Friedrich-Alexander-Universität
Technische Fakultät

Deep Neural Network
&
Mixed-Integer Linear Programming

Presentation
By
Anuraag Mishra

Master Seminar
Selected Topics in Data Science and Optimization

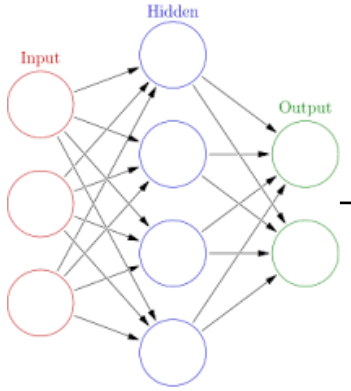
Agenda

1. Introduction
2. DNN
   - 2.1. Neuron
   - 2.2. Artificial Neuron
   - 2.3. Activation function (ReLU)
3. MILP
   - 3.1. What is MILP?
   - 3.2. Example
   - 3.3. Branch & Bound
   - 3.2. Bound Tightening
4. Formulate DNN for MILP
   - 4.1. Define ReLU as LP
   - 4.2. ReLU Bounds
5. Application
   - 5.1. Experimental Setup
   - 5.2. Feature Visualization
   - 5.3. Adversarial Examples
6. Result & Analysis
7. Why MILP?
8. Conclusion & Future work
9. Summary

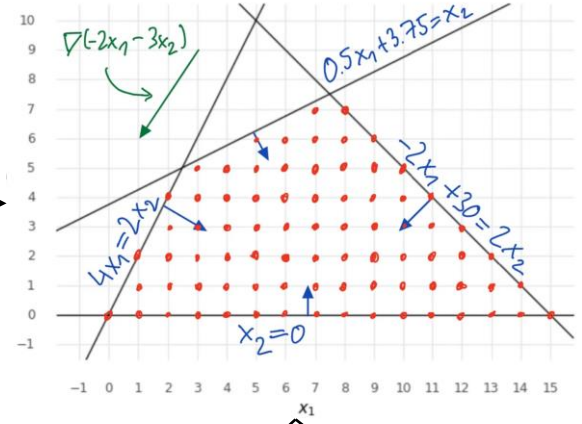
Introduction

DNN & MILP

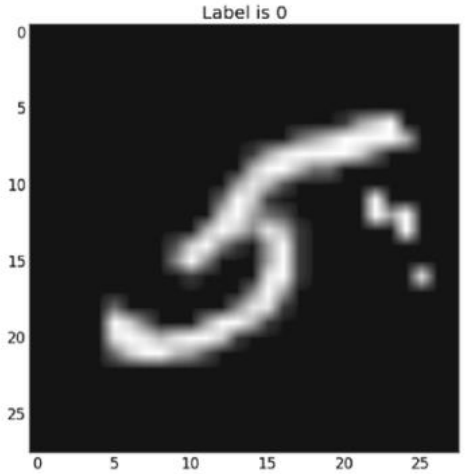
DNN



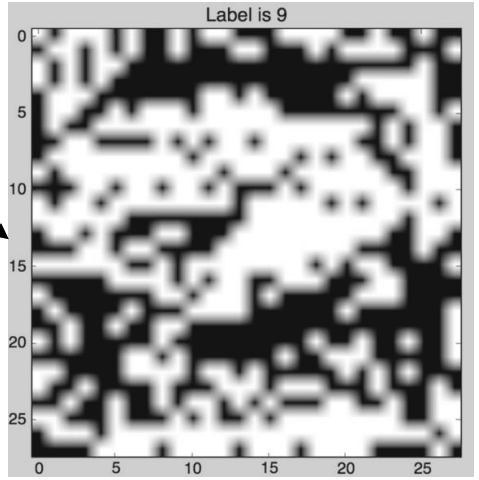
MILP



Adversarial Example

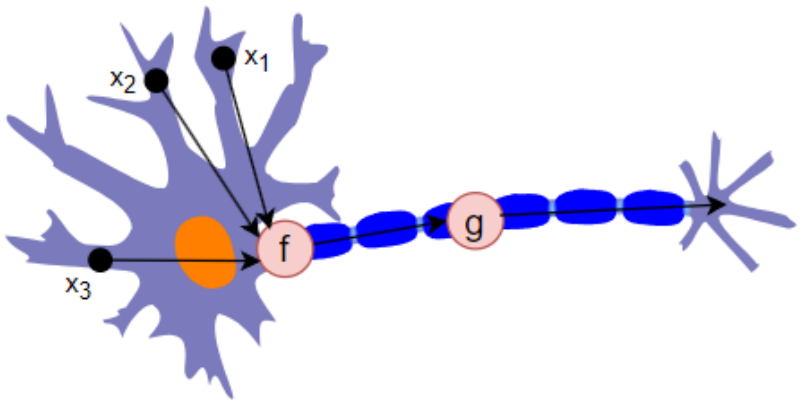


Feature Visualization

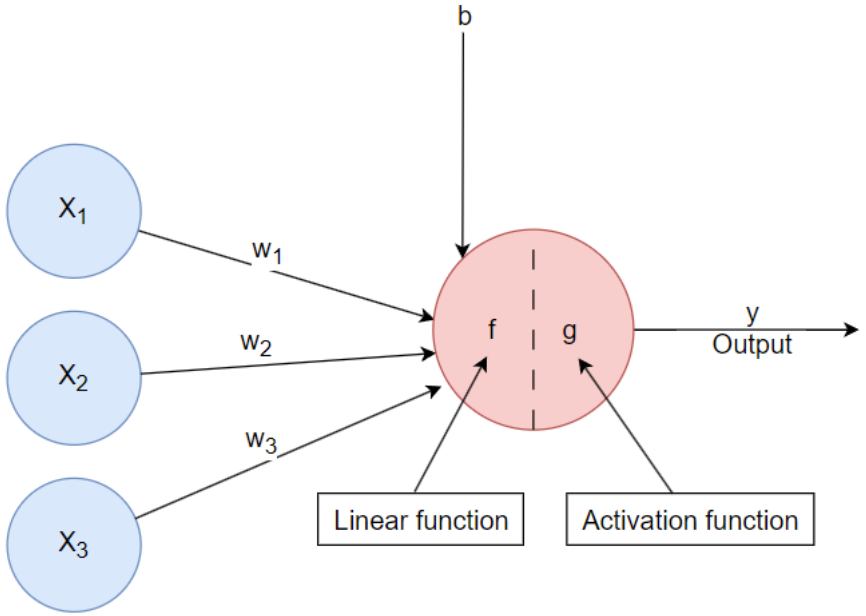


Deep Neural Networks (DNNs)

Neuron



Neuron



Artificial Neuron

Deep Neural Networks (DNNs)

Artificial Neuron

$$f(x) = w^T x + b$$

$$y = g(f(x))$$

$$y = g(w^T x + b)$$

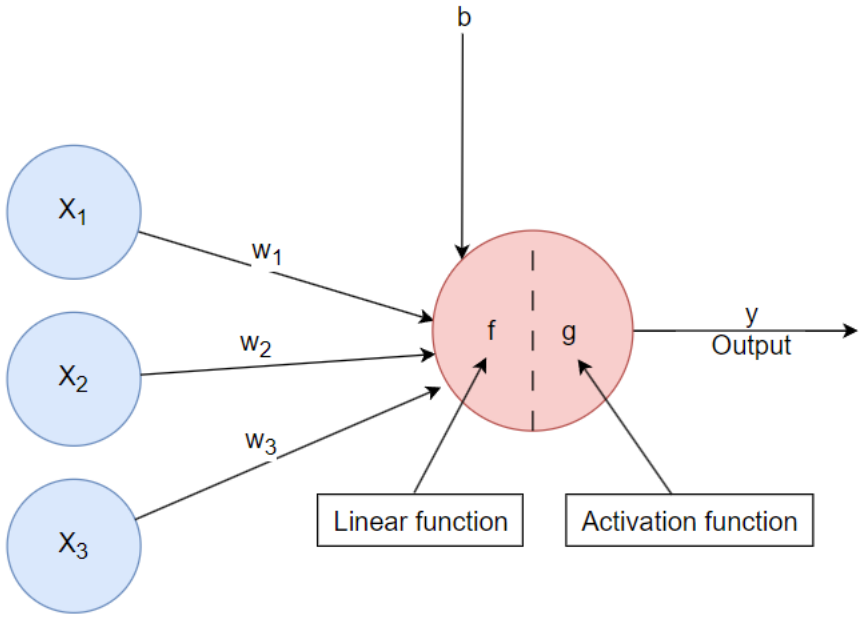
where g is the activation function,

w^T is transpose of w

$\mathrm{x} = [x_1, x_2, x_3]$

$w = [w_1, w_2, w_3]$

b is constant



Deep Neural Networks (DNNs)

