Convex Relaxation for Training Neural Networks

Fariba Zohrizadeh, Mohsen Kheirandishfard, Shahrouz Ryan Alimo, Farhad Kamangar, and Ramtin Madani

***Abstract*—This paper investigates the application of convex optimization in training neural networks (NNs). Our main contribution is a convex relaxation tailored for NNs that can serve as an alternative to the existing relaxations from the area of nonlinear optimization. We prove that by incorporating a family of regularization terms the proposed relaxation is guaranteed to be exact, using which the training task can be cast as a sequence of convex problems. This approach improves upon the common-practice gradient-based methods by enabling the incorporation of hard constraints. Lastly, the potential of the proposed approach is corroborated on the problem of imbalanced classification.**

I. Introduction

Neural networks (NNs) have been demonstrated to have special abilities in extracting sophisticated information from raw data. This renders them as suitable tools for a wide variety of applications in artificial intelligence and machine learning including classification [1], [2] and depth estimation [3], [4] among many. Despite the widespread use of NNs, a complete understanding of their success is still lacking and theoretical studies are mainly limited to the networks with special architectures [5], [6].

The primary challenge in training NNs arises from the non-convexity of the training problems. Gradient-based approaches such as stochastic gradient descent, conjugate gradient, and Limited-memory Broyden-Fletcher-Goldfarb-Shanno (L-BFGS) are among the most popular numerical methods for training NNs [7]–[9]. These approaches rely descent directions obtained via recursively calculation of gradient with respect to network parameter. There is considerable theoretical and empirical evidence indicating the effectiveness of the gradient-based methods in converging to global optimality (or satisfactory local optimality), under various assumptions [10], [11]. However, for general network architectures, gradient-based methods often suffer from several problems such as the vanishing/exploding gradient phenomena. Moreover, they cannot incorporate hard constraints which have been proven to be advantageous in many applications [12], [13].

In the light of empirical success of NNs, a question arises as to whether more sophisticated optimization techniques can be leveraged to train these networks. In response to this question, we investigate a well-studied technique in nonlinear optimization literature, namely convex relaxation, which reduces non-convex problems into convex surrogate. Among various relaxation approaches, semidefinite programming (SDP) [14] stand out for offering high-quality solutions. However, its applicability is limited to moderate size problems since they increase the problem size quadratically and require computationally expensive conic inequalities. Moreover, in the presence of constraints, the optimal solutions of these relaxations are not necessarily feasible for the original non-convex problem.

A. Contributions

In this work, we aim to establish a bridge between the areas of artificial neural networks and convex optimization by developing a powerful and flexible training approach. To serve this purpose, we first transform the training problem into an equivalent constrained optimization. Then, we convexify this constrained optimization by means of a novel convex quadratic relaxation and the well-known difference of convex programming technique. To ensure that the convexified problem provides a feasible point to the training problem, a novel regularization term is incorporated into the objective of the relaxed problems.

On the theoretical front, we derive certain conditions under which the regularized relaxation is guaranteed to provide feasible points for the training problem. Moreover, we theoretically prove that, if certain assumptions are met, solving a sequence of the regularized problem results in a sequence of feasible points whose objective values monotonically improve. The proposed approach, called Convex-NN, offers various theoretical and practical advantages: it jointly estimates the network parameters, admits additional convex constraints, and provides a flexible framework for further study and exploration. The potential of the proposed approach is corroborated on the problem of imbalanced classification.

B. Notation

Throughout this paper, symbols $\mathbb{R}^n$, $\mathbb{R}^{n\times m}$, and $\mathbb{S}_n$ denote the set of real n-dimensional vectors, real $n\times m$ matrices, and $n\times n$ real symmetric matrices, respectively. For a given matrix $\boldsymbol{a}$, the notations a_{ij} and $\boldsymbol{a}_{i*}$ respectively refer to the $(i,j)^{th}$ element and the i^{th} row of $\boldsymbol{a}$. In addition, given matrices $\boldsymbol{a}$ and $\boldsymbol{b}$ of the same size, symbols $\boldsymbol{a}\circ\boldsymbol{b}$ and $\langle \boldsymbol{a},\boldsymbol{b}\rangle$ stand for their Hadamard product and inner product. Notation $\|.\|_p$ refers to either the matrix norm or the vector norm depending on the context, $\|.\|_{\mathrm{F}}$ shows the Frobenius norm, and $|.|$ indicates the absolute value. Symbols $\mathrm{tr}\{.\}$ and $(.)^\top$, respectively, denote the trace operator and transpose operator. The notation $\mathcal{I}_n$ represents the index set $\{1,\ldots,n\}$. Notations $\mathbf{1}_n$ and $\boldsymbol{I}_n$ refer to $n\times 1$ column vector of all ones and identity matrix of size $n\times n$.

II. Problem Formulation

Consider an L layer neural network with the l^{th} layer consisting of n_l nodes, involving linear transformation with weights $\boldsymbol{w}^l\in\mathbb{R}^{n_{l-1}\times n_l}$ and biases $\boldsymbol{b}^l\in\mathbb{R}^{n_l}$, for every $l\in\mathcal{I}_L$. The network can be formulated as a function:

$$f(\boldsymbol{w},\boldsymbol{b})\triangleq \boldsymbol{w}^L h^{L-1}(\ldots \boldsymbol{w}^2 h^1(\boldsymbol{w}^1\boldsymbol{x}^1+\boldsymbol{b}^1\mathbf{1}_m^\top)+\boldsymbol{b}^2\mathbf{1}_m^\top\ldots)+\boldsymbol{b}^L\mathbf{1}_m^\top,$$

where $\boldsymbol{x}^1\in\mathbb{R}^{n_1\times m}$ accounts for the input matrix, $\boldsymbol{b}=\{\boldsymbol{b}^l\}_{l\in\mathcal{I}_L}$ and $\boldsymbol{w}=\{\boldsymbol{w}^l\}_{l\in\mathcal{I}_L}$ represent all of the bias vectors

and all of the weight matrices, respectively, and for every $l \in \mathcal{I}_L$, the function $h^l : \mathbb{R} \to \mathbb{R}$ represents the element-wise activation of the l^{th} layer. Following the common-practice and with no loss of generality, we assume that h^L is the identity function [15], [16].

