**Генетическое программирование для трансфера образцов. Обучение в символьной регрессии.**

Аннотация

Transfer learning недавно привлекло больше внимания в сообществе машинного обучения. Он направлен на улучшение результатов обучения в интересующей нас области с помощью уже имеющихся аналогичных данных из этой же области. Тем не менее, есть только ограниченное количество исследований по изучению этого метода в генетическом программировании для символьной регрессии. Эта статья призвана заполнить этот пробел, предложив новый *фреймворк, основанный на важности весов уже полученных экземпляров* (??) для алгоритма transfer learning в области генетического программирования и символьной регрессии. В новом фреймворке дифференциальное развитие используется для поиска оптимальных весов для исходных экземпляров данных, которые помогают генетическому программированию выявлять наиболее полезные экземпляры исходных данных и учиться на них. В то же время, метод оценки плотности, который обеспечивает хороший старт алгоритма, показывает, что может помочь в поиске оптимального веса в процессе отбрасывания некоторых невалидных или неважных для обучения данных до составления модели регрессионного обучения. Экспериментальные результаты показывают, что, сравнивая с генетическое программирование и регрессию с опорным вектором, которая учится только с помощью целевых экземпляров, и обучение на смеси исходных и целевых данных без какого-либо transfer-компонента, можно заметить, что рассматриваемый нами метод может развивать регрессионные модели, которые не только достигают заметно большей эффективности в стабильности междоменной генерализации, но и эффективно снизить тенденцию к переобучению. Также стоит отметить, что эти модели, как правило, намного проще, чем модели, сгенерированные другими методами генетического программирования.

Введение

Transfer learning является относительно новым сценарием обучения в ML, который направлен на улучшение обучения в области интересов (называемый также таргетным доменом, target domain), используя знания, которые уже были получены от аналогичных доменов (их называют исходными доменами). Обоснование этого нового сценария обучения заключается в том, что люди могут более эффективно решать новые проблемы, используя знания об аналогичных старых. Высокая эффективность transfer learning была признана в двух типичных сценариях, где имеется достаточно экземпляров в исходных доменах, но не доступен таргетный домен, либо когда распределение данных может меняться со временем. При таких сценариях традиционные техники машинного обучения, которые предполагают, что данные обучения и те данные, о которых мы еще не знаем, соответствуют одному и тому же распределению, как правило, не работают хорошо. Обозначим базовые распределения исходных доменов как (s значит sourse) и для таргетных доменов (t – target), где и – выходные данные, а – выходные данные. В transfer learning используется для лучшей аппроксимации , поскольку выражение может быть выведено из двумя способами: с помощью адаптации или .

В последние годы техники transfer learning были предложены во многих областях, например, reinforcement learning (обучение с подкреплением) и supervised learning (обучение с учителем). По сравнению со многими исследованиями в этих областях, transfer learning в GPSR (genomics, protenomics, systems resource) все еще встречается редко, можно найти небольшое количество работ. Все они стремятся передать знания, полученные из исходных доменов, к целевому домену с помощью обученных моделей. Именно в GSPR transfer learning до сих пор не рассматривался как важный подход. Недавно мы предложили новый метод генетического программирования – Transfer Learning in Genetic Programming в символьной регрессии. TLGP был использован, чтобы показать возможность улучшения обучения в целевой области с помощью повторного использования знаний из исходного домена. Это достигается путем использования нового фреймворка (new instance weighting framework, о нем говорилось тут: *фреймворк*), который использует веса экземпляров: улучшает их для исходных экземпляров домена в процессе алгоритма наряду с эволюционным процессом генетического программирования. Несмотря на многообещающую способность к обучению данного алгоритма, у него есть некоторые потенциальные ограничения. С увеличением разницы между количеством экземпляров исходных данных и экземплярами целевых данных (когда большое количество исходных данных, но небольшое количество целевых данных) TLGP становится гораздо труднее идентифицировать полезные экземпляры исходных данных, что ограничивает его возможности улучшения обучения в целевой области. Кроме того, обычно требуется много тяжелых вычислений для поиска оптимальных весов для большого количества исходных данных.

А. Цели

Цель этой статьи – предложить новый фреймворк (структура поиска весов) для transfer learning в GPSR. Он предназначен для того, чтобы устранить вышеописанные ограничения и сделать процесс обучения (поиска весов) более эффективным и действенным, чем [11 – ссылка на какой-то метод из книги «Дифференциальная эволюция transfer learning, основанного на исходных данных, в генетическом программировании для символьной регрессии»]. Предлагаемый метод направлен на дальнейшее улучшение междоменной способности к обобщению развитых регрессионных моделей с помощью эффективного выбора и взвешивания (??) исходных доменов, а также более точной корректировки разности между предельными распределениями и . В частности, эта статья имеет следующие основные цели:

1. Разработать фреймворк, который учитывает веса уже изученных данных, для GSPR с помощью оценки коэффициента плотности (??), чтобы эффективно выбирать и взвешивать экземпляры исходных данных и исследовать, нужна ли новая система выбора весов (сможет ли TLGP улучшить свои характеристики для более хорошего обучения целевых данных)
2. Исследовать, может ли новая система взвешивания эффективно получать знания, выбирая и взвешивая данные исходного домена, чтобы получить лучшую способность к обобщению GSPR только для данных целевого домена.
3. Исследовать, сможет ли новый фреймворк более полезно исследовать переносимую информацию (transferable knowledge), таким образом приводя к лучшему междоменному обобщению, чем GSPR, c помощью обучения как из исходных данных, так и целевого домена.

Бэкграунд и концепты

А. Transfer learning

Формальное определение трансфертного обучения (transfer learning) [12 – тоже ссылка на литературу] выглядит следующим образом.

Определение 1 (transfer learning): учитывая исходную область и исходную учебную задачу , целевую область и целевую учебную задачу , где или , процесс transfer learning использует знания в и для улучшения обучения в .

