Uncertainty Quantification and Subsequential Decision-making

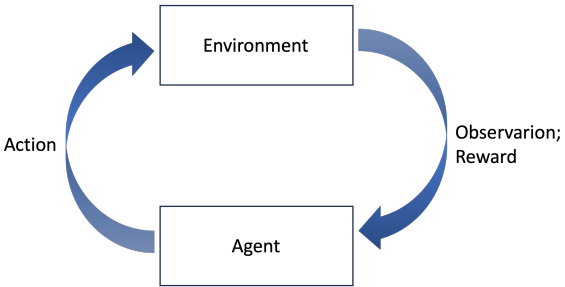
Posteiror Sampling beyond Conjugacy

Yingru Li and Zhi-Quan (Tom) Luo

The Chinese University of Hong Kong
Shenzhen Research Institute of Big Data
National Research Institute of Health Data, Shenzhen, China

September 23, 2023

Reinforcement Learning (AI agents)



Sequential decision-making under **uncertainty**.

The probability problems involved are formidable ... [but] the theory of **sequential design** will be of the greatest importance to **mathematical statistics** and to **science** ... – **Robins 1952**

Huge Increase in Interest

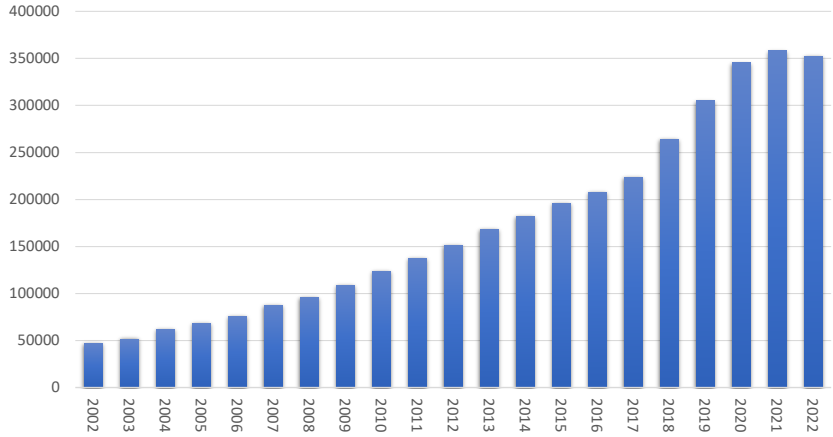


Figure: Growth of published "reinforcement learning" related papers and articles. (Data from Semantic Scholar.)

Application: Learning Plasma Control for Fusion Science

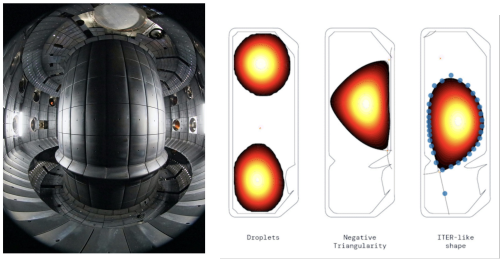


Figure: Image credits: left Alain Herzog / EPFL, right DeepMind & SPC/EPFL. Degrave et al. Nature 2022. "Magnetic control of tokamak plasmas through deep reinforcement learning."

- Action: Control Coil Voltages
- Observation: Physical measurements
- Uncertainty: Unknown physical dynamics in real system
- Goal
  1. Keep the Plasma Alive;
  2. Stabilize the plasma location;
  3. Shape Control.
- Ultimate Goal: **Sustainable Fusion Energy**

Application: Chip design

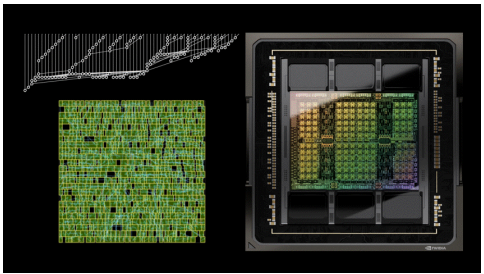


Figure: Designing Arithmetic Circuits with Deep Reinforcement Learning, Nvidia. https://developer.nvidia.com/blog/designing-arithmetic-circuits-with-deep-reinforcement-learning/

- Action: Sequentially modify the circuit design
- Observation: A graph representing circuit
- Uncertainty: Unknown circuit delay/area (Use circuit synthesis as simulator to gain information)
- Goal
  1. **Small**: A lower area so that more circuits can fit on a chip.
  2. **Fast**: A lower delay to improve the performance of the chip.
  3. **Energy-saving**: A lower power consumption of the chip.

Application: Ride-hailing Order Dispatching (Matching)

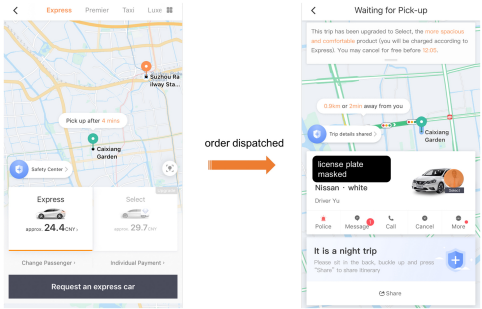


Figure: Ride-hailing Order Dispatching on DiDi via Reinforcement Learning. 2019 Informs Wagner Prize Winner; 2022 KDD paper for real-world deployment.

- Action: match user and driver
- Observation: orders and completed traffic routes
- Uncertainty: unknown traffic dynamics and user behaviors
- Goal
  1. Maximize **driver income**
  2. Maximize **order completion rate**
- Ultimate goal: **efficient urban mobility and shared economy**

Outline

Background and motivations

Adaptation: Hypermodel for uncertainty estimation

Decision: Hypermodel-based index sampling

Sequential decision-making

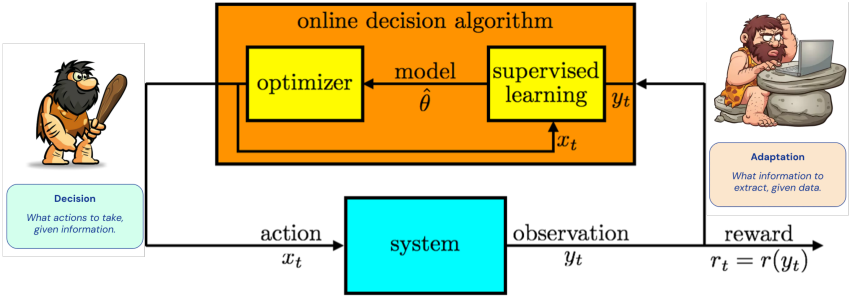


Figure: An **AI agent (online decision algorithm)** interacts with **the environment (system)**.

