Ch. 10 Vector Quantization

Advantages & Design

Advantages of VQ

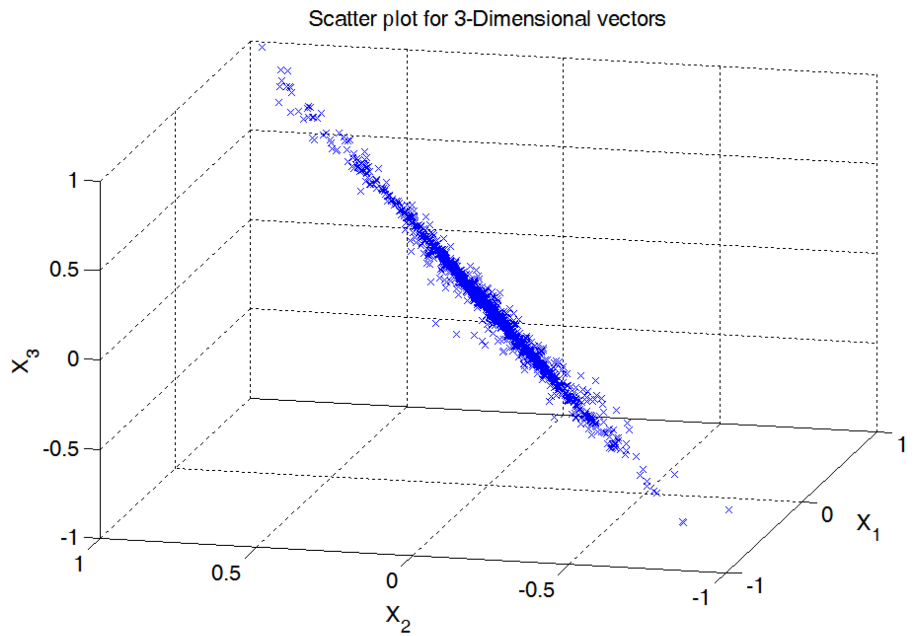
There are (at least) 3 main characteristics of VQ that help it outperform SQ:

1. Exploit ***Correlation*** within vectors
2. Exploit ***Shape Flexibility*** in L-D space
