A two layers incremental discretization based on order statistics

Christophe Salperwyck and Vincent Lemaire

Abstract Large amounts of data are produced today: network logs, web data, social network data... The data amount and their arrival speed make them impossible to be stored. Such data are called streaming data. The stream specificities are: (i) data are just visible once and (ii) are ordered by arrival time. As these data can not be kept in memory and read afterwards, usual data mining techniques can not apply. Therefore to build a classifier in that context requires to do it incrementally and/or to keep a subset of the information seen and then build the classifier. This paper focuses on the second option and proposed a two layers approach based on order statistics. The first layer uses the Greenwald and Khanna quantiles summary and the second layer a supervised method such as MODL.

Key words: Incremental learning, discretization, order statistics

1 Introduction

Many companies produce today large amounts of data. Sometimes data can be kept into a database, sometimes their arrival speed makes them impossible to be stored. In that specific case mining data is called stream mining. The stream specificities are: (i) data are just visible once and (ii) are ordered by arrival time. Such an amount of data leads to the impossibility to keep them in memory and to read them afterwards. Therefore to build a classifier in that context requires doing it incrementally and/or to keep a subset of the information seen and then build the classifier. In this paper the focus is on the second option. This can be achieved by: keeping a subset of the stream examples, calculating a density estimation or having order statistics. The work presented in this paper focuses on numeric attributes discretization based on

Christophe Salperwyck
Orange Labs, Lannion, France – LIFL, Université de Lille 3, Villeneuve d'Ascq, France
e-mail: christophe.salperwyck@orange.com

Vincent Lemaire
Orange Labs, Lannion, France
e-mail: vincent.lemaire@orange.com

order statistics. The discretization is then used as a pretreatment step for a supervised classifier.

2 Related works

Incremental discretization is mainly used in two fields: (i) data mining field to be able to discretize large data set or to discretize data on the fly; (ii) Data Base Management Systems (DBMS) to have order statistics (quantiles estimates) on tables for building efficient query plans. This section gives a brief state of art of the main incremental discretization methods used in these two fields to have order statistics.

2.1 Data mining field

Gaussian density approximation: The main idea of this method relies on the hypothesis that the observed data distribution follows a Gaussian law. Only two parameters are needed to store a Gaussian law: the mean and the standard deviation. The incremental version required one more parameter: the number of elements. An improved version for supervised classification on stream can be found in [11] but it needs a parameter to set up the number of bins derived from the Gaussian. This method has one of the lowest memory footprints.

PiD: Gama and Pinto in [8], proposed a two layers incremental discretization method. The first layer is a mix of a discretization based on the methods named "Equal Width" and "Equal Frequency" (algorithm details: [8] p. 663). This first layer is updated incrementally and needs to have much more bins than the second one. The second layer uses information of the first one to build a second discretization. Many methods can be used on the second layer such as: Equal Width, Equal Frequency, Entropy, Kmeans... The advantage of this method is to have a fast first layer which can be used to build different discretizations on it (second layer).

Online histogram: Ben-Haim et al [1], presented an incremental and online discretization for decision trees. Their algorithm is based on three methods: (i) UPDATE - add a new example. It can be done by inserting the new example directly in an existing histogram or create a new bin with it and then do a merge, (ii) MERGE - merge two bins in one, (iii) UNIFORM: use a trapezoid method to build the final Equal Frequency bins. This method has a low computational requirement and is incremental but it introduces some errors. In case of skewed distributions the authors recommend to use bound error algorithms.

2.2 DBMS field

MLR: Manku et al. [10] developed an algorithm to approximate quantiles based on a pool of buffers. Their approach has three operation: (i) NEW - takes an empty buffer and fill it with new values from the stream, (ii) COLLAPSE - when all buffers are full, some need to be merged to get new empty buffers – this operation takes

at least two buffers and merges them to have just one full at the end, (iii) OUTPUT - this operation collapses all the buffers into one and returns the quantile value for the given parameter. This method has a theoretical bound on the error ε and on the required space: $\frac{1}{\varepsilon}log^2(\varepsilon N)$, where N is the size of stream.

GK: Quantiles provide order statistics on the data. The ϕ-quantile, with $\phi \in [0,1]$ is defined as the element in the position $\lceil \phi N \rceil$ on a sorted list of N values. ε is the maximum error on the position of the element: an element is an ε approximation of a $\phi - quantile$ if its rank is between $\lceil (\phi - \varepsilon)N \rceil$ and $\lceil (\phi + \varepsilon)N \rceil$. It corresponds to an "Equal Frequency" discretization; the number of quantiles being in that case the number of intervals.

