Санкт-Петербургский государственный университет

Кафедра системного программирования

Группа 20.Б10-мм

Генетический алгоритм для настройки параметров алгоритмов ADAS

Матвеева Татьяна Павловна

Отчёт по учебной практике
в форме «Производственное задание»

Научный руководитель:
Ю.В. Литивнов

Консультант:
инженер-программист АО "Кама" М.С. Осечкина

Санкт-Петербург
2023

Оглавление

Введение

ADAS [16] (Advanced Driver-Assistance Systems) — системы помощи водителю, которые в настоящее время активно интегрируются в автомобили. Основной их задачей является ассистирование в сложных ситуациях, возникающих во время движения. Архитектура ADAS подразумевает наличие датчиков (лидары, радары, камера), процессора восприятия, где происходит распознавание полос движения, профиля дороги, пешеходов, знаков, автомобилей и основного контроллера, обрабатывающего информацию и принимающего решения относительно действий системы на основании результата работы процессора восприятия.

Данные, полученные с лидаров имеют формат облака точек и кластеризуются для создания автоматизированной разметки данных на кадре, которая, после минимальных правок человека, может быть использована как обучающая выборка для алгоритмов машинного обучения. Стоит заметить, что установка датчика типа «лидар» на автомобиль делает его дороже в несколько раз, поэтому наиболее популярным решением в данной области являются камеры. Как бы то ни было, алгоритмы ADAS требуют семантической разметки кадров, что на данный момент реализуется непосредственно во время движения автомобиля с помощью нейронных сетей. Для обучения модели такой нейронной сети необходимо создать тестовую выборку с верной семантической разметкой, для чего и могут быть использованы автоматически кластеризованные облака точек, отображенные на изображения камер.

Алгоритмы кластеризации точек лидара имеют ряд гиперпараметров, количество которых может быть очень велико. Оптимальные значения часто зависят от внешних факторов, например, погодные условия и условия видимости. Подбор гиперпараметров — трудоемкий процесс, требующий значительного количества времени разработчиков. Данная проблема может быть решена путем автоматизации процесса подбора. Один из самых перспективных подходов к решению задачи автоматизации — переформулировать ее в терминах генетического алгоритма [28],

использование которого мотивировано устойчивостью к выбору алгоритма кластеризации.

Генетический алгоритм представляет собой имитацию естественного отбора, где в качестве генов используются переменные для оптимизации. Одной из важных составляющих алгоритма является *функция приспособленности* — метрика для оценки кандидата (отдельного элемента популяции). Исходя из значений функции можно судить о корректности выбранных параметров, поэтому верный ее выбор является важным шагом при разработке алгоритма.

Существуют различные варианты реализации генетических алгоритмов, самые распространенные из которых перечислены в статье [30]. Алгоритм такого рода — популярное решение в областях, где требуется оптимизация или поиск экстремумов функций, и может быть приспособлен для решения прикладных задач путем варьирования отдельных частей, таких как способ выбора кандидатов для создания следующего поколения, количество поколений и др.

В рамках работы будут рассмотрены алгоритмы кластеризации данных, полученных с лидара из датасета a2d2 [1], который представляет собой набор снимков камер, установленных на автомобиле с соответствующими им облаками точек, и датасета SemanticKitty [29], который представлен последовательностью облаков точек с соответствующей им семантической разметкой.

1. Постановка задачи

Цель этой работы — разработка решения для автоматизации выбора параметров различных алгоритмов кластеризации облаков точек. Задача может быть разбита на следующие подзадачи:

- сделать обзор альтернативных решений для автоматического подбора гиперпараметров;
- сделать обзор существующих библиотек для реализации генетических алгоритмов, выбрать наиболее оптимальную;
- предложить подходящую функцию приспособленности, которая будет служить метрикой для оценки качества кластеризации;
- реализовать алгоритм подбора гиперпараметров самостоятельно или с использованием выбранной библиотеки;
- проверить результаты работы алгоритма на различных алгоритмах кластеризации. Оценивать результат предлагается с помощью референсной метрики IoU, а также с помощью визуализации.

2. Обзор

В этом разделе будет приведен обзор алгоритмов кластеризации, существующих решений задачи подбора параметров для машинного обучения и генетического алгоритма, так как задача подбора гиперпараметров довольно распространена в данной области.

2.1. Алгоритмы поиска плоскости и кластеризации

В этом разделе будут рассмотрены два наиболее известных алгоритма кластеризации DBSCAN и Kmeans, а также разработанный подход к выделению плоскости дороги.

2.1.1. Поиск плоскости

Одним из самых важных этапов работы систем ADAS является определение на входных данных проезжей части. Кластеризация дороги необходима для дальнейшего распознавания полос, а также для отслеживания местоположения транспортного средства относительно нее. Размер этого кластера слишком велик, поэтому параметры, подобранные для определения остальных объектов, могут не подходить для его детекции, кроме того, благодаря математической интерпретации в виде уравнения плоскости, проезжая часть может быть обнаружена специальным, более простым алгоритмом. Одним из возможных решений является алгоритм RANSAC (RANdom SAmple Consensus) [8], с помощью которого выделяются точки, принадлежащие дороге. Основным недостатком такого подхода к выделению дороги является случайная составляющая алгоритма. Из-за этой особенности RANSAC множество точек, принадлежащих дороге различно при каждом новом запуске алгоритма, а, следовательно, кластеризуемая область, зависящая от крайних точек определяется неоднозначно. Таким образом, невозможно дать точную оценку качеству кластеризации. Чтобы решить эту проблему, было предложено рассмотреть альтернативные подходы к выделению плоскости, такие как

1. поиск плоскости через SVD разложение

2. поиск плоскости дороги как самой нижней плоскости на кадре

Главной характеристикой плоскости является ее нормаль. После сравнения нормалей плоскостей, полученных альтернативными алгоритмами было замечено, что нормаль плоскости, полученной вторым способом ближе к нормали, полученной с помощью алгоритма RANSAC, поэтому было решено использовать в работе именно его. Кроме того, при визуальной оценке алгоритма на рис. 1 и 2 можно увидеть, что плоскость при подходе, использующем SVD, выделяется не полностью.