В этом определении есть два важных элемента: область (домен) и задача. Область (домен) имеет две составляющие: пространство признаков и распределение предельных вероятностей , где , d-количество объектов. Задача , где - пространство меток (может быть имеется в виду искомое пространство признаков), а - функция прогнозирования. Состояние означает, что либо , либо , а состояние показывает, что или . Более поздний случай, который также известен как адаптация домена [12], [13], находится в центре внимания этой статьи.

Исходя из того, что нужно передать, методы transfer learning можно сгруппировать в четыре категории, а именно: 1) основанные на экземплярах; 2) основанные на представлении данных (на представлении характеристик этих данных); 3) основанные на параметрах; и 4) реляционные подходы к обучению передаче знаний [12]. В этой статье основное внимание уделяется transfer learning на основе экземпляров.

B. Transfer learning на основе экземпляров

Взвешивание экземпляров, как обобщенная форма статистических методов коррекции смещений, имеет потенциал для коррекции смещения при отборе экземпляров (?? о чем тут речь?). Поэтому это рассматривается как важный тип метода transfer learning для коррекции расстояния (разницы) между исходной и целевой областями. Обоснование при взвешивании экземпляров для transfer learning заключается в том, что из-за различных распределений в исходной и целевой областях некоторые из данных исходного домена могут быть вредоносными (снижающими качество обучения), в то время как некоторые другие могут быть повторно использованы для обучения в целевом домене путем повторного взвешивания, чтобы исправить предельную разницу между этими двумя доменами.

Учитывая распределение исходного домена / плотность (распределения) и распределение целевого домена , соотношение плотности показало свою эффективность в уменьшении разницы распределения между этими двумя доменами. Таким образом, оно было широко использовано для transfer learning. Поскольку и обычно неизвестны в реальности, оценка отношения плотности, таким образом, становится ключевой проблемой, например, взвешивания. Для переноса обучения/адаптации предметной области был предложен ряд методов, непосредственно оценивающих соотношение плотности [14], [15].

Предлагается среднее соответствие ядра (KMM – kernel mean matching) [14], чтобы сделать взвешенное распределение исходного домена аналогичным распределению целевого домена. KMM изучает , сопоставляя средние значения между данными исходной области и данными целевой области в гильбертовом пространстве воспроизводящего ядра (RKHS – reproducing-kernel Hilbert space). Процедура оценки важности Kullback–Leibler (KLIEP) [15] предложила аналогичную идею для оценки w(x) непосредственно путем минимизации дивергенции Kullback–Leibler для оценки весов. Эти исследования доказали, что теоретически он способен изучать точные модели в целевой области путем взвешивания экземпляров исходной области с правильными значениями . Однако трудно иметь общий метод для точной оценки для различных алгоритмов обучения.

Для итеративного выявления и использования более релевантных данных исходного домена предлагается алгоритм бустинга TrAdaBoost [16]. TrAdaBoost уменьшает относительный вес исходного экземпляра, который неправильно классифицирован, одновременно увеличивая вес неправильно классифицированного целевого экземпляра. Таким образом, TrAdaBoost будет сосредоточен на исходных экземплярах, которые больше похожи на целевые экземпляры, а также на более сложных экземплярах. Позже TrAdaBoost был расширен для регрессионных задач [17]. Однако, как сообщалось в [17], эффективность transfer learning TrAdaBoost на регрессионных задачах не так хороша, как ожидалось, поскольку экземпляры в исходной области будут доминировать в процессе обучения.

Выбор экземпляра исходных данных также является чувствительной проблемой для transfer learning. Лоуренс и Платт [18] расширили информационную векторную машину (IVM) для выбора экземпляров для многозадачных работ. Расширенный метод IVM выбирает больше экземпляров из задач, которые предоставляют больше информации для процесса обучения.

C. Эволюционный transfer learning

Мин и др. [19] разработали основанный на метаэвристике метод отбора экземпляров (MIST) для обучения передаче данных. MIST использует популяционную метаэвристику, где каждое решение представляет подмножество экземпляров исходного домена. Качество решения оценивается по расчетной ошибке обобщения гауссовского процесса (estimated generalization error in Gaussian process), которая учится на комбинации выбранных исходных данных и целевых обучающих данных. MIST ищет оптимальное подмножество экземпляров исходного домена, руководствуясь их вкладом в процесс transfer learning. Это похоже на transfer learning component, предложенный в этой статье. С другой стороны, каждый из выбранных исходных экземпляров в MIST будет в равной степени способствовать процессу transfer learning, в то время как, назначая веса выбранному источнику, предлагаемая система взвешивания экземпляров, как ожидается, поможет более точно организовать процесс трансферного обучения. Гупта и др. [20] предложили многофакторный эволюционный алгоритм (MFEA) для многозадачного обучения. MFEA выполняет неявный параллельный поиск оптимальных решений для нескольких задач. Они вводят важное понятие фактора мастерства (skill factor) для определения задачи, на которой каждый экземпляр наиболее эффективен среди данных задач. Фактор мастерства передается по наследству от родителей к детям как своего рода вертикальная культурная наследственность. Ученые показали, что MFEA может решать разнообразные проблемы одновременно и сходиться быстрее. Фэн и др. [21] изучили полезность обмена знаниями между различными задачами и предложили новый метод решения эволюционной многозадачности с помощью явного генетического переноса между задачами. Denoising autoencoder используется для построения отображения любых двух задач. Решения передаются из одной задачи в другую с помощью этих отображений задач. Показано, что предложенный метод способен решать как простые, так и сложные задания более эффективно, чем современные методы. Чандра и др. [22], [23] предложили коэволюционные алгоритмы для решения многозадачных работ с помощью нейронных сетей. Они предложили различные стратегии для разложения проблемы на несколько задач. Знания (информация) передаются по каскадам в структуре нейронных сетей. Эффективность алгоритмов была доказана путем сравнения с современными методами нейроэволюции.