Consider an input matrix $\boldsymbol{x}^1 \in \mathbb{R}^{n_1 \times m}$ and an output matrix $\boldsymbol{y} \in \mathbb{R}^{n_L \times m}$ representing m feature data and their corresponding labels, respectively. Training the neural network is tantamount to learning the coefficients $\boldsymbol{w}$ and $\boldsymbol{b}$ such that the misfit between the predicted output and the true labels is minimized in terms of a desired loss function. Using the ℓ_2-norm loss function, this problem can be formulated as:

$$\underset{\boldsymbol{w},\boldsymbol{b}}{\text{minimize}} \quad \|f(\boldsymbol{w},\boldsymbol{b}) - \boldsymbol{y}\|_2^2. \tag{1}$$

Due to nonlinearity of the nested function f the training problem (1) is non-convex and computationally challenging [10], [17], [18]. Despite the efficiency of common-practice local search algorithms for solving the problem (1), they suffer from several issues such as "vanishing gradient", wherein the gradient elements corresponding to the weights in early layers diminish, resulting in slow convergence [15], [19], [20].

In order to remedy these issues and to enable the potential of imposing hard constraints, in this paper, we employ convex relaxation for training neural networks. To this end, we break the functional dependencies of f, by defining auxiliary variables $\boldsymbol{z} = \{\boldsymbol{z}^l \in \mathbb{R}^{n_l \times m}\}_{l \in \mathcal{I}_L}$ and $\boldsymbol{a} = \{\boldsymbol{a}^l \in \mathbb{R}^{n_l \times m}\}_{l \in \mathcal{I}_{L-1}}$, accounting for the output of linear transformations and activation functions of layers, respectively. Using the auxiliary variables, the problem (1) can be reformulated as:

$$\begin{aligned}
\underset{\boldsymbol{w},\boldsymbol{b},\boldsymbol{z},\boldsymbol{x}}{\text{minimize}} \quad & \|\boldsymbol{z}^L - \boldsymbol{y}\|_{\text{F}}^2 && && \text{(2a)}\\
\text{subject to} \quad & \boldsymbol{z}^l = \boldsymbol{w}^l\boldsymbol{x}^l + \boldsymbol{b}^l\boldsymbol{1}_m^\top, && l \in \mathcal{I}_L, && \text{(2b)}\\
& \boldsymbol{x}^{l+1} = h^l(\boldsymbol{z}^l), && l \in \mathcal{I}_{L-1}, && \text{(2c)}
\end{aligned}$$

Notice that the problem (2a)–(2c) offers a more interpretable formulation of (1) in which bilinear terms $\{\boldsymbol{w}^l\boldsymbol{x}^l\}_{l \in \mathcal{I}_L}$ and nonlinear activation functions $\{h^l\}_{l \in \mathcal{I}_{L-1}}$ are the sources of nonconvexity. In this work, we transform (2a)–(2c) into a class of computationally tractable convex programs by means of a novel parabolic relaxation and the classical difference of convex programming technique. The following section offers brief survey of the relevant literature.

III. Related Work

There has been a recent surge of interest in developing convex formulations of neural network and its variant [21]–[24]. [21] showed that training a neural network can be seen as a convex optimization problem involving an infinite number of parameters. [22] developed a convex formulation of multi-layer learning using normalized kernels. [23] cast the problem of training a convolutional neural network as a low-rank minimization problem which is further relaxed to obtain a convex formulation.

From a different viewpoint, an alternative line of research has considered replacing training problem (1) with a constrained non-convex problem of form (2a)–(2c) or its variants [15], [20]. [20] proposed the method of auxiliary coordinates (MAC) which introduces auxiliary variables as a proxy of the network activations and then incorporate a quadratic penalty term into the objective function to approximately impose the non-convex relation between them. Closely related to MAC, [15] proposed to solve problem (2a)–(2c) using a highly parallelizable approach based on the alternating direction method of multipliers. The proposed approach alternatively solves a sequence of minimization sub-steps that each enjoys a closed-form solution. However, their proposed approach is not necessary compatible with additional hard constraints.

IV. Convexified Neural Networks

This section describes our proposed approach, called *convexified neural network* (Convex-NN). We first introduce a computationally tractable convex relaxation for the problem (2a)–(2c). Then, we present a regularization method to ensure that the solution of the relaxed problem satisfies the constraints of the original problem (1). Finally, we devise a sequential approach to obtain near optimal feasible points.

