Strong mixed-integer programming formulations for trained neural networks

Joey Huchette[1]

with Ross Anderson[2], Will Ma[4], Christian Tjandraatmadja[2], and Juan Pablo Vielma[2,3]

[1]Rice University, Computational and Applied Mathematics
[2]Google Research, Operations Research Team
[3]MIT, Sloan School of Management
[4]Columbia University, Columbia Business School

Research at Google

Strong mixed-integer programming formulations for trained neural networks

Joey Huchette[1]

with Ross Anderson[2], Will Ma[4], Christian Tjandraatmadja[2], and Juan Pablo Vielma[2,3]
and Yeesian Ng[5] and Ondrej Sykora[2]

[1]Rice University, Computational and Applied Mathematics
[2]Google Research, Operations Research Team
[3]MIT, Sloan School of Management
[4]Columbia University, Columbia Business School
[5]MIT, Operations Research Center

Research at Google

Strong mixed-integer programming formulations for trained neural networks

Joey Huchette[1]

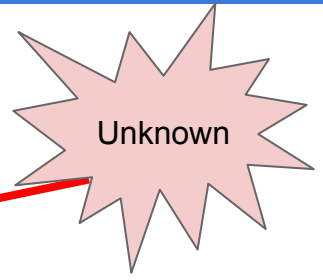
with Ross Anderson[2], Will Ma[4], Christian Tjandraatmadja[2], and Juan Pablo Vielma[2,3]
and Yeesian Ng[5] and Ondrej Sykora[2]
and Craig Boutilier[6], Martin Mladenov[6], and Moonkyung Ryu[6]

[1]Rice University, Computational and Applied Mathematics
[2]Google Research, Operations Research Team
[3]MIT, Sloan School of Management
[4]Columbia University, Columbia Business School
[5]MIT, Operations Research Center
[6]Google Research, MudCats

Research at Google

The Problem

Optimize



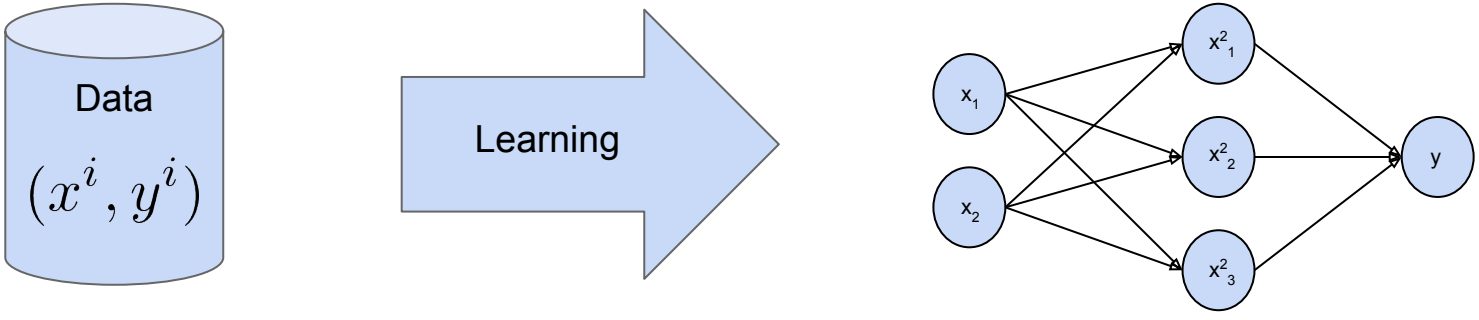
$$\begin{aligned} \text{maximize} \quad & f(x) \\ \text{such that} \quad & x \in \Omega \end{aligned}$$

End goal is ***optimization***: Make the best possible decision $x^* \in \Omega$

Predict,

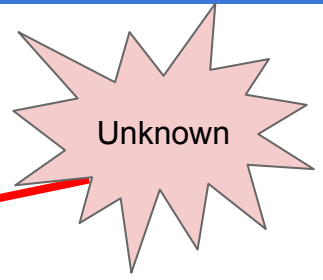
Supervised learning: Learn a function using historical input/output data

$$y \approx \mathrm{NN}(x)$$



End goal is ***generalization***: Given "reasonable" unseen point (x^*, y^*), want $y^* \approx \mathrm{NN}(x^*)$

