Neural Logic Reinforcement Learning

Team: Pikachu

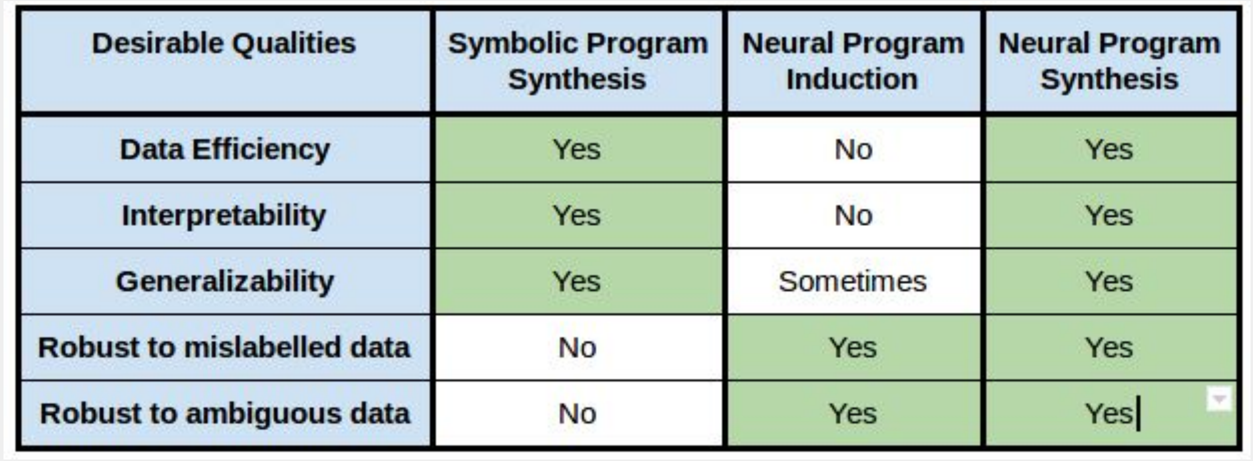
Aishwarya S. (20171046), Kritika P. (20161039), Pooja S. (20171403)

INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY
HYDERABAD

Introduction

- Policies learnt by Deep Reinforcement Learning algorithms are not interpretable or generalizable to similar environments.
- Hence, the performance of most deep reinforcement learning (DRL) algorithms is largely affected even by minor modifications of the training environment.

“Learn programs (procedures) from examples using optimization, given program structure”

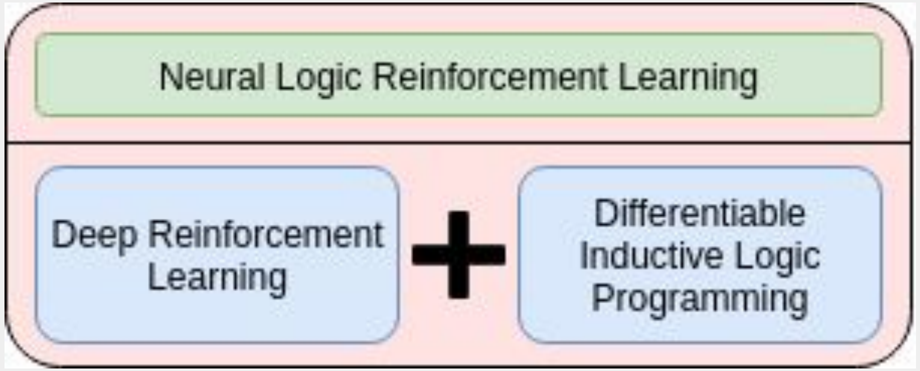


Desirable Qualities	Symbolic Program Synthesis	Neural Program Induction	Neural Program Synthesis
Data Efficiency	Yes	No	Yes
Interpretability	Yes	No	Yes
Generalizability	Yes	Sometimes	Yes
Robust to mislabelled data	No	Yes	Yes
Robust to ambiguous data	No	Yes	Yes

INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY
HYDERABAD

Solution Approach

- A new RL method, Neural Logic Reinforcement Learning (NLRL) represents a neural program synthesis method to learn the policies in reinforcement learning by first-order logic, thus addressing the challenges faced by DRL algorithms.
- Notable features of Neural Logic Reinforcement Learning:
  - Compatible with policy gradients
  - Much more interpretable and generalizable


Neural Logic Reinforcement Learning
Deep Reinforcement Learning
Differentiable Inductive Logic Programming

First Order Logic Programming

No	Element	Symbol	Definition
1.	Variable	V	Variable V coming from a fixed countably infinite set of symbols, representing objects in the domain
2.	Constant	C	Constant C - Represent objects in the domain
3.	Term	T	Term T := V | C
4.	Predicate	P	$P := \{ DC_1\ DC_2 \ldots DC_n \} \mid \{ \gamma_1 \gamma_2 \ldots \gamma_n \}$ Maps one or more objects onto truth values.
5.	Atom	α	Atom $\alpha := P (T_1\ T_2 \ldots T_n)$, where P is an n-ary predicate,
6.	Atoms Set	**A**	Set of all atoms: **A**
7.	Definite Clause	DC	$DC := \alpha \leftarrow \{ \alpha_1 \alpha_2 \ldots \alpha_n \}$, where $\alpha_1 \land \alpha_2 \land \ldots \land \alpha_n \Rightarrow \alpha$ (conjunction) Head Atom := α, Body: = $\alpha_1 \alpha_2 \ldots \alpha_n$, $n > 0$

INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY
HYDERABAD

ILP: Inductive Logic Programming

ILP Problem

Given target predicate $\boldsymbol{P_T}$, Inductive Logic Programming (ILP) problem is defined as Tuple $\boldsymbol{< \mathscr{L}, B, P, N >}$ of ground atoms, where:

- $\mathscr{L}$ is a language frame
- ***B***: Set of background assumptions
- ***P***: Set of positive instances
- ***N***: Set of negative instances taken outside the extension of the target predicate

Aim

To construct a logic program that explains the set of positive instances ***P*** and rejects the set of negative instances ***N***.

positive examples + negative examples + background knowledge $\Rightarrow$ hypothesis

Solution

Find the set R of definite clauses such that:

- $B, R \vDash \gamma$, for every $\gamma \in P$
- $B, R \nvDash \gamma$, for every $\gamma \in N$

INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY
HYDERABAD

ILP: Forward Chaining

A set **R** of clauses is said to entail a ground atom ɣ if every model satisfying **R** also satisfies ɣ.
Represented as: **R** $\vDash$ ɣ
ɣ $\in$ cn∞(R)
(Immediate Consequence: $cn_R(X) = X \cup \{\gamma \mid \gamma \leftarrow \gamma_1 \ldots \gamma_m \in \text{ground}(R), \wedge \gamma_i \in X\}$)
This is known as forward chaining (Modus Ponens).
It is a bottom-up approach of making a conclusion based on known facts or data, by starting from the initial state and reaches the goal state.
Data driven approach, data given.
Modus Ponens: whenever $p \rightarrow q$ is true and p is true, q must also be true.

P	Q	P → Q
T	T	T
T	F	F
F	T	T
F	F	T

INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY
HYDERABAD

ILP: Example

Example: Even Numbers

Given:

$$\mathcal{B} = \{zero(0), succ(0,1), succ(1,2), succ(2,3), ...\}$$

Positive and negative examples of even predicate are:

$$\begin{aligned} \mathcal{P} &= \{\text{even}(0), \text{even}(2), \text{even}(4), \text{even}(6), ...\} \\ \mathcal{N} &= \{\text{even}(1), \text{even}(3), \text{even}(5), \text{even}(7), ...\} \end{aligned}$$

Solution:

