A Dual–SPMA Framework for Convex MDPs

Fenchel Duality + Softmax Policy Mirror Ascent

Shervin Khamooshian Ahmed Magd Pegah Aryadoost Danielle Nguyen

School of Computing Science, Simon Fraser University

Project Presentation

Main Claim

Fenchel duality + a fast policy optimizer (SPMA) gives a simple, competitive way to solve convex MDPs; we compare this Dual–SPMA recipe against NPG–PD.

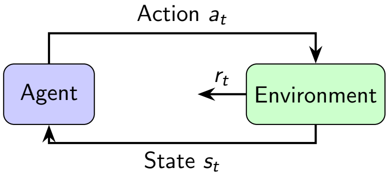
Outline

What is Reinforcement Learning?

Reinforcement Learning (RL) is a learning framework where an agent learns to make decisions by interacting with an environment.

At each time step t, the agent:

1. Observes a **state** s_t
2. Chooses an **action** a_t (based on a policy)
3. Receives a **reward** r_t
4. Transitions to a new **state** s_{t+1}



Markov Decision Process (MDP): Formal Definition

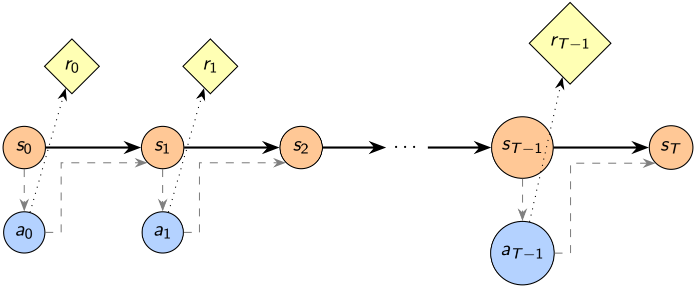
An **MDP** is defined by the tuple $(\mathcal{S}, \mathcal{A}, P, r, \gamma, \rho)$:

Symbol	Meaning
$\mathcal{S}$	State space (set of all possible states)
$\mathcal{A}$	Action space (set of all possible actions)
$P(s' \mid s, a)$	Transition probability: probability of reaching s' from (s, a)
$r(s, a)$	Reward function: immediate reward for taking action a in state s
$\gamma \in [0, 1)$	Discount factor: how much to value future vs. immediate rewards
$\rho(s)$	Initial state distribution

Policy $\pi(a \mid s)$: probability of taking action a in state s.

Trajectory and Return

A **trajectory** τ is a sequence of states, actions, and rewards:



Discounted Return

The **expected discounted return** under policy π is: $J(\pi) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)\right]$

Goal of RL: Find $\pi^\star = \arg\max_\pi J(\pi)$

Occupancy Measure: Where the Policy Spends Time

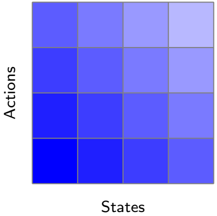
Discounted Occupancy Measure

$$d_\pi(s, a) = (1 - \gamma) \sum_{t=0}^{\infty} \gamma^t \Pr_\pi(s_t = s, a_t = a)$$

Notation:

- $d_\pi(s, a)$: probability of being in state s and taking action a under policy π
- $(1 - \gamma)$: normalization factor
- $\Pr_\pi$: probability under policy π

Key property: d_π is a probability distribution: $\sum_{s,a} d_\pi(s, a) = 1$



Think of d_π as a **heatmap**: bright $=$ often visited

Key Identity: RL is Linear in Occupancy

Fundamental Identity

$$\langle r, d_\pi \rangle = \sum_{s,a} r(s,a)\, d_\pi(s,a) = (1-\gamma)\, J(\pi)$$

What does this mean?

- Up to the constant $(1-\gamma)$, the RL objective is **linear** in d_π
- Weights $=$ how often we visit each (s,a) pair
- Standard RL $\Rightarrow$ maximize a **linear** function of d_π

But many interesting goals are NOT linear in d_π:

Goal	**Objective**
Safety constraints	$\max J_r(\pi)$ subject to $\langle c, d_\pi \rangle \leq \tau$
Imitation learning	$\min \lVert d_\pi - d_{\text{expert}} \rVert$
Exploration	$\max J_r(\pi) + \alpha H(d_\pi)$

Why Convex MDPs?

