International Journal of Engineering Science Invention
ISSN (Online): 2319 – 6734, ISSN (Print): 2319 – 6726
www.ijesi.org ||Volume 4 Issue 12|| December 2015 || PP.58-63

Algorithms in Data Mining: Top 5

Jagannath Ray1, Nibedita Sahoo2, Suren Ku. Sahu3
[1,2]*Associate Professor, Department of Computer Science Engineering, Gandhi Institute For Technology (GIFT), Bhubaneswar*
[3] *Assistant Professor, Department of Computer Science Engineering, Gandhi Engineering College, Bhubaneswar*

Abstract: *This paper presents the top 10 data mining algorithms identified by the IEEE International Conference on Data Mining (ICDM) in December 2006: C4.5, k-Means, SVM, Apriori, EM, PageRank, AdaBoost, kNN, Naive Bayes, and CART. These top 10 algorithms are among the most influential data mining algorithms in the research community. With each algorithm, we provide a description of the algorithm, discuss the impact of the algorithm, and review current and further research on the algorithm. These 10 algorithms cover classification, clustering, statistical learning, association analysis, and link mining, which are all among the most important topics in data mining research*
and development.

I. Introduction

In an effort to identifying some of the most influential algorithms that have been widely used in the data mining community, the IEEE International Conference on Data Mining (ICDM,

http://www.cs.uvm.edu/icdm/) identified the top 10 algorithms in data mining for presentation at ICDM ’06 in Hong Kong.

As the first step in the identification process, we invited the ACM KDD Innovation Award and

IEEE ICDM Research Contributions Award winners in September 2006 to each nominate up to 10 bestknown algorithms in data mining. All except one in this distinguished set of award winners responded to our invitation. We asked each nomination to come with the following information: (a) the algorithm name, (b) a brief justification, and (c) a representative publication reference. We also advised that each nominated algorithm should have been widely cited and used by other researchers in the field, and the

nominations from each nominator as a group should have a reasonable representation of the different areas in data mining.

After the nominations in Step 1, we verified each nomination for its citations on Google Scholar in late October 2006, and removed those nominations that did not have at least 50 citations. All remaining (18) nominations were then organized in 10 topics: association analysis, classification, clustering, statistical learning, bagging and boosting, sequential patterns, integrated mining, rough sets, link mining, and graph mining. For some of these 18 algorithms such as k-means, the representative publication was not necessarily the original paper that introduced the algorithm, but a recent paper that highlights the importance of the technique. These representative publications are available at the ICDM website (http://www.cs.uvm.edu/ icdm/algorithms/ CandidateList.shtml). In the third step of the identification process, we had a wider involvement of the research community.

We invited the Program Committee members of KDD-06 (the 2006 ACM SIGKDD International Conference on Knowledge Discovery and Data Mining), ICDM ’06 (the 2006 IEEE International Conferenceon Data Mining), and SDM ’06 (the 2006 SIAM International Conference on Data Mining), aswell as the ACM KDD Innovation Award and IEEE ICDM Research Contributions Award winners toeach vote for up to 10 well-known algorithms from the 18-algorithm candidate list. The voting resultsof this step were presented at the ICDM ’06 panel on Top 10 Algorithms in Data Mining.

At the ICDM ’06 panel of December 21, 2006, we also took an open vote with all 145 attendees on the top 10 algorithms from the above 18-algorithm candidate list, and the top 10 algorithms fromthis open vote were the same as the voting results from the above third step. The 3-hour panel wasorganized as the last session of the ICDM ’06 conference, in parallel with 7 paper presentation sessionsof the Web Intelligence (WI ’06) and Intelligent Agent Technology (IAT ’06) conferences at the same location, and attracting 145 participants to this panel clearly showed that the panel was a great success.

1 The K _-Means Algorithm

The k-means algorithm is a simple iterative method to partition a given dataset into a user-specified number of clusters, _. This algorithm has been discovered by several researchers across different disciplines, most notably Lloyd (1957,1982)[40], Forgey (1965), Friedman and Rubin(1967), and Mc- Queen(1967).A detailed history of k-means alongwith descriptions of several variations are given in Jain and Dubes [36]. Gray and Neuhoff [28] provide a nice historical background for k-means placed in the larger context of hill-climbing algorithms.

