Санкт-Петербургский Государственный Университет
Математико-механический факультет

Курсовая работа

Сравнение алгоритмов классификации на предмет устойчивости к зашумлению входных данных

Выполнил:
студент 243 группы
Владимир Назаренко

Научный руководитель:
профессор Б.А. Новиков

Санкт-Петербург
2014

1 Введение

Машинное обучение имеет множество применений в самых разных областях, например, в фильтрации спама [1], или в построении некоторых заключений на основе показаний датчиков [2]. Одной из возможностей также является анализ изображений.

Одним из методов машинного обучения является обучение по прецедентам. Но качество работы алгорима, использующего этот метод, напрямую зависит от корректности и количества прецедентов (training set). К сожалению, информация о прецедентах зачастую содержит ошибки. Этой теме и посвящена данная работа – исследуем устойчивость популярных алгоритмов машинного обучения по прецедентам к шумам в данных.

2 Обзор предметной области

2.1 Основные понятия

С основной терминологией можно ознакомиться в статье Domingos [3], а описания рассмотренных алгоритмов содержит работа [4], но для удобства дадим определения понятиям, используемым далее в работе.

Предположим, у нас есть множество объектов $x_i \in X$, каждый из которых описывается своим *вектором признаков* $x_i = (x_{i1}, .., x_{i2})$ и принадлежит некторому *классу* $y_i \in Y = \{1, .., M\}$. Тогда *целевой фунцией* является отображение f, такое что $f(x_i) = y_i \ \ \forall i \in I$. Пусть нам известны значения целевой функции на конечном подмножестве $\{x_1, .., x_n\} \subset X$. Множество пар $X' = (x_i, y_i)_{i=1}^k$ называется *тренировочным множеством*. Тогда задачей *обучения по прецедентам* будет восстановление целевой функции по тренировочному множеству.

Классификатор – это система (алгоритм), которая на основе тренировочного множества строит приближение целевой функции. Соответственно, классифицировать объект означает определить, какому классу он принадлежит. Кроме того, приближение целевой функции зависит не только от объекта, но и от модели алгоритма: $f(x, \theta) : X \to Y$. *Модель алгоритма* – параметрическое семейство θ. Подбор оптимального значений этого параметрического семейства называется *тренировкой алгоритма*.

Для проверки точности классификатора нередко от тренировочного множества отделяется некоторая часть, называемая *тестовым множеством*, результаты работы классификатора на которой сравниваются с предоставленными метками классов.

2.2 Классификатор на основе дерева выбора

Деревом выбора будем называть дерево, каждому внутреннему узлу которого соответствует некая функция с конечным числом значений от свойства объекта, и каждому значению этой функции соответствует потомок узла.

Каждому листовому узлу сопоставлена метка класса объекта. Задачей алгоритма является построение оптимального дерева выбора. Затем достаточно было бы пройти по дереву от корня, переходя в поддеревья, соответствующие свойствам объекта, и получить предполагаемую метку класса. Однако построение оптимального дерева является NP-полной задачей, поэтому приходится применять методы, развернуто описанные в работе [5].

2.3 Байесовский классификатор

Пусть x - вектор свойств, y - метка класса.
Рассмотрим так называемый наивный байесовский классификатор, так как другие байесовские классификаторы являются его улучшениями. В основе метода лежат две вещи:

- теорема Байеса

$$P(y = C|x) = \frac{P(C)P(x|y = C)}{P(x)}, \qquad x = (x_1, ..., x_n) \tag{1}$$

  Здесь P(A|B) означает вероятность наступления события A при истинности B; P(A) – полная вероятность наступления события A.

- Предположение о том, что координаты вектора x меняются независимо (увы, это далеко не всегда истинно)

Тогда

$$P(x|y = C) = \prod_{i=1}^{n} P(x_i|y = C) \tag{2}$$

$$P(x) = \prod_{i=1}^{n} P(x_i) \tag{3}$$

$$P(y = C|x) = \frac{P(C)\prod_{i=1}^{n} P(x_i|y = C)}{\prod_{i=1}^{n} P(x_i)} \tag{4}$$

Заметим, что правая часть формулы (4) вычисляется приближенно из данных тренировочного множества. Тогда задача сводится к нахождению

$$\max_{C \in \{C_1,..,C_n\}} P(y = C|x) \tag{5}$$

Подробно модель байесовского классификатора и детали реализации рассмотрены в работе [6].