The GK quantiles summary, proposed by Greenwald and Khanna [9] is an algorithm to compute quantiles using a memory of $O(\frac{1}{\varepsilon}log(\varepsilon N))$ in the worst case. This method does not need to know the size of the data in advance and is insensitive to the arrival order of the examples. The algorithm can be configured either with the number of quantiles or with a bound on the error. Its internal structure is based on a list of tuples $< v_i, g_i, \Delta_i >$ where :

- v_i is a value of an explanatory feature of the data stream
- g_i corresponds to the number of values between v_{i-1} and v_i
- Δ_i is the maximal error on g_i

2.3 Summary

This subsection aims to present a synthetic overview of the methods described above. This overview uses two criteria taken from [6]. The first criterion: *global/local* corresponds to the way methods use data to build intervals. A method using all data for building all bins is considered as *global*. A method splitting data into subset and doing local decision is considered as *local*. The second criterion: *supervised* corresponds to methods using class labels to build the discretization. We added two other criteria: *parametric* - a non-parametric method finds the number of intervals automatically, *online/stream* - evaluate the ability to work online and to deal with data streams.

Table 1 presents the comparison of all methods seen above versus these four criteria. The second part of the table reports the widely used offline methods *Equal Width* and *Equal Frequency*, and also two competitive supervised methods used in the next section: MDLP and MODL.

3 Our proposal

3.1 Objective

Our proposal aims to used at best the data mining field and the DBMS field to propose an incremental discretization method which intrinsically realizes a compromise

Method	Global / local	Parametric	Supervised	Online / stream
Gaussian	Global	Yes	No	Yes
PID (Layer 1)	Global	Yes	No	Yes
Online histogram	Global	Yes	No	Yes
MLR	Global	Yes	No	Yes
GK	Global	Yes	No	Yes
Equal Width/Freq	Global	Yes	No	No
MDLP	Local	No	Yes	No
MODL	Global	No	Yes	No

Table 1 Discretization methods comparison

between the error ε and the memory used. This method will also have to be robust and accurate for classification problems.

3.2 Proposal

The idea is to use a two layer incremental discretization method as PiD [8] but in our case bounds in memory. The first layer summarizes (using counts per class) the input data, using much more intervals than required, in a single scan over the data stream. The second layer processes the first layer summary and produces the final discretization. The memory is used at best to have the lowest error.

For the first layer, the Greenwald and Khanna quantiles summary (GK) suits this requirement the best and provides order statistics. We adapted the GK summary to store directly the class counts in tuples. For the second layer, among methods using order statistics two are particularly interesting considering their performances: Recursive Entropy Discretization (MDLP) [7] and Minimum Optimized Description Length (MODL) [3]. They both use an entropy based criterion to build the discretization and the MDL (Minimum Description Length) criterion to stop finding intervals. They are supervised and known to be robust. The choice to use GK for the first layer and either MDLP or MODL in the second layer is coherent since the complete structure is based on order statistics. The errors on cut points depends mainly on the number of bins used the first layer. Because the second layer used the MODL approach and since MODL is a discretization method based on counts, the error on a split position of our two level method is related at worst to: $\arg\max_i(g_i + \Delta_i)/2$, where i is the index of the interval int the first level. This indicates that when the number of intervals of the first level increases, the error decreases.

Figure 1 shows how our method proceeds. A GK summary is created for each feature and updated after the arrival of a new example. When the model is needed, GK summaries provide univariate contingency tables to the discretization method (MODL or MDLP). A new contingency table (expected smaller) is built and returned by these methods. Using GK quantile values and the contingency table after discretization give at the same time cut points, density estimations per interval and conditional density estimation per interval for this numeric feature. Finally a classifier based on order statistics is built, as for example a naive Bayes.

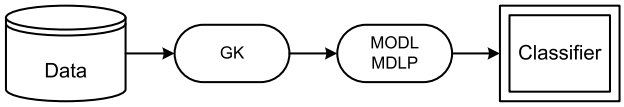

Fig. 1 Two layers discretization.

