VARIABLE METRIC FOR BINARY VECTOR QUANTIZATION

Ismo Kärkkäinen, Pasi Fränti

{ismo.karkkainen,pasi.franti}@cs.joensuu.fi
Department of Computer Science, University of Joensuu, P.O.Box 111, FI-80101 Joensuu, Finland

ABSTRACT

We present a new method for performing vector quantization of binary vectors. The proposed method varies the distance metric and updates the centroids in an optimal manner regarding the current metric. The corresponding centroids change from "soft", allowing variables of codevectors to have values between zero and one, to hard, pure binary codevectors.

1. INTRODUCTION

We consider the clustering problem in binary vector quantizer design. The aim is to find a set of *codevectors* (*codebook*) for a given data or *training set* so that the average pairwise distance between the data vector and corresponding codevector is minimized.

There is a multitude of methods for codebook generation [1], among the most used is *generalized Lloyd algorithm* (GLA) [2]. Unfortunately GLA is not well applicable to binary data since it is a descent method that improves the codebook gradually until a local optimum is reached. This is due to binary vectors having only two values, zero and one, which means that gradual changes in codevectors will not happen easily. The same problem also occurs with simulated annealing [3], pairwise nearest neighbor [4] and iterative splitting method [5].

1.1. The soft centroids method

In [6] the use of *soft centroids* was proposed. Instead of using binary codevectors the variables are allowed to have values in range [0, 1]. Euclidean distance is used and the codevectors are the mean vectors of each partition. Euclidean distance is a special case of L_d distance with $d = 2$. The L_d distance between a data vector x_i and centroid c_j of dimensionality K is given by:

$$L_d(x_i, c_j) = \left(\sum_{k=1}^{K} |x_{ik} - c_{jk}|^d \right)^{\frac{1}{d}} \quad (1)$$

For pure binary vectors, raising to power d has no effect since the difference is either 0 or 1.

Distortion between a data set X and a codebook C is expressed with mean error. Let p_i be the index of the codevector that is closest to x_i. Distortion is then:

$$E(X, C) = \frac{1}{N} \sum_{i=1}^{N} L_d(x_i, c_{p_i}) \quad (2)$$

Rounding to pure binary values occurs after the solution is obtained from the algorithm.

In this work, we generalize the method so that the soft centroids will turn gradually into hard centroids. Thus we avoid the sudden change that occurs when rounding is performed and let the softness disappear smoothly.

2. PROPOSED METHOD

The soft centroid method can be generalized by varying the metric used in distance calculations and by updating the codevectors so that they are optimal to the current metric. The exponent of the L_d metric (1) is changed by gradually decreasing it from a large value to 1 during the algorithm. The method is best suited for iterative algorithms such as GLA. The exponent d is decreased until it reaches one. Decrease can be e.g. linear or exponential. Note that d need not to be an integer. Especially when changing d from 2 to 1, several steps must be performed. Otherwise the result is practically the same as rounding the soft centroid in a single step as in [6].

Pseudocode for the proposed method applied to GLA is shown in Figure 1. Initially d is set to the maximum value and then decreased until the final value is reached.

```
Create initial solution.
Set d to d_max.
While d > d_min, do:
    Perform one GLA iteration.
    Decrease d.
Set d to d_min.
Iterate GLA until solution has converged.
Return final solution.
```

Figure 1. Pseudocode for the proposed algorithm.

0-7803-8554-3/04/$20.00 ©2004 IEEE.

To find the optimal value of the kth scalar of the jth centroid, for a given partition, we find the zero of the derivative of the internal distortion. In the following, for partition j and dimension k, we have q_{jk} zeroes and r_{jk} ones. For binary data, the internal distortion for one variable is:

$$D_{jk} = q_{jk} c_{jk}^{d} + r_{jk} \left(1 - c_{jk}\right)^{d} \tag{3}$$

Deriving this with respect to c_{jk}, and set to zero we get:

$$dq_{jk} c_{jk}^{d-1} - dr_{jk} \left(1 - c_{jk}\right)^{d-1} = 0$$

$$q_{jk} c_{jk}^{d-1} = r_{jk} \left(1 - c_{jk}\right)^{d-1} \tag{4}$$

$$\frac{c_{jk}}{1 - c_{jk}} = \left(\frac{r_{jk}}{q_{jk}}\right)^{\frac{1}{d-1}}$$

Now, let:

$$\alpha = 1/(d-1) \tag{5}$$

Note that d must be greater than 1. Solving for c_{jk}, we get:

$$c_{jk} = \left(\frac{r_{jk}}{q_{jk}}\right)^{\alpha} - \left(\frac{r_{jk}}{q_{jk}}\right)^{\alpha} c_{jk}$$

$$\left(1 + \left(\frac{r_{jk}}{q_{jk}}\right)^{\alpha}\right) c_{jk} = \left(\frac{r_{jk}}{q_{jk}}\right)^{\alpha} \tag{6}$$

