Who is Undercover

Yucong Zhang **Yuyan Zhou** **Ming Zhong** **Fengming Zhu**

Game review

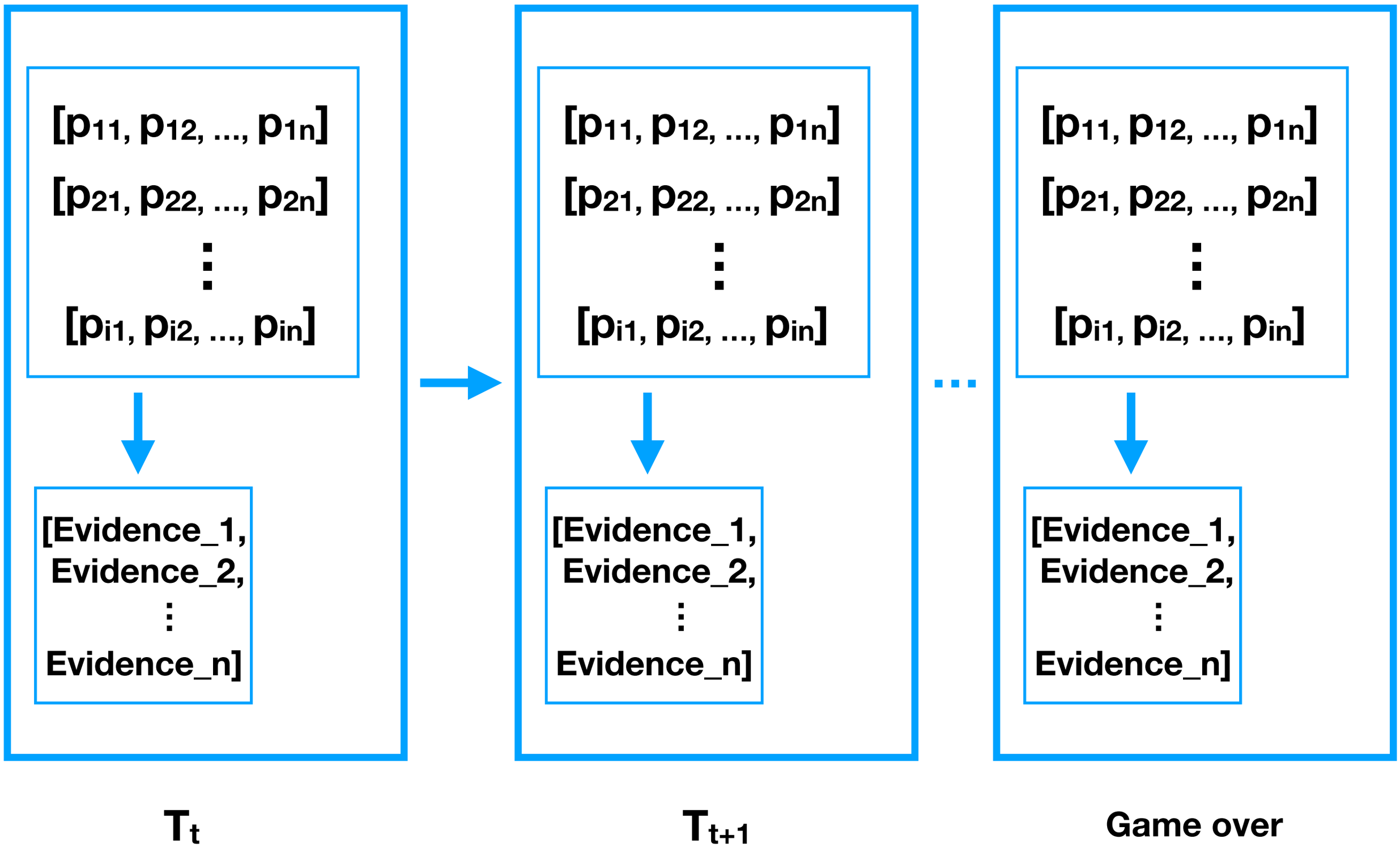
- Several players [i AI agents and (n-i) human players], one undercover

