Сравнение алгоритмов машинного обучения для предсказания времени работы пользовательских заданий в рамках оптимизации использования ресурсов суперкомпьютерного кластера МСЦ РАН

[1]А.А. Рыбаков, [2]С.С. Шумилин

МСЦ РАН – филиал ФГУ ФНЦ НИИСИ РАН, Москва, Россия

E-mails: [1]rybakov@jscc.ru,[2]shumilin@jscc.ru

Аннотация. Оценка ресурсов, необходимых для выполнения программы на суперкомпьютерном кластере, является трудновыполнимой задачей. Пользователю необходимо практически угадывать время работы программы, что приводит к тому, что некоторые задания принудительно завершаются планировщиком из-за превышения лимита по времени. Большое число подобных событий приводит к тому, что распределение ресурсов кластера теряет эффективность. Для того, чтобы оптимизировать распределение ресурсов, необходимо, чтобы планировщик получал более точные оценки параметров задания. Ранее для решения этой проблемы были предприняты попытки использовать алгоритмы машинного обучения. В данной статье рассмотрено применение таких алгоритмов к данным суперкомпьютерного кластера МСЦ РАН. Получены результаты по качеству регрессионных моделей на различных дополнительных признаках, среди которых проведен отбор наиболее информативных. Для отбора признаков использованы различные техники, широко используемые в машинном обучении. Данные признаки, которые характеризуют пользователя кластера, могут быть использованы для повышения точности уже полученных моделей.

Ключевые слова. HPC, машинное обучение, предсказательная аналитика, планировщик заданий, анализ поведения пользователя.

Введение

При формировании задания пользователь кластера МСЦ РАН должен указать время работы и количество процессорных ядер. Оценить время выполнения, основываясь на коде программы, очень сложно. Поэтому пользователи зачастую указывают приближенные оценки времени выполнения. Также следует отметить, что пользователи кластера МСЦ РАН в основном являются представителями сфер, не связанных с программированием, и поэтому было бы неправильно возлагать на них дополнительные задачи по оценке времени выполнения программы.

На основе указанных параметров в применяемой в МСЦ РАН СУППЗ [1] (Система управления прохождением параллельных заданий) формируется паспорт задания, вместе с которым задание попадает в очередь. Если задание завершается раньше указанного пользователем времени, то это вредит системе планирования, так как если бы точное время выполнения задания было известно изначально, то ресурсы были бы распределены более эффективно. Поэтому ставится задача предсказания времени работы задания по указанным характеристикам.

То есть необходимо, чтобы на вход планировщика приходила не пользовательская оценка времени выполнения, а время, рассчитанное моделью (рис. 1).

Предсказание времени выполнения задания представляет собой задачу, основанную на представлении паспорта задания как входного набора признаков для алгоритмов машинного обучения (МО). Главная идея заключается в том, что каждому пользователю свойственны определенные шаблоны задач. То есть высока вероятность, что определенный пользователь постоянно запускает задачи, схожие по площади, или заказывает определенное время работы. То есть в целом его задачи имеют тенденцию отрабатывать за определенное время. Также высока вероятность, что пользователи из схожих

групп (проект, организация, сфера деятельности) имеют схожие статистические метрики. Данная задача называется задачей пользовательского моделирования (user modeling) и представляет собой использование поведенческих характеристик пользователя для предсказывания его дальнейших действий [2, 3]. В нашем случае мы совмещаем признаки пользователя и задания и получаем модель, отражающую опыт пользователя.

Критическим остается вопрос выбора признаков для модели. В дополнение к исходным признакам в работе формируются новые: временные и персональные. Далее проводится отбор наиболее информативных признаков.

В данной работе приведены результаты применения алгоритмов машинного обучения для решения задачи *регрессии* для предсказания времени выполнения задания. Рассмотрены несколько моделей, которые относятся к различным классам алгоритмов. Основная метрика, используемая для оценки качества алгоритмов, это скорректированный коэффициент детерминации R^2_{adj} . На основе данной метрики выбраны наиболее качественные модели и произведено их сравнение с базовым алгоритмом, использующим скользящее среднее для оценки времени выполнения задания.

