Параллельная реализация машины опорных векторов с использованием методов кластеризации

Е.В. Котельников, А.В. Козвонина

Статья посвящена одному из возможных вариантов распараллеливания процесса обучения машины опорных векторов (Support Vector Machine, SVM). Для параллельной реализации SVM предлагается использовать алгоритмы кластеризации, благодаря которым становится возможной декомпозиция по данным. Применение параллельных методов SVM позволяет более эффективно решать задачи классификации и распознавания образов.

1. Введение

Среди современных методов классификации и распознавания образов одно из ведущих мест занимают машины опорных векторов (Support Vector Machines, SVM). Системы, разработанные на их основе, успешно решают задачи в таких областях, как биоинформатика, машинное зрение, категоризация текстов, распознавание рукописных символов и др. [1].

Подобные системы обладают высокой точностью классификации, однако при этом время обучения, особенно на больших наборах данных, очень велико и увеличивается пропорционально квадрату числа обучающих примеров. Решить проблему недостаточной скорости обучения SVM возможно при использовании многопроцессорных вычислительных систем. Однако существующие методы не приспособлены для реализации на таких платформах, поэтому актуальной является задача распараллеливания алгоритмов SVM.

Статья посвящена одному из возможных вариантов решения данной задачи, а именно параллельной реализации процесса обучения машины опорных векторов на базе алгоритма минимальной последовательной оптимизации Дж. Платта с применением методов кластеризации.

2. Машины опорных векторов

Будем рассматривать задачу классификации для объектов двух классов.

Пусть заданы:

- множество X обучающих объектов, заданных векторами[1] признаков: $X=\{\mathbf{X}_1, \mathbf{X}_2, ..., \mathbf{X}_N\}$, $X \subset R^d$ (X является подмножеством евклидова пространства размерности d);
- множество Y ответов для обучающих объектов: $Y = \{y_1, y_2, ..., y_N\}$, $y_i \in \{-1, +1\}$ ($i = 1..N$).

Тогда задача классификации состоит в построении такой функции f (классификатора), которая каждому вектору $\mathbf{X}_i$ ($i = 1..N$) сопоставляет правильный ответ y_i.

В методе SVM в качестве функции f выбрана плоскость, расстояния до которой ближайших векторов обоих классов равны (см. рис. 1). Ближайшие точки-векторы называются *опорными*. При этом для всех объектов одного класса должно выполняться неравенство $f(\mathbf{X_i}) > 0$, а для всех объектов другого класса – неравенство $f(\mathbf{X_i}) < 0$ [2, 3].

Уравнение разделяющей плоскости имеет следующий вид:

$$w_1x_1 + w_2x_2 + ... + w_dx_d + w_0 = 0,$$

где d – размерность пространства признаков;
$\mathbf{W} = (w_1, w_2, ..., w_d)$ – направляющий вектор;
w_0 – скалярный порог.

Или в векторной форме:

$$(\mathbf{W}, \mathbf{X}) + w_0 = 0.$$

[1] Векторы будем обозначать жирным шрифтом.

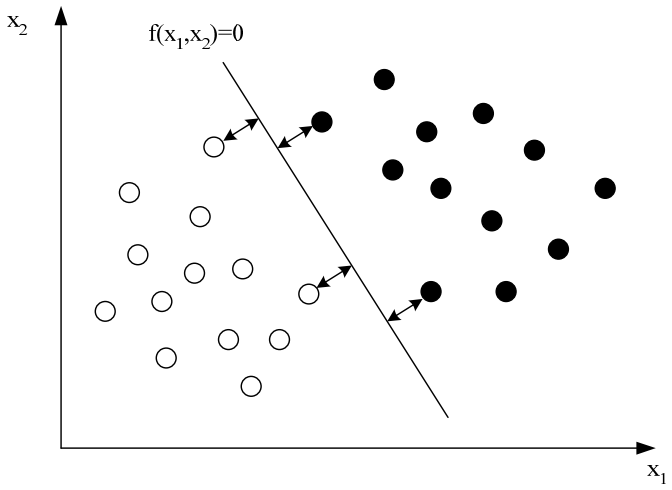

Рис. 1. Иллюстрация метода SVM для двумерного пространства

В методе опорных векторов выделяют два этапа: этап обучения и этап распознавания. На первом этапе из множества обучающих примеров отбираются опорные векторы, на основе которых строится разделяющая плоскость. Этап распознавания заключается в том, что на вход полученного классификатора подается пример **X**, о классовой принадлежности которого ничего не известно. Классификатор должен выдать ответ, к какому классу относится вектор **X**.

