LDPC Codes

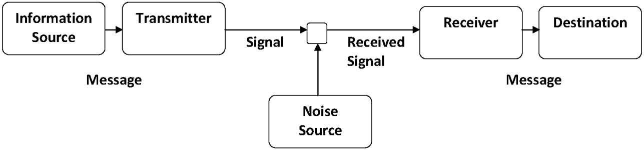
A Very Short Introduction

Dmitry Adamskiy

UCL

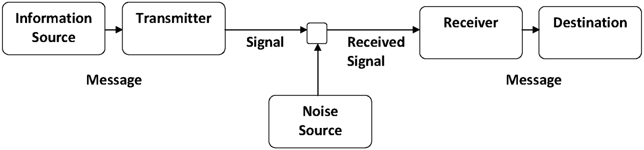
The problem

- Reliable communication over noisy channel (Shannon model):



The problem

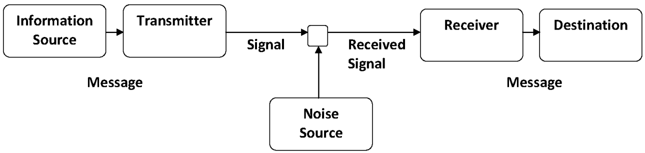
- Reliable communication over noisy channel (Shannon model):



- If we don't to anything to our message (Signal $=$ Message), then if the channel is noisy, the reliable communication is not possible.

The problem

- Reliable communication over noisy channel (Shannon model):



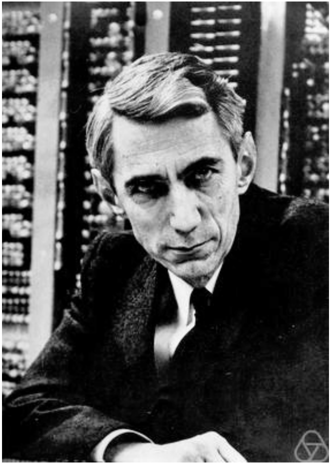
- If we don't to anything to our message (Signal = Message), then if the channel is noisy, the reliable communication is not possible.
- We need to add some redundancy to the message $\Longrightarrow$ codes.

Channels and Codes

- A channel is a triple of (A_1, A_2, P), where A_1 and A_2 are the input and output alphabets and $P(y|x)$ is a probability that y is received given that x was transmitted.
- A code is a (finite) set of vectors over the input alphabet. In a block code all these vectors have the same length n.
- If we have K different vectors, where $K = 2^k$, then we call k/n a *rate* of a code (in n times use of a channel, we transmitted k bits).

The problem

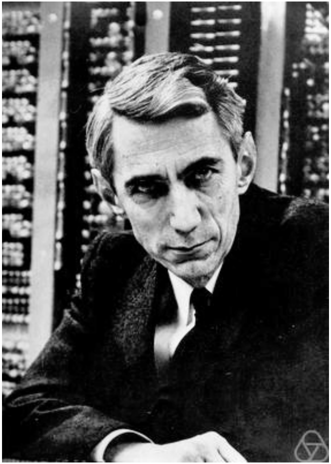
- Famous Shannon-Hartley Theorem about channel capacity
- Gives the guarantee that the reliable transmission is possible for rates below capacity (and not possible for the rates above).



Claude Shannon (1916-2001)

The problem

- Famous Shannon-Hartley Theorem about channel capacity
- Gives the guarantee that the reliable transmission is possible for rates below capacity (and not possible for the rates above).
- But is not constructive!



Claude Shannon (1916-2001)

Channels

Two common channels are

- Binary Erasure Channel: $A_1 = \mathbb{F}_2$, $A_2 = \mathbb{F}_2 \cup \{E\}$ and each bit is erased with probability p (and preserved with probability $1 - p$). Capacity is $1 - p$.
- Binary Symmetric Channel: $A_1 = A_2 = \mathbb{F}_2$ and each bit is flipped with probability p. Capacity is $1 - H(p) = 1 + p \log_2 p + (1 - p) \log_2(1 - p)$

Channels and codes: encoding and decoding

- How do we decode? In principle, if the channel is noisy, we could not be absolutely sure.

Channels and codes: encoding and decoding

- How do we decode? In principle, if the channel is noisy, we could not be absolutely sure.
- We could list all the codewords and calculate their likelihood $P(X|Y)$ and find the most likely one. This is expensive and could still make an error but that's the best we could do.
- Shannon theorem states that there are codes with rates arbitrarily close to the capacity, such that the probability of error ML-decoder make goes to zero with increasing block length.
  - How to build codes close to the capacity?
  - How to encode/decode such codes?

