Discrete Bayes Decision Rule Machine Learning Project

Subhadarshi Panda[1] and Shweta Garg[2]

***Abstract*— Discrete Bayesian decision rule is one of the foundation techniques of machine learning for solving classification problems. We use discrete Bayes rule on a sufficiently large dataset with a fixed memory size constraint M and train a classifier. We improve the classifier by a quantization optimization and smoothing.**

I. INTRODUCTION

Pattern recognition is the science of making inferences from perceptual data, using tools such as statistics and probability. In Machine Learning, classification is a problem of finding the class or label for a particular data sample. This is usually done after training a classifier with some labeled data. The goal of a classifier is to partition feature space into class decision regions.

Bayes' theorem[1] is a formula that describes how to update the probabilities of hypotheses when given evidence. It follows simply from the axioms of conditional probability. It can be used powerfully for a wide range of problems in Machine Learning and is the foundation of Bayes decision rule.

Bayesian decision theory[4][5] is a fundamental statistical approach to the problem of classification. Here it is assumed that the decision problem is posed in probabilistic terms and all relevant probability values are known. Knowing all possible probability values rarely occurs in practice. However, prior probabilities can be used to determine the optimal (Bayes) classifier against which we can compare all other classifiers.

The Bayes decision rule is used extensively in many Machine Learning problems. It has been applied to problems related to climate change, sensor planning, image classification, statistical machine translation and many other novel problems.

We do experiments on a given dataset and built a classifier with maximum expected gain.[1] We use the given prior probabilities $p_0 = 0.4$ and $p_1 = 0.6$, for class 0 and class 1 respectively. To achieve maximum expected gain, we optimize the quantizer. Further we find out expected gains for different values of smoothing parameter k. We use the memory constraint $M = 10000$ for all the experiments. On testing our best classifier model on the test dataset, we get accuracy upto 96.46%.

[1]Student at the Graduate Center, CUNY, New York `spanda at gradcenter.cuny.edu`

[2]Student at the Graduate Center, CUNY, New York `sgarg at gradcenter.cuny.edu`

[1]The dataset can be found at http://haralick.org/ML/pr_data.

II. PROBLEM STATEMENT

The problem is to build a classifier using discrete Bayes decision rule for a given dataset. The problem also involves finding optimum quantization for the classifier and subsequently performing a technique called bin smoothing.

We are given a training dataset D which has N data samples $(d_1, d_2, ..., d_N)$. Each data sample d_i has n feature values and a class c. The goal is to train a classifier which will maximize the expected gain. The number of classes K is given an input. The economic gain matrix which has size $K \times K$ is also given. Other inputs are the prior probabilities $P(c)$.

To build a classifier which uses not more than a specific memory size, it is necessary to quantize the values of each given data sample, i.e., for each dimension. Suppose for dimension i, we have L_i quantized levels. Then the data samples need to be mapped to a measurement space D where $D = \prod_{i=1}^{n} L_i$.

In the problem formulation, it is important to make use of linear addresses to access each hypercube in the measurement space D. So it is necessary to create a function which outputs the corresponding linear address given a position of the hypercube as tuple.

From the class conditional probabilities $P(d|c)$ and the prior probabilities $P(c)$, the posterior probabilities $P(c|d)$ can be calculated using the Bayes Theorem. Then the discrete Bayes rule can be applied to obtain the decision rule for classification.

The decision rule can be applied on another dataset (called the validation set) to get a confusion matrix, which is a $K \times K$ matrix containing probabilities telling how correct is the machine learning model. Using the confusion matrix and the gain matrix (which is given as input), the expected gain can be calculated.

The quantizers used above can be optimized so as to get maximum expected gain. Further optimization can be done using bin smoothing, a technique to smooth bin probabilities. This optimization is done after fixing the quantization boundaries.

To find the decision rule, fair game assumption is used, i.e., the decision rule must only use the measurement data d and cannot use the true class identification. Given d, the assigned class which the decision rule determines and the true class which nature determines are statistically independent.

III. METHODOLOGY

A. Bayes Theorem

In machine learning we are often interested in selecting the best hypothesis c given data d. In a classification problem, our hypothesis c may be the class to assign for a new data instance d.

