Московский Государственный Университет имени М.В. Ломоносова
Факультет Вычислительной Математики и Кибернетики
Кафедра Математических Методов Прогнозирования

Отчет по участию в соревновании JRS 2012

Фонарев Александр
317 группа

Апрель 2012

Постановка задачи

Была рассмотрена задача, предлагаемая в рамках конкурса «JRS 2012 Data Mining Competition: Topical Classification of Biomedical Research Papers». Задача является задачей классификации объектов (научных статей) на 83 пересекающихся класса. Каждая объект (статья) задается вектором длины 25640 из целых положительных чисел, каждый элемент в котором характеризует «важность» соответствующего признака (термина) в объекте.

Обучающая выборка состояла из признакового описания 10000 объектов и бинарной информации о принадлежности их каждому из 83 классов. Тестовая выборка также состояла из признакового описания 10000 объектов. Также особенность данных заключалась в том, что матрица данных в 25640-мерном пространстве сильно разрежена.

Функционалом качества была F-мера.

Исследование и алгоритмы

Описание исследования изложено почти в хронологическом порядке

Общие замечания

Для начала были удалены все признаки, которые являются нулевыми на всех объектах из обучения.

Также надо заметить, что почти все мои решения получают вектора ответов для объектов путем бинаризации некоторого промежуточного вещественного вектора ответов для объекта. Если вдруг после бинаризации получается полностью нулевой вектор, то в единицу обращается максимальный элемент вещественного вектора. Далее об этом говорить отдельно не будем.

Кроссвалидация и простейшее решение

Для начала работы с задачей я решил попробовать отправить тривиальное решение на сайт, чтобы удостовериться, что все делаю правильно и примерно оценить точность кроссвалидации.

В качестве тривиального решения просто выбирались наиболее популярные тематики среди всех объектов обучения, и они выдавались в качестве ответа для каждого из тестовых объектов. Такое решение набирало около 0.25 в таблице результатов.

Кроссвалидация показала результаты похожие на предварительные в таблице результатов. Таким образом можно было рассчитывать на более-менее однородные данные в обучении и контроле.

Вероятностные идеи

Самое первое, что пришло в голову – попробовать примернить вероятностный подход к этой задаче. А именно: рассматривалась матрица размера 83 (количество классов) на 25640 (количество признаков), каждый элемент которой был равен вероятности наличия соответствующего класса, если «важность» признака больше некоторого порога c_1 общего для всей матрицы. Далее, такая матрица приводилась к матрице ответа бинаризацией по порогу c_2. Пороги подбирались.

К моему удивлению и сожалению такой алгоритм выдал плохой результат на публичной части контрольной выборки (около 0.20). Конечно, стоило бы попробовать различные нормировки для исходной матрицы данных, и они могли бы улучшить результат. Однако в данном случае мной они опробованы не были.

Простейший метод ближайших соседей

Далее мною были опробованы метрические алгоритмы, основанные на методе ближайших соседей.

Большой недостаток моего дальнейшего исследования состоит в том, что я тестировал все свои алгоритмы на одной и той же случайной подвыборке обучения из 1000 элеметов (т.е. 9000 оставалось для нового обучения). Вообще говоря, надо было обобщить такую валидацию для нескольких случайных подвыборок. Тогда локальные результаты оценки параметров алгоритмов были бы куда точнее. Просто нужно было бы это аккуратно запрограммировать для дальнейшего удобного использования. Однако мною этого не было сделано, в силу отсутствия свободного времени.

Сначала я реализовал обычный метод ближайших соседей. Т.е. я рассматривал N ближайших соседей по некоторой метрике, суммировал их вектора ответов с коэффициентом $1/N$ и бинаризовал по некоторому порогу. Была идея использовать отдельные пороги для каждого из 83 классов, однако она не была реализована.

