UNIVERSITÀ DI PARMA

Department of Engineering and Architecture
Master of Science in Communication Engineering
(Laurea Magistrale)

# Exploring Machine Learning Algorithms for Decoding Linear Block Codes

Advisor:
Prof. Riccardo Raheli
Co-Advisors:
Prof. Henry D. Pfister
Dr. Christian Häger
Dr. Marco Martalò

Thesis presented by:
Fabrizio Carpi

Academic Year 2017-2018

---

**Abstract**

In recent years, machine learning has become one of the fastest growing areas in many technological fields. In general, it provides to a computer the capability to learn from data, without being explicitly programmed for a specific task. In this thesis, we investigate the use of machine-learning-inspired decoding algorithms for linear block codes, such as Reed-Muller (RM) codes. We consider the recently proposed RM codes with Parity Check (PC) matrix $\boldsymbol{H}$ composed of minimum-weight parity checks. All the experiments are performed in Python and Tensorflow.

In the first part of this thesis, the goal is to optimize Belief Propagation (BP) iterative decoding through supervised learning. The BP computation graph can be parameterized introducing coefficients that scale the messages passed within the BP iterations. Recent work has shown that these coefficients can be optimized, with machine learning methods, to improve the decoding performance. The number of parameters may be huge, on the order of the number of edges of the Tanner graph. We propose an optimized decoder which has a number of parameters on the order of the number of BP iterations. This solution, which adds low complexity to the decoder, obtains improved performance with respect to standard BP decoding. The experiments for this part of the thesis are performed on the Binary-Input Additive White Gaussian Noise (BI-AWGN) channel.

In the second part of this thesis, we propose a novel Reinforcement Learning (RL)-inspired method for Bit Flipping (BF) decoding. We approach the decoding problem as a game, exploiting the interesting results achieved in literature for deep RL architectures applied to games and robotics. One can view the RL decoder as a player of a game where the goal is to correct the error pattern. At first, the RL architecture has no prior knowledge about the code, but it learns to act by trial and error. In particular, our results show that standard BF decoding can be outperformed in some specific scenarios. The experiments of this part of the thesis are performed on the Binary Symmetric Channel (BSC).

## Belief-Propagation Decoding Optimization with Supervised Learning

The key idea is that the feed-forward unrolled BP iterations can be interpreted as a deep Neural Network (NN), where the neurons are Variable Nodes (VNs) and Check Nodes (CNs), whereas the connections are drawn by the code PC matrix $\boldsymbol{H}$. We introduce two scaling schemes to parameterize the BP decoder. First, we apply one weight per iteration, which scales all messages from CNs to VNs, so that every edge for a given iteration share the same optimized weight. Second, we introduce a damping coefficient among successive iterations, which mitigates the message updates oscillations and improves convergence. As an example, Figure **(a)** shows the block diagram for the parameterized decoder for 3 BP iterations. Then, the weights and the damping coefficient are optimized through supervised learning.

Since RM codes are linear, the all-zero codeword is included in the codebook. Without loss of generality, we can always transmit the all-zero codeword, because the BP performance is independent of the transmitted

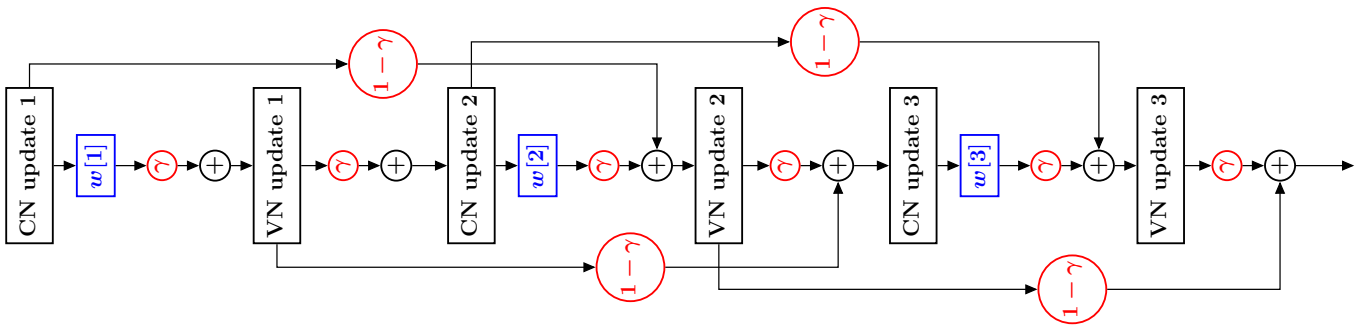
codeword for a binary memoryless symmetric channel. Then, we can build our training set where the true labels are always the all-zero codeword $\mathbf{0}$ and the received sequences $\boldsymbol{y} = x(\mathbf{0}) + \boldsymbol{n}$ are the noisy samples, where $x(\mathbf{0})$ is the modulated signal and $\boldsymbol{n}$ is the AWGN. Hence, the training set is $\mathcal{D} = \{(x(\mathbf{0}), \boldsymbol{y}^i)\}_{i=1}^{K}$, where $K$ is the number of data points.

