ISSN No. 0976-5697

Volume 4, No. 5, May 2013 (Special Issue)

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

An Efficient Bayesian Classifier in SQL with PCA

DS.Bhupal Naik
Member IEEE, Department Of CSE, Vignan University
Vadlamudi,Guntur,A.P-India
.dsb_cse@vignanuniversity.org

S.Deva kumar
Department Of CSE, Vignan University
Vadlamudi,Guntur,A.P-India
2sdk_cse@vignanuniversity.org

G Sridhar Reddy
Department Of CSE, Vignan University
Vadlamudi,Guntur,A.P-India
sridharreddy_cse@vignanuniversity.org

***Abstract*—** As Bayesian classifier is a fundamental classification technique. We focus on an efficient Bayesian classifier programmed in sql with PCA. We consider three classifiers: Naive Bayes and a classifier based on class decomposition using K-means clustering and Bayesian classifier with dimensionality reduction technique PCA. We consider two complementary tasks: model computation and scoring a data set. We introduce one of the dimensionality reduction techniques, Principal component analysis (PCA) to achieve more accuracy and reduction of storage space. We study several layouts for tables and several indexing alternatives. We analyse how to transform equations into efficient SQL queries. We also analyse how to calculate covariance matrix for PCA using SQL. We perform experiments on wbcancer and bscale datasets to evaluate classification accuracy, query optimization & scalability. Our approach shows improvement in accuracy over the existing approach.

***Keywords*—** Bayesian classification, principal component analysis, covariance matrix, Mean, variance.

I. INTRODUCTION

CLASSIFICATION is a fundamental problem in machine learning and statistics. Bayesian classifiers stand out for their robustness, interpretability, and accuracy. They are deeply related to maximum likelihood estimation and discriminant analysis, highlighting their theoretical importance. In this work, we focus on Bayesian classifiers considering two variants: Naı¨ve Bayes [2] and a Bayesian classifier based on class decomposition using clustering [7].

In this work, we integrate Bayesian classification algorithms into a DBMS. Such integration allows users to directly analyze data sets inside the DBMS and to exploit its extensive capabilities (storage management, querying, concurrency control, fault tolerance, and security). We use SQL queries as the programming mechanism, since it is the standard language in a DBMS. More importantly, using SQL eliminates the need to understand and modify the internal source, which is a difficult task. Unfortunately,

SQL has two important drawbacks: it has limitations to manipulate vectors and matrices and has more overhead than a systems language like C. Keeping those issues in mind; we study how to evaluate mathematical equations with several tables' layouts and optimized SQL queries.

Our contributions are the following: We present two efficient SQL implementations of Naı¨ve Bayes for numeric and discrete attributes. We introduce a classification algorithm that builds one clustering model per class, which is a generalization of K-means [1], [4]. Our main contribution is a Bayesian classifier programmed in SQL, extending Naı¨ve Bayes, which uses K-means to decompose each class into clusters. We generalize queries for clustering adding a new problem dimension. That is, our novel queries combine three dimensions: attribute, cluster, and class subscripts. We identify euclidean distance as the most time-consuming computation. Thus, we introduce several schemes to efficiently compute distance considering different storage layouts for the data set and classification model. We also develop query optimizations that may be applicable to other distance-based algorithms. A horizontal layout of the cluster centroids table and a denormalized table for sufficient statistics are essential to accelerate queries. Past research has shown the main alternatives to integrate data mining algorithms without modifying DBMS source code are SQL queries.

The article is organized as follows: Section 2 introduces definitions. Section 3 introduces two alternative Bayesian classifiers in SQL: Naive Bayes and a Bayesian classifier based on K-means. Section 4 presents an experimental evaluation on accuracy, optimizations, and scalability. Related work is discussed in Section 5. Section 6 concludes the paper.

