Outline

- Motivation for meta-reinforcement learning
- Problem setups (RL, meta learning, meta-RL)
- Common approaches
  - Black-box adaptation (based on recurrent policies)
  - Optimization-based methods
  - Inference-based methods (solving equivalent POMDP)
- Comparison
- Task design (unsupervised meta-RL)
- Summary and conclusions
Outline

- Motivation for meta-reinforcement learning
- Problem setups (RL, meta learning, meta-RL)
- Common approaches
  - Black-box adaptation (based on recurrent policies)
  - Optimization-based methods
  - Inference-based methods (solving equivalent POMDP)
- Comparison
- Task design (unsupervised meta-RL)
- Summary and conclusions
Why do we care about meta-rl?

- Humans can learn new skills very quickly, efficiently adapting to new environments and tasks.
- Can we design algorithms that learn to reinforcement learn?
Outline

- Motivation for meta-reinforcement learning
- **Problem setups (RL, meta learning, meta-RL)**
- **Common approaches**
  - Black-box adaptation (based on recurrent policies)
  - Optimization-based methods
  - Inference-based methods (solving equivalent POMDP)
- **Comparison**
- **Task design (unsupervised meta-RL)**
- **Summary and conclusions**
The reinforcement learning problem

Markov Decision Process: $M = \{S, A, P, r\}$

- $S$: state space
- $A$: action space
- $P$: transition probability, $p: S \times A \rightarrow S$
- $r$: reward function, $r: S \times A \rightarrow \mathbb{R}$

$\pi(a|s)$: the policy, $\pi: S \rightarrow \Delta(A)$ or $\pi: S \rightarrow A$

Transitions: $\{s_t, a_t, r_t, s_{t+1}\}_i$

Trajectory: $\tau = \{s_0, a_0, r_0, s_1, a_1, r_1, ..., s_T, a_T, r_T\}$

The reinforcement learning problem

Goal:
learn a policy that maximizes the expected (discounted) sum of rewards

Parameterized policy (infinite horizon):

\[ \theta^* = \arg \max_\theta \mathbb{E}_{\pi_\theta} \left[ \sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \right] \]

Expectation over (discounted) state visitation distribution
\[ \theta^* = \arg \max_\theta \mathbb{E}_{\pi_\theta(s,a)} [r(s,a)] \]
General procedure

1. get initial state $s_0 \sim p(s)$
2. choose an action from policy $a_t \sim \pi_\theta(\cdot | s_t)$
3. observe reward $r_t = r(s_t, a_t)$ and new state $s_{t+1} \sim p(\cdot | s_t, a_t)$
4. optimize $\theta^* = \arg \max_\theta \mathbb{E}_{\pi_\theta}[R(\tau)]$
5. store experiences $(s_t, a_t, s_{t+1}, r_t)$ in replay buffer
6. repeat until convergence

How do we optimize our policy?

1. policy gradient
2. value function or Q function estimation
3. model learning + MPC
The meta learning problem

\( \theta \): meta parameter
\( \phi_i \): task specific adaptation parameter
\( p(\mathcal{D}) \): a distribution over meta training dataset (or tasks)

learn \( \theta \) such that \( \phi_i = f_\theta(\mathcal{D}_i^{tr}) \) fits \( \mathcal{D}_i^{ts} \) well

**Probabilistic view:**

\[ \theta^* = \arg \max_\theta \sum \log p(\phi_i | \mathcal{D}_i^{ts}) \]

**Deterministic view:**

\[ \theta^* = \arg \min_\theta \sum L_i(\phi_i, \mathcal{D}_i^{ts}) \]
Meta learning + RL

**Traditional (supervised) learning:**
\[ \theta^* = \arg \min_\theta \mathcal{L}(\theta, \mathcal{D}) \]

**Traditional (supervised) meta learning:**
\[ \theta^* = \arg \min_\theta \sum \mathcal{L}_i(\phi_i, \mathcal{D}_i^{ts}) \text{ where } \phi_i = f_\theta(\mathcal{D}_i^{tr}) \]
\[ \mathcal{D}_{meta-train} = \{(D_0^{tr}, D_0^{ts}), (D_1^{tr}, D_1^{ts}), \ldots \} \]

**Meta reinforcement learning:**
\[ \theta^* = \arg \max_\theta \mathbb{E}_{\tau \sim \pi_\theta} [R(\tau)] = f_{RL}(\mathcal{M}) \]