Thus, the required update formula for $d > 1$ is:

$$c_{jk} = \frac{r_{jk}^{\alpha}}{q_{jk}^{\alpha} + r_{jk}^{\alpha}} \tag{7}$$

For $d = 1$, c_{jk} is set to 1 if $q_{jk} \le r_{jk}$, zero otherwise. If $d = 2$ the result of formula (7) simplifies to the mean value of the variable, since $\alpha = 1$, as seen from (5). Then (7) becomes number of ones divided by the sum of the number of ones and zeroes in one dimension, which is the number of vectors in the partition.

If decresing of d is done in linear manner, then the step value is subtracted from the previous value of d and the algorithm continues until d reaches 1. If change is exponential, d_{min} is subtracted from d, then the difference is multiplied with a constant g and then d_{min} is added to the result. Thus d_{min} is approached slowly in the end regardless of the value of d_{min}. Update formula is:

$$d_i = g\left(d_{i-1} - d_{\min}\right) + d_{\min} \tag{8}$$

The effect of d on the c_{jk} is that for large values of d, even a few differing values in the partition tend to pull the result towards 0.5. Therefore the centroids are all initially close to each other at the center of the space and then move towards corners as d decreases, each settling to a corner of the space when $d = 1$. Figure 2 shows how value of c_{jk} changes as function of d as d decreases from 6 to 1. It can be clearly seen that initially c is quite close to 0.5 regardless of how many zeroes and ones there are. Once d approaches 1 all curves approach the dominant value. In the case of $q = r$ the value of c remains at 0.5 until it is rounded to 1 in the end. The mean of q and r can be seen at $d = 2$.

Since the exponent of the metric needs to be changed during the process, the method is really suitable only for iterative algorithms. When using hierarchical algorithms the results from previous levels would need to be updated or left to contain sub-optimal values if d would change as clusters are joined. The optimal location of the centroid changes during the iterations as a result of (7) even if the partitioning would remain the same. Therefore, we cannot expect that e.g. GLA would converge before d has reached the final value. During testing we performed additional iterations after d had reached the final value to ensure that the solution has converged.

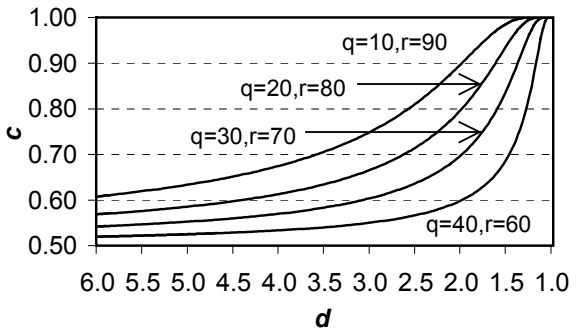


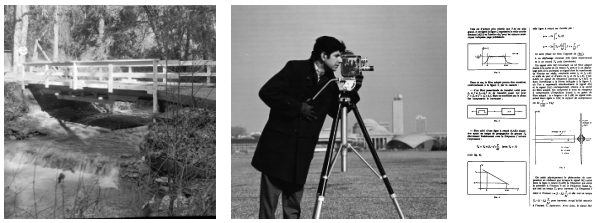
Figure 2. Effect of d to the centroid values.

3. TEST RESULTS

The four binary data sets used in testing were Bridge, Camera, CCITT-5, and DNA. Bridge and Camera are 4×4 pixel blocks taken from gray-scale images after a BTC-type quantization into two levels according to the mean value of the block [7]. CCITT-5 is obtained by taking 4×4 pixel blocks from the standard CCITT-5 binary test image. Data set characteristics and the size of the codebook used are shown in Table 1 and the original images in Figure 3.

Table 1. Binary data set statistics

Data set	Dimensionality	Vectors	Codebook size
Bridge	16	2813	256
Camera	16	3172	256
CCITT-5	16	1784	256
DNA	58	270	4



Bridge Camera CCITT-5

Figure 3. Images used for the binarized image data sets.

For testing the effect of changing the L_d metric we used GLA and simulated annealing (SA). After d has changed, we update the partitions and centroids during a single iteration of GLA or SA. After d no longer changes we iterate GLA until the solution converges, which usually required only one or two additional iterations. We compare the results of our algorithm to those computed in [6]. The algorithms using Euclidean distance and distortion given in (2) are GLA, simulated annealing, pairwise nearest neighbor (PNN) and iterative splitting method (Split).

