Mixed Integer Linear Programming for Optimizing a Hopfield Network

Bodo Rosenhahn[1]

[1]Institut für Informationsverarbeitung, Leibniz University Hannover, Germany,
rosenhahn@tnt.uni-hannover.de

Abstract. This work presents an approach to optimize the weights of a discrete Hopfield network as mixed integer linear program (MILP). As the original formulation involves a sign-function, it is not differentiable, but parameter optimization using a (mixed integer) LP is possible. As autoassociative memory, a key question is the amount of patterns which can be stored in such a Hopfield network. In this work it is shown, that the traditional storage description models are far inferior to a globally optimized solution which can be obtained with a MILP. In contrast to a gradient descent based optimization is the proposed approach nearly parameter free and independent from seeding and other factors which are crucial for differentiable programming. Additionally it is possible to enforce sparsity constraints on the weights. Such additional constraints improve the generalization of such a model and make the Hopfield network more stable for the case of outliers or missing values. Several experiments demonstrate the effectiveness of the model.

Keywords: Mixed Integer Linear Program, Hopfield network, sparse models

1 Introduction

The basic concepts of neural networks go back to the 40th and 50ths, e.g. in 1943, McCulloch and Pitts formulated their idea for logical calculus using concepts from nervous activities [27]. Based on these foundations, binary Hopfield networks were introduced as associative memories in 1982 [18]. It is a one-layer recurrent network that can store and retrieve patterns. In a d-dimensional space, the later explained *Information storage prescription* can store about $0.138d$ patterns [9], whereas the *Pseudoinverse rule* allows to store up to d patterns. The theoretic analysis in [38] estimates $2d$ as the upper limit of patterns which can be stored in a Hopfield model. It will turn out, that our proposed MILP solution can reach this upper limit. To the best of our knowledge, this is the first practical solution for estimating network weights providing maximal storage.

Nearly all existing neural network models are nowadays modified to differentiable neurons and commonly trained using gradient descent based optimization and auto differentiation. Such optimized systems can be summarized with the term *differentiable programming* [7]. Indeed, it is well known that the gradient

descent based optimization is prone to convergence problems, overfitting or getting stuck in local minima, so that many different (differential) approaches exist to overcome these issues, e.g. using special optimizers (adam, sgdm, etc.), dropout layers, etc. Note, that even though the Hopfield networks are an established topology, they received increasing attention in the past [32]. Please also note, that so-called *modern Hopfield Networks* consist of non-binary output variables which have a significantly higher capacity [2].

In this work, the non-differentiable Rosenblatt perceptron is revisited and used for modeling a Hopfield network. For the optimization of such a network, a mixed integer linear program is proposed. The optimized network weights can then be assembled to a Hopfield network and used for forward inference on unseen data. The formulation as MILP also allows the optimization of integer constraints and therefore the optimization of the network weights in presence of an exact step function. Additionally, integration of additional constraints on the network, e.g. to enforce sparsity, symmetry or binary constraints is possible. Such constraints allow for an optimization of a most efficient memory representation. The commonly used and later described *information storage prescription* or *pseudoinverse rule* to compute the weights of a Hopfield Network as well as differential programming can not fully exploit the storage capacities in contrast to the proposed MILP variants.

To summarize, this work presents the following **contributions**:

1. Formulation and optimization of a non-differentiable Hopfield network as mixed integer LP for given training data.
2. Formulation of additional constraints such as sparseness on weights, symmetry properties or binary network weights.
3. Evaluation of the optimized model with respect to storage capacities, stability with respect to outliers or missing data and experiments on image retrieval or classification.
4. Example code is available online[1].

In the following the general formulation of a Hopfield network as well as the classical storage prescription is described. Then, the basic concepts of mixed integer linear programming (MILP) are introduced and used to formulate such Hopfield networks.

Note, that in this work only discrete Hopfield networks with the output of $\{-1, 1\}$ are investigated. Still, on different datasets a competitive performance of the globally optimized Hopfield network is shown, even though the network is rather simple.

2 Foundations

2.1 The Hopfield Network

Introduced in 1982 [18], a Hopfield network consists of a one-layer recurrent network with the input dimension being equal to the output dimension. Its main

[1] http://www.tnt.uni-hannover.de/staff/rosenhahn/HopfieldNetExamples.zip

purpose is to act as autoassociative memory, e.g. for a given tiny sample, the memory should retrieve a piece of data. Hopfield networks can be applied in denoising or removing interference from an input or they can be used to determine whether the given input is *known* or *unknown*.

The units in Hopfield nets are binary threshold units $\in \{-1, 1\}$, the interactions $\omega_{i,j}$ between neurons are defined to be symmetric ($\omega_{i,j} = \omega_{j,i}$) and no unit has a connection with itself ($\omega_{i,i} = 0$). The classical form of a Hopfield network is shown in the left of figure 1.

Such a model can be based on the formulations of McCulloch and Pitts [27]. They formulated their idea for logical calculus using concepts from nervous activities. In the original formulation of McCulloch and Pitts, each neuron can emit two states $y_i \in \{0, 1\}$, thus it can *fire* or *not fire*. A neuron consists of n input lines on which the signals $(x_1 \dots x_n)$ are present. Calculation works as follows, the input signals $(x_1 \dots x_n)$ are added to the sum x. The sum x is compared with the threshold (or bias) b. If the sum of the excitations is greater than or equal to b, the neuron returns 1, otherwise it returns 0. In 1958, Frank Rosenblatt published his perceptron model which extends the summation to a scalar product, followed by a step function [34]. This is the basis for neural networks up till now. The perceptron can be summarized as

$$y_i = \begin{cases} 1 : \sum_j \omega_{ij} x_j + b_i > 0 \\ 0 : else \end{cases} \tag{1}$$

The bias value b corresponds to the decision threshold and ω_{ij} are learnable parameters. A combination of such perceptrons in a directed acyclic graph leads to a classic (e.g. fully connected) neural network.