Отметим, что метод SVM может применяться как для линейно разделимых образов, так и для линейно неразделимых. Во втором случае используется либо линейная разделяющая функция с мягкой границей (т. е. допускается минимальное количество ошибок классификации), либо осуществляется переход в пространство большей размерности, в котором образы становятся линейно разделимыми.

Для построения оптимальной разделяющей плоскости существует множество подходов, среди которых можно выделить градиентный алгоритм Kernel-Adatron [4], алгоритм «образования фрагментов» (chunking) [5], метод декомпозиции Э. Осуны [6], инкрементные и декрементные методы [7], вероятностный SVM [8] и др. В нашей работе использовался метод последовательной минимальной оптимизации Дж. Платта (Sequential Minimal Optimization, SMO), так как на сегодняшний день данный метод является одним из наиболее быстродействующих [9, 10].

3. Метод декомпозиции по данным

Для разработки параллельных алгоритмов используются методы декомпозиции. При этом выделяются следующие основные подходы [11]: рекурсивная декомпозиция, декомпозиция по данным, поисковая декомпозиция и спекулятивная декомпозиция.

Для параллельной реализации машины опорных векторов был выбран метод декомпозиции по данным, так как задачи классификации и распознавания образов предполагают большие и сверхбольшие наборы обучающих данных, порядка 10^5–10^7 примеров, которые могут эффективно обрабатываться за счет разделения на подмножества.

Метод декомпозиции по данным включает два этапа. На первом исходное множество данных дробится на подмножества, на втором этапе полученные подмножества используются для разбиения вычислений на подзадачи.

В задаче классификации применение метода декомпозиции по данным заключается в следующем. Из исходного обучающего множества примеров выделяются два подмножества – примеры, принадлежащие первому классу, и примеры второго класса. Каждое из полученных подмножеств разбивается на M групп. Далее из групп составляются пары, в каждую из которых входит группа примеров первого класса и группа примеров второго класса. Для каждой пары строится отдельный классификатор. Затем полученные классификаторы должны объединяться.

Как было отмечено выше, в SVM на этапе обучения из множества исходных примеров выделяются опорные векторы и разделяющая плоскость строится только на их основе, остальные примеры уже не учитываются. Этот факт можно использовать следующим образом – для каж-

дой пары групп отбирать опорные векторы в общее множество опорных векторов, а итоговый классификатор строить на основе этого множества.

Такой алгоритм можно представить блок-схемой (рис. 2).