II. DEFINITION

We focus on computing classification models on a data set X= fx1; . . . ; xng with d attributes X1; . . .;Xd, one discrete attribute G (class or target), and n records (points). We assume G has m = 2 values. Data set X represents a d≤ n matrix, where xi represents a column vector. We study two complementary models: 1) each class is approximated by a normal distribution or histogram and 2) fitting a mixture model with k clusters on each class with K-means. We use subscripts i; j; h; g as follows: i = 1 . . . n; j= 1 . . . k; h = 1 . . . d; g =1 . . .m. The T superscript indicates matrix transposition.

Throughout the paper, we will use a small running example, where d = 4; k = 3 (for K-means) and m = 2 (binary).

III. BAYESIAN CLASSIFIER AND PCA

A. Pre-processing data set

Pre-processing of input data includes, removing noise and null values. To remove the null values in the dataset we use binning, null values are removed by placing mean of

CONFERENCE PAPER
"National Conference on Networks and Soft Computing"
On 25-26 March 2013
Organized by
Vignan University, India

each attribute. And we arrange them in proper order to load the data set into base table.

A data set X ={ x1; . . . ; xn} , g with d attributes X_1; . . .; X_d, one discrete attribute G (class or target), and n records (points). We assume G has m =2 values. Data set X represents a d *n matrix, where x_i represents a column vector.

B. Naïve Bayesian Classification

In data mining, we have two types of Data sets. First, it contains numeric attributes and second, it contains discrete attributes [3]. In this naïve Bayesian, numeric attributes will be improved with class decomposition. NB assumes attributes are independent, and thus, the joint class conditional probability can be estimated as the product of probabilities of each attribute. In this naïve Bayesian classification, it is done based on multivariate Gaussian. NB has no input parameters. Each class is modeled as a single normal distribution with mean vector Cg and a diagonal variance matrix Rg. Scoring assumes a model is available and there exists a data set with the same attributes in order to predict class G.

The model is computed in two passes: a first is to get the mean per class and a second is to compute the variance per class. The mean per class is given by equation 1.1, where Yg is subset of X are the records in class g. Equation 1.2 gives a diagonal variance matrix Rg, which is numerically stable, but requires two passes over the data set.

$$Cg = \sum_{xi \in Yg} \frac{xi}{Ng} \quad \ldots .1.1$$

$$Rg = \frac{1}{Ng} \sum_{xi \in Yg}^{n} (xi - Cg)(xi - Cg)T \quad .....1.2$$

The SQL implementation for numeric NB follows the mean and variance equations introduced above. We compute three aggregations grouping by Cg with two queries. The first query computes the mean Cg of class g with a sum (Xh)/ count (*) aggregation and class priors πg with a count ( ) aggregation. The second query computes Rg with sum (Xh -μh) 2. Here, the joint probability computation is not done. To classify a record, we need to calculate the probabilities. For this scoring we use the Gaussian parameters as input to classify an input point to the most probable class, with one query in one pass over X. Each class probability is evaluated as a Gaussian. To avoid numerical issues when a variance is zero, the probability is set to 1 and the joint probability is computed with a sum of probability logarithms instead of a product of probabilities. A CASE statement pivots probabilities and avoids a max () aggregation. A final query determines the predicted class, being the one with maximum probability, obtained with a CASE statement.

We now discuss NB for discrete attributes. For numeric NB, we used Gaussians because they work well for large data sets and because they are easy to manipulate mathematically. That is, NB does not assume any specific probability density function (pdf). Assume X1; . . .;Xd can be discrete or numeric. If an attribute Xh is discrete (categorical) NB simply computes its histogram: probabilities are derived with counts per value divided by the corresponding number of points in each class. Otherwise, if the attribute Xh is numeric then binning is required.

