QUICK INTRODUCTION TO
A VARIETY OF LOSSY SOURCE CODING TECHNIQUES

Outline

- Generic Techniques
  - Fast VQ partitioning
  - Structured VQ
    - Tree-structured VQ (TSVQ)
    - Two-stage and multi-stage quantization
    - Lattice VQ
    - Shape-gain, Polar and Pyramid quantization
    - Transform, subband and wavelet coding
  - Nonblock coders
    - Differential pulse code modulation (DPCM)
    - Trellis quantization
- Techniques for
  - Image coding
  - Video coding
  - Speech coding
  - Audio coding

FAST PARTITIONING FOR UNSTRUCTURED VQ
AKA FAST SEARCH OF UNSTRUCTURED VQ CODEBOOKS

Here we describe several fast methods for finding the closest codevector in a k-dimensional codebook $C = \{\underline{w}_1,\ldots,\underline{w}_M\}$ to a given vector $\underline{x}$. All but the first of these obtain reduced computational complexity at the expense of increased storage.

References: A number of references are listed in R. Gray and D. Neuhoff, "Quantization," *IEEE Trans. Inform. Thy,* Oct. 1998.

1. Partial Distortion

Suppose we have found that the closest codeword to $\underline{x}$ among $\underline{w}_1,\ldots,\underline{w}_{n-1}$ is at distance d from $\underline{x}$, and suppose we are now computing

$$||\underline{x}-\underline{w}_n||^2 = (x_1-w_{n,1})^2 + (x_2-w_{n,2})^2 + (x_3-w_{n,3})^2 + \ldots$$

by successively computing and accumulating the square terms.

If after a number of terms have been accumulated this sum should become larger than d^2, then we know $\underline{w}_n$ is not the closest codeword, so we need not compute any more terms, but move on to computing $||\underline{x}-\underline{w}_{n+1}||^2$.

Since most codewords are quite far from $\underline{x}$, this method frequently saves a lot of computation, typically, 25 to 75%.

2. Successive Narrowing of Search

Initialization: The distances between all pairs of codevectors are precomputed and stored in a table.

a. Choose an initial codevector $\underline{y}$.

b. Eliminate all codevectors $\underline{w}_i$ such that

$$||\underline{w}_i - \underline{y}|| > 2\,||\underline{x} - \underline{y}||.$$

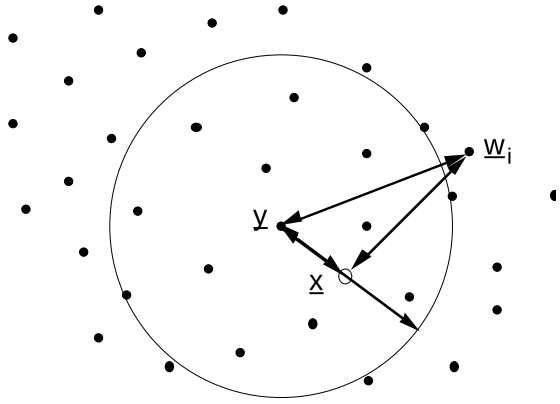
By the triangle inequality

$$||\underline{w}_i - \underline{y}|| \le ||\underline{w}_i - \underline{x}|| + ||\underline{x} - \underline{y}||,$$

so that

$$||\underline{w}_i - \underline{x}|| \ge ||\underline{w}_i - \underline{y}|| - ||\underline{x} - \underline{y}|| > ||\underline{x} - \underline{y}||.$$

$\Rightarrow$ $\underline{w}_i$ could not be closest codeword.



c. Successively search the codevectors not eliminated to find one closer to $\underline{x}$ than $\underline{y}$, which then replaces $\underline{y}$.

d. Eliminate all remain'g cdvects such that $||\underline{w}_i - \underline{y}|| > 2\,||\underline{x} - \underline{y}||$.

e. Repeat steps c. and d. until all codevectors have been considered or eliminated from consideration.

This method reduces # ops/sample at the expense of storage.

There are many variations of and improvements to this method.

3. Coarse-to-Fine Search

In this method, a very fast coarse quantization is done first. Then a full search is performed on a small subset of C.

