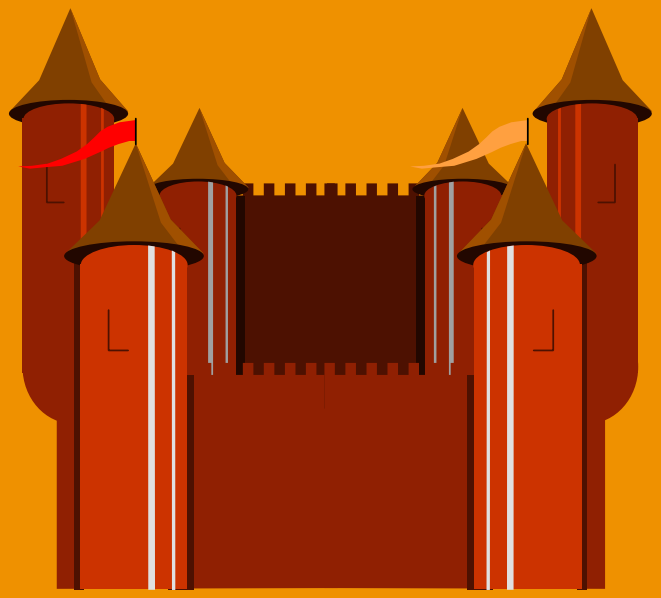
ORF 544

Stochastic Optimization and Learning

Spring, 2019



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

Week 2

Chapter 3: Recursive learning

Overview

Types of learning problems

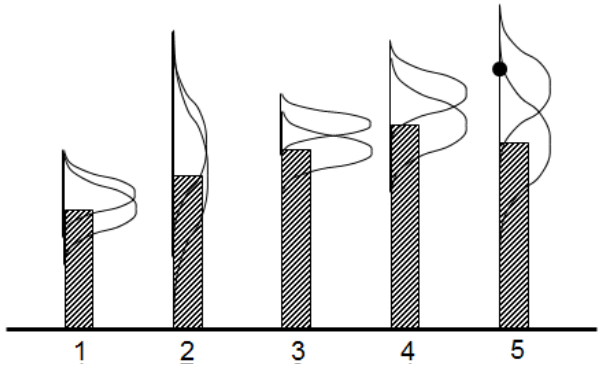
Types of approximation strategies

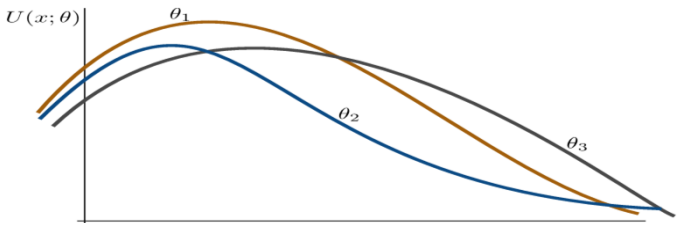
Learning problems

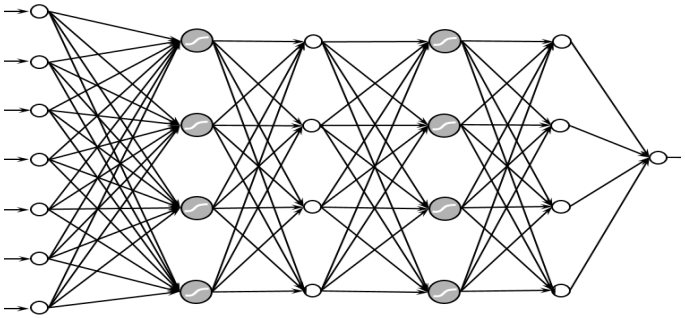
- Classes of learning problems in stochastic optimization
  1) Approximating the objective
  $$\bar{F}(x|\theta) \approx \mathbb{E}F(x,W).$$
  2) Designing a policy $X^{\pi}(S|\theta)$.
  3) A value function approximation
  $$\bar{V}_t(S_t|\theta) \approx V_t(S_t).$$
  4) Designing a cost function approximation:
     - The objective function $\bar{C}^{\pi}(S_t,x_t|\theta)$.
     - The constraints $X^{\pi}(S_t|\theta)$
  5) Approximating the transition function
  $$\bar{S}^M(S_t,x_t,W_{t+1}|\theta) \approx S^M(S_t,x_t,W_{t+1})$$

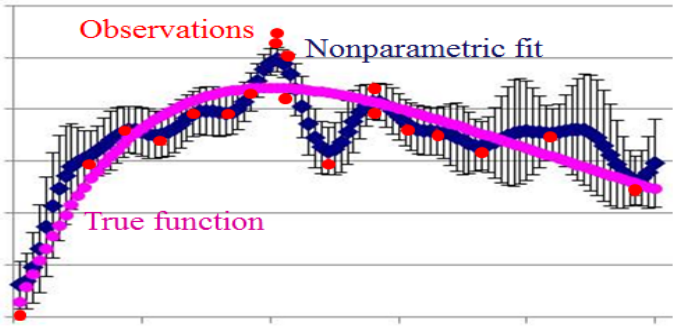
Approximation strategies

- 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





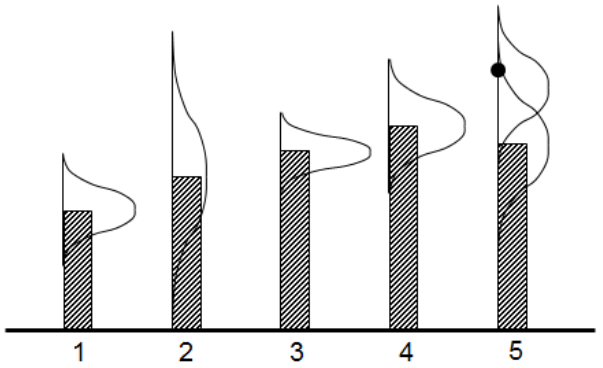




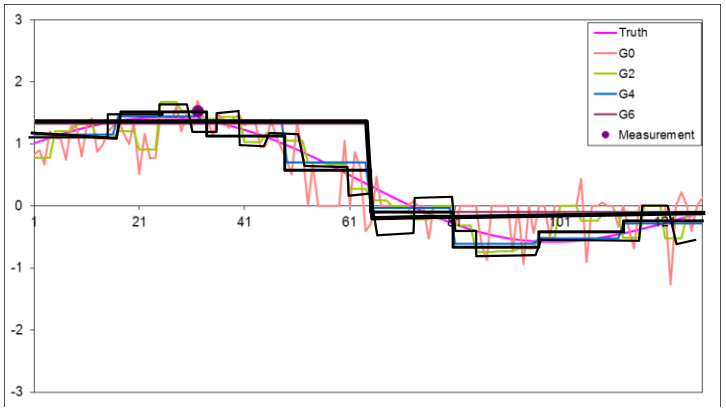
Approximating strategies

- 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

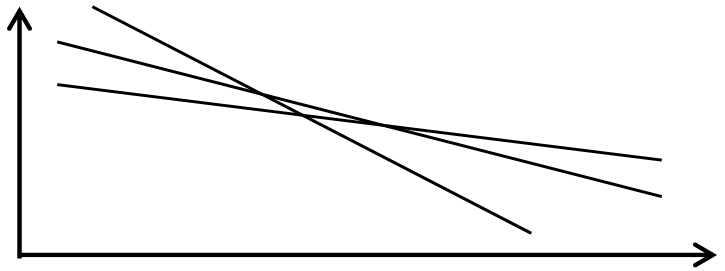
$$F(x|\theta) = \sum_{f \in F} \theta_f \phi_f(x)$$

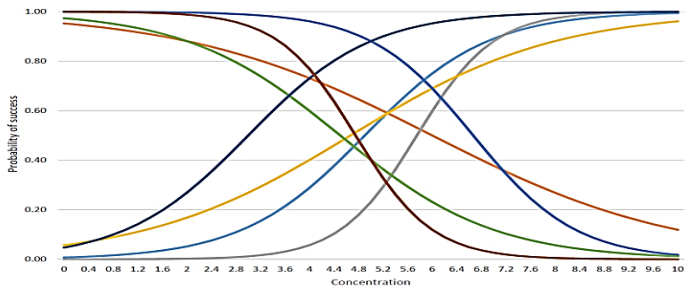
    - Might include sparse-additive, where many parameters are zero.
  - » Nonlinear models

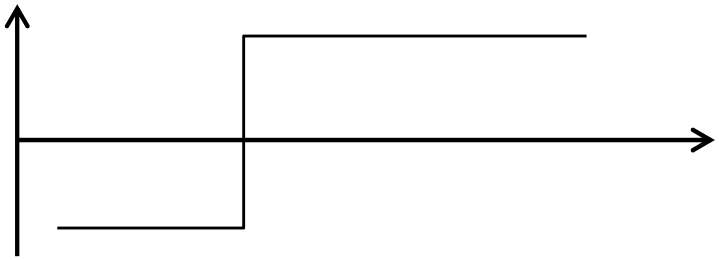
$$F(x|\theta) = \frac{e^{\theta_0 + \theta_1 \phi_1(x) + \ldots}}{1 + e^{\theta_0 + \theta_1 \phi_1(x) + \ldots}}$$

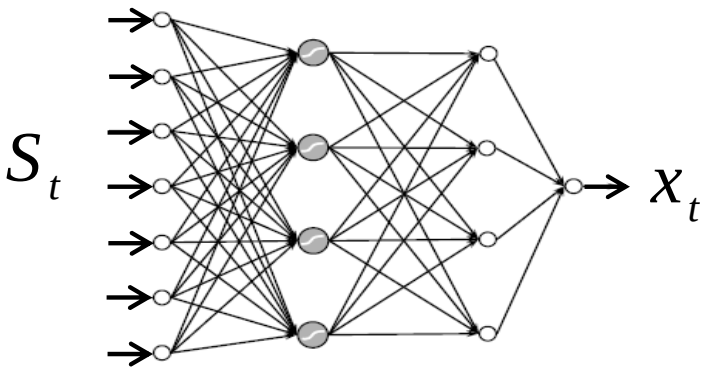
$$X^{\pi}(S_t|\theta) = \begin{cases} +1 & \text{if } p_t < \theta^{\text{charge}} \\ 0 & \text{if } \theta^{\text{charge}} < p_t < \theta^{\text{discharge}} \\ -1 & \text{if } p_t > \theta^{\text{charge}} \end{cases}$$

    - (Shallow) Neural networks



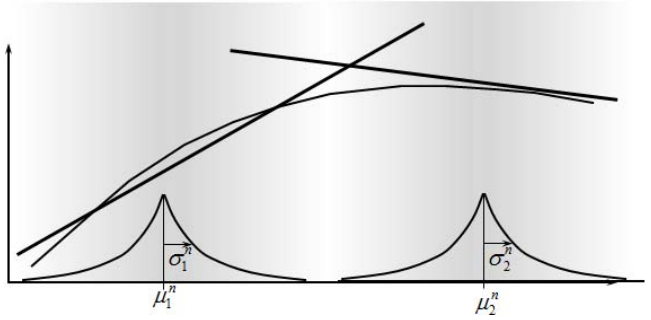


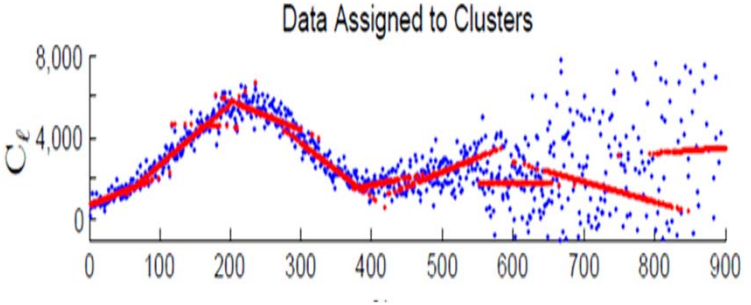


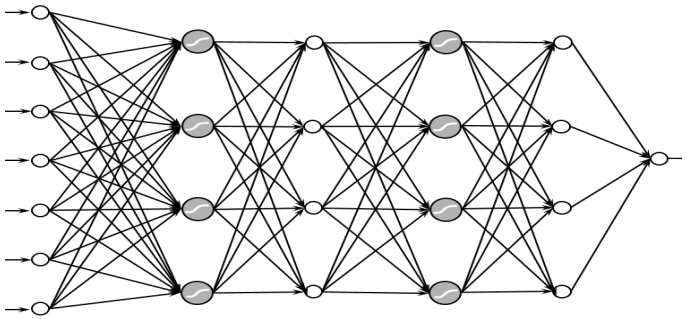


Approximating strategies

- 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







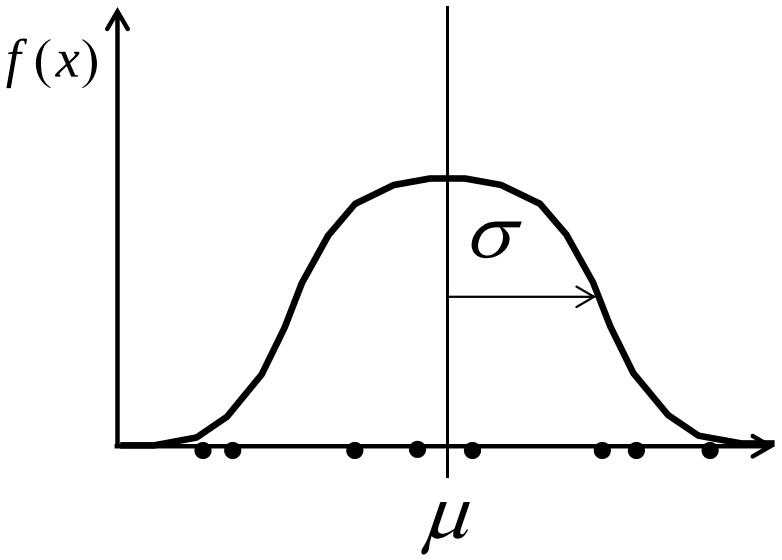
Approximating strategies

- Parametric vs. nonparametric
  - » Working definition of nonparametric:
    - Given an infinitely large dataset, a nonparametric model can provide a perfect fit.
  - » "Parametric" models are inherently low dimensional.
    - Might have hundreds of parameters, perhaps more.
    - Includes "small" neural networks.
  - » "Nonparametric" models are inherently high dimensional.
    - Deep neural networks may have tens of thousands of parameters. In principle can fit anything given a large enough dataset.

Approximating strategies

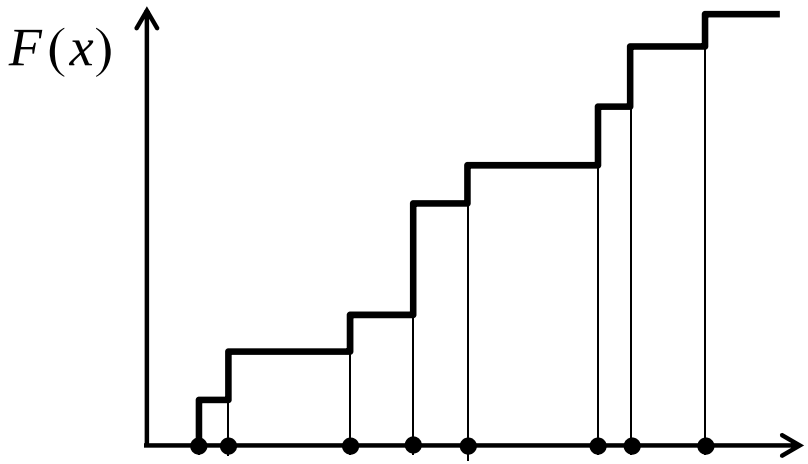
- Parametric vs. sampled distributions
  - » A sampled representation of a distribution is a form of nonparametric model

Parametric distribution (pdf)



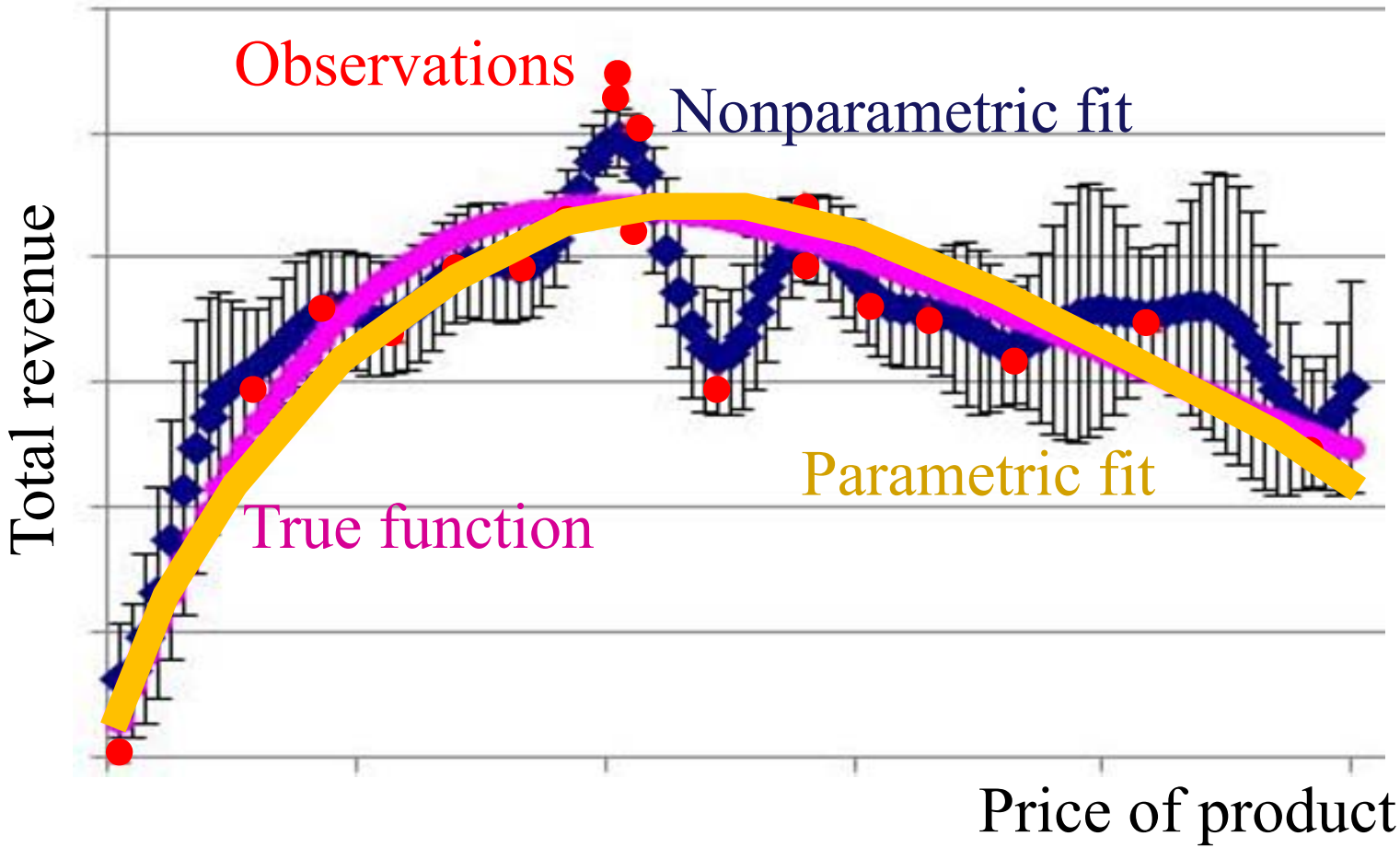
$$f(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\left(\frac{(x-\mu)^2}{\sigma^2}\right)}$$

Nonparametric distribution (cdf)



Approximating strategies

- Parametric vs. nonparametric



» Robust CFAs are *parametric*

» Scenario trees are *nonparametric*

Approximating strategies

- Notes:
  - » World of "big data" encourages use of high-dimensional architectures:
    - Kernel regression
    - Locally linear models
    - Support vector machines
    - Deep neural networks
  - » Modern uses of neural networks (e.g. image recognition) depend on massive datasets (e.g. millions of records).
  - » In stochastic optimization, we are often estimating functions iteratively, starting with little or no data and growing.
  - » There are many applications where we are limited to dozens of iterations, although we will have settings where we may run an algorithm hundreds or even thousands of iterations.

Approximating strategies

- What we are going to focus on:
  - » Lookup tables
    - Frequentist vs. Bayesian
    - Independent beliefs
    - Correlated beliefs
    - Hierarchical representations (time permitting)
  - » Parametric models
    - Linear in the parameters
    - Nonlinear in the parameters
  - » Sampled models
    - This can be thought of as a nonparametric distribution.
  - » Recursive estimation
    - Classical machine learning is batch
    - Our algorithms will always require adaptive learning (known as “online machine learning” in the computer science community).

Learning with lookup tables

Bayesian vs. frequentist

Independent beliefs

Correlated beliefs

Learning with lookup tables

- Notation:
  - » $x \in X = \{x_1, x_2, \ldots, x_M\}$ (assume M is "not too large")
  - » μ_x – truth (random variable)
  - » $\bar{\mu}_x^n$ – estimate of μ_x after n experiments have been run
  - » W_x^{n+1} – results of n+1st experiment at $x = x^n$
- Examples:
  - » x is a drug for diabetes, W_x^{n+1} is the reduction in blood sugar
  - » x is a lineup for a basketball team, W_x^{n+1} is the points scored minus points against during the game.
  - » x is a price, W_x^{n+1} is the revenue
  - » x is the bid for an add on google, W_x^{n+1} is the number of clicks
  - » x is buy and sell prices for charging/discharging a battery using electricity from the grid, W_x^{n+1} is net profit for a day.
  - » x is a movie to display on Netflix, W_x^{n+1} is whether or not a user chooses the movie.

Learning with lookup tables

- The information and decision process

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

- Where:
  - » S^n =Belief about the value of μ_x, $x \in X = \{x_1, \ldots, x_M\}$.
  - » 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, $n = 1,2, \ldots, N$

Learning with lookup tables

- The information process:

The remainder of this chapter will focus on methods for updating lookup table belief models. For this setting, we are going to use W_x^n as the observation of our function $f(x)$, which means that if we choose to evaluate the function at $x = x^n$, then we are going to observe

$$\begin{aligned} W_{x^n}^{n+1} &= f(x^n) + \varepsilon^{n+1} \\ &= \mu_{x^n} + \varepsilon^{n+1}. \end{aligned}$$

Below, we describe how to use frequentist and Bayesian statistics to produce estimates of μ_x for each x.

  - » We make the decision x^n using the results of experiment *n*, and then run experiment *n+1*, from which we obtain a noisy observation $W_{x^n}^{n+1}$.
  - » We use the convention that any variable index by *n* contains the information from $W^1,\ldots,W^n$

Learning with lookup tables

- Frequentist view
  - » Estimates are based purely on experimental observations.
  - » Does not require any prior knowledge; similarly, it is not able to use any prior knowledge that might be available.

Learning with lookup tables

2.2 THE FREQUENTIST VIEW

The frequentist view is arguably the approach that is most familiar to people with an introductory course in statistics. Assume we are trying to estimate the mean μ of a random variable W which might be the performance of a device or policy. Let W^n be the nth sample observation. Also let $\bar{\mu}^n$ be our estimate of μ, and $\hat{\sigma}^{2,n}$ be our estimate of the variance of W. We know from elementary statistics that we can write $\bar{\mu}^n$ and $\hat{\sigma}^{2,n}$ using

$$\bar{\mu}^n = \frac{1}{n}\sum_{m=1}^{n} W^m \tag{2.2}$$

$$\hat{\sigma}^{2,n} = \frac{1}{n-1}\sum_{m=1}^{n}(\hat{f}^m - \bar{\mu}^n)^2. \tag{2.3}$$

The estimate $\bar{\mu}^n$ is a random variable (in the frequentist view) because it is computed from other random variables, namely $W^1, W^2, \ldots, W^n$. Imagine if we had 100 people each choose a sample of n observations of W. We would obtain 100 different estimates of $\bar{\mu}^n$, reflecting the variation in our observations of W. The best estimate of the variance of the estimator $\bar{\mu}^n$ is easily found to be

$$\bar{\sigma}^{2,n} = \frac{1}{n}\hat{\sigma}^{2,n}.$$

Note that as $n \to \infty$, $\bar{\sigma}^{2,n} \to 0$, but $\hat{\sigma}^{2,n} \to \sigma^2$ where σ^2 is the true variance of W. If σ^2 is known, there would be no need to compute $\hat{\sigma}^{2,n}$ and $\bar{\sigma}^{2,n}$ would be given as above with $\hat{\sigma}^{2,n} = \sigma^2$.

Learning with lookup tables

- Recursive formulas

$$\bar{\mu}^n = \left(1 - \frac{1}{n}\right)\bar{\mu}^{n-1} + \frac{1}{n}W^n, \tag{2.4}$$

$$\hat{\sigma}^{2,n} = \begin{cases} \frac{1}{n}(W^n - \bar{\mu}^{n-1})^2 & n = 2, \\ \frac{n-2}{n-1}\hat{\sigma}^{2,n-1} + \frac{1}{n}(W^n - \bar{\mu}^{n-1})^2 & n > 2. \end{cases} \tag{2.5}$$

$$K^n_{freq} = (\bar{\mu}^n, \hat{\sigma}^{2,n}, n).$$

		batch			recursive	
n	W^n	mean	variance		mean	variance
1	74	74.00			74.00	
2	66	70.00	32.00		70.00	32.00
3	70	70.00	16.00		70.00	16.00
4	49	64.75	120.92		64.75	120.92
5	76	67.00	116.00		67.00	116.00
6	79	69.00	116.80		69.00	116.80
7	47	65.86	166.48		65.86	166.48
8	52	64.13	166.70		64.13	166.70
9	53	62.89	159.61		62.89	159.61
10	64	63.00	142.00		63.00	142.00

Learning with lookup tables

- Bayesian view
  - » The Bayesian view treats the true value of $\mu_x = \mathbb{E}_W F(x, W)$ as a random variable with some assumed distribution.
  - » We start with a prior distribution of belief about unknown parameters, even before we have collected any data.
  - » Useful when we have prior knowledge about a problem:
    - We may know something about how a patient might respond to a drug because of experience with other patients.
    - We have an idea how the market will respond to the price of a book because we see how it responded to other similar books.

Learning with lookup tables

- Belief state
  - $B^{Bayes,n} = (\bar{\mu}^n, \beta^n)$

  » Sometimes use S^n – The notation S^n will eventually be used for more general states, but B^n is always the belief state. Many authors use the term "belief state." We use belief state and knowledge state interchangeably.

- Conjugacy
  » Conjugacy means the *posterior distribution* is in the same family as the *prior distribution.*
  » *Normal* prior plus *normal* observation produces *normal* posterior. So, "normal-normal" is called a *conjugate family.*

Learning with lookup tables

- Conditional expectations
  - » We often have to take expectations given what we know at time n. There are several ways to write this. Consider the estimate $\bar{\mu}_x^n$ of the value of a choice x after we have made n observations.
  - » We use the convention that by indexing the estimate by n that it has seen the observations $W^1, W^2, \ldots, W^n$.
  - » The following are equivalent:

$$\mathbb{E}^n \bar{\mu}_x^n = \mathbb{E}[\bar{\mu}_x^n | W^1, \ldots, W^n] = \mathbb{E}[\bar{\mu}_x^n | S^n]$$

  - » This applies to variances, which are a form of expectation. Recall that we can write $Var(X) = \mathbb{E}\big(X - \mathbb{E}(X)\big)^2$. Similarly we can write

$$Var^n \bar{\mu}_x^{n+1} = Var[\bar{\mu}_x^{n+1} | W^1, \ldots, W^n] = Var[\bar{\mu}_x^{n+1} | S^n]$$

  - » So what is $Var^n \bar{\mu}_x^n$?

Learning with lookup tables

- Bayesian updating – independent beliefs
  - » Setup:
    - Assume that our belief about μ_x is normally distributed with initial distribution (the "prior"):
    - $\mu_x \sim N(\bar{\mu}_x^0, \sigma_x^{2,0})$
    - Now assume we choose to run experiment $x = x^0$ (chosen based on what we know at time 0) and observe W_x^1.
    - Assume that W_x^1 is a noisy observation of a function f(x):

$$W_x^1 = f(x) + \varepsilon^{n+1}$$

      where we might assume $\varepsilon^n \sim N(0, \sigma_W^2)$.
  - » We can show that if the prior belief about μ_x is normally distributed, and we then make a noisy observation that is normally distributed, then the posterior distribution of belief is also normally distributed.

Learning with lookup tables

- Bayesian updating – independent beliefs
  - » It is helpful to introduce the idea of the *precision*. This is just the inverse of the variance.
  - » The precision of our observation noise is

$$\beta^W = \frac{1}{\sigma_W^2}$$

  - » Let $\bar{\sigma}_x^{2,n}$ be our estimate of the variance of μ_x after n experiments. The precision of our belief about μ after n experiments is

$$\beta_x^n = \frac{1}{\bar{\sigma}_x^{2,n}}$$

  - » The updating equations for $\bar{\mu}_x^n$ and β_x^n are given by

$$\bar{\mu}_x^{n+1} = \frac{\beta_x^n \bar{\mu}_x^n + \beta^W w}{\beta_x^n + \beta^W}.$$

$$\beta^{n+1} = \beta^n + \beta^W.$$

Learning with lookup tables

- Notes:
  - » In theory, this only applies if our belief about μ_x is normally distributed, and observation errors are normally distributed.
  - » In practice, we almost *always* use the normal-normal model, because of the central limit theorem.
  - » The reason is that *averages* are approximately normal. That is, an estimate

$$\bar{\mu}^n = \frac{1}{N}\sum_{n=1}^{N} W^n$$

    is a random variable that is approximately normally distributed.
  - » This convergence is *very fast!* An average will be approximately normal for samples as small as 5-10.
  - » To illustrate, if we flip a coin 50 times and total the number of heads (=1) and tails (=0), we will get a number that is normally distributed with mean $50p$, and a variance $50p(1-p)$ where $p = .5$. If we do this 51 times, the total is still normally distributed.

Learning with lookup tables

- Illustration

Measurement				
Variance	133.3333			
Precision	0.0075			
Bayesian updating				
n	W^n	mean	variance	precision
Prior		50.00	900.00	0.001111
1	51	50.87	116.13	0.008611
2	54	52.33	62.07	0.016111
3	70	57.94	42.35	0.023611
4	62	58.92	32.14	0.031111
5	50	57.19	25.90	0.038611
6	79	60.73	21.69	0.046111
7	56	60.07	18.65	0.053611
8	77	62.15	16.36	0.061111
9	68	62.79	14.57	0.068611
10	76	64.09	13.14	0.076111

Learning with lookup tables

- The predictive variance:

$$\tilde{\sigma}^{2,n} = Var^n[\bar{\mu}^{n+1}] \quad (2.9)$$

$$= Var^n[\bar{\mu}^{n+1} - \bar{\mu}^n] \quad (2.10)$$

where $Var^n[\cdot] = Var[\cdot \mid W^1, ..., W^n]$ denotes the variance of the argument given the information we have through n observations. For example,

$$Var^n[\bar{\mu}^n] = 0$$

since, given the information after n observations, $\bar{\mu}^n$ is a number that we can compute deterministically from the prior history of observations.

The parameter $\tilde{\sigma}^{2,n}$ can be described as the variance of $\bar{\mu}^{n+1}$ given the information we have collected through iteration n, which means the only random variable is W^{n+1}. Equivalently, $\tilde{\sigma}^{2,n}$ can be thought of as the *change* in the variance of $\bar{\mu}^n$ as a result of the observation of W^{n+1}. Equation (4.5) is an equivalent statement since, given the information collected up through iteration n, $\bar{\mu}^n$ is deterministic and is therefore a constant. We use equation (4.5) to offer the interpretation that $\tilde{\sigma}^{2,n}$ is the *change* in the variance of our estimate of the mean of μ.

