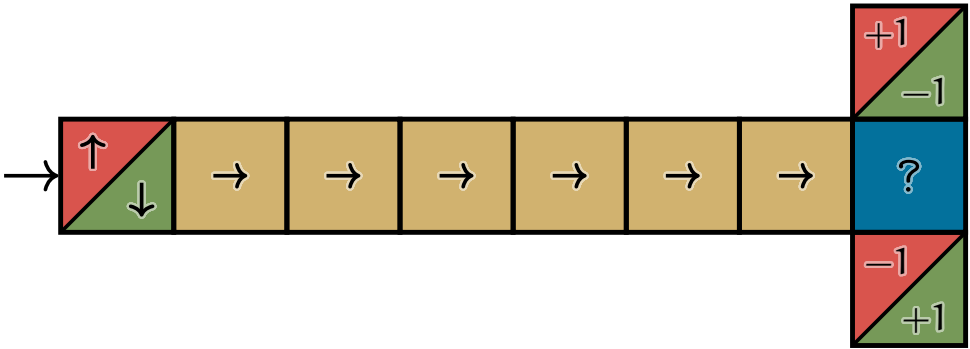
Partially Observable RL with Memory Traces

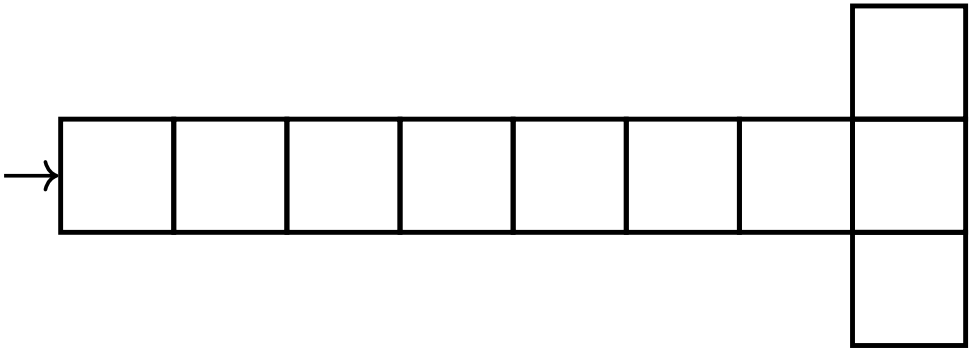
Onno Eberhard[12] · Michael Muehlebach[1] · Claire Vernade[2]

