In coding theory one transforms signals from a source representation into an encoded representation that is suitable for transmission through a (possibly noisy) medium, and upon reception one decodes to reconstruct an approximation to the original signal. In autoassociative network theory one reconstructs a signal given incomplete (and possibly noisy) information about it. We derive some new results by combining these two approaches in the form of vector quantisation (VQ) theory and topographic mapping (TM) theory. We use a VQ model (with a noisy transmission medium) to model the processes that occur in TMs, which leads to the standard TM training algorithm, albeit with a slight modification to the encoding process (minimum distortion rather than nearest neighbour encoding). To emphasise this difference we call our model a topographic vector quantiser (TVQ). In the continuum limit of the one-dimensional (scalar) TVQ we find that the density of code vectors is proportional to $P(x)^{\alpha}$ ($\alpha = \frac{1}{3}$) (which is the same as the result obtained from a standard scalar quantiser), assuming that the transmission medium introduces additive noise with a zero-mean, symmetric, monotonically decreasing probability density (which is equivalent to using a symmetrically tapered neighbourhood in a TM). Our $\alpha = \frac{1}{3}$ result is dramatically different from the $\alpha = \frac{1}{2}$ result that is predicted when the standard TM training algorithm is used with a uniform symmetric neighbourhood $[-n, n]$, and we note that this difference arises entirely from using minimum distortion rather than nearest neighbour encoding. We verify our new result by performing a numerical experiment using $P(x) \propto x$.

I. INTRODUCTION

This paper is concerned with the overlap between two related subjects. On the one hand there is much literature on vector quantisation (VQ) and scalar quantisation theory [14], where an encoding/decoding scheme is optimised in such a way as to minimise a distortion measure. On the other hand there is an interesting class of transformations (called topographic mappings (TM)) in the neural network literature [15] that can be trained to perform optimal mappings of high-dimensional input vectors into low-dimensional output vectors.

In [5] and [6] we unified these two subjects by formulating the problem of training a TM in terms of minimising a distortion measure. This required a slight modification of the original TM training algorithm, but the observed side effects of this were minimal. We call this type of mapping a topographic vector quantiser (TVQ) in order to emphasize its close relationship to a standard VQ, and to distinguish it from the TM method. A TVQ has the important property that it encodes information in such a way that it is robust with respect to the damaging effects of a noise process which makes transitions between code indices which lie within the same topographic neighbourhood. A good example of the use of this approach in designing robust codes can be found in [7].

The question of the asymptotic properties of TVQs (versus those of VQs) naturally arises. In a recent study [8] the asymptotic code vector (CV) density in a scalar TM (one dimension mapped to one dimension) was derived and was found to be proportional to $P(x)^{\alpha}$, where $\alpha = \frac{1}{2}$, $P(x)$ being the probability density of input scalars and $n$ the half-width of the update neighbourhood used when training the TM. The $n = 0$ case reduces to $\alpha = \frac{1}{2}$, which is consistent with the result that is expected from a scalar quantiser [1].

We wish to derive the corresponding TVQ result for the case where we solve a minimum $L_2$ distortion problem (as formulated in [3] and [5]), rather than the standard TM problem (as formulated in [4]). For simplicity, we shall restrict our attention in this paper to the scalar case, so our results become an extension of the results that were reported in [8].

In Section [11] we use a uniform topographic neighbourhood function to derive the asymptotic CV density in a scalar TVQ (this directly corresponds to the scalar TM derivation in [8]). In Section [11] we extend this result to the more general case of a tapered (i.e., symmetric and monotonically decreasing to zero) topographic neighbourhood function. In Section [11] we present the results of some numerical simulations that verify the theoretical results in Section [11]. In the Appendix we briefly explain the origin of and cure for a stability problem that can arise when training TVQs.

It is important to note that although our derivations refer to the case of one input dimension (i.e. the scalar case), we frequently use the word “vector” when referring to the input. Although this might seem bizarre, it reduces the amount of additional nomenclature that we have to introduce. Thus we speak of a “topographic vec-
tor quantiser” when we should strictly say “topographic scalar quantiser”, and similarly “code vector” instead of “code scalar”.

II. DENSITY WITH A UNIFORM TOPOGRAPHIC NEIGHBOURHOOD FUNCTION

In this section we derive conditions for the stationarity (with respect to variation of the encoding/decoding functions) of the distortion caused by TVQ. We use these results to derive the asymptotic CV density in the case of a uniform topographic neighbourhood function (as used in [8]).

