Structural One-Class Support Vector Machine on Hadoop

Yuxing Qian[1,a], Aimin Feng[2,a]

[1]Master degree candidate of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing 210000, China

[2]Associate professor of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing 210000, China

a.Email: yuxingqian_nj@sina.com

Keywords: (S)OCSVM, Big Data, MapReduce, Hadoop

Abstract. SOCSVM, as the derivative algorithm of OCSVM, is powerful in One-Class Classification. But in big data era, the computing and the storage requirement increase rapidly with the number of training vectors, putting many practical problems out of reach. For applying SOCSVM to large scale data mining, parallel SOCSVM is proposed by means of the idea of MapReduce and the iterative Hadoop model. The experiment results show SOCSVM on Hadoop is efficiency on practical problems.

Introduction

Recently, One-Class Classifier (OCC)[1] is widely used in so many fields such as machine fault diagnosis[2], credit card fraud[3] and Intrusion detection[4] etc.. Since it only has one class of samples to train the classifier, OCC belongs to unsupervised learning and is more difficult than binary or multi class classifier[5]. The main three different kinds of OCC are density-based[6], clustering-based 错误!未找到引用源。and domain based classifiers 错误!未找到引用源。. Among them, domain based classifier is extremely popular because it is derived from SVM and the classical algorithms of which OCSVM(One-Class Support Vector Machine)错误!未找到引用源。 and SVDD(Support Vector Data Description)错误!未找到引用源。. In details, OCSVM uses the origin as the representative of negative class and observes the optimal hyperplane by maximizing the Euclidean distance between the origin and the target data. In order to further integrate the global and local information in the same framework, SOCSVM(Structural One-Class Support Vector Machine)错误!未找到引用源。错误!未找到引用源。 embeds the scatter of class into OCSVM and can obtain better performance for structure data.

MapReduce and Hadoop

The MapReduce programming model was proposed by Google to support data-intensive applications running on parallel computers. Two important functional programming primitives in MapReduce are Map and Reduce. The Map function is applied on application-specific input data to generate a list of intermediate <key, value> pairs while the Reduce function is applied to the set of intermediate pairs with the same key. Typically, the Reduce function produces zero or more output pairs that are finally sorted based on their key values.

Hadoop is a successful implementation of the MapReduce model. The Hadoop framework consists of two main components: the MapReduce language and the Hadoop's Distributed File

System(HDFS). The Hadoop runtime system is coupled with HDFS and manages the details of parallelism and concurrency to provide ease of parallel programming with reinforced reliability.

Structural One-Class Support Vector Machine(SOCSVM)

SOCSVM integrates global and local information into one framework. The corresponding objective function is as follow:

$$\begin{aligned} &\min_{w\zeta\rho} \frac{1}{2} w^T w + \frac{\lambda}{2} w^T \sum w - \rho + \frac{1}{vn} \sum_{i=1}^{n} \zeta_i \\ &s.t. w^T (x_i) \geq \rho - \zeta_i, \zeta_i \geq 0, i = 1, \cdots, n \end{aligned} \tag{2.1}$$

λ is the normalized factor to balance the local and global information. Obviously, the value is bigger, the global information is more critical, on the contrary, the value of λ is smaller, the local information is more critical. $\lambda \in [0, \infty)$.

The geometric interpretation of formula (2.1) maximizes the margin between classes and minimizes within-class scatter at simultaneously.

The formula (3.1) that uses Lagrange multipliers gets the following expressions:

$$\begin{aligned} L(w, \zeta, \rho, \alpha, \beta) = &\frac{1}{2} w^T w + \frac{\lambda}{2} w^T \sum w + \frac{1}{vn} \sum_i \zeta_i - \rho \\ &- \sum_i \alpha_i (w \cdot x_i - \rho + \zeta)_i - \sum_i \beta \zeta_i \end{aligned} \tag{2.2}$$