The algorithm operates on a set of _-dimensional vectors, _______! _"#$_&%'_)(*(*(*__+ , where denotes the _! -,/.10 #3254 data point. The algorithm is initialized by picking _points in .60 as the initial _cluster representatives or "centroids". Techniques for selecting these initial seeds include sampling at random from the dataset, setting them as the solution of clustering a small subset of the data or perturbing the global mean of the data _times. Then the algorithm iterates between two steps till convergence:
Step 1: Data Assignment. Each data point is assigned to its closest centroid, with ties broken arbitrarily.
This results in a partitioning of the data.
Step 2: Relocation of "means". Each cluster representative is relocated to the center (mean) of all data points assigned to it. If the data points come with a probability measure (weights), then the relocation is to the expectations (weighted mean) of the data partitions.

The algorithm converges when the assignments (and hence the 798 values) no longer change. The algorithm execution is visually depicted in Fig. 1. Note that each iteration needs +;:<_comparisons, which determines the time complexity of one iteration. The number of iterations required for convergence varies and may depend on +, but as a first cut, this is an algorithm can be considered linear in the dataset size.
One issue to resolve is how to quantify "closest" in the assignment step. The default measure of closeness is the Euclidean distance, in which case one can readily show that the non-negative cost function, will decrease whenever there is a change in the assignment or the relocation steps, and hence convergence is guaranteed in a finite number of iterations. The greedy-descent nature of k-means on a non-convex cost also implies that the convergence is only to a local optimum, and indeed the algorithm is typically quite sensitive to the initial centroid locations. Fig. 2 3 illustrates how a poorer result is obtained for the same dataset as in Fig. 1 for a different choice of the three initial centroids. The local minima problem can be countered to some extent by running the algorithm multiple times with different initial centroids, or by doing limited local search about the converged solution.

2.2 Limitations

In addition to being sensitive to initialization, the k-means algorithm suffers from several other problems. First, observe that k-means is a limiting case of fitting data by a mixture of _Gaussians with identical, isotropic covariance matrices (PQ_SR_)T), when the soft assignments of data points to mixture omponents ae hardened to allocate each data point solely to the most likely component. So, it will falter whenever the data is not well described by reasonably separated spherical balls, for example, if there are non-covex shaped clusters in the data. This problem may be alleviated by rescaling the data to "whiten" it before clustering, or by using a different distance measure that is more appropriate for the dataset. For example, information-theoretic clustering uses the KL-divergence to measure the distance between two data points representing two discrete probability distributions. It has been recently shown that if one measures distance by selecting any member of a very large class of divergences called Bregman divergences during the assignment step and makes no other changes, the essential properties of k-means, including guaranteed convergence, linear separation boundaries and scalability, are retained
[2]. This result makes k-means effective for a much larger class of datasets so long as an
appropriate divergence is used.

k-means can be paired with another algorithm to describe non-convex clusters. One first clusters the data into a large number of groups using k-means. These groups are then agglomerated into larger clusters using single link hierarchical clustering, which can detect complex shapes. This approach also makes the solution less sensitive to initialization, and since the hierarchical method provides results at multiple resolutions, one does not need to pre-specify _either.

2 Support Vector Machines

by Qiang Yang4 In today's machine learning applications, support vector machines (SVM) [66] are considered a must try - it offers one of the most robust and accurate methods among all well-known algorithms. It has a sound theoretical foundation, requires only a dozen examples for training, and is insensitive to the number of dimensions. In addition, efficient methods for training SVM are also being developed at
a fast pace. In a two-class learning task, the aim of SVM is to find the best classification function to distinguish between members of the two classes in the training data. The metric for the concept of the "best" classification

function can be realized geometrically. For a linearly separable dataset, a linear classification function corresponds to a separating hyperplane X@ZY Othat passes through the middle of the two classes,

separating the two. Once this function is determined, new data instance Y\[can be classified by simply testing the sign of the function X@ZY [O; Y[belongs to the positive class if X@ZY [O_Q]. Because there are many such linear hyperplanes, what SVM additionally guarantee is that the best such function is found by maximizing the margin between the two classes. Intuitively, the margin is defined as the amount of space, or separation between the two classes as defined by the hyperplane. Geometrically, the margin corresponds to the shortest distance between the closest data points to a point on the hyperplane. Having this geometric definition allows us to explore how to maximize the argin, so that even though there are an infinite number of hyperplanes, only a few qualify as the solution to SVM.