D. Transfer learning в символьной регрессии

По сравнению со многими существующими работами в области transfer learning для традиционной регрессии, в литературе имеется лишь небольшое количество исследований по transfer learning в GPSR. Все они сосредоточены на модульном обучении передаче знаний. Динь и др. [7] передавали (transferred) знания, полученные из исходной области, в трех типах моделей: лучшие модели (хорошо приспособленные), субмодели финального поколения и лучшие генерационные модели (модели поколения) GP на исходных данных. При решении проблемы целевой области эти модели присоединятся к первоначальному набору GP. Усилия по передаче знаний с использованием всех этих моделей ограничены. Хаслам и др. [8] провели дальнейшее исследование в [7]. Они добавили дополнительные компоненты для обнаружения многоразовых моделей, а также приблизились к передаче параметров (фич). О'Нил и др. [9] собрал потенциально полезные строительные блоки в GP на данных исходного домена и расширил набор функций GP при обучении из целевого домена. Чжун и др. [24] расширили MFEA [20] для решения GPSR как многозадачной проблемы. В их многофакторном методе GP знания могут быть перенесены через задачи символической регрессии и некоторые другие задачи, например, даже паритет.

III. Представляемый метод

В этой статье рассматривается задача transfer learning в символьной регрессии, используя генетическое программирование с новым фреймворком (он основывается на весах у исходных данных, о нем говорилось выше). Суть предложенного алгоритма (фреймворка) заключается в том, чтобы найти оптимальные веса для исходных данных, которые позволят использовать их заново, причем эффективно для таргетных данных, выделяя только те исходные данные, которые будут полезны для наших таргетных, и устраняя эффекты бесполезных для обучения исходных данных (забавная статья, уже четвертую страницу повторяет одно и то же).

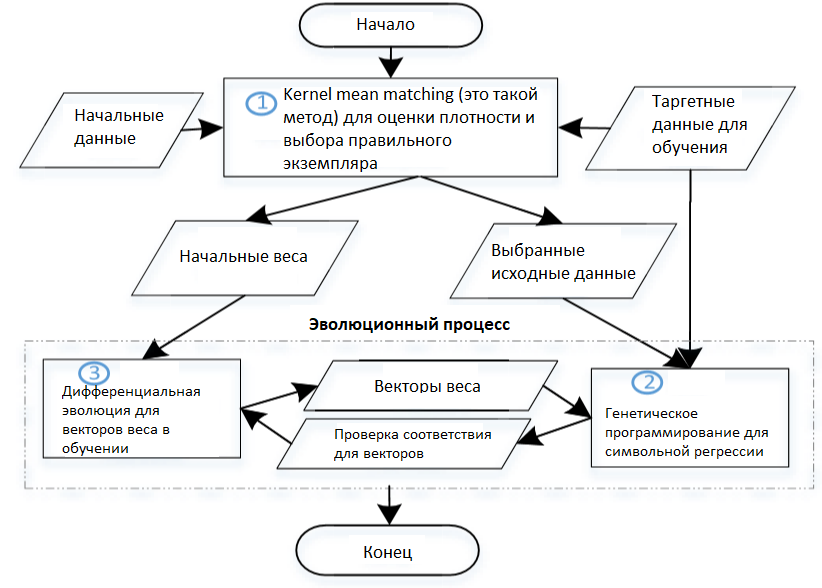
Мы провели некоторое предварительное исследование в статье [11]. Из-за ограничения по страницам Чен и его друзья [11] не описали детально именно такой всеобъемлющий алгоритм. При чрезмерно большом количестве исходных данных и одновременно с небольшим количеством таргетных данных, поиск оптимальных весов с помощью алгоритма [11] становится менее эффективным. В этом случае исходные данные могут доминировать при процессе transfer learning'а. В этой статье предлагается новый фреймворк, который направлен на то, чтобы облегчить (уменьшить ??) доминирование экземпляров исходных данных в процессе обучения, предлагаемым этой статьей. Более конкретно, метод в статье [11] ищет оптимальный вектор веса для экземпляров исходного домена, пока ищет вектор оптимальных весов для выбранных экземпляров.

Новый фреймворк стремится искать оптимальные веса, поэтому требуются численные методы поиска. Эволюционные алгоритмы, такие как particle swarm optimization (PSO), генетические алгоритмы (GAs), дифференциальная эволюция (DE) [25], можно использовать для выполнения процесса поиска в предлагаемом фреймворке. В сравнении с PSO и GAs, DE в целом проще и имеет более разнообразную совокупность (??) [26] : так, используется более гибкая версия DE - SaDE [27].

А. Общая структура

GP (генетическое программирование), оснащенное новым фреймворком, называется Instance transferring GP (ITGP). Общая структура и поток данных в таком алгоритме показаны на рисунке ниже.

Как показано на схеме, ITGP имеет три ключевых компонента. Kernel mean matching, который является подходом к оценке коэффициента плотности (?? Непонятно, что за коэффициент, density ratio estimation) используется для оценки начальных весов для экземпляров исходных данных (исходной области). Этот метод используется как метод поиска начального вектора, то есть он считается хорошей отправной точкой для поиска оптимальных весов. Необходимо понимать, что процесс выбора подходящих экземпляров, базирующихся на этих весах, происходит во время работы этого алгоритма (то есть одновременно). Эволюционный процесс (видимо, итерационный??) выполняется уже на выбранных экземплярах исходных данных, которые соответствуют векторам веса. В каждом поколении (итерации??) алгоритм генетического программирования развивает наборы регрессионных моделей, в то время как алгоритм дифференциальной эволюции [27] развивает вектор весов (под развитием здесь видимо понимается улучшение, на каждой итерации векторы/наборы данных стремятся к идеальному варианту). Взаимосвязь между алгоритмами генетического программирования и дифференциальной эволюции заключается в том, что генетическое программирование учится на разных наборах взвешенных экземпляров из данных исходной области с использованием весов, которые посчитал алгоритм дифференциальной эволюции, а он, в свою очередь, оптимизирует эти самые веса, используя обратную связь от метода генетического программирования, то есть получаем эффективность обученных регрессионных моделей на таргетных данных, как полноту весов данных (?? Непонятна суть предложения).

