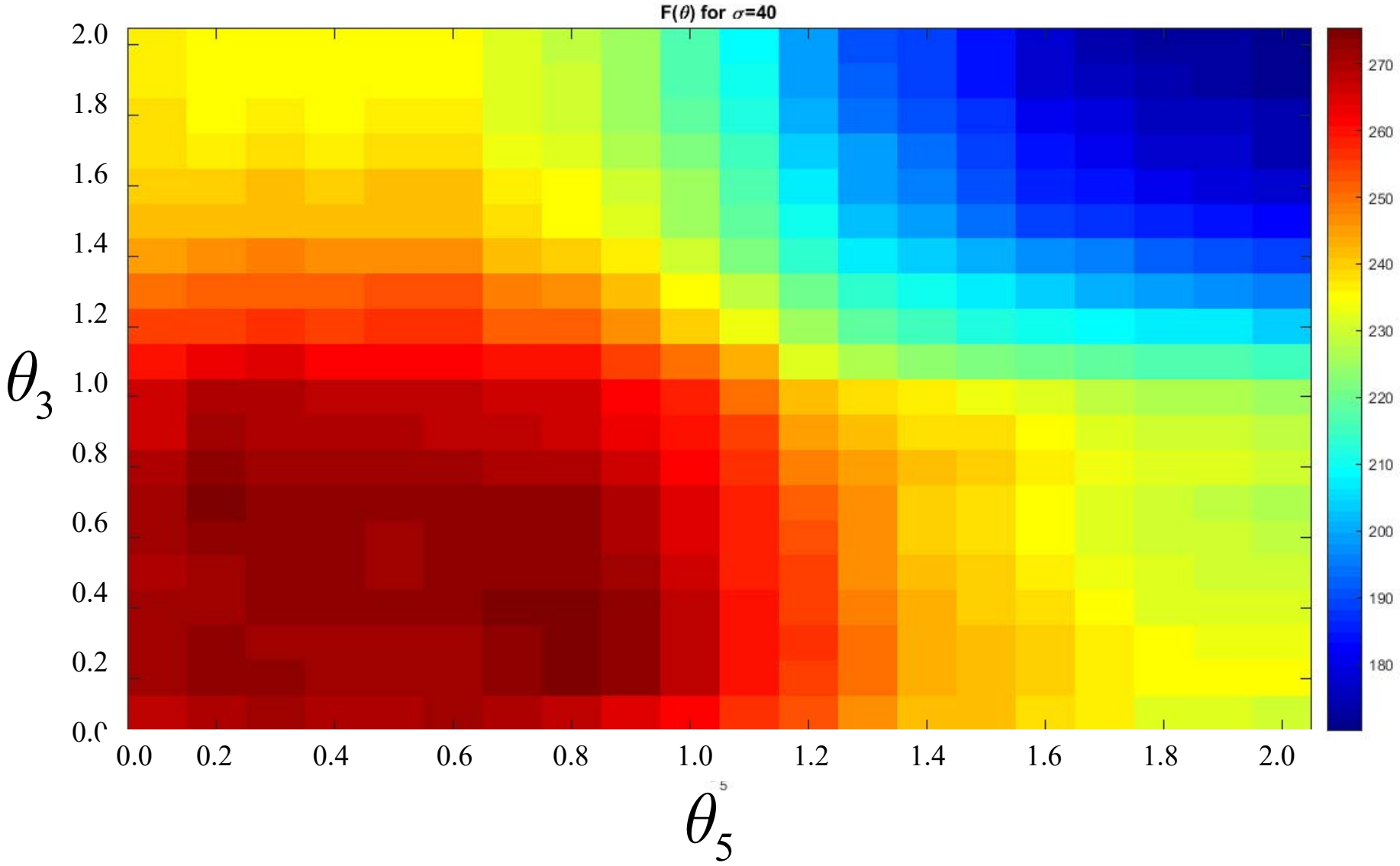
Parametric cost function approximation



Parametric cost function approximation



Parametric cost function approximation



Numerical derivatives

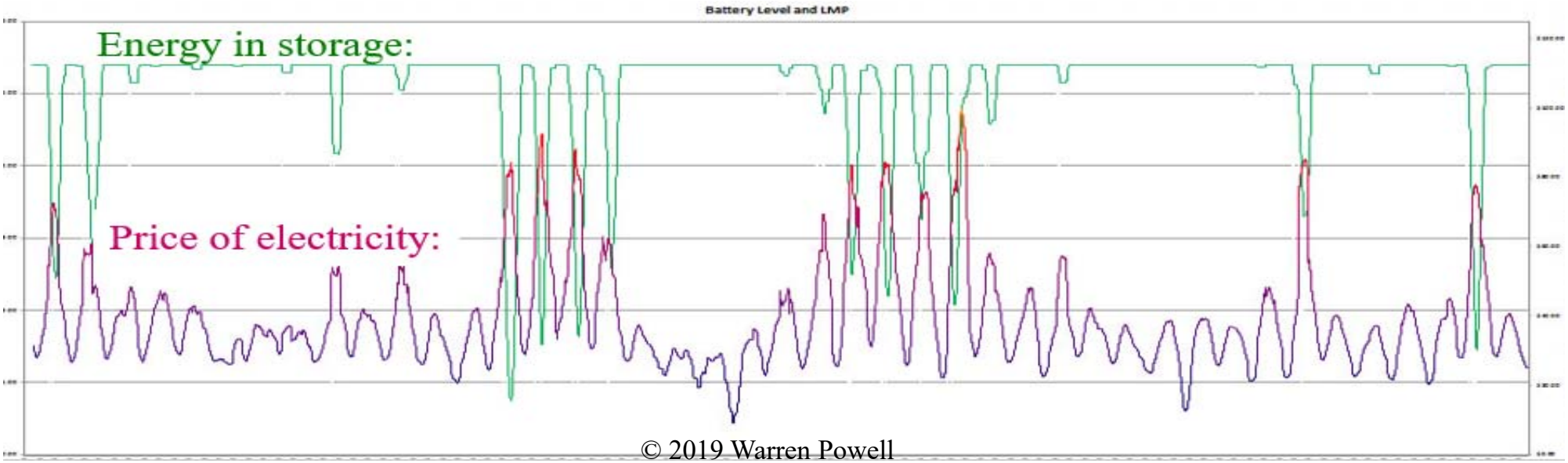
- Simultaneous perturbation stochastic approximation
  - » Let:
    - x^n be a p – dimensional vector.
    - δ^n be a scalar perturbation
    - Z^n be a p –dimensional vector, with each element drawn from a normal (0,1) distribution.
  - » We can obtain a sampled estimate of the gradient $\nabla_x F(x^n, W^{n+1})$ using two function evaluations: $F(x^n + \delta^n Z^n)$ and $F(x^n + \delta^n Z^n)$

$$\nabla_x F(x^n, W^{n+1}) = \begin{bmatrix} \dfrac{F(x^n + \delta^n Z^n) - F(x^n + \delta^n Z^n)}{2\delta^n Z_1^n} \\ \dfrac{F(x^n + \delta^n Z^n) - F(x^n + \delta^n Z^n)}{2\delta^n Z_2^n} \\ \vdots \\ \dfrac{F(x^n + \delta^n Z^n) - F(x^n + \delta^n Z^n)}{2\delta^n Z_p^n} \end{bmatrix}$$

