Guaranteeing Deep Neural Network Outputs in a Feasible Region

Hiroshi Maruyama

Preferred Networks, Inc.
Tokyo, Japan
hm2@preferred.jp

***Abstract*—A deep neural network (DNN) maps a point in the R^m input space into a point in the R^n output space. Any point in the output space may appear depending on the combination of the training data set, the input data point, and the hyper parameters of the DNN. In real applications, some of these output points may not be feasible solutions and should not be used, for example, as an input to a safety-critical system. We propose a post-DNN transformation that maps the R^n into the feasible space. By making this transformation differentiable, the teacher signals can be applied in this transformed space.**

***Index Terms*—machine learning, safety, feasible solutions**

I. Introduction

Let us consider a deep neural network (DNN) that controls a machine, such as a drone. The drone has a set of sensors and depending on the sensor input, the controller DNN calculates the signal inputs to the on-board electric motors. Here, we consider that the control signals must satisfy a certain set of conditions. We assume that the control signal is represented as a 3-D space point $p =< x, y, z >$ (called reference point) where the drone should move to. In a confined area (such as an indoor room), the reference point p must be in the interior of some region S, that is, p must satisfy $p \in S$ (see Fig. 1). We call such p a *feasible solution* and the set S the *feasible region.*

Our goal of this work is to architect the DNN so as to produce only feasible solutions regardless the input data point, the training data set, nor the hyper parameters.

II. Policy Filter

In our examples below we assume $n = 2$, that is, the output of our DNN is in the 2-dimensional Euclid space but the same argument applies to any $n > 1$. We further assume that the feasible region is a convex set.

Figure 2 shows an example. The blue convex set shows the feasible region and the red dots represents possible outputs from our DNN (either in the training time or in the inference time). Note that some red points fall outside of the feasible region.

The simplest way for guaranteeing that the output is always feasible is to remove infeasible solutions by a *policy filter* as shown in Fig. 3.

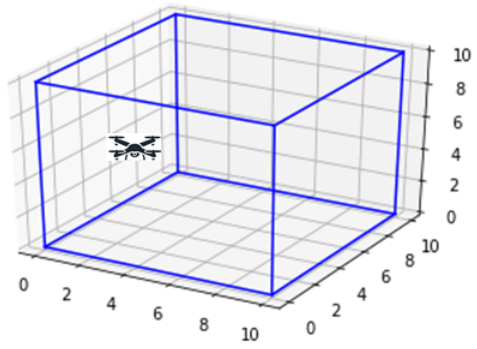

Fig. 1. Feasible Region

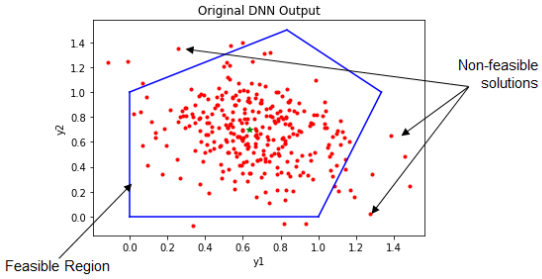

Fig. 2. Possible DNN Outputs and Feasible Region

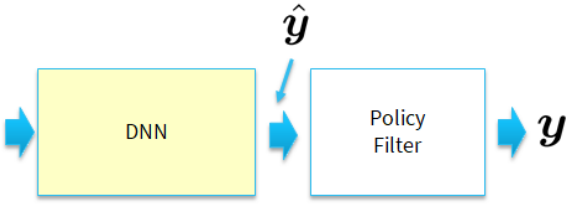

Fig. 3. Possible DNN Outputs and Feasible Region

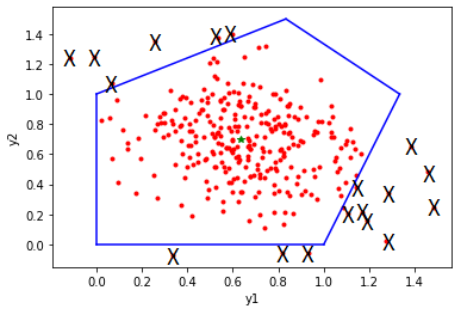

Fig. 4. Nonfeasible solutions are removed (marked as X).