We propose a loss function defined as $\ell(\boldsymbol{L}) = \sum_{j=1}^{n} [1 - \tanh(L_j)]$, where $\boldsymbol{L} = (L_1, ..., L_n)$ is the vector of marginal log-likelihood ratios at the end of the BP iterations, for a code with length $n$. Since the all-zero codeword is always transmitted, this loss is proportional to the bit error rate, which is minimized to achieve better performance. To this end, Stochastic Gradient Descent (SGD) is applied to minimize the loss function with respect to the NN parameters, i.e., the weights and the damping coefficient introduced above. After learning, our optimized BP decoder obtains a gain from 0.3 to 0.8 dB with respect to the standard BP decoding, and a penalty from 1 to 0.25 dB compared with a quasi-maximum-likelihood decoder.

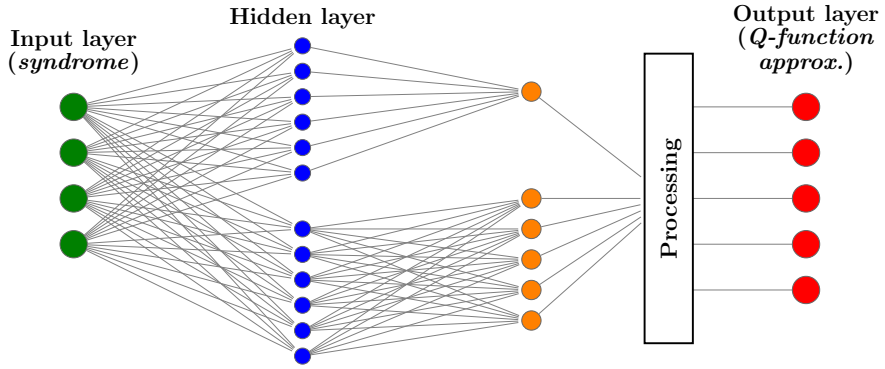
### Learning Bit-Flipping Decoding with Reinforcement Learning

RL is the branch of machine learning that is concerned with making sequences of decisions. RL consists of mapping the state of an environment into an action taken by an agent that moves within it. This environment is characterized as a Markov Decision Process (MDP). For every action, the environment provides a feedback to the agent in terms of reward. The sequence of chosen actions has to maximize the cumulative reward. We propose a scheme where the decoding problem is modeled as an MDP. The state of this MDP is the syndrome of the received sequence and the possible action is to flip one of the bits, as in standard BF decoding. The environment returns to the agent a high reward when it correctly decodes a codeword. We adopt the Q-learning algorithm, which is a form of model-free RL, for the training process.

We use a NN to approximate a Q-function, which can be interpreted as the quality function that expresses the expected future reward for an action, given a starting state. An example of NN architecture is shown in Figure (b). Then, the agent uses the NN output to choose the most promising action for a given state. The agent starts with no previous knowledge about the environment and/or the code structure. This approach allows us to train agents that potentially could learn to decode any linear block code, given a sufficient number of training epochs and well-shaped rewards. After training, the RL agent is able to decode with the same performance as the standard BF algorithm. Furthermore, for some scenarios, e.g., using small PC matrices, the RL agent may perform better than the standard BF algorithm. Our intuition is that the agent is able to see a higher-dimensional representation of the syndrome, exploiting the hidden layers of the NN.




(a) Unrolled Belief-Propagation parameterized iterations with weights (blue) and damping coefficient (red).




(b) Neural Network architecture for the Q-function approximation.