Gurobi Machine Learning
Using Trained Machine Learning Predictors in Gurobi
Agenda

Motivating Example

gurobi-machinelearning

Related Improvements in Gurobi 10.0

Performance Evaluation
Motivating Example

• Selling avocados in the US
  • Market is split into 8 regions $r \in R$
  • Total supply $S$
  • Want to decide shipment to each region
  • Maximizing profit:
    • sales – shipping costs – unsold penalty
    • with given
      • prices $p_r$, shipping costs $c_r$, waste penalty $w$
      • demand $d_r$ in each region
  • Demand estimated using a regression model

See webinar by Rahul Swamy and Jerry Yurchisin
Motivating Example: Estimating Demand

• Historical data of avocado sales from Hass Avocado Board (HAB) available on Kaggle and HAB website
• Features correlated to demand: year, peak season, region, price
• Regression gives reasonably good prediction of demand with those:
  • \( d = g(year, season, r, p) \)
• Regression performed with some machine learning package like scikit-learn
  • Linear regression
  • Logistic regression
  • Neural networks
  • Decision trees
  • Gradient boosted trees
  • ...
Motivating Example: Price Optimization

- A more complex problem: optimize the price $p_r$
- To do so, we need to model the relationship

$$d = g(year, season, r, p)$$

in the optimization problem
- $d$ and $p$ become variables for the optimization
- Notebook developed by J. Yurchisin and R. Swamy
In an optimization model we want to formulate $y = g(x)$

- $x$ input variables for the regression
- $g$ prediction function for trained regression model
- $y$ output variables

- $x$ and $y$ are regular decision variables:
  - Can appear in other constraints
  - Can be partially fixed (fixed features)

- $g$ should be trained a priori by a (popular) python framework

Related works:

- Janos (Bergman et al. 2019)
- OptiCL (Maragno et al. 2021)
- ReluMIP (Schweidtmann, Mitsos 2018, 2021)
- OMLT (Ceccon et al. 2022)
- …
Gurobi Machine Learning
Gurobi Machine Learning

• Open source python package:
  • https://github.com/Gurobi/gurobi-machinelearning
  • https://gurobi-machinelearning.readthedocs.io/

• Apache License 2.0
• Initial release 1.0.0 last November
• Version 1.2.0 recently released

• Supported only on a good-will basis, not through usual Gurobi support
  • But we will certainly do our best!
Regression Models Understood

- Linear/Logistic regression
- Decision trees
- Neural network with ReLU activation
- Random Forests
- Gradient Boosting
- Preprocessing:
  - Simple scaling
  - Polynomial features of degree 2
  - Column transformers
- Pipelines to combine them

Keras
- Dense layers
- ReLU layers
- Object Oriented, functional or sequential

PyTorch
- Dense layers
- ReLU layers
- Only torch.nn.Sequential models
Example: Regression Model with sklearn

$R^2$ value in the test set is 0.90, training set is 0.91

$R^2 \in (\infty, 1]$: coefficient of determination
Example: Creating the Variables

\[
m = \text{gp.Model("Avocado_Price_Allocation")}
\]

\[
p = \text{gppd.add_vars(m, data, lb=0.0, ub=2.0)}
\]

\[
d = \text{gppd.add_vars(m, data)}
\]

\[
u = m.addVar()
\]

Variables

- \( p_r \) selling price per unit
- \( d_r \) demand
- \( u \) total unsold products

<table>
<thead>
<tr>
<th>Region</th>
<th>Price</th>
<th>Demand</th>
</tr>
</thead>
<tbody>
<tr>
<td>Midsouth</td>
<td>(&lt;\text{gurobi.Var price[Midsouth]}&gt;)</td>
<td>(&lt;\text{gurobi.Var demand[Midsouth]}&gt;)</td>
</tr>
<tr>
<td>Northeast</td>
<td>(&lt;\text{gurobi.Var price[Northeast]}&gt;)</td>
<td>(&lt;\text{gurobi.Var demand[Northeast]}&gt;)</td>
</tr>
<tr>
<td>SouthCentral</td>
<td>(&lt;\text{gurobi.Var price[SouthCentral]}&gt;)</td>
<td>(&lt;\text{gurobi.Var demand[SouthCentral]}&gt;)</td>
</tr>
<tr>
<td>Southeast</td>
<td>(&lt;\text{gurobi.Var price[Southeast]}&gt;)</td>
<td>(&lt;\text{gurobi.Var demand[Southeast]}&gt;)</td>
</tr>
<tr>
<td>West</td>
<td>(&lt;\text{gurobi.Var price[West]}&gt;)</td>
<td>(&lt;\text{gurobi.Var demand[West]}&gt;)</td>
</tr>
</tbody>
</table>

© 2023 Gurobi Optimization, LLC. Confidential, All Rights Reserved | 11
Example: Objective and Constraints

\[
\text{max } \sum_r (p_r - c_r) d_r - w \cdot u \quad \text{(maximizing revenue)}
\]

\[
s.t.
\]

\[
\sum_r d_r + u = S, \quad \text{(allocate supply)}
\]

\[
d_r = g(\text{year}, \text{season}, r, p_r) \text{ for } r \in R \quad \text{(demand depends on price)}
\]

```python
m.setObjective(((p - c) * d).sum() - w * u, GRB.MAXIMIZE)
m.addConstr(d.sum() + u == S)
add_predictor_constr(m, pipeline, feats, d)
```
Example: Input of Regression Constraints

\[ d_r = g(\text{year}, \text{season}, r, p_r) \text{ for } r \in R \]

feats = pd.DataFrame(
    data={
        "year": 2020,
        "peak": 1,
        "region": regions,
        "price": p
    },
    index=regions)
Example: Adding Regression Constraints

from gurobi_ml import add_predictor_constr
pred_constr = add_predictor_constr(m, pipeline, feats, d)
pred_constr.print_stats()

