Belief Propagation on Dense Graphs with few Strong Links

Etienne Mallard

MSc by Research in Pattern Analysis and Neural Networks

ASTON UNIVERSITY

September 2005

ASTON UNIVERSITY

Belief Propagation on Dense Graphs with few Strong Links

ETIENNE MALLARD

MSc by Research in Pattern Analysis and Neural Networks, 2005

Thesis Summary

Message passing techniques in general and belief propagation in particular are highly useful tools for inference, especially when the problem can be represented by a sparse graph. They have been used to solve a range of computational problems such as decoding, graph colouring, satisfiability problems etc.

In this approach, messages containing probabilistic information about nodes in the graph are being passed from node (variable) to node to facilitate the calculation of joint and conditional probabilities of the variables, which are then used for inferring their most likely value. One of the great advantages of message passing techniques on sparse graphs is that they are distributed and scale well with the system size (i.e., the computational effort involved grows linearly with the system size).

Extending the problem to densely connected systems where the number of messages is very high has been recently suggested. This approach is based on looking at macroscopic properties of sums of messages and modifying them locally (e.g., they can be represented by a Gaussian distribution of some mean and variance which could be modified locally).

The task in the project is to devise an algorithm for carrying out updates in a case where sparse strong links exist in conjunction with dense weak links, using features of both algorithms.

Keywords: Belief propagation, Sparse graph, Dense graph

Acknowledgements

I would like to thank all the people who made this year

- all the NCRG staff and our teachers who taught us an interesting mathematics field
- my supervisor David Saad who helped me and took lot of time to explain again and again techniques and concepts
- all the french team *Big Chicken, P0il P0il P0il, LetIsGo and Ludo*
- Amy Louise Rostron with who we had so much fun
- all the PHD (Thomas, Remi, Boremi and Stephane) for our talks and for playing football and badminton
- papa and momon who allowed me to have this fantastic year.

And finally thanks everyone for spending time to repeat your sentences due to my bad comprehension.

Contents

List of Figures

Chapter 1

Inference and Belief Propagation

1.1 Inference, the Bayesian approach

Inference is the process of predicting an observation using some facts that are already known. For example, if you see through the window that it's raining, it's quite natural to infer that the sky is grey. The process of inference was already studied by the Greeks philosophers who defined inference rules to help people to prove a deduction. The method called "syllogism" is a set of three inferences, the most famous of which is:

All men are mortal
Socrates is a man
Therefore Socrates is mortal.

This kind of reasoning was later formally enunciated by logicians, who created quantifiers like:

$$\begin{aligned} \text{the negation of X}: &\quad \neg X \\ \text{for all X}: &\quad \forall X \\ \text{it exists an X}: &\quad \exists X \end{aligned}$$

Some softwares like the programming language Prolog has been created to automatically find if a given fact can be inferred provided a set of propositions and rules.

Some scientists prefer to follow another principled way for probabilistic inference which is called the Bayesian framework [1], [2]. It uses rules about probabilities to find the best result, prediction or explanation to a problem. According to them the probability is identified as the belief of a proposition. When this proposition is certainly true, its probability is 1, and on the contrary, when it is certainly false, the proposition is assigned the probability 0. The way to find the best answer to a problem is to select the one which has the highest probability among all possible answers. The name of

that field of mathematics was given by a central rule called the Bayes theorem which enables one to define the probability of an event A given an event B as a function of the probability of B given A:

Theorem 1.1.1 Bayes Rule

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

1.2 Graph representation and message passing

Message passing techniques are used to solve problems involving variables which are linked by a set of constraints. The goal of such problems is to find the values of variables compatible with all constraints and possibly with some probabilistic bias. If all of them cannot be respected, the number of violated constraints tries to be minimised. This can also be referred to as *inferring the most probable value of the set of variables.*

1.2.1 Graph representation

Because there could be many dependencies between a given set of events, it could be a good idea to summarise them by a graph. This representation simplifies the probabilistic dependencies and makes them easy to group and identify. Each node represents a variable, and a link is created between the variables linked by a constraint (Figure 1.1). That representation by a graph is very useful when the number of dependencies is quite high.

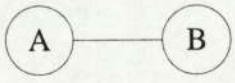

Figure 1.1: A graphical representation of a probabilistic dependence between two variables.

1.2.2 Intuitive approach

The boolean satisfiability problem (SAT) is a decision problem considered in complexity theory. The question is: given the expression, is there some assignment of TRUE and FALSE values to the variables that will make the entire expression true? For example the following expression can be considered:

$$E = (b_1 \text{ or } \neg b_3 \text{ or } \neg b_4) \text{ and } (\neg b_2 \text{ or } b_3 \text{ or } b_4),$$

where E has two clauses and four literals (b_1, b_2, b_3, b_4) and { b_1=TRUE, b_2=FALSE, b_3=TRUE, b_4=FALSE } is one solution of the problem. Figure 1.2 shows how to model this problem by a graph.

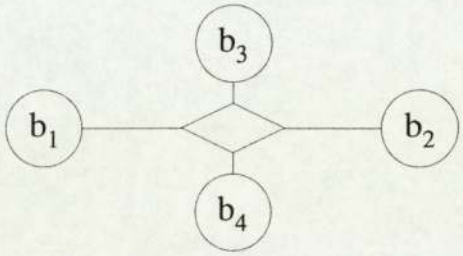

Figure 1.2: SAT problem modelled by a graph where circles represent the variables and edges a constraint between two variables

Once the problem has been modelled by a graph, a message passing technique may be used for infering the most probable solution, consisting in exchanging messages of probabilistic nature along the links. The problems described above are NP-complete [3] and the algorithms to find a solution are generally computationaly very expensive. Message passing provides a solution by looking at constraints locally and tries to converges through iterations.

Message can be considered as a degree of belief given by all the nodes linked by a constraint. The neighbours of a node send to it the probability that it can take a value. An intuitive explanation is given below with the satisfiability problem.

Let us assume that each node were initialised with a random value $\{b_1$=FALSE, b_2=FALSE, b_3=TRUE, b_4=TRUE $\}$. Then we can focus on the constraint (b_1 or $\neg b_3$ or $\neg b_4$). Given the values of b_3 and b_4, which value should b_1 take to satisfy the constraint (Figure 1.3)? the resulting constraint on b_1 will be message passed through the edge.

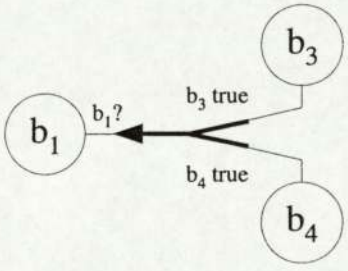

Figure 1.3: Message passing in the SAT problem.

By iterating these messages, the values of variables progressively change to satisfy the constraints. The next section shows explicitly the expression of the messages from which an iterative algorithm can be devised.

1.3 Message passing in a bipartite graph

In what follows, we'll not devise an algorithm for general graphs, but for a subset called bipartite graph because our problems will be modelled using such graphs. A bipartite graph is made of two sets of nodes $\{x_k\}_{k=1..K}$ and $\{y_\mu\}_{\mu=1..N}$ and edges link two nodes of the different sets. That is to say that links exists only between x_k, and y_μ. Let us first define the notation used in the case of a bipartite graph. We denote the set of linked nodes by $K(\mu) = \{k : x_k \text{ is linked to } y_\mu\}$ and $M(k) = \{\mu : y_\mu \text{ is linked to } x_k\}$

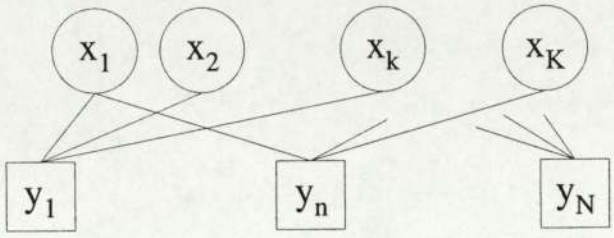

Figure 1.4: A bipartite graph.

Two messages will be passed through edges: one from y_μ to x_k and another one in the other direction. Let us define $r_{\mu k} = P(y_\mu | x_k, \{y_{\nu \neq \mu}\})$ and $q_{\mu k} = P(x_k | \{y_{\nu \neq \mu}\})$. Then we'll pass messages using the following formulae:

$$r_{\mu k} = \sum_{x_{l \neq k}, l \in K(\mu)} P(y_\mu | x_k, \{x_l\}) \prod_{l \neq k} q_{\mu l} \tag{1.1}$$

$$q_{\mu k} = \alpha_{\mu k} P(x_k) \prod_{\nu \in M(k), \nu \neq \mu} r_{\nu k} \tag{1.2}$$

Looking at $r_{\mu k}$ one can notice that the message sent from y_μ to x_k and depends on the messages that the others neighbours of y_μ send to it (Figure 1.5).

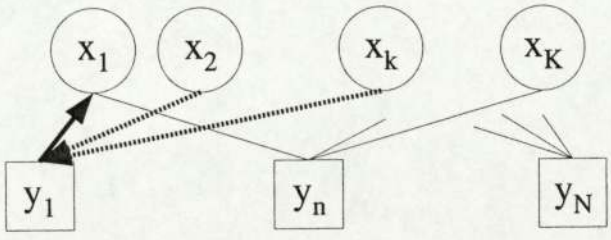

Figure 1.5: Message in the bipartite graph.

