Vector Quantization: a Limiting Case of EM

1. Introduction & definitions

Assume that you are given a data set $\mathbf{X} = \{\mathbf{x}_j\}$, $j \in \{1, 2, \ldots, n\}$, of n d-dimensional vectors. The vector quantization (VQ) problem requires that we find a set of prototype vectors $\mathbf{Z} = \{\mathbf{z}_i\}$, $i \in \{1, 2, \ldots, L\}$, $L \ll n$, such that the total distortion D,

$$D = \sum_{j=1}^{n} \min_i \, dist(\mathbf{x}_j, \mathbf{z}_i) \tag{1}$$

is minimized. In equation (1), $dist(\mathbf{x}_j, \mathbf{z}_i)$ is a distance metric given by either,

$$dist(\mathbf{x}, \mathbf{z}) = \|\mathbf{x} - \mathbf{z}\| \text{ (Euclidean distance),} \tag{2}$$

or, more generally,

$$dist(\mathbf{x}, \mathbf{z}) = (\mathbf{x} - \mathbf{z})^T Q (\mathbf{x} - \mathbf{z}) \tag{3}$$

where Q is a positive-definite, symmetric matrix, $Q > 0$. Weighting the distance along each dimension through the Q matrix can normalize the distance measure in equation (3) with respect to different scaling along different dimensions of the $\{\mathbf{x}_j\}$ vectors.

The vectors $\mathbf{Z}$ are known as the VQ codebook. Two applications of v ector quantization are (1) redundant data compression, and (2) approximating continuous probability distributions with approximate discrete ones (i.e. histograms), where each $\mathbf{x}_j$ is replaced with the label i such that,

$$dist(\mathbf{x}_j, \mathbf{z}_i) \le dist(\mathbf{x}_j, \mathbf{z}_l), \ \forall l . \tag{4}$$

We will see later that this is especially useful in hidden Markov modeling.

2. The k-means algorithm

A. Algorithm definition

The k-means algorithm is an algorithm for generating $\mathbf{Z}$ the VQ codebook of prototype vectors. It is guaranteed to converge to a local minimum of D. The algorithm proceeds as follows:

1. **Initialization**: Choose some initial setting for the L codes $\{\mathbf{z}_i\}$ in the VQ codebook. One way to do this is to initialize the $\{\mathbf{z}_i\}$ to some random subset of L vectors in $\mathbf{X}$.

Loop

2. **Classification**: Classify each $\mathbf{x}_j$ into cluster or class ω_i such that,

$$dist(\mathbf{x}_j, \mathbf{z}_i) \le dist(\mathbf{x}_j, \mathbf{z}_l), \ \forall l . \tag{5}$$

3. **Codebook update**: Update the code for every cluster ω_i by computing its centroid,

$$\mathbf{z}_i = \frac{1}{n_i}\left(\sum_{\mathbf{x}_j \in \omega_i} \mathbf{x}_j\right) \tag{6}$$

where n_i is the number of vectors $\mathbf{x}_j$ in cluster ω_i.

4. **Termination**: Stop when the distortion D has decreased below some threshold level, or when the algorithm has converged to a constant level of distortion.

B. Example #1

Here, we investigate the convergence properties of the k-means algorithm with a simple example. Let $\mathbf{X}$ be a set of $n = 1000$ 2-dimensional vectors $\{\mathbf{x}_j\}$, $j \in \{1, 2, \ldots, 1000\}$, distributed uniformly in the unit square,

$$0 \le x_1, x_2 \le 1 \tag{7}$$

as shown in Figure 1 below.

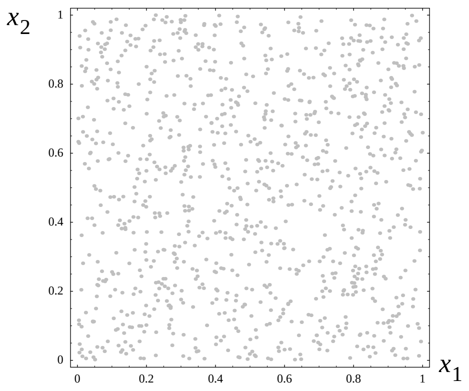


Figure 1

Assuming infinite data, the distortion for two codes $\{\mathbf{z}_1, \mathbf{z}_2\}$ is given by,

$$D(\mathbf{z}_1, \mathbf{z}_2) = \int_{A_1} dist(\mathbf{x}_j, \mathbf{z}_1)dA_1 + \int_{A_2} dist(\mathbf{x}_j, \mathbf{z}_2)dA_2 \tag{8}$$

where A_i denotes the area in the unit square that is part of cluster ω_i, $i \in \{1, 2\}$. Therefore, the globally optimal solution $\{\mathbf{z}_1{}^*, \mathbf{z}_2{}^*\}$ — in other words, the minimum-distortion solution — for two codes, $\{\mathbf{z}_1, \mathbf{z}_2\}$ is given by,

$$\{\mathbf{z}_1{}^*, \mathbf{z}_2{}^*\} = \underset{\{\mathbf{z}_1, \mathbf{z}_2\}}{\operatorname{argmin}} [D(\mathbf{z}_1, \mathbf{z}_2)] \tag{9}$$

