Learning and Solving Many-Player Games Through a Cluster-Based Representation

Sevan Ficici David Parkes Avi Pfeffer

School of Engineering and Applied Sciences

Harvard University

Game Theory With Many Agents

- We want to model situations with a large number of interacting agents
  - e.g. electronic markets, online gaming
- We want to use game theory

But…

- Game theory is typically restricted to situations with few agents
- The size of the game increases exponentially with the number of agents

⇒ Cannot even represent games with many agents, let alone solve them

One Approach

- Assume that the game has underlying structure, that makes both representation and solution tractable
  - graphical games [Kearns, Littman & Singh 01]
  - action graph games [Bhat & Leyton-Brown 04]
  - anonymous games [Daskalakis & Papadimitriou 07]
  - summarization games [Kearns & Mansour 02]

Our Approach

1. Don’t assume the game itself has structure; instead, find a compact approximation
   - hopefully, the approximation will capture enough of the actual game that the strategies obtained will be good
2. Since we cannot represent the full game, we learn the approximation from data

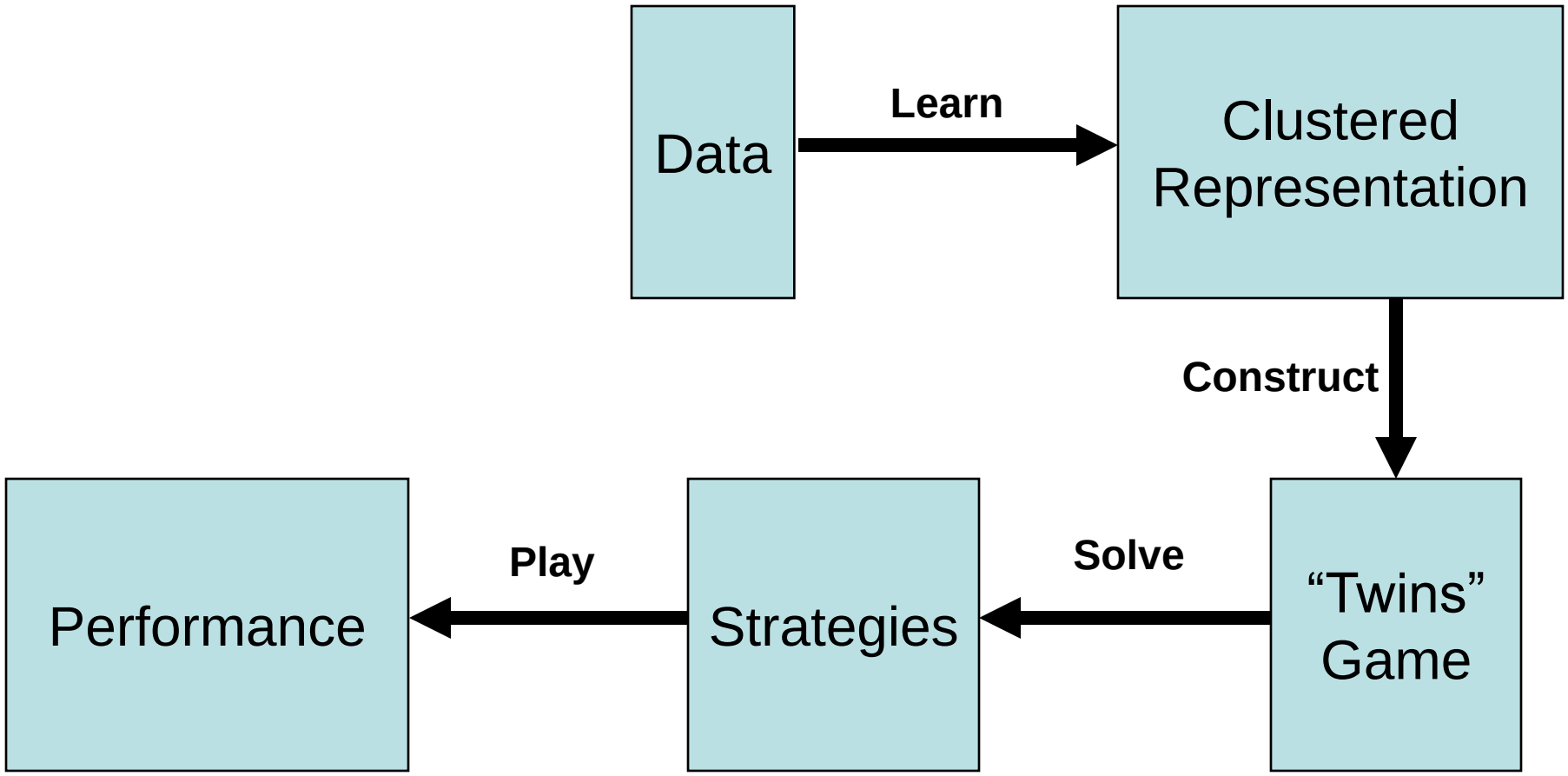
Idea Behind Our Approximation

- In many games, different agents have a similar strategic view of the game
  - similar payoffs
  - similar effect on others' payoffs
  - e.g. companies in the same sector
  - e.g. similar players in an online game
- We cluster similar agents together to obtain a cluster-based representation