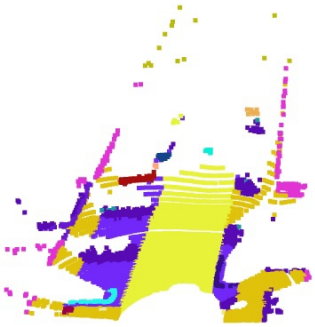

Рис. 1: SVD подход

Рис. 2: Наивный подход

Рассмотрим наивный подход к выделению плоскости, основанный на предположении, что плоскость дороги – множество точек, имеющих минимальное значение координаты z:

- выделение точек, лежащих на уровне точки с минимальной координатой по оси Oz как полосы заданной ширины
- построение уравнения плоскости по выделенным ранее точкам
- выбор точек, которые удовлетворяют полученному уравнению с заданным пороговым значением в качестве точек, принадлежащих плоскости

Таким образом, с помощью предложенного подхода удалось стабилизировать оценки качества работы алгоритма. Алгоритм в последствие был немного усложнен в связи с присутствием в данных шумовых точек, лежащих ниже плоскости дороги.

2.1.2. Кластеризация

Алгоритмы кластеризации в системах помощи водителю используются для распознавания объектов, непосредственно связанных с дорожным движением: пешеходов, транспортных средств и знаков. Кроме того, объекты окружающей среды могут не учитываться алгоритмом. Таким образом, объединение в один кластер двух объектов окружающей среды не считается ошибкой, а объединение объекта окружающей среды и участника дорожного движения — считается, что продемонстрировано на рис. 3, 4. Объектами для кластеризации являются облака точек, предоставленные в датасете [1].

Рис. 3: Автомобиль и знак в одном кластере

Рис. 4: Автомобиль и знак в разных кластерах

1. DBSCAN

   Алгоритм DBSCAN (Density-Based Spatial Clustering of Applications with Noise) [2] — плотностной алгоритм пространственной кластеризации с присутствием шума. Алгоритм удобен тем, работает без предварительного обучения. Сам алгоритм, а также его расширения, такие как hdbscan и FastDbscan широко используется во многих областях анализа данных. Все рассматриваемые точки в результате работы будут разделены на внутренние, граничные и шум. *Радиус окрестности* для определения границ кластера является параметром алгоритма и должен быть определён пользователем, так же как и *минимальное число точек* в окрестности рассматриваемого объекта, чтобы считать его точкой кластера.

2. Kmeans

   Kmeans [17] — итеративный алгоритм, который разделяет данные на K непересекающихся групп. Точка определяется в кластер таким образом, чтобы сумма квадратов расстояний между точками в кластере и центром была наименьшей. Гиперпараметрами алгоритма являются *количество кластеров*, *расположение центров* и *количество итераций*. В отличие от алгоритма DBSCAN, логика Kmeans основывается на определении расстояния между точками облака, а не на плотности их распределения. Хотя такой подход может приводить к неверному определению ленточных кластеров, как показано на рис. 5, иногда в литературе встречаются примеры [10], иллюстрирующие преимущества работы Kmeans над DBSCAN при кластеризации облаков точек.

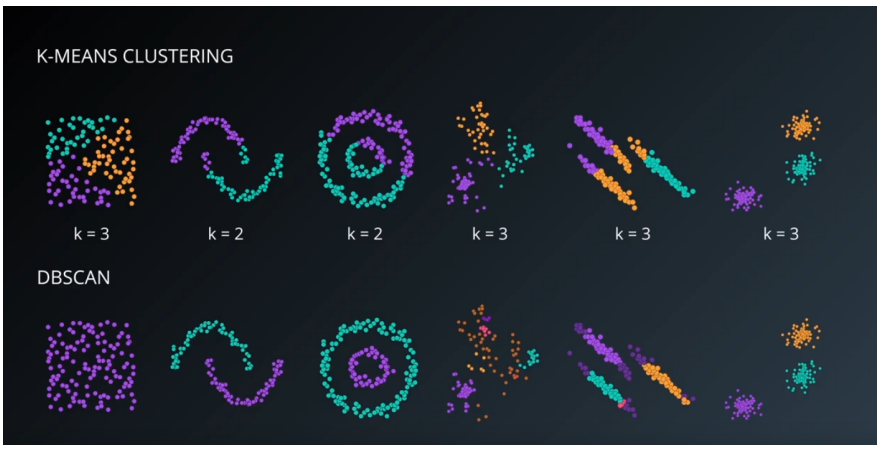

Рис. 5: Сравнение алгоритмов DBSCAN и Kmeans

2.2. Существующие решения подбора параметров

Выбор параметров — давно известная проблема, которая встала наиболее остро с развитием нейронных сетей и методов машинного обучения. Математически, оптимизация гиперпараметров — задача минимизации функции потерь. Все современные алгоритмы для решения данной задачи можно разделить на алгоритмы поиска и планировщики испытаний [31]. Рассмотрим самые популярные из них.