Denoting,

$$\mathbf{z}_1 = \{z_{11}, z_{12}\} \text{ and } \mathbf{z}_2 = \{z_{21}, z_{22}\}, \tag{10}$$

it appears that $D(\mathbf{z}_1, \mathbf{z}_2)$ must be optimized over four independent scalars: $\{z_{11}, z_{12}, z_{21}, z_{22}\}$. Since the data is distributed symmetrically about $(1/2, 1/2)$, the optimal prototype vectors $\{\mathbf{z}_1{}^*, \mathbf{z}_2{}^*\}$ are, however, constrained by,

$$z_{21}{}^* = 1 - z_{11}{}^* \tag{11}$$

$$z_{22}{}^* = 1 - z_{12}{}^* \tag{12}$$

Therefore, we can explicitly plot $D(\mathbf{z}_1, \mathbf{1} - \mathbf{z}_1)$, where $\mathbf{1} = \{1, 1\}$, as a function of $\{z_{11}, z_{12}\}$, as shown in Figure 2 below. In Figure 2, red shades indicate the smallest distortions, and we see that there are four globally optimal solutions $\{\mathbf{z}_1{}^*, \mathbf{z}_2{}^*\}$:

$$\{\mathbf{z}_1{}^*, \mathbf{z}_2{}^*\} = \left\{\left(\frac{1}{2}, \frac{1}{4}\right), \left(\frac{1}{2}, \frac{3}{4}\right)\right\}, \{\mathbf{z}_2{}^*, \mathbf{z}_1{}^*\} = \left\{\left(\frac{1}{2}, \frac{1}{4}\right), \left(\frac{1}{2}, \frac{3}{4}\right)\right\} \tag{13}$$

$$\{\mathbf{z}_1{}^*, \mathbf{z}_2{}^*\} = \left\{\left(\frac{1}{4}, \frac{1}{2}\right), \left(\frac{3}{4}, \frac{1}{2}\right)\right\}, \{\mathbf{z}_2{}^*, \mathbf{z}_1{}^*\} = \left\{\left(\frac{1}{4}, \frac{1}{2}\right), \left(\frac{3}{4}, \frac{1}{2}\right)\right\} \tag{14}$$

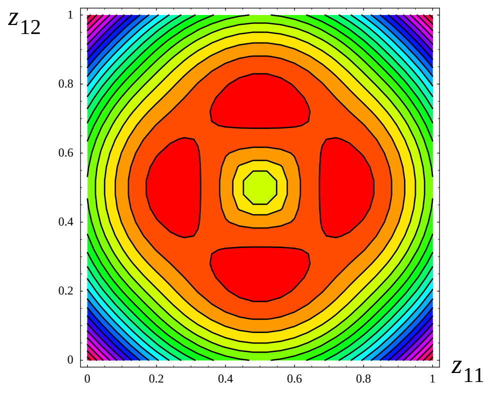


Figure 2

Note that the four solutions are the same, except for a switch in the two axes, as well as a switch in the prototype vector labels.

Now that we know what the theoretical minimum-distortion two-code solutions are, we conduct the following experiment. We run the k means algorithm for 100 initial random prototypes in the interval,

$$(0, 0) \le \mathbf{z}_i \le (1, 1), \ i \in \{1, 2\} \tag{15}$$

and observe the values of $\{\mathbf{z}_1, \mathbf{z}_2\}$ to which the algorithm converges. Figure 3 below plots the results of the 100 trials. Note that all 100 trials converge to approximately the optimal solutions in (13) and (14). The decision regions between class ω_1 and ω_2 for each solution pair $\{\mathbf{z}_1, \mathbf{z}_2\}$ is given by either,

$$x_1 \approx 0.5 \ \text{ or } \ x_2 \approx 0.5 \,. \tag{16}$$

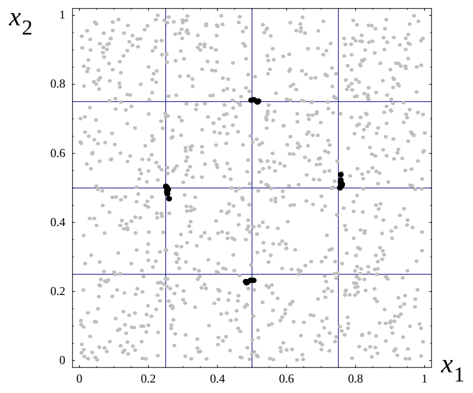


Figure 3

C. Example #2

Here, we investigate the convergence properties of the k-means algorithm for the uniform distribution $\mathbf{X}$ of 1000 points in the annular region shown in Figure 4 below, where the inside and outside radii are given by $r_1 = 0.12$ and $r_2 = 0.35$, respectively.

Since the distribution is radially symmetric about the point $(1/2, 1/2)$, the locus of globally optimal (minimum-distortion) 2-code solutions is necessarily described by the circle,

$$(x_1 - 1/2)^2 + (x_2 - 1/2)^2 = r^2 \tag{17}$$

Once again assuming infinite data, we can compute the globally optimal value for r by recognizing that the two classes ω_1 and ω_2 can be described by the solid and dashed lines as indicated in Figure 4. Note that the

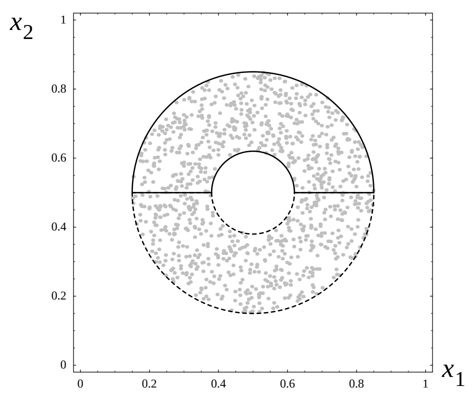


Figure 4

delineated regions are only one possible description of ω_1 and ω_2; the regions can, of course, be rotated by an angle θ, $0 \le \theta \le 2\pi$, without loss of optimality.