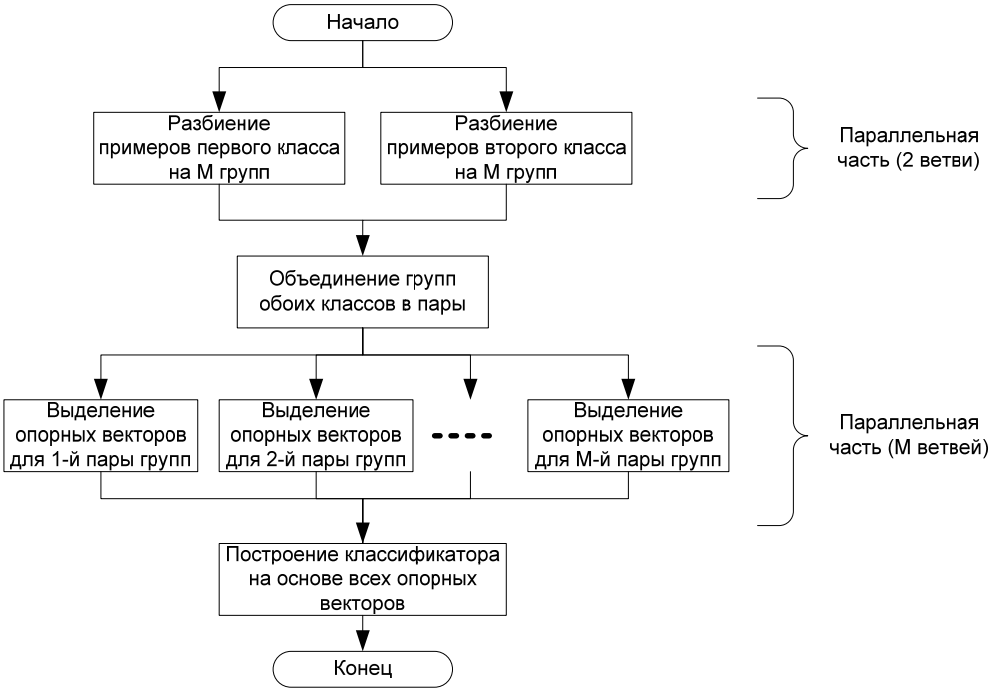

Рис. 2. Блок-схема параллельного алгоритма обучения SVM

Рассмотрим процессы разбиения исходного обучающего множества на группы объектов и объединения групп в пары. Разбиение и объединение произвольным образом не даст требуемого результата, так как может оказаться, что векторы, которые не являются опорными для пары групп, будут таковыми для задачи в целом. Пример такого неудачного разбиения приведен на рис. 3. Векторы *V1–V5* рассматриваются совместно и оказывается, что для этих векторов опорными будут векторы *V3* и *V4*, таким образом, вектор *V1* далее не рассматривается, а он является опорным при рассмотрении множества в целом.

Следовательно, необходима стратегия, которая позволила бы не упускать на этапе выделения опорных векторов для пар групп «настоящие» опорные векторы.

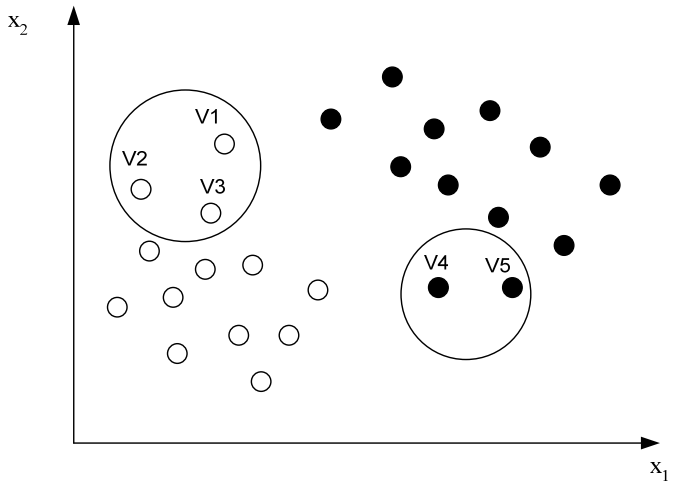

Рис. 3. Пример неудачного разбиения на группы

В качестве такой стратегии в данной работе предлагается использовать методы кластеризации для выделения групп объектов, а также объединять группы по принципу максимальной близости центров кластеров.

4. Задача кластеризации

Задача кластеризации может быть поставлена следующим образом. Пусть имеется множество X объектов, заданных векторами признаков: $X=\{\mathbf{X}_1, \mathbf{X}_2, ..., \mathbf{X}_N\}$ и число кластеров N_c. Требуется разделить множество X на N_c кластеров на основе некоторой меры сходства объектов. В качестве меры сходства используется расстояние между объектами, в данной работе применяется евклидово расстояние. Таким образом, кластеризация относится к задачам классификации без учителя (никакой априорной информации о принадлежности объекта к классу нет).

Существует множество методов кластеризации, в том числе алгоритм k-средних (k-means) [12], алгоритм Fuzzy C-Means [13], агломеративные и дивизимные алгоритмы [14], алгоритм динамической кластеризации [15] и др. В данной работе используется метод k-means, в связи с тем, что он обладает высокой скоростью кластеризации при удовлетворительном качестве.

В этом методе осуществляется разбиение на k кластеров. Алгоритм можно описать следующим образом:

1) произвольно выбирается k объектов из множества X, которые являются первоначальными центрами кластеров;
2) каждый из оставшихся объектов присоединяется к тому кластеру, центр которого находится ближе всего к данному объекту;
3) координаты центров пересчитываются как покоординатные средние всех точек, принадлежащих данному кластеру;
4) переход к пункту 2.

Пункты 2–4 повторяются до тех пор, пока координаты центров кластеров не перестанут изменяться или не будет превышено максимальное заданное число итераций.

