Эвристические методы для эволюционных стратегий в задачах машинного обучения с подкреплением

М.Е. Наумов[1], А.В. Благов[1]

[1]Самарский национальный исследовательский университет им. академика С.П. Королева, Московское шоссе 34А, Самара, Россия, 443086

Аннотация. Существует множество методов решения задач машинного обучения, каждый из которых применяется в зависимости от решаемой задачи. В данной статье описаны несколько классических алгоритмов эволюционных стратегий в задаче машинного обучения с подкреплением, улучшение простейшей реализации с помощью эвристических алгоритмов, описание экспериментальной системы, а также полученные результаты.

1. Введение

В данной статье предлагается улучшение простейшей реализации эволюционных стратегий с помощью эвристических методов, а также сравнение с традиционными подходами на задаче балансировки обратного маятника находящегося на подвижной тележке. В качестве системы, решающей задачу используется искусственная нейронная сеть со 128 скрытыми нейронами.

Практическая ценность улучшения существующих алгоритмов заключается в улучшении скорости сходимости к решению, а также расширении области начальных приближений из которых метод будет сходится к решению.

Эволюционная стратегия – метод оптимизации, основанный на идеях эволюции. Возможные решения кодируются как вектора вещественных чисел. Данный метод успешно применяется для решения задач машинного обучения с подкреплением [1][2][3].

Эволюционные стратегии имеют следующие особенности:

1. Не зависят от оптимизируемой модели. Они способны оптимизировать любые модели, которые можно выразить через вектора вещественных чисел. На модель не накладывается никаких других ограничений (не обязательно даже существование градиента).
2. Не требуется вычисления градиента параметров оптимизируемой модели.
3. Не требуется совершать обратное распространение градиентов, что позволяет избежать множества проблем при их перемножении.
4. Инвариантны по отношению к задержке перед откликом системы на воздействие, что упрощает их использование в задачах машинного обучения с подкреплением.
5. Очень высокая степень параллелизма, а также малое количество требуемых коммуникаций.

Есть подходы которые используют эволюционные стратегии не для оптимизации, а для численного расчёта градиента оптимизируемой функции. Таким образом возможно

использование более высокоуровневых и точных методов, без необходимости вычисления градиента функции (которого может и не быть).

2. Традиционные методы эволюционных стратегий

Основной цикл эволюционной стратегии состоит из двух этапов: мутации и отбора.

Рассмотрим эти этапы на примере максимизации некоторой векторной функции $f(x)$:

1. Производится выбор начального решения. Примем начальное решение за текущее.
2. Шаг мутации: из текущего решения создадим μ новых, путём добавления к текущему решению μ случайных векторов, распределённых по многомерному нормальному распределению с мат. ожиданием равным текущему решению и корреляционной матрицей σI.
3. Вычислим значение функции f в каждом из возможных решений.
4. Шаг отбора: в качестве текущего решения возьмём лучшее решение из возможных (или оставим текущее, если оно лучше), либо посчитаем взвешенное среднее возможных решений, учитывая что более удалённые решения имеют больший вес.
5. Повторяем шаги 2-4 до тех пор, пока не получим приемлемое решение.

На шаге 4 используется следующая формула:

$$\hat{x} = \frac{1}{\mu\sigma}\sum_{k=1}^{\mu}\left[x_k \frac{f(x_k) - \hat{f}}{\tilde{f}}\right], \quad (1)$$

где $\hat{x}$ – усреднённое решение, x_k – возможное решение под номером k, $\hat{f}$ – выборочное среднее значение $f(x_k)$, $\tilde{f}$ – выборочная дисперсия $f(x_k)$.

Данный алгоритм покрывает не все возможные эволюционные стратегии, но показывает их отличительные особенности: решение представляется в виде вектора вещественных чисел; на каждом шаге отбирается одно (или несколько) ключевое решение, используемое на следующем шаге как основа для новых решений; отбор происходит детерминированным образом, а значит может производиться на кластере без пересылки данных; относительная простота алгоритма, которая отражается в быстроте алгоритма.

Заметим что высокое быстродействие и большой параллелизм алгоритма позволяет производить большее количество итераций по сравнению с другими методами, решающими задачи машинного обучения с подкреплением.

Приведённый выше алгоритм так же называют простой эволюционной стратегией, так как он не изменяет параметр σ в зависимости от уже известных данных, что ухудшает сходимость, или делает нахождение решения невозможным. Данный алгоритм был реализован в этой работе. Вычислительная сложность одной итерации равна $O(N)$ с малой константой, где N– размерность задачи.