Artificial Neural Network

$$f = w^T x + b$$

$$y = g(f)$$

$$y = g(w^T x + b)$$

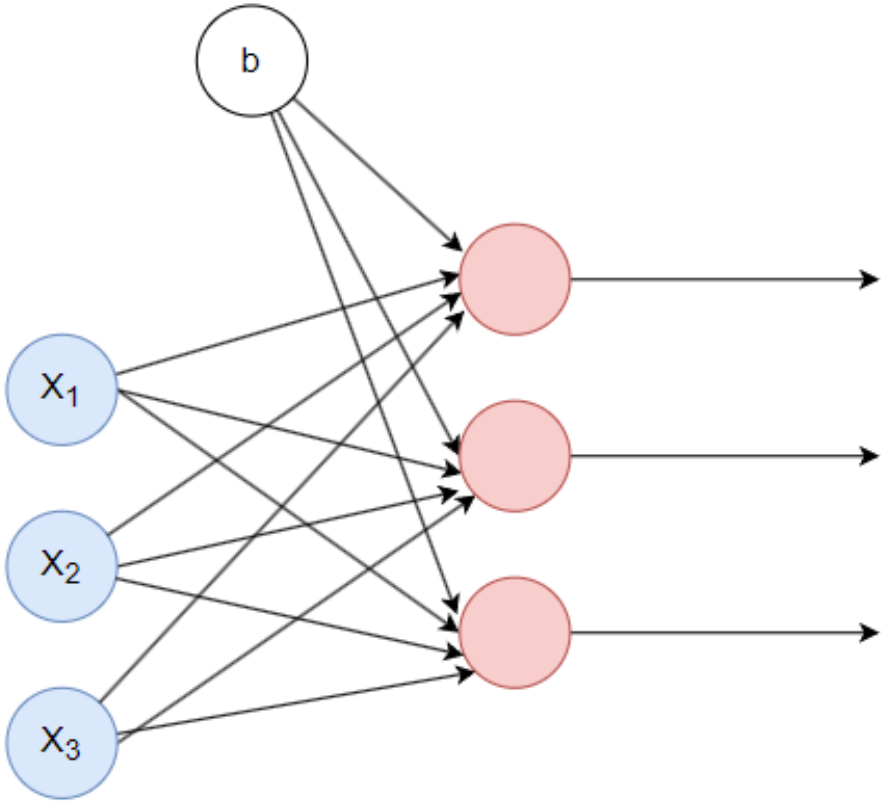
where g is the activation function,

w^T is transpose of w

$$\mathrm{x} = [x_1, x_2, x_3]$$

$$w = \begin{bmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \\ w_{31} & w_{32} & w_{33} \end{bmatrix}$$

$$\mathrm{b} = [b_1, b_2, b_3]$$



Deep Neural Networks (DNNs)

For a layer k,

$$x^k = g((W^k)^T . x^{k-1} + b^k)$$

where g is the activation function,

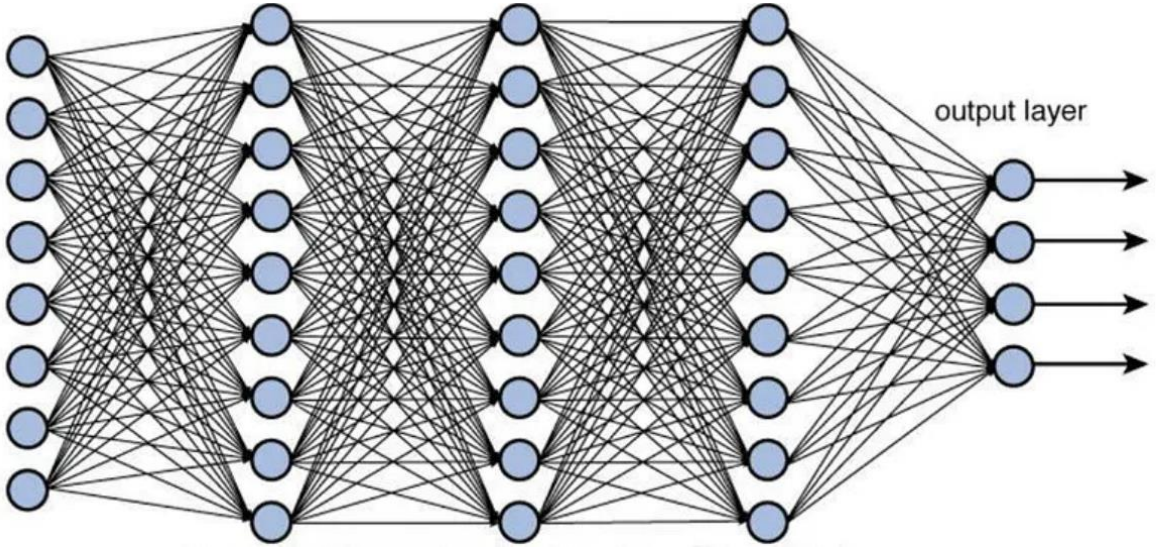
W^T is transpose of W

$$x^k = [x_1^k, x_2^k, \ldots . x_n^k]$$

$$w^k = \begin{bmatrix} w_{11}^k & \cdots & w_{1n}^k \\ \vdots & \ddots & \vdots \\ w_{m1}^k & \cdots & w_{mn}^k \end{bmatrix}$$

m - # neurons in the hidden layer k

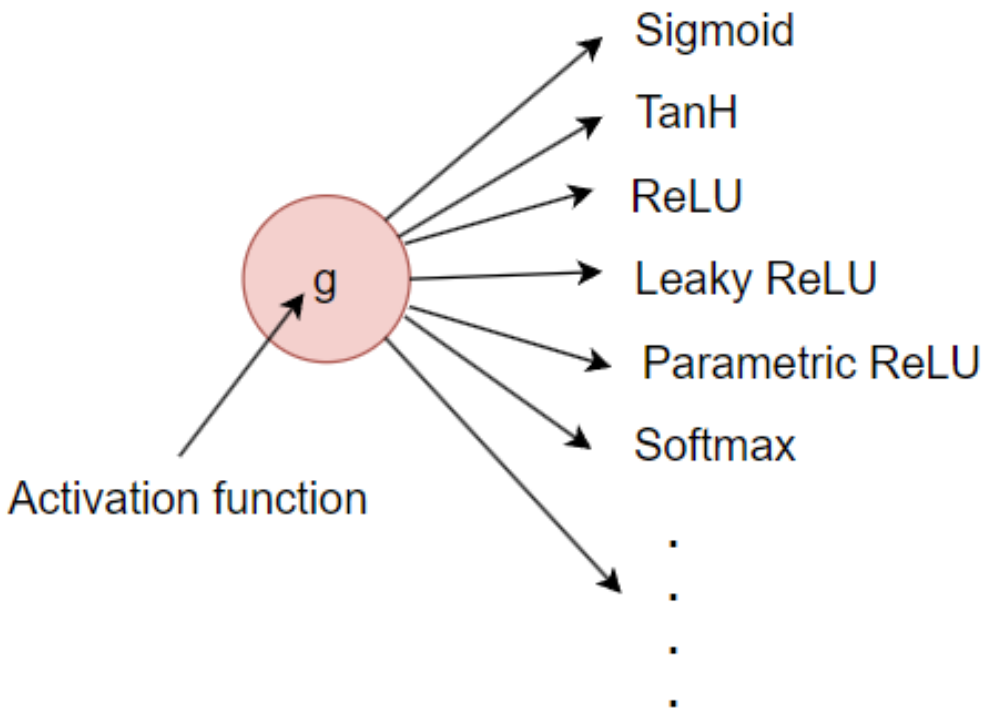
n - # inputs to the hidden layer k

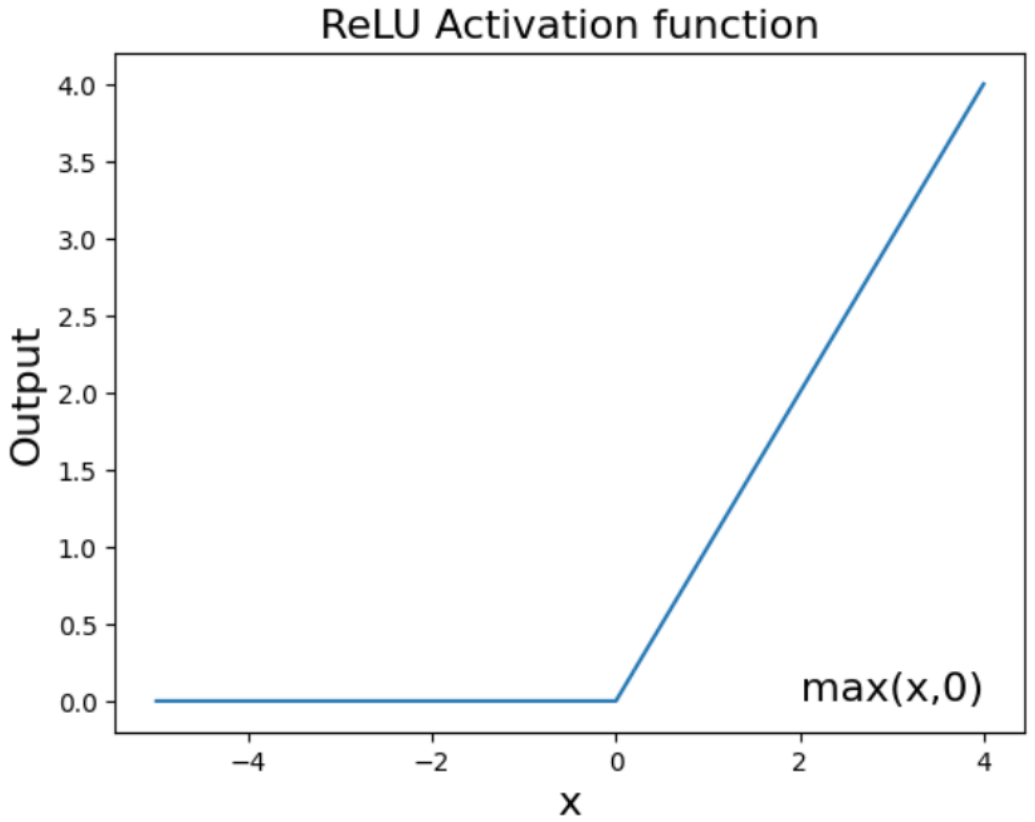


(PDF) Energy-Efficient IoT Sensor Calibration With Deep Reinforcement Learning (researchgate.net)

