Strong Generalization and Efficiency in Neural Programs

Кнунянц Иван, 171

Введение
Две важных цели

Корректность и эффективность.

- Немногие алгоритмы смогли продемонстрировать сильное обобщение, т.е. корректность
- Эффективность либо было сложно показать, либо это не было главным направлением исследования

Введение
Neural Program Induction

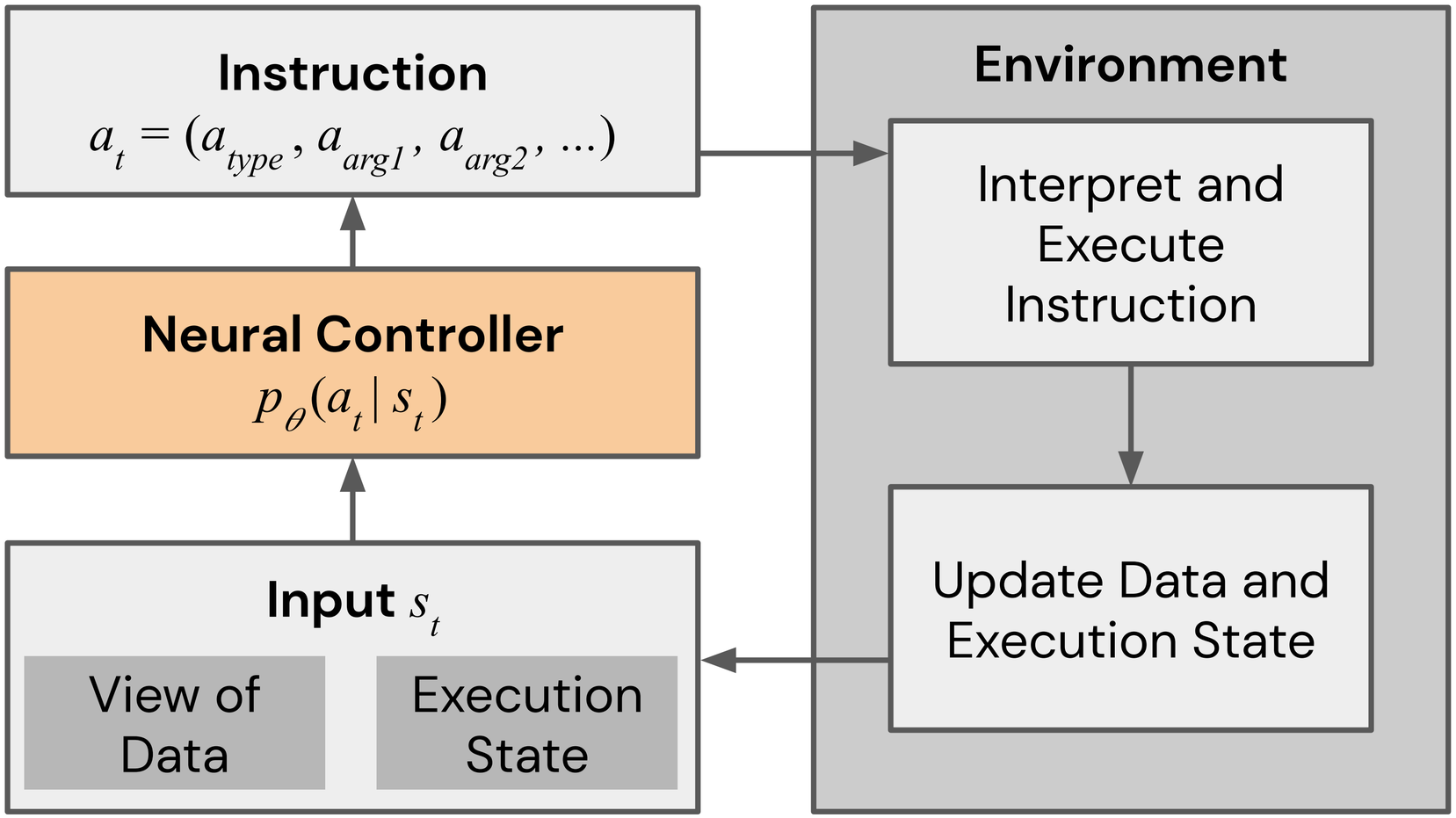
- Нейронный контроллер, который для состояния s возвращает инструкцию $a = f(s)$
- Итерируется до критерия останова или максимального количества шагов
- Нейронная сеть параметризует policy распределения $p(a \mid s)$

Введение

Два важных результата статьи:

- Для эффективности и обобщения важно, какой вид у входа и выхода
- В то время, как обучение с учителем может подстроить алгоритм под учителя, RL может помочь превзойти его

Описание метода



Описание метода

Важные составляющие: входные данные, инструкции и модель.
Для эффективности и обобщения используется константный размер входа.