3. Achieve ***Fractional Bits*** per sample

} **Helpful even when source is IID** (items 2 and 3)

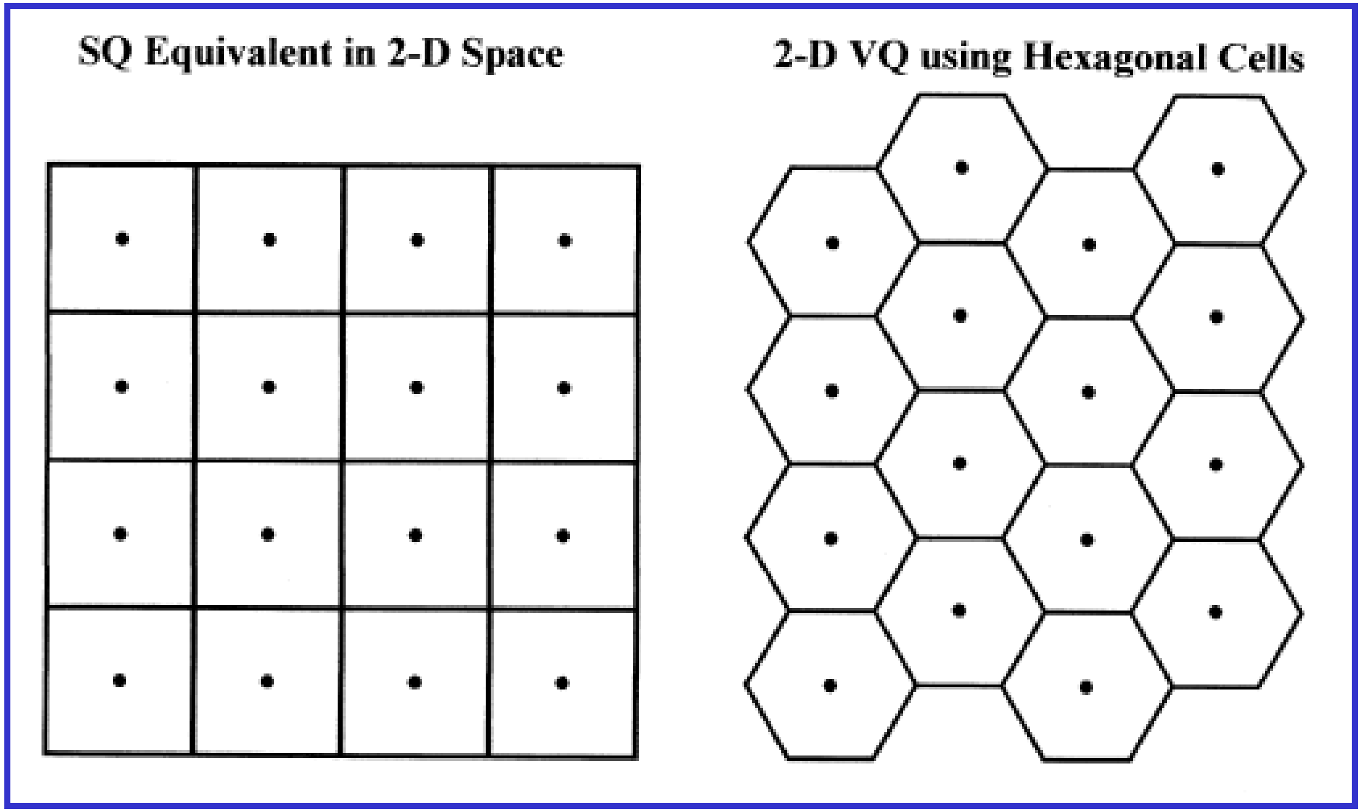
1. Correlation – Correlation leads to regions in L-D space where vectors are very unlikely to occur…

