Optimal Multidimensional Quantization for Pattern Recognition

Mingzhou Song[a] and Robert M. Haralick[b]

[a]Department of Computer Science, Queens College of CUNY, Flushing, NY 11367
[b]Computer Science Doctoral Program, Graduate Center of CUNY, New York, NY 10016

ABSTRACT

In non-parametric pattern recognition, the probability density function is approximated by means of many parameters, each one for a density value in a small hyper-rectangular volume of the space. The hyper-rectangles are determined by appropriately quantizing the range of each variable. Optimal quantization determines a compact and efficient representation of the probability density of data by optimizing a global quantizer performance measure. The measure used here is a weighted combination of average log likelihood, entropy and correct classification probability. In multi-dimensions, we study a grid based quantization technique. Smoothing is an important aspect of optimal quantization because it affects the generalization ability of the quantized density estimates. We use a fast generalized *k* nearest neighbor smoothing algorithm. We illustrate the effectiveness of optimal quantization on a set of not very well separated Gaussian mixture models, as compared to the expectation maximization (EM) algorithm. Optimal quantization produces better results than the EM algorithm. The reason is that the convergence of the EM algorithm to the true parameters for not well separated mixture models can be extremely slow.

Keywords: density estimation, grid quantization, smoothing

1. INTRODUCTION

We discuss optimal quantization, a non-parametric technique, for pattern recognition. Non-parametric methods do not have the large modeling biases inherent in parametric models. Optimal quantization tends to alleviate the on-line computation and storage burden as presented in standard non-parametric techniques, by finding the most effective non-parametric representation of the data, for given computation resources, in terms of both CPU cycles, memory requirement and the targeted performance. It requires intensive off-line training. Ideally, the representation should be compact. Some choices of representation yield very fast on-line algorithms whose time and space complexities are not directly related to the sample size. Optimal quantization possesses the scalability to trade more resources for performance. Most standard non-parametric models do not scale up or down and in many situations become prohibitive to apply.

A better approach to discretization of the space is to find a partition scheme to optimize a quantizer performance measure. We define such a measure that considers average log likelihood, correct classification probability and entropy. We study a genetic algorithm to find a grid globally and then refine the grid using a fast local algorithm. The grid is further used to obtain density estimates using a fast smoothing algorithm, as compared to the *k* nearest neighbor algorithm.

We review some related work in Section 2. In Section 3, we give the definition of the quantizer performance measure. In Section 4, we present global and local algorithms to find an optimal quantizer when the quantization pattern is a multidimensional grid. In Section 5, we extend the kernel smoothing idea and introduce a fast method to smooth the quantized density function. We compare the performance of the optimal quantization and the expectation maximization (EM) algorithm for multi-class Gaussian mixture data in Section 6 and finally we conclude our study in Section 7.

Further author information: M.S. E-mail: msong@cs.qc.edu. R.M.H. E-mail: haralick@gc.cuny.edu

2. RELATED WORK

Vector quantization, perhaps the most widely used quantization technique, assigns a vector to each cell. The goal of vector quantization is often to approximate data accurately in the squared error sense.[1, 2]

Multivariate histogram is an example of quantization for density estimation. Lugosi and Nobel[3, 4] have established the consistency results of multivariate histogram density estimate and proposed general partition schemes based on their results. Entropy and likelihood[5, 6, 7] have been proposed as quantizer measures. Splines or local polynomials can be used as representation of the density of a cell, but their computational efficiency is a problem in a multidimensional space.

For discretization, Simonoff[8] describes a way of finding a variable width bin histogram by inversely transforming an equal width bin histogram obtained from the transformed data. However, it is not clear how the transform function should be chosen in general.

Quite many activities[9, 10, 11, 12, 13] concentrate on discretization of multi-class 1-D data, in part because 1-D discretization algorithms serve as the building blocks for decision tree construction. Only the type of algorithms originally given by Fulton et al.[10] is optimal with respect to additive quantizer measures, where dynamic programming is used. Elomaa and Rousu[14] improve the practical efficiency of the algorithm on well-behaved additive quantizer measures. They show that many frequently used quantizer measures for classification purpose are well-behaved.

For multidimensional discretization, the concept of statistically equivalent blocks[15] is an early tree structured partitioning scheme. The CART[16] algorithm also uses the tree-structured classifier. In a multidimensional space, grid based partition schemes are studied, e.g., multivariate ASH,[17] STING[18] algorithm, OptiGrid[19] algorithm, STUCCO[20] algorithm, adaptive grids[21] and maximum marginal entropy quantization.[22] In Hearne and Wegman,[6, 7] the grid is acquired randomly. In Scott,[17] the grid lines are equally spaced. In Bay,[20] the grid is adjusted by merging adjacent intervals by hypothesis testing. The adaptive grid[21] uses a strategy to merge dense cells.

WARPing[23] and averaging shifted histograms (ASH)[24] are techniques that smooth density estimates of the cells. Otherwise there are relatively few histogram smoothing methods. Ideally, we would like to have all cells with a non-zero density estimate, but there is no available method for such including ASH. Besides the non-zero requirement, smoothing is important in maintaining consistency of density estimates and deserves further study.

3. QUANTIZER PERFORMANCE MEASURE

Let the random vector $X \in \mathbb{R}^D$ represent an individual data or pattern. $X(d)$ is the d-th dimension random variable of X. We call the sequence $\mathcal{X}_N = \{x_1, x_2, \cdots, x_N\}$ a data set or a sample of size N. This set contains i.i.d. data vectors from x_1 to x_N. Let K be the total number of classes and $\{1, 2, \cdots, K\}$ be the class label set. Let random variable $Y \in \{1, 2, \cdots, K\}$ be the class assignment of X. We call the sequence $\mathcal{Y}_N = \{y_1, y_2, \cdots, y_N\}$ the class assignment set of $\mathcal{X}_N$, where x_1 has class label y_1, x_2 has class label y_2 and so on. We also consider a more general case where the class assignment is not exclusive, but instead weighted. Let random vector $W \in [0,1]^K$ be the weighted class assignment vector. $W(y)$ is the weight for class y. We also require $\sum_{y=1}^{K} W(y) = 1$. We call the sequence $\mathcal{W}_N = \{w_1, w_2, \cdots, w_N\}$ the weighted class assignments of the data set $\mathcal{X}_N$,

DEFINITION 3.1 (QUANTIZER). *A quantizer Q is a function that maps $\mathbb{R}^D$ to $\mathcal{I}$. $\mathcal{I}$, called quantization index set, is a finite set of integers or integer vectors.*

DEFINITION 3.2 (QUANTIZATION CELL). *A quantization cell of Q is a set $\mathcal{C} \subset \mathbb{R}^D$ such that for any $x_1, x_2 \in \mathcal{C}$, $Q(x_1) = Q(x_2)$.*

For each cell, we assign an index $q \in \mathcal{I}$ to it. We use the function notation $q = Q(x)$ to denote that x is quantized to q or x belongs to cell q. We use $N(y)$ to denote the total number of class y data in the $\mathcal{X}_N$, that is

$$N(y) = \sum_{n=1}^{N} w_n(y)$$

Let $V(q)$ be the volume of cell q. Let N_q be the total number of data in cell q. Let $N_q(y)$ be the total number of data of class y in cell q, that is

$$N_q(y) = \sum_{n \in \{n | q = Q(x_n)\}} w_n(y)$$

We assume there are L cells. The index to the first cell is $q = 1$, and $q = L$ for the last cell.