Problem: Linear RL is insufficient for:

- Safety constraints
- Matching expert behavior
- Encouraging exploration
- Risk-sensitive objectives

Solution: Convex MDPs

$$\min_{\pi} f(d_\pi)$$

where f is a **convex function**

Challenge:

- Optimizing over occupancy measures is hard
- High-dimensional constrained space
- Can't directly apply standard RL

Our approach:

- Use **Fenchel duality**
- Transform to min-max game
- Reduce to shaped-reward RL

Convex MDP: Examples

General Form

$\min_{d \in \mathcal{D}} f(d)$ where $\mathcal{D}$ = feasible occupancy measures, f = convex

Example 1: Standard RL (linear, trivially convex)

$$f(d) = -\langle r, d \rangle = -\sum_{s,a} r(s,a)\, d(s,a)$$

Example 2: Entropy-Regularized RL

$$f(d) = -\langle r, d \rangle + \alpha \sum_{s,a} d(s,a) \log d(s,a)$$

Example 3: Constrained Safety (CMDP)

$$f(d) = -\langle r, d \rangle + \mu \max\{0, \langle c, d \rangle - \tau\}$$

where $c(s,a)$ = cost function, τ = threshold, μ = fixed penalty weight

Roadmap

✓ Background: Reinforcement Learning

✓ Motivation: Convex MDPs

→ **Problem Formulation: Fenchel Duality**

Related Work: “Reward Is Enough” & SPMA

Our Method: Dual–SPMA

Experiments

Conclusion & Future Work

Fenchel Conjugate: Definition

Given a convex function $f : \mathbb{R}^n \to \mathbb{R} \cup \{\infty\}$

Fenchel Conjugate (Convex Conjugate)

$$f^*(y) = \sup_{x \in \mathbb{R}^n} \{\langle y, x \rangle - f(x)\}$$

Notation:

- $f^*(y)$: the conjugate function evaluated at dual variable y
- $\langle y, x \rangle = \sum_i y_i x_i$: inner product (dot product)
- sup: supremum (least upper bound)

Intuition: $f^*(y)$ measures "how much $\langle y, x \rangle$ can exceed $f(x)$"

Fenchel–Moreau Theorem

Fenchel–Moreau Identity

For any proper, closed, convex function f:

$$f(d) = \sup_{y} \{\langle y, d\rangle - f^*(y)\}$$

What does this say?

- We can **recover** f from its conjugate f^*
- f is the conjugate of its conjugate: $f = (f^*)^*$
- This is called **biconjugation**

Why is this useful?

- Transforms a minimization problem into a **min-max** problem
- Introduces a **dual variable** y that we can optimize over

Applying Fenchel Duality to Convex MDPs

Step 1: Start with the convex MDP problem

$$\min_{d \in \mathcal{D}} f(d)$$

Step 2: Apply Fenchel–Moreau identity

$$\min_{d \in \mathcal{D}} f(d) = \min_{d \in \mathcal{D}} \sup_{y} \{\langle y, d \rangle - f^*(y)\}$$

Step 3: This is a **convex-concave saddle-point problem**

$$= \min_{d \in \mathcal{D}} \max_{y} \{\langle y, d \rangle - f^*(y)\}$$

(Under standard conditions, solving this saddle-point is equivalent to the original problem.)

Step 4: Replace d with d_π (occupancy induced by policy)

Saddle-Point Formulation

$$\min_{\pi} \max_{y} \underbrace{\langle y, d_\pi \rangle - f^*(y)}_{L(\pi, y)}$$

Two-Player Game Interpretation

Saddle-Point Problem

$$\min_{\pi} \max_{y} L(\pi, y), \quad \text{where } L(\pi, y) = \langle y, d_\pi \rangle - f^*(y)$$

This is a **min-max game** between two players:

Policy Player (min)

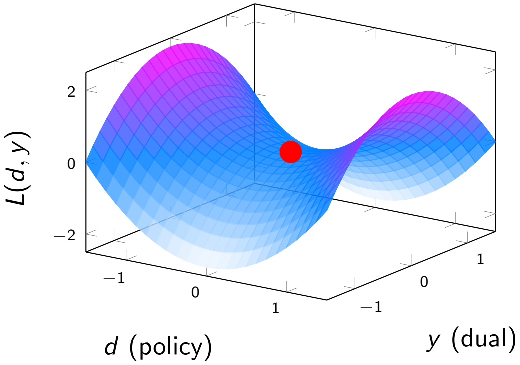
- Chooses policy π
- Wants to minimize L
- Controls occupancy d_π

Dual Player (max)

- Chooses dual variable y
- Wants to maximize L
- Shapes the reward signal

$\boxed{\text{At equilibrium: policy player finds optimal } \pi^*, \text{ dual player finds optimal } y^*}$

Visualizing the Saddle Point



Min-max solution:

$$\min_d \max_y L(d, y)$$

At the **red point** (saddle point):

- Along d-direction: **minimal**
- Along y-direction: **maximal**

Policy player (min) and dual player (max) reach **equilibrium** here.

