

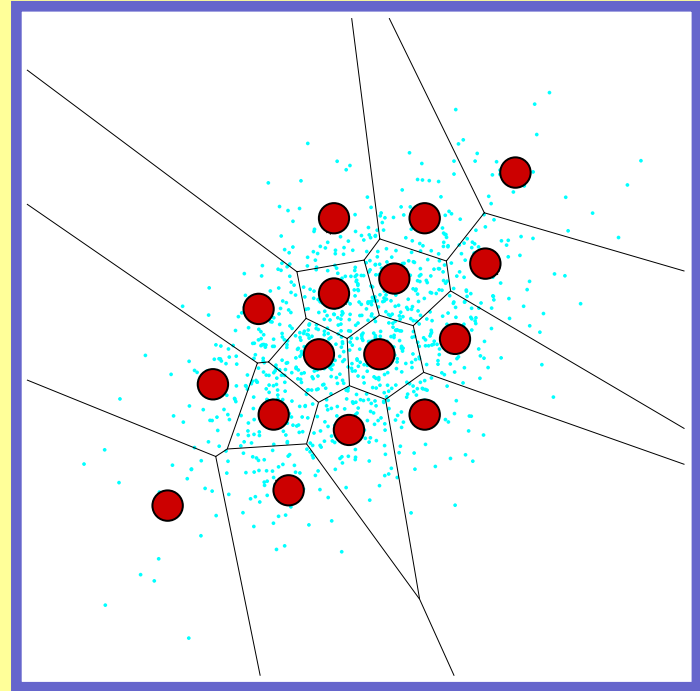
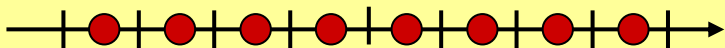
Self-organizing Maps for Index Assignment in Vector Quantization: an Application to MDC

Giovanni Poggi
Università “Federico II” di Napoli

Agenda

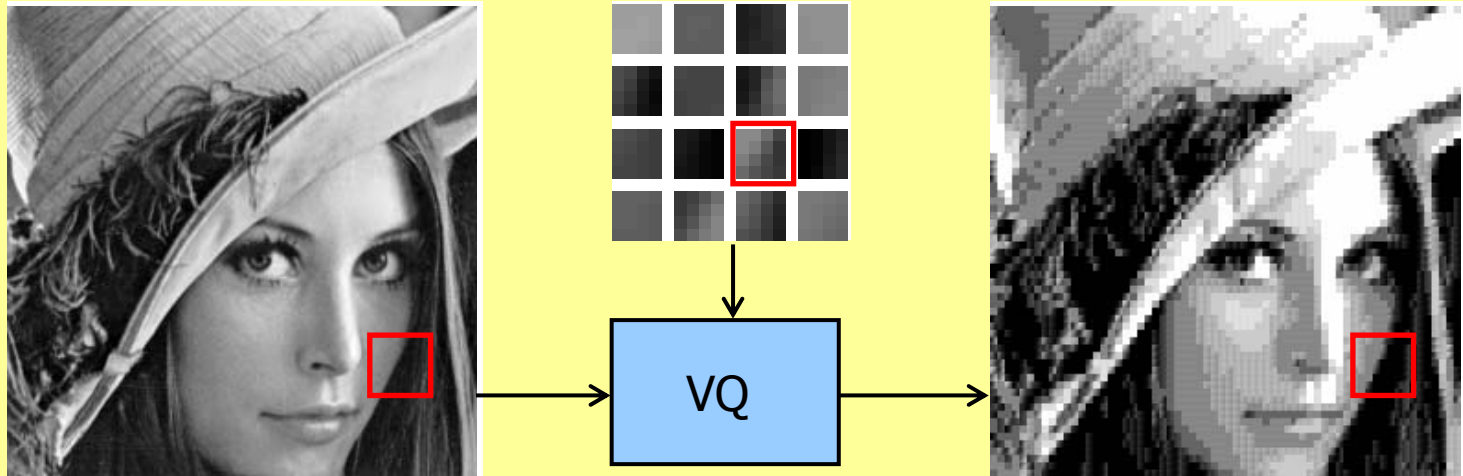
- **Basics on VQ**
- VQ codebook ordering: why? And how?
- Self-organizing Maps (SOM)
- Use of SOM in problems involving codebook ordering
- Multiple-description VQ based on the SOM

Scalar and Vector quantization

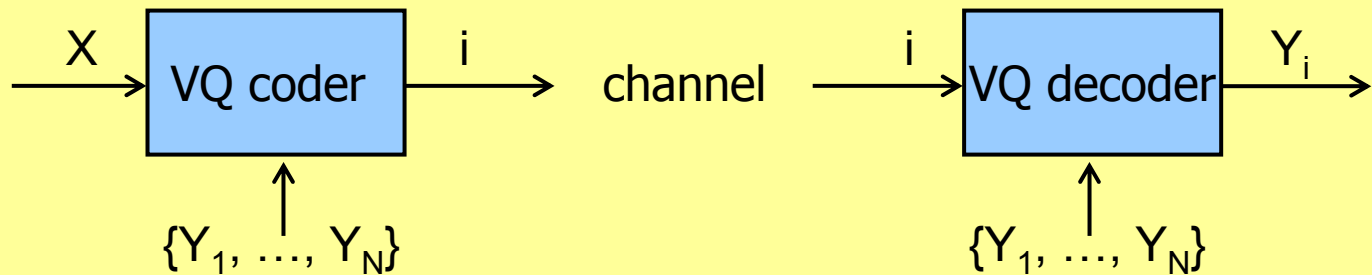


- Each input vector is approximated by the nearest codeword (also a classification problem)

VQ, an image coding example



- ...or more precisely



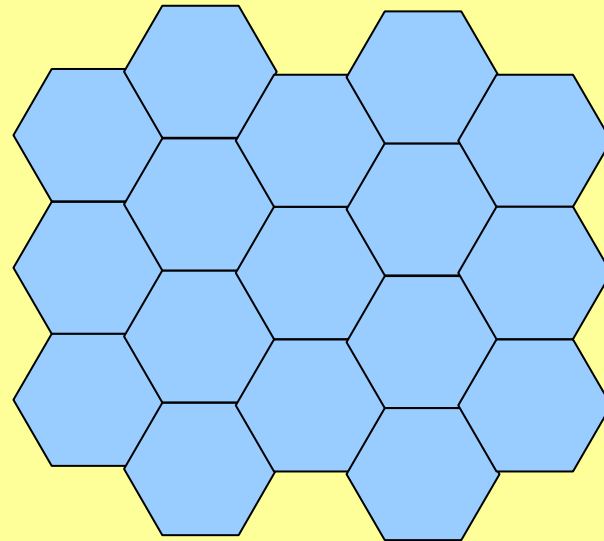
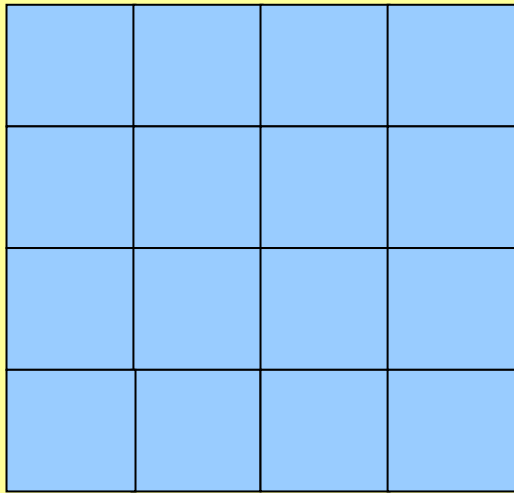
Why a talk on VQ?