Message passing is used in problems such as K-Sat or graph colouring, but the issues I've tackled in the projects are error correction and CDMA multiuser detection. The first one is an application on a sparse graph whereas the second is an application on a dense graph. Finally a toy problem has been addressed in order to mix both approaches.

Besides, the belief propagation algorithm is exact only when the associated graph is a tree, that is to say it has no cycles [1]. But in lots of application there are cycles, so we'll just apply the algorithm, hoping that it will converge to the right solution. Some studies of the convergence in graphs with loops show that only short cycles are practically a problem.

Chapter 2

Belief Propagation (BP) in a Sparse Graph

The first case where the algorithm will be applied is a sparse graph. A sparse graph is a graph where there are a few links between nodes (for example 3 or 4 links by node where there are 1000 nodes). We'll see why this kind of graph has been chosen by looking at the complexity of the algorithm.

The message $r_{\mu k}$ is defined as $\sum_{x_{l\neq k}, l\in K(\mu)} P(y_\mu|x_k, \{x_l\}) \prod_{l\neq k} q_{\mu l}$. If we denote K the number of neighbours, then this message is a sum of 2^{K-1} terms. Looking at inside the sum, the number of terms of the product is $K-1$. So, to compute one term, $K2^{K-1}$ operations are needed. Moreover, there are NK messages to compute for each iteration.

The result is that the number of calculations highly depends on the number of neighbours. And this is exactly why belief propagation were first applied to sparse graphs. In this chapter we'll be interested in a problem that can be modelled with this type of graph, it is the well known problem referred to as decoding in Low Density Parity Check (LDPC) error correcting codes.

2.1 Noisy channel

In information theory, many models of communication may be considered, and the one we are interested in, in this chapter, is a communication through a channel known as Binary Symmetric Channel (BSC). The word *Binary* means that bits (0 or 1) are sent. During transmission, some noise will corrupt the message. Assume that we want to send a value 0, the probability that this is (correctly) received is p and the probability that an incorrect value (1) is (incorrectly) received is $1-p$. This channel is called *Symmetric* because the probability of a 1 becoming a 0 and of a 0 becoming a 1 are assumed to be the same as represented in Figure 2.1.

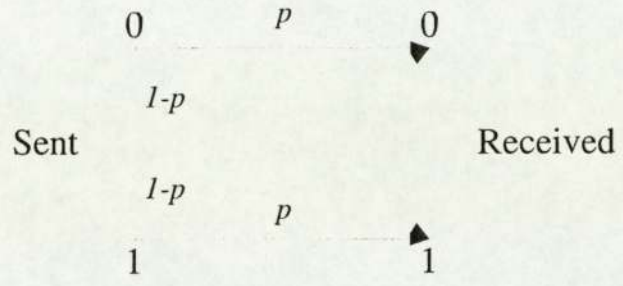

Figure 2.1: Representation of a Binary Symmetric Channel (BSC).

Another commonly used model which will be used later is the Gaussian Channel, which is also referred to as Additive White Gaussian Noise (AWGN) channel. In this case, real numbers are transmitted instead of bits. Then during the transmission some noise n is added, where n is a zero normally distributed random variable with some variance σ. The channel becomes noisier as the value of σ grows.

2.2 Error correcting codes

Error correction and error detection are areas of information theory which have great importance. They enable people who are transmitting data, to check them in order to ensure that the message has not been corrupted during transmission. Error detection only determines if data were corrupted during transmission, so that they need to be re-sent. Error correction is more powerful since it allows to retrieve the original data from the corrupted message. It is used when there is a need for reliable data, typically during transmission through noisy media or for data storage on noisy devices.

In order to correct errors, data need to be added to the sent message. An obvious way to correct errors is to repeat the message several times and then choose the value according to the majority. The process whereby data is added to provide better protection against corruption is called structural redundancy. However, detecting and correcting errors can be done with far less redundant data than that of basic repetition codes. A very popular method is called the Reed-Solomon codes [4] which are used in Compact Disc for example. Turbo codes [5] are another class of powerful error correction codes that were introduced in 1993.

2.3 Low-density parity check codes

In this chapter we're interested in a method known as low density parity-check (LDPC) error correction. LDPC codes were invented by Robert Gallager [6] in his PhD thesis. The principle is as follows.

First, a generator matrix $\boldsymbol{G}$ is used to encode the signal $\boldsymbol{s}$. The errors occurring during the transmission can be viewed as a noise vector $\boldsymbol{n}$ corrupting the transmission giving rise to a received vector $\boldsymbol{r}$ such that $\boldsymbol{r} = \boldsymbol{G}^T\boldsymbol{s} + \boldsymbol{n}$ (mod 2). In decoding, one uses a matrix $\boldsymbol{H}$ known as parity check matrix, that has the property $\boldsymbol{H}\boldsymbol{G}^T$ (mod 2) $= 0$. For Gallager codes, the parity-check matrix is a matrix built as a concatenation $\boldsymbol{H} = [\boldsymbol{C}_1|\boldsymbol{C}_2]$ of two sparse matrices, $\boldsymbol{C}_2$ being an invertible square matrix. On the other hand $\boldsymbol{G}^T = [\boldsymbol{I}|\boldsymbol{C}_2^{-1}\boldsymbol{C}_1]$ (mod 2) , where I is the identity matrix.

Using the parity-check matrix, we can write: $\boldsymbol{H}\boldsymbol{r} = \boldsymbol{H}(\boldsymbol{G}^T\boldsymbol{s} + \boldsymbol{n}) = \boldsymbol{H}\boldsymbol{n}$. With $\boldsymbol{z} = \boldsymbol{H}\boldsymbol{r}$ (called the *syndrome vector*), the decoding problem is summarised by the following equation: $\boldsymbol{z} = \boldsymbol{H}\boldsymbol{n}$ (mod 2).

The syndrome vector $\boldsymbol{z}$ is computed from the received data and $\boldsymbol{n}$ is the vector we want to find. Then knowing the noise vector and the received one, it is easy to retrieve the original signal.

2.4 Messages

Belief propagation is an iterative algorithm based on the following recursive messages [7], [8]:

$$P^{t+1}\left(z_\mu|n_k,\{z_{\nu\neq\mu}\}\right) = \sum_{n_{l\neq k}} P(z_\mu|\boldsymbol{n}) \prod_{l\neq k} P^t(n_l|\{z_{\nu\neq\mu}\}), \tag{2.1}$$

$$P^t\left(n_k|\{z_{\nu\neq\mu}\}\right) = \alpha^t_{\mu k} P(n_k) \prod_{\nu\neq\mu} P^t\left(z_\nu|n_k,\{z_{\sigma\neq\nu}\}\right), \tag{2.2}$$

where, as defined previously, $\boldsymbol{n}$ describes a noise vector, $\boldsymbol{z}$ the syndrome vector and $\alpha_{\mu k}$ a normalisation constant.

If we look at the first message, the number of terms in the sum is $O(2^{K-1})$ with K the size of $\boldsymbol{n}$; the product has $K-1$ terms. That means that the complexity of one message is $O(K2^K)$ which is very high.

But here we're doing an approximation: the only n components taken into account are the ones selected by the corresponding row of the matrix $\boldsymbol{H}$ for any given syndrome element. This matrix is sparse, that is to say that it is mostly filled with 0, which reduces consistently the complexity of the calculations . It depends on the code used, but usually, the size of $\boldsymbol{n}$ is 4 or 5. So the complexity of computing messages is finally not high.

The problem can be modelled by a graph, where a link between n_k and z_μ is created each time there is a 1 in the matrix $\boldsymbol{H}$. It gives the representation shown in Figure 2.2.

Belief Propagation can be viewed as an algorithm passing messages between the

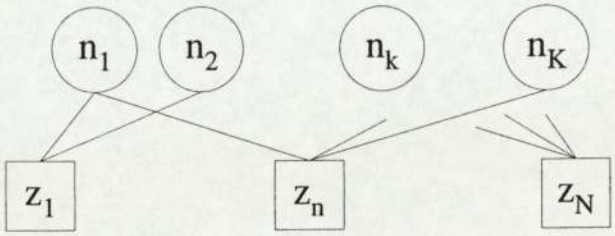

Figure 2.2: Representation of the code by a bipartite graph.

two kinds of nodes through edges:

$$r^0_{\mu k} = P^{t+1}\left(z_\mu | n_k = 0, \{z_{\nu\neq\mu}\}\right) \tag{2.3}$$
$$r^1_{\mu k} = P^{t+1}\left(z_\mu | n_k = 1, \{z_{\nu\neq\mu}\}\right) \tag{2.4}$$

$$q^0_{\mu k} = P^t\left(n_k = 0 | \{z_{\nu\neq\mu}\}\right) \tag{2.5}$$
$$q^1_{\mu k} = P^t\left(n_k = 1 | \{z_{\nu\neq\mu}\}\right) \tag{2.6}$$

2.5 Improvement

There is no need to pass 4 messages through nodes because all the information is contained in the difference between q^0, q^1 and r^0 and r^1 respectively. We therefore pass the following messages:

$$\delta r_{\mu k} = r^0_{\mu k} - r^1_{\mu k} \tag{2.7}$$
$$\delta q_{\mu k} = q^0_{\mu k} - q^1_{\mu k}. \tag{2.8}$$

In [8] the messages are finally expressed as:

$$\delta r_{\mu k} = (-1)^{z_\mu} \prod_l \delta q_{\mu_l} \tag{2.9}$$
$$\delta q_{\mu k} = q^0_{\mu k} - q^1_{\mu k}. \tag{2.10}$$

using:

$$r^0_{\mu k} = (1 + \delta r_{\mu k})/2 \tag{2.11}$$
$$r^1_{\mu k} = (1 - \delta r_{\mu k})/2 \tag{2.12}$$
$$q^0_{\mu k} = \alpha_{\mu k} P(n_k = 0) \prod_{\nu\in M(k),\nu\neq\mu} r^0_{\nu k} \tag{2.13}$$
$$q^1_{\mu k} = \alpha_{\mu k} P(n_k = 1) \prod_{\nu\in M(k),\nu\neq\mu} r^1_{\nu k}. \tag{2.14}$$

The initial values of $q^0_{\mu k}$ and $q^1_{\mu k}$ are set according to the flip probability which is the probability of having $n_k = 1$.

