Pierre Bonami
Gurobi R&D
bonami@gurobi.com

GUROBI
OPTIMIZATION

Gurobi Machine Learning

Using Trained Machine Learning Predictors in Gurobi

Agenda

Motivating Example

gurobi-machinelearning

Related improvements in Gurobi Optimizer 10.0

Motivating example

- Selling avocados in the US
  - Market is split in regions R
  - Total supply S
  - Want to decide shipment to each region
  - Maximizing revenue:
    (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.

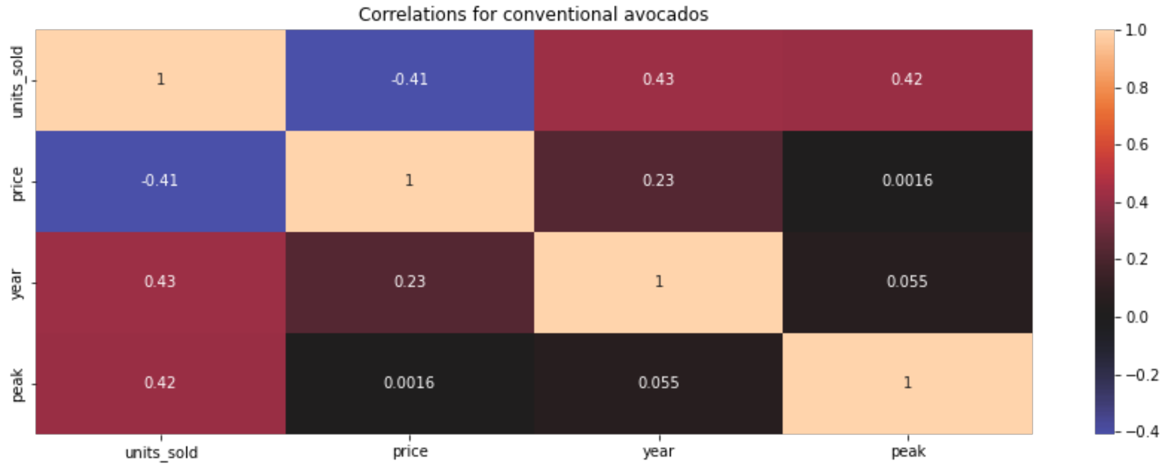
Motivating Example: Estimating Demand

- Historical data of avocado sales from Hass Avocado Board (HAB) available on Kaggle and HAB website
- Features correlated to demand: price, region, year, peak season
- Linear regression gives reasonably good prediction of demand with those:

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

- In the case of linear regression, g is an affine function

$$d = \phi^T(p, r, year, season) + \phi_0$$



Motivating Example: Price Optimization

- In a more complex problem, we optimize the price p_r
- To do so, we need to insert
  $d = g(p, r, year, season)$
  in the optimization problem
- d becomes a variable for the optimization
- Notebook developed by J. Yurchisin and R. Swamy

Motivating Example: Optimization Model

- Constants
  - R set of regions
  - c_r transportation costs
  - w penalty for waste
  - S total supply
  - Year and season of interest
- Variables
  - p_r selling price per unit
  - d_r demand
  - u total unsold products

$\max \sum_r (p_r - c_r) d_r - w * u$ *(maximize revenue)*

$s.t.$

$\sum_r d_r + u = S$ *(allocate supply)*

$d_r = g(p_r, r, year, season)$ for r $\in R$ *(define demand with regression model)*

Motivating Example: Conclusions

- Resulting model non-convex QP solved fast with Gurobi
- But now what if we need a more accurate prediction with a more complex regression:
  - Decision tree, Neural network, …
- Our goals:
  1. Simplify the process of importing a trained machine learning model built with a popular ML package into an optimization model.
  2. Improve algorithmic performance to enable the optimization model to explore a sizable space of solutions that satisfy the variable relationships captured in the ML model.

