Мосин Василий

В данной работе рассматриваются два алгоритма:

1. algorithm1 – простой и быстрый алгоритм. Описание классифицируемого объекта из тестовой выборки сравнивается по очереди с описанием каждого примера из положительного контекста. Если совпадение преодолевает порог C, то данный пример из положительного контекста “голосует” за классифицируемый объект. Такие же сравнения проводятся с примерами из отрицательного контекста. Далее суммы голосов нормируются количеством примеров в «+/-» контекстах соответственно. Классификация происходит уже по сравнению нормированных голосов.

$$if\ \frac{\left|g'_\tau\ \cap g'_{+/-}\right|}{|g'_\tau|} > C, then\ votes(+/-)\ +=1; \frac{votes(+)}{|G_+|} <> \frac{votes(-)}{|G_-|}$$

2. algorithm2 – более замысловатый и более медленный алгоритм. Формируются пересечения описания классифицируемого объекта с описаниями примеров из «+» контекста. Для каждого такого пересечения проверяется количество вложений данного пересечения в описания примеров из «-» контекста. Если это количество не преодолевает порог C, то данный пример из положительного контекста “голосует” за классифицируемый объект. Аналогично, для примеров из «-» контекста. Далее суммы голосов нормируются, как и в случае algorithm1. Классификация происходит уже по сравнению нормированных голосов.

$$if \frac{\left|\left(g'_\tau\ \cap g'_{+/-}\right) \subseteq g'_{-/+}\right|}{|G_{-/+}|} <\ C, then\ votes(+/-)+=1;\ \frac{votes(+)}{|G_+|} <> \frac{votes(-)}{|G_-|}$$

Для оценки работы алгоритмов вычислялись такие метрики качества классификации как accuracy, precision, recall и F_measure. При прогоне использовался метод кросс-валидации с параметром k=3. Использовались 10 файлов train#.csv массива Tic-Tac-Toe. Ниже приведена сводная таблица усредненных метрик по 10 запускам.

	accuracy	precision	recall	F_measure
Algorithm1(C=0.7)	0.988888888889	1.0	0.98275862069	0.991304347826
Algorithm1(C=0.6)	0.833333333333	0.921568627451	0.810344827586	0.862385321101
Algorithm1(C=0.8)	NA	NA	NA	NA
Algorithm2(C=0)	0.977272727273	0.966666666667	1.0	0.983050847458
Algorithm2(C=0.01)	0.852272727273	0.961538461538	0.862068965517	0.909090909091

В процессе анализа работы алгоритмов выяснилось, что наиболее оптимальным пороговым значением для algorithm1 является C=0.7 (при C=0.8 algorithm1 перестает работать, выдавая contradictory на каждой проверке). Для algorithm2 выяснилось, что лучше установить порог нулевым, то есть вообще запретить вложения пересечения описания объекта с описанием примера в описания примеров другого класса. Допуская даже 1% таких вложений, метрики качества ухудшаются.

В целом же, более простой алгоритм оказался лучше более сложного. При правильном подборе оптимального порогового значения метрики качества классификации (кроме recall) и время работы algorithm1 оказались лучше, чем у algorithm2. Хотя при анализе выяснилось, что при определенных пороговых значениях algorithm1 теряет устойчивость.