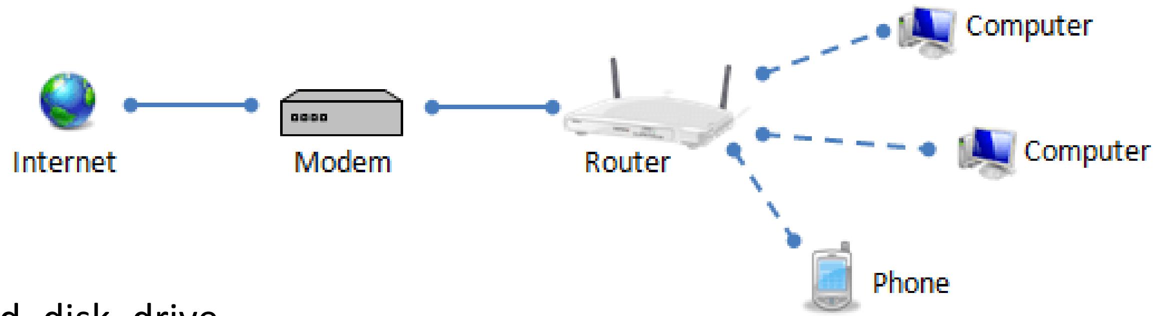
Error-correcting codes

Algorithm Interest Group
presentation by Eli Chertkov

http://www.computer-questions.com/what-to-do-when-error-code-8003-happens/

Society needs to communicate over noisy communication channels



https://en.wikipedia.org/wiki/Hard_disk_drive

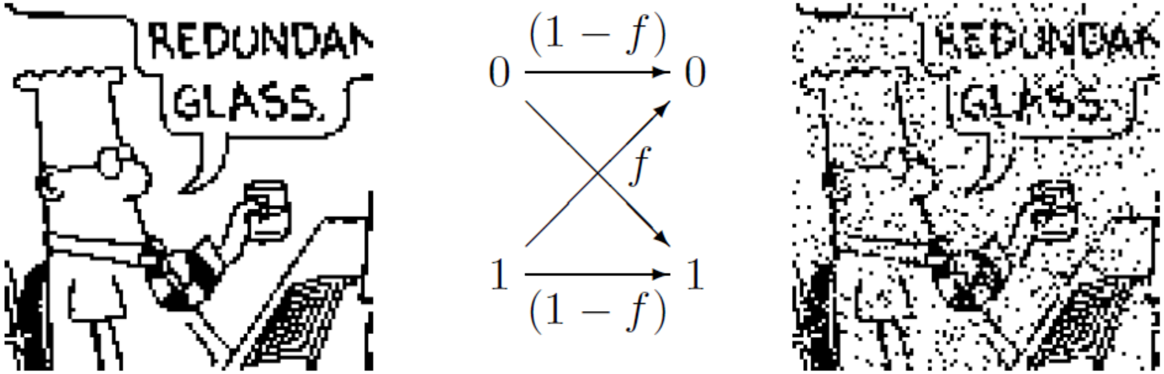
http://www.diffen.com/difference/Modem_vs_Router

https://en.wikipedia.org/wiki/Cell_site

https://www.nasa.gov/sites/default/files/tdrs_relay.jpg

Noisy bits

We will visualize noise in data through random flipping of pixels in a black and white image.

