Pierre Bonami, Tobias Achterberg
Gurobi R&D

GUROBI
OPTIMIZATION

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:
  - $d = g(year, season, \mathrm{r}, p)$
- 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

$$d = g(year, season, \mathrm{r}, p)$$

  in the optimization problem
- d and p become variables for the optimization
- Notebook developed by J. Yurchisin and R. Swamy

Definitions and Scope

- 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)
- …

Bergman, Huang, Brooks, Lodi, Raghunathan

Maragno, Wiberg, Bertsimas, Birbil, den Hertog, Fajemisin

Ceccon, Jalving, Haddad, Thebelt, Tsay, Laird, Misener

Gurobi Machine Learning

Gurobi Machine Learning

GUROBI OPTIMIZATION

- 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

scikit learn

- 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

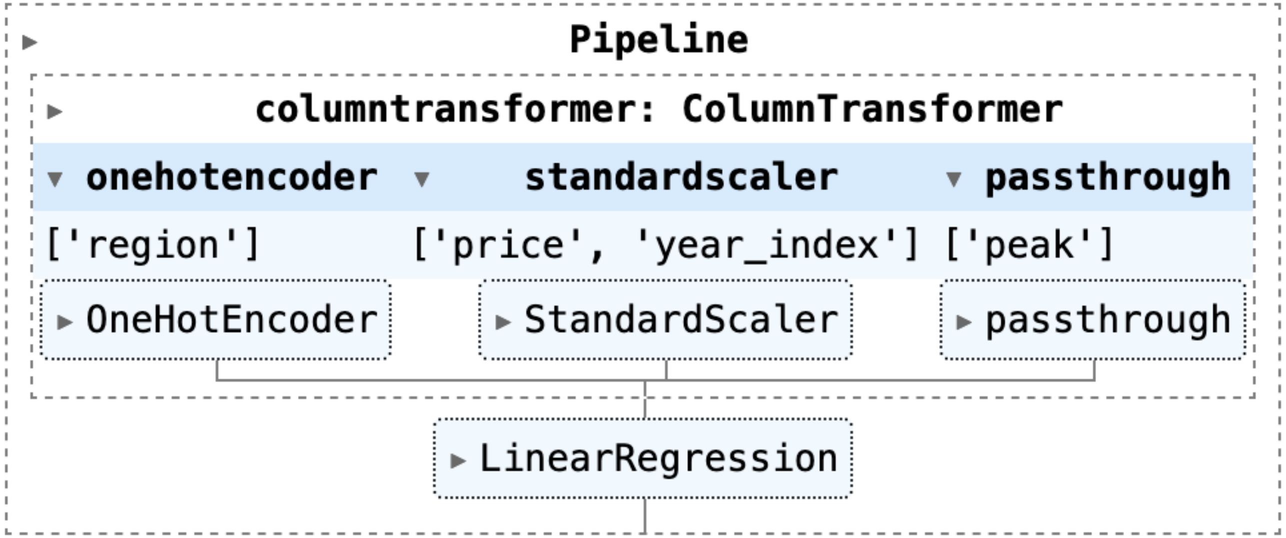
Keras

- 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 = gp.Model("Avocado_Price_Allocation")

p = gppd.add_vars(m, data, lb=0.0, ub=2.0)
d = gppd.add_vars(m, data)
u = m.addVar()
```

gurobipy-pandas

- Variables
  - p_r selling price per unit
  - d_r demand
  - u total unsold products

	price	demand
Great_Lakes	<gurobi.Var price[Great_Lakes]>	<gurobi.Var demand[Great_Lakes]>
Midsouth	<gurobi.Var price[Midsouth]>	<gurobi.Var demand[Midsouth]>
Northeast	<gurobi.Var price[Northeast]>	<gurobi.Var demand[Northeast]>
Northern_New_England	<gurobi.Var price[Northern_New_England]>	<gurobi.Var demand[Northern_New_England]>
SouthCentral	<gurobi.Var price[SouthCentral]>	<gurobi.Var demand[SouthCentral]>
Southeast	<gurobi.Var price[Southeast]>	<gurobi.Var demand[Southeast]>
West	<gurobi.Var price[West]>	<gurobi.Var demand[West]>
Plains	<gurobi.Var price[Plains]>	<gurobi.Var demand[Plains]>

Example: Objective and Constraints

waste penalty

transportation costs

$$\max \sum_r (p_r - c_r) d_r - w \cdot u \qquad \text{(maximize revenue)}$$

$s.t.$

total supply

$$\sum_r d_r + u = S, \qquad \text{(allocate supply)}$$

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

```
m.setObjective(((p - c) * d).sum() - w * u, GRB.MAXIMIZE)
m.addConstr(d.sum() + u == S)
add_predictor_constr(m, pipeline, feats, d)
```

gurobipy-machinelearning

Example: Input of Regression Constraints

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

```
feats = pd.DataFrame(
    data={
        "year": 2020,
        "peak": 1,
        "region": regions,
        “price": p
    },
    index=regions)
```

	year	peak	region	price
Great_Lakes	2020	1	Great_Lakes	<gurobi.Var price[Great_Lakes]>
Midsouth	2020	1	Midsouth	<gurobi.Var price[Midsouth]>
Northeast	2020	1	Northeast	<gurobi.Var price[Northeast]>
Northern_New_England	2020	1	Northern_New_England	<gurobi.Var price[Northern_New_England]>
SouthCentral	2020	1	SouthCentral	<gurobi.Var price[SouthCentral]>
Southeast	2020	1	Southeast	<gurobi.Var price[Southeast]>
West	2020	1	West	<gurobi.Var price[West]>
Plains	2020	1	Plains	<gurobi.Var price[Plains]>

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:

--------------------------------------------------------------------------------
Step            Output Shape    Variables              Constraints
                                                Linear    Quadratic      General
================================================================================
col_trans            (8, 10)           24           16            0            0

lin_reg               (8, 1)           64            8            0            0

--------------------------------------------------------------------------------
```


Example: Optimizing

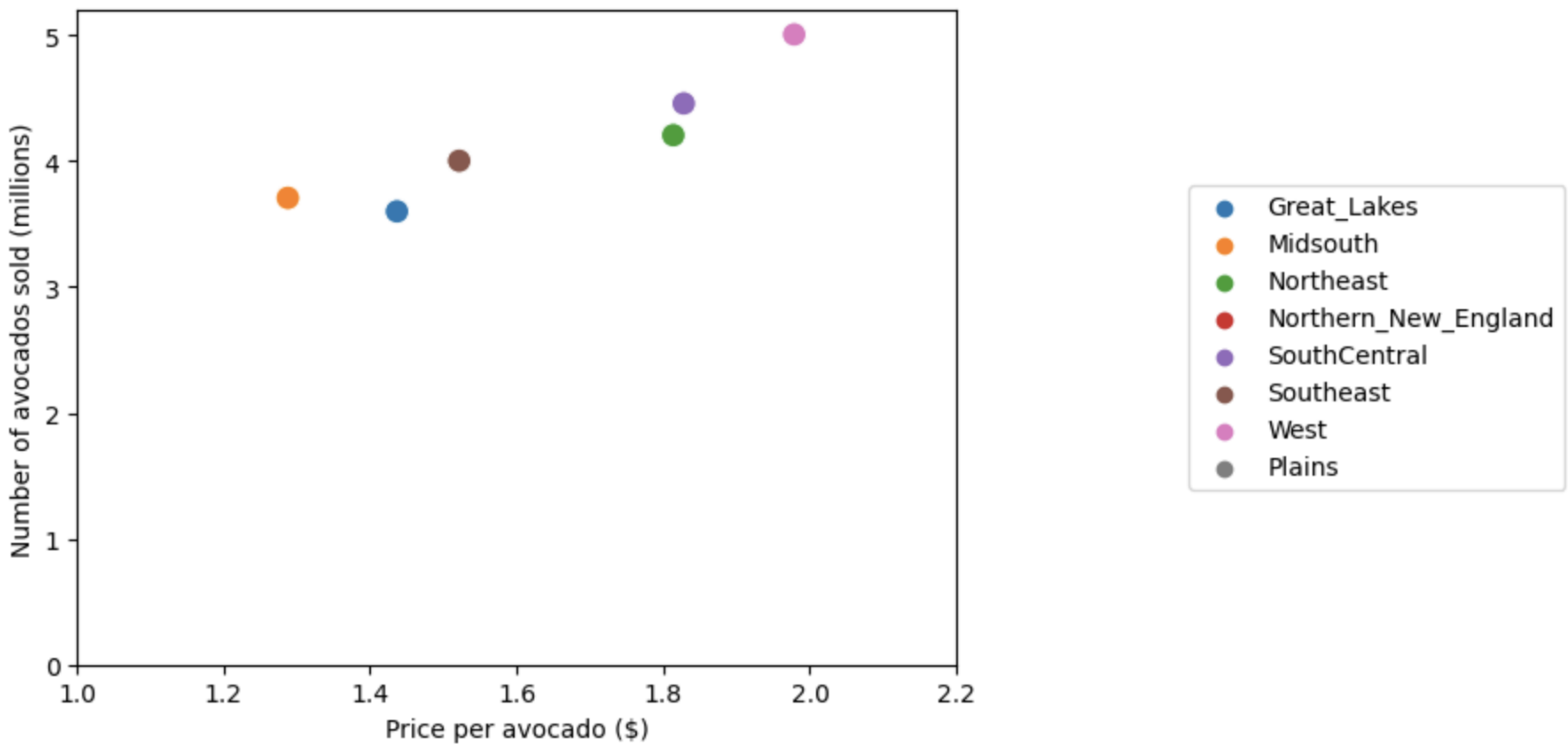
```
m.Params.NonConvex = 2
m.optimize()
```

```
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

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

```
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

Gurobi 10.0

- 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

Gurobi 10.0

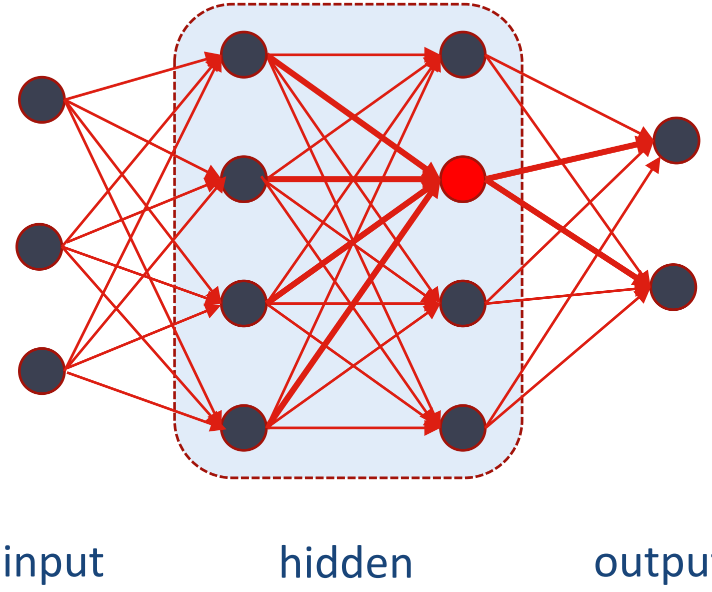
- New features for models with ML predictor constraints
  - Logistic function as general function constraint
- Performance improvements relevant for models with ML predictor constraints
  - Optimization based bound tightening (OBBT)
  - Neural network detection

