Правительство Российской Федерации
Федеральное государственное автономное образовательное учреждение высшего профессионального образования
Национальный исследовательский университет
«Высшая школа экономики»
Факультет компьютерных наук
Образовательная программа «Науки о данных»

Отчёт по «большому домашнему заданию»

по курсу «Упорядоченные множества в анализе данных»

Выполнила:
Елена Андреева
(м15НоД_ИССА)

Проверил:
Строк Ф. В.

Москва, 2015 г.

Введение

В рамках выполнения домашнего задания были разработаны два алгоритма ленивой классификации, зависящие от некоего параметра «порог». Был реализован поиск оптимального параметра для каждого алгоритма (но не глобального, а применительно к каждому отдельно взятому датасету). Проведена оценка параметров «точность» (accuracy) и времени выполнения программы. Результаты двух алгоритмов сравнивались между собой, а также с результатами наивного байесовского классификатора и классификатора RandomForest.

Алгоритмы имеют многопоточную реализацию. Кроме того, присутствуют некоторые оптимизации для ускорения работы.

Использованные датасеты:

- tic-tac-toe
- kr-vs-kp
- mushrooms (https://archive.ics.uci.edu/ml/datasets/Mushroom)

Программа реализована на языке Python2.7 с использованием пакета scikit-learn.

Оглавление

Описание алгоритма 1

def fca_support_classify(min_intersection)

Для каждого $g_?$:

Заводим две переменные:

positive_support_sum = 0

negative_support_sum = 0

для каждого g_+ находим $g_? \cap g_+$. Если $|g_? \cap g_+| <$ min_intersection, пропускаем это пересечение. Иначе считаем $|g_? \cap g_+|^+$ и прибавляем $\frac{(||g_? \cap g_+|^+ | * |g_? \cap g_+|)}{|G_+|}$ к positive_support_sum.

для каждого g_- находим $g_? \cap g_-$. Если $|g_? \cap g_-| <$ min_intersection, пропускаем это пересечение. Иначе считаем $\frac{(||g_? \cap g_-|^- | * |g_? \cap g_-|)}{|G_-|}$ и прибавляем это к negative_support_sum.

Решение:

$g_?$ принимает тот класс, для которого support_sum больше.

Оптимизация:

очевидно, что пересечения повторяются несколько раз по всему датасету. Поэтому мы храним в словарях пары {intersect : positive_support } и {intersect : negative_support}. Поэтому при попадании на то пересечение, которое уже было, нам нет нужды снова идти по всему +/- контексту, мы получаем значение support за O(1).

Результаты работы:

Интуитивно понятно, что чем больше размер пересечения, тем больше точность. И по идее нужно брать самое большое возможное значение пресечения. Однако результаты работы на датасете mushrooms показали, что это не всегда так.

Таблица 1 - результаты для tic-tac-toe

Порог	Точность
1	65.3%
2	69.3%
3	74.2%
4	77.1%
5	87.9%
6	94.6%
7	99.8%

Таблица 2 - результаты для kr-vs-kp

Порог	Точность
1	54.3%
2	54.3%
3	54.9%
4	55.7%
5	57.1%
6	58.5%
7	60.4%
8	63.5%
9	64.6%

Таблица 3 - результаты для mushrooms

Порог	Точность
1	90.7%
2	91.9%
3	91.8%
4	90.8%
5	91.4%
6	94.8%
7	98.6%
8	99.3%
9	98.8%

Описание алгоритма 2

def fca_stranger_classify():

Для каждого $g_?$:

Заводим две переменные:

positives = 0

negatives = 0

для каждого g_+ находим $g_? \cap g_+$. Если $|g_? \cap g_+|$ < min_intersection, пропускаем это пересечение. Иначе считаем $|g_? \cap g_+|^-$, и если оно равно нулю (т.е. пересечение не вложилось ни в один пример из -контекста), то инкрементируем positives. Иначе не делаем ничего.

для каждого g_- находим $g_? \cap g_-$. Если $|g_? \cap g_-|$ < min_intersection, пропускаем это пересечение. Иначе считаем $|g_? \cap g_-|^+$, и если оно равно нулю (т.е. пересечение не вложилось ни в один пример из +контекста), инкрементируем negatives. Иначе не делаем ничего.

Принятие решения:

Если positives > negatives, класс +.

Иначе класс -.

Однако если для всех $g_?$ получилась ситуация, когда positives+negatives = 0, такие результаты не засчитываются.