Set of Clauses R :=

$$\begin{aligned} even(X) &\leftarrow zero(X) \\ even(X) &\leftarrow even(Y), succ2(Y,X) \\ succ2(X,Y) &\leftarrow succ(X,Z), succ(Z,Y) \end{aligned}$$

∂ILP: DILP Architecture

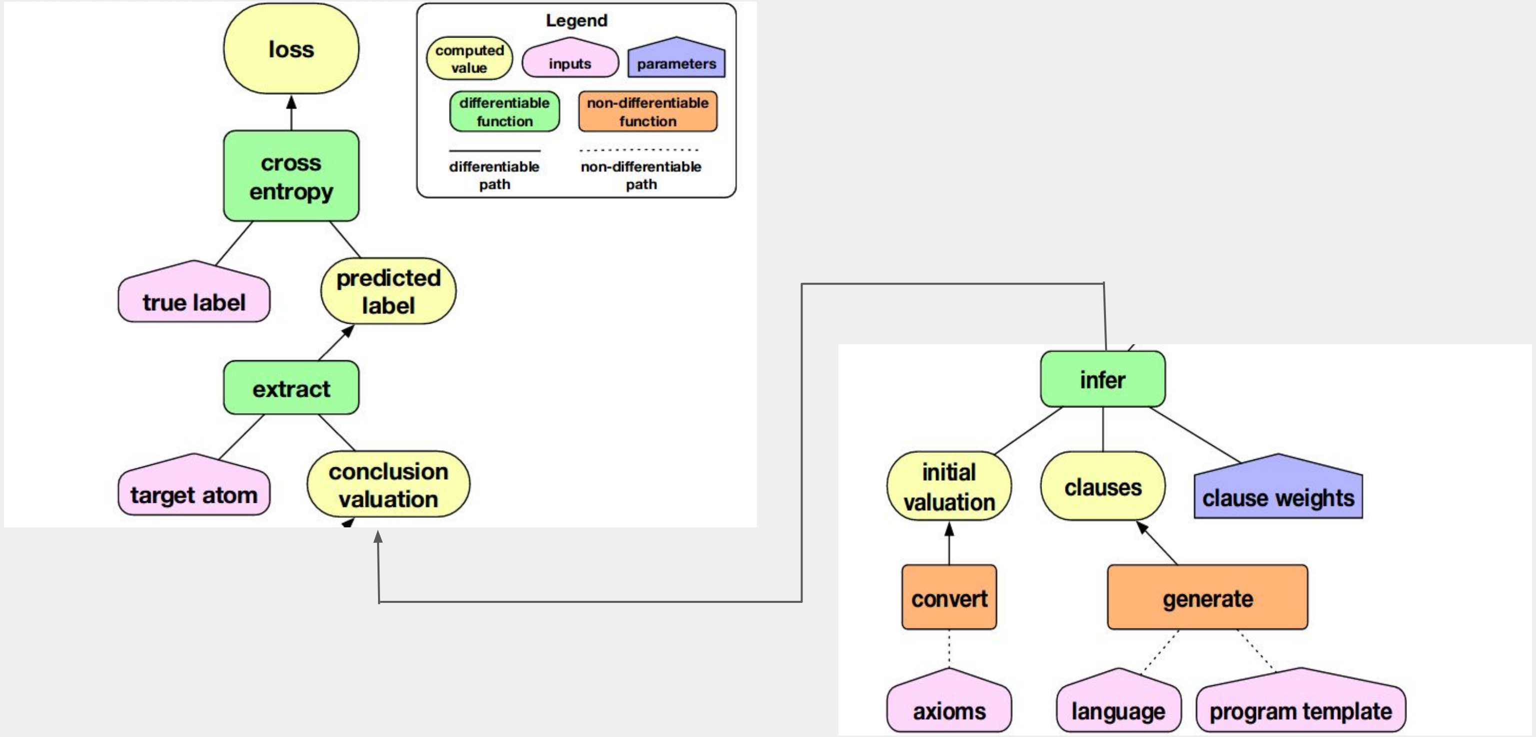
- ∂ILP is an attempt to combine ILP with differentiable programming.
- It uses forward chaining inference.
- For each predicate, ∂ILP generates a series of potential clauses combinations in advance based on rules templates.
- Rule Template:

  τ := (v, int) pair
  where, v = number of existentially quantified variables allowed in the clause.
  , int = tells whether intensional predicates are allowed or not in the clause

- Each clause has maximum 2 predicates in its body.
- We define a set of weights which give us the probability distribution over clauses.
- On combining clauses according to the weights we get the confidence value for a ground atom.

INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY
HYDERABAD

∂ILP: DILP Architecture (contd..)



INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY
HYDERABAD

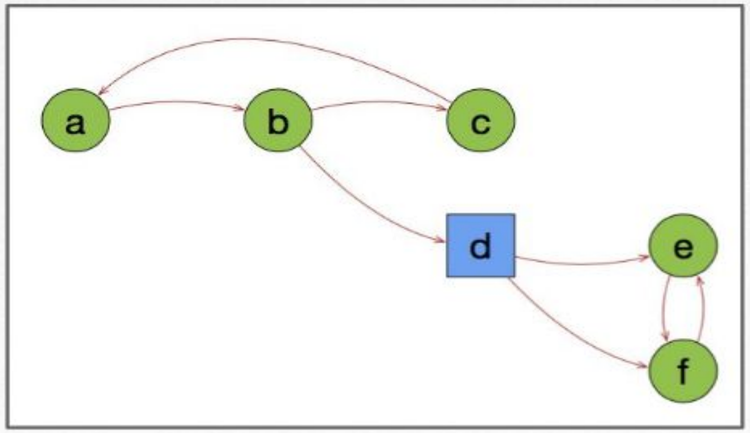
DILP: Forward Chaining Example

- A set **R** of clauses is said to entail a ground atom ɣ if every model satisfying **R** also satisfies ɣ. This is known as forward chaining.
- The central idea behind the differentiable implementation of inference is that each clause c induces a function
- $\mathcal{F}_C$: $[0,1]^n \rightarrow [0,1]^n$ on valuations.
- Applying c = p(X)←q(X), treated as a function $\mathcal{F}_C$, to valuations a_0 and a_1.
- Example:

G	**a_0**	**$\mathcal{F}_C(a_0)$**	**a_1**	**$\mathcal{F}_C(a_1)$**
p(a)	0.0	0.1	0.2	0.7
p(b)	0.0	0.3	0.9	0.4
q(a)	0.1	0.0	0.7	0.0
q(b)	0.3	0.0	0.4	0.0
⊥	0.0	0.0	0.0	0.0

Applications of ∂ILP in Different Domains

- Arithmetic
  - Example: Learning even or odd
- Relational
  - Example: Learning the greater than or lesser than function for natural numbers
- Group Theory
  - Example: Detecting Graph Cyclicity



```
cycle(X) ← pred(X, X).

pred(X, Y) ← edge(X, Y).

pred(X, Y) ← edge(X, Z), pred(Z, Y)
```

DRLM: Differentiable Recurrent Logic Machine

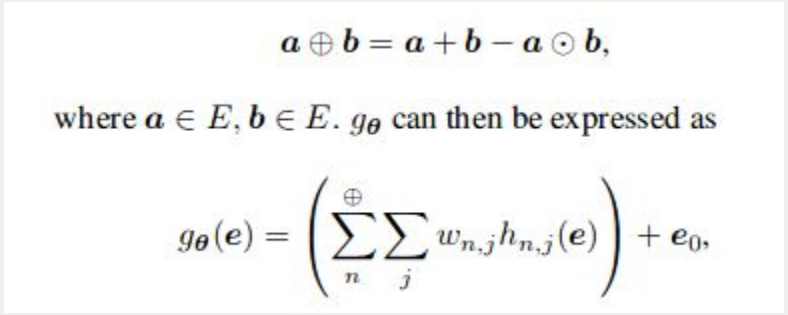
- Set of ground action atoms, **G** = { α_1 , α_2 , … , α_n }
- Number of ground action atoms, |**G**| = n
- Valuation vector (valuation of each atom α_i) **E** = $[0,1]^n$ = [e_1 e_2 . . . e_n]
- Value e_i : Confidence that ground atom α_i is **True**
- **DRLM** is a mapping f_θ : E → E which produces deductions of the facts e_0, using weights w associated with possible clauses.

