(2)

RSRE
MEMORANDUM No. 4392

ROYAL SIGNALS & RADAR ESTABLISHMENT

ASYMPTOTIC CODE VECTOR DENSITY IN TOPOGRAPHIC VECTOR QUANTISERS

Author: S P Luttrell

PROCUREMENT EXECUTIVE,
MINISTRY OF DEFENCE,
R S R E MALVERN,
WORCS.

R S R E MEMORANDUM No. 4392

CONDITIONS OF RELEASE

0074989 BR-114336

************************** DRIC U

************************** DRIC Y

Reports quoted are not necessarily available to members of the public or to commercial organisations.

Royal Signals and Radar Establishment

Memorandum 4392

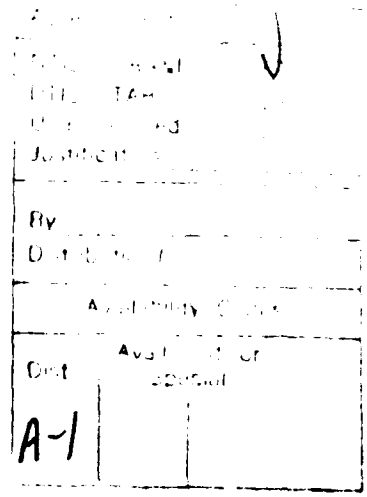
Asymptotic code vector density in topographic vector quantisers

Stephen P Luttrell
Pattern Processing Principles Section
SP4 Division, RSRE
St Andrews Rd, Malvern, WORCS, WR14 3PS

24th May 1990

Abstract

In this memorandum we use a noise-robust vector quantiser model to derive expressions for the asymptotic code vector density ρ in various types of topographic vector quantisers. A topographic vector quantiser is not identical to a standard (ie Kohonen) topographic mapping, but the differences are minimal. In all the cases that we study (scalar and vector quantisation with various symmetric topographic neighbourhoods) we obtain the asymptotic result $\rho \propto P^{\frac{N}{N+2}}$, where N is the input dimensionality and P is the input probability density. Thus the asymptotic code vector densities of a topographic vector quantiser and a standard vector quantiser are the same.



Contents

List of Figures

List of Tables

1 Introduction

There is much literature on vector quantisation (VQ) and scalar quantisation theory [1, 2, 3], where an encoding/decoding scheme is optimised in such a way as to minimise a distortion measure (or Lyapunov function).

There is also an interesting class of transformations (called topographic mappings (TM) in the neural network literature [4]) that can be trained to perform mappings of high dimensional input vectors into low dimensional output vectors. In recent work [5, 6] we showed how to reformulate the problem of training a TM as a problem of minimising a distortion measure: this required a slight modification of the original training algorithm, but the side effects of this were minimal. We call this type of mapping a topographic vector quantiser (TVQ) in order to emphasise its close relationship to a plain VQ, and to distinguish it from the standard TM method. A TVQ has the important property that the codes that they produce are robust with respect to the damaging effects of code noise corresponding to the topographic neighbourhood used during training (ie minimisation of the distortion measure).

The question of the asymptotic properties of TVQs (versus those of VQs) naturally arises. In a recent study [7] the asymptotic code vector (CV) density in a scalar TM (1 dimension mapped to 1 dimension) was derived, and found to be proportional to $P(x)^{\alpha}$ where $\alpha = (2n+1)^2/3\,((n+1)^2 - n^2)$, $P(x)$ is the probability density of input scalars, and n is the half-width of the update neighbourhood used when training the TM. The $n = 0$ case reduces to $\alpha = 1/3$, which is consistent with the result expected from a scalar quantiser.

We wish to derive the corresponding TVQ result for the case where we solve a minimum L_2 distortion problem (as formulated in [5, 6]), rather than the standard TM problem (as formulated in [4]).

In §2 we shall present the TVQ (as opposed to the TM) version of the derivation that appeared in [7]. For convenience, we present a summary of [7] in appendix A [1]. In §3 we shall extend this result to the more general case of a symmetric (and monotonically decreasing to zero) neighbourhood. In §4 we extend these results to the full vectorial case by introducing a simple VQ model that can be used to derive the asymptotic CV density, and in §5 we shall extend this model to the TVQ case. In all the cases that we study we obtain the same result for α (in the TVQ case) that we would have obtained in the corresponding VQ case. In general $\alpha = N/(N+2)$, where N is the dimensionality of the input vector.

2 Density in 1 dimension

In this section we shall derive the asymptotic CV density for a topographic scalar quantiser, which maps a 1 dimensional input scalar to one of a set of CVs (or, strictly speaking, code scalars). We shall formulate an L_2 (or Euclidean) distortion to take account of the same

[1] We also correct a number of serious typographical errors that appeared in [7].

update neighbourhood that was used in [7] in order that we may obtain comparable results. For completeness, in appendix A we summarise the derivation in [7].

2.1 Euclidean distortion

Introduce an L_2 distortion measure D_1 as

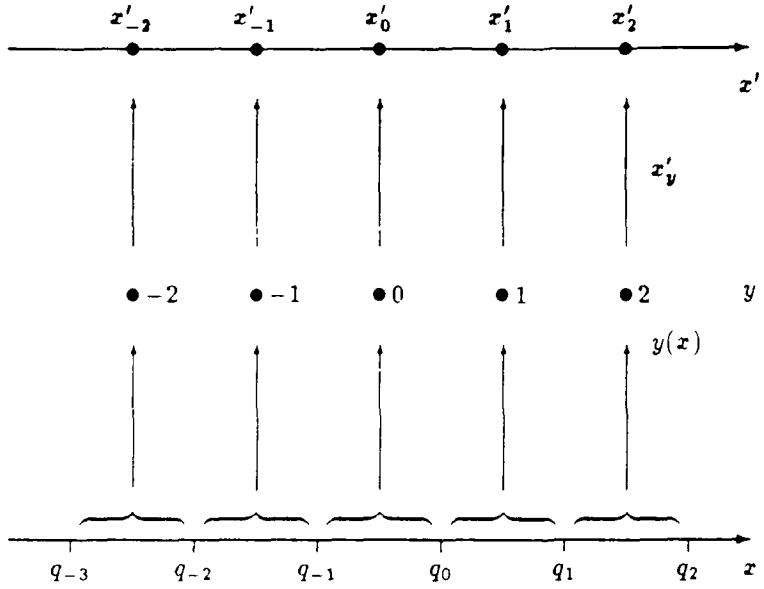


Figure 1: Encoding and decoding

$$\begin{aligned} D_1 &= \int dx\, P(x) \left(x - x'_{y(x)}\right)^2 \\ &= \sum_y \int_{q_{y-1}}^{q_y} dx\, P(x) \left(x - x'_y\right)^2 \end{aligned} \tag{1}$$

In figure 1 we show as a network the various steps that are involved in calculating D_1: the figure reads from the bottom to the top. The input x is a scalar which we represent by the horizontal axis at the bottom of figure 1. $y(x)$ is an encoding function that maps from the input x to an index y. The interval $[q_{y-1}, q_y]$ is defined as follows

$$[q_{y-1}, q_y] \equiv \{x : y = y(x)\} \tag{2}$$

which we use in equation (1) to partition the range of integration into a set of convenient intervals. The output x'_y is the CV (or decoding function) associated with code index y, and it sits on the horizontal axis x'_y that we have drawn at the top of figure 1. The set of x'_y (ranging over all vales of the index y) comprises the codebook that is used in this encoding/decoding operation. The overall goal is to choose the encoding $y(x)$ and decoding x'_y functions in such a way as to minimise the mean L_2 distortion D_1 between input x and output $x'_{y(x)}$

2.2 Minimum Euclidean distortion

In order to minimise D_1 we must differentiate it with respect to the various free parameters: in this case the q_y (which parameterise the encoding function) and the x'_y (which parameterise the decoding function $y(x)$, see equation (2)). Thus we obtain the partial derivatives as

$$\begin{aligned}\frac{\partial D_1}{\partial q_y} &= P(q_y)\left[\left(q_y - x'_y\right)^2 - \left(q_y - x'_{y+1}\right)^2\right] \\ &= 2P(q_y)\left(x'_{y+1} - x'_y\right)\left(q_y - \frac{x'_y + x'_{y+1}}{2}\right) \qquad (3)\end{aligned}$$

$$\frac{\partial D_1}{\partial x'_y} = -2\int_{q_{y-1}}^{q_y} dx\, P(x)(x - x'_y) \qquad (4)$$

whence the stationary points of D_1 must satisfy

$$q_y = \frac{1}{2}\left(x'_y + x'_{y+1}\right) \qquad (5)$$

$$x'_y = \frac{\int_{q_{y-1}}^{q_y} dx\, P(x)\, x}{\int_{q_{y-1}}^{q_y} dx\, P(x)} \qquad (6)$$

Note that equation (5) requires that q_y lies midway between the adjacent CVs, so it defines a *nearest neighbour* encoding function $y(x)$.

2.3 Robust Euclidean distortion

Now we shall generalise the L_2 distortion measure of equation (1) to include a neighbourhood function $\pi_{y',y}$ that specifies the extent to which y' is in the neighbourhood of y (later on we shall define this notion more precisely). Thus introduce the L_2 distortion measure D_2 as

$$\begin{aligned}D_2 &= \int dx\, P(x) \sum_{y'} \pi_{y',y(x)} \left(x - x'_{y'}\right)^2 \\ &= \sum_y \int_{q_{y-1}}^{q_y} dx\, P(x) \sum_{y'} \pi_{y',y} \left(x - x'_{y'}\right)^2 \qquad (7)\end{aligned}$$

The $y(x)$ (and hence the q_y) and the $x'_{y'}$ used in D_2 are to be understood to be different from those used in D_1. In figure 2 we show how a modified version of figure 1 in which the effect of the $\pi_{y',y}$ is represented (for simplicity, we show only $\pi_{y\pm1,y}$). The action of the $\pi_{y',y}$ is interposed between the action of $y(x)$ and the action of $x'_{y'}$, and it can be interpreted as the relative probability with which index y is corrupted by some distortion process to become index y'. With this interpretation in mind, it is evident that minimising D_2 with respect to the choice of $y(x)$ and x'_y will cause the encoding/decoding process to become robust with respect to the damaging effects of the distortion process modelled by $\pi_{y',y}$ [5, 6].

