A novel supervised quantization and smoothing method for discrete Bayes rule

Motahare Mounesan
The Graduate Center, CUNY
`cststm@gmail.com`
Done under supervison of professor Robert M Haralick

Abstract

With the rapid rate of evolution of machine learning during the last decades, many classification methods have been developed and applied. Nevertheless, a simple Bayesian classifier is still one of the most popular statistical methods. On the other hand, a wise quantization method along with a simple classifier can outperform some of the most complicated classifiers. In this work, we proposed a supervised quantization method followed by a smoothing technique. We use 10,000,000 records of a 5-dimensional data set to train a novel classifier which incredibly improves the result of a discrete Bayes classifier. In our experiments, the result has been elevated by 20 percent and reached 94.98%.

1 Introduction

A large number of machine learning and statistical techniques can only be applied to data sets composed entirely of nominal variables; However, many real-world classification tasks exist that involve continuous features where such algorithms could not be applied unless the continuous features are first discretized. The bayesian classifier is one of the popular statistical classifiers which require the estimation of probabilities and the continuous explanatory attributes are not so easy to handle, as they often take too many different values for a direct estimation of frequencies.

Continuous variable discretization received significant attention in the machine learning community during recent decades. Different discretization methods have been proposed and applied along with different classifiers. Yet, often, uniform binning of the data is used to produce the necessary data transformations for a learning algorithm, and the significant importance of an optimal discretization before the learning process is ignored. while not only does an optimal discretization improve the results of the classification, it also increases the speed of induction algorithms.

Unfortunately, the number of ways to discretize a continuous attribute is infinite. The goal of discretization is to find a set of cut points to partition the range into a small number of intervals that have good class coherence, which is usually measured by an evaluation function.

Although many discretization methods have been developed for other classifiers, the requirement for an effective discretization differs between the Bayesian classifier and others. The discretization method should result in an accurate estimate of the conditional probability of a class c given an instance x, $p(C = c|X = x)$, by substituting the categorical value of the numeric variables. As a result, the best quantization method for the Bayesian classifier has not been identified yet.

There are three different axes by which discretization methods can be classified: global vs. local, supervised vs. unsupervised, and static vs. dynamic. Local methods produce partitions that are applied to localized regions of the instance space. Global methods, such as binning, produce a mesh over the entire n-dimensional continuous instance space, where each feature is partitioned into regions independent of the other attributes. The mesh contains $\prod_{i=1}^{n} k_i$ where k_i is the number of blocks of the $i-th$ feature.

Several discretization methods do not make use of instance labels in the discretization process. In analogy to supervised versus unsupervised learning methods, we refer to these as unsupervised discretization methods. In contrast, discretization methods that utilize the class labels are referred to as supervised discretization methods.

Many discretization methods require some parameter, k, indicating the maximum number of intervals produced by discretizing a feature. Static methods perform one discretization pass of the data for each feature and determine the value of k for each feature independent of the other features. Dynamic methods conduct a search through the space of possible k values for all features simultaneously, thereby capturing interdependencies in feature discretization. The method we are presented in this paper is a global supervised dynamic quantization that uses expected gain as an evaluation function for improving its result. We will discuss our method in detail later.

We present an overview of the existing methods for feature discretization in Section 2. In Section 3, we have a brief overview of a discrete Bayes rule. In section 4, we describe in detail the methods we used for quantization and smoothing. We explain our experiments and results in Section 4. Section 5 is reserved for a discussion and summary of this work.

2 Discretization process and methods

Discretization is usually performed prior to the learning process and it can be broken into two tasks. The first task is to find the number of discrete intervals. The second task is to find the width, or the boundaries, of the intervals given the range of values of a continuous attribute. Most of the discretization algorithms perform an iterative greedy heuristic search in the space of candidate discretizations, using different types of scoring functions for evaluating a discretization.