Deep Neural Networks (DNNs)

Activation function: ReLU





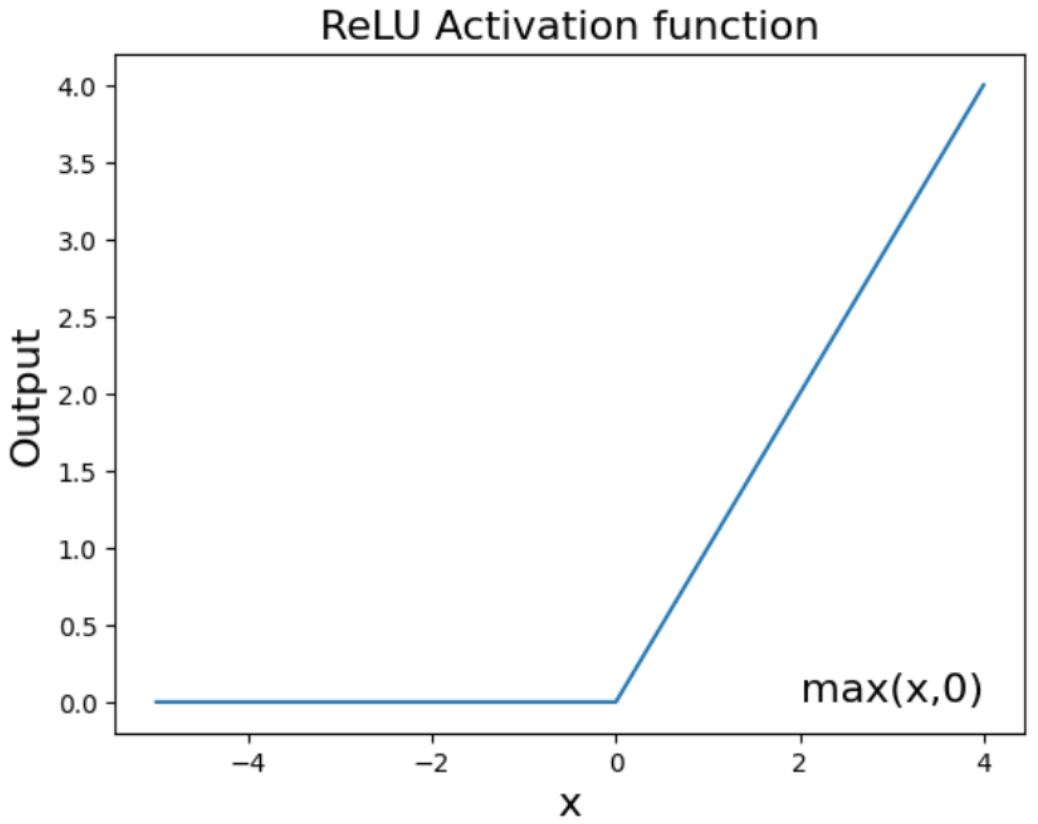
FAU_DCN_AvH_MasterThesis_may2023.pdf

Deep Neural Networks (DNNs)

Activation function: ReLU

$$x^k = ReLU((w^k)^T . x^{k-1} + b^k)$$

$$ReLU(x) = max(x, 0)$$



Mixed Integer Linear Programming (MILP)

What is MILP?

Definition: MILP is a mathematical method used for solving optimization problems where some of the decision variables are constrained to be integers.

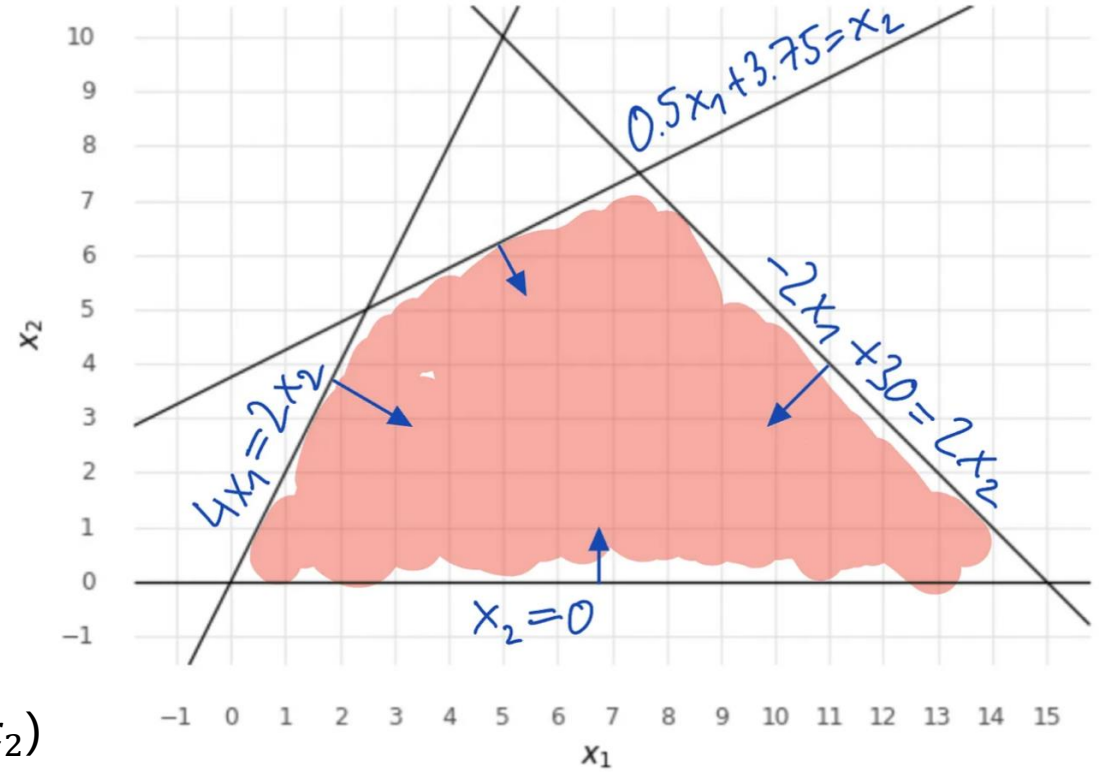
Components of MILP:

Decision Variables

Constraints

Objective Function

$$(max - 2x_1 - 3x_2)$$



MILP: Formal definition and solution space | by István Módos | Towards Data Science

Mixed Integer Linear Programming (MILP)

Small-Scale Lemonade Production

Example: Small-Scale Lemonade Production (with a budget of 10 Euro).

Constraints:

- **Cost Constraint** : $2L + 3S \leq 10$
- **Production Constraints**: $L \geq 1 \ \& \ \ S \geq 2$
- **Quality Constraint**: $L \leq 4$

Objective function: Maximize the cost of producing lemonade.

$$\text{Cost}(max)\text{: } 2L + 3S$$

Where 2 and 3 are the costs per unit of lemons(L) and sugar(S) packets, respectively.

L is Number of lemons & S is Number of sugar packets

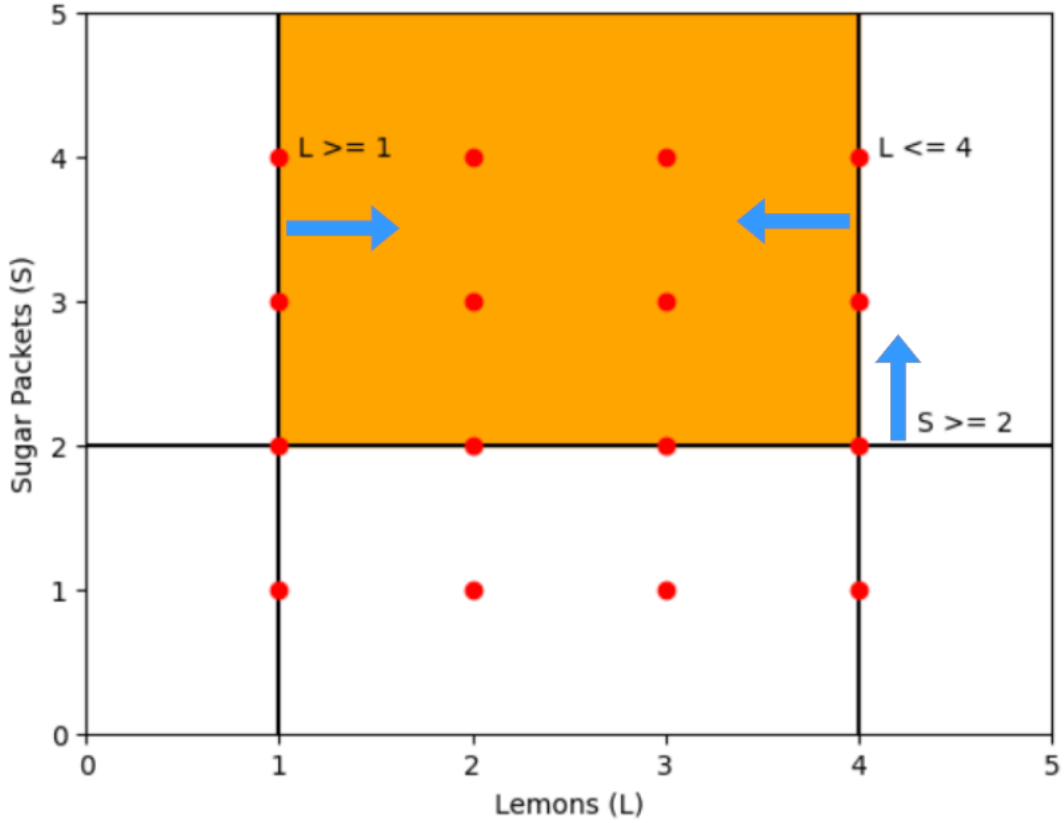
Mixed Integer Linear Programming (MILP)

