Relation between PCA and SVD

Eigen Value Decomposition of M

$$M = X^T X = V\Sigma^T U^T U \Sigma V^T = V\Sigma^T \Sigma V^T$$

- The right singular vectors of X (i.e. columns of V) are the Eigen vectors of M
- Eigen values of M are squared singular values of X

So can use SVD of X for dimension reduction.

Keeping the top-k singular values (and setting rest to 0) is the best rank-k approximation to a matrix under various norms (e.g., spectral, nuclear, forbenius).

[Eckert-Young Theorem]

Finding SVD: via computing eigenvalues and vectors for $X^T X$

Computing eigen values/vectors

How to compute eigen values/vectors of a symmetric matrix M?

Power iteration

- Widely used in practice
- Comes up in other scenarios as well, for example, Pagerank

High level steps:

- Compute the principal eigen vector (to get the first eigen vector)
- Subtract out the principal eigen vector from the matrix
- Iterate again using power iteration to find the principal eigen vector (which will be the second eigen vector of the original matrix)

Computing eigen values/vectors

Power iteration to find the principal eigen value/vector

Assume there is (strict) gap between the top 2 eigen vectors.

Power iteration algorithm:

1. Pick a random vector y_0 (say, each coordinate N(0,1))
2. $y_{i+1} = \frac{My_i}{||My_i||_2}$ (i.e. rescaling to norm 1)
3. Iterate until convergence to get (approximate) v_1
4. $\lambda_1 = v_1^T M v_1$

Then subtract this out, $M - \lambda_1 v_1 v_1^T$

And use power iteration again.

Intuition behind power iteration

<on board>

PCA Algorithm recap

Preprocess the data

Compute the "covariance matrix" M = $\sum_{i=1}^{n} x_i x_i^T$

Find eigen value decomposition of M = $V \Lambda V^T$

where $\Lambda = \mathrm{diag}(\lambda_1, \lambda_2 \ldots)$ and $\lambda_1 \geq \lambda_2 \geq \lambda_3 \ldots$

$V = [v_1 \; v_2 \; \ldots \; v_D]$

Set the linear transformation matrix as $S = \begin{bmatrix} v_1^T \\ v_2^T \\ \vdots \\ v_k^T \end{bmatrix}$

When PCA does not work?

- PCA finds a linear approximation. If the low dimensionality of the data is due to non-linear relationships then PCA cannot find it. E.g. (x, y) with y = x^2

- If normalization is not done incorrectly

15-750: Graduate Algorithms

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*!

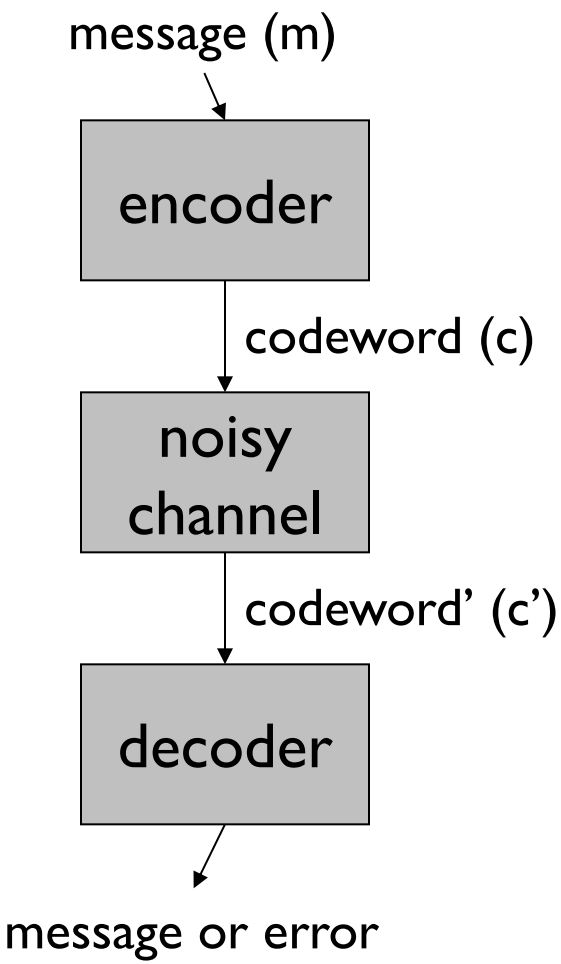
What do these sentences say?

