CS639A: Project Report

December 6, 2020

Group members: Nikhil Kumar Singh (19211265), Shatroopa Saxena (19211270), Priyanka Talwar(19211268)

1 Aim

Implementation of Star based abstractions [6] for the Outside-the-Box Framework and comparison vis-a-via Box Abstraction.

2 Introduction

2.1 Framework explained: Outside the Box

Outside the box tool uses a monitor over a layer of neural network. As shown in fig:1, the monitor builds box abstractions to watch layer's output so that it can tell if an input is novel. In case the point lies outside all boxes, the framework declares it as an unknown input.

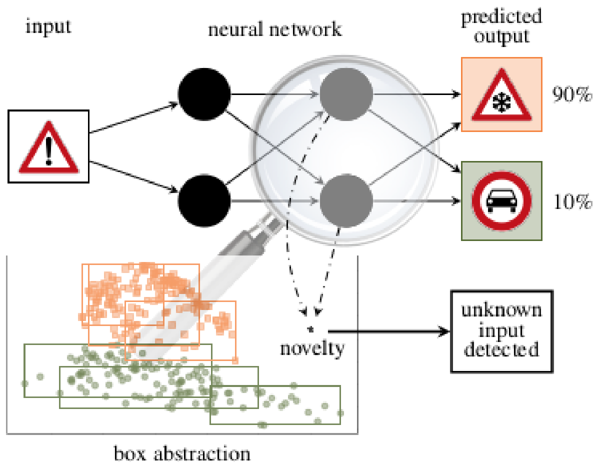

Figure 1: Monitoring framework and box abstraction by out of the box tool

2.2 Star Abstraction

A Star [6] is modeled as a 3-tuple $\langle c, V, P\rangle$ where: $c \in \mathbb{R}^n$ denotes the center, $V = \{v_1, \cdots, v_m\}$ is the set of m basis vectors $\in \mathbb{R}^m$ and $P : \mathbb{R}^m \longrightarrow \{\top, \perp\}$ is a predicate. Any bounded convex polyhedron can be represented as a star.
We referred to [1] for our implementation.

3 Implementation

3.1 Class Star

This class represents a Star. The *Star* class inherits *PointCollection* class. This means that a star object takes a collection of points as input and updates the constraints accordingly. Using the Proposition 1 in [6], we can represent any bounded convex polyhedron $P = \{x|Cx \leq d, x \in \mathbb{R}^n\}$ as a star. So, in order to generate a star we first create a polyhedron using the given set of vertices. As we know, a polyhedron can be represented in 2 forms [4]:

- **H-representation**
  In this representation, the polyhedron is represented as set of constraints $Ax \leq b$.

- **V-representation**
  In this representation, the polyhedron is represented as a set of vertices $V = \{v_1, ..., v_n\}$.

Once we create a polyhedron using the given vertices, we convert it to H-rep (get matrix A & b). Then, using these constraints as predicates and an identity matrix as basis matrix, we generate a star.

To build a star, we follow an incremental approach -

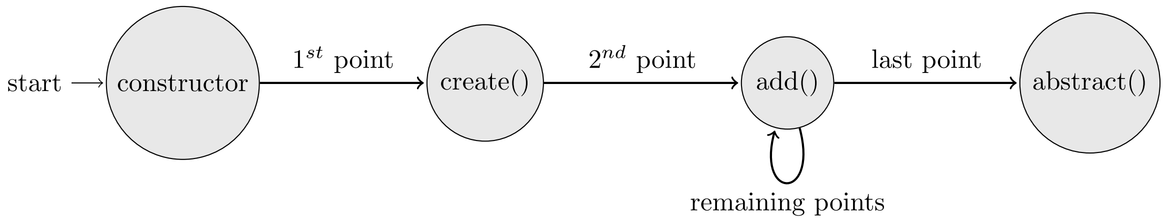

Figure 2: Flowchart for incrementally creating a star out of a set of points

As shown in fig: 2, Building a star requires 3 methods as explained below:

- *Constructor:* This method creates an empty star with no constraints.
- We add each point incrementally and modify the constraints representing the star:
  - For the 1^{st} point, we call the *create()* method. This method defines the initial constraints on the star.
  - Thereafter, for adding any number of points to the star, we call the *add()* method. This function accepts one point at a time and incrementally updates the constraints.