Small-Scale Lemonade Production: Problem Visualization

Cost Constraint : $2L + 3S \leq 10$

Production Constraints: $L \geq 1 \;\&\; S \geq 2$

Quality Constraint: $L \leq 4$



Mixed Integer Linear Programming (MILP)

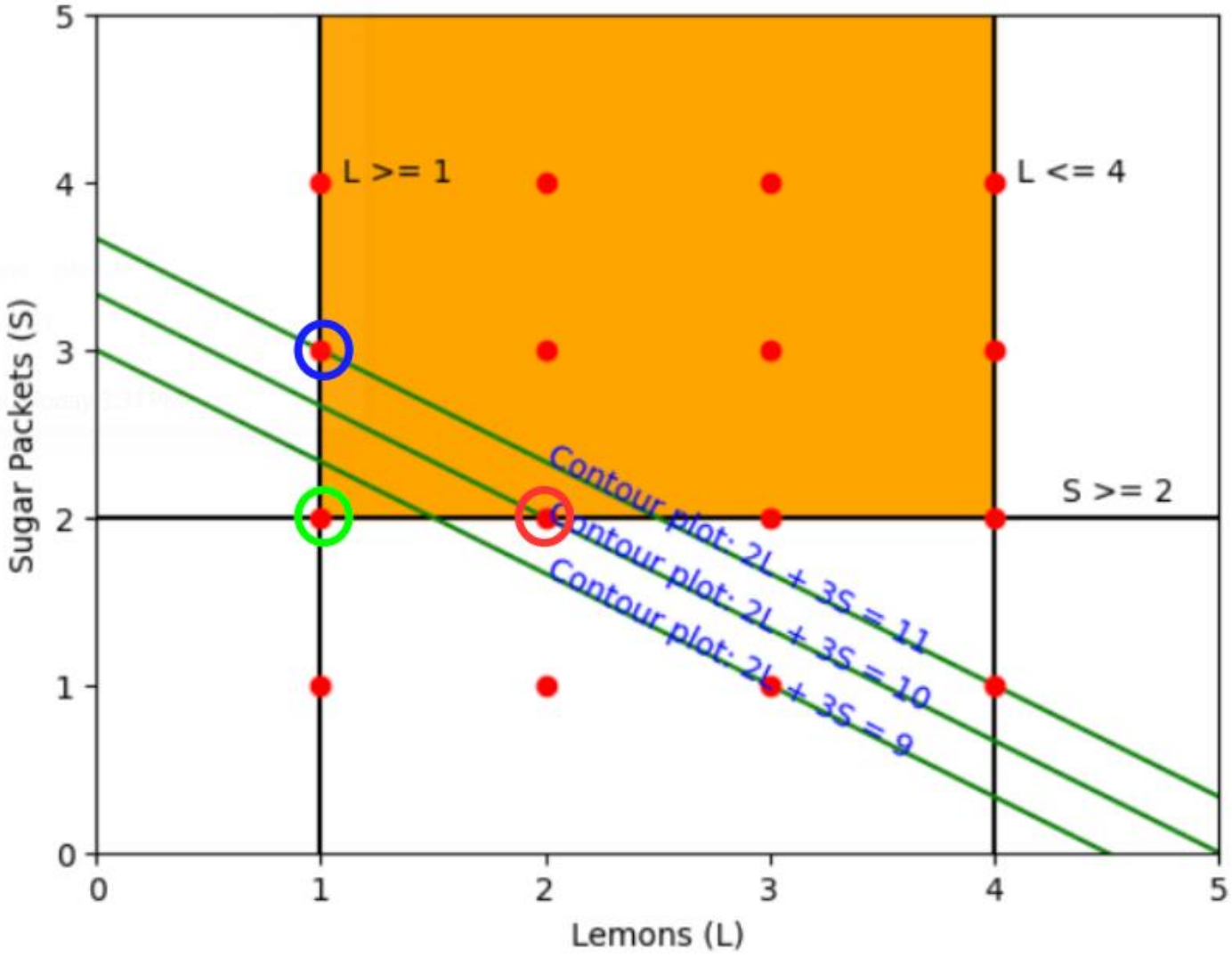
Small-Scale Lemonade Production: Solution Visualization

Cost Constraint : $2L + 3S \leq 10$

Production Constraints: $L \geq 1 \;\&\; S \geq 2$

Quality Constraint: $L \leq 4$

Objective function: $(\max) 2L + 3S$



Mixed Integer Linear Programming (MILP)

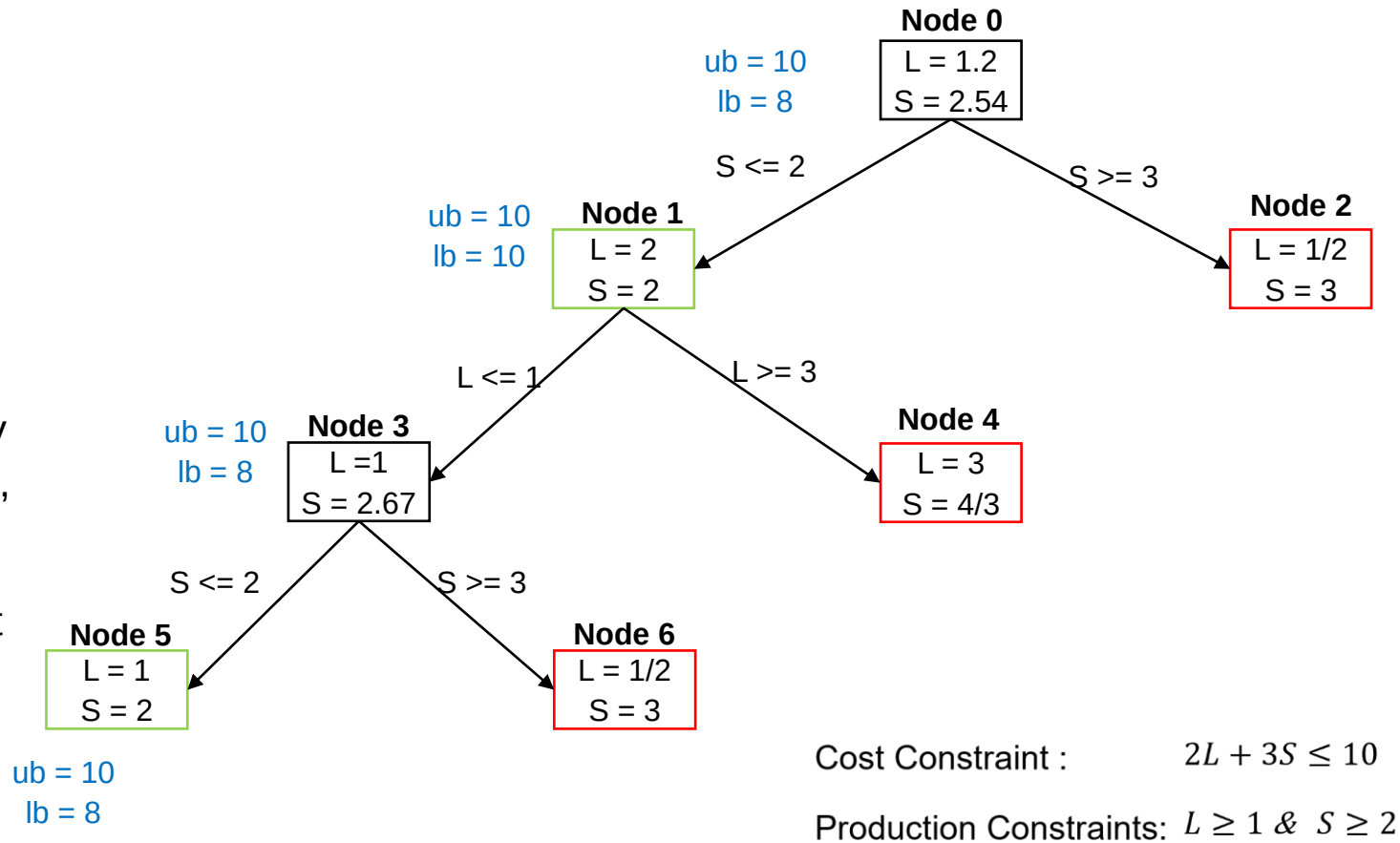
Branch and Bound

LP Relaxation: Ignore the integer constraints, gives us a baseline for solution.

Start with L = 1.2 & find S with cost constraint. Starting value of S is set by solving the cost constraint given L=1.2

First Branch: If the LP solution yields a fractional value for the integer variable (Sugar packets, S), create two new subproblems (branches) by rounding this value up and down. For example, if S =2.67, create branches with S=2 and S=3.

Subsequent Branching: Each subproblem might still have fractional values for S, so continue branching until you reach solutions where S is an integer.



Cost Constraint : $2L + 3S \leq 10$

Production Constraints: $L \geq 1 \;\&\; S \geq 2$

Quality Constraint: $L \leq 4$

Objective function: $2L + 3S$

Mixed Integer Linear Programming (MILP)

Bound Tightening

Tightening Bounds for S:

Starting with $L = 1$ (its lowest value) ⇒ cost constraint ⇒ $S \leq 8/3 = 2.67$

Since S is an integer, its maximum possible value here is 2 (upper bound).

Since S>=2, therefore $S \Rightarrow lb = ub = 2$

Cost Constraint :	$2L + 3S \leq 10$
Production Constraints:	$L \geq 1$ & $S \geq 2$
Quality Constraint:	$L \leq 4$
Objective function:	$2L + 3S$

Tightening Bounds for L:

Starting with $S = 2$ (its lowest value), ⇒ cost constraint ⇒ $L = 2$ (upper bound).