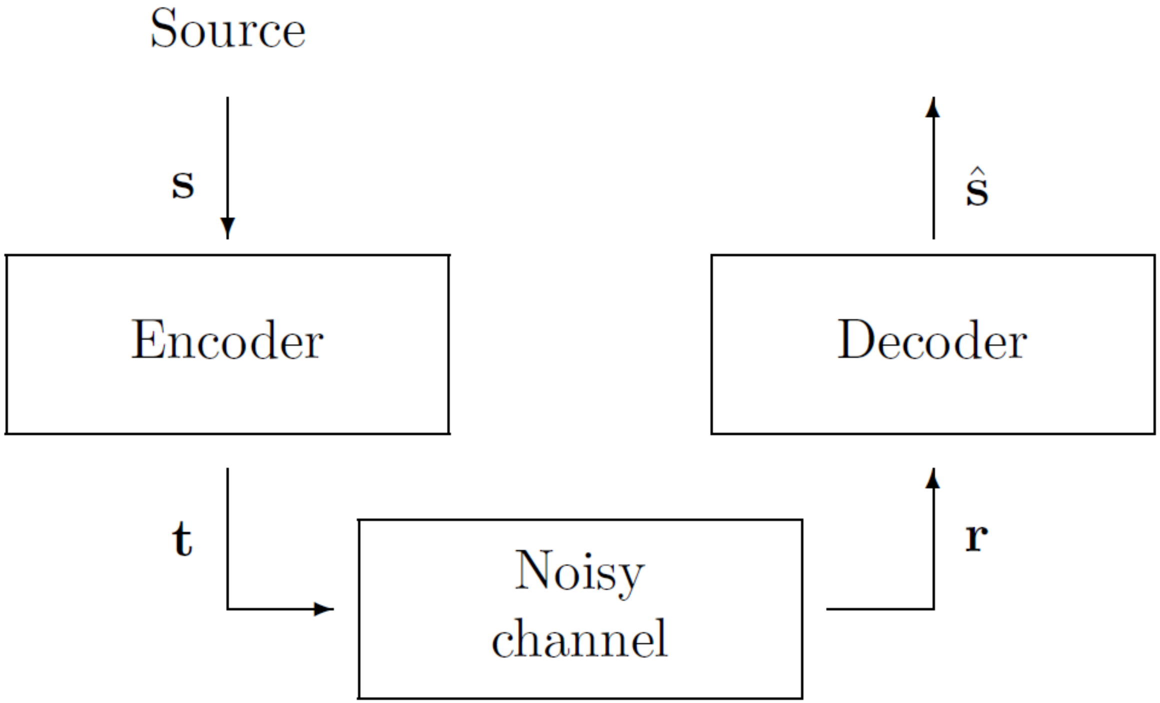


$f =$ probability of flipping a bit from 0 to 1 or vice versa

$1 - f =$ probability of a bit staying the same

Noisy channel coding

To minimize the noise picked up by source data $\boldsymbol{s}$ as it passes through a noisy channel, we can convert the data into a redundant signal $\boldsymbol{t}$.



Example: Repetition codes

The simplest encoding one can think of is repetition coding R_n: repeat each bit n times.

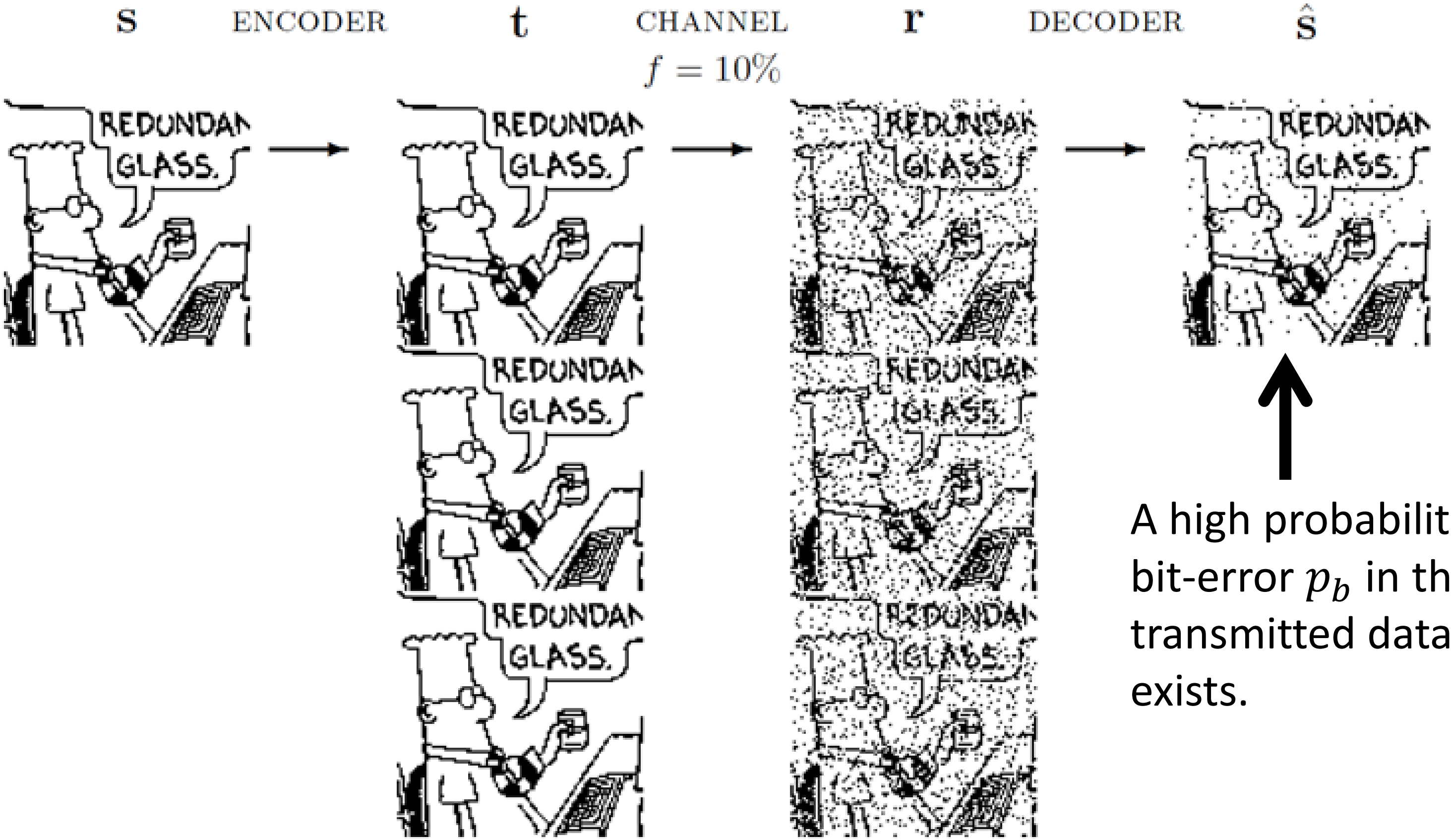
Encoding 0101 $\to^{R_5}$ 00000 11111 00000 11111