Table	Content	PK non-Key Columns
XH	Normalized data	i, g,x1,x2,x3,…xd
CH	Centroids	g, C11,C12,C13,…Cdk
XD	Distances	i,g, d1,d2,d3,..dk
XN	Nearest cluster	i,g, j
NLQ	Suff statistics	g,j Ng,L1,L2,…Ld,Q1,Q2,…Qd
WCR	Mixture model	g,j pi,C1,C2,…Cd,R1,R2,..Rd

Binning requires two passes over the data set, pretty much like numeric NB. In the first pass, bin boundaries are determined. On the second pass, one-dimensional frequency histograms are computed on each attribute.

TABLE 1
BKM TABLES

The implementation in SQL of discrete NB is straightforward. For discrete attributes, no pre-processing is required. For numeric attributes, the minimum, maximum, and mean can be determined in one pass in a single query. The variance for all numeric attributes is computed on a second pass to avoid numerical issues. Then, each attribute is discretized finding the interval for each value. Once we have a binned version of X, then we compute histograms on each attribute with SQL aggregations. Probabilities are obtained dividing by the number of records in each class. Scoring requires determining the interval for each attribute value and retrieving its probability. Each class probability is also computed by adding logarithms. NB has an advantage over other classifiers: it can handle a data set with mixed attribute types (i.e., discrete and numerical).

C. Bayesian Classifier Based on K-Means

In this module, first we divide the data set into clusters that means we are making the class decomposition by using k-means clustering then, we apply Bayesian classification. BKM is a generalization of NB; where NB has one cluster per class and the Bayesian classifier has k > 1 clusters per class. We consider two major tasks for BKM:Model computation ,Scoring a data set

The most time-consuming phase is building the model. Scoring is equivalent to an E step from K-means executed on each class. We fit a mixture of k clusters to each class with K-means. The output are priors π (same as NB), mk weights (W), mk centroids (C), and mk variance matrices (R). We generalize NB notation with k clusters [4] per class g with notation g:j, meaning the given vector or matrix refers to j^{th} cluster from class g. Sums are defined over vectors (instead of variables).

Class priors are analogous to NB: We now introduce sufficient statistics in the generalized notation. Let Xg:j represent partition j of points in class g (as determined by K-means). Then, L, the linear sum of points is: $L_{g:j} = \sum_{xi \in Xg:j} x_i$ and the sum of "squared" points for cluster g: j becomes: $Rg = \sum_{xi \in Xg:j} x_i x_i^T$ Based on L; Q, the Gaussian parameters per class g are:

$$Cg:j = \frac{1}{Ng} Lg:j$$

$$Rg = \frac{1}{Ng:j} Qg:j - \frac{1}{Ng:j^2} Lg:j \, Lg:j$$

CONFERENCE PAPER
"National Conference on Networks and Soft Computing"
On 25-26 March 2013
Organized by
Vignan University, India

We generalize K-means to compute m models, fitting a mixture model to each class. K-means is initialized, and then, it iterates until it converges on all classes.

The algorithm is as given below [1]:

Initialization:

1. Get global N; L; Q and σ, μ;
2. Get k random points per class to initialize C.

While not all m models converge:

1. E step: get k distances j per g; find nearest cluster j per g; update N; L; Q per class.
2. M step: update W; C; R from N; L; Q per class; compute model quality per g; monitor convergence.

For scoring, in a similar manner to NB, a point is assigned to the class by choosing highest probability among all the calculated probabilities. The probabilities are calculated based on Gaussian parameter equation. The probability gets multiplied by the class prior by default, but it can be ignored for imbalanced classification problems. We now study how to create an efficient implementation of BKM in SQL.

In the following table 3.1, the tables for NB are extended with the j subscript and since there is a clustering algorithm behind, we introduce tables for distance computation. There is a single set of tables for the classifier.

All m models are updated on the same table scan. This eliminates the need to create multiple temporary tables. We introduce a fast distance computation mechanism based on a flattened version of the Centroids; temporary tables have fewer rows. We use a CASE statement to get closest cluster avoiding the use of standard SQL aggregations. A join between two large tables is avoided. We delete points from classes whose model has converged. The algorithm works with smaller tables as models converge.

