TÉCNICO LISBOA

Planning, Learning and Intelligent Decision Making

Lecture 8

PADInt 2024

What we've done so far...

- Formulate sequential decision problems as (PO)MDPs
- Use our knowledge of transitions and costs to compute optimal policy:
  - Which actions to take in each situation (state)

Planning

What is learning?

What is ML?

- "Field of study that gives computers the ability to learn without being explicitly programmed"

Arthur Samuel, 1901-1990

Creator of world's first self-learning program

What is ML?

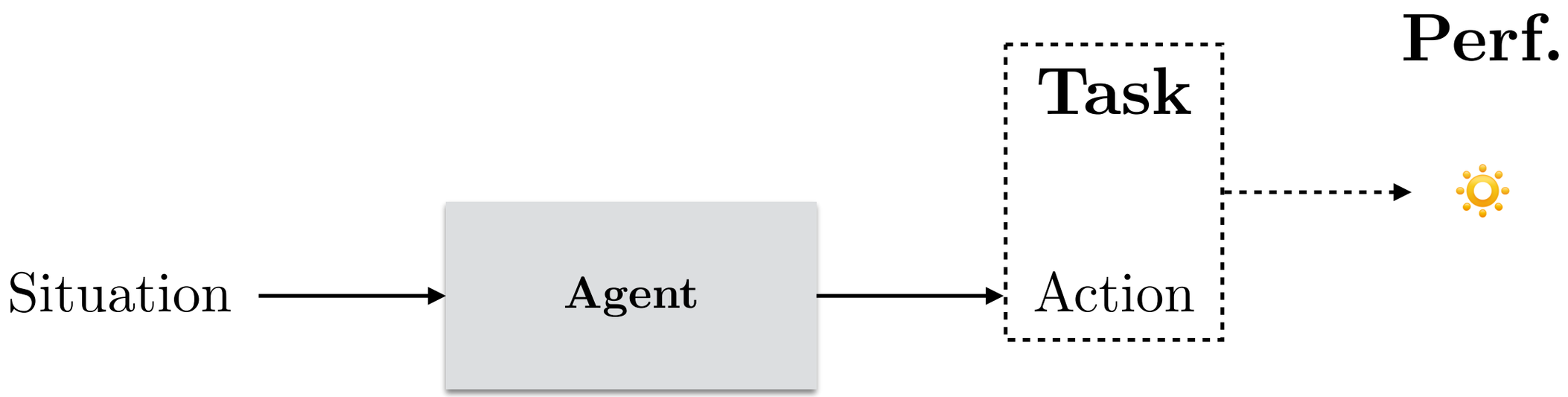
- "A computer program is said to learn from experience **E** with respect to some class of tasks **T** and performance measure **P** if its performance at tasks in **T**, as measured by **P**, improves with experience **E**."

Tom Mitchell, 1951-today

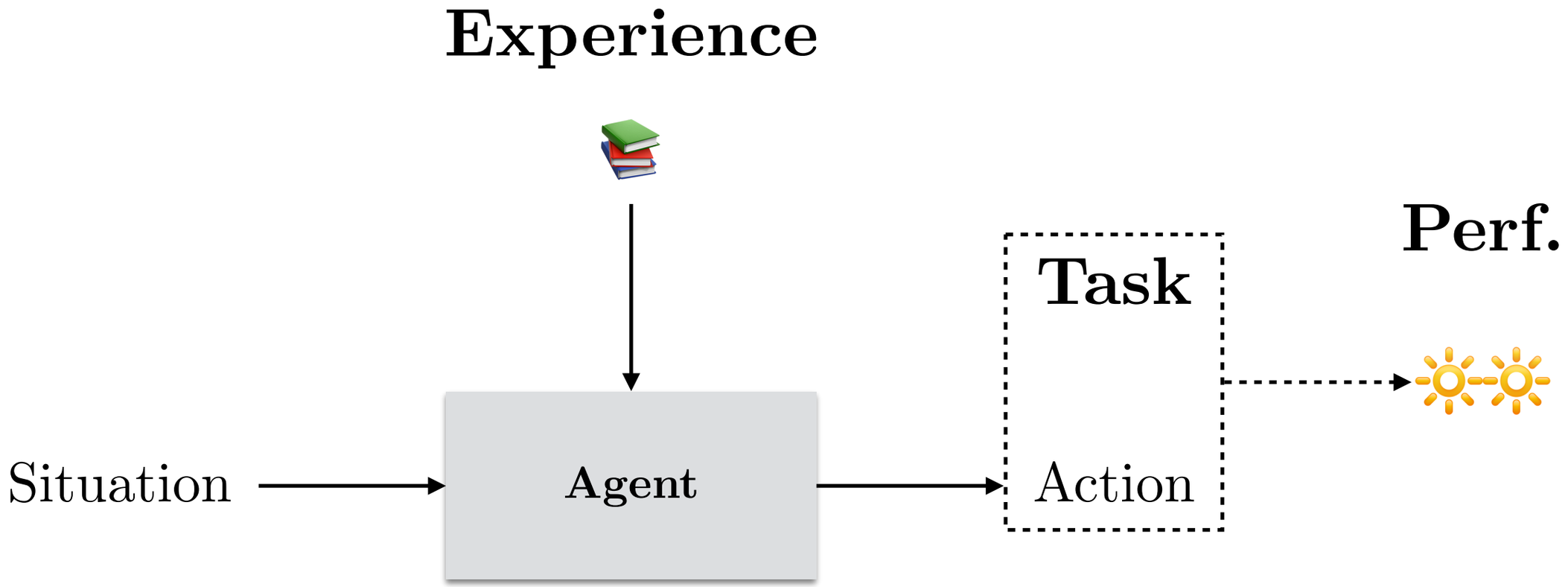
Author of book "Machine Learning"

Can computers improve their performance in a given task with experience/data?

What is ML?



What is ML?



Learning problem

- Specifying a learning problem involves:
  - Identifying the **task**, **T**
  - Defining the **performance measure**, **P**
  - Identifying/selecting the **training experience**, **E**

Types of learning

Types of learning I

- **Learning from examples**
  - **Task:** Learning a situation-action correspondence (decision rule)
  - **Performance:** Correctness of actions in all possible situations
  - **Training experience:** Set of situation-action pairs

Supervised learning

Types of learning II

- **Learning by trial-and-error**
  - **Task:** Learning a situation-action correspondence (decision rule)
  - **Performance:** Accumulated cost of actions in all possible situations
  - **Training experience:** Set of situation-action-cost triplets

Reinforcement learning

Types of learning III

- **Find structure in data**
  - **Task:** Find hidden structure in data
  - **Performance:** Quality of structure found
  - **Training experience:** Data to be analyzed

- We will not be doing this

Unsupervised learning

Learning from examples

Learning from examples

- We provide the agent with **examples** of situations-actions
- We would like the agent to learn the **underlying task**
  - Perform similarly in known situations
  - Generalize to situations never encountered before

How to learn?

Example

- Compute an action (a or b) given the state x
- State is described by 3 binary features, $\phi_1(x)$, $\phi_2(x)$, $\phi_3(x)$
- Examples:

$\phi_1(x)$	$\phi_2(x)$	$\phi_3(x)$	action
0	1	0	a
0	0	1	a
0	1	1	b
1	0	0	b

What is the function?

Two solutions

$\phi_1(x)$	$\phi_2(x)$	$\phi_3(x)$	action
0	0	0	a
0	1	0	a
0	0	1	a
0	1	1	b
1	0	0	b
1	0	1	a
1	1	0	b
1	1	1	b

$\phi_1(x)$	$\phi_2(x)$	$\phi_3(x)$	action
0	0	0	b
0	1	0	a
0	0	1	a
0	1	1	b
1	0	0	b
1	0	1	b
1	1	0	a
1	1	1	a

What is the problem?

No free lunches...

- Beyond the given examples, everything is possible!

No free lunch theorem in machine learning

For every task in which your decision rule does great, there is another task in which it does terrible...

So is there no hope?

No free lunches?

- Some hypothesis are **much more likely** than others
- We build our algorithms around **assumptions** on the problem
  - Assumptions translate knowledge about the problem (which hypotheses are most likely)
  - Assumptions greatly reduce search space (this reduction is known as **inductive bias**)

Example

- Compute an action (a or b) that depends only on **conjunctions** and **disjunctions** of $\phi_1(x)$, $\phi_2(x)$ and $\phi_3(x)$

$$\phi_1(x) \vee \phi_2(x) \vee \phi_3(x)$$

$$(\phi_1(x) \wedge \phi_2(x)) \vee \phi_3(x)$$

$$\phi_1(x) \wedge (\phi_2(x) \vee \phi_3(x))$$

$$\phi_1(x) \vee (\phi_2(x) \wedge \phi_3(x))$$

$$(\phi_1(x) \vee \phi_2(x)) \wedge \phi_3(x)$$

$$\phi_1(x) \wedge \phi_2(x) \wedge \phi_3(x)$$

$\phi_1(x)$	$\phi_2(x)$	$\phi_3(x)$	action
0	1	0	a
0	0	1	a
0	1	1	b
1	0	0	b

What is the function?

Example

- Compute an action (a or b) that depends only on **conjunctions** and **disjunctions** of $\phi_1(x)$, $\phi_2(x)$ and $\phi_3(x)$:

$$\phi_1(x) \vee \phi_2(x) \vee \phi_3(x)$$

$$(\phi_1(x) \wedge \phi_2(x)) \vee \phi_3(x)$$

$$\phi_1(x) \wedge (\phi_2(x) \vee \phi_3(x))$$

$$\phi_1(x) \vee (\phi_2(x) \wedge \phi_3(x))$$

$$(\phi_1(x) \vee \phi_2(x)) \wedge \phi_3(x)$$

$$\phi_1(x) \wedge \phi_2(x) \wedge \phi_3(x)$$

$\phi_1(x)$	$\phi_2(x)$	$\phi_3(x)$	action
0	1	0	a
0	0	1	a
0	1	1	b
1	0	0	b

$\phi_1(x) \vee (\phi_2(x) \wedge \phi_3(x))$

Example

- We can now generalize to new situations!
- What is the action for (1, 1, 0)?
  - $1 \vee (1 \wedge 0) = 1 \qquad (b)$

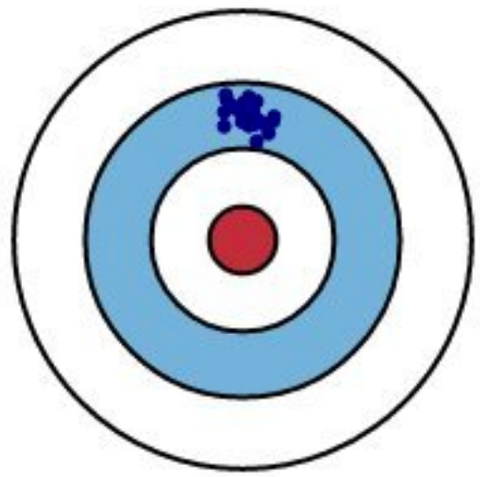
$\phi_1(x)$	$\phi_2(x)$	$\phi_3(x)$	action
0	1	0	a
0	0	1	a
0	1	1	b
1	0	0	b

Inductive bias

- Assumptions about the input-output relation allow us to **generalize**
  - Limit set of hypotheses under consideration
  - Reduce the set of hypotheses that match the observed data
  - Able to generalize the input-output relation in the observed data to unseen inputs

Inductive bias (cont.)

- **Wrong assumptions**
  - The hypotheses considered poorly represent the desired input-output correspondence
    - Learned hypothesis will not properly represent the data



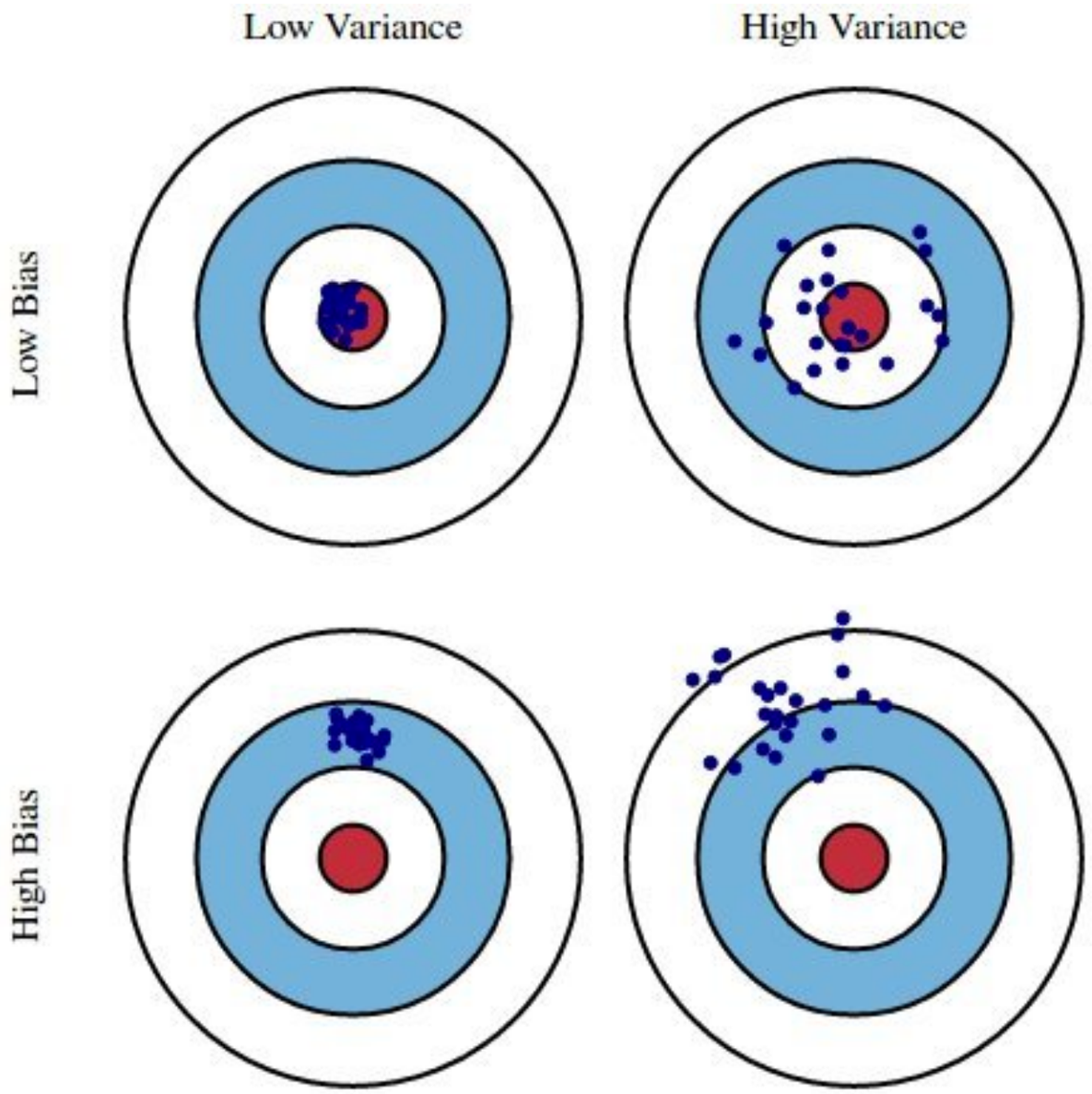
Tackling bias

- Make as few assumptions as possible?
  - Set of possible hypotheses "sufficiently large"
  - Make sure that the desired input-output correspondence is considered

