ESANN 2022 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 5-7 October 2022, i6doc.com publ., ISBN 978287587084-1. Available from http://www.i6doc.com/en/.

Constraint Guided Gradient Descent: Guided Training with Inequality Constraints

Quinten Van Baelen[1,2,3] and Peter Karsmakers[1,2,3] *

1-KU Leuven, Dept. of Computer Science, ADVISE-DTAI,
Kleinhoefstraat 4, B-2440 Geel, Belgium.
2-Leuven.AI - KU Leuven institute for AI.
3-Flanders Make - DTAI-FET.
{quinten.vanbaelen,peter.karsmakers}@kuleuven.be

Abstract. Deep learning is typically performed by learning a neural network solely from data in the form of input-output pairs ignoring available domain knowledge. In this work, the Constraint Guided Gradient Descent (CGGD) framework is proposed that enables the injection of domain knowledge into the training procedure. The domain knowledge is assumed to be described as a conjunction of hard inequality constraints which appears to be a natural choice for several applications. Compared to other neuro-symbolic approaches, the proposed method converges to a model that satisfies any inequality constraint on the training data and does not require to first transform the constraints into some ad-hoc term that is added to the learning (optimisation) objective. Under certain conditions, it is shown that CGGD can converge to a model that satisfies the constraints on the training set, while prior work does not necessarily converge to such a model. It is empirically shown on two independent and small data sets that CGGD makes training less dependent on the initialisation of the network and improves the constraint satisfiability on all data.

1 Introduction

Machine learning and especially deep learning are successful in many research areas. In most cases, supervised learning is employed that, based on example input-output pairs, automatically finds a function that relates the input to the corresponding output data. However, available domain knowledge is typically ignored requiring it to be rediscovered by the learning algorithm. When domain knowledge can be inserted during the learning stage, it is expected that learning becomes more efficient, meaning that less example pairs are required to let a model represent the desired relation.

This study restricts itself to the use of a conjunction of hard inequality constraints. Hence, models should satisfy all imposed inequality constraints for all the data (even for unseen data, not used during learning, the model should satisfy the constraints). This work proposes a novel algorithm Constraint Guided Gradient Descent (CGGD), which adds supervision to the learning cycle by means of hard inequality constraints. CGGD aims at solving the potential numerical problems and the crispness issues that occur in previous work. Moreover, in CGGD

*This research received funding from the Flemish Government (AI ResearchProgram). This research has received support of Flanders Make.

ESANN 2022 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 5-7 October 2022, i6doc.com publ., ISBN 978287587084-1. Available from http://www.i6doc.com/en/.

the constraints do not need to be differentiable, and they provably dominate the gradient of the loss function during training when they are not satisfied.

There are two main classes of approaches that enable injecting constraints in the training procedure. The first class of approaches uses fuzzy-logic [1, 2]. Here, the constraints are replaced by almost everywhere smooth functions. As said in [3], this approach has as its main downside that this transformation typically leads to a loss of the crisp formulation of the constraints. Additionally, there can occur numerical problems when optimising the new objective. For example, the gradients of the loss function and the constraints can cancel out each other. However, `CGGD` solves both the crispness issue as well as the vanishing gradient phenomenon.

The second class of approaches can be summarised as using (probabilistic) logic reasoning in order to define gradients for training the network and/or as regularisation. The constraints in this setting are logical formulas, where the variables in the formulas correspond to Boolean, probabilistic or discrete output variables of the network. The methods that are most related to `CGGD` are: `NeuroLog` [4], `DeepProbLog` [5], and the semantic loss [3]. Each method does not require the theory to be differentiable, but uses results from reasoning on the logic theory to tune the gradient with which the network is updated. All three methods are not applicable in the setup of this work because adjusting the truth value of an inequality constraint requires an additional reasoning mechanism.

The main contributions of this work are: (a) the design of the novel `CGGD` method that learns a neural network model for a regression task while satisfying a conjunction of hard inequality constraints, (b) the empirical observation that incorporating prior knowledge in terms of inequality constraints can make learning less dependent on the initialisation of the model parameters.

