A Preprocessing Scheme for High-Cardinality Categorical Attributes in Classification and Prediction Problems

Daniele Micci-Barreca
ClearCommerce Corporation
11500 Metric Blvd.
Austin, TX 78732
daniele@clearcommerce.com

ABSTRACT

Categorical data fields characterized by a large number of distinct values represent a serious challenge for many classification and regression algorithms that require numerical inputs. On the other hand, these types of data fields are quite common in real-world data mining applications and often contain potentially relevant information that is difficult to represent for modeling purposes.

This paper presents a simple preprocessing scheme for high-cardinality categorical data that allows this class of attributes to be used in predictive models such as neural networks, linear and logistic regression. The proposed method is based on a well-established statistical method (empirical Bayes) that is straightforward to implement as an in-database procedure. Furthermore, for categorical attributes with an inherent hierarchical structure, like ZIP codes, the preprocessing scheme can directly leverage the hierarchy by blending statistics at the various levels of aggregation.

While the statistical methods discussed in this paper were first introduced in the mid 1950's, the use of these methods as a preprocessing step for complex models, like neural networks, has not been previously discussed in any literature.

Keywords

Categorical attributes, predictive models, neural networks, hierarchical attributes, Empirical Bayes.

1. INTRODUCTION

It is well known that data preprocessing often represents over 80% of the time required to carry out any real-world data mining project. High-cardinality categorical data fields are one of the most challenging data types to handle for data mining algorithms, in particular for pattern recognition and statistical models such as neural networks, linear and logistic regression, polynomial models and support vector machines (SVM). In fact, while some machine learning algorithms, like decision trees and other rule-induction methods (CART, C5.0, etc.), can handle high-cardinality categorical attributes without the need for external preprocessing, regression-type methods strictly require every input attribute to be represented in a numerical format. Unfortunately, when accuracy is the key requirement for the application, this second class of models is often the one that delivers the best results for classification and prediction problems. Furthermore, high-cardinality categorical data elements, which are defined as non-ordinal fields characterized by a very large number of unique values, are quite typical in real-world databases. Examples of this type of fields include ZIP codes, SIC[1]s, product codes, email domains, IP address blocks, country codes, part codes, procedure codes, diagnosis codes, etc.

[1] Standard Industrial Classification

This paper presents a simple data-preprocessing scheme that transforms high-cardinality categorical attributes into quasi-continuous scalar attributes suited for use in regression-type models. The key transformation used in the proposed scheme is one that maps each instance (value) of a high-cardinality categorical to the probability estimate of the target attribute. In a classification scenario, the numerical representation corresponds to the posterior probability of the target, conditioned by the value of the categorical attribute. In a prediction scenario, the numerical representation corresponds to the expected value of the target given the value of the categorical attribute. This scheme is statistically sound and produces a representation that preserves most of the predictive power of the original categorical attribute. It is applicable to both classification (binary and multi-class) and prediction problems and it elegantly handles missing data. Furthermore, the preprocessing scheme is simple enough to be carried out completely as an in-database process, enabling a scalable implementation of the procedure and ease of model deployment. Finally, in the case of categorical attributes with an inherent hierarchical structure, like ZIP codes, this scheme has a natural extension that takes direct advantage of the hierarchy, combining target statistics at different levels of aggregation.

This paper is organized as follows. Section 2 presents a review of the most common approaches to the preprocessing of high-cardinality categorical attributes. Section 3 describes the preprocessing scheme that is the topic of this paper. Section 4 presents an extension of the scheme for categorical attributes with hierarchical structure is presented, and Section 5 concludes the paper.

2. PREPROCESSING SCHEMES FOR CATEGORICAL ATTRIBUTES

For low-cardinality categorical attributes the most widely used numerical representation method is the one that uses N binary derived inputs, one for each possible value of the original attribute. This encoding scheme represents each value of the original categorical with a binary vector with the *i-th* component set to one, and the rest set to zero. This *1-to-N* mapping is commonly applied when N is relatively small (e.g. less than ten). But, as N grows, the number of inputs to the model increases and consequently the number of parameters to be estimated increases. Thus, this method is not applicable to high-cardinality attributes with hundreds or thousands of distinct values.