Other methods of this class are:

- *plot():* This method is used to plot a star.
- *isempty():* This method checks if a star is empty or not.
- *issubset():* This method checks if a star is subset of another star or not.
- *mergeStars():* This method is used to merge 2 stars if need be.
- *contains():* This method checks if a point is contained in a star or not.

3.2 Class : Star Abstraction

This class basically represents a set of stars so we may refer to this as a Set based collection. As we already know, Outside the Box code uses kMeans clustering on labelled data. As many clusters; those many stars. Apart from the constructor, a key method of this class is *initialize()* which initializes an empty abstraction based on the number of neurons (of Neural Network) being watched. This also denotes the dimension of each star in the abstraction.

4 Tasks accomplished

- Implemented the class for Star abstraction and also the required methods in python.
- Integration of the Star based abstraction into the Outside-the-box framework.
- Implementation of a function *toStar* for the Box objects that can convert a box object to a star object.

5 Results

Here we provide the comparison vis-a-vis Box abstraction.

5.1 Toy Model

We show the plot of Star abstraction for the Toy Model in Figure 3. The corresponding plot box abstraction is shown in Figure 4.

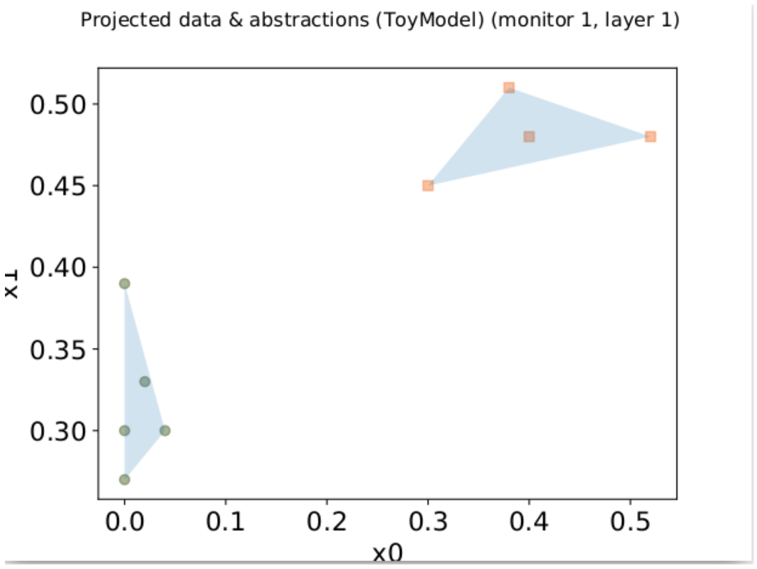

Figure 3: Star abstraction for Toy Model with basis as identity matrix [1 0; 0 1]

