Proceedings of Machine Learning Research vol 144:1–12, 2021

Convexifying Neural Networks

Yuexin Bian YUBIAN@UCSD.EDU
University of California, San Diego

1. Objective

Neural network model. Consider a two-layer feed-forward ReLU network f with m neurons in the hidden layer $f : \mathbb{R}^d \to \mathbb{R}$:

$$f(x) = \sum_{j}^{m} \phi(x^T u_j) a_j \tag{1}$$

where $\phi(x) = \max(x, 0)$ is the ReLU activation function, $u_j \in \mathbb{R}^d, a_j \in \mathbb{R}$ are the weights for hidden and output layers, respectively. We denote the neural network as $\theta = \{a_j, u_j\}_{j=1}^m$

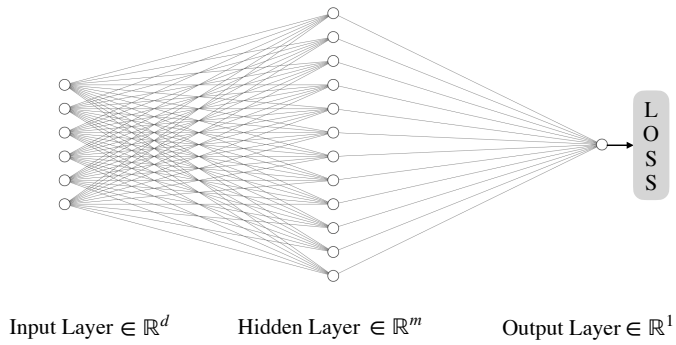


Figure 1: The neural network model

Searching for the optimal neural network. Given the dataset including n data pairs: $\{x_i \in \mathbb{R}^d, y_i \in \mathbb{R}\}$, or alternatively, the input data matrix $X \in \mathbb{R}^{n \times d}$ and a label vector $y \in \mathbb{R}^n$, **our goal** is to search for the optimal neural network $\theta = \{u_j, a_j\}$ that achieves the lowest loss $\mathcal{L}_\beta(\theta)$, as illustrated in (2).

$$\begin{aligned} p^* = \min_{\{a_j, u_j\}_{j=1}^m} \mathcal{L}_\beta(\theta) :&= \frac{1}{2} \| \sum_{i=1}^{n} f(x_i) - y_i \|_2^2 + \frac{\beta}{2} \sum_{j=1}^{m} (\|u_j\|_2^2 + a_j^2) \\ &= \frac{1}{2} \| \sum_{j=1}^{m} (Xu_j)_+ a_j - y \|_2^2 + \frac{\beta}{2} \sum_{j=1}^{m} (\|u_j\|_2^2 + a_j^2). \end{aligned} \tag{2}$$

In (2), the first term is the squared loss objective, and the second term is the squared l_2-norm of parameters with a regularization parameter $\beta > 0$. The l_2 regularization is beneficial to avoid over-fitting, and is computationally efficient because of having analytical solutions. Besides, without regularization, the set of optimal solutions contains infinitely many points. The loss $\mathcal{L}_\beta(\theta)$ is clearly non-convex, and the non-convexity comes from the non-linear ReLU activation and product between hidden and output layer weights.

2. Background and Motivation

Over the past decade, the research community, along with industry, have realised the significant potential that neural networks possess to perform complex tasks as humans do. This is due to the strong representing ability that neural network perform non-linear mapping from the feature to the output. However, this non-linear characteristics lead to the non-convexity of training loss. Gradients-descent methods can get stuck in local minima, and this is well recognized problem in machine learning Gori and Tesi (1992); Du et al. (2018). There have been various approaches proposed in the literature to help escape poor local optima, such as stochastic gradient descent, adding perturbations and heuristic methods.

Recently, there are literature that consider convex neural network training Bengio et al. (2005); Bach (2017). The advantage of convexity is that one can get the optimal network that is not stuck at local minimum. In earlier works, convexity arguments are restricted to infinite width networks. Authors Pilanci and Ergen (2020) turn attention to finite dimensional optimization problems. They develop a novel duality theory and introduce polynomial time finite dimensional convex programs, which are exact and computationally tractable. Authors Wang et al. (2021) leverage the theorem and mapping, provide an explicit construction of a continuous path between any neural network and the global minimum of its sublevel set. And work Ergen and Pilanci (2021) shows multiple ReLU sub-networks that beyond two-layer neural networks can be equivalently stated as a convex optimization problem. Research Bartan and Pilanci (2021) introduces semidefinite optimization to find global minimal for quantized neural networks.