Neural Networks with ReLU

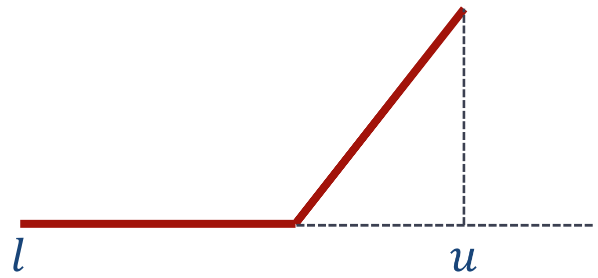
- Each neuron k has the following constraints/variables:

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

- The $\max$ function is nonlinear and formulated using a binary variable and big-M constraints



$y = \max(x, 0) : l \leq x \leq u$

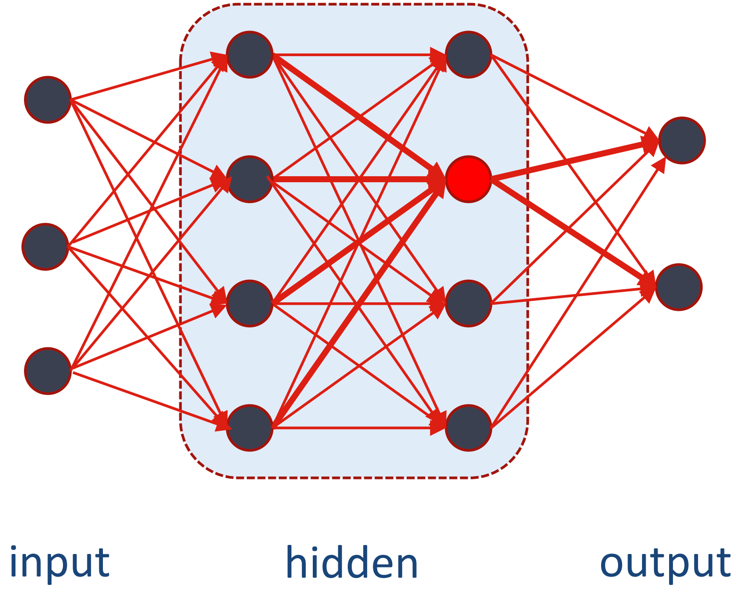


Neural Networks with ReLU

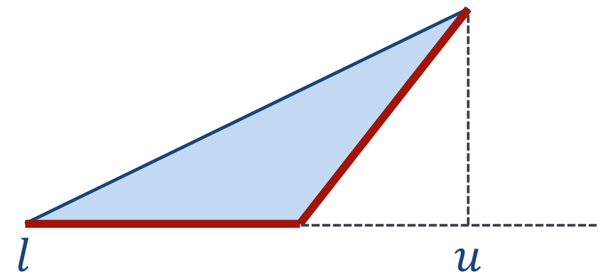
- Each neuron k has the following constraints/variables:

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

- 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_{mix}
- Optimization based bound tightening known to be essential for adversarial NN
  - e.g., Fischetti, Jo 2018, Weng et.al. 2018

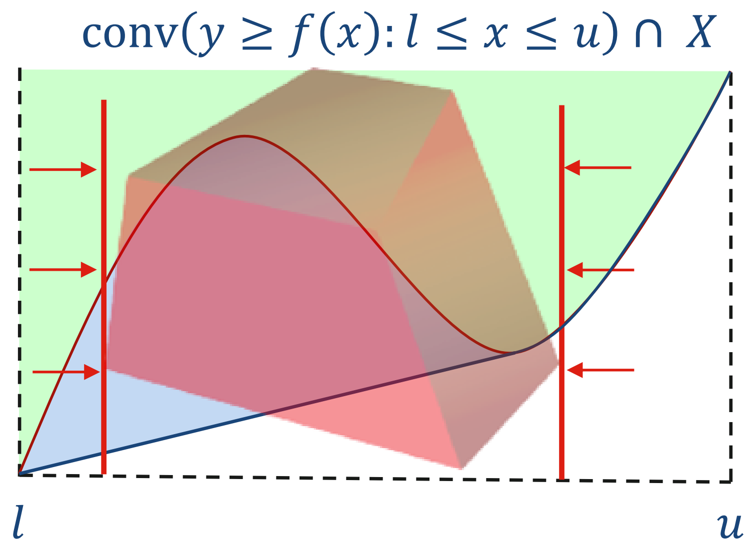


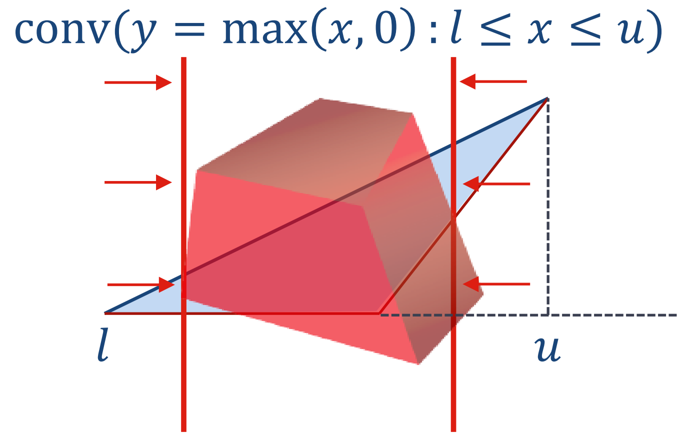
$$\text{conv}(y = \max(x, 0) : l \leq x \leq u)$$



Optimization Based Bound Tightening

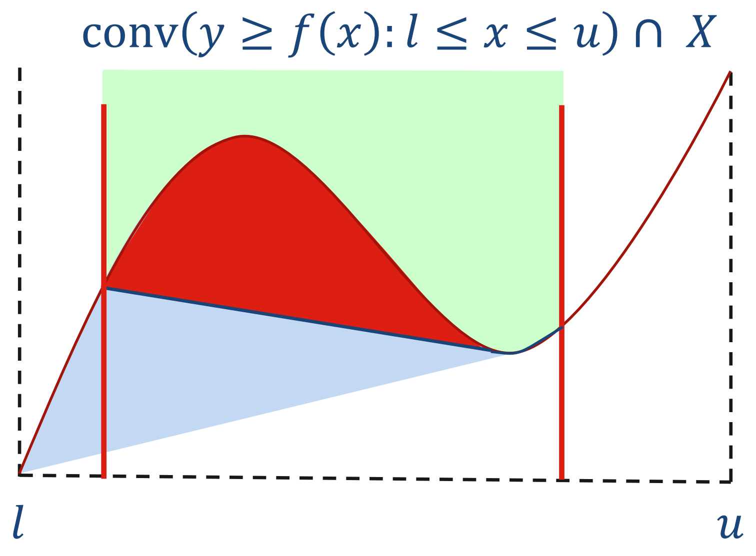
- Common technique for MINLP solvers
- Given the LP relaxation of a (non-convex) MI(NL)P
- For each variable x
  - Minimize/maximize x value over relaxation
  - Use optimal value as lower/upper bound for x

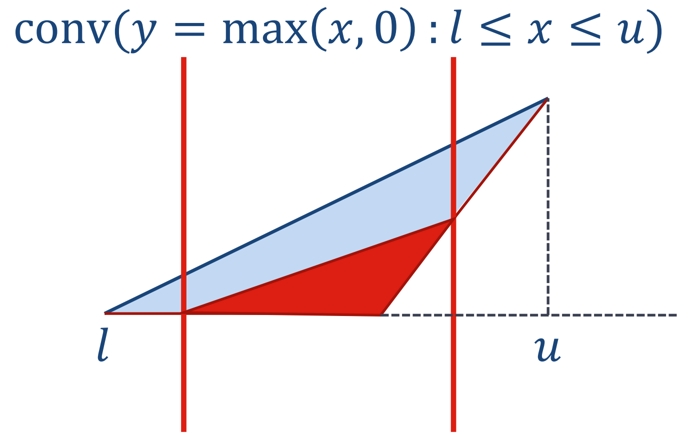




Optimization Based Bound Tightening

- 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

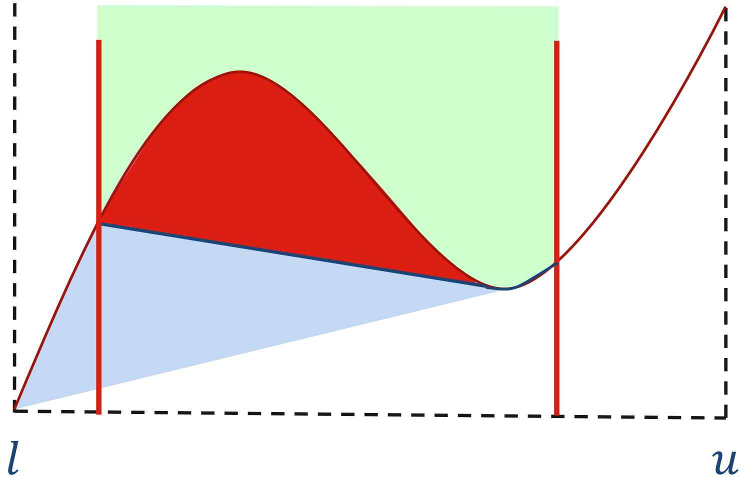




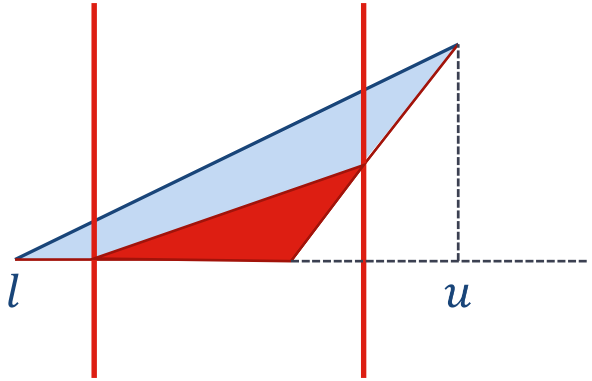
Optimization Based Bound Tightening

- For non-convex MIQCP:
  - 23% improvement overall
  - 61% improvement on models solved in ≥ 100 sec.
- For MIP and convex MIQP/MIQCP:
  - 1% improvement on models solved in ≥ 100 sec.
  - But big improvement on models with ReLU neural networks

$\mathrm{conv}(y \geq f(x): l \leq x \leq u) \cap X$



$\mathrm{conv}(y = \max(x, 0) : l \leq x \leq u)$



ReLU Neural Network MIP Formulation

- Each neuron k has the following constraints/variables:

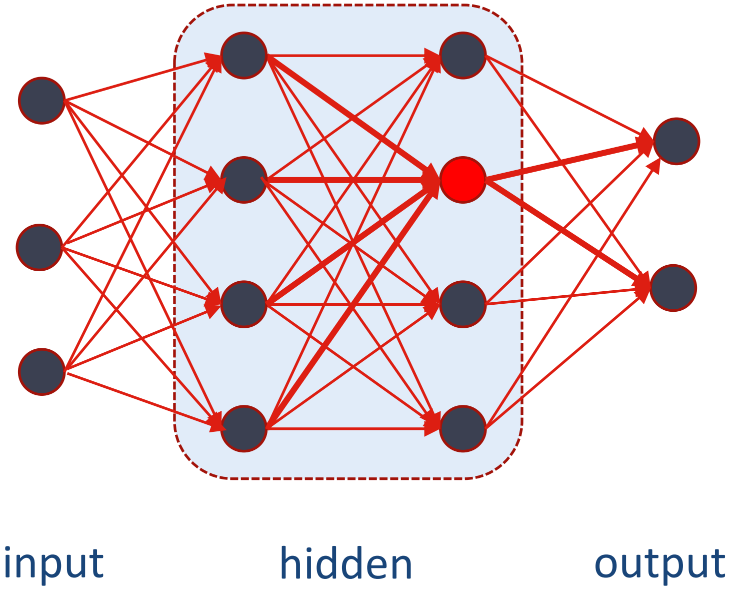
$$y_{\mathrm{mix}} = w^T x_{\mathrm{in}} + d$$
$$y_{\mathrm{out}} = \max(y_{\mathrm{mix}}, 0)$$

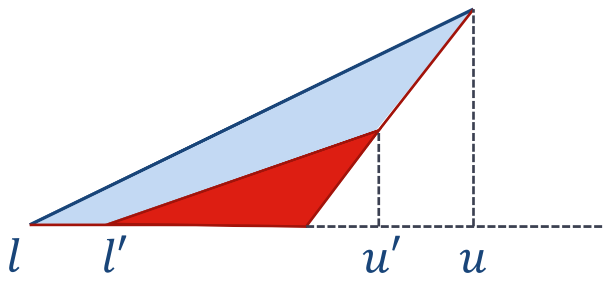
- MIP formulation:

$$y^+ - y^- = w^T x_{\mathrm{in}} + d$$
$$y^+ \leq u^+ \cdot z$$
$$y^- \leq u^- \cdot (1 - z)$$
$$0 \leq y^+ \leq u^+$$
$$0 \leq y^- \leq u^-$$

  - Output of neuron is just $y_{\mathrm{out}} := y^+$

- Strength of Big-M formulation of indicators depends on bounds $-u^- \leq y_{\mathrm{mix}} \leq u^+$ of y_{mix}





Bound Propagation

- Constraints for $y_{l,k} = y^+_{l,k} - y^-_{l,k}$ of k'th neuron in layer l:

  $$y^+_{l,k} - y^-_{l,k} = w^T_{l,k} y^+_{l-1,\cdot} + d_{l,k}$$

  $$y^+_{l+1,i} - y^+_{l+1,i} = w_{l+1,i,k} y^+_{l,k} + \sum_{j \neq k} w_{l+1,i,j} y^+_{l,j} + d_{l+1,i} \text{ for all } i$$

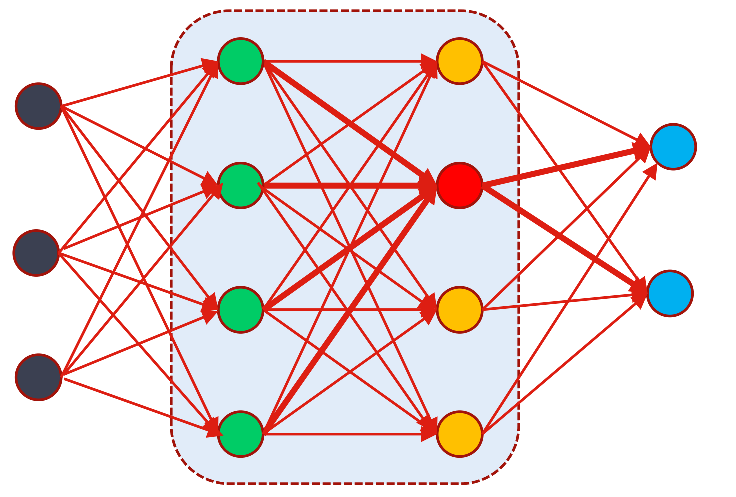
- Constraints to propagate:

  $$y^+_{l,k} = y^-_{l,k} + w^T_{l,k} y^+_{l-1,\cdot} + d_{l,k}$$

  $$y^-_{l,k} = y^+_{l,k} - w^T_{l,k} y^+_{l-1,\cdot} - d_{l,k}$$

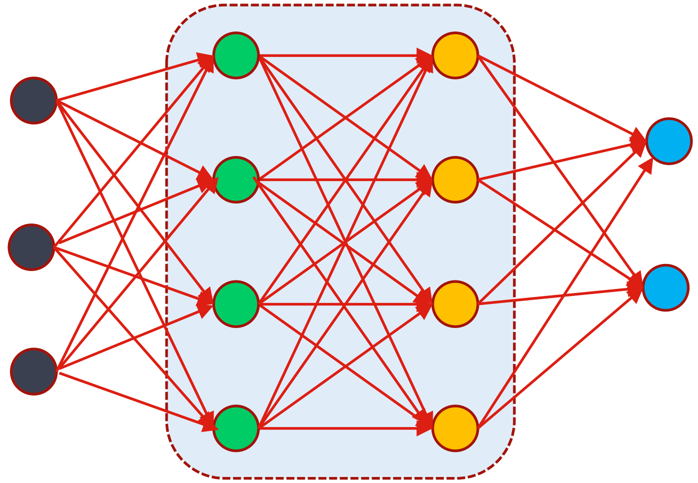
  $$y^+_{l,k} = \frac{1}{w_{l+1,i,k}} \left(y^+_{l+1,i} - y^+_{l+1,i} - \sum_{j \neq k} w_{l+1,i,j} y^+_{l,j} - d_{l+1,i} \right) \text{ for all } i$$

- 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?



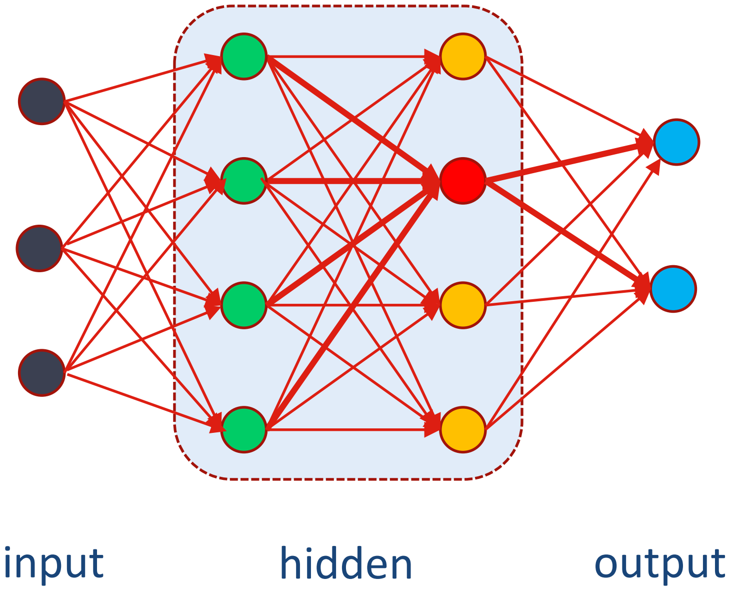
Constraint Matrix Nonzero Pattern

	input			layer 1								layer 2								output	
	x_1^{in}	x_2^{in}	x_3^{in}	$y_{2,1}^{+}$	$y_{2,2}^{+}$	$y_{2,3}^{+}$	$y_{2,4}^{+}$	$y_{2,1}^{-}$	$y_{2,2}^{-}$	$y_{2,3}^{-}$	$y_{2,4}^{-}$	$y_{3,1}^{+}$	$y_{3,2}^{+}$	$y_{3,3}^{+}$	$y_{3,4}^{+}$	$y_{3,1}^{-}$	$y_{3,2}^{-}$	$y_{3,3}^{-}$	$y_{3,4}^{-}$	y_1^{out}	y_2^{out}
layer 1	•	•	•	•				•													
	•	•	•		•				•												
	•	•	•			•				•											
	•	•	•				•				•										
layer 2				•	•	•	•					•				•					
				•	•	•	•						•				•				
				•	•	•	•							•				•			
				•	•	•	•								•				•		
output												•	•	•	•					•	
												•	•	•	•						•



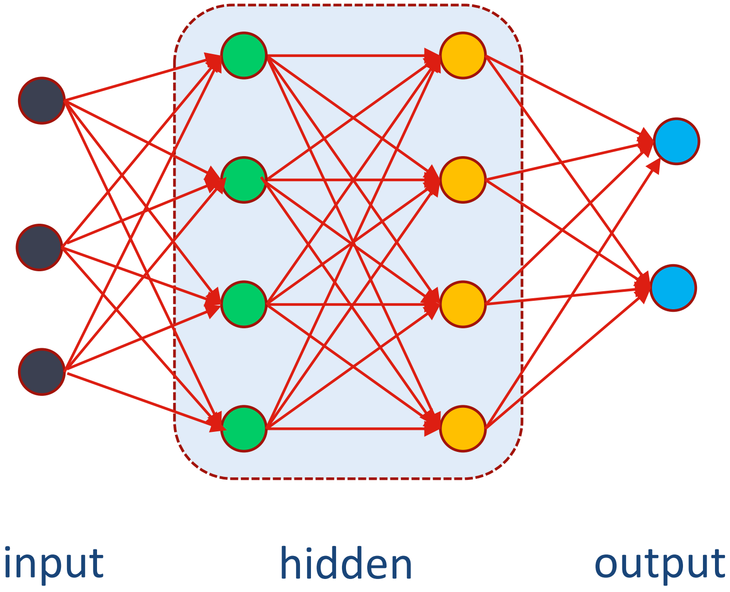
Constraint Matrix Nonzero Pattern

		input					layer 1							layer 2						output	
	x_1^{in}	x_2^{in}	x_3^{in}	$y_{2,1}^+$	$y_{2,2}^+$	$y_{2,3}^+$	$y_{2,4}^+$	$y_{2,1}^-$	$y_{2,2}^-$	$y_{2,3}^-$	$y_{2,4}^-$	$y_{3,1}^+$	$y_{3,2}^+$	$y_{3,3}^+$	$y_{3,4}^+$	$y_{3,1}^-$	$y_{3,2}^-$	$y_{3,3}^-$	$y_{3,4}^-$	y_1^{out}	y_2^{out}
layer 1	●	●	●	●				●													
	●	●	●		●				●												
	●	●	●			●				●											
	●	●	●				●				●										
layer 2				●	●	●	●					●				●					
				●	●	●	●						●				●				
				●	●	●	●							●				●			
				●	●	●	●								●				●		
output												●	●	●	●					●	
												●	●	●	●						●