4 Experiments

4.1 Large scale learning challenge

The Delta training dataset from the large scale learning challenge[1] is used for a first experiment. This dataset contains 500,000 examples; each example consists of 500 numerical features and a boolean label. 100,000 examples were kept for a test set and train examples were taken from the 400,000 remaining instances. We adapted the MODL discretization so that it uses GK quantiles summary as an input and built a naive Bayes classifier on this discretization. The GK quantiles summary is set up with 10 and 100 quantiles: GK10 (200K bytes) and GK100 (2M bytes). The reservoir sampling approach [13] is also used as a baseline method to compare performances with a bounded memory technique: GK10, GK100 corresponds respectively to a reservoir of 50 and 500 examples.

Figure 2 shows the comparison of theses approaches using the AUC (Area Under learning Curve) performance indicator. The two reservoir sizes used are equivalent to the memory consumed by GK10 and GK100. With limited memory GK methods are performing much better than reservoir sampling. Compared to the naive Bayes classifier using all data into memory, the GK performances with 100 quantiles (GK100) are almost the same.

This first experiment shows that with a small given amount of memory our method performances are similar to the one loading all the data into memory.

4.2 ICML Exploration and Exploitation challenge

The ICML 2011 challenge[2] aims to show the online ability and robustness of our two levels discretization. This challenge data are very unbalanced and contain a high level of noise. The dataset contains 3 millions examples; each example consists of 100 numerical and 19 nominal features labels by a boolean (click/no-click). The purpose of the challenge is to evaluate online content selection algorithms. Each algorithm has to perform a sequence of iterations. For each iteration, a batch of six visitor-item pairs is given to the algorithm. The goal is to select the instance which is most likely to provoke a click. This challenge has strong technical constraints: (i) a time limit (100ms per round), (ii) a limited space (1.7GB of memory). These challenge data contain nominal features; we dealt with them using a hash based solution: nominal values are hashed and put into a fixed number of buckets. The

[1] http://largescale.ml.tu-berlin.de

[2] http://explo.cs.ucl.ac.uk/

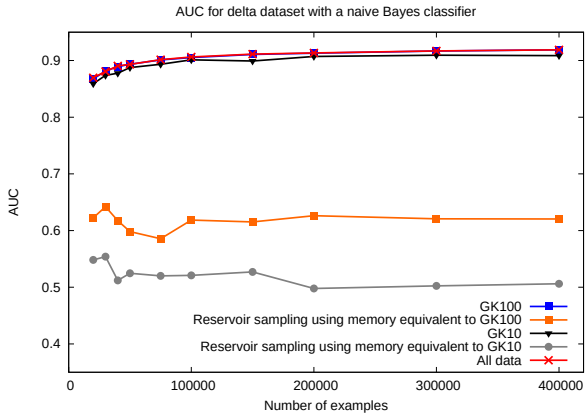

Fig. 2 Naive Bayes AUC performances with all data, GK and reservoir sampling.

click/no-click counts are stored for each bucket. These buckets are used as numerical bins so that we can deal with nominal features as numerical ones.

Due to the challenge specificities our two layers approach was adapted. This dataset is very unbalanced: clicks are very scarce - 0.24%. Methods as MODL are known to be robust but this robustness with noisy data leads to a late discovery of cut points as shown by the MODL curve on Figure 3. As the challenge score was cumulative rewards, the model has to make decision even with just few clicks. Waiting to make decision could provide a better classifier at the end but a lower score on this challenge evaluation. To be more reactive probability estimation tree (PETs: [12]) were built on our first level summaries. A tree can be seen as a discretization method (our 2^{nd} layer). The final step (corresponding to the *Classifier* on Figure 1) is a predictor composed of an averaging of PETs' predictions .

Figure 3 shows results on this challenge: Inria (ranked 1^{st}), our submission with PETs (ranked 2^{nd}) and a random predictor. Our approach was competitive and provides good density estimations for building online tree classifiers.

5 Future works

5.1 Extension to nominal features

The work presented before only addresses the discretization for numerical features. Many classification problems contain nominal features. Moreover the MODL approach for grouping modalities [2] is a competitive method to find groups on nom-

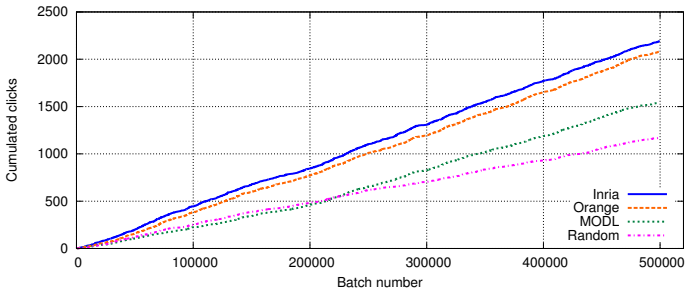

Fig. 3 ICML Exploration and Exploitation challenge 2011 results.

inal features. The simplest summary for nominal features just keeps counts – it requires low memory and processing time and is practicable if the number of different values is low. Unfortunately the number of nominal values can be large, for example: *client ids, cookies, city names, etc*. As we want to bound memory we can only afford to focus on frequent values. This can be done using a hash function: nominal values are hashed and put into a fixed number of buckets in which classes counts are stored. In order to reduce errors several hashing functions may be combined as proposed in the count-min sketch algorithm [4].

