15-780: Grad AI
Lecture 21: Bayesian learning, (PO)MDPs

Geoff Gordon (this lecture)
Tuomas Sandholm
TAs Erik Zawadzki, Abe Othman

Admin

- Reminder: project milestone reports due today
- Reminder: HW5 out

Review: numerical integration

- Parallel importance sampling
  - allows $ZR(x)$ instead of $R(x)$
  - biased, but asymptotically unbiased
- Sequential sampling (for chains, trees)
- Parallel IS + ***resampling*** for sequential problems = ***particle filter***

Review: MCMC

- Metropolis-Hastings: randomized search procedure for high R(x)
- Leads to ***stationary distribution*** = R(x)
- Repeatedly tweak current x to get x'
  - If $R(x') \geq R(x)$, move to x'
  - If $R(x') << R(x)$, stay at x
- Requires good one-step proposal $Q(x' \mid x)$ to get acceptable acceptance rate and mixing rate

Review: Gibbs

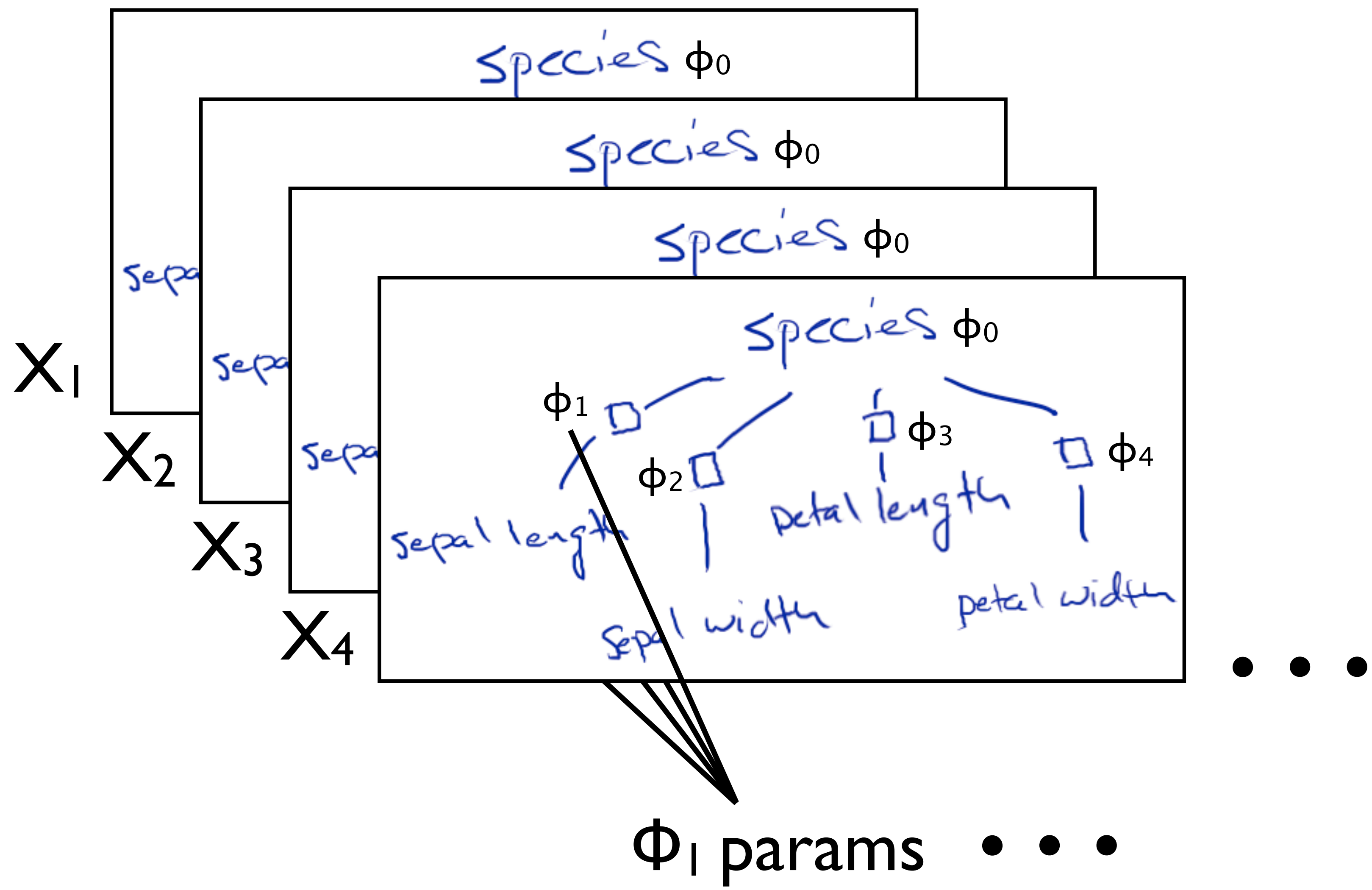
- Special case of MH for $\mathbf{x}$ divided into blocks
- Proposal Q:
  - pick a block i uniformly (or round robin, or any other schedule)
  - sample $\mathbf{x}_{B(i)} \sim P(\mathbf{x}_{B(i)} \mid \mathbf{x}_{\neg B(i)})$
- Acceptance rate = 100%

Review: Learning

- $P(M \mid \mathbf{X}) = P(\mathbf{X} \mid M)\, P(M) \,/\, P(\mathbf{X})$
- $P(M \mid \mathbf{X}, \mathbf{Y}) = P(\mathbf{Y} \mid \mathbf{X}, M)\, P(\mathbf{X} \mid M) \,/\, P(\mathbf{Y} \mid M)$
- Example: framlings
- Version space algorithm: when prior is uniform and likelihood is 0 or 1

Bayesian Learning

Recall iris example



- $\mathcal{H}$ = factor graphs of given structure
- Need to specify entries of ϕs

Factors

ϕ_0

setosa	p
versicolor	q
virginica	$1-p-q$

$\phi_1 - \phi_4$

	lo	m	hi
set.	p_i	q_i	$1-p_i-q_i$
vers.	r_i	s_i	$1-r_i-s_i$
vir.	u_i	v_i	$1-u_i-v_i$

Continuous factors

ϕ_1

	lo	m	hi
set.	p_l	q_l	$l-p_l-q_l$
vers.	r_l	s_l	$l-r_l-s_l$
vir.	u_l	v_l	$l-u_l-v_l$

$$\Phi_1(\ell, s) = \exp(-(\ell - \ell_s)^2/2\sigma^2)$$

parameters ℓ_{set}, ℓ_{vers}, ℓ_{vir}; constant σ^2

Discretized petal length

Continuous petal length

Simpler example

H	p
T	$1-p$

Coin toss

Parametric model class

- $\mathcal{H}$ is a ***parametric*** model class: each H in $\mathcal{H}$ corresponds to a vector of parameters $\theta = (p)$ or $\theta = (p, q, p_1, q_1, r_1, s_1, \ldots)$
- H_θ: $\mathbf{X} \sim P(\mathbf{X} \mid \theta)$ (or, $Y \sim P(Y \mid \mathbf{X}, \theta)$)
- Contrast to ***discrete*** $\mathcal{H}$, as in version space
- Could also have ***mixed*** $\mathcal{H}$: discrete choice among parametric (sub)classes

Continuous prior

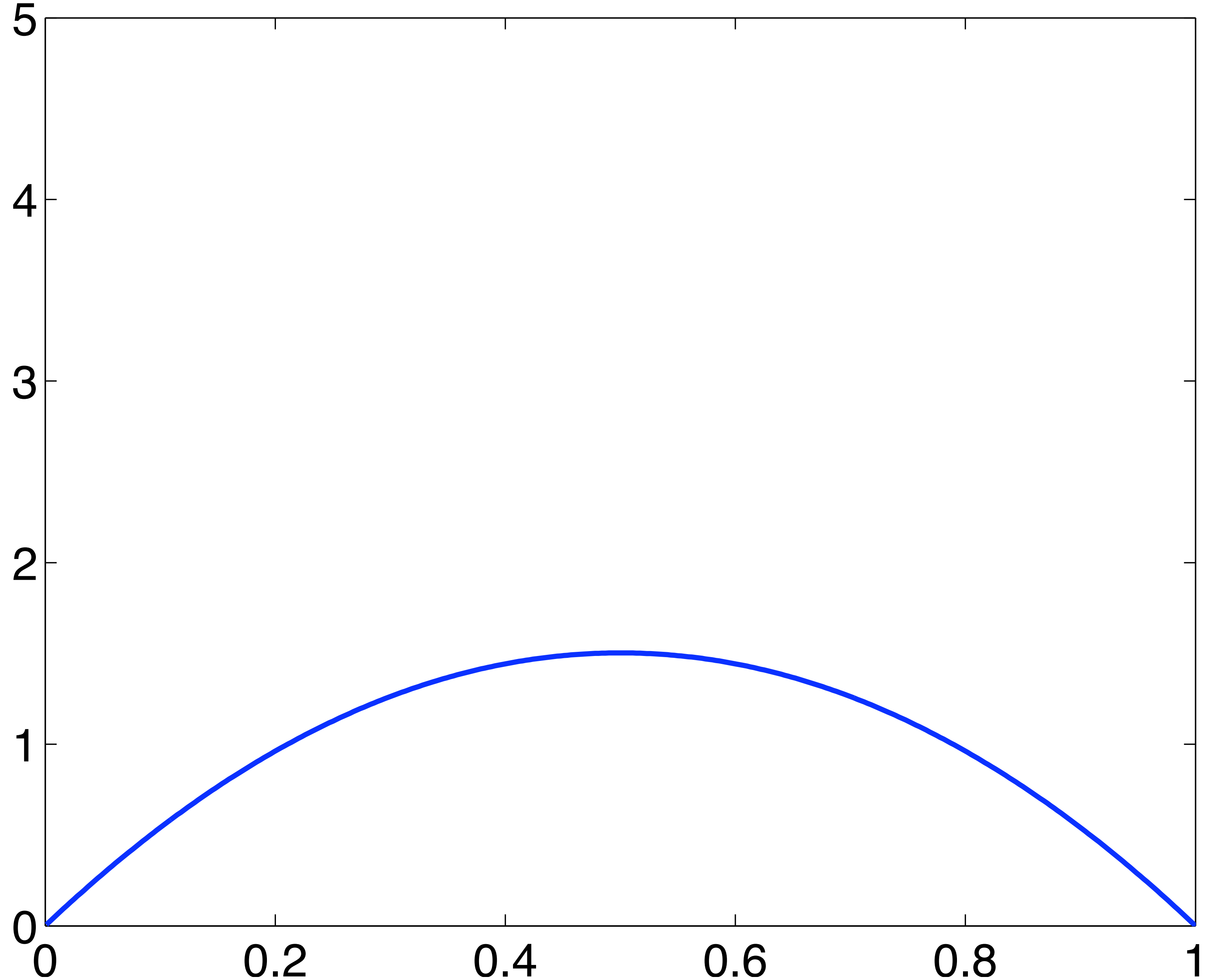
- E.g., for coin toss, p ~ Beta(a, b):

$$P(p \mid a, b) = \frac{1}{B(a, b)} p^{a-1} (1-p)^{b-1}$$

- Specifying, e.g., a = 2, b = 2:

$$P(p) = 6p(1-p)$$

Prior for *p*



Coin toss, cont'd

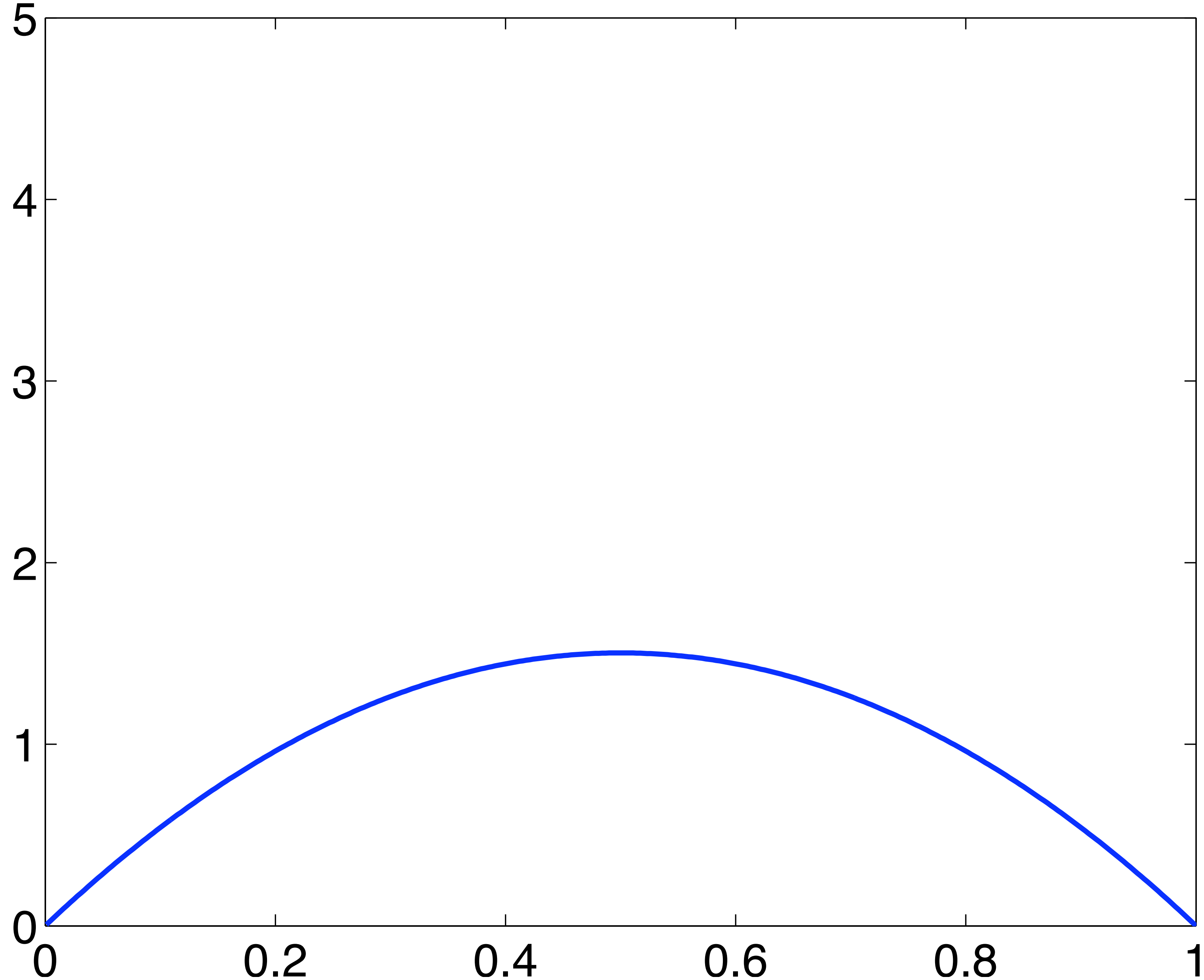
- Joint dist'n of parameter p and data x_i:

$$\begin{aligned} P(p, \mathbf{x}) &= P(p) \prod_i P(x_i \mid p) \\ &= 6p(1-p) \prod_i p^{x_i} (1-p)^{1-x_i} \end{aligned}$$