In the simplest discretization methods, Equal Width Discretization (EWD), also called equal interval quantizing, the number line between minimum and maximum of values is divided into k intervals of equal width. A related method, Equal Frequency Intervals (EFD), also called equal probability quantizing, divides the sorted values into *k* interval so that each interval contains approximately the same number of training instances. Both EWD and EFD suffer much attribute information loss since *k* is determined without reference to the properties of the training data.(0) (0)

Entropy Minimization Discretization (EMD) evaluates as a candidate cut point the midpoint between each successive pair of the sorted values. It does the evaluation by discretizing the data into two intervals and calculating the resulting class information entropy. A binary discretization is applied recursively, always selecting the best cut point which has minimal entropy.(0)

Iterative Discretization (ID) initially forms a set of intervals using EWD or MED, and then iteratively adjusts the intervals to minimize a classifiers classification error on the training data. ID is not a feasible solution dealing with large data sets because of its immense time consumption.

Proportional k-Interval Discretization (PKID) adjusts discretization bias and variance by tuning the interval size and number and further adjusts the naive Bayes probability estimation bias and variance to achieve lower classification error. This method returns sub-optimal results for smaller data sets but it is quite effective in large ones.(0)

Weighted Proportional k-Interval Discretization (WPKID) is an improved version of PKID which weighs discretization variance reduction more than bias reduction by setting a minimum interval size to make more reliable the probability estimation. It mitigates the disadvantage of PKID and keeps its advantages.(0)

Lazy Discretization (LD) defers discretization until classification time estimating $p(X_i = x_i|C = c)$ for each attribute of each test instance. For the value of X_i from the test instance, it selects a pair of cut points such that the value is in the middle of its corresponding interval whose size is the same as created by EFD with k = 10. LD tends to have high memory and computational requirements because of its lazy methodology. (0)

Non-Disjoint Discretization (NDD) forms overlapping intervals for X_i, always locating a value x_i toward the middle of its corresponding interval $(a_i, b_i]$.(0)

3 Discrete Bayes Rule

Bayes theorem provides an optimal way to predict the class of an unseen instance described by a conjunction of attribute values $X = x_1 \wedge x_2 \wedge \wedge X_n$. The predicted class is the one with the highest probability given the instance X:

$$P(c_i|X) = \frac{P(c_i)P(X|c_i)}{P(X)} \qquad (1)$$

The application of this formula in machine learning is restricted by the inability to determine accurate values for $P(X|c_i)$. In standard machine learning applications, these probabilities must be estimated from the training data. If there were sufficient randomly sampled examples of every possible combination of attribute values, such estimation would be straightforward and acceptably reliable and accurate. Furthermore, if $P(X)$ is not given, we can always calculate it through the marginal probability formula

$$P(X) = \sum_{c_i \in C} P(c_i, X) = \sum_{c_i \in C} P(X|c_i)P(c_i) \tag{2}$$

In both equation (1) and (2) the $P(c_i)$ is the prior probability of the class c_i and it is better to get it from the prior knowledge of the world that we are doing classification for, rather than calculating it directly from the data set.

The important point that is worth mentioning here is that if we use the given prior probability we should calculate $P(X)$ using the equation (2), and if we calculate the prior probability from the dataset, we should use the $P(X)$ that is directly achieved by counting the training set.

As we mentioned before a Bayesian decision rule assign the most probable class to a given instance and it is made by the following equation

$$f_d = argmax_{c_j \in C} P(c_j|X) \tag{3}$$

After calculating the decision rule using train data, the test set is used in order to evaluate the results. Expected gain is one of the most sound evaluation methods for classification tasks which will be defined here.

In statistical classification, a confusion matrix is a specific table layout that allows visualization of the performance of a supervised learning algorithm. Each row of the matrix represents the instances in a predicted class while each column represents the instances in an actual class.

Gain matrix or economic gain matrix is an $|C| \times |C|$ matrix, with the same size of confusion matrix, for the classification task that defines the gain is achieved by assigning a right class label to a given instance or the loss for assigning a wrong class label to an instance.