Noise from channel 01100 01101 00000 10001

The optimal decoding of a repetition code is to take the majority vote of each n bits.

Decoding 01100 01101 00000 10001 $\to^{R_5}$ 0100

Repetition code visualization

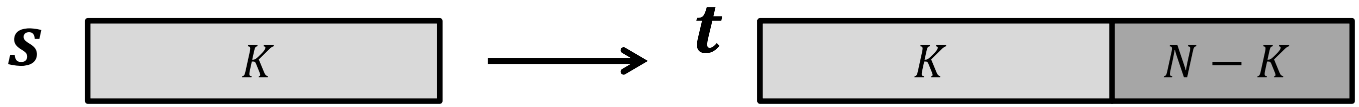


A high probability of bit-error p_b in the transmitted data still exists.

Easy to see and understand how it works, but not a useful code.

Example: Linear block codes

A linear length N block code adds redundancy to a length $K < N$ sequence of source bits.



The extra $K - N$ bits are called *parity-check bits*, which are linear combinations of the source bits mod 2.

(7,4) Hamming code example

$$\boldsymbol{t} = \boldsymbol{G}^T \boldsymbol{s}$$

$$\boldsymbol{t} = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 1 \\ 1 \\ 0 \\ 1 \end{pmatrix} \quad \boldsymbol{G}^T = \begin{pmatrix} 1 & & & \\ & 1 & & \\ & & 1 & \\ & & & 1 \\ 1 & 1 & 1 & \\ & 1 & 1 & 1 \\ 1 & & 1 & 1 \end{pmatrix} \boldsymbol{s} = \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \end{pmatrix}$$

More about linear block codes

Linear block codes are a large family of error-correcting codes, which include:

Reed-Solomon codes, Hamming codes, Hadamard codes, Expander codes, Golay codes, Reed-Muller codes, …