A. Euclidean distortion

Introduce an $L_2$ distortion measure $D_1$ as

$$D_1 = \int dx P(x) (x - x'_{y(x)})^2$$

$$= \sum_q \int_{q_{y-1}}^{q_y} dx P(x) (x - x'_{y})^2 \tag{2.1}$$

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. An encoding function, $y(x)$, maps from the input $x$ to a code index $y$. The interval $[q_{y-1}, q_y]$ depends on $y(x)$ as follows:

$$[q_{y-1}, q_y] = \{x : y = y(x)\} \tag{2.2}$$

which we use in Equation 2.1 to partition the range of integration into a set of convenient intervals. The output $x'_{y}$ is the decoding function (i.e., the CV) 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 values 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 $x'_{y(x)}$. Note that we have simplified Figure 1 by spacing the CVs at constant intervals for illustrative purposes.

B. 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 2.2). Thus we obtain the partial derivatives as

$$\frac{\partial D_1}{\partial q_y} = P(q_y) [(q_y - x'_{y})^2 - (q_y - x'_{y+1})^2]$$

$$= 2P(q_y) (x'_{y+1} - x'_{y}) (q_y - x'_{y+1}) \tag{2.3}$$

$$\frac{\partial D_1}{\partial x'_{y}} = -2 \int_{q_{y-1}}^{q_y} dx P(x) (x - x'_{y}) \tag{2.4}$$

whence the stationary points of $D_1$ must satisfy

$$q_y = \frac{x'_{y} + x'_{y+1}}{2} \tag{2.5}$$

$$x'_{y} = \frac{\int_{q_{y-1}}^{q_y} dx P(x)x}{\int_{q_{y-1}}^{q_y} dx P(x)} \tag{2.6}$$

Note that Equation 2.5 requires that $q_y$ lie midway between the adjacent CVs, so it defines a nearest neighbour encoding function $y(x)$.

C. Topographic Euclidean distortion

Now we generalise the $L_2$ distortion measure of Equation 2.1 to include a neighbourhood function $\pi_{y' y}$ that specifies the extent to which $y'$ is in the neighbourhood of $y$ (later on we define this notion more precisely). Thus introduce the $L_2$ distortion measure $D_2$ as

$$D_2 = \int dx P(x) \sum_{y'} \pi_{y', y(x)} (x - x'_{y'})^2$$

$$= \sum_y \int_{q_{y-1}}^{q_y} dx P(x) \sum_{y'} \pi_{y', y(x)} (x - x'_{y})^2 \tag{2.7}$$

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 a modified version of Figure 1 in which the effect of the $\pi_{y' y}$ is represented (for simplicity, we show only $\pi_{y+1, y}$). 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
the TV Q (minimum distortion encoding) and TM (near-
largest differences between the asymptotic CV density in
these small differences between encoding schemes lead to
casionally produces a different encoding of the input, but

cification of Equation 2.5 to become Equation 2.10 only oc-

obtained in Equation 2.5.

reco ver the nearest neighbour encoding property that we

so the stationary points of \( D_2 \) must satisfy

\[
\sum_{y'} \pi_{y',y} (q_y - x'_y)^2 = \sum_{y'} \pi_{y',y+1} (q_y - x'_y)^2 \quad (2.10)
\]

\[
x'_y = \frac{\sum_{y'} \int_{y'-1}^{y'} dx P(x) \pi_{y,y'} (x - x'_y)}{\sum_{y'} \int_{y'-1}^{y'} dx P(x) \pi_{y,y'}} \quad (2.11)
\]

In general, when \( \pi_{y',y} \neq \delta_{y',y} \) in Equation 2.10 we cannot recover the nearest neighbour encoding property that we obtained in Equation 2.5.

In practical applications we have found that the modification of Equation 2.5 to become Equation 2.10 only occasionally produces a different encoding of the input, but these small differences between encoding schemes lead to large differences between the asymptotic CV density in the TVQ (minimum distortion encoding) and TM (nearest neighbour encoding) cases.

Figure 2: Network representation of encoding an input \( x \) to produce a code \( y \), followed by corruption of the code by a transition matrix \( \pi \) and then decoding to produce a reconstruction \( x' \).

E. Finite differences and derivatives

In this subsection we bring 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 [8] we now assume a specific form for \( \pi_{y',y} \)

\[
\pi_{y',y} = \begin{cases} 
1 & \text{if } |y' - y| \leq n \\
0 & \text{if } |y' - y| > n 
\end{cases} \quad (2.12)
\]

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 call a \([-n,+n]\) neighbourhood.

With this assumption we can solve Equation 2.10 to yield

\[
q_y = \frac{1}{2} (x'_{y-n} + x'_{y+n+1}) \quad (2.13)
\]

which should be compared with the result in Equation 2.3 (which corresponds to making the choice \( \pi_{y',y} = \delta_{y',y} \) or \( n = 0 \), in Equation 2.13). The effect of the \([-n,+n]\) neighbourhood is to replace the midpoint of the interval \([x'_{y-1}, x'_{y+1}]\) by the midpoint of the larger interval \([x'_{y-n}, x'_{y+n+1}]\). Note that \( x'_{y} \leq q_y \leq x'_{y+1} \) is not necessarily true when \( n > 0 \).

