FOPPS Lectures: Probabilistic Bisimulation Metrics and Their Applications to Representation Learning

Lecture 3: Representation learning and the MiCo distance

Prakash Panangaden
School of Computer Science, McGill University
Montreal Institute of Learning Algorithms
School of Informatics, The University of Edinburgh

February 2023, Bertinoro

Outline

Outline

Outline

Outline

Main collaborators

Tyler Kastner, Pablo Castro and Mark Rowland.

Basic goals in RL

- We are often dealing with *large* or *infinite* transition systems whose behaviour is probabilistic.

Basic goals in RL

- We are often dealing with *large* or *infinite* transition systems whose behaviour is probabilistic.
- The system responds to stimuli (actions) and moves to a new state probabilistically and outputs a (possibly) random reward.

Basic goals in RL

- We are often dealing with *large* or *infinite* transition systems whose behaviour is probabilistic.
- The system responds to stimuli (actions) and moves to a new state probabilistically and outputs a (possibly) random reward.
- We seek optimal policies for extracting the largest possible reward in expectation.

Basic goals in RL

- We are often dealing with *large* or *infinite* transition systems whose behaviour is probabilistic.
- The system responds to stimuli (actions) and moves to a new state probabilistically and outputs a (possibly) random reward.
- We seek optimal policies for extracting the largest possible reward in expectation.
- $V^\pi(s) = \mathbb{E}_{a\sim\pi(s)}[\mathcal{R}_s^a + \gamma\mathbb{E}_{x\sim P_s^a}[V^\pi(x)]]$

Basic goals in RL

- We are often dealing with *large* or *infinite* transition systems whose behaviour is probabilistic.
- The system responds to stimuli (actions) and moves to a new state probabilistically and outputs a (possibly) random reward.
- We seek optimal policies for extracting the largest possible reward in expectation.
- $V^{\pi}(s) = \mathbb{E}_{a \sim \pi(s)}[\mathcal{R}_s^a + \gamma \mathbb{E}_{x \sim P_s^a}[V^{\pi}(x)]]$
- This optimisation problem above appears to have multiple objectives (one for each coordinate of V) there is a policy that simultaneously maximises all coordinates.

Basic goals in RL

- We are often dealing with *large* or *infinite* transition systems whose behaviour is probabilistic.
- The system responds to stimuli (actions) and moves to a new state probabilistically and outputs a (possibly) random reward.
- We seek optimal policies for extracting the largest possible reward in expectation.
- $V^\pi(s) = \mathbb{E}_{a\sim\pi(s)}[\mathcal{R}_s^a + \gamma\mathbb{E}_{x\sim P_s^a}[V^\pi(x)]]$
- This optimisation problem above appears to have multiple objectives (one for each coordinate of V) there is a policy that simultaneously maximises all coordinates.
- This policy can be taken to be deterministic!

Basic goals in RL

- We are often dealing with *large* or *infinite* transition systems whose behaviour is probabilistic.
- The system responds to stimuli (actions) and moves to a new state probabilistically and outputs a (possibly) random reward.
- We seek optimal policies for extracting the largest possible reward in expectation.
- $V^\pi(s) = \mathbb{E}_{a\sim\pi(s)}[\mathcal{R}_s^a + \gamma\mathbb{E}_{x\sim P_s^a}[V^\pi(x)]]$
- This optimisation problem above appears to have multiple objectives (one for each coordinate of V) there is a policy that simultaneously maximises all coordinates.
- This policy can be taken to be deterministic!
- In reinforcement learning, we are often interested in finding, or approximating, from direct interaction with the MDP in question via sample trajectories, without knowledge of the explicit form of the transition probabilities.

Algorithms

- Algorithms to learn from samples are called stochastic approximation algorithms.

Algorithms

- Algorithms to learn from samples are called stochastic approximation algorithms.
- The basic algorithm to learn the value function is called *value iteration*. From this we can greedily extract an optimal policy.

Algorithms

- Algorithms to learn from samples are called stochastic approximation algorithms.
- The basic algorithm to learn the value function is called *value iteration*. From this we can greedily extract an optimal policy.
- An algorithm that directly works by improving the policies is called *policy iteration*.

Representation learning

- For large state spaces, learning value functions $S \times \mathcal{A} \longrightarrow \mathbb{R}$ is not feasible.

Representation learning

- For large state spaces, learning value functions $S \times \mathcal{A} \longrightarrow \mathbb{R}$ is not feasible.
- Instead we define a new space of *features* M and try to come up with an embedding $\phi : S \longrightarrow \mathbb{R}^M$.

Representation learning

- For large state spaces, learning value functions $S \times \mathcal{A} \longrightarrow \mathbb{R}$ is not feasible.
- Instead we define a new space of *features* M and try to come up with an embedding $\phi : S \longrightarrow \mathbb{R}^M$.
- Then we can try to use this to predict values associated with state,action pairs.

Representation learning

- For large state spaces, learning value functions $S \times \mathcal{A} \longrightarrow \mathbb{R}$ is not feasible.
- Instead we define a new space of *features* M and try to come up with an embedding $\phi : S \longrightarrow \mathbb{R}^M$.
- Then we can try to use this to predict values associated with state,action pairs.
- Representation learning means learning such a ϕ.

Representation learning

- For large state spaces, learning value functions $S \times \mathcal{A} \longrightarrow \mathbb{R}$ is not feasible.
- Instead we define a new space of *features* M and try to come up with an embedding $\phi : S \longrightarrow \mathbb{R}^M$.
- Then we can try to use this to predict values associated with state,action pairs.
- Representation learning means learning such a ϕ.
- The elements of M are the "features" that are chosen. They can be based on any kind of knowledge or experience about the task at hand.