Numerical derivatives

- Finite differences
  - » We use finite differences in MOLTE-DB:
    - We wish to optimize the decision of when to charge or discharge a battery

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

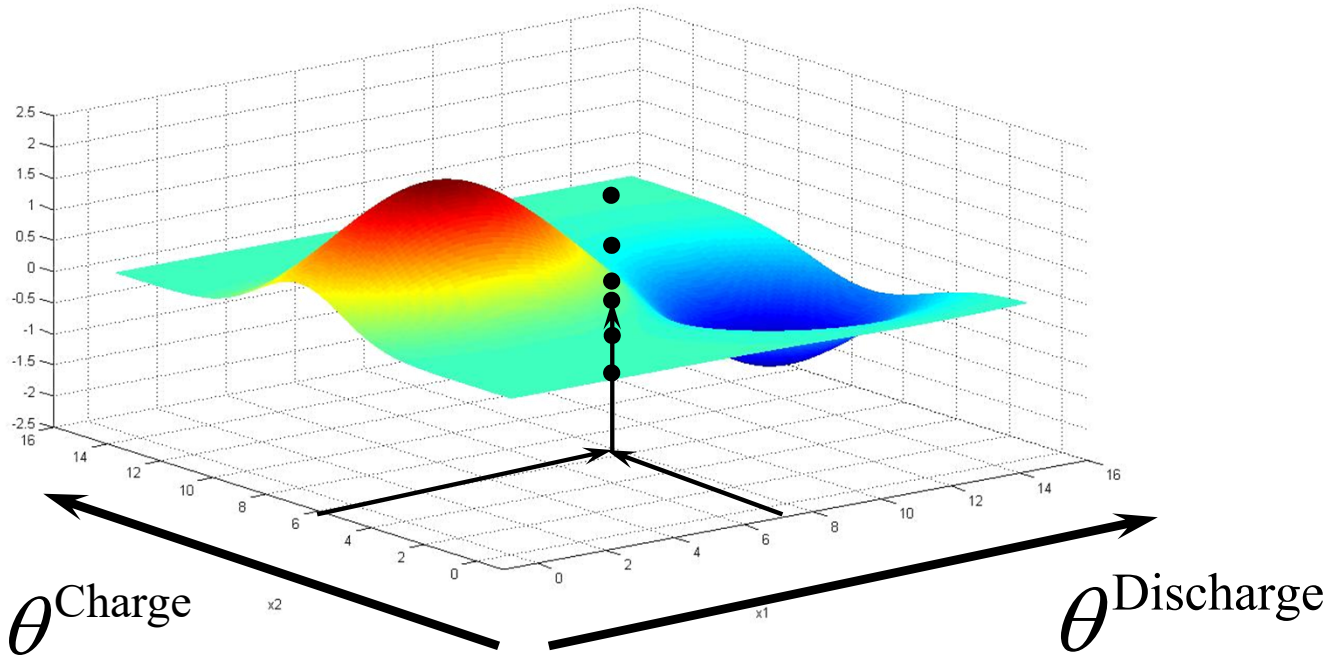


Numerical derivatives

- Finding the best policy
  - » We need to maximize

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

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

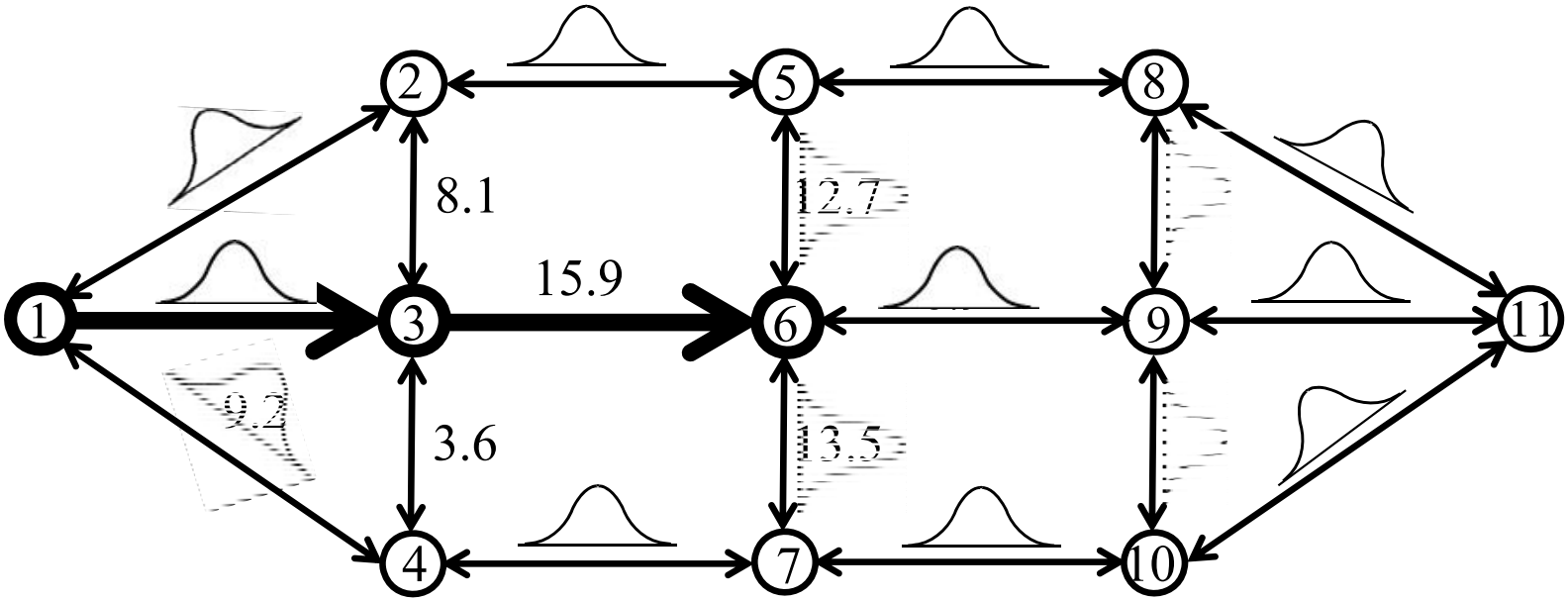


Cost function approximation

Stochastic shortest path

Stochastic shortest path problem

- A stochastic network, costs revealed as we arrive to a node:



Stochastic shortest path problem

- Modeling:
  - » State variable (iteration n, time t)
    - $R_t^n = current\ node = i_t^n$ during pass n
    - $I_t^n = information = (\bar{c}_{tij}^n) = estimates\ of\ costs\ at\ time\ t$ during pass n
    - $S_t^n = (R_t^n, I_t^n)$
  - » Decisions

    $x_{t,i_t^n,j}$=1 if we traverse , (i_t^n, j) at time t.
    - We want a policy $X_t^\pi(S_t) = \left(x_{t,i_t^n=i,j}^\pi\right)_{ij}$ for all links i, j.
  - » Costs
    - $\hat{c}_{tij}$ =Actual realization of costs at time t to traverse (i, j)

Stochastic shortest path problem

- Modeling:
  - » Objective function
    - Cost per period

$$\mathrm{C}(\mathrm{S_t}, \mathrm{X_t^\pi}(S_t)) = (X_t^\pi\ (S_t))^T \hat{c}_t$$

$$= \sum_{i,j} x^\pi_{t,i_t^n,j}\ \hat{c}_{tij}$$

=Costs incurred at time t.