2.6 Experiments

The first task to do some experiments is to generate sparse graphs. $K = 3$ parity checks has been used and $\boldsymbol{H}$ has been set to a $(750, 1000)$ matrix. That means that each z_μ is linked to 3 n_k, and each n_k to 4 z_μ.

In order to study the efficiency of the algorithm, experiments have been carried out: 10 graphs were generated, and for each of them, 50 noise vectors were created. The algorithm was applied to each of the graphs and noise vectors and the number of errors between the estimated noise vector and the found one was recorded. The flip probability has been increased and recorded the percentage of incorrect bits and the results are shown in Figure 2.3.

The efficiency of decoding a code is measured by its *dynamical transition* which is the limit where the code achieves excellent results. Considering the chosen values, the transition currently achieved by error correcting codes is about 0.16 where the Shannon Limit is 0.21. My experiments gives similar results as shown in Figure 2.3.

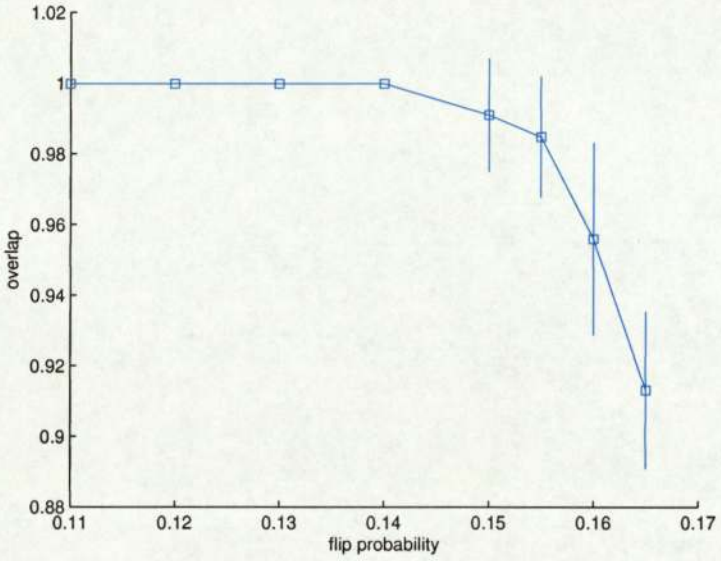

Figure 2.3: Percentage of correctly decoded noise components found in the estimated noise vector. The size of the noise vector is 1000, and the number of parity-checks is 3. The dynamical transition is a bit below what is expected (flip probability of 0.15 instead of 0.16).

The second part of the experiment was to record the evolution of the system through iterations. The results are shown in Figure 2.4. The graph settings are the same as above, and as expected, the algorithm converges quite fast for a small amount of noise. It finishes when all the constraints due to the parity-check matrix are respected. But when the flip probability becomes high, a maximum number of iterations has been

fixed. This maximum of iterations directly influences the results of the limit where we can consider that the algorithm converges or not, i.e. the dynamical transition. For example, if the maximum number of iterations had been set to 10, then the algorithm doesn't allways converges from a flip probability of 0.13 .

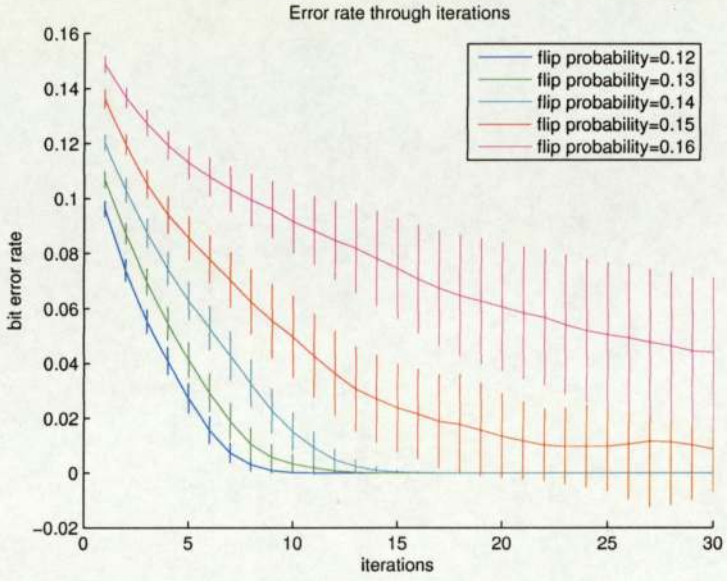

Figure 2.4: Percentage of errors found between the true noise vector, and the estimated one after each iteration. We can see that after 30 iterations, when the signal is too noisy, the algorithm doesn't converge.

When the update rules are known, the algorithm to update the messages is not really difficult to devise. However there are two points to take care of. The first one is the initial state of the system, we have seen that it is a logical choice to start by setting the value of $P^t(n_k = 1|\{z_{\nu\neq\mu}\})$ to the flip probability used to generate the noise vector. The other aspect of the programming task is to choose how to carry out the updates on messages. There are basically two approaches. The first one is the sequential update where, when a message is computed, it is used immediately when updating the others. And the second approach is the simultaneous update where all the messages updates are computed, and at the end of the iteration, they are all updated in the same time. There wasn't big differences, so the simultaneous update, which was the first tried, was kept.

Chapter 3

BP in Dense Graphs

3.1 CDMA

In the telecommunication area, people often face the problem to send different signals at the same time. This creates the problem of interferences between signals and the need for a method that enables one to avoid such interference. Several methods have been devised to that effect. The first of the multiple access technique used today is the Frequency Division Multiple Access (FDMA). In this scheme, the signals are sent on different frequencies to avoid interference. Message retrieval is done by "listening" at the desired frequency. This technique is used with radios of wide spectrum. A new problem raises when several people need to communicate through the same frequency channel.

That's why a second technique, Time Division Multiple Access (TDMA) may be used. It allows users to share the same frequency by dividing the time in slots. The users transmit in rapid succession their data, one after each other, using their own time slot (Figure 3.1). Thus, this allows multiple users to share the same frequency band. To get the data, the receiver need to synchronise their transmission with the time slots.

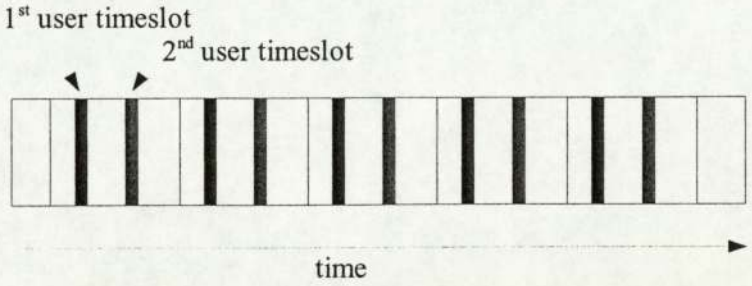

Figure 3.1: TDMA: Users send their data one after each other.

Used in the GSM, PDC and iDEN digital cellular standards, among others, TDMA is also used extensively in satellite systems and local area networks. But a drawback of

TDMA systems is that they create interference at frequencies which are directly linked to the time slot length. This is the irritating buzz which can sometimes be heard if a GSM phone is left next to a radio. Another disadvantage is that "dead time" between time slots limits the potential bandwidth of a TDMA channel. That's why the next mobile phone generation will use a technique called CDMA for Code Division Multiple Access. This is the scheme used in the UMTS, one of the third-generation mobile phone technologies.

CDMA can be referred to as the use of spreading codes by transmitters to share the same frequency channel. The transmitted data are first multiplied by a user spreading code and then summed leading in a single signal containing all data, as will be explained in details later.

The term CDMA is also widely used to refer to a family of specific implementations of the CDMA technique pioneered by Qualcomm for use in digital cellular telephony.

3.2 CDMA in detail

This chapter is based on the work of Kabashima who studied this dense case in [9]. We'll look at how the users bits are mixed, using spreading codes. A CDMA signal is expressed as

$$y_\mu = \frac{1}{\sqrt{N}} \sum_{k=1}^{K} s_{\mu k} b_k, \tag{3.1}$$

where $\mu \in \{1, 2, \ldots, N\}$ and $k \in \{1, 2, \ldots, K\}$ are indices for samples and users, respectively. That is to say that K users try to send their bit b_k, using their spreading code sequence $s_{\mu k}$, and the resulting signal length is $N > K$.