Average log likelihood. Kullback-Leibler divergence from $\hat{p}(x)$ to $p(x)$ is

$$D(p||\hat{p}) = \int p(x) \log \frac{p(x)}{\hat{p}(x)} dx = \mathbf{E}[\log p(X)] - \mathbf{E}[\log \hat{p}(X)]$$

which, being non-negative (zero only when $\hat{p}(x) = p(x)$), should be minimized. As $p(x)$ is fixed, maximizing $\mathbf{E}[\log \hat{p}(X)]$ is equivalent to minimizing $D_{KL}(p||\hat{p})$. Let $p(q|y)$ be the density of cell q. Then $\mathbf{E}[\log \hat{p}(X|Y)]$ can be estimated by $\frac{1}{N(y)} \log \prod_{q=1}^{L} (p(q|y))^{N_q(y)}$. The *overall average log likelihood* of a quantizer Q is

$$J(Q) = \frac{1}{N} \sum_{y=1}^{K} N(y) \mathbf{E}[\log \hat{p}(X|y)] = \frac{1}{N} \sum_{y=1}^{K} \log \prod_{q=1}^{L} (p(q|y))^{N_q(y)}$$

When there is reason to believe that the class number ratio $N(1) : N(2) : \cdots : N(K)$ is representative for the true data, the overall average log likelihood is preferred, with the log likelihood of popular classes being emphasized.

The *mean class average log likelihood* is

$$J(Q) = \frac{1}{K} \sum_{y=1}^{K} \mathbf{E}[\log \hat{p}(X|y)] = \frac{1}{K} \sum_{y=1}^{K} \frac{1}{N(y)} \log \prod_{q=1}^{L} (p(q|y))^{N_q(y)}$$

When the class number count $N(y)$ is randomly decided or every class is considered to have equal importance, the mean class average log likelihood is preferred, with every class contributing equally to the log likelihood of the quantizer.

Correct classification probability. Let $P(y)$ be the prior probability of class Y. Within cell q, the Bayes' rule is equivalent to $y_q^* = \underset{y}{\operatorname{argmax}} P(y) N_q(y) / N(y)$. Let $N_c(q)$ be the number of correct decisions in cell q, i.e.,$N_c(q) = N_q(y_q^*)$. We give the definition of the correct classification probability in two situations. The *overall correct classification probability* is

$$P_c(Q) = \frac{\sum_{q=1}^{L} N_c(q)}{N} \tag{1}$$

The *mean class correct classification probability* is

$$P_c(Q) = \frac{1}{K} \sum_{y=1}^{K} \sum_{q=1}^{L} I(y = y_q^*) \frac{N_c(q)}{N(y)} \tag{2}$$

In the above two equations, I is indicator function. The choice of either should follow the considerations explained for the choice of average log likelihood.

Entropy. Similar to the case of average log likelihood, we give two options: overall entropy and mean class entropy. Again, the choice of either should follow the considerations explained for the choice of average log likelihood and correct classification probability. We define the *overall entropy* by

$$H(Q) = \frac{N_q}{N} \log \frac{N}{N_q} \tag{3}$$

We define *mean class entropy* by

$$H(Q) = \frac{1}{K} \sum_{y=1}^{K} \sum_{q=1}^{L} \frac{N_q(y)}{N(y)} \log \frac{N(y)}{N_q(y)} \tag{4}$$

Entropy has been used as a class impurity measure. But we use entropy as a measure of the consistence or generalization ability of the training results.

The performance measure function. We define the quantizer performance measure function, by linearly combining average log likelihood, entropy and the log of correct classification probability, as follows

$$T(Q) = W_J J(Q) + W_H H(Q) + W_c \log P_c(Q) \tag{5}$$

where W_J, W_H and W_c are given weights for average log likelihood $J(Q)$, entropy $H(Q)$ and log of correct classification probability $\log P_c(Q)$, respectively.

4. OPTIMAL GRID QUANTIZATION

An equal spacing grid is not efficient statistically. Variable spacing grid lines can dramatically improve the statistical efficiency while having low computational complexity. We consider the grid pattern quantizers with hyper-rectangular cells, shown in Fig. 1.

DEFINITION 4.1 (GRID). *Let G represent a grid in $\mathbb{R}^D$. There are $L_1, L_2, \cdots, L_D$ quantization levels, respectively, for dimension from 1 to D. Let L_d be the quantization levels in the d-th dimension. The decision boundaries of the d-th dimension are $G[d,0], G[d,1], \cdots, G[d,L_d]$. The values of the decision boundaries increase strictly monotonically, i.e., $G[d,0] < G[d,1] < \cdots < G[d,L_d]$.*

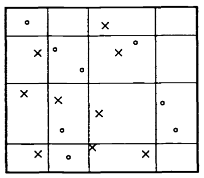


Figure 1. A grid pattern.

Issues involved in designing these quantizers include the orientation of the grid, relative quantization levels in each dimension and how to find an optimal grid for the given data. We introduce a global grid optimization algorithm using genetic algorithms and a fast local grid refinement algorithm.

4.1. Orientation

It has long been known that the performance of classifiers does not increase with dimensions monotonically, because the required sample size will grow exponentially with the increase in dimension. Dimension reduction may be necessary. Popular techniques including principle component analysis, projection pursuit and independent component analysis. In addition to dimension reduction, for data in a high dimension, we may also want to view the data in a new coordinate system such that the most interesting dimensions come first, where we would like to use more quantization levels.

4.2. Relative Quantization Levels in Each Dimension

For a given number L of quantization cells, the quantization levels $L_1, \cdots, L_D$ for each dimension are to be assigned. We use marginal entropy as a guide to allocate resources, with the bit-allocation rule

$$\frac{H_1(X)}{\log_2 L_1} = \frac{H_2(X)}{\log_2 L_2} = \cdots = \frac{H_D(X)}{\log_2 L_D} \tag{6}$$

$$\log_2 L_1 + \log_2 L_2 + \cdots + \log_2 L_D = \log_2 L \tag{7}$$

where $H_d(X)$ is the marginal histogram entropy for the d-th dimension of X. Solving above equations for L_d $(d = 1, 2, \cdots, D)$, we get

$$L_d = L^{\frac{H_d(X)}{\sum_{m=1}^{D} H_m(X)}} \quad (d = 1, 2, \cdots, D) \tag{8}$$

4.3. Global Optimization of a Grid

We cast the grid optimization problem into the genetic algorithm model in the following way. An individual is a grid. An individual is assumed to have only one chromosome. Therefore a chromosome also represents a grid. A gene is the sequence of the decision boundaries in a particular dimension of the grid. A nucleotide is a single decision boundary in the grid. For a complete and general description of genetic algorithms, refer to.[25]

How well an individual adapts to the environment is measured by the fitness function. The choice of a fitness function depends on how selection is carried out. When we use fitness proportionate selection, the chance of an individual being selected is in proportion to its fitness. Therefore, we would normally require the fitness function be non-negative in order to directly relate it with probabilities.