Constraint Matrix Nonzero Pattern

	input			layer 1								layer 2								output	
	x_1^{in}	x_2^{in}	x_3^{in}	$y_{2,1}^+$	$y_{2,2}^+$	$y_{2,3}^+$	$y_{2,4}^+$	$y_{2,1}^-$	$y_{2,2}^-$	$y_{2,3}^-$	$y_{2,4}^-$	$y_{3,1}^+$	$y_{3,2}^+$	$y_{3,3}^+$	$y_{3,4}^+$	$y_{3,1}^-$	$y_{3,2}^-$	$y_{3,3}^-$	$y_{3,4}^-$	y_1^{out}	y_2^{out}
layer 1	•	•	•	•				•													
	•	•	•		•				•												
	•	•	•			•				•											
	•	•	•				•				•										
layer 2				•	•	•	•					•				•					
				•	•	•	•						•				•				
				•	•	•	•							•				•			
				•	•	•	•								•				•		
output												•	•	•	•					•	
												•	•	•	•						•



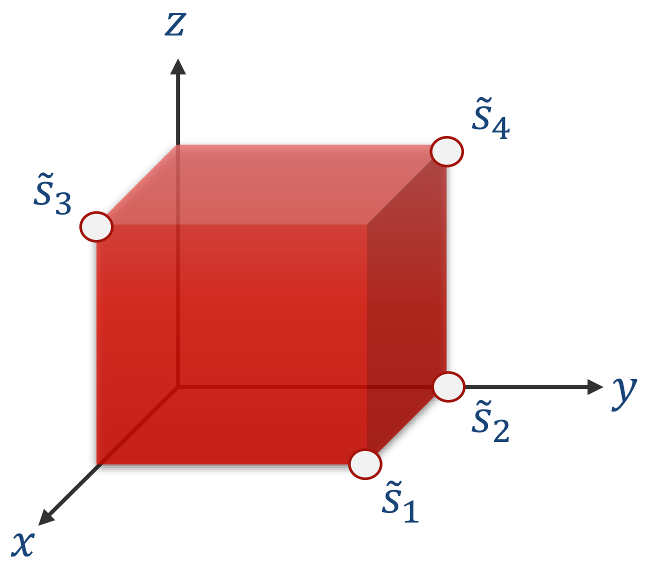
- Variables $y_{l,k}^+$ within the same layer have almost identical non-zero patterns (same for x_k^{in})
  - This layer's constraints: different pattern (but each variable only in one constraint)
  - Next layer's constraints: identical non-zero pattern, except for $w_{l,i,j} = 0$ (each var in many constraints)
- Consequence:
  - $\mathrm{p}(i,j) = \mathrm{supp}(A_{\cdot,i})^T \mathrm{supp}(A_{\cdot,j}) / (\|\mathrm{supp}(A_{\cdot,i})\| \cdot \|\mathrm{supp}(A_{\cdot,j})\|)$ is large $\Leftrightarrow$ i and j in same layer

Identifying Layers

- Clustering algorithm for vectors $v_j = \mathrm{supp}(A_{\cdot,j}) \in \{0,1\}^m$
  - 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

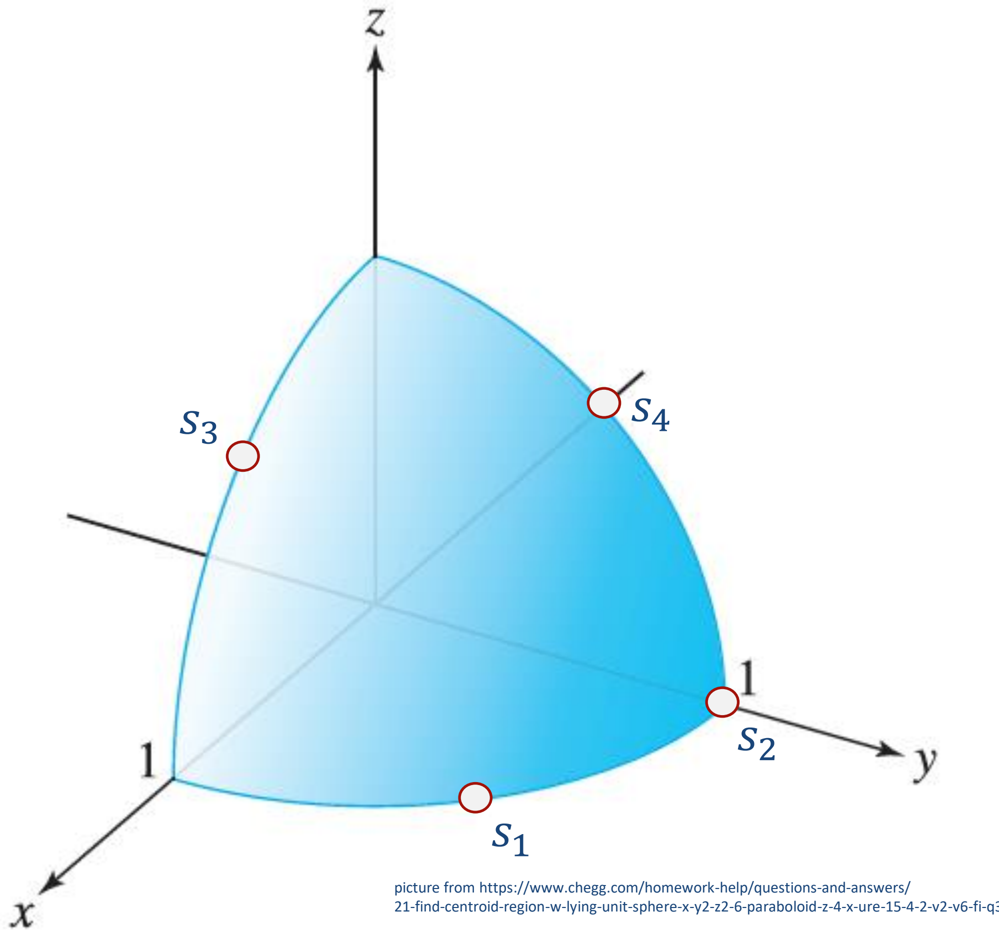
- Identify paired variables y^+, y^- with $y^+ y^- = 0$
  - Consider them to be a single variable
- Collect candidates j with at least 5 nonzeros
- Let $\tilde{s}_j = \mathrm{supp}(A_{\cdot,j}) \in \{0,1\}^m$ and $s_j = \tilde{s}_j / \|\tilde{s}_j\|$



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 $\tilde{s}_j = \text{supp}(A_{\cdot,j}) \in \{0,1\}^m$ and $s_j = \tilde{s}_j / \|\tilde{s}_j\|$

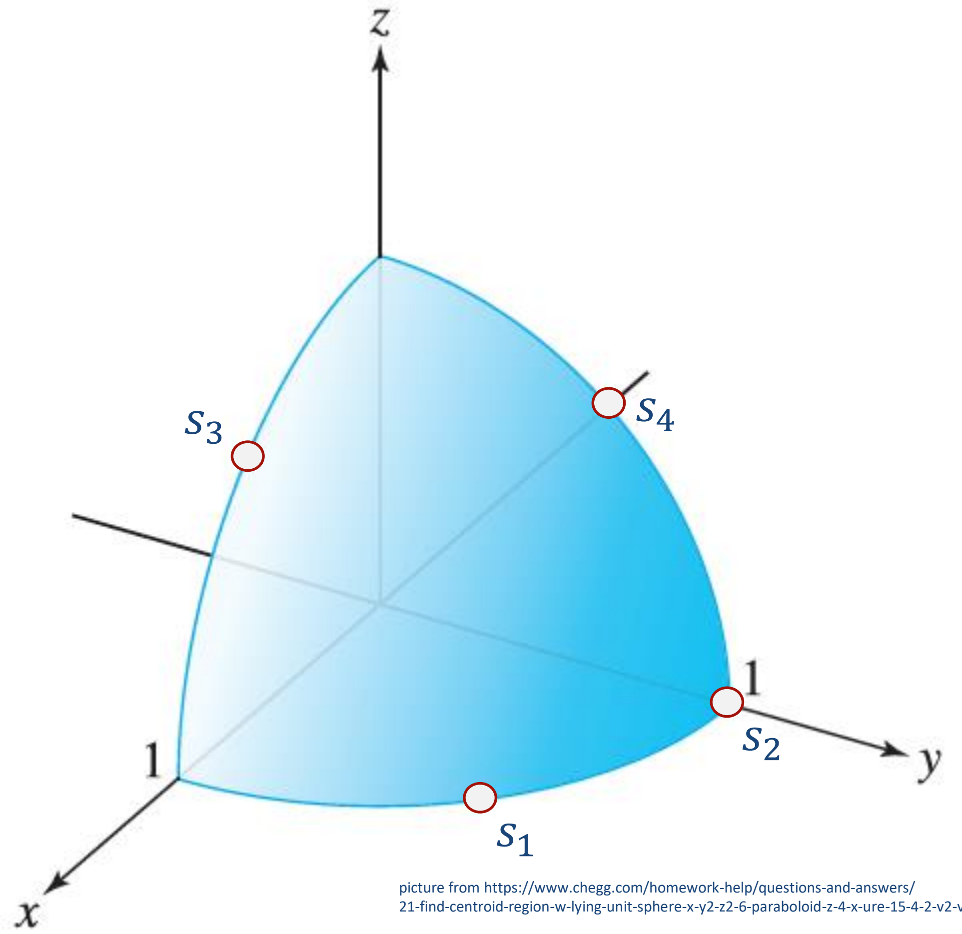


picture from https://www.chegg.com/homework-help/questions-and-answers/21-find-centroid-region-w-lying-unit-sphere-x-y2-z2-6-paraboloid-z-4-x-ure-15-4-2-v2-v6-fi-q34717404

