“отрицательная”. Данную задачу можно рассматривать как задачу классификации на три и два класса соответственно, далее мы будем рассматривать задачу с двумя возможными вариантами тональной оценки, так как задача классификации на три и более класса является более сложной в техническом отношении. Для решения задачи классификации эффективными являются методы машинного обучения с учителем.

Для того, чтобы методы решения задач классификации можно было применить для анализа тональности текста, необходимо текст представить в виде математического вектора. С этой целью применяется векторная модель “мешок слов” - модель текста, предложенная в 1975 году Дж. Солтоном, и в настоящее время одной из самых распространенных в различных областях лингвистических исследований. Текст в данной векторной модели рассматривается как неупорядоченное множество слов. Вектор, являющийся модельным представлением текста в векторном пространстве, образуется упорядочением весов всех слов (включая те, которых нет в конкретном тексте). Размерность этого вектора равна количеству различных слов во всей коллекции, и является одинаковой для всех текстов коллекции.

Так как при применении модели «мешок слов» теряется информации о позиции слов относительно друг друга, а эта информация может влиять на качество получаемой модели, то целесообразно применять модель не “мешок слов”, а “мешок N-грамм”. N-грамма - последовательность из N элементов (слов). Последовательность из трех элементов называют триграмма, последовательность из двух элементов называется биграмма. N-граммы меньшей длины, как правило, дают лучшие результаты, чем N-граммы большей длины, т.к. обучающая выборка в большинстве случаев недостаточно большая для нахождения статистических закономерностей N-грамм большой длины. Для многих практических приложений можно получить хороший результат, используя в качестве признаков одиночные слова[1], биграммы и триграммы[3].

Модель “мешок слов” некорректно работает со словами, меняющими тональность выражения на противоположное. Например, фразы “мне нравится этот фильм” и “мне не нравится этот фильм” будут иметь положительную тональность, хотя у второй фразы она должна быть отрицательной. Чтобы решить эту проблему, можно объединять слово “не” со следующим словом, в результате в данном примере мы получим слово “не-нравится” и модель будет работать корректно[3]. Также эту проблему можно решать при помощи N-грамм, но, как правило, это вынуждает использовать N-граммы большей длины.

Для ликвидации неоднозначности, вызванной возможностью одного и того же слова быть различными частями речи, применяется тегирование частей речи - определение для каждого слова в предложении его части речи по положению в предложении и/или грамматической форме[4].

Полученную задачу классификации можно решить различными методами машинного обучения: наивный байесовский классификатор, логистическая регрессия, метод опорных векторов, методы нейронных сетей и т.д. Сравнив их временную сложность, качество полученных моделей, масштабируемость можно выбрать наиболее подходящий для конкретных данных и конкретной задачи[1].

Список использованных источников:

1. Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. Thumbs up? Sentiment classification using machine learning techniques. In Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 79–86, 2002.
2. Kushal Dave, Steve Lawrence, and David M. Pennock. Mining the peanut gallery: Opinion extraction and semantic classification of product reviews. In Proceedings of WWW, pages 519–528, 2003.
3. Sanjiv Das and Mike Chen. Yahoo! for Amazon: Extracting market sentiment from stock message boards. In Proceedings of the Asia Pacific Finance Association Annual Conference (APFA), 2001.
4. Yorick Wilks and Mark Stevenson. The grammar of sense: Using part-of-speech tags as a first step in semantic disambiguation. Journal of Natural Language Engineering, 4(2):135–144, 1998.

ГЕНЕТИЧЕСКИЕ АЛГОРИТМЫ И ТЕХНИКИ ОПТИМИЗАЦИИ

Белорусский государственный университет информатики и радиоэлектроники
г. Минск, Республика Беларусь

Гутковский В.Н.

Жвакина А. В. – канд. техн. наук, доцент

Генетические алгоритмы − адаптивные методы, которые могут быть использованы для решения проблем поиска и оптимизации, основанные на генетических процессах биологических организмов. В данном докладе анализируются принцип действия генетических алгоритмов и техники их оптимизации.

Генетические алгоритмы используют прямую аналогию с естественным поведением. Они работают с набором особей, каждая из которых представляет собой возможное решение проблемы. Каждой особи присваивается оценка ее пригодности в соответствии с тем, насколько хорошее решение оно представляет. Самым пригодным особям дают возможность воспроизведения путем скрещивания с другими особями набора. Это порождает новых особей как потомков, которые имеют некоторые характеристики, взятые у каждого из родителей. Наименее пригодные члены с меньшей вероятностью будут отобраны для размножения.