**Reinforcement learning:**
\[ \theta^* = \arg \max_\theta \sum \mathbb{E}_{\tau \sim \pi_{\phi_i}} [R(\tau)] \text{ where } \phi_i = f_\theta(\mathcal{M}_i) \]
\[ \mathcal{D}_{meta-train} = \{\mathcal{M}_0, \mathcal{M}_1, \ldots\} \]
Meta learning RL procedure

1. [initialization] given a distribution over MDPs $p(\mathcal{M})$, draw $\mathcal{M}_i \sim p(\mathcal{M})$
2. [task adaptation] get our policy $\pi_{\phi_i}$ by the meta learner $f_\theta(\mathcal{M}_i)$
3. [data collection] explore or exploit $\mathcal{M}_i$ with $\pi_{\phi_i}$ and collect experiences
4. [meta learning] maximize the meta parameter $\theta$ with collected data
5. repeat
Core problem

\[ \theta^* = \arg \max_{\theta} \sum \mathbb{E}_{\tau \sim \pi_i} [R(\tau)] \text{ where } \phi_i = f_\theta(M_i) \]

How do we design \( f_\theta(M_i) \)? What does \( f_\theta(M_i) \) do?
1. \( f_\theta \) improves the policy with experiences from \( M_i \)
2. \( f_\theta \) can also choose how to interact with \( M_i \) (exploration vs exploitation)
Popular approaches to meta-rl

- Memory-based approach (black-box adaptation)
  - Recurrent policy (RNN, LSTM)
  - Attention + temporal convolution
  - Mean field assumption

- Optimization-based approach
  - MAML and its variants

- POMDP perspective
  - Task inferences and embedding
Outline

- Motivation for meta-reinforcement learning
- Problem setups (RL, meta learning, meta-RL)
- Common approaches
  - Black-box adaptation (based on recurrent policies)
  - Optimization-based methods
  - Inference-based methods (solving equivalent POMDP)
- Comparison
- Task design (unsupervised meta-RL)
- Summary and conclusions
Memory-based approach (black-box adaptation)

- Key idea: in order to learn a "good" prior, we need to somehow 1) "memorize" experiences we've seen so far, 2) and to “adapt” quickly to new tasks with our memory.

- “Good” prior:
  - Internalize the dynamics about the MDP; interactions with previous tasks help future tasks
- “Memorization”:
  - Recurrent networks, temporal convolutions + attentions
- “Adapt”:
  - Few shot experiences from the test MDP lead to a decent policy
Memory-based approach (black-box adaptation)

- Key idea: in order to learn a "good" prior, we need to somehow 1) "memorize" experiences we've seen so far, 2) and to "adapt" quickly to new tasks with our memory.

- Recipe:
  - Augmented “observation space”: include past experience (states, actions, rewards)
  - A policy that takes into account all its past trajectory in a MDP by using this augmented observation (RNN policy for example)

Graph: ICML 2019 tutorial on meta learning
Memory-based approach (black-box adaptation)

- Key idea: in order to learn a "good" prior, we need to somehow 1) "memorize" experiences we've seen so far, 2) and to "adapt" quickly to new tasks with our memory.

- Procedures:
  - Sample a new MDP
  - Reset the hidden state
  - Collect trajectories and update the model by maximizing total return (using RL algorithms)

Memory-based approach (black-box adaptation)

- Key idea: in order to learn a "good" prior, we need to somehow 1) "memorize" experiences we've seen so far, 2) and to “adapt” quickly to new tasks with our memory.

- How to design architectures for the memory?
  - RNN, LSTM, GRU
  - Attention + temporal convolution

Duan et al. RL**2: Fast Reinforcement Learning via Slow Reinforcement Learning [2016]

Mishra, Rohaninejad et al. A Simple Neural Attentive Meta-Learner [2018]
Memory-based approach (black-box adaptation)

- Problems?
  - [Learnability] Memory (gradient vanishing/explosion during BPTT, etc)
  - [Data efficiency] Works mostly in conjunction with on-policy RL algorithms
  - [Optimality] Trade-offs between exploration and exploitation
Outline

● Motivation for meta-reinforcement learning
● Problem setups (RL, meta learning, meta-RL)
● Common approaches
  ○ Black-box adaptation (based on recurrent policies)
  ○ Optimization-based methods
  ○ Inference-based methods (solving equivalent POMDP)
● Comparison
● Task design (unsupervised meta-RL)
● Summary and conclusions
Optimization-based approach

- Most of the works in this category is based on ideas from MAML.