➔ Put big cells where vectors are unlikely
➔ Put small cells were vectors are likely



2. <u>Shape Gain</u> – Uniform SQ equivalent in L-D space gives cells that are L-D hypercubes…

- ➔ VQ cells can be <u>any shape</u>
- ➔ Leads to lower distortion for same rate

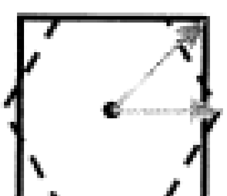


Both of these have the same density of RLs… Both have <u>Same Rate</u>

➔ Hexagonal shaped cells give "shape gain"

Shape Gain helps even for IID!!!

➔ VQ is <u>*ALWAYS*</u> better than SQ



Square cells have larger distortion since maximum distance from center is larger than for hexagon

3. Fractional Bits – The # of bits must always be an integer for SQ

➔ Doesn't allow fine degree of choice of rate

➔ This is especially true at low rates

Example: Say original A/D-sampled signal uses B = 12 bits/sample

Compress further using SQ

If we use 3-bit SQ ➔ CR = 12/3 = 4:1

If we use 2-bit SQ ➔ CR = 12/2 = 6:1

Can't get any CR between these when using SQ

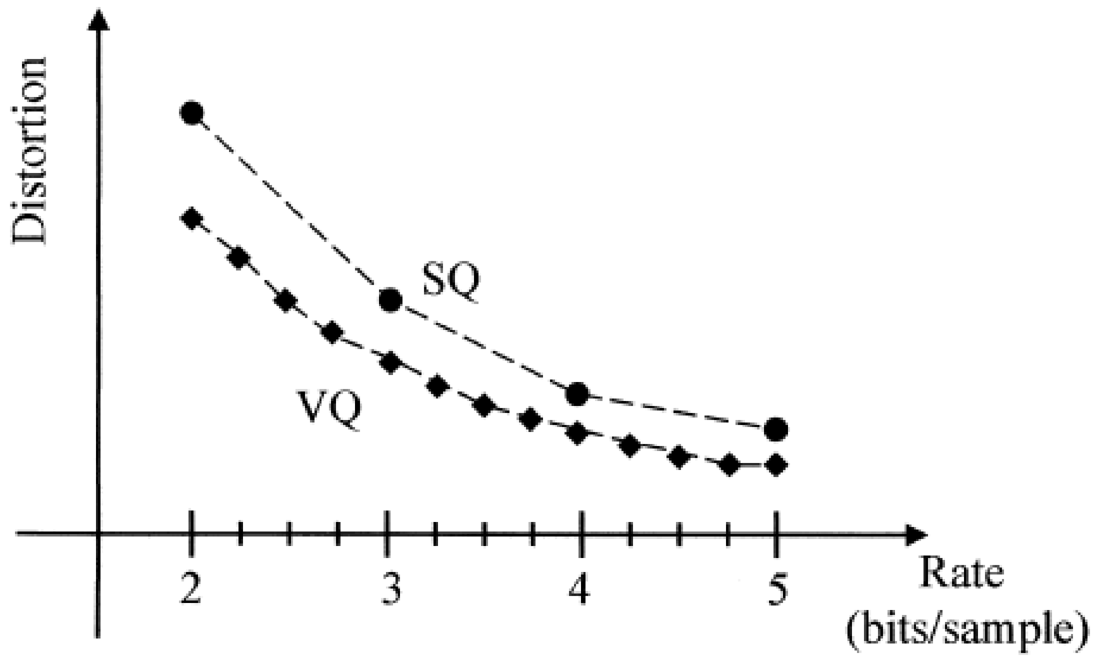
But with VQ we **can**…

Consider L = 4 for the VQ Dimension…

M	bits/vector = $\log_2 M$	bits/sample = $(\log_2 M)/L$	$CR = B/[(\log_2 M)/L]$
256	8	2.00	6.00
512	9	2.25	5.33
1024	10	2.50	4.80
2048	11	2.75	4.36
4096	12	3.00	4.00

Next

Due to the ability to use fractional bits/sample…
VQ gives more flexibility to trade-off R & D



VQ gives more flexibility to trade-off R & D due to fractional bits/sample

Designing an *L*-D VQ

Note: Despite potential confusion… I use the same symbol here to represent the random vector and the dummy-variable vector

In Theory: Given an *L*-order PDF of the signal… i.e., $f_{\mathbf{x}}(\mathbf{x})$

Can design an *L*-D VQ using a vector version of the Lloyd-Max method that was given for SQ

In Practice: Don't know the *L*-order PDF $f_{\mathbf{x}}(\mathbf{x})$!!!

Use a training set of representative signals
This can be viewed in essence as:

Training Set → *L*-Dim Histogram ≈ *L*-Order PDF

But… we will use the training set in a more direct way to design an *L*-D VQ

Linde-Buzo-Gray (LBG) Design

L is VQ Dimension

Recall: VQ Design = Specifying M reconstruction points

Let $x_1[n]$, $x_2[n]$, …. $x_p[n]$ be p signals in the training set.

Break each signal into L-pt. blocks (non-overlapping) too get N training vectors (want N to be large).

→ **Training Vectors**: $\{\mathbf{x}_n\}_{n=1}^{N}$

Steps in Design

1. **Choose Initial Recon. Vectors**: $\{\mathbf{y}_i^{(0)}\}_{i=1}^{M}$

• This choice is important… but for now we'll ignore how we do it

Initialize:

$\{\mathbf{y}_i^{(0)}\}_{i=1}^{M}$ → Initial RLs

$k = 0$ → Iteration Counter

$D^{(-1)} = 0$ → Initial Distortion

$\varepsilon < 1$ → Termination Threshold

Book Error… uses $D^{(0)}$ instead…

Next

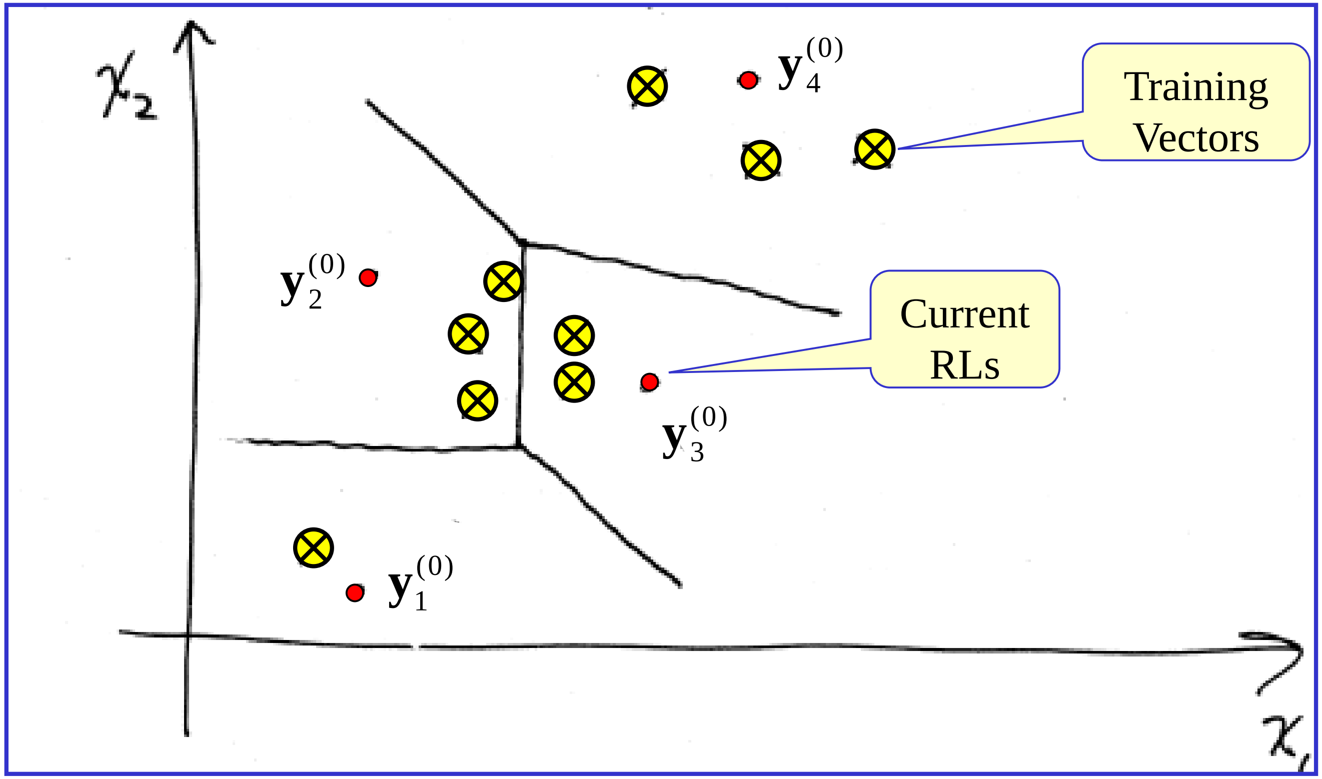
2. **Find Quantization Clusters**: $\left\{V_i^{(k)}\right\}_{i=1}^{M}$

Each $V_i^{(k)}$ is a ***set*** of training vectors clustered around current iteration RL $\mathbf{y}_i^{(k)}$

Definition of Clusters:

$$V_i^{(k)} = \left\{\mathbf{x}_n : d(\mathbf{x}_n, \mathbf{y}_i) < d(\mathbf{x}_n, \mathbf{y}_j), \forall j \neq i\right\}, \quad i = 1, 2, \ldots, M$$

(For ***now***, assume no $V_i^{(k)}$ is empty… deal with the empty cell problem later)