Варианты метрик и модификации данных

Мною были опробованы несколько метрик: косинусная, евклидова, городских кварталов. Также были опробованы различные варианты модификации исходных данных: возведение в степень (большую и меньшую единицы), прибавление константы к ненулевым признакам, нормировки. Среди нормировок были опробованы нормировки по норме (для соответствующей метрики), по максимуму, по среднему среди ненулевых элеметов. Это все было опробовано как для построчных нормировок, так и для постолбцовых (причем как для обучения и контроля отдельно, так и вместе). Среди всех рассмотренных комбинаций наболее удачными оказалась косинусная мера в совокупности с неизменной исходной матрицей данных (что было достаточно неожиданно, ведь ни одно из преобразований матрицы данных не улучшало результат).

Значение качества более-менее стабилизировалось при N=50. Далее я почти всегда рассматривал N=70. Поскольку разница между размером обучения в валидации (9000 объектов) и фильнальным обучением (10000 объектов) небольшая, то такое значение N принималось и для финального решения. Оптимальным значением порога стало c=0.23. Наибольшим значением качества на фиксированной локальной выборке было 0.4733.

Взвешенный метод ближайшего соседа

Далее я рассмотрел тот же самый метод ближайших соседей, однако суммирование векторов ответов проводил с весами и делил полученную сумму на сумму весов. В качестве веса для конкретного соседа я пробовал рассматривать различные функции, зависящие от порядкового номера соседа и от расстояния до него. Удачную зависимость от порядкового номера найти не удалось. Было перебрано несколько вариантов зависимости веса от расстояния, и наиболее удачным оказался наболее интуитивной ясный вариант: вес обратно пропорциональный расстоянию до соседа. Качество в этом случае доходило до 0.4767 на локальной подвыборке.

Использование наборов ответов

Вообще говоря, скорее всего элементы вектора ответа зависят друг от друга. Таким образом некоторые варианты векторов ответов, например, заведомо невозможны. Чтобы избежать подобных противоречий, попробуем выдавать в качестве ответов лишь те вектора ответов, которые уже встречались в обучающей выборке.

Для этого проделаем следующую процедуру. Каждому уникальному значению вектора ответов поставим в соответствие усредненные вектора объектов, соответствующих этому ответу, в признаковом пространстве. Таким образом мы получили в некотором роде «эталон» для данного вектора ответов.

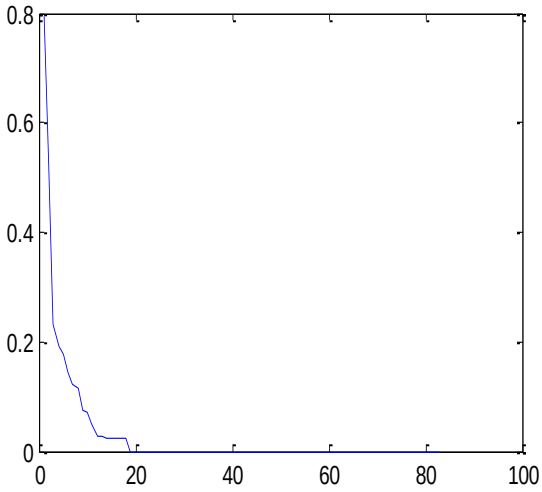
Далее, для каждого искомого объекта попробуем искать ближайший к нему эталон. И выдавать соответствующий эталону вектор ответа. Здесь еще надо учитывать, что некоторые вектора ответов встречаются у многих объектов, а некоторые лишь у одного-двух. Учитывая это, в голову приходит несколько эвристик: например, не рассматривать вектора ответов, которым соответствует меньше с объектов или делить расстояние до эталона на монотонную функцию от количества объектов, соответствующих данному эталону. Однако наилучший результат среди все этих вариант был 0.3898.