- Learn a proper **initialization** of the parameters so that after **few-shot experiences** from the new MDP, the policy nicely **adapts** to the new task.

- The learned meta parameter lies in the parameter space where it’s close to the optimal task specific parameters on average.

- The meta parameters and the task-specific parameters coincide.
Optimization-based approach

- A quick recap of MAML (meta-rl as an optimization problem)

**Meta reinforcement learning goal:**

\[ \theta^* = \arg \max_{\theta} \sum_{i} \mathbb{E}_{\tau \sim \pi_{\phi_i}} [R(\tau)] \text{ where } \phi_i = f^\text{generic}_\theta (\mathcal{M}_i) \]

In MAML we have \( \theta = \phi \), so the goal of MAML RL:

\[ \theta^* = \arg \max_{\theta} \sum_{i} \mathbb{E}_{\tau \sim \pi_{\theta_i}} [R(\tau)] \text{ where } \theta_i = f^{\text{MAML}}_\theta (\mathcal{M}_i) \]

Where the meta learner takes a specific form:

\[ f^{\text{MAML}}_\theta (\mathcal{M}_i) = \theta + \alpha \nabla J_i(\theta) \]

When in the context of reinforcement learning:

\[ J_i(\theta) = \mathbb{E}_{\tau \sim \pi_{\theta}} [R(\tau)], \text{ the expected sum of rewards in } \mathcal{M}_i \]

Which can be estimated by interacting with \( \mathcal{M}_i \)
Optimization-based approach

- **Recipe:**

  *In traditional RL, we optimize our parameter via:*
  \[ \theta \leftarrow \theta + \alpha \nabla \theta J(\theta) \]

  *In MAML, we optimize our parameter via:*
  \[ \theta \leftarrow \theta + \beta \sum_i \nabla \theta J_i \left( \theta + \alpha \nabla \theta J_i(\theta) \right) \]

- **Interpretations:**
  - Run one iteration of ascent and update our parameter based on how much such one step optimization can help with the task.
  
  - We want to optimize the parameter so that when we later do one step gradient ascent (task adaptation) on the test task, the objective is maximized in expectation (over the task distribution)
Optimization-based approach

- **Recipe:**
  
  In traditional RL, we optimize our parameter via:
  \[ \theta \leftarrow \theta + \alpha \nabla_{\theta} J(\theta) \]

  In MAML, we optimize our parameter via:
  \[ \theta \leftarrow \theta + \beta \sum_i \nabla_{\theta} J_i \left( \theta + \alpha \nabla_{\theta} J_i(\theta) \right) \]

- **Procedure:**
  - Pick a random task \( i \)
  - Make one (or more) gradient step(s) to find its adapted parameter \( \theta + \alpha \nabla_{\theta} J_i(\theta) \)
  - Optimize the objective based on how good this adapted parameter performs \( \nabla_{\theta} J_i(\theta + \alpha \nabla_{\theta} J_i(\theta)) \)
  - So the final parameter results in policy that performs well on average

---

Optimization-based approach

- Procedure:
  - Pick a random task \( i \)
  - Make one (or more) gradient step(s) to find its adapted parameter \( \theta + \alpha \nabla_{\theta} J_i(\theta) \)
  - Optimize the objective based on how good this adapted parameter performs \( \min_{\theta} J_i(\theta + \alpha \nabla_{\theta} J_i(\theta)) \)
  - So the final parameter results in policy that performs well on average

gif. tristandeleu/pytorch-maml-rl
Optimization-based approach

- One (or few) shot learning with new sampled goals in robotic controls
- Major drawbacks
  - Requires Hessian calculation. Tricks for approximation or acceleration?
  - What if the optimal parameters are not in the vicinity of each other in the parameter space? Do we have guarantees on the generalization and adaptation power?

