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 $\rightarrow$ you're limited to describe the how
  - With a model $\rightarrow$ 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 $\rightarrow$ you're limited to describe the how
  - With a model $\rightarrow$ 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 $\rightarrow$ you're limited to describe the how
  - With a model $\rightarrow$ 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 $\rightarrow$ you're limited to describe the how
  - With a model $\rightarrow$ 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

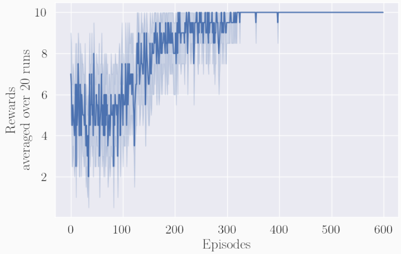
Why modeling?

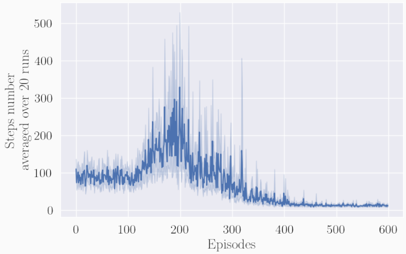
- Posit: You understand a system if you can simulate it *What I cannot create, I do not understand. –Richard Feynman*
- If you have a good enough model you may uncover mechanisms that explain a phenomena
  - Without a model $\rightarrow$ you're limited to describe the how
  - With a model $\rightarrow$ 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 $\rightarrow$ you're limited to describe the how
  - With a model $\rightarrow$ 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

Recap of previous episodes: it's converging! ☺





Why is it converging now?

- Lights cues in the state?
- Start training once replay buffer is full (5000 transitions) instead of when there are enough transitions for a batch (32 transitions)
- Soft update of the networks weights (instead of sharp transition)
- Huber loss instead of mean squared error $\rightarrow$ should be less sensible to outliers
- Remove ReLU on output layer!

Why is it converging now?

- Lights cues in the state?
- Start training once replay buffer is full (5000 transitions) instead of when there are enough transitions for a batch (32 transitions)
- Soft update of the networks weights (instead of sharp transition)
- Huber loss instead of mean squared error $\rightarrow$ should be less sensible to outliers
- Remove ReLU on output layer!

Why is it converging now?

- Lights cues in the state?
- Start training once replay buffer is full (5000 transitions) instead of when there are enough transitions for a batch (32 transitions)
- Soft update of the networks weights (instead of sharp transition)
- Huber loss instead of mean squared error $\rightarrow$ should be less sensible to outliers
- Remove ReLU on output layer!

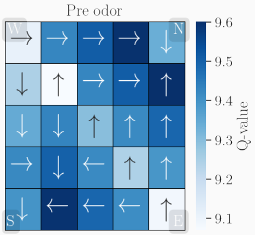
Why is it converging now?

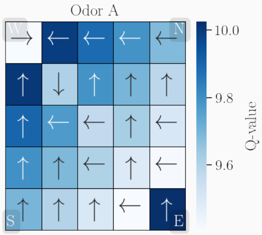
- Lights cues in the state?
- Start training once replay buffer is full (5000 transitions) instead of when there are enough transitions for a batch (32 transitions)
- Soft update of the networks weights (instead of sharp transition)
- Huber loss instead of mean squared error $\rightarrow$ should be less sensible to outliers
- Remove ReLU on output layer!

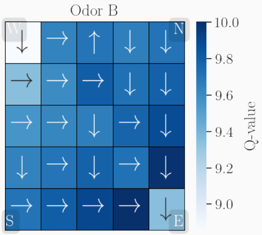
Why is it converging now?

- Lights cues in the state?
- Start training once replay buffer is full (5000 transitions) instead of when there are enough transitions for a batch (32 transitions)
- Soft update of the networks weights (instead of sharp transition)
- Huber loss instead of mean squared error $\rightarrow$ should be less sensible to outliers
- Remove ReLU on output layer!

What did it learn?







What do we want to know?

- Understand what the network learns → What function does it learns?
- How the constrains 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 $\rightarrow$ What function does it learns?
- How the constrains 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 $\rightarrow$ What function does it learns?
- How the constrains 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?
  $\rightarrow$ How to test it?
  - From indexed locations (current) to coordinate system
  - Merged actions space