- Foundations in Information Theory
- Research takes off with the Linde-Buzo-Gray paper (1980) but Interest vanishes quickly after first results on wavelets, e.g. Antonini et al. (1992)
- It was never very effective for image compression (better results on audio)
- Quite complex (both design and operation)

Quantization *is* compression

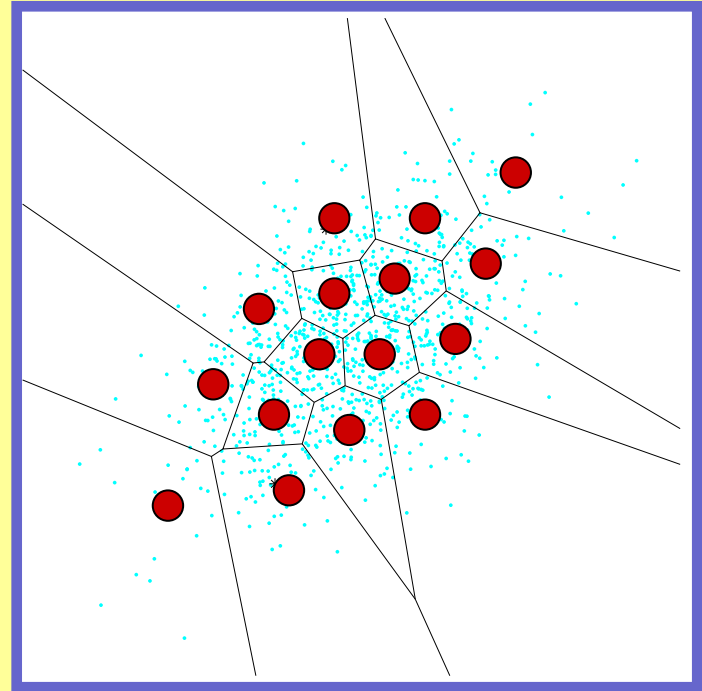
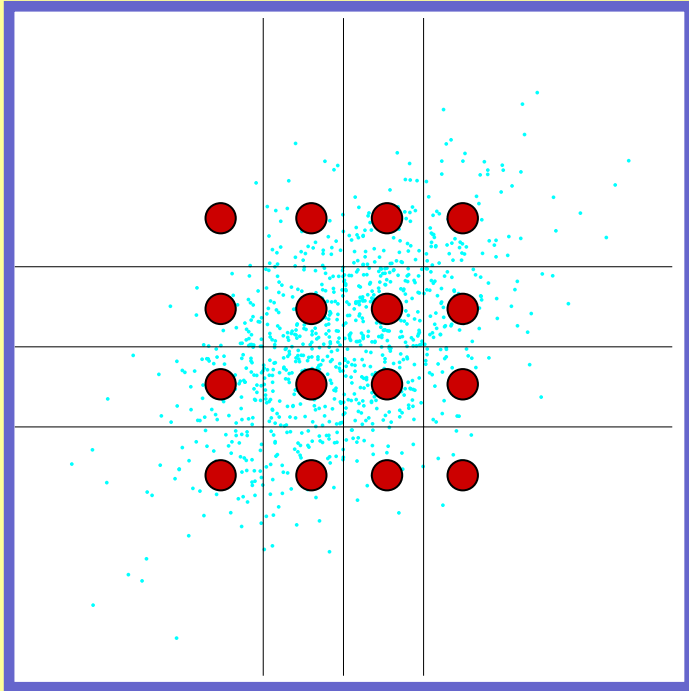
- There is **no compression without quantization**
- It is better to quantize vectors rather than scalars
- Any compression technique can be regarded as a (suboptimal) form of vector quantization
- VQ should be regarded as a *tool* to be used with a grain of salt

The cell-shape advantage



- For uniform pdf in $2d$, at high rate, the gain is 0.17 dB
- For gaussian pdf, $d \gg 1$, 1 b/s, the gain is 1.62 dB

The memory advantage

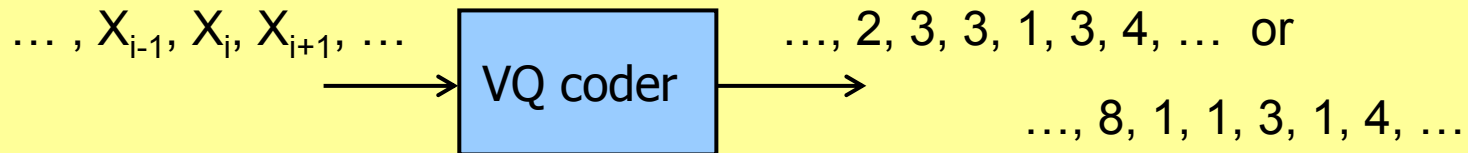
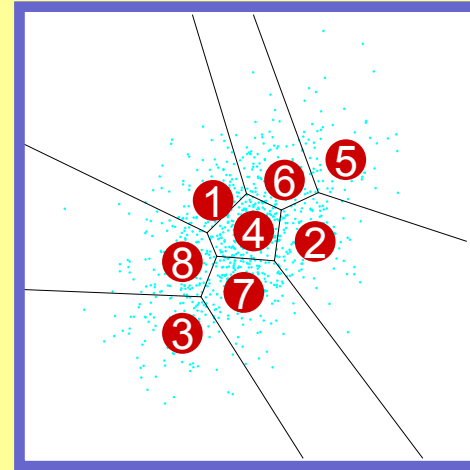
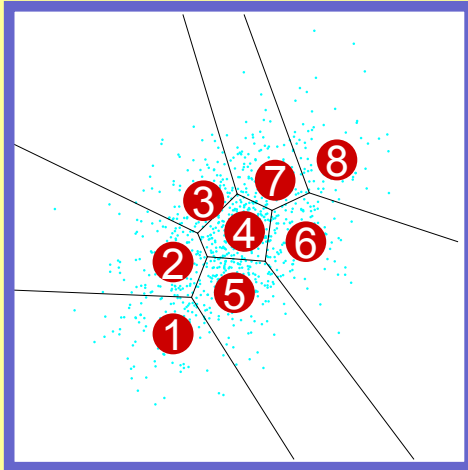


Codewords can **follow the input distribution**

Agenda

- Basics on VQ
- **VQ codebook ordering: what? why? And how?**
- Self-organizing Maps (SOM)
- Use of SOM in problems involving codebook ordering
- Multiple-description VQ based on the SOM

Codeword indexing



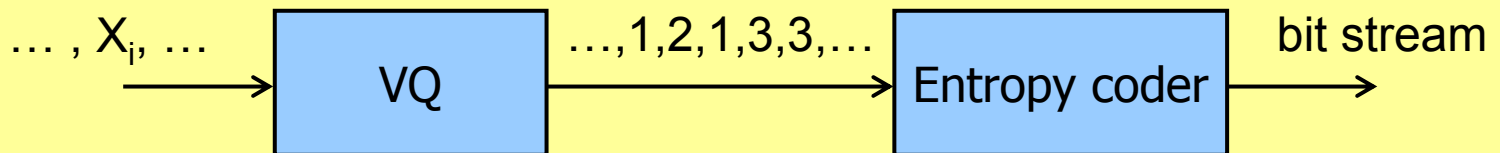
A **degree of freedom**: does it make any difference?

Codeword indexing: what for?

