Guidelines to implement the code

Md Mahmudul Hasan

November 29, 2021

1 Background

The paper [1] proposes a way to variationally estimate mutual information between two random variables. Since for given joint random variables X and Z, the mutual information can be defined as the KL divergence between their joint distribution and the product of their marginal distribution, the paper used the expression given by (1) as their cost function to estimate the mutual information $I(X;Z)$.

$$\mathcal{L}(\{(x_i,z_i)\}_{\mathcal{B}},\theta)=\frac{1}{|\mathcal{B}|}\sum_{i=1}^{|\mathcal{B}|}T_\theta(x_i,z_i)_{\mathcal{B}}-\log\left[\frac{1}{|\mathcal{B}|}\sum_{i=1}^{|\mathcal{B}|}e^{T_\theta(\bar{x}_i,\bar{z}_i)_{\mathcal{B}}}\right] \tag{1}$$

where

- $\mathcal{L}(.)$ is the objective function that is maximized.
- $|\mathcal{B}|$ is the cardinality of batch $\mathcal{B}$.
- T is the function approximated by the neural network.
- θ is the set of parameters of the neural network.

Though the paper [1] maximizes the objective function given by (1), we for the convenience of coding minimize the $-\mathcal{L}(.)$ and the end find the negative of the optimized value to get the estimated mutual information.

2 Preparation of the input set

In our case X is 10 dimensional and Z is one dimensional arbitrary joint distributed random variables. We generated the 11 dimensional 'joint data' from the joint distribution (X,Z) i.e. $(x_1,\ldots,x_{10},z)$. We generated the 'marginal data' by juxtaposing the marginal samples $\bar{x}$ and $\bar{z}$ of X and Z respectively i.e. $(\bar{x}_1,\ldots,\bar{x}_{10},\bar{z})$. Finally we prepared our 22 dimensional input for our model by placing the marginal data next to the joint data i.e. $(x_1,\ldots,x_{10},z,\bar{x}_1,\ldots,\bar{x}_{10},\bar{z})$.

Since GitHub does not allow me to upload larger files, I have uploaded a small dataset of only 20 data points for your reference. You can generate and use your own data to implement the code.

3 How the code works

the 'mi-main' file has all the hyper-parameters with default values that you can change as required. 'mi-layers' file has the structure of the neural network laid down. 'mi-model' file defines the model input output relationships, calculates the cost function and calls in the structure form 'mi-layers' to do so.

The 'mi-main' file has the 'training step' defined in it that requires the the calculation of cost function defined in 'mi-model'. 'mi-main' has the training process ordered in terms of the number of epochs and iterations. 'mi-main' calls in the 'mi-models' to calculate and update the gradients of the cost function with respect to the model parameters in every iteration. The program stops after it runs for the default number of 'epochs' set at the parameter list. The file 'mi-data-processing' is called in by the main file 'mi-main' to get the dataset shuffled and batched according to the batch size before every epoch.

References

[1] M. I. Belghazi, A. Baratin, S. Rajeshwar, S. Ozair, Y. Bengio, A. Courville, and D. Hjelm, "Mutual information neural estimation," in *International Conference on Machine Learning*. PMLR, 2018, pp. 531–540.