2.2.1. Поиск по решетке

Поиск по решетке — алгоритм полного перебора всех параметров на множестве, определенном пользователем. Данный метод является самым очевидным и самым точным способом решения рассматриваемой задачи, если множество определено пользователем корректно. Самым значительным недостатком является нерациональное расходование вычислительных ресурсов, которые растут экспоненциально с ростом размерности пространства. На практике алгоритм применяется только опытными пользователями, которые могут определить минимальное возможное множество параметров. По ним и будет осуществляться сходимость.

2.2.2. Случайный поиск

Базовым улучшением алгоритма поиска по решетке является случайный поиск. Как понятно из названия, алгоритм представляет собой поиск, где новые исследуемые значения выбираются из возможных вариантов случайным образом. В большинстве случаев эта модификация алгоритма лучше поиска по решетке. Несмотря на это, алгоритм все еще является вычислительно дорогим и требует опыта пользователя, чтобы сократить рассматриваемое множество. Часто используется как отправная точка для оценки новых разрабатывающихся алгоритмов.

2.2.3. Оптимизация методом Байеса и ее варианты

Оптимизация по Байесу [19] — алгоритм, разработанный в 1975 – 1978 годах, который стал популярным после применения к решению проблемы нахождения глобального оптимума. Ключевой особенностью алгоритма является способность принимать лучшее решение на основании всей полученной ранее информации, а также работа с апостериорными вероятностями и гауссовским процессом, подробно описанная в [22]. Основным преимуществом перед ранее рассмотренными методами является вычислительная эффективность, потому что каждые следующие рассматриваемые параметры выбираются неслучайно. Также пользователю совсем не требуется вручную ограничивать множество, по которому будет происходить сходимость.

Недостатком подхода является необходимость обучающей выборки, наличие которой не предусмотрено, в рамках поставленной задачи, а также высокое потребление вычислительных ресурсов, когда речь идет о глубоких нейронных сетях. В настоящее время алгоритм комбинируется с другими методами для лучшей производительности.

2.2.4. Генетический алгоритм

Генетический алгоритм — способ оптимизации, построенный на эмуляции эволюции, развивающейся по принципу "выживает сильнейший". Таким образом, лучшие, с точки зрения алгоритма, элементы становят-

ся наиболее приоритетными, в то время как менее успешные участники отбора со временем исчезают из рассмотрения.

Основными объектами алгоритма являются хромосомы — набор тестируемых параметров или генов, в терминах генетического алгоритма. Впоследствии каждая хромосома будет оценена с помощью функции приспособленности, и лучшие с точки зрения функции кандидаты будут выбраны, чтобы сформировать следующее поколение. Важным этапом является начальная реализация поколения — присваивание начальных параметров. От того, насколько удачной будет исходная выборка, зависит как быстро произойдёт сходимость.

Генетический алгоритм состоит из нескольких последовательных операций, представленных на рис. 6, которые повторяются в цикле некоторое число раз, зависящее от количества поколений:

1. сначала все кандидаты оцениваются функцией приспособленности, например в случае оптимизации параметров, можно взять функцию потерь или любую метрику, применимую в предметной области, и сортируются;

2. вторым этапом происходит отбор лучших хромосом для образования следующего поколения;

3. затем происходит кроссовер, в котором родительские хромосомы обмениваются генами, и мутация — случайное изменение в генах кандидатов;

Главным преимуществом генетического алгоритма является легкость адаптации под различные алгоритмы кластеризации и всевозможные наборы гиперпараметров, а также работа без тренировочной выборки.

2.3. Выбор инструментария

В этом разделе рассмотрены популярные готовые решения для поиска гиперпараметров. Выбор инструментов для рассмотрения основывался на статье [31], где приводится подробная характеристика каждого