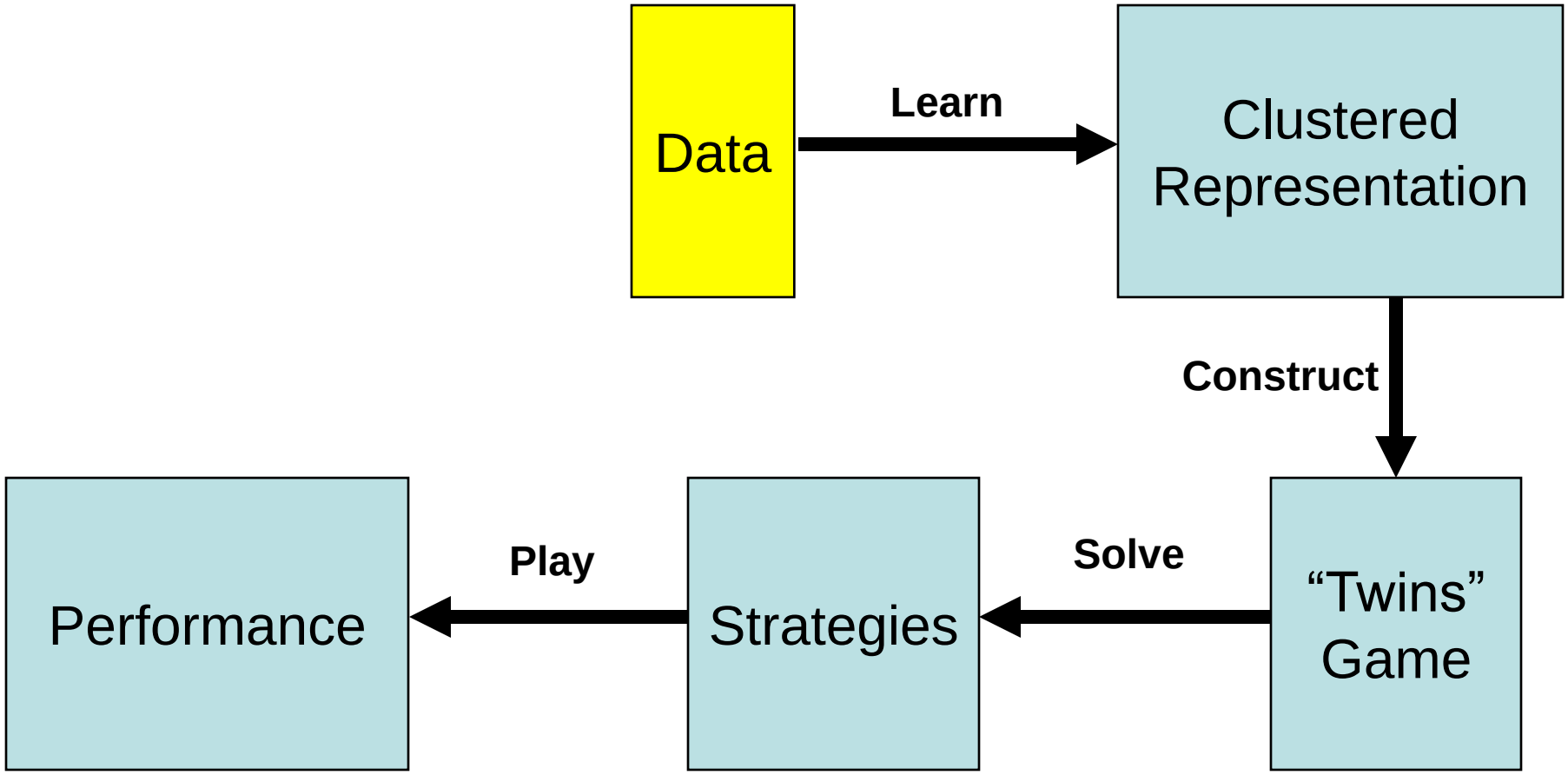
Example: Vendor Game

- There are many vendors, who need to choose a location to sell their goods
- Vendors fall into categories
  - e.g. sandwiches and drinks
- Within categories, individual vendors are further differentiated
  - e.g. lemonade and beer
- Different products may be substitutes or complements
- Substitute/complement relationships hold in general between categories, but there is variation between individual vendors