- Total costs:

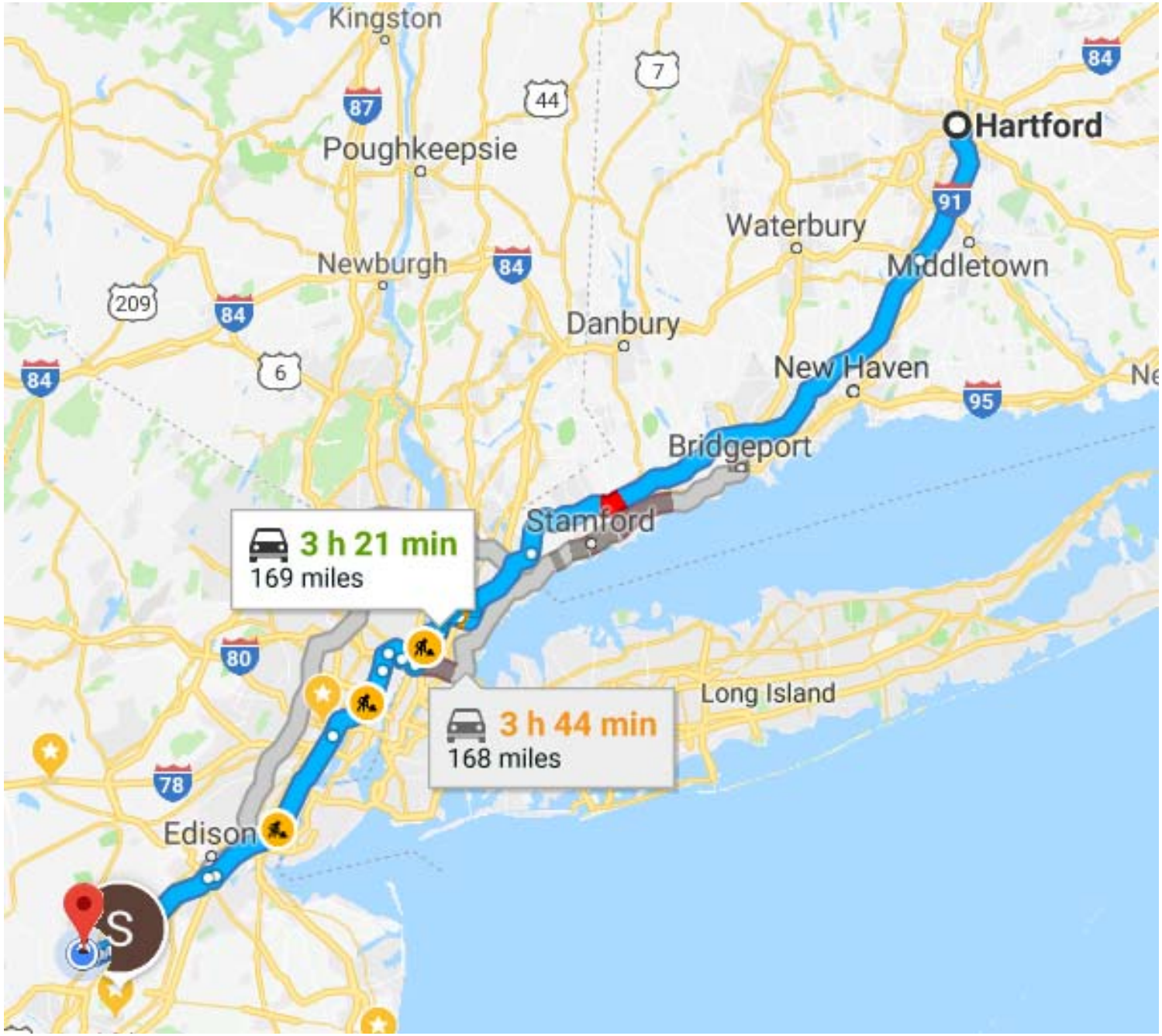
$$\min_\pi \mathbb{E} \sum_{t=0}^{T} \mathrm{C}(\mathrm{S_t}, \mathrm{X_t^\pi}(S_t))$$

  - » This is the base model.

