Global Journal of Computer Science and Technology
Software & Data Engineering
Volume 12 Issue 13 Version 1.0 Year 2012
Type: Double Blind Peer Reviewed International Research Journal
Publisher: Global Journals Inc. (USA)
Online ISSN: 0975-4172 & Print ISSN: 0975-4350

Bayesian Classifiers Programmed in SQL Using PCA

By K.Venkat Nagarjuna & P.V Subba Reddy
QIS College of Engg & Technology Ongole, Andhrapradesh, India

Abstract - The Bayesian classifier is a fundamental classification technique. We also consider different concepts regarding Dimensionality Reduction techniques for retrieving lossless data. In this paper, we proposed a new architecture for pre-processing the data. Here we improved our Bayesian classifier to produce more accurate models with skewed distributions, data sets with missing information, and subsets of points having significant overlap with each other, which are known issues for clustering algorithms. so, we are interested in combining Dimensionality Reduction technique like PCA with Bayesian Classifiers to accelerate computations and evaluate complex mathematical equations. The proposed architecture in this project contains the following stages: pre-processing of input data, Naïve Bayesian classifier, Bayesian classifier, Principal component analysis, and database. Principal Component Analysis(PCA) is the process of reducing components by calculating Eigen values and Eigen Vectors. We consider two algorithms in this paper: Bayesian Classifier based on KMeans(BKM) and Naïve Bayesian Classifier Algorithm(NB).

Keywords : *Dimensionality Reduction, PCA, Classifiers, K-means.*

GJCST-C Classification: *H.2.3*

Strictly as per the compliance and regulations of:

Bayesian Classifiers Programmed in SQL Using PCA

K.Venkat Nagarjuna[α] & P.V Subba Reddy[σ]

Abstract - The Bayesian classifier is a fundamental classification technique. We also consider different concepts regarding Dimensionality Reduction techniques for retrieving lossless data. In this paper, we proposed a new architecture for pre-processing the data. Here we improved our Bayesian classifier to produce more accurate models with skewed distributions, data sets with missing information, and subsets of points having significant overlap with each other, which are known issues for clustering algorithms. so, we are interested in combining Dimensionality Reduction technique like PCA with Bayesian Classifiers to accelerate computations and evaluate complex mathematical equations. The proposed architecture in this project contains the following stages: pre-processing of input data, Naïve Bayesian classifier, Bayesian classifier, Principal component analysis, and database. Principal Component Analysis(PCA) is the process of reducing components by calculating Eigen values and Eigen Vectors. We consider two algorithms in this paper: Bayesian Classifier based on KMeans(BKM) and Naïve Bayesian Classifier Algorithm(NB).

Keywords : Dimensionality Reduction, PCA, Classifiers, K-means.

I. Introduction

In this paper, we focus on programming Bayesian classifiers in SQL using Principal Component Analysis(PCA). PCA allows us to compute a linear transformation that maps data from a high dimensional space to a lower dimensional space.PCA "combines" the essence of attributes by creating an alternative, smaller set of variables. In this paper, We studied two complementary aspects: increasing accuracy and generating efficient SQL code. We introduce two classifiers: Naive Bayes and a classifier based on class decomposition using K-means clustering. We consider two complementary tasks: model computation and scoring a data set. We study several layouts for tables and several indexing alternatives. We analyse how to transform equations into efficient SQL queries and introduce several query optimizations.

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.

Author α : M.tech student, Dept of CSE, QIS College of Engg & Technology Ongole, Andhrapradesh, India.
Author σ : Associate Professor, Dept of CSE, QIS College of Engg & Technology, Ongole, Andhrapradesh, India.

II. Definitions

We focus on computing classification models on a data set X={x1 . . . ; xn} with d attributes X1. . . ..Xd, one discrete attribute G(class or target), and n records (points). We assume G has m=2values. Data set X represents a d x 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 Classifiers Programmed in SQL using Principal Component Analysis

a) Classification

Classification and prediction are two forms of data analysis that can be used to extract models describing important data classes or to predict future data trends. Such analysis can help provide us with a better understanding of the data at large. Whereas classification predicts categorical (discrete, unordered) labels, prediction models continuous valued functions.

b) Bayesian Classification

The Bayesian Classification represents a supervised learning method as well as a statistical method for classification. It can solve diagnostic and predictive problems.