Representation learning

- For large state spaces, learning value functions $S \times \mathcal{A} \longrightarrow \mathbb{R}$ is not feasible.
- Instead we define a new space of *features* M and try to come up with an embedding $\phi : S \longrightarrow \mathbb{R}^M$.
- Then we can try to use this to predict values associated with state,action pairs.
- Representation learning means learning such a ϕ.
- The elements of M are the “features” that are chosen. They can be based on any kind of knowledge or experience about the task at hand.
- Can we *learn* representations of the state space that accelerate the learning process?

How to learn representations

- Force the agent to make additional predictions: auxiliary tasks.

How to learn representations

- Force the agent to make additional predictions: auxiliary tasks.
- Typically, one uses extra network parameters: implicit representation shaping.

How to learn representations

- Force the agent to make additional predictions: auxiliary tasks.
- Typically, one uses extra network parameters: implicit representation shaping.
- More likely to learn useful features if it has to solve several related tasks.

How to learn representations

- Force the agent to make additional predictions: auxiliary tasks.
- Typically, one uses extra network parameters: implicit representation shaping.
- More likely to learn useful features if it has to solve several related tasks.
- What is a good auxiliary task if the ultimate goal is to learn value functions?

How to learn representations

- Force the agent to make additional predictions: auxiliary tasks.
- Typically, one uses extra network parameters: implicit representation shaping.
- More likely to learn useful features if it has to solve several related tasks.
- What is a good auxiliary task if the ultimate goal is to learn value functions?
- The bisimulation metric!

How to learn representations

- Force the agent to make additional predictions: auxiliary tasks.
- Typically, one uses extra network parameters: implicit representation shaping.
- More likely to learn useful features if it has to solve several related tasks.
- What is a good auxiliary task if the ultimate goal is to learn value functions?
- The bisimulation metric!
- This was done by several groups (Gelada et al. 2019, Zhang et al. 2021, Agarwal et al. 2021).

How to learn representations

- Force the agent to make additional predictions: auxiliary tasks.
- Typically, one uses extra network parameters: implicit representation shaping.
- More likely to learn useful features if it has to solve several related tasks.
- What is a good auxiliary task if the ultimate goal is to learn value functions?
- The bisimulation metric!
- This was done by several groups (Gelada et al. 2019, Zhang et al. 2021, Agarwal et al. 2021).
- All required some additional assumptions on the MDP.

Problems with bsimulation metrics 1: computational complexity

- We defined T_K in Lecture 2. We can iterate this to compute the bisimulation metric.

Problems with bsimulation metrics 1: computational complexity

- We defined T_K in Lecture 2. We can iterate this to compute the bisimulation metric.
- Contraction rate is γ, so to approximate it with error bounded by ε we need $O(\log \varepsilon / \log \gamma)$ iterations.

Problems with bsimulation metrics 1: computational complexity

- We defined T_K in Lecture 2. We can iterate this to compute the bisimulation metric.
- Contraction rate is γ, so to approximate it with error bounded by ε we need $O(\log \varepsilon / \log \gamma)$ iterations.
- Each iteration requires the computation of $O(|S|^2|\mathcal{A}|)$ instances of W-metric.

Problems with bsimulation metrics 1: computational complexity

- We defined T_K in Lecture 2. We can iterate this to compute the bisimulation metric.
- Contraction rate is γ, so to approximate it with error bounded by ε we need $O(\log \varepsilon / \log \gamma)$ iterations.
- Each iteration requires the computation of $O(|S|^2|\mathcal{A}|)$ instances of W-metric.
- Each instance costs $O(|S|^3)$.

Problems with bsimulation metrics 1: computational complexity

- We defined T_K in Lecture 2. We can iterate this to compute the bisimulation metric.
- Contraction rate is γ, so to approximate it with error bounded by ε we need $O(\log \varepsilon / \log \gamma)$ iterations.
- Each iteration requires the computation of $O(|S|^2|\mathcal{A}|)$ instances of W-metric.
- Each instance costs $O(|S|^3)$.
- Total cost is $O(|S|^5|\mathcal{A}| \log(\varepsilon)/\log(\gamma)$.

Problems 2: Bias in sampling

- Computing T_K requires access to P_s^a for each (s, a)-pair. We do not have access to these things in many learning problems.

Problems 2: Bias in sampling

- Computing T_K requires access to P_s^a for each (s, a)-pair. We do not have access to these things in many learning problems.
- An *unbiased* sampling approach is one such that the mean gives the correct value.

Problems 2: Bias in sampling

- Computing T_K requires access to P_s^a for each (s, a)-pair. We do not have access to these things in many learning problems.
- An *unbiased* sampling approach is one such that the mean gives the correct value.
- Sampling methods proposed for estimating the bisimulation metric are biased.

Problems 3: Lack of connection to non-optimal policies

- The result of Ferns et al. gives a tight connection between the bisimulation metric and the optimal value function.

Problems 3: Lack of connection to non-optimal policies

- The result of Ferns et al. gives a tight connection between the bisimulation metric and the optimal value function.
- But it does not give any connection to non-optimal policies.

Problems 3: Lack of connection to non-optimal policies

- The result of Ferns et al. gives a tight connection between the bisimulation metric and the optimal value function.
- But it does not give any connection to non-optimal policies.
- May not be that useful for algorithms like policy iteration.

A crude approximation to bisimulation

- Castro et al. invented a version that is easy to estimate from samples.