To study the effect of the range of d, we tried several ranges of d so that the minimum and final value of d was always 1. Table 2 shows average distortions obtained with GLA and SA. Columns marked linear mean that d was changed by subtracting the given value until d reached 1. Columns marked exponential had d changed using (8) and the value given is g. Values are averages of 100 runs each. We see that exponential decrease of d gives better results than linear for GLA. Also slower decrease yields better results in general. Using greater starting value d_0 than 2 is apparently required for better results since the distortion drops considerably when d_0 grows from 2 to 3. Table 3 shows the same results for DNA. They are similar to those in Table 2. The results for the two remaining data sets are also similar.

Table 2. Average distortion with different parameter combinations for Bridge

	GLA					SA	
	Linear			Exponential		Exponential	
d_0	0.05	0.1	0.25	0.95	0.99	0.95	0.99
2	1.325	1.335	1.359	1.318	1.314	1.278	1.254
3	1.319	1.327	1.346	1.312	1.310	1.269	**1.245**
4	1.318	1.325	1.344	**1.310**	1.309	**1.268**	1.246
5	**1.317**	1.324	1.344	1.311	1.309	1.270	1.249
6	1.318	**1.323**	**1.342**	1.311	**1.307**	1.271	1.252

We present distortion values along with the results from [6]. All original results for SA are averages from 10 runs and for GLA averages from 100 runs. For the proposed method, results for SA are from 100 runs. For comparison, results with pure binary centroids are also included.

Table 3. Average distortion with different parameter combinations for DNA

	GLA					SA	
	Linear			Exponential		Exponential	
d_0	0.05	0.1	0.25	0.95	0.99	0.95	0.99
2	6.679	6.718	6.728	6.646	6.668	6.571	6.571
3	**6.646**	6.688	6.701	**6.628**	6.641	**6.570**	**6.570**
4	6.685	6.644	**6.647**	6.643	6.641	**6.570**	**6.570**
5	6.662	6.661	6.671	6.630	**6.627**	**6.570**	**6.570**
6	6.663	**6.643**	6.659	6.637	6.682	**6.570**	**6.570**

Results for Bridge and Camera, see Figures 4 and 5, show improvement when using the proposed method. SA with the variable metric outperforms all other algorithms. Also simple GLA with variable metric outperforms Split and PNN, for which the proposed method is not applicable.

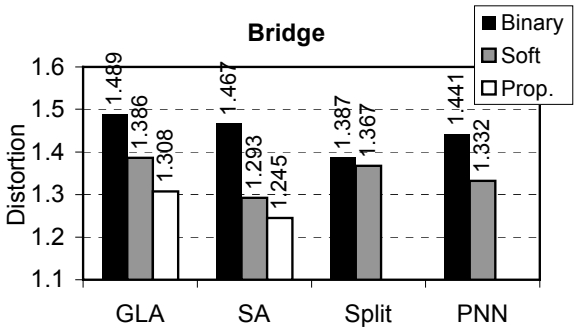


Figure 4. Results for Bridge data set.

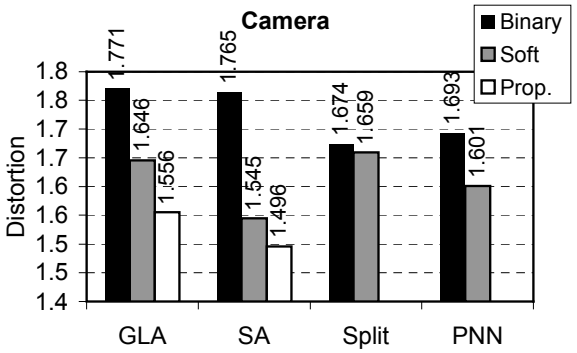


Figure 5. Results for Camera data set.

With CCITT-5 the results with GLA using variable metric are worse than even those of binary GLA, as can be seen from Figure 6. SA gets quite close to the best result by PNN. Split underperforms badly for some reason. Results for DNA data set are shown in Figure 7. SA with variable metric outperforms Split and gets very close to PNN.

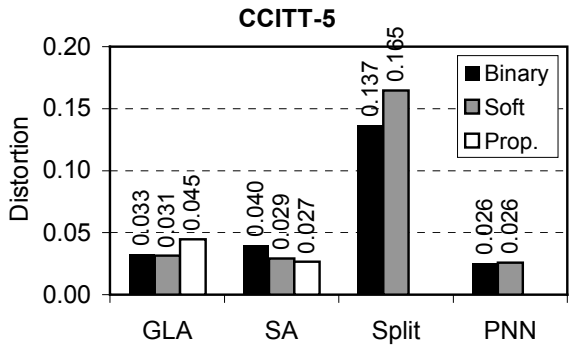


Figure 6. Results for CCITT-5 data set.

