Project: Convexifying Neural Networks

Roshan Karande
rskarande@ucsd.edu

Introduction

Deep Neural Networks (DNNs) have gained popularity due to their impressive empirical performance in many machine learning applications. Despite their widespread use in machine learning, there are significant gaps in our theoretical understanding of neural nets. The non-convexity of training DNNs poses a substantial theoretical challenge. Unlike convex optimization problems, theoretical guarantees are often weak or non-existent In this project, we are going to study the convex representation of training two layer neural networks with ReLU activation, and present numerical simulations verifying the theoretical claims.

1

1.a Describe the problem setting and the objective.

- The authors have developed exact representations of training two layer neural networks with ReLU activation function in terms of single convex program. These results have been derived by developing a novel approach in duality theory.
- In this problem, we have a two layer network $f : \mathbb{R}^d \to \mathbb{R}$ with **m** neurons in the hidden layer.

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

$x \in \mathbb{R}^d$ - input to neural network
$u_j \in \mathbb{R}^d$ - weight corresponding to j^{th} hidden neuron in layer 1
$\alpha_j \in \mathbb{R}$ - weight corresponding to j^{th} neuron in layer 2 (output layer)
$\phi(t) = (t)_+ := \max(t, 0)$ is the ReLU activation

- The objective is to find the model that correctly maps input to output $f(x)$ such that loss associated with incorrect mapping (squared loss) is minimized.

1.b Specify the problem parameters and the neural network architecture used.

- The neural network architecture used consists of 2 layers (1 hidden layer and 1 output layer).
- The input is d dimensional and the output is 1 dimensional
- The non-linear activation function, which is used only in the hidden layer is ReLU.

1.c What is the non-convex optimization problem used to train the network?

$$p^* := \min_{\{\alpha_j, u_j\}_{j=1}^m} \frac{1}{2} \| \sum_{j=1}^{m} (Xu_j)_+ \alpha_j - y \|_2^2 + \frac{\beta}{2} \sum_{j=1}^{m} \left(\|u_j\|_2^2 + \alpha_j^2 \right) \tag{2}$$

- $X \in \mathbb{R}^{n \times d}$ is data matrix. There are n samples of dimension d. One sample is represented in one row of X
- $y \in \mathbb{R}^n$ is the label vector. y_i is the label corresponding to input in row i of X
- $u_j \in \mathbb{R}^d$ - (parameters) weight corresponding to j^{th} hidden neuron in layer
- $\alpha_j \in \mathbb{R}$ - (parameters) weight corresponding to j^{th} neuron in layer 2 (output layer)
- $\phi(t) = (t)_+ := \max(t, 0)$ is the ReLU activation
- $\beta > 0$ is the regularization parameter .
- The objective is to minimize the squared loss objective and squared ℓ_2-norm of all parameters

A more compact representation of the above problem is given below -

$$\begin{aligned} p^* := \min \quad & L\left(\phi\left(XW_1\right)W_2, y\right) + \beta \left(\|W_1\|_F^2 + \|W_2\|_F^2 \right) \\ & W_1 \in \mathbb{R}^{d \times m} \\ & W_2 \in \mathbb{R}^{m \times 1} \end{aligned} \tag{3}$$

1.d Why is this problem non-convex?

- The non-linear activation function applied between two Weight matrices makes the the problem non-convex. Alternatively, the ReLU acting between hidden layer u_j and α_j makes is non-convex.

1.e What kind of loss is used?

- The loss that is used is L_2 norm of the difference of output of neural network and true labels of corresponding to the inputs i.e. squared error loss
- In addition regularization is also accounted in the loss which is the ℓ_2-norm of all parameters

1.f What kind of regularization is used and what is the rationale for this choice?

- The regularization that is used here is it prevents the parameters from becoming very large.
- Conventional theory suggests that using regularization prevents from overfitting the training data and promotes generalization.
- However, the paper also mentions a case when regularization is not used at all i.e. $\beta \to 0$

2

2.a What is the convex problem authors propose to solve in [1]?

The authors propose to solve the following finite dimensional convex problem -

$$\begin{aligned} p^* := \min_{\{v_i, w_i\}_{i=1}^P} & \frac{1}{2} \| \sum_{i=1}^{P} D_i X \left(v_i - w_i\right) \quad - y \|_2^2 + \beta \sum_{i=1}^{P} \left(\|v_i\|_2 + \|w_i\|_2 \right) \\ \text{s.t. } & \left(2D_i - I_n\right) X v_i \geq 0, \left(2D_i - I_n\right) X w_i \geq 0, \forall i. \end{aligned} \tag{4}$$

2.b How was the non-convex problem converted into a convex problem?