This Classification is named after Thomas Bayes (1702-1761), who proposed the Bayes Theorem. Bayesian classification provides practical learning algorithms and prior knowledge and observed data can be combined. Bayesian Classification provides a useful

perspective for understanding and evaluating many learning algorithms. It calculates explicit probabilities for hypothesis and it is robust to noise in input data. Bayesian classification is based on Bayes' theorem. A simple Bayesian classifier is known as the naïve Bayesian classifier. Bayesian classifiers have exhibited high accuracy and speed when applied to large databases.

c) Naïve Bayesian Classification

It is based on the Bayesian theorem It is particularly suited when the dimensionality of the inputs is high. Parameter estimation for naive Bayes models uses the method of maximum likelihood. In spite over-simplified assumptions, it often performs better in many complex real world situations

The main advantage of Naïve Bayesian classification is it requires a small amount of training data to estimate the parameters.

d) Data Reduction

Data reduction techniques can be applied to obtain a reduced representation of the data set that is much smaller in volume, yet closely maintains the integrity of the original data.

Strategies for data reduction include the following:

Data cube aggregation
Attribute subset selection
Dimensionality reduction
Numerosity reduction
Discretization and concept hierarchy generation

e) Dimensionality reduction

Data encoding or transformations are applied so as to obtain a reduced or "compressed" representation of the original data. If the original data can be reconstructed from the compressed data without any loss of information, the data reduction is called lossless. If, instead, we can reconstruct only an approximation of the original data, then the data reduction is called lossy. Although they are typically lossless, they allow only limited manipulation of the data. In this section, we instead focus on two popular and effective methods of lossy dimensionality reduction: wavelet transforms and principal components analysis.

f) Principal Components Analysis

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. Principal components analysis, or PCA (also called the Karhunen-Loeve, or K-L, method), searches for k n dimensional orthogonal vectors that can best be used to represent the data, where k _ n. The original data are thus projected onto a much smaller space, resulting in dimensionality reduction. Unlike attribute subset selection, which reduces the attribute set size by retaining a subset of the initial set of attributes, PCA "combines" the essence of attributes by creating an alternative, smaller set of variables. The initial data can then be projected onto this smaller set.

g) Methodology

Suppose $x_1, x_2, ..., x_M$ are Nx1 vectors

Step 1: $x = \frac{1}{M}\sum_{i=1}^{M} xi$

Step 2: subtract the mean: $F_i = x_i - x$

Step 3: form the matrix $A = [F_1 F_2 \ldots F_M]$ (NxM matrix), then compute:

$$C = = AA^T \quad \frac{1}{M}\sum_{i=0}^{M} FnFnT$$

(Sample covariance matrix, NxN, characterizes the scatter of the data)

Step 4: compute the Eigen values of C: $\lambda_1 > \lambda_2 > \ldots > \lambda_N$

Step 5: compute the eigenvectors of C: $u_1, u_2, \ldots, u_N$

Since C is symmetric, $u_1, u_2, \ldots, u_N$ form a basis, (i.e., any vector x or actually $(x - \bar{x})$, can be written as a linear combination of the eigenvectors):

$$x - \bar{x} = b_1u_1 + b_2u_2 + \ldots + b_N u_N = \sum_{l=1}^{N} biui$$

Step 6: (dimensionality reduction step) keep only the terms corresponding to the K largest Eigen values: $\hat{x} - \bar{x} =$

$$\sum_{i=1}^{k} biui$$

where K << N.