Overview of the Approach



Overview of the Approach



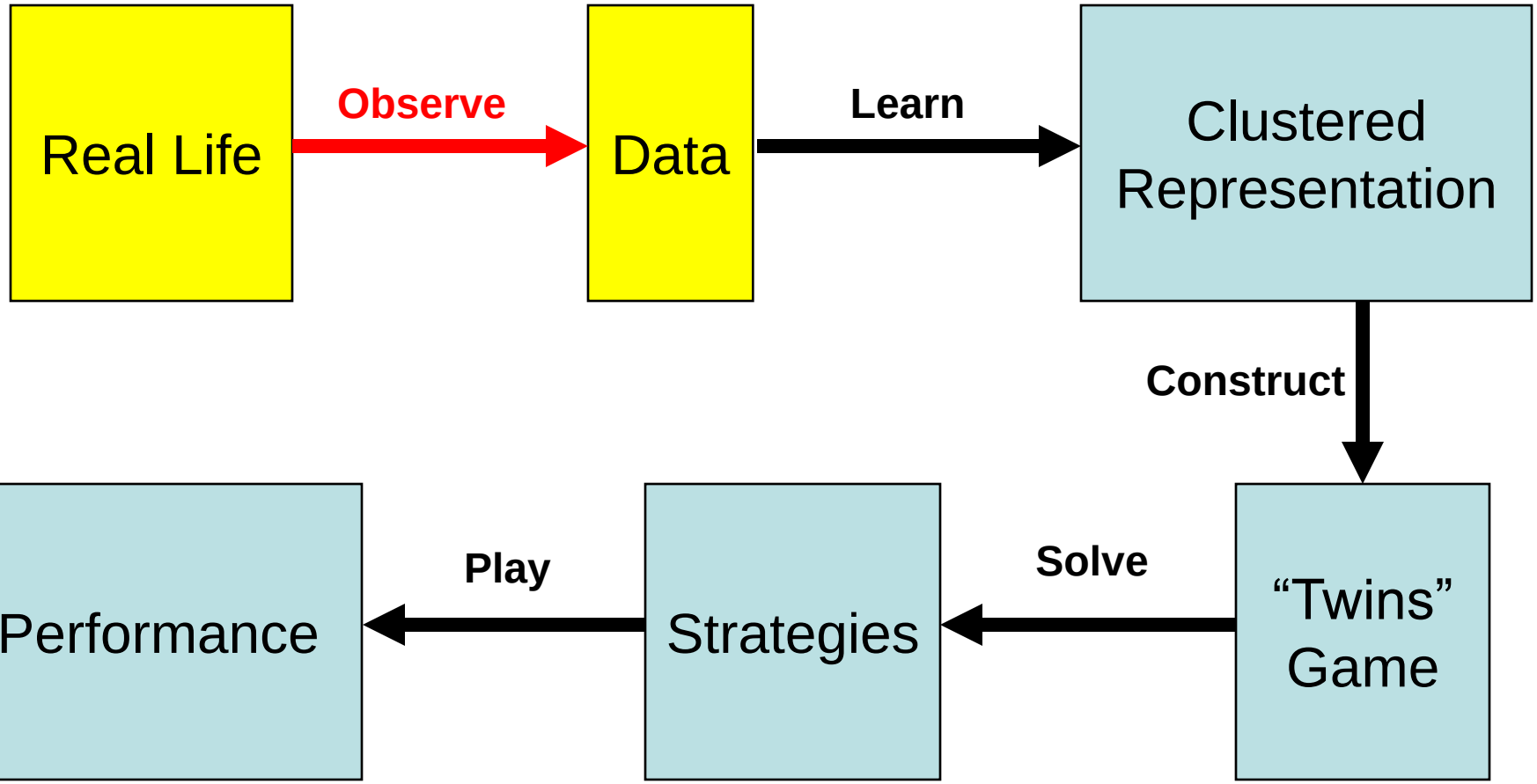
Game Definition

- *N* agents
- Pure strategy set **S**
- Each agent has the same strategy set **S**
- Each agent *i* has a payoff function that maps strategy profiles of all agents to utilities

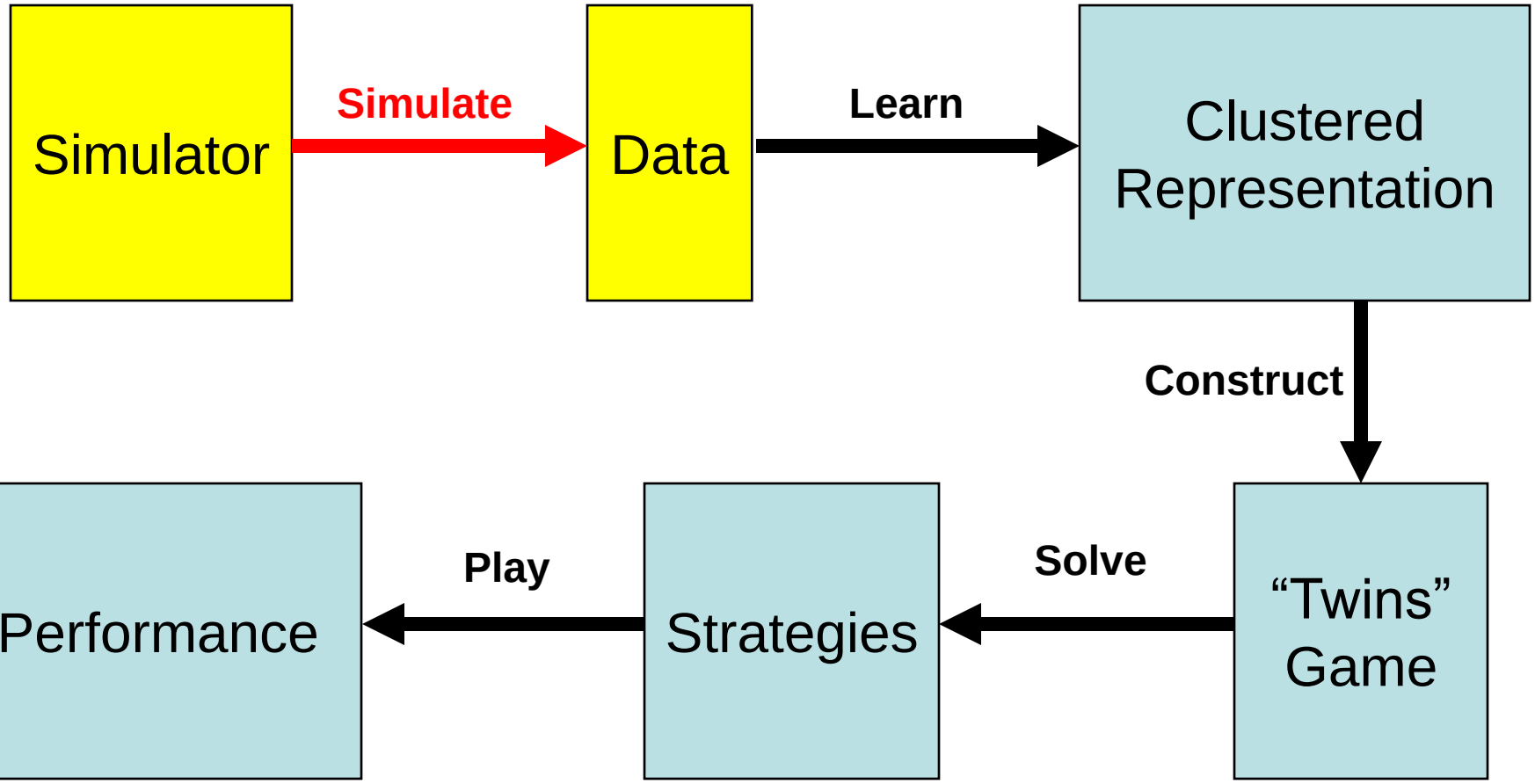
Data

Agent	1	2	3	4	5	6	7	8	9	10
Round 1 Action	L	R	L	L	L	R	R	R	L	L
Round 1 Payoff	1	-1	4	2	0	-3	3	5	2	0
Round 2 Action	R	L	R	R	L	R	R	L	R	R
Round 2 Payoff	3	-1	1	3	2	0	4	-3	-2	-1
Round 3 Action	L	L	R	L	R	L	L	R	R	L
Round 3 Payoff	-2	4	3	5	0	2	-3	2	1	3