Now introduce a pair of expressions to relate the finite differences of \( x'_y \) to the derivatives \( \frac{dx'_{y}}{dy} \) and \( \frac{dx'_{y}}{d^2y} \) of \( x'_y \).

\[
x'_{y+k} - x'_{y-k} = 2k \frac{dx'_{y}}{dy} + \mathcal{O}(k^3 \frac{dx'_{y}}{d^2y}) \\
x'_{y+k} + x'_{y-k} - 2x'_{y} = k^2 \frac{d^2x'_{y}}{dy^2} + \mathcal{O}(k^4 \frac{d^3x'_{y}}{dy^3}) \quad (2.14)
\]

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 2.14.

Using Equation 2.13 and Equation 2.14 we may derive the midpoint \( u_y \) and the half-length \( a_y \) of the interval \([q_{y-n-1}, q_{y+n}]\) as

\[
u_y = \frac{1}{2} (q_{y-n-1} + q_{y+n}) \\
u_y = \frac{1}{2} (x'_{y-2n-1} + x'_{y+2n+1} + 2x'_{y}) \\
u_y \approx x'_{y} + \frac{(2n+1)^2}{4} \frac{d^2x'_{y}}{dy^2} \\
u_y \approx \frac{1}{2} (q_{y+n} - q_{y-n-1}) \\
u_y \approx \frac{1}{2} (x'_{y+2n+1} - x'_{y-2n-1}) \quad (2.15)
\]

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:

\[
\rho(x'_y) = \frac{dy}{dx'_y} \quad (2.17)
\]

Thus \( \rho(x'_y) \) is the number of CV indices \( y \) per unit change in CV position \( x'_y \). In Equation 2.14 we encountered derivatives of \( x'_y \) which we now express in terms of
derivatives of \( \rho(x') \). Thus,

\[
\begin{align*}
\frac{dx'}{dy} & = \frac{1}{\rho(x')} \\
\frac{d^2x'}{dy^2} & = -\frac{1}{\rho(x')} \frac{d\rho(x')}{dy}
\end{align*}
\tag{2.18}
\]

where we have used \( \frac{d}{dy} \frac{dx'}{dy} = \frac{1}{\rho(x')} \frac{d}{dy} \frac{d}{dx'} \) to perform the differentiation. We may thus express the results for \( u_y \) (Equation 2.15) and \( a_y \) (Equation 2.16) in terms of derivatives of \( \rho(x') \) as follows:

\[
\begin{align*}
\frac{u_y}{x'} & \simeq \frac{(2n+1)^2}{4\rho(x')} \frac{d\rho(x')}{dy} \\
\frac{a_y}{\rho(x')} & \simeq \frac{2n+1}{2\rho(x')}
\end{align*}
\tag{2.19, 2.20}
\]

We now have all the basic theoretical results that are needed to perform a Taylor expansion of Equation 2.11 to relate the derivative of \( P(x) \) to the derivative of \( \rho(x) \). Insert the \( \pi_{y',y} \) (defined in Equation 2.12), 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 2.15 and Equation 2.16 respectively) in Equation 2.11 to obtain

\[
\begin{align*}
x'(y) & = \int_{-a_y}^{a_y} dx' P(x)(u_y+x) \\
& = u_y + \sum_{m=1}^{\infty} \frac{1}{m} \left( \sum_{m=1}^{\infty} \frac{\rho(m)(u_y)_{m+1}^{m}}{m} \right) \\
& = u_y + \sum_{m=1}^{\infty} \frac{1}{m} \left( \sum_{m=1}^{\infty} \frac{\rho(m)(u_y)_{m+1}^{m}}{m} \right) \\
& = u_y + a_y \left( \frac{dP(u_y)}{dy} + \frac{a_y^2 P''(u_y)}{3P(u_y)} \right) + \text{h.o.t.} \\
& = u_y + a_y \left( \frac{dP(u_y)}{dy} + \frac{a_y^2 P''(u_y)}{3P(u_y)} \right) + \text{h.o.t.}
\end{align*}
\tag{2.21}
\]

Throughout this derivation we use the notation \( P^{(n)}(u_y) \) to denote the \( n \)th derivative of \( P(u_y) \) with respect to \( u_y \) (and similarly \( P^{(n)}(x') \)). We assume the dimensionless inequality \( a_y^2 \frac{P''(u_y)}{P(u_y)} \ll 1 \) for \( n \geq 1 \), and we assume that the \( n \geq 3 \) terms are negligible in comparison with the leading terms. We can therefore discard terms with squares (and higher powers) of \( a_y \frac{P''(u_y)}{P(u_y)} \). In the last stage of this derivation note that the effect of the change \( u_y \rightarrow x' \) appears only in the next to leading order terms.

