Ансамблевый метод машинного обучения, основанный на рекомендации классификаторов

Кашницкий Ю. С.

Национальный Исследовательский Университет
Высшая Школа Экономики
Москва
ykashnitsky@hse.ru

Аннотация В статье дается краткое введение в ансамбли классификаторов в машинном обучении и описывается алгоритм, повышающий качество классификации за счет рекомендации классификаторов объектам. Гипотеза, заложенная в основу алгоритма, состоит в том, что классификатор скорее правильно классифицирует объект, если он правильно предсказал метки соседей этого объекта из обучающей выборки. Автор иллюстрирует принцип алгоритма на простом примере и описывает тестирование на реальных данных. .

Введение

Тема ансамблевых методов хорошо изучена в машинном обучении. Подобные системы в разных статьях появляются под разными именами: смесь экспертов (mixture of experts), committee machines, ансамбли классификаторов (classifier ensembles), смесь классификаторов (classifier fusion) и прочими.

В основе всех подобных систем лежит идея обучения нескольких (базовых) классификаторов на одной и той же обучающей выборке и комбинации их предсказаний для новых тестируемых объектов [1]. Математическим обоснованием этой идеи служит теорема Кондорсье о жюри присяжных (the Condorcet's jury theorem), датируемая еще XVIII веком. Согласной этой теореме, если каждый присяжный в среднем вероятнее всего судит о виновности обвиняемого верно, нежели ошибочно, то вердикт большинства заслуживает доверия. В пределе вероятность того, что большинство присяжных проголосует правильно, стремится к 1 по мере увеличения числа присяжных. Таким же образом, имея несколько слабых классификаторов (под слабостью классификатора имеется в виду, что его ошибка на обучающей выборке менее 50%, но более 0%), можно объединить их предсказания и достичь более высокой точности классификации объектов из тестовой выборки.

Среди наиболее известных ансамблевых методов классификации выделим бэггинг (bagging) [2], бустинг (boosting) [3], случайные леса (random forests) [4] и стекинг (stacked generalization, stacking) [5].

В данной статье автор представляет еще один алгоритм подобного рода, основанный на рекомендации классификаторов — RCE (Recommender-based Classifier Ensemble). В основе алгоритма лежит предположение, что классификатор скорее всего правильно классифицирует объект, если он правильно предсказал метки соседей этого объекта из обучающей выборки.

1 Ансамблевые методы классификации

В данной главе мы рассматриваем три известных типа ансамблевых алгоритмов классификации.

Бэггинг Исследователи, занимающиеся статистикой, уже давно используют метод, называемый "bootstrap sampling", что условно может быть переведено на русский как "вариация загрузочной выборки". Одно из воплощений этой идеи в машинном обучении - "bootstrap aggregating", или сокращенно "bagging", то есть объединение результатов при различных загрузках.
Идея бэггинга состоит в том, что при отсутствии большой обучающей выборки можно создавать много случайных выборок из исходной простым выбором с замещением. Хотя элементы в выборках могут пересекаться или дублироваться, на практике все же результаты объединения по многим выборкам оказываются точнее, чем только по одной начальной. Метод так называется, поскольку он объединяет результаты предсказания различных классификаторов, обученных на случайных подмножествах.
Бэггинг оказывается полезен только в случае разных классификаторов и нестабильности, когда малые изменения в начальной выборке приводят к существенным изменениям классификации [2].

Бустинг В работе 1984 года [6] были представлены теоретические основы PAC-модели вероятно почти корректного обучения (Probably Approximately Correct), при котором рассматривалась возможность улучшить алгоритм классификации с помощью нескольких слабых классификаторов.
В 1989 году Шапир (Shapire) первым придумал такой алгоритм с полиномиальной сложностью и опубликовал в статье с ярким названием "Сила слабого обучения" [7], а через год Фрейнд (Freund) разработал более эффективную реализацию [3], которая стала основой алгоритма AdaBoost, представленного в 1995 году Шапиром и Фрейндом уже вместе.
Бустинг (boosting, улучшение) — это процедура последовательного построения композиции алгоритмов машинного обучения, когда каждый следующий алгоритм стремится компенсировать недостатки композиции всех предыдущих алгоритмов. Бустинг представляет собой жадный алгоритм построения композиции алгоритмов. Изначально понятие бустинга возникло в работах по вероятно почти корректному обучению в связи с вопросом: возможно ли, имея множество плохих (незначительно отличающихся от случайных) алгоритмов обучения, получить хороший.

