



# Research plan

Lab meeting

---

Andrea Pierré


May 20, 2024

Brown University

# Outline

1. Context 

2. Experiments & expected results  

3. Roadmap 

# Outline

1. Context 

2. Experiments & expected results  

3. Roadmap 

# Why modeling?

- Posit: You understand a system if you can simulate it  
*What I cannot create, I do not understand.*  
–Richard Feynman
- If you have a good enough model you may uncover mechanisms that explain a phenomena
  - Without a model → you're limited to describe the how
  - With a model → you may be able to explain the why
- Test hypothesis
- Abstraction of the system: makes you think of the parameters/inputs/outputs
- Find out what is needed to reproduce experimental results, what explains those results

# Why modeling?

- Posit: You understand a system if you can simulate it  
*What I cannot create, I do not understand.*  
–Richard Feynman
- If you have a good enough model you may uncover mechanisms that explain a phenomena
  - Without a model → you're limited to describe the how
  - With a model → you may be able to explain the why
- Test hypothesis
- Abstraction of the system: makes you think of the parameters/inputs/outputs
- Find out what is needed to reproduce experimental results, what explains those results

# Why modeling?

- Posit: You understand a system if you can simulate it  
*What I cannot create, I do not understand.*  
–Richard Feynman
- If you have a good enough model you may uncover mechanisms that explain a phenomena
  - Without a model → you're limited to describe the **how**
  - With a model → you may be able to explain the **why**
- Test hypothesis
- Abstraction of the system: makes you think of the parameters/inputs/outputs
- Find out what is needed to reproduce experimental results, what explains those results

# Why modeling?

- Posit: You understand a system if you can simulate it  
*What I cannot create, I do not understand.*  
–Richard Feynman
- If you have a good enough model you may uncover mechanisms that explain a phenomena
  - Without a model → you're limited to describe the **how**
    - With a model → you may be able to explain the **why**
- Test hypothesis
- Abstraction of the system: makes you think of the parameters/inputs/outputs
- Find out what is needed to reproduce experimental results, what explains those results

# Why modeling?

- Posit: You understand a system if you can simulate it  
*What I cannot create, I do not understand.*  
–Richard Feynman
- If you have a good enough model you may uncover mechanisms that explain a phenomena
  - Without a model → you're limited to describe the **how**
  - With a model → you may be able to explain the **why**
- Test hypothesis
- Abstraction of the system: makes you think of the parameters/inputs/outputs
- Find out what is needed to reproduce experimental results, what explains those results



# Why modeling?

- Posit: You understand a system if you can simulate it  
*What I cannot create, I do not understand.*  
–Richard Feynman
- If you have a good enough model you may uncover mechanisms that explain a phenomena
  - Without a model → you're limited to describe the **how**
  - With a model → you may be able to explain the **why**
- Test hypothesis
- Abstraction of the system: makes you think of the parameters/inputs/outputs
- Find out what is needed to reproduce experimental results, what explains those results

# Why modeling?

- Posit: You understand a system if you can simulate it  
*What I cannot create, I do not understand.*  
–Richard Feynman
- If you have a good enough model you may uncover mechanisms that explain a phenomena
  - Without a model → you're limited to describe the **how**
  - With a model → you may be able to explain the **why**
- Test hypothesis
- Abstraction of the system: makes you think of the parameters/inputs/outputs
- Find out what is needed to reproduce experimental results, what explains those results

# Why modeling?

- Posit: You understand a system if you can simulate it  
*What I cannot create, I do not understand.*  
–Richard Feynman
- If you have a good enough model you may uncover mechanisms that explain a phenomena
  - Without a model → you're limited to describe the **how**
  - With a model → you may be able to explain the **why**
- Test hypothesis
- Abstraction of the system: makes you think of the parameters/inputs/outputs
- Find out what is needed to reproduce experimental results, what explains those results

# What do we want to know?

- Understand what the network learns → What **function** does it learn?
- How the constraints of the task affect learning & the representations learned?
- Does the network learn something related to the real neurons? (million \$\$\$ question)

# What do we want to know?

- Understand what the network learns → What **function** does it learn?
- How the constraints of the task affect learning & the representations learned?
- Does the network learn something related to the real neurons? (million \$\$\$ question)

