Progress Report
4-20 June 2011

Raj K. Kumar, Amir Hesam Salavati
E-mail: raj.kumar@epfl.ch, hesam.salavati@epfl.ch

Supervisor: Prof. Amin Shokrollahi
E-mail: amin.shokrollahi@epfl.ch

Algorithmics Laboratory (ALGO)
Ecole Polytechnique Federale de Lausanne (EPFL)

September 20, 2011

1 Summary

Expander codes are closely related to our neural constraint enforcing algorithm as expander graphs and bit flipping algorithms are used in both cases. Therefore, we read a bit more about applications of expander graphs in coding methods. The considered papers were interesting in the sense that they provide some nice mathematical background which could be helpful in finding suitable expander graphs for our purposes and also help us in obtaining theoretical bounds on the number of correctable errors using the suggested method.

2 Expander Codes

In [2] the authors propose a coding techniques that achieves relatively good performances with linear time decoding. The codes are based on expander graphs and, hence, are called *expander codes*. The authors use a (c, d)-regular bipartite graph in which c and d are kept constant as the size of the graph increases. This graph is interpreted as the parity check graph of a low density parity check code. An expander code if formally defined as follows:

Definition 1. *Let B be a (c, d)-regular bipartite graph between a set of variable nodes $\{V_1, \ldots, V_n\}$ and constraint codes $\{C_1, \ldots, C_m\}$, where $m = cn/d$. Further, let $b(i, j)$ be a function such that for each constraint node C_i, the set of its neighbors is given by $V_{b(i,1)}, \ldots, V_{b(i,d)}$. Now let $\mathcal{S}$ be an error correcting code* ***with block length*** *d (instead on n). Then, $C(B, \mathcal{S})$ is an* expander code *with* block length *n if for all $1 \leq i \leq m$, $(x_{b(i,1)}, \ldots, x_{b(i,d)})$ is a codeword of $\mathcal{S}$.*

Note that the above definition means that each constraint imposes $(1 - r)d$ linear restrictions on a subset of size d of the codewords of $C(B, \mathcal{S})$. Because each such subset is a codeword in $\mathcal{S}$. Therefore, the graph B is not necessarily the parity check decoding graph. In other words, **the adjacency matrix of B is not necessarily the same as the parity check matrix of the code** $C(B, \mathcal{S})$. For such expander codes, a constraint node C_i is set to be satisfied if its neighboring variable nodes are a codeword of $\mathcal{S}$.

The following theorem gives us some bounds on the rate and minimum distance of an expander code:

Theorem 1. *Let B be a $(c, d, \alpha, \frac{c}{7}\epsilon d)$ expander and $\mathcal{S}$ an error correcting code of* block length *d,* ***rate*** *$r > \frac{c-1}{c}$ and minimum distance ϵd. Then, $C(B, \mathcal{S})$ has rate* at least *$cr - (c - 1)$ and minimum distance αn.*

The general *parallel* decoding algorithm is as follows, but one is extremely advised to take a look at a simple example given further down before going through this algorithm.

1. "For each constraint, if the variables in that constraint differ from a codeword of $\mathcal{S}$ in at most $d\epsilon/4$ places, then send a flip message to each variable that differs".

2. "In parallel, flip the value of each variable that receives *at least one* flip message".

Note that the algorithm requires to send different messages over different links. **Therefore, it is not suitable for our neural case in its current format**.

The theorem below shows the existence of expander codes and its convergence:

Theorem 2. *For all ϵ such that $1 - 2H(\epsilon) > 0$, where $H(\epsilon)$ is the binary entropy function, there exists a polynomial-time constructible family of expander codes of rate $1 - 2H(\epsilon)$ and minimum distance arbitrarily close to $\epsilon^2 n$ in which any αn errors can be corrected if $\alpha < \epsilon^2/48$.*

A special case that the adjacency matrix of B is the same as the parity check matrix of the code $C(B, \mathcal{S})$ is the case of $\mathcal{S}$ being an even parity check code, i.e. the modulo 2 sum of the d bits attached to each constraint should be equal to zero. For this code, the following simple decoding algorithm is propose:

1. "If there is a variable that is in more unsatisfied than satisfied constraints, then flip the value of that variable".

2. "Repeat until no such variable remains."

The authors prove that the above algorithm corrects $\alpha n/2$ of errors in the input for a $(c, d, \alpha, 3c/4)$ expander graph.

In [1], the authors use expander-based arguments to prove that the message passing algorithms are able to correct a linear number of erroneous symbols for an irregular low density parity check code. Note that this is a general argument and applies to a number of message passing algorithms including those of Gallager's. The main goal of this paper is to show that in order to correct a linear number of input errors, it is not necessary to do message passing first and then switch to Sipser and Spielman's expander decoding method [2] as the same expansion argument also applies to the message passing decoding algorithm. Theorem 1 in the paper proves that the Gallager's hard decision decoding approach is capable of decoding a linear number of errors in the input if the LDPC graph is a $(\gamma, \delta, \rho, \lambda)$ expander. More specifically, given the expansion properties and with $\beta \geq 1/2$, Gallager's algorithm is capable of correcting *nearly* θn errors where $\theta = \frac{\delta - b\hat{e}ta}{(1-b\hat{e}ta)l_{max}}\gamma$ and $b\hat{e}ta = \beta + \frac{1-\beta}{l_{min}}$. Theorem 3 proves the same result for the belief propagation method (Gallager's soft decoding algorithm) with the same number of correctable error bits.

3 Conclusions and Future Works

The above two papers yield some nice theoretical results on expander codes and decoding algorithms. In their current format, these papers may help us obtain nice theoretical bounds on the number of correctable errors in our neural error correcting algorithm. More specifically, theorem 7 in [2] seems to be exactly the same situation in our neural work. Because theorem 7 essentially provides concrete bounds on the codes minimum distance, which will give us the number of correctable errors. The only difference with our work is that here, the authors assume the codeword of length n is obtained by shuffling a number of codewords of length d, where d is the degree of constraint nodes. There is of course a limitation on the rate of the code $\mathcal{S}$ as we must have $r > \frac{c-1}{c}$ but this simply means it is not necessary to have a lot of redundancy in this code and all decoding is taken care of in the expander graph.

Furthermore, proof of theorem 10 might end up useful in our case to find tighter bounds on the number of correctable errors. In this proof, first the authors show that as long as the initial number of errors is less than $\alpha n/2$, then there is a variable node (corrupted or uncorrupted) with more than half of its neighbors unsatisfied, so it will get updated by the algorithm. This means that the number o unsatisfied constraints monotonically reduce in each iteration. Therefore, the only

way the algorithm can fail is that in some iteration, the number of corrupted nodes become larger than the expansion criteria, i.e. αn. This may only happen if the algorithm incorrectly updates so many *uncorrupted* nodes. However, the authors show that this can never happen as the number of corrupted nodes must reach the value αn, which consequently means the number of unsatisfied constraints must be greater than $c\alpha n/2$ (see equation (1)). This is a contradiction to the fact that the initial number of errors was at most $\alpha n/2$ because with that many errors, the maximum number of initial unsatisfied constraints is $c\alpha n/2$ and this number decreases in each iteration.

References

[1] D. Burshtein, G. Miller, *Expander graph arguments for message passing algorithms*, IEEE Trans. Inf. Theory, Vol. 47, No. 2, 2001, pp. 782-790.

[2] M. Sipser and D. Spielman, *Expander codes*, IEEE Trans. Inform. Theory, Vol. 42, No. 6,1996, pp. 1710-1722.