The method that is probably most frequently used today for high-cardinality attributes is *clustering*. The basic idea is to reduce the original *1-to-N* mapping problem to a *1-to-K* mapping problem, with $K << N$. To accomplish this, the cardinality of the data is first reduced by grouping individual values into K sets of values. Then each set is represented with a binary derived input. Thus the encoding first identifies the group to which the value belongs and then sets the corresponding bit in the numerical representation.

Key to an effective representation is a value-grouping scheme that maximally preserves the information contained in the original attribute. An effective approach is to group values that exhibit similar target statistics. This can be accomplished via, for example, a hierarchical clustering algorithm [8] that uses as a distance metric based on the difference in the statistics of the target (e.g. probability or average, depending on the nature of the target) between to groups. Alternatively, one could use some information-theoretical metric (like the gain ratio [12]) to measure the effect of merging each possible pair of clusters. The two clusters with the smallest distance (or greatest improvement of the gain ratio) will be merged, reducing the number of clusters by one. The process is the iterated until no improvement is achieved or the desired number of clusters has been reached. This method is used in decision tree induction algorithms like C4.5 [11] and in commercial data mining packages like Unica's Affinium Model [7] and Accrue Decision Series [1], where value clustering is used as a preprocessing step for a variety of predictive algorithms, including logistic regression and neural networks.

The method presented in Section 3 is practically very similar to clustering since categorical values that exhibit similar statistics with respect to the target have a similar representation. However, while offering similar benefits, the proposed method is simpler to implement and does not require a final binary *1-to-K* encoding like clustering does. Furthermore, while clustering basically reduces the information contained in the original categorical attribute to K distinct states of the derived attributes, the proposed scheme provides more granular representation.

Projection (principal component) techniques have also been proposed in the statistical literature to map categorical data to a numerical representation. For example, Dual Scaling [10] and CRIMCOORD [5] are two of the methods that utilize principal component-type analysis to reduce categorical data to a numerical representation. However, for high-cardinality attributes these methods require the manipulation of extremely large matrices. Furthermore, the representation does not depend on the value of the target attribute, but only on the distribution of the independent categorical attribute.

3. CATEGORICAL ENCODING USING TARGET STATISTICS

The basic principle motivating the proposed preprocessing scheme is to map individual values of a high-cardinality categorical independent attribute to an estimate of the probability or the expected value of the dependent attribute. Basically, this preprocessing scheme transforms the original categorical values via a simple single-attribute model (specifically Bayesian model, as it is shown in Section 3.1) before presenting it to the actual model.

The next section illustrates the details of the transformation in the case of a binary dependent attribute. The transformation applies immediately to the case of continuous targets (Section 3.2) and, with minor changes, to multi-valued classification problems (Section 3.3).

3.1 Binary Target

When the target attribute Y is binary, $Y \in \{0,1\}$, the transformation maps individual values X_i of a high-cardinality categorical attribute X to a scalar, S_i, representing an estimate of the probability of $Y=1$ given that $X=X_i$:

$$X_i \rightarrow S_i \cong P(Y|X=X_i) \qquad (1)$$

Notice that since S_i represents a probability, the transformed attribute S is automatically normalized between 0 and 1, which is also a desirable feature for neural network models.

The next step is to estimate the probabilities required by the transformation. We will make the assumption that the available data set has been partitioned into a training set containing n_{TR} records and a test set containing n_{TS} records. Since the estimation of the probabilities becomes an integral part of the model training process, only the records in the training set should be used. Notice that not all the possible values of X may occur in the training set; in fact, some may only be present in the test set or will be observed when the model is applied to new data. Therefore the transformation scheme must be able to handle unforeseen values for the categorical attribute.

If the number of training set cases such that $X=X_i$ were sufficiently large for every value of X, the probability estimates could be reasonably computed as the ratio between the number of observations with $Y=1$, n_{iY}, and the size of the cell n_i:

$$S_i = \frac{n_{iY}}{n_i} \qquad (2)$$

Unfortunately, the more typical scenario for a high-cardinality categorical attribute is one where the number of unique values of X is very large and the records are unevenly distributed across the possible values of X. In this scenario many of the cells will be characterized by a small sample size n_i, therefore, the direct estimate of $P(Y|X=X_i)$ in (2) would be highly unreliable.