Model for pipe:
88 variables
24 constraints
Input has shape (8, 4)
Output has shape (8, 1)

Pipeline has 2 steps:

<table>
<thead>
<tr>
<th>Step</th>
<th>Output Shape</th>
<th>Variables</th>
<th>Linear</th>
<th>Quadratic</th>
<th>General</th>
</tr>
</thead>
<tbody>
<tr>
<td>col_trans</td>
<td>(8, 10)</td>
<td>24</td>
<td>16</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>lin_reg</td>
<td>(8, 1)</td>
<td>64</td>
<td>8</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Example: Optimizing

```python
m.Params.NonConvex = 2
m.optimize()

Explored 1 nodes (75 simplex iterations) in 0.04 seconds (0.00 work units)
Thread count was 8 (of 8 available processors)

Solution count 2: 38.7675 36.5918

Optimal solution found (tolerance 1.00e-04)
Best objective 3.876747585682e+01, best bound 3.876937455959e+01, gap 0.0049%
```
Example: Solution

Optimal net revenue: 38.1 million, unsold avocados: 0.34 millions
Comparison of Models for Price Optimization

<table>
<thead>
<tr>
<th>Model</th>
<th>R² test</th>
<th>R² train</th>
<th>train time</th>
<th>optimization time</th>
<th>size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear Regression</td>
<td>0.898</td>
<td>0.909</td>
<td>0.02</td>
<td>0.05</td>
<td>1.0</td>
</tr>
<tr>
<td>Linear Regression polynomial feats</td>
<td>0.918</td>
<td>0.922</td>
<td>0.03</td>
<td>0.06</td>
<td>6.3</td>
</tr>
<tr>
<td>MLP Regression layers=[8]*2</td>
<td>0.941</td>
<td>0.950</td>
<td>1.08</td>
<td>0.97</td>
<td>6.1</td>
</tr>
<tr>
<td>Decision Tree max_leaf_nodes=50</td>
<td>0.921</td>
<td>0.941</td>
<td>0.02</td>
<td>0.02</td>
<td>3.9</td>
</tr>
<tr>
<td>Random Forest n_estimators=10,</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>max_leaf_nodes=100</td>
<td>0.943</td>
<td>0.966</td>
<td>0.04</td>
<td>0.10</td>
<td>66.2</td>
</tr>
<tr>
<td>Gradient Boosting</td>
<td>0.946</td>
<td>0.958</td>
<td>0.15</td>
<td>0.41</td>
<td>84.5</td>
</tr>
</tbody>
</table>

```python
for r in regressions_models:
    pred_constr = add_predictor_constr(m, r, feats, d)
    m.optimize()
    pred_constr.remove()
```

(size is the ratio between the size of the compressed lp files for regression model and linear regression)
Other Examples

• Gurobi Machine Learning package documentation:
  • Surrogate models (Polynomial features + NN)
  • Student Enrollment (Logistic regression)
  • Adversarial learning (Neural networks)

• Extra notebooks:
  • Variants of adversarial using Keras and Pytorch
  • Variants of Student Enrollment with Decision Trees, Gradient Boosted Trees and Random Forests

• References:
  • Bergman et al. 2019
  • Maragno et al. 2021
  • Schweidtmann, Mitsos 2018, 2021
  • Leyffer et al. 2022
Gurobi 10 Enhancements
• New features for models with ML predictor constraints
  • Logistic function as general function constraint
• Performance improvements relevant for models with ML predictor constraints
  • Optimization based bound tightening (OBBT)
  • Neural network detection
• New features for models with ML predictor constraints
  • Logistic function as general function constraint
• Performance improvements relevant for models with ML predictor constraints
  • Optimization based bound tightening (OBBT)
  • Neural network detection
Each neuron $k$ has the following constraints/variables:

\[
y_{\text{mix}} = w^T x_{\text{in}} + d \\
y_{\text{out}} = \max(y_{\text{mix}}, 0)
\]

The $\max$ function is nonlinear and formulated using a binary variable and big-M constraints.
Neural Networks with ReLU

- Each neuron $k$ has the following constraints/variables:
  \[ y_{\text{mix}} = w^T x_{\text{in}} + d \]
  \[ y_{\text{out}} = \max(y_{\text{mix}}, 0) \]
- The $\max$ function is nonlinear and formulated using a binary variable and big-M constraints.
- Tightness of the formulation depends on bounds that can be inferred for $y_{\text{mix}}$
- Optimization based bound tightening known to be essential for adversarial NN
  - e.g., Fischetti, Jo 2018, Weng et.al. 2018
Optimization Based Bound Tightening

- Common technique for MINLP solvers
- Given the LP relaxation of a (non-convex) MI(NL)P
- For each variable $x$
  - Minimize/maximize $x$ value over relaxation
  - Use optimal value as lower/upper bound for $x$
Optimization Based Bound Tightening

- Common technique for MINLP solvers
- Given the LP relaxation of a (non-convex) MI(NL)P
- For each variable $x$
  - Minimize/maximize $x$ value over relaxation
  - Use optimal value as lower/upper bound for $x$
  - Tighten coefficients of relaxation using new bounds
- Enhancements for OBBT (Gleixner et al. 2017)
  - Filter variables
  - Exploit warm starts
  - Use dual solution of OBBT LPs to tighten bounds in the tree
• For non-convex MIQCP:
  • 23% improvement overall
  • 61% improvement on models solved in $\geq 100$ sec.