Optimize



$$\begin{aligned} \text{maximize} \quad & f(x) \\ \text{such that} \quad & x \in \Omega \end{aligned}$$

End goal is ***optimization***: Make the best possible decision $x^* \in \Omega$

Predict, then Optimize

$$\begin{aligned} \text{maximize} \quad & \text{NN}(x) \\ \text{such that} \quad & x \in \Omega \end{aligned}$$

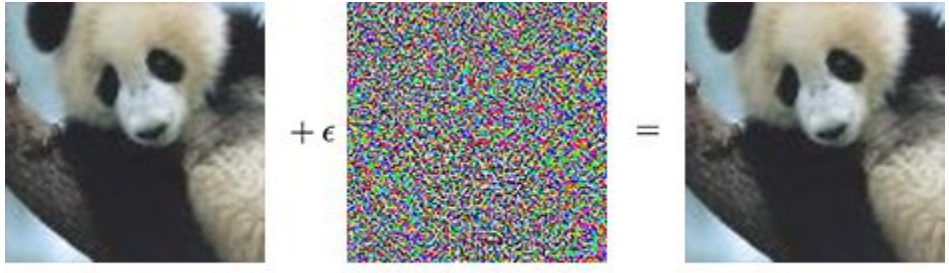
Nonconvex

Complex?
Combinatorial?

End goal is ***optimization***: Make the best possible decision $x^* \in \Omega$

Application: Verifying robustness

Adversarial Examples: find small change in the input to change the output



$$\begin{aligned} \max \quad & \mathbb{P}[x \text{ is a gibbon}] - \mathbb{P}[x \text{ is a panda}] \\ \text{s.t.} \quad & ||x - (\text{reference panda})||_\infty \leq \epsilon \end{aligned}$$

Predict: Classify input image

Optimize: Prove (or disprove) robustness of model to small perturbations

Very popular research topic in ML community right now

Note that “proofs” of robustness (not just feasible solutions) are very useful!

Application: Deep reinforcement learning

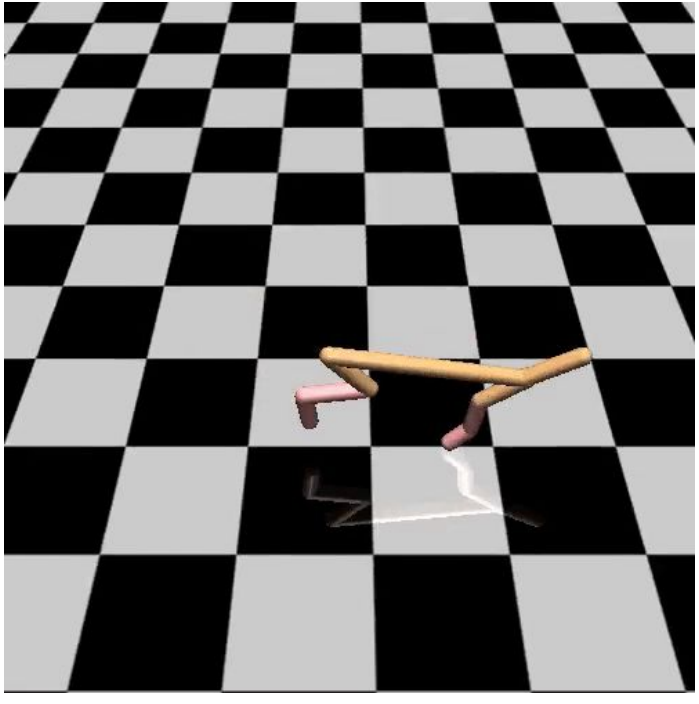
Predict: future costs for action *a* in state *x*:

$$Q(x, a) \approx \mathrm{NN}(x, a)$$

Optimize: to pick the lowest cost action:

$$\min_a \; c(x, a) + \alpha \cdot \mathrm{NN}(x, a)$$

In a combinatorial or continuous action space, optimization can be hard!