A. Convex Relaxation

Problem (2a)–(2c) is nonconvex and possibly intractable. The first source of non-convexity arises from the presence of bilinear products $\boldsymbol{w}^l\boldsymbol{x}^l$. In order to tackle the non-convexity, we propose a novel convex relaxation which transforms the constraints (2b) to a set of convex quadratic inequalities. For every $l \in \mathcal{I}_L$, let $\boldsymbol{W}^l \in \mathbb{R}^{n_l \times n_l}$ and $\boldsymbol{X}^l \in \mathbb{R}^{m \times m}$ account for the quadratic terms $\boldsymbol{w}^l(\boldsymbol{w}^l)^\top$ and $(\boldsymbol{x}^l)^\top\boldsymbol{x}^l$, respectively. The constraint (2b) can be equivalently reformulated as:

$$\begin{bmatrix} \boldsymbol{W}^l & (\boldsymbol{z}^l - \boldsymbol{b}^l\boldsymbol{1}_m^\top) \\ (\boldsymbol{z}^l - \boldsymbol{b}^l\boldsymbol{1}_m^\top)^\top & \boldsymbol{X}^l \end{bmatrix} = \begin{bmatrix} \boldsymbol{w}^l \\ (\boldsymbol{x}^l)^\top \end{bmatrix} \begin{bmatrix} (\boldsymbol{w}^l)^\top & \boldsymbol{x}^l \end{bmatrix}, \tag{3}$$

for every $l \in \mathcal{I}_L$. The literature of optimization theory suggests the relaxation of (3) into the following linear matrix inequality:

$$\begin{bmatrix} \boldsymbol{W}^l & (\boldsymbol{z}^l - \boldsymbol{b}^l\boldsymbol{1}_m^\top) \\ (\boldsymbol{z}^l - \boldsymbol{b}^l\boldsymbol{1}_m^\top)^\top & \boldsymbol{X}^l \end{bmatrix} \succeq \begin{bmatrix} \boldsymbol{w}^l \\ (\boldsymbol{x}^l)^\top \end{bmatrix} \begin{bmatrix} (\boldsymbol{w}^l)^\top & \boldsymbol{x}^l \end{bmatrix}. \tag{4}$$

This technique is regarded as the semidefinite programming relaxation of constraint (3). However, despite its strength the main drawback of this approach is the curse of dimensionality caused by the introduction of the $m \times m$ matrix $\boldsymbol{X}^l$.

As a result, in this work, we adopt an alternative parabolic relaxation approach. To obtain a computationally tractable surrogates for the problem (2a)–(2c), we relax the constraint (3) into the following convex quadratic inequalities:

$$\begin{aligned}
& W_{ii}^l + X_{jj}^l + 2\,(z_{ij}^l - b_i^l) \geq \|\boldsymbol{w}_{*i}^l + \boldsymbol{x}_{*j}^l\|^2, && i,j \in \mathcal{I}_{n_l}, \mathcal{I}_m && \text{(5a)}\\
& W_{ii}^l + X_{jj}^l - 2\,(z_{ij}^l - b_i^l) \geq \|\boldsymbol{w}_{*i}^l - \boldsymbol{x}_{*j}^l\|^2, && i,j \in \mathcal{I}_{n_l}, \mathcal{I}_m && \text{(5b)}\\
& W_{ii}^l \geq \|\boldsymbol{w}_{*i}^l\|^2, && i \in \mathcal{I}_{n_l} && \text{(5c)}\\
& X_{jj}^l \geq \|\boldsymbol{x}_{*j}^{l-1}\|^2, && j \in \mathcal{I}_m && \text{(5d)}
\end{aligned}$$

Notice that the off-diagonal elements of $\boldsymbol{W}^l$ and $\boldsymbol{X}^l$ do not appear in the relaxed inequities (5a)–(5d). Hence, for every layer $l \in \mathcal{I}_L$,

- a single auxiliary variable W_{ii}^l is introduced for every node $i \in \mathcal{I}_{n_l}$, and
- a single auxiliary variable X_{jj}^l is introduced per data point $j \in \mathcal{I}_m$.

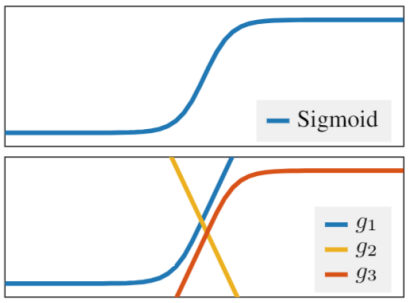

Fig. 1: Convex-decomposition of Sigmoid activation function. Sigmoid function can be cast as difference of convex functions $g_1 + g_2$ and $-g_3$.

As a result, the problem size grows linearly with respect to the data points. This is the primary strength of the proposed relaxation compared to the common-practice methods such as the semidefinite programming relaxations [25].

Definition 1. *For every $l \in \mathcal{I}_L$, define $\mathcal{Q}^l$ to be the set of all quadruplets $(\boldsymbol{w}^l, \boldsymbol{x}^l, \boldsymbol{W}^l, \boldsymbol{X}^l)$ that satisfy (5a)–(5d).*

The second source of non-convexity is induced by the non-linearity of the activation functions $\{h^l\}_{l\in\mathcal{I}_{L-1}}$, for which we employ the difference of convex (DC) programming method.

Proposition 1. *Any energy function with bounded Hessian can be decomposed as a difference of convex functions [26].*

Proposition 1 states that every activation function h^l with bounded second derivative can be decomposed as $h^l(\boldsymbol{z}^l) = p^l(\boldsymbol{z}^l) - q^l(\boldsymbol{z}^l)$ where $p^l : \mathbb{R} \to \mathbb{R}$ and $q^l : \mathbb{R} \to \mathbb{R}$ are convex functions. To illustrate, as shown in Figure 1, the widely-used Sigmoid activation function $h_{\text{sig}}(z) = \frac{1}{1+e^{-z}}$ can be written as the difference of the convex functions $p_{\text{sig}}(z) = \bar{h}(z) - \frac{z}{4} - \frac{1}{2}$ and $q_{\text{sig}}(z) = \bar{h}(-z) - 1$, where

$$\bar{h}(z) \triangleq \begin{cases} \frac{1}{1+e^{-z}} & z \le 0 \\ \frac{z}{4} + \frac{1}{2} & z > 0 \end{cases}.$$

Similarly, the Softplus activation $h_{\text{soft}}(z) = \frac{1}{\tau}\log(1+e^{\tau z})$ [27], which is a smoothed version of the ReLU activation, can be decomposed as $p_{\text{soft}}(z) - q_{\text{soft}}(z)$ where $p_{\text{soft}}(z) = \frac{1}{\tau}\log(1+e^{\tau z})$ and $q_{\text{soft}}(z) = 0$.