One of the ways of selecting the most probable hypothesis given the data that we have that we can use as our prior knowledge about the problem. Bayes Theorem[2][3] provides a way that we can calculate the probability of a hypothesis given our prior knowledge. Bayes Theorem is stated as:

$$P(c|d) = \frac{P(d|c)P(c)}{P(d)} \tag{1}$$

B. Discrete Bayes Decision Rule

Let $e(c^j, c^k)$ denote the economic gain if true class is c^j and assigned class is c^k. Then a decision rule f is called a Bayes decision rule iff $E[e; f] \geq E[e; g]$ for any decision rule g where,

$$E[e; f] = \sum_{d \in D} \sum_{k=1}^{K} f_d(c^k) \left[\sum_{j=1}^{K} e(c^j, c^k) P_T(c^j, d) \right] \tag{2}$$

In the above equation, $P_T(c^j, d)$ denotes the joint probability where c^j is the true class. The above equation in other words means that for each d, there exists a unique m such that for each k (where c^k is the assigned class),

$$\sum_{j=1}^{K} e(c^j, c^k) P_T(c^j, d) \leq \sum_{j=1}^{K} e(c^j, c^m) P_T(c^j, d) \tag{3}$$

The above inequality should be valid for each k. Then a decision rule can be obtained as follows:

$$fd(c_m) = 1 \tag{4}$$

$$fd(c_k) = 0, k \neq m \tag{5}$$

This is one definition among many reasonable definitions of decision rule.

C. Confusion Matrix

A confusion matrix is a matrix used to describe the performance of a classifier on a set of test data for which the true values are known. The table has probability values $P(c^j, c^k)$, where c^j is the true class and c^k is the assigned class. The probabilities $P(c^j, c^k)$ are found as follows

$$P(c^j, c^k) = \sum_{d \in D} f_d(c^k) P(c^j, d) \tag{6}$$

D. Expected Gain

Given a gain matrix e, the expected gain $E(e)$ is

$$E[e] = \sum_{j=1}^{K} \sum_{k=1}^{K} e(c^j, c^k) P(c^j, c^k) \tag{7}$$

where c^j is the true class and c^k is the assigned class.

E. Initial Quantization

The dataset used to train the classifier usually has real values for each feature. This may require a huge memory space for saving the class conditional probabilities. Therefore, before applying the discrete Bayes decision rule it is important to quantize it and have digital possibilities for each real number. Then the quantized values are used for calculating class conditional probabilities.

To start with, we assign every real feature value to one of $N/10$ quantization levels. Here, N is the number of data samples in the training data. We uniformly quantize, i.e., the quantization boundaries equispaced.

Let M be the total memory size we want to use. We calculate the entropy by using an unbiased entropy estimator for calculating the number of bins, so that our class conditional probability will use memory space of M bins. In other words, the total number of quantizing bins for all dimensions are M.

Suppose we observe for each dimension the training data samples $x_1, x_2, ..., x_N$, where each $x_n \in \{1, ..., K\}$. Here K is the number of quantization levels for each dimension. So $K = N/10$. We count the number of occurrences m_k for each bin k.

$$m_k = \#\{n|x_n = k\} \tag{8}$$

From the above the bin probability p_k can be calculated as follows

$$p_k = \frac{m_k}{N} \tag{9}$$

The number of zero counts n_0 is

$$n_0 = \#\{k|m_k = 0\} \tag{10}$$

We then calculate $\hat{H}_j$, an unbiased estimator of entropy for the dimension j.

$$\hat{H}_j = -\sum_{k=1}^{K} p_k \log_2 p_k + \frac{n_0 - 1}{2N \log_e 2} \tag{11}$$

We define f_j for each dimension j as

$$f_j = \frac{\hat{H}_j}{\sum_{j=1}^{J} \hat{H}_j} \tag{12}$$

In the above equation, J is the total number of dimensions.

Let L_j denote the number of bins for the jth feature. Given the memory M, it can be calculated as

$$L_j = M^{f_j} \tag{13}$$

To get an integer value for L_j, we use the floor function. We could also use the ceiling function which results in an integer greater than or equal to L_j.