To compute r we now simply have to compute the centroid of the solid-line region — let's call this region A_1 — in the above figure so that,

$$r = \left(\int_{A_1} x_2 dA_1\right) \Big/ \left(\int_{A_1} dA_1\right) - \frac{1}{2} \tag{18}$$

$$r = \frac{1789}{3525\pi} \approx 0.1615 \tag{19}$$

Thus, the set of globally optimal solutions for the codes $\{\mathbf{z}_1, \mathbf{z}_2\}$ is given by,

$$\{\mathbf{z}_1{}^*, \mathbf{z}_2{}^*\} = \{(1/2 + r\cos\theta, 1/2 + r\sin\theta), (1/2 + r\cos[\theta + \pi], 1/2 + r\sin[\theta + \pi])\}, \forall\theta . \tag{20}$$

We now conduct the following experiment. We run the k-means algorithm for five initial random codes in the interval,

$$(0, 0) \le \mathbf{z}_i \le (1, 1), \ i \in \{1, 2\} \tag{21}$$

and observe the values of $\{\mathbf{z}_1, \mathbf{z}_2\}$ to which the algorithm converges. The figure below plots the results of the five trials. Note that all five trials converge to approximately the optimal solution locus in (20).

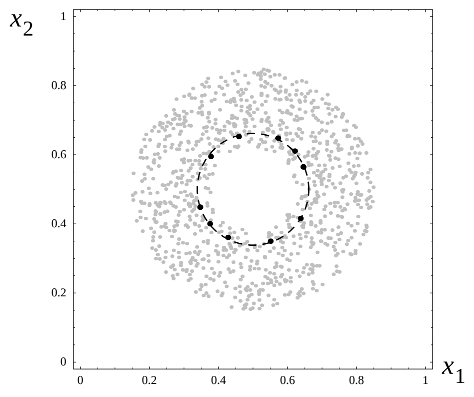


Figure 5

D. Convergence

In the previous two examples, we showed that the k-means algorithm converges to good near-optimal solutions for some simple, idealized cases. It turns out that this convergence is, in fact, guaranteed, since the k-means algorithm is simply a limiting case of the EM algorithm for estimating the parameters of a mixture of Gaussians. Recall that for the EM algorithm, the reestimation of the means μ_i (identical to the $\mathbf{z}_i$ here) is given by,

$$\mu_i = \frac{\sum_{j=1}^{n} P(\omega_i|\mathbf{x}_j)\mathbf{x}_j}{\sum_{j=1}^{n} P(\omega_i|\mathbf{x}_j)} \tag{22}$$

Assuming equal priors $P(\omega_i)$ and equal variances $\sigma_i^2 = \sigma^2$, it can be easily shown that,

$$\lim_{\sigma^2 \to 0} \mu_i = \lim_{\sigma^2 \to 0} \left[\frac{\sum_{j=1}^{n} P(\omega_i|\mathbf{x}_j)\mathbf{x}_j}{\sum_{j=1}^{n} P(\omega_i|\mathbf{x}_j)} \right] = \frac{1}{n_i}\left(\sum_{\mathbf{x}_j \in \omega_i} \mathbf{x}_j \right) = \mathbf{z}_i \tag{23}$$

Intuitively, as $\sigma^2 \to 0$,

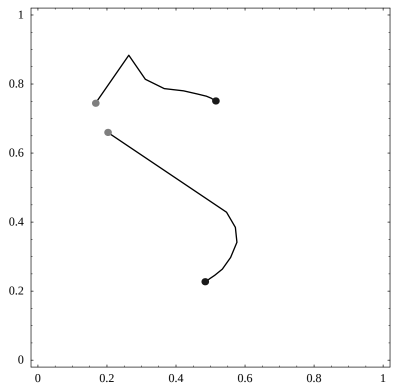
$$p(\mathbf{x}_j|\mu_i) \gg p(\mathbf{x}_j|\mu_l),\ i \neq l, \text{ where,} \tag{24}$$

$$D(\mathbf{x}_j, \mu_i) < D(\mathbf{x}_j, \mu_l) \tag{25}$$

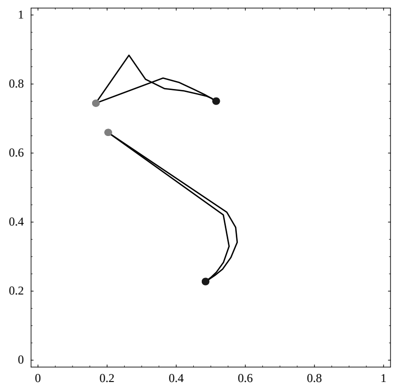
so that the likelihoods $p(\mathbf{x}_j|\mu_l)$, $i \neq l$ become insignificantly small compared to $p(\mathbf{x}_j|\mu_i)$.

Figure 6, for example, illustrates the convergence trajectory for the VQ and EM algorithms for one of the 100 trials in example #1. Note that the VQ and EM trajectories appear very similar for $\sigma = 0.001 \approx 0$.

