Integrating Machine Learning with GAMSPy

H. Burak Usul

Who I am?

- H. Burak Usul
- Software Engineer @ GAMS
- https://github.com/hbusul/
- https://www.linkedin.com/in/hburaku/
- busul@gams.com

Agenda

1. GAMSPy very brief intro
2. Our Goal
3. Pathways
4. Matrix API
5. Adverserial Image Generation Example
6. Conclusion

GAMSPy very brief intro

Free for Academics!

GAMSPy is now officially available for academics...

FOR FREE

Install GAMSPy today and get started:

Fully compatible with:

GAMSPy

- Indexed-algebra style mathematical modeling in Python
- Eco-system of Python and power of GAMS execution system

Example GAMSPy code

Let's write down the following expression:

$$\sum_j a_{ij} \times x_{ij} \geq b_i \quad \forall i$$

```python
import gamspy as gp

m = gp.Container()
i = gp.Set(m)
j = gp.Set(m)
a = gp.Parameter(m, domain=[i, j])
b = gp.Parameter(m, domain=[i])
x = gp.Variable(m, domain=[i, j])
e = gp.Equation(m, domain=[i])
e[i] = gp.Sum(j, a[i, j] * x[i, j]) >= b[i]
```

GAMSPy

- Indexed-algebra style mathematical modeling in Python
- Eco-system of Python and power of GAMS execution system
- Enabling applications that were difficult to implement with GAMS

Our Goal

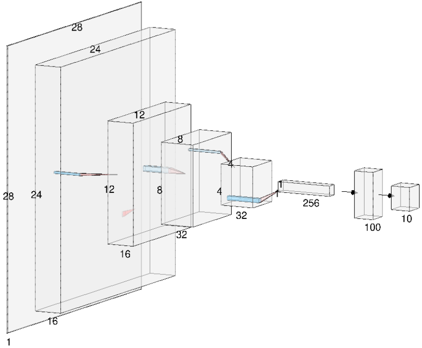
What do want to achieve?

Using GAMSPy:

- Embed
  - Trained neural networks in optimization problems
  - Classical machine learning methods in optimization problems
- Describe a neural network using GAMSPy
  - To infer from it
  - To train it
  - To sparsify it
  - ...

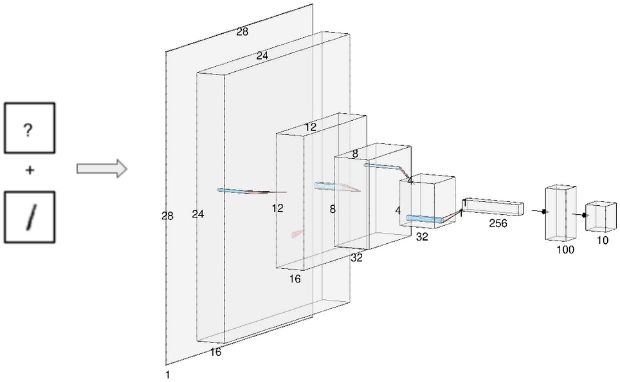
Optical Character Recognition

- Input is 28x28 pixel grayscale digits from MNIST dataset
- Output is the label of the digit



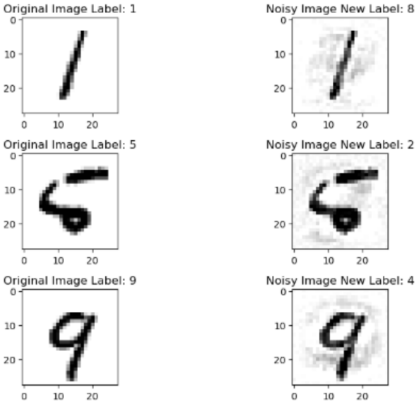
Adverserial Example Generation

- Perturb the input image minimally so that guessed label changes



Adverserial Example Generation

- Perturb the input image minimally so that guessed label changes

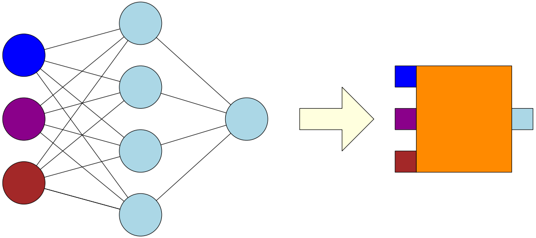


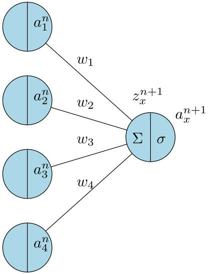
Pathways

How we enable integration?