- **Adaptation**: At time t, **the agent** extracts information from history data $D_{t-1} = (x_1, y_1, \ldots, x_{t-1}, y_{t-1})$. E.g., estimate model $\hat{\theta}$ for unknown system.
- **Decision**: Then, **the agent** selects action x_t accordingly and observes the outcome y_t.

Sequential decision-making

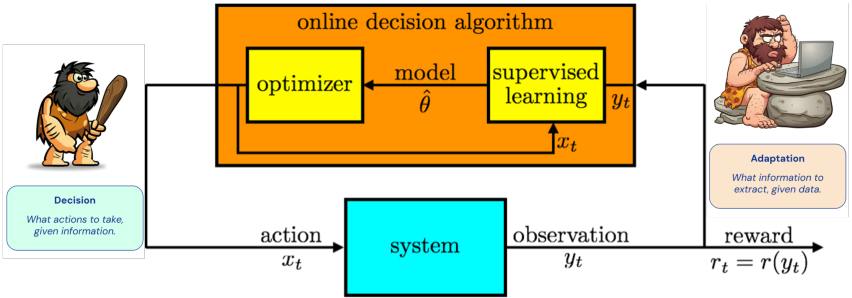


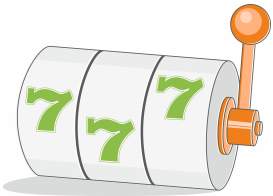
Figure: An **AI agent (online decision algorithm)** interacts with **the environment (system)**.

- **Goal**: Select actions $(x_t)_{t \geqslant 1}$ to maximize total expected future reward $\mathbb{E}[\sum_t r(y_t))]$.

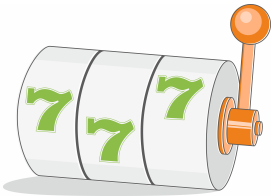
Exploration-Exploitation tradeoff.

May require balancing long term & immediate rewards.

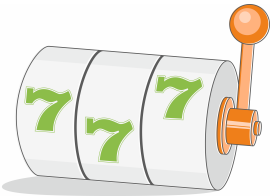
A simple setup: Bernoulli bandits



(a) Action 1: $\theta_1^* = 0.6$



(b) Action 2: $\theta_2^* = 0.4$



(c) Action 3: $\theta_3^* = 0.7$

- 3 actions with mean rewards $\theta^* = \{\theta_1^* = 0.6, \theta_2^* = 0.4, \theta_3^* = 0.7\}$, unknown to the AI agent but fixed.
- Each time t, an action $x_t = k$ is selected and the observation

$$y_t \sim \text{Bernoulli}(\theta_k^*)$$

is revealed, resulting the reward $r_t = y_t$.

Source of uncertainty: unknown rewards and insufficient data

- $\theta^* = \{\theta_1^* = 0.6, \theta_2^* = 0.4, \theta_3^* = 0.7\}$ unknown.
- The agent begin with an independent uniform prior belief over each θ_k^*.
- The agent's beliefs in any given time period about these mean rewards can be expressed in terms of posterior distributions.
  - **Posterior** $\propto$ **Prior** $\times$ **Data likelihoods**
  - More Data $\Rightarrow$ Posterior concentrates!
  - Less Data $\Rightarrow$ Posterior spreads!

Epistemic Uncertainty due to **insufficient** data.

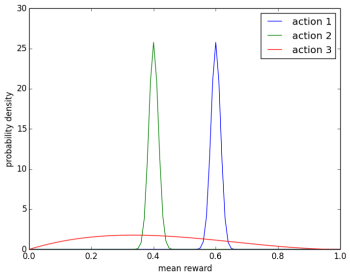


Figure: Posterior p.d.f. over mean rewards after the agent tries **actions 1 and 2 one thousand times each, action 3 three times**, receives cumulative rewards of 600, 400, and 1.

Why AI agent needs to track the degree of uncertainty: Greed is no good

- **Greedy** algorithm (maximize expected mean reward with current belief) will always select action 1.
- Under current belief: Reasonable to avoid action 2, since it is extremely unlikely $\theta_2^* > \theta_1^*$.
- Because of high uncertainty in θ_3^*, there is some chance $\theta_3^* > \theta_1^*$. In the long run, the agent should try action 3.

Greedy algorithm fails to account for uncertainty information in θ_3^*, causing suboptimal decision.

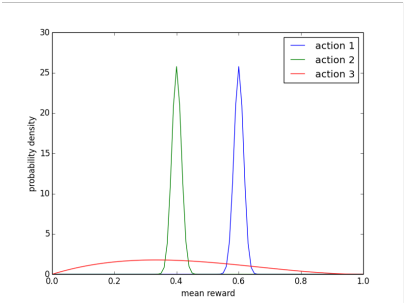


Figure: Posterior p.d.f. over mean rewards after the agent tries **actions 1 and 2 one thousand times each, action 3 three times**, receives cumulative rewards of 600, 400, and 1. Ground truth $\{\theta_1^* = 0.6, \theta_2^* = 0.4, \theta_3^* = 0.7\}$.

Why AI agent needs to track the degree of uncertainty: Thompson sampling

Algorithm: Thompson sampling (TS)

- Given prior distribution $p_0(\theta^*)$ over model θ^*. Set initial dataset $D_0 = \emptyset$.
- For $t = 1, \ldots, T$,
  - **Sample** $\tilde{\theta}_t \sim p(\theta^* \mid D_{t-1})$ from posterior
  - **Select** $x_t = \arg\max_{x \in \mathcal{A}} \mathbb{E}[r(y_t) \mid x_t = x, \theta^* = \tilde{\theta}_t]$ and **observe** y_t and $r_t = r(y_t)$
  - **Update** the history dataset $D_t = D_{t-1} \cup \{(x_t, y_t)\}$

- TS would sample actions 1, 2, or 3, with prob. $\approx$ 0.82, 0, and 0.18, respectively.
- TS explores θ_3^* to solve its uncertainty and finally identifies the optimal action

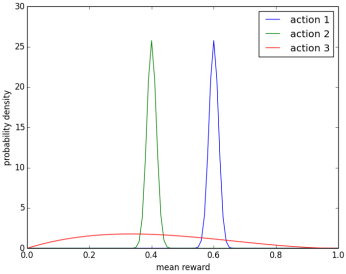


Figure: Posterior p.d.f. over mean rewards after the agent tries **actions 1 and 2 one thousand times each, action 3 three times**, receives cumulative rewards of 600, 400, and 1. Ground truth $\{\theta_1^* = 0.6, \theta_2^* = 0.4, \theta_3^* = 0.7\}$.

