A Kernel Approach for Vector Quantization with Guaranteed Distortion Bounds

Michael E. Tipping & Bernhard Schölkopf
Microsoft Research, 1 Guildhall Street, Cambridge, UK
{mtipping,bsc}@microsoft.com

Abstract

We propose a kernel method for vector quantization and clustering. Our approach allows *a priori* specification of the maximally allowed distortion, and it automatically finds a sufficient representative subset of the data to act as codebook vectors (or cluster centres). It does not find the minimal number of such vectors, which would amount to a combinatorial problem; however, we find a 'good' quantization through linear programming.

1 Introduction

Vector quantization (VQ) is a ubiquitous data summarization technique [3], commonly utilised in signal processing applications, most notably in speech and image coding. The purpose of VQ is to represent a set of ℓ data vectors

$$x_1, \dots, x_\ell \in \mathcal{X} \tag{1}$$

by a reduced number of m 'codebook' vectors

$$y_1, \dots y_m \in \mathcal{X}, \tag{2}$$

such that some measure of average distortion is minimized when each x is represented by the 'nearest' (in terms of some desired metric over $\mathcal{X}$) y.

Often, a 'good' codebook is found by a greedy local minimization of the distortion measure, an example of which is that based on the standard ℓ_2 metric:

$$E_{VQ} = \sum_{i=1}^{\ell} \|x_i - y(x_i)\|^2, \tag{3}$$

where

$$y(x_i) = \operatorname*{argmin}_{y_j} \|x_i - y_j\|^2. \tag{4}$$

In practice, one would specify m, initialize y and utilise one's favourite optimization algorithm to minimize a distortion measure such as (3). Compression is then realized since, given receiver knowledge of the codebook, each x can be encoded in $\log_2 m$ bits.

The approach of pre-specification of m is natural, as it may be influenced by external factors such as bandwidth or storage capacity. Here, however, we consider the case where it is desired to guarantee (for the 'training' set) a *maximum* level of distortion for any encoded x and automatically find an appropriate value for m. A restriction we impose is that the codebook vectors must be a subset of the data. Finding the minimal such subset which can represent the data with a given level of distortion is of course a combinatorially hard problem, but we can obtain an effective 'sparse' codebook using linear programming methods as detailed in the next section. In Section 3 we offer some illustrative examples of application of the method, and wrap up with some discussion in Section 4.

We are aware of one other VQ implementation (which is quite different in that there is no distortion guarantee) which exploits linear programming [4], while there has been recent interest in providing guarantees for other algorithms [5].

For other kernel approaches to unsupervised learning using linear programming formulations, cf. [1, 6].

2 Clustering via Linear Programming

Suppose we are given data

$$x_1, \dots, x_\ell \in \mathcal{X}, \tag{5}$$

where $\mathcal{X}$ is some space endowed with a metric d.

By the term *kernel* we will presently refer to functions

$$k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}. \tag{6}$$

In particular, we will consider the kernel that indicates whether two points lie within a distance of $R \geq 0$,

$$k(x, x') = \mathbf{1}_{\{(x,x') \in \mathcal{X} \times \mathcal{X}:\ d(x,x') \leq R\}} \tag{7}$$

We consider what in the Support Vector (SV) community is sometimes called the empirical kernel map [7]

$$\phi_\ell(x) = (k(x_1, x), \ldots, k(x_\ell, x)). \tag{8}$$

Suppose we can find a vector $\mathbf{w} \in \mathbb{R}^\ell$ such that

$$\mathbf{w}^\top \phi_\ell(x_i) > 0 \tag{9}$$

holds true for all $i = 1, \ldots, \ell$. Then each point x_i lies within a distance R of some point x_j which has a positive weight $w_j > 0$. To see this, note that otherwise all nonzero components of $\mathbf{w}$ would get multiplied by a component of ϕ_ℓ which is 0, and the above dot product would equal 0.