... projected to the unit sphere in the positive orthant

Clustering Algorithm in Gurobi 10.0

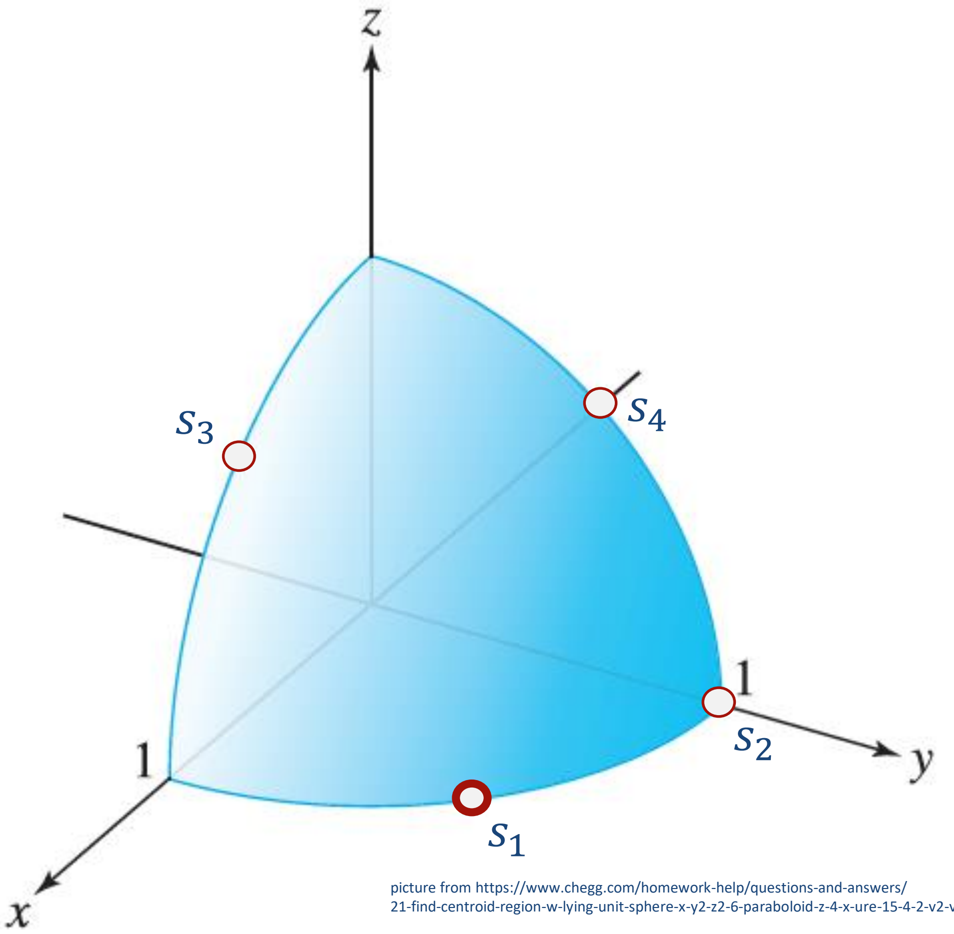
- Identify paired variables y^+, y^- with $y^+ y^- = 0$
  - Consider them to be a single variable
- Collect candidates j with at least 5 nonzeros
- Let $\tilde{s}_j = \text{supp}(A_{\cdot,j}) \in \{0,1\}^m$ and $s_j = \tilde{s}_j / \|\tilde{s}_j\|$
- Start with $\text{C} = 0$ clusters and $\delta = \varepsilon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



picture from https://www.chegg.com/homework-help/questions-and-answers/21-find-centroid-region-w-lying-unit-sphere-x-y2-z2-6-paraboloid-z-4-x-ure-15-4-2-v2-v6-fi-q34717404

Clustering Algorithm in Gurobi 10.0

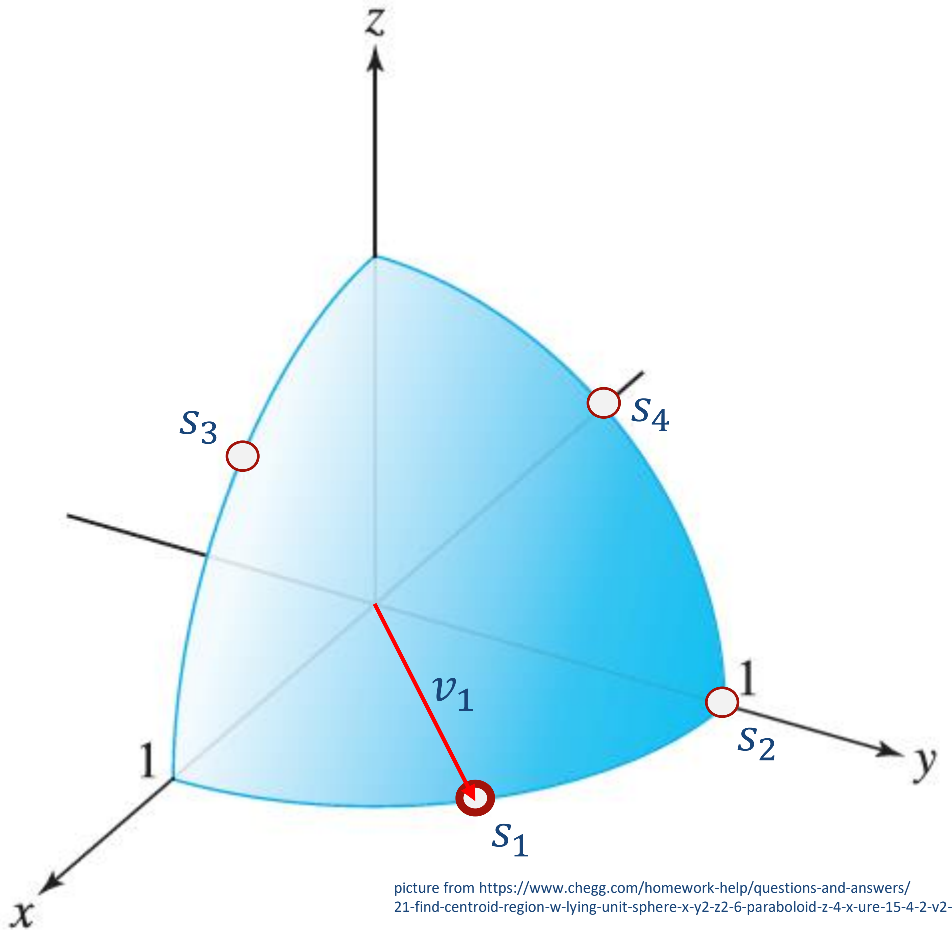
- Identify paired variables y^+, y^- with $y^+ y^- = 0$
  - Consider them to be a single variable
- Collect candidates j with at least 5 nonzeros
- Let $\tilde{s}_j = \operatorname{supp}(A_{.,j}) \in \{0,1\}^m$ and $s_j = \tilde{s}_j / \|\tilde{s}_j\|$
- Start with $\mathrm{C} = 0$ clusters and $\delta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



picture from https://www.chegg.com/homework-help/questions-and-answers/21-find-centroid-region-w-lying-unit-sphere-x-y2-z2-6-paraboloid-z-4-x-ure-15-4-2-v2-v6-fi-q34717404

Clustering Algorithm in Gurobi 10.0

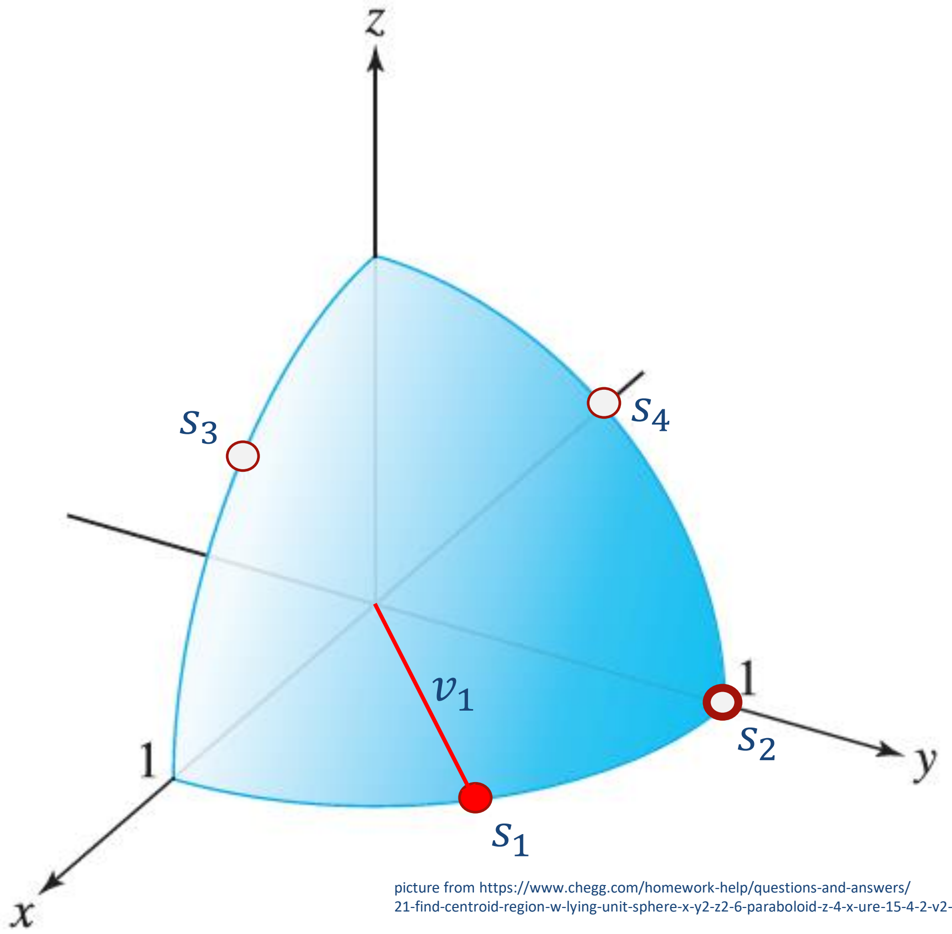
- Identify paired variables y^+, y^- with $y^+y^- = 0$
  - Consider them to be a single variable
- Collect candidates j with at least 5 nonzeros
- Let $\tilde{s}_j = \text{supp}(A_{\cdot,j}) \in \{0,1\}^m$ and $s_j = \tilde{s}_j / \|\tilde{s}_j\|$
- Start with $\mathrm{C} = 0$ clusters and $\delta = \varepsilon = 0.5$
- At most 50 times:
  - For all candidates j in random order:
    - Find closest cluster center vector v_k, if any
    - If $\mathrm{d}(j,k) := 1 - v_k^T s_j < \varepsilon$: 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



picture from https://www.chegg.com/homework-help/questions-and-answers/21-find-centroid-region-w-lying-unit-sphere-x-y2-z2-6-paraboloid-z-4-x-ure-15-4-2-v2-v6-fi-q34717404