Let $e_{|C| \times |C|}$ be the economic gain matrix and $P_{TA_{|C| \times |C|}}$ be the confusion matrix of the decision rule *f*, the expected gain of the this decision rule can be calculated with the following equation:

$$E[e] = \sum_{j=1}^{K} \sum_{k=1}^{K} e(c_j, c_k) P_{TA}(c_j, c_k) \tag{4}$$

4 Our Optimization and Smoothing method

In this section, we described our proposed method in detail. Our learning algorithm first starts by dividing the dataset into 3 parts: a training set, a development set, and a test set. We use the training set for finding the number of levels and then we use both train and development sets in the optimization and smoothing process. In the optimization process, we repeatedly perturb a boundary and train a discrete Bayesian rule on the train set and by applying the rule on the development set we check improvements. The smoothing process is pretty similar to optimization in terms of using datasets. Finally, we apply our optimal classifier to the test set and report the results.

4.1 Calculating number of levels in each dimension

The first step is to identify the number of quantized levels that we have in each dimension. Given a dimension, applying a naive equal width discretization could be an initial step to calculate unbiased entropy that gives us the number of levels in that dimension. For a particular dimension, considering the range of data in that dimension we can discretize our dataset into *k* equal width bins. This *k* could be chosen in the interval of [10,10000] based on the size of the dataset but our experiments reveal that it does not make any difference at least in our case.

Let us consider a dataset with J dimensions, $X = X_1, .., X_J$ is a form of an observation in this dataset. By applying equal width discretization on *j-th* dimension, we will have k initial bins $Z_j = z_{j1}, z_{j2}, ..., z_{jk}$. Let m_k, $m_k = \#\{n|x_n = k\}$, be the frequency of observations in each of these k bins, we can calculate the probability of each bins by dividing frequencies by the total number of observation in the train set, $p_k = m_k/N$. Substituting the probabilities in equation (5), we can calculate the unbiased entropy for the $j-th$ dimension.

$$\hat{H} = -\sum_{k=1}^{K} p_k \log_2 p_k + \frac{n_0 - 1}{2N log_e 2} \tag{5}$$

where n_0 is the number of zero counts bins.

Besides, suppose that M is the total number of bins for all of the dimensions which should be less or equal to the number of memory cells that we have to apply the process. Then there is a relation between the number of levels that we can have for each dimension and M as follows

$$M = \prod_{i=1}^{J} L_i \tag{6}$$

,where L_i is the number of levels for *i-th* dimension.

Let $\hat{H}_j$ be the entropy of the *j-th* component of dimension components calculated with equation (5). And let f_j of the *j-th* dimension be $f_j = \frac{\hat{H}_j}{\sum_{i=1}^{J} \hat{H}_i}$. Then the number of levels for *j-th* dimension is calculated by $L_j = \lceil M^{f_j} \rceil$.

Next, by fixing the number of levels in each of the dimensions we can apply an equal-frequency discretization on all of the dimensions.

Let $z_{(1)j}, ..., z_{(N)j}$ be the N values of the *j-th* component of data in ascending order. Then $[z_{(1)j}, z_{(\frac{N}{L_j}+1)j}, ..., z_{(\frac{kN}{L_j}+1)j}, z_{(\frac{(L_j-1)N}{L_j}+1)j}]$ is the initial L_j-level quantization which is the input to optimization algorithm that we will explain in next section.

4.2 Optimizing the boundaries

Having those initial L_i-level quantization for each of the dimensions, we optimize the boundaries based on algorithm 1 which is proposed in professor Haralick's lectures. In this algorithm, we repeatedly perturb a random boundary in a random dimension and seeking the highest expected gain that we get by applying the obtained decision rule on the development set. We repeat this process until no perturb can make the result better than it is. That is the point that our boundaries are fixed and we can move to the smoothing step.

4.3 Smoothing

In predicting a value by a classifier we always face unobserved or uncommon values. If the sample size is not large enough, the maximum likelihood estimation probability estimates may not be representative. To alleviate the sparseness problem we find those hyperspaces that do not have enough samples and apply a smoothing technique on them. We need to fix the parameter k, which is the minimum number of samples that is enough for our classifier to predict well.