[1]Max Planck Institute for Intelligent Systems [2]University of Tübingen

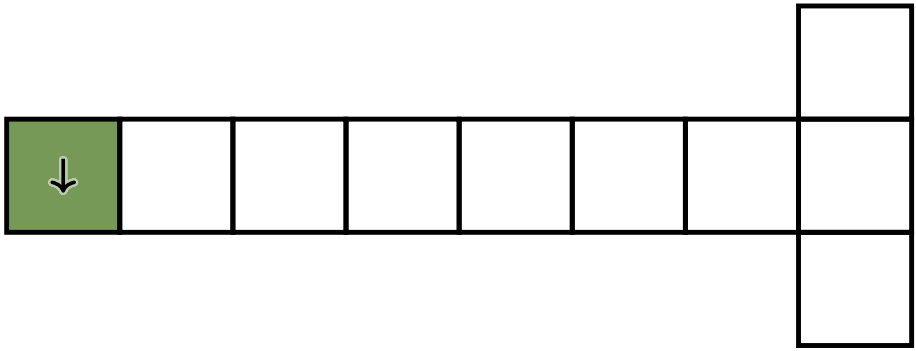
Bakker's T-maze



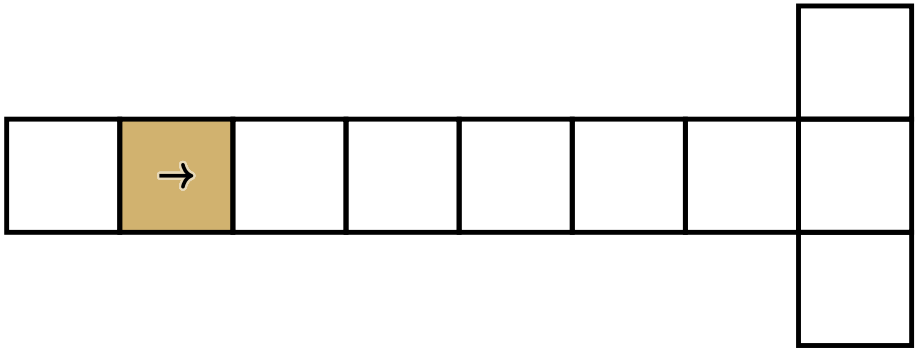
Bakker's T-maze



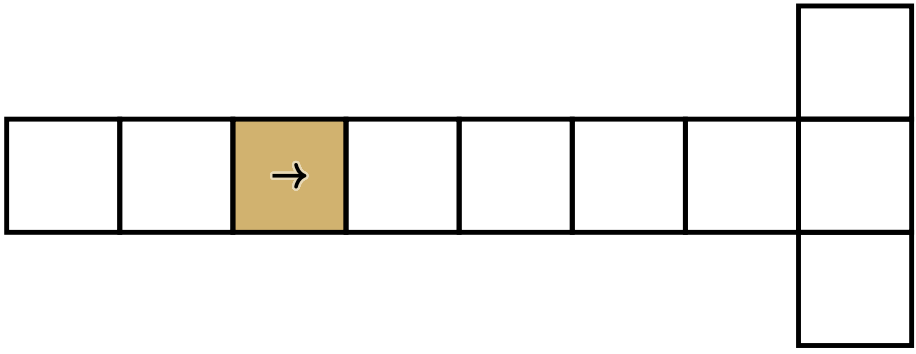
Bakker's T-maze



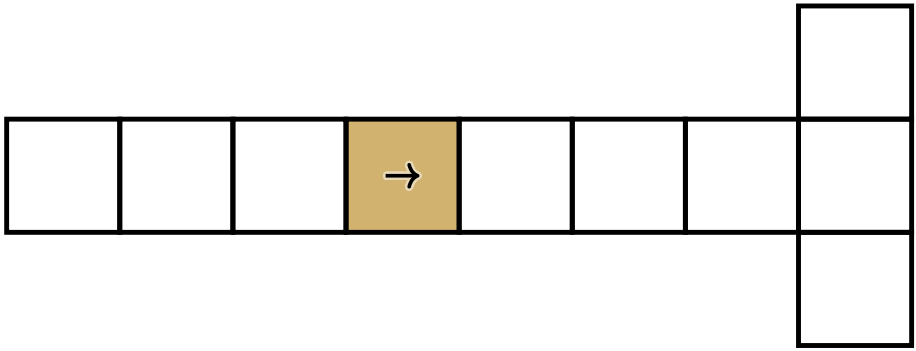
Bakker's T-maze



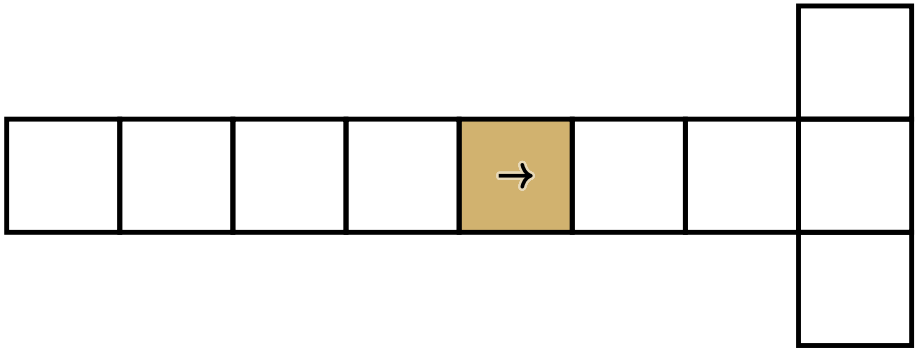
Bakker's T-maze



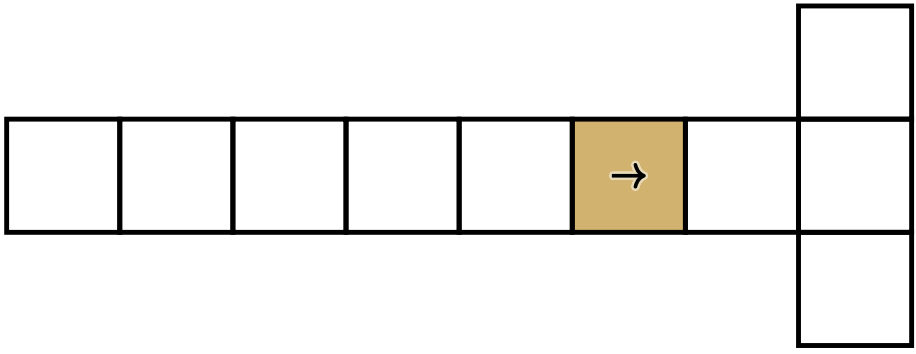
Bakker's T-maze



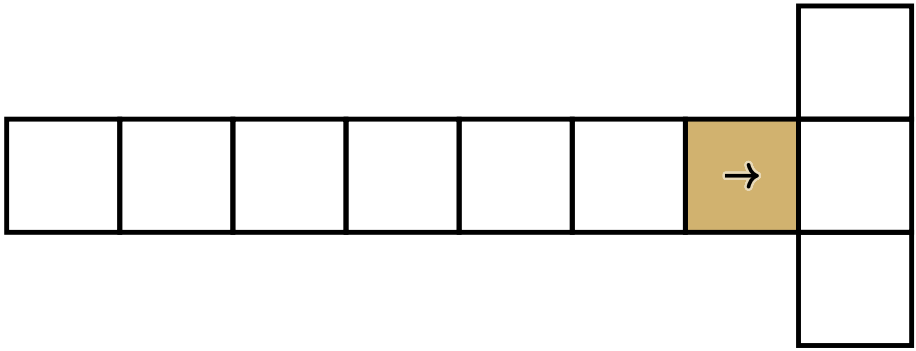
Bakker's T-maze



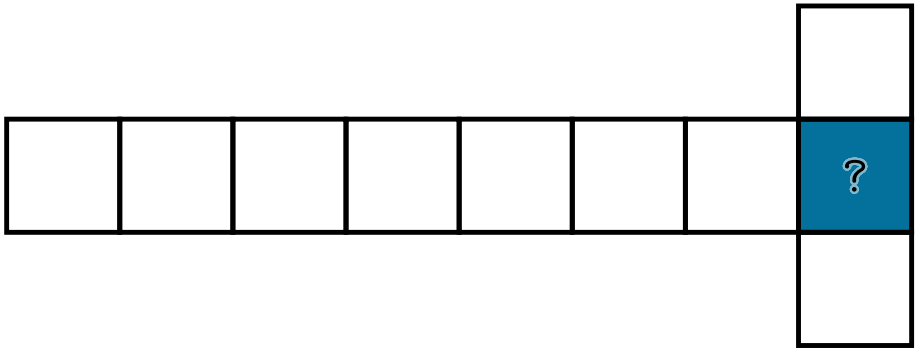
Bakker's T-maze



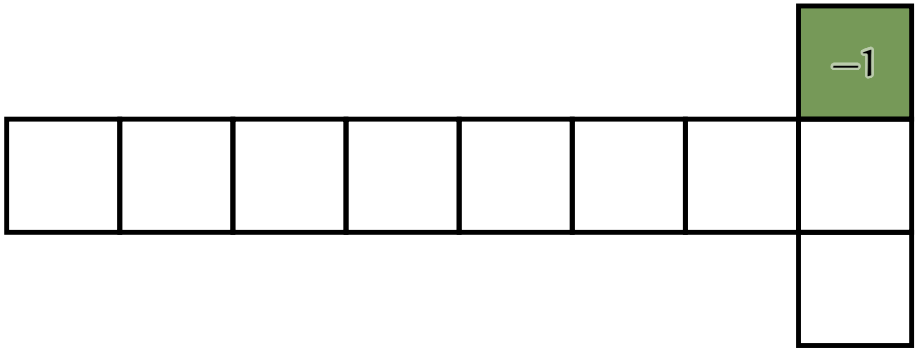
Bakker's T-maze



Bakker's T-maze



Bakker's T-maze



Memory

- Memory is necessary in many partially observable environments
  - $\rightarrow$ *Memory*: a compressed representation of the history of observations

Memory

- Memory is necessary in many partially observable environments
  - $\rightarrow$ *Memory*: a compressed representation of the history of observations
- Length-m window ("frame stacking"):

$$\mathrm{win}_m(\underbrace{y_0, y_{-1}, \ldots}_{\text{history } h}) \doteq (y_0, y_{-1}, \ldots, y_{-m+1})$$

Memory

- Memory is necessary in many partially observable environments
  - $\rightarrow$ *Memory*: a compressed representation of the history of observations
- Length-m window ("frame stacking"):

$$\mathrm{win}_m(\underbrace{y_0, y_{-1}, \ldots}_{\text{history } h}) \doteq (y_0, y_{-1}, \ldots, y_{-m+1})$$

- Our approach: the *memory trace* with forgetting factor $\lambda \in [0, 1)$:

$$z_\lambda(h) \doteq (1-\lambda) \sum_{k=0}^{|h|-1} \lambda^k y_{-k}$$

Memory

- Memory is necessary in many partially observable environments
  - → *Memory*: a compressed representation of the history of observations
- Length-m window ("frame stacking"):

$$\mathrm{win}_m(\underbrace{y_0, y_{-1}, \dots}_{\text{history } h}) \doteq (y_0, y_{-1}, \dots, y_{-m+1})$$

- Our approach: the *memory trace* with forgetting factor $\lambda \in [0, 1)$:

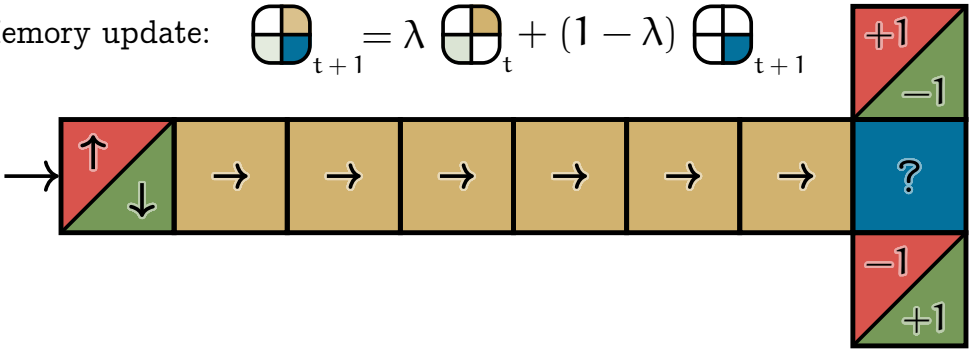
$$z_\lambda(h) \doteq (1-\lambda) \sum_{k=0}^{|h|-1} \lambda^k y_{-k}$$

  - → Recursively computable: $z_\lambda(y_0, y_{-1}, \dots) = \lambda z_\lambda(y_{-1}, y_{-2}, \dots) + (1-\lambda) y_0$