3. Methods

3.1. Local search heuristics: backpropagation

One computational efficient way is to use the gradient to update the parameters of NNs, and find parameters that minimizes the loss function.

$$\theta_{t+1} = \theta_t - \eta \frac{\partial L_\beta(\theta)}{\partial \theta} \tag{3}$$

However, such methods can get stuck in local minima, and this is a well-recognized problem in machine learning Gori and Tesi (1992); Du et al. (2018).

3.2. Global search: an exact convex optimization formulation with polynomial time

In work Pilanci and Ergen (2020), the authors proposes that the two-layer ReLU networks with m hidden neurons (1) with regard to the loss function (2) can be globally optimized via the convex second order cone program.

We provide the table of the symbol and description for better readability.

Symbol	Description
$[n]$	the set of $\{1, 2, \ldots, n\}$
X	the data matrix $X = [x_1, x_2, \ldots, x_n]^T \in \mathbb{R}^{n\times d}$
S, S^c	S is the subset of $[n]$, and S^c is the complement of the set S
$\mathcal{H}_X$	the set of vectors $:= \cup\{\{\text{sign}(Xu)\} \in \mathbb{R}^n, u \in \mathbb{R}^d\}$
$\mathcal{S}_X$	the collection of sets $:= \{\{\cup_{h_i=1} i\} : h \in \mathcal{H}_X\}$
$D(S)$	the diagonal matrix $D(S) \in \mathbb{R}^{n\times n}, D(S)_{ii} = 1$ if $i \in S$, otherwise 0
P_S	$\{u \in \mathbb{R}^d : D(S)Xu \geq 0, (I_n - D(S))Xu \leq 0\}$
M	$M = \lvert S_X\rvert$, the number of elements in S_X

Table 1: Symbol and description

In table 1, we denote $M = |S_X|$, consider $r = \text{rank}(X) \leq n$, we have the following result Ojha (2000).

$$M \leq \sum_{k=0}^{r-1} \binom{n-1}{k} \leq 2r(\frac{e(n-1)}{r})^r \tag{4}$$

We first presents the theorem in Pilanci and Ergen (2020) that the NNs can be globally optimized via the convex second order cone program. We then give the proof and discussion of the convex formulation accordingly. Note that in work Pilanci and Ergen (2020) uses P and M to denote the distinct D and $|S_X|$, and P and M are exactly the same, for better readability, we only use M for all the case.

$$\begin{aligned} \min_{\substack{v_i, w_i \in \mathbb{R}^d, \\ i\in[M]}} \quad & \frac{1}{2}\|\sum_{i=1}^{M} D(S_i)X(v_i - w_i) - y\|_2^2 + \beta\sum_{i=1}^{M}(\|v_i\|_2 + \|w_i\|_2), \\ \text{s.t.} \quad & (2D(S_i) - I_n)Xv_i \geq 0, (2D(S_i) - I_n)Xw_i \geq 0, \forall i \in [M]. \end{aligned} \tag{5}$$

Theorem 1 *Consider $v_i^*, w_i^*, i \in [M]$ is the optimizer of the convex program (5). The convex program (5) and the non-convex problem (2) have identical optimal values when $m \geq m^*$, where $m^* = \sum_{i=1, v_i^*\neq 0}^{M} 1 + \sum_{i=1, w_i^*\neq 0}^{M} 1$. In particular, for $m = m^*$, an optimal solution to (2) can be constructed as follows:*

$$(u_{j1i}^*, a_{j1i}^*) = (\frac{v_i^*}{\sqrt{\|v_i^*\|_2}}, \sqrt{\|v_i^*\|_2}), \text{ if } v_i^* \neq 0, \tag{6a}$$

$$(u_{j2i}^*, a_{j2i}^*) = (\frac{w_i^*}{\sqrt{\|w_i^*\|_2}}, -\sqrt{\|w_i^*\|_2}), \text{ if } w_i^* \neq 0. \tag{6b}$$

How authors convert the non-convex problem (2) into a convex problem (5) can be summarized in Fig 2.