Why AI agent needs to track the degree of uncertainty

Definition 1 (Performance metric: Regret).

$$Regret(T) = \sum_{t=1}^{T} \mathbb{E}[\max_{x} \mathbb{E}[r(y) \mid x, \theta^*] - r(y_t)]$$

In previous bernoulli bandit example, $\theta_1^* = 0.6, \theta_2^* = 0.4, \theta_3^* = 0.7$ and

$$\max_{x} \mathbb{E}[r(y) \mid x, \theta^*] = \theta_3^*.$$

Therefore, $\text{Regret}(T) = T\theta_3^* - \mathbb{E}[\sum_{t=1}^{T} r(y_t)]$.

Why AI agent needs to track the degree of uncertainty: Thompson sampling

Algorithm: Thompson sampling (TS)

- Given prior distribution $p_0(\theta^*)$ over model θ^*. Set initial dataset $D_0 = \emptyset$.
- For $t = 1, \ldots, T$,
  - **Sample** $\tilde{\theta}_t \sim p(\theta^* \mid D_{t-1})$ from posterior
  - **Select** $x_t = \arg\max_{x \in \mathcal{A}} \mathbb{E}[r(y_t) \mid x_t = x, \theta^* = \tilde{\theta}_t]$ and **observe** y_t and $r_t = r(y_t)$
  - **Update** the history dataset $D_t = D_{t-1} \cup \{(x_t, y_t)\}$

Theorem 1 (Thompson sampling for K-armed bandit).

*K actions with mean parameter $\{\theta_1^*, \ldots, \theta_K^*\}$, and when played, any action yields the observation $y_t \sim \text{Bernoulli}(\theta_k^*)$ and resulting the reward $r_t = r(y_t)$. The regret **lower bound** is $\Omega(\sqrt{KT})$. Thompson sampling achieves near-optimal regret up to a $\log K$ factor,*

$$\textit{Regret}(T) = O(\sqrt{KT \log K}).$$

Adaptation: Conjugacy property makes posterior computation tractable

Given data $D_T = \{(x_t, y_t), t = 1, \ldots, T\}$, compute the posterior of θ^* by the **Bayes rule**

$$p(\theta^* \mid D_T) \propto p(D_T \mid \theta^*) p_0(\theta^*)$$

Example: Beta-Bernoulli model

- Prior: $\theta^* \in \mathbb{R}^K$ each $\theta_k \sim p_0 : \text{Beta}(\alpha_k, \beta_k)$
- $y_t \sim \text{Bernoulli}(\theta_{x_t})$
- Posterior over $\theta_k \mid D_T$ still Beta with parameters

$$\left(\alpha_k + \sum_{t=1}^{T} y_t \mathbb{I}_{x_t = k}, \beta_k + \sum_{t=1}^{T} (1 - y_t) \mathbb{I}_{x_t = k} \right)$$

Example: Linear-Gaussian model

- Prior: $\theta^* \in \mathbb{R}^d \sim p_0 : N(\mu_0, \Sigma_0)$
- $y_t = \langle \theta^*, x_t \rangle + \omega_t^*$ and $\omega_t^* \sim N(0, \sigma^2)$
- Gaussian Posterior $\theta^* \mid D_T \sim N(\mu_T, \Sigma_T)$

$$\Sigma_T = \left(\frac{1}{\sigma^2} \sum_{t=1}^{T} x_t x_t^\top + \Sigma_0^{-1} \right)^{-1},$$

$$\mu_T = \Sigma_T \left(\frac{1}{\sigma^2} \sum_{t=1}^{T} x_t y_t + \Sigma_0^{-1} \mu_0 \right).$$

Adaptation: Conjugacy property makes posterior computation tractable

Given data $D_T = \{(x_t, y_t), t = 1, \ldots, T\}$, compute the posterior of θ^* by the **Bayes rule**

$$p(\theta^* \mid D_T) \propto p(D_T \mid \theta^*) p_0(\theta^*)$$

Example: Beta-Bernoulli model

- Prior: $\theta^* \in \mathbb{R}^K$ each $\theta_k \sim p_0 : \text{Beta}(\alpha_k, \beta_k)$
- $y_t \sim \text{Bernoulli}(\theta_{x_t})$
- Posterior over $\theta_k \mid D_T$ still Beta with parameters

$$\left(\alpha_k + \sum_{t=1}^{T} y_t \mathbb{I}_{x_t = k}, \beta_k + \sum_{t=1}^{T} (1 - y_t) \mathbb{I}_{x_t = k} \right)$$

Example: Linear-Gaussian model

- Prior: $\theta^* \in \mathbb{R}^d \sim p_0 : N(\mu_0, \Sigma_0)$
- $y_t = \langle \theta^*, x_t \rangle + \omega_t^*$ and $\omega_t^* \sim N(0, \sigma^2)$
- Gaussian Posterior $\theta^* \mid D_T \sim N(\mu_T, \Sigma_T)$

$$\Sigma_T = \left(\frac{1}{\sigma^2} \sum_{t=1}^{T} x_t x_t^\top + \Sigma_0^{-1} \right)^{-1},$$

$$\mu_T = \Sigma_T \left(\frac{1}{\sigma^2} \sum_{t=1}^{T} x_t y_t + \Sigma_0^{-1} \mu_0 \right).$$