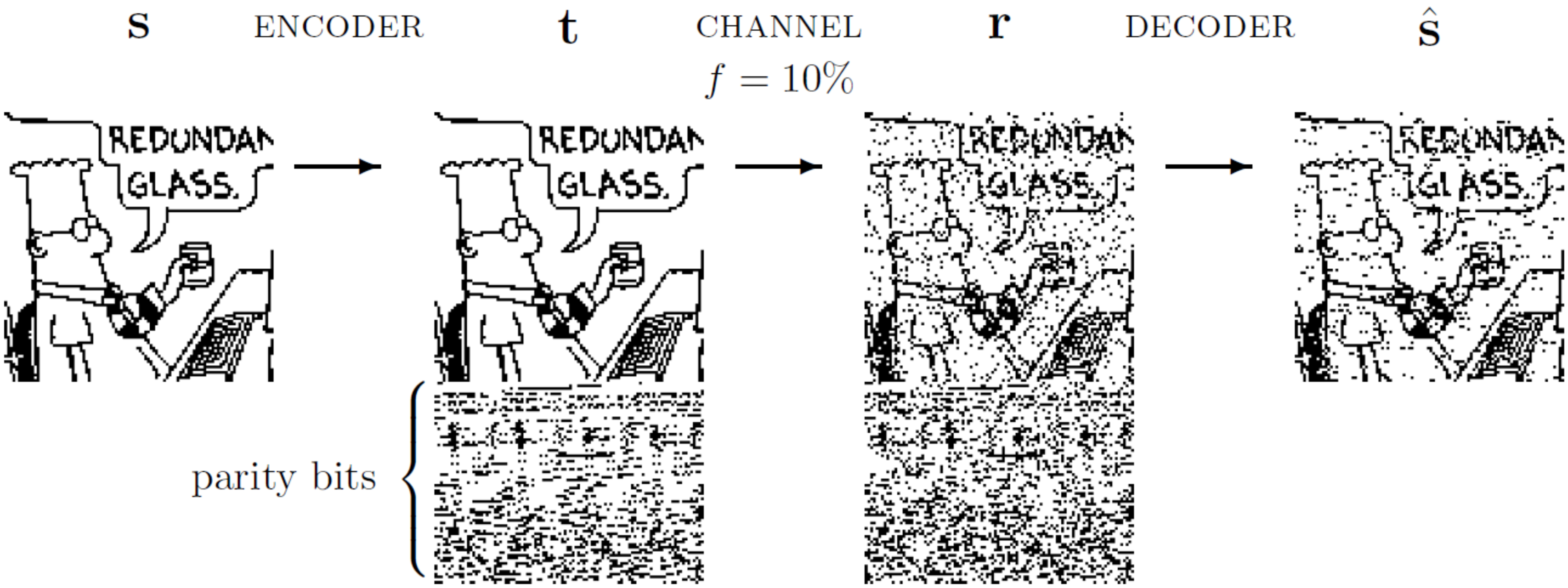
They differ by the linear transformation from $\boldsymbol{s}$ to $\boldsymbol{t}$.

The rate of a block code is $R = \frac{K}{N} = \frac{\boldsymbol{message\ size}}{\boldsymbol{block\ size}}$

Decoding can become tricky for these codes, and is unique to the specific type of code used.