- Compression of VQ indexes
- Zero-redundancy error protection
- Progressive transmission
- **Multiple description coding**

Compression of VQ indexes

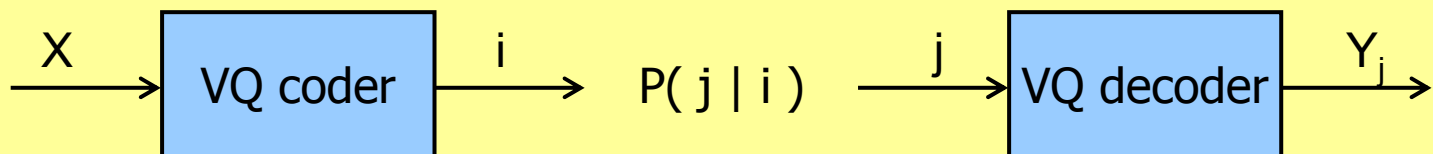
- To limit complexity, only small blocks are used in VQ
 - only short-range dependencies are exploited
- But VQ indexes are not independent
 - they can be further compressed by entropy coding



Statistics of indexes do matter

Zero-redundancy error protection

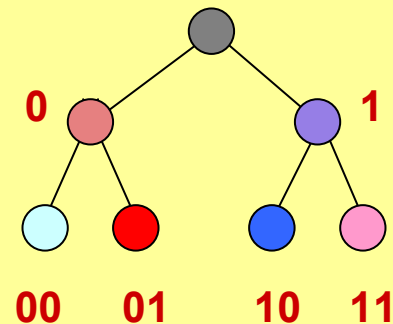
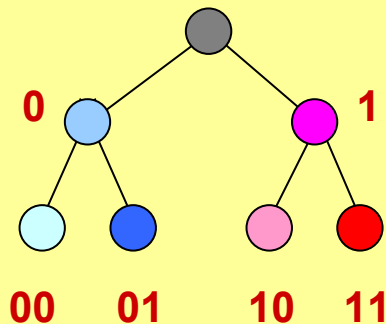
- Channel errors increase distortion...
- ...from $\|X - Y_i\|^2$ to $\|X - Y_j\|^2 \approx \|X - Y_i\|^2 + \|Y_i - Y_j\|^2$
- But the $P(j | i)$ are not all equal:
 - e.g., given “000”, “001” is more likely than “111”



Index assignment can reduce the effects of errors

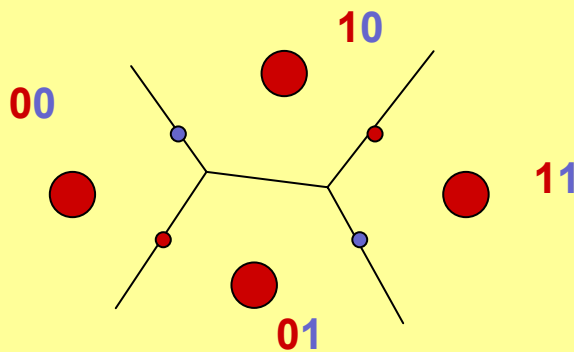
Progressive transmission

- Transmission of indexes by bit-planes
- e.g.: bit “0” is used to address a codeword Y_0
- ...which should be a good approximation of all codewords of the type Y_{0x}
- **Codewords with “close” indexes should be similar**

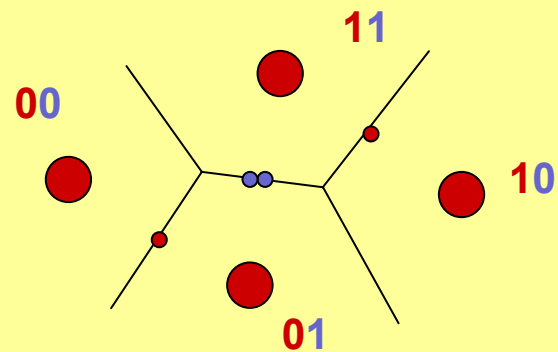


Multiple-description VQ

- Each VQ index is split in two (or more parts) and sent over different channels
- Each part should be useful by itself to cope with losses
- **Good index assignment is the key problem**



GOOD



BAD: 2^o bit useless

How to obtain an ordered codebook?

- Design a generic codebook and change indexing afterwards
 - Optimal ordering has huge complexity
 - Suboptimal techniques, different for each problem
- Design an ordered codebook
 - Define in advance the desired codebook structure
 - Run the Self-organizing Maps (SOM) algorithm

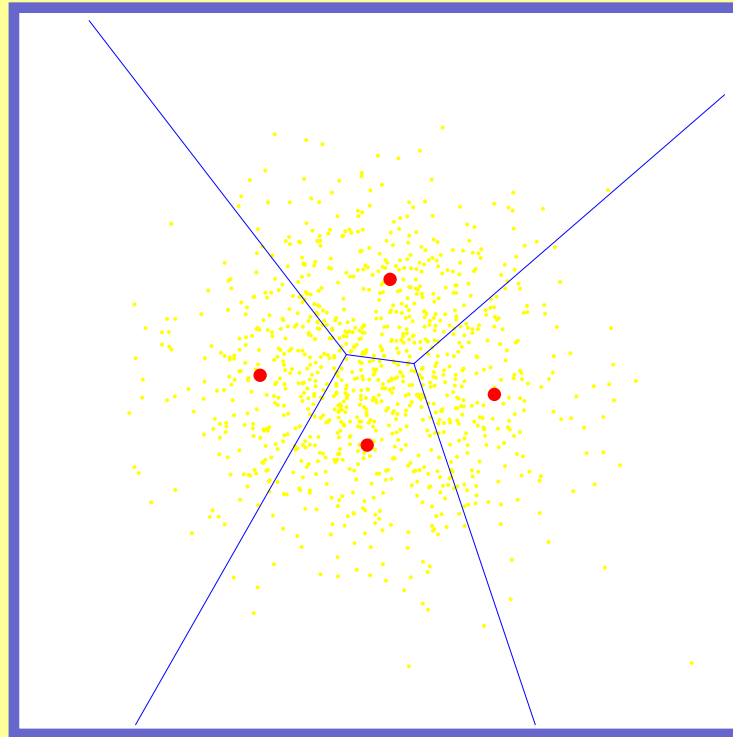
Agenda

- Basics on VQ
- VQ codebook ordering: why? And how?
- **Self-organizing Maps (SOM)**
- Use of SOM in problems involving codebook ordering
- Multiple-description VQ based on the SOM

Codebook design for low-rate VQ

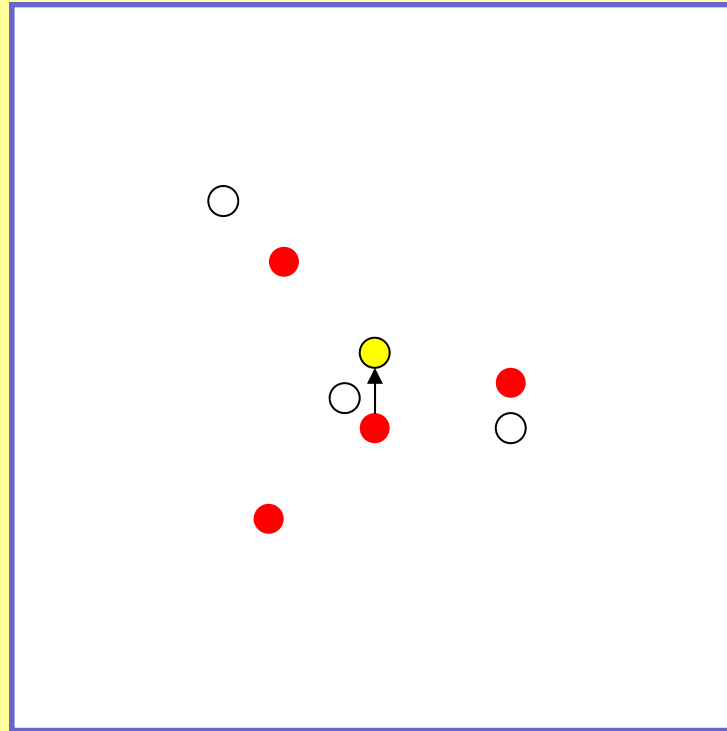
- Case of interest for most compression applications (typically we do have few bits!)
- Main problem: codebook design
 - Generalized Lloyd algorithm (or LBG)
 - Tree-structured (and other structured) codebooks
 - Pairwise nearest-neighbors
 - Annealing
 - **Self-organizing maps** (a.k.a. Learning VQ)
 - Etc. etc.