Таблица 4 - результаты для tic-tac-toe

Порог	Точность
1	94.2%
2	94.2%
3	94.2%
4	94.1%
5	94.0%
6	92.8%
7	99.1%

Таблица 5 - результаты для kr-vs-kp

Порог	Точность
1	50.91%
2	50.91%
3	50.91%
4	50.91%
5	50.91%
6	50.91%
7	50.91%
8	50.87%
9	50.75%

Таблица 6 - результаты для mushrooms

Порог	Точность
1	99.7%
2	99.7%
3	99.7%
4	99.7%
5	99.7%
6	99.7%
7	99.7%
8	99.7%
9	99.5%

Сводные результаты

Результаты на датасете tic-tac-toe

Название алгоритма	Параметр	Accuracy	Time (s)
FCA-support(initial)	1	65.3%	7.75
FCA-support(best)	7	99.8%	1.12
FCA-stranger(initial)	1	94.2%	3.39
FCA-stranger(best)	7	99.1%	1.05
Naive Bayes	-	68.5%	0.27
Random Forest	n_estimators=100	98.7%	1.37

Результаты на датасете kr-vs-kp

Название алгоритма	Параметр	Accuracy	Time
FCA-support(initial)	1	54.3%	23.98
FCA-support(best)	9	64.6%	9.54
FCA-stranger(initial)	1	50.9%	15.18
FCA-stranger(best)	7	50.9%	11.97
Naive Bayes	-	87.6%	8.25
Random Forest	n_estimators=100	99.2%	4.82

Результаты на датасете mushrooms

Название алгоритма	Параметр	Accuracy	Time
FCA-support(initial)	1	90.7%	137.55
FCA-support(best)	8	99.33%	76.25
FCA-stranger(initial)	1	99.7%	111.35
FCA-stranger(best)	8	99.7%	76.77
Naive Bayes	-	94.1%	5.45
Random Forest	n_estimators=100	100%	11.07

Выводы

Как видно, на первом дасатесе наилучшие результаты как по точности показали FCA-алгоритмы. Второе место по точности занимает Random Forest, но время его работы дольше. Наивный Байес работает быстрее всех, но точность его крайне мала.

Второй датасет требует пересмотра пространства признаков, т.к. точность близка к рандому. Впрочем, первый FCA-алгоритм в лучшем случае показывает 64% точности.

На третьем датасете лучше всего проявил себя Random Forest, но оба fca-алгоритма близки к нему по точности. Наивный Байес снова хуже всех, но не критично. Недостаток FCA-алгоритмов состоит во времени работы, что вполне объяснимо – сложность полного перебора N^2 , и время работы алгоритма по сравнению с супер-оптимизированными RandomForest из scikit-learn растёт очень быстро.

Как видно, для второго алгоритма изменение порога не сильно меняет качество. Поэтому цель второго алгоритма – найти такой порог, при котором классификация будет занимать меньше всего времени с сохранением максимально возможного качества (очевидно, что чем выше порог, тем больше пересечений будет пропущено и тем самым перебор сократится).

Для первого алгоритма изменение порога может драматически изменять качество (датасет ticatactoe: 65.3% => 99.8%). «Отсев» малых пересечений позволяет избавиться от шума, а так же от тех пересечений, которые не несут полезной информации (т.к. встречаются и в том и в другом классах).

Итоги:

1. Реализованы два FCA-алгоритма
2. Реализован поиск оптимального параметра (минимальный размер пересечения)
3. Реализована многопоточная классификация

4. Проведено сравнение точности и времени работы fca-алгоритмов с наивным байесом и random forest
5. Для анализа были использованы три датасета: tic-tac-toe, kr-vs-kp, mushrooms. На первом датасете победа fca, но втором провал fca, на третьем результаты по точности сравнимы с random forest.
6. При выполнении классификации использовалась кросс-валидация.