The problem was converted into its convex representation by using duality theory. The authors use weak duality and strong duality to get to the final form of convex problem.

- The authors use $l1$ penalized representation of the problem in 4. They state that $l1$ penalized representation is equivalent to the original problem.

Thus the equation that needs to be converted into its dual is

$$p^* = \min_{\substack{\|u_j\|_2 \leq 1 \\ \forall j \in [m]}} \min_{\{\alpha_j\}_{j=1}^m} \frac{1}{2} \left\| \sum_{j=1}^m (Xu_j)_+ \alpha_j - y \right\|_2^2 + \beta \sum_{j=1}^m |\alpha_j| .$$

Interchanging the order of min and $\max$, gives the lower-bound d^* via weak duality

$$p^* \geq d^* := \max_{\substack{v \in \mathbb{R}^n \text{ s.t.} \\ \left|v^T (Xu)_+\right| \leq \beta \forall u \in \mathcal{B}_2}} -\frac{1}{2}\|y - v\|_2^2 + \frac{1}{2}\|y\|_2^2. \tag{5}$$

- The inner minimization problem is converted into its dual form.
- The dual form of the inner problem gives a max form problem.
- Thus we get a min-max problem which can be dualized by converting into a max-min problem (swapping min-max)
- Thus we get a convex problem that falls in the category of semi-infinite optimization problems as it has finite variables and infinite constraints.
- The optimal value of the thus obtained d^* would be at most p^* by weak duality
- The authors go further to prove that strong duality holds in this case i.e. $d^* = p^*$ under certain condition. Thus the optimal value of the primal problem can be obtained by solving the dual problem.
- The final form that is obtained in mentioned in 4.

2.c What kind of convex optimization problem is it?

- The intermediate problem in 5 is semi-infinite optimization problem (one with finite variables and infinite constraints)
- The final form that is obtained 4 is a constrained finite dimensional optimization problem with $d * 2P$ and $n * 2P$ linear inequality constraints.

3

3.a How do the authors propose to train the neural network?

- The authors perform numerical experiments on a simpler dataset - one with 5 samples each 1 dimensional and add bias term.
- The authors then repeat these experiments for different values of inputs, dimensions and number of hidden neurons.
- The neural network used is a 2 layered neural network with ReLU activation function and method used is stochastic gradient descent.

3.b What are the theoretical guarantees given in [1]? Under what conditions do these guarantees hold?

- Strong duality as mention previous holds under a certain condition. The condition being, $m \geq m*$.
- If this condition is met, then both the problems have same optimal values.

- This guarantees that solving the convex problem will final the global optimal of the 2-layered neural network with m neurons.
- The authors mention this unlike local search heuristics like back propagation.
- Furthermore, the authors state in theorem 1 state that the optimal solution to non-convex problem 2 can be constructed from the optimal solution to 4 as follows

$$\begin{aligned}
\left(u_{j_{1i}}^*, \alpha_{j_{1i}}^*\right) &= \left(\frac{v_i^*}{\sqrt{\|v_i^*\|_2}}, \sqrt{\|v_i^*\|_2}\right) \quad \text{if} \quad v_i^* \neq 0 \\
\left(u_{j_{2i}}^*, \alpha_{j_{2i}}^*\right) &= \left(\frac{w_i^*}{\sqrt{\|w_i^*\|_2}}, -\sqrt{\|w_i^*\|_2}\right) \quad \text{if} \quad w_i^* \neq 0
\end{aligned} \tag{6}$$

where $\{v_i^*, w_i^*\}_{i=1}^P$ are the optimal solutions to 4

3.c What is the computational complexity? Is the complexity polynomial or exponential in problem size?

- The computational complexity is at the most $O\left(d^3 r^3 \left(\frac{n}{r}\right)^{3r}\right)$ using standard interior-point solvers.
- Here $r = rank(X) \leq min(n, d)$
- The complexity is polynomial in n and m for fixed rank r
- It is exponential in r thus exponential in d for a full rank matrix X i.e. $r = d$ (which would usually be the case)
- The paper says that this is an exponential improvement over the existing SOTA approaches.

4

4.a Note that in the convex formulation, the inner sum consists of P terms (instead of m terms as in the non-convex problem). What does P denote?

