Off-Policy Learning in Partially Observed Markov Decision Processes under Sequential Ignorability

Jongmin Mun
University of Southern California
jongminm@usc.edu

Abstract

Learning treatment policies from historical mobile health data is challenging due to partial observability and the absence of online experimentation. Off-policy evaluation with history-dependent importance weighting provides unbiased estimates but induces a highly ill-conditioned, nonconvex objective, making naive gradient-based optimization unstable. We propose a weighted second-moment preconditioned gradient ascent method that stabilizes learning by accounting for the variance and curvature of importance-weighted gradients. In simulations motivated by glucose regulation, our method converges faster, reduces variability across runs, and achieves higher estimated policy values than naive gradient ascent. These results underscore the importance of geometry-aware optimization in off-policy learning under partial observability.

1 Introduction and Research Motivation

The growing use of smartphone-based health applications has created new opportunities for delivering personalized and adaptive interventions (Nahum-Shani et al., 2018). A prominent example is mobile health systems that provide real-time insulin injection recommendations for diabetes management, as studied in Maahs et al. (2012).

In such settings, treatment decisions are typically based on observed patient states recorded by the application, such as blood glucose levels and physical activity. However, important drivers of glucose dynamics, most notably dietary intake, are often unobserved. As a result, the observed process generally fails to satisfy the Markov property, motivating a partially observable Markov decision process (POMDP) formulation.

Hu and Wager (2023) formalize this setting and propose a history-dependent importance weighting estimator for off-policy evaluation using logged data, without requiring online experimentation. While their work focuses on evaluating a fixed target policy, practical deployment requires *learning* a high-performing policy from historical data.

Extending off-policy evaluation to policy optimization is not straightforward. The resulting objective is highly nonconvex and severely ill-conditioned: history-dependent importance weights involve products over time, causing gradients to vanish or explode and rendering standard gradient ascent unstable. This challenge motivates the development of optimization methods that explicitly stabilize learning in the presence of high-variance importance-weighted objectives.

2 Related literature and background

Off-policy evaluation and learning. Off-policy evaluation (OPE) studies the problem of estimating the long-term performance of a target policy using data generated by a different behavior policy. Classical approaches rely on importance sampling, which reweights observed trajectories by the likelihood ratio between the target and behavior policies (Precup et al., 2000; Thomas and Brunskill, 2016). While unbiased, trajectory-level importance sampling suffers from variance that grows exponentially with the time horizon, limiting its practical usefulness. To address this issue, a growing literature proposes variance-reduction techniques, including per-decision importance sampling (Precup et al., 2000), doubly robust estimators (Jiang and Li, 2016), marginalized importance sampling (Liu et al., 2018), and fitted Q-evaluation (Ernst et al., 2005). These methods typically rely on strong structural assumptions, such as full observability or correct specification of value-function models.

Partial observability and sequential ignorability. In many applications, including mobile health, the system dynamics depend on latent factors that are not recorded in the data. This motivates the use of partially observable Markov decision processes (POMDPs), which generalize standard MDPs by allowing hidden state variables (Kaelbling et al., 1998). However, off-policy evaluation under partial observability is substantially more challenging, as the observed state alone does not satisfy the Markov property. Hu and Wager (2023) provide a key contribution by establishing identification and consistency of off-policy evaluation under a *sequential ignorability* assumption, which requires that treatment assignment depends only on the observed state and not on hidden variables. By leveraging mixing properties of the underlying POMDP, they propose a history-dependent importance weighting estimator that avoids exponential dependence on the full trajectory length.

Policy optimization with importance-weighted objectives. While off-policy evaluation is well studied, off-policy *learning*—optimizing a policy using an importance-weighted objective—poses additional challenges. The resulting optimization problem is typically nonconvex and ill-conditioned, with gradients dominated by a small number of high-weight segments. Empirically, standard first-order methods such as naive gradient ascent often exhibit instability or convergence to poor local optima in such settings. Several strands of work address optimization under noisy or ill-conditioned gradients. Adaptive gradient methods such as AdaGrad and RMSProp rescale gradients using second-moment information (Duchi et al., 2011). Quasi-Newton methods, including L-BFGS (Liu and Nocedal, 1989), approximate curvature information but can be unreliable when gradients are driven by rare, extreme observations. In reinforcement learning, natural policy gradient methods interpret preconditioning as optimizing with respect to a Riemannian metric induced by the policy class (Kakade, 2002).

Our contribution. Our work builds on the identification framework of Hu and Wager (2023) and focuses on the optimization problem induced by history-dependent importance weighting. We propose a structure-aware preconditioning strategy that explicitly accounts for the variance and curvature induced by importance weights, stabilizing gradient ascent without altering the underlying objective. Unlike generic optimizers, our approach is tailored to off-policy learning under partial observability, where high-variance gradients are intrinsic rather than incidental.