Memory traces in the T-maze

new memory old memory new observation

Memory update: $m_{t+1} = \lambda\, m_t + (1-\lambda)\, o_{t+1}$



Memory traces in the T-maze

new memory old memory new observation

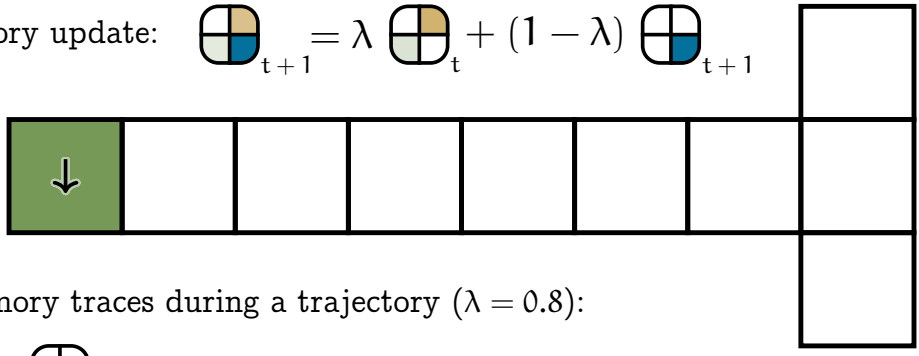
Memory update:  $= \lambda$  $+ (1 - \lambda)$ 

Memory traces during a trajectory ($\lambda = 0.8$):

Memory traces in the T-maze

new memory old memory new observation

Memory update: $= \lambda$ $+ (1 - \lambda)$ $_{t+1}$ $_{t}$ $_{t+1}$

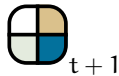
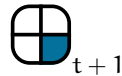


Memory traces during a trajectory ($\lambda = 0.8$):

$_0$ $_1$

Memory traces in the T-maze

new memory old memory new observation

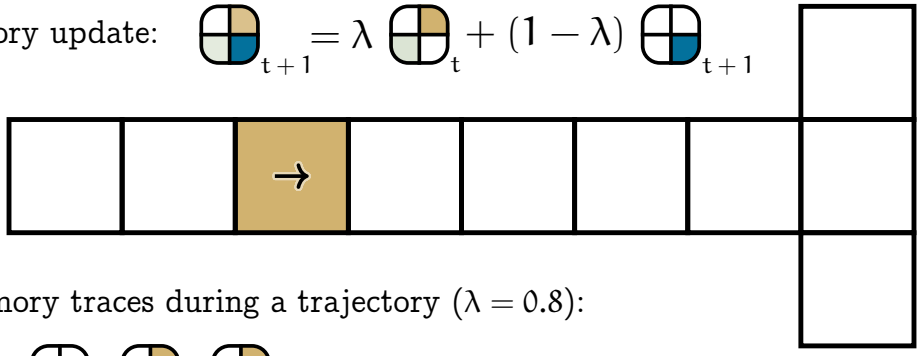
Memory update:  $= \lambda$  $+ (1 - \lambda)$ 

Memory traces during a trajectory ($\lambda = 0.8$):

Memory traces in the T-maze

new memory old memory new observation

Memory update: $\mathbf{m}_{t+1} = \lambda\, \mathbf{m}_t + (1-\lambda)\, \mathbf{o}_{t+1}$



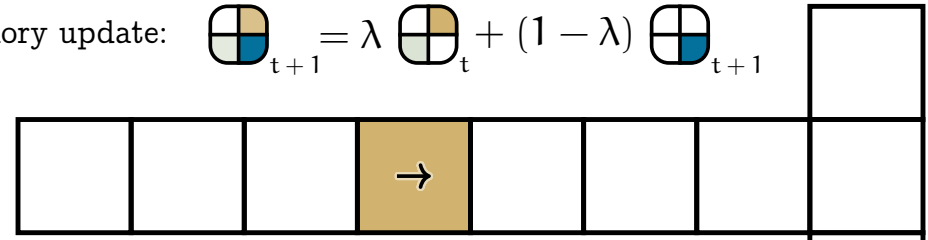
Memory traces during a trajectory ($\lambda = 0.8$):

$\mathbf{m}_0$ $\mathbf{m}_1$ $\mathbf{m}_2$ $\mathbf{m}_3$

Memory traces in the T-maze

new memory old memory new observation

Memory update: $= \lambda$ $+ (1 - \lambda)$ $_{t+1}$ $_{t}$ $_{t+1}$



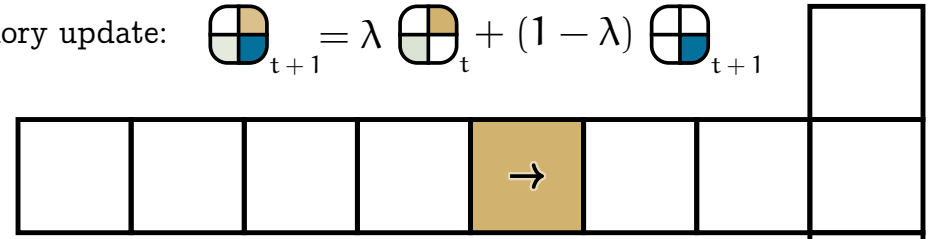
Memory traces during a trajectory ($\lambda = 0.8$):

$_0$ $_1$ $_2$ $_3$ $_4$

Memory traces in the T-maze

new memory old memory new observation

Memory update: $= \lambda \; _{t} + (1 - \lambda) \; _{t+1}$



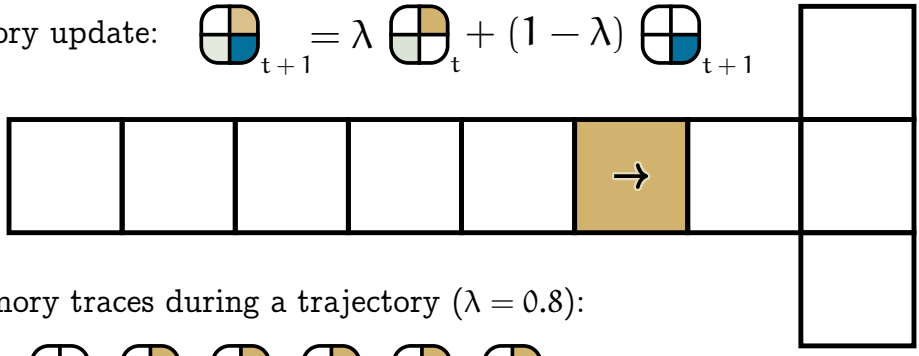
Memory traces during a trajectory ($\lambda = 0.8$):

$_0$ $_1$ $_2$ $_3$ $_4$ $_5$

Memory traces in the T-maze

new memory old memory new observation

Memory update: $= \lambda$ $+ (1 - \lambda)$ $_{t+1}$ $_{t}$ $_{t+1}$



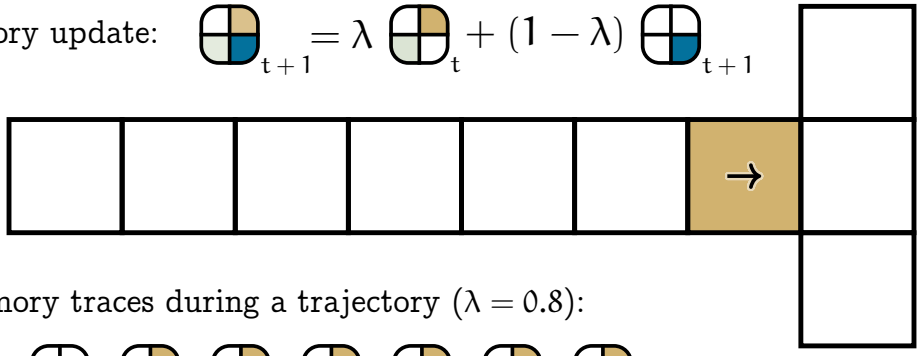
Memory traces during a trajectory ($\lambda = 0.8$):

$_0$ $_1$ $_2$ $_3$ $_4$ $_5$ $_6$

Memory traces in the T-maze

new memory old memory new observation

Memory update: $\mathbf{e}_{t+1} = \lambda\, \mathbf{e}_t + (1-\lambda)\, \mathbf{x}_{t+1}$



