Stochastic Vector Quantization, and Stochastic VQ With State Feedback Using Neural Networks

Earl Levine
Dept. of Electrical Engineering, Stanford University, CA 94305
email: earl@isl.stanford.edu

1 Introduction

Memoryless vector quantization (VQ) is the process of representing a continuous-valued vector by a vector function of a discrete-valued index. Ideally the chosen vector is a low-error representation of the original vector. In "plain" Lloyd-optimal VQ, an encoder deterministically chooses the index of the lowest-error vector in a collection of a finite number of vectors (the codebook). Then a decoder simply looks up that code vector in the codebook using the encoded index. A simple training scheme can be used to design an optimal codebook for this scheme.

A novel system which performs VQ stochastically is introduced. This system uses an encoder which produces, for a given vector to be quantized, a probability distribution over the N indices, then randomly chooses one of the N indices using that distribution. The structure used to compute the distribution is a feedforward neural network classifier[4]. The decoder is a simple codebook lookup exactly as in plain VQ. The performance of this memoryless stochastic scheme must be suboptimal.

By taking the error to be minimized as the expected value, over the set of random choices, of the error between original vector and decoder output, it is possible to find the error gradient over the encoder/classifier's outputs and decoder's codebook vectors. Then gradient descent (backpropagation) may be used to train all the system's parameters in order to minimize the error.

When the data to be quantized is a correlated sequence of vectors, each vector could be quantized separately by a memoryless VQ system. However, performance can be improved in theory by using some state information about the past. The usual state-feedback extension of plain VQ is Finite-State Vector Quantization (FSVQ). Unfortunately, there is no known optimal training procedure for FSVQ.

The stochastic VQ scheme can also be extended to use feedback state information. Unlike for FSVQ, the optimal training scheme (given the structure of the system) is known. By using gradient descent (backpropagation-through-time), the system can be trained to optimize its parameters.

Experimental results show that this stochastic system with state-feedback can perform up to 0.45 dB better than FSVQ on a Gauss-Markov source.

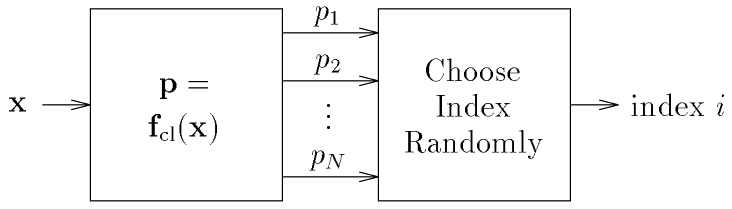

Figure 1: Encoder for Stochastic VQ System

2 Stochastic Vector Quantization

The system introduced in this section performs the same task as plain memoryless VQ, that is, generates a representation of a continuous-valued vector using a discrete index. As in plain VQ, once an index is generated by the encoder, the decoder simply outputs the vector found at the indexed location in the codebook table. However the encoder uses an entirely different method to generate that index.

Specifically, as illustrated in Figure 1, the encoder computes a probability distribution $\mathbf{p}$ over indices 1 through N using the formula

$$\begin{aligned} \mathbf{p} &= \mathbf{f}_{\mathrm{cl}}(\mathbf{x}) \\ &= \mathbf{f}_{\mathrm{normexp}}\big(\mathbf{b}_{\mathrm{u}} + \mathbf{W}_{\mathrm{u}}\mathbf{f}_{\mathrm{tanh}}(\mathbf{b}_{\mathrm{l}} + \mathbf{W}_{\mathrm{l}}\mathbf{x})\big), \text{ where} \end{aligned} \tag{1}$$

$$\mathbf{f}_{\mathrm{normexp}}(\mathbf{y}) = \frac{1}{\sum_{i=1}^{N} \exp(y_i)} \begin{pmatrix} \exp(y_1) \\ \exp(y_2) \\ \vdots \\ \exp(y_N) \end{pmatrix} \tag{2}$$

and where $\mathbf{f}_{\mathrm{tanh}}(\mathbf{z})$ is the hyperbolic tangent function tanh() applied component-wise to each element of $\mathbf{z}$. The function $\mathbf{f}_{\mathrm{cl}}(\mathbf{x})$ is of the type used for multinomial classification problems in the neural network field[4]. The parameters $\mathbf{b}_{\mathrm{u}}$, $\mathbf{b}_{\mathrm{l}}$, $\mathbf{W}_{\mathrm{u}}$, and $\mathbf{W}_{\mathrm{l}}$ are collectively known as the "weights" of the system.[1] Note that if the weights and $\mathbf{x}$ are finite, since the tanh() function's range is finite, it can be shown that all the elements of $\mathbf{p}$ are limited to the range $(0, 1)$. In addition, the elements of $\mathbf{p}$ sum to 1. These properties allow $\mathbf{p}$ to be interpreted as a probability distribution.