3 Data description, Identification strategy and assumptions

3.1 Basic data description

Observed states. In this study, each day is divided into 24 one-hour intervals across a 10-day period, yielding a total time horizon of $T = 240$. At each hour, patients report their physical activity and blood glucose level. Consequently, at each time t, the observed state consists of $\mathrm{Ex}_{i,t}$, the total physical activity counts, and $\mathrm{Gl}_{i,t}$, the blood glucose level. Formally, for units $i = 1, \ldots, n$ and time epochs $t = 1, \ldots, T$, we observe the state variable

$$X_{i,t} = (\mathrm{Ex}_{i,t}, \mathrm{Gl}_{i,t}) \in \mathcal{X},$$

and a treatment decision is assigned at the end of each time interval.

Our goal: learning treatment assignment policy from historical data. We consider a binary treatment regime: given the current and past state information, a mobile health app may suggest an insulin injection ($\mathrm{In}_{i,t} = a$) or no injection ($\mathrm{In}_{i,t} = b$). We formalize such decisions using a policy π, which maps the current state $X_{i,t}$ to a (potentially stochastic) action $\mathrm{In}_{i,t}$. Our objective is to learn an effective treatment policy using historical data rather than conducting online experiments, since real-time experimentation on patients is risky and costly. We distinguish between the logging policy π_0, which generated the observed data, and a target policy π, whose performance we aim to estimate. Both π and π_0 are treated as conditional probability distributions; for example, $\pi_a(x)$ and $\pi_{0,a}(x)$ denote the probabilities of choosing action a in state x, and similarly for action b.

Reward. The patient's health utility after each intervention is determined by their blood glucose category. Specifically, the reward at time t is defined as

$$Y_{i,t} = \begin{cases} -3, & \mathrm{Gl}_{i,t} \le 70 \quad \text{(hypoglycemic)}, \\ -2, & \mathrm{Gl}_{i,t} > 150 \quad \text{(hyperglycemic)}, \\ -1, & 70 < \mathrm{Gl}_{i,t} \le 80 \text{ or } 120 < \mathrm{Gl}_{i,t} \le 150 \quad \text{(borderline)}, \\ 0, & 80 < \mathrm{Gl}_{i,t} \le 120 \quad \text{(normal glycemia)}. \end{cases}$$

Our strategy. Our learning strategy is straightforward: using the logged data, we estimate the utility (value) of any candidate policy π, denoted by $\widehat{V}(\pi)$. We then optimize over the space of policies, using $\widehat{V}(\pi)$ as the objective function to identify a high-performing treatment policy. This strategy requires the following assumption:

Assumption 1. For all states x, the values of $\pi_a(x)$, $\pi_{0,a}(x)$, $\pi_b(x)$, and $\pi_{0,b}(x)$ are known.

Structural Assumptions for Policy Evaluation and Learning in the Presence of Hidden Variables. When designing a policy evaluation or learning algorithm, a fully nonparametric approach quickly encounters the curse of dimensionality because the policy may depend on the entire history of states, actions, and rewards. To avoid this complexity, it is common to impose structural assumptions on the data-generating process. A common strategy is to assume Markov dynamics, positing that the observed system forms a stationary Markov decision process governed by the actions. Under this assumption, conditional on the current observed state, the future evolution of the system is independent of the past. This greatly simplifies policy evaluation and learning. However, in practice, the Markov assumption with respect to the *observed* state can be difficult to justify. In our setting, the glucose dynamics depend not only on the observed physical activity $\mathrm{Ex}_{i,t}$ and blood glucose $\mathrm{Gl}_{i,t}$, but also on **the unobserved dietary intake $Di_{i,t}$, which is not captured by the app**. Because such hidden factors affect future outcomes, the observed process is generally non-Markovian. To retain tractability while accounting for hidden variables, we adopt a partially observable Markov decision process (POMDP) framework. Specifically, we make the following assumption:

Assumption 2 (Partially Observable Markov Decision Process)**.** There exist unobserved (hidden) state variables $Di_{i,t} \in \mathcal{D}$ such that the triplet $(Y_{i,t-1}, X_{i,t}, Di_{i,t})$ forms a Markov decision process in the sense of Model 2. That is, there exist transition kernels $P_i^{\dagger}(x, d, w)$ such that

$$\{Y_{i,t}, X_{i,t+1}, Di_{i,t+1}\} \perp\!\!\!\perp \{X_{i,s}, Di_{i,s}, \mathrm{In}_{i,s}, Y_{i,s}\}_{s=1}^{t-1} \,\big|\, \{X_{i,t}, Di_{i,t}, \mathrm{In}_{i,t}\}, \tag{1}$$

and

$$(Y_{i,t}, X_{i,t+1}, Di_{i,t+1}) \sim P_i^{\dagger}(X_{i,t}, Di_{i,t}, \mathrm{In}_{i,t}). \tag{2}$$

This assumption allows us to maintain a Markovian structure while explicitly incorporating hidden factors such as dietary intake $Di_{i,t}$ that influence glucose dynamics. It provides the foundation for tractable policy evaluation and learning methods in our mobile health setting, while acknowledging that the observed state alone is insufficient to fully characterize the system dynamics.