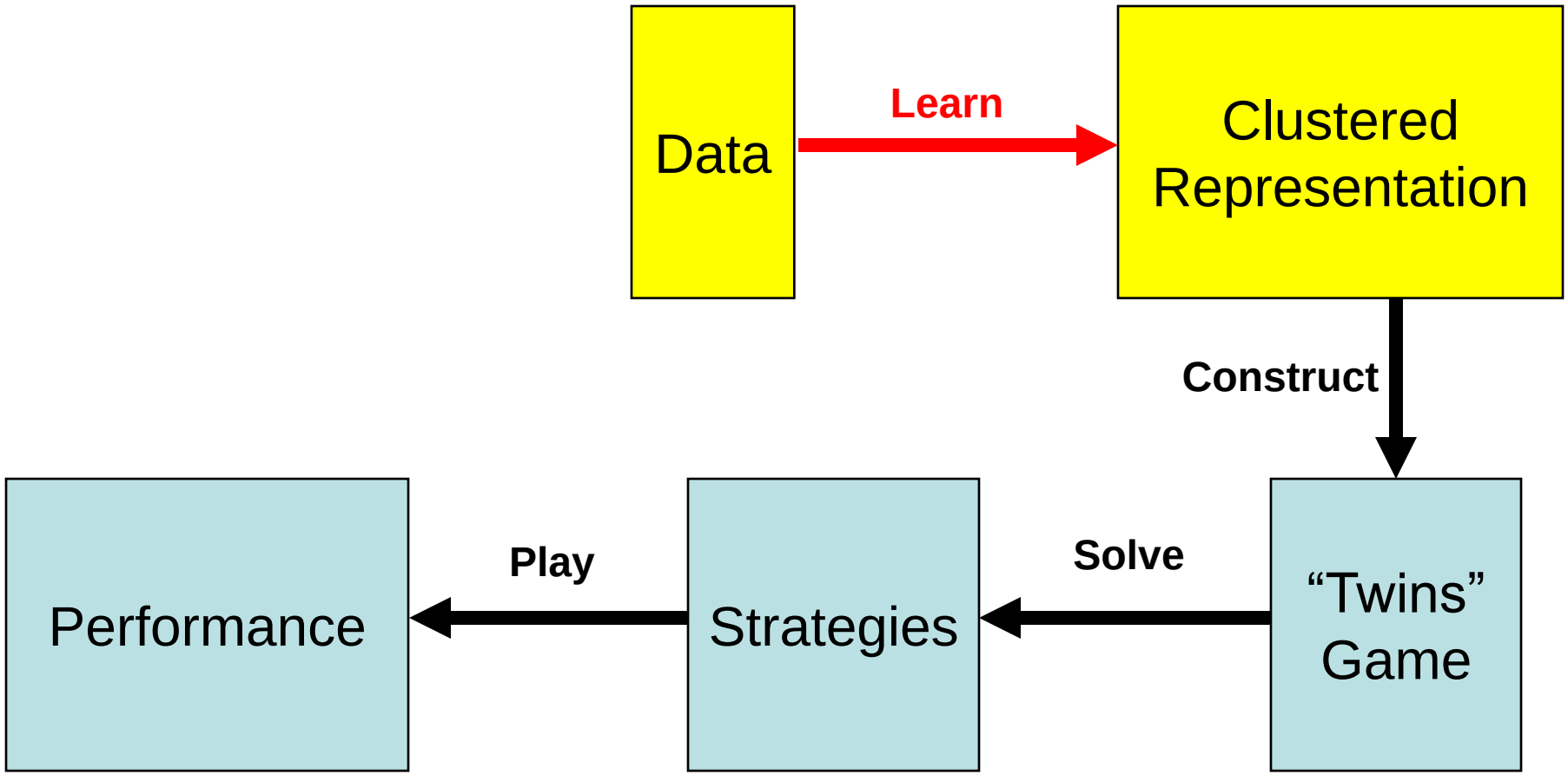
Where Does the Data Come From?



Where Does the Data Come From?



Where Does the Data Come From?



Clustered Representation

- Cluster *N* agents into *K* clusters
- Assumptions of our approximation (not of the real game)
  - agents in the same cluster receive the same payoffs when they play the same strategies
  - agents in the same cluster have the same influences on other agents
  - payoffs depend only on proportion of agents in each cluster playing each strategy
  - payoffs are linear in those proportions

Regression Equations

- For each cluster C and pure strategy s, we specify a regression equation
  - specifies the payoff obtained by an agent in cluster C playing s, given the proportion of agents in each cluster playing each strategy
- Number of terms in each equation is $|\mathbf{S}|^K + 1$
  - one term for each cluster strategy profile, and a constant

Example: 2 Clusters, A and B

Parameter for cluster strategy profile LL for agent in cluster *A* playing L

Proportion of agents in cluster *A* who play L

Proportion of agents in cluster *B* who play L

$$\hat{\pi}_A(\mathrm{L}) = \beta^{\mathrm{LL}}_{A(\mathrm{L})} P(\mathrm{L} \mid A) P(\mathrm{L} \mid B)$$