The distribution $\mathbf{p}$ is used by a random index generator to choose an index i. Then $\hat{\mathbf{x}}$, the quantized representation of $\mathbf{x}$, is generated by looking up the ith vector $\hat{\mathbf{x}}_i$ in the codebook $\mathbf{C} = \big(\hat{\mathbf{x}}_1 \;\; \hat{\mathbf{x}}_2 \;\; \ldots \;\; \hat{\mathbf{x}}_N\big)$. Since the behavior of the entire system is stochastic (unlike plain VQ), the reproduction error for a given vector is an expected value over the random choice of index.

[1]The subscripts "u" and "l" refer to the "upper layer" and "lower layer" in neural network terminology.

2.1 Suboptimality

The decoder codebook could satisfy the centroid condition for the encoder. However, it can be shown that this encoder structure can never satisfy the nearest-neighbor condition exactly. Lloyd-optimality would require the distribution $\mathbf{p}$ to be deterministic[2] for a single index over each Voronoi region of the domain of $\mathbf{x}$. That is, all the elements of $\mathbf{p}$ would be exactly 0, except for a single element which would be exactly 1. But two facts about $\mathbf{f}_{\mathrm{cl}}(\mathbf{x})$ clearly conflict with this requirement: the elements of $\mathbf{p}$ can approach 0 or 1 but can never exactly equal either value, and the Lloyd-optimal discontinuous transition in the value of $\mathbf{p}$ at the boundaries between Voronoi regions can only be approximated by the continuous function $\mathbf{f}_{\mathrm{cl}}(\mathbf{x})$. For example, if the distortion function is mean squared error, the Voronoi regions are convex polytopes; the angular junctions of 3 or more polytopes are difficult to approximate well with an $\mathbf{f}_{\mathrm{cl}}(\mathbf{x})$ with a small number of parameters. It is notable however that for the special case $N = 2$ there are no such junctions, and the only region boundary is a single hyperplane which can be asymptotically achieved with a minimal-parameter $\mathbf{f}_{\mathrm{cl}}(\mathbf{x})$.

By itself this suboptimal scheme may not be useful. However the state-feedback extension of this scheme (discussed later in Section 3) can outperform the usual state-feedback extension of Lloyd-optimal VQ.

2.2 Randomness in Quantization

The use of randomness in quantization is not a new idea. Scalar dithering is a random technique that has long been used in scalar waveform coding[2]. This stochastic VQ scheme might be thought of as a kind of "vector dithering". Both techniques use a random choice of index when encoding. Unlike plain VQ, this results in error which is not a purely deterministic function of the signal being quantized. For signals which are to be perceived by humans, such as sound waveforms or pixel mapped images, error which is not perfectly correlated with the original signal may be less perceptible than correlated error, even if the nominal distortion measure is calculated to be higher. For example in scalar quantization of sound waveforms with a low value of N, the error from a plain VQ system might be heard as degrading "fuzz" distortion, while the error from a quantization system with dithering might be perceived as a "clean" signal and white noise heard together. An appropriate perceptual distortion measure for such signals would correctly model these effects by penalizing for correlation of error and signal, which would imply that the optimal VQ scheme must be stochastic.

2.3 Gradient Descent Training

The encoder's weight parameters and the decoder's codebook vectors are system parameters which need to be trained. At this point the distortion function is required to be continuous and differentiable. (Mean squared error is an example of such a function.) Then for this scheme, the total distortion to be minimized over a training

[2]This assumes a deterministic distortion function; see Section 2.2.

set of vectors $\mathbf{x}$ is a continuous and differentiable function of all the parameters. This allows the use of an iterative gradient descent algorithm to seek a set of parameters which minimize distortion.

An initial set of parameters is selected (for example, randomly) and then the parameters are iteratively adjusted along the gradient of error ε by

$$\mathbf{w}_{\text{next}} = \mathbf{w}_{\text{current}} - \mu \frac{\partial \varepsilon}{\partial \mathbf{w}_{\text{current}}}, \tag{3}$$

where μ is a positive constant known as the "learning rate," and $\mathbf{w}$ represents all system parameters. With a sufficiently small μ, $\mathbf{w}$ will eventually settle in a local minimum of ε, which is also usually[3] the global minimum.

The gradient over the parameters must be found. First consider the distortion ε for a single training set vector $\mathbf{x}$, which is computed as an expected value

$$\varepsilon = \sum_{i=1}^{N} p_i d(\mathbf{x}, \hat{\mathbf{x}}_i) \tag{4}$$

where $d(\mathbf{x}, \hat{\mathbf{x}}_i)$ is the given distortion function (for example, $\sum_{i=1}^{N}(\mathbf{x}-\hat{\mathbf{x}}_i)^2$). Therefore

$$\frac{\partial \varepsilon}{\partial \hat{\mathbf{x}}_i} = p_i \frac{\partial d(\mathbf{x}, \hat{\mathbf{x}}_i)}{\partial \hat{\mathbf{x}}_i}, \qquad \forall i \tag{5}$$

and

$$\frac{\partial \varepsilon}{\partial p_i} = d(\mathbf{x}, \hat{\mathbf{x}}_i), \qquad \forall i. \tag{6}$$

