Министерство образования и науки РФ
Федеральное государственное автономное образовательное
учреждение высшего профессионального образования
«Уральский федеральный университет
имени первого Президента России Б. Н. Ельцина»
Институт фундаментального образования
Кафедра интеллектуальных информационных технологий

К ЗАЩИТЕ ДОПУСТИТЬ

Заведующий кафедрой ИИТ

_____ И. Н. Обабков

« ____»_____20___ г.

ПОСТРОЕНИЕ ПРИЗНАКОВ ДЛЯ КЛАССИФИКАЦИИ МЕДИЦИНСКИХ ИЗОБРАЖЕНИЙ ПОСРЕДСТВОМ ГЕНЕТИЧЕСКОГО ПРОГРАММИРОВАНИЯ

МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ

Пояснительная записка

Руководитель А. А. Мокрушин

Нормоконтролер И. М. Гайнияров

Студент гр. ФОМ–251101 А. А. Ибакаева

Екатеринбург – 2017

РЕФЕРАТ

Выпускная квалификационная работа на соискание академической степени магистра 75 с., 8 рис., 52 источника.

КЛАССИФИКАЦИЯ МЕДИЦИНСКИХ ИЗОБРАЖЕНИЙ, ПОСТРОЕНИЕ ПРИЗНАКОВ, ГЕНЕТИЧЕСКОЕ ПРОГРАММИРОВАНИЕ

Объект исследования – классификация изображений.

Предмет исследования – алгоритм построения признаков для классификации.

Цель работы – нахождение и отработка методики построения признаков для улучшения точности классификации.

Методы исследования: исследование предметной области, формализация задачи, анализ программного обеспечения.

Результаты работы: разработан алгоритм построения признаков для классификации.

Выпускная квалификационная работа выполнена в текстовом редакторе Microsoft Word и представлена в твердой копии.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

В последнее время одним из актуальных направлений развития компьютерных технологий в медицине становится обработка цифровых изображений. Медицинские изображения – это структурно-функциональный образ органов человека, предназначенный для диагностики заболеваний и изучения анатомо-физиологической картины организма. Распознавание патологических процессов является одной из наиболее важных предпосылок для начала своевременного лечения заболевания.

Классификация заключается в прогнозировании значения категориального атрибута (класса) на основе значений других атрибутов. Цель классификации – взять объект, оцениваемый набором признаков, и назначить его одному дискретному классу. Каждый экземпляр данных определяется в соответствии с набором атрибутов или переменных.

Для решения задачи классификации выполняется определенный вид предварительной обработки, известной как преобразование представления, которое, учитывая исходный вектор признаков F_0 и обучающее множество L, состоит в создании представления F, полученного из F_0, которое максимизирует некоторый критерий, и, по крайней мере, так же хорошо, как F_0 по отношению к этому критерию.

Разработка хорошего пространства признаков является предпосылкой для достижения высокой производительности в любой задаче машинного обучения. Однако часто неясно, каким должно быть оптимальное представление признака. В результате общий подход заключается в том, чтобы использовать все доступные атрибуты в качестве признаков и оставить проблему идентификации полезных наборов признаков модели обучения. Такой упрощенный подход не всегда работает хорошо. С развитием технологий аппаратного и программного обеспечения и увеличением объема данных количество признаков, используемых системами машинного обучения, измеряется десятками тысяч и миллионами признаков.

Для решения проблем нерелевантности признаков и взаимодействия признаков используются методы построения признаков. Конструирование признака включает в себя преобразование заданного набора входных признаков для создания нового набора более мощных признаков, которые затем используются для прогнозирования. Поскольку вновь созданные признаки учитывают взаимодействия в предыдущем пространстве признаков, они более значимы и приводят к более кратким и точным классификаторам.

В данной работе применяется генетическое программирование для автоматического конструирования признаков с целью повышения различающей эффективности классификатора.

Актуальность выбранной темы обосновывается тем, что своевременное распознавание патологических процессов в организме человека приведет к оказанию необходимой медицинской помощи. Проблема классификации патологических процессов по данным медицинских изображений не может быть правильно решена, если важные взаимодействия и отношения между оригинальными признаками, не принимаются во внимание. Таким образом, многие исследователи согласились, что выделение признаков является наиболее важным ключом к любому распознаванию образов и проблеме классификации: «Точный выбор признаков, пожалуй, самая сложная задача распознавания образов» [26], «идеальная функция извлечения даст представление, что сделает работу классификатора тривиальной» [6]. В большинстве случаев выделение признаков осуществляется человеком на основе знаний исследователя, опыта и / или интуиции.

Проблема. Эффективность работы классификатора сильно зависит от входного множества признаков. Как выбрать оптимальное множество признаков для классификатора?

Цель – нахождение и отработка методики построения признаков для улучшения точности классификации.

Для достижения поставленной цели необходимо выполнить следующие задачи:

– изучить соответствующую литературу;

– разработать алгоритм построения признаков;

– реализовать программу по данному алгоритму;

– оценить эффективность работы алгоритма и сравнить с результатами классификации без построения признаков;

– обобщить полученные результаты и сделать соответствующие выводы.

Объектом исследования выступает классификация изображений. Предметом исследования является алгоритм построения признаков для классификации.

Основные результаты:

– разработан метод построения признаков для классификации;

– оценен результат применения метода генетического программирования для построения признаков.

Практическая значимость. Результаты данной работы могут быть применены в медицинских учреждениях для распознавания опухолей.

Структура работы. Диссертационная работа включает введение, две главы, заключение и список использованных источников.

1 Теоретическая часть

1.1 Постановка задачи

Для установления диагноза и выбора лечения врачи могут использовать медицинские изображения. Медицинские изображения – это структурно-функциональный образ органов человека, предназначенный для диагностики заболеваний и изучения анатомо-физиологической картины организма. Диагностика медицинских изображений предполагает выявление патологий у пациентов. Распознавание патологических процессов является одной из наиболее важных задач обработки и анализа медицинских изображений.

Необходимо решить следующую задачу: установить на основе медицинского изображения, является ли данное изображение патологическим или нет.

Для решения поставленной задачи требуется разработать алгоритм и реализовать программу по нему, которая будет точно классифицировать входные данные в виде медицинских изображений на классы.

Классификатору для эффективного разделения необходимо предоставить признаки, значения которых наиболее точно будут разделять пространство входных данных на классы.

Поиск такого набора признаков будет осуществляться с помощью генетического программирования, которое позволяет получить набор сконструированных и наиболее точно решающих задачу классификации признаков, вид которых заранее неизвестен.

Следовательно, требуется разработать алгоритм и реализовать программу по нему, который решает задачу построения множества признаков для эффективной классификации медицинских изображений методом генетического программирования.

1.1.1 Формальные определения

Пусть:

1. $x \in X$, где X – область входных значений.
2. $y \in Y$, где Y – область выходных значений.
3. S – набор обучающих примеров таких, что

$$S = \{(x_i, y_i)\}_{i=1\ldots m}. \quad (1)$$

4. S' – набор тестовых примеров таких, что

$$S' = \{(x_i, y_i)\}_{i=1..m'}. \quad (2)$$

5. Err_h будет ошибкой гипотезы h по сравнению с истинной основной гипотезой h_T.

Мы можем рассматривать каждый x как вектор фиксированной длины значений признаков в исходном пространстве признаков, т.е.

$$x = < f_1, f_2, f_3, \ldots, f_n >, \quad (3)$$

где $f_i \in F_0 \; \forall \; i = 1\ldots n$ и $n = |F_0|$.

Цель любой обучающей машины – узнать предиктор $h: X -> Y$ из S, с маленькой ошибкой Err_h. В парадигме построения признака каждый исходный вектор признаков x преобразуется в новый вектор признаков

$$x' = \varphi(x) = < \varphi_1(x), \varphi_2(x), \varphi_3(x), \ldots, \varphi_N(x) >, \quad (4)$$

где $\varphi_i \in F_T \; \forall \; i = 1\ldots N$ и $N = |F_T|$.

Каждое преобразованное значение признака $\varphi_i(x)$ получается путем оценки некоторой функции по всем исходным f_i. Мы хотим вывести гипотезу $h': \varphi(x) -> Y$, предполагая, что ее истинная ошибка $Err_{h'}$ меньше Err_h. В большинстве практических сценариев Err_h и $Err_{h'}$ будут вычисляться путем измерения производительности h и h' на тестовом множестве S'.

1.1.2 Требования к системе

Создаваемый алгоритм должен удовлетворять следующим критериям:

– на основе входных данных в виде множества оригинальных признаков, обучающего и тестового множеств исходных медицинских

изображений, набора классификаторов, а также максимально возможной погрешности выдавать результат в виде множеств сконструированных признаков, которые обеспечивают точность классификации не меньше заданной;

– иметь возможность изменения параметров генетического программирования – мощность начальной популяции, максимальная глубина дерева, вероятности скрещивания, репродукции и мутации;

– быть удобной в использовании, т.е. предоставлять промежуточные и конечные результаты работы в удобном для пользователя виде.

1.2 Обзор литературы

1.2.1 Классификация

Классификация является одной из наиболее изученных проблем машинного обучения и интеллектуального анализа данных [14], [46]. Она состоит в прогнозировании значения категориального атрибута (класса) на основе значений других атрибутов (предсказывающих элементов). Алгоритм поиска используется для индукции классификатора из набора правильно классифицированных экземпляров данных, которые называются обучающей выборкой. Другой набор правильно классифицированных экземпляров данных, известных как множество испытаний, используется для измерения качества полученного классификатора. Различные виды моделей, такие как деревья решений или правила, могут быть использованы для представления классификаторов.

Классификация объекта – номер или наименование класса, выдаваемый алгоритмом классификации в результате его применения к данному конкретному объекту.

В машинном обучении задача классификации относится к разделу обучения с учителем. В контролируемом обучении атрибуты экземпляров данных разделяются на два типа: входы или независимые переменные и

выходы или зависимые переменные. Цель процесса обучения состоит в предсказании значения выходов из значения входов. Для того, чтобы достичь этой цели, обучающий набор данных (в том числе экземпляры данных значений входных и выходных переменных с известными значениями) используется для управления процессом обучения. Регрессия и классификация – это два типа контролируемых задач обучения. В регрессии предсказываются непрерывные выходные значения, в то время как при классификации выходы дискретны. В неконтролируемом обучении не существует никакого различия по типу между переменными экземпляров данных. Как следствие, мы не можем говорить об обучающих данных, так как мы не можем иметь набор данных с известным выходом. Целью неконтролируемого обучения является нахождение внутренней структуры, отношений, или родства, которые могут присутствовать в данных. Примерами неконтролируемых задач обучения являются кластеризация и обнаружение ассоциации. Цель кластеризации состоит в том, чтобы разделить данные на различные группы, находя наборы данных, которые сильно отличаются друг от друга, либо члены, которых очень похожи друг на друга. Цель ассоциации состоит в нахождении значений данных, которые часто появляются вместе.

1.2.1.1 Формальная постановка задачи классификации

Пусть X – множество описаний объектов (признаков), Y – конечное множество номеров (имён, меток) классов. Существует неизвестная целевая зависимость – отображение $y^*: X \to Y$, значения которой известны только на объектах конечной обучающей выборки $X^m = \{(x_1, y_1), \dots, (x_m, y_m)$. Требуется построить алгоритм $a: X \to Y$, способный классифицировать произвольный объект $x \in X$.

Однако более общей считается вероятностная постановка задачи. Предполагается, что множество пар «объект, класс» $X \times Y$ является

вероятностным пространством с неизвестной вероятностной мерой P. Имеется конечная обучающая выборка наблюдений $X^m = \{(x_1, y_1), \dots, (x_m, y_m)$, созданная согласно вероятностной мере P. Требуется построить алгоритм $a: X \to Y$, способный классифицировать произвольный объект $x \in X$.

1.2.1.2 Классификаторы

Для того, чтобы решить задачу классификации, необходимо получить классификатор. Классификатор представляет собой модель, кодирующую набор критериев, которые позволяют экземплярам данных быть отнесенными к определенному классу в зависимости от значения некоторых переменных.

1.2.1.2.1 Классификатор ближайших соседей

Классификатор ближайших соседей (англ. k-nearest neighbors algorithm) [4] представляет собой непараметрический, нелинейный и относительно простой классификатор. Он классифицирует новый образец на основе измерения «расстояния» к числу моделей, которые хранятся в памяти. Класс, который метод ближайшего соседа определяет для этого нового образца, решается путем наложения шаблона, который больше всего напоминает его, то есть тот, который имеет наименьшее расстояние до него. Общая функция расстояния, используемая в данном классификаторе, – это евклидово расстояние. Вместо того, чтобы брать единственный ближайший образец, он обычно принимает решение большинством голосов от k ближайших соседей.

Алгоритм классификации по правилу ближайшего соседа достаточно прост. Представим выборку S в виде последовательности экспериментальных точек $x_1, x_2, \dots, x_n$ с известной принадлежностью классам. На каждом шаге при наблюдении экспериментальной точки делается предположение о ее классе и классификатору сообщается правильный ответ. Предположение относительно x_j производится после исследования предыдущих j–1 экспериментальных точек и нахождения k ближайших к x_j точек. При k=1

найденная точка будет ближайшим соседом точки x_j, так что x_j относят к тому классу, которому принадлежит этот ближайший сосед. Если k>1, то правило классификации сильно усложняется. Поэтому обычно применяют простые правила (например, голосование по большинству).

Метод ближайшего соседа был впервые описан Фиксом и Ходжесом в 1951 г [9]. Исследования, которые применяли классификацию по правилу ближайшего соседа, обычно получали хорошие результаты. Кавер и Харт [3, 4] доказали для случая k=1 теорему, которая дает объяснение этому явлению.

