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 70 с., 6 рис., 10 источников.

КЛАССИФИКАЦИЯ МЕДИЦИНСКИХ ИЗОБРАЖЕНИЙ, ПОСТРОЕНИЕ ПРИЗНАКОВ, ГЕНЕТИЧЕСКОЕ ПРОГРАММИРОВАНИЕ

Объект исследования – классификация изображений.

Предмет исследования – алгоритм построения признаков для классификации.

Цель работы – нахождение и отработка методики восстановления функциональной зависимости данных.

Методы исследования: изучение 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» (Micheli-Tzanakou, 2000), «идеальная функция извлечения даст представление, что сделает работу классификатора тривиальной» (Duda, Hart, & Stork, 2001). 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 решения задачи 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, 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

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

4) S' – набор тестовых примеров таких, что

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

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.

$$x = < f_1, f_2, f_3, \ldots, f_n >,$$

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

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

где $\varphi_i \in F_T \; \forall \; i = 1...N$ 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 [1], [2]. 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, присутствующих в данных. Примерами неконтролируемых задач обучения являются кластеризация и обнаружение ассоциации. Цель кластеризации состоит в том, чтобы разделить данные на различные группы, находя группы 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 KNN

Классификатор ближайшего соседа [Cover & Hart, 1967] 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, …, 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 []. Исследования, которые применяли классификацию по правилу ближайшего соседа, обычно получали хорошие результаты. Кавер и Харт (1967; см. также Кавер, 1969) 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

$$E(C_n) = R_n.$$

Ясно, что R_n с ростом n стремится к R_∞ (формальное доказательство см. [Кавер и Харт, 1967]). Пусть R^* будет риском ошибочной классификации любого случайным образом выбранного наблюдения при использовании некоторого неизвестного байесовского метода оптимальной классификации. Кавер и Харт доказали, что для случая двух классов

$$R^* \le R_\infty \le 2R^*(1 - R^*) \quad (65)$$

В табл. 4.3 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 [Коэн, 1969]. 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.

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 SVM

Метод опорных векторов (SVM – 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.

Даны обучающая выборка D – набор из n объектов, имеющих p параметров:

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

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:

$$w * x - b = 0,$$

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:

$$w * x - b = 1$$

$$w * x - b = -1$$

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 справедливо:

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

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

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

Далее аналитическим способом решаем задачу оптимизации:

$$\|w\| \to min$$

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

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:

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

Путем введения множителей Лагранжа, задача с ограничениями может быть выражена как задача без ограничений:

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

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:

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

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:

$$w * x_i - b = \frac{1}{y_i} = y_i \leftrightarrow b = w * x_i - y_i.$$

Последнее позволяет определить смещение *b*.

На практике применяют усреднение N_{sv} по всем опорным векторам:

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

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:

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

где $a_i \geq 0$.

Ограничение минимизации для b

$$\sum_{i=1}^{n} a_i y_i = 0$$

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

W может быть вычислено благодаря условиям:

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

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 C4.5

Алгоритм C4.5 строит классификатор в форме дерева решений. 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 того, что случайно выбранный пример из множества *S* будет принадлежать к классу C_j

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

Согласно теории информации, количество содержащейся в сообщении информации, зависит от ее вероятности

$$\log_2(\frac{1}{p}) \text{ (1)}$$

Поскольку мы используем логарифм с двоичным основанием, то выражение (1) дает количественную оценку в битах.

Выражение

$$Info(T) = -\sum_{j=1}^{k} \frac{freq(C_j,T)}{|T|} * \log_2 \frac{freq(C_j,T)}{|T|} \text{ (2)}$$

дает оценку среднего количества информации, необходимого для определения класса примера из множества *T*. В терминологии теории информации выражение (2) называется энтропией множества *T*.

Ту же оценку, но только уже после разбиения множества *T* по *X*, дает следующее выражение:

$$Info_x(T) = \sum_{i=1}^{n} \frac{T_i}{T} * Info(T_i) \text{ (3)}$$

Тогда критерием для выбора атрибута будет являться следующая формула:

$$Gain(X) = Info(T) - Info_x(T) \text{ (4)}$$

Критерий (4) 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 (4) 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 (3) 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 (3) будет иметь минимальное значение и, соответственно, критерий (4) 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 можно выбрать среднее между значениями v_i и v_{i+1}

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

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

Формулы (2), (3) и (4) последовательно применяются ко всем потенциальным пороговым значениям и среди них выбирается то, которое дает максимальное значение по критерию (4). Далее это значение сравнивается со значениями критерия (4), подсчитанными для остальных атрибутов. Если выяснится, что среди всех атрибутов данный числовой атрибут имеет максимальное значение по критерию (4), 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 $D^m = \{(d_1, c_1), \ldots, (d_m, c_m),$ 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.