Equation 5 gives the gradient for the codebook parameters, and Equation 6 can be used to find the gradient of the encoder's weight parameters. From $\frac{\partial \varepsilon}{\partial p_i}$ the chain rule of differentiation may be used to find the gradient of ε at the inputs of the $\mathbf{f}_{\text{normexp}}()$ function. This gradient in turn can be used to find $\frac{\partial \varepsilon}{\partial \mathbf{b}_{\text{u}}}$, $\frac{\partial \varepsilon}{\partial \mathbf{W}_{\text{u}}}$, and the gradient at the outputs of the $\mathbf{f}_{\text{tanh}}()$ function. Again the chain rule is used to find the gradient at the inputs of the $\mathbf{f}_{\text{tanh}}()$ function, from which $\frac{\partial \varepsilon}{\partial \mathbf{b}_{\text{l}}}$, $\frac{\partial \varepsilon}{\partial \mathbf{W}_{\text{l}}}$, and $\frac{\partial \varepsilon}{\partial \mathbf{x}}$ (if desired) can be found. This method of repeatedly applying the chain rule to propagate the computation of gradient further back (toward the ultimate input of the function) is known as "backpropagation" in the neural network field. After backpropagation the gradient over all parameters is known.

Since there are usually multiple vectors $\mathbf{x}$ in the training set, the expected value of the gradient over the entire set is used for gradient descent. This is simply found as an average (weighted by any prior probabilities over the set) of the gradients computed for each individual $\mathbf{x}$.

[3]As the number of parameters increases the likelihood of settling in a non-global minimum becomes increasingly smaller; non-global minima are usually only a concern for very small numbers of parameters, say, less than 20. Of course this is all dependent on the specific error function $\varepsilon = \mathbf{f}_{\varepsilon}(\mathbf{w})$.

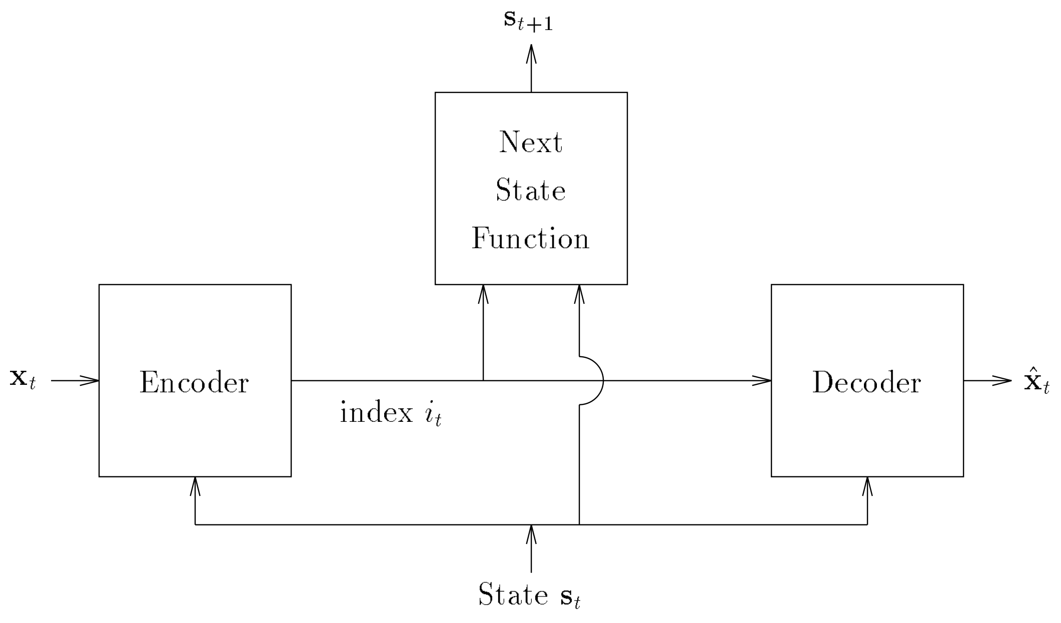

Figure 2: VQ with State Feedback

3 Stochastic VQ With State Feedback

Many sources to be quantized have memory, that is, there is correlation between different samples. Unlike memoryless VQ, a system which can use internal state memory about the past can take advantage of that correlation to improve performance. Figure 2 shows a general structure for such a system, which is known as a *recursive coding system.*

Finite-State Vector Quantization (FSVQ) is a recursive extension to memoryless Lloyd-optimal VQ. In FSVQ the state is a discrete index from 1 to N_s which selects one codebook from a set of N_s codebooks. That codebook is then used by the encoder and decoder in the usual Lloyd-optimal way. Nonoptimal *ad hoc* methods used to train the codebooks and next-state function can yield improvements over memoryless VQ, but there is no known optimal training method for the FSVQ codebooks nor for the next-state function[5].