Initialization: Select a scalar quantizer. Make a table with one row for each cell of the k-fold product quantizer. (Such cells are rectangles.) The row corresponding to a given cell of the k-fold product quantizer contains a list of the indices of all codevectors in C whose Voronoi regions intersect the given cell.

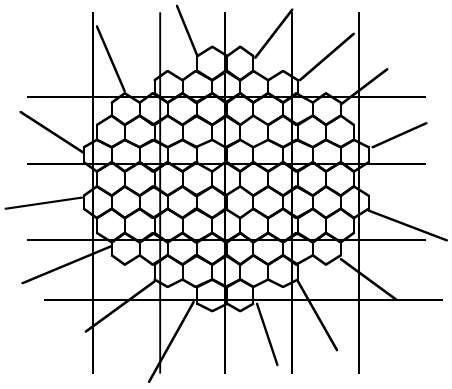
Operation:

a. Scalar quantize each component of $\underline{x}$.

b. Use the sequence of k scalar quantizer outputs to address the table and obtain a set of codevector indices.

c. Compute the distance between $\underline{x}$ and the codevectors whose indices were found in Step b.

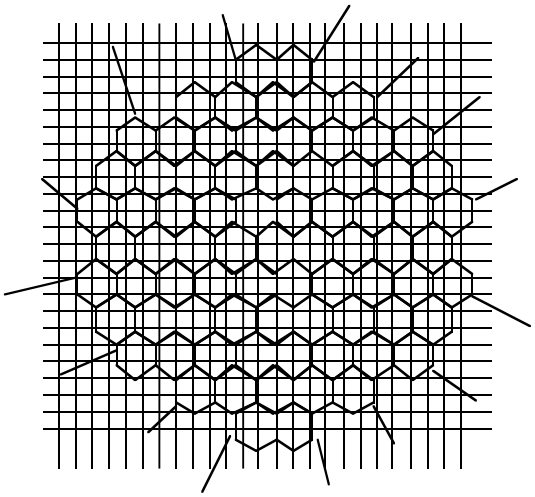
d. Output the index of the cell whose codevector is closest.



There are variations on this method where Step a. is replaced by some other coarse quantization method, e.g. one of the fast structured VQ's to be discussed later.

4. Fine-to-Coarse Search

This is the logical continuation of method 3. This is like coarse-to-fine, except that the rate of the initial scalar quantizer is sufficiently higher than that of the desired VQ that most of the cells of the product quantizer are contained in one and only one Voronoi region of C. In this case, Step c. is eliminated in most cases, or reduced to just a few computations.



5. Hierarchical table lookup.

To be described later.

STRUCTURED VECTOR QUANTIZATION

These are VQ techniques that impose "structure" on the codebook that permits the partitioning to be done in a simpler fashion than brute force full search.

The codebooks of these techniques are suboptimal. For example, they may fail to satisfy one or both of the optimality criteria.

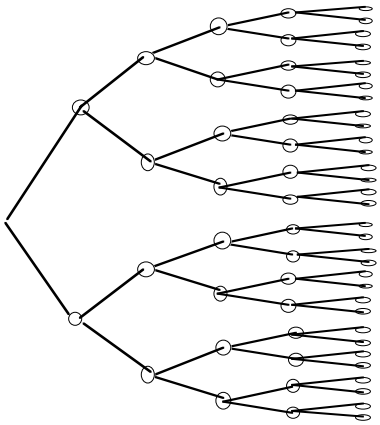
For many of these methods, the partition is not the Voronoi partition for the given codebook.

However, the "structure" often enables such a large reduction in complexity that VQ's with higher dimension become practical. These higher dimensional structured VQ's work better than optimal VQ's of the same complexity, which have smaller dimensions.

Here we just briefly introduce the techniques and how well they work. Later we'll analyze some of them, e.g. with Bennett's integral.