5. Обучение с использованием кластеризации

При использовании кластеризации алгоритм обучения SVM принимает следующий вид. Сначала множество примеров одного класса при помощи метода k-means разбивается на k кластеров. Примеры другого класса также разделяются на k кластеров. Затем полученные кластеры объединяются в пары, причем в каждой паре находятся кластеры обоих классов. Объединение осуществляется по критерию ближайших центров, т. е. для первого кластера *K1* первого класса выбирается тот кластер другого класса, центр которого расположен ближе всего к центру *K1*. Кластеры образуют пару и далее не участвуют в формировании пар.

В полученных парах кластеров при помощи алгоритма последовательной минимальной оптимизации SMO выбираются опорные векторы, причем выбор производится одновременно среди всех пар кластеров. Отобранные векторы затем отправляются в итоговый классификатор (также на основе SMO), который формирует результирующую разделяющую плоскость.

Количество кластеров k следует выбирать исходя из числа процессоров p в используемой вычислительной системе. Оптимально, если $k = p$.

Блок-схема описанного алгоритма представлена на рис. 4.

Пример обучения с использованием кластеризации приведен на рис. 5. В примере число кластеров $k = 2$. На первом этапе примеры первого класса разбиваются на кластеры *K1* и *K2*, примеры второго класса – на *K3* и *K4*. Далее кластеры объединяются в пары (*K1*, *K3*) и (*K2*, *K4*), так как из кластеров *K3* и *K4* центр кластера *K3* наиболее близок к центру кластера *K1*.

Сформированные пары рассматриваются совместно и выделяются опорные векторы – *V1*, *V2*, *V3* и *V4*. Полученные векторы используются для обучения итогового классификатора, который строит плоскость $f(x_1, x_2) = 0$, являющуюся результатом решения задачи обучения.

6. Экспериментальная часть

На основе предложенного метода обучения SVM была реализована тестирующая программа. Она позволяет вводить обучающий набор данных, выполнять обучение на базе разработанного метода и с использованием только алгоритма SMO, замерять временные затраты для обоих способов обучения. Параллельное выполнение осуществлено в виде потоков.

В качестве тестовых обучающих данных применялся случайно сгенерированный линейно разделимый набор, состоящий из разного числа примеров – от 1000 до 10000. Первоначальные эксперименты на двухпроцессорном компьютере (2xIntel Xeon, 3 ГГц, 1 Гб ОЗУ) показали, что разработанный метод позволяет получить ускорение обучения в среднем на 80%.