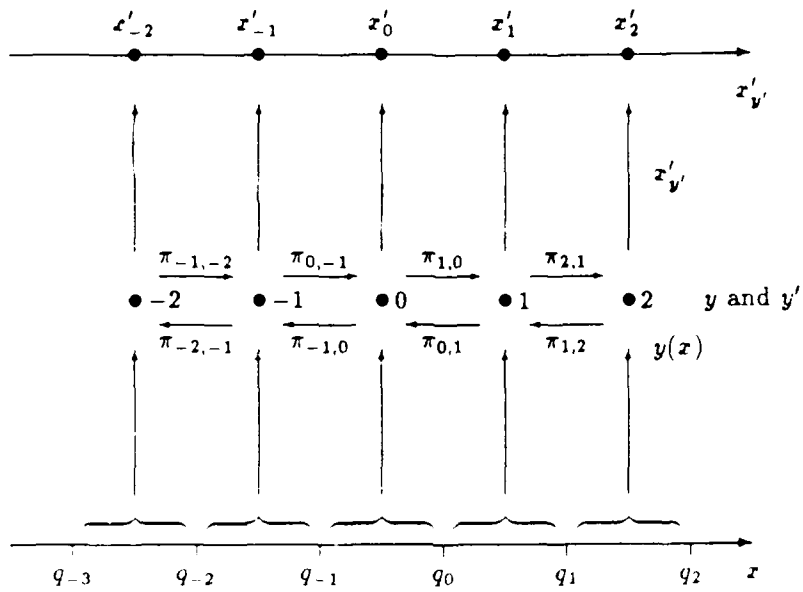


Figure 2: Encoding and decoding in the presence of code distortion

2.4 Minimum robust Euclidean distortion

We may now repeat the derivation of §§2.2 for D_2 (rather than D_1). The partial derivatives are

$$\frac{\partial D_2}{\partial q_y} = P(q_y)\left[\sum_{y'} \pi_{y',y}\left(q_y - x'_{y'}\right)^2 - \sum_{y'} \pi_{y',y+1}\left(q_y - x'_{y'}\right)^2\right] \tag{8}$$

$$\frac{\partial D_2}{\partial x'_y} = -2\sum_{y'} \int_{q_{y'-1}}^{q_{y'}} dx\, P(x)\,\pi_{y,y'}\,(x - x'_y) \tag{9}$$

so the stationary points of D_2 must satisfy

$$\sum_{y'} \pi_{y',y}\left(q_y - x'_{y'}\right)^2 = \sum_{y'} \pi_{y',y+1}\left(q_y - x'_{y'}\right)^2 \tag{10}$$

$$x'_y = \frac{\sum_{y'} \int_{q_{y'-1}}^{q_{y'}} dx\, P(x)\,\pi_{y,y'}\,x}{\sum_{y'} \int_{q_{y'-1}}^{q_{y'}} dx\, P(x)\,\pi_{y,y'}} \tag{11}$$

In equation (10) note that the effect of $\pi_{y',y}$ is to destroy the nearest neighbour encoding property that we obtained in equation (5).

2.5 Finite differences and derivatives

In this subsection we collect together various useful results that we need in order to derive the asymptotic CV density. In order to make direct contact with the results that were

reported in [7] we shall now assume a specific form for $\pi_{y',y}$

$$\pi_{y',y} = \begin{cases} 1 & \text{if } |y' - y| \leq n \\ 0 & \text{if } |y' - y| > n \end{cases} \tag{12}$$

This $\pi_{y',y}$ defines a uniform neighbourhood that ranges from $y - n$ to $y + n$ in the neighbourhood of code index y, which we shall call a $[-n, +n]$ neighbourhood.

With this assumption we can solve equation (10) to yield

$$q_y = \frac{1}{2}\left(x'_{y-n} + x'_{y+n+1}\right) \tag{13}$$

which should be compared with the result in equation (5) (which corresponds to making the choice $\pi_{y',y} = \delta_{y',y}$, or $n = 0$, in equation (13)). The effect of the $[-n, +n]$ neighbourhood is to replace the midpoint of the interval $[x'_y, x'_{y+1}]$ by the midpoint of the larger interval $[x'_{y-n}, x'_{y+n+1}]$[2].

Now introduce a pair of expressions to relate the *finite differences* of x'_y to the *derivatives* dx'_y/dy and $d^2x'_y/dy^2$ of x'_y

$$\begin{aligned} x'_{y+k} - x'_{y-k} &= 2k\frac{dx'_y}{dy} + \mathcal{O}\left(k^3\frac{d^3x'_y}{dy^3}\right) \\ x'_{y+k} + x'_{y-k} - 2x'_y &= k^2\frac{d^2x'_y}{dy^2} + \mathcal{O}\left(k^4\frac{d^4x'_y}{dy^4}\right) \end{aligned} \tag{14}$$

These two expressions can easily be obtained by Taylor expanding (about the point where the code index has value y) the various terms on the left hand sides of equation (14).

Using equation (13) and equation (14) we may derive the midpoint u_y and the half-length a_y of the interval $[q_{y-n-1}, q_{y+n}]$ as

$$\begin{aligned} u_y &\equiv \frac{1}{2}\left(q_{y-n-1} + q_{y+n}\right) \\ &= \frac{1}{4}\left(x'_{y-2n-1} + x'_{y+2n+1} + 2x'_y\right) \\ &= x'_y + \frac{(2n+1)^2}{4}\frac{d^2x'_y}{dy^2} + \mathcal{O}\left(n^4\frac{d^4x'_y}{dy^4}\right) \end{aligned} \tag{15}$$

$$\begin{aligned} a_y &\equiv \frac{1}{2}\left(q_{y+n} - q_{y-n-1}\right) \\ &= \frac{1}{4}\left(x'_{y+2n+1} - x'_{y-2n-1}\right) \\ &= \frac{2n+1}{2}\frac{dx'_y}{dy} + \mathcal{O}\left(n^3\frac{d^3x'_y}{dy^3}\right) \end{aligned} \tag{16}$$

We now wish to transform our results into the language of density of CVs $\rho(x'_y)$. We can relate $\rho(x'_y)$ to quantities that we have already introduced, as follows

$$\rho(x'_y) = \frac{dy}{dx'_y} \tag{17}$$

[2] Note that $x'_y \leq q_y \leq x'_{y+1}$ is not necessarily true when $n > 0$.

Thus $\rho(x'_y)$ is the number of CV indices y per unit change in CV position x'_y. In equation (14) we encountered derivatives of x'_y (up to the fourth order, if we include the next to leading terms) which we shall now express in terms of derivatives of $\rho(x'_y)$. Thus

$$\begin{aligned}
\frac{dx'_y}{dy} &= \frac{1}{\rho(x'_y)} \\
\frac{d^2x'_y}{dy^2} &= -\frac{1}{\rho(x'_y)^3}\frac{d\rho(x'_y)}{dy} \\
\frac{d^3x'_y}{dy^3} &= -\frac{1}{\rho(x'_y)^4}\frac{d^2\rho(x'_y)}{dy^2} + \frac{3}{\rho(x'_y)^5}\left(\frac{d\rho(x'_y)}{dy}\right)^2 \\
\frac{d^4x'_y}{dy^4} &= -\frac{1}{\rho(x'_y)^5}\frac{d^3\rho(x'_y)}{dy^3} + \frac{10}{\rho(x'_y)^6}\frac{d^2\rho(x'_y)}{dy^2}\frac{d\rho(x'_y)}{dy} - \frac{15}{\rho(x'_y)^7}\left(\frac{d\rho(x'_y)}{dy}\right)^3
\end{aligned} \tag{18}$$

where we have used $d\ dy = (dx'_y\ dy)d\ dx'_y = \rho(x'_y)^{-1}d\ dx'_y$, to perform the differentiation. We may thus express the results for u_y (equation (15)) and a_y (equation (16)) in terms of derivatives of $\rho(x'_y)$ as follows

$$u_y = x'_y - \frac{(2n-1)^2}{4\rho(x'_y)^3}\frac{d\rho(x'_y)}{dy} + \mathcal{O}\left(\frac{n^4}{\rho(x'_y)^7}\left(\frac{d\rho(x'_y)}{dy}\right)^3\right) \tag{19}$$

$$a_y = \frac{2n-1}{2\rho(x'_y)} + \mathcal{O}\left(\frac{n^3}{\rho(x'_y)^5}\left(\frac{d\rho(x'_y)}{dy}\right)^2\right) \tag{20}$$

2.6 Approximate optimal code vector positions

We now have all the basic theoretical results that are needed to perform a Taylor expansion of equation (11) to relate the derivative of $P(x)$ to the derivative of $\rho(x)$. Insert the $\pi_{y',y}$ (defined in equation (12)), and use the definitions of the midpoint u_y and half-length a_y of the interval (q_{y-n-1}, q_{y-n}) (in equation (15) and equation (16), respectively), in equation (11) to obtain

$$\begin{aligned}
x'_y &= \frac{\int_{-a_y}^{+a_y} dx\ \left(P(u_y) + x\frac{dP(u_y)}{du_y} + \mathcal{O}\left(x^2\frac{d^2P(u_y)}{du_y^2}\right)\right)(u_y + x)}{\int_{-a_y}^{+a_y} dx\ \left(P(u_y) + x\frac{dP(u_y)}{du_y} + \mathcal{O}\left(x^2\frac{d^2P(u_y)}{du_y^2}\right)\right)} \\
&= \frac{2a_yu_yP(u_y) + \frac{2a_y^3}{3}\frac{dP(u_y)}{du_y} + \mathcal{O}\left(u_ya_y^3\frac{d^2P(u_y)}{du_y^2}\right)}{2a_yP(u_y) + \mathcal{O}\left(a_y^3\frac{d^2P(u_y)}{du_y^2}\right)} \\
&= u_y + \frac{a_y^2}{3P(u_y)}\frac{dP(u_y)}{du_y} + \mathcal{O}\left(\frac{u_ya_y^2}{P(u_y)}\frac{d^2P(u_y)}{du_y^2}\right) \\
&= u_y + \frac{a_y^2}{3P(x'_y)}\frac{dP(x'_y)}{dx'_y} + \mathcal{O}\left(\frac{u_ya_y^2}{P(x'_y)}\frac{d^2P(x'_y)}{dx'^2_y}\right)
\end{aligned} \tag{21}$$

In the last stage of this derivation note that the effect of the change $u_y \to x'_y$ appears only in the next to leading order terms.