Bias-variance trade-off

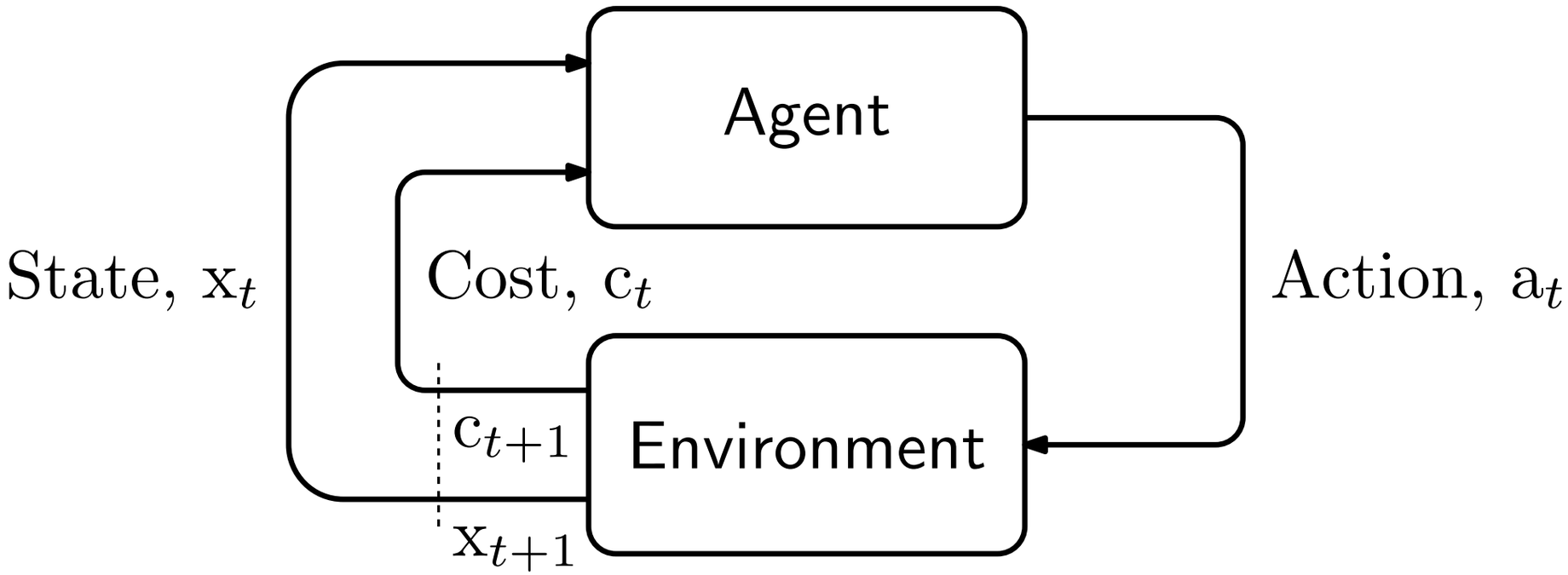
- Sources of error in learning algorithm:
  - **Bias:** error introduced by assumptions about the target hypothesis
  - **Variance:** error introduced due to the variability of the training set
- Assumptions must compromise between the two

Bias-variance trade-off



Learning from examples in MDPs

Markov decision process



Computing Q^*

- We have that

$$Q^*(x,a) = c(x,a) + \gamma \sum_{x' \in \mathcal{X}} \mathbf{P}_a(x' \mid x) \min_{a'} Q^*(x',a')$$

- We can then compute

$$\pi^*(x) = \arg\min_a Q^*(x,a)$$

Computing Q^*

- We have that

$$Q^*(x,a) = c(x,a) + \gamma \sum_{x' \in \mathcal{X}} \mathbf{P}_a(x' \mid x) \min_{a'} Q^*(x',a')$$

We must know the cost

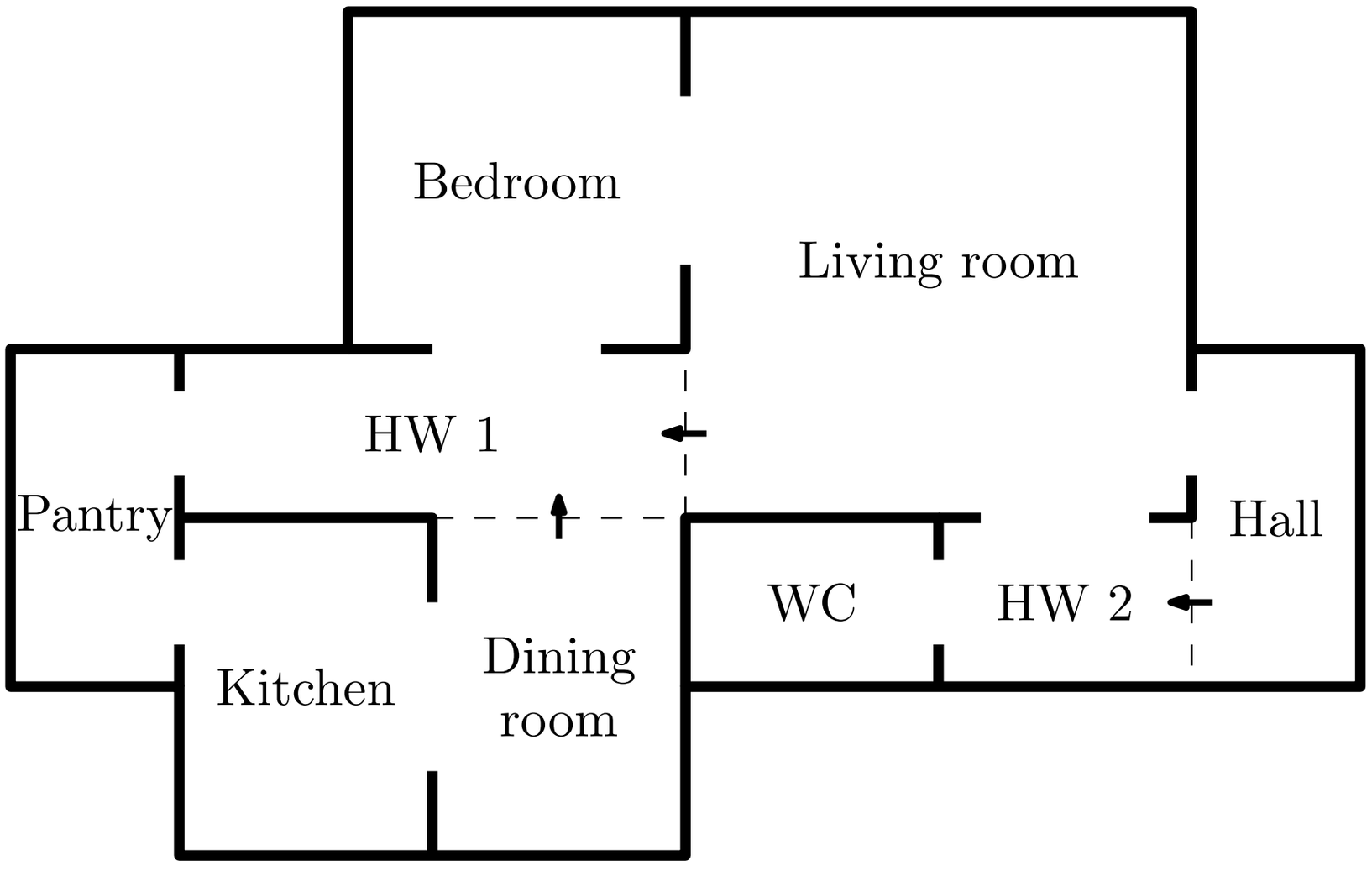
We must know the transition probabilities

What if we don’t?

The household robot

Household robot

- Consider the household

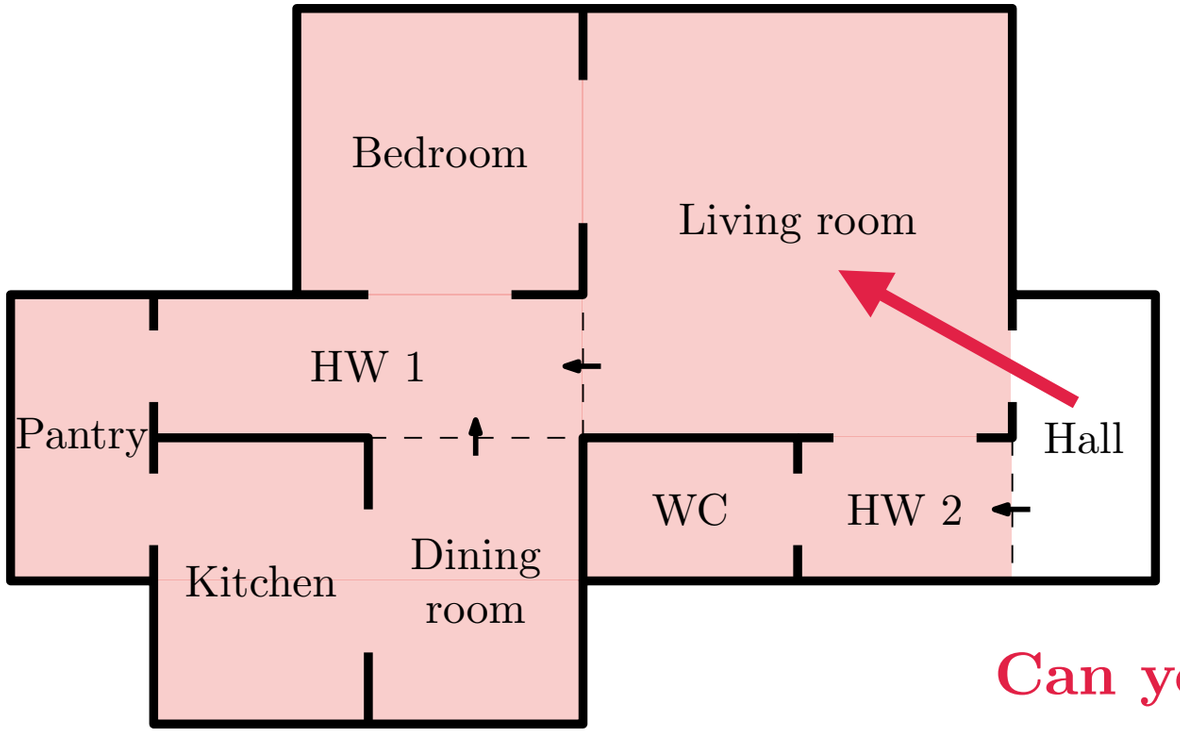


Household robot

- Robot moves in the environment, assisting human users
- Human wants to program the robot to perform the task of moving to a specific room
- Human does not know how to build "cost functions"
- However, human can demonstrate how to move

Household robot

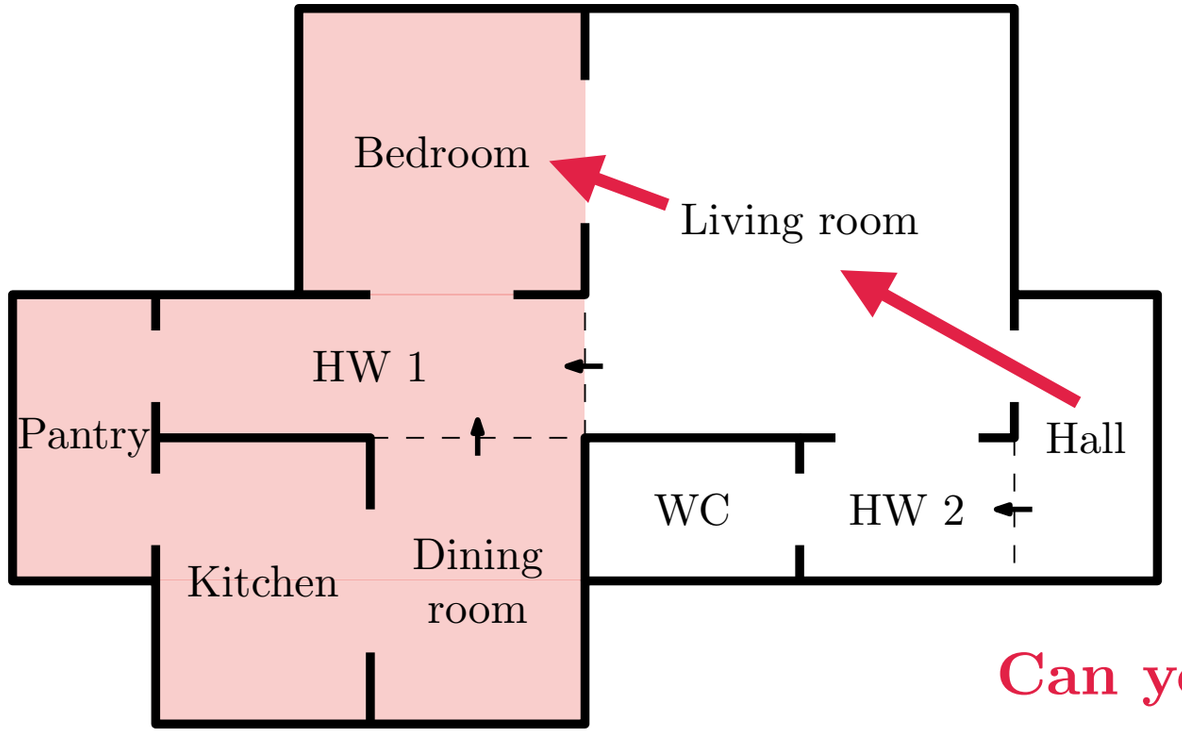
- Human indications:
  - In the Hall, move to the living room.



Can you guess the task?

Household robot

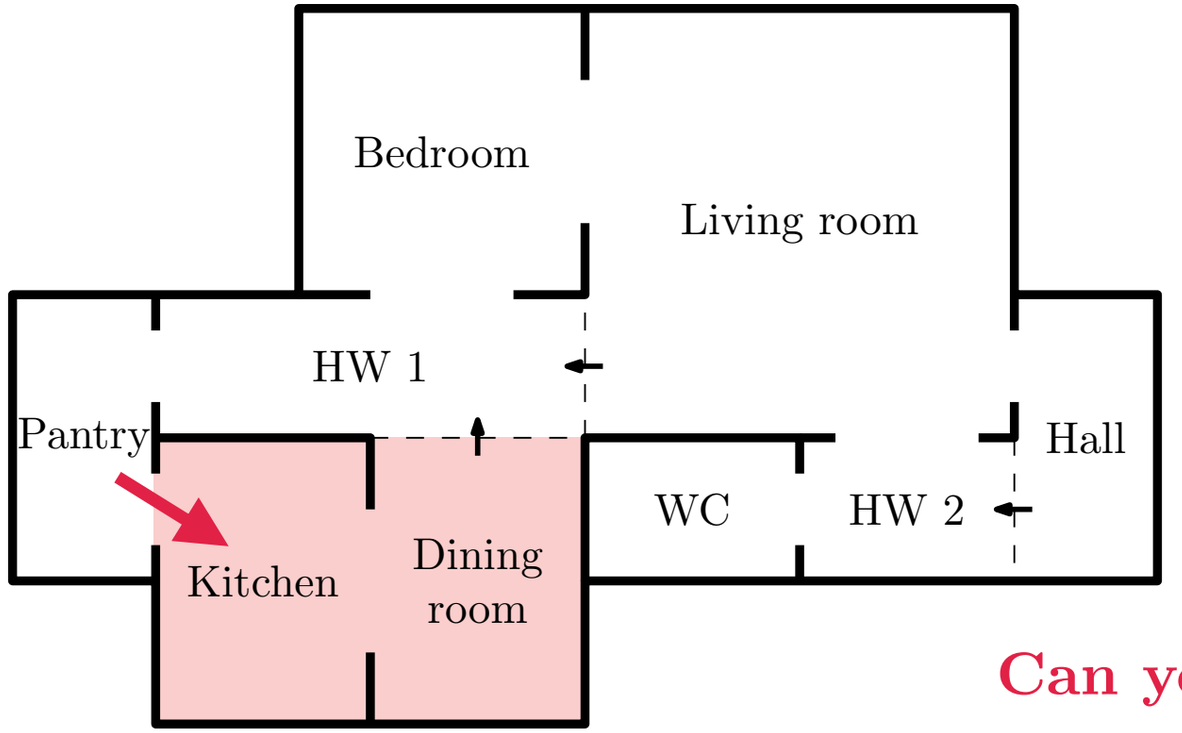
- Human indications:
  - In the Hall, move to the Living room.
  - In the Living room, move to the Bedroom



Can you guess the task?