• For MIP and convex MIQP/MIQCP:
  • 1% improvement on models solved in $\geq 100$ sec.
  • But big improvement on models with ReLU neural networks
ReLU Neural Network MIP Formulation

• Each neuron $k$ has the following constraints/variables:
  
  \[ y_{\text{mix}} = w^T x_{\text{in}} + d \]
  \[ y_{\text{out}} = \max(y_{\text{mix}}, 0) \]

• MIP formulation:
  
  \[ y^+ - y^- = w^T x_{\text{in}} + d \]
  \[ y^+ \leq u^+ \cdot z \]
  \[ y^- \leq u^- \cdot (1 - z) \]
  \[ 0 \leq y^+ \leq u^+ \]
  \[ 0 \leq y^- \leq u^- \]

  • Output of neuron is just $y_{\text{out}} = y^+$

  • Strength of Big-M formulation of indicators depends on bounds $-u^- \leq y_{\text{mix}} \leq u^+$ of $y_{\text{mix}}$
**Bound Propagation**

- Constraints for $y_{l,k} = y_{l,k}^+ - y_{l,k}^-$ of $k$‘th neuron in layer $l$:
  
  $y_{l,k}^+ - y_{l,k}^- = w_{l,k}^T y_{l-1,k}^+ + d_{l,k}$
  
  $y_{l+1,i}^+ - y_{l+1,i}^- = w_{l+1,i,k} y_{l,k}^+ + \sum_{j \neq k} w_{l+1,i,j} y_{l,j}^+ + d_{l+1,i}$ for all $i$

- Constraints to propagate:
  
  $y_{l,k}^+ = y_{l,k}^- + w_{l,k}^T y_{l-1,k}^+ + d_{l,k}$
  
  $y_{l,k}^- = y_{l,k}^+ - w_{l,k}^T y_{l-1,k}^- - d_{l,k}$
  
  $y_{l,k}^+ = \frac{1}{w_{l+1,i,k}}\left( y_{l+1,i}^+ - y_{l+1,i}^- - \sum_{j \neq k} w_{l+1,i,j} y_{l,j}^+ - d_{l+1,i} \right)$ for all $i$

- Tighter bounds for neuron propagate into previous, same and next layer

- OBBT should be applied layer by layer (see Fischetti, Jo 2018)
  
  - How to identify the layers from the constraint structure?
## Constraint Matrix Nonzero Pattern

<table>
<thead>
<tr>
<th></th>
<th>input</th>
<th>layer 1</th>
<th>layer 2</th>
<th>output</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_1$</td>
<td></td>
<td>$y_{2,1}$</td>
<td>$y_{3,1}$</td>
<td>$y_{1,1}$</td>
</tr>
<tr>
<td>$x_2$</td>
<td></td>
<td>$y_{2,2}$</td>
<td>$y_{3,2}$</td>
<td>$y_{1,2}$</td>
</tr>
<tr>
<td>$x_3$</td>
<td></td>
<td>$y_{2,3}$</td>
<td>$y_{3,3}$</td>
<td>$y_{1,3}$</td>
</tr>
<tr>
<td>$y_2$</td>
<td></td>
<td>$y_{2,4}$</td>
<td>$y_{3,4}$</td>
<td>$y_{1,4}$</td>
</tr>
</tbody>
</table>

The diagram on the right visualizes the connections between layers and outputs.
## Constraint Matrix Nonzero Pattern

<table>
<thead>
<tr>
<th>input</th>
<th>layer 1</th>
<th>layer 2</th>
<th>output</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_1^\text{in}$</td>
<td>$y_{2,1}^+$</td>
<td>$y_{3,1}^+$</td>
<td>$y_{3,1}^\text{out}$</td>
</tr>
<tr>
<td>$x_2^\text{in}$</td>
<td>$y_{2,2}^+$</td>
<td>$y_{3,2}^+$</td>
<td>$y_{3,2}^\text{out}$</td>
</tr>
<tr>
<td>$x_3^\text{in}$</td>
<td>$y_{2,3}^+$</td>
<td>$y_{3,3}^+$</td>
<td>$y_{3,3}^\text{out}$</td>
</tr>
<tr>
<td>$y_1$</td>
<td>$y_{2,1}^-$</td>
<td>$y_{3,1}^-$</td>
<td>$y_{3,1}^-\text{out}$</td>
</tr>
<tr>
<td>$y_2$</td>
<td>$y_{2,2}^-$</td>
<td>$y_{3,2}^-$</td>
<td>$y_{3,2}^-\text{out}$</td>
</tr>
<tr>
<td>$y_3$</td>
<td>$y_{2,3}^-$</td>
<td>$y_{3,3}^-$</td>
<td>$y_{3,3}^-\text{out}$</td>
</tr>
</tbody>
</table>

### Diagram

The diagram illustrates the connections between input, layer 1, layer 2, and output nodes. Each node represents a variable, and the connections indicate the nonzero pattern in the constraint matrix.
### Constraint Matrix Nonzero Pattern

<table>
<thead>
<tr>
<th>Input</th>
<th>Layer 1</th>
<th>Layer 2</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_1^{in}$</td>
<td>$y_{1,1}$, $y_{1,2}$, $y_{1,3}$</td>
<td>$y_{3,1}$, $y_{3,2}$, $y_{3,3}$, $y_{3,4}$</td>
<td>$y_{3,1}^{out}$, $y_{3,2}^{out}$</td>
</tr>
<tr>
<td>$x_2^{in}$</td>
<td>$y_{2,1}$, $y_{2,2}$, $y_{2,3}$</td>
<td>$y_{3,1}$, $y_{3,2}$, $y_{3,3}$, $y_{3,4}$</td>
<td>$y_{3,1}^{out}$, $y_{3,2}^{out}$</td>
</tr>
<tr>
<td>$x_3^{in}$</td>
<td>$y_{3,1}$, $y_{3,2}$, $y_{3,3}$</td>
<td>$y_{3,1}$, $y_{3,2}$, $y_{3,3}$, $y_{3,4}$</td>
<td>$y_{3,1}^{out}$, $y_{3,2}^{out}$</td>
</tr>
</tbody>
</table>

- Variables $y_{l,k}^{+}$ within the same layer have almost identical non-zero patterns (same for $x_k^{in}$)
  - This layer's constraints: different pattern (but each variable only in one constraint)
  - Next layer's constraints: identical non-zero pattern, except for $w_{l,i,j} = 0$ (each var in many constraints)