A crude approximation to bisimulation

- Castro et al. invented a version that is easy to estimate from samples.
- In spirit it is closely related to the bisimulation metric but it is a crude approximation

A crude approximation to bisimulation

- Castro et al. invented a version that is easy to estimate from samples.
- In spirit it is closely related to the bisimulation metric but it is a crude approximation
- and is not even technically a metric!

The MICo distance

- MICo: matching under independent couplings.

The MICo distance

- MICo: matching under independent couplings.
- Do not try to find the optimal coupling use a simple known coupling, the independent coupling.

The MICo distance

- MICo: matching under independent couplings.
- Do not try to find the optimal coupling use a simple known coupling, the independent coupling.
- We define a new update $T_M : \mathbb{R}^{S \times S} \longrightarrow \mathbb{R}^{S \times S}$ instead of T_K.

The MICo distance

- MICo: matching under independent couplings.
- Do not try to find the optimal coupling use a simple known coupling, the independent coupling.
- We define a new update $T_M : \mathbb{R}^{S \times S} \longrightarrow \mathbb{R}^{S \times S}$ instead of T_K.
- We define $r^\pi(x) := \mathbb{E}_{a \sim \pi(s)}[\mathcal{R}(x, a)]$ and

The MICo distance

- MICo: matching under independent couplings.
- Do not try to find the optimal coupling use a simple known coupling, the independent coupling.
- We define a new update $T_M : \mathbb{R}^{S\times S} \longrightarrow \mathbb{R}^{S\times S}$ instead of T_K.
- We define $r^\pi(x) := \mathbb{E}_{a\sim\pi(s)}[\mathcal{R}(x,a)]$ and
- $P^\pi(x) = \sum_a \pi(x)(a)P^a(x)$

The MICo distance

- MICo: matching under independent couplings.
- Do not try to find the optimal coupling use a simple known coupling, the independent coupling.
- We define a new update $T_M : \mathbb{R}^{S \times S} \longrightarrow \mathbb{R}^{S \times S}$ instead of T_K.
- We define $r^\pi(x) := \mathbb{E}_{a \sim \pi(s)}[\mathcal{R}(x, a)]$ and
- $P^\pi(x) = \sum_a \pi(x)(a) P^a(x)$
- $(T_M^\pi U)(x, y) = |r^\pi(x) - r^\pi(y)| + \gamma \mathbb{E}_{x' \sim P^\pi(x), y' \sim P^\pi(y)}[U(x', y')]$.

The MICo distance

- MICo: matching under independent couplings.
- Do not try to find the optimal coupling use a simple known coupling, the independent coupling.
- We define a new update $T_M : \mathbb{R}^{S\times S} \longrightarrow \mathbb{R}^{S\times S}$ instead of T_K.
- We define $r^\pi(x) := \mathbb{E}_{a\sim\pi(s)}[\mathcal{R}(x,a)]$ and
- $P^\pi(x) = \sum_a \pi(x)(a)P^a(x)$
- $(T^\pi_M U)(x,y) = |r^\pi(x) - r^\pi(y)| + \gamma\mathbb{E}_{x'\sim P^\pi(x), y'\sim\ P^\pi(y)}[U(x',y')]$.
- If we use the L^∞ norm, T_M is a contraction so we have a fixed point by Banach's fixed point theorem.

The MICo distance

- MICo: matching under independent couplings.
- Do not try to find the optimal coupling use a simple known coupling, the independent coupling.
- We define a new update $T_M : \mathbb{R}^{S\times S} \longrightarrow \mathbb{R}^{S\times S}$ instead of T_K.
- We define $r^\pi(x) := \mathbb{E}_{a\sim\pi(s)}[\mathcal{R}(x,a)]$ and
- $P^\pi(x) = \sum_a \pi(x)(a)P^a(x)$
- $(T_M^\pi U)(x,y) = |r^\pi(x) - r^\pi(y)| + \gamma\mathbb{E}_{x'\sim P^\pi(x), y'\sim\ P^\pi(y)}[U(x',y')]$.
- If we use the L^∞ norm, T_M is a contraction so we have a fixed point by Banach's fixed point theorem.
- Call the fixed point U^π.

The MICo distance

- MICo: matching under independent couplings.
- Do not try to find the optimal coupling use a simple known coupling, the independent coupling.
- We define a new update $T_M : \mathbb{R}^{S \times S} \longrightarrow \mathbb{R}^{S \times S}$ instead of T_K.
- We define $r^\pi(x) := \mathbb{E}_{a \sim \pi(s)}[\mathcal{R}(x, a)]$ and
- $P^\pi(x) = \sum_a \pi(x)(a) P^a(x)$
- $(T^\pi_M U)(x, y) = |r^\pi(x) - r^\pi(y)| + \gamma \mathbb{E}_{x' \sim P^\pi(x), y' \sim \, P^\pi(y)}[U(x', y')]$.
- If we use the L^∞ norm, T_M is a contraction so we have a fixed point by Banach's fixed point theorem.
- Call the fixed point U^π.
- For any policy π, we have $|V^\pi(x) - V^\pi(y)| \leq U^\pi(x, y)$.

The MICo distance

- MICo: matching under independent couplings.
- Do not try to find the optimal coupling use a simple known coupling, the independent coupling.
- We define a new update $T_M : \mathbb{R}^{S\times S} \longrightarrow \mathbb{R}^{S\times S}$ instead of T_K.
- We define $r^\pi(x) := \mathbb{E}_{a\sim\pi(s)}[\mathcal{R}(x,a)]$ and
- $P^\pi(x) = \sum_a \pi(x)(a)P^a(x)$
- $(T_M^\pi U)(x,y) = |r^\pi(x) - r^\pi(y)| + \gamma\mathbb{E}_{x'\sim P^\pi(x), y'\sim\ P^\pi(y)}[U(x',y')]$.
- If we use the L^∞ norm, T_M is a contraction so we have a fixed point by Banach's fixed point theorem.
- Call the fixed point U^π.
- For any policy π, we have $|V^\pi(x) - V^\pi(y)| \leq U^\pi(x,y)$.
- Of course this will not give us a metric!