Estimated payoff for agent in cluster *A* for playing L

Example : 2 Clusters, A and B

$$\hat{\pi}_A(\mathrm{L}) = \beta^{\mathrm{LL}}_{A(\mathrm{L})} P(\mathrm{L}\,|\,A)P(\mathrm{L}\,|\,B)$$
$$+ \beta^{\mathrm{LR}}_{A(\mathrm{L})} P(\mathrm{L}\,|\,A)P(\mathrm{R}\,|\,B)$$
$$+ \beta^{\mathrm{RL}}_{A(\mathrm{L})} P(\mathrm{R}\,|\,A)P(\mathrm{L}\,|\,B)$$
$$+ \beta^{\mathrm{RR}}_{A(\mathrm{L})} P(\mathrm{R}\,|\,A)P(\mathrm{R}\,|\,B)$$
$$+ \beta_{A(\mathrm{L})}$$

Estimated payoff for agent in cluster *A* for playing L

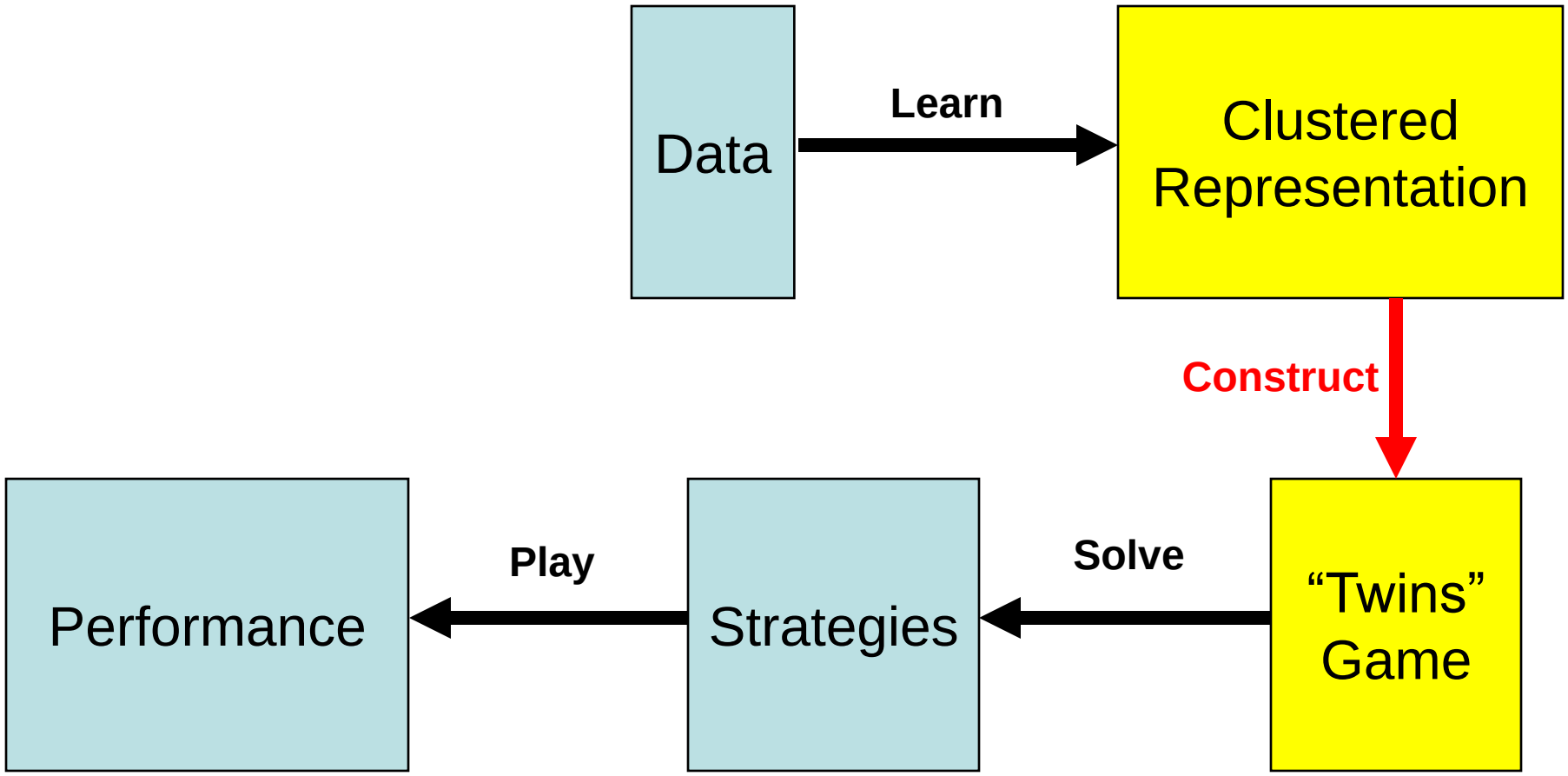
Learning: Linear Regression

- Given a clustering, each agent's payoff in each round gives us
  - the cluster of the agent
  - the strategy of the agent
  - the proportion of agents in each cluster playing each strategy
  - the payoff to an agent in that cluster for playing that strategy
- A round gives us *N* estimates of the regression equations

Learning: Clustering

- We don’t assume the clusters are known in advance
- Instead, we learn the clusters from data
- We use k-means
  - features are the average payoffs agents receive for different strategies

Overview of the Approach



A Natural Idea

- Construct a *K*-player game
  - each cluster represented by one player
- Estimate the payoffs using regression equations
- Solve the game, and instruct all agents in a cluster to play the cluster's equilibrium strategy

Problem with this Idea

- Suppose a cluster's optimal strategy in this game is to play L
- We will instruct all agents in the cluster to play L
- But if all other agents in the cluster are playing L, maybe an agent will want to play R
  - especially if agents in the cluster are substitutes
- So individual agents may have incentive to deviate from the prescribed strategies
- We say that the representation is not **individually responsive**

The Twins Game

- Better idea: construct a 2*K*-player game
- Each cluster *C* represented by two identical twin players $\mathbb{C}$ and $\mathbb{C}$'
- $\mathbb{C}$ and $\mathbb{C}$' have symmetric payoff functions, and the same effect on others' payoffs