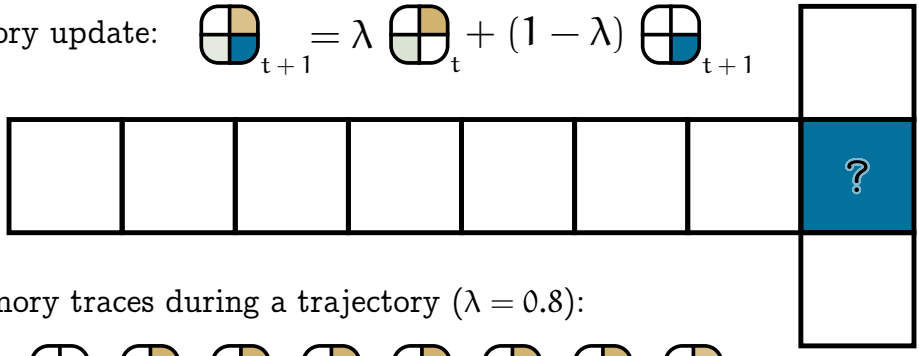
Memory traces during a trajectory ($\lambda = 0.8$):

$\mathbf{e}_0$ $\mathbf{e}_1$ $\mathbf{e}_2$ $\mathbf{e}_3$ $\mathbf{e}_4$ $\mathbf{e}_5$ $\mathbf{e}_6$ $\mathbf{e}_7$

Memory traces in the T-maze

new memory old memory new observation

Memory update: $\mathbf{m}_{t+1} = \lambda\, \mathbf{m}_t + (1-\lambda)\, \mathbf{o}_{t+1}$



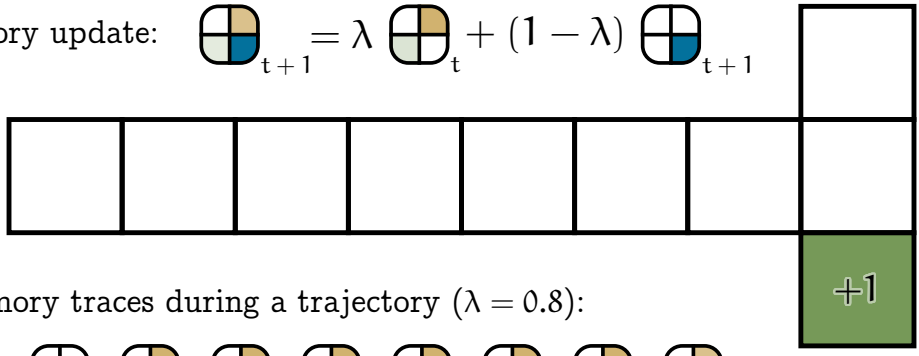
Memory traces during a trajectory ($\lambda = 0.8$):

$\mathbf{m}_0$ $\mathbf{m}_1$ $\mathbf{m}_2$ $\mathbf{m}_3$ $\mathbf{m}_4$ $\mathbf{m}_5$ $\mathbf{m}_6$ $\mathbf{m}_7$ $\mathbf{m}_8$

Memory traces in the T-maze

new memory old memory new observation

Memory update: $\square_{t+1} = \lambda \square_{t} + (1-\lambda) \square_{t+1}$



Memory traces during a trajectory ($\lambda = 0.8$):

$\square_0$ $\square_1$ $\square_2$ $\square_3$ $\square_4$ $\square_5$ $\square_6$ $\square_7$ $\square_8$

Offline on-policy evaluation

- We consider the problem of *policy evaluation* with offline data
  - $\rightarrow$ The environment $\mathcal{E}$ is a hidden Markov model (no explicit policy)
  - $\rightarrow$ The observation space $\mathcal{Y}$ is one-hot

Offline on-policy evaluation

- We consider the problem of *policy evaluation* with offline data
  - $\rightarrow$ The environment $\mathcal{E}$ is a hidden Markov model (no explicit policy)
  - $\rightarrow$ The observation space $\mathcal{Y}$ is one-hot
- Q: How much data do we need to accurately estimate the value function?

Offline on-policy evaluation

- We consider the problem of *policy evaluation* with offline data
  - $\rightarrow$ The environment $\mathcal{E}$ is a hidden Markov model (no explicit policy)
  - $\rightarrow$ The observation space $\mathcal{Y}$ is one-hot
- Q: How much data do we need to accurately estimate the value function?
- Given a function class $\mathcal{F} \subset \{\mathcal{Y}^\infty \rightarrow [\underline{v}, \overline{v}]\}$, find $f \in \mathcal{F}$ that minimizes

$$\mathcal{R}_{\mathcal{E}}(f) \doteq \mathbb{E}_{\mathcal{E}}\left[\left\{f(y_0, y_{-1}, \dots) - \sum_{t=0}^{\infty} \gamma^t r(y_{t+1})\right\}^2\right],$$

where $r : \mathcal{Y} \rightarrow [\underline{r}, \overline{r}]$ (rewards), $\gamma \in [0, 1)$ (discount), $\underline{v} \doteq \underline{r}/(1-\gamma)$, and $\overline{v} \doteq \overline{r}/(1-\gamma)$

Offline on-policy evaluation

- We consider the problem of *policy evaluation* with offline data
  - → The environment $\mathcal{E}$ is a hidden Markov model (no explicit policy)
  - → The observation space $\mathcal{Y}$ is one-hot
- Q: How much data do we need to accurately estimate the value function?
- Given a function class $\mathcal{F} \subset \{\mathcal{Y}^\infty \to [\underline{v}, \overline{v}]\}$, find $f \in \mathcal{F}$ that minimizes

$$\mathcal{R}_\mathcal{E}(f) \doteq \mathbb{E}_\mathcal{E}\left[\left\{f(y_0, y_{-1}, \dots) - \sum_{t=0}^{\infty} \gamma^t r(y_{t+1})\right\}^2\right],$$

  where $r : \mathcal{Y} \to [\underline{r}, \overline{r}]$ (rewards), $\gamma \in [0, 1)$ (discount), $\underline{v} \doteq \underline{r}/(1-\gamma)$, and $\overline{v} \doteq \overline{r}/(1-\gamma)$
- Window memory: $\mathcal{F}_m \doteq \{f \circ \mathrm{win}_m \mid f : \mathcal{Y}^m \to [\underline{v}, \overline{v}]\}$

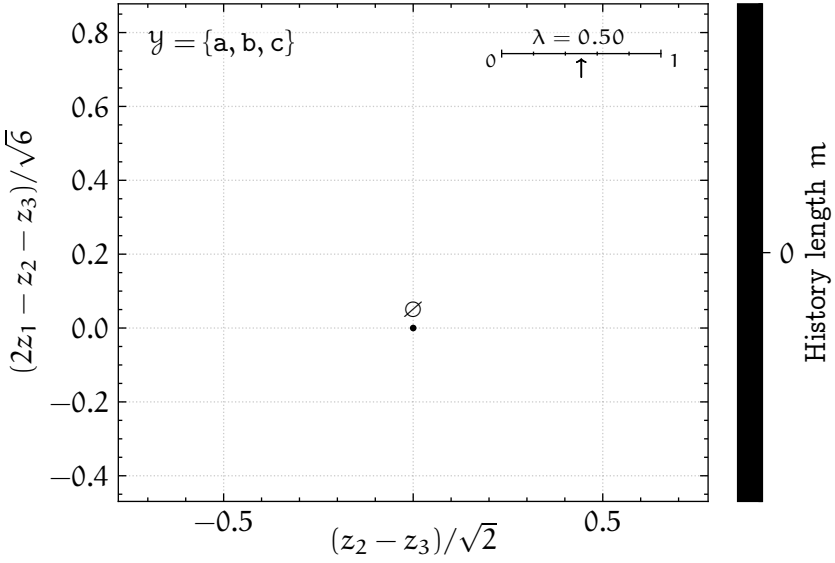
Offline on-policy evaluation

- We consider the problem of *policy evaluation* with offline data
  - → The environment $\mathcal{E}$ is a hidden Markov model (no explicit policy)
  - → The observation space $\mathcal{Y}$ is one-hot
- Q: How much data do we need to accurately estimate the value function?
- Given a function class $\mathcal{F} \subset \{\mathcal{Y}^\infty \to [\underline{v}, \overline{v}]\}$, find $f \in \mathcal{F}$ that minimizes