### G. Code vector density

Finally, inserting the expressions for \( u_y \) and \( a_y \) (from Equation 2.19 and Equation 2.20) into Equation 2.21 we obtain

\[
\frac{1}{\rho(x')} \frac{d\rho(x')}{dy} = \frac{1}{3P(x')} \frac{dP(x')}{dy}
\tag{2.22}
\]

We may solve this to yield asymptotically

\[
\rho(x') = P(x')^{1/3}
\tag{2.23}
\]

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 [5] for a TM as defined in [4]. These differences arise because we use minimum distortion encoding within the context of our TVQ model, rather than nearest neighbour encoding within the context of the TM model.
III. DENSITY WITH A TAPERED TOPOGRAPHIC NEIGHBOURHOOD FUNCTION

In this section we extend the results of Section II to the case where \( \pi_{y',y} \) defines a tapered topographic neighbourhood function. Specifically, we shall use a symmetric function that monotonically decreases to zero.

A. Topographic Euclidean distortion

We now define the distortion matrix \( \pi_{y',y} \) (used in the definition of the \( L_2 \) distortion \( D_2 \) in Equation 2.7) in such a way that it satisfies

\[
\begin{align*}
\pi_{y'+1,y+1} &= \pi_{y',y} \quad \text{(Toeplitz matrix)} \\
\pi_{y',y} &= \pi_{y,y'} \quad \text{(symmetric matrix)}
\end{align*}
\] (3.1)

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 Section II, we decompose \( \pi_{y',y} \) as a weighted sum over (symmetric) neighbourhood functions of the type defined in Equation 2.12

\[
\pi_{y',y} = \sum_{s:|y' - y| \leq n_s} h_s
\] (3.2)

As discussed in the Appendix, we should ensure that \( \pi_{y',y} \) is a monotonically decreasing function of \( y' - y \) by setting \( h_s > 0 \). In Figure 3 we give an example of the type of tapered neighbourhood function that is described by this model. We show in Figure 3(a) a typical \( \pi_{y',y} \) neighbourhood function and in Figure 3(b) its decomposition as a sum over \([-n_s, +n_s]\) neighbourhoods for various \( s \).

Using the definition of \( \pi_{y',y} \) in Equation 3.2 we can simplify \( D_2 \) in Equation 2.7 to become \( D_3 \) given by

\[
D_3 = \sum_{y} \int_{q_y}^{q_{y+1}} dx P(x) \sum_{s} \sum_{y' = -n_s}^{+n_s} (x - x' y + y')^2
\] (3.3)

Figure 3: An example of the decomposition of a symmetrically tapered topographic neighbourhood function into a sum of rectangular pieces. (a) Net neighbourhood function. (b) Decomposed neighbourhood function.

B. Minimum topographic Euclidean distortion

Now differentiate \( D_3 \) to obtain \( \frac{\partial D_3}{\partial q_y} \) and \( \frac{\partial D_3}{\partial x_y} \). The stationary points of \( D_3 \) must then satisfy

\[
q_y = \frac{\sum_{s} h_s (x_{y+n_s+1} - x_{y-n_s}) (x_{y+n_s+1} + x_{y-n_s})}{2 \sum_{s} h_s (x_{y+n_s+1} - x_{y-n_s})}
\] (3.4)

\[
x_y' = \frac{\sum_{s} h_s \int_{q_{y-n_s-1}}^{q_{y+n_s}} dx P(x) x}{\sum_{s} h_s \int_{q_{y+n_s}}^{q_{y+n_s+1}} dx P(x)}
\] (3.5)

Equation 3.4 replaces Equation 2.13 and Equation 3.5 replaces Equation 2.11.

Note that the positivity of the \( h_s \) in Equation 3.2 guarantees the stability of the solution for the \( q_y \) and the \( x_y' \) in Equation 3.4 and Equation 3.5 because the denominators are strictly positive. Note that 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.

Unfortunately, the expression for \( q_y \) in Equation 3.4 is sufficiently complicated that we have to perform a large amount of algebra to derive the asymptotic relationship between \( P(x) \) and \( \rho(x) \) (i.e., the generalisation of Equa-
C. Finite differences and derivatives