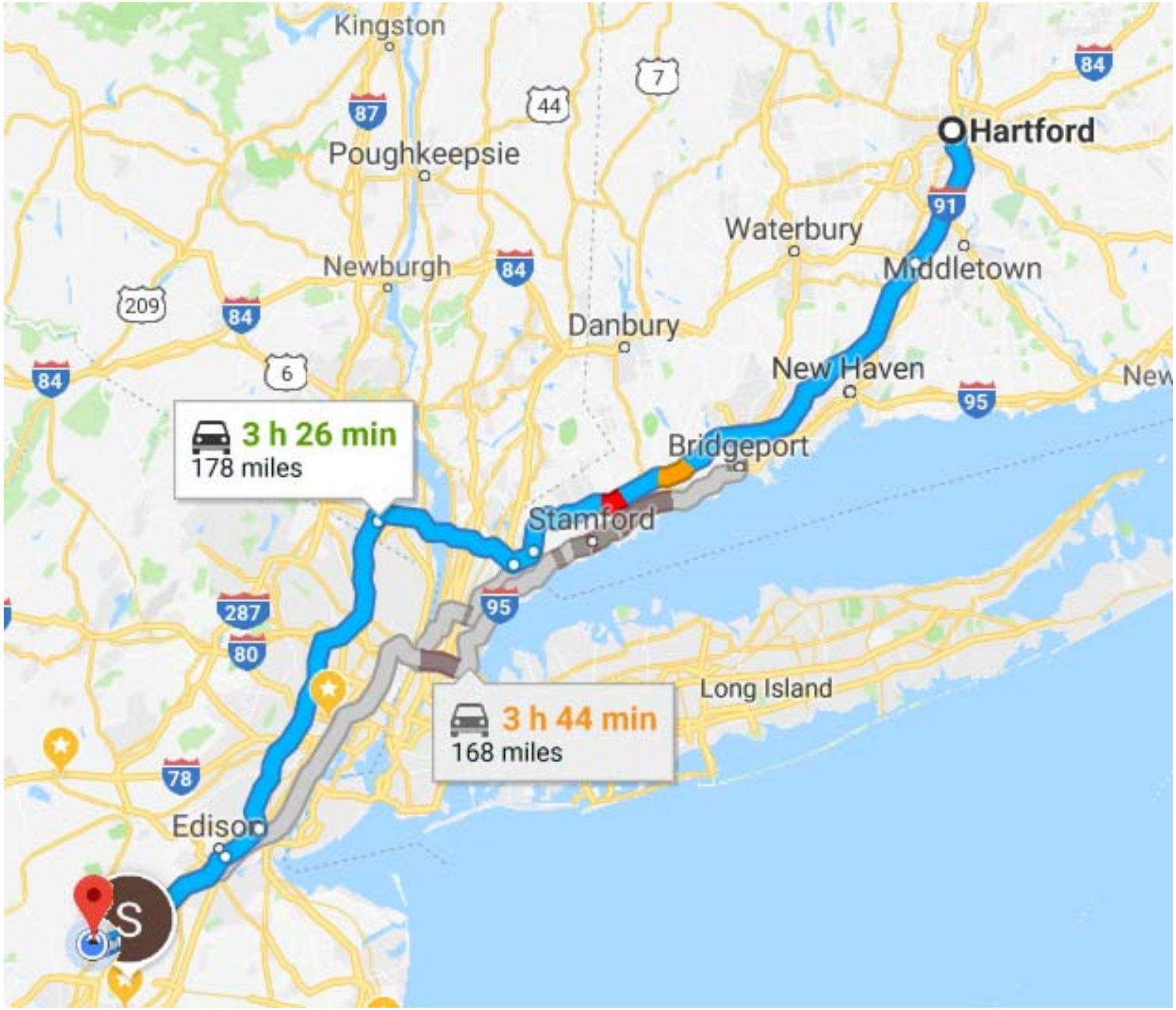
Stochastic shortest path problem

- A policy based on a lookahead model
  - » At each time t we are going to optimize over an estimate of the network that we are going to call the *lookahead model*.
  - » Notation: all variables in the lookahead model have tilde's, and two time indices.
    - First time index, t, is the time at which we are making a decision. This determines the information content of all parameters (e.g. costs) and decisions.
    - A second time index, t', is the time within the lookahead model.
  - » Decisions
    - $\tilde{x}_{tij} = 1$ if we plan on traversing link (i, j) in the lookahead model.
    - $\tilde{c}_{tij}$ = Estimated cost at time t of traversing link (i, j) in the lookahead model.