To calculate bin boundaries before computing the decision rule, for the jth feature we sort the data in ascending order and set the boundary of bin interval in a way that each interval will carry equal number of data points. For instance,

the kth quantizing interval $[a_{jk}, b_{jk})$ for the jth component is defined by

$$a_{jk} = d_{((k-1)N/(K+1))j} \tag{14}$$

$$b_{jk} = d_{(kN/(K+1))j} \tag{15}$$

F. Quantization Optimization

The bin boundaries initially obtained can be perturbed to see if a higher expected gain can be obtained.

First we quantize the data to use a fixed memory space. Our aim now is to get the bin boundaries in a way which will give us a Bayes decision rule whose expected gain is maximum among all possible expected gains calculated by perturbing the bin boundaries.

The process for boundary optimization is as follows: For each dimension and each interval boundary, we perturb the boundary and use training dataset to determine the decision rule. Then we apply the decision rule on validation dataset to estimate the expected gain. The above procedure is repeated until there is no improvement in expected gain.

The process of boundary optimization consists of two parts. We describe these parts as (a) random perturbations and (b) sequential check.

(a) Random Perturbations: We pick a dimension randomly and then pick a boundary for that dimension randomly. We perturb the picked boundary by picking a random point between an interval defined by the the mid points of the picked boundary with the two adjacent boundaries. If the leftmost boundary is picked, we consider 0 as the previous boundary. Similarly, if the rightmost boundary is picked, we consider 1 as the next boundary. Upon perturbing the picked boundary, we build the decision rule using the training dataset and find the expected gain using the validation dataset. If the perturbation improves the expected gain, we assign this point to be the new boundary. Otherwise, we do not perturb the picked boundary. Then we repeat the whole process again by picking a random dimension and then a random boundary for that dimension. We come to a halt only when we see the expected gain does not increase for $patience$ number of iterations, where in each iteration we choose a random boundary for a random dimension. We use $patience = 100$ for our experiments.

(b)Sequential Check: After the random perturbation part is done, we choose a dimension and quantizer boundary in order and perturb the boundary half way in both directions. We find the distance between the mid points of the current boundary with the previous and next boundaries. To perturb, we first find a δ by dividing this distance by a number. This number can be chosen based on our choice of number of perturbations. In our experiments, we find δ by dividing by 10 and it gives us 11 perturbations for each boundary. We calculate the expected gain every time we perturb and see if there is an improvement in expected gain. If there is no improvement for all the boundaries for all dimensions, we assume we have reached convergence and found the optimal bin boundaries. On the other hand, if there is an improvement in the expected gain for any perturbation, we assign that perturbation to be the new boundary and go back to the random perturbation part (Part (a) explained above) and start all over again.

G. Bin Smoothing: Density Estimation Using Fixed Count k

So far we have not taken into account the uniform distribution of probabilities of M bins, but it is possible that many of the bins have probability 0. To avoid this scenario, we try to find a smoothing parameter k. Moreover, we cannot choose k to be very small or very large otherwise we will end up with over-fitting or under-fitting respectively. We choose $k = jN/20M$, where $j \in \{1, 2, 3, 4\}$. For each bin m, we find the volume v_m and the count t_m. Let $m_1, m_2, ..., m_I$ be the indexes of the I closest bins to bin m satisfying

$$\sum_{i=1}^{I} t_{m_i} \geq k \tag{16}$$

and,

$$\sum_{i=1}^{I-1} t_{m_i} < k \tag{17}$$

We calculate the updated number of counts b_m for each bin using

$$b_m = \sum_{i=1}^{I} t_{m_i} \tag{18}$$

and the updated volume V_m^* for each bin using

$$V_m^* = \sum_{i=1}^{I} v_{m_i} \tag{19}$$

The bin probability p_m is

$$p_m = (\alpha b_m / V_m^*) v_m \tag{20}$$

where $\alpha = \frac{1}{\sum_{m=1}^{M} b_m v_m / V_m^*}$ to ensure that all bin probabilities sum to 1.

Finally, we fix $k^* = k$ which maximizes the expected gain.