1. Model Computation

Model computation is sub module in the Bayesian classification; in this sub module, we will build the model which results the prior probabilities, mean of each cluster and variance of each cluster, for this, we need to initialize the CH table with k random points per class; In this manner, K-means computes Euclidean distance with all attributes being on the same relative scale. The normalized data set is stored in table XH, which also has n rows.

We consider a horizontal scheme to compute distances. All k distances per class are computed as SELECT terms. This produces a temporary table with mkn rows. Then, the temporary table is pivoted and aggregated to produce a table having mn rows with k columns. Such a layout enables efficient computation of the nearest cluster in a single table scan. The SQL code pairs each attribute from X with the corresponding attribute from the cluster. Computation is efficient because CH has only m rows and the join computes a table with n rows (for building the model) or mn rows for scoring.

The following SQL statement computes k distances for each point, corresponding to the g^{th} model. This statement is also used for scoring, and therefore, it is convenient to include g.

```
INSERT INTO XD SELECT i,XH.g ,(X1-C1_X1)**2 + .. +(X4-C1_X4)**2, ..,(X1-C3_X1)**2 + .. +(X4-C3_X4)**2 FROM XH,CH WHERE XH.g=CH.g;
```

At this point, we have computed k distances per class and we need to determine the closest cluster. There are two basic alternatives: pivoting distances and using SQL standard aggregations. Using case statements, we determine the minimum distance. For the first alternative, XD must be pivoted into a bigger table. Then, the minimum distance is determined using the min() aggregation. The closest cluster is the subscript of the minimum distance, which is determined by joining XH [1].

In the second alternative, we just need to compare every distance against the rest using a CASE statement. Since the second alternative does not use joins and is based on a single table scan, it is much faster than using a pivoted version of XD. The SQL to determine closest cluster per class is given as an example below. This statement can also be used for scoring.

```
INSERT INTO XN SELECT i,g,CASE WHEN d1<=d2 AND d1<=d3 THEN 1 d2<=d3 THEN 2 ELSE 3 END AS j FROM XD;
```

Once we have determined the closest cluster for each point on each class, we need to update sufficient statistics, which are just sums. This computation requires joining XH and XN and partitioning points by class and cluster number. Since table NLQ is denormalized, the join computation is demanding for joining two tables with rows. For BKM, it is unlikely that, variance can have numerical issues. But, it is feasible. In such a case, sufficient statistics are substituted by a two-pass computation like NB.

```
INSERT INTO NLQ SELECT XH.g,j,sum(1.0) as Ng                    /*N*/
,sum(X1) , .. ,sum(X4)                                          /* L */
Sum(X!**2) , .. ,sum(X4**2)          /* Q*/
FROM XH,XN WHERE XH.i=XN.i GROUP BY XH.g,j;
```

We now discuss the M step to update W; C; R. Computing WCR from NLQ is straightforward since both tables have the same structure and they are small. There is a Cartesian product between NLQ and the model quality table [1]. The latter table has only one row. Finally, to speed up computations, we delete points from XH for classes whose model has converged: a reduction in Distance Computation size is propagated to the nearest cluster along with sufficient statistics queries.

```
INSERT INTO WCR SELECT
NLQ.g,NLQ.j ,Ng/MODEL.n                 /* pi */
,L1/Ng, ..,L4/Ng                        /* C */
,Q1/Ng-(L1/Ng)**2,..,Q4/Ng-(L4/Ng)**2 /*R */
 FROM NLQ,MODEL WHERE NLQ.g=MODEL.g;
```

2 Scoring

Scoring is sub module in the Bayesian classification; in this submodule we will make the classification. In this module, we consider two alternatives: based on probability (default) or distance. Scoring is similar to the E step. But, there are differences. First, the new data set must be normalized with the original variance used to build the model. Such variance should be similar to the variance of the new data set. Second, we need to compute mk probabilities (distances) instead of only k distances because we are trying to find the most probable (closest) cluster among all. Thus, the join condition between XH and CH gets eliminated. Third, once we have mk probabilities (distances), we need to pick the maximum (minimum) one

CONFERENCE PAPER
"National Conference on Networks and Soft Computing"
On 25-26 March 2013
Organized by
Vignan University, India

and then the point is assigned to the corresponding cluster [1]. To have a more elegant implementation, we predict the class in two stages.