TREE-STRUCTURED VECTOR QUANTIZATION (TSVQ)

Binary Search Tree

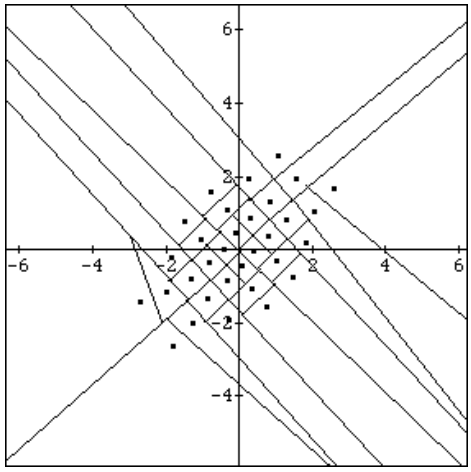


depth = kR

hyperplane test at each node

$$\sum_{i=1}^{k} x_i v_i \;\begin{matrix}<\\>\end{matrix}\; T$$

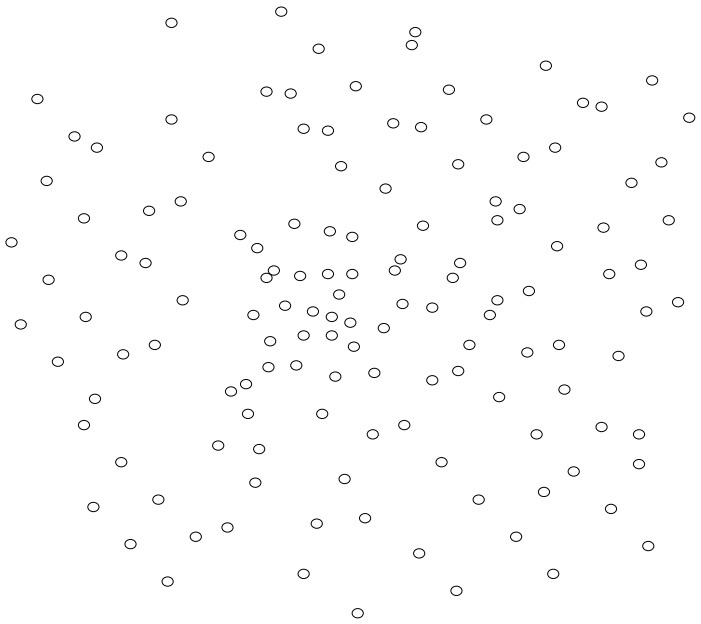
Partition and Codebook

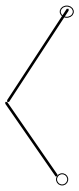


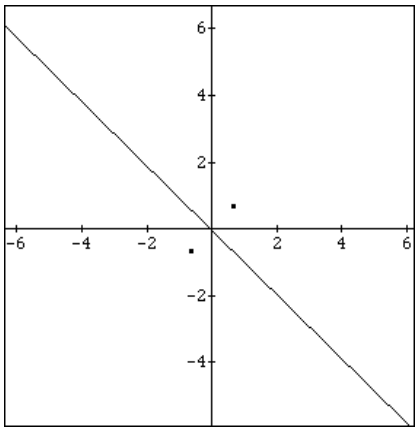
k = 2, N = 32, R = 2.5

- Op's sample reduced from ~ $3N = 3\times 2^{kR}$ to ~$2 \log_2 N = 2kR$
- Still requires large storage -- approximately bkN bits
- Any nonuniform scalar quantizer can be implemented with a tree structure.