Definitions and scope

GUROBI OPTIMIZATION

- In an optimization model we want to formulate $g(x) = y$
  - g prediction function for trained regression model
  - x input variables for the regression
  - 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 OPTIMIZATION

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 1.1.0 recently released

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 Continued

- Constants
  - R set of regions
  - c_r transportation costs
  - w penalty for waste
  - S total supply
  - Year and season of interest
- Variables
  - p_r selling price per unit
  - d_r demand
  - u total unsold products

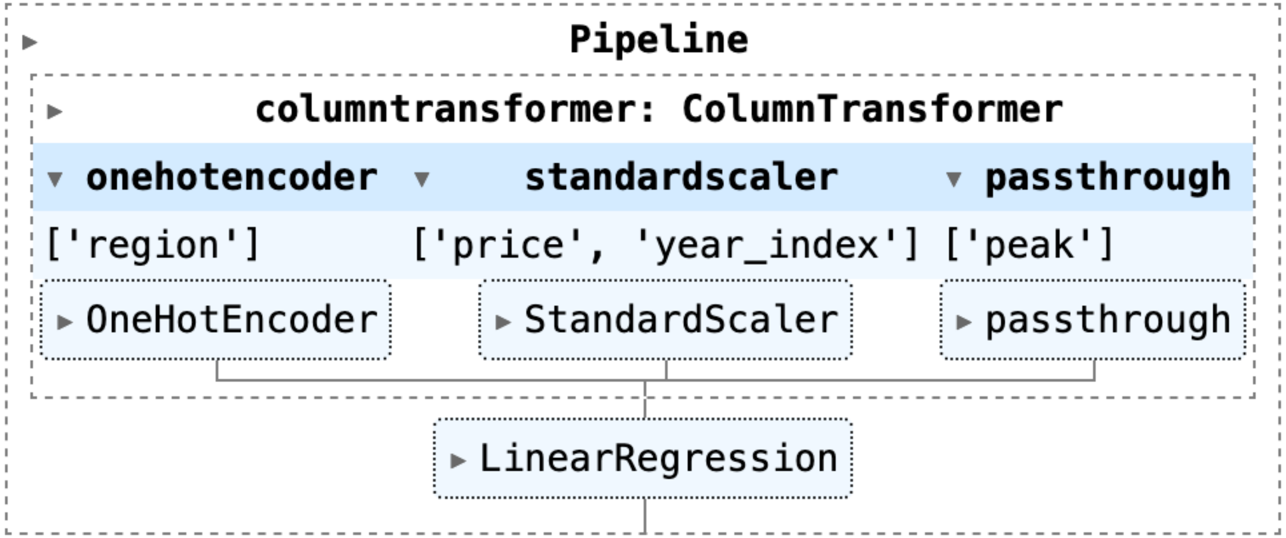
$\max \sum_r (p_r - c_r) d_r - w * u$ *(maximize revenue)*

$s.t.$

$\sum_r d_r + u = S$ *(allocate supply)*

$d_r = g(p_r, r, year, season)$ for $r \in R$ *(define demand with regression model)*

Example: Regression Model with sklearn



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

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()
```

- 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

$\max \sum_r (p_r - c_r) x_r - w * u$ *(maximize revenue)*

$s.t.$

$\sum_r d_r + u = S,$ *(allocate supply)*

```
m.setObjective(((p – c) * d).sum() - w * u, GRB.MAXIMIZE)

m.addConstr(d.sum() + u == S)
```

Example: Input of regression constraints

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