В течение последних 10 лет бустинг остается одним из наиболее популярных методов машинного обучения наряду с нейронными сетями и машинами опорных векторов. Основные причины — простота, универсальность, гибкость (возможность построения различных модификаций), и, главное, высокая обобщающая способность.

Бустинг над решающими деревьями считается одним из наиболее эффективных методов с точки зрения качества классификации. Во многих экспериментах наблюдалось практически неограниченное уменьшение частоты ошибок на независимой тестовой выборке по мере наращивания композиции. Более того, качество на тестовой выборке часто продолжало улучшаться даже после достижения безошибочного распознавания всей обучающей выборки. Это перевернуло существовавшие долгое время представления о том, что для повышения обобщающей способности необходимо ограничивать сложность алгоритмов. На примере бустинга стало понятно, что хорошим качеством могут обладать сколь угодно сложные композиции, если их правильно настраивать.

Впоследствии феномен бустинга получил теоретическое обоснование. Оказалось, что взвешенное голосование не увеличивает эффективную сложность алгоритма, а лишь сглаживает ответы базовых алгоритмов. Количественные оценки обобщающей способности бустинга формулируются в терминах отступа. Эффективность бустинга объясняется тем, что по мере добавления базовых алгоритмов увеличиваются отступы обучающих объектов. Причем бустинг продолжает раздвигать классы даже после достижения безошибочной классификации обучающей выборки.

К сожалению, теоретические оценки обобщающей способности дают лишь качественное обоснование феномену бустинга. Хотя они существенно точнее более общих оценок Вапника-Червоненкиса, все же они сильно завышены, и требуемая длина обучающей выборки оценивается величиной порядка $10^4 \dots 10^6$. Более основательные эксперименты показали, что иногда бустинг все же переобучается.[1]

Стекинг Стековое обобщение (stacked generalization), или просто стекинг (stacking) — еще один способ объединения классификаторов, вводящий понятие мета-алгоритма обучения. В отличие от бэггинга и бустинга, при стекинге используются классификаторы разной природы. Идея стекинга такова [5]:

1. разбить обучающую выборку на два непересекающихся подмножества;
2. обучить несколько базовых классификаторов на первом подмножестве;
3. тестировать базовые классификаторы на втором подмножестве;
4. используя предсказания из предыдущего пункта как входные данные, а истинные классы объектов как выход, обучить мета-алгоритм.

[1] http://www.machinelearning.ru/wiki/index.php?title=Бустинг

2 Основные определения Анализа Формальных Понятий

Формальный контекст — это тройка $K = (G, M, I)$, где G — множество объектов, M — признаков, а бинарное отношение $I \subseteq G \times M$ определяет, какой объект каким признаком обладает. Предикат gIm означает, что объект g имеет признак m. Для подмножеств множеств объектов и признаков $A \subseteq G$ и $B \subseteq M$ *операторы Галуа* определяются следующим образом:
$A' = \{m \in M \mid gIm$ для всех $g \in A\}$,
$B' = \{g \in G \mid gIm$ для всех $m \in B\}$.
Оператор $''$ (применение оператора $'$ дважды) называется *оператором замыкания*. Множество объектов $A \subseteq G$, таких что $A'' = A$, называется *замкнутым*.
Пара (A, B), такая что $A \subset G, B \subset M, A' = B$ и $B' = A$, называется *формальным понятием* контекста K. Множества A и B замкнуты и называются *объемом* и *содержанием* формального понятия (A, B) соответственно. Для множества объектов A множество их общих признаков A' определяет сходство объектов множества A, а замкнутое множество A'' есть множество схожих объектов (с общими признаками из A').
Общее число формальных понятий контекста $K = (G, M, I)$ довольно существенно: в худшем случае $2^{min\{|G|,|M|\}}$. Существуют способы уменьшения этого количества, например, отбор понятий по стабильности или индексу мощности объема [9].
В контексте (G, M, I) понятие $X = (A, B)$ *более и столь же общее*, как понятие $Y = (C, D)$ (или $X \leq Y$), если $A \subseteq C$ или, эквивалентно, $D \subseteq B$. Для двух понятий X и Y, таких что $X \leq Y$ и не существует понятия Z: $Z \neq X, Z \neq Y, X \leq Z \leq Y$, понятие X называется *соседом снизу* понятия Y, а Y, соответственно - *соседом сверху* понятия X. Это отношение обозначается как $X \prec Y$. Формальные понятия, упорядоченные этим отношением, образуют *полную решетку понятий*, которая может быть представлена *диаграммой Хассе* [10]. Несколько алгоритмов построения формальных понятий и их решеток (включая *Close by One*) рассмотрены также в [10].
Некоторые примеры формальных понятий и решеток, а также примеры их применения в приложениях анализа данных можно найти в [8] и [11]. При описании алгоритма RCE на искусственном наборе данных также демонстрируется применение аппарата АФП в реальной задаче.
Однако, в некоторых приложениях нет необходимости находить все формальные понятия или строить всю решетку понятий. Решетки понятий, которые могут включать только частые формальные понятия, называются айсбергами (*iceberg lattices*). Было показано, что они могут служить компактным представлением ассоциативных правил и частых множеств признаков [11].
Мы упростили алгоритм *Close by One*, так что он находит только самое верхнее формальное понятие и его ближайших соседей снизу. Описание оригинального алгоритма и его упрощенной версии остается за рамками данной статьи.