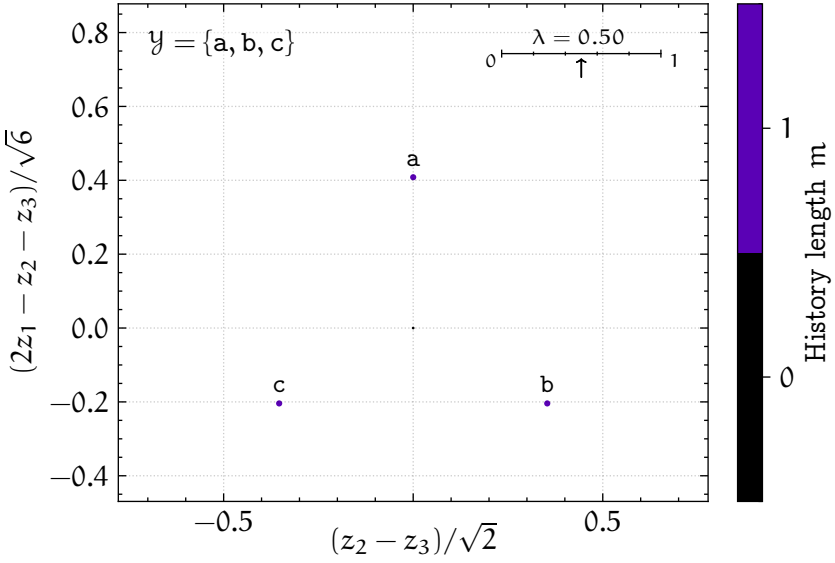
$$\mathcal{R}_\mathcal{E}(f) \doteq \mathbb{E}_\mathcal{E}\left[\left\{f(y_0, y_{-1}, \dots) - \sum_{t=0}^{\infty} \gamma^t r(y_{t+1})\right\}^2\right],$$

  where $r : \mathcal{Y} \to [\underline{r}, \overline{r}]$ (rewards), $\gamma \in [0, 1)$ (discount), $\underline{v} \doteq \underline{r}/(1-\gamma)$, and $\overline{v} \doteq \overline{r}/(1-\gamma)$
- Window memory: $\mathcal{F}_m \doteq \{f \circ \mathrm{win}_m \mid f : \mathcal{Y}^m \to [\underline{v}, \overline{v}]\}$
- Memory traces: $\mathcal{F}_\lambda \doteq \{f \circ z_\lambda \mid f : \mathcal{Z}_\lambda \to [\underline{v}, \overline{v}]\}$, where $\mathcal{Z}_\lambda \doteq \{z_\lambda(h) \mid h \in \mathcal{Y}^\infty\}$

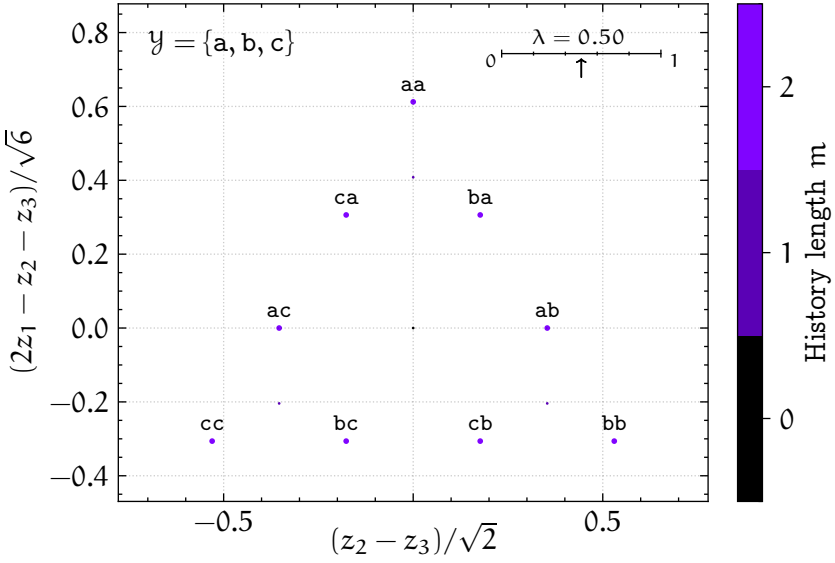
The geometry of the *trace space* $\mathcal{Z}_\lambda$



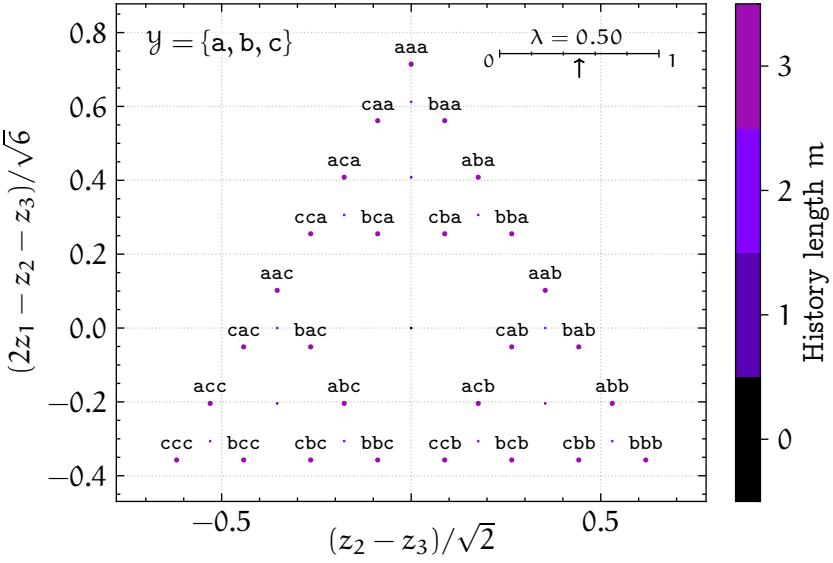
The geometry of the *trace space* $\mathcal{Z}_\lambda$



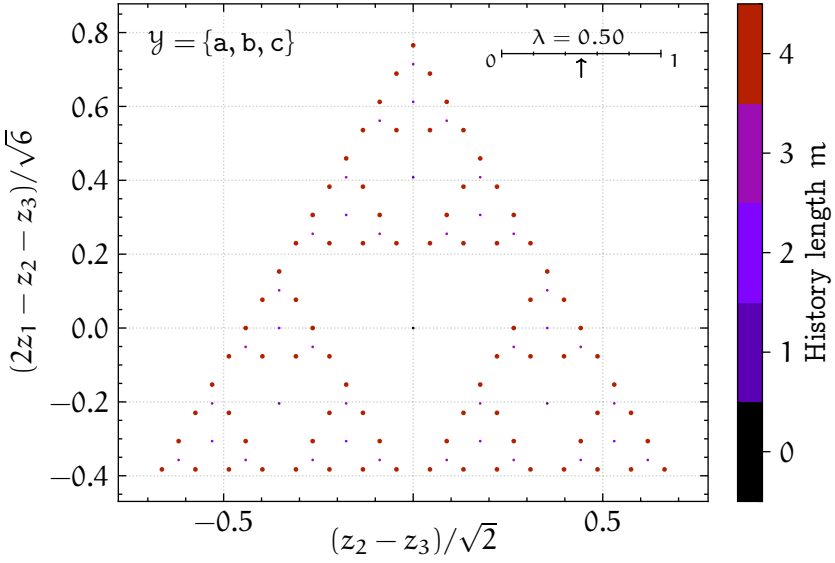
The geometry of the *trace space* $\mathcal{Z}_\lambda$