```
feats = pd.DataFrame(
    data={
        "year": 2020,
        "peak": 1,
        "region": regions,
    },
    index=regions)
feats = pd.concat(
[feats, p],
 axis=1)
```

	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

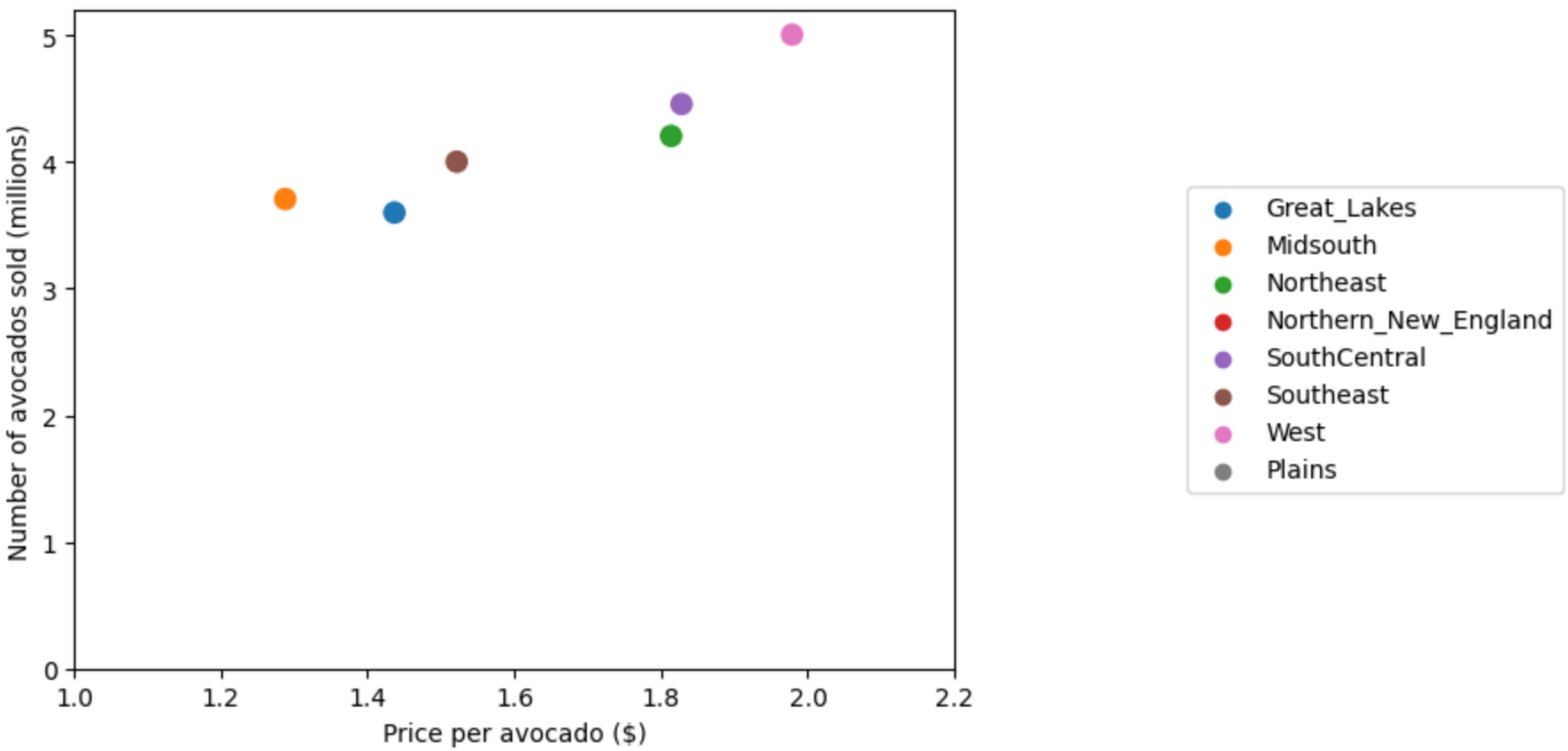
```
m.Params.NonConvex = 2
m.optimize()
```