Given the decompositions $h^l(\boldsymbol{z}^l) = p^l(\boldsymbol{z}^l) - q^l(\boldsymbol{z}^l)$. We can cast the constraint (2c) into the following equations:

$$\frac{\boldsymbol{u}^{l+1} + \boldsymbol{x}^{l+1}}{2} = p^l(\boldsymbol{z}^l) \;\wedge\; \frac{\boldsymbol{u}^{l+1} - \boldsymbol{x}^{l+1}}{2} = q^l(\boldsymbol{z}^l), \quad \forall l \in \mathcal{I}_{L-1}, \tag{6}$$

where for every $l \in \mathcal{I}_{L-1}$, the auxiliary variable $\boldsymbol{u}^l \in \mathbb{R}^{n_l \times m}$ accounts for $p^l(\boldsymbol{z}^l) + q^l(\boldsymbol{z}^l)$. Next, we relax the above equations to the following convex inequalities:

$$\frac{\boldsymbol{u}^{l+1} + \boldsymbol{x}^{l+1}}{2} \ge p^l(\boldsymbol{z}^l) \;\wedge\; \frac{\boldsymbol{u}^{l+1} - \boldsymbol{x}^{l+1}}{2} \ge q^l(\boldsymbol{z}^l), \quad \forall l \in \mathcal{I}_{L-1}. \tag{7}$$

in order to convexify the problem (2a)–(2c).

Definition 2. *For every $l \in \mathcal{I}_{L-1}$, define $\mathcal{U}^l$ to be the set of all triplets $(\boldsymbol{z}^l, \boldsymbol{x}^l, \boldsymbol{u}^l)$ that satisfy the inequalities* (7).

In practice, the aforementioned convex relaxations are not necessarily exact, which means that the optimal weights and biases obtained by imposing the relaxed constraints (5a) – (5d) and (7) may not be feasible for the original non-convex problem (2a)–(2c). Next, we address this issue by incorporating a regularization term into the objective function.

B. Regularization

In order to enforce the equality constraints (2b)–(2c), we introduce a regularization technique that promotes feasible and meaningful solutions. To this end, consider arbitrary weight and bias coefficients $\check{\boldsymbol{w}} = \{\check{\boldsymbol{w}}^l\}$ and $\check{\boldsymbol{b}} = \{\check{\boldsymbol{b}}^l\}$ as the initial point.

Definition 3. *Define the functions $\mathbf{z}^l$ as*

$$\mathbf{z}^l(\boldsymbol{w}, \boldsymbol{b}) \triangleq \boldsymbol{w}^l h^{l-1}(\dots h^2(\boldsymbol{w}^2 h^1(\boldsymbol{w}^1 \boldsymbol{x}^1 + \boldsymbol{b}^1) + \boldsymbol{b}^2)\dots) + \boldsymbol{b}^l,$$

accounting for the outputs of linear transformations of each layer.

We employ regularization terms of the form

$$\begin{aligned} &r_{\check{\boldsymbol{w}},\check{\boldsymbol{b}}}(\boldsymbol{w}, \boldsymbol{b}, \boldsymbol{x}, \boldsymbol{u}, \boldsymbol{z}, \boldsymbol{W}, \boldsymbol{X}) \triangleq \\ &\sum_{l\in\mathcal{I}_L} \mu_{\boldsymbol{z}}^l \mathrm{tr}\{\boldsymbol{W}^l - 2\boldsymbol{w}^l(\check{\boldsymbol{w}}^l)^\top + \check{\boldsymbol{w}}^l(\check{\boldsymbol{w}}^l)^\top\} + \\ &\sum_{l\in\mathcal{I}_L} \mu_{\boldsymbol{z}}^l \mathrm{tr}\Big\{\boldsymbol{X}^l - 2h(\mathbf{z}^l(\check{\boldsymbol{w}}, \check{\boldsymbol{b}}))^\top \boldsymbol{x}^l + h(\mathbf{z}^l(\check{\boldsymbol{w}}, \check{\boldsymbol{b}}))^\top h(\mathbf{z}^l(\check{\boldsymbol{w}}, \check{\boldsymbol{b}}))\Big\} + \\ &\sum_{l\in\mathcal{I}_{L-1}} \mu_{\boldsymbol{p}}^l \mathrm{tr}\Big\{\frac{\boldsymbol{u}^{l+1} + \boldsymbol{x}^{l+1}}{2} \mathbf{1}_{m\times n_l} - \dot{p}^l(\mathbf{z}^l(\check{\boldsymbol{w}}, \check{\boldsymbol{b}}))(\boldsymbol{z}^l - \mathbf{z}^l(\check{\boldsymbol{w}}, \check{\boldsymbol{b}}))^\top\Big\} + \\ &\sum_{l\in\mathcal{I}_{L-1}} \mu_{\boldsymbol{q}}^l \mathrm{tr}\Big\{\frac{\boldsymbol{u}^{l+1} - \boldsymbol{x}^{l+1}}{2} \mathbf{1}_{m\times n_l} - \dot{q}^l(\mathbf{z}^l(\check{\boldsymbol{w}}, \check{\boldsymbol{b}}))(\boldsymbol{z}^l - \mathbf{z}^l(\check{\boldsymbol{w}}, \check{\boldsymbol{b}}))^\top\Big\} \end{aligned} \tag{8}$$

where $\{\mu_z^l > 0\}_{l\in\mathcal{I}_L}$, $\{\mu_p^l > 0\}_{l\in\mathcal{I}_{L-1}}$ and $\{\mu_q^l > 0\}_{l\in\mathcal{I}_{L-1}}$ are positive constants.