Just as we let $Var^n[\cdot]$ be the variance given what we know after n experiments, let $\mathbb{E}^n$ be the expectation given what we know after n experiments. That is, if $W^1, \ldots, W^n$ are the first n experiments, we can write

$$\mathbb{E}^n \bar{\mu}^{n+1} \equiv \mathbb{E}(\bar{\mu}^{n+1} | W^1, \ldots, W^n) = \bar{\mu}^n.$$

We note in passing that $\mathbb{E}\bar{\mu}^{n+1}$ refers to the expectation before we have made any experiments, which means $W^1, \ldots, W^n$ are all random, as is W^{n+1}. By contrast, when we compute $\mathbb{E}^n \bar{\mu}^{n+1}$, $W^1, \ldots, W^n$ are assumed fixed, and only W^{n+1} is random. By the same token, $\mathbb{E}^{n+1}\bar{\mu}^{n+1} = \bar{\mu}^{n+1}$, where $\bar{\mu}^{n+1}$ is some number which is fixed because we assume that we already know $W^1, \ldots, W^{n+1}$. It is important to realize that when we write an expectation, we have to be explicit about what is random (that is, what variable(s) we are averaging over), and what we are conditioning on.

Using this property, we can write $\bar{\mu}^{n+1}$ in a different way that brings out the role of $\tilde{\sigma}^n$. Assume that we have made n observations and let

$$Z = \frac{\bar{\mu}^{n+1} - \bar{\mu}^n}{\tilde{\sigma}^n}.$$

We note that Z is a random variable only because we have not yet observed W^{n+1}. Normally we would index $Z = Z^{n+1}$ since it is a random variable that depends on W^{n+1}, but we are going to leave this indexing implicit. It is easy to see that $\mathbb{E}^n Z = 0$ and $Var^n Z = 1$. Also, since W^{n+1} is normally distributed, then Z is normally distributed, which means $Z \sim N(0,1)$. This means that we can write

$$\bar{\mu}^{n+1} = \bar{\mu}^n + \tilde{\sigma}^n Z. \tag{2.14}$$

Equation (2.14) makes it clear how $\bar{\mu}^n$ evolves over the observations. It also reinforces the idea that $\tilde{\sigma}^n$ is the change in the variance due to a single observation.

Derivation of posterior distribution for normal-normal model

Optional – not covered in lecture

Derivation of posterior

- Note:
  - » We are going to derive the very simple Bayesian updating formulas for the mean and the precision.
  - » The formulas are simple, but the derivation is a bit messy, and you may feel as if you have landed in the middle of an ORF 309 problem set.
  - » This is the only time that we will do a derivation like this in class.
  - » It will be repeated on one problem set.
  - » It will not be on the midterm.
  - »

Derivation of posterior

2.4 DERIVATION OF BAYESIAN UPDATING EQUATIONS FOR INDEPENDENT BELIEFS*

Bayesian analysis begins with a simple formula that everyone learns in their first probability course. Given events A and B, the basic properties of conditional probability imply

$$P(A, B) = P(A|B)P(B) = P(B|A)P(A)$$

which implies

$$P(B|A) = \frac{P(A|B)P(B)}{P(A)}.$$

This expression is famously known as Bayes theorem. In a learning setting, the event A refers to a measurement (or some type of new information), while B refers to the event that a parameter (say, the mean of a distribution) takes on a particular value. $P(B)$ refers to our initial (or prior) distribution of belief about the unknown parameter before we make a measurement, and $P(B|A)$ is the distribution of belief

Derivation of posterior

We can apply the same idea for continuous variables. We replace B with the event that $\mu = u$ (to be more precise, we replace B with the event that $u \leq \mu \leq u + du$), and A with the event that we observed $W = w$. Let $g(u)$ be our prior distribution of belief about the mean μ, and let $g(u|w)$ be the posterior distribution of belief about μ given that we observed $W = w$. We then let $f(w|u)$ be the distribution of the random variable W if $\mu = u$. We can now write our posterior $g(u|w)$, which is the density of μ given that we observe $W = w$, as

$$g(u|w) = \frac{f(w|u)g(u)}{f(w)}, \qquad \textit{Easy}$$

where $f(w)$ is the unconditional density of the random variable W which we compute using

$$f(w) = \int_u f(w|u)g(u). \qquad \textit{Hard}$$

Equation (2.46) gives us the density of μ given that we have observed $W = w$.

Derivation of posterior

We illustrate these calculations by assuming that our prior $g(u)$ follows the normal distribution with mean μ^0 and variance $\sigma^{2,0}$, given by

$$g(u) = \frac{1}{\sqrt{2\pi}\sigma^0} \exp\left(-\frac{1}{2}\frac{(u-\mu^0)^2}{\sigma^{2,0}}\right).$$

We further assume that the observation W is also normally distributed with mean μ and variance σ_ϵ^2, which is sometimes referred to as the measurement or observation error. The conditional distribution $f(w|u)$ is

$$f(w|u) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2}\frac{(w-u)^2}{\sigma_\epsilon^2}\right).$$

We can compute $f(w)$ from $f(w|u)$ and $g(u)$, but it is only really necessary to find the density $g(u|w)$ up to a normalization constant ($f(w)$ is part of this normalization constant). For this reason, we can write

$$g(u|w) \propto f(w|u)g(u). \tag{2.47}$$

Using this reasoning, we can drop coefficients such as $\frac{1}{\sqrt{2\pi}\sigma^0}$, and write

$$\begin{aligned} g(u|w) &\propto \left[\exp\left(-\frac{1}{2}\frac{(w-u)^2}{\sigma^2}\right)\right] \cdot \left[\exp\left(-\frac{1}{2}\frac{(u-\mu^0)^2}{\sigma^{2,0}}\right)\right], \\ &\propto \exp\left[-\frac{1}{2}\left(\frac{(w-u)^2}{\sigma^2} + \frac{(u-\mu^0)^2}{\sigma^{2,0}}\right)\right]. \end{aligned} \tag{2.48}$$

Derivation of posterior

$$\propto exp\left[\frac{-1}{2}(\beta^W(w-u)^2+\beta^2(u-u^0)^2)\right]$$

$$\propto exp\left[\frac{-1}{2}(\beta^W w^2-2\beta^W wu+\beta^W u^2+\beta^0 u^2-2\beta^0 uu^0+\beta^0(u^0)^2)\right]$$

$$\propto exp\left[\frac{-1}{2}(\beta^W+\beta^0)u^2-2(\beta^W w+\beta^0 u^0)u+\ \ldots(constant)\right]$$

$$\propto exp\left[\frac{-1}{2}(\beta^W+\beta^0)\left(u^2-2\frac{(\beta^W w+\beta^0 u^0)}{(\beta^W+\beta^0)}u+\left(\frac{(\beta^W w+\beta^0 u^0)}{(\beta^W+\beta^0)}\right)^2\right)\ \right]$$

$$\propto exp\left[\frac{-1}{2}(\beta^W+\beta^0)(u-u^1)^2\right]$$

Where: $u^1=\frac{\beta^W w+\beta^0 u^0}{\beta^W+\beta^0}$. Now let $\beta^1=\beta^0+\beta^W$.

Derivation of posterior

Where: $u^1 = \frac{\beta^W w + \beta^0 u^0}{\beta^W + \beta^0}$. Now let $\beta^1 = \beta^0 + \beta^W$. This gives

$$\propto exp\left[\frac{-1}{2}(\beta^W + \beta^0)(u - u^1)^2\right]$$

$$\propto exp\left[\frac{-1}{2}\beta^1(u - u^1)^2\right]$$

This looks just like a normal distribution:

$$\frac{1}{\sqrt{2\pi}\sigma} exp \frac{-1}{2}\left(\frac{(x-\mu)^2}{\sigma^2}\right)$$

Now convert to precisions

$$\sqrt{\frac{\beta}{2\pi}} exp \frac{-1}{2}\beta(x-\mu)^2$$

This gives us our normalizing constant, so our posterior distribution is

$$g(u|w) = \sqrt{\frac{\beta^1}{2\pi}} exp \frac{-1}{2}\beta^1(w-u)^2$$

… which is the density for a normal distribution.

Derivation of posterior

- Notes:
  - » The “normal-normal” updating equations tend to be used most of the time regardless of the underlying distributions of the random variables, for two reasons:
  - » 1) The distribution of an average is approximately normal due to the central limit theorem. In fact, the normal is a good approximation of an average when you have samples as small as 5 to 10 observations.
  - » 2) If you have a normal prior (see (1) above) and you then observe some random outcome (this could even be 0 or 1) is normal (see (1) above).
  - » One exception is when you are observing success/failure (0/1) outcomes, and the number of observations is small. In this case, we would use a beta-Bernoulli distribution (we will do this later)

Lookup tables

Correlated beliefs

Lookup tables – correlated beliefs

- Bayesian updating – correlated beliefs

EXAMPLE 3.1

We are interested in finding the price of a product that maximizes total revenue. We believe that the function $R(p)$ that relates revenue to price is continuous. Assume that we set a price p^n and observe revenue R^{n+1} that is higher than we had expected. If we raise our estimate of the function $R(p)$ at the price p^n, our beliefs about the revenue at nearby prices should be higher.

EXAMPLE 3.2

We choose five people for the starting lineup of our basketball team and observe total scoring for one period. We are trying to decide if this group of five people is better than another lineup that includes three from the same group with two different people. If the scoring of these five people is higher than we had expected, we would probably raise our belief about the other group, since there are three people in common.

EXAMPLE 3.3

A physician is trying to treat diabetes using a treatment of three drugs, where she observes the drop in blood sugar from a course of a particular treatment. If one treatment produces a better-than-expected response, this would also increase our belief of the response from other treatments that have one or two drugs in common.

Lookup tables – correlated beliefs

- Bayesian updating – correlated beliefs

Let μ_x^n be our belief about alternative x after n measurements. Now let

$$Cov^n(\mu_x, \mu_y) \quad = \quad \text{the covariance in our belief about } \mu_x \text{ and } \mu_y.$$

We let Σ^n be the covariance matrix, with element $\Sigma_{xy}^n = Cov^n(\mu_x, \mu_y)$. Just as we defined the precision β_x^n to be the reciprocal of the variance, we are going to define the precision matrix B^n to be

$$B^n = (\Sigma^n)^{-1}.$$

Let e_x be a column vector of zeroes with a 1 for element x, and as before we let W^{n+1} be the (scalar) observation when we decide to measure alternative x. We could label W^{n+1} as W_x^{n+1} to make the dependence on the alternative more explicit. For this discussion, we are going to use the notation that we choose to measure x^n and the resulting observation is W^{n+1}. If we choose to measure x^n, we can also interpret the observation as a column vector given by $W^{n+1}e_{x^n}$. Keeping in mind that μ^n is a column vector of our beliefs about the expectation of μ, the Bayesian equation for updating this vector in the presence of correlated beliefs is given by

$$\mu^{n+1} \quad = \quad (B^{n+1})^{-1}\left(B^n\mu^n + \beta^W W^{n+1}e_{x^n}\right), \tag{3.17}$$

where B^{n+1} is given by

$$B^{n+1} \quad = \quad (B^n + \beta^W e_{x^n}(e_{x^n})^T). \tag{3.18}$$

Lookup tables – correlated beliefs

- Bayesian updating – correlated beliefs

Let $\lambda^W = \sigma_W^2 = 1/\beta^W$ be the variance of our measurement W^{n+1}. We are going to simplify our notation by assuming that our measurement variance is the same across all alternatives x, but if this is not the case, we can replace λ^W with λ_x^W throughout. Using the Sherman-Morrison formula, and letting $x = x^n$, we can rewrite the updating equations as

$$\mu^{n+1}(x) = \mu^n + \frac{W^{n+1} - \mu_x^n}{\lambda^W + \Sigma_{xx}^n}\Sigma^n e_x, \tag{3.20}$$

$$\Sigma^{n+1}(x) = \Sigma^n - \frac{\Sigma^n e_x (e_x)^T \Sigma^n}{\lambda^W + \Sigma_{xx}^n}. \tag{3.21}$$

where we express the dependence of $\mu^{n+1}(x)$ and $\Sigma^{n+1}(x)$ on the alternative x which we have chosen to measure.

To illustrate, assume that we have three alternatives with mean vector

$$\mu^n = \begin{bmatrix} 20 \\ 16 \\ 22 \end{bmatrix}.$$

Assume that $\lambda^W = 9$ and that our covariance matrix Σ^n is given by

$$\Sigma^n = \begin{bmatrix} 12 & 6 & 3 \\ 6 & 7 & 4 \\ 3 & 4 & 15 \end{bmatrix}.$$

Lookup tables – correlated beliefs

- Bayesian updating – correlated beliefs

Assume that we choose to measure $x = 3$ and observe $W^{n+1} = W_3^{n+1} = 19$. Applying equation (3.20), we update the means of our beliefs using

$$\begin{aligned}
\mu^{n+1}(3) &= \begin{bmatrix} 20 \\ 16 \\ 22 \end{bmatrix} + \frac{19-22}{9+15} \begin{bmatrix} 12 & 6 & 3 \\ 6 & 7 & 4 \\ 3 & 4 & 15 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \\
&= \begin{bmatrix} 20 \\ 16 \\ 22 \end{bmatrix} + \frac{-3}{24} \begin{bmatrix} 3 \\ 4 \\ 15 \end{bmatrix} \\
&= \begin{bmatrix} 19.625 \\ 15.500 \\ 20.125 \end{bmatrix}.
\end{aligned}$$

Lookup tables – correlated beliefs

- Bayesian updating – correlated beliefs

The update of the covariance matrix is computed using

$$\Sigma^{n+1}(3) = \begin{bmatrix} 12 & 6 & 3 \\ 6 & 7 & 4 \\ 3 & 4 & 15 \end{bmatrix} - \frac{\begin{bmatrix} 12 & 6 & 3 \\ 6 & 7 & 4 \\ 3 & 4 & 15 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} [0\ 0\ 1] \begin{bmatrix} 12 & 6 & 3 \\ 6 & 7 & 4 \\ 3 & 4 & 15 \end{bmatrix}}{9+15}$$

$$= \begin{bmatrix} 12 & 6 & 3 \\ 6 & 7 & 4 \\ 3 & 4 & 15 \end{bmatrix} - \frac{1}{24} \begin{bmatrix} 3 \\ 4 \\ 15 \end{bmatrix} [3\ 4\ 15]$$

$$= \begin{bmatrix} 12 & 6 & 3 \\ 6 & 7 & 4 \\ 3 & 4 & 15 \end{bmatrix} - \frac{1}{24} \begin{bmatrix} 9 & 12 & 45 \\ 12 & 16 & 60 \\ 45 & 60 & 225 \end{bmatrix}$$

$$= \begin{bmatrix} 12 & 6 & 3 \\ 6 & 7 & 4 \\ 3 & 4 & 15 \end{bmatrix} - \begin{bmatrix} 0.375 & 0.500 & 1.875 \\ 0.500 & 0.667 & 2.500 \\ 1.875 & 2.500 & 9.375 \end{bmatrix}$$

$$= \begin{bmatrix} 11.625 & 5.500 & 1.125 \\ 5.500 & 6.333 & 1.500 \\ 1.125 & 1.500 & 5.625 \end{bmatrix}.$$

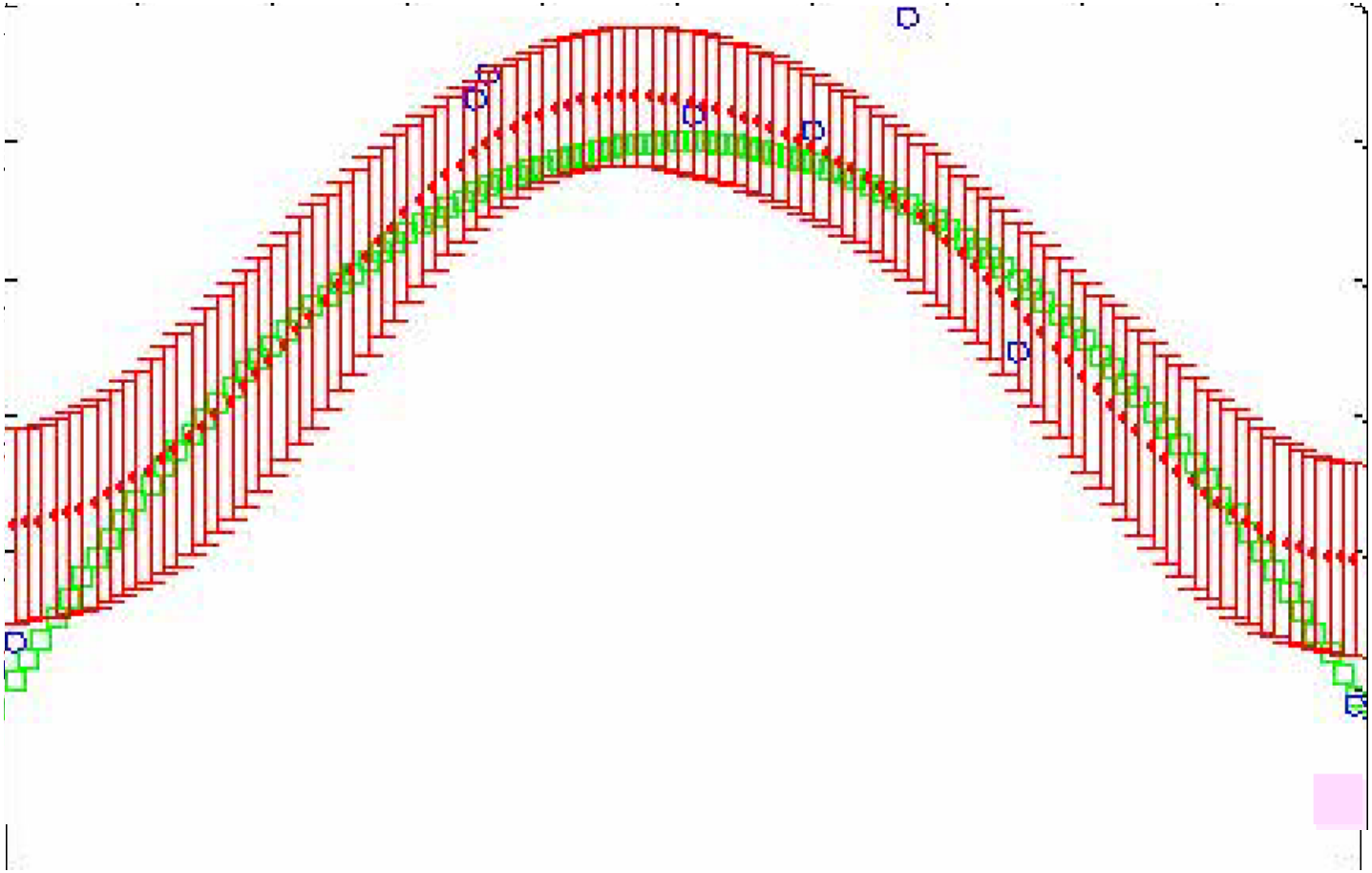
Lookup tables – correlated beliefs

- Estimating continuous surfaces
  - » Assume $x = \{x_1, \ldots, x_K\}$ is a continuous vector, and $f(x)$ is a continuous function. Let $\mu(x)$ be the random variable describing our belief about $f(x)$.
  - » It is reasonable to assume that our belief about $f(x)$ and $f(x')$ is correlated as a function of the distance $\left|\left|x - x'\right|\right| = \sqrt{\sum_{k=1}^{K}(x_k - x'_k)^2}$.
  - » Let $Cov(\mu_x, \mu_{x'})$ be the covariance between our beliefs about μ_x and $\mu_{x'}$. It is reasonable to assume that