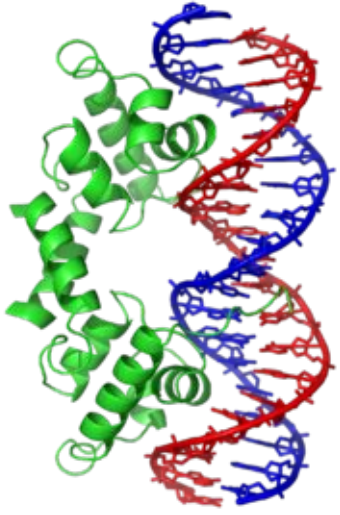


Teach a cheetah to run (without falling over)

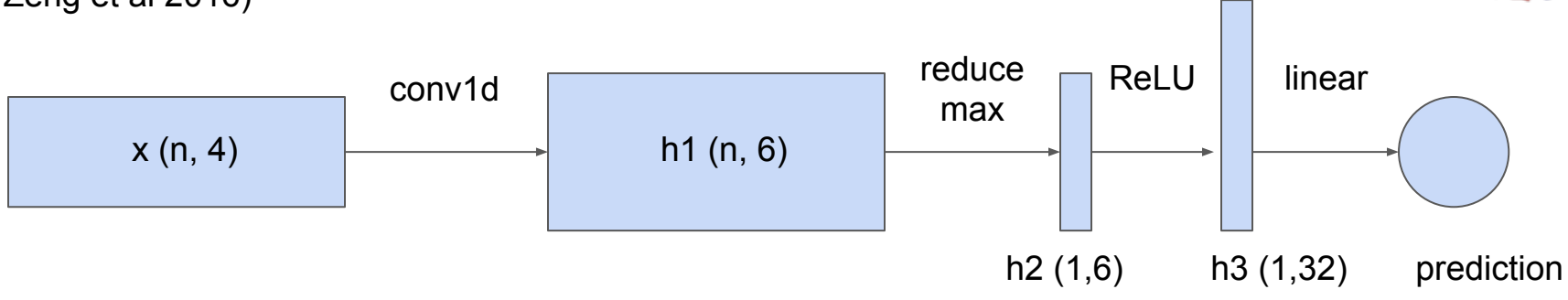
Application: Designing DNA for protein binding

Predict: probability/strength of DNA sequence binding to a given protein

Optimize: find the best/many binding sequences (potentially with side constraints)

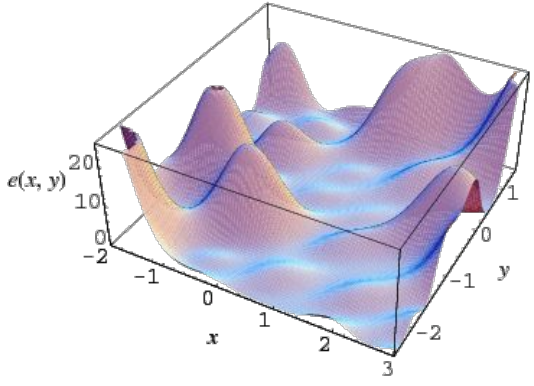


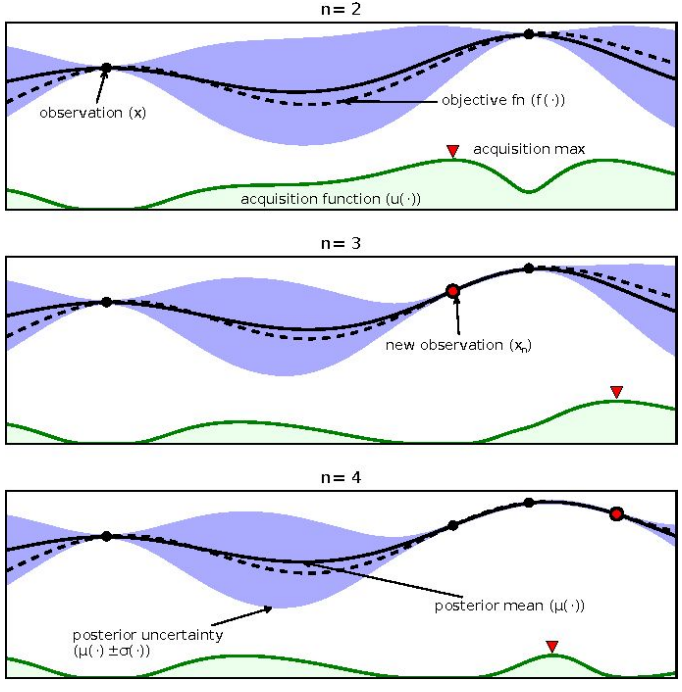
A typical architecture for predicting protein binding, e.g. (Alipanahi et al 2015, Zeng et al 2016)