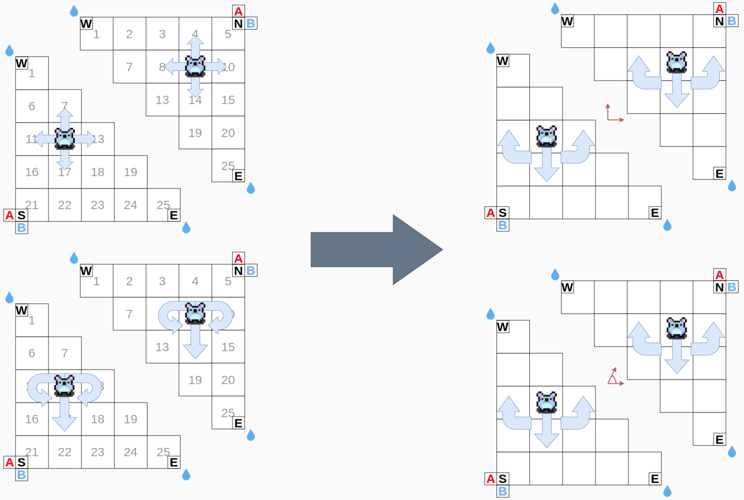
Compositional hypothesis

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

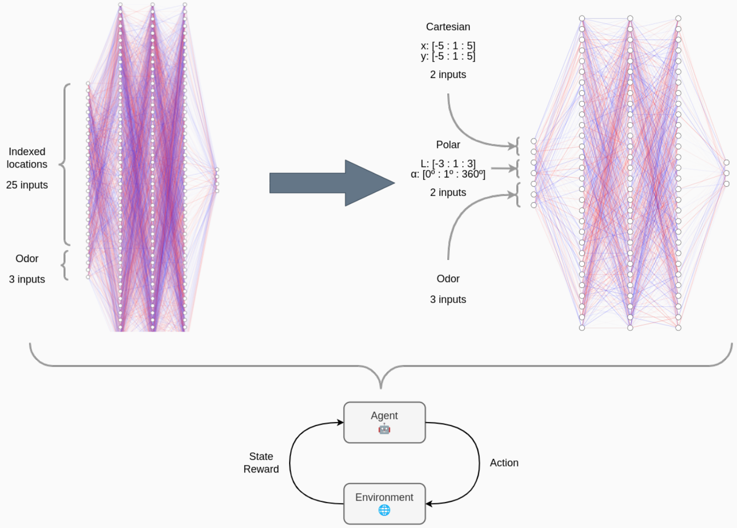
Compositional hypothesis

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

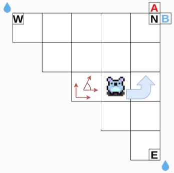
Compositional hypothesis



Implementation

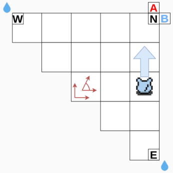


Example episode



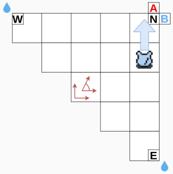
	x	y	L	α	odor
State	1	0	1	0	0

Step	Reward
0	0



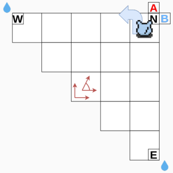
	x	y	L	α	odor
State	2	0	2	0	0

Step	Reward
1	0



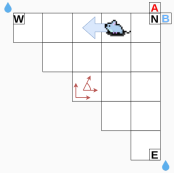
	x	y	L	α	odor
State	2	1	2.23	26.6	0

Step	Reward
2	0



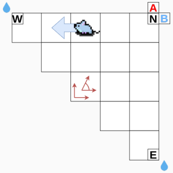
	x	y	L	α	odor
State	2	2	2.83	45	1

Step	Reward
3	0



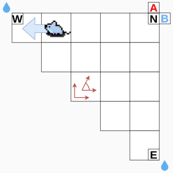
	x	y	L	α	odor
State	1	2	2.23	63.4	1

Step	Reward
4	0



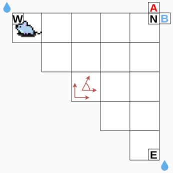
	x	y	L	α	odor
State	0	2	2	90	1

Step	Reward
5	0



	x	y	L	α	odor
State	-1	2	2.23	-63.4	1

Step	Reward
6	0



	x	y	L	α	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 $\rightarrow$ we expect the weights are close to zero on Cartesian representation?
- Train on east/west task $\rightarrow$ we expect the weights are close to zero on polar representation?

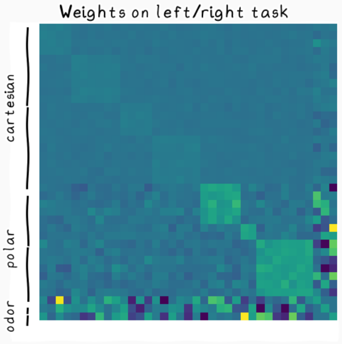
1) How training impacts the representations learned?