Interpretation of Twins

- From the point of view of $\mathbb{C}$:
  - $\mathbb{C}$ represents a single agent in cluster *C*
  - $\mathbb{C}$' represents the cluster *C* in aggregate
- From the point of view of $\mathbb{C}$':
  - $\mathbb{C}$' represents a single agent in cluster *C*
  - $\mathbb{C}$ represents the cluster *C* in aggregate
- From the point of view of a player corresponding to a different cluster:
  - $\mathbb{C}$ and $\mathbb{C}$' together represent cluster *C*

Computing Payoffs in Twins Game

$\mathbb{A}$ plays L, $\mathbb{A}$' plays R, $\mathbb{B}$ plays L, $\mathbb{B}$' plays L

Computing Payoffs in Twins Game

$\mathbb{A}$ plays L, $\mathbb{A}$' plays R, $\mathbb{B}$ plays L, $\mathbb{B}$' plays L

Payoff to player $\mathbb{A}$: $\beta^{\mathrm{RL}}_{A(\mathrm{L})} + \beta_{A(\mathrm{L})}$

Payoff to agent in cluster *A* for playing L, when cluster *A* as a whole plays R, and cluster *B* plays L

Computing Payoffs in Twins Game

$\mathbb{A}$ plays L, $\mathbb{A}$' plays R, $\mathbb{B}$ plays L, $\mathbb{B}$' plays L

Payoff to player $\mathbb{A}$: $\beta^{\mathrm{RL}}_{A(\mathrm{L})} + \beta_{A(\mathrm{L})}$

Payoff to player $\mathbb{A}$': $\beta^{\mathrm{LL}}_{A(\mathrm{R})} + \beta_{A(\mathrm{R})}$

Payoff to agent in cluster *A* for playing R, when cluster *A* as a whole plays L, and cluster *B* plays L

Computing Payoffs in Twins Game

$\mathbb{A}$ plays L, $\mathbb{A}$' plays R, $\mathbb{B}$ plays L, $\mathbb{B}$' plays L

Payoff to player $\mathbb{A}$: $\beta^{\mathrm{RL}}_{A(\mathrm{L})} + \beta_{A(\mathrm{L})}$

Payoff to player $\mathbb{A}$': $\beta^{\mathrm{LL}}_{A(\mathrm{R})} + \beta_{A(\mathrm{R})}$

Payoff to player $\mathbb{B}$: $\frac{\beta^{LL}_{B(L)} + \beta^{RL}_{B(L)}}{2} + \beta_{B(L)}$

Payoff to agent in cluster *B* for playing L, when cluster *B* as a whole plays L, and half of cluster *A* plays L and the other half plays R

Computing Payoffs in Twins Game

$\mathbb{A}$ plays L, $\mathbb{A}$' plays R, $\mathbb{B}$ plays L, $\mathbb{B}$' plays L

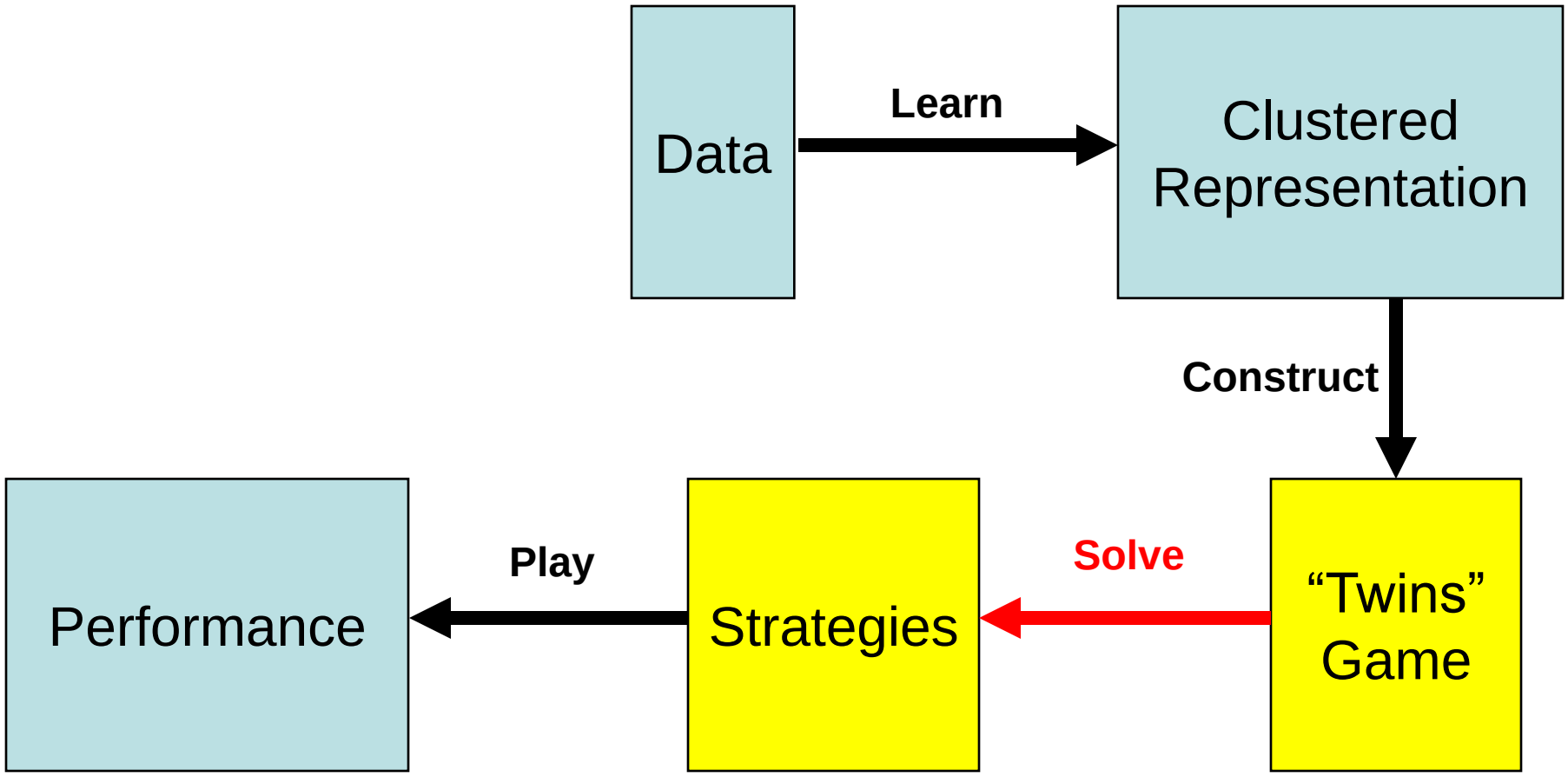
Payoff to player $\mathbb{A}$: $\beta^{\mathrm{RL}}_{A(\mathrm{L})} + \beta_{A(\mathrm{L})}$