Therefore, up to an accuracy of R (measured in the metric d), the x_j with nonzero coefficients w_j can be considered an approximation of the complete training set (formally, they define an R-cover of the training set in the metric d). However, so far we have said nothing that prevents us from simply using the trivial solution, $w_i = 1$ for all i. In order to keep the number of nonzero coefficients small, we will consider the following optimization problem: for some $q \geq 0$, compute

$$\min_{\mathbf{w} \in \mathbb{R}^\ell} \quad \|\mathbf{w}\|_q \tag{10}$$

$$\text{subject to} \quad \mathbf{w}^\top \phi_\ell(x_i) \geq 1. \tag{11}$$

In this formulation, we have slightly modified the constraint (9). To understand this, note that whenever we have a vector $\mathbf{w}$ satisfying (9), we can rescale it to satisfy (11). For the optimization, however, we must use (11) to ensure that the target function (10) cannot be trivially minimized by shrinking $\mathbf{w}$.

In vector quantization, we are looking for a small codebook of vectors to approximate the complete training set. Therefore, the ideal thing to do would be to utilize $q = 0$, in which case $\|\mathbf{w}\|_q$ simply counts the number of nonzero coefficients of $\mathbf{w}$. This, however, leads to a combinatorial optimization problem. 'Nice' optimization problems can be obtained using $q = 1$ or $q = 2$. We will use the former, which is closer to $q = 0$. From the work on sparse decompositions (e.g. [2]), linear programming machines (e.g. [8]), and reduced set SVM methods [7], it is well known that penalizers of the form $\|\mathbf{w}\|_1$ lead to sparse solutions.

In practice, we use a slightly adjusted penaliser of the form $\sum_i \gamma_i |w_i|$, where $\gamma_i := n_i^{-1}$, with n_i the number of examples in the support of $k(x_i, x)$. While not affecting the constraints, this assists in the removal of more 'redundant' codebook vectors.

To derive the final optimization problem, we introduce the matrix $K_{ij} = k(x_i, x_j)$ and decompose $\mathbf{w}$ according to $w_i = \alpha_i - \beta_i$, ending up with

$$\min_{\boldsymbol{\alpha}, \boldsymbol{\beta} \in \mathbb{R}^\ell} \quad \sum_{i=1}^{\ell} \gamma_i (\alpha_i + \beta_i) \tag{12}$$

$$\text{subject to} \quad K(\boldsymbol{\alpha} - \boldsymbol{\beta}) \geq 1 \tag{13}$$

$$\boldsymbol{\alpha}, \boldsymbol{\beta} \geq 0, \tag{14}$$

where the inequalities are understood to hold for each component of the lhs vectors.

At the end of the optimization, the nonzero w_i correspond to a sufficient set of codebook vectors y_i. Due to inherent symmetries in many tasks, this set can still be unnecessarily large. We thus perform a simple final 'pruning' step to remove some of the remaining redundant vectors in the codebook, by sequentially removing any codebook vector that does not exclusively explain any data vector. Typically, this results in the removal of a further 1%–5% of the codebook, and so the majority of the sparsification still occurs within the linear program.

3 Examples

3.1 Synthetic data in 2 dimensions

As a simple illustrative example, Figure 1 shows quantizations obtained using the above linear programming technique (LP-VQ) of two-dimensional data uniformly distributed in the unit square, for values of maximum distortion $R = 0.1$, 0.2, 0.3 and 0.4, and d being the Euclidean ℓ_2 distance.

3.2 Block image coding

A popular application of VQ methods is to compress/encode images. Although we would not propose LP-VQ (or even VQ in general — fixed basis transforms such as JPEG are generally superior) as the best methodology for such a task, image coding provides an easily visualized task on which to demonstrate the method at work.