Let the partial derivatives of w, ζ, ρ to be 0 and the results are as follow:

$$w = \sum_i \alpha_i \varphi(x_i) \tag{2.3}$$

$$\alpha_i = \frac{1}{vn} - \beta_i \leq \frac{1}{vn} \tag{2.4}$$

$$\sum_{i=1}^{n} \alpha_i = 1 \tag{2.5}$$

Put (2.3),(2.4) and (2.5)into (2.2), the matrix expression is as follows:

$$\begin{aligned} &\min_{\alpha} \frac{1}{2} \alpha^T X^T (I + \lambda \Sigma)^{-1} X \alpha \\ &\text{s.t.} \alpha^T I = I, 0 \leq \alpha \leq \frac{1}{vn} I \end{aligned} \tag{2.6}$$

Actually, when the covariance matrix of SOCSVM is an identity matrix, (2.6) becomes the dual expression of OCSVM:

$$\begin{aligned} &\min_{\alpha} \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j K(x_i, x_j) \\ &\text{s.t.} \sum_{i=1}^{n} \alpha_i = 1, 0 \leq \alpha_i \leq \frac{1}{vn}, i = 1, \cdots, n; \end{aligned} \tag{2.7}$$

SOCSVM_Hadoop

The SOCSVM_Hadoop is based on the iterative SOCSVM model, each cycle is used as filter to help us to find the support vector. This makes it drive partial solutions towards the global optimum 错误!未找到引用源。 directly. Through this way, large scale data problems can be divided into independent, smaller optimizations.

The flow chart of SOCSVM_Hadoop is as follow:

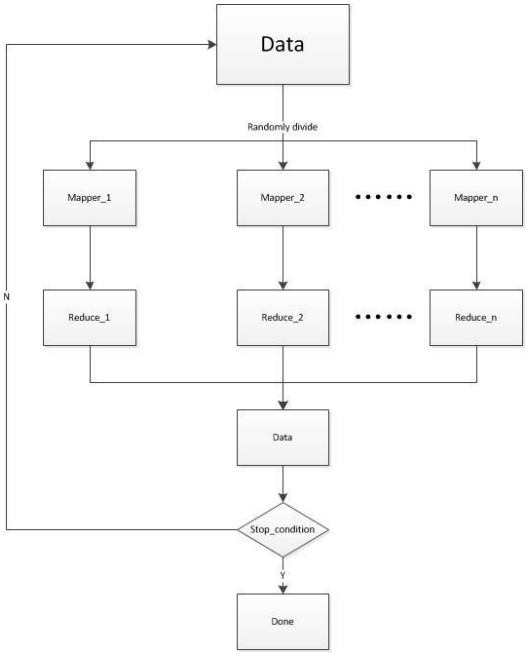


Figure 1. Iterative process flow chart

As Figure 1 says, original training sample data set randomly divides into M(M is the number of computer under the framework of Hadoop) data groups, usually M is decided according to the size of the data, here M=n. Each data set can be assigned to a Mapper node and execute independently. In this framework, let's choose a node as a Master for scheduling of parallel tasks. After each Mapper core calculates its own required tasks and communicates with the Master, the Master schedules each matching Reduce nodes for processing and returns the result to determine whether the next iteration is necessary. Each corresponding Mapper_i and Reduce_i is in one computer. After each cycle the size of data becomes smaller and smaller, the training speed of each Mapper will be quicker and quicker.

The linear programming for SOCSVM_Hadoop

SOCSVM_Hadoop is quadratic programming and the complexity is $o(n^3)$错误!未找到引用源。.

In order to further improve the efficiency, inspired by the liner programming algorithm of OCSVM, the linear programming for SOCSVM_Hadoop (SOCSVM_Hadoop(lp)) is proposed. The main idea is to minimize the distance between the mean of target data and the hyperplane. The corresponding objective function is as follow:

$$\begin{aligned} \min W(\alpha,\rho) &= \frac{1}{n}\left[\alpha^T X^T \left(I+\lambda \Sigma\right)^{-1} X \right] - \rho^T I \\ s.t. X^T &\left(I+\lambda \Sigma\right)^{-1} X\alpha \geq \rho 1 \\ \alpha^T 1 &= 1, \alpha \geq 0 \end{aligned} \tag{2.8}$$

