Федеральное государственное автономное образовательное учреждение высшего образования
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»
Факультет компьютерных наук

ПРОЕКТ НА ТЕМУ

РЕАЛИЗАЦИЯ «ЛЕНИВОГО» МЕТОДА БИНАРНОЙ КЛАССИФИКАЦИИ

Студент: Рог Алексей Сергеевич
Группа: мНОД19_ИССА

Москва 2019

Задача

Данная работа посвящена решению задачи бинарной классификации: на основе обучающей выборки (train dataset), для которой известны метки целевого класса, необходимо восстановить значения целевого класса для элементов тестовой выборки (test dataset).

Предлагается использовать метод ленивой классификации, то есть не строить всё множество классификаторов на основе обучающей выборки, а принимать решение по классификации нового объекта при его поступлении.

Описание используемых алгоритмов

При выполнении работы были использованы несколько (а конкретно три) подхода на основе «генераторов».

1. *Алгоритм 1*

Данный алгоритм выглядит наиболее естественным и в то же время имеет наименьшую вычислительную сложность. Используется подход большинства голосов, мы считаем, что объект голосует, если он имеет достаточно много общих атрибутов с квалифицируемым объектом, тогда число голосов определяется как

$$\left|\left\{x \in G^{\pm}: \frac{|(g' \cap x')|}{|M|} \geq \alpha\right\}\right|$$

где g' – описание классифицируемого объекта (оператор “prime”), G^+ (G^-) – множество всех объектов с целевым классом "+" ("-"), α – численный параметр, который мы определяем в данной работе.

Выбор целевого класса для нового объекта происходит на основе сравнения нормированного числа голосов:

$$\left|\left\{x \in G^{\pm}: \frac{|(g' \cap x')|}{|M|} \geq \alpha\right\}\right| / |G^{\pm}|$$

2. *Алгоритм 2*

Самый тяжёлый (в смысле вычислительной сложности) из всех использованных алгоритмов. Для каждого нового объекта (из test dataset) и для каждого имеющегося объекта (из train dataset) мы находим количество элементов противоположного класса (относительно имеющегося объекта) в который укладываются пересечение атрибутов нового и имеющегося объектов. Данную величину называют достоверностью. Каждый объект из train set либо голосует, либо не голосует за новый объект на основе величины достоверности. Приведём вид агрегирующей функции[1] для наглядности:

$$\left|\left\{x \in G^{\pm}: \frac{\left|(g' \cap x')^{\mp}\right|}{|G^{\mp}|} \leq \alpha\right\}\right|$$

По приведённой формуле мы считаем голоса за положительный целевой класс (выбор верхнего знака в формуле) и за отрицательный (выбор нижнего знака в формуле). Присуждение того или иного класса происходит на основе сравнения нормированного числа этих голосов:

$$\left|\left\{x \in G^{\pm}: \frac{\left|(g' \cap x')^{\mp}\right|}{|G^{\mp}|} \leq \alpha\right\}\right| / |G^{\pm}|$$

[1] В самой формуле также использованы неочевидные операторы “+” и “-“. Эти операторы действуют точно также, как и оператор ‘prime’, но имеют другую область значений: $+: 2^M \to 2^{G^+}$

3. *Алгоритм 3*

Данный алгоритм также сравнительно быстрый (в смысле вычислительной сложности) и кажется довольно естественным. Идея выглядит следующим образом: мы не ведём подсчёт голосов, но вычисляем количество общих признаков отдельно для каждого объекта x из train dataset и данного классифицируемого объекта y. Затем, если наибольшее пересечение[2] соответствует объекту x, принадлежащему целевому классу "+", то и объекту y присваивается класс "+". Аналогично, мы присваиваем объекту y класс "-", если x из класса "-". Может оказаться, что y имеет наибольшее пересечение с несколькими элементами x_1 и x_2, принадлежащим разным классам. В этом случае мы рассматриваем вторые по количеству общих атрибутов элементы из train dataset и т.д. В случае если число общих атрибутов среди выбранных наибольших по пересечению объектов окажется эквивалентным для объектов различных классов, данный классифицируемый объект остаётся неопределённым (undefined).

В алгоритме 3 нет естественного параметра для минимизации, но в работе мы рассмотрели работу данного алгоритма при различных значениях числа α рассматриваемых наибольших по числу общих атрибутов объектов. Логично ожидать, что точность данного алгоритма будет расти при увеличении этого числа и выйдет на некоторый постоянный уровень, соответствующий такому значению α, что для каждого нового объекта (каждого объекта из test dataset) алгоритм сможет однозначно присудить ему значение целевого класса.