Firstly, express \( x'_{y+k} \pm x'_{y+l} \) in terms of \( \frac{dx'}{dy} \) and \( \frac{d^2x'}{dy^2} \):

\[
x'_{y+k} \pm x'_{y+l} = \begin{cases} 
\frac{1}{2} (x'_{y+k} + x'_{y-k} - 2x'_{y}) + \frac{1}{2} (x'_{y+k} - x'_{y-k}) + \frac{1}{2} (x'_{y+l} + x'_{y-l} - 2x'_{y}) + \frac{1}{2} (x'_{y+l} - x'_{y-l}) \\
\frac{1}{2} (x'_{y+k} + x'_{y-k} - 2x'_{y}) + \frac{1}{2} (x'_{y+k} - x'_{y-k}) + \frac{1}{2} (x'_{y+l} + x'_{y-l} - 2x'_{y}) + \frac{1}{2} (x'_{y+l} - x'_{y-l}) 
\end{cases}
\]

\[
\simeq \left( \frac{k^2+l^2}{2} \right) \frac{dx'}{dy} + (k \pm l) \frac{d^2x'}{dy^2} + (x'_{y+k} \pm x'_{y+l})
\] (3.6)

where we have used the finite difference expressions in Equation [2.14]. Note that we use ± signs consistently throughout Equation 3.6 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 \( q_{y+n_x} \) and \( q_{y-n_x-1} \) to obtain

\[
\begin{align*}
q_{y+n_x} &= \frac{\sum h_1(\xi_1(t)+\xi_2(s,t))(\eta_0+\eta_1(s)+\eta_2(s,t))}{2\sum h_1(\xi_1(t)+\xi_2(s,t))} \\
q_{y-n_x-1} &= \frac{\sum h_1(\xi_1(t)-\xi_2(s,t))(\eta_0-\eta_1(s)-\eta_2(s,t))}{2\sum h_1(\xi_1(t)-\xi_2(s,t))}
\end{align*}
\] (3.7)

where we have defined \( \xi_1(t) \), \( \xi_2(s,t) \), \( \eta_0 \), \( \eta_1(s) \) and \( \eta_2(s,t) \) as

\[
\begin{align*}
\xi_1(t) &= (2n_t+1) \frac{dx'}{dy} \\
\xi_2(s,t) &= (n_s+n_t+1)^2(n_x-n_s)^2 \frac{d^2x'}{dy^2} \\
\eta_0 &= 2x' \\
\eta_1(s) &= (2n_s+1) \frac{dx'}{dy} \\
\eta_2(s,t) &= (n_s+n_t+1)^2(n_x-n_s)^2 \frac{d^2x'}{dy^2}
\end{align*}
\] (3.8)

We may now introduce the midpoint \( u'^*_y \) and half-length \( a'^*_y \) of the interval \([q_{y-n_x-1}, q_{y+n_x}]\) and use Equation 3.7 to simplify their expressions. For compactness, we gather these two results into a single vector equation (where \( u'^*_y \) is the upper element and \( a'^*_y \) the lower element in a two-component column vector):

\[
\begin{pmatrix}
u'^*_y \\
a'^*_y
\end{pmatrix} = \frac{1}{2} \begin{pmatrix}
q_{y+n_x} + q_{y-n_x-1} \\
q_{y+n_x} - q_{y-n_x-1}
\end{pmatrix} + \eta_1(s) \left( \begin{pmatrix}
\xi_1(t)^2 \\
\xi_1(t)^2
\end{pmatrix} \right) + \eta_0(\xi_1(t)^2 - 2\xi_1(t)\xi_2(s,t) + \xi_2(s,t)^2)
\]

\[
= \frac{1}{2} \sum_{i',t'} h_{i't'}(\xi_1(t')^2 - \xi_2(s,t')^2 - \xi_1(t')\xi_2(s,t') - \xi_2(s,t')\xi_1(t'))
\] (3.9)

We have made use of the fact that \( \sum_{i',t'} h_{i't'}(\xi_1(t')\xi_2(s,t') - \xi_1(t)\xi_2(s, t')) = 0 \) (by symmetry) to simplify 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 \( y \) (see Equation 3.8):

\[
\xi_2(s,t)(\xi_1(t)^2 + \xi_1(t)\xi_2(s,t)) \simeq 0 \quad (3.10)
\]
We have expressed these results in such a way that they may readily be compared with the analogous results in Equation 2.19 and Equation 2.20. When comparing Equation 3.11 with Equation 2.19, note that there is a leading order correction term caused by the presence of more than one component in the tapered neighbourhood function, but note that Equation 2.20 needs no such leading order correction to become Equation 3.12.