λ, just like in formula (2.1), is the normalized factor for the balance of local and global information. In order to avoid the influence of noise and outliers, the relaxation factor $\varepsilon = [\varepsilon_1, \cdots, \varepsilon_n]^T$ can be introduced into the formula (4.1):

$$\begin{aligned} \min W(\alpha,\rho) &= \frac{1}{n}\left[X^T\left(I+\lambda\Sigma\right)^{-1}X\alpha-\rho 1+\frac{1}{v}\varepsilon\right]^T I \\ \text{s.t.} X^T\left(I+\lambda\Sigma\right)^{-1}X\alpha &\geq \rho 1-\varepsilon, \varepsilon \geq 0 \\ \alpha^T I &= I \end{aligned} \qquad (2.9)$$

For the data partitioning part, because the data feature is kept, the above method is still being using. And it is also based on the iterative SOCSVM model.

Experiment results and analysis

In order to verify the performance of SOCSVM_Hadoop and its linear programming algorithm, here we design the experiment and compare them with SOCSVM to evaluate their efficiency.

The experimental code was written in JAVA and compiled by Eclipse. SOCSVM ran on a 3.4 GHz i7 processor and the operating system is Windows 7. The size of the main memory was about 3.89 Gigabytes. SOCSVM_Hadoop and SOCSVM_Hadoop(lp) ran on three computers which have 3.89 Gigabytes, 3.4 GHz i7 processor and the operating system is Windows 7.

The database MNIST 错误!未找到引用源。 is available from the website. It has 10 classes of handwritten data about 0 to 9 and contains 60,000 training samples and 10,000 testing samples. The preprocessing was done by LeCun's research group.

Before verify the performance of the algorithm, the most appropriate numbers of computer for the MNIST database is determined.

In the training time aspect, both SOCSVM_Hadoop and SOCSVM_Hadoop(lp) are about half of SOCSVM. Indeed, the time of SOCSVM_Hadoop(lp) is less than SOCSVM_Hadoop's 723s. This means that linear programming is quicker than quadratic programming.

Meanwhile to further prove the result of 5.1, algorithms still run under three framework with M=3,4,5.The comparisons of the three algorithms are shown in the Figure 2 and 3.

Figure 2. Total time of SOCSVM/SOCSVM_Hadoop/SOCSVM_Hadoop(lp) with different m

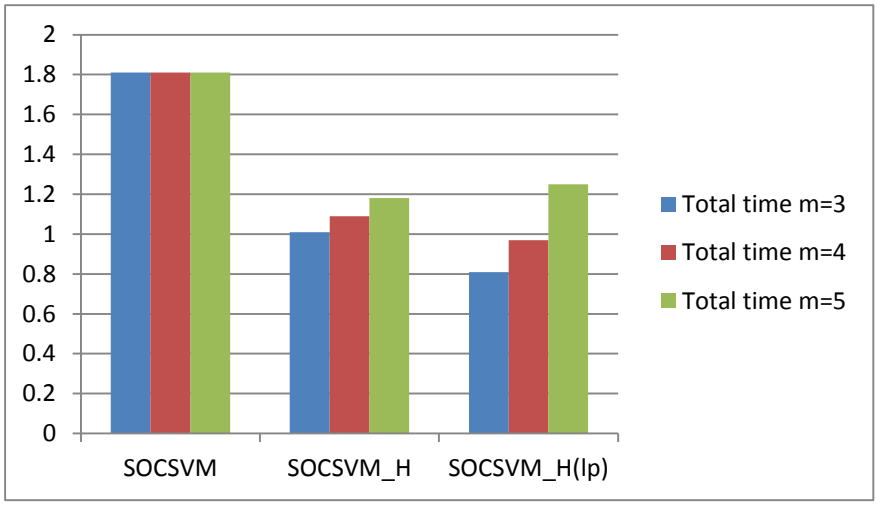
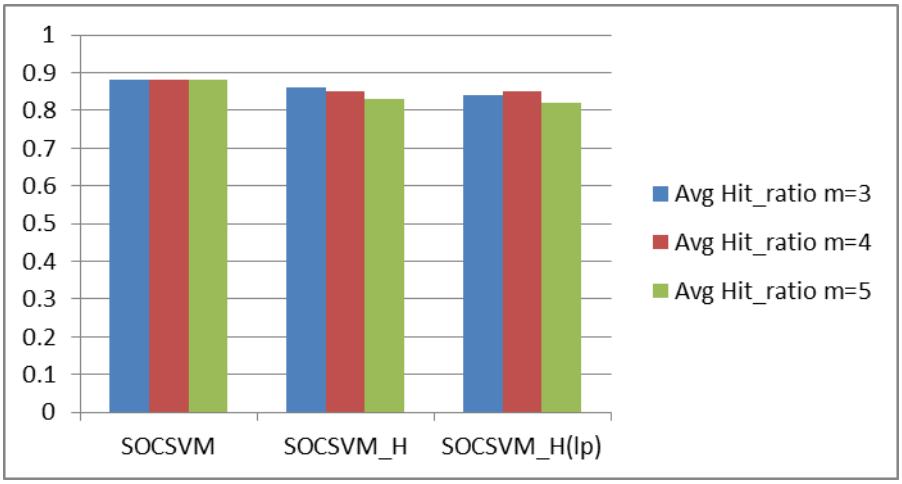