Household robot

- Human indications:
  - In the Pantry, move to the kitchen



Can you guess the task?

Learning from examples

- Robot is learning from examples by the human user
- Each example consists of a pair $(x,\ a)$:
  - In the Hall, move to the living room

    $(x = \text{Hall},\ a = \text{Move to Living room})$
  - In the Pantry, move to the kitchen

    $(x = \text{Pantry},\ a = \text{Move to Kitchen})$

Learning from examples

- We consider a simple setting:
  - The agent wants to learn the optimal policy for an MDP
  - We know the states of the world, $\mathcal{X}$
  - We know the actions available, $\mathcal{A}$
  - We know the dynamics of the world, $\{\mathbf{P}_a\}$
  - **We don't know the task (i.e., the cost c)**

Some nomenclature

TASK:

- We want to learn a policy:
  - Deterministic $\pi : \mathcal{X} \to \mathcal{A}$
  - Stochastic $\pi : \mathcal{X} \to \Delta(\mathcal{A})$

Some nomenclature

EXPERIENCE:

- Dataset of examples $\mathcal{D} = \{(x_0, a_0), (x_1, a_1), \ldots, (x_N, a_N)\}$

- Pairs (x, a) generated from an **unknown distribution** μ_D

$$\mu_D(x, a) = \mathbb{P}\left[\mathrm{x} = x, \mathrm{a} = a\right]$$

Some nomenclature

EXPERIENCE:

- Dataset of examples $\mathcal{D} = \{(x_0, a_0), (x_1, a_1), \ldots, (x_N, a_N)\}$

- Pairs (x, a) generated from an **unknown distribution** μ_D

$$\mu_D(x, a) = \mathbb{P}\left[\mathrm{a} = a \mid \mathrm{x} = x\right] \mathbb{P}\left[\mathrm{x} = x\right]$$

Some nomenclature

EXPERIENCE:

- Dataset of examples $\mathcal{D} = \{(x_0, a_0), (x_1, a_1), \ldots, (x_N, a_N)\}$
- Pairs (x, a) generated from an **unknown distribution** μ_D

$$\mu_D(x, a) = \pi^*(a \mid x)\mathbb{P}\left[\mathrm{x} = x\right]$$

Teacher's policy

Some nomenclature

PERFORMANCE:

- **Expected cost** or **risk** of policy π:

$$L(\pi) = \mathbb{E}_{\mu_D}\left[\ell(\mathrm{x}, \mathrm{a}; \pi)\right] = \sum_{x,a} \ell(x, a; \pi)\mu_D(x, a)$$

Input-output distribution

Cost of π given correspondence (x, a)

Some nomenclature

PERFORMANCE:

- **Expected cost** or **risk** of policy π:

$$L(\pi) = \mathbb{E}_{\mu_D}\left[\ell(\mathrm{x}, \mathrm{a}; \pi)\right] = \sum_{x,a} \ell(x, a; \pi)\mu_D(x, a)$$

- Goal of the agent is to compute a policy

$$\pi^* = \underset{\pi}{\operatorname{argmin}}\, L(\pi)$$

How do we compute this?

Problem

- We want to compute

$$L(\pi) = \mathbb{E}_{\mu_D}\left[\ell(\mathrm{x}, \mathrm{a}; \pi)\right] = \sum_{x,a} \ell(x, a; \pi)\mu_D(x, a)$$

Evaluate error in all situations?

We don't know this distribution...

Solution

- Evaluate the policy in the provided examples:

$$L(\pi) \approx \hat{L}_N(\pi) = \frac{1}{N} \sum_{n=1}^{N} \ell(x_n, a_n; \pi)$$

Empirical risk

Empirical risk minimization

Inductive learning

The inductive learning assumption

If we learn from a sufficiently large set of examples, we will do well in the actual task

How does this apply to MDPs?

Learning from examples

- We want to replicate the policy observed from the teacher
  - An example is a pair (state, action)
  - When the teacher provides an example (x, a), it indicates the learner that the optimal action in state x is a
- Equivalently,

$$\pi^*(x) = a$$

... then...

- We can:

  1. ... "clone" the observed behavior, using the MDP structure to generalize → **MDP-induced metric**

  or

  2. ... "invert" the MDP → **Inverse reinforcement learning**

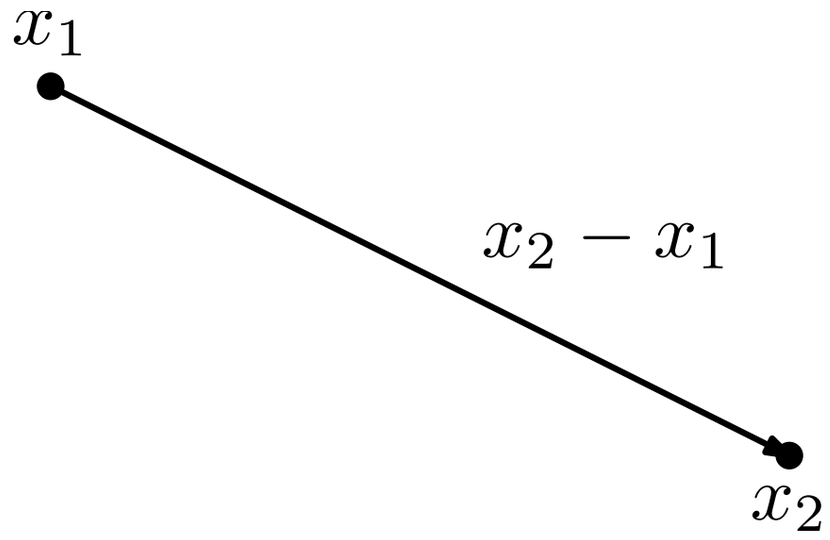
MDP-induced metric

What is a "metric"?

- A way to measure distances
- Example: Euclidean distance (ℓ_2)

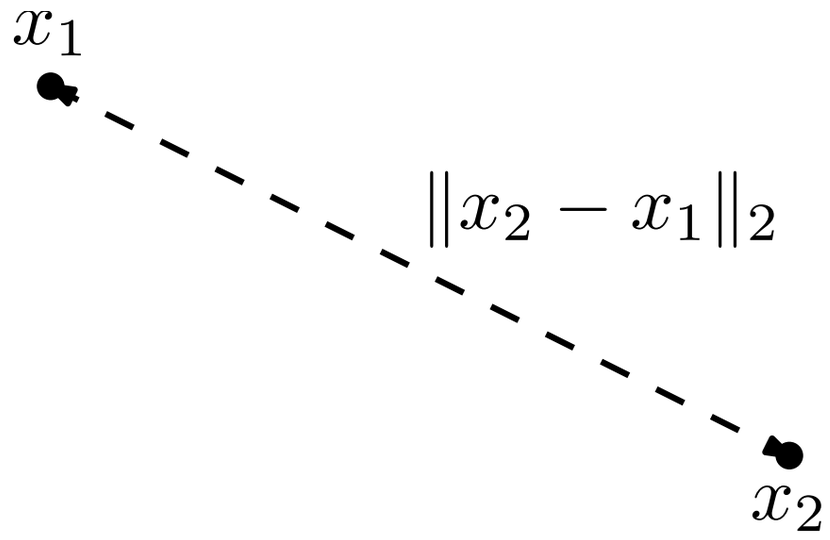
What is a "metric"?

- A way to measure distances
- Example: Euclidean distance (ℓ_2)



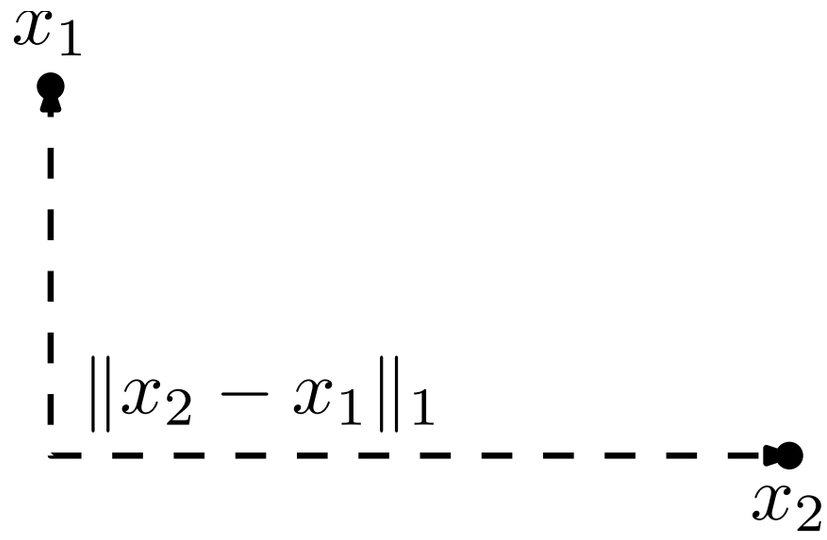
What is a "metric"?

- A way to measure distances
- Example: Euclidean distance (ℓ_2)



What is a "metric"?

- A way to measure distances
- Example: Manhattan distance (ℓ_1)



What is a "metric"?

- A way to measure **similarity**
  - Points that are close are similar
  - Points that are far are dissimilar

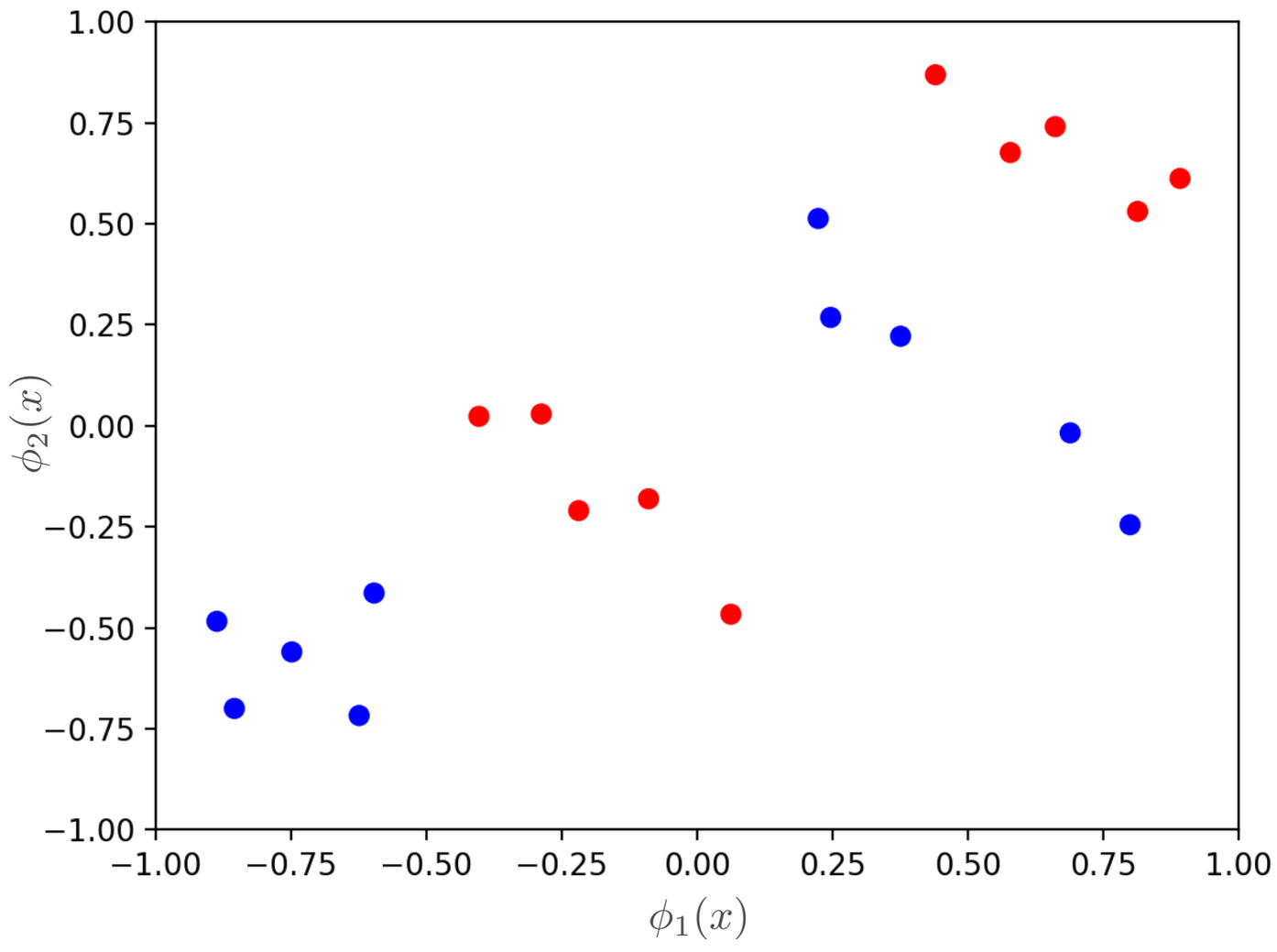
Why do we care?

- Metrics are at the core of similarity-based methods
  - E.g., k-nearest neighbors (KNN)

How does KNN work?