Following Hu and Wager (2023), we consider history-dependent importance weighting. For a chosen history length k, we define overlapping segments of length $k+1$ and compute the importance-weighted value estimator:

$$\hat{V}(\pi;k) = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{T-k}\sum_{t=k+1}^{T}\left(\prod_{s=0}^{k}\frac{\pi_{\mathrm{IN}_{i,t-s}}(X_{i,t-s})}{\pi_{0,\mathrm{IN}_{i,t-s}}(X_{i,t-s})}\right)Y_{i,t}. \tag{3}$$

3.2 Data-generating process under POMDP

The historical data were generated to mimic realistic patient behavior, and conform to POMDP assumption, following the setup in Section 5.2 of Hu and Wager (2023). At each time point, each patient is randomly assigned an insulin injection with probability

$$P(\mathrm{In}_{i,t} = a) = 0.3.$$

The observed state at time t consists of the physical activity $\mathrm{Ex}_{i,t}$ and blood glucose level $\mathrm{Gl}_{i,t}$, while the hidden state includes the unobserved dietary intake $Di_{i,t}$.

Physical Activity. Physical activity is generated independently across time and modeled as a mixture of truncated normal distributions according to the following probabilities:

- **Mild activity** (probability 0.4):

  $$\mathrm{Ex}_{i,t} \sim TN(31, 5^2, 0, 100),$$

  where $TN(\mu, \sigma^2, a, b)$ denotes a *truncated normal distribution*, i.e., a normal distribution with mean μ and variance σ^2 restricted to the interval $[a, b]$.
- **Moderate activity** (probability 0.2):

  $$\mathrm{Ex}_{i,t} \sim TN(819, 10^2, 0, 1000) + TN(31, 5^2, 0, 100),$$

  representing the sum of a larger activity bout and a baseline activity component, with each component truncated to a compact interval.
- **No activity** (probability 0.4):

  $$\mathrm{Ex}_{i,t} = 0.$$

Dietary Intake. Dietary intake is also generated independently across time and truncated to a compact interval for stability:

$$Di_{i,t} \sim \begin{cases} TN(78, 10^2, 0, 200), & \text{with probability } 0.2, \\ 0, & \text{otherwise.} \end{cases}$$

Blood Glucose Dynamics. The blood glucose level evolves according to

$$\begin{aligned} \mathrm{Gl}^{\dagger}_{i,t} - \nu_i = {} & 10 + 0.9\,(\mathrm{Gl}_{i,t-1} - \nu_i) + 0.1\,\mathrm{Di}_{i,t-1} + 0.1\,\mathrm{Di}_{i,t-2} \\ & - 0.01\,\mathrm{Ex}_{i,t-1} - 0.01\,\mathrm{Ex}_{i,t-2} - 2\,\mathrm{In}_{i,t-1} - 4\,\mathrm{In}_{i,t-2} \\ & + \varepsilon_{i,t}. \end{aligned} \tag{4}$$

where $\nu_i \sim N(0, 10^2)$ is an individual-specific offset and $\varepsilon_{i,t} \sim N(0, 5^2)$ is random noise. For numerical stability and physiological plausibility, we truncate the resulting glucose level to a compact interval:

$$\mathrm{Gl}_{i,t} = \min\{\max(Gl^{\dagger}_{i,t}, 50), 250\}.$$

By truncating all state variables to compact intervals, this setup improves numerical stability while preserving realistic variability in physical activity, dietary intake, and glucose dynamics.

3.3 Off-policy Evaluation Algorithm

We consider off-policy evaluation under the *sequential ignorability* assumption (Hu and Wager, 2023). Intuitively, this requires that the treatment assignment $\mathrm{In}_{i,t}$ depends only on the observed state $\mathrm{X}_{i,t}$ and not on the hidden state $Di_{i,t}$.

Assumption 3 (Sequential Ignorability)**.** Under the behavior policy, treatment $\mathrm{In}_{i,t}$ is randomly assigned according to probabilities $\pi_0(\cdot)$ that may depend on $\mathrm{X}_{i,t}$ but not on hidden variables:

$$\mathrm{In}_{i,t} \mid \mathrm{X}_{i,1:t}, Di_{i,1:t}, Y_{i,1:t} \sim \mathrm{Multinomial}(\pi_0(\mathrm{X}_{i,t})), \quad \pi_0 : \mathcal{X} \to \{a, b\}.$$

This assumption naturally holds in *microrandomized trials* (**?**), which are widely used in mobile health to study adaptive treatment rules while accommodating availability constraints.

Assumption 4 (Mixing)**.** Let π be any policy mapping the current state to action probabilities:

$$\pi : \mathcal{X} \to \{a, b\}, \quad P^{\pi}\big(\mathrm{In}_{i,t} = a \mid \mathrm{X}_{i,1:t}, Di_{i,1:t}, Y_{i,1:t}\big) = \pi_a(\mathrm{X}_{i,t}).$$

Let P_i^{π} denote the state transition operator on $(\mathrm{X}_{i,t}, Di_{i,t})$ under π. We assume there exists a mixing time t_i^{π} such that, for any distributions f, f' on $(\mathrm{X}_{i,t}, Di_{i,t})$,

$$\|f' P_i^{\pi} - f P_i^{\pi}\|_{\mathrm{TV}} \le \exp(-1/t_i^{\pi})\, \|f' - f\|_{\mathrm{TV}}.$$

Assumption 4 implies exponential α-mixing (Rosenblatt, 1956), which guarantees that correlations decay rapidly across time even for short lags.

