This paper was submitted to Neural Computation on 10th May 1993

A Bayesian Analysis of Self-Organising Maps

Stephen P Luttrell
Adaptive Systems Theory Section
Defence Research Agency
St Andrews Rd, Malvern,
Worcestershire, WR14 3PS,
United Kingdom
luttrell@signal.dra.hmg.gb

Abstract

In this paper Bayesian methods are used to analyse some of the properties of a special type of Markov chain. The forward transitions through the chain are followed by inverse transitions (using Bayes' theorem) backwards through a copy of the same chain; this is called a folded Markov chain. If an appropriately defined Euclidean error (between the original input and its "reconstruction" via Bayes' theorem) is minimised in the space of Markov chain transition probabilities, then the familiar theories of both vector quantisers and self-organising maps emerge. This approach is also used to derive the theory of self-supervision, in which the higher layers of a multi-layer network supervise the lower layers, even though overall there is no external teacher.

1. Introduction

Self-organised maps (SOM) have been widely studied since the publication of [1]. There have been many attempts to relate the training algorithm to an error minimisation problem [2,3,4,5,6]. These studies used an error function that was proportional to a weighted average of the Euclidean error between the input vector and each of the reference vectors in the SOM neighbourhood of the winning vector. This type of error function was also introduced in a non-neural context to design an optimal vector quantiser (VQ) codebook for encoding data for transmission along a noisy channel [7,8,9], which also happened to be the author's interpretation of SOMs in [2,3,4,5].

It was noted in [2,4,5,9] that the above Euclidean error function is not minimised when the nearest neighbour prescription is used. Further studies using the Euclidean error function have shown that exact minimisation requires a new type of winner-take-all prescription to be used; the nearest neighbour prescription must be replaced by the so-called minimum distortion prescription, in which the choice of winner is influenced by the reference vectors to which it is connected by the SOM neighbourhood function. The minimum distortion

© Crown Copyright 1993
Defence Research Agency
Farnborough, Hants, GU14 6TD, U.K.

prescription was used in [10] to study the equilibrium density of reference vectors, where it was reported that the result was insensitive to the choice of neighbourhood function used in the SOM. The corresponding results using a standard SOM with the nearest neighbour prescription appeared in [11], where it was shown that a neighbourhood-sensitive density of reference vectors emerged.

Hierarchcical multi-layer SOMs were studied in [12,13]. For instance, a 2-layer version of this type of network splits the input space into a number of lower dimensional subspaces, and trains a SOM in each subspace. Simultaneously, another SOM is trained on the outputs of the above SOMs to produce the final network output. This approach can be cascaded to more stages if required. This type of multi-layer SOM can be refined by introducing an error function which measures the average Euclidean error between the input vector and a vector that is constructed from the final network output [12,13]. This forces the optimisation of the layers to be tied together in such a way that they have to be trained co-operatively. For instance, in a 2-layer network the outputs from the SOMs in the first layer have to be matched to the capabilities of the SOM in the second layer; this is achieved by sending back-propagation signals from the second layer to the first layer. However, these back-propagation signals are not derived from an external supervisor, so this type of training algorithm is called "self-supervised".

A unified framework is needed to combine these various aspects of SOM theory. The best existing framework is vector quantisation for noisy channels [2,3,5,6,7,8,9], but this approach is limited because it is assumes a winner-take-all network behaviour, and thus can not describe more sophisticated types of SOM in which there is more than one winner. The sought after theory must be backwardly compatible with the above VQ approach, yet offer a smooth upgrade path to more sophisticated SOMs.

The principal new result that is contained in this paper is a Bayesian derivation of the properties of SOMs. Thus a Markov chain of probabilistic transformations of the input vector[1] is inverted by sending the (probabilistic) output of the chain back through the inverse probabilistic transformations (derived from Bayes' theorem) to eventually re-emerge as a (probabilistic) reconstructed version of the input vector. This is called a folded Markov chain (FMC). This FMC is then optimised by minimising the average Euclidean error between the input vector and its reconstruction. Various constraints can be placed on this optimisation. For instance, if a 2-stage FMC (i.e. 2 stages of probabilistic transformation) is considered, and only its first stage is optimised, then the theory of vector quantisation for noisy channels (and hence SOMs) emerges. Alternatively, if the state space of each stage of a 2-stage FMC is split into two lower dimensional subspaces, then the theory of self-supervision emerges.

The advantage of the FMC approach is that its basic language is probabilistic, which reduces in special cases to delta functions (Dirac and/or Kronecker) to yield winner-take-all behaviour. FMCs are very amenable to theoretical analysis, so the properties of SOMs (of the winner-take-all variety) can be readily investigated by pretending that they are in

[1] This is not to be confused with a Markov chain analysis of the convergence properties of the training algorithm.

fact probabilistic (i.e. FMCs), and taking the limit of a winner-take-all network only at the end of the calculation.

The structure of this paper is as follows. In section 2 the the idea of an FMC will be introduced, where Bayes' theorem is used to invert the Markov chain of probabilistic transformations. In Section 3 the relationship between FMCs and VQs and SOMs is derived, and it is shown how a 1-stage (or 2-stage) FMC contains a VQ (or SOM) as a special case. In Section 4 these results are extended to the case of a pair of coupled FMCs, and it is shown how self-supervision emerges naturally. In the first appendix the relationship between the continuum notation that is used in this paper and the more usual discrete notation is explained. In the second appendix a more complete and technically rigorous derivation of the results of Section 3 is presented, for completeness. The main new results are contained in Section 3 and Section 4.

2. Basic Theory

The notation that is used for probabilities is very carefully chosen so as to avoid ambiguities that could arise if the notation $P(\cdots)$ were used blindly to denote "the probability density of ...". However, occasional use of the ambiguous $P(\cdots)$ notation is made where there is no possibility of an ambiguity arising. Also, a careful distinction is drawn between the notation that is used for inverse probabilities (obtained from Bayes' theorem), and the notation that is used for forward probabilities; the former always appear with a tilde over the *P*, which thus appears as $\tilde{P}$.

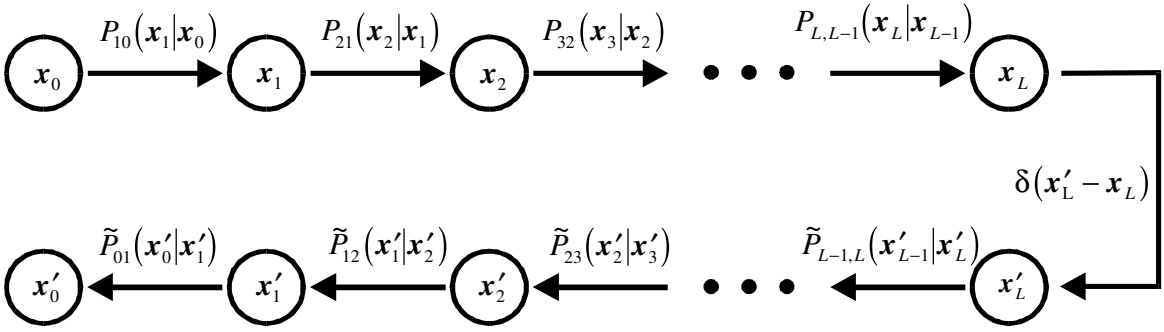

Figure 1. A folded Markov chain (i.e. both the forwards and backwards directions are represented). The top (bottom) half of the diagram represents the forward (backward) pass through the chain. The conditional probability $P_{k+1,k}(\boldsymbol{x}_{k+1}|\boldsymbol{x}_k)$ is used as a probabilistic transformation that generates the probable states of layer *k*+1 from the state of layer *k*, and the conditional probability $\tilde{P}_{k,k+1}(\boldsymbol{x}'_k|\boldsymbol{x}'_{k+1})$ is similarly used to generate the probable states of layer *k* from the state of layer *k*+1; this is also referred to generically as stage *k* of the FMC. These two conditional probabilities are related by Bayes' theorem.

Also the state space(s) will be assumed to be continuous, except where otherwise noted. This is because the corresponding discrete space results can be written down by inspection of the continuum results, and because the meaning of a continuum calculation is usually more transparent than its discrete counterpart.

The type of Markov chain that will be considered is shown in Figure 1.

This is a folded Markov chain (FMC) which performs an L-stage transformation of an input vector $\boldsymbol{x}_0$ to an output vector $\boldsymbol{x}_L$ (via L-1 intermediate vectors $\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_{L-1}$), and then performs the Bayes inverse transformation to arrive eventually at a reconstructed input vector $\boldsymbol{x}_0'$. The delta function $\delta(\boldsymbol{x}_L' - \boldsymbol{x}_L)$ is used to ensure that $\boldsymbol{x}_L' = \boldsymbol{x}_L$.

The conditional probabilities are related by Bayes' theorem as follows

$$\tilde{P}_{k,k+1}(\boldsymbol{x}_k|\boldsymbol{x}_{k+1})P_{k+1}(\boldsymbol{x}_{k+1}) = P_{k+1,k}(\boldsymbol{x}_{k+1}|\boldsymbol{x}_k)P_k(\boldsymbol{x}_k) \tag{1}$$

where $P_k(\boldsymbol{x}_k)$ denotes the marginal probability (density) of $\boldsymbol{x}_k$. In order to construct any joint probability in the FMC system it is both necessary and sufficient to specify $P_0(\boldsymbol{x}_0)$ and the L transformations $P_{10}(\boldsymbol{x}_1|\boldsymbol{x}_0), P_{21}(\boldsymbol{x}_2|\boldsymbol{x}_1), \cdots, P_{L,L-1}(\boldsymbol{x}_L|\boldsymbol{x}_{L-1})$.

Thus

$$\begin{aligned} P(\boldsymbol{x}_0, \boldsymbol{x}_1, \cdots, \boldsymbol{x}_L; \boldsymbol{x}_L', \cdots, \boldsymbol{x}_1', \boldsymbol{x}_0') = {} & P_0(\boldsymbol{x}_0)P_{10}(\boldsymbol{x}_1|\boldsymbol{x}_0)P_{21}(\boldsymbol{x}_2|\boldsymbol{x}_1)\cdots P_{L,L-1}(\boldsymbol{x}_L|\boldsymbol{x}_{L-1}) \\ & \times\delta(\boldsymbol{x}_L' - \boldsymbol{x}_L) \\ & \times\tilde{P}_{L-1,L}(\boldsymbol{x}_{L-1}'|\boldsymbol{x}_L')\cdots\tilde{P}_{12}(\boldsymbol{x}_1'|\boldsymbol{x}_2')\tilde{P}_{01}(\boldsymbol{x}_0'|\boldsymbol{x}_1') \end{aligned} \tag{2}$$

The marginal probability $P_k(\boldsymbol{x}_k)$ is obtained by integrating over all variables other than $\boldsymbol{x}_k$, which yields

$$P_k(\boldsymbol{x}_k) = \int d\boldsymbol{x}_0\, d\boldsymbol{x}_1 \cdots d\boldsymbol{x}_{k-1}\; P_0(\boldsymbol{x}_0)P_{10}(\boldsymbol{x}_1|\boldsymbol{x}_0)P_{21}(\boldsymbol{x}_2|\boldsymbol{x}_1)\cdots P_{k,k-1}(\boldsymbol{x}_k|\boldsymbol{x}_{k-1}) \tag{3}$$

Note that Bayes' theorem guarantees that the marginal probabilities of $\boldsymbol{x}_k$ and $\boldsymbol{x}_k'$ are the same.

In this paper attention will be restricted to FMCs with 1 or 2 stages only (i.e. $L = 1$ or 2). Furthermore, whenever the FMC is to be optimised, only its first stage will be optimised.

3. 1 and 2-Stage Folded Markov Chains

FMCs are especially interesting in adaptive network design theory, because they turn out to contain well-known systems as special cases: a VQ is a special case of a 1-stage FMC, and an SOM is a special case of a 2-stage FMC.

The success of these derivations relies on the similarities that exist between the following two situations:

1. The direct/inverse probabilities that occur in Bayes' theorem as applied to a Markov chain, which might be used in the analysis of scattering problems in layered media, for instance. The sequence of processing operations is source→scattering→inverse scattering→reconstruction.

2. The encoding/decoding operations that occur in VQs and SOMs, which might be used in the analysis of information transmission down a (noisy) communication channel, for instance. The sequence of processing operations is source→encode→decode→ reconstruction.

3.1. 1-Stage Folded Markov Chain: Vector Quantiser

A 1-stage FMC is shown in Figure 2.

