CS521 Course Project Presentation

Monotonic Neural Networks

Group:

Shaurya Gomber (sgomber2)
Karan Aggarwal (karana5)

Introduction & Motivation

- What is monotonicity in Neural Networks?
  - Requiring that **function's output increases with increasing values of specific input features** (reverse case for function output decreasing can be handled similarly)

- Why and where is monotonicity required?
  - DNNs are increasingly being used to make sensitive decisions.
  - It is of utmost importance for **safety, ethical, and legal reasons** that some decisions made are monotonic. Some examples include:
    - Increase in salary gives increased predicted loan amount
    - Increase in number of rooms gives increased predicted house price
    - Decrease in crime rate of area gives increased predicted house price
    - Increase in hours per week gives increased predicted income level

Related Works

1. Monotonicity by construction: Guaranteed monotonicity, but restrict the hypothesis class
   - Min-Max Networks[5]
   - Deep Lattice Networks[6]
   - Learning networks with positive coefficients[7]

2. Monotonicity by Learning Process: These methods do not guarantee complete monotonicity but aim to be flexible and more scalable:
   - Training the networks with heuristic monotonicity regularizations[3]
   - Extending Training to include constraints[4]
   - Including a “risk” function that needs to be minimized along with the loss function[8]

Approach

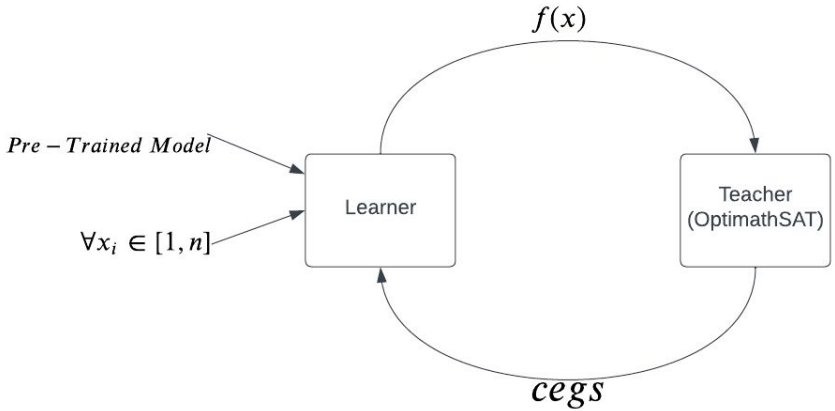
We implemented, extended and evaluated two approaches that incorporate monotonicity by modifying the training process

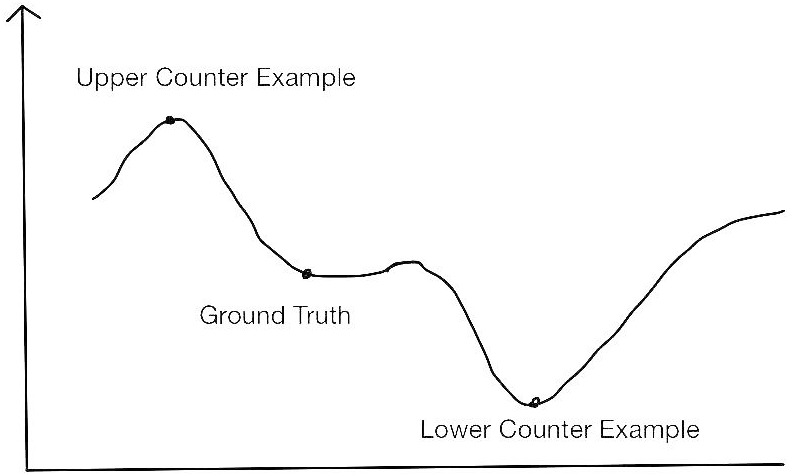
1. **Counter example guided learning**: Enriching the training data by finding examples that violate the monotonicity constraints and adding those to the training set.

2. **Loss function modification:** Modifying the loss function to penalize the model if it violates given constraints for monotonicity.