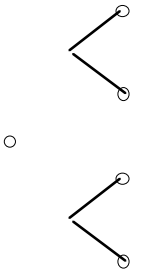
GREEDY DESIGN ALGORITHM

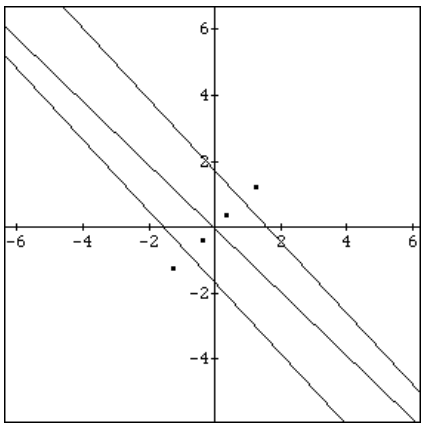




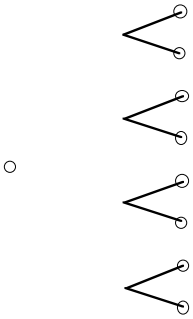


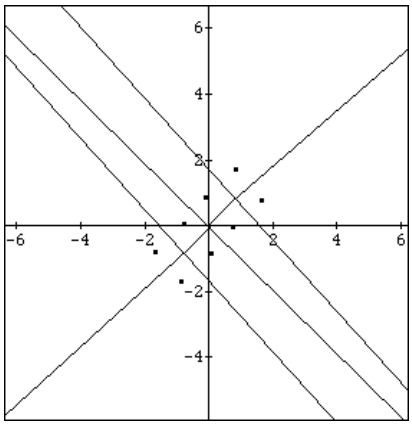
T-9



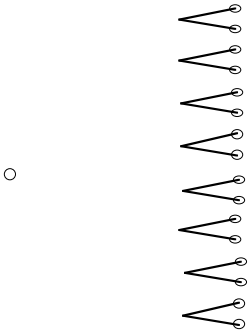


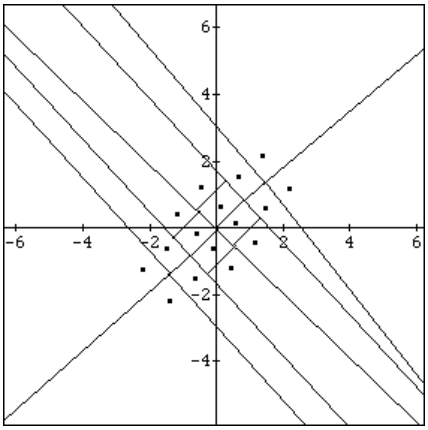
T-10



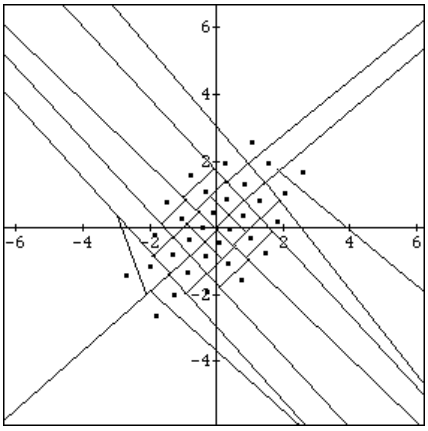


T-11



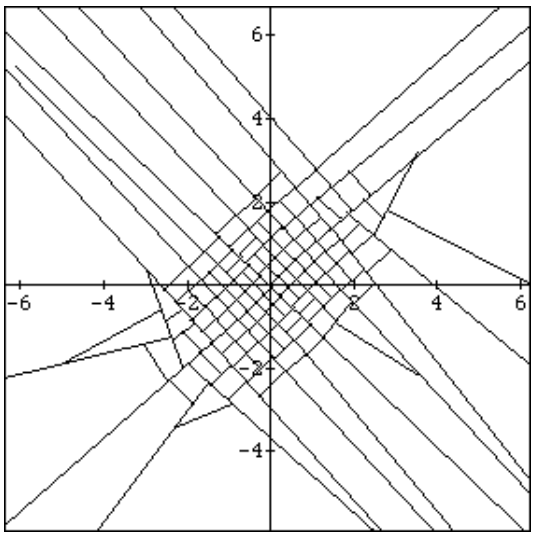


T-12

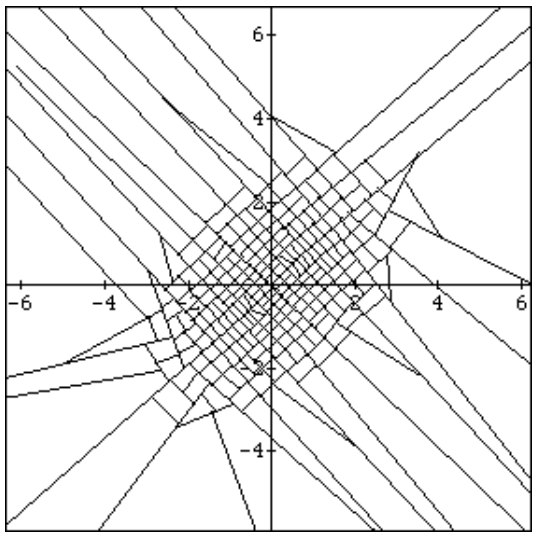


GAUSS-MARKOV SOURCE, CORR. COEFF. $\rho = .9$

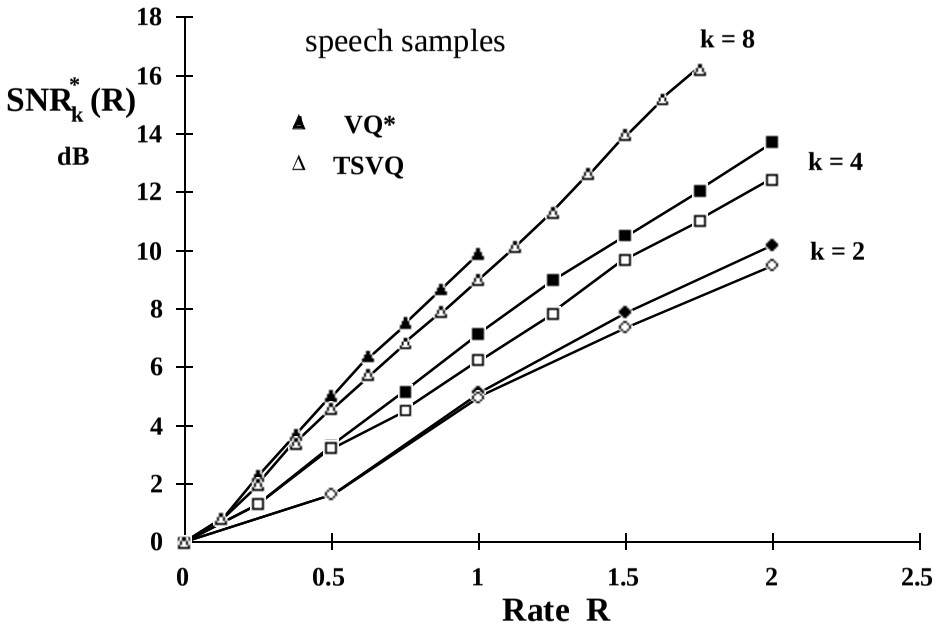
k = 2, N = 128, R = 3.5



k = 2, N = 256, R = 4

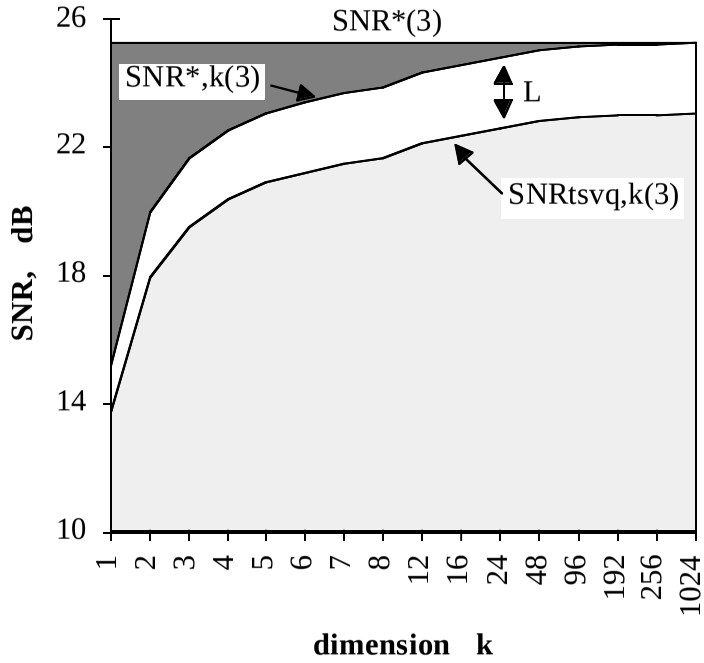