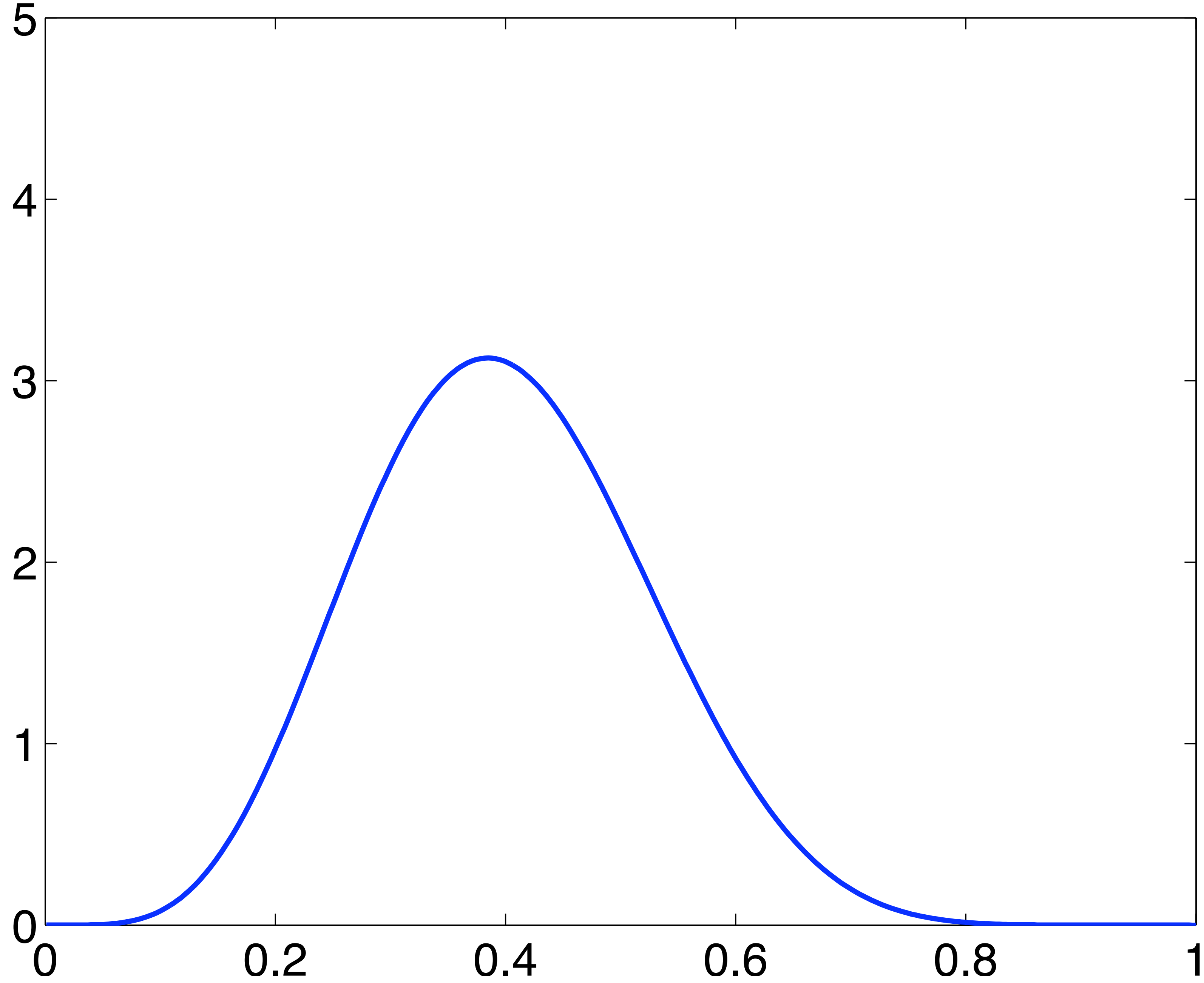
Coin flip posterior

$$
\begin{aligned}
P(p \mid \mathbf{x}) &= P(p) \prod_i P(x_i \mid p) / P(\mathbf{x}) \\
&= \frac{1}{Z} p(1-p) \prod_i p^{x_i} (1-p)^{1-x_i} \\
&= \frac{1}{Z} p^{1+\sum_i x_i} (1-p)^{1+\sum_i (1-x_i)} \\
&= \mathrm{Beta}(2 + \textstyle\sum_i x_i,\ 2 + \sum_i (1-x_i))
\end{aligned}
$$

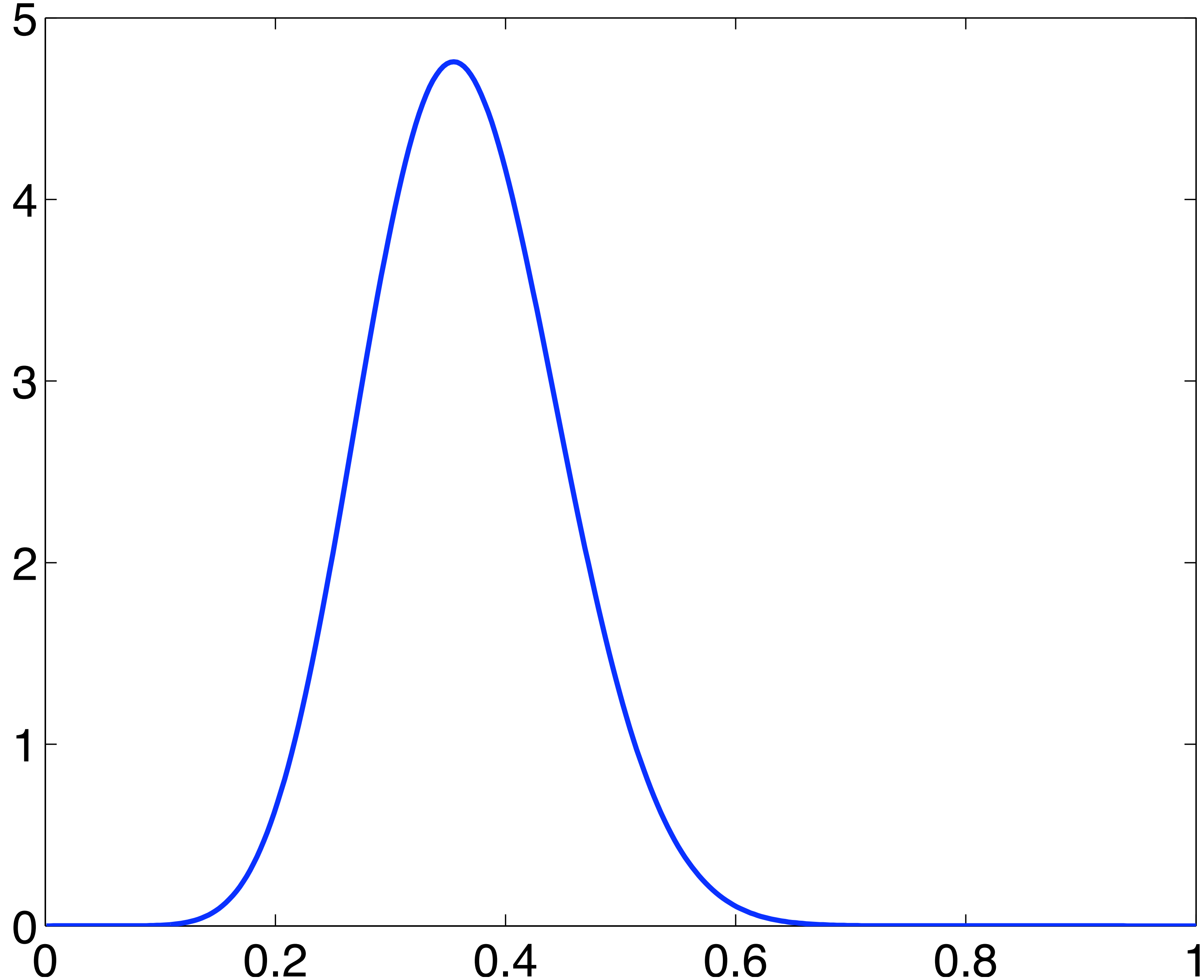
Prior for p



Posterior after 4 H, 7 T



Posterior after 10 H, 19 T



Predictive distribution

- Posterior is nice, but doesn't tell us directly what we need to know
- We care more about $P(x_{N+1} \mid x_1, \ldots, x_N)$
- By law of total probability, conditional independence:

$$P(x_{N+1} \mid \mathbf{D}) = \int P(x_{N+1}, \theta \mid \mathbf{D}) d\theta$$
$$= \int P(x_{N+1} \mid \theta) P(\theta \mid \mathbf{D}) d\theta$$

Coin flip example

- After 10 H, 19 T: $p \sim \text{Beta}(12, 21)$
- $E(x_{N+1} \mid p) = p$
- $E(x_{N+1} \mid \theta) = E(p \mid \theta) = a/(a+b) = 12/33$
- So, predict 36.4% chance of H on next flip

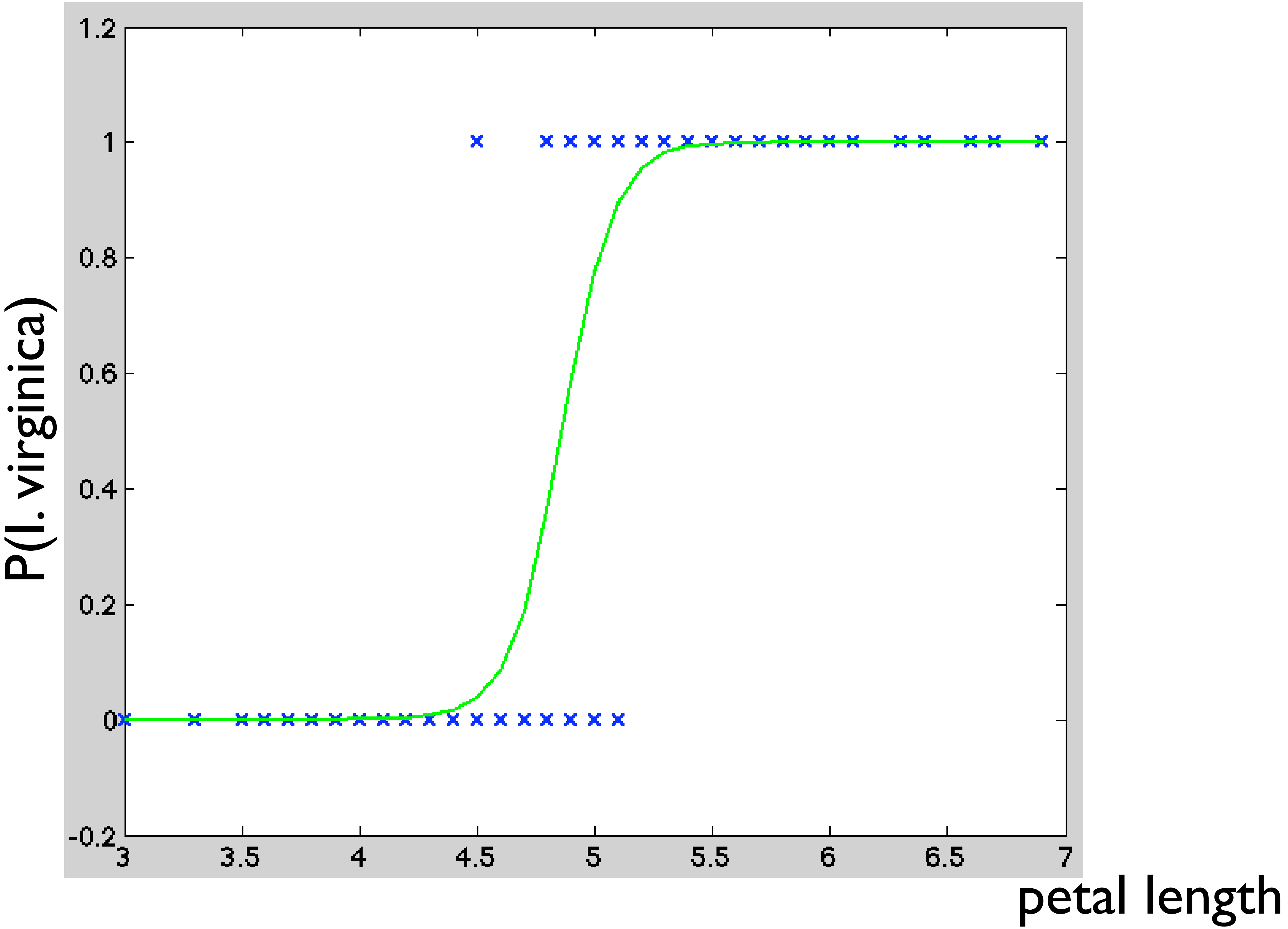
Approximate Bayes

Approximate Bayes

- Coin flip example was easy
- In general, computing posterior (or predictive distribution) may be hard
- Solution: use the approximate integration techniques we've studied!

Bayes as numerical integration

- Parameters θ, data $\mathbf{D}$
- $P(\theta \mid \mathbf{D}) = P(\mathbf{D} \mid \theta)\, P(\theta) / P(\mathbf{D})$
- Usually, $P(\theta)$ is simple; so is $Z\, P(\mathbf{D} \mid \theta)$
- So, $P(\theta \mid \mathbf{D}) \propto Z\, P(\mathbf{D} \mid \theta)\, P(\theta)$
- Perfect for MH