The MICo distance

- MICo: matching under independent couplings.
- Do not try to find the optimal coupling use a simple known coupling, the independent coupling.
- We define a new update $T_M : \mathbb{R}^{S\times S} \longrightarrow \mathbb{R}^{S\times S}$ instead of T_K.
- We define $r^\pi(x) := \mathbb{E}_{a\sim\pi(s)}[\mathcal{R}(x,a)]$ and
- $P^\pi(x) = \sum_a \pi(x)(a) P^a(x)$
- $(T^\pi_M U)(x,y) = |r^\pi(x) - r^\pi(y)| + \gamma \mathbb{E}_{x'\sim P^\pi(x), y'\sim\ P^\pi(y)}[U(x',y')]$.
- If we use the L^∞ norm, T_M is a contraction so we have a fixed point by Banach's fixed point theorem.
- Call the fixed point U^π.
- For any policy π, we have $|V^\pi(x) - V^\pi(y)| \leq U^\pi(x,y)$.
- Of course this will not give us a metric!
- But who knows, maybe it tells us something good.

The MICo distance

- MICo: matching under independent couplings.
- Do not try to find the optimal coupling use a simple known coupling, the independent coupling.
- We define a new update $T_M : \mathbb{R}^{S\times S} \longrightarrow \mathbb{R}^{S\times S}$ instead of T_K.
- We define $r^\pi(x) := \mathbb{E}_{a\sim\pi(s)}[\mathcal{R}(x,a)]$ and
- $P^\pi(x) = \sum_a \pi(x)(a)P^a(x)$
- $(T^\pi_M U)(x,y) = |r^\pi(x) - r^\pi(y)| + \gamma\mathbb{E}_{x'\sim P^\pi(x), y'\sim\ P^\pi(y)}[U(x',y')]$.
- If we use the L^∞ norm, T_M is a contraction so we have a fixed point by Banach's fixed point theorem.
- Call the fixed point U^π.
- For any policy π, we have $|V^\pi(x) - V^\pi(y)| \leq U^\pi(x,y)$.
- Of course this will not give us a metric!
- But who knows, maybe it tells us something good.
- Complexity is $O(|S|^4)$ still not good, but Google has fancy hardware!

What good is MICo?

- Computational complexity down to $O(|S|^4)$, a bit better. Also no factor of $|\mathcal{A}|$ since we are sticking to a particular policy.

What good is MICo?

- Computational complexity down to $O(|S|^4)$, a bit better. Also no factor of $|\mathcal{A}|$ since we are sticking to a particular policy.
- We can use online updates rather than iterating the actual T_M operator.

What good is MICo?

- Computational complexity down to $O(|S|^4)$, a bit better. Also no factor of $|\mathcal{A}|$ since we are sticking to a particular policy.
- We can use online updates rather than iterating the actual T_M operator.
- If stepsizes $(\varepsilon_t(x, y))$ decrease according to some specific conditions (Robbins-Munro) then we get convergence for the following sequence of updates

$$U_{t+1}(x, y) \to (1 - \varepsilon_t(x, y))U_t(x, y) + \varepsilon_t(x, y)(|r - \tilde{r}| + \gamma U_t(x', y'))$$

What good is MICo?

- Computational complexity down to $O(|S|^4)$, a bit better. Also no factor of $|\mathcal{A}|$ since we are sticking to a particular policy.
- We can use online updates rather than iterating the actual T_M operator.
- If stepsizes $(\varepsilon_t(x,y))$ decrease according to some specific conditions (Robbins-Munro) then we get convergence for the following sequence of updates

$$U_{t+1}(x,y) \rightarrow (1-\varepsilon_t(x,y))U_t(x,y) + \varepsilon_t(x,y)(|r-\tilde{r}| + \gamma U_t(x',y'))$$

- where we are updating using a pair of transitions (x_t, a_t, r_t, x'_t) and $(y_t, b_t, \tilde{r}_t, y'_t)$.

What good is MICo?

- Computational complexity down to $O(|S|^4)$, a bit better. Also no factor of $|\mathcal{A}|$ since we are sticking to a particular policy.
- We can use online updates rather than iterating the actual T_M operator.
- If stepsizes $(\varepsilon_t(x, y))$ decrease according to some specific conditions (Robbins-Munro) then we get convergence for the following sequence of updates

$$U_{t+1}(x, y) \to (1 - \varepsilon_t(x, y))U_t(x, y) + \varepsilon_t(x, y)(|r - \tilde{r}| + \gamma U_t(x', y'))$$

- where we are updating using a pair of transitions (x_t, a_t, r_t, x'_t) and $(y_t, b_t, \tilde{r}_t, y'_t)$.
- $|V^\pi(x) - V^\pi(y)| \leq U^{\pi(x,y)}$.

A new type of distance

Diffuse metric

A new type of distance

Diffuse metric

1. $d(x, y) \geq 0$

A new type of distance

Diffuse metric

1. $d(x, y) \geq 0$
2. $d(x, y) = d(y, x)$

A new type of distance

Diffuse metric

1. $d(x, y) \geq 0$
2. $d(x, y) = d(y, x)$
3. $d(x, y) \leq d(x, z) + d(z, y)$

A new type of distance

Diffuse metric

1. $d(x,y) \geq 0$
2. $d(x,y) = d(y,x)$
3. $d(x,y) \leq d(x,z) + d(z,y)$
4. Do not require $d(x,x) = 0$

What is MICo?