D. Approximate optimal code vector positions

We now form a Taylor expansion of the integrands in the numerator and denominator of Equation 3.13. The steps in the derivation are analogous to those used to derive the result in Equation 2.21 so we present only the final result, which is

\[
\delta y_i \simeq \frac{\sum_{t \neq t'} h_t h_{t'} \xi_1(t) \xi_1(t') \left( (n_y + n_2(s,t)) \right)}{d^2 x_i / dy^2}
= x_i' + \frac{\sum_{t \neq t'} h_t (2n_{i+1}) \left( (n_y + n_1(t)) \left( (n_x + n_2(t)) \right) \right)}{2 \sum_{t} h_t (2n_{i+1})} d^2 x_i / dy^2
\]

\[
\simeq x_i' - \frac{\sum_{t \neq t'} h_t h_{t'} \xi_1(t) \xi_1(t') n_t(s)}{2 \sum_{t} h_t (2n_{i+1})}
\]

\[
a_{y_i} \simeq \frac{\sum_{t \neq t'} h_t h_{t'} \xi_1(t) \xi_1(t')}{2 \sum_{t} h_t (2n_{i+1})}
\]

Finally, inserting the results of Equation 3.15 into the leading order Taylor expansion in Equation 3.13 yields (in leading order) the same differential equation that we obtained in Equation 2.22. Thus we have shown that for the class of \( \pi_{y', y} \) corresponding to symmetric monotonically decreasing neighbourhood functions the asymptotic CV density is given by \( \rho(x' y) = P(x' y)^{1/3} \) (i.e., the same as Equation 2.23).

IV. NUMERICAL SIMULATION

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

A. Numerical experimental procedure

We now describe a variant of the numerical experiment that was performed in [8].

\[
x_y^{\text{new}} = x_y^{\text{old}} + 0.1 \left( x - x_y^{\text{old}} \right) |y - y(x)| \leq n
\]

(4.1)

1. Define a finite support for \( x : x \in [0, 1] \).
2. Define a probability density \( P(x) \) of input scalars: \( P(x) = 2x \) (i.e., a ramp).
3. Choose the number \( n_{cv} \), of CVs (in this case, code scalars) that you wish to use. We use \( n_{cv} = 30 \) because we find that this is large enough for the results of the numerical experiment to approximate those that we would expect in the asymptotic (i.e., \( n_{cv} \rightarrow \infty \)) case.
4. Choose the number \( n \) that determines the size of the \( [-n, +n] \) neighbourhood, in the form given in Equation 2.12.

\[
\sum_{s} h_s \left( \right) \simeq \frac{\sum_{t \neq t'} h_t h_{t'} \xi_1(t) \xi_1(t')}{2 \sum_{t} h_t (2n_{i+1})}.
\]

where we note that the contribution of the correction term in Equation 3.11 disappears (by symmetry).
5. Adapt the positions of the CVs using the standard training scheme for TMs with nearest neighbour encoding. In Equation (4) we use an update step size $\epsilon = 0.1$, and we train using 500,000 inputs $x$ that are sampled independently from $P(x)$. 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.

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 $\Delta$ centred at $x = b$. We use ten bins, so $\Delta = 0.1$ and $b = 0.05, 0.15, \cdots, 0.85, 0.95$. These bins are used to estimate the relative frequency with which the CVs land in each of the ten intervals.

In our experiments we increment 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 500,000 samples) will be the mixture of all histograms that occurred as training progressed toward 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 $\rho_i$ 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 $(P_i, \rho_i)$ coordinates from the previous step on a $(P, \rho)$ graph ($P$ being the abscissa and $\rho$ the ordinate).

(c) Define a parametric model $\rho(P) = AP^\alpha$.

(d) Find the $A$ and $\alpha$ that minimise the sum of squared errors $\sum (\rho(P_i) - \rho_i)^2$. 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 $b = 0.05$ (leftmost) and $b = 0.95$ (rightmost) histogram bins. We therefore discard these two bins and use only the central eight (out of ten) bins to estimate $A$ and $\alpha$.

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

Although our numerical experimental procedure is somewhat cruder than in [8], we nevertheless found that by using the standard TM update procedure we could reproduce the numerical results that are quoted in [8].

We then made two changes to the numerical experiment in order to simulate a TVQ rather than a TM. First, we generalised step 4 of the numerical experiment to permit either minimum distortion encoding or nearest neighbour encoding to be used. Second, we introduced a tapered symmetric neighbourhood of the type shown in Figure 3. In our simulations we used two components: a $[0,0]$ neighbourhood term, plus various types of $[-1,1]$ neighbourhood terms. This leads to the update equation

$$x_{y(x)}^{\prime(new)} = x_{y(x)}^{\prime(old)} + 0.1 \left( x - x_{y(x)}^{\prime(old)} \right)$$

$$x_{y(x)\pm1}^{\prime(new)} = x_{y(x)\pm1}^{\prime(old)} + \epsilon' \left( x - x_{y(x)\pm1}^{\prime(old)} \right)$$

(4.2)

where one must choose signs consistently, and $\epsilon'$ takes one of three values: $\epsilon' = 0.025, 0.050, 0.075$.