Outline

- Motivation for meta-reinforcement learning
- Problem setups (RL, meta learning, meta-RL)
- Common approaches
  - Black-box adaptation (based on recurrent policies)
  - Optimization-based methods
  - Inference-based methods (solving equivalent POMDP)
- Comparison
- Task design (unsupervised meta-RL)
- Summary and conclusions
POMDP perspective

- How is meta-rl fundamentally different from generic reinforcement learning?
- Or is it different?
- In fact, meta-rl can be seen as a regular reinforcement learning except that the state has to be partially observable.
POMDP perspective

- A quick recap of partially observed Markov decision process

**Augment a regular MDP with observation space and emission probability**

$M = \{S, A, P, r, \mathcal{O}, \mathcal{E}, \mathcal{O}\}$ where

$\mathcal{O}$: observation space

$\mathcal{E}$: emission probability, i.e. $p(o_t|s_t)$

Graph: ICML 2019 tutorial on meta learning
POMDP perspective

- Under POMDPs, policy can only **act on observations** instead of the underlying states.
- POMDPs are known to be **extremely difficult to solve** as it requires reasoning about true states.
- Typically, to solve POMDPs:
  - **State estimations**: model the distribution of states given the observations (history), and apply usual RL procedures to find the optimal policy.
  - **Use policies with memory**: implicitly infer the internal dynamics of the MDP based on previous experiences.
POMDP perspective

- Meta-rl in the lens of regular rl in POMDPs:
  - Key idea:
    1. encapsulate **task-specific information with a latent variable** on which the policy depends
    2. Learning involves **inferring the task context variable** and **optimizing the policy**

Meta RL in the lens of POMDPs:
1. POMDP: \( \tilde{M} = \{ \tilde{S}, \tilde{A}, \tilde{P}, r, \epsilon, O \} \) where

\[
\tilde{S} = S \times Z, \text{ the concatenation of the state space and the task context}
\]
\[
\tilde{P} = p(\tilde{s}_{t+1}|\tilde{s}_t, a_t), \text{ the new transition function on the new state space}
\]
\[
O = S
\]
\[
\epsilon = p(0_t|\tilde{s}_t)
\]
2. policy: \( \pi_\theta(a|s, z) \)
POMDP perspective

Meta RL in the lens of POMDPs:

1. POMDP: \( \tilde{\mathcal{M}} = \{ \tilde{\mathcal{S}}, \tilde{\mathcal{A}}, \tilde{\mathcal{P}}, r, \mathcal{E}, O \} \) where

\[ \tilde{\mathcal{S}} = \mathcal{S} \times \mathcal{Z}, \text{ the concatenation of the state space and the task context} \]

\[ \tilde{\mathcal{P}} = p(\tilde{s}_{t+1}|\tilde{s}_t, a_t), \text{ the new transition function on the new state space} \]

\[ O = \mathcal{S} \]

\[ \mathcal{E} = p(o_t|\tilde{s}_t) \]

2. policy: \( \pi_{\theta}(a|s, \quad \text{task context}) \)

- Remember, typically, to solve POMDPs:
  - **State estimations**: model the distribution of states given the observations (history), and apply usual RL procedures to find the optimal policy
  - **Use policies with memory**: implicitly infer the internal dynamics of the MDP based on previous experiences.
POMDP perspective

$$\tilde{\mathcal{M}} = \{\tilde{S}, \tilde{A}, \tilde{P}, r, \varepsilon, O\}, \text{policy: } \pi_\theta(a|s, z_{\text{task context}})$$

The goal is to estimate the posterior probability of the task context variable given experiences

$$p\left(z_t \mid s_{1:t}, a_{1:t}, r_{1:t}\right)$$

Use locked objects to block the room
POMDP perspective

\[ \tilde{\mathcal{M}} = \left\{ \tilde{S}, \tilde{A}, \tilde{P}, r, \varepsilon, O \right\}, \text{policy: } \pi_\theta(a|s, z) \]

The goal is to estimate the posterior probability of the task context variable given experiences

\[ p\left( z_t \mid s_{1:t}, a_{1:t}, r_{1:t} \right) \]

posterior sampling with latent context:

1. sample the latent variable with our current model
   \[ z \sim \tilde{p}(z_t | s_{1:t}, a_{1:t}, r_{1:t}) \]

2. act according to \( \pi_\theta(a|s, z) \) to collect more data

- Comments on posterior sampling:
  - Often uses variational inference to approximate the posterior
  - Enables exploration
  - Not optimal
  - Works well in practice
POMDP perspective

Goal: optimize both the policy $\pi_\theta(a_t|s_t, z_t)$ and the posterior context variable $q_\lambda(z_t|\tau_{1:t})$

Optimization:

$$(\theta, \lambda) = \text{arg max}_{\theta,\lambda} \frac{1}{N} \sum_{i=1}^{n} \mathbb{E}_{q_\lambda, \pi_\theta} \left[ R_i(\tau) \right] - D_{KL}(q_\lambda(z)||p(z))$$

Comments:
- We can think of the return as the likelihood as in VI, which means we want to find the task context variable that makes high trajectory rewards more likely.