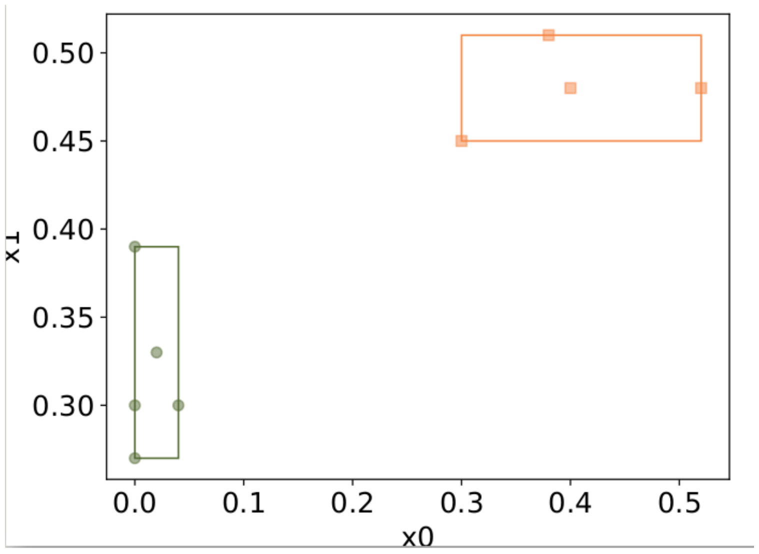

Figure 4: Box abstraction for Toy Model

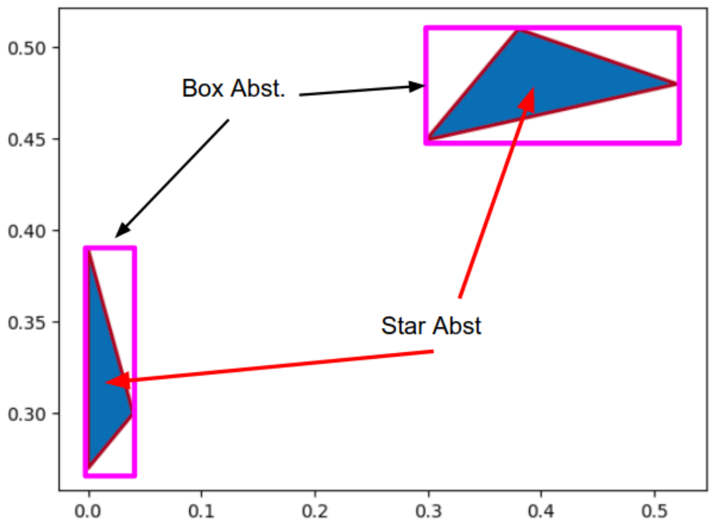

Figure 5: Box v/s Star abstraction for Toy Model

On comparing the Box abstraction with Star abstraction, we observe that Star abstraction is much more precise. Even on this simple example we see 50-60% smaller size of abstraction (Figure 5).

Some other possible star abstractions generated with different basis matrix is shown in Figure 6.

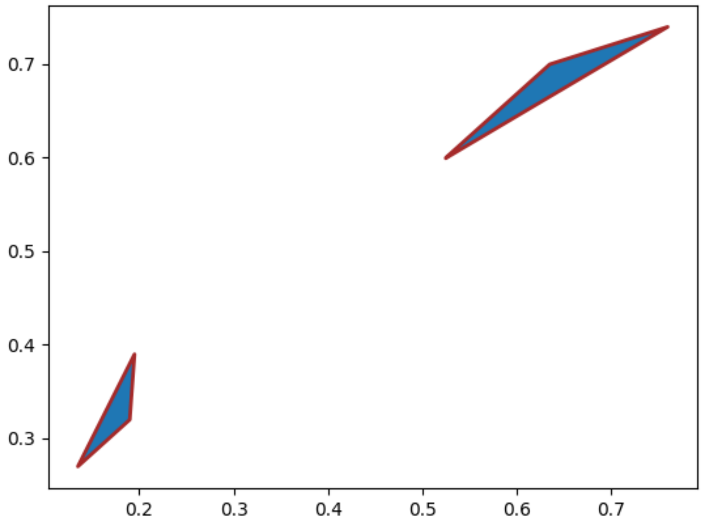

Figure 6: Star abstraction for Toy Model with basis as [1 0.5; 0.5 1]