Выбор оптимальных параметров

Для выбора параметров (α в описанных алгоритмах) был использован метод скользящего контроля (cross-validation). Метод был реализован следующим образом: сначала выбирается некоторое значение параметра α, которое далее варьируется для поиска оптимального значения. Для данного train dataset мы производили разбиение на три части ($k = 3$)[3], последовательно предсказывали классы элементов одной из частей разбиения ("тренируясь" на других), каждый раз вычисляя метрики качества. Результаты затем были усреднены по трём итерациям. Описанная процедура выполняется для каждого из десяти тренировочных наборов данных и результаты (метрики качества) усреднялись по всем десяти наборам.

Метод скользящего контроля был реализован для различных значений параметров α, а результатом данной процедуры стал выбор значений параметров α, дающих наилучшие результаты. Полученные результаты представлены в таблице 2. Лучшие значения параметров с соответствующими метриками качества вынесены в таблицу 1:

Таблица 1. Оптимальные параметры для всех алгоритмов.

Algorithm number	Parameter	Accuracy	Precision	Recall	F1 score
1	0,70	0,978	0,997	0,969	0,983
2	0,00	0,972	0,962	0,997	0,979
3	25	0,977	0,968	0,997	0,983

[2] Мы используем тут термин «пересечение», так как для двух объектов x и y количество их общих признаков равно $|x' \cap y'|$

[3] Такое значение k было выбрано в связи с высокой вычислительной сложностью второго алгоритма. Для сравнения алгоритмов между собой разумно провести метод скользящего контроля для одного и того же значения k

Таблица 2. Сводка полученных по методу скользящего контроля результатов.

	Parameter	Accuracy	Precision	Recall	F1 score
Algorithm 1	0,55	0,739	0,845	0,737	0,787
	0,60	0,831	0,931	0,802	0,861
	0,65	0,831	0,931	0,802	0,861
	0,70	0,978	0,997	0,969	0,983
	0,75	0,978	0,997	0,969	0,983
	0,80	0,000	0,000	0,000	0,000
	0,85	0,000	0,000	0,000	0,000
	0,90	0,000	0,000	0,000	0,000
	0,95	0,000	0,000	0,000	0,000
	1,00	0,000	0,000	0,000	0,000
Algorithm 2	0,45	0,693	0,809	0,697	0,747
	0,40	0,649	0,817	0,596	0,689
	0,35	0,611	0,773	0,573	0,657
	0,30	0,620	0,768	0,600	0,673
	0,25	0,601	0,771	0,554	0,644
	0,20	0,597	0,741	0,591	0,657
	0,15	0,612	0,739	0,628	0,679
	0,10	0,625	0,744	0,650	0,693
	0,05	0,676	0,806	0,666	0,729
	0,00	0,972	0,962	0,997	0,979
Algorithm 3	5	0,858	0,992	1,000	0,996
	10	0,975	0,978	0,999	0,988
	15	0,975	0,972	0,999	0,985
	20	0,976	0,969	0,998	0,983
	25	0,977	0,968	0,997	0,983
	30	0,977	0,968	0,997	0,983
	35	0,977	0,968	0,997	0,983
	40	0,977	0,968	0,997	0,983
	45	0,977	0,968	0,997	0,983
	50	0,977	0,968	0,997	0,983

Полученные результаты легко объяснимы и соответствуют ожиданиям.

Для первого алгоритма наилучший результат мы получаем при высоком значении параметра $\alpha = 0.7$, и действительно, разумно требовать, чтобы объект голосовал, когда число общих атрибутов велико. При слишком же больших значениях параметра $\alpha \geq 0.8$ при проведении скользящего контроля объекты перестают голосовать, что означается, что совпадений атрибутов на более чем 80% не было обнаружено.

Мы получили хаотичные результаты для значений параметра $0.25 < \alpha \leq 0.45$ для второго алгоритма. Это говорит о том, что значение достоверности большее, чем 0.25, не может служить серьёзным основанием для классификации объекта на наших данных. Далее при уменьшении значения параметра α результаты улучшаются. Оказывается, что наилучшие результаты соответствуют нулевому значению достоверности: объект будет голосовать только если общие для

него и квалифицируемого объекта атрибуты не включаются ни в один из элементов противоположного класса.