The *regularized relaxation* problem can be formulated as:

$$\begin{aligned} \underset{\substack{\boldsymbol{w},\boldsymbol{b},\boldsymbol{z},\boldsymbol{x},\boldsymbol{u} \\ \boldsymbol{W},\boldsymbol{A}}}{\text{minimize}} \quad & \|\boldsymbol{z}^L - \boldsymbol{y}\|_F^2 + \eta \times r_{\check{\boldsymbol{w}},\check{\boldsymbol{b}}}(\boldsymbol{w}, \boldsymbol{b}, \boldsymbol{x}, \boldsymbol{u}, \boldsymbol{z}, \boldsymbol{W}, \boldsymbol{X}) && && \text{(9a)} \\ \text{subject to} \quad & (\boldsymbol{w}^l, \boldsymbol{x}^l, \boldsymbol{W}^l, \boldsymbol{X}^l) \in \mathcal{Q}^l, && l \in \mathcal{I}_L, && \text{(9b)} \\ & (\boldsymbol{z}^l, \boldsymbol{x}^l, \boldsymbol{u}^l) \in \mathcal{U}^l, && l \in \mathcal{I}_{L-1}, && \text{(9c)} \end{aligned}$$

where the constant parameter $\eta > 0$ controls the emphasis on the regularization term. Observe that the problem (9a)–(9c) is convex and can be solved effectively using standard solvers.

The following theorem provides a sufficient conditions to guarantee that the proposed regularized relaxation produces feasible points for the original nonconvex problem (2a) – (2c). In other words, the relaxation of non-convex constraints (2b)–(2c) to (9b)–(9c) is lossless if η is sufficiently large.

Theorem 1. *If η is sufficiently large, then every optimal solution $(\overset{*}{\boldsymbol{w}}, \overset{*}{\boldsymbol{b}}, \overset{*}{\boldsymbol{z}}, \overset{*}{\boldsymbol{x}}, \overset{*}{\boldsymbol{u}}, \overset{*}{\boldsymbol{W}}, \overset{*}{\boldsymbol{X}})$ of the convex problem (9a)–(9c) satisfies the nonconvex constraints* (2b) *and* (2c)*. Moreover, $g(\overset{*}{\boldsymbol{w}}, \overset{*}{\boldsymbol{b}}) \le g(\check{\boldsymbol{w}}, \check{\boldsymbol{b}})$.*

Proof. To prove the theorem, consider the auxiliary problem

$$\underset{\substack{\boldsymbol{w},\boldsymbol{b},\boldsymbol{u},\boldsymbol{x},\boldsymbol{z}\\ \boldsymbol{W},\boldsymbol{X}}}{\text{minimize}} \quad \|\boldsymbol{z}^L-\boldsymbol{y}\|_{\mathrm{F}}^2+\eta\times r_{\tilde{\boldsymbol{w}},\tilde{\boldsymbol{x}}}(\boldsymbol{w},\boldsymbol{b},\boldsymbol{x},\boldsymbol{u},\boldsymbol{z},\boldsymbol{W},\boldsymbol{X}) \tag{10a}$$

$$\text{subject to} \quad \begin{bmatrix} \boldsymbol{W}^l & \boldsymbol{z}^l-\boldsymbol{b}^l\boldsymbol{1}_m^\top \\ (\boldsymbol{z}^l-\boldsymbol{b}^l\boldsymbol{1}_m^\top)^\top & \boldsymbol{X}^l \end{bmatrix}=\begin{bmatrix} \boldsymbol{w}^l \\ (\boldsymbol{x}^l)^\top \end{bmatrix}\begin{bmatrix} \boldsymbol{w}^l \\ (\boldsymbol{x}^l)^\top \end{bmatrix}^\top \quad l\in\mathcal{I}_L \tag{10b}$$

$$\frac{\boldsymbol{u}^{l+1}+\boldsymbol{x}^{l+1}}{2}=p^l(\boldsymbol{z}^l) \qquad l\in\mathcal{I}_{L-1} \tag{10c}$$

$$\frac{\boldsymbol{u}^{l+1}-\boldsymbol{x}^{l+1}}{2}=q^l(\boldsymbol{z}^l) \qquad l\in\mathcal{I}_{L-1} \tag{10d}$$

It can be easily verified that the problems (10a)–(10d) and (1) are equivalent if $\eta=0$. Denote the Lagrange multipliers associated with the constraints (10b), (10c), and (10d) by

$$\boldsymbol{\Lambda}^l=\begin{bmatrix} \boldsymbol{\lambda}_{\boldsymbol{W}}^l & \boldsymbol{\lambda}_{\boldsymbol{z}}^l \\ * & \boldsymbol{\lambda}_{\boldsymbol{X}}^l \end{bmatrix}\in\mathbb{S}_{n_l+m}, \quad \boldsymbol{\lambda}_p^l\in\mathbb{R}^{n_l\times m}, \quad \boldsymbol{\lambda}_q^l\in\mathbb{R}^{n_l\times m}, \tag{11}$$