$$f_{\boldsymbol{\theta}}^{t}(\boldsymbol{e}_0) = \begin{cases} g_{\boldsymbol{\theta}}(f_{\boldsymbol{\theta}}^{t-1}(\boldsymbol{e}_0)), & \text{if } t > 0. \\ \boldsymbol{e}_0, & \text{if } t = 0. \end{cases},$$

here, t is the deduction step, g_θ implements one step deduction of all possible clauses weighted by their confidence.

DRLM: Differentiable Recurrent Logic Machine

- The probabilistic sum is denoted as ⊕, and



$$\boldsymbol{a} \oplus \boldsymbol{b} = \boldsymbol{a} + \boldsymbol{b} - \boldsymbol{a} \odot \boldsymbol{b},$$

where $\boldsymbol{a} \in E, \boldsymbol{b} \in E$. $g_{\boldsymbol{\theta}}$ can then be expressed as

$$g_{\boldsymbol{\theta}}(\boldsymbol{e}) = \left(\sum_{n}^{\oplus} \sum_{j} w_{n,j} h_{n,j}(\boldsymbol{e}) \right) + \boldsymbol{e}_0,$$

here, $h_{n,j}(e)$ implements one-step deduction of the valuation vector e using jth possible definition of *n*th clause. The weights w_c is the vector of weights associated with predicate c.

- The predicates are more **flexible** in DRLM, in comparison to ∂ILP as the weights are directly associated with clauses and not weights of clauses.

