GAMS
MODEL – SOLVE – DEPLOY

GAMSPy

Embedding neural networks into optimization models

Michael Bussieck, Frederik Fiand, Hamdi Burak Usul

OR 2025

Bielefeld

Outline

- What is GAMSPy? 🧩
- GAMSPy ML features 🤖
- Surrogate model example 🧪

What is GAMSPy?

GAMS
MODEL – SOLVE – DEPLOY

trnsport.py

```python
m = Container()

i = Set(m, "i")
j = Set(m, "j")
a = Parameter(m, "a", i)
b = Parameter(m, "b", j)
c = Parameter(m, "c", [i, j])
x = Variable(m, "x", "positive", [i, j])
supply = Equation(m, "supply", domain=i)
demand = Equation(m, "demand", domain=j)

supply[i] = Sum(j, x[i, j]) <= a[i]
demand[j] = Sum(i, x[i, j]) >= b[j]

```

Abstract Model

Declared/defined over sets

Like “writing on paper”

Compact

Logically consistent (no domain violations, uncontrolled sets)

Completely abstract (no data)

Leverages operator overloading

What is GAMSPy?

GAMS
MODEL – SOLVE – DEPLOY

trnsport.py

```python
transport = Model(
    m,
    name="transport",
    equations=m.getEquations(),
    problem="LP",
    sense=Sense.MIN,
    objective=Sum((i, j), c[i, j] * x[i, j]),
)

transport.solve(solver="highs",...)
```

choose from 30+ solvers

Abstract Model

Declared/defined over sets

Like “writing on paper”

Compact

Logically consistent (no domain violations, uncontrolled sets)

Completely abstract (no data)

Leverages operator overloading

Ease of installation

GAMS
MODEL – SOLVE – DEPLOY

```
pip install gamspy
```

-  Stable version released: **https://pypi.org/project/gamspy/** 
- Documentation: **https://gamspy.readthedocs.io/en/latest/**
- Model library (100+ models): **https://github.com/GAMS-dev/gamspy-examples**
- Google Colab Notebooks: **https://gamspy.readthedocs.io/en/latest/user/examples.html**

Licensing

GAMS
MODEL – SOLVE – DEPLOY

- Free academic license
  - Valid for 12 months, completely renewable
  - No model size restrictions
  - Node – limited to 2 – no internet connection
  - Network – limited to 2 (e.g. Docker, Google Colab, etc.) – requires internet
  - Full access to all available open-source solvers
  - Full access to CPLEX, XPRESS/Global, COPT, CONOPT, MOSEK
  - Full access to Gurobi through the GAMS Gurobi Link

🔗 `academic.gams.com`

Why GAMSPy needs ML features?

GAMS
MODEL – SOLVE – DEPLOY

Python is the language for Machine Learning

Combining ML into your optimization problems becomes much easier!

New opportunities!

Similar packages

GAMS
MODEL – SOLVE – DEPLOY

- MathOptAI.jl for JuMP
- gurobi-machinelearning for gurobipy
- OMLT for Pyomo
- PySCIPOpt-ML for PySCIPOpt

💡 **`gamspy.formulations`** is the only package *simultaneously* offering

- 🌀 seamless Python ML and data library interaction
- 🚀 high performance GAMS execution system
- 🤓 many built-in but also easily extensible formulations
- 💪 Solver independence

Comparison paper from Dowson et al. 🔗 **`arxiv.org/abs/2507.03159`**

Machine Learning in GAMSPy

GAMS
MODEL – SOLVE – DEPLOY

Machine Learning in GAMSPy	
Matrix Interface	Formulations

- Indices with no physical meaning
- Matrix operations

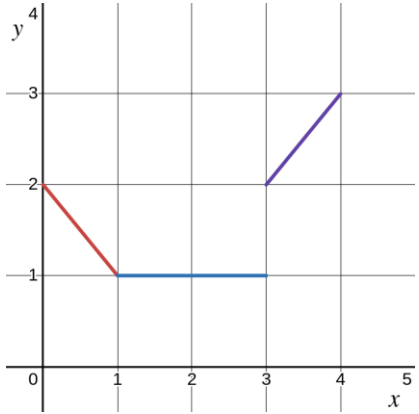
multiply.py

```python
from gamspy.math import dim

...
w = gp.Parameter(m, domain=dim([10, 20]))
x = gp.Variable(m, domain=dim([40, 10]))

some_eq[...] = y == x @ w
```