PERFORMANCE OF TSVQ & OPTIMUM VQ



EXAMPLE OF PREDICTED SNR FOR TSVQ

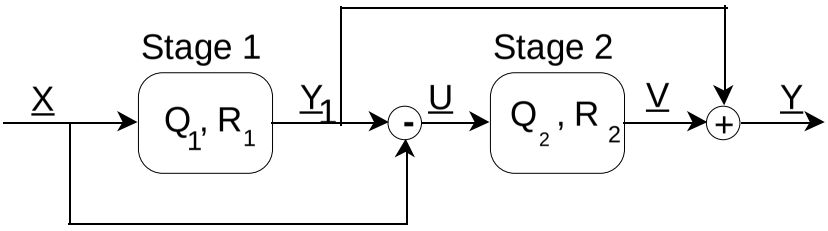
- Gauss AR Source, ρ = .9
- R = 3



OTHER TREE-STRUCTURED VQ'S

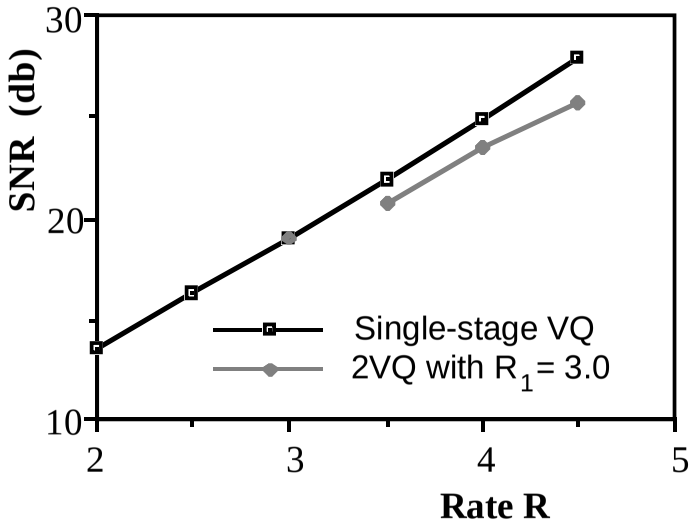
- M-ary trees
  - + Complexity increases from ~2kR to ~3 $\frac{M-1}{\log_2 M}$ kR.
  - + Point density would, apparently, not be improved.
  - + Cell shapes might be improved.
- Variable-depth trees
  - + Tree growing
  - + Tree pruning

TWO-STAGE VECTOR QUANTIZATION (2VQ)



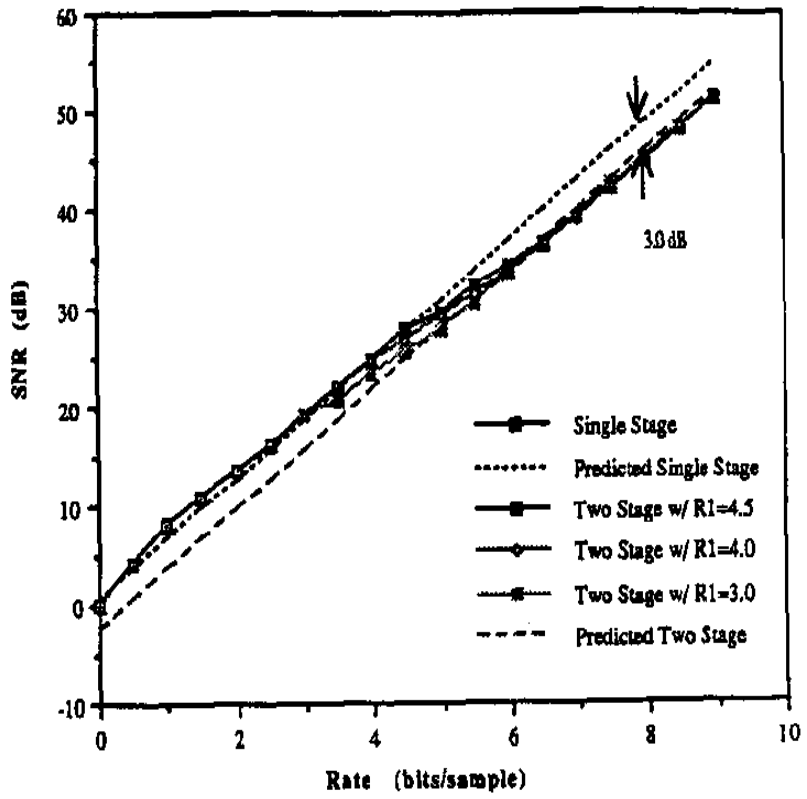
- Operation:
  1. Quantize $\underline{X}$ using Q_1 with rate R_1
  2. Quantize the error/residual $\underline{U} = \underline{X} - \underline{Y}_1$ using Q_2 with rate R_2
- Rate: $R = R_1 + R_2$
- Distortion: $D = \frac{1}{k} E\,||\underline{U} - \underline{V}||^2$ = Stage 2 distortion, because $(\underline{X} - \underline{Y}) = (\underline{U} - \underline{V})$
- Complexity: much lower than single-stage VQ -- M_1+M_2 vs. M_1M_2
- Greedy Design:
  1. design Q_1 to minimize $E||\underline{X}-\underline{Y}_1||^2$,
  2. design Q_2 to minimize $E||\underline{U}-\underline{V}||^2$, e.g. on training sequence of first-stage errors.