Let bin *m* have volume v_m and count t_m, when volume is the multiplication of edges of the bin in each dimension. And let $m_1, ..., m_I$ be the indexes of the I closest bins to bin *m* satisfying $\sum_{i=1}^{I} t_m \geq k$ and $\sum_{i=1}^{I-1} t_m < k$.

Further, let define $b_m = \sum_{im}^{I} t_m$ and $v_m^* = \sum_{i=1}^{I} v_{m_i}$. Density of each point in bin *m* is $\frac{\alpha b_m}{v_m^*}$, and since the volume of the bin is v_m the smoothed probability of the bin *m* will be $p_m = p(\frac{\alpha b_m}{v_m^*})v_m$.

We need to set α so that the density integrates to 1, i.e. $1 = \sum_{m=1}^{M} \frac{\alpha b_m v_m}{v_m^*}$. As a result, we can substitute α in the smoothed probability formula, with $\alpha = \frac{1}{\sum_{m=1}^{M} \frac{\alpha b_m v_m}{v_m^*}}$.

The proposed values for parameter *k* in Professor Haralick's slides are $k = jN/20M$, with $j1, 2, 3, 4$, where N is the number of observations and M is number of total bins. We will apply the smoothing technique 4 times. Each time we will smooth the probabilities of bins using a train set and then making a decision rule. Then we will apply the decision rule on the development set and compare the obtained expected gains with each of the assigned values. Then we pick the *k* with the highest expected gain and fix it.

By fixing both boundaries and smoothing parameters, our improvements on the decision rule have been finished and we can test our classifier on the test set.

5 Experimental Protocol and Results

In this section, we describe the details of implementation and experiments and finally report the results.

5.1 Dataset

The dataset we are experimenting on consists of 10,000,000 records with 5-dimensions. Each of these points has been labeled as class 0 or class 1. The prior probability of each of the classes has been given and is equal to 0.4 for class *0*, and 0.6 for class *1*, i.e. in the world that the decision rule will be qualified in, 40% of the instances belongs to class 0 and the remaining 60% belongs to class 1.

Furthermore, this dataset has been simulated for this purpose. It means that instances have been generated in a way that if the optimization algorithm works well it can achieve 100% accuracy. As a result, the best possible result is the accuracy of 100% and expected gain of *2.2*.

We divided the data set into three parts, the first one-third of the data is used for training, the second-third is used for development, and the last

Algorithm 1 Optimize The Quantizing Boundaries

```
1:  repeat
2:      Randomly choose a component j and an interval k
3:      Randomly choose a small perturbation $\delta$ ( can be positive or negative)
4:      Randomly choose a small integer M (No collision with neighboring boundaries)
5:      $b_{k_j}^{new} = b_{k_j} - \delta(M+1)$
6:      for $m = 0; m < 2M; m++$ do
7:          $b_{k_j}^{new} = b_{k_j}^{new} + \delta$
8:          Compute New Probabilities
9:          Recompute Bayes Rule
10:         Save expected gain
11:     Replace $b_{k_j}$ by the boundary position associated with the highest gain
12: until no change
```

one-third is used for evaluating the classifier performance. This division has been done randomly but with a constant seed which lets us replicate and compare the results of different runs.

In both the optimization and the smoothing processes, the training set is used to train a classifier, and then the development set is used to evaluate the classifier on the second part of the data. We try different boundaries in the optimization process and different smoothing parameters in the smoothing process through this perturbation and evaluation pairs. Finally, when we come up with our best boundaries and optimal smoothing parameter, we test our classifier on the third part of the data and report the results. This helps us to achieve an unbiased estimate of the confusion matrix, the probability of correct classification, and the expected gain.