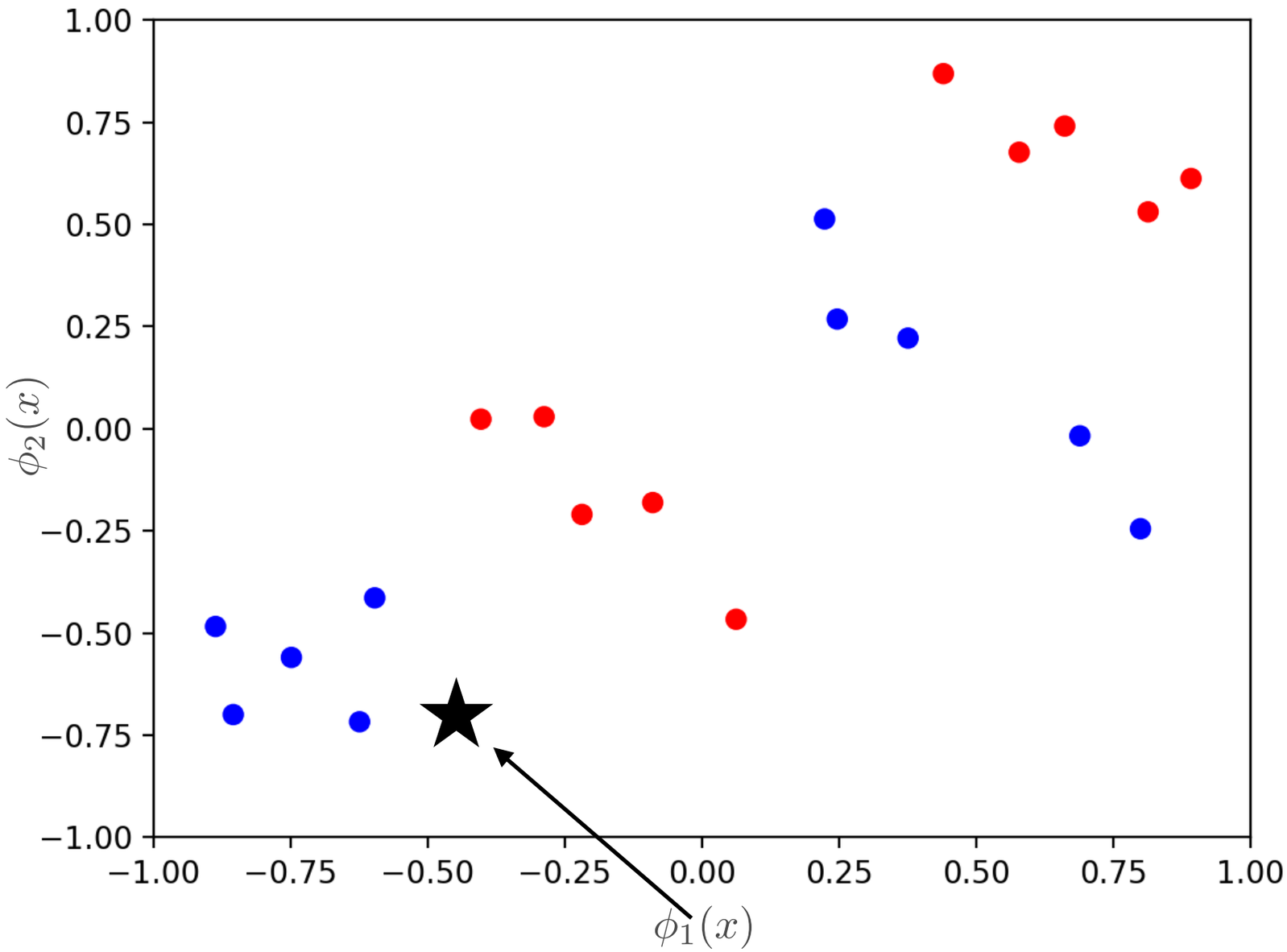
Example

- States are described by two numerical features, ϕ_1 and ϕ_2
- Two actions available, a (●) and b (●)



Example

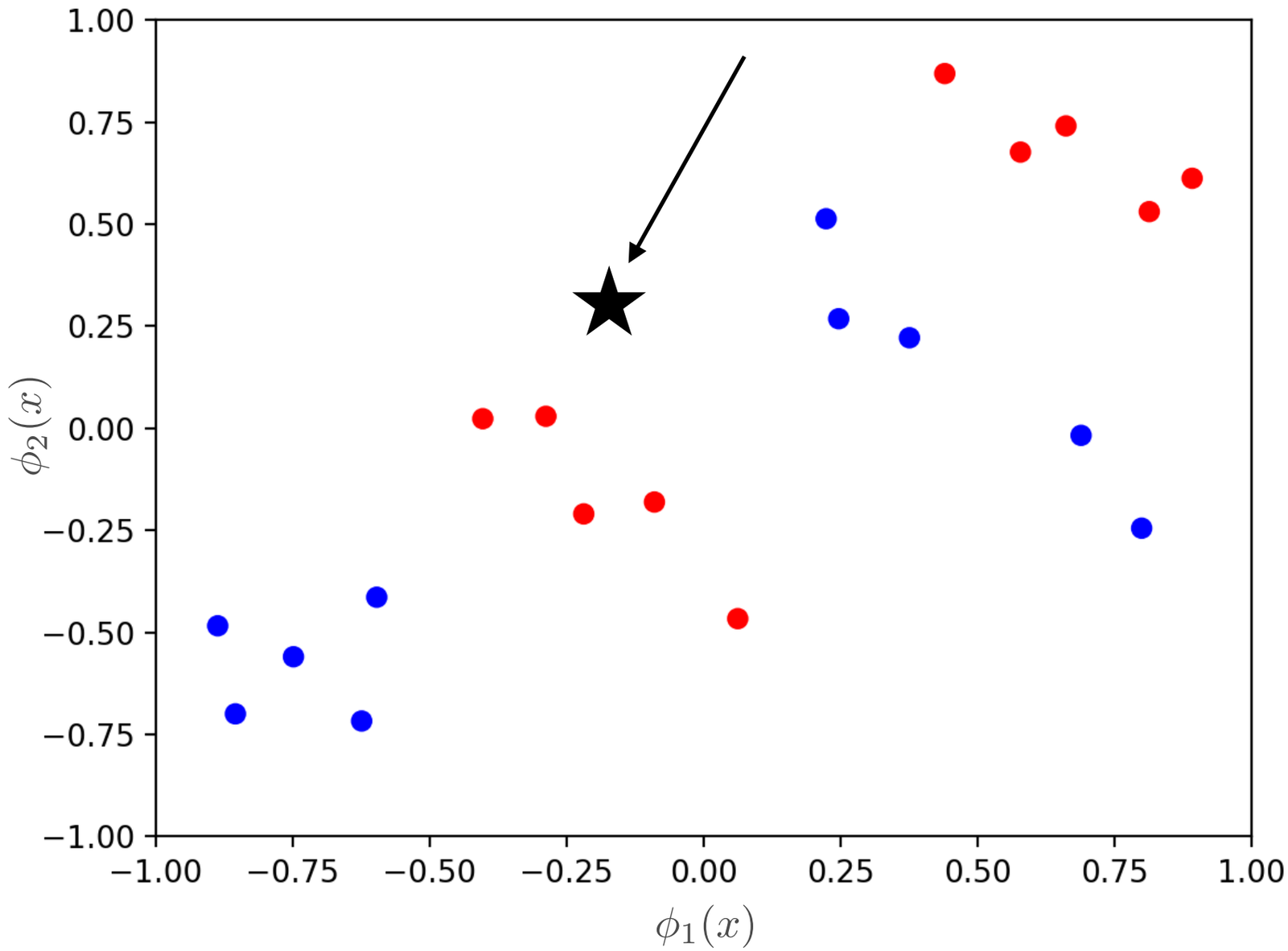
- What class do you think this point is?



Blue

Example

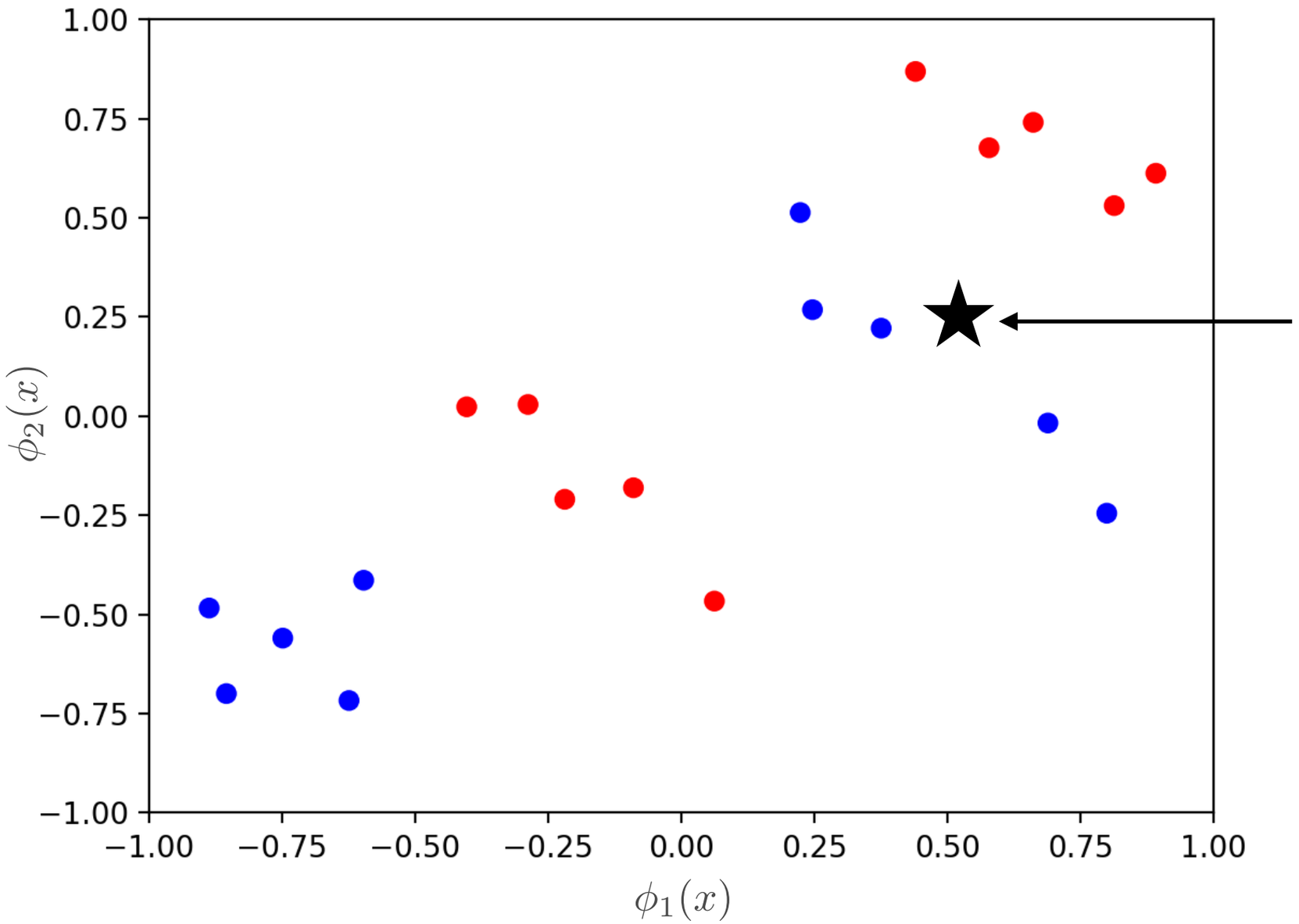
- What class do you think this point is?



Red

Example

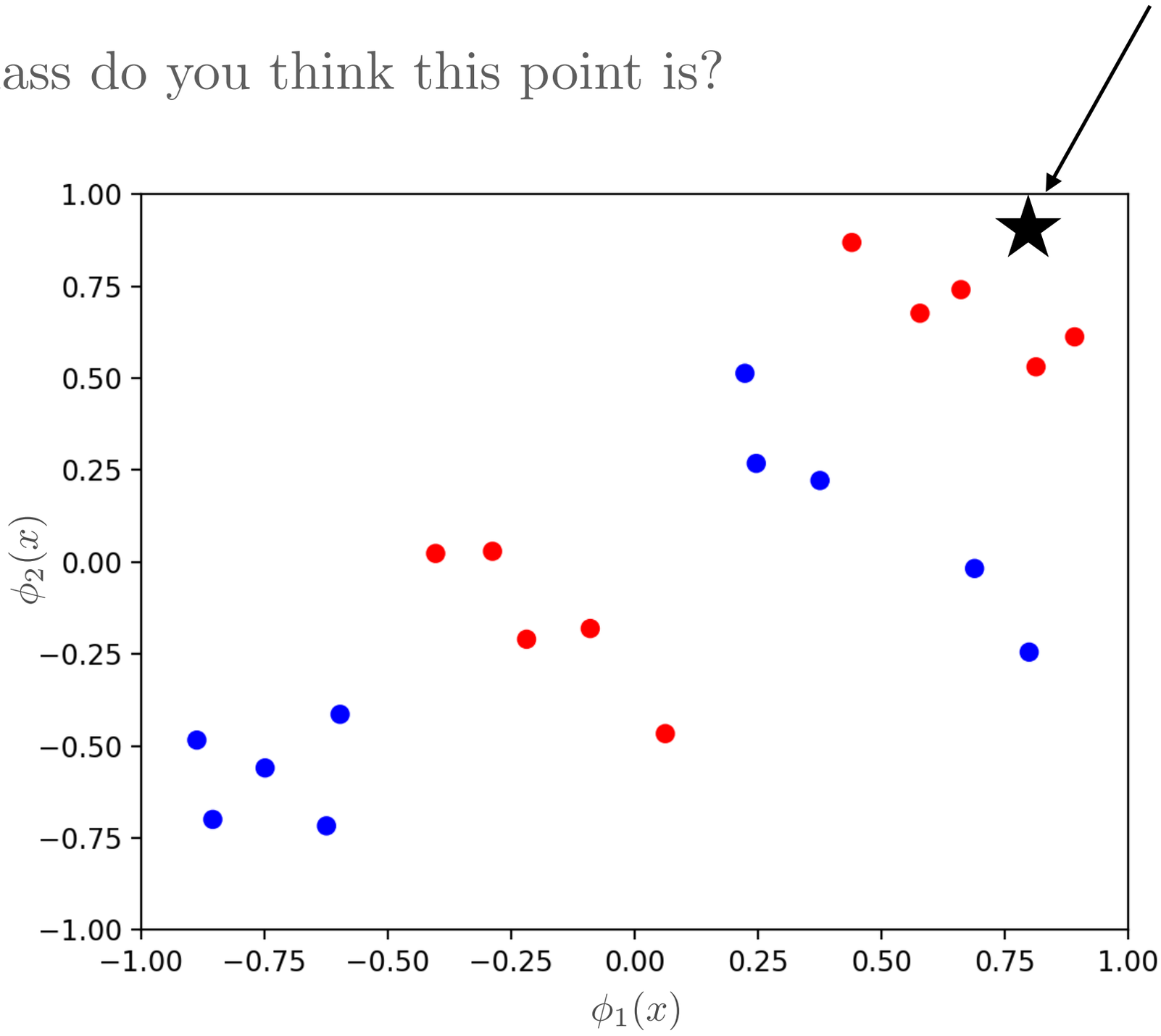
- What class do you think this point is?



Blue

Example

- What class do you think this point is?



Red

Why?