# What do we want to know?

- Understand what the network learns → What **function** does it learn?
- How the constraints of the task affect learning & the representations learned?
- Does the network learn something related to the real neurons? (million \$\$\$ question)

# Compositional hypothesis

- Does the network learn a generalizable policy?
  - How to test it?
    - From indexed locations (current) to coordinate system
    - Merged actions space

# Compositional hypothesis

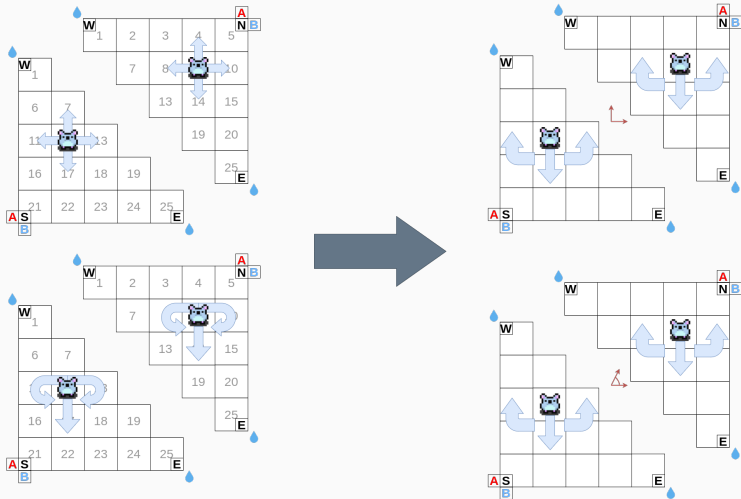
- Does the network learn a generalizable policy?
  - How to test it?
    - From indexed locations (current) to coordinate system
    - Merged actions space



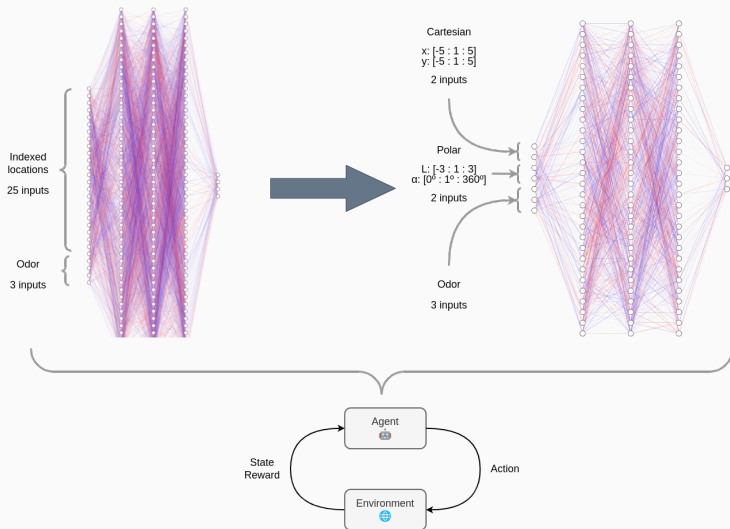
# Compositional hypothesis

- Does the network learn a generalizable policy?
  - How to test it?
    - From indexed locations (current) to coordinate system
    - Merged actions space