Since $L \geq 1$ (lower bound), therefore $1 \leq L \leq 2$.

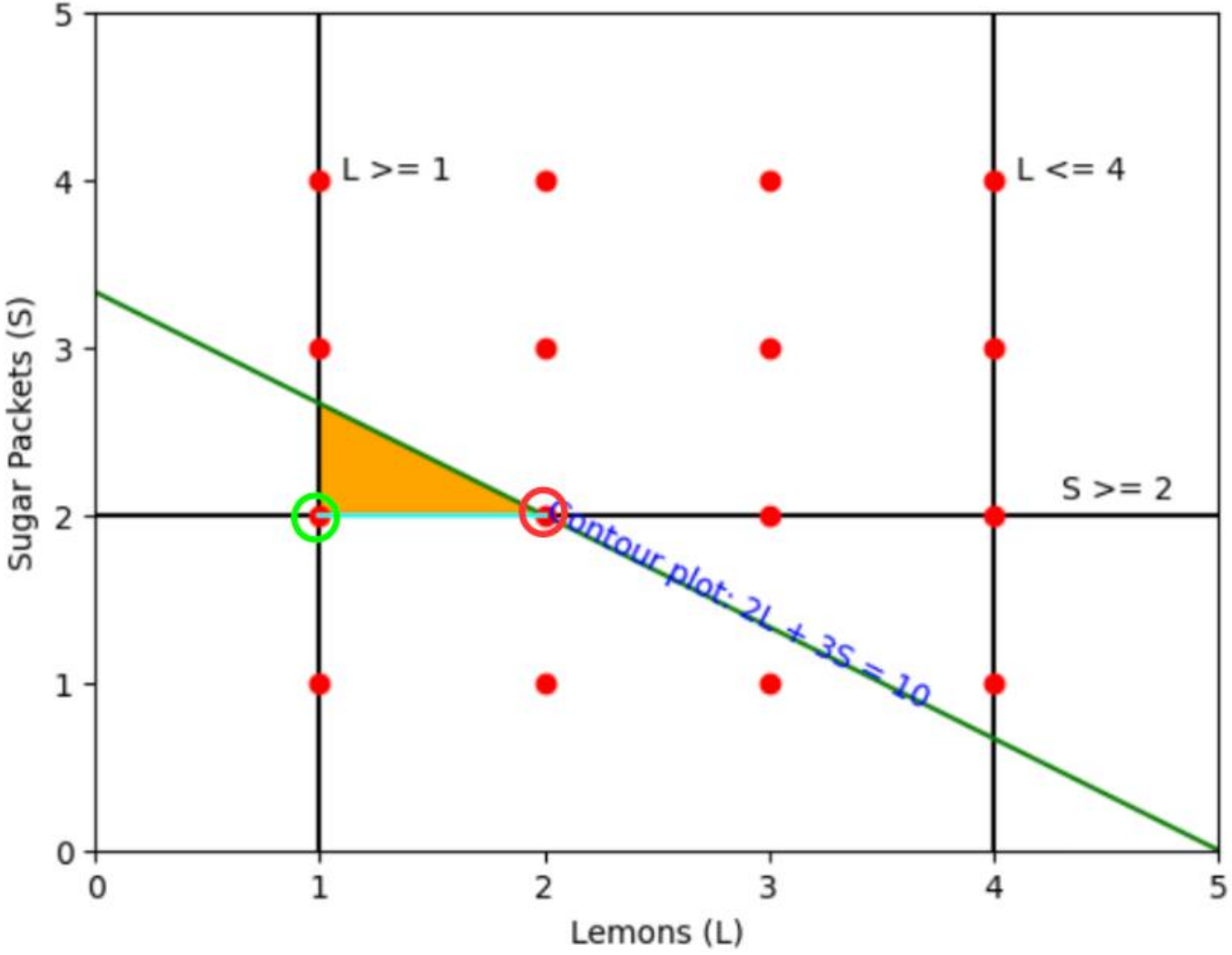
Mixed Integer Linear Programming (MILP)

Bound Tightening applied to solution-space

Bounds:

$For\ S \Rightarrow lb = ub = 2$

$For\ L\ \Rightarrow lb = 1, ub = 2$



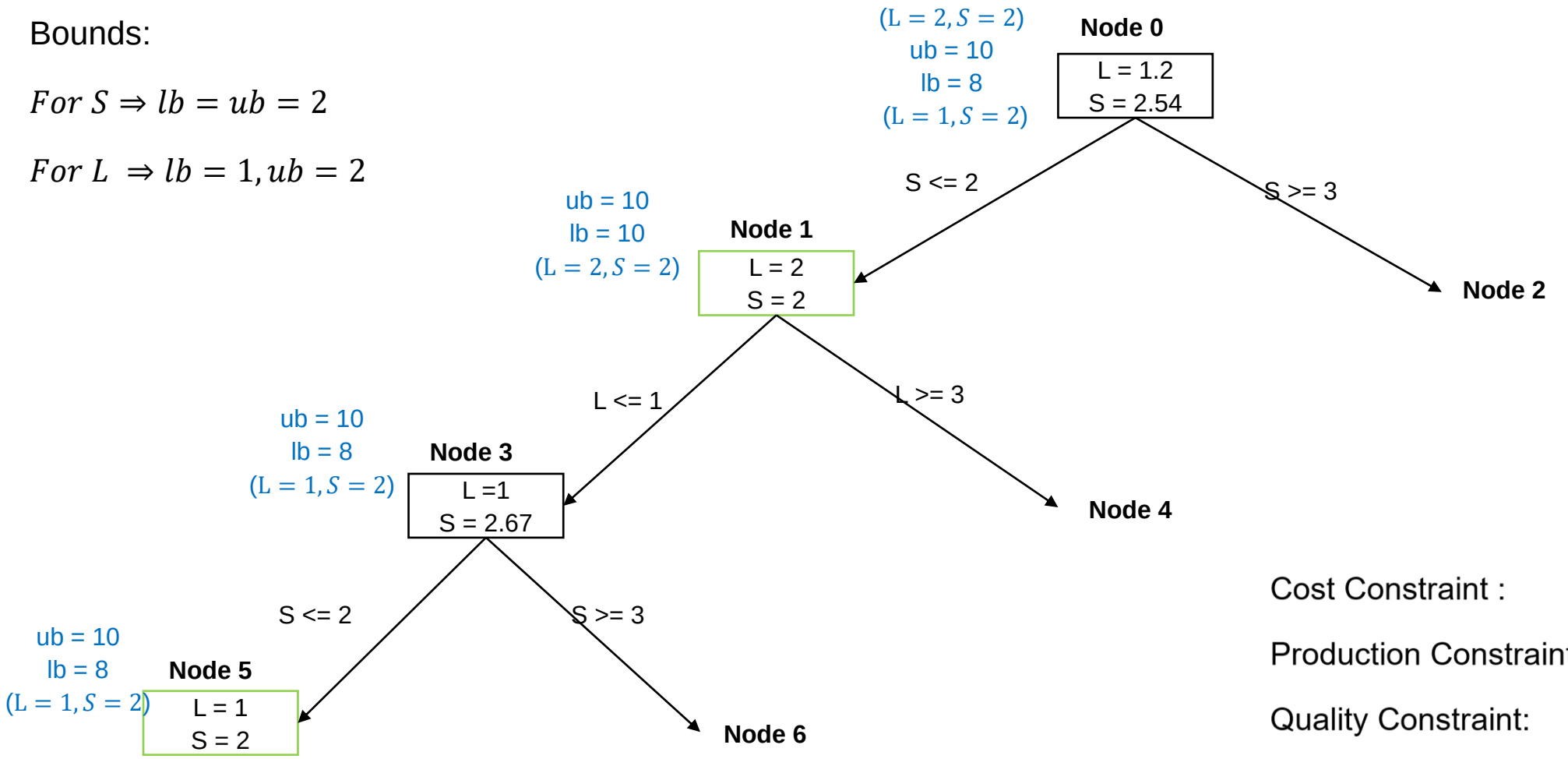
Mixed Integer Linear Programming (MILP)

Bound Tightening applied to branch-n-bound

Bounds:

$For\ S \Rightarrow lb = ub = 2$

$For\ L \Rightarrow lb = 1, ub = 2$



Cost Constraint : $2L + 3S \leq 10$

Production Constraints: $L \geq 1\ \&\ S \geq 2$

Quality Constraint: $L \leq 4$

Objective function: $2L + 3S$

Formulate DNN for MILP

Define ReLU as LP

$$x^k = ReLU((w^k)^T x^{k-1} + b^k) \quad (1)$$

$$ReLU(x) = \max(x, 0) \quad (2)$$

$$(w^k)^T x^{k-1} + b^k = x^k - s^k, \qquad x^k \geq 0, s^k \geq 0 \quad (3)$$

The solution (x, s) of constraints (3) is not unique (as it should be because ReLU() is in fact a function), because one can always take any scalar δ ≥ 0 and obtain a still-feasible solution $(x + \delta, s + \delta)$. Imposing $\delta = 0$ is therefore the only critical issue to be addressed when modeling the ReLU operator.