Since the k-means VQ algorithm is a limiting case of the EM algorithm, its convergence is also guaranteed.[1] And, because the VQ reestimation equations are much faster to compute then the EM reestimation equations, the VQ algorithm is sometimes preferred in practice.



Example #1 VQ convergence.



Example #1 EM convergence for $\sigma = 0.1$ and $\sigma = 0.001$.

Figure 6

1. *One potential problem in VQ algorithms is that during convergence, one or more clusters (or classes) ω_i might become "empty." A typical solution to this problem splits the cluster which currently exhibits the largest distortion in two, and reassigns the empty class to part of the large-distortion cluster.*

3. The LBG VQ algorithm

A. Algorithm description

The well known LBG vector quantization (VQ) algorithm was proposed by Linde, Buzo and Gray [1] in 1980. It addresses the problem of VQ codebook initialization by iteratively generating codebooks $\{\mathbf{z}_i\}$, $i \in \{1, \ldots, 2^m\}$, $m \in \{0, 1, 2, \ldots\}$, of increasing size. The algorithm proceeds as outlined below. Note that the inner loop in the LBG VQ algorithm is equivalent to the k-means algorithm for the current value of L.

1. **Initialization**: Set $L = 1$, where L is the number of VQ codes in $\mathbf{Z}$, and let $\mathbf{z}_1$ be the centroid (e.g. mean) of the data set $\mathbf{X}$.

Outer loop

2. **Splitting**: Split each VQ code $\mathbf{z}_i$ into two codes, $\{\mathbf{z}_i, \mathbf{z}_{i+L}\}$,

   $$\mathbf{z}_{i+L} = \mathbf{z}_i - \underset{\sim}{\varepsilon} \text{ and } \mathbf{z}_i = \mathbf{z}_i + \underset{\sim}{\varepsilon}, \ \forall i, \tag{26}$$

   where,

   $$\underset{\sim}{\varepsilon} = \varepsilon\{b_1, b_2, \ldots, b_d\}, \tag{27}$$

   and ε is some small number, typically 0.0001. The b_k can be set to all 1's or to a random value of ± 1. Since the number of VQ codes in $\mathbf{Z}$ has been doubled, let,

   $$L = 2L \tag{28}$$

   Inner loop

   1. **Classification**: Classify each $\mathbf{x}_j$ into cluster or class ω_i such that,

      $$dist(\mathbf{x}_j, \mathbf{z}_i) \le dist(\mathbf{x}_j, \mathbf{z}_l), \ \forall l. \tag{29}$$

   2. **Codebook update**: Update the code for every cluster ω_i by computing its centroid,

      $$\mathbf{z}_i = \frac{1}{n_i}\left(\sum_{\mathbf{x}_j \in \omega_i} \mathbf{x}_j\right) \tag{30}$$

      where n_i is the number of vectors $\mathbf{x}_j$ in cluster ω_i.

   3. **Termination #1**: Stop when the distortion D has decreased below some threshold level, or when the algorithm has converged to a constant level of distortion.

3. **Termination #2**: Stop when L is the desired VQ codebook size.

There are two main advantages of the LBG VQ algorithm over the standard k-means algorithm. First, the algorithm is self-starting in the sense that problem-specific initialization is not required. Second, the LBG VQ algorithm automatically generates codebooks of size 2^m, $m \in \{0, 1, 2, \ldots\}$. This can be useful when we do not know *a priori* how large the VQ codebook needs to be for a specific application with a required maximum level of distortion. Presently, the LBG VQ algorithm is probably the VQ algorithm used most often in a number of different applications.

B. Example #1

Here, we illustrate the LBG algorithm with a simple example. Let $\mathbf{X}$ be a set of $n = 1000$ 2-dimensional vectors $\{\mathbf{x}_j\}$, $j \in \{1, 2, \ldots, 1000\}$, distributed uniformly in the unit square. Figure 7 illustrates the LBG VQ algorithm for the deterministic perturbation vector $\underset{\sim}{\varepsilon} = \{0.0001, 0.0001\}$. The codes for each L from 1 to $2^5 = 32$, are those at the end of the inner loop of the LBG algorithm, and the lines in each plot delineate the regions of the 2-dimensional space that are part of cluster ω_i.