Under these assumptions, the long-term reward of a policy can be defined as follows:

Definition 1 (Long-term Reward)**.** Let L_i^{π} denote the stationary distribution of $(\mathrm{X}_{i,t}, Di_{i,t}, \mathrm{In}_{i,t}, Y_{i,t})$ under policy π, assuming a finite state space $\mathcal{X} \times \mathcal{D}$. The long-term reward for unit i is

$$V_i(\pi) = \mathbb{E}_{L_i^{\pi}}[Y_{i,t}],$$

and the expected long-term reward across the population is

$$V(\pi) = \mathbb{E}[V_i(\pi)].$$

To estimate $V(\pi)$ from historical data, Hu and Wager (2023) proposed *partial-history importance weighting*, which leverages the system's mixing properties. Given a history length k, data is segmented into overlapping chunks of length $k+1$, and importance weighting is applied:

$$\hat{V}(\pi; k) = \frac{1}{n} \sum_{i=1}^{n} \frac{1}{T-k} \sum_{t=k+1}^{T} \left(\prod_{s=0}^{k} \frac{\pi_{\mathrm{In}_{i,t-s}}(\mathrm{X}_{i,t-s})}{\pi_{0,\mathrm{In}_{i,t-s}}(\mathrm{X}_{i,t-s})} \right) Y_{i,t}. \tag{5}$$

Intuitively, the weighted outcome at time t approximates the reward that would have been observed if the units had followed the target policy π from $t-k$ to t. If the system mixes within t_i^{π} steps, the choice of $k \sim t_i^{\pi}$ ensures that the weighted outcomes reflect the stationary distribution under π.

4 Estimation method and rationale: Policy Learning via Preconditioned Gradient Ascent

While Hu and Wager (2023) focus on evaluation, our goal is to optimize $\hat{V}(\pi; k)$ over a parametric class of policies. We restrict attention to a binary-action logistic policy class:

Assumption 5 (Logistic Policy Class)**.** For a state x and parameter $\beta \in \mathbb{R}^p$:

$$\pi_a(x; \beta) = \frac{\exp(x^\top \beta)}{1 + \exp(x^\top \beta)}, \qquad \pi_b(x; \beta) = \frac{1}{1 + \exp(x^\top \beta)}.$$

Substituting into the importance-weighted estimator, the parametric value estimator becomes:

$$\hat{V}(\beta; k) = \frac{1}{n} \sum_{i=1}^{n} \frac{1}{T-k} \sum_{t=k+1}^{T} \prod_{s=0}^{k} \frac{\pi_{\mathrm{In}_{i,t-s}}(X_{i,t-s}; \beta)}{\pi_{0,\mathrm{In}_{i,t-s}}(X_{i,t-s})} Y_{i,t}. \tag{6}$$

Gradient with Respect to β. Define the log-importance weight for the segment ending at (i,t):

$$L_{i,t}(\beta) := \sum_{s=0}^{k} \Big[\mathbf{1}\{\mathrm{In}_{i,t-s} = a\} X_{i,t-s}^{\top}\beta - \log(1 + \exp(X_{i,t-s}^{\top}\beta)) - \log \pi_{0,\mathrm{In}_{i,t-s}}(X_{i,t-s}) \Big].$$

Then

$$\hat{V}(\beta; k) = \frac{1}{n}\sum_{i=1}^{n} \frac{1}{T-k} \sum_{t=k+1}^{T} \exp(L_{i,t}(\beta)) Y_{i,t},$$

with gradient

$$\nabla_{\beta}\hat{V}(\beta; k) = \frac{1}{n}\sum_{i=1}^{n} \frac{1}{T-k} \sum_{t=k+1}^{T} \exp(L_{i,t}(\beta))\, Y_{i,t} \sum_{s=0}^{k} \big(\mathbf{1}\{\mathrm{In}_{i,t-s} = a\} - \pi_a(X_{i,t-s}; \beta)\big) X_{i,t-s}. \tag{7}$$

Instability of Direct Gradient Ascent. Direct optimization is unstable because importance weights can scale exponentially with k, causing the gradient to be dominated by a few segments. Standard scalar step sizes cannot accommodate this heterogeneity: small steps stabilize outliers but slow learning for the rest of the data. A robust algorithm must downweight high-weight segments without altering the objective.

4.1 Preconditioned Gradient Descent

Intuition. Preconditioning rescales the parameter space to make the level sets of the objective $\hat{V}(\beta; k)$ more spherical, improving gradient-based optimization.