Пусть R_n – вероятность сделать ошибку при классификации по правилу ближайшего соседа в выборке S размера n. Не зная распределений вероятностей (включая соответствующие параметры), порождающих S, нельзя точно определить R_n, но эту величину можно оценить статистически. Положим

C_{n-1} = число наблюдений $\{x_i\}$ в последовательности x_1, x_2, …, x_n, которые были бы неверно классифицированы по правилу ближайшего соседа в выборке $S - x_j$.

Заметим, что это точная оценка только в предположении, что x_j – независимые случайные выборки с возвращением. C_n – случайная величина с математическим ожиданием $E(C_n)$, которое описывается формулой (5).

$$E(C_n) = R_n \tag{5}$$

Ясно, что R_n с ростом n стремится к R_∞ (формальное доказательство представлено в [4]). Пусть R^* будет риском ошибочной классификации любого случайным образом выбранного наблюдения при использовании некоторого неизвестного байесовского метода оптимальной классификации. Кавер и Харт [4] доказали, что для случая двух классов выполняется неравенство (6).

$$R^* \leq R_\infty \leq 2R^*(1 - R^*) \tag{6}$$

В таблице 1 приведены значения верхней границы для R_∞, соответствующей некоторым типичным уровням бейесовского риска

(нижней границы). Чтобы решить, приемлемы ли эти границы, рассмотрим «мысленный эксперимент». Пусть даны два черных ящика, представляющие собой устройство классификации. Они могут быть либо оба байесовскими, либо оба по правилу ближайшего соседа, либо различными. Наша задача заключается в выяснении, какая из этих возможностей на самом деле осуществляется. Чтобы решить ее, потребуем, чтобы каждое устройство классификации работало с различными выборками из N объектов. Так как выборки различны, можно было бы ожидать, что устройство классификации по правилу ближайшего соседа сделает больше ошибок. Поэтому, если шансов менее 1 из 100, что число наблюдаемых ошибок будет таким же, как в случае одинаковых устройств классификации, мы решаем признать одно устройство байесовским, другое – по правилу ближайшего соседа. Если же расхождение не слишком велико, то будем считать эти устройства классификации одинаковыми. Как велико должно быть число N, чтобы, а) у нас были хорошие шансы правильно считать устройства классификации различными каждый раз, когда они действительно различны, или б) правильно решать этот вопрос в той же ситуации в трех случаях из четырех? Ответ можно получить, используя для оценки возможностей неравенства (65) и затем применяя статистические методы оценки мощности критерия [51]. В столбцах 3а и 3б таблицы 1 приведены необходимые размеры выборки для каждого эксперимента с различными уровнями байесовского риска. Эти оценки могут оказаться лишь заниженными, поскольку при подготовке таблицы предполагалось, что устройство классификации по правилу ближайшего соседа функционирует с максимально возможным риском.

Таблица 1 – Размеры выборки для раличения

1. Байесовский риск (нижняя граница)	2. Верхняя граница	3. Размер выборки, необходимый для различения	
		3а. Мощность=0.5	3б. Мощность=0.75

0.050	0.095	470	930
0.100	0.180	220	430
0.150	0.225	170	340
0.200	0.320	160	310
0.250	0.375	160	320

В распознавании образов по методу ближайшего соседа проблема состоит в том, что найти ближайшего соседа за разумное время не просто. В распоряжении должна быть достаточная память для хранения всей выборки, а также процедура поиска, позволяющая быстро находить ближайшего соседа к вновь поступившему элементу.

Метод ближайшего соседа чувствителен к числу случаев в выборке, тогда как другие методы чувствительны к числу переменных. Это надо помнить при выборе метода решения конкретной задачи.

В заключение следует сказать, что метод ближайшего соседа дает удивительно хорошие результаты, если учитывать его простоту. Он особенно полезен в ситуациях, когда в значительной мере нарушается предположение о линейной отделимости. Основные слабости метода ближайшего соседа заключаются не в недостатке точности, а в требовании большой памяти и в том, что он (как и другие методы, использующие идею близости) очень сильно зависит от предположения об евклидовости пространства.

1.2.1.2.2 Метод опорных векторов

Метод опорных векторов (англ. support vector machine) использует гиперплоскость, чтобы классифицировать данные по классам. SVM позволяет спроецировать данные в пространство большей размерности, что дает возможность определить лучшую гиперплоскость, которая разделит данные на классы.

Дана обучающая выборка D – набор из n объектов, имеющих p параметров, формула (7):

$$D = \{(x_i, y_i) | x_i \in R^p, y_i \in \{-1,1\}\}_{i=1}^{n}, \quad (7)$$

где y принимает значения -1 или 1, определяя, какому классу принадлежит каждая точка.

Каждая точка x_i – это вектор размерности p.

Требуется найти гиперплоскость максимальной разности, которая разделяет наблюдения $y_i = 1$ от объектов $y_i = -1$.

Используя знания аналитической геометрии, любую гиперплоскость можно записать как множество точек x, удовлетворяющих условию (8).

$$w * x - b = 0, \quad (8)$$

где * – скалярное произведение нормали к гиперплоскости на вектор x.

Параметр $\frac{b}{\|w\|}$ определяет смещение гиперплоскости относительно начала координат вдоль нормали w.

Если обучающие данные являются линейно разделимыми, мы можем выбрать две гиперплоскости таким образом, что они отделят данные и точек между ними не будет. Затем, будем максимизировать расстояние между ними.

Область, ограниченная двумя гиперплоскостями, называется «разностью».

Эти гиперплоскости могут быть описаны уравнениями (9) и (10):

$$w * x - b = 1 \quad (9)$$

$$w * x - b = -1 \quad (10)$$

Используя геометрическую интерпретацию, находим расстояние между этими гиперплоскостями – $\frac{2}{\|w\|}$.

Для того чтобы дистанция была максимальной, минимизируем $\|w\|$.

Чтобы исключить все точки из полосы, мы должны убедиться, что для всех наблюдений справедливы неравенства (11) и (12):

$$w * x_i - b \geq 1, \text{для } x_i \text{ из первого класса} \quad (11)$$

$$w * x_i - b \leq -1, \text{для } x_i \text{ из второго класса} \quad (12)$$

Эквивалентно $y_i(w * x_i - b) \geq 1$ для $0 \leq i \leq n$.

Далее аналитическим способом решаем задачу оптимизации, заданную формулой (13):

$$\begin{gathered} \|w\| \to min \\ y_i(w * x_i - b) \geq 1 \text{ для } 0 \leq i \leq n. \end{gathered} \tag{13}$$

Задача оптимизации, представленная выше, трудно разрешима, так как она зависит от нормы *w*, которая включает в себя квадратный корень.

Однако задачу можно упростить, заменив на $\frac{1}{2}\|w\|^2$ (коэффициент от $\frac{1}{2}$ используются для математического удобства) без изменения решения, т.е. применив квадратичную оптимизацию.

Более точно, нужно найти минимум: $\frac{1}{2}\|w\|^2 \to min$.

При ограничениях, заданных неравенством (14).

$$y_i(w * x_i - b) \geq 1 \text{ для } 0 \leq i \leq n. \tag{14}$$

Путем введения множителей Лагранжа, задача с ограничениями может быть выражена как задача без ограничений, формула (15).

$$\min_{w,b} \max_{a \geq 0} \{\frac{1}{2}\|w\|^2 - \sum_{i=1}^{n} a_i * [y_i(w * x_i - b) - 1]\} \tag{15}$$

При этом все точки, которые могут быть отделены $y_i(w * x_i - b) \geq 1$, не имеют значения, поскольку мы должны найти точку равную нулю. Задача может быть решена с помощью квадратичного программирования.

«Стационарность» по Куна-Такеру означает, что решение может быть выражено как линейная комбинация обучающих векторов (формула (16)).

$$w = \sum_{i=1}^{n} a_i y_i x_i. \tag{16}$$

Только несколько множителей a_i будет больше 0.

Соответствующий x_i – опорный вектор, который лежит на краю и выражен как $y_i(w * x_i - b) = 1$.

Из этого следует, что опорные вектора также удовлетворяют условию (17).

$$w * x_i - b = \frac{1}{y_i} = y_i \leftrightarrow b = w * x_i - y_i \quad (17)$$

Выражение (17) позволяет определить смещение *b*.

На практике применяют усреднение N_{sv} по всем опорным векторам (формула (18)).

$$b = \frac{1}{N_{sv}} \sum_{i=1}^{N_{sv}} w * x_i - y_i. \quad (18)$$

Описание правила классификации в своей безусловной форме показывает, что максимальная маржа гиперплоскости и, следовательно, задача классификации является лишь функцией опорных векторов.

Наблюдения для обучения лежат на краю.

Используя факт $||w|| = w * w$ и подставляя $w = \sum_{i=1}^{n} a_i y_i x_i$, можно показать, что вторая форма метода опорных векторов позволяет решить проблему оптимизации:

Максимизировав по a_i, получим формулу (19).

$$L(a) = \sum_{i=1}^{n} a_i - \frac{1}{2} \sum_{i,j}^{n} a_i a_j y_i y_j x_i^T x_i = \sum_{i=1}^{n} a_i - \frac{1}{2} \sum_{i,j}^{n} a_i a_j y_i y_j k(x_i, x_j), \quad (19)$$

где $a_i \geq 0$.

Ограничение минимизации для *b* выражается формулой (20).

$$\sum_{i=1}^{n} a_i y_i = 0 \quad (20)$$

Ядро определено как $k(x_i x_j) = x_i x_j$.

W может быть вычислено благодаря условиям формулы (21).

$$w = \sum_{i=1}^{n} a_i y_i x_i. \quad (21)$$

Этот метод требует обучения. Чтобы показать SVM, что такое классы, используется набор данных – только после этого он оказывается способен классифицировать новые данные.

Слабыми сторонами этого метода являются необходимость выбора ядра и плохая интерпретируемость.

1.2.1.2.3 Деревья решений

Алгоритм C4.5 строит классификатор в форме дерева решений [35]. Чтобы сделать это, ему нужно передать набор уже классифицированных данных.

Классификация методом дерева решений создает некое подобие блок-схемы для распределения новых данных. В каждой точке блок-схемы задается вопрос о значимости того или иного атрибута, и в зависимости от этих атрибутов входные данные попадают в определенный класс.

Пусть нам дано множество примеров T, где каждый элемент этого множества описывается m атрибутами. Количество примеров в множестве T будем называть мощностью этого множества и будем обозначать $|T|$.

Пусть метка класса принимает следующие значения C_1, C_2, … C_k.

Задача заключается в построении иерархической классификационной модели в виде дерева из множества примеров T. Процесс построения дерева будет происходить сверху вниз. Сначала создается корень дерева, затем потомки корня и т.д.

На первом шаге мы имеем пустое дерево (имеется только корень) и исходное множество T (ассоциированное с корнем). Требуется разбить исходное множество на подмножества. Это можно сделать, выбрав один из атрибутов в качестве проверки. Тогда в результате разбиения получаются n (по числу значений атрибута) подмножеств и, соответственно, создаются n потомков корня, каждому из которых поставлено в соответствие свое подмножество, полученное при разбиении множества T. Затем эта процедура рекурсивно применяется ко всем подмножествам (потомкам корня) и т.д.

Рассмотрим подробнее критерий выбора атрибута, по которому должно пойти ветвление. Очевидно, что в нашем распоряжении m (по числу атрибутов) возможных вариантов, из которых мы должны выбрать самый

подходящий. Некоторые алгоритмы исключают повторное использование атрибута при построении дерева, но в нашем случае мы таких ограничений накладывать не будем. Любой из атрибутов можно использовать неограниченное количество раз при построении дерева.

Пусть мы имеем проверку X (в качестве проверки может быть выбран любой атрибут), которая принимает n значений $A_1, A_2, ..., A_n$. Тогда разбиение T по проверке X даст нам подмножества $T_1, T_2, ..., T_n$, при X равном соответственно $A_1, A_2, ..., A_n$. Единственная доступная нам информация – то, каким образом классы распределены в множестве T и его подмножествах, получаемых при разбиении по X. Именно этим мы и воспользуемся при определении критерия.

Пусть $freq\ (C_j, S)$ – количество примеров из некоторого множества S, относящихся к одному и тому же классу C_j. Тогда вероятность P того, что случайно выбранный пример из множества S будет принадлежать к классу C_j выражается формулой (22).

$$P = \frac{freq(C_j, S)}{|S|} \tag{22}$$

Согласно теории информации, количество содержащейся в сообщении информации, зависит от ее вероятности и определяется формулой (23).

$$\log_2(\frac{1}{p}) \tag{23}$$

Поскольку мы используем логарифм с двоичным основанием, то выражение (23) дает количественную оценку в битах.

$$Info(T) = -\sum_{j=1}^{k} \frac{freq(C_j, T)}{|T|} * \log_2 \frac{freq(C_j, T)}{|T|}$$

Выражение (24) дает оценку среднего количества информации, необходимого для определения класса примера из множества T. В терминологии теории информации выражение (24) называется энтропией множества T.

$$Info(T) = -\sum_{j=1}^{k} \frac{freq(C_j, T)}{|T|} * \log_2 \frac{freq(C_j, T)}{|T|} \quad (24)$$

Ту же оценку, но только уже после разбиения множества *T* по *X*, дает выражение (25).

$$Info_x(T) = \sum_{i=1}^{n} \frac{T_i}{T} * Info(T_i) \quad (25)$$

Тогда критерием для выбора атрибута будет являться формула (26).

$$Gain(X) = Info(T) - Info_x(T) \quad (26)$$

Критерий (26) считается для всех атрибутов. Выбирается атрибут, максимизирующий данное выражение. Этот атрибут будет являться проверкой в текущем узле дерева, а затем по этому атрибуту производится дальнейшее построение дерева. Т.е. в узле будет проверяться значение по этому атрибуту и дальнейшее движение по дереву будет производиться в зависимости от полученного ответа.