```
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
```

Remarks

- Function `add_predictor_constr` creates the formulation for regression model and returns a *modeling object*.
- Can query statistic about modeling object, remove it, query solution after solve and error in solutions.
- If input of `add_predictor_constr` has several rows, introduce one corresponding model for each row
- Models for logistic regression use a piecewise linear approximation and can have modeling error (controlled by parameters).
- Models for decision tree, can also introduce small errors at threshold values of node splitting (can be controlled).

Comparison of models for price optimization

	R2 test	R2 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

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, GBT, Random Forests

References:

- Bergman et al. 2019, Maragno et al. 2021. Schweidtmann, Mitsos 2018, 2021, Leyffer et al. 2022.

Gurobi 10 Enhancements and Performance

Gurobi 10.0

Relevant improvements for models with ML predictor constraints

- Logistic general function
- Optimization Based Bound Tightening

Logistic General Constraint

$$p(x) = \frac{1}{1 + e^{-(x)}}$$

Function constraints in Gurobi

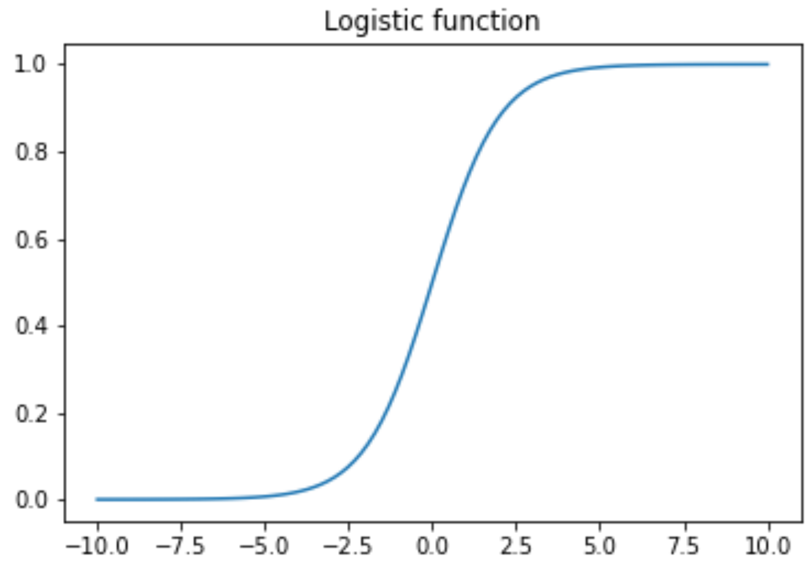
Allow to state $y = f(x)$

- f is a predefined function
- y and x are one-dimensional variables