2.7 Code vector density

Finally, inserting the expressions for u_y and a_y (from equation (19) and equation (20)) into equation (21), we obtain

$$\frac{1}{\rho(x'_y)}\frac{d\rho(x'_y)}{dy} = \frac{1}{3P(x'_y)}\frac{dP(x'_y)}{dy} + \text{ h.o.t.} \tag{22}$$

where h.o.t. denotes *higher order terms*. We may solve this to yield in leading order

$$\rho(x'_y) \propto P(x'_y)^{1/3} \tag{23}$$

This power law dependence is the same as that observed in the equivalent scalar quantiser (which corresponds to a neighbourhood function $\pi_{y',y} = \delta_{y',y}$), but is different from the result that was obtained in [7] for a TM as defined in [4].

3 Density in 1 dimension: symmetric neighbourhood functions

In this section we shall extend the results of §2 to the case where $\pi_{y',y}$ defines a symmetric (monotonically decreasing to zero) neighbourhood function surrounding each code index. Furthermore, we shall restrict our attention to symmetric neighbourhoods.

3.1 Robust Euclidean distortion

We shall now define the distortion matrix $\pi_{y',y}$ (used in the definition of the L_2 distortion D_2 in equation (7)) in such a way that it satisfies

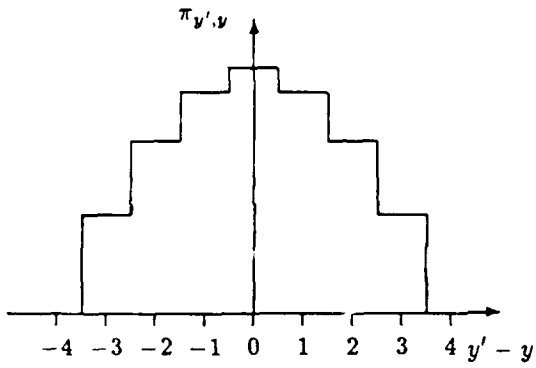
$$\begin{array}{lcll} \pi_{y'+1,y+1} & = & \pi_{y',y} & \text{(Toeplitz matrix)} \\ \pi_{y',y} & = & \pi_{y,y'} & \text{(symmetric matrix)} \end{array} \tag{24}$$

For such matrices it is sufficient to specify the form of a single row (or column) of $\pi_{y',y}$ as a symmetric function of $y' - y$. This type of matrix specifies a distortion that treats each code index y on an equal footing (the Toeplitz property), and it implies a symmetric topographic neighbourhood (the symmetric property).

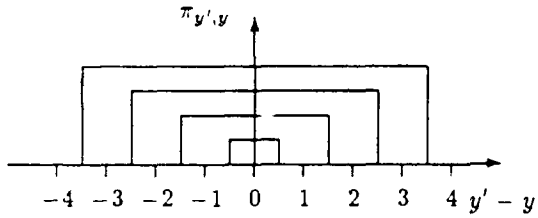
For convenience, and to make contact with the derivation presented in §2, we decompose $\pi_{y',y}$ as a weighted sum over (symmetric) neighbourhood functions of the type defined in equation (12)

$$\pi_{y',y} = \sum_{s:|y'-y|\leq n_s} h_s \tag{25}$$

where we constrain $h_s > 0$ to ensure that $\pi_{y',y}$ is a monotonically decreasing function of $y' - y$. Note that this monotonicity constraint is *in addition to* the properties that we specified in equation (24): we impose it to cure some stability problems that can arise when training TVQs. In figure 3 we give an example of the type of composite neighbourhood



(a) Net neighbourhood function



(b) Decomposed neighbourhood function

Figure 3: Symmetric composite neighbourhood function

function that is described by this model. We show in figure 3(a) a typical $\pi_{y',y}$ neighbourhood function, and we show in figure 3(b) its decomposition as a sum over neighbourhood functions spanning the intervals $[-n_s, +n_s]$ for various s.

Using the definition of $\pi_{y',y}$ in equation (25) we can simplify D_2 equation (7) to become D_3 given by

$$D_3 = \sum_y \int_{q_{y-1}}^{q_y} dx\, P(x) \sum_s h_s \sum_{y'=-n_s}^{+n_s} \left(x - x'_{y+y'}\right)^2 \tag{26}$$

3.2 Minimum robust Euclidean distortion

Now differentiate D_3 to obtain $\partial D_3/\partial x'_y$ and $\partial D_3/\partial q_y$. The stationary points of D_3 must then satisfy

$$q_y = \frac{1}{2} \frac{\sum_s h_s \left(x'_{y+n_s+1} - x'_{y-n_s}\right)\left(x'_{y+n_s+1} + x'_{y-n_s}\right)}{\sum_s h_s \left(x'_{y+n_s+1} - x'_{y-n_s}\right)} \tag{27}$$

$$x'_y = \frac{\sum_s h_s \int_{q_{y-n_s-1}}^{q_{y+n_s}} dx\, P(x)\, x}{\sum_s h_s \int_{q_{y-n_s-1}}^{q_{y+n_s}} dx\, P(x)} \tag{28}$$

Equation (28) replaces equation (11) and equation (27) replaces equation (13).

Note that the positivity of the h_s in equation (25) guarantees the stability of the solution for the x'_y and the q_y in equation (28) and equation (27), because the denominators are strictly positive [3].

Unfortunately, the expression for q_y in equation (27) is sufficiently complicated that we have to perform a large amount of algebra to derive the asymptotic relationship between $P(x)$ and $\rho(x)$ (ie the generalisation of equation (23)).

3.3 Finite differences and derivatives

Firstly, express $x'_{y+k} \pm x'_{y+l}$ in terms of dx'_y/dy and $d^2x'_y/dy^2$.

$$\begin{aligned} & & \tfrac{1}{2}\left(x'_{y+k} + x'_{y-k} - 2x'_y\right) & + & \tfrac{1}{2}\left(x'_{y+k} - x'_{y-k}\right) \\ x'_{y+k} \pm x'_{y-l} &= \pm & \tfrac{1}{2}\left(x'_{y+l} + x'_{y-l} - 2x'_y\right) & \pm & \tfrac{1}{2}\left(x'_{y+l} - x'_{y-l}\right) \\ & + & x'_y \pm x'_y & & \\ & \simeq & \left(\frac{k^2 \pm l^2}{2}\right)\frac{d^2x'_y}{dy^2} + (k \pm l)\frac{dx'_y}{dy} + \left(x'_y \pm x'_y\right) & & \end{aligned} \tag{29}$$

where we have used the finite difference expressions in equation (14). Note that we use $\pm$ signs consistently throughout equation (29), such that if one were to choose the upper sign in one part of the equation then one *must* choose the upper sign throughout the rest of the equation (a similar remark applies to the lower sign). We may use these results to simplify q_{y+n_s} and q_{y-n_s-1} to obtain

$$\begin{aligned} q_{y+n_s} &= \frac{1}{2}\frac{\sum_t h_t\,(\xi_1(t) + \xi_2(s,t))\,(\eta_0 + \eta_1(s) + \eta_2(s,t))}{\sum_t h_t\,(\xi_1(t) + \xi_2(s,t))} \\ q_{y-n_s-1} &= \frac{1}{2}\frac{\sum_t h_t\,(\xi_1(t) - \xi_2(s,t))\,(\eta_0 - \eta_1(s) + \eta_2(s,t))}{\sum_t h_t\,(\xi_1(t) - \xi_2(s,t))} \end{aligned} \tag{30}$$

where we have defined $\xi_1(t)$, $\xi_2(s,t)$, η_0, $\eta_1(s)$ and $\eta_2(s,t)$ as

$$\begin{aligned} \xi_1(t) &\equiv (2n_t + 1)\frac{dx'_y}{dy} \\ \xi_2(s,t) &\equiv \frac{1}{2}\left[(n_s + n_t + 1)^2 - (n_s - n_t)^2\right]\frac{d^2x'_y}{dy^2} \\ \eta_0 &\equiv 2x'_y \\ \eta_1(s) &\equiv (2n_s + 1)\frac{dx'_y}{dy} \\ \eta_2(s,t) &\equiv \frac{1}{2}\left[(n_s + n_t + 1)^2 + (n_s - n_t)^2\right]\frac{d^2x'_y}{dy^2} \end{aligned} \tag{31}$$

[3] We assume that the input probability density $P(x)$ is well-behaved, in the sense that each code index y is indeed associated with a finite probability mass.

Note that $\xi_1(t)$ and $\eta_1(s)$ are trivially related to each other.