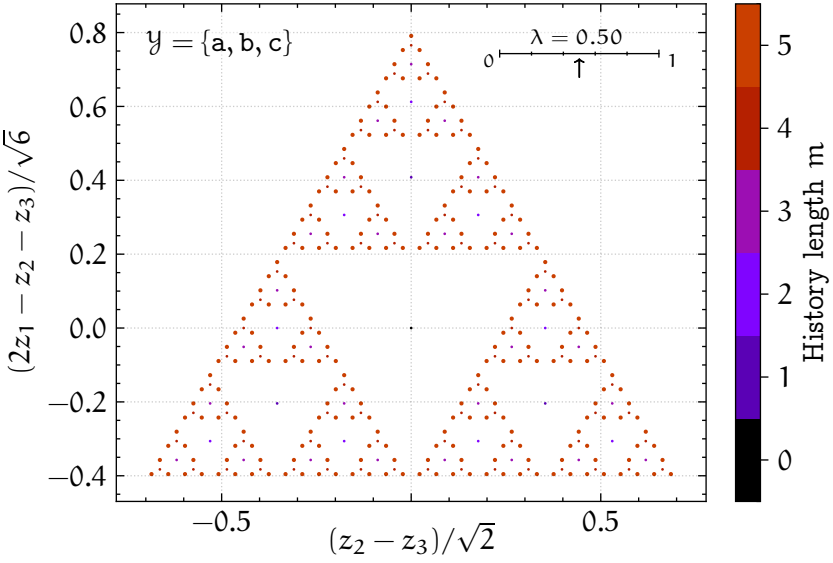
The geometry of the *trace space* $\mathcal{Z}_\lambda$



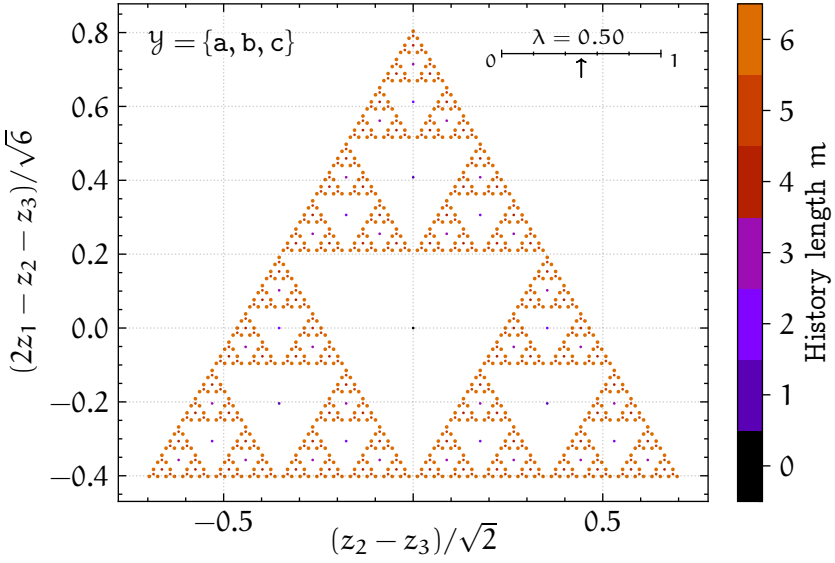
The geometry of the *trace space* $\mathcal{Z}_\lambda$



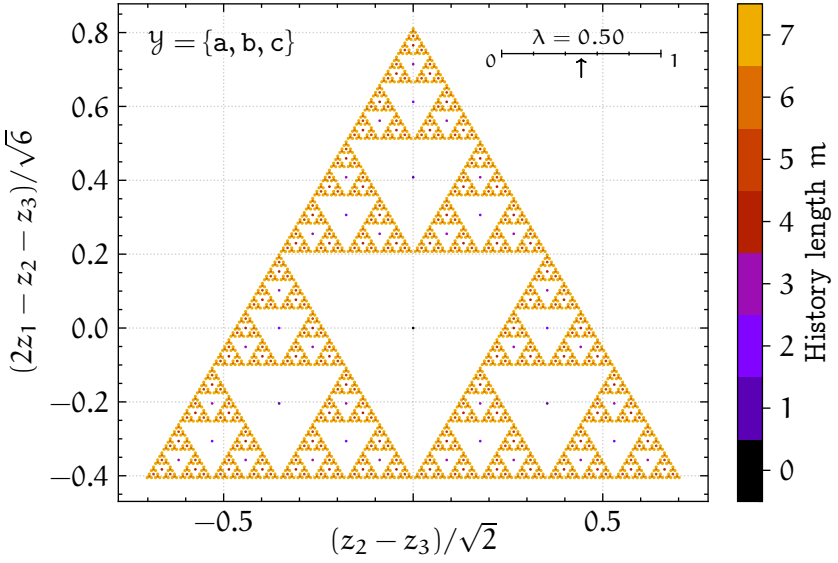
The geometry of the *trace space* $\mathcal{Z}_\lambda$



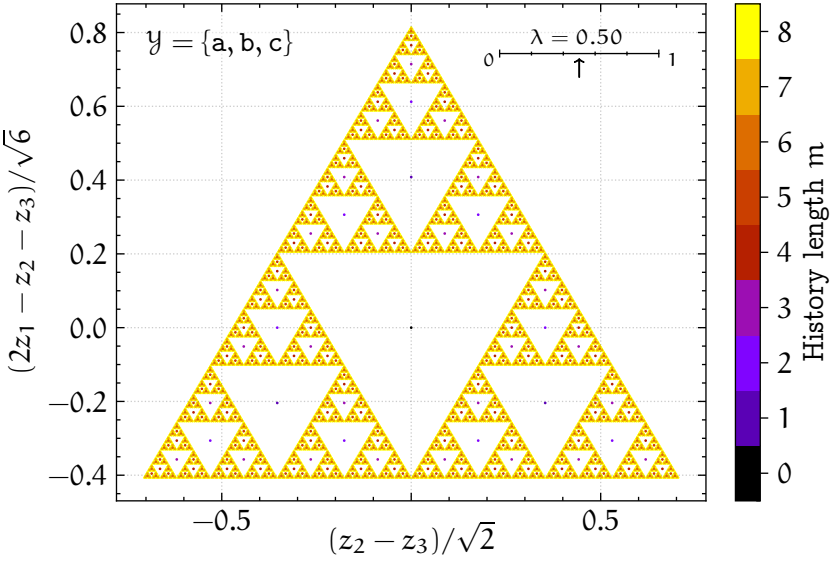
The geometry of the *trace space* $\mathcal{Z}_\lambda$



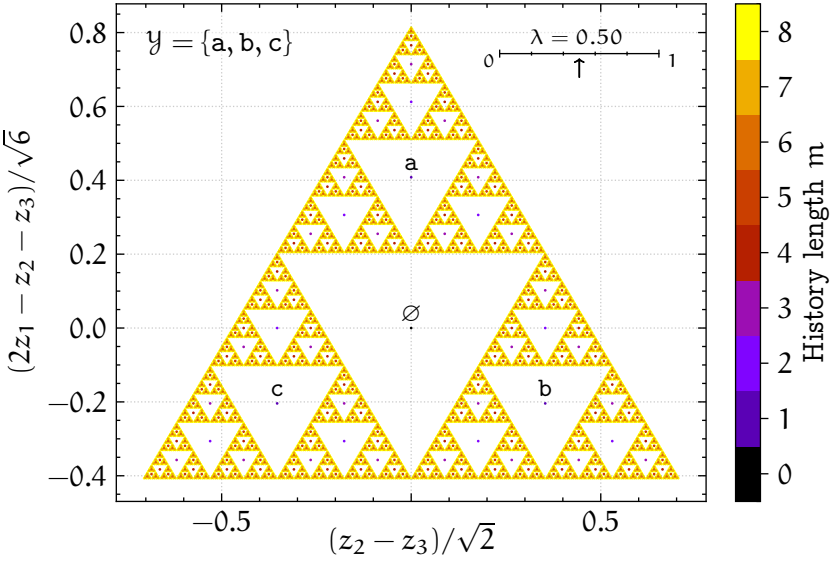
The geometry of the *trace space* $\mathcal{Z}_\lambda$



The geometry of the *trace space* $\mathcal{Z}_\lambda$



The geometry of the *trace space* $\mathcal{Z}_\lambda$



Learning value functions

Theorem (Hoeffding bound)

Learning a good value estimate seems easier if the *metric entropy* $H_\epsilon(\mathcal{F})$ of the function class $\mathcal{F}$ is small.