Generalized Lloyd Algorithm



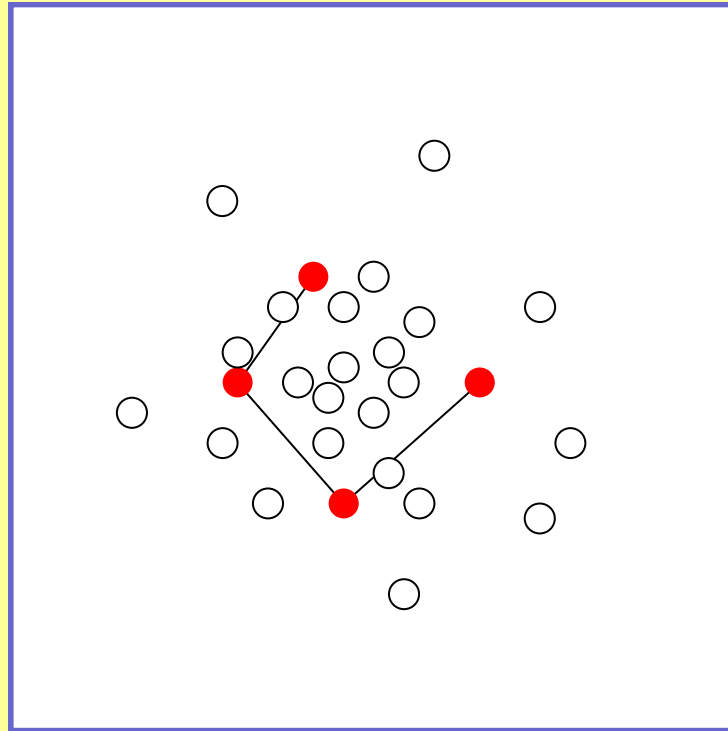
Alternate optimization of coder and decoder

K-means



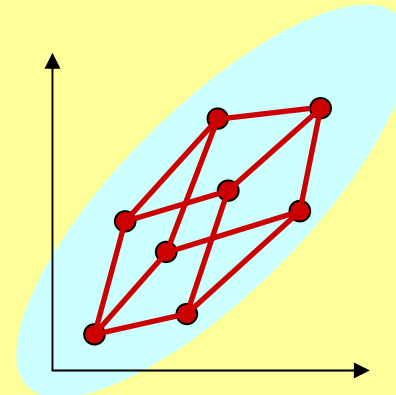
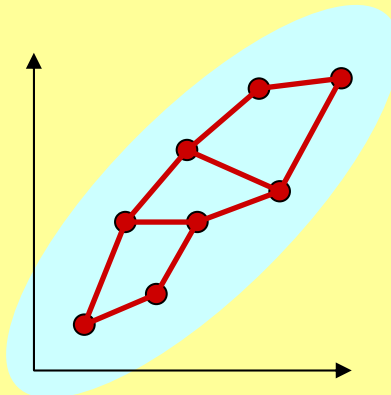
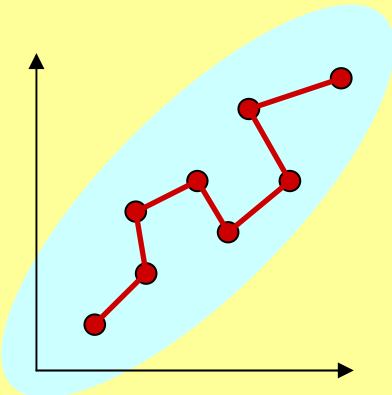
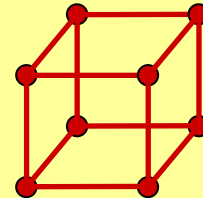
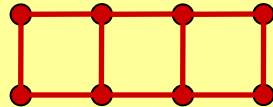
Like GLA but adaptation is at each training vector

Self-organizing Maps



Like K-means but all codevectors adapt at each step

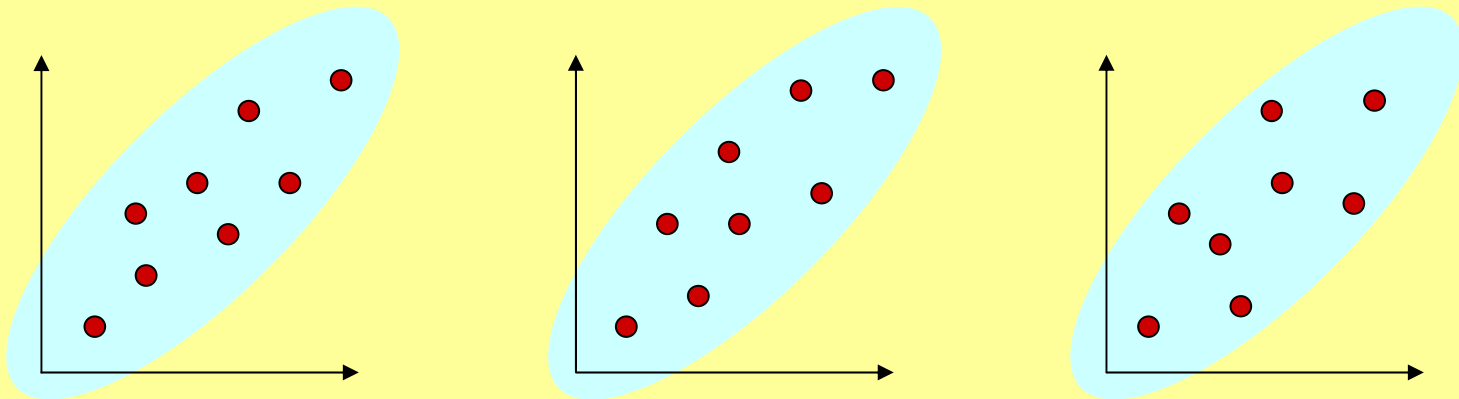
Same source, different structures



- Structure is defined through index distance
- e.g. $d(i,j)=|i-j|$ (case 1) $d(i,j)=h(i,j)$ (case 3)

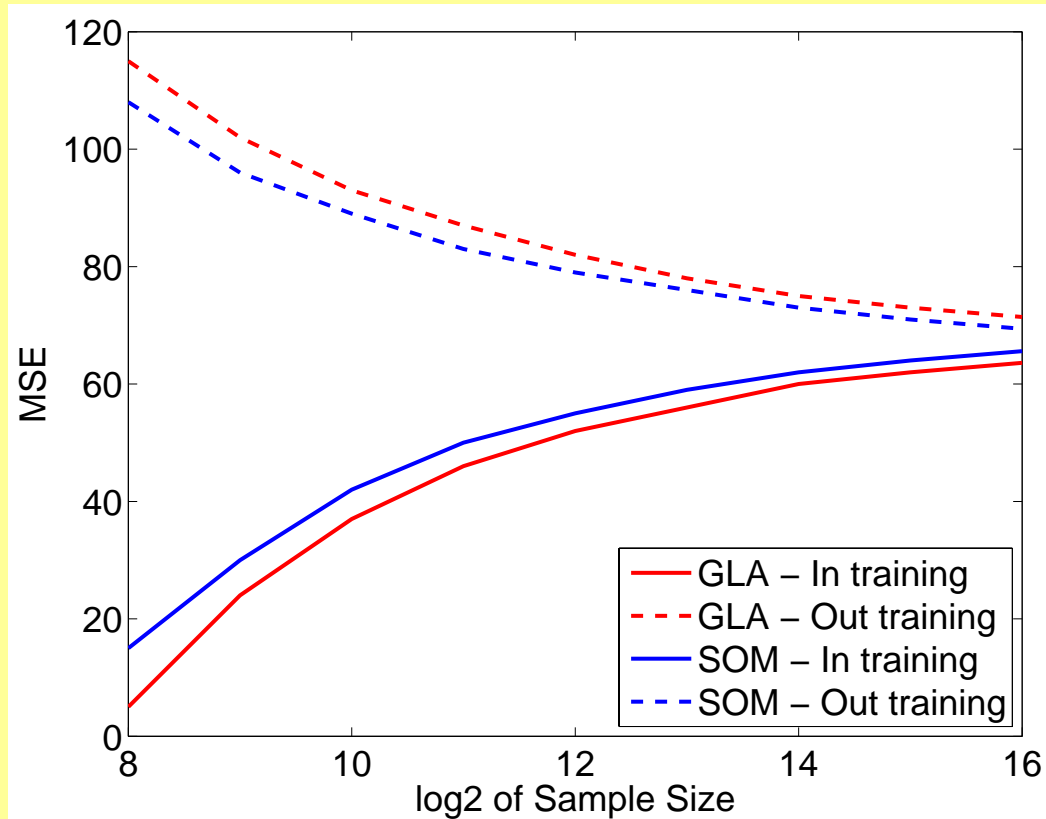
Same source, different structures

- The structure is not part of the codebook



- The codebooks cover about equally well the input pdf

SOM is not optimal...