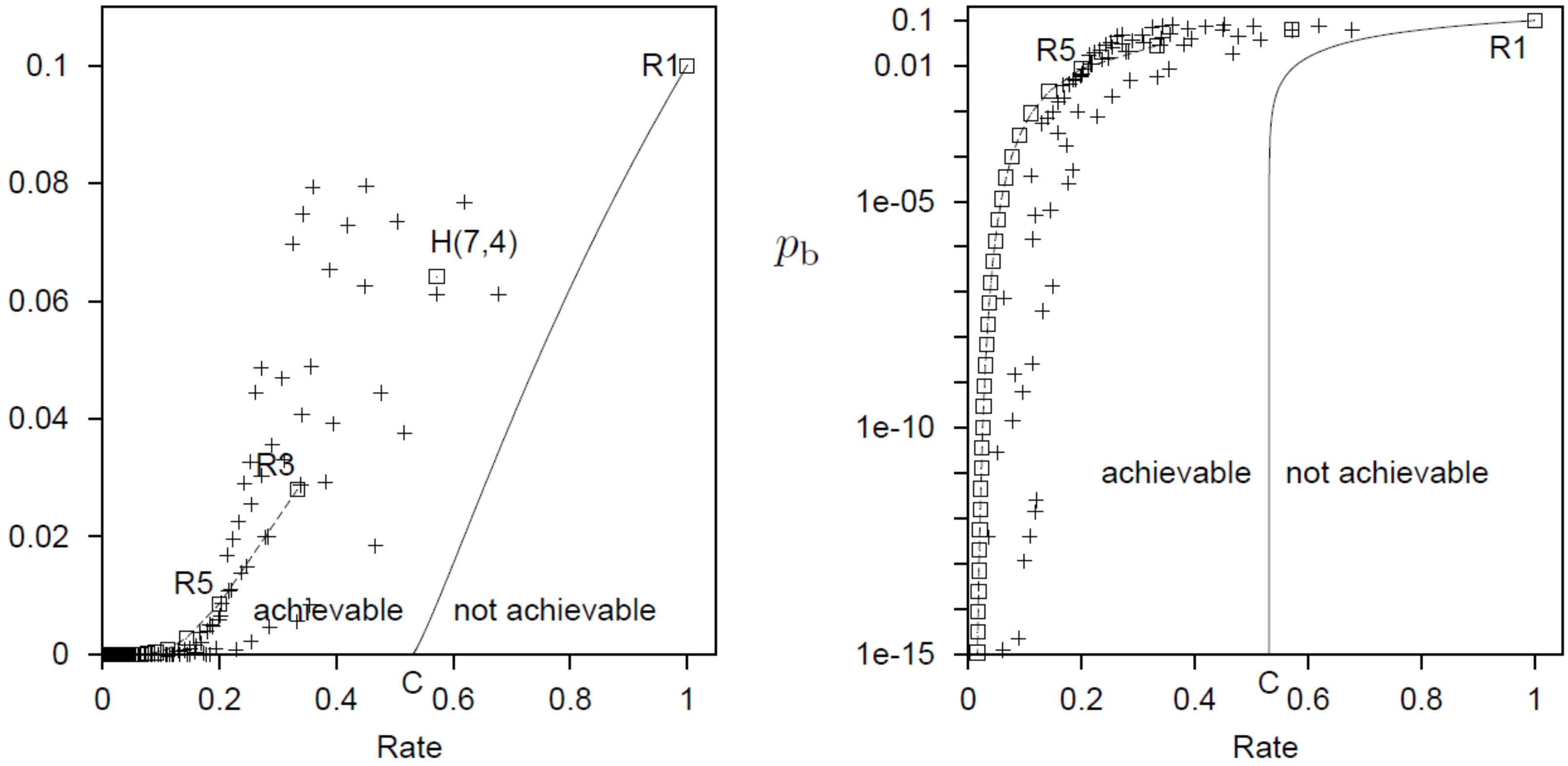
Hamming codes, for instance, are nice because there is a simple and visual way, using Hamming distances, to optimally decode.

Linear block code visualization



There is less redundancy in the error-coding ($\boldsymbol{s} \to \boldsymbol{t}$) compared to repetition coding, but the probability of error scales the same as repetition coding $p_b = O(f^2)$.