Similar to, but not the same as, partial metrics (Matthews) or weak partial pseudometrics (Heckmann). They require stronger conditions than our triangle and they can then extract a real metric and something like a “norm”. Our examples violate their conditions.

What is MICo?

Similar to, but not the same as, partial metrics (Matthews) or weak partial pseudometrics (Heckmann). They require stronger conditions than our triangle and they can then extract a real metric and something like a “norm”. Our examples violate their conditions.

MICo distance is a diffuse metric.

MICo loss

- Nearly all machine learning algorithms are optimization algorithms.

MICo loss

- Nearly all machine learning algorithms are optimization algorithms.
- One often introduces extra terms into the objective function that push the solution in a desired direction.

MICo loss

- Nearly all machine learning algorithms are optimization algorithms.
- One often introduces extra terms into the objective function that push the solution in a desired direction.
- We defined a loss term based on the MICo distance.

MICo loss

- Nearly all machine learning algorithms are optimization algorithms.
- One often introduces extra terms into the objective function that push the solution in a desired direction.
- We defined a loss term based on the MICo distance.
- We assume a value-based agent learning as estimate based on two function approximators ψ, ϕ with their own sets of parameters.

MICo loss

- Nearly all machine learning algorithms are optimization algorithms.
- One often introduces extra terms into the objective function that push the solution in a desired direction.
- We defined a loss term based on the MICo distance.
- We assume a value-based agent learning as estimate based on two function approximators ψ, ϕ with their own sets of parameters.
- We then define a loss term based on the MICo distance.

MICo loss

- Nearly all machine learning algorithms are optimization algorithms.
- One often introduces extra terms into the objective function that push the solution in a desired direction.
- We defined a loss term based on the MICo distance.
- We assume a value-based agent learning as estimate based on two function approximators ψ, ϕ with their own sets of parameters.
- We then define a loss term based on the MICo distance.
- A crucial quantity that emerges is $\Pi U^\pi := U^\pi(x,y) - \frac{1}{2}U^\pi(x,x) - \frac{1}{2}U^\pi(y,y)$.

MICo loss

- Nearly all machine learning algorithms are optimization algorithms.
- One often introduces extra terms into the objective function that push the solution in a desired direction.
- We defined a loss term based on the MICo distance.
- We assume a value-based agent learning as estimate based on two function approximators ψ, ϕ with their own sets of parameters.
- We then define a loss term based on the MICo distance.
- A crucial quantity that emerges is $\Pi U^{\pi} := U^{\pi}(x, y) - \frac{1}{2}U^{\pi}(x, x) - \frac{1}{2}U^{\pi}(y, y)$.
- For details as well as implementation and experiments read `https://psc-g.github.io/posts/research/rl/mico/`.

Recap of Hilbert spaces

- Hilbert to von Neumann: “What are these Hilbert spaces you keep talking about?”

Recap of Hilbert spaces

- Hilbert to von Neumann: "What are these Hilbert spaces you keep talking about?"
- A Hilbert space $\mathcal{H}$ is a: (i) Vector space

Recap of Hilbert spaces

- Hilbert to von Neumann: “What are these Hilbert spaces you keep talking about?”
- A Hilbert space $\mathcal{H}$ is a: (i) Vector space
- with a (ii) inner product $\langle \cdot, \ \cdot \rangle : \mathcal{H} \times \mathcal{H} \longrightarrow \mathbb{R}$ (or $\mathbb{C}$),

Recap of Hilbert spaces

- Hilbert to von Neumann: “What are these Hilbert spaces you keep talking about?”
- A Hilbert space $\mathcal{H}$ is a: (i) Vector space
- with a (ii) inner product $\langle \cdot, \ \cdot \rangle : \mathcal{H} \times \mathcal{H} \longrightarrow \mathbb{R}$ (or $\mathbb{C}$),
- satisfying the usual axioms (sesquilinear for $\mathbb{C}$).

Recap of Hilbert spaces

- Hilbert to von Neumann: “What are these Hilbert spaces you keep talking about?”
- A Hilbert space $\mathcal{H}$ is a: (i) Vector space
- with a (ii) inner product $\langle \cdot, \cdot \rangle : \mathcal{H} \times \mathcal{H} \to \mathbb{R}$ (or $\mathbb{C}$),
- satisfying the usual axioms (sesquilinear for $\mathbb{C}$).
- The inner product induces a *norm* which induces a *metric*.

Recap of Hilbert spaces

- Hilbert to von Neumann: “What are these Hilbert spaces you keep talking about?”
- A Hilbert space $\mathcal{H}$ is a: (i) Vector space
- with a (ii) inner product $\langle \cdot, \ \cdot \rangle : \mathcal{H} \times \mathcal{H} \to \mathbb{R}$ (or $\mathbb{C}$),
- satisfying the usual axioms (sesquilinear for $\mathbb{C}$).
- The inner product induces a *norm* which induces a *metric*.
- The space $\mathcal{H}$ is **complete** with respect to this metric.

Recap of Hilbert spaces

- Hilbert to von Neumann: “What are these Hilbert spaces you keep talking about?”
- A Hilbert space $\mathcal{H}$ is a: (i) Vector space
- with a (ii) inner product $\langle \cdot, \cdot \rangle : \mathcal{H} \times \mathcal{H} \longrightarrow \mathbb{R}$ (or $\mathbb{C}$),
- satisfying the usual axioms (sesquilinear for $\mathbb{C}$).
- The inner product induces a *norm* which induces a *metric*.
- The space $\mathcal{H}$ is **complete** with respect to this metric.
- Examples: $\mathbb{R}^n$ with the euclidean inner product, ℓ^2, $L^2(\mathbb{R})$.