First, we determine the most probable cluster per class, and then, compare the probabilities of such clusters. Column XH.g is not used for scoring purposes, but it is used to measure accuracy, when known.

D. Principal component analysis (PCA)

In this module, PCA is mainly used to reduce the dimensionality of the data set by performing a covariance analysis between factors. PCA allows us to compute a linear transformation that maps data from a high dimensional space to a lower dimensional space. Suppose that, the data to be reduced consist of tuples or data vectors described by n attributes or dimensions [7].

In this PCA, we are using the following table to store the values and these tables will behave as a input for the next tables and from these we are computing the covariance matrix. To find the covariance matrix we use the Sql queries.

Table	Content	PK non-Key Columns
XH	Normalized data	i, g,x1,x2,x3,…xd
NL	Centroids	g, L1,L2,L3….Ld
MEAN	Distances	i,g, M1,M2,M3,….Md
AFD	Nearest cluster	i,g, X!,X2,X3,…..Xd
COVAR	Covariancematrix	g X!,X2,X3,…..Xd

TABLE 2
PCA TABLES

The first table is the base table to our application, In that we have sample data in the form of attribute. It should be in the numeric form. To calculate the linear some of each attribute of the dataset we will the following query. We are computing each attribute sum by using sum ().the result is storing in the NL table for future purpose [1].

INSERT INTO NL SELECT XH.g,j,sum(1.0) as Ng,sum(X1) , .. , sum(X4), WHERE XH GROUP BY g;

After finding linear sum of each attribute, we have to calculate the mean of each attribute .It is very easy process. It can be calculated in single step. We have linear sum (Ld) and Table number of record (Ng) in each attribute in NL Table, by using the following query, we will insert mean of each attribute into Mean table

INSERT INTO MEAN SELECT NL.g,Ng ,L1/Ng, ..,L4/Ng , FROM NL ;

Once we have mean of each attribute, we need to subtract each vale from mean, to do that process, we use MEAN table and base table (XH), from these two tables we calculate the difference of mean and each value and we insert those values into AFD table by using single query.

INSERT INTO AFD SELECT X1-M1,X2-M2,.., FROM XH,MEAN WHERE XH.G=MEAN.G;

From above SQL query we find difference of mean and each value in base table. Next step is, we need to calculate the covariance matrix [7], by using the following query we insert in to COVARIANCE Table.

INSERT INTO COVARIANCE SELECT SUM(X1*X1)/NG,SUM(X1*X2),……. SUM(X1*XD …SUM (XD*X1), SUM(XD*X2) ,…….. SUM(XD*XD) FROM PCAAFM;

Now, we get covariance matrix, from the covariance matrix, we have to calculate Eigen values and Eigen vectors to calculate principal components. In Sql, it is very difficult process. We studied different methods like QR, LU decomposition iteration methods. To optimize that process, we select direct method, i.e. eig () in mat lab. From eig () we can calculate eigen values and eigen vectors easily.

IV EXPERIMENTAL EVALUATION

We analyze three major aspects: 1) classification accuracy, 2) query optimization, and 3) time complexity and speed. We compare the accuracy of NB, BKM, and decision trees (DTs).

A. Setup

We used the Teradata DBMS running on a server with a 3.2 GHz CPU, 2 GB of RAM, and a 750 GB disk. Parameters were set as follows: We set _ ¼ 0:001 for K-means.