Channels and codes: encoding and decoding

- How do we decode? In principle, if the channel is noisy, we could not be absolutely sure.
- We could list all the codewords and calculate their likelihood $P(X|Y)$ and find the most likely one. This is expensive and could still make an error but that's the best we could do.
- Shannon theorem states that there are codes with rates arbitrarily close to the capacity, such that the probability of error ML-decoder make goes to zero with increasing block length.
  - How to build codes close to the capacity?
  - How to encode/decode such codes?

Random codes are capacity-achieving, but then we need a big codebook.

Linear codes

Suppose that $A_1 = \mathbb{F}_2$. Then we could get efficient encoding by restricting to *linear codes*.

Linear codes

Suppose that $A_1 = \mathbb{F}_2$. Then we could get efficient encoding by restricting to *linear codes*.

- Linear code of block length n and dimension (or rank) k is a linear sub-space of dimension k of the vector space $\mathbb{F}_2^n$

Linear codes

Suppose that $A_1 = \mathbb{F}_2$. Then we could get efficient encoding by restricting to *linear codes*.

- Linear code of block length n and dimension (or rank) k is a linear sub-space of dimension k of the vector space $\mathbb{F}_2^n$
- The rate of such code is k/n

Linear codes

Suppose that $A_1 = \mathbb{F}_2$. Then we could get efficient encoding by restricting to *linear codes*.

- Linear code of block length n and dimension (or rank) k is a linear sub-space of dimension k of the vector space $\mathbb{F}_2^n$
- The rate of such code is k/n
- And then encoding is easy (if we find a basis of this subspace, we could just take a linear combination with coefficients $(x_1, \cdots, x_k)$).

Linear codes

Suppose that $A_1 = \mathbb{F}_2$. Then we could get efficient encoding by restricting to *linear codes*.

- Linear code of block length n and dimension (or rank) k is a linear sub-space of dimension k of the vector space $\mathbb{F}_2^n$
- The rate of such code is k/n
- And then encoding is easy (if we find a basis of this subspace, we could just take a linear combination with coefficients $(x_1, \cdots, x_k)$).

Also, *random linear codes* achieve capacity! Good news?

Linear codes

Suppose that $A_1 = \mathbb{F}_2$. Then we could get efficient encoding by restricting to *linear codes*.

- Linear code of block length n and dimension (or rank) k is a linear sub-space of dimension k of the vector space $\mathbb{F}_2^n$
- The rate of such code is k/n
- And then encoding is easy (if we find a basis of this subspace, we could just take a linear combination with coefficients $(x_1, \cdots, x_k)$).

Also, *random linear codes* achieve capacity! Good news? Well, ML-decoding is still hard.

Linear code example

A linear code could be defined by *parity check matrix* H:

$$H = \begin{pmatrix} 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 \end{pmatrix}$$

The codewords are all x such that $Hx = 0$
Each row is a parity check. E.g. the last one is

$$x_1 + x_4 + x_5 = 0$$

where summation is in $\mathbb{F}_2$.

How to build a generator matrix?

If H is full-rank, then we could decompose it to the echelon form by Gaussian elimination. Then it would have the form (up to column permutation):

$$\begin{bmatrix} I_{N-K} & P \end{bmatrix},$$

where I_{N-K} is the identity matrix. Then we could select G as

$$G = \begin{bmatrix} P \\ I_K \end{bmatrix}$$

Indeed, $HGx = (P + P)t = 0$. This is a *systematic* encoding as bits of the original message are copied in the known locations of the codeword. This makes the decoding (from the codeword) trivial.
Exercise: Let's build the generator matrix for H from the previous slide.

LDPC-codes

- Invented by Robert Gallager in his PhD thesis in 1963.
- Largely forgotten for about 30 years.
- Rediscovered by David J.C. MacKay and Radford Neal.

Robert Gallager (b. 1931)

LDPC codes decoding

Let's build the probabilistic model for the messages $P(x, y)$ (in, say, BSC case). For each bit x_i it gets inverted with probability p. So

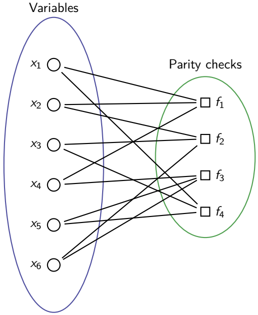
$$p(y|x) = \prod_{n=1}^{N} p(y_n|x_n) = \prod_{n=1}^{N} q^{y_n + x_n}(1-q)^{y_n + x_n + 1}$$

And all we know about x is that the probability of getting $P(Hx \neq 0) = 0$. So the model is:

$$p(x, y) \propto p(y|x) I[Hx = 0] = \prod_{n=1}^{N} p(y_n|x_n) \prod_{m=1}^{M} I[h_m^T x = 0]$$

We now try to estimate the marginals $p(x_n|y)$ and obtain the decoding $x_n^* = \arg\max p(x_n|y)$.