- Consequence:
  - $p(i, j) = \text{supp}(A_{i,j})^T \text{supp}(A_{i,j})/(||\text{supp}(A_{i,j})|| \cdot ||\text{supp}(A_{i,j})||)$ is large $\Leftrightarrow i$ and $j$ in same layer
• Clustering algorithm for vectors \( v_j = \text{supp}(A,j) \in \{0,1\}^m \)
  • Number of clusters not known a priori
  • Need to exploit sparsity of data vectors
  • Cannot afford to calculate full distance matrix between all pairs of vectors

• Using a centroid-based clustering algorithm
  • Similar to k-means, but with ability to dynamically open up new clusters
  • Identify \((y_{l,k}^+, y_{l,k}^-)\) pairs in advance in the big-M indicator constraints
    • Merge \(y_{l,k}^+\) and \(y_{l,k}^-\) columns to identify more general types of neural networks

• Alternative clustering algorithms that may make sense
  • DBSCAN
  • OPTICS
  • Affinity propagation – probably too slow
  • Mean shift
Clustering Algorithm in Gurobi 10.0

- Identify paired variables $y^+, y^-$ with $y^+y^- = 0$
  - Consider them to be a single variable
- Collect candidates $j$ with at least 5 nonzeros
- Let $\bar{s}_j = \text{supp}(A,j) \in \{0,1\}^m$ and $s_j = \bar{s}_j / \|\bar{s}_j\|

Support vectors are vertices of the $m$-dimensional cube ...
Clustering Algorithm in Gurobi 10.0

- Identify paired variables $y^+, y^-$ with $y^+ y^- = 0$
  - Consider them to be a single variable
- Collect candidates $j$ with at least 5 nonzeros
- Let $\bar{s}_j = \text{supp}(A, j) \in \{0,1\}^m$ and $s_j = \bar{s}_j / \|\bar{s}_j\|$

... projected to the unit sphere in the positive orthant
Clustering Algorithm in Gurobi 10.0

- Identify paired variables \( y^+, y^- \) with \( y^+ y^- = 0 \)
  - Consider them to be a single variable
- Collect candidates \( j \) with at least 5 nonzeros
- Let \( \tilde{s}_j = \text{supp}(A, j) \in \{0,1\}^m \) and \( s_j = \tilde{s}_j/\|\tilde{s}_j\| \)
- Start with \( C = 0 \) clusters and \( \delta = \epsilon = 0.5 \)
- At most 50 times:
  - For all candidates \( j \) in random order:
    - Find closest cluster center vector \( v_k \), if any
    - If \( d(j, k) = 1 - v_k^T s_j < \epsilon \): assign \( j \) to cluster \( k \)
      - Update \( v_k := (v_k + \delta s_j)/\|v_k + \delta s_j\| \in [0,1]^m \)
    - Else if \( C < 100 \): \( C := C + 1 \), \( v_C := s_j \)
    - Else: do not assign \( j \) to any cluster

© 2023 Gurobi Optimization, LLC. Confidential, All Rights Reserved
Clustering Algorithm in Gurobi 10.0

- Identify paired variables $y^+, y^-$ with $y^+ y^- = 0$
  - Consider them to be a single variable
- Collect candidates $j$ with at least 5 nonzeros
- Let $\tilde{s}_j = \text{supp}(A,j) \in \{0,1\}^m$ and $s_j = \tilde{s}_j / \|\tilde{s}_j\|
- Start with $C = 0$ clusters and $\delta = \epsilon = 0.5$
- At most 50 times:
  - For all candidates $j$ in random order:
    - Find closest cluster center vector $v_k$, if any
      - If $d(j,k) := 1 - v_k^T s_j < \epsilon$: assign $j$ to cluster $k$
        - Update $v_k := (v_k + \delta s_j) / \|v_k + \delta s_j\| \in [0,1]^m$
      - Else if $C < 100$: $C := C + 1$, $v_C := s_j$
      - Else: do not assign $j$ to any cluster
  - Update $\delta := 0.97 \delta$ and $\epsilon := 0.98 \epsilon$.
- If all $v_j^T v_k > 0.7$: stop (success)
- If all clusters have less than 10 variables: stop (fail)
Clustering Algorithm in Gurobi 10.0

- Identify paired variables $y^+, y^-$ with $y^+y^- = 0$
  - Consider them to be a single variable
- Collect candidates $j$ with at least 5 nonzeros
- Let $\bar{s}_j = \text{supp}(A,j) \in \{0,1\}^m$ and $s_j = \bar{s}_j / \|\bar{s}_j\|
- Start with $C = 0$ clusters and $\delta = \epsilon = 0.5$
- At most 50 times:
  - For all candidates $j$ in random order:
    - Find closest cluster center vector $v_k$, if any
    - If $d(j,k) = 1 - v_k^T s_j < \epsilon$: assign $j$ to cluster $k$
      - Update $v_k := (v_k + \delta s_j)/\|v_k + \delta s_j\| \in [0,1]^m$
    - Else if $C < 100$: $C := C + 1, v_C := s_j$
    - Else: do not assign $j$ to any cluster

© 2023 Gurobi Optimization, LLC. Confidential, All Rights Reserved | 37
Clustering Algorithm in Gurobi 10.0

- Identify paired variables $y^+, y^-$ with $y^+ y^- = 0$
  - Consider them to be a single variable
- Collect candidates $j$ with at least 5 nonzeros
- Let $\tilde{s}_j = \text{supp}(A, j) \in \{0,1\}^m$ and $s_j = \tilde{s}_j / \|\tilde{s}_j\|
- Start with $C = 0$ clusters and $\delta = \epsilon = 0.5$
- At most 50 times:
  - For all candidates $j$ in random order:
    - Find closest cluster center vector $v_k$, if any
    - If $d(j, k) := 1 - v_k^T s_j < \epsilon$: assign $j$ to cluster $k$
      - Update $v_k := (v_k + \delta s_j) / \|v_k + \delta s_j\| \in [0,1]^m$
    - Else if $C < 100$: $C := C + 1$, $v_C := s_j$
    - Else: do not assign $j$ to any cluster
- If all $s_j^T v_k > 0.7$: stop (success)
- If all clusters have less than 10 variables: stop (fail)
Clustering Algorithm in Gurobi 10.0

