Naive Bayes Classifiers Programmed in Query Language

[1]Y.V. Siddartha Reddy, [2]Dr.Supreethi K.P

[1]Student M.Tech, [2]Assistant Professor,

JNTU Hyderabad,

yisddarhareddy@gmail.com, supreethi.pujari@gmail.com

Abstract: *For classifications the fundamental technique is "Bayesian Classifier". However, it has been used programmatically. This paper throws light on programming Bayesian classifiers in query Language. Two classifiers are introduced in this paper. They are a classifier which uses K-means clustering based on class decomposition and Naïve Bayes Classifier. Scoring a data set and model computation are two tasks associated to those classifiers. The transformations of equations into query language queries are described including many query optimizations. The experiments are done to evaluate scalability, query optimizations and classification accuracy. The results revealed that the proposed Bayesian Classifier is more accurate when compared to Decision Trees and Naïve Bayes. Pivoting, renormalization, and horizontal layout of tables accelerated distance computation. Other observation is that query language implementation of Naïve Bayes is 4 times slower than its C++ counterpart. However, Bayesian classifier in SQL bestows high classification accuracy as it has linear scalability, and can effectively analyze large data sets.*

Index Terms: *Query optimization, k-means, and classification*

1. Introduction

In statistics and machine learning “classification” is a fundamental problem [2]. The classifiers known for accuracy, interoperability and robustness are Bayesian Classifiers [2]. In this work two variant of Bayesian Classifiers are considered. They are a Bayesian classifier that uses k-means clustering [7] and Naïve Bayes [2]. Classification algorithms are integrated with a DBMS. With this direct analysis of data is possible by exploiting DBMS capabilities. The database features that could be exploited include security, fault tolerance, concurrency control, and querying and storage management. Using SQL the tasks become easy. However, there are two drawbacks with query language.

Our Contributions

The contributions of this paper are as given below.

For numeric and discrete attributes two effective Naïve Bayes implementations are presented. 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 naïve bNayes classifier programmed in SQL, extending Naý¨ve Bayes, which uses K-means to decompose each class into clusters. We generalize queries for clustering adding a new problem dimension. 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.

2. Definitions

Two complementary models are studied. 1) A histogram or normal distribution is used to approximate each class. 2) On each class with K-means fitting a mixture model with k clusters. The following sub sections provide information about all classifiers.

3. Naive Bayes Classifiers Programmed in Query Language

Two classifiers are introduced and two approaches are introduced. The first classifier is NB (Naïve Bayes) and the second classifier is BKM (Bayesian classifier based on K-means). The approaches considered are computing the model and scoring the data set based on the model. Optimizations are done without modifying the model accuracy.

3.1 Naive Bayes

Two versions of NB is considered. One for descrete attributes while the other one is for numeric attributes. Class decomposition is used to improve numeric NB. As per NB assumption, attributes are independent. Therefore the product of probabilities of each attribute is used to estimate the joint class conditional probability [2]. Based on Gaussian, the NB is now discussed. No input parameters are required by NB. With diagonal variance matrix (Rg) and mean vector (Cg), each class is modeled as a single normal distribution. In order to predict class (G), scoring assumes. (i) A model is available (ii) there exists a same data set.

When data set is very large, there are some problems. When the variance is much smaller compared to large attribute values or when data set has mix of very large and very small numbers, variance computation based on sufficient statistics [5] in one pass is numerically unstable. Computed variance can be significantly different or even become negative for ill-conditioned data sets. The first pass is used to obtain the mean per class while the second class is used to compute the variance per class. Cg1/2 Pxi2Yg xi=Ng is used to compute mean per class. Here Yg and X are the records in class g. The other equation which is numerically stable is used to calculate diagonal variance mtrix (Rg).

3.2 Naive Bayes Classifier Based on K-Means

In this section BKM is presented which is a Bayesian classifier that is based on class decomposition that is obtained with the help of K-means

algorithm. NB has one cluster per class while the Bayesian classifier has k > 1 clusters per class. Thus BKM is a generalization of NB. Same k is used for each class for ease and exposition and elegance. Two major tasks are considered. They are computation of model and scoring data set based on the computed model. However, building the model is the most time consuming phase. From K-means executed on each class, the scoring is equivalent to an E step.