However a recurrent extension to the stochastic VQ scheme of Section 2 can be trained to optimality. As the results of Section 4 show, this optimally trained recursive system based on a suboptimal memoryless structure can sometimes outperform a non-optimally trained recursive system based on an optimal memoryless structure (FSVQ).

Figure 3 shows the structure of the system. The state $\mathbf{s}_t$ is a continuous-valued vector. The encoder is like the stochastic encoder of Section 2 except that $\mathbf{f}_{\mathrm{cl}}()$ must be expanded to accommodate the additional input $\mathbf{s}_t$:

$$\begin{aligned}\mathbf{p}_t &= \mathbf{f}_{\mathrm{cl}}(\mathbf{x}_t, \mathbf{s}_t) \\ &= \mathbf{f}_{\mathrm{normexp}}\big(\mathbf{b}_{\mathrm{u}} + \mathbf{W}_{\mathrm{u}}\mathbf{f}_{\mathrm{tanh}}(\mathbf{b}_{\mathrm{l}} + \mathbf{W}_{\mathrm{l,x}}\mathbf{x}_t + \mathbf{W}_{\mathrm{l,enc,s}}\mathbf{s}_t)\big)\end{aligned} \tag{7}$$

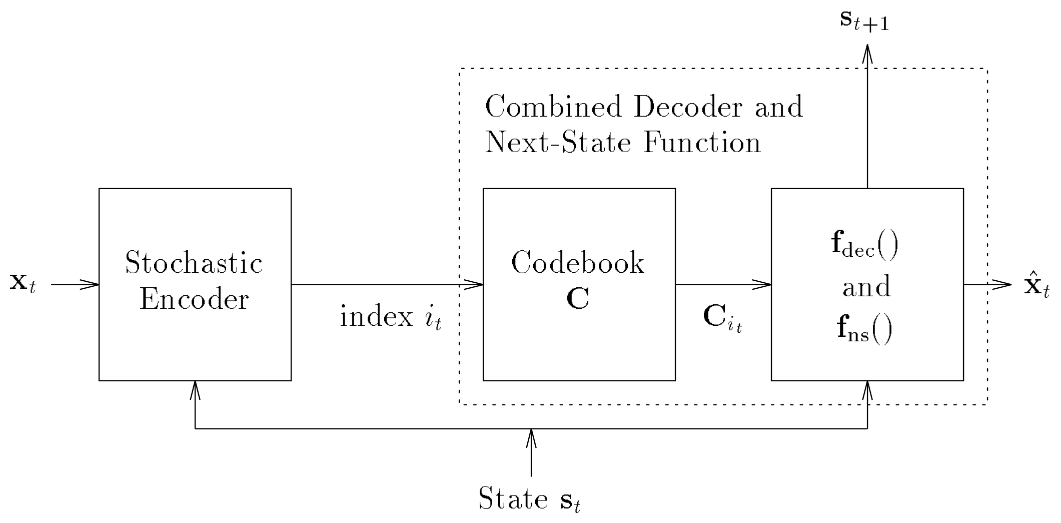

Figure 3: Stochastic VQ with State Feedback

In order to reduce complexity, the decoder and the next-state function are combined. As in the memoryless case, the index i_t is used to select the vector $\mathbf{C}_{i_t}$ from a codebook matrix $\mathbf{C}$. Then $\hat{\mathbf{x}}_t$ and $\mathbf{s}_{t+1}$ are found by

$$\begin{aligned}\mathbf{s}_{t+1} &= \mathbf{f}_{\text{ns}}(\mathbf{C}_{i_t}, \mathbf{s}_t) \\ &= \mathbf{f}_{\text{tanh}}\big(\mathbf{b}_{\text{ns}} + \mathbf{W}_{\text{ns}}\mathbf{f}_{\text{tanh}}(\mathbf{b}_{\text{l,dec}} + \mathbf{W}_{\text{l,dec,}\mathbf{C}}\mathbf{C}_{i_t} + \mathbf{W}_{\text{l,dec,}\mathbf{s}}\mathbf{s}_t)\big)\end{aligned} \tag{8}$$

and

$$\begin{aligned}\hat{\mathbf{x}}_t &= \mathbf{f}_{\text{dec}}(\mathbf{C}_{i_t}, \mathbf{s}_t) \\ &= \mathbf{b}_{\text{bias}} + \mathbf{W}_{\text{scale}}\mathbf{f}_{\text{tanh}}\big(\mathbf{b}_{\text{u,dec}} + \mathbf{W}_{\text{u,dec}}\mathbf{f}_{\text{tanh}}(\mathbf{b}_{\text{l,dec}} + \mathbf{W}_{\text{l,dec,}\mathbf{C}}\mathbf{C}_{i_t} + \mathbf{W}_{\text{l,dec,}\mathbf{s}}\mathbf{s}_t)\big)\end{aligned} \tag{9}$$