Figure 2 shows a 384×256 test image (actually of 24-bit colour depth, although, of course, rendered here in black and white). This image was decomposed into non-overlapping 8×8 blocks, each of which was processed to give a 192-dimensional vector of bytes (one byte per RGB colour component per pixel). The linear programming VQ procedure was then performed in this high-dimensional space, and we show results

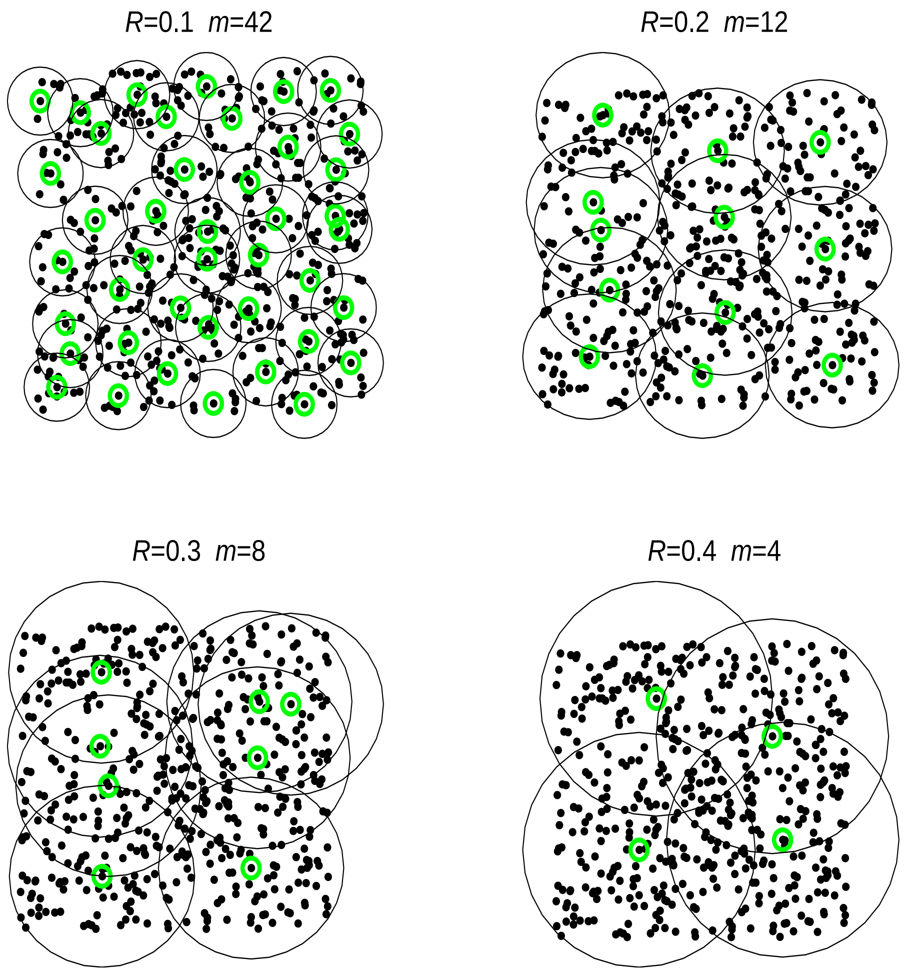


Figure 1: Results of performing guaranteed-distortion LP-VQ on two dimensional uniform data with varying distortion levels R. The m quantizing vectors, found automatically, are shown circled and their number is given at the head of each plot. Circles of $d(x, y) = R$ are also shown for each codebook vector.

for two values of maximum distortion, $R = 200$ and 500, with d being the ℓ_2 distance.[1]

For comparison, a reconstruction by the the popular *Linde-Buzo-Gray* (LBG) algorithm [3] algorithm is also given in Figure 2. This is a conventional fixed-codebook-size, but variable maximum distortion, VQ method, which has spawned many derivatives. The codebook size used was identical to the LP-VQ $R = 500$ case.

The corresponding sizes of the compressed images are shown in the figure. These are computed as the sum of the codebook size, which of course is automatically determined by the algorithm in the LP-VQ case, and the number of bits required to code all the blocks with respect to that codebook. Note that here we are assuming that the codebook is to be transmitted to the decoder.[2] Statistics summarising the performance of the compression algorithms are given in Table 1.

[1]Other choices for d are possible, too. For instance, the ℓ_∞ distance would lead to a pixel-wise upper bound on the distortion. One can generalize the approach further by using kernels which are non-constant within their support. If, for instance, a kernel decays with increasing distance, then it is more expensive for the algorithm to code a point which is far away from a given codebook vector. Effectively, this corresponds to different cost functions for measuring the distortion error (cf. (3)). Provided the support is bounded, we retain the guaranteed distortion bound.