Application: Bayesian optimization

- Goal: Minimize black-box function
  - “Neural architecture search”
  - Drug design
- Sequential experimentation, each sample is costly





Predict: Amount new point improves model

Optimize: Choose best point to sample

- Pros of Gaussian process:
  - Bayesian calculations are straightforward
- (Potential) pros of neural network:
  - Incremental training; handles discontinuities + high dims (“model-free”), input constraints
- Uncertainty models: dropout, ensembling, Thompson sampling, etc.

The Solution

How to “Optimize” in “Predict, then Optimize”

Heuristic/Primal

- **Gradient Descent (Projected/Conditional)**
- Local Search
  - Over input domain
  - **Over neuron activity**
- Cross Entropy Method
- Genetic Algorithms
- (Quasi-)Second Order Methods

Bounds/Dual

- Interval Arithmetic
- Abstract Domains
- Lagrangian Relaxation
- **Linear Programming**
- Semidefinite Programming

Exploit Problem-Specific Methods within Generic Solvers

Exact Methods

- **Mixed Integer Programming**
- SAT/SMT

A (not comprehensive) Literature Sampling

- Gradient descent on continuous problems (e.g. deep dream, style transfer, adversarial examples)
- Cross Entropy Method (importance sampling based guesses, brain robotics)
- Input Convex Neural Networks + LP (Amos 2016)
- The big-M MIP (too many authors to list all)
  - Tightening big-Ms: (Tjeng 2017, Fischetti 2018)
- CP/SAT/SMT (often from the “verifiability” community)
  - Reluplex (Katz 2017), Planet (Elhers 2017)
- Lagrangian Relaxation (Dvijotham 2018)
- Abstract Domains/Zonotopes (Singh 2019)
- Nice recent survey (Bunel 2017)

Contributions and Outline

1. MIP formulations for popular neural network building blocks

   Stronger than existing approaches (theoretically and empirically)

   - Most common case: Affine function on box domain → ReLU nonlinearity
   - Other nonlinearities: max-pooling, reduce max, clipped ReLU, …
   - Other domains: Product of simplices (*one-hot encodings*)
   - Structure: Tight big-M formulation + efficiently separable cuts
2. General machinery

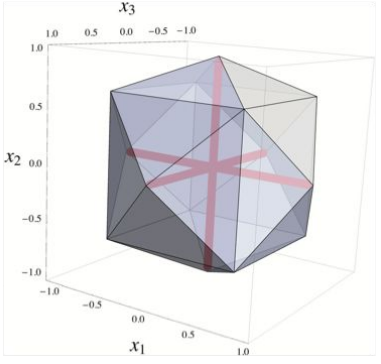
   Recipes for strong MIP formulations for the maximum of d affine functions

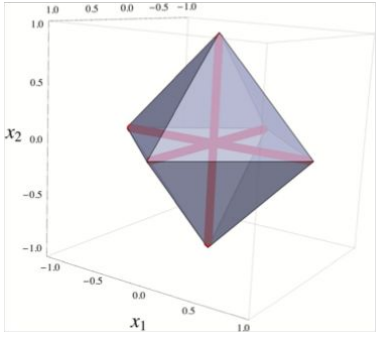
   - One for ideal formulations, another for hereditarily sharp ones
   - Efficiently separable via subgradient-method
   - A series of simplifications under common setting
3. `tf.opt`: Software with TensorFlow-like modeling, multiple backends
4. Computational results on verification and reinforcement learning problems

MIP Formulations for Neural Networks

MIP formulations in one slide

- A MIP formulation for some set $S \subseteq R^n$ is:
  - A polyhedra $Q \subseteq R^{n+r}$, where
  - $Proj_x(\{ (x,z) \in Q \mid z \in Z^r \}) = S$
- What makes a MIP formulation good?
  - *Size*: r is small, Q is "simple"
  - *Sharp*: $Proj_x(Q) = Conv(S)$
  - *Hereditarily sharp:* Sharp after any fixings of binary variables z
  - *Ideal (perfect)*: $ext(Q) \subseteq R^n \times Z^r$
- Ideal ⇒ sharp, so...