DEFINITION 4.2 (FITNESS FUNCTION). *The fitness function of a grid is*

$$\varphi(G) = \exp(T(G)) = [\exp(J(G))]^{W_J} [\exp(H(G))]^{W_H} [P_c(G)]^{W_c} \tag{9}$$

$\varphi(G)$ is non-negative since it is an exponential function of $T(G)$. $\exp(J(G))$ corresponds to the geometric average of likelihood which is a probability. $\exp(H(G))$ is the information content of the grid expressed in terms of the range of information coding. $P_c(G)$ is exactly the correct classification probability. $T(G)$ is a weighted geometric combination of the above components.

Alg. 1 Optimize-Grid-Genetic outlines the main steps of the grid optimization genetic algorithm. It starts with an initial population of N_p random grids. The population evolves itself by going through the selection-reproduction-selection cycle as described in the **for** loop from line 2 to 16. In every cycle, or generation, N_p children are reproduced in the **for** loop from line 7 to 15. Every execution of the loop produces two children C_1 and C_2 by parents G_1 and G_2. G_1 and G_2 are randomly selected (line 8 and 9) from the population and the chance of their being selected is in proportion to their fitness function values. A cross-over site d_r is randomly decided for the parent chromosomes or grids G_1 and G_2. The cross-over happens with a probability of P_r. Once the cross-over is finished, two children C_1 and C_2 are produced (line 11). For each of the two children grids, some of their decision boundaries are randomly changed by Mutate (line 12 to 13). Then these two children are added to next generation of population (line 14). The best ever grid is being kept as G^*. G^* is returned after a certain number of generations have been evolved.

Algorithm 1 Optimize-Grid-Genetic($\mathcal{X}_N$, $\mathcal{Y}_N$, N_p, N_g, P_r, P_u)

```
1: $\mathcal{P}^0 \leftarrow$ a population of $N_P$ random grids;
2: for $j \leftarrow 0$ to $N_g - 1$ do
3:   if $\varphi(G^*) < \max_{G \in \mathcal{P}^j} \varphi(G)$ then
4:     $G^* \leftarrow \operatorname{argmax}_{G \in \mathcal{P}^j} \varphi(G)$;
5:   end if
6:   $\mathcal{P}^{j+1} \leftarrow \phi$;
7:   for $i \leftarrow 0$ to $N_p - 1$ with increment of 2 do
8:     Randomly select a grid $G_1$ from $\mathcal{P}^j$ with a probability proportion to fitness value;
9:     Randomly select a grid $G_2$ from $\mathcal{P}^j$ with a probability proportion to fitness value;
10:    Randomly decide a dimension $d_r$ as the cross-over site;
11:    Exchange the decision boundaries of dimensions 1 to $d_r$ between $C_1$ and $C_2$ with probability $P_r$ or do not ex-
       change;
12:    Mutation: randomly adjust each decision boundary of $C_1$ with probability $P_u$;
13:    Mutation: randomly adjust each decision boundary of $C_2$ with probability $P_u$;
14:    $\mathcal{P}^{j+1} \leftarrow \mathcal{P}^{j+1} \cup \{C_1, C_2\}$;
15:  end for
16: end for
17: if $\varphi(G^*) < \max_{G \in \mathcal{P}^{N_g}} \varphi(G)$ then
18:   $G^* \leftarrow \operatorname{argmax}_{G \in \mathcal{P}^{N_g}} \varphi(G)$;
19: end if
20: return $G^*$;
```

4.4. A Fast Grid Refinement Algorithm

When the current best solution is close to the global optimal, genetic algorithm will be less efficient. Here we propose a more efficient grid refinement algorithm by adjusting the decision boundaries one by one. An adjusted boundary is accepted only when the performance measure increases.

By definitions of $J(G)$,$H(G)$ and $P_c(G)$, they are additive, i.e.,

$$J(G) = \sum_{q=1}^{L} J(q), \quad H(G) = \sum_{q=1}^{L} H(q), \quad P_c(G) = \sum_{q=1}^{L} P_c(q)$$

Therefore, when one decision boundary of a grid is moved, we need only recalculate the change of the additive measures on those affected cells and data.

Alg. 2 describes the Refine-Grid algorithm . The input includes a grid G, data sets $\mathcal{X}_N$ and $\mathcal{Y}_N$, number of refinements N_R and the convergence parameter δ. J, H, P_c and T record the performance measures of current grid. The refinement is done dimension by dimension as shown by the **for** loop of line 6. For each dimension, the boundaries of that dimension are refined one by one in the **for** loop starting from line 7.

A sub-grid G_s is formed by all the cells that is affected by each current decision boundary. The data that fall in the sub-grid are kept in sets $\mathcal{X}_{N'}$ and $\mathcal{Y}_{N'}$. Other data will not affect the refinement of the current boundary. The additive measures J^s, H^s and P_c^s are calculated for the sub-grid G_s. Line 12 actually finds the optimal decision boundary that maximizes the overall measure $T(G_s)$ and assigns the optimal boundary to the optimal sub-grid G_s^*. The changes in the additive measures of the sub-grid are calculated and the additive measures of the grid G are updated by the changes (line 13 to 16). The optimal decision boundary replaces the original one in G (line 17). The overall measure T is also updated. The refinement repeats until convergence is achieved as measured by $\Delta T/T$.

Algorithm 2 Refine-Grid(G, $\mathcal{X}_N$, $\mathcal{Y}_N$, N_R, δ)

```
 1: $J \leftarrow J(G), H \leftarrow H(G), P_c \leftarrow P_c(G)$;
 2: $T \leftarrow W_J J + W_H H + W_c \log P_c$;
 3: $j \leftarrow 0$;
 4: **repeat**
 5:   $T^- \leftarrow T$;
 6:   **for** $d \leftarrow 1$ to $D$ **do**
 7:     **for** $q \leftarrow 1$ to $L_d - 1$ **do**
 8:       Form a sub-grid $G_s$ by the cells sharing the decision boundaries $G[d,q]$;
 9:       $G_s^* \leftarrow G_s$;
10:       $\mathcal{X}_{N'}, \mathcal{Y}_{N'} \leftarrow \{(x_n, y_n) | x_n(d) \in (G_s[d,0], G_s[d,2]]\}$;
11:       $J^s \leftarrow J(G_s), H^s \leftarrow H(G_s), P_c^s \leftarrow P_c(G_s)$, all on $\mathcal{X}_{N'}, \mathcal{Y}_{N'}$;
12:       $G_s^*[d,1] \leftarrow \underset{G_s[d,1]}{\operatorname{argmax}}\ T(G_s)$ on $\mathcal{X}_{N'}, \mathcal{Y}_{N'}$;
13:       $\Delta J \leftarrow J(G_s^*) - J^s, \Delta H \leftarrow H(G_s^*) - H^s, \Delta P_c \leftarrow P_c(G_s^*) - P_c^s$, all on $\mathcal{X}_{N'}, \mathcal{Y}_{N'}$;
14:       $J \leftarrow J + \Delta J$;
15:       $H \leftarrow H + \Delta H$;
16:       $P_c \leftarrow P_c + \Delta P_c$;
17:       $G[d,q] \leftarrow G_s^*[d,1]$;
18:       $T \leftarrow W_J J + W_H H + W_c \log P_c$;
19:     **end for**
20:   **end for**
21:   $j \leftarrow j + 1$;
22:   $\Delta T \leftarrow T - T^-$;
23: **until** $j = N_R$ or $|\Delta T/T| < \delta$ ;
```