Пример вывода программы

2015-12-23 21:56:50.034601 Start classifying (support) 93 items

2015-12-23 21:56:51.313019 File: test1.csv thr: 1 TRUE: 61 FALSE: 32

2015-12-23 21:56:52.214630 File: test1.csv thr: 2 TRUE: 65 FALSE: 28

2015-12-23 21:56:52.966435 File: test1.csv thr: 3 TRUE: 70 FALSE: 23

2015-12-23 21:56:53.566612 File: test1.csv thr: 4 TRUE: 67 FALSE: 26

2015-12-23 21:56:53.978694 File: test1.csv thr: 5 TRUE: 83 FALSE: 10

2015-12-23 21:56:54.177247 File: test1.csv thr: 6 TRUE: 85 FALSE: 8

2015-12-23 21:56:54.298104 File: test1.csv thr: 7 TRUE: 93 FALSE: 0

2015-12-23 21:56:54.341487 Start classifying (stranger) 93 items

2015-12-23 21:56:54.709851 File: test1.csv thr: 1 TRUE: 88 FALSE: 5

2015-12-23 21:56:55.160191 File: test1.csv thr: 2 TRUE: 88 FALSE: 5

2015-12-23 21:56:55.619323 File: test1.csv thr: 3 TRUE: 88 FALSE: 5

2015-12-23 21:56:56.007918 File: test1.csv thr: 4 TRUE: 88 FALSE: 5

2015-12-23 21:56:56.317730 File: test1.csv thr: 5 TRUE: 88 FALSE: 5

2015-12-23 21:56:56.544786 File: test1.csv thr: 6 TRUE: 89 FALSE: 4

2015-12-23 21:56:56.671868 File: test1.csv thr: 7 TRUE: 93 FALSE: 0

2015-12-23 21:56:56.733015 Start classifying (support) 87 items

2015-12-23 21:56:57.582634 File: test2.csv thr: 1 TRUE: 51 FALSE: 36

2015-12-23 21:56:58.321753 File: test2.csv thr: 2 TRUE: 54 FALSE: 33

2015-12-23 21:56:59.107342 File: test2.csv thr: 3 TRUE: 58 FALSE: 29

2015-12-23 21:56:59.823688 File: test2.csv thr: 4 TRUE: 64 FALSE: 23

2015-12-23 21:57:00.381600 File: test2.csv thr: 5 TRUE: 73 FALSE: 14

2015-12-23 21:57:00.639483 File: test2.csv thr: 6 TRUE: 82 FALSE: 5

2015-12-23 21:57:00.765098 File: test2.csv thr: 7 TRUE: 87 FALSE: 0

2015-12-23 21:57:00.822031 Start classifying (stranger) 87 items

2015-12-23 21:57:01.281711 File: test2.csv thr: 1 TRUE: 77 FALSE: 10

2015-12-23 21:57:01.723105 File: test2.csv thr: 2 TRUE: 77 FALSE: 10

2015-12-23 21:57:02.149822 File: test2.csv thr: 3 TRUE: 77 FALSE: 10

2015-12-23 21:57:02.436559 File: test2.csv thr: 4 TRUE: 77 FALSE: 10

2015-12-23 21:57:02.664802 File: test2.csv thr: 5 TRUE: 77 FALSE: 10

2015-12-23 21:57:02.811064 File: test2.csv thr: 6 TRUE: 78 FALSE: 9

2015-12-23 21:57:02.923525 File: test2.csv thr: 7 TRUE: 85 FALSE: 2

<……..>

015-12-23 21:57:50.234597 File: test10.csv thr: 4 TRUE: 87 FALSE: 4

2015-12-23 21:57:50.477796 File: test10.csv thr: 5 TRUE: 85 FALSE: 6

2015-12-23 21:57:50.627864 File: test10.csv thr: 6 TRUE: 83 FALSE: 8

2015-12-23 21:57:50.733657 File: test10.csv thr: 7 TRUE: 91 FALSE: 0

=== Summary for dataset tictactoe ===

FCA-support:

Best threshold is 7, accuracy: 0.997752808989, time: 1.32876896858 s

FCA-stranger:

Best threshold is 7, accuracy: 0.990670523718, time: 1.23316478729 s

Accuracies1 = [0, 0.6534736343466454, 0.693233004006709, 0.7422847675329967, 0.7709772560251038, 0.8792441846211041, 0.9460607296541598, 0.997752808988764, 0.0]

Time1 = [0, 8.417440414428711, 7.848337173461914, 7.670274257659912, 6.69479513168335, 4.652583837509155, 2.4015796184539795, 1.3287689685821533, 0.5079076290130615]

Accuracies2 = [0, 0.9415407152956377, 0.9415407152956377, 0.9415407152956377, 0.9405407152956377, 0.9403965766685269, 0.9279708595689382, 0.9906705237178611, 0]

Time2 = [0, 3.9262075424194336, 4.033624887466431, 3.757425308227539, 3.4433517456054688, 2.760694980621338, 1.7388365268707275, 1.2331647872924805, 0]

Naive Bayes:

Accuracy: 0.685456574277, time: 0.320230484009

Random Forest:

Accuracy: 0.986617287178, time: 1.48429656029

Запуск

>> python lazyfca.py

В папке, откуда идёт запуск, должен находиться файл config. В нём перечислены датасеты и их параметры (имя датасета, количество частей для кросс-валидации, название + класса).

Для каждого датасета должна лежать папка с именем, совпадающим с названием датасета. В папке находятся файлы train_#.csv, test_#.csv. Для добавления нового датасета необходимо воспользоваться скриптом split_data.py (автор – Строк Ф. В., я внесла пару изменений), а затем вписать датасет в config.