В данном методе каждая инструкция имеет тип a_{type} и аргументы

$a_{arg1}, a_{arg2}, \cdots$

Представление входных данных вместе с инструкциями формируют класс алгоритмов, который может представить нейронный контроллер.

Описание метода
Модель

Отображение $a_t = f(s_t)$ определяется через $p_\theta(a_t \mid s_t)$:

$$p_\theta(a_t \mid s_t) = p_\theta(a_{type} \mid s_t) p_\theta(a_{args} \mid a_{type}, s_t)$$

$$p_\theta(a_{args} \mid a_{type}, s_t) = p_\theta(a_{arg1} \mid a_{type}, s_t) p_\theta(a_{arg2} \mid a_{arg1}, a_{type}, s_t)$$

В зависимости от того, имеет s_t фиксированный размер или переменный, мы используем или MLPs или GNNs.

Обучение
С учителем

Учитель генерирует $(s_0, a_0, s_1, a_1, \cdots)$

Минимизирует $-\sum_t \log p_\theta(a_t \mid s_t)$

Быстро обучается, но не может превзойти учителя.

Обучение
RL

Награда за каждый шаг: $r_t = -c$.

Обновляем параметры θ и ϕ по направлению:
$[G_t - V_\phi(s_t)] \nabla_\theta \log p_\theta(a_t \mid s_t) + \mu \nabla_\phi [G_t - V_\phi(s_t)]^2$, где

$$G_t = r_t + \gamma r_{t+1} + \cdots + \gamma^{n-1} r_{t+n-1} + \gamma^n V_\phi(s_{t+n})$$

μ - скалярный вес

Добавляется лосс имитации $-\lambda \log p_\theta(a'_t \mid s_t)$, где $a'_t = T(s_t)$

Сортировка
Переменный размер входа

Инструкция: Swap(i, j)

s_t содержит информацию о всем листе на каждом шаге. Устроен как полный граф из n вершин, поэтому размер входа переменный и используется GNNs.

Получаем сложность $O(n^3)$

Сортировка
Константный размер входа

Обычный алгоритмы не смотрят на весь список.

1. Стоимость каждого шага становится константной
2. Алгоритм не зависит от длины последовательности

Будем работать с k индексами и включать во вход сравнение $\mathbf{A}[i]$ с $\mathbf{A}[i-1]$ и $\mathbf{A}[i+1]$. Плюс для каждого j, i будем знать сравнение $\mathbf{A}[i]$ с $\mathbf{A}[j]$

k = 4, $v_1, v_3 = 0; v_2, v_4 = n-1$

Добавим инструкции: SwapWithNext(i), MoveVar(i, +1/-1):$v_i \leftarrow min\{v_i, n-1\}$ или $max\{v_i - 1, 0\}$, AssignVar(i, j): $v_i \leftarrow v_j$

Сортировка
Результаты

Instance size	5	10	20	30	40	50	1000	Complexity
Full view	72	**100**	**100**	33	0	0	0	$O(n^3)$
Interface view	**100**	**100**	**100**	**100**	**100**	**100**	**100**	$O(n^2)$

Повышение эффективности
Функции позволяют разделять и властвовать

- FunctionCall(id,l_1,...,l_p,o_1,...,o_p,r_1,...,r_p)

id - ID функции, l_i - ID локальной переменной, o_i - ID внешней переменной, r_i - ID внешней переменной, получающей возвращенное значение

- Return(l_1' , ..., l_q')

l_i' - ID возвращаемой локальной переменной

s_t содержит информацию, что и раньше, ID функции и кодировку прошлого состояния

Для задачи сортировки будем использовать 2 функции и Swap(i, j)