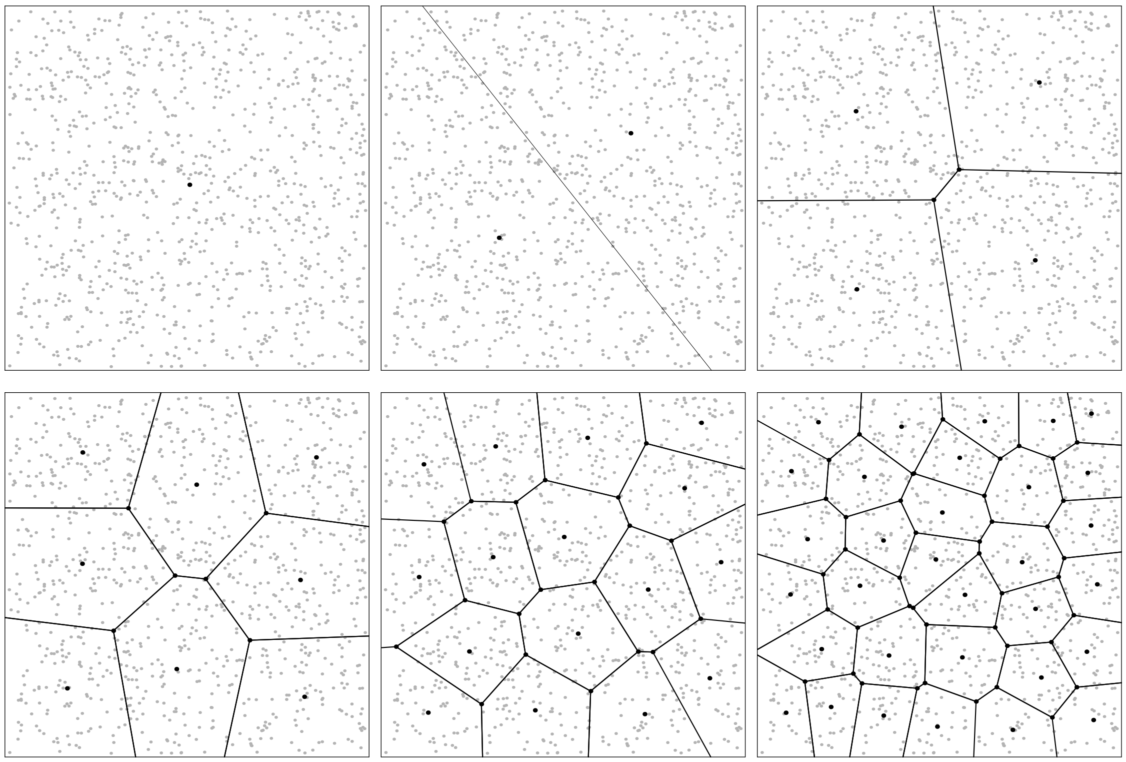


Figure 7: The LBG vector quantization for some random 2D data, as L ***equals 1, 2, 4, 8, 16 and 32.***

Each inner loop usually converges in only a few steps. Consider, for example, Figure 8, which illustrates the convergence of the inner loop as two codes are split into four. Note that within two steps (labeled arrows) the four codes are already located very close to their final values.

If a randomized perturbation vector $\underset{\sim}{\varepsilon}$ is used, the LBG algorithm may end up with slightly different codes $\mathbf{Z}$. Consider the two 32-prototype codebooks in Figure 9 below. These two VQ codebooks are generated for the same uniform data $\mathbf{X}$ using the LBG algorithm, but with different randomized perturbation vectors $\underset{\sim}{\varepsilon}$. Note that even though the two codebooks are slightly different, their total distortion D is almost the same.

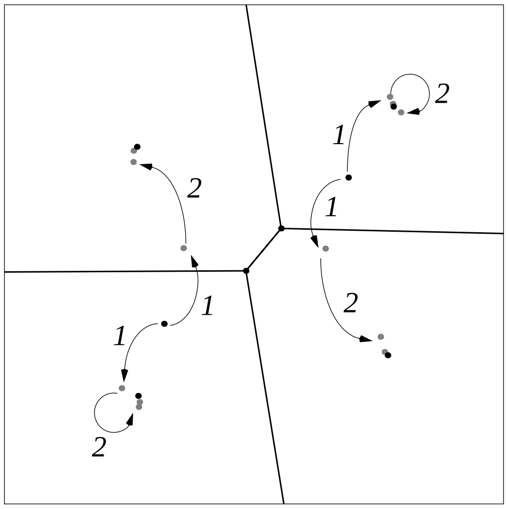
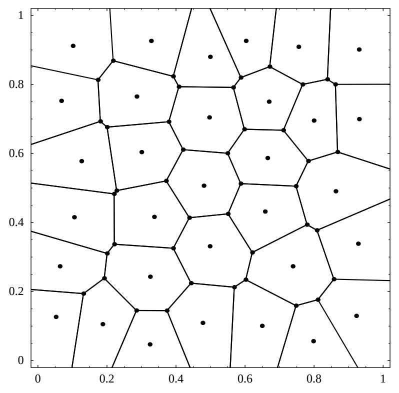
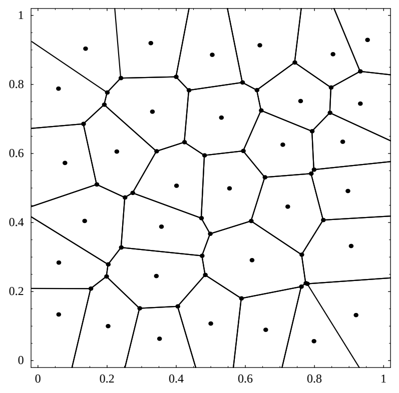


Figure 8: The inner loop of the LBG VQ algorithm when two codes are split into four.



Final distortion D = 2.19.

Figure 9



Final distortion D = 2.21.

C. Example #2

Here, we illustrate the LBG algorithm with another simple example. Let $\mathbf{X}$ be a set of $n = 1000$ 2-dimensional vectors $\{\mathbf{x}_j\}$, $j \in \{1, 2, \ldots, 1000\}$, distributed over the shaded region in Figure 10. Figure 10 illustrates the LBG VQ algorithm for the deterministic perturbation vector $\underset{\sim}{\varepsilon} = \{0.0001, 0.0001\}$. The codes for each L from 1 to $2^5 = 32$, are those at the end of the inner loop of the LBG algorithm, and the lines in each plot delineate the regions of the 2-dimensional space that are part of cluster ω_i.