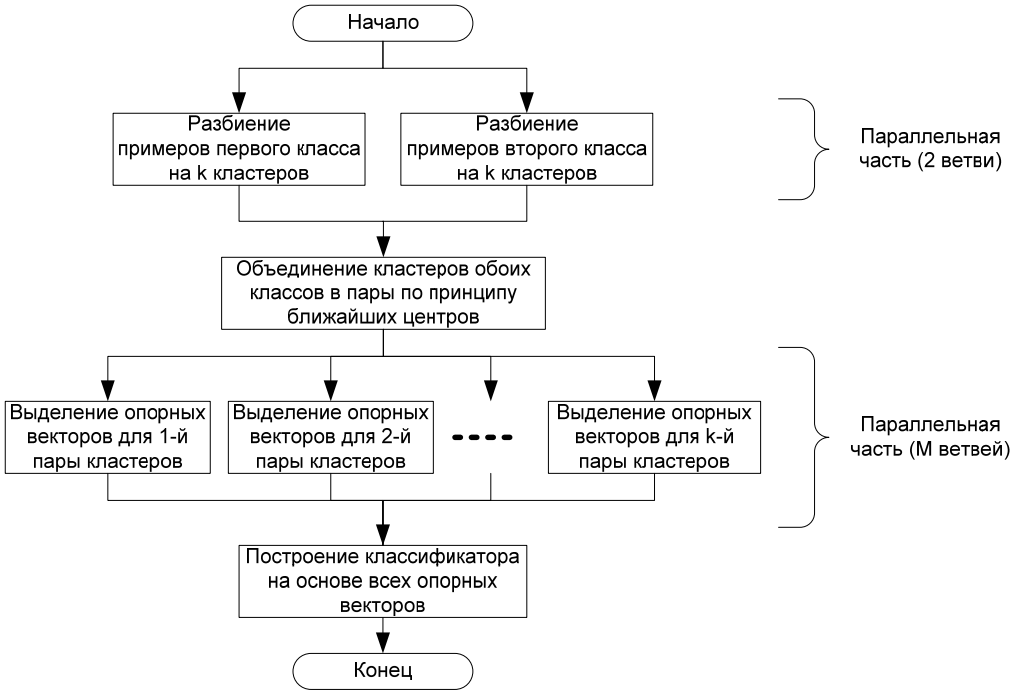

Рис. 4. Блок-схема алгоритма обучения с кластеризацией

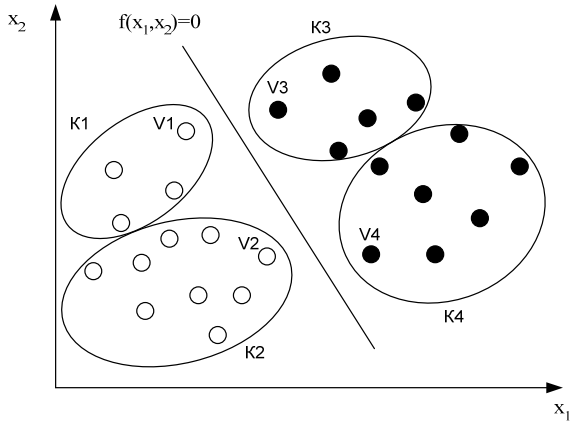

Рис. 5. Пример обучения с кластеризацией (k=2)

7. Заключение

Таким образом, был разработан параллельный метод обучения машины опорных векторов SVM на основе применения алгоритма кластеризации k-средних (k-means). Метод включает две параллельные части – кластеризация объектов и выделение опорных векторов для пар сформированных кластеров.

Метод был программно реализован с использованием алгоритма SMO Дж. Платта. Первоначальное экспериментальное тестирование показало, что применение разработанного метода позволяет на 80% ускорить обучение SVM по сравнению с классическим SMO.

В дальнейшем предполагается продолжить экспериментальные исследования на более крупных, а также нелинейных наборах данных.

Литература

1. Burges C.A. Tutorial on Support Vector Machines for Pattern Recognition. Kluwer Academic Publishers, Boston, 1998.
2. Vapnik V.N. Statistical Learning Theory. New York: Viley, 1998.
3. Vapnik V.N. The Nature of Statistical Learning Theory. Springer-Verlag, 1995.
4. Campbell C., Frie T.-T., Cristianini N. The Kernel Adatron Algorithm: A Fast and Simple Learning Procedure for Support Vector Machines // 15th International Conference Machine Learning, Morgan Kaufman, 1998.
5. Boser B.E., Guyon I.M., Vapnik V.N. A training algorithm for optimal margin classifiers // Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory, ACM Press, 1992. Pp. 144–152.
6. Osuna E., Freund R., Girosi F. An improved training algorithm for support vector machines // Proceedings of IEEE NNSP'97, Amelia Island, FL, 1997.
7. Cauwenberghs G., Poggio T. Incremental and decremental support vector machine learning // Proceedings Advances in Neural Information Processing Systems, Vancouver, Canada, 2000.
8. Fung G., Mangasarian O. L. Proximal support vector machine classifiers // F. Provost, R. Sricant, editors, Proceedings KDD-2001: Knowledge Discovery and Data Mining, San Francisco CA, New York, 2001.
9. Platt J. Sequential Minimal Optimization: A Fast Algorithm for Training Support Vector Machines // Advances in Kernel Methods Support Vector Machine, MIT Press, Cambridge, 1999. Pp. 185–208.
10. Mak J. The implementation of Support Vector Machines using the Sequential Minimal Optimization algorithm. McGill University, Montreal, Canada, 2000.
11. Grama A., Gupta A., Karypis G., Kumar V. Introduction to Parallel Computing. Addison-Wesley, 2003.
12. MacQueen J. Some methods for classification and analysis of multivariate observations // Proc. 5th Berkeley Symposium on Mathematical Statistics and Probability, Vol. 1, 1967. Pp. 281–297.
13. Bruske J., Ahrns L., Sommer G. An integrated architecture for learning of reactive behaviors based on dynamic cell structures // Robotics and Autonomous Systems, 22, 1998. Pp. 81–102.
14. Барсегян А.А., Куприянов М.С., Степаненко В.В., Холод И.И. Методы и модели анализа данных: OLAP и Data Mining. – СПб.: БХВ-Петербург, 2004.
15. Moody J., Darken C. Fast learning in networks of locally-turned processing units // Neural Computation, 1(2), 1989. Pp. 281–294.