IV. EXPERIMENT

The given dataset has $10,000,000$ data points. Each point consists of 5 features and each feature value is a number in the range $[0, 1]$ with a precision of five decimal places. Moreover, each data point belongs to either class 0 or class 1. Given prior probability of class 0 is 0.4 and that of class 1 is 0.6 and the economic gain matrix is

$$\begin{bmatrix} 1 & -1 \\ -2 & 3 \end{bmatrix}$$

First, we divided the dataset randomly into three equal parts namely training dataset, validation dataset and test dataset. For our experiments we used random seed 1234 for data splitting. We labeled the three parts training, validation and test datasets respectively. Recall that our goal is to use

only the training dataset and validation dataset to build our classifier and finally, calculate the final expected gain on test dataset.

To check our Bayes classifier, we used the toy example used in page #50 in [2]. We obtained the same results upon using the same inputs as in the toy example.

On training dataset, we calculated the number of quantizing levels for each dimension as well as equal probability quantizers for each dimension by choosing $N/10$ quantization levels for each dimension. Then we quantized the training dataset by using the interval boundaries obtained. We calculated the probabilities $P(c|d)$ as well as calculated the decision rule on quantized training dataset by using Bayes theorem and bayes decision rule respectively.

We quantized the validation dataset by using the boundaries generated, calculated the probabilities $P(c|d)$ and further applied the decision rule on quantized validation dataset for estimating expected gain.

To optimize the bin boundaries, we chose a dimension and quantizer boundary and perturbed the boundary half way in both directions. We performed both parts: random perturbations and sequential check, as mentioned in *Quantization* subsection in section III. In our experiments, we used random seed 1237 for quantization optimization. For each perturbation we quantized the training dataset and calculated the decision rule, applied it on the validation dataset after quantizing the validation dataset and found the expected gain. In the sequential check part, we perturbed each boundary of each dimension 11 times by uniformly dividing the interval into 10 small intervals.

After quantization, we got fixed quantizers to proceed to the next step.

Now we optimized the smoothing parameter k. We quantized the training dataset and test dataset by using the fixed quantizers generated above. We used $k = jN/20M$, where $j \in \{0, 1, 2, 3, 4\}$ and calculated the smoothed probabilities on quantized training dataset, followed by computation of decision rule. We then applied the decision rule on quantized validation dataset and estimated the expected gain. We finally chose the k value which gave the maximum expected gain on the validation dataset. We saw in our experiments that setting $j = 0$, or equivalently, $k = 0$, gave us same results as without smoothing, as we expected.

Then we quantized the test dataset by using the quantizer we have obtained. We used the decision rule obtained by the optimum k to estimate the confusion matrix, expected gain and accuracy for the test dataset.

V. RESULTS

We used the training dataset to find the quantization levels for the 5 dimensions. For the memory space $M = 10,000$, we found the number of quantization levels to be $[6, 6, 6, 6, 6]$. For finding integer value for L_j's we used the floor function. We could have also used the ceiling function. We used the floor function because (a) product of all L_j's gives us a number less than $M = 10000$, and (b) we wanted to compare our boundaries with those given in the website[3]. Before the optimization of quantization boundaries, we obtained the following boundaries

$$\begin{bmatrix} 0.1662 & 0.3329 & 0.4996 & 0.6664 & 0.8331 \\ 0.1666 & 0.3333 & 0.5001 & 0.6665 & 0.8333 \\ 0.1668 & 0.3332 & 0.5001 & 0.6668 & 0.8333 \\ 0.1664 & 0.3333 & 0.4995 & 0.6662 & 0.8331 \\ 0.1667 & 0.3332 & 0.4998 & 0.6670 & 0.8334 \end{bmatrix}$$

, where each row is for a dimension.

For the quantization optimization, we observed 14 sequential checks. After the first 13 sequential checks, we went back to random perturbations part to start all over. We achieved convergence only after the $14th$ sequential check. It took 73 hours for convergence. After the optimization of quantization boundaries, we obtained the following boundaries

$$\begin{bmatrix} 0.0204 & 0.1811 & 0.5637 & 0.9057 & 0.9871 \\ 0.4774 & 0.4916 & 0.5360 & 0.5895 & 0.6213 \\ 0.0835 & 0.1504 & 0.2421 & 0.6088 & 0.9443 \\ 0.0271 & 0.2259 & 0.2550 & 0.3242 & 0.8545 \\ 0.1034 & 0.1896 & 0.5116 & 0.6029 & 0.7600 \end{bmatrix}$$

, where, again, each row is for a dimension.