Такие же рассуждения можно применить к полученным подмножествам T_1, T_2, …, T_n и продолжить рекурсивно процесс построения дерева, до тех пор, пока в узле не окажутся примеры из одного класса.

Одно важное замечание: если в процессе работы алгоритма получен узел, ассоциированный с пустым множеством (т.е. ни один пример не попал в данный узел), то он помечается как лист, и в качестве решения листа выбирается наиболее часто встречающийся класс у непосредственного предка данного листа.

Здесь следует пояснить почему критерий (26) должен максимизироваться. Из свойств энтропии нам известно, что максимально возможное значение энтропии достигается в том случае, когда все его сообщения равновероятны. В нашем случае, энтропия (25) достигает своего максимума, когда частота появления классов в примерах множества *T* равновероятна. Нам же необходимо выбрать такой атрибут, чтобы при разбиении по нему один из классов имел наибольшую вероятность

появления. Это возможно в том случае, когда энтропия (25) будет иметь минимальное значение и, соответственно, критерий (26) достигнет своего максимума.

Как быть в случае с числовыми атрибутами? Понятно, что следует выбрать некий порог, с которым должны сравниваться все значения атрибута. Пусть числовой атрибут имеет конечное число значений. Обозначим их $\{v_1, v_2, \ldots, v_n\}$. Предварительно отсортируем все значения. Тогда любое значение, лежащее между v_i и v_{i+1}, делит все примеры на два множества: те, которые лежат слева от этого значения $\{v_1, v_2, \ldots, v_i\}$, и те, что справа $\{v_{1+1}, v_{i+2}, \ldots, v_n\}$. В качестве порога TH_i можно выбрать среднее между значениями v_i и v_{i+1} (формула (27)).

$$TH_i = \frac{v_i + v_{i+1}}{2} \tag{27}$$

Таким образом, мы существенно упростили задачу нахождения порога, и привели к рассмотрению всего $n-1$ потенциальных пороговых значений $TH_1, TH_2, \ldots, TH_{n-1}$.

Формулы (24), (25) и (26) последовательно применяются ко всем потенциальным пороговым значениям и среди них выбирается то, которое дает максимальное значение по критерию (26). Далее это значение сравнивается со значениями критерия (26), подсчитанными для остальных атрибутов. Если выяснится, что среди всех атрибутов данный числовой атрибут имеет максимальное значение по критерию (26), то в качестве проверки выбирается именно он.

Следует отметить, что все числовые тесты являются бинарными, т.е. делят узел дерева на две ветви.

Этот метод требует обучения, здесь тренировочный набор данных размечается классами. C4.5 не решает самостоятельно, к какому классу относятся входные данные. Как мы уже говорили, он создает дерево решений, которое используется для принятия решений.

Отличительные особенности данного метода:

1) Использование информационной энтропии при создании дерева решений.

2) Применение однопроходного прореживания для избегания переобучения.

3) Возможность работы с дискретными и непрерывными значениями путем ограничения диапазонов и установления порогов данных, обращая непрерывные данные в дискретные.

4) Обработка пропущенных данных.

Преимуществами использования деревьев решений являются их простая интерпретация и довольно высокая скорость работы.

1.2.1.2.4 Наивный байесовский классификатор

Наивный байесовский классификатор – это семейство алгоритмов классификации, которые принимают одно допущение: каждый параметр классифицируемых данных рассматривается независимо от других параметров класса.

Пусть D – множество признаков объектов, C – множество номеров (или наименований) классов. На множестве пар «объект, класс» $D \times C$ определена вероятностная мера P. Имеется конечная обучающая выборка независимых наблюдений $D^m = \{(d_1, c_1), \dots, (d_m, c_m),$ полученных согласно вероятностной мере P.

Задача классификации заключается в том, чтобы построить алгоритм $a: D \to C$, способный классифицировать произвольный объект $d \in D$.

В байесовской теории классификации эта задача разделяется на две.

1) Построение оптимального классификатора при известных плотностях классов. Эта подзадача имеет простое и окончательное решение.

2) Восстановление плотностей классов по обучающей выборке. В этой подзадаче сосредоточена основная сложность байесовского подхода к классификации.

Пусть для каждого класса $c \in C$ известна априорная вероятность P_c того, что появится объект класса c, и плотности распределения $p_c(d)$ каждого из классов, называемые также функциями правдоподобия классов. Требуется построить алгоритм классификации $a(d)$, доставляющий минимальное значение функционалу среднего риска.

В основе наивного байесовского классификатора лежит теорема Байеса (формула (28)).

$$P(c|d) = \frac{P(d|c)P(c)}{P(d)}, \tag{28}$$

где

1) $P(c|d)$ – вероятность что объект d принадлежит классу c;

2) $P(d|c)$ – вероятность встретить объект d среди всех объектов класса c;

3) $P(c)$ – безусловная вероятность встретить объект класса c в наборе объектов;

4) $P(d)$ – безусловная вероятность объекта d в наборе объектов.

Теорема Байеса позволяет переставить местами причину и следствие. Зная с какой вероятностью, причина приводит к некоему событию, эта теорема позволяет рассчитать вероятность того что именно эта причина привела к наблюдаемому событию.

Средний риск определяется как математическое ожидание ошибки $R(a)$ (формула (29)).

$$R(a) = \sum_{c \in C} \sum_{s \in C} \lambda_c P_c P_{(d,c)}\{a(d) = s|c\}, \tag{29}$$

где λ_c – цена ошибки или штраф за отнесение объекта класса y к какому-либо другому классу.

Теорема. Решением этой задачи является алгоритм, заданный выражением (30).

$$a(d) = arg \max_{c \in C} \lambda_c P_c p_c(d) \tag{30}$$

Значение $P(c|d) = P_c p_c(d)$ интерпретируется как апостериорная вероятность того, что объект d принадлежит классу c.

Если классы равнозначны, $\lambda_c P_c = const(c)$, то объект d просто относится к классу с наибольшим значением плотности распределения в точке d.

Наивный байесовский классификатор делает предположение, что объекты описываются независимыми признаками.

Исходя из этого предположения условная вероятность объекта аппроксимируется произведением условных вероятностей всех признаков, описывающих объект (формула (31)).

$$P(d|c) \approx P(w_1|c)P(w_2|c) \dots P(w_n|c) = \prod_{i=1}^{n} P(w_i|c) \qquad (31)$$

Предположение о независимости существенно упрощает задачу, так как оценить n одномерных плотностей гораздо легче, чем одну n-мерную плотность. К сожалению, оно крайне редко выполняется на практике, отсюда и название метода. Наивный байесовский классификатор может быть, как параметрическим, так и непараметрическим, в зависимости от того, каким методом восстанавливаются одномерные плотности.

Этот метод требует обучения, поскольку алгоритм использует размеченный набор данных для построения таблицы.

Алгоритм состоит из простой арифметики: умножение и деление. Когда частотные таблицы уже вычислены, классификация неизвестного входного параметра включает в себя только вычисления вероятностей для всех классов, а затем выбор наибольшей вероятности.

1.2.1.2.5 Искусственная нейронная сеть

Искусственная нейронная сеть (artificial neural network) – это математическая модель, построенная по принципу организации и функционирования биологических нейронных сетей.

Нейрон (нервная клетка) представляет собой биологическую клетку, которая обрабатывает информацию (рисунок 1). Он состоит из тела клетки, или сомы, и двух типов внешних древоподобных ветвей: аксона и дендритов. Тело клетки включает ядро, которое содержит наследственную информацию, и плазму, которая производит необходимые для нейрона материалы. Нейрон получает сигналы (импульсы) от других нейронов через дендриты (приемники) и передает сигналы, сгенерированные телом клетки, вдоль аксона (передатчика), который в конце разделяется на волокна. На окончаниях этих волокон находятся синапсы – места для передачи нервного импульса между нейронами.

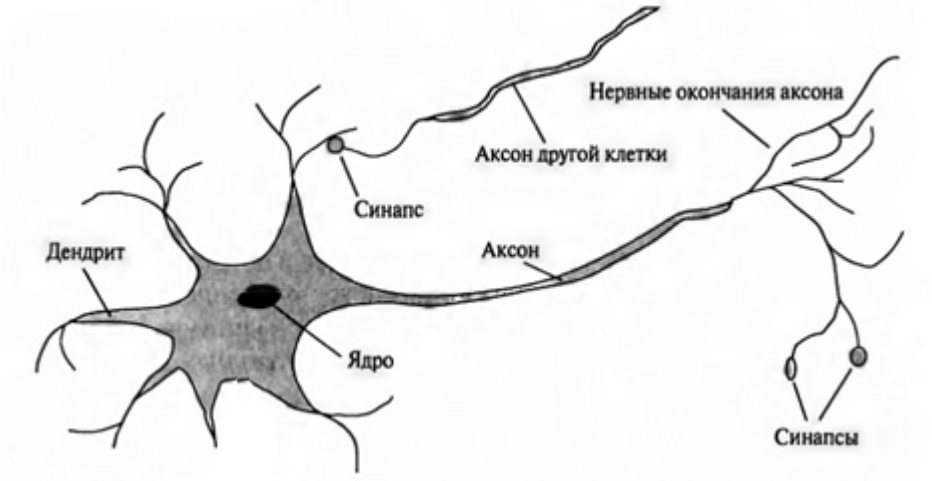

Рисунок 1 – Биологический нейрон

Искусственная нейронная сеть представляют собой систему соединённых и взаимодействующих между собой простых искусственных нейронов. Такие нейроны обычно довольно просты. Каждый нейрон подобной сети имеет дело только с сигналами, которые он периодически получает, и сигналами, которые он периодически посылает другим процессорам. Однако при соединение подобных нейронов в достаточно большую сеть, они способны решать сложные задачи, в частности задачу классификации изображений.

Искусственный нейрон, наподобие биологическому, состоит из синапсов, связывающих входы нейрона с ядром; ядра нейрона, которое осуществляет обработку входных сигналов и аксона, который связывает

нейрон с нейронами следующего слоя. Рисунок 2 демонстрирует схему искусственного нейрона.

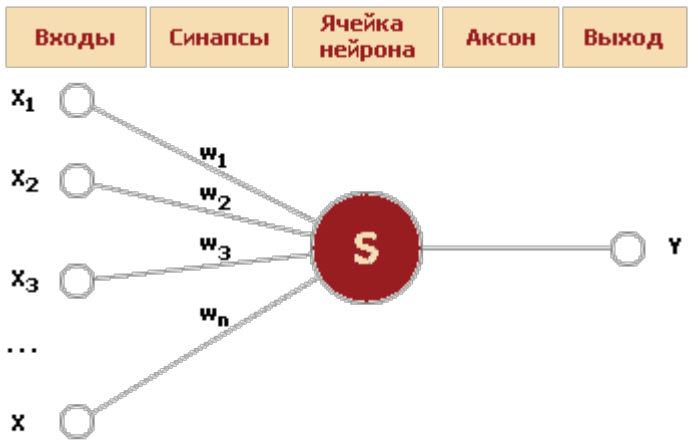

Рисунок 2 – Искусственный нейрон

Каждый синапс имеет вес, который определяет, насколько соответствующий вход нейрона влияет на его состояние. Состояние нейрона определяется по формуле (32).

$$S = \sum_{i=1}^{n} x_i w_i, \tag{32}$$

где n – число входов нейрона, x_i – значение i-го входа нейрона, w_i – вес i-го синапса.

Затем определяется значение аксона нейрона по формуле (33).

$$Y = f(s), \tag{33}$$

где f – некоторая активационная функция. Наиболее часто в качестве активационной функции используется так называемый сигмоид, который имеет вид, представленный формулой (34).

$$f(x) = \frac{1}{1 + e^{-ax}} \tag{34}$$

1.2.2 Общее описание построения признаков

Конструирование признаков может применяться для достижения двух различных целей: уменьшения размерности данных и улучшения показателей прогнозирования. В данной работе будет рассматриваться использование построения признаков для увеличения эффективности классификации медицинских изображений.

Концептуально любой метод построения признака можно рассматривать как выполнение следующих действий:

1) Выбор начального пространства признаков F_0 (ручное построение признаков).

2) Преобразование F_0 для построения нового пространства признаков F_N (преобразование признаков).

3) Выбор подмножества признаков F_i из F_N (выбор признака).

   a. Определение полезности F_i для задачи классификации.

   b. Если некоторые критерии завершения не достигнуты:

      i. Вернитесь к шагу 3.

   c. Иначе множество $F_T = F_i$.

4) F_T – это сконструированное пространство признаков.

Исходное пространство признаков F_0 состоит из признаков, созданных вручную, которые часто кодируют некоторые базовые знания области. Различные методы построения признаков отличаются тем, как они реализуют каждый из этих этапов. Ясно, что тремя важными аспектами любого метода построения признаков являются: метод трансформации, метод выбора подмножества признаков F_i и критерий полезности для подмножества признаков.

1.2.2.1 Преобразование признаков

Общим подходом к созданию преобразованных признаков является применение набора операторов (например, {+, –, *}) к исходным значениям признаков. Выбор операторов основан на знании области и типе признаков. К числу наиболее часто используемых операторов относятся:

1) Логические функции: конъюнкция, дизъюнкция, отрицание и т. д.

2) Номинальные характеристики: декартово произведение и т. д.

3) Численные характеристики: минимум, максимум, сложение, вычитание, умножение, деление, среднее, неравенство и т. д.