Gurobi automatically performs a piecewise-linear approximation of f in the domain of x.

Added logistic function to our set of predefined f.

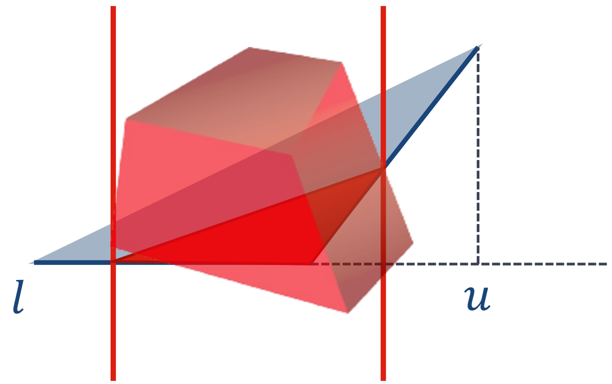
In Gurobi ML by default construct approximation with 10^{-2} maximal error.



Optimization Based Bound Tightening

- Common technique for MINLP solvers
- Given LP relaxation of a (non-convex) MI(NL)P.
- For each variable x
  - Minimize/maximize x over relaxation
  - Use optimal value as lower/upper bound for x
- Tighten coefficients of relaxation using new bounds
- Added in Gurobi 10:
  - For non-convex MIQCP: 14% av. improvement
  - Also, for MIP/MIQP/MIQCP: 1% av. Improvement on affected model

e.g.: $\operatorname{conv}(y = \max(x, 0) : l \leq x \leq u)$

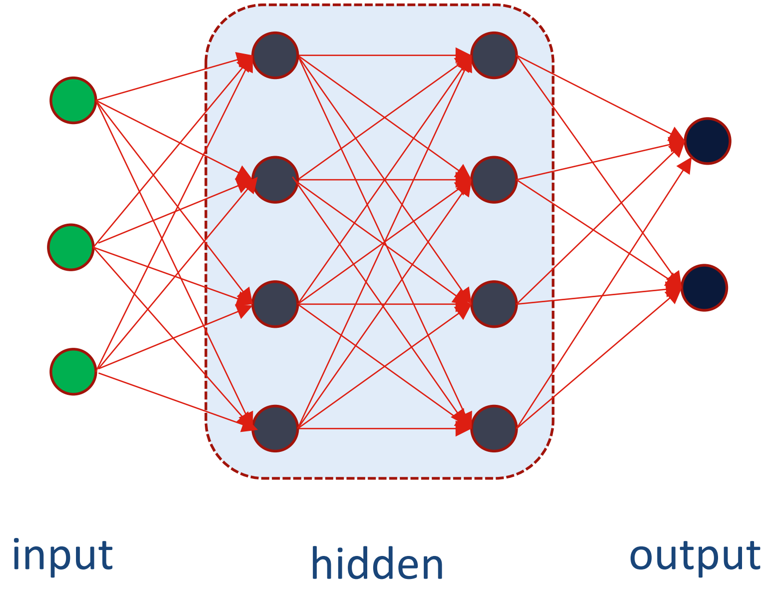


Why OBBT for NN network with ReLU

- Each neuron k has the following constraints/variables:

$$y_{\text{mix}} = w^T x_{in} + w_0$$
$$y_{out} = \max(y_{mix}, 0)$$

- The $\max$ function is nonlinear and formulated using a binary variable and big-M constraints
- Tightness of the formulation (M) depends on bounds that can be inferred for y_{mix}
- Known as essential for adversarial NN (e.g., Fischetti, Jo 2017, Weng et.al. 2018)



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 network 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 model, 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: 220 instances
