Mutual Information Neural Estimator

During one of my research in machine learning, I came across a neural network [1] that can model mutual information, and it can be used in ML to train the output of models to be more dependent or independent of each other. So, I took a deeper look to understand how it is done.

Given 2 random variables *X* and *Z*, mutual information is defined as

$$I(X;Z) = \int_{X\times Z} log\frac{d\mathbb{P}_{XZ}}{d\mathbb{P}_X \otimes \mathbb{P}_Z} d\mathbb{P}_{XZ}$$

which is essentially the KL-divergence of $\mathbb{P}_{XZ}$ and $\mathbb{P}_X \otimes \mathbb{P}_Z$. As we know independence is proofed by $P(x|z) = P(x) \Rightarrow P(x|z) \times P(z) = P(x) \times P(z) \Rightarrow \mathbb{P}_{XZ} = \mathbb{P}_X \otimes \mathbb{P}_Z$, the more similar the $cross\ entropy(\mathbb{P}_{XZ}|\mathbb{P}_X \otimes \mathbb{P}_Z)$ and $entropy(\mathbb{P}_{XZ})$, the smaller the mutual information and hence the more independent are the two random variables.

To find the best neural network *T* to approximate MI, we rely on the theory:

$$D_{KL}(\mathbb{P}||\mathbb{Q}) = sup_{T:\Omega\to\mathbb{R}} \mathbb{E}_{\mathbb{P}}[T] - \log\left(\mathbb{E}_{\mathbb{Q}}[e^T]\right)$$

The proof can be found in the appendix of [1], and it is mainly based on the fact that T is used to model a Gibbs distribution $\mathcal{g}$ defined by $d\mathcal{g} = \frac{1}{Z} e^T d\mathbb{Q}$, where $Z = \mathbb{E}_{\mathbb{Q}}[e^T]$.

So, we just train *T* to maximize the above equation to get a good approximation of MI by plugging in $\mathbb{P}_{XZ}$ as $\mathbb{P}$ and $\mathbb{P}_X \otimes \mathbb{P}_Z$ as $\mathbb{Q}$.

To capture the non-linear statistical dependencies between *X* and *Z*, we need to use non-linear functions as the activation functions in T. Multi-layer perceptrons (artificial neurons) are used instead of a single layer one to deal with the possible high complexity dependency problem. Therefore, a possible structure of the neural network *T* is as follows:

$$Merge(Dense(x, w_1), Dense(z, w_2)) \to relu(i) \to Dense(i, w_3)$$

where i refers to the previous input vector and w_i is the weights to be trained. $Merge$ is a layer to concatenate vectors, $relu$ is an activation layer doing bitwise operation

$y = \max(x, 0)$ and $Dense(a, b)$ is doing matrix multiplication $\begin{pmatrix} b_{1,1} & \cdots & b_{1,n} \\ \vdots & \ddots & \vdots \\ b_{n,1} & \cdots & b_{n,n} \end{pmatrix} * \begin{pmatrix} a_1 \\ \vdots \\ a_n \end{pmatrix}$

In practice, we only have samples of *X* and *Z* without knowing what $\mathbb{P}_{XZ}$ and $\mathbb{P}_X \otimes \mathbb{P}_Z$ are. So, we use *x,z* in *(X,Z)* and *x,z'* in *X,shuffle(Z)* as inputs to T to approximate the joint distribution and marginal distribution respectively.

References

1. Mohamed Ishmael Belghazi, A. B. (2018, Jun 7). *Mutual Information Neural Estimation.* Retrieved from arxiv: https://arxiv.org/pdf/1801.04062.pdf