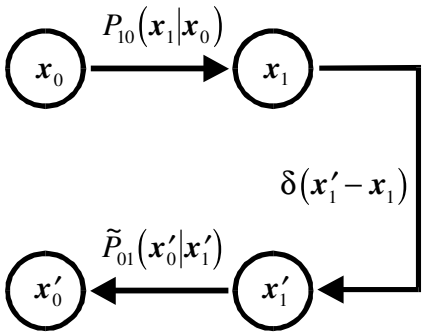

Figure 2. A 1-stage folded Markov chain. When $P_{01}(\boldsymbol{x}_1|\boldsymbol{x}_0)=\delta(\boldsymbol{x}_1-\boldsymbol{x}_1(\boldsymbol{x}_0))$, and D is minimised with respect to $\boldsymbol{x}_1(\boldsymbol{x}_0)$, this reduces to a vector quantiser with an infinite number of reference vectors (i.e. continuum limit). See the appendix for a detailed discussion of the relationship between the continuum and discrete cases. Bayes' theorem ensures that $\tilde{P}_{10}(\boldsymbol{x}_0'|\boldsymbol{x}_1')$ cannot be varied independently of $P_{01}(\boldsymbol{x}_1|\boldsymbol{x}_0)$.

The derivation starts from the definition of the average Euclidean error D between $\boldsymbol{x}_0$ and $\boldsymbol{x}_0'$ in a 1-stage FMC.

$$D=\int d\boldsymbol{x}_0\; d\boldsymbol{x}_1\; d\boldsymbol{x}_0'\; d\boldsymbol{x}_1'\; P_0(\boldsymbol{x}_0)P_{10}(\boldsymbol{x}_1|\boldsymbol{x}_0)\delta(\boldsymbol{x}_1'-\boldsymbol{x}_1)\tilde{P}_{01}(\boldsymbol{x}_0'|\boldsymbol{x}_1')\,\|\boldsymbol{x}_0'-\boldsymbol{x}_0\|^2 \tag{4}$$

After integrating out the (irrelevant) $\boldsymbol{x}_1'$ using the delta function $\delta(\boldsymbol{x}_1'-\boldsymbol{x}_1)$, using Bayes' theorem in the form $\tilde{P}_{01}(\boldsymbol{x}_0|\boldsymbol{x}_1)P_1(\boldsymbol{x}_1)=P_{10}(\boldsymbol{x}_1|\boldsymbol{x}_0)P_0(\boldsymbol{x}_0)$ (i.e. Equation 1 with k=0), and rearranging the order of the integrations, this reduces to

$$D=\int d\boldsymbol{x}_1\; P_1(\boldsymbol{x}_1)\int d\boldsymbol{x}_0\; d\boldsymbol{x}'_0\; \tilde{P}_{01}(\boldsymbol{x}_0|\boldsymbol{x}_1)\;\tilde{P}_{01}(\boldsymbol{x}_0'|\boldsymbol{x}_1)\,\|\boldsymbol{x}_0'-\boldsymbol{x}_0\|^2 \tag{5}$$

The next step is to expand the norm as $\|\boldsymbol{x}_0\|^2+\|\boldsymbol{x}_0'\|^2-2\boldsymbol{x}_0.\boldsymbol{x}_0'$ and perform the double integration over $\boldsymbol{x}_0$ and $\boldsymbol{x}_0'$ to obtain

$$D=2\int d\boldsymbol{x}_1\; P_1(\boldsymbol{x}_1)\left(\left\langle\|\boldsymbol{x}_0\|^2\right\rangle_{\tilde{P}_{01}(\boldsymbol{x}_0|\boldsymbol{x}_1)}-\left\|\left\langle\boldsymbol{x}_0\right\rangle_{\tilde{P}_{01}(\boldsymbol{x}_0|\boldsymbol{x}_1)}\right\|^2\right) \tag{6}$$

where the notation $\langle\cdots\rangle_P$ has been used to denote the average of $(\cdots)$ under probability P. Then the standard result $\left\langle\|\cdots\|^2\right\rangle-\|\langle\cdots\rangle\|^2\equiv\left\langle\|(\cdots)-\langle\cdots\rangle\|^2\right\rangle$ is used to express this as

$$D = 2\int d\boldsymbol{x}_1 \; P_1(\boldsymbol{x}_1)\left\langle \left\| \boldsymbol{x}_0 - \langle \boldsymbol{x}_0 \rangle_{\tilde{P}_{01}(\boldsymbol{x}_0|\boldsymbol{x}_1)} \right\|^2 \right\rangle_{\tilde{P}_{01}(\boldsymbol{x}_0|\boldsymbol{x}_1)} \tag{7}$$

or, more explicitly, as

$$D = 2\int d\boldsymbol{x}_1 \; P_1(\boldsymbol{x}_1) \int d\boldsymbol{x}_0 \; \tilde{P}_{01}(\boldsymbol{x}_0|\boldsymbol{x}_1) \left\| \boldsymbol{x}_0 - \langle \boldsymbol{x}_0 \rangle_{\tilde{P}_{01}(\boldsymbol{x}_0|\boldsymbol{x}_1)} \right\|^2 \tag{8}$$

The derivation of Equation 8 from Equation 5 is well-known; it says that the average Euclidean error between pairs of vectors drawn independently from $\tilde{P}_{01}(\boldsymbol{x}_0|\boldsymbol{x}_1)$ is twice the variance of vectors drawn from $\tilde{P}_{01}(\boldsymbol{x}_0|\boldsymbol{x}_1)$. Finally, Bayes' theorem (i.e. Equation 1 with k=0) can be used to obtain the required result.

$$D = 2\int d\boldsymbol{x}_0 \; P_0(\boldsymbol{x}_0) \int d\boldsymbol{x}_1 \; P_{10}(\boldsymbol{x}_1|\boldsymbol{x}_0) \left\| \boldsymbol{x}_0 - \langle \boldsymbol{x}_0 \rangle_{\tilde{P}_{01}(\boldsymbol{x}_0|\boldsymbol{x}_1)} \right\|^2 \tag{9}$$

Equation 9 has all of the right structure to relate FMCs to VQs. It has a source of input vectors $P_0(\boldsymbol{x}_0)$, an encoder $P_{10}(\boldsymbol{x}_1|\boldsymbol{x}_0)$ that permits more than one winner, a reference vector $\langle \boldsymbol{x}_0 \rangle_{\tilde{P}_{01}(\boldsymbol{x}_0|\boldsymbol{x}_1)}$ with which to compare the input vector to compute a Euclidean distortion. The only differences between this FMC and a standard VQ are:

1. $P_{10}(\boldsymbol{x}_1|\boldsymbol{x}_0)$ is not a winner-take-all encoder. Each input vector $\boldsymbol{x}_0$ is transformed into each possible output vector $\boldsymbol{x}_1$ with probability $P_{10}(\boldsymbol{x}_1|\boldsymbol{x}_0)$. In the language of neural networks, it is as if each possible output vector had an "activity" specified by $P_{10}(\boldsymbol{x}_1|\boldsymbol{x}_0)$. A winner-take-all would result if $P_{10}(\boldsymbol{x}_1|\boldsymbol{x}_0)$ were replaced by a probability whose mass was concentrated all at one point; this would be a delta function $\delta(\boldsymbol{x}_1 - \boldsymbol{x}_1(\boldsymbol{x}_0))$.

2. The reference vector $\langle \boldsymbol{x}_0 \rangle_{\tilde{P}_{01}(\boldsymbol{x}_0|\boldsymbol{x}_1)}$ is dependent on the encoder $P_{10}(\boldsymbol{x}_1|\boldsymbol{x}_0)$; they are related by Bayes' theorem (i.e. Equation 1 with k=0). In a VQ the reference vector and the encoder are also related because the encoder is a nearest neighbour prescription, which in turn depends on the location of the reference vectors. It is not at all obvious that these two pictures (the FMC and the VQ) are related.

Now consider a modified form of D

$$D = 2\int d\boldsymbol{x}_0 \; P_0(\boldsymbol{x}_0) \int d\boldsymbol{x}_1 \; P_{10}(\boldsymbol{x}_1|\boldsymbol{x}_0) \left\| \boldsymbol{x}_0 - \boldsymbol{x}_0'(\boldsymbol{x}_1) \right\|^2 \tag{10}$$

where $\langle \boldsymbol{x}_0 \rangle_{\tilde{P}_{01}(\boldsymbol{x}_0|\boldsymbol{x}_1)}$ is replaced by the function $\boldsymbol{x}_0'(\boldsymbol{x}_1)$. Functionally differentiate this expression for D with respect to $\boldsymbol{x}_0'(\boldsymbol{x}_1)$ to obtain

$$\frac{\delta D}{\delta \boldsymbol{x}_0'(\boldsymbol{x}_1)} = -4\int d\boldsymbol{x}_0 \; P_0(\boldsymbol{x}_0) \; P_{10}(\boldsymbol{x}_1|\boldsymbol{x}_0) \left(\boldsymbol{x}_0 - \boldsymbol{x}_0'(\boldsymbol{x}_1)\right) \tag{11}$$

The stationary point $\delta D / \delta \boldsymbol{x}_0'(\boldsymbol{x}_1) = 0$ is obtained when $\boldsymbol{x}_0'(\boldsymbol{x}_1)$ satisfies

$$\boldsymbol{x}_0'(\boldsymbol{x}_1) = \frac{\int d\boldsymbol{x}_0 \; P_0(\boldsymbol{x}_0) P_{10}(\boldsymbol{x}_1|\boldsymbol{x}_0)\, \boldsymbol{x}_0}{\int d\boldsymbol{x}_0 \; P_0(\boldsymbol{x}_0) P_{10}(\boldsymbol{x}_1|\boldsymbol{x}_0)} \tag{12}$$

By using Bayes' theorem (i.e. Equation 1 with k=0) this stationarity condition reduces to

$$\boldsymbol{x}_0'(\boldsymbol{x}_1) = \langle \boldsymbol{x}_0 \rangle_{\tilde{P}_{01}(\boldsymbol{x}_0|\boldsymbol{x}_1)} \tag{13}$$

So the modified expression for D in Equation 10 reduces to the original expression for D in Equation 9 *provided that* $\boldsymbol{x}_0'(\boldsymbol{x}_1)$ is chosen to minimise D. This is a major simplification, because the coupling (via Bayes' theorem) between $P_{10}(\boldsymbol{x}_1|\boldsymbol{x}_0)$ and $\langle \boldsymbol{x}_0 \rangle_{\tilde{P}_{01}(\boldsymbol{x}_0|\boldsymbol{x}_1)}$ that appeared in Equation 9 can now safely be ignored by the simple trick of using Equation 10 instead (with the proviso that $\boldsymbol{x}_0'(\boldsymbol{x}_1)$ should always be optimised so as to minimise D).

At this point the minimisation of D (as written in Equation 10) with respect to $P_{10}(\boldsymbol{x}_1|\boldsymbol{x}_0)$ should be considered. However, this derivation is messy, so it is presented in the appendix. Instead, a simplified (and strictly incomplete) version of this derivation is presented below.

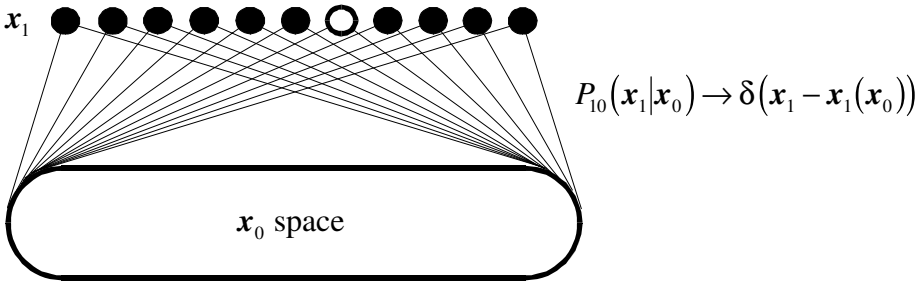

Figure 3. A discrete VQ represented as a network. The bottom layer is the input space (or $\boldsymbol{x}_0$), the top layer is the output space (or $\boldsymbol{x}_1$), and the connections between the two represent a soft encoding operation $P_{10}(\boldsymbol{x}_1|\boldsymbol{x}_0)$, akin to that used in [14]. A winner-take-all VQ uses an encoder of the form $P_{10}(\boldsymbol{x}_1|\boldsymbol{x}_0) = \delta(\boldsymbol{x}_1 - \boldsymbol{x}_1(\boldsymbol{x}_0))$. The input and output layers of this network are represented in different ways: the input layer is the vector $\boldsymbol{x}_0$, whereas each node of the the output layer corresponds to exactly one possible state of the vector $\boldsymbol{x}_1$. An example of the winner in the output layer is represented by the open circle.