Counter Example Guided Learning

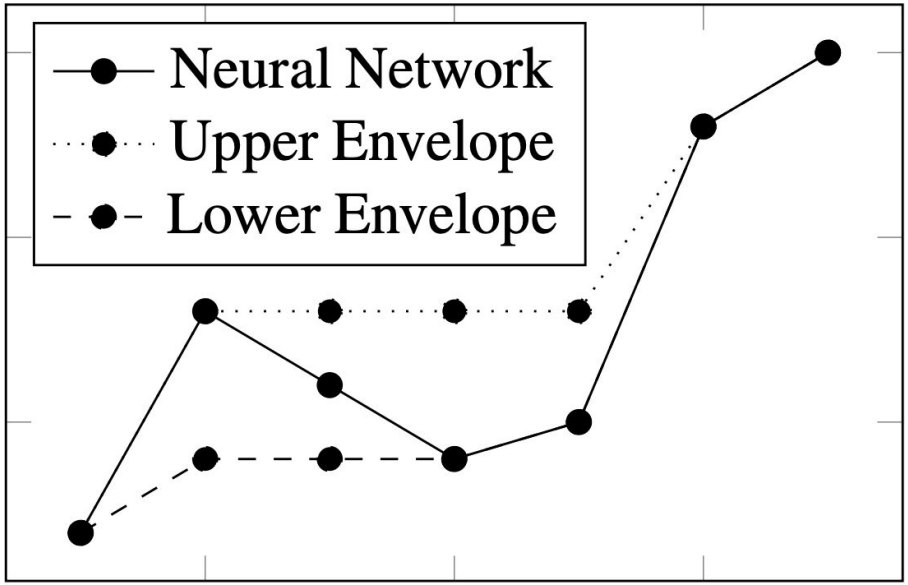
We implemented COMET[3] which is an approach to extend the training data by finding examples in the hypothesis space that violate monotonicity and training the model again with those examples.





Enveloping

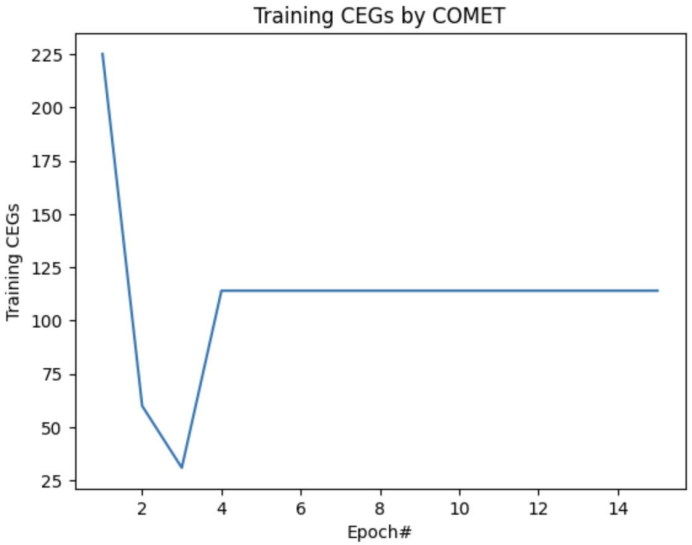
- COMET tries to enforce monotonicity as an inductive bias.
- Though this can be good for model to learn better (monotonicity acts as a good regularizer), it is not sufficient to guarantee monotonicity.
- Thus, it employs enveloping for the CGL based model to guarantee monotonicity.



COMET Results

- We experimented with the Boston Housing Group Dataset setting the monotonicity constraint on the number of rooms in a dwelling.

- COMET did improve the training loss but did not enforce monotonicity efficiently.

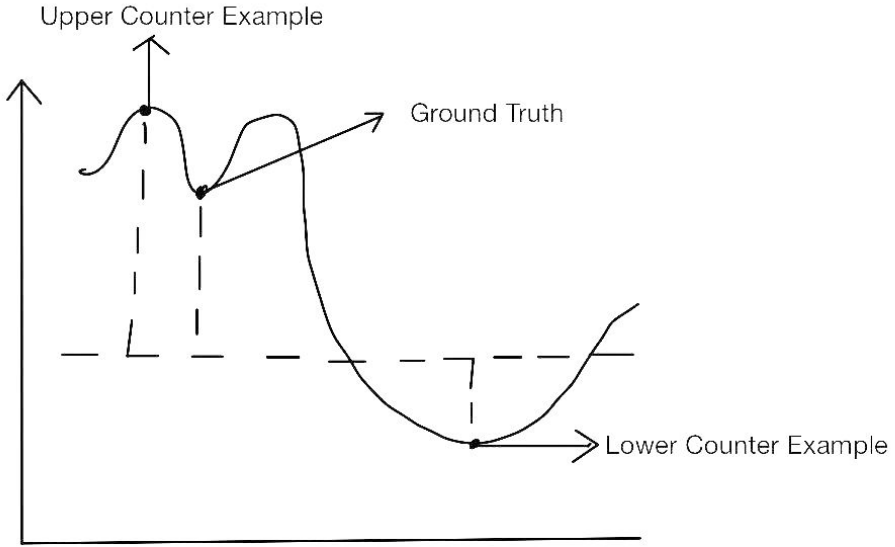


	Original	CGL	Original Envelope	COMET Envelope
Train	53.81	43.66	54.3	43.64
Test	53.54	46.28	50.07	43.23