Обучение моделей проводилось на данных суперкомпьютера МВС-10П «Торнадо», являющегося одной из вычислительных систем МСЦ РАН. Данные представляют собой информацию об использовании ресурсов за 2013–2017 гг. В исходную таблицу входят 110 тыс. строк, каждая из которых представляет собой информацию об уникальном задании, в том числе данные указанные пользователем при регистрации задания.

Таким образом, данная работа привносит следующий вклад:

- Исследование данных МСЦ РАН.
- Выбор наиболее информативных признаков для обучения моделей.
- Сравнения с простым алгоритмом, использующим скользящее среднее.

Рис. 1 –Взаимодействие модели и планировщика заданий.

Машинное обучение для предсказания времени работы задания

Попытки использования алгоритмов машинного обучения для работы с данными суперкомпьютерного кластера для оптимизации работы планировщика проводились еще два десятилетия назад [4-6].

В работе [7] авторы приводят результаты построения моделей на данных суперкомпьютера. Чтобы определить количество используемых ресурсов (количество CPU и RAM) авторы используют алгоритмы регрессии. Для того чтобы определить, будет ли задание принудительно завершено планировщиком, авторы используют алгоритмы бинарной классификации.

В работе [8] авторы также используют регрессионные алгоритмы для предсказания количества заказанных процессоров. Для того, чтобы предсказать, будет ли задание завершено планировщиком принудительно, используются алгоритмы классификации. Авторы приходят к выводу, что при добавлении в обучающую выборку времени, запрошенного пользователем, качество регрессии значительно увеличивается (R^2), а качество классификации (F-мера) изменяется незначительно.

Авторы [9] описывают модель, которая предсказывает количество необходимой памяти для выполнения программы.

Стоит отметить, что полученные оценки моделей, указанные в приведенных работах, сложно сравнивать, потому что данные у каждой группы исследователей отличаются. Во-первых, отличаются размеры

обучающих выборок. Во-вторых, у каждого суперкомпьютерного центра может быть своя специфика пользователей. Предположим, у одного из центров большая часть пользователей – это люди, имеющие достаточный опыт для априорной оценки ресурсов задания. Следовательно, модель, обученная на таких данных, будет отличаться высокой точностью.

Построение новых признаков

Исходные признаки, указанные пользователем не отражают всей информации доступной для модели. Поэтому необходимо на основе имеющихся данных построить новые признаки, которые позволяют дифференцировать задания.

Для того, чтобы понять, какие признаки позволяют получить наиболее точную модель, были выделены следующие группы признаков:

- Исходные (И).
- Временные (В).
- Персональные (П).

Обозначения типов признаков: В – вещественные, К – категориальные, Б – бинарные.

Исходные признаки представляют собой данные, которые поступают на вход планировщика (табл. 1). К исходным признакам был добавлен признак *square,* представляющий собой площадь задания (ядер × час).

Табл. 1 – Исходные признаки.

Имя	***Тип***	***Описание***
userid	К	id пользователя
nproc	В	заказанные ядра
ntime	В	заказанное время
quantum	В	квант
gid	К	id группы
domain	К	область исследований
orgid	К	id организации
square	В	площадь задания

Временные признаки представляют собой признаки, которые отражают момент, когда задание было зарегистрировано пользователем (табл. 2). Их применение основано на гипотезе о том, что пользователям свойственны определенные шаблоны поведения, связанные с временем запуска заданий. Это может найти отражение в сезонных изменениях. К примеру, средняя длительность заданий к концу года или в период праздников может иметь определенную тенденцию. Так же, можно предположить, что время запуска определенного пользователя в течение рабочего дня может иметь выраженный характер и улавливаться моделью.

Табл. 2 – Временные признаки.

Имя	***Тип***	***Описание***
day	В	день
hour	В	час
month	В	месяц
dow	В	день недели

Персональные признаки формируются исходя из статистики пользователя (табл. 3). Использование данных признаков основано на гипотезе, что определенным пользователям свойственны определенные шаблоны поведения в пользовании суперкомпьютерным кластером МСЦ РАН. На основе прошлых запусков пользователя формируется статистика, которая предполагаемо описывает данного пользователя и позволяет модели обучиться и выделять группы пользователей со схожей статистикой.

Табл. 3 – Персональные признаки.

