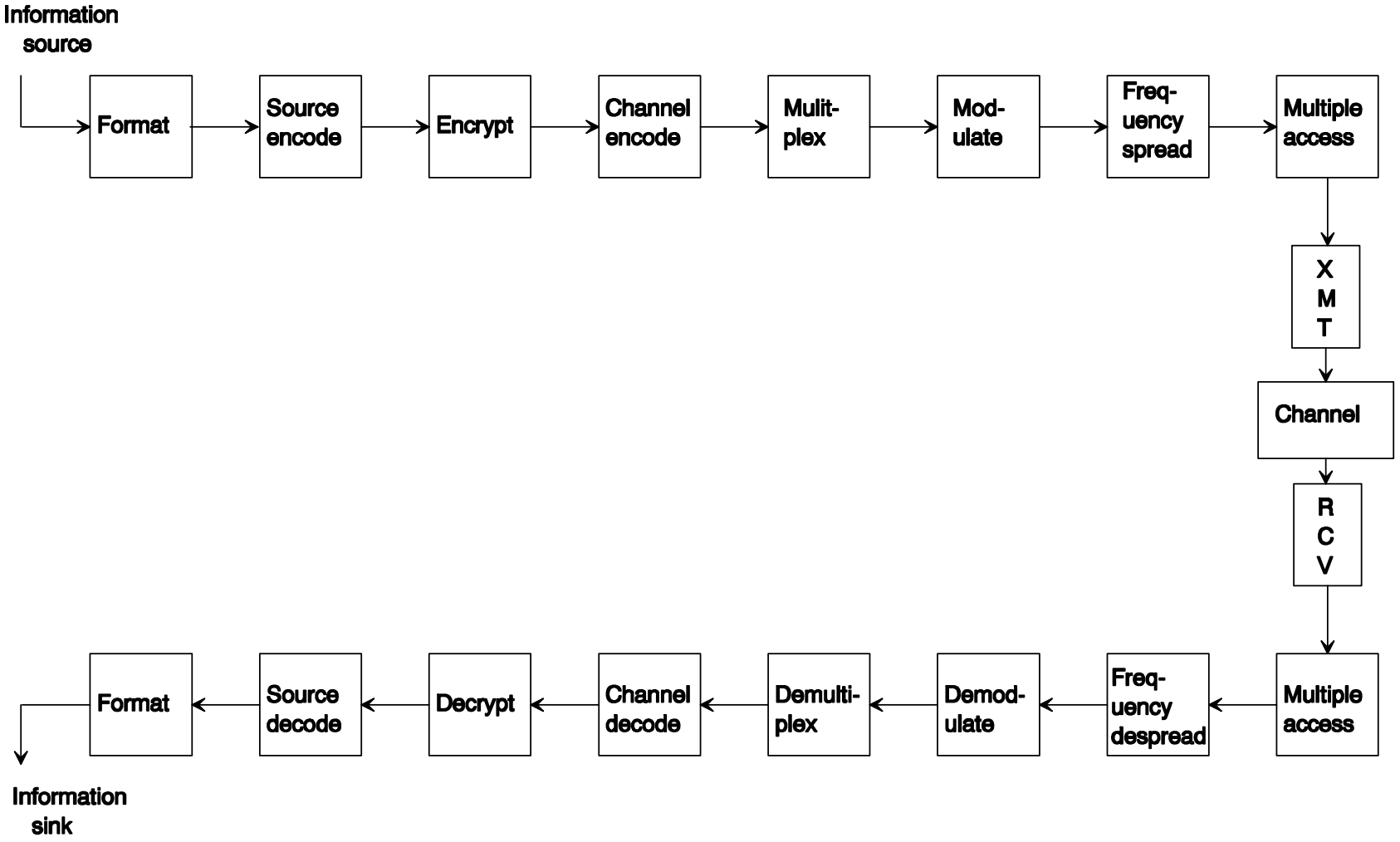
Linear Block Codes

Digital Communication System



What is Coding?