3.2.1 Mixture Model and Decomposition Algorithm

To each class with K-means [4], we fit a mixture of k clusters. Mk variance matrices(R), mk centroids (C), mk weights (W) and priors same as NB are the outputs. With notation g:j, we generalize NB notation with k cluster per class g. This means that the given matrix or vector refers to jth cluster from class g. Over vectors sums are defined instead of variables. Class priors are analogous to NB:

Table	Content	PK	non-key columns
XH	Normalized data	i	g, X_1,……,X_d
CH	Centroids	g	C_{11},C_{12},…..,C_{dk}
XD	Distances	i,g	d_1,……,d_k
XN	Nearest cluster	i,g	j
NLQ	Suff. Stats.	g,j	N_g,L_1,…..L_d, Q_1,……,Q_d
WCR	Mixture model	g,j	prior, C_1,…,C_d,
XP	Prob. Per cluster		R_1,…,R_d
XC	Predicted class	i,g	p_1,……,p_k
MQ	Model quality, Ng	i	g
		-	

Table 1: BMK Tables

In ter generalized notation, sufficient statistics are now introduced. Let Xg:j represent partition j of points in class g. This is as determined by K-means. Afterwards, L, the linear sum of points is: Lg:j=Pxi2Xgjxi, while the sum of “squared” points for cluster g:j becomes: Qg:j=Pxi2Xg:jxixTi. The Guassian parameters per class g based on L:Q are:

Cg:j =1/Ng:jLg:j,

Rg:j =1/Ng:jQg:j -1/(Ng:j)2 Lg:j(Lg:j)T

K-means is generalized to compute models “m”, thus fitting a mixture model to each class. K-means iterates after initialization and continues iterations until it converges on all classes. The following is the algorithm for the same.

Initialization:

1. Get global N; L;Q
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.

3.2.2 Naïve Bayes Classifier Programmed in Query Language

We now study as to how to create an efficient implementation of BKM in query language based on the mathematical framework introduced in the previous section. BKM’s working tables are summarized in the table 1. The tables for NB are in this case extended with the j subscript. And since there is a clustering algorithm behind, table for distance computation are introduced. Since computations require accessing all attributes/distances/ probabilities for each point at one time like NB, tables have a physical organization which stores all values for point I on the same address and there is a primary index on I for efficient hash join computation. In this manner, we force the query optimizer to perform n I/Os instead of the actual number of rows. There is a single set of tables for the classifier: all m models are updated on the same table scan;

TABLE 2
Query Optimization: Distance Computation

Distance scheme	I/Os
Completely horizontal	n
Horizontal (default)	2n
Horizontal temp	2kn+2n
Horizontal nested query	2kn + 2n
Hybrid vertical	dn
Vertical	dkn

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 less 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.

3.2.3 Model Computation

Table CH is initialized with k random points per class; this initialization requires building a stratified sample per class.

The global variance is used to normalize X to follow a Nð0; 1Þ pdf. The global mean and global variance are derived as in NB. 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. BKM uses XH thereafter at each iteration.

The simplest, but inefficient, way to compute distances is to create pivoted versions of X and C having one value per row.Such a table structure enables the usage of SQL standard aggregations (e.g., sum(), count()). Evaluating a distance query involves creating a temporary table with mdkn rows, which will be much larger than X. Since the vertical variant is expected to be the slowest, it is not analyzed in the experimental section. Instead, we use a hybrid distance computation in which we have k distances per row. We start by considering a pivoted version of X called XV. The corresponding SQL query computes k distances per class using an intermediate table having dn rows. We call this variant hybrid vertical. 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 but k columns. Such a layout enables efficient computation of the nearest cluster in a single table scan. The query optimizer decides which join algorithm to use and does not index any intermediate table. We call this approach nested horizontal.