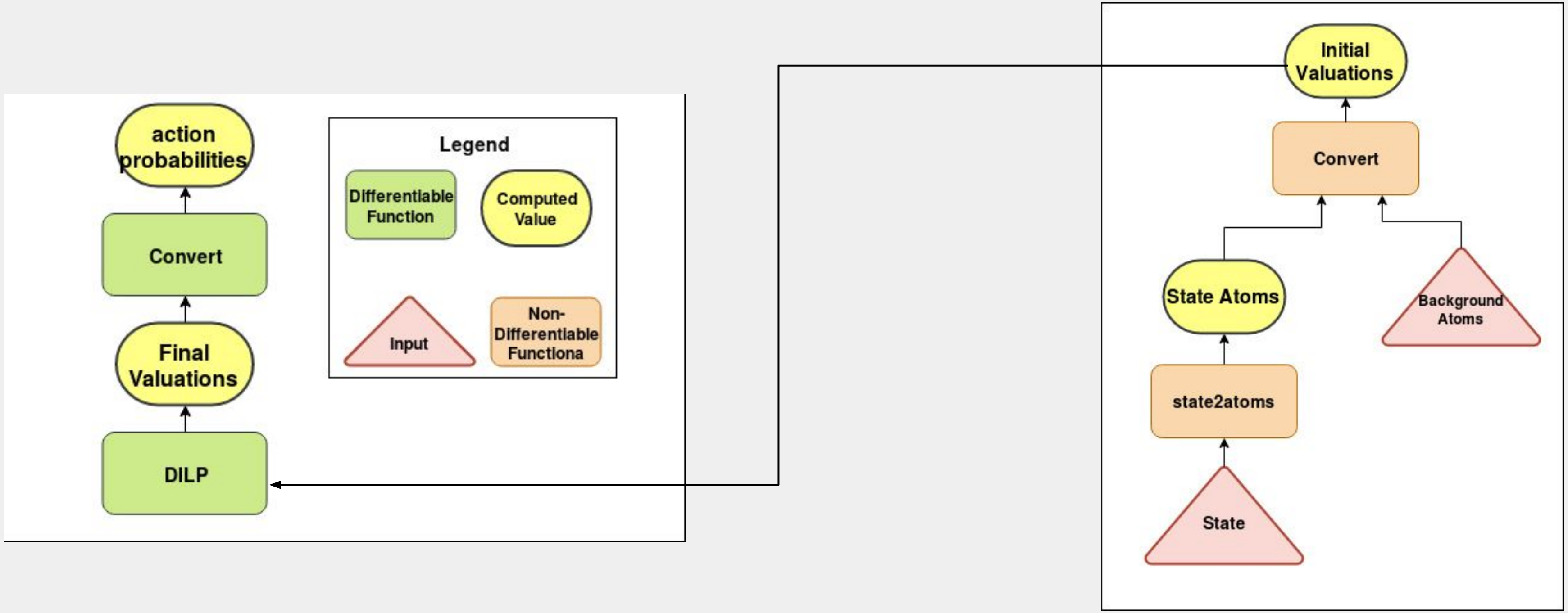
MDP with Logic Interpretation

- An MDP with logic interpretation is a triplet (M, p_S, p_A):
  - $M = (S, A, T, R)$ is a finite horizon MDP
  - $p_S : S \rightarrow 2^G$ is the state encoder, that maps each state to a set of extensional atoms including both information of the current state and background knowledge
  - $p_A : [0,1]^{|D|} \rightarrow [0,1]^{|A|}$ is the action decoder, maps the valuation of a set of atoms D to probability of actions.

- p_A, should be differentiable as we train the system with policy gradient methods on both discrete and stochastic.

  $p_A(a|e) = \{(l(e,a)/ \sigma, \sigma \geq 1\}$
- Action determination: adding a set of action predicates to the architecture, which in turn depends on the evaluation of these action atoms.
- The advantage of using neural network architecture to represent p_A is that, the decision making process is much more flexible.

NLRL: Neural Logic Reinforcement Learning



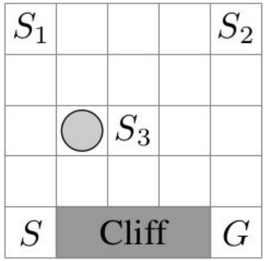
NLRL: Algorithm

Steps of Algorithm

```
1.  forward_n <- number of times we chain
2.  C <- constants
3.  $P_e$ <- extensional predicates          // predicates whose values are initially known
4.  $P_t$ <- target predicates               // action predicates we want to predict values of
5.  $P_a$ <- auxiliary predicates            // extra predicates which help us in better learning
6.  $P_i$ <- $P_t$ + $P_a$                   // set of intensional atoms
7.  P <- $P_e$ + $P_i$                       // set of all predicates
8.  G <- get_ground_atoms()
9.  base_valuation <- array, where base_valuation[atom] = 1 if atom is True independent of the state, 0 otherwise $\forall$ atom $\in$ G
10. clauses <- dict{pred, create_clauses(pred)} $\forall$ pred $\in$ $P_i$
11. weights <- dict{pred, dict{clause, w}}, $\forall$ clause $\in$ clauses[pred], $\forall$ atom $\in$ $P_i$, sum(w)=1
12. batch <- create_batch(num_episodes)
13. train(itr)
```

NLRL: Cliff Walking Example

Example: Cliff Walking



state: $current(1, 2)$

background: $zero(0)$, $last(4)$, $succ(0, 1)$, $succ(1, 2)$, $succ(2, 3)$, $succ(3, 4)$