3 RCE

В этой главе мы описываем ансамблевый метод машинного обучения, основанный на рекомендации классификаторов — Recommender-based Classifier Ensemble (RCE). Псевдокод алгоритма представлен в листинге 1.

Вход алгоритма:

1. X_{train}, y_{train} — обучающая выборка, X_{test} — тестовая выборка;
2. $C = \{cl_1, cl_2, ..., cl_K\}$ — множество K базовых классификаторов;
3. $sim(x_1, x_2)$ — функция сходства объектов, определенная в пространстве признаков. Это может быть расстояние Минковского (включая Хэмминга и Евклидово), расстояние, взвешенное важностью признаков, и другие;
4. k, n_fold - параметры, смысл которых проясняется ниже;
5. $topCbO(context)$ - функция для нахождения верхнего формального понятия формального контекста и его ближайших соседей снизу.

Алгоритм состоит из следующих шагов:

1. Скользящий контроль на обучающей выборке. Все K классификаторов обучаются на $(n\_folds - 1)$ частях обучающей выборки X_{train}. Затем формируется классификационная таблица (контекст), в которой крест для объекта i и классификатора cl_j ставится, если cl_j правильно предсказал метку объекта i, обучившись на $(n\_folds - 1)$ частях обучающей выборки, не включающих объект i;
2. Запуск базовых классификаторов. Все K классификаторов обучаются на всей выборке X_{train}. Затем формируется таблица (контекст) предсказаний для тестовой выборки, в которой позиция (i, j) содержит метку, предсказанную классификатором cl_j для объекта i;
3. Нахождение верхних формальных понятий классификационного контекста. Запускается алгоритм *topCbO* для нахождения верхнего формального понятия классификационного контекста и его ближайших соседей снизу. Объемы этих понятий имеют максимальное число объектов, а содержания — минимальное число классификаторов;
4. Нахождение соседей объектов из тестовой выборки X_{test}. Эти объекты обрабатываются один за другим. Для каждого объекта из тестовой выборки X_{test} находятся k его ближайших соседей из обучающей выборки X_{train} согласно определенной метрике $sim(x_1, x_2)$. Обозначим множество этих соседей просто через $Neighbors$. Затем находится формальное понятие классификационного контекста, чей объем имеет максимальное пересечение с множеством $Neighbors$. Если содержание самого верхнего понятия пустое (то есть ни один классификатор не предсказал правильно метки всех без исключения объектов из X_{train}, что чаще всего и имеет место для реальных данных), то верхнее понятие $\{G, \emptyset\}$ игнорируется. Таким образом выбирается формальное понятие классификационного контекста, содержание которого — множество классификаторов, которое мы назовем C_{sel};

Таблица 1. Пример набора данных из 10 объектов с 4 бинарными признаками и бинарным целевым классом

G/M	m_1	m_2	m_3	m_4	Label
1	×	×		×	1
2	×			×	1
3		×	×		0
4	×		×	×	1
5	×	×	×		1
6		×	×	×	0
7	×	×	×		1
8			×	×	0
9	×	×	×	×	?
10		×		×	?

Таблица 2. Классификационный контекст

G/C	cl_1	cl_2	cl_3	cl_4
1	×		×	×
2		×	×	
3	×			×
4		×	×	
5	×	×		
6	×	×		×
7		×		×
8		×	×	×

5. Классификация. Если C_{sel} состоит всего из одного классификатора, то для текущего объекта из X_{test} предсказывается класс, совпадающий с предсказанием этого классификатора. Если выбраны несколько классификаторов ($|C_{sel}| > 1$), то конечное предсказание определяется с помощью голосования большинством среди этих классификаторов.