- By coding we do NOT mean
  - Source coding where the data is compressed in a lossless coding scheme (to remove redundancy)
  - Encryption where the data is mapped in a certain way in order to disguise it from an unauthorized receiver

- By coding we do mean to add extra data to an information stream so that if that if the resulting code word becomes corrupted we can recover the original data with a reasonable probability

Error Correction Capability

- Error correction capability of a code depends on the minimum distance of the code
  - Measures how close code words are to each other
  - The distance measure is Hamming distance which is the number of places where the vectors differ
  - Example: 1 1 0 0 1 vs. 1 0 0 1 0 (Hamming distance 3)
  - Minimum distance is the smallest of all distances
- For a linear code the minimum distance is simply the fewest number of 1's found in all the nonzero code vectors
- A code can correct $t = (d_{min} – 1)/2$ or fewer errors

Coding Concepts

- Channel encoder adds redundancy to data by mapping k information bits to *n* code bits (codeword)
  - resulting codewords do not look like each other
  - Code rate R = k / n
  - Input rate is r_b => output rate is $r_c = r_b / R$ bps
- An M-ary modulator maps L encoded bits to one of $M=2^L$ waveforms
  - Each waveform is of T sec duration
  - Output symbol rate $r_s = 1 / T = r_b / RL$ symbols per sec
  - Modulation is typically phase or amplitude and phase
    - Others are possible

Block Codes

Encoding Block Codes

- For k information bits we have 2^k possible messages
  - Code space consists of 2^n places for encoder to map the information bits
- An (n,k) linear code is a k-dim subspace of the n-tuple vector space V_n
- Can generate the block code by k linearly independent binary n-tuples $g_0, g_1, \ldots, g_{k-1}$
- Let the message be $u = (u_0, u_1, \ldots, u_{k-1})$
- Then the codeword is

$$v = u_0 g_0 + u_1 g_1 + \cdots + u_{k-1} g_{k-1}$$

Encoding Block Codes (cont.)

- Can write

  v = u • G

  where

$$\mathbf{G} = \begin{bmatrix} \mathbf{g}_0 \\ \mathbf{g}_1 \\ \vdots \\ \mathbf{g}_{k-1} \end{bmatrix} = \begin{bmatrix} g_{00} & g_{01} & \cdots & g_{0,n-1} \\ g_{10} & g_{11} & \cdots & g_{1,n-1} \\ \vdots & \vdots & \vdots & \vdots \\ g_{k-1,0} & g_{k-1,1} & \cdots & g_{k-1,n-1} \end{bmatrix}$$

  – G is called the generator matrix of the code

Encoding Block Codes (cont.)

- Can write G in systematic form so that the information bits are actually part of the codeword explicitly, i.e., the message is mapped as

$$(u_0, u_1, \ldots, u_{k-1}) \rightarrow (v_0, v_1, \ldots, v_{n-k-1}, u_0, u_1, \ldots, u_{k-1})$$

- In this case $\mathbf{G} = [\mathbf{P} \quad \mathbf{I}_k]$

  where

$$\mathbf{P} = \begin{bmatrix} p_{00} & p_{01} & \cdots & p_{0,n-k-1} \\ p_{10} & p_{11} & \cdots & p_{1,n-k-1} \\ \vdots & \vdots & \vdots & \vdots \\ p_{k-1,0} & p_{k-1,1} & \cdots & p_{k-1,n-k-1} \end{bmatrix}.$$

Encoding Block Codes (cont.)

- Since **v** = **u** • **G**, if **G** is in systematic form we can write **v** = [**a u**] where **a** = **u** • **P** which implies **a** + **u** • **P** = 0 which may be written as

$$\left[\mathbf{a}\ \ \mathbf{u}\right]\cdot\begin{bmatrix}\mathbf{I}_{n-k}\\ \mathbf{P}\end{bmatrix}=\mathbf{0}$$

- Thus

$$\mathbf{v}\cdot\mathbf{H}^{T}=0$$

where

$$\mathbf{H}=\left[\mathbf{I}_{n-k}\ \ \mathbf{P}^{T}\right].$$

  - **H** is called the parity check matrix and specifies the code like **G** does

Example 1

Example. Consider (7, 4) linear code with generator

$$\mathbf{G}^{*} = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Replacing row 3 with row 3 + row 4 and then replacing row 2 with row 2 + row 3 we get G in systematic form as

$$\mathbf{G} = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Example 2

Example. Consider (7, 4) linear code with generator

$$\mathbf{G} = \begin{bmatrix} g_0 \\ g_1 \\ g_2 \\ g_3 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Let the message

$$\mathbf{u} = (1\ 1\ 0\ 1)\ .$$

Compute **v** and the parity check matrix **H** and verify that $\mathbf{vH}^{\mathrm{T}} = \mathbf{0}$.