Now we can analyze the running time of the Refine-Grid algorithm. We assume a constant time is spent for Alg. 2 line 12. Since the sub-grid G_s has $2L/L_d$ cells, calculating the measure on the sub-grid takes about $N' \log_2(L/L_d) \leq N \log_2(L/L_d)$ time. Therefore the worst case running time is

$$O\left(N_R \sum_{d=1}^{D} L_d N \log_2(2L/L_d)\right) = O\left(N_R N \log_2 L \sum_{d=1}^{D} L_d\right)$$

When the data points are distributed pretty evenly in each cell, we can expect N' to be about $2N/L_d$. Then the expected running time is:

$$O\left(N_R \sum_{d=1}^{D} L_d (2N/L_d) \log_2(2L/L_d)\right) = O(N_R N \log_2 L)$$

The expected running time is linear in N and logarithm in L. In the time equivalent to the evaluation of the grid over the entire data set N_R times, all the decision boundaries will have been adjusted N_R times, modulo the constant time used in stepping for the best decision boundary.

4.5. The Refine-Genetic-Grid Algorithm

Alg. 3 Refine-Genetic-Grid combines the global Optimize-Grid-Genetic algorithm and the local algorithm Refine-Grid. The algorithm first calls the global algorithm and then refines the grid by the local algorithm.

```
Algorithm 3 Refine-Genetic-Grid($N_p$, $N_g$, $P_r$, $P_u$, $N_R$, $\delta$)
1: $G \leftarrow$ Optimize-Grid-Genetic($\mathcal{X}_N$, $\mathcal{Y}_N$, $N_p$, $N_g$, $P_r$, $P_u$);
2: Refine-Grid($G$, $\mathcal{X}_N$, $\mathcal{Y}_N$, $N_R$, $\delta$);
3: return $G$;
```

5. DENSITY IN A CELL AND SMOOTHING

The consistency of a quantizer is determined by the bias and variance of the density estimates. To reduce the bias, the sample size N must be large, and N_q/N must be small. To diminish the variance, the sample size N_q within a cell must be large.

Over-memorization is a serious potential problem when the emptiness issue is not properly handled. The symptom of over-memorization is that the quantizer works extraordinarily well on the training sample, but fails miserably on any unseen samples. Over-memorization manifests the large variance in cell density estimates, as a result of cell emptiness. Smoothing is an effort to reduce the variance of the density estimates. Over-smoothing is the other end of the spectrum. The symptom is that the quantizer gives very consistent result on both the training sample or an unseen sample, but carries unfortunately large biases.

We offer an algorithm to assign a density estimate to a quantization cell, with smoothing integrated. The algorithm is related to the k nearest neighbor density estimation method.

Let $V_k(q)$ be the volume of a minimum neighborhood that contains at least k points. The definition of the minimum neighborhood depends on the distance metric used and also the shape of the neighborhood. We do not require the shape of a neighborhood be a ball in the metric space chosen. Let k_q be the actual number of points in the neighborhood. Then a smoothed probability density estimate of cell q is

$$p(q) = \frac{\rho(q)}{\sum_r \rho(r)V(r)} = \frac{k_q}{V_k(q)\sum_r \frac{k_r}{V_k(r)}V(r)} \tag{10}$$

How to construct the k nearest neighborhood is an issue. Searching for the exact k-th nearest neighbor by going through the data is not very pleasing because of the computation involved. Here we first lay the foundation of an approximate, but no less than k, nearest neighbor smoothing algorithm by some basic definitions.

DEFINITION 5.1 (NEIGHBOR CELLS). *We call cell a and b neighbor cells if they share at least a partial decision boundary.*

DEFINITION 5.2 (RADIUS 0 NEIGHBORHOOD OF A CELL). *The radius* 0 *neighborhood of cell q is a set that contains exactly the cell itself. We use the notation* $\mathcal{N}(q,0)$ *to denote the radius* 0 *neighborhood of cell q.*

DEFINITION 5.3 (RADIUS R NEIGHBORHOOD OF A CELL). *The radius R* ($R \in \mathbb{Z}^+$) *neighborhood of cell q,* $\mathcal{N}(q,R)$*, is the union of the radius* $R-1$ *neighborhood* $\mathcal{N}(q,R-1)$*, and the set of all the neighbor cells of the cells in* $\mathcal{N}(q,R-1)$.

Fig. 2 shows the radius neighborhood of a cell with different radii. In Fig. 2(a), the cell of interest is the cell in gray. The cell itself is also its radius 0 neighborhood. Fig 2(b) and (c) draw its radius 1 and 2 neighborhoods.

DEFINITION 5.4 (k NEIGHBORHOOD OF A CELL). *k neighborhood of a cell is the smallest radius R neighborhood of the cell that contains at least k points.*

Alg. 4 Radius-Smoothing is based on the k neighborhood concept. The algorithm searches for a minimum radius R neighborhood of current cell with at least k data points. Then the density of the k neighborhood is assigned to the cell as its density estimate. M is the total mass on the density support. M can be considered an adjusted data count by smoothing and is related to N. For the cells containing less than k data points, the initial guess of R is the radius of the k

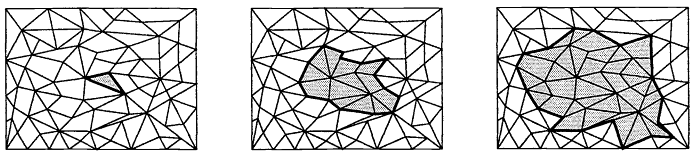

(a) $R = 0$ (b) $R = 1$ (c) $R = 2$

Figure 2. Radius R neighborhood of a cell.

neighborhood of the previous cell. To make this initial guess more realistic, we shall go through the cells in an order that every pair of adjacent cells are neighbor cells. k_q is the actual total number of data points in current radius R neighborhood. Based on the data count in current radius R neighborhood, we either increase R until there are at least k data points in the neighborhood, or decrease R until the $R-1$ neighborhood contains less than k data points.