Ideal formulations = Best possible = Our goal

Neural networks = Piecewise linear functions

- Standard ReLU-based network: $x^r = \mathrm{NN}(x^0)$, where

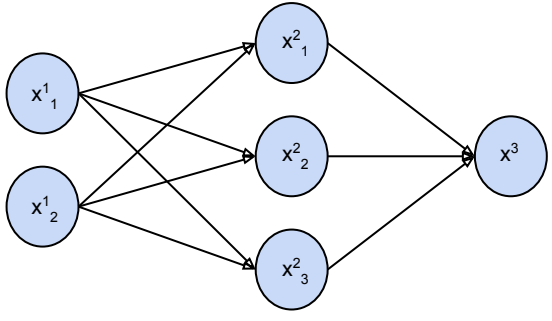
$$x_i^j = \max\left\{0, w^{j,i} \cdot x_i^{j-1} + b^{j,i}\right\} \qquad \forall j = 1, \ldots r, \;\; i = 1, \ldots, m_j$$

- Big-M formulation for single ReLU neuron

$$\left\{(x, y) : \begin{array}{c} L \le x \le U \\ y = \max\{0, w \cdot x + b\} \end{array}\right\}$$

is

$$\begin{aligned} y &\ge 0 \\ y &\ge w \cdot x + b \\ y &\le M^+ z \\ y &\le w \cdot x + b - M^-(1 - z) \\ (x, y, z) &\in [L, U] \times \mathbb{R} \times \{0, 1\} \end{aligned}$$

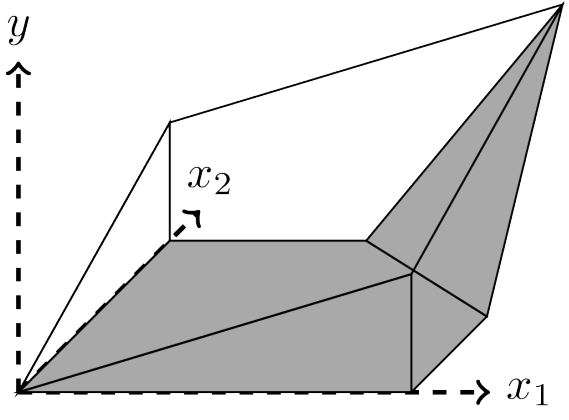


- How strong is it?

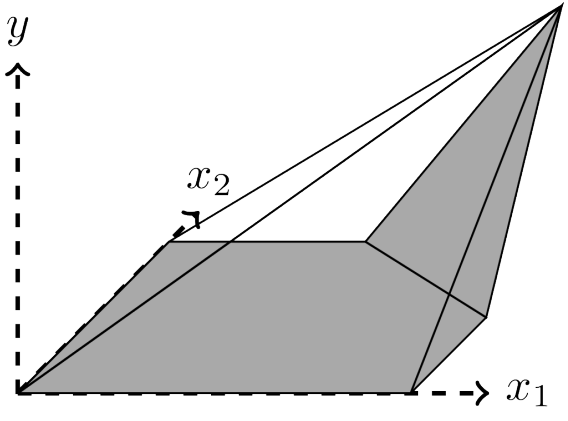
MIP formulation strength

- How strong is it? *Not very!*

  Can be arbitrarily bad, even in fixed input dimension



big-M formulation



tightest possible formulation

- How to close the gap?

An ideal formulation for ReLU neurons

Theorem (Anderson, H., Tjandraatmadja, Vielma 2018)

- An ideal formulation for $\{(x, y = \max\{0, w \cdot x + b\}) : L \leq x \leq U\}$ is

$$y \geq w \cdot x + b \tag{1a}$$

$$y \leq \sum_{i \in I} w_i \left(x_i - L_i(1 - z)\right) + \left(b + \sum_{i \notin I} w_i U_i\right) z \quad \forall I \subseteq [n] \tag{1b}$$

$$(x, y, z) \in [L, U] \times \mathbb{R}_{\geq 0} \times \{0, 1\}. \tag{1c}$$

- Each inequality in (1b) is facet-defining (under very mild conditions).
- Moreover, we can identify the most violated constraint in (1b) in O(n) time.