We assume that the signal is transmitted through a Gaussian Channel, thus some Gaussian noise is added during transmission, resulting in the following received signal:

$$y_\mu = \frac{1}{\sqrt{N}} \sum_{k=1}^{K} s_{\mu k} b_k + \sigma_0 n_\mu, \tag{3.2}$$

where n_μ is a Gaussian white noise component with zero mean and unit variance and σ_0 the standard deviation of Additive White Gaussian Noise (AWGN). The transmission process is summarised in Figure 3.2. Using these normalisations, the signal to noise ratio is defined as $SNR = \beta/(2\sigma_0^2)$ where $\beta = K/N$ is referred to as the system load. In the following, we assume a situation where both of N and K are large keeping β finite.

The decoding problem can be modelled by a graph, where each b_k is linked to all y_μ if b_k. So the graph is dense and we cannot apply the previous belief propagation algorithm because there are too many neighbours, and also very short cycles as shown in Figure 3.3.

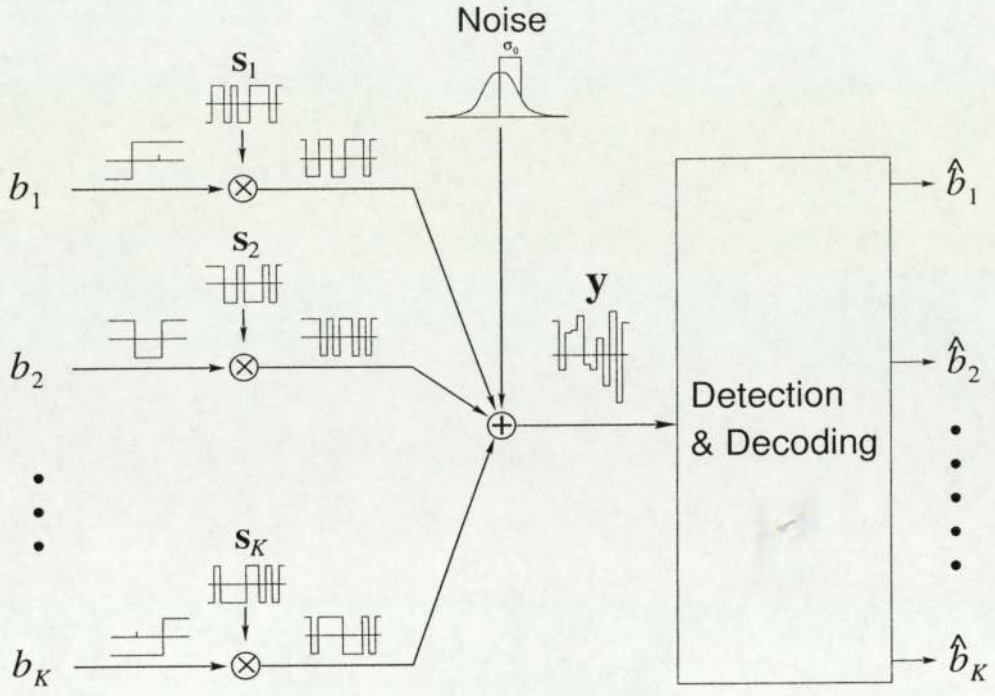

Figure 3.2: CDMA: User messages are spread on modulated transmission.

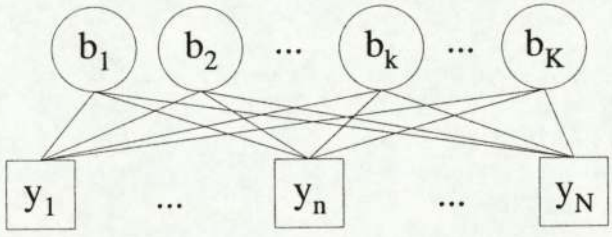

Figure 3.3: Graph modelling the CDMA decoding problem.

3.3 Messages

Also here, belief propagation can be defined as an algorithm passing messages between the two kinds of nodes through edges as

$$P^{t+1}\left(y_\mu|b_k,\{y_{\nu\neq\mu}\}\right) = \sum_{b_{l\neq k}} P(y_\mu|\boldsymbol{b}) \prod_{l\neq k} P^t(b_l|\{y_{\nu\neq\mu}\}), \tag{3.3}$$

$$P^t\left(b_k|\{y_{\nu\neq\mu}\}\right) = \alpha^t_{\mu k} \prod_{\nu\neq\mu} P^t\left(y_\nu|b_k,\{y_{\sigma\neq\nu}\}\right), \tag{3.4}$$

where $t = 1, 2, \ldots$ is an index for counting the number of updates. There is also a constant $\alpha^t_{\mu k}$ due to the constraints $\sum_{y_\mu=\pm1} P^t\left(y_\mu|b_k,\{y_{\nu\neq\mu}\}\right) = 1$ and $\sum_{b_k=\pm1} P^t\left(b_k|\{y_{\nu\neq\mu}\}\right) = 1$, respectively. The marginalised posterior at tth update is evaluated from $P^t\left(y_\mu|b_k,\{y_{\nu\neq\mu}\}\right)$ as $P^t(b_k|\boldsymbol{y}) = \alpha_k \prod_{\mu=1}^N P^t\left(y_\mu|b_k,\{y_{\nu\neq\mu}\}\right)$, where α_k is again a normalisation constant.

As b_k is a binary variable, one can parameterise the above functions as

$$\begin{aligned} P^t\left(y_\mu|b_k,\{y_{\nu\neq\mu}\}\right) &\propto (1+\hat{m}^t_{\mu k}b_k)/2, \\ P^t\left(b_k|\{y_{\nu\neq\mu}\}\right) &= (1+m^t_{\mu k}b_k)/2 \\ \text{and} \quad P^t(b_k|\boldsymbol{y}) &= (1+m^t_k b_k)/2 \end{aligned}$$

without loss of generality, which simplifies the expressions to

$$\hat{m}^{t+1}_{\mu k} = \frac{\sum_{\boldsymbol{b}} b_k P(y_\mu|\boldsymbol{b}) \prod_{l\neq k}\left(\frac{1+m^t_{\mu l}b_l}{2}\right)}{\sum_{\boldsymbol{b}} P(y_\mu|\boldsymbol{b}) \prod_{l\neq k}\left(\frac{1+m^t_{\mu l}b_l}{2}\right)}, \tag{3.5}$$

$$m^t_{\mu k} = \tanh\left(\sum_{\nu\neq\mu}\tanh^{-1}\hat{m}^t_{\nu k}\right). \tag{3.6}$$

Employing these variables, the approximated posterior average of b_k at tth update can be computed as $m^t_k = \tanh\left(\sum_{\mu=1}^N \tanh^{-1}\hat{m}^t_{\mu k}\right)$. After convergence, the value of b_k is 1 if $m^t_k > 0$ and -1 if $m^t_k < 0$.

3.4 Complexity

To calculate $\hat{m}_{\mu k}$, a total of $O(K2^K)$ computations are required. It grows so fast that it becomes quickly intractable. That is the reason why some approximations need to be done. The first approximation used in the previous chapter which was to consider only a few neighbours, cannot be used here. But there is another way to approximate the big sum due to its large number of weakly correlated terms.

3.5 Derivation

The noise model leads to the following expression:

$$P(y_\mu|\boldsymbol{b}) = \frac{1}{\sqrt{2\pi\sigma_0^2}}\exp\left[-\frac{1}{2\sigma_0^2}\left(y_\mu-\Delta_\mu\right)^2\right], \tag{3.7}$$

where $\Delta_\mu = \frac{1}{\sqrt{N}}\sum_{k=1}^K s_{\mu k}b_k$.

Since $s_{\mu k}b_k/\sqrt{N}$ is small for large N, we expand this conditional probability as

$$\begin{aligned} P(y_\mu|\boldsymbol{b}) &\simeq \frac{1}{\sqrt{2\pi\sigma_0^2}}\exp\left[-\frac{\left(y_\mu-\Delta_{\mu k}\right)^2}{2\sigma_0^2}+\frac{s_{\mu k}(y_\mu-\Delta_{\mu k})}{\sqrt{N}\sigma_0^2}b_k\right] \\ &\simeq \frac{1}{\sqrt{2\pi\sigma_0^2}}\exp\left[-\frac{\left(y_\mu-\Delta_{\mu k}\right)^2}{2\sigma_0^2}\right]\left(1+\frac{s_{\mu k}(y_\mu-\Delta_{\mu k})}{\sqrt{N}\sigma_0^2}b_k\right), \end{aligned} \tag{3.8}$$

where $\Delta_{\mu k}\equiv\sum_{l\neq k}s_{\mu l}b_l/\sqrt{N}$. As the spreading codes are generated independently, $s_{\mu l}$ and b_l would be uncorrelated when b_l is generated from $P^t(b_l|\{y_{\nu\neq\mu}\}) = (1+m^t_{\mu l}b_l)/2$.