Algorithm 4 Radius-Smoothing(Q, k)

```
1: $M \leftarrow 0, R \leftarrow -1$;
2: for each cell $q$ do
3:   if $N(q) = k$ then
4:     $R \leftarrow 0, k_q \leftarrow N(q)$;
5:   else
6:     if $R < 0$ then
7:       $R \leftarrow 0, k_q \leftarrow N(q)$;
8:     else
9:       $\mathcal{A} \leftarrow \mathcal{N}(q^-,R) \cap \mathcal{N}(q,R)$;
10:      $k_q \leftarrow k_q - |\mathcal{N}(q^-,R) - \mathcal{A}| + |\mathcal{N}(q,R) - \mathcal{A}|$;
11:    end if
12:    while $k_q > k$ and $R > 0$ do
13:      $k_q \leftarrow k_q - |\mathcal{N}(q,R) - \mathcal{N}(q,R-1)|, R \leftarrow R-1$;
14:    end while
15:    while $k_q < k$ and $R < R_{\max}$ do
16:      $k_q \leftarrow k_q + |\mathcal{N}(q,R+1) - \mathcal{N}(q,R)|, R \leftarrow R+1$;
17:    end while
18:  end if
19:  $\rho(q) \leftarrow \frac{k_q}{V(\mathcal{N}(q,R))}, M \leftarrow M + \rho(q)V(q), q^- \leftarrow q$
20: end for
21: for each cell $q$ do
22:   $p(q) \leftarrow \frac{\rho(q)}{M}$;
23: end for
```

The k neighborhood search of one cell goes through at most all L cells. Thus the total running time of the Radius-Smoothing has an upper bound of $O(L^2)$. However, because we (1) use the radius of the previous k neighborhood as an initial guess instead of starting from $R = 0$ and (2) find the data count in the current neighborhood by adjusting data count in the previous neighborhood, we can expect a constant time in finding out the k neighborhood of current cell. Hence, we would have total expected running time of L for doing all the cells.

The extent of smoothing is usually controlled by a parameter on the size of the local neighborhood. This naturally brings up the question of how to measure the quantizer consistency as a function of the control parameter. For the non-parametric approach, the only information available is the training sample, on which we will base any decision. We use cross-validation to determine an optimal control parameter k^* for smoothing, such that it maximizes the average quantizer performance.

6. EXAMPLE 2-D THREE CLASS GAUSSIAN MIXTURES

We compare the performance of optimal quantization and EM algorithm for multi-class multivariate Gaussian mixtures. We compare the performance by the average log likelihood and the correct classification probability on test data. We use both exclusive class assignment and using weighted class assignment. The performance of the optimal quantization is also studied as a function of the sample size N and quantization level L.

6.1. Data

There are three classes of data, each with a different Gaussian mixtures distribution. Let $f_N(x|\mu,\Sigma)$ be the probability density function of the Gaussian distribution with mean vector μ and covariance matrix Σ. The distributions for the three classes are

- Class 1 is a Gaussian mixture with two components.

$$p(x|Y=1) = 0.5 f_N\left(x \middle| \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 9 & 0 \\ 0 & 3 \end{bmatrix}\right) + 0.5 f_N\left(x \middle| \begin{bmatrix} 0 \\ 5 \end{bmatrix}, \begin{bmatrix} 5 & 0 \\ 0 & 7 \end{bmatrix}\right)$$

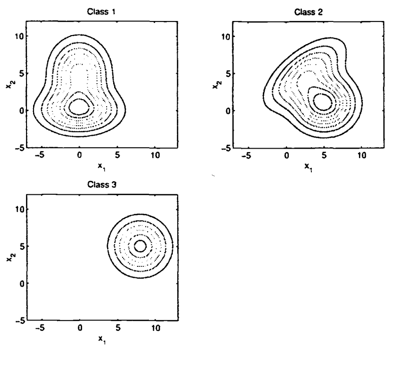
- Class 2 is also a Gaussian mixture with two components.

$$p(x|Y=2) = 0.6 f_N\left(x \middle| \begin{bmatrix} 5 \\ 1 \end{bmatrix}, \begin{bmatrix} 6 & 0 \\ 0 & 5 \end{bmatrix}\right) + 0.4 f_N\left(x \middle| \begin{bmatrix} 2.5 \\ 5 \end{bmatrix}, \begin{bmatrix} 9 & 5 \\ 5 & 7 \end{bmatrix}\right)$$

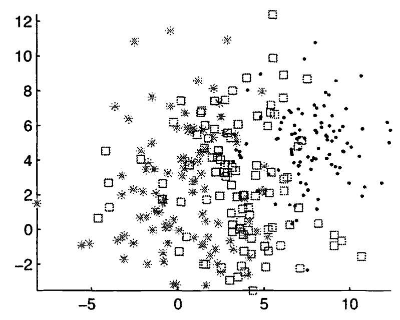
- Class 3 is Gaussian.

$$p(x|Y=3) = f_N\left(x \middle| \begin{bmatrix} 8 \\ 5 \end{bmatrix}, \begin{bmatrix} 5 & 0 \\ 0 & 5 \end{bmatrix}\right)$$

Fig 3(a) shows the contour plots of the true class densities. There are five training sets. The sample sizes are, respectively,:



(a) Contour plots of the true class probability density functions.



(b) Scatter plot of a sample of size $N = 300$, with stars, squares and dots representing class 1, 2 and 3, respectively.

Figure 3. True class densities and simulated data.

$10^2 \times 3$, $10^3 \times 3$, $10^4 \times 3$, $10^5 \times 3$, $10^6 \times 3$. $N \times K$ means there are N items of data for each of the K classes. Fig 3(b) shows the scatter plot of the $10^2 \times 3$ training set. We have two types of class assignment for training data:

1. Exclusive class assignment. In a training set, if a data vector x_n is simulated from the distribution of class y, then the class assignment of x_n is $y_n = y$.

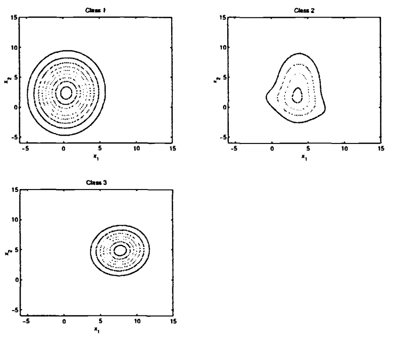
2. Weighted class assignment. In a training set, the weighted class assignment vector w_n of a data vector x_n is determined by

$$w_n(y) = \frac{p(x_n|y)}{\sum_{k=1}^{K} p(x_n|k)}, \quad y = 1 \cdots K$$

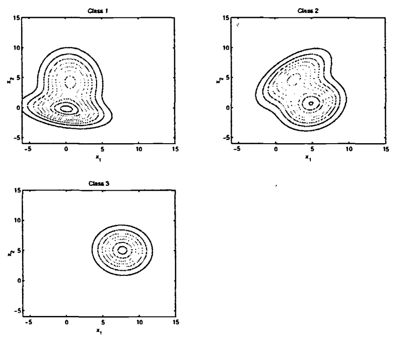
The test data has a sample size of $(2 \times 10^6) \times 3$, i.e. there are 2×10^6 data in each of the three classes. The overall average log likelihood is of the test data using the true class densities is $J_0 = -4.79889$. The overall correct classification probability using the true densities is $P_0 = 74.4407\%$.

6.2. Results of the EM Algorithm

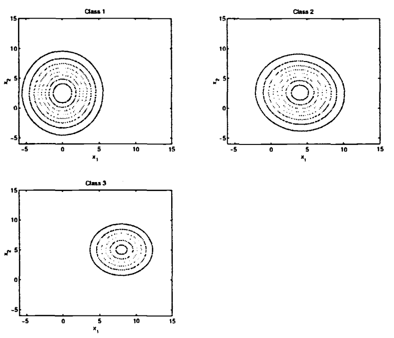
We use EM algorithm to estimate the Gaussian mixtures of each class. We set the maximum number of mixture components for each class to 2. The actual number of mixture components of each class is cross-validated. Fig. 4 presents the contour plots of the estimated densities using 300 and 3000000 sample sizes. Both the results of using exclusive class assignment and weighted class assignment are shown in each figure.