Помимо этого, гиперплоскости, логические правила и битовые строки также могут использоваться для создания новых пространств признаков. Операторы обычно применяются итеративно. Поэтому каждый новый признак $\varphi_i \in F_N$ можно представить, используя дерево операторов и исходные значения признаков, как показано на рисунке 3.

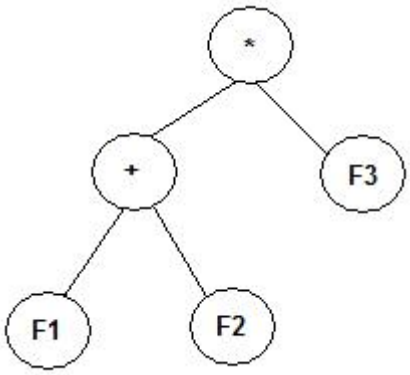

Рисунок 3 – Древовидное представление сконструированного признака $\varphi_i \in F_N$

1.2.2.2 Выбор признаков

Выбор признаков является важным шагом в процессе построения признака. Так как преобразованное пространство признаков F_N велико, нам нужно выбрать подмножество F_T из F_N. Проблема выбора оптимального подмножества является NP трудной, и методы обычно выполняют какой-то неоптимальный жадный поиск. Используемые критерии для измерения полезности пространства признаков F_i часто включают в себя получение информации, коэффициент корреляции, точность предсказания на некотором множестве проверки и т. д.

В литературе было представлено множество различных методов отбора [10, 13]. Мы можем свободно классифицировать эти методы по двум категориям: фильтры и обертки [17].

Фильтрующие методы выбирают подмножества признаков независимо от предиктора. Они, по существу, действуют как этап предварительной обработки данных, прежде чем обучить предиктор. В эту категорию входят

подходы с переменным ранжированием, которые включают ранжирование отдельных признаков с использованием теоретико-информационных или корреляционных критериев, а затем построение подмножества с высокими показателями выигрыша. Фильтры имеют преимущество в том, что они быстрее, чем обертки. Более того, они, как правило, обеспечивают общий порядок построения признаков, не настроенный на конкретный метод обучения. Однако недостатком является то, что выбранное подмножество может быть не самым подходящим для конкретного классификатора.

Оберточные методы выбора признаков используют метод обучения, который будет использоваться для прогнозирования как черный ящик для выбора подмножеств признаков. Эти методы обычно делят обучающий набор на набор обучения и проверки (набор тестов является отдельным). Для любого заданного подмножества признаков предиктор обучается набору обучения и тестируется на наборе проверки. Точность прогнозирования на наборе проверки рассматривается как оценка подмножества признаков. Таким образом, мы в конечном счете хотели бы выбрать подмножество с наивысшей оценкой. Из-за повторяющихся циклов обучения и тестирования для каждого подмножества признаков, обертки имеют тенденцию быть намного более вычислительно затратными по сравнению с фильтрами. Обычно цель состоит в том, чтобы пропустить пространство признаков таким образом, чтобы число проверяемых подмножеств было сведено к минимуму. Очевидным преимуществом является то, что выбранное подмножество настроено на предиктор.

1.2.3 Обзор аналогов

Задача построения соответствующих признаков часто очень специфична для области применения и трудоемка. Таким образом, создание автоматизированных методов построения признаков, требующих минимальных усилий пользователя, является сложной задачей. В частности, необходимы методы, которые:

1) Создают набор признаков, которые помогут улучшить точность прогнозирования.

2) Вычислительно эффективны.

3) Являются обобщаемыми для разных классификаторов.

4) Позволяют легко добавлять знания области.

Был предложен ряд различных методов. В следующих подразделах эти методы будут классифицированы на основе методик, которые они используют для определения и поиска пространства признаков. Ранние методы в основном базировались на деревьях решений, в то время как последние подходы были в большей степени сосредоточены на индуктивном логическом программировании и аннотациях.

1.2.3.1 Деревья решений

Один из ранних алгоритмов построения признаков принадлежит Пагалло [33], создателю FRINGE, который адаптивно увеличивал первоначальный набор атрибутов для изучения концепции дизъюнктивной нормальной формы (ДНФ). На каждой итерации новые признаки строятся путем комбинирования пар признаков в существующем пространстве признаков с использованием операторов отрицания и конъюнкции. Поскольку данные операторы являются полным набором булевых операторов, общее пространство признаков состоит из всех булевых функций исходных признаков. Чтобы справиться с проблемой чрезвычайно большого пространства новых признаков, FRINGE объединяет только пары признаков, которые появляются на краю каждой из положительных ветвей дерева решений. Процесс создания признака повторяется до тех пор, пока не будут созданы новые признаки.

CITRE [25] и DC Fringe [48] – два других алгоритма построения признака на основе деревьев решений. Алгоритмы используют конъюнкции и дизъюнкции для объединения различных операндов, таких как корень (выбирает первые два признака каждой положительной ветви), границу

(подобно FRINGE), корневую границу (комбинация корня и границы), смежную (выбирает все соседние пары вдоль каждой ветви) и все (всё вышеперечисленное).

Проблема с алгоритмами на основе деревьев решений заключается в том, что с тех пор, как новые признаки добавляются в пространство признаков на каждой итерации, количество входных признаков, которые необходимо передать в алгоритм построения дерева решений, становится очень большим, делая процесс вычислительно неэффективным. В результате в каждой итерации некоторые низко оцениваемые признаки отбрасываются. Помимо использования обычных методов обрезки дерева решений, все признаки, которые не использовались при построении дерева решений, также могут быть исключены.

Все методы, рассмотренные ранее, использовали только логические операторы для генерации признаков. Чтобы разработать более гибкий подход, Маркович и Розенштейн [24] представили FICUS, общую структуру построения признаков. Их подход обеспечивает набор функций (операторов) конструктора вместе с исходным набором признаков и примерами для алгоритма построения признаков. Затем алгоритм обогащает исходное пространство признаков добавлением дополнительных перспективных признаков. Функции конструктора могут быть либо одним, либо несколькими обычно используемыми операторами, либо могут поставляться некоторым экспертом предметной области.

Вход в FICUS задается с использованием «языка спецификаций признаков» (ЯСП). Пользователь может предоставить информацию о типе (например, номинальном, непрерывном и т.д.), области и диапазоне основных признаков и функций конструктора. Также пользователь может дополнительно указать набор булевых ограничений на тип признаков, которые могут быть сгенерированы или использованы с определенными функциями. Таким образом, новое пространство признаков представляет собой набор всех признаков, которые могут быть сгенерированы на основе

ЯСП. Обход пространства поиска осуществляется с помощью четырех разных операторов:

1) Композиция. Один или два признака используются в качестве входных данных. На их основе вычисляется новый набор признаков с использованием всех допустимых функций.

2) Вставка. На вход подаются два признака. Новый признак получается путем вставки одного признака в другой.

3) Замена. Из двух начальных признаков формируется новый признак с помощью замены какого-либо компонента одного признака на компонент другого признака.

4) Интервал. Из одного исходного признака получаются новые логические признаки, проверяющие, лежит ли начальный признак в каком-либо определенном диапазоне.

Стратегия поиска является вариантом лучевого поиска. На каждом шаге поддерживаются два набора признаков: текущий набор признаков и предыдущий набор признаков, который создал текущий набор через применение четырех указанных выше операторов. Сохранение предыдущего набора позволяет системе выполнить один уровень обратного отслеживания. Полезность набора признаков вычисляется на основе размера и сложности дерева решений, которое генерируется по множеству примеров. Полезность отдельных признаков в наборе вычисляется с использованием критериев разделения (получения информации). На каждой итерации используется лучший набор признаков для создания нового набора признаков. Авторы показали, что их метод достиг значительного прироста производительности в разных областях и классификаторах.

Обладая высокой гибкостью, с одной стороны, у FICUS было два потенциальных недостатка, с другой стороны. Во-первых, критерии выбора подмножества признаков не учитывали взаимодействия признаков, что приводило к несколько узкому поиску в пространстве признаков. Во-вторых,

как обсуждалось ранее, при заданной проблеме выбор операторов часто неясен, что затрудняет выбор правильного набора функций конструктора.

1.2.3.2 Индуктивное логическое программирование

Индуктивное логическое программирование используется для разработки описаний предикатов из примеров и базовых знаний. Методы построения признаков на основе индуктивного логического программирования могут обеспечить обобщенную структуру для включения знаний области в процесс создания признаков. Первое использование предикатов первого порядка в качестве признаков было применено в программе LINUS [21]. В последующей литературе проблема идентификации хороших признаков с представлением первого порядка была рассмотрена в пропозициональных (основанных на характеристиках) подходах [19].

В работах [41, 42] использовался подход, основанный на индуктивном логическом программировании, для повышения точности прогнозирования в задаче определения значения слова. Основная идея заключается в применении двухэтапного подхода:

1) Конструирование признака. Индуктивное логическое программирование используется для изучения нового набора конъюнктивных признаков.

2) Выбор признака. Выбор подмножества признаков на основе их полезности в процессе прогнозирования.

Система изучает предложения, заданные формулой (35).

$$h_i: Class\ (x, c) < -\varphi_i\ (x), \tag{35}$$

где $\varphi_i(x)$: $X -> 0, 1$.

φ_i – это конъюнкция над некоторыми значениями признака экземпляра x и вычисляется как TRUE (1), если является конъюнкцией, иначе – FALSE (0). Предложения h_i можно рассматривать как правила вида: «Если какой-либо признак $\varphi(x) = 1$, то экземпляр x должен принадлежать классу c.

Когда такие правила выводятся из набора примеров, пространство признаков обогащается добавлением всех индивидуальных конъюнкций φ_i в качестве признаков. Таким образом, они используют индуктивное логическое программирование для идентификации подмножества конъюнкций из пространства всех возможных конъюнкций исходных признаков. Дополнительным преимуществом использования индуктивного логического программирования является то, что конъюнктивные предложения h_i также могут быть получены из источников, отличных от тренировочных примеров. Это позволяет легко встраивать знания области в процесс построения признака. После создания исходного набора признаков, они затем используют метод выбора признаков в соответствии с подходом обертки, о котором говорилось выше. На каждой итерации оценивается производительность прогнозирования набора признаков и генерируются два новых набора признаков. Один, исключая худший признак, а другой, добавляя наиболее эффективный признак в другой случайный образец признаков из исходного набора признаков. Полученная модель показала впечатляющий прирост производительности в задаче разрешения лексической многозначности.

1.2.3.3 Аннотации

Подходы, основанные на аннотациях, позволяют пользователям предоставлять знания области в виде аннотаций вместе с примерами обучения. Затем на основе этих аннотаций будет изучено пространство признаков. Таким образом, исключается необходимость в определении операторов.

Ротт и Смолл [37] предложили интерактивный протокол построения пространства признаков. Вместо того, чтобы предопределять большое пространство признаков с помощью операторов, их подход позволяет создавать динамическое пространство признаков, основанное на взаимодействии между обучающей машиной и экспертом предметной

области. Во время учебного процесса, когда учащийся представляет какой-то пример эксперту по предметной области, эксперт использует знания области, чтобы предоставить дополнительные аннотации (а не только ярлыки). Учащийся теперь должен включить дополнительные знания через модификации пространства признаков и изучить модель. Таким образом, эксперт области может напрямую кодировать знания об области без необходимости определения операторов. Они применили свой метод к проблеме распознавания именованного объекта при использовании семантически связанных списков слов [8, 34] в качестве внешнего знания. Семантически связанные списки слов группируют вместе лексические элементы, принадлежащие к одной семантической категории. Например, Направление_Компаса = {восток, запад, север, юг}. Протокол работает следующим образом:

1) Учащийся начинает с изучения первоначальной гипотезы h в исходном пространстве признаков F_0 и оценки всех экземпляров, использующих его.

2) Затем функция запроса Q используется для выбора лучших экземпляров и представления их эксперту области. Цель состоит в том, чтобы выбрать Q таким образом, чтобы он минимизировал взаимодействие с пользователем при максимальном воздействии.

3) Эксперт рассматривает экземпляры и отмечает некоторые признаки для абстракции. Например, на рисунке 4 ученик ошибается при маркировке «Chicagoland» как организации.

4) Эксперт отмечает признаки, относящиеся к «западу» для абстракции.

5) Основываясь на этом взаимодействии, пространство признаков изменяется путем замены признаков φ_{east}, φ_{west}, φ_{north} в исходном пространстве признаков F_0 на $\varphi_{Compass}$-$Direction$= $\varphi_{east} \vee \varphi_{west} \vee \varphi_{north}$

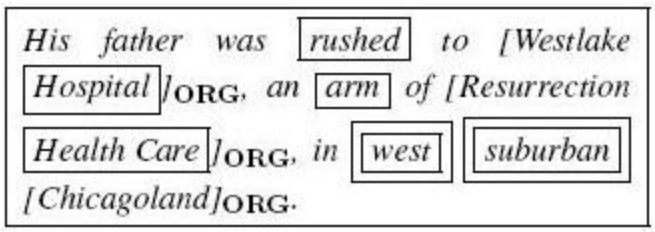

Рисунок 4 – Пример работы протокола CCCC

Рисунок 4 демонстрирует работу протокола. Все лексические элементы с членством в семантически связанном списке слов обведены квадратом. Элементы, используемые для неверного предсказания для «Chicagoland», имеют двойное обведение. Эксперт может выбрать любой обведенный элемент для проверки семантически связанного списка слов.

Другой интерактивный метод принадлежит Рагхавану и Аллену [36], которые представляют тандемный алгоритм обучения для выбора признаков для классификации текста. Алгоритм начинается с небольшого количества помеченных экземпляров и на каждой итерации рекомендует экземпляры и признаки для людей для маркировки. Однако в отличие от семантически связанных списков слов, в их случае пространство признаков остается статичным.