- Performance: Example -- Gaussian AR source, $\rho = .9$, VQ dimension $k = 2$



PREDICTED AND ACTUAL SNR

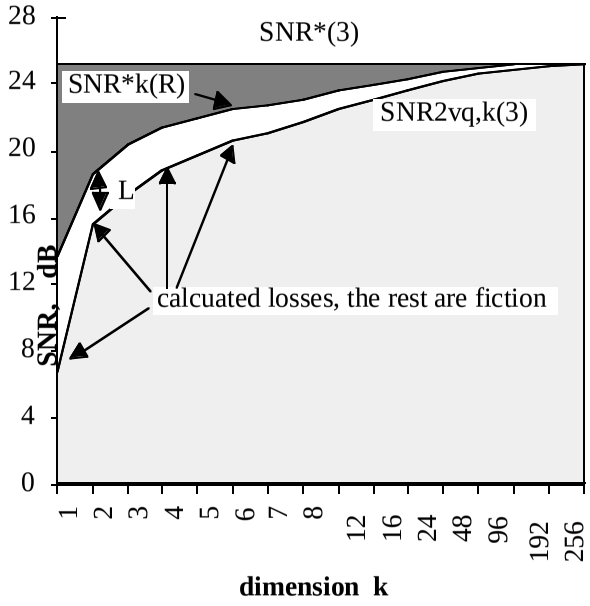
Gaussian AR Source, $\rho = .9$, $k = 2$



EXAMPLE OF PREDICTED SNR FOR 2VQ

- Gaussian AR Source -- corr. coef. $\rho = .9$
- $S_{tsvq}(k,R) = S^{*}(k,R) - 10 \log_{10} L$, $R = 3$

k	1	2	4	6
L dB	6.9	3.0	2.53	1.81

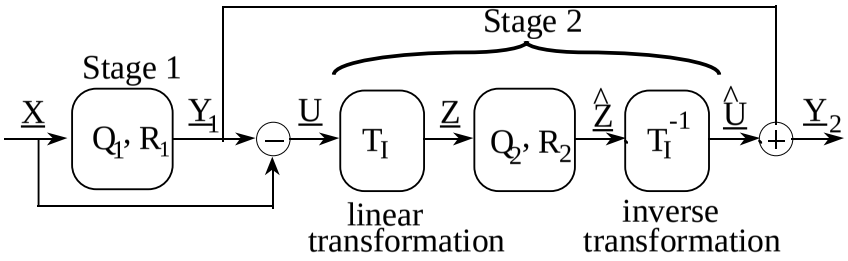


VARIATIONS OF TWO-STAGE VQ

- Can use other types of lossy coders in two stages. (e.g. VQ & scalar Q, transform Q and DPCM)
- Multistage quantization: Two or more stages.
- Cell-Conditioned Two-Stage VQ

CELL-CONDITIONED TWO-STAGE VQ.

With a little extra complexity in the second stage, can make 2VQ work almost as well as single-stage VQ.

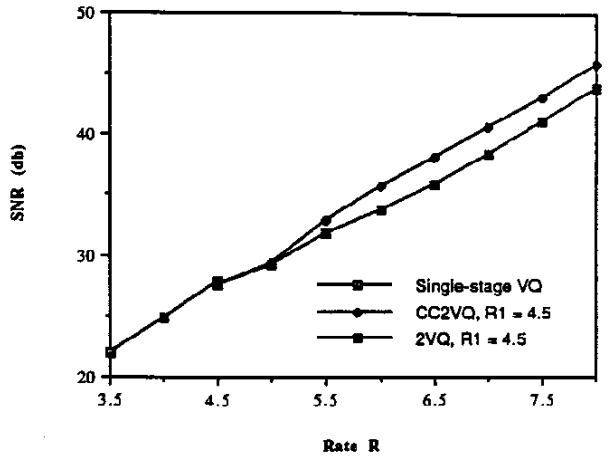


It usually suffices for the transform to be a simple scaling and for the second stage quantizer to have uniform point density.