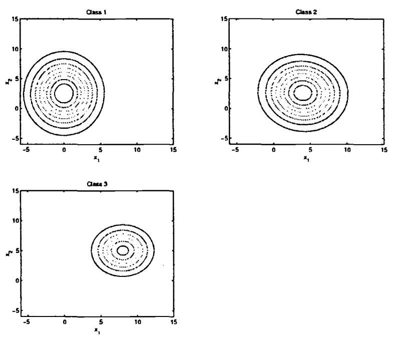
(a) $N = 100 \times 3$, training data have exclusive class assignment.



(b) $N = 300$, training data have weighted class assignment.



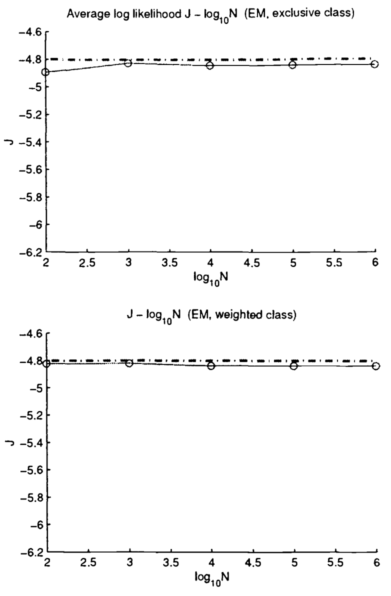
(c) $N = 1{,}000{,}000 \times 3$, training data have exclusive class assignment.



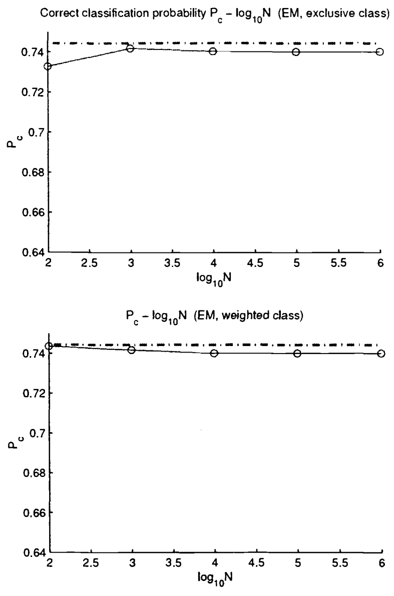
(d) $N = 3{,}000{,}000$, training data have weighted class assignment.

Figure 4. Contour plots of the density estimates using EM algorithm.

We can see that for class 3, its single Gaussian component can always be correctly estimated. For class 1 and 2, who both have two mixture components, the results vary. Only for the 300 data set with weighted class assignment, the EM algorithm is able to accurately capture the two mixture components for class 1. The same EM algorithm fails on all the other data sets in either correctly detecting the number of mixtures or correctly estimating the parameters. For class 2, EM



(a) Average log likelihood J, as a function of sample size N



(b) Correct classification probability P_c, as a function of sample size N

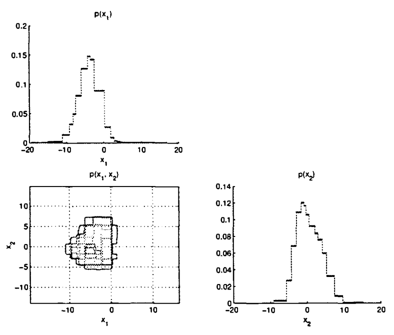
Figure 5. Performance of the EM algorithm on 6,000,000 test data. The top two plots were obtained from training data with exclusive class assignments. The bottom two plots were obtained from training data with weighted class assignment. The dash-dotted lines are the performance on the test data using the true class densities, for which the average log likelihood is $J_0 = -4.79889$, shown in the left two plots, and the correct classification probability is $P_0 = 74.4407\%$, shown in the right two plots.

algorithm is able to detect the correct number of mixture components and the parameters only on data sets $N = 100 \times 3$. For $N = 100 \times 3$ data set with exclusive class assignment, the estimated parameters are not very accurate. In some of the results, the number of components is estimated correctly, but still the component parameter estimates collapse to the same single Gaussian component. Only on the data set $N = 100 \times 3$ with weighted class assignment, EM algorithm is able to give very accurate density estimates of all the classes. As noted by Xu and Jordan,[26] the convergence is much faster when the components are well separated than when they are too close to each other. Our example is indeed a hard example for the EM algorithm. However, the makeup of the data is not totally artificial. It is based on some real-world data sets in which the classes are quite mixed together.

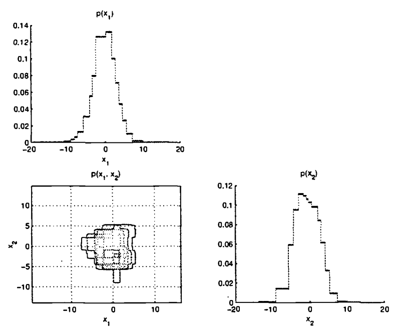
Fig. 5 shows the overall performance. It is expected that the overall performance does not increase with sample size for parametric models, as long as the sample size is sufficiently larger than the number of parameters. In the figure, the performance as a function of N indeed confirms this independence of sample size. As we have analyzed the individual density estimate results, it is not surprising that $N = 300$ with weighted class assignment gives the best performance on the test data, which is very close to the gold standards of both the average log likelihood and the correct classification probability.

6.3. Results of Optimal Quantization

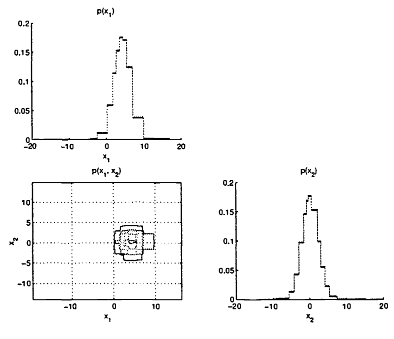
The data are rotated by their principle component vectors without scaling. No scaling makes the average log likelihood comparable with other density estimates or the true density. We try an increasing number of quantization cells L: 16, 256, 4096, 65536, and 1048576. The quantization levels of each dimension is assigned by the marginal entropy ratio and the given total quantization level L. The parameters used for the genetic algorithm are: size of population $N_p = 20$, number of generations $N_g = 20$, crossover probability $P_r = 0.8$, mutation probability $P_u = 0.5$. We use a 5-fold cross-validation for



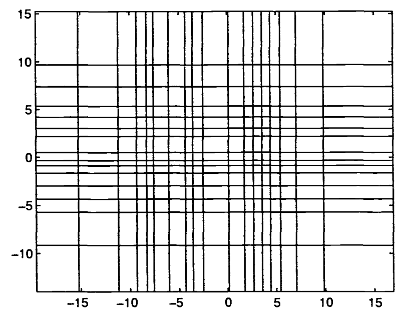
(a) Class 1.



(b) Class 2.



(c) Class 3.



(d) The grid.