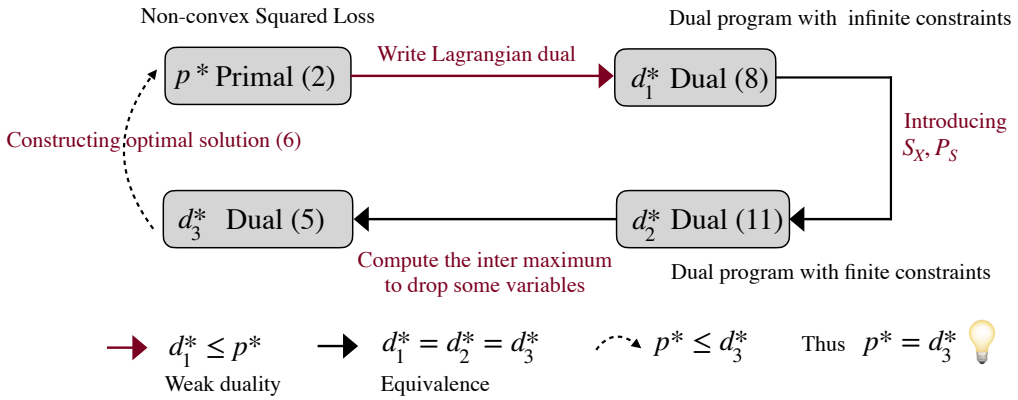


Figure 2: The main procedure to convert the original non-convex problem into a convex problem.

3.2.1. Proof for Theorem 1

Starting with an alternative representation (7) that is equivalent to (2), authors write the dual problem derived from (7) as (8). (8) is a second order cone program with infinite constraints.

$$p^* = \min_{\|u_j\|_2 \le 1} \min_{a_j} \frac{1}{2} \| \sum_{j=1}^{m} (Xu_j)_+ a_j - y \|_2^2 + \beta \sum_{j=1}^{m} |a_j|. \tag{7}$$

$$d^* = \max_{v \in \mathbb{R}^n} \quad -\frac{1}{2}\|y - v\|_2^2 + \frac{1}{2}\|y\|_2^2, \tag{8a}$$

$$\text{s.t.} \quad \max_{\|u\|_2 \le 1} \|v^T (Xu)_+\|_2 \le \beta. \tag{8b}$$

By introducing $\mathcal{H}_X, S_X, D(S), P_S$, authors can write the equivalent expression of the infinite constraints (8b):

$$\max_{S \in S_X, \|u\|_2 \le 1, u \in P_S} v^T D(S) X u \le \beta \cap \max_{S \in S_X, \|u\|_2 \le 1, u \in P_S} -v^T D(S) X u \le \beta \tag{9}$$

Recall that $M = |S_X|$, we then denote $S_1, S_2, \ldots, S_M \in S_X$ is the M elements of S_X. Using the strong duality, left side of (9) can be represented as M second order constraints:

$$\forall i \in [M], \exists a_i, b_i \in \mathbb{R}^n, \text{s.t. } a_i, b_i \ge 0, \|X^T D(S_i)(v + a_i + b_i) - X^T b_i\|_2 \le \beta. \tag{10}$$

Therefore, consider both sides of (9), the infinite constraints can be expressed as $2M$ second order constraints, where the the convex problem becomes the finite dimensional second order cone program (11).

$$d^* = \max_{v \in \mathbb{R}^n, a_i, b_i, a_i', b_i' \ge 0 \in \mathbb{R}^n, i \in [M]} \quad -\frac{1}{2}\|v - y\|_2^2 + \frac{1}{2}\|y\|_2^2, \tag{11a}$$

$$\text{s.t.} \quad \|X^T D(S_i)(v + a_i + b_i) - X^T b_i\|_2 \le \beta, \forall i \in [M], \tag{11b}$$

$$\|X^T D(S_i)(-v + a_i' + b_i') - X^T b_i'\|_2 \le \beta, \forall i \in [M]. \tag{11c}$$

With $\beta > 0$ and $v = a_i = b_i = a_i' = b_i' = 0, \forall i \in [M]$, the $2M$ second order constraints are strictly feasible, then strong duality holds and (11) can be wrriten as:

$$\begin{aligned} d^* = \min_{\lambda, \lambda' \geq 0 \in \mathbb{R}^M} \max_{\substack{v \in \mathbb{R}^n, \\ a_i, b_i, a_i', b_i' \geq 0 \in \mathbb{R}^n, i \in [M]}} & -\frac{1}{2}\|v - y\|_2^2 + \frac{1}{2}\|y\|_2^2 \\ & + \sum_{i=1}^M \lambda_i (\beta - \|X^T D(S_i)(v + a_i + b_i) - X^T b_i\|_2) \\ & + \sum_{i=1}^M \lambda_i (\beta - \|X^T D(S_i)(-v + a_i' + b_i') - X^T b_i'\|_2). \end{aligned} \tag{12}$$

We have $-\|x - y\|_2 = \min r^T(x - y)$, then we can introduce $r_1, \ldots, r_M, r_1', \ldots, r_M' \in \mathbb{R}^d$ and transform the l_2 norm in (12) to vector product.