We may now introduce the midpoint u_y^s and half-length a_y^s of the interval $[q_{y-n_s-1}, q_{y+n_s}]$, and use equation (30) to simplify their expressions. For compactness, we gather these two results into a single vector equation (where u_y^s is the upper element and a_y^s is the lower element in a two-component column vector)

$$\begin{aligned}\begin{pmatrix} u_y^s \\ a_y^s \end{pmatrix} &\equiv \frac{1}{2}\begin{pmatrix} q_{y+n_s} + q_{y-n_s-1} \\ q_{y+n_s} - q_{y-n_s-1} \end{pmatrix} \\ &= \frac{\frac{1}{2}\sum_{t,t'} h_t h_{t'} \left[\begin{array}{l} (\eta_0 + \eta_2(s,t)) \begin{pmatrix} \xi_1(t)\xi_1(t') - \xi_2(s,t)\xi_2(s,t') \\ \xi_1(t')\xi_2(s,t) - \xi_1(t)\xi_2(s,t') \end{pmatrix} \\ + \ \eta_1(s) \begin{pmatrix} \xi_1(t')\xi_2(s,t) - \xi_1(t)\xi_2(s,t') \\ \xi_1(t)\xi_1(t') - \xi_2(s,t)\xi_2(s,t') \end{pmatrix} \end{array} \right]}{\sum_{t,t'} h_t h_{t'} \left(\xi_1(t)\xi_1(t') - \xi_2(s,t)\xi_2(s,t')\right)} \end{aligned} \tag{32}$$

We have made use of the fact that $\sum_{t,t'} h_t h_{t'} \left(\xi_1(t')\xi_2(s,t) - \xi_1(t)\xi_2(s,t')\right) = 0$ (by symmetry) to simplfy the denominator. Now introduce some approximations which are valid in the leading order of the expansion in terms of derivatives of x'_y with respect to y.

$$\begin{aligned} \xi_2(s,t)\xi_2(s,t') &\simeq 0 \\ \eta_2(s,t)\ \left(\xi_1(t')\xi_2(s,t) - \xi_1(t)\xi_2(s,t')\right) &\simeq 0 \end{aligned} \tag{33}$$

When we insert these approximations into equation (32), and we make use of the relationships in equation (18), we obtain

$$\begin{aligned} u_y^s &\simeq \frac{1}{2}\frac{\sum_{t,t'} h_t h_{t'} \xi_1(t)\xi_1(t')\ (\eta_0 + \eta_2(s,t))}{\sum_{t,t'} h_t h_{t'} \xi_1(t)\xi_1(t')} \\ &= x'_i + \left[\frac{1}{4}\frac{\sum_t h_t\ (2n_t+1)\left[(n_s+n_t+1)^2 + (n_s-n_t)^2\right]}{\sum_t h_t\ (2n_t+1)}\right]\frac{d^2x'_y}{dy^2} \\ &\simeq x'_y - \left[\frac{(2n_s+1)^2}{4} + \frac{1}{2}\frac{\sum_t h_t\ (2n_t+1)\,(n_t-n_s)\,(n_s+n_t+1)}{\sum_t h_t\ (2n_t+1)}\right]\frac{1}{\rho(x'_y)^3}\frac{d\rho(x'_y)}{dx'_y} \end{aligned} \tag{34}$$

$$\begin{aligned} a_y^s &\simeq \frac{1}{2}\frac{\sum_{t,t'} h_t h_{t'} \xi_1(t)\xi_1(t')\,\eta_1(s)}{\sum_{t,t'} h_t h_{t'} \xi_1(t)\xi_1(t')} \\ &\simeq \frac{2n_s+1}{2\rho(x'_y)} \end{aligned} \tag{35}$$

We have expressed these results in such a way that they may readily be compared with the analogous results in equation (19) and equation (20).

When comparing equation (34) with equation (19) note that there is a leading order correction term caused by the presence of more than one component in the composite neighbourhood function, but note that equation (20) needs no such leading order correction to become equation (35).

3.4 Approximate optimal code vector positions

We shall now form a Taylor expansion of the integrands in the numerator and denominator of equation (28). The steps in the derivation are similar to those used to derive the result

in equation (21) so we shall not repeat them. The final result is

$$x'_y \simeq \frac{\left\langle a^s_y u^s_y \right\rangle}{\left\langle a^s_y \right\rangle} + \frac{\left\langle \left(a^s_y\right)^3 \right\rangle}{3 \left\langle a^s_y \right\rangle P(x'_y)} \frac{dP(x'_y)}{dx'_y} \tag{36}$$

where our angle bracket notation represents a average weighted by the h_s, which is defined as

$$\langle \cdots \rangle \equiv \frac{\sum_s h_s (\cdots)}{\sum_s h_s} \tag{37}$$

The ratios of weighted averages in equation (36) may be evaluated by taking appropriate averages of the results in equation (34) and equation (35), to yield

$$\begin{aligned} \frac{\left\langle a^s_y u^s_y \right\rangle}{\left\langle a^s_y \right\rangle} &= x'_y - \frac{\left\langle (2n_s + 1)^3 \right\rangle}{4 \langle 2n_s + 1 \rangle \rho(x'_y)^3} \frac{d\rho(x'_y)}{dx'_y} \\ \frac{\left\langle \left(a^s_y\right)^3 \right\rangle}{\left\langle a^s_y \right\rangle} &= \frac{\left\langle (2n_s + 1)^3 \right\rangle}{4 \langle 2n_s + 1 \rangle \rho(x'_y)^2} \end{aligned} \tag{38}$$

where we note that the contribution of the correction term in equation (34) disappears (by symmetry).

3.5 Code vector density

Finally, inserting the results of equation (38) into the leading order Taylor expansion in equation (36) yields (in leading order) the same differential equation that we obtained in equation (22). Thus we have shown that for the class of $\pi_{y',y}$ corresponding to symmetric monotonically decreasing neighbourhood functions, and retaining only leading order terms, the CV density is given by $\rho(x'_y) \propto P(x'_y)^{1/3}$ (ie the same as equation (23)).

4 Density in N dimensions: standard vector quantiser

In this section we shall present a simple derivation of the CV density for a standard VQ. We believe this to be a novel way of deriving this result, and it serves to underpin the TVQ case that we discuss in §5.

Unfortunately, the derivation that we present is based upon a qualitative model of the distortions that occur during encoding/decoding, so the results that we obtain remain open to some criticism [4].

4.1 Vector quantiser model

We present our VQ model in figure 4 where we use a fully vectorised form of the notation that

[4] We would welcome comments and suggestions on how one might improve on this derivation.

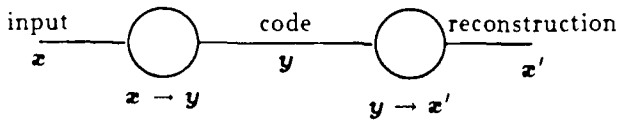


Figure 4: Standard vector quantiser

we used in earlier sections of this paper. The input vector $\boldsymbol{x}$ is encoded (by an encoding function $\boldsymbol{y}(\boldsymbol{x})$) to produce an index $\boldsymbol{y}$, which is then decoded (by a decoding function $\boldsymbol{x}'(\boldsymbol{y})$) to produce a reconstructed vector $\boldsymbol{x}'$. The goal is to choose $\boldsymbol{y}(\boldsymbol{x})$ and $\boldsymbol{x}'(\boldsymbol{y})$ so as to minimise the average of an appropriately chosen distortion measure between the original $\boldsymbol{x}$ and reconstructed $\boldsymbol{x}'$ versions of the input vector.

4.2 Euclidean distortion

Introduce an L_2 distortion measure D_4 as

$$D_4 = \int d\boldsymbol{x}\, P(\boldsymbol{x}) \, | \boldsymbol{x} - \boldsymbol{x}'(\boldsymbol{y}(\boldsymbol{x})) |^2 \tag{39}$$

where $P(\boldsymbol{x})$ is the probability density over possible inputs, and the notation $|\cdots|$ denotes the *norm* of the enclosed vector.

In preparation for our reformulation of D_4 in terms of a CV density we shall reexpress D_4 as an integral over $\boldsymbol{y}$ by using the identity

$$\int d\boldsymbol{y}\, \delta(\boldsymbol{y} - \boldsymbol{y}(\boldsymbol{x})) = 1 \tag{40}$$

where $\delta(\cdots)$ is the Dirac delta function. Thus we may rewrite equation (39) as follows

$$\begin{aligned}
D_4 &= \int d\boldsymbol{y} \int d\boldsymbol{x}\, P(\boldsymbol{x})\, \delta(\boldsymbol{y} - \boldsymbol{y}(\boldsymbol{x}))\, \|\boldsymbol{x} - \boldsymbol{x}'(\boldsymbol{y}(\boldsymbol{x}))\|^2 \\
&= \int d\boldsymbol{y} \int d\boldsymbol{x}\, P(\boldsymbol{x})\, \delta(\boldsymbol{y} - \boldsymbol{y}(\boldsymbol{x}))\, \|\boldsymbol{x} - \boldsymbol{x}'(\boldsymbol{y})\|^2 \\
&= \int d\boldsymbol{y}\, P(\boldsymbol{y}) \left[\left\langle \|\boldsymbol{x}\|^2 \right\rangle_{\boldsymbol{x}|\boldsymbol{y}} - 2\, \langle \boldsymbol{x} \rangle_{\boldsymbol{x}|\boldsymbol{y}} \cdot \boldsymbol{x}'(\boldsymbol{y}) + \|\boldsymbol{x}'(\boldsymbol{y})\|^2 \right] \\
&= \int d\boldsymbol{y}\, P(\boldsymbol{y}) \left[\left\| \langle \boldsymbol{x} \rangle_{\boldsymbol{x}|\boldsymbol{y}} - \boldsymbol{x}'(\boldsymbol{y}) \right\|^2 + \left\langle \left\| \boldsymbol{x} - \langle \boldsymbol{x} \rangle_{\boldsymbol{x}|\boldsymbol{y}} \right\|^2 \right\rangle_{\boldsymbol{x}|\boldsymbol{y}} \right] \\
&\xrightarrow{\boldsymbol{x}'(\boldsymbol{y}) = \langle \boldsymbol{x} \rangle_{\boldsymbol{x}|\boldsymbol{y}}} \int d\boldsymbol{y}\, P(\boldsymbol{y}) \left\langle \left\| \boldsymbol{x} - \langle \boldsymbol{x} \rangle_{\boldsymbol{x}|\boldsymbol{y}} \right\|^2 \right\rangle_{\boldsymbol{x}|\boldsymbol{y}}
\end{aligned} \tag{41}$$

where (using a rather careless notation) $P(\boldsymbol{y}) = \int d\boldsymbol{x}\, P(\boldsymbol{x})\, \delta(\boldsymbol{y} - \boldsymbol{y}(\boldsymbol{x}))$ is the probability density over possible codes [5], and the angle brackets $\langle \ldots \rangle_{\boldsymbol{x}|\boldsymbol{y}}$ denote an average over the

[5] Strictly, we should use a different notation, say $P_{\boldsymbol{x}}(\boldsymbol{x})$ and $P_{\boldsymbol{y}}(\boldsymbol{y})$, for the two probability densities, because they are *different* functions of their arguments. However, for our purposes, it is always possible to deduce from the context (ie $\boldsymbol{x}$ or $\boldsymbol{y}$) which probability density is required.

posterior probability density (of $\boldsymbol{x}$ given $\boldsymbol{y}$)—in this case this is simply an average over those $\boldsymbol{x}$ that map to $\boldsymbol{y}$ via $\boldsymbol{y}(\boldsymbol{x})$.