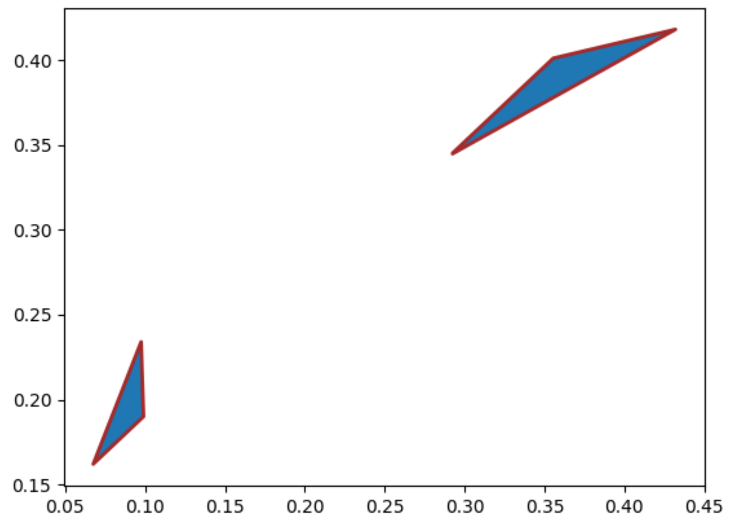

Figure 7: Star abstraction for Toy Model with basis as [0.6 0.25; 0.25 0.6]

5.2 MNIST Dataset

In case of MNIST dataset, each point belongs to a 40 dimensional space. This leads to a too much computation time to generate constraints for the Star. Hence, we show only plots for two-dimensional points (dim 3 and 13) in Figure 8 and four-dimensional points (dim 3, 9, 13 and 17) in Figure 10 for the Star Abstraction. In both these case we only plot dim-3 v/s dim-13. In Figure 9, we show Box abstraction on MNIST dataset (considering all 40 dimensions and plotting dim-3 v/s dim-13).

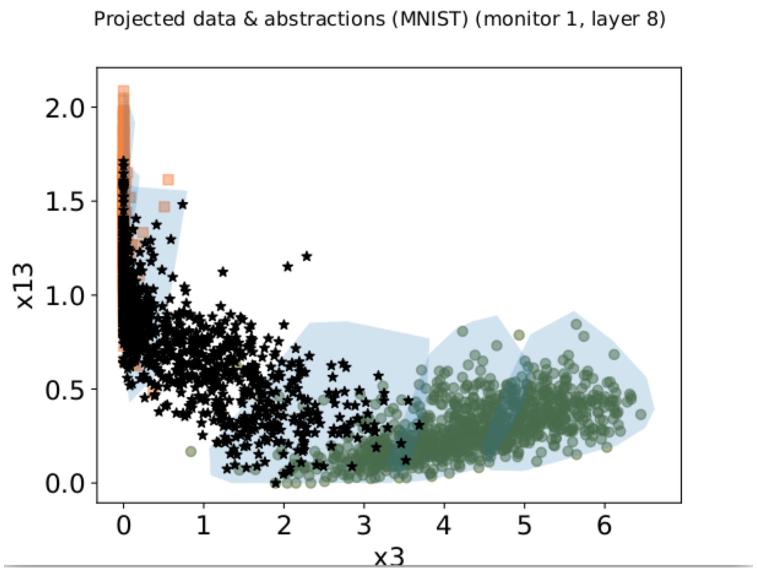

Figure 8: Star abstraction when considering only dim-3 and dim-13 for MNIST dataset