- Popular programming constructs
- Activation functions



Matrix Interface

GAMS
MODEL – SOLVE – DEPLOY

old.py

```
1 i = gp.Set(m, name="i", records=range(10))
2 j = gp.Set(m, name="j", records=range(20))
3
4 w = gp.Parameter(m, name="w", domain=[i, j])
```

Old way

When you needed to define a matrix,

Declare sets manually

Matrix Interface

GAMS
MODEL – SOLVE – DEPLOY

new.py

```python
from gamspy.math import dim

w = gp.Parameter(
    m,
    name="w",
    domain=dim([10, 20]),
)
```

New way

When you need a matrix,

Use dim to provide dimensions.

Matrix Interface

GAMS
MODEL – SOLVE – DEPLOY

multiply.py

```python
from gamspy.math import dim

...
w = gp.Parameter(m, domain=dim([10, 20]))
x = gp.Variable(m, domain=dim([40, 10]))

some_eq[...] = y == x @ w
```

New way

Use @ for matrix multiplication.

Works with matrices, vectors and batched matrices!

You can stack as you wish!

Matrix Interface

GAMS
MODEL – SOLVE – DEPLOY

multiply.py

```python
from gamspy.math import dim

...
w = gp.Parameter(m, domain=dim([15, 20])) # <<
x = gp.Variable(m, domain=dim([40, 10]))

some_eq[...] = y == x @ w
```

New way

Tracks the dimensions

```
    452 elif lr == (2, 2):
    453     # Matrix multiplication
    454     if not utils.set_base_eq(left.domain[1], right.domain[0]):
--> 455         raise ValidationError(dim_no_match_err)
    457     left_domain = left.domain[0]
    458     right_domain = right.domain[1]

ValidationError: Matrix multiplication dimensions do not match
```

Activation Functions

GAMS
MODEL – SOLVE – DEPLOY

activate.py

```python
from gamspy.math import dim
from gamspy.math import relu_with_binary_var

w = gp.Parameter(m, domain=dim([10, 20]))
x = gp.Variable(m, domain=dim([40, 10]))

some_eq[...] = y == x @ w
some_eq2[...] = y2 == relu_with_binary_var(y)
```

Commonly used activation functions!

ReLU: reLU(x) = max(0,x)

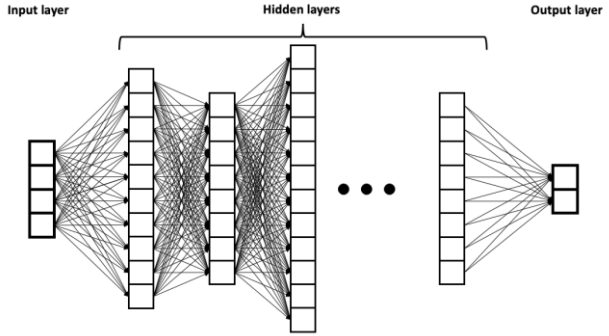
- With binary variables
- Complementarity conditions
- SOS1 variables

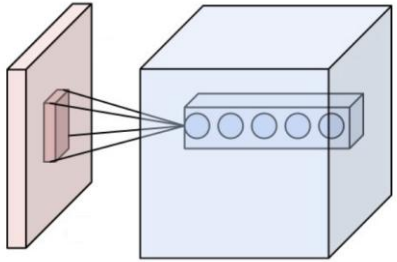
TanH

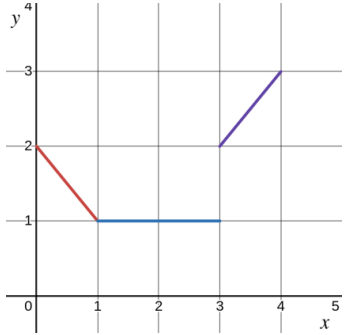
Softmax

Log Softmax

Sigmoid







Formulations

Some constructs in mathematical modeling are more commonly used than others. Provide users with a way to easily construct those!