In the final line of equation (41) we replace D_4 by its minimum with respect to $\boldsymbol{x}'(\boldsymbol{y})$: thus we shall henceforth set $\boldsymbol{x}'(\boldsymbol{y}) = \langle \boldsymbol{x} \rangle_{\boldsymbol{x}|\boldsymbol{y}}$. We may interpret equation (41) as follows.

1. $\int d\boldsymbol{y} P(\boldsymbol{y})$ is an average over the possible codes $\boldsymbol{y}$.
2. $\left\langle \left\| \boldsymbol{x} - \langle \boldsymbol{x} \rangle_{\boldsymbol{x}|\boldsymbol{y}} \right\|^2 \right\rangle_{\boldsymbol{x}|\boldsymbol{y}}$ breaks down thus:
   (a) $\langle \boldsymbol{x} \rangle_{\boldsymbol{x}|\boldsymbol{y}}$ is the mean of the set of $\boldsymbol{x}$ that map to $\boldsymbol{y}$. We call this the posterior mean, because the average uses the posterior probability $P(\boldsymbol{x}|\boldsymbol{y})$.
   (b) $\boldsymbol{x} - \langle \boldsymbol{x} \rangle_{\boldsymbol{x}|\boldsymbol{y}}$ is the error vector between the posterior mean and the true input vector $\boldsymbol{x}$.
   (c) Finally, the whole of this expression is the posterior mean of the *norm squared* of the error vector.

Thus equation (41) reduces to the average posterior L_2 distortion.

4.3 Code vector density

Let us rewrite equation (41) by introducing a CV density $\rho(\boldsymbol{x})$ which describes the number of CVs $\boldsymbol{x}'(\boldsymbol{y}(\boldsymbol{x}))$ per unit volume in the input space. There is a distortion volume in input space associated with CV index $\boldsymbol{y}$ that corresponds to the set of $\boldsymbol{x}$ that satisfy $\boldsymbol{y} = \boldsymbol{y}(\boldsymbol{x})$, and the characteristic radius L of this volume is given by

$$L = \rho(\boldsymbol{x})^{-\frac{1}{N}} \tag{42}$$

where we assume that the input space is N-dimensional. Note that we have assumed without proof that a single radius characterises the distortion volume. This is valid because the optimal shape for distortion volumes is spherical, but we shall not let ourselves be sidetracked by such details, since the purpose of this section is to present an alternative strategy for deriving the CV density that occurs in a standard VQ. For a complete derivation of this result (not assuming isotropy) see appendix B

The final result in equation (41) tells us that each CV contributes to the overall distortion D_4 an amount which is the product of its probability of selection times a posterior mean squared error. Dimensional analysis tells us that D_4 may be modelled as

$$D_4 \simeq \int d\boldsymbol{x}\, P(\boldsymbol{x})\, \rho(\boldsymbol{x})^{-\frac{2}{N}} \tag{43}$$

Thus we have replaced the rather complicated expression in equation (41) by a very simple (approximately equivalent) expression formulated in terms of the CV density.

4.4 Optimum code vector density

We shall now minimise D_4 in equation (43) subject to the condition that the total number C of CVs is held constant. C is given by

$$C = \int d\boldsymbol{x}\, P(\boldsymbol{x})\, \rho(\boldsymbol{x}) \tag{44}$$

so we must minimise the composite distortion D_4' given by

$$D_4' = D_4 + \lambda C \tag{45}$$

where λ is a Lagrange multiplier, whose value will be chosen later to guarantee that C takes the correct value.

Functionally differentiating D_4' with respect to $\rho(\boldsymbol{x})$ yields

$$\frac{\delta D_4'}{\delta \rho(\boldsymbol{x})} = -\frac{2}{N} P(\boldsymbol{x})\, \rho(\boldsymbol{x})^{-\frac{N+2}{N}} + \lambda \tag{46}$$

Thus $\delta D_4' / \delta \rho(\boldsymbol{x}) = 0$ when

$$\rho(\boldsymbol{x}) \propto P(\boldsymbol{x})^{\frac{N}{N+2}} \tag{47}$$

This is the required CV density, which correctly reduces to $\rho(\boldsymbol{x}) \propto P(\boldsymbol{x})^{1/3}$ in the 1-dimensional case, as expected. This result may easily be generalised to the case of an L_r distortion measure to yield $\rho(\boldsymbol{x}) \propto P(\boldsymbol{x})^{\frac{N}{N+r}}$.

5 Density in N dimensions: topographic vector quantiser

In this section we shall generalise the result of §4 to take account of a non-zero topographic neighbourhood, thus transforming a standard VQ into a TVQ.

5.1 Topographic vector quantiser model

We present our TVQ model in figure 5 which is basically the same as figure 4 except that

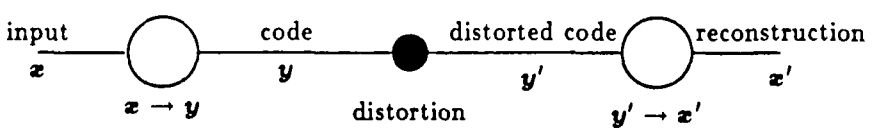


Figure 5: Topographic vector quantiser

we introduce a distortion process that acts to corrupt the code $\boldsymbol{y}$ to produce a distorted code $\boldsymbol{y}'$. We choose our notation carefully to be a vectorised form of the notation that we used in earlier sections of this paper.

5.2 Robust Euclidean distortion

Equation (39) becomes

$$D_5 = \int d\boldsymbol{x}\, P(\boldsymbol{x}) \int d\boldsymbol{n}\, \pi(\boldsymbol{n})\, \|\boldsymbol{x} - \boldsymbol{x}'(\boldsymbol{y}(\boldsymbol{x}) + \boldsymbol{n})\|^2 \tag{48}$$

where we now average over all inputs $\boldsymbol{x}$ and distortions $\boldsymbol{n}$. $\pi(\boldsymbol{n})$ is the probability density over distortions, where we have assumed that $\boldsymbol{n}$ is additive and statistically independent of $\boldsymbol{x}$. If we compare equation (39) with equation (26), we observe that $\pi(\boldsymbol{n})$ describes a multidimensional (in code space) version of the Toeplitz matrix distortion process that we used in the scalar case.

By analogy with equation (41) we can derive

$$\begin{aligned} D_5 &= \int d\boldsymbol{y} \int d\boldsymbol{x}\, P(\boldsymbol{x})\, \pi(\boldsymbol{y} - \boldsymbol{y}(\boldsymbol{x}))\, \|\boldsymbol{x} - \boldsymbol{x}'(\boldsymbol{y})\|^2 \\ &\underset{\boldsymbol{x}'(\boldsymbol{y}) = \langle \boldsymbol{x} \rangle_{\boldsymbol{x}|\boldsymbol{y}}}{\longrightarrow} \int d\boldsymbol{y}\, P(\boldsymbol{y}) \left\langle \|\boldsymbol{x} - \langle \boldsymbol{x} \rangle_{\boldsymbol{x}|\boldsymbol{y}}\|^2 \right\rangle_{\boldsymbol{x}|\boldsymbol{y}} \end{aligned} \tag{49}$$

Note that the $\delta(\boldsymbol{y} - \boldsymbol{y}(\boldsymbol{x}))$ in equation (41) has simply been replaced by $\pi(\boldsymbol{y} - \boldsymbol{y}(\boldsymbol{x}))$ in equation (49). The angle brackets $\langle \ldots \rangle_{\boldsymbol{x}|\boldsymbol{y}}$ still denote an average over the posterior probability density (of $\boldsymbol{x}$ given $\boldsymbol{y}$), which is

$$P(\boldsymbol{x}|\boldsymbol{y}) = \frac{\pi(\boldsymbol{y} - \boldsymbol{y}(\boldsymbol{x}))\, P(\boldsymbol{x})}{P(\boldsymbol{y})} \tag{50}$$

This yields the correct result in the special case that $\pi(\boldsymbol{y} - \boldsymbol{y}(\boldsymbol{x})) \rightarrow \delta(\boldsymbol{y} - \boldsymbol{y}(\boldsymbol{x}))$, as in equation (41).

It is important to note that the CV $\boldsymbol{x}'(\boldsymbol{y})$ that minimises D_5 is still the mean of the posterior probability $\boldsymbol{x}'(\boldsymbol{y}) = \langle \boldsymbol{x} \rangle_{\boldsymbol{x}|\boldsymbol{y}}$. In fact, the only difference between equation (41) and equation (49) is the replacement $\delta(\boldsymbol{y} - \boldsymbol{y}(\boldsymbol{x})) \rightarrow \pi(\boldsymbol{y} - \boldsymbol{y}(\boldsymbol{x}))$.

5.3 Topographic code vector density

The distortion process that is modelled by $\pi(\boldsymbol{n})$ describes a neighbourhood function that corresponds to a Toeplitz matrix (as in equation (24)), so the distortion volume that is associated with the posterior probability $P(\boldsymbol{x}|\boldsymbol{y}')$ is *proportional to* to the distortion volume that is associated with $P(\boldsymbol{x}|\boldsymbol{y})$, with the same constant of proportionality for each CV. We may thus replace equation (43) by

$$D_5 \propto \int d\boldsymbol{x}\, P(\boldsymbol{x})\, \rho(\boldsymbol{x})^{-\frac{2}{N}} \tag{51}$$

Thus the same optimum CV density emerges as we obtained in the standard VQ derivation starting at equation (42).