4 Искусственный пример

Принцип алгоритма мы продемонстрируем на искусственном наборе данных, представленном таблицей 1. Рассматривается задача бинарной классификации с 8 объектами, составляющими обучающую выборку, и 2 — тестовую. Каждый объект имеет четыре бинарных признака и бинарный целевой класс. Мы обучаем 4 классификатора на данной обучающей выборке и предсказываем целевой класс для объектов 9 и 10.

Согласно традиции АФП, обозначим через $G = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$ — множество объектов, $G_{test} = \{9, 10\}$ — тестовую выборку, $G_{train} = G \backslash G_{test}$ — обучающую выборку, $M = \{m_1, m_2, m_3, m_4\}$ — множество признаков, $C = \{cl_1, cl_2, cl_3, cl_4\}$ — множество классификаторов.

Здесь мы применяем скользящий контроль “Оставь одного” (leave-one-out cross-validation) для 4 классификаторов. Заполняется таблица 2, в которой крест для объекта i и классификатора cl_j ставится, если cl_j правильно предсказал метку объекта i в процессе скользящего контроля. Например, крест в таблице для объекта 3 и классификатора cl_4 означает, что cl_4 правильно классифицировал объект 3 после обучения на остальных объектах из обучающей выборки, то есть на множестве $G_{train}\backslash\{3\} = \{1, 2, 4, 5, 6, 7, 8\}$
Таблицу 2 можно рассматривать как формальный контекст с объектами G и признаками C (то есть теперь классификаторы выступают в роли признаков). Назовем этот контекст классификационным. Решетка понятий этого контекста представлена на рис. 1.

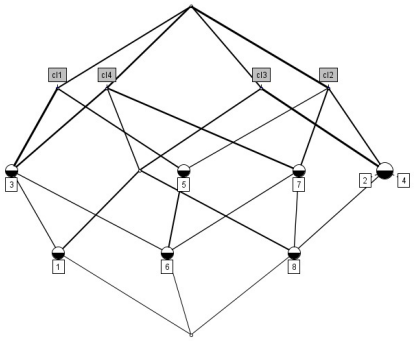

Рис. 1. Решетка понятий классификационного контекста

Как было упомянуто, число формальных понятий формального контекста $K = (G, M, I)$ может быть велико — в худшем случае экспоненциально по отношению к $min\{|G|, |M|\}$. Но в данном случае решетка понятий представлена просто как иллюстрация. Для алгоритма RCE нужна не вся решетка, а только ее верхнее понятие и его ближайшие соседи снизу (рис. 2).
Вот эти верхние понятия классификационного контекста:

- $\{G, \emptyset\}$;
- $\{[1, 3, 5, 6], cl_1\}$;
- $\{[2, 4, 5, 6, 7, 8], cl_2\}$;
- $\{[1, 2, 4, 8], cl_3\}$;
- $\{[1, 3, 6, 7, 8], cl_4\}$.

Для классификации объектов из G_{test} сначала найдем k их ближайших соседей из G_{train} согласно некоторой метрике сходства объектов. Здесь мы берем $k = 3$ и используем метрику Хэмминга. При таких условиях 3 ближайших соседа объекта 9 — объекты 4, 5 и 7, а 10 – 1, 6 и 8.

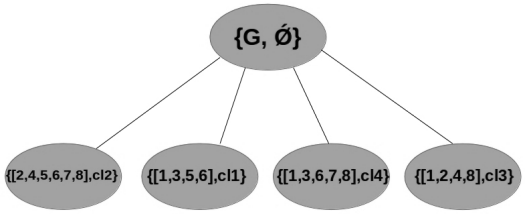

Рис. 2. Верхние понятия классификационного контекста

Далее находим максимальные пересечения множеств этих соседей $Neighb_9 = \{4, 5, 7\}$ и $Nieghb_{10} = \{1, 6, 8\}$ с объемами формальных понятий классификационного контекста, представленными выше (игнорируя $\{G, \emptyset\}$). Содержания (то есть классификаторы) соответствующих понятий — это рекомендуемые классификаторы для рассматриваемых объектов из G_{test}. Процесс рекомендации классификаторов компактно представлен в таблице 3 (под "понятием класс. контекста" имеется в виду формальное понятие классификационного контекста, чей объем имеет максимальное пересечение со множеством соседей).