Machine Learning related Formulations

Activation functions

ReLU

- With binary variables
- Complementarity conditions
- SOS1 variables

TanH

Softmax

Log Softmax

Sigmoid

Formulations

- Linear layer (a.k.a. Dense layer)
- Conv2D
- Min/Max/Avg Pooling
- Flattening

💡 Can be easily extended by the user

Details in the docs:
🔗 `gamspy.readthedocs.io/en/latest/reference/gamspy.formulations.html`

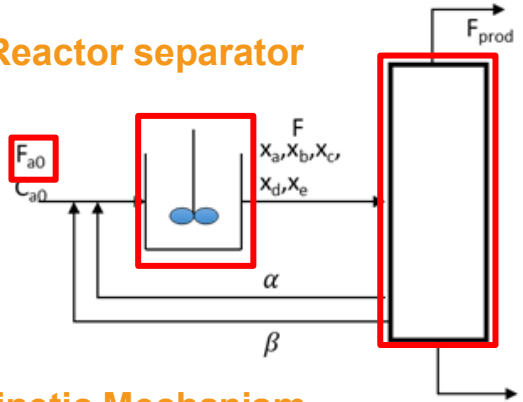
Surrogate Model Example

From chemical engineering

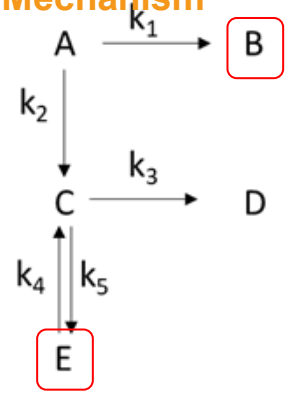
Example Problem

GAMS
MODEL – SOLVE – DEPLOY

Reactor separator



Kinetic Mechanism



Bhosekar & Ierapetritou (2020)

Mass balance equations

$$F_{a0} - x_a F(1-\alpha) - VC_{a0}(k_1 + k_2)x_a = 0$$

$$-Fx_b(1-\alpha) + VC_{a0}k_1 x_a = 0$$

$$-Fx_c + VC_{a0}(k_2 x_a - (k_3 + k_4)x_c + k_5 x_e) = 0$$

$$-Fx_d(1-\beta) + VC_{a0}k_3 x_c = 0$$

$$-Fx_e(1-\beta) + VC_{a0}(k_4 x_c - k_5 x_d) = 0$$

$$x_a + x_b + x_c + x_d + x_e - 1 = 0$$

Given

Demands for products B and E

Problem

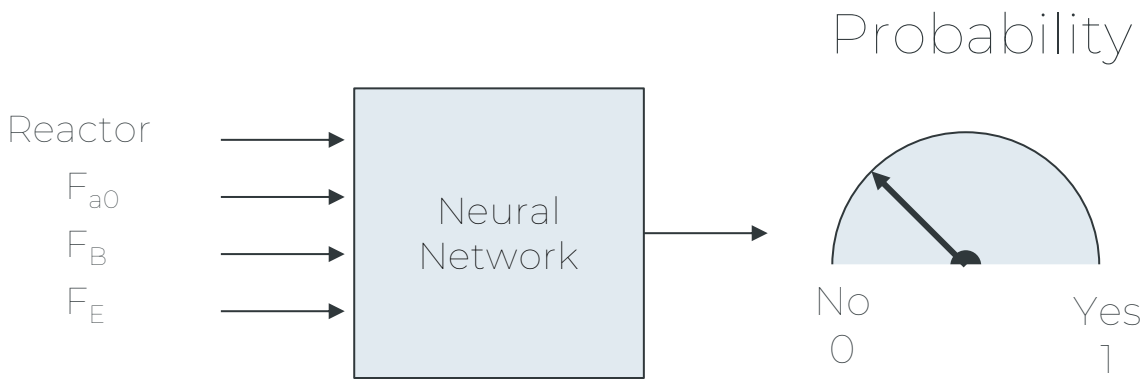
Select reactor and separator modules that minimize cost and satisfy variability in the demand