Example 2 Solution

Example. Consider (7, 4) linear code with generator

$$\mathbf{G} = \begin{bmatrix} g_0 \\ g_1 \\ g_2 \\ g_3 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Let the message

$$\mathbf{u} = (1\ 1\ 0\ 1)$$

then

$$\mathbf{v} = \mathbf{uG}$$
$$= (1\ 1\ 0\ 1\ 0\ 0\ 0) + (0\ 1\ 1\ 0\ 1\ 0\ 0) + (1\ 0\ 1\ 0\ 0\ 0\ 1)$$
$$= (0\ 0\ 0\ 1\ 1\ 0\ 1)$$

Note $\mathbf{G}$ is in systematic form: $\mathbf{G} = \left[\mathbf{P}\ \ \mathbf{I}_4\right]$.

$\mathbf{P}$ is $4 \times 3 = k \times (n-k)$.

Example 2 Solution (cont.)

The parity check matrix $\mathbf{H}$ is $\mathbf{H} = \left[\mathbf{I}_3 \;\; \mathbf{P}^T\right]$

$$\mathbf{H} = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{bmatrix}$$

$\mathbf{H}$ is $(n-k)\times n$.

For $\mathbf{u} = (1\ 1\ 0\ 1)$ we got $\mathbf{v} = (0\ 0\ 0\ 1\ 1\ 0\ 1)$. Observe

$$\mathbf{vH}^T = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}$$

Decoding Block Codes

- Hamming weight of codeword **v** is the total number of nonzero components

- Hamming distance d(**v**,**w**) between two codewords is the number of places in which v and w differ

- Minimum Hamming distance is the smallest Hamming distance between all possible distinct pairs of codewords

Decoding Block Codes (cont.)

- In maximum likelihood decoding we want to choose the estimate of the transmitted codeword as the vector **v** that maximizes the conditional density p(**r**|**v**) where **r** is the received vector
  - For a binary symmetric channel with hard decisions at the demodulator and transition probability p we have

$$p(\mathbf{r}|\mathbf{v}) = p^{d(\mathbf{r},\mathbf{v})}(1-p)^{n-d(\mathbf{r},\mathbf{v})}.$$

  - Now p(**r**|**v**) > p(**r**|**w**) iff d(**r**,**v**) < d(**r**,**w**) so the best thing we can do is find the codeword vector v that minimizes d(**r**,**v**)

Decoding Block Codes (cont.)

- For a Gaussian channel using soft decisions at the demodulator we have

  $r_i = x_i + n_i$

  – where n_i is a Gaussian sample of mean 0 and variance σ^2 and x_i is +1 or –1 and represents the transmitted data

- In this case

$$p(\mathbf{r}|\mathbf{v}) = \prod_{i=0}^{n-1} \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(r_i - x_i)^2}{2\sigma^2}\right)$$

- Which may be written as

$$p(\mathbf{r}|\mathbf{v}) = \left(\frac{1}{\sqrt{2\pi}\sigma}\right)^n \exp\left(-\sum_{i=0}^{n-1} \frac{(r_i - x_i)^2}{2\sigma^2}\right).$$

Block Codes (cont.)

- This last function is maximized when

$$d_E^2 = \sum_{i=0}^{n-1} (r_i - x_i)^2$$

  is minimized

- Thus the optimal decoder computes the squared Euclidean distance between the received sample and all the modulated codewords and selects the codeword that minimizes this last expression as the transmitted codeword estimate
  - Hard to do for long codes which is why we study various other encoding/decoding techniques

Example 3

Exercise. BSC. Hard Decisions.

Let

$$v = 1\ 1\ 0\ 0\ 1$$

say

$$r = 1\ 0\ 0\ 1\ 1$$

If

$$P(1 \to 0) = P(0 \to 1) = p$$

then compute $p(\mathrm{r}|\mathrm{v})$.

Example 3 Solution

Example. BSC. Hard Decisions.

Let

$$v = 1\ 1\ 0\ 0\ 1$$

say

$$r = 1\ 0\ 0\ 1\ 1$$

Suppose $P(1 \to 0) = P(0 \to 1) = p$ then

$$p(r \mid v) = (1-p)\,p(1-p)\,p(1-p) = p^2(1-p)^3 = p^{d(r,v)}(1-p)^{n-d(r,v)}$$

Hamming Codes

Hamming Codes

- Hamming codes were the first class of linear codes designed for error correction
- For any positive integer m >= 3 there is a Hamming code such that
  - Code length: $n = 2^m - 1$
  - Number of information symbols: $k = 2^m – m – 1$
  - Number of parity check symbols: $n – k = m$
  - Error-correcting capability: $t = 1$ ($d_{min} = 3$)

How Hamming did it

- The syndrome results from writing a 0 for each of the parity bits that are correct and a 1 for each failure
- We can think of the syndrome as an m-bit number so it can represent at most 2^m things
- We need to represent the state of all the message symbols being correct plus the location of any single error in the n bits thus $2^m >= n+1$

Example 4

Example. Let us design for k = 4 message digits. Let m = 3. Note that 2^3 >= (4+3)+1 = 8. In fact, 2^3 = 8.