 Рассмотрим примерный алгоритм ITGP:

1. Первый шаг. Алгоритм kernel mean matching инициализирует m векторов веса и выбирает экземпляры исходной области с наибольшим весом. Весовые векторы определяются как – вектор из n элементов. У каждого элемента два измерения – индекс экземпляра в исходных данных и вес этого экземпляра.

Веса это то что в источнике !!- это n x m

1. Второй шаг. Алгоритм генетического программирования инициализирует набор моделей.
2. Третий шаг. Алгоритм дифференциальной эволюции использует вектор, полученный kernel mean matching, в качестве начального.
3. Четвертый шаг. Повторяем эволюционный процесс ITGP, пока не будет выполнен критерий останова.

А. Шаг 4.1: GP берет весовые векторы у DE и оценивает каждый из полученных результатов соответственно взвешенным исходным данным.

Б. Шаг 4.2: GP повторно оценивает топ результатов GP (что за топ?? Имеются в виду элементы вектора??) на таргетных тренировочных данных и предоставляет значения плотности для DE.

В. Шаг 4.3: DE берет данные у GP и генерирует дочерние данные.

Г. Шаг 4.4: GP выбирает родителей и генерирует дочерние данные.

F то что оптимизируем, p деревьев

Веса – разностная эволюция, ГП это для деревьев.

Нужна целевая функция

M весов и P Деревьев. Берем дерево P, считаем значение на домене источника для каждого sample

Для каждого семпла есть вес – считаем взвешенную сумму  
Для каждого дерева Р m значений функции

Чтоб получить для каждого m берем минимум по p

В формуле 4 ошибка относительно источника

Для каждого вектора весов m определили лучшую модель из p

Модель вычисляем на данных из целевого домена

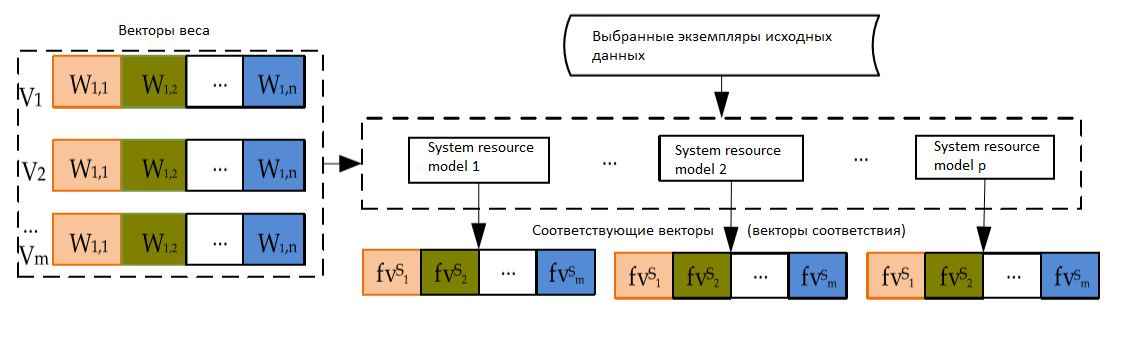
Целевая функция для конкретного набора весов – это наибольшее

Детали каждого шага ITGP рассматриваются далее.

В. Основные компоненты данных в ITGP

В этом разделе представлены две основных компоненты данных в рассматриваемом алгоритме. Описание компонентов данных необходимо для понимания процесса transfer learning в ITGP.

1. *Векторы веса в ITGP*: В ITGP наборы весовых векторов инициализируются КММ и развиваются DE. Как показано на рисунке:



Эти векторы веса имеют фиксированную длину, которая равна числу выбранных экземпляров из исходных данных. Длина обычно меньше, чем количество исходных экземпляров исходных данных. Каждый вектор веса представляет собой подмножество взвешенных экземпляров домена источника (логично.). При этом различные весовые векторы в наборе из DE относятся к разным подмножествам экземпляров исходного домена с различными весовыми коэффициентами (обратите внимание, что в ходе алгоритма индексы экземпляров не меняются). Это ключевое отличие от вектора экземпляров, представленного в [11], где векторы весов представляют различные векторы для одного и того же экземпляра.

1. *Вектор соответствия, содержащий значения ошибок в исходных данных:* другой компонентой данных является вектор соответствия (жесткости?, пригодности?), который также показан на рисунке выше. Как мы можем видеть, ITGP рассматривает производительность модели на исходных данных, используя все весовые коэффициенты, предоставленные DE (это ). Таким образом, в отличие от действительного числа, обозначающего значение ошибки в стандартном GP, соответствие (fitness) модели в ITGP – вектор, имеющий m измерений, то есть m – количество векторов веса. Каждое измерение представляет собой взвешенную ошибку в выбранных элементах исходных данных.

Подробная информация о том, как получить взвешенную ошибку, находится в разделе III-D.