where $\mathbf{W}_{\text{scale}}$ is a diagonal matrix. The terms $\mathbf{b}_{\text{bias}}$ and $\mathbf{W}_{\text{scale}}$ allow the ranges of the components of $\hat{\mathbf{x}}_t$ to be unrestricted. The argument of the common term

$$\mathbf{f}_{\text{tanh}}(\mathbf{b}_{\text{l,dec}} + \mathbf{W}_{\text{l,dec,}\mathbf{C}}\mathbf{C}_{i_t} + \mathbf{W}_{\text{l,dec,}\mathbf{s}}\mathbf{s}_t) \tag{10}$$

can be reformulated by defining a new codebook $\mathbf{C}'$ such that

$$\mathbf{C}' = (\mathbf{b}_{\text{l,dec}}, \mathbf{b}_{\text{l,dec}}, \ldots \mathbf{b}_{\text{l,dec}}) + \mathbf{W}_{\text{l,dec,}\mathbf{C}}\mathbf{C}, \tag{11}$$

which allows the computation of merely

$$\mathbf{f}_{\text{tanh}}(\mathbf{C}'_{i_t} + \mathbf{W}_{\text{l,dec,}\mathbf{s}}\mathbf{s}_t) \tag{12}$$

further reducing the implementation complexity.

3.1 Gradient Descent Training

The parameters of the system which need to be trained are $\mathbf{b}_{\text{u}}$, $\mathbf{W}_{\text{u}}$, $\mathbf{b}_{\text{l}}$, $\mathbf{W}_{\text{l,}\mathbf{x}}$, and $\mathbf{W}_{\text{l,enc,}\mathbf{s}}$ from Equation 7; $\mathbf{b}_{\text{ns}}$ and $\mathbf{W}_{\text{ns}}$ from Equation 8; $\mathbf{b}_{\text{bias}}$, $\mathbf{W}_{\text{scale}}$, $\mathbf{b}_{\text{u,dec}}$, and $\mathbf{W}_{\text{u,dec}}$ from Equation 9; $\mathbf{C}'$ and $\mathbf{W}_{\text{l,dec,}\mathbf{s}}$ from Equation 12; and an initial state vector[4]

[4]The value of $\mathbf{s}_1$ may be established arbitrarily (for example $\mathbf{0}$) but the additional complexity to train the value to optimality is negligible.

$\mathbf{s}_1$. Let $\mathbf{w}$ represent all those system parameters except $\mathbf{s}_1$. Assuming a continuous and differentiable distortion function, since the total distortion is a continuous function of $\mathbf{w}$ and $\mathbf{s}_1$, a gradient descent algorithm can be used to train $\mathbf{w}$ and $\mathbf{s}_1$.

Consider a sequence of T vectors $\{\mathbf{x}_1, \mathbf{x}_2, \ldots \mathbf{x}_T\}$. Define the total current and future distortion at step t given this sequence, given that the current state is $\mathbf{s}_t$, and given that the system parameters are $\mathbf{w}$, as $\varepsilon_t(\mathbf{s}_t, \mathbf{w})$; that is

$$\varepsilon_t(\mathbf{s}_t, \mathbf{w}) = \sum_{i=1}^{N} \mathbf{p}_{i_t}\big(d(\mathbf{x}_t, \hat{\mathbf{x}}_t) + \varepsilon_{t+1}(\mathbf{s}_{t+1}, \mathbf{w})\big). \tag{13}$$

The total distortion over the entire sequence, which is the quantity to be minimized, is

$$\varepsilon = \varepsilon_1(\mathbf{s}_1, \mathbf{w}). \tag{14}$$

The gradients $\frac{\partial \varepsilon}{\partial \mathbf{w}}$ and $\frac{\partial \varepsilon}{\partial \mathbf{s}_1}$ need to be found. Omitting some arguments for readability, differentiation of Equation 13 gives

$$\begin{aligned}\frac{\partial \varepsilon_t(\mathbf{s}_t, \mathbf{w})}{\partial \mathbf{w}} = \sum_{i=1}^{N} \Bigg(& \frac{\partial \mathbf{f}_{\mathrm{cl},i}(\mathbf{s}_t, \mathbf{x}_t, \mathbf{w})}{\partial \mathbf{w}} \big(d(\mathbf{x}_t, \hat{\mathbf{x}}_t) + \varepsilon_{t+1}(\mathbf{s}_{t+1}, \mathbf{w})\big) \\ & + \mathbf{p}_{i_t} \Bigg(\frac{\partial \mathbf{f}_{\mathrm{dec}}(\mathbf{s}_t, i, \mathbf{w})}{\partial \mathbf{w}} \frac{\partial d(\mathbf{x}_t, \mathbf{f}_{\mathrm{dec}}())}{\partial \mathbf{f}_{\mathrm{dec}}} \\ & + \frac{\partial \mathbf{f}_{\mathrm{ns}}(\mathbf{s}_t, i, \mathbf{w})}{\partial \mathbf{w}} \frac{\partial \varepsilon_{t+1}(\mathbf{s}_{t+1}, \mathbf{w})}{\partial \mathbf{s}_{t+1}} + \frac{\partial \varepsilon_{t+1}(\mathbf{s}_{t+1}, \mathbf{w})}{\partial \mathbf{w}} \Bigg)\Bigg)\end{aligned} \tag{15}$$