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

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:

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

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

Теорема. Решением этой задачи является алгоритм

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

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.

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

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.

1.2.2 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 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 1.

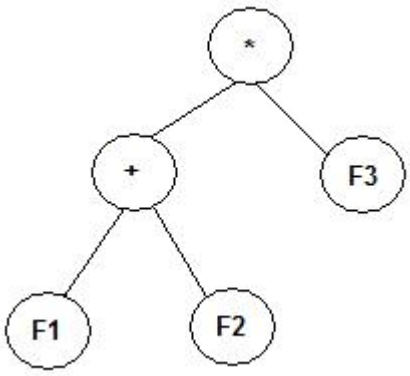

Рисунок 1 – Древовидное представление сконструированного признака $\varphi_i \in F_N$

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 (см. [Guyon and Elisseeff, 2003] и [Forman, 2003]). Мы можем свободно классифицировать эти методы по двум категориям: фильтры и обертки [Kohavi and John, 1997].

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 [Pagallo, 1989], 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 [Matheus and Rendell, 1989] и DC Fringe [Yang et al., 1991] – 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 [Маркович и Розенштейн, 2002] 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 [Lavrac et al., 1991]. В последующей литературе проблема

идентификации хороших признаков с представлением первого порядка была рассмотрена в пропозициональных (основанных на характеристиках) подходах [Kramer et al., 2000].

В работах Specia et. Al. [Specia et al., 2007; 2009] 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 вида

$$h_i: Class\ (x, c) < -\varphi_i\ (x),$$

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 [Roth и Small, 2009] 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 [Fellbaum, 1998; Pantel and Lin, 2002] 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 2 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 $\varphi_{Compass}$-$Direction$= $\varphi_{east} \vee \varphi_{west} \vee \varphi_{north}$

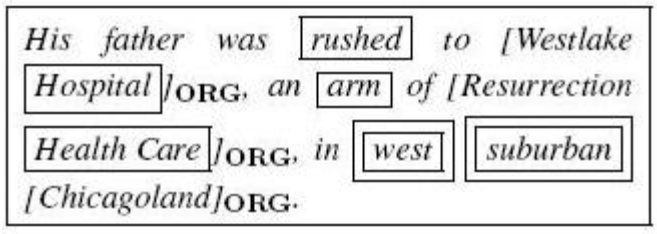

Рисунок 2 – Пример работы протокола CCCC

Рисунок 2 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 [Raghavan and Allan, 2007], 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 [Хуан и Митчелл, 2006; Зайдан и Эйснер, 2007 год; Druck et al., 2008; Зайдан и Эйснер, 2008 год; Lim et al., 2007]. В частности, Лим и др. [Lim et al., 2007] 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 / 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 Описание 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 создания 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.

1.3.2 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 Начальные структуры

1.3.2.2 Приспособленность

1.3.2.3 Репродукция

1.3.2.4 Скрещивание

1.3.2.5 Мутация

2 Практическая часть

Была поставлена задача разработки и реализации алгоритма автоматического построения признаков для классификации медицинских изображений на два класса: с патологией и без патологии. Поиск готовых образцов не дал результатов, найденные программы выдавали результат, точность которого была меньше необходимой. 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 http://www.python.org 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 Особенности программы

2.2.1 Структура данных

2.2.2 Структура программы

2.3 Практические результаты

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 проблемы.

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 для тестового набора данных. Результат тестирования продемонстрировал успешность решения задачи построения признаков для классификации с помощью генетического программирования.

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) для решения задач высокой точности.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Эволюционные вычисления [Электронный ресурс]. URL: http://www.intuit.ru/studies/courses/14227/1284/info

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

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

4. Koza, John R. Genetic programming: on the programming of computers by means of natural selection. MIT Press, Cambridge, MA, USA, 1992.

5. Langdon, William B. Genetic programming and data structures. Department of Computer Science, University College London, 1996.

6. Mitchell Melanie. An introduction to Genetic Algorithms. MIT Press, Cambridge, London, 1998.

7. The Python Tutorial [Электронный ресурс]. URL: https://docs.python.org/2/tutorial/index.html

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

9. Isabelle Guyon and Andr´e Elisseeff. An introduction to variable and feature selection. J. Mach. Learn. Res., 3:1157–1182, 2003.