$$\begin{aligned} d^* = \min_{\lambda, \lambda' \geq 0 \in \mathbb{R}^M} \max_{\substack{v \in \mathbb{R}^n, \\ a_i, b_i, a_i', b_i' \geq 0 \in \mathbb{R}^n, i \in [M]}} \min_{\substack{r_i, r_i' \in \mathbb{R}^d \\ \|r_i\|_2 \leq 1, \|r_i'\|_2 \leq 1}} & -\frac{1}{2}\|v - y\|_2^2 + \frac{1}{2}\|y\|_2^2 \\ & + \sum_{i=1}^M \lambda_i (\beta - r_i^T (X^T D(S_i)(v + a_i + b_i) - X^T b_i)) \\ & + \sum_{i=1}^M \lambda_i (\beta - r_i'^T (X^T D(S_i)(-v + a_i' + b_i') - X^T b_i')). \end{aligned} \tag{13}$$

The objective is concave in v, a_i, b_i and convex in r_i, r_i', the sets $\|r_i\|_2 \leq 1, \|r_i'\|_2 \leq 1$ are convex and compact, according to Sion's minimax theorem, (13) can be written as

$$\begin{aligned} d^* = \min_{\lambda, \lambda' \geq 0 \in \mathbb{R}^M} \min_{\substack{r_i, r_i' \in \mathbb{R}^d \\ \|r_i\|_2 \leq 1, \|r_i'\|_2 \leq 1}} \max_{\substack{v \in \mathbb{R}^n, \\ a_i, b_i, a_i', b_i' \geq 0 \in \mathbb{R}^n, i \in [M]}} & -\frac{1}{2}\|v - y\|_2^2 + \frac{1}{2}\|y\|_2^2 \\ & + \sum_{i=1}^M \lambda_i (\beta - r_i^T (X^T D(S_i)(v + a_i + b_i) - X^T b_i)) \\ & + \sum_{i=1}^M \lambda_i (\beta - r_i'^T (X^T D(S_i)(-v + a_i' + b_i') - X^T b_i')). \end{aligned} \tag{14}$$

We can then get the maximum by computing the corresponding v^*, a_i^*, b_i^*, (14) is

$$d^* = \min_{\lambda, \lambda' \geq 0 \in \mathbb{R}^M} \min_{\substack{\|r_i\|_2 \leq 1, \|r_i'\|_2 \leq 1, \\ (2D(S_i) - I_n) X r_i \geq 0, \\ (2D(S_i) - I_n) X r_i' \geq 0}} \frac{1}{2}\| \sum_{i=1}^M \lambda_i D(S_i) X r_i' - \lambda_i' D(S_i) X r_i - y\|_2^2 + \beta \sum_{i=1}^M (\lambda_i + \lambda_i'). \tag{15}$$

Define $v_i = \lambda_i r_i, w_i = \lambda_i' r_i'$, then $\|v_i\|_2 \leq \lambda_i, \|w_i\|_2 \leq \lambda_i'$. When $\|v_i\|_2 = \lambda_i, \|w_i\|_2 = \lambda_i'$, the problem (15) is feasible and optimal, then we derived the following expression:

$$d^* = \min_{v_i, w_i \in P_{S_i}} \frac{1}{2}\| \sum_{i=1}^M D(S_i) X (v_i - w_i) - y\|_2^2 + \beta \sum_{i=1}^M (\|v_i\|_2 + \|w_i\|_2). \tag{16}$$

Note that by description in Table 1, $v_i, w_i \in P_{S_i}$ can be expressed as constrains $(2D(S_i)-I_n)Xv_i \geq 0, (2D(S_i)-I_n)Xw_i \geq 0, \forall i \in [M]$. We can now write the clear and final expression below, which is equilvalent to (5):

$$\begin{aligned} d^* = \min_{v_i, w_i \in \mathbb{R}^d} \quad & \frac{1}{2}\| \sum_{i=1}^{M} D(S_i)X(v_i - w_i) - y\|_2^2 + \beta \sum_{i=1}^{M} (\|v_i\|_2 + \|w_i\|_2), \\ \text{s.t.} \quad & (2D(S_i) - I_n)Xv_i \geq 0, (2D(S_i) - I_n)Xw_i \geq 0, \forall i \in [M]. \end{aligned} \tag{17}$$