- Identify paired variables $y^+, y^-$ with $y^+ y^- = 0$
  - Consider them to be a single variable
- Collect candidates $j$ with at least 5 nonzeros
- Let $\bar{s}_j = \text{supp}(A, j) \in \{0,1\}^m$ and $s_j = \bar{s}_j / \|\bar{s}_j\|
- Start with $C = 0$ clusters and $\delta = \epsilon = 0.5$
- At most 50 times:
  - For all candidates $j$ in random order:
    - Find closest cluster center vector $v_k$, if any
    - If $d(j, k) := 1 - v_k^T s_j < \epsilon$: assign $j$ to cluster $k$
      - Update $v_k := (v_k + \delta s_j) / \|v_k + \delta s_j\| \in [0,1]^m$
    - Else if $C < 100$: $C := C + 1, v_C := s_j$
    - Else: do not assign $j$ to any cluster

© 2023 Gurobi Optimization, LLC. Confidential, All Rights Reserved | 39
Clustering Algorithm in Gurobi 10.0

- Identify paired variables $y^+, y^-$ with $y^+y^- = 0$
  - Consider them to be a single variable
- Collect candidates $j$ with at least 5 nonzeros
- Let $\bar{s}_j = \text{supp}(A, j) \in \{0,1\}^m$ and $s_j = \bar{s}_j/\|\bar{s}_j\|$
- Start with $C = 0$ clusters and $\delta = \epsilon = 0.5$
- At most 50 times:
  - For all candidates $j$ in random order:
    - Find closest cluster center vector $v_k$, if any
    - If $d(j, k) := 1 - v_k^T s_j < \epsilon$: assign $j$ to cluster $k$
      - Update $v_k := (v_k + \delta s_j)/\|v_k + \delta s_j\| \in [0,1]^m$
    - Else if $C < 100$: $C := C + 1, v_C := s_j$
    - Else: do not assign $j$ to any cluster

© 2023 Gurobi Optimization, LLC. Confidential, All Rights Reserved |
Identify paired variables $y^+, y^-$ with $y^+y^- = 0$
  - Consider them to be a single variable
Collect candidates $j$ with at least 5 nonzeros
Let $\bar{s}_j = \text{supp}(A,j) \in \{0,1\}^m$ and $s_j = \bar{s}_j/\|\bar{s}_j\|
Start with $C = 0$ clusters and $\delta = \epsilon = 0.5$
At most 50 times:
  - For all candidates $j$ in random order:
    - Find closest cluster center vector $v_k$, if any
    - If $d(j,k) := 1 - v_k^T s_j < \epsilon$: assign $j$ to cluster $k$
      - Update $v_k := (v_k + \delta s_j)/\|v_k + \delta s_j\| \in [0,1]^m$
    - Else if $C < 100$: $C := C + 1, v_C := s_j$
    - Else: do not assign $j$ to any cluster

© 2023 Gurobi Optimization, LLC. Confidential, All Rights Reserved | 41
Clustering Algorithm in Gurobi 10.0

- Identify paired variables $y^+, y^-$ with $y^+y^- = 0$
  - Consider them to be a single variable
- Collect candidates $j$ with at least 5 nonzeros
- Let $\tilde{s}_j = \text{supp}(A, j) \in \{0,1\}^m$ and $s_j = \tilde{s}_j/\|\tilde{s}_j\|$
- Start with $C = 0$ clusters and $\delta = \epsilon = 0.5$
- At most 50 times:
  - For all candidates $j$ in random order:
    - Find closest cluster center vector $v_k$, if any
    - If $d(j, k) := 1 - v_k^Ts_j < \epsilon$: assign $j$ to cluster $k$
      - Update $v_k := (v_k + \delta s_j)/\|v_k + \delta s_j\| \in [0,1]^m$
    - Else if $C < 100$: $C := C + 1, v_C := s_j$
    - Else: do not assign $j$ to any cluster
- If all $s_j^Tv_k > 0.7$: stop (success)
- If all clusters have less than 10 variables: stop (fail)
Clustering Algorithm in Gurobi 10.0

- Identify paired variables $y^+, y^-$ with $y^+y^- = 0$
  - Consider them to be a single variable
- Collect candidates $j$ with at least 5 nonzeros
- Let $\tilde{s}_j = \text{supp}(A_j) \in \{0,1\}^m$ and $s_j = \tilde{s}_j / \|\tilde{s}_j\|$  
- Start with $C = 0$ clusters and $\delta = \epsilon = 0.5$
- At most 50 times:
  - For all candidates $j$ in random order:
    - Find closest cluster center vector $v_k$, if any
    - If $d(j,k) = 1 - v_k^T s_j < \epsilon$: assign $j$ to cluster $k$
      - Update $v_k := (v_k + \delta s_j) / \|v_k + \delta s_j\| \in [0,1]^m$
    - Else if $C < 100$: $C := C + 1, v_C := s_j$
    - Else: do not assign $j$ to any cluster
- If all $s_j^T v_k > 0.7$: stop (success)
- If all clusters have less than 10 variables: stop (fail)
Identify paired variables $y^+, y^-$ with $y^+y^- = 0$
  - Consider them to be a single variable
Collect candidates $j$ with at least 5 nonzeros
Let $\tilde{s}_j = \text{supp}(A, j) \in \{0,1\}^m$ and $s_j = \tilde{s}_j/\|\tilde{s}_j\|
Start with $C = 0$ clusters and $\delta = \epsilon = 0.5$
At most 50 times:
  - For all candidates $j$ in random order:
    - Find closest cluster center vector $v_k$, if any
    - If $d(j, k) := 1 - v_k^T s_j < \epsilon$: assign $j$ to cluster $k$
      - Update $v_k := (v_k + \delta s_j)/\|v_k + \delta s_j\| \in [0,1]^m$
    - Else if $C < 100$: $C := C + 1, v_C := s_j$
    - Else: do not assign $j$ to any cluster
  - Update $\delta := 0.97\delta$ and $\epsilon := 0.98\epsilon$
  - If all $s_j^T v_{kj} > 0.7$: stop (success)
  - If all clusters have less than 10 variables: stop (fail)
Performance Evaluation
Benchmarks: Test Set