Recap of Hilbert spaces

- Hilbert to von Neumann: “What are these Hilbert spaces you keep talking about?”
- A Hilbert space $\mathcal{H}$ is a: (i) Vector space
- with a (ii) inner product $\langle \cdot, \ \cdot \rangle : \mathcal{H} \times \mathcal{H} \longrightarrow \mathbb{R}$ (or $\mathbb{C}$),
- satisfying the usual axioms (sesquilinear for $\mathbb{C}$).
- The inner product induces a *norm* which induces a *metric*.
- The space $\mathcal{H}$ is **complete** with respect to this metric.
- Examples: $\mathbb{R}^n$ with the euclidean inner product, ℓ^2, $L^2(\mathbb{R})$.
- Be careful of L^2, its elements are *not* functions but equivalence classes of *almost everywhere equal* functions.

Duality and Riesz

- A function between Hilbert spaces is continuous iff it is bounded.

Duality and Riesz

- A function between Hilbert spaces is continuous iff it is bounded.
- The space of bounded linear functions from $\mathcal{H}$ to $\mathbb{R}$ is itself a Hilbert space called $\mathcal{H}^*$ which is *isomorphic* to $\mathcal{H}$.

Duality and Riesz

- A function between Hilbert spaces is continuous iff it is bounded.
- The space of bounded linear functions from $\mathcal{H}$ to $\mathbb{R}$ is itself a Hilbert space called $\mathcal{H}^*$ which is *isomorphic* to $\mathcal{H}$.
- Given an element $v \in \mathcal{H}$, we have a map $\lambda_v : \mathcal{H} \longrightarrow \mathbb{R}$ given by $x \mapsto \langle v, x \rangle$; this is bounded and linear.

Duality and Riesz

- A function between Hilbert spaces is continuous iff it is bounded.
- The space of bounded linear functions from $\mathcal{H}$ to $\mathbb{R}$ is itself a Hilbert space called $\mathcal{H}^*$ which is *isomorphic* to $\mathcal{H}$.
- Given an element $v \in \mathcal{H}$, we have a map $\lambda_v : \mathcal{H} \longrightarrow \mathbb{R}$ given by $x \mapsto \langle v,\ x \rangle$; this is bounded and linear.
- The Riesz representation theorem says: *all* bounded linear maps arise this way.

Duality and Riesz

- A function between Hilbert spaces is continuous iff it is bounded.
- The space of bounded linear functions from $\mathcal{H}$ to $\mathbb{R}$ is itself a Hilbert space called $\mathcal{H}^*$ which is *isomorphic* to $\mathcal{H}$.
- Given an element $v \in \mathcal{H}$, we have a map $\lambda_v : \mathcal{H} \longrightarrow \mathbb{R}$ given by $x \mapsto \langle v, x \rangle$; this is bounded and linear.
- The Riesz representation theorem says: *all* bounded linear maps arise this way.
- For any bounded linear function $\lambda : \mathcal{H} \longrightarrow \mathbb{R}$ $\exists! l \in \mathcal{H}$ such that $\langle l, x \rangle = \lambda(x)$.

Definition of RKHS

- RKHS is “reproducing kernel Hilbert space”.

Definition of RKHS

- RKHS is "reproducing kernel Hilbert space".
- Let X be a set, no other structure assumed.

Definition of RKHS

- RKHS is “reproducing kernel Hilbert space”.
- Let X be a set, no other structure assumed.
- The vector space of all functions from X to $\mathbb{R}$ is written $\mathcal{F}(X, \mathbb{R})$ or just $\mathcal{F}(X)$.

Definition of RKHS

- RKHS is "reproducing kernel Hilbert space".
- Let X be a set, no other structure assumed.
- The vector space of all functions from X to $\mathbb{R}$ is written $\mathcal{F}(X, \mathbb{R})$ or just $\mathcal{F}(X)$.
- $\mathcal{H}$ is an RKHS on X if it is a vector subspace of $\mathcal{F}(X)$.

Definition of RKHS

- RKHS is "reproducing kernel Hilbert space".
- Let X be a set, no other structure assumed.
- The vector space of all functions from X to $\mathbb{R}$ is written $\mathcal{F}(X, \mathbb{R})$ or just $\mathcal{F}(X)$.
- $\mathcal{H}$ is an RKHS on X if it is a vector subspace of $\mathcal{F}(X)$.
- $\mathcal{H}$ has an inner product $\langle \cdot, \ \cdot \rangle$ making it into a Hilbert space.

Definition of RKHS

- RKHS is "reproducing kernel Hilbert space".
- Let X be a set, no other structure assumed.
- The vector space of all functions from X to $\mathbb{R}$ is written $\mathcal{F}(X, \mathbb{R})$ or just $\mathcal{F}(X)$.
- $\mathcal{H}$ is an RKHS on X if it is a vector subspace of $\mathcal{F}(X)$.
- $\mathcal{H}$ has an inner product $\langle \cdot, \ \cdot \rangle$ making it into a Hilbert space.
- For *every* $x \in X$, the function $ev_x : \mathcal{H} \longrightarrow \mathbb{R}$ given by $f \mapsto f(x)$ is bounded (hence continuous).