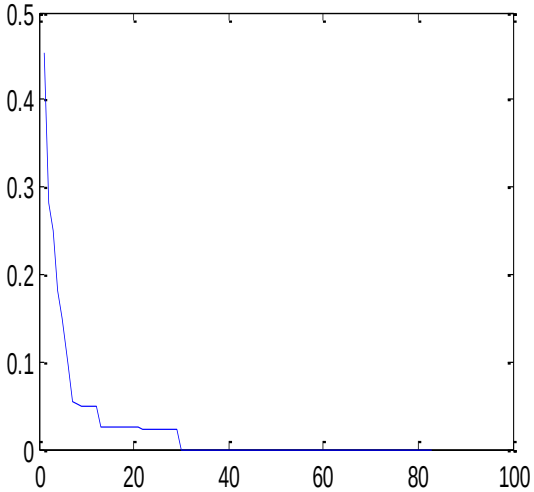
Результат потенциально можно было бы улучшить кластеризацией с автоматическим выбором количества классов внутри каждого уникального вектора ответов. Это действительно оправдано: пусть есть два объекта, принадлежащие, условно, одному классу «хирургия». Причем в первом один важный термин «скальпель», а во втором тоже только один термин «операция на сердце». Тогда эталон, равный среднему между ними будет плохо отображать действительность. Поэтому удачно было бы кластеризовать эти два объекта на два класса и хранить их для этого вектора ответов. Однако я не успел это реализовать.

Подобные вещи можно было бы проделать для каждой из 83 тематик отдельно, но у меня не хватило времени. Интуитивно, это должно было бы хорошо работать, особенно в совокупности с вышеописанной кластеризацией.

Введение нескольких порогов

При просмотре небинаризованных векторов ответов, полученных таким суммированием, можно было заметить, что все объекты довольно четко делятся на две категории: те, у которых есть несколько классов с очень больших значением до бинаризации (близким к 1), и те, у которых их нет. На следующих двух изображениях показаны отсортированные небинаризованные значения вектора ответов для типичных представителей каждой из категорий.





Видно, что для объекта слева алгоритм куда более уверен в классификации, нежели для объекта справа. И интуитивно неправильно было бы использовать одни и те же пороги для объектов первого и второго категорий.

Формализацией принадлежности первой или второй категории стало простое сравнение максимального значения небинаризованного вектора ответов с некоторым порогом c_0. Если максимальный элемент больше c_0, то бинаризация ведется по порогу c_1, иначе по порогу c_2. Оптимальными значениями стали c_0=0.75, c_1=0.33, c_2=0.22. Было достигнуто качество 0.4820 на локальной подвыборке.

Также был испробован «метод крутого склона» для отбора удачных классов. Однако разные его вариации не смогли принести улучшение результата.

Комбинирование способов бинаризации

Далее была опробована комбинация вышеописанных идей: алгоритма с несколькими порогами и алгоритма с использованием готовых наборов ответов. А именно: среди ближайших соседей находим самый популярный вектор ответов. Если он достаточно популярен (встречается чаще некоторого порога), то принимаем его в качестве ответа, иначе используем вышеописанный метод с тремя порогами. Оптимальным значением порога «частоты» для вектора ответов стало значение 0.25. Качество достигло 0.4868 на локальной подвыборке. Вообще говоря, я провел не полный перебор параметров для этого алгоритма. Так что, скорее всего, качество еще было бы улучшаемо до 0.49 на локальной подвыборке.

Однако несмотря на достаточно высокий результат я не стал использовать этот алгоритм на финальном решении, поскольку, как я уже писал выше, параметры подбирались по одной подвыбрке из 1000 элементов. И у меня не хватило времени спроектировать программу для полноценной и удобной кроссвалидации. А поскольку параметров в таком алгоритме много, то он наверняка заметно переобучался на этой подвыборке.

Отбор признаков

Была замечена следующие две вещи. Во-первых, для двух объектов с одинкаковыми векторами ответов расстояние по косинусной мере между ними очень велико. Но это еще интуитивно ясно (вышеописанный пример для двумя разными объектами, принадлежащие одному классу «хирургия»). Однако второе наблюдение было действительно неожиданно. Наоборот, очень

близкие объекты часто имеют абсолютно(!) разную классификацию. Удивительно, что несмотря на такие грубые ошибки, удалось достигать качества заметно выше 0.45.