Имя	***Тип***	***Описание***
duration_user	В	среднее время работы задания
nproc_user	В	среднее заказанное число ядер
ntime_user	В	среднее заказанное время
resourses_user	В	среднее число использованных ресурсов
exceedtime_user	В	среднее преувеличение заказанного времени
experience_user	В	количество дней между первым и последним запуском (опыт пользователя)

Таким образом, мы получаем 18 признаков.

Отбор признаков

Отбор признаков является распространенным этапом в подготовке данных. Его необходимость обуславливается рядом причин.

Очевидно, что не все признаки являются информативными и связаны с целевой переменной. Наличие таких признаков негативно влияет на качество модели и на ее обобщающую способность. Не всегда можно на этапе формирования данных отличить неинформативные признаки, поэтому изначально включаются все имеющиеся.

Для того чтобы отобрать важные признаки существуют несколько подходов. Самым простым подходом является метод при котором рассчитывается связь между каждым из признаков по отдельности и целевой переменной. На рисунке 2 приведены коэффициенты корреляции Пирсона для всех вещественных признаков. Отметим, что наибольшая корреляция прослеживается со средним временем выполнения задания пользователя. То есть каждому пользователю присущи задания определенной длины. Однако, проблема данного подхода заключается в том, что одномерный отбор не позволяет выявить комбинации признаков, которые имеют информативность выше, чем у каждого из входящих признаков в отдельности. Поэтому в МО существуют другие методы отбора признаков, позволяющие, учитывать их взаимодействие.

Линейные модели. Ответом линейной модели является взвешенная сумма признаков на данном объекте:

$$a(x) = \sum_{j=1}^{d} w_j x^j$$

где d - количество признаков, w - вес признака x.

Таким образом, признаки с большими весами вносят больший вклад в ответ модели, следовательно, являются более важными. Для этого модель обучается на признаках и веса, которые были получены в процессе обучения, интерпретируются как важность того или иного признака. Важным условием является масштаб признаков. То есть необходимо, чтобы признаки были выровнены относительно нуля. В данной работе к данным было применено масштабирование. Каждое значение признака заменяется на его z-оценку по формуле:

$$z = \frac{x - \bar{x}}{\sigma}$$

где x - значение признака, $\bar{x}$ - среднее значение признака на всех объектах, σ - стандартное отклонение.

Для того, чтобы не допустить переобучения модели параметры масштабирования – среднее значение и стандартное отклонение вычисляются на обучающей выборке и потом применяются к тестовой.

Так как задача заключается в отборе признаков, то для того, чтобы занулить как можно больше неинформативных признаков, можно использовать L1-регуляризацию. Алгоритм линейной регрессии с L1-регуляризатором использует l_1-норму вектора весов. Алгоритм описывается следующим образом:

$$\arg\min_{w \in \mathbb{R}^m} \left(\frac{1}{2m} \|Xw - y\|^2 + \lambda \|w\|_1 \right)$$

где X - матрица признаков, w - вектор весов, y - целевая переменная, m - количество объектов, λ - параметр регуляризации. С увеличением λ увеличивается количество зануленных неинфоматичных признаков.

Линейная модель с L2-регуляризатором использует l_2-норму вектора весов.

Решающие деревья. Решающие деревья строятся жадно – от корня к вершине. Для того, чтобы произвести разбиение вершины нужно выбрать признак и порог, по которому определяется в какое из поддеревьев отправится данный объект.

Для этого в каждой вершине вычисляется взвешенная сумма критериев информативности левого и правого поддеревьев. По тому, как сильно признак уменьшил это значение можно судить о том, какой вклад он вносит в ответ модели.

Так же для этих целей можно использовать композиции деревьев в случайном лесе или градиентном бустинге. В данном случае уменьшение критерия информативности для данного признака вычисляется как сумма по всем деревьям композиции. Чем больше данная сумма, тем важнее признак для данной композиции.

Жадный метод отбора. Предположим мы имеем начальное множество признаков:

$$J_1 = \{i_1, ..., i_k, ..., i_n\}$$

На каждой итерации мы обучаем мо-

дель и получаем оценки важности признаков. Например, для линейных моделей оценками являются веса признаков. Далее признак с наименьшей важностью удаляется и модель обучается на новом наборе признаков:

$$J_2 = \{i_1, ..., i_{k-1}, i_{k+1}, ..., i_n\}$$

Алгоритм продолжает работать пока количество признаков не соответствует минимальному указанному числу или пока ошибка не начнет увеличиваться.