Position	Binary Rep
1	0 0 1
2	0 1 0
3	0 1 1
4	1 0 0
5	1 0 1
6	1 1 0
7	1 1 1

We will use positions 1, 2, 4 of the code vector for the 3 parity checks. Every position that has a 1 in the last column of the binary rep. is the first parity check. The second parity check covers the next column over and so forth.

The 1st parity check covers positions 1, 3, 5, 7.
The 2nd parity check covers positions 2, 3, 6, 7.
The 3rd parity check covers positions 4, 5, 6, 7.

Example 4 (cont.)

Encode:

Position	1	2	3	4	5	6	7
Message	-	-	1	-	0	1	1
Encode	0	1	1	0	0	1	1
Receive	0	1	**0**	0	0	1	1
			error				

Decode:

Check 1	1	3	5	7
	0	0	0	1 **fails => 1**
Check 2	2	3	6	7
	1	0	1	1 **fails => 1**
Check 3	4	5	6	7
	0	0	1	1 correct =>0

The syndrome is 0 1 1 which is the binary rep. of 3 so position 3 is in error. We add 1 to position 3 of the received vector to get

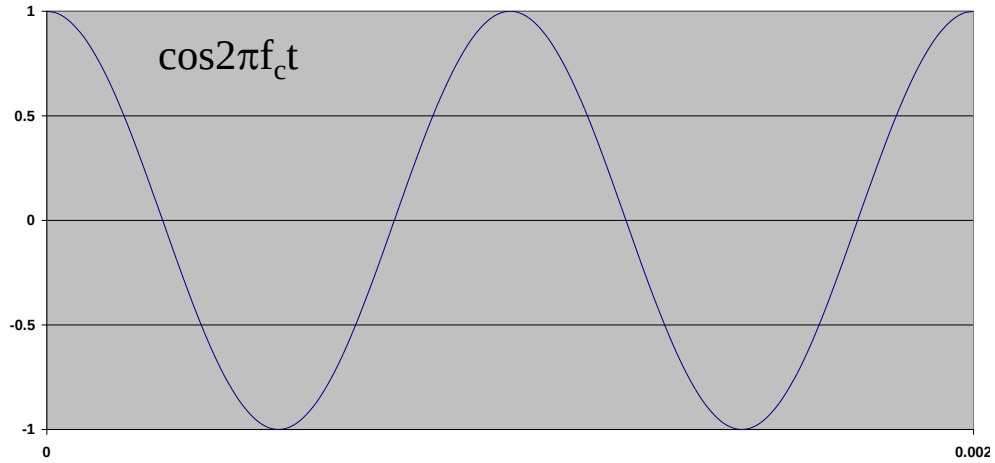
0 1 1 0 0 1 1

as the decoded error vector.