Таблица 3. Рекомендация классификаторов для объектов из тестовой выборки G_{test}

G_{test}	9	10
1 ближ. сосед	4	1
2 ближ. сосед	5	6
3 ближ. сосед	7	8
Соседи	$\{4, 5, 7\}$	$\{1, 6, 8\}$
Понятие класс. контекста	$\{[2, 4, 5, 6, 7, 8], cl_2\}$	$\{[1, 3, 6, 7, 8], cl_4\}$
Рекомендуемый классификатор	cl_2	cl_4

Наконец, RCE предсказывает для объектов 9 и 10 те же метки, что и классификаторы cl_2 и cl_4 соответственно.
Напоследок сделаем несколько замечаний:

1. Мы бы не проигнорировали верхнее понятие классификационного контекста с объемом G, если бы его содержание не было пусто. То есть если бы верхнее понятие имело вид $\{G, cl_j\}$, это означало бы, что cl_j правильно классифицировал все объекты из обучающей выборки в процессе скользящего контроля, и тогда мы рекомендовали бы его как лучший и для объектов их тестовой выборки;
2. Также может возникнуть ситуация, что два и более классификатора проявили себя одинаково хорошо на скользящем контроле

при классификации объектов из G_{train}. Это бы означало, что соответствующие колонки в классификационной таблице одинаковы, а следовательно, содержание некоторого формального понятия классификационного контекста содержит более одного классификатора. В таком случае нет никаких причин предпочитать один классификатор другому при обработке объектов из тестовой выборки, а потому метка для соответствующего объекта определится в результате голосования большинством среди этих одинаково себя проявивших классификаторов;

3. Здесь мы рассматривали входной набор данных с бинарными признаками и целевым классом, однако идея RCE распространяется и на случаи числовых признаков и задачи с несколькими целевыми классами.

5 Эксперименты

Описанный выше алгоритм был реализован на языке Python версии 2.7.3 и запускался на 2-процессорной машине (Core i3-370M, 2.4 Ггц) с 3.87 GB ОЗУ.

Мы использовали 3 набора данных из репозитория UCI — `mushrooms`, `ionosphere` и `digits`.[2]. Каждый из наборов данных был поделен на обучающую и тестовую выборки в пропорции 70:30.

Таблица 4. Точность классификации 6 алгоритмов на 3 наборах данных UCI

Алгоритм, набор данных	mushrooms	ionosphere	digits
SVM с RBF-ядром	0.998 (C=1, $\gamma = 0.02$)	0.931 (C=10, $\gamma = 0.01$)	0.865 (C=10^4, $\gamma = 0.02$)
Логистич. регрессия	0.959 (C=1)	0.832 (C=10)	0.854 (C=10^3
kNN (евклид.)	0.989 (k=5)	0.666 (k=3)	0.811 (k=3)
RCE (k=2, n_folds=4)	0.998	0.946	0.917
Bagging SVM 50 выборок	0.999 (C=1, $\gamma = 0.02$)	0.999 (C=10, $\gamma = 0.01$)	0.878 (C=10^4, $\gamma = 0.02$)
AdaBoost on decision stumps, 50 итераций	0.873	0.997	0.889

[2] http://archive.ics.uci.edu/ml/datasets

На этих данных были обучены 3 классификатора из библиотеки `SCIKIT-LEARN` (написана на Python [12]), которые также служили базовыми алгоритмами для RCE. Это были машина опорных векторов (Support Vector Machine, SVM) с гауссовым ядром (`svm.SVC()` в `Scikit`), логистическая регрессия
(`sklearn.linear_model.LogisticRegression()`) и k ближайших соседей
(`sklearn.neighbors.classification.`
`KNeighborsClassifier()`).
Точность классификации для каждого алгоритма и набора данных предствлена в таблице 4. Более того, для сравнения представлены результаты для `Scikit`-версии бэггинга с SVM в качестве базового классификатора и AdaBoost на основе решающих деревьев глубины 1 (decision stumps).
Как видно, во всех 3 случаях RCE превзошел свои базовые классификаторы и оказался лучше бэггинга и бустинга только в случае задачи классификации с несколькими целевыми классами.