Figure 2. Average Hit_ratio of SOCSVM/OCSVM_Hadoop/SOCSVM_Hadoop(lp) with different m



In figure 2 and 3, some points are shown:

When M is bigger, the performance of algorithm is worse and the time of SOCSVM_Hadoop(lp) is always less than the SOCSVM_Hadoop's.

In stark contrast the hit_ratio of SOCSVM_Hadoop is always accurate.

In a word, the training time of SOCSVM_Hadoop is reduced about 50% and its hit_ratio is in the acceptable range. This fully shows that the performances of the algorithms are significantly improved.

Conclusion

In this paper, parallel SOCSVM model based on iterative Hadoop is proposed its linear programming is also implemented for further improve the efficiency. Through analysis, we knows that SOCSVM_Hadoop do reduce the computation time greatly, meanwhile it also inherits the SOCSVM's characteristics and the high accuracy of classification. And SOCSVM_Hadoop(lp), to some extent, improves the operation efficiency of the algorithm, but brings a little reduction of accuracy rate.

We also knows that the value of M is key to keep the performance of SOCSVM_Hadoop and SOCSVM_Hadoop(lp). It implies that the data partitioning destroys the structural feature. Dividing data sets better self-adapting becomes a new research subject in our future work. We will find an efficient way to keep structural feature of data better and make SOCSVM_Hadoop more accurate.

References

[1]. Shahid N, Naqvi I H, Qaisar S B. One-class support vector machines: analysis of outlier detection for wireless sensor networks in harsh environments[J]. Artificial Intelligence Review, 2015, 43(4):515-563.

[2]. Chun-Lin W U, Pan H X. Research Artillery Automatic Machine Fault Diagnosis[J]. Mechanical Management & Development, 2013.

[3]. Akhilomen J. Data Mining Application for Cyber Credit-Card Fraud Detection System[M]// Advances in Data Mining. Applications and Theoretical AspectsSpringer Berlin Heidelberg, 2013:218-228.

[4]. Kreibich C, Crowcroft J. Honeycomb: creating intrusion detection signatures using honeypots[J]. Acm Sigcomm Computer Communication Review, 2004, 34(1):51-56.

[5]. Qian H, Mao Y, Xiang W, et al. Recognition of human activities using SVM multi-class classifier[J]. Pattern Recognition Letters, 2010, 31(2):100–111.

[6]. Breunig M, Kriegel H P, Ng R T, et al. LOF: Identifying Density-Based Local Outliers[C]// PROCEEDINGS OF THE 2000 ACM SIGMOD INTERNATIONAL CONFERENCE ON MANAGEMENT OF DATA.