Существует множество алгоритмов, позволяющих решить проблему постоянного параметра σ. В данной статье рассматривается метод под названием CMA-ES (covariance matrix adaptation evolution strategy) [4] как наиболее популярный. В качестве реализации используется библиотека для Python под названием pycma. Рассмотрим упрощённую версию этого алгоритма (отражающую основные идеи алгоритма), при размерности задачи равной N:

1. Выбрать количество возможных решений на каждом шаге μ, обычно выбирается значение больше четырёх. Выбрать начальное решение и сделать его текущим (m). Выбрать начальный вектор параметров σ длины N, отвечающий за длину шага по каждому направлению. Задать корреляционной матрицу C равной единичной матрице, с размерностью $N \times N$.
2. Сгенерировать μ случайных векторов x_k, соответствующих возможным решениям. Генерация происходит из многомерного нормального распределения с корреляционной матрицей C и мат. ожиданием равным m.
3. Рассчитать значение функции f в каждом возможном решении.

4. Упорядочить возможные решения по значению функции f.
5. Обновить значение корреляционной матрицы C, взяв за основу данные о распределении возможных решений. Вычисления производятся по формуле:

$$\hat{x} = \frac{1}{\mu\sigma}\sum_{k=1}^{\mu}\left[x_k \frac{f(x_k) - \hat{f}}{\hat{f}}\right], \quad (2)$$

где x_k – k случайный вектор; $N_{\text{лучших}}$ – количество лучших решений, берущихся для рассмотрения; $\hat{x}$ – среднее арифметическое $N_{\text{лучших}}$ векторов, соответствующих лучшим возможным решениям.

6. Обновить значение текущего решения m, по формуле:

$$C_{ij} = \frac{1}{N_{\text{лучших}}}\sum_{k=1}^{N_{\text{лучших}}}\left(x_{k_i} - \hat{x}_i\right)\left(x_{k_j} - \hat{x}_j\right) \quad (3)$$

7. Повторять шаги 2-6 до тех пор, пока не получим подходящее решение.

Данный алгоритм выполняет шаги 2-6 за $O(N^2)$ (в силу того, что $N_{\text{лучших}}$ – константа), где N – размерность задачи. Есть подходы, которые сводят сложность к $O(N)$ с большой константой. В данной статье используется наиболее полная и корректная реализация алгоритма из библиотеки pycma для языка Python.

3. Эвристические методы эволюционных стратегий

Приведённые выше алгоритмы либо были неточными (простая эволюционная стратегия), либо были медленными (адаптивные эволюционные стратегии). Возможно ли построить алгоритм с вычислительной сложностью $\boldsymbol{O(N)}$ с малой константой и при этом большей точностью чем простая реализация эволюционных стратегий?

Рассмотрим следующую модификацию простой эволюционной стратегии:

1. Производится выбор начального решения. Примем начальное решение за текущее.
2. Шаг мутации: из текущего решения создадим μ новых, путём добавления к текущему решению μ случайных векторов, распределённых по многомерному нормальному распределению с мат. ожиданием равным текущему решению и корреляционной матрицей σI.
3. Вычислим значение функции f в каждом из возможных решений.
4. Шаг отбора: в качестве текущего решения возьмём лучшее решение из возможных (или оставим текущее, если оно лучше), либо посчитаем взвешенное среднее возможных решений, учитывая что более удалённые решения имеют больший вес.
5. Рассчитаем новое значение σ с помощью одной из эвристических функций, которые будут рассмотрены дальше.
6. Повторяем шаги 2-5 до тех пор, пока не получим приемлемое решение.

На 5 шаге, вместо вычисления параметра $\boldsymbol{\sigma}$ можно рассчитывать всю корреляционную матрицу, однако это увеличит вычислительную сложность алгоритма с $\boldsymbol{O(N)}$ до $\boldsymbol{O(N^2)}$.

Под эвристикой понимают совокупность приёмов и методов, облегчающих и упрощающих решение познавательных, конструктивных, практических задач. В данной статье под эвристическими функциями понимаются функции, которые позволяют улучшить точность вычислений, не повысив при этом асимптотическую сложность. Рассмотрим несколько примеров таких функций:

$$\boldsymbol{h_1(fx, i, N) = \frac{1}{1+e^{-fx}}}; \quad (4)$$

$$\boldsymbol{h_2(fx, i, N) = arctan(fx) + \frac{\pi}{2}}; \quad (5)$$

$$\boldsymbol{h_3(fx, i, N) = \frac{N-i}{N}}; \quad (6)$$

$$\boldsymbol{h_4(fx, i, N) = 1 - sin^2\left(\frac{i}{N}10\pi\right)}, \quad (7)$$

где (4)-(7) – эвристические функции; $\boldsymbol{fx}$ – значение оптимизируемой функции в точке текущего решения; $\boldsymbol{i}$ – номер текущей итерации; $\boldsymbol{N}$ – предельное количество итераций.
Функция (6) реализует идею, аналогичную температуре в методе имитации отжига [5]. В начале оптимизации алгоритм рассматривает более отдалённые точки, а затем постепенно "остывает" и переходит к более точному поиску решения. Такой подход позволяет процессу оптимизации выходить на более перспективные решения в самом начале, а затем уточнять их.
Эвристическая функция (7) представляет собой довольно интересный случай: она проходит несколько полных периодов за время оптимизации функции. Это позволяет процессу оптимизации выходить из локальных оптимумов, ускоряя нахождение решения.
Не все эвристические функции похожи на перечисленные, однако этот список даёт некоторую стартовую точку для поиска подходящего вида функции.
Заметим также что из приведённых функции можно составить линейную комбинацию, используя положительные коэффициенты, и получить новую эвристическую функцию.