The reason why SVM insists on finding the maximum margin hyperplanes is that it offers the best generalization ability. It allows not only the best classification performance (e.g., accuracy) on the training data, but also leaves much room for the correct classification of the future data. To ensure that the maximum margin hyperplanes are actually found, an SVM classifier attempts to maximize the following function with respect to _^ and ': where ois the number of training examples, and h__p#q__%'_)()()(!_poare non-negative numbers such that the derivatives of acb with respect to h_are zero. h_are the Lagrange multipliers and a$b is called the Lagrangian. In this equation, the vectors _^ and constant 'define the hyperplane. There are several important questions and related extensions on the above basic formulation of support vector machines. We list these questions and extensions below.

1. Can we understand the meaning of the SVM through a solid theoretical foundation?
2. Can we extend the SVM formulation to handle cases where we allow errors to exist, when even the best hyperplane must admit some errors on the training data?
3. Can we extend the SVM formulation so that it works in situations where the training data are not linearly separable?
4. Can we extend the SVM formulation so that the task is to predict numerical values or to rank the instances in the likelihood of being a positive class member, rather than classification?
5. Can we scale up the algorithm for finding the maximum margin hyperplanes to thousands and millions of instances?

3 The EM Algorithm

by Geoffrey J. McLachlan6 and Angus Ng

Finite mixture distributions provide a flexible and mathematical-based approach to the modeling and clustering of data observed on random phenomena. We focus here on the use of normal mixture models, which can be used to cluster continuous data and to estimate the underlying density function. These mixture models can be fitted by maximum likelihood via the EM (Expectation-Maximization) algorithm.

Introduction

Finite mixture models are being increasingly used to model the distributions of a wide variety of random phenomena and to cluster data sets [43]. Here we consider their application in the context of cluster analysis. We let the p-dimensional vector (**y** _ @i__)(*(*(*_i_¦OI§) contain the values of ¨variables measured on each of H(independent) entities to be clustered, and we let **y**Jdenote the value of **y** corresponding to the ©th entity (©ª_«%'_)(*(*(*_pH). With the mixture approach to clustering, **y**__)(*(*(*_**y**[are assumed to be an observed random sample from mixture of a finite number, say g, of groups in some unknown proportions ¬__)(*(*(*__¬K-.

The mixture density of **y**Jis expressed as where the mixing proportions ¬__)(*(*(*__¬-sum to one and the group-conditional density X_@iJ'._°_Ois specified up to a vector °_of unknown parameters (#<_²%'_)(*(*(*_E). The vector of all the unknown parameters is given by

where the superscript ³denotes vector transpose. Using an estimate of ®, this approach gives a probabilistic clustering of the data into Eclusters in terms of estimates of the posterior probabilities of component membership,

where ´_@**y**JOis the posterior probability that iJ(really the entity with observation iJ) belongs to the #th component of the mixture (#y_µ%'_)(*(*(*_E.I©¶_.%'_)(*(*(*_pH). The parameter vector ®can be estimated by maximum likelihood. The maximum likelihood estimate (MLE) of ®, ®· , is given by an appropriate root of the likelihood equation, ¸ where is the log likelihood function for ®. Solutions of (6) corresponding to local maximizers can be obtained via the expectation-maximization (EM) algorithm [13].

For the modeling of continuous data, the component-conditional densities are usually taken to belong to the same parametric family, for example, the normal. In this case,

where ☐@iJj.p»¼_¯P Odenotes the ¨-dimensional multivariate normal distribution with mean vector »and covariance matrix P. One attractive feature of adopting mixture models with elliptically symmetric components such as the normal or odensities, is that the implied clustering is invariant under affine transformations of the data (that is, under operations relating to changes in location, scale, and rotation of the data). Thus the clustering process does not depend on irrelevant factors such as the units of measurement or the orientation of the clusters in space.