Historical Data

Reactor Size	F_{a0}	F_B	F_E	Possible
$5m^3$	30	0	11	Yes (1)
$5m^3$	30	18	0	Yes (1)
$5m^3$	30	18	14	No (0)

36.000 rows calculated via simulation

Learn Feasibility



- Split dataset into training and validation
- 28.800 training samples
- 7.200 validation samples
- 99% prediction accuracy on validation set

MIP Model

GAMS
MODEL – SOLVE – DEPLOY

total costs

$$\min_x \quad \sum_{r \in R} Cost_r \cdot y_r + \sum_{s \in S} Cost_s \cdot y_s + Cost_a \cdot F_a$$

s.t.

$$\sum_{r \in R} y_r = 1$$

reactor choice

$$\sum_{s \in S} y_s = 1$$

selector choice

$$Cap = \sum_{r \in R} Cap_r \cdot y_r$$

available reactor capacity

feasibility check via NN

$$NN(Cap, F_a, F_B, F_E) \geq 0.99$$

threshold for NN prediction (0/1 in data)

$$F_a \leq max_s + (1 - y_s) \cdot M \quad \forall s \in S$$

$$F_a + (1 - y_s) \cdot M \geq min_s \quad \forall s \in S$$

flow in operational limits of separator

variable domains

$$F_a \in \mathbb{R}_+$$

$$y_r \in \{0, 1\} \quad \forall r \in R$$

$$y_s \in \{0, 1\} \quad \forall s \in S$$

MIP Model

GAMS
MODEL – SOLVE – DEPLOY

$$\min_x \quad \sum_{r \in R} Cost_r \cdot y_r + \sum_{s \in S} Cost_s \cdot y_s + Cost_a \cdot F_a$$

$$\text{s.t.} \quad \sum_{r \in R} y_r = 1$$

$$\sum_{s \in S} y_s = 1$$

$$Cap = \sum_{r \in R} Cap_r \cdot y_r$$

$$\boxed{NN(Cap, F_a, F_B, F_E) \geq 0.99}$$

Formulations!