Adaptation: Conjugacy property makes posterior computation tractable

Given data $D_T = \{(x_t, y_t), t = 1, \dots, T\}$, compute the posterior of θ^* by the **Bayes rule**

$$p(\theta^* \mid D_T) \propto p(D_T \mid \theta^*) p_0(\theta^*)$$

Example: Beta-Bernoulli model

- Prior: $\theta^* \in \mathbb{R}^K$ each $\theta_k \sim p_0 : \text{Beta}(\alpha_k, \beta_k)$
- $y_t \sim \text{Bernoulli}(\theta_{x_t})$
- Posterior over $\theta_k \mid D_T$ still Beta with parameters

$$\left(\alpha_k + \sum_{t=1}^{T} y_t \mathbb{I}_{x_t = k}, \beta_k + \sum_{t=1}^{T} (1 - y_t) \mathbb{I}_{x_t = k} \right)$$

Example: Linear-Gaussian model

- Prior: $\theta^* \in \mathbb{R}^d \sim p_0 : N(\mu_0, \mathbf{\Sigma}_0)$
- $y_t = \langle \theta^*, x_t \rangle + \omega_t^*$ and $\omega_t^* \sim N(0, \sigma^2)$
- Gaussian Posterior $\theta^* \mid D_T \sim N(\mu_T, \mathbf{\Sigma}_T)$

$$\mathbf{\Sigma}_T = \left(\frac{1}{\sigma^2} \sum_{t=1}^{T} x_t x_t^\top + \Sigma_0^{-1} \right)^{-1},$$

$$\mu_T = \mathbf{\Sigma}_T \left(\frac{1}{\sigma^2} \sum_{t=1}^{T} x_t y_t + \Sigma_0^{-1} \mu_0 \right).$$

Adaptation: Conjugacy property allows incremental posterior update

- Update $\boldsymbol{\Sigma}_t$ by Sherman-Morrison formula

$$\boldsymbol{\Sigma}_t = \left(\boldsymbol{\Sigma}_{t-1}^{-1} + \frac{1}{\sigma^2} x_t x_t^\top\right)^{-1} = \boldsymbol{\Sigma}_{t-1} - \frac{\boldsymbol{\Sigma}_{t-1} x_t x_t^\top \boldsymbol{\Sigma}_{t-1}}{\sigma^2 + x_t^\top \boldsymbol{\Sigma}_{t-1} x_t}$$

- (Incrementally) Update $p_t := \Sigma_t^{-1}\mu_t$ with

$$\underbrace{\boldsymbol{\Sigma}_t^{-1}\mu_t}_{p_t} = \underbrace{\boldsymbol{\Sigma}_{t-1}^{-1}\mu_{t-1}}_{p_{t-1}} + \frac{1}{\sigma^2} x_t y_t \tag{1}$$

- Compute μ_t

$$\mu_t = \boldsymbol{\Sigma}_t p_t$$

Research question: What if no conjugacy

Fact: Without conjugacy properties, exact Bayesian posterior inference is intractable

Questions:

- **Adaptation**: How to track posterior efficiently in complex environments where no conjugacy could be exploited? E.g. reward model is nonlinear.
- **Decision**: How does the agent select the action given the posterior?

General idea: Calls for approximation!

Exisiting solution: Ensemble models and ensmeble sampling

- **Ensemble models** are a collection of models $\{\tilde{\theta}_1, \ldots, \tilde{\theta}_M\}$, each of which is a R.V.

Ensmeble sampling

- Initialize each m-th model $\tilde{\theta}_{0,m} \sim N(\mu_0, \Sigma_0)$ independently for $m \in \{1, \ldots, M\}$
- For $t = 1, \ldots, T$ do
  - **Decision**: Sample $m \sim \text{unif}\{1, \ldots, M\}$. Select $x_t = \arg\max_{x \in \mathcal{A}} x^\top \tilde{\theta}_{t-1,m}$ and observe y_t
  - **Adaptation**: Incrementally update each m-th model according to

$$\tilde{\theta}_{t,m} = \boldsymbol{\Sigma}_t \left(\boldsymbol{\Sigma}_{t-1}^{-1} \tilde{\theta}_{t-1,m} + \frac{y_t + \sigma \mathbf{z}_{t,m}}{\sigma^2} x_t \right) \tag{2}$$

where each $\mathbf{z}_t = \begin{pmatrix} \mathbf{z}_{t,1} & \cdots & \mathbf{z}_{t,M} \end{pmatrix}^\top \sim N(0, I_M)$ is an independent perturbation.

Key Intuition: Conditioned on **fixed dataset** $D_t = \{(x_s, y_s)_{s=1}^t\}$, each ensemble $\theta_{t,m}$ are i.i.d to the true posterior distribution.

Optimization formulation of ensmebles

- Each model $\tilde{\theta}_{t,m}$ in Equation (2) can be represented as a **learned model** $\hat{\theta}_{t,m}$ plus a **prior perturbation** $\theta_{0,m}$, i.e. $\tilde{\theta}_{t,m} = \hat{\theta}_{t,m} + \theta_{0,m}$;
- The learned model $\hat{\theta}_{t,m}$ is the closed form solution of the following perturbed optimization problem

$$\hat{\theta}_{t,m} = \arg\min_{\theta_m} \ell_m(\theta_m; D_t) := \frac{1}{\sigma^2} \sum_{s=1}^{t} (g_{\theta,m}(x_s) - y_s - \sigma \mathbf{z}_{s,m})^2 + \theta_m^\top \Sigma_0^{-1} \theta_m \qquad (3)$$

where

$$g_{\theta,m}(x) = \langle \theta_m + \theta_{0,m}, x \rangle$$

is a linear function with perturbed bias $\theta_{0,m}$.

In general, $g(\cdot)$ could be any function, including nonlinear mapping, e.g. neural networks.

Good approximation requires large ensmeble size

- **Histogram effect**: Larger M, uniform distribution over M models $\mathcal{U}(\tilde{\theta}_1, \ldots, \tilde{\theta}_M)$ better approximate the true posterior distribution.
- **Dependence issue**: For online sequential decision tasks, because of the dependency between the sampled model $\tilde{\theta}_{t,m}$ and the chosen action x_t, the dataset is not fixed anymore. Therefore, the ensembles $\tilde{\theta}_{t,m}$ are not i.i.d. to the true posterior given the dataset. To solve this issue, we need $M \gg O(1)$ ensembles to decouple the dependence.

Statistics vs Computation tradeoff.