C. KMM для инициализации исходных векторов и выбора экземпляров исходного домена

В ITGP KMM [14], который является хорошо известным алгоритмом оценки отношения плотности, используется для выбора экземпляра из исходных данных и обеспечения изначального набора, который будет состоять из набора весовых векторов для DE. KMM базируется на сопоставлении моментов в бесконечном порядке и эффективном сопоставлении всех моментов, используя функции ядра (о чем речь в этом предложении??). KMM оценивает отношение плотности путем минимизации максимального среднего расхождения (MMD) между взвешенным распределением и распределением в воспроизведении ядра гильбертова пространства (??? RKHS, Reproducing kernel Hilbert space), которое является картой особенностей из в . Используя две матрицы ядер Гамма (чаще всего используется радиально-базисная функция (RBF):

(1)

И

(2)

(выше это как раз эти матрицы ядер)

Где – экземпляр исходного домена, – экземпляр таргетного домена, – ядро, - номера (или количество?) исходных и таргетных экземпляров соответственно. MMD можно получить с помощью следующей формулы:

И подходящий набор весов можно найти, решив квадратичную задачу путем минимизации :

Где и ,

– граничное (видимо, сверху) значение для . Как показано в алгоритме ниже, предоставляя m различных значений (обычно , как предположено в [14], KMM получит набор весов. Чтобы предотвратить доминирование данных в исходной области в процессе обучения, выбираем лишь фиксированное количество экземпляров исходной области, имеющих наибольшие веса. Это число выбранных экземпляров исходного домена выбирается с помощью (это количество элементов таргетного домена) и параметра масштаба , который определяется из . Обычно у t небольшое значение. Ряд экспериментов показывает, что значение является хорошим выбором. В этой статье для задач, в которых количество доступных экземпляров таргетного домена небольшое, устанавливается равным 3. Иначе – равным 2 (то есть обратная зависимость). Что еще более важно, за счет увеличения числа экземпляров исходного домена при сохранении действительно важных для данного обучения (оцененных по КММ), произойдет уменьшение пространства поиска алгоритму DE. Таким образом, потенциально может быть повышена эффективность и результативность поиска оптимального вектора веса.

**Алгоритм: KMM для инициализации векторов весов и выбора нужных экземпляров в ITGP**

*Вход:* – какое-то количество экземпляров начальных данных из области , таргетных экземпляров , параметр масштаба , количество векторов веса , набор ;

*Выход*: набор векторов веса

**for**  to  **do** получение весов экземпляров из исходных доменов с помощью КММ и выбор исходных экземпляров

Инициализируем длинами из

Ищем из формулы выше (1), с использованием радиально-базисной функции;

Ищем с помощью формулы (2)

Вычисляем

Ищем = решатель квадратичной задачи() – по уравнению (3)

Ранжируем , где – где – индексы исходных экземпляров;

**for**  to  **do**

Добавляем в ;

Возвращаем ;

D. Эволюционный процесс ITGP (сам процесс алгоритма)

Допустим, у нас уже есть начальные входные данные для алгоритма, сгенерированные GP и DE, то есть мы имеем деревьев (моделей) в GP и m весовых векторов в DE, где каждый вектор содержит n измерений, соответствующих весам для n выбранных экземпляров исходных данных. Тогда процесс эволюции векторов, показанный на рисунке в оригинальной статье, состоит из следующих этапов:

1. GP оценивает элементов/моделей на выбранных экземплярах исходных данных, и для каждой модели существует взвешенных значений ошибок (соответствующих весовым векторам).
2. GP выбирает моделей, сравнивая векторы плотности (? Или векторы соответствия ?) всех моделей в каждом измерении (в каждой компоненте вектора, видимо). Для каждого измерения k мы можем найти модель с наименьшей взвешенной ошибкой . Эта модель считается лучшей в измерении k. Иначе говоря, лучшая модель в измерении k имеет наименьшее .
3. GP повторно оценивает m моделей, измеряя их ошибки уже на экземплярах целевой (таргетной) области.
4. DE собирает обратную связь от GP, то есть повторно оценивает ошибки m моделей, как полноту весовых векторов.
5. DE генерирует некоторую совокупность векторов веса следа (?? О каком следе речь? Возможно, имеется в виду вектор на текущей итерации, который был изменен с увеличением числа итераций), используя стратегии мутации и оператор кроссовера (какие стратегии?)
6. Повторяем шаги 1-4, используя векторы весов следа (?) и получаем их значения.
7. DE генерирует новое поколение векторов веса, сравнивая значения весовых векторов и соответствующих им значений следа, а GP обновляет топ-m экземпляров в соответствии с выжившими (видимо, имеется в виду, что веса соответствуют установленным рамкам и «выжили» в проверке на эти ограничения) весовыми векторами.

E. Оценка в ITGP

Процесс оценки (видимо, полученных результатов) в ITGP состоит из двух частей – оценки исходных данных и повторной оценки целевых (таргетных) данных для обучения. Суть оценивания состоит в том, чтобы использовать производительность GP на взвешенных исходных экземплярах, при этом повторная оценка измеряет, как наилучшие варианты GP (??) работают в целевой области (то есть сначала пробуем на исходных данных, а потом на таргетных и проверяем как на таргетных работает ???).

Во время такого процесса используются две функции соответствия (fitness):

1. *Оценка исходных данных.* При оценке моделей GP в ITGP взвешенная ошибка данных, которые были выбраны из исходных, измеряется как взвешенная среднеквадратичная ошибка (WMSE – weighted mean squared error, там используется предиктор-корректор?), которая вычисляется как , где – количество выбранных экземпляров исходного домена, – вес -ого выбранного экземпляра, – выход модели (выходные данные в смысле), – таргетное значение -ого экземпляра (о чем тут речь? Таргетное значение это то, которое мы уже знаем и сравниваем выход с реальным, или тут сравниваются два набора данных – исходных и таргетных). Каждый вектор соответствия (жесткости?) имеет компонент: . Каждое из значений получается с помощью моделей на наборе моделей данных, выбранных из исходных, с соответствующими им весами.
2. *Переоценка целевых данных.* Чтобы измерить производительность (точность?) моделей, обученных на основе взвешенных исходных экземпляров, в целевой области и производительность векторов/DE веса, ITGP выбирает моделей для повторной оценки. Эти моделей имеют наименьшие соответственно. Они являются лучшими GP-элементами по каждому весовому вектору и их эффективность в обучении по целевым данным сильно зависит от того, сколько различий может скорректировать вектор веса между распределением исходных данных и таргетных доменов. Таким образом, ошибка обучения этих моделей на целевых данных может рассматриваться как хороший показатель точности векторов веса. В процессе переоценки обычно используется мера жесткости (fitness, мб мера соответствия) – среднеквадратичная ошибка (MSE):

*(5)*

Где – количество целевых (таргетных) экземпляров для обучения, – прогнозируемое значение модели в -ом экземпляре для обучения, – соответствующее целевое значение.

