Обзор GOLEM

Владимир Латыпов

06-02-2024

NSS Lab ITMO

История появления

Изначально была библиотека `FEDOT` для AutoML, основана на pipeline-ах ≈произвольной структуры (dag), поиск происходит посредством эволюции. Но алгоритм графовой оптимизации оказался полезен и для кучи других задач, в т.ч. проектов лаборатории:

- BAMT (Bayesian AutoML Tool)
- NAS (Neural Architecture Search)
- GEFEST (**G**enerative Evolution For Encoded STructures)
- пользовательские применения (коллаборация с химической лабораторией, например — btw полезный подход)

Поэтому было решено выделить эту часть в отдельную библиотеку — GOLEM.

Возможности FEDOT

		STREAMLINE	Auto-Keras	H20-3 AutoML	MLme	LAMA	FEDOT	FLAML	ALIRO	PYCARET	Auto-Gluon	MLJAR-supervised	Ludwig	TPOT	Auto-Sklearn	Auto-PyTorch	GAMA	Hyperopt-sklearn	Auto-WEKA	RECIPE	ML-PLan	TransmogrifAI	MLBox	Xcessiv	Auto_ML
Target	Binary Classification	YES	YES	YES	YES	YES	YES	YES	YES	YES	YES	YES	YES	YES	YES	YES	YES	YES	YES	YES	YES	YES	YES	YES	YES
	MultiClass Classification	NO	YES	YES	YES	YES	YES	YES	YES	YES	YES	YES	YES	YES	YES	YES	YES	YES	YES		YES	YES	YES	YES	YES
	Regression	NO	YES	YES	NO	YES	YES	YES	YES	YES	YES	YES	YES	YES	YES	YES	YES	YES	YES	NO	YES	YES	YES	YES	YES
	Multi-Task	NO	YES	NO	NO	NO	NO	NO	NO	NO	NO	NO	YES	NO	NO	NO	NO	NO	NO	NO	NO	NO	NO	NO	NO
	Multi-Label	NO	NO	NO	NO	NO	NO		NO	NO	NO	NO	NO	NO	YES	NO	NO	NO	NO	NO	NO	NO	NO	NO	NO
	Clustering	NO	NO	NO	NO	NO	YES	NO	NO	YES	NO	NO	NO	NO	NO	NO	NO	NO	NO	NO	NO	NO	NO	NO	NO
	Anomoly Detection	NO	NO	NO	NO	NO	NO	NO	NO	YES	NO	NO	NO	NO	NO	NO	NO	NO	NO	NO	NO	NO	NO	NO	NO

Глобально про алгоритм

Тюнинг, разные подходы

Многокритериальная оптимизация

Операторы

Мутации и кроссовер учитывают `GraphGenerationRequirements`, а также `GraphVerifier`.

- Мутация
  - ↳ Заменить аттрибут каждой ноды с заданной вероятность на то, что сгенерирует `NodeFactory` и так, как посоветует `ChangeAdvisor` (например, в химии некоторые связи невозможны)
  - ↳ Добавление ребра между случайными нодами
  - ↳ Вставка случайной ноды в разрыв ребра
  - ↳ Заменить/выбросить ноду
  - ↳ Заменить поддерево ноды на случайное
  - ↳ ...пользовательские, особенно семантические

- Кроссовер
  - ↳ Замена случайных поддеревьев
  - ↳ Нахождение структурно эквивалентных подграфов и замена случайных нод или поддеревьев в них
  - ↳ Смена родителями с аналогичной нодой в другом графе
  - ↳ ... пользовательские
- Селекция
  - ↳ Tournament(fraction): каждый раз из группы размера $\approx$ population_size $*$ fraction выбирается лучший в итоговую популяцию и убирается из кандидатов.
  - ↳ SPEA-2

Strength — количество особей, которые доминирует заданная:

$$S(\boldsymbol{i}) = \left|\left\{\boldsymbol{j} \mid \boldsymbol{j} \in P_t + \overline{P_t} \wedge \boldsymbol{i} \succ \boldsymbol{j}\right\}\right|$$

Тогда назначаем raw fitness-ом сумму strengths всех, кого особь доминирует (то есть особо выгодно доминировать крутых):

$$R(\boldsymbol{i}) = \sum_{\boldsymbol{j} \in P_t + \overline{P_t}, \boldsymbol{i} \succ \boldsymbol{j}} S(\boldsymbol{j})$$