Stochastic shortest path problem

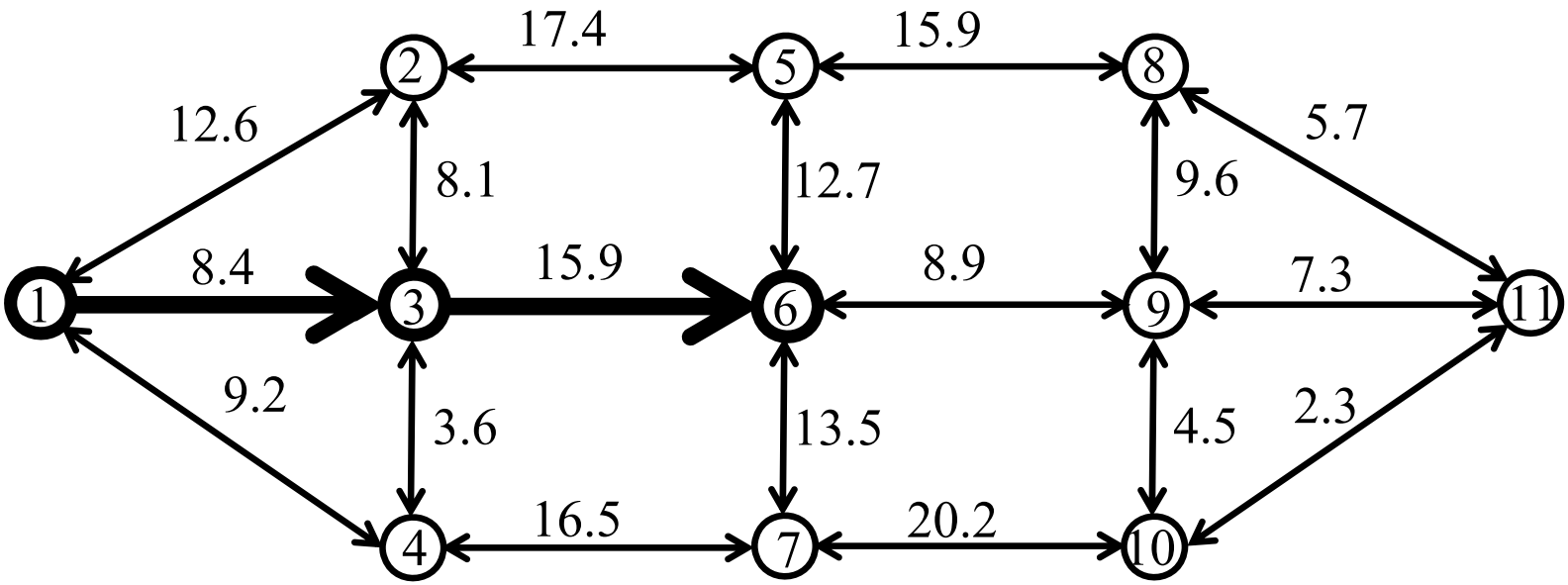


Stochastic shortest path problem



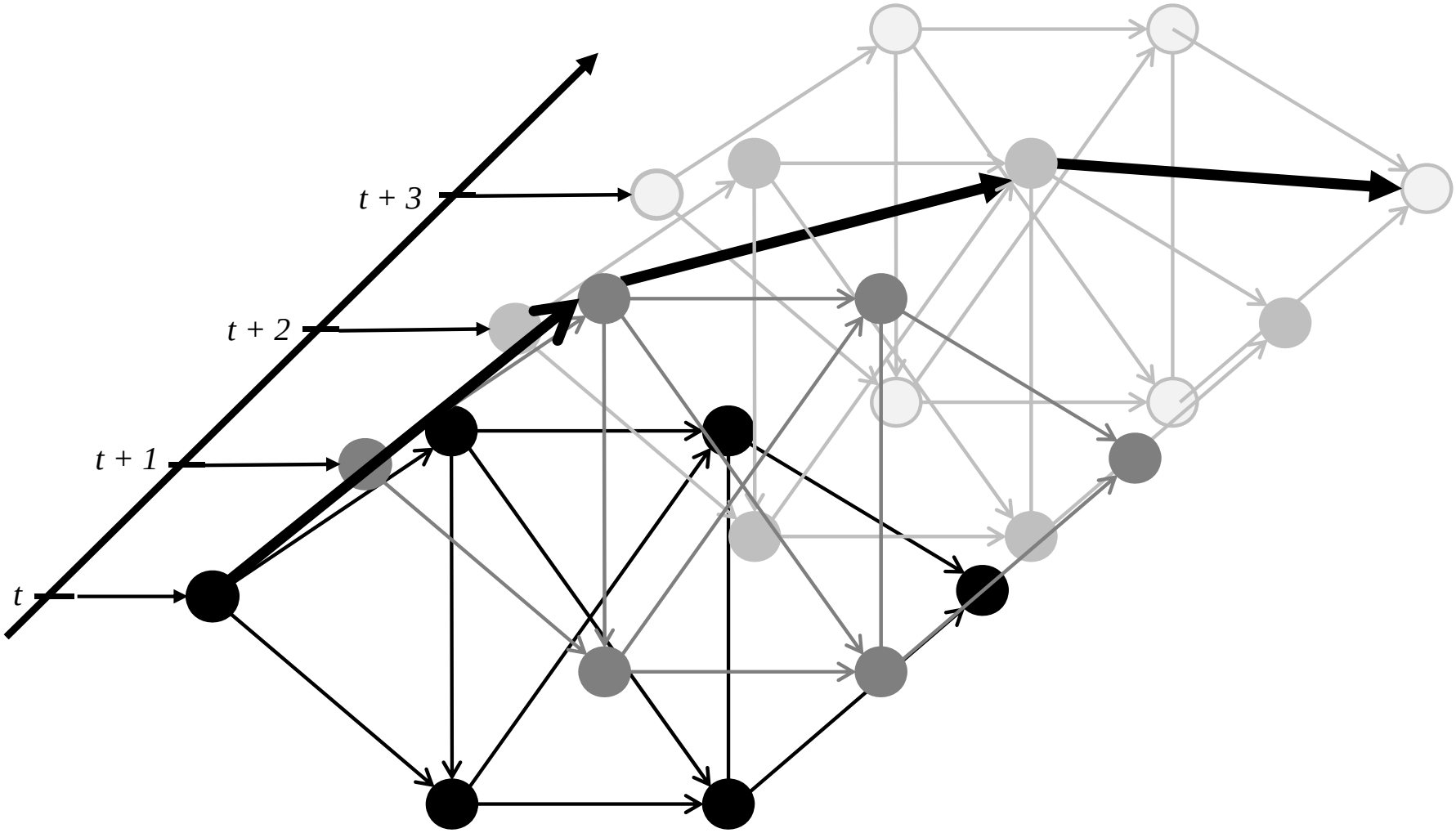
Stochastic shortest path problem

- A static, deterministic network



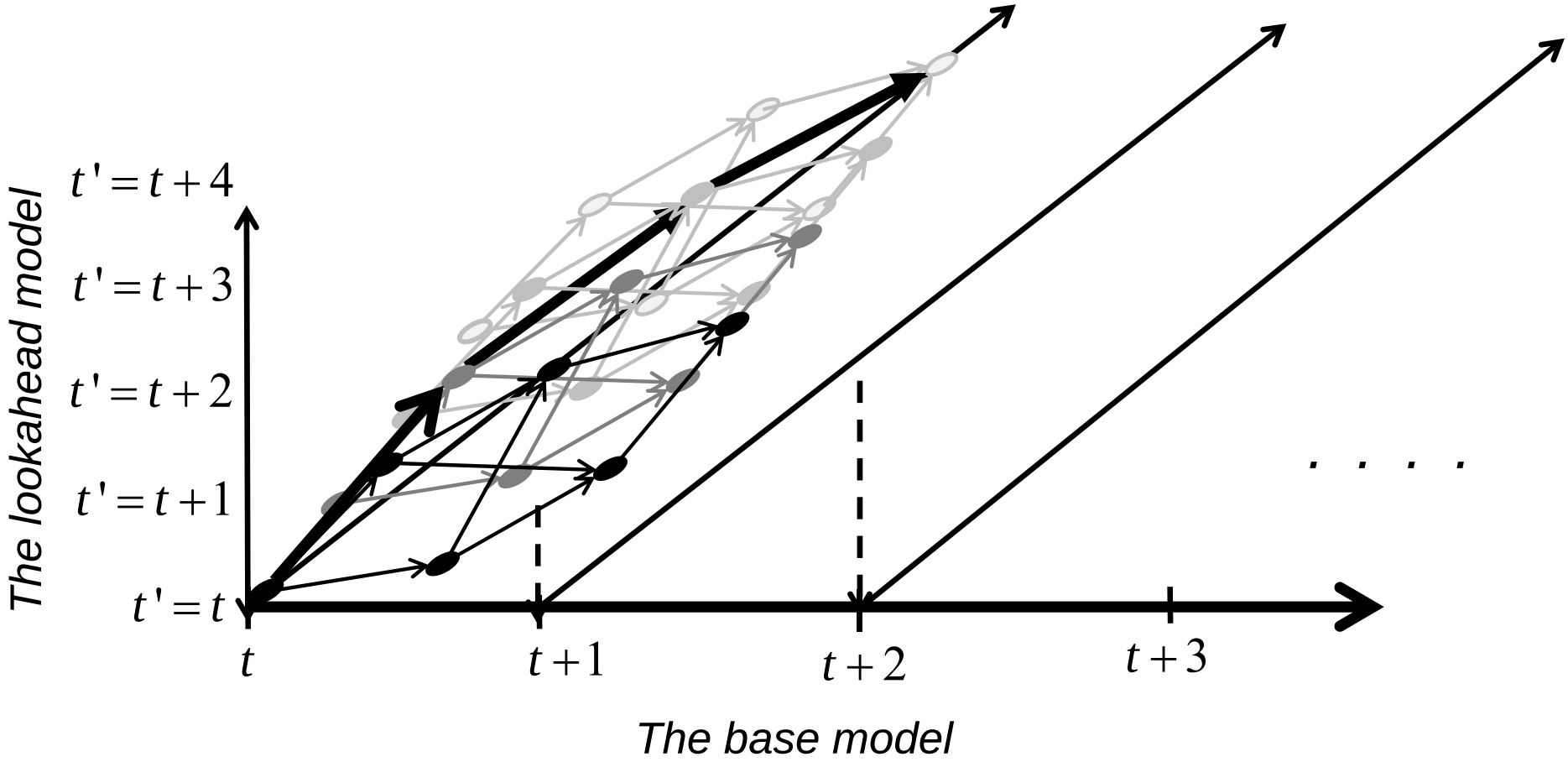
Stochastic shortest path problem

- A time-dependent, deterministic network



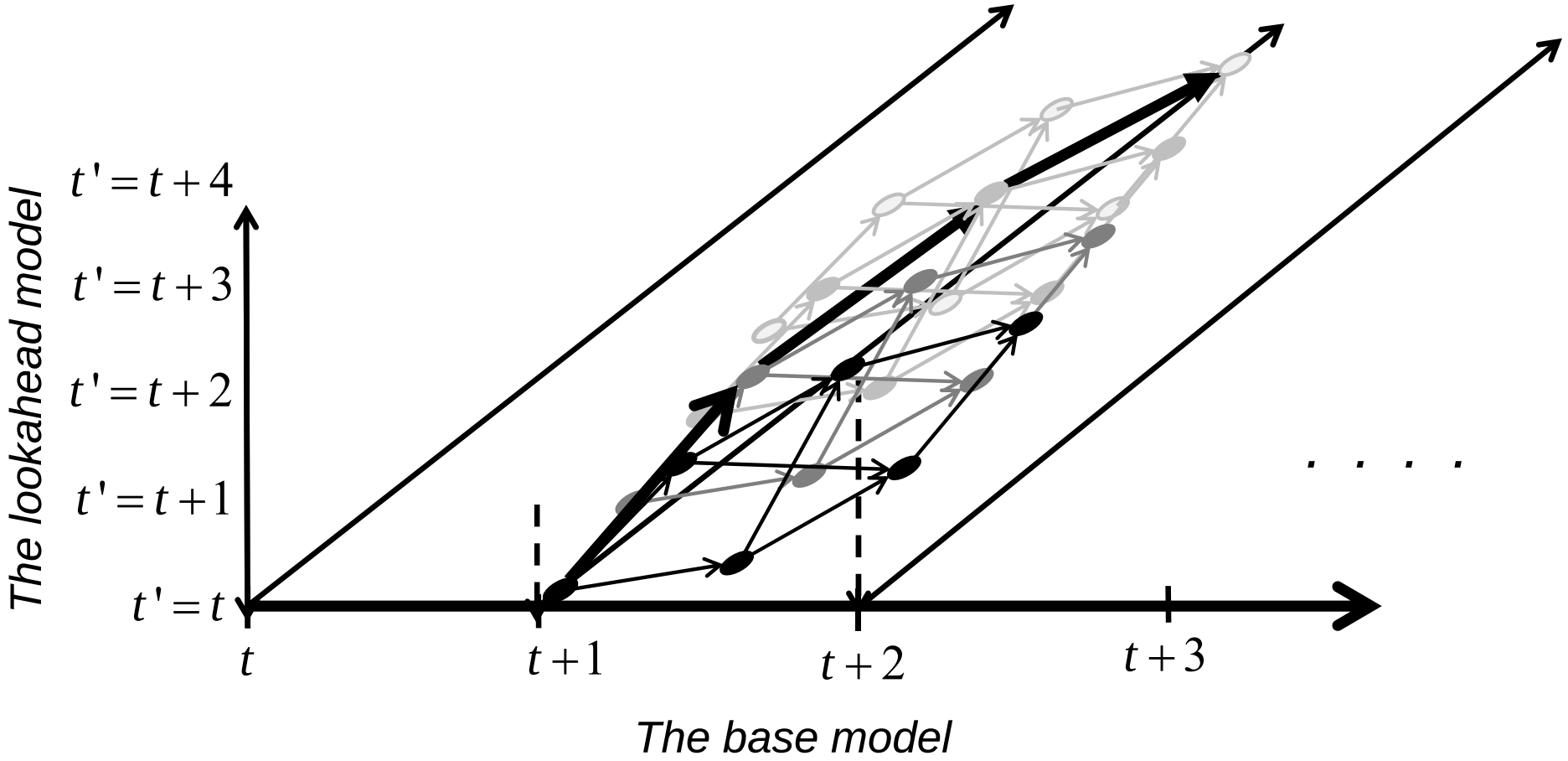
Stochastic shortest path problem

- A time-dependent, deterministic lookahead network



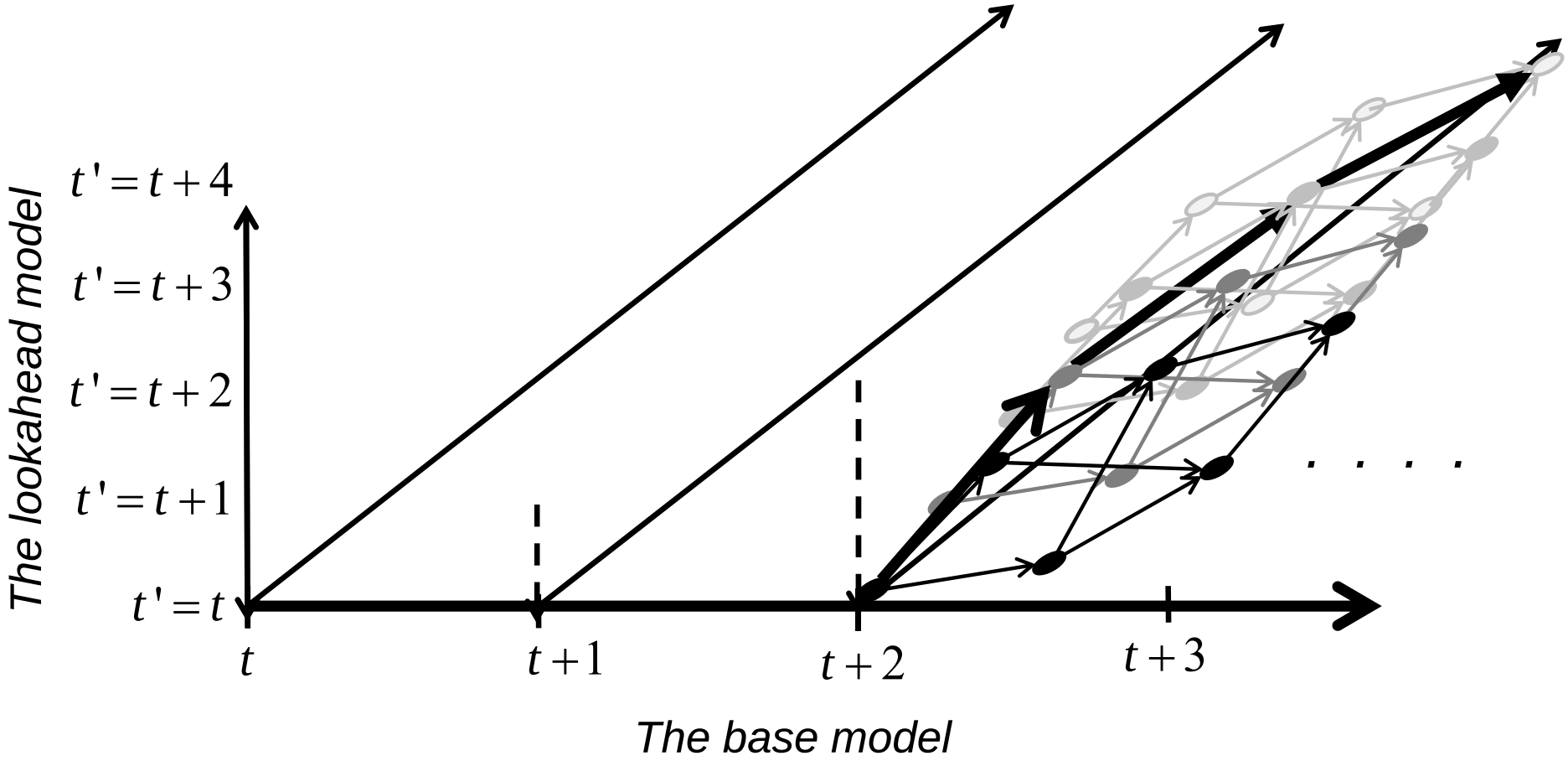
Stochastic shortest path problem

- A time-dependent, deterministic lookahead network



Stochastic shortest path problem

- A time-dependent, deterministic lookahead network



Stochastic shortest path problem

- Imagine that the lookahead is just a black box:
  - » Solve the optimization problem

$$X_t^{\pi}(S_t^n) = \arg\min \sum_{i \in N} \sum_{j \in N_i^+} \tilde{c}_{tij}^n \tilde{x}_{tij}$$

  - » subject to

$$\sum_j \tilde{x}_{i_t^n, j} = 1 \quad \text{Flow out of current node where we are located}$$

$$\sum_i \tilde{x}_{i,r} = 1 \quad \text{Flow into destination node } r$$

$$\sum_i \tilde{x}_{i,j} - \sum_k \tilde{x}_{j,k} = 0 \quad \text{for all other nodes.}$$

  - » This is a deterministic shortest path problem that we could solve using Bellman's equation, but for now we will just view it as a black box optimization problem.

Stochastic shortest path problem

- Simulating a lookahead policy

We would like to compute

$$F^{\pi} = \mathbb{E}\sum_{t=0}^{T}\sum_{i,j} X^{\pi}_{t,ij}(S_t)\hat{c}_{t,ij}$$

but this is intractable.