Другие методы, которые позволяют эксперту области напрямую указывать информацию об пространстве признаков с помощью аннотаций, включают [5, 16, 22, 49; 50]. В частности, Лим и др. [22] предлагают метод построения признаков, основанный на объяснительном обучении, для задачи распознавания китайских символов. В их случае знания кодировались в виде «штрихов», которые использовались для генерации символов (или экземпляров). Затем признаки генерировались на основе наличия или отсутствия одинаковых штрихов в разных классах.

Для классификации медицинских изображений необходим метод, обладающий большой гибкостью в операторах, поскольку невозможно заранее предположить какое построение признаков окажется наиболее точным. Рассмотренные методы позволяют легко внедрять знания

предметной области, но не дают преимущества в ситуациях, когда сложно предсказать какие именно сочетания признаков дадут наилучший результат.

Для создания такого нового пространства признаков, строение которых заранее неизвестно, можно использовать генетическое программирование.

1.3 Обзор алгоритма

1.3.1 Генетическое программирование

Многие, казалось бы, разные проблемы в искусственном интеллекте, символьной обработке и машинном обучении можно рассматривать как требующие компьютерной программы, вычисляющей некоторый требуемый результат в зависимости от входных параметров.

Процесс решения этих проблем можно сформулировать как поиск наиболее подходящей индивидуальной компьютерной программы среди всех возможных компьютерных программ. Пространство поиска состоит из всех возможных компьютерных программ, составленных из функций и терминальных символов, соответствующих проблемной области. Генетическое программирование предоставляет способ поиска этих наиболее подходящих индивидуальных компьютерных программ.

В генетическом программировании популяции сотен или тысяч компьютерных программ генетически выводятся. Эта селекция осуществляется с помощью принципа выживания сильнейших и воспроизводства наиболее приспособленных особей вместе с генетической рекомбинацией (скрещиванием) организмов путем применения операций, подходящих для компьютерных программ. Компьютерная программа, которая решает (или приблизительно решает) определенную проблему может возникнуть из комбинации естественного отбора Дарвина и генетических операций.

Генетическое программирование начинается с генерации случайным образом выбранной начальной популяции компьютерных программ из

функций и терминалов, соответствующих проблемной области. Функции могут быть стандартными арифметическими операциями, операциями программирования, математическими, логическими или предметно-ориентированными фикциями. В зависимости от конкретной задачи, компьютерная программа может работать с логическими значениями, целыми, вещественными или комплексными числами, векторами, символами. Создание этой начальной популяции в действительности «слепой» случайный поиск в пространстве проблемной области.

Каждая программа в популяции оценивается, насколько хорошо она выполняет свои задачи в проблемной среде. Эта оценка носит название меры пригодности. Ее вид зависит от проблемы.

Для многих задач пригодность естественно измерять ошибкой, погрешностью компьютерной программы. Чем ближе эта ошибка к нулю, тем лучше данная программа. Также приспособленность может быть комбинацией таких факторов, как корректность, экономность и бережливость.

Как правило, каждая компьютерная программа популяции выполнятся для нескольких значений входных параметров. Тогда пригодность будет считаться в виде суммы или среднего арифметического значений приспособленности всех входных параметров. Например, пригодность компьютерной программы может быть суммой абсолютной величины от разности вычисленного программой значения и корректного решения проблемы. Эта сумма может быть получена из выборки 50 различных входных значений программы.

За исключением случаев, когда проблема мала и проста, она не может быть легко решена путем «слепого» случайного поиска, компьютерные программы нулевого поколения будут иметь очень плохую пригодность. Тем не менее, некоторые особи в популяции будут немного пригоднее остальных. Эти различия в эффективности следует использовать в дальнейшем.

Принципы репродукции и выживания наиболее приспособленных особей и генетические операции половой рекомбинации (скрещивание) используются для создания нового поколения индивидуальных компьютерных программ из текущей популяции.

Операция репродукции включает в себя селекцию, пропорциональную значениям приспособленности, компьютерных программ из текущей популяции и позволяет отобранным особям выжить путем копирования в новую популяцию.

Генетический процесс полового скрещивания двух родителей – компьютерных программ – используется для создания новых потомков от родителей, выбранных пропорционально значениям пригодности. Программы-родители обычно имеют различный размер и форму. Программы-потомки составляются из подвыражений (поддеревьев, подпрограмм) родителей. Эти потомки, как правило, различаются размером и видом от своих родителей.

Интуитивно, если две компьютерные программы несколько эффективны в решении проблемы, то их части, возможно, тоже немного пригодны. Скрещивая случайно выбранные части относительно пригодных программ, мы можем получить новую компьютерную программу, которая даже лучше решает проблему.

После выполнения операций репродукции и скрещивания с текущей популяцией, популяция потомков (новое поколение) помещается в старую популяцию (прошлое поколение).

Каждая особь новой популяции компьютерных программ затем проверяется на пригодность, и процесс повторяется в течение многих поколений.

На каждом этапе этого параллельного, локально управляемого, децентрализованного процесса состоянием будет являться только текущая популяция особей. Движущая сила этого процесса состоит только в наблюдении за пригодностью особей текущей популяции.

Данный алгоритм будет производить популяцию компьютерных программ, которые через много поколений, как правило, демонстрируют увеличение средней пригодности. Кроме того, эти популяции могут быстро и эффективно приспосабливаться к изменениям в окружающей среде.

Как правило, лучшая особь, которая появляется в любом по счету поколении, обозначается как результат генетического программирования.

Важной особенностью генетического программирования является иерархический характер производимых программ. Результаты генетического программирования по своей природе иерархичны.

Динамическая изменчивость также является важным признаком компьютерных программ, созданных генетическим программированием. Было бы трудно и неестественно пытаться заранее уточнить или ограничить размер и форму возможного решения. Более того, такая предварительная спецификация сужает пространство поиска решений и может исключить нахождение решения вообще.

Еще одной важной особенностью генетического программирования выступает отсутствие или сравнительно малая роль предварительной обработки входных данных и постобработки выходных значений. Входные параметры, промежуточные результаты и выходные значения обычно выражаются непосредственно в терминах естественной терминологии предметной области. Элементы функционального множества также естественны для проблемной области.

И наконец, структуры, подвергающиеся адаптации, активны. Они не являются пассивными кодировками решения проблемы. Вместо этого с учетом компьютера, на котором происходит запуск, программы в генетическом программировании – это активные структуры, способные выполниться в их текущем виде.

Парадигма генетического программирования является независимой от проблемной области. Это обеспечивает единый, унифицированный подход к проблеме нахождения решения в виде компьютерной программы.

1.3.1.1 Начальные структуры

Начальные структуры в генетическом программировании состоят из особей исходной популяции, каждая из которых представляет решение проблемы.

Создание каждого выражения начальной популяции выполняется в виде дерева со случайно выбранным корнем и упорядоченными ветвями.

Начинаем со случайного выбора одной функции из функционального множества *F*, которая станет корнем дерева. Мы ограничиваем выбор функциональным множеством, поскольку нам необходимо создать иерархическую структуру, а не вырожденную структуру, состоящую из одного терминального символа.

Рисунок 5 демонстрирует начало создания случайного дерева программы. Функция сложения (с двумя аргументами) была выбрана случайно из функционального множества *F* в качества корня дерева.

Рисунок 5 – Создание корня дерева

Когда узел дерева помечается функцией *f* из *F*, то *z(f)* линий, где *z(f)* – количество аргументов функции *f*, выходит из этого узла. Затем для каждой такой линии случайно выбирается элемент из объединенного множества $C = F \cup T$, функций и терминалов, для конечной точки этой линий.

Если в качестве узла была выбрана функция, то дальнейшее создание дерева продолжается рекурсивно так, как было описано выше. Например, на рисунке 6, функция умножения была выбрана из множества $C = F \cup T$ в качестве внутреннего узла (номер два) для конечной точки левой линии корневого узла (функция сложения, номер один). Функция умножения принимает два аргумента, поэтому из второй вершины выходят две линии.

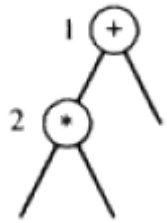

Рисунок 6 – Выбор внутреннего узла

Если в качестве любого узла выбирается терминальный символ, то этот узел становится листом, и процесс создания поддерева для этой вершины прекращается. Например, на рисунке 7 терминальный символ *A* был выбран в качестве вершины для левой линии функции умножения. Аналогичным образом, терминалы *B* и *C* стали вершинами двух правых линий функций умножения и сложения соответственно. Этот процесс продолжается рекурсивно слева направо, пока все дерево не создано, как показано на рисунке 3.

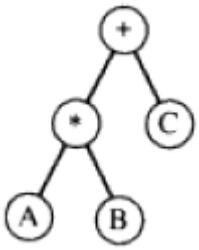

Рисунок 7 – Созданное дерево программы

Данный процесс генерации может быть реализован различными способами, приводящими к получению случайных начальных деревьев разного размера и вида. Два основных метода называются «полным» методом и «растущим» методом. Длина дерева определяется, как длина самого длинного пути от корня к листу.

Полный метод создания начальной популяции состоит в генерации дерева, у которого длина каждого пути от листа к корню равна указанной максимальной глубине. Такой вид дерева можно получить путем ограничения выбора функциональным множеством для вершин, у которых

глубина меньше заданной. А затем ограничить выбор только терминальным множеством для вершин с максимальной глубиной.

Растущий метод генерации начальной популяции состоит в создании деревьев различной формы. Длина пути от листа до корня не больше заданной максимальной глубины. Это достигается путем случайного выбора внутреннего узла из множества $C = F \cup T$, при этом длина пути от корня к узлу меньше максимальной глубины. А вершины дерева, чья длина равна максимальной, становятся листами и выбираются из терминального множества.

Лучший результат для широкого диапазона проблемы дает объединенный метод, при котором начальная популяция создается чередованием полного и растущего методов. В генетическом программировании мы, как правило, не знаем заранее (или не хотим указывать) размер и форму решения. Объединенный метод генерации деревьев создает широкое разнообразие деревьев различного размера и вида.

Объединенный метод генерации – это смешанный метод, включающий и растущий, и полный методы. Данный способ состоит из создания равного количества деревьев с глубиной, которая находится в интервале от 2 до максимальной заданной глубины. Например, если максимальная глубина равна 6, то 20% деревьев будут иметь глубину 2, 20% деревьев глубину 3, и так далее до глубины 6. Затем для каждого значения глубины 50% деревьев создаются полным методом, а остальные 50% растущим методом.

Заметим, что у всех деревьев, созданных полным методом с заданной глубиной, длина пути от корня к листу одинаковая, равная максимальной глубине, и поэтому эти деревья имеют одинаковую форму. В отличие от этого, у всех деревьев, полученных растущим методом с данным значением глубины, ни один путь от корня дерева до листа не превышает максимальной глубины. Поэтому эти деревья значительно отличаются друг от друга по виду даже при одинаковой максимальной глубине.

Таким образом, объединенный метод создает деревья с большим разнообразием размеров и видов.

Повторяющиеся особи в начальной популяции непродуктивны: они тратят вычислительные ресурсы и приводят к нежелательному сокращению генетического разнообразия популяции. Поэтому желательно, но необязательно, предотвратить появление дубликатов в начальной случайным образом созданной популяции. В генетическом программировании вероятность появления повторяющихся особей в начальной популяции особенно высока, когда деревья малы. Таким образом, каждое созданное выражение проверяется на уникальность перед добавлением в популяцию. Если новое выражение является дубликатом, то процесс повторяется до создания уникальной особи. Иногда (например, для маленьких деревьев) мы должны подставить большее по размеру дерево во время процесса создания, если исчерпано множество возможных деревьев данного размера.

Разнообразие популяции – это доля особей, у которых нет точной копии во всей популяции. Если выполняется проверка на дублирование при создании особей, то разнообразие начальной популяции должно быть равно 100%. В последующих поколениях появление одинаковых особей при использовании репродукции является неотъемлемой частью генетического процесса.

1.3.1.2 Приспособленность

Приспособленность является движущей силой естественного отбора по Дарвину и, следовательно, генетического программирования.

В природе приспособленность – это вероятность того, что особь доживет до репродукционного возраста и воспроизведется. Данный показатель может учитываться при расчете числа потомков. В искусственном мире математических алгоритмов мы оцениваем пригодность каким-либо способом, а затем используем значение приспособленности для контроля

применения операций, изменяющих структуры в нашей искусственной популяции.

Приспособленность может быть вычислена при помощи различных методов, явных и неявных.

Наиболее распространенным подходом вычисления приспособленности является создание определенной меры пригодности для каждой особи популяции. Данный подход используется подавляющим большинством генетических алгоритмов. Каждой особи популяции присваивается скалярное значение приспособленности при помощи некоторой четко и явно определенной процедуры оценки.

Приспособленность также может быть вычислена путем совместной эволюции (коэволюции), при которой пригодность игровой стратегии определяется применением этой стратегии против всей популяции (или отобранного числа особей), ведущей противоположную стратегию.

Тот факт, что особи существуют и выживают в популяции, а также успешно воспроизводятся, может свидетельствовать об их приспособленности. Такое неявное определение приспособленности часто используется в научных исследованиях [15]. Однако, на данный момент, мы сосредоточимся на более общей ситуации, когда приспособленность вычисляется явно.

Существует 4 меры приспособленности:

– исходная приспособленность;

– стандартизованная приспособленность;

– отрегулированная приспособленность;

– нормированная приспособленность.

1.3.1.2.1 Исходная приспособленность

Исходная приспособленность – это измерение пригодности, сформулированное в естественной терминологии проблемы. Например,

исходной приспособленностью для классификации является погрешность полученного результата классификации объекта с используемым построенным набором признаков относительно желаемого. Чем меньше погрешность, тем лучше.