2 Constraint Guided Gradient Descent

This work targets an algorithm that searches for the weights of a neural network Φ by optimising some loss function L while letting the network satisfy a fixed finite set of predefined hard inequality constraints $\{C_i\}_{i=1}^{N}$ on the training set. More formally, this can be expressed as the constrained optimisation problem:

$$\begin{aligned} \underset{\boldsymbol{W}}{\text{argmin}} \quad & L(\boldsymbol{x}, \Phi(\boldsymbol{x}), \boldsymbol{y}, \boldsymbol{W}) \\ \text{s.t.} \quad & \forall (x, \Phi(x)) \in (\boldsymbol{x}, \Phi(\boldsymbol{x})) : C_i(x, \Phi(x)) \leq 0, \text{ for } i = 1, \ldots, N. \end{aligned}$$

In the previous equation, $\boldsymbol{x}$ and $\boldsymbol{y}$ denote a set of input vectors and output vectors respectively, x and y denote a single input vector and output vector respectively, $\Phi(x)$ denotes the predictions of the network as well as any prediction of any hidden layer, and $\boldsymbol{W}$ denotes the collection of trainable weight matrices of the model. The set of models that satisfy all constraints for a set of training examples is called the **feasible region** FR. `CGGD` aims at finding a model in FR that locally minimises L.

ESANN 2022 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 5-7 October 2022, i6doc.com publ., ISBN 978287587084-1. Available from http://www.i6doc.com/en/.

The constrained optimisation problem is solved by optimising the loss function with gradient descent and adjusting the update step according to the constraints in case they are not satisfied. When some constraints are not satisfied, then for each unsatisfied constraint a direction is computed to move to in order to satisfy the constraint eventually. Hence, the update step for a trainable parameter w is defined by

$$w^{(i+1)} := w^{(i)} - \eta_i(\nabla L(\Phi(x)) + 1.5 \ \vec{dir}(C(x, \Phi(x))) \max\{\varepsilon, \|\nabla L(\Phi(x))\|\}), \quad (1)$$

where η_i denotes the step size for iteration i, $\vec{dir}(C(x, \Phi(x)))$ denotes the direction corresponding to the constraints $C := \{C_i\}_{i=1}^N$, 1.5 is a factor that is referred to the rescale factor that controls the relative weight of the constraints compared to the gradient of the loss function, $\|\cdot\|$ denotes the L_2-norm, and $\varepsilon > 0$ is a lower bound for the relative weight compared to the gradient of the loss function to allow to move past local optima outside FR. Note that the proposed update step (1) does not introduce a new hyperparameter that needs to be chosen correctly, and the rescale factor is set larger than 1, which is equivalent with the constraints being more important than the loss function.

The following assumption is needed to guarantee convergence when the constraints do not have any influence on the optimisation procedure at some point in time and onwards, for example when the initialisation and every point in the optimisation procedure are in FR.

Assumption 1. Let $L : \mathbb{R}^n \to \mathbb{R}$ satisfy conditions needed to let a non-convex optimisation algorithm based on gradient descent converge to a local solution.

The main result of this paper is stated now.

Theorem 2. Let $L : \mathbb{R}^n \to \mathbb{R}$ be a loss function satisfying Assumption 1 and for which ∇L is M-Lipschitz continuous. Consider the inequality constraints $\{C_i\}_{i=1}^N$ for some strictly positive integer N. Let $\vec{dir}(C(x, \Phi(x)))$ be the direction of the shortest path with respect to the Euclidean distance from FR to w for $w \in \mathbb{R}^n \setminus FR$. Then, there exists a sequence $\{\eta_j\}_j$ such that the iteration procedure defined by applying (1) converges to a point in the closure of FR.

The proof[1] of this theorem consists of (i) showing that the size of the update step can be decreased over different iterations by decreasing η_i as a function of ε and $\|\nabla L\|$, and (ii) showing that the point obtained from one iteration is closer to the feasible region than the previous point. The direction of the constraints being defined by the shortest path to FR is a sufficient condition but not a necessary condition. For example, if $FR = [1, 2] \cup [3, 4]$. Then the direction of the constraint can be chosen as -1 for $w < 3$ and 1 for $w > 4$. This leads to `CGGD` converging to $w \in [3, 4]$ when initialised at $w = 2.1$.

An example is given to illustrate the importance of Theorem 2. Let $L : \mathbb{R} \to \mathbb{R} : w \mapsto (w-2)(w-4)(w-3)(w-1.5)(w-1)(w-2.75)(w-5)^2 + 7$.

[1] A full proof can be found at `https://arxiv.org/abs/2206.06202`.

ESANN 2022 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 5-7 October 2022, i6doc.com publ., ISBN 978287587084-1. Available from http://www.i6doc.com/en/.