Payoff to player $\mathbb{A}$': $\beta^{\mathrm{LL}}_{A(\mathrm{R})} + \beta_{A(\mathrm{R})}$

Payoff to player $\mathbb{B}$: $\frac{\beta^{LL}_{B(L)} + \beta^{RL}_{B(L)}}{2} + \beta_{B(L)}$

Payoff to player $\mathbb{B}$': $\frac{\beta^{LL}_{B(L)} + \beta^{RL}_{B(L)}}{2} + \beta_{B(L)}$

Overview of the Approach



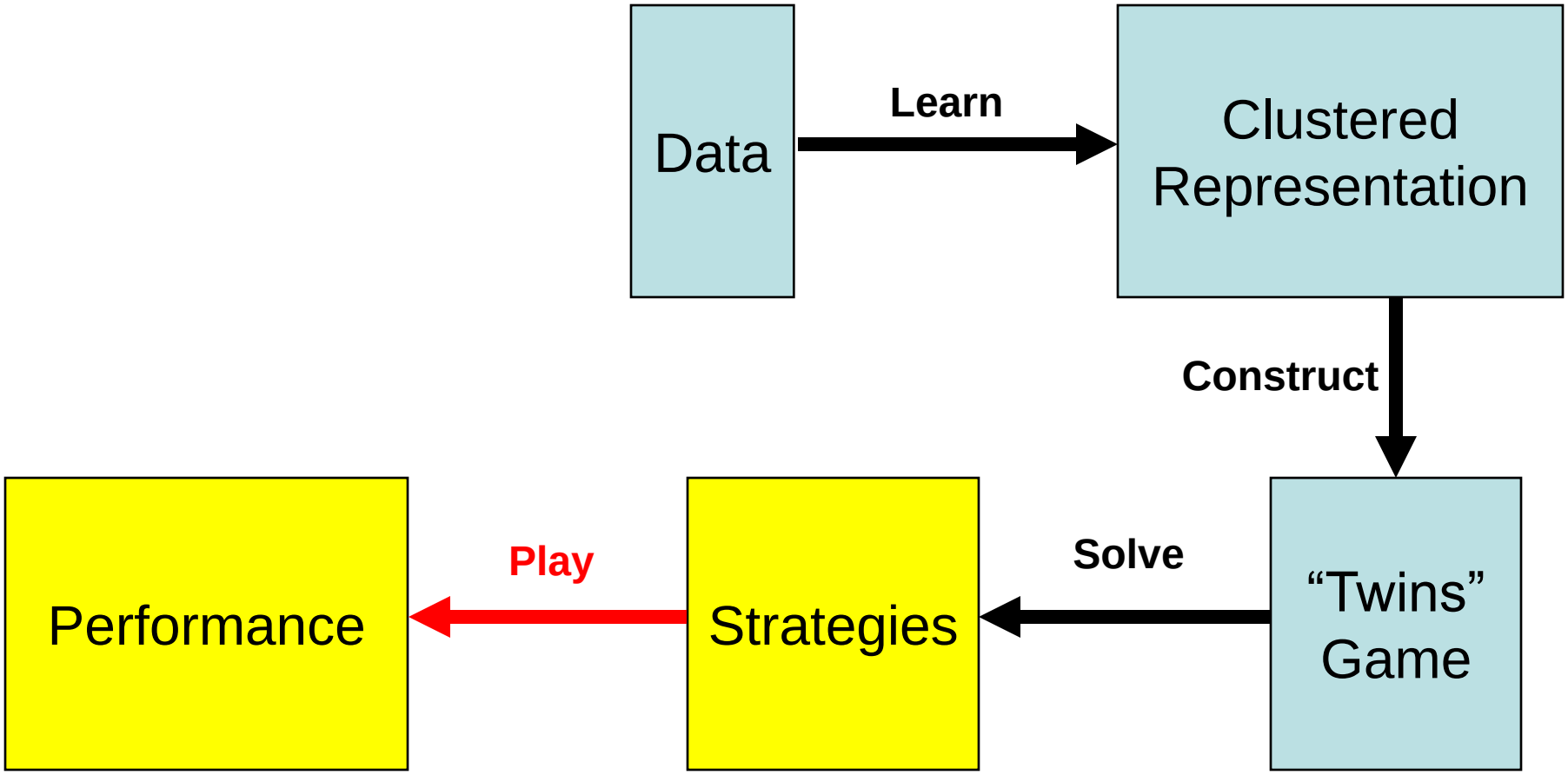
Twin-Symmetric Equilibrium

- Three step process:
  1. Solve the twins game for Nash equilibrium
  2. Assign strategies of twins to corresponding clusters
  3. Assign cluster strategies to individual agents
- To accomplish step 2, we need Nash equilibria in which both twins representing a cluster play identical (mixed) strategies
- Such equilibria are called **twin-symmetric Nash equilibria (TSNE)**
  - guaranteed to exist