From Saddle Point to Shaped Reward

For **fixed** dual variable y, the policy player solves: $\min_{\pi} \langle y, d_\pi \rangle$

Expand using the occupancy definition:

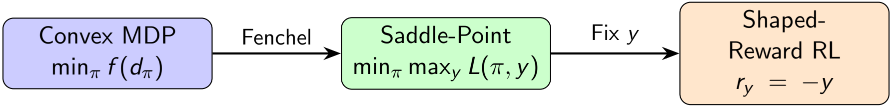
$$\langle y, d_\pi \rangle = \sum_{s,a} y(s,a) \cdot d_\pi(s,a) = (1-\gamma)\mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t y(s_t, a_t) \right]$$

Key Insight: Shaped Reward

$$\min_{\pi} \langle y, d_\pi \rangle = \min_{\pi} \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t y(s_t, a_t) \right] = \max_{\pi} \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t \underbrace{(-y(s_t, a_t))}_{r_y(s_t, a_t)} \right]$$

Conclusion: Policy player just does **standard RL** with shaped reward: $\boxed{r_y(s,a) = -y(s,a)}$

Summary: The Fenchel Dual Reduction



Algorithm structure:

1. **Dual step:** Update y using gradient of L w.r.t. y

$$y_{k+1} = y_k + \alpha \left(d_{\pi_k} - \nabla f^*(y_k) \right)$$

2. **Policy step:** Run RL algorithm with reward $r_{y_k} = -y_k$

$$\pi_{k+1} = \text{RL-Update}(\pi_k, r_{y_k})$$

Roadmap

- ✓ Background: Reinforcement Learning
- ✓ Motivation: Convex MDPs
- ✓ Problem Formulation: Fenchel Duality
- → **Related Work: “Reward Is Enough” & SPMA**
- Our Method: Dual–SPMA
- Experiments
- Conclusion & Future Work

Where We Are

We’ve established the Fenchel dual reduction. Now we review the key papers that inform our method: the theoretical foundation and the policy optimizer we’ll use.

Related Work: "Reward Is Enough" (Zahavy et al., 2021)

Main contributions of this foundational paper:

1. **Fenchel dual reduction:**
   - Reformulate convex MDP as saddle-point problem
   - The theoretical foundation we just presented
2. **Meta-algorithm:**
   - Alternating updates between policy and dual players
   - Any RL algorithm can be the policy player
   - Any online convex optimization (OCO) can be the dual player
3. **Unification:**
   - Shows many RL paradigms are special cases of convex MDPs
   - Imitation learning, constrained RL, entropy-regularized RL

Our contribution: Implement this framework with SPMA as the policy player

Related Work: Softmax Policy Mirror Ascent (Asad et al., 2024)

Softmax Policy Mirror Ascent (SPMA):

- Mirror ascent in *logit space* using log-sum-exp mirror map
- Achieves **linear convergence** in tabular MDPs

Tabular SPMA Update Rule

$$\pi_{t+1}(a|s) = \pi_t(a|s) \cdot \left(1 + \eta A^{\pi_t}(s, a)\right)$$

where $A^{\pi}(s, a) = Q^{\pi}(s, a) - V^{\pi}(s)$ is the advantage function[1]

Why use SPMA as our policy player?

- **Geometry-aware:** Updates respect the simplex structure
- **No normalization:** Unlike vanilla PG, no per-state renormalization
- **Fast:** Linear convergence vs. sublinear for vanilla PG
- **Function approximation:** Clean extension via convex classification

[1] In tabular theory, this uses $[1 + \eta A]_+$ and implicit renormalization; simplified form shown here.