Как было сказано ранее, точность третьего алгоритма выходит на постоянный уровень после значения параметра $\alpha = 25$.

При оптимальных значениях параметров α все используемые алгоритмы показывают высокую точность; алгоритм 2 незначительно уступает другим по метрике F1 score.

Использование алгоритмов на тестовых данных

Рассмотрим работу наших классификаторов на десяти предоставленных тестовых наборах данных. В качестве значений параметров были использованы найденные ранее оптимальные величины. Результаты приведены в таблице 3.

Таблица 3. Метрики качества при классификации объектов из тестовых наборов данных.

Algorithm number	Metric	Average	set1	set2	set3	set4	set5	set6	set7	set8	set9	set10
1	Accuracy	0,986	0,935	0,989	1,000	1,000	0,955	1,000	1,000	0,991	1,000	0,989
	Precision	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000
	Recall	0,979	0,902	0,980	1,000	1,000	0,935	1,000	1,000	0,986	1,000	0,983
	F1 score	0,989	0,948	0,990	1,000	1,000	0,967	1,000	1,000	0,993	1,000	0,991
2	Accuracy	0,990	1,000	0,977	1,000	0,978	0,978	1,000	0,991	0,991	0,990	1,000
	Precision	0,986	1,000	0,962	1,000	0,967	0,969	1,000	0,986	0,986	0,986	1,000
	Recall	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000
	F1 score	0,993	1,000	0,981	1,000	0,983	0,984	1,000	0,993	0,993	0,993	1,000
3	Accuracy	0,987	1,000	0,977	0,980	0,978	0,989	1,000	0,974	0,991	0,990	0,989
	Precision	0,980	1,000	0,962	0,970	0,967	0,984	1,000	0,959	0,986	0,986	0,983
	Recall	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000
	F1 score	0,990	1,000	0,981	0,985	0,983	0,992	1,000	0,979	0,993	0,993	0,992

По результатам, приведённым в таблице 3, можно ещё раз убедиться, что алгоритмы обладают высокой точностью, и нет причин выделять какой-либо из них среди остальных.

Использование собственного набора данных

Было предложено проверить работу наших методов на собственных данных. Мною был выбран набор данных “Divorce Predictors data set Data Set” из репозитория UCI Machine Learning Repository. Данный набор содержит 170 записей и 54 атрибута, что принципиально отличается от нашего набора Tic-Tac-Toe, содержащего больше число записей и всего 9 атрибутов.

Выбранный набор данных был составлен на основе анкетирования людей, а целевой класс – разведётся эта пара (“1” или “+”) или не разведётся (“0” или “-“). Значения всех атрибутов – это

числа от нуля до четырёх (в анкетировании предлагалось для каждого утверждения сказать на сколько оно правдиво по 4-х бальной шкале).

Мы используем метод скользящего контроля, как и ранее, для определения точности наших алгоритмов на данном методе. Результаты представлены в таблице 4.

Таблица 4. Результаты скользящего контроля для собственных данных[4]

	Parameter	Accuracy	Recall	Precision	F1 score
Algorithm 1	0,18	0,977	0,952	1,000	0,976
	0,16	0,977	0,952	1,000	0,976
	0,14	0,977	0,952	1,000	0,976
	0,12	0,977	0,952	1,000	0,976
	0,10	0,977	0,952	1,000	0,976
	0,08	0,953	0,916	1,000	0,956
	0,06	0,941	0,903	1,000	0,948
	0,04	0,941	0,949	1,000	0,974
	0,02	0,906	0,948	1,000	0,973
	0,00	0,000	0,000	0,000	0,000
Algorithm 2	0,45	0,700	0,754	1,000	0,858
	0,40	0,706	0,754	1,000	0,858
	0,35	0,789	0,883	1,000	0,937
	0,30	0,836	0,933	1,000	0,965
	0,25	0,847	0,933	1,000	0,965
	0,20	0,859	0,934	1,000	0,965
	0,15	0,871	0,934	1,000	0,965
	0,10	0,882	0,934	1,000	0,965
	0,05	0,912	0,936	1,000	0,967
	0,00	0,965	0,939	1,000	0,968
Algorithm 3	2	0,977	0,952	1,000	0,976
	4	0,977	0,952	1,000	0,976
	6	0,977	0,952	1,000	0,976
	8	0,977	0,952	1,000	0,976
	10	0,977	0,952	1,000	0,976
	12	0,977	0,952	1,000	0,976
	14	0,977	0,952	1,000	0,976
	16	0,977	0,952	1,000	0,976
	18	0,977	0,952	1,000	0,976
	20	0,977	0,952	1,000	0,976