4 AdaBoost

by Zhi-Hua Zhou8

4.1 Description of the Algorithm

Ensemble learning [16] deals with methods which employ multiple learners to solve a problem. The generalization ability of an ensemble is usually significantly better than that of a single learner, so ensemble methods are very attractive. The AdaBoost algorithm [20] proposed by Yoav Freund and Robert Schapire is one of the most important ensemble methods, since it has solid theoretical foundation, very
accurate prediction, great simplicity (Schapire said it needs only "just 10 lines of code"), and wide and successful applications.

Let 7 denote the instance space and 8the set of class labels. Assume 8 ___ALv%'_¯Uu%. Given a weak or base learning algorithm and a training set _@*9 __i_O_@*9 __i_O_k)k _@*9 .Â_i.Owhere and 9 _ ,:7 i_,;8 @#{_ %'_k)k _GO, the AdaBoost algorithm works as follows. First, it assigns equal weights to all the training examples @*9 __ij_O@#,__A%'_k)k _GO. Denote the distribution of the weights at the o-th learning round as _2. From the training set and _2the algorithm generates a weak or base learner _2=<7 > 8 by calling the base learning algorithm. Then, it uses the training examples to test _2, and the weights of the incorrectly classified examples will be increased. Thus, an updated weight distribution _2|_is obtained. From the training set and _2|_AdaBoost generates another weak learner by calling the base learning algorithm again. Such a process is repeated for ³rounds, and the final model is derived by weighted majority voting of the ³weak learners, where the weights of the learners are determined during the training process. In practice, the base learning algorithm may be a learning algorithm which can use weighted training examples directly; otherwise the weights can be exploited by sampling the training examples according to the weight distribution _2.

4.2 Impact of the Algorithm

As mentioned in Section 7.1, AdaBoost is one of the most important ensemble methods, so it is not strange that its high impact can be observed here and there. In this short article we only briefly introduce two issues, one theoretical and the other applied. In 1988, Kearns and Valiant posed an interesting question, i.e., whether a weak learning algorithm that performs just slightly better than random guess could be "boosted" into an arbitrarily accurate strong learning algorithm. In other words, whether two complexity classes, weakly learnable and strongly learnable problems, are equal. Schapire [53] found that the answer to the question is "yes", and the proof he gave is a construction, which is the first Boosting algorithm. So, it is evident that AdaBoost was born with theoretical significance. AdaBoost has given rise to abundant research on theoretical aspects of ensemble methods, which can be easily found in machine learning and statistics literature. It is worth mentioning that for their AdaBoost paper [20], Schapire and Freund won the Godel Prize, which is one of the most prestigious awards in theoretical computer science, in the year of 2003.

AdaBoost and its variants have been applied to diverse domains with great success. For example, Viola and Jones [67] combined AdaBoost with a cascade process for face detection. They regarded rectangular features as weak learners, and by using AdaBoost to weight the weak learners, they got very intuitive features for face detection. In order to get high accuracy as well as high efficiency, they used a cascade process (which is beyond the scope of this article). As the result, they reported a very strong face detector: On a 466MHz machine, face detection on a WYX[Z<: dXYX image cost only 0.067 seconds, which is 15 times faster than state-of-the-art face detectors at that time but with comparable accuracy. This face detector has been recognized as one of the most exciting breakthroughs in computer vision (in particular, face detection) during the past decade. It is not strange that "Boosting" has become a buzzword in computer vision and many other application areas.