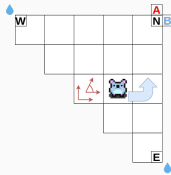
# Compositional hypothesis



# Implementation

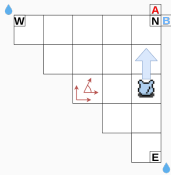


# Example episode



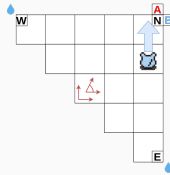
	x	y	L	α	odor
State	1	0	1	0	0

Step	Reward
0	0



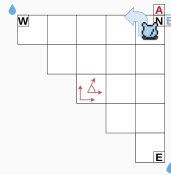
	x	y	L	a	odor
State	2	0	2	0	0

Step	Reward
1	0



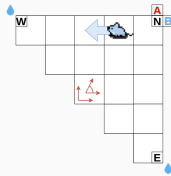
	x	y	L	a	odor
State	2	1	2.23	26.6	0

Step	Reward
2	0



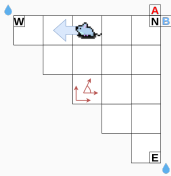
	x	y	L	a	odor
State	2	2	2.83	45	1

Step	Reward
3	0



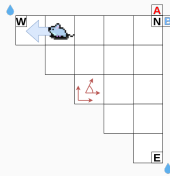
	x	y	L	a	odor
State	1	2	2.23	63.4	1

Step	Reward
4	0



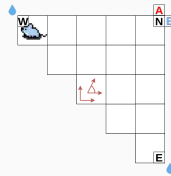
	x	y	L	$\alpha$	odor
State	0	2	2	90	1

Step	Reward
5	0



	x	y	L	a	odor
State	-1	2	2.23	-63.4	1

Step	Reward
6	0



	x	y	L	a	odor
State	-2	2	2.83	-45	1

Step	Reward
7	1

# Outline

1. Context 

2. Experiments & expected results  

3. Roadmap 

# 1) How training impacts the representations learned?

- Feed both coordinates information (Cartesian & polar) to the input layer (+ merge actions spaces in a common one)
- Train on left/right task → we expect the weights are close to zero on Cartesian representation?
- Train on east/west task → we expect the weights are close to zero on polar representation?

# 1) How training impacts the representations learned?

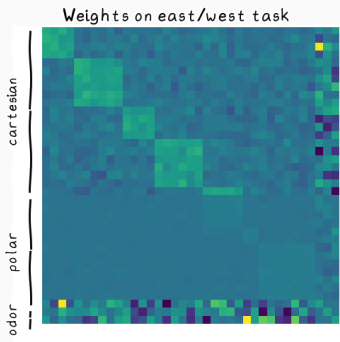
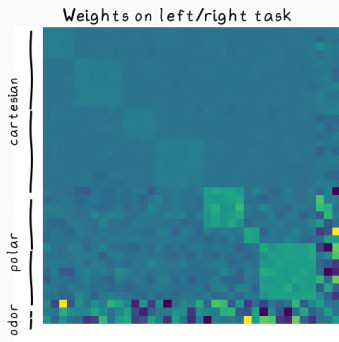
- Feed both coordinates information (Cartesian & polar) to the input layer (+ merge actions spaces in a common one)
- Train on left/right task → we expect the weights are close to zero on Cartesian representation?
- Train on east/west task → we expect the weights are close to zero on polar representation?

# 1) How training impacts the representations learned?

- Feed both coordinates information (Cartesian & polar) to the input layer (+ merge actions spaces in a common one)
- Train on left/right task → we expect the weights are close to zero on Cartesian representation?
- Train on east/west task → we expect the weights are close to zero on polar representation?



# 1) How training impacts the representations learned?



## 2) Does the network learn a coordinate system?

1. After training, move the population of agents in a translated coordinate system → we expect the population of agents to be able to solve the task with zero shot learning
2. Train with both coordinates information (Cartesian & polar), after training feed incorrect polar angles
  - On the left/right task → we expect the population of agents still solves the task consistently
  - On the east/west task → we expect the network won't converge to a stable policy (i.e all the agents don't solve the task consistently)

## 2) Does the network learn a coordinate system?

1. After training, move the population of agents in a translated coordinate system → we expect the population of agents to be able to solve the task with zero shot learning
2. Train with both coordinates information (Cartesian & polar), after training feed incorrect polar angles
  - On the left/right task → we expect the population of agents still solves the task consistently
  - On the east/west task → we expect the network won't converge to a stable policy (i.e all the agents don't solve the task consistently)

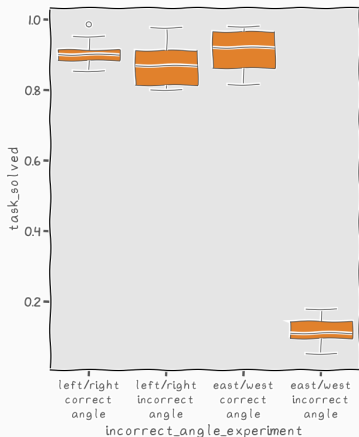
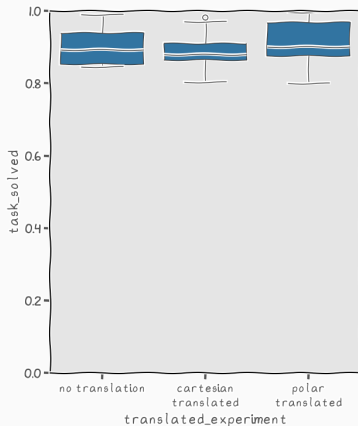
## 2) Does the network learn a coordinate system?