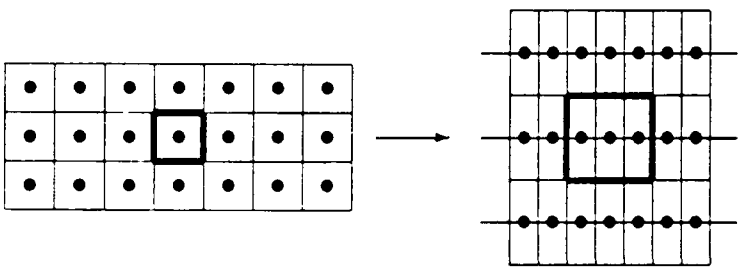
In this sketch derivation of the asymptotic CV density in a TVQ we have *assumed* that the optimum distortion volume (corresponding to $P(\boldsymbol{x}|\boldsymbol{y}')$) is characterised by a single

length. The full proof of the TVQ case follows a similar pattern to that found in appendix B for the VQ case [6].

5.4 Intuitive interpretation of the equivalence between topographic and plain vector quantisers

We have presented a lot of mathematics in our various derivations of asymptotic CV densities. In all cases the result has been unmodified by neighbourhood functions provided that we consistently take their effect into account during the encoding/decoding processes (ie use minimum distortion encoding rather than nearest neighbour encoding).

We wish to present an intuitive picture of the processes at work that conspire to make the standard VQ and the TVQ equivalent, in the sense of code vector densities. In figure 6



(a) Plain vector quantiser (b) Topographic vector quantiser

Figure 6: Equivalence of standard and topographic vector quantisers

we show how the equivalence emerges when the input $\boldsymbol{x}$ is 2-dimensional and the code $\boldsymbol{y}$ has a 1-dimensional topology (which can be turned on and off at will). For the purposes of this argument we shall assume that $P(\boldsymbol{x})$ is uniform. In figure 6(a) we idealise the code vectors (and their cells) of a VQ as sitting on a square lattice[7]. In figure 6(b) we show how these same code vectors (and their cells) would be modified when a 1-dimensional neighbourhood function is introduced. In our example we use a neighbourhood that ranges over the $[-1,+1]$ neighbourhood about each code vector index.

The net effect of the neighbourhood function is to attract neighbouring code vectors together, and simultaneously to repel more distantly separated pieces of the 1-dimensional "line" of code vectors. The overall effect is to preserve the density of code vectors that would have been found in a standard VQ (zero neighbourhood size), and to ensure that the posterior covariance $P(\boldsymbol{x}|\boldsymbol{y}')$ is isotropic. In our example there is a compression by a factor of $\sqrt{3}$ in the horizontal direction, and a expansion in the vertical direction by a factor of $\sqrt{3}$, thus guaranteeing that the area associated with each code vector remains

[6] We leave the details as an exercise for the reader! Although, before attempting to derive the result it should prove useful to read the intuitive arguments in §§5.4.

[7] In fact, this arrangement does not minimise D_6, but it will serve to illustrate our argument.

unchanged (ie density is unchanged). Furthermore, $P(\boldsymbol{x}|\boldsymbol{y}')$ is represented by the bold square superimposed on each diagram: in both cases it is isotropic [8]. Note how the posterior covariance includes 3 quantisation cells in the TVQ, due to the $[-1,+1]$ neighbourhood function.

This intuitive picture is very convenient, because we can imagine that the sole effect of introducing a topology in the code book is to attract neighbouring code vectors whilst effectively repelling more distant code vectors (due to conservation of the total number of code vectors). In the case of an d-dimensional neighbourhood function, the code vectors would attract themselves into d-dimensional sheets embedded in the input space. This sheet would repeatedly fold over on itself (like puff pastry) to fill the input space. However, the sheet would not collaps onto itself because there is an effective repulsion between different folds of the sheet. Overall, the net density of code vectors is the same as would have been encountered in a standard VQ.

Remember that nearest neighbour encoding will *not* lead to this convenient result: you have to use minimum distortion encoding which takes into account the effect of the process that distorts the code indices.

6 Numerical simulation

In this section we shall present the results of a simple numerical simulation that verifies our theoretical prediction $\rho(x) \propto P(x)^{1/3}$ in the 1-dimensional case.

6.1 Numerical experimental procedure

We shall now describe the numerical experiment that was performed in [7].

1. Define a finite support for x: $x \in [0,1]$.
2. Define a probability density $P(x)$ of input scalars: $P(x) = 2x$ (ie a ramp).
3. Choose the number n_{cv} of CVs (in this case, code scalars) that you wish to use. In [7] $n_{cv} = 100$ was used, but we shall use $n_{cv} = 30$ because we find that the results still approximate those that we would expect in the asymptotic (ie $n_{cv} \to \infty$) case.
4. Choose the number n that determines the size of the $[-n,+n]$ topographic neighbourhood, in the form given in equation (12).
5. Adapt the positions of the CVs using the standard training scheme for TMs [4]. We use a simplified version of the method in [7] where we use an update step size $\epsilon = 0.1$ (in equation (54)), and we train for 500000 presentations of x samples chosen randomly from $P(x)$ [9]. Combining the ϵ and $\pi_{y,y(x)}$ factors in equation (54), we use an overall

[8] At least, it is as isotropic as the lattice model that we have introduced will allow!

[9] It is not clear that this guarantees complete convergence, but the training schedule is good enough to demonstrate the point that we wish to make.

update scheme

$$x_y'^{(new)} = x_y'^{(old)} + 0.1\,(x - x_y'^{(old)}) \qquad |y - y(x)| \leq n \tag{52}$$

6. Break up the $[0,1]$ interval into histogram bins, each of which covers a small interval $[b - \Delta/2, b + \Delta/2]$ of width Δ centred at $x = b$. As in [7] we choose to use 10 bins, so $\Delta = 0.1$ and the b are drawn from the set $\{0.05, 0.15, \ldots, 0.85, 0.95\}$. These bins are used to estimate the relative frequency with which the CVs land in each of the 10 intervals.

   In our experiments we modified the procedure that was used in [7] by incrementing the bins after every n_{cv} (= 30 in our experiments) training samples. Each bin thus contains a cumulative count of the number of CVs that have appeared in it (summed over all of the "snapshots" taken at intervals of n_{cv} samples).

   This procedure for incrementing the histogram has an infinitely long memory time, so the final histogram (after 500000 samples) will be the mixture of all histograms that occurred as training progressed towards convergence. This is obviously undesirable, and could be cured by imposing a finite memory by, for instance, making the histogram bins "leaky". We do not implement such refinements.

7. Do a least squares fit of $\rho(x)$ versus $P(x)$ as follows:

   (a) For histogram bin $[b - \Delta/2, b + \Delta/2]$ determine two quantities:

      i. The probability P_i that $P(x)$ generates a point lying in bin i: this can be *calculated* to be $2b_i\Delta$.

      ii. The probability ρ_i that a CV lies in bin i: this is *estimated* from the outcome of the numerical experiment as the proportion of CVs that land in bin i.

   (b) Plot the (P_i, ρ_i) coordinates from the previous step on a (P, ρ) graph (P is the abscissa, and ρ is the ordinate).

   (c) Define a parametric model $\rho(P) = A\,P^{\alpha}$.

   (d) Find the A and α that minimise the sum of squared errors $\sum_i (\rho(P_i - \rho_i))^2$. Because we use a rather small value of n_{cv} we find that edge effects (near $x = 0$ and $x = 1$) adversely affect the number of counts in the $b = 0.05$ (left-most) and $b = 0.95$ (right-most) histogram bins. We therefore discard these two bins and use only the central 8 (out of 10) bins to estimate A and α.

      In the fitting process we do *not* do a least squares fit to a logarithmic plot, because it is the estimated ρ_i (*not* the estimated $\log \rho_i$) that has approximately Gaussian errors.

8. Repeat the experiment many times in order to obtain many estimates for A and α, so that an improved estimate with a reduced standard error can be deduced. We do not implement this step.

We wrote a program to implement this numerical experiment, and we found that we could reproduce the results quoted in [7], when we used the standard TM training scheme [4].

In order to modify the experiment so that it conforms to our TVQ formulation of TMs, we need to alter the neighbourhood function used in step 4 of the numerical experiment. We used a neighbourhood of the type shown in figure 3 with just two components: a $[0,0]$ neighbourhood term, plus various types of $[-1,+1]$ neighbourhood term. Combining the ϵ and $\pi_{y,y(x)}$ factors in equation (54), we use an overall update scheme

$$\begin{aligned} x'^{(new)}_{y(x)} &= x'^{(old)}_{y(x)} + 0.1\,(x - x'^{(old)}_{y(x)}) \\ x'^{(new)}_{y(x)\pm 1} &= x'^{(old)}_{y(x)\pm 1} + \epsilon'\,(x - x'^{(old)}_{y(x)\pm 1}) \end{aligned} \tag{53}$$

where we choose ϵ' to be one of $\{0.025, 0.050, 0.075\}$.

We also need to change step 5 of the numerical experiment. In this step we should use *minimum distortion encoding* rather than *nearest neighbour* encoding. In fact, we perform experiments using both types of encoding scheme in order to compare the two outcomes.

Apart from these two changes, the TVQ experimental procedure is identical to the uniform neighbourhood function experimental procedure described above.

6.2 Numerical experimental results

The final estimated values of α (ie after 500000 updates) are shown in table 1.

ϵ'	TM	TVQ
0.025	0.51	0.32
0.050	0.51	0.31
0.075	0.52	0.35

Table 1: Table of asymptotic power law α for various simple neighbourhood functions. Both the topographic mapping case and the topographic vector quantiser case are shown.

These are the results of a *single* run, so we have not attempted to quote a standard error. Note how the TM case consistently produces an α which is much larger than the $\alpha = 1/3$ that a standard VQ would produce, but the TVQ case produces a result which is the same as a standard VQ. If our asymptotic theory is correct, then the slight differences (between 1/3 in the VQ case and the 0.31-0.35 observed in the TVQ case) are because next to leading order corrections contribute slightly for the $n_{cv} = 30$ case that we simulate. There are other possible explanations for these slight differences, such as the crudity of our process for estimating α.

We show in figure 7 a plot of the estimates for α as a function of (the logarithm to base 10 of) the number of training steps. These plots fall into two clearly separated categories which asymptotically approach (as the number of training steps increases) the results tabulated in table 1.