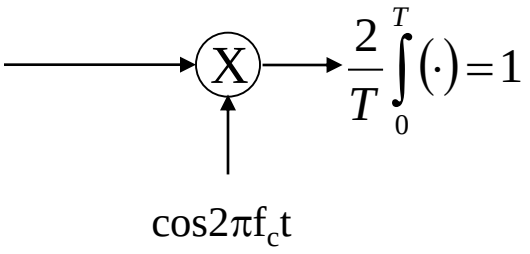
Supplement Presentation

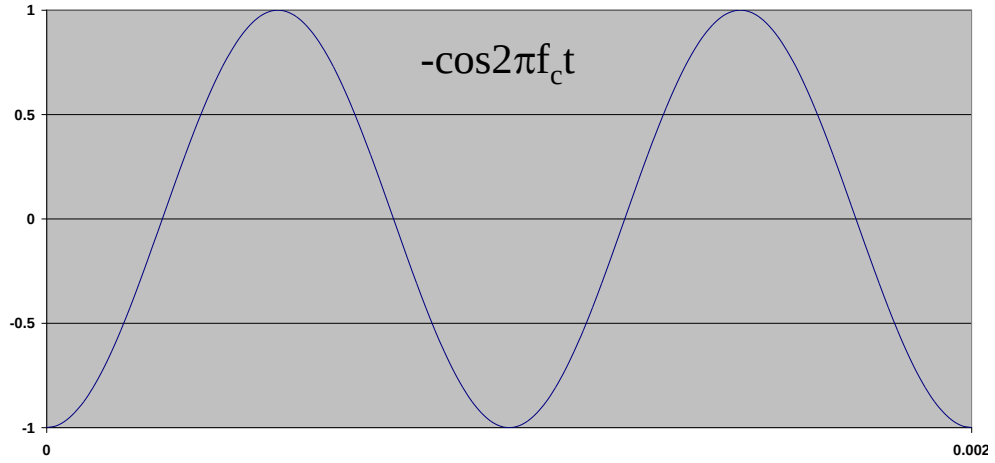
Where do the 1’s and 0’s come from?

Modulation/Demodulation Example

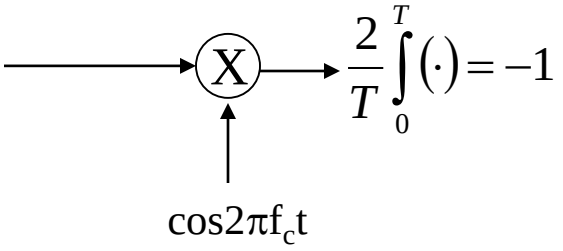


1



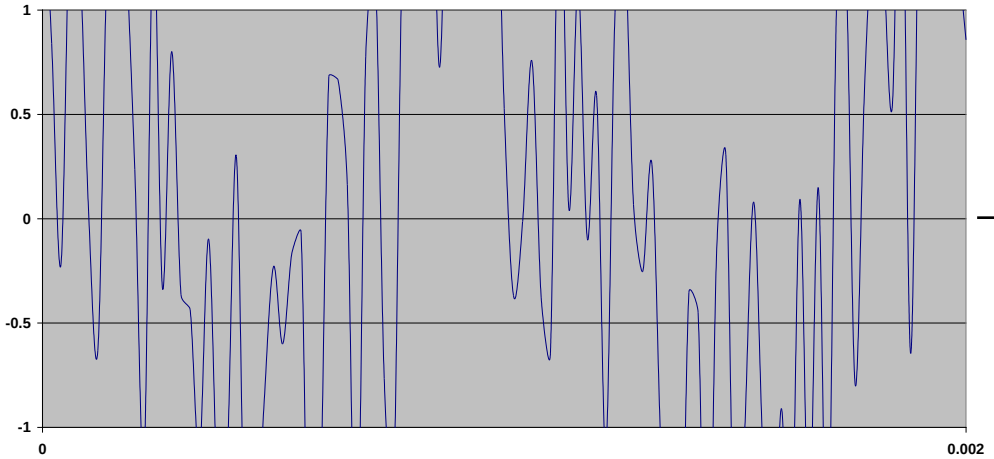


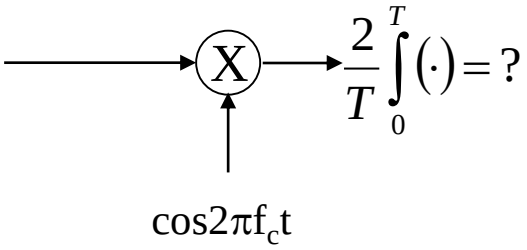
0



Note: When we modulate we can map a 0 to -1

Modulation/Demodulation Example (cont.)





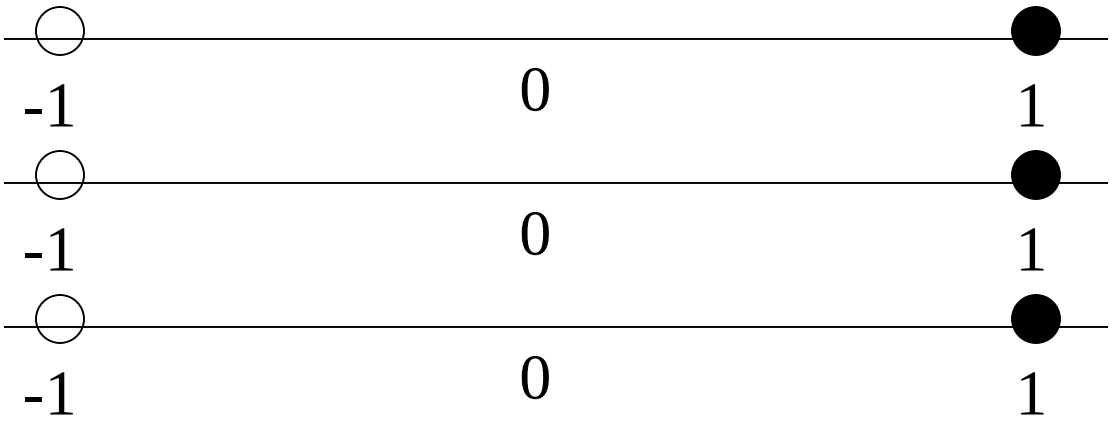
The above is a noisy version of a +1 signal