3.5.1 Macroscopic property

In the previous algorithm, each neighbour was considered in order to calculate the messages. This could be referred to as looking at microscopic properties of the node. But, because there are many considered nodes in this dense case, we can use some quantity describing the whole set. This macroscopic property of the neighbours is the sum found in $P(y_\mu|\boldsymbol{b})$ which is $\Delta_{\mu k} = \sum_{l\neq k} s_{\mu l} b_l/\sqrt{N}$. This is represented on Figure 3.4 by an area linking $\boldsymbol{b}$ components to y_μ.

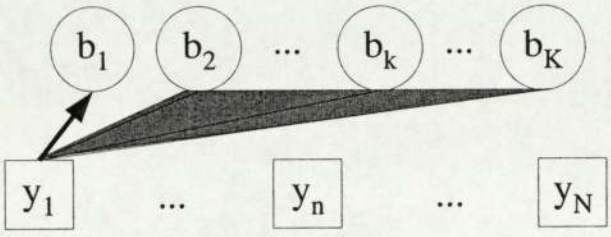

Figure 3.4: The messages use macroscopic properties from the neighbours.

The central limit theorem implies that $\Delta_{\mu k}$ obeys a normal distribution whose mean is $\left\langle \Delta^t_{\mu k} \right\rangle_\mu$ and whose variance can be written $\beta(1 - Q^t_{\mu k})$.

$$\left\langle \Delta^t_{\mu k} \right\rangle_\mu = \sum_{l\neq k} s_{\mu l} m^t_{\mu l}/\sqrt{N} \text{ and } Q^t_{\mu k} \equiv (1/K) \sum_{l\neq k} (m^t_{\mu l})^2$$

. Those two quantities don't depend explicitly on $\boldsymbol{b}$ anymore but only on messages coming from neighbours.

Now that we know the distribution of $\Delta_{\mu k}$, we can use the following property

$$P(y_\mu|\boldsymbol{b}) = \int d\Delta_{\mu k} P(\Delta_{\mu k}) P(y_\mu|\Delta_{\mu k}),$$

which finally doesn't depend at all on the particular components of $\boldsymbol{b}$. This is a great improvement since it is calculated only once for each message. Further calculations in [9] leads to final formulae which are much simpler and the computational cost has been reduced to $O(K)$ per pair (μk) which implies a total of $O(NK^2)$ computations per update. Without optimisations it would have been $O(N2^K K)$ per update.

3.6 Experiments

The algorithm has been implemented and then used with increasing values of σ_0 in order to see how the algorithm converges. The number of users has been set to $K = 500$, and the number of message samples is $N = 1000$. So the graphic shows how the number of errors evolves when the noise grows. The following graphs are made by averaging 500 experiments with the same level of noise.

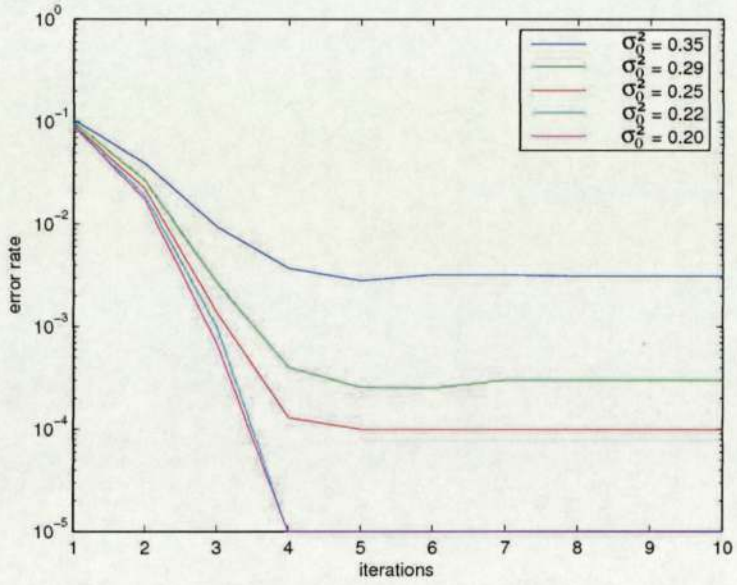

Figure 3.5: Experiment on a dense graph. It shows the number of errors between the true vector and the estimated one after each iteration. The divergence threshold is about $\sigma_0^2 = 0.25$

Figure 3.5 shows the influence of noise parameter σ_0^2 on algorithm convergence. Each points represents the simulation average after each iteration, and error bars are too small to be viewed. We can see that the algorithm converges to the right solution until $\sigma_0^2 = 0.22$ and starts to diverge after. After each iteration, the posterior probability was evaluated and recorded. It shows that the convergence is very fast; after 4 or 5 iterations the number of errors doesn't change a lot.

Another quantity is commonly used instead of σ_0^2 is called the signal to noise ratio defined as $SNR = \beta/(2\sigma_0^2)$ with $\beta = K/N$.

Chapter 4

BP in a Dense Graph with a Few Strong Links

The main purpose of this chapter is to mix the two previous algorithms in order to use belief propagation in dense graphs where there are still a few strong links. A practical problem is not tackled here but instead we've created a toy model to investigate the efficiency of the algorithm. This is a first step towards its application to a real problem.

4.1 A toy problem

The issue addressed in the toy problem is very similar to the CDMA decoding problem. K bits are modulated using spreading coefficients and the resulting signal of size N is sent through a Gaussian channel. The components y_μ, $\mu \in \{1..N\}$ of the received signal can be written:

$$y_\mu = \frac{1}{\sqrt{K}} \sum_{k=1}^{K} s_{\mu k} b_k + \frac{1}{\sqrt{J}} \sum_{j=1}^{J} s_{\mu j} b_j + \sigma_0 n_\mu, \tag{4.1}$$

The number of signal bits N is a quite big number, and K scales as $O(N)$ whereas J scales as $O(1)$. The factors $\frac{1}{\sqrt{K}}$ and $\frac{1}{\sqrt{J}}$ are very important since they measure the influence of a $\boldsymbol{b}$ component on the final signal.

Looking at $\frac{1}{\sqrt{K}} \sum_{k=1}^{K} s_{\mu k} b_k$, we notice that each b_k does not contribute a lot (its participation scales as $O(\frac{1}{\sqrt{K}})$), whereas each b_j is much more important since its contribution scales as $O(1)$. Links between b_k and y_μ are termed *weak*, whereas a link between b_j and y_μ are termed as *strong*. Figure 4.1 shows the two groups of nodes.

The other components of the signal are the spreading code $s_{\mu i} \in \{-1, 1\}$ which is generated such as $P(s_{\mu i} = +1) = P(s_{\mu i} = -1) = 1/2$, n_μ which is a Gaussian white noise sample with zero mean and unit variance and σ_0 the standard deviation of AWGN.

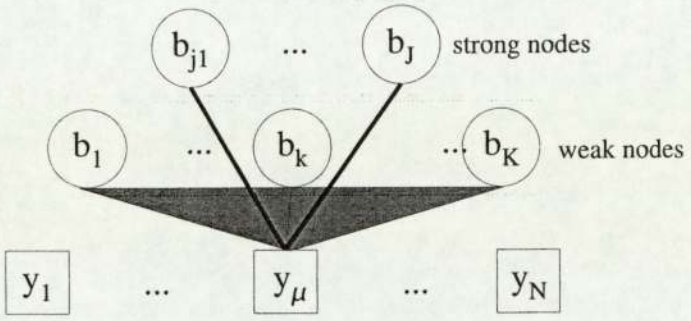

Figure 4.1: Representation of the toy problem by a graph: there are a lot of weak links, and only a few strong.

4.2 Messages

4.2.1 Messages overview

Using the same notation as in the CDMA decoding problem, we are interested in computing the following message for each link:

$$\hat{m}_{\mu k}^{t+1} = \frac{\sum_{\boldsymbol{b}} b_k P(y_\mu|\boldsymbol{b}) \prod_{l\neq k} \left(\frac{1+m_{\mu l}^t b_l}{2}\right)}{\sum_{\boldsymbol{b}} P(y_\mu|\boldsymbol{b}) \prod_{l\neq k} \left(\frac{1+m_{\mu l}^t b_l}{2}\right)},$$

where, due to the Gaussian noise added through the transmission,

$$P(y_\mu|\boldsymbol{b}) = \frac{1}{\sqrt{2\pi\sigma_0^2}} \exp\left[-\frac{1}{2\sigma_0^2}\left(y_\mu - \Delta_\mu\right)^2\right],$$

and $\Delta_\mu = \frac{1}{\sqrt{K}} \sum_{k=1}^{K} s_{\mu k} b_k + \frac{1}{\sqrt{J}} \sum_{j=1}^{J} s_{\mu j} b_j$, which now includes the strong components.

The value Δ_μ has to be computed for each possible value of $\boldsymbol{b}$. That's why, in the same way as what was done in the previous chapter, we'll look at the macroscopic properties of the incoming messages. If such general property can be found, there will be no need to compute Δ_μ for each new set.

The problem is that it cannot be found with respect to Δ_μ because the strong b_js are too influent. And for each new set of $\boldsymbol{b}$ components, the statistical properties of Δ_μ may change completely. But, we can split the marginalisation over the whole set into one sum over the strong one ($\boldsymbol{b_s}$) and another one over the weak one ($\boldsymbol{b_w}$). This enables us to consider that for each strong set, the quantity $\frac{1}{\sqrt{J}} \sum_{j=1}^{J} s_{\mu j} b_j$ is fixed and known.