4.3 Further Research

Many interesting topics worth further studying. Here we only discuss on one theoretical topic and one applied topic. Many empirical study show that AdaBoost often does not overfit, i.e., the test error of AdaBoost often tends to decrease even after the training error is zero. Many researchers have studied this and several theoretical explanations have been given, e.g. [32]. Schapire et al. [54] presented a marginbased explanation. They argued that AdaBoost is able to increase the margins even after the training error is zero, and thus it does not overfit even after a large number of rounds. However, Breiman [6] indicated that larger margin does not necessarily mean better generalization, which seriously challenged the margin-based explanation. Recently, Reyzin and Schapire [51] found that Breiman considered minimum margin instead of average or median margin, which suggests that the margin-based explanation still has chance to survive. If this explanation succeeds, a strong connection between AdaBoost and VM could be found. It is obvious that this topic is well worth studying.

Many real-world applications are born with high dimensionality, i.e., with a large amount of input features. There are two paradigms that can help us to deal with such kind of data, i.e., dimension reduction and feature selection. Dimension reduction methods are usually based on mathematical projections, which attempt to transform the original features into an appropriate feature space. After dimension reduction, the original meaning of the features is usually lost. Feature selection methods directly select some original features to use, and therefore they can preserve the original meaning of the features, which is very desirable in many applications. However, feature selection methods are usually based on heuristics, lacking solid theoretical foundation. Inspired by Viola and Jones's work [67], we think AdaBoost could be very useful in feature selection, especially when considering that it has solid theoretical foundation. Current research mainly focus on images, yet we think general AdaBoost-based
feature selection techniques are well worth studying.

5 kNN: k-Nearest Neighbor Classification

by Michael Steinbach 9

5.1 Description of the Algorithm

One of the simplest, and rather trivial classifiers is the Rote classifier, which memorizes the entire training data and performs classification only if the attributes of the test object match one of the training examples exactly. An obvious drawback of this approach is that many test records will not be classified because they do not exactly match any of the training records. A more sophisticated approach, k-nearest neighbor (kNN) classification [19, 60], finds a group of k objects in the training set that are closest to the test object, and bases the assignment of a label on the predominance of a particular class in thisneighborhood. There are three key elements of this approach: a set of labeled objects, e.g., a set of stored records, a distance or similarity metric to compute distance between objects, and the value of k, the number of nearest neighbors. To classify an unlabeled object, the distance of this object to the labeled objects is computed, its k-nearest neighbors are identified, and the class labels of these nearest neighbors are then used to determine the class label of the object. Figure 6 provides a high-level summary of the nearest-neighbor classification method. Given a training set _and a test object Y_ @_._i.O, the algorithm computes the distance (or similarity) between ½and all the training objects @**x**_iO,M_ to determine its nearest-neighbor list, _\. (**x** is the data of a training object, while iis its class. Likewise, **x**.is the data of the test object and i.is its class.) **Input:** _ be the set of _training objects and test object ½Â_ @**x** **Process:** ._i .O
Compute _@**x**._**x**O, the distance between ½and every object, @**x**_iO,._ .

Select _\^]Q_, the set of _closest training objects to ½.

Output: i._argmax _ Ä Á**x**ÖÎRÖÃCÏDa'cb@_d _i_O Figure 6: The _-nearest neighbor classification algorithm. Once the nearest-neighbor list is obtained, the test object is classified based on the majority class of its nearest neighbors: Majority Voting: i._argmax _ >
where dis a class label, ij_ is the class label for the #I254 nearest neighbors, and b@kOis an indicator
function that returns the value 1 if its argument is true and]otherwise.

5.2 Issues

There are several key issues that affect the performance of kNN. One is the choice of k. If k is too small, then the result can be sensitive to noise points. On the other hand, if k is too large, then th neighborhood may include too many points from other classes.

Another issue is the approach to combining the class labels. The simplest method is to take a majority vote, but this can be a problem if the nearest neighbors vary widely in their distance and the closer neighbors more reliably indicate the class of the object. A more sophisticated approach, which is usually much less sensitive to the choice of k, weights each object's vote by its distance, where the weight factor is often taken to be the reciprocal of the squared distance: e___%¹_@**x**.C_**x**_O_. This amounts to replacing Step 5 of Algorithm 6 with the following: Distance-Weighted Voting: i._argmax _ >