  - scikit-learn: #layers 2 of #neurons ∈ $\{50,100\}$ and 6 layers of 500 neurons
  - Tensorflow: #layers ∈ $\{2,3\}$ of #neurons ∈ $\{50,100,200\}$

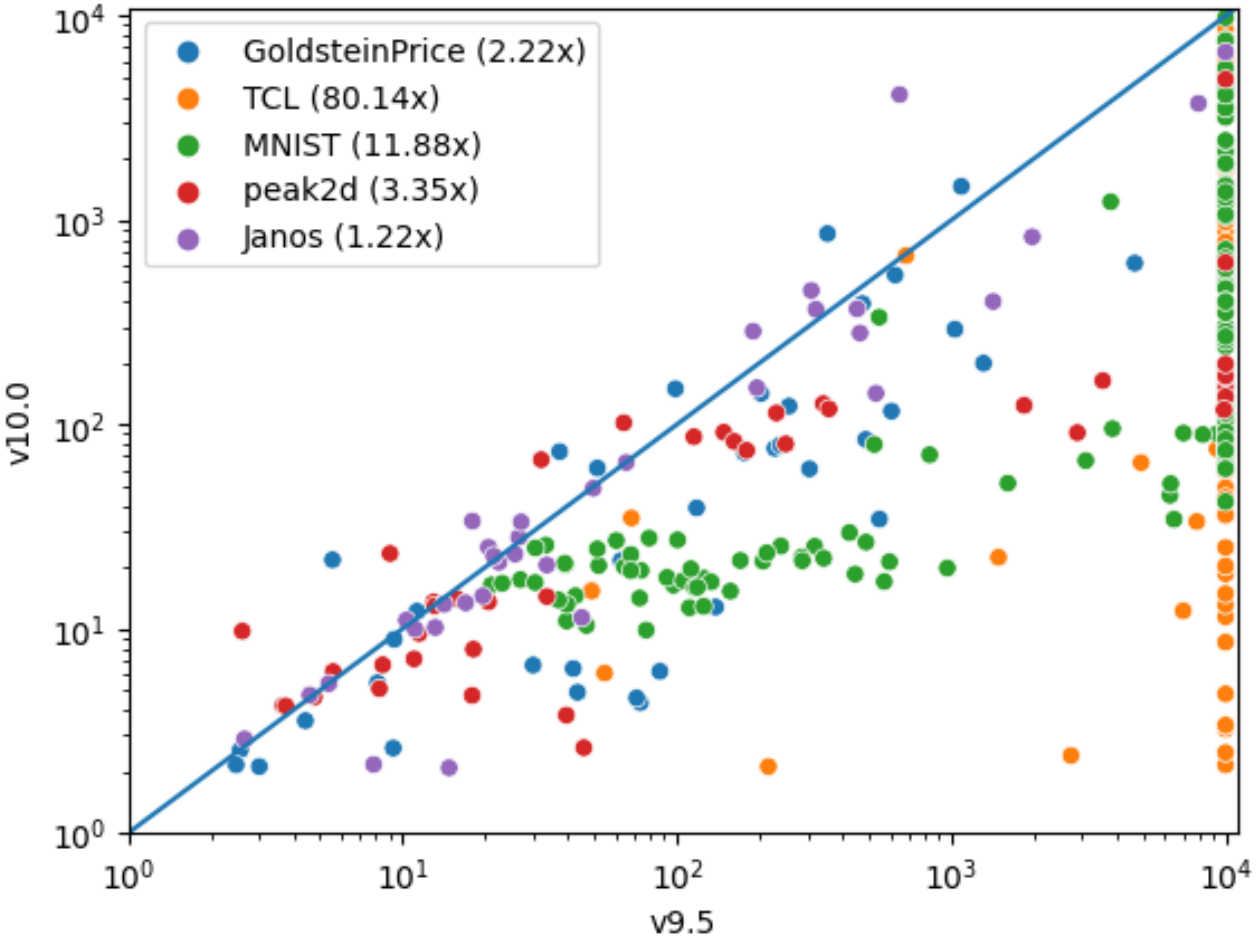
Computational Setup

- Models solved on Intel(R) Xeon(R) CPU E3-1240 CPUs, 4 cores, 4 threads
- 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

Gurobi 9.5 vs Gurobi 10.0

		V9.5		V10.0		
Models	# 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	**>80.1**
MNIST	136	47%	1614	100%	135	**>11.9x**

Gurobi 9.5 vs Gurobi 10.0



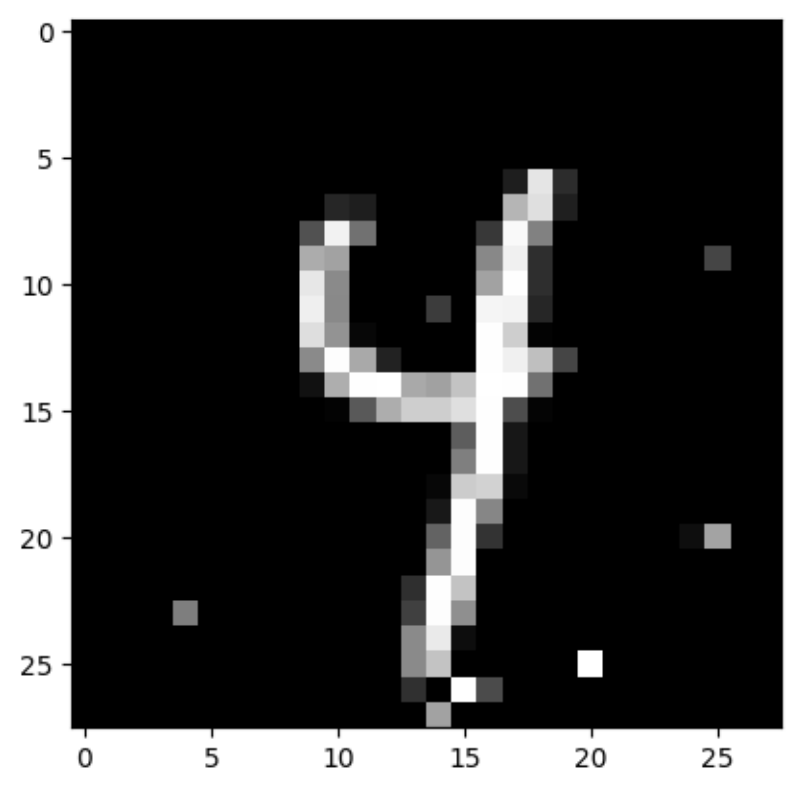
Adversarial Machine Learning

Given a trained neural network and one training example $\overline{x}$

In a small neighborhood of $\overline{x}$ show that either

- Everything is classified like training example, or
- Find a misclassified counter-example

See Fischetti, Jo 2017, Kouvaros Lomuscio 2018



Adversarial Model: Detailed Results

				V9.5		V10.0		
	# Layers	Size	# models	% solved	Time	% solved	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	---

Conclusions

Gurobi Machine Learning:

https://github.com/Gurobi/gurobi-machinelearning

Input very welcome

Performance for models with Neural Networks in Gurobi 10

Dangers and Pitfalls

ML models we can hope to handle is still limited

Methodological questions:

- How to make sure that optimization doesn't misuse results of the predictor?
- How to decide which prediction model to use?