We compared our boundaries with the boundaries provided in the website. We made surprising observations. We found that our boundaries for the first 4 dimensions are same or in the vicinity of the best boundaries given in the website. For the fifth dimension 4 of our 5 boundaries are nearly same as those in the website.

We found the expected gain and accuracy on the validation dataset for different values of smoothing parameter k. The results are summarized in TABLE I. Best results are obtained for $k = 0$, or in the case when there is no smoothing. Upon using $k = 0$ on the test dataset, we found the confusion matrix to be

$$\begin{bmatrix} 0.4699 & 0.0311 \\ 0.0043 & 0.4947 \end{bmatrix}$$

, the expected gain to be 1.9144 and the accuracy to be 96.46%.

k	Expected Gain	Accuracy (%)
0	**1.9149**	**96.47**
21	1.6857	84.97
42	1.6332	82.38
63	1.5899	80.25
84	1.5693	79.29

TABLE I: Expected gain and accuracy on validation dataset for different values of smoothing parameter k

VI. CONCLUSION

We used the Bayes decision rule to train a classifier on a given dataset. We improved our trained model by optimizing the quantizers. We applied bin smoothing and found out

[2]http://haralick.org/ML/discrete_bayes.pdf

[3]http://haralick.org/ML/quantizing_boundaries.txt

expected gain variations upon changing smoothing parameter k. We used a memory space parameter M while training. Using this classifier we are are able to get 1.9144 expected gain and 96.46% accuracy on the test dataset. We believe that this project gave us a meaningful and practical learning experience. We also believe that we have developed a ready-to-use toolkit for any classification problem in Machine Learning.

REFERENCES

[1] Bayes, Thomas & Price, Richard (1763). "An Essay towards solving a Problem in the Doctrine of Chance. By the late Rev. Mr. Bayes, communicated by Mr. Price, in a letter to John Canton, A. M. F. R. S." (PDF). Philosophical Transactions of the Royal Society of London. 53 (0): 370418. doi:10.1098/rstl.1763.0053.

[2] Richard Price (1991). Price: Political Writings. Cambridge University Press. p. xxiii. ISBN 978-0-521-40969-8. Retrieved 16 June 2013.

[3] Stigler, Stephen M. (1986). The History of Statistics: The Measurement of Uncertainty before 1900. Harvard University Press, Chapter 3.

[4] Book Pattern Classification (2nd Edition) Wiley-Interscience New York, NY, USA 2000 ISBN:0471056693

[5] Book Pattern Recognition and Machine Learning (Information Science and Statistics) Springer-Verlag Berlin, Heidelberg 2006 ISBN:0387310738

APPENDIX

We compared our code and initial report draft with fellow students Soumik Dey (Subhadarshi's partner), Enis Coban (Shweta's partner) and Daniel Ladner. We received a lot of positive feedback and used those to improve our experimental results as well as report.

We initially were reading the training data every time we learned the decision rule. This slowed down our experiments, especially the quantization optimization part where we are training multiple times. So we loaded the data into a data frame to store it in memory. Now we could access the data without reading the file again and again.

In quantization optimization, we were initially just doing both the random perturbation part and the sequential check part (see subsection *F: Quantization Optimization* of Section III) by finding a δ and then picking specific number of steps determined by δ in the perturbation interval, which we chose to be between halfway from the previous and next boundaries. By doing so, we were doing a lot of unnecessary iterations, which increased the time of the experiment and apparently also did not provide improvement in the expected gain. After getting feedback from our fellow partners, we updated our code for random perturbations part. Instead of finding a δ and calculating expected gain for all perturbation for a particular boundary, we just picked a random number in the possible range and considered this as the perturbation. Here possible range means the range between the midpoints of the previous and next boundaries with current boundary.

In k smoothing, Enis followed a method to find out smoothed estimates as specified in page #27 in [4]. In this method the volume of all the hypercubes are assumed to be same, i.e., for any bin m, $v_m = v$. We found that the results of k smoothing are nearly same for our technique and Enis's technique.

We also updated our report, especially section III: *Methodology* and section IV: *Experiments*. We also added relevant references to our report.

[4] http://haralick.org/ML/quantization.pdf