On Hard and Soft Decisions

- For some codes such as Hamming codes and Reed-Solomon codes we usually use hard decisions in the decoding algorithms
  - By this we mean we make a decision on each received bit coming out of the demodulator as being a 1 or 0 or perhaps (1 or –1)
- For codes such as turbo codes we make soft decisions
  - By this we mean we quantize the demodulator output using more than 1 bit
  - This way we have more resolution in the demod outputs which we can often use to improve the BER performance over using hard decisions

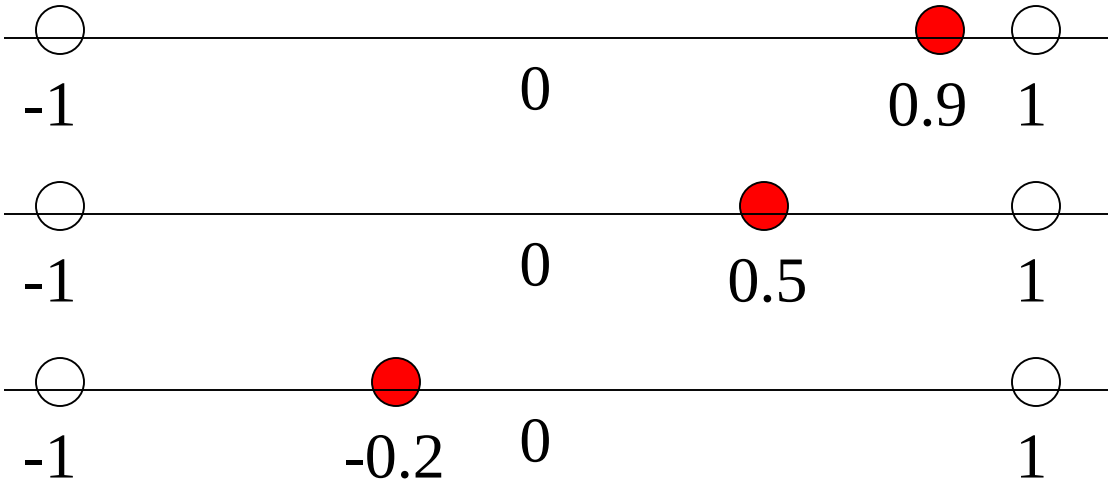
Example

- Suppose we use a simple repeat code (repeat the bit 3 times) and transmit 1 1 1



Example (cont.)

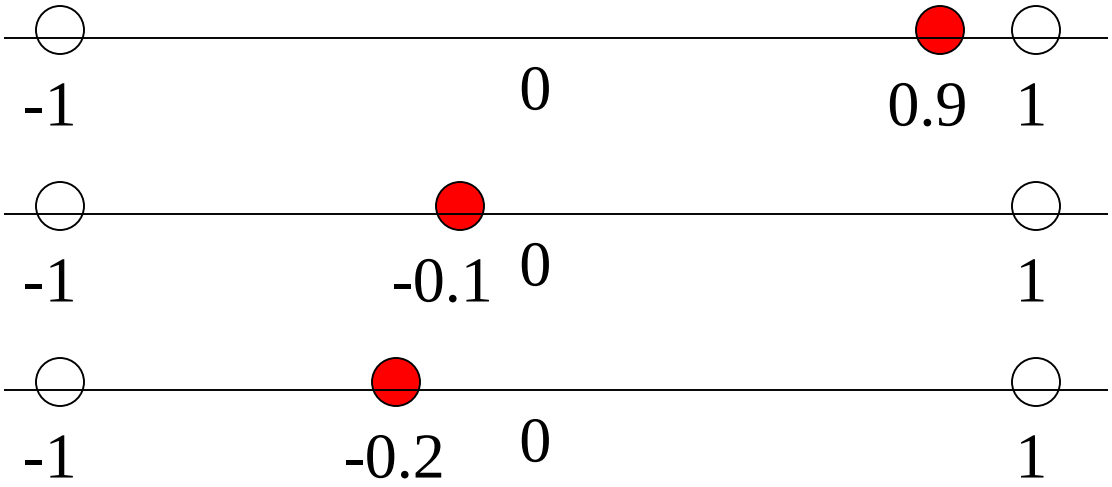
- Say we receive the following



- Making hard decisions we get 1 1 0 (1 1 -1) so we decide 1 was sent

Example (cont.)