MSE Loss Comparison with different combinations

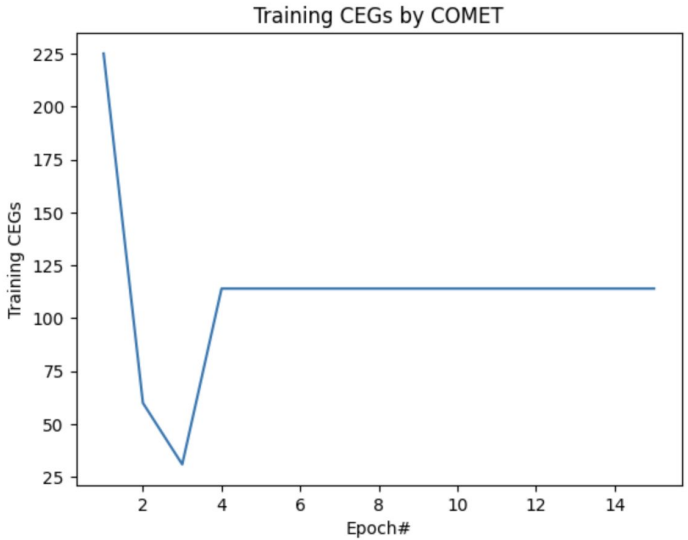
Can we do better?

- To put the counter examples in training set, we need to label them. For this, COMET takes the average of the predicted values of UEG, GT and LEG.
- Modifying the counter examples using an educated guess from the ground truths might do better.
- Using the training data, we get an approximate value of how changing the monotonic feature changes the target value. This value is then used to predict labels for the new training points.



Maybe not?

- We used the Boston Housing Group Dataset again with the same monotonicity constraint on the number of rooms in a dwelling.
- Again, the loss values are improved, but we are not doing great on monotonicity front.
- Adding counter examples to training data seems too weak to enforce monotonicity.



	Original	CGL	Original Envelope	COMET Envelope
Train	53.81	41.56	54.3	39.68
Test	53.54	44.74	50.07	42.13

MSE Loss Comparison with different combinations

Monotonicity Constraint in Loss Function

- Along with penalizing the model to match the labels, we also penalize it to achieve monotonicity
- This is done by changing the loss function to include a measure of non-monotonicity.
- Currently, a typical neural network operates on a loss function similar to the following:

$$\min \mathcal{L}_{mono} = \min \left\{ \quad + \mathcal{L}_{NN} \right\} \tag{1}$$

Monotonicity Constraint in Loss Function

- Along with penalizing the model to match the labels, we also penalize it to achieve monotonicity
- This is done by changing the loss function to include a measure of non-monotonicity.
- As suggested in Gupta et. al. [4], this measure of non-monotonicity can be approximated by the following point-wise loss function:

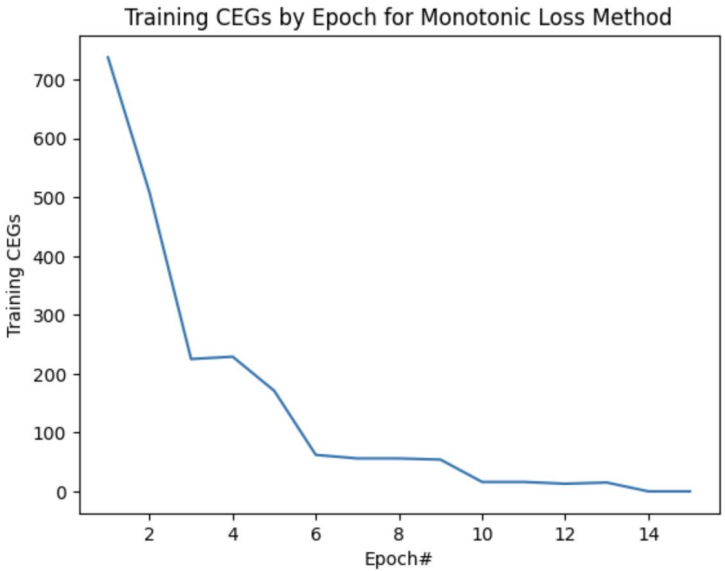
$$\min \mathcal{L}_{mono} = \min \left\{ \sum_{i=1}^{n} \max \left(0, -\nabla \cdot_M f(x_i; \theta) \right) + \mathcal{L}_{NN} \right\} \tag{1}$$