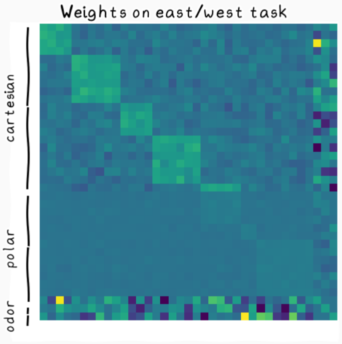
- Feed both coordinates information (Cartesian & polar) to the input layer (+ merge actions spaces in a common one)
- Train on left/right task $\rightarrow$ we expect the weights are close to zero on Cartesian representation?
- Train on east/west task $\rightarrow$ we expect the weights are close to 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 $\rightarrow$ we expect the weights are close to zero on Cartesian representation?
- Train on east/west task $\rightarrow$ 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 $\rightarrow$ 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 $\rightarrow$ we expect the population of agents still solves the task consistently
   - On the east/west task $\rightarrow$ 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 $\rightarrow$ 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 $\rightarrow$ we expect the population of agents still solves the task consistently
    - On the east/west task $\rightarrow$ we expect the network won't converge to a stable policy (i.e all the agents don't solve the task consistently)

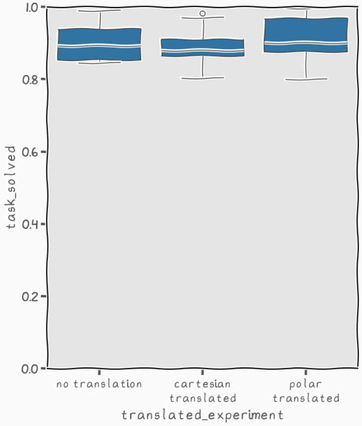
2) Does the network learn a coordinate system?

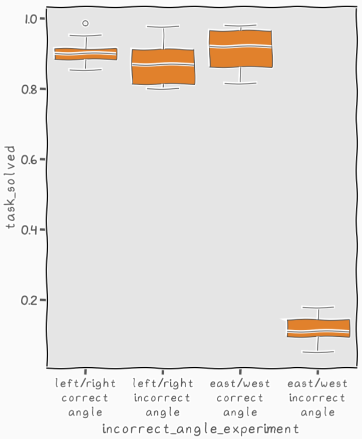
1. After training, move the population of agents in a translated coordinate system $\rightarrow$ we expect the population of agents to be able to solve the task with zero shot learning
2. Train with both coordinates information (Cartesian & polar), after training feed incorrect polar angles
    - On the left/right task $\rightarrow$ we expect the population of agents still solves the task consistently
    - On the east/west task $\rightarrow$ 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?

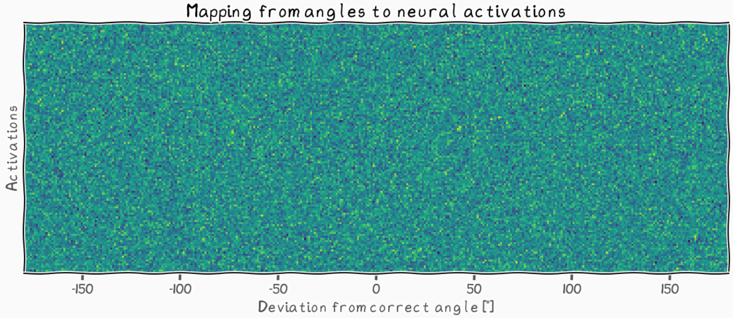
1. After training, move the population of agents in a translated coordinate system $\rightarrow$ we expect the population of agents to be able to solve the task with zero shot learning
2. Train with both coordinates information (Cartesian & polar), after training feed incorrect polar angles
    - On the left/right task $\rightarrow$ we expect the population of agents still solves the task consistently
    - On the east/west task $\rightarrow$ 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	Training estimation [hours]
left/right Cartesian coordinates from center arena	20	6
left/right Cartesian coordinates from 3 ports	20	6
east/west polar coordinates from center arena	20	6
east/west polar coordinates from 3 ports	20	6
No translation	20	6
Cartesian translated	20	6
Polar translated	20	6
left/right correct angle	20	6
left/right incorrect angle	20	6
east/west correct angle	20	6
east/west incorrect angle	20	6
Total	220	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]

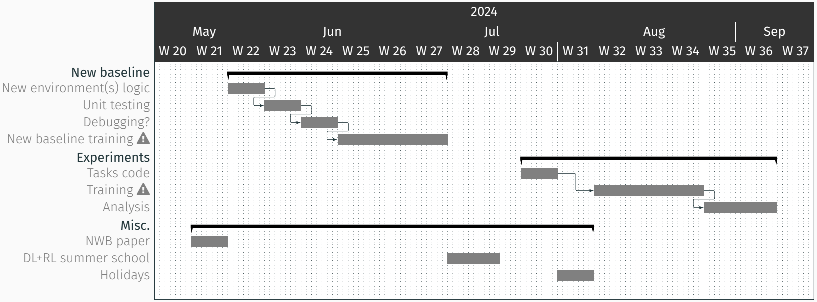
Milestones/how to get there

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


2024
May
Jun
Jul
Aug
Sep
W 20
W 21
W 22
W 23
W 24
W 25
W 26
W 27
W 28
W 29
W 30
W 31
W 32
W 33
W 34
W 35
W 36
W 37
New baseline
New environment(s) logic
Unit testing
Debugging?
New baseline training ⚠
Experiments
Tasks code
Training ⚠
Analysis
Misc.
NWB paper
DL+RL summer school
Holidays

Thanks!