Make the following replacement in Equation 10 (which basically replaces $P_{10}(\boldsymbol{x}_1|\boldsymbol{x}_0)$ by the result that is derived at some length in the appendix)

$$P_{10}(\boldsymbol{x}_1|\boldsymbol{x}_0) \to \delta(\boldsymbol{x}_1 - \boldsymbol{x}_1(\boldsymbol{x}_0)) \tag{14}$$

which converts $P_{10}(\boldsymbol{x}_1|\boldsymbol{x}_0)$ into a winner-take-all encoder. With these replacements D becomes

$$D = 2\int d\boldsymbol{x}_0 \; P_0(\boldsymbol{x}_0) \left\| \boldsymbol{x}_0 - \boldsymbol{x}'_0(\boldsymbol{x}_1(\boldsymbol{x}_0)) \right\|^2 \tag{15}$$

which is exactly what would be written for the continuum version of a VQ (apart from the trivial overall factor of 2).

The network representation of a VQ is shown in Figure 3 (this is the discrete version of a VQ, not the continuum version), which should be compared with the FMC representation shown in Figure 2.

The gradient of D w.r.t. $\boldsymbol{x}'_0(\boldsymbol{x}_1)$ is given by the functional derivative $\delta D / \delta \boldsymbol{x}'_0(\boldsymbol{x}_1)$

$$\frac{\delta D}{\delta \boldsymbol{x}'_0(\boldsymbol{x}_1)} = -4\int d\boldsymbol{x}_0 \; P_0(\boldsymbol{x}_0)\, \delta\left(\boldsymbol{x}_1 - \boldsymbol{x}_1(\boldsymbol{x}_0)\right)\left(\boldsymbol{x}_0 - \boldsymbol{x}'_0(\boldsymbol{x}_1)\right) \tag{16}$$

By inspecting the dependence of Equation 15 on $\boldsymbol{x}_1(\boldsymbol{x}_0)$, and by setting $\delta D / \delta \boldsymbol{x}'_0(\boldsymbol{x}_1)$ in Equation 16 to zero, the following batch training prescription for minimising D can be obtained

$$\begin{aligned} \boldsymbol{x}_1(\boldsymbol{x}_0) &= \underset{\boldsymbol{x}_1}{\arg\min} \left(\boldsymbol{x}_0 - \boldsymbol{x}'_0(\boldsymbol{x}_1)\right) \\ \boldsymbol{x}'_0(\boldsymbol{x}_1) &= \frac{\int d\boldsymbol{x}_0 \; P_0(\boldsymbol{x}_0)\delta\left(\boldsymbol{x}_1 - \boldsymbol{x}_1(\boldsymbol{x}_0)\right)\boldsymbol{x}_0}{\int d\boldsymbol{x}_0 \; P_0(\boldsymbol{x}_0)\delta\left(\boldsymbol{x}_1 - \boldsymbol{x}_1(\boldsymbol{x}_0)\right)} \end{aligned} \tag{17}$$

These results merit the following remarks:

1. The function $\boldsymbol{x}'_0(\boldsymbol{x}_1)$ can be interpreted as the continuum version of a VQ codebook, where $\boldsymbol{x}_1$ is the (continuum) code index and $\boldsymbol{x}'_0(\boldsymbol{x}_1)$ is the code vector associated with that index. The result for $\boldsymbol{x}'_0(\boldsymbol{x}_1)$ corresponds to Equation 12 with $P_{10}(\boldsymbol{x}_1|\boldsymbol{x}_0) \to \delta\left(\boldsymbol{x}_1 - \boldsymbol{x}_1(\boldsymbol{x}_0)\right)$, which is the well-known "centroiding" prescription for updating the code vectors after a batch of training data has been presented to a VQ, as used in the LBG algorithm [15].
2. The result for $\boldsymbol{x}_1(\boldsymbol{x}_0)$ is the "nearest neighbour" encoding prescription for encoding the input of a VQ.

An on-line training prescription can also be obtained to implement updates to $\boldsymbol{x}'_0(\boldsymbol{x}_1)$ after each input vector $\boldsymbol{x}_0$ is selected at random from $P_0(\boldsymbol{x}_0)$. The on-line prescription is

$$\boldsymbol{x}'_0(\boldsymbol{x}_1) \to \boldsymbol{x}'_0(\boldsymbol{x}_1) + \varepsilon\, \delta\left(\boldsymbol{x}_1 - \boldsymbol{x}_1(\boldsymbol{x}_0)\right)\left(\boldsymbol{x}_0 - \boldsymbol{x}'_0(\boldsymbol{x}_1)\right) \tag{18}$$

Note that the delta function permits non-zero updates only for $\boldsymbol{x}_1 = \boldsymbol{x}_1(\boldsymbol{x}_0)$; this is the continuum version of updating the nearest neighbour code vector towards the input vector. The relationship between this continuum result and the corresponding discrete result is discussed in the appendix.

This completes the demonstration that an optimal 1-stage FMC is a VQ. Note that the use of a Euclidean error is sufficient (but not necessary) for this result to emerge, and

furthermore that there are other choices of error measure for which this result does not emerge.

3.2. 2-Stage Folded Markov Chain: Self-Organising Map

The above derivation may be extended to a 2-stage FMC, as shown in Figure 4.

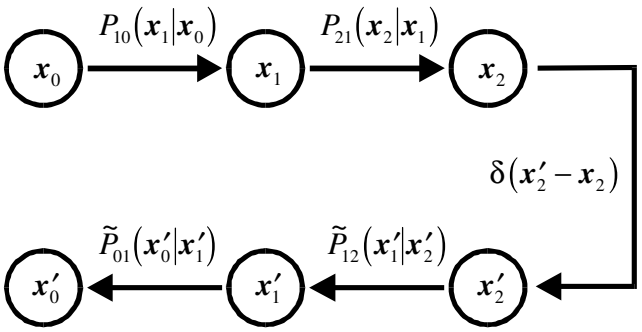

Figure 4. A 2-stage folded Markov chain. This is basically the same as Figure 2, except that $P_{21}(\boldsymbol{x}_2|\boldsymbol{x}_1)$ causes the information that flows through the folded Markov chain to be corrupted before it can begin its return journey.

Because the derivation is so similar to the case of a 1-stage FMC, only the first and last lines will be shown. For simplicity, the following notation will be used

$$\begin{aligned} P_{20}(\boldsymbol{x}_2|\boldsymbol{x}_0) &= \int d\boldsymbol{x}_1 \; P_{21}(\boldsymbol{x}_2|\boldsymbol{x}_1) P_{10}(\boldsymbol{x}_1|\boldsymbol{x}_0) \\ \tilde{P}_{02}(\boldsymbol{x}_0|\boldsymbol{x}_2) P_2(\boldsymbol{x}_2) &= P_{20}(\boldsymbol{x}_2|\boldsymbol{x}_0) P_0(\boldsymbol{x}_0) \end{aligned} \tag{19}$$

$P_{20}(\boldsymbol{x}_2|\boldsymbol{x}_0)$ is the probability (density) that $\boldsymbol{x}_2$ will be generated by $\boldsymbol{x}_0$, taking into account all of the possible values that the intermediate state $\boldsymbol{x}_1$ might take. $\tilde{P}_{02}(\boldsymbol{x}_0|\boldsymbol{x}_2)$ is the corresponding inverse probability that is obtained from Bayes' theorem.

Using this notation the expression for *D* becomes (compare Equation 4)

$$\begin{aligned} D = \int & d\boldsymbol{x}_0 \; d\boldsymbol{x}_1 \; d\boldsymbol{x}_2 \; d\boldsymbol{x}_0' \; d\boldsymbol{x}_1' \; d\boldsymbol{x}_2' \\ & \times P_0(\boldsymbol{x}_0) P_{10}(\boldsymbol{x}_1|\boldsymbol{x}_0) P_{21}(\boldsymbol{x}_2|\boldsymbol{x}_1) \delta(\boldsymbol{x}_2' - \boldsymbol{x}_2) \tilde{P}_{12}(\boldsymbol{x}_1'|\boldsymbol{x}_2') \tilde{P}_{01}(\boldsymbol{x}_0'|\boldsymbol{x}_1') \left\| \boldsymbol{x}_0' - \boldsymbol{x}_0 \right\|^2 \end{aligned} \tag{20}$$

which simplifies to (compare Equation 9)

$$D = 2 \int d\boldsymbol{x}_0 \; P_0(\boldsymbol{x}_0) \int d\boldsymbol{x}_2 \; P_{20}(\boldsymbol{x}_2|\boldsymbol{x}_0) \left\| \boldsymbol{x}_0 - \langle \boldsymbol{x}_0 \rangle_{\tilde{P}_{02}(\boldsymbol{x}_0|\boldsymbol{x}_2)} \right\|^2 \tag{21}$$

Consider a modified form of *D* (compare Equation 10)

$$D = 2 \int d\boldsymbol{x}_0 \; P_0(\boldsymbol{x}_0) \int d\boldsymbol{x}_2 \; P_{21}(\boldsymbol{x}_2|\boldsymbol{x}_0) \left\| \boldsymbol{x}_0 - \boldsymbol{x}_0'(\boldsymbol{x}_2) \right\|^2 \tag{22}$$

Set $\delta D / \delta \boldsymbol{x}_0'(\boldsymbol{x}_2) = 0$ to obtain (compare Equation 12 and Equation 13)

$$\boldsymbol{x}_0'(\boldsymbol{x}_2) = \frac{\int d\boldsymbol{x}_0 \; P_0(\boldsymbol{x}_0) P_{20}(\boldsymbol{x}_2|\boldsymbol{x}_0) \, \boldsymbol{x}_0}{\int d\boldsymbol{x}_0 \; P_0(\boldsymbol{x}_0) P_{20}(\boldsymbol{x}_2|\boldsymbol{x}_0)} = \langle \boldsymbol{x}_0 \rangle_{\tilde{P}_{02}(\boldsymbol{x}_0|\boldsymbol{x}_2)} \tag{23}$$

The modified expression for D in Equation 22 reduces to the original expression in Equation 21 *provided that* $\boldsymbol{x}_0'(\boldsymbol{x}_2)$ is chosen to minimise D, so Equation 22 will be used in preference to Equation 21 (with the proviso that $\boldsymbol{x}_0'(\boldsymbol{x}_2)$ should always be chosen to minimise D).

Make the replacement $P_{10}(\boldsymbol{x}_1|\boldsymbol{x}_0) \to \delta(\boldsymbol{x}_1 - \boldsymbol{x}_1(\boldsymbol{x}_0))$ which converts $P_{10}(\boldsymbol{x}_1|\boldsymbol{x}_0)$ into a winner-take-all encoder (see Equation 14, and the appendix for more details) to obtain D as (compare Equation 15)

$$D = 2\int d\boldsymbol{x}_0 \; P_0(\boldsymbol{x}_0) \int d\boldsymbol{x}_2 \; P_{21}(\boldsymbol{x}_2|\boldsymbol{x}_1(\boldsymbol{x}_0)) \left\|\boldsymbol{x}_0 - \boldsymbol{x}_0'(\boldsymbol{x}_2)\right\|^2 \tag{24}$$

The functional derivative $\delta D/\delta \boldsymbol{x}_0'(\boldsymbol{x}_2)$ is thus (compare Equation 16)

$$\frac{\delta D}{\delta \boldsymbol{x}_0'(\boldsymbol{x}_2)} = -4\int d\boldsymbol{x}_0 \; P_0(\boldsymbol{x}_0) P_{21}(\boldsymbol{x}_2|\boldsymbol{x}_1(\boldsymbol{x}_0)) (\boldsymbol{x}_0 - \boldsymbol{x}_0'(\boldsymbol{x}_2)) \tag{25}$$

which leads to the following batch training prescription for minimising D (compare Equation 17)

$$\begin{aligned} \boldsymbol{x}_1(\boldsymbol{x}_0) &= \underset{\boldsymbol{x}_1}{\arg\min} \int d\boldsymbol{x}_2 \; P_{21}(\boldsymbol{x}_2|\boldsymbol{x}_1) \left\|\boldsymbol{x}_0 - \boldsymbol{x}_0'(\boldsymbol{x}_2)\right\|^2 \\ \boldsymbol{x}_0'(\boldsymbol{x}_2) &= \frac{\int d\boldsymbol{x}_0 \; P_0(\boldsymbol{x}_0) P_{21}(\boldsymbol{x}_2|\boldsymbol{x}_1(\boldsymbol{x}_0)) \, \boldsymbol{x}_0}{\int d\boldsymbol{x}_0 \; P_0(\boldsymbol{x}_0) P_{21}(\boldsymbol{x}_2|\boldsymbol{x}_1(\boldsymbol{x}_0))} \end{aligned} \tag{26}$$

and the following on-line training prescription (compare Equation 18)

$$\boldsymbol{x}_0'(\boldsymbol{x}_2) \to \boldsymbol{x}_0'(\boldsymbol{x}_2) + \varepsilon \; P_{21}(\boldsymbol{x}_2|\boldsymbol{x}_1(\boldsymbol{x}_0)) (\boldsymbol{x}_0 - \boldsymbol{x}_0'(\boldsymbol{x}_2)) \tag{27}$$

These results correspond to the results that were reported in [2,4,5]. They can be interpreted as the generalisation of the VQ results to the case where the output of the VQ is corrupted by the action of $P_{21}(\boldsymbol{x}_2|\boldsymbol{x}_1)$ before Bayes' theorem is then used in an attempt to reconstruct the input vector.

