Alternating Mixed-Integer Programming and Neural Network Training for Approximating Stochastic Two-Stage Problems

Taehyeon Kwon

Department of Industrial and Manufacturing Engineering
The Pennsylvania State University

Overview

- **Key Points:**
  - Efficient method to solve two-stage stochastic problems (2SP)
  - Combine mixed-integer programming (MIP) with neural network (NN) training
  - Aims to reduce computational costs and enhance decision-making under uncertainty.

Two-Stage Stochastic Problem

$$\min_{x \in \mathcal{X}} \; G(x) + \mathbb{E}_{\mathbb{P}} \left(\min_{y \in \mathcal{Y}(x,\xi)} c^\top y \right) \qquad (1)$$

Notation

- x : First Stage decision variable; $x \in \mathcal{X}$
- y : Second Stage decision variable; $y \in \mathcal{Y}(x, \xi)$, where $\mathcal{Y}(x, \xi) \subseteq \mathbb{R}^n \times \{0,1\}^l$
- ξ : Random vector; $\xi \in \Omega$, where $\Omega \subseteq \mathbb{R}^l$
- $\mathbb{P}$: Probability distribution; $\mathbb{P} \in \mathcal{P}(\Omega)$

Two-Stage Stochastic Problem

Original 2SP

$$\min_{x \in \mathcal{X}} \; G(x) + \mathbb{E}_{\mathbb{P}}\left(\min_{y \in \mathcal{Y}(x,\xi)} c^\top y \right)$$

$\rightarrow$

Modified 2SP

$$\min_{x \in \mathcal{X}} \; G(x) + Q(x)$$

$$\text{s.t.} \quad Q(x) := \frac{1}{m} \sum_{j=1}^{m} \left(\min_{y \in \mathcal{Y}(x,\xi_j)} c^\top y \right)$$

, where ξ_j denotes scenarios sampled from $\mathbb{P}$

Aim: Learn the mapping $x \mapsto Q(x)$, by **training** a NN and **embedding** trained NN to *original 2SP.*

Methodology Overview

- **Key Points:**
  - Novel integration of **MIP** and **NN** training
  - MIP optimizes decisions under uncertainty (ξ)
    - NN predicts second-stage outcomes ($Q(x)$).

Feedforward Neural Network

- **Properties:**
  - **4** layer: 2 hidden layer with 3 nodes each. Input layer and output layer have 2 and 1 nodes respectively.
  - Each layer have **ReLu** (i.e., $\mathrm{ReLu}(x) = \max(0,x)$) function.