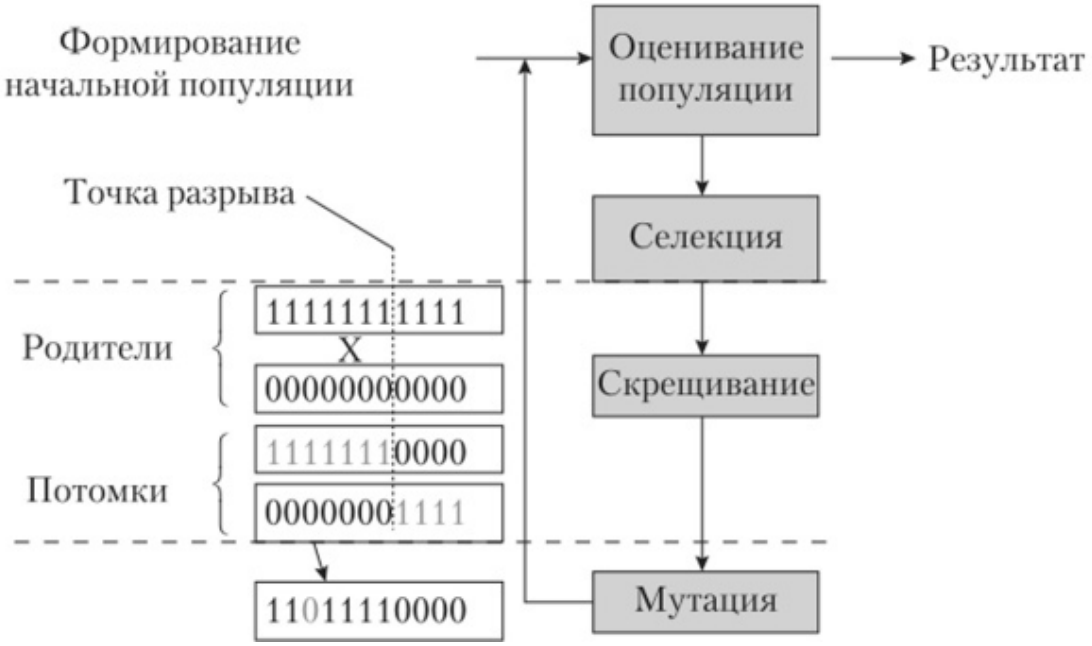

Рис. 6: Этапы генетического алгоритма

решения. При обзоре особое внимание уделялось доступности инструмента для использования в рамках некоммерческого проекта.

В условиях поставленной задачи не предполагается наличие тестовой выборки, так как ее разметка для задачи кластеризации — долгий процесс, чреватый внесением ошибок размечающего. Поэтому, несмотря на обилие существующих решений, посвященных подбору параметров для методов машинного обучения с помощью генетических алгоритмов [15] [11], их применение не представляется возможным.

2.3.1. Google Vizier

Google Vizier [13] — простой в использовании инструмент для подбора гиперпараметров, базовая конфигурация которого не требует глубоких знаний машинного обучения, но может быть изменена при желании пользователя. Реализует такие стратегии как поиск по решетке, случайный поиск и Байесовскую оптимизацию. Опытные пользователи также могут заменять предложенные алгоритмы на свои, что делает потенциально возможным внедрение генетического алгоритма. Проект является коммерческим, так что его использование не представляется

возможным в рамках данной работы.

2.3.2. Automatic model tuning in Amazon SageMaker

Инструмент [5] служит для упрощения процесса построения моделей. Поддерживает сложные модели и датасеты при помощи распараллеливания. Поддерживаются такие стратегии как Случайный поиск и Байесовская оптимизация, тем не менее, так же как и при использовании Google Vizier, опытные пользователи могут модифицировать предложенные алгоритмы. Код проекта также не является открытым.

2.3.3. Neural Network Intelligence

NNI [18] — набор инструментов для оптимизации гиперпараметров с открытым кодом. В отличие от перечисленных выше закрытых проектов, он предоставляет не только основу для тренировки модели и настройки гиперпараметров, но и большую свободу для кастомизации. NNI может быть внедрен как в автономные устройства, так и в удаленные серверы. Основным недостатком является то, что многие инструменты разработаны для конкретных алгоритмов или особых задач (например, распознавания образов). Существует возможность добавления к уже существующим решениям реализации генетического алгоритма, однако инструмент NNI требует наличие обучающей выборки, что противоречит условиям поставленной задачи.

2.3.4. Ray.Tune

Ray.Tune [23] — библиотека, представляющая собой распределенный фреймворк для тренировки моделей, большинство методов которой может быть оптимизировано с помощью распараллеливания. Значительными недостатками являются отсутствие удобного пользовательского интерфейса, высокий порог входа и значительные ограничения в используемых ресурсах по сравнению с закрытыми аналогами. После тщательного изучения достоинств и недостатков подхода, было принято решение отказаться от использования рассматриваемого инструмента.

Таким образом, ни один из рассмотренных выше решений не удовлетворяет необходимым требованиям в полной мере, что делает невозможным их использование для реализации поставленной задачи.

2.4. Используемые технологии

Написание собственного решения требует выбора языка программирования, а также обзора его библиотек, на базе которых может быть осуществлено внедрение генетического алгоритма в процесс выбора гиперпараметров.

Так как в документации к выбранному датасету приведены некоторые необходимые методы обработки входных данных на языке Python, а работа с облаками точек в нем облегчается методами библиотеки open3D [20], именно он был выбран как основной язык для написания исходного кода.

В Python существует множество инструментов, позволяющих реализовать генетический алгоритм. Чтобы сделать выбор в пользу оптимального решения был проведен обзор готовых пакетов, основные сведения о которых представлены в таблице 1. Основными требованиями к инструменту были:

- подробная документация;
- возможность задания пользователем функции приспособленности.

Как видно из таблицы, в некоторых библиотеках накладываются ограничения на функцию приспособленности, такие как характер оптимизации функции (минимизации, максимизации), определенное число аргументов и тд. Кроме того, некоторые библиотеки не предоставляют реализацию генетического алгоритма, лишь отдельные его части. Это делает невозможным использование рассмотренных Python-пакетов для решения поставленной задачи. Таким образом, EasyGa и GeneAI были выбраны как две наиболее перспективные библиотеки для реализации. Из-за простоты кода и большего количества доступных ма-

Название библиотеки	Функции приспособленности	Документация
DEAP [6]	нет	есть
pyeasyga [35]	есть ограничения	есть
PyGAD [21]	есть ограничения	есть
EasyGA [9]	есть	есть
geneticalgorithm [32]	есть ограничения	есть
geneticalgorithm2 [33]	есть ограничения	есть
GeneAl [12]	есть	есть
pyStep [3]	нет	не актуальна
geneticalgs [34]	нет информации	нет

Таблица 1: Сравнительный анализ существующих инструментов

териалов и примеров использования, в качестве оптимального варианта была выбрана библиотека EasyGa.