- Consider an arbitrary vector $u \in \mathbb{R}^d$.
- The number of distinct partitions that the hyperplane passing through origin characterized by u can create of the is denoted by P.
- Thus, P is the number of regions in a partition of $\mathbb{R}^d$ by hyperplanes passing through the origin, and are perpendicular to the rows of X.
- Also the total number of distinct partitions is limited by $P \leq 2\sum_{k=0}^{r-1} \begin{pmatrix} n-1 \\ k \end{pmatrix} \leq 2r\left(\frac{e(n-1)}{r}\right)^r$ for $r \leq n$ where $r = rank(X)$

4.b What does each matrix Di, i = 1, . . . , P correspond to in this setting?

- diagonal matrices are formed such that $\text{Diag}(1[Xu \geq 0])$ where $u \in \mathbb{R}^d$ is arbitrary and $1[Xu \geq 0] \in \{0,1\}^n$ is an indicator vector with Boolean elements $\left[1\left[x_1^T u \geq 0\right], \ldots, 1\left[x_n^T u \geq 0\right]\right]$.
- If we enumerate all such distinct diagonal matrices that can be obtained for all possible $u \in \mathbb{R}^d$, and denote them as $D_1, \ldots, D_P$
- P thus represents total number of distinct partitions that are possible and diagonal matrices are enablers in this partition.

4.c Why do you think such matrices Di show up in the convex problem? Explain.

- As stated previously, P represents total number of distinct partitions that are possible and diagonal matrices are enablers in this partition.
- These diagonal matrices show up because of the special activation function ReLU that is used which can be seen as an enabler / clamper like these diagonal matrices.
- The authors also state that had it been some other activation function, these special diagonal matrices might not have emerged.

5

5.a How does the optimization problem change if the underlying network architecture is a Convolutional Neural Network (CNN)?

- The authors introduce extensions to the previous approach to Convolutional neural networks (CNNs).
- It is mentioned that, for a 2 layer CNN with m hidden neurons, input (image / tensor) can broken into K patch matrices each of dimension d.
- Thus, each input/image can be represented using K patch matrices $X_1, X_2, \ldots, X_K$ where each $X_i \in \mathbb{R}^{n \times d}$.
- In the 2-layer CNN, the hidden layer contains m hidden neurons each representing a CNN filter.
- $u_j \in \mathbb{R}^d \quad j = 1, \ldots, m$ represents weights corresponding to j the hidden neuron (or filter). each filter, the hidden layer outputs K vectors each of length n given by $X_k u_j \quad k = 1, \ldots, K$.
- The hidden layer output, for each filter index $j = 1, \ldots, m$, passes through a activation activation layer resulting in $\phi(X_k u_j) \in \mathbb{R}^n \quad k = 1, \ldots, K$. The final output layer is a fully connected layer with weights represented by $\{\alpha_{jk} \in \mathbb{R} \quad j = 1, .., m; \quad k = 1, .., K\}$

$$f(X_1, \ldots, X_K) = \sum_{j=1}^{m} \sum_{k=1}^{K} \phi(X_k u_j) \alpha_{jk}$$

- The authors consider 3 cases
  1. **Separable ReLU Convolutional Networks** - In this case the problem problem gets reduced to fully connected case 2
  2. **Linear Convolutional Networks** - In this case with linear activations, strong duality holds and the dual reduces to the semi definite program (which is the nuclear norm penalized convex optimization problem)
  3. **Linear Circular Convolutional Networks** - In this case, when patches are padded with enough zeros and extracted with stride one, SDP reduces to one in previous case but under a DFT transformation.

6

6.a Study the paper [2] and the theoretical guarantees concerning the convex problem in Theorem 1. What does it say? How are those guarantees different from the ones given in [1]?

- In paper 1, the authors present 1 optimal mapping from the solutions of convex program to parameters of the non-convex program (2-layer ReLU neural network in this case)
- In paper 2, this idea is further extended.
- Following theorem is stated by the authors in paper 2 - All optimal solutions of non-convex program can be found from optimal solutions of convex up to permutation and neuron splitting. Hence, the optimal set of non-convex program is convex up to equivalence.

7

7.a Implement the first experiment from Fig 2 in [1]. Generate the data randomly. In particular for a small n (the sample size) and m = 5 (small number of hidden neurons), use the Stochastic Gradient Descent method to solve the non-convex problem with different initializations.

7.b Plot the training objective value vs the number of iterations. Next, solve the convex program with CVX/CVXPY and plot its optimal value.

7.c What do you observe? Does the SGD always converge to the optimal value of the convex program?