Since (8) is the dual program of (2), by weak duality, we have $d^* \leq p^*$. By constructing optimal solution as suggested using (6), then insert $\{u_j^*, a_j^*\}_{j=1}^{m^*}$ into (2), we have

$$\begin{aligned} p^* &\leq \frac{1}{2}\| \sum_{j=1}^{m^*} (Xu_j^*)_+ a_j^* - y\|_2^2 + \frac{\beta}{2} \sum_{\substack{i=1 \\ v_i^* \neq 0}}^{M} (\| \frac{v_i^*}{\sqrt{\|v_i^*\|_2}}\|_2^2 + \| \sqrt{\|v_i^*\|_2}\|_2^2) + \frac{\beta}{2} \sum_{\substack{i=1 \\ w_i^* \neq 0}}^{M} (\| \frac{w_i^*}{\sqrt{\|w_i^*\|_2}}\|_2^2 + \| \sqrt{\|w_i^*\|_2}\|_2^2) \\ &= \frac{1}{2}\| \sum_{\substack{i=1, \\ v_i^* \neq 0}}^{M} (D(S_i)X \frac{v_i^*}{\sqrt{\|v_i^*\|_2}} \sqrt{\|v_i^*\|_2} - y\|_2^2 + \frac{1}{2}\| \sum_{\substack{i=1, \\ w_i^* \neq 0}}^{M} (D(S_i)X \frac{w_i^*}{\sqrt{\|w_i^*\|_2}} \cdot -\sqrt{\|w_i^*\|_2} - y\|_2^2 \\ &\quad + \beta \sum_{i=1}^{M} (\|v_i^*\|_2 + \|w_i^*\|_2) \\ &= \frac{1}{2}\| \sum_{\substack{i=1, \\ v_i^* \neq 0}}^{M} D(S_i)Xv_i^* - y\|_2^2 + \frac{1}{2}\| \sum_{\substack{i=1, \\ w_i^* \neq 0}}^{M} -D(S_i)Xw_i^* - y\|_2^2 + \beta \sum_{i=1}^{M} (\|v_i^*\|_2 + \|w_i^*\|_2). \end{aligned} \tag{18}$$

Then $p^* = d^*$ under $m \geq m^*$. Proved.

3.2.2. Discussion of the proposed formulation

Theoretical guarantees in Theorem 1 showed that we can globally optimize the non-convex problem by optimizing the convex problem (5) under the conditions that $m \geq m^*$. Without $m \geq m^*$, we can not have the inequality (18), and can not guarantee $p^* = d^*$. Recall the result in (4), $M \leq 2r(\frac{e(n-1)}{r})^r$. Work Goldfarb and Liu (1990) suggests that for convex quadratic programming, interior point algorithm needs $O(n^3L)$ arithmetic operations, where n is the number of variables and L is the input length for the quadratic program. Since the formulated program has $2dP$ variables, the computational complexity is at most

$$O((2dP)^3) = O(d^3r^3(\frac{n}{r})^{3r}). \tag{19}$$

For fix rank r or dimensional d, the complexity is polynomial in the data size n and the number of hidden neurons m.

The key of the paper is that authors introduce M(also P in the paper) to denote the number of distinct $diag(Xu), u \in \mathbb{R}^d$ (or alternatively, the number of S_X). In this way, they can use the variable $D(S)Xu, u \in P_S$, to express $(Xu)_+$, where the non-convex activation function can be expressed with convex form. Since M is bounded above, the previous infinite problem can be expressed with the finite constraints.

3.2.3. Extension to Convolutional Neural Networks

Two-layer convolutional neural networks (CNNs) with m hidden neurons of dimension d and fully connected output layer weights can be described by patch matrices $X_k \in \mathbb{R}^{n\times d}$. And for flattened activations, $f(X_1,\dots,X_K) = \sum_{j=1}^{m}\sum_{k=1}^{K}\phi(X_k u_j)a_{jk}$.

If we consider the separate case over the patch index k, then

$$f(X_1,\dots,X_K) = \sum_{k=1}^{K}\sum_{j=1}^{m}\phi(X_k u_j)_+ a_j$$

is exactly the same as the previous two-layer ReLU network, where the complexity is the same, r is the size of filters.

If we consider the non-separate case, the dual of the primal is an semi-definite problem. The formulated convex optimization is the dual of the dual, which is

$$\min_{z_k\in\mathbb{R}^d} \frac{1}{2}\|\sum_{k=1}^{K} X_k z_k - y\|_2^2 + \beta\| \begin{bmatrix} z_1 & \dots & z_K \end{bmatrix} \|_*. \tag{20}$$

where $\|\cdots\|_*$ is the nuclear norm.

3.3. Global Search: the entire set of global optimum

Work Wang et al. (2021) follows the theorem in work Pilanci and Ergen (2020), and proposed to use mapping and construct the entire set of global optimum. Note in work Pilanci and Ergen (2020), the authors proposed to construct one global optimal neural network given the solution of (6). Authors here, different from the previous work, state that all optimal solutions can be found via the optimal solution of the convex formulation up to permutation and splitting/merging of the neurons. Also, they simplyfy the convex formulation by using w to denote both w, v in (5).