Thus, a Hopfield network can be expressed as a fully connected recurrent layer with the amount of neurons being equal to the input dimension. The matrix of the network weights should be symmetric and form a hollow matrix. Note, that lifting the binary values $y \in \{0, 1\}$ to $y' \in \{-1, 1\}$ can be accomplished by the simple (linear) operation $y' = 2y - 1$.

Let $X = [x_1, \dots, x_n]$ be a matrix of size $m \times n$ containing m-dimensional vectors with binary values $\in \{-1, 1\}$. For the computation of the network weights, the *information storage prescription* proposed in [9] can be applied:

$$\omega = XX^T \tag{2}$$

$$\forall i = 1 \dots n : \quad \omega_{i,i} = 0 \tag{3}$$

$$b_i = 0 \tag{4}$$

Note, that ω is a symmetric matrix with zero entries on the diagonal. This formulation is the so-called $Baseline_1$ in the experiments.

An alternative to compute the weight factors is to use the so-called *pseudoinverse* or *projection rule* [31], which is defined as

$$W = XX^+ \tag{5}$$

The matrix X^+ denotes here the pseudoinverse. We call this method $Baseline_2$ in the experiments. Once the weights $\omega_{i,j}, b_j$ are computed, the Hopfield network

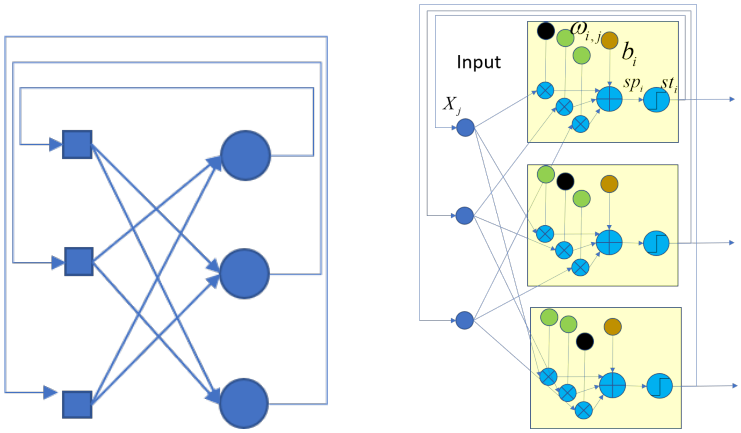

Fig. 1. Left: Visualization of a Hopfield Network. Right: Elements of a Hopfield Network as MILP. The black nodes indicate that these weights are set to zero.

can be evaluated as

$$x_i' = \begin{cases} 1 : \sum_j \omega_{i,j} x_j + b_j > 0 \\ -1 : else \end{cases} \tag{6}$$

Note, that b_j is set to zero for both baselines and later for the neural network and all MILP variants to allow for a fair comparison across the methods. As the output generates a vector with entries $\in \{-1, 1\}$, it can immediately be used again as input, until a convergence (e.g. fix point) is reached.

Hopfield nets have a scalar value associated with each state of the network, referred to as the *energy*, E, of the network, where

$$E = -\frac{1}{2} \sum_{i,j} \omega_{i,j} x_i x_j + \sum_i b_i x_i \tag{7}$$

The optimization of this energy formulation in combination with the involved step-function will be later used in the proposed mixed integer linear program.

The capacity of the Hopfield Network to store patterns depends on the used learning algorithm. The first learning rule applied to store patterns was Hebbian learning [18], which yields $Baseline_1$. Other rules have been formulated and considered in [1] and [12]. The implementation of Hopfield Networks in hardware and learning sparse networks have been considered in [36]. Hopfield neural networks have been applied for image processing [30, 17] or solving combinatorial problems [33, 20].

2.2 Mixed Integer Linear Programming

Linear programming (LP) is a method for the minimization (or maximization) of a linear objective function, subject to linear equality or inequality constraints

[11]. To summarize, any LP can be expressed as

$$\min\ c^T x \quad s.t.\ Ax \leq b \tag{8}$$

A standard approach for solving such a LP is the simplex algorithm. Note, that solvers also accept equality constraints of the form $A_{eq}x = b_{eq}$ which are directly transformed in two inequality constraints of the form $A_{eq}x \leq b_{eq}$ and $-A_{eq}x \leq -b_{eq}$. It is also possible to allow for the optimization of integer constraints [28, 13], which is then called a mixed integer linear program. The solvers are then using concepts such as branch-and-cut or branch-and-bound.

A (MI)LP is a very powerful tool, which allows the efficient optimization of graph problems, e.g. graph cut, graph matching, max-flow or network optimization [4, 23], logic inference [25] and even reasonable efficient optimization of NP-hard problems, such as the traveling salesman problem. The work [39] gives a comprehensive introduction to formulate logic calculus (and beyond) using mixed integer linear programs. It is also possible to optimize decision trees [40] or support vector machines [29] using respective LP formulations.

Neural networks and linear programming have been brought together in the context of network fooling and model checking [16], already trained networks can be analyzed using LPs [5]. The work [35] introduces an approach to remove unneccessary units and layers of a neural network while not changing its output using a MILP. The formulation of a linear threshold unit (LTU) as LP, which is similar to the perceptron desribed above, has been presented in [26]. Based on this formulation, several LTUs are iteratively connected using a so-called multi-surface method tree. The special case of binary neural networks using MILP has been presented in [22]. Interestingly, binarized neural networks (BNNs) which are neural networks with weights and activations in $(-1, 1)$ can gain comparable test performance to standard neural networks but allow for highly efficient implementations on resource limited systems [21]. In [37] a mixed integer linear program to optimize neural network weights is presented. This work also focusses on network compression whereas in Hopfield networks it is not possible to reduce the amount of neurons. Instead, in this work different sparsity constraints are proposed and evaluated.