This winner-take-all version of a 2-stage FMC turns out to be an SOM whose network representation is shown in Figure 3 (the discrete version of an SOM, not the continuum version), which should be compared with the FMC representation that is shown in Figure 4.

The SOM interpretation of these results for optimising a 2-stage FMC is as follows:

1. The function $\boldsymbol{x}_0'(\boldsymbol{x}_2)$ can be interpreted as the continuum version of the SOM reference vectors, where $\boldsymbol{x}_2$ is a (continuum) index and $\boldsymbol{x}_0'(\boldsymbol{x}_2)$ is the reference vector associated with that index. The batch update prescription for $\boldsymbol{x}_0'(\boldsymbol{x}_1)$ is a generalisation of the LBG "centroiding" prescription [15] which accounts for the effect of $P_{21}(\boldsymbol{x}_2|\boldsymbol{x}_1)$.

2. The result for $\boldsymbol{x}_1(\boldsymbol{x}_0)$ does not correspond to the winner-take-all prescription for selecting the nearest neighbour reference vector. Rather, it says that $\boldsymbol{x}_1(\boldsymbol{x}_0)$ is the value that $\boldsymbol{x}_1$ must take in order to ensure that the contribution to the distortion D

from input vectors in the infinitesimal neighbourhood of $\boldsymbol{x}_0$ is minimised. Thus the nearest neighbour encoding prescription has become a minimum distortion encoding prescription. This reduces to the nearest neighbour encoding prescription when $P_{21}(\boldsymbol{x}_2|\boldsymbol{x}_1) \to \delta(\boldsymbol{x}_2 - \boldsymbol{x}_1)$, as expected.

3. The on-line training prescription is the continuum version of the standard SOM training prescription, where $P_{21}(\boldsymbol{x}_2|\boldsymbol{x}_1)$ plays the role of the neighbourhood function. $P_{21}(\boldsymbol{x}_2|\boldsymbol{x}_1)$ also has this interpretation in the batch prescription.

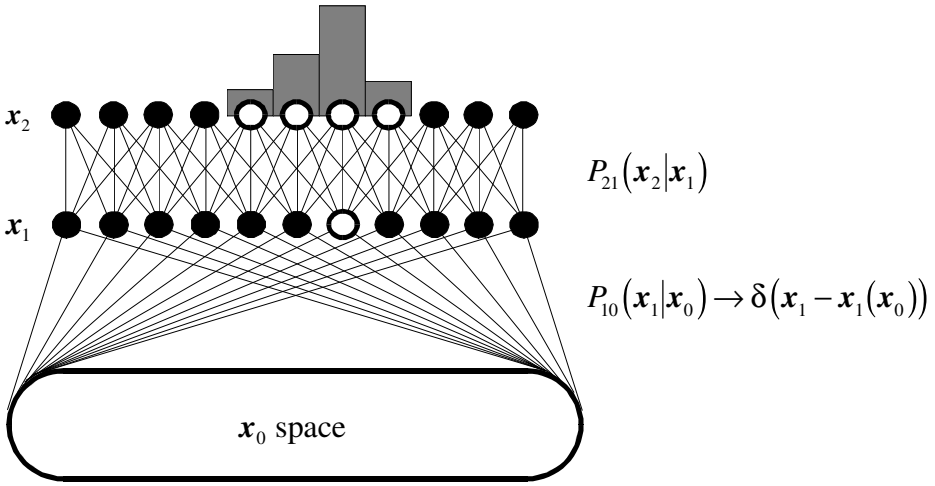

Figure 5. A discrete SOM represented as a network. This is the same as the VQ network in Figure 3 with an additional stage of processing $P_{21}(\boldsymbol{x}_2|\boldsymbol{x}_1)$ applied to its output layer. Each node in the hidden and output layer of this SOM network corresponds to exactly one possible state of the vector $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$, respectively. $P_{21}(\boldsymbol{x}_2|\boldsymbol{x}_1)$ serves as the SOM neighbourhood function by connecting together states of $\boldsymbol{x}_1$ so that they are ordered (across the page in this case). Note that the VQ network in Figure 3 does not have this ordering property, although the states of $\boldsymbol{x}_1$ (or nodes) are still drawn in an ordered fashion, for convenience. An example of a winner in the hidden layer is drawn as an open circle, as are each of the corresponding soft winners in the output layer. An example of the degree to which each node in the output layer is activated is indicated by a histogram, which records $P_{21}(\boldsymbol{x}_2|\boldsymbol{x}_1)$ for each possible state that $\boldsymbol{x}_2$ might take.

This completes the demonstration that an optimal 2-stage FMC is an SOM. Note that minimum distortion encoding is used, rather than nearest neighbour encoding, so this type of SOM is only an approximation to the standard SOM that was discussed in [1].

4. Coupled 2-Stage Folded Markov Chains

In [12,13] some interesting results were reported where the behaviour of a multi-layer SOM could be interpreted as if the higher network layers were supervising the lower layers, and the term "self-supervision" was introduced to describe this effect. The purpose of this section is to show how these results can be derived from the theory of FMCs.

4.1. Splitting the Markov Chain State Spaces

The derivation starts by splitting each state of a 2-stage FMC into two lower dimensional pieces, as shown in Figure 6.

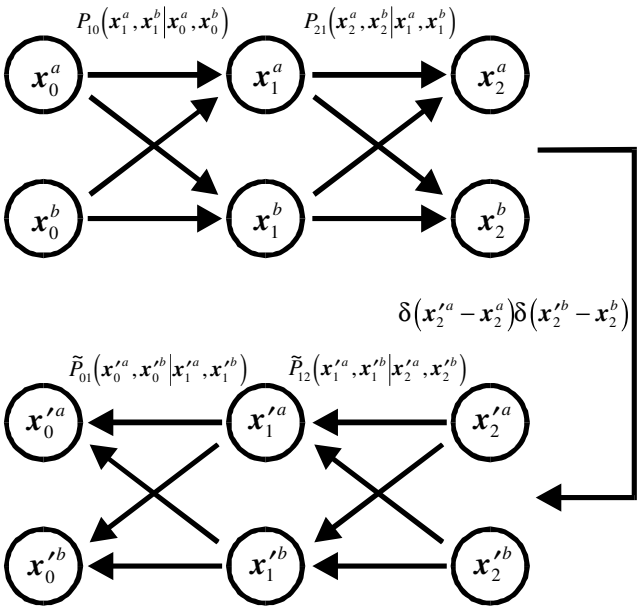

Figure 6. A 2-stage folded Markov chain with each state split into two lower dimensional pieces. This is basically the same as Figure 4, except that the states have been exploded to reveal their internal structure.

Figure 6 can be obtained by making the following changes to the notation in Figure 4

$$\begin{aligned} \boldsymbol{x}_0 &\to \left(\boldsymbol{x}_0^a,\boldsymbol{x}_0^b\right) & \boldsymbol{x}_0' &\to \left(\boldsymbol{x}_0'^a,\boldsymbol{x}_0'^b\right) \\ \boldsymbol{x}_1 &\to \left(\boldsymbol{x}_1^a,\boldsymbol{x}_1^b\right) & \boldsymbol{x}_1' &\to \left(\boldsymbol{x}_1'^a,\boldsymbol{x}_1'^b\right) \\ \boldsymbol{x}_2 &\to \left(\boldsymbol{x}_2^a,\boldsymbol{x}_2^b\right) & \boldsymbol{x}_2' &\to \left(\boldsymbol{x}_2'^a,\boldsymbol{x}_2'^b\right) \end{aligned} \tag{28}$$

so the FMC in Figure 6 is the same as the FMC in Figure 4, apart from the notation that is used to describe its state spaces. This tautology is motivated by the need to break up high dimensional spaces, such as might occur in image processing problems, into a number of coupled lower dimensional spaces. The coupled FMCs in Figure 6 are much easier to describe in approximate terms than the FMC in Figure 4. For instance, if the coupling between the FMCs is weak, then a series expansion about zero coupling strength could be attempted. The lowest order term in this expansion would correspond to a pair of

uncoupled FMCs, and the higher order terms would correspond to interactions between the FMCs.

If the new notation in Equation 28 is used together with the generalisation of Equation 14

$$P_{10}\left(\boldsymbol{x}_1^a,\boldsymbol{x}_1^b\middle|\boldsymbol{x}_0^a,\boldsymbol{x}_0^b\right)\rightarrow\delta\left(\boldsymbol{x}_1^a-\boldsymbol{x}_1^a\left(\boldsymbol{x}_0^a,\boldsymbol{x}_0^b\right)\right)\delta\left(\boldsymbol{x}_1^b-\boldsymbol{x}_1^b\left(\boldsymbol{x}_0^a,\boldsymbol{x}_0^b\right)\right) \tag{29}$$

in the expression for the distortion D in a 2-stage FMC (see Equation 21), then D reduces to

$$\begin{aligned}D=2\int d\boldsymbol{x}_0^a\ d\boldsymbol{x}_0^b\ P_0\left(\boldsymbol{x}_0^a,\boldsymbol{x}_0^b\right)\int d\boldsymbol{x}_2^a\ d\boldsymbol{x}_2^b\ P_{21}\left(\boldsymbol{x}_2^a,\boldsymbol{x}_2^b\middle|\boldsymbol{x}_1^a\left(\boldsymbol{x}_0^a,\boldsymbol{x}_0^b\right),\boldsymbol{x}_1^b\left(\boldsymbol{x}_0^a,\boldsymbol{x}_0^b\right)\right)\\ \times\left(\left\|\boldsymbol{x}_0^a-\left\langle\boldsymbol{x}_0^a\right\rangle_{\tilde{P}_{02}\left(\boldsymbol{x}_0^a\middle|\boldsymbol{x}_2^a,\boldsymbol{x}_2^b\right)}\right\|^2+\left\|\boldsymbol{x}_0^b-\left\langle\boldsymbol{x}_0^b\right\rangle_{\tilde{P}_{02}\left(\boldsymbol{x}_0^b\middle|\boldsymbol{x}_2^a,\boldsymbol{x}_2^b\right)}\right\|^2\right)\end{aligned} \tag{30}$$

where the Euclidean error has been split into a sum of contributions from the a and the b subspaces of $\boldsymbol{x}_0$. The Euclidean error terms have an interesting structure. For instance, in FMC a the Euclidean distance between $\boldsymbol{x}_0^a$ and $\left\langle\boldsymbol{x}_0^a\right\rangle_{\tilde{P}_{02}\left(\boldsymbol{x}_0^a\middle|\boldsymbol{x}_2^a,\boldsymbol{x}_2^b\right)}$ is computed, which explicitly depends via $\tilde{P}_{02}\left(\boldsymbol{x}_0^a\middle|\boldsymbol{x}_2^a,\boldsymbol{x}_2^b\right)$ on the outputs $\left(\boldsymbol{x}_2^a,\boldsymbol{x}_2^b\right)$ of both FMC a and FMC b. In the previous section terms like $\left\langle\boldsymbol{x}_0\right\rangle_{\tilde{P}_{02}\left(\boldsymbol{x}_0\middle|\boldsymbol{x}_2\right)}$ turned out to correspond to SOM reference vectors, so by analogy it is expected that $\left\langle\boldsymbol{x}_0^k\right\rangle_{\tilde{P}_{02}\left(\boldsymbol{x}_0^k\middle|\boldsymbol{x}_2^a,\boldsymbol{x}_2^b\right)}$ (k=a, b) will also turn out to be the reference vectors for a pair of coupled SOMs corresponding to FMC a and FMC b.