[2]Alternatively, we might attempt to elucidate a good (and no doubt much larger) general-purpose codebook from a large number of images. Knowledge of the codebook would then be shared *a priori* by both sender and receiver, and a compression gain might be obtained through avoiding the necessity to expensively code all the quantization vectors with the image.

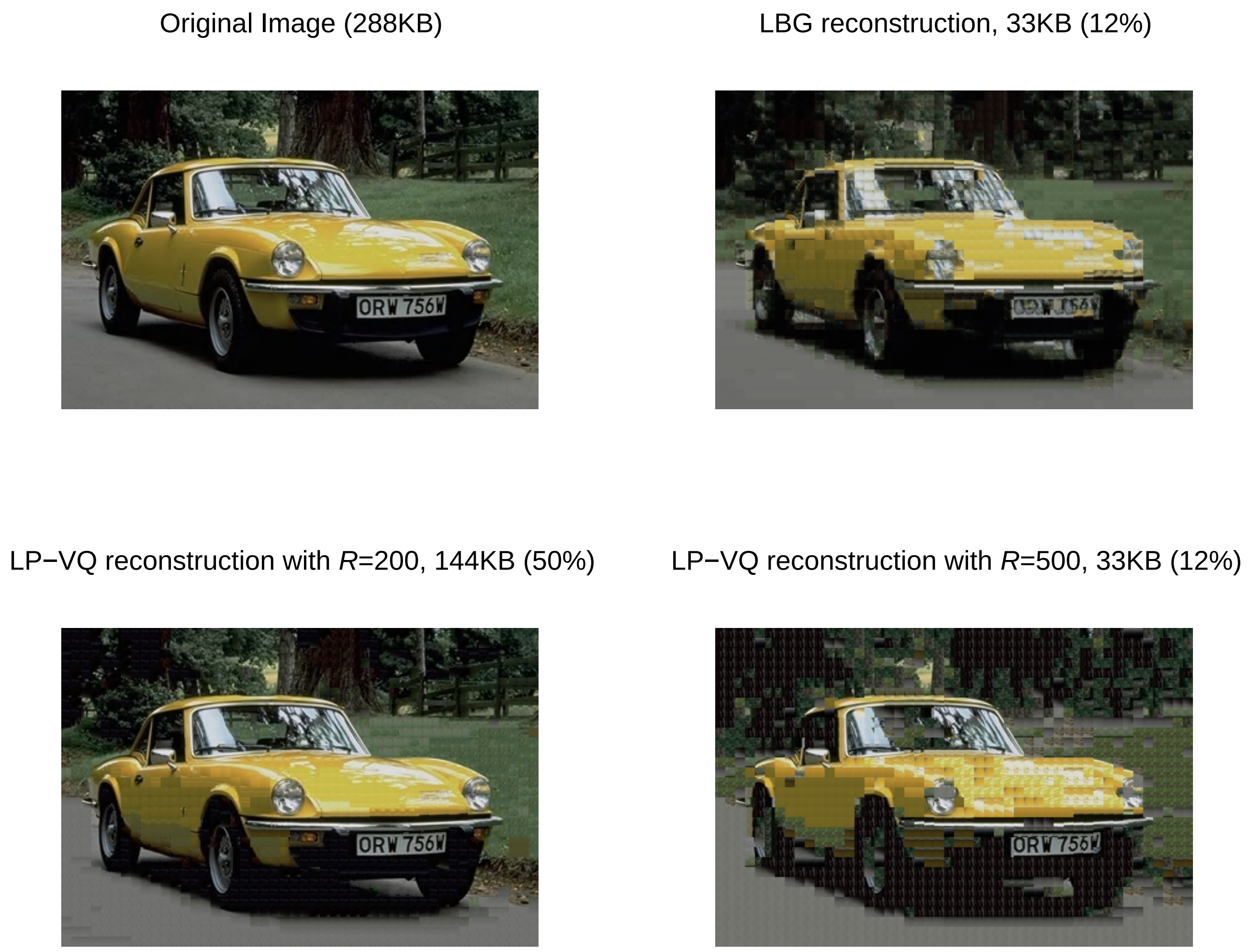
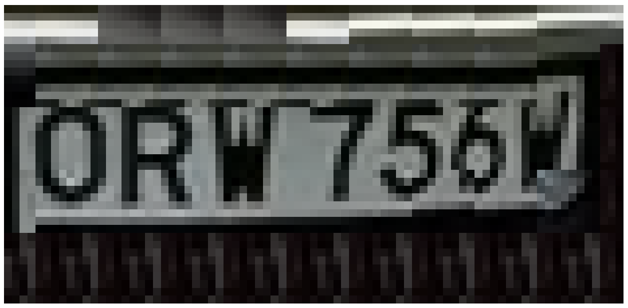