$$P(y \mid x) = \sigma(ax + b)$$

$$\sigma(z) = 1/(1 + exp(-z))$$

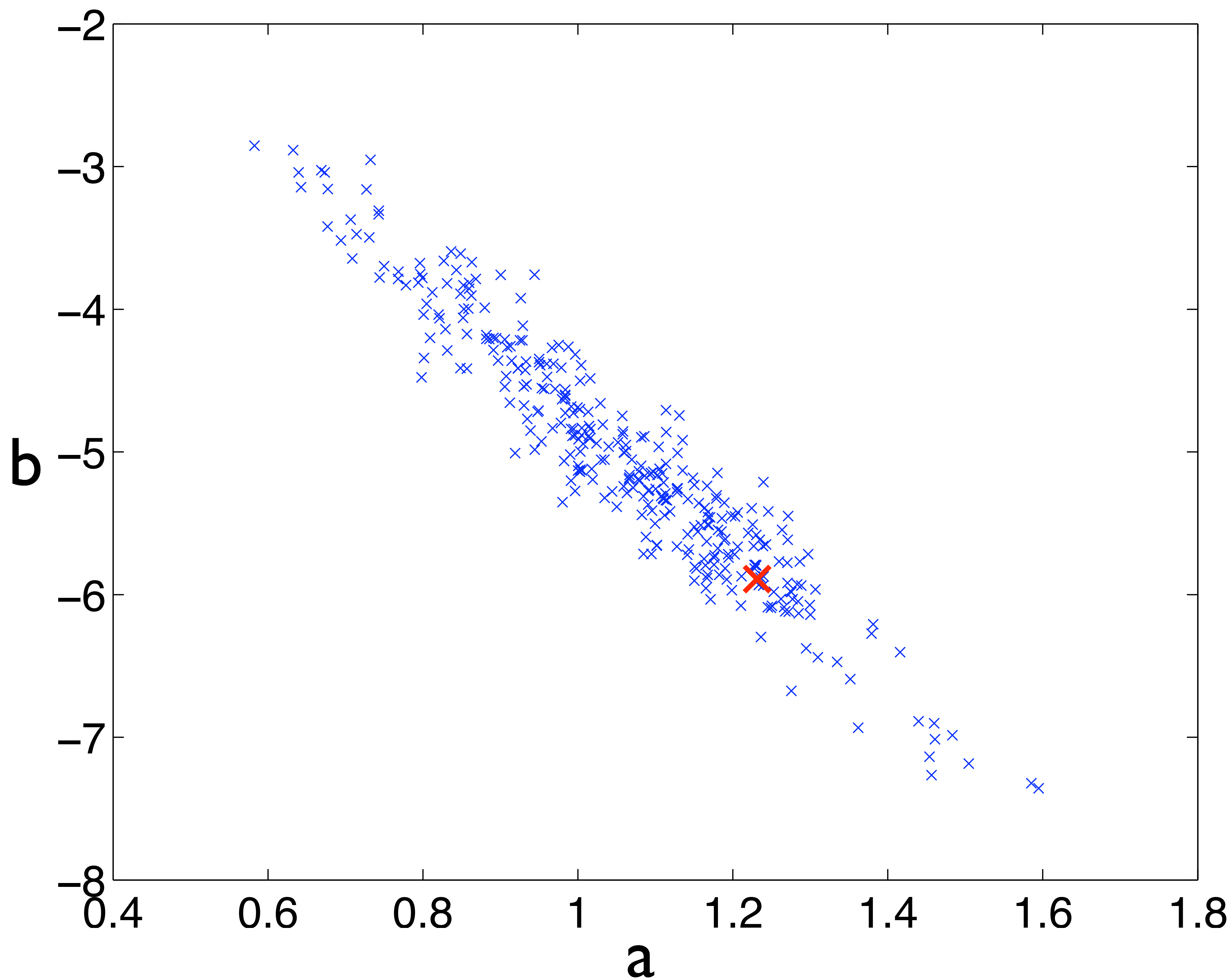
Posterior

$$P(a, b \mid x_i, y_i) =$$

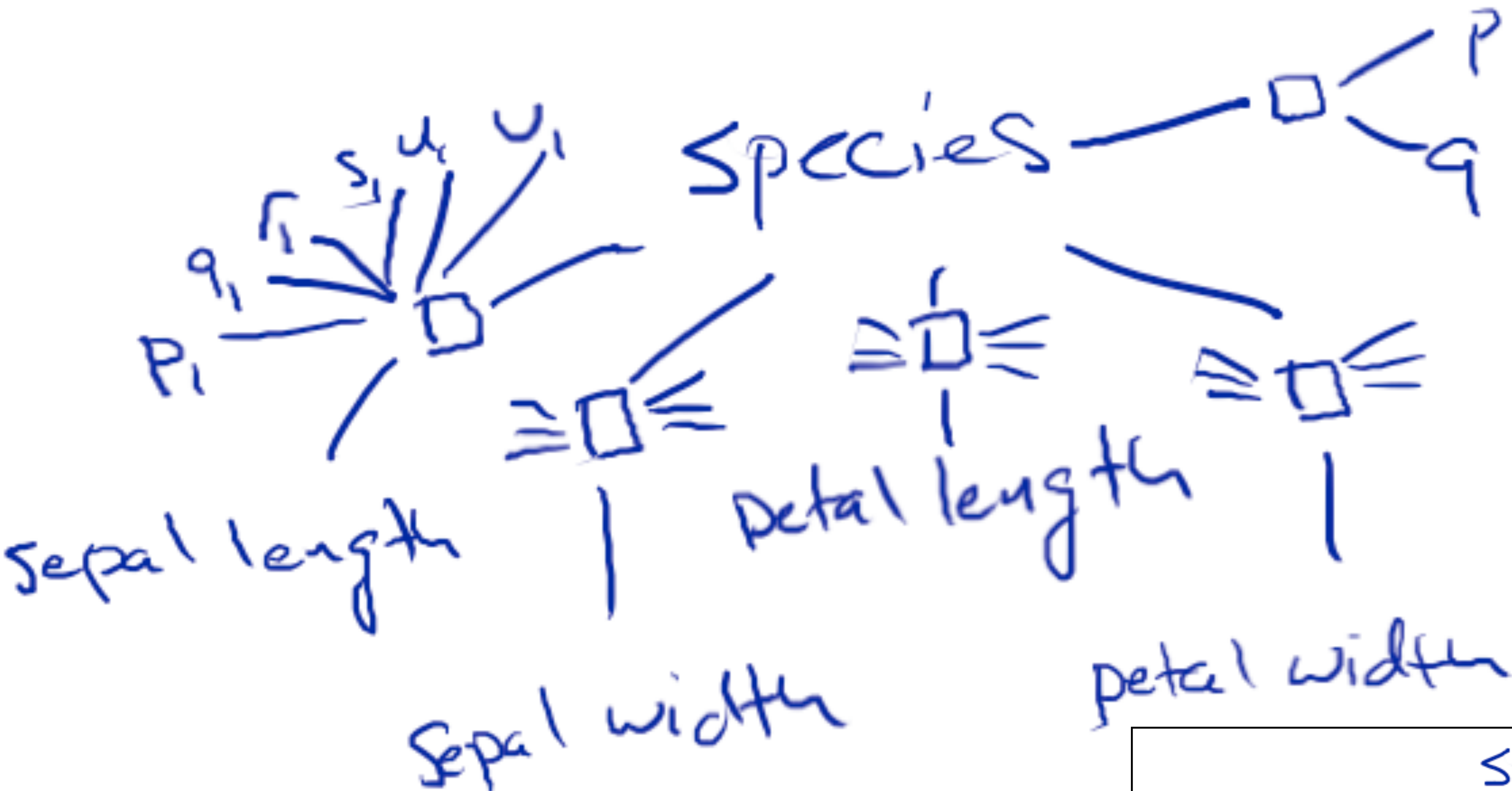
$$ZP(a, b) \prod_i \sigma(ax_i + b)^{y_i} \sigma(-ax_i - b)^{1-y_i}$$

$$P(a, b) = N(0, I)$$

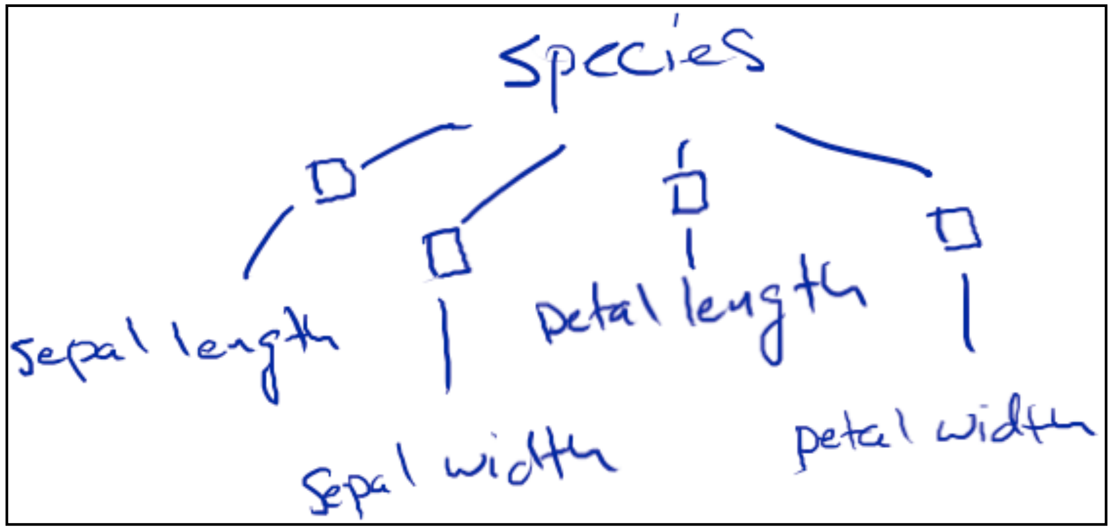
Sample from posterior



Expanded factor graph



original factor graph:



Cheaper approximations

Getting cheaper

- Maximum a posteriori (MAP)
- Maximum likelihood (MLE)
- Conditional MLE / MAP

- Instead of true posterior, just use single most probable hypothesis

MAP

$$\arg\max_{\theta} P(D \mid \theta) P(\theta)$$

- Summarize entire posterior density using the maximum

MLE

$$\arg\max_{\theta} P(D \mid \theta)$$

- Like MAP, but ignore prior term

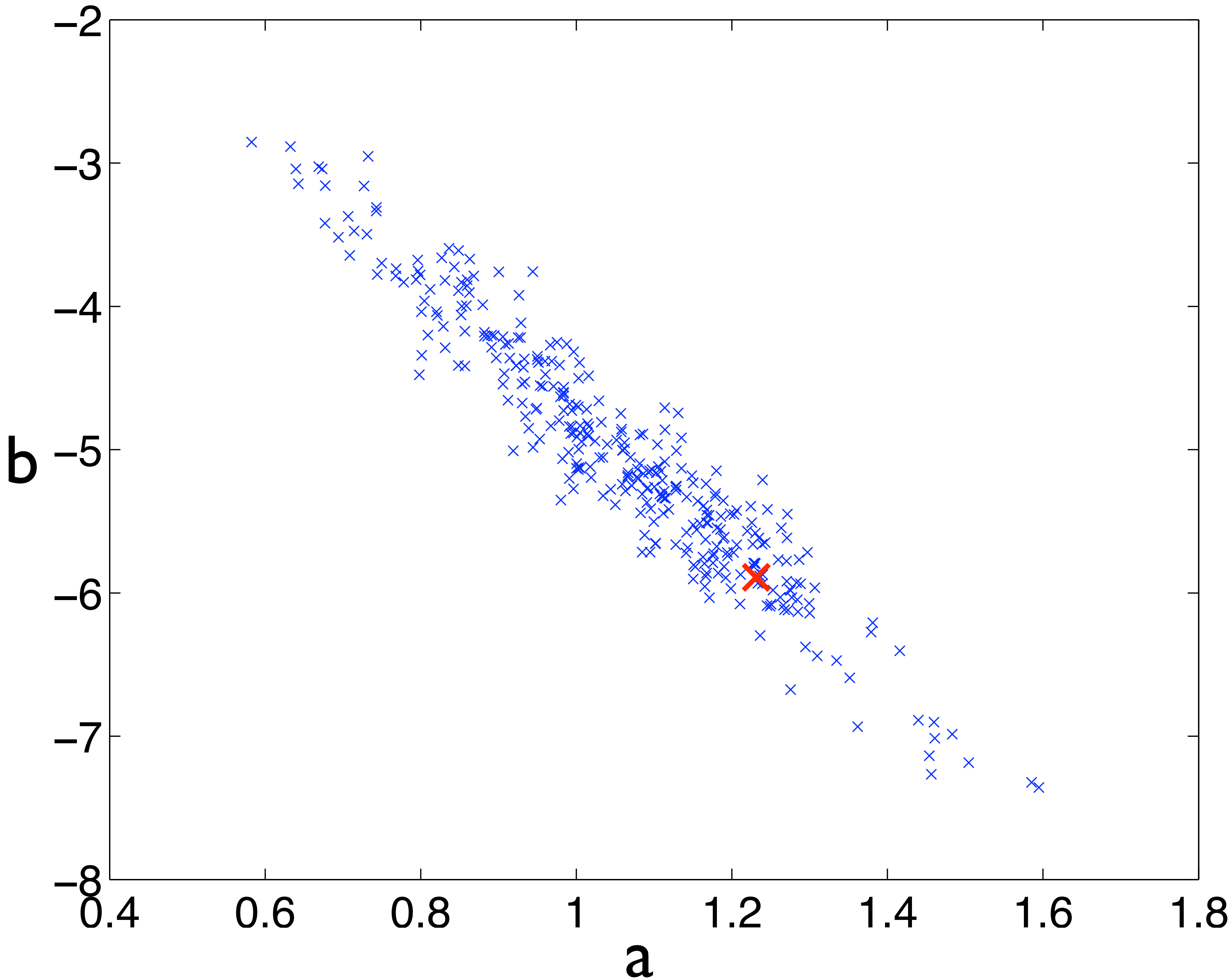
Conditional MLE, MAP

$$\arg\max_\theta P(\mathbf{y} \mid \mathbf{x}, \theta)$$

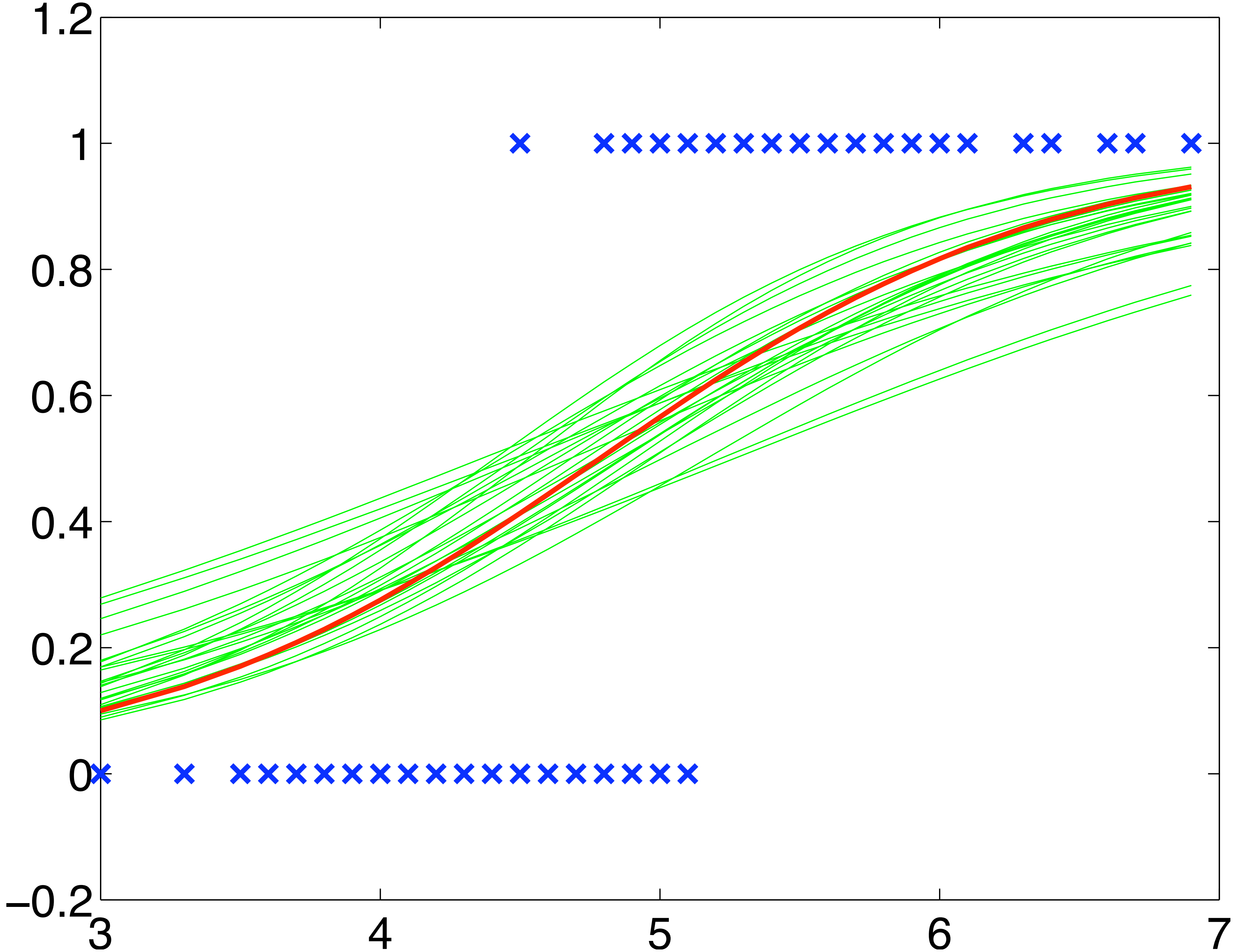
$$\arg\max_\theta P(\mathbf{y} \mid \mathbf{x}, \theta) P(\theta)$$