$$F_a \leq max_s + (1 - y_s) \cdot M \quad \forall s \in S$$

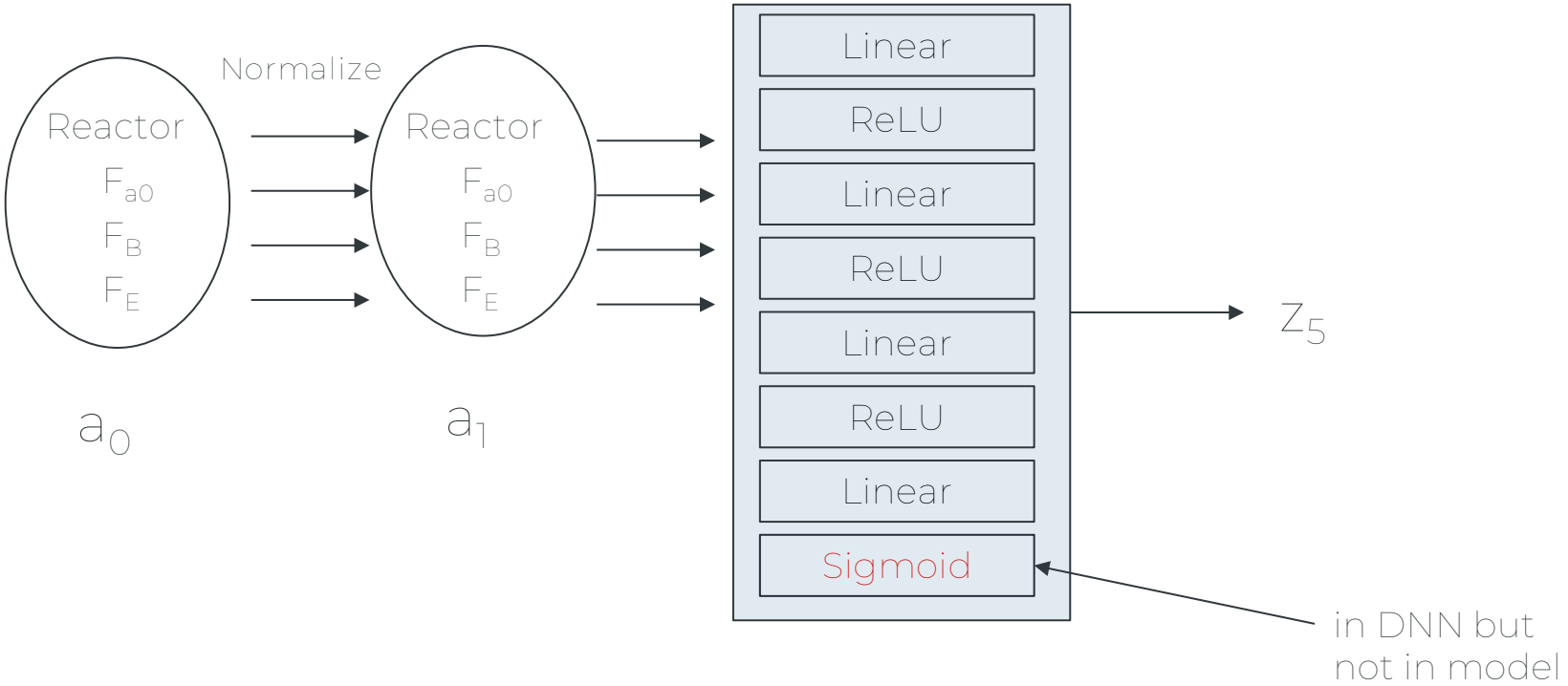
$$F_a + (1 - y_s) \cdot M \geq min_s \quad \forall s \in S$$