There are two arguments why a MILP solution may be disadvantageous, as stated by [14] (a) scaling to large datasets is problematic, as the size of the generated equations scales with the size of the training set and (b) solutions with provable optimal training error can overfit, thus training data is perfectly explained, but test performance is much worse. One reason for (a) is, that all data, as well as the network computations, need to be stored simulatenously and expressed as equality and inequality constraints. One option to alleviate this is an iterative training procedure: It is possible to globally optimize for batches and to train the weights iteratively over a couple of epochs [37]. During each iteration it is required to add constraints which enforce a small distance to earlier computed weights. Since it is comparable to a gradient based optimization, this variant is omitted here and only small datasets which can be globally optimized are considered. In this work, the second challenge is adressed by the proposed sparsity constraints.

3 Modeling a Hopfield Network as MILP

The Hopfield network requires the computation of a neuron which yields to the computation of a scalar product, followed by a step function which can also be expressed as a *greater as*, $>$. The formulation for $>$ can be expressed as MILP using a so-called Big-M formulation. Note, that the letter M refers to a (sufficiently) large number associated with the artificial variables (see also [39]). The condition

$$\texttt{a > b} \leftrightarrow \delta \texttt{ =1}$$

can be formulated as

$$a \geq b + \epsilon - M(1 - \delta) \tag{9}$$
$$a \leq b + M\delta \tag{10}$$
$$\delta \in \{0, 1\} \tag{11}$$

The verification of these equations is straight forward:

$$\begin{aligned} a > b \rightarrow \quad & a \geq b + \epsilon - M(1 - \delta) \quad \rightarrow \delta = 1 \vee 0 \\ \text{and} \quad & a \leq b + M\delta \quad \rightarrow \delta = 1 \end{aligned} \tag{12}$$

$$\begin{aligned} a \leq b \rightarrow \quad & a \geq b + \epsilon - M(1 - \delta) \quad \rightarrow \delta = 0 \\ \text{and} \quad & a \leq b + M\delta \quad \rightarrow \delta = 0 \vee 1 \end{aligned} \tag{13}$$

Based on equation (1), the scalar product and the $>$ formulation it is possible to model the activation of a perceptron with weights ω, bias b and input x. Please note, that the outcome is a binary vector which is later denoted as slack variable st. This output can then be uplifted to $\{-1, 1\}$ and used for energy minimization.

Let $X = [x_1, \ldots, x_n]$ be a set of m-dimensional vectors with binary values $\in \{-1, 1\}$. The parameter I indicates the example of the dataset and i the index of a neuron. Note, that the amount of neurons corresponds to the input

dimension. Then the MILP looks as follows:

$$\min f^T x \quad s.t. \tag{14}$$

$$\forall I = 1 \dots n, i = 1 \dots m$$

$$\sum_{j=1}^{m} \omega_{ij} x_{j,I} + b_i - sp_{I,i} = 0 \tag{15}$$

$$\epsilon - M(1 - st_{I,i}) - sp_{I,i} \leq 0 \tag{16}$$

$$sp_{I,i} - M st_{I,i} \leq 0 \tag{17}$$

$$2st_{I,i} - \delta_{I,i} = 1 \tag{18}$$

$$\sum_{j=1}^{m} \frac{1}{2} \delta_{I,j} x_{j,I} - E_I = 0 \tag{19}$$

$$\forall j = 1 \dots m : \omega_{j,j} = 0 \tag{20}$$

$$\omega_{i,j} - \omega_{j,i} = 0 \tag{21}$$

$$st_{I,i} \in \{0, 1\} \tag{22}$$

$$f(ind(E_I)) = -1 \tag{23}$$

The slack variables *sp* encode results of scalar products Eq. (15), *st* encode the evaluation of the step function Eqs. (16-17), δ the lifting from $\{0, 1\}$ to $\{-1, 1\}$ Eq. (18) and E Eq. (19) the energy to optimize. Equations (20-21) enforce the symmetry properties and the 0-entries on the diagonal matrix. The vector f contains only zero entries and only a factor -1 at the positions of the slack variables containing the loss E_I, Eq. (23). All used variables can be ordered as a large vector and acessed via an index-operation $ind(.)$. Thus, minimizing the objective function $f^T x$ means to minimize the loss. Note, that the variables $\omega_{i,i}$ can also be removed from the system but they are kept for readability.

The equations lead to sparse vector expressions and describe a mixed integer linear program which can be optimized with standard methods [15].

3.1 Sparsity Constraints

As mentioned in [14], the MILP solution can be problematic for generalization as no margin or other factor is optimized jointly with the optimization function. Motivated from other works, such as [3, 6, 10, 8] sparsity can be beneficial for generalization and fortunately it is easy to enforce during the MILP optimization. In the following, two variants are proposed, one is enforcing the sparsity integer weights e.g. $\in [-1, 1]$, whereas the other variant can be applied to images. It is explicitly exploiting the grid-structure and enforcing only weights in the neighborhood. For the first variant, the MILP-constraints from the earlier section need to be extended with

$$\forall i, j = 1 \dots m : \quad \omega_{i,j} - \omega^B_{i,j} \leq 0 \tag{24}$$

$$-\omega_{i,j} - \omega^B_{i,j} \leq 0 \tag{25}$$

$$f(\omega^B_{i,j}) = 0.001 \tag{26}$$

The additional slack variable $\omega^B_{i,j}$ contains the absolute value of the weight $\omega_{i,j}$. The vector f is modified with a small penalizer (with a scale factor of 0.001) to enforce the sparsity. Minimizing $\omega^B_{i,j}$ means to maximize the amount of zero-entries in $\omega_{i,j}$.

In the case of images or data with a clear topological structure it is also possible to enforce only connectivity of weights in a close neighborhood. This is known as local receptive field in image processing. The constraints simply enforce all weights outside a specific local distance d to be zero:

$$\forall i,j = 1 \dots m: \quad \omega_{i,j} = 0 \leftrightarrow dist(i,j) > d \tag{27}$$

Note, that these variables can also be completely removed from the MILP. For the experiments on image data (e.g. mnist examples) the pixel distance d has been set to the value $d = 3$. The function $dist(i,j)$ is defined as the Euclidean distance of the point positions.