5.2 Dialogs between two layers

A dialog between the two layers to control the number of tuples in the first layer could be beneficial to share memory between different features summaries. In a classification problem with many features, some of them may need a very fine discretization and some may not need it. As the second layer is non parametric and supervised it can inform the first layer if it needs more or less bins.

5.3 Online trees

The stream mining community often uses trees to build online and incrementally classifiers. The most known ones are the Hoeffding trees proposed in VFDT [5]. In those trees, leaves keep statistics on the data. A leaf is transformed into a node using a split criterion (usually entropy or Gini index). As a split is a definitive decision the Hoeffding bound is used to set the confidence (δ) in the split. Another parameter (τ) is used to break ties between two attributes with similar criterion to avoid late splits.

Our two layers discretization can be used as summaries in the tree leaves. Moreover the second layer of our discretization method (MODL method applied on the first layer summary) gives cut points for a feature with a quality index. This quality index allows selecting the feature on which to split and the cut points where to split.

If a feature is not considered as informative its index equals zero. The tree can expend by splitting a leaf on the feature having the greatest non-null index as it is sure that this feature is informative: the Hoeffding bound is not anymore needed. With this MODL criterion there is no need to have the two previous parameters δ and τ to build online trees.

6 Conclusion

The first experiments validate that with large data sets and bounded memory, our two layers discretization has a strong interest. Our approach uses order statistics on both levels and can be set up to use a fixed memory size or to stay beyond a given error. We used Greenwald and Khanna quantiles summary for the first layer and MODL discretization for the second layer as they are known to be amongst the most competitive methods to build quantiles summary and perform supervised discretization. Classifiers assuming features independence can be easily built on the summary as shown on the first experiment with a naive Bayes classifier. Some other classifiers such as online trees can also take advantage of our method.

References

1. Ben-Haim, Y., Tom-Tov, E.: A streaming parallel decision tree algorithm. Journal of Machine Learning **11**, 849–872 (2010)
2. Boullé, M.: A Bayes Optimal Approach for Partitioning the Values of Categorical Attributes. Journal of Machine Learning Research **6**(04), 1431–1452 (2005)
3. Boullé, M.: MODL: A Bayes optimal discretization method for continuous attributes. Machine Learning **65**(1), 131–165 (2006)
4. Cormode, G., Muthukrishnan, S.: An improved data stream summary: the count-min sketch and its applications. Journal of Algorithms **55**(1), 58–75 (2005)
5. Domingos, P., Hulten, G.: Mining high-speed data streams. In: Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 71–80. ACM New York, NY, USA (2000)
6. Dougherty, J., Kohavi, R., Sahami, M.: Supervised and unsupervised discretization of continuous features. In: In Proceedings of the Twelfth International Conference on Machine Learning, pp. 194–202. Morgan Kaufmann Publishers Inc. (1995)
7. Fayyad, U., Irani, K.: Multi-interval discretization of continuous-valued attributes for classification learning. Proceedings of the International Joint Conference on Uncertainty in AI pp. 1022–1027 (1993)
8. Gama, J., Pinto, C.: Discretization from data streams: applications to histograms and data mining. In: Proceedings of the 2006 ACM symposium on Applied computing, pp. 662–667 (2006)
9. Greenwald, M., Khanna, S.: Space-efficient online computation of quantile summaries. ACM SIGMOD Record **30**(2), 58–66 (2001)
10. Manku, G.S., Rajagopalan, S., Lindsay, B.G.: New York, New York, USA
11. Pfahringer, B., Holmes, G., Kirkby, R.: Handling numeric attributes in hoeffding trees. Advances in Knowledge Discovery and Data Mining pp. 296–307 (2008)
12. Provost, F., Domingos, P.: Tree induction for probability-based ranking. Machine Learning **52**(3), 199–215 (2003)
13. Vitter, J.S.: Random sampling with a reservoir. ACM Transactions on Mathematical Software **11**(1), 37–57 (1985)