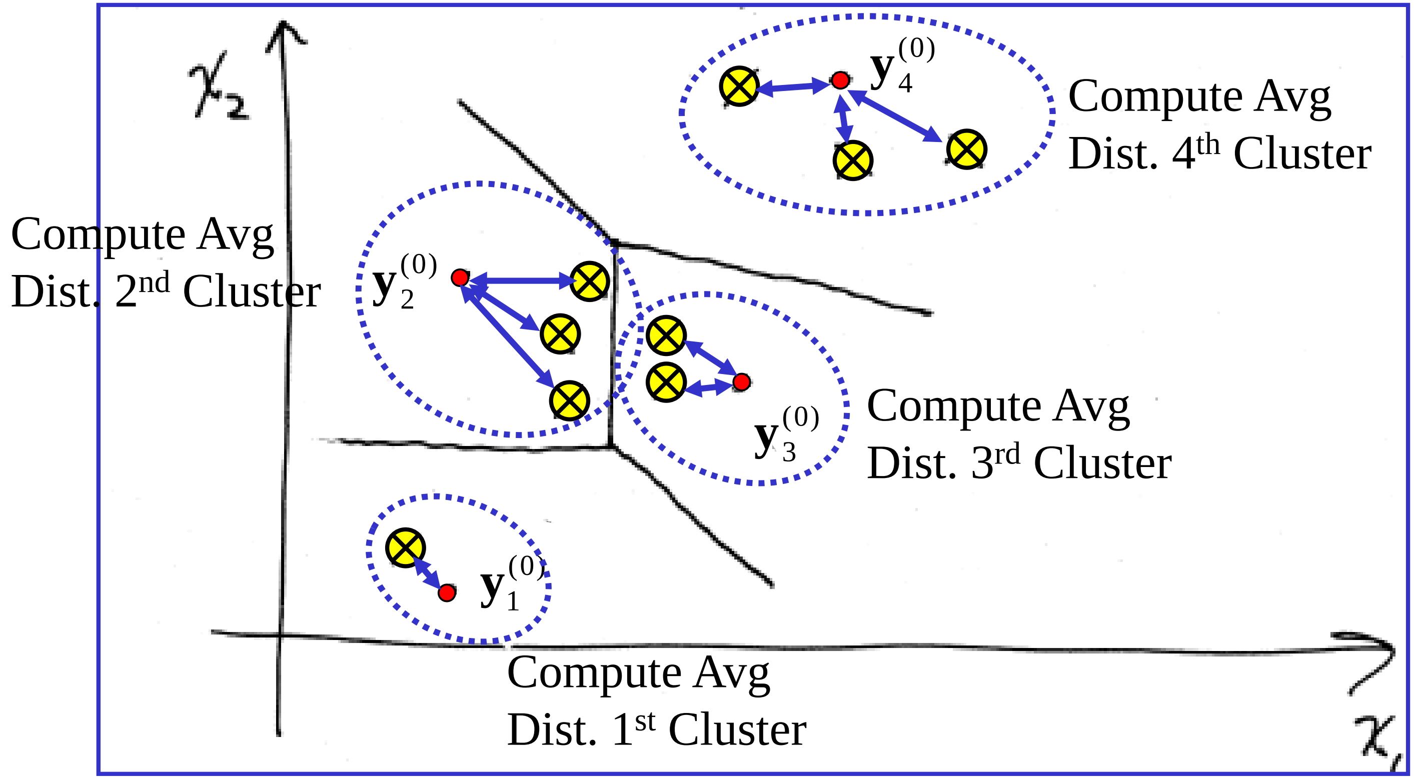
Next

3. <u>Compute Resulting Distortion</u>: ***"Finding Goodness of Current RLs"***

For each cluster let $N_i^{(k)}$ = # elements in $V_i^{(k)}$ and compute:

$$D^{(k)} = \frac{1}{NM}\sum_{i=1}^{M}\left[\sum_{\mathbf{x}_n \in V_i^{(k)}} \left\| \mathbf{x}_n - \mathbf{y}_i^{(k)} \right\|^2\right]$$

N vectors in *M* dimensions ➔ *NM* samples



Next

4. **Check for Convergence**: ***Stop if Change in Distortion is Small***

Stop if: $$\frac{\left|D^{(k)} - D^{(k-1)}\right|}{D^{(k)}} < \varepsilon$$

Note that $D^{(-1)} = 0$ and $\varepsilon<1$ ensures the first iteration's test checks if $1< \varepsilon$ and therefore can't stop on 1st iteration