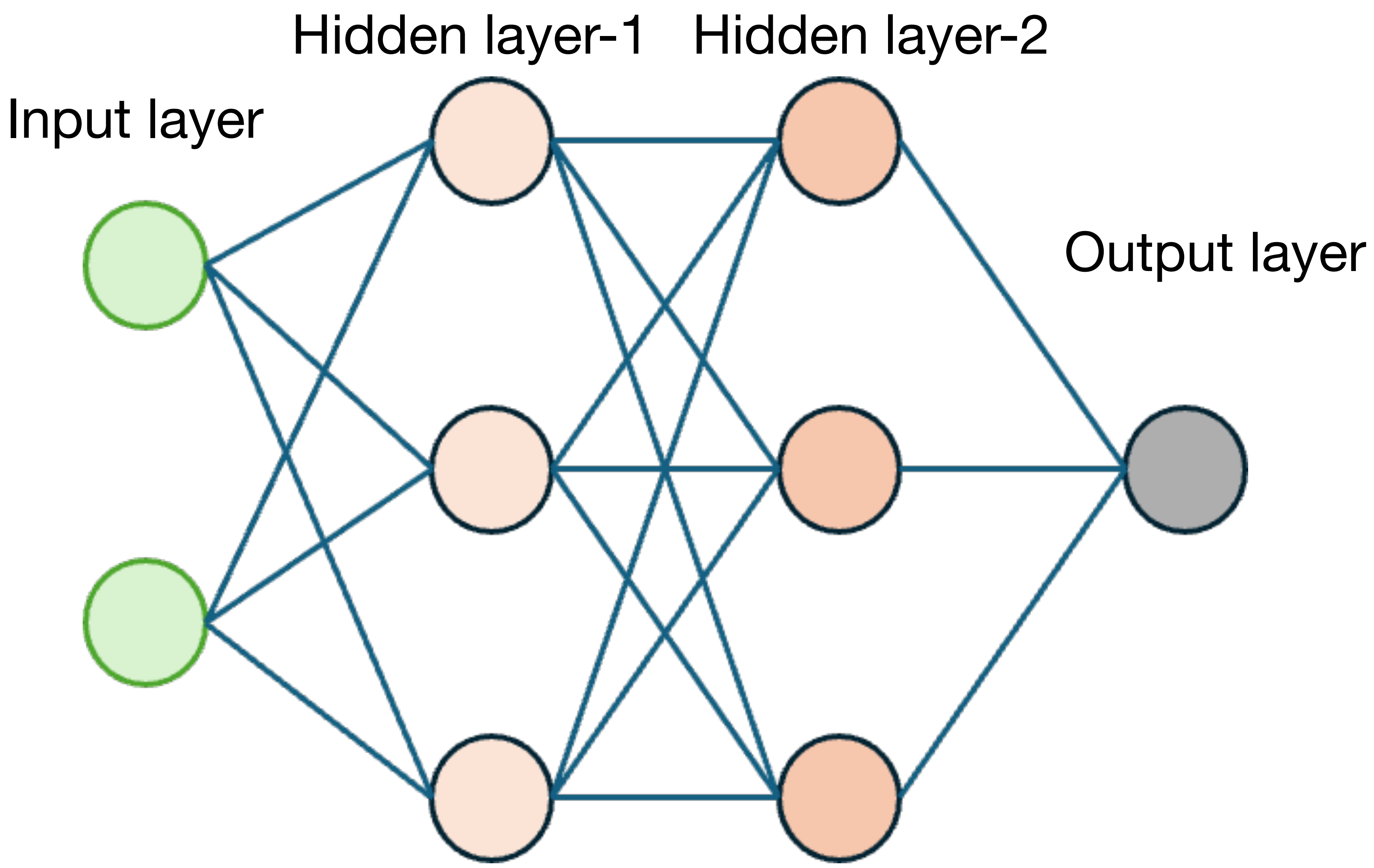
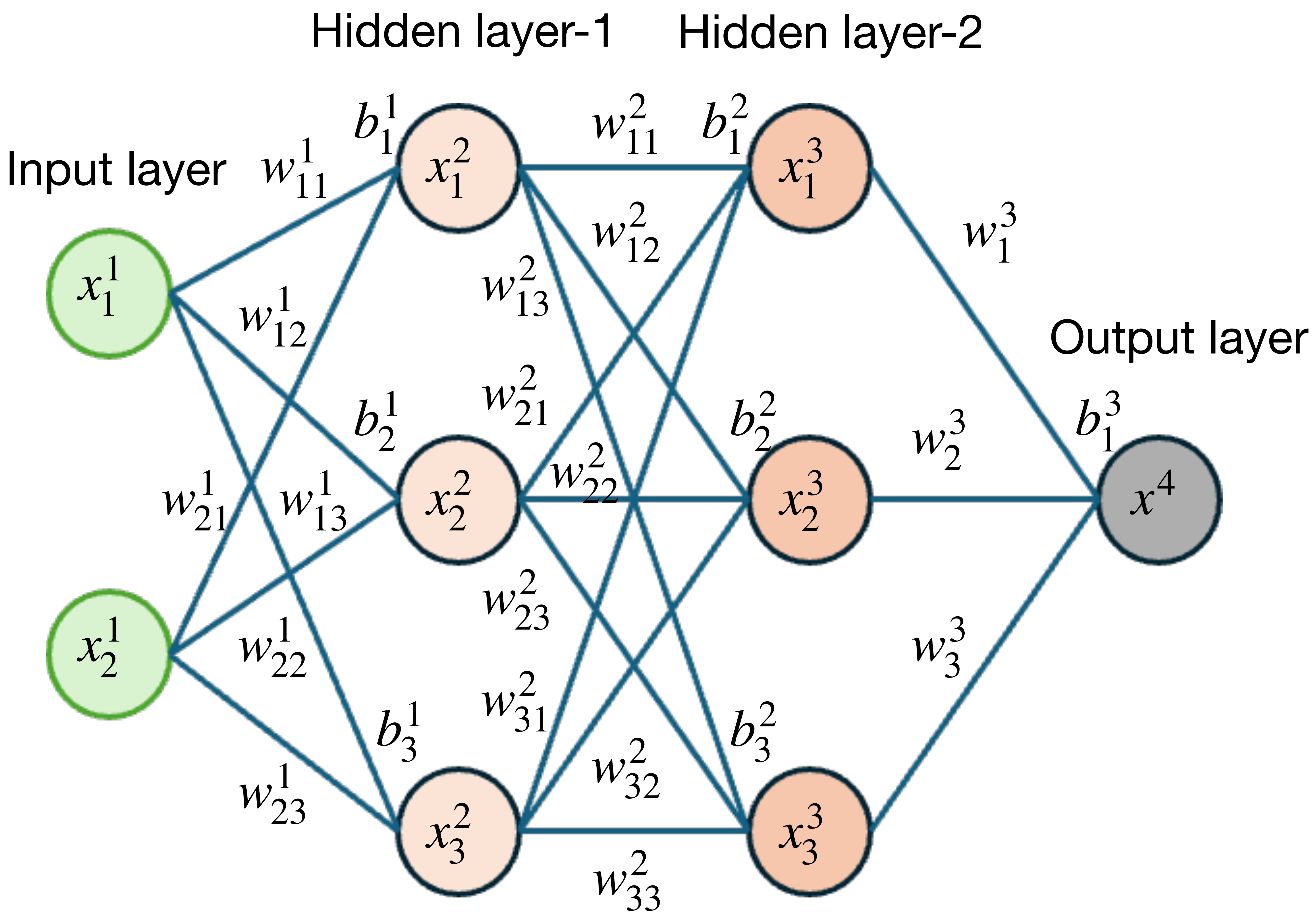