and

$$\begin{aligned}\frac{\partial \varepsilon_t(\mathbf{s}_t, \mathbf{w})}{\partial \mathbf{s}_t} = \sum_{i=1}^{N} \Bigg(& \frac{\partial \mathbf{f}_{\mathrm{cl},i}(\mathbf{s}_t, \mathbf{x}_t, \mathbf{w})}{\partial \mathbf{s}_t} \big(d(\mathbf{x}_t, \hat{\mathbf{x}}_t) + \varepsilon_{t+1}(\mathbf{s}_{t+1}, \mathbf{w})\big) \\ & + \mathbf{p}_{i_t} \Bigg(\frac{\partial \mathbf{f}_{\mathrm{dec}}(\mathbf{s}_t, i, \mathbf{w})}{\partial \mathbf{s}_t} \frac{\partial d(\mathbf{x}_t, \mathbf{f}_{\mathrm{dec}}())}{\partial \mathbf{f}_{\mathrm{dec}}} + \frac{\partial \mathbf{f}_{\mathrm{ns}}(\mathbf{s}_t, i, \mathbf{w})}{\partial \mathbf{s}_t} \frac{\partial \varepsilon_{t+1}(\mathbf{s}_{t+1}, \mathbf{w})}{\partial \mathbf{s}_{t+1}} \Bigg)\Bigg)\end{aligned} \tag{16}$$

The term $\frac{\partial d(\mathbf{x}_t, \mathbf{f}_{\mathrm{dec}}())}{\partial \mathbf{f}_{\mathrm{dec}}}$ is computed as a derivative of the distortion function. The terms $\frac{\partial \mathbf{f}_{\mathrm{cl},i}(\mathbf{s}_t, \mathbf{x}_t, \mathbf{w})}{\partial \mathbf{w}}$ and $\frac{\partial \mathbf{f}_{\mathrm{cl},i}(\mathbf{s}_t, \mathbf{x}_t, \mathbf{w})}{\partial \mathbf{s}_t}$, $\frac{\partial \mathbf{f}_{\mathrm{dec}}(\mathbf{s}_t, i, \mathbf{w})}{\partial \mathbf{w}}$ and $\frac{\partial \mathbf{f}_{\mathrm{dec}}(\mathbf{s}_t, i, \mathbf{w})}{\partial \mathbf{s}_t}$, and $\frac{\partial \mathbf{f}_{\mathrm{ns}}(\mathbf{s}_t, i, \mathbf{w})}{\partial \mathbf{w}}$ and $\frac{\partial \mathbf{f}_{\mathrm{ns}}(\mathbf{s}_t, i, \mathbf{w})}{\partial \mathbf{s}_t}$ are computed by chain-rule differentiation (backpropagation) of their respective functions at the given input values.

Equations 13, 15, and 16 together define a recursive relationship which allows one to find $\varepsilon_t(\mathbf{s}_t, \mathbf{w})$, $\frac{\partial \varepsilon_t(\mathbf{s}_t, \mathbf{w})}{\partial \mathbf{w}}$, and $\frac{\partial \varepsilon_t(\mathbf{s}_t, \mathbf{w})}{\partial \mathbf{s}_t}$ from their corresponding values at step $t+1$. Note that $\varepsilon_{T+1}(\mathbf{s}_{T+1}, \mathbf{w}) = 0$, therefore $\frac{\partial \varepsilon_{T+1}(\mathbf{s}_{T+1}, \mathbf{w})}{\partial \mathbf{w}} = \mathbf{0}$ and $\frac{\partial \varepsilon_{T+1}(\mathbf{s}_{T+1}, \mathbf{w})}{\partial \mathbf{s}_{T+1}} = \mathbf{0}$, which provides initial conditions for recursion. After recursion all the way back to $t = 1$, $\frac{\partial \varepsilon}{\partial \mathbf{w}} = \frac{\partial \varepsilon_1(\mathbf{s}_1, \mathbf{w})}{\partial \mathbf{w}}$ and $\frac{\partial \varepsilon}{\partial \mathbf{s}_1} = \frac{\partial \varepsilon_1(\mathbf{s}_1, \mathbf{w})}{\partial \mathbf{s}_1}$ are known. The method of recursively calculating backward in t to find the error gradient is known in the neural network field as backpropagation-through-time (BPTT)[4, 3]. As usual, for multiple sequences in the training set, each sequence's gradient is calculated individually and then the gradients are averaged over the set.