- Given 2 similar/relative item, everyone is anonymous

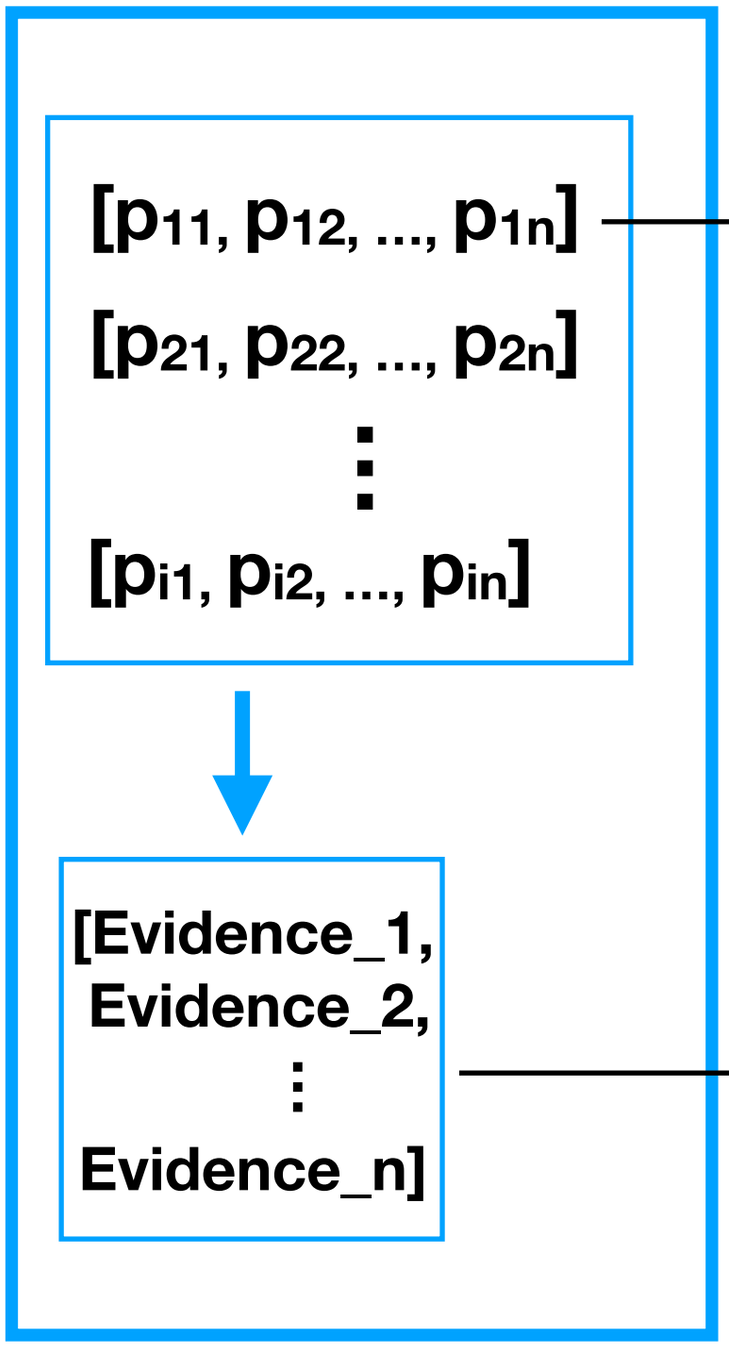
- Each round,
  - give a word to describe your item
  - try to vote for kicking out the potential undercover

- The undercover tries to survive

- HMM


[p11, p12, ..., p1n]
[p21, p22, ..., p2n]
[pi1, pi2, ..., pin]
[Evidence_1,
Evidence_2,
Evidence_n]
[p11, p12, ..., p1n]
[p21, p22, ..., p2n]
[pi1, pi2, ..., pin]
[Evidence_1,
Evidence_2,
Evidence_n]
[p11, p12, ..., p1n]
[p21, p22, ..., p2n]
[pi1, pi2, ..., pin]
[Evidence_1,
Evidence_2,
Evidence_n]
Tt
Tt+1
Game over

- HMM



T_t

A ‘belief’ vector:

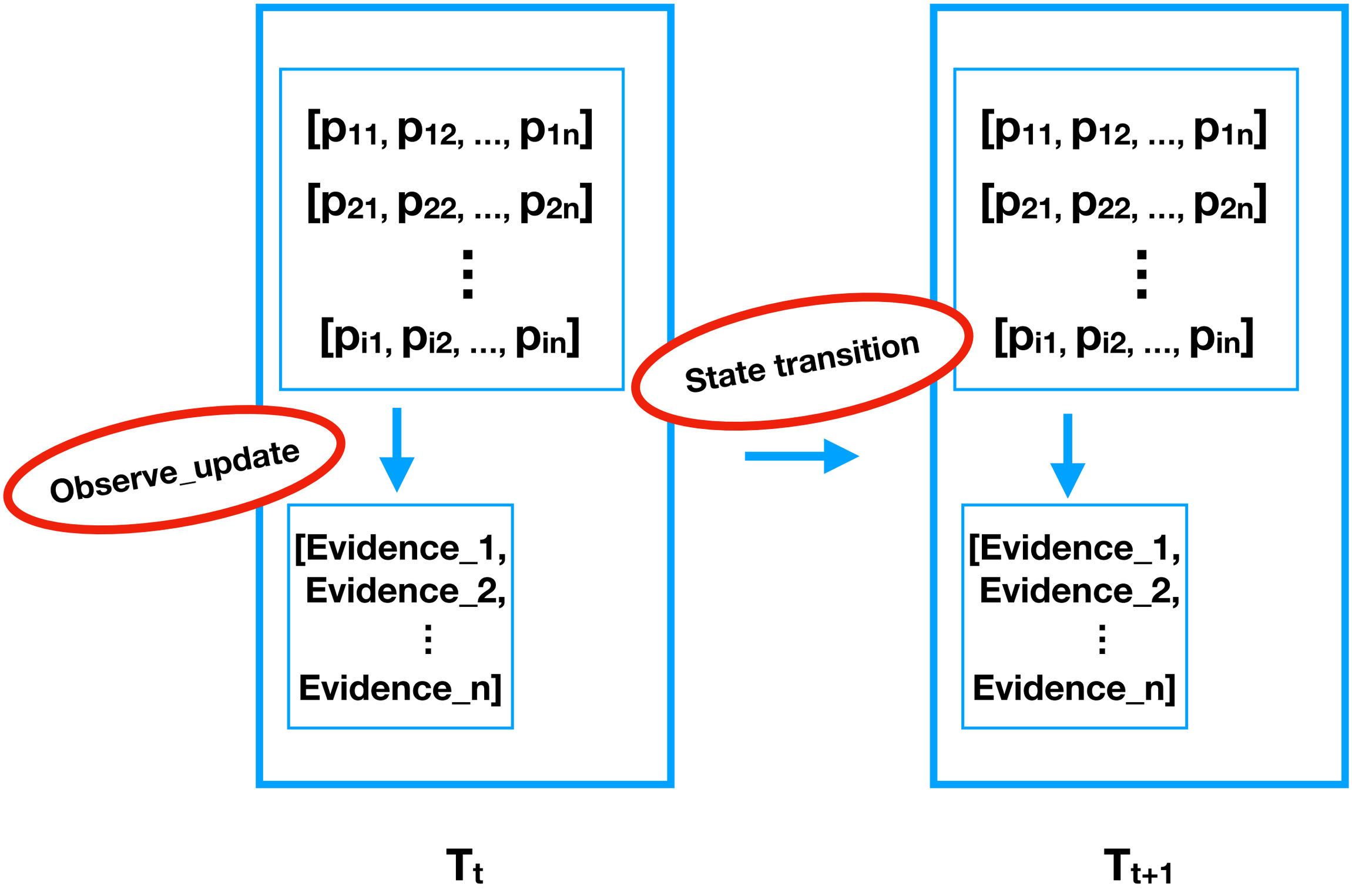
Each agent_i hold its ‘belief’ vector

P_{ij} = From this agent_i ’s perspective, the probability of player_j being he undercover.

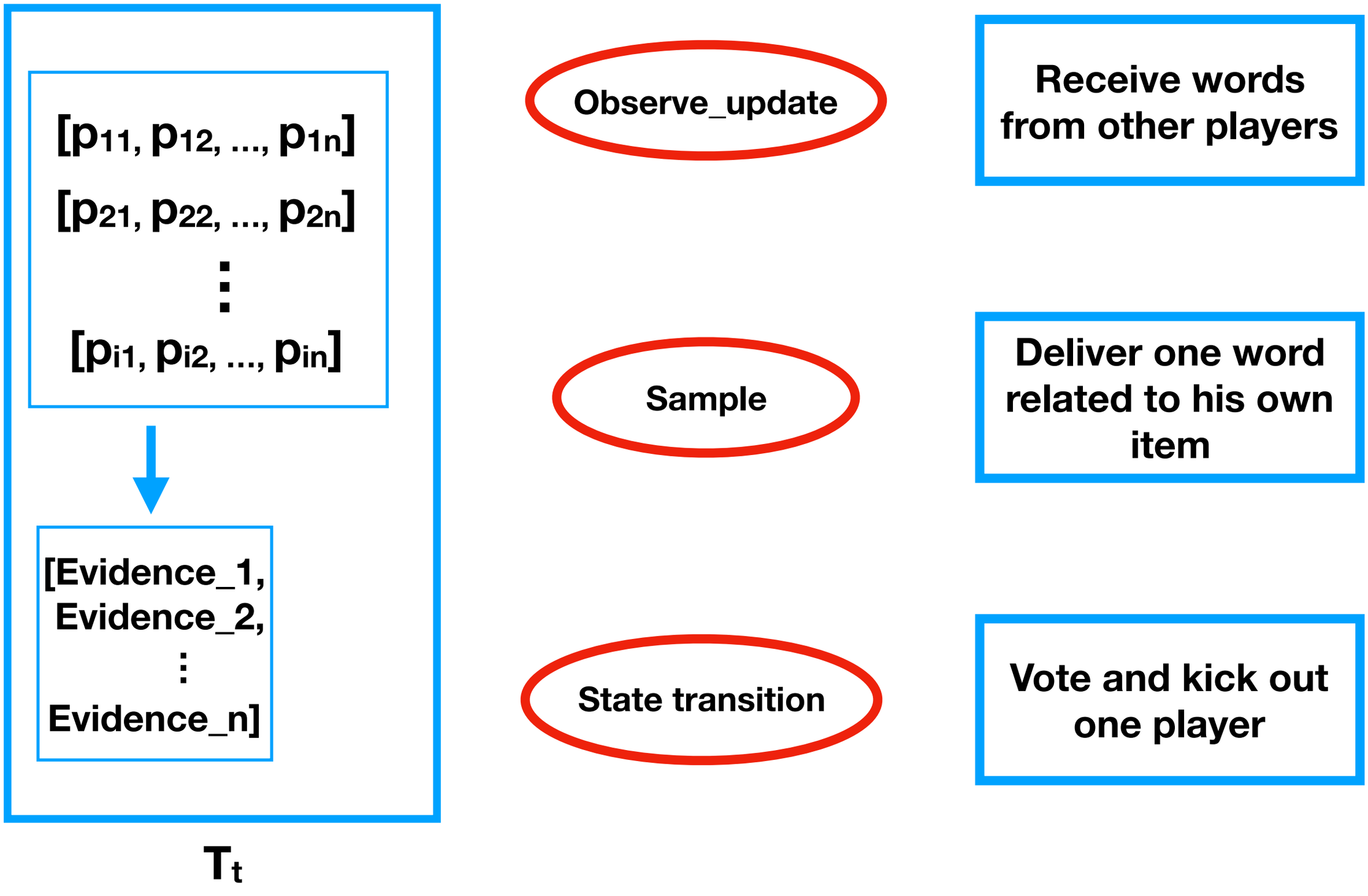
An ‘evidence’ vector:

Evidence_i is a word delivered by player_i in this round.

- HMM


[p11, p12, ..., p1n]
[p21, p22, ..., p2n]
[pi1, pi2, ..., pin]
State transition
Observe_update
[Evidence_1,
Evidence_2,
Evidence_n]
[p11, p12, ..., p1n]
[p21, p22, ..., p2n]
[pi1, pi2, ..., pin]
[Evidence_1,
Evidence_2,
Evidence_n]
Tt
Tt+1

- HMM - mapping into the game



- HMM - observe_update

Purpose:

For agent_i:

$[p_{i1}, p_{i2}, \ldots, p_{in}]$ ⟶ **Probabilities of each being undercover**

↓ **After receiving words**

$[p'_{i1}, p'_{i2}, \ldots, p'_{in}]$

[If only one agent]

$[p_{i1}, p_{i2}, \ldots, p_{in}]$

↓ **Degenerate to**

$[p_1, p_2, \ldots, p_n]$

- HMM - observe_update - e.g. [1 AI agent, 4 human players]

Lemma: [$p_1, p_2, ..., p_5$] updating

P_1:

Probability of himself being undercover.

float[num_player-1] P'_list:

Each element is the probability of other player's card being same as his.

P(the agent is not undercover)
= P(the agent is not undercover, another player's word is different from his)
+ P(the agent is not undercover, another player's word is same as his)
= 1-P_1

P' = P(another player's word is same as the agent)
= P(another player's word is same as the agent, agent is undercover)
+ P(another player's word is same as the agent, agent is not undercover)
= 0+ P(another player's word is same as the agent, agent is not undercover)

P_1 ← P(another player's word is same as the agent, agent is not undercover)
= P(agent is not undercover, another player's word is same as the agent)
= 1-P_1-P'

- HMM - sampling

For agent_i:

$[p_{i1}, p_{i2}, ..., p_{in}]$ ⟶ **Probabilities of each being undercover**

$[1-p_{i1}, 1-p_{i2}, ..., 1-p_{in}]$ ⟶ **Probabilities of each being innocent**

Self evaluation:

Pii ⟶ **Probabilities of himself being undercover**

1-Pii ⟶ **Probabilities of himself being innocent**

- HMM - sampling - e.g. [1 AI agent, 4 human player]

After receiving words: [word1, word2, word3, word4]

[own_item, word1, word2, word3, word4]

Doesn’t match most

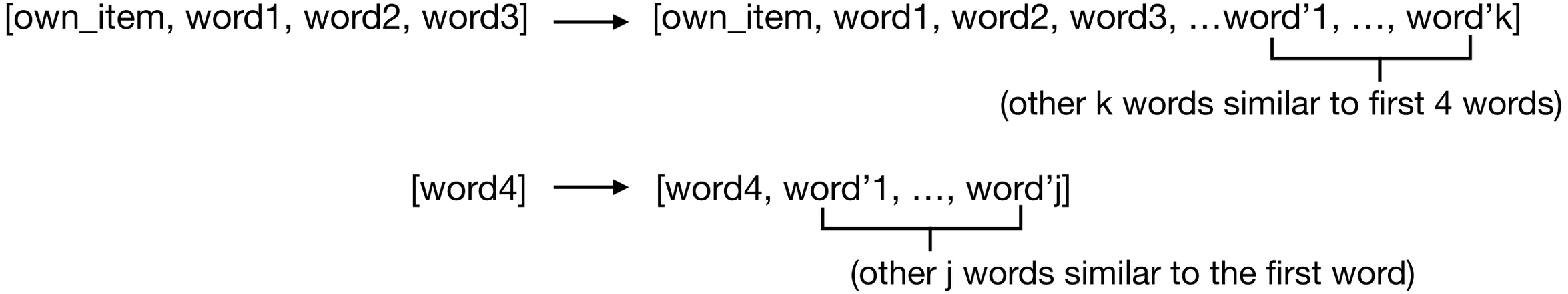
[own_item, word1, word2, word3] [word4]