By augmenting the constraints (10b)–(10d) into the objective, one can verify that for every $l\in\mathcal{I}_L$, we have

$$\overset{*}{\boldsymbol{\lambda}}_{\boldsymbol{W}}^l=\eta\mu_{\boldsymbol{z}}^l\boldsymbol{I}, \qquad \overset{*}{\boldsymbol{\lambda}}_{\boldsymbol{X}}^l=\eta\mu_{\boldsymbol{z}}^l\boldsymbol{I}. \tag{12}$$

The other Lagrange multipliers for an arbitrary minimizer $(\overset{*}{\boldsymbol{w}},\overset{*}{\boldsymbol{b}},\overset{*}{\boldsymbol{u}},\overset{*}{\boldsymbol{x}},\overset{*}{\boldsymbol{z}},\overset{*}{\boldsymbol{W}},\overset{*}{\boldsymbol{A}})$ can be derived recursively as follows:

$$\overset{*}{\boldsymbol{\lambda}}_{\boldsymbol{z}}^L=\overset{*}{\boldsymbol{z}}^L-\boldsymbol{y} \tag{13a}$$

$$\overset{*}{\boldsymbol{\lambda}}_p^l=\eta\mu_p^l+2\eta\mu_{\boldsymbol{z}}^{l+1}\Big[\overset{*}{\boldsymbol{x}}^{l+1}-h^{l+1}(\boldsymbol{z}^{l+1}(\check{\boldsymbol{w}},\check{\boldsymbol{b}}))\Big] +2(\overset{*}{\boldsymbol{w}}^{l+1})^\top\overset{*}{\boldsymbol{\lambda}}_z^{l+1} \tag{13b}$$

$$\overset{*}{\boldsymbol{\lambda}}_q^l=\eta\mu_q^l-2\eta\mu_{\boldsymbol{z}}^{l+1}\Big[\overset{*}{\boldsymbol{x}}^{l+1}-h^{l+1}(\boldsymbol{z}^{l+1}(\check{\boldsymbol{w}},\check{\boldsymbol{b}}))\Big] -2(\overset{*}{\boldsymbol{w}}^{l+1})^\top\overset{*}{\boldsymbol{\lambda}}_z^{l+1} \tag{13c}$$

$$\overset{*}{\boldsymbol{\lambda}}_{\boldsymbol{z}}^l=2^{-1}[\overset{*}{\boldsymbol{\lambda}}_{\boldsymbol{p}}^l\circ\dot{p}^l(\overset{*}{\boldsymbol{z}}^l)-\eta\mu_p^l\dot{p}^l(\boldsymbol{z}^l(\check{\boldsymbol{w}},\check{\boldsymbol{b}}))] +2^{-1}[\overset{*}{\boldsymbol{\lambda}}_{\boldsymbol{q}}^l\circ\dot{q}^l(\overset{*}{\boldsymbol{z}}^l)-\eta\mu_q^l\dot{q}^l(\boldsymbol{z}^l(\check{\boldsymbol{w}},\check{\boldsymbol{b}}))] \tag{13d}$$

Due to optimality of $(\overset{*}{\boldsymbol{w}},\overset{*}{\boldsymbol{b}},\overset{*}{\boldsymbol{u}},\overset{*}{\boldsymbol{x}},\overset{*}{\boldsymbol{z}},\overset{*}{\boldsymbol{W}},\overset{*}{\boldsymbol{X}})$, we have:

$$\begin{aligned}&\|\overset{*}{\boldsymbol{z}}^L-\boldsymbol{y}\|_{\mathrm{F}}^2+\eta\times r_{\tilde{\boldsymbol{w}},\tilde{\boldsymbol{x}}}(\overset{*}{\boldsymbol{w}},\overset{*}{\boldsymbol{b}},\overset{*}{\boldsymbol{u}},\overset{*}{\boldsymbol{x}},\overset{*}{\boldsymbol{z}},\overset{*}{\boldsymbol{W}},\overset{*}{\boldsymbol{X}})\le\\ &\|\overset{*}{\boldsymbol{z}}^L-\boldsymbol{y}\|_{\mathrm{F}}^2+\eta\times r_{\tilde{\boldsymbol{w}},\tilde{\boldsymbol{x}}}(\check{\boldsymbol{w}},\check{\boldsymbol{b}},\check{\boldsymbol{u}},\check{\boldsymbol{x}},\check{\boldsymbol{z}},\check{\boldsymbol{W}},\check{\boldsymbol{X}})=\|\check{\boldsymbol{z}}^L-\boldsymbol{y}\|_{\mathrm{F}}^2\end{aligned} \tag{14}$$

Hence,

$$0\le\lim_{\eta\to\infty}\eta^{-1}\|\overset{*}{\boldsymbol{\lambda}}_{\boldsymbol{z}}^L\|_{\mathrm{F}}\le\lim_{\eta\to\infty}\|\eta^{-1}\check{\boldsymbol{z}}^L-\boldsymbol{y}\|_{\mathrm{F}}=0 \tag{15}$$

Assume by induction that

$$\lim_{\eta\to\infty}\|\eta^{-1}\overset{*}{\boldsymbol{\lambda}}_{\boldsymbol{z}}^l\|_{\mathrm{F}}=0 \tag{16}$$

for an $l\in\mathcal{I}_L\setminus\{1\}$. Then according to (13b)–(13d) and (14), we have:

$$\lim_{\eta\to\infty}\|\eta^{-1}\overset{*}{\boldsymbol{\lambda}}_p^{l-1}-\mu_p^l\|_{\mathrm{F}}=0, \tag{17a}$$

$$\lim_{\eta\to\infty}\|\eta^{-1}\overset{*}{\boldsymbol{\lambda}}_q^{l-1}-\mu_q^l\|_{\mathrm{F}}=0, \tag{17b}$$

$$\lim_{\eta\to\infty}\|\eta^{-1}\overset{*}{\boldsymbol{\lambda}}_{\boldsymbol{z}}^{l-1}\|_{\mathrm{F}}=0. \tag{17c}$$

