# Belief Propagation

Reminder: No Lecture on May 6

Belief propagation is a widely used heuristic for decoding high-rate classical LDPC codes. It is not so useful for the surface code, so a few years ago I would not have considered including it in lectures on QEC. But with the advent of high-rate qLDPC CSS codes BP is frequently employed in QEC now, so is worthwhile to discuss. We'll consider a classical linear code, but you can think of it as the X or Z syndrome decoding of a quantum CSS code.

$H$ is $(n-k)\times n$ parity check matrix. The error $e$ is $n$-component binary vector, indicating which bits have flipped. $He = s$ where the syndrome $s$ is $(n-k)$ components. We observe $s$ and want to infer $e$, but there are $2^k$ possibilities. Among these, we wish to find the most likely $e$, as determined by our noise model, which assigns probability $P(e)$ to error $e$.

According to Bayes: $P(e|s) = \dfrac{P(e)P(s|e)}{P(s)}$

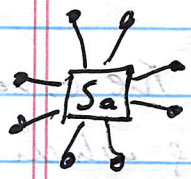
In Maximum Likelihood Decoding (MLD) we seek the Maximum Likelihood Estimate (MLE) of $e$ given $s$:

$$e^* = \operatorname{argmax} P(e|s)$$

We need $P(s)$ in the denominator for normalization, but when $s$ is known all the dependence on $e$ is in the numerator, so we are looking for $e$ that maximizes $P(e)P(s|e)$. Here $P(e)$ comes from the noise model, but $s$ is a deterministic function of $e$ — that is, for each $e$, one particular $s$ occurs with probability 1.

Consider the code's Tanner graph. Check nodes are labeled $a, b, c, \ldots$, and variable nodes are labeled $i, j, k, \ldots$. We write $P(s|e)$ as a product of deterministic factors, one for each of the $n-K$ checks

$$P(s|e) = \prod_{a \in C} \phi_a(e_{\partial a}, s_a).$$




Here $e_{\partial a}$ denotes the components of $e$ that are neighbors of $a$ in the Tanner graph (its "boundary"), and $\phi_a = 1$ if the parity of these neighbors matches the observed syndrome bit $s_a$. Otherwise, it is zero. So $\prod_a \phi_a = 1$ if $He = s$ and is zero otherwise.

Let's suppose independent noise, so $P(e)$ factorizes, but not necessarily identical on all bits

$$P(e) = \prod_{i \in V} \psi_i(e_i).$$

Hence $\psi_i(e_i)$ is the prior distribution, which should be updated after we observe $s$.

We've looking for the error $e$ that maximizes

$$P(e|s) = \frac{1}{Z(s)} \prod_{i \in V} \psi_i(e_i) \prod_{a \in C} \phi_a(e_{\partial a}, s_a)$$

(where we write $P(s) = Z(s)$ to emphasize the stat mech analogy). We are trying to minimize the free energy of a spin-glass model, in the Gibbs distribution. The $\{e_i\}$ are the binary spins — they are warm, with $\psi_i(e_i)$ the Boltzmann factor for a spin in an external field. The $\{\phi_a\}$ are the interaction terms — they are cold, i.e., the interaction energy is $\gg$ temp. The trouble is that for a general Tanner graph and a typical $s$, the free energy $-\log P(e|s)$ is a nonconvex function which is NP-hard to minimize.

Given $s$, the distribution $P(e|s)$ viewed globally, may have complex correlations among the bits that are difficult to characterize. In BP we set the more modest goal of estimating marginal distributions for each variable

$$P_i(e_i|s) = \sum_{e \setminus e_i} P(e|s).$$

Then we'll guess ~~no~~ error occurred if

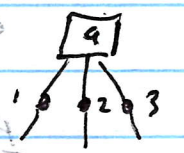
$$\frac{P_i(e_i=0|s)}{P_i(e_i=1|s)} > 1 \quad \text{or log-likelihood ratio}$$

$$LLR = \ln\left(\frac{P_i(0)}{P_i(1)}\right) > 0$$

4 May 2026 (4)

In BP, each variable has a belief about its own marginal; each check has beliefs about the marginals of all its neighboring variables, and in addition the knowledge that the check is satisfied. It does not claim to know anything else. BP is an iterative message-passing algorithm on the Tanner graph in which in one round variables send messages about their beliefs to their neighboring checks, and in the following round checks send messages about their beliefs to their neighboring variables. Beliefs are updated in each round using the information just obtained, and the procedure is iterated many time until beliefs converge and no new updates are occuring. The goal is to reach an equilibrium in which the beliefs of checks and variables are consistent. We initialize the procedure by having variables believe their marginals are given by the prior $\psi_i(e_i)$.