Similarity

- You naturally assume that "geometry matters":
  - Points close by are alike
  - We can extend that intuition to build a classifier

k Nearest Neighbors

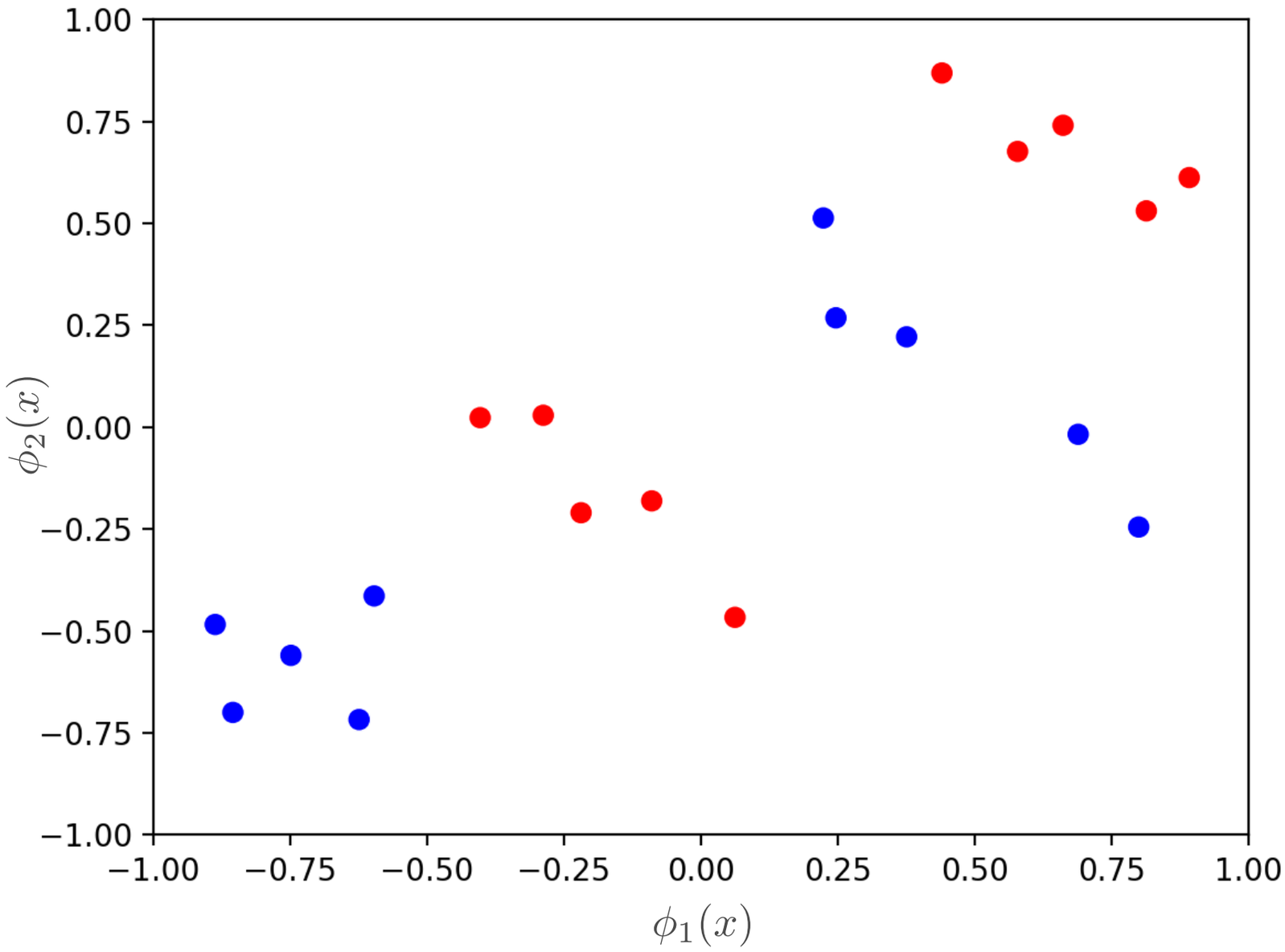
- No "explicit training"
- Dataset is kept in memory

k Nearest Neighbors

- How does it work?
  - When new point arrives, check k closest points
  - Select the class of the majority (odd k)