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:

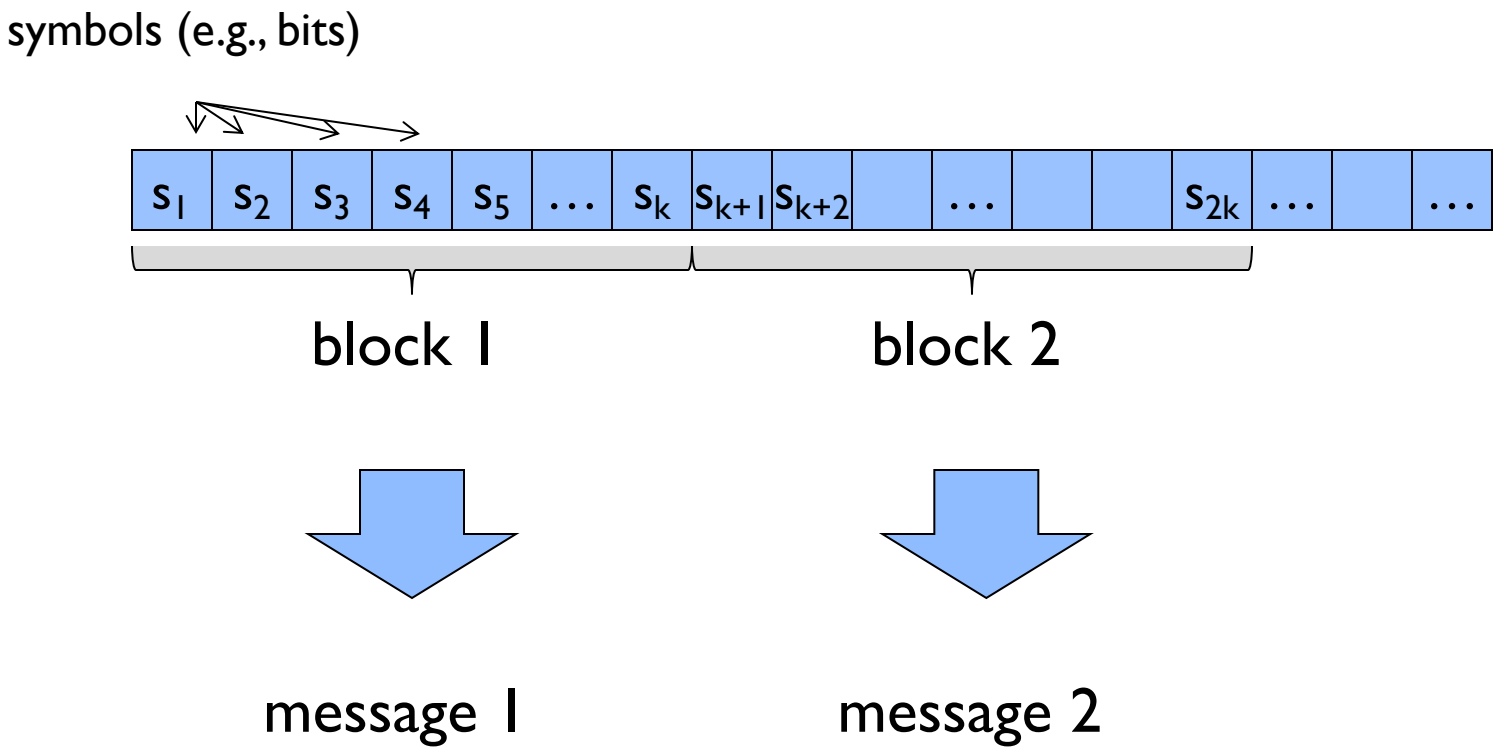
Some examples

- **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.