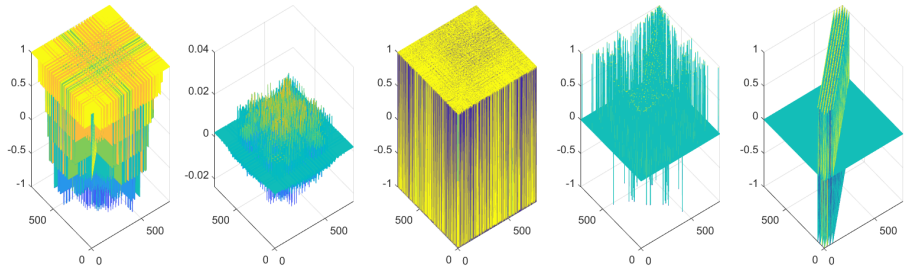

Fig. 2. Obtained weight-matrices ω for the different proposed variants. (From left to right): (a) Baseline ω_{B1}, (b) Baseline ω_{B2}, (c) MILP-optimized weights ω, (d) MILP-optimized weights with Sparsity constraints ω_{SP} (e) MILP-optimized weights as local receptive field ω_{LRF}

Figure 2 visualizes the outcome of different weight matrices obtained from the five presented variants, from left to right: (a) Baseline1 ω_{B1} (Eq. (2)), (b) Baseline ω_{B2} (Eq. (5)), (c) MILP-optimized weights ω (Eqs. (14-23)), (d) MILP-optimized weights with Sparsity constraints ω_{SP} (Eqs. (14-23 + 24-26)) (e) MILP-optimized weights as local receptive field (LRF) ω_{LRF} (Eqs. (14-23 + 27)). As dataset examples from mnist have been used. The dataset contains images of size 28×28 which are reordered as a 784 dimensional vector. Whereas the Baseline and the MILP optimized weight matrices are fully occupied, the sparse and the LRF variant have a completely different shape on locality and sparseness. The visualization shows that the constraints work exactly as intended.

4 Experiments

Two main research questions are analyzed in the following experiments: (**Q1**) What is the maximum storage capacity which can be achieved with a MILP

optimized Hopfield network? (**Q2**) How does the MILP optimized Hopfield network perform for applications such as classification under missing data? In the following respective experiments on the MILP optimized Hopfield networks are presented to answer the raised research questions, to analyze the storage capacities and the stability with respect to noisy data. Afterwards we use the Hopfield networks for classification and compare the generalization and the compensation of missing values on different datasets. Here, the focus is specifically on learning with only a few examples.

For the experiments, matlab2021(b) has been used to generate the (in)equality constraints, boundary conditions and the objective function. For optimization itself, gurobi v9.1.2 has been used. The timelimit has been set to three hours and the standard optimization parameters (MIPFocus, Presolve, etc.) have not been changed. The computation has been performed on 10 physical cores on a local linux computer.

4.1 Storage Capacities

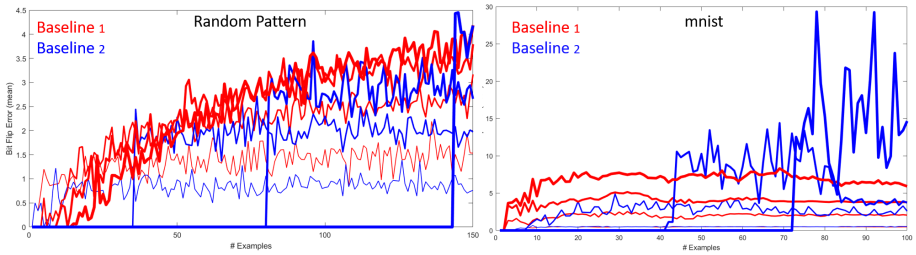

Fig. 3. Capacities for the two baseline methods with increasing amount of neurons (indicated by the line thickness). The x-axis shows the amount of training data to store. The y-axis shows the average amount of bitflips as error measure ($\frac{1}{N}\sum_{i=1}^{N} sum(x_i \neq x_i')$). The left image shows the capacities for storing random patterns and the right image the capacities for the same dimensions on rescaled mnist data. As the mnist data is highly correlated, the storage capacities decrease significantly. The red line shows the memory capacities of $Baseline_1$ and the blue line the capacities of $Baseline_2$. For random patterns, $Baseline_2$ can store $N-1$ patterns successfully. The largest model consists of 144 neurons and manages to store exactly 143 samples when using $Baseline_2$. Afterwards the model completely fails. For mnist data, for 144 given neurons the $Baseline_2$ only manages to store approx. 73 data samples.

In the first experiment, the general capacities of the baseline methods for an increasing amount of neurons are analyzed. As stated in the introduction, the Hopfield Network can store up to N uncorrelated patterns. To verify this, we generate random binary patches of sizes $3 \times 3, 6 \times 6, 9 \times 9, 12 \times 12$ and apply both baselines on a different amount of training images. Figure 3 shows in the left the outcome for storing random (uncorrelated) patterns, whereas the right

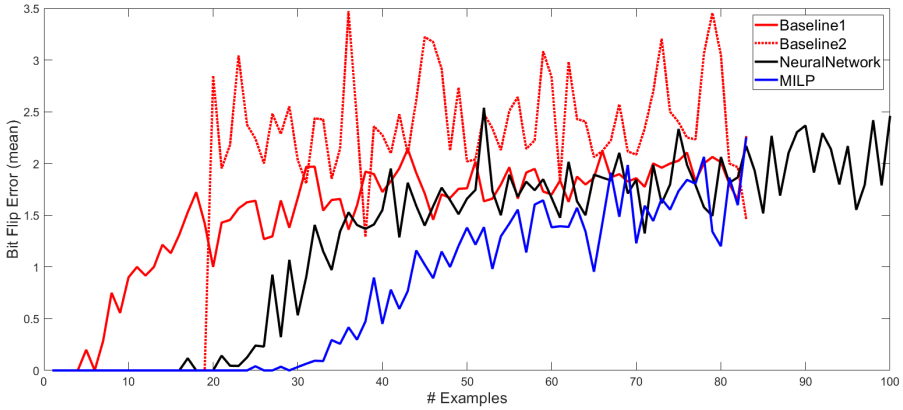

Fig. 4. Capacities for the two baseline methods, a shallow neural network and the MILP solution. The amount of neurons is set to 20. As observed before, $Baseline_1$ (red) can only store a few random patterns. $Baseline_2$ (red dashed) exactly 19 (and then it fails), the neural network (black) slightly more ($\sim$ 25), whereas the MILP version can go up to around 35 elements.