F. Размножение в ITGP (разведение? О чем речь?)

В каждом новом поколении (на каждой новой итерации??) ITGP поддерживает две группы данных (экземпляров, составляющих) – GP-элементы/регрессионные модели и DE-наборы, представляющие набор весовых векторов. По сравнению со стандартным GP размножение регрессионных моделей в ITGP используют два новых механизма, а именно: по отделению лучших (элитарных) экземпляров (?? Видимо, имеется в виду в виду выбор изначальных экземпляров из всей source data) и как выбрать родителей для дальнейшего размножения. Элитные модели в ITGP это лучшие модели из победителей (?? По каким-то параметрам??), которые были подвергнуты процессу переоценки. С другой стороны, для генерации оставшихся элементов в новом поколении, то есть новых элементов, родители выбираются с использованием нового оператора «турнира».

Как было упомянуто в предыдущем разделе, вместо действительного числа, модель соответствия (fitness model) представляется в виде вектора жесткости (вектора соответствия), который содержит измерений, то есть . В связи с новым представлением этой самой модели соответствия в ITGP, мы расширили [определение] оригинального оператора выбора турнира (?? о чем речь??), чтобы сравнить векторы соответствия. Новый оператор выбора сравнивает векторы устойчивости элементов-кандидатов по ряду измерений, которые обозначаются как . В нашей предварительной работе [11] -измерения выбираются случайным образом, где каждое из исходных -измерений имеет одинаковую возможность определить победителя турнира (какого турнира…). На самом деле, важность каждого измерения, которое определяется качеством вектора веса, не одинакова. При отношении к ним одинаково, мы можем потерять хороших кандидатов в родители. Чтобы сравнить родителей-кандидатов в ITGP, D-измерения выбираются с помощью колеса рулетки. В частности, процесс выбора можно описать следующими шагами:

1. Шаг первый. GP-элементы отбираются случайным образом в соответствии с размером турнира.
2. Шаг второй. Измерения выбираются с помощью выбора колеса рулетки (это какой-то метод? Или рандомайзер?), то есть для определения вероятности подходящего измерения (что именно это измерение подходит (мб размерность) с наибольшей вероятностью) выбираются соответствия (фитнесы) весовых векторов.
3. Шаг 3. В парах сравниваются эти D-измерения (размерности) с величинами соответствия T-элементов. Тот, который имеет значения не хуже (равен или больше), чем все остальные величины соответствия для всех D-измерений является победителем и будет выбран как родитель.

Значение D связано с общим количеством измерений в векторе соответствий (или значении, речь все о том же фитнесе) в ITGP, то есть с m. Обычно для большего m требуется больший D. Значение D также в некоторой степени влияет на строгость отбора. Чем больше D, тем выше строгость, процесс фокусируется в основном на лучших элементах. Это уменьшает/ограничивает разнообразие GP-наборов и может привести к преждевременной сходимости алгоритма. Слишком маленький увеличит случайность процесса выбора и упустит хорошие решения. В этой статье D считается равным 3 в соответствии со сравнительно небольшим значением .

Между тем, для генерации новых весовых векторов в используются четыре общепринятые и эффективные стратегии генерации вектора следа (?? все еще неясно что за вектор такой), которые вычисляются как

(6)

(7)

(8)

(9)

(похоже на какой-то предикторно-корректорный метод, но это неточно)

Здесь индексы являются векторами случайно выбранных элементов. и являются коэффициентами масштаба, которые обычно имеют положительное значение для масштабирования вектора разности. Регулировку параметров, таких как и , следует делать рекомендованным способом в [27]. Процесс выбора для генерации нового набора в SaDE определяется сравнением между точками вектора соответствия на векторе трассы (может, трассировки? о чем речь?) и целевым вектором, которые определяются эффективностью соответствующих лучших элементов на целевых данных обучения (о чем предложение непонятно).

Таким образом, ITGP использует новую структуру взвешивания экземпляров, чтобы направлять эволюционный процесс путем развития весовых векторов. По сравнению с нашей предыдущей (предварительной) работой [11], используется новый метод выбора экземпляров и инициализации вектора весов, а также новый оператор выбора, который отсеивает родителей исходя из более точного сравнения. Таким образом, регрессионные модели и векторы веса учатся вместе с крепким взаимодействием друг с другом на нескольких этапах эволюционного процесса. С такими улучшениями алгоритма ожидается, что ITGP станет более эффективным для извлечения большого количества полезной информации из исходного домена для повышения производительности на целевых данных.

IV. Разработка эксперимента

А. Контрольные методы (измерения)

Для исследования эффективности предложенного метода в экспериментах его сравнивают с семью эталонными методами. Четыре из них являются методами GP, а остальные три – методы опорных векторов (SVR – support vector regression – методы обучения с учителем; у нас это было как метод искусственного базиса) [28]. Эти эталонные контрольные методы описываются следующим образом.