$$F_a \in \mathbb{R}_+$$

$$y_r \in \{0, 1\} \quad \forall r \in R$$

$$y_s \in \{0, 1\} \quad \forall s \in S$$

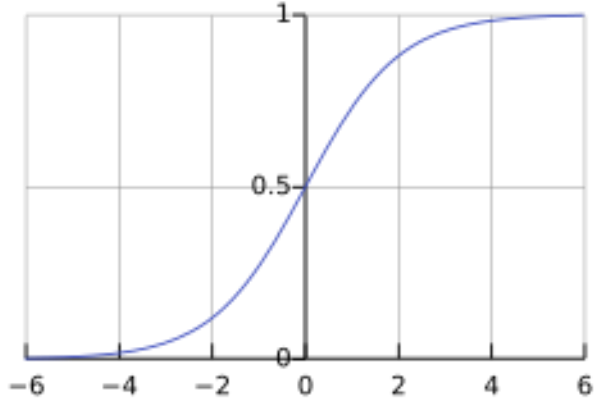
Neural Network architecture



Full source code: 🔗 `https://github.com/GAMS-dev/surrogate-model`

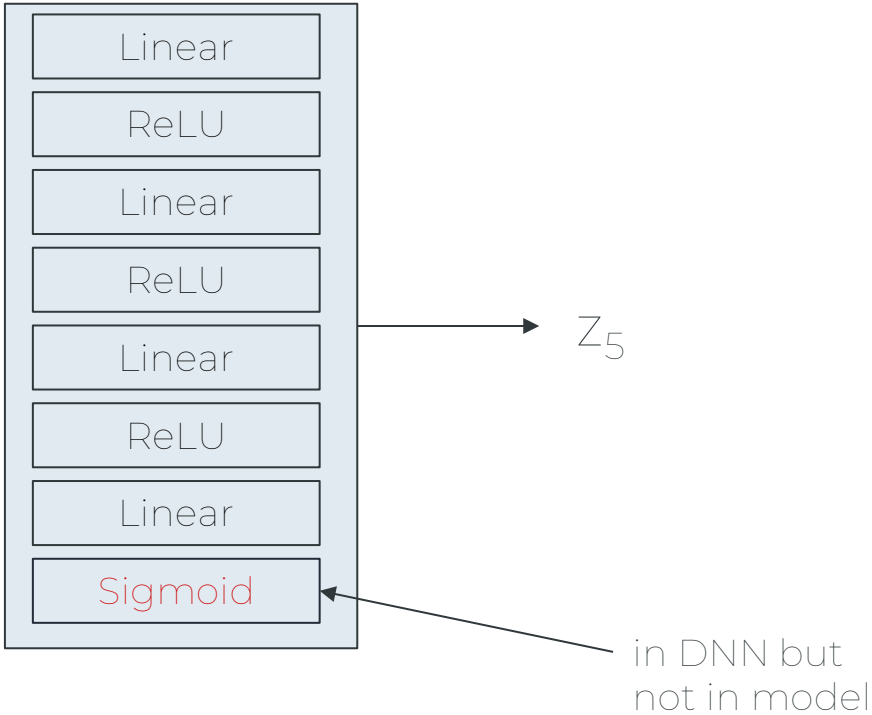
Neural Network architecture

GAMS
MODEL – SOLVE – DEPLOY



$$Sigmoid(x) \geq 0.99 \Rightarrow x \geq 4.59511985013459$$

→ Linear Model



Full source code: 🔗 `https://github.com/GAMS-dev/surrogate-model`

Neural Network plugged into model

GAMS
MODEL – SOLVE – DEPLOY

model.py

```python
import gamspy as gp
from torch import nn
...
model = nn.Sequential(
        nn.Linear(4, 10),
        nn.ReLU(),
        nn.Linear(10, 10),
        nn.ReLU(),
        nn.Linear(10, 15),
        nn.ReLU(),
        nn.Linear(15, 1),
        nn.Sigmoid()
    )
... # train network on simulation/historical data
a0 = gp.Variable(m, name="a0", domain=gp.math.dim([4]))  # input to neural network
a1 = gp.Variable(m, name="a1", domain=gp.math.dim([4]))  # normalized
...
normalize_input = gp.Equation(m, name="normalize_input", domain=a0.domain)
normalize_input[...] = a1 == (a0 - x_mean_par) / x_std_par

drop_sigmoid_model = nn.Sequential(*list(model.children())[:-1])
seq_formulation = gp.formulations.TorchSequential(m, drop_sigmoid_model)
z5, _ = seq_formulation(a1)
check_feasibility = gp.Equation(m, name="check_feasibility")
check_feasibility[...] = z5[0] ≥ 4.59511985013459  # 0.99 probability
```

Embedding Neural Network

Define input `a0`

Normalize if needed

Set bounds, suggested

Forward propagate (drop nonlinear sigmoid)

NN output `z5` used in feasibility constraint of MIP

Final non-linear activation missing: `σ(4.59512) ≈ 0.99`

Summary & Outlook

GAMS
MODEL – SOLVE – DEPLOY

- Easily embedding NN into GAMSPy models allows…
  - *Surrogate models*
  - Model verification
  - Adversarial input generation
  - …
- Also supported: Decision & regression trees and random forest
- Get started at `gamspy.readthedocs.io` and run

```
pip install gamspy
```

Thank you for your attention!

Contact us: support@gams.com