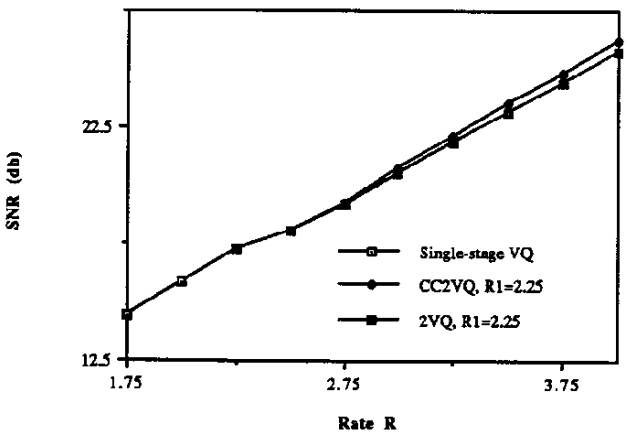
CC2VQ VS. 2VQ

Gaussian AR Source, $\rho = .9$

k = 2



k = 4



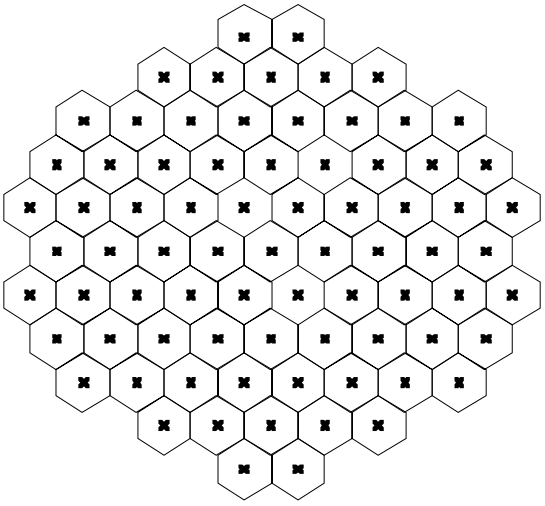
LATTICE (VECTOR) QUANTIZER

- A lattice is an infinite set of points in R^k that is closed under vector addition and substraction.

  Equivalently, a lattice is generated by a basis $\{\underline{u}_1,...,\underline{u}_k\}$ (set of linearly independent vectors). The lattice is

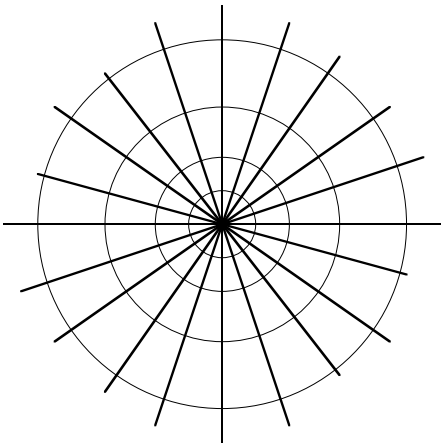
  $$\{\underline{w} = \sum_{i=1}^{k} c_i \underline{u}_i : \ c_1,...,c_k \text{ are integers}\}$$

Hexagonal Lattice in 2 Dimensions

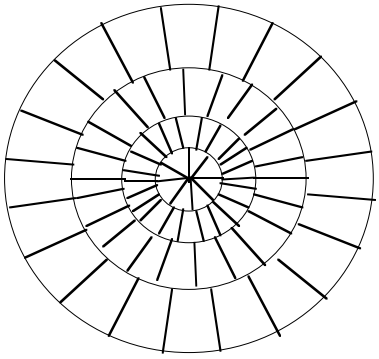


- A "finite" lattice VQ is a "contiguous" subset of the lattice.
- There are low complexity partitioning algorithms for many lattices (linear in kR)
- One also needs a low complexity indexing algorithm. In some special cases, there exists such.
- They can also be used with variable-rate coding. But to my knowledge this has not been well explored.

POLAR QUANTIZATION



- Independently quantize magnitude & angle
- Especially good for circularly symmetric source densities such as IID Gauss



- Unrestricted Polar Quantizers

  Permit the number of quantized angles to vary with the quantized magnitude.