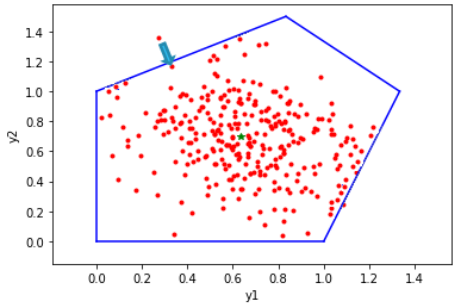

Fig. 5. Nonfeasible solutions are snapped to the nearest feasible solutions.

Let the DNN output be $\hat{y}$. The filtered output y is defined as follows:

$$y = \begin{cases} \hat{y} & if\ P(\hat{y}) \\ \perp & Otherwise \end{cases}$$

Here $\perp$ represents that no output is generated. This strategy works when there is a back-up mechanism that generates an alternative solution (e.g., there is a default position where the drone has to go, or a human operator supplies the destination in case the DNN fails to do so).

If no such back-up mechanism is available, we may be able to use the nearest point in the feasible region as the output.

$$y = \begin{cases} \hat{y} & if\ P(\hat{y}) \\ g(\hat{y}) & Otherwise \end{cases}$$

Here, $g(\hat{y})$ is a function that returns the nearest feasible solution. Fig. 5 shows a nonfeasible solution that is to be moved into the feasible region.

Filtering is simple and effective, but it fails to reflect the fact that the DNN proposed a non-feasible solution. The DNN should have inferred some patterns (i.e., a mathematical model) during the training, and there must be reasons why the DNN generated such outputs even though they are undesirable or unsafe. For example, the drone may travel faster if the given reference point is further away. A human operator may give such an out-of-bound reference point with an intention to move the reference point back to the feasible region in the subsequent control inputs. For a safety-critical, fully-autonomous system, however, it may be required that all the reference points are within the feasible region.

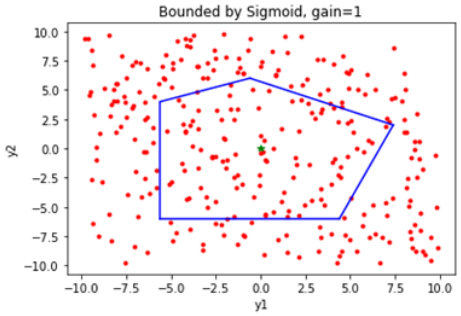

Fig. 6. Outputs are bounded by a Sigmoid function

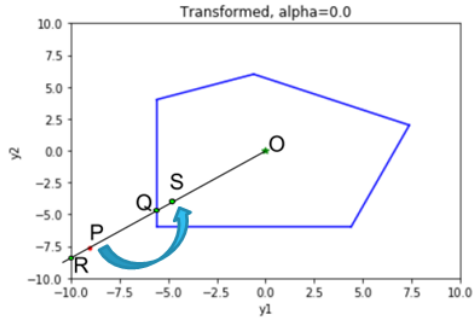

Fig. 7. Move every point along with the half line originated from the pivot.

III. Transformation to Feasible Space

We propose a transformation from the R^n space to the feasible region. This transformation is done in the following two steps. First, the unbounded R^n space is transformed into the n-dimensional hypercube $(0,1)^n$. One way to do this is to use the Sigmoid function $\varsigma(x)$.

$$\varsigma(x) = \frac{1}{1+e^{-x}}$$

The results of this bounding transformation are shown in Fig. 6.

Next, we select one interior point in the feasible region. We call it the *pivot*. Without loss of generality, we can assume that the pivot coincides with the origin of the hypercube.

For every bounded point $P = \varsigma(\hat{y})$, we draw a half line from the origin O to P, and let the intersections of OP with the feasible region boundary and the hypercube be Q and R, respectively (see Fig. 7). We move P to another point S on this half line so that $|OP| : |OR| = |OS| : |OQ|$.

The results of this two-step transformation is shown in 8. With this transformation, every transformed point is in the interior of the feasible region for any combination of the input point, the training data set, and the hyperparameters.