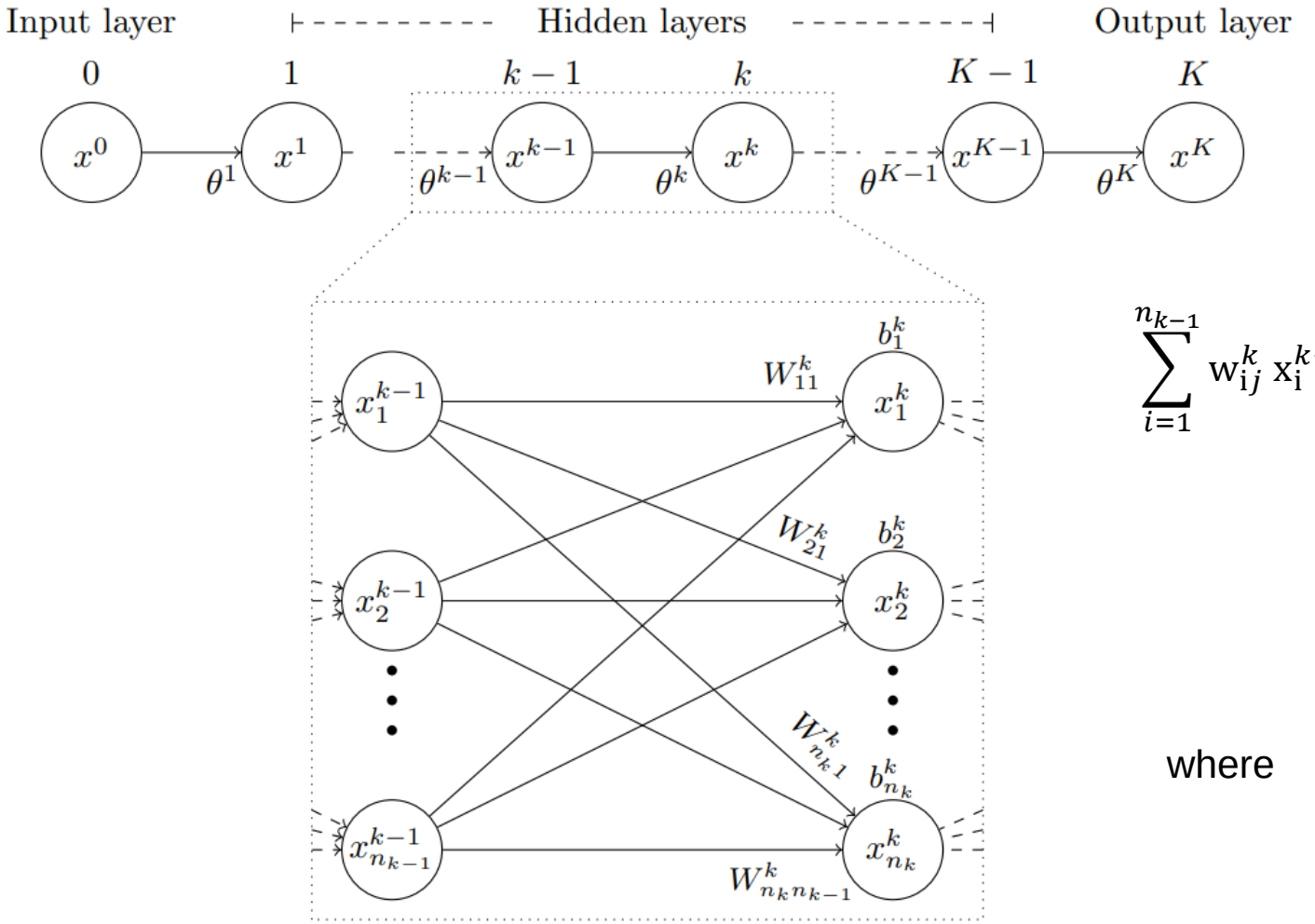
$$z = 1 \rightarrow x = 0 \quad (4a)$$

$$z = 0 \rightarrow s = 0 \quad (4b)$$

$$z \in \{0, 1\} \quad (4c)$$

Formulate DNN for MILP

Define ReLU as LP



$$\sum_{i=1}^{n_{k-1}} \mathrm{w}_{ij}^k \, \mathrm{x}_i^{k-1} + b_j^k = \mathrm{x}_j^k - s_j^k, \qquad \mathrm{x}_j^k \geq 0, \mathrm{s}_j^k \geq 0 \qquad \text{(5a)}$$

$$z_j^k = 1 \rightarrow x_j^k \leq 0 \qquad \text{(5b)}$$

$$z_j^k = 0 \rightarrow s_j^k \leq 0 \qquad \text{(5c)}$$

$$z_j^k \in \{0, 1\} \qquad \text{(5d)}$$

where $k(hidden\ layer) = 1, \ldots, K-1,$
$j\ (neuron\ at\ layer\ k) = 1, \ldots, n_k$

ReLU networks as surrogate models in mixed-integer linear programs - ScienceDirect

Formulate DNN for MILP

ReLU bounds

Bounds:

$$lb_j^0 \leq \mathrm{x}_j^0 \leq \mathrm{ub}_j^0 \tag{6a}$$

$$lb_j^k \leq \mathrm{x}_j^k \leq ub_j^k \tag{6b}$$

$$\overline{\mathrm{lb}_j^k} \leq \mathrm{s}_j^k \leq \overline{\mathrm{ub}_j^k} \tag{6c}$$

$$lb_j^k = \overline{\mathrm{lb}_j^k} = 0, \qquad k \geq 1 \tag{6d}$$

$$\mathrm{u}b_j^k, \overline{\mathrm{ub}_j^k} \in R_+ + \cup \{+\infty\}, \qquad k \geq 1 \tag{6e}$$

Optimization function:

$$min \sum_{k=0}^{K} \sum_{j=1}^{n_k} c_j^k \, \mathrm{x}_j^k + \sum_{k=0}^{K} \sum_{j=1}^{n_k} \gamma_j^k \, \mathrm{z}_j^k \tag{7}$$

c_j^k and γ_j^k can be defined according to the specific problem.

Application

Experimental Setup

Classification problem: hand-written digit recognition of an input 28 x 28 figure (784 pixels/inputs x^0)

Pixels are normalized to grayscale level in $[0,1]$, where $1 \rightarrow white,\ 0 \rightarrow black$

Deep Neural Network is trained with 3 hidden layers with (8,8,8) ReLUs, plus a last layer made up of 10 units to classify digits ‘0’ to ‘9’

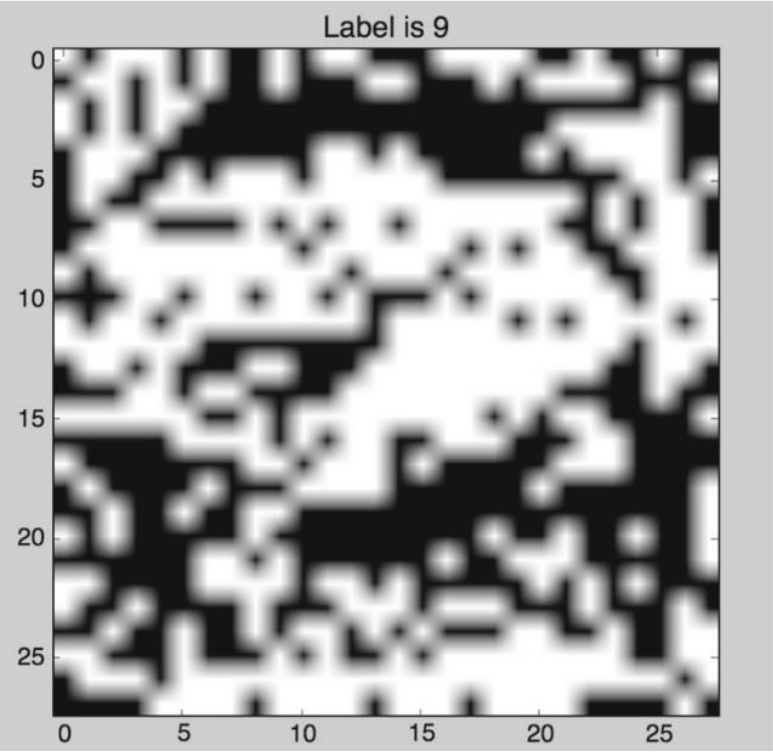
Training for 50 epochs, reaching an accuracy of 93.04% on the test set.