What should a check tell a variable? Suppose, for example, that check a has degree 3, i.e. computes parity of 3 variables (it's a row of H with weight 3). In a previous round, a received estimates of $p_i(e_i)$ for each variable. It passes what it knows to, say, variable 1, assuming (perhaps incorrectly) that variables 2 and 3 are independent.



check a estimates, assuming $s_a = 0$

$$P_1^{new}(0) = P_2^{old}(0) P_3^{old}(0) + P_2^{old}(1) P_3^{old}(1)$$

$$P_1^{new}(1) = P_2^{old}(0) P_3^{old}(1) + P_2^{old}(0) P_3^{old}(1)$$

If on the other hand $s_a = 1$, then the estimate is flipped: $P_1^{new}(0) \leftrightarrow P_1^{new}(1)$

It is convenient to reformulate this "sum-product" formula for updated probability in terms of "bias" $P(0) - P(1)$.

$$\underbrace{P_1^{new}(0) - P_1^{new}(1)}_{\text{new bias}} = (-1)^{s_a} \left(\underbrace{P_2^{old}(0) - P_2^{old}(1)}_{\text{old bias}}\right)\left(\underbrace{P_3^{old}(0) - P_3^{old}(1)}_{\text{old bias}}\right)$$

Expanding the product of binomials recovers the previous formula, and $(-1)^{s_a}$ flips the bias if $s_a = 1$.

A further convenience is using "log-likelihood ratios" (LLRs), which frees us from explicitly renormalizing probabilities, and transforms multiplication into addition.

$$L = \ln\left(\frac{P(0)}{P(1)}\right) \Rightarrow$$

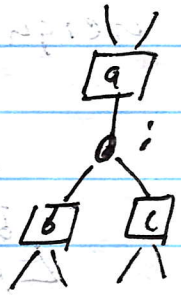
$$P(0) - P(1) = \frac{P(0)/P(1) - 1}{P(0)/P(1) + 1} = \frac{e^L - 1}{e^L + 1} = \frac{e^{L/2} - e^{-L/2}}{e^{L/2} + e^{-L/2}} = \tanh\left(\frac{L}{2}\right)$$

So the bias update rule becomes

$$\tanh\left(\frac{L_1^{new}}{2}\right) = (-1)^{s_a} \tanh\left(\frac{L_2^{old}}{2}\right) \tanh\left(\frac{L_3^{old}}{2}\right)$$

This $L_1^{new}$ is reported by the check to variable 1. Similarly, the check computes $L_2^{new}$ and $L_3^{new}$ and reports to the corresponding variable nodes.

How does the variable use the information it receives from the checks? It assumes (perhaps incorrectly) that the neighboring checks are independent witnesses, and uses the new information to update its prior. It further assumes that each witness is independent of the prior distribution $P_i(e_i)$ for variable $i$. Under this local independence assumption, the variable multiplies probabilities (up to normalization) to update the prior




$$P_i^{new}(e_i) \propto P_i^{prior}(e_i) P_i^{old,a}(e_i) P_i^{old,b}(e_i) P_i^{old,c}$$

In terms of log-likelihood ratio

$$L_i^{new} = \ln \frac{P_i^{new}(0)}{P_i^{new}(1)} = \ln \left( \frac{P_i^{prior}(0) \cdot P_i^{old,a}(0) \cdot P_i^{old,b}(0) \cdot P_i^{old,c}(0)}{\text{same } (0 \to 1)} \right)$$

$$= L_i^{prior} + L_i^{old,a} + L_i^{old,b} + L_i^{old,c}$$

This is variable $i$'s updated belief about its own LLR.

In the message passing protocol, we denote by $L_{a \to i}$ the message check $a$ sends to variable $i$ regarding check $a$'s belief about variable $i$, updated using the sum-product rule. As above, this message satisfies

$$\tanh\left(\frac{L_{a\to i}^{new}}{2}\right) = (-1)^{s_a} \prod_{j \in a \setminus i} \tanh\left(\frac{L_{i\to a}^{old}}{2}\right)$$

Importantly, using the local independence assumption, this update is based on information check a received in the previous round from all of its neighbors except for variable i.