Otherwise, Continue…

Next

5. <u>Update RLs</u>: *Find Better Reconstruction Vectors*

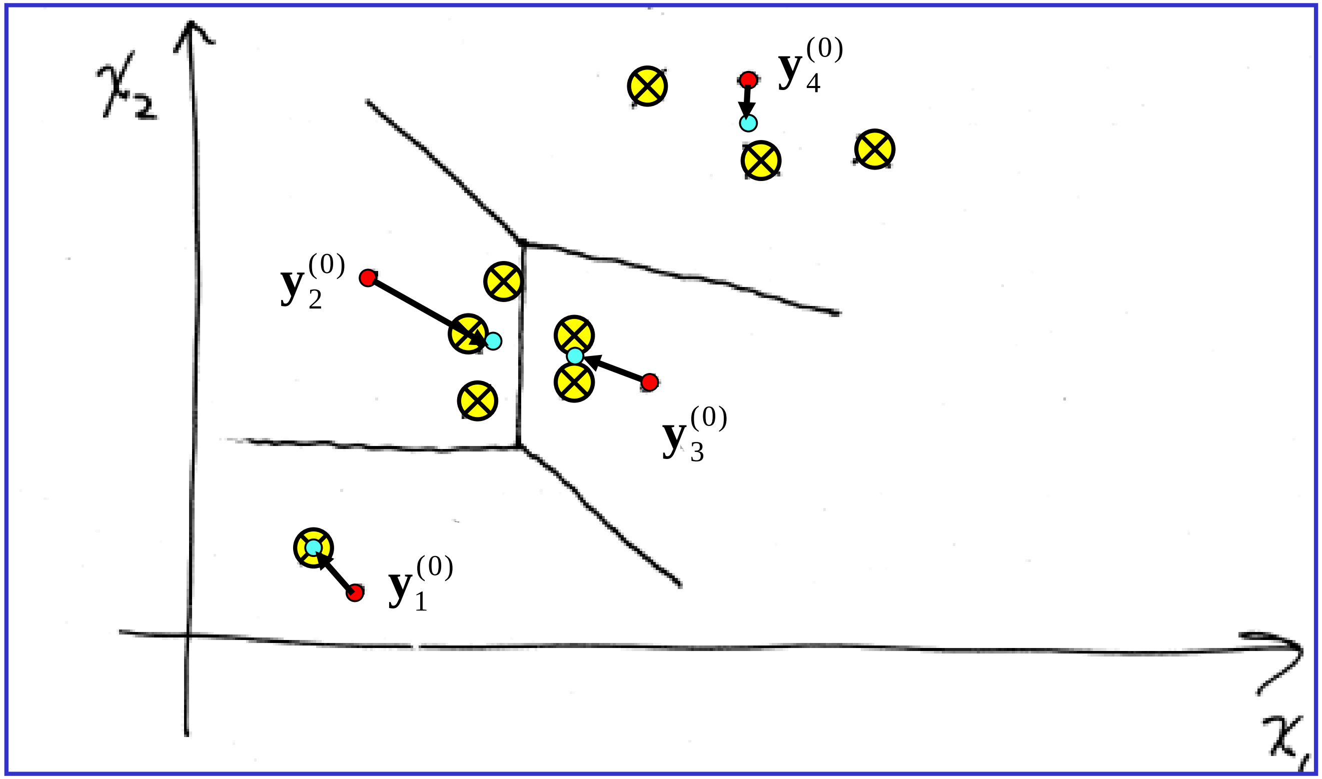
- Set $k = k + 1$
- For each $i = \{1, 2, \ldots, M\}$ compute

$$\mathbf{y}_i^{(k)} = Avg\left\{\mathbf{x}_n \in V_i^{(k-1)}\right\}$$

$$= \frac{1}{N_i^{(k)}} \sum_{\mathbf{x}_n \in V_i^{(k)}} \mathbf{x}_n$$

New RLs are <u>Centroids</u> of Clusters

<u>Data</u> Averaging



Go To Step #2

Next

Problems With LBG Design

- **LBG Finds Local Minimum**
  - Although LBG is guaranteed to <u>not increase</u> the distortion at each iteration…
    - …it may not yield the optimal design
- Empty Cell Problem
  - If in some iteration there is an RL that has an empty cluster…
    - You can't find an a new RL (recall: New RL = Centroid of Cluster)
- Different Initializations Give Different Designs w/ Different Performance
  - This is linked to this problem…