... but better than GLA **out of training**

Pros / Cons of SOM

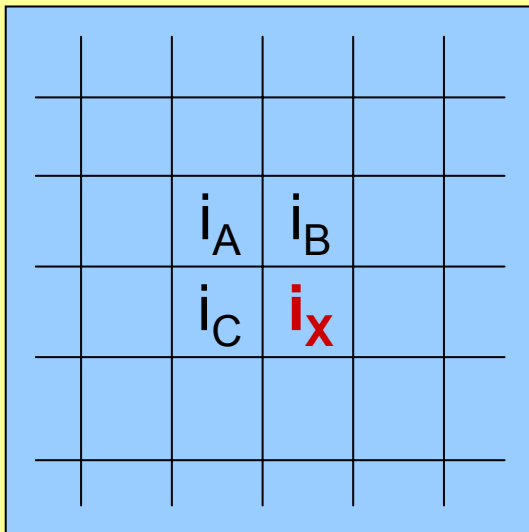
- ▼ No optimality
- ▼ Complexity (higher than GLA, no fast algorithms)
- ▼ Needs some parameter setting
- ▲ Good quality
- ▲ Robustness (works with smaller training sets)
- ▲ **Allows simple and flexible codebook ordering**

Agenda

- Basics on VQ
- VQ codebook ordering: why? And how?
- Self-organizing Maps (SOM)
- **Use of SOM in problems involving codebook ordering**
- Multiple-description VQ based on the SOM

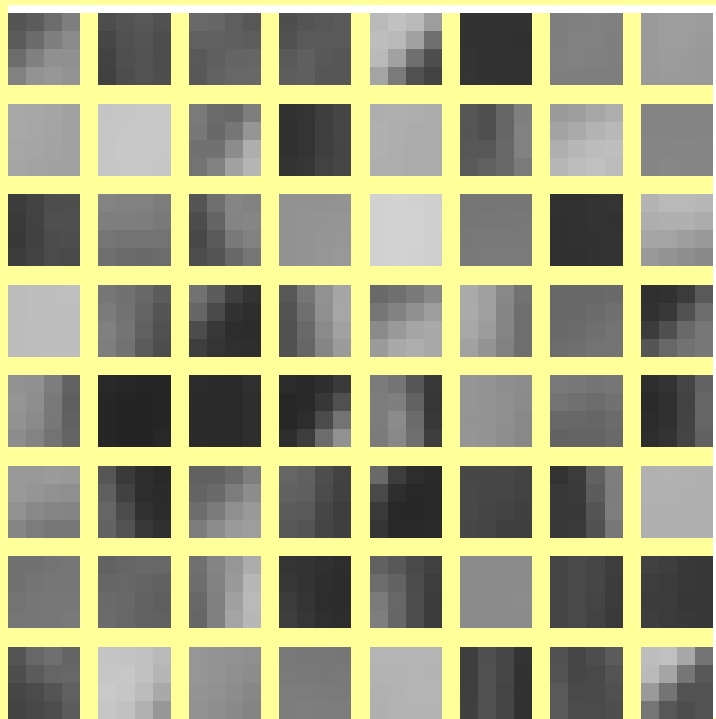
Index compression: Address VQ

- Exploits dependencies among indexes by prediction
- Entropy coding based on $\Pr(\mathbf{i}_x \mid i_A, i_B, i_C, \dots)$
- Huge complexity since dependency is **non-linear**

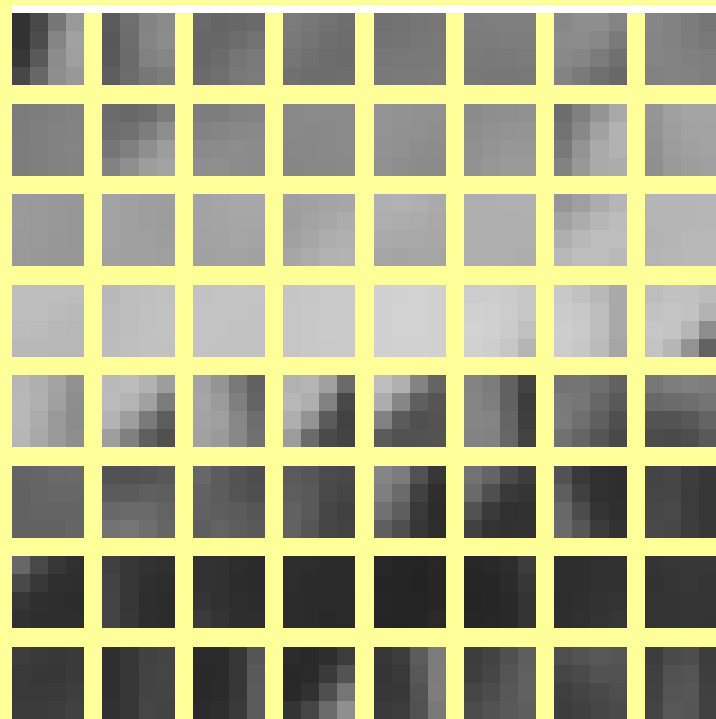


- But **linear** dependency can emerge by suitable codeword indexing

VQ codebook with linear structure



GLA



SOM



Address VQ with linear prediction



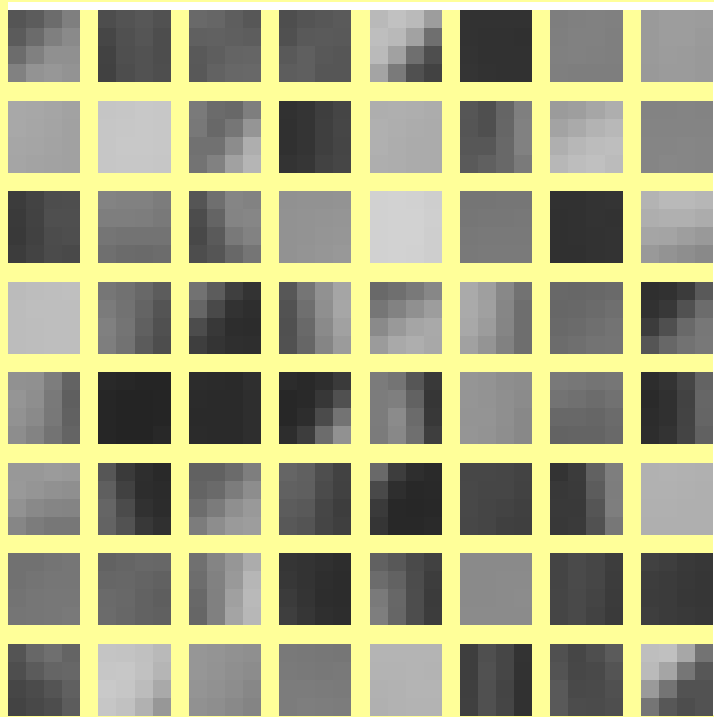
84	62	16	17
84	59	16	18
84	59	15	16
86	53	15	15