Note that our transformation is continuous and almost everywhere differentiable. Thus, we can supply teacher signal in the transformed space and back-propagate the error through this transformation.

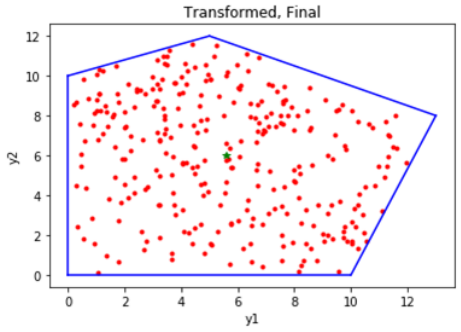

Fig. 8. Transformed outputs

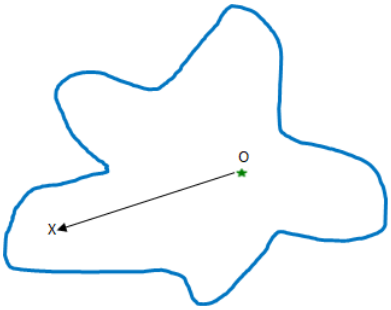

Fig. 9. Star-Shaped Region.

The discussion so far assumed that the feasible region is a convex set. Actually, the same discussion holds when the feasible region is star-shaped as in Fig. 9. A star-shaped region has at least one interior point O such that for any interior point X, the line segment OX is included in the region. In this case, we use O as the pivot in our transformation.

IV. Related Works

The general AI safety issues have been discussed especially in the context of reinforcement learning for controlling real-world machines (e.g., robots and drones). There are many sources of unsafe behaviours such as having a wrong reward function and exploring unsafe regions during the learning process, and they are extensively discussed in [1].

Some approaches such as [2] express the constraints within the reward function (e.g., giving exponentially large penalties when the control input gets closer to the safety boundary). Other approaches include searching the worst-case inputs (formulated as an optimization problem) for a given DNN [3] and estimating possible behaviour of the current DNN using a Bayesian validation method [4].

These approaches are to give probabilistic guarantees to the safety, which may not be acceptable for highly critical systems. Also they assume that the pretrained model is given, which requires that if the training data set or the hyper parameters are changed the varification results cannot be used. Our method guarantees that no infeasible solutions are produced no matter how the training is done. In addition, our method is very simple to implement, which is a very important factor for safty-critical systems.

V. Conclusion

Statistical machine learning can be seen as a new paradigm in programming – instead of building a software system by the traditional top-down approach (i.e., starting from the requirement and gradually breaking down it to smaller components), machine learning enables a bottom-up, inductive approach to build a software system. Just as we studied the knowledge on how traditional software development can be done in a safe and effective way and organized the knowledge as *Software Engineering*, we believe that now is the time to start a new engineering discipline *Machine Learning Systems Engineering (MLSE)* so that we organize the knowledge on how ML-based systems are safely and effectively developed and maintained [5]. The work reported here will be one of the first attempts of the upcoming MLSE studies.

References

[1] D. Amodei, C. Olah, J. Steinhardt, P. Christiano, J. Schulman, and D. Mané, "Concrete problems in ai safety," *arXiv preprint arXiv:1606.06565*, 2016.

[2] P. Geibel and F. Wysotzki, "Risk-sensitive reinforcement learning applied to control under constraints," *Journal of Artificial Intelligence Research*, vol. 24, pp. 81–108, 2005.

[3] K. Dvijotham, R. Stanforth, S. Gowal, T. Mann, and P. Kohli, "A dual approach to scalable verification of deep networks," *arXiv preprint arXiv:1803.06567*, 2018.

[4] J. F. Fisac, A. K. Akametalu, M. N. Zeilinger, S. Kaynama, J. Gillula, and C. J. Tomlin, "A general safety framework for learning-based control in uncertain robotic systems," *arXiv preprint arXiv:1705.01292*, 2017.

[5] H. Maruyama and T. Kido, "Machine learning engineering and reuse of ai work products," in *The First International Workshop on Sharing and Reuse of AI Work Products*, 2017.