We implement complex blocks, abstracting away the complexity for users.

We provide fundamental building blocks that users can combine in their preferred ways





$$z_x^{n+1} = a_1^n.w_1 + a_2^n.w_2 + a_3^n.w_3 + a_4^n.w_4$$
$$a_x^{n+1} = \sigma(z_x^{n+1})$$
$$z_x^{n+1} = A^n @ W$$

How we enable integration?

We implement complex blocks, abstracting away the complexity for users.

- Implement OMLT like functionality

https://github.com/cog-imperial/OMLT

We provide fundamental building blocks that users can combine in their preferred ways

- Implement matrix multiplication
- Implement activation functions
- Implement commonly used functions

How we enable integration?

GAMS
MODEL – SOLVE – DEPLOY

We implement complex blocks, abstracting away the complexity for users.

- Implement OMLT interface ~~like functionality~~ with GAMSPy

Coming soon!

We provide fundamental building blocks that users can combine in their preferred ways

- Implement matrix multiplication
- Implement activation functions
- Implement commonly used functions

We started with this

Matrix API

Creating matrices

GAMS
MODEL – SOLVE – DEPLOY

```python
import gamspy as gp
import numpy as np
from gamspy.math import dim

w1_data = np.random.rand(784, 20)
m = gp.Container()

w1 = gp.Parameter(m, name="w1", domain=dim([784, 20]),
                  records=w1_data)

a1 = gp.Variable(m, name="a1", domain=dim([64, 784]))
```

Introducing matrix multiplication

```
...
w1 = gp.Parameter(m, name="w1", domain=dim([784, 20]),
                  records=w1_data)

a1 = gp.Variable(m, name="a1", domain=dim([64, 784]))

z2 = gp.Variable(m, name="z2", domain=dim([64, 20]))

calc_z2 = gp.Equation(m, name="calc_z2", domain=dim([64, 20]))

calc_z2[...] = z2 == a1 @ w1
```

Allows many use cases

GAMS
MODEL – SOLVE – DEPLOY

```
...
sq_data = np.eye(4) # identity matrix
sq = gp.Parameter(m, name="sq", domain=dim([4, 4]),
                  records=sq_data)

sq2 = gp.Parameter(m, name="sq2", domain=dim([4, 4]))

sq2[...] = sq @ sq @ sq @ sq @ sq

sq2.toDense()
# array([[1., 0., 0., 0.],
#        [0., 1., 0., 0.],
#        [0., 0., 1., 0.],
#        [0., 0., 0., 1.]])
```

Dimension tracking

```python
...
((a1 @ w1) @ (w1.t() @ a1)).domain
# [<Set ‘DenseDim64_1‘ (0x...)>, <Set ‘DenseDim20_1‘ (0x...)>]
```

```
--------------------------------------------------------------------------
ValidationError                           Traceback (most recent call last)
Cell In[16], line 24
     19 a1 = gp.Variable(m , name = "a1" , domain=dim ([64 , 784]))
     22 (a1 @ w1).gamsRepr()
---> 24 ((a1 @ w1) @ (w1.t() @ a1))

File ~/anaconda3/envs/ml/lib/python3.10/site-packages/gamspy/_algebra/operable.py:131, in Operable.__matmul__(self, other)
    128 import gamspy._algebra.operation as operation
    129 from gamspy.math.matrix import _validate_matrix_mult_dims
--> 131 left_domain, right_domain, sum_domain = _validate_matrix_mult_dims(
    132     self, other
    133 )
    134 return operation.Sum(
    135     [sum_domain], self[left_domain] * other[right_domain]
    136 )

File ~/anaconda3/envs/ml/lib/python3.10/site-packages/gamspy/math/matrix.py:455, in _validate_matrix_mult_dims(left, right)
    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

```python
...
from gamspy.math import tanh, sigmoid
from gamspy.math import relu_with_binary_var
from gamspy.math import relu_with_complementarity_var
from gamspy.math import relu_with_sos1_var

a2 = gp.Variable(m, name="a2", domain=dim([64, 20]))
calc_a2 = gp.Equation(m, name="calc_a2", domain=dim([64, 20]))
calc_a2[...] = a2 == tanh(z2)

# ReLU is bit special
a3, constraints = relu_with_binary_var(z2) # OMLT
a4, constraints = relu_with_sos1_var(z2) # (Turner et al., 2024)
a5, constraints = relu_with_complementarity_var(z2) # OMLT
```

Turner, Mark, et al. "PySCIPOpt-ML: Embedding trained machine learning models into mixed-integer programs." arXiv preprint arXiv:2312.08074 (2023).
Ceccon, Francesco, et al. "OMLT: Optimization & machine learning toolkit." Journal of Machine Learning Research 23.349 (2022): 1-8.

Activation functions - cont.