- Requires a huge number of models ($M \approx 100$) for posterior approximation and good decision. [Li et al., 2022, Dwaracherla et al., 2020, Osband et al., 2021]
- 100 neural networks for each decision step is computationally expensive.

What to expect from this talk

- For practitioners:
  - **Adaptation**: Introduce a method, called Hypermodel, for tracking the posterior **efficiently**.
  - **Decision**: Introduce an algorithm based on Hypermodel, called Index Sampling, for **data-efficient** sequential decision-making
- For theorists:
  - **Statistics vs computation trade-off** for uncertainty estimation and sequential decision-making
  - **New probability tools** for (sequential) random projection

Outline

Background and motivations

Adaptation: Hypermodel for uncertainty estimation

Decision: Hypermodel-based index sampling

Introducing Hypermodel

- **Hypermodel** is a function f_ν parameterized with ν receives data $x \in \mathbb{R}^d$ and an random index $\xi \sim P_\xi$ in $\mathbb{R}^M$ and outputs a scalar

$$(x, \xi) \mapsto f_\nu(x, \xi) \in \mathbb{R}.$$

- **Sampling from Linear-Gaussian**: $P_\xi = N(0, I_M)$, $\nu = (\boldsymbol{A}, \mu)$ and sample via hypermodel

$$f_\nu(x, \xi) = \langle x, \mu + \boldsymbol{A}\xi \rangle \sim N(x^\top \mu, x^\top \boldsymbol{A}\boldsymbol{A}^\top x).$$

- **Ensemble sampling**: $P_\xi = \mathcal{U}(e_1, \ldots, e_M)$ and $\nu = [\theta_1, \ldots, \theta_M]$,

$$f_\nu(x, \xi) = \langle x, \nu\xi \rangle = \langle \theta_{\mathrm{id}(\xi)}, x \rangle.$$

  where $\mathrm{id}(e_i) = i$. Ensemble is a special case of hypermodel.

- **Generally**, $f_\nu(\cdot)$ can be any function, e.g. neural networks, transform P_ξ to any distribution.

The goal of hypermodel

Goal 1 **Uncertainty estimation**: Capture the uncertainty on the input data x via the variation over the hypermodel predictions

$$\{f_\nu(x, \xi_i)\}_{i=1,\dots}, \quad \xi_i \sim P_\xi.$$

Goal 2 **Adaptation**: Model ν is **trainable** to adjust its uncertainty representation when seeing more data. The reference distribution P_ξ remain fixed throughout the training.

Adaptation: Training objective for hypermodel

- Given the dataset $D_T = \{(x_t, y_t), t = 1, \ldots, T\}$, we **sample random vectors** $(\mathbf{z}_t)_{t=1,\ldots,T}$ from $P_{\mathbf{z}}$ (e.g. uniform over unit sphere) and augment the dataset as

$$D_T = \{(x_t, y_t, \mathbf{z}_t), t = 1, \ldots, T\}.$$

- The loss for single index ξ is

$$\ell^{\sigma,\beta}(\nu; D, \xi) = \sum_{(x,y,\mathbf{z})\in D} (f_\nu(x, \xi) - y - \sigma \mathbf{z}^\top \xi)^2 + \beta \|\nu\|^2 \tag{4}$$

  Recall the optimization in Equation (3) for one ensemble model is a special case of Equation (4) with ξ being one-hot vector.

- **Final objective**: integrate over all probable indices

$$\nu_T = \arg\min_{\nu} L(\nu; D_T) := \mathbb{E}_{\xi\sim P_\xi}[\ell^{\sigma,\beta}(\nu; D_T, \xi)]. \tag{5}$$

Index Sampling algorithm (with generic hypermodel)

Algorithm Index Sampling

1: **Input**: M, σ, β, ν_0
2: **for** $t = 1, 2, \ldots, T$ **do**
3: **Decision**: Sample $\xi_t \sim P_\xi$ independently and select

$$x_t = \arg\max_{x \in \mathcal{X}} f_{\nu_{t-1}}(x, \xi_t)$$

and observe y_t.
4: **Adaptation**: Sample $\mathbf{z}_t \sim P_Z$ independently and data.add$(x_t, y_t, \mathbf{z}_t)$. (Incrementally) solve the optimization problem in Equation (5) to obtain ν_t.
5: **end for**

Adaptation: How can this work?

Setup for main theorem

- Let data $(x_t, y_t), t = 1, \ldots, T$ generated by the linear-Gaussian model $y = \langle \theta^*, x \rangle + \omega^*$ where $\theta^* \sim N(\mu_0, \boldsymbol{\Sigma}_0)$ and $\omega^* \sim N(0, \sigma^2)$. Then the posterior is $\theta^* \mid D_t \sim N(\mu_T, \boldsymbol{\Sigma}_T)$.
- Let $P_{\tilde{\xi}}$ be the Gaussian distribution $N(0, \boldsymbol{I}_M)$.
- Let the $P_{\mathbf{z}}$ be the uniform distribution over unit sphere $\mathbb{S}^{M-1}$. Let $\mathbf{z}_t \sim P_{\mathbf{z}}$ for each t and $\mathbf{Z}_0 \in \mathbb{R}^{d \times M}$ be a random matrix with each row sampled from $P_{\mathbf{z}}$.
- Let the trainable parameters be $\nu = (\boldsymbol{A}, b)$ and the hypermodel be

$$
\begin{aligned}
f_\nu(x, \xi) = & \underbrace{\langle \mathbf{A}\xi + b, x \rangle}_{\textbf{Learnable } f^L_\nu(x,\xi)} + \underbrace{\langle \boldsymbol{\Sigma}_0^{1/2} \mathbf{Z}_0 \xi + \mu_0, x \rangle}_{\textbf{Fixed prior } f^P(x,\xi)} \\
= & \langle (b + \mu_0), x \rangle + \langle (\boldsymbol{A} + \boldsymbol{\Sigma}_0^{1/2} \mathbf{Z}_0)\xi, x \rangle
\end{aligned}
$$

is **Gaussian distributed**.

Adaptation: How can this work?

Theorem 2 (Approximate posterior matching).