Скользящий контроль показал интересные результаты для первого алгоритма. Оптимальное значение параметра оказывается существенно ниже, чем для данных Tic-Tac-Toe, что на самом деле неудивительно. При таком широком разнообразии атрибутов (54 атрибута по 5 возможных значений для каждого) высокая схожесть различных записей (в смысле количества

[4] В таблицу включены только значения параметров из содержательных областей, в то время как скользящий контроль осуществлялся для большего числа различных значений параметров.

общих атрибутов) очень маловероятна, тем более, что в выбранном наборе данных имеется всего 170 записей (сравнительно мало). Примечательно также, что даже при очень малых значениях параметра $\alpha = 0.02$ точность метода довольно велика. Это говорит о том, что схожесть (в смысле числа общих атрибутов) между объектами из различных классов очень мала.

В целом абсолютно ожидаемыми получились результаты скользящего контроля для второго и третьего алгоритмов и качественно похожи на ранее полученные результаты для набора данных Tic-Tac-Toe.

В таблице 4 также выделяется колонка для метрики Precision. Все значения в ней равны 1. Фактически это означает, что наши алгоритмы не делают ложных предсказаний положительного класса, что является скорее особенностью выбранных данных, чем использованных алгоритмов.

Узорные структуры

Ранее в работе мы всюду использовали шкалирование данных, которое удачно применимо для наших наборов данных, так как мы знаем все возможные значения каждого из атрибутов. Однако данные подход имеет некоторые недостатки, в частности используя его мы значительно увеличиваем количество атрибутов и для используемых данных о разводах вместо работы с 54-мя атрибутами мы создаём 270.

Существует более общий подход, позволяющий избежать шкалирование данных, основанный на узорных структурах. Мы рассмотрим использование узорных структур для набора данных о разводах и будем работать с интервалами. Вводятся следующие операции, являющиеся аналогами рассмотренных ранее:

$$A^{\square} = \prod_{g \in A} \delta(g) \quad \text{for} \quad A \in G$$
$$d^{\square} = \{g \in G | d \sqsubseteq \delta(g)\} \quad \text{for} \quad d \in (D, \sqcap)$$

В приведённых формулах $A^{\square}$ выступает аналогом действия оператора ‘prime’. Для интервалов операция $\sqcap$ и отношение $\sqsubseteq$ определяются следующим образом:

$$[a_1, b_1] \sqcap [a_2, b_2] = [\min(a_1, a_2), \max(b_1, b_2)$$
$$[a_1, b_1] \sqsubseteq [a_2, b_2] \Leftrightarrow [a_1, b_1] \sqcap [a_2, b_2] = [a_1, b_1] \Leftrightarrow [a_1, b_1] \supseteq [a_2, b_2]$$

На основе узорных структур были использованы два очень похожих метода, приведём используемые в них агрегирующие функции:

1) Первый метод основан на определении близости между двумя объектами и использует подход большинства голосов. Введём функцию $dist: D \to R^+$:

$$\forall x \in D \quad \left(x = [[a_1, b_1], [a_2, b_2], \dots, [a_{54}, b_{54}]]\right) dist(x) = \sum_{i=1}^{54} b_i - a_i$$

Тогда для любых двух элементов $a, b \in G$ мы можем рассчитать $dist((a, b)^{\square})$ и на основе этого ввести производить подсчёт голосов в соответствии с формулой[5]:

$$\left|\left\{x \in G^{\pm}: \frac{dist((g, x)^{\square})}{216} < \alpha\right\}\right|$$

[5] В данной формуле деление на 216 (максимально возможное расстояние для наших данных) производится для нормировки расстояния.

Классификация нового объекта g происходит на основе сравнения нормированного числа голосов:

$$\left|\left\{x \in G^{\pm}: \frac{dist((g,x)^{\square})}{216} < \alpha\right\}\right| / |G^{\pm}|$$

2) Второй метод является модификацией предыдущего и накладывает дополнительное ограничение при подсчёте голосов; теперь голоса считаются следующим образом:

$$\left|\left\{x \in G^{\pm}: |(g,x)^{\square\square} \cap G^{\mp}| = 0 \;\&\; \frac{dist((g,x)^{\square})}{216} < \alpha\right\}\right| / |G^{\pm}|$$

Этот метод является своеобразным аналогом алгоритма 2, рассмотренного ранее, и запрещает поглощение пересечений $\delta(g) \sqcap \delta(x)$ описанием любого из элементов противоположного класса.