They denote the solution of convex formulation (5) $W = (w_1, w_2, \dots, w_{2p}) \in W^*$ (you can see $v_i := w_i, i \in [p+1, 2p]$. And denote $\mathcal{I} = \{i_1,\dots,i_{|W|}\} \subset [2p]$ the set of indices such that $w_i \neq 0$. Set

$$(u_j, a_j) = \left(\frac{w_{i_j}}{\sqrt{\|w_{i_j}\|_2}}, y_{i_j}\sqrt{\|w_{i_j}\|_2}\right) \tag{21}$$

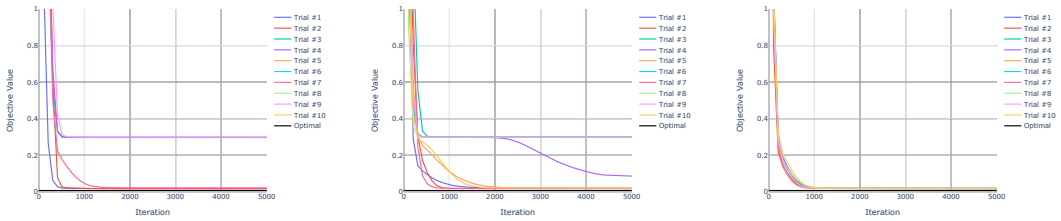
where $y_i = 1$ if $i \leq p$ and $y_i = -1$ if $i > p$. This is equivalent to theorem 1. For a neuron (u_i, a_i), one can split the neuron to more than one neurons by $(u_{i_j}, a_{i_j}) = (\sqrt{y_j}u_i, \sqrt{y_j}a_i)$ where $y_j \geq -, \sum_{j=1}^{k} y_j = 1$. And split neurons can be merged back to their original form. In this way, one can search for the entire set of global optimum.

4. Experiment Results

Source code and input data for all experiments are available in https://github.com/alwaysbyx/convexifying-nerual-networks. In all the experiments, we use the package **cvxpy** and solver **gurobi** to solve the formulated convex problem. We choose $\beta = 1e-4$.

4.1. Small dataset with $n = 5, d = 1$

We consider a one-dimensional dataset with $n = 5$, $d = 1$. The data matrix and label is randomly generated from $[-10, 10], \{-1, 1\}$ and we ensure that the same data does not point to the different label. We consider bias term in the two-layer neural network by concatenating a column of ones to the data X. We perform 10 independent experiments using stochastic gradient descent method and one experiment to solve the convex optimization problem. Fig 3 shows the loss computed by (2) using SGD (Trial index) and convex formulation (Optimal). We can see that the convex formulation achieves lowest training loss. As m is small, SGD would stuck at local minimum. As m increases, SGD is more likely to converge to the optimal value. In Fig 3, using convex formulation achieves lower objective value than differet trials using SGD. Theorem 1 guarantees that with $m \geq m^*$, where $1 \leq m^* \leq n$, the convex formulation achieve the global optimal objective. Fig 3 is consistent with the theorem.



(a) $m = 5$ (b) $m = 10$ (c) $m = 100$

Figure 3: Training loss (2) of a two-layer ReLU network trained with SGD (10 individual trials) on a one dimensional dataset ($n = 5, d = 1$), where Optimal denotes using convex formulation. SGD can be stuck at local minima for small m, while the proposed approach is optimal as guaranteed by Theorem. Here the dataset is $X = \begin{bmatrix} -1 & -5 & 2 & 0 & 3 \end{bmatrix}, Y = \begin{bmatrix} 1 & -1 & 1 & -1 & 1 \end{bmatrix}$.

4.2. Larger dataset with $d = 1$

For larger $n = 50, 100$, we choose $m = 150, 200$ for SGD, respectively. Fig 4 shows that for larger n, all the approach can not classify the data accurately (even the optimal solution achieves the objective value larger than 1). This might be the dimension of the data is too small, and two-layer neural networks can not capture the pattern between the feature and labels. Optimal solution achieves the smallest objective value, and approximate achieves similar small objective value as optimal does, but takes less time and is very efficient.