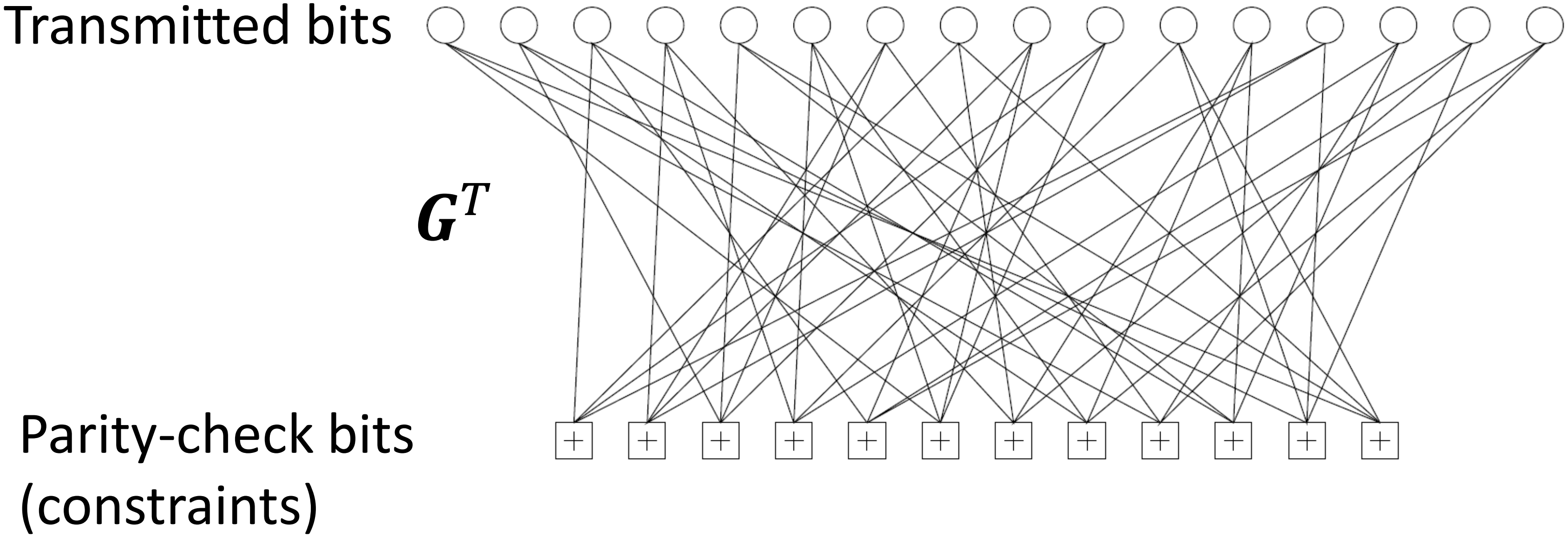
Shannon’s noisy-channel coding theorem



In 1948, Claude Shannon showed that 1) there is a boundary between achievable and not achievable codes in the (R, p_b) plane and that 2) codes can exist where R does not vanish when the error probability p_b goes to zero.

Note: This does not mean that codes near the boundary can be efficiently decoded!

Sparse graph codes



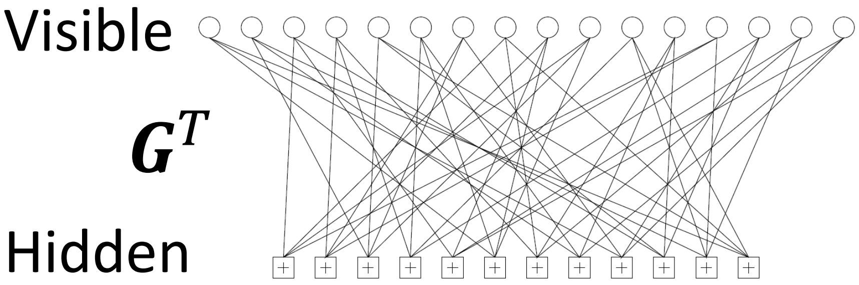
A **low-density parity check code** (or Gallager code) is a randomly generated linear block code represented by a sparse bipartite graph (sparse $\boldsymbol{G}^T$).

Another example of a useful sparse graph code is a turbo code.

Belief Propagation

It is in general an NP-complete problem to decode low-density parity check codes.

However, a practically efficient approximate method exists, called Belief Propagation (BP) or the Sum-Product algorithm.



It is a message passing algorithm that solves an inference problem on a probabilistic graphical model

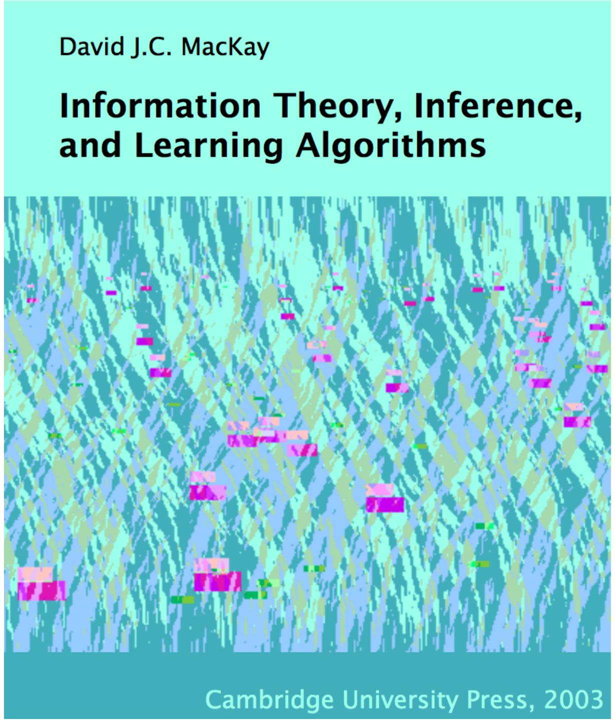
BP is a physics-inspired algorithm. It casts a probability distribution represented by a graph in terms of a Boltzmann distribution. Then it attempts to find the fixed point of the Free Energy under the Bethe approximation. It is exact for graphical models, which are trees.

Details can wait for another talk…

References

- Awesome resource (especially for physicists):

  Information Theory, Inference, and Learning Algorithms by David MacKay.