B. Numerical experimental results

The final estimated values of $\alpha$ (i.e., after 500,000 updates) are shown in Table 3. These are the results of a single run, so we have not attempted to quote a standard error. However, the TM case consistently produces an $\alpha$ 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) could arise from a combination of factors involving nonvanishing next to leading order corrections, fluctuations in the training, unsophisticated estimation of $\alpha$, etc. It is not necessary for us to perform a sophisticated numerical simulation to detect the changes brought about by using minimum distortion rather than nearest neighbour encoding, because the differences between the TM and the TVQ entries in Table 3 are dramatic enough to swamp any uncertainties that arise from our crude numerical simulation.

We show in Figure 4 some plots of the estimate of $a$ as a function of (the logarithm to base 10 of) the numbers of training steps. These plots fall into two clearly separated categories which approach (as the number of training steps increases) the results tabulated in Table 3. It is clear from Figure 4 that we have not trained all the way to convergence, but the additional effort of either running our crude algorithm for longer or writing an cleverer algorithm is not justified by the simple point that we are trying to make. We see unequivocally that minimum distortion encoding leads to a dramatically different $\alpha$ from
Table I: Asymptotic power law $\alpha$ for various simple neighbourhood functions, showing both the topographic mapping case and the topographic vector quantiser case.

<table>
<thead>
<tr>
<th>$\epsilon'$</th>
<th>TM</th>
<th>TVQ</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.025</td>
<td>0.51</td>
<td>0.32</td>
</tr>
<tr>
<td>0.050</td>
<td>0.51</td>
<td>0.31</td>
</tr>
<tr>
<td>0.075</td>
<td>0.52</td>
<td>0.35</td>
</tr>
</tbody>
</table>

nearest neighbour encoding, and we see with less certainty (but still fairly convincingly) that asymptotically $p(x)$ has the predicted $P(x)^{1/3}$ power law dependence.

\[ \alpha = \frac{1}{3} \frac{(2n+1)^2}{(n+1)^2} \]

is derived for a scalar quantiser using nearest neighbour encoding, where $n$ is the half-width of a uniform symmetric topographic neighbourhood. We obtain the result $\alpha = \frac{1}{3}$ for a scalar quantiser using minimum distortion encoding, both for the uniform symmetric neighbourhood case and for the more general symmetric monotonically decreasing neighbourhood.

We may replace a standard topographic mapping by an equivalent topographic vector quantiser by merely changing the encoding scheme from nearest neighbour to minimum distortion. The distortion that is caused to a given input is thus calculated as a weighted average over the distortions that would have arisen from each of the code vectors in the topographic neighbourhood of the encoded input. Evidently, this is a more complicated scheme than nearest neighbour encoding, but it does lead to a simpler optimisation problem (i.e., minimum $L_2$ distortion).

It is important to note some restrictions that apply to our new theoretical result (see the Appendix): one must use a topographic neighbourhood that is a decreasing function of distance. As one approaches a uniform topographic neighbourhood function one must use ever smaller updates in order not to destabilise the code vectors. We have not found this to be a restriction in practice.

In practical applications, both nearest neighbour and minimum distortion encoding will lead to topographic mappings that resemble each other superficially except for the different asymptotic power law in $p(x) \propto P(x)^{1/3}$. The only circumstance where one can be certain that minimum distortion encoding is the correct method to use is when one needs to optimise a codebook to minimise the $L_2$ distortion that arises after passing the code index through a noisy channel [7]. However, in less well defined problems the decision about which method is best to use is unclear.

In cases where it does not matter which encoding method is used, and where it is required to estimate $P(x)$ from the unknown positions of the code vectors, minimum distortion encoding has a distinct advantage, because $p(x) \propto P(x)^{1/3}$ irrespective of the shape of the topographic neighbourhood function (subject to mild constraints). Thus $p(x)$ can be estimated from the local density of code vector positions, whence $P(x)$ can be estimated using $P(x) \propto p(x)^3$ without having to worry about side effects that arise from the presence of a topographic neighbourhood function.

Another advantage of the minimum distortion encoding approach is that it is likely to benefit directly from new results in vector quantiser theory, simply because it is formulated in the language of vector quantisation. For instance, in [8] a coarse-to-fine strategy is suggested for training the codebook of a vector quantiser, which results in a much more efficient algorithm than the brute-force approach of optimising a fully populated codebook. This idea can be carried over directly to a topographic vector quantiser and by analogy can be applied to train topo-
graphic mappings more efficiently than hitherto, as we showed in [9].

Other techniques may be transferred from vector quantisers, via topographic vector quantisers, to topographic mappings. For instance, multilevel (or hierarchical) codebooks that speed up the search time for locating the nearest neighbour from $O(n_{cv})$ to $O(\log n_{cv})$ can be trained.

Appendix A: Appendix: Stability Problems

If we implement our TVQ model using the minimum distortion encoding prescription as in Equation 2.5 and Equation 2.6 with the $[-n, +n]$ neighbourhood defined in Equation 2.12, we rapidly discover that as training proceeds the CVs tend to collapse into clusters, each of which contains $2n + 1$ CVs. Unfortunately, this type of clustering violates our assumption that we can discard next to leading order terms in our derivation. For instance, the higher derivative terms in Equation 2.14 should not be ignored.

