1 План

- Описать интерфейсы для ленивой классификации: контекст, признак, объект
- Написать реализации интерфейсов и использовать их для классификации объектов с числовыми признаками
- Для какой-то конкретной модели подобрать оптимальный параметр по результатам скользящей валидации
- Сравнить полученный результат с результатом, полученным с использованием какой-нибудь стандартной библиотеки

2 Выбор модели

В коде на гитхабе есть только одна реализация контекста и две реализации объекта, поэтому выбирать тут мало из чего. Реализации объекта отличаются только тем, могут ли они быть признаны похожими, если по какому-то из признаков они не схожи. Далее есть много классов признаков. Все они работают по принципу узорных структур, но есть BinaryDiscreteParam, который эмулирует обычный бинарный признак. Поскольку данные числовые, был выбран класс RealParam, который работает по правилу пересечения вещественных чисел, описанному на парах. И, самое интересное, классификаторы. Сначала я хотел использовать в классификаторе каким-то образом информацию о том, насколько "узкое"пересечение у оцениваемого объекта и у объекта из + или - контекста. Для этого у признаков есть weight(). Но оказалось, что это не улучшает результат.

- WeightedGeneratorClassifier: если у пересечения тестируемого объекта и объекта из $\pm$ контекста нет контрпримера, то сумму $\pm$ поддержку пересечения в этом контексте, умноженную на вес признаков. После прохода по обоим контекстам сумма сравнивается с порогом и если сумма $\geq$ порога, то 1, иначе 0. Порог шевелится, чтобы максимизировать accuracy.
- GeneratorClassifier: то же самое, только не считается вес признаков. Честно говоря, вес признаков сейчас сырой хотя бы потому что нет никакой нормировки на разброс признаков и его единицы измерения. Это глупо, поэтому все, что использует вес, неадекватно работает.
- EdgedGeneratorClassifier: считается support пересечения проверяемого объекта и тренировочного и если он $>$ параметра, то один голос за этот контекст. Голосов > 0, тогда $+$. Иначе -.
- FullClassifier: ищется поддержка пересечения тестируемого объекта и какого-то тренировочного в положительном контексте и вычитается поддержка в отрицательном контексте. Сумма по всем тренировочным объектам сравнивается с порогом, который является параметром.

- QuantileClassifier: ради интереса решил отбросить самые "похожие"и самые "непохожие"судя по весу признаков объекты и считать только центральные объекты. +1 за отобранный объект из + контекста и -1 за отобранный из минус контекста.

Размер тестовых контекстов 100 объектов в тренировочном и 100 в тестовом. Без скользящей валидации скрипт классифицирует 100 объектов порядка 20 секунд. Плюс надо подобрать параметр. В итоге, я выбрал одинк классификвтор - самый адекватный и часто используемый - EdgedGeneratorClassifier. Прогон порогов от $\frac{3}{150}$ до $\frac{19}{150}$ выбрал порог $\frac{14}{150}$ с accuracy $= 0.67$. Стоит заметить, что тривиальный классификатор, который всем присваивает 1 класс имел бы accuracy=0.644, то есть чуть хуже.

3 Сравнение со стандартной библиотекой

Была использована библиотека sklearn и выбран классификатор sklearn.tree.DecisionTreeClassifier. Там много параметров, я пошевелил только $min\_weight\_fraction\_leaf$. Максимизация accuracy привела к значению $min\_weight\_fraction\_leaf = 0.16$ с $accuracy = 0.73$.

Настроенный за 10 минут метод классификации с помощью деревьев решений оказался и быстрее, и точнее, чем то, что написано мной.

4 Выводы

- Интерфейсы описаны и созданы их реализации
- Для EdgedGeneratorClassifier с помощью скользящей валидации был подобран параметр порога $\frac{14}{150}$ при accuracy=0.67.
- Результат еле-еле превосходит результат тривального классификтаора, при этом работает долго и проигрывает деревьям решений даже при том, что данные не сбалансированы. Но не все так плохо: есть широкие простор для экспериментов и с признаками, и с классификаторами. Так, например, для придания более адекватного веса пересечениям признаков имеет смысл сделать препроцессинг и найти, скажем, выборочную дисперсию внутри признака, чтобы использовать ее корень как нормаировку. Если говорить о узорных структурах с вещественными признаками, то можно немного расширить интервал, в котором считать объект обладающи пересечением признаков.