Definition of RKHS

- RKHS is "reproducing kernel Hilbert space".
- Let X be a set, no other structure assumed.
- The vector space of all functions from X to $\mathbb{R}$ is written $\mathcal{F}(X, \mathbb{R})$ or just $\mathcal{F}(X)$.
- $\mathcal{H}$ is an RKHS on X if it is a vector subspace of $\mathcal{F}(X)$.
- $\mathcal{H}$ has an inner product $\langle \cdot, \ \cdot \rangle$ making it into a Hilbert space.
- For *every* $x \in X$, the function $ev_x : \mathcal{H} \longrightarrow \mathbb{R}$ given by $f \mapsto f(x)$ is bounded (hence continuous).
- By Riesz, $\forall x \in X, \exists ! k_x \in \mathcal{H}$ such that $\forall f \in \mathcal{H}, \langle k_x, \ f \rangle = f(x)$.

Definition of RKHS

- RKHS is "reproducing kernel Hilbert space".
- Let X be a set, no other structure assumed.
- The vector space of all functions from X to $\mathbb{R}$ is written $\mathcal{F}(X, \mathbb{R})$ or just $\mathcal{F}(X)$.
- $\mathcal{H}$ is an RKHS on X if it is a vector subspace of $\mathcal{F}(X)$.
- $\mathcal{H}$ has an inner product $\langle \cdot, \ \cdot \rangle$ making it into a Hilbert space.
- For *every* $x \in X$, the function $ev_x : \mathcal{H} \longrightarrow \mathbb{R}$ given by $f \mapsto f(x)$ is bounded (hence continuous).
- By Riesz, $\forall x \in X, \exists! k_x \in \mathcal{H}$ such that $\forall f \in \mathcal{H}, \langle k_x, \ f \rangle = f(x)$.
- Note that $L^2(\mathbb{R})$ is **not** an RKHS.

Definition of RKHS

- RKHS is “reproducing kernel Hilbert space”.
- Let X be a set, no other structure assumed.
- The vector space of all functions from X to $\mathbb{R}$ is written $\mathcal{F}(X, \mathbb{R})$ or just $\mathcal{F}(X)$.
- $\mathcal{H}$ is an RKHS on X if it is a vector subspace of $\mathcal{F}(X)$.
- $\mathcal{H}$ has an inner product $\langle \cdot, \ \cdot \rangle$ making it into a Hilbert space.
- For *every* $x \in X$, the function $ev_x : \mathcal{H} \longrightarrow \mathbb{R}$ given by $f \mapsto f(x)$ is bounded (hence continuous).
- By Riesz, $\forall x \in X, \exists! k_x \in \mathcal{H}$ such that $\forall f \in \mathcal{H}, \langle k_x, \ f \rangle = f(x)$.
- Note that $L^2(\mathbb{R})$ is **not** an RKHS.
- It is easy to construct a sequence of functions f_n such that $\lim_{n \longrightarrow \infty} ||f_n|| = 0$ but there is a point x such that $\lim_{n \longrightarrow \infty} f_n(x) = \infty$.

Definition of RKHS

- RKHS is "reproducing kernel Hilbert space".
- Let X be a set, no other structure assumed.
- The vector space of all functions from X to $\mathbb{R}$ is written $\mathcal{F}(X,\mathbb{R})$ or just $\mathcal{F}(X)$.
- $\mathcal{H}$ is an RKHS on X if it is a vector subspace of $\mathcal{F}(X)$.
- $\mathcal{H}$ has an inner product $\langle\cdot,\ \cdot\rangle$ making it into a Hilbert space.
- For *every* $x \in X$, the function $ev_x : \mathcal{H} \longrightarrow \mathbb{R}$ given by $f \mapsto f(x)$ is bounded (hence continuous).
- By Riesz, $\forall x \in X, \exists! k_x \in \mathcal{H}$ such that $\forall f \in \mathcal{H}, \langle k_x,\ f\rangle = f(x)$.
- Note that $L^2(\mathbb{R})$ is **not** an RKHS.
- It is easy to construct a sequence of functions f_n such that $\lim_{n\longrightarrow\infty} ||f_n|| = 0$ but there is a point x such that $\lim_{n\longrightarrow\infty} f_n(x) = \infty$.
- $\ell^2(\mathbb{N})$ is an RKHS.

Why this strange name?

- Note that k_x is a function of type $X \longrightarrow \mathbb{R}$.

Why this strange name?

- Note that k_x is a function of type $X \longrightarrow \mathbb{R}$.
- So we can apply it to $y \in X$.

Why this strange name?

- Note that k_x is a function of type $X \longrightarrow \mathbb{R}$.
- So we can apply it to $y \in X$.
- We define $K(x, y) = k_x(y) = \langle k_x,\ k_y \rangle$.

Why this strange name?

- Note that k_x is a function of type $X \longrightarrow \mathbb{R}$.
- So we can apply it to $y \in X$.
- We define $K(x, y) = k_x(y) = \langle k_x,\ k_y \rangle$.
- K is called the kernel of the RKHS.

Constructing RKHS on finite sets

- Let X be a finite set.