Unfortunately, because of the coupling between FMC a and FMC b, $\left\langle\boldsymbol{x}_0^k\right\rangle_{\tilde{P}_{02}\left(\boldsymbol{x}_0^k\middle|\boldsymbol{x}_2^a,\boldsymbol{x}_2^b\right)}$ (k=a, b) depend on both $\boldsymbol{x}_2^a$ and $\boldsymbol{x}_2^b$. The purpose of the following derivation is to find an approximate way of optimising D that does not involve these simultaneous dependencies on both $\boldsymbol{x}_2^a$ and $\boldsymbol{x}_2^b$, and which uses quantities like $\left\langle\boldsymbol{x}_0^k\right\rangle_{\tilde{P}_{02}\left(\boldsymbol{x}_0^k\middle|\boldsymbol{x}_2^k\right)}$ rather than $\left\langle\boldsymbol{x}_0^k\right\rangle_{\tilde{P}_{02}\left(\boldsymbol{x}_0^k\middle|\boldsymbol{x}_2^a,\boldsymbol{x}_2^b\right)}$, for k=a, b.

Thus each term can be split in the following manner for k=a, b

$$\left\|\boldsymbol{x}_0^k-\left\langle\boldsymbol{x}_0^k\right\rangle_{\tilde{P}_{02}\left(\boldsymbol{x}_0^k\middle|\boldsymbol{x}_2^a,\boldsymbol{x}_2^b\right)}\right\|^2=\left\|\left(\boldsymbol{x}_0^k-\left\langle\boldsymbol{x}_0^k\right\rangle_{\tilde{P}_{02}\left(\boldsymbol{x}_0^k\middle|\boldsymbol{x}_2^k\right)}\right)+\left(\left\langle\boldsymbol{x}_0^k\right\rangle_{\tilde{P}_{02}\left(\boldsymbol{x}_0^k\middle|\boldsymbol{x}_2^k\right)}-\left\langle\boldsymbol{x}_0^k\right\rangle_{\tilde{P}_{02}\left(\boldsymbol{x}_0^k\middle|\boldsymbol{x}_2^a,\boldsymbol{x}_2^b\right)}\right)\right\|^2 \tag{31}$$

where the distinction between $\left\langle\boldsymbol{x}_0^k\right\rangle_{\tilde{P}_{02}\left(\boldsymbol{x}_0^k\middle|\boldsymbol{x}_2^a,\boldsymbol{x}_2^b\right)}$ and $\left\langle\boldsymbol{x}_0^k\right\rangle_{\tilde{P}_{02}\left(\boldsymbol{x}_0^k\middle|\boldsymbol{x}_2^k\right)}$ has been carefully used to separate the $\boldsymbol{x}_0^k$ and $\left\langle\boldsymbol{x}_0^k\right\rangle_{\tilde{P}_{02}\left(\boldsymbol{x}_0^k\middle|\boldsymbol{x}_2^a,\boldsymbol{x}_2^b\right)}$ terms.

The following two definitions are introduced

$$D_0(k) \equiv 2\int d\boldsymbol{x}_0^a \; d\boldsymbol{x}_0^b \; P_0\left(\boldsymbol{x}_0^a, \boldsymbol{x}_0^b\right)$$
$$\times \int d\boldsymbol{x}_2^k \; P_{21}\left(\boldsymbol{x}_2^k \middle| \boldsymbol{x}_1^a\left(\boldsymbol{x}_0^a, \boldsymbol{x}_0^b\right), \boldsymbol{x}_1^b\left(\boldsymbol{x}_0^a, \boldsymbol{x}_0^b\right)\right) \left\| \boldsymbol{x}_0^k - \left\langle \boldsymbol{x}_0^k \right\rangle_{\tilde{P}_{02}\left(\boldsymbol{x}_0^k \middle| \boldsymbol{x}_2^k\right)} \right\|^2 \qquad k = a,b \tag{32}$$
$$D_1(k) \equiv 2\int d\boldsymbol{x}_2^a \; d\boldsymbol{x}_2^b \; P_2\left(\boldsymbol{x}_2^a, \boldsymbol{x}_2^b\right) \left\| \left\langle \boldsymbol{x}_0^k \right\rangle_{\tilde{P}_{02}\left(\boldsymbol{x}_0^k \middle| \boldsymbol{x}_2^k\right)} - \left\langle \boldsymbol{x}_0^k \right\rangle_{\tilde{P}_{02}\left(\boldsymbol{x}_0^k \middle| \boldsymbol{x}_2^a, \boldsymbol{x}_2^b\right)} \right\|^2 \qquad k = a,b$$

and used together with Bayes' theorem in the form

$$P_0\left(\boldsymbol{x}_0^a, \boldsymbol{x}_0^b\right) P_{21}\left(\boldsymbol{x}_2^a, \boldsymbol{x}_2^b \middle| \boldsymbol{x}_1^a\left(\boldsymbol{x}_0^a, \boldsymbol{x}_0^b\right), \boldsymbol{x}_1^b\left(\boldsymbol{x}_0^a, \boldsymbol{x}_0^b\right)\right) = \tilde{P}_{02}\left(\boldsymbol{x}_0^a, \boldsymbol{x}_0^b \middle| \boldsymbol{x}_2^a, \boldsymbol{x}_2^b\right) P_2\left(\boldsymbol{x}_2^a, \boldsymbol{x}_2^b\right) \tag{33}$$

to obtain D as

$$D = D_0(a) + D_0(b) - D_1(a) - D_1(b) \tag{34}$$

The first two terms in Equation 34 have a structure that is simpler than in Equation 30. For instance $D_0(a)$ depends on the Euclidean distance between $\boldsymbol{x}_0^a$ and $\left\langle \boldsymbol{x}_0^a \right\rangle_{\tilde{P}_{02}\left(\boldsymbol{x}_0^a \middle| \boldsymbol{x}_2^a\right)}$, which depends only on the output of FMC a and not on the output of FMC b. An analogous remark applies to $D_0(b)$.

The last two terms in Equation 34 contain the undesirable terms such as $\left\langle \boldsymbol{x}_0^a \right\rangle_{\tilde{P}_{02}\left(\boldsymbol{x}_0^a \middle| \boldsymbol{x}_2^a, \boldsymbol{x}_2^b\right)}$. In the following derivation these terms will be discarded to obtain an approximate scheme for minimising D.

4.2. Least Upper Bound Optimisation Scheme

Note that the following inequalities hold

$$D_0(a) \geq 0 \quad D_0(b) \geq 0 \quad D_1(a) \geq 0 \quad D_1(b) \geq 0 \quad D \geq 0 \tag{35}$$

These lead to the following constraint on D

$$0 \leq D \leq D_0(a) + D_0(b) \tag{36}$$

Although ideally D itself should be minimised, it turns out to be much simpler to minimise its upper bound $D_0(a) + D_0(b)$. This least upper bound prescription achieves what is required, namely the elimination of the undesirable $D_1(a)$ and $D_1(b)$ terms which depend on $\left\langle \boldsymbol{x}_0^k \right\rangle_{\tilde{P}_{02}\left(\boldsymbol{x}_0^k \middle| \boldsymbol{x}_2^a, \boldsymbol{x}_2^b\right)}$ for k=a, b.

This approximate approach to optimising the FMC is shown in Figure 7.

At this point it is appropriate to introduce a modified form of $D_0(k)$ in which $\left\langle \boldsymbol{x}_0^k \right\rangle_{\tilde{P}_{02}\left(\boldsymbol{x}_0^k \middle| \boldsymbol{x}_2^k\right)}$ in Equation 32 is replaced by the function $\boldsymbol{x}_0'^k\left(\boldsymbol{x}_2^k\right)$

$$D_0(k) = 2\int d\boldsymbol{x}_0^a \; d\boldsymbol{x}_0^b \; P_0\left(\boldsymbol{x}_0^a, \boldsymbol{x}_0^b\right)$$
$$\times \int d\boldsymbol{x}_2^k \; P_{21}\left(\boldsymbol{x}_2^k \middle| \boldsymbol{x}_1^a\left(\boldsymbol{x}_0^a, \boldsymbol{x}_0^b\right), \boldsymbol{x}_1^b\left(\boldsymbol{x}_0^a, \boldsymbol{x}_0^b\right)\right) \left\| \boldsymbol{x}_0^k - \boldsymbol{x}_0'^k\left(\boldsymbol{x}_2^k\right) \right\|^2 \qquad k = a,b \tag{37}$$

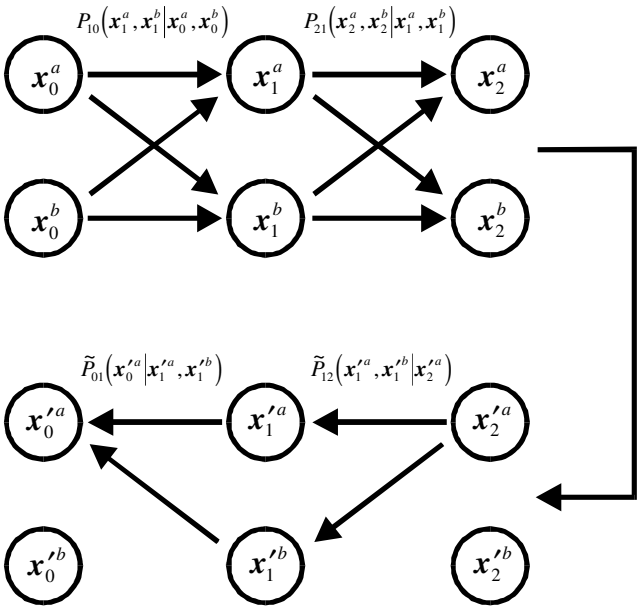

Figure 7. An approximation to a 2-stage folded Markov chain with each state split into two lower dimensional pieces. Only the $\tilde{P}_{02}\left(\boldsymbol{x}_0^a\middle|\boldsymbol{x}_2^a\right)$ part is indicated on the reverse part of the chain. The approximation arises because $\boldsymbol{x}_2^a$, rather then $\left(\boldsymbol{x}_2^a,\boldsymbol{x}_2^b\right)$, is used as the initial state on the reverse part of the chain. An analogous discussion holds for $\tilde{P}_{02}\left(\boldsymbol{x}_0^b\middle|\boldsymbol{x}_2^b\right)$.

The functional derivative $\delta D_0(k)\big/\delta \boldsymbol{x}_0'^k\left(\boldsymbol{x}_2^k\right)$ then becomes

$$\frac{\delta D_0(k)}{\delta \boldsymbol{x}_0'^k\left(\boldsymbol{x}_2^k\right)} = -4\int d\boldsymbol{x}_0^a\; d\boldsymbol{x}_0^b\; P_0\left(\boldsymbol{x}_0^a,\boldsymbol{x}_0^b\right) \times P_{21}\left(\boldsymbol{x}_2^k\middle|\boldsymbol{x}_1^a\left(\boldsymbol{x}_0^a,\boldsymbol{x}_0^b\right),\boldsymbol{x}_1^b\left(\boldsymbol{x}_0^a,\boldsymbol{x}_0^b\right)\right)\left(\boldsymbol{x}_0^k-\boldsymbol{x}_0'^k\left(\boldsymbol{x}_2^k\right)\right) \qquad k=a,b \tag{38}$$

and the cross derivatives $\delta D_0(a)\big/\delta \boldsymbol{x}_0'^b\left(\boldsymbol{x}_2^b\right)$ and $\delta D_0(b)\big/\delta \boldsymbol{x}_0'^a\left(\boldsymbol{x}_2^a\right)$ are zero. The stationary point $\delta D_0(k)\big/\delta \boldsymbol{x}_0'^k\left(\boldsymbol{x}_2^k\right)=0$ is obtained when $\boldsymbol{x}_0'^k\left(\boldsymbol{x}_2^k\right)$ satisfies

$$\boldsymbol{x}_0'^k\left(\boldsymbol{x}_2^k\right)=\frac{\int d\boldsymbol{x}_0^a\; d\boldsymbol{x}_0^b\; P_0\left(\boldsymbol{x}_0^a,\boldsymbol{x}_0^b\right)\,P_{21}\left(\boldsymbol{x}_2^k\middle|\boldsymbol{x}_1^a\left(\boldsymbol{x}_0^a,\boldsymbol{x}_0^b\right),\boldsymbol{x}_1^b\left(\boldsymbol{x}_0^a,\boldsymbol{x}_0^b\right)\right)\boldsymbol{x}_0^k}{\int d\boldsymbol{x}_0^a\; d\boldsymbol{x}_0^b\; P_0\left(\boldsymbol{x}_0^a,\boldsymbol{x}_0^b\right)\,P_{21}\left(\boldsymbol{x}_2^k\middle|\boldsymbol{x}_1^a\left(\boldsymbol{x}_0^a,\boldsymbol{x}_0^b\right),\boldsymbol{x}_1^b\left(\boldsymbol{x}_0^a,\boldsymbol{x}_0^b\right)\right)} \qquad k=a,b \tag{39}$$

Note that this result can readily be generalised to an arbitrary choice of $P_{10}\left(\boldsymbol{x}_1^a,\boldsymbol{x}_1^b\middle|\boldsymbol{x}_0^a,\boldsymbol{x}_0^b\right)$ (i.e. *not* assuming Equation 29).