These are the steps we should follow to perform principal component analysis (PCA) to reduce dimensionality of the high dimensional data.

h) Naı¨ve Bayes

We consider two versions of NB: one for numeric attributes and another for discrete attributes. Numeric NB 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 [2]. We now discuss NB based on a multivariate Gaussian. NB has no input parameters. Each class is modelled 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. Variance computation based on sufficient statistics [5] in one pass can be numerically unstable when the variance is much smaller compared to large attribute values or when the data set has a mix of very large and very small numbers. For ill conditioned data sets, the computed variance can be significantly different from the actual

variance or even become negative. Therefore, the model is computed in two passes: a first pass to get the mean per class and a second one to compute the variance per class. The mean per class is given by Cg = Pxi2Yg xi=Ng, where Yg _ X are the records in class g. Equation Rg = 1=NgPn xi2Ygðxi _CgÞðxi _ CgÞT gives a diagonal variance matrix Rg, which is numerically stable, but requires two passes over the data set.

The SQL implementation for numeric NB follows the mean and variance equations introduced above. We compute three aggregations grouping by g with two queries. The first query computes the mean Cg of class g with a sum ∂XhÞ=count ∂_Þ aggregation and class priors _g with a count() aggregation. The second query computes Rg with sum(∂Xh _ _hÞ2Þ. Note the joint probability computation is not done in this phase. Scoring uses 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. 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.

The bin boundaries (interval ranges) may impact the accuracy of NB due to skewed distributions or extreme values. Thus, we consider two techniques to bin attributes: 1) creating k uniform intervals between min and max and 2) taking intervals around the mean based on multiples of the standard deviation, thus getting more evenly populated bins. We do not study other binning schemes such as quantiles (i.e., equidepth binning).The implementation in SQL of discrete NB is Straight forward. 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).be in single-column format and must be centered.

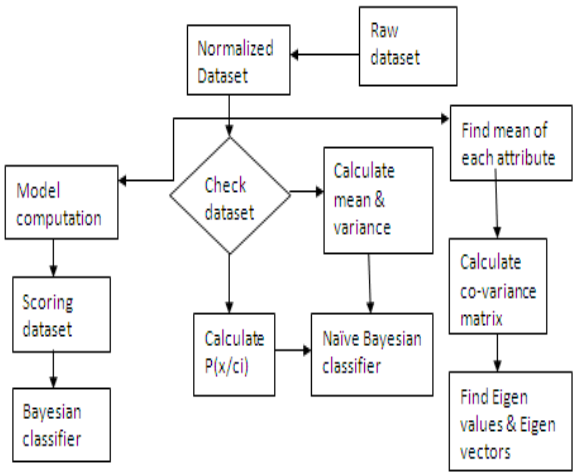

Figure 3 : Architecture of the proposed method

IV. Algorithms

We consider two algorithms in this paper: Bayesian Classifier Based on K-Means(BKM) algorithm and Naïve Bayesian classifier algorithm(NB).

a) Naïve Bayesian classifier algorithm(NB)

NB has no input parameters. Each class is modeled as a single normal distribution with mean vector (Cg) and a diagonal variance matrix (Rg). Therefore, the model is computed in two passes:

1. A first pass to get the mean per class and
2. Second one to compute the variance per class.

b) Bayesian Classifier Based on K-Means

We now present BKM, a Bayesian classifier based on class decomposition obtained with the K-means algorithm. BKM is a generalization of NB, where NB has one cluster per class and the Bayesian classifier has k > 1 clusters per class. L, the linear sum of points; L,Q, the Gaussian parameters.

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.
Initialization:

1. Get global N; L;Q and mean and standard deviation
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.

V. 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 €= 0:001 for K-means. The number of clusters per class was k=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 = 6; n = 768), spam (d = 7; n = 4.601), bscale (d = 4; n = 625), and wbcancer (d =7; n = 569). Categorical attributes (≥3 values) were transformed into binary attributes.

b) Model Accuracy

In this section, we measure the accuracy of predictions when using Bayesian classification models. We used 5-fold cross validation 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.