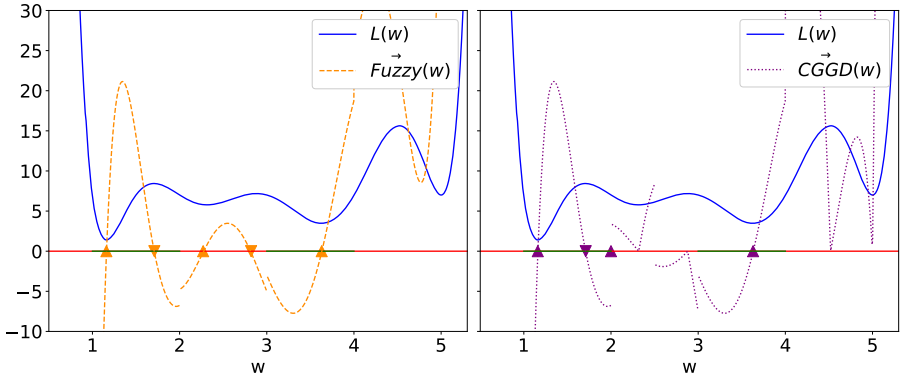

Figure 1: Loss function L with the gradient of the fuzzy loss function $\vec{Fuzzy}$ (left) and the CGGD update step $\vec{CGGD}$ (right). The local solutions of the optimisation procedure are indicated with triangles on the horizontal axis. The triangles pointing upwards and downwards indicate if the convergence is stable or not, respectively. The convergence is not stable when it can only converge if initialised at this point. The feasible region is shown in green on the horizontal axis.

Suppose that the constraint is given by $(w-1)(w-2)(w-3)(w-4) \leq 0$ for $w \in \mathbb{R}$. This leads to the feasible region being $[1,2] \cup [3,4]$. From the visualisation of L in Figure 1 is clear that the local minima are given by $w \approx 1.16$, $w = 2$, $w \approx 3.63$. Moreover, Figure 1 illustrates the value of the update steps for a fuzzy loss function, which adds the constraint as regularisation term to the learning objective before optimising with gradient descent, and CGGD. Note that a fuzzy approach requires the constraints to be almost everywhere differentiable, while this is not necessary for constraints in CGGD. From determining the points where the update step is equal to 0 or the sign of the update step is negative to the left and positive to the right of the point in case of a discontinuity, it follows that the fuzzy approach can converge to $w \approx 1.16$, $w \approx 1.71$ (when initialised at this point), $w \approx 2.27$, $w \approx 2.82$ (when initialised at this point), and $w = 3.63$, while CGGD can converge to $w \approx 1.16$, $w \approx 1.71$ (when initialised at this point), $w = 2$ and $w \approx 3.63$. This illustrates the fact that the gradient of a fuzzy loss function can vanish even when constraints are not satisfied. While the points that can be obtained as convergence points of CGGD satisfy the constraints.

Another major difference with fuzzy approaches that optimises an objective function with gradient descent require almost everywhere differentiable constraints, while in CGGD the constraints can be non-differentiable for a set of strictly positive measure. For example, consider the constraint $-2 \leq C(w) \leq 2$, where $C : \mathbb{R} \to \mathbb{R} : w \mapsto w\chi_{\mathbb{Q}}(w) - w\chi_{\mathbb{R}\setminus\mathbb{Q}}(w)$ with χ_A the indicator function on the set A. Observe that this function is only continuous in $w = 0$. Therefore, it is not almost everywhere differentiable. Note that the direction of the shortest path for CGGD can be taken -1 if $w < -2$ and 1 if $w > 2$.

ESANN 2022 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 5-7 October 2022, i6doc.com publ., ISBN 978287587084-1. Available from http://www.i6doc.com/en/.

3 Experiments

The presented method, CGGD, is tested for its performance compared to two baselines[2]. The size of the data sets is 750 examples. The division into training, validation and test set is 200/250/250. The first baseline (Baseline) is the model trained without any constraints. The second baseline (Fuzzy) is obtained using the loss function used in DL2 [1]. The training procedure discussed for DL2 is not used, since it is not feasible to adjust it to the constraints considered here. Each setup is repeated 4 times with different initialisations of the network, and the mean and standard deviation of each metric are reported.

The first data set is the Bias Correction[3] (BC) data set. The task is to predict the maximal and minimal temperature of the next day given some information of the current day. The constraints considered for this data set are: upper and lower bounds on the values for both the minimal temperature and the maximal temperature, and the constraint that the minimal temperature should be smaller or equal than the maximal temperature.