- Ordering allows simpler coding (even lossy)
- Saving of 40% to 60% in bit-rate

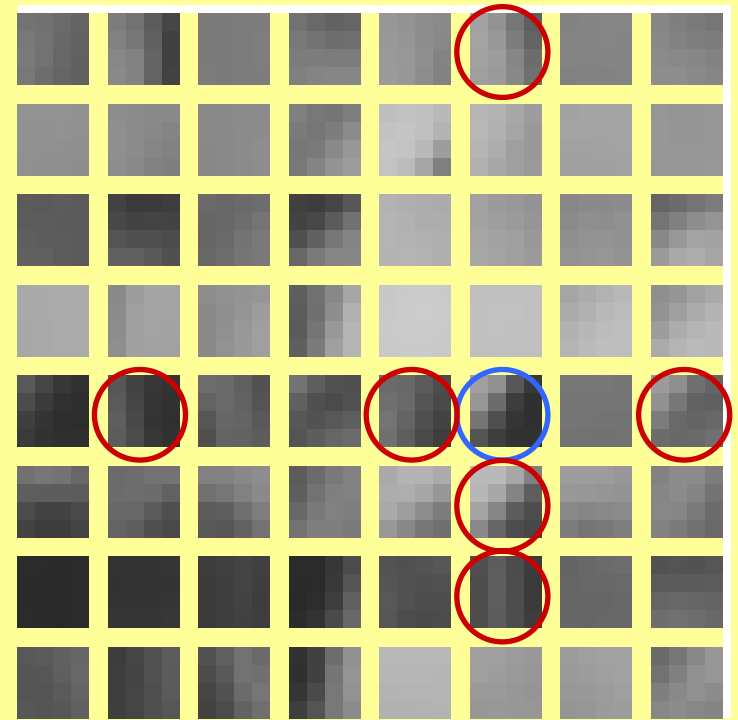
Zero-redundancy error protection

- Codebook ordering by brute-force is too complex
- Binary switching algorithm
 - Compute the contribution of each cw to channel distortion
 - Try switching the worst cw with all others
 - Accept the best switching
 - Stop when no switch reduces distortion anymore
- ▲ Guarantees a local optimum,
- ▼ **Still very complex**, works only for small BERs,

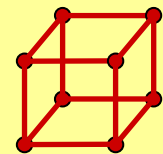
VQ codebook with hypercubic structure



GLA



SOM



Example results: BER $2e-2$



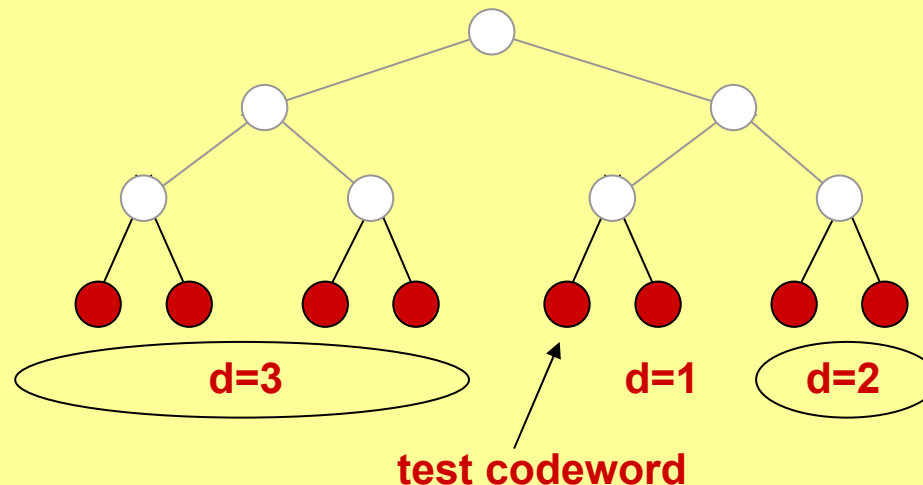
GLA: SNR=20.5 dB



SOM: SNR=21.4 dB

Progressive Transmission

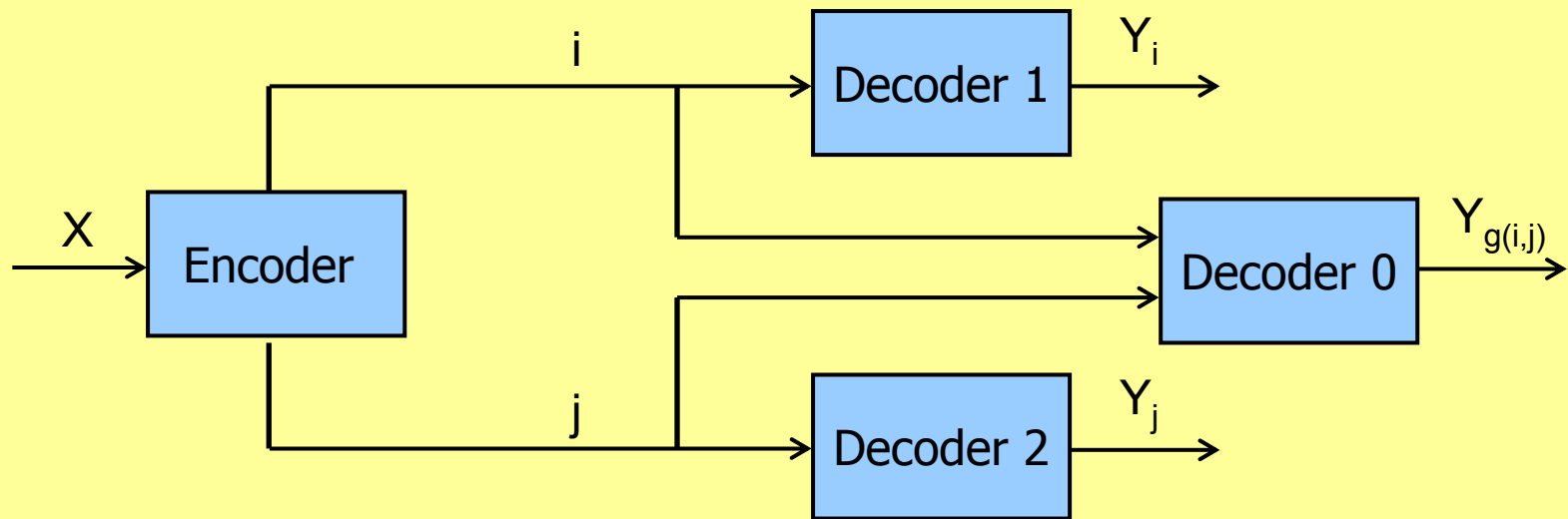
- Tree-structured VQ is a well-known solution but
- ... it uses a **constrained** codebook, hence suboptimal
- SOM with a tree-structured index distance provides a better performance



