# Автоматическая генерация потоков работ с использованием мета-обучения и генетического программирования

Пендряк А.А., магистрант кафедры компьютерных технологий Университета ИТМО
pendryak@rain.ifmo.ru
Фильченков А.А., к.ф.-м.н., доцент кафедры компьютерных технологий Университета ИТМО
aaafil@mail.ru

## Аннотация

Количество данных, генерируемых людьми стремительно растет. Вместе с этим растет и число алгоритмов машинного обучения, использующих эти данные. Учитывая, что комбинации алгоритмов часто показывают лучшую эффективность, получаем огромное число вариантов комбинаций. Однако ручной поиск слишком ресурсозатратен и утомителен. Для решения этой проблемы исследователи начали создавать системы для автоматического построения последовательностей алгоритмов по заданному набору данных. В данной работе мы представляем фреймворк для оптимизации структуры потоков работ с целью увеличения эффективности задачи классификации, использующий подходы мета-обучения и генетического программирования.

## Введение

В последнее десятилетие количество данных, генерируемых людьми, заметно выросло. Вместе с увеличением количества данных, увеличивается и число различных утилит и библиотек для их обработки и анализа. Данные библиотеки содержат огромное множество различных алгоритмов. Но это одновременно является и их недостатком - при решении очередной задачи возникает проблема выбора подходящего алгоритма. А так как часто комбинация алгоритмов дает результаты лучше, чем отдельные алгоритмы, то мы получаем экспоненциальное число различных вариантов.

В настоящее время есть множество задач машинного обучения: кластеризация, регрессия, ранжирование и т.д. В этом исследовании мы

сфокусируемся на задаче классификации, так как она является наиболее частой и востребованной задачей в машинном обучении. Разработано огромное множество алгоритмов классификации, однако, ни один из них не является оптимальным алгоритмом для всех наборов данных [1], что вынуждает исследователей искать подходящие алгоритмы и их параметры для каждого набора данных.

В данной работе мы представляем подход для автоматической генерации потоков работ. Мы используем подход мета-обучения, для оценки эффективности алгоритмов для данного набора данных и используем эту оценку при выборе каждого алгоритма в потоке работа. Также мы используем генетическое программирование в качестве метода глобальной оптимизации при поиске оптимальной структуры потока работ и параметров каждого алгоритма.

## Метод

Поток работ для классификации — это направленный ациклический граф, узлами которого являются алгоритмы машинного обучения, а результат работы этих алгоритмов при подаче на вход набора данных - метка класса. То есть с точки зрения машинного обучения поток работ для классификации тоже является классификатором.

В нашем исследовании мы рассматриваем только линейные потоки работ, то есть такие потоки работ, где каждый узел имеет не более одного входящего ребра и не более одного выходящего. Узел, не имеющий входящего ребра, принимает данные, а узел, не имеющий выходящего ребра, возвращает результат. Пример линейного потока работ представлен на Рис. 1.

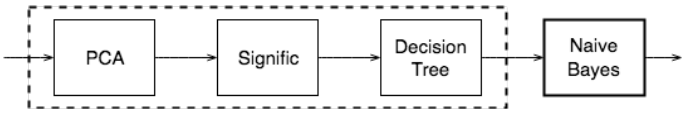


Рис. 1: Пример линейного потока работ

Алгоритмы, являющиеся узлами потока работ далее будем называть базовыми алгоритмами. Каждый базовый алгоритм принимает на вход набор данных и отдает преобразованные данные в следующий за ним узел. В данном исследовании базовые алгоритмы можно разделить на две группы: алгоритмы классификации и алгоритмы преобразования данных. Сам же линейный поток работ может быть представлен двумя компонентами: внутренней последовательностью базовых алгоритмов и финальным классификатором.

Обработка поданного набора данных происходит последовательно: сначала по очереди данные преобразуются узлами внутренней последовательности, а далее к преобразованным данным применяется финальный алгоритм классификации и его результат является результатом работы всего потока работ. Если очередной узел внутренней последовательности это алгоритм преобразования, то данный узел преобразует данные соответствующим для него образом. Если же очередной узел это классификатор, то результат классификации текущего набора данных добавляется в качестве нового атрибута для сохранения полученной информации.