Figure 6. The grid of quantization levels $L \approx 256$ and its density estimates, calculated from 3,000,000 training data with weighted class assignment.

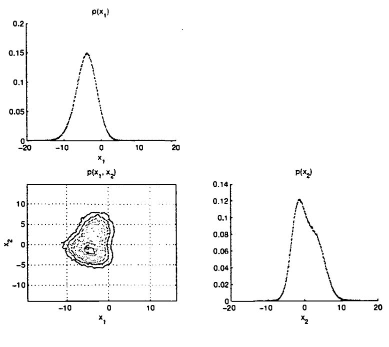
k of each class. k are selected for each class individually. When the data have weighted assignment, we threshold the data into different classes to get a cross-validated k. The quantizer performance measure is $T(G) = J(G) + H(G) + \log P_c(G)$.

The density estimate results obtained from the training set with $N = 1{,}000{,}000 \times 3$ data are shown. Fig. 6 and Fig. 7 present the grids, density estimates and their marginal densities using $L \approx 256$, and $L \approx 65536$, respectively. These two sets of optimal quantization density estimates clearly show dependency on the quantization level L. For quantization levels $L \approx 256$, we have already observed the trend towards the true underlying densities. As L increases, the density estimates approach the true density visually in all three classes.

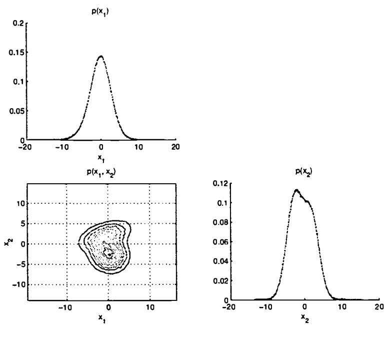
Fig. 9 shows the average log likelihood and the correct classification probability of the test data. The performance is shown as functions of quantization level L and sample size N. The performance plots in Fig. 9 show that the performance almost always improves with sample size N and quantization level L. Fig. 8 shows the optimal number of neighbors k for smoothing as a function of the sample size N and quantization level L.

Fixing L, we inspect the relation between optimal k and N. When N is small, optimal k increases with N. optimal k continues to increase as N becomes larger. However, optimal k stops increasing after $N > N_0(L)$. The reason is that each individual cell has enough data in it. The generalization reaches an appropriate level whose performance is acceptable and no over-memorization occurs. Hence smoothing is not necessary any more as N is sufficiently large, as compared to $N_0(L)$.

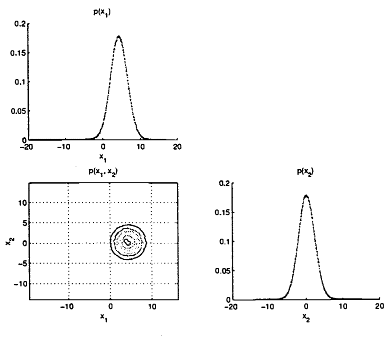
Fixing N, we inspect the relation between optimal k and L. When L is relative small to the sample size N, quantization effect is sever, resulting large error in density estimate. The performance is highly sensitive to where the decision boundaries are. Very large k is needed to maintain acceptable consistency, by smoothing out quantization effect. When L



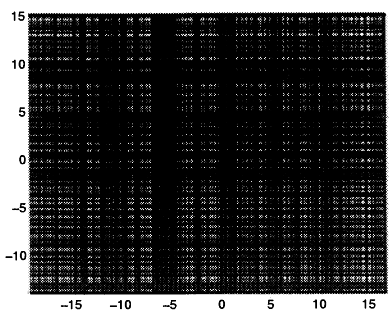
(a) Class 1.



(b) Class 2.



(c) Class 3.



(d) The grid. (If reading online, blow up to see the details.)

Figure 7. The grid of quantization levels $L \approx 65536$ and its density estimates, calculated from 3,000,000 training data with weighted class assignment.

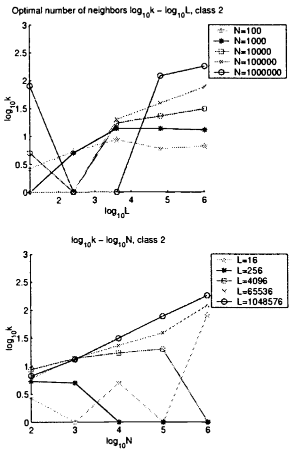
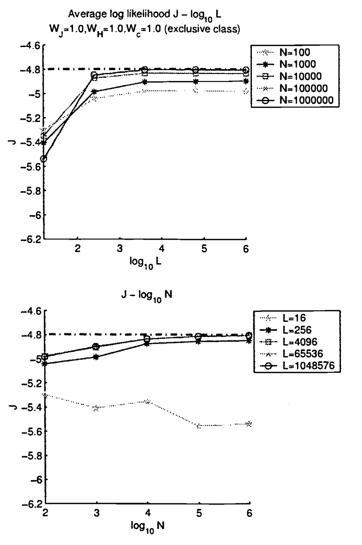
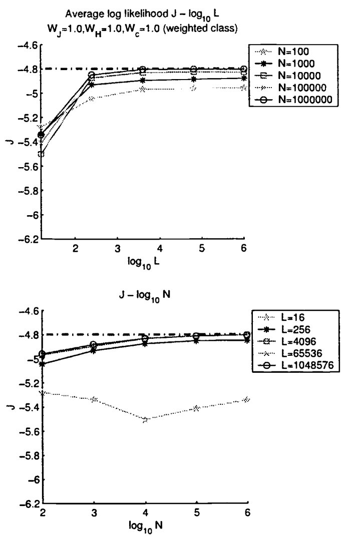


Figure 8. Optimal number of neighbors k, as a function of quantization level L (top plot of each sub-figure), and as a function of sample size N (bottom plot of each sub-figure).

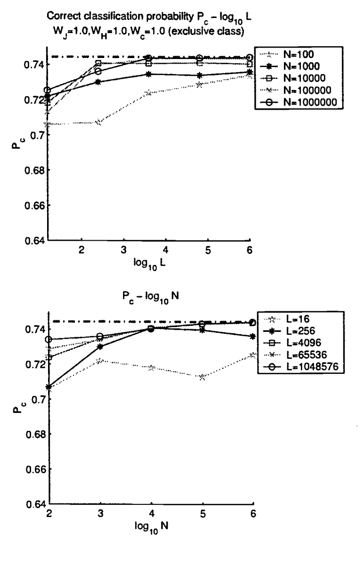
is moderate large, optimal k is relative large but is smaller than the optimal k for the canonical k nearest neighbor method. This is true because of the nature of the radius neighborhood smoothing algorithm. When L is very large, quantization



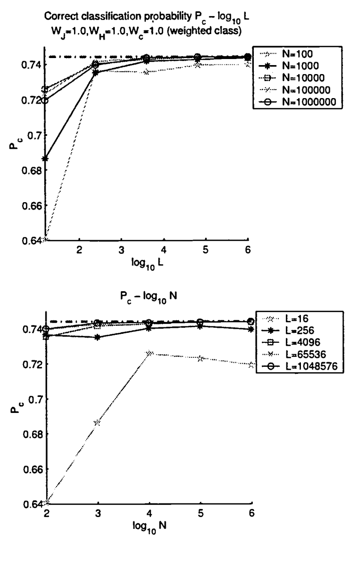
(a) J(exclusive class assignment.)



