ECOLE POLYTECHNIQUE FEDERALE DE LAUSANNE
School of Computer and Communication Sciences

Handout 11-2 Statistical Physics for Communication and Computer Science
Final Project May 13th, 2015, Due Date: June 22th

Problem 1 (Source Coding)**.** In this project you will consider the problem of lossy source coding. This problem is quite similar to the K-SAT problem.

Consider a binary symmetric source, emitting i.i.d. Bernoulli random variables X_i, $i = 1, \ldots, n$, with parameter $\frac{1}{2}$. We write X_i if we think that the source takes values in $\{0, 1\}$ and S_i if we think of the corresponding spin model, with $S_i \in \{\pm 1\}$.

We want to compress this source but keep the distortion as small as possible. More precisely, for $\underline{x}, \underline{y} \in \{0, 1\}^n$, let $d_H(\underline{x}, \underline{y}) = \sum_{i=1}^n \mathbf{1}_{\{x_i \neq y_i\}}$ denote the Hamming distortion. Let $f : \{0, 1\}^n \mapsto \{0, 1\}^k$, $1 \leq k \leq n$, be the encoding function and $g : \{0, 1\}^k \mapsto \{0, 1\}^n$ be the decoding function. This means that the encoding function f compresses the n bits into k bits, and the decoding function g takes the k bits and reconstructs a codeword of length n. As a measure of faith fullness of the reconstruction we pick

$$\mathbb{E}[\frac{1}{n} d_H(g(f(\underline{X})), \underline{X})],$$

i.e., we pick the *average* distortion where the average is over all source outcomes. Our aim is to minimize this distortion for a fixed k/n. More precisely, if we fix $R = k/n$, the *rate* of the compression, and if we let n tend to infinity, then we want to minimize the average distortion, call it $D(R)$. Equivalently, we want to minimize the rate R as a function of D.

If you took the information theory class you know that for the above case this rate-distortion function (the best possible trade-off) is given by

$$R(D) = \begin{cases} 1 - h_2(D), & 0 \leq D \leq \frac{1}{2}, \\ 0, & D \geq \frac{1}{2}. \end{cases}$$

In words, $1 - h_2(D)$ is the smallest rate which we can have (regardless of the algorithm we use) if we want to represent the source with average distortion at most D.

Our aim is to construct and analyze a practical lossy source compression scheme using low-density generator-matrix (LDGM) ensembles and message-passing guided decimation (similar to what was used for the K-SAT problem). We do not aim to get close to the optimal rate-distortion curve. As for standard channel coding, this could be done by optimization over the ensemble. We skip this part, since we are interested mainly in the basic analysis.

- We will use LDGM ensembles. Such an ensemble is defined as follows. We start with k code bits, $k < n$, call them $Y_1, Y_2, \ldots, Y_k$. These are the bits which represent the source in compressed form.

  We expand those into n bits, call them $\hat{X}_1, \hat{X}_2, \ldots, \hat{X}_n$. These n bits will be our reconstruction of the source word $X_1, X_2, \ldots, X_n$.

  The relationship between these two is given by a bipartite graph. We have k code bit nodes (for $Y_1, \ldots, Y_k$), n *generator* nodes, and n source reconstruction nodes (for $\hat{X}_1, \ldots, \hat{X}_n$). In the (l, r)-regular case each code bit node has degree r, each source reconstruction node has degree 1, and each generator node has degree $l + 1$.

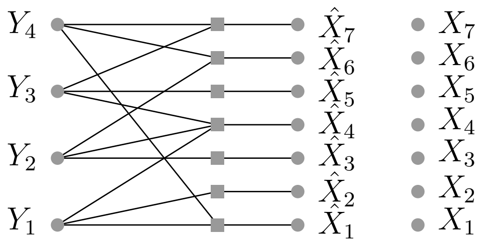

Figure 1: The factor graph associated to the source coding problem.

