PROJECT PROPOSAL :
Optimization using Neural Networks

Original formulation:

In this project, we want to approximate the result of a minimization problem using a neural network (NN). Our problem $P(\cdot)$ is composed of a linear objective with linear and box constraints:

$$P(\boldsymbol{b}, \boldsymbol{d}, c) = \left[\min_{\boldsymbol{\phi}} \left[\boldsymbol{\phi}^T \boldsymbol{d} \right] \quad \text{subject to} \quad \begin{cases} 0 \leq \boldsymbol{\phi} \leq 1 \\ \boldsymbol{\phi}^T \boldsymbol{b} = c \end{cases} \right], \tag{1}$$

where $\boldsymbol{\phi}, \boldsymbol{b}, \boldsymbol{d} \in \mathbb{R}^5$, $c \in \mathbb{R}$, and then, $P : \mathbb{R}^{11} \to \mathbb{R}$. We define the subset $\mathcal{A} \subset \mathbb{R}^{11}$ as the set where the values of $\boldsymbol{b}, \boldsymbol{d}$, and c can exists.

We want to find an approximate function $\tilde{P}(\cdot)$ such that

$$||P - \tilde{P}||_{L^\infty(\mathcal{A})} \leq C,$$

where C is an arbitrary positive constant, the idea is that evaluate $\tilde{P}(\cdot)$ is faster than evaluate the original function $P(\cdot)$.

Suggested approximation

The function $\tilde{P}(\cdot)$ can be a trained NN using real solutions of (1).

Suggested literature and tutorials

1. `https://scikit-learn.org/stable/`. It has tutorials, and it is suitable for standard NN.

2. `https://pytorch.org/`. It is useful if you want to design your architecture or to play with loss functions.

3. `https://www.fast.ai/`. It is more practical and friendly for people no familiar with math.

These links were suggested by a Ph.D. student researching AI during an AI summer school (`http://acai2019.tuc.gr/`).

Second formulation (Lanza):

In this second formulation we want to approximate

$$P(\boldsymbol{b},\boldsymbol{d},\boldsymbol{Q},c) = \left[\min_{\boldsymbol{\phi}} \left[\boldsymbol{\phi}^T\boldsymbol{d}\right] \quad \text{subject to} \quad \begin{cases} 0 \leq \boldsymbol{\phi} \leq 1 \\ \boldsymbol{\phi}^T\boldsymbol{Q}\boldsymbol{\phi} + \boldsymbol{\phi}^T\boldsymbol{b} = c \end{cases} \right], \tag{2}$$

where $\boldsymbol{\phi},\boldsymbol{d},\boldsymbol{b} \in \mathbb{R}^{12}$, and $Q \in M^{12\times 12}$. As the matrix $\boldsymbol{Q}$ has only 8 non-zero entries, we storage it as a vector.

About the data

The data is in Matlab format, and they are cells. Each file has a cell with dimensions 1×225. Each one of these cells has inside a 1×5 cell with 5 arrays. The arrays have dimensions 12, 1, 12, 8, and 1, respectively. These 5 arrays represent a single data point, where the first 4 arrays are the input, and the 5-th array is the output.

In other words, each data point is composed by:

$$\Bigg(\underbrace{\boldsymbol{b}, c, \boldsymbol{d}, \boldsymbol{Q}}_{Inputs}, \underbrace{P(\boldsymbol{b},\boldsymbol{d},\boldsymbol{Q},c)}_{Output}\Bigg).$$