Variable i's updated belief about itself is

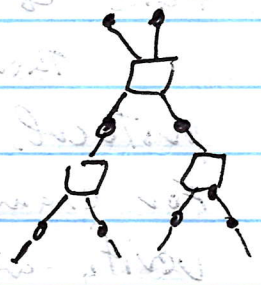
$$L_i^{belief,new} = L_i^{prior} + \sum_{b \in i} L_{b\to i}^{old}$$

Importantly, vertex i does not report this belief to its neighboring check a. This belief is based in part on information i received from a, and so would violate the local independence assumption. Instead i leaves this information out of its message to a, and instead sends

$$L_{i\to a}^{new} = L_i^{prior} + \sum_{b \in i \setminus a} L_{b\to i}^{old}$$

These rules define the BP message-passing protocol on the Tanner graph, which we iterate multiple times. If we are lucky, messages tilt toward $\tanh L/2 \to 1$ (strong bias favoring 0) or $\tanh L/2 \to -1$ (strong bias favoring 1), so we are confident in inferring e from these estimated marginal distributions.

If the Tanner graph is a tree graph, then we can apply the BP rules starting with prior beliefs about variables on the leaves. Information propagates only inward, away from the leaves. (Variables don't send info received from a check back to the same check.)



Also the local independence assumption is true, as information is aggregated from disjoint subtrees. Hence BP converges to the correct marginals in time scaling like the graph's diameter (the maximum distance between leaves).

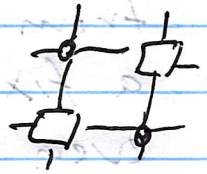
Also true, but less obvious: the marginals descended from the global joint distribution $P(v|s)$ yield a fixed point of BP (updates don't change beliefs), even though local independence does not hold. It's because the rule a check uses to update marginals is just a local version of computing marginals starting from the global distribution. This gives us hope that if we iterate enough times, marginals converge to the correct value.

The trouble is that, on a graphs with loops (especially short loops) local independence can be badly violated, because information that flows away from a node can propagate back to the same node after traveling around a loop.

As a result, there can be fixed points of BP other than the correct marginals, or BP may fail to converge at all.

Therefore, BP is typically augmented by another protocol, such as "ordered statistics decoding" (OSD). For example, if BP does converge, we can verify whether the fixed point $e^*$ we found really satisfies $He^* = s$. If not, order the bits $e_i^*$ according to how close $e_i^*$ & LLR $L_i^*$ is to $\tanh(\frac{L_i^*}{2}) = \pm 1$. We accept the $e_i^*$ in which we have the most confidence, and use Gaussian elimination to solve for the rest.

Other decoders outperform BP for e.g. the surface code because its Tanner graph contains short loops (length = 4).



But for Tanner graphs of high-rate codes like HGP codes from double-expanders, loops are rather large ("graph has a large "girth.") so BP + OSD is pretty effective.

We assumed here that the syndrome $s$ is accurate, but what if it's not because there are syndrome measurement errors? We might have to repeat syndrome measurements to improve reliability. But LDPC codes defined by Tanner graph with sufficient expansion have the "single-shot" property, meaning that a syndrome measured just once can be decoded reliably.

The idea is that can decode a syndrome using a greedy algorithm, a sequence of "small-set flips" that reduce the weight of the syndrome step-by-step. For an ideal syndrome, this eventually reduces the weight to zero, identifying a candidate e with a small prob of a logical error. Even if the syndrome has noise, because each decoding step is small and uses only local information on the Tanner graph, steps based on faulty information don't cause a large deviation between the true and apparent error, and with high prob we find an approximate codeword that is close to the correct codeword.

If we do need to measure the syndrome multiple times, we can apply BP to the syndrome history graph. As for our discussion of the surface code, bit errors occur on "horizontal" or "spacelike" edges of this graph and measurement errors occur on "vertical" or "timelike" edges. The check nodes identify where a change occurs in the syndrome between round $t$ and $t+1$. The change can occur because a data bit flipped, or a syndrome error occured. In BP, messages are passed in both timelike and spacelike directions, seeking ~~of~~ locally consistent ~~prob dist~~ marginals for data + syndrome errors.