The following SQL statement computes k distances for each point, corresponding to the gth 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;
```

We also exploited UDFs which represent an extensibility mechanism [8] provided by the DBMS. UDFs are programmed in the fast C language and they can take advantage of arrays and flow control statements (loops, if-then). Based on this layout, we developed a UDF which receives Cg as parameter and can internally manipulate it as a matrix. The UDF is called mn times and it returns the closest cluster per class. A big drawback about UDFs is that they are dependent on the database system architecture. Therefore, they are not portable and optimizations developed for one database system may not work well in another one.

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 to 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 joining XH. In the secondalternative, 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 for our running example is given below. This statement can also be used for scoring.

```
INSERT INTO XN SELECT i,g
   ,CASE WHEN d1<=d2 AND d1<=d3 THEN 1
   WHEN 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 jth vector and the jth matrix are computed together. The join computation is demanding since we are joining two tables with OðnÞ rows. For BKM, it is unlikely that variance can have numerical issues, but 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
 ,sum(X1) , .. ,sum(X4) /* L */
 ,sum(X1**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, but this 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

TABLE 3
Accuracy Varying k

Dataset	K=2	k=4	k=6	k=8	k=16
Pima	72%	74%	72%	71%	75%
Spam	75%	75%	64%	72%	52%
Bscale	55%	59%	56%	60%	65%
Wbcancer	93%	92%	93%	92%	89%