diagram shows the outcome on mnist data [24] of similar size. The x-axis denotes the amount of example images ranging from 1 to 150 which are independently used for training. Figure 3 (left) shows that the model can capture exactly $N-1$ (random, uncorrelated) patterns with $Baseline_2$. The storage capacities decrease significantly for mnist data (right). For example on the image size $12 \times 12 = 144$ only around 72 images can be memorized effectively. This effect is one of the reasons, why Hopfield and colleagues also discussed the concept of unlearning in Hopfield networks [19].

The work [38] estimates an upper bound of $2N$ as storage capacity of a shallow neural network with N neurons. Furthermore, the author notes that the result is not tied to any particular algorithmic formulation of neural networks. Figure 4 shows the storage capacities for 20 memory elements on both baselines, a neural network implementation and the MILP optimized model for random patterns. The outcome for the baselines is in line with the outcome of figure 3 (left), for 20 neurons, exactly 19 patterns can be stored with $Baseline_2$. The neural network is a shallow network with 20 neurons and a tanh activation function. It is optimized with classical back propagation using the sgdm optimizer (100 epochs, InitialLearnRate 0.001, MiniBatchSize 5). Here, the symmetry of the weight matrix, as well as the zero entries on the diagonal matrix have been enforced during optimization. This has been achieved by using weight sharing and by setting the diagonal entries and its learning rate to zero. The neural network performs better than the baselines. In contrast to these approaches, the MILP-Model performs superior. One reason is, that the MILP can take all information simultaneously into account for an optimal arrangement of the weights. As the patterns are based on random samples, locality and sparsity have been

omitted for this experiment. Note, that the MILP-optimized network is getting very close to the upper bound of $2N$ which has been theoretically proven nearly 40 years ago [38]. Please note, that a MILP-optimized Hopfield network can generate a global optimal solution, thus the storage capacities are maximized in the proposed formulation.

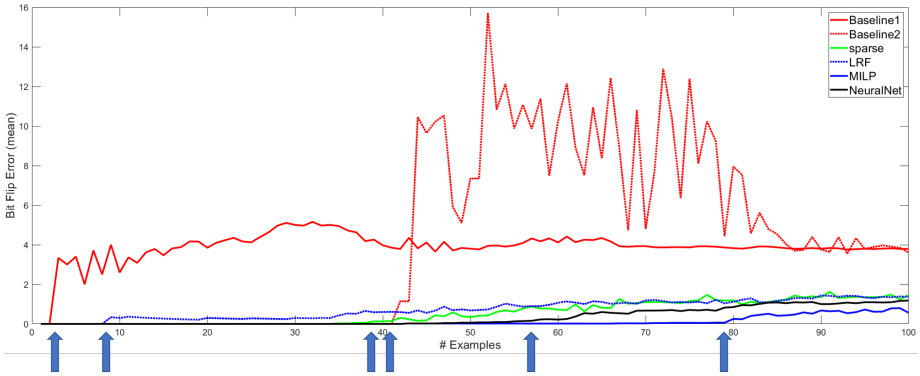

Fig. 5. Capacities of the Hopfield Networks on mnist examples for the two Baselines, a neural network, as well as the standard MILP, the MILP with Sparsity constraints and the MILP with LRF.

In the following we perform a similar analysis on the mnist dataset and summarize the outcome in figure 5: For the first experiment, mnist images have been downscaled to the size 9×9 pixels resulting in 81 neurons for memorization. Then the Hopfield network weights for the two baselines, an iteratively trained neural network (as before) as well as the standard MILP, the MILP with Sparsity constraints and the MILP with LRF have been trained for an increasing amount of data points. Afterwards, the memorization for each training sample is recalled and the average amount of bit flips is used as error metric, similar as before. Whereas the $Baseline_1$ can only memorize a few images, does $Baseline_2$ perform much better for a small amount of data. $Baseline_2$ manages to store around 40 images. Once a certain limit has been reached, $Baseline_2$ starts to fail completley, as described before. The neural netork performs better and can store approx. 60 patterns before errors are introduced. The MILP can store the most amount of patterns, around 80. Additionally, the storage capacities of the sparse variants are shown. The overall capacities are (slightly) smaller, as many entries are enforced to be zero. Especially the LRF variant has significantly less parameters available for optimization which causes a slight error reasonable early. For an increasing amount of examples, this error is getting similar to the other variants (including the neural network).

Figure 6 shows the amount of zero-entries of the computed weight matrices in percent. As can be seen, the Neural network, the standard MILP and the baselines make use of the full matrix and $Baseline_2$ degenerates after 40 samples.

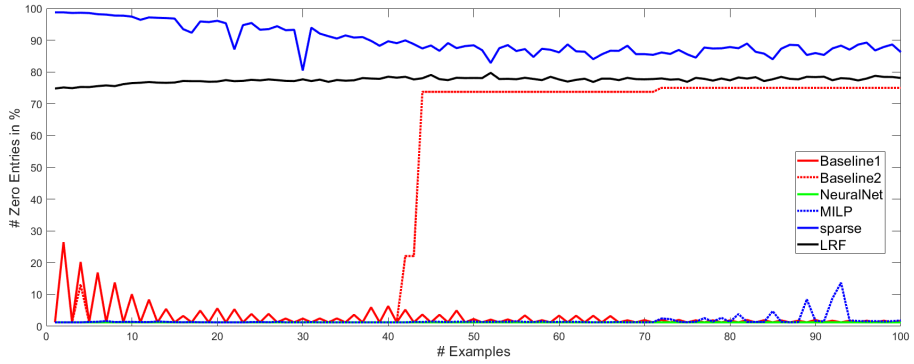

Fig. 6. Sparsity (y-axis) of the Hopfield Networks for different amounts of mnist examples (x-axis) for the two Baselines, a neural network, as well as the standard MILP, the MILP with Sparsity constraints and the MILP with LRF.