Let ω be a sample realization of costs

$$\hat{c}_{t,t',ij}(\omega), \hat{c}_{t+1,t',ij}(\omega), \hat{c}_{t+2,t',ij}(\omega), \ldots$$

Now simulate the policy

$$\hat{F}^{\pi}(\omega^n) = \sum_{t=0}^{T}\sum_{i,j} X^{\pi}_{t,ij}(S_t(\omega^n))\hat{c}_{t,ij}(\omega^n)$$

Talk through how this works.

Finally, get the average performance

$$\bar{F}^{\pi} = \frac{1}{N}\sum_{n=1}^{N}\hat{F}^{\pi}(\omega^n)$$

Stochastic shortest path problem

- Notes:
  - » The deterministic lookahead is still a policy for a stochastic problem.
  - » Can we make it better?
- Idea:
  - » Instead of using the expected cost, what about using a percentile.
  - » Use pdf of $\hat{c}_{ij}$ to find θ percentile (e.g. $\theta = .8$). Let

$$\tilde{c}_{ij}^{p}(\theta) = \text{The } \theta - \text{percentile of } \hat{c}_{ij}$$

  - » Which means $Prob\left[\hat{c}_{ij} \leq \tilde{c}_{ij}^{p}(\theta)\right] = \theta.$

Stochastic shortest path problem

- The θ −percentile policy.
  - » Solve the linear program (shortest path problem):

$$X_t^{\pi}(S_t^n \mid \theta) = \arg\min \sum_{i \in N} \sum_{j \in N_i^+} \tilde{c}_{tij}^{p}(\theta)\tilde{x}_{tij} \quad \text{(Vector with } x_{tij} = 1 \text{ if decision is to take } (i, j)\text{)}$$

  - » subject to

$$\sum_j \tilde{x}_{t,i_t^n,j} = 1 \quad \text{Flow out of current node where we are located}$$

$$\sum_i \tilde{x}_{tir} = 1 \quad \text{Flow into destination node } r$$

$$\sum_i \tilde{x}_{tij} - \sum_k \tilde{x}_{tjk} = 0 \quad \text{for all other nodes.}$$

  - » This is a deterministic shortest path problem that we could solve using Bellman's equation, but for now we will just view it as a black box optimization problem.

Stochastic shortest path problem

- Simulating a lookahead policy

  Let ω be a sample realization of costs

  $$\hat{c}_{t,t',ij}(\omega), \hat{c}_{t+1,t',ij}(\omega), \hat{c}_{t+2,t',ij}(\omega), \ldots$$

  Now simulate the policy

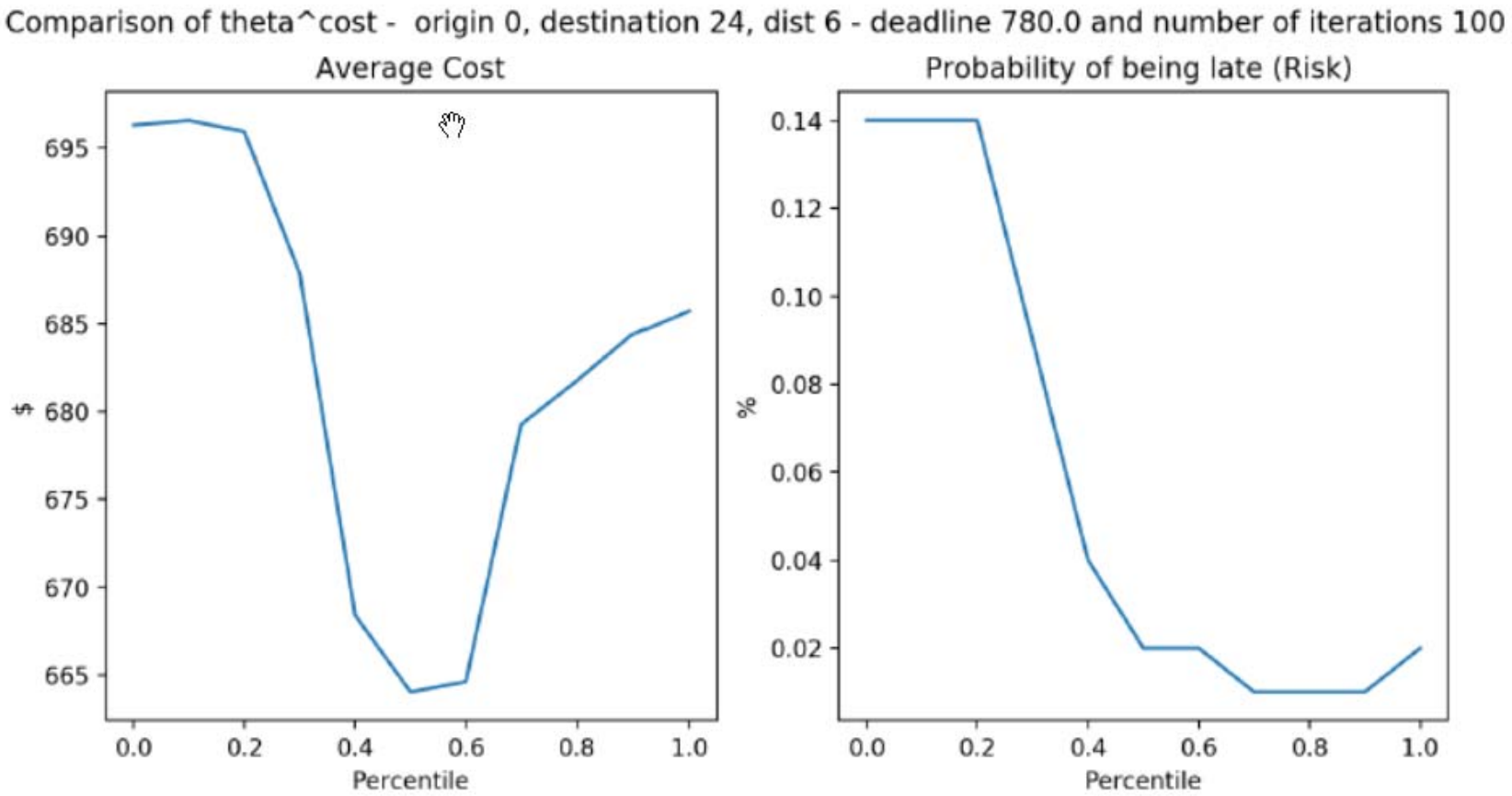
  $$\hat{F}^{\pi}(\omega^n) = \sum_{t=0}^{T} \sum_{i,j} \hat{c}_{t,t',ij}(\omega) X_t^{\pi}(S_t(\omega^n) \mid \theta)$$

  Finally, get the average performance

  $$\bar{F}^{\pi}(\theta) = \frac{1}{N} \sum_{n=1}^{N} \hat{F}^{\pi}(\omega^n)$$

Stochastic shortest path problem

- Policy tuning
  - » Cost vs. lateness (risk)



Notes on cost function approximations

Cost function approximations

- Notes:
  - » CFAs are the dirty secret of real-world stochastic optimization. They are easy to understand, easier to solve, and make it possible to incorporate problem structure.
  - » The first challenge is to identify an effective parameterization. This requires having insights into the problem.
  - » The second challenge is to optimize over the parameters.
  - » There are close parallels between policy search and machine learning.