To mitigate the effect of small cell counts the probability estimates S_i are calculated as the mixture of two probabilities: the posterior probability of Y given $X=X_i$, calculated via (2), and the prior probability of Y (the "baseline" probability based on the entire training set). The two probabilities are then "blended" using a weighting factor that is a function of the sample size:

$$S_i = \lambda(n_i)\frac{n_{iY}}{n_i} + (1-\lambda(n_i))\frac{n_Y}{n_{TR}}, \qquad (3)$$

where n_Y is the total number of cases such that $Y=1$ and the weighting factor $\lambda(n_i)$ is a monotonically increasing function on n_i bounded between 0 and 1.

The rationale for the empirical formula (3) is that when the cell size is large ($\lambda \cong 1$) we should assign more credit to the posterior probability estimate provided by (2). However, if the sample size is small ($\lambda \cong 0$), then we replace the probability estimate with the *null hypothesis* given by the prior probability of the dependent attribute. In other words, we assume that knowing that $X=X_i$

provides no additional information regarding the likelihood of the target.

Although, as it will be shown, the basic blending formula (3) has been used extensively in both the statistics and machine learning communities, every approach can be reduced to a specific functional form for $\lambda(n)$. Typically $\lambda(n)$ is chosen as a parametric function with one or more tunable parameters and could be optimized based on the characteristics of the data. For example, $\lambda(n)$ can be chosen as:

$$\lambda(n) = \frac{1}{1+e^{-\frac{(n-k)}{f}}}, \qquad (4)$$

which is a s-shaped function that assumes value 0.5 for $n=k$. The parameter f provides control over the slope of the function around the inflexion point. The parameter k determines half of the minimal sample size for which we completely "trust" the estimate based on the sample in the cell. The parameter f controls the rate of transition between the cell's posterior probability and the prior probability. For $f\rightarrow\infty$ the formula (3) becomes a hard threshold between the posterior and the prior probability. Notice that the implementation of categorical grouping in C4.5 uses a threshold to exclude low-count categorical values from the grouping process, which basically corresponds to this extreme case.

The probability estimation formula in (3) has a long history in statistics and actuarial science. In fact, this formula is the primary achievement of a branch of Bayesian probability that goes under the name of Empirical Bayesian (EB) [13][1]. Although the weighting of the posterior probability and the prior probability, known in statistics as the *shrinkage factor*, derives directly from Bayes rule, the contribution of the EB approach was to compute the prior from the available data, rather then requiring a postulated prior.

The general formula used in Empirical Bayes estimation is:

$$\hat{P}_i = B_i y_i + (1-B_i)\bar{y}, \qquad (5)$$

where $\bar{y}$ is the prior, computed from the data, and y_i is the empirical posterior probability computed for the cell $X=X_i$. The shrinkage factor B_i ($0<B_i<1$) takes different forms depending on the specific EB estimation method used. When the probability distribution of the data within a cell and the probability distribution of the estimated posterior are both assumed Gaussian, B_i takes the form of:

$$B_i = \frac{n_i\tau^2}{\sigma^2 + n_i\tau^2}, \qquad (6)$$

where σ^2 is the variance within the cell, τ^2 is the variance of the entire sample and n_i is the size of the cell. Basically, B_i is a particular specification of $\lambda(n)$, which takes into account not only the sample size, but also the intra-cell variance and the overall variance of the data.

The formula in (3) is also related to probability estimation formulas proposed in the machine learning literature, primarily used as a criteria for pruning decision trees. Cestnik [4] and Cestnik and Bratko [3] proposed a variation on the well-known Laplace's law of succession [6] for estimating probabilities. Their formula also produces estimates that are bounded between the posterior probability and the prior probability of the target. They called this the *m-probability estimate*, defined by:

$$p_c^* = \frac{n_c + p_c m}{n+m}, \qquad (7)$$

where p_c is the prior probability of class c, n_c is the number of cases that belong to class c, n is the total number of cases in the cell and m is a parameter which depends on the data domain.