Comparing Accuracy: NB, BKM, and DT

Dataset	Algorithm	Global	Class-0	Class-1
pima	NB	76%	80%	68%
	BKM	76%	87%	53%
	DT	68%	76%	53%
spam	NB	70%	87%	45%
	BKM	73%	91%	43%
	DT	80%	85%	72%
bscale	NB	50%	51%	30%
	BKM	59%	59%	60%
	DT	89%	96%	0%
wbcancer	NB	93%	91%	95%
	BKM	93%	84%	97%
	DT	95%	94%	96%

BKM ran until K-means converged on all classes. Decision trees used the CN5.0 algorithm splitting nodes until they reached a minimum percentage of purity or became too small. Pruning was applied to reduce over fit. The number of clusters for BKM by default was k = 4.

c) Query Optimization

Our best distance strategy is two orders of magnitude faster than the worst strategy and it is one order of magnitude faster than its closest rival. The explanation is that I/O is minimized to n operations and computations happen in main memory for each row through SQL arithmetic expressions. Note a standard aggregation on the pivoted version of X (XV) is faster than the horizontal nested query variant .Table 6 compares SQL with UDFs to score the data set(computing distance and finding nearest cluster per class). We exploit scalar UDFs [5]. Since finding the nearest cluster is straight forward in the UDF, this comparison considers both computations as one. This experiment favors the UDF since SQL requires accessing large tables in separate queries. As we can see, SQL (with arithmetic expressions) turned out be faster than the UDF. This was interesting because both approaches used the same table as input and performed the same I/O reading a large table. We expected SQL to be slower because it required a join and XH and XD were accessed. The explanation was that the UDF has overhead to pass each point and model as parameters in each call.

Query Optimization: Distance Computation $n = 100$k (Seconds)

Distance scheme	$d = 8$ $k = 8$	$d = 8$ $k = 16$	$d = 16$ $k = 8$	$d = 16$ $k = 16$
Horizontal	2	7	2	7
Horizontal temp	80	211	99	225
Horizontal nested q.	201	449	311	544
Hybrid Vertical	84	155	146	155

Query Optimization: SQL versus UDF (Scoring, $n = 1$M) (Seconds)

k	$d = 4$ SQL	$d = 4$ UDF	$d = 8$ SQL	$d = 8$ UDF
2	23	36	34	61
4	31	47	42	71
6	40	55	61	85
8	56	62	66	111

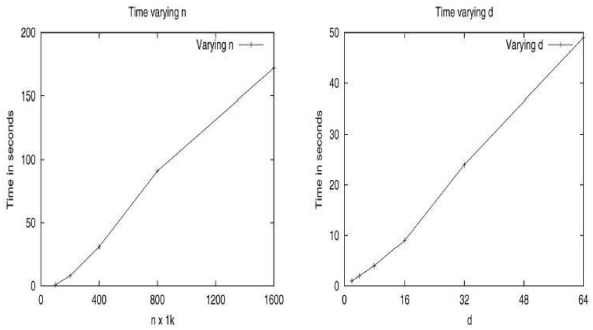

Fig. 1 : BKM Classifier: Time Complexity; (default d=4, k=4, n=100k)

d) Speed and Time Complexity

We compare SQL and C++ running on the same computer. We also comp are the time to export with ODBC. C++ worked on flat files exported from the DBMS. We used binary files in order to get maximum performance in C++. Also, we shut down the DBMS when C++ was running. In short, we conducted a fair comparison. Table 7 compares SQL, C++, and ODBC varying n. Clearly, ODBC is a bottleneck. Overall, both languages scale linearly. We can see SQL performs better as n grows because DBMS overhead becomes less important. However, C++ is about four times faster. Fig. 1 shows BKM time complexity varying n; d with large datasets. Time is measured for one iteration. BKM is linear in n and d , highlighting its scalability.