2.4 Классификатор на основе метода опорных векторов

Основная идея в том, что вектор свойств интерпретируется геометрически – как точка в d-мерном пространстве. Тогда, если у нас есть только два класса, то нужно их разделить гиперплоскостью, которую можно задать функцией $f(x)$, где x - вектор свойств объекта. Чтобы узнать метку класса объекта с вектором свойств x_1 достаточно посмотреть на знак $f(x_1)$. Если классов более двух, достаточно использовать метод "один против всех" (one-vs.-all). Но, во-первых, встаёт вопрос существования разделяющей плоскости, а, во-вторых, выбора оптимальной плоскости. Эти вопросы рассмотрены в статье [7].

3 Постановка задачи

Будем моделировать шум на наборе данных. Затем проведем два эксперимента: будем обучать классификатор на зашумленном тренировочном множестве и на незашумленном. Далее проверим качество работы классификатора на зашумленном тестовом множестве. Критерием качества будем считать процент верно классифицированных элементов в тестовом множестве.

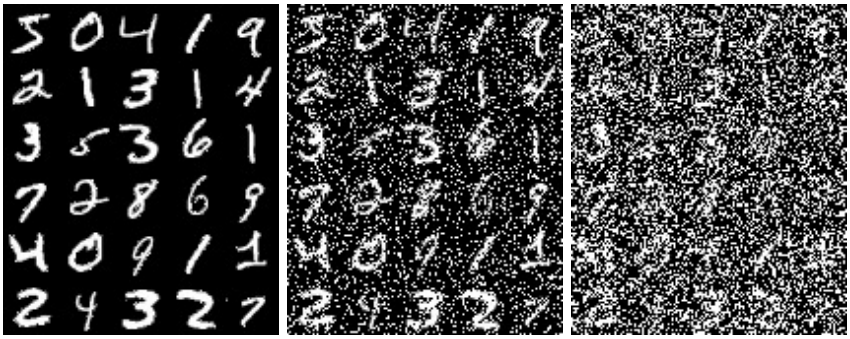
4 Работы в данной области

Есть достаточно много связанных с выбранной темой статей. В статье [8] произведено исследование деградации качества работы алгоритма наивный байес на зашумленных данных. В работе [9] произведено обширное исследование пяти алгоритмов (в том числе SMO) на качество работы при ассиметричном зашумлении атрибутов. Quinlan произвел схожее исследование алгоритма на основе дерева выбора (ID3) в своей работе [10]. Недостатком этих работ является отсутствие тестов на датасетах с большим вектором признаков, что имеет место, например, при классификации изображений.

5 Описание эксперимента

В качестве набора данных был выбран MNIST Database[11], содержащий 60000 черно-белых изображений рукописных десятичных цифр в виде 784 байтов, каждый из которых описывает яркость пикселя. Далее на изображения наносится шум следующим образом - случайно выбирается $c*784$ пикселей, где c - процент испорченных пикселей, и каждый выбранный пиксель случайным образом либо становится белым, что соответствует значению 255, либо чёрным (0). Пример зашумления изображений показан на рис. 1. Ввиду ограниченности ресурсов все рассчёты проводились на подмноже-

стве датасета размером 15000 элементов, разделенном на тренировочное и тестовое множество в пропорции 2:1.



Изображения без шума | 30% пикселей испорчены | 60% пикселей испорчены

Рис. 1: Пример зашумления изображений

Для эксперимента были использованы реализации выбраных алгоритмов в пакете анализа данных Weka: Random Tree, Naive Bayes, SMO.

Для Random Tree и Naive Bayes были использованы стандартные параметры, но классификатору “SMO” требуется значительное количество времени для тренировки, в сравнении с другими выбранными алгоритмами, так что было решено провести небольшое дополнительное исследование этого алгоритма на зависимость времени и качества его работы от параметра complexity. Было выбрано подмножество MNIST Database размером 15000 элементов. Затем алгоритм SMO был запущен 10 раз для каждого значения параметра на датасете с уровнем шума от 0.0 до 0.9. Шум наносился указанным выше образом. Исследование производилось на компьютере, оснащенном процессором Intel Core i7 2.5GHz. Как видно по рис. 2, лучшие результаты алгоритм показывает при значении параметра complexity 0.001 - это значение и будем использовать. Также отметим, что временные характеристики работы SMO очень сильно зависят от количества шума в данных.