Original Image (288KB)
LBG reconstruction, 33KB (12%)
LP−VQ reconstruction with R=200, 144KB (50%)
LP−VQ reconstruction with R=500, 33KB (12%)
ORW 756W

Figure 2: Two reconstructions using LP-VQ (bottom left & right), and one using LBG (top right), of a colour image, original shown top left. The maximum distortion value R is given for the LP-VQ reconstructions, along with the size of the images, in kilobytes, for all cases. In the LBG case, the codebook size used was the same as that found automatically by the $R = 500$ LP-VQ case.

Image	**Size**	**Ratio**	R	m	E_{max}	E_{rms}
Original	288KB	100%	0	1536	0	0
LP-VQ Reconstruction	144KB	50%	200	757	199.9	88.7
LP-VQ Reconstruction	33KB	12%	500	170	499.5	283.8
LBG Reconstruction	33KB	12%	-	170	816.4	229.8

Table 1: Statistics for the image encodings of Figure 2. Along with the image sizes and value of R, the maximum distortion guarantee, given in the original figure, values of m, the codebook size, and E_{max} and E_{rms}, the maximum and root-mean-square distortion are given. Note that E_{max} for LP-VQ is bounded by R, and is considerably larger for LBG, where E_{rms} is lower.

LP−VQ (R=500) detail



LBG equivalent

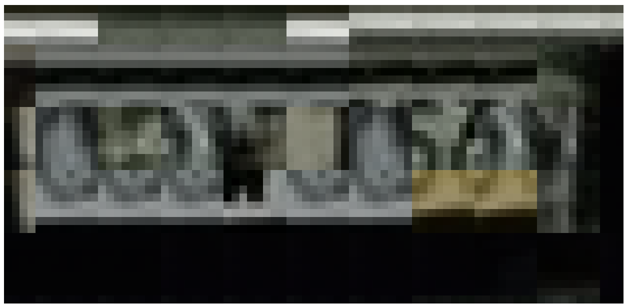


Figure 3: Detail of the reconstruction of the number plate in Figure 2 by the $R = 500$ LP-VQ model and the equivalently sized LBG coder. In addition to the poorer reconstruction in the latter case, the lower right of the plate exhibits a shading effect due to mixing of a block of yellow car body with part of the number plate (not visible in the black-and-white rendition). This is a consequence of the adaptation of codebook vectors in that case.

We utilize a close-up detail of this image to illustrate a contrastive feature of our approach when compared with more conventional VQ methods. A clear advantage of the LBG algorithm, in terms of overall distortion minimization, is that codebook vectors need not coincide with the data points but are free to adapt. The result of this may be seen in Table 1 where, as expected, the LBG algorithm gives both lower average distortion and greater maximum distortion for the same codebook size.