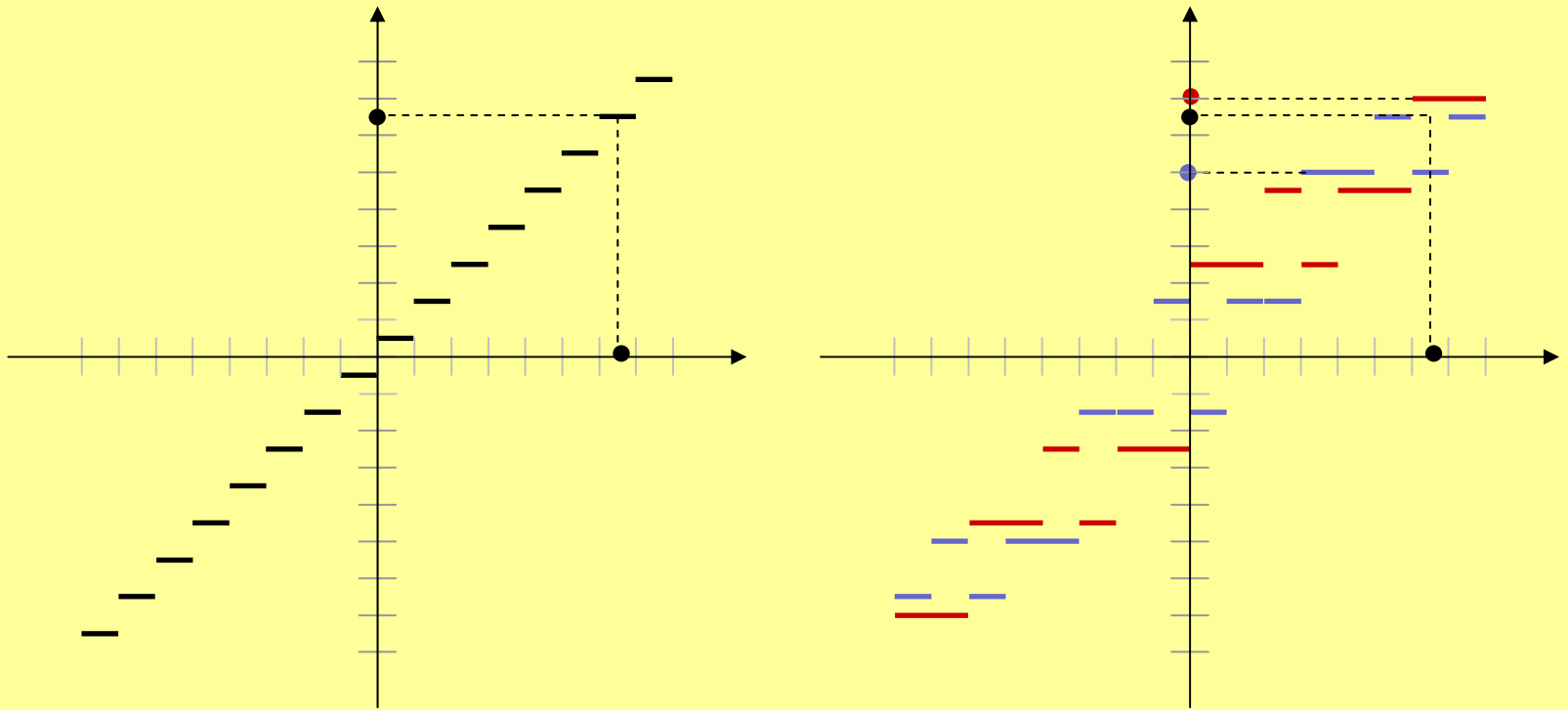
Agenda

- Basics on VQ
- VQ codebook ordering: why? And how?
- Self-organizing Maps (SOM)
- Use of SOM in problems involving codebook ordering
- **Multiple-description VQ based on the SOM**

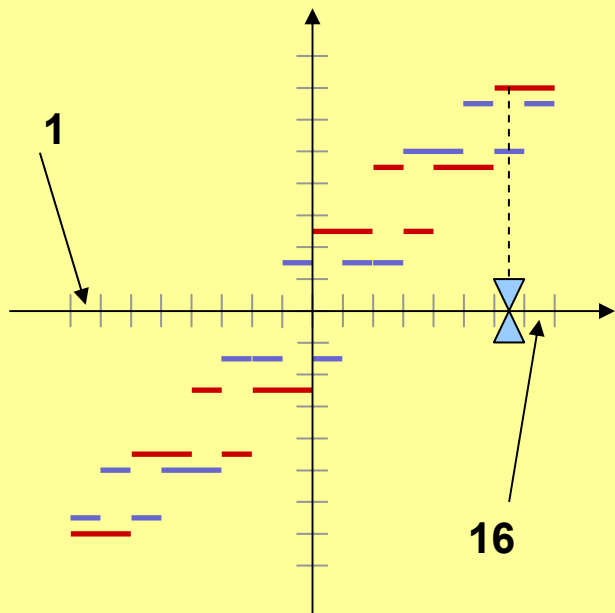
Multiple Description Coding



- When all channels work, high quality is obtained
- ..but each description is useful by itself



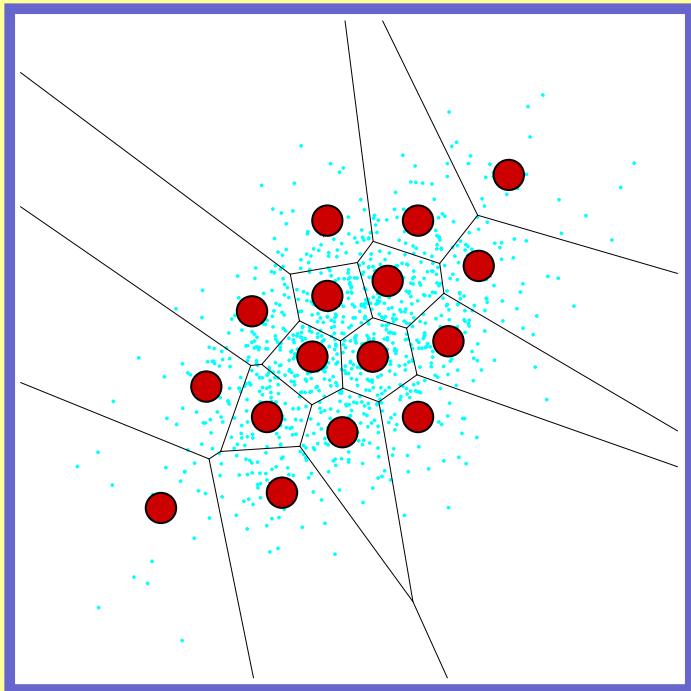
Descriptions is slightly redundant $2 \log 6 \approx 5.16 > 4$



1	2	3	4	5	6
---	---	---	---	---	---

1	1	3				
2	2	4	5			
3		6	7	9		
4			8	10	11	
5				12	13	15
6					14	16

Multiple description VQ



	1	2	3	4	5	6
1	1	3				
2	2	4	5			
3		6	7	9		
4			8	10	11	
5				12	13	15
6					14	16

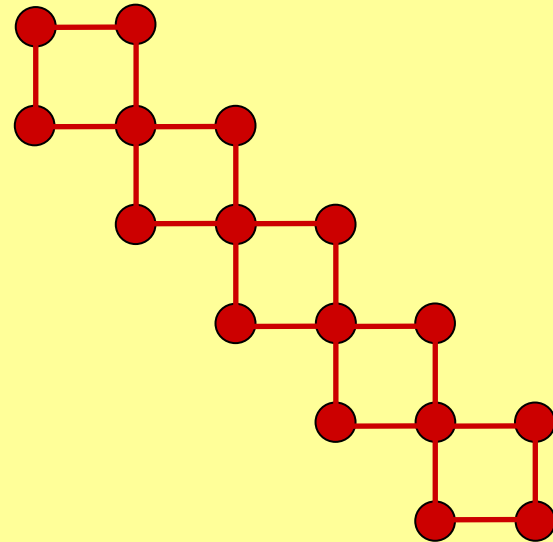
- Maybe the same matrix could work here too but ... which codeword is #1, #2, and so on?