1. GP-Tar, тоже GP-метод, только использует только целевые данные для обучения.
2. GP-Comb, который является стандартным GP-методом, обучающимся на основе комбинации набора исходных данных и целевых данных.
3. TLGP, предложенный в работе [11] и основной ITGP. По сравнению с ITGP TLGP неявно выполняет выбор экземпляра в исходном домене и случайным образом инициализирует набор входных данных для DE. Сравнение между TLGP и ITGP состоит в том, чтобы подкрепить затраты ресурсов на поиск нужных экземпляров и инициализацию набора данных для генерации совокупности векторов веса, а также нового оператора выбора, основанного как на векторе соответствия весов, так и на векторе соответствия GP-элементов.
4. TLGP-NS, который является модифицированным методом TLGP с использованием нового оператора выбора. Сравнение между TLGP, TLGP-NS и ITGP должно подтвердить достижение нового оператора выбора (?? о чем предложение непонятно).
5. SVR-Tar и SVR-Comb, которые обозначают SVR с двумя видами данных для обучения. Производительность SVR с радиальным RBF только с использованием таргетных данных и комбинации данных в исходной и целевой областях используются для сравнения.
6. SVR-W, который относится к обучению SVR на основе комбинации данных из исходной и целевой областей с использованием весов, полученных KMM. Веса также используются в ITGP.

Мы также рассмотрели сравнение с TLGP с использованием KMM. Из-за ограничения в количестве страниц, мы показали дополнительное сравнение в дополнительном онлайн-материале этой статьи. Все методы GP реализуются в фреймворке ECJ GP. Методы SVR реализуются в R-пакетах “e1071” с использованием функции ядра RBF и других настроек по умолчанию.

B. Датасеты

В виду отсутствия тестовой проблемы для transfer learning в символьной регрессии, некоторые недавние связанные работы модифицировали традиционные регрессионные датасеты для тестирования методов transfer learning [17], [31]. В этой статье мы будем следовать этим работам и модифицировать 6 датасетов, в том числе 5 реальных проблем из хранилищ данных Delve2 и UCI [32] и один синтетический набор данных, сгенерированный известным Фридманом-1 [33].

В оригинальной задаче Фридмана каждый экземпляр имеет 10 переменных , где каждая компонента соответствует равномерному распределению . Но только первые 5 входных переменных относятся к выходной переменной :

Где и , параметры, и это нормальное распределение. Для этих параметров устанавливаются разные значения при генерации данных в исходной и целевой областях. Для исходных данных и , в то время как взяты из , а выбирается из для целевых (таргетных) данных [17].

В двух реальных задачах исходные и целевые данные представляют собой два датасета с одинаковыми пространствами признаков (областями определения?) и похожими задачами. Для трех других реальных датасетов, в соответствии с одной специфичной особенностью, различные значения, из которых обычно можно выделить распределение данных, датасеты делятся на исходные данные и таргетные данные (странное предложение). Затем одни и те же признаки в двух областях возмущаются (вносится погрешность) с помощью случайных чисел, следующих за различными нормальными распределениями. Это помогает убедиться, что распределение из двух экземпляров доменных данных различно, но не слишком разное, что является ключевым предположением для методов transfer learning в этой статье. Детальное представление реальных наборов данных заключается в следующем.

1. Набор данных о качестве вина (Wine), который предназначен для прогнозирования качества вина, для красных и белых образцов. Мы используем проблему прогнозирования качества для белого вина как исходные данные и прогноз качества красного вина в качестве задачи для целевой области.
2. Датасет Kin – семейство наборов данных. Эта статья использует n датасетов, (n для нелинейных), где “nm” (nonlinear medium variance – нелинейная средняя дисперсия) используется как набор данных исходной области, а “nh” (nonlinear high variance – нелинейная высокая дисперсия ??) – датасет таргетных данных. Особенности (фичи) в таргетных данных были возмущены с помощью рандомных чисел из распределения .
3. Набор данных об успеваемости учащихся (Student), который аппроксимирует успеваемость студентов среднего образования в двух школах Португалии. Два исходных набора данных обеспечивают представление двух предметов – математики и португальского языка. Набор данных, содержащий успехи в математике, рассматривается как исходные данные, в то время как набор для предсказания результатов по португальскому языку используется в качестве набора данных исходного домена.
4. Набор данных о жилье (House), который предназначен для прогнозирования “MEDV.” (что это за показатель? Медиана объема?). Набор данных делится на два, в соответствии с характеристикой «TAX.» (налог). Исходные данные – экземпляры с налогом меньше или равным 600, а целевые – экземпляры с налогом больше 600. Между тем, три характеристики: “RM.”, “AGE.”, “B” возмущены там, где исходные данные возмущены с помощью рандомных чисел из нормального распределения с параметрами 0,1, 0,1 - , в то время как таргетные данные возмущены случайными числами из распределений соответственно.
5. Датасет морского ушка (Abalone – это такие моллюски), который предсказывает возраст моллюска исходя из физических измерений. Оригинальный датасет имеет информацию об особях как мужского, так и женского пола. Мы рассматриваем мужские данные в качестве исходного домена и женские в качестве целевого.

Количество объектов (характеристик) и экземпляров во всех наборах данных подытожены в таблице 1 (см. ее в документе). Обращаем внимание, что разница между распределениями в двух областях в каждой задаче проверены с помощью многовариантного критерия Колмогорова-Смирнова (KS) [34]. Относительная квадратичная ошибка (RSE – relative square error, у нас это было как стандартная ошибка остатков) на целевых данных сообщается для более простых сравнений (о чем предложение?). RSE вычисляется как:

*(10)*

Где – i-ый целевой выход, – i-ый выход модели, – выборочное среднее выходных данных. Относительная ошибка показывает, как результаты модели кандидата (?? О чем речь?? Может, о предикторе-корректоре? Там сумма квадратов разности…) - – сравнивается с простым предиктором, используя среднее значение игреков: . Для каждого из 5 GP-методов, каждый алгоритм был независимо запущен 50 раз по каждой из 6 проблем. Значения параметров для всех методов GP суммируются в таблице II (см. документ), и большинство из них являются общими установленными параметрами для GP.