- Split D = (**x**, **y**)
- Condition on **x**, try to explain only **y**

Iris example: MAP vs. posterior



Irises: MAP vs. posterior

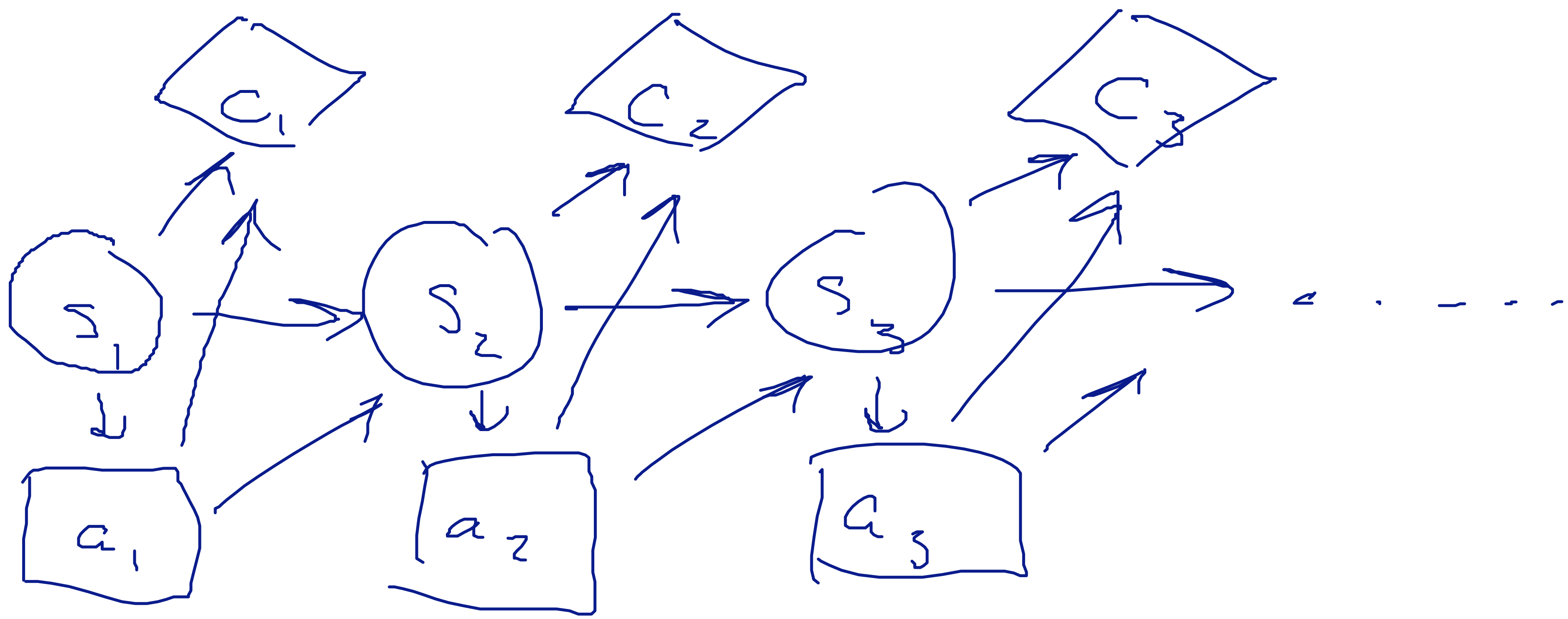


Too certain

- This behavior of MAP (or MLE) is typical: we are too sure of ourselves
- But, often gets better with more data
- Theorem: MAP and MLE are consistent estimates of true θ, if "data per parameter" $\rightarrow \infty$

Sequential Decisions

Markov decision process: influence diagram

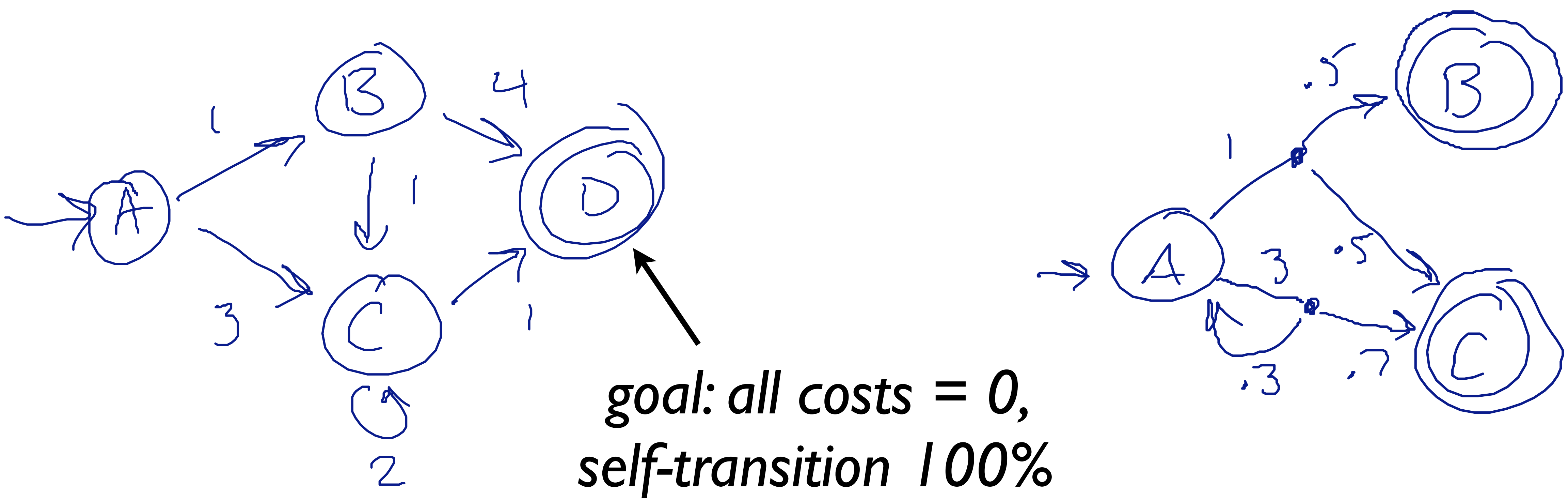


- States, actions, costs $C(s,a) \in [C_{min}, C_{max}]$, transitions $T(s' \mid s, a)$, initial state s_1

Influence diagrams

- Like a Bayes net, except:
  - diamond nodes are costs/rewards
    - must have no children
  - square nodes are decisions
    - we pick the CPTs (before seeing anything)
    - minimize expected cost
- Circles are ordinary r.v.s as before

Markov decision process: state space diagram



- States, actions, costs $C(s,a) \in [C_{min}, C_{max}]$, transitions $T(s' \mid s, a)$, initial state s_I

Choosing actions

- Execution trace: $\tau = (s_1, a_1, c_1, s_2, a_2, c_2, \ldots)$
  - $c_1 = C(s_1, a_1)$, $c_2 = C(s_2, a_2)$, etc.
  - $s_2 \sim T(s \mid s_1, a_1)$, $s_3 \sim T(s \mid s_2, a_2)$, etc.
- Policy $\pi: S \to A$
  - or randomized, $\pi(a \mid s)$
- Trace from π: $a_1 \sim \pi(a \mid s_1)$, etc.
  - τ is then an r.v. with known distribution
  - we'll write $\tau \sim \pi$ (rest of MDP implicit)

Choosing ***good*** actions

- Value of a policy:

discount factor in (0,1)

$$J^\pi = \frac{1-\gamma}{\gamma}\,\mathbb{E}\left[\sum_t \gamma^t c_t \;\middle|\; \tau \sim \pi\right]$$

- Objective:

$$J^* = \min_\pi J^\pi$$

$$\pi^* \in \arg\min_\pi J^\pi$$

Why a discount factor?

Why a discount factor?

- A1: to make the sums finite

Why a discount factor?

- A1: to make the sums finite
- A2: interest rate $1/\gamma - 1$ per period

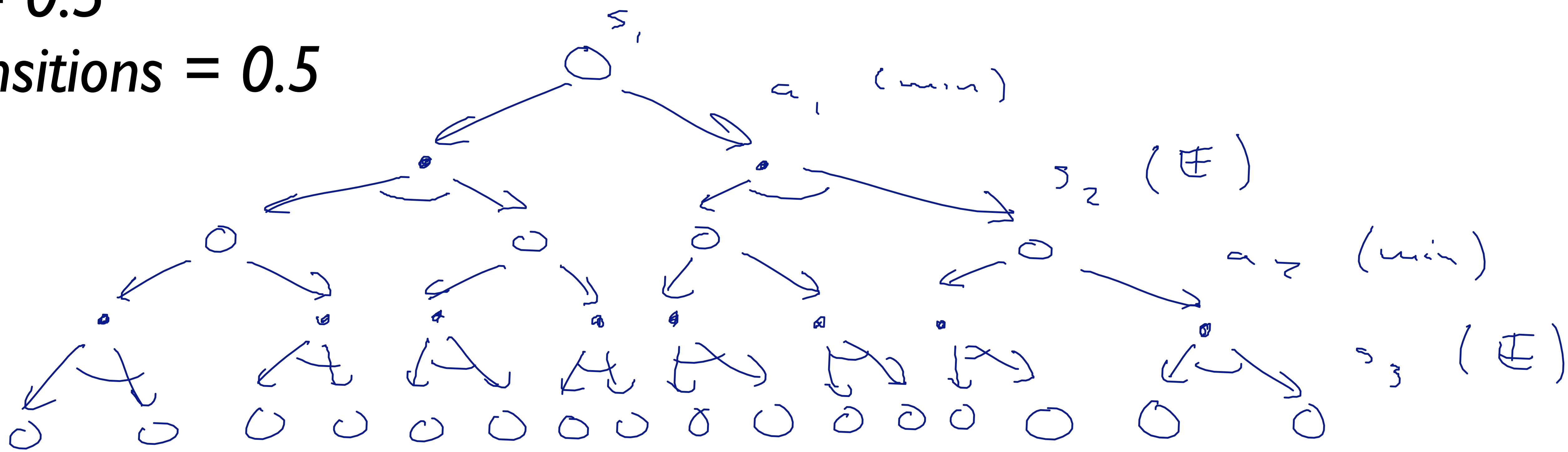
Why a discount factor?

- A1: to make the sums finite
- A2: interest rate $1/\gamma - 1$ per period
- A3: model mismatch
  - probability $(1-\gamma)$ that something unexpected happens on each step and my plan goes out the window

Tree search

γ = *0.5*

transitions = 0.5



- Root node = current state
- Alternating levels: action and outcome
  - min and expectation
- Build out tree until goal or until γ^t small enough

Interpreting the result

- Number at each ○ node: optimal cost if starting from state s instead of s_1
  - call this $J^*(s)$—so, $J^* = J^*(s_1)$
  - ***state-value*** function
- Number at each · node: optimal cost if starting from parent's s, choosing incoming a
  - call this $Q^*(s,a)$
  - ***action-value*** function
- Similarly, $J^\pi(s)$ and $Q^\pi(s, a)$

The update equations

- For $\cdot$ node

$$Q^*(s,a) \quad = \quad (1-\gamma)C(s,a) + \gamma\mathbb{E}[J^*(s') \mid s' \sim T(\cdot \mid s,a)]$$

- For $\circ$ node

$$J^*(s) \quad = \quad \min_a \; Q^*(s,a)$$

(1–γ) × immediate cost + γ × future cost

Updates for a fixed policy

- For $\cdot$ node

$$Q^\pi(s,a) \quad = \quad (1-\gamma)C(s,a) + \gamma\mathbb{E}[J^\pi(s') \mid s' \sim T(\cdot \mid s,a)]$$

- For $\circ$ node

$$J^\pi(s) \quad = \quad \mathbb{E}[Q^\pi(s,a) \mid a \sim \pi(\cdot \mid s)]$$

(1–γ) × immediate cost + γ × future cost

Speeding it up

- Can't do DPLL-style pruning: outcome node depends on ***all*** children
- Can do some pruning: e.g., low-probability outcomes when branch is already clearly bad
- Or, use scenarios: subsample outcomes at each expectation node

Receding-horizon planning

- Stop building tree at 2k levels, evaluate leaf nodes with ***heuristic*** h(s)
  - or at 2k–1 levels, evaluate with h(s, a)
- Minimal guarantees, but often works well in practice
- Can also use adaptive horizon
- Just as in deterministic search, a good heuristic is essential!

Good heuristic

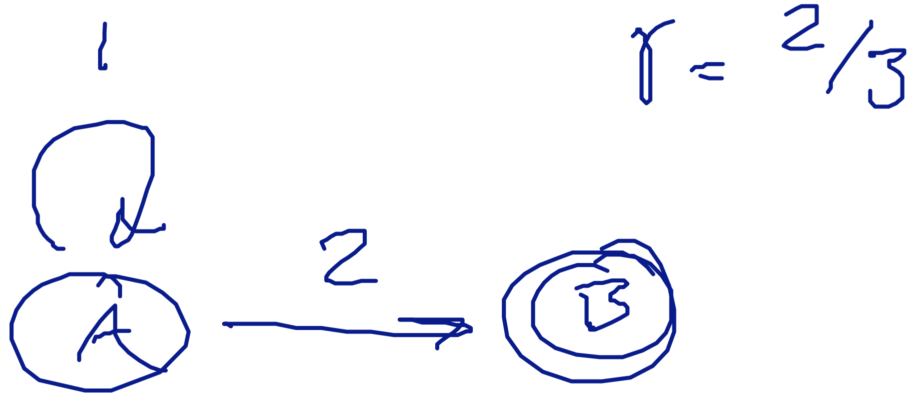
- Good heuristic: $h(s) \approx J^*(s)$ or $h(s, a) \approx Q^*(s,a)$
- If we have $h(s) = J^*(s)$, only need to build first two levels of tree (action and outcome) to choose optimal action at s_1
- With $h(s, a) = Q^*(s,a)$, only need to build first (action) level
- Often try to use $h \approx J^\pi$ or Q^π for some good π

Dynamic programming

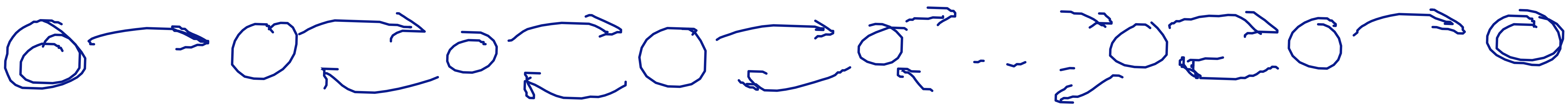
- If there are a small number of states and actions, makes sense to ***memoize*** tree search
  - compute an entire level of the tree at a time, working from bottom up
  - store only $S \times A$ numbers r.t. b^d

DP example: should I stay or should I go?

