GAMA - это пакет AutoML, предназначенный для конечных пользователей и исследователей AutoML. Он генерирует оптимизированные конвейеры машинного обучения с учетом конкретных входных данных и ограничений на ресурсы. Конвейер машинного обучения содержит предварительную обработку данных (например, PCA, нормализацию), а также алгоритм машинного обучения (например, логистическая регрессия, случайный лес) с точной настройкой гиперпараметров (например, количество деревьев в случайном лесу).

Для поиска таких конвейеров реализовано несколько процедур поиска. GAMA также может объединять несколько настроенных конвейеров машинного обучения в ансамбль, что в среднем должно повысить производительность модели. В настоящее время GAMA ограничена задачами классификации и регрессии на табличных данных.

Помимо того, что GAMA является универсальной функциональностью AutoML, она также предназначена для исследователей AutoML. В процессе оптимизации GAMA ведет обширный журнал прогресса. Используя этот журнал, можно получить представление о поведении процедуры поиска. Например, можно построить график, отображающий эффективность работы pipeline с течением времени: график эффективности с течением времени

> Примечание: временно отключили поддержку панели GAMA Dashboard, но в конце этого года снова будет доступна визуализацию из коробки.

Лицензия Apache-2.0 License.

Принцип работы

Применяется для решения задач классификации и регрессии.

GAMA выполняет поиск по конвейерам (pipeline) машинного обучения. Например: сначала нормализация данных, а затем использование классификатора ближайших соседей для построения прогноза на нормализованных данных. Более формально конвейер машинного обучения представляет собой последовательность из одного или нескольких *компонентов*. Компонент - это алгоритм, выполняющий либо преобразование данных, либо предсказание. Это означает, что компонентами могут быть алгоритмы

предварительной обработки, такие как PCA или стандартное масштабирование, или предиктор, такой как дерево решений или машина опорных векторов. Таким образом, конвейер машинного обучения состоит из нуля или более компонентов предварительной обработки, за которыми следует компонент предсказания.

Имея набор данных, GAMA начинает **поиск наилучшей модели машинного обучения**. После поиска **лучшая из найденных моделей** может быть использована для прогнозирования. Кроме того, GAMA может **объединить несколько моделей в ансамбль**, чтобы при прогнозировании учитывать несколько моделей. Для удобства использования в GAMA предусмотрены функции **fit, predict и predict_proba**, аналогичные scikit-learn.

Установка

```
pip install gama
```

Минимальный пример для регрессии

```
from sklearn.datasets import load_diabetes
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
from gama import GamaRegressor

if __name__ == "__main__":
    X, y = load_diabetes(return_X_y=True)
    X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=0)

    automl = GamaRegressor(max_total_time=180, store="nothing", n_jobs=1)
    print("Starting `fit` which will take roughly 3 minutes.")
    automl.fit(X_train, y_train)

    predictions = automl.predict(X_test)

    print("MSE:", mean_squared_error(y_test, predictions))
```

Данные можно загружать непосредственно из файлов csv и ARFF.

Для *Attribute-Relation File Format (ARFF)*-файлов GAMA может использовать дополнительную информацию, например, какие признаки являются категориальными. Для файлов csv GAMA выводит типы столбцов, но это может привести к ошибкам. требуется указать пути к используемым файлам. Скрипт примера может быть запущен, например, с

использованием файлов breast_cancer_train.arff и breast_cancer_test.arff. Целевой показатель всегда должна быть указан в качестве последнего столбца, если не указан столбец target_column. Обязательно скорректируйте путь к файлу, если он выполняется не из каталога примеров.

```
from gama import GamaClassifier

if __name__ == "__main__":
    file_path = "../tests/data/breast_cancer_{}.arff"

    automl = GamaClassifier(max_total_time=180, store="nothing", n_jobs=1)
    print("Starting `fit` which will take roughly 3 minutes.")
    automl.fit_from_file(file_path.format("train"))

    label_predictions = automl.predict_from_file(file_path.format("test"))
    probability_predictions = automl.predict_proba_from_file(file_path.format("test"))
```

Поддержка CSV и ARFF есть также для GamaRegressor. Преимущество использования ARFF-файла перед такими файлами, как numpy-массив или csv-файл, заключается в том, что в нем указываются типы признаков. При передаче только массивов numpy (например, через fit(X, y)) GAMA не может знать, является ли конкретный признак номинальным или числовым. Это означает, что GAMA может использовать неправильное преобразование признака для данных (например, одноточечное кодирование для числового признака или масштабирование для категориального признака). Однако, это относится не только к GAMA, но и к любому фреймворку, принимающему числовые данные без метаданных.

Сравнение с TPOT

TPOT - пакет, наиболее тесно связанный с GAMA. Как и GAMA, он представляет собой пакет AutoML, основанный на генетическом программировании на языке Python. Ниже приведен список основных различий и сходств пакетов.