Related Work: NPG–PD (Our Baseline)

Natural Policy Gradient Primal–Dual (Ding et al., 2020):
For constrained MDPs with Lagrangian:

CMDP Lagrangian

$$L(\pi, \lambda) = J_r(\pi) + \lambda(J_c(\pi) - \tau), \quad \lambda \geq 0$$

Primal step (policy):

- Natural policy gradient ascent
- Uses Fisher information matrix
- Geometry-aware like SPMA

Dual step (constraint):

$$\lambda_{k+1} = [\lambda_k + \beta(J_c(\pi_k) - \tau)]_+$$

- Projected gradient ascent
- $[\cdot]_+ = \max(0, \cdot)$

Guarantees: $\mathcal{O}(1/\sqrt{T})$ optimality gap and constraint violation

Roadmap

✓ Background: Reinforcement Learning

✓ Motivation: Convex MDPs

✓ Problem Formulation: Fenchel Duality

✓ Related Work: "Reward Is Enough" & SPMA

→ **Our Method: Dual–SPMA**

Experiments

Conclusion & Future Work

Our Contributions

1. **Dual–SPMA Framework:**
   - Complete implementation of outer dual loop + SPMA policy oracle
   - Supports entropy-regularized RL and constrained safety (CMDP)
2. **Three Occupancy Estimators:**
   - Tabular Monte Carlo
   - Feature-based Monte Carlo
   - MLE-style estimator (following Barakat et al., 2024)
3. **NPG–PD Baseline:**
   - Faithful implementation for fair comparison
   - Same architecture and hyperparameters where possible
4. **Empirical Comparison:**
   - SPMA vs NPG–PD on constrained safety tasks

Dual–SPMA Loop: High-Level View

Saddle-Point Problem

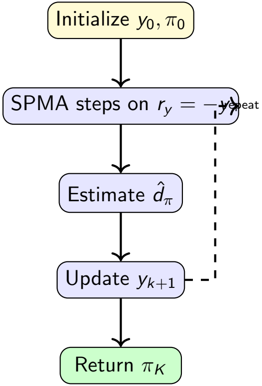
$$\min_{\pi} \max_{y} \underbrace{\langle y, d_\pi \rangle - f^*(y)}_{L(\pi, y)}$$

Outer loop (dual):

$$y_{k+1} = y_k + \alpha(\hat{d}_{\pi_k} - \nabla f^*(y_k))$$

Inner loop (policy):

- Run K_{in} SPMA steps
- Shaped reward: $r_{y_k} = -y_k$



Policy Player: SPMA Actor Loss

Standard Policy Gradient Loss:

$$\mathcal{L}_{\text{PG}} = -\mathbb{E}\left[\log \pi(a|s) \cdot A(s,a)\right]$$

SPMA adds a "stay close" regularizer:

SPMA Actor Loss

$$\mathcal{L}_{\text{SPMA}} = \mathbb{E}\left[-\Delta \log \pi \cdot A + \frac{1}{\eta} \underbrace{\left(\exp(\Delta \log \pi) - 1 - \Delta \log \pi\right)}_{\text{KL-like regularizer}}\right]$$

Notation:

- $\Delta \log \pi = \log \pi_{\text{new}}(a|s) - \log \pi_{\text{old}}(a|s)$: change in log-probability
- $A(s,a)$: advantage function
- η: step size parameter (chosen via Armijo line search)

Occupancy Estimation: MC vs MLE

Monte Carlo (our default)

Tabular Estimator

$$\hat{d}_\pi(s, a) = \frac{1-\gamma}{N} \sum_{i=1}^{N} \sum_{t=0}^{T} \gamma^t \mathbf{1}\{s_t^{(i)} = s, a_t^{(i)} = a\}$$

- Simple: count discounted visits
- Variance grows with $|\mathcal{S}||\mathcal{A}|$

MLE (Barakat et al., 2024)

Log-Linear Model

$$\lambda_\omega(s, a) \propto \exp(\omega^\top \phi(s, a))$$

- Fit ω by max-likelihood
- Error: $O(\sqrt{m/n})$
- Independent of $|S||A|$!

Sanity check: $\sum_{s,a} \hat{d}_\pi(s, a) \approx 1$ verified in all our tests

Baseline: NPG–PD Implementation

Same Lagrangian as Dual–SPMA:

$$L(\pi, \lambda) = J_r(\pi) + \lambda(J_c(\pi) - \tau)$$

Primal (policy):

- Natural PG on shaped reward $r_\lambda = r - \lambda c$
- Diagonal Fisher approximation
- Same actor-critic as SPMA

Dual (constraint):

$$\lambda_{k+1} = [\lambda_k + \beta(J_c - \tau)]_+$$

- One NPG step per iteration
- Evaluate J_c via Monte Carlo

Fair comparison: Same networks, same hyperparameters where possible

Example: Constrained Safety CMDP

Problem: Maximize reward subject to safety constraint

$$\max_{\pi} J_r(\pi) \quad \text{s.t.} \quad J_c(\pi) \leq \tau$$

where $J_r(\pi) = \mathbb{E}_\pi[\sum_t \gamma^t r(s_t, a_t)]$ and $J_c(\pi) = \mathbb{E}_\pi[\sum_t \gamma^t c(s_t, a_t)]$.