By using Bayes' theorem in the form shown in Equation 33 this reduces to

$$\boldsymbol{x}_0'^k\left(\boldsymbol{x}_2^k\right)=\left\langle \boldsymbol{x}_0^k\right\rangle_{\tilde{P}_{02}\left(\boldsymbol{x}_0^k\middle|\boldsymbol{x}_2^k\right)} \qquad k=a,b \tag{40}$$

The modified expression for $D_0(k)$ in Equation 37 reduces to the original expression for $D_0(k)$ in Equation 32 *provided that* $\boldsymbol{x}_0'^k\left(\boldsymbol{x}_2^k\right)$ is chosen to minimise $D_0(k)$, so Equation 37 will be used in preference to Equation 32 (with the proviso that $\boldsymbol{x}_0'^k\left(\boldsymbol{x}_2^k\right)$ should always be chosen to minimise $D_0(k)$).

Equation 37 leads to the following result for $\boldsymbol{x}_1^k\left(\boldsymbol{x}_0^a,\boldsymbol{x}_0^b\right)$

$$\boldsymbol{x}_1^k\left(\boldsymbol{x}_0^a,\boldsymbol{x}_0^b\right)=\underset{\boldsymbol{x}_1^k}{\arg\min}\left(\begin{array}{l}\int d\boldsymbol{x}_2^a\ P_{21}\left(\boldsymbol{x}_2^a\middle|\boldsymbol{x}_1^a,\boldsymbol{x}_1^b\right)\left\|\boldsymbol{x}_0^a-\boldsymbol{x}_0'^a\left(\boldsymbol{x}_2^a\right)\right\|^2\\ +\int d\boldsymbol{x}_2^b\ P_{21}\left(\boldsymbol{x}_2^b\middle|\boldsymbol{x}_1^a,\boldsymbol{x}_1^b\right)\left\|\boldsymbol{x}_0^b-\boldsymbol{x}_0'^b\left(\boldsymbol{x}_2^b\right)\right\|^2\end{array}\right)\qquad k=1,2 \tag{41}$$

and the following on-line training prescription

$$\boldsymbol{x}_0'^k\left(\boldsymbol{x}_2^k\right)\rightarrow\boldsymbol{x}_0'^k\left(\boldsymbol{x}_2^k\right)+\varepsilon\ P_{21}\left(\boldsymbol{x}_2^k\middle|\boldsymbol{x}_1^a\left(\boldsymbol{x}_0^a,\boldsymbol{x}_0^b\right),\boldsymbol{x}_1^b\left(\boldsymbol{x}_0^a,\boldsymbol{x}_0^b\right)\right)\left(\boldsymbol{x}_0^k-\boldsymbol{x}_0'^k\left(\boldsymbol{x}_2^k\right)\right)\qquad k=1,2 \tag{42}$$

These results correspond to the self-supervised training scheme that was proposed in [12,13].

The following remarks can be made about these results:

1. The functions $\boldsymbol{x}_0'^a\left(\boldsymbol{x}_2^a\right)$ and $\boldsymbol{x}_0'^b\left(\boldsymbol{x}_2^b\right)$ are the continuum versions of the reference vectors of the pair of SOMs corresponding to FMC *a* and FMC *b*, respectively.
2. The results for $\boldsymbol{x}_1^a\left(\boldsymbol{x}_0^a,\boldsymbol{x}_0^b\right)$ and $\boldsymbol{x}_1^b\left(\boldsymbol{x}_0^a,\boldsymbol{x}_0^b\right)$ are modified forms of the minimum distortion encoding prescription, in which the coupling between the FMC *a* and FMC *b* manifests itself through $P_{21}\left(\boldsymbol{x}_2^a\middle|\boldsymbol{x}_1^a,\boldsymbol{x}_1^b\right)$ and $P_{21}\left(\boldsymbol{x}_2^b\middle|\boldsymbol{x}_1^a,\boldsymbol{x}_1^b\right)$.
3. In both the batch and the on-line training prescription $P_{21}\left(\boldsymbol{x}_2^a\middle|\boldsymbol{x}_1^a,\boldsymbol{x}_1^b\right)$ and $P_{21}\left(\boldsymbol{x}_2^b\middle|\boldsymbol{x}_1^a,\boldsymbol{x}_1^b\right)$ play the role of neighbourhood functions for SOM *a* and SOM *b*, respectively. However, the coupling between the SOMs causes these neighbourhood functions to be dependent on the input data, as will be discussed in more detail below.

4.3. Network Representation of Self-Supervision

In Figure 8 the network representation of the pair of coupled SOMs is shown (the discrete version, not the continuum version, is shown).

This type of coupling between SOMs occurs in image processing, where two adjacent patches of image $\boldsymbol{x}_0^a$ and $\boldsymbol{x}_0^b$ are pre-processed to produce a pair of outputs $\boldsymbol{x}_1^a$ and $\boldsymbol{x}_1^b$ (which might be features, for instance), which are subsequently post-processed to produce $\boldsymbol{x}_2^a$ and $\boldsymbol{x}_2^b$ (which might be connected features, for instance). If subsequently $\boldsymbol{x}_2^a$ and $\boldsymbol{x}_2^b$ are to be used to attempt to reconstruct the inputs $\boldsymbol{x}_0^a$ and $\boldsymbol{x}_0^b$ with minimum average Euclidean error, then this type of image processing can be directly related to a pair of coupled 2-stage FMCs. In both cases (image processing and coupled FMCs) the choice of post-processing stage influences the optimal choice of pre-processing stage.

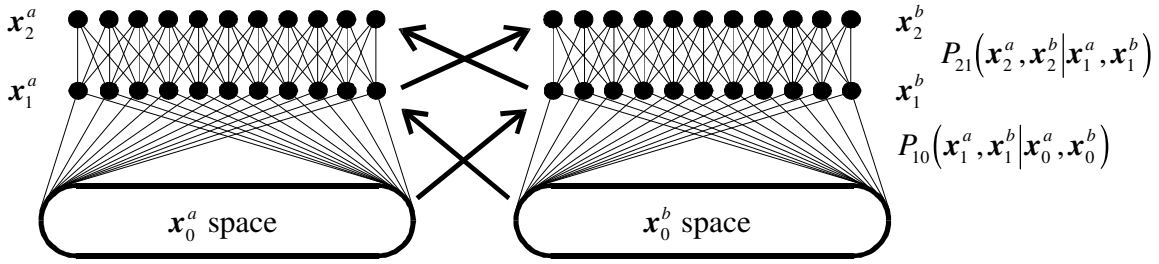

Figure 8. A pair of coupled discrete SOMs represented as a pair of coupled networks of the type shown in Figure 5. As derived above, the coupling $P_{21}\left(\boldsymbol{x}_2^a, \boldsymbol{x}_2^b \middle| \boldsymbol{x}_1^a, \boldsymbol{x}_1^b\right)$ between networks *a* and *b* expresses itself as a pair of data dependent neighbourhood functions $P_{21}\left(\boldsymbol{x}_2^a \middle| \boldsymbol{x}_1^a, \boldsymbol{x}_1^b\right)$ and $P_{21}\left(\boldsymbol{x}_2^b \middle| \boldsymbol{x}_1^a, \boldsymbol{x}_1^b\right)$. When the first stage of each of these coupled SOMs is then optimised, taking account of $P_{21}\left(\boldsymbol{x}_2^a \middle| \boldsymbol{x}_1^a, \boldsymbol{x}_1^b\right)$ and $P_{21}\left(\boldsymbol{x}_2^b \middle| \boldsymbol{x}_1^a, \boldsymbol{x}_1^b\right)$, the encoding functions $\boldsymbol{x}_1^a\left(\boldsymbol{x}_0^a, \boldsymbol{x}_0^b\right)$ and $\boldsymbol{x}_1^b\left(\boldsymbol{x}_0^a, \boldsymbol{x}_0^b\right)$ become mutually coupled.

The detailed interpretation of Equation 41 and Equation 42 is shown in Figure 9.

The marginal probabilities $P_{21}\left(\boldsymbol{x}_2^a \middle| \boldsymbol{x}_1^a, \boldsymbol{x}_1^b\right)$ and $P_{21}\left(\boldsymbol{x}_2^b \middle| \boldsymbol{x}_1^a, \boldsymbol{x}_1^b\right)$ are obtained by projecting $P_{21}\left(\boldsymbol{x}_2^a, \boldsymbol{x}_2^b \middle| \boldsymbol{x}_1^a, \boldsymbol{x}_1^b\right)$ onto $\boldsymbol{x}_2^a$ and $\boldsymbol{x}_2^b$, respectively. The data dependence of $P_{21}\left(\boldsymbol{x}_2^a, \boldsymbol{x}_2^b \middle| \boldsymbol{x}_1^a, \boldsymbol{x}_1^b\right)$ causes these marginal probabilities to be data dependent. In particular, they can be biassed as shown in Figure 9, which causes the neighbourhood functions (in the SOM interpretation) in $\boldsymbol{x}_1^a$ and $\boldsymbol{x}_1^b$ space to be biassed.

The data dependence has another subtle side effect in Figure 9. The input transformations $\boldsymbol{x}_1^a\left(\boldsymbol{x}_0^a, \boldsymbol{x}_0^b\right)$ and $\boldsymbol{x}_1^b\left(\boldsymbol{x}_0^a, \boldsymbol{x}_0^b\right)$, which select the starting positions on the $\boldsymbol{x}_1^a$ and $\boldsymbol{x}_1^b$ axes, depend on the marginal probabilities $P_{21}\left(\boldsymbol{x}_2^a \middle| \boldsymbol{x}_1^a, \boldsymbol{x}_1^b\right)$ and $P_{21}\left(\boldsymbol{x}_2^b \middle| \boldsymbol{x}_1^a, \boldsymbol{x}_1^b\right)$, because the input transformations satisfy a minimum distortion criterion which depends on these marginal probabilities (see Equation 41). However, the marginal probabilities themselves are data dependent, because they depend on $\boldsymbol{x}_1^a$ and $\boldsymbol{x}_1^b$, which in turn depend on the input transformations. Overall, the marginal probabilities and the input transformations are mutually dependent, which makes the minimum distortion encoding prescription quite subtle to implement [13].