4. Описание тестирующей системы
В качестве проблемы, на которой будут сравниваться реализации приведённых выше алгоритмов, было взято окружение CartPole-v1 из Python библиотеки gym. Задача состоит в балансировке обратного маятника, находящегося на подвижной тележке. За каждый шаг симуляции, при котором маятник направлен вверх, внешняя среда выдаёт сигнал, равный 1. Результатом симуляции будем считать суммарный отклик внешней среды. Симуляция заканчивается, если маятник отклонился более чем на 15 градусов от вертикального положения, тележка сдвинулась более чем на 2,4 условных единиц от центра или прошло 150 шагов симуляции. Последнее ограничение наложено искусственно, для ускорения эксперимента.
В качестве оптимизируемой модели, которая управляет тележкой, была взята искусственная нейронная сеть, состоящая из одного скрытого слоя, содержащего 128 узлов. Нейронная сеть реализованна на numpy, с применением jit компилятора numba, что позволило значительно ускорить вычисления, а также увеличить максимальное число итераций в эксперименте.
В качестве эвристической функции использовалась следующая функция:

$$h(fx, i, N) = \left(\frac{1}{1 + e^{fx}} + \frac{2N - i}{2N}\right)\frac{1}{2}. \quad (8)$$

5. Итоговая таблица сравнения методов
Данные, полученные в результате эксперимента, приведены в таблице 1.

Таблица 1. Значения среднего времени (сек.) и среднего результата оптимизации.

Количество итераций	Простая ES		Эвристическая ES		CMA-ES	
	Результат	Время	Результат	Время	Результат	Время
100	51.00	41.5	83.26	40.9	150.00	357.4
200	66.25	87.1	94.01	88.2	-	-
300	62.41	136.9	103.58	135.4	-	-

6. Заключение
Как видно из таблицы 1, метод эвристических эволюционных стратегий работает значительно лучше, чем метод простых эволюционных стратегий. Также, как и ожидалось, алгоритм CMA-ES обходит остальные по результатам оптимизации.
Из эксперимента становится очевидным ещё один недостаток простых эволюционных стратегий: при увеличении числа итераций возможно ситуация, когда результат не становится лучше. Это объясняется тем, что алгоритм может придти к такому состоянию, в котором он

будет «переступать» через оптимальное решение, из-за неоптимальной длины шага. Применение метода эвристических эволюционных стратегий значительно снижает вероятность попадания в такую ситуацию.

Исходя из приведённых данных, делаем следующий вывод: наиболее оптимальной стратегией использования приведённых выше алгоритмов является комбинация CMA-ES и эвристических эволюционных стратегий. В начале решения задачи используется эвристический метод, позволяя быстро получить примерное решение, а затем применяется CMA-ES к полученному решению, уточняя его.

7. Литература

[1] Salimans, T. Evolution Strategies as a Scalable Alternative to Reinforcement Learning / T. Salimans, J. Ho, X.Chen, S. Sidor, I. Sutskever [Electronic resource]. – Access mode: https://arxiv.org/pdf/1703.03864 (4.11.2018).

[2] Lehman, J. ES Is More Than Just a Traditional Finite-Difference Approximator / J. Lehman, J. Chen, J. Clune, K.O. Stanley [Electronic resource]. – Access mode: https://arxiv.org/pdf/1712.06568 (4.11.2018).

[3] Clune, J. Improving Exploration in Evolution Strategies for Deep Reinforcement Learning via a Population of Novelty-Seeking Agents / J. Clune, E. Conti, V. Madhavan, F. Petroski Such, J. Lehman, K.O. Stanley [Electronic resource]. – Access mode: https://arxiv.org/pdf/1703.03864 (4.11.2018).

[4] Hansen, N. The CMA Evolution Strategy: A Tutorial [Electronic resource]. – Access mode: https://arxiv.org/pdf/1604.00772 (4.11.2018).

[5] Kirkpatrick, S. Optimization by simulated annealing / S. Kirkpatrick, C.D. Gelatt, M.P. Vecchi // Science. – 1983. – Vol. 220(4598). – P. 671-680.

Heuristic methods for evolutionary strategies in reinforcement learning problems

M. Naumov[1], A. Blagov[1]

[1]Samara National Research University, Moskovskoe Shosse 34A, Samara, Russia, 443086

Abstract. The article is devoted to the study of methods for evolutionary strategies in reinforcement learning problems. The authors propose the Heuristic method that has certain advantages over existing methods.