Clustering Algorithm in Gurobi 10.0

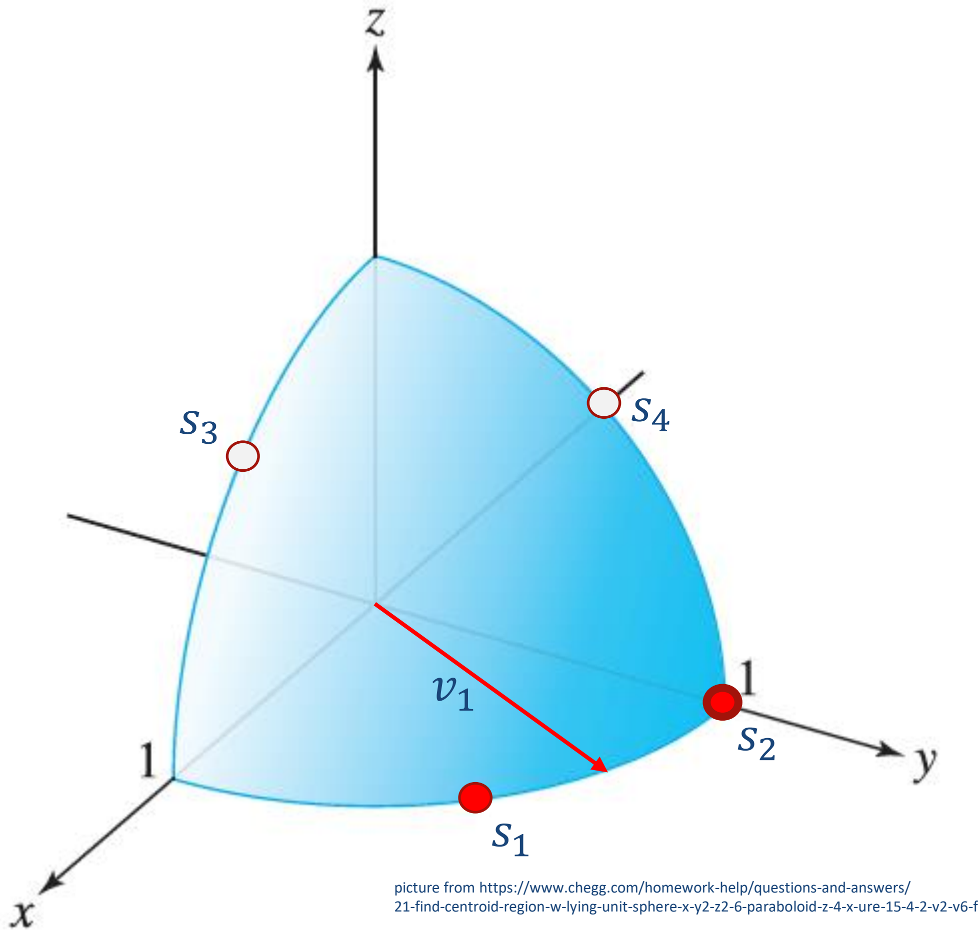
- Identify paired variables y^+, y^- with $y^+ y^- = 0$
  - Consider them to be a single variable
- Collect candidates j with at least 5 nonzeros
- Let $\tilde{s}_j = \text{supp}(A_{\cdot,j}) \in \{0,1\}^m$ and $s_j = \tilde{s}_j / \|\tilde{s}_j\|$
- Start with $\text{C} = 0$ clusters and $\delta = \varepsilon = 0.5$
- At most 50 times:
  - For all candidates j in random order:
    - Find closest cluster center vector v_k, if any
    - If $\text{d}(j,k) := 1 - v_k^T s_j < \varepsilon$: 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



picture from https://www.chegg.com/homework-help/questions-and-answers/21-find-centroid-region-w-lying-unit-sphere-x-y2-z2-6-paraboloid-z-4-x-ure-15-4-2-v2-v6-fi-q34717404

Clustering Algorithm in Gurobi 10.0

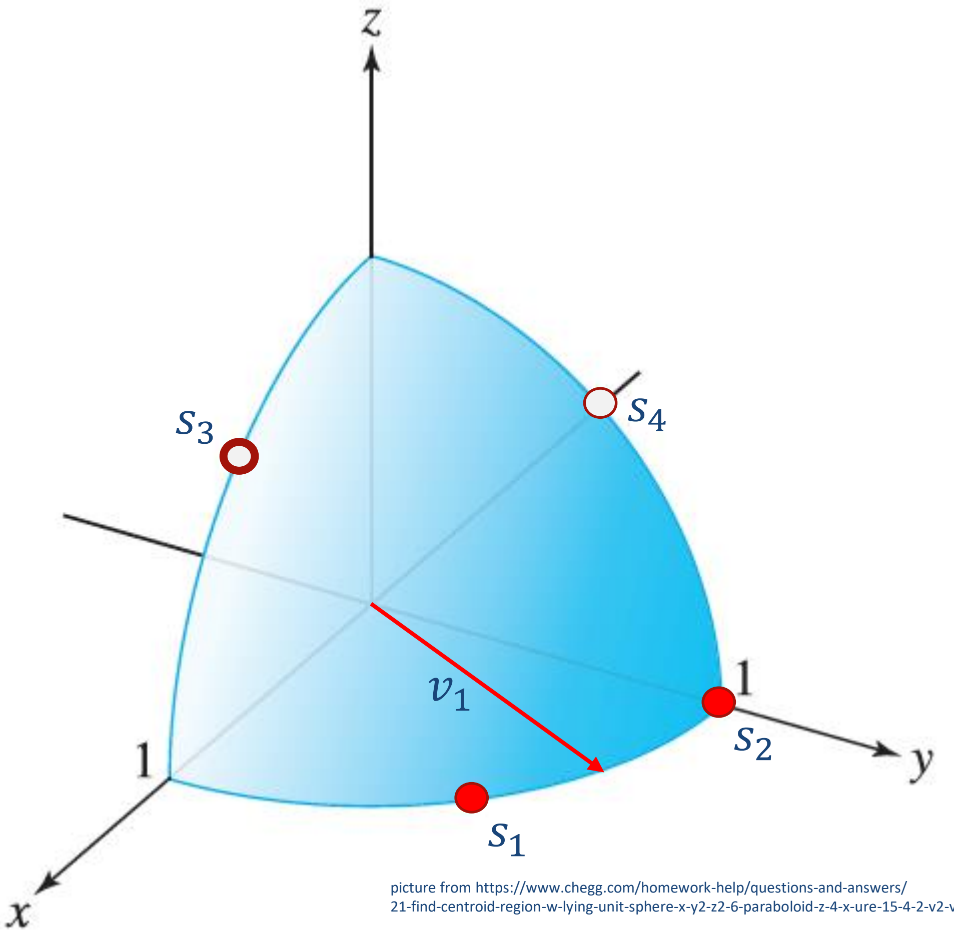
- Identify paired variables y^+, y^- with $y^+y^- = 0$
  - Consider them to be a single variable
- Collect candidates j with at least 5 nonzeros
- Let $\tilde{s}_j = \mathrm{supp}(A_{.,j}) \in \{0,1\}^m$ and $s_j = \tilde{s}_j / \|\tilde{s}_j\|$
- Start with $\mathrm{C} = 0$ clusters and $\delta = \varepsilon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



picture from https://www.chegg.com/homework-help/questions-and-answers/21-find-centroid-region-w-lying-unit-sphere-x-y2-z2-6-paraboloid-z-4-x-ure-15-4-2-v2-v6-fi-q34717404

Clustering Algorithm in Gurobi 10.0

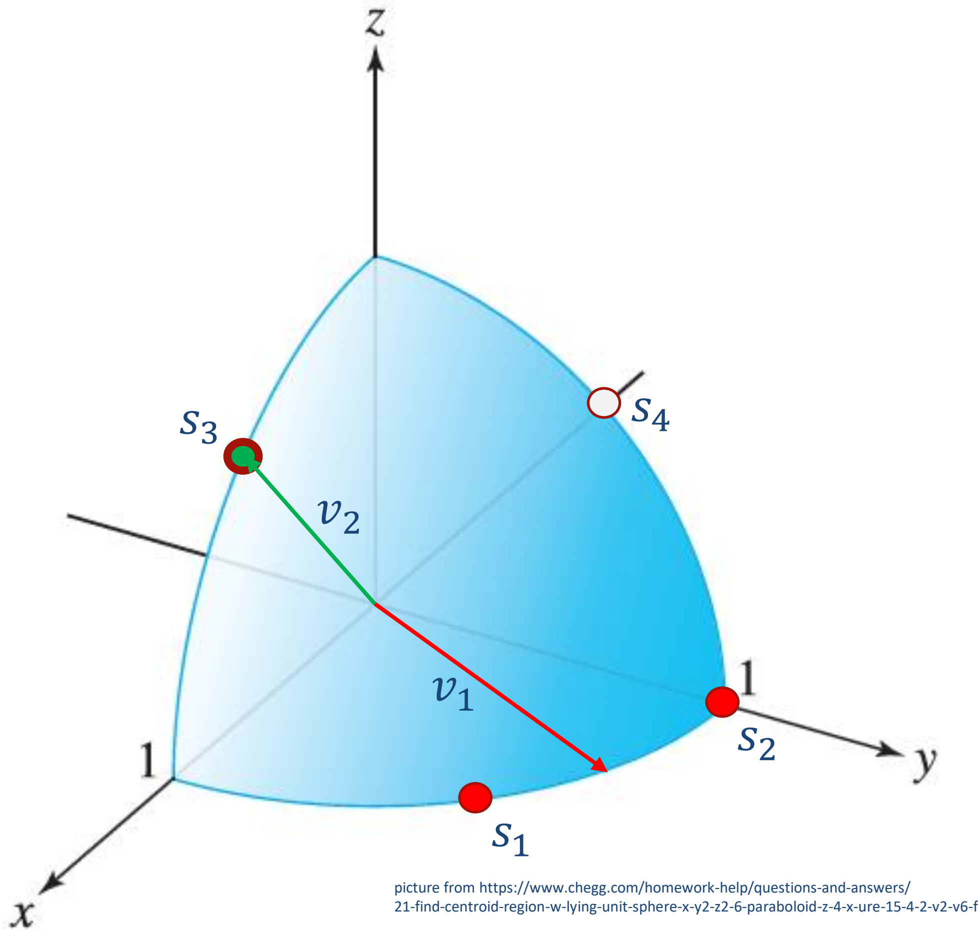
- Identify paired variables y^+, y^- with $y^+ y^- = 0$
  - Consider them to be a single variable
- Collect candidates j with at least 5 nonzeros
- Let $\tilde{s}_j = \mathrm{supp}(A_{\cdot,j}) \in \{0,1\}^m$ and $s_j = \tilde{s}_j / \|\tilde{s}_j\|$
- Start with $C = 0$ clusters and $\delta = \varepsilon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



picture from https://www.chegg.com/homework-help/questions-and-answers/21-find-centroid-region-w-lying-unit-sphere-x-y2-z2-6-paraboloid-z-4-x-ure-15-4-2-v2-v6-fi-q34717404

Clustering Algorithm in Gurobi 10.0

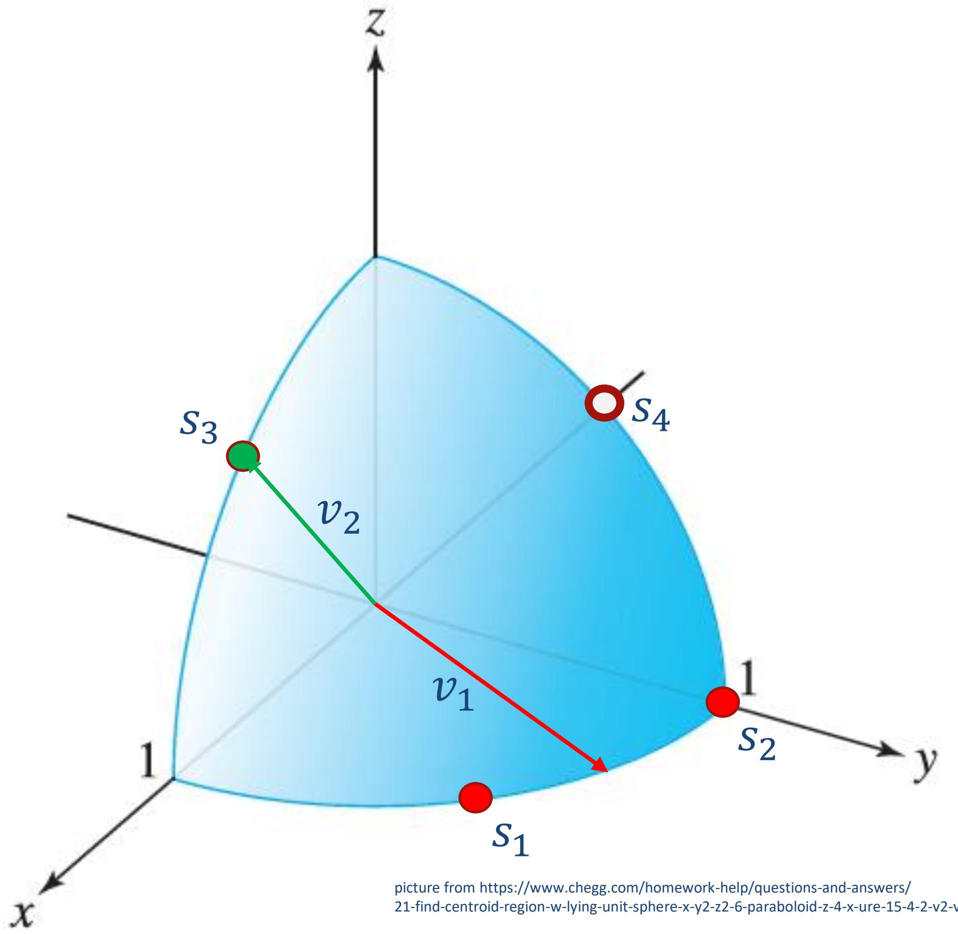
- Identify paired variables y^+, y^- with $y^+ y^- = 0$
  - Consider them to be a single variable
- Collect candidates j with at least 5 nonzeros
- Let $\tilde{s}_j = \mathrm{supp}(A_{.,j}) \in \{0,1\}^m$ and $s_j = \tilde{s}_j / \|\tilde{s}_j\|$
- Start with $\mathrm{C} = 0$ clusters and $\delta = \varepsilon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