5.2 Decision Rule

The decision rule has been calculated by counting the number of instances that fall in each of the bins and applying the Bayes relation on it. The prior probabilities have been given with the dataset and their values have been mentioned in the previous section. The economic gain of this classification task is the following matrix

$$EconomicGain = \begin{pmatrix} 1 & -1 \\ -2 & 3 \end{pmatrix}$$

which means the award of assigning a true label to class 0 and 1 are 1 and 3, respectively. The penalty for assigning the wrong label to class 1 is -2 and the penalty for assigning the wrong label to class 0 is -1.

The process of calculating probabilities and making the decision rule has been examined with a small self-created example that showed all the calculations are exact.

5.3 Quantization

An initial equal interval quantizing method has been applied to the training set in order to calculate the initial probabilities needed for determining the number of levels for each dimension. Though different equal interval boundaries have been experimented with, there was not any difference between the number of levels that resulted from them. In our implementation, we split the range into 10000 equal intervals.

The number of levels in each dimension is calculated through the entropy-based method explained in section 4 with a small modification which is considering the floor of $L(f_j)$ as the number of levels for *i-th* dimension instead of the ceiling. Number of levels are $k =< 6, 6, 6, 6, 6 >$ which means in each of the dimensions we have 6 bins.

The reason behind the modification is that the dataset has been generated with 6 boundaries in each dimension and the same number of bins give us the chance to compare the results better and track the rate of optimization. On the other hand, since the computational complexity of the process is relatively high, having fewer bins speeds up the process.

Having 6 bins in each dimension results in the total number $6^6 = 7666$ bins which are mapped to a memory address with algorithm **??**. In this algorithm, a tuple $< B0_a, B1_b, B1_c, B1_d, B1_e, >$ is mapped to the unique memory cell which keeps the counts of points fall in each bin and later is used to calculate the decision rule.

For each dimension, the unique values are

Dimension	Boundary(0)	B(1)	B(2)	B(3)	B(4)	B(5)	B(6)
0	0.0000	0.16636	0.33378	0.50134	0.668	0.83327	1.0000
1	0.0000	0.16504	0.33191	0.49908	0.66538	0.83164	1.0000
2	0.0000	0.16726	0.33475	0.50072	0.66887	0.8348	1.0000
3	0.0000	0.16574	0.33455	0.50153	0.66765	0.83463	1.0000
4	0.0000	0.16619	0.33215	0.49911	0.66566	0.8323	1.0000

Table 1: The initial boundaries determined by equal probability quantization. This boundaries are input of optimization algorithm.

Dimension	Boundary(0)	B(1)	B(2)	B(3)	B(4)	B(5)	B(6)
0	0.00000	0.02037	0.18117	0.56374	0.90676	0.98718	1.00000
1	0.00000	0.47436	0.49150	0.53603	0.58948	0.62023	1.00000
2	0.00000	0.08321	0.15193	0.24154	0.60653	0.94493	1.00000
3	0.00000	0.02697	0.22574	0.25494	0.32456	0.85316	1.00000
4	0.00000	0.06348	0.10362	0.19162	0.51352	0.60375	1.00000

Table 2: Optimal boundaries that dataset has been generated based on them. These boundaries lead to accuracy of 100% and expected gain of 2.2.

sorted using Python embedded sort method and equal frequency quantizing is applied. The output of this process is the initial quantization which we calculate probabilities of each of the 7776 bins based on which. Then a discrete Bayes decision rule is calculated which is the baseline of our optimization process.

To speed up the optimization process we used 1%, 10%, and 100% of the dataset iteratively to optimize the boundaries. The important point here is that we should be careful not to use the test data in the optimization process. As a result, we shuffle and split the whole (100% of) dataset at first and then choose 1%,10%, and 100% of each training and development set to optimize quantizing boundaries.

In each iteration, the random optimization process continues perturbing boundaries until it converged. After 100 rounds of perturbation without any improvements, we stop random optimization. An ordered optimization algorithm is then applied to perturbed the boundaries in order. The original version of inorder optimization should come back to random optimization after the first change, but we wait until we go through all the boundaries. Each iteration will stop when the last 100 perturbation of random optimization and the 25 perturbation of ordered optimization does not show any improvements.