1. After training, move the population of agents in a translated coordinate system → we expect the population of agents to be able to solve the task with zero shot learning
2. Train with both coordinates information (Cartesian & polar), after training feed incorrect polar angles
  - On the left/right task → we expect the population of agents still solves the task consistently
  - On the east/west task → we expect the network won't converge to a stable policy (i.e all the agents don't solve the task consistently)

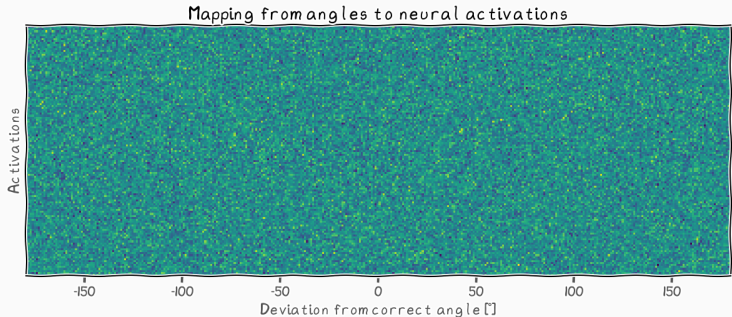
## 2) Does the network learn a coordinate system?

1. After training, move the population of agents in a translated coordinate system → we expect the population of agents to be able to solve the task with zero shot learning
2. Train with both coordinates information (Cartesian & polar), after training feed incorrect polar angles
  - On the left/right task → we expect the population of agents still solves the task consistently
  - On the east/west task → we expect the network won't converge to a stable policy (i.e all the agents don't solve the task consistently)

## 2) Does the network learn a coordinate system?




## 2) Does the network learn a coordinate system?



# Outline

1. Context 

2. Experiments & expected results  

3. Roadmap 



# Experiments table

Experiment	Agents	Hours estimation
left/right Cartesian coordinates from center arena	30	6
left/right Cartesian coordinates from 3 ports	30	6
east/west polar coordinates from center arena	30	6
east/west polar coordinates from 3 ports	30	6
No translation	30	6
Cartesian translated	30	6
Polar translated	30	6
left/right correct angle	30	6
left/right incorrect angle	30	6
east/west correct angle	30	6
east/west incorrect angle	30	6
Total	330	66