3.2 Stochastic Gradient Descent Training

The recursive form of Equations 13, 15, and 16 implies that, when beginning with a given initial state $\mathbf{s}_1$, at step t there are N^t next states $\mathbf{s}_{t+1}$ over which terms to be summed must be calculated. This means that a total number of $\sum_{t=1}^{T} N^t$ such terms must be calculated in order to find the gradients as described in Section 3.1. Since[5] for large T, $\sum_{t=1}^{T} N^t \approx \frac{N^{T+1}}{N-1}$, the growth in complexity is nearly exponential in T; therefore computation of the gradients using this method is prohibitively expensive for nontrivial values of T.

One solution to this problem is to compute an estimate of the true gradients by stochastically choosing a smaller number of indices at each t and $\mathbf{s}_t$. In the extreme, if only a single index is chosen at each step t, the number of terms to be computed is $\sum_{t=1}^{T} 1^t = T$, a much more reasonable complexity than $\frac{N^{T+1}}{N-1}$.

To compute these estimates, given state $\mathbf{s}_t$ at step t, for each trial choose a random value j_t of the index using distribution $q_{t,1}, q_{t,2}, \ldots, q_{t,N}$ and calculate $\hat{\mathbf{x}}_t$ and $\mathbf{s}_{t+1}$ using that index. Equations 13, 15, and 16 may then be re-expressed as expected values over many such trials:

$$\varepsilon_t(\mathbf{s}_t, \mathbf{w}) = \mathrm{E}_{j_t}\left[\frac{\mathbf{p}_{i_t}}{\mathbf{q}_{j_t}}\left(d(\mathbf{x}_t, \hat{\mathbf{x}}_t) + \varepsilon_{t+1}(\mathbf{s}_{t+1}, \mathbf{w})\right)\right], \tag{17}$$

$$\begin{aligned}\frac{\partial \varepsilon_t(\mathbf{s}_t, \mathbf{w})}{\partial \mathbf{w}} = \mathrm{E}_{j_t}\Bigg[& \frac{1}{\mathbf{q}_{j_t}} \frac{\partial \mathbf{f}_{\mathrm{cl},i}(\mathbf{s}_t, \mathbf{x}_t, \mathbf{w})}{\partial \mathbf{w}} \left(d(\mathbf{x}_t, \hat{\mathbf{x}}_t) + \varepsilon_{t+1}(\mathbf{s}_{t+1}, \mathbf{w})\right) \\ & + \frac{\mathbf{p}_{i_t}}{\mathbf{q}_{j_t}}\Bigg(\frac{\partial \mathbf{f}_{\mathrm{dec}}(\mathbf{s}_t, i, \mathbf{w})}{\partial \mathbf{w}} \frac{\partial d(\mathbf{x}_t, \mathbf{f}_{\mathrm{dec}}())}{\partial \mathbf{f}_{\mathrm{dec}}} \\ & \qquad + \frac{\partial \mathbf{f}_{\mathrm{ns}}(\mathbf{s}_t, i, \mathbf{w})}{\partial \mathbf{w}} \frac{\partial \varepsilon_{t+1}(\mathbf{s}_{t+1}, \mathbf{w})}{\partial \mathbf{s}_{t+1}} + \frac{\partial \varepsilon_{t+1}(\mathbf{s}_{t+1}, \mathbf{w})}{\partial \mathbf{w}} \Bigg)\Bigg],\end{aligned} \tag{18}$$

and

$$\begin{aligned}\frac{\partial \varepsilon_t(\mathbf{s}_t, \mathbf{w})}{\partial \mathbf{s}_t} = \mathrm{E}_{j_t}\Bigg[& \frac{1}{\mathbf{q}_{j_t}} \frac{\partial \mathbf{f}_{\mathrm{cl},i}(\mathbf{s}_t, \mathbf{x}_t, \mathbf{w})}{\partial \mathbf{s}_t} \left(d(\mathbf{x}_t, \hat{\mathbf{x}}_t) + \varepsilon_{t+1}(\mathbf{s}_{t+1}, \mathbf{w})\right) \\ & + \frac{\mathbf{p}_{i_t}}{\mathbf{q}_{j_t}}\left(\frac{\partial \mathbf{f}_{\mathrm{dec}}(\mathbf{s}_t, i, \mathbf{w})}{\partial \mathbf{s}_t} \frac{\partial d(\mathbf{x}_t, \mathbf{f}_{\mathrm{dec}}())}{\partial \mathbf{f}_{\mathrm{dec}}} + \frac{\partial \mathbf{f}_{\mathrm{ns}}(\mathbf{s}_t, i, \mathbf{w})}{\partial \mathbf{s}_t} \frac{\partial \varepsilon_{t+1}(\mathbf{s}_{t+1}, \mathbf{w})}{\partial \mathbf{s}_{t+1}} \right)\Bigg].\end{aligned} \tag{19}$$

Assuming a sufficiently small learning rate, the expected values may be estimated by the bracketed values for a single sequence of indices $\{j_1, j_2, \cdots, j_N\}$. Such estimates effectively add zero-mean noise to the true gradient. The gradient descent algorithm can tolerate such noise, which can even be helpful in avoiding non-global minima[3].