The best accuracy on the development set is 94.98% so far and the expected gain is 1.825. The result of the test set is shown in table 4.

5.4 Smoothing

In this step the bin probabilities are smoothed with the method we proposed in section 4.3.

As we mentioned before, each time the likelihood probability is smoothed by parameter k on the train set, and the results are compared by applying the new classifier to the development set. The result of the 4 iterations of our experiments have been entered in table 5. Unfortunately, on this scale smoothing did not show any improvement

[[1589720, 72418], [79413, 1591782]] 6 0.954 1.84

The final step was to test our classifier on the test set. We applied our discrete Bayes rule classifier on the test set of size 3,333,333 which is almost one-third of our data set. And we achieved the expected gain of and the accuracy of

We believe our classifier is stuck in local optimal quantization and it is not the best result that we could achieve. Nevertheless, we are working on another run of the optimization function and we will be reported when better results are achieved.

6 Conclusion and Future Work

Most of the popular classifiers work well with discrete values but when it comes to continuous values they do not perform well. While most of the real-world applications are continuous attributes. Discretization is the method we deploy to adapt the real-world data to discrete classifiers.

Dimension	Boundary(0)	B(1)	B(2)	B(3)	B(4)	B(5)	B(6)
0	0.0	0.022	0.18188	0.564	0.566	0.906	1.0
1	0.0	0.47461	0.49	0.53458	0.58957	0.61999	1.0
2	0.0	0.08492	0.154	0.24	0.606	0.944	1.0
3	0.0	0.028	0.25498	0.324	0.854	0.99897	1.0
4	0.0	0.07	0.18984	0.508	0.514	0.602	1.0

Table 3: Final quantization achieved by our optimization algorithm. These boundaries lead to accuracy of 94.98% and expected gain of 1.825.

	accuracy	expected gain
Development set	94.98%	1.825
Test Set		

Table 4: The expected gain and accuracy of development set and test set after optimization process.

j	k	accuracy	expected gain
1	21	84.86	1.502
2	43	84.82	1.502
3	64	84.79	1.499
4	86	84.77	1.499

Table 5: The expected gain and accuracy results after smoothing with different k

Our experiments demonstrated the incredible effect of the optimal quantization method with a simple discrete Bayes rule classifier. We applied a supervised quantization technique, which finds optimal boundaries by evaluating the result with the expected gain of a discrete Bayes classifier. Furthermore, we use a smoothing method to improve the probability estimates even more. Results support the improvement claim since the difference between a simple equal-frequency discretization and our optimal quantization is around 20 percent.

The experiments also showed that in some cases quantization method stuck in some local optimum solutions. In the future, we can try modification on the perturbation part of the optimization algorithm and make it search all the lines from the previous boundary to the next boundary for the picked boundary.

Another drawback of the system is that dealing with large data set is not efficient in terms of performance. In the future, we will deploy parallel computing techniques as well.

Appendix

The project has been coded in *python 2.7.pandas* and *numpy* packages are two python packages is used. The project packages consist of 4 files as follows

- *BayesDecisionRule.py* which contains all the needed functions to calculate probabilties and create the classifier.
- *Optimization.py* which contains the optimization and updating functions
- *Smoothing.py* which contains the smoothing functions
- *Main.py* which is the main core of the program which is need to be run. Every other function and file are called inside this code. As a result the running of the project is pretty easy by just setting the address of the data in the first line of the main file and run it.

References

Z. Zheng and G.I. Webb. Lazy Learning of Bayesian Rules Machine Learning 41: 53. https://doi.org/10.1023/A:1007613203719, 2000.

S. Kotsiantis, and D. Kanelloupos. Discretization techniques: A recent survey GESTS International Transaction on Computer Science and Engneering 32.1, 47-58, 2006.

J. Dougherty, R. Kohavi, and M. Sahami. Supervized and unsupervised discretization of continuous features Machine Learning: proceedings of the twelfth international conference, Vol. 12, 1995.

R. Haralick. Machine Learning Lectures http://haralick.org/ML/lecture_slides.html , 2018.