Таким образом, создается полностью новая совокупность возможных решений, путем выбора лучших особей из нынешнего поколения и объединяя их для создания нового набора. Это новое поколение содержит

более высокую долю характеристик, которыми обладают хорошие представители предыдущего поколения. Таким образом, в течение многих поколений хорошие характеристики распространяются по всему набору, будучи смешанными и обменивающимися с другими хорошими характеристиками по мере их поступления.

Существующие специализированные методы решения конкретных проблем, вероятно, будут превосходить генетические алгоритмы как с точки зрения скорости, так и с точностью конечного результата. Генетические алгоритмы используются в трудных областях, где таких методов не существует. Даже те методы, которые работают хорошо возможно улучшить путем их скрещивания с принципами генетических алгоритмов.

Перед использованием генетических алгоритмов необходимо разработать подходящее кодирование или представление проблемы. Также нам понадобится функция «fitness», которая присваивает каждому решению оценку его пригодности. Во время работы алгоритма должны быть отобраны родители для размножения и рекомбинированы для создания потомства.

Предполагается, что потенциальное решение проблемы может быть представлено в виде набора параметров. Эти параметры, известные как гены, объединяются вместе, чтобы сформировать строку значений, часто называемую хромосомой.

Функция оценки пригодности («fitness») должна быть разработана отдельно для каждой решаемой задачи. Принимая определенную хромосому, функция возвращает числовую оценку пригодности, которая должна быть пропорциональна полезности или способностям особи, которую эта хромосома представляет.

Во время репродуктивной фазы ГА особи выбираются из популяции и рекомбинируются, производя потомство, которое будет включать следующее поколение. Родители выбираются случайным образом из популяции, используя схему, которая благоприятствует наиболее подходящим индивидам. Хорошие особи, вероятно, будут выбраны несколько раз, тогда как менее пригодные особи, возможно, не будут вообще. Выбрав двух родителей, их хромосомы рекомбинируют, обычно используя механизмы скрещивания и мутации.

Скрещивание делит их хромосомные строки в некотором случайно выбранном положении, чтобы произвести два головных и два конечных сегмента. Затем конечные сегменты меняются местами для получения двух новых полноразмерных хромосом. Такое скрещивание называется одноточечным. Мутация применяется к каждому потомку отдельно после скрещивания. Она случайным образом изменяет каждый ген с небольшой вероятностью.

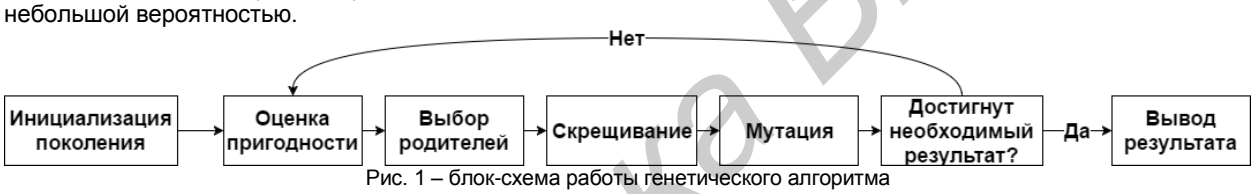

Рис. 1 – блок-схема работы генетического алгоритма

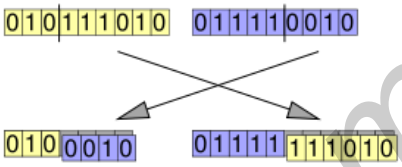

Рис. 2 - одноточечное скрещивание

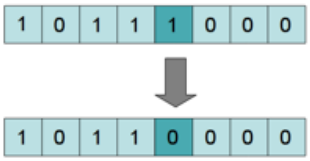

Рис. 3 – мутация

Если ГА будет правильно внедрен, набор будет развиваться в течение последующих поколений, так что пригодность наилучшей и средней особи в каждом поколении стремится к глобальному оптимуму. Конвергенция - это развитие, направленное в сторону повышения единообразия. Ген сходится, когда доля набора имеет одно и то же значение. Соответственно, популяция сходится, когда все гены сходятся.

Аналогичные техники:

- Случайный поиск. Сложность полного перебора зависит от количества всех возможных решений задачи. Если пространство решений очень велико, то полный перебор может не дать результатов в течение нескольких лет или даже столетий.
- Алгоритм имитации отжига. Начиная с произвольной точки в пространстве поиска происходит случайный шаг, если этот шаг ведет нас к высшей точке, он принимается, а если он ведет нас к нижней точке, он принимается только с вероятностью p(t), где t-время. В начале поиска функция принимает значение 1, но постепенно сводится к нулю. Как и случайный поиск, имитация отжига работает только с одним вариантом решения в отдельный момент времени, в то время и поэтому не выстраивает общей картины пространства поиска. Также никакая информация с предыдущих ходов не учитывается при следующем ходе.