Наиболее общим определением исходной приспособленности является ошибка. То есть исходная приспособленность отдельного выражения – это сумма расстояний для всех значений независимых переменных между полученным результатом для конкретного значения свободной переменной и желаемым результатом для этой же переменной. Выражение может быть логическим, целочисленным, вещественным, комплексным, вектором или символьным значением.

Если выражение целочисленное или вещественное, то сумма дистанций вычисляется в виде суммы абсолютных значений разности полученного и необходимого результатов. Когда исходной приспособленностью является ошибка, исходная приспособленность *r(i, t)* отдельного выражения *i* в популяции размера *M* любого поколения шага *t* вычисляется по формуле (36).

$$r(i,t) = \sum_{j=1}^{N} |S(i,j) - C(i,j)|, \qquad (36)$$

где *S(i, j)* – значение выражения *i* для номера *j* значения переменной; *N* – количество значений переменных; *C(j)* – необходимое значение для *j*-ого значения свободной переменной.

Если выражение логическое или символьное, то сумма расстояний будет эквивалентна числу несоответствий. Если выражение комплексное или векторное, то сумма расстояний получается отдельно для каждого компонента структуры.

Если выражение вещественное или целочисленное, то квадратный корень суммы квадратов расстояний может быть использован в качестве альтернативы для измерения приспособленности.

Исходя из того, что исходная приспособленность основывается на естественной терминологии проблемы, чем лучше значение, тем оно меньше (когда в качестве меры приспособленности берется ошибка) или больше (когда используется достигнутая выгода).

1.3.1.2.2 Стандартизованная приспособленность

Стандартизованная приспособленность *s(i, t)* пересчитывает исходную приспособленность так, чтобы наименьшее численное значение всегда было лучшим результатом. Например, для проблемы оптимального управления, заключающейся в минимизации расходов, меньшее значение исходной приспособленности лучше. Также, если проблема заключается в минимизации ошибки, то меньшее значение исходной приспособленности лучше.

Для таких проблем, когда меньшее значение исходной приспособленности является лучшим, стандартизованная приспособленность равняется исходной приспособленности (формула (37)).

$$s(i, t) = r(i, t) \tag{37}$$

Удобно и желательно сделать лучшим значением стандартизованной приспособленности 0.

Если для определенных проблем большее значение исходной приспособленности лучше, то стандартизованная приспособленность вычисляется из исходной приспособленности. Стандартизованная приспособленность равняется разности максимально возможного значения исходной приспособленности и данной приспособленности (формула (38)).

$$s(i,\ t) = r_{max} - r(i,\ t) \quad (38)$$

Если верхняя граница неизвестна, а большее значение исходной приспособленности является лучшим, то отрегулированная и нормализованная приспособленности могут быть вычислены непосредственно из исходной приспособленности.

1.3.1.2.3 Отрегулированная приспособленность

Кроме того, для решения проблемы используется оптимальная регулировка приспособленности. Отрегулированная приспособленность *a(i,t)* вычисляется из стандартизованной приспособленности по формуле (39):

$$a(i,t) = \frac{1}{1+s(i,t)}, \quad (39)$$

где *s(i, t)* – стандартизованная приспособленность для особи *i* поколения *t*.

Отрегулированная приспособленность находится в промежутке от 0 до 1. Чем больше его значение, тем лучше эта особь в популяции.

Необязательно использовать отрегулированную приспособленность в генетическом программировании, но это облегчает решение задачи. Отрегулированная приспособленность хорошо преувеличивает важность малых различий в значениях стандартизованной приспособленности. Таким образом, по мере улучшения популяции большой акцент делается на небольшие различия, что позволяет увидеть разницу между хорошей и очень хорошей особью. Это преувеличение особенно велико, если стандартизованная приспособленность достигает нуля, когда лучшее решение проблемы найдено. Например, если стандартизованная приспособленность находится в промежутке от 0 (лучший) до 64 (худший), отрегулированная пригодность двух плохих особей, оцениваемых 64 и 63, будет 0.0154 и 0.0159 соответственно. А для особей с оценкой 4 и 3 отрегулированная приспособленность равна 0.20 и 0.25 соответственно.

Данный эффект слабеет (но все равно значителен), когда лучшее значение стандартизованной приспособленности не может быть определено.

Заметим, что для других методов выбора, отличающихся от пропорциональных значениям приспособленности, отрегулированная функция неуместна и не используется.

1.3.1.2.4 Нормализованная приспособленность

Если метод селекции основывается на пропорциональности приспособленности, то понятие нормализованной приспособленности также необходимо.

Нормализованная приспособленность *n(i, t)* вычисляется с помощью значения отрегулированной приспособленности по формуле (40):

$$n(i,t) = \frac{a(i,t)}{\sum_{k=1}^{M} a(k,t)} \quad (40)$$

У нормализованной приспособленности есть три характеристики:

– она находится в промежутке от 0 до 1;

– чем больше ее значение, тем лучше особь популяции;

– сумма значений нормализованной приспособленности равна 1.

1.3.1.3 Репродукция

Операция воспроизведения для генетического программирования является основной движущей силой естественного отбора Дарвина и выживания наиболее приспособленных особей.

Репродукция состоит из двух этапов. Во-первых, одна особь выбирается из популяции в соответствии с каким-либо способом отбора, основанным на приспособленности. Во-вторых, выбранная особь копируется без изменений, из текущей популяции в новую популяцию (т.е. новое поколение).

Существует множество различных методов отбора, основанных на приспособленности. Наиболее популярным является селекция, пропорциональная значениям приспособленности.

Если $f(s_i(t))$ – это приспособленность особи s_i в популяции поколения t, то при пропорциональном приспособленности отборе вероятность того, что особь s_i будет скопирована в следующее поколение, как результат операции воспроизведения, будет вычисляться по формуле (41).

$$\frac{f(s_i(t))}{\sum_{j=1}^{M} f(s_j(t))}, \tag{41}$$

где $f(s_i(t))$ – это нормализованная приспособленность $n(s_i(t))$.

Среди альтернативных методов селекции можно выделить турнирный отбор и селекцию по рангу. При ранговом отборе выбор основывается на ранге (нечисловом значении) значений приспособленности особей в популяции. Данный метод уменьшает потенциально доминирующие эффекты сравнительно высокой приспособленности особей популяции путем создания предсказуемого, ограниченного отбора таких индивидов. В то же время ранговая селекция преувеличивает разницу между близко находящимися значениями приспособленности.

В турнирном отборе определенное количество особей (обычно два) выбираются случайным образом из популяции. Затем из них выбирается одна особь с лучшей приспособленностью.

Заметим, что родитель остается в популяции на протяжении всей селекции. Это означает, что разрешен повторный выбор. Родители могут быть выбраны и, в общем случае, выбираются более одного раза для воспроизведения в текущем поколении.

1.3.1.4 Скрещивание

Операция скрещивания (половая рекомбинация) в генетическом программировании изменяет популяцию путем создания нового потомства,

которое состоит из частей, взятых от каждого родителя. Скрещивание начинается с выбора двух особей-родителей и заканчивается созданием двух особей-потомков.

Первый родитель выбирается из популяции таким же способом отбора, основанным на значении пригодности, как и в операции репродукции, т.е. вероятность выбора первого родителя равна его нормированной пригодности. Второй родитель выбирается аналогично первому.

В начале операции скрещивании случайным образом выбирается одна случайная вершина у каждого дерева-родителя. Эта вершина становится точкой скрещивания для этих двух родителей. Следует отметить, что особи-родители, как правило, имеют разный размер.

Фрагмент скрещивания для определенного родителя – это поддерево, корнем которого является точка скрещивания этого родителя, состоящее из всего поддерева родителя, находящего ниже точки скрещивания. Это поддерево иногда может состоять из одного терминального символа.

Первый потомок получается путем удаления фрагмента скрещивания у первого родителя, а затем вставки фрагмента скрещивания второго родителя в точку скрещивания первого родителя. Второй потомок получается симметричным образом.

Если в точке пересечения одного из родителей находится терминальный символ, то поддерево второго родителя вставляется на место терминала в первом родителе (вводя тем самым поддерево вместо одной точки терминала), а терм первого родителя вставляется на место расположения поддерева во втором родителе. Это часто будет давать эффект создания потомства большей глубины.

Если терминальные символы расположены на обеих точках скрещивания, то операция скрещивания просто поменяет местами эти терминалы. Эффект скрещивания в данном случае равен эффекту узловой мутации.

1.3.1.5 Мутация

Операция мутации вносит случайные изменения в структуры популяции. Мутация – это бесполая операция, изменяющая только одно дерево.

Операция начинается с выбора вершины дерева случайным образом. Эта точка мутации может быть внутренним узлом дерева (функцией) или внешним узлом – листом (терминальным символом). Мутация удаляет значение выбранного узла и все, что находится ниже его по уровню, а затем вставляет случайным образом сгенерированное поддерево в точку мутации.

Результат мутации дерева можно увидеть на рисунке 8.

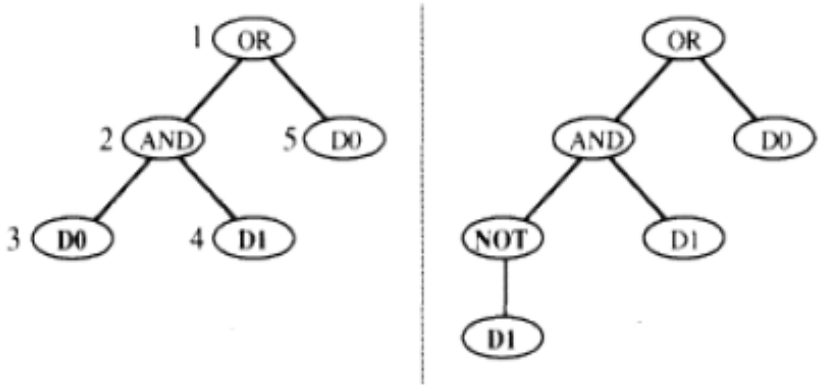

Рисунок 8 – Результат мутации дерева

Для деревьев используются следующие операторы мутации: узловая, усекающая и растущая.

Узловая мутация выполняется следующим образом:

– выбрать случайным образом узел и определить его тип;

– случайным образом выбрать из соответствующего множества вариантов узлов узел, отличный от рассматриваемого узла;

– поменять исходный узел на выбранный узел.

Усекающая мутация производится так:

– выбирается узел;

– случайным образом выбирается терминальный символ;

– обрезается ветвь узла мутации;

– вместо обрезанной ветви помещается выбранный терминальный символ.

Растущая мутация выполняется следующим образом:

– случайным образом определяется терминальный узел мутации;

– вычислить размер (сложность) остатка дерева;

– вырастить случайным образом новое дерево так, чтобы размер нового построенного дерева не превышал заданный порог.

1.3.2 Применение генетического программирования для построения признаков

В парадигме построения признаков генетическое программирование используется для получения нового набора признаков из исходного. Индивиды часто представляют собой признаки, подобные деревьям, функция пригодности обычно основывается на характеристиках прогнозирования классификатора, обученного этим признакам, в то время как операторы могут быть специфичными для области применения. Этот метод по существу выполняет поиск в новом пространстве признаков и помогает создавать высокопроизводительное подмножество признаков. Новые сгенерированные признаки часто могут быть более понятными и интуитивными, чем исходный набор признаков, что делает генетическое программирование хорошо подходящим для таких задач. Методология оценки основана на подходе обертки, рассмотренном в разделе 1.2.2.2.

Использование нового пространства признаков, полученное из построенных генетическим программированием признаков, может помочь улучшить точность классификации.

1.3.2.1 Описание алгоритма

Пусть есть множество классификаторов K (формула (42)).

$$K = \{k_1, k_2, \dots, k_n\}, \quad (42)$$

где k_i – классификатор определенного вида, например, kNN.

И пусть есть лес деревьев A (формула (43)).

$$A = \{a_1, a_2, \dots, a_m\}, \quad (43)$$

где каждое дерево a_i представляет собой сконструированный признак.

Каждое дерево оценивается a_i с помощью функции пригодности равной определенному классификатору k_i.

Тогда представим множество P (формула (44)).

$$P = \{P_1, P_2, \dots, P_n\}, \quad (44)$$

где $P_i = (A_{i1}, A_{i2}, \dots, A_{im}|k_i)$, при этом $k_i \neq k_j$.

Множество P представляет собой набор лесов A_{ij}, каждый из которых оценивается определенным классификатором. Каждый лес A_{ij} имеет оценку пригодности, которая равна погрешности классификации.

Формируем начальную популяцию P_{init}: случайным образом создаем деревья, собираем их в леса, а затем выбираем и назначаем каждому классификатор из оставшихся свободных классификаторов.

Далее выполняем репродукцию для оценки пригодности каждого леса. Оцениваем эффективность набора сконструированных признаков с помощью назначенного классификатора.

Затем на основе полученной пригодности проводится операция скрещивания. Особи выбираются пропорционально их значениям приспособленности.

С малой долей вероятности происходит мутация отдельных деревьев и их узлов. Полученные новые деревья добавляются к текущей популяции.

Вся текущая популяция оценивается и сокращается до нужных размеров, путем удаления наименее приспособленных лесов.

Выбираем лучший построенный набор признаков – лес – и запоминаем его. Также оцениваем среднюю приспособленность всей популяции для оценки общего качества полученных набором признаков.

Если пригодность лучшего сконструированного набор признаков – ошибка классификации – меньше заданной, то прекращаем выполнение алгоритма, т.к. решение найдено. В противном случае повторяем все стадии заново, постепенно улучшая общую приспособленность популяции решений.

Разберем подробно каждую рассмотренную операцию.