- HMM - sampling - e.g. [1 AI agent, 4 human players]

[own_item, word1, word2, word3, word4]

Doesn't match most

[own_item, word1, word2, word3] [word4]



- HMM - sampling - e.g. [1 AI agent, 4 human players]

[own_item, word1, word2, word3] ⟶ [own_item, word1, word2, word3, …word'1, …, word'k]

(other k words similar to first 4 words)

[word4] ⟶ [word4, word'1, …, word'j]

(other j words similar to the first word)

[own_item, word1, word2, word3, …word'1, …, word'k]

Gauss_1

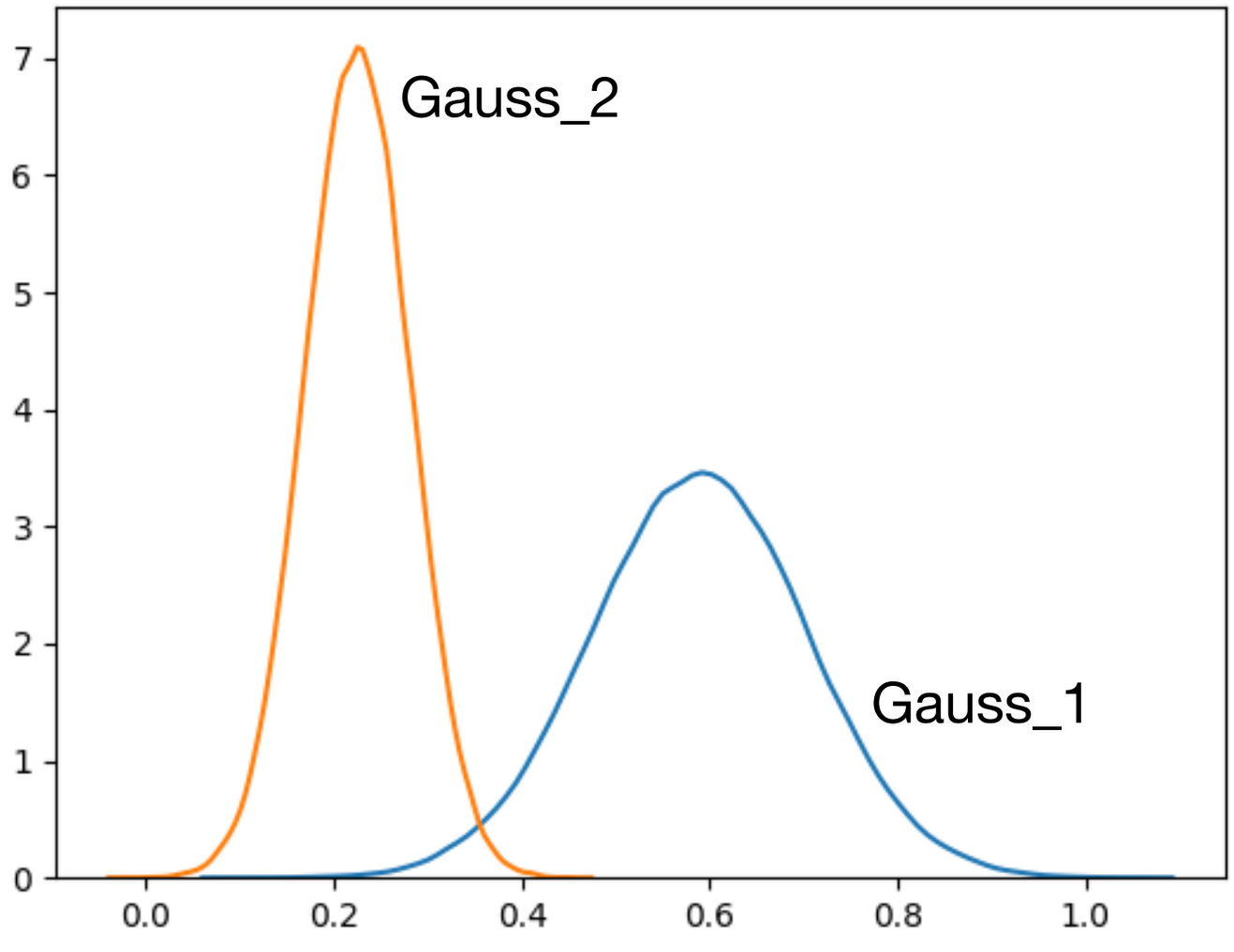
↓

Gauss_mix = (P_innocent)*Gauss_1
+ (P_undercover)*Gauss_2

↑

Gauss_2

[word4, word'1, …, word'j]