$$Cov(\mu_x, \mu_{x'}) = \sigma^2 \exp^{\beta||x-x'||}$$

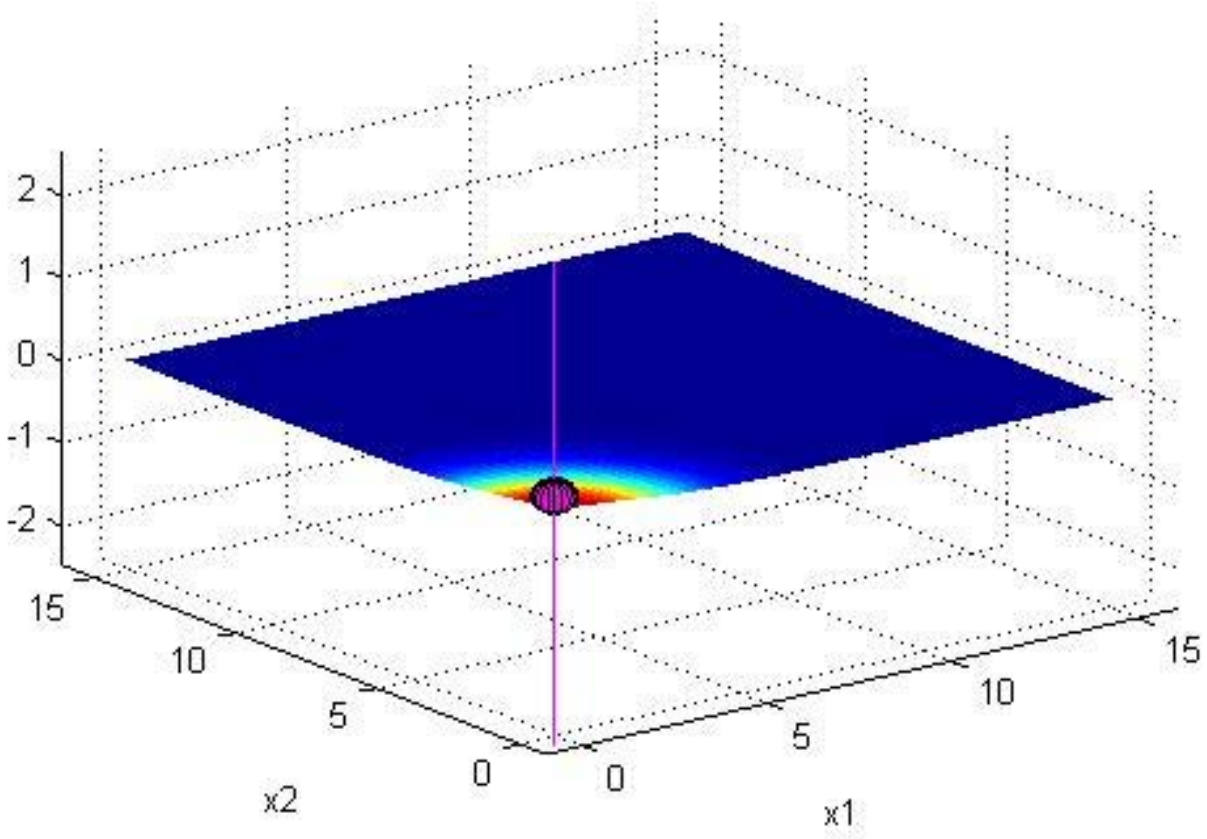
  - » If $x = x'$

Lookup tables – correlated beliefs



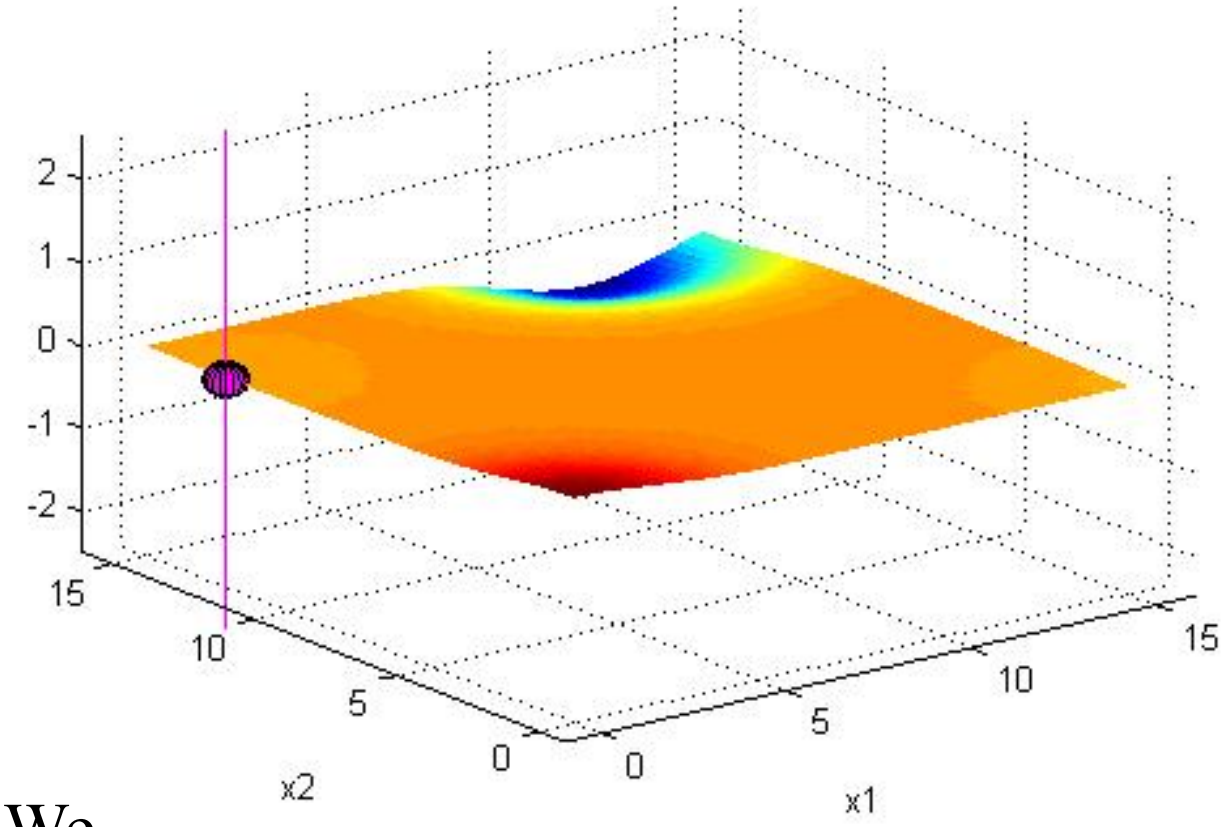
Lookup tables – correlated beliefs

- Learning a two-dimensional surface



Lookup tables – correlated beliefs

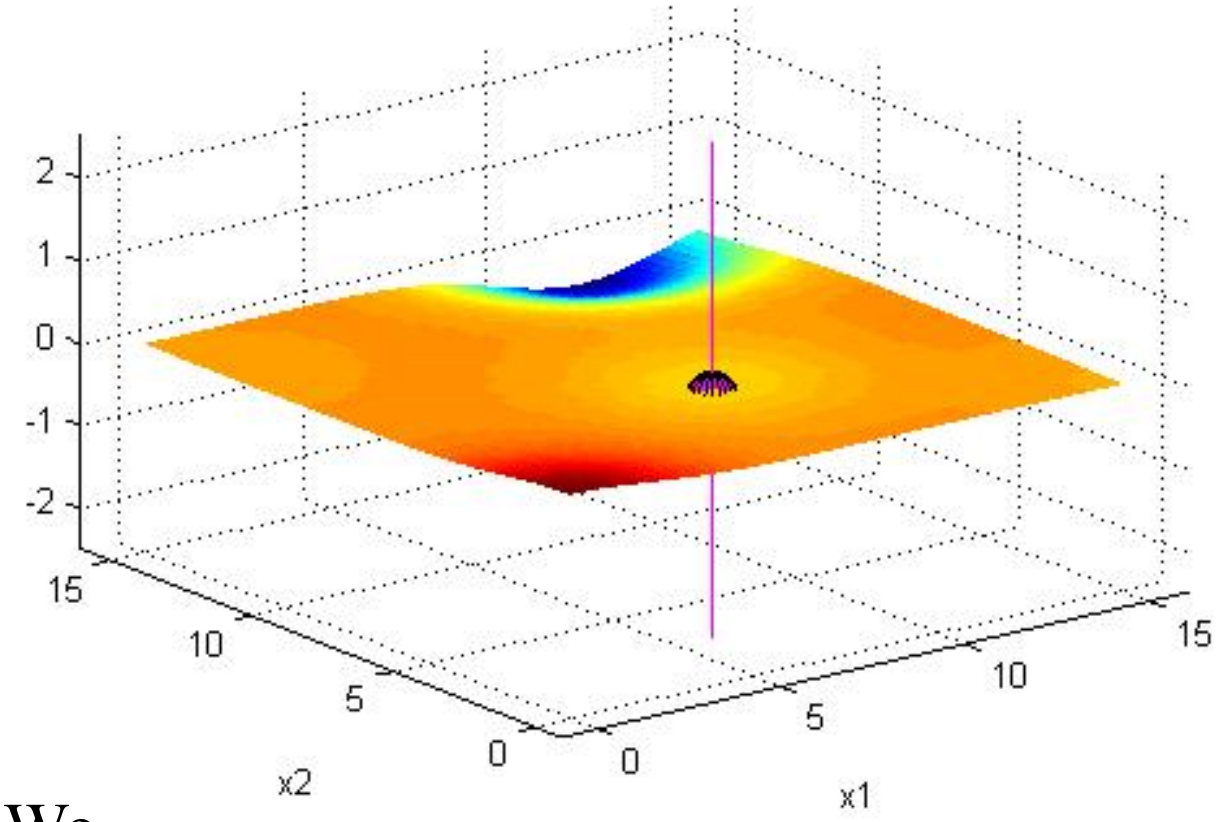
- Learning a two-dimensional surface



» We

Lookup tables – correlated beliefs

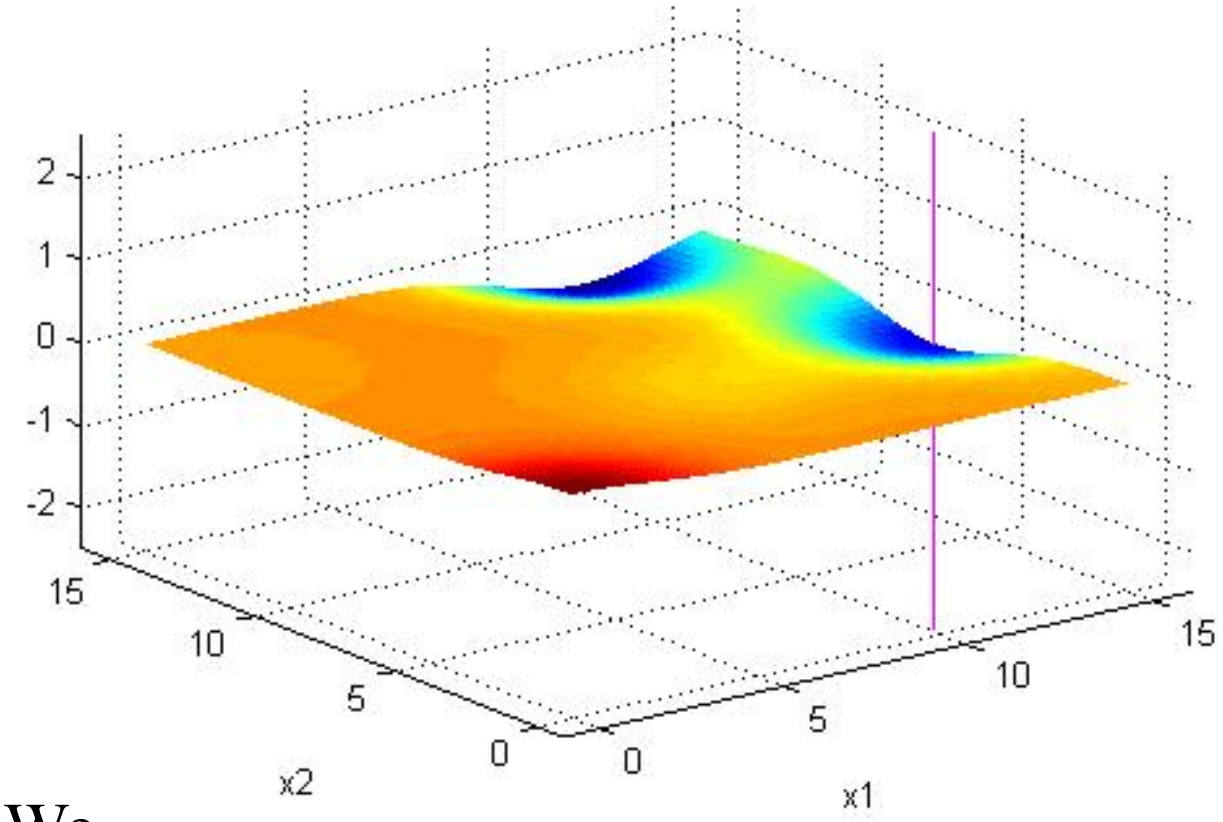
- Learning a two-dimensional surface



» We

Lookup tables – correlated beliefs

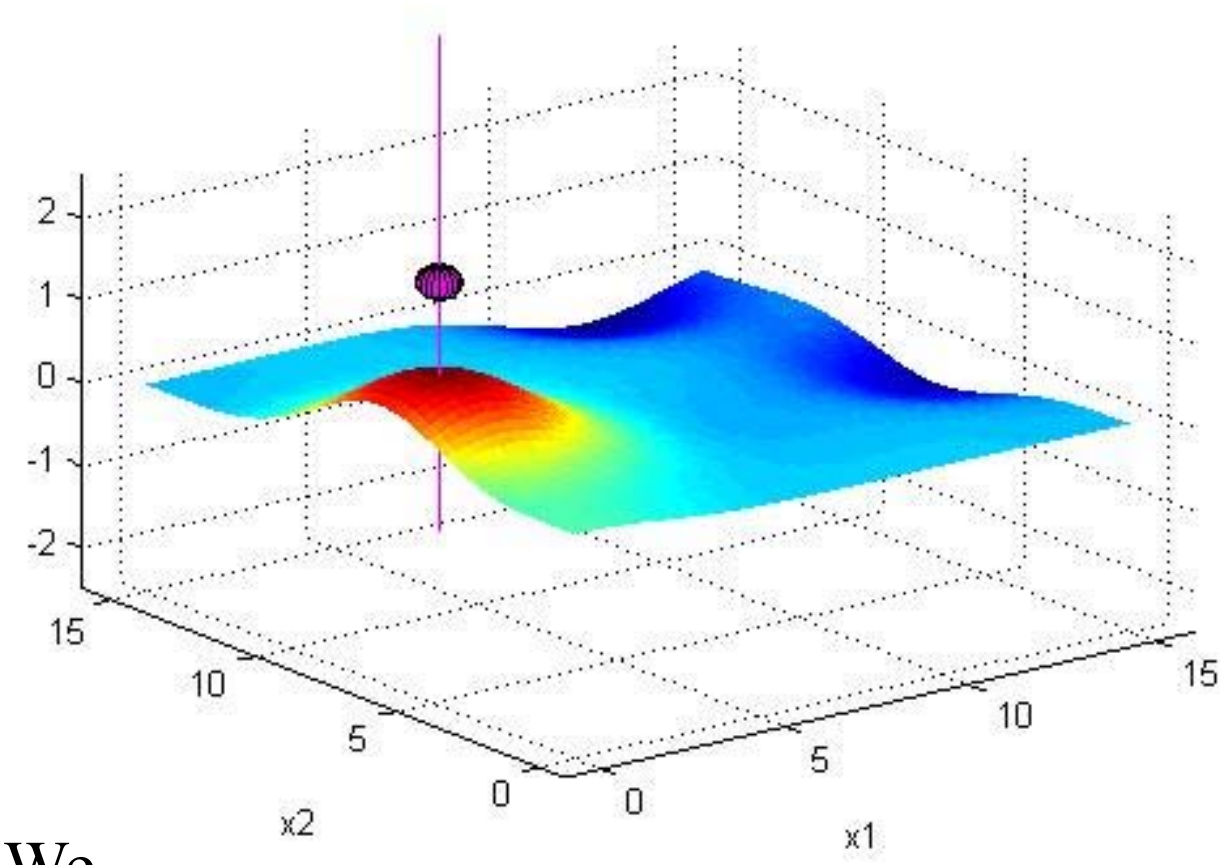
- Learning a two-dimensional surface



» We

Lookup tables – correlated beliefs

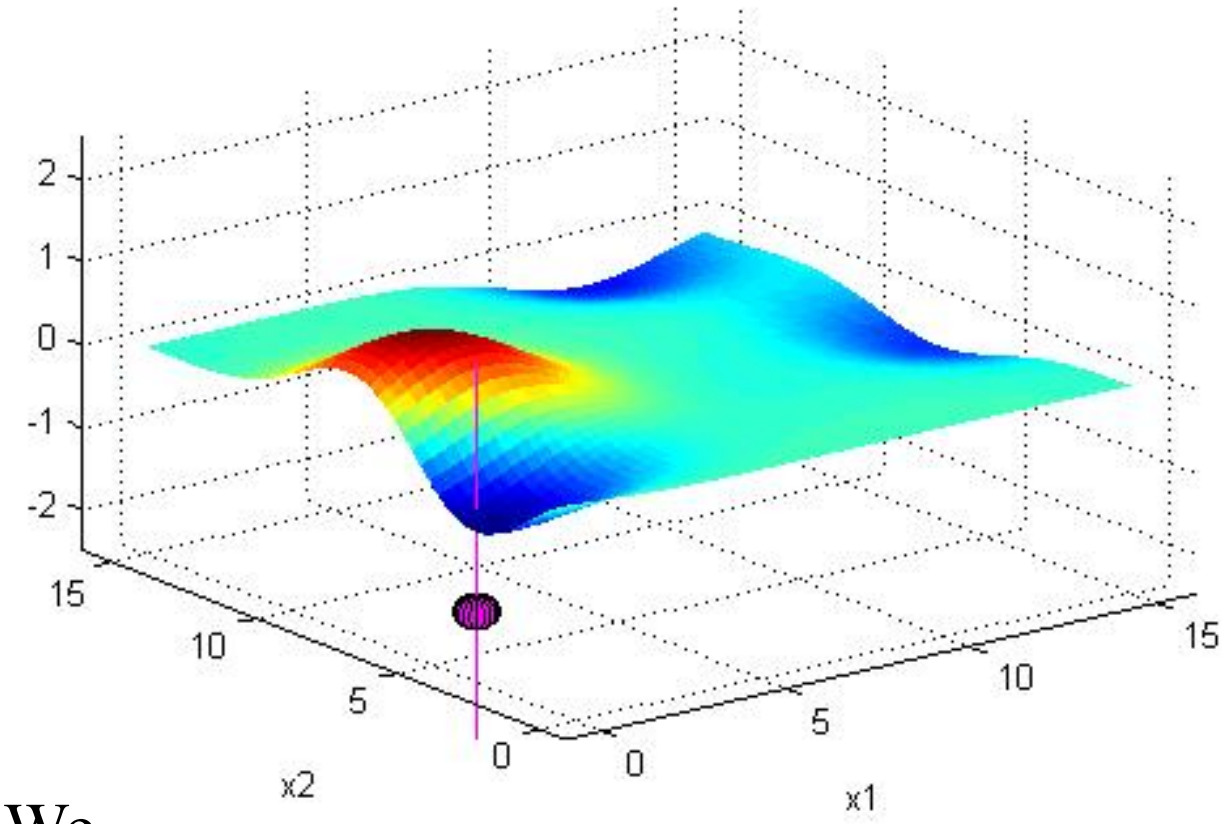
- Learning a two-dimensional surface



» We

Lookup tables – correlated beliefs

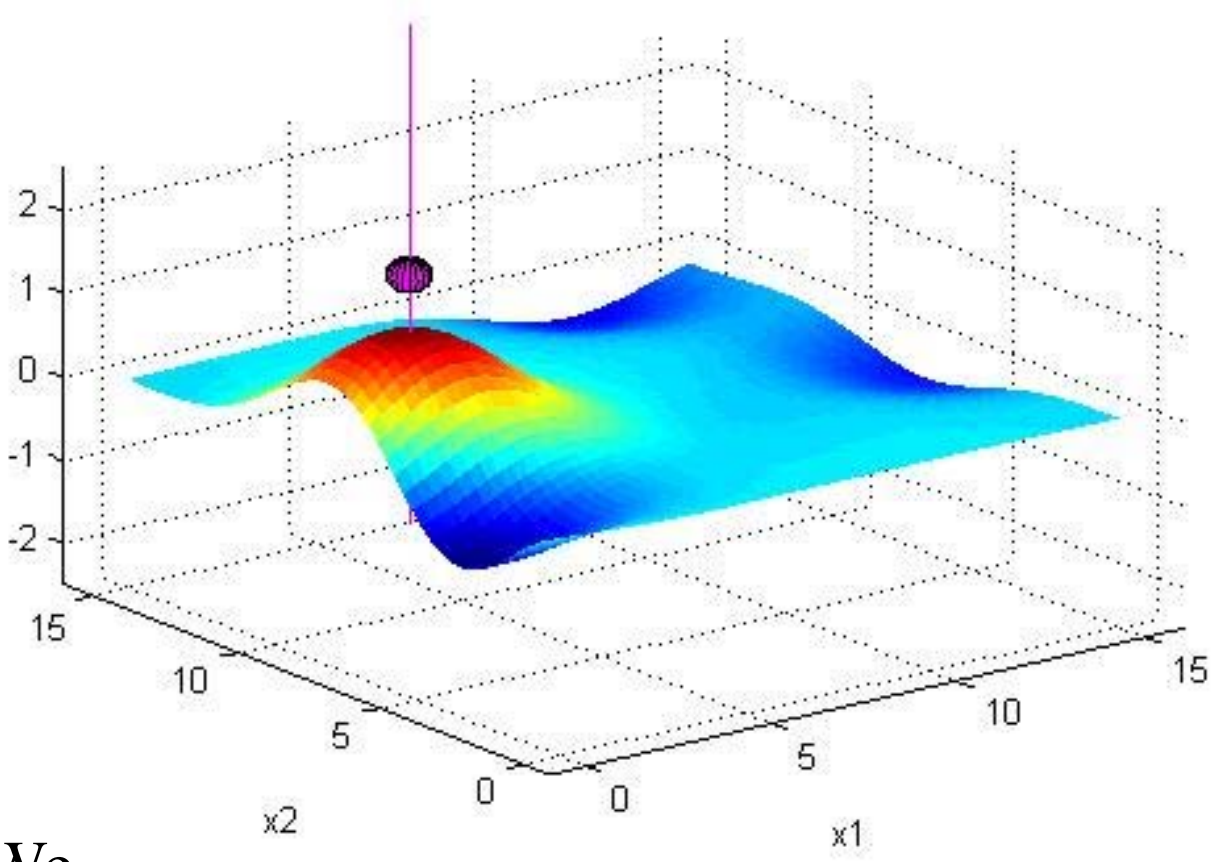
- Learning a two-dimensional surface



» We

Lookup tables – correlated beliefs

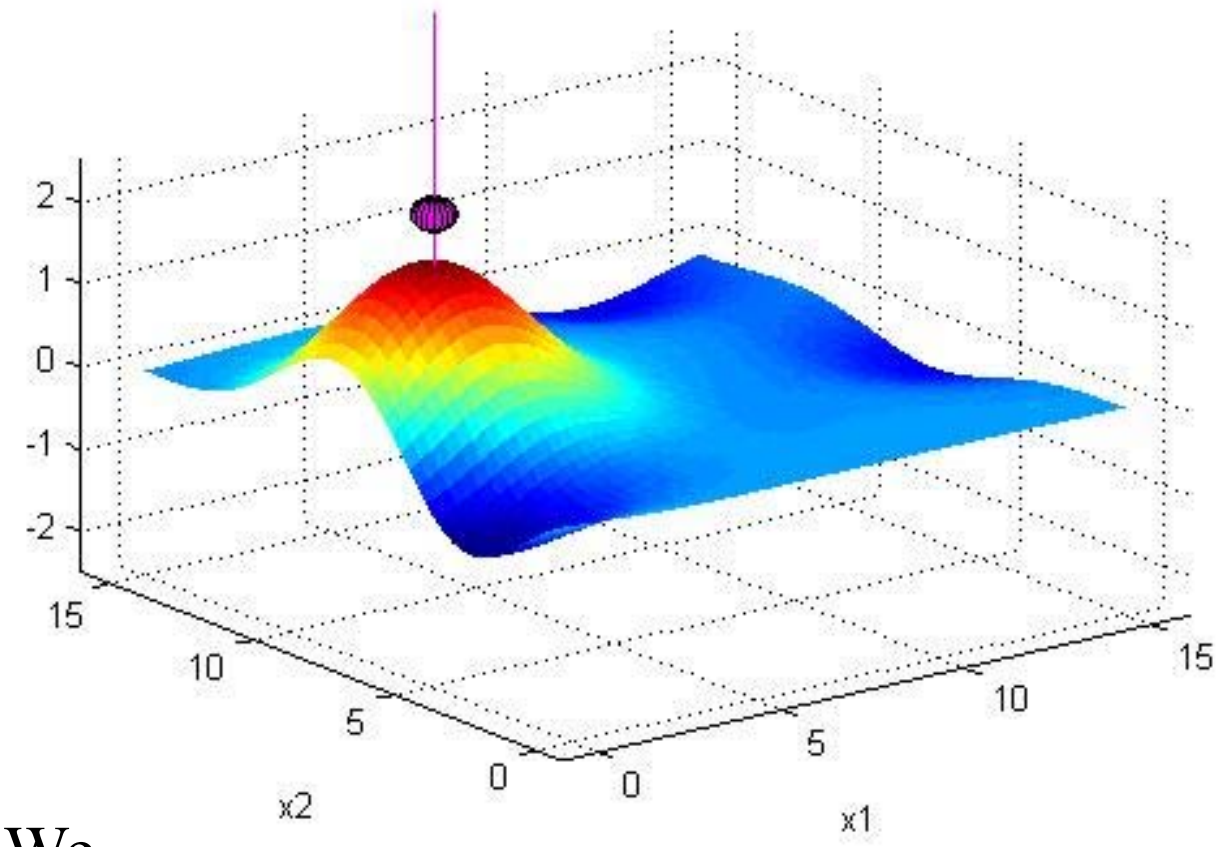
- Learning a two-dimensional surface



» We

Lookup tables – correlated beliefs

- Learning a two-dimensional surface



» We

Lookup tables – correlated beliefs

- Notes:
  - » Correlated beliefs is a powerful way of estimating continuous surfaces with a relatively small number of observations.
  - » These updates require the use of a "distance function" that measures the distance between two points on the surface. Care needs to be used if you have more than about 5 dimensions.
  - » This logic can be used for categorical data:
    - Consumer choices based on age, location, gender, …
    - Performance of different materials that share characteristics.
    - Behavior of a drug as a function of patient characteristics.

Lookup tables

Derivation of posterior for normal-normal model. We will not cover this in lecture.

2.4 DERIVATION OF BAYESIAN UPDATING EQUATIONS FOR INDEPENDENT BELIEFS*

Bayesian analysis begins with a simple formula that everyone learns in their first probability course. Given events A and B, the basic properties of conditional probability imply

$$P(A, B) = P(A|B)P(B) = P(B|A)P(A)$$

which implies

$$P(B|A) = \frac{P(A|B)P(B)}{P(A)}.$$

This expression is famously known as Bayes theorem. In a learning setting, the event A refers to a measurement (or some type of new information), while B refers to the event that a parameter (say, the mean of a distribution) takes on a particular value. $P(B)$ refers to our initial (or prior) distribution of belief about the unknown parameter before we make a measurement, and $P(B|A)$ is the distribution of belief

We can apply the same idea for continuous variables. We replace B with the event that $\mu = u$ (to be more precise, we replace B with the event that $u \leq \mu \leq u + du$), and A with the event that we observed $W = w$. Let $g(u)$ be our prior distribution of belief about the mean μ, and let $g(u|w)$ be the posterior distribution of belief about μ given that we observed $W = w$. We then let $f(w|u)$ be the distribution of the random variable W if $\mu = u$. We can now write our posterior $g(u|w)$, which is the density of μ given that we observe $W = w$, as

$$g(u|w) = \frac{f(w|u)g(u)}{f(w)}, \qquad \textit{Easy}$$

where $f(w)$ is the unconditional density of the random variable W which we compute using

$$f(w) = \int_u f(w|u)g(u). \qquad \textit{Hard}$$

Equation (2.46) gives us the density of μ given that we have observed $W = w$.

We illustrate these calculations by assuming that our prior $g(u)$ follows the normal distribution with mean μ^0 and variance $\sigma^{2,0}$, given by

$$g(u) = \frac{1}{\sqrt{2\pi}\sigma^0} \exp\left(-\frac{1}{2}\frac{(u-\mu^0)^2}{\sigma^{2,0}}\right).$$

We further assume that the observation W is also normally distributed with mean μ and variance σ_ϵ^2, which is sometimes referred to as the measurement or observation error. The conditional distribution $f(w|u)$ is

$$f(w|u) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2}\frac{(w-u)^2}{\sigma_\epsilon^2}\right).$$

We can compute $f(w)$ from $f(w|u)$ and $g(u)$, but it is only really necessary to find the density $g(u|w)$ up to a normalization constant ($f(w)$ is part of this normalization constant). For this reason, we can write

$$g(u|w) \propto f(w|u)g(u). \tag{2.47}$$

Using this reasoning, we can drop coefficients such as $\frac{1}{\sqrt{2\pi}\sigma^0}$, and write

$$\begin{aligned} g(u|w) &\propto \left[\exp\left(-\frac{1}{2}\frac{(w-u)^2}{\sigma^2}\right)\right] \cdot \left[\exp\left(-\frac{1}{2}\frac{(u-\mu^0)^2}{\sigma^{2,0}}\right)\right], \\ &\propto \exp\left[-\frac{1}{2}\left(\frac{(w-u)^2}{\sigma^2} + \frac{(u-\mu^0)^2}{\sigma^{2,0}}\right)\right]. \end{aligned} \tag{2.48}$$

- Do the algebra….
  - » Replace variances with precisions.
  - » Expand 2.48, obtaining six terms.
  - » Collect terms involving u^2 – this will produce $\beta^1 u$. Note that u^2 is "u squared" while β^n and μ^n for n=0, 1, …, represent estimates at time n.
  - » Collect terms involving "-2" – this will produce the term $\beta^1 \cdot u \cdot \mu^1$
  - » Remaining terms are not a function of u, so you can just add whatever constant is needed to fill out the rest of the quadratic. The constant goes in the proportionality constant.

After some algebra, we find that

$$g(u|w) \propto \exp -\frac{1}{2}\beta^1(u-\mu^1)^2 \tag{2.49}$$

where

$$\mu^1 = \frac{(\beta^W w + \beta^0\mu^0)}{\alpha+\beta^0}, \tag{2.50}$$

$$\beta^1 = \beta^W + \beta^0. \tag{2.51}$$

The next step is to find the normalization constant (call it K) which we do by solving

$$K\int_u g(u|w)du = 1.$$

We could find the normalization constant by solving the integral and picking K so that $g(u|w)$ integrates to 1, but there is an easier way. What we are going to do is look around for a known probability density function with the same structure as (2.49), and then simply use its normalization constant. It is fairly easy to see that (2.49) corresponds to a normal distribution, which means that the normalization constant $K = \sqrt{\frac{\beta^1}{2\pi}}$. This means that our posterior density is given by

$$g(u|w) = \sqrt{\frac{\beta^1}{2\pi}} \exp -\frac{1}{2}\beta^1(u-\mu^1)^2. \tag{2.52}$$

Bayesian updating for sampled models

- Notes:
  - » The "normal-normal" updating equations tend to be used most of the time regardless of the underlying distributions of the random variables, for two reasons:
    - 1) The distribution of an average is approximately normal due to the central limit theorem. In fact, the normal is a good approximation of an average when you have samples as small as 5 to 10 observations.
    - 2) If you have a normal prior (see (1) above) and you then observe some random outcome (this could even be 0 or 1) is normal (see (1) above).
  - » One exception is when you are observing success/failure (0/1) outcomes, and the number of observations is small. In this case, we would use a beta-Bernoulli distribution (we will do this later).

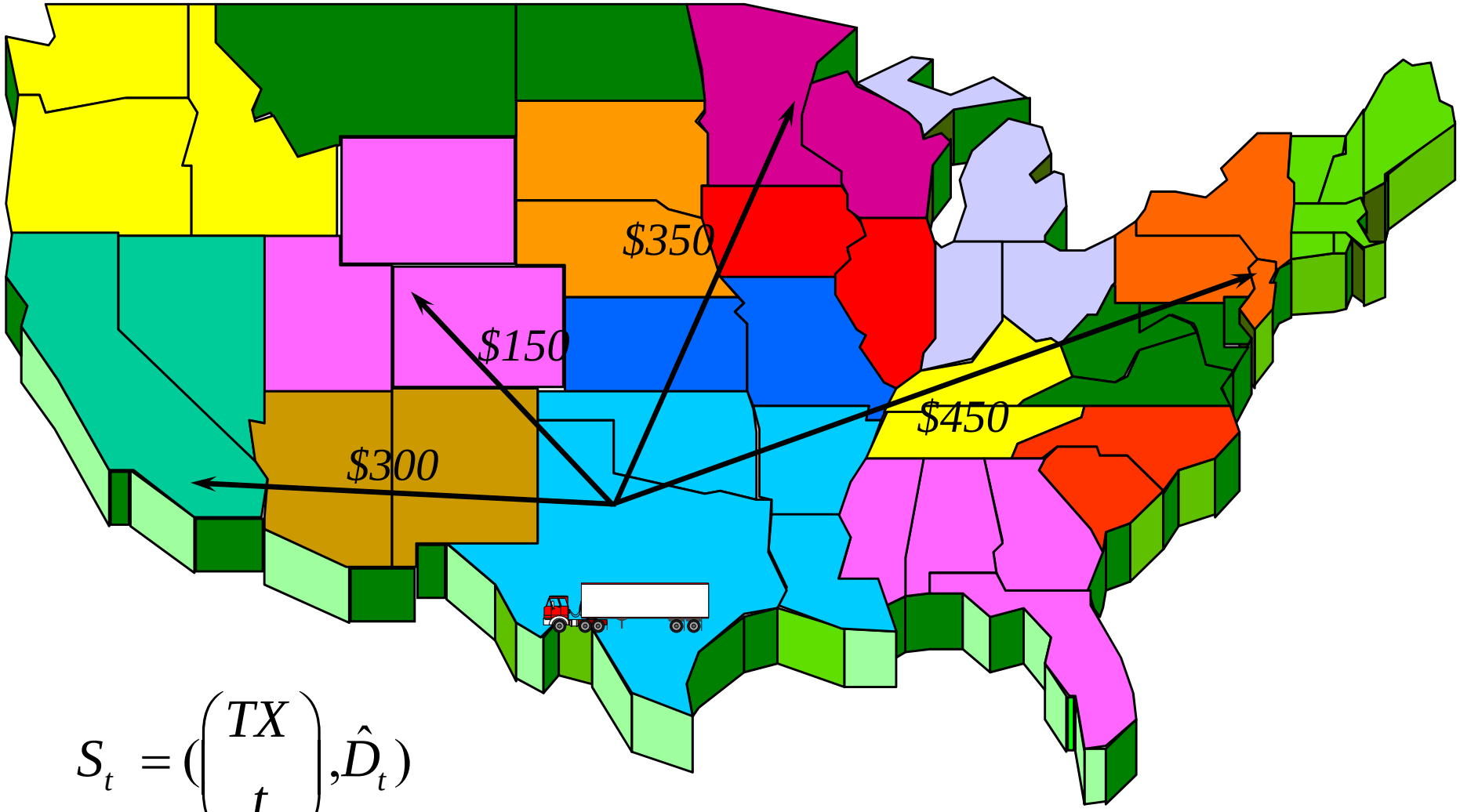
Lookup tables

Hierarchical learning

(I will just skim this in lecture)

Approximate dynamic programming

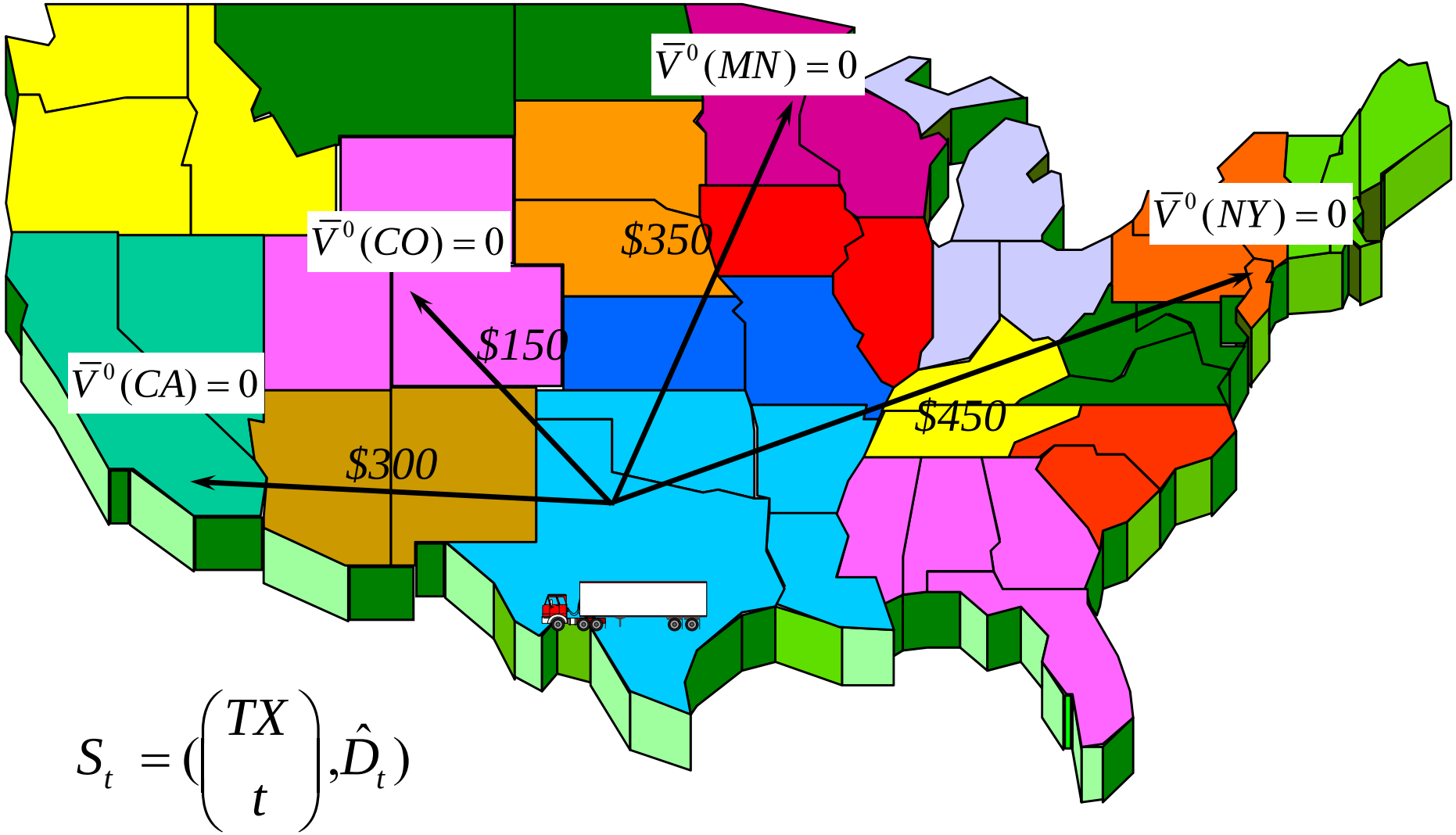
- Pre-decision state: we see the demands



$$S_t = \left(\begin{pmatrix} TX \\ t \end{pmatrix}, \hat{D}_t\right)$$

Approximate dynamic programming

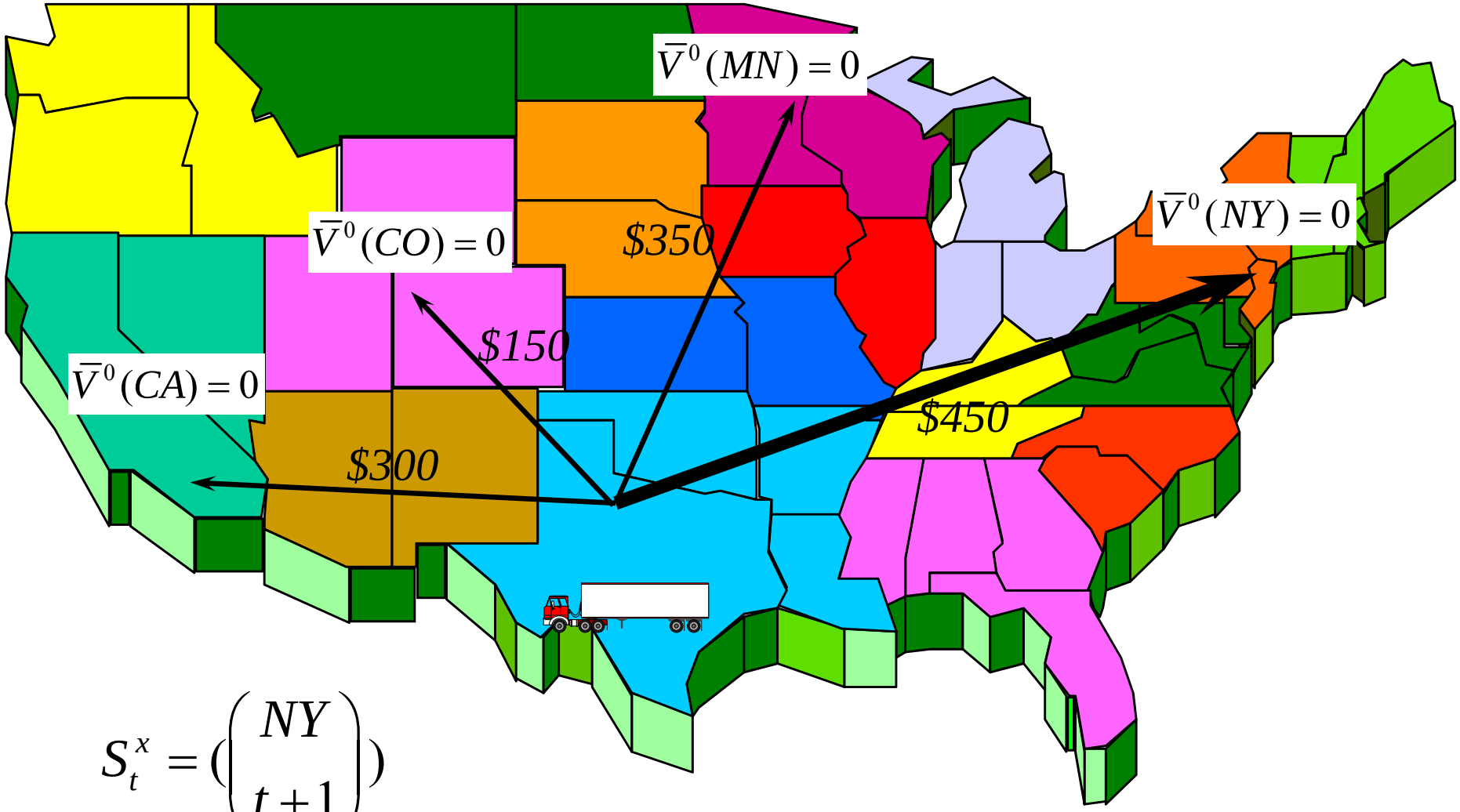
- We use initial value function approximations…