Learnt Policy

$0.990 : right() \leftarrow current(X, Y), succ(Z, Y)$

$0.561 : down() \leftarrow pred(X), last(X)$

$0.411 : down() \leftarrow current(X, Y), last(X)$

$0.988 : pred(X) \leftarrow zero(Y), current(X, Z)$

$0.653 : left() \leftarrow current(X, Y), succ(X, X)$

$0.982 : up() \leftarrow current(X, Y), zero(Y)$

Experiments

Domain: Block Manipulation

- Performance examined on 3 subtasks: STACK, UNSTACK and ON
- For training, 5 entities: 4 blocks and a floor
- Ground Atoms:
  - on(X,Y): block X is on entity Y
  - top(X): X is on top of the column
- Auxiliary predicates: 4
- Background knowledge:
  - Common to all tasks: isFloor(floor)
  - For ON task: goalOn(X,Y)
- Termination:
  - reaching goal state
  - Taking 50 steps
- States are represented using tuples of tuples.
- Action atoms(25): move(X,Y), move X on top of Y.
- Variations: swap top 2, swap right 2, swap middle 2, 2 columns, 4 blocks, 5 blocks, 6 blocks, 7 blocks

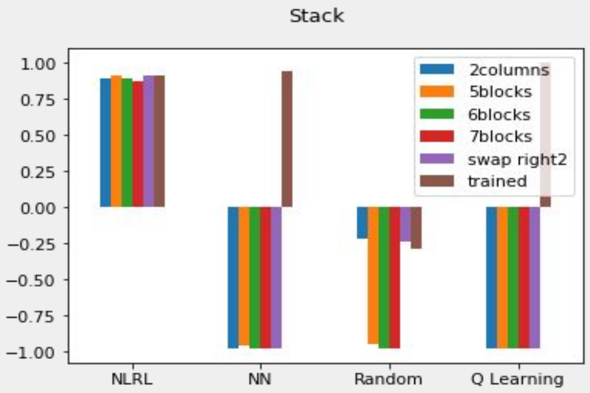
Experiments

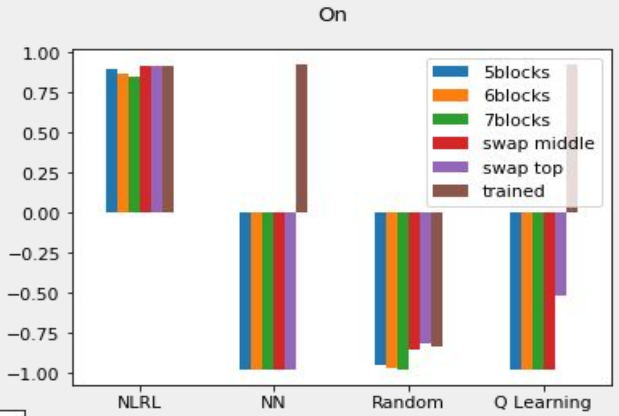
Domain: Cliff Walking

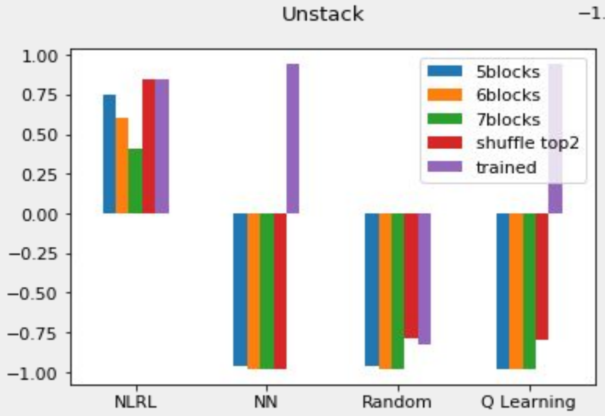
- Performance examined on 2 subtasks: cliff-walking, windy cliff-walking
- For training, 5 by 5 grid
- Ground Atoms:
  - current(X,Y): X, Y represent coordinates of agent on the grid
- Auxiliary predicates: 4
- Rewards:
  - Cliff position, -1
  - Goal position, +1
  - For each step: -0.02
- Termination:
  - reaching cliff or goal state
  - Taking 50 steps
- State representation: current(X,Y)
- Action atoms:up(), down(), left() and right()
- Variations: top left, top right, center, 5 by 5, 6 by 6, 7 by 7