<u>Q(A, stay)   Q(A, go)   J(A)</u>

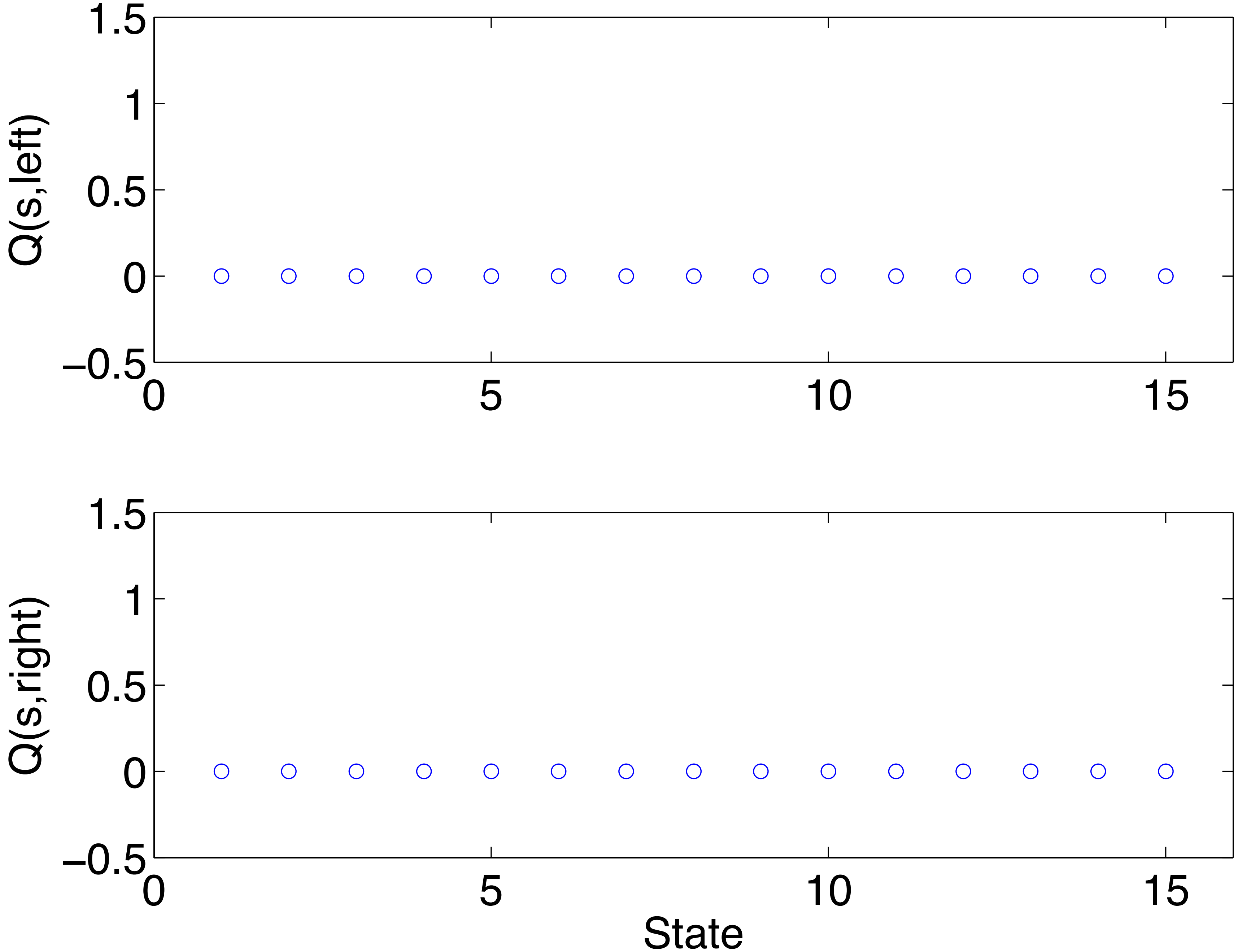


DP example 2

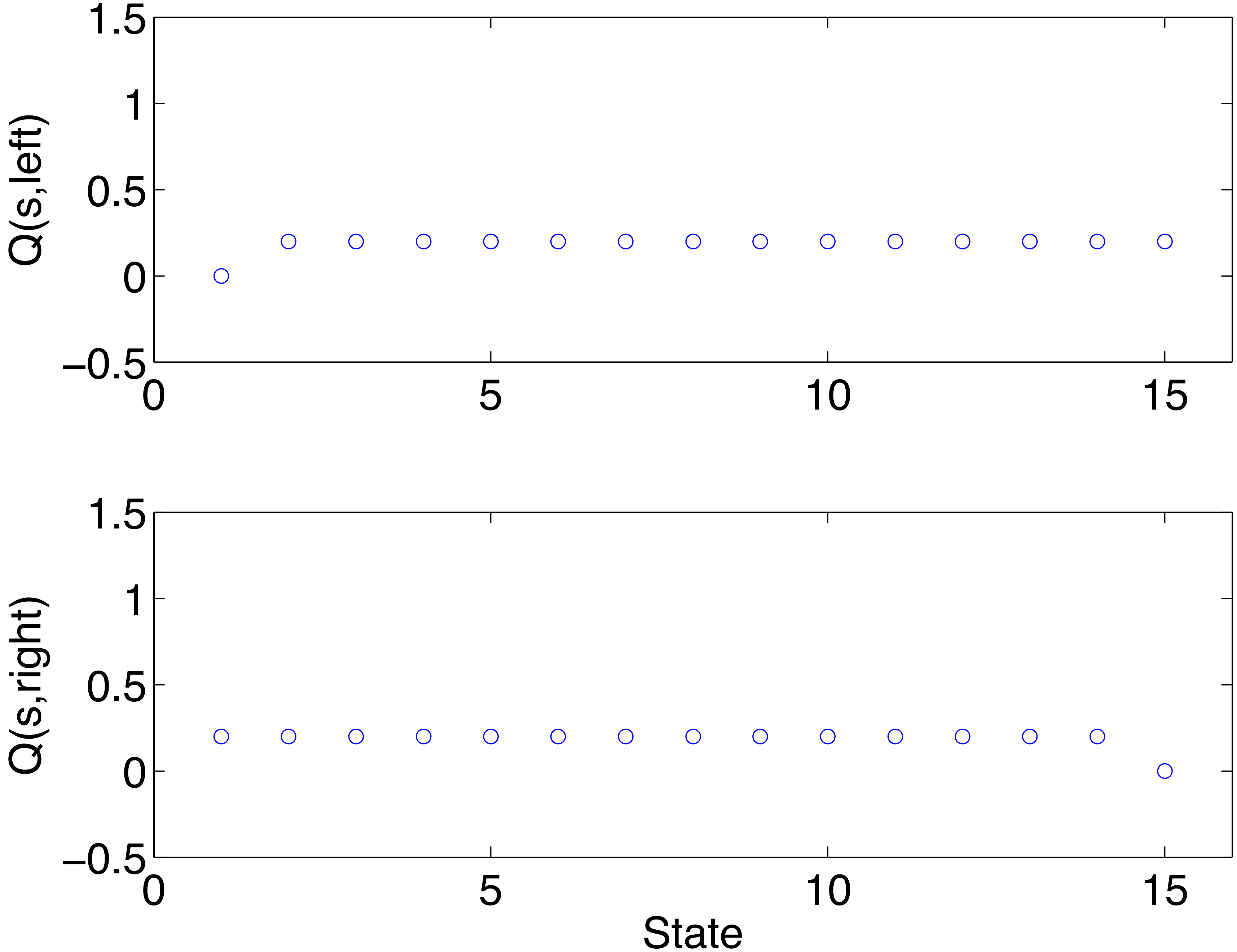


- each step costs 1
- discount 0.8

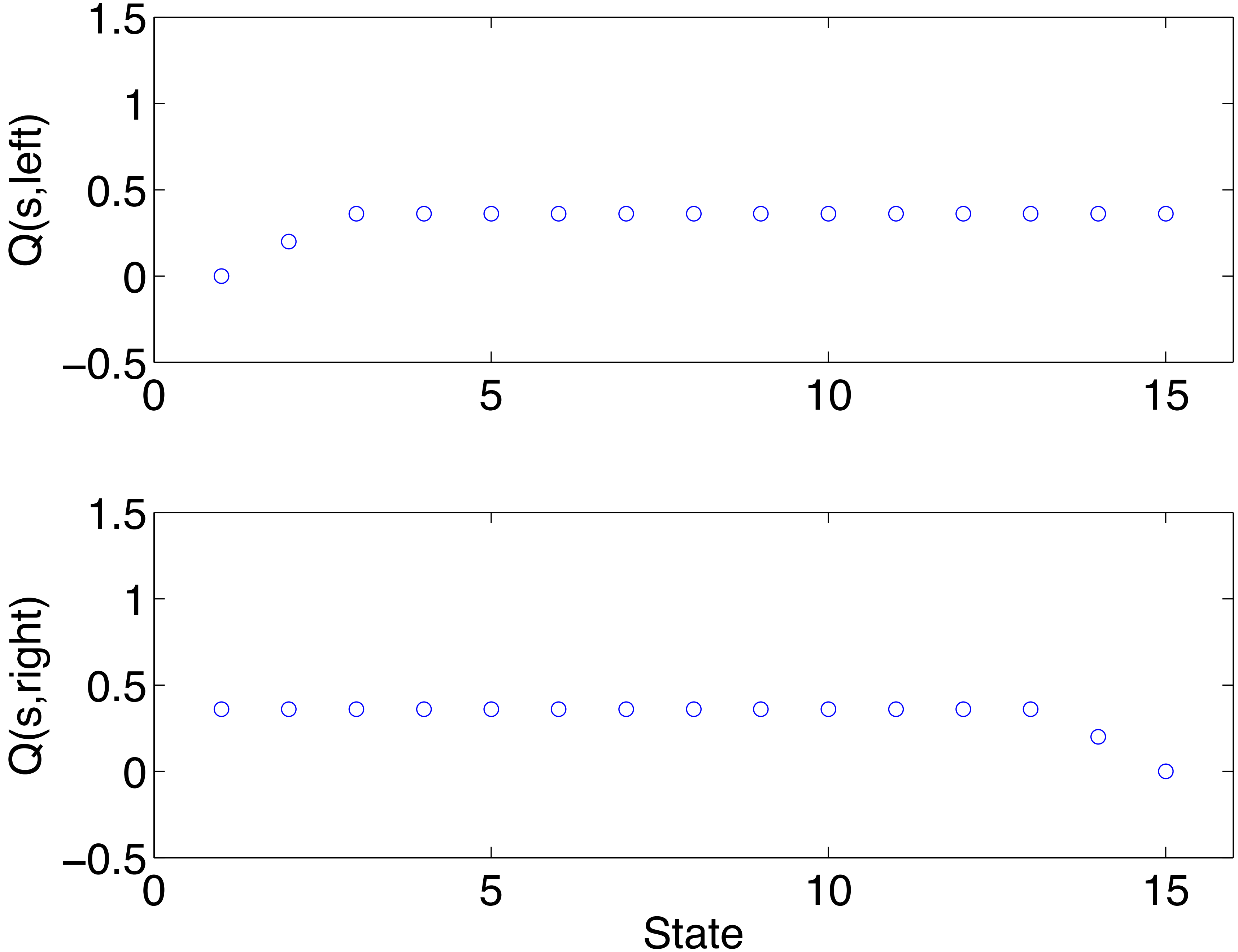
DP example 2



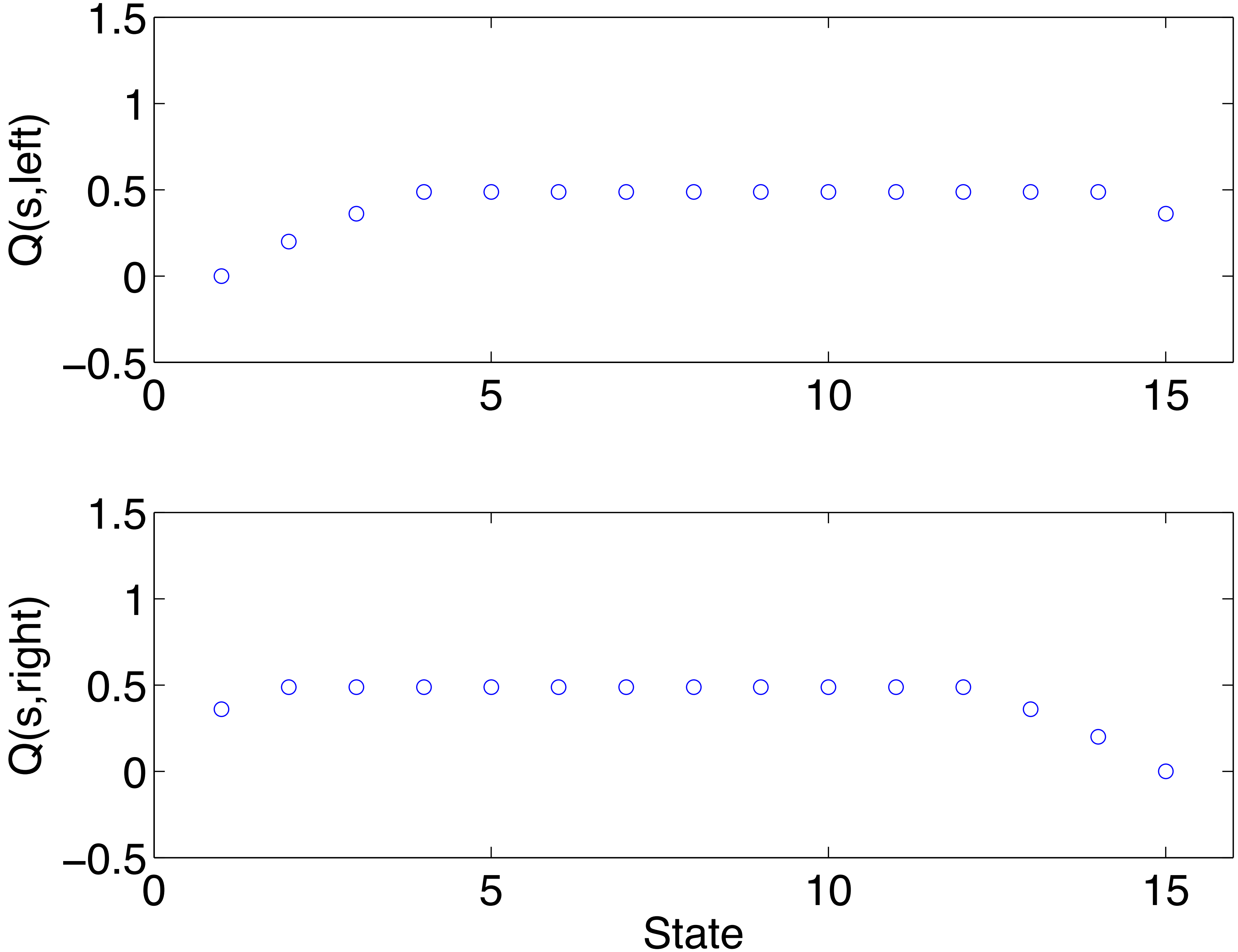
DP example 2



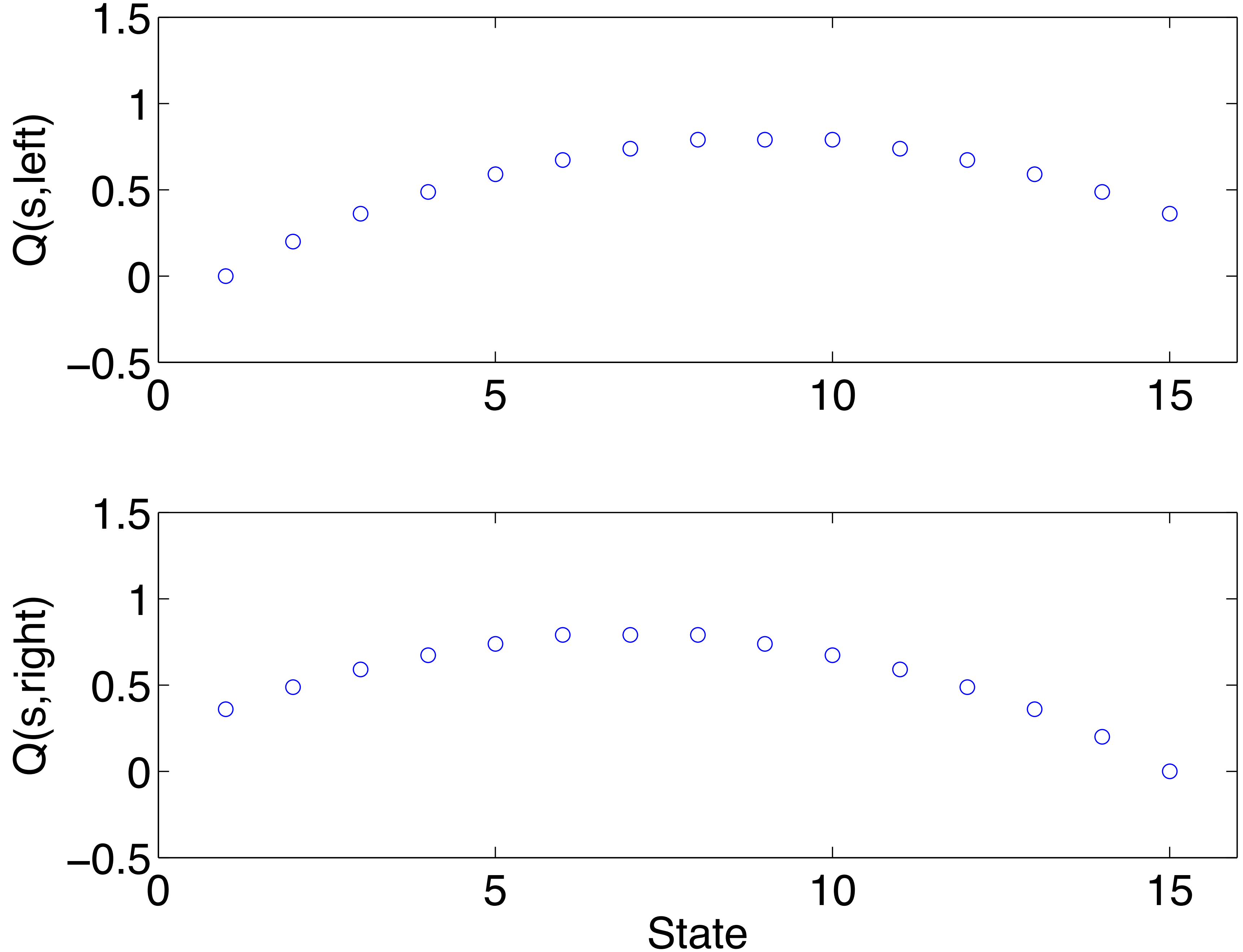
DP example 2



DP example 2



DP example 2



Discussion

- Terminology: backup, sweep, value iteration
- VI makes max error converge linearly to 0 at rate γ per sweep
- Works well for up to 1,000,000s of states, as long as we can evaluate min and expectation efficiently (e.g., few actions, sparse outcomes)
  - tricks: replace J(s) by backed up value immediately (not at end of sweep); schedule backups by ***priority*** = estimate of how much J(s) will change