Dual–SPMA approach:

1. Build dual variable: $y_\lambda(s, a) = \lambda c(s, a) - r(s, a)$
2. Policy sees shaped reward: $r_y = -y = r - \lambda c$
3. Run SPMA inner loop on r_y
4. Update dual: $\lambda_{k+1} = [\lambda_k + \beta(J_c(\pi_k) - \tau)]_+$

SPMA vs NPG–PD: Same dual update, different policy optimizer!
Only difference is Step 3: SPMA loss vs. natural gradient

Roadmap

- ✓ Background: Reinforcement Learning
- ✓ Motivation: Convex MDPs
- ✓ Problem Formulation: Fenchel Duality
- ✓ Related Work: “Reward Is Enough” & SPMA
- ✓ Our Method: Dual–SPMA
- $\rightarrow$ **Experiments**
- Conclusion & Future Work

Experimental Setup

Environments:

- FrozenLake 4×4 (tabular)
- Deterministic transitions
- Unsafe states = holes (cost $c = 1$)

Methods Compared:

- Dual–SPMA (ours)
- NPG–PD baseline

Metrics:

- $J_r(\pi)$: reward return
- $J_c(\pi)$: cost return
- $J_c - \tau$: constraint violation
- $\sum \hat{d}_\pi$: estimator sanity

Hyperparameters:

- Discount $\gamma = 0.99$
- Safety threshold $\tau = 0.1$
- 30 outer iterations
- 2048 steps/rollout

Results: Entropy-Regularized RL

$L(\pi, y)$ vs iterations

(Add plot here)

Saddle value $L(\pi, y)$ vs iterations

$\sum \hat{d}_\pi$ vs iterations

(Add plot here)

$\sum_{s,a} \hat{d}_\pi(s, a)$ vs iterations

Observations:

- Saddle value $L(\pi, y)$ converges smoothly
- Occupancy estimate stays near 1 (estimator is consistent)

Results: Constrained Safety (Dual–SPMA)

J_r, J_c vs iterations

(Add plot here)

$J_r(\pi_k)$ and $J_c(\pi_k)$ vs iterations

Constraint violation

(Add plot here)

Constraint violation $J_c - \tau$ and λ_k

Observations:

- λ increases when constraint violated, decreases otherwise
- Constraint violation $\to 0$ as training progresses

Results: Dual–SPMA vs NPG–PD

J_r comparison
SPMA vs NPG-PD

(Add plot here)

$J_r(\pi_k)$ vs outer iterations

Constraint violation comparison

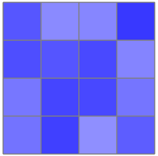
(Add plot here)

Constraint violation vs iterations

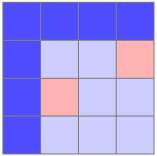
Takeaways:

- Both methods eventually satisfy the constraint
- SPMA: larger, geometry-aware steps
- NPG–PD: smoother but requires careful step-size tuning

Results: Occupancy Heatmaps



Unconstrained



Constrained

Left: Unconstrained
Policy explores more broadly

Right: Safety-Constrained
Policy avoids unsafe states (holes)

Takeaways & Limitations

The Recipe:

$$\text{Convex MDP} \xrightarrow{\text{Fenchel}} \text{Saddle-Point Game} \xrightarrow{\text{SPMA}} \text{Shaped-Reward RL}$$

What we built:

- Dual–SPMA loops for entropy-regularized RL and constrained safety
- NPG–PD baseline for fair comparison
- Three occupancy estimators (tabular MC, feature MC, MLE)

Limitations:

- Experiments only in low-dimensional (tabular) environments
- MLE estimator not yet stress-tested on large continuous tasks
- Hyperparameter sensitivity not fully characterized

Future Work

1. **Scale up:**
   - Test on larger CMDPs (continuous states/actions)
   - MuJoCo safety tasks
2. **Better estimation:**
   - Evaluate MLE estimator on high-dimensional tasks
   - Compare variance of different estimators
3. **More objectives:**
   - Risk-sensitive RL
   - Imitation learning via convex MDP
4. **Theoretical analysis:**
   - Convergence rates for Dual–SPMA
   - Sample complexity comparison with NPG–PD

Questions?