$$S_t = \left(\begin{pmatrix} TX \\ t \end{pmatrix}, \hat{D}_t\right)$$

Approximate dynamic programming

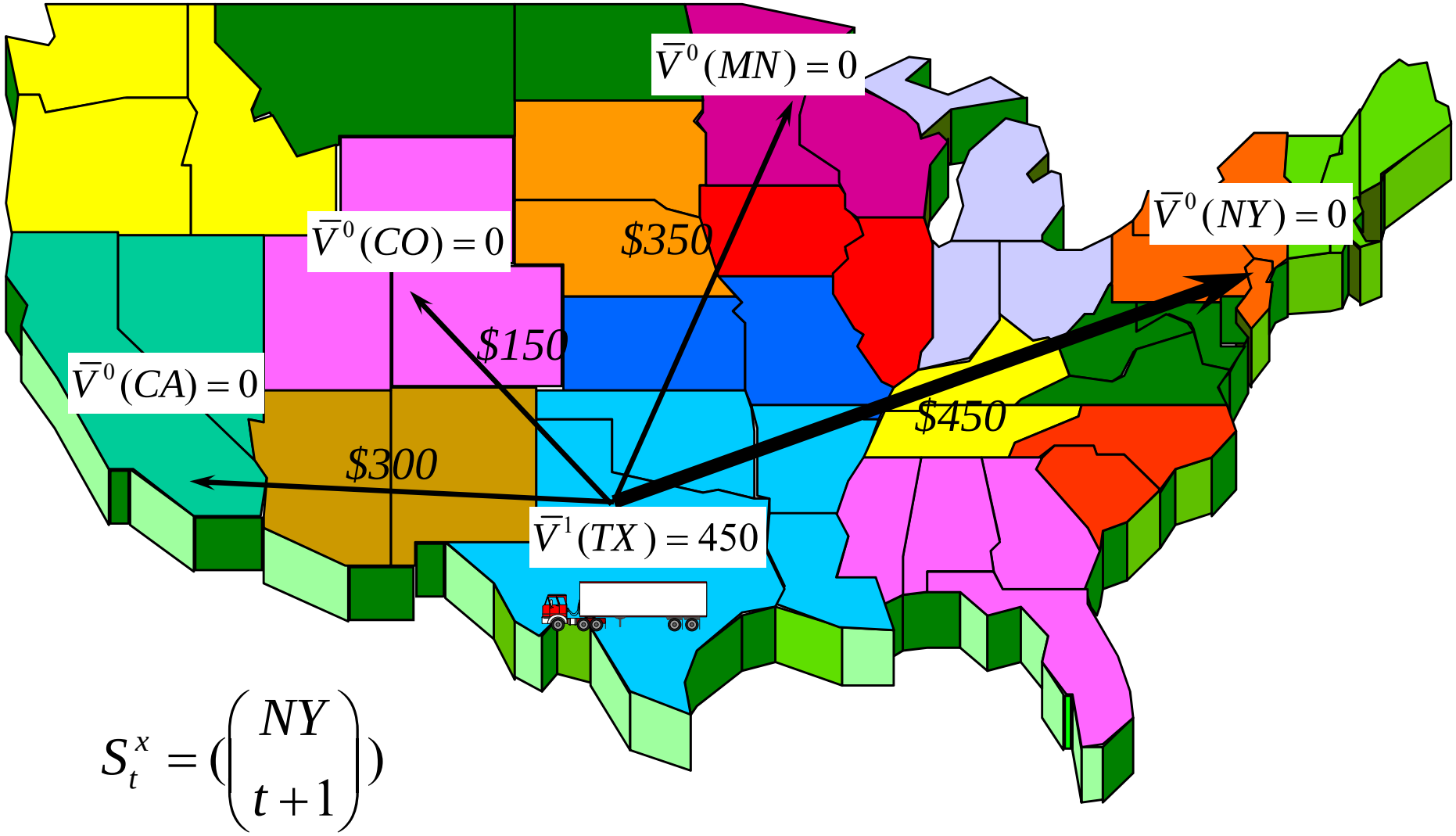
- … and make our first choice: x^1



$$S_t^x = \begin{pmatrix} NY \\ t+1 \end{pmatrix}$$

Approximate dynamic programming

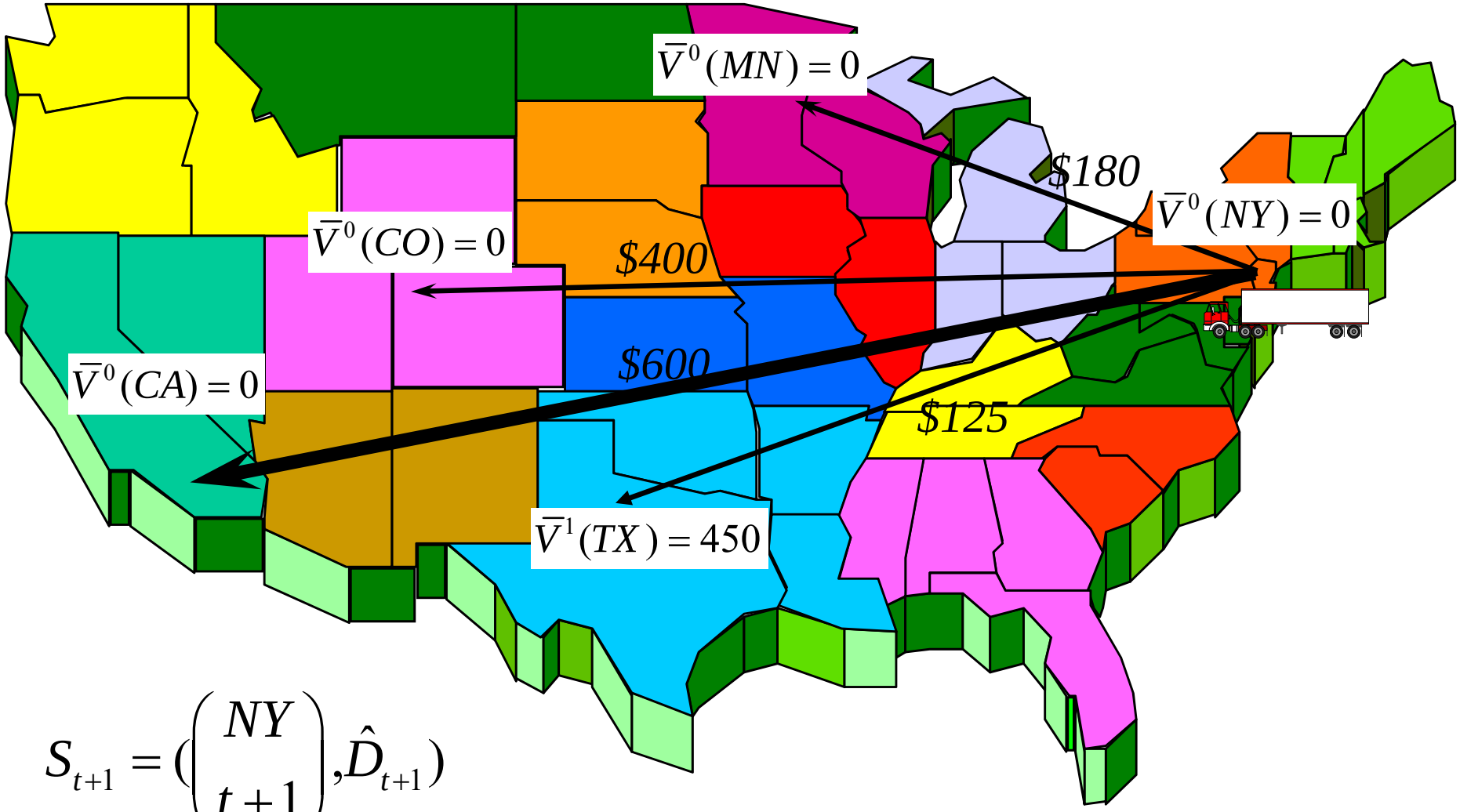
- Update the value of being in Texas.



$$S_t^x = \begin{pmatrix} NY \\ t+1 \end{pmatrix}$$

Approximate dynamic programming

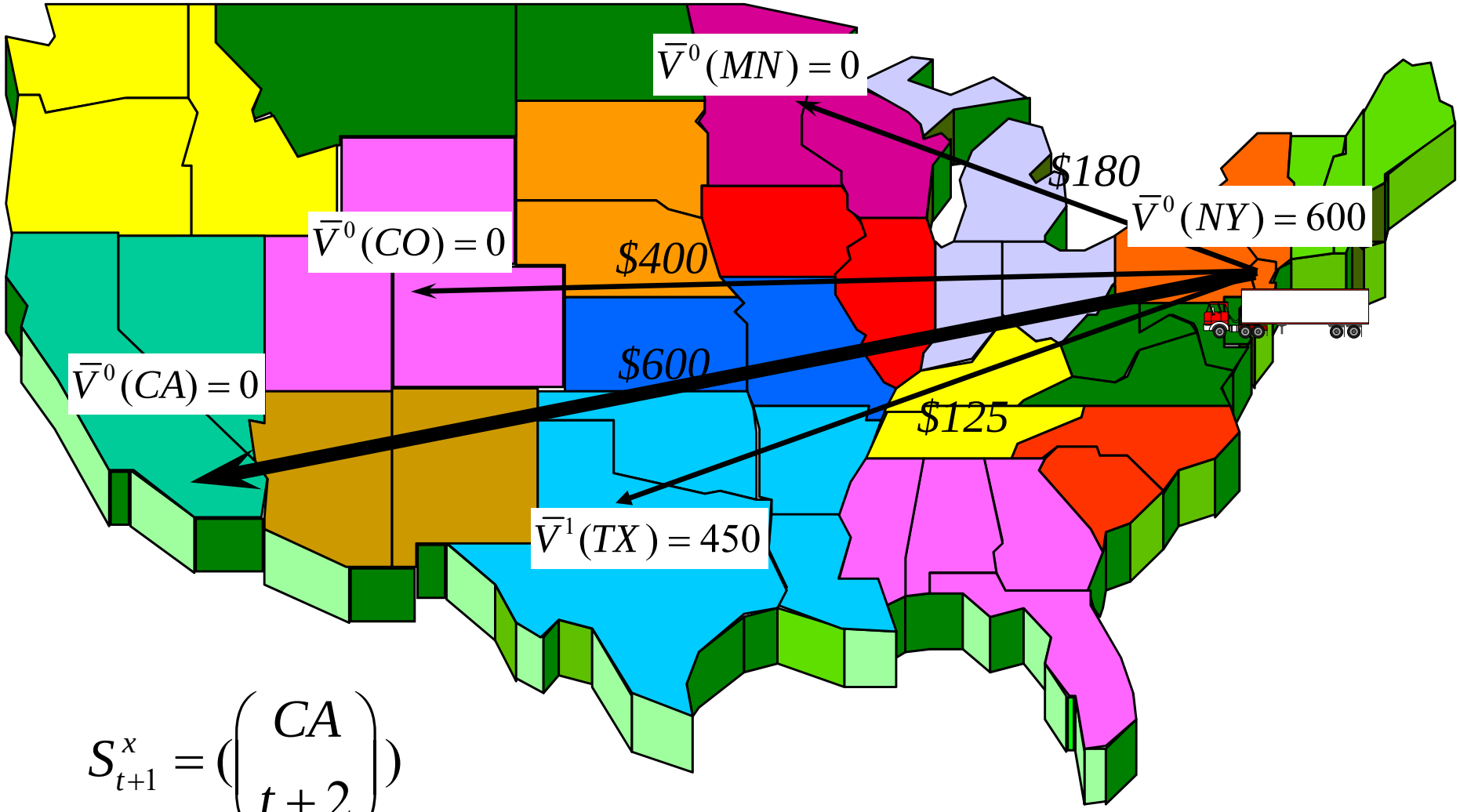
- Now move to the next state, sample new demands and make a new decision



$$S_{t+1} = \left(\begin{pmatrix} NY \\ t+1 \end{pmatrix}, \hat{D}_{t+1}\right)$$

Approximate dynamic programming

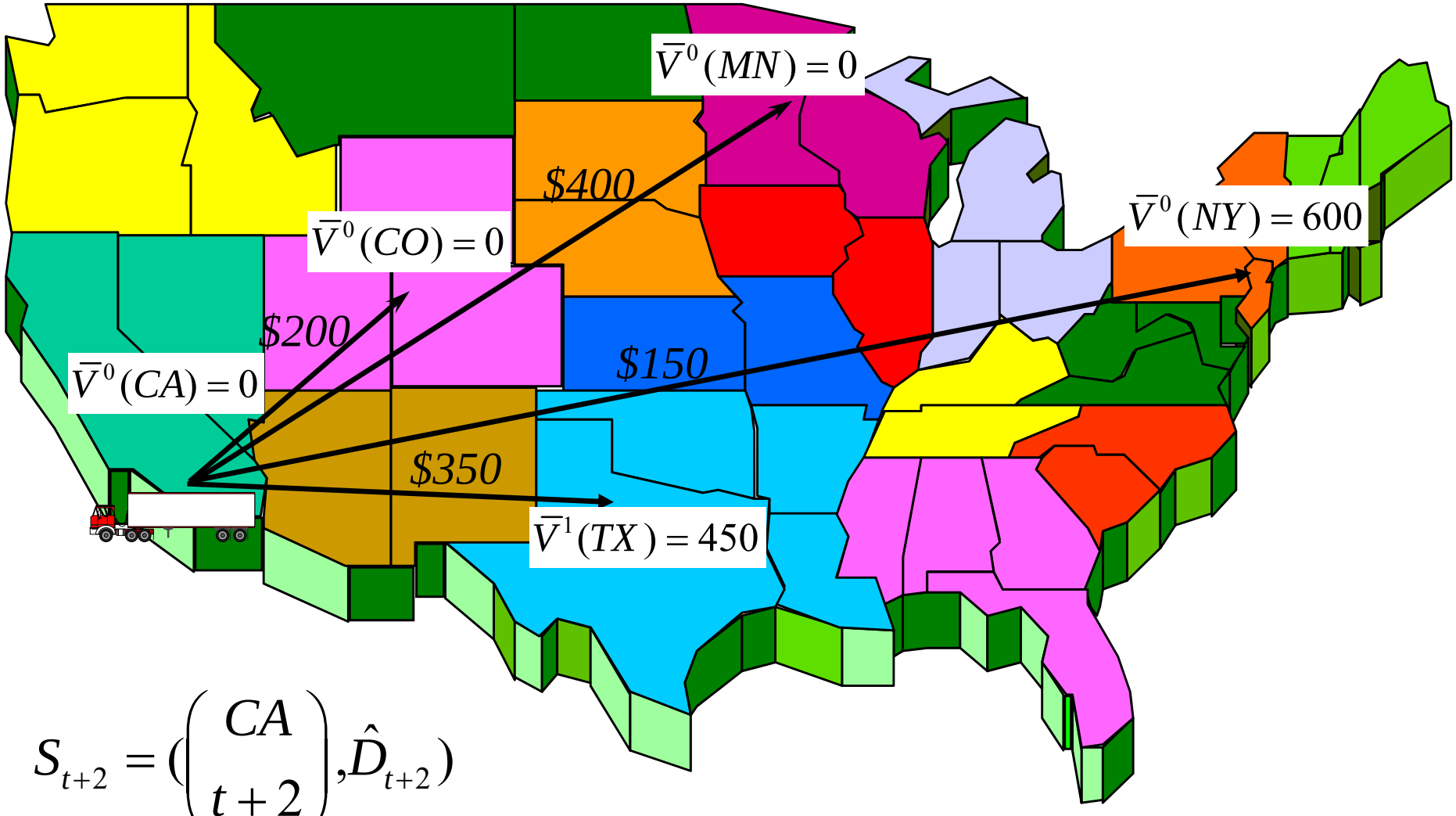
- Update value of being in NY



$$S_{t+1}^x = \begin{pmatrix} CA \\ t+2 \end{pmatrix}$$

Approximate dynamic programming

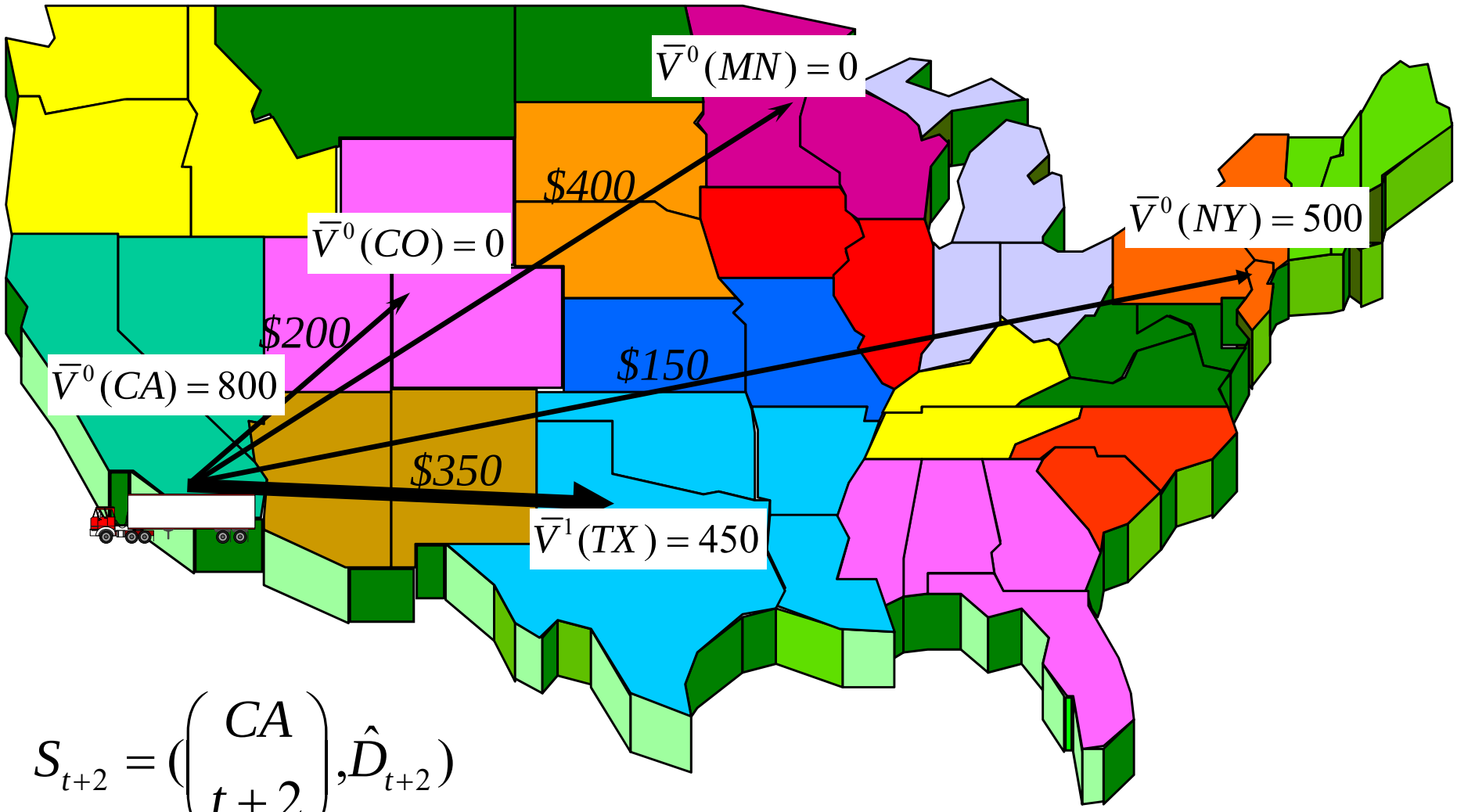
- Move to California.



$$S_{t+2} = \left(\begin{pmatrix} CA \\ t+2 \end{pmatrix}, \hat{D}_{t+2}\right)$$

Approximate dynamic programming

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



$$S_{t+2} = \left(\begin{pmatrix} CA \\ t+2 \end{pmatrix}, \hat{D}_{t+2}\right)$$

Approximate dynamic programming

- Updating the value function:

Old value:

$$\bar{V}^1(TX) = \$450$$

New estimate:

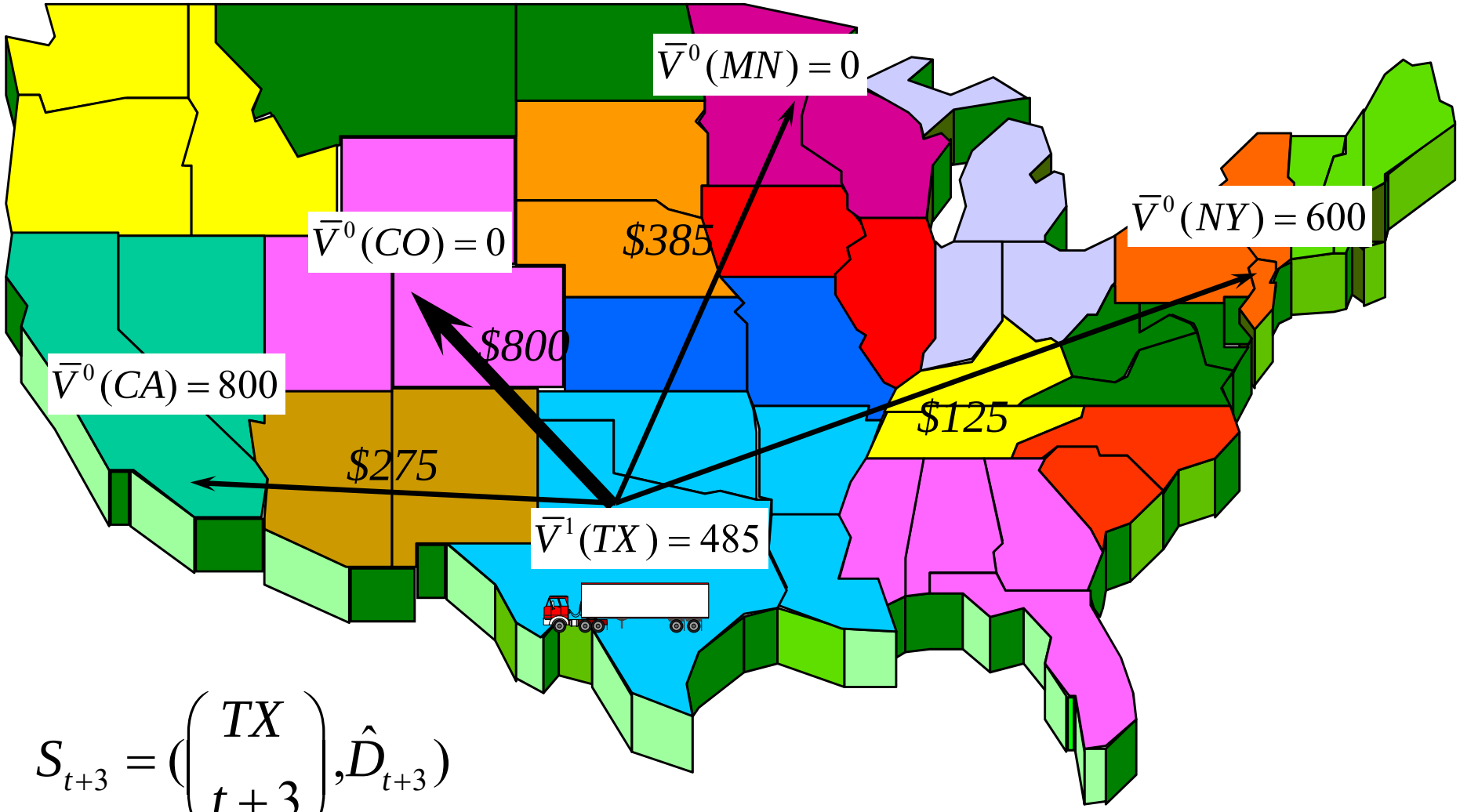
$$\hat{v}^2(TX) = \$800$$

How do we merge old with new?

$$\begin{aligned}\bar{V}^2(TX) &= (1-\alpha)\bar{V}^1(TX) + (\alpha)\hat{v}^2(TX) \\ &= (0.90)\$450 + (0.10)\$800 \\ &= \$485\end{aligned}$$

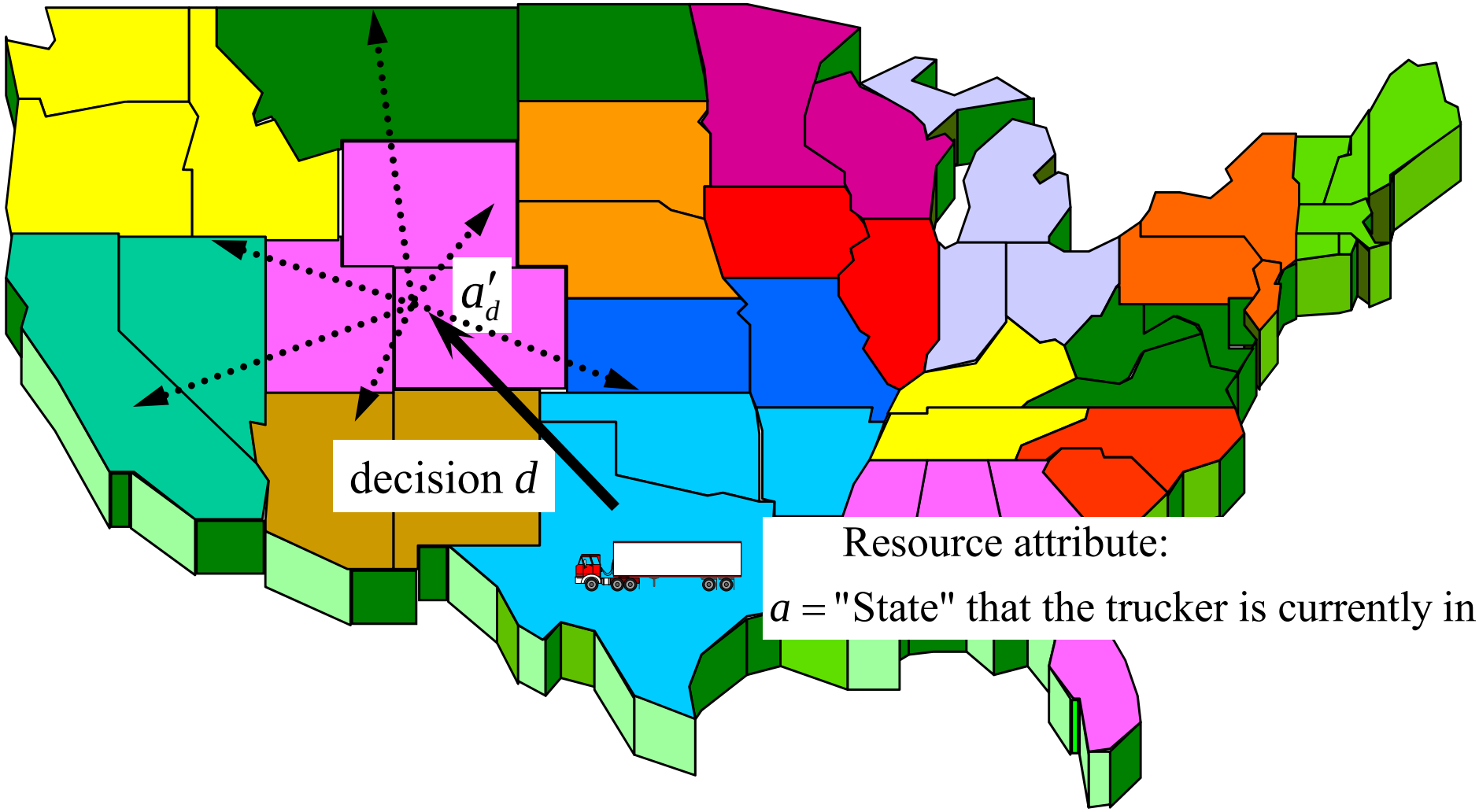
Approximate dynamic programming

- An updated value of being in TX



$$S_{t+3} = \left(\begin{pmatrix} TX \\ t+3 \end{pmatrix}, \hat{D}_{t+3}\right)$$