- Goldstein-Price and Peak2d: 60 instances each,
  - Approximation of a nonlinear function with a neural network
  - \#layers \in \{2,3\} of \#neurons \in \{56,128,256\} each
  - 10 networks for each architecture trained with different seeds using scikit-learn
- Janos (Bergman et.al. 2019): 128 instances
  - 500 predictor constraints for each model
  - All regression models of scikit-learn, various hyperparameters
- TCL (Amasyali et.al. 2022): 70 instances
  - 40 PyTorch models, 30 scikit-learn: \#layers \in \{2,3\} of \#neurons \in \{128, 256\} each
  - Application in electrical engineering find valid input/output within bounds minimizing costs
- Adversarial machine learning on MNIST: 210 instances
  - scikit-learn: 2 layers of \#neurons \in \{50,100\} and 6 layers of 500 neurons, 30 models each
  - Tensorflow: \#layers \in \{2,3\} of \#neurons \in \{50, 100, 200\}, 20 models each
Computational Setup

- Models solved on Intel(R) Xeon(R) CPU E3-1240 CPUs
  - 3.5 GHz, 4 cores, 4 threads, 32 GB RAM
- Run Gurobi 9.5 and Gurobi 10.0
- Time limit 10,000 seconds
- Models with logistic regression excluded (9.5 can’t solve)
- Models not solved by any in the time limit excluded
- Solve means 0.01% gap reached
  - Typically, most of the time is spent on proving the dual bound
  - Best solution is usually found much earlier
# Gurobi 9.5 vs Gurobi 10.0

<table>
<thead>
<tr>
<th>test set</th>
<th># models</th>
<th>% solved</th>
<th>time</th>
<th>% solved</th>
<th>time</th>
<th>speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>GoldsteinPrice</td>
<td>43</td>
<td>100%</td>
<td>55</td>
<td>100%</td>
<td>24</td>
<td>2.2x</td>
</tr>
<tr>
<td>Peak2d</td>
<td>41</td>
<td>83%</td>
<td>120</td>
<td>100%</td>
<td>35</td>
<td>3.3x</td>
</tr>
<tr>
<td>Janos</td>
<td>38</td>
<td>97%</td>
<td>48</td>
<td>100%</td>
<td>39</td>
<td>1.2x</td>
</tr>
<tr>
<td>TCL</td>
<td>65</td>
<td>23%</td>
<td>5130</td>
<td>100%</td>
<td>63</td>
<td>80.1x</td>
</tr>
<tr>
<td>MNIST</td>
<td>136</td>
<td>47%</td>
<td>1614</td>
<td>100%</td>
<td>135</td>
<td>11.9x</td>
</tr>
</tbody>
</table>
Gurobi 9.5 vs Gurobi 10.0

© 2023 Gurobi Optimization, LLC. Confidential, All Rights Reserved | 49
Adversarial Machine Learning

- Given a trained neural network and one training example $\bar{x}$
- In a small neighborhood of $\bar{x}$ show that either
  - Everything is classified like training example, or
  - Find a misclassified counter-example

- See
  - Fischetti, Jo 2018
  - Kouvaros, Lomuscio 2018
## Adversarial Model: Detailed Results

<table>
<thead>
<tr>
<th># layers</th>
<th>size</th>
<th># models</th>
<th>% solved</th>
<th>time</th>
<th>% solved</th>
<th>time</th>
<th>speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Keras</strong></td>
<td>2</td>
<td>50</td>
<td>100%</td>
<td>21</td>
<td>100%</td>
<td>3</td>
<td>5.4x</td>
</tr>
<tr>
<td></td>
<td></td>
<td>100</td>
<td>95%</td>
<td>404</td>
<td>100%</td>
<td>20</td>
<td>19.3x</td>
</tr>
<tr>
<td></td>
<td></td>
<td>200</td>
<td>50%</td>
<td>2764</td>
<td>95%</td>
<td>28</td>
<td>94.5x</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>50</td>
<td>95%</td>
<td>197</td>
<td>100%</td>
<td>7</td>
<td>24.6x</td>
</tr>
<tr>
<td></td>
<td></td>
<td>100</td>
<td>35%</td>
<td>4977</td>
<td>95%</td>
<td>75</td>
<td>65.7x</td>
</tr>
<tr>
<td></td>
<td></td>
<td>200</td>
<td>15%</td>
<td>9272</td>
<td>95%</td>
<td>105</td>
<td>87.6x</td>
</tr>
<tr>
<td><strong>scikit-learn</strong></td>
<td>2</td>
<td>50</td>
<td>100%</td>
<td>17</td>
<td>100%</td>
<td>12</td>
<td>1.4x</td>
</tr>
<tr>
<td></td>
<td>100</td>
<td>93%</td>
<td>511</td>
<td>100%</td>
<td>66</td>
<td>7.7x</td>
<td></td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>500</td>
<td>0%</td>
<td>10000</td>
<td>0%</td>
<td>10000</td>
<td>—</td>
</tr>
</tbody>
</table>
### Adversarial Model: Fischetti/Jo Test Set