Рис. 2 - Корреляция между временем выполнения задания и вещественными признаками.

Описание выбранных алгоритмов

Для обучения моделей использовалась популярная библиотека алгоритмов машинного обучения *scikit-learn* [10]. Были выбраны несколько регрессионных алгоритмов, представляющие различные классы. Документация к ним может быть найдена в [11].

- *Decision Tree Regressor*: данный алгоритм использует дерево решений для выполнения регрессии целевой переменной.
- *Random Forest Regressor*: модель представляет собой композицию деревьев, составленных с использованием рандомизации. Так, каждое отдельное дерево составляется при помощи случайной подвыборки объектов из обучающей выборки. Также при составлении правила разбиения узла, выбираются либо все признаки, либо случайное подмножество.
- *Lasso Regression*: линейный алгоритм, который добавляет к функции потерь l_1–регуляризатор, позволяющий снижать влияние несущественных признаков, обнуляя их.
- *Ridge Regression*: линейный алгоритм, который добавляет к функции потерь l_2–регуляризатор, что позволяет уменьшать величину отдельных весов.
- *Gradient Boosting* Regressor: строит композицию деревьев с помощью жадной стратегии. Алгоритм обучает новое дерево, которое корректирует ошибку предыдущего. Таким образом, ответ модели является суммой ответов всех деревьев композиции.
- *K-Nearest Neighbors:* алгоритм, который предсказывает ответ для нового объекта при помощи интерполяции значений целевых переменных *k* соседей данного объекта.

Выбор метрики и обучение моделей

Для оценки регрессионных моделей обычно используется стандартная метрика R^2 – коэффициент детерминации, определяемый по формуле:

$$R^2 = 1 - \frac{SS_{res}}{SS_{tot}}$$

где SS_{res} - сумма квадратов остатков регрессии, SS_{tot} - общая сумма квадратов.

Но данная метрика имеет существенный недостаток. По мере того как добавляются новые признаки к существующей модели R^2 может только увеличиваться. Предположим, что мы добавили к модели новые шумовые признаки. Существует вероятность, то они объясняют часть дисперсии зависимой переменной. Поэтому для того, чтобы сравнивать оценки регрессионных моделей с разным числом признаков, применяется скорректированный (adjusted) коэффициент детерминации:

$$R^2_{adj} = 1 - (1 - R^2)\frac{(n-1)}{(n-k-1)} \le R^2$$

где n - количество объектов в модели, k - количество признаков.

Несмотря на то, объектов значительно больше, чем признаков, а следовательно корректирующий коэффициент несущественно снижает исходный R^2 , для корректности в

качестве критерия качества модели применяется R^2_{adj}.

Исходный набор данных был разделен на обучающую и тестовую выборки в пропорции 7:3, соответственно. Так как данные представляют временной ряд, то стандартная k-fold кросс-валидация здесь не подходит, так как в этом случае модель будет тестироваться на валидационной выборке, которая по времени находится раньше, чем обучающая выборка. Поэтому для кросс-валидации был выбран объект *TimeSeriesSplit* с пятью разбиениями.

Скользящее среднее

Сегодня в виду повсеместного применения машинного обучения часто возникает вопрос в целесообразности применения алгоритмов МО. Поэтому для проверки целесообразности применения приведенных алгоритмов в данной задаче был применен простой алгоритм, основанный на скользящем среднем.

Принцип работы данного алгоритма заключается в том, что для вновь пришедшей задачи, время ее выполнения рассчитывается как среднее арифметическое времени работы последних *n* задач пользователя,

$$t_k = \frac{1}{n}\sum_{i=1}^{n} t_{k-i}$$

где t_k - оценка времени выполнения k-ого задания пользователя.

Данный алгоритм был применен для значений $n = 1...9$. Если пользователь на момент отправки нового задания не имеет ни одного задания, то в качестве оценки времени выполнения задания применяется указанное пользователем значение. Для каждого значения *n* рассчитано соответствующее значение R^2. Результаты представлены на рисунке 3.

Наилучший показатель $R^2 = 0.425$ достигается при *n* = 3.