Constructing RKHS on finite sets

- Let X be a finite set.
- We call $k : X \times X \longrightarrow \mathbb{R}$ a positive-definite kernel if it is symmetric and

Constructing RKHS on finite sets

- Let X be a finite set.
- We call $k : X \times X \longrightarrow \mathbb{R}$ a positive-definite kernel if it is symmetric and
- for any $x_1, x_2, \ldots, x_n \in X$ and any $c_1, \ldots, c_n \in \mathbb{R}$ we have

Constructing RKHS on finite sets

- Let X be a finite set.
- We call $k : X \times X \longrightarrow \mathbb{R}$ a positive-definite kernel if it is symmetric and
- for any $x_1, x_2, \ldots, x_n \in X$ and any $c_1, \ldots, c_n \in \mathbb{R}$ we have
- $$\sum_{i=1}^{n} \sum_{j=1}^{n} c_i c_j k(x_i, x_j) \geq 0.$$

Constructing RKHS on finite sets

- Let X be a finite set.
- We call $k : X \times X \longrightarrow \mathbb{R}$ a positive-definite kernel if it is symmetric and
- for any $x_1, x_2, \ldots, x_n \in X$ and any $c_1, \ldots, c_n \in \mathbb{R}$ we have
- $$\sum_{i=1}^{n} \sum_{j=1}^{n} c_i c_j k(x_i, x_j) \geq 0.$$
- We can construct an RKHS $\mathcal{H}_k$ of functions on X with k as its reproducing kernel.

Embeddings

- We can embed X in $\mathcal{H}$ naturally: $x \mapsto k_x =: \varphi(x)$.

Embeddings

- We can embed X in $\mathcal{H}$ naturally: $x \mapsto k_x =: \varphi(x)$.
- We can think of this as embedding a “state space” into a “feature space.”

Embeddings

- We can embed X in $\mathcal{H}$ naturally: $x \mapsto k_x =: \varphi(x)$.
- We can think of this as embedding a “state space” into a “feature space.”
- Given a probability measure μ on X we define $\Phi(\mu) := \int_X \varphi(x)\mathrm{d}\mu \in \mathcal{H}$.

Embeddings

- We can embed X in $\mathcal{H}$ naturally: $x \mapsto k_x =: \varphi(x)$.
- We can think of this as embedding a “state space” into a “feature space.”
- Given a probability measure μ on X we define $\Phi(\mu) := \int_X \varphi(x)\mathrm{d}\mu \in \mathcal{H}$.
- We can show $\langle f,\ \Phi(\mu)\rangle = \int_X f \mathrm{d}\mu$.

Metrics on probability distributions

- Gretton et al. define a metric on probability distributions using the last embedding.

Metrics on probability distributions

- Gretton et al. define a metric on probability distributions using the last embedding.
- $$MMD(k)(\mu, \nu) := ||\Phi(\mu) - \Phi(\nu)||_{\mathcal{H}_k}.$$

Metrics on probability distributions

- Gretton et al. define a metric on probability distributions using the last embedding.
- $$MMD(k)(\mu, \nu) := ||\Phi(\mu) - \Phi(\nu)||_{\mathcal{H}_k}.$$
- MMD stands for “minimum mean discrepancy”.

Metrics on probability distributions

- Gretton et al. define a metric on probability distributions using the last embedding.
- $$MMD(k)(\mu, \nu) := ||\Phi(\mu) - \Phi(\nu)||_{\mathcal{H}_k}.$$
- MMD stands for "minimum mean discrepancy".
- This has a close connection with so-called "energy distances" which are used in statistics and machine learning.

Kernels instead of metrics

- Instead of quantifying the difference between states through a metric we quantify their “similarity” through a kernel.

Kernels instead of metrics

- Instead of quantifying the difference between states through a metric we quantify their "similarity" through a kernel.
- Metrics like bisimulation have the following pattern: $d(x, y) = d_1(x, y) + const \cdot d_2(\mathcal{P}(x), \mathcal{P}(y))$.

Kernels instead of metrics

- Instead of quantifying the difference between states through a metric we quantify their "similarity" through a kernel.
- Metrics like bisimulation have the following pattern: $d(x,y) = d_1(x,y) + const \cdot d_2(\mathcal{P}(x), \mathcal{P}(y))$.
- Here d_1 is some kind of "one-step" difference and d_2 represents what happens later.

Kernels instead of metrics

- Instead of quantifying the difference between states through a metric we quantify their "similarity" through a kernel.
- Metrics like bisimulation have the following pattern: $d(x,y) = d_1(x,y) + const \cdot d_2(\mathcal{P}(x), \mathcal{P}(y))$.
- Here d_1 is some kind of "one-step" difference and d_2 represents what happens later.
- We will follow a similar pattern with kernels.

Outline of the approach

- We make the space of kernels into a complete metric space.

Outline of the approach

- We make the space of kernels into a complete metric space.
- We define an "update operator" on the space of kernels and show that it is contractive, hence has a unique fixed point.

Outline of the approach

- We make the space of kernels into a complete metric space.
- We define an “update operator” on the space of kernels and show that it is contractive, hence has a unique fixed point.
- We use this kernel to construct an RKHS.

Outline of the approach

- We make the space of kernels into a complete metric space.
- We define an “update operator” on the space of kernels and show that it is contractive, hence has a unique fixed point.
- We use this kernel to construct an RKHS.
- This gives a (semi)metric which we call the “kernel similarity metric” (ksme).

Outline of the approach

- We make the space of kernels into a complete metric space.
- We define an “update operator” on the space of kernels and show that it is contractive, hence has a unique fixed point.
- We use this kernel to construct an RKHS.
- This gives a (semi)metric which we call the “kernel similarity metric” (ksme).
- This ksme distance is exactly the same as the reduced MICo distance we defined earlier.

Outline of the approach

- We make the space of kernels into a complete metric space.
- We define an “update operator” on the space of kernels and show that it is contractive, hence has a unique fixed point.
- We use this kernel to construct an RKHS.
- This gives a (semi)metric which we call the “kernel similarity metric” (ksme).
- This ksme distance is exactly the same as the reduced MICo distance we defined earlier.
- This approach gives many other interesting results: low-distortion embeddings, bounds on value function differences etc.