It is easy to show that (7) is a special case of (3) when:

$$\lambda(n) = \frac{n}{m+n} \qquad (8)$$

Compared to (4) the formulation of $\lambda(n)$ in (8) has the advantage of being a single parameter function. On the other hand, for small values of $n \ll m$, (7) is basically a linear function of n, while (4) can be tuned via f to become nearly flat for very small values on n, further mitigating the effect of small sample cells. Also notice that (7) is equivalent to the B factor used in empirical Bayes, when m is equal to the ratio σ^2/τ^2. This is actually quite interesting. In their paper Cestnik and Bratko simply stated that m is domain-dependent and depends on the expected noise of the domain; if the expected noise is small then m should be small and vice versa. The EB literature basically provides a specific recommendation for m, which is the ratio of the variances. This is consistent with Cestnik and Bratko's intuitive recommendation: the noisier the data in the sample compared to the overall noise in the dataset, the larger is m. Another advantage of the EB formulation is that the shrinkage varies from cell to cell, depending on the intra-cell variance. Given two cells with the same size, more shrinkage will occur in the cell that exhibits more relative variability.

Another, relatively recent, example of shrinkage methods in the machine learning literature is the work of McCallum et al. [9], in which EB was used to significantly improve a hierarchical text classification scheme.

3.1.1 Handling Missing Data

In most real-world data mining problems the dataset contains *missing data,* namely records for which the value of an attribute X is not specified for a given record in the dataset. Depending on the nature of the attribute X and the type of algorithm adopted, missing data elements can be treated in different ways. For example, if X is numeric, the missing data element is often "filled" with the average of X, or an estimate of X based on other independent attributes. If X is a low-cardinality categorical attribute represented with a *one-to-m* binary encoding, the missing case could be represented as a vector of m zeros. Finally, most machine learning algorithms (like Naïve Bayes and decision trees) simply ignore missing values or treat them just as another value.

The proposed preprocessing scheme handles missing values by treating them just as any other value. This can be done by introducing an additional value for X, the *null* value X_0, and estimating the probability of the target for $X=X_0$, using the standard formula (3):

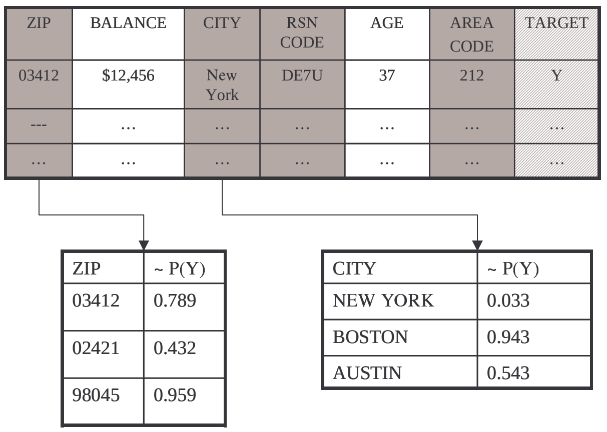
$$S_0 = \lambda(n_0)\frac{n_{0Y}}{n_0} + (1-\lambda(n_0))\frac{n_Y}{n_{TR}} \quad (9).$$

The advantage of this approach is that *if* the occurrence of a missing value for attributes *X* has indeed predictive relevance for the dependent, then S_0 with capture that information. However, if missing values have no particular relationship with the dependent, then S_0 is going to converge toward the prior probability of the target, which corresponds to a "neutral" representation of the missing value.

3.1.2 *Implementation and Model Deployment*

One of the main advantages of the proposed preprocessing schema is that its practical implementation is relatively straightforward. In most cases, the preprocessing can be completely implemented as an in-database procedure, leveraging the scalability and performance of today's database systems.

The typical in-database implementation of the scheme requires the creation, for each transformed attribute, of a lookup table indexed by the value of the original categorical attributes (Figure 1).



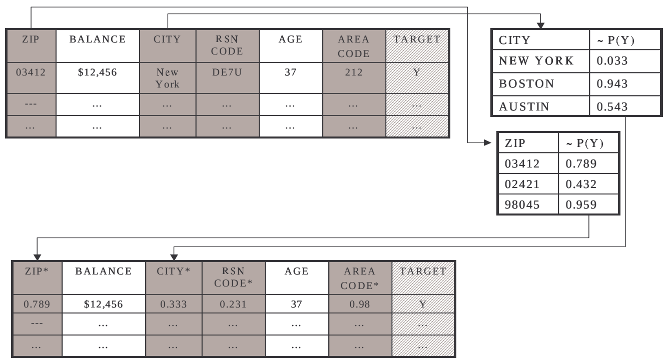
ZIP	BALANCE	CITY	RSN CODE	AGE	AREA CODE	TARGET
03412	$12,456	New York	DE7U	37	212	Y
---	…	…	…	…	…	…
…	…	…	…	…	…	…