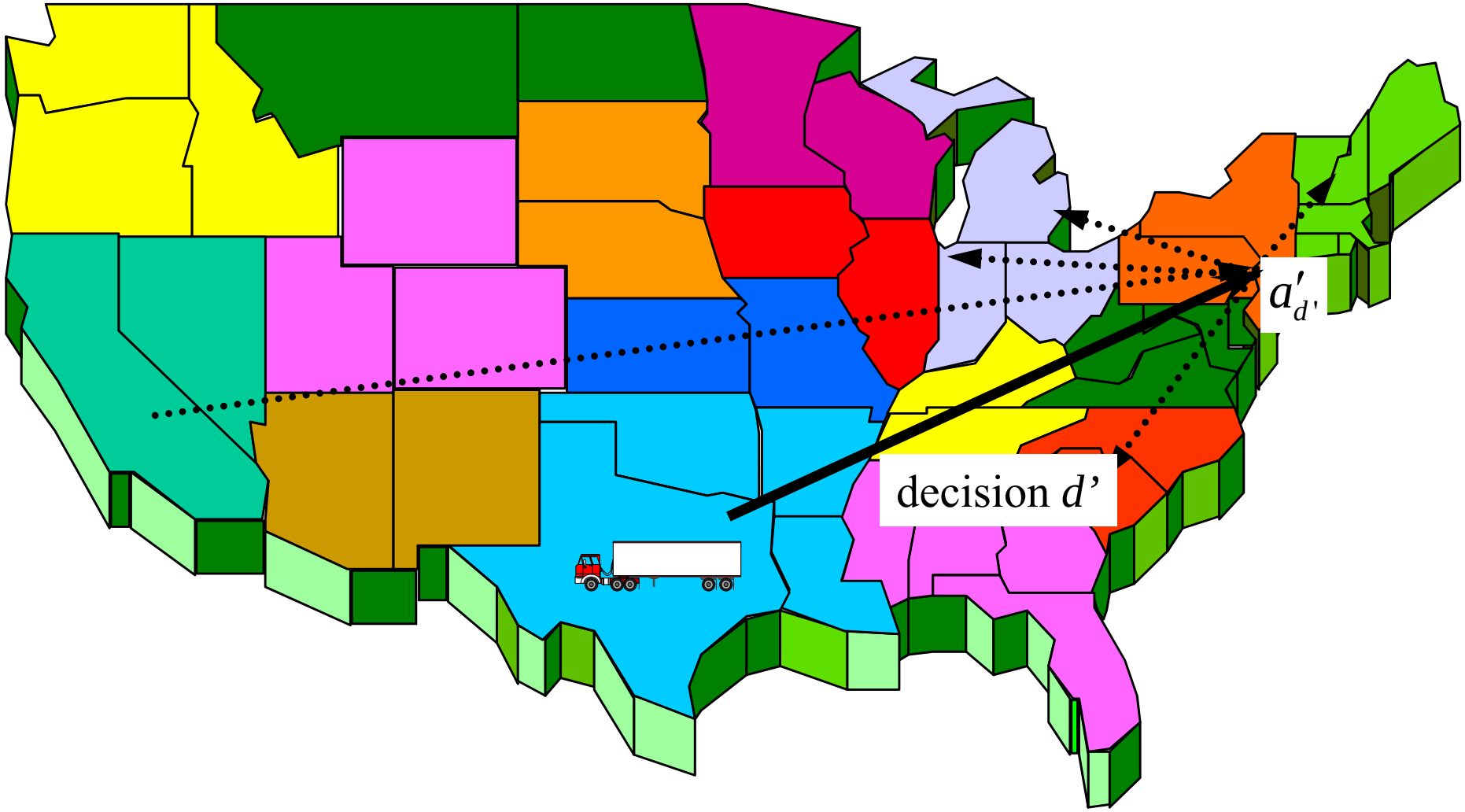
Approximate dynamic programming



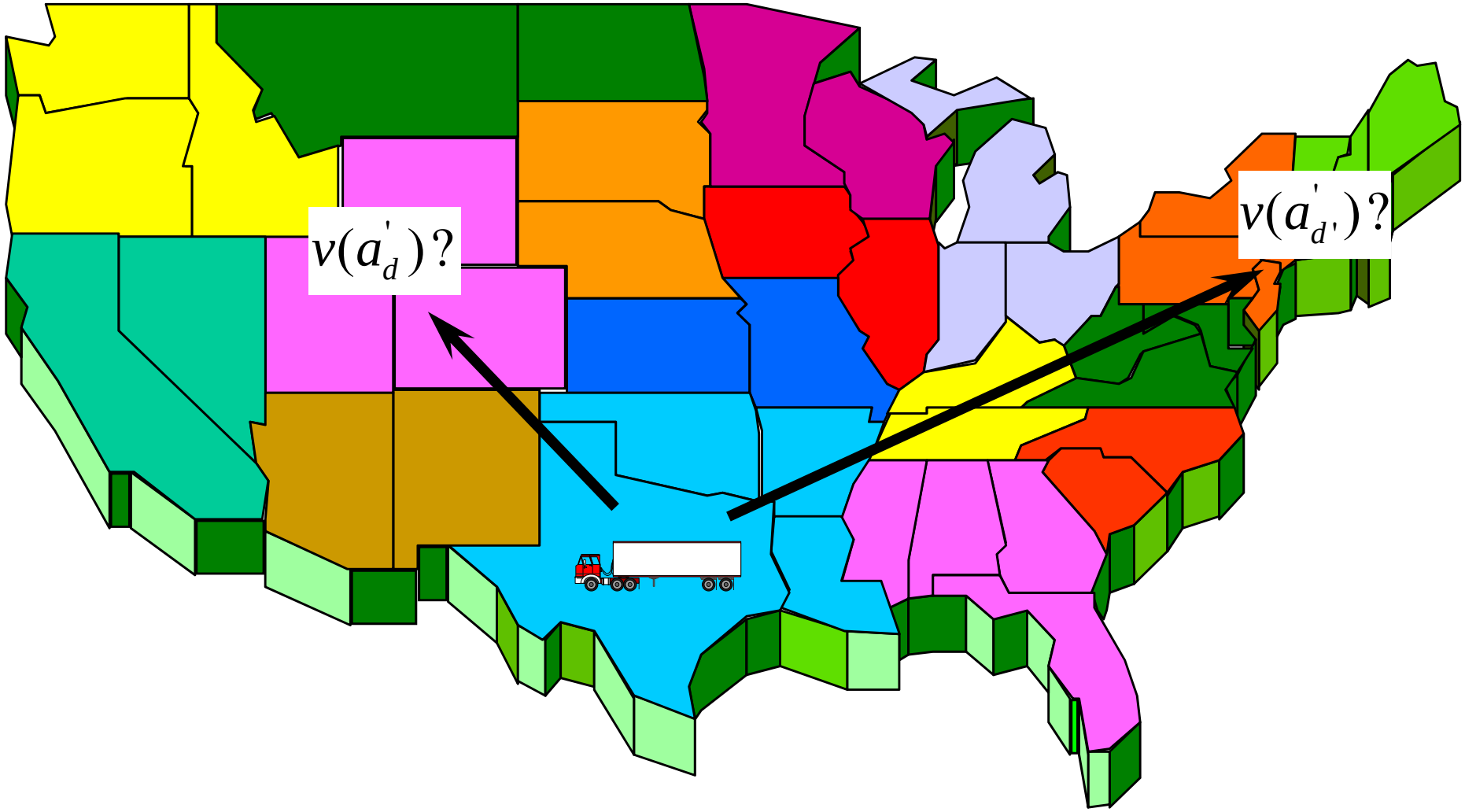
Resource attribute:

a = "State" that the trucker is currently in

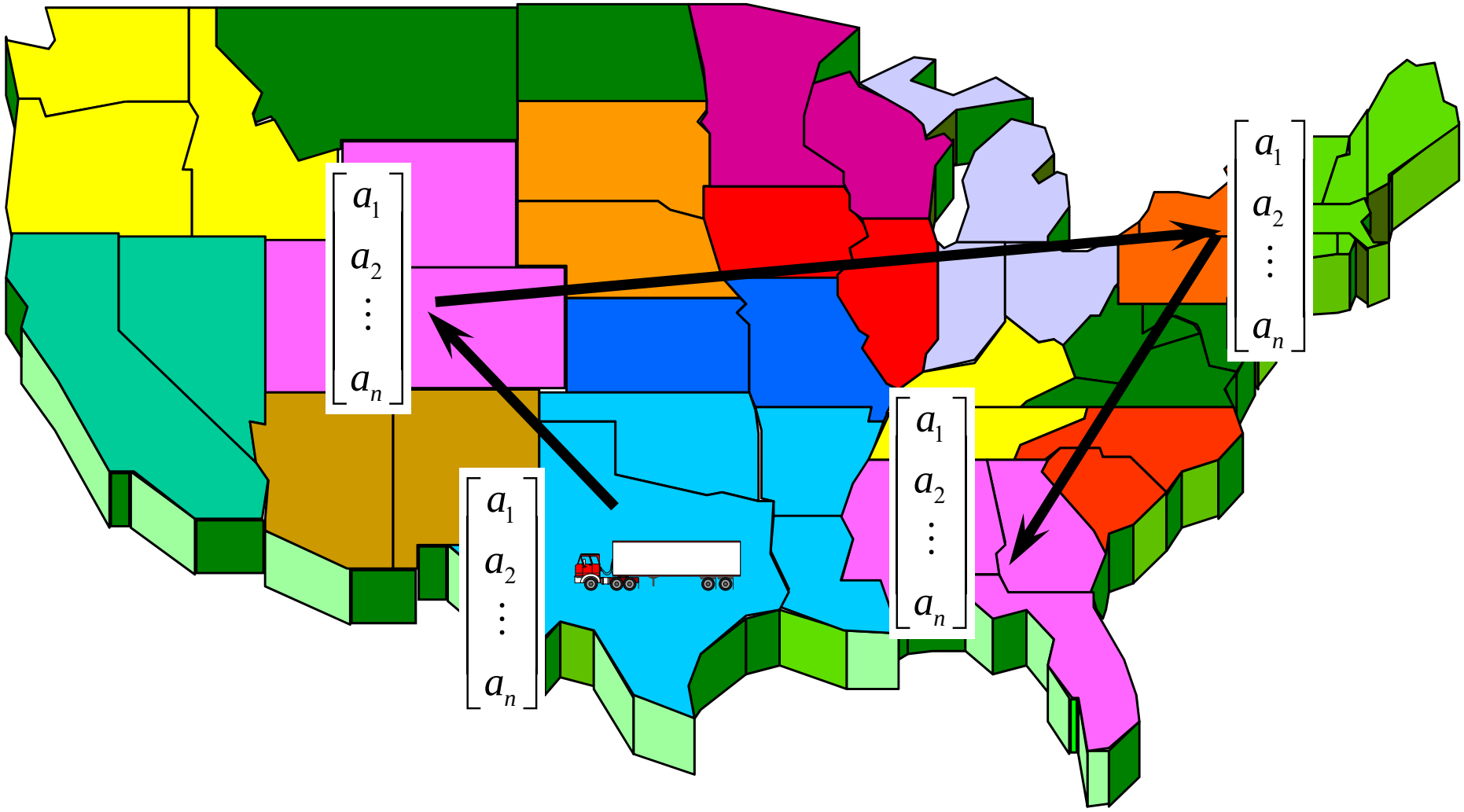
Approximate dynamic programming



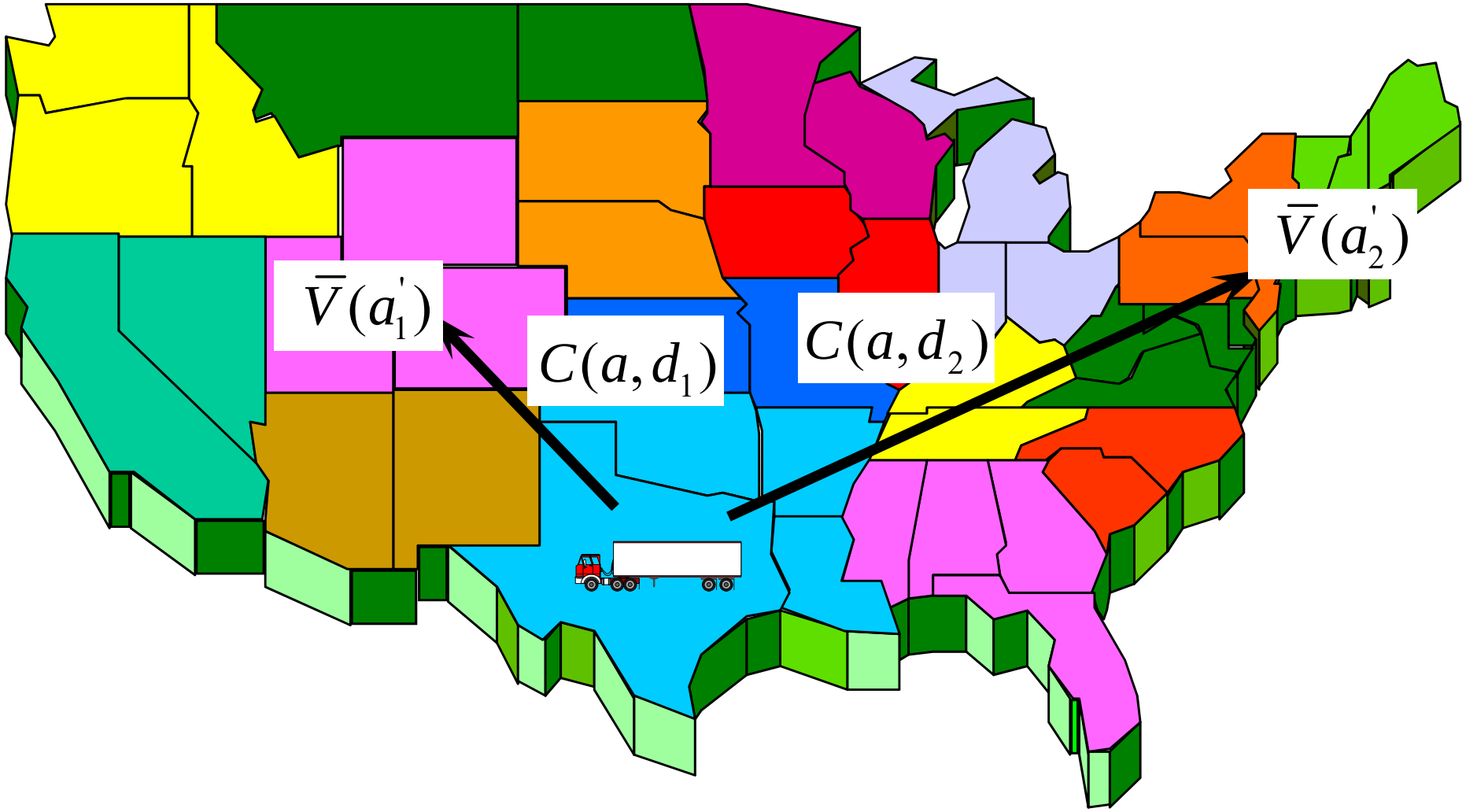
Approximate dynamic programming



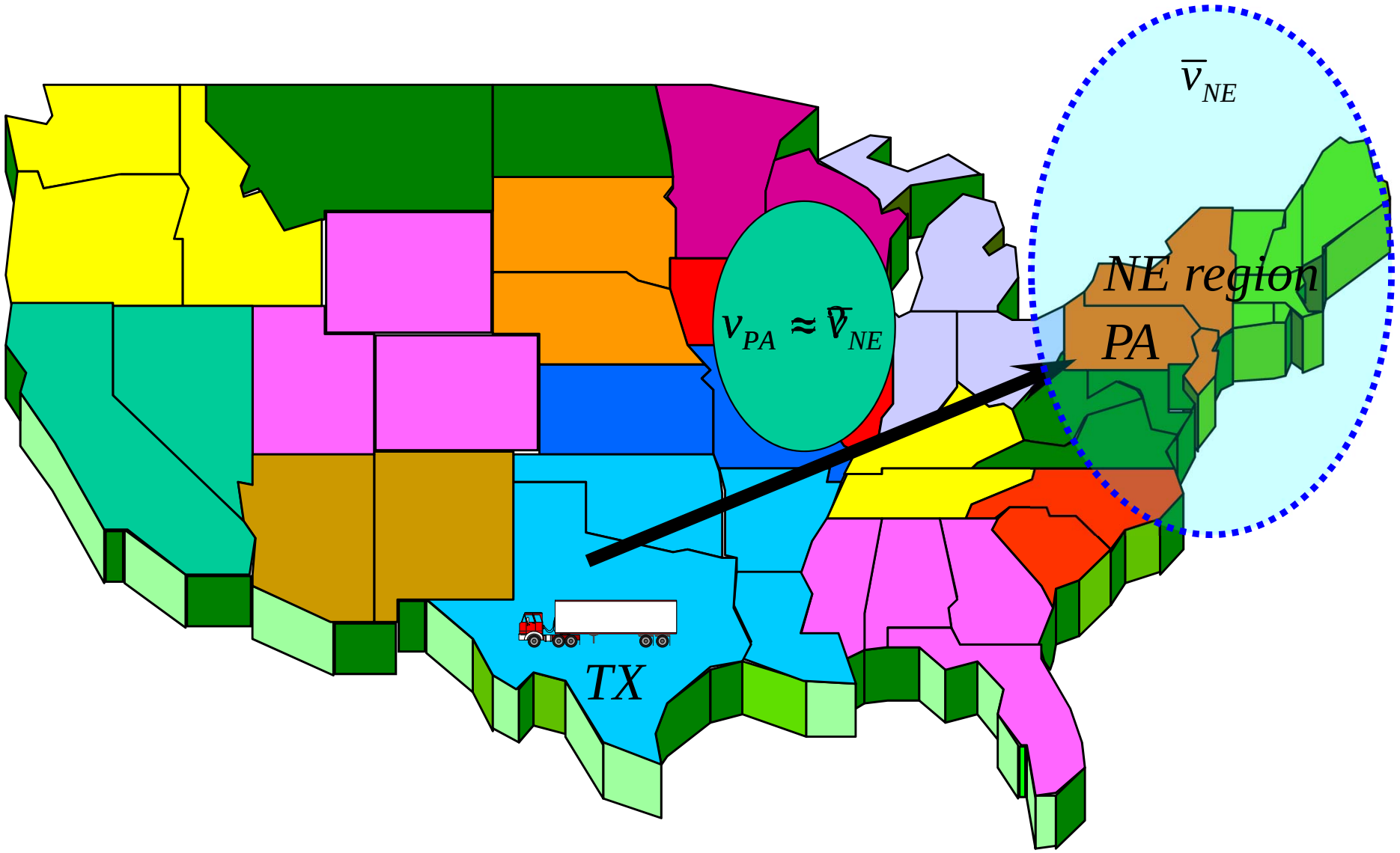
Approximate dynamic programming



Approximate dynamic programming



Approximate dynamic programming



Hierarchical aggregation

- Discuss:
  - » Curse of dimensionality for lookup tables
  - » Hierarchical aggregation:
    - Ignoring attributes
    - Aggregating attributes
- Examples:
  - » Drugs and drug classes
  - » Spatial
  - » Temporal

Aggregation level	Location	Fleet type	Domicile	Size of state space
0	Sub-region	Fleet	Region	$400 \times 5 \times 100 = 200,000$
1	Region	Fleet	Region	$100 \times 5 \times 100 = 50,000$
2	Region	Fleet	Zone	$100 \times 5 \times 10 = 5,000$
3	Region	Fleet	-	$100 \times 5 \times 1 = 500$
4	Zone	-	-	$10 \times 1 \times 1 = 10$

Hierarchical aggregation

$$\hat{v}\begin{pmatrix}\text{Loc}\\ \text{Equip}\\ \text{Home}\\ \text{DOThrs}\\ \text{Days}\end{pmatrix} = \max_a \big(C(s,a) + \overline{V}(S^M(s,a))\big).$$

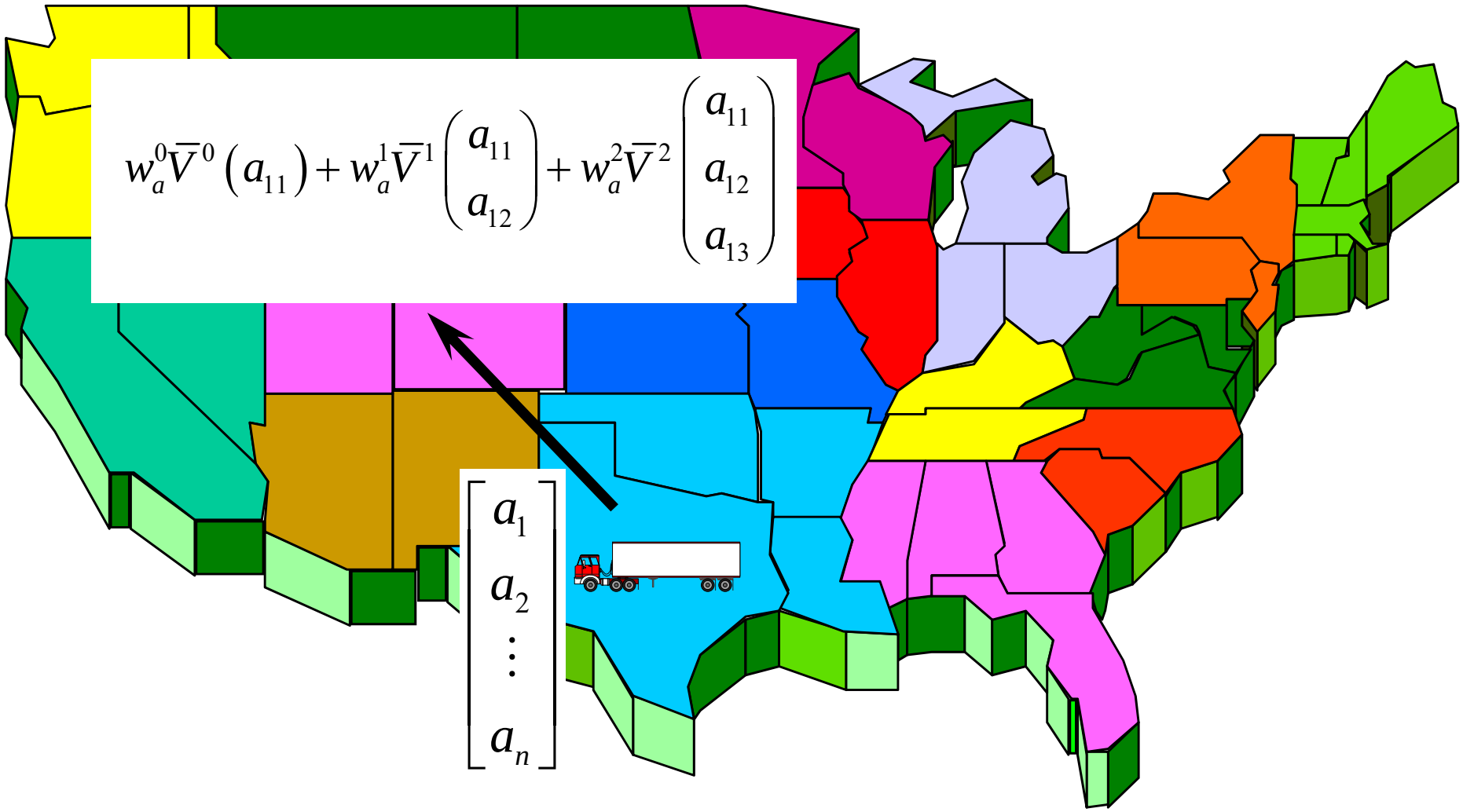
We now wish to use this estimate of the value of a driver with state s to produce value functions at different levels of aggregation. We can do this by simply smoothing this disaggregate estimate in with estimates at different levels of aggregation, as in

$$\bar{v}^{(1,n)}\begin{pmatrix}\text{Loc}\\ \text{Equip}\\ \text{Home}\end{pmatrix} = (1-\alpha^{(1)}_{s,n-1})\bar{v}^{(1,n-1)}\begin{pmatrix}\text{Loc}\\ \text{Equip}\\ \text{Home}\end{pmatrix} + \alpha^{(1)}_{s,n-1}\hat{v}\begin{pmatrix}\text{Loc}\\ \text{Equip}\\ \text{Home}\\ \text{DOThrs}\\ \text{Days}\end{pmatrix},$$

$$\bar{v}^{(2,n)}\begin{pmatrix}\text{Loc}\\ \text{Equip}\end{pmatrix} = (1-\alpha^{(2)}_{s,n-1})\bar{v}^{(2,n-1)}\begin{pmatrix}\text{Loc}\\ \text{Equip}\end{pmatrix} + \alpha^{(2)}_{s,n-1}\hat{v}\begin{pmatrix}\text{Loc}\\ \text{Equip}\\ \text{Home}\\ \text{DOThrs}\\ \text{Days}\end{pmatrix},$$

$$\bar{v}^{(3,n)}\begin{pmatrix}\text{Loc}\end{pmatrix} = (1-\alpha^{(3)}_{s,n-1})\bar{v}^{(3,n-1)}\begin{pmatrix}\text{Loc}\end{pmatrix} + \alpha^{(3)}_{s,n-1}\hat{v}\begin{pmatrix}\text{Loc}\\ \text{Equip}\\ \text{Home}\\ \text{DOThrs}\\ \text{Days}\end{pmatrix}.$$

Hierarchical aggregation



Hierarchical aggregation

3.3.4 Combining multiple levels of aggregation

Rather than try to pick the best level of aggregation, it is intuitively appealing to use a weighted sum of estimates at different levels of aggregation. The simplest strategy is to use

$$\bar{v}_s^n = \sum_{g \in \mathcal{G}} w^{(g)} \bar{v}_s^{(g)}, \tag{3.34}$$

where $w^{(g)}$ is the weight applied to the g^{th} level of aggregation. We would expect the weights to be positive and sum to one, but we can also view these simply as coefficients in a regression function. In such a setting, we would normally write the regression as

$$\overline{V}(s|\theta) = \theta_0 + \sum_{g \in \mathcal{G}} \theta_g \bar{v}_s^{(g)},$$

Hierarchical aggregation

- State dependent weight:

$$\bar{v}_s^n = \sum_{g \in \mathcal{G}} w_s^{(g)} \bar{v}_s^{(g,n)}.$$

$$w_s^{(g)} \propto \left((\bar{\sigma}_s^2)^{(g,n)} \right)^{-1}.$$

Since the weights should sum to one, we obtain

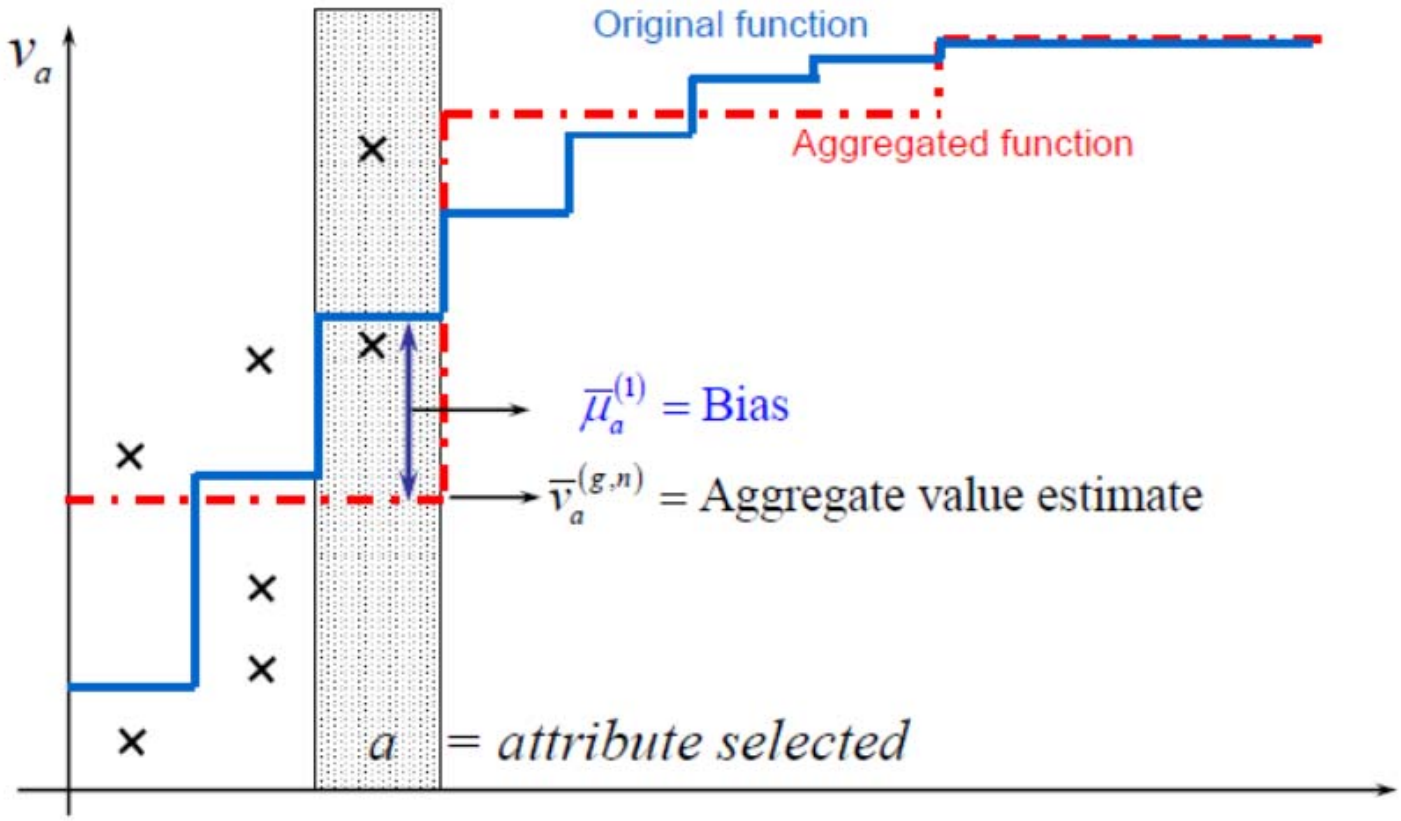
$$w_s^{(g)} = \left(\frac{1}{(\bar{\sigma}_s^2)^{(g,n)}} \right) \left(\sum_{g \in \mathcal{G}} \frac{1}{(\bar{\sigma}_s^2)^{(g,n)}} \right)^{-1}. \tag{3.35}$$

These weights work if the estimates are unbiased, which is clearly not the case. This is easily fixed by using the total variation (variance plus the square of the bias), producing the weights

$$w_s^{(g,n)} = \frac{1}{\left((\bar{\sigma}_s^2)^{(g,n)} + \left(\bar{\mu}_s^{(g,n)} \right)^2 \right)} \left(\sum_{g' \in \mathcal{G}} \frac{1}{\left((\bar{\sigma}_s^2)^{(g',n)} + \left(\bar{\mu}_s^{(g',n)} \right)^2 \right)} \right)^{-1}. \tag{3.36}$$

Hierarchical aggregation

- Computing bias and variance



Hierarchical aggregation

- Computing bias and variance:

$$\hat{\mu}^n = \mu(n) + \varepsilon^n,$$

where $\mathbb{E}\varepsilon^n = 0$ and $Var[\varepsilon^n] = \sigma^2$. Previously, ε^n was the error between our previous estimate and our latest observation. Here, we treat this as an exogenous measurement error. With this model, we can compute the variance of μ^n using

$$Var[\bar{\mu}^n] = \lambda^n \sigma^2, \tag{3.22}$$

where λ^n can be computed from the simple recursion

$$\lambda^n = \begin{cases} (\alpha_{n-1})^2, & n = 1, \\ (1-\alpha_{n-1})^2\lambda^{n-1} + (\alpha_{n-1})^2, & n > 1. \end{cases} \tag{3.23}$$

To see this, we start with $n = 1$. For a given (deterministic) initial estimate μ^0, we first observe that the variance of $\bar{\mu}^1$ is given by

$$\begin{aligned} Var[\bar{\mu}^1] &= Var[(1-\alpha_0)\bar{\mu}^0 + \alpha_0\hat{\mu}^1] \\ &= (\alpha_0)^2 Var[\hat{\mu}^1] \\ &= (\alpha_0)^2\sigma^2. \end{aligned}$$

For $\bar{\mu}^n$ for $n > 1$, we use a proof by induction. Assume that $Var[\bar{\mu}^{n-1}] = \lambda^{n-1}\sigma^2$. Then, since $\bar{\mu}^{n-1}$ and $\hat{\mu}^n$ are independent, we find

$$\begin{aligned} Var[\bar{\mu}^n] &= Var\left[(1-\alpha_{n-1})\bar{\mu}^{n-1} + \alpha_{n-1}\hat{\mu}^n\right] \\ &= (1-\alpha_{n-1})^2 Var\left[\bar{\mu}^{n-1}\right] + (\alpha_{n-1})^2 Var[\hat{\mu}^n] \\ &= (1-\alpha_{n-1})^2\lambda^{n-1}\sigma^2 + (\alpha_{n-1})^2\sigma^2 \qquad (3.24) \\ &= \lambda^n\sigma^2. \qquad (3.25) \end{aligned}$$

Equation (3.24) is true by assumption (in our induction proof), while equation (3.25) establishes the recursion in equation (3.23). This gives us the variance, assuming of course that σ^2 is known.

Hierarchical aggregation

situation of approximating value functions where the true value is evolving (for example, it may be increasing) over the iterations. To avoid confusion with our static notation, we are going to use $\bar{\mu}_x^n$ as our estimate of this time-varying function at x. We are then going to let $\hat{\mu}_x^n$ be a noisy observation of $mu(n)_x$.

We will use the following recursive formula for $\bar{\mu}^n$ (for any point x)

$$\bar{\mu}^n = (1 - \alpha_{n-1})\bar{\mu}^{n-1} + \alpha_{n-1}\hat{\mu}^n,$$

where $\hat{\mu}^n = \mu(n) + \epsilon^n$ is an unbiased observation (that is, $\mathbb{E}\hat{\mu}^n = \mu(n)$) which is assumed to be independent of μ^{n-1}. We note that when we are trying to estimate a static function such as $\mathbb{E}F(x, W)$, the parameter we are trying to estimate, $\mu_x(n) = \mu_x = \mathbb{E}F(x, W)$ is static. However, when we are trying to estimate a value function as we illustrated in equation (3.2), then we are trying to estimate a parameter $\bar{\mu}_x^n$ that actually varies with n. We will see this behavior, first in chapter 14 when we introduce basic dynamic programs, and then in chapters 17 and 18 when we cover approximate dynamic programming.