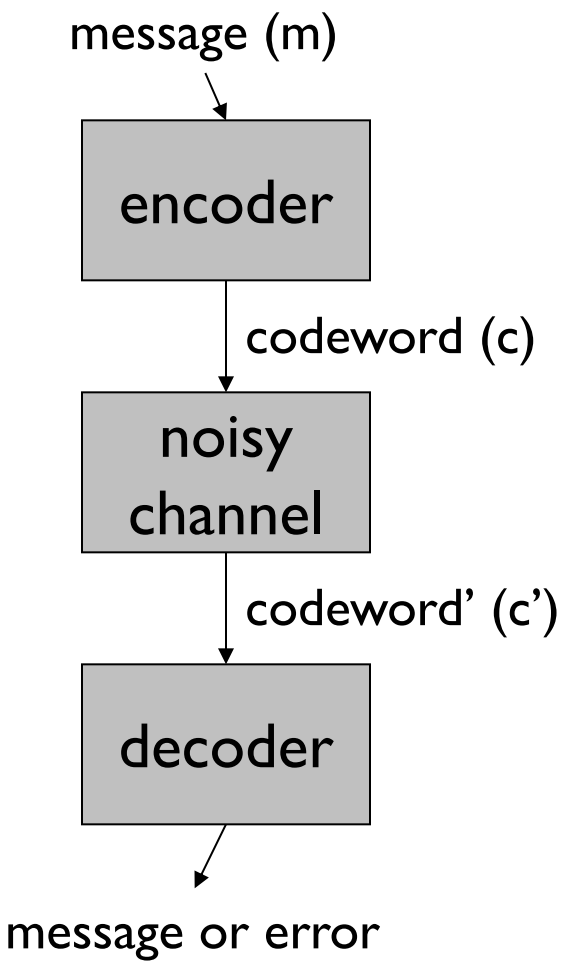
Low density parity check codes (LDPC) codes used for 4G (and 5G) communication and NAND flash

Block Codes



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

Block Codes



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

k = |m|

length of the message

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?

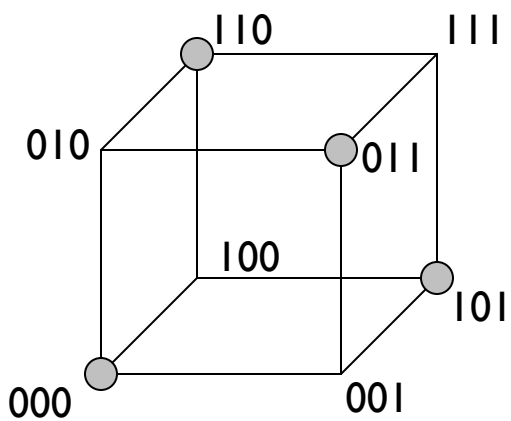
Need to find out **where** the errors are!

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.



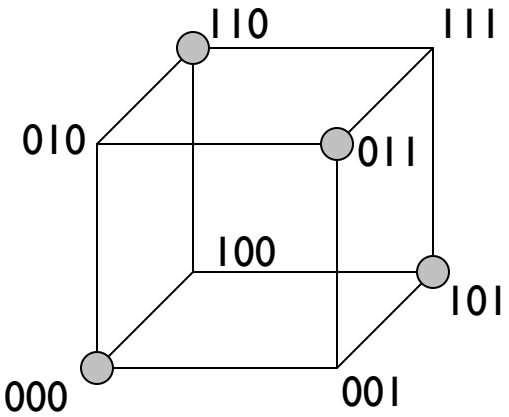
codeword

n = 3 = dimensionality

2^n = 8 = number of nodes

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**?