ZIP	~ P(Y)
03412	0.789
02421	0.432
98045	0.959

CITY	~ P(Y)
NEW YORK	0.033
BOSTON	0.943
AUSTIN	0.543

Figure 1: Creating lookup tables

The content of the lookup tables should be determined utilizing only data in the training dataset, if the model is being evaluated using a traditional cross-validation scheme. After the tables are created, both the training and the testing data records will be transformed by joining the original categorical value with the corresponding lookup table (Figure 2).



ZIP	BALANCE	CITY	RSN CODE	AGE	AREA CODE	TARGET
03412	$12,456	New York	DE7U	37	212	Y
---	…	…	…	…	…	…
…	…	…	…	…	…	…

CITY	~ P(Y)
NEW YORK	0.033
BOSTON	0.943
AUSTIN	0.543

ZIP	~ P(Y)
03412	0.789
02421	0.432
98045	0.959

ZIP*	BALANCE	CITY*	RSN CODE*	AGE	AREA CODE*	TARGET
0.789	$12,456	0.333	0.231	37	0.98	Y
---	…	…	…	…	…	…
…	…	…	…	…	…	…

Figure 2: Mapping values to numerical representation

The prior two steps, lookup generation and mapping, are easily implemented via primitive database operations such as GROUP BY and JOINS. The computation of the formula (3) requires the prior calculation of the prior probability, easily done by a summary query against the entire training table, and the computation of the *λ(n)*, which could be pre-computed as a separate lookup table or implemented via stored procedures or other embedded code.

If the final deployment of the model involves batch scoring of new data, the same database transformations can be used for the final model deployment. However, if the model is used to evaluate real-time data that in not previously stored in a supporting database, the actual lookup transformations will need to be implemented. Notice, however, that the probability estimates are only computed once and thus the required logic is simply that of a lookup in a static table.

3.2 Continuous Targets

The preprocessing scheme proposed for binary targets can also be applied to the case of a continuous target attribute *Y*. Although there is a significant difference from the formal standpoint (estimating expected values vs. estimating probabilities) the formula in (3) remains basically unchanged. However, now we consider the average of Y across the training data, *E[Y]* as the "null hypothesis", and the average of *Y* when $X=X_i$ as the raw estimate for the cell:

$$S_i = \lambda(n_i)\frac{\sum_{k\in L_i} Y_k}{n_i} + (1-\lambda(n_i))\frac{\sum_{k=1}^{N_{TR}} Y_k}{n_{TR}}\,, \qquad (10)$$

where L_i is the set of observations, of size n_i, for which $X=X_i$.

Again, the principle is to weight the estimate of the target toward $E[Y|X=X_i]$ when the sample size is large, and toward the (training) population average when the sample size is small.

In practice, the formulation for binary and continuous targets is completely identical if the binary case is represented as a numerical target with values 0 and 1.

3.3 Extension to Multi-Valued Categorical Targets

Extending the preprocessing scheme to the case of an *m*-valued categorical target, $Y\in[Y_1, Y_2,\ldots,Y_m]$ is very straightforward. For each possible value Y_j of the dependent attribute (which has typically low-cardinality in classification tasks) a derived input attribute X_j^* is created in substitution of the original high-cardinality categorical independent attribute *X*. Each derived attribute X_j^* will represent an estimate of $P(Y=Y_j|X=X_i)$, using the formula in (3). Notice that this representation is actually redundant, since only *m-1* of the *m* derived components will be linearly independent. Therefore, any one (but only one) of the X_j^* can be dropped without loss of information.

This formulation of the transformation can significantly increase the number of actual model inputs if the cardinality of the dependent attribute is high and the number of high-cardinality categorical independent attributes in also high. In most cases, however, multi-class classification problems tend to have a small number of output classes, in practice limiting the dimensionality of the final model.