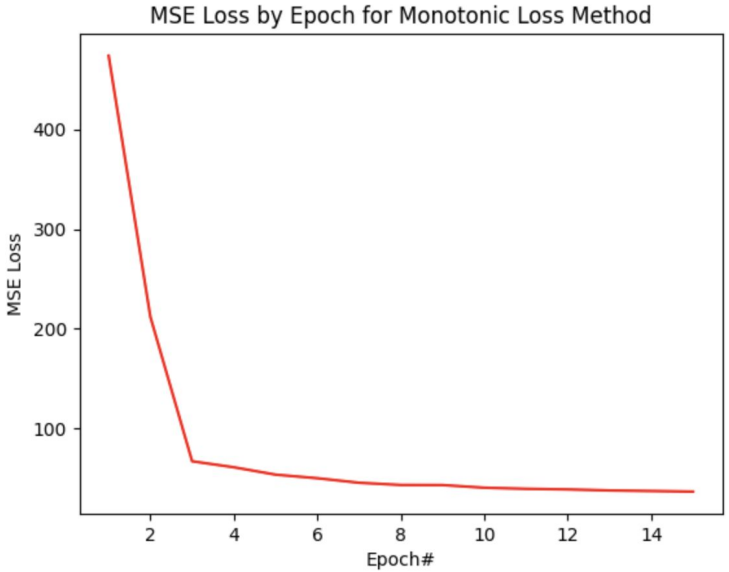
where $\nabla \cdot_M$ is divergence with respect to feature set $x[M]$, i.e., $\sum_j \frac{\partial f(x_i;\theta)}{\partial x[M]_i^j} \quad \forall \quad j \in M$, θ are the trainable parameters, and $\mathcal{L}_{NN}$ refers to the empirical risk minimization for neural networks.

- The monotonic loss in the method basically tries to make the gradient with respect to the monotonic features positive by adding a loss value for negative gradients.

Results for monotonicity loss

- We implemented this approach on the Boston House Pricing dataset with a modification: we scaled the penalization for monotonicity since the loss from training data is considerably larger when compared to the monotonic penalty.
- As can be seen, this method is more efficient in enforcing monotonicity while still maintaining accuracy.





Summary

1. While modern neural networks do a good job at ensuring accuracy, they may not always satisfy monotonic constraints which are sometimes hard requirements for a model to be used in a practical settings.

2. Some approaches try to guarantee monotonicity by construction, while others are more flexible in that they try to enforce monotonicity in any network (but do not guarantee it).

3. COMET, inspired from the Learner-Teacher framework, tries to enforce monotonicity by enriching the training data with counter examples and thus induce monotonicity as an inductive bias.

4. Though monotonicity as an inductive bias is seen to be a good regularizer and works well for accuracy, it is not efficient for ensuring monotonicity.

5. The results from implementing the monotonic loss method (from Gupta et. al. [4]) imply that changing the loss function to include a measure of non-monotonicity is a much efficient way to get monotonicity than augmenting the training data set.

Issues / Future Work

1. Currently, COMET uses OptimathSAT to generate counter example, this makes it difficult to scale up to a larger dataset or a larger network. Also, COMET currently only works well with a ReLU activated network or other activations that can be encoded efficiently in a SMT solver.

2. Both of the approaches mentioned above currently deal with monotonic features with the same monotonic direction, this can handle some requirements may be not all for example: Increasing crime rate leads to a reduction of price and increasing number of rooms leads to an increase in price

3. Both these approaches can be unified to create a common framework for training of neural networks which could potentially increase the efficiency of the neural network in terms of monotonicity.

Bibliography

1. Joseph Sill and Yaser S Abu-Mostafa. Monotonicity hints. In Advances in neural information processing systems, pages 634–640, 1997.
2. Mahdi Milani Fard, Kevin Canini, Andrew Cotter, Jan Pfeifer, and Maya Gupta. Fast and flexible monotonic functions with ensembles of lattices. In Advances in Neural Information Processing Systems, pages 2919–2927, 2016.
3. Aishwarya Sivaraman, Golnoosh Farnadi, Todd Millstein, Guy Van den Broeck: Counterexample-Guided Learning of Monotonic Neural Networks
4. Akhil Gupta, Naman Shukla, Lavanya Marla, Arinbjörn Kolbeinsson, Kartik Yellepeddi: How to Incorporate Monotonicity in Deep Networks While Preserving Flexibility?
5. Ye Luo, Shiqing Fan: Min-Max-Plus Neural Networks
6. Seungil You, David Ding, Kevin Canini, Jan Pfeifer, Maya Gupta: Deep Lattice Networks and Partial Monotonic Functions
7. N.P. Archer; S. Wang: Learning bias in neural networks and an approach to controlling its effect in monotonic classification
8. Marco Loog, Tom Viering: Minimizers of the Empirical Risk and Risk Monotonicity

THANK YOU