- *Following the setup in previous slide. The posterior of the model on direction x is* $\langle \theta^*, x \rangle \mid D_T \sim N(x^\top \mu_T, x^\top \boldsymbol{\Sigma}_T x)$.
- *After solving the optimization problem in Equation* (5) $\nu_T = (b_T, \boldsymbol{A}_T) = \arg\min L(\nu; D_T)$.
- ***Posterior mean**: Let* $\tilde{b}_T = b_T + \mu_0$, *we have* $\tilde{b}_T = \mu_T$.
- ***Posterior variance**: Let* $\tilde{\boldsymbol{A}}_T = \boldsymbol{A}_T + \boldsymbol{\Sigma}_0^{1/2} \mathbf{Z}_0$, *the* (ε, δ)*-approximate posterior matching holds: with probability at least* $1 - \delta$,

$$(1-\varepsilon) x^\top \boldsymbol{\Sigma}_T x \leqslant x^\top \tilde{\boldsymbol{A}}_T \tilde{\boldsymbol{A}}_T^\top x \leqslant (1+\varepsilon) x^\top \boldsymbol{\Sigma}_T x, \quad \forall x \in \mathcal{X}$$

if $M \simeq \varepsilon^{-2}(\log|\mathcal{X}| + \log(1/\delta))$ *for finite set* $\mathcal{X}$, *and* $M \simeq \varepsilon^{-2}(d + \log(1/\delta))$ *for compact* $\mathcal{X}$, *e.g.* $\mathbb{R}^d$.

Statistics vs computation trade-off.

Analysis: a reduction to random projection

- Let the random matrix be

$$\mathbf{Z}_T^\top = (\mathbf{Z}_0^\top, \mathbf{z}_1, \ldots, \mathbf{z}_T) \in \mathbb{R}^{M \times (d+T)}$$

  and the data matrix be

$$\boldsymbol{X}_T = (\boldsymbol{\Sigma}_0^{-1/2}, x_1/\sigma, \ldots, x_T/\sigma)^\top \in \mathbb{R}^{(d+T)\times d}.$$

- Notice the inverse posterior covariance $\boldsymbol{\Sigma}_T^{-1} = \boldsymbol{\Sigma}_0^{-1} + (1/\sigma^2)\sum_{t=1}^T x_t x_t^\top = \boldsymbol{X}_T^\top \boldsymbol{X}_T$.
- The solution from Equation (5) is

$$\tilde{\mathbf{A}}_T = \boldsymbol{\Sigma}_T \left(\boldsymbol{\Sigma}_0^{-1/2} \mathbf{Z}_0 + \frac{1}{\sigma} \sum_{t=1}^T x_t \mathbf{z}_t^\top \right) = \boldsymbol{\Sigma}_T \boldsymbol{X}_T^\top \mathbf{Z}_T.$$

Analysis: a reduction to random projection

- Then $\tilde{\mathbf{A}}_T\tilde{\mathbf{A}}_T^\top = \boldsymbol{\Sigma}_T \boldsymbol{X}_T^\top \mathbf{Z}_T \mathbf{Z}_T^\top \boldsymbol{X}_T \boldsymbol{\Sigma}_T$ and $\boldsymbol{\Sigma}_T = \boldsymbol{\Sigma}_T \boldsymbol{X}_T^\top \boldsymbol{X}_T \boldsymbol{\Sigma}_T$.
- The (ε, δ)-approximation goal for $x^\top \boldsymbol{\Sigma}_T x \approx x^\top \tilde{\boldsymbol{A}}_T \tilde{\boldsymbol{A}}_T^\top x, \forall x \in \mathcal{X}$ becomes a random projection argument with random projection matrix $\mathbf{Z}_T^\top \in \mathbb{R}^{M \times (d+T)}$ and the vector to be projected $\boldsymbol{X}_T \boldsymbol{\Sigma}_T x$.

Random projection

With probability at least $1 - \delta$,

$$(1-\varepsilon)\|\boldsymbol{X}_T\boldsymbol{\Sigma}_T x\|^2 \leqslant \|\mathbf{Z}_T^\top \boldsymbol{X}_T \boldsymbol{\Sigma}_T x\|^2 \leqslant (1+\varepsilon)\|\boldsymbol{X}_T\boldsymbol{\Sigma}_T x\|^2, \quad \forall x \in \mathcal{X}.$$

Lemma 1 (Distributional JL lemma [Johnson and Lindenstrauss, 1984]).

For any $0 < \varepsilon, \delta < 1/2$ and $d \geqslant 1$ there exists a distribution $\mathcal{D}_{\varepsilon,\delta}$ on $\mathbb{R}^{M \times d}$ for $M = O\left(\varepsilon^{-2} \log(1/\delta)\right)$ such that for any $x \in \mathbb{R}^d$

$$\mathop{\mathbb{P}}_{\Pi \sim \mathcal{D}_{\varepsilon,\delta}} \left(\|\Pi x\|_2^2 \notin \left[(1-\varepsilon)\|x\|_2^2, (1+\varepsilon)\|x\|_2^2 \right] \right) < \delta$$

- Previous technique can handle the distribution such that
  - (1) Π has independent entries with subGaussian tails
  - (2) columns of Π are independent and the entries within each column of Π could be dependent but are chosen from $\{0, +1, -1\}$.
- In our case, each column of Π is sampled from $\mathcal{U}(\mathbb{S}^{M-1})$, which can not be handled by any existing JL proof technique in literature.

Calls for new proof ideas.

Comparison on the covered distribution of JL lemma

Matrix Dist.	iid Gaussian	iid from $\mathcal{U}\{+1,-1\}$	iid SG entries	Sparse JL	iid SG columns e.g. $\mathcal{U}(\mathbb{S}^{M-1})$
IM98	✓				
A01		✓			
M08	✓	✓	✓		
KN14				✓	
Ours	✓	✓	✓	✓	✓

- **We develop a new unified proof of JL lemma** via high-dimensional Hanson-Wright inequality.
- The technique allows each column vector in Π being subGaussian random vectors and the entries within each entry $\in \mathbb{R}$ could be dependent.
- **The first time** covering all important JL distributions including the one with each column $\sim \mathcal{U}(\mathbb{S}^{M-1})$. May result in new JL distributions with useful properties.