Pegah:	Background & Motivation
Danielle:	Problem Formulation & Fenchel Duality
Shervin:	Related Work & Our Method
Ahmed:	Experiments & Conclusion

References

1. **Zahavy, T., et al.** (2021). "Reward is Enough for Convex MDPs." *NeurIPS 2021*. arXiv:2108.06389
2. **Asad, A., et al.** (2024). "Fast Convergence of Softmax Policy Mirror Ascent." *arXiv:2405.09781*
3. **Ding, D., et al.** (2020). "Natural Policy Gradient Primal-Dual Method for Constrained Markov Decision Processes." *NeurIPS 2020*
4. **Barakat, A., et al.** (2024). "Reinforcement Learning with General Utilities: Simpler Variance Reduction and Large State-Action Space." *ICML 2024*
5. **Schulman, J., et al.** (2015). "Trust Region Policy Optimization." *ICML 2015*
6. **Sutton, R. & Barto, A.** (2018). "Reinforcement Learning: An Introduction." *MIT Press*

Backup: Flow Constraints (Occupancy Polytope)

The set $\mathcal{D}$ of valid occupancy measures satisfies **Bellman flow constraints**:
For all states s:

$$\sum_a d(s,a) = (1-\gamma)\rho(s) + \gamma \sum_{s',a'} P(s|s',a')\, d(s',a')$$

Also: $d(s,a) \geq 0$ for all (s,a)

Interpretation:

- Flow into state s = initial distribution + discounted flow from other states
- This is a **convex polytope** in $\mathbb{R}^{|S|\times|A|}$

Backup: Entropy-Regularized Conjugate

For entropy-regularized objective:

$$f(d) = -\langle r, d \rangle + \alpha \sum_{s,a} d(s,a) \log d(s,a)$$

$$f^*(y) = \alpha \log \sum_{s,a} \exp\left(\frac{y(s,a) + r(s,a)}{\alpha}\right)$$

$$\nabla f^*(y) = \mathrm{softmax}\left(\frac{y + r}{\alpha}\right)$$

The gradient is a softmax distribution—very convenient for computation!

Backup: SPMA with Function Approximation

In function approximation, the SPMA projection step becomes:

$$\theta_{t+1} = \arg\min_{\theta} \sum_{s} d^{\pi_t}(s)\, \mathrm{KL}\big(\pi_{t+1/2}(\cdot|s) \,\|\, \pi_\theta(\cdot|s)\big)$$

This is a **convex optimization problem** (weighted KL minimization).

Equivalent to **softmax classification**:

- Labels: actions from $\pi_{t+1/2}$
- Weights: state occupancies $d^{\pi_t}(s)$
- Features: state representations

Backup: Algorithm Pseudocode

Dual–SPMA Algorithm:

1. Initialize dual variable $y_1 = 0$, policy π_1 randomly
2. For $k = 1, 2, \ldots, K_{\text{outer}}$:
    1. **Policy step:** Run K_{inner} SPMA iterations on shaped reward $r_{y_k} = -y_k$
    $$\pi_{k+1} = \text{SPMA}(\pi_k, r_{y_k}, K_{\text{inner}})$$
    2. **Estimate occupancy:** Collect trajectories, compute $\hat{d}^{\pi_{k+1}}$
    3. **Dual step:** Update dual variable
    $$y_{k+1} = y_k + \alpha\left(\hat{d}^{\pi_{k+1}} - \nabla f^*(y_k)\right)$$
3. Return final policy π_K

Backup: Notation Summary

Symbol	Meaning
$\mathcal{S}, \mathcal{A}$	State and action spaces
$\pi(a\|s)$	Policy (probability of action a in state s)
$r(s, a)$	Reward function
$c(s, a)$	Cost function (for CMDPs)
γ	Discount factor
$d_\pi(s, a)$	Occupancy measure under policy π
$J(\pi)$	Expected return
$f(d)$	Convex objective over occupancies
$f^*(y)$	Fenchel conjugate of f
y	Dual variable
λ	Lagrange multiplier (for CMDPs)
τ	Safety threshold
η	SPMA step size
α	Dual step size