The metric entropy is a measure of "size" for the (infinite) function class.

Learning value functions

Theorem (Hoeffding bound)

Given a dataset $\mathcal{D}$ of n trajectories from an environment $\mathcal{E}$, a function class $\mathcal{F}$, and some $\epsilon > 0$, let $\mathcal{F}^\epsilon$ be the smallest ϵ-cover of $\mathcal{F}$ and $f_n \doteq \arg\min_{f \in \mathcal{F}^\epsilon} \sum_{\mathcal{D}} \{f(h) - \sum_{t=0}^{\infty} \gamma^t r(y_{t+1})\}^2$. Then, with probability at least $1 - \delta$,

$$\mathcal{R}_{\mathcal{E}}(f_n) \leqslant \mathcal{R}_{\mathcal{E}}(\mathcal{F}) + (\bar{v} - \underline{v})^2 \sqrt{\frac{H_\epsilon(\mathcal{F}) + \log \frac{2}{\delta}}{2n}} + \mathcal{O}(\epsilon).$$

Let (X, ρ) be a metric space and $T \subset X$. The *ϵ-covering number* $N_\epsilon(T)$ is the cardinality of the smallest set $S \subset X$ such that for every $x \in T$, there exists a $y \in S$ with $\rho(x, y) \leqslant \epsilon$. The metric entropy of T is defined as $H_\epsilon(T) \doteq \log N_\epsilon(T)$.

Metric entropy and memory

- For windows of length m, we have $\boxed{H_\epsilon(\mathcal{F}_m) \in \Theta(|\mathcal{Y}|^m)}$
  - $\rightarrow$ Exponential in m: long windows are expensive!

Metric entropy and memory

- For windows of length m, we have $\boxed{H_\epsilon(\mathcal{F}_m) \in \Theta(|\mathcal{Y}|^m)}$
  - $\rightarrow$ Exponential in m: long windows are expensive!
- Memory traces with forgetting factor $\lambda < \frac{1}{2}$ remember everything
  - $\rightarrow$ There is no compression: z_λ is invertible and therefore $\boxed{H_\epsilon(\mathcal{F}_\lambda) = \infty}$

Metric entropy and memory

- For windows of length m, we have $\boxed{H_\epsilon(\mathcal{F}_m) \in \Theta(|\mathcal{Y}|^m)}$
  - $\rightarrow$ Exponential in m: long windows are expensive!
- Memory traces with forgetting factor $\lambda < \frac{1}{2}$ remember everything
  - $\rightarrow$ There is no compression: z_λ is invertible and therefore $\boxed{H_\epsilon(\mathcal{F}_\lambda) = \infty}$
- Need to "zoom in" to differentiate histories that only differ far in the past

Metric entropy and memory

- For windows of length m, we have $\boxed{H_\epsilon(\mathcal{F}_m) \in \Theta(|\mathcal{Y}|^m)}$
  - $\rightarrow$ Exponential in m: long windows are expensive!
- Memory traces with forgetting factor $\lambda < \frac{1}{2}$ remember everything
  - $\rightarrow$ There is no compression: z_λ is invertible and therefore $\boxed{H_\epsilon(\mathcal{F}_\lambda) = \infty}$
- Need to "zoom in" to differentiate histories that only differ far in the past
- The "resolution" of a function class is given by its Lipschitz constant

Metric entropy and memory

- For windows of length m, we have $\boxed{H_\epsilon(\mathcal{F}_m) \in \Theta(|\mathcal{Y}|^m)}$
  - $\rightarrow$ Exponential in m: long windows are expensive!
- Memory traces with forgetting factor $\lambda < \frac{1}{2}$ remember everything
  - $\rightarrow$ There is no compression: z_λ is invertible and therefore $\boxed{H_\epsilon(\mathcal{F}_\lambda) = \infty}$
- Need to "zoom in" to differentiate histories that only differ far in the past
- The "resolution" of a function class is given by its Lipschitz constant
- We consider the class $\mathcal{F}_{\lambda,L} \doteq \{f \circ z_\lambda \mid f : \mathcal{Z}_\lambda \to [\underline{v}, \overline{v}], f \text{ is } L\text{-Lipschitz}\}$

Metric entropy and memory

- For windows of length m, we have $\boxed{H_\epsilon(\mathcal{F}_m) \in \Theta(|\mathcal{Y}|^m)}$
  - $\rightarrow$ Exponential in m: long windows are expensive!
- Memory traces with forgetting factor $\lambda < \frac{1}{2}$ remember everything
  - $\rightarrow$ There is no compression: z_λ is invertible and therefore $\boxed{H_\epsilon(\mathcal{F}_\lambda) = \infty}$
- Need to "zoom in" to differentiate histories that only differ far in the past
- The "resolution" of a function class is given by its Lipschitz constant
- We consider the class $\mathcal{F}_{\lambda,L} \doteq \{f \circ z_\lambda \mid f : \mathcal{Z}_\lambda \to [\underline{v}, \overline{v}], f \text{ is } L\text{-Lipschitz}\}$

Lemma (metric entropy of $\mathcal{F}_{\lambda,L}$)

$$H_\epsilon(\mathcal{F}_{\lambda,L}) \in \mathcal{O}\big(L^{\min\{d_\lambda, |\mathcal{Y}|-1\}}\big) \qquad \text{where} \qquad d_\lambda \doteq \frac{\log|\mathcal{Y}|}{\log(1/\lambda)}$$

Fast forgetting: $\lambda < \frac{1}{2}$

Theorem (window $\to$ trace)

Let $m \in \mathbb{N}$ be a window length, $0 < \lambda < \frac{1}{2}$ a forgetting factor, and define

$$L(m) = \frac{\bar{\nu} - \underline{\nu}}{\sqrt{2(1 - 2\lambda)}\lambda^{m-1}}.$$

Windows are not more efficient than memory traces.

Then, for every $\epsilon > 0$ and every environment $\mathcal{E}$,

$$\mathcal{R}_{\mathcal{E}}\big(\mathcal{F}_{\lambda,L(m)}\big) \leqslant \mathcal{R}_{\mathcal{E}}(\mathcal{F}_m) \quad \text{and} \quad H_\epsilon\big(\mathcal{F}_{\lambda,L(m)}\big) \in \mathcal{O}(|\mathcal{Y}|^m) = \mathcal{O}(H_\epsilon(\mathcal{F}_m)).$$

Fast forgetting: $\lambda < \frac{1}{2}$

Theorem (window $\to$ trace)

Let $m \in \mathbb{N}$ be a window length, $0 < \lambda < \frac{1}{2}$ a forgetting factor, and define

$$L(m) = \frac{\overline{\nu} - \underline{\nu}}{\sqrt{2}(1 - 2\lambda)\lambda^{m-1}}.$$

Then, for every $\epsilon > 0$ and every environment $\mathcal{E}$,

$$\mathcal{R}_{\mathcal{E}}\big(\mathcal{F}_{\lambda,L(m)}\big) \leqslant \mathcal{R}_{\mathcal{E}}(\mathcal{F}_m) \quad \text{and} \quad H_\epsilon\big(\mathcal{F}_{\lambda,L(m)}\big) \in \mathcal{O}(|\mathcal{Y}|^m) = \mathcal{O}(H_\epsilon(\mathcal{F}_m)).$$

Fast forgetting: $\lambda < \frac{1}{2}$

Theorem (trace → window)

Let $\lambda \in [0, 1)$ be a forgetting factor, $L > 0$ a Lipschitz constant, $\epsilon \in (0, L)$, and define

$$m(\lambda, L) = \left\lceil \frac{\log(L/\epsilon)}{\log(1/\lambda)} \right\rceil.$$

Then, [illegible]

$$\mathcal{R}_\varepsilon\big(\mathcal{F}_{m(\lambda,L)}\big) \leq \mathcal{R}_\varepsilon(\mathcal{F}_{\lambda,L}) + \mathcal{O}(\epsilon) \quad \text{and} \quad H_\epsilon\big(\mathcal{F}_{m(\lambda,L)}\big) \in \mathcal{O}(L^{d_\lambda}).$$

If $\lambda < \frac{1}{2}$, then $d_\lambda < |\mathcal{Y}| - 1$.