1.3.2.1.1 Формирование начальной популяции

Пусть есть функциональное множество $F = \{f_1, f_2, \ldots, f_n\}$ и множество терминалов $T = \{t_1, t_2, \ldots, t_m\}$, где f_i – функция, t_j – признак.

Случайным образом выбираем функцию f_i из функционального множества F. Эта функция становится корнем дерева.

Для каждой функции известно число ее аргументов – $z(f)$. Мы выбираем для каждого аргумента элемент из множества $C = F \cup T$, который становится узлом дерева.

Если в качестве узла была выбрана функция, то дальнейшее создание дерева продолжается рекурсивно так, как было описано выше.

Если в качестве любого узла выбирается терминальный символ, то этот узел становится листом, и процесс создания поддерева для этой вершины прекращается.

Таким образом создается необходимое количество деревьев. Затем деревья случайным образом объединяются в леса. Когда необходимое количество лесов собрано, мы формируем из них наборы и назначаем каждому свой уникальный классификатор.

1.3.2.1.2 Селекция

Вся текущая популяция, состоящая из набора лесов, оценивает на пригодность. Набор сконструированных признаков подается классификатору,

который предсказывает значение класса для обучающей выборке изображений. Ошибка классификации будет являться значением приспособленности входного набора признаков – леса A.

Далее необходимо сократить размеры популяции для сокращения вычислительной нагрузки. Для этого мы начинаем исключать наименее приспособленные леса. Если в конце отбора значения приспособленности двух последних лесов совпадают, то исключается лес, чья сложность – длина дерева – больше. Данное решение объясняется необходимостью получения не только точных, но и понятных признаков.

1.3.2.1.3 Кроссинговер

Случайным образом выбираются два множества P_i и P_j. Из каждого из этих множеств случайным образом выбирается родитель, вероятность выбора которого пропорциональна значению его пригодности.

Из двух выбранных лесов формируем один лес, который будет выступать в качестве временной отдельной популяции для обычного генетического программирования.

Для полученной популяции можно применить оператор кроссинговера. В связи с тем, что мы не может правильно оценить эффективность одного дерева – построенного признака – решения задачи классификации, деревья-родители выбираются из популяции случайным образом. При этом нет ограничения на повторное скрещивание.

После выбора родителей необходимо определить точки скрещивания. Случайным образом выбирается одна вершина у каждого дерева-родителя. Эта вершина и становится точкой скрещивания для этих двух родителей.

Далее выполняется обмен поддеревьями и получаются два симметричных потомка. Эти новые деревья будут добавлены в текущую популяцию, когда завершится процесс скрещивания. Полученные потомки не принимают участие в текущем кроссинговере, т.е. не могут быть выбраны в качестве родителей.

Когда процедура рекомбинации особей завершается, мы получаем новый лес – набор признаков. Необходимо выбрать для него лучший классификатор из двух начальных классификаторов множеств P_i и P_j. С этой целью оцениваем пригодность построенного набора признаков сначала для классификатора k_i, а затем для классификатора k_j. Выбираем наименьшую погрешность классификации – лучшую пригодность, и помещаем полученный лес в множество, которое соответствует наиболее эффективному классификатору.

1.3.2.1.4 Мутация

Задаются вероятности мутации: обмен лесов, замена дерева в лесе случайным образом созданным деревом, мутация одного дерева в лесе.

Случайным образом выбирается множество P_i. Каждый лес $A_l \in P_i$ проверяется на возможность мутации в соответствии с определенным коэффициентом мутации.

Если была выбрана мутация в виде обмена лесов, то из другого случайным образом выбранного множества P_j, где $P_i \neq P_j$, случайным образом выбирается лес $A_m \in P_j$. Далее леса меняются местами, т.е. меняется принадлежность лесов их множествам: $A_l \in P_j, A_m \in P_i$. Данный вид мутации обладает самой маленькой вероятностью наступления события, т.к. в ходе процесса выполнения генетического программирования в каждом множестве P_i должны образовываться наборы признаков, которые наиболее эффективны при их использовании классификатором множества P_i. В связи с этим рассмотренный вид мутации при удачном применении оказывает положительный эффект в виде увеличения разнообразия особей в популяции, но при частом использовании способен ухудшить качество классификации.

Если для любого леса A_i была выбрана мутация в виде замены деревьев, то случайным образом выбирается дерево и исключается из леса A_i. Далее создается новое дерево, как при формировании начальной популяции,

и включается в набор деревьев леса A_i. Данная мутация также обладает низким приоритетом, по рассмотренной в мутации обмена лесами причине возможного ухудшения прогнозирования.

При выборе мутации одного дерева в лесе применятся стандартная мутация генетического программирования. Выбирается вид мутации: узловая, усекающая или растущая. Затем случайным образом находится узел дерева, который будет подвергнут мутации. Далее выполняется выбранный вид мутации и получается новое дерево, которое добавляется в текущий лес, заменяя исходное дерево. Этот вид мутации обладает самым высоким приоритетом вероятности наступления события из разобранных, т.к. его воздействие на изменение структуры леса мало по сравнению с другими разновидностями мутации и совпадает с операцией скрещивания.

Были изучены разные способы автоматического построения признаков, позволяющие улучшить качество классификации при работе с медицинскими изображениями. Исследование проблемы классификации дало представление о новом возможном способе решения задачи повышения эффективности классификации. Было подробно изучено генетическое программирование, возможность его применения для конструирования признаков. Генетическое программирование позволяет создавать видоизменяемые программы и находить среди них наиболее подходящую программу. В связи с этим данный метод является подходящим для решения задачи увеличения различающей способности классификатора путем подачи автоматически построенных признаков, которые наиболее точно определяют важные в рамках задачи параметры изображения.

2 Практическая часть

Была поставлена задача автоматического построения признаков для решения задачи классификации медицинских изображений методом генетического программирования. Поиск готовых решений не дал результатов. Поэтому было принято решение о разработке собственной программы, предоставляющей необходимый результат.

2.1 Выбор языка программирования

В качестве языка программирования был выбран Python.

Python позволяет использовать эффективные высокоуровневые структуры данных и предлагает простой, но эффективный подход к объектно-ориентированному программированию. Сочетание изящного синтаксиса, динамической типизации в интерпретируемом языке делает Python идеальным языком для написания сценариев и ускоренной разработки приложений в различных сферах и на большинстве платформ.

Интерпретатор Python и разрастающаяся стандартная библиотека находятся в свободном доступе в виде исходников и двоичных файлов для всех основных платформ на официальном сайте Python [44] и могут распространяться без ограничений.

Python даёт возможность писать компактные и читабельные программы. Программы, написанные на Python, отличаются большей краткостью, чем эквивалентные на C, C++ или Java, по нескольким причинам:

– высокоуровневые типы данных позволяют вам выражать сложные операции в одной инструкции;

– группировка инструкций выполняется отступами, а не операторными скобками;

– нет необходимости в описании переменных или аргументов.

Python поддерживает несколько парадигм программирования, в том числе структурное, объектно-ориентированное, функциональное, императивное и аспектно-ориентированное. Основные архитектурные черты – динамическая типизация, автоматическое управление памятью, полная интроспекция, механизм обработки исключений, поддержка многопоточных вычислений и удобные высокоуровневые структуры данных. Код в Python организовывается в функции и классы, которые могут объединяться в модули.

2.2 Использованные библиотеки

2.2.1 scikit-learn

Для работы с классификаторами изображений была использована библиотека scikit-learn [23]. В scikit-learn оценочной функцией для классификации является объект Python, который реализует методы *fit (X, y)* и *pred (t)*.

Данная библиотека содержит такие классификаторы изображений, как классификатор ближайшего соседа, наивный байесовский классификатор, деревья решений C4.5, искусственную нейронную сеть и метод опорных векторов. Каждый классификатор подробно описан в документации и сопровождается примерами использования.

Библиотека также содержит модули для неконтролируемого обучения, в котором работа с входными данными, для которых неизвестно целевое значение. Целью таких задач может быть обнаружение групп схожих объектов в данных, это называется кластеризацией, или определение распределения данных во входном пространстве, обозначается оценкой плотности, или проецирование данных из высоко размерных пространств до двух или трех измерений с целью визуализации.

В связи с тем, что в данной работе используется контролируемое обучения для предсказания значения класса входного изображения, из

рассмотренной библиотеки будут использоваться только необходимые классификаторы.

2.2.2 OpenCV-Python

OpenCV (Open Source Computer Vision Library) [32] – это библиотека с лицензией BSD с открытым исходным кодом, которая включает в себя несколько сотен алгоритмов компьютерного зрения.

OpenCV-Python – это библиотека Python, предназначенная для решения задач компьютерного зрения.

OpenCV-Python использует Numpy, который является высоко оптимизированной библиотекой для числовых операций с синтаксисом стиля MATLAB. Все структуры массива OpenCV преобразуются в массивы Numpy и обратно. Это также упрощает интеграцию с другими библиотеками, использующими Numpy, такими как SciPy и Matplotlib.

OpenCV имеет модульную структуру, это означает, что пакет включает несколько библиотек. В данной работе были использованы следующие модули:

– core – компактный модуль, определяющий основные структуры данных, включая многомерный массив Mat и базовые функции, используемые всеми другими модулями.

– imgproc – модуль обработки изображений, который включает в себя фильтрацию линейных и нелинейных изображений, преобразования геометрических изображений (изменение размера, аффинное и перспективное преобразования), преобразование цветового пространства и гистограммы.

Рассмотренная библиотека была использована для получения признаков из входных изображений.

2.3 Особенности программы

2.3.1 Структура данных

Получение признаков из исходных изображений расположено в отдельном классе. Каждому признаку соответствует статичная функция класса, принимающая в качестве входного параметра изображение и дополнительные параметры, которые зависят от способа получения данной особенности изображения.

Для использования классификаторов был создан абстрактный класс с методами обучения и предсказания значения класса. Все классификаторы унаследованы от абстрактного классификатора и реализуют его методы, принимая дополнительные параметры по мере необходимости.

Особи популяции – деревья – представлены собственным классом. Его полями являются структура дерева, отображение дерева и пригодность. Структура дерева – это список списков номеров потомков, т.е. определенная позиция в списке означает номер вершины дерева, которой соответствует список потомков. Отображение дерева представлено в виде словаря, ключом которого является номер вершины, а значением – функция или терминальный символ. Поле пригодности – это вещественное число, обозначающее суммарную квадратичную ошибку определенного дерева. Класс дерева также содержит необходимые для проведения генетических операций поля: позиция скрещивания (номер вершины, которая вместе со своим поддеревом будет заменена при скрещивании) и потомок (дерево, полученное в результате скрещивания). Данный класс включает в себя функцию нахождения глубины дерева, методы нахождения, удаления и добавления поддерева, необходимые для половой рекомбинации, а также функции замены у дерева случайно выбранного терминального символа или функции на другой терм или функцию, нужные для проведения операции мутации.

Лес деревьев представлен собственным классом, который содержит деревья данного леса в виде списка и хранит свою последнюю полученную пригодность.

Для создания начальной популяции был создан отдельный класс. Полями класса являются глубина дерева (целое число) и созданная особь (дерево). Методы класса позволяют создавать деревья «полным» и «растущим» способами определенной глубины. При создании листов дерева происходит проверка на наличие переменных в данном дереве. Если переменных нет, то значением определенного листа становится переменная, случайным образом выбранная из множества всех переменных. Если в дереве уже есть переменные, то значение данного листа выбирается с равной долей вероятности из множества переменных и констант.

Оператор репродукции представлен собственным классом. Он включает в себе такие поля, как начальная популяция (список деревьев), выбранная популяция (список деревьев) и значения сумм отрегулированной пригодности для каждого набора значений свободных переменных (список действительных чисел). Данный класс также содержит методы получения суммарной отрегулированной пригодности начальной популяции, получения значения пригодности для отдельного дерева и метод отбора наиболее приспособленных особей. Также есть функция нахождения отрегулированной пригодности на основе стандартизованной пригодности и метод проверки наличия в дереве нетерминальных элементов (если в дереве нет функций, то такое дерево не должно попасть в следующую популяцию).

Для проведения операции скрещивания был создан отдельный класс, включающий в себя следующие поля: первый родитель, второй родитель, первый потомок и второй потомок, представленные в виде деревьев. Метод скрещивания случайным образом выбирает позиции у деревьев-родителей, проверяет функции, соответствующие вершинам этих позиций, на арность, и если она совпадает, то меняет местами определенные поддеревья, в противном случае поиск вершин для замены продолжается рекурсивно. Если

глубина рекурсии достигнет своего максимального значения равного 10, происходит выход из данного метода без создания потомков деревьев.

2.3.2 Структура программы

Производится выборка начальных данных: свободных и зависимых значений переменных.

Первоначально создаем начальную популяцию мощностью в 500 особей. Глубина каждого начального дерева не больше 5. Деревья генерируются растущим и полным методами напополам.

Далее выполняется репродукция. Для каждого дерева вычисляется значение его пригодности. В качестве меры приспособленности используется квадратичная ошибка. Это значение нормализуется, и на его основе выбираются наиболее подходящие особи.

На следующем шаге выполняется цикл до момента, когда количество особей нового поколения не станет равным 500. Новое поколение формируется из потомков и лучших особей предыдущего поколения. С вероятностью 0.9 случайным образом выбираются деревья-родители из особей, отобранных на шаге репродукции, и скрещиваются. Если размер потомков не превышает заданный (максимальная глубина дерева равна 16), то полученные деревья помещаются в следующее поколение. С вероятностью 0.1 в новую популяцию добавляется случайно выбранное из наиболее пригодных особей дерево текущего поколения.

