**Project Progress Report** **Course:** IFT6269-A2024
**December 2, 2024** **Professor:** Simon Lacoste-Julien

**Group no. 11: David Wardan, Khalil Sabri, Miquel Florensa**

## Implementing the Toy Example

So far, we have implemented the Tractable Approximate Gaussian Inference (TAGI) method. We have accomplished the following:

- **Manual Formulation of Forward and Inference Steps:** We formulated all the forward and inference steps required for TAGI shown in Figure 1. This process included deriving covariances that were not explicitly covered in the original paper.

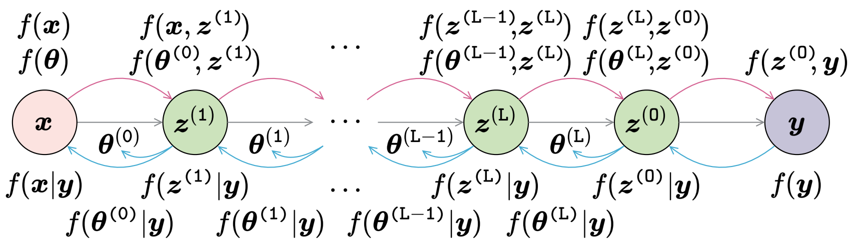
$f(\boldsymbol{x})$
$f(\boldsymbol{\theta})$
$f(\boldsymbol{x}, \boldsymbol{z}^{(1)})$
$f(\boldsymbol{\theta}^{(0)}, \boldsymbol{z}^{(1)})$
$f(\boldsymbol{z}^{(L-1)}, \boldsymbol{z}^{(L)})$
$f(\boldsymbol{\theta}^{(L-1)}, \boldsymbol{z}^{(L)})$
$f(\boldsymbol{z}^{(L)}, \boldsymbol{z}^{(0)})$
$f(\boldsymbol{\theta}^{(L)}, \boldsymbol{z}^{(0)})$
$f(\boldsymbol{z}^{(0)}, \boldsymbol{y})$
$\boldsymbol{x}$
$\boldsymbol{z}^{(1)}$
$\boldsymbol{z}^{(L)}$
$\boldsymbol{z}^{(0)}$
$\boldsymbol{y}$
$\boldsymbol{\theta}^{(0)}$
$\boldsymbol{\theta}^{(1)}$
$\boldsymbol{\theta}^{(L-1)}$
$\boldsymbol{\theta}^{(L)}$
$f(\boldsymbol{x}|\boldsymbol{y})$
$f(\boldsymbol{z}^{(1)}|\boldsymbol{y})$
$f(\boldsymbol{z}^{(L)}|\boldsymbol{y})$
$f(\boldsymbol{z}^{(0)}|\boldsymbol{y})$
$f(\boldsymbol{y})$
$f(\boldsymbol{\theta}^{(0)}|\boldsymbol{y})$
$f(\boldsymbol{\theta}^{(1)}|\boldsymbol{y})$
$f(\boldsymbol{\theta}^{(L-1)}|\boldsymbol{y})$
$f(\boldsymbol{\theta}^{(L)}|\boldsymbol{y})$

Figure 1: Graphic illustration of TAGI's forward and inference steps for a neural network architecture.

- **Development of a NumPy-Based Application:** We developed a NumPy-based application to replicate the regression toy example from the paper shown in 2. This implementation serves as a verification step to make sure that the formulation is identical with what is presented in the paper.

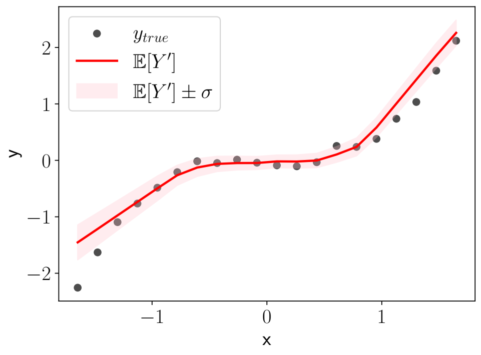

Figure 2: Obtained results on the replicated toy example.

## Implementing TAGI for Classification

We are currently working on implementing a probabilistic softmax activation function for classification tasks. However, as we will detail in the final report, the probabilistic softmax has shown poor performance in our experiments. To address this, we plan to implement an alternative activation function, called *remax*, which offers better performance. In the remax function, the standard softmax operation $a_i = \frac{\exp(z_i)}{\sum \exp(\mathbf{z})}$ is replaced with $a_i = \frac{\text{mrelu}(z_i)}{\sum \text{mrelu}(\mathbf{z})}$ where *mReLU* is a Gaussian-mixture ReLU. This implementation will be integrated into the `cuTAGI` [?] library to ensure optimal computational efficiency.

## Comparing TAGI with Backpropagation

We will compare `pyTAGI` and `PyTorch` on a regression task (Boston Housing) and a classification task (MNIST) using the following setups:

- **Architectures:** FNN, CNN and ResNet.
- **Configurations:** Different numbers of layers (1, 3, 5), neurons per layer (16 - 128), batch sizes (1, 8, 16, 256), batch and layer normalization.
- **pyTorch-Specific:** Varying learning rates values (e.g., 1e-3, 1e-2, 0.1).
- **pyTAGI-Specific:** Varying $\sigma_v$ values (e.g., 0.01, 0.05, 0.1, 0.2, 0.5, 1.0).
- **Metrics:** Accuracy, convergence speed, and training efficiency.

The goal is to collect results under different setups to find optimal configurations and determine scenarios where pyTAGI outperforms PyTorch. This is in order to determine the advantages of PyTAGI compared to classical NNs.