Curse of dimensionality

- Sadly, 1,000,000s of states don't necessarily get us very far
- E.g., 10 state variables, each with 10 values: 10^{10} states

Alternate algorithms for "small" systems—policy evaluation

$$Q^\pi(s,a) = (1-\gamma)C(s,a) + \gamma\mathbb{E}[J^\pi(s') \mid s' \sim T(\cdot \mid s,a)]$$
$$J^\pi(s) = \mathbb{E}[Q^\pi(s,a) \mid a \sim \pi(\cdot \mid s)]$$

- Linear equations: so, Gaussian elimination, biconjugate gradient, Gauss-Seidel iteration, …
  - DP is essentially the Jacobi iterative method for matrix inversion
- SARSA: stochastic-gradient-descent-like
  - and related methods: TD(λ), Q-learning

Alternate algorithms for "small" systems—policy optimization

- Policy iteration: alternately
  - use any above method to evaluate current π
  - replace π with ***greedy*** policy: at each state s, $\pi(s) := \arg\max_a Q(s,a)$
- Actor-critic: like policy iteration, but ***interleave*** solving for J^π and updating π
  - e.g., run biconjugate gradient for a few steps
  - warm start: each J^π probably similar to next
- SARSA = AC w/ SARSA critic, ϵ-greedy policy

Alternate algorithms for "small" systems—policy optimization

- (Stochastic) policy gradient
  - pick a parameterized policy class $\pi_\theta(a \mid s)$
  - compute or estimate $g = \nabla_\theta J^\pi(s_1)$
  - $\theta \leftarrow \theta - \eta g$, repeat
- More detail:
  - can estimate g quickly by simulating a few trajectories
  - can also use ***natural*** gradient to get faster convergence

Alternate algorithms for "small" systems—policy optimization

- Linear programming
  - analogy: use an LP to compute min(3, 6, 5)
  - note min v. max

$$\max\ J \quad \text{s.t.}$$

$$J \le 3$$

$$J \le 6$$

$$J \le 5$$

Linear programming

$$\max \; J(s_1) \;\; \text{s.t.}$$

$$Q(s,a) = (1-\gamma)C(s,a) + \gamma\mathbb{E}[J(s') \mid s' \sim T(\cdot \mid s,a)]$$

$$J(s) \leq Q(s,a) \qquad \forall s,a$$

- Variables J(s) and Q(s,a) for all s, a
- Note: dual of this LP is interesting
  - generalizes single-source shortest paths

Model requirements

- What we have to know about the MDP in order to plan?
  - full model
  - simulation model
  - no model: only the real world

Model requirements

- VI and LP require full model
- PI and actor-critic inherit requirements of policy-evaluation subroutine
- SARSA, policy gradient: OK with simulation model or no model
  - horribly data-inefficient if used directly on real world with no model

A word on performance measurements

- Multiple criteria we might care about:
  - data (from real world)
  - runtime
  - calls to model (under some API)
- Measure convergence rate of:
  - $J(s)$ or $Q(s, a)$
  - $\pi(s)$
  - actual (expected total discounted) cost

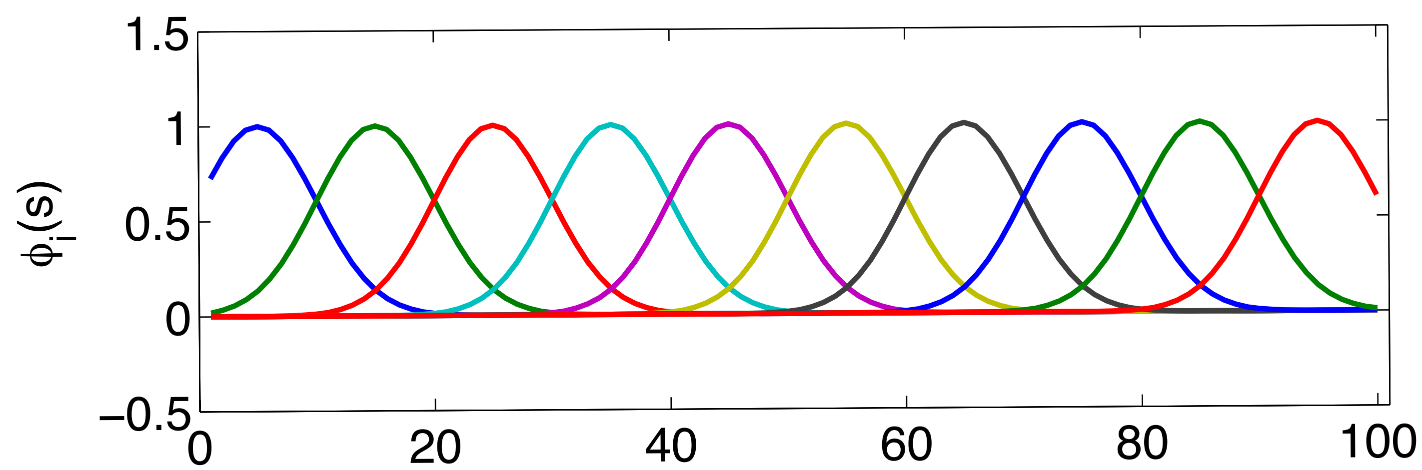
Building a model

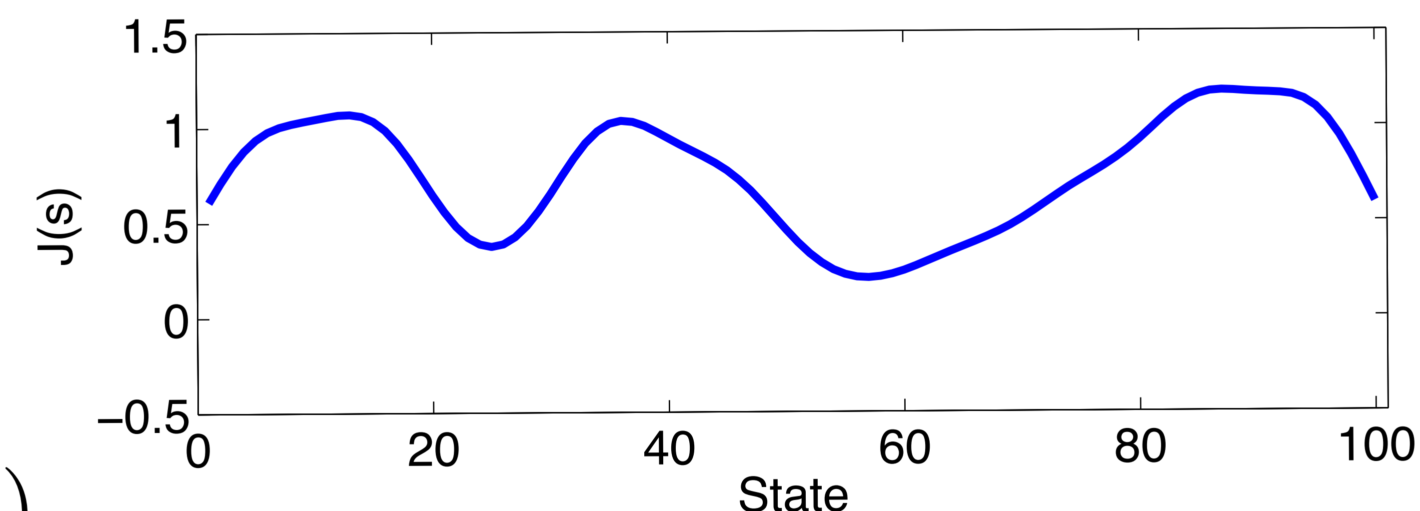
- How to handle lack of model without horrible data inefficiency? Build one!
  - hard inference problem; getting it wrong is bad
- What do we do with posterior over models?
  - just use MAP model ("certainty equivalent")
  - compute posterior over π^*: slow, still wrong
  - even slower: $\max_{\pi} \mathbb{E}(J^{\pi}(s) \mid \text{data, model class})$
    - unless we're doing policy gradient (Ng's helicopter)

Algorithms for large systems

- Policy gradient: no change
- Any value-based method: can't even write down J(s) or Q(s,a)
- So,

$$
\begin{aligned}
J(s) &= \sum_i w_i \phi_i(s) \\
Q(s,a) &= \sum_i w_i \phi_i(s,a)
\end{aligned}
$$





Algorithms for large systems

- Evaluation: SARSA, LSTD
- Optimization:
  - policy iteration or actor-critic
    - e.g., LSTD → LSPI
  - approximate LP
  - value iteration: only special cases, e.g., finite-element grid

Least-squares temporal differences (LSTD)

$$\begin{aligned} Q^\pi(s,a) &= (1-\gamma)C(s,a) + \gamma\mathbb{E}[J^\pi(s') \mid s' \sim T(\cdot \mid s,a)] \\ J^\pi(s) &= \mathbb{E}[Q^\pi(s,a) \mid a \sim \pi(\cdot \mid s)] \end{aligned}$$

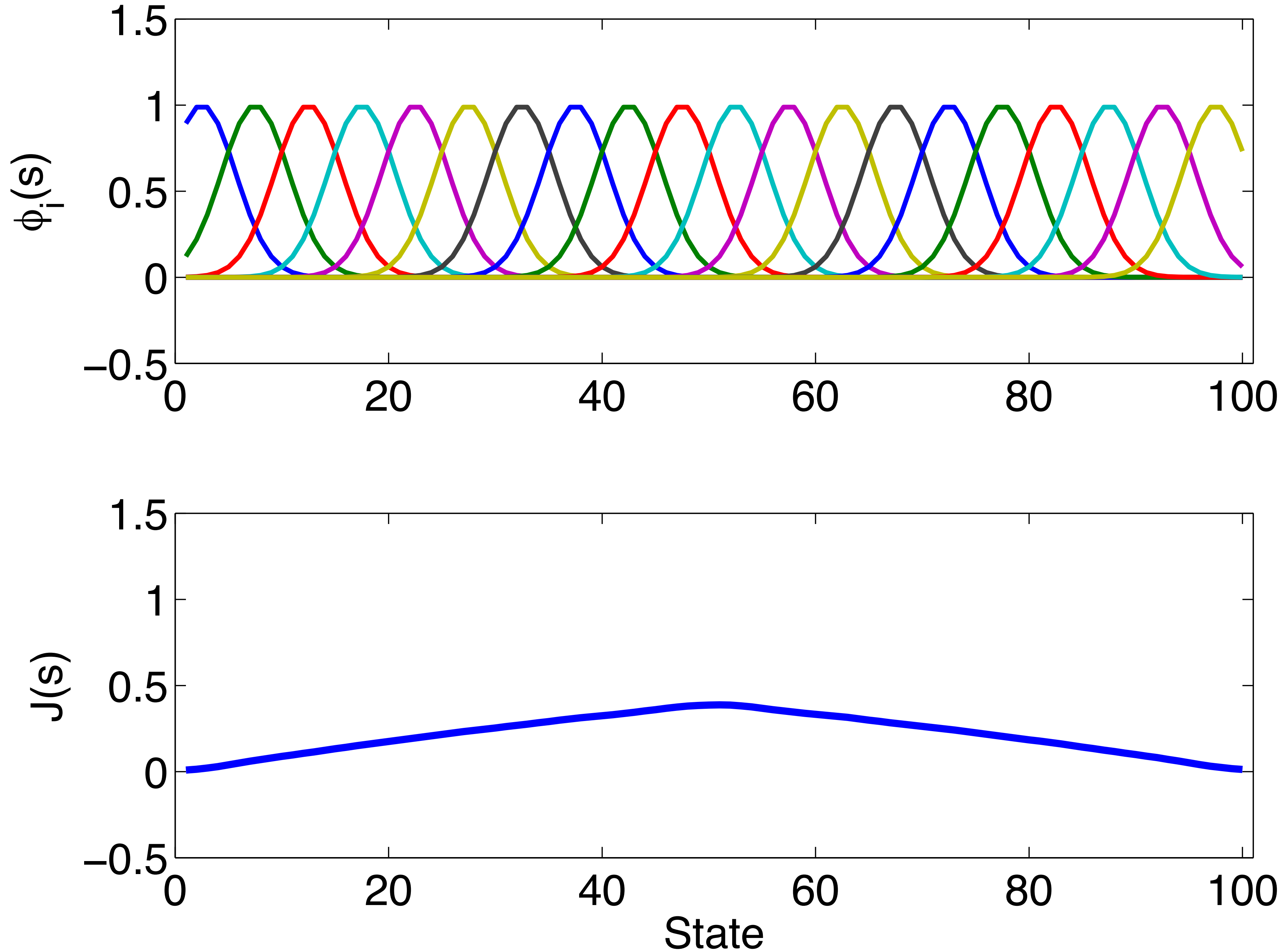
- Data: $\tau = (s_1, a_1, c_1, s_2, a_2, c_2, \ldots) \sim \pi$
- Want $Q(s_t, a_t) \approx (1-\gamma)c_t + \gamma Q(s_{t+1}, a_{t+1})$
  - $w^T\Phi(s_t, a_t) \approx (1-\gamma)c_t + \gamma w^T\Phi(s_{t+1}, a_{t+1})$
  - Φ = vector of k features, w = weight vector

LSTD

- $w^T\Phi(s_t, a_t) \approx (1-\gamma)c_t + \gamma w^T\Phi(s_{t+1}, a_{t+1})$
- Vector notation:
  - $Fw \approx (1-\gamma)c_t + \gamma F_1 w$
- Overconstrained: multiply both sides by F
  - $F^TFw = (1-\gamma)F^Tc_t + \gamma F^TF_1 w$