При обучении с учителем получаем имитацию быстрой сортировки с $O(n \log n)$

Повышение эффективности

RL превосходит учителя

Shaping награда: $h(\mathbf{A}) = \sum_{i=low}^{high-1} \mathbb{I}[\mathbf{A}[i] \leq \mathbf{A}[i+1]]$, когда RL from scratch

Общая награда: $r_t = h(\mathbf{A_{t+1}}) - h(\mathbf{A_t}) - c$

Обучение быстрее, но есть риск попасть в локальный минимум

Instance size	5	10	20	30	50	100	200	500	1000	10000
Bubble sort	13.5	68.0	293.6	677.8	1,874.7	7,527.0	30,046.8	187,506.2	750,519.3	-
Insertion sort	13.7	53.4	208.6	475.3	1,275.7	5,077.5	20,134.3	125,136.8	501,051.7	-
RL from scratch	8.7	49.7	252.0	617.5	1,763.7	7,320.5	29,623.0	186,305.1	748,784.5	-
RL + imitation	**8.2**	**44.3**	**190.7**	**446.6**	**1,228.5**	**4,981.2**	**19,939.2**	**124,643.0**	**500,057.0**	-
Quick sort	27.4*	87.5*	241.8	399.9	788.8	1,840.2	4,217.1	12,519.4	27,633.5	368,338.6
RL from scratch	**13.5***	**56.0***	**220.6**	495.5	1,315.6	5,166.5	20,321.4	125,622.0	502,035.5	-
RL + imitation	24.8*	79.9*	223.7	**377.2**	**728.3**	**1,717.4**	**3,914.4**	**11,578.2**	**25,635.0**	**338,947.9**

Общность

Поиск в упорядоченном списке

Используется тот же входной интерфейс, MoveVar, AssignVar

Добавляется:

- AssignMidVar(I, j, k): $v_i \leftarrow \lfloor (v_j + v_k)/2 \rfloor$
- Found(i): элемент q найден по индексу v_i
- NotFound(): элемент q не найден в $\mathbf{A}$

Эта настройка определяет семейство контроллеров, которое включает в себя как алгоритм линейного сканирования $O(n)$, так и алгоритм двоичного поиска $O(logn)$.

Общность

Поиск в упорядоченном списке

Instance Size	5	10	20	30	50	100	200	500	1000
Linear Search	2.2	4.6	9.7	14.2	23.3	50.0	106.7	220.7	446.6
Binary Search	2.6	3.9	6.3	7.5	9.2	11.7	**14.6**	**19.1**	**21.8**
RL from scratch	**1.3**	**2.5**	4.3	5.9	9.9	15.5	29.7	85.7	168.6
RL + imitation	1.4	2.6	**4.1**	**4.7**	**7.0**	**10.6**	18.4	41.5	85.8

Общность

0/1 задача о рюкзаке

$$w_i, v_i \sim U[0,1],\ W = \frac{1}{2}\sum_i w_i$$

Инструкции: Pop(i), Put(i)

- MoveVar(+1/-1): $i \leftarrow i + 1$
- Knapsack() - рекурсивно вызывает функцию knapsack
- Return() - возвращает текущий уровень рюкзака

Это семейство алгоритмов включает DFS

$r_t =$ (лучшая ценность после a_t) $-$ (лучшая ценность до a_t)

Общность

0/1 задача о рюкзаке

Budget	Size	2	4	6	8	10	20	40	80	160	320	640
20x size	DFS	0.45	1.25	1.82	2.47	3.14	5.94	11.10	21.67	42.11	82.01	163.42
	RL	0.45	**1.26**	**2.03**	**2.83**	**3.51**	**6.51**	**11.78**	**22.40**	**43.27**	**83.07**	**164.56**
100x size	DFS	0.45	1.26	**2.04**	2.81	3.43	6.43	11.75	22.39	42.95	82.82	164.22
	RL	0.45	1.26	2.03	**2.86**	**3.70**	**6.98**	**12.41**	**23.15**	**44.16**	**84.05**	**165.60**

Вопросы

1. Распишите, как выглядит policy с объяснением.
2. Как выглядит reward вместе с shaping reward и зачем его добавлять?
3. Расскажите о приложении к задаче о рюкзаке.