The messages then become:

$$\hat{m}_{\mu k}^{t+1} = \frac{\sum_{\boldsymbol{b_s}} \sum_{\boldsymbol{b_w}} b_k P(y_\mu|\boldsymbol{b}) \prod_{l\neq k} \left(\frac{1+m_{\mu l}^t b_l}{2}\right)}{\sum_{\boldsymbol{b_s}} \sum_{\boldsymbol{b_w}} P(y_\mu|\boldsymbol{b}) \prod_{l\neq k} \left(\frac{1+m_{\mu l}^t b_l}{2}\right)},$$

$$m_{\mu k}^t = \tanh\left(\sum_{\nu\neq\mu}\tanh^{-1}\hat{m}_{\nu k}^t\right).$$

There are some differences when derivating the messages through a strong or weak link. on the other hand. The whole derivations can be found in Appendix A. Moreover derivation on the numerator and the denominator are very similar only the numerator is studied in detail here.

Let us define $S(\mu)$ the subset of indices such as $b_l, l \in S(\mu)$ denotes a strong $\boldsymbol{b}$ component. In the same way, $b_l, l \in W(\mu)$ denotes a weak component.

The numerator is first split into two sums:

$$\begin{aligned}&\sum_{\boldsymbol{b_s}}\sum_{\boldsymbol{b_w}} b_k P(y_\mu|\boldsymbol{b})\prod_{j\neq k}\left(\frac{1+m_{\mu j}^t b_j}{2}\right)\\ =\ &\sum_{\boldsymbol{b_s}}\prod_{l\neq k, j\in S(\mu)}\left(\frac{1+m_{\mu l}^t b_l}{2}\right)\sum_{\boldsymbol{b_w}} b_k P(y_\mu|\boldsymbol{b})\prod_{l\neq k, l\in W(\mu)}\left(\frac{1+m_{\mu l}^t b_l}{2}\right).\end{aligned}$$

In both case, the sum over $\boldsymbol{b_s}$ cannot be avoided, so we'll focus on the quantity $b_k P(y_\mu|\boldsymbol{b})$. We want to find it independent on the value of $\boldsymbol{b_w}$. This derivation is not the same if the message is sent through a strong or a weak link because if b_k is strong then it can be moved out of the sum over $\boldsymbol{b_w}$.

4.2.2 Derivation if k is strongly connected

In this case, we look at the quantity $\Delta_\mu = \frac{1}{\sqrt{K}}\sum_{k=1}^K s_{\mu k}b_k + \frac{1}{\sqrt{J}}\sum_{j=1}^J s_{\mu j}b_j$, where $\frac{1}{\sqrt{J}}\sum_{j=1}^J s_{\mu j}b_j$ is considered as a constant. Using the central limit theorem, Δ_μ obeys a Gaussian distribution. The mean is $\left\langle \Delta_\mu^t \right\rangle_\mu = \frac{1}{\sqrt{K}}\sum_l s_{\mu l}m_{\mu l}^t + \frac{1}{\sqrt{J}}\sum_{j=1}^J s_{\mu j}b_j$ and the variance $(1-Q^t)$ where $Q^t = (1/K)\sum_{k\in W(\mu)}(m_k^t)^2$. The effect of the strong b_j is finally a shift of the mean.

Then using the marginalisation over Δ_μ,

$$\begin{aligned}P(y_\mu|\boldsymbol{b}) &= \int d\Delta_\mu P(\Delta_\mu)P(y_\mu|\Delta_\mu)\\ &= \frac{1}{\sqrt{2\pi}}\frac{1}{\sqrt{\sigma_0^2+1-Q^t}}\exp\left[-\frac{(y_\mu-\langle\Delta_\mu\rangle)^2}{2\left(\sigma_0^2+1-Q^t\right)}\right],\end{aligned}$$

which doesn't depend on $\boldsymbol{b_w}$ anymore.

And finally using the property $\sum_{\boldsymbol{b_w}}\prod_{l\in W(\mu)}\left(\frac{1+m_{\mu l}^t b_l}{2}\right) = 1$ the final message is:

$$\hat{m}_{\mu k}^{t+1} = \frac{\sum_{\boldsymbol{b_s}} b_k P(y_\mu|\boldsymbol{b})\prod_{l\in S(\mu)}\left(\frac{1+m_{\mu l}^t b_l}{2}\right)}{\sum_{\boldsymbol{b_s}} P(y_\mu|\boldsymbol{b})\prod_{l\in S(\mu)}\left(\frac{1+m_{\mu l}^t b_l}{2}\right)}$$

4.2.3 Derivation if k is weakly connected

The differences with the strongly connected case is that the macroscopic property of $\Delta_{\mu k} = \frac{1}{\sqrt{K}} \sum_{l=1, l\neq k}^{K} s_{\mu l} b_l + \frac{1}{\sqrt{J}} \sum_{j=1}^{J} s_{\mu j} b_j$ is studied instead of Δ_μ. The term containing b_k is removed.

Once again, this quantity obeys a Gaussian distribution whose mean is $\left\langle \Delta_{\mu_k}^t \right\rangle_\mu = \frac{1}{\sqrt{K}} \sum_{l\neq k} s_{\mu l} m_{\mu l}^t + \frac{1}{\sqrt{J}} \sum_{j=1}^{J} s_{\mu j} m_{\mu j}^t$ and the variance $(1-Q^t)$ where $Q^t = (1/K) \sum_k (m_k^t)^2$. The detailed derivation can be found in Appendix A, and the following result is found in the same way as the previous part:

$$\hat{m}_{\mu k}^{t+1} = \frac{\sum_{\boldsymbol{b_s}} A.B \prod_{l\in S(\mu)} \left(\frac{1+m_{\mu l}^t b_l}{2} \right)}{\sum_{\boldsymbol{b_s}} B \prod_{l\in S(\mu)} \left(\frac{1+m_{\mu l}^t b_l}{2} \right)}$$

where

$$A = \frac{\left[y_\mu - \langle \Delta_{\mu k} \rangle \right] s_{\mu k}}{\left[\sigma_0^2 + 1 - Q^t \right] \sqrt{K}}$$

and

$$B = \frac{1}{\sqrt{2\pi(\sigma_0^2 + 1 - Q^t)}} \exp\left[-\frac{\left(y_\mu - \langle \Delta_{\mu k} \rangle \right)^2}{2\left(\sigma_0^2 + 1 - Q^t \right)} \right].$$

4.3 Experiments

To validate the results obtained so far, numerical experiments were performed on systems of size $N = 1000$ and $K = 750$. The number of strong bits was set to $J = 3$. All the graphs and vectors were randomly generated, and as in the sparse case, very short cycles were avoided in the strong links generation.

The simulations results are shown on Figures 4.2 and 4.3. Each point represents the average error on 500 trials and an error bar represents the variance obtained from the simulations results. As it may be done in other papers, error is not represented as the overlap between the original vector and the decoded one, but as a percentage of wrong bits on the whole signal.

On Figure 4.2 this number of errors was first recorded, while increasing the level of noise. It was set gradually increased and we can see that the true signal is almost perfectly retrieved until a Gaussian noise with variance 0.25 .

And finally, we can see the convergence of the algorithm through iterations. This is what Figure 4.3 shows. We can notice that the convergence is slower than in the CDMA case but it still converges until $\sigma = 0.3$.

As in the previous problems, the number of errors was recorded as a function of σ. But the effect of σ is not the same as in the previous case, because the signal is made of 2 sums in the toy problem whereas there is only one in CDMA decoding. Moreover, the normalisation is not exactly the same ($1/\sqrt{K}$ and $1/\sqrt{J}$ instead of $1/\sqrt{N}$). So in

order to compare the efficiency of the algorithms, it is better to compare the number of errors using the signal to noise ratio (SNR) which is defined at the end of the previous chapter.

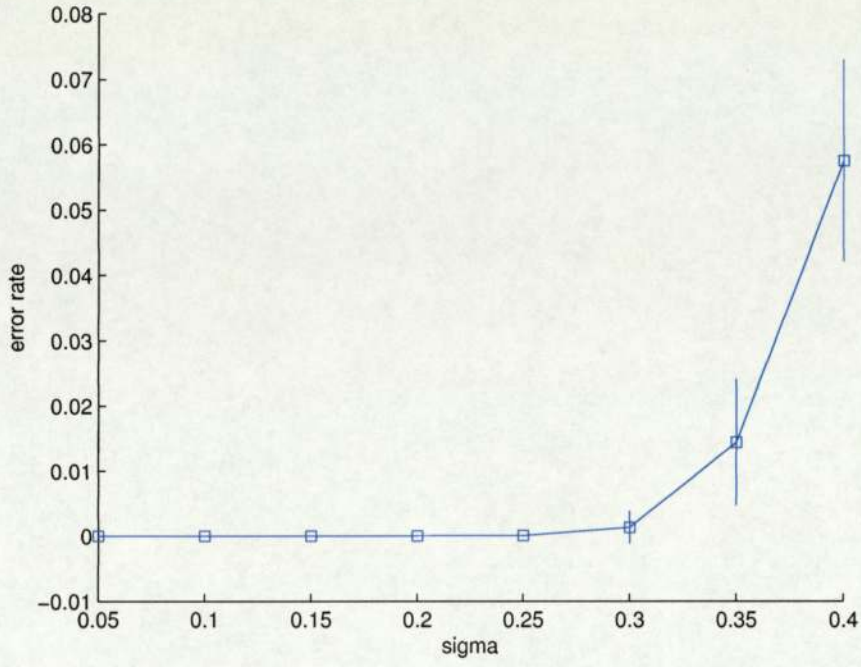

Figure 4.2: Experiment carried out with a dense graph and a few strong links. The number of weak links for each y_μ was set to 747 and there was 3 strong links.