VI. Related Work

The most widely used approach to integrate data mining Algorithms into a DBMS is to modify the internal source code. Scalable K-means (SKM) [1] and O-cluster [3] are two examples of clustering algorithms internally integrated with a DBMS. A discrete Naive Bayes classifier has been internally integrated with the SQL Server DBMS. On the other hand, the two main mechanisms to integrate data mining algorithms without modifying the DBMS source code are SQL queries and UDFs. A discrete Naı¨ve Bayes classifier programmed in SQL is introduced in [6]. We summarize differences with ours. The data set is assumed to have discrete attributes: binning numeric attributes is not considered. Ituses an inefficient large pivoted intermediate table, whereas our discrete NB model can directly work on a horizontal layout. Our proposal extends clustering algorithms in SQL [4] to perform classification, generalizing Naı¨ve Bayes [2]. K-means clustering was programmed with SQL queries introducing three variants [4]: standard, optimized, and incremental. We generalized the optimized variant. Note that classification represents a significantly harder problem than clustering. User-Defined Functions are identified as an important extensibility mechanism to integrate data mining algorithms [5], [8]. Atlas [8] extends SQL syntax with object-oriented constructs to define aggregate and table functions (with initialize, iterate, and terminate clauses), providing a user friendly interface to the SQL standard. We point out several differences with our work. First, we propose to generate SQL code from a host language, thus achieving Turing-completeness in SQL (similar to embedded SQL). We showed the Bayesian classifier can be solved more efficiently with SQL queries than with UDFs. Even further, SQL code provides better portability and Atlas requires modifying the DBMS source code. Class decomposition with clustering is shown to improve NB accuracy [7]. The classifier can adapt to skewed distributions and overlapping subsets of points by building better local models. In [7], EM was the algorithm to fit a mixture per class. Instead, we decided to use K-means because it is faster, simpler, better understood in database research and has less numeric issues.

VII. Conclusions

We presented two Bayesian classifiers programmed in SQL: the Naı¨ve Bayes classifier (with discrete and numeric Versions) and a generalization of Naı¨ve Bayes (BKM), based on decomposing classes with K-means clustering. We studied two complementary aspects: increasing accuracy and generating efficient SQL code. We introduced query optimizations to generate fast SQL code. The best physical storage layout and primary index for large tables is based on the point subscript. Sufficient statistics are stored on denormalized tables. The Euclidean distance computation uses a flattened (horizontal) version of the cluster centroids matrix, which enables arithmetic expressions. The nearest cluster per class, required by Kmeans, is efficiently determined avoiding joins and aggregations. Experiments with real data sets compared NB, BKM, and decision trees. The numeric and discrete versions of NB had similar accuracy.BKM was more accurate than NB and was similar to decision trees in global accuracy. However, BKM was more accurate when computing a breakdown of accuracy per class. A low number of clusters

produced good results in most cases. We compared Equivalent implementations of NB in SQL and C++ with large data sets: SQL was four times slower. SQL queries were faster than UDFs to score, highlighting the importance of our optimizations. NB and BKM exhibited linear scalability in data set size and dimensionality. There are many opportunities for future work. We want to derive incremental versions or sample-based methods to accelerate the Bayesian classifier. We want to improve our Bayesian classifier to produce more accurate models with skewed distributions, data sets with missing information, and subsets of points having significant overlap with each other, which are known issues for clustering algorithms. We are interested in combining dimensionality reduction techniques like PCA or factor analysis with Bayesian classifiers. UDFs need further study to accelerate computations and evaluate complex mathematical equations.

References Références Referencias

1. P. Bradley, U. Fayyad, and C. Reina, "Scaling Clustering Algorithms to Large Databases," Proc. ACM Knowledge Discovery and Data Mining (KDD) Conf., pp. 9-15, 1998.
2. T. Hastie, R. Tibshirani, and J.H. Friedman, The Elements of Statistical Learning, first ed. Springer, 2001.
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. C. Ordonez, "Integrating K-Means Clustering with a Relational DBMS Using SQL," IEEE Trans. Knowledge and Data Eng., vol. 18, no. 2, pp. 188-201, Feb. 2006.
5. C. Ordonez, "Building Statistical Models and Scoring with UDFs," Proc. ACM SIGMOD, pp. 1005-1016, 2007.
6. S. Thomas and M.M. Campos, SQL-Based Naive Bayes Model Building and Scoring, US Patent 7,051,037, US Patent and Trade Office, 2006.
7. 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.
8. H. Wang, C. Zaniolo, and C.R. Luo, "ATLaS: A Small but Complete SQLExtension for Data Mining and Data Streams "