Отличается алгоритм эволюции. В TPOT используется синхронная форма эволюции, в то время как в GAMA - асинхронная (по умолчанию). Преимущество асинхронной эволюции заключается в том, что теоретически она позволяет лучше использовать вычислительные ресурсы. Это связано с тем, что эволюция на основе поколений в каждом поколении ожидает оценки последней особи в популяции, в то время как асинхронная эволюция не имеет точки, в которой все оценки должны быть завершены одновременно.

TPOT обладает рядом удобных функций, таких как менее ограниченный интерфейс командной строки и интеграция с DASK (что позволяет использовать кластер Dask для конвейерных оценок и дальнейших оптимизаций).

GAMA в большей степени *ориентирована на расширяемость и удобство для исследований*: в ней есть возможность конфигурировать конвейер AutoML (например, использовать другую стратегию поиска), шаг постобработки ансамблей, визуализация процесса оптимизации и поддержка файлов ARFF.

Стандартные возможности

CLI

Можно запускать из командной строки терминала полный список аргументов:

```
gama -h
```

Экспорт модели в виде кода

```
codestr = gama.Gama.export_script(f'gama_ASH_Asm{MAX_TIME}.py')
```

Основные методы и гиперпараметры

```
# создание объекта для поиска
gama = GamaRegressor(search=BaseSearch, max_total_time=MAX_TIME,
                     scoring=scoring,
                     random_state=RANDOM_STATE,
                     regularize_length=regularize_length,
                     # store="logs",
                     preset=preset,
                     n_jobs=JOBS, verbosity=verbose)

gama.fit(X_train, y_train)
gama.score(X_test, y_test)
predictions = gama.predict(X_test)

gama.export_script("gama_EA_sample.py")
```

Главные гиперпараметры оптимизации поиска:

scoring: string (по умолчанию = 'neg_log_loss' для классификации и 'mean_squared_error' для регрессии) Задает метрику для оптимизации. Убедитесь, что вы оптимизируете по той метрике, которая хорошо отражает то, что для вас важно. Принимается любая строка, которая может построить *scikit-learn scorer*. Возможные варианты: *roc_auc, accuracy* и *neg_log_loss* для классификации, *neg_mean_squared_error* и *r2* для регрессии.

regularize_length: bool (default=True) Если значение равно True, то помимо оптимизации по метрике, заданной в скоринге, поиск будет также направлен в сторону *более коротких* конвейеров. В настоящее время эта настройка не влияет на методы поиска, не используемые по умолчанию.

Базовые гиперпараметры управления ресурсами:

n_jobs: int, необязательно (по умолчанию=None) Определяет, сколько процессов может выполняться параллельно во время подгонки. Это в наибольшей степени влияет на то, сколько конвейеров машинного обучения может быть оценено. Если значение равно -1, то используются все ядра. Если установлено значение None (по умолчанию), то используется половина ядер. Изменение этого параметра на использование заданного количества (меньшего) ядер приведет к уменьшению количества оцениваемых конвейеров, но необходимо, если вы не хотите, чтобы GAMA использовала все ресурсы.

max_total_time: int (по умолчанию=3600) Максимальное время в секундах, которое GAMA должна стремиться использовать для построения модели из данных. По умолчанию GAMA использует один час. Для больших наборов данных может потребоваться больше времени для получения полезных результатов.

max_eval_time: int (по умолчанию=300) Максимальное время в секундах, которое GAMA может использовать для оценки одного конвейера машинного обучения. По умолчанию установлено значение 5 минут. Для больших наборов данных может потребоваться больше времени для получения полезных результатов.

Этапы подбора AutoML Pipeline

Алгоритмы поиска

Random Search Случайный поиск: Случайный выбор конвейеров машинного обучения из пространства поиска и их оценка.

Asynchronous Evolutionary Algorithm Асинхронный эволюционный алгоритм: Эволюция популяции конвейеров машинного обучения, отбор новых конвейеров машинного обучения из лучших в популяции.

[[Asynchronous Successive Halving Algorithm]] Асинхронный алгоритм последовательного перебора: Подход, основанный на бандите, при котором множество конвейеров машинного обучения итеративно оценивается и исключается на больших долях данных.

Post-processing (постобработка)

None: постобработка не производится. Это означает, что конечный конвейер не будет обучен, а функции predict и predict_proba будут недоступны. Это может быть интересно, если вас интересует только процедура поиска.

FitBest: подгонка единственного лучшего конвейера машинного обучения, найденного в ходе поиска.

Ensemble: создание ансамбля из оцененных конвейеров машинного обучения. Это требует больше времени, но может привести к лучшим результатам.

Дополнительные возможности и пользовательские настройки

Пространство поиска

По умолчанию GAMA строит конвейеры из алгоритмов *scikit-learn*, как для предварительной обработки, так и для обучения моделей. Это пространство поиска можно модифицировать, изменяя алгоритмы или диапазоны гиперпараметров, которые будут рассматриваться.

Пространство поиска определяется словарем search_space, передаваемым при инициализации. Установки по умолчанию содержатся в файлах classification.py и regression.py для GamaClassifier и GamaRegressor, соответственно.