It is easy to convince oneself how this type of observed clustering behavior could arise. Consider the simple case of a uniform input probability density $P(x) = P = \text{constant}$ with a $[-n, +n]$ neighbourhood as defined in Equation 2.12. There are two cases to consider.

If we assume that the CVs are uniformly spaced with a separation $a$, then the Euclidean distortion $D_2$ associated with a single CV (see Equation 2.7) reduces to

$$D_2 = P \int_{-n/2}^{+n/2} dx \sum_{k=-n}^{+n} (x - ka)^2$$

$$= Pa^3 \left[ \sum_{k=-n}^{+n} \frac{1}{6} + 6 \sum_{k=1}^{n} k^2 \right]$$

(A1)

where we use $\sum_{k=1}^{n} k^2 = \frac{1}{6} n(n + 1)(2n + 1)$.

If we assume that the CVs are clustered into sets of $2n + 1$ CVs superimposed on each other and that these clusters are uniformly spaced with a separation $(2n + 1)a$ (to ensure that the overall density of CVs is the same as in the previous case), then the Euclidean distortion $D_2$ associated with a single cluster reduces to

$$D_2 = P \int_{-(2n+1)/2}^{+(2n+1)/2} dx x^2$$

$$= Pa^3 (2n + 1)^3$$

(A2)

where we have made use of the normalisation condition $\sum_{k=-n}^{+n} \pi_k = 2n + 1$ to ensure compatibility with Equation A1 and Equation A2. This change does not affect the clustered case in Equation A1 because the distortion process has no affect there, but it modifies the uniform case in Equation A1 to become

$$D_2 = \frac{Pa^3 \left( 2n + 1 + 24 \sum_{k=1}^{n} k^2 \pi_k \right)}{12}$$

(A3)

where we impose the normalisation condition $\sum_{k=1}^{n} \pi_k = 2n + 1$ to ensure compatibility with Equation A1 and Equation A2. This change does not affect the clustered case in Equation A1 because the distortion process has no affect there, but it modifies the uniform case in Equation A1 to become

$$D_2 = \frac{Pa^3 \left( 2n + 1 + 24 \sum_{k=1}^{n} k^2 \pi_k \right)}{12}$$

(A4)

Note that all of this distortion is associated with just one of the $2n + 1$ CVs in the cluster (the central CV, in fact), while the remaining $2n$ CVs act as a buffer of $n$ CVs on either side of the central CV. This prevents the code index distortion process from moving the code index into one of the two adjacent clusters of $2n + 1$ CVs, which would have caused a large distortion to occur. Thus the encoding process maps only to the central CV in each cluster, so that subsequently the code index distortion process has no observable effect.

The uniform case in Equation A1 and the clustered cases in Equation A2 both lead to the same overall Euclidean distortion $D_2$ when all CVs are accounted for. This is a counterexample to our assumption that the optimal CV density is a smoothly varying function of position, because we have presented two different solutions to the problem of minimising $D_2$, one of which does not have a smooth CV density.

We may slightly generalise the above result by modifying Equation 2.12 to define a symmetrical neighbourhood function as

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

(A3)

where we impose the normalisation condition $\sum_{k=-n}^{+n} \pi_k = 2n + 1$ to ensure compatibility with Equation A1 and Equation A2. This change does not affect the clustered case in Equation A1 because the distortion process has no affect there, but it modifies the uniform case in Equation A1 to become

$$D_2 = \frac{Pa^3 \left( 2n + 1 + 24 \sum_{k=1}^{n} k^2 \pi_k \right)}{12}$$

(A4)

where we have made use of the normalisation condition. As expected, Equation A4 reduces to Equation A1 in the case $\pi_k = 1, k = 1, 2, \ldots, n$.

By inspecting Equation A4 we see that $D_2$ is smaller for uniform spacing than for clustered spacing when $\pi_k$ is chosen in such a way that it suppresses the contribution from large values of $k$ in the summation; these terms would otherwise dominate the summation owing to the factor $k^2$. This amounts to saying that a “tapered” neighbourhood function ensures the stability of uniformly spaced CVs with respect to clustered CVs.

Although the arguments here have been based entirely on the assumption that $P(x) = P = \text{constant}$, we assume that the same general conclusion may be carried over to the more general case of a spatially varying $P(x)$. This extension is broadly valid provided that $P(x)$ varies slowly over lengths of the order of the distance between adjacent CVs. By performing some simple numerical investigations one indeed discovers that the use of tapered neighbourhood functions stabilises the training of our TVQ model.


Trans. Inform. Theory, 6(1), 7-12.