The second data set is the Family Income[4] (FI) data set. The task is to predict certain expenses of a family given information about the household income and some information about the properties owned by the household such as the number of personal computers. Also for this data set, upper and lower bounds are set on all the predicted values. Moreover, the total food expenditure prediction should be larger than the sum of the prediction of the bread and cereals, the meat, and the vegetables expenditure. The last constraint is that the total income of the family (input) should be larger or equal than the sum of all the expenses.

While CGGD can be more generically applied to different architectures, in this work, only dense neural networks are considered. The hidden layers have ReLU activation functions and the final layer has a linear activation function. All networks are trained and tested using the Means Squared Error (MSE) as loss function. The satisfaction ratio (SR) is introduced as a metric to indicate how many constraints are satisfied. The satisfaction ratio is the ratio between the total number of satisfied constraints and the total number of constraints.

4 Results

The results of the experiments are shown in Table 1. The experiments show that the proposed method has less problems with having a decent or good performance for small training sets compared to the other methods. In particular, the results indicate that CGGD seems to depend less on the initialisation of the network because for the BC data set the performance of some initialisations result in a poor performing network for training without constraints. For the FI data set it appears that the initialisations were rather good which resulted in

[2]See https://github.com/KULeuvenADVISE/CGGD for the code of the experiments.

[3]Available on https://archive.ics.uci.edu/ml/datasets/Bias+correction+of+numerical+prediction+model+temperature+forecast [6].

[4]Available on https://www.kaggle.com/grosvenpaul/family-income-and-expenditure.

ESANN 2022 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 5-7 October 2022, i6doc.com publ., ISBN 978287587084-1. Available from http://www.i6doc.com/en/.

	BC		FI	
Method	MSE	SR	MSE	SR
Baseline	0.7441±0.5593	74.90±10.21	**0.0012±0.0001**	98.69±0.22
Fuzzy	0.0129±0.0049	99.52± 0.48	0.0066±0.0020	95.87±0.26
`CGGD`	**0.0079±0.0084**	**99.96± 0.05**	0.0017±0.0005	**99.89±0.12**

Table 1: The mean and standard deviation for the mean squared error (MSE) and satisfaction ratio (SR) for the Bias Correction (BC) data set and the Family Income (FI) data set. The best results for each setup are shown in bold.

only a small difference in performance. This phenomenon was also shown in the one-dimensional examples in Section 2. The main reason for this phenomenon to occur is that loss functions of neural networks are known to be highly non-convex, which increases the likelihood of a vanishing gradient in fuzzy approaches.

5 Conclusion

The proposed method, `CGGD`, enables the use of a conjunction of hard inequality constraints during the learning cycle of neural networks. The method succeeds in fixing the crispness issue by not transforming the constraints, and the vanishing gradient phenomenon by including a rescale factor that is strictly larger than 1 and the lower bound on the norm of the gradient of the loss function. For the purpose of regression on small data sets, the performance in terms of mean squared error and constraints satisfiability was empirically verified on two data sets. The loss was comparable to the other approaches, but the satisfiability of the constraints was always the highest for `CGGD`.

References

[1] Marc Fischer, Mislav Balunovic, Dana Drachsler-Cohen, Timon Gehr, Ce Zhang, and Martin Vechev. DL2: Training and querying neural networks with logic. *36th International Conference on Machine Learning, ICML 2019*, 2019-June:3411–3427, 2019.

[2] Stephen H. Bach, Matthias Broecheler, Bert Huang, and Lise Getoor. Hinge-Loss Markov random fields and probabilistic soft logic. *Journal of Machine Learning Research*, 18:1–67, 2017.

[3] Jingyi Xu, Zilu Zhang, Tal Friedman, Yitao Liang, Guy Van Den Broeck, and Guy Van Den Broeck. A semantic loss function for deep learning with symbolic knowledge. *35th International Conference on Machine Learning, ICML 2018*, 12:8752–8760, 2018.

[4] Efthymia Tsamoura, Timothy Hospedales, and Loizos Michael. Neural-symbolic integration: A compositional perspective. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35 (6), pages 5051–5060, 2021.

[5] Robin Manhaeve, Sebastijan Dumančí, Angelika Kimmig, Thomas Demeester, and Luc De Raedt. Neural probabilistic logic programming in deepproblog. *Artificial Intelligence*, 298:103504, 2021.

[6] Dongjin Cho, Cheolhee Yoo, Jungho Im, and Dong Hyun Cha. Comparative assessment of various machine learning-based bias correction methods for numerical weather prediction model forecasts of extreme air temperatures in urban areas. *Earth and Space Science*, 7, 4 2020.