Figure 3 shows a close-up of the reconstructed number plate from the car of Figure 2. Of interest is the effect the maximum distortion guarantee in the LP-VQ case has on the image characteristics. In 'typical' areas of image, most notably the car body and the background, the LBG algorithm provides the better rendering (indeed, overall it is arguably subjectively superior). However, in atypical areas, such as the car number plate in Figure 3, the LP-VQ approach offers significantly greater fidelity. This is a consequence of the fact that the LBG method (which is closely related to K-means) seeks to represent, in some sense, the *density* of the data vectors while the LP-VQ method only seeks to model the *support*. Thus the LBG algorithm can afford a large error on one vector, as this may be offset by many lower errors for vectors in regions of high density. The LP-VQ algorithm by construction must code *all* vectors to within the set distortion level R. It should be noted that if the data were contaminated by bad outliers, then the compression rate would suffer. Nevertheless, if the 'outliers' correspond to meaningful structure, then the present approach may be advantageous.

4 Discussion

We have proposed a novel approach to vector quantization which lets us specify an upper bound on the distortion incurred for each coded vector. The algorithm automatically determines the number of codebook vectors required. It does not find the *minimal* such number, which would amount to a combinatorial problem, but it solves the problem via linear programming, which can be performed efficiently with existing optimization packages,[3] and which is provably of polynomial time complexity.

That the codebook vectors must be a subset of the data is clearly a limitation which will inevitably lead to poorer overall distortion performance. However, this does imply that LP-VQ reconstructed vectors are representative of the data, which need not be the case in an LBG-style approach where codebook vectors may potentially be located in regions where the data has no support. This was exemplified by part of the detail of Figure 3. We thus expect the proposed linear programming method to be effective in applications where either this property or the facility to impose a bound on the distortion may be advantageous (the potential effects of which were again showed by Figure 3). An additional further use that we have not explored in detail is to use the LP-VQ codebook as an *initialization* for one of the conventional methods (the importance of making a good choice of initial codebook in VQ algorithms is well-known).

Of course, one could propose heuristic algorithms with similar distortion constraints: for example, an LBG-

[3]Results obtained within this paper were derived using the `linprog` program from MATLAB's optimization toolbox.

style algorithm which incorporates additional centroids until the maximum distortion was below a desired value, similar to the method of [9]. However, it is generally well appreciated that such greedy sequential algorithms exhibit a considerable degree of suboptimality, while the LP approach presented here offers an effective and principled way of finessing the combinatorial difficulty.

References

[1] C. Campbell and K. Bennett. A linear programming approach to novelty detection. 2000. Submitted.

[2] S. Chen, D. Donoho, and M. Saunders. Atomic decomposition by basis pursuit. *Siam Journal of Scientific Computing*, 20(1):33–61, 1999.

[3] A. Gersho and R. M. Gray. *Vector Quantization and Signal Compression.* Kluwer, 1992.

[4] J.-H. Lin and J. S. Vitter. Nearly optimal vector quantization via linear programming. In J. A. Storer and M. Cohn, editors, *Data Compression Conf.*, pages 22–31. IEEE Computer Society Press, 1992.

[5] P. Niyogi and N. Karmarkar. An approach to data reduction and clustering with theoretical guarantees. In P. Langley, editor, *Proceedings of the 17th International Conference on Machine Learning*, pages 679 – 686, San Francisco, 2000. Morgan Kaufman.

[6] G. Rätsch, B. Schölkopf, S. Mika, and K.-R. Müller. SVM and boosting: One class. In *Advances in Neural Information Processing Systems*, 2000. Submitted.

[7] B. Schölkopf, S. Mika, C. Burges, P. Knirsch, K.-R. Müller, G. Rätsch, and A. Smola. Input space vs. feature space in kernel-based methods. *IEEE Transactions on Neural Networks*, 10(5):1000 – 1017, 1999.

[8] J. Weston, A. Gammerman, M. Stitson, V. Vapnik, V. Vovk, and C. Watkins. Support vector density estimation. In B. Schölkopf, C. J. C. Burges, and A. J. Smola, editors, *Advances in Kernel Methods — Support Vector Learning*, pages 293–306, Cambridge, MA, 1999. MIT Press.

[9] A. Ypma and R. P. W. Duin. Support objects for domain approximation. In L. Niklasson, M. Boden, and T. Ziemke, editors, *Proceedings of the 8th International Conference on Artificial Neural Networks*, 1998.