4. APPLICATION TO CATEGORICAL DATA WITH HIERARCHICAL STRUCTURE

Some types of high-cardinality categorical attributes, usually codes, are characterized by a well-defined hierarchical structure that lends itself to grouping and aggregation. A typical example is represented by the US postal coding system: all 5-digit zip codes (ZIP5) with the same first three digits (ZIP3) tend to be in the same metropolitan area. Therefore, it is possible to geographically aggregate individual ZIP5 values based on the ZIP3 or ZIP4 values. Another example is the Standard Industry Code (SIC) used in the US to classify commercial organizations based on their primary line of business. The SIC code is a four-digit numerical code that features a well-defined hierarchical structure. For example, SIC code 2514 indicates "Metal House Hold Furniture", the SIC group 251, to which 2514 belongs, indicates "Household Furniture", and finally the SIC Major Group 25 indicates "Furniture and Fixtures". Other examples of high-cardinality codes with hierarchical structure are phone numbers, product codes, Vehicle Identification Numbers (VIN), IP addresses, etc.

Often practitioners take advantage of the hierarchical nature of these special attributes to reduce the cardinality of the attribute by aggregating values. For example, if the data is too sparse at the ZIP5 level, one may decide to aggregate ZIP5 values to the ZIP3 level to obtain fewer and better-populated cells. This is basically equivalent to achieving cardinality reduction via clustering using domain knowledge.

For high-cardinality categorical attribute with hierarchical structure, the proposed scheme can be easily extended to take direct advantage of the hierarchy. The advantage with respect to the basic aggregation approach is that this approach can take advantage of large cells that may exist at different levels of aggregation. For example, it may appear that for a given dataset the ZIP5-level data is generally too sparse for estimating probabilities (the average sample size is very small, therefore, by (3), most estimates will be very close to the prior probability) therefore, it may seem appropriate to estimate probabilities at the ZIP3 level. However, if the data contains a particularly large proportion of California ZIP codes, it may be reasonable to work at the ZIP5 level for those records, while for some other regions even the ZIP3 level is too sparse, and may be better to aggregate even further. A simple extension of (3) allows the preprocessing scheme to automatically aggregate hierarchical codes at the appropriate level based on the size of the cells.

Consider the probability estimation formula (3) (the same approach is applicable to continuous targets) for a categorical attribute like ZIP code: The ZIP5 level formula reads:

$$S^5{}_i = \lambda(n_i)\frac{n_{i1}}{n_i} + (1-\lambda(n_i))\frac{n_Y}{n_{TR}}. \qquad (11)$$

The formula estimates the target probability for a ZIP5 cell value as the blending between the frequency-based target probability in the cell and the prior probability $\frac{n_Y}{n_{TR}}$. Instead of choosing the prior probability of the target as the "null hypothesis", it is reasonable to replace it with the estimated probability at the next higher level of aggregation in the attribute hierarchy, in this case ZIP4. Thus, the new formula reads:

$$S^5{}_i = \lambda(n_i)\frac{n_{i1}}{n_i} + (1-\lambda(n_i))S_i{}^4, \qquad (12)$$

by expanding $S^4{}_i$ using (3) we obtain*:*

$$S^5{}_i = \lambda(n^5{}_i)\frac{n^5{}_{i1}}{n^5{}_i} + \left(1-\lambda(n^5{}_i)\right)\cdot\left(\lambda(n^4{}_i)\frac{n^4{}_{i1}}{n^4{}_i} + \left(1-\lambda(n^4{}_i)\right)\frac{n_Y}{n_{TR}}\right), \quad (13)$$

which, in a simplified notation reads:

$$S^5{}_i = \lambda^5 P^5_i + \left(1-\lambda^5\right)\cdot\left(\lambda^4 P^4_i + \left(1-\lambda(n_i{}^4)\right)\frac{n_Y}{n_{TR}}\right) \qquad (14)$$

It should be easy to notice how this formulation automatically adjusts the estimate based on the density of the data across the various levels of the hierarchy. For example, if the ZIP5 sample size is sufficiently large ($\lambda^5 \cong 1$), the probability estimate will tend toward the raw estimate in the cell. However, if the sample size at the ZIP5 level is small, the formula automatically considers the sample size at the ZIP4 level, and if the aggregation leads to a large enough sample, then the ZIP4 level raw estimates is used, otherwise, the estimate converges toward the overall prior of the dependent. While the example only shows the case for two level of aggregation, its extension to more than two levels is straightforward (e.g. ZIP5 + ZIP4 + ZIP3).