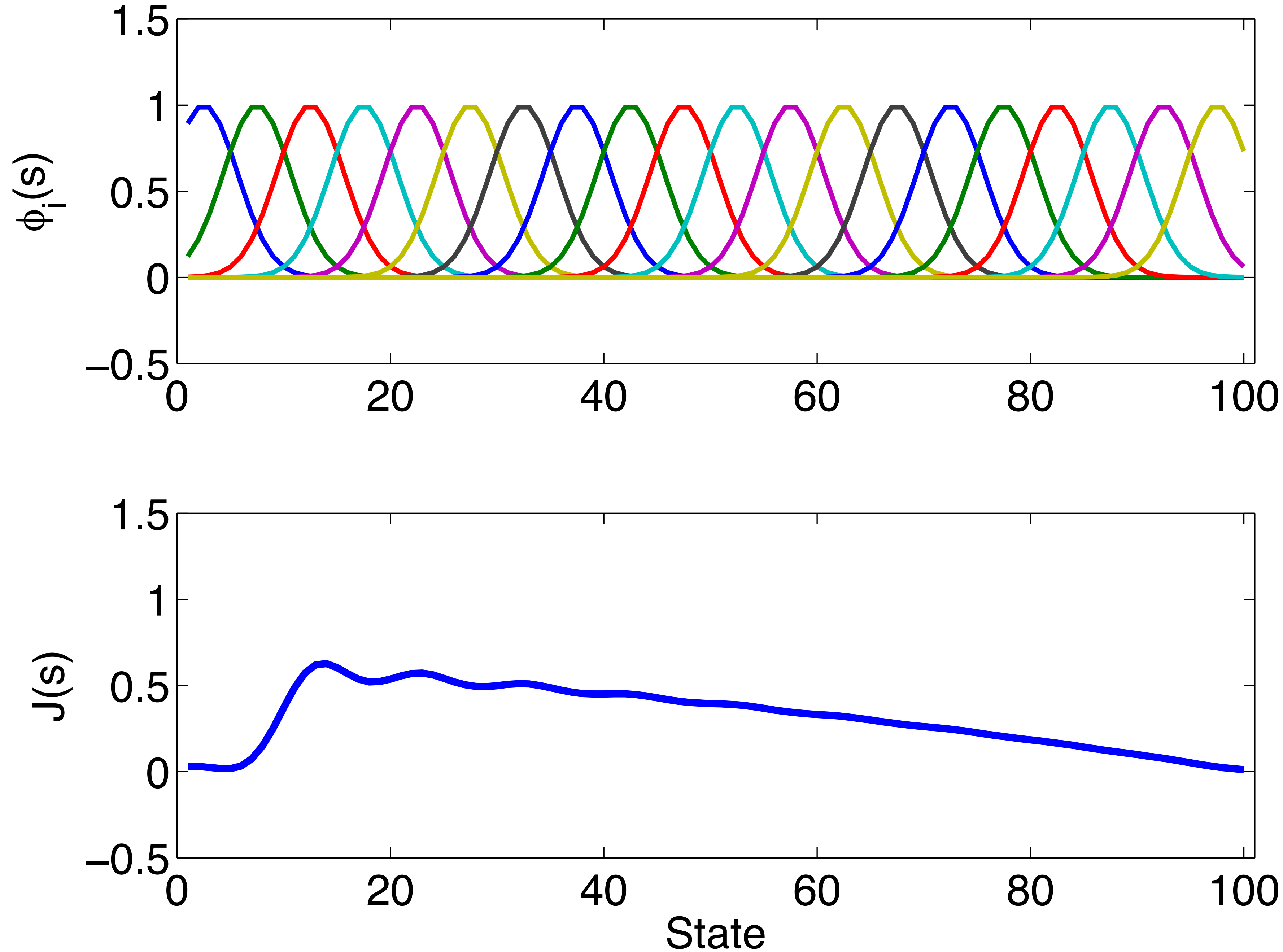
LSTD: example

- 100 states in a line; move left or right at cost 1 per state; goals at both ends; discount 0.99

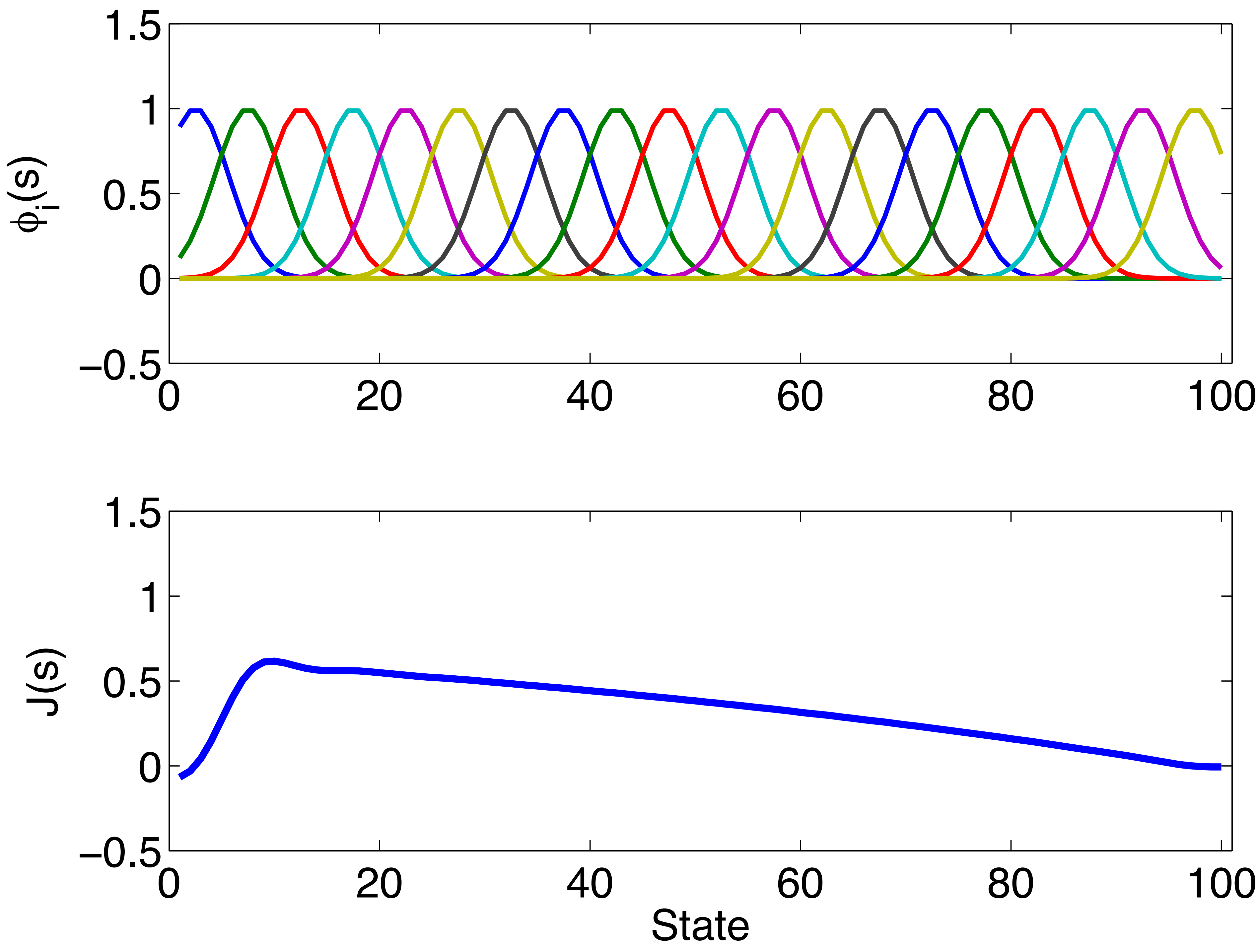


LSTD: example

- 100 states in a line; move left or right at cost 1 per state; goals at both ends; discount 0.99

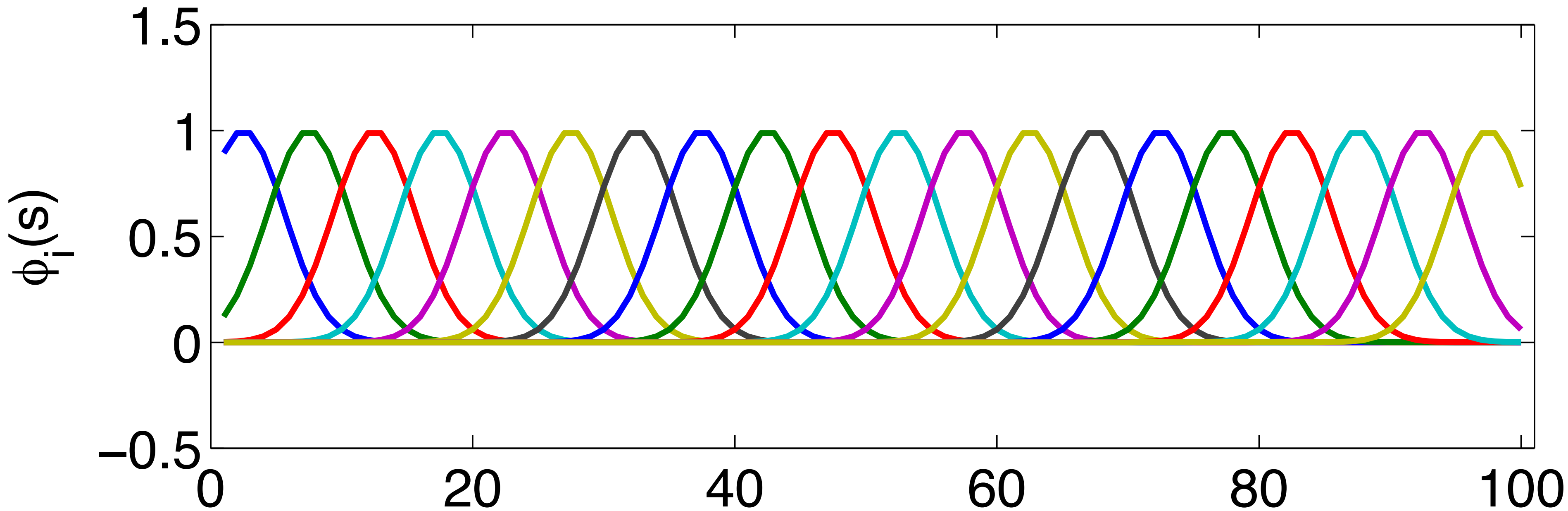


LSPI

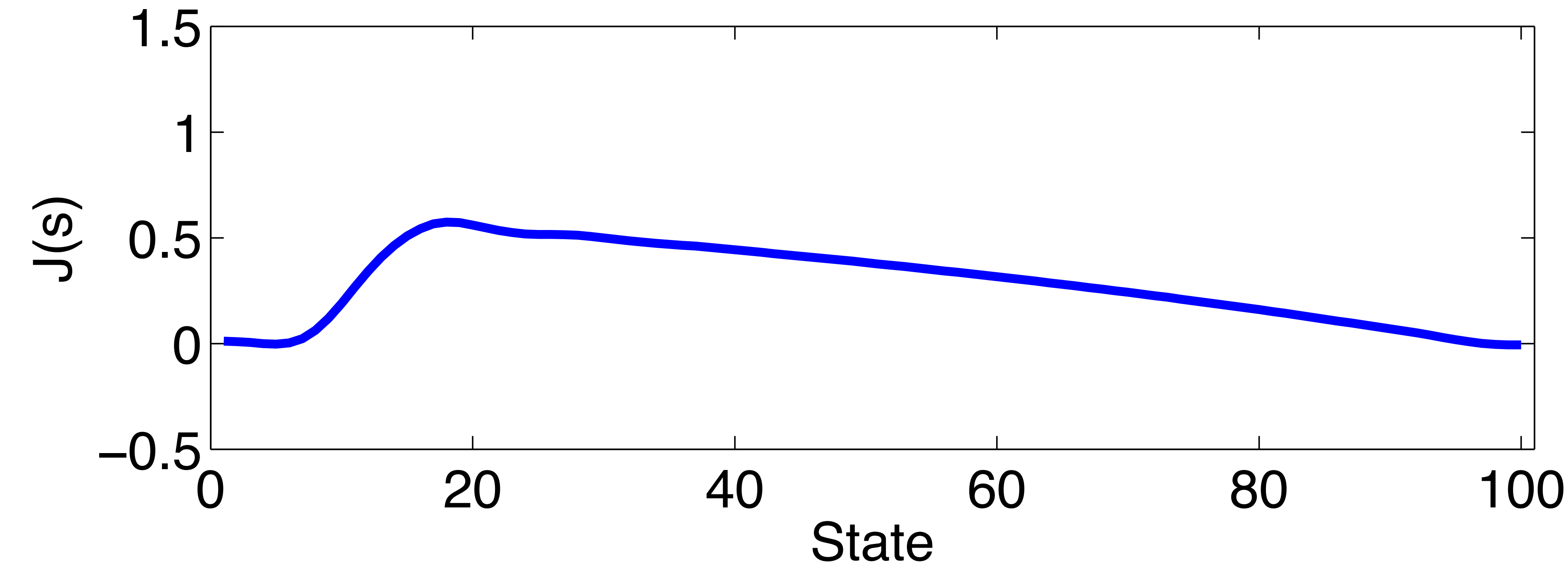


1.5
1
0.5
0
−0.5
$\phi_i(s)$
0
20
40
60
80
100
1.5
1
0.5
0
−0.5
J(s)
0
20
40
60
80
100
State