This is actually an important design choice.

POMDP perspective

- How do we optimize the policy?

- How do we parameterize the variational family?

- Can we choose other “likelihood” function?
POMDP perspective

- How do we optimize the policy?
  - Using soft actor critic (SAC)

- How do we parameterize the variational family?
  - Mean-field assumption (permutation invariance of MDP encoding)
  - Accept **variable** length of history

- Can we choose other “likelihood” function?
  - Maximize the return (as mentioned before)
  - Reconstruction of the MDP (reward and dynamics modeling)
  - Model state, or state-action value functions

- PEARL = all above

---

Outline

- Motivation for meta-reinforcement learning
- Problem setups (RL, meta learning, meta-RL)
- Common approaches
  - Black-box adaptation (based on recurrent policies)
  - Optimization-based methods
  - Inference-based methods (solving equivalent POMDP)
- Comparison
- Task design (unsupervised meta-RL)
- Summary and conclusions
Model-free meta-rl perspectives summary

Recipes for three model free perspectives:

1. memory based:
   \[ f_\theta(M_i) := RNN(\tau_{1:t}) \]
   \[ := \text{TemporalConvAttentive}(\tau_{1:t}) \]

2. bi level optimization:
   \[ f_\theta(M_i) := \theta + \alpha \nabla \theta J_i(\theta) \]

3. POMDP and posterior inference:
   task context aware policy \( \pi_\theta(a|s, z) \)
   posterior \( p(z_t|\tau_{1:t}) \)

- Relationships:
  - 3 is the **stochastic** version of 1 where the task context variable \( z \) is the adaptation parameter.
  - 2 is the same as 1 and 3 conceptually except that it chooses a specific form of the meta learner than a black-box function approximator.
Model-free meta-rl perspectives summary

Recipes for three model free perspectives:

1. **memory based**:
   \[ f_\theta(M_i) := RNN(\tau_{1:t}) \]
   \[ := \text{TemporalConvAttentive}(\tau_{1:t}) \]
   1. **memory based**:
   simple to understand and implement
   vulnerable to meta overfitting
   optimization challenges

2. **bi level optimization**:
   \[ f_\theta(M_i) := \theta + \alpha \nabla_\theta J_i(\theta) \]
   2. **bi level optimization**:
   consistency
   poor sample efficiency

3. **POMDP and posterior inference**:
   task context aware policy \[ \pi_\theta(a|s, z) \]
   posterior \[ p(z_t|\tau_{1:t}) \]
   3. **POMDP and posterior inference**:
   effective exploration
   special perspective
   same problems as memory based approaches
Outline

- Motivation for meta-reinforcement learning
- Problem setups (RL, meta learning, meta-RL)
- Common approaches
  - Black-box adaptation (based on recurrent policies)
  - Optimization-based methods
  - Inference-based methods (solving equivalent POMDP)
- Comparison
- Task design (unsupervised meta-RL)
- Summary and conclusions
How to design the meta training tasks?

- All the methods we talked about so far **take as granted a distribution of tasks (MDPs)**.

- In lots of scenarios, **the performance of the meta testing heavily depends on this distribution**:
  - Are the tasks structurally related?
  - Is the testing task in-distribution or similar to tasks from the meta training distribution?
  - Are the tasks rich enough to provide powerful prior?
  - How to systematically design such tasks for different problem?
  - ...

- Successful applications of these methods often are coupled with **hand-crafted tasks**.

- Can we **automate** task designing while maintaining the power of meta RL?
Unsupervised Meta Reinforcement Learning

- Designing general task proposal algorithm can be infeasible.
- We restrict our attention to the setting where all tasks only differ in the reward function.
  - In this case, the dynamics of the environment serves as the supervision for our task proposal algorithm
Unsupervised Meta Reinforcement Learning

- In essence, the **optimal unsupervised meta RL learner** for a Controlled Markov process (MDP without reward functions) is the procedure producing the **policy which achieves the minimal worst case regret**. (Appendix for rigorous definition)
  - Worst case over all possible reward distribution (task distribution).
  - Minimal regret on expectation over the worst-case reward function distribution.