picture from https://www.chegg.com/homework-help/questions-and-answers/21-find-centroid-region-w-lying-unit-sphere-x-y2-z2-6-paraboloid-z-4-x-ure-15-4-2-v2-v6-fi-q34717404

Clustering Algorithm in Gurobi 10.0

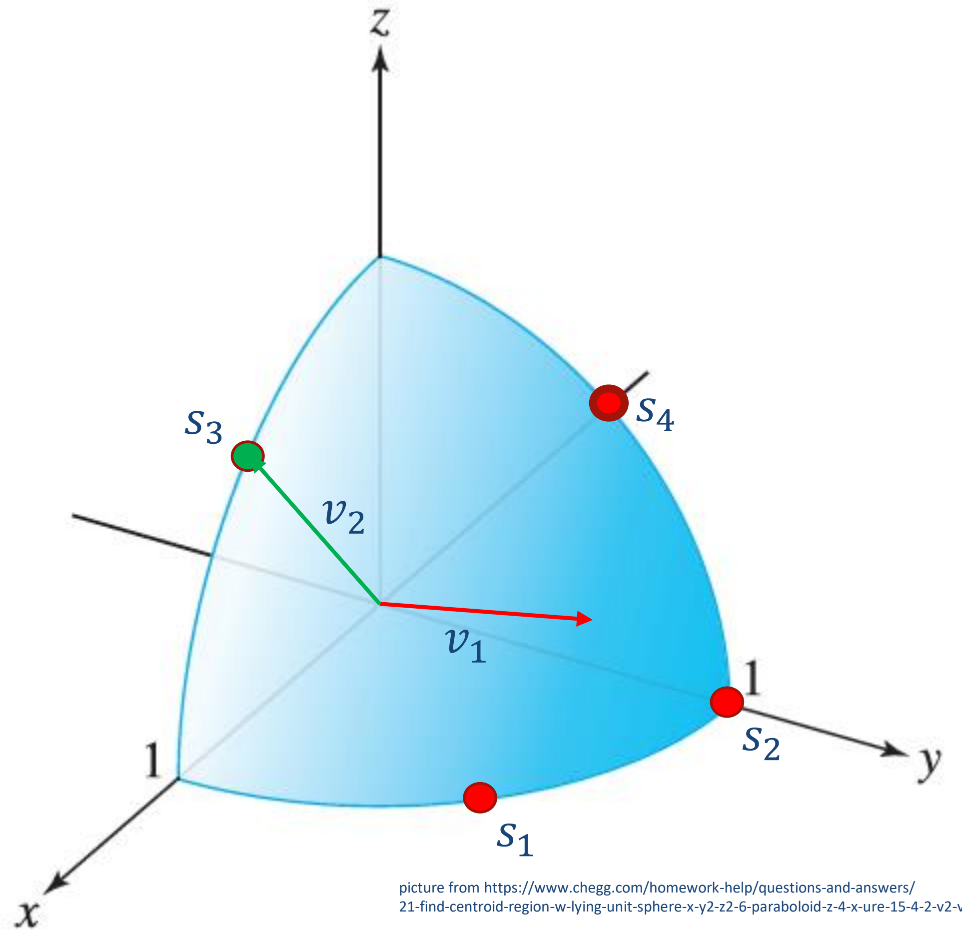
- Identify paired variables y^+, y^- with $y^+ y^- = 0$
  - Consider them to be a single variable
- Collect candidates j with at least 5 nonzeros
- Let $\tilde{s}_j = \mathrm{supp}(A_{\cdot,j}) \in \{0,1\}^m$ and $s_j = \tilde{s}_j / \|\tilde{s}_j\|$
- Start with $\mathrm{C} = 0$ clusters and $\delta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



picture from https://www.chegg.com/homework-help/questions-and-answers/21-find-centroid-region-w-lying-unit-sphere-x-y2-z2-6-paraboloid-z-4-x-ure-15-4-2-v2-v6-fi-q34717404

Clustering Algorithm in Gurobi 10.0

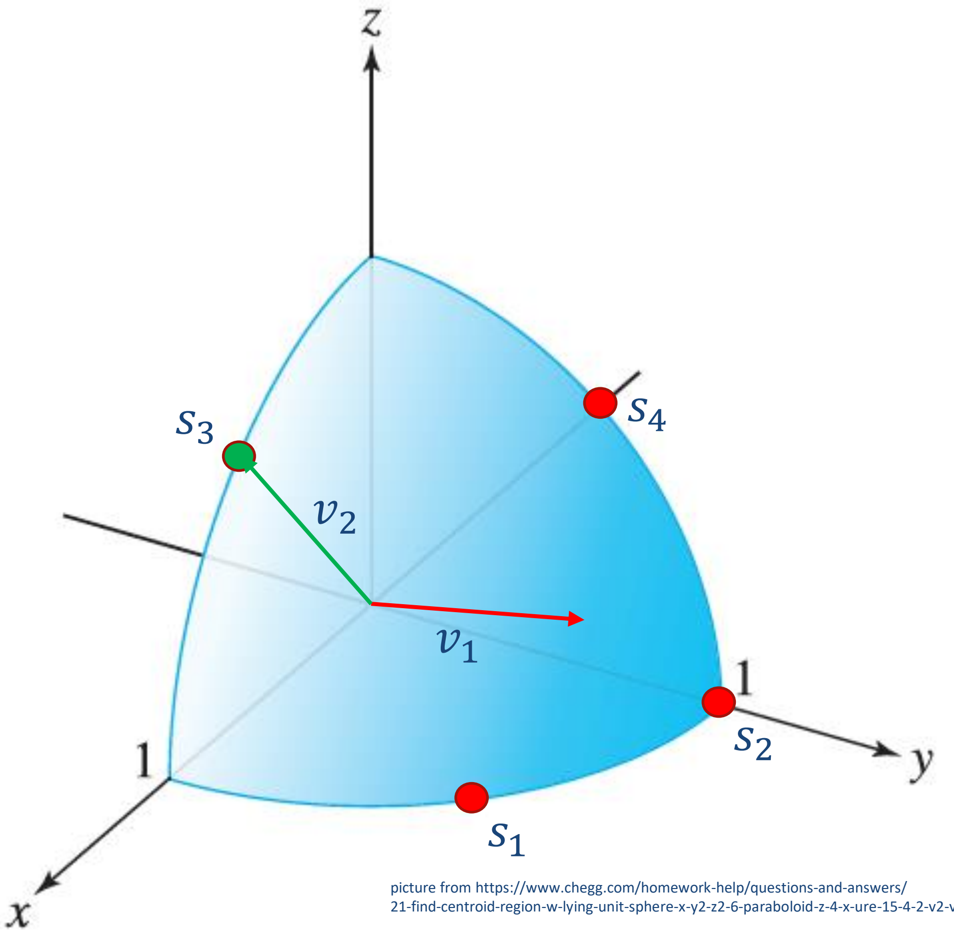
- Identify paired variables y^+, y^- with $y^+y^- = 0$
  - Consider them to be a single variable
- Collect candidates j with at least 5 nonzeros
- Let $\tilde{s}_j = \mathrm{supp}(A_{\cdot,j}) \in \{0,1\}^m$ and $s_j = \tilde{s}_j / \|\tilde{s}_j\|$
- Start with $\mathrm{C} = 0$ clusters and $\delta = \varepsilon = 0.5$
- At most 50 times:
  - For all candidates j in random order:
    - Find closest cluster center vector v_k, if any
    - If $\mathrm{d}(j,k) := 1 - v_k^T s_j < \varepsilon$: 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



picture from https://www.chegg.com/homework-help/questions-and-answers/21-find-centroid-region-w-lying-unit-sphere-x-y2-z2-6-paraboloid-z-4-x-ure-15-4-2-v2-v6-fi-q34717404

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 $\tilde{s}_j = \mathrm{supp}\left(A_{\cdot,j}\right) \in \{0,1\}^m$ and $s_j = \tilde{s}_j / \left\|\tilde{s}_j\right\|$
- Start with $\mathrm{C} = 0$ clusters and $\delta = \varepsilon = 0.5$
- At most 50 times:
  - For all candidates j in random order:
    - Find closest cluster center vector v_k, if any
    - If $\mathrm{d}(j,k) := 1 - v_k^T s_j < \varepsilon$: 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 $\varepsilon := 0.98\varepsilon$
  - If all $s_j^T v_{k_j} > 0.7$: stop (success)
  - If all clusters have less than 10 variables: stop (fail)



picture from https://www.chegg.com/homework-help/questions-and-answers/21-find-centroid-region-w-lying-unit-sphere-x-y2-z2-6-paraboloid-z-4-x-ure-15-4-2-v2-v6-fi-q34717404

Performance Evaluation

© 2023 Gurobi Optimization, LLC. Confidential, All Rights Reserved | 45

Benchmarks: Test Set

- Goldstein-Price and Peak2d: 60 instances each,
  - Approximation of a nonlinear function with a neural network
  - #layers ∈ $\{2,3\}$ of #neurons ∈ $\{56,128,256\}$ 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 ∈ $\{2,3\}$ of #neurons ∈ $\{128,256\}$ 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 ∈ $\{50,100\}$ and 6 layers of 500 neurons, 30 models each
  - Tensorflow: #layers ∈ $\{2,3\}$ of #neurons ∈ $\{50,100,200\}$, 20 models each