Outline

Background and motivations

Adaptation: Hypermodel for uncertainty estimation

Decision: Hypermodel-based index sampling

Decision: Problem setting

- **To simplify the exposition**, we consider linear bandit problem (single-stage stateless RL).
  - Movie recommendation system could be modeled as a linear bandit problem
- **Action**: select feature $\mathbf{x}_t \in \mathcal{X} \subseteq \mathbb{R}^d$.
  - A special case: one-hot feature $\mathcal{X} = \{e_1, \ldots, e_K\}$ where e_i is a one-hot vector. (K-armed bandit problem.)
- **Reward**: $y_t = \langle \theta^*, \mathbf{x}_t \rangle + \omega_t$ where $\theta^* \in \mathbb{R}^d$ is the unknown parameter and $\omega_t \sim \mathcal{N}(0, \sigma^2)$ is the noise.
- **Goal**: maximizes the expected total reward $\mathbb{E}[\sum_{t=1}^T \langle \theta^*, \mathbf{x}_t \rangle]$.
- **Performance metric**: regret

$$\text{Regret}(T) = \sum_{t=1}^{T} \mathbb{E}[\max_{\boldsymbol{x} \in \mathcal{X}} \langle \theta^*, \boldsymbol{x} \rangle - \langle \theta^*, \mathbf{x}_t \rangle]$$

Index Sampling algorithm in linear bandit problem

Algorithm Index Sampling

1: **Input**: M and $\mu_0, \mathbf{\Sigma}_0$

2: **Init**: Sample $\mathbf{Z}_0 = (\mathbf{z}_{0,1}, \ldots, \mathbf{z}_{0,d})^\top \in \mathbb{R}^{d \times M}$ with each $\mathbf{z}_{0,i}$ from $P_{\mathbf{z}}$, and let $\mathbf{A}_0 = \mathbf{\Sigma}_0^{1/2} \mathbf{Z}_0$.

3: **for** $t = 1, 2, \ldots, T$ **do**

4: **Decision**: Sample $\xi_t \sim P_\xi$ independently and select

$$\mathbf{x}_t = \arg\max_{x \in \mathcal{X}} f_{t-1}(x, \xi_t) := \langle x, \mu_{t-1} + \mathbf{A}_{t-1} \xi_t \rangle$$

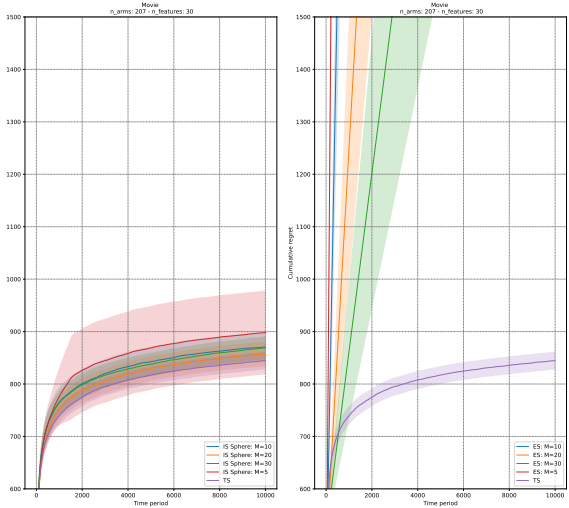
and observe $y_t = \langle \theta^*, \mathbf{x}_t \rangle + w_t$

5: **Adaptation**: Sample $\mathbf{z}_t \sim P_Z$ independently and update $\mathbf{A}_{t-1} \to \mathbf{A}_t$ and $\mu_{t-1} \to \mu_t$ as

$$\mathbf{A}_t = \mathbf{\Sigma}_t \left(\mathbf{\Sigma}_{t-1}^{-1} \mathbf{A}_{t-1} + \frac{1}{\sigma} \mathbf{x}_t \mathbf{z}_t^\top \right) \quad \mu_t = \mathbf{\Sigma}_t \left(\mathbf{\Sigma}_{t-1}^{-1} \mu_{t-1} + \frac{1}{\sigma^2} \mathbf{x}_t y_t \right)$$

6: **end for**

Figure: Regret (1000 expes) in Linear bandit problem (Movie Recommendation):
Index (Left, Much Better) v.s. Ensemble (Right) - Varies M - x-axis: Time period; y-axis: Regret



Theoretical comparison

- To achieve the same regret order as Thompson Sampling, the required size M is

Algorithms	Finite action set		Compact action set
	One-hot features	General features	General features
TS(Regret)	$O(\sqrt{KT\log K})$	$O(\sqrt{dT\log K})$	$O(d\sqrt{T\log T})$
ES(M)	$M = O(KT/d)$	$M = O(KT/d)$	N/A in literature
IS(M)	$M = O(\log K \log KT)$	$M = O(d\log T)$	$M = O(d\log T)$

- Index Sampling with Gaussian index distribution has **exponential improvement** on M than Ensmeble Sampling.
- Enable us to use a much smaller ensemble size M. Improve the data and computation efficiency. Our proof technique also **enables the analysis for compact action set.**

Core argument: sequential factorization

Definition 2 ((ϵ, δ, T)-approximate posterior covariance factorization).

We say an algorithm is (ϵ, δ, T)-approximation algorithm if it produces a sequence of factors $(\mathbf{A}_t \in \mathbb{R}^{d \times M}, t = 0, 1, \ldots, T)$ such that with probability at least $1 - \delta$,

$$(1-\varepsilon)x^\top \mathbf{\Sigma}_t x \leqslant x^\top \mathbf{A}_t \mathbf{A}_t^\top x \leqslant (1+\varepsilon)x^\top \mathbf{\Sigma}_t x, \quad \forall x \in \mathcal{X}, \forall t \in \{0, 1, \ldots, T\}.$$

where $\mathbf{\Sigma}_t$ is the posterior covariance given data D_t under linear-Gaussian model.

- In sequential decision making, the data $(\mathbf{x}_t)$ are adaptively generated and thus are statistically dependent with the random vectors $(\mathbf{z}_t)$.
- This makes the analysis much more challenging. All analysis in the literature are based on the assumption that the data are fixed non-random or generated independently.

We develop a **new sequential random projection argument** based on stopping-time analysis and method of mixtures to handle the dependence.