The distributions $\mathbf{q}_t$ must be chosen. Any choice of $\mathbf{q}_t$ with $q_{t,j} \neq 0$ for all j and all t will allow estimates with finite error. For simplicity $\mathbf{q}_t = \mathbf{p}_t$ is chosen for each t.

[5]This is assuming that $N > 1$, without which this system can only produce a single output sequence $\{\hat{\mathbf{x}}_t\}$.

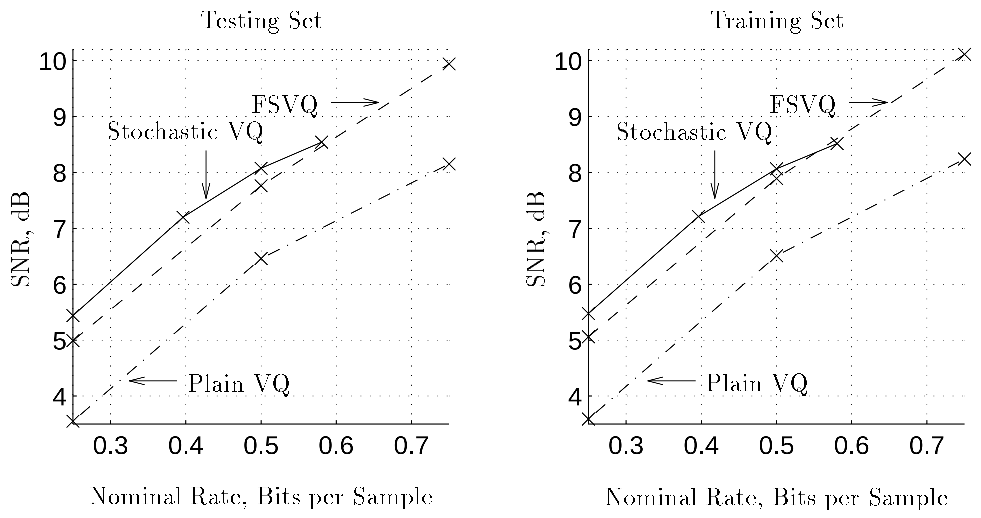

Figure 4: Performance of Stochastic VQ with State-Feedback, FSVQ, and Plain VQ for Gauss-Markov Data

4 Experimental Results

For direct comparison to the FSVQ results of Foster, Gray, and Dunham[5], this stochastic system with state-feedback was trained on a Gauss-Markov source, and experimental rate-distortion performance was observed.

Specifically, the source used was a Gauss-Markov source $\{y_n\}$ defined by

$$y_{n+1} = a y_n + r_n \tag{20}$$

where $\{r_n\}$ is a zero-mean unit-variance independent, identically distributed Gaussian series, and $a = 0.9$. A training sequence of length $128,000$ samples and a separate test sequence of the same length were generated. For each of the sequences $\{\mathbf{x}_t\}$, a sequence of vectors of dimension 4, was made by partitioning $\{y_n\}$ into vectors of 4 consecutive samples, that is $\mathbf{x}_t = \begin{pmatrix} y_{4t} & y_{4t+1} & y_{4t+2} & y_{4t+3} \end{pmatrix}$. Squared difference (MSE) was used as the distortion function. This experimental setup is exactly as in [5].

Figure 4 shows a comparison of performance of stochastic VQ with state-feedback, FSVQ, and plain VQ on the Gauss-Markov testing and training data sets. The rate is reported as nominal rate in bits per sample ($\frac{1}{4}\log_2(N)$) rather than as entropy, in keeping with the results in [5]. The FSVQ results are for "omniscient labeled transition FSVQ," the best of 4 training methods considered in [5].

For the testing set, which predicts the performance on novel data better than does the training set, stochastic VQ can perform up to 1.89 dB better than plain VQ and 0.45 dB better than FSVQ, although the improvements fall off at higher rates. It is also noteworthy that the difference between the training and testing performance ("overfitting" of the training data) is much less for stochastic VQ (less than 0.04 dB) than for FSVQ (0.13 dB or higher).

References

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

[2] N. S. Jayant and P. Noll, *Digital Coding of Waveforms, Principles and Applications to Speech and Video,* Prentice-Hall, 1984.

[3] J. Hertz, A. Krogh, and R. G. Palmer, *Introduction to the Theory of Neural Computation,* Addison Wesley, 1991.

[4] Y. Chauvin and D. E. Rumelhart, editors, *Backpropagation: Theory, Architectures, and Applications,* Lawrence Erlbaum Associates, 1995.

[5] J. Foster, R. M. Gray, and M. O. Dunham, "Finite-State Vector Quantization for Waveform Coding," *IEEE Transactions on Information Theory,* vol. 31, no. 3, May 1985, pp. 348-359.