Only the sparse and the LRF variant heavily reduce the amount of entries while maintaining a competitive performance.

To summarize, all MILP solutions perform competitive, especially for a larger amount of data to memorize and also the error increase behaves more linear and incremental. Note, that the MILP provides a global optimal solution for the network weights, thus a better performance in memorization is not possible. Compared to $Baseline_2$, the amount of stored data (figure 5) is nearly doubled which is consistent with figure 4.

Figure 7 shows memorization results in presence of noise. The first row contains the input image with increasing noise (starting form 0). The other rows show the results using (row 2) the Baseline 1 ω_B (Equation (2)), (row 3) the Baseline 2 (Equation (5)), (row 4) a shallow neural network, trained with backpropagation, (row 5) the MILP-optimized weights ω (Equations (12-21)), (row 6) the MILP-optimized weights with Sparsity constraints ω_{SP} (Equations (12-21 + 22-24)) and (row 7) the MILP-optimized weights as local receptive field ω_{LRF} (Equations (12-21 + 25)). The amount of data to memorize has been selected appropriately so that (except Basline 1) all approaches can successfully memorize the input data (left column). Even though the memorization capacities of the MILP versions (red) are much higher compared to the neural network, especially the sparse variant generalizes in a similar quality of the neural network in the presence of noise. Note, that once the memory limits of the baselines have been reached, they produce useless results. Only the MILP and the neural network show an incremental error behavior.

In this section the storage capacities of MILP optimized Hopfield networks are analyzed. It is shown that the proposed three variants (MILP, MILP with sparsity and MILP with LRF) can store more data compared to the baseline implementations or differentiable programming.

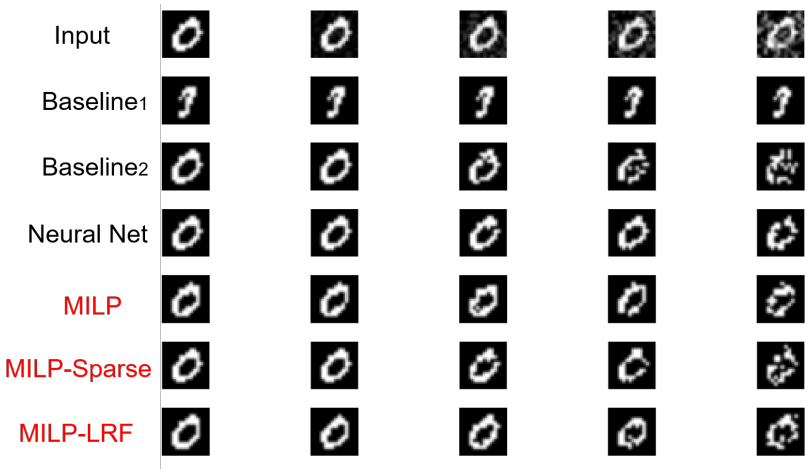

Fig. 7. Memorization results in presence of noise: First row: Input image with increasing noise (starting form 0). Reconstruction using: (2nd row) the Baseline 1, (3rd row) the Baseline 2, (4th row) a shallow neural network, (5th row) the MILP-optimized weights ω, (6th row) the MILP-optimized weights with Sparsity constraints ω_{SP}, (7th row) the MILP-optimized weights as local receptive field ω_{LRF}. The amount of training data was selected to keep (except Basline1) competitive in the original memorization (left column).

4.2 Classification using Hopfield Networks

For the next series of experiments, a learned Hopfield network is applied to a classification task. The classical *wine*, *zoo* and *breastEW* dataset have been used. The first two datasets are multicriterial classification tasks, with 3 categories for the wine dataset and 7 categories for the zoo dataset, whereas the breastEW dataset is a binary task.

To transform a numerical feature in one dimension into a binary one, k-means-clustering on this dimension (e.g. $k = 3$) is applied and used for a one-hot encoding. If e.g. the value 0.78 falls into cluster 2 (out of 3), this value is encoded as $(-1, 1, -1)$. Thus, e.g. the 30-dimensional features of the breastEW dataset are converted into a 90-dimensional binary dataset. From this dataset examples are used to train the Hopfield network and an unseen example is classified by feeding it into the Network and by performing a matching of the obtained values with the memorized dataset.

In the following, two factors are analyzed, (a) the performance behavior when using an increasing amount of samples for training, and (b) the stability with respect to missing data (from 0 to 50%). For these experiments we use a decision tree, a random forest and a neural network for comparison (on exactly the same train and test data). All experiments have been repeated 10 times and the diagram in figure 8 shows the obtained mean performance values. The neural network is a shallow network with the same amount of neurons as the Hopfield