2.5. Метрики, используемые в предметной области

Метрики в области кластеризации можно разделить на те, для которых существуют априори верные данные, и те, что работают автономно. Так как в предметной области обучение происходит без обучающей выборки, в этом разделе рассмотрены метрики второго типа [36].

2.5.1. Silhouette score

Коэффициент Silhouette [27] определяется для каждого кластера и состоит из следующих компонент: a — среднее расстояние между выбранной точкой и всеми остальными точками в классе и b — среднее расстояние между выбранной точкой и всеми остальными точками следующего ближайшего кластера, тогда коэффициент Silhouette определяется следующим образом:

$$s = \frac{b - a}{\max(a, b)}$$

Коэффициент определяется как среднее среди всех значений Silhouette для каждого рассматриваемого класса и может изменяться от -1 до 1. Значения, близкие к 1, означают хорошую кластеризацию, а коэффици-

енты для наименее удачных вариантов кластеризации будут стремиться к -1. Значительным недостатком метрики является вычислительная сложность.

2.5.2. Calinski-Harabasz Index

Другим способом оценки кластеризации является индекс Calinski-Harabasz [25], который, так же как коэффициент Silhouette, максимален на разреженных и хорошо разделенных кластерах, что отвечает стандартному концепту кластера. Математически определяется как

$$s = \frac{tr(B_k)}{tr(W_k)} \times \frac{n_E - k}{k - 1}$$

где n_E — размер кластеризуемого датасета, k — число полученных кластеров, $tr(B_k)$ — след матрицы дисперсии между кластерами, а $tr(W_k)$ — след матрицы дисперсии внутри кластеров.

2.5.3. Davies-Bouldin Index

Индекс Davies Bouldin [26] сравнивает расстояние между кластерами с размером самих кластеров. В отличие от предыдущих метрик, показывает значения, близкие нулю для более удачных вариантов кластеризации. Определим функцию схожести между кластерами i и j как

$$R_{i,j} = \frac{s_i + s_j}{d_{i,j}}$$

где за s_i обозначается среднее расстояние от точки кластера до его центра, а $d_{i,j}$ является расстоянием между центрами кластеров i и j. В таком случае метрика определяется формулой

$$DB = \frac{1}{k} \sum_{i=1}^{k} \max_{i \neq j} R_{i,j}$$

Индекс может быть использован только для евклидовых пространств.

2.5.4. CDbw

Метрика CDbw учитывает все критерии «хорошей» кластеризации, среди которых *Плотность* данных внутри кластера, *Разделение* различных кластеров, а также *Компактность*. Формулы для каждого критерия могут быть найдены в статьях [7], [14], общая формула для коэффициента выглядит следующим образом:

$$CDbw = \text{Плотность} \cdot \text{Разделение} \cdot \text{Компактность}$$

Коэффициент является функцией максимизации.

3. Реализация

В данном разделе подробно рассмотрен разработанный алгоритм. Особое внимание уделено выбору функции приспособленности и предварительной обработке входных данных. Также в разделе рассмотрены задачи, которые были решены в процессе работы с помощью подробного анализа используемых алгоритмов.

3.1. Алгоритм

Процесс нахождения оптимальных параметров состоит из следующих шагов:

1. Добавление соответствующего алгоритма кластеризации и его параметров в конфигурационный файл.
2. Выбор диапазона кадров, на котором будет проведена оптимизация.
3. Распознавание плоскости дороги от остальных объектов на облаке точек с помощью разработанного алгоритма.
4. Предобработка входных данных для улучшения сходимости к оптимальным параметрам.
5. Создание начальной популяции хромосом. Гены принимают случайные значения для числовых параметров и значения из заданного диапазона для строковых.
6. Запуск генетического алгоритма (число поколений и количество хромосом — гиперпараметры решения).
7. Опциональная оценка результатов работы алгоритмов кластеризации с помощью описанной далее метрики IoU.

Для улучшения качества подобранных параметров рекомендуется запустить алгоритм несколько раз. Индикатором успешного процесса

подбора является многократная сходимость алгоритма к близким между собой значениям.

3.2. Функция приспособленности

В качестве функций приспособленности использовались как рассмотренные выше метрики, существующие в предметной области, так и самостоятельно сконструированные коэффициенты для измерения качества кластеризации.

1. Функция, основанная на существующих метриках

   Рассмотрим функцию приспособленности на основе коэффициента Silhouette.

   Пусть n — количество кадров, по которому необходимо провести оптимизацию, $\alpha_1, \ldots, \alpha_m$ — параметры алгоритма кластеризации. Каждому кадру соответствует облако точек, которое можно обозначить как pcd_i, где $i = 1, \ldots n$. Количество кластеров обозначим как cl, max и min — максимальное и минимальное возможное число кластеров на кадре соответственно. Ограничения обусловлены тем, что метрика показывает наилучшие значения для наименьшего количества кластеров, что потенциально опасно определением всего облака в один кластер. Тогда функция приспособленности для i-го кадра может быть вычислена по следующей формуле:

   $$f_i(\alpha_1, \ldots, \alpha_m) = \begin{cases} silhouette(\alpha_1, \ldots, \alpha_m), & \min < cl < \max \\ -1, & \text{иначе} \end{cases}$$

   Важно заметить, что выбранная метрика принимает значения в пределах $[-1, 1]$, где 1 — соответствует наилучшему разделению на классы.