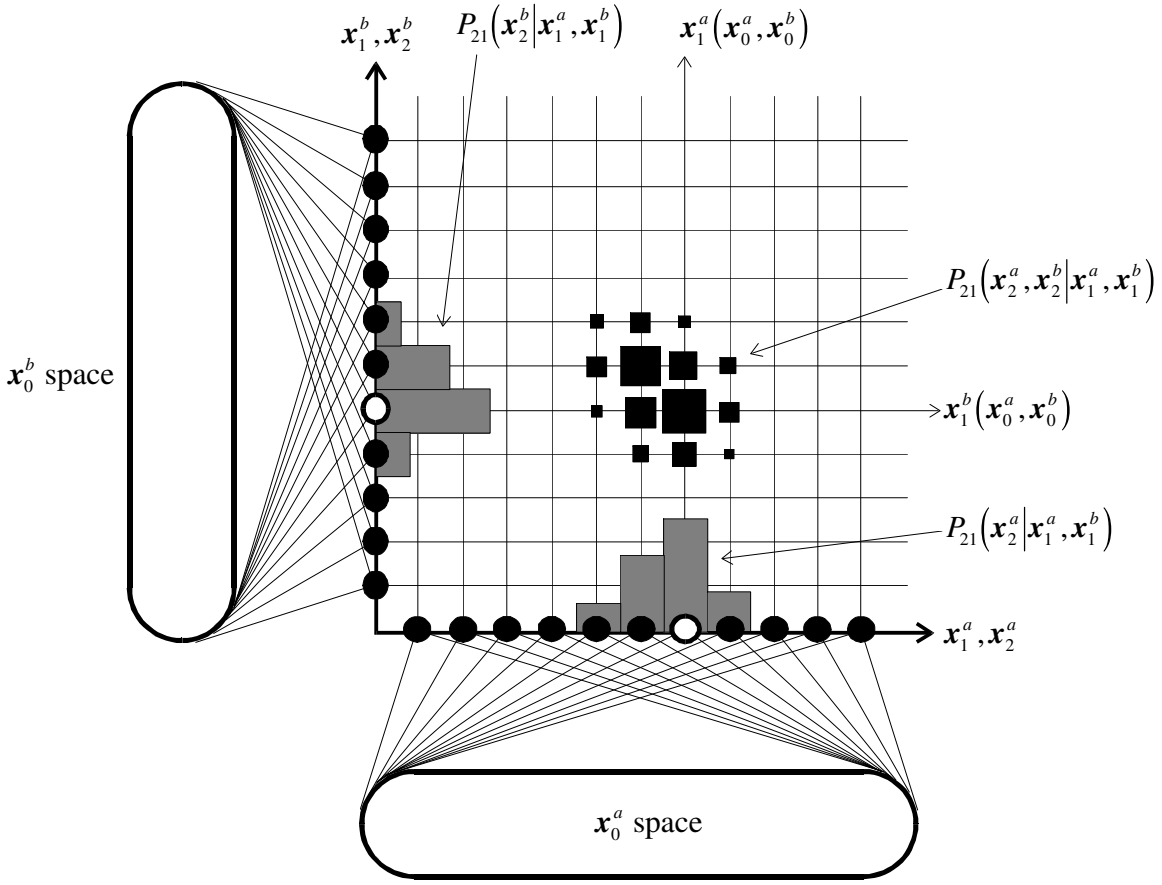

Figure 9. Diagram showing the detailed operation of a pair of coupled FMCs. For simplicity, $\left(\boldsymbol{x}_2^a, \boldsymbol{x}_2^b\right)$ is assumed to sit in the same vector space as $\left(\boldsymbol{x}_1^a, \boldsymbol{x}_1^b\right)$, although this is not a necessary assumption. Examples of the winners $\boldsymbol{x}_1^a\left(\boldsymbol{x}_0^a, \boldsymbol{x}_0^b\right)$ and $\boldsymbol{x}_1^b\left(\boldsymbol{x}_0^a, \boldsymbol{x}_0^b\right)$ are indicated by the open circles. These are then jointly smeared into the distribution $P_{21}\left(\boldsymbol{x}_2^a, \boldsymbol{x}_2^b \middle| \boldsymbol{x}_1^a, \boldsymbol{x}_1^b\right)$, which is drawn as a 2-dimensional histogram. As derived above, the optimisation depends on a pair of neighbourhood functions $P_{21}\left(\boldsymbol{x}_2^a \middle| \boldsymbol{x}_1^a, \boldsymbol{x}_1^b\right)$ and $P_{21}\left(\boldsymbol{x}_2^b \middle| \boldsymbol{x}_1^a, \boldsymbol{x}_1^b\right)$, which are the marginal probabilities of $P_{21}\left(\boldsymbol{x}_2^a, \boldsymbol{x}_2^b \middle| \boldsymbol{x}_1^a, \boldsymbol{x}_1^b\right)$, and which are drawn as 1-dimensional histograms in the diagram. These data-dependent neighbourhood functions not only influence the optimisation of the FMC, but also determine the winners that should have been used in the first place.

Further details on self-supervision can be found in [12,13], where a detailed discussion and numerical simulation of the consequences of using a particular type of $P_{21}\left(\boldsymbol{x}_2^a, \boldsymbol{x}_2^b \middle| \boldsymbol{x}_1^a, \boldsymbol{x}_1^b\right)$ are presented, a comparison is made between nearest neighbour and minimum distortion encoding, and a comparison is made between using mutually dependent neighbourhood

functions $P_{21}\left(\boldsymbol{x}_2^k \middle| \boldsymbol{x}_1^a, \boldsymbol{x}_1^b\right)$ (k=a, b) and independent neighbourhood functions $P_{21}\left(\boldsymbol{x}_2^a \middle| \boldsymbol{x}_1^a\right)$ and $P_{21}\left(\boldsymbol{x}_2^b \middle| \boldsymbol{x}_1^b\right)$.

5. Conclusions

In this paper it has been demonstrated that VQ theory, SOM theory, and the theory of self-supervision all emerge naturally when an FMC is optimised so as to minimise the expected Euclidean error between an input vector and its attempted reconstruction (using Bayes' theorem).

FMC theory is not merely a fancy way of thinking about VQs and SOMs; it can be used to facilitate many computations that would otherwise be theoretically and/or numerically intractable. In effect, the "soft" probabilities that are used in the FMC are easier to compute with than the "hard" delta functions in the corresponding winner-take-all VQs and SOMs. The results contained in this paper guarantee that these "soft" computations reduce to the required "hard" computations when the first stage of the FMC is optimised.

6. Acknowledgements

The author is indebted to the following people for critically reading this paper: Eric Jakeman, David Lowe, Graeme Mitchison.

7. Appendix

7.1. Relationship between Continuum and Discrete Vector Quantisers

Throughout this paper continuum notation is used. In the case of a VQ this has the effect that the index that is used to select the winning reference vector is assumed to be a continuous-valued quantity, rather than a discrete-valued quantity. The purpose of this appendix is to relate the continuum case to the discrete case.

It is sufficient to discuss the meaning of the on-line training prescription in Equation 17 and Equation 18, which is presented again here for convenience.

$$\begin{aligned} \boldsymbol{x}_1(\boldsymbol{x}_0) &= \underset{\boldsymbol{x}_1}{\arg\min} \left(\boldsymbol{x}_0 - \boldsymbol{x}_0'(\boldsymbol{x}_1)\right) \\ \boldsymbol{x}_0'(\boldsymbol{x}_1) &\to \boldsymbol{x}_0'(\boldsymbol{x}_1) + \varepsilon\, \delta\left(\boldsymbol{x}_1 - \boldsymbol{x}_1(\boldsymbol{x}_0)\right)\left(\boldsymbol{x}_0 - \boldsymbol{x}_0'(\boldsymbol{x}_1)\right) \end{aligned} \tag{43}$$

where $\boldsymbol{x}_1$ is assumed to lie in the unit N-dimensional hypercube. The discrete counterpart of this on-line prescription is

$$\begin{aligned} \boldsymbol{k}_1(\boldsymbol{x}_0) &= \underset{\boldsymbol{k}_1}{\arg\min} \left(\boldsymbol{x}_0 - \boldsymbol{x}_{0,\boldsymbol{k}_1}'\right) \\ \boldsymbol{x}_{0,\boldsymbol{k}_1}' &\to \boldsymbol{x}_{0,\boldsymbol{k}_1}' + \varepsilon\, \delta_{\boldsymbol{k}_1, \boldsymbol{k}_1(\boldsymbol{x}_0)} \left(\boldsymbol{x}_0 - \boldsymbol{x}_{0,\boldsymbol{k}_1}'\right) \end{aligned} \tag{44}$$

where $\boldsymbol{k}_1$ is a point on an N-dimensional cubic lattice.

The relationship between the nearest neighbour encoding prescriptions $\boldsymbol{x}_1(\boldsymbol{x}_0)$ and $\boldsymbol{k}_1(\boldsymbol{x}_0)$ is simple to understand, so no further comment is required. On the other hand, the relationship between the update prescriptions for $\boldsymbol{x}_0'(\boldsymbol{x}_1)$ and $\boldsymbol{x}_{0,\boldsymbol{k}_1}'$ is quite subtle, so they will now be discussed in some detail.

In order to interpret the continuum prescription correctly it is necessary to integrate $\boldsymbol{x}_1$ over a small region S with volume δV that encloses the point $\boldsymbol{x}_1 = \boldsymbol{x}_1(\boldsymbol{x}_0)$. This leads to the result

$$\boldsymbol{x}_0'(\boldsymbol{x}_1) \to \boldsymbol{x}_0'(\boldsymbol{x}_1) + \frac{\varepsilon}{\delta V}\left(\boldsymbol{x}_0 - \boldsymbol{x}_0'(\boldsymbol{x}_1)\right) \quad \text{where } \boldsymbol{x}_1 \in S \text{ and } \boldsymbol{x}_1(\boldsymbol{x}_0) \in S \tag{45}$$

Note that $\varepsilon/\delta V$, rather than ε, determines the size of the updates of the reference vectors that are attached to points $\boldsymbol{x}_1$ inside S. The smaller the volume δV is, the smaller ε has to be to keep the updates the same size. The most natural prescription is to impose a lower bound δL on the length scale of interest of $\boldsymbol{x}_1$, which then imposes a natural size on the volume of S such that $\delta V = \delta L^N$. Since $\boldsymbol{x}_1$ is assumed to lie in the unit hypercube, then this prescription leads to a finite effective number of reference vectors given by δL^{-N}, which would correspond to an N-dimensional cubic lattice whose size was $1/\delta L$ lattice spacings in each dimension.

An interesting side effect of introducing this inner length scale δL (which is effectively a regularisation constant) is that it automatically generates a SOM on-line training prescription. To see this, note that in Equation 45 a whole volume δV of reference vectors $\boldsymbol{x}_1$ in the neighbourhood S of the point $\boldsymbol{x}_1 = \boldsymbol{x}_1(\boldsymbol{x}_0)$ is updated. This is precisely the type of update prescription that creates a SOM.

7.2. Full Optimisation of 2-Stage Folded Markov Chains

In this appendix it will be shown that if a 2-stage FMC is optimised with respect to $P_{10}(\boldsymbol{x}_1|\boldsymbol{x}_0)$, then it necessarily reduces to $P_{10}(\boldsymbol{x}_1|\boldsymbol{x}_0) = \delta\left(\boldsymbol{x}_1 - \boldsymbol{x}_1(\boldsymbol{x}_0)\right)$. Because the 2-stage FMC contains the 1-stage FMC as a special case, it is necessary to present the derivation only for a 2-stage FMC. The result that is obtained depends on certain properties of the Euclidean distortion measure; it is not a general property of FMCs.

There are two constraints that must be respected during the optimisation of $P_{10}(\boldsymbol{x}_1|\boldsymbol{x}_0)$. Total probability must be conserved

$$\int d\boldsymbol{x}_1 \; P_{10}(\boldsymbol{x}_1|\boldsymbol{x}_0) = 1 \quad \text{for all } \boldsymbol{x}_0 \tag{46}$$

and $\tilde{P}_{01}(\boldsymbol{x}_0|\boldsymbol{x}_1)$ must vary in response to variations of $P_{10}(\boldsymbol{x}_1|\boldsymbol{x}_0)$ in such a way that Bayes' theorem is respected

$$\tilde{P}_{01}(\boldsymbol{x}_0|\boldsymbol{x}_1)\left(\int d\boldsymbol{u}_0 \; P_{10}(\boldsymbol{x}_1|\boldsymbol{u}_0)P_0(\boldsymbol{u}_0)\right) = P_{10}(\boldsymbol{x}_1|\boldsymbol{x}_0)P_0(\boldsymbol{x}_0) \quad \text{for all } \boldsymbol{x}_0, \boldsymbol{x}_1 \tag{47}$$

The first constraint will be imposed for all $\boldsymbol{x}_0$ by using a Lagrange multiplier function $\lambda(\boldsymbol{x}_0)$, whereas the second constraint will be imposed by using Bayes' theorem to eliminate $\tilde{P}_{01}(\boldsymbol{x}_0|\boldsymbol{x}_1)$ from D.

The overall quantity $\overline{D}$ to be minimised is therefore (dropping terms that have zero derivative with respect to $P_{10}(\boldsymbol{x}_1|\boldsymbol{x}_0)$)

$$\begin{aligned}\overline{D} = &\int d\boldsymbol{x}_0\, d\boldsymbol{x}_1\, d\boldsymbol{x}_2\, d\boldsymbol{x}_0'\, d\boldsymbol{x}_1'\, d\boldsymbol{x}_2' \\ &\times P_0(\boldsymbol{x}_0) P_{10}(\boldsymbol{x}_1|\boldsymbol{x}_0) P_{21}(\boldsymbol{x}_2|\boldsymbol{x}_1) \delta(\boldsymbol{x}_2' - \boldsymbol{x}_2) \tilde{P}_{12}(\boldsymbol{x}_1'|\boldsymbol{x}_2') \tilde{P}_{01}(\boldsymbol{x}_0'|\boldsymbol{x}_1') \|\boldsymbol{x}_0' - \boldsymbol{x}_0\|^2 \\ &-\int d\boldsymbol{x}_0\, \lambda(\boldsymbol{x}_0) \int d\boldsymbol{x}_1\, P_{10}(\boldsymbol{x}_1|\boldsymbol{x}_0)\end{aligned} \tag{48}$$