- HMM - sampling - e.g. [1 AI agent, 4 human players]

[own_item, word1, word2, word3] ⟶ [own_item, word1, word2, word3, …word'1, …, word'k]

(other k words similar to first 4 words)

[word4] ⟶ [word4, word'1, …, word'j]

(other j words similar to the first word)

[own_item, word1, word2, word3, …word'1, …, word'k]

Gauss_1

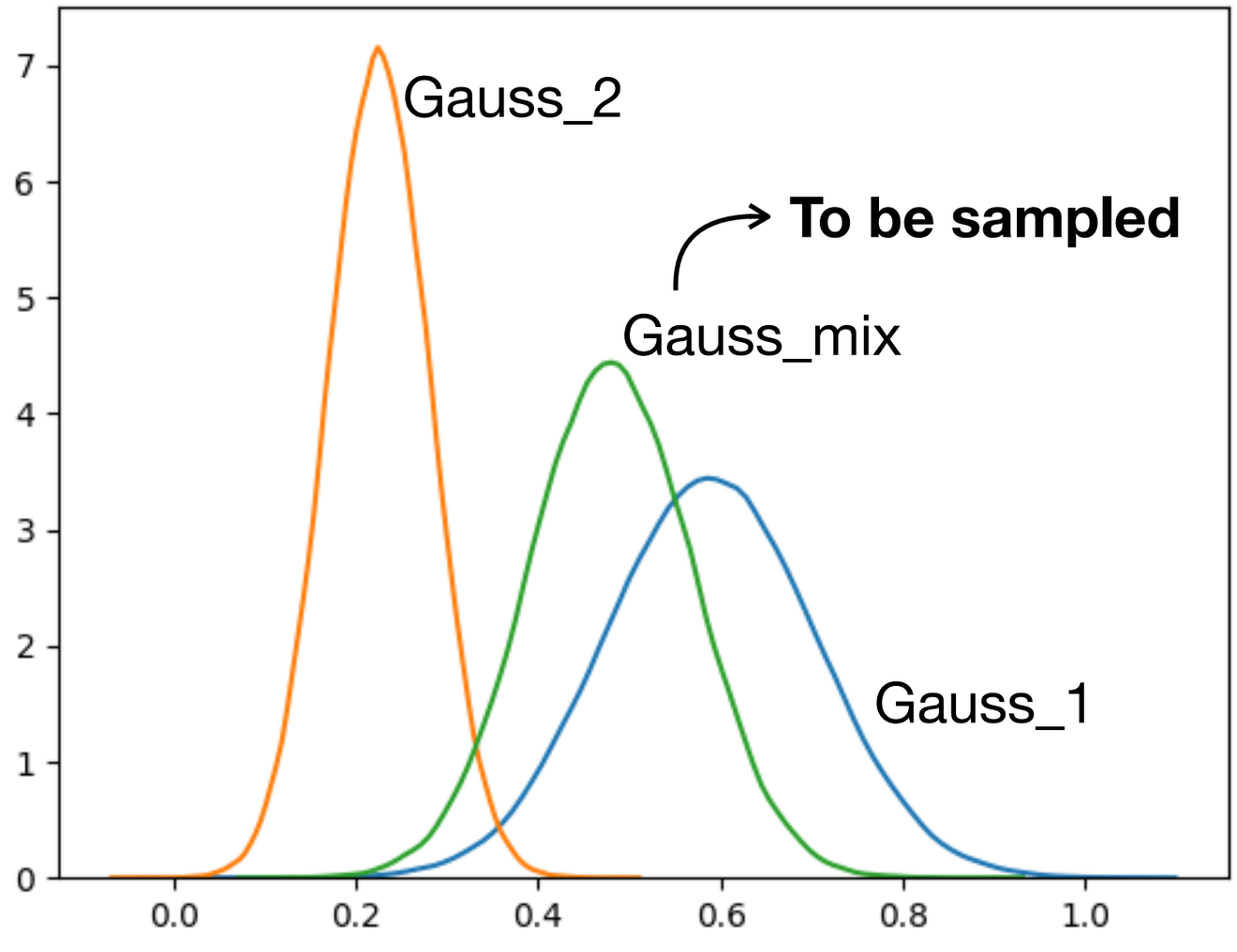
↓

Gauss_mix = (1-P_1)*Gauss_1 + (P_1)*Gauss_2

↑

Gauss_2

[word4, word'1, …, word'j]