We are interested in estimating the variance of μ^n and its bias, which is defined by $\mu^{n-1} - \mu^n$. We start by computing the variance of μ^n. We assume that our observations of μ^n can be represented using

$$\hat{\mu}^n = \mu(n) + \varepsilon^n,$$

where $\mathbb{E}\varepsilon^n = 0$ and $Var[\varepsilon^n] = \sigma^2$. Previously, ε^n was the error between our previous estimate and our latest observation. Here, we treat this as an exogenous measurement error. With this model, we can compute the variance of μ^n using

$$Var[\bar{\mu}^n] = \lambda^n\sigma^2, \tag{3.22}$$

Hierarchical aggregation

$$Var[\bar{\mu}^n] = \lambda^n \sigma^2, \tag{3.22}$$

where λ^n can be computed from the simple recursion

$$\lambda^n = \begin{cases} (\alpha_{n-1})^2, & n = 1, \\ (1-\alpha_{n-1})^2 \lambda^{n-1} + (\alpha_{n-1})^2, & n > 1. \end{cases} \tag{3.23}$$

To see this, we start with $n = 1$. For a given (deterministic) initial estimate μ^0, we first observe that the variance of $\bar{\mu}^1$ is given by

$$\begin{aligned} Var[\bar{\mu}^1] &= Var[(1-\alpha_0)\bar{\mu}^0 + \alpha_0 \hat{\mu}^1] \\ &= (\alpha_0)^2 Var[\hat{\mu}^1] \\ &= (\alpha_0)^2 \sigma^2. \end{aligned}$$

For $\bar{\mu}^n$ for $n > 1$, we use a proof by induction. Assume that $Var[\bar{\mu}^{n-1}] = \lambda^{n-1}\sigma^2$. Then, since $\bar{\mu}^{n-1}$ and $\hat{\mu}^n$ are independent, we find

$$\begin{aligned} Var[\bar{\mu}^n] &= Var\left[(1-\alpha_{n-1})\bar{\mu}^{n-1} + \alpha_{n-1}\hat{\mu}^n\right] \\ &= (1-\alpha_{n-1})^2 Var\left[\bar{\mu}^{n-1}\right] + (\alpha_{n-1})^2 Var[\hat{\mu}^n] \\ &= (1-\alpha_{n-1})^2 \lambda^{n-1}\sigma^2 + (\alpha_{n-1})^2 \sigma^2 \qquad (3.24) \\ &= \lambda^n \sigma^2. \qquad (3.25) \end{aligned}$$

Equation (3.24) is true by assumption (in our induction proof), while equation (3.25) establishes the recursion in equation (3.23). This gives us the variance, assuming of course that σ^2 is known.

Hierarchical aggregation

The bias of our estimate is the difference between our current estimate and the true value, given by

$$\beta^n = \mathbb{E}[\bar{\mu}^{n-1}] - \mu^n.$$

We note that $\bar{\mu}^{n-1}$ is our estimate of μ^n computed using the information up through iteration $n-1$. Of course, our formula for the bias assumes that $\mu(n)$ is known. These two results for the variance and bias are called the *parameters-known* formulas.

We next require the mean-squared error, which can be computed using

$$\mathbb{E}\left[\left(\bar{\mu}^{n-1} - \mu(n)\right)^2\right] = \lambda^{n-1}\sigma^2 + (\beta^n)^2. \tag{3.26}$$

See exercise 3.3 to prove this. This formula gives the variance around the known mean, μ^n. For our purposes, it is also useful to have the variance around the observations $\hat{\mu}^n$. Let

$$\nu^n = \mathbb{E}\left[\left(\bar{\mu}^{n-1} - \hat{\mu}^n\right)^2\right]$$

be the mean squared error (including noise and bias) between the current estimate $\bar{\mu}^{n-1}$ and the observation $\hat{\mu}^n$. It is possible to show that (see exercise 3.4)

$$\nu^n = (1 + \lambda^{n-1})\sigma^2 + (\beta^n)^2, \tag{3.27}$$

where λ^n is computed using (3.23).

Hierarchical aggregation

In practice, we do not know σ^2, and we certainly do not know $\mu(n)$. As a result, we have to estimate both parameters from our data. We begin by providing an estimate of the bias using

$$\bar{\beta}^n = (1 - \eta_{n-1})\bar{\beta}^{n-1} + \eta_{n-1}(\bar{\mu}^{n-1} - \hat{\mu}^n),$$

where η_{n-1} is a (typically simple) stepsize rule used for estimating the bias and variance. As a general rule, we should pick a stepsize for η_{n-1} which produces larger stepsizes than α_{n-1} because we are more interested in tracking the true signal than producing an estimate with a low variance. We have found that a constant stepsize such as .10 works quite well on a wide range of problems, but if precise convergence is needed, it is necessary to use a rule where the stepsize goes to zero such as the harmonic stepsize rule (equation (6.12)).

To estimate the variance, we begin by finding an estimate of the total variation ν^n. Let $\bar{\nu}^n$ be the estimate of the total variance which we might compute using

$$\bar{\nu}^n = (1 - \eta_{n-1})\bar{\nu}^{n-1} + \eta_{n-1}(\bar{\mu}^{n-1} - \hat{\mu}^n)^2.$$

Using $\bar{\nu}^n$ as our estimate of the total variance, we can compute an estimate of σ^2 using

$$(\bar{\sigma}^n)^2 = \frac{\bar{\nu}^n - (\bar{\beta}^n)^2}{1 + \lambda^{n-1}}.$$

We can use $(\bar{\sigma}^n)^2$ in equation (3.22) to obtain an estimate of the variance of μ^n.

If we are doing true averaging (as would occur if we use a stepsize of $1/n$), we can get a more precise estimate of the variance for small samples by using the recursive form of the small sample formula for the variance

$$(\hat{\sigma}^2)^n = \frac{n-2}{n-1}(\hat{\sigma}^2)^{n-1} + \frac{1}{n}(\bar{\mu}^{n-1} - \hat{\mu}^n)^2. \tag{3.28}$$

The quantity $(\hat{\sigma}^2)^n$ is an estimate of the variance of $\hat{\mu}^n$. The variance of our estimate $\mu^{(n)}$ is computed using

$$(\bar{\sigma}^2)^n = \frac{1}{n}(\hat{\sigma}^2)^n.$$

Hierarchical aggregation

$$\hat{v}\begin{pmatrix}\text{Loc}\\ \text{Equip}\\ \text{Home}\\ \text{DOThrs}\\ \text{Days}\end{pmatrix} = \max_a \left(C(s,a) + \overline{V}(S^M(s,a))\right).$$

We now wish to use this estimate of the value of a driver with state s to produce value functions at different levels of aggregation. We can do this by simply smoothing this disaggregate estimate in with estimates at different levels of aggregation, as in

$$\bar{v}^{(1,n)}\begin{pmatrix}\text{Loc}\\ \text{Equip}\\ \text{Home}\end{pmatrix} = (1-\alpha^{(1)}_{s,n-1})\bar{v}^{(1,n-1)}\begin{pmatrix}\text{Loc}\\ \text{Equip}\\ \text{Home}\end{pmatrix} + \alpha^{(1)}_{s,n-1}\hat{v}\begin{pmatrix}\text{Loc}\\ \text{Equip}\\ \text{Home}\\ \text{DOThrs}\\ \text{Days}\end{pmatrix},$$

$$\bar{v}^{(2,n)}\begin{pmatrix}\text{Loc}\\ \text{Equip}\end{pmatrix} = (1-\alpha^{(2)}_{s,n-1})\bar{v}^{(2,n-1)}\begin{pmatrix}\text{Loc}\\ \text{Equip}\end{pmatrix} + \alpha^{(2)}_{s,n-1}\hat{v}\begin{pmatrix}\text{Loc}\\ \text{Equip}\\ \text{Home}\\ \text{DOThrs}\\ \text{Days}\end{pmatrix},$$

$$\bar{v}^{(3,n)}\begin{pmatrix}\text{Loc}\end{pmatrix} = (1-\alpha^{(3)}_{s,n-1})\bar{v}^{(3,n-1)}\begin{pmatrix}\text{Loc}\end{pmatrix} + \alpha^{(3)}_{s,n-1}\hat{v}\begin{pmatrix}\text{Loc}\\ \text{Equip}\\ \text{Home}\\ \text{DOThrs}\\ \text{Days}\end{pmatrix}.$$

Hierarchical aggregation

We can estimate the variance of $\bar{v}_s^{(g,n)}$ using the techniques described in section 3.3.2. Let

$$(s_s^2)^{(g,n)} = \text{The estimate of the variance of observations made of state } s, \text{ using data from aggregation level } g, \text{ after } n \text{ observations.}$$

$(s_s^2)^{(g,n)}$ is the estimate of the variance of the observations $\hat{v}$ when we observe a state $\hat{s}^n$ which aggregates to state s (that is, $G^g(\hat{s}^n) = s$). We are really interested in the variance of our estimate of the mean, $\bar{v}_s^{(g,n)}$. In section 3.3.2, we showed that

$$\begin{aligned}(\bar{\sigma}_s^2)^{(g,n)} &= Var[\bar{v}_s^{(g,n)}] \\ &= \lambda_s^{(g,n)}(s_s^2)^{(g,n)}, \end{aligned} \tag{3.30}$$

where $(s_s^2)^{(g,n)}$ is an estimate of the variance of the observations $\hat{v}_t^n$ at the g^{th} level of aggregation (computed below), and $\lambda_s^{(g,n)}$ can be computed from the recursion

$$\lambda_s^{(g,n)} = \begin{cases} (\alpha_{s,n-1}^{(g)})^2, & n = 1, \\ (1-\alpha_{s,n-1}^{(g)})^2\lambda_s^{(g,n-1)} + (\alpha_{s,n-1}^{(g)})^2, & n > 1. \end{cases}$$

Note that if the stepsize $\alpha_{s,n-1}^{(g)}$ goes to zero, then $\lambda_s^{(g,n)}$ will also go to zero, as will $(\bar{\sigma}_s^2)^{(g,n)}$. We now need to compute $(s_s^2)^{(g,n)}$ which is the estimate of the variance of observations $\hat{v}^n$ for states $\hat{s}^n$ for which $G^g(\hat{s}^n) = s$ (the observations of states that aggregate up to s). Let $\bar{\nu}_s^{(g,n)}$ be the total variation, given by

$$\bar{\nu}_s^{(g,n)} = (1-\eta_{n-1})\bar{\nu}_s^{(g,n-1)} + \eta_{n-1}(\bar{v}_s^{(g,n-1)} - \hat{v}_s^n)^2,$$

where η_{n-1} follows some stepsize rule (which may be just a constant). We refer to $\bar{\nu}_s^{(g,n)}$ as

Hierarchical aggregation

where η_{n-1} follows some stepsize rule (which may be just a constant). We refer to $\bar{\nu}_s^{(g,n)}$ as the total variation because it captures deviations that arise both due to measurement noise (the randomness when we compute $\hat{v}_s^n$) and bias (since $\bar{v}_s^{(g,n-1)}$ is a biased estimate of the mean of $\hat{v}_s^n$).

We finally need an estimate of the bias. There are two types of bias. In section 3.3.2 we measured the transient bias that arose from the use of smoothing applied to a nonstationary time series using

$$\bar{\beta}_s^{(g,n)} = (1-\eta_{n-1})\bar{\beta}_s^{(g,n-1)} + \eta_{n-1}(\hat{v}^n - \bar{v}_s^{(g,n-1)}). \tag{3.31}$$

This bias arises because the observations $\hat{v}^n$ may be steadily increasing (or decreasing) with the iterations. When we smooth on past observations, we obtain an estimate $\bar{v}_s^{(g,n-1)}$ that tends to underestimate (overestimate if $\hat{v}^n$ tends to decrease) the true mean of $\hat{v}^n$.

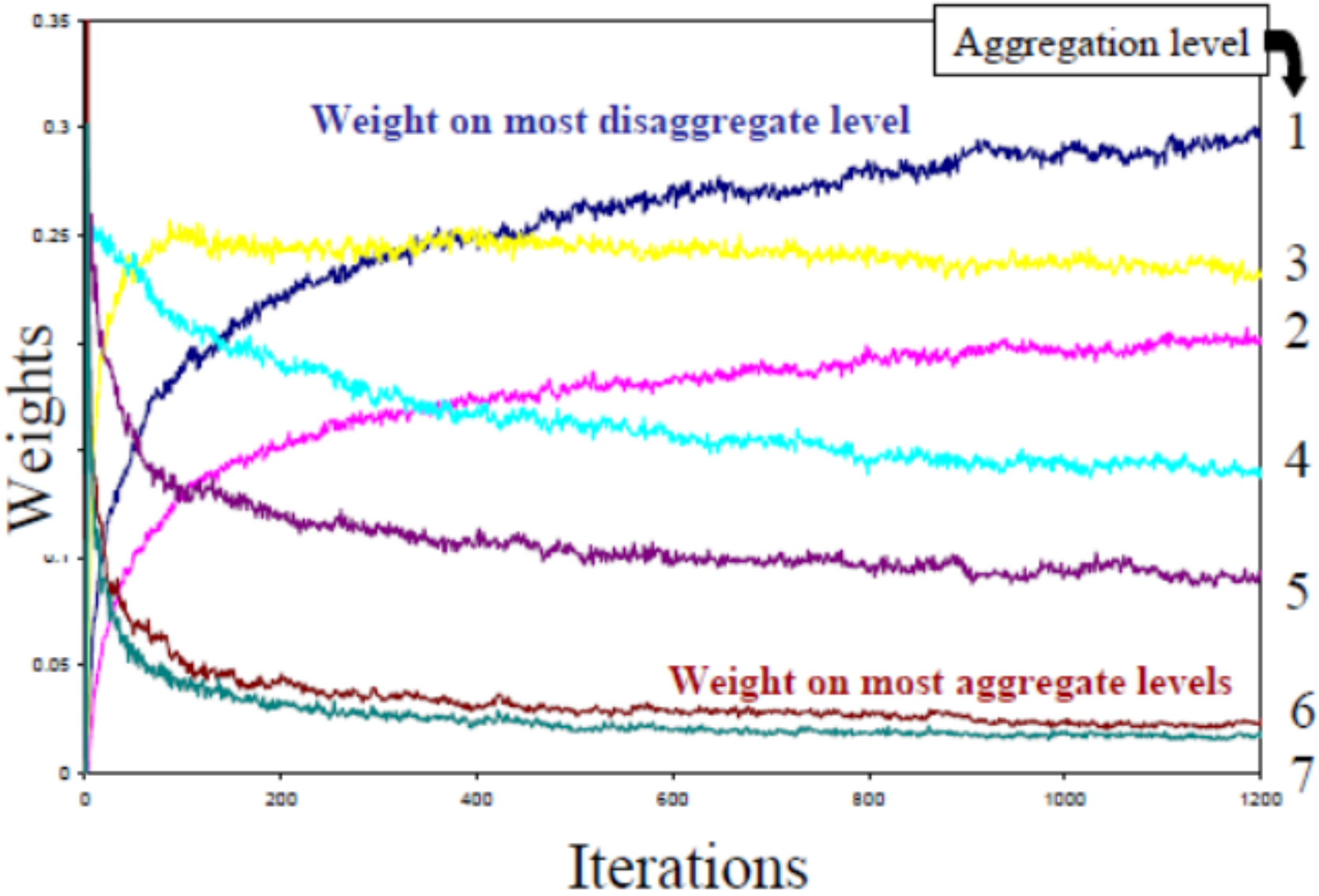
The second form of bias is the aggregation bias given by the difference between the estimate at an aggregate level and the disaggregate level. We compute the aggregation bias using

$$\bar{\mu}_s^{(g,n)} = \bar{v}_s^{(g,n)} - \bar{v}_s^{(0,n)}. \tag{3.32}$$

Using the same reasoning presented in section 3.3.2, we can separate out the effect of bias to obtain an estimate of the variance of the error using

$$(s_s^2)^{(g,n)} = \frac{\bar{\nu}_s^{(g,n)} - (\bar{\beta}_s^{(g,n)})^2}{1+\lambda^{n-1}}. \tag{3.33}$$

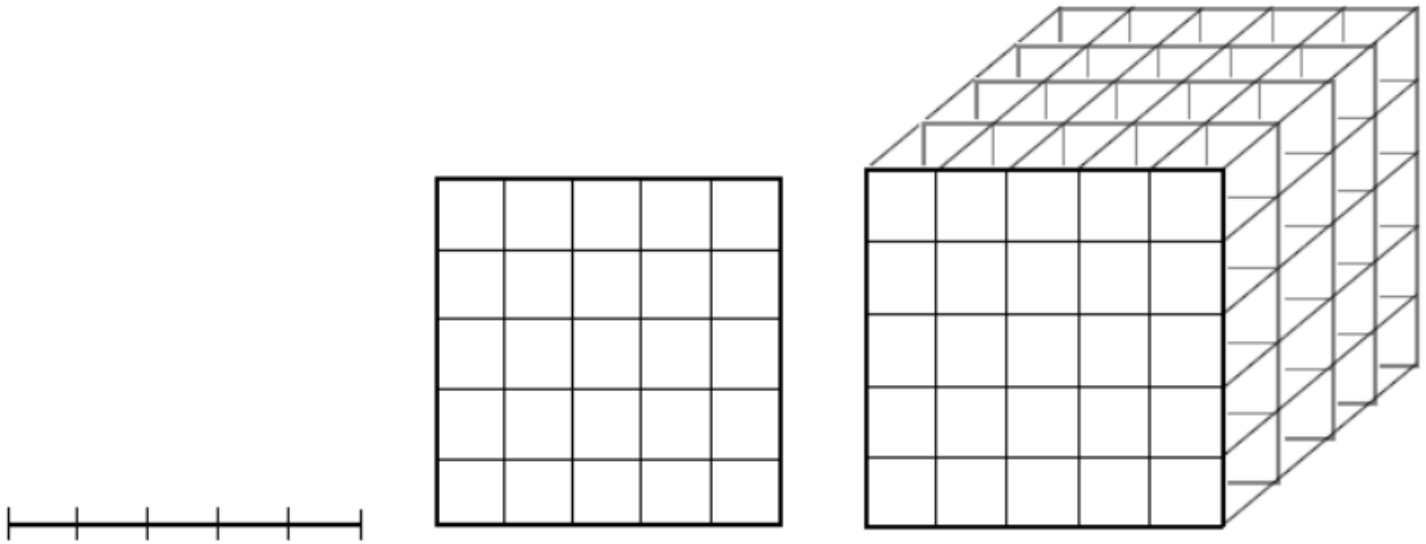
Hierarchical aggregation



The curse of dimensionality

The curse of dimensionality

- The curse of dimensionality
  - » This is often confused with any vector-valued learning problem (esp. in the context of dynamic programming).
  - » It *only* arises when using lookup table representations of functions:



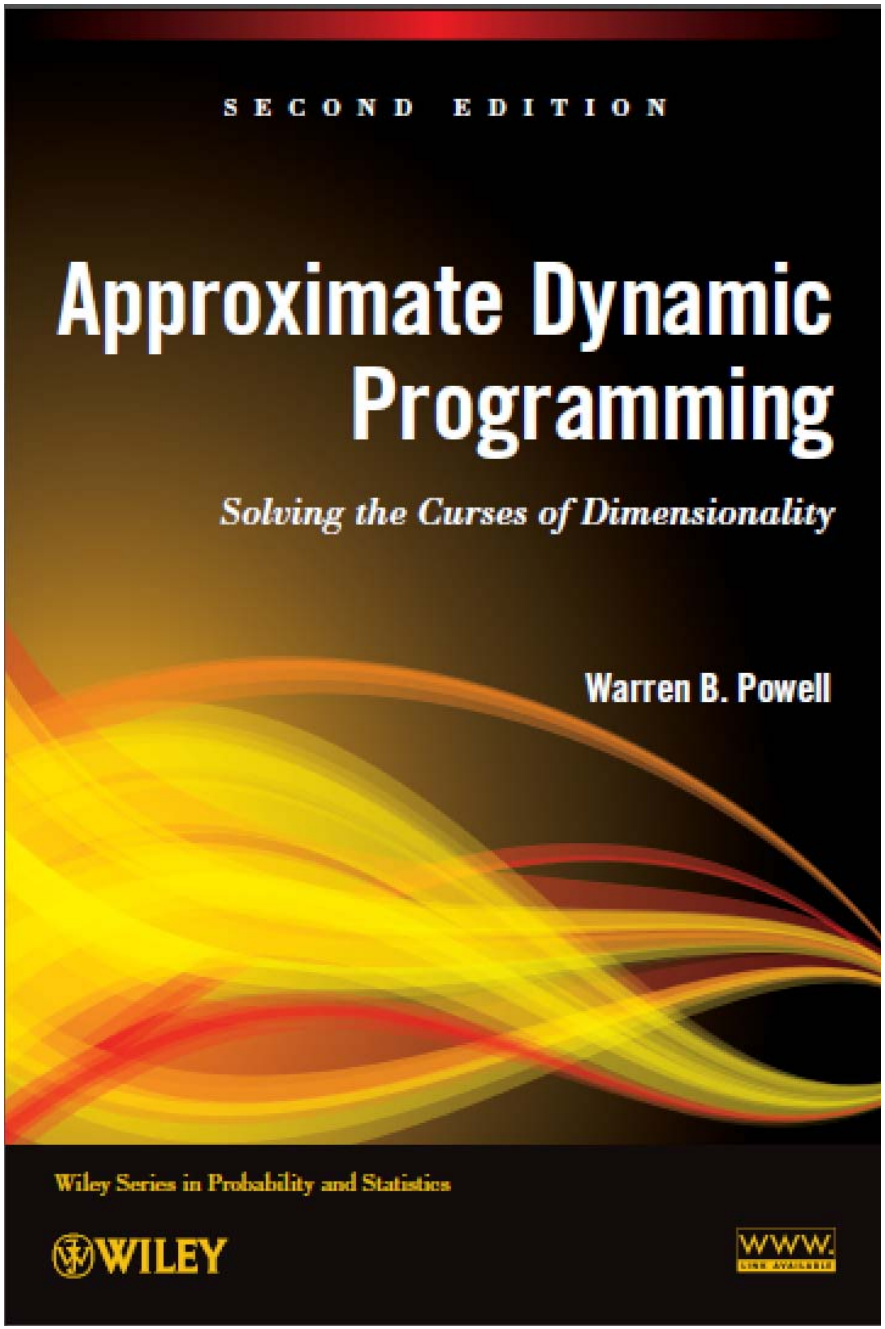
The curse of dimensionality

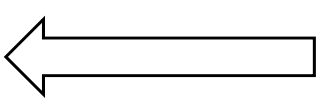
- "Solving" the curse of dimensionality is like inventing a perpetual motion machine

  » The curse of dimensionality is a true curse. It cannot be "solved" without structure
    - Convexity, linearity, monotonicity, …

  » Claims to approximate functions without structure should be treated like perpetual motion machines.

The curse of dimensionality





Parametric models

Linear models

Linear models

- Examples:

$$y = \theta_0 + \sum_{i=1}^{I} \theta_i x_i + \varepsilon. \tag{3.38}$$

The variables x_i might be called independent variables, explanatory variables, or covariates, depending on the community. In dynamic programming where we want to estimate a value function $V^\pi(S_t)$, we might write

$$\overline{V}(S|\theta) = \sum_{f \in \mathcal{F}} \theta_f \phi_f(S),$$

In fact, if we write our policy of the form

$$X^\pi(S_t|\theta) = \sum_{f \in \mathcal{F}} \theta_f \phi_f(S_t),$$

we would refer to $X^\pi(S_t|\theta)$ as an *affine policy* ("affine" is just a fancy name for linear, by which we mean linear in θ).

» Independent variables, covariates, basis functions

Linear models

3.4.1 Linear regression review

Let y^n be the n^{th} observation of our dependent variable (what we are trying to predict) based on the observation $(x_1^n, x_2^n, \ldots, x_I^n)$ of our independent (or explanatory) variables (the x_i are equivalent to the basis functions we used earlier). Our goal is to estimate a parameter vector θ that solves

$$\min_{\theta} \sum_{m=1}^{n} \left(y^m - (\theta_0 + \sum_{i=1}^{I} \theta_i x_i^m) \right)^2 . \tag{3.39}$$

This is the standard linear regression problem. Let μ^n be the optimal solution for this problem. Throughout this section, we assume that the underlying process from which the observations y^n are drawn is stationary (an assumption that is almost never satisfied in approximate dynamic programming).

If we define $x_0 = 1$, we let

$$x^n = \begin{pmatrix} x_0^n \\ x_1^n \\ \vdots \\ x_I^n \end{pmatrix}$$

Linear models

Letting θ be the column vector of parameters, we can write our model as

$$y = \theta^T x + \varepsilon.$$

We assume that the errors $(\varepsilon^1, \ldots, \varepsilon^n)$ are independent and identically distributed. We do not know the parameter vector θ, so we replace it with an estimate $\bar{\theta}$ which gives us the predictive formula

$$\bar{y}^n = (\bar{\theta})^T x^n,$$

where $\bar{y}^n$ is our predictor of y^{n+1}. Our prediction error is

$$\hat{\varepsilon}^n = y^n - (\bar{\theta})^T x^n.$$

Our goal is to choose θ to minimize the mean squared error

$$\min_{\theta} \sum_{m=1}^{n} (y^m - \theta^T x^m)^2. \tag{3.40}$$

It is well known that this can be solved very simply. Let X^n be the n by $I+1$ matrix

$$X^n = \begin{pmatrix} x_0^1 & x_1^1 & & x_I^1 \\ x_0^2 & x_1^2 & & x_I^2 \\ \vdots & \vdots & \cdots & \vdots \\ x_0^n & x_1^n & & x_I^n \end{pmatrix}.$$

Next, denote the vector of observations of the dependent variable as

$$Y^n = \begin{pmatrix} y^1 \\ y^2 \\ \vdots \\ y^n \end{pmatrix}.$$

The optimal parameter vector $\bar{\theta}$ (after n observations) is given by

$$\bar{\theta} = [(X^n)^T X^n]^{-1} (X^n)^T Y^n. \tag{3.41}$$

Linear models

- Sparse models
  - » Assume we approximate a function using a linear model

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

  - » Now consider what happens when the set of features F is large.
  - » Imagine we have a dataset consisting of $(f^n, S^n)_{n=1}^N$ where f^n is the observed response given information in S^n. We can fit our model using

$$\min_{\theta} \left(\sum_{n=1}^{N} (f^n - \bar{F}(S^n|\theta) \right)^2$$

  - » We may have too many features, but it is easily the case that we may have nonzero values of θ_i, even though there is no explanatory value.

Linear models

- Sparse models
  - » We can control how many elements of the vector θ are nonzero by including a *regularization factor:*

$$\min_{\theta}\left(\sum_{n=1}^{N}(f^n-\bar{F}(S^n|\theta)\right)^2+\lambda\sum_{i=1}^{I}|\theta_i|$$

  - » We have to test different values of λ, get the corresponding vector $\theta(\lambda)$, and then see how well this fits the data.
  - » Standard procedure is to fit on 80 percent of the data, test on 20 percent, and then rotate through the dataset five times so that each fifth of the data is used for testing at one point.
  - » Search over different values of λ to find the value of λ that produces the smallest error on the *testing* data.
  - » This is called *cross validation.*

Linear models

- Recursive least squares-Stationary data

3.5.1 Recursive least squares for stationary data

In the setting of adaptive algorithms in stochastic optimization, estimating the coefficient vector θ using batch methods such as equation (3.46) would be very expensive. Fortunately, it is possible to compute these formulas recursively. The updating equation for θ is

$$\theta^n = \theta^{n-1} - H^n \phi^n \hat{\varepsilon}^n, \tag{3.47}$$

where H^n is a matrix computed using

$$H^n = \frac{1}{\gamma^n} B^{n-1}. \tag{3.48}$$

The error $\hat{\varepsilon}^n$ is computed using

$$\hat{\varepsilon}^n = \overline{V}_s(\theta^{n-1}) - \hat{v}^n. \tag{3.49}$$

Note that it is common in statistics to compute the error in a regression using "actual minus predicted" while we are using "predicted minus actual" (see also equation (17.21) above). Our sign convention is motivated by the derivation from first principles of optimization. B^{n-1} is an $|\mathcal{F}|$ by $|\mathcal{F}|$ matrix which is updated recursively using

$$B^n = B^{n-1} - \frac{1}{\gamma^n}(B^{n-1}\phi^n(\phi^n)^T B^{n-1}). \tag{3.50}$$

γ^n is a scalar computed using

$$\gamma^n = 1 + (\phi^n)^T B^{n-1} \phi^n. \tag{3.51}$$

Parametric models

Nonlinear models

Nonlinear parametric models

3.6.1 Maximum likelihood estimation

The most general method for estimating nonlinear models is known as maximum likelihood estimation. Let $f(x|\theta)$ the function given θ, and assume that we observe

$$y = f(x|\theta) + \epsilon$$

where $\epsilon\, N(0, \sigma^2)$ is the error with density

$$f^{\epsilon}(w) = \frac{1}{\sqrt{2\pi}\sigma} \exp \frac{w^2}{2\sigma^2}.$$

Now imagine that we have a set of observations $(y^n, x^n)_{n=1}^N$. The likelihood of observing $(y^n)_{n=1}^N$ is given by

$$L(y|x, \theta) = \Pi_{n=1}^N \exp \frac{(y^n - f(x^n|\theta))^2}{2\sigma^2}.$$

It is common to use the log likelihood $\mathcal{L}(y|x, \theta) = \log L(y|x, \theta)$, which gives us

$$\mathcal{L}(y|x, \theta) = \sum_{n=1}^{N} (y^n - f(x^n|\theta))^2, \tag{3.61}$$

where we are always dropping constants.

Equation (3.61) can be used by nonlinear programming algorithms to estimate the parameter vector θ. This assumes that we have a batch dataset $(y^n, x^n)_{n=1}^N$, which is not our typical setting. In addition, the log likelihood $\mathcal{L}(y|x, \theta)$ can be nonconvex when $f(x|\theta)$ is nonlinear in θ, which further complicates the optimization challenge.

Nonlinear parametric models

- Notes:
  - » Maximum likelihood estimation for nonlinear models is highly nonlinear and nonconvex.
  - » These are often considered to be some of the hardest classes of nonlinear programming problems.
  - » We will introduce algorithms for this next week.