Even for $n_{cv} = 30$ we can easily produce a result for α that is remarkably close to the theoretically predicted asymptotic (ie $n_{cv} \to \infty$) result. This is very encouraging because we may use our asymptotic result with confidence to predict the density of CVs as a function

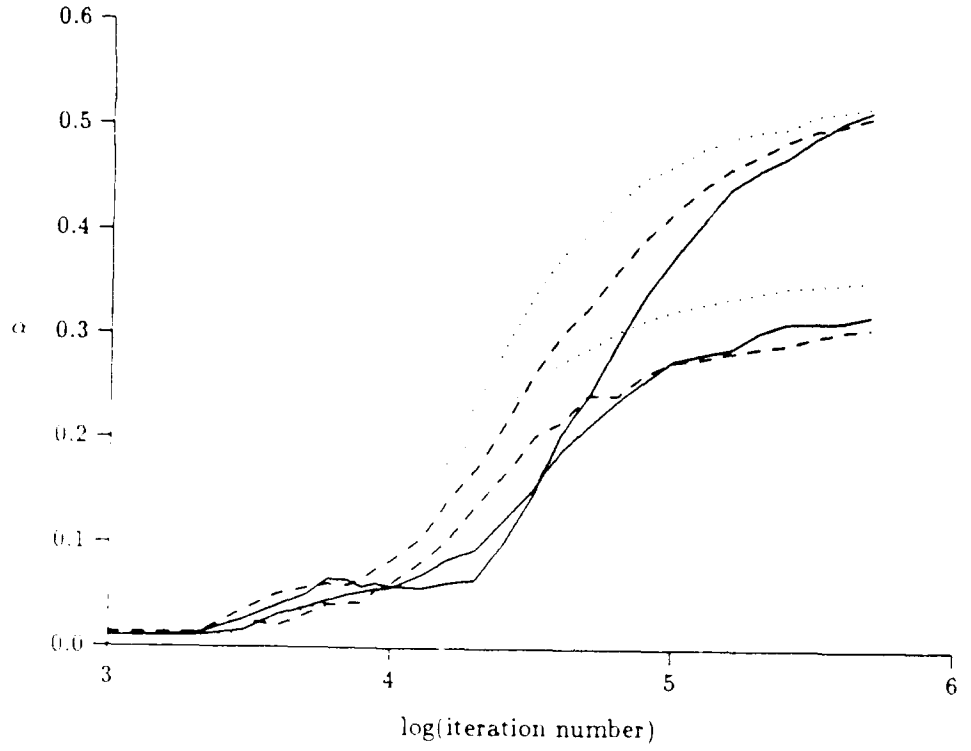


Figure 7: Plot of α versus the number of training steps for both the TM and the TVQ cases. The plots are coded as follows: $\epsilon' = 0.025$ (solid), $\epsilon' = 0.050$ (dashes), $\epsilon' = 0.075$ (dots)

of the input probability density, apart from some possible edge effects (that we artificially removed from our numerical simulation).

7 Conclusions and discussion

The asymptotic code vector density observed in topographic mappings depends on whether we use the method advocated in [4] (ie nearest neighbour encoding), or our own method (ie minimum L_2 distortion encoding). Our own method of optimising topographic mappings has the advantage of having a simple vector quantiser interpretation, and it leads to an asymptotic density which is the same (in leading order) as the plain vector quantiser result.

However, one has to be careful to avoid certain instabilities that can occur when optimising topographic vector quantisers. For instance, a uniform $[-n, +n]$ neighbourhood function leads to metastable solutions in which the code vectors tend to collapse into clusters of $2n + 1$ code vectors. This does not adversely affect the distortion, which is the same

as would have been obtained had the clusters been smoothed out. However, we find that for monotonically decreasing (as a function of radius) neighbourhood functions this clustering instability does not occur, although for slowly decreasing neighbourhood functions one has to perform small update steps during training in order to avoid clustering problems.

Our derivation of the asymptotic code vector density in N dimensions leaves something to be desired, and it should be regarded as no more than a suggestive model of what might actually occur in N dimensions.

A possible application of our result is to estimate the total input probability associated with each code vector. The simple relationship between code vector density and input probability density makes this calculation simple to perform. More generally, it should prove to be theoretically convenient that we have obtained a code vector density that is the same as in the plain vector quantiser.

Finally, the use of a topographic neighbourhood function disrupts the properties of a vector quantiser less than had hitherto been thought, *provided that* we use minimum distortion (rather than nearest neighbour) encoding.

Appendix A Density in 1 dimension: Ritter's theory

In this appendix we shall summarise the theory of the density of code vectors in a scalar topographic mapping as presented in [7]. We have deliberately changed the notation of [7] to conform to our own choice of notation, but have otherwise retained the theory intact.

A.1 Update procedure

The derivation in [7] is based upon the training procedure in [4], which is given as a prescription for updating a set of vectors x'_y. We shall refer to these x'_y vectors as code vectors in order to conform with our own vector quantiser version of topographic mappings, although the update scheme in [4] does *not* have a simple vector quantiser interpretation.

The update scheme is

$$x'^{(new)}_y = x'^{(old)}_y + \epsilon \pi_{y,y(x)}(x - x'^{(old)}_y) \tag{54}$$

where $\pi_{y,y(x)}$ is a neighbourhood function. If $\pi_{y,y(x)} = \delta_{y,y(x)}$ then only $x'_{y(x)}$ is updated. If, on the other hand, $\pi_{y,y(x)}$ has non-zero off-diagonal elements then other (neighbouring) x'_y ($y \neq y(x)$) are updated as well. The theory in [7] uses the same $\pi_{y,y(x)}$ that we defined in equation (12), which specifies that equation (54) updates those code vectors x'_y whose index y lies within the $[-n, +n]$ neighbourhood of $y(x)$.

A.2 Finite differences and derivatives

In [7] (and in [4]) it is *assumed* that the q_y are given by

$$q_y = \frac{1}{2}\left(x'_y + x'_{y+1}\right) \tag{55}$$

which is *the same as* the q_y used in a minimum L_2 distortion vector quantiser (see equation (5)), but is *different from* the q_y used in the topographic version of the same vector quantiser (see equation (13)) (assuming a $[-n, +n]$ neighbourhood). It is the difference between equation (55) and equation (13) that prevents equation (54) from leading to a minimum L_2 distortion when a non-zero neighbourhood size is used..

The midpoint and half-length analogous to our equation (15) and equation (16) are then given by

$$\begin{aligned}
u_i &= \frac{1}{2}\left(q_{i-n-1} + q_{i+n}\right) \\
&= \frac{1}{4}\left(x'_{i-n-1} + x'_{i-n} + x'_{i+n} + x'_{i+n+1}\right) \\
&= \frac{1}{4}\left(x'_{i-n-1} + x'_{i+n+1} - 2x'_i\right) + \frac{1}{4}\left(x'_{i-n} + x'_{i+n} - 2x'_i\right) + x'_i \\
&= x'_i + \frac{(n+1)^2 + n^2}{4}\frac{d^2x'_i}{di^2} + \mathcal{O}\left(n^4\frac{d^4x'_i}{di^4}\right)
\end{aligned} \tag{56}$$

$$\begin{aligned}
a_i &= \frac{1}{2}\left(q_{i+n} - q_{i-n-1}\right) \\
&= \frac{1}{4}\left(x'_{i+n} + x'_{i+n+1} - x'_{i-n-1} - x'_{i-n}\right) \\
&= \frac{1}{4}\left(x'_{i+n} - x'_{i-n}\right) + \frac{1}{4}\left(x'_{i+n+1} - x'_{i-n-1}\right) \\
&= \frac{2n+1}{2}\frac{dx'_i}{di} + \mathcal{O}\left(n^3\frac{d^3x'_i}{di^3}\right)
\end{aligned} \tag{57}$$

A.3 Update equilibrium

When the update procedure in equation (54) has converged then *on average* there is no net tendency for any of the code vectors to move to the right or the left: equilibrium has been attained. This may be expressed as

$$\int dx\, P(x)\, \pi_{y,y(x)}(x - x'_y) = 0 \tag{58}$$

which we may rearrange by dividing up the range of integration over x into intervals $[q_{y-1}, q_y]$ to yield

$$\sum_{y'} \int_{q_{y'-1}}^{q_{y'}} dx\, P(x)\, \pi_{y,y'}(x - x'_y) = 0 \tag{59}$$

which should be compared with the derivative in equation (9).

In [7], an equation for x'_y is then obtained that is identical to our equation (11), except that his q_y are specified by equation (55) rather than equation (13)

We do not need to repeat the derivation of an approximation to x'_y, because our result in equation (21) is applicable, provided that we use the expressions for u_y and a_y given in equation (56) and equation (57).

A.4 Code vector density

Finally in [7], an equation that is analogous to equation (22) is obtained

$$\frac{1}{\rho(x'_y)}\frac{d\rho(x'_y)}{dy} \simeq \frac{\alpha}{P(x'_y)}\frac{dP(x'_y)}{dy} \tag{60}$$

where α is given by

$$\alpha \equiv \frac{1}{3}\frac{(2n+1)^2}{(n+1)^2+n^2} \tag{61}$$

In leading order, the solution to the first order differential equation in equation (60) is

$$\rho(x'_y) \propto P(x'_y)^\alpha \tag{62}$$

When $n = 0$ the power law α becomes $\alpha = 1/3$, and in the limit $n \to \infty$ the power law α becomes $\alpha = 2/3$. Equation (62)

A.5 Comparison of Kohonen/Ritter method with Luttrell method

We observe that the use of the standard update procedure in equation (54) based on nearest neighbour coding in equation (55) leads to a complicated power law dependence, whereas our own scheme whereby we minimise an appropriately chosen L_2 distortion measure leads to a power law dependence that is *the same as* that observed in a standard vector quantiser (with zero neighbourhood size).

The price to pay for the convenience of using a minimum L_2 distortion approach (rather than the standard approach in [4]) is that encoding method is no longer nearest neighbour, and there are certain stability problems that can arise when $\pi_{y',y}$ is chosen inappropriately.

Appendix B Code vector density: full derivation

In this appendix we shall present a more extensive derivation of the optimum code vector density that does *not* assume that the distortion volume is characterised by a single radius. Thus we introduce a more sophisticated model in which we use a full covariance matrix to model the possibly anisotropic shape of the distortion volume, and we relate this covariance matrix to the required code vector density in order to make contact with the problem that we are trying to solve.