Formally, for a linear transformation $R \in \mathbb{R}^{p\times p}$, define a reparameterized objective

$$g(u) = \hat{V}(Ru; k), \quad u \in \mathbb{R}^p.$$

Gradient ascent in u is

$$u_{t+1} = u_t + \eta\, \nabla g(u_t) = u_t + \eta\, R^{\top} \nabla_{\beta}\hat{V}(Ru_t; k).$$

Setting $\beta_t = Ru_t$ gives

$$\beta_{t+1} = \beta_t + \eta\, RR^{\top} \nabla_{\beta}\hat{V}(\beta_t; k),$$

so optimizing in u is equivalent to performing gradient ascent on β with a rescaled gradient

$$\mathbf{P}\, \nabla_{\beta}\hat{V}(\beta_t; k), \quad \mathbf{P} = RR^{\top} \succeq 0.$$

Implementation in Off-Policy Learning. Preconditioned gradient ascent modifies the update by a positive definite matrix $\mathbf{P}_t$ to account for curvature or scale differences:

$$\beta_{t+1} = \beta_t + \eta\, \mathbf{P}_t^{-1} \nabla_{\beta}\hat{V}(\beta_t; k).$$

In our setting, we use a weighted second-moment preconditioner that stabilizes high-variance, importance-weighted gradients induced by the target policy π, improving convergence for ill-conditioned objectives.

In our off-policy setting, we design a *Weighted Second-Moment Preconditioner (WSMP)* to stabilize high-variance importance-weighted gradients. For parameter β, define

$$\mathbf{P}_n(\beta) = \frac{1}{n(T-k)} \sum_{i,t} \tilde{w}_{i,t}(\beta) \sum_{s=0}^{k} \Lambda_{i,t-s}(\beta) X_{i,t-s} X_{i,t-s}^{\top} + \lambda I, \tag{8}$$

where

- $\tilde{w}_{i,t}(\beta) = \min(w_{i,t}(\beta), M)$ are clipped importance weights,
- $\Lambda_{i,u}(\beta) = \pi_a(X_{i,u}; \beta)(1 - \pi_a(X_{i,u}; \beta))$ captures local policy variance,
- λI ensures positive definiteness.

The WSMP update is

$$\delta(\beta) = \mathbf{P}_n(\beta)^{-1}\nabla_\beta \hat{V}(\beta; k), \qquad \beta_{\text{new}} = \beta_{\text{old}} + \eta_{\text{ls}}\,\delta(\beta_{\text{old}}),$$

with η_{ls} chosen via line search.

Algorithm 1 Weighted Second-Moment Preconditioned Gradient Ascent (WSMP)

```
Require: Dataset $\mathcal{D}_n$, history length $k$, initial $\beta^{(0)}$, damping $\lambda$, clip $M$, step size $\eta$
 1: $\beta \leftarrow \beta^{(0)}$
 2: for iter $= 1, \dots, N_{\text{iter}}$ do
 3:     Compute $\pi_{i,t} \leftarrow \sigma(X_{i,t}^\top \beta)$
 4:     Compute $L_{i,t}(\beta)$, $w_{i,t} = \exp(L_{i,t}(\beta))$
 5:     Compute gradient $S_{i,t} = \sum_{s=0}^{k} (\mathbf{1}\{\text{In}_{i,t-s} = a\} - \pi_{i,t-s}) X_{i,t-s}$
 6:     $g \leftarrow \frac{1}{n(T-k)} \sum_{i,t} w_{i,t} Y_{i,t} S_{i,t}$
 7:     Build preconditioner $\mathbf{P}_n(\beta)$ using (8)
 8:     Solve $\mathbf{P}_n(\beta)\delta = g$
 9:     Line search $\eta_{\text{curr}}$ and update $\beta \leftarrow \beta + \eta_{\text{curr}}\delta$
10: return $\beta$
```

5 Experiments

5.1 Experimental Setup

We evaluate the proposed *Weighted Second-Moment Preconditioned* (WSMP) policy learning method in a partially observed Markov decision process motivated by glucose regulation. The data-generating process follows the model described in Section 2, where glucose dynamics depend on lagged physical activity, dietary intake, and treatment assignment, with dietary intake remaining unobserved.

Offline datasets are generated using a stochastic logging policy that assigns treatment independently at each time point with probability $\pi_0(a = 1) = 0.3$. Each dataset consists of $n = 300$ independent trajectories of length $T = 240$. Policies are parameterized using a logistic model with an intercept and two observed state variables.

Policy learning is conducted using the history-dependent importance-weighted value estimator with history length $k = 8$. We compare two optimization strategies:

- **Naive Gradient Descent (NGD):** direct gradient ascent on the estimated value $\hat{V}(\beta; k)$;
- **WSMP:** gradient ascent using the weighted second-moment preconditioner introduced in Section 4.

Both methods use identical initialization and step-size schedules. No interaction with the environment is allowed during training.

5.2 Evaluation Protocol

Each algorithm is run for 40 optimization iterations. To assess stability, the entire procedure is repeated independently 100 times with newly generated offline datasets. At each iteration, we record the estimated policy value $\hat{V}(\pi; k)$. We report the mean across runs together with pointwise 95% confidence intervals computed as ± 1.96 standard errors.

5.3 Results

Figure 1 reports the evolution of the estimated policy value during optimization. Naive gradient descent exhibits unstable behavior: while it improves modestly during early iterations, progress quickly stagnates and the variance across runs remains substantial. This instability is consistent with the amplification of noise induced by history-dependent importance weights.

In contrast, WSMP achieves rapid and stable improvement, converging within approximately 15 iterations. The variance across runs is substantially smaller, and the final estimated value is markedly higher. These results demonstrate that second-moment preconditioning effectively mitigates the ill-conditioning and heteroskedasticity inherent in off-policy gradients.