Nonlinear parametric models

8.2.3 Bayesian updating equations

The next step is to design the updating equations for the probability vector p^n after running an experiment with $x = x^n$ and observing a response y^{n+1}. We start with Bayes theorem

$$\mathbb{P}(A|B) = \frac{\mathbb{P}(B|A)\mathbb{P}(A)}{\mathbb{P}(B)}.$$

Now interpret the event A as the event that $\theta = \theta_k$, and B as the new information y^{n+1}. We are going to let H^n be the history of our experiments where

$$H^n = (S^0, x^0, \hat{y}^1, x^1, \hat{y}^2, \ldots, x^{n-1}, \hat{y}^n).$$

We can write our belief probabilities as

$$p_k^n = \mathbb{P}[\theta = \theta_k | H^n].$$

We note that we can write Bayes' theorem where all of the probabilities are conditioned on a third event C, as in

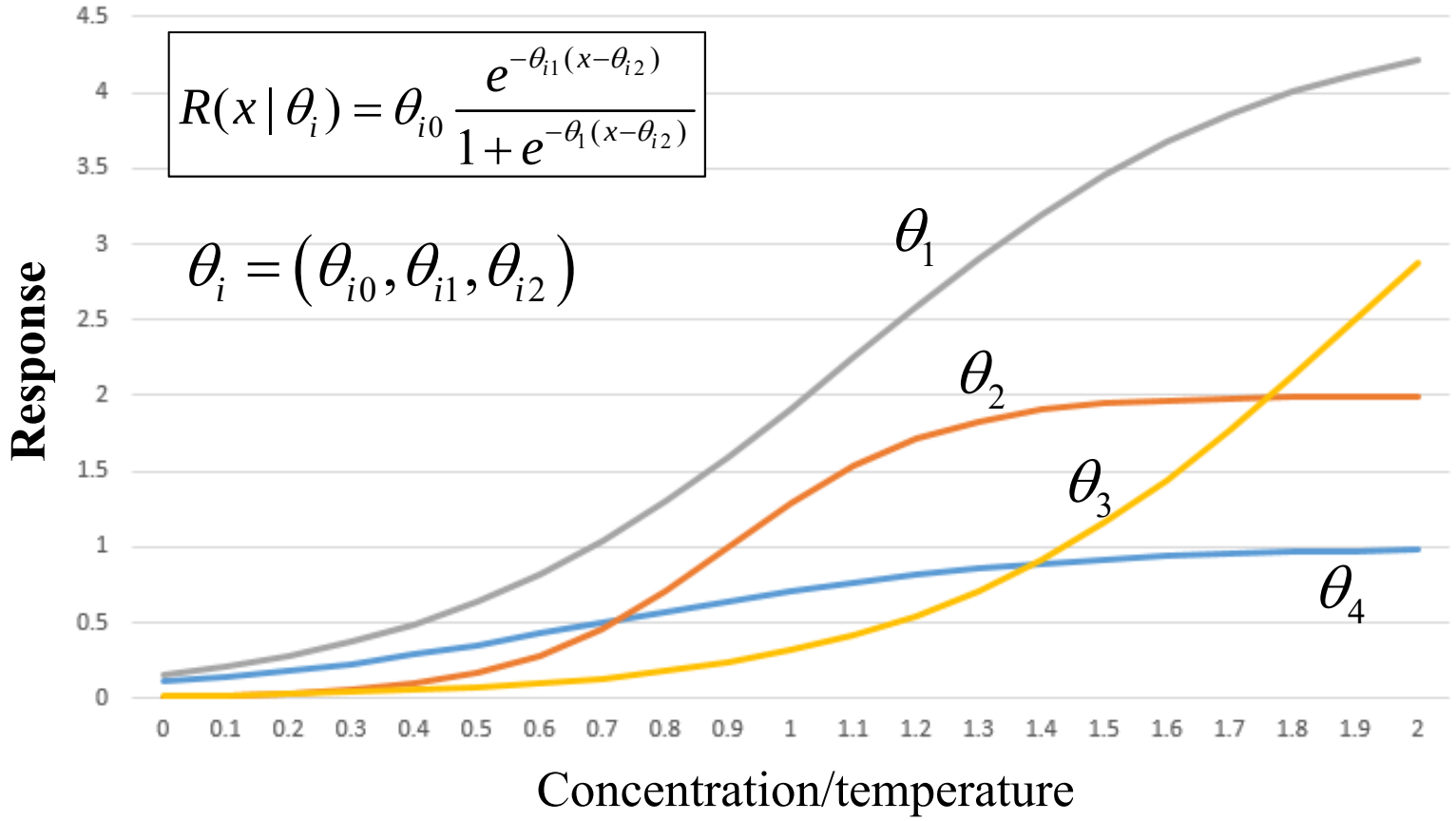
$$\mathbb{P}(A|B,C) = \frac{\mathbb{P}(B|A,C)\mathbb{P}(A|C)}{\mathbb{P}(B|C)}.$$

In our setting, the new event C is our history H^n. Adapting this to our setting gives us

$$\mathbb{P}[\theta = \theta_k | y^{n+1} = y, H^n] = \frac{\mathbb{P}[\hat{y}^{n+1} = y | \theta_k, H^n]\mathbb{P}[\theta = \theta_k | H^n]}{\mathbb{P}[\hat{y}^{n+1} = y | H^n]}$$

Sampled belief models

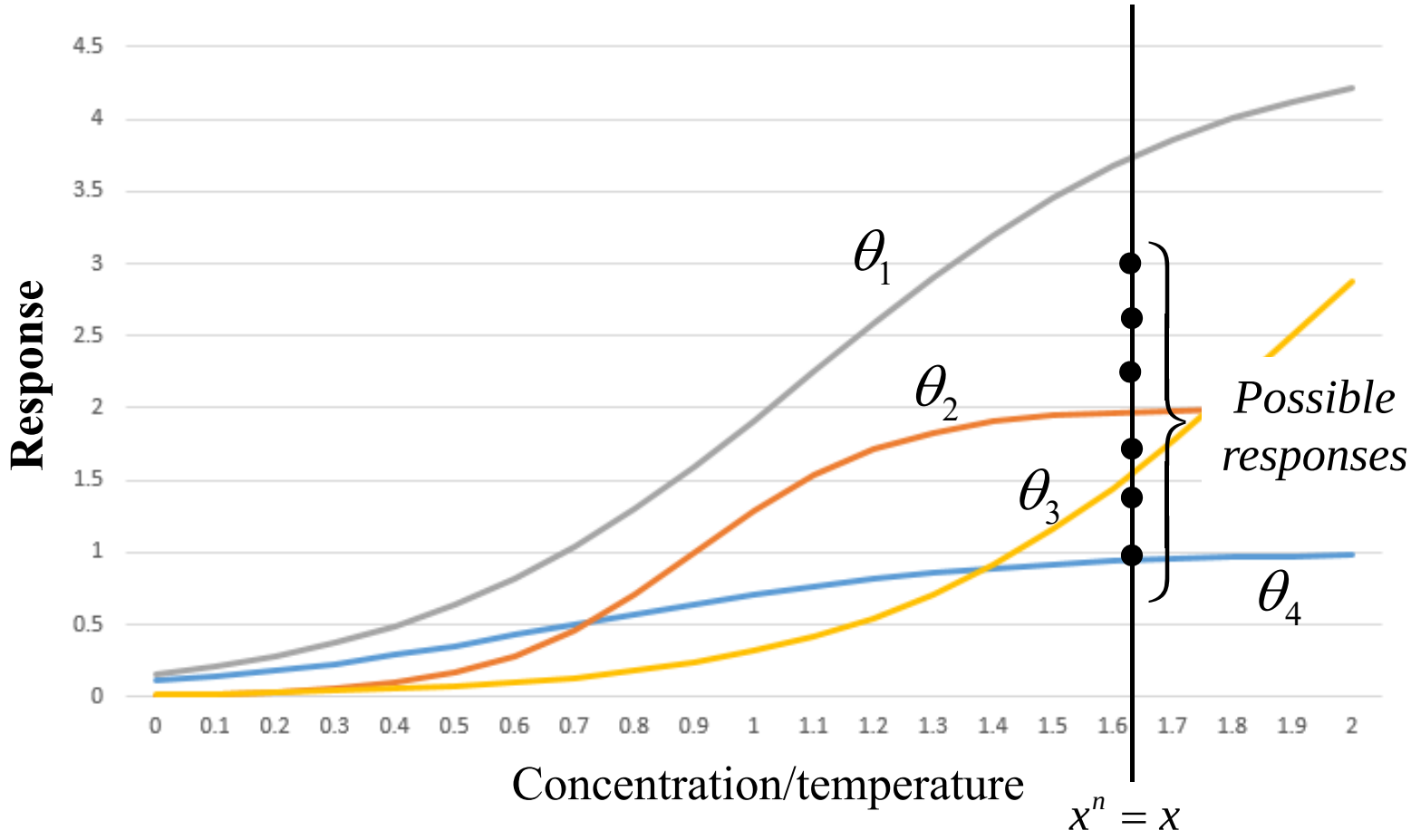
- A sampled belief model



Thickness proportional to p_k^n

Sampled belief models

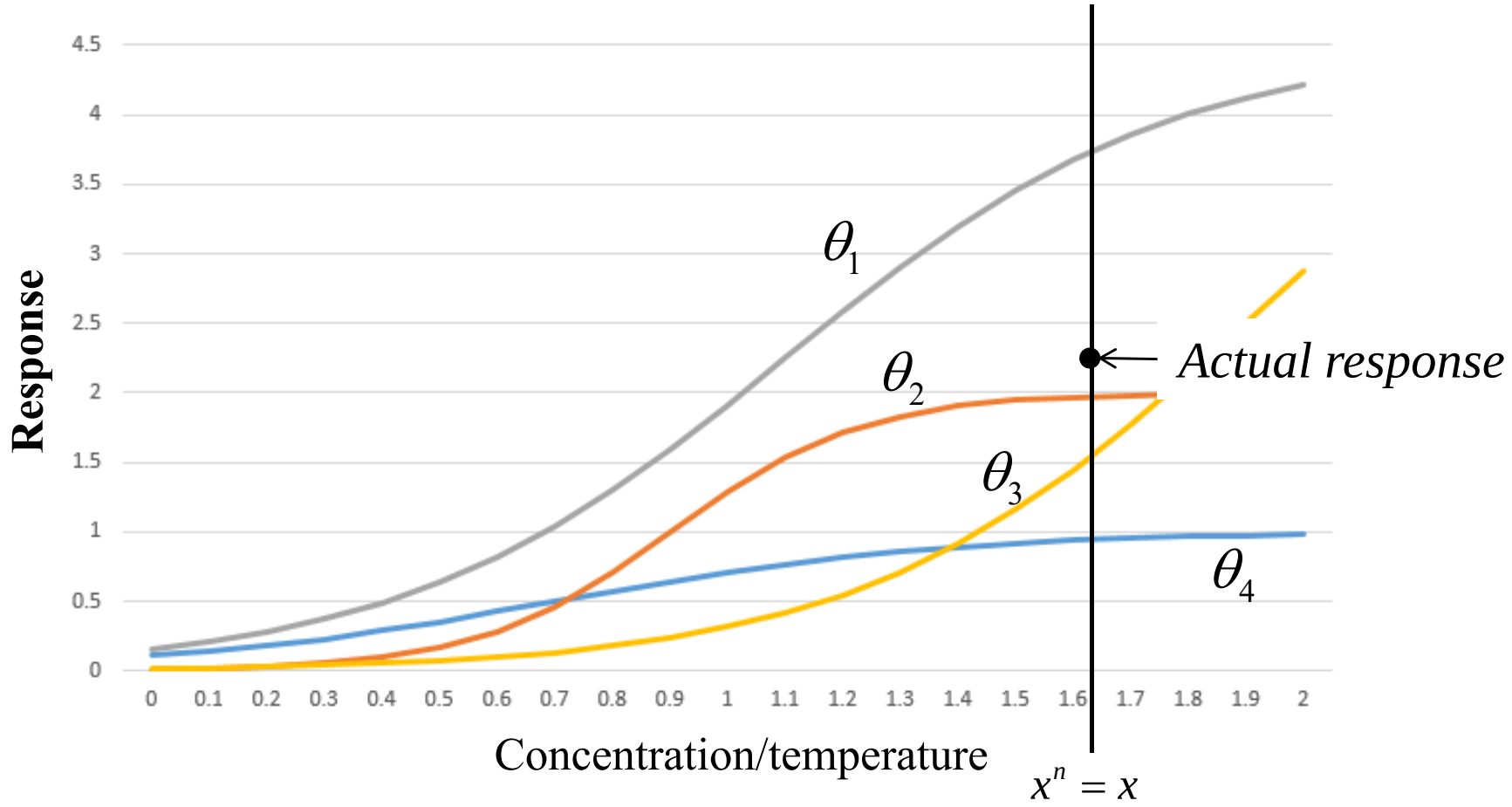
- Possible experimental outcomes



Thickness proportional to p_k^n

Sampled belief models

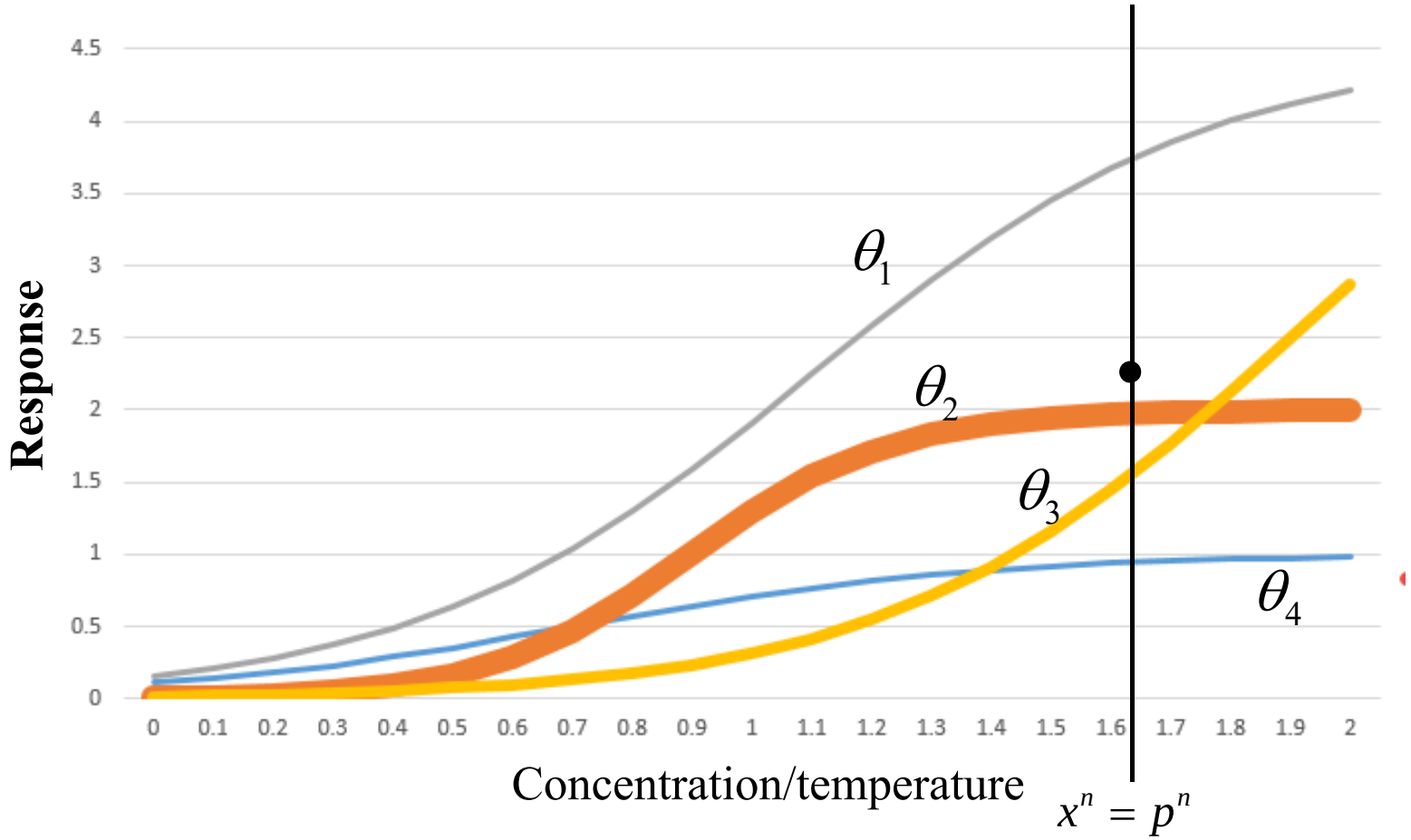
- Running an experiment – here we see the actual r



Thickness proportional to p_k^n

Sampled belief models

- Updated posterior reflects proximity to each curve



Thickness proportional to p_k^n

Sampled belief models

- Notes:
  - » Working with a sampled belief model can be exceptionally easy and flexible.
  - » We can apply rules to make sure we are creating valid sampled estimates.
    - E.g. we might want to ensure that some coefficients are positive.
  - » The challenge is that if θ is high-dimensional, a small sample (e.g. 100) may not contain any samples that are close to the true optimal solution.
  - » There are strategies that involve periodically re-generating new samples that can overcome this.

Neural networks

(only covered briefly in lecture)

Neural networks

- Start by illustrating with linear model

Up to now, we have considered approximation functions of the form

$$\overline{V}(S) = \sum_{f \in \mathcal{F}} \theta_f \phi_f(S),$$

where $\mathcal{F}$ is our set of features, and $(\phi_f(S))_{f \in \mathcal{F}}$ are the basis functions which extract what are felt to be the important characteristics of the state variable which explain the value of being in a state. We have seen that when we use an approximation that is linear in the parameters, we can estimate the parameters θ recursively using standard methods from linear regression. For example, if R_i is the number of resources of type i, our approximation might look like

$$\overline{V}(R|\theta) = \sum_{i \in \mathcal{I}} \left(\theta_{1i} R_i + \theta_{2i} R_i^2\right).$$

Now assume that we feel that the best function might not be quadratic in R_i, but we are not sure of the precise form. We might want to estimate a function of the form

$$\overline{V}(R|\theta) = \sum_{i \in \mathcal{I}} \left(\theta_{1i} R_i + \theta_{2i} R_i^{\theta_3}\right).$$

Now we have a function that is nonlinear in the parameter vector $(\theta_1, \theta_2, \theta_3)$, where θ_1 and θ_2 are vectors and θ_3 is a scalar. If we have a training dataset of state-value observations, $(\hat{v}_n, R_n)_{n=1}^N$, we can find θ by solving

$$\min_{\theta} \sum_{n=1}^{N} \left(\hat{v}_n - \overline{V}(R_n|\theta)\right)^2,$$

which generally requires the use of nonlinear programming algorithms. One challenge is that nonlinear optimization problems do not lend themselves to the simple recursive updating equations that we obtained for linear (in the parameters) functions. But more problematic is that we have to experiment with various functional forms to find the one that fits best.

Neural networks are, ultimately, a form of statistical model which, for our application, predicts the value of being in a state as a function of the state, using a series of observations of states and values. The simplest neural network is nothing more than a linear regression model. If we are in post-decision state S_t^a, we are going to make the random transition $S_{t+1} = S^M(S_t, a_t, W_{t+1}(\omega))$ and then observe a random value $\hat{v}_{t+1}$ from being in state S_{t+1}. We would like to estimate a statistical function $f_t(S_t^a)$ (the same as $\overline{V}_t(S_t^a)$) that predicts $\hat{v}_{t+1}$. To write this in the traditional notation of regression, let $X_t = S_t^a$ where $X_t = (X_{t1}, X_{t2}, \ldots, X_{tI})$ (we assume that all the components X_{ti} are numerical). If we use a linear model, we might write

$$f_t(X_t) = \theta_0 + \sum_{i=1}^{I} \theta_i X_{ti}.$$

In the language of neural networks, we have I inputs (we have $I + 1$ parameters since we also include a constant term), which we wish to use to estimate a single output $\hat{v}_{t+1}$ (a random observations of being in a state). The relationships are illustrated in figure 3.5 where we show the I inputs which are then "flowed" along the links to produce $f_t(X_t)$. After this, we then learn the sample realization $\hat{v}_{t+1}$ that we were trying to predict, which allows us to compute the error $\epsilon_{t+1} = \hat{v}_{t+1} - f_t(X_t)$. We would like to find a vector θ that solves

$$\min_{\theta} \mathbb{E}\frac{1}{2}(f_t(X_t) - \hat{v}_{t+1})^2.$$

Let $F(\theta) = \mathbb{E}(0.5(f_t(X_t) - \hat{v}_{t+1})^2)$, and let $F(\theta, \hat{v}_{t+1}(\omega)) = 0.5(f_t(X_t) - \hat{v}_{t+1}(\omega))^2$ where $\hat{v}_{t+1}(\omega)$ is a sample realization of the random variable $\hat{v}_{t+1}(\omega)$. As before, we can solve this iteratively using a stochastic gradient algorithm

$$\theta_{t+1} = \theta_t - \alpha_t \nabla_\theta F(\theta_t, \hat{v}_{t+1}(\omega)),$$

Neural networks

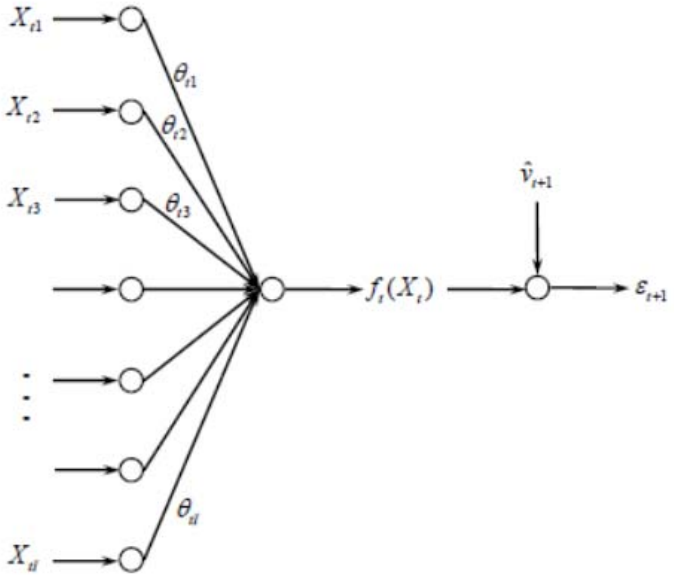
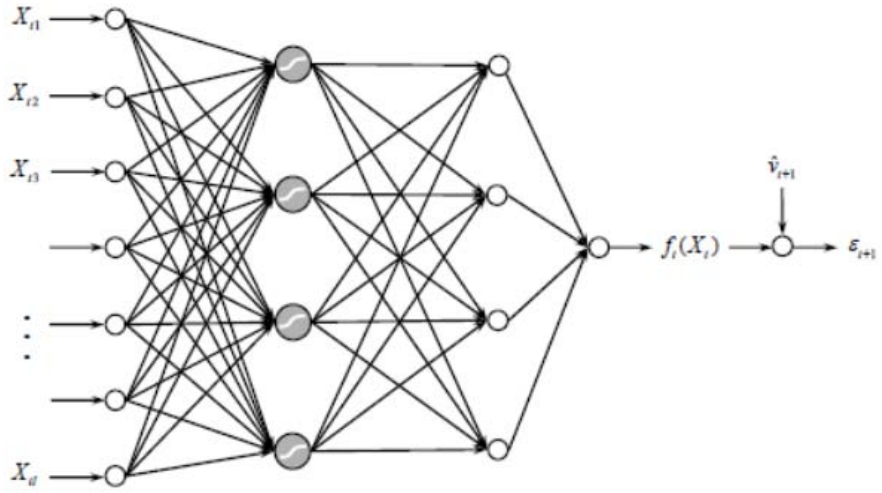


Figure 3.5 Neural networks with a single layer.



Neural networks

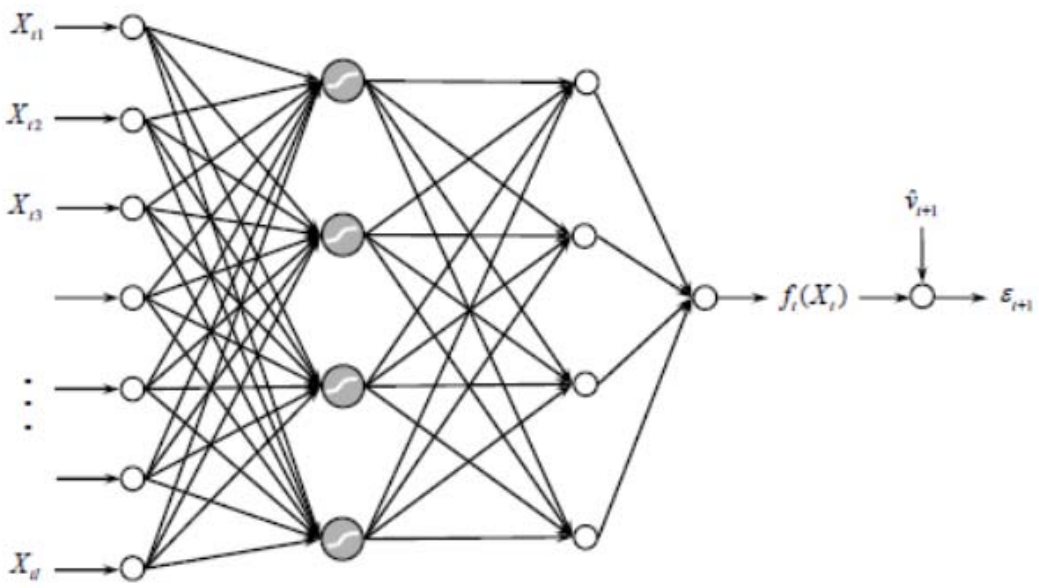


Figure 3.6 A three-layer neural network.

Here, the first linear layer produces J outputs given by

$$Y_{tj}^{(2)} = \sum_{i \in \mathcal{I}^{(1)}} \theta_{ij}^{(1)} X_{ti}^{(1)}, \quad j \in \mathcal{I}^{(2)}.$$

$Y_{tj}^{(2)}$ becomes the input to a nonlinear *perceptron* node which is characterized by a nonlinear function that may dampen or magnify the input. A typical functional form for a perceptron

Neural networks

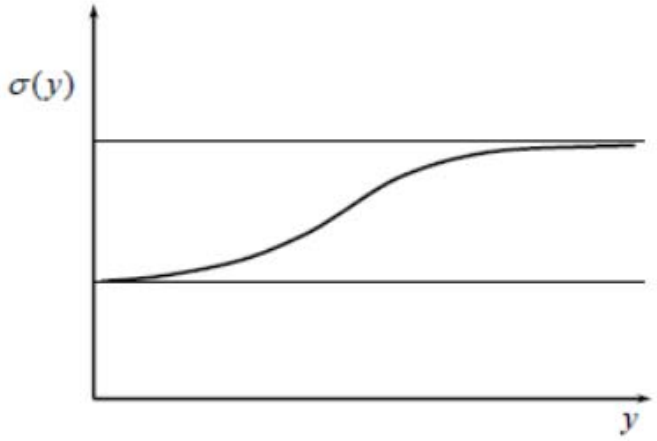


Figure 3.7 Illustrative logistics function for introducing nonlinear behavior into neural networks.

node is the logistics function given by

$$\sigma(y) = \frac{1}{1 + e^{-\beta y}},$$

where β is a scaling coefficient. The function $\sigma(y)$ is illustrated in figure 3.7. $\sigma(x)$ introduces nonlinear behavior into the communication of the "signal" X_t.

We next calculate

$$X_{ti}^{(2)} = \sigma(Y_{ti}^{(2)}), \quad i \in \mathcal{I}^{(2)}$$

and use $X_{ti}^{(2)}$ as the input to the second linear layer. We then compute

$$Y_{tj}^{(3)} = \sum_{i \in \mathcal{I}^{(2)}} \theta_{ij}^{(2)} X_{ti}^{(2)}, \quad j \in \mathcal{I}^{(3)}.$$

Finally, we compute the single output using

$$f_t = \sum_{i \in \mathcal{I}^{(3)}} \theta_i^{(3)} X_{ti}^{(3)}.$$

Neural networks

- Notes:
  - » Neural networks are very high-dimensional architectures
  - » When used with classical batch learning, they require very large datasets.
  - » There are settings where we can use iterative learning (e.g. for value functions and policies), but these may require many millions of iterations (e.g. for deep neural networks).
  - » Shallow neural networks have been popular for many years for deterministic engineering control problems.
  - » Risk of overfitting stochastic problems is high.

Case study with neural networks

Neural network case study

yieldPlanet | Maximizing hotel revenue. Simply

English | Login

Products | Become a partner | YieldPlanet | Blog | Contact us

We help **hotels** manage their **online channels** from a single **point of control**

YieldPlanet provides online Distribution and Revenue Management solutions for more than 2500 hotels of all types and sizes around the world. With advanced capabilities built into easy to use tools, we will help you to increase occupancy, grow revenue and maximize profit.

Discover our solutions

Apartments, vacation | Independent hotels | Hotel chains

Neural network case study

- Basic idea
  - » Hotels have a range of room layouts as well as reservation plans, all with their own rates. But one is known as the "BAR" rate (best available rate).
  - » We assume that the hotel has worked out a reasonable solution of how the different room rate plans should be priced relative to the BAR rate.
  - » The "practical policy" is to create a lookup table policy that maps the "state" (what we know) to an adjustment of the BAR rate.
    - State variable requires using expert judgment to identify important factors
    - State variables require combining numerical and qualitative factors (such as competition).

Neural network case study

- Zak's rules
  - » Developed by award-winning Revenue Manager Zak Ali from YieldPlanet
  - » Lookup table approach to price-setting
  - » Derived from 6 binary factors and time to stay date
    - Based on intuition and refined through observations
    - Not gradient-based: same effect regardless of small changes within individual factors
    - Minimal look-back: Depends heavily on comparing data from "this year" (TY) to "last year" (LY).