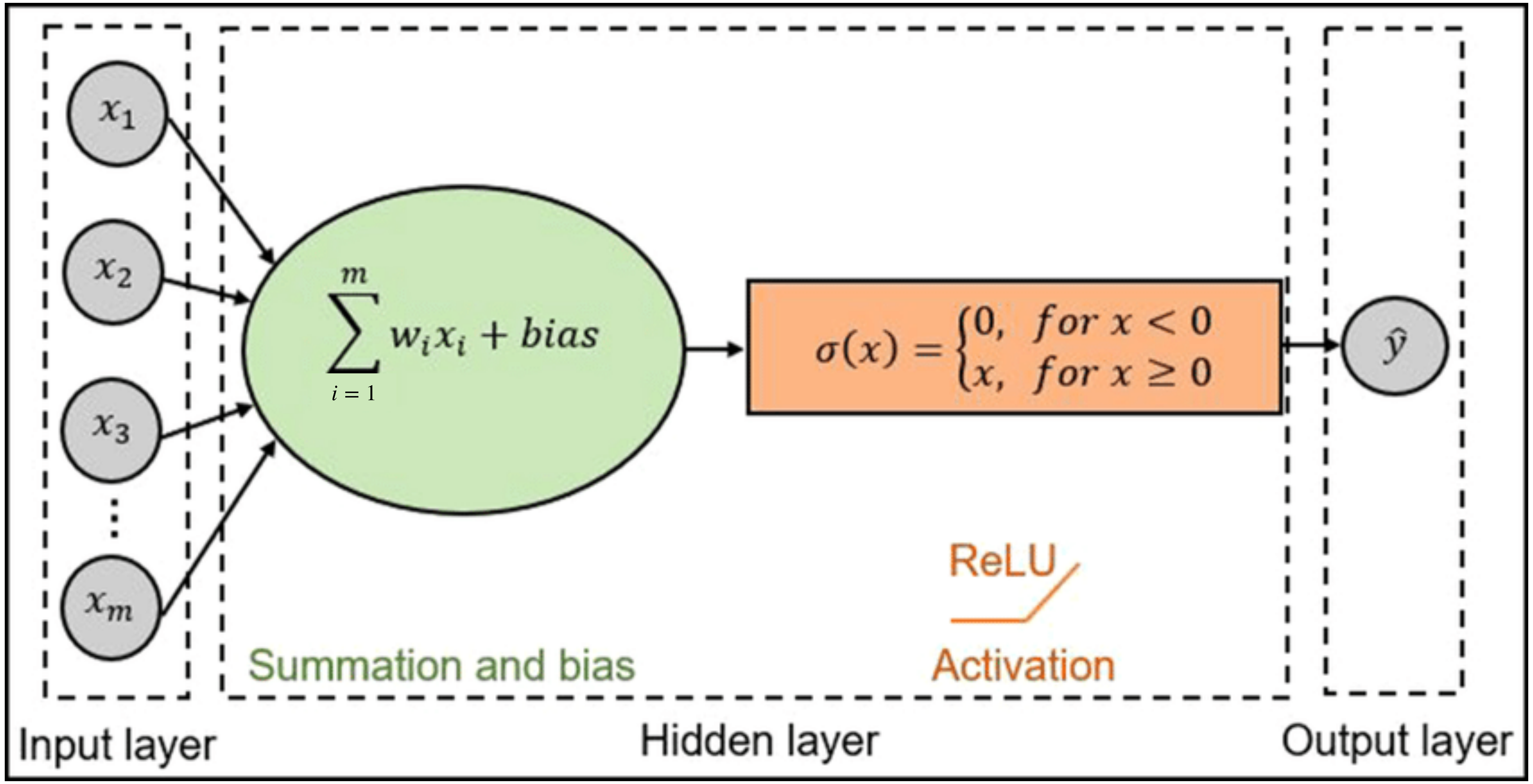