Individual Responsiveness

- In a NE, player $\mathbb{C}$ plays a best response to the other players, including $\mathbb{C}$'
- By construction, the payoff to $\mathbb{C}$ is the payoff to an individual agent in *C* when the cluster as a whole plays as $\mathbb{C}$' does
- Thus the agent's strategy is a best response to the strategy of player $\mathbb{C}$'
- But in a TSNE, the strategy assigned to cluster *C* is the same as that of player $\mathbb{C}$' (and $\mathbb{C}$)
- Therefore each agent plays a best response to its cluster's strategy, and no agent has incentive to deviate from its cluster's strategy

Overview of the Approach



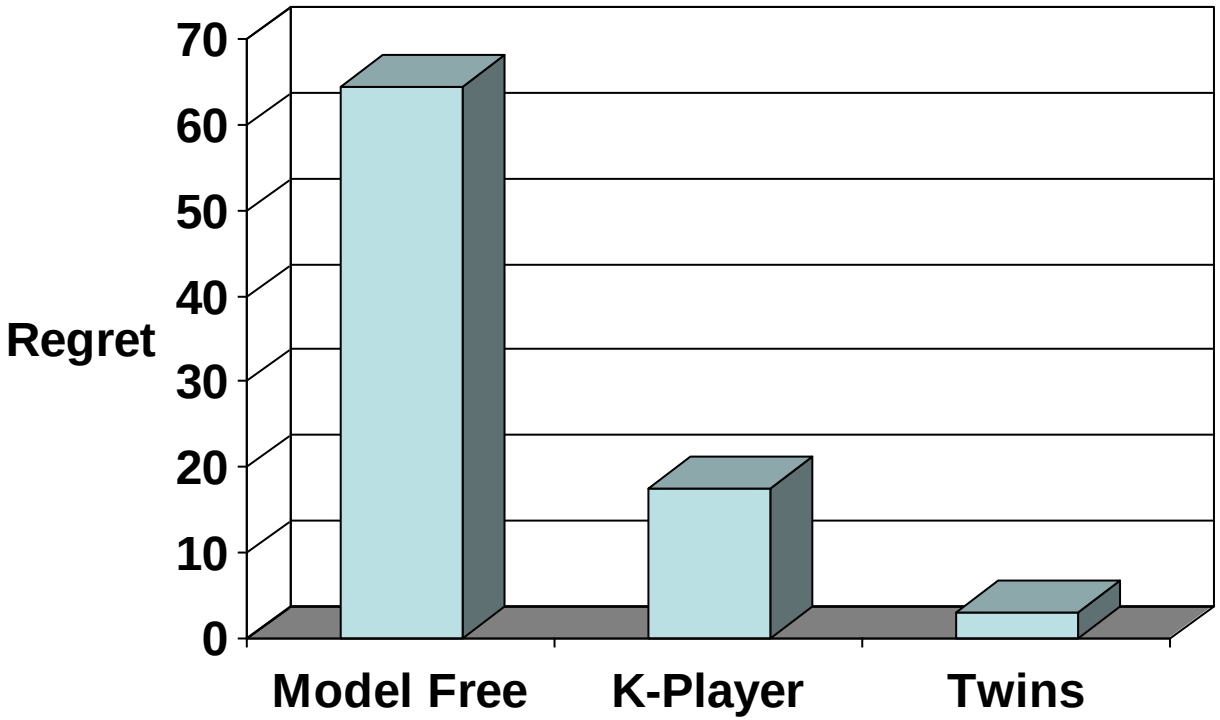
Measuring Performance

- After solving the twins game and assigning strategies to agents, the agents all play the game for some number of rounds
- We compute average regret
  - **regret:** how much better an agent could have done if it picked the best possible fixed strategy, given what all other agents did
    - long-run regret for Nash equilibrium is zero

Methods to Compare

- Our 2*K*-player twins game
- The *K*-player game
- A model free approach
  - each agent picks the action that would have maximized its past rewards
- [Wellman et al. 05] (WEL)
  - only for symmetric games
  - no learning

Results for Vendor Game



100 agents, 2 clusters, 15 observations

results averaged over 10 trials

Other Observations

- Similar results under a variety of conditions
- We are able to learn quite well with as few as 3 rounds of play in the data set
- Our method scales up well to 200 agents
  - and beyond?

Santa Fe Bar Problem

- *N* agents have to decide whether or not to go to the El Farol bar
- If the bar is not full, agents prefer to go to the bar
- If the bar is full, agents prefer to stay home
- Properties
  - symmetric game (one cluster)
  - payoff is non-linear in proportion of agents who go to the bar

Experimental Setup

- For our approach, we construct a 2-player twins game
  - our method produces symmetric equilibria
- We compare to WEL with 2 and 5 clusters
  - 2 clusters is a direct comparison
  - WEL produces both symmetric and asymmetric equilibria, so we compare to both
- We varied the bar capacity

Results for Santa Fe Bar

- When the capacity of the bar is an exact multiple of the cluster size, WEL is able to find an exact asymmetric NE
- Otherwise, our method
  - gets lower regrets than WEL with two clusters
  - gets slightly higher regrets than WEL with five clusters
  - is not sensitive to bar capacity

Conclusion

- We present a method that learns an approximate representation of a many-playered game from data
- We cluster the agents into *K* clusters, and estimate payoffs by linear regression
- We construct and solve a 2*K*-player twins game to ensure individual responsiveness
- Results show that our method works well with few observations and many agents
- Future work:
  - good algorithms for finding TSNE
  - more sophisticated clustering algorithms