Big-M formulation = (1a), (1c), and two constraints from (1b)

Idea: Start with big-M formulation, use cut callbacks to separate (1b) as-needed

How do we get here?

Modeling the maximum of d affine functions over shared input domain D:

$$\left\{(x, y) \in D \times \mathbb{R} : y = \max_{k=1}^{d} w^k \cdot x + b^k\right\}$$

Step 1: Write down ideal “multiple choice” formulation (i.e. the “Balas” formulation):

$$(x, y) = \sum_{k=1}^{d} (\tilde{x}^k, w^k \cdot \tilde{x}^k + b^k z_k)$$

$$w^k \cdot \tilde{x}^k + b^k z_k \geq w^\ell \cdot \tilde{x}^k + b^\ell z_k \qquad \forall k, \ell \in [d] : \; k \neq \ell$$

$$\tilde{x}^k \in z_k \cdot D \qquad \forall k \in [d]$$

$$z \in \Delta^d.$$

How do we get here?

Modeling the maximum of d affine functions over shared input domain D:

$$\left\{(x, y) \in D \times \mathbb{R} : y = \max_{k=1}^{d} w^k \cdot x + b^k\right\}$$

Step 2: Re-write constraints in "set" form:

$$\begin{aligned}(x, y) &= \sum_{k=1}^{d} (\tilde{x}^k, w^k \cdot \tilde{x}^k + b^k z_k) \\ \tilde{x}^k &\in z_k \cdot D_{|k} && \forall k \in [d] \\ z &\in \Delta^d,\end{aligned}$$

where

$$\begin{aligned}D_{|k} &\equiv \left\{x \in D : k \in \arg\max_{\ell=1}^{d} w^\ell \cdot x + b^\ell\right\} \\ &= \left\{x \in D : w^k \cdot x + b^k \geq w^\ell \cdot x + b^\ell \quad \forall \ell \neq k\right\}\end{aligned}$$

How do we get here?

Modeling the maximum of d affine functions over shared input domain D:

$$\left\{(x, y) \in D \times \mathbb{R} : y = \max_{k=1}^{d} w^k \cdot x + b^k\right\}$$

Step 3: Rewrite all logic as bounds on output y (a *primal* characterization):

$$y \leq \overline{g}(x, z) \equiv \max_{\tilde{x}^1, \ldots, \tilde{x}^d} \left\{ \sum_{k=1}^{d} w^k \cdot \tilde{x}^k + b^k z_k : \begin{array}{ll} x = \sum_k \tilde{x}^k & \\ \tilde{x}^k \in z_k \cdot D_{|k} & \forall k \end{array} \right\}$$

$$y \geq \underline{g}(x, z) \equiv \min_{\tilde{x}^1, \ldots, \tilde{x}^d} \left\{ \sum_{k=1}^{d} w^k \cdot \tilde{x}^k + b^k z_k : \begin{array}{ll} x = \sum_k \tilde{x}^k & \\ \tilde{x}^k \in z_k \cdot D_{|k} & \forall k \end{array} \right\}$$

$$(x, z) \in D \times \Delta^d,$$

where

$$\begin{aligned} D_{|k} &\equiv \left\{x \in D : k \in \arg\max_{\ell=1}^{d} w^\ell \cdot x + b^\ell\right\} \\ &= \left\{x \in D : w^k \cdot x + b^k \geq w^\ell \cdot x + b^\ell \quad \forall \ell \neq k\right\} \end{aligned}$$

How do we get here?

Modeling the maximum of d affine functions over shared input domain D:

$$\left\{(x,y) \in D \times \mathbb{R} : y = \max_{k=1}^{d} w^k \cdot x + b^k \right\}$$

Step 4: Apply Lagrangian relaxation to aggregation constraints (a *dual* characterization)

$$y \leq \overline{\alpha} \cdot x + \sum_{k=1}^{d} \left(\max_{x^k \in D_{|k}} \{(w^k - \overline{\alpha}) \cdot x^k\} + b^k \right) z_k \quad \forall \overline{\alpha} \in \mathbb{R}^\eta$$

$$y \geq \underline{\alpha} \cdot x + \sum_{k=1}^{d} \left(\min_{x^k \in D_{|k}} \{(w^k - \underline{\alpha}) \cdot x^k\} + b^k \right) z_k \quad \forall \underline{\alpha} \in \mathbb{R}^\eta$$

$$(x,z) \in D \times \Delta^d.$$

End of analysis: in general, $D_{|k}$ is complicated.