Cost function approximations

- Tuning the parameters
  - » Let

$$X^{\pi}(S^n|\theta) = \text{Tunable policy}$$

  - » The simulated performance of the policy is given by
    - Final reward:
      - Let $x^{\pi,N}(\theta)$ be the solution produced by following policy π parameterized by θ. The performance of the policy is given by

$$F^{\pi}(\theta, W) = F(x^{\pi,N}(\theta), \widehat{W})$$

    - Cumulative reward

$$F^{\pi}(\theta, W) = \sum_{t=0}^{T} C(S_t, X^{\pi}(S^n|\theta))$$

Cost function approximations

- Tuning the parameters
  - » We can write any parameter tuning problem as

$$\max_{\theta} \mathbb{E}_W F(\theta, W)$$

  - » This is a basic stochastic learning problem. We can approach it using:
    - Derivative-based stochastic search – Will probably need to use numerical derivatives.
    - Derivative-free stochastic search – We might represent the set of possible values of $\theta \in \{\theta_1, \ldots, \theta_K\}$ and then search over this finite set.

Learning and tuning policies

Learning policies

- Simulating a policy
  - » Regardless of our belief model (frequentist or Bayesian), a learning process looks like

$$(S^0, x^0, W^1_{x^0}, S^1, x^1, W^2_{x^1}, \ldots, S^{N-1}, x^{N-1}, W^N_{x^{N-1}}, S^N)$$

  - » Our observations might come from

$$W^{n+1}_{x^n} = \mu_{x^n} + \varepsilon^{n+1}$$

    or perhaps

$$\hat{F}^n = F(x^n, W^{n+1})$$

    There are different ways to interpret our "W" variable.
  - » Imagine that we have three policies π^A, π^B, and π^C. Let $\Pi = (\pi^A, \pi^B, \pi^C)$.
  - » In offline learning, we only care about our final answer which we call:

$$x^{\pi,N} = \max_x \bar{\mu}^N_x$$

Learning policies

- Learning the best policy
  - » Imagine that we have three policies π^A, π^B, and π^C. Let $\Pi = (\pi^A, \pi^B, \pi^C)$.
  - » In offline (final reward) learning, we will optimize the final reward

$$\max_{\pi} \mathbb{E}_{\mu} \mathbb{E}_{W^1,\ldots,W^N|\mu} \bar{\mu}_x^N$$

or

$$\max_{\pi} \mathbb{E}_{\mu} \mathbb{E}_{W^1,\ldots,W^N|\mu} \mathbb{E}_{\widehat{W}|\mu} F(x^{\pi,N}, \widehat{W})$$

  - » In online (cumulative reward) learning, we optimize the cumulative rewards:

$$\max_{\pi} \mathbb{E}_{\mu} \mathbb{E}_{W^1,\ldots,W^N|\mu} \mathbb{E}_{\widehat{W}|\mu} \sum_{n=0}^{N-1} F(X^{\pi}(S^n|\theta), W^{n+1})$$

  - » x

Learning policies

- Finding a policy
  - » Imagine that we have three classes of policies π^A, π^B, and π^C. Let $\Pi = (\pi^A, \pi^B, \pi^C)$.
  - » For each class of policy, we might have a set of parameters $\theta \in \Theta^A$ (for example) that have to be tuned.
  - » In offline learning, we only care about our final answer which we call:

$$x^{\pi,N} = \max_x \bar{\mu}_x^N$$

Note that the policy π is implicit in the estimate $\bar{\mu}_x^N$. We make it explicit when we write the final design $x^{\pi,N}$.

  - » Now we wish to solve the optimization problem that finds the best policy (best class, and best within the class).

Learning policies

- Notes
  - » Buried in the estimate of $\bar{\mu}_x^N$ is:
    - The truth μ_x
    - The set of observations:

$$W_{x^0}^1, W_{x^1}^2, \ldots, W_{x^{N-1}}^N$$

  - » …where the choices $x^0, x^1, \ldots, x^{N-1}$ are determined by our learning policy $x^n = X^\pi(S^n)$. This is the reason we label

$$x^{\pi,N} = \max_x \bar{\mu}_x^N$$

with the policy π.

Learning policies

- Evaluating a policy
  - » The final design $x^{\pi,N}$ is a random variable since it depends on:
    - The true μ_x (in our Bayesian model, we treat this as a random variable)
    - The observations $W^1, W^2, \ldots, W^N$
  - » We can evaluate a policy $X^\pi(S)$ using

$$F^\pi = \mathbb{E}\mu_{x^{\pi,N}} = \mathbb{E}\bar{\mu}^N_{x^{\pi,N}}$$ (these are equivalent - more on this later).

  - » Finally, we write our optimization problem as

$$\max_\pi \mathbb{E}\bar{\mu}^N_{x^{\pi,N}}$$

  - » It is useful to express what the expectations are over, so we write

$$\max_\pi \mathbb{E}_\mu \mathbb{E}_{W^1,W^2,\ldots,W^N|\mu} \bar{\mu}^N_{x^{\pi,N}}$$

Learning policies

- Simulating a policy
  - » We cannot compute expectations, so we have to learn how to simulate:

$$\max_{\pi} \mathbb{E}_{\mu} \mathbb{E}_{W^1, W^2, \ldots, W^N | \mu} \bar{\mu}^N_{x^{\pi}, N}$$

  - » Let $\mu(\psi)$ be a sampled truth, and let $W^1(\omega), \ldots, W^N(\omega)$ be a sample path of function observations.
  - » Assume we have truths $\psi^{\ell}, \ell = 1, \ldots, L$, and sample paths $\omega^1, \ldots, \omega^K$.
  - » Let $\bar{\mu}_x^{\pi,N}(\psi^{\ell}, \omega^k)$ be the estimate of μ_x when the truth is ψ^{ℓ} and the sample path of realizations is ω^k, while following experimental policy π.
  - » Now evaluate the policy using

$$\bar{F}^{\pi} = \max_{\pi} \frac{1}{L} \sum_{\ell=1}^{L} \frac{1}{K} \sum_{k=1}^{K} \bar{\mu}_x^{\pi,N}(\psi^{\ell}, \omega^k)$$

“Contextual” learning

Contextual learning

- What if we are given information before making a decision?
  - » We see the attributes of a patient before prescribing treatment.
  - » We are given a weather report before having to decide how many newspapers to put in our kiosk.
  - » Let's call this an "environmental state" variable (this is not standard, but there is not a standard name for this).
- Example:
  - » Newsvendor problem with a dynamic state variable (the price):

$$\max_{x \geq 0} \mathbb{E}\, C(S, x, W) = \mathbb{E}\left\{ p_t \min(x, W) - cx \right\}$$

  - » Further assume that p_t is independent of any previous prices, and that it is revealed before we make our decision x.
- The price p_t is known as a "context." This would be called a "contextual learning problem" or a "contextual bandit problem."

Contextual learning

- Why is this special?
  - » Without the dynamic information, we would be searching for the best x.
  - » With the contextual information, we are now looking for $x(p)$ where p is the revealed price. This means that instead of looking for a deterministic variable (or vector), we are now looking for a function.
  - » When we are looking for a decision as a function of a state variable (even part of a state variable), then this is what we call a *policy*.
  - » The learning literature makes a big deal about "contextual bandit problems." For us, there is nothing different about this than any state-dependent problem.