The number of clusters per class was k ¼ 4 (setting experimentally justified). All query optimizations were turned on by default (they do not affect model accuracy). Experiments with DTs were performed using a data mining tool.

We used real data sets to test classification accuracy (from the UCI repository) and synthetic data sets to analyze speed (varying d; n). Real data sets include pima (d ¼ 6; n ¼ 768), spam (d ¼ 7; n ¼ 4;601), bscale (d ¼ 4; n ¼ 625), and wbcancer (d ¼ 7; n ¼ 569). Categorical attributes (_3 values) were transformed into binary attributes.

B. Model Accuracy

We used the system with a configuration 3.2 GHz CPU, 1 GB of RAM, and a 120 GB disk. We worked on an oracle Express edition 10g database. We used real data sets to test classification accuracy (from the UCI repository) and synthetic data sets to analyze speed (varying d; n). Real data sets include bscale (n=4, d=625), and wbcancer (d=7; n=569). Categorical attributes (>=3 values) were transformed into binary attributes.

TABLE 3

ACCURACY BY VARYING K (NUMBER OF CLUSTERS) IN BKM.

Data set	k=2	K=3	k=4	k=6	k=8	k=16
Bscale	55%	57%	59%	56%	60%	68%
wbcancer	93%	94%	92%	93%	93%	89%

We have concentrated mainly on accuracy; we measure the accuracy of predictions when using Bayesian classification models. For each run, the data set was partitioned into a training set and a test set. The training set was used to compute the model, whereas the test set was used to independently measure accuracy. The training set size was 80 percent and the test set was 20 percent.

The number of clusters (k) is the most important parameter to tune BKM accuracy. The Table 5.1 shows accuracy behaviour as k increases. As size of k increases, it

CONFERENCE PAPER
"National Conference on Networks and Soft Computing"
On 25-26 March 2013
Organized by
Vignan University, India

produces more complex models. Intuitively, a higher k should achieve higher accuracy because each class can be better approximated with localized clusters.

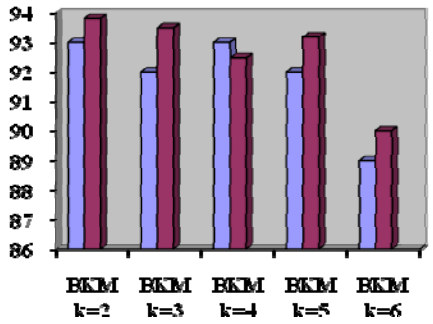

Fig 1: comparing accuracy of BKM after PCA, before PCA by varying k

On the other hand, a higher k than necessary can lead to empty clusters. As can be seen, a lower k produces better models for spam and wbcancer, whereas a higher k produces higher accuracy for bscale. Therefore, there is no ideal setting for k, but, in general, k <=20 produce good results. Based on these results, we set k=3 by default since it provides reasonable accuracy for all data sets.

In the following Table5.2, we compare the accuracy of the three models, including the overall accuracy as well as a breakdown per class. Accuracy per predicted class is important to understand issues with imbalanced classes and detecting subsets of highly similar points.

TABLE 4
ACCURACY BY VARYING K VALUE FOR NB, BKM, AND BKM AFTER PCA.

	Method	Global	Class=0	Class=1
Bscale	NB	50%	51%	30%
	BKM	59%	59%	60%
	BKM After PCA	60%	61%	58%
wbcancer	NB	93%	91%	95%
	BKM	93%	84%	97%
	BKM After PCA	94%	86%	98%

In practice, one class is generally more important than the other one (asymmetric), depending on whether the classification task is to minimize false positives or false negatives. This is a summary of findings. First of all, considering global accuracy, and next by considering individual class.