The main benefit of this hierarchical approach is that is does not require an a priori decision about a fixed level of aggregation, since the weighting will account for distribution of the data at the various levels of aggregation. Furthermore, the multiplicative effect of the weights and their dependency on the cell size at each level makes the choice of the best aggregation level a "soft" decision rather than a crisp decision.

5. CONCLUSIONS

This paper presented a data transformation method for high-cardinality categorical attributes. The method leverages a robust estimation formula that finds its roots in the Empirical Bayesian framework. Because the method basically requires only aggregation functions, it can be easily carried out via native database operations. This represents a significant advantage compared to clustering methods, which require more complex algorithms and also lead to greater information reduction. Furthermore, the hierarchical version of the proposed method takes advantage of categorical data that can be aggregated to different levels of granularity, providing an effective data-driven variant aggregation scheme.

A brief review of the statistical and machine learning literature has shown that, while "shrinking" the observed probabilities toward the prior probabilities is the common theme, different forms of weighting factor $\lambda(n)$ have been proposed. Thus, a comparative experimental analysis of the various forms of $\lambda(n)$ is warranted and it will be addressed in subsequent publications.

Although probability estimation methods are nearly 50 years old, their adoption in the machine learning community is only relatively recent. Furthermore, specifically in the neural network community little has been done to address the use of high-cardinality categorical attributes with these types of models. Indeed, often practitioners simply ignore these data fields although they could provide significant value to their models. As this paper has hopefully shown, high-cardinality categorical attributes can be included in any predictive model via a statistically sound transformation that it relatively straightforward to implement. This is one more example of how well established statistical methods complement modern data mining techniques.

6. ACKNOWLEDGMENTS

I would like to thank Joydeep Gosh, Pavel Berkhin and J.T. Lehman for their helpful comments, suggestions and references provided for this paper.

7. REFERENCES

[1] Becher J.D., Berkhin P. and Freeman E., Automating Exploratory Data Analysis for Efficient Data Mining, KDD-2000, p. 424-429

[2] Carlin, B.P. and Louis T.A. Bayes and Empirical Bayes Methods for Data Analysis, New York, Chapman & Hall, 1996

[3] Cestnik B. & Bratko, On Estimating Probabilities in Tree Pruning, Proc. of European Workshop in Symbolic Learning (EWSL'91), 138-150, 1991

[4] Cestnik B., Estimating Probabilities: A Crucial Task in Machine Learning, Proc. of the 9th European Conf. on Artificial Intelligence, ECAI'90, 147-149, 1990

[5] Gnanadesikan, R., Methods for Statistical Data Analysis of Multivariate Observations, Wiley, New York, 1977

[6] Good, L.J., Probability and the weighting of evidence, London, Charles Griffing & Company Limited, 1950

[7] http://www.unica-usa.com

[8] Johnson, S.C. Hierarchical Clustering Schemes, Psychometrika, 2:241-254, 1967

[9] McCallum A., Rosenfeld R., Mitchell T. and Ng A., Improving Text Classification by Shrinkage in a Hierarchy of Classes, Proceedings of the 15th International Conference on Machine Learning, 1998

[10] Nishisato, S. Analysis of Categorical Data: Dual Scaling and Its Applications, Toronto: Toronto University Press, 1980

[11] Quinlan, J. R. C4.5: Programs for Machine Learning, San Mateo, Calif., Morgan Kaufmann, 1992

[12] Quinlan, J.R. Induction of decision trees. Machine Learning, 1:81-106, 1986

[13] Robbins, H. An empirical Bayes approach to statistics, In Proc. 3rd Berkeley Symposium on Math Statistics and Probability, 1, Berkeley, CA: University of California Press, 157-164, 1955

ABOUT THE AUTHOR:

Daniele Micci-Barreca is currently Director of Data Mining at ClearCommerce Corporation, where he is responsible for the development of fraud detection technology for Internet commerce. Previously he developed data mining applications for tax compliance and warranty fraud at Intelligent Technologies Corporation, and retail assortment planning at NeoVista Software.

He received a Ph.D. in Cognitive and Neural Systems at Boston University, a Masters degree in Computer Science from the University of Houston and a *Laurea* in Electrical Engineering from the University of Pisa.