Integration over $\boldsymbol{x}_2$ and use of Bayes' theorem on both $\tilde{P}_{12}(\boldsymbol{x}_1'|\boldsymbol{x}_2')$ and $\tilde{P}_{01}(\boldsymbol{x}_0'|\boldsymbol{x}_1')$ transforms $\overline{D}$ into a form that is suitable for differentiation with respect to $P_{10}(\boldsymbol{x}_1|\boldsymbol{x}_0)$

$$\begin{aligned}\overline{D} = &\int d\boldsymbol{x}_0\, d\boldsymbol{x}_1\, d\boldsymbol{x}_0'\, d\boldsymbol{x}_1'\, d\boldsymbol{x}_2'\; P_0(\boldsymbol{x}_0) P_{10}(\boldsymbol{x}_1|\boldsymbol{x}_0) P_{10}(\boldsymbol{x}_1'|\boldsymbol{x}_0') P_0(\boldsymbol{x}_0') \\ &\times \frac{P_{21}(\boldsymbol{x}_2'|\boldsymbol{x}_1) P_{21}(\boldsymbol{x}_2'|\boldsymbol{x}_1')}{\left(\int d\boldsymbol{u}_0\, d\boldsymbol{u}_1\; P_{21}(\boldsymbol{x}_2'|\boldsymbol{u}_1) P_{10}(\boldsymbol{u}_1|\boldsymbol{u}_0) P_0(\boldsymbol{u}_0)\right)} \|\boldsymbol{x}_0' - \boldsymbol{x}_0\|^2 \\ &-\int d\boldsymbol{x}_0\, \lambda(\boldsymbol{x}_0) \int d\boldsymbol{x}_1\, P_{10}(\boldsymbol{x}_1|\boldsymbol{x}_0)\end{aligned} \tag{49}$$

Functional differentiation of $\overline{D}$ with respect to $P_{10}(z_1|z_0)$, and using the symmetry of $\overline{D}$ under interchange of $\boldsymbol{x}_0$ and $\boldsymbol{x}_0'$, then leads to

$$\begin{aligned}\frac{\delta \overline{D}}{\delta P_{10}(z_1|z_0)} = &2\int d\boldsymbol{x}_0\, d\boldsymbol{x}_1\, d\boldsymbol{x}_0'\, d\boldsymbol{x}_1'\, d\boldsymbol{x}_2'\; \delta(\boldsymbol{x}_0 - z_0) \delta(\boldsymbol{x}_1 - z_1) P_0(\boldsymbol{x}_0) P_{10}(\boldsymbol{x}_1'|\boldsymbol{x}_0') P_0(\boldsymbol{x}_0') \\ &\times \frac{P_{21}(\boldsymbol{x}_2'|\boldsymbol{x}_1) P_{21}(\boldsymbol{x}_2'|\boldsymbol{x}_1')}{\left(\int d\boldsymbol{u}_0\, d\boldsymbol{u}_1\; P_{21}(\boldsymbol{x}_2'|\boldsymbol{u}_1) P_{10}(\boldsymbol{u}_1|\boldsymbol{u}_0) P_0(\boldsymbol{u}_0)\right)} \|\boldsymbol{x}_0' - \boldsymbol{x}_0\|^2 \\ &-\int d\boldsymbol{x}_0\, d\boldsymbol{x}_1\, d\boldsymbol{x}_0'\, d\boldsymbol{x}_1'\, d\boldsymbol{x}_2'\; P_0(\boldsymbol{x}_0) P_{10}(\boldsymbol{x}_1|\boldsymbol{x}_0) P_{10}(\boldsymbol{x}_1'|\boldsymbol{x}_0') P_0(\boldsymbol{x}_0') \\ &\times \frac{P_{21}(\boldsymbol{x}_2'|\boldsymbol{x}_1) P_{21}(\boldsymbol{x}_2'|\boldsymbol{x}_1')}{\left(\int d\boldsymbol{u}_0\, d\boldsymbol{u}_1\; P_{21}(\boldsymbol{x}_2'|\boldsymbol{u}_1) P_{10}(\boldsymbol{u}_1|\boldsymbol{u}_0) P_0(\boldsymbol{u}_0)\right)^2} \\ &\times \left(\int d\boldsymbol{v}_0\, d\boldsymbol{v}_1\, \delta(\boldsymbol{v}_0 - z_0) \delta(\boldsymbol{v}_1 - z_1) P_{21}(\boldsymbol{x}_2'|\boldsymbol{v}_1) P_0(\boldsymbol{v}_0)\right) \|\boldsymbol{x}_0' - \boldsymbol{x}_0\|^2 \\ &-\int d\boldsymbol{x}_0\, \lambda(\boldsymbol{x}_0) \int d\boldsymbol{x}_1\, \delta(\boldsymbol{x}_0 - z_0) \delta(\boldsymbol{x}_1 - z_1)\end{aligned} \tag{50}$$

which simplifies to

$$\frac{\delta\overline{D}}{\delta P_{10}(z_1|z_0)} = 2P_0(z_0)\int dx_2'\ P_{21}(x_2'|z_1)\int dx_1'\ \tilde{P}_{12}(x_1'|x_2')$$
$$\times\int dx_0'\ \tilde{P}_{01}(x_0'|x_1')\,\|x_0'-z_0\|^2$$
$$-P_0(z_0)\int dx_2'\ P_{21}(x_2'|z_1)\int dx_1\ \tilde{P}_{12}(x_1|x_2')\int dx_1'\ \tilde{P}_{12}(x_1'|x_2') \qquad (51)$$
$$\times\int dx_0\ dx_0'\ \tilde{P}_{01}(x_0|x_1)\tilde{P}_{01}(x_0'|x_1')\,\|x_0'-x_0\|^2$$
$$-\lambda(z_0)$$

Expansion and factorisation eventually lead to the final result

$$\frac{\delta\overline{D}}{\delta P_{10}(z_1|z_0)} = 2P_0(z_0)\int dx_2'\ P_{21}(x_2'|z_1)\left\|\int dx_0'\ dx_1'\ \tilde{P}_{01}(x_0'|x_1')\tilde{P}_{12}(x_1'|x_2')\,x_0'-z_0\right\|^2-\lambda(z_0) \qquad (52)$$

The stationarity condition $\delta\overline{D}/\delta P_{10}(x_1|x_0)=0$ is therefore

$$2P_0(x_0)\int dx_2'\ P_{21}(x_2'|x_1)\left\|\int dx_0'\ dx_1'\ \tilde{P}_{01}(x_0'|x_1')\tilde{P}_{12}(x_1'|x_2')\,x_0'-x_0\right\|^2=\lambda(x_0) \qquad (53)$$

The right hand side of Equation 53 is a function only of x_0, whereas the left hand side is a function of both x_0 and x_1, where x_1 appears only in the $P_{21}(x_2'|x_1)$ factor. Because x_1 appears only on the left hand side of this equation, it must somehow vanish.

There are two types of solution in which the dependence on x_1 vanishes:

1. $P_0(x_0)=0$. The x_1 dependence of the left hand side of Equation 53 is suppressed by the zero-valued $P_0(x_0)$ factor. Any legal probability $P_{10}(x_1|x_0)$ is thus permitted for values of x_0 that have $P_0(x_0)=0$. This solution may be eliminated because for those x_0 that have $P_0(x_0)=0$ there is no need to define $P_{10}(x_1|x_0)$ anyway.

2. $P_0(x_0)>0$. Note also that the variance-like factor $\left\|\int dx_0'\ dx_1'\ \tilde{P}_{01}(x_0'|x_1')\tilde{P}_{12}(x_1'|x_2')\,x_0'-x_0\right\|^2\geq 0$, so there are two situations to consider:

   (a) $\|\cdots\|^2>0$. In order to suppress the x_1 dependence of the left hand side of Equation 53 x_1 must be uniquely determined by the value of x_0. So $x_1=x_1(x_0)$ and $P_{10}(x_1|x_0)=\delta(x_1-x_1(x_0))$.

   (b) $\|\cdots\|^2=0$. Apparently, any legal probability $P_{10}(x_1|x_0)$ is permitted. However, any $P_{10}(x_1|x_0)\neq\delta(x_1-x_1(x_0))$ will guarantee that the variance-like factor $\|\cdots\|^2>0$, so the only $P_{10}(x_1|x_0)$ that can possibly survive are $P_{10}(x_1|x_0)=\delta(x_1-x_1(x_0))$. Note that with $P_{10}(x_1|x_0)=\delta(x_1-x_1(x_0))$ it is still possible that $\|\cdots\|^2>0$, in which case this solution can be eliminated.

The only solution that remains is $P_{10}(\boldsymbol{x}_1|\boldsymbol{x}_0)=\delta(\boldsymbol{x}_1-\boldsymbol{x}_1(\boldsymbol{x}_0))$. This result establishes the fact that the replacement of $P_{10}(\boldsymbol{x}_1|\boldsymbol{x}_0)$ by $\delta(\boldsymbol{x}_1-\boldsymbol{x}_1(\boldsymbol{x}_0))$ used in Equation 14 (and in Equation 29, in the case of coupled FMCs) emerges naturally from minimising D in the function space of probabilities $P_{10}(\boldsymbol{x}_1|\boldsymbol{x}_0)$.

It is important to note that the choice of a Euclidean error measure is sufficient (but not necessary) for $P_{10}(\boldsymbol{x}_1|\boldsymbol{x}_0)=\delta(\boldsymbol{x}_1-\boldsymbol{x}_1(\boldsymbol{x}_0))$ to emerge. Also, note that it is not true in general that optimisation of $P_{10}(\boldsymbol{x}_1|\boldsymbol{x}_0)$ leads to $P_{10}(\boldsymbol{x}_1|\boldsymbol{x}_0)=\delta(\boldsymbol{x}_1-\boldsymbol{x}_1(\boldsymbol{x}_0))$.

Two counterexamples in which $\|\boldsymbol{x}_0-\boldsymbol{x}_0'\|^2$ is replaced by some other functional form are now presented.

$$\|\boldsymbol{x}_0-\boldsymbol{x}_0'\|^2 \rightarrow \begin{cases} A(\boldsymbol{x}_0)+B(\boldsymbol{x}_0') & \text{counterexample 1} \\ A(\boldsymbol{x}_0)B(\boldsymbol{x}_0') & \text{counterexample 2} \end{cases} \tag{54}$$

Although these might not be considered to be sensible error measures, the fact that counterexamples exist is in itself important. Here are some comments on these counterexamples:

1. Counterexample 1 leads to a D which has no dependence on $P_{10}(\boldsymbol{x}_1|\boldsymbol{x}_0)$. Optimisation of $P_{10}(\boldsymbol{x}_1|\boldsymbol{x}_0)$ will allow any legal probability, so $P_{10}(\boldsymbol{x}_1|\boldsymbol{x}_0)\neq\delta(\boldsymbol{x}_1-\boldsymbol{x}_1(\boldsymbol{x}_0))$ is permitted in general.
2. Counterexample 2 is more complicated to analyse. However, it is possible to show that $P_{10}(\boldsymbol{x}_1|\boldsymbol{x}_0)$, when viewed as a matrix, has a block diagonal structure. This type of $P_{10}(\boldsymbol{x}_1|\boldsymbol{x}_0)$ does not imply a deterministic relationship between $\boldsymbol{x}_0$ and $\boldsymbol{x}_1$, so $P_{10}(\boldsymbol{x}_1|\boldsymbol{x}_0)\neq\delta(\boldsymbol{x}_1-\boldsymbol{x}_1(\boldsymbol{x}_0))$ is permitted in general.

8. References

[1] Kohonen T, Self organisation and associative memory, Springer-Verlag, 1984

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

[3] Luttrell S P, Hierarchical vector quantisation, Proc. IEE Part I, 1989, **136**, 405-413

[4] Luttrell S P, Hierarchical self-organising networks, Proc. 1st IEE Conf. on Artificial Neural Networks, London, 1989, 2-6

[5] Luttrell S P, Derivation of a class of training algorithms, IEEE Trans. NN, 1990, **1**, 229-232

[6] Ritter H and Schulten K, Kohonen's self-organising maps: exploring their computational capabilities, Proc. 2nd IEEE Int. Conf. on Neural Networks, San Diego, 1988, **1**, 109-116

[7] Kumazawa H, Kasahara M and Namekawa T, A construction of VQs for noisy channels, Electronics and Engineering in Japan, 1984, **67B**, 39-47

[8] Farvardin N, A study of vector quantisation for noisy channels, IEEE Trans. IT, 1990, **36**, 799-809

[9] Farvardin N and Vaishampayan V, On the performance and complexity of channel-optimised VQs, IEEE Trans. IT, 1991, **37**, 155-160.

[10] Luttrell S P, Code vector density in topographic mappings: scalar case, IEEE Trans. NN, 1991, **2**, 427-436

[11] Ritter H, Asymptotic level density for a class of vector quantisation processes, IEEE Trans. NN, 1991, **2**, 173-175

[12] Luttrell S P, Self-supervised training of hierarchical VQs, Proc. 2nd IEE Conf. on Artificial Neural Networks, Bournemouth, 1991, 5-9

[13] Luttrell S P, Self-supervision in multilayer adaptive networks, Proc. IEE Part F, 1992, **139**(6), 371-377

[14] Yair E, Zeger K and Gersho A, Competitive learning and soft competition for vector quantiser design, IEEE Trans. SP, 1992, **40**, 294-309

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