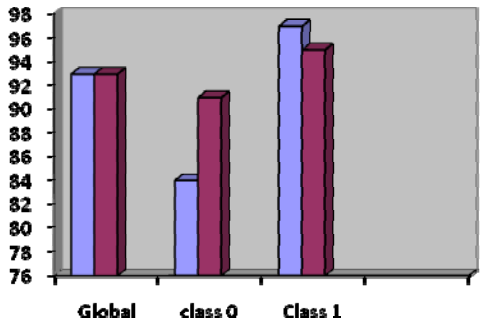

Fig 2: comparing accuracy of NB, BKM.

We applied PCA for wbcancer data set and removed six attributes (6, 13, 14, 15, 19, 23, and 21) because these are showing poor Eigen values and after removing these attributes from data set we again applied the BKM and compute the results.

V. CONCLUSIONS

We presented three Bayesian classifiers programmed in SQL: the Naive Bayes classifier (with discrete and numeric versions), a generalization of Naive Bayes (BKM), based on decomposing classes with K-means clustering and BKM with dimensionality reduction technique PCA. And we studied two complementary aspects: increasing accuracy and generating efficient SQL code. We presented one dimensionality reduction technique Principal component analysis (PCA) to produce more accurate results in classification. The nearest cluster per class, required by K-means, is efficiently determined avoiding joins and aggregations. Experiments with real data sets are performed and compared the results of NB, BKM, PCA, BKM after PCA, and BKM before PCA. The numeric and discrete versions of NB had similar accuracy. BKM was more accurate than NB in global accuracy. However, BKM was more accurate when computing a breakdown of accuracy per class. It was observed that, when the number of clusters formed is less, it produced a better result. After applying PCA on the dataset, as a result it reduced the number of attributes in the dataset and given better result in terms of accuracy.

NB and BKM exhibited linear scalability in data set size and dimensionality. There is much scope for future work. We can derive incremental versions or sample-based methods to accelerate the Bayesian classifier. We are interested in combining one more dimensionality reduction technique factor analysis with Bayesian classifiers in SQL.

VI. REFERENCES

[1] Carlos Ordonez and Sasi K. Pitchaimalai," Bayesian Classifiers Programmed in SQL,"Proc. IEEE Transactions On Knowledge And Data Engineering,VOL. 22, NO. 1, January 2010.

[2] P. Bradley, U. Fayyad, and C. Reina, "Scaling Clustering Algorithms toLarge Databases," Proc. ACM Knowledge Discovery and Data Mining (KDD)Conf., pp. 9-15, 1998.

[3] B.L. Milenova and M.M. Campos, "O-Cluster: Scalable Clustering of Large High Dimensional Data Sets," Proc. IEEE Int'l Conf. Data Mining (ICDM), pp. 290-297, 2002.

[4] Tapas Kanungo, Senior Member, IEEE, David M. Mount, Member, IEEE, Nathan S. Netanyahu, Member, IEEE, Christine D. Piatko, Ruth Silverman, and Angela Y. Wu, Senior Member, IEEE, "An Efficient k-Means Clustering Algorithm: Analysis and Implementation ," IEEE Trans. Knowledge and Data Eng., vol. 18, no. 2, pp. 188-201, Feb. 2006.

[5] R. Vilalta and I. Rish, "A Decomposition of Classes via Clustering to Explain and Improve Naive Bayes," Proc. European Conf. Machine Learning (ECML), pp. 444-455, 2003.

[6] Thair Nu Phyu, "Survey of Classification Techniques in Data Mining",International MultiConference of Engineers and Computer Scientists 2009, Vol I IMECS 2009, March 18 - 20, 2009,

[7] D.Napoleon S.Pavalakodi, "A New Method for Dimensionality Reduction Using K-Means Clustering Algorithm for High Dimensional Data Set ".International Journal of Computer Applications (0975 – 8887) Volume 13– No.7, January

CONFERENCE PAPER
"National Conference on Networks and Soft Computing"
On 25-26 March 2013
Organized by
Vignan University, India