- We observe that the training loss of convex optimizer is always below SGD based methods.
- In spite of the sampling method that we use for diagonal matrices (sign pattern generation), the convex optimizer model does very well.

7.d Perform the same experiment with larger values of m.

7.e Do your observations agree with the theoretical guarantees?

- Yes the observations are in agreement with the theoretic guarantees.
- If SGD based methods are allowed to run on large datasets for long time, usually they converge to the minimum that is found by convex optimization programs.
- However, sometimes the non-convex approach get stuck at local minimal, especially when the data is relatively less.

8

8.a Replicate this experiment for a larger value of n and an appropriate m.

8.b Plot the objective value against the number of iterations.

8.c How does this experiment differ from the case for small n?

- For large values, it is seen that the variance of SGD reduces and they approach the optimal point.
- Also for large values, the results are in sync with the popular fact that SGD methods in non-convex settings(neural network) tend to find approach the optimizer.

8.d In addition to the original convex program, solve the approximate convex program described in Remark 3.3.

8.e Compare the objective value of the approximate version with both SGD and convex program.

1. We see that for larger values of n, the objective value of SGD is still higher than that of convex optimization method.
2. However, it might happen that with higher number of neurons, the non-convex method might do much better.
3. The authors also mention in one of the talks that for deep neural network, the convex optimization counterpart is rather slow.

8.f What is the advantage of solving the approximate convex program?

- The approximate convex optimisation is much faster to solve than the original convex program.
- As the number of samples and dimension of those samples grow, it becomes very difficult to enumerate all the possible sign patterns (partitions).

- The approximate convex optimization problem helps to overcome this issue.
- As stated earlier, it can be seen that the approximate convex program still performs very well as compared to neural network approach.
- It is surprisingly fast and more accurate, especially for small sized problems.

9

9.a Perform an experiment for the binary classification task on CIFAR-10 dataset. Choose any two classes you prefer. Choose the value of m appropriately. Train the network using SGD on the non-convex problem for different initializations. Solve the approximate convex program with CVX/CVPY. Plot objective value vs iterations. Evaluate the performance of trained networks on the test set, and plot test accuracy vs iterations. Comment on your observations.

10

Q10 Summarize your overall conclusions about what you learned from this project.

- Convexifying neural networks is a very recent development in the literature of machine learning.
- We see that the mathematics is much more involved. It is so easy to train a neural network, however, to get an understanding of what happens can be very complex.
- However, it is important to study, optimization from a mathematical point of view as it might potentially have benefits / uncover perspectives that were not known before.
- For example, I did not know that backpropagation (gradient descent) on the non-convex loss is a heuristic for the convex program. The authors also discovered, what happens when there is no regularization used in the original non-convex program.
- It can be seen that theory of non-convex optimization and its relation to convex optimization is a new field. In the field of deep learning, the practice seems to be much more evolved than the theory.
- Theory of duality and its applications can be very involved.
- I was listening to a presentation of one of the authors, and they are already working on seeing how this can be applied to deep neural network. They say that it is possible using layer wise approach. Furthermore, it was stated that more efficient convex optimization methods can be developed that target this particular setting.

A few personal general remarks

- The theory of duality seems to be interesting, however quite complex.
- I was wanting to spend more time on understanding duality. I watched a few courses, but still it was hard to follow up with the proofs.
- Overall, optimization seems to be a very interesting topic but needs much more time
- I found that convex solvers are not easy to use / debug. If you get it right you are good. However, I was planning to use ECOS solver with another interface and got different results. Had a hard time debugging and finally settled on python.

11 Conclusion

- We can train ReLU Neural Networks using convex optimization techniques using appropriate transformations. Thus convex optimization theory and solvers can be applied.
- Neural networks seek sparsity

- layer ReLU neural networks are convex in higher dimensions.
- Architecture search corresponds to regularizer search in the convex equivalent problem.
- Faster algorithms are need to solve high-dimensional convex programs whose solutions are sparse.

References

[1] Pilanci, Mert, and Tolga Ergen. "Neural networks are convex regularizers: Exact polynomial-time convex optimization formulations for two-layer networks." Interna- tional Conference on Machine Learning. PMLR, 2020.

[2] Pilanci, Mert, and Tolga Ergen. "Neural networks are convex regularizers: Exact polynomial-time convex optimization formulations for two-layer networks." Interna- tional Conference on Machine Learning. PMLR, 2020.

[3] `github.com/pilancilab` - code reference