Further, each generator node is connected to l code bit nodes and exactly one source reconstruction node. This means that each source reconstruction variable is the XOR of l code bits: $Y_{j1} + \dots Y_{jl} = \hat{X}_j$, $j = 1, \dots n$. We write the relationship as $\underline{\hat{X}} = g(\underline{Y})$. The corresponding factor graph is shown in Fig. 1.

Decoding: The decoding part is easy. Take the k code bits $\underline{Y}$ and generate from these the n code reconstruction bits $\underline{\hat{X}}$. In terms of our map, $\underline{\hat{X}} = g(\underline{Y})$.

Encoding: Assume we are given the source word $\underline{X}$ of length n. Our aim is to represent it as faithfully as possible by a reconstruction word $\underline{\hat{X}}$ by setting the k code bits $\underline{Y}$ to appropriate values. In principle we can try out all 2^k possible values of $\underline{Y}$ and pick one of those which minimize the distortion. I.e., we define the encoding function as

$$\begin{aligned} f(\underline{X}) &= \text{argmin}_{\underline{Y}}\{d_H(g(\underline{Y}), \underline{X})\} \\ &= \text{argmin}_{\underline{Y}}\Big\{\sum_{j=1}^{n} \mathbf{1}_{\{Y_{j1}+\cdots+Y_{jl} \neq X_j\}}\Big\}. \end{aligned}$$

Message-Passing Guided Decimation: We want to compute the following expression

$$\mu_i(Y_i) = \min_{\underline{Y}_{\sim i}} \sum_{j=1}^{n} \mathbf{1}_{\{Y_{j1}+\cdots+Y_{jl} \neq X_j\}}.$$

Note, here Y_{jk}, $k = 1, \dots, l$, are the l code bits which the j-th generator node is attached to. The function $\mu_i(Y_i)$ has the following interpretation. When we set $Y_i = 0$ then $\mu_i(Y_i)$ gives us the minimum distortion which we can achieve by picking the remaining $k-1$ code bits and when Y_i is frozen to 0. The interpretation for $\mu_i(Y_i = 1)$ is similar.

The above function is the minimization over a function which consists of many summands. This is in analogy to our usual sum-product set-up where we sum (marginalize) a function which is highly factorized. Formally, these two problems can be translated into each other if we replace multiplication by summation and summation (marginalization) by minimization. The resulting message-passing rules are identical modulo this mapping. The resulting algorithm is called the *min-sum* algorithm for obvious reasons (our original algorithm was called *sum-product* algorithm). As for the sum-product algorithm, the min-sum algorithm is optimal on trees.

We proceed now as follows. We run the min-sum algorithm. After a certain number of iterations we assume that the messages have converged. Pick a variable which has a difference $\mu_i(Y_i = 0) - \mu_i(Y_i = 1) \neq 0$. Set this variable to that value of Y_i which gives the smaller value of μ_i. Propagate this value of Y_i to the generator nodes and update the X_j. Then expurgate the graph.

If you implement this algorithm you will see that at the very beginning the algorithm does not get started. All messages stay trivial. Can you explain why this happens? To avoid this problem. Whenever the messages stay trivial, simply pick a code bit at random and set it to a random value. Update the system and then expurgate the graph and start anew.

Simulation: Use message passing guided decimation as explained above and run simulations for the $(3, 6)$ ensemble for $n = 10^4$. What is the average distortion? How does this compare to the optimal trade-off?

Analysis:

(i) Write down the expression for the Bethe ground state energy. This is supposed to approximate the minimal average distortion, and would be exact on a tree. Note, similarly to the sum-product algorithm you get this expression - from the usual Bethe free energy - by formally mapping multiplication to summation and log-summation to minimization.

(ii) Use the population dynamics approach to compute density evolution. (Again, you might need a small modification at the beginning to get started. E.g., you might want to try a random initialization.)

(iii) Plug in the density which you get from the population dynamics approach into the expression for the Bethe free energy. This gives you an expression which is supposedly the distortion of this system (if you had used an optimal algorithm). What number do you get? Does it make sense?

You are expected to hand in a small report before the due date.