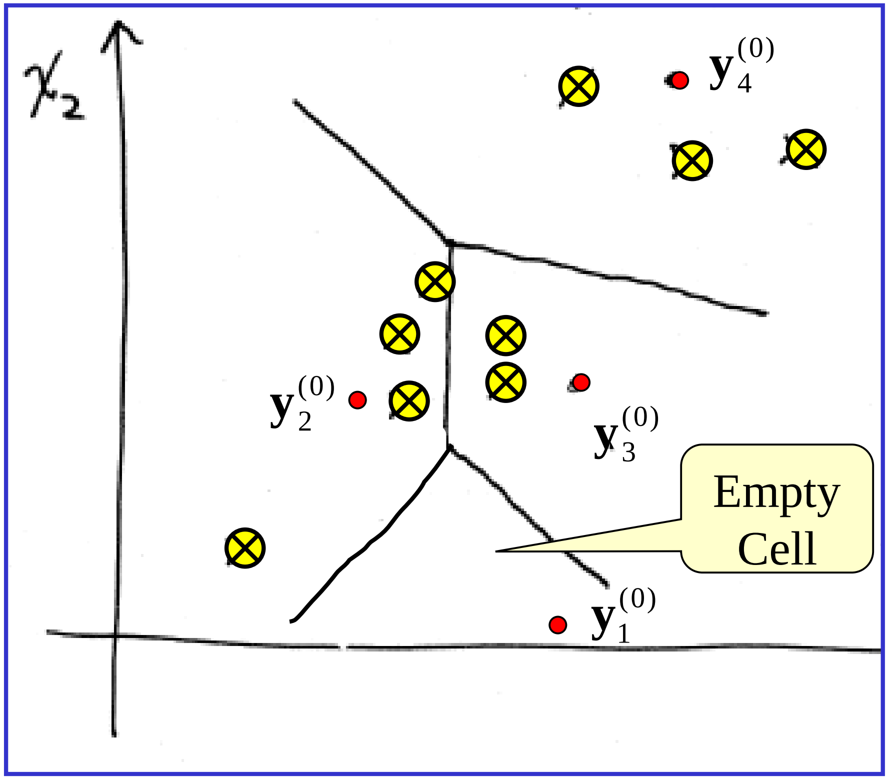
Empty Cell Problem

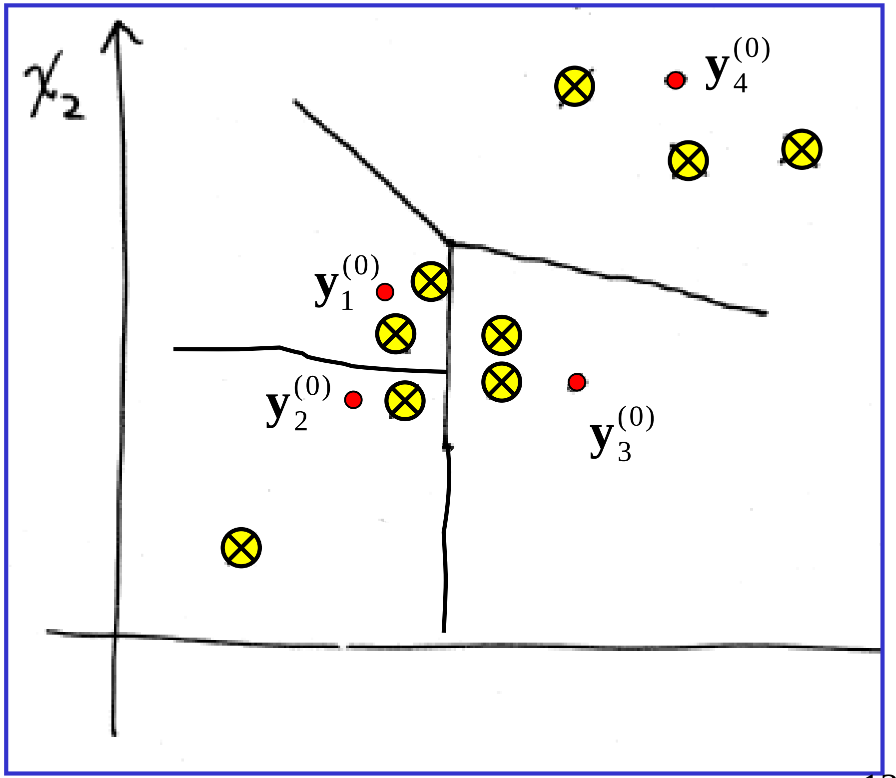
If in Step #2 (*Find Clusters*) some RL $\mathbf{y}_i^{(k)}$ gets no cluster, then $V_i^{(k)}$ is empty