Time limit as in Fischetti/Jo: 300 sec

<table>
<thead>
<tr>
<th>basic models</th>
<th>network layers</th>
<th># mod</th>
<th>Gurobi 9.5</th>
<th>CPLEX 12.7*</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>8-8-8</td>
<td>99</td>
<td>100%</td>
<td>100%</td>
</tr>
<tr>
<td></td>
<td>8-8-8-8-8</td>
<td>99</td>
<td>100%</td>
<td>97%</td>
</tr>
<tr>
<td></td>
<td>20-10-8-8</td>
<td>99</td>
<td>100%</td>
<td>64%</td>
</tr>
<tr>
<td></td>
<td>20-10-8-8-8</td>
<td>99</td>
<td>99%</td>
<td>24%</td>
</tr>
<tr>
<td></td>
<td>20-20-10-10-10</td>
<td>100</td>
<td>27%</td>
<td>7%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Fischetti/Jo MIP-OBBT</th>
<th>network layers</th>
<th># mod</th>
<th>Gurobi 9.5</th>
<th>CPLEX 12.7*</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>8-8-8</td>
<td>99</td>
<td>100%</td>
<td>100%</td>
</tr>
<tr>
<td></td>
<td>8-8-8-8-8-8</td>
<td>99</td>
<td>100%</td>
<td>100%</td>
</tr>
<tr>
<td></td>
<td>20-10-8-8</td>
<td>99</td>
<td>100%</td>
<td>100%</td>
</tr>
<tr>
<td></td>
<td>20-10-8-8-8</td>
<td>99</td>
<td>100%</td>
<td>98%</td>
</tr>
<tr>
<td></td>
<td>20-20-10-10-10</td>
<td>100</td>
<td>89%</td>
<td>67%</td>
</tr>
</tbody>
</table>

* From Fischetti/Jo paper, run on a 2.3 GHz Intel i7 laptop with 16 GB RAM (in 2017)

<table>
<thead>
<tr>
<th>basic model</th>
<th>% solved</th>
<th>% gap</th>
<th>nodes</th>
<th>time (s)</th>
<th>improved model</th>
<th>% solved</th>
<th>% gap</th>
<th>nodes</th>
<th>time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>DNN1</td>
<td>100</td>
<td>0.0</td>
<td>1,903</td>
<td>1.0</td>
<td>100</td>
<td>0.0</td>
<td>552</td>
<td>0.6</td>
<td></td>
</tr>
<tr>
<td>DNN2</td>
<td>97</td>
<td>0.2</td>
<td>7,787</td>
<td>48.2</td>
<td>100</td>
<td>0.0</td>
<td>11,851</td>
<td>7.5</td>
<td></td>
</tr>
<tr>
<td>DNN3</td>
<td>64</td>
<td>11.6</td>
<td>228,632</td>
<td>158.5</td>
<td>100</td>
<td>0.0</td>
<td>20,309</td>
<td>12.1</td>
<td></td>
</tr>
<tr>
<td>DNN4</td>
<td>24</td>
<td>38.1</td>
<td>282,694</td>
<td>263.0</td>
<td>98</td>
<td>0.7</td>
<td>68,563</td>
<td>43.9</td>
<td></td>
</tr>
<tr>
<td>DNN5</td>
<td>7</td>
<td>71.8</td>
<td>193,725</td>
<td>290.9</td>
<td>67</td>
<td>11.4</td>
<td>76,714</td>
<td>171.1</td>
<td></td>
</tr>
</tbody>
</table>

Table 1 of Fischetti and Jo (2018)

- MIP based OBBT of Fischetti/Jo helps significantly for both Gurobi 9.5 and CPLEX 12.7
### Adversarial Model: Fischetti/Jo Test Set

Time limit as in Fischetti/Jo: 300 sec

<table>
<thead>
<tr>
<th>basic models</th>
<th>network layers</th>
<th># mod</th>
<th>Gurobi 9.5 % solved</th>
<th>Gurobi 10.0 % solved</th>
<th>speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>8-8-8</td>
<td>99</td>
<td>100%</td>
<td>100%</td>
<td>1.7x</td>
</tr>
<tr>
<td></td>
<td>8-8-8-8-8-8</td>
<td>99</td>
<td>100%</td>
<td>100%</td>
<td>2.4x</td>
</tr>
<tr>
<td></td>
<td>20-10-8-8</td>
<td>99</td>
<td>100%</td>
<td>100%</td>
<td>3.2x</td>
</tr>
<tr>
<td></td>
<td>20-10-8-8-8</td>
<td>99</td>
<td>99%</td>
<td>100%</td>
<td>4.0x</td>
</tr>
<tr>
<td></td>
<td>20-20-10-10-10</td>
<td>100</td>
<td>27%</td>
<td>88%</td>
<td>4.2x</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Fischetti/Jo MIP-OBBT</th>
<th>network layers</th>
<th># mod</th>
<th>Gurobi 9.5 % solved</th>
<th>Gurobi 10.0 % solved</th>
<th>speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>8-8-8</td>
<td>99</td>
<td>100%</td>
<td>100%</td>
<td>1.0x</td>
</tr>
<tr>
<td></td>
<td>8-8-8-8-8-8</td>
<td>99</td>
<td>100%</td>
<td>100%</td>
<td>1.0x</td>
</tr>
<tr>
<td></td>
<td>20-10-8-8</td>
<td>99</td>
<td>100%</td>
<td>100%</td>
<td>1.1x</td>
</tr>
<tr>
<td></td>
<td>20-10-8-8-8</td>
<td>99</td>
<td>100%</td>
<td>100%</td>
<td>1.3x</td>
</tr>
<tr>
<td></td>
<td>20-20-10-10-10</td>
<td>100</td>
<td>89%</td>
<td>90%</td>
<td>1.2x</td>
</tr>
</tbody>
</table>