INTERNATIONAL INSTITUTE OF
INFORMATION TECHNOLOGY
HYDERABAD

Results (Block Manipulation)

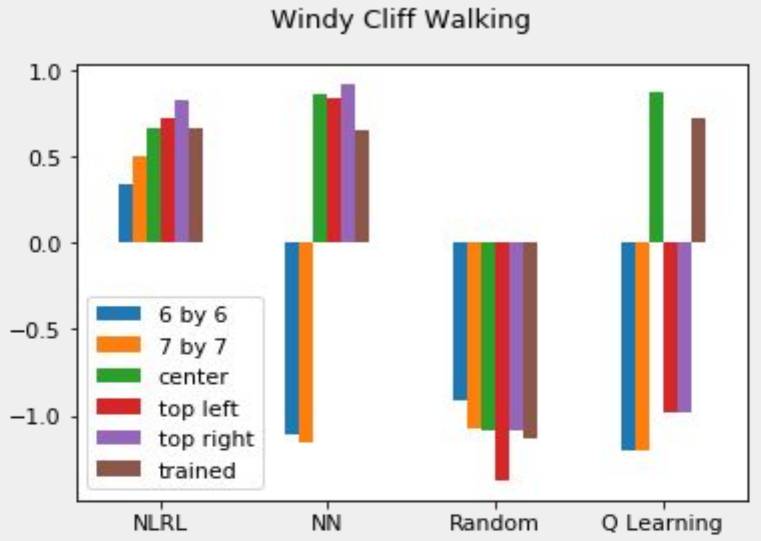


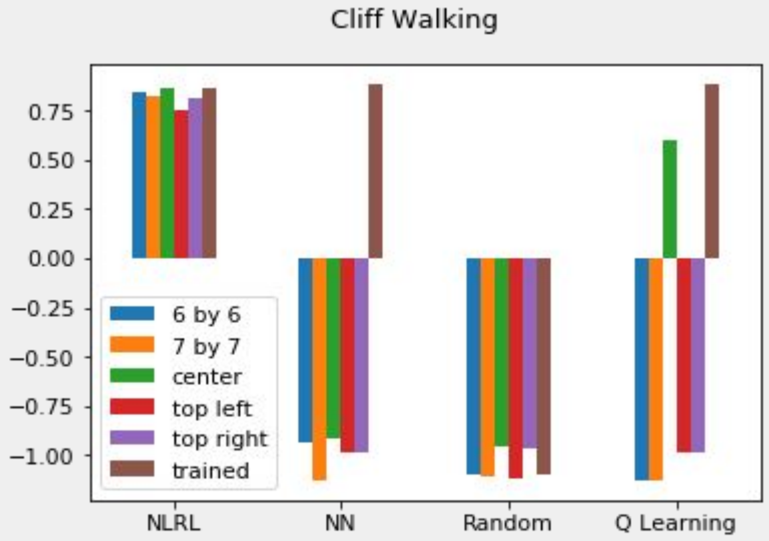




INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY
HYDERABAD

Results (Cliff Walking)





INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY
HYDERABAD

Analysis

NLRL

- NLRL achieves a near-optimal performance in all the training environments and also successfully adapts to all the variations of the environments.
- In most of the generalization tests, the agents manage to keep the performance in the near optimal level

Neural Network

- In varied environments, the neural networks perform even worse than a random player.

RL Agent

- The Q Learning agent learns optimal policy in all the training environments for all subtasks.
- The Q Learning agent appears to only to perform well for states it has already visited in the training environment but is unable to perform for unvisited state i.e. it is unable to learn a general approach for the environments.

INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY
HYDERABAD

Thank You

Team Pikachu

INTERNATIONAL INSTITUTE OF
INFORMATION TECHNOLOGY
HYDERABAD