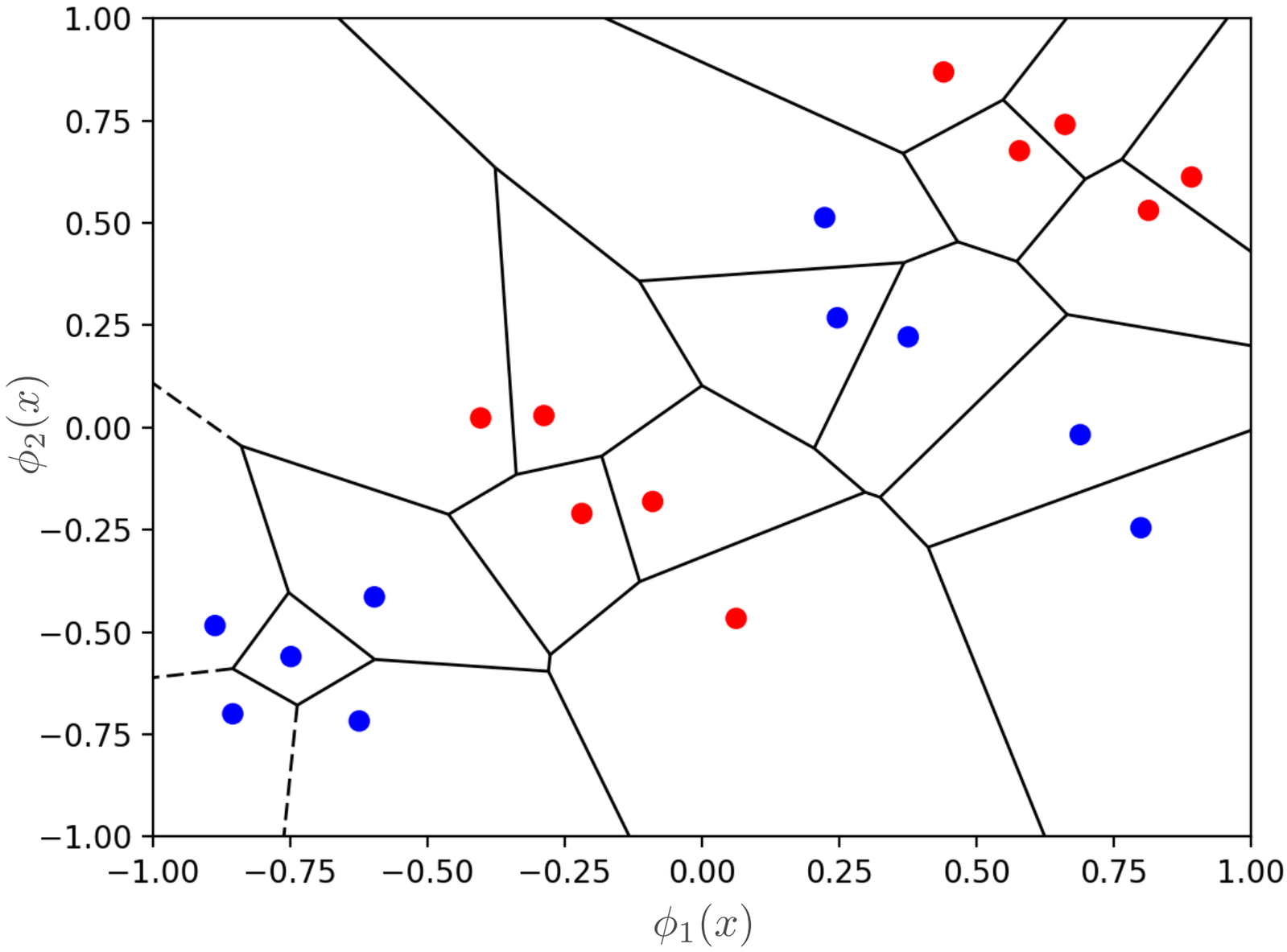
Example

- 1-NN


1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00
$\phi_2(x)$
$\phi_1(x)$

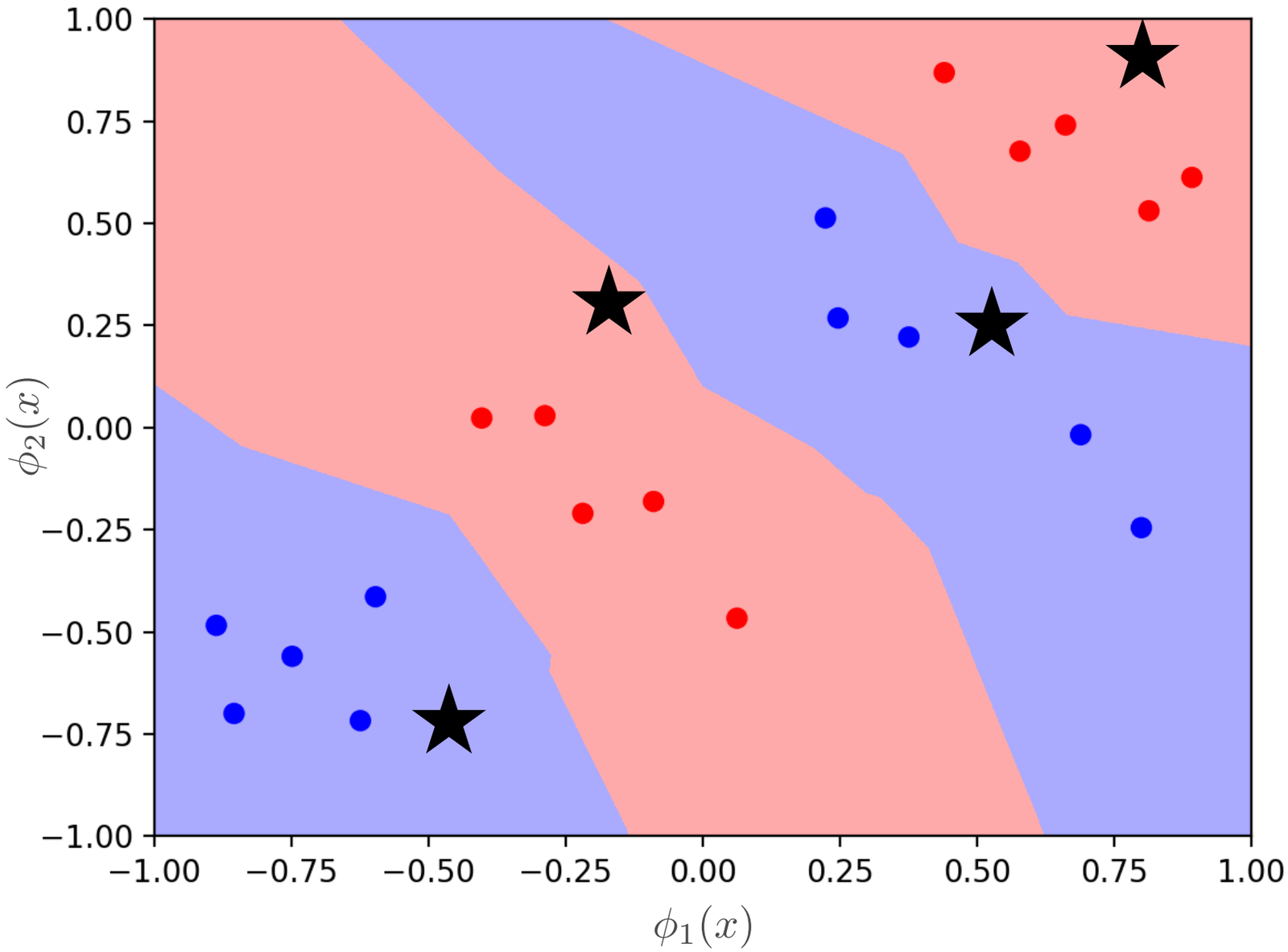
Example

- 1-NN


1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00
$\phi_2(x)$
$\phi_1(x)$

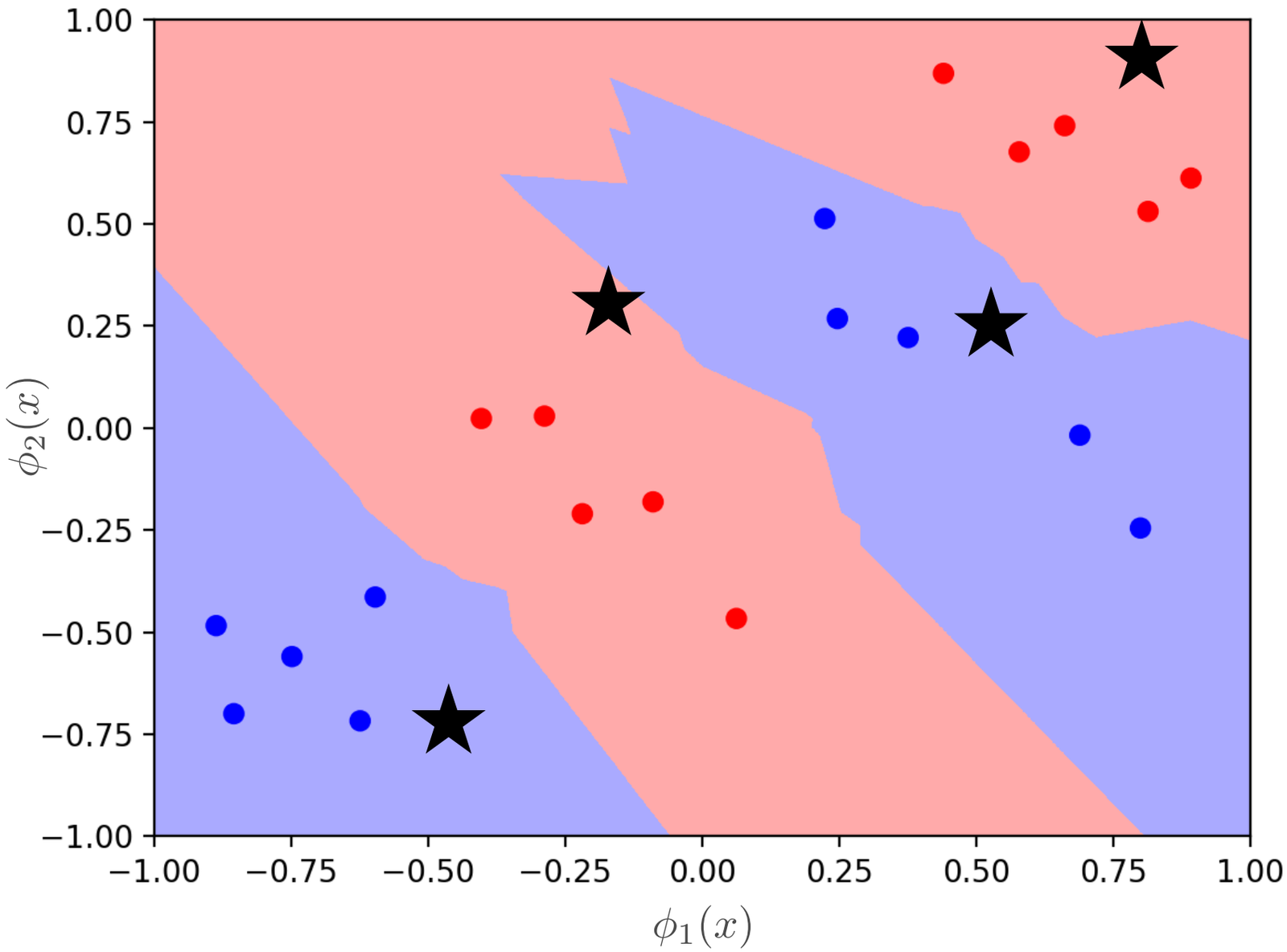
Example

- 1-NN


1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00
$\phi_2(x)$
$\phi_1(x)$

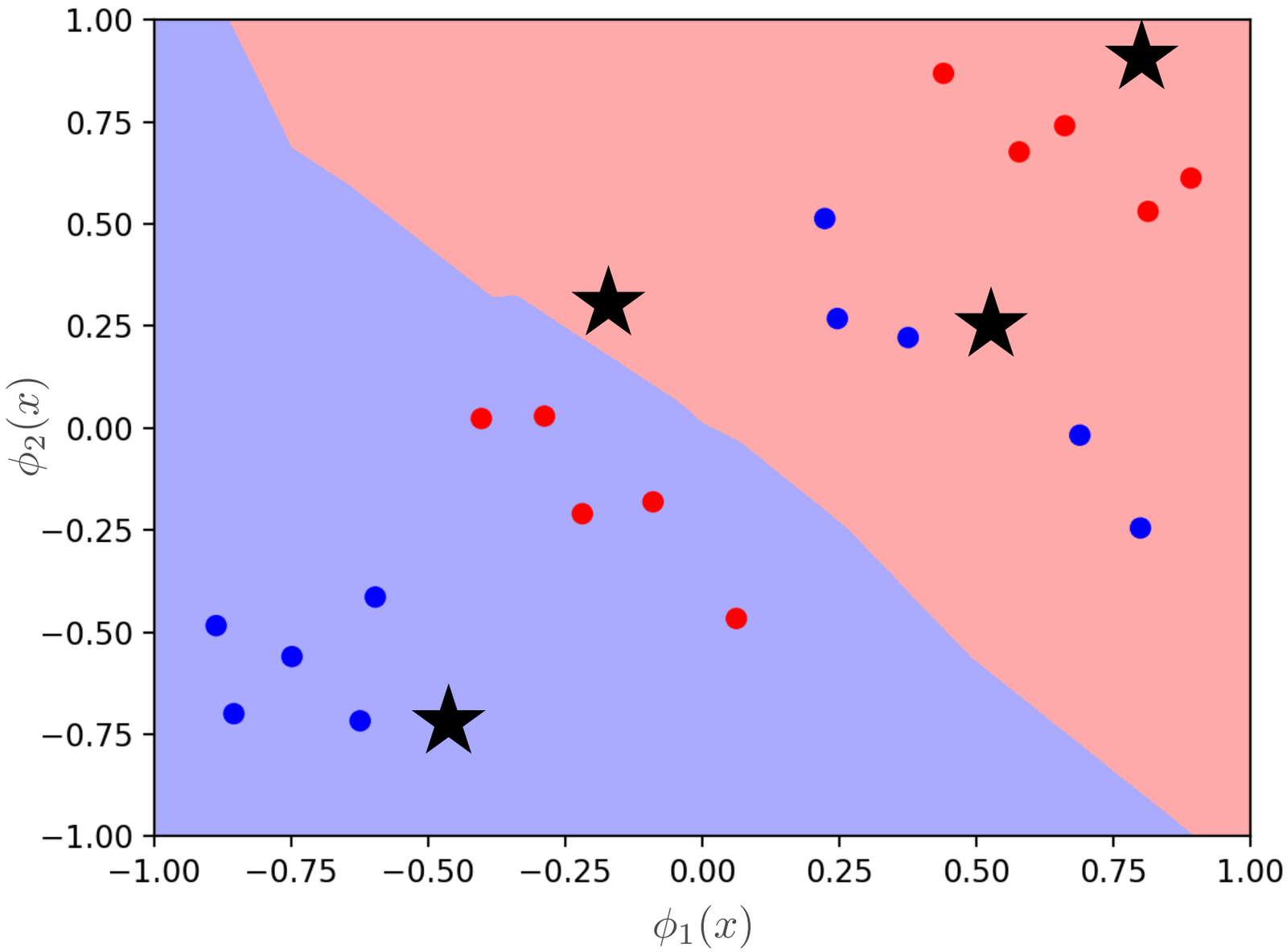
Example

- 5-NN


1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00
$\phi_2(x)$
$\phi_1(x)$

Example

- 15-NN



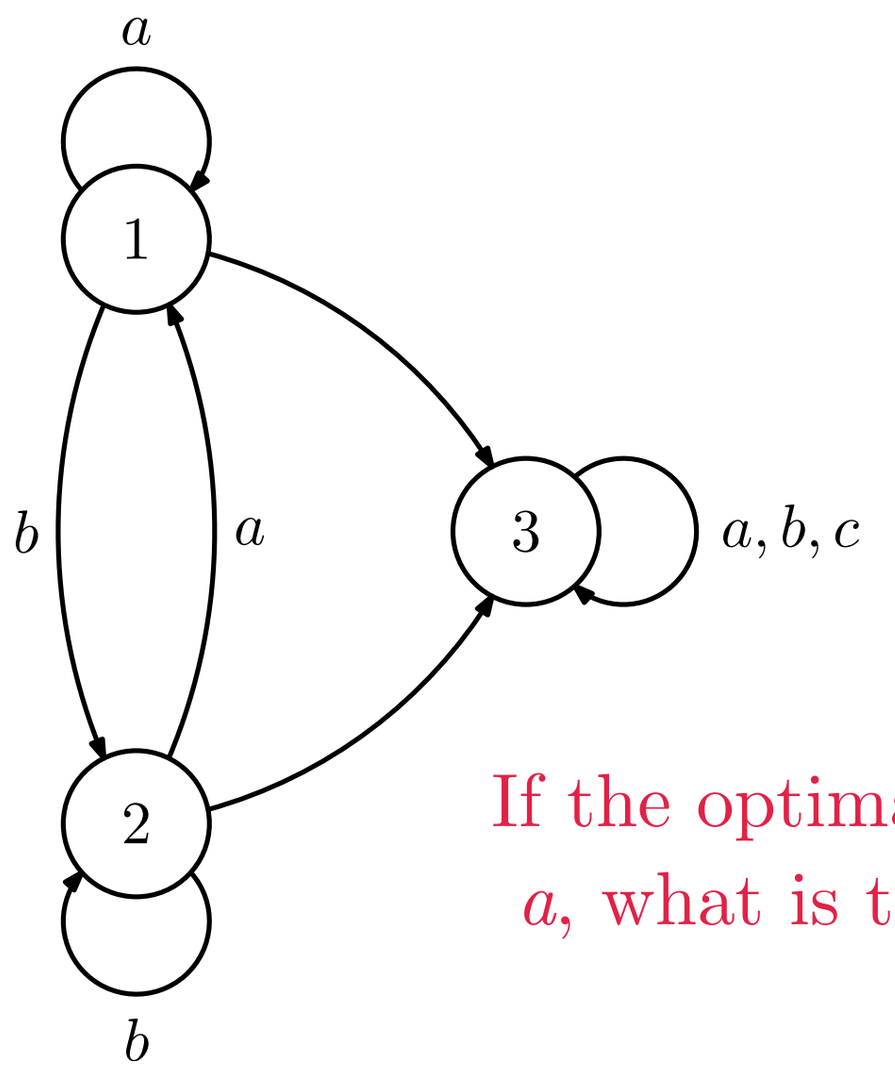
Can we define a metric between states in an MDP?

Metric for MDPs

- If we could...
  - ... examples of actions in one state can be generalized to "nearby" states
- Then, if the "teacher" follows the optimal policy...
  - ... the learner can learn the optimal policy without planning

Measuring similarity

- Consider the MDP (without costs):



If the optimal action in state 1 is a, what is the optimal action in state 2?

Measuring similarity

- If all costs are allowed, no generalization is possible

Why?

All policies are possible, so no generalization between states is possible

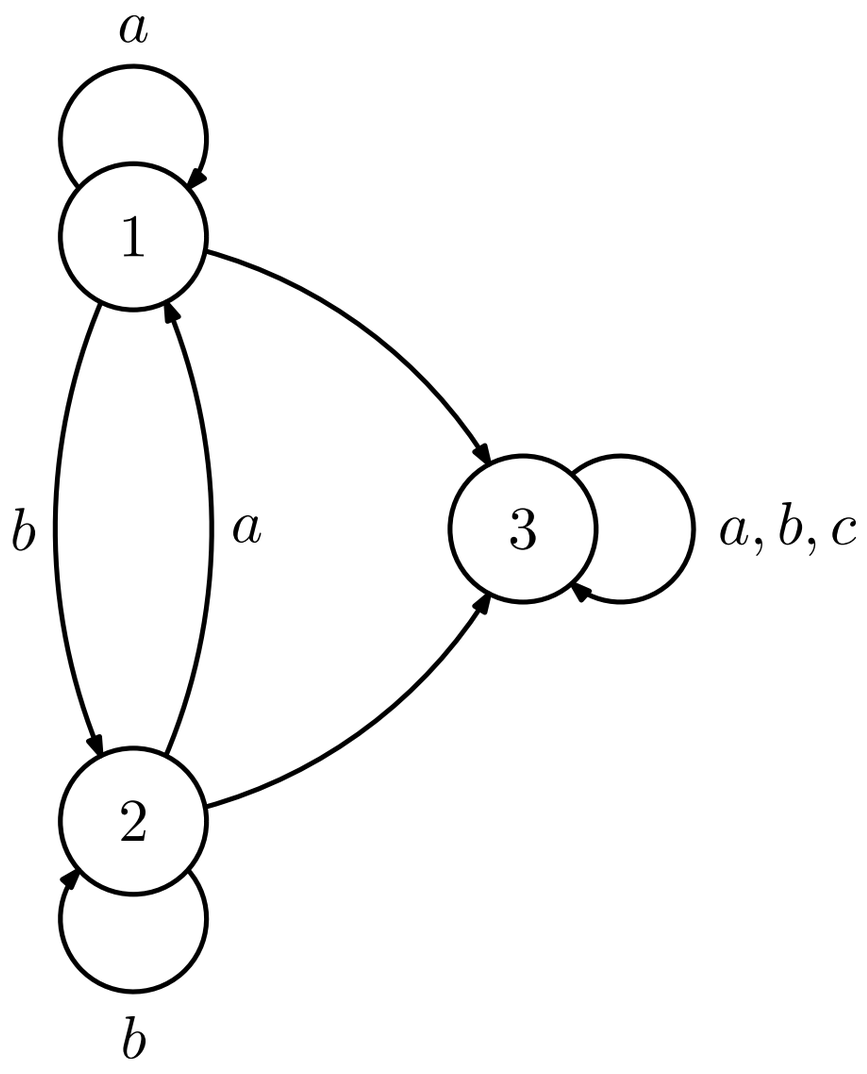
Measuring similarity

- Inductive bias assumption: **costs are the same for all actions** (only depend on state)
- If the optimal action in state 1 is a, what is the optimal action in state 2?

↓

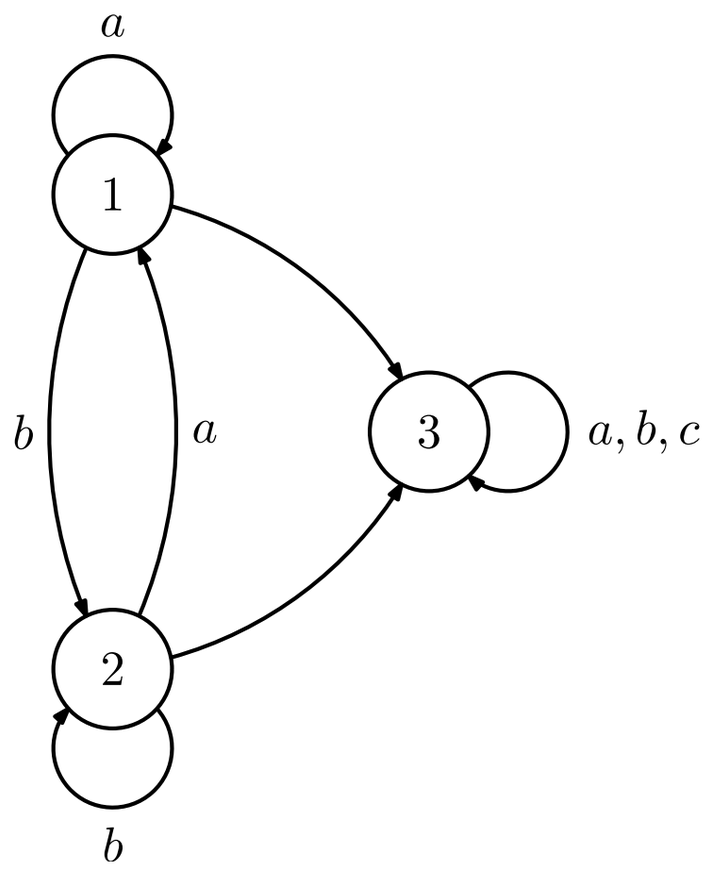
Also a

Actions are alike in states 1 and 2



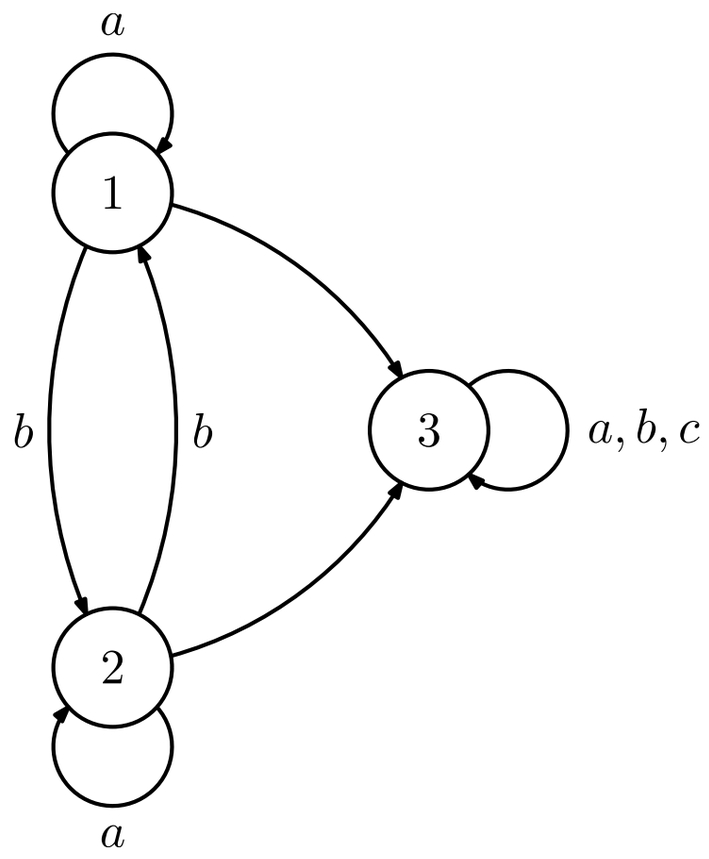
Measuring similarity

- How to compare states in MDPs?
  - Same actions have same outcomes → states are similar



Measuring similarity

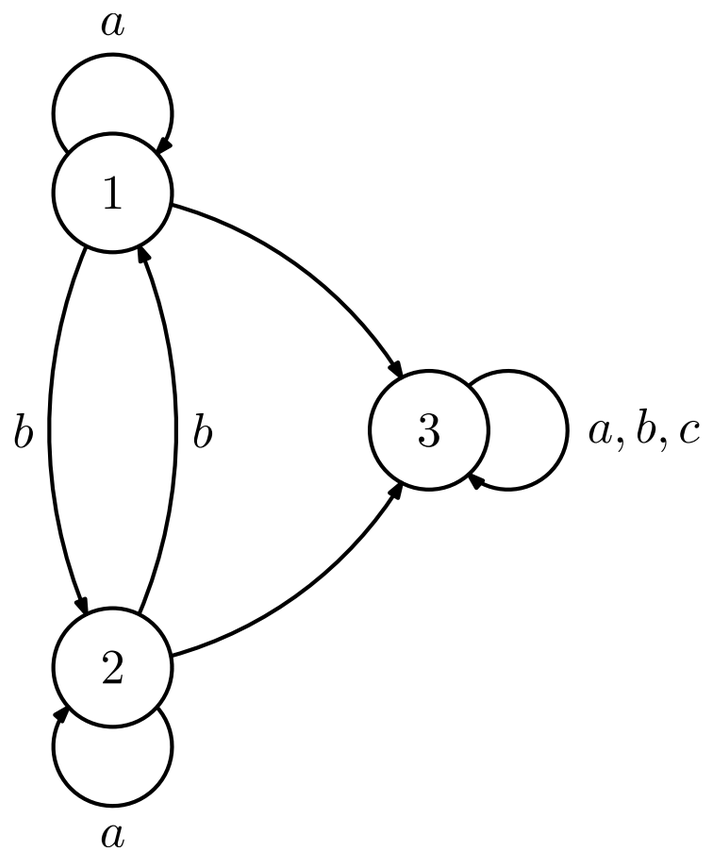
- How to compare states in MDPs?
  - Same actions have same outcomes → states are similar



What if we relabel actions?