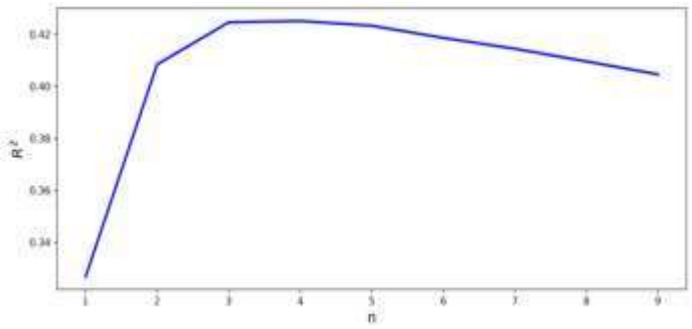


Рис. 3 – Зависимость R^2 от *n* (число последних задач пользователя) при использовании алгоритма скользящего среднего.

Результаты

Отбор признаков на основе моделей. Для отбора признаков была использована следующая схема: обучение на полном наборе признаков → отбор трех наиболее информативных признаков → обучение модели на выбранных признаках.

В каждом обучении для поиска параметров использовалась сетка, поэтому параметры моделей на разных наборах признаков не обязательно совпадают.

На рисунке 4а представлены результаты отбора признаков с использованием оценок различных алгоритмов. В таблице 4 представлены результаты обучения моделей на полном наборе признаков и на трех самых информативных признаках.

Так как алгоритм K-ближайших соседей не оценивает важность признаков, то для сравнения был использован набор из трех признаков, отобранных градиентным бустингом.

Все алгоритмы на основе решающих деревьев выбирают основным признаком *ntime* (время, заказанное пользователем). Линейные алгоритмы ставят его на второе место, в то время как *duratiom_m_per_user* (среднее время длительности задания для пользователя) они оценивают выше всего.

Признак *time_exceeded_share_per_user* (средняя доля

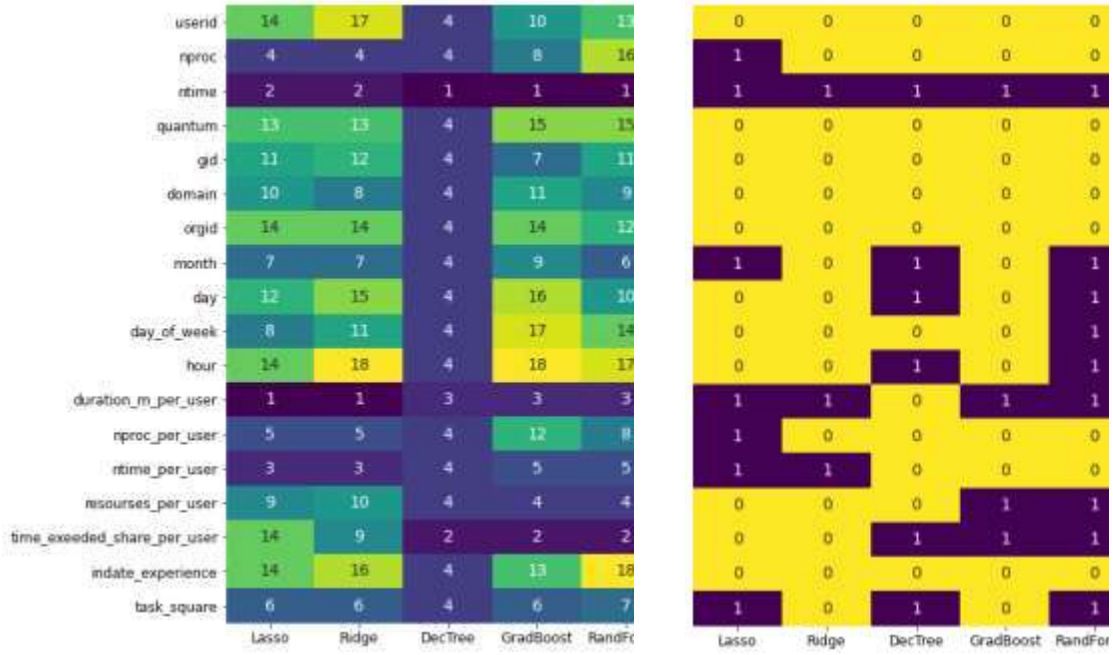
неиспользованного заказанного времени для пользователя) является вторым по значимости признаком для всех алгоритмов, основанных на дереве решений. Данные алгоритмы показали лучший результат. Стоит отметить, что в смежных работах дерево решений так же часто оказывается лидером. Это может быть связано с тем, что значительная доля признаков является категориальными, это облегчает задачу разделения по вершинам дерева, но усложняет регрессию.