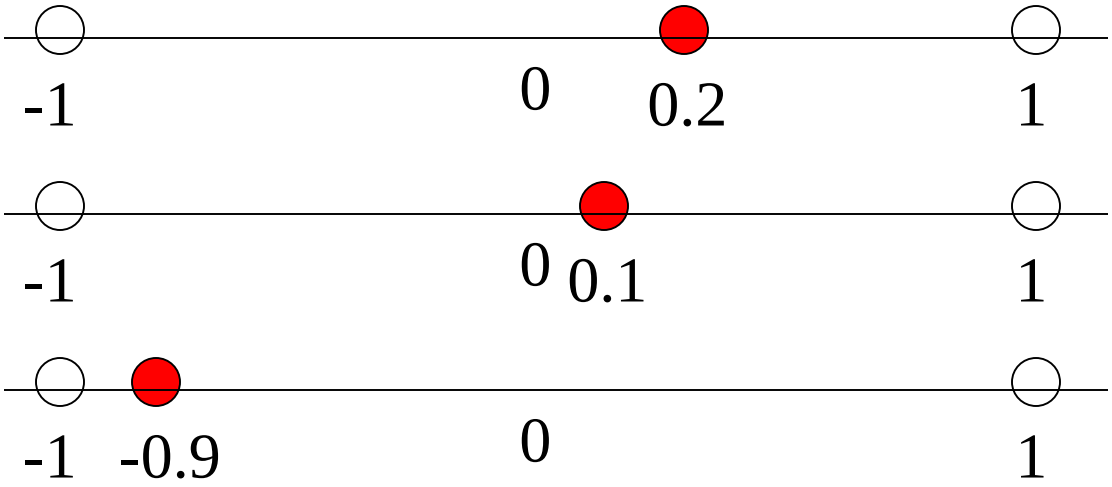
- But if we receive



- Making hard decisions we get 1 0 0 so we decide 0 was sent and thus we make an error
- But if we use soft decisions in this example we get the metrics 2.66 (for 1) and 5.06 (for –1) so we decide correctly that 1 was sent

Example (cont.)