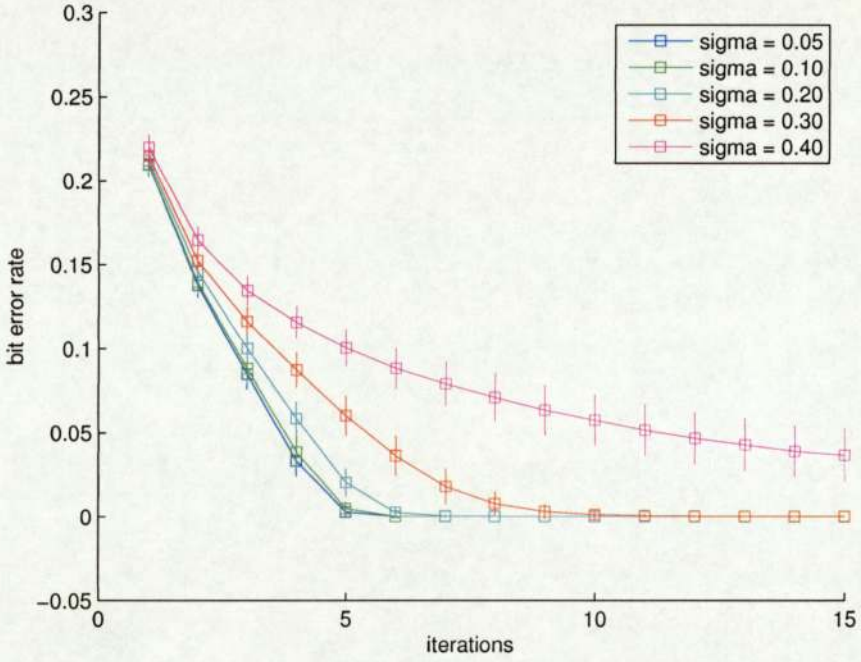

Figure 4.3: The number of wrong estimated bits where recorded after each iteration. The algorithm converges very slowly with a high level of noise

Chapter 5

Conclusion

In the first part of the project, we have seen that message passing techniques are a useful tools for inference. In this approach, messages containing probabilistic information about nodes in the graph are being passed from node to node. Some equations were devised from which an algorithm can be deduced to solve iteratively our problems modelled by a graph. This algorithm is known as "belief propagation".

However, the number of calculations on each iteration scales exponentially as the number of neighbours of a node. This is why the first application of belief propagation has been done on a sparse graph. Such a sparse graph is found in the error correction theory, and more precisely in Low Density Parity Check Codes. The efficiency of the algorithm has been shown through experiments were the estimated vector matches the true one until a flip probability of 0.15 . Moreover, a look at the iterations shows that the belief propagation algorithm converges quickly and is thus computationally efficient.

Due to the exponential complexity of the algorithm, belief propagation to a dense graph couldn't be applied in a reasonable amount of time. Moreover, a dense graph is made of a lot of short cycles, which is a problem as regards algorithm convergence. However, it has been recently performed successfully on a dense graph modelling a decoding problem used in the CDMA protocol. Contrary to the previous error decoding problem, a Gaussian channel is used instead of a binary channel during transmission.

The reason why there are so many computations to do in the message, is the marginalisation over the set of neighbours linked to the node which is about 750. This means a sum of 2^{750} terms. But that big number of nodes allows one to study the set globally, looking for some macroscopic properties. This property was found to be independent of the particular values of the nodes, it was just a function of the incoming messages. The marginalisation over the set of node was not needed anymore and update rules were simplified. The final rules were finally surprisingly simple to program, so the results were very close to those in the corresponding article.

The final task of the project was to mix both approaches. A toy problem wad

considered, based on the CDMA decoding problem. The signal was made of two sums: one made of a lot of weakly connected terms (about 750), and the other one of only a few strongly connected. Both terms were "normalised" by the squared root of the number of term. Both sum contribution were of a similar order. That "normalising" factor also means that the terms in the big sum had less effects that the one in the sum of only a few terms. The first were said weak, and the other strong.

The derivation of update rules uses both previous approaches, since the marginalisation over the strong components couldn't be avoided whereas the one over the weak links was simplified looking at the macroscopic property of the nodes. Finally an algorithm was devised, where the messages through strong links were not the same as the messages through weak links. Due to the strong links, the update rules could not have been simplified as much as the CDMA detection rules. The experiments show that the algorithm converges quite quickly for a low level of noise and converges slower as the noise is increased. It could be compared to the convergence in a dense graph, it seems slower due to the strong links.

Finally, some future work can be done on error correcting codes. Indeed, it is known that some correlation exists between nodes which are not directly linked. This motivates the view that nodes which participates in the parity check are strongly connected, whereas all others are weakly connected. Unfortunately, update rules could not be easily devised in this case. Besides the new decoding algorithm devised in this project considers a real signal transmitted to a Gaussian channel. The updates rules also use the fact that the infered values are bits, so using the update rules to another problem may need some adaptation.

Appendix A

Toy Problem: Message Derivation

In this chapter, two cases need to be examinated when calculating m_{μ_k}. If k is in the strongly connected local neighbourhood then the equations will be build as in the case of the sparse graph, whereas the approximations used in the CDMA problem are used to deal with the case where k is in the weakly connected group.

A.1 Strong case : k is in the strongly connected local neighbourhood

The messages $\hat{m}_{\mu k}$ and $m_{\mu k}$ can be decomposed by writing two sums, one over the strongly connected bits, and the other over the weakly connected bits.

$$\begin{aligned} \hat{m}_{\mu k}^{t+1} &= \frac{\sum_{\boldsymbol{b_s}} \sum_{\boldsymbol{b_w}} b_k P(y_\mu|\boldsymbol{b}) \prod_{l\neq k} \left(\frac{1+m_{\mu l}^t b_l}{2}\right)}{\sum_{\boldsymbol{b_s}} \sum_{\boldsymbol{b_w}} P(y_\mu|\boldsymbol{b}) \prod_{l\neq k} \left(\frac{1+m_{\mu l}^t b_l}{2}\right)}, \\ m_{\mu k}^t &= \tanh\left(\sum_{\nu\neq\mu} \tanh^{-1} \hat{m}_{\nu k}^t\right). \end{aligned}$$

Considering $\Delta_\mu = \frac{1}{\sqrt{K}} \sum_{l=1}^{K} s_{\mu l} b_l + \frac{1}{\sqrt{J}} \sum_{j=1}^{J} s_{\mu j} b_j$, and using the central limit theorem: Δ_μ obeys a Gaussian distribution $\mathcal{N}\left(\left\langle \Delta_\mu^t \right\rangle_\mu, (1 - Q_\mu^t)\right)$, where

$$\left\langle \Delta_\mu^t \right\rangle_\mu = \frac{1}{\sqrt{K}} \sum_l s_{\mu l} m_{\mu l}^t + \frac{1}{\sqrt{J}} \sum_{j=1}^{J} s_{\mu j} b_j \tag{A.1}$$

and

$$Q_\mu^t = (1/K) \sum_l (m_{\mu l}^t)^2. \tag{A.2}$$

which is well approximated by

$$Q^t = (1/K)\sum_k (m_k^t)^2. \tag{A.3}$$

Back to the probability

$$P(y_\mu|\boldsymbol{b}) = \frac{1}{\sqrt{2\pi\sigma_0^2}} \exp\left[-\frac{1}{2\sigma_0^2}\left(y_\mu - \Delta_\mu\right)^2\right];$$

as we know the probability density function of Δ_μ, we can use the following formula:

$$P(y_\mu|\boldsymbol{b}) = \int d\Delta_\mu P(\Delta_\mu) P(y_\mu|\Delta_\mu),$$

which leads to:

$$\begin{aligned} P(y_\mu|\boldsymbol{b}) &= \int d\Delta_\mu \frac{1}{\sqrt{2\pi\sigma_0^2}} \exp\left[-\frac{\left(y_\mu - \Delta_\mu\right)^2}{2\sigma_0^2}\right] \frac{1}{\sqrt{2\pi(1-Q^t)^2}} \exp\left[-\frac{\left(\Delta_\mu - \langle\Delta_\mu\rangle\right)^2}{2(1-Q^t)^2}\right], \\ &= \frac{1}{\sqrt{2\pi}} \frac{1}{\sqrt{\sigma_0^2 + 1 - Q^t}} \exp\left[-\frac{\left(y_\mu - \langle\Delta_\mu\rangle\right)^2}{2\left(\sigma_0^2 + 1 - Q^t\right)}\right] \end{aligned}$$