После выполняется операция мутации. С вероятностью 0.01 каждая особь может мутировать с помощью узловой, растущей или усекающей мутации.

Последний этап состоит в проверке значения приспособленности каждой особи. Если значение меньше заданной точности, то рассматриваемое дерево становится одним из решений задачи. В случае если ни одна из особей популяции не достигла такого результата, возвращаемся к стадии

репродукции и начинаем сначала с новым поколением. Максимальное количество итераций равно 500.

Из всех полученных решений выбирается функция с минимальной квадратичной ошибкой для второстепенных данных. Даже если пригодность этого решения для основного множества значений не является наилучшей, решающее значение имеет наибольшая схожесть поведения тестовой и полученной функций, поэтому конечное решение выбирается на основе минимальной квадратичной ошибки для второстепенных данных.

2.4 Практические результаты

Для проверки работы программы был проведен эксперимент. Условия эксперимента: определим набор классификаторов, набор обучающих и проверочных изображений и множество исходных признаков. Эксперимент будет считаться успешным, если ошибка классификации при использовании найденного программой набора построенных признаков будет меньше 10%. Эксперимент будет считаться неуспешным, если многократный запуск программы (не менее 10 раз) не приведет к нахождению сконструированного набора признаков с требуемой ошибкой прогнозирования.

Эксперимент будет выполняться на тестовой машине с процессором Intel(R) Core(TM) i7-3517U 1.90GHz и ОЗУ 4.0 ГБ DDR3.

В набор классификаторов будут входить следующие классификаторы:

– классификатор ближайшего соседа;

– метод опорных векторов;

– деревья решений C4.5;

– наивный байесовский классификатор;

– искусственная нейронная сеть.

Набор обучающих и проверочных изображений был взят из доступного в интернете международного ресурса для разработки, обучения и оценки методов компьютерной диагностики для выявления и диагностики рака

легких, который носит название Консорциум базы данных изображений лёгких [43]. Данная коллекция изображений состоит из диагностических снимков и снимков раковых опухолей, полученных с помощью торакальной компьютерной томографии, с отмеченными аннотированными поражениями.

Множество исходных признаков представлено следующим набором:

– центр тяжести;

– коэффициент асимметрии;

– центроид;

– гистограмма;

– контрастность;

– фильрация;

– усреднение;

– размытие Гаусса;

– медианное размытие;

– двустороння фильтрация;

– эрозия;

– растяжение;

– морфологический градиент;

– производные Лапласа;

– операторы Собеля;

– преобразование Фурье;

– контуры.

Представленные классификаторы обучаются на исходных рассмотренных выше признаках, возвращая 1, если медицинское изображение содержит патологию и 0 в противном случае.

В результате работы программы были получены наборы сконструированных признаков со средней ошибкой классификации равной 10%. При этом использование лучшего набора признаков дает ошибку классификации не более 8%.

Следует отметить, что на результат работы программы сильно влияет исходный набор признаков. Поэтому достаточность входного набора признаков для эффективной классификации изображений является самостоятельной задачей и требует отдельного решения.

Также можно заметить, что полученные программой сконструированные признаки имеют достаточно сложный вид, что создает неудобства для их практического применения.

Данное решение задачи автоматического построения признаков для классификации медицинских изображений методом генетического программирования дает результат достаточной точности, что позволяет сделать вывод об успешном решении задачи.

ЗАКЛЮЧЕНИЕ

Было проведено исследование решения задачи построения признаков для классификации медицинских изображений методом генетического программирования.

Изучение медицинских изображений позволило предположить, что классификация такого рода данных с помощью интуитивного выделения признаков и применения классификатора будет выдавать результаты с большой погрешностью.

Поэтому были рассмотрены методы, позволяющие автоматически конструировать признаки для классификатора с целью уменьшения погрешности разделения медицинских изображений на классы.

Для решения задачи построения признаков для классификации медицинских изображений был выбран метод генетического программирования. Данный метод был подробно изучен, благодаря чему сформировалось понимание о структуре представления данных в программе, а также операций, применение которых способно привести к решению поставленной задачи.

Далее была разработан алгоритм решения задачи построения признаков на основе генетического программирования. Подробно рассмотрены генетические операторы, их влияние на промежуточный и конечный результаты работы, и на основании проделанной работы выбраны наиболее подходящие операторы репродукции, скрещивания и мутации.

По разработанному алгоритму была реализована программа. Проверено на практике выполнение и результат всех этапов алгоритма.

Затем было проведено тестирование работы программы на проверяющем наборе изображений. Результат тестирования продемонстрировал успешность решения задачи автоматического построения признаков для эффективной классификации медицинских изображений методом генетического программирования.

Следует отметить, что на результат работы программы сильно влияет исходный набор признаков. Поэтому достаточность входного набора признаков для эффективной классификации изображений является самостоятельной задачей и требует отдельного решения.

Также можно заметить, что полученные программой сконструированные признаки имеют достаточно сложный вид, что создает неудобства для их практического применения

Достоинством использования метода генетического программирования для решения задачи автоматического построения признаков для классификации медицинских изображений является точность прогнозирования результата. Если у нас есть дополнительная информация о патологии на изображении, мы легко можем использовать ее при определении исходного набора признаков. Концептуальная простота генетического программирования также является важным преимуществом его практического применения.

В качестве недостатка можно выделить отсутствие эффективных критериев окончания работы программы. Мы не можем предсказать, приведет ли дальнейшее выполнение программы к результату или нет, возможно, мы сможем получить более точный результат, чем уже найденный. Также выполнение программы требует значительных вычислительных ресурсов (увеличение мощности популяции и максимальной глубины дерева), что может стать ограничением для широкого применения и получения результатов высокой точности.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Alfred R. A genetic-based feature construction method for data summarisation //International Conference on Advanced Data Mining and Applications. – Springer Berlin Heidelberg, 2008. – С. 39-50.

2. Bhowan U. et al. Reusing genetic programming for ensemble selection in classification of unbalanced data //IEEE Transactions on Evolutionary Computation. – 2014. – Т. 18. – №. 6. – С. 893-908.

3. Cover T. M. Learning in pattern recognition //Methodologies of Pattern Recognition. – 1969. – Т. 39. – С. 111-132.

4. Cover T., Hart P. Nearest neighbor pattern classification //IEEE transactions on information theory. – 1967. – Т. 13. – №. 1. – С. 21-27.

5. Druck G., Mann G., McCallum A. Learning from labeled features using generalized expectation criteria //Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval. – ACM, 2008. – С. 595-602.

6. Duda, R., Hart, P., & Stork, D. (2001). Pattern classification. New York: Wiley.

7. Espejo P. G., Ventura S., Herrera F. A survey on the application of genetic programming to classification //IEEE Transactions on Systems, Man and Cybernetics, Part C: Applications and Reviews. – 2010. – Т. 40. – №. 2. – С. 121-144.

8. Fellbaum C. WordNet. – Blackwell Publishing Ltd, 1998.

9. Fix E., Hodges Jr J. L. Discriminatory analysis-nonparametric discrimination: consistency properties. – California Univ Berkeley, 1951.

10. Forman G. An extensive empirical study of feature selection metrics for text classification //Journal of machine learning research. – 2003. – Т. 3. – №. Mar. – С. 1289-1305. MLA.

11. Goldberg D.E. Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley, reading, MA, 1989.

12. Guo L. et al. Automatic feature extraction using genetic programming: An application to epileptic EEG classification //Expert Systems with Applications. – 2011. – Т. 38. – №. 8. – С. 10425-10436.

13. Guyon I., Elisseeff A. An introduction to variable and feature selection //Journal of machine learning research. – 2003. – Т. 3. – №. Mar. – С. 1157-1182.

14. Han J., Kamber M. Classification and prediction //Data mining: Concepts and techniques. – 2006. – С. 347-350.

15. Holland J.P. Adaptation in Natural and Artificial Systems.An Introductionary Analysis With Application to Biology? Control and Artificial Intelligence. University of Michigan, 1975.

16. Huang Y., Mitchell T. M. Text clustering with extended user feedback //Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval. – ACM, 2006. – С. 413-420.

17. Kohavi R., John G. H. Wrappers for feature subset selection //Artificial intelligence. – 1997. – Т. 97. – №. 1-2. – С. 273-324.

18. Koza J. R. Genetic programming: on the programming of computers by means of natural selection. – MIT press, 1992. – Т. 1.

19. Kramer S., Lavrač N., Flach P. Propositionalization approaches to relational data mining //Relational data mining. – Springer Berlin Heidelberg, 2001. – С. 262-291.

20. Langdon W. B. Genetic programming and data structures. – University College London. Department of Computer Science, 1996.

21. Lavrač N., Džeroski S., Grobelnik M. Learning nonrecursive definitions of relations with LINUS //Machine learning—EWSL-91. – Springer Berlin/Heidelberg, 1991. – С. 265-281.

22. Lim S. H., Wang L. L., DeJong G. Explanation-Based Feature Construction //IJCAI. – 2007. – Т. 7. – С. 931-936.

23. Machine Learning in Python [Электронный ресурс]. URL: http://scikit-learn.org/stable/ (дата обращения – 05.03.2017).

24. Markovitch S., Rosenstein D. Feature generation using general constructor functions //Machine Learning. – 2002. – Т. 49. – №. 1. – С. 59-98.

25. Matheus C. J., Rendell L. A. Constructive Induction On Decision Trees //IJCAI. – 1989. – Т. 89. – С. 645-650.

26. Mitchell M. An introduction to genetic algorithms. – MIT press, 1998.

27. Muni D. P., Pal N. R., Das J. Genetic programming for simultaneous feature selection and classifier design //IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics). – 2006. – Т. 36. – №. 1. – С. 106-117.

28. Neshatian K., Zhang M. Pareto front feature selection: using genetic programming to explore feature space //Proceedings of the 11th Annual conference on Genetic and evolutionary computation. – ACM, 2009. – С. 1027-1034.

29. Neshatian K., Zhang M., Andreae P. A filter approach to multiple feature construction for symbolic learning classifiers using genetic programming //IEEE Transactions on Evolutionary Computation. – 2012. – Т. 16. – №. 5. – С. 645-661.

30. O'Neill M. et al. Open issues in genetic programming //Genetic Programming and Evolvable Machines. – 2010. – Т. 11. – №. 3-4. – С. 339-363.

31. Omar F. Zaidan and Jason Eisner. Using annotator rationales to improve machine learning for text categorization. In In NAACL-HLT, pages 260–267, 2007.

32. OpenCV documentation [Электронный ресурс]. URL: http://docs.opencv.org/3.0-beta/doc/py_tutorials/py_tutorials.html (дата обращения – 10.03.2017).

33. Pagallo G. Learning DNF by Decision Trees //IJCAI. – 1989. – Т. 89. – С. 639-644.

34. Pantel P., Lin D. Discovering word senses from text //Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining. – ACM, 2002. – С. 613-619.

35. Quinlan J. R. C4. 5: Programs for Empirical Learning Morgan Kaufmann //San Francisco, CA. – 1993.

36. Raghavan H., Allan J. An interactive algorithm for asking and incorporating feature feedback into support vector machines //Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval. – ACM, 2007. – С. 79-86.

37. Roth D., Small K. Interactive feature space construction using semantic information //Proceedings of the Thirteenth Conference on Computational Natural Language Learning. – Association for Computational Linguistics, 2009. – С. 66-74.

38. Smith M. G., Bull L. Genetic programming with a genetic algorithm for feature construction and selection //Genetic Programming and Evolvable Machines. – 2005. – Т. 6. – №. 3. – С. 265-281.

39. Smith M., Bull L. Improving the human readability of features constructed by genetic programming //Proceedings of the 9th annual conference on Genetic and evolutionary computation. – ACM, 2007. – С. 1694-1701.

40. Sondhi P. Feature construction methods: a survey //sifaka. cs. uiuc. edu. – 2009. – Т. 69. – С. 70-71.

41. Specia L. et al. An investigation into feature construction to assist word sense disambiguation //Machine Learning. – 2009. – Т. 76. – №. 1. – С. 109-136.

42. Specia L. et al. Word sense disambiguation using inductive logic programming //International Conference on Inductive Logic Programming. – Springer Berlin Heidelberg, 2006. – С. 409-423.

43. The Lung Image Database Consortium image collection [Электронный ресурс]. URL: https://wiki.cancerimagingarchive.net/display/Public/LIDC-IDRI (дата обращения – 01.04.2017).

44. The official home of the Python Programming Language [Электронный ресурс]. URL: https://www.python.org/ (дата обращения – 12.10.2016).

45. Tzanakou E. M. (ed.). Supervised and unsupervised pattern recognition: feature extraction and computational intelligence. – CRC Press, 1999.

46. Vipin P. N. T. M. S. Kumar, Introduction to Data Mining [M]. – 2006.

47. Xinjie Yu, Mitsuo Gen. Introduction to Evolutionary Algorithms. Springer, London Limited 2010.

48. Yang D. S., Rendell L. A., Blix G. A Scheme for Feature Construction and a Comparison of Empirical Methods //IJCAI. – 1991. – С. 699-704.

49. Zaidan O. F., Eisner J. Modeling annotators: A generative approach to learning from annotator rationales //Proceedings of the Conference on Empirical Methods in Natural Language Processing. – Association for Computational Linguistics, 2008. – С. 31-40.

50. Zaidan O., Eisner J., Piatko C. D. Using" Annotator Rationales" to Improve Machine Learning for Text Categorization //HLT-NAACL. – 2007. – С. 260-267.

51. Коэн П. Д. Теория множеств и континуум-гипотеза. – М.: Мир, 1969. – С. 13-86.

52. Эволюционные вычисления [Электронный ресурс]. URL: http://www.intuit.ru/studies/courses/14227/1284/info (дата обращения – 01.12.2015).