Then, in Step #5 (*Update RLs*) you can't find a new RL…

➔ Loss of an RL reduces the "Designed-to" Rate of the VQ… Not Desirable

Solution: In Step #2…Replace the empty cell's $\mathbf{y}_i^{(k)}$ with a new RL randomly placed inside the *most populous* cell… then, re-cluster and proceed.





Next

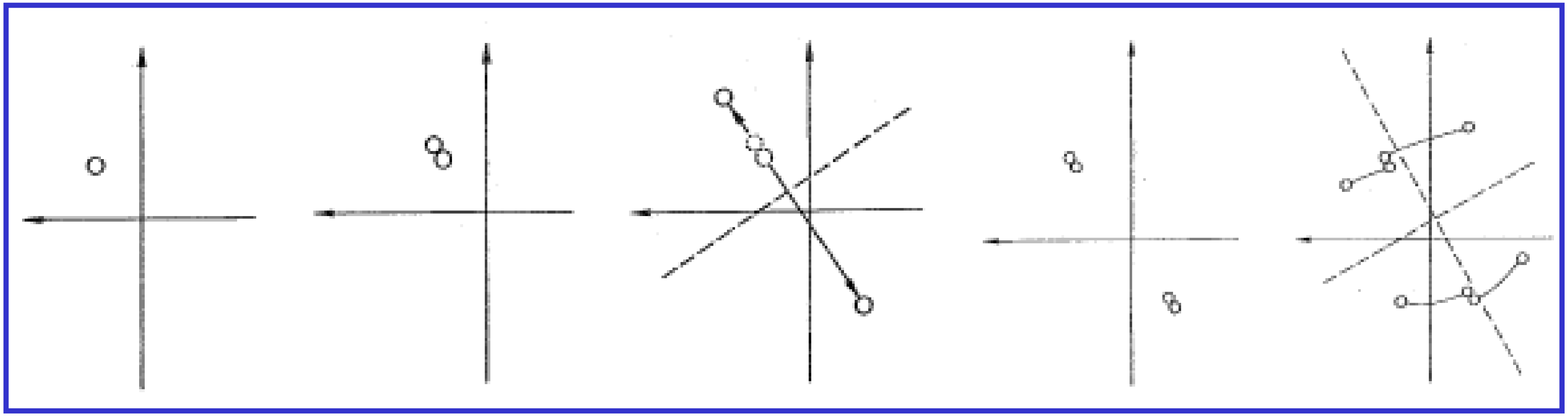
Initialization

Different initial RLs give different designs… w/ different perform.

Four Alternative Approaches

1. Splitting
   - Start w/ $M = 1$ (i.e., a 1-level VQ) and use LBG to design the VQ
   - Randomly split the level (small perturbation) to get 2 RLs
     - Now, $M = 2$ … Run LBG to design $M = 2$ VQ
   - Split again and again to get $M = 4, 8, 16, 32$, etc…

$M = 1$ | Perturbed, $M = 2$ | LBG'd, $M = 2$ | Perturbed, $M = 4$ | LBG'd, $M = 4$



From Gray's Paper on VQ in *IEEE ASSP Magazine*, April 1984

2. Randomly Pick *M* RLs from Training Vectors
   - Randomly Pick *M* RLs
   - Run LBG to get M-Level Design
   - Repeat these two steps several times & pick the best resulting design

3. Use all Training Vect. as RLs and Combine to get Desired *M*
   - Pair-wise Combine "Nearest Neighbors"
     - Paired to give smallest increase in Dist.
   - Combine each pair into a single new RL = Avg of two in pair
   - Repeat until # of RLs has decreased to Desired *M*
   - Use resulting *M* RLs as initial RLs for the LBG design

4. Design decent SQ for the prob. & form *L*-D "Rect"-Grid VQ
   - Use resulting rectangles' centroids as initial RLs for LBG

End