   Для всех кадров функция вычисляется как:

   $$f(\alpha_1, \ldots, \alpha_m) = \frac{\sum_{i=1}^{n} f_i(\alpha_1, \ldots, \alpha_m)}{n}$$

Таким же образом можно определить функцию, работающую на основе любой другой существующей метрики, заменив ей коэффициент Silhouette.

2. Мудрость толпы

   Метрика основана на принципе "мудрость толпы" [4]. Расстановка коэффициентов перед значениями метрик вызвана необходимостью привести их к одному типу оптимизации – максимизации или минимизации, а также сделать так, чтобы области определения лежали в близких диапазонах, таким образом, область значений функции Silhouette не изменится и будет представлять собой закрытый интервал $[-1, 1]$. Davies-Bouldin изначально является функцией минимизации, неограниченной сверху, поэтому после взятия к ней обратного, он станет функцией максимизации. Функция Calinski-Harabasz уже является функцией максимизации, но так как ее значения велики по сравнению с остальными коэффициентами, имеет смысл так же как в предыдущем случае взять обратную к ней величину, а затем поменять знак, чтобы функция осталась функцией максимизации. Лучшее значение при таком отображении будет 0. Таким образом, полученная метрика является функцией максимизации и учитывает результаты работы трех существующих внутренних метрик.

$$CrowdWisdom(X) = S(X) + \frac{1}{DB(X)} - \frac{1}{CH(X)}$$

   где X — метки, полученные в результате кластеризации, S – метрика Silhouette, DB – метрика Davies-Bouldin, CH – метрика Calinski-Harabasz.

3. Filter

   Идея метрики состоит в том, что кластеризация оценивается последовательно тремя метриками. До оценки следующей метрикой допускаются лишь те особи, чей показатель удовлетворил мини-

мальному допустимому значению, установленному для предыдущей метрики.

В настоящий момент основная проблема при работе с функциями приспособленности состоит в том, что необходимо искусственно ограничивать количество кластеров на кадре, что чревато выбором неверных параметров для кадров, количество кластеров на которых мало или велико. Одним из возможных решений этой проблемы может стать использование семантических признаков, извлеченных из облака точек. Подобный процесс описан в статье [24] и может быть применен в процессе развития проекта.

3.3. Предобработка данных

Для улучшения сходимости алгоритма к наилучшим параметрам по функции приспособленности было принято решение модифицировать входные данные в формате облаков точек.

1. Audi

   - определение области, находящейся непосредственно над предполагаемой дорогой (пример такого преобразования показан на рис. 7, 8).

2. SemanticKitti

   В случае работы с датасетом SemanticKitti кроме предыдущего пункта необходимо также:

   - выделение области, находящейся перед лидаром. Это вызвано с тем, что в случае достижения автомобилем перекрестка невозможно определить область, находящуюся над дорогой
   - равномерное уменьшение количества точек в облаке

Благодаря преобразованию первого типа шум, возникающий при кластеризации объектов вне проезжей части, не будет препятствовать

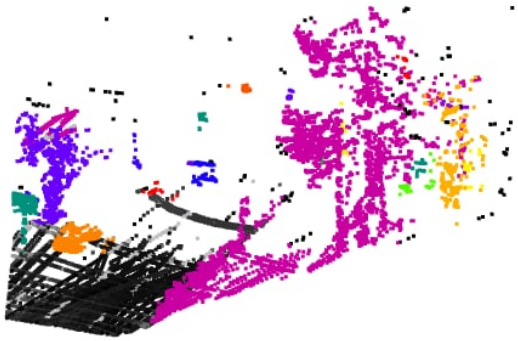

Рис. 7: Облако до преобразования

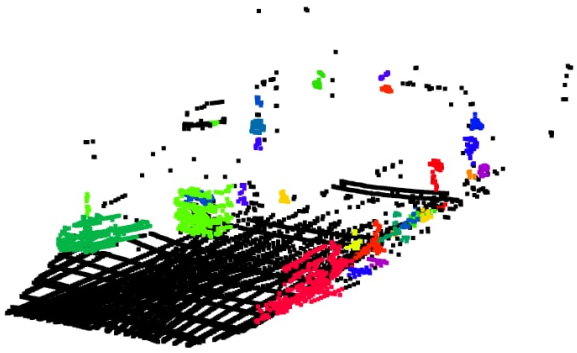

Рис. 8: Облако после преобразования

вычислению верной оценки кадра, а уменьшение количества точек увеличит скорость работы алгоритма.

4. Оценка качества работы

Для автоматической оценки подобранных алгоритмом параметров, было решено ввести специальную референсную метрику.

4.1. Коэффициент IoU

IoU является отношением пересечения к объединению. Коэффициент является популярной референсной метрикой качества работы алгоритмов кластеризации и детекции.

- для каждого кластера из облака, кластеризованного с помощью алгоритма, вычисляется соответствующий кластер в облаке с правильной разметкой
- для кластера считается коэффициент IoU

$$IoU = \frac{N_I}{N_U}$$

, где N_I – количество элементов в пересечении множеств точек рассматриваемого кластера на разметке датасета $SemanticKitty$, и на разметке, полученной с помощью алгоритма кластеризации. N_U – количество элементов, в объединении вышеупомянутых множеств точек.

