15-750:Algorithms in the Real World

Quantization (lossy)

Scalar Quantization

Quantize regions of values into a single value

E.g. Drop least significant bit

(Can be used to reduce # of bits for a pixel)

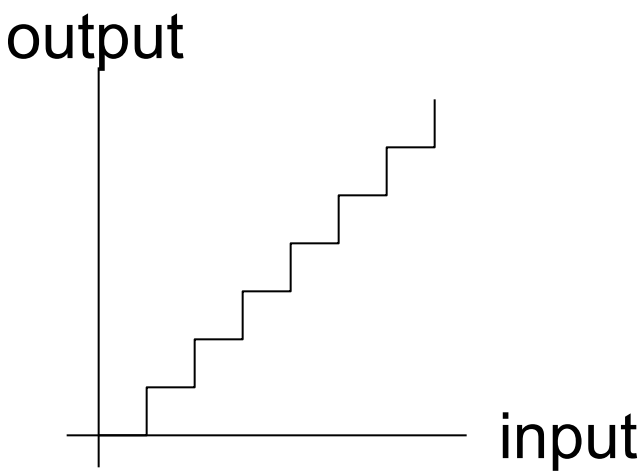
Q: Why is this lossy?

Many-to-one mapping

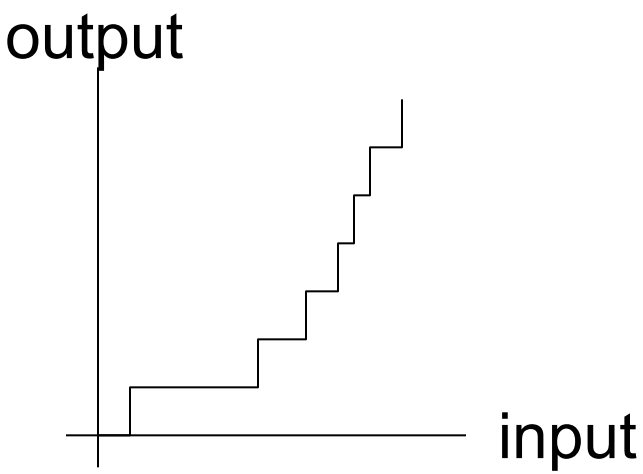
Two types

- Uniform: Mapping is linear
- Non-uniform: Mapping is non-linear

Scalar Quantization



uniform



non uniform

Q: Why use non-uniform?
Error metric might be non-uniform.
E.g. Human eye sensitivity to specific color regions

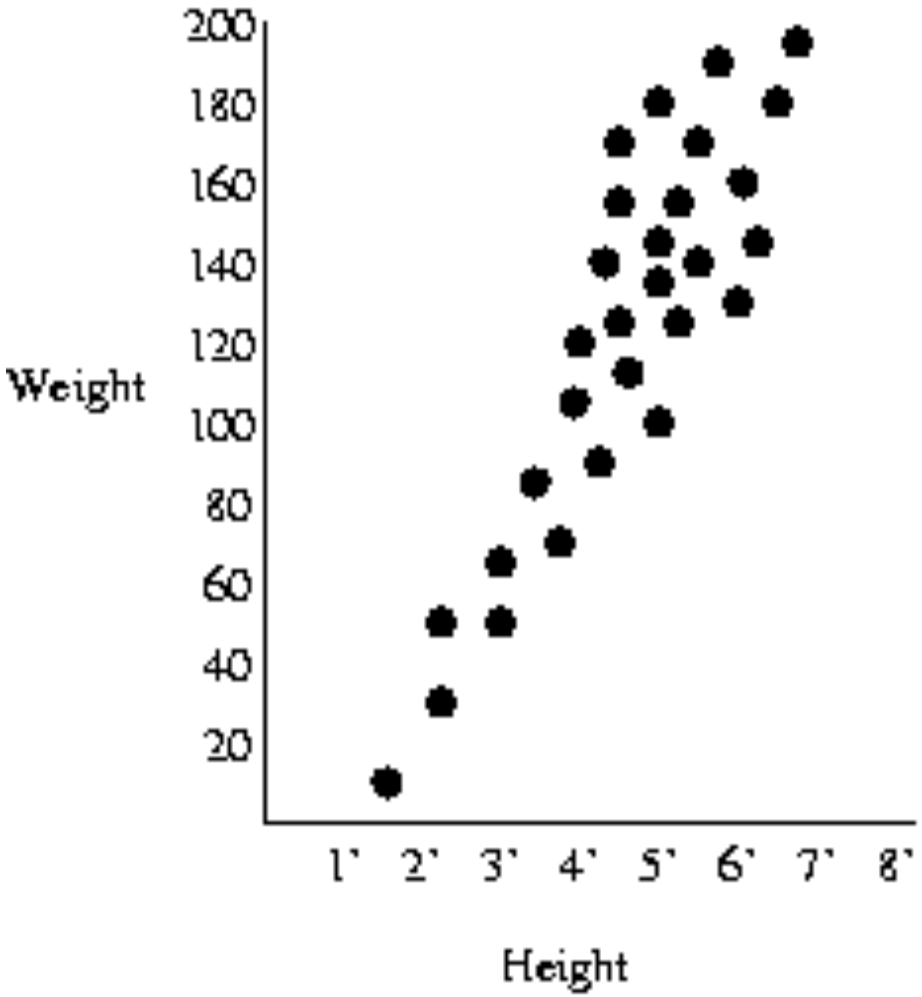
Can formalize the mapping problem as an optimization problem