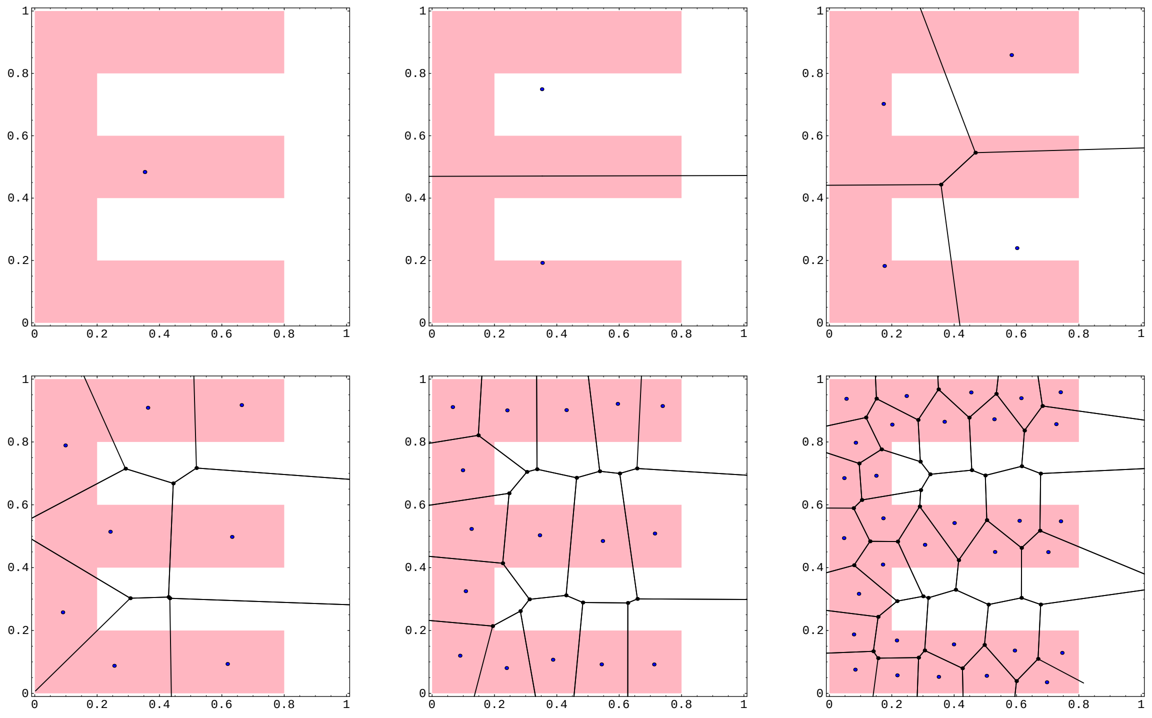


Figure 10: The LBG vector quantization for some random 2D data, as L ***equals 1, 2, 4, 8, 16 and 32.***

D. Example #3: color-based object recognition

In this example, we are interested in differentiating between two different model cars as they race at high speeds along a toy race track, as shown in Figure 11 below. Because of the interlaced nature of the NTSC signal, and the high scaled speed of the cars, the actual images of the cars are quite noisy, and vary significantly depending on where the cars are located along the track. Figures 12 and 13, for example, show three examples of cars #1 and #2 as they actually appear in the digitized images.

Here, we will use vector quantization to model each car as a discrete probability distribution over pixel color values, in order to discriminate between the two cars. First, we record the RGB (red, green, blue) pixel values for approximately 200 examples of each car; let us denote these as $\mathbf{X}_1$ and $\mathbf{X}_2$, respectively. Figure 14 plots