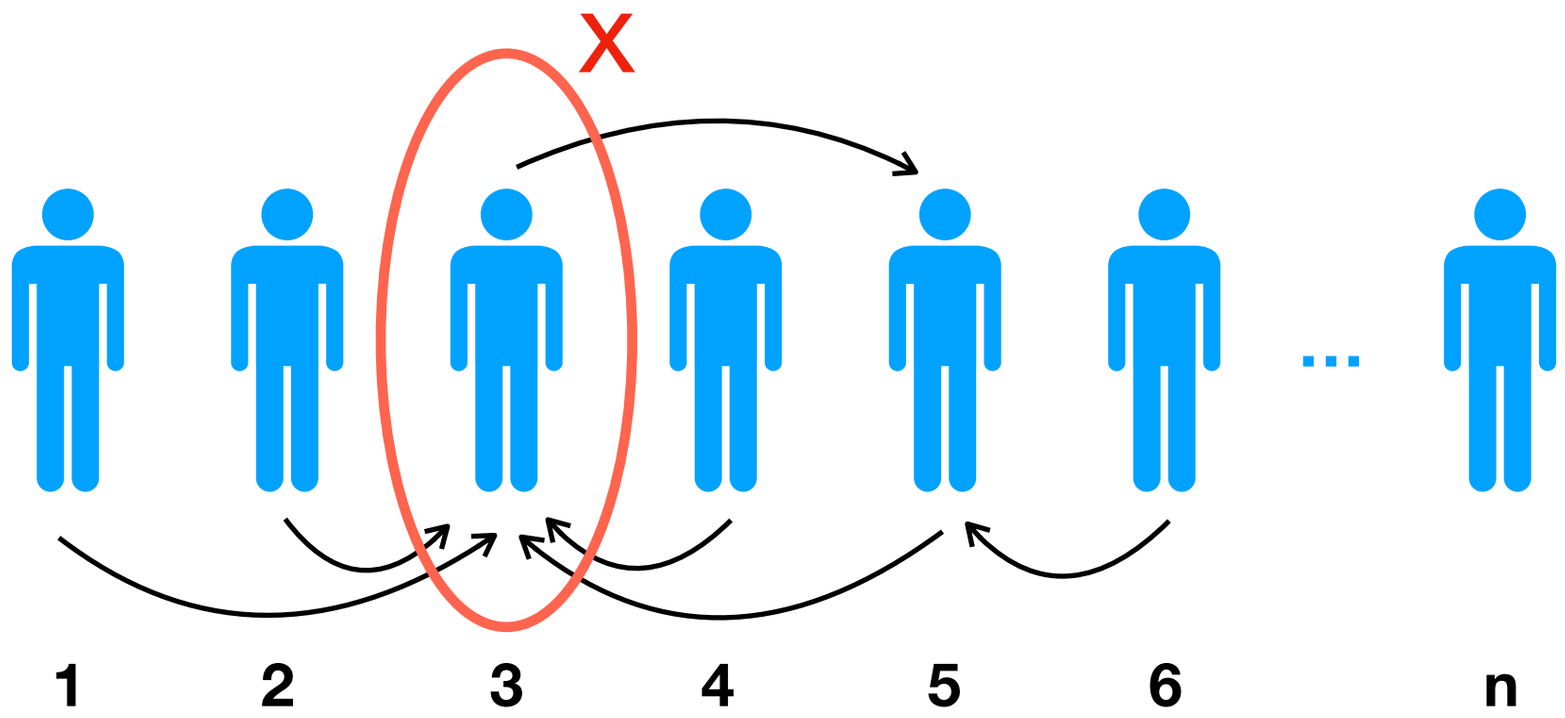
- HMM - state_transition

For agent_i:

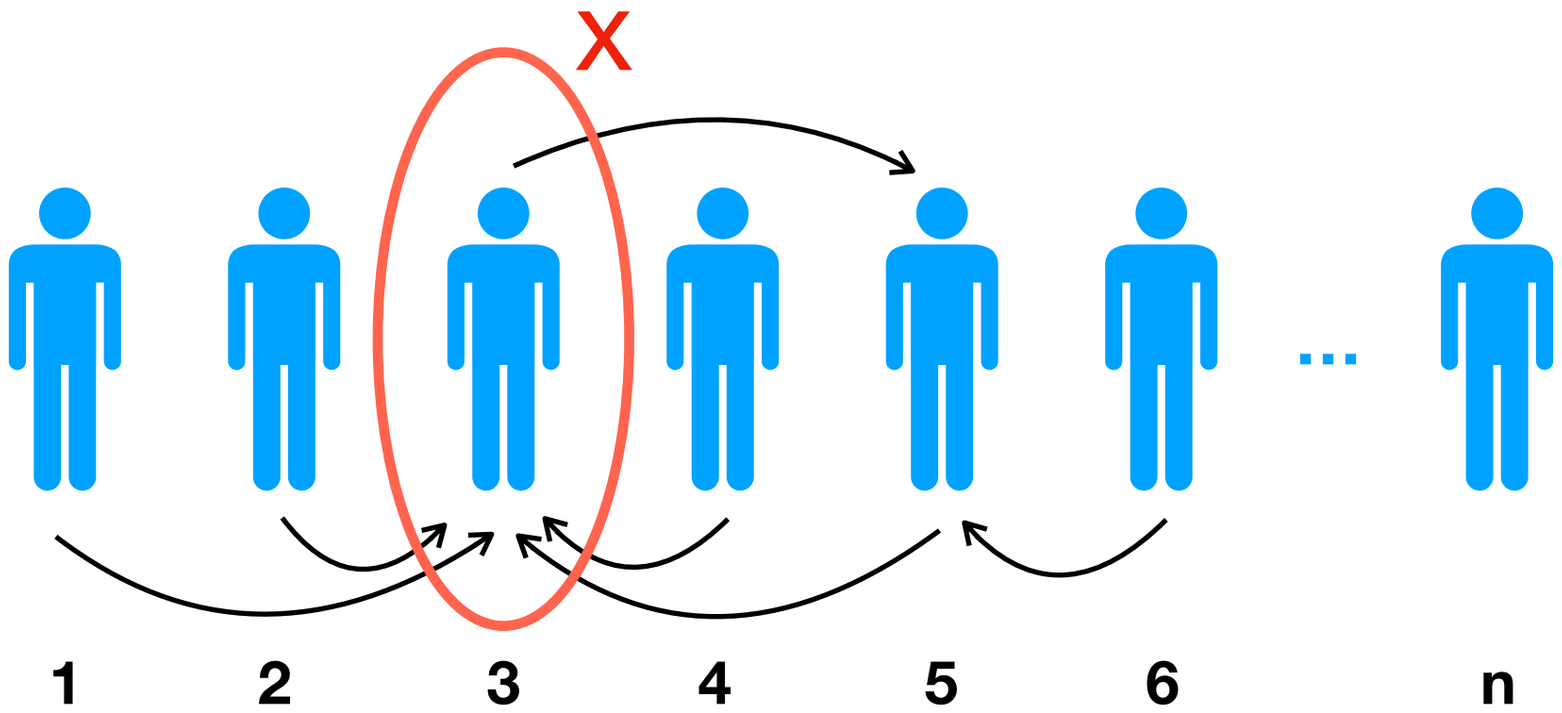
$[p_{i1}, p_{i2}, ..., p_{in}]$

$\max\{p_{i1}, p_{i2}, ..., p_{in}\} = p_{ij}$

vote for player_j



- HMM - state_transition



For agent_i:

$[p_{i1}, p_{i2}, p_{i3}, p_{i4}, \ldots, p_{in}]$

$[p_{i1}, p_{i2}, p_{i4}, \ldots, p_{in}]$

$[p'_{i1}, p'_{i2}, p'_{i4}, \ldots, p'_{in}]$

Normalise

Demo

{Mr.D} = dog (undercover)

Player {2,3,4,5} = cat (innocent)

```
*******Game Started*******
Mr.D is thinking...
/home/cheeseburg/.local/lib/python3.5/site-pack
rsion of the second argument of issubdtype from
ture, it will be treated as `np.int64 == np.dty
  if np.issubdtype(vec.dtype, np.int):
Round 1
-------------------------------------
Mr.D says white
Player 2's turn to give a word: cute
Player 3's turn to give a word: lovely
Player 4's turn to give a word: furry
Player 5's turn to give a word: animal
```

{Mr.D} = dog (undercover)

Player {2,3,4,5} = cat (innocent)

```
Mr.D votes player 2!
Player 2, vote for the Undercover(give a number between 1~5): 3
Player 3, vote for the Undercover(give a number between 1~5): 5
Player 4, vote for the Undercover(give a number between 1~5): 1
Player 5, vote for the Undercover(give a number between 1~5): 5
Player 5 out!
Round 2
```

{Mr.D} = dog (undercover)

Player {2,3,4,5} = cat (innocent)

```
Round 2
--------------------------------------
/home/cheeseburg/.local/lib/python3.5/site-packages
ning: arrays to stack must be passed as a "sequence
-sequence iterables such as generators is deprecate
 the future.
  vectors = vstack(self.word_vec(word, use_norm=Tru
Mr.D says paws
Player 2's turn to give a word: small
Player 3's turn to give a word: soft
Player 4's turn to give a word: flexible
```

{Mr.D} = dog (undercover)

Player {2,3,4,5} = cat (innocent)

```
Mr.D votes player 4!
Player 2, vote for the Undercover(give a number between 1~5): 4
Player 3, vote for the Undercover(give a number between 1~5): 4
Player 4, vote for the Undercover(give a number between 1~5): 1
Player 4 out!
```

{Mr.D} = dog (undercover)

Player {2,3,4,5} = cat (innocent)

```
Round 3
--------------------------------------------------
Mr.D says puppy
Player 2's turn to give a word: friendly
Player 3's turn to give a word: pretty
```

```
Mr.D votes player 2!
Player 2, vote for the Undercover(give a number between 1~5): 1
Player 3, vote for the Undercover(give a number between 1~5): 2
Player 2 out!
Undercover wins!
Mr.D wins! He is undercover, his word is dog!
```

Thanks