Процесс ассоциации кластеров состоит из следующих этапов:

1. выделение индексов, соответствующих меток истинного кластера;
2. выделение меток алгоритма кластеризации, находящихся на индексах из предыдущего пункта;
3. вычисление соответствующего кластера;

В процессе работы были рассмотрены два подхода к выполнению последнего пункта алгоритма:

1. Выбор кластера по наибольшему количеству меток, присутствующих в области

2. Выбор кластера по наибольшему значению IoU

Так как второй подход позволил избежать ошибочной ассоциации в случае перекрытия большого кластера маленькими, в окончательном варианте алгоритма был использован именно он. Стоит отметить, что метрика учитывает только кластеры, относящиеся к интересным с точки зрения дорожного движения объектам, которые специфицируются с помощью семантической разметки датасета SemanticKitty [29]. Метрика для каждого облака считается как среднее по всем кластерам. Значение метрики для последовательности кадров считается как среднее значение по последовательности. На рисунках 9 и 10 можно зависимость IoU от подобранных параметров кластеризации.

Рис. 9: IOU = 0.89

Рис. 10: IOU = 0.23

Также, при подстановке метрики IoU в генетический алгоритм в качестве функции приспособленности, могут быть получены примерные оценки на лучшие возможные результаты.

В случаях, когда для датасета не предусмотрена верная разметка, оценка подобранных гиперпараметров происходит визуально.

4.2. Эксперименты

Эксперименты проводились на последовательностях датасета [29] посредствам оценки подобранных параметров с помощью метрики IoU.

Алгоритм Kmeans

Метрики	**Параметры**	**IoU**
Silhouette	(0.66, 14)	0.7499
Calinski-Harabaz	(0.3429, 1)	0.8402
Davies-Bouldin	(0.7469, 1)	0.8062
CrowdWisdom	(0.638, 1)	0.855

Таблица 2: Результаты на рассмотренных метриках

Метрики	**Параметры**	**IoU**
Silhouette	(21, 327, 10)	0.617
Calinski-Harabaz	(30, 305, 14)	0.297
Davies-Bouldin	(99, 396, 14)	0.788
CrowdWisdom	(30, 341, 9)	0.771

Таблица 3: Результаты на рассмотренных метриках

5. Заключение

В результате проделанной работы были решены следующие задачи:

- проведен обзор альтернативных решений для автоматического подбора гиперпараметров;
- рассмотрены библиотеки для реализации генетического алгоритма и выбрана оптимальная для данной работы;
- предложена функция приспособленности, основанная на метриках, существующих в предметной области;
- реализовано решение, позволяющее внедрить генетический алгоритм в процесс выбора гиперпараметров;
- разработан алгоритм выделения плоскости на кадре;
- предложена референсная метрика IoU для оценки результатов работы генетического алгоритма;
- алгоритм приспособлен к работе с различными алгоритмами кластеризации с последующей оценкой на датасете SemanticKitti

Возможности развития проекта:

- усовершенствование функции приспособленности на основе извлечения семантической информации;

Исходный код доступен по ссылке
https://github.com/osechkina-masha/adas_spbu/tree/GA_Matveeva.

Список литературы

[1] A2D2: Audi Autonomous Driving Dataset / Jakob Geyer, Yohannes Kassahun, Mentar Mahmudi et al. — 2020. — 2004.06320.

[2] ADBSCAN: Adaptive Density-Based Spatial Clustering of Applications with Noise for Identifying Clusters with Varying Densities / Mohammad Mahmudur Rahman Khan, Md. Abu Siddique, Rezoana Arif, Mahjabin Oishe. — 2018. — 09.

[3] API Genetic Programming. A light Genetic Programming API that allows the user to easily evolve populations of trees with precise grammatical and structural constraints // Documentation. — URL: https://pypi.org/project/pySTEP/.

[4] Alizadeh Hosein, Yousefnezhad Muhammad, Minaei Behrouz. Wisdom of Crowds Cluster Ensemble // Intelligent Data Analysis. — 2015. — 06. — Vol. 19.

[5] Amazon. Amazon SageMaker // Documentation. — URL: https://aws.amazon.com/ru/sagemaker/resources/?ar-cards-sagemaker.sort-by=item.additionalFields.datePublished&ar-cards-sagemaker.sort-order=desc.

[6] DEAP. DEAP is a novel evolutionary computation framework for rapid prototyping and testing of ideas. // Documentation. — URL: https://deap.readthedocs.io/en/master/.

[7] Density-Based Clustering Validation / Davoud Moulavi, Pablo Andretta Jaskowiak, Ricardo Campello et al. — 2014. — 04.

[8] Derpanis Konstantinos G. Overview of the RANSAC Algorithm // Image Rochester NY. — 2010. — Vol. 4, no. 1. — P. 2–3.

[9] EasyGa. EasyGA is a python package designed to provide an easy-to-use Genetic Algorithm // Repository. — URL: https://github.com/danielwilczak101/EasyGA.

[10] Florent Poux Ph.D. 3D Point Cloud Clustering Tutorial with K-means and Python // Towards Data Science. — URL: https://towardsdatascience.com/3d-point-cloud-clustering-tutorial-with-k-means-and-python-c870

[11] Franz David Krüger Mohamad Nabeel. Hyperparameter Tuning Using Genetic Algorithms. A study of genetic algorithms impact and performance for optimization of ML algorithms. — MID SWEDEN UNIVERSITY.