- Use a **latent variable** to control the reward function.

- Therefore, the most important design decision is the mapping from the latent variable to the reward function.
Unsupervised Meta Reinforcement Learning

- A practical implementation of the unsupervised reinforcement learning algorithm

**Given a CMP**

1. **obtain the reward proposal procedure**

2. **sample latent task variable z~p(z)**

3. **define task reward r_z using the reward proposal procedure and z**

4. **use standard meta learning algorithm with r_z**

- Reward proposal procedure can be defined in many ways
  - Randomly initialized
  - Or **optimized with some objective**

- Latent task variable can be simple distribution

- The proposal procedure takes in the value of the latent variable and produces a family of reward functions

Gupta, Eysenbach et al. Unsupervised Meta-Learning for Reinforcement Learning [2019]
Unsupervised Meta Reinforcement Learning

**DIAYN**: a method for learning useful skills without a reward function

**DIAYN** does not depend on the rewards and only uses dynamics as supervision.

Given $D_\theta(z|s)$, define reward function $r_z(s,a) := \log(D_\theta(z|s))$

Use DIAYN to optimize the mutual information by training a discriminator $D_\theta(z|s)$ which predicts which latent variable was used to generate the rollout according to the policy $\pi(a|s,z)$.


Eysenbach et al. *Diversity is All You Need: Learning Skills without a Reward Function* [2018]
Unsupervised Meta Reinforcement Learning

Gupta, Eysenbach et al. Unsupervised Meta-Learning for Reinforcement Learning [2019]
Summary

● What we’ve covered:
  ○ Model free meta reinforcement learning
    ■ Black-box adaptation
    ■ Optimization based methods
    ■ Inference on POMDP
  ○ Unsupervised Task designs (kinda of)

● What we haven’t covered:
  ○ Model based meta reinforcement learning
  ○ Hybrid methods
  ○ Enhanced exploration
  ○ Optimization beyond gradient descent (evolution strategies)
  ○ Heterogeneous architectures to handle different state and action spaces
  ○ …
References

- **General**
  - Finn, Sergey. *ICML 2019 tutorial on meta learning*

- **Black-box adaptation**
  - Wang et al. (2016) *Learning to Reinforcement Learn*
  - Duan et al. (2016) *RL^2: Fast Reinforcement Learning via Slow Reinforcement Learning*
  - Mishra, Rohaninejad et al. (2018) *A Simple Neural Attentive Meta-Learner*

- **Optimization-based methods**

- **POMDP perspective**
  - Rakelly, Zhou et al. (2019) *Efficient Off-Policy Meta-Reinforcement Learning via Probabilistic Context Variables*

- **Unsupervised meta learning**
Thank you!

- Questions are welcome
Appendix: Unsupervised Meta Reinforcement Learning

**Controlled Markov process (CMP)**
\[ C = \{S, A, P\} \]

**A learning procedure (meta learner)**
\[ f : \mathcal{D}(\mathcal{M}_i) \rightarrow \pi \]

**Evaluation of the meta learner for a specific reward function**
\[ R(f, r_z) = \sum_i \mathbb{E}_{\pi=f(\tau_1, \ldots, \tau_{i-1})} \left[ \sum_{t} r_z(s_t, a_t) \right] \]

**Task distribution**
- distribution over latent variable \( z \)
- distribution over reward functions
Appendix: Unsupervised Meta Reinforcement Learning

The optimal learning procedure under a specific reward function distribution
\[ f^* := \arg \max_f \mathbb{E}_{p(r_z)}[R(f, r_z)] \]

Regret of a learning procedure under a specific reward function distribution
\[ \text{REGRET}(f, p(r_z)) := \mathbb{E}_{p(r_z)}\left[ R(f^*, r_z) - R(f, r_z) \right] \]

Regret of a learning procedure for a CMP
\[ \text{REGRET}_{WC}(f, C) := \max_{p(r_z)} \text{REGRET}(f, p(r_z)) \]

Optimal unsupervised learning procedure
\[ f^*_C := \arg \min_f \text{REGRET}_{WC}(f, C) \]