Техники оптимизации скрещивания:

1. Двухточечное скрещивание. Двухточечное скрещивание делит хромосомные строки в двух случайно выбранных положениях. Затем двое родителей обмениваются сегментами, находящимися между этими положениями, тем самым образуя двое новых потомков. При этом, строка преобразуется в петлю для корректной обработки ситуаций, когда вторая точка конца расположена ближе к началу, чем точка начала.
2. Универсальное скрещивание. В основе универсального скрещивания лежит использования маски скрещивания. Маска скрещивания представляет собой сгенерированную случайным образом для каждой пары родителей строку бинарных значений той же длины, что и хромосома родителей. С помощью этой маски производится скрещивание хромосом родителей. Каждый ген потомка наследуется от одного из родителей согласно маске скрещивания. Если в

некотором положении ген маски принимает значение 1, то ген потомка в том же положении наследуется от первого родителя, если 0 – то второго.

Недостатки:

1. Проблема определения функции оценки пригодности. Решающее значение в эффективности генетических алгоритмов принимает функция оценки пригодности и схема кодирования. Общее правило построения функции оценки является то, что оно должно отражать пригодность хромосом посредством специализированных методов.
   К сожалению, пригодность хромосомы, рассчитанная специализированным методом, не всегда является полезной величиной для генетического поиска. В задачах комбинаторной оптимизации, где есть много ограничений, большинство точек в поисках места зачастую представляют недопустимые хромосомы и, следовательно, имеют нулевую пригодность.
2. Проблема локального оптимума заключается в том, что гены нескольких хромосом с сравнительно высокой пригодностью (но не оптимальной), особи могут доминировать в поколении, заставляя пригодность хромосом следующего поколения приблизиться к локальному максимуму. Как только популяция сошлась, способность генетического алгоритма найти лучшие решения фактически исключены, т.к. скрещивание почти идентичных хромосом производит мало нового. Остаются только мутации, которые просто выполняют медленный, случайный поиск. Для того, чтобы избежать таких ситуаций, мы должны изменить способ выбора особей для размножения.
3. Проблема завершения работы алгоритма. После многих поколений, популяция в большей степени сошлась, но глобальный максимум не достигнут. Средняя стоимость пригодности поколения будет высокой и разница пригодности между лучшими и средними особями будет небольшой. Следовательно, функция оценки пригодности следует оптимизировать, чтобы алгоритм достиг глобального максимума.

Генетические алгоритмы активно используются при оптимизации функций, при решении разнообразных задач на графах, задач компоновки и при настройке и обучении нейронных сетей.

Исследованиеподдержанопроектом CERES. Centers of Excellence for young RESearchers (Reg.no. 544137-TEMPUS-1-2013-SK-JPHES),

Co-funded by the Tempus Programme of the European Union

Список использованных источников:

1. Рутковская Д., Пилиньский М., Рутковский Л. Нейронные сети, генетические алгоритмы и нечеткие системы — 2-е изд. — С. 452.
2. Гладков Л. А., Курейчик В. В., Курейчик В. М. Генетические алгоритмы: Учебное пособие. — 2-е изд. — С. 320.
3. Скобцов Ю. А. Основы эволюционных вычислений. — Донецк 2008. — С. 326.
4. Schmitt, Lothar M; Nehaniv, Chrystopher L; Fujii, Robert H (1998), Linear analysis of genetic algorithms, Theoretical Computer Science 208 – С. 111–148
5. Schmitt, Lothar M (2001), Theory of Genetic Algorithms, Theoretical Computer Science 259 – С. 1–61

АРХИТЕКТУРА REDUX

Белорусский государственный университет информатики и радиоэлектроники
г. Минск, Республика Беларусь

Казаков С.Г.

Сиротко С.И. – канд. физ.-мат. наук, доцент

С развитием интернета, скорости соединения и возможностями браузеров, к одностраничным веб приложениям (SPA — single page application) увеличились требования, из-за чего нам необходимо управлять и поддерживать все больше и больше событий и состояний. Состояния включают в себя кэшировние данные, ответы сервера, а также данные созданные локально, но еще не сохраненные на сервере. Это касается и UI (User Interface) состояниям, таким как текущий путь (route), выбранная вкладка, отображение загрузки или навигация на страницах.

Основные проблемы современных приложения сводятся к увеличению сложности, что порождает ошибки программы из-за смешивания двух концепций асинхронность и изменение состояния. Другие решения справляются с проблемой на уровне отображения, убирая асинхронность и прямое манипулирование DOM. Архитектура Redux делает изменения состояния приложения предсказуемыми, путем введения некоторых ограничений на то, как и когда могут произойти обновления.

Redux является предсказуемым контейнером состояния для JavaScript приложений. Такая архитектура подразумевает создание нового состояния приложения при каждом изменении или действии, таким образом