Для обоих методов, как и ранее, существует параметра α, для которого следует найти оптимальное значение. Была проведена процедура скользящего контроля, результаты которой приведены в таблице 5.

Таблица 5. Результаты кросс-валидации при использовании узорных структур

Parameter	Method	Accuracy	Recall	Precision	F1 score
0,65	Второй	0,918	0,959	1,000	0,979
	Первый	0,906	0,972	1,000	0,986
0,6	Второй	0,935	0,949	1,000	0,974
	Первый	0,929	0,948	1,000	0,973
0,55	Второй	0,959	0,951	1,000	0,975
	Первый	0,959	0,951	1,000	0,975
0,5	Второй	0,977	0,952	1,000	0,976
	Первый	0,977	0,952	1,000	0,976
0,45	Второй	0,977	0,952	1,000	0,976
	Первый	0,977	0,952	1,000	0,976
0,4	Второй	0,977	0,952	1,000	0,976
	Первый	0,977	0,952	1,000	0,976
0,35	Второй	0,977	0,952	1,000	0,976
	Первый	0,977	0,952	1,000	0,976
0,3	Второй	0,977	0,952	1,000	0,976
	Первый	0,977	0,952	1,000	0,976
0,25	Второй	0,971	0,951	1,000	0,975
	Первый	0,971	0,951	1,000	0,975
0,2	Второй	0,929	0,948	1,000	0,973
	Первый	0,929	0,948	1,000	0,973

Точность использованных методов оказалась так же высока, как и точность использованных ранее алгоритмов. Лучшие результаты обоих методов совпадают, но для больших значений параметра $\boldsymbol{\alpha}$ второй метод обладает несколько более высокой точностью. Таким образом, дополнительное ограничение, использованное во втором методе при подсчёте голосов, действительно помогает отсеять неверные голоса. Этот эффект оказался мал, что вероятно обусловлено сильным различием между классами и относительно небольшим числом записей. Можно ожидать, что в общем случае и при работе с большими количествами данных второй метод покажет себя лучше.

Для сравнения результатов были также использованы некоторые популярные методы классификации, а все полученные результаты приведены в таблице 6.

Таблица 6. Итоговая сводка результатов для различных методов классификации

Model	Accuracy	Recall	Precision	F1 score
Random Forest	0,977	0,952	1,000	0,974
Logistic Regression	0,971	0,952	0,989	0,969
K neighbors	0,977	0,952	1,000	0,974
First Alg	0,977	0,952	1,000	0,976
Second Alg	0,965	0,939	1,000	0,968
Third Alg	0,977	0,952	1,000	0,976
Pattern first	0,977	0,952	1,000	0,976
Pattern second	0,977	0,952	1,000	0,976

По таблице 6 мы видим, что использованные в работе методе не уступают по точности предсказания популярным аналогам, также стоит отметить, что почти для всех методов $precision = 1.000$; и как было сказано ранее, это является некой особенностью наших данных, которую можно объяснить и логически.

Выводы

Поставленные задачи были выполнены. Рассмотренные в работе алгоритмы показали высокую точность для двух совершенно разных наборов данных. Отметим, что методы на основе узорных структур выигрывают в скорости у алгоритмов, использующих шкалирование, однако в точности отдать предпочтение какому-либо алгоритму классификации не удалось. Также было выявлено, что данные могут быть очень специфичны, в силу чего значительно могут меняться не только оптимальные значения параметров тех или иных алгоритмов, но и значения метрик качества, соответствующих подобранным оптимальным значениям.

Список литературы

1. Mehdi Kaytoue, Sergei O. Kuznetsov, Amedeo Napoli, Sebastien Duplessis: Mining gene expression data with patter structures in formal concept analysis, published by Elsevier Inc 2010
2. Alexey Masyutin and Yury Kashnitsky: Query-based versus tree-based classification: application to banking data, National Research University HSE Moscow, Russia
3. Sergei O. Kuznetsov: Scalable Knowledge Discovery in Complex Data with Pattern Structures, School of Applied Mathematics and Information Science, National Research University HSE, Moscow, Russia