6 Заключение

В данной статье мы описали идею, лежащую в основе ансамблевых методов классификации в машинном обучении, вкратце описали бэггинг, бустинг и стекинг. Затем мы представили подобную систему, которая показала более высокую точность классификации, чем ее базовые классификаторы, а также две конкретные реализации бэггинга и AdaBoost в задаче классификации с несколькими целевыми классами.
Дальнейшая работа над алгоритмом планируется по следующим направлениям:

- изучение влияния различных метрик сходства объектов (как, например, основанные на важности признаков или приросте информации) на производительность алгоритма;
- экспериментирование над различными типами базовых для RCE классификаторов;
- исследование условий, при которых предпочтительно использование RCE. В частности, когда он превосходит бэггинг и бустинг;
- работа над временной сложностью алгоритма;
- исследование проблемы переобучения для данного алгоритма.

Список литературы

1. Izenman, A.J.: Modern Multivariate Statistical Techniques, Springer Texts in Statistics. Springer Science+Business Media New York (2013)
2. Breiman, L.: Bagging predictors. Machine Learning, 24, 123–140. (1996)
3. Freund, Y., Schapire, R. E.: A Short Introduction to Boosting. Journal of Japanese Society for Artificial Intelligence, 14 (5), 771-780 (1999)
4. Breiman, L.: Random Forests. Machine Learning. 45, 1 , 5–32 (2001)
5. Wolpert, D. H.: Stacked Generalization. Neural Networks, 5, 241–259 (1992)
6. Valiant, L. G.: A theory of the learnable. Communications of the ACM, 27(11), 1134–1142 (1984)
7. Schapire, R.: The strength of weak learnability. Machine Learning, 5(2), 197–227 (1990)
8. Ganter, B., Wille, R.: Formal concept analysis: Mathematical foundations. Springer, Berlin (1999)
9. Kuznetsov, S. O.: On stability of a formal concept. Annals of Mathematics and Artificial Intelligence. 49, 101—115 (2007)
10. Kuznetsov, S., Obiedkov, S.: Comparing the performance of algorithms for generating concept lattices. Journal of Experimental & Theoretical Artificial Intelligence, 14(2-3), 189–216 (2002)
11. Stumme, G., Taouil, R., Bastide, Y., Lakhal, L.: Conceptual Clustering with Iceberg Concept Lattices. Proceedings of GI-Fachgruppentreffen Maschinelles Lernen, University of Dortmund (2001)
12. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E.: Scikit-learn: Machine Learning in Python, Journal of Machine Learning Research, 12, 2825-2830 (2011)

Algorithm 1 Ансамблевый метод машинного обучения, основанный на рекомендации классификаторов

Вход: $X_{train}, y_{train}, X_{test}$ — обучающая и тестовая выборки,
$C = \{cl_1, cl_2, ..., cl_K\}$ — базовые классификаторы,
$topCbO(context, n)$ — функция для нахождения верхнего формального понятия формального контекста и его ближайших соседей снизу,
$sim(x_1, x_2)$ — метрика сходства объектов,
k — параметр (число ближайших соседей),
n_fold — параметр (число разбиений обучающей выборки при скользящем контроле)
Выход: y_{test} — предсказанные метки для объектов из X_{test}

$train\_class\_context = [\,][\,]$ — двумерный массив
$test\_class\_context = [\,][\,]$ — двумерный массив
for $i \in 0 \dots len(X_{train}) - 1$ **do**
  for $cl \in 0 \dots len(C) - 1$ **do**
    классификатор cl обучается на $(n\_fold - 1)$ частях обучающей выборки, не включающих $X_{train}[i]$
    $pred =$ предсказанная классификатором cl метка для $X_{train}[i]$
    $train\_class\_context[i][cl] = (pred == y_{train}[i])$
  end for
end for
for $cl \in 0 \dots len(C) - 1$ **do**
  классификатор cl обучается на всей обучающей выборке X_{train}
  $pred =$ предсказанные классификатором cl метки для объектов из X_{test}
  $test\_class\_context[:][cl] = pred$
end for
$top\_concepts = topCbO(class\_context)$
for $i \in 0 \dots len(X_{test}) - 1$ **do**
  $Neighbors = k$ ближайших соседей объекта $X_{test}[i]$ из X_{train} согласно метрике $sim(x_1, x_2)$
  $concept = argmax(c.extent \cap Neighbors), c \in top\_concepts$
  $C_{sel} = concept.intent$
  $labels =$ предсказания для $X_{test}[i]$, сделанные классификаторами из C_{sel}
  $y_{test}[i] = argmax(count\_freq(labels))$
end for