Необходимо отметить, что *userid, orgid, gid* являются плохими предикторами времени работы задания.

Из временных признаков *month* является самым информативным, что подтверждает предположение о сезонных тенденциях в запусках.

Главным выводом является то, что модель, обученная на меньшем числе признаков показывает такой же или больший результат. Следовательно, значительная часть выбранных признаков является малоинформативными.

Жадный отбор признаков. Отбор признаков производился с помощью объекта RFECV (Recursive Feature Elimination with Cross Validation) из библиотеки *sklearn*. Данный алгоритм производит отбор признаков, производя кросс-валидацию на каждой итерации, и исключает один из признаков. В качестве минимального количества признаков, которые необходимо оставить, выбрано n = 3. RFECV исключает признаки, пока не останется три или пока метрика не станет уменьшаться.



а) б)

Рис. 4 – Результаты отбора признаков.
а - Результат ранжирования признаков по степени информативности на основе моделей.
б - Результаты жадного отбора признаков (1 – признак информативен, 0 – не информативен).

Таблица 4 – Результаты обучения моделей на полном наборе признаков и на отобранных на основе моделей.

	Lasso	**Ridge**	**Dec. Tree**	**Grad. Boost.**	**Rand. Forest**	**KNN**
R^2_{adj} на всех признаках	0.35	0.35	0.66	0.68	0.68	0.28
R^2_{adj} на 3-ех самых важных признаках	0.35	0.35	0.67	0.69	0.68	0.56

Таблица 5 – Результаты обучения моделей на признаках, отобранных жадным методом.

	Lasso	**Ridge**	**Dec. Tree**	**Grad. Boost.**	**Rand. Forest**
R^2_{adj} на отобранных признаках	0.35	0.34	0.67	0.68	0.68

Использована следующая схема: жадный отбор признаков на обучающей выборке → поиск лучших параметров модели на отобранных признаках → окончательная оценка модели на тестовой выборке. Результаты приведены на рисунке 4б.

Отметим, что *duration_m* и *n_time* снова оказываются в лидерах. Также *month* и *time_exceeded_share_per_user* оцениваются как важные тремя алгоритмами из пяти.

В таблице 5 приведены результаты данных моделей. Они остаются почти идентичными к результатам в таблице 4.

Из результатов очевидно, что персональные признаки являются более информативными по сравнению с временными.

Интересно, что такие признаки как *user_id, orgid, gid* оказались не востребованными. Можно предположить, что персональные признаки помогают модели обобщать группы пользователей со схожей статистикой, в то время как id группы или пользователя является категориальным признаком и модель не может объединять их в группы. Также может играть роль небольшое число пользователей в данных n = 264 и организаций n = 54.

Отдельное внимание стоит обратить на скользящее среднее. Данный алгоритм демонстрирует приемлемый для своей простоты результат. Значение R^2 для него оказывается выше, чем для всех моделей, основанных на линейной регрессии. Это может объясняться спецификой пользователей кластера МСЦ РАН. Можно предположить, что пользователи часто запускают задания со схожим временем выполнения или в случае ошибки перезапускают задание снова и снова, поэтому среднее арифметическое времени работы трех последних заданий показывает такие хорошие результаты. Несомненным плюсом данного алгоритма является его интерпретируемость.

Дальнейшая работа

В дальнейшем планируется использование полученных моделей для работы с потоком заданий, смоделированных с помощью симулятора, например, симулятор SLURM [12]. Это позволит дать конечную оценку для моделей, так как позволит рассчитать и сравнить показатели использования ресурсов.

Заключение

В работе приведены результаты применения алгоритмов МО для предсказания времени работы заданий, поступающих от пользователей. В качестве обучающей выборки были использованы данные суперкомпьютера МВС-10П «Торнадо» за 2013 – 2017 гг.

Для сравнения выбраны несколько алгоритмов, которые относятся к различным классами. Приведено сравнение алгоритмов на основе скорректированного коэффициента детерминации R^2_{adj}. Лучшими алгоритмами оказались Decision Tree Regressor, Random Forest Regressor, Gradient Boosting.