- However if we receive



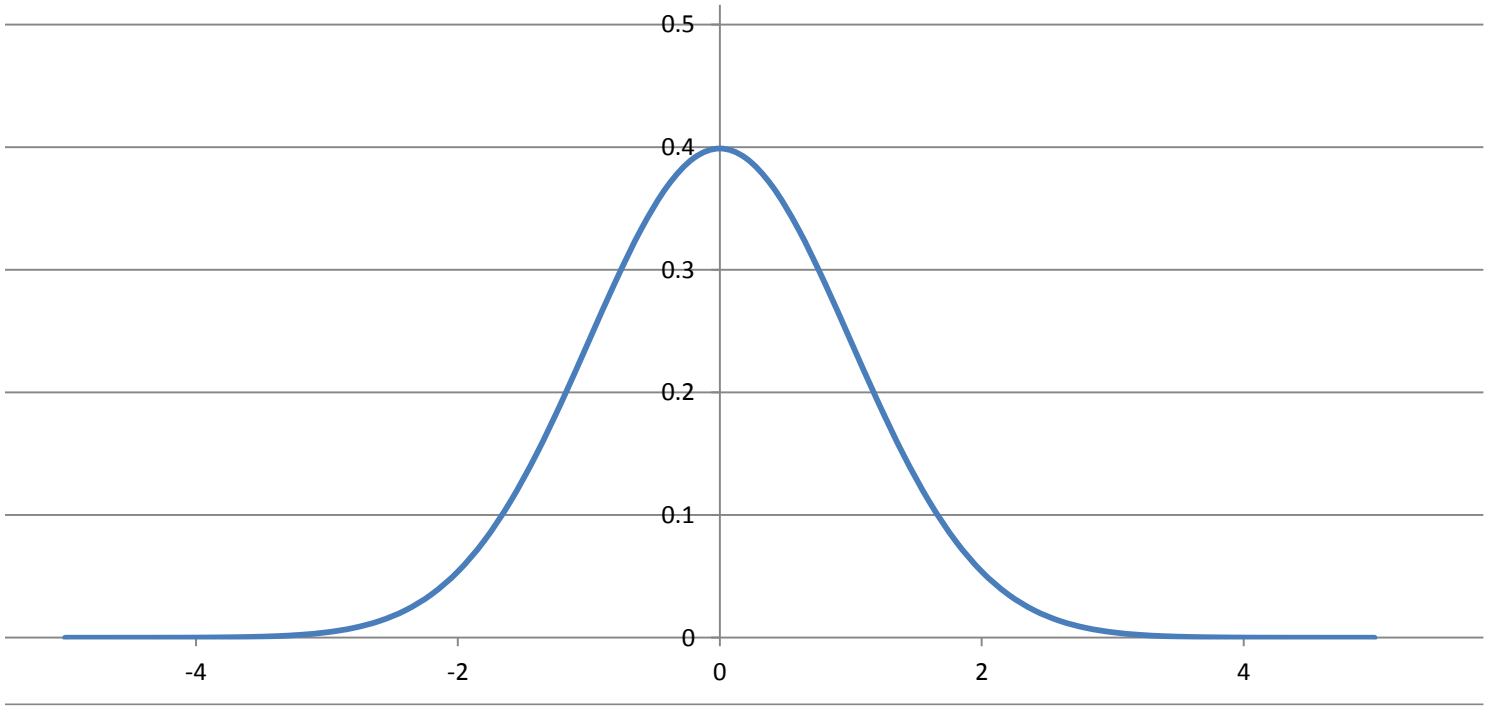
- Making hard decisions we decide 1 was sent
- Making soft decisions we decide 0 was sent (an error)
- Despite this last example soft decisions give better BER performance and are preferred if the decoding algorithm can utilize soft decisions

Confidence Intervals

Simulation and Coding

- Suppose we have a code for which we do not have an analytic expression for the decoded probability of codeword error
  - Many recently developed codes fall into this category (turbo codes, LDPC codes, etc.)
- We can simulate the decoding process and estimate the probability of codeword error by counting errors that occur (we know the information sequence so we can do this)
- Question: How many errors should you count before you can state confidently that you have a reliable estimate for the probability of codeword error for the code?

Standard Normal Density



Confidence Intervals

P_{cw} = probability of codeword error

$\hat{P}_{cw}$ = estimated probability of codeword error

$\hat{P}_{cw} = \frac{x}{n}$, x = number of errors counted, n = number of codewords simluated

Note that $n\hat{P}_{cw}$ is binomially distributed with mean nP_{cw} and standard deviation $\sqrt{nP_{cw}(1-P_{cw})}$

So,

$$\frac{n\hat{P}_{cw} - nP_{cw}}{\sqrt{nP_{cw}(1-P_{cw})}} = \frac{\hat{P}_{cw} - P_{cw}}{\sqrt{\frac{P_{cw}(1-P_{cw})}{n}}} \sim Z, \text{ where } Z \text{ is the standard normal random variable}$$

Note : We need $n\hat{P}_{cw} \geq 5$ and $n(1-\hat{P}_{cw}) \geq 5$ for this approximation to be accurate enough and since $\hat{P}_{cw}$ is unknown we use

$$\frac{\hat{P}_{cw} - P_{cw}}{\sqrt{\frac{\hat{P}_{cw}(1-\hat{P}_{cw})}{n}}} \sim Z$$

Confidence Intervals (cont.)

We can solve

$$-Z_{\alpha/2} < \frac{\hat{P}_{cw} - P_{cw}}{\sqrt{\frac{\hat{P}_{cw}(1-\hat{P}_{cw})}{n}}} < Z_{\alpha/2}$$

to get a $100(1-\alpha)\%$ confidence interval for P_{cw} as

$$\hat{P}_{cw} \pm Z_{\alpha/2}\sqrt{\frac{\hat{P}_{cw}(1-\hat{P}_{cw})}{n}}$$

Example: Suppose with $n = 100{,}000$ we counted 100 errors so $\hat{P}_{cw} = 0.001$ and we want a 95% confidence interval (so $Z_{\alpha/2} = 1.96$).

We find the confidence interval to be

$$0.001 \pm 1.96 \times 10^{-4}$$