Эти наблюдения наводят на мысль о том, что можно было бы попробовать увеличить вес некоторых важных признаков. Сделаем это следующим образом: рассмотрим пары очень близких по косинусной мере объектов, которые имеют существенно разные вектора ответов. Были испробованы разные формализации понятия «существенно разные»: например, такими считались объекты без общих классов вообще, объекты у которых общих классов меньше какого-то порога и т.п.. В них сильно различающиеся признаки и были отмечены как важные. Далее были попытки домножить эти важные признаки на константу, большую 1, тем самым повышая важность таких признаков.

Удивительно, но все эти действия не привели к хоть какому-нибудь улучшению результата. Скорее всего это указывает на то, что метод ближайших соседей в этой задаче действительно плохо применим.

Финальное решение

В результате был использован следующий алгоритм. Я рассматривал 70 ближайших соседей по косинусной мере. Суммировал их вектора ответов с весами обратно пропорциональными расстояниям до соответствующего соседа и делил полученный вектор на сумму таких весов. Далее я бинаризовал этот вектор следующим образом: если элемент вектора больше 1.17**(среднее элементов, больших* 0.05*)*, то он преобразовывался в 1, иначе в 0. Если вдруг получался полностью нулевой вектор ответа, то искался максимальный элемент в векторе до бинаризации.

Финальное решение давало результат 0.4840 на локальной подвыборке, что немного меньше чем комбинированный метод, описанные выше. Однако финальное решение содержало достаточно мало параметров, что понижало вероятность переобучения. Именно поэтому в результате был выбран этот вариант.

На публичной части контроля на сайте это решение набрало 0.464. Финальная оценка такого решения – 0.47792 .

Другие идеи

Линейные алгоритмы классификации не показались мне интересными для исследования, даже несмотря на их очевидную актуальность в данной задаче (судя по высоким результатам одногруппников).

Среди возможных идей для реализации, к которым я не приступал, например, алгоритм RSM (Random Subspace Method). Интуитивно он как нельзя кстати подходит к этой задаче.

К сожалению, многие идеи не удалось довести до конца в виду отсутствия свободного времени.

Заключение

Опыт полученный в результате исследования

Если в начале того как я приступил к задаче все появляющиеся идеи и гипотезы требовали достаточных усилий для проверки, то под конец я освоился работать с данными, их визуализацией

и т. п. Также я выработал для себя удобный способ проведения исследования и сохранения отчетности в Matlab по проведенным действиям. И это оказалось очень полезно.

Зачастую приходилось очень аккуратно писать код, чтобы не вылезть за границы оперативной памяти и не начать использовать медленную подкачку.

Также реально была испытана необходимость в мультипоточных вычислениях, что было бы очень удобно на многоядерных компьютерах. Времени разбираться с многопоточностью в Matlab у меня не было, так что я просто запускал несколько копий Matlab на различных кусках данных.

Что могу предоставить

Дополнительно могу предоставить коды вышеописанных (и не только) алгоритмов, отчеты по результатам работы на некоторых этапах, оптимальные значения параметров на некоторых этапах, финальное решение, краткий отчет на английском, отправленный организаторам соревнования.

Работа в команде

К сожалению, мне совсем не удалось поработать в команде. Во-первых, я не принес никакой пользы группе. Возникающие идеи я, конечно, обсуждал с одногруппниками в устной форме, однако я не считал их достаточно существенными для публикации на сайте. Во-вторых, я практически не использовал чужие идеи, а те что использовал, все-равно проверял.

Выражаю благодарность:

- Пете Ромову – за матрицу данных для Matlab и код вывода результата в файл;
- Диме Кондрашкину – за код для подсчета F-меры;
- Ане Потапенко и Мише Гаврикову – за готовность выслушать мои бредовые идеи.

Советы новичкам

- Вести отчетность по всем своим действиям;
- Выделить много времени для участия в соревновании;
- Изначально выстраивать грамотную архитектуру основных функций в программе.