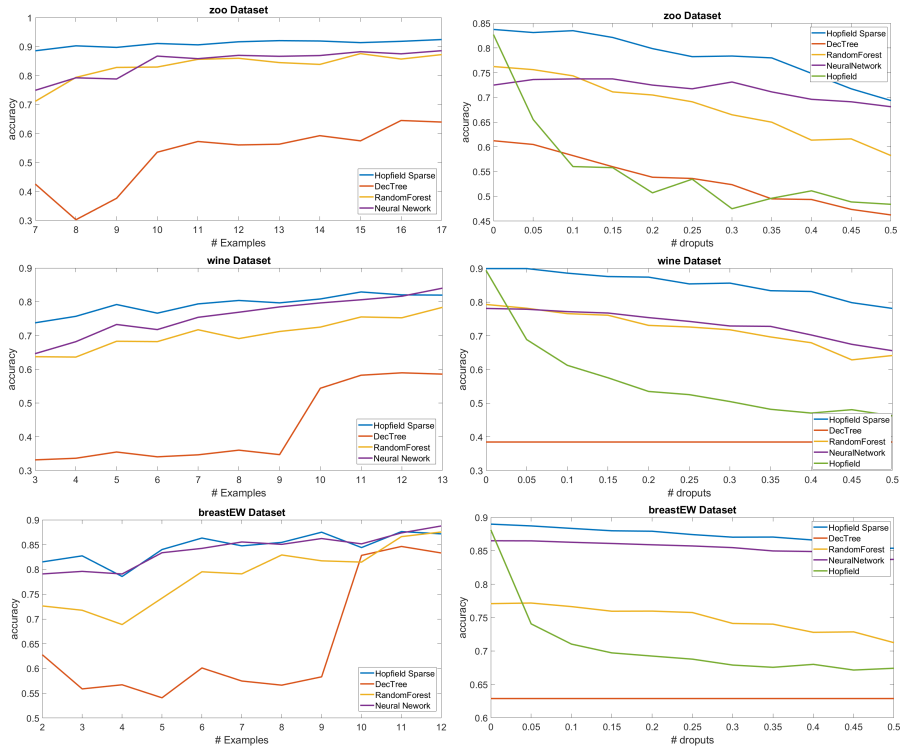

Fig. 8. Left: Classification performance for learning from small data. The x-axis shows the number of examples which have been used for training and the y-axis the classification performance on the unseen test data. Right: Classification performance for missing data. The x-axis shows the amount of missing data (up to 50 %). The Hopfield network can memorize from the missing data and therefore classify the test data better then the decision tree, random forest or a neural network. The sparse MILP solution is superior to the non-sparse MILP solution which is highly sensitive to missing data.

network and a softmax classification layer. As can be seen, already with two or three examples (for each class), a competitive classification performance can be obtained which is pretty constant for the Hopfield network and increasing for a decision tree (which is known to be sensitive to overfitting) and a random forest. For the experiment in figure 8 (right), the number of examples has been kept fixed to the number 4. For inference, the samples have been disturbed with a random amout of zero-entries, ranging from 0 to 50%. Here the outcome is even more clear: The (sparse) Hopfield networks can memorize from missing data and yield a much better classification accuracy on the test sets. The sparse MILP solution is superior to the non-sparse MILP solution which is in line with the discussion in [14].

5 Summary

This work presents an approach to optimize the weights of a Hopfield network as mixed-integer linear program. As the original formulation involves a sign-function, it is not differentiable, but parameter optimization using a (mixed integer) LP is possible. As autoassociative memory this work analyzes the amount of patterns which can be stored in such a discrete Hopfield network. It is shown, that the traditional storage description models and neural network based training with differential programming are inferior to a globally optimized solution which can be obtained with a MILP. As a global optimal solution is obtained by using a MILP, the approach is nearly parameter free and independent from seeding and other factors which are important for differentiable programming. Additionally it is possible to enforce sparsity constraints on the weights. Such additional constraints can improve the generalization of such a model. Two baseline methods, a neural network, the MILP formulation and two sparse variants have been formulated and compared. It turns out, that the MILP variant is capable for an optimal usage of the available memory, but can be more sensitive in case of noise. Experiments on the storage capacities and examples on classification using Hopfield networks demonstrate the general applicability of the optimized model.

Acknowledgments

This work was supported by the Federal Ministry of Education and Research (BMBF), Germany under the project LeibnizKILabor (grant no. 01DD20003), the Deutsche Forschungsgemeinschaft (DFG) under Germany's Excellence Strategy within the Cluster of Excellence PhoenixD (EXC 2122) and the Erskine Programme at the University of Canterbury, New Zealand. The author also thanks Dr. Roberto Henschel for fruitful discussions and hints.

References

1. Abbott, L.F.: Learning in neural network memories. Network: Computation In Neural Systems **1**, 105–122 (1990)
2. Abu-Mostafa, Y., St. Jacques, J.: Information capacity of the hopfield model. IEEE Transactions on Information Theory **31**(4), 461–464 (1985)
3. Aharon, M., Elad, M., Bruckstein, A.: K-svd: An algorithm for designing overcomplete dictionaries for sparse representation. IEEE Transactions on Signal Processing **54**(11), 4311–4322 (2006)
4. Almohamad, H., Duffuaa, S.: A linear programming approach for the weighted graph matching problem. IEEE Transactions on Pattern Analysis and Machine Intelligence **15**(5), 522–525 (1993)
5. Anderson, R., Huchette, J., Ma, W.: Strong mixed-integer programming formulations for trained neural networks. Math. Program **183**, 3–39 (2020)
6. Bae, W., Lee, S., Lee, Y., Park, B., Chung, M., Jung, K.H.: Resource optimized neural architecture search for 3d medical image segmentation. In: Medical Image Computing and Computer Assisted Intervention – MICCAI 2019. p. 228–236. Springer-Verlag, Berlin, Heidelberg

