1. Mistakes and incompleteness of the formulation:
2. **Hidden layer(Layer1) incompleteness:** Each hidden node receives only a single input variable, without proper weighted summation.
3. **Biases mistake:** All the hidden nodes incorrectly share the same bias(), but actually every hidden node has its independent bias.
4. **Symbol mistake:** In the objective function, The output is only subtracted from only one of the hidden layer outputs while being added to the other outputs of hidden layer and the bias of output layer.
5. Mathematical programming problem without using integer variables：

Mixed integer programming problem：

1. Here the MIQP problem becomes:

When a solution time = 150s is set, using COPT solver, we can not obtain the global solution.

And the incumbent solution is

；；

；；；

；；；

；；；

;;

;.

The incumbent objective value is 0.000311309901887319.



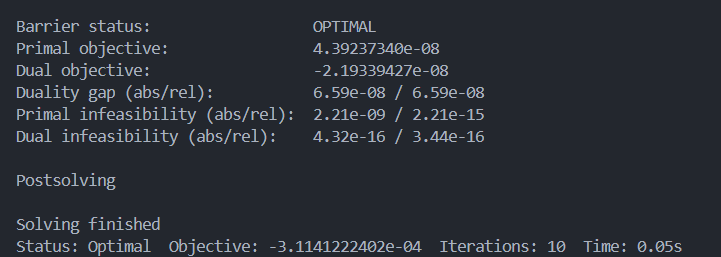
Lower bound is -0.00030149881473562345.



Code file: model\_fixed\_w1&b1-Pyomo.py

1. Here the MIQP problem becomes QP problem(After relaxing):

Although we set a 150s time limit, the COPT solver actually finds the global solution in just 0.05 seconds. Compared to MIQP, the QP problem converges more easily and yield a smaller objective value.



And the global solution is

；；

；；；

；；；

；；；

;;

;.

The global best objective value is .



1. Use PyTorch to train this ANN:

loss function: MSELoss

optimizer : Adam

batch size: 64

epochs: 1000

learning rate: 0.001

It uses 7.31s to complete the training process, the best loss is 0.00071.

Compared to formulating the problem as a MIQP and solving it using branch-and-bound in (c), the method based on matrix operations and gradient descent in PyTorch is more suitable for optimizing neural networks parameters. I think that scaling up the dataset will tighten MIQP constraints drastically and cause the number of decision variables to grow explosively, increase the difficulty of model convergence.