B.1 Euclidean distortion

First of all write the final result in equation (41) as

$$\begin{aligned} D_6 &= \int d\boldsymbol{y}\, P(\boldsymbol{y}) \operatorname{trace} \sigma(\boldsymbol{y}) \\ &= \int d\boldsymbol{y}\, P(\boldsymbol{x}) \operatorname{trace} \sigma(\boldsymbol{y}(\boldsymbol{x})) \end{aligned} \tag{63}$$

where $\sigma(\boldsymbol{y})$ is the posterior covariance matrix defined as

$$\sigma(\boldsymbol{y}) \equiv \left\langle \left(\boldsymbol{x} - \langle \boldsymbol{x} \rangle_{\boldsymbol{x}|\boldsymbol{y}}\right) \left(\boldsymbol{x} - \langle \boldsymbol{x} \rangle_{\boldsymbol{x}|\boldsymbol{y}}\right)^T \right\rangle_{\boldsymbol{x}|\boldsymbol{y}} \tag{64}$$

where the superscript "T" denotes "transpose".

B.2 Correspondence between code vector density and posterior covariance

By comparing equation (63) with equation (43) we may identify the correspondence

$$\rho(\boldsymbol{x})^{-\frac{2}{N}} \leftrightarrow \operatorname{trace} \sigma(\boldsymbol{y}(\boldsymbol{x})) \tag{65}$$

We shall use this relationship to determine the effective $\rho(\boldsymbol{x})$ that corresponds to any particular choice of $\sigma(\boldsymbol{y}(\boldsymbol{x}))$.

B.3 Functional derivative of Euclidean distortion

We need to minimise D_6 with respect to the elements of the matrix $\sigma(\boldsymbol{y})$, subject to the condition that the total volume associated with $\sigma(\boldsymbol{y})$ (for all $\boldsymbol{y}$) is held constant. Thus introduce C by analogy with equation (44) as

$$C = \int d\boldsymbol{x} \left[\det \sigma(\boldsymbol{y}(\boldsymbol{x}))\right]^{-\frac{1}{2}} \tag{66}$$

and by analogy with equation (45) we minimise $D_6' = D_6 + \lambda C$ with respect to the elements of the matrix $\sigma(\boldsymbol{y})$.

In order to functionally differentiate D_6 with respect to $\sigma_{ij}(\boldsymbol{\xi})$ we need the following two results

$$\frac{\delta \operatorname{trace} \sigma(\boldsymbol{y})}{\delta \sigma_{ij}(\boldsymbol{\xi})} = \delta_{ij}\, \delta(\boldsymbol{y} - \boldsymbol{\xi}) \tag{67}$$

$$\begin{aligned} \frac{\delta \left[\det \sigma(\boldsymbol{y})\right]^{-\frac{1}{2}}}{\delta \sigma_{ij}(\boldsymbol{\xi})} &= -\frac{1}{2} \left[\det \sigma(\boldsymbol{y})\right]^{-\frac{1}{2}} \frac{\delta \log \det \sigma(\boldsymbol{y})}{\delta \sigma_{ij}(\boldsymbol{\xi})} \\ &= -\frac{1}{2} \left[\det \sigma(\boldsymbol{y})\right]^{-\frac{1}{2}} \frac{\delta \operatorname{trace} \log \sigma(\boldsymbol{y})}{\delta \sigma_{ij}(\boldsymbol{\xi})} \\ &= -\frac{1}{2} \left[\det \sigma(\boldsymbol{y})\right]^{-\frac{1}{2}} \left[\sigma(\boldsymbol{y})\right]^{-1}_{ji} \delta(\boldsymbol{y} - \boldsymbol{\xi}) \end{aligned} \tag{68}$$

By inserting the results in equation (67) and equation (68) into $\delta D_6'/\delta\sigma_{ij}(\boldsymbol{y})$ we obtain

$$\begin{aligned}\frac{\delta D_6'}{\delta\sigma_{ij}(\boldsymbol{y})} &= \int d\boldsymbol{x}\, P(\boldsymbol{x})\,\delta_{ij}\,\delta(\boldsymbol{y}(\boldsymbol{x})-\boldsymbol{y}) - \frac{\lambda}{2}\int d\boldsymbol{x}\,[\sigma(\boldsymbol{y})]^{-1}_{ji}\,\delta(\boldsymbol{y}(\boldsymbol{x})-\boldsymbol{y})\,[\det\sigma(\boldsymbol{y})]^{-\frac{1}{2}} \\ &= \int d\boldsymbol{x}\,\delta(\boldsymbol{y}(\boldsymbol{x})-\boldsymbol{y})\left\{P(\boldsymbol{x})\,\delta_{ij} - \frac{\lambda}{2}\,[\det\sigma(\boldsymbol{y})]^{-\frac{1}{2}}\,[\sigma(\boldsymbol{y})]^{-1}_{ji}\right\} \qquad (69)\end{aligned}$$

B.4 Minimum Euclidean distortion

We must now solve the stationarity condition $\delta D_6'/\delta\sigma_{ij}(\boldsymbol{y}) = 0$ to find the minimum distortion choice of $\sigma_{ij}(\boldsymbol{y})$. If we ignore the assumed small variation of $P(\boldsymbol{x})$ as $\boldsymbol{x}$ ranges over inputs that map to $\boldsymbol{y} = \boldsymbol{y}(\boldsymbol{x})$ (this approximation is valid in leading order), then the stationarity condition reduces to requiring that the *integrand* of equation (69) be zero. Thus

$$P(\boldsymbol{x})\,\delta_{ij} \simeq \frac{\lambda}{2}\,[\det\sigma(\boldsymbol{y}(\boldsymbol{x}))]^{-\frac{1}{2}}\,[\sigma(\boldsymbol{y}(\boldsymbol{x}))]^{-1}_{ji} \qquad (70)$$

which implies that

$$\sigma_{ij}(\boldsymbol{y}) \simeq \sigma_0(\boldsymbol{y})\,\delta_{ij} \qquad (71)$$

where the $\sigma_0(\boldsymbol{y})$ on the right hand side is a scalar. We have thus deduced that (for each code vector) the optimum posterior covariance matrix is characterised by a single parameter $\sigma_0(\boldsymbol{y})$ corresponding to the (squared) radius of the corresponding distortion volume.

B.5 Code vector density

Combining equation (70) and equation (71) yields

$$\begin{aligned}P(\boldsymbol{x}) &\propto [\sigma_0(\boldsymbol{y}(\boldsymbol{x}))]^{-\frac{N}{2}}\,[\sigma_0(\boldsymbol{y}(\boldsymbol{x}))]^{-1} \\ &\propto [\sigma_0(\boldsymbol{y}(\boldsymbol{x}))]^{-\frac{N+2}{2}} \qquad (72)\end{aligned}$$

which we may invert to obtain the optimum $\sigma_0(\boldsymbol{y}(\boldsymbol{x}))$ in terms of the input probability density $P(\boldsymbol{x})$

$$\sigma_0(\boldsymbol{y}(\boldsymbol{x})) \propto P(\boldsymbol{x})^{-\frac{2}{N+2}} \qquad (73)$$

and using the correspondence in equation (65) we may also obtain $\rho(\boldsymbol{x})$ in terms of $P(\boldsymbol{x})$ as

$$\rho(\boldsymbol{x}) \propto P(\boldsymbol{x})^{\frac{N}{N+2}} \qquad (74)$$

which is the required result.

References

[1] Lloyd S P. Least squares quantisation in PCM. *IEEE Trans. IT*, **28**, 129-137, 1982.

[2] Max J. Quantising for minimum distortion. *IRE Trans. Inform. Th.*, **6**, 7-12, 1960.

[3] Linde Y, Buzo A and Gray R M. An algorithm for vector quantiser design. *IEEE Trans. COM*, **28**, 84–95, 1980.

[4] Kohonen T. *Self organisation and associative memory.* Springer-Verlag, 1984.

[5] Luttrell S P. Self-organisation: a derivation from first principles of a class of learning algorithms. In *Proc. 3rd IEEE Int. Joint Conf. on Neural Networks*, volume 2, pages 495–498, Washington DC, 1989.

[6] Luttrell S P. Hierarchical self-organisation. In *Proc. 1st IEE Conf. on Artificial Neural Networks*, pages 2–6, London, 1989.

[7] Ritter H. Asymptotic level density for a class of vector quantisation processes. *Helsinki University Report, Laboratory of Computer and Information Science*, **A9**, 1989.

REPORT DOCUMENTATION PAGE

DRIC Reference Number (if known) .....

Overall security classification of sheet .....Unclassified.....

(As far as possible this sheet should contain only unclassified information. If it is necessary to enter classified information, the field concerned must be marked to indicate the classification eg (R), (C) or (S).

Originators Reference/Report No.	Month	Year
MEMO 4392	MAY	1990

Originators Name and Location
RSRE, St Andrews Road
Malvern, Worcs WR14 3PS

Monitoring Agency Name and Location

Title
ASYMPTOTIC CODE VECTOR DENSITY IN TOPOGRAPHIC VECTOR QUANTISERS

Report Security Classification	Title Classification (U, R, C or S)
UNCLASSIFIED	U

Foreign Language Title (in the case of translations)

Conference Details

Agency Reference	Contract Number and Period

Project Number	Other References

Authors	Pagination and Ref
LUTTRELL, S P	26

Abstract

In this memorandum we use a noise-robust vector quantiser model to derive expressions for the asymptotic code vector density ρ in various types of topographic vector quantisers. A topographic vector quantiser is not identical to a standard (ie Kohonen) topographic mapping, but the differences are minimal. In all the cases that we study (scalar and vector quantisation with various symmetric topographic neighbourhoods), we obtain the asymptotic result $\rho \propto P^{N/N+2}$, where N is the input dimensionality and P is the input probability density. Thus the asymptotic code vector densities of a topographic vector quantiser and a standard vector quantiser are the same.

Abstract Classification (U,R,C or S)
U

Descriptors

Distribution Statement (Enter any limitations on the distribution of the document)
UNLIMITED

S80 48