Back to the messages, we have now approximated the first probability $P(y_\mu|\boldsymbol{b})$ and it doesn't depend on the b_k. This is important because now it can be moved out of the sum over the weakly connected bits. However there is still another term in that sum which would lead to many calculations if not modified : $\prod_{l\in W(\mu)}\left(\frac{1+m_{\mu l}^t b_l}{2}\right)$. This term is quite simple to remove because we can also write it as :

$$\prod_{l\in W(\mu)} \sum_{\boldsymbol{b_w}} \left(\frac{1+m_{\mu l}^t b_l}{2}\right) = 1$$

$$\hat{m}_{\mu k}^{t+1} = \frac{\sum_{\boldsymbol{b_s}} b_k P(y_\mu|\boldsymbol{b}) \prod_{l\neq k, l\in S(\mu)}\left(\frac{1+m_{\mu l}^t b_l}{2}\right) \sum_{\boldsymbol{b_w}} \prod_{l\in W(\mu)}\left(\frac{1+m_{\mu l}^t b_l}{2}\right)}{\sum_{\boldsymbol{b_s}} P(y_\mu|\boldsymbol{b}) \prod_{l\neq k, l\in S(\mu)}\left(\frac{1+m_{\mu l}^t b_l}{2}\right) \sum_{\boldsymbol{b_w}} \prod_{l\in W(\mu)}\left(\frac{1+m_{\mu l}^t b_l}{2}\right)} \tag{A.4}$$

$$= \frac{\sum_{\boldsymbol{b_s}} b_k P(y_\mu|\boldsymbol{b}) \prod_{l\in S(\mu)}\left(\frac{1+m_{\mu l}^t b_l}{2}\right)}{\sum_{\boldsymbol{b_s}} P(y_\mu|\boldsymbol{b}) \prod_{l\in S(\mu)}\left(\frac{1+m_{\mu l}^t b_l}{2}\right)} \tag{A.5}$$

A.2 Weak case : k is weakly connected

One of the main difference with the first case, is that we don't consider Δ_μ. Instead we remove one term, the one depending on b_k as it was done with the CDMA signal.

We now consider $\Delta_{\mu_k} = \frac{1}{\sqrt{K}}\sum_{l=1,l\neq k}^K s_{\mu l} b_l + \frac{1}{\sqrt{J}}\sum_{j=1}^J s_{\mu j} b_j$, and using the central limit theorem: $\Delta_{\mu k}$ obeys a Gaussian distribution $\mathcal{N}\left(\left\langle\Delta_{\mu k}^t\right\rangle_\mu, (1-Q_{\mu k}^t)\right)$, where

$$\left\langle\Delta_{\mu k}^t\right\rangle_\mu = \frac{1}{\sqrt{K}}\sum_{l,l\neq k} s_{\mu l} m_{\mu l}^t + \frac{1}{\sqrt{J}}\sum_{j=1}^J s_{\mu j} b_j \tag{A.6}$$

and

$$Q^t_{\mu k} = (1/K)\sum_l (m^t_{\mu l})^2. \tag{A.7}$$

The conditional probability which was written previously as

$$P(y_\mu|\boldsymbol{b}) = \frac{1}{\sqrt{2\pi\sigma_0^2}} \exp\left[-\frac{1}{2\sigma_0^2}\left(y_\mu - \Delta_\mu\right)^2\right]$$

is now written using $\Delta_{\mu k}$

$$P(y_\mu|\boldsymbol{b}) = \frac{1}{\sqrt{2\pi\sigma_0^2}} \exp\left[-\frac{1}{2\sigma_0^2}\left(y_\mu - \Delta_{\mu k} - s_{\mu k}b_k/\sqrt{K}\right)^2\right].$$

And since $s_{\mu k}b_k/\sqrt{N}$ is small for large N, we expand the conditional probability as

$$\begin{aligned} P(y_\mu|\boldsymbol{b}) &\simeq \frac{1}{\sqrt{2\pi\sigma_0^2}} \exp\left[-\frac{\left(y_\mu - \Delta_{\mu k}\right)^2}{2\sigma_0^2} + \frac{s_{\mu k}(y_\mu - \Delta_{\mu k})}{\sqrt{K}\sigma_0^2}b_k\right] \\ &\simeq \frac{1}{\sqrt{2\pi\sigma_0^2}} \exp\left[-\frac{\left(y_\mu - \Delta_{\mu k}\right)^2}{2\sigma_0^2}\right]\left(1 + \frac{s_{\mu k}(y_\mu - \Delta_{\mu k})}{\sqrt{K}\sigma_0^2}b_k\right), \end{aligned}$$

The fact, that it now depends linearly on b_k is very important. That is why lot of terms in the sum will be removed due to two properties. The first one is that the sum over b_k of a quantity which doesn't depend on it is 0, and the other is that the product $b_k b_k = 1$.

As we know the probability density function of $\Delta_{\mu k}$ is

$$P(y_\mu|\boldsymbol{b}) = \int d\Delta_{\mu k} P(\Delta_{\mu k}) P(y_\mu|\Delta_{\mu k}).$$

Then it is expanded to provide

$$\begin{aligned} P(y_\mu|\boldsymbol{b}) = \int d\Delta_{\mu k} \;\; & \tfrac{1}{\sqrt{2\pi\sigma_0^2}} \exp\left[-\tfrac{\left(y_\mu - \Delta_{\mu k}\right)^2}{2\sigma_0^2}\right]\left(1 + \tfrac{s_{\mu k}(y_\mu - \Delta_{\mu k})}{\sqrt{K}\sigma_0^2}b_k\right) \\ & \tfrac{1}{\sqrt{2\pi(1-Q^t)}} \exp\left[-\tfrac{\left(\Delta_{\mu k} - \langle\Delta_{\mu k}\rangle\right)^2}{2(1-Q^t)}\right], \end{aligned}$$

So the final result is

$$P(y_\mu|\boldsymbol{b}) = \left[1 + \frac{\left[y_\mu - \langle\Delta_{\mu k}\rangle\right] s_{\mu k}b_k}{\left[\sigma_0^2 + 1 - Q^t\right]\sqrt{K}}\right] \frac{1}{\sqrt{2\pi(\sigma_0^2 + 1 - Q^t)}} \exp\left[-\frac{\left(y_\mu - \langle\Delta_{\mu k}\rangle\right)^2}{2\left(\sigma_0^2 + 1 - Q^t\right)}\right]$$

Back to the message with :

$$A = \frac{\left[y_\mu - \langle\Delta_{\mu k}\rangle\right] s_{\mu k}}{\left[\sigma_0^2 + 1 - Q^t\right]\sqrt{K}}$$

and

$$B = \frac{1}{\sqrt{2\pi(\sigma_0^2 + 1 - Q^t)}} \exp\left[-\frac{(y_\mu - \langle \Delta_{\mu k} \rangle)^2}{2\left(\sigma_0^2 + 1 - Q^t\right)}\right]$$

then :

$$\hat{m}_{\mu k}^{t+1} = \frac{\sum_{\boldsymbol{b_s}} b_k P(y_\mu|\boldsymbol{b}) \prod_{l \in S(\mu)} \left(\frac{1+m_{\mu l}^t b_l}{2}\right)}{\sum_{\boldsymbol{b_s}} P(y_\mu|\boldsymbol{b}) \prod_{l \in S(\mu)} \left(\frac{1+m_{\mu l}^t b_l}{2}\right)} \tag{A.8}$$

$$= \frac{\sum_{\boldsymbol{b_s}} A.B \prod_{l \in S(\mu)} \left(\frac{1+m_{\mu l}^t b_l}{2}\right)}{\sum_{\boldsymbol{b_s}} B \prod_{l \in S(\mu)} \left(\frac{1+m_{\mu l}^t b_l}{2}\right)} \tag{A.9}$$

$$\tag{A.10}$$

If we evaluate the complexity of the found formula, we can see that there is still a sum over $\boldsymbol{b_s}$, as in the sparse case.

Bibliography

[1] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of plausible Inference.* Morgan Faufmann, San Mateo, 1988.

[2] F. Jensen. *An introduction to Bayesian Networks.* UCL Press, 1996.

[3] Michael R. Garey and David S. Johnson. *Computers and Intractability.* Freeman, 1979.

[4] I. Reed and G. Solomon. *Polynomial Codes over Certain Finite Fields.* SIAM J. Appl. Math., pp. 300–304, 1960.

[5] P. Thitimajshima C. Berrou, A. Glavieux. *Near Shannon Limit error-correcting coding and decoding: Turbo-codes.* Proc. ICC'93, pp.1064–1070, 1993.

[6] R. G. Gallager. *Low Density Parity-Check Codes.* MIT Press, Cambridge, MA, 1963.

[7] R. Vicente. *Low Density Parity Check Code - A Statistical Physics Prospective.* Advances in Imaging and Electron Physics. 125. Elsevier Science, 2002.

[8] D. J. C. MacKay and Radford M. Neal. *Good Codes based on Very Sparse Matrices.* Cryptography and Coding - LNCS 1025, 1995.

[9] Yoshiyuki Kabashima. *A CDMA multiuser detection algorithm on the basis of belief propagation.* J. Phys. A **36** page 11111, 2003.