The graph



Factor-graph for LDPC-decoding. Not singly-connected, but we can still try to do belief propagation and "see what happens".

Message passing

We need to select a timetable. Let's send all the variable-to-factor messages first and then all factor-to-variable ones.

1. Initialisation: $\mu_{x_n \to f_m}(x_n) = p(y_n|x_n)$
2. Recomputing factor-to-variable messages:

$$\mu_{f_m \to x_n}(x_n) = \sum_{x_{n'}:n' \in N(m) \setminus n} I[x_n + \Sigma_{n'} x_{n'} = 0] \prod_{x_{n'}:n' \in N(m) \setminus n} \mu_{x_{n'} \to f_m}(x_{n'})$$

3. Recomputing variable-to-factor messages...

$$\mu_{x_n \to f_m} \propto p(y_n|x_n) \prod_{m' \in N(n) \setminus m} \mu_{f_{m'} \to x_n}(x_n)$$

4. ...and the marginals:

$$\hat{p}_n(x_n|y) \propto p(y_n|x_n) \prod_{m' \in N(n)} \mu_{f_{m'} \to x_n}(x_n)$$

Why low-density

Of course we could build this factor graph for *any* linear code. Why low-density?

- The running time of this algorithm depends on the number of edges in the graph.
- Moreover, it turns out short cycles in the graph are detrimental to performance.

This means that we want our matrix H to be sparse. Other design choices when selecting H are an active area of research.

A bag of tricks (advanced)

Step 2 in the algorithm could take a lot of time as we have to sum over all possible combination of neighbour states. Could we make it faster?

- Let $x_1, x_2, \ldots, x_{N(m)-1}$ be all the neighbours of m-th factor apart from x_n (just a renaming). Then consider the chain graph on the variables $s_1, s_2, \ldots, s_{N(m)-1}$, where $s_i = \sum_{j=1}^{i} x_j$. Then setting $p(s_1) = \mu_{x_1 \to f_m}(s_1)$ and $p(s_i|s_{i-1}) = \mu_{x_i \to f_m}(0)$, if $s_i = s_{i-1}$ Then $\mu_{f_m \to x_n}(x_n)$ is the same vector as marginal $p(s_{N(m)-1})$, which we could calculate efficiently.
- Let $\delta\mu_{x_n \to f_m} = \mu_{x_n \to f_m}(0) - \mu_{x_n \to f_m}(1)$ and $\delta\mu_{f_m \to x_n} = \mu_{f_m \to x_n}(0) - \mu_{f_m \to x_n}(1)$. It is possible to show that

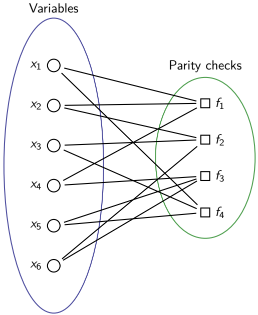
$$\delta\mu_{f_m \to x_n} = \prod_{n' \in N(m) \setminus n} \delta\mu_{x_{n'} \to f_m}$$

Then the messages could be recovered as

$$\mu_{f_m \to x_n}(0) = \frac{1}{2}(1 + \delta\mu_{f_m \to x_n}); \mu_{f_m \to x_n}(1) = \frac{1}{2}(1 - \delta\mu_{f_m \to x_n}).$$

A toy example for BEC

For BEC the algorithm becomes deterministic (in a way: the messages are either $(1, 0)$ or $(0, 1)$ or $(0.5, 0.5)$). Let's have a look...



Suppose we received this message: '1?1?01'. Could you decode it? What about '1?1?0?'?

References

Claude Shannon (1948)
A Mathematical Theory of Communication
Bell System Technical Journal 27, 379 – 423.

Robert Gallager (1963)
Low Density Parity-Check Codes
MIT Press, Cambridge, MA

Amin Shokrollahi (2004)
LDPC Codes: An Introduction
in *Coding, Cryptography and Combinatorics*, 85–110
Birkhäuser Basel