В работе приведено описание методов отбора признаков. Рассмотрены методы отбора на основе моделей и жадный метод. Приведены результаты отбора.

Персональные признаки пользователя, на основе статистики пользования суперкомпьютерным кластером МСЦ РАН чаще чем временные признаки оказывались значимыми.

Также к данным был применен простой алгоритм скользящего среднего. Алгоритм показал результат, превышающий результаты моделей, основанных на линейной регрессии. Приемлемый результат данного алгоритма демонстрирует специфику пользователей МСЦ РАН. Пользователи склонны запускать задания со схожим временем выполнения, что позволяет предсказывать время выполнения эффективнее, чем с помощью линейной регрессии.

Работа выполнена при поддержке гранта РФФИ № 18-29-03236.

Comparison of machine learning algorithms for predicting the user job runtime in the framework of optimizing the use of resources of supercomputer cluster of JSCC RAS

A.A. Rybakov, S.S. Shumilin

Abstract. Assessing the resources needed to run a program on a supercomputer cluster is a difficult task. The user needs to practically guess the time of the program, which leads to the fact that some tasks are forcibly terminated by the scheduler due to exceeding the time limit. A large number of such events leads to the fact that the distribution of cluster resources loses effectiveness. In order to optimize the distribution of resources, it is necessary that the scheduler receive more accurate estimates of the task parameters. Earlier, attempts were made to use machine learning algorithms to solve this problem. This article discusses the application of these algorithms to the data of the supercomputer cluster of the JSCC RAS. The results are obtained on the quality of regression models on various additional features, among which the most informative ones were selected. To do this various techniques commonly used in machine learning were used. These features that characterize the user of the cluster can be used to improve the accuracy of already existing models.

Keywords. HPC, machine learning, predictive analytics, scheduler, user modeling.

Литература

1. Система управления прохождением параллельных заданий. Руководство программиста (пользователя), (2016). http://www.jscc.ru/wp-content/uploads/2017/06/SUPPZ-user-guide-2016.pdf
(дата обращения: 14.02.2020).
2. Webb, G. I., Pazzani, M. J., and Billsus, D, “Machine learning for user modeling” User modeling and user-adapted interaction, vol. 11 pp. 19-29, 2001.
3. Yao, Y., Zhao, Y., Wang, J., and Han, S, “A model of machine learning based on user preference of attributes”. International Conference on Rough Sets and Current Trends in Computing, pp. 587-596. Springer, Berlin, Heidelberg, 2006.
4. Warren Smith, Ian Foster, and Valerie Taylor. Predicting application run times using historical information. In Workshop Job Scheduling Strategies for Parallel Processing (JSSPP). pp. 122–142, 1998.
5. Warren Smith, Valerie Taylor, and Ian Foster. Using run-time predictions to estimate queue wait times and improve scheduler performance. In Workshop on Job Scheduling Strategies for Parallel Processing (JSSPP). 202–219, 1999.
6. Dan Tsafrir, Yoav Etsion, and Dror G Feitelson. Backflling using systemgenerated predictions rather than user runtime estimates. 2007 IEEE Transactions on Parallel and Distributed Systems 18, 6 (2007).
7. Dan Andresen, William Hsu, Huichen Yang, and AdedolapoOkanlawon. 2018. Machine Learning for Predictive Analytics of Compute Cluster Jobs. *CoRR* abs/1806.01116 (2018). http://arxiv.org/abs/1806.01116 (дата обращения 20.03.20).
8. Rezaei, Mahdi &Salnikov, Alexey. (2018). Machine Learning Techniques to Perform Predictive Analytics of Task Queues Guided by Slurm. 1-6. 10.1109/GloSIC.2018.8570130.
9. Rodrigues, Eduardo & Cunha, Renato &Netto, Marco & Spriggs, Michael. (2016). Helping HPC Users Specify Job Memory Requirements via Machine Learning. 10.1109/HUST.2016.006.
10. 2019. Getting Started with Scikit-learn for Machine Learning. In *Python® MachineLearning*. John Wiley & Sons, Inc., 93–117. https://doi.org/10.1002/9781119557500.
11. Документация к sklearn. URL: https://scikit-learn.org (дата обращения 20.03.20).

12. Симулятор SLURM. URL: https://github.com/ubccr-slurm-simulator/slurm_simulator. (дата обращения 20.03.20).