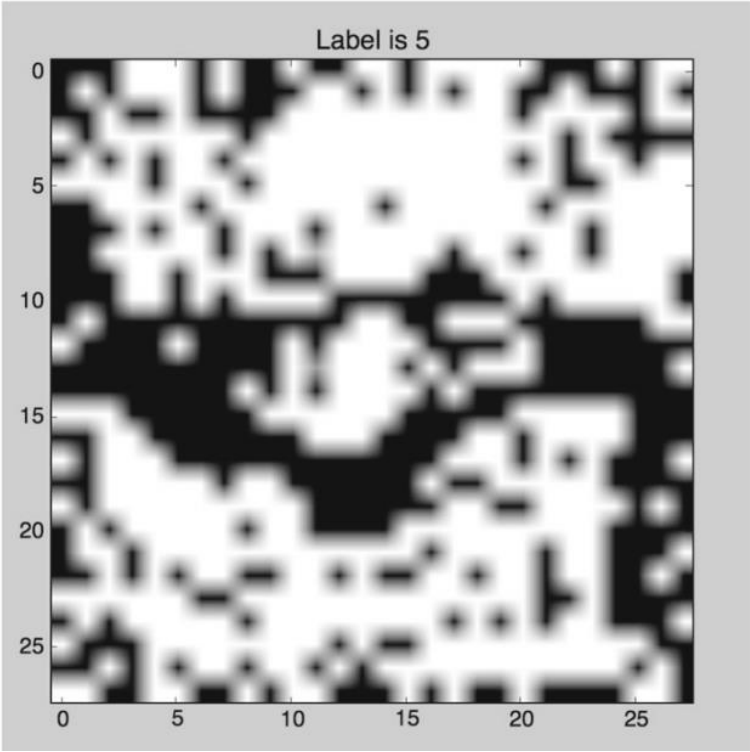
Application

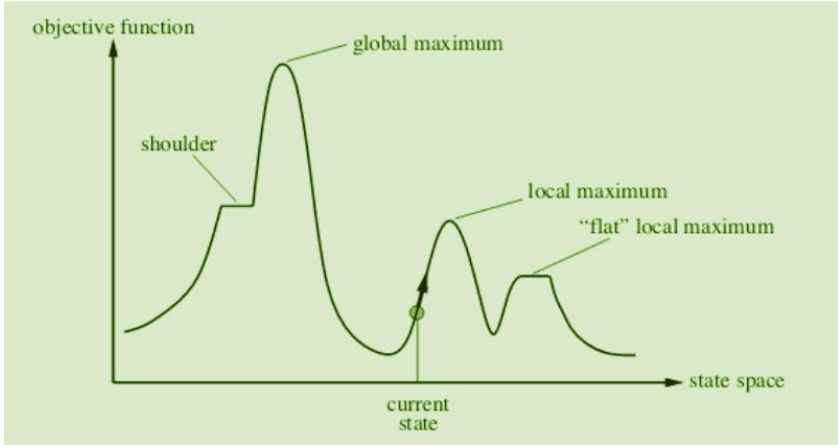
Feature Visualization

Used MILP model (defined in previous slides) to find input examples that maximize the activation x_j^k of each unit $neuron(j,k)$ such that its output is the expected label.

In most literature max activations is performed using greedy ascent, it can be trapped by local optimal solutions or may not even reach any meaningful solution.







Feature Engineering | SpringerLink

Application

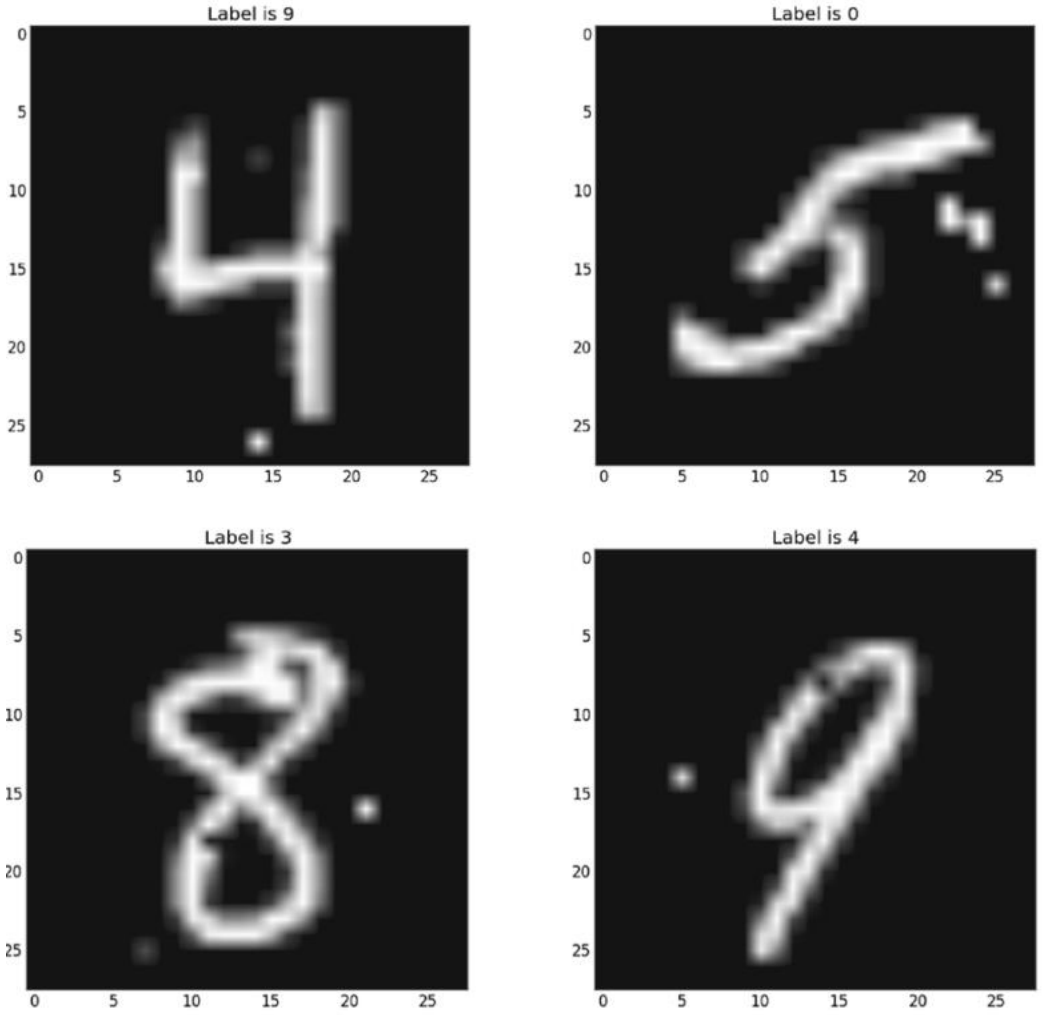
Adversarial Examples

Given original input $\tilde{x}^0$ which DNN correctly classify $\tilde{l}$.

Find x^0 (similar to $\tilde{x}^0$) which is classified as l ($\neq \tilde{l}$).

For figure on the right:

$$l = (\tilde{l} + 5) mod\ 10$$



Application

Adversarial Examples

Given original input $\tilde{x}^0$ which DNN correctly classify $\tilde{l}$.

Find x^0 (similar to $\tilde{x}^0$) which is classified as l ($\neq \tilde{l}$).

For figure on the right:

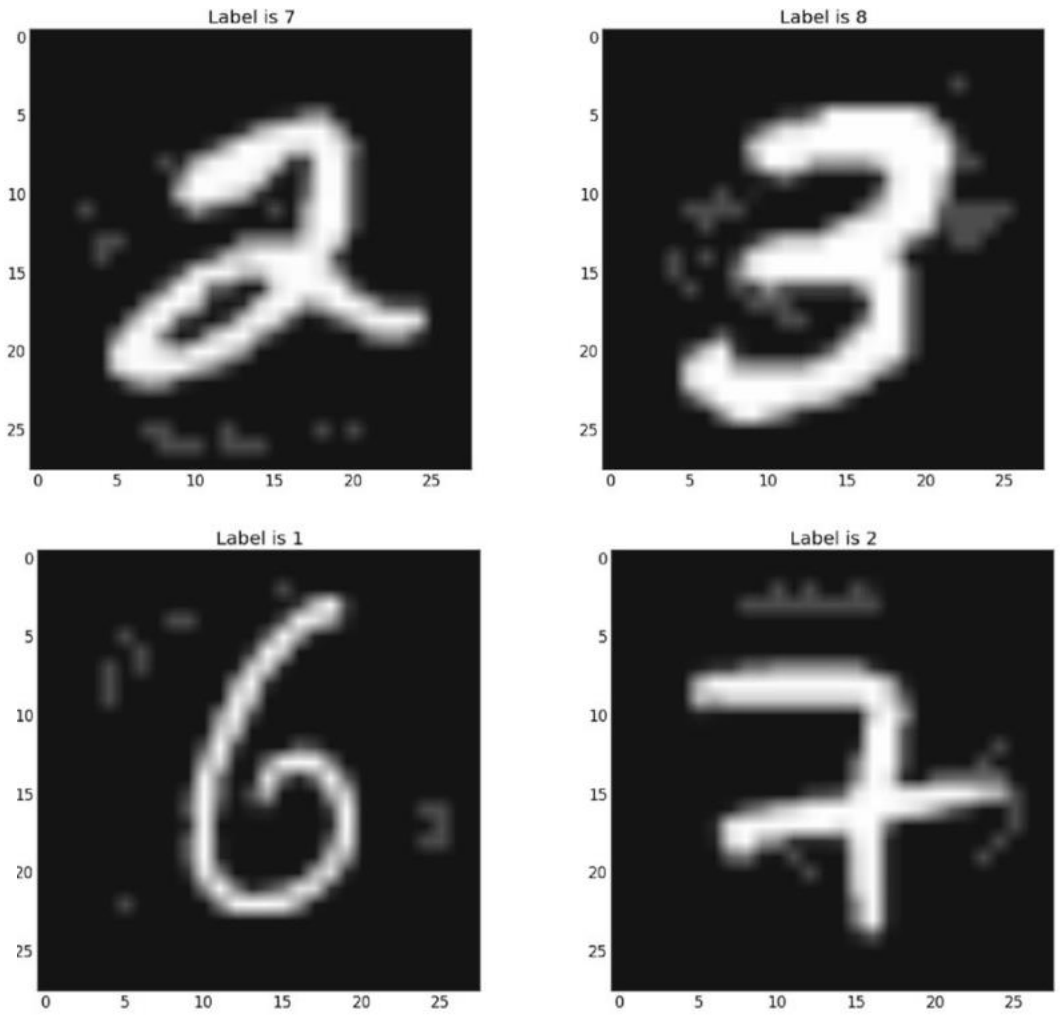
$$l = (\tilde{l} + 5) mod\ 10 \qquad (8)$$

$$\mathrm{x}_{d+1}^{K} \geq 1.2\ \mathrm{x}_{j+1}^{K}, \qquad j \in \{0, \ldots, 9\}\{d\} \qquad (9)$$

To keep new input x^0 close to the original $\tilde{x}^0$, we can reduce L1 distance between them with an additional objective function.