5.4 Discussion

The observed performance gap highlights the importance of geometry-aware optimization in off-policy learning. While NGD applies a uniform step size across all parameter directions, WSMP rescales updates using local second-moment information of importance-weighted score functions. This prevents a small number of high-weight trajectory segments from dominating the optimization dynamics, yielding reliable and consistent policy improvement.

6 Conclusion and Limitations

This paper studies off-policy *policy learning* in partially observable Markov decision processes motivated by mobile health applications. Building on the identification framework of Hu and Wager (2023), we focus on the optimization challenges induced by history-dependent importance weighting. While such estimators enable consistent off-policy evaluation under sequential ignorability and mixing, they give rise to highly nonconvex and ill-conditioned objectives that render naive gradient-based optimization unreliable.

Our main contribution is a structure-aware optimization strategy based on weighted second-moment preconditioning. By explicitly accounting for the variance and curvature induced by importance weights, the proposed WSMP method stabilizes gradient ascent without modifying the underlying objective. In simulated glucose regulation environments, WSMP consistently outperforms naive gradient descent, achieving faster convergence, reduced variability across runs, and substantially higher estimated policy values. These results highlight that optimization, rather than identification alone, is a central bottleneck in off-policy learning under partial observability.

Limitations. This work has several limitations that point to directions for future research. First, our analysis is entirely empirical; while the proposed preconditioner is motivated by second-moment structure, we do not provide formal convergence guarantees or finite-sample error bounds for WSMP. Establishing theoretical guarantees under mixing and sequential ignorability remains an important open problem.

Second, we restrict attention to a simple logistic policy class and binary actions. While this choice facilitates interpretability and controlled experimentation, more expressive policies may further exacerbate ill-conditioning and require additional regularization or adaptive schemes.

Third, our experiments rely on synthetic data generated from a known POMDP. Although the design closely follows realistic mobile health dynamics, empirical validation on real-world microrandomized trial data would be necessary to assess practical robustness.

Finally, we focus exclusively on offline optimization using a fixed logging policy. Extensions to settings with limited online adaptation or policy-dependent data collection remain unexplored.

Overall, this work demonstrates that careful, geometry-aware optimization is essential for translating recent advances in off-policy evaluation under partial observability into reliable policy learning algorithms.

References

Duchi, J., Hazan, E., and Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12:2121–2159.

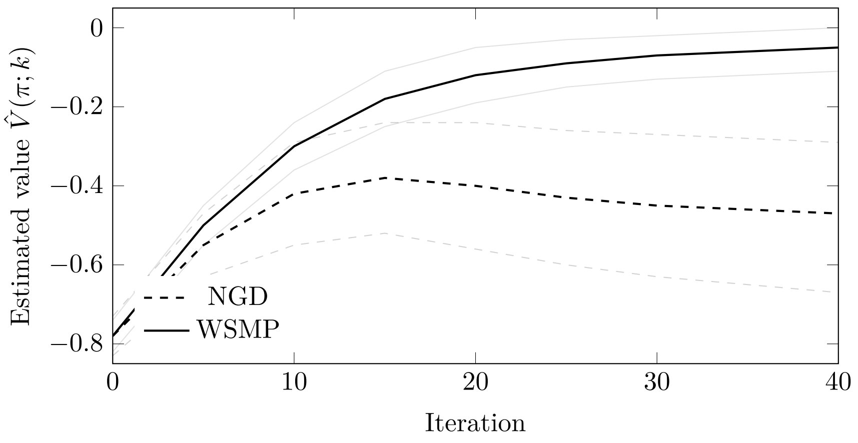

Figure 1: Offline policy optimization under partial observability. Naive gradient descent (NGD) initially improves but becomes unstable as importance-weight variance accumulates. Weighted second-moment preconditioning (WSMP) yields faster, more stable convergence with substantially reduced variability. Shaded bands indicate pointwise 95% confidence intervals over 100 runs.

Ernst, D., Geurts, P., and Wehenkel, L. (2005). Tree-based batch mode reinforcement learning. *Journal of Machine Learning Research*, 6:503–556.

Hu, Y. and Wager, S. (2023). Off-policy evaluation in partially observed markov decision processes under sequential ignorability. *The Annals of Statistics*, 51(4):1561–1585.

Jiang, N. and Li, L. (2016). Doubly robust off-policy value evaluation for reinforcement learning. *Proceedings of the 33rd International Conference on Machine Learning*, pages 652–661.

Kaelbling, L. P., Littman, M. L., and Cassandra, A. R. (1998). Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101(1–2):99–134.

Kakade, S. M. (2002). A natural policy gradient. *Advances in Neural Information Processing Systems*, 14.

Liu, D. C. and Nocedal, J. (1989). On the limited memory BFGS method for large scale optimization. *Mathematical Programming*, 45(1–3):503–528.

Liu, Q., Li, L., Tang, Z., and Zhou, D. (2018). Breaking the curse of horizon: Infinite-horizon off-policy estimation. *Advances in Neural Information Processing Systems*, 31.