Challenge for analysis: Dependence structure in the index sampling

Unfolding the update rule in line 5 of Algorithm 2,

$$\boldsymbol{\Sigma}_t^{-1} = \boldsymbol{\Sigma}_0^{-1} + \frac{1}{\sigma^2}\sum_{i=1}^{t}\mathbf{x}_i\mathbf{x}_i^\top, \quad \boldsymbol{\Sigma}_t^{-1}\mathbf{A}_t = \boldsymbol{\Sigma}_0^{-1/2}\mathbf{Z}_0 + \frac{1}{\sigma}\sum_{i=1}^{t}\mathbf{x}_i\mathbf{z}_i^\top, \quad \boldsymbol{\Sigma}_t^{-1}\mu_t = \boldsymbol{\Sigma}_0^{-1}\mu_0 + \frac{1}{\sigma^2}\sum_{i=1}^{t}\mathbf{x}_i y_i$$

Let $\mathscr{F}_{t-1} = \sigma(\mathbf{Z}_0, \mathbf{x}_1, y_1, \mathbf{z}_1, \ldots, \mathbf{x}_{t-1}, y_{t-1}, \mathbf{z}_{t-1}, \mathbf{x}_t, y_t)$. We could see the dependence structure in the algorithm as follows:

- $\mathbf{z}_t$ is independent of $\mathscr{F}_{t-1}$.
- ξ_{t+1} is independent of $\mathscr{F}_{t-1}$ and $\mathbf{z}_t$.
- $\mathbf{x}_{t+1}$ is deterministic conditioned on $\mathscr{F}_{t-1}, \mathbf{z}_t$ and ξ_{t+1}.

We could draw the dependence graph of the algorithm as Figure 12.

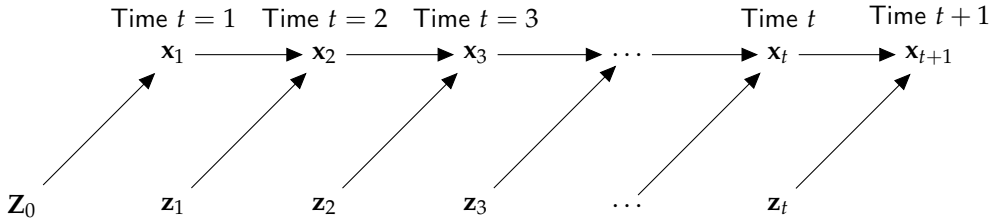


Figure: Dependence graph of the Index Sampling (Algorithm 2)

Towards general decision-making with Deep Neural Networks

- Nonlinear feature mapping $\phi_w(\cdot)$ with parameters w. E.g. hidden layers in neural networks.
- Together, with trainable $\nu = \{A, b, w\}$ and fixed parameters $\{A_0, b_0, w_0\}$

$$f_\nu(x, \xi) = \underbrace{\langle A\xi + b, \phi_w(x)\rangle}_{\textbf{Learnable } f^L_\nu(x,\xi)} + \underbrace{\langle A_0\xi + b_0, \phi_{w_0}(x)\rangle}_{\textbf{Fixed prior } f^P(x,\xi)}$$

- [Osband et al., 2021, 2022] use synthetic neural network generated data to demostrate that hypermodel can achieve better uncertainty estimation with much less model size and computation than ensembles.
- [Dwaracherla et al., 2020, Li et al., 2022] demonstrate that hypermodel for exploration in neural bandit and deep RL problems is both regret and computation efficient in several benchmarks, including the Atari and deepsea benchmark.

Data and computation efficiency in Deep RL benchmarks

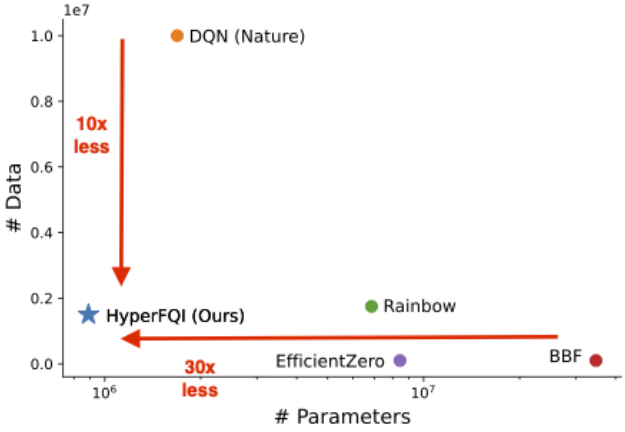


Figure: To achieve **human-level performance in Atari benchmark**: Based on Hypermodel, our HyperFQI algorithm uses $30x$ less parameters than BBF (STOA in 2023 by deepmind), 10x less data usage than DQN (Nature 2015 by deepmind).

Summary

- For practitioners:
  - **Adaptation**: Introduce a method, called Hypermodel, for tracking the posterior in a **computation-efficient** way
  - **Decision**: Introduce an algorithm based on Hypermodel, called Index Sampling, for **data-efficient** sequential decision-making
- For theorists:
  - **Statistics vs computation trade-off** for uncertainty estimation and sequential decision-making
  - **New probability tools** for (sequential) random projection

Thank you for listening!

References I

V. Dwaracherla, X. Lu, M. Ibrahimi, I. Osband, Z. Wen, and B. V. Roy. Hypermodels for exploration. In International Conference on Learning Representations, 2020. URL https://openreview.net/forum?id=ryx6WgStPB.

W. B. Johnson and J. Lindenstrauss. Extensions of lipschitz mappings into a hilbert space. In Conference on Modern Analysis and Probability, volume 26, pages 189–206. American Mathematical Society, 1984.

Z. Li, Y. Li, Y. Zhang, T. Zhang, and Z.-Q. Luo. HyperDQN: A randomized exploration method for deep reinforcement learning. In International Conference on Learning Representations, 2022. URL https://openreview.net/forum?id=X0nrKAXu7g-.

I. Osband, Z. Wen, S. M. Asghari, V. Dwaracherla, M. Ibrahimi, X. Lu, and B. Van Roy. Epistemic neural networks. arXiv preprint arXiv:2107.08924, 2021.

References II

I. Osband, Z. Wen, S. M. Asghari, V. Dwaracherla, X. Lu, M. Ibrahimi, D. Lawson, B. Hao, B. O'Donoghue, and B. Van Roy. The neural testbed: Evaluating joint predictions. Advances in Neural Information Processing Systems, 35:12554–12565, 2022.