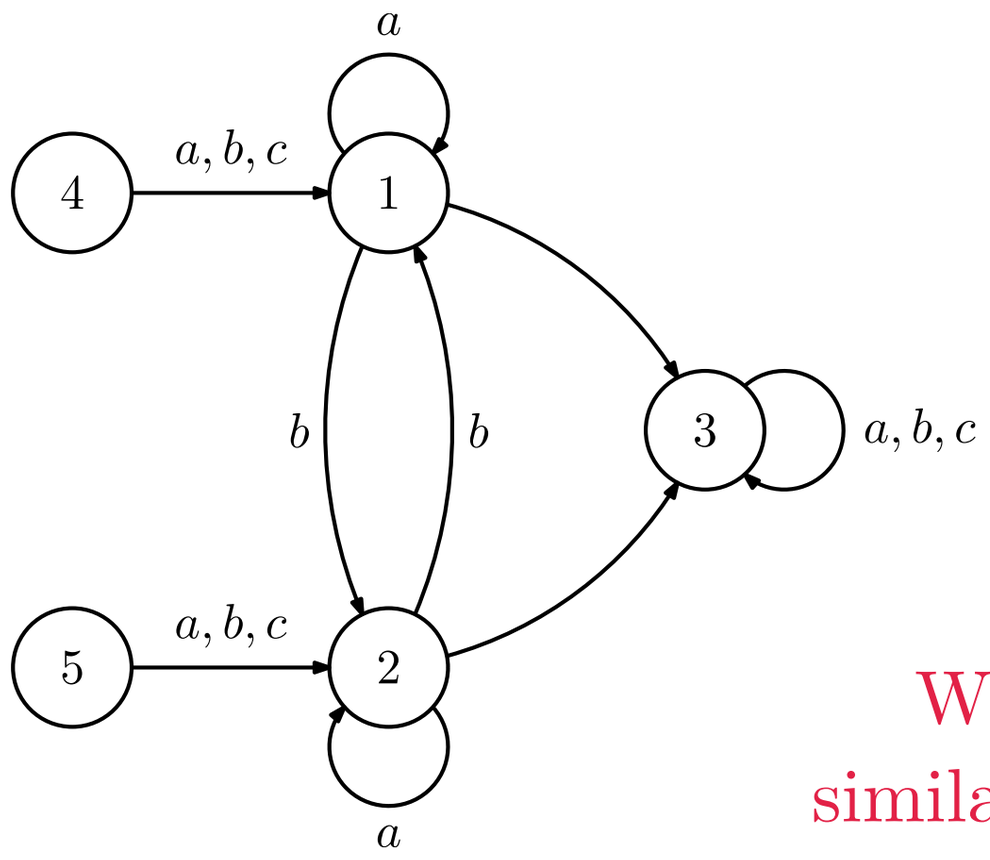
Measuring similarity

- How to compare states in MDPs?
  - Actions have similar outcomes → states are similar



Measuring similarity

- How to compare states in MDPs?
  - Actions have similar outcomes → states are similar



What if actions lead to similar (but different) states?

Measuring similarity

- How to compare states in MDPs?
  - Actions have similar outcomes → states are similar
  - We need some notion of "long term similarity"

↓

Bissimulation metric

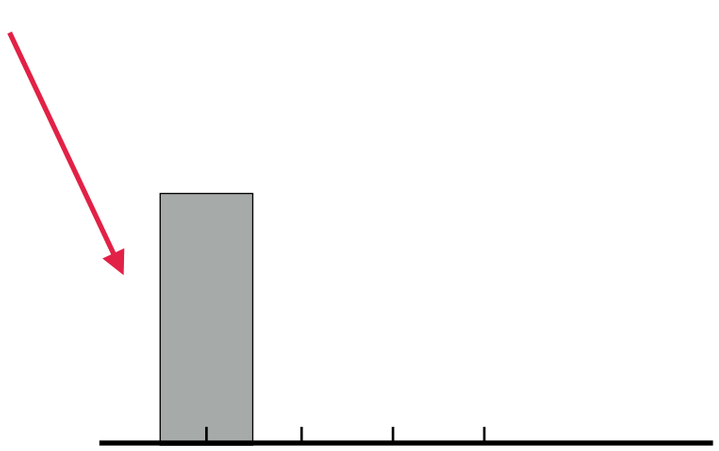
Measuring similarity

- We depart from a distance between states d

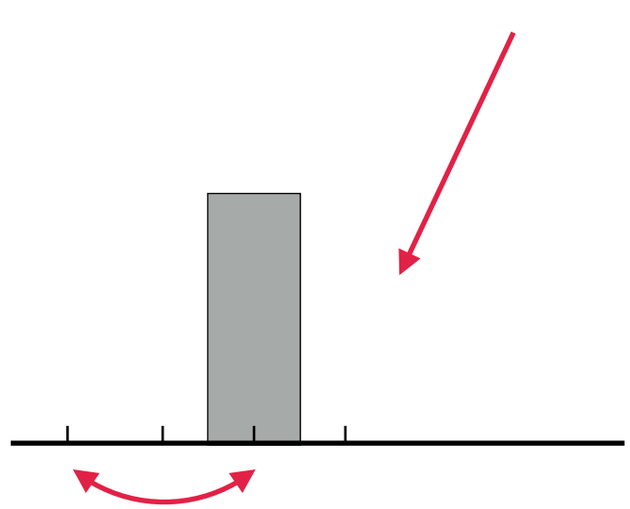
Earth Mover's Distance

  - Build a distance between distributions (EMD)

Distribution A



Distribution B



Distance between points

Measuring similarity

- We depart from a distance between states d
  - Build a distance between distributions (EMD)

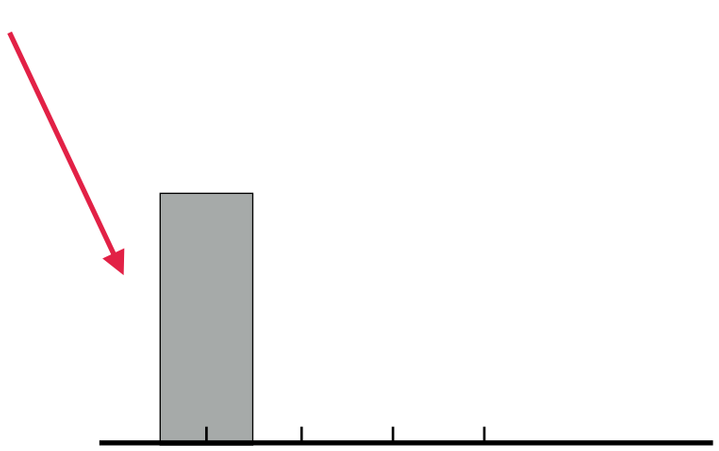
All "mass" must
move 2 units

Distance between A and B: 2

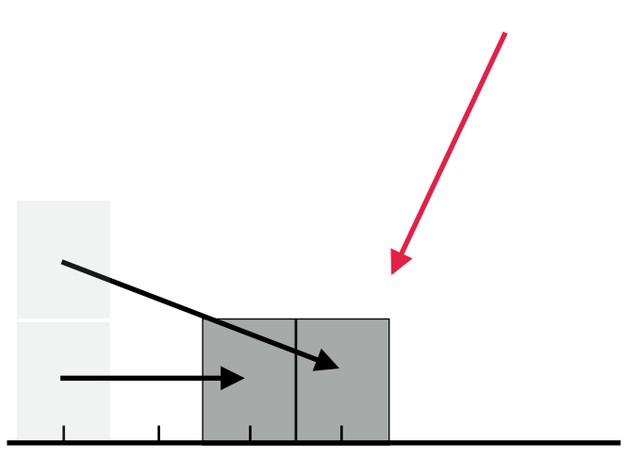
Measuring similarity

- We depart from a distance between states d
  - Build a distance between distributions (EMD)

Distribution A



Distribution C



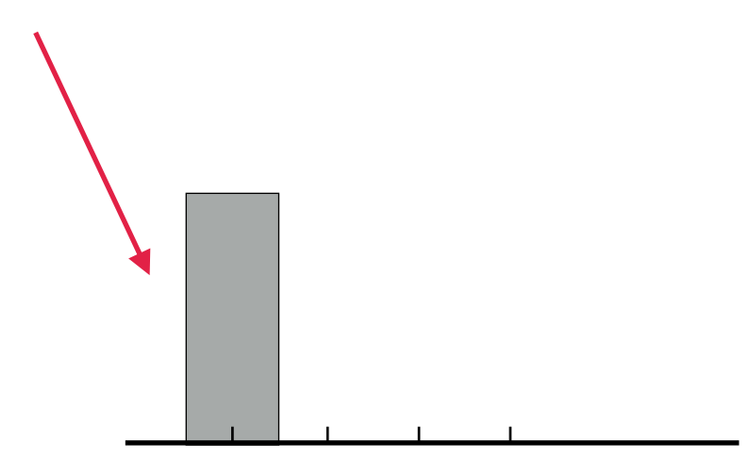
Half the "mass" must move 2 units

Half the "mass" must move 3 units

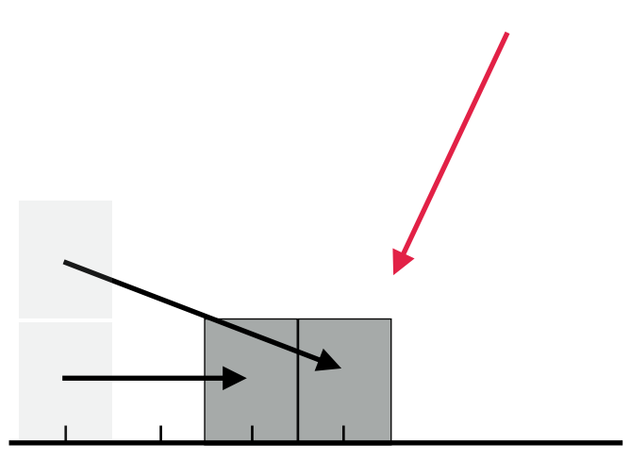
Measuring similarity

- We depart from a distance between states d
  - Build a distance between distributions (EMD)

Distribution A



Distribution C



Distance between A and C: 2.5

Bissimulation metric

- Using the distance d between **distributions...**
- We define a 1-step dissimilarity between a pair $(x,\ a)$ and a pair $(y,\ b)$ as

$$\delta(x, a; y, b) = kd(\mathsf{P}(\cdot \mid x, a), \mathsf{P}(\cdot \mid y, b))$$

Dissimilarity between transition probabilities in $(x,\ a)$ and $(y,\ b)$

Bissimulation metric

- From δ, we remove the dependence on the actions:

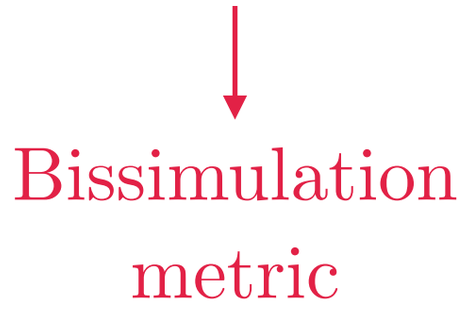
$$d'(x,y) = \max\left\{ \max_{a\in\mathcal{A}} \min_{b\in\mathcal{A}} \delta(x,a;y,b), \max_{b\in\mathcal{A}} \min_{a\in\mathcal{A}} \delta(x,a;y,b) \right\}$$

New distance between states

Dissimilarity of transition probabilities of most similar actions

Bissimulation metric

- We take this new distance, *d*', and repeat the process, to get d"
- We repeat the process until the distance is the same in two consecutive steps



Bissimulation metric

- We can now use bissimulation metric to generalized observed examples
  - E.g., using KNN

- Disadvantage:
  - The bissimulation metric is computationally intensive

TÉCNICO LISBOA

INVERSE

Inverse Reinforcement Learning

Going back to MDPs

- The optimal policy for an MDP can be computed as

$$\pi^*(x) = \arg\min_{a \in \mathcal{A}} Q^*(x, a)$$

$$= \arg\min_{a \in \mathcal{A}} \left[c(x, a) + \gamma \sum_{y \in \mathcal{X}} \mathbf{P}_a(y \mid x) J^*(y) \right]$$

If someone shows the optimal policy, can we recover the task?

Going back to MDPs

- If we are given the optimal policy, then for all $x \in \mathcal{X}$ and all $a \in \mathcal{A}$,

$$J^*(x) \leq Q^*(x, a)$$

Going back to MDPs

- If we are given the optimal policy, then for all $x \in \mathcal{X}$ and all $a \in \mathcal{A}$,

$$\mathbf{c}_\pi + \gamma \mathbf{P}_\pi \mathbf{J}^* \leq \mathbf{c}_a + \gamma \mathbf{P}_a \mathbf{J}^*$$

$\downarrow$

Ignoring dependence on a

Going back to MDPs

- If we are given the optimal policy, then for all $x \in \mathcal{X}$ and all $a \in \mathcal{A}$,

$$\mathbf{c} + \gamma \mathbf{P}_\pi \mathbf{J}^* \leq \mathbf{c} + \gamma \mathbf{P}_a \mathbf{J}^*$$

$$\downarrow$$

$$(\mathbf{P}_\pi - \mathbf{P}_a)\mathbf{J}^* \leq 0$$

$$\downarrow$$

$$(\mathbf{P}_\pi - \mathbf{P}_a)(\mathbf{I} - \gamma \mathbf{P}_\pi)^{-1}\mathbf{c} \leq 0$$

$$\downarrow$$

We're computing cost from policy

Inverse Reinforcement Learning (IRL)

However…

...