Но ещё хотим учитывать разнообразие. Про пространство оптимизации в общем случае ничего не знаем, поэтому считаем разнообразие в пространстве objective functions:

$$R(i) = \frac{1}{\sigma_{\boldsymbol{i}}^k + 2}$$

, где σ_i^k — расстояние k-го ближайшего среди популяции + архива, а k выбирают $\sqrt{N + \overline{N}}$

- ↳ Селекция новой популяции: производим селекцию турниром по F с заменой (участников турнира убираем из кандидатов)
- ↳ Селекция архива: выбираем недоминированных, но если не соответствует целевому размеру:
  - ↳ Слишком большой: truncat-им по фитнессу (раз недоминированные, эквивалентно, по плотности): на каждому шагу убираем лексикографически меньшего по вектору расстояний до k-го ближайшего:

$$\boldsymbol{i} \leq_d \boldsymbol{j} \quad :\Leftrightarrow \quad \forall\, 0 < k < |\overline{\boldsymbol{P}}_{t+1}| \;:\; \sigma_{\boldsymbol{i}}^k = \sigma_{\boldsymbol{j}}^k \quad \vee$$
$$\exists\, 0 < k < |\overline{\boldsymbol{P}}_{t+1}| \;:\; \left[\left(\forall\, 0 < l < k \;:\; \sigma_{\boldsymbol{i}}^l = \sigma_{\boldsymbol{j}}^l\right) \wedge\; \sigma_{\boldsymbol{i}}^k < \sigma_{\boldsymbol{j}}^k\right]$$

  ↳ Слишком маленький: добавляем оставшихся сортировкой по фитнессу.

- «Регуляризация» — тоже пытается минимизировать сложность моделей (по умолчанию — отключен): рассматривает все валидные и уникальные поддеревья и выбирает среди родителей и детей лучших (эти поддеревья проще и хорошо, если они окажутся не хуже родителей)
- Genetic scheme

  - ↳ `generational`: новая популяция занимает место старой $(\mu, \lambda), \mu = \lambda$
  - ↳ `steady-state`: каждый раз добавляем по одному $-$ $(\mu + 1)$
  - ↳ `parameter-free`: $(\mu + x) - x$ растёт, если давно не было улучшений ($\Rightarrow$ нужно увеличивать exploration)
- Регулятор репродукции (Population $\to$ Population): есть min_size, max_size; некоторые операторы вероятностны и не всегда генерируют подходящие под `GraphRequirements` графы: он пробует применять операторы несколько раз + изучает, сколько в среднем процентов успешны.
- Elitism: поддерживать HallOfFame, во внеочередном подярке добавлять individual-ов оттуда.

Проект под кодовым названием ~~GAMLET~~ <anonymized>

Адаптивность

`OperatorAgent`: интерфейс подборщика мутаций, по умолчанию — `RandomAgent`, но может обучаться.

Изначальные рекомендации pipleline-ов

Surrogate model

Ближе к коду

Язык: Python. Библиотеки:

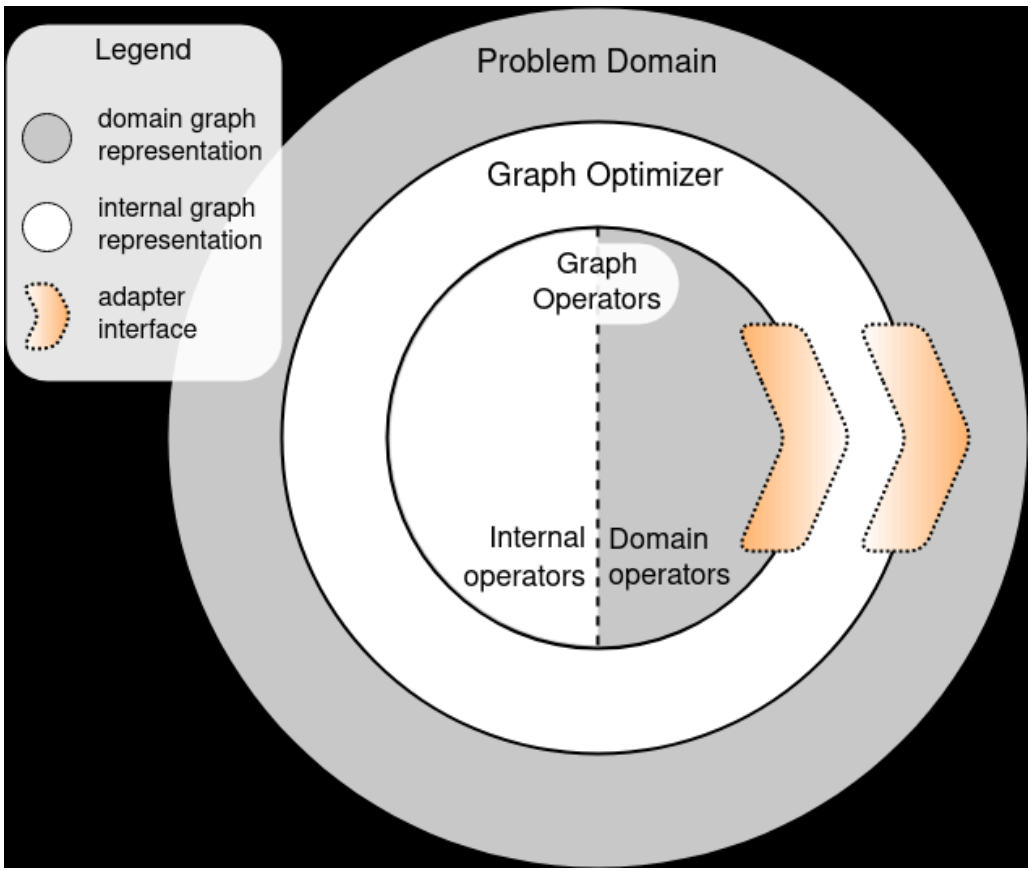
- joblib + multiprocessing
- torch + mabwiser + karateclub для контекстуального бандита на GNN
- Package core contains the main classes and scripts.
- Package core.adapter is responsible for transformation between domain graphs and internal graph representation used by optimisers.
- Package core.dag contains classes and algorithms for representation and processing of graphs.

- Package core.optimisers contains graph optimisers and all related classes (like those representing fitness, individuals, populations, etc.), including optimization history.
- Package core.optimisers.genetic contains genetic (also called evolutionary) graph optimiser and operators (mutation, selection, and so on).
- Package core.utilities contains utilities and data structures used by other modules.
- Package serializers contains class Serializer with required facilities, and is responsible for serialization of project classes (graphs, optimization history, and everything related).

- Package visualisation contains classes that allow to visualise optimization history, graphs, and certain plots useful for analysis.
- Package examples includes several use-cases where you can start to discover how the framework works.
- All unit and integration tests are contained in the test directory.

The sources of the documentation are in the docs directory.

Adapter Subsystem (преобразование между представлениями графа)



- Обычно предметная область имеет своё представление графа (например, из внешней либы): химия, BAMT, FEDOT
- Fitness, операторы
- `@register_native`, e.g. `GraphVerifier`
- Поставляется адаптер к NetworkX

Сериализация

- Pickle (e.g. бандиты)
- json (в т.ч. pipleline-ы в FEDOT)

Issues

Направления развития

NOTEARS

Theorem 1. *A matrix $W \in \mathbb{R}^{d\times d}$ is a DAG if and only if*

$$h(W) = \operatorname{tr}\left(e^{W\circ W}\right) - d = 0, \tag{7}$$

where $\circ$ is the Hadamard product and e^A is the matrix exponential of A. Moreover, $h(W)$ has a simple gradient

$$\nabla h(W) = \left(e^{W\circ W}\right)^T \circ 2W, \tag{8}$$

and satisfies all of the desiderata (a)-(d).

⟶ сводим к задаче непрерывной оптимизации; целевая функция и ограничения — дифференцируемы. Решаем с помощью метода расширенной Лагранжианы.

Ограничения:

- Целевая функция должна естественно (дифференцируемо, поменьше константа Липшица, легко вычислима) продолжаться на вещественные веса
- Как задать на пространстве, содержащем категориальные переменные?

Метаэволюция

Коэволюция

см. Признаки

Поддержка разнообразия

Expressive encodings