Memory traces with $\lambda < \frac{1}{2}$ seem no more efficient than windows.

Fast forgetting: $\lambda < \frac{1}{2}$

Theorem (trace $\to$ window)

Let $\lambda \in [0, 1)$ be a forgetting factor, $L > 0$ a Lipschitz constant, $\epsilon \in (0, L)$, and define

$$m(\lambda, L) = \left\lceil \frac{\log(L/\epsilon)}{\log(1/\lambda)} \right\rceil.$$

Then, for every environment $\mathcal{E}$,

$$\mathcal{R}_{\mathcal{E}}\big(\mathcal{F}_{m(\lambda,L)}\big) \leqslant \mathcal{R}_{\mathcal{E}}(\mathcal{F}_{\lambda,L}) + \mathcal{O}(\epsilon) \quad \text{and} \quad H_{\epsilon}\big(\mathcal{F}_{m(\lambda,L)}\big) \in \mathcal{O}(L^{d_\lambda}).$$

If $\lambda < \frac{1}{2}$, then $d_\lambda < |\mathcal{Y}| - 1$.

Slow forgetting: $\lambda \geqslant \frac{1}{2}$

Theorem (T-maze)

There exists a sequence $(\mathcal{E}_k)$ of environments (with constant observation space $\mathcal{Y}$) with the property that, for every $\epsilon > 0$,

$$\min\{H_\epsilon(\mathcal{F}_m) \mid \mathcal{R}_{\mathcal{E}_k}(\mathcal{F}_m) = 0\} \in \Omega(|\mathcal{Y}|^k), \text{ and}$$
$$\min_{\lambda\in[0,1)} \min_{L\geqslant 0}\{H_\epsilon(\mathcal{F}_{\lambda,L}) \mid \mathcal{R}_{\mathcal{E}_k}(\mathcal{F}_{\lambda,L}) = 0\} \in \mathcal{O}(k^{|\mathcal{Y}|-1}).$$

In particular, the *T-maze* with corridor length k is such a sequence. In this case, the minima are attained at $m_k = k$, $\lambda_k = \frac{k-1}{k}$, and $L_k \leqslant \sqrt{2}ek$.

Memory traces ($\lambda \geqslant \frac{1}{2}$) can be significantly more efficient than windows.

Slow forgetting: $\lambda \geqslant \frac{1}{2}$

Theorem (T-maze)

There exists a sequence $(\mathcal{E}_k)$ of environments (with constant observation space $\mathcal{Y}$) with the property that, for every $\epsilon > 0$,

$$\min_{m \in \mathbb{N}} \{H_\epsilon(\mathcal{F}_m) \mid \mathcal{R}_{\mathcal{E}_k}(\mathcal{F}_m) = 0\} \in \Omega(|\mathcal{Y}|^k), \text{ and}$$

$$\min_{\lambda \in [0,1)} \min_{L \geqslant 0} \{H_\epsilon(\mathcal{F}_{\lambda,L}) \mid \mathcal{R}_{\mathcal{E}_k}(\mathcal{F}_{\lambda,L}) = 0\} \in \mathcal{O}(k^{|\mathcal{Y}|-1}).$$

In particular, the *T-maze* with corridor length k is such a sequence. In this case, the minima are attained at $m_k = k$, $\lambda_k = \frac{k-1}{k}$, and $L_k \leqslant \sqrt{2}ek$.

Forgetting: fast and slow

- For $\lambda < \frac{1}{2}$, learning with windows and traces seems equivalent

Forgetting: fast and slow

- For $\lambda < \frac{1}{2}$, learning with windows and traces seems equivalent
- For $\lambda \geqslant \frac{1}{2}$, there exist environments where traces are much more efficient

Why can traces be more efficient?

Forgetting: fast and slow

- For $\lambda < \frac{1}{2}$, learning with windows and traces seems equivalent
- For $\lambda \geqslant \frac{1}{2}$, there exist environments where traces are much more efficient

Why can traces be more efficient?

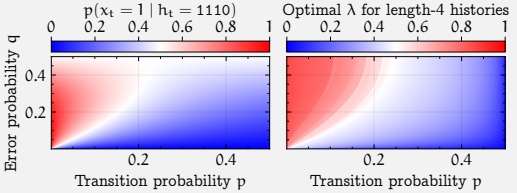
- In the T-maze, most of the $|\mathcal{Y}|^k$ histories are irrelevant $\rightarrow$ allows for small L

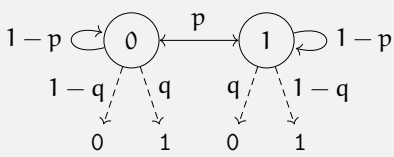
Forgetting: fast and slow

- For $\lambda < \frac{1}{2}$, learning with windows and traces seems equivalent
- For $\lambda \geqslant \frac{1}{2}$, there exist environments where traces are much more efficient

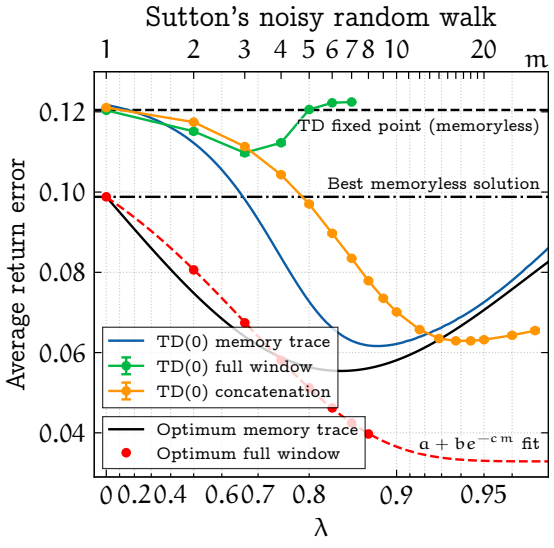
Why can traces be more efficient?

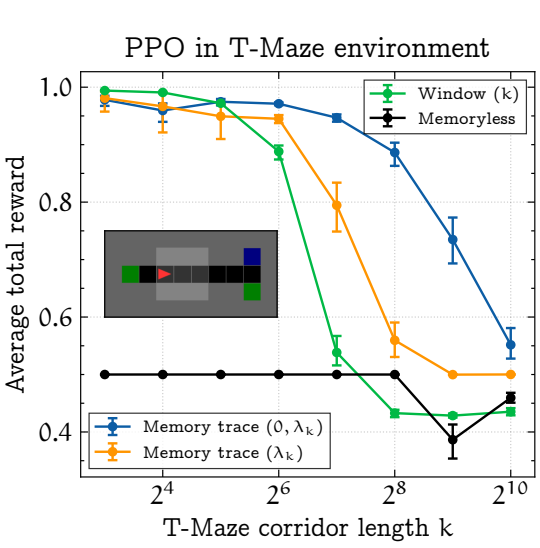
- In the T-maze, most of the $|\mathcal{Y}|^k$ histories are irrelevant $\rightarrow$ allows for small L
- In other environments, memory traces can effectively smooth out noise





Deep RL with memory traces





Summary

- Memory is important in partially observable environments
- We analyze *memory traces*: exponential moving averages of past observations
- There is a close connection to *sliding window* memory
- We prove that memory traces are strictly more powerful than sliding windows
- Memory traces are an effective drop-in replacement for *frame stacking*
- Paper, code & more at `onnoeberhard.com/memory-traces`

Thank you for listening!

Bibliography

Bakker, Bram (2001). “Reinforcement Learning with Long Short-Term Memory”. In: *Advances in Neural Information Processing Systems*. Vol. 14. LINK.