Vector Quantization (VQ)

- Mapping a multi-dimensional space into a smaller set of messages
- What do we use as vectors?
  - Color (Red, Green, Blue)
    - Can be used, for example to reduce 24bits/pixel to 8bits/pixel
    - Used in some monitors to reduce data rate from the CPU (colormaps)
  - K consecutive samples in audio
  - Block of K pixels in an image
- How do we decide on a codebook
  - Typically done with **clustering**

VQ most effective when the variables along the dimensions of the space are correlated

Vector Quantization: Example



Observations:

1. Highly correlated:
   Concentration of representative points

2. Higher density is more common regions.

Case Study: JPEG

A nice example since it uses many techniques:

- Transform coding (Cosine transform)
- Scalar quantization
- Residual coding
- Run-length coding
- Huffman or arithmetic coding

15-750: Algorithms in the Real World

Algorithms for coding

(Error Correcting Codes)

Welc**e t* t*is clas* o* c*d*ng .
Y*u a** in f*r a f*n rid*!

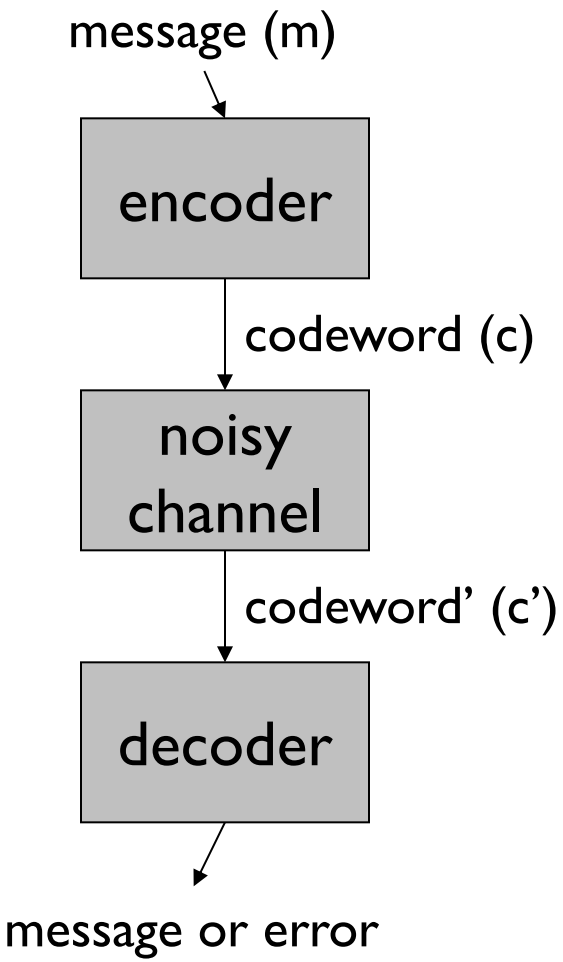
What do these sentences say?

Why did this work?

Redundancy!

Codes are clever ways of judiciously adding redundancy to enable recovery under “noise”.

General Model



"Noise" introduced by the channel:

- changed fields in the codeword vector (e.g. a flipped bit).
  - Called **errors**
- missing fields in the codeword vector (e.g. a lost byte).
  - Called **erasures**

How the decoder deals with errors and/or erasures?

- **detection** (only needed for errors)
- **correction**

Applications

Numerous applications:

Some examples

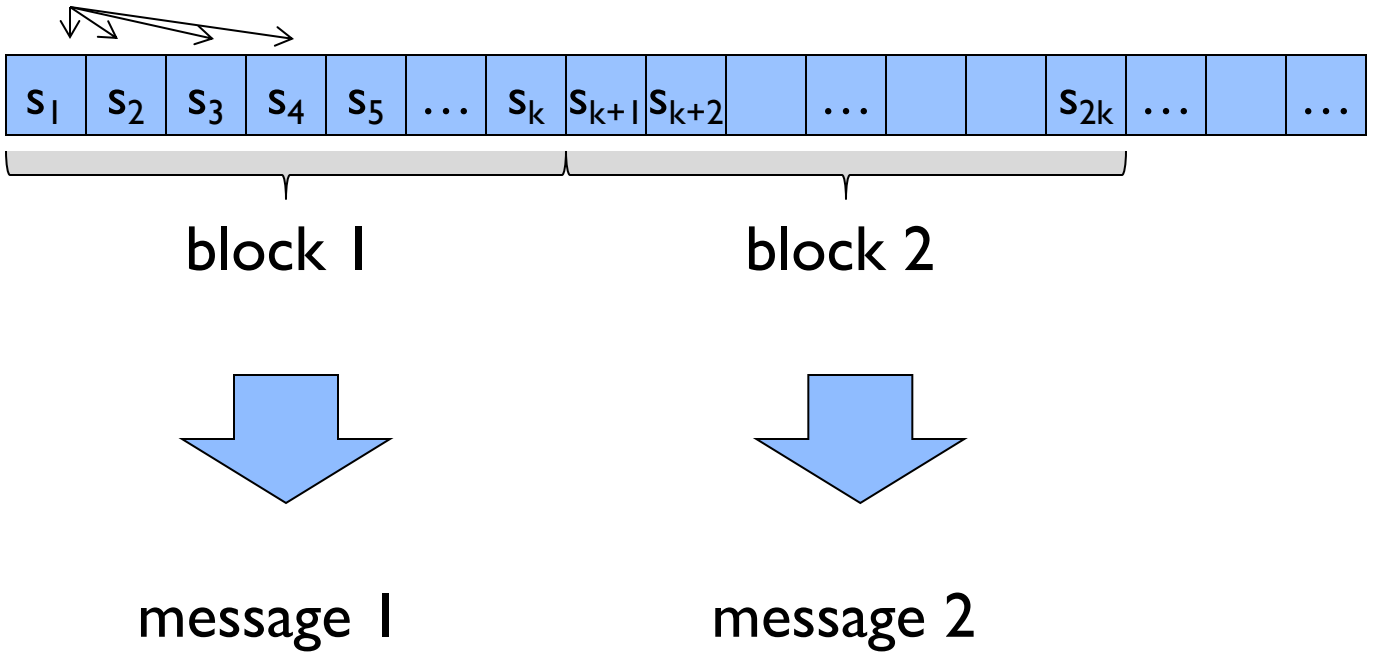
- **Storage**: Hard disks, cloud storage, NAND flash…
- **Wireless**: Cell phones, wireless links,
- **Satellite and Space**: TV, Mars rover, …

Reed-Solomon codes are by far the most used in practice.

Low density parity check codes (LDPC) codes used for 4G (and 5G) communication and NAND flash

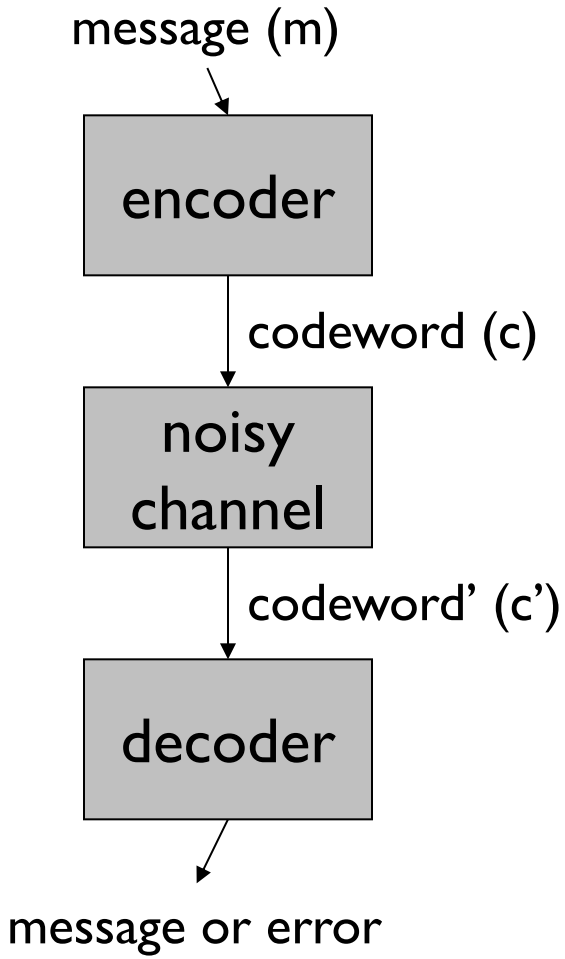
Block Codes

symbols (e.g., bits)



Other kind: convolutional codes (we won't cover it)…

Block Codes



- Each message and codeword is of fixed size
- Notation:

"dimension of the code"

k = |m|

length of the message

"length of the code"

n = |c|

length of the codeword

C = "code" = set of codewords

Simple Examples

3-Repetition code: k=1, n=3

Message		Codeword
0	->	000
1	->	111

- How many **erasures** can be recovered?
- How many **errors** can be **detected**?
- Up to how many **errors** can be **corrected**?