4.3. Large dataset with $n = 50, d = 2$

Here we consider the data dimension that is equal to 2. The data matrix is randomly generated and the label is generated according to its distance. To solve the problem, the optimal approach takes 1004.62 seconds, the approximate approach takes 13.71 seconds,

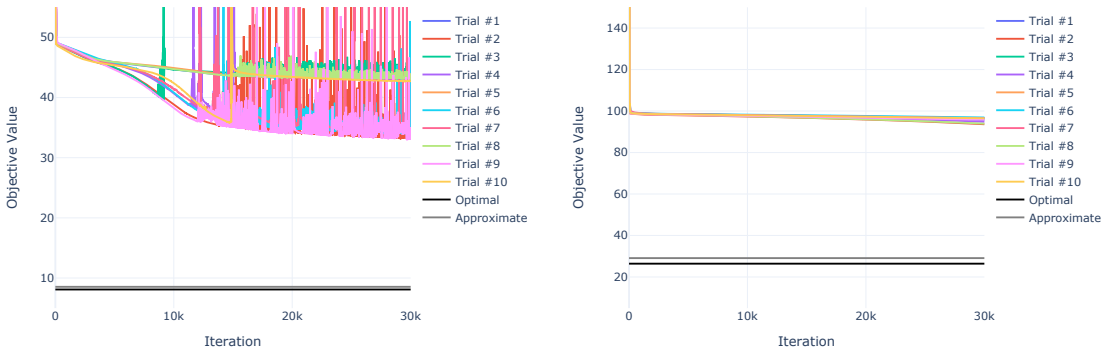


Figure 4: Training loss (2) of a two-layer ReLU network trained with SGD (10 individual trials) on a one dimensional dataset (Left: $n = 50$, Right: $n = 100$), where Optimal denotes using convex formulation, and Approximate denotes generate random $D(S_i)$ to approximate the exact all the distinct $D(S_i)$.

	Optimal		Approximate		SGD	
	Time	Objective	Time	Objective	Time	Objective
5	0.52	**0.006**	-	-	8.71	0.017
50	8.86	**8.09**	**2.85**	8.53	14.66	45.87
100	62.23	**26.41**	**3.52**	29.09	15.51	95.90

Table 2: Time cost and objective value using different approaches for different data size for $d = 1$.

and the SGD method takes 40.70 seconds. We can see that using approximate approach takes much less time but achieves similar results as the exact convex formulation.

4.4. Binary Classification

We choose class airplane and class automobile from CIFAR-10 dataset Krizhevsky et al. (2009), and down-sample the data to dimension 100 to perform the binary classification experiment here. We choose $m = 40$ and for convex formulation, we random generate u from normal distribution and get $(Xu)_+$ accordingly. The training set contains 120 samples and the test set contains 100 samples. Fig 6 shows that approximate approach achieves smallest objective value, and achieves better test accuracy than the most of the SGD trials, while approximate approach takes 13.2 seconds and SGD takes 31.4 seconds to train.

5. Discussion

In this project, we consider using convex formulation and standard gradient descent to train the neural networks via supervised learning. Convex formulation contains two method, the

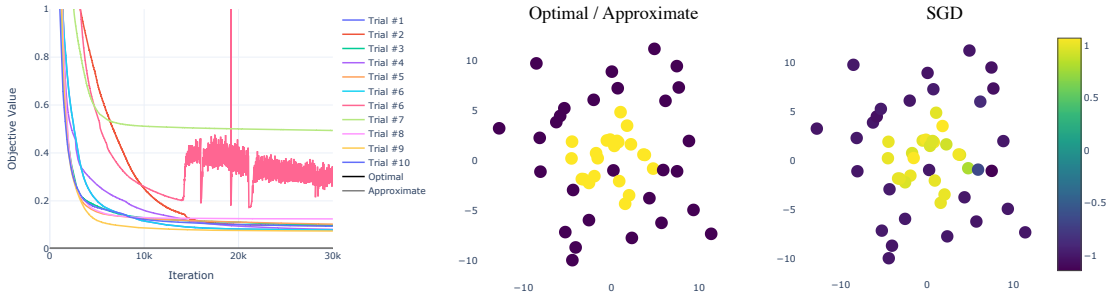


Figure 5: (Left) Training loss (2) of a two-layer ReLU network trained with SGD (10 individual trials) on a two dimensional dataset, where Optimal denotes using convex formulation, and Approximate denotes generate random $D(S_i)$ to approximate the exact all the distinct $D(S_i)$. (Right) The predicted label using different approaches, where the result using convex formulation is the same as the ground truth.