Так как функция эффективности (точность, F-мера и т.д.) нетривиальным образом зависит от структуры потока работ, то для оптимизации данной функции можно применить любой метод оптимизации, не требующий каких-то определенных свойств от функции эффективности. Например, метод генетического программирования. В качестве особей в популяции будет выступать сам поток работ с вектором значений параметров алгоритмов, из которых он построен, а функция эффективность будет в данном случае функцией приспособления.

В данной работе используется схема $(1 + \lambda) - ES$, то есть на каждой итерации для потока работ с предыдущей итерации генерируется $\lambda$ потомков с помощью оператора мутаци, и на следщую итерацию переходит лучшая особь среди родительской особи и $\lambda$ потомков. Для уменьшения вероятности нахождения локального оптимума, производится несколько перезапусков и выбирается лучший результат. Описанная схема представлена в Листинге 1.

**Листинг 1:** Алгоритм построения потока работ для набора данных

```
Input: D — набор данных,
       N — количество итераций
Result: поток работ для поданного набора данных D

bestWorkflow ← null;
for i ← 1 to N do
    workflow ← optimize(generate(D), D);
    if score(workflow, D) > score(bestWorkflow, D) then
        bestWorkflow ← workflow;
return bestWorkflow;
```

### *Генерация потока работ*

Для работы эволюционных алгоритмов обычно требуется сначала инициализировать популяцию, а затем оптимизировать функцию приспособленности, применяя к особям генетические операторы. Мы предлагаем следующий способ построения потока работ: для случайной длины, выбранной из заданного заранее диапазона, и на основе текущего набора данных мы выбираем базовый алгоритм, изменяем набор данных, применяя этот алгоритм, как было описано ранее, и передаем измененный набор данных на следующую итерацию. На последнем шаге аналогичным образом выбираем финальный классификатор.

Так как пространство линейных потоков работ очень велико, то выбирая очередной алгоритм случайным образом, мы можем значительно увеличить время поиска оптимального решения. Но порождая потоки работ не случайно, а как можно ближе к оптимальному решению, можно значительно снизить количество итераций генетического алгоритма. Однако непосредственный подсчет эффективности алгоритмов ресурсоемкая задача. Поэтому необходимо найти другой способ оценки эффективности алгоритмов, требующий меньше вычислений.

Для решения подобной задачи можно применить подход мета-обучения, который главным образом основывается на предположении, что для похожих наборов данных алгоритмы работают похожим образом [2]. Обычно в задачах мета-обучения набор данных описывается при помощи фиксированного набора признаков, которые называются «мета-признаками» (meta feature), и «близость» между наборами данных рассчитывается как расстояние в соответствующем пространстве [3].

В данном исследовании для формирования пространства используется 14 мета-признаков из трех групп: общие, статистические, теоретико-информационные, — которые обычно используются в задачах мета-обучения [4].

Для оценки работы алгоритмов с помощью мета-обучения необходимо знать, как алгоритмы работают на некотором множестве данных. Для этого мы применили каждый алгоритм к 180 наборам данных из различных доменов, значительно различающихся числом объектов и числом признаков, чтобы наиболее полно покрыть пространство мета-признаков, и измерялась эффективность на каждом их них. Для классификаторов измерения производились непосредственно — подсчитывалась F-мера для каждого набора данных. Оценка эффективности алгоритмов преобразования измерялась как средняя величина F-меры по всем классификаторам, запущенных на преобразованных данных. Кос-

венный метод оценки обусловлен тем, что эффективность алгоритмов преобразования обычно оценивается в соответствии с эффективностью работы других алгоритмов на преобразованных данных.

Зная эффективность алгоритмов на некотором множестве набором данных, мы можем оценить их эффективность на поданном наборе данных следующим образом:

$$Perf(A)_D = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{dist(D, D_i)} Perf(A)_{D_i},$$

где $A$ — алгоритм, эффективность которого мы оцениваем, $D$ — поданный набор данных, $N$ — число ближайших рассматриваемых наборов данных, $D_i$ — $i$-й ближайший набор данных к $D$ в пространстве мета-признаков, $dist$ — функция расстояния между наборами данных в пространстве мета-признаков.

В качестве следующего алгоритма мы можем выбрать тот, у которого оценка эффективности наибольшая. Но тогда каждый перезапуск генетического алгоритма будет порождать потоки работ с одинаковым префиксом, что значительно уменьшает число исследованных элементов пространства решений. Для увеличения разнообразия решений будем выбирать следующий алгоритм случайным образом, отдавая предпочтение тем алгоритмам, у которых оценка эффективности больше.