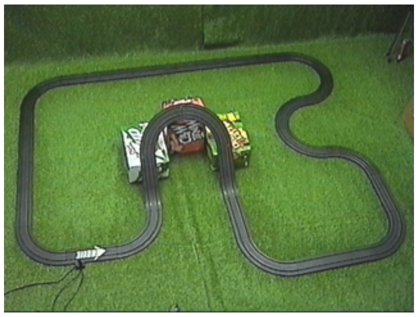


Figure 11

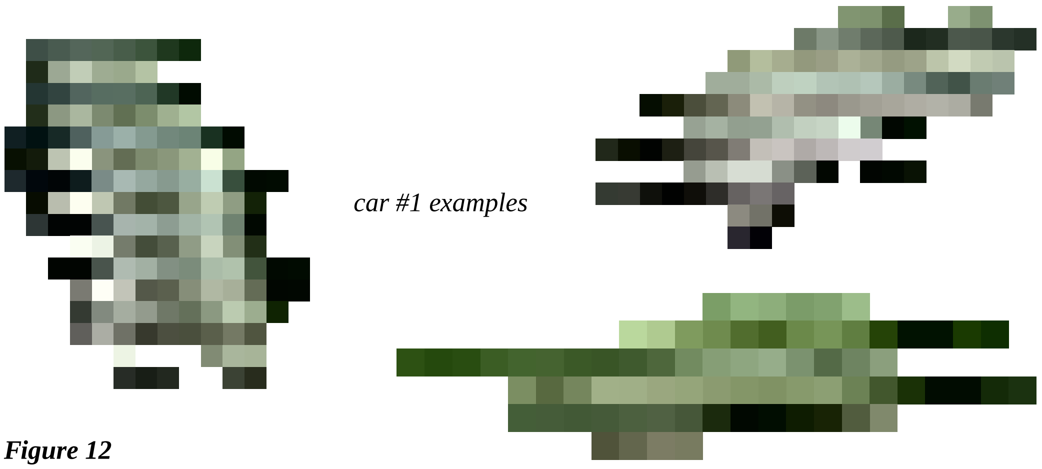


Figure 12

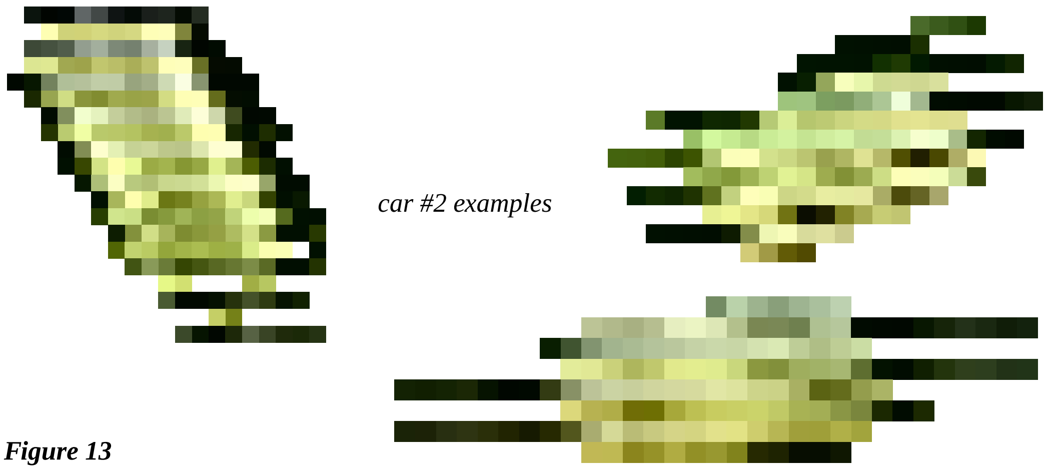
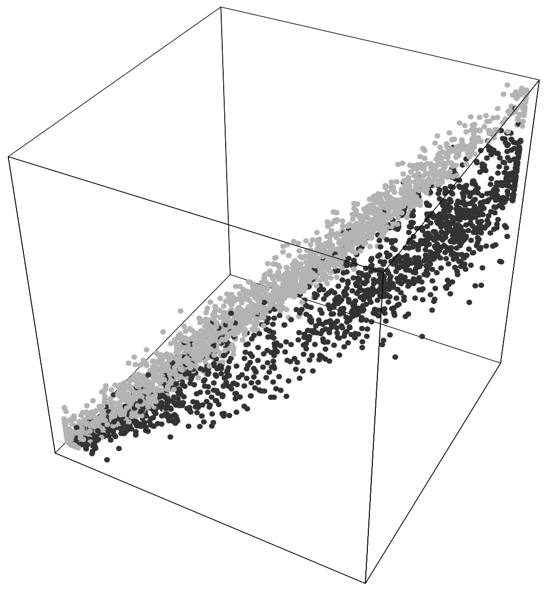


Figure 13


distribution of pixel values in RGB space for car #1 (light gray) and car #2 (dark gray) training data

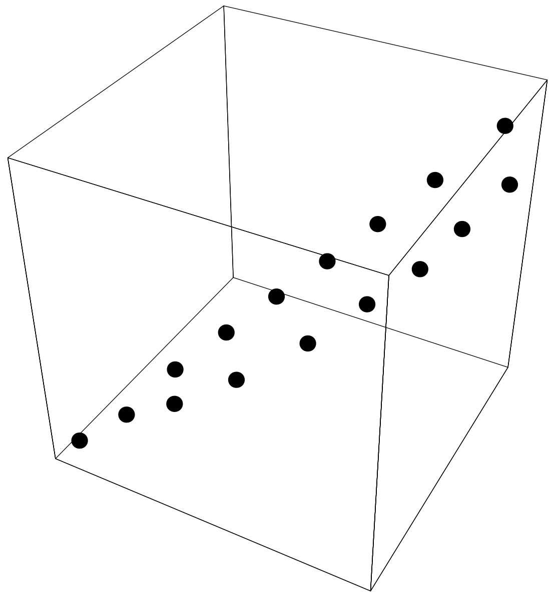

corresponding vector codebook with 16 prototype vectors (computed with the LBG algorithm)

Figure 14

these data sets in RGB space, where the light gray points correspond to $\mathbf{X}_1$, and the dark gray points correspond to $\mathbf{X}_2$.

We now compute a 16-prototype vector codebook $\mathbf{Z}$ using the LBG VQ algorithm, for the joint data set $\{\mathbf{X}_1, \mathbf{X}_2\}$. The resulting VQ codebook, which is also plotted in Figure 14, is next used to quantize both data sets $\mathbf{X}_1$ and $\mathbf{X}_2$. We then count the frequencies of occurrence of each prototype vector $\{1, 2, \ldots, 16\}$ in each data set, and normalize to fit probabilistic constraints. The resulting discrete probability models, λ_1 and λ_2, are plotted in Figure 15 below, and represent the discretized distribution of RGB color for each car.