Can separate over via subgradient method, or...

How do we get here?

Modeling the maximum of d affine functions over shared input domain D:

$$\left\{(x, y) \in D \times \mathbb{R} : y = \max_{k=1}^{d} w^k \cdot x + b^k\right\}$$

Step 1: Write down ideal “multiple choice” formulation (i.e. the “Balas” formulation):

$$(x, y) = \sum_{k=1}^{d} (\tilde{x}^k, w^k \cdot \tilde{x}^k + b^k z_k)$$

$$w^k \cdot \tilde{x}^k + b^k z_k \geq w^\ell \cdot \tilde{x}^k + b^\ell z_k \qquad \forall k, \ell \in [d] : \; k \neq \ell$$

$$\tilde{x}^k \in z_k \cdot D \qquad \forall k \in [d]$$

$$z \in \Delta^d.$$

How do we get here?

Modeling the maximum of d affine functions over shared input domain D:

$$\left\{(x, y) \in D \times \mathbb{R} : y = \max_{k=1}^{d} w^k \cdot x + b^k\right\}$$

Step 2: Re-write constraints in "set" form:

$$\begin{aligned}(x, y) &= \sum_{k=1}^{d} (\tilde{x}^k, w^k \cdot \tilde{x}^k + b^k z_k) \\ \tilde{x}^k &\in z_k \cdot D_{|k} && \forall k \in [d] \\ z &\in \Delta^d,\end{aligned}$$

where

$$\begin{aligned}D_{|k} &\equiv \left\{x \in D : k \in \arg\max_{\ell=1}^{d} w^\ell \cdot x + b^\ell\right\} \\ &= \left\{x \in D : w^k \cdot x + b^k \geq w^\ell \cdot x + b^\ell \quad \forall \ell \neq k\right\}\end{aligned}$$

How do we get here?

Modeling the maximum of d affine functions over shared input domain D:

$$\left\{(x, y) \in D \times \mathbb{R} : y = \max_{k=1}^{d} w^k \cdot x + b^k \right\}$$

Step 3: Relax domain constraint:

$$(x, y) = \sum_{k=1}^{d} (\tilde{x}^k, w^k \cdot \tilde{x}^k + b^k z_k)$$

$$\tilde{x}^k \in z_k \cdot D \qquad \forall k \in [d]$$

$$z \in \Delta^d.$$

Still a valid formulation.

How do we get here?

Modeling the maximum of d affine functions over shared input domain D:

$$\left\{(x, y) \in D \times \mathbb{R} : y = \max_{k=1}^{d} w^k \cdot x + b^k\right\}$$

Step 4: Rewrite all logic as bounds on output y (a *primal* characterization):

$$y \leq \overline{h}(x, z) \equiv \max_{\tilde{x}^1, \ldots, \tilde{x}^d} \left\{ \sum_{k=1}^{d} w^k \cdot \tilde{x}^k + b^k z_k : \begin{array}{ll} x = \sum_k \tilde{x}^k & \\ \tilde{x}^k \in z_k \cdot D & \forall k \end{array} \right\}$$

$$y \geq \underline{h}(x, z) \equiv \min_{\tilde{x}^1, \ldots, \tilde{x}^d} \left\{ \sum_{k=1}^{d} w^k \cdot \tilde{x}^k + b^k z_k : \begin{array}{ll} x = \sum_k \tilde{x}^k & \\ \tilde{x}^k \in z_k \cdot D & \forall k \end{array} \right\}$$

$$(x, z) \in D \times \Delta^d.$$

How do we get here?