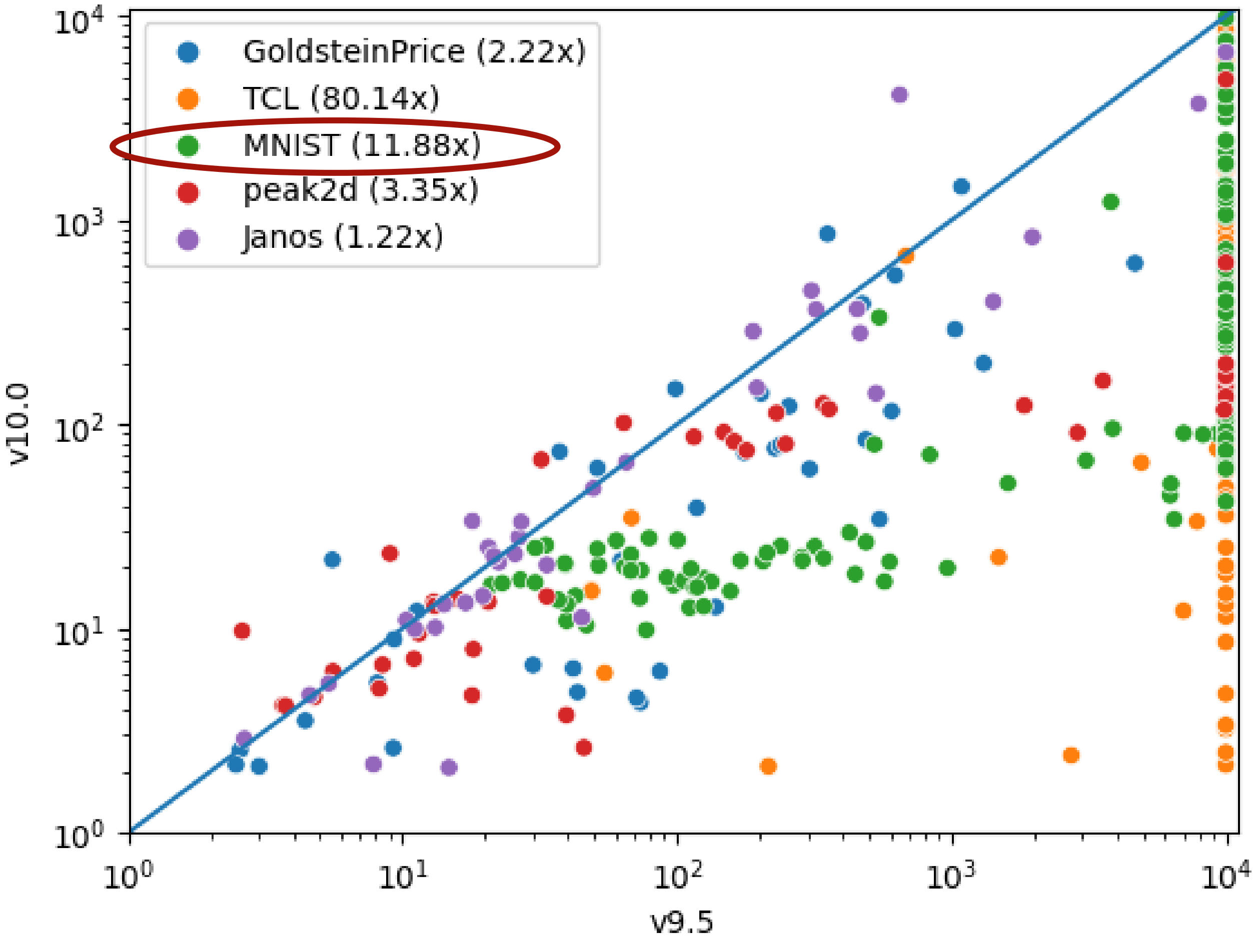
Computational Setup

- 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

		Gurobi 9.5		Gurobi 10.0		
test set	# models	% solved	time	% solved	time	speedup
GoldsteinPrice	43	100%	55	100%	24	**2.2x**
Peak2d	41	83%	120	100%	35	**3.3x**
Janos	38	97%	48	100%	39	**1.2x**
TCL	65	23%	5130	100%	63	**80.1x**
MNIST	136	47%	1614	100%	135	**11.9x**

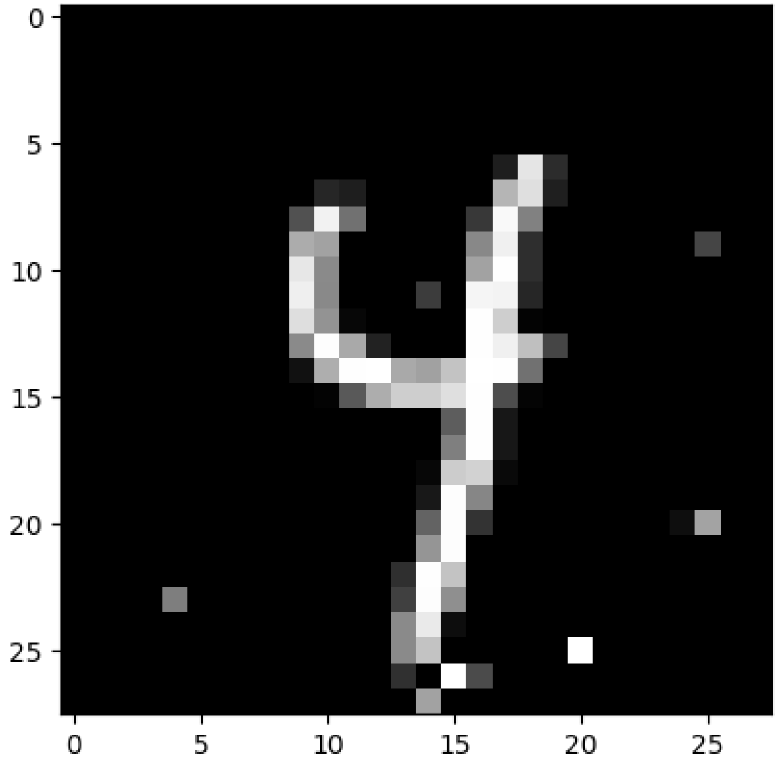
Gurobi 9.5 vs Gurobi 10.0



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

	# layers	size	# models	Gurobi 9.5 % solved	Gurobi 9.5 time	Gurobi 10.0 % solved	Gurobi 10.0 time	speedup
Keras	2	50	20	100%	21	100%	3	5.4x
		100	20	95%	404	100%	20	19.3x
		200	20	50%	2764	95%	28	94.5x
	3	50	20	95%	197	100%	7	24.6x
		100	20	35%	4977	95%	75	65.7x
		200	20	15%	9272	95%	105	87.6x
scikit-learn	2	50	30	100%	17	100%	12	1.4x
		100	30	93%	511	100%	66	7.7x
	6	500	30	0%	10000	0%	10000	–

Adversarial Model: Fischetti/Jo Test Set

Time limit as in Fischetti/Jo: 300 sec

basic models		Gurobi 9.5	CPLEX 12.7*
network layers	# mod	% solved	% solved
8-8-8	99	100%	100%
8-8-8-8-8-8	99	100%	97%
20-10-8-8	99	100%	64%
20-10-8-8-8	99	99%	24%
20-20-10-10-10	100	27%	7%

Fischetti/Jo MIP-OBBT		Gurobi 9.5	CPLEX 12.7*
network layers	# mod	% solved	% solved
8-8-8	99	100%	100%
8-8-8-8-8-8	99	100%	100%
20-10-8-8	99	100%	100%
20-10-8-8-8	99	100%	98%
20-20-10-10-10	100	89%	67%

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

	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

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

GUROBI OPTIMIZATION

Time limit as in Fischetti/Jo: 300 sec

basic models		Gurobi 9.5	Gurobi 10.0	
network layers	# mod	% solved	% solved	speedup
8-8-8	99	100%	100%	1.7x
8-8-8-8-8-8	99	100%	100%	2.4x
20-10-8-8	99	100%	100%	3.2x
20-10-8-8-8	99	99%	100%	4.0x
20-20-10-10-10	100	27%	88%	4.2x

Fischetti/Jo MIP-OBBT		Gurobi 9.5	Gurobi 10.0	
network layers	# mod	% solved	% solved	speedup
8-8-8	99	100%	100%	1.0x
8-8-8-8-8-8	99	100%	100%	1.0x
20-10-8-8	99	100%	100%	1.1x
20-10-8-8-8	99	100%	100%	1.3x
20-20-10-10-10	100	89%	90%	1.2x

- 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

basic models		Gurobi 9.5	Gurobi 10.0		Gurobi 10.0, OBBT=3	
network layers	# mod	% solved	% solved	speedup	% solved	speedup
8-8-8	99	100%	100%	1.7x	100%	0.8x
8-8-8-8-8-8	99	100%	100%	2.4x	100%	2.2x
20-10-8-8	99	100%	100%	3.2x	100%	4.1x
20-10-8-8-8	99	99%	100%	4.0x	100%	4.8x
20-20-10-10-10	100	27%	88%	4.2x	95%	6.5x

Fischetti/Jo MIP-OBBT		Gurobi 9.5	Gurobi 10.0		Gurobi 10.0, OBBT=3	
network layers	# mod	% solved	% solved	speedup	% solved	speedup
8-8-8	99	100%	100%	1.0x	100%	0.6x
8-8-8-8-8-8	99	100%	100%	1.0x	100%	0.9x
20-10-8-8	99	100%	100%	1.1x	100%	1.2x
20-10-8-8-8	99	100%	100%	1.3x	100%	1.5x
20-20-10-10-10	100	89%	90%	1.2x	96%	2.5x

- 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

basic models		Gurobi 9.5	Gurobi 10.0		Gurobi 10.0, OBBT=3		dev version		dev version, OBBT=3	
network layers	# mod	% solved	% solved	speedup	% solved	speedup	% solved	speedup	% solved	speedup
8-8-8	99	100%	100%	1.7x	100%	0.8x	100%	1.6x	100%	0.6x
8-8-8-8-8-8	99	100%	100%	2.4x	100%	2.2x	100%	2.5x	100%	2.3x
20-10-8-8	99	100%	100%	3.2x	100%	4.1x	100%	3.1x	100%	4.4x
20-10-8-8-8	99	99%	100%	4.0x	100%	4.8x	100%	4.2x	100%	5.6x
20-20-10-10-10	100	27%	88%	4.2x	95%	6.5x	92%	4.5x	98%	7.9x

Fischetti/Jo MIP-OBBT		Gurobi 9.5	Gurobi 10.0		Gurobi 10.0, OBBT=3		dev version		dev version, OBBT=3	
network layers	# mod	% solved	% solved	speedup	% solved	speedup	% solved	speedup	% solved	speedup
8-8-8	99	100%	100%	1.0x	100%	0.6x	100%	1.0x	100%	0.5x
8-8-8-8-8-8	99	100%	100%	1.0x	100%	0.9x	100%	1.0x	100%	0.9x
20-10-8-8	99	100%	100%	1.1x	100%	1.2x	100%	1.2x	100%	1.3x
20-10-8-8-8	99	100%	100%	1.3x	100%	1.5x	100%	1.2x	100%	1.6x
20-20-10-10-10	100	89%	90%	1.2x	96%	2.5x	87%	1.3x	99%	3.5x

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?

GUROBI
OPTIMIZATION

Thank You!