- Gurobi 10 much faster than 9.5 on basic model
- Only small speedup when bounds are already tightened in input model
  - Indicates that performance comes mostly from Gurobi 10's OBBT
### Adversarial Model: Fischetti/Jo Test Set

Time limit as in Fischetti/Jo: 300 sec

<table>
<thead>
<tr>
<th>basic models</th>
<th>Gurobi 9.5</th>
<th>Gurobi 10.0</th>
<th>Gurobi 10.0, OBBT=3</th>
</tr>
</thead>
<tbody>
<tr>
<td># mod</td>
<td>% solved</td>
<td>% solved</td>
<td>speedup</td>
</tr>
<tr>
<td>8-8-8</td>
<td>99</td>
<td>100%</td>
<td>100%</td>
</tr>
<tr>
<td>8-8-8-8-8-8</td>
<td>99</td>
<td>100%</td>
<td>100%</td>
</tr>
<tr>
<td>20-10-8-8</td>
<td>99</td>
<td>99%</td>
<td>100%</td>
</tr>
<tr>
<td>20-10-8-8-8</td>
<td>99</td>
<td>99%</td>
<td>100%</td>
</tr>
<tr>
<td>20-20-10-10-10</td>
<td>100</td>
<td>27%</td>
<td>88%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Fischetti/Jo MIP-OBBT</th>
<th>Gurobi 9.5</th>
<th>Gurobi 10.0</th>
<th>Gurobi 10.0, OBBT=3</th>
</tr>
</thead>
<tbody>
<tr>
<td># mod</td>
<td>% solved</td>
<td>% solved</td>
<td>speedup</td>
</tr>
<tr>
<td>8-8-8</td>
<td>99</td>
<td>100%</td>
<td>100%</td>
</tr>
<tr>
<td>8-8-8-8-8-8-8</td>
<td>99</td>
<td>100%</td>
<td>100%</td>
</tr>
<tr>
<td>20-10-8-8</td>
<td>99</td>
<td>100%</td>
<td>100%</td>
</tr>
<tr>
<td>20-10-8-8-8-8</td>
<td>99</td>
<td>100%</td>
<td>100%</td>
</tr>
<tr>
<td>20-20-10-10-10-10</td>
<td>100</td>
<td>89%</td>
<td>90%</td>
</tr>
</tbody>
</table>

- Gurobi 10 much faster than 9.5 on basic model
- Only small speedup when bounds are already tightened in input model
  - Indicates that performance comes mostly from Gurobi 10’s OBBT
- Aggressive OBBT helps further on larger networks
### Adversarial Model: Fischetti/Jo Test Set

Time limit as in Fischetti/Jo: 300 sec

<table>
<thead>
<tr>
<th>basic models</th>
<th>Gurobi 9.5</th>
<th>Gurobi 10.0</th>
<th>Gurobi 10.0, OBBT=3</th>
<th>dev version</th>
<th>dev version, OBBT=3</th>
</tr>
</thead>
<tbody>
<tr>
<td>network layers</td>
<td># mod</td>
<td>% solved</td>
<td>% solved</td>
<td>speedup</td>
<td>% solved</td>
</tr>
<tr>
<td>8-8-8</td>
<td>99</td>
<td>100%</td>
<td>100%</td>
<td>1.7x</td>
<td>100%</td>
</tr>
<tr>
<td>8-8-8-8-8-8</td>
<td>99</td>
<td>100%</td>
<td>100%</td>
<td>2.4x</td>
<td>100%</td>
</tr>
<tr>
<td>20-10-8-8</td>
<td>99</td>
<td>100%</td>
<td>100%</td>
<td>3.2x</td>
<td>100%</td>
</tr>
<tr>
<td>20-10-8-8-8</td>
<td>99</td>
<td>99%</td>
<td>100%</td>
<td>4.0x</td>
<td>100%</td>
</tr>
<tr>
<td>20-20-10-10-10</td>
<td>100</td>
<td>27%</td>
<td>88%</td>
<td>4.2x</td>
<td>95%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Fischetti/Jo MIP-OBBT</th>
<th>Gurobi 9.5</th>
<th>Gurobi 10.0</th>
<th>Gurobi 10.0, OBBT=3</th>
<th>dev version</th>
<th>dev version, OBBT=3</th>
</tr>
</thead>
<tbody>
<tr>
<td>network layers</td>
<td># mod</td>
<td>% solved</td>
<td>% solved</td>
<td>speedup</td>
<td>% solved</td>
</tr>
<tr>
<td>8-8-8</td>
<td>99</td>
<td>100%</td>
<td>100%</td>
<td>1.0x</td>
<td>100%</td>
</tr>
<tr>
<td>8-8-8-8-8-8</td>
<td>99</td>
<td>100%</td>
<td>100%</td>
<td>1.0x</td>
<td>100%</td>
</tr>
<tr>
<td>20-10-8-8</td>
<td>99</td>
<td>100%</td>
<td>100%</td>
<td>1.1x</td>
<td>100%</td>
</tr>
<tr>
<td>20-10-8-8-8</td>
<td>99</td>
<td>100%</td>
<td>100%</td>
<td>1.3x</td>
<td>100%</td>
</tr>
<tr>
<td>20-20-10-10-10</td>
<td>100</td>
<td>89%</td>
<td>90%</td>
<td>1.2x</td>
<td>96%</td>
</tr>
</tbody>
</table>
Conclusions

• Gurobi Machine Learning:
  • https://github.com/Gurobi/gurobi-machinelearning
  • Input very welcome: questions, suggestions, bug reports, pull requests, ...

• Performance for models with neural networks in Gurobi 10

• Also interesting for data science: gurobipy-pandas
  • https://github.com/Gurobi/gurobipy-pandas

• Dangers and Pitfalls
  • Complexity of ML models we can hope to handle is still limited
  • Methodological questions:
    • How to decide which prediction model to use?
    • How to make sure that optimization doesn’t misuse results of the predictor?
Thank You!