Fig: Feedfoward Neural Network Architecture

Embedding trained NN to 2SP



Feedforward Neural Network



Embedding trained NN to 2SP

How to embed the trained NN to 2SP regrading ReLu function?

$$(w_i^l)^\top x^l + b_i^l \leq x_i^{l+1}, \quad \forall l \in [L-1]$$
$$(w_i^l)^\top x^l + b_i^l - (1-\sigma_i^l)\text{LB}_i^{l+1} \geq x_i^{l+1}$$
$$x_i^{l+1} \leq \sigma_i^l \text{UB}_i^{l+1}, \quad \forall l \in \{2,\ldots,L\}$$
$$x_i^{l+1} \geq 0, \quad \forall i \in [n_{l+1}]$$
$$\sigma_i^l \in \{0,1\}, \quad \forall l \in \{2,\ldots,L\}$$

- l : layer number; $l \in \{1,\ldots,L\}$, our case $L = 4$
- i : node number; $i \in \{1,\ldots,n_{l+1}\}$, where n_{l+1} denotes number of nodes in $(l+1)$-th layer.
- σ : Indicator of ReLu function.
- LB & UB : lower bound and upper bound of results of linear combination with weight, input and bias.
  - $\text{LB}_i^{l+1} \leq (w_i^l)^\top x_l + b_i^l \leq \text{UB}_i^{l+1}$

New 2SP by embedding trained NN

$$
\begin{aligned}
\min \quad & G(x^1) + x^L \\
\text{s.t.} \quad & W^l x^l + b^l \le x^{l+1} \quad , \forall l \in [L-1] \\
& W^l x^l + b^l - \operatorname{diag}(LB^{l+1})(1-\sigma^{l+1}) \ge x^{l+1} \quad \forall l \in [L-1] \\
& x^l \le \operatorname{diag}(UB^l)\sigma^l \quad \forall l \in \{2,\ldots,L\} \\
& x^L = W^{L-1}x^{L-1} + b^{L-1} \\
& \sigma^l \in \{0,1\}^{n_l}, \quad \forall l \in \{2,\ldots,L\} \\
& x^l \in \mathbb{R}_+^{n_l} \quad \forall l \in [L-1] \\
& x^1 \in X \ , x^L \in \mathbb{R}
\end{aligned}
\tag{5}
$$

x^L is the output of the neural network $\approx Q(x^1)$, where x^1 := input layer

Alternating MIP-NN Algorithm for 2SP (MIP-NN 2SP)

Algorithm 1 Alternating MIP-NN Algorithm for 2SP (MIP-NN 2SP)

```
Input Initial training data 𝒳̄, α = 0.99
Output Approximately optimal solution x*_100 for (1)
Train a NN 2 × 40 with training data set 𝒳̄
for k = 1, . . . , 100 do
    x*_k ← argmin (5)
    for i = 1, . . . , 50 do
        Sample x_i ∈ X uniformly
        x_i ← αx*_k + (1 − α)x_i
        𝒳̄ ← 𝒳̄ ∪ {x_i}
    end for
    Retrain NN with 𝒳̄
end for
```

End of the training, $\bar{\mathcal{X}}$ have approximately **5100** data point