7. Baydin, A.G., Pearlmutter, B.A., Radul, A.A., Siskind, J.M.: Automatic differentiation in machine learning: A survey. J. Mach. Learn. Res. **18**(1), 5595–5637 (Jan 2017)
8. Bellec, G., Kappel, D., Maass, W., Legenstein, R.: Deep rewiring: Training very sparse deep networks. arXiv **abs/1711.05136** (2018)
9. Cooper, L., Liberman, F., Oja, E.: A theory for the acquisition and loss of neuron specificity in visual cortex. Biol. Cybernetics **33**, 9–28 (1979)
10. Cun, Y.L., Denker, J.S., Solla, S.A.: Optimal brain damage. In: Advances in Neural Information Processing Systems. pp. 598–605. Morgan Kaufmann (1990)
11. Dantzig, G.B.: Maximization of a linear function of variables subject to linear inequalities. Activity analysis of production and allocation **13**, 339–347 (1951)
12. Davey, N., Adams, R.: High capacity associative memories and connection constraints. Connect. Sci. **16**, 47–65 (03 2004)
13. Dennis, J.E., Schnabel, R.B.: Numerical Methods for Unconstrained Optimization and Nonlinear Equations. Society for Industrial and Applied Mathematics (1996)
14. Gambella, C., Ghaddar, B., Naoum-Sawaya, J.: Optimization problems for machine learning: A survey. European Journal of Operational Research **290**(3), 807–828 (May 2021)
15. Gurobi Optimization, L.: Gurobi optimizer reference manual (2021), http://www.gurobi.com
16. Heo, J., Joo, S., Moon, T.: Fooling neural network interpretations via adversarial model manipulation. In: Wallach, H., Larochelle, H., Beygelzimer, A., Alche-Buc, F., Fox, E., Garnett, R. (eds.) Advances in Neural Information Processing Systems. vol. 32. Curran Associates, Inc. (2019)
17. Hillar, C., Mehta, R., Koepsell, K.: A hopfield recurrent neural network trained on natural images performs state-of-the-art image compression. In: 2014 IEEE International Conference on Image Processing (ICIP). pp. 4092–4096 (2014)
18. Hopfield, J.J.: Neural networks and physical systems with emergent collective computational abilities. Proceedings of the National Academy of Sciences **79**(8), 2554–2558 (1982)
19. Hopfield, J., Feinstein, D., Palmer, R.: 'unlearning' has a stabilizing effect in collective memories. Nature **304**(5922), 158–159 (1983)
20. Hopfield, J., Tank, D.: Neural computation of decisions in optimization problems. Biological cybernetics **52**, 141–52 (02 1985). https://doi.org/10.1007/BF00339943
21. Hubara, I., Courbariaux, M., Soudry, D., El-Yaniv, R., Bengio, Y.: Binarized neural networks. In: Lee, D., Sugiyama, M., Luxburg, U., Guyon, I., Garnett, R. (eds.) Advances in Neural Information Processing Systems. vol. 29. Curran Associates, Inc. (2016)
22. Icarte, R.T., Illanes, L., Castro, M.P., Ciré, A., McIlraith, S.A., Beck, J.C.: Training binarized neural networks using mip and cp. In: Schiex, de Givry (eds.) Principles and Practice of Constraint Programming. Springer, LNCS 11802 (2019)
23. Komodakis, N., Tziritas, G.: Approximate labeling via graph cuts based on linear programming. IEEE Transactions on Pattern Analysis and Machine Intelligence **29**(8), 1436–1453 (2007)
24. LeCun, Y., Cortes, C., Burges, C.: Mnist handwritten digit database. ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist **2** (2010)
25. Makhortov, S., Ivanov, I.: Equivalent transformation of the reasoning model in production zeroth-order logic. In: 2020 International Conference on Information Technology and Nanotechnology (ITNT). pp. 1–4 (2020)
26. Mangasarian, O.L.: Mathematical programming in neural networks. ORSA Journal on Computing **5**, 349–360 (1993)

27. McCulloch, W., Pitts, W.: A logical calculus of ideas immanent in nervous activity. Bulletin of Mathematical Biophysics **5**, 127–147 (1943)
28. Murty, K.: Linear Programming. Wiley (1983)
29. Nguyen, H.T., Franke, K.: A general lp-norm support vector machine via mixed 0-1 programming. In: Perner, P. (ed.) Machine Learning and Data Mining in Pattern Recognition. pp. 40–49. Springer Berlin Heidelberg, Berlin, Heidelberg (2012)
30. Pajares, G., Guijarro, M., Ribeiro, A.: A hopfield neural network for combining classifiers applied to textured images. Neural Networks **23**(1), 144–153 (2010)
31. Personnaz, L., Guyon, I., Dreyfus, G.: Information storage and retrieval in spin-glass like neural networks. Journal De Physique Lettres **46**, 359–365 (1985)
32. Ramsauer, H., Schäfl, B., Lehner, J., Seidl, P., Widrich, M., Adler, T., Gruber, L., Holzleitner, M., Pavlović, M., Sandve, G.K., Greiff, V., Kreil, D., Kopp, M., Klambauer, G., Brandstetter, J., Hochreiter, S.: Hopfield networks is all you need. arXiv **2008.02217** (2021)
33. Recanatesi, S., Katkov, M., Romani, S., Tsodyks, M.: Neural network model of memory retrieval. Frontiers in Computational Neuroscience **9** (2015)
34. Rosenblatt, F.: The perceptron: A probabilistic model for information storage and organization in the brain. Psychological Review **65**(6), 386–408 (1958)
35. Serra, T., Kumar, A., Ramalingam, S.: Lossless compression of deep neural networks. In: Hebrard, E., Musliu, N. (eds.) Integration of Constraint Programming, Artificial Intelligence, and Operations Research. pp. 417–430. Springer International Publishing, Cham (2020)
36. Tanaka, G., Nakane, R., Takeuchi, T., Yamane, T., Nakano, D., Katayama, Y., Hirose, A.: Spatially arranged sparse recurrent neural networks for energy efficient associative memory. IEEE Transactions on Neural Networks and Learning Systems **31**(1), 24–38 (2020)
37. Thorbjarnarson, T., Yorke-Smith, N.: On training neural networks with mixed integer programming. arXiv **2009.03825** (2020), https://www.cmu.edu/epp/patents/events/aaai21/aaaicontent/papers/training-integer-valued-neural-networks-with-mixed-integer-programming.pdf
38. Venkatesh, S.S.: Epsilon capacity of neural networks. In: AIP Conference Proceedings 1986. vol. 151, pp. 440–445 (2014)
39. Williams, H.P.: Logic and Integer Programming. Springer Publishing Company, Incorporated, 1st edn. (2009)
40. Zhu, H., Murali, P., Phan, D., Nguyen, L., Kalagnanam, J.: A scalable mip-based method for learning optimal multivariate decision trees. In: Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M.F., Lin, H. (eds.) Advances in Neural Information Processing Systems. vol. 33, pp. 1771–1781. Curran Associates, Inc. (2020)