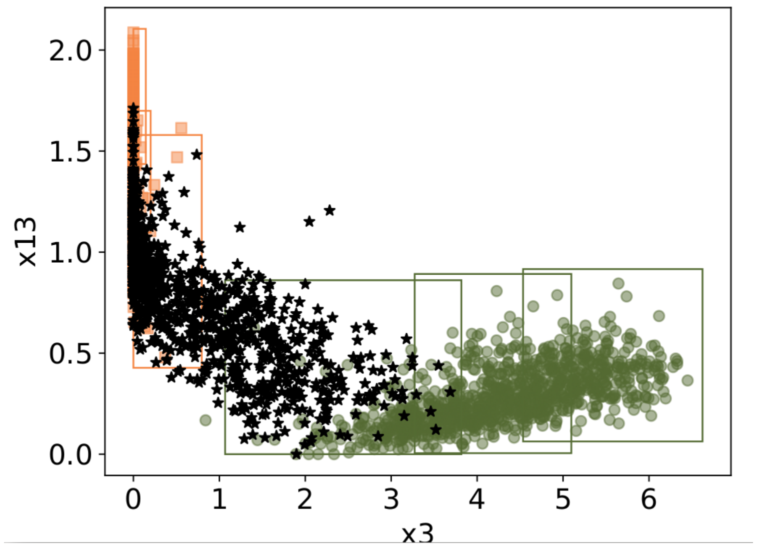

Figure 9: Box abstraction for MNIST dataset considering only dim-3 and dim-13

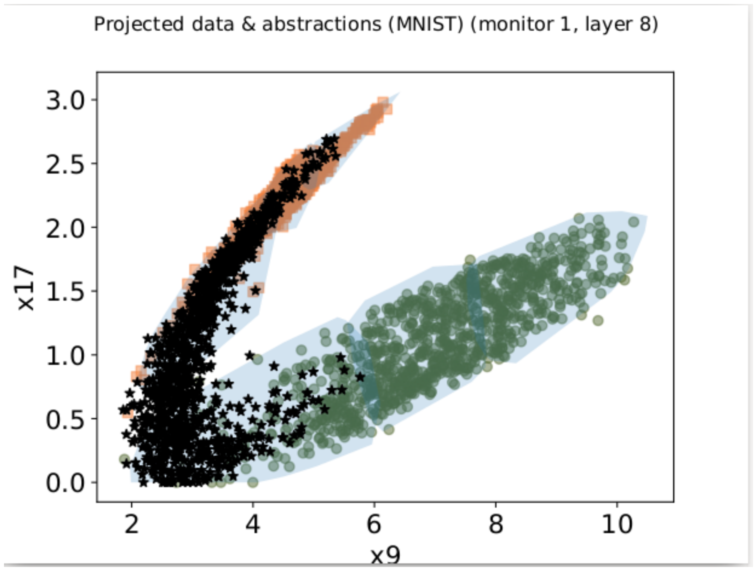

Figure 10: Star abstraction when considering only dim-9 and dim-17 for MNIST dataset

Legend: Green & Orange are known classes; Black represents the unknown input.

6 Tools used

- Outside the Box [2] Code (implemented in python)
- NNV Tool [5] (Implemented in matlab)
- pypoman, parma polyhedra library [3] libraries.

7 Conclusion

We observed the following differences when comparing box and star abstractions:

- Star abstractions offer more precision (Worst case going upto the convex hull which is still better than the box).
- Star abstractions are very expensive to compute. The time increases further with an increase in dimension.

So, it's basically a precision vs cost tradeoff.

References

[1] Linda Deneen and Gary Shute. "Polygonizations of Point Sets in the Plane". In: *Discrete Comput. Geom.* 3.1 (Dec. 1988), pp. 77–87. ISSN: 0179-5376.

[2] Thomas A. Henzinger, Anna Lukina, and Christian Schilling. "Outside the Box: Abstraction-Based Monitoring of Neural Networks". In: *ECAI 2020 - 24th European Conference on Artificial Intelligence, 29 August-8 September 2020, Santiago de Compostela, Spain, August 29 - September 8, 2020 - Including 10th Conference on Prestigious Applications of Artificial Intelligence (PAIS 2020).* Ed. by Giuseppe De Giacomo et al. Vol. 325. Frontiers in Artificial Intelligence and Applications. IOS Press, 2020, pp. 2433–2440. DOI: `10.3233/FAIA200375`. URL: `https://doi.org/10.3233/FAIA200375`.

[3] *Parma Polyhedra Library.* `https://www.labri.fr/perso/vdelecro/pplpy`. [Online].

[4] *Polyhedra and Polytopes.* `https://scaron.info/teaching/polyhedra-and-polytopes.html`. [Online].

[5] Hoang-Dung Tran et al. *NNV: The Neural Network Verification Tool for Deep Neural Networks and Learning-Enabled Cyber-Physical Systems.* 2020. arXiv: `2004.05519 [eess.SY]`.

[6] Hoang-Dung Tran et al. "Star-Based Reachability Analysis of Deep Neural Networks". In: *Formal Methods – The Next 30 Years.* Ed. by Maurice H. ter Beek, Annabelle McIver, and José N. Oliveira. Cham: Springer International Publishing, 2019, pp. 670–686. ISBN: 978-3-030-30942-8.