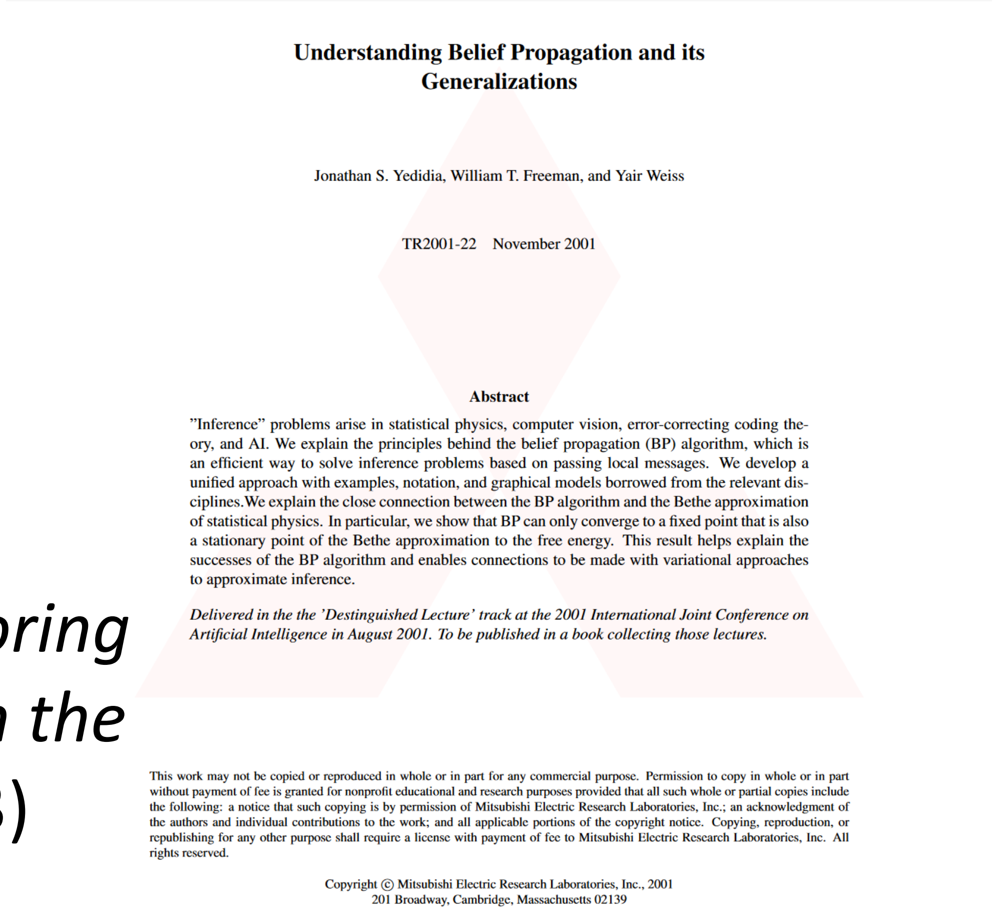

David J.C. MacKay
Information Theory, Inference, and Learning Algorithms
Cambridge University Press, 2003

(Basically the whole presentation is based off of the material in this book.)

References (continued)

- Resource on Belief Propagation:

  Yedidia, J.S.; Freeman, W.T.; Weiss, Y., “Understanding Belief Propagation and Its Generalizations”, *Exploring Artificial Intelligence in the New Millennium* (2003) Chap. 8, pp. 239-269.



Understanding Belief Propagation and its Generalizations

Jonathan S. Yedidia, William T. Freeman, and Yair Weiss

TR2001-22 November 2001

Abstract

"Inference" problems arise in statistical physics, computer vision, error-correcting coding theory, and AI. We explain the principles behind the belief propagation (BP) algorithm, which is an efficient way to solve inference problems based on passing local messages. We develop a unified approach with examples, notation, and graphical models borrowed from the relevant disciplines.We explain the close connection between the BP algorithm and the Bethe approximation of statistical physics. In particular, we show that BP can only converge to a fixed point that is also a stationary point of the Bethe approximation to the free energy. This result helps explain the successes of the BP algorithm and enables connections to be made with variational approaches to approximate inference.

Delivered in the the 'Destinguished Lecture' track at the 2001 International Joint Conference on Artificial Intelligence in August 2001. To be published in a book collecting those lectures.