```
...
from gamspy.math import log_softmax
from gamspy.math import softmax

a2 = gp.Variable(m, name="a2", domain=dim([64, 20]))
calc_a2 = gp.Equation(m, name="calc_a2", domain=dim([64, 20]))
calc_a2[...] = a2 == tanh(z2)

a6, constraints = log_softmax(z2)
a7, constraints = softmax(z2)
```

You can even use these if you want to train a neural network in GAMSPy!

Vector norms

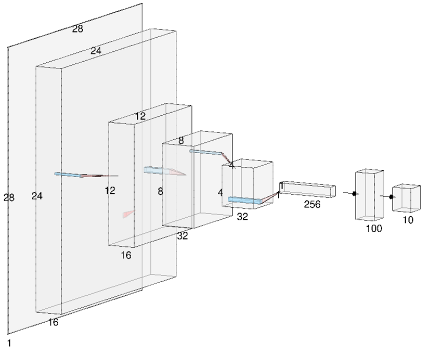
GAMS
MODEL – SOLVE – DEPLOY

```python
import gamspy as gp
import numpy as np
from gamspy.math import vector_norm, dim

m = gp.Container()

vec = gp.Parameter(m, "vec", domain=dim([2]),
                   records=np.array([3, 4]))
vlen = gp.Parameter(m, "vlen")
vlen[...] = gp.math.vector_norm(vec)
vlen.records
#       value
# 0       5.0

vlen.toDense()
#       value
# 0       5.0
```

Adverserial Image Generation Example

Step 1

- Train a simple Fully Convolutional Network using PyTorch
- Train it using MNIST dataset
- Get the weights into numpy arrays



Step 1

```python
class SimpleModel(nn.Module):
    def __init__(self):
        ...

    def forward(self, x):
        x = self.conv1(x)
        x = self.activation(x)
        x = self.avg_pool(x)

        x = self.conv2(x)
        x = self.activation(x)
        x = self.avg_pool(x)

        x = self.conv3(x)
        x = x.reshape(x.shape[0], -1)
        x = self.l1(x)
        x = self.activation(x)
        logits = self.l2(x)

        return logits
```

Step 2

Declare Parameters for weights:

```
conv1_weight = network.conv1.weight.detach().numpy()
conv1_bias   = network.conv1.bias.detach().numpy()

conv2_weight = network.conv2.weight.detach().numpy()
conv2_bias   = network.conv2.bias.detach().numpy()

conv3_weight = network.conv3.weight.detach().numpy()
conv3_bias   = network.conv3.bias.detach().numpy()

l1_weight = network.l1.weight.detach().numpy().T
b1_weight = network.l1.bias.detach().numpy()

l2_weight = network.l2.weight.detach().numpy().T
b2_weight = network.l2.bias.detach().numpy()

image_data = single_image.numpy()
image_target = single_target.numpy()
```

Step 3

Declare GAMSPy Conv2d and AvgPool2d counterparts:

```
from gamspy.math import Conv2d, AvgPool2d #TODO include it to next
                                           release

m = gp.Container()
avg_pool = AvgPool2d(m, (2, 2), 2, 0) # near identical signature with
                                       PyTorch

conv1 = Conv2d(m, 1, 16, 5, stride=1, bias=True) # near identical
                                                   signature with PyTorch
conv1.load_weights(conv1_weight, conv1_bias)

conv2 = Conv2d(m, 16, 32, 5, stride=1, bias=True)
conv2.load_weights(conv2_weight, conv2_bias)

conv3 = Conv2d(m, 32, 256, 4, stride=1, bias=True)
conv3.load_weights(conv3_weight, conv3_bias)
```

Step 4

Declare Other Parameters:

```
image = gp.Parameter(m, name="image", domain=dim(image_data.shape),
                     records=image_data)

w1 = gp.Parameter(m, name="w1", domain=dim(l1_weight.shape),
                  records=l1_weight)

b1 = gp.Parameter(m, name="b1", domain=dim(b1_weight.shape),
                  records=b1_weight)

w2 = gp.Parameter(m, name="w2", domain=dim(l2_weight.shape),
                  records=l2_weight)

b2 = gp.Parameter(m, name="b2", domain=dim(b2_weight.shape),
                  records=b2_weight)
```

Step 5

Declare Input to Network:

```
a1 = gp.Variable(m, name="x1", domain=dim([1, 1, 28, 28]))

add_noise_and_normalize = gp.Equation(m, "eq1", domain=dim([1, 1, 28,
                                                            28]))
add_noise_and_normalize[...] = \
    a1 == (image + noise - mean[0]) / std[0]

#ensure bounds
a1.lo[...] =     - mean[0] / std[0]
a1.up[...] = (1 - mean[0]) / std[0]
```

Step 6

More forward propagation:

```
z2, _ = conv1(a1)
a2, _ = relu_with_binary_var(z2)
z3, _ = avg_pool(a2)


z3_2, _ = conv2(z3)
a3, _ = relu_with_binary_var(z3_2)
z4, _ = avg_pool(a3)
z5, _ = conv3(z4)

z5_2 = z5[:, :, "0", "0"]  # (1 x 256 x 1 x 1) -> (1 x 256)
```

Step 7

Define variable/equations for linear layers:

```
z6 = gp.Variable(m, name="z6", domain=dim([1, 100]))
set_z6 = gp.Equation(m, name="set_z6", domain=z6.domain)
set_z6[...] = z6[...] == z5_2 @ w1 + b1

a6, _ = relu_with_binary_var(z6)

z7 = gp.Variable(m, name="z7", domain=dim([1, 10]))
set_z7 = gp.Equation(m, name="set_z7", domain=z7.domain)
set_z7[...] = z7[...] == a6 @ w2 + b2
```

Step 5

Define Problem Specific constraints:

```
# Set epsilon as you wish, higher it is, harder to solve
eps = 0.01

trick_nn = gp.Equation(m, name="false_label", ...)
trick_nn[...] = \
        z7["0", "8"] >= z3["0", "6"] + eps
```

Step 6

Define objective function

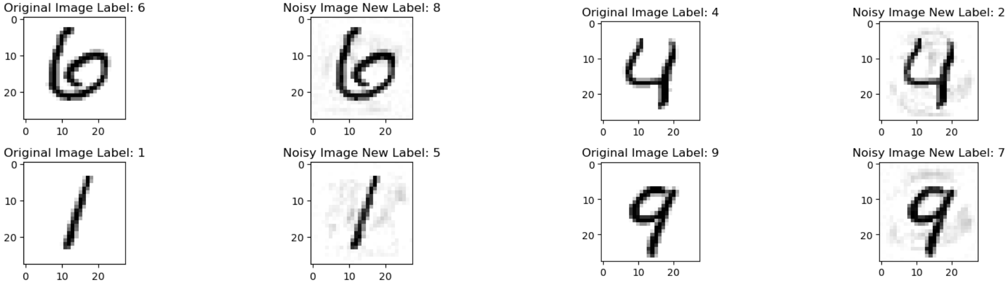
```
from gamspy.math import vector_norm
obj = gp.Variable(m, name="obj")

set_obj = gp.Equation(m, "set_obj")
set_obj[...] = obj == vector_norm(noise) ** 2
```

Step 7

Optimize

```python
model = gp.Model(
    m,
    "min_noise",
    equations=m.getEquations(),
    objective=obj,
    sense="min",
    problem="MIQCP"
)

# Pick the solver you want from ~20 MIQCP solvers
# that we support
model.solve(output=sys.stdout, solver="cplex")
model.solve(output=sys.stdout, solver="gurobi")
```

Some examples



Conclusion

Future work

- 3-4 months implementing backend for OMLT
- Implement more Computer Vision Formulations
  - MaxPool2d
  - ...
- Implement Recurrent Formulations
  - RNNs
  - LSTMs
  - GRUs
  - Transformers
  - ...
- Investigate Reduced-Space Formulations

Upcoming talks:

- GAMSPy - Where Convenience of Python Meets GAMS' Performance
  Today 11:30am - 1:00pm @ Theresianum 2605
- The Lifecycle of OR Solutions: From Rapid Prototypes to Market Deployment
  Todat 2:00pm - 3:30pm @ Theresianum 2605