V. Результаты и их анализ

A. Успехи алгоритма на таргетных данных

Эффективность восьми алгоритмов сравнивается с использованием подходов transfer learning. В таблице III (см. документ) приводится сводная информация об относительных квадратичных ошибках (RSE), полученных с помощью обученных моделей на таргетных данных. В то время как медианные значения и стандартные отклонения RSE, полученные из 50 лучших моделей на таргетных тренировочных данных и таргетных тестовых данных были представлены для пяти методов GP; RSE полученный с помощью моделей SVR, также представлен. Поскольку методы опорных векторов (SVR) являются детерминированными методами обучения, для каждого набора данных был получен только один RSE. Обращаем внимание, что тренировочные RSE вычисляются только на целевых обучающих данных, которые не являются полным набором для обучения для GP-Comb, SVR-Comb и SVR-W. Два набора непараметрических значимых статистических тестов, критерий Уилкоксона (это что?) с уровнем значимости 0.05, представлены для сравнения тренировочных и тестовых RSE у лучших моделей ITGP (и GP-Tar) с другими методами GP. Z-тест используется для того, чтобы проверить, есть ли существенная разница между методами GP (с 50 группами результатов) и методов SVR (с одной группой результатов). В таблице III представлены два набора результатов статистических тестов, которые ясно показывают, может ли какой-то один метод значительно превзойти GP-Tar и ITGP. Сравнивая с GP-Tar, легко узнать, выполняет ли GP-метод позитивное transfer learning. «-» означает, что ITGP (GP-Tar) работает значительно хуже, чем сравниваемый метод, «+» - что ITGP (GP-Tar) – значительно лучше, а «=» означает отсутствие существенной разницы. «N/A» означает отсутствие сравнения, когда сравниваемым методом является сам ITGP (или GP-Tar).

1. *Сравнение на таргетных тренировочных данных.* В таблице III явно видно, что методы GP, оснащенные новым фреймворком, имеют похожий паттерн. У них обычно есть гораздо более высокая тренировочная ошибка чем у GP-Tar и SVR методов на рассмотренной проблеме. В частности, TLGP, TLGP-NS и ITGP имеют показатели тренировок хуже, чем у GP-Tar, по 5 из 6 наборов тренировочных данных, за исключением Фридмана; их показатели обучения (тренировки) хуже, чем у методов SVR на 4 из 6 наборов данных, кроме Стьюдента и Фридмана. При сравнении с GP-Comb, в датасетах Wine, Student и Friedman получилось, что они имеют меньшие ошибки при обучении, но это не распространяется на три других датасета. Согласно критерию статистической значимости (было где-то выше?), все различия в эффекте обучения являются значимыми. Результаты на целевых (таргетных) данных не являются неожиданными. Причина в том, что данные исходного и таргетного домена подчиняются разным распределениям, и в ITGP (то же самое для TLGP и TLGP-NS) таргетные обучающие экземпляры вовлечены в процесс тренировки только с помощью выбора моделей и оценки векторов веса. По сравнению с методами, где эти экземпляры непосредственно используются в обучении моделей регрессии, весьма вероятно, что ITGP (и два других метода GP) будут иметь менее эффективное обучение, соответствующее данным целевой области.

*Среди методов GP:* с учетом эффективности обучения из трех GP с методами взвешивания экземпляров TLGP-NS имеет немного лучшую тренировочную способность, чем TLGP, который указывает, что новый оператор отбора помогает направлять процесс обучения к регрессионным моделям, которые лучше учатся в целевой области, устанавливая более высокую вероятность рассмотрения вектора веса. Новый предложенный метод ITGP может иметь лучшую производительность обучения, чем TLGP и TLGP-NS в большинстве рассмотренных случаев. На пяти из шести наборов данных ITGP имеет несколько лучшие результаты обучения, чем TLGP и TLGP-NS, в то время как на датасете Abalone ITGP показывает лучшие результаты, чем два других метода. Это указывает на то, что при добавлении в алгоритм возможности выбора экземпляра, которая стремится сохранить более важные экземпляры исходного домена при отбрасывании менее полезных, это помогает выпустить (release, может тут в значении убрать, снизить) доминирование данных исходного домена в течение процесса обучения. Однако ошибка обучения в ITGP все еще намного выше, чем у GP-Tar в большинстве случаев.

*GP против SVR:* сравнение методов GP и методов SVR, где используются одни и те же данные для обучения, то есть сравнения между GP-Tar и SVR-Tar или GP-Comb и SVR-Comb показывает, что на Wine, Kin и Abalone, где доступно большое количество экземпляров таргетного домена, GP-Tar имеет более высокую ошибку, чем SVR-Tar. Тем не менее, на трех других наборах данных, где существует относительно меньшее количество обучающих экземпляров (для тренировки), GP-Tar превосходит SVR-Tar. При воздействии на комбинацию данных исходного и целевого домена GP-Comb имеет лучшие результаты в обучении, чем SVR-Comb на Kin и Abalone, но это не относится к другим четырем наборам данных. Все результаты показывают, что по сравнению с SVR GP хорошо учится на меньшем количестве экземпляров.

*Сравнение методов без использования данных исходного домена:* сравнивая методы, которые были обучены непосредственно из обоих доменов (исходного и целевого) с методом, который обучается только на целевых данных, то есть GP-Comb против GP-Tar и SVR-Comb по сравнению с против SVR-Tar, GP-Comb и SVR-Comb обычно имеют худшие тренировочные показатели, чем их аналоги, обучающиеся только из таргетных данных напрямую. Это особенно касается Фридмана, где разница велика разница у распределений исходного и целевого доменов. Худшие характеристики у GP-Comb и SVR-Comb, то есть при обучении непосредственно из данных исходного домена, в основном в сложности понимания, какие именно данные необходимо передать, без какой-то стратегии transfer learning. Также можно отметить, что доминироание экземпляров исходного домена в процессе обучения также является потенциальной причиной. В этом случае SVR-W с взвешенными экземплярами исходного домена может эффективно улучшить производительность обучения в некоторых случаях (но не во всех), например, в датасетах Kin и Friedman.