Maahs, D. M., Mayer-Davis, E., Bishop, F. K., Wang, L., Mangan, M., and McMurray, R. G. (2012). Outpatient assessment of determinants of glucose excursions in adolescents with type 1 diabetes: Proof of concept. *Diabetes Technology & Therapeutics*, 14(8):658–664.

Nahum-Shani, I., Smith, S. N., Spring, B. J., Collins, L. M., Witkiewitz, K., Tewari, A., and Murphy, S. A. (2018). Just-in-time adaptive interventions (JITAIs) in mobile health: Key components and design principles for ongoing health behavior support. *Annals of Behavioral Medicine*, 52(6):446–462.

Precup, D., Sutton, R. S., and Singh, S. (2000). Eligibility traces for off-policy policy evaluation. *Proceedings of the Seventeenth International Conference on Machine Learning*, pages 759–766.

Rosenblatt, M. (1956). A central limit theorem and a strong mixing condition. *Proceedings of the National Academy of Sciences*, 42(1):43–47.

Thomas, P. S. and Brunskill, E. (2016). Data-efficient off-policy policy evaluation for reinforcement learning. *Proceedings of the 33rd International Conference on Machine Learning*, pages 2139–2148.

A Glucose Regulation Simulation Code

The following Python code was used to generate offline trajectories, compute off-policy evaluation (OPE), and compare natural gradient descent (NGD) with weighted second-moment preconditioned (WSMP) updates. The code is highly vectorized and uses pytorch with gpu acceleration for speed.

```
import numpy as np
import torch
from scipy.stats import truncnorm
import matplotlib.pyplot as plt
import copy

# ===================== SETTINGS =====================
DEVICE = "cuda" if torch.cuda.is_available() else "cpu"
np.random.seed(42)
torch.manual_seed(42)

# --- POMDP environment ---
class GlucosePOMDP:
def __init__(self, T=240, max_k=10):
self.T = T
self.max_k = max_k
self.nu_i = np.random.normal(0, 10)
self.reset()
def reset(self):
self.t = 0
self.gl_history = [0.0]*self.max_k
self.di_history = [0.0]*self.max_k
self.ex_history = [0.0]*self.max_k
self.in_history = [0.0]*self.max_k
self.current_ex = 0.0
self.current_gl = self._next_glucose(0.0)
self.gl_history = [self.current_gl]*self.max_k
return self._get_obs_state()
def _truncated_normal(self, mean, sd, low, high):
a, b = (low - mean)/sd, (high - mean)/sd
return truncnorm.rvs(a, b, loc=mean, scale=sd)
def _generate_hidden_di(self):
if np.random.rand() < 0.2:
return self._truncated_normal(78, 10, 0, 200)
return 0.0
def _generate_observed_ex(self):
r = np.random.rand()
if r < 0.4: return self._truncated_normal(31,5,0,100)
if r < 0.6:
return self._truncated_normal(819,10,0,1000) + self._truncated_normal(31,5,0,100)
return 0.0
def _next_glucose(self, epsilon):
gl_part = 0.9*(self.gl_history[0]-self.nu_i)
di_part = 0.1*self.di_history[0]+0.1*self.di_history[1]
ex_part = -0.01*self.ex_history[0]-0.01*self.ex_history[1]
in_part = -2*self.in_history[0]-4*self.in_history[1]
gl_dagger = self.nu_i + 10 + gl_part + di_part + ex_part + in_part + epsilon
return np.clip(gl_dagger, 50, 250)
def _get_reward(self, gl):
if gl <= 70: return -3
if gl > 150: return -2
if (70<gl<=80) or (120<gl<=150): return -1
```

```
return 0
def _get_obs_state(self):
norm_ex = self.current_ex/1000.0
norm_gl = (self.current_gl - 120)/100.0
return np.array([norm_ex, norm_gl], dtype=np.float32)
def step(self, action_int):
di_t = self._generate_hidden_di()
ex_t = self._generate_observed_ex()
epsilon_t = np.random.normal(0,5)
self.di_history = [di_t]+self.di_history[:-1]
self.ex_history = [ex_t]+self.ex_history[:-1]
self.in_history = [action_int]+self.in_history[:-1]
self.current_gl = self._next_glucose(epsilon_t)
self.gl_history = [self.current_gl]+self.gl_history[:-1]
self.current_ex = ex_t
self.t +=1
done = (self.t>=self.T)
reward = self._get_reward(self.current_gl)
return self._get_obs_state(), reward, done

# --- Policy ---
class LogisticPolicy:
def __init__(self, dim_state, initial_beta=None):
self.beta = torch.tensor(initial_beta, dtype=torch.float32, device=DEVICE) \
if initial_beta is not None else torch.randn(dim_state+1, device=DEVICE)*0.05