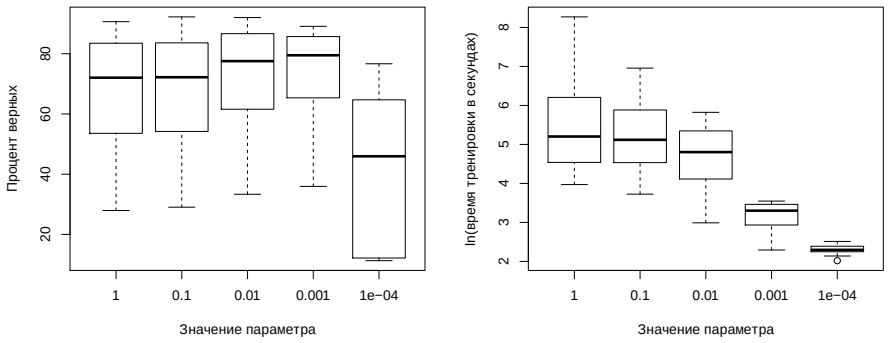


Рис. 2: Зависимость качества и времени работы алгоритма SMO от параметра complexity

6 Результаты эксперимента

График на рис. 3 иллюстрирует зависимость качества работы алгоритма от количества испорченных пикселей. Легко видеть, что эффективность Desicion Tree быстро падает и уже при 10% испорченных пикселей количество корректных результатов падает до 50%. По графику на рис. 4 в свою очередь видно, что при тренировке на незашумленных данных уже эффективность Naive Bayes падает очень быстро.