Therefore, if η is sufficiently large, then the matrix $\overset{*}{\boldsymbol{\Lambda}}^l$ is diagonally dominant for every $l\in\mathcal{I}_L$ and

$$\overset{*}{\boldsymbol{\lambda}}_p^{l-1}\ge 0 \qquad \overset{*}{\boldsymbol{\lambda}}_q^{l-1}\ge 0 \tag{18}$$

for every $l\in\mathcal{I}_{L-1}$. As a result, the relaxation of (10b)–(10d) into (5a)–(5d) and (5a)–(5d) and (7) is lossless. □

Algorithm 1 Sequential Regularized Relaxation.

Input: initialize $(\boldsymbol{w}_0,\boldsymbol{b}_0)$, $k=0$ and a fixed $\eta>0$

1: **repeat**
2: Obtain $(\overset{*}{\boldsymbol{w}},\overset{*}{\boldsymbol{b}},\overset{*}{\boldsymbol{u}},\overset{*}{\boldsymbol{x}},\overset{*}{\boldsymbol{z}},\overset{*}{\boldsymbol{W}},\overset{*}{\boldsymbol{X}})$ by solving the convex problem (9a)–(9c) with $\eta\times r_{\boldsymbol{w}_k,\boldsymbol{b}_k}$ regularization.
3: $(\boldsymbol{w}_{k+1},\boldsymbol{b}_{k+1})\leftarrow(\overset{*}{\boldsymbol{w}},\overset{*}{\boldsymbol{b}})$.
4: $k\leftarrow k+1$
5: **until** stopping criteria is met.

Output: $(\boldsymbol{w}_k,\boldsymbol{b}_k)$

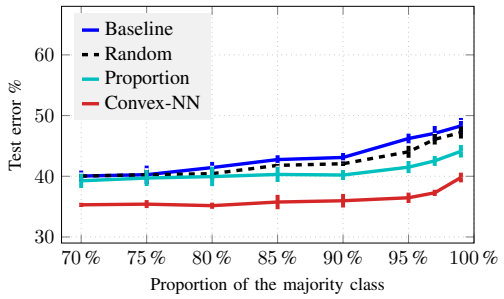

Fig. 2: Comparison of Convex-NN with hard constraint to the existing tricks for the imbalance classification problem.

C. Sequential Regularized Relaxation

In light of Theorem 1, we propose Algorithm 1 to infer feasible and near-optimal points for the non-convex problem (2a)–(2c) by solving the regularized relaxation problem (9a)–(9c) sequentially. The algorithm is terminated when the improvement of the objective value, between two consecutive round, is less than a positive value ε (e.g. $\varepsilon=10^{-7}$).

V. Experiments

This section demonstrates the potential of sequential convex relaxation in imposing hard constraints on training problems. There has been a recent surge of interest in imposing task-specific constraints on the output of neural networks for improving the quality of predictions [12], [13], [28]. Such constraints are often incorporated as penalty terms into the loss function. Despite the simplicity and empirical success, this strategy suffers from two major drawbacks. Firstly, adjusting relative weights between the loss function and penalty terms can be extremely challenging. Secondly, there is no guarantee that the resulting solution satisfies the hard constraints. Recently, [28] has demonstrated that replacing the soft constraints by the hard ones improves the behavioral performance of neural networks and can effectively address the aforementioned drawbacks.

We consider the problem of imbalanced classification [29], [30] in which the training samples are distributed highly unbalanced in different classes. The traditional classifiers exhibit

poor performance in solving this problem as they tend to misclassify the minority class instances as the majority. In what follows, we present experimental results that compare the performance of Convex-NN against two commonly used tricks for the problem of imbalanced classification. Notice that, it is not the intention of this work to compete with the existing algorithms that specifically target the imbalanced classification problem, but rather to show the potential and applicability of Convex-NN on real-world problems. We use the Yeast dataset from the UCI machine learning repository [31] and subsample classes 2 and 3 to create an imbalanced binary classification problem. We train a simple neural network architecture with two fully-connected hidden layers of 6 nodes each, and Sigmoid activation function. Constraints of the form

$$\sum_{i \in \mathcal{C}_j} \| \boldsymbol{z}_{*i}^L - \boldsymbol{y}_i \|_1 \leq \kappa_j, \tag{19}$$

are imposed to reduce the effect of unbalanced training samples, where $\boldsymbol{z}_{*i}^L$ denotes the i^{th} column of the matrix $\boldsymbol{z}^L$ and index set $\mathcal{C}_j \subseteq \mathcal{I}_m$ refers to the training samples belonging to the j^{th} class. Intuitively, the constraint (19) ensures that the misfit between predicted outputs and the true labels for class j does not exceed κ. We impose the constraint (19) on both majority and minority classes with $\kappa_j = 0.5|\mathcal{C}_j|$. This enables Convex-NN to automatically assign the relative weights and avoid generating a biased classifier.