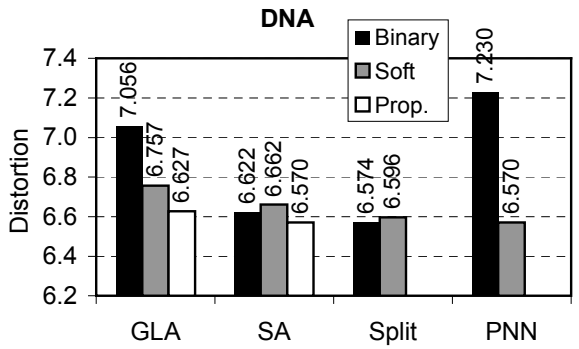


Figure 7. Results for DNA data set.

Table 3 shows how the best results for each data set with different manners of decreasing d relate to the best results obtained using either pure binary centroids or soft centroids. Negative numbers indicate that the solution was better than one obtained with binary centroids or soft centroids. The best results are taken over all ranges that were used for the specific method of changing d. This clearly shows that GLA can get quite close to other methods using variable metric with the exception of CCITT-5 data set where the results are clearly worse with over 70 % greater distortion. Using variable centroids produces considerably worse results even when compared to GLA with binary and soft centroids, as seen from Figure 6.

Table 3. Percentage of distortion difference relative to the best distortion obtained using binary or soft centroids.

	GLA					SA	
	Linear			Exponential		Exponential	
Data set	0.05	0.1	0.25	0.95	0.99	0.95	0.99
Bridge	1.91	2.38	3.90	1.35	1.18	-1.84	**-3.64**
Camera	1.37	1.91	3.28	0.80	0.70	-1.48	**-3.14**
CCITT-5	75.1	76.1	78.9	74.4	71.9	13.9	**2.67**
DNA	1.16	1.12	1.17	0.89	0.86	**0**	**0**

Comparing the results to the best of all results, shown in Table 4, we see more clearly that slow exponential decrease of d gives the best results for both GLA and SA. Considering the fact that GLA will not converge until d has converged, there are no practical speed advantages of using GLA instead of SA.

Table 4. Percentage of distortion difference relative to the best distortion obtained using binary or soft centroids.

	GLA					SA	
	Linear			Exponential		Exponential	
Data set	0.05	0.1	0.25	0.95	0.99	0.95	0.99
Bridge	5.76	6.24	7.83	5.18	5.01	1.87	**0**
Camera	4.66	5.22	6.63	4.07	3.97	1.72	**0**
CCITT-5	75.1	76.1	78.9	74.4	71.9	13.9	**2.67**
DNA	1.16	1.12	1.17	0.89	0.86	**0**	**0**

4. CONCLUSIONS

We have presented a way of performing vector quantization on binary data so that the problems arising from the use of pure binary codevectors can be decreased. Transition to pure binary vectors occurs naturally as the L_d metric is gradually changed to Manhattan distance and no rounding step is needed. We have shown that for SA this produces better results in all cases and for GLA better results in most cases than using soft centroids and then rounding the results. Future work includes investigating if the idea is suitable also for non-binary data.

5. REFERENCES

[1] Gersho, A. and R.M. Gray, *Vector Quantization and Signal Compression*, Kluwer Academic Publishers, Dordrecht, 1992.

[2] Y. Linde, A. Buzo, and R.M. Gray, "An Algorithm For Vector Quantizer Design", *IEEE Trans. Communications* 28(1), pp. 84-95, January 1980.

[3] K. Zeger, and A. Gersho, "Stochastic Relaxation Algorithm for Improved Vector Quantizer Design", *Electronics Letters*, 25(14), pp. 896-898, July 1989.

[4] W.H. Equitz, "A New vector Quantization Clustering Algorithm", *IEEE Trans. Acoust. Speech Signal Process*. 37(19), pp. 1568-1575, October 1989.

[5] X. Wu, and K. Zhang, "A Better Tree-structured Vector Quantizer", *IEEE Proc. Data Compression Conf.*, Snowbird, Utah, pp. 392-401, 8-11 April 1991.

[6] P. Fränti, and T. Kaukoranta, "Binary Vector Quaantizer Design Using Soft Centroids", *Signal Processing: Image Communication* 14, pp. 677-681, 1999.

[7] P. Fränti, T. Kaukoranta, and O. Nevalainen, "On The Design of a Hierarchical BTC-VQ Compression System", *Signal Processing: Image Communication* 8(6), pp. 551-562, September 1996.