10. George Forman. An extensive empirical study of feature selection metrics for text classification. J. Mach. Learn. Res., 3:1289–1305, 2003.

11. Ron Kohavi and George H. John. Wrappers for feature subset selection. Artif. Intell., 97(1-2):273–324, 1997.

12. Giulia Pagallo. Learning dnf by decision trees. In Proceedings of the Eleventh International Joint Conference on Artificial Intelligence, pages 639–644. Morgan Kaufmann, 1989.

13. Christopher Matheus and Larry A. Rendell. Constructive induction on decision trees. In Proceedings of the Eleventh International Joint Conference on Artificial Intelligence, pages 645–650. Morgan Kaufmann, 1989.

14. Dershung Yang, Larry Rendell, and Gunnar Blix. A scheme for feature construction and a comparison of empirical methods. In Proceedings of the Twelfth International Joint Conference on Artificial Intelligence, pages 699–704. Morgan Kaufmann, 1991.

15. Shaul Markovitch and Dan Rosenstein. Feature generation using general constructor functions. Mach. Learn., 49(1):59–98, 2002.

16. Nada Lavrac, Saso Dzeroski, and Marko Grobelnik. Learning nonrecursive definitions of relations with linus. In EWSL-91: Proceedings of the European working session on learning on Machine learning, pages 265–281, New York, NY, USA, 1991. Springer-Verlag New York, Inc.

17. Stefan Kramer, Nada Lavraˇc, and Peter Flach. Propositionalization approaches to relational data mining, pages 262–286, 2000.

18. Lucia Specia, Ashwin Srinivasan, Ganesh Ramakrishnan, and Maria Das Volpe Nunes. Word sense disambiguation using inductive logic programming, pages 409–423, 2007.

19. Lucia Specia, Ashwin Srinivasan, Sachindra Joshi, Ganesh Ramakrishnan, and Maria Gracas Volpe Nunes. An investigation into feature construction to assist word sense disambiguation. Mach. Learn., 76(1):109–136, 2009.

20. Dan Roth and Kevin Small. Interactive feature space construction using semantic information. In Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL-2009), pages 66–74, Boulder, Colorado, June 2009. Association for Computational Linguistics.

21. Christiane Fellbaum. WordNet: An Electronic Lexical Database. MIT Press, 1998.

22. Patrick Pantel and Dekang Lin. Discovering word senses from text. In In Proceedings of ACM SIGKDD Conference on Knowledge Discovery and Data Mining, pages 613–619, 2002.

23. Hema Raghavan and James Allan. An interactive algorithm for asking and incorporating feature feedback into support vector machines. In SIGIR'07: Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval, pages 79–86, New York, NY, USA, 2007. ACM.

24. Yifen Huang and Tom M. Mitchell. Text clustering with extended user feedback. In Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pages 413–420. SIGIR, 2006.

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

26. Omar Zaidan and Jason Eisner. Modeling annotators: A generative approach to learning from annotator rationales. In EMNLP, pages 31–40. ACL, 2008.

27. Gregory Druck, Gideon S. Mann, and Andrew McCallum. Learning from labeled features using generalized expectation criteria. In Sung H. Myaeng, Douglas W. Oard, Fabrizio Sebastiani, Tat S. Chua, Mun K. Leong, Sung H. Myaeng, Douglas W. Oard, Fabrizio Sebastiani, Tat S. Chua, and Mun K. Leong, editors, SIGIR, pages 595–602. ACM, 2008.

28. Shiau Hong Lim, Li-Lun Wang, and Gerald DeJong. Explanation-based feature construction. In IJCAI07, the Twentieth International Joint Conference on Artificial Intelligence, pages 931–936, 2007.

29. J. Han and M. Kamber, Data Mining—Concepts and Technique (The Morgan Kaufmann Series in Data Management Systems), 2nd ed. San Mateo, CA: Morgan Kaufmann, 2006.

30. P.-N. Tan, M. Steinbach, and V. Kumar, Introduction to Data Mining. Reading, MA: Addison-Wesley, 2005.

31. J. Ross Quinlan. C4.5: Programs for Machine learning. Morgan Kaufmann Publishers 1993.

32. К. Шеннон. Работы по теории информации и кибернетике. М. Иностранная литература, 1963

33.