Following [30], we compare our results with conventional techniques for the imbalanced classification problem: 1) Proportion method weights the training samples of each class in proportion to the inverse of the class size; 2) Random approach weights the samples based on a rectified Gaussian distribution. We run these methods on the same network architecture and employ the Adam optimizer [8] with $\beta_1 = 0.9$, $\beta_2 = 0.999$, and learning rate 0.005 to train the network. Note that these parameters are well-tuned to obtain the best performance of the Adam optimizer. The results of this experiment are demonstrated in Figure 2 as the value of test classification error across various imbalance ratios. The scores in Figure 2 are obtained by averaging 10 independent runs with random splits and random starting points, where the weights are initialized using He's initializer and the bias parameters are all set to zero. Observe that imposing hard constraints using Convex-NN has resulted in better test accuracy.

VI. Conclusion

In this paper, we presented a novel convexification approach, called Convex-NN, which reduces the problem of training neural networks into solving a sequence of convex programs. We theoretically proved that under certain assumptions, the proposed approach results in a sequence of feasible points whose objective values monotonically improve. Convex-NN improves upon the common-practice gradient-based methods by jointly estimating the network parameters and admitting additional convex constraints. Numerical results corroborated the potential of the proposed approach on the problem of imbalanced classification.

References

[1] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun, "Overfeat: Integrated recognition, localization and detection using convolutional networks," in *ICLR*, 2014.

[2] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *CVPR*, 2016.

[3] Y. Zhang and T. Funkhouser, "Deep depth completion of a single RGB-D image," in *CVPR*, 2018.

[4] I. Laina, C. Rupprecht, V. Belagiannis, F. Tombari, and N. Navab, "Deeper depth prediction with fully convolutional residual networks," in *3DV*, 2016.

[5] Y. Zhang, J. Lee, M. Wainwright, and M. Jordan, "On the learnability of fully-connected neural networks," in *AISTATS*, 2017.

[6] S. S. Du, X. Zhai, B. Poczos, and A. Singh, "Gradient descent provably optimizes over-parameterized neural networks," in *ICLR*, 2019.

[7] J. Nocedal, "Updating quasi-newton matrices with limited storage," *Mathematics of computation*, pp. 773–782, 1980.

[8] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *ICLR*, 2015.

[9] S. J. Reddi, S. Kale, and S. Kumar, "On the convergence of adam and beyond," in *ICLR*, 2018.

[10] A. Choromanska, M. Henaff, M. Mathieu, G. B. Arous, and Y. LeCun, "The loss surfaces of multilayer networks," in *AISTATS*, 2015.

[11] M. Hardt and T. Ma, "Identity matters in deep learning," in *ICLR*, 2017.

[12] D. Pathak, P. Krahenbuhl, and T. Darrell, "Constrained convolutional neural networks for weakly supervised segmentation," in *ICCV*, 2015.

[13] S. K. Roy, Z. Mhammedi, and M. Harandi, "Geometry aware constrained optimization techniques for deep learning," in *CVPR*, 2018.

[14] L. Vandenberghe and S. Boyd, "Semidefinite programming," *SIAM review*, pp. 49–95, 1996.

[15] G. Taylor, R. Burmeister, Z. Xu, B. Singh, A. Patel, and T. Goldstein, "Training neural networks without gradients: A scalable ADMM approach," in *ICML*, 2016.

[16] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida, "Spectral normalization for generative adversarial networks," in *ICLR*, 2018.

[17] Y. N. Dauphin, R. Pascanu, C. Gulcehre, K. Cho, S. Ganguli, and Y. Bengio, "Identifying and attacking the saddle point problem in high-dimensional non-convex optimization," in *NIPS*, 2014.

[18] M. Janzamin, H. Sedghi, and A. Anandkumar, "Beating the perils of non-convexity: Guaranteed training of neural networks using tensor methods," *arXiv preprint arXiv:1506.08473*, 2015.

[19] D. Erhan, Y. Bengio, A. Courville, and P. Vincent, "Visualizing higher-layer features of a deep network," 2009.

[20] M. Carreira-Perpinan and W. Wang, "Distributed optimization of deeply nested systems," in *AISTATS*, 2014.

[21] Y. Bengio, N. L. Roux, P. Vincent, O. Delalleau, and P. Marcotte, "Convex neural networks," in *NIPS*, 2006.

[22] Ö. Aslan, X. Zhang, and D. Schuurmans, "Convex deep learning via normalized kernels," in *NIPS*, 2014.

[23] Y. Zhang, P. Liang, and M. J. Wainwright, "Convexified convolutional neural networks," in *ICML*, 2017.

[24] F. Bach, "Breaking the curse of dimensionality with convex neural networks," *JMLR*, pp. 1–53, 2017.

[25] V. Ganapathiraman, X. Zhang, Y. Yu, and J. Wen, "Convex two-layer modeling with latent structure," in *NIPS*, 2016.

[26] A. L. Yuille and A. Rangarajan, "The concave-convex procedure (CCCP)," in *NIPS*, 2002.

[27] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *AISTATS*, 2011.

[28] S. N. Ravi, T. Dinh, V. S. R. Lokhande, and V. Singh, "Constrained deep learning using conditional gradient and applications in computer vision," *arXiv preprint arXiv:1803.06453*, 2018.

[29] S. H. Khan, M. Hayat, M. Bennamoun, F. A. Sohel, and R. Togneri, "Cost-sensitive learning of deep feature representations from imbalanced data," *IEEE transactions on neural networks and learning systems*, pp. 3573–3587, 2018.

[30] M. Ren, W. Zeng, B. Yang, and R. Urtasun, "Learning to reweight examples for robust deep learning," in *ICML*, 2018.

[31] D. Dheeru and E. Karra Taniskidou, "UCI machine learning repository," 2017. [Online]. Available: http://archive.ics.uci.edu/ml