Errors are much harder to deal with than erasures.

Why?

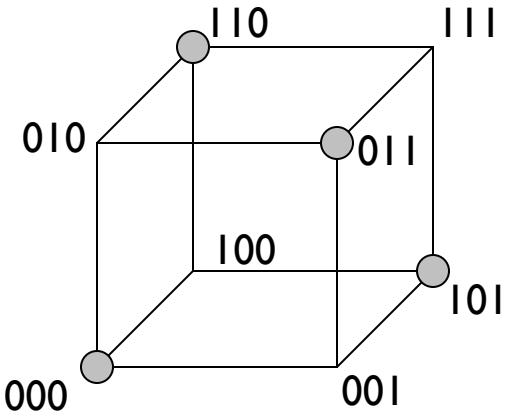
Need to find out **where** the errors are!

Simple Examples

Single parity check code: k=2, n=3

Message		Codeword
00	->	000
01	->	011
10	->	101
11	->	110

Consider codewords as vertices on a hypercube.



- codeword

n = 3 (hypercube dimensionality)

$2^n = 8$ (number of nodes)

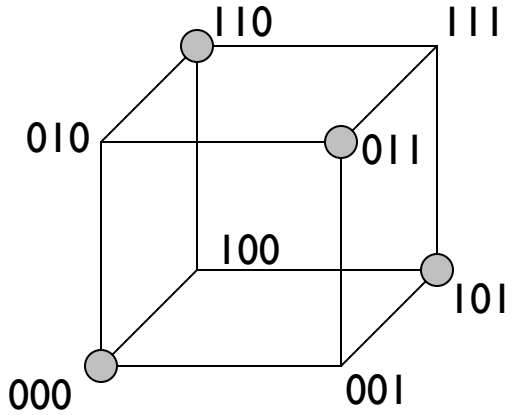
Systematic codes

Definition: A **Systematic code** is one in which the message symbols appear in the codeword in uncoded form

message	codeword
000	**000**000
001	**001**011
010	**010**101
011	**011**110
100	**100**110
101	**101**101
110	**110**011
111	**111**000

Simple Examples

Single parity check code: k=2, n=3



- How many **erasures** can be recovered?
- How many **errors** can be **detected**?
- Up to how many **errors** can be **corrected**?

Erasure correction = 1, error detection = 1, error correction = 0

Cannot even correct single error. Why?

Codewords are too "close by"

Let's formalize this notion of distance..

Block Codes

Notion of distance between codewords: **Hamming distance**

$\Delta(\mathbf{x},\mathbf{y})$ = number of positions s.t. $x_i \neq y_i$

Minimum distance of a code

$\mathbf{d} = \min\{\Delta(x,y) : x,y \in C, x \neq y\}$

Code described as: $\mathbf{(n, k, d)_q}$ ←

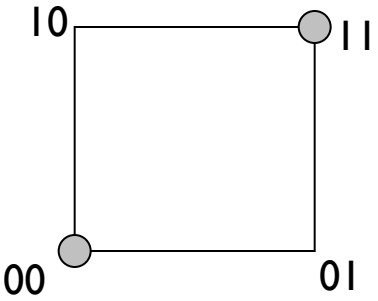
- Σ = alphabet
- $\mathbf{q} = |\Sigma|$ = alphabet size
- $\mathbf{C} \subseteq \Sigma^n$ (codewords)

Question:

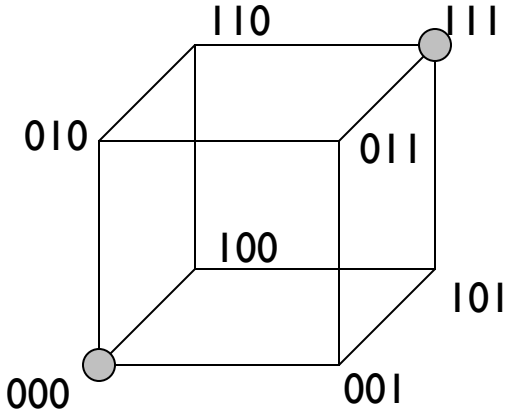
What alphabet did we use so far?

Error Correcting One Bit Messages

How many bits do we need to correct a one bit error on a one bit message?



2 bits
0 -> 00, 1-> 11
(n=2,k=1,d=2)



3 bits
0 -> 000, 1-> 111
(n=3,k=1,d=3)

In general need $d \geq 3$ to correct one error. Why?