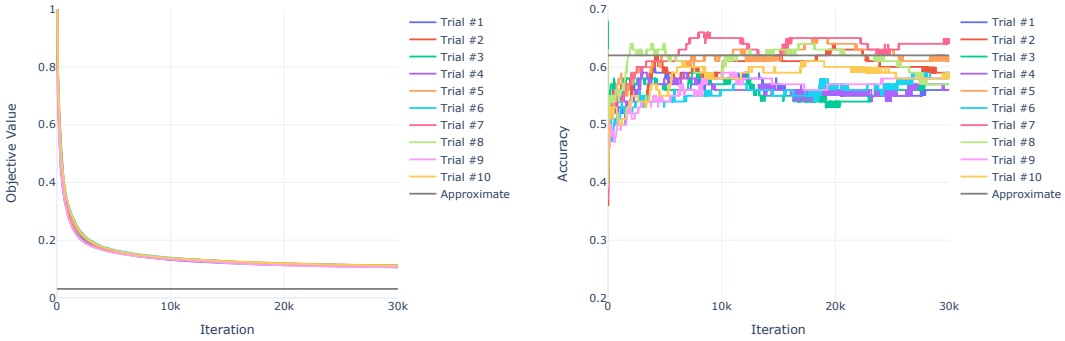


Figure 6: (Left) Training loss (2) of a two-layer ReLU network trained with SGD (10 individual trials) on a 100-dimensional dataset derived from CIFAR-10, where Approximate denotes generate random $D(S_i)$ to approximate the exact all the distinct $D(S_i)$. (Right) The test accuracy curve.

first one is exact convex formulation, that is find all the distinct $D(S_i)$ for the data X, and the second one is approximate convex formulation, which random generated normalized u and get the approximate distinct $D(S_i)$. From the experiments we find convex formulation achieve optimal objective, which is consistent with the theorem. For the large size or larger dimensional dataset, exact convex formulation can take much time to solve. Approximate convex formulation takes much less time, while maintains better performance than standard gradient descent methods. The standard gradient descent methods are simple than the convex formuation, and they can apply to larger dataset while maintain the good performance.

The drawback is that they can stuck in the local minimum for small m, different trials generate different results, and the most results are worse than the convex formulation.

When reading the literature and doing the experiments, I get a much deeper understanding of the relationship with non-convex neural network loss and convex formulation to search for the optimal minimum. In the meantime, I learn the rigorous proof and construction for the dual formulation. For experiment part, I write the code for both convex formulation and SGD, get a more thorough understanding of the advantages and disadvantages both methods. The most inspiring thing is how authors write the proof for the main theorem, and I even draw the image of the process. They ended up to proof that the dual of the dual is same as the primal, and construct convex formulation and corresponding optimal solution for the neural network is really interesting!

6. Acknowledge

As mentioned in the Sec 4, the source code and input data for all experiments are available in https://github.com/alwaysbyx/convexifying-nerual-networks. The data for section 4.1-4.3 is generated randomly, you can refer to the code for more detail. The data for section 4.4 is derived from CIFAR-10 dataset Krizhevsky et al. (2009), and you can get access to the complete dataset via https://www.cs.toronto.edu/~kriz/cifar.html. For the data used in our experiments, you can also get it in the above github repository.

References

Francis Bach. Breaking the curse of dimensionality with convex neural networks. *The Journal of Machine Learning Research*, 18(1):629–681, 2017.

Burak Bartan and Mert Pilanci. Training quantized neural networks to global optimality via semidefinite programming. In *International Conference on Machine Learning*, pages 694–704. PMLR, 2021.

Yoshua Bengio, Nicolas Roux, Pascal Vincent, Olivier Delalleau, and Patrice Marcotte. Convex neural networks. *Advances in neural information processing systems*, 18, 2005.

Simon Du, Jason Lee, Yuandong Tian, Aarti Singh, and Barnabas Poczos. Gradient descent learns one-hidden-layer cnn: Don't be afraid of spurious local minima. In *International Conference on Machine Learning*, pages 1339–1348. PMLR, 2018.

Tolga Ergen and Mert Pilanci. Global optimality beyond two layers: Training deep relu networks via convex programs. In *International Conference on Machine Learning*, pages 2993–3003. PMLR, 2021.

Donald Goldfarb and Shucheng Liu. An o (n 3 l) primal interior point algorithm for convex quadratic programming. *Mathematical programming*, 49(1):325–340, 1990.

Marco Gori and Alberto Tesi. On the problem of local minima in backpropagation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(1):76–86, 1992.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Piyush C Ojha. Enumeration of linear threshold functions from the lattice of hyperplane intersections. *IEEE Transactions on Neural Networks*, 11(4):839–850, 2000.

Mert Pilanci and Tolga Ergen. Neural networks are convex regularizers: Exact polynomial-time convex optimization formulations for two-layer networks. In *International Conference on Machine Learning*, pages 7695–7705. PMLR, 2020.

Yifei Wang, Jonathan Lacotte, and Mert Pilanci. The hidden convex optimization landscape of regularized two-layer relu networks: an exact characterization of optimal solutions. In *International Conference on Learning Representations*, 2021.