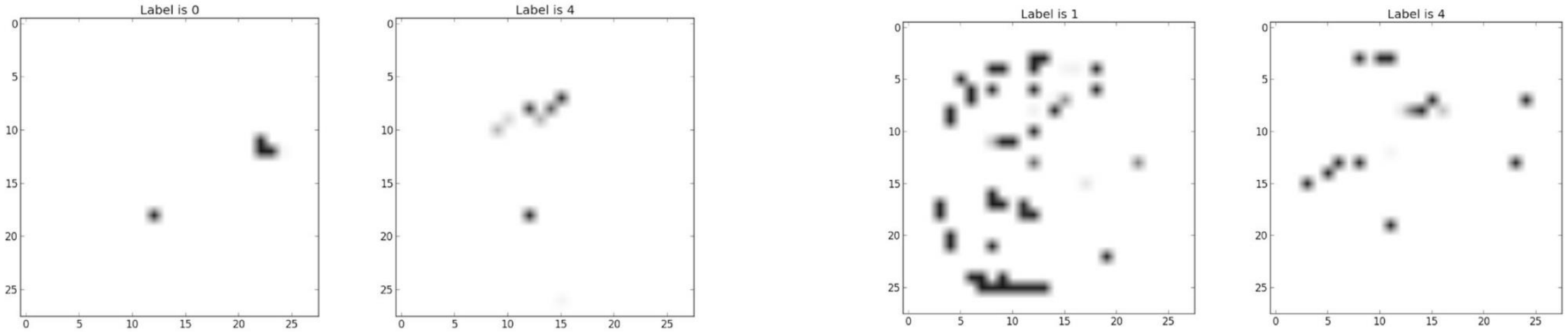
$$-d_j \leq \mathrm{x}_j^o - \tilde{x}_j^0 \leq d_j, \qquad \sum_{j=1}^{n_0} d_j \qquad (10)$$

MILP solvers are flexible enough that all of these equations can be added to the original linear inequalities (5-7).



Application

Adversarial Examples



Pixel changes (absolute value) that suffice to trick the DNN: The right two figures correspond to the model where pixels can change arbitrarily, while two on the left refer to the case where each pixel cannot change by more than 0.2

Results and Analysis

Considering different DNN models

- DNN1: 8+8+8 internal units in 3 hidden layers.
- DNN2: 8+8+8+8+8+8 internal units in 6 hidden layers
- DNN3: 20+10+8+8 internal units in 4 hidden layers
- DNN4: 20+10+8+8+8 internal units in 5 hidden layers
- DNN5: 20+20+10+10+10 internal units in 5 hidden layer

- **Basic model**: best weights/biases selected for each DNN
- **Improved Model**: Preprocessing is applied to the basic model to tightened the bounds for x_j^k, s_j^k (eq 5) .
- The preprocessing needs to be done only once prior. This time is not included in the table.
- **Task**: Solve/find 100 instances of adversarial example.

	Basic model				Improved model			
	%solved	%gap	Nodes	Time (s)	%solved	%gap	Nodes	Time (s)
DNN1	100	0.0	1,903	1.0	100	0.0	552	0.6
DNN2	97	0.2	77,878	48.2	100	0.0	11,851	7.5
DNN3	64	11.6	228,632	158.5	100	0.0	20,309	12.1
DNN4	24	38.1	282,694	263.0	98	0.7	68,563	43.9
DNN5	7	71.8	193,725	290.9	67	11.4	76,714	171.1

- **Nodes** - Avg. no. of branching nodes
- **Time(s)** - Computing time
- **%gap** - percentage gap between the best ub and the best lb
- **%solved** - percentage of solved instances

Results and Analysis

Considering different DNN models

Performance of the improved model with a time limit of 300 seconds, with exact vs weaker bounds (the latter being computed with a time limit of 1 sec. for each bound computation).

	Improved model									
	Exact bounds					Weaker bounds				
	t.pre.	%sol.	%gap	Nodes	Time (s)	t.pre.	%sol.	%gap	Nodes	Time (s)
DNN4	1,112.1	98	0.7	68,563	43.9	69.4	98	0.4	80,180	45.5
DNN5	4,913.1	67	11.4	76,714	171.1	72.6	57	16.9	84,328	185.0

Why MILP

Why MILP: Versatility, Precision, Optimization.

Applications of MILP:

- Supply chain optimization
- production planning, scheduling
- Finance
- energy management

Challenges in MILP:

- Complexity: Can be computationally intensive, especially as the number of variables and constraints increases.
- Solving Techniques: Techniques like branch-and-bound, cutting planes, vs greedy.

Conclusion and Future Directions

Conclusion:

Model not suitable for training, as the problem will become bilinear (learning both weights/biases and inputs), but good for optimization task like feature visualization and adversarial example generation to learn about the DNN models.

For small DNNs, our model can be solved to proven optimality in a matter of seconds on a standard notebook.

However, for larger (realistic) DNNs the computing time can become too large. DNNs of size (30, 20, 10, 10, 10, 8, 8, 8) or (50, 50, 50, 20, 8, 8) lead to computing times of one hour or more. Resort to heuristics, possibly based on a restricted version of the model itself.

Future work:

Reduction of the computational effort involved in the exact solution of the model

Find new heuristic methods for building adversarial examples for large DNNs

Find new application of MILP in DNN

Summary

Summary

- DNN : Network, Activation Function(ReLU)
- MILP: Branch-n-Bound, Bound Tightening
- Define DNN (ReLU) as Linear problem
- Feature Visualization
- Adversarial Example generation
- Improved vs Weak Model
- Why MILPs, advantages and challenges.
- Conclusion and Future Work.

Thank you for the attention!

Next: Questions

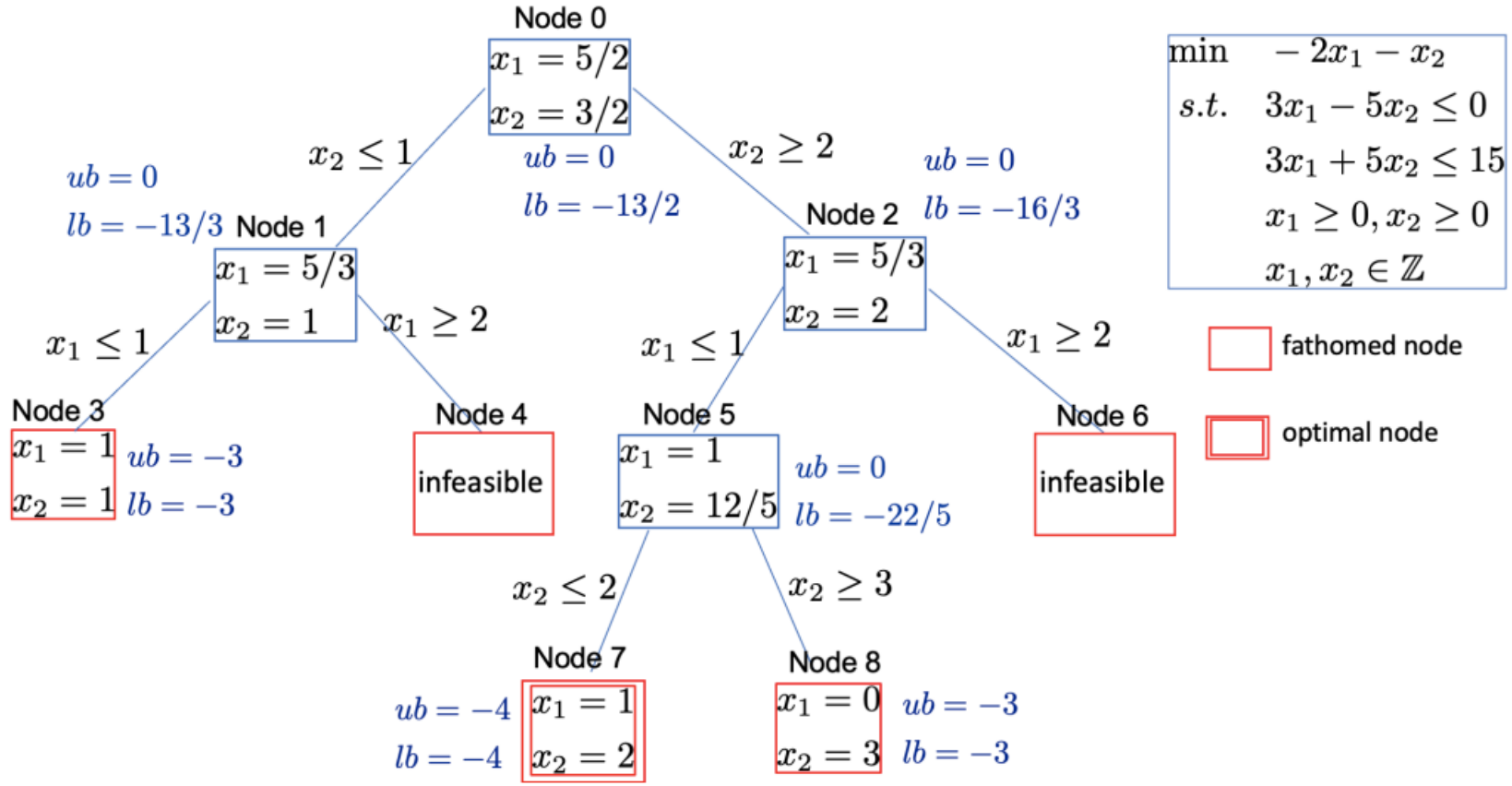


Fig. 2. Adopting B&B to solve a minimization integer linear programming.

2111.06257.pdf (arxiv.org)