LSPI

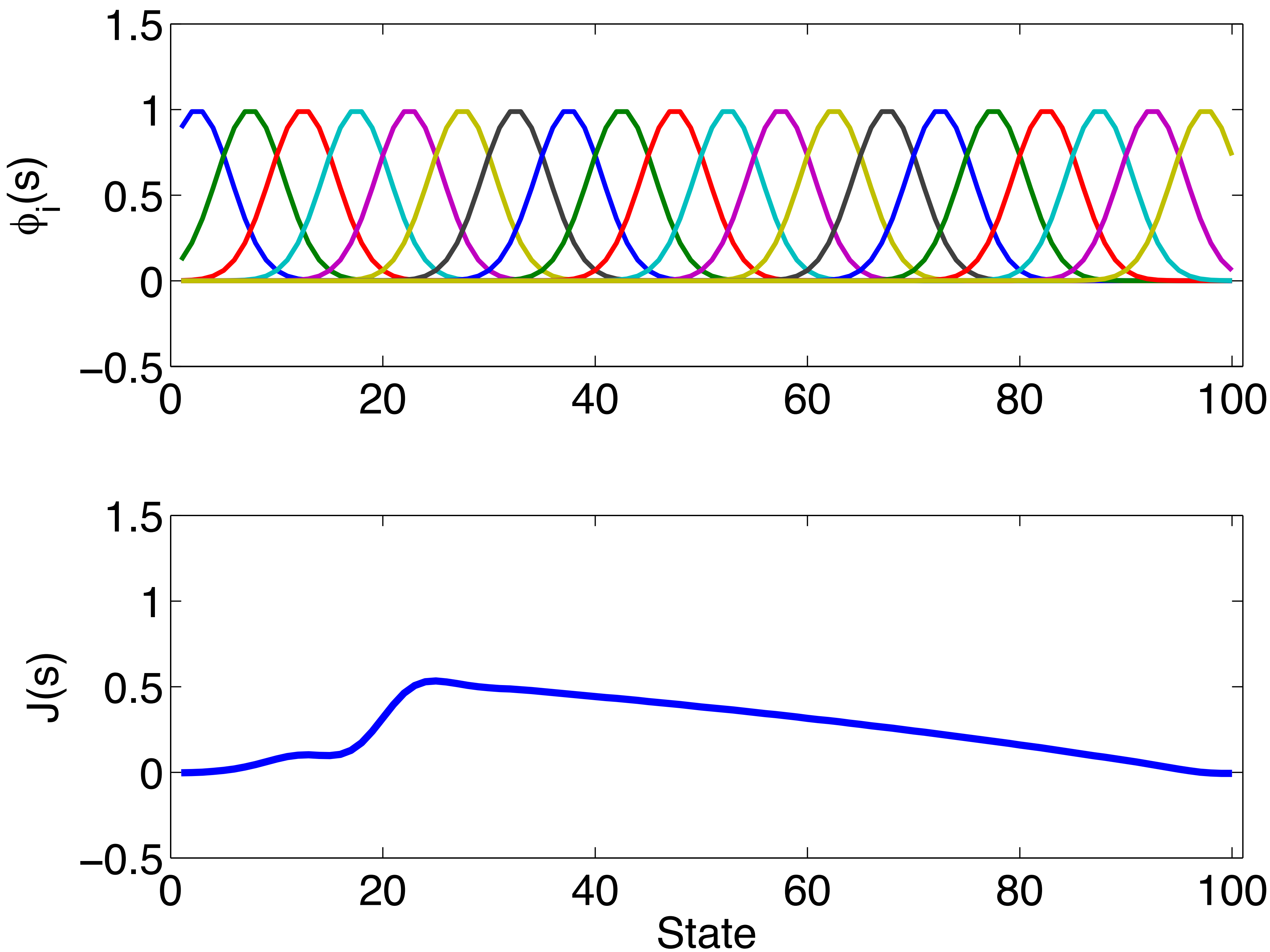


1.5
1
0.5
0
−0.5
φi(s)
0
20
40
60
80
100



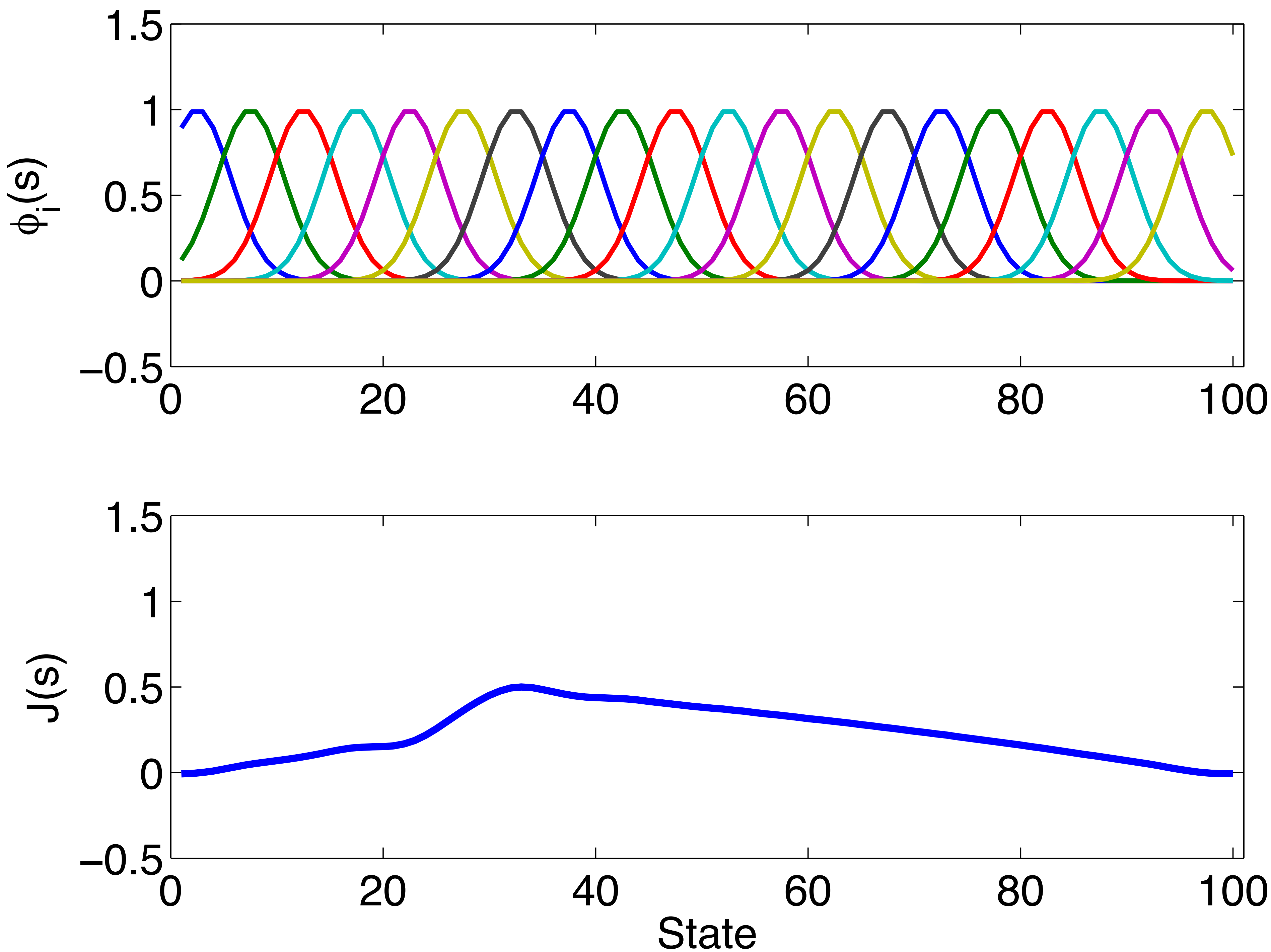
1.5
1
0.5
0
−0.5
J(s)
0
20
40
60
80
100
State

LSPI



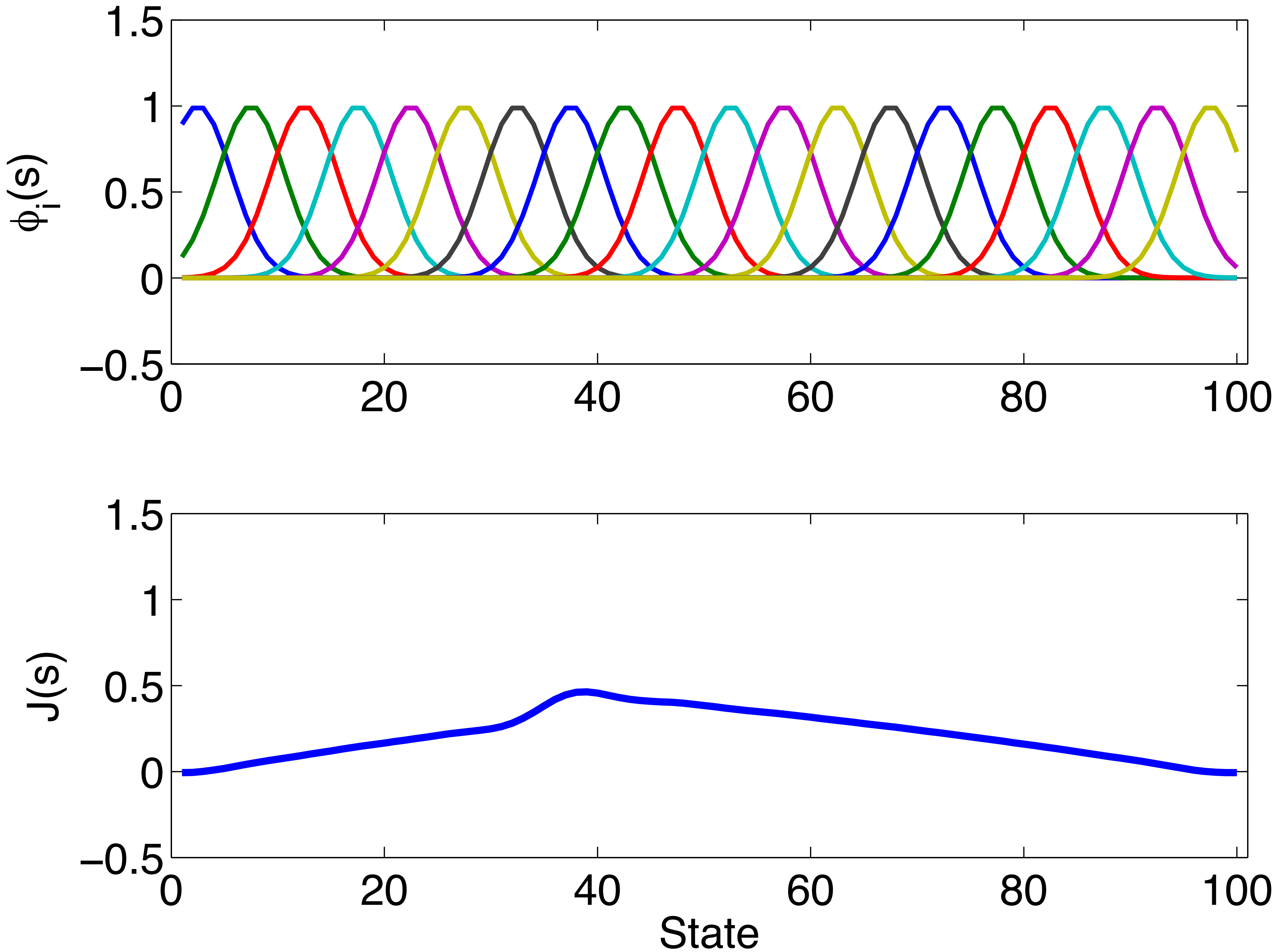
1.5
1
0.5
0
−0.5
$\phi_i(s)$
0
20
40
60
80
100
1.5
1
0.5
0
−0.5
J(s)
0
20
40
60
80
100
State

LSPI



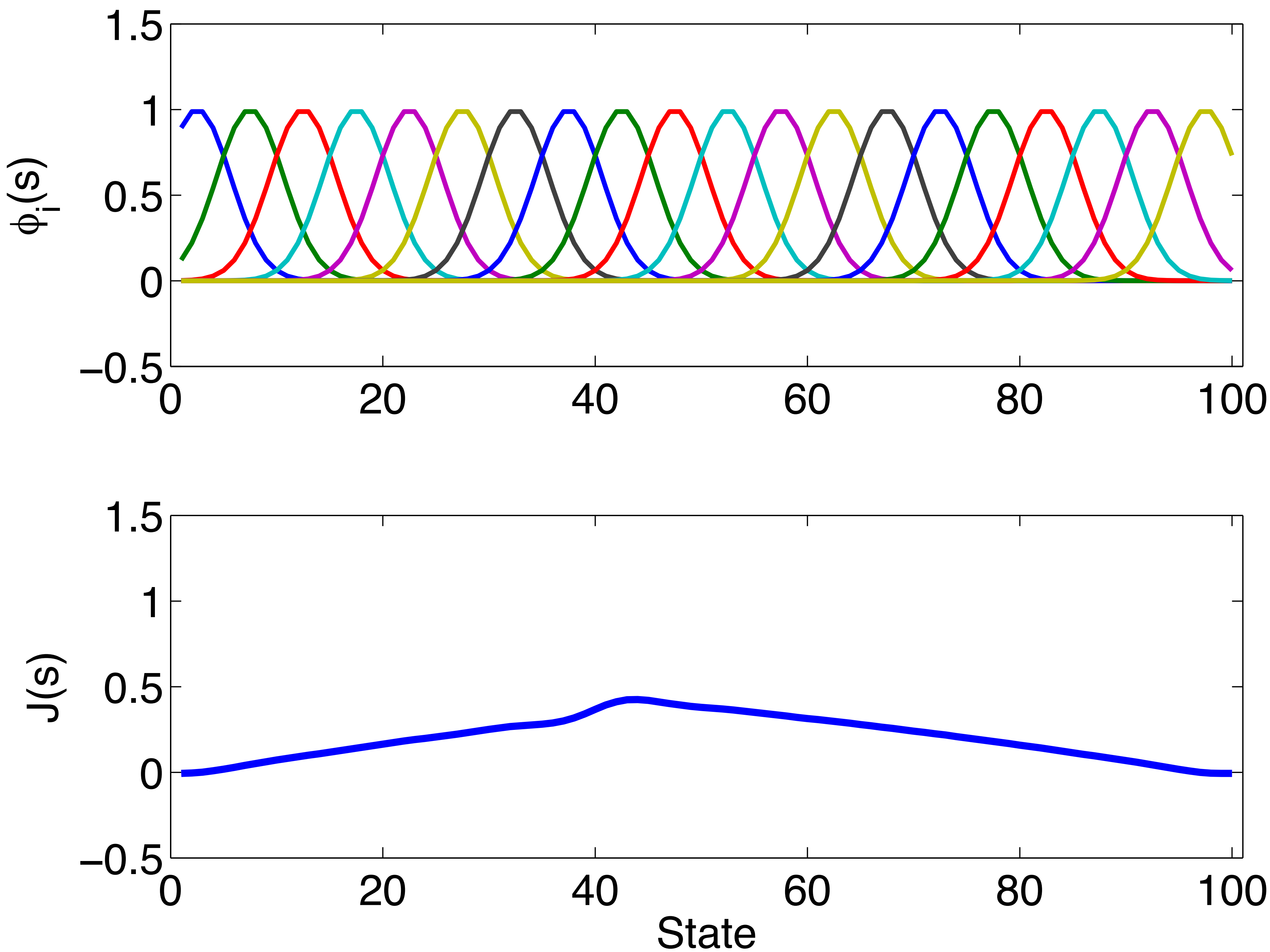
1.5
1
0.5
0
−0.5
0
20
40
60
80
100
φi(s)
1.5
1
0.5
0
−0.5
0
20
40
60
80
100
J(s)
State

LSPI

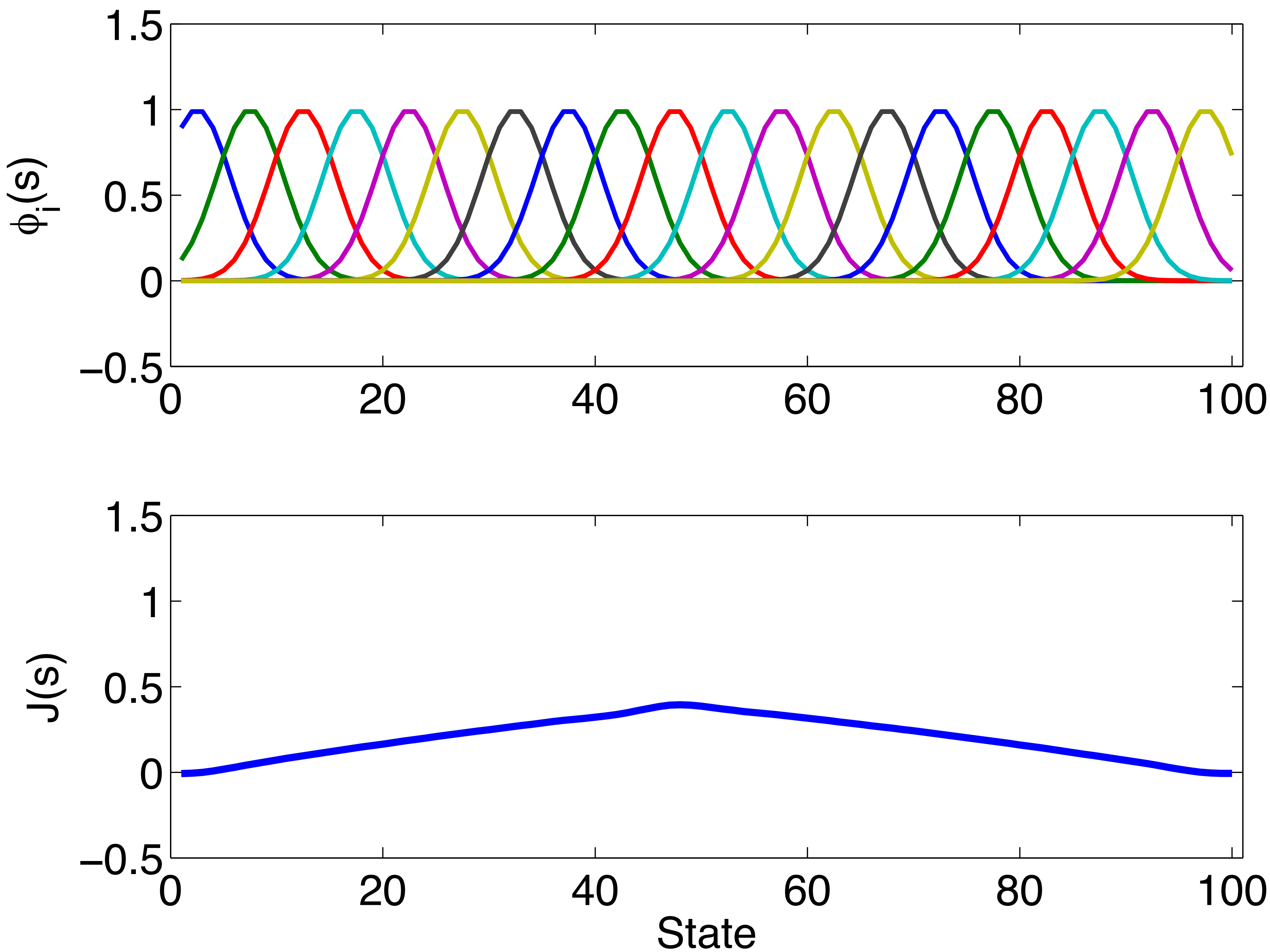


1.5
1
0.5
0
−0.5
0
20
40
60
80
100
φ_i(s)
1.5
1
0.5
0
−0.5
0
20
40
60
80
100
J(s)
State

LSPI


1.5
1
0.5
0
−0.5
φi(s)
0
20
40
60
80
100
1.5
1
0.5
0
−0.5
J(s)
0
20
40
60
80
100
State

LSPI



1.5
1
0.5
0
−0.5
φi(s)
0
20
40
60
80
100
1.5
1
0.5
0
−0.5
J(s)
0
20
40
60
80
100
State