[12] GeneAl. s a python library implementing genetic algorithms (GAs). It has functionality for both binary and continuous GA, as well as specific use case applications such as a solver for the Travelling Salesman Problem // Documentation. — URL: https://github.com/diogomatoschaves/geneal.

[13] Google. Open Source Vizier: Reliable and Flexible Black-Box Optimization. // Documentation. — URL: https://cloud.google.com/ai-platform/optimizer/docs/overview (online; accessed: 14.12.2022).

[14] Halkidi Maria, Vazirgiannis Michalis. A density-based cluster validity approach using multi-representatives // Pattern Recognition Letters. — 2008. — 04. — Vol. 29. — P. 773–786.

[15] Hyper-parameter Tuning using Genetic Algorithms for Software Effort Estimation / Leonardo Villalobos-Arias, Christian Quesada-López, Marcelo Jenkins, Juan Murillo-Morera // 2021 16th Iberian Conference on Information Systems and Technologies (CISTI). — 2021. — P. 1–6.

[16] Jayan Keerthi, Muruganantham B. Advanced Driver Assistance System Technologies and Its Challenges Toward the Development of Autonomous Vehicle // Intelligent Computing and Applications / Ed. by Subhransu Sekhar Dash, Swagatam Das, Bijaya Ketan Panigrahi. — Singapore : Springer Singapore, 2021. — P. 55–72.

[17] Li Youguo, Wu Haiyan. A Clustering Method Based on K-Means Algorithm // Physics Procedia. — 2012. — Vol. 25. — P. 1104–1109. — International Conference on Solid State Devices and Materials Science, April 1-2, 2012, Macao. URL: https://www.sciencedirect.com/science/article/pii/S1875389212006220.

[18] Microsoft. An open source AutoML toolkit for hyperparameter optimization, neural architecture search, model compression and feature engineering. // Repository. — URL: https://nni.readthedocs.io/en/stable/.

[19] Nguyen Vu. Bayesian Optimization for Accelerating Hyper-Parameter Tuning // 2019 IEEE Second International Conference on Artificial Intelligence and Knowledge Engineering (AIKE). — 2019. — P. 302–305.

[20] Open3D. A Modern Library for 3D Data Processing // Documentation. — URL: http://www.open3d.org/docs/release/.

[21] PyGAD. PyGAD is an open-source Python library for building the genetic algorithm and optimizing machine learning algorithms. It works with Keras and PyTorch. // Documentation. — URL: https://pygad.readthedocs.io/en/latest/.

[22] Rasmussen Carl Edward, Williams Christopher K. I. Gaussian processes for machine learning. Adaptive computation and machine learning. — MIT Press, 2006. — P. I–XVIII, 1–248. — ISBN: 026218253X.

[23] Ray.Tune. Tune: Scalable Hyperparameter Tuning // Repository. — URL: https://docs.ray.io/en/latest/tune/index.htmll.

[24] SHREC 2020: 3D point cloud semantic segmentation for street scenes / Tao Ku, Remco C. Veltkamp, Bas Boom et al. // Computers Graphics. — 2020. — Vol. 93. — P. 13–24. — URL: https://www.sciencedirect.com/science/article/pii/S0097849320301400.

[25] Scikit Metrics Calinski-harabasz score // Documentation. — URL: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.silhouette_score.html.

[26] Scikit Metrics Davies-Bouldin score. Документация // Documentation. — URL: https://scikit-learn.org/stable/modules/clustering.html#davies-bouldin-index.

[27] Scikit Metrics Silhouette. Silhouette Coefficient // Documentation. — URL: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.silhouette_score.html.

[28] Thavasimani Karthikayini, Srinath N. Optimal Hyper-Parameter Tuning using Custom Genetic Algorithm on // Journal of Engineering Science and Technology. — 2022. — 04. — Vol. 17. — P. 1532–1549.

[29] Towards 3D LiDAR-based semantic scene understanding of 3D point cloud sequences: The SemanticKITTI Dataset / J. Behley, M. Garbade, A. Milioto et al. // The International Journal on Robotics Research. — 2021. — Vol. 40, no. 8-9. — P. 959–967.

[30] Variations of Genetic Algorithms / Alison Jenkins, Vinika Gupta, Alexis Myrick, Mary Lenoir. — 2019. — 11.

[31] Yu Tong, Zhu Hong. Hyper-Parameter Optimization: A Review of Algorithms and Applications // ArXiv. — 2020. — Vol. abs/2003.05689.

[32] geneticalgorithm. An easy implementation of genetic-algorithm (GA) to solve continuous and combinatorial optimization problems with real, integer, and mixed variables in Python // Documentation. — URL: https://pypi.org/project/geneticalgorithm/.

[33] geneticalgorithm2. Supported highly optimized and flexible genetic algorithm package for python // Documentation. — URL: https://pypi.org/project/geneticalgorithm2/.

[34] geneticalgs. Implementation of standard, diffusion and migration models of genetic algorithms. // Documentation. — URL: https://pypi.org/project/geneticalgs/.

[35] pyeasyga. pyeasyga. — URL: https://pypi.org/project/pyeasyga/.

[36] scikit User Guide. Simple and efficient tools for predictive data analysis // Documentation. — URL: https://https://scikit-learn.org/stable/user_guide.html (online; accessed: 14.12.2022).