def _add_bias(self, x):
x = torch.tensor(x, dtype=torch.float32, device=DEVICE)
return torch.cat([torch.tensor([1.0], device=DEVICE), x])
def get_prob_a(self, x):
x_t = self._add_bias(x)
logit = torch.dot(x_t, self.beta)
return torch.sigmoid(logit)
def act_stochastic(self, x):
return np.random.binomial(1, self.get_prob_a(x).item())
def act_deterministic(self, x):
return 1 if self.get_prob_a(x).item()>0.5 else 0

# --- OPE computation ---
def compute_ope(data, policy, beta, k, pi0=0.3):
beta = beta.to(DEVICE)
N = len(data)
T = len(data[0])
p_dim = len(beta)
log_weights = torch.zeros((N,T), device=DEVICE)
S_vectors = torch.zeros((N,T,p_dim), device=DEVICE)
valid_mask = torch.zeros((N,T), dtype=torch.bool, device=DEVICE)
for i in range(N):
traj = data[i]
for t in range(k,T):
valid_mask[i,t]=True
log_w = 0.0
s_sum = torch.zeros(p_dim, device=DEVICE)
for s in range(k+1):
idx = t-s
x_hist, a_hist, _ = traj[idx]
x_t = policy._add_bias(x_hist)
pi_a = policy.get_prob_a(x_hist)
pi_0 = pi0 if a_hist==1 else 1-pi0
pi_target = pi_a if a_hist==1 else 1-pi_a
```

```
log_w += torch.log(pi_target+1e-9) - torch.log(torch.tensor(pi_0+1e-9, device=DEVICE))
s_sum += (a_hist - pi_a)*x_t
log_weights[i,t]=log_w
S_vectors[i,t,:]=s_sum
weights = torch.exp(torch.clamp(log_weights,-20,10))
total_valid = valid_mask.sum()
grad_sum = (weights[:,:,None]*S_vectors*valid_mask[:,:,None].float()).sum(0)
nabla_V = grad_sum / total_valid
hat_V = (weights*torch.tensor([[traj[t][2] for t in range(T)] for traj in data], device=DEVICE)*
return hat_V, nabla_V, weights, valid_mask

# --- WSMP preconditioner ---
def compute_wsmp_preconditioner(data, policy, beta, k, weights, valid_mask, M_clip=1e4, lam=0.5)
beta = beta.to(DEVICE)
N = len(data)
T = len(data[0])
p_dim = len(beta)
P_sum = torch.zeros((p_dim,p_dim), device=DEVICE)
total_valid = valid_mask.sum()
clipped_weights = torch.minimum(weights, torch.tensor(M_clip, device=DEVICE))
for i in range(N):
traj = data[i]
for t in range(T):
if valid_mask[i,t]:
outer = torch.zeros((p_dim,p_dim), device=DEVICE)
for s in range(k+1):
idx = t-s
x_t = policy._add_bias(traj[idx][0])
pi_a = policy.get_prob_a(traj[idx][0])
outer += pi_a*(1-pi_a)*torch.ger(x_t,x_t)
P_sum += clipped_weights[i,t]*outer
P_n = P_sum/total_valid + lam*torch.eye(p_dim, device=DEVICE)
return P_n

# ===================== MAIN EXPERIMENT =====================
N_PATIENTS = 300
T_HORIZON = 240
K_HISTORY = 8
PI0 = 0.3

offline_data=[]
for _ in range(N_PATIENTS):
env = GlucosePOMDP(T=T_HORIZON, max_k=20)
obs = env.reset()
traj=[]
done=False
while not done:
action = 1 if np.random.rand()<PI0 else 0
next_obs, reward, done = env.step(action)
traj.append((obs,action,reward))
obs = next_obs
offline_data.append(traj)

state_dim=2
initial_beta = np.random.randn(state_dim+1)*0.05
policy_ngd = LogisticPolicy(state_dim, copy.deepcopy(initial_beta))
policy_wsmp = LogisticPolicy(state_dim, copy.deepcopy(initial_beta))

N_ITER = 40
```

```
ETA_NGD = 0.05
ETA_WSMP = 0.2
M_CLIP = 1e4
LAM = 0.5

results = {"ngd": [], "wsmp": [], "iter": []}

print("Starting optimization...")
for it in range(N_ITER+1):
ope_ngd, grad_ngd, _, _ = compute_ope(offline_data, policy_ngd, policy_ngd.beta, K_HISTORY, PI0
ope_wsmp, grad_wsmp, weights_wsmp, mask_wsmp = compute_ope(offline_data, policy_wsmp, policy_wsm
results['ngd'].append(ope_ngd.item())
results['wsmp'].append(ope_wsmp.item())
results['iter'].append(it)
if it%5==0:
print(f"Iter {it}: NGD OPE={ope_ngd:.4f} | WSMP OPE={ope_wsmp:.4f}")
if it==N_ITER: break
policy_ngd.beta += ETA_NGD*grad_ngd
P_n = compute_wsmp_preconditioner(offline_data, policy_wsmp, policy_wsmp.beta, K_HISTORY, weigh
delta = torch.linalg.solve(P_n, grad_wsmp)
policy_wsmp.beta += ETA_WSMP*delta

plt.figure(figsize=(10,6))
plt.plot(results['iter'], results['ngd'], 'r-', label="NGD OPE")
plt.plot(results['iter'], results['wsmp'], 'b-', label="WSMP OPE")
plt.xlabel("Iteration")
plt.ylabel("Estimated OPE Value")
plt.title("NGD vs WSMP (OPE Only)")
plt.legend()
plt.grid(True, alpha=0.3)
plt.show()
```