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;
```

3.2.4 Scoring

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 (this is a subtle, but important difference in SQL code generation); 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, and then, the point is assigned to the corresponding cluster. To have a more elegant implementation, we predict the class in two stages: we first determine the most probable cluster per class, and then, compare the probabilities of such clusters.

4. 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).

4.1 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.

4.2 Model Accuracy

Accuracy of predictions is measured. Fivefold cross validation is used. In each run, the training set was used to compute the model while the test set was used to measure accuracy. Their size is 80% and 20% respectively. Until K-means converged on all classes, BKM ran. To reduce overfit pruning was applied. By default the number of clusters for BKM is 4 which is most important for the accuracy of BKM. Accuracy behavior is shown in table 3 when k increases. Table 4 compares the accuracy of the three models, that include accuracy and break down per class. Considering global accuracy,

BKM is better than NB on two data sets and becomes tied with the other two data sets. When compared with DTs, it is better in one case and worse in other case. NB shows wider gap though it follows same pattern.

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

Table 6: Query Optimization, SQL versus UDF

4.3 Query Optimization

The most demanding computation for BKM that is experiment details on distance computation are provided in Table 5. When compared with worst strategy, our best distance strategy is two orders of magnitude faster and one time faster than its rival. In main memory through SQL arithmetic expressions for each row, the I/O is minimized to n operations and computations. When compared with the horizontal nested query variant, the standard aggregation on the pivoted version of X (XV) is faster.

UDFs and SQL comparision is made in Table 6. The details are finding nearest cluster per class and the computing distance. Scalar UDFs are exploited here [5]. In UDF finding a nearest cluster is straight forward. Thus this experiment favours UDF.

4.4 Speed and Time Complexity

C++ and SQL were compared on same computer with same resources available. Another comparison made is exporting with ODBC. Exported from the DBMS, C++ worked on flat files. In C++, binary files are used for maximum performance. While running C++ programs the DBMS was shut down. This means that a fair comparison is made between them. SQL, ODBC and C++ comparisons are made in Table 7. The observation is that ODBC is not performing well. Both languages, overall, scale linearly. When n grows

SQL performs better as DBMS overhead becomes less important. Nevertheless, C++ is about 4 times faster.

n X 1M	SQL	C++	ODBC
1	10	2	510
2	19	5	1021
4	37	9	2041
8	76	18	4074
16	119	36	*

Table 7: Comparing SQL, C++, and ODBC with NB

5. Related Work

Data mining algorithms can be integrated with DBMS. This is most widely used as it helps to modify internal source code. O-Cluster [3] and Scalable K-means (SKM) [1] are two examples for clustering algorithms that are internally integrated with DBMS. A classifier by name Discrete Naïve Bayes classifier has been internally integrated with the SQL Server DBMS. There are two main mechanisms to integrate data mining algorithms without modifying DBMS source code are UDFs and SQL queries. In [6] a discrete Naïve Bayes classifier programmed in SQL is introduced. Summarization of it with this paper is given. Discrete attributes are assumed in data set. What not considered here is binning numeric attributes. A large pivoted intermediate table is used by this which is inefficient. On the other hand the discrete NB model can work directly on horizontal layout.

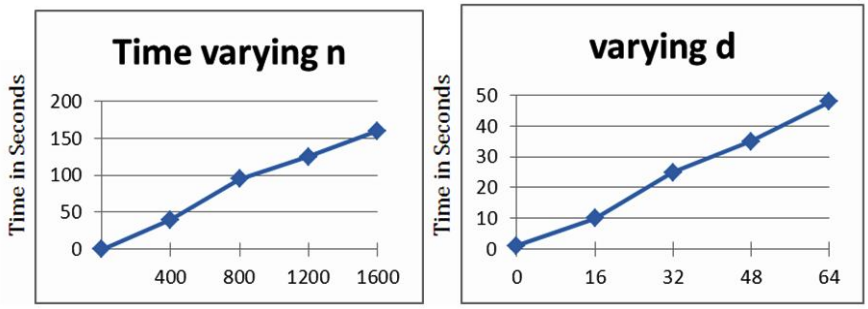

Fig. 1. BKM Classifier: Time complexity; (default *d*=4,*k*=4,*n*=100k).

Clustering algorithms in SQL [4] are extended by this paper in order to perform classification and generalizing Naïve Bayes [2]. With SQL queries, K-means clustering was programmed. This concept introduced three varians [4]. They are incremental, optimized and standard. In this paper the optimized variant has been generalized. Classification is a bit harder problem than clustering. In order to integrate data mining algorithms [5], [8], UDFs (User Defined Functions) are identified as an important extensibility mechanisms. SQL syntax is extended by ATLaS [8]. It does it by using object oriented constructs. Thus it defines aggregate and table functions by using clauses like terminate, iterate, and initialize. Thus user-friendly interface is provided to SQL standard.

6. Conclusions

This paper presents two new classifiers. They are Naïve Bayes Classifier and BKM based on decomposing classes with K-means. Two complementary aspects studied are generating efficient SQL code and increasing accuracy. Query optimizations are made to generate fast SQL code. NB is less accurate when compared with BKM. SQL implementations are four times slower than their programming counterparts in C++. Bayesian classifier can be improved further by producing more accurate models

combining dimensionality reduction techniques like PCA with Bayesian classifiers. To enhance computations UDFs need further study.

References

[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. Available online at: ftp://ftp.research.microsoft.com/pub/tr/tr-98-37.pdf

[2] T. Hastie, R. Tibshirani, and J.H. Friedman, The Elements of Statistical Learning, first ed. Springer, 2001. Available online at: http://www.stanford.edu/~hastie/local.ftp/Springer/OLD//ESLII_print4.pdf

[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. Available online at: http://www.oracle.com/technetwork/database/options/odm/overview/o-cluster-algorithm-129983.pdf

[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. Available online at: http://ieeexplore.ieee.org/Xplore/login.jsp?url=http%3A%2F%2Fieeexplore.ieee.org%2Fiel5%2F69%2F33191%2F01563982.pdf%3Farnumber%3D1563982&authDecision=-203

[5] C. Ordonez, "Building Statistical Models and Scoring with UDFs," Proc. ACM SIGMOD, pp. 1005-1016, 2007. Available online at: http://dl.acm.org/citation.cfm?id=1247599

[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. Available online at: http://www.freepatentsonline.com/7051037.pdf

[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. Available online at: http://www.research.ibm.com/people/r/rish/papers/paperECML2003.pdf

[8] H. Wang, C. Zaniolo, and C.R. Luo, "ATLaS: A Small but Complete SQL Extension for Data Mining and Data Streams," Proc. Very Large Data Bases (VLDB) Conf., pp. 1113-1116, 2003. Available online at: http://www.cs.ucla.edu/~zaniolo/papers/vldb03demo.pdf

* * * * *