- All policies are optimal if $\mathbf{c} \equiv \mathbf{0}$...

$$(\mathbf{P}_\pi - \mathbf{P}_a)(\mathbf{I} - \gamma \mathbf{P}_\pi)^{-1}\mathbf{c} \leq 0$$

↓

Ill-defined problem

IRL

- Original approaches select among possible solutions using heuristics
  - E.g., the difference between the value of best action and second best action is as large as possible

↓

Cumbersome

Arbitrary

Doesn't handle imperfect teachers

A probabilistic approach

- We adopt a probabilistic model for the teacher
  - We do not assume that the teacher is optimal
  - We assume that, given cost c, it selects each action a in state x with probability

$$\pi_c(a \mid x) = \frac{e^{-\eta Q_c^*(x,a)}}{\sum_{a' \in \mathcal{A}} e^{-\eta Q_c^*(x,a')}}$$

Optimal Q for cost c

Confidence in expert

A probabilistic approach

- We adopt a probabilistic model for the teacher
- Assume that the examples are independent
- Inferring the cost can then be done...
- ... using Bayesian inference (assume cost over costs):

$$\mathbb{P}\left[\mathrm{c} = c \mid \mathcal{D}\right] \propto \prod_{n=1}^{N} \pi_c(a_n \mid x_n)\mathbb{P}\left[\mathrm{c} = c\right]$$

Prior

A probabilistic approach

- We adopt a probabilistic model for the teacher
- Assume that the examples are independent
- Inferring the cost can then be done...
- ... using Bayesian inference (assume cost over costs)
- ... using simple maximum-likelihood:

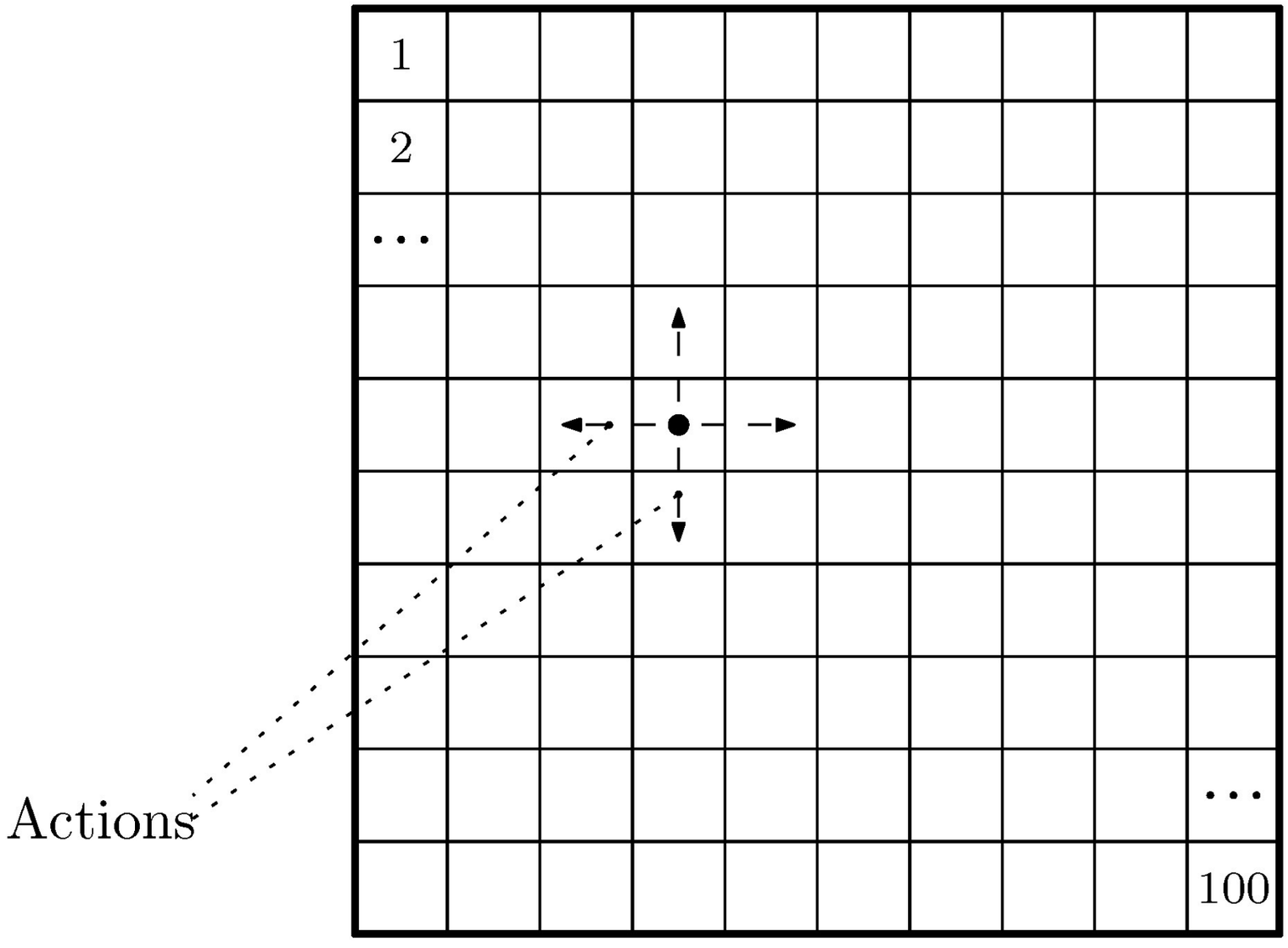
$$c^* = \operatorname*{argmax}_c \sum_{n=1}^{N} \log \pi_c(a_n \mid x_n)$$

Use gradient ascent

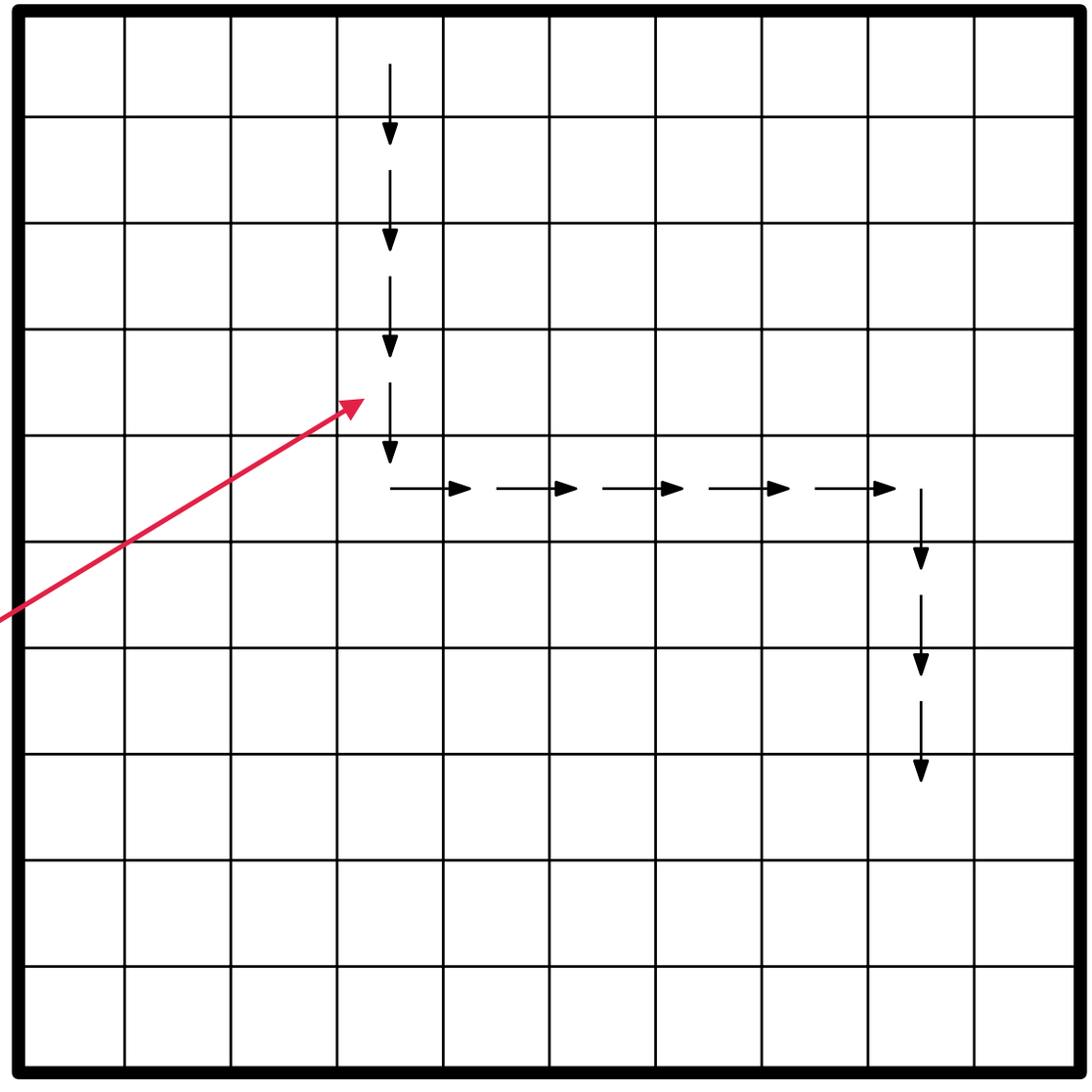
A probabilistic approach

- We adopt a probabilistic model for the teacher
- Assume that the examples are independent
- Inferring the cost can then be done...
  - ... using Bayesian inference (BIRL)
  - ... using simple maximum-likelihood with gradient ascent (GIRL)

Example

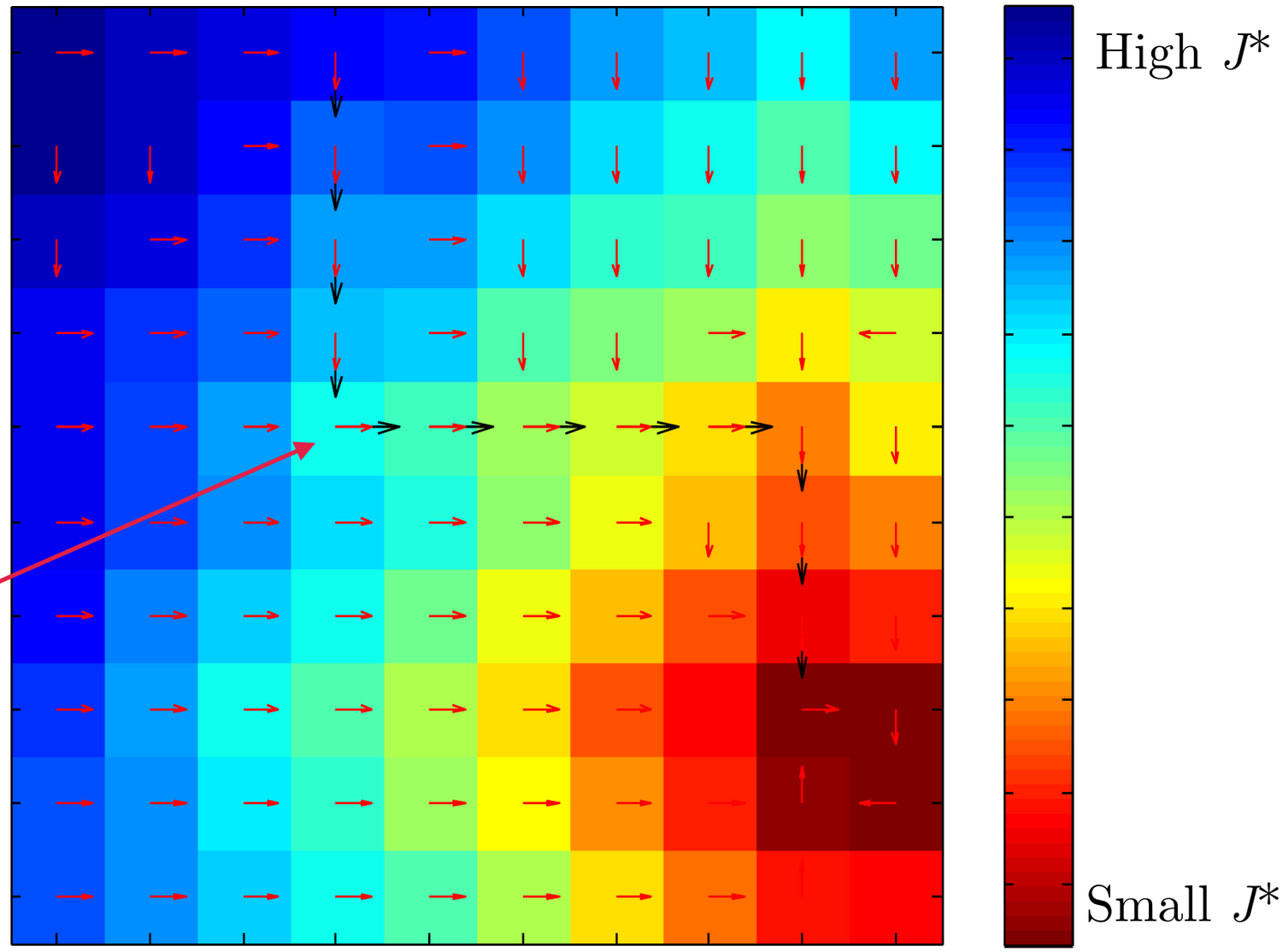


Example



Demonstrated actions

Example



Learned policy follows demonstrations

What if we can't get a demonstration?

Preferences revisited...

Evaluative feedback

- In complex tasks it may be too difficult to:
  - Provide demonstrations of the "desired behavior"
  - Come up with a cost function that expresses the "desired behavior"

It may be possible to compare different behaviors and indicate which one is better...

Evaluative feedback

- Given two trajectories:

$$\tau^0 = \{x_0, a_0, x_1, a_1, \ldots, x_{M-1}, a_{M-1}\}$$

$$\tau^1 = \{x'_0, a'_0, x'_1, a'_1, \ldots, x'_{M-1}, a'_{M-1}\}$$

the "teacher" now indicates whether τ^0 or τ^1 is better, i.e., whether

$$\tau^0 \succ \tau^1 \qquad \text{or} \qquad \tau^1 \succ \tau^0$$

Evaluative feedback

- We thus build a dataset

$$\mathcal{D} = \{(\tau_n^0, \tau_n^1, u_n), n = 1, \ldots, N\}$$

The value of u_n is 0 or 1,
depending on whether
τ_n^0 or τ_n^1 is better

Evaluative feedback

- We thus build a dataset

$$\mathcal{D} = \{(\tau_n^0, \tau_n^1, u_n), n = 1, \ldots, N\}$$

- We now have a binary classification problem!
  - Input: a pair (τ_n^0, τ_n^1)
  - Output: 0 or 1

Reconstructing the cost

- Given two trajectories τ^0 and τ^1, let

Similar expression used for probabilistic IRL

$$\mathbb{P}[\tau^0 \succ \tau^1 \mid c] = \frac{e^{-\sum_{m=0}^{M-1} c(x_m, a_m)}}{e^{-\sum_{m=0}^{M-1} c(x_m, a_m)} + e^{-\sum_{m=0}^{M-1} c(x'_m, a'_m)}}$$

Just adding costs, no discounting

Reconstructing the cost

- Given two trajectories τ^0 and τ^1, let

$$\mathbb{P}[\tau^0 \succ \tau^1 \mid c] = \frac{e^{-\sum_{m=0}^{M-1} c(x_m, a_m)}}{e^{-\sum_{m=0}^{M-1} c(x_m, a_m)} + e^{-\sum_{m=0}^{M-1} c(x'_m, a'_m)}}$$

- We can now optimize $\mathbf{c}$ using maximum likelihood:

$$c^* = \arg\max \sum_{n=0}^{N} u_n \log \mathbb{P}[\tau^1 \succ \tau^0 \mid c] + (1 - u_n)\mathbb{P}[\tau^0 \succ \tau^1 \mid c]$$

Use gradient ascent

Chat-GPT

- Chat-GPT used this same approach to build a reward function to fine tune the GPT-3 model