Neural network case study

- ABOBTY vs. ABOBLY
  - » ABOB = Actual Business on the Books
    - Refers to the number of reservations made for the stay date
  - » Compare to the equivalent date last year *at the same number of days prior to arrival*
- APRICETY vs. APRICELY
  - » APRICE = Actual Price
  - » Takes into account the current BAR (Best Available Rate), determined from business booked and forecasting relative to budget

Neural network case study

- Average Weekly Pickup TY vs. LY
  - » The average rate that business is booked for stay date
    - Use data from the previous 7 days
- FTY vs. FLY
  - » Forecast this year is total final occupancy forecasted in terms of number of rooms booked for stay date
    - Based on (more quantitatively derived) forecasting estimates calculated by dividing customers into segments
    - Done by separate forecasting team
  - » Forecast last year is final realized occupancy

Neural network case study

- COMPPRICE vs. MYPRICE
  - » COMPPRICE = Prices of competition
  - » How do we determine competitors?
    - Difficult question to answer
    - Geography, Star Rating, Similar Price Range?
    - Historically been based on qualitative/anecdotal evidence
- Is Price Elastic?
  - » Elasticity: % Change in Quantity / % Change in Price
  - » In practice, harder to determine
    - Use price range of competition for comparable room type/rate plan combination
    - Competition not always available (or could be irrelevant due to special convention or sold out status)

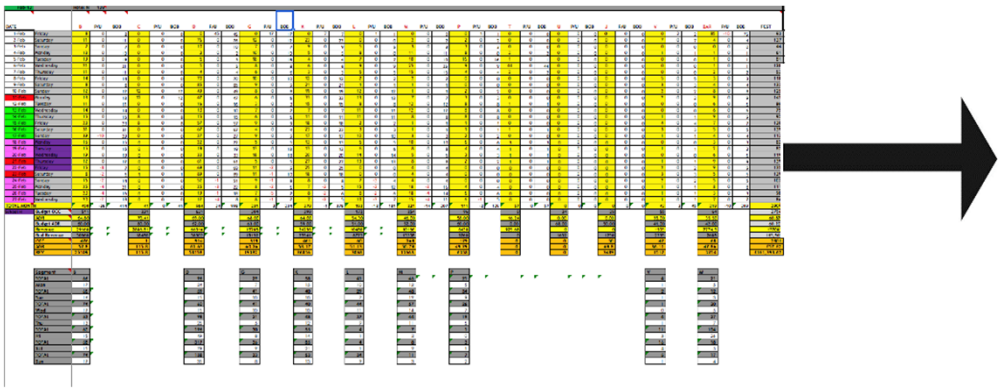
Neural network case study

- Pricing spreadsheet
  - » First six columns determine “state”
  - » Last column is recommended price above/below “BAR” rate

1	ABOBTY/ABOBLY	APRICETY/APRICELY	AVGPICKTY/AVGPICKLY(WEEKLY)	FLY/FTH	MYPRICE/COMPPRICE	ELASTIC(YES/NO)	PERCENTAGE
3	1	1	1	1	1	1	20.00%
4	1	1	1	1	1	-1	-11.00%
5	1	1	1	1	-1	1	16.00%
6	1	1	1	1	-1	-1	-3.00%
7	1	1	1	-1	1	1	9.00%
8	1	1	1	-1	1	-1	1.00%
9	1	1	1	-1	-1	1	6.00%
10	1	1	1	-1	-1	-1	-8.00%
11	1	1	-1	1	1	1	5.50%
12	1	1	-1	1	1	-1	-15.00%
13	1	1	-1	1	-1	1	14.00%
14	1	1	-1	1	-1	-1	-8.00%
15	1	1	-1	-1	1	1	12.00%
16	1	1	-1	-1	1	-1	-2.00%
17	1	1	-1	-1	-1	1	12.00%
18	1	1	-1	-1	-1	-1	-8.00%
19	1	-1	1	1	1	1	15.00%
20	1	-1	1	1	1	-1	-8.00%
21	1	-1	1	1	-1	1	7.00%
22	1	-1	1	1	-1	-1	-8.00%
23	1	-1	1	-1	1	1	7.50%

Neural network case study

- Example
  - » Assume we are 5 days away from stay date



Sample data

	2014	2015
ABOB	6423.42	7302.64
BAR	126.4	130.2
Avg Pickup	4.32	2.43
Forecast	7800	8400
Actual	7645.23	n/a
Comp. Price	140.5	145.5
Elasticity	1.43	1.37

  - » Use numbers to compute state
    - If ABOBTY > ABOBLY then $S_{t1} = 1,$ else $= -1$
    - If APRICETY > APRICELY then $S_{t2} = 1,$ else $= -1$
    - Pickup TY < Pickup LY then $S_{t3} = 1,$ else $= -1$
    - FTY > FLY then $S_{t4} = 1,$ else $= -1$
    - COMPRICE > MYPRICE then $S_{t5} = 1,$ else $= -1$
    - Unit demand is elastic [>1] then $S_{t6} = 1,$ else $= -1$

$$S_t = (S_{t1}, S_{t2}, S_{t3}, S_{t4}, S_{t5}, S_{t6}) = \text{"state" at time } t$$

Neural network case study

- The lookup table policy for Zak's rules:

$$P^{\pi}(S_t) = p^{BAR} \cdot \left(1 - \delta p^{ZAK}(S_t)\right)$$

S_t

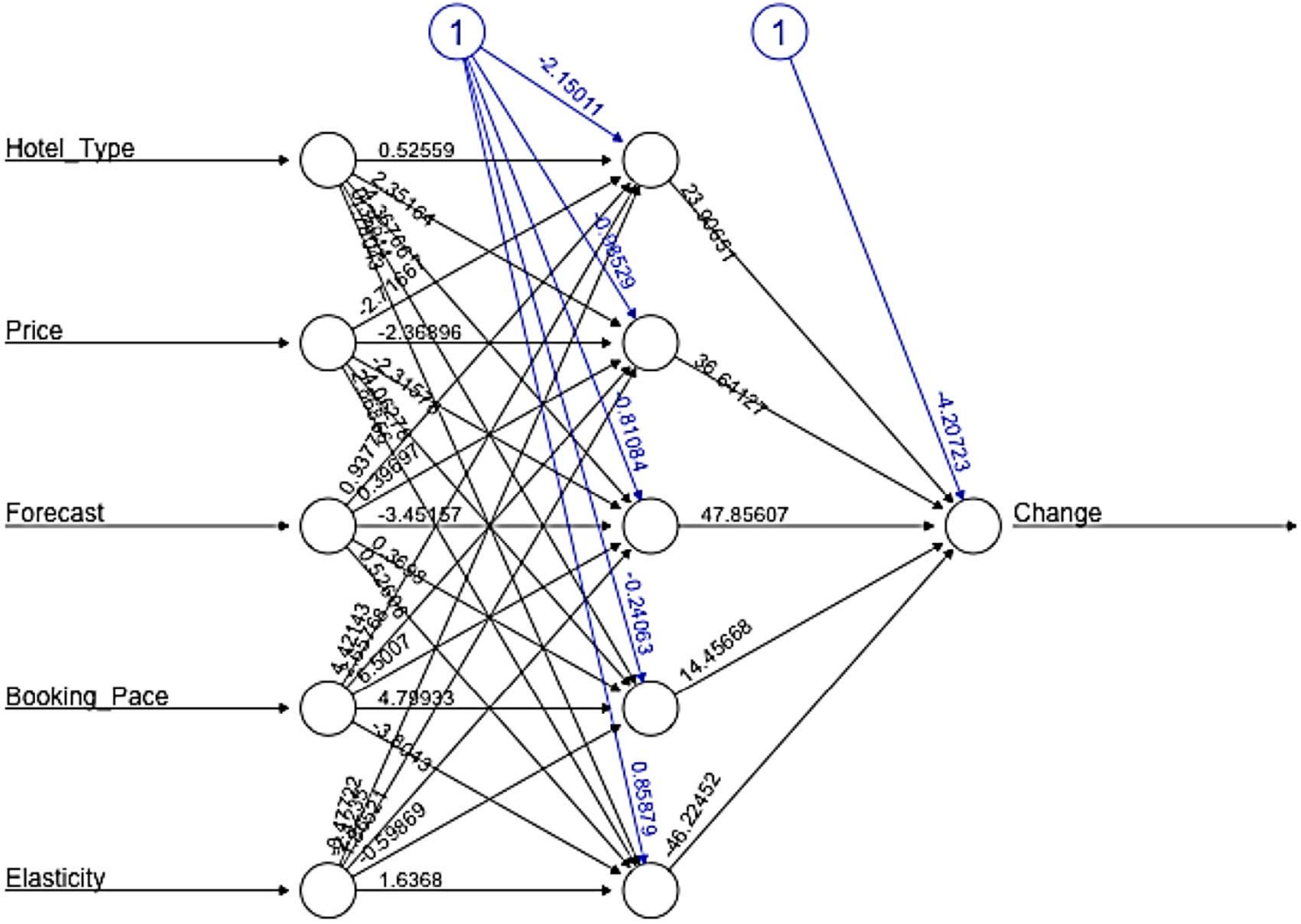
	ABOBTY/ABOBLY	APRICETY/APRICELY	AVGPICKTY/AVGPICKLY(WEEKLY)	FLY/FTH	MYPRICE/COMPPRICE	ELASTIC(YES/NO)	PERCENTAGE
3	1	1	1	1	1	1	20.00%
4	1	1	1	1	1	-1	-11.00%
5	1	1	1	1	-1	1	16.00%
6	1	1	1	1	-1	-1	-3.00%
7	1	1	1	-1	1	1	9.00%
8	1	1	1	-1	1	-1	1.00%
9	1	1	1	-1	-1	1	6.00%
10	1	1	1	-1	-1	-1	-8.00%
11	1	1	-1	1	1	1	5.50%
12	1	1	-1	1	1	-1	-15.00%
13	1	1	-1	1	-1	1	14.00%
14	1	1	-1	1	-1	-1	-8.00%
15	1	1	-1	-1	1	1	12.00%
16	1	1	-1	-1	1	-1	-2.00%
17	1	1	-1	-1	-1	1	12.00%
18	1	1	-1	-1	-1	-1	-8.00%
19	1	-1	1	1	1	1	15.00%
20	1	-1	1	1	1	-1	-8.00%
21	1	-1	1	1	-1	1	7.00%
22	1	-1	1	1	-1	-1	-8.00%
23	1	-1	1	-1	1	1	7.50%

Neural network case study

- Zak's rules:
  - » Use binary factors and number of days prior to arrival (0-3, 4-7, 8-14, etc.) to determine percentage increase/decrease in price
  - » Price changes become more dramatic closer to stay date
    - -20% to 20% range in lookup table corresponding to 0-3 days prior to arrival
    - -8% to 8% range on decision tree corresponding to 71-90 days prior to arrival
  - » The numbers themselves are more based on intuition rather than quantitative analysis
    - Percentages are typically integer increases/decreases, rather than more precise numerical estimates

Neural network case study

- Neural network for pricing hotel rooms

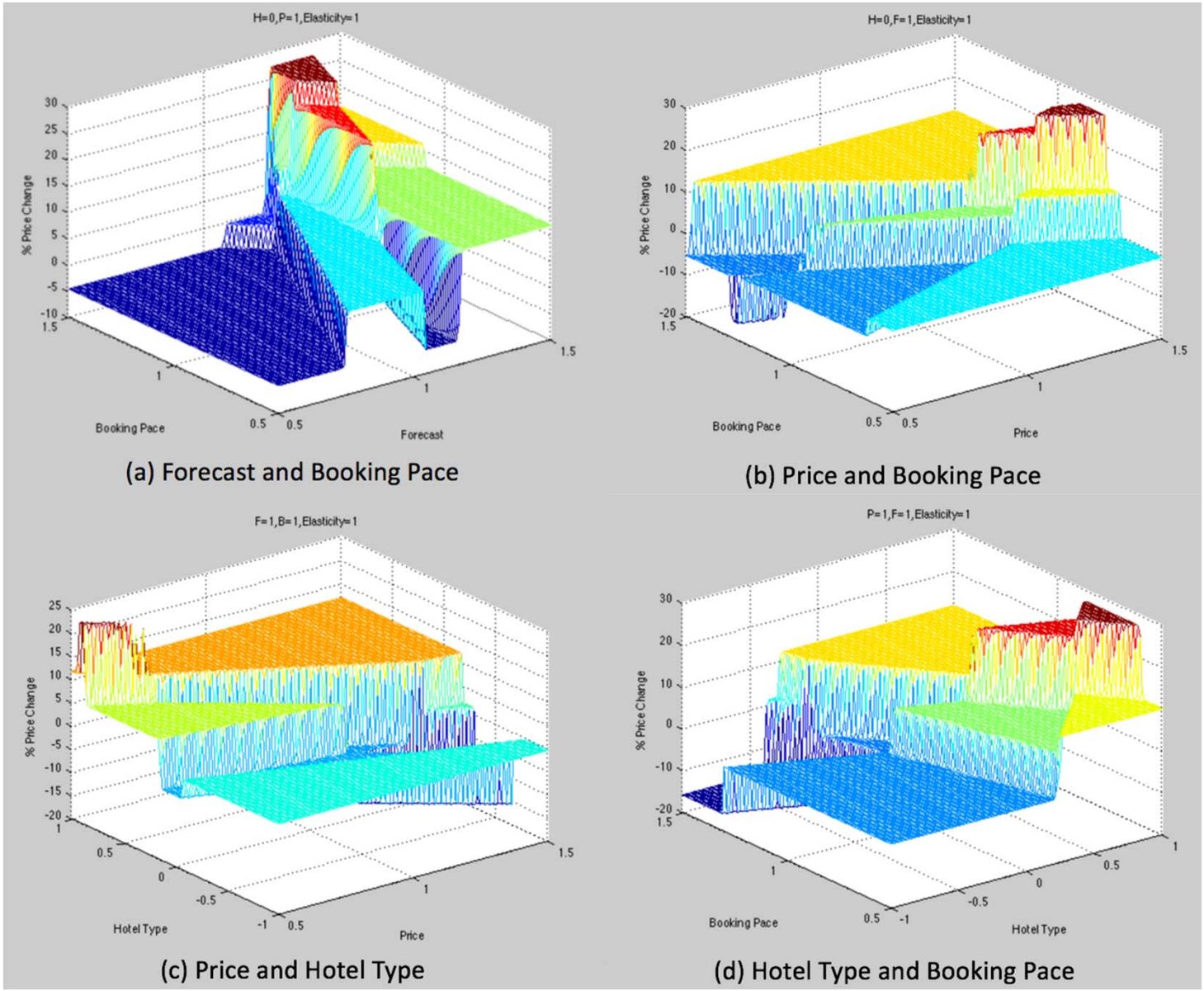


Error: 2170.018915 Steps: 609

Neural network case study

- Notes:
  - » This neural network, which is quite small, was fitted on a dataset of state-action pairs constructed with the expert "Zak."
  - » The graphs that follow show the behavior of the neural network as a function of different inputs (e.g. price, hotel type, ..)
  - » The curves are not well behaved. What is happening here is that there is overfitting. Even though this is a small neural network, the number of parameters is still large relative to the size of the dataset.

Neural network case study



Nonparametric models

Nonparametric methods

- Definition:
  - » A nonparametric model has the property that as the amount of data goes to infinity, modeling errors go to zero.

- Illustrate:
  - » K-nearest neighbor
  - » Kernel regression

Nonparametric methods

3.7.1 K-nearest neighbor

Perhaps the simplest form of nonparametric regression forms estimates of functions by using a weighted average of the k-nearest neighbors. As above, we assume we have a response y^n corresponding to a measurement $x^n = (x_1^n, x_2^n, \ldots, x_I^n)$. Let $\rho(x, x^n)$ be a distance metric between a query point x (in dynamic programming, this would be a state), and an observation x^n. Then let $\mathcal{N}^n(x)$ be the set of the k-nearest points to the query point x, where clearly we require $k \leq n$. Finally let $\bar{Y}^n(x)$ be the response function, which is our best estimate of the true function $Y(x)$ given the observations $x^1, \ldots, x^n$. When we use a k-nearest neighbor model, this is given by

$$\bar{Y}^n(x) = \frac{1}{k} \sum_{n \in \mathcal{N}^n(x)} y^n. \tag{3.64}$$

Thus, our best estimate of the function $Y(x)$ is made by averaging the k points nearest to the query point x.

Using a k-nearest neighbor model requires, of course, choosing k. Not surprisingly, we obtain a perfect fit of the data by using $k = 1$ if we base our error on the training dataset.

A weakness of this logic is that the estimate $\bar{Y}^n(x)$ can change abruptly as x changes continuously, as the set of nearest neighbors changes. An effective way of avoiding this behavior is using kernel regression, which uses a weighted sum of all data points.

Nonparametric methods

3.7.2 Kernel regression

Kernel regression has attracted considerable attention in the statistical learning literature. As with k-nearest neighbor, kernel regression forms an estimate $\bar{Y}(x)$ by using a weighted sum of prior observations which we can write generally as

$$\bar{Y}^n(x) = \frac{\sum_{m=1}^n K_h(x, x^m) y^m}{\sum_{m=1}^n K_h(x, x^m)} \tag{3.66}$$

where $K_h(x, x^m)$ is a weighting function that declines with the distance between the query point x and the measurement x^m. h is referred to as the *bandwidth* which plays an important scaling role. There are many possible choices for the weighting function $K_h(x, x^m)$. One of the most popular is the Gaussian kernel, given by

$$K_h(x, x^m) = e^{-\left(\frac{\|x - x^m\|}{h}\right)^2}.$$

where $\|\cdot\|$ is the Euclidean norm. Here, h plays the role of the standard deviation. Note that the bandwidth h is a tunable parameter that captures the range of influence of a measurement x^m. The Gaussian kernel, often referred to as *radial basis functions* in the ADP literature, provide a smooth, continuous estimate $\bar{Y}^n(x)$. Another popular choice of kernel function is the symmetric Beta family, given by

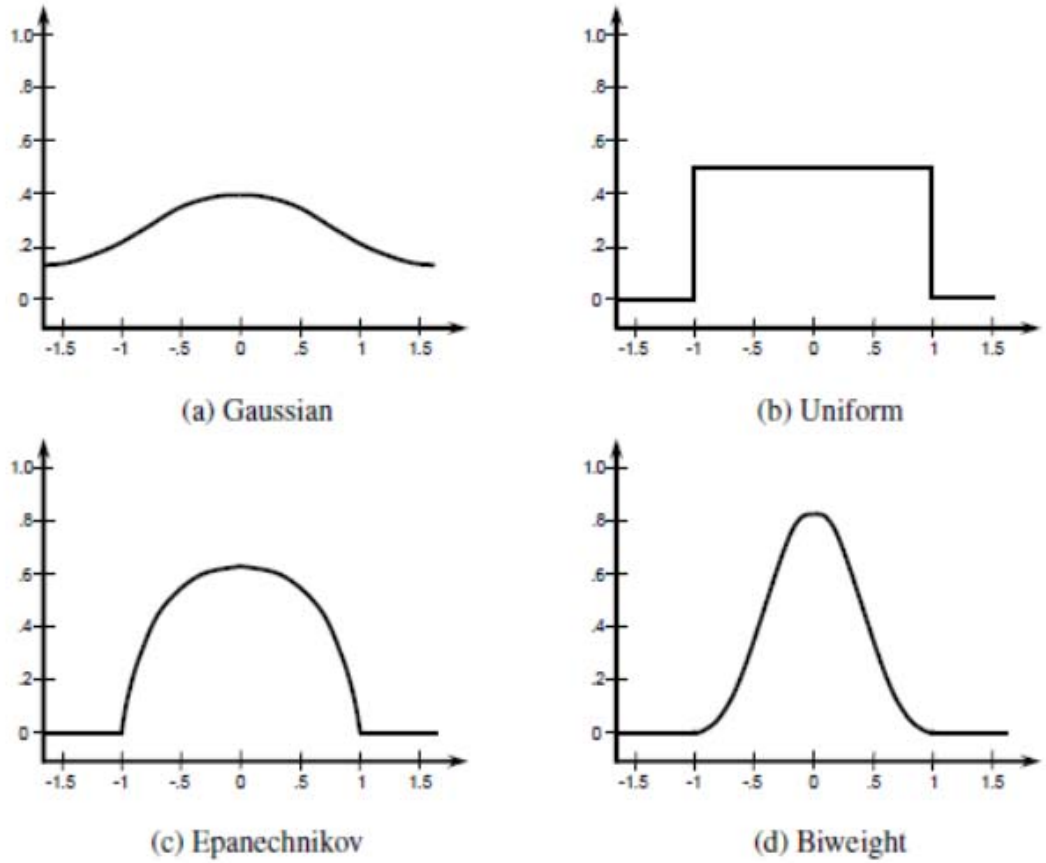
$$K_h(x, x^m) = \max(0, (1 - \|x - x^m\|)^2)^h.$$

Here, h is a nonnegative integer. $h = 1$ gives the uniform kernel; $h = 2$ gives the Epanechnikov kernel; and $h = 3$ gives the biweight kernel. Figure 3.8 illustrates each of these four kernel functions.

We pause to briefly discuss some issues surrounding k-nearest neighbors and kernel regression. First, it is fairly common in the ADP literature to see k-nearest neighbors

Nonparametric methods

- Kernels



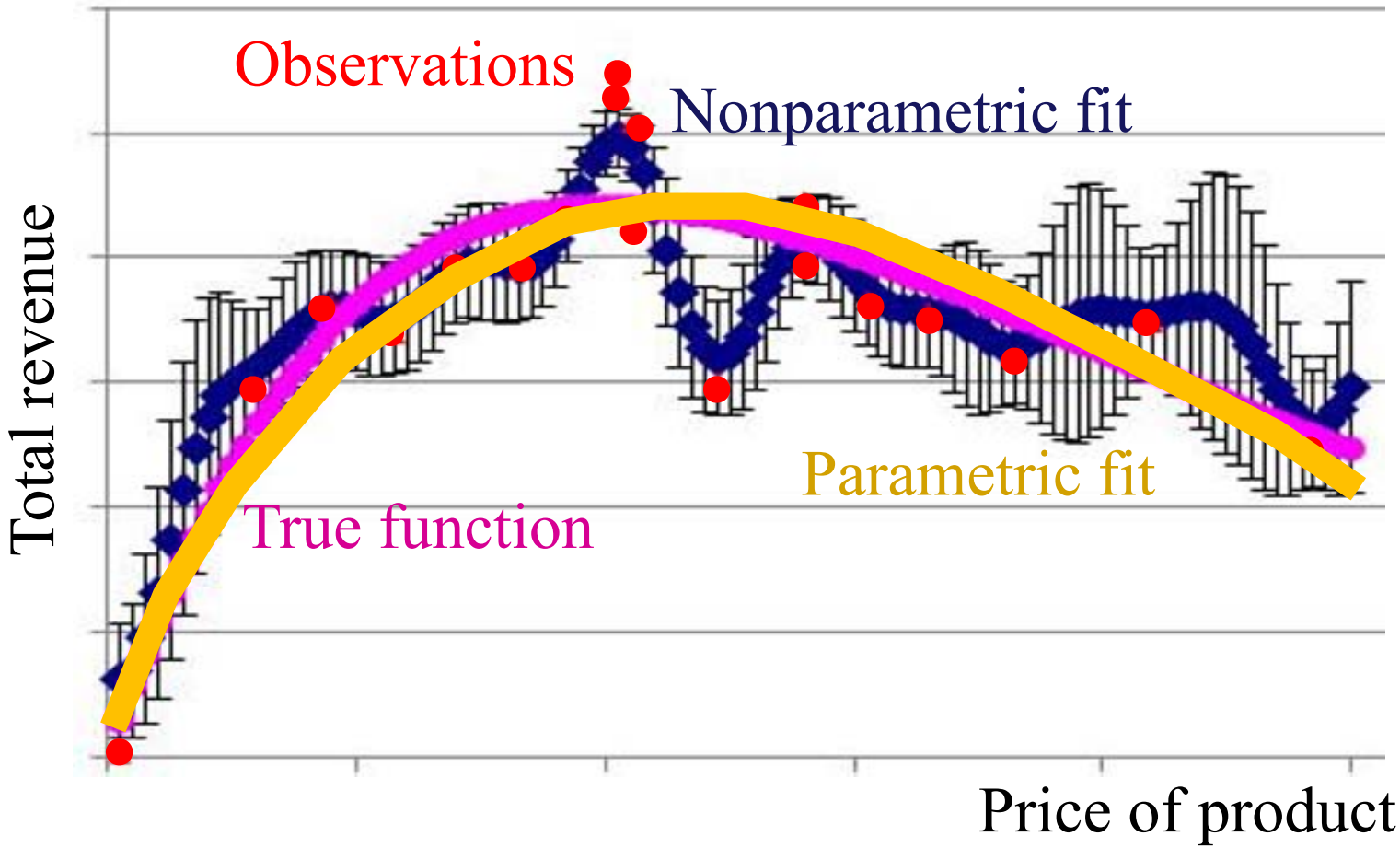
(a) Gaussian (b) Uniform

(c) Epanechnikov (d) Biweight

Figure 3.8 Illustration of Gaussian, uniform, Epanechnikov and biweight kernel weighting functions.

Nonparametric methods

- Parametric vs. nonparametric

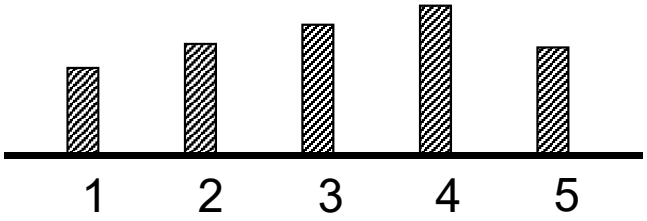


» Robust CFAs are *parametric*

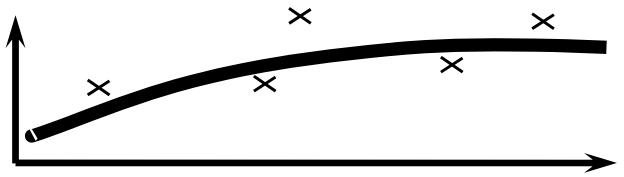
» Scenario trees are *nonparametric*

Nonparametric methods

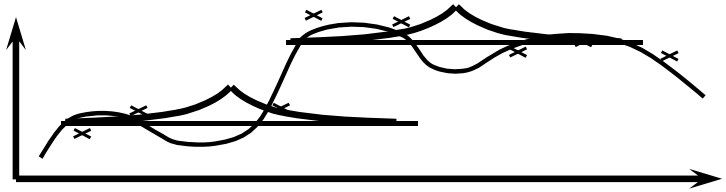
- Lookup table belief models (Frazier)
  - » Independent beliefs
  - » Correlated beliefs



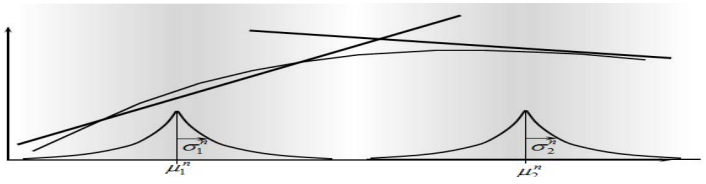
- Linear, parametric belief models (Frazier)



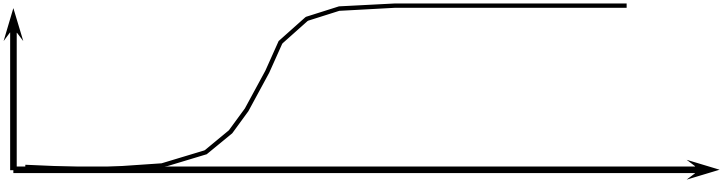
- Nonparametric models
  - » Hierarchical aggregation (Mes)
  - » Kernel regression (Barut)



- Local parametric models
  - » Dirichlet clouds with RBF (Jamshidi)
  - » KG derivation (Harvey Cheng)

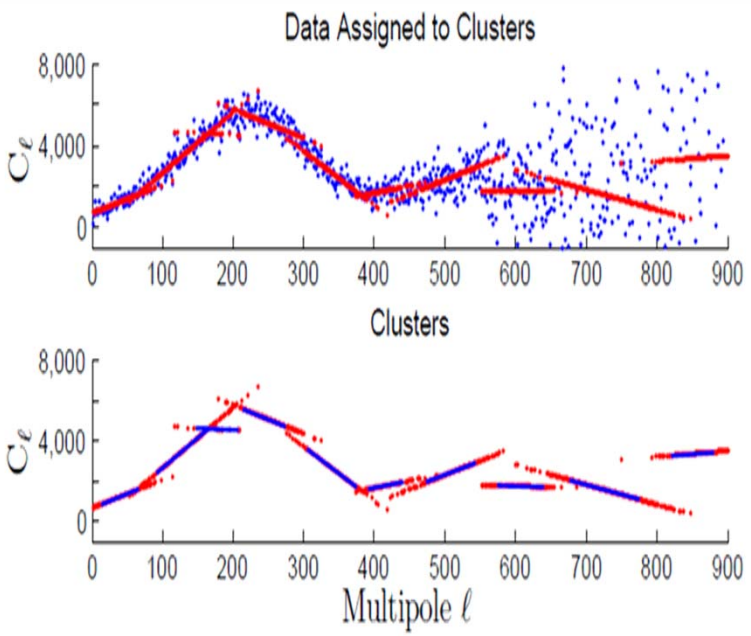


- Generalized linear models
  - » KG derivation (Si Chen)

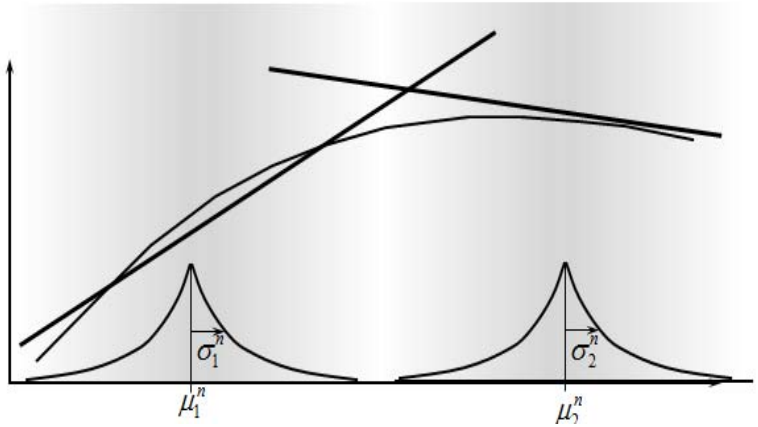


Nonparametric methods

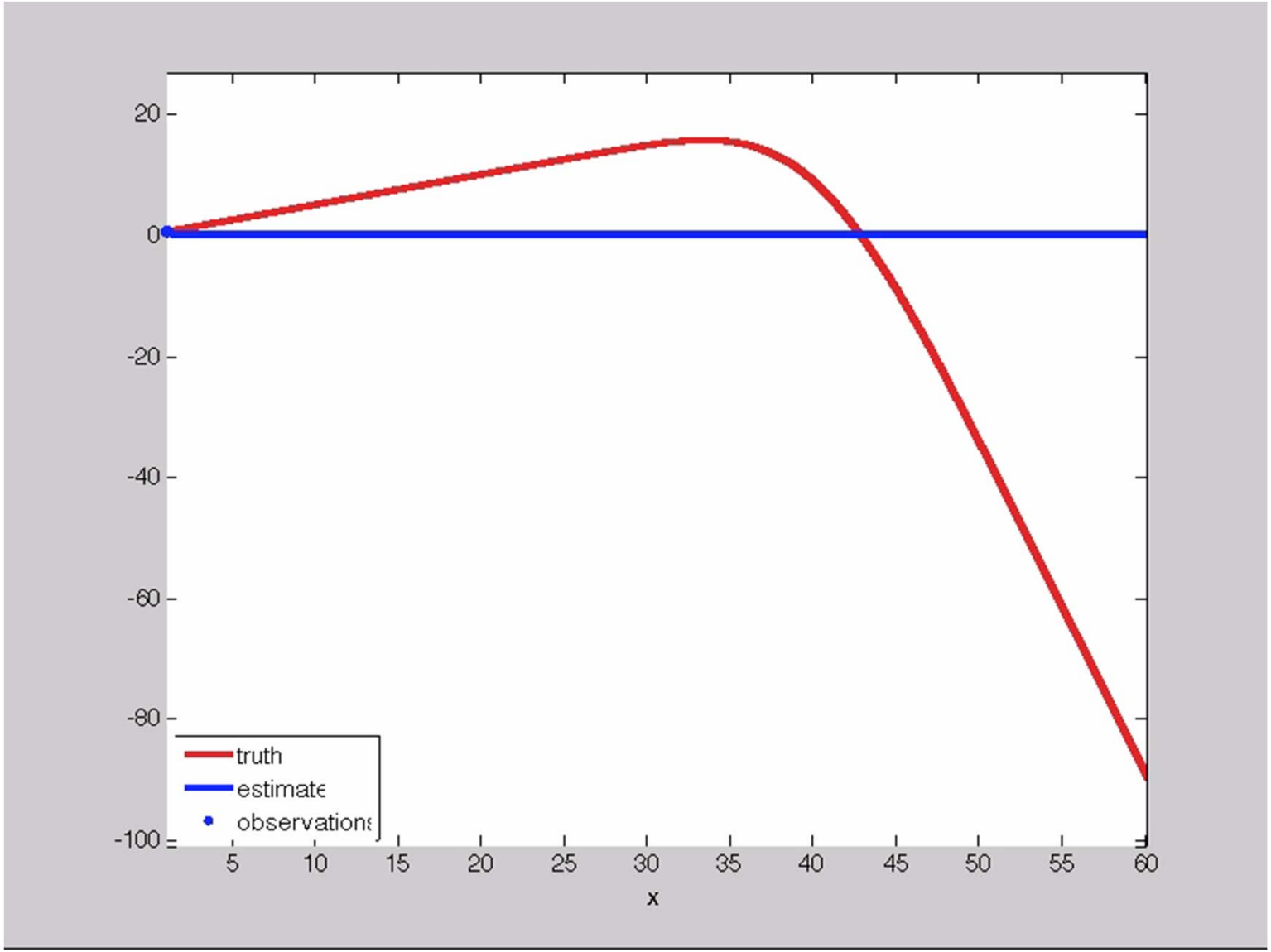
- Dirichlet process mixtures of generalized linear regression models (Hannah, Blei and WBP, 2011).
  - » Very general, but very slow.
  - » Cannot be implemented recursively.



- Dirichlet clouds using radial basis functions (Jamshidi and WBP, 2012)
  - » Local parametric approximations.
  - » Produces variable order approximation that adapts to the function.



Nonparametric methods



Nonparametric methods

- Discussion:
  - » Local smoothing methods struggle with curse of dimensionality. In high dimensions, no two data points are ever "close."
  - » Nonparametric representations are very high-dimensional, which makes it hard to "store" a model.

Nonparametric methods

- More advanced methods
  - » Deep neural networks
    - Before, we described neural networks as a nonlinear parametric model.
    - Deep neural networks, which have the property of being able to approximate any function, are classified as nonparametric.
    - These have proven to be very powerful on image processing and voice recognition, but not in stochastic optimization.
  - » Support vector machines (classification)
  - » Support vector regression (continuous)
    - We have had surprising but limited success with SVM, but considerably more empirical research is needed.