Общая схема генерации потока работ по заданному набору данных представлена в Листинге 2.

**Листинг 2:** Алгоритм построения потока работ для набора данных

```
function generate(D)
    Input: D — набор данных
    Result: сгенерированный поток работ для набора данных D
    currentData ← D;
    workflow ← empty();
    for i ← 1 to random(⟨0, 1, ..., K − 1⟩) do
        algorithm ← next(classifiers + transformers, currentData);
        add(workflow, algorithm);
        currentData ← apply(algorithm, currentData);
    classifier ← next(classifiers, currentData);
    add(workflow, algorithm);
    return workflow;
```

### *Генетические операторы*

Для поиска оптимальной особи для эволюционных алгоритмов необходимо не только определить, как мы инициализируем популяцию, но и генетические операторы. Так как мы используем схему $(1+\lambda)-ES$, то достаточно определить операцию мутации. В нашем случае мутация потока работ это перестроение случайного суффикса цепочки алгоритмов таким же алгоритмом, что мы используем при генерации.

Общая схема поиска оптимального потока работ приведена в Листинге 3.

**Листинг 3:** Алгоритм оптимизации потока работ

```
function optimize(workflow, D)
    Input: workflow — оптимизируемый поток работ
           D — набор данных
    Result: новый поток работ для набора данных D
    parent ← workflow;
    for i ← 1 to M do
        bestChild ← null;
        for j ← 1 to λ do
            child ← mutate(parent);
            if score(child, D) > score(bestChild, D) then
                bestChild ← child;
        if score(bestChild, D) > score(parent, D) then
            parent ← bestChild;
    return parent;
```

## Результаты

Для сравнения эффективности предложенного метода мы выбрали несколько известных и хорошо работающих на многих классах данных классификаторов: Random Forest и SVM, а так же TPOT [5] — другой фреймворк для построения потоков работ для классификации. Результаты, представленные в Таблице 1, показывают, что предложенный метод генерирует потоки работ, эффективность которых сравнима с эффективностью популярных классификаторов или же превосходит ее.

| **Набор данных** | **Random Forest** | **SVM** | **TPOT** | **Workflow** |
|---|---|---|---|---|
| connect-4 | 0.5407 | 0.6146 | 0.5828 | **0.7309** |
| gina-agnostic | 0.9457 | 0.9377 | 0.9411 | **0.9619** |
| letter | 0.9229 | 0.949 | **0.973** | 0.956 |
| mfeat-zernike | 0.7864 | 0.8322 | 0.8418 | **0.884** |
| pendigits | 0.9862 | **0.994** | 0.9937 | **0.994** |
| spect | 0.7536 | 0.8029 | **0.8194** | 0.7742 |
| splice | **0.9696** | 0.9629 | 0.9606 | 0.9574 |
| sylva-agnostic | 0.9591 | **0.9717** | 0.9687 | 0.9708 |
| waveform-21 | 0.8554 | 0.8678 | 0.8631 | **0.8734** |

Таблица 1: Сравнение эффективности классификаторов на некоторых наборах данных

## Заключение

В данном исследовании рассмотрена проблема построения потока работ для классификации по заданному набору данных. Результаты показывают, что данный метод может автоматически генерировать потоки работ для классификации, эффективность которых сравнима или даже превосходит эффективность популярных алгоритмов классификации.

## Литература


[1] Duda, R.O., Hart, P.E., Stork, D.G.: Pattern Classification. J. Wiley & Sons, New York, 2001.

[2] Giraud-Carrier C., Keller J., Meta-learning // Dealing with the data flood: mining data, text and multimedia. 2002. C. 832—844.

[3] Vivalta R., Drissi Y. A perspective view and survey of meta-learning // Artificial Intelligence Review. 2002. Vol. 18. C. 77—95

[4] Castiello C., Castellano G., Fanelli A.M. Meta-data: Characterization of Input Features for Meta-learning // Modeling Decisions for Artificial Intelligence: Second International Conference. 2005. C. 457—468.

[5] Olson R.S., Bartley N., Urbanowicz R.J., Moore J.H.. Evaluation of a Tree-based Pipeline Optimization Tool for Automating Data Science // Proceedings of the Genetic and Evolutionary Computation Conference. 2016. C. 485–492.