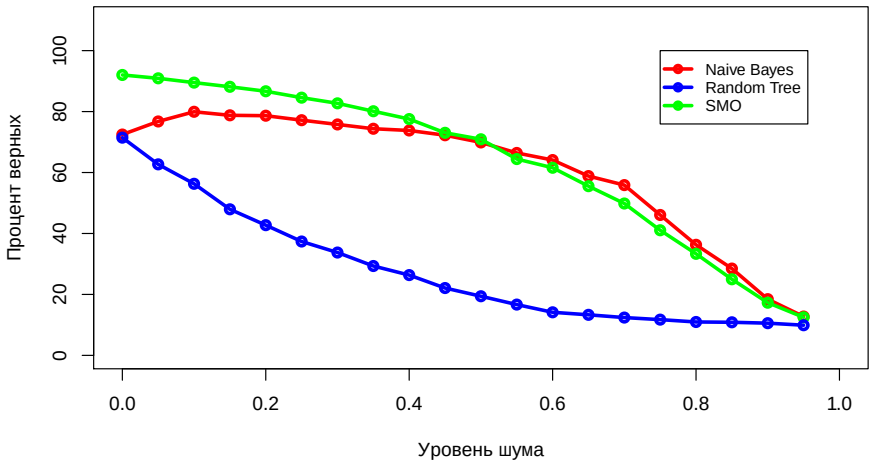


Рис. 3: Эффективность алгоритмов при построении модели на зашумленных данных

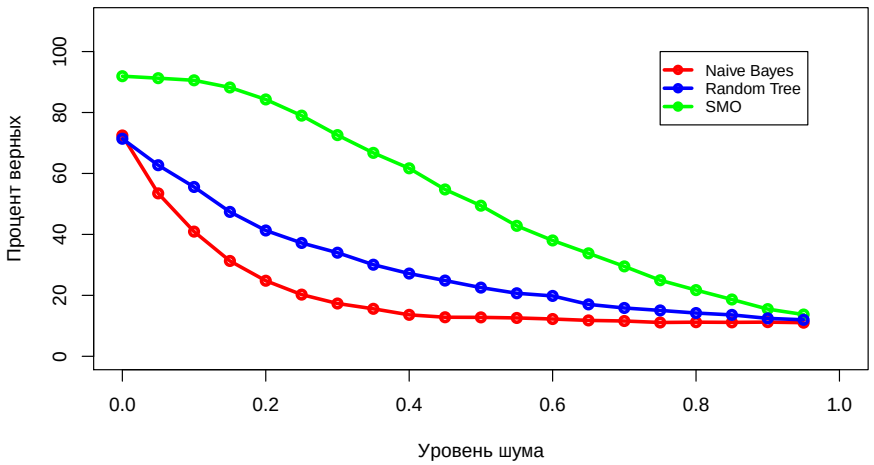


Рис. 4: Эффективность алгоритмов при построении модели на незашумленных данных

7 Анализ результатов

По рис. 3 видно, что Naive Bayes и SMO показывают примерно одинаковые результаты. Тем не менее наблюдается существенно более высокий процент верно классифицированных элементов алгоритмом при уровне шума меньше 0.4. Это можно объяснить, во-первых, тем, что параметр complexity был подобран оптимальным для данного датасета, а во-вторых, тем, что Naive Bayes использует достаточно простую гипотезу, так что, более продвинутый вероятностный классификатор показал бы более впечатляющие результаты. Исследованием этого занимались McCalum & Nigam [12]. Но подбор оптимальных параметров и оптимальных улучшений алгоритма выходит за рамки данной работы.

Плохие результаты Random Tree на рис. 3 скорее всего вызваны теми же причинами, что и у Naive Bayes, кроме того, проблемы могут быть вызваны переобучением, так как вектор признаков в выбранном датасете достаточно большой. Тем не менее видно, что характеристики работы алгоритма сильно падают с количеством шума.

Первое, что видно на рис. 4: Naive Bayes практически перестаёт работать уже при относительно небольшом уровне шума. И вот почему. Вспомним тождество:

$$P(y = C|x) = \frac{P(C)\prod_{i=1}^{n} P(x_i|y = C)}{\prod_{i=1}^{n} P(x_i)} \tag{4}$$

Правая его часть оценивается статистически из тренировочного множества. Но, когда тестовое и тренировочное множество зашумлены ассиметрично, эта статистическая оценка становится бесполезной. Отметим, что схожие данные были получены авторами статьи [8].

Также отметим, что на качество работы алгоритма SMO имел высокое качество работы в обоих случаях.

8 Выводы

В работе было произведено сравнение алгоритмов Naive Bayes, SMO и Random Tree на предмет устойчивости к зашумлению входных данных.

- Алгоритм “Random Tree” оказался неустойчив к шуму
- Алгоритм “Naive Bayes” показал удовлетворительные результаты при обучении на зашумленном множестве и неудовлетворительные при обучении на незашумленном
- Классификатор “SMO” показал лучшие результаты из представленных алгоритмов

Список литературы

[1] Shlomo Hershkop and Salvatore J. Stolfo. Identifying spam without peeking at the contents. *Crossroads*, 11(2):3–3, December 2004.

[2] Andrea Mannini and Angelo Maria Sabatini. Machine learning methods for classifying human physical activity from on-body accelerometers. *Sensors*, 10(2):1154–1175, 2010.

[3] Pedro Domingos. A few useful things to know about machine learning. *Commun. ACM*, 55(10):78–87, October 2012.

[4] Xindong Wu, Vipin Kumar, J Ross Quinlan, Joydeep Ghosh, Qiang Yang, Hiroshi Motoda, Geoffrey J McLachlan, Angus Ng, Bing Liu, S Yu Philip, et al. Top 10 algorithms in data mining. *Knowledge and Information Systems*, 14(1):1–37, 2008.

[5] S. Rasoul Safavian and David Landgrebe. A survey of decision tree classifier methodology. *IEEE Transactions on Systems, Man, and Cybernetics*, 21(3):660–674, 1991.

[6] Kevin P Murphy. Naive bayes classifiers. *University of British Columbia*, 2006.

[7] Kristin P Bennett and Colin Campbell. Support vector machines: hype or hallelujah? *ACM SIGKDD Explorations Newsletter*, 2(2):1–13, 2000.

[8] Meir Glick, Anthony E Klon, Pierre Acklin, and John W Davies. Enrichment of extremely noisy high-throughput screening data using a naive bayes classifier. *Journal of biomolecular screening*, 9(1):32–36, 2004.

[9] Michael Mannino, Yanjuan Yang, and Young Ryu. Classification algorithm sensitivity to training data with non representative attribute noise. *Decision Support Systems*, 46(3):743 – 751, 2009. Wireless in the Healthcare.

[10] J.R. Quinlan. Induction of decision trees. *Machine Learning*, 1(1):81–106, 1986.

[11] Yann Lecun and Corinna Cortes. The MNIST database of handwritten digits.

[12] Andrew Mccallum and Kamal Nigam. A comparison of event models for naive bayes text classification. In *AAAI-98 Workshop on 'Learning for Text Categorization'*, 1998.