The choice of the distance measure is another important consideration. Although various measures can be used to compute the distance between two points, the most desirable distance measure is one for which a smaller distance between two objects implies a greater likelihood of having the same class. Thus, for example, if kNN is being applied to classify documents, then it may be better to use the cosine measure rather than Euclidean distance. Some distance measures can also be affected by the high dimensionality of the data. In particular, it is well known that the Euclidean distance measure become less discriminating as the number of attributes increases. Also, attributes may have to be scaled to prevent distance measures from being dominated by one of the attributes. For example, consider a data set where the height of a person varies from 1.5m to 1.8m, the weight of a person varies from 90lb to 300lb, and the income of a person varies from $10,000 to $1,000,000. If a distance measure is used without scaling, the income attribute will dominate the computation of distance and thus, the assignment of class labels. A number of schemes have been developed that try to compute the weights of each individual attribute based upon a training set [26].

In addition, weights can be assigned to the training objects themselves. This can give more weight to highly reliable training objects, while reducing the impact of unreliable objects. The PEBLS system by by Cost and Salzberg [10] is a well known example of such an approach.

KNN classifiers are lazy learners, that is, models are not built explicitly unlike eager learners (e.g., decision trees, SVM, etc.). Thus, building the model is cheap, but classifying unknown objects is relatively expensive since it requires the computation of the k-nearest neighbors of the object to be labeled. This, in general, requires computing the distance of the unlabeled object to all the objects in the labeled set, which can be expensive particularly for large training sets. A number of techniques have been developed for efficient computation of k-nearest neighbor distance that make use of the structure in the data to avoid having to compute distance to all objects in the training set. These techniques, which are particularly applicable for low dimensional data, can help reduce the computational cost without affecting classification accuracy.

II. Conclusions

Data mining is a broad area that integrates techniques from several fields including machine learning, statistics, pattern recognition, artificial intelligence, and database systems, for the analysis of large volumes of data. There have been a large number of data mining algorithms rooted in these fields to perform different data analysis tasks. The 10 algorithms identified by the IEEE International Conference on Data Mining (ICDM) and presented in this article are among the most influential algorithms for classification, clustering, statistical learning, association analysis, and link mining. We hope this paper can inspire more researchers in data mining to further explore these algorithms, including their impact and new research issues.

References

[1]. Agrawal, R. and Srikant, R. “Fast Algorithms for Mining Association Rules”, Proceedings of the20th VLDB Conference, pages 487-499, 1994.

[2]. A. Banerjee, S. Merugu, I. Dhillon and J. Ghosh. ”Clustering with Bregman Divergences,” Journalof Machine Learning Research (JMLR) Vol.6, 1705-1749. 2005.

[3]. Bezdek, J. C., Chuah, S. K., and Leep, D. 1986. “Generalized k-nearest neighbor rules”. FuzzySets Syst. 18, 3 (Apr. 1986), 237-256. DOI= http://dx.doi.org/10.1016/0165-0114(86)90004-7

[4]. Bloch, D.A. RA Olshen, MG Walker. (2002) “Risk Estimation for Classification Trees”. Journalof Computational & Graphical Statistics, vol 11, 263-288.

[5]. Breiman, L. (1968). Probability Theory. Addison-Wesley, Reading, MA. Republished (1991) inClassics of Mathematics. SIAM, Philadelphia.

[6]. L. Breiman, “Prediction games and arcing classifiers”, Neural Computation, 11(7):1493-1517,1999.

[7]. Breiman, L., Friedman, J. H., Olshen, R. A., and Stone, C. J. (1984). “Classification and RegressionTrees.” Belmont, CA: Wadsworth.

[8]. S. Brin and L. Page. The Anatomy of a Large-Scale Hypertextual Web Search Sngine. ComputerNetworks, 30(1-7), pp. 107-117, 1998.

[9]. Cheung, D. W. and Han, J. and Ng V. and Wong, C. Y.,“Maintenance of Discovered AssociationRules in Large Databases: An Incremental Updating Technique”, Proc. of the ACM SIGMODInternational Conference on Management of Data, pages 13-23, 1996.

Jagannath Ray "Algorithms in Data Mining: Top 5" International Journal of Engineering Science Invention (IJESI), Vol. 04, No. 12, 2015, PP 58-63.