If we now have an unknown car, as represented by a collection of RGB pixel values $\mathbf{X} = \{\mathbf{x}_j\}$, $j \in \{1, 2, \ldots, n\}$, that we want to classify as being either car #1 or car #2, we can evaluate the probability of $\mathbf{X}$ given each model λ_1 and λ_2,

$$P(\mathbf{X}|\lambda_k) = \prod_{j=1}^{n} P(\mathbf{x}_j|\lambda_k) = \prod_{j=1}^{n} P(l|\lambda_k), \; k \in \{1, 2\}, \tag{31}$$

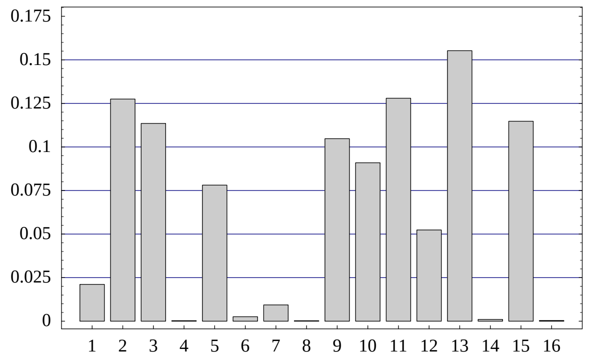
where l corresponds to the VQ prototype vector label that is closest to $\mathbf{x}_j$, such that,

$$dist(\mathbf{z}_l, \mathbf{x}_j) \le dist(\mathbf{z}_i, \mathbf{x}_j), \; \forall i. \tag{32}$$

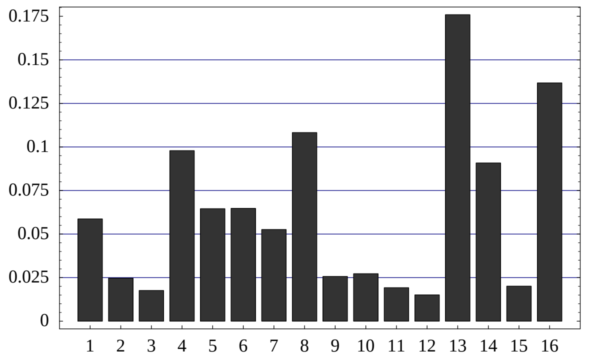
Of course, we will classify the unknown car as car #1 if $P(\mathbf{X}|\lambda_1) > P(\mathbf{X}|\lambda_2)$, and as car #2 otherwise. In Figure 16, for example, we plot $\hat{P}(\mathbf{X}|\lambda_k)$, $k \in \{1, 2\}$, for the six car examples (three each) in Figures 12 and 13, where,

$$\hat{P}(\mathbf{X}|\lambda_k) = 10^{\log P(X|\lambda_k)/n}, \; k \in \{1, 2\}. \tag{33}$$

In other words, $\hat{P}(\mathbf{X}|\lambda_k)$ simply represents the probability $P(\mathbf{X}|\lambda_k)$ normalized with respect to the number of RGB values n in $\mathbf{X}$. Note from Figure 16 that the VQ-based probability models in Figure 15 give us very good discrimination between the two cars.

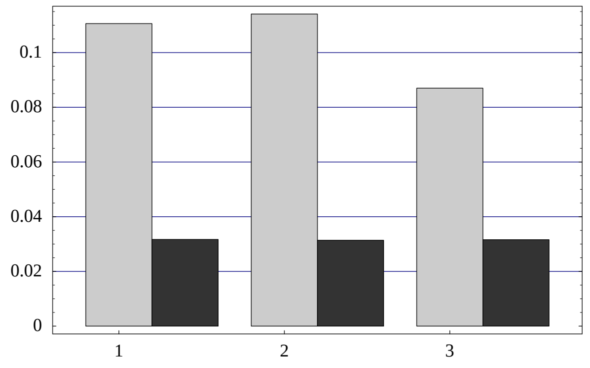


VQ-based, discrete probability model for car #1

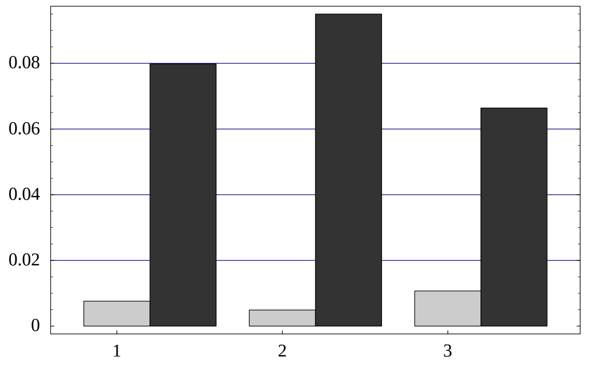


VQ-based, discrete probability model for car #2

Figure 15



$\hat{P}(\mathbf{X}|\lambda_1)$ *(light gray) and* $\hat{P}(\mathbf{X}|\lambda_2)$ *(dark gray) for car #1 examples in Figure 12*



$\hat{P}(\mathbf{X}|\lambda_1)$ *(light gray) and* $\hat{P}(\mathbf{X}|\lambda_2)$ *(dark gray) for car #2 examples in Figure 13*

Figure 16

[1] Y. Linde, A. Buzo and R. M. Gray, “An Algorithm for Vector Quantizer Design,” *IEEE Trans. on Communication*, vol. COM-28, no. 1, pp. 84-95, 1980.

[2] X. D. Huang, Y. Ariki and M. A. Jack, *Hidden Markov Models for Speech Recognition*, Chapter 4, pp. 111-35, Edinburgh University Press, Edinburgh, 1990.