(b) J(weighted class assignment.)



(c) P_C(exclusive class assignment.)



(d) P_C(weighted class assignment.)

Figure 9. J and P_C on 6,000,000 test data, as a function of quantization level L (top two plots), and as a function of sample size N (bottom two plots). The dash-dotted line in each plot is the performance on the test data using the true class densities, for which the average log likelihood is $J_0 = -4.79889$ and the correct classification probability is $P_0 = 74.44079\%$.

effect is minor, optimal k does not change too much with L any more and approaches the optimal k for the canonical nearest neighbor method.

Using weighted class training data outperforms using exclusive class training data. The advantage is clearly shown when sample size is small. When sample size gets larger, the advantage dissolves. The phenomenon is true for both EM and optimal quantization.

6.4. Comparison between EM and Optimal Quantization

When $N \geq 10000 \times 3$ and $L \geq 4096$, the performance of optimal quantization is no less than the best attainable performance by the EM algorithm. Actually the performance of optimal quantization is very close to the gold standard. Even if all EM estimates had converged to the correct parameters, optimal quantization will not be too much worse if not better. This result is significant because EM has the correct model information, but optimal quantization does not assume anything about the form of the model. When a parametric model is complex, it is of great value if the parameters can be estimated correctly. However, if the parameters can not be found with good accuracy, a parametric model would not be as good as a non parametric model, even if the model is correct. In more general situations, the form of the parametric model may not be assumed and optimal quantization is apparently at advantage.

7. CONCLUSIONS

Optimal quantization substantially reduces the time and space demands as required by other non-parametric methods, while maintaining the benefit of non-parametric methods. The improvement in efficiency is a result of the adaptive behavior of optimal quantization. Resources are prioritized towards regions where it is necessary. In addition, better results can always be achieved with more quantization levels, which provides a natural way of balancing resources and performance. Even when compared to algorithms for known parametric model families, optimal quantization produces comparable or better results. In our experiment, the EM algorithm may diverge from the true parameters for not very well separated Gaussian mixture models, while optimal quantization can capture the underlying distributions very well without assuming anything in advance. In real-world applications, when parametric models can not be assumed or the convergence to the true model parameters is slow, optimal quantization is a definite solution to produce efficient and consistent representation of the data for pattern recognition tasks.

REFERENCES

1. R. M. Gray and D. L. Neuhoff, "Quantization," *IEEE Trans. Inform. Theory* **44**, pp. 2325–83, October 1998.
2. S. Graf and H. Luschgy, *Foundations of Quantization for Probability Distributions*, vol. 1730 of *Lecture notes in mathematics*, Springer, 2000.
3. G. Lugosi and A. B. Nobel, "Consistency of data-driven histogram methods for density estimation and classification.," *Annals of Statistics* **24**, pp. 687–706, 1996.
4. A. B. Nobel, "Histogram regression estimation using data-dependent partitions," *Annals of Statistics* **24**(3), pp. 1084–1105, 1996.
5. R. M. Haralick, "The table look-up rule," *Communications in Statistics – Theory and Methods* **A5**(12), pp. 1163–91, 1976.
6. L. B. Hearne and E. J. Wegman, "Maximum entropy density estimation using random tessellations," in *Computing Science and Statistics*, **24**, pp. 483–7, 1992.
7. L. B. Hearne and E. J. Wegman, "Fast multidimensional density estimation based on random-width bins," in *Computing Science and Statistics*, **26**, pp. 150–5, October 1994.
8. J. S. Simonoff, *Smoothing Methods in Statistics*, Springer series in statistics, Springer, New York, 1996.
9. U. M. Fayyad and K. B. Irani, "Multi-interval discretization of continuous-valued attributes for classification learning," in *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence*, pp. 1022–1029, 1993.
10. T. Fulton, S. Kasif, and S. L. Salzberg, "Efficient algorithms for finding multi-way splits for decision trees," in *Proc. 12th International Conference on Machine Learning*, pp. 244–251, Morgan Kaufmann, 1995.
11. R. Kohavi and M. Sahami, "Error-based and entropy-based discretization of continuous features," in *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, pp. 114–119, 1996.
12. T. Elomaa and J. Rousu, "General and efficient multisplitting of numerical attributes," *Machine Learning* **36**, pp. 201–44, 1999.
13. J. Rousu, *Efficient Range Partitioning in Classification Learning*. Ph.D. Dissertation, Department of Computer Science, University of Helsinki, Finland, January 2001.

14. T. Elomaa and J. Rousu, "Speeding up the search for optimal partitions," in *Principles of Data Mining and Knowledge Discovery, Proc. 3rd PKDD, Lecture Notes in Artificial Intelligence*, pp. 89–97, 1999.
15. M. P. Gessaman, "A consistent nonparametric multivariate density estimator based on statistically equivalent blocks," *The Annals of Mathematical Statistics* **41**, pp. 1344–46, 1970.
16. L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, *Classification and Regression Trees*, Statistics/Probability Series, Wadsworth & Brooks/Cole, Pacific Grove, California, 1984.
17. D. W. Scott and G. Whittaker, "Multivariate applications of the ASH in regression," *Communications in Statistics A: Theory and Methods* **25**, pp. 2521–30, 1996.
18. W. Wang, J. Yang, and R. R. Muntz, "STING: A statistical information grid approach to spatial data mining," in *Proceedings of the 23rd VLDB Conference*, pp. 186–195, 1997.
19. A. Hinneburg and D. A. Keim, "Optimal grid-clustering: Towards breaking the curse of dimensionality in high-dimensional clustering," in *VLDB'99, Proceedings of 25th International Conference on Very Large Data Bases, September 7-10, 1999, Edinburgh, Scotland, UK*, M. P. Atkinson, M. E. Orlowska, P. Valduriez, S. B. Zdonik, and M. L. Brodie, eds., pp. 506–517, Morgan Kaufmann, 1999.
20. S. D. Bay, "Multivariate discretization for set mining," *Knowledge and Information Systems* **3**(4), pp. 491–512, 2001.
21. H. Nagesh, S. Goil, and A. Choudhary, "Adaptive grids for clustering massive data sets," in *Proceedings of the First SIAM International Conference on Data Mining, April 5-7, 2001, Chicago, IL USA*, V. Kumar and R. Grossman, eds., pp. 506–517, Society for Industrial & Applied Mathematics, 2001.
22. T. Chau, "Marginal maximum entropy partitioning yields asymptotically consistent probability density functions," *IEEE Trans. PAMI* **23**, pp. 414–417, Apr. 2001.
23. W. Härdle and D. W. Scott, "Smoothing by weighted averaging of rounded points," *Computational Statistics* **7**, pp. 97–128, 1992.
24. D. W. Scott, *Multivariate Density Estimation – Theory, Practice and Visualization*, Wiley Series in Probability and Mathematical Statistics, John Wiley & Sons, 1992.
25. D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley, 1989.
26. L. Xu and M. I. Jordan, "On convergence properties of the EM algorithm for gaussian mixtures," *Neural Computation* **8**, pp. 129–151, 1996.