Пример алгоритмов, которые GAMA использует по умолчанию:

- логистическая регрессия
- random forest classifier
- naive bayes
- support vector machines
- PCA - метод главных компонент
- normalization
- ICA - метод независимых компонент

Конфигурация пространства поиска задается в словаре python. Для справки, минимальный пример конфигурации пространства поиска может выглядеть следующим образом:

```
from sklearn.naive_bayes import BernoulliNB
search_space = {
    'alpha': [1e-3, 1e-2, 1e-1, 1., 10., 100.],
    BernoulliNB: {
        'alpha': [],
        'fit_prior': [True, False]
    }
}
```

Ключи в словаре могут быть:

строка со списком в качестве значения: Она задает имя гиперпараметра с его возможными значениями. Определив гиперпараметр на верхнем уровне, можно ссылаться на него как на гиперпараметр для любого конкретного алгоритма. Для этого необходимо определить его с тем же именем и установить его возможные значения в виде пустого списка (см. alpha в примере). Преимущество такого подхода заключается в том, что несколько алгоритмов могут совместно использовать гиперпараметрическое пространство, которое определяется только один раз. Кроме того, в процессе эволюции это позволяет узнать, какие значения гиперпараметров могут пересекаться между различными алгоритмами.

класс, в котором в качестве значения используется словарь: Ключ задает алгоритм, вызов которого должен инстанцировать алгоритм. В словаре указываются гиперпараметры по именам и их возможные значения в виде списка. Все указанные гиперпараметры должны быть приняты в качестве аргументов при инициализации алгоритма. Гиперпараметр, заданный на верхнем уровне словаря, может иметь общее имя с гиперпараметром алгоритма. Для использования значений, предоставляемых общим гиперпараметром, необходимо установить возможные значения в пустой список. Если вместо этого будет указан список значений, то он не будет использовать значения общего гиперпараметра.

Журнал

GAMA использует стандартный модуль протоколирования Python. Это означает, что журналы могут собираться на разных уровнях и обрабатываться одним из нескольких StreamHandlers.

Наиболее часто для ведения журнала используется запись полного журнала в файл, а также печать важных сообщений в stdout. Запись сообщений журнала в stdout поддерживается GAMA непосредственно через гиперпараметр verbosity (по умолчанию - logging.WARNING).

По умолчанию GAMA также сохраняет несколько различных журналов. Это можно отключить с помощью гиперпараметра store. Гиперпараметр store позволяет хранить журналы, а также модели и прогнозы. По умолчанию сохраняются журналы (в том числе и данные оценки), а модели и прогнозы отбрасываются.

Допустимые значения:

- DEBUG: Сообщения для разработчиков.
- INFO: Общая информация о процессе оптимизации.
- WARNING: Серьезные ошибки, не препятствующие завершению работы GAMA (но результаты могут быть неоптимальными).
- ERROR: Ошибки, которые не позволяют завершить работу GAMA.

События

Также можно программно получать данные в ходе процесса оптимизации через события:

```
from gama import GamaClassifier

def print_evaluation(evaluation):
    print(f'{evaluation.individual.pipeline_str()} was evaluated. Fitness is
{evaluation.score}.')

automl = GamaClassifier()
automl.evaluation_completed(print_evaluation)
automl.fit(X, y)
```

Функция, передаваемая в evaluation_completed, должна принимать в качестве единственного аргумента gama.genetic_programming.utilities.evaluation_library.Evaluation. Любые исключения, возникшие, но не обработанные в обратном вызове, будут проигнорированы, но занесены в журнал на уровне logging.WARNING. *Во время обратного вызова может быть поднят сигнал stopit.utils.TimeoutException. Этот сигнал обычно указывает GAMA на необходимость перехода к следующему шагу конвейера AutoML. Если он будет пойман обратным вызовом, то GAMA может превысить отведенное ей время. По этой причине рекомендуется делать короткие обратные вызовы после поимки stopit.utils.TimeoutException. Если stopit.utils.TimeoutException не будет пойман, то GAMA корректно завершит свой шаг в конвейере AutoML и продолжит работу в обычном режиме.*

Создание своих алгоритмов подбора и постобработки

Для реализации собственной процедуры поиска или постобработки требуется создать класс, производный от базового класса

Особенности генетического алгоритма

Построен на базе *компонентов*.

Components Определяют строительные блоки для Individuals. Индивидуумы представляют конвейеры машинного обучения, *не зависящие от бэкенда*. Индивидуум может быть преобразован в его представление, специфичное для бэкенда (например, в конвейер scikit-learn), путем вызова свойства pipeline, если для преобразования индивидуума в него была предоставлена функция.

Individuals создаются с помощью:

Терминалы. Определение конкретного значения для конкретного гиперпараметра. Неизменяемые.

Примитивы. Определение конкретного алгоритма. Неизменяемые. Определяется входом терминала, типом выхода и операцией.

PrimitiveNodes. Мутабельные для удобства выполнения операций (например, мутации). Инстанцированный примитив с определенными терминалами.

Fitness. Хранит информацию об оценке особи.

Mutation. Содержит функции мутации для генетического программирования. Каждая функция мутации берет особь и изменяет ее на месте.

Crossover. Функции, которые принимают два индивидуума и производят по крайней мере один новый индивидуум.