Design of MD VQ codebooks

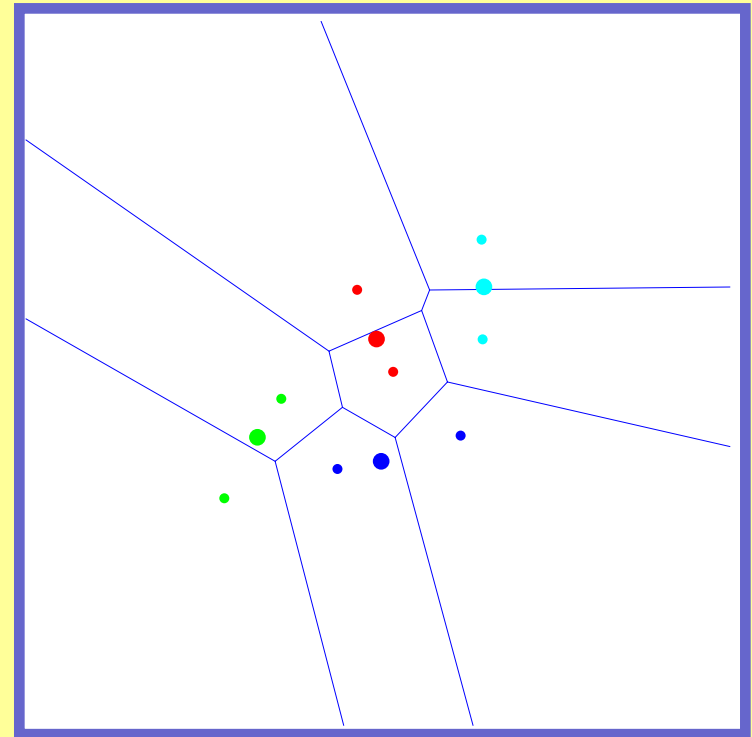
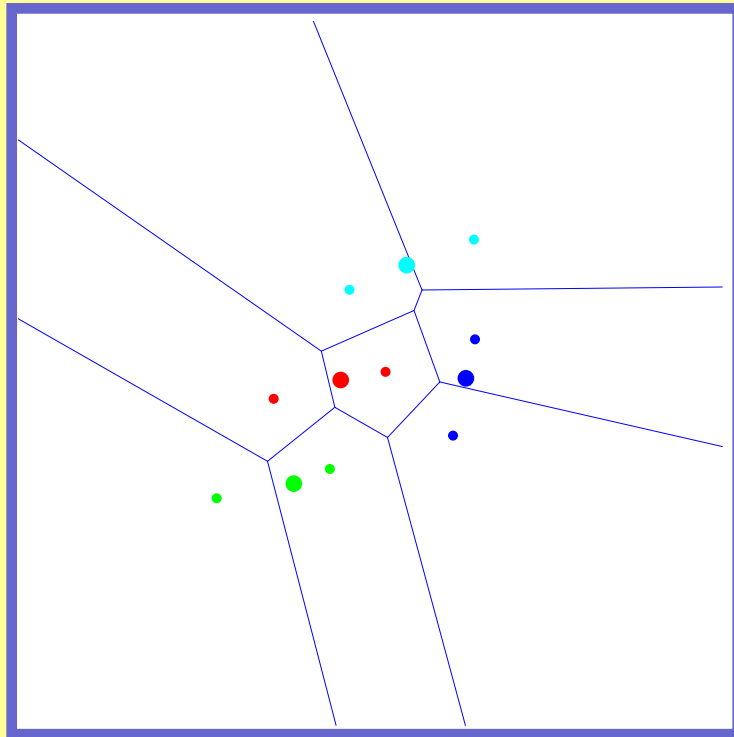
- Variant of BSA
 - Good performance
 - High complexity, acceptable for case of **2 descriptions**
- Tree structured codebook
 - Simple
 - Works only for **non-redundant descriptions**
- Lattice VQ
 - OK when number of codewords is very large and pdf given
- Others...

Let us use SOM: structure is given!

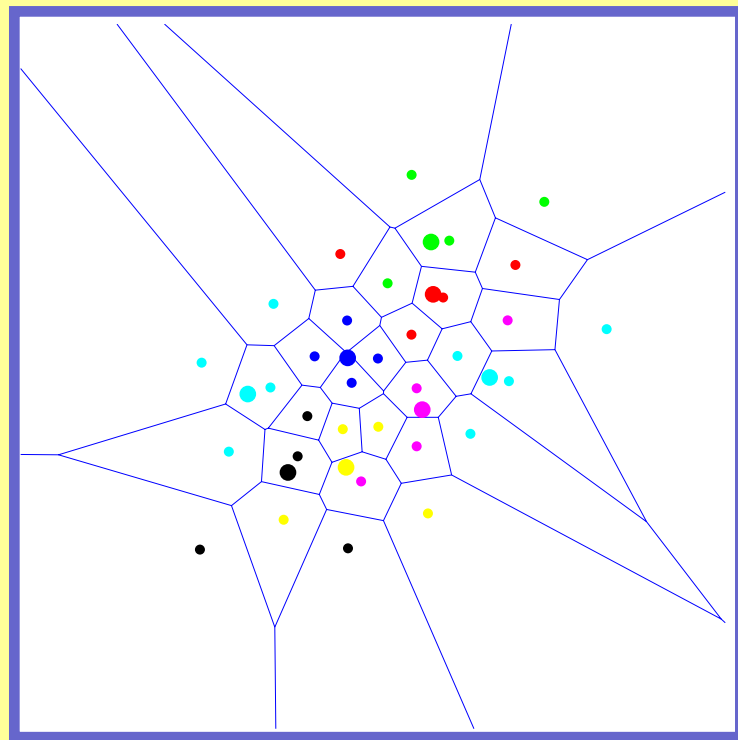
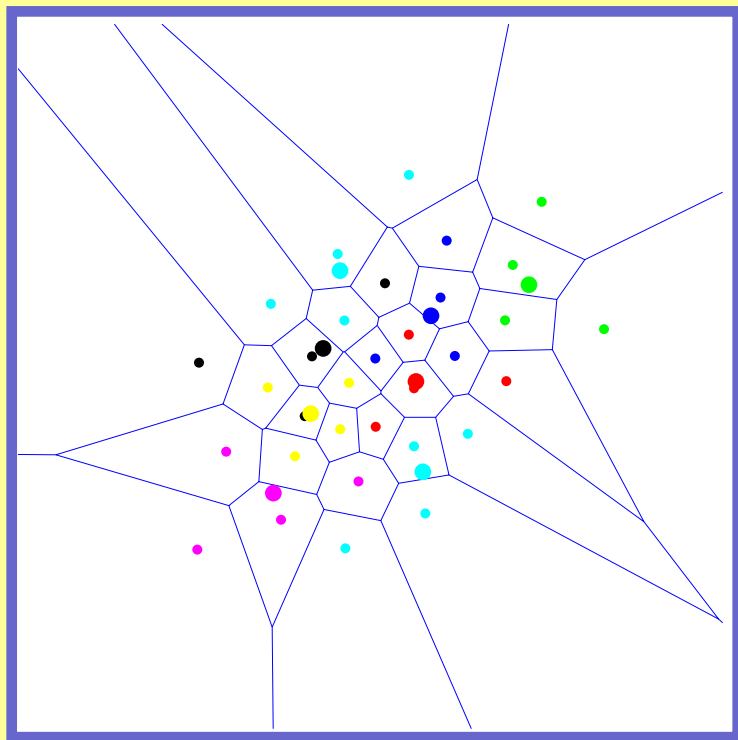
1	3				
2	4	5			
	6	7	9		
		8	10	11	
			12	13	15
				14	16



- Codewords along the same row or column will remain close throughout the codebook development



Gaussian source, $d=2$, $\rho=0.5$, $N_0=8$, $N_{\text{side}} = [4 \ 4]$



Gaussian source, $d=2$, $\rho=0.5$, $N_0=32$, $N_{\text{side}} = [8 \ 8]$

Example with images



$SNR_0=22.1$ dB, $SNR_1=18.22$ dB, $SNR_2=18.07$ dB

Image is readable with a single description (redundancy)

2-d Gaussian i.i.d. unit power, $N_0=64$, $N_{\text{side}} = [16 \ 16]$

	MSE_0	MSE_1	MSE_2	T_{CPU}
GLA	0.032	0.277	0.274	22
GLA+BSA	0.032	0.155	0.155	189
SOM	0.031	0.174	0.168	55
SOM+BSA	0.031	0.152	0.148	163

$MSE_k > 0.111$ obtained with 4-bit unconstrained

2-d Gaussian i.i.d. unit power, $N_0=128$, $N_{\text{side}} = [16 \ 16]$

	MSE_0	MSE_1	MSE_2	T_{CPU}
GLA	0.015	0.451	0.462	116
GLA+BSA	0.015	0.268	0.274	1148
SOM	0.015	0.307	0.313	357
SOM+BSA	0.015	0.251	0.251	1111

$MSE_k > 0.111$ obtained with 4-bit unconstrained

Conclusions

- VQ is a valuable tool for any lossy coding technique
- SOM allows for the design of VQ codebooks with any desired indexing of codewords
- Various applications for ordered VQ codebooks
- Good potential for SOM-based Multiple Description VQ: great flexibility, promising early results