# Milestones/how to get there

## 1. Rewrite the environment(s) ★★☆☆

1.1 Code logic for new environment [~1 week]

1.2 Check everything works as expected (unit testing)  
[~1 week]

1.3 Bugs? [~1 week]

1.4 Baseline training on new environment (convergence,  
hyperparameter tweaking, etc.) ★★★⚠  
[1 week – 1 month]

## 2. Experiments

2.1 Task code ★☆☆ [~1 week]

2.2 Training ★★☆☆⚠ [~2 week]

2.3 Analysis code ★☆☆ [~2 week]

# Milestones/how to get there

## 1. Rewrite the environment(s) ★★☆☆

1.1 Code logic for new environment [~1 week]

1.2 Check everything works as expected (unit testing)  
[~1 week]

1.3 Bugs? [~1 week]

1.4 Baseline training on new environment (convergence,  
hyperparameter tweaking, etc.) ★★★⚠  
[1 week – 1 month]

## 2. Experiments

2.1 Task code ★☆☆ [~1 week]

2.2 Training ★★☆☆⚠ [~2 week]

2.3 Analysis code ★☆☆ [~2 week]

# Milestones/how to get there

## 1. Rewrite the environment(s) ★★☆☆

1.1 Code logic for new environment [~1 week]

1.2 Check everything works as expected (unit testing)  
[~1 week]

1.3 Bugs? [~1 week]

1.4 Baseline training on new environment (convergence,  
hyperparameter tweaking, etc.) ★★★⚠  
[1 week – 1 month]

## 2. Experiments

2.1 Task code ★☆☆ [~1 week]

2.2 Training ★★☆☆⚠ [~2 week]

2.3 Analysis code ★☆☆ [~2 week]

# Milestones/how to get there

## 1. Rewrite the environment(s) ★★☆☆

1.1 Code logic for new environment [~1 week]

1.2 Check everything works as expected (unit testing)  
[~1 week]

1.3 Bugs? [~1 week]

1.4 Baseline training on new environment (convergence,  
hyperparameter tweaking, etc.) ★★★⚠  
[1 week – 1 month]

## 2. Experiments

2.1 Task code ★☆☆ [~1 week]

2.2 Training ★★☆☆⚠ [~2 week]

2.3 Analysis code ★☆☆ [~2 week]

# Milestones/how to get there

## 1. Rewrite the environment(s) ★★★☆

1.1 Code logic for new environment [~1 week]

1.2 Check everything works as expected (unit testing)  
[~1 week]

1.3 Bugs? [~1 week]

1.4 Baseline training on new environment (convergence, hyperparameter tweaking, etc.) ★★★⚠  
[1 week – 1 month]

## 2. Experiments

2.1 Task code ★☆☆ [~1 week]

2.2 Training ★★☆☆ [~2 week]

2.3 Analysis code ★☆☆ [~2 week]

# Milestones/how to get there

## 1. Rewrite the environment(s) ★★☆☆

1.1 Code logic for new environment [~1 week]

1.2 Check everything works as expected (unit testing)  
[~1 week]

1.3 Bugs? [~1 week]

1.4 Baseline training on new environment (convergence,  
hyperparameter tweaking, etc.) ★★★⚠  
[1 week – 1 month]

## 2. Experiments

2.1 Task code ★☆☆ [~1 week]

2.2 Training ★★☆☆⚠ [~2 week]

2.3 Analysis code ★★☆☆ [~2 week]

# Milestones/how to get there

## 1. Rewrite the environment(s) ★★☆☆

1.1 Code logic for new environment [~1 week]

1.2 Check everything works as expected (unit testing)  
[~1 week]

1.3 Bugs? [~1 week]

1.4 Baseline training on new environment (convergence,  
hyperparameter tweaking, etc.) ★★★⚠  
[1 week – 1 month]

## 2. Experiments

2.1 Task code ★☆☆ [~1 week]

2.2 Training ★★☆☆⚠ [~2 week]

2.3 Analysis code ★★☆☆ [~2 week]



# Milestones/how to get there

## 1. Rewrite the environment(s) ★★☆☆

1.1 Code logic for new environment [~1 week]

1.2 Check everything works as expected (unit testing)  
[~1 week]

1.3 Bugs? [~1 week]

1.4 Baseline training on new environment (convergence,  
hyperparameter tweaking, etc.) ★★★⚠  
[1 week – 1 month]

## 2. Experiments

2.1 Task code ★☆☆ [~1 week]

2.2 Training ★★★⚠ [~2 week]

2.3 Analysis code ★★☆☆ [~2 week]

# Milestones/how to get there

## 1. Rewrite the environment(s) ★★☆☆

1.1 Code logic for new environment [~1 week]

1.2 Check everything works as expected (unit testing)  
[~1 week]

1.3 Bugs? [~1 week]

1.4 Baseline training on new environment (convergence,  
hyperparameter tweaking, etc.) ★★★⚠  
[1 week – 1 month]

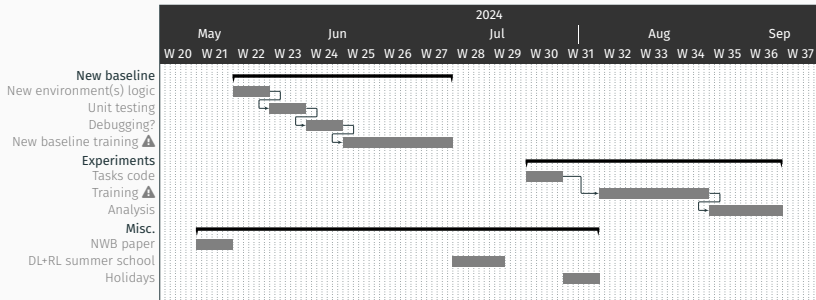
## 2. Experiments

2.1 Task code ★☆☆ [~1 week]

2.2 Training ★★★⚠ [~2 week]

2.3 Analysis code ★★★ [~2 week]

# Planning



Thanks!