Modeling the maximum of d affine functions over shared input domain D:

$$\left\{(x,y) \in D \times \mathbb{R} : y = \max_{k=1}^{d} w^k \cdot x + b^k \right\}$$

Step 5: Replace lower bounds for a *hereditarily sharp* formulation:

$$y \leq \overline{h}(x,z) \equiv \max_{\tilde{x}^1,\ldots,\tilde{x}^d} \left\{ \sum_{k=1}^{d} w^k \cdot \tilde{x}^k + b^k z_k : \begin{array}{l} x = \sum_k \tilde{x}^k \\ \tilde{x}^k \in z_k \cdot D \quad \forall k \end{array} \right\}$$

$$y \geq w^k \cdot x + b^k \quad \forall k \in [d]$$

$$(x,z) \in D \times \Delta^d.$$

How do we get here?

Modeling the maximum of d affine functions over shared input domain D:

$$\left\{(x, y) \in D \times \mathbb{R} : y = \max_{k=1}^{d} w^k \cdot x + b^k \right\}$$

Step 6: Apply Lagrangian relaxation to aggregation constraints:

$$y \leq \overline{\alpha} \cdot x + \sum_{k=1}^{d} \left(\max_{x^k \in D} \{(w^k - \overline{\alpha}) \cdot x^k\} + b^k \right) z_k \qquad \forall \overline{\alpha} \in \mathbb{R}^\eta$$

$$y \geq w^k \cdot x + b^k \quad \forall k \in [d]$$

$$(x, z) \in D \times \Delta^d.$$

How do we get here?

Modeling the maximum of d affine functions over shared input domain D:

$$\left\{(x, y) \in D \times \mathbb{R} : y = \max_{k=1}^{d} w^k \cdot x + b^k\right\}$$

Step 7: Analyze further:

- **Proposition** If d=2, then the hereditarily sharp formulation is ideal.
- **Proposition** If the domain is a product of simplices, separation over hereditarily sharp formulation reduces to a transportation problem.
- **Proposition** If d=2 and the domain is a product of simplices, the transportation problem has a closed form solution and efficient separation.

An ideal formulation for ReLU neurons

Theorem (Anderson, H., Tjandraatmadja, Vielma 2018)

- An ideal formulation for $\{(x, y = \max\{0, w \cdot x + b\}) : L \leq x \leq U\}$ is

$$y \geq w \cdot x + b \tag{1a}$$

$$y \leq \sum_{i \in I} w_i \left(x_i - L_i(1 - z)\right) + \left(b + \sum_{i \notin I} w_i U_i\right) z \quad \forall I \subseteq [n] \tag{1b}$$

$$(x, y, z) \in [L, U] \times \mathbb{R}_{\geq 0} \times \{0, 1\}. \tag{1c}$$

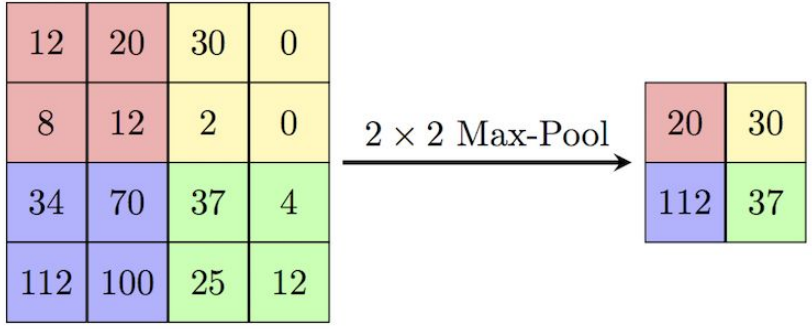
- Each inequality in (1b) is facet-defining (under very mild conditions).
- Moreover, we can identify the most violated constraint in (1b) in O(n) time.

Big-M formulation = (1a), (1c), and two constraints from (1b)

Idea: Start with big-M formulation, use cut callbacks to separate (1b) as-needed

Hereditarily sharp formulation for max-of-d affine functions

- Max-of-d affine functions ≡
  - Max pooling (small d)
  - Reduce max (large d)



Proposition (Anderson, H., Ma, Tjandraatmadja, Vielma 2019)

- A hereditarily sharp MIP formulation for $\left\{\left(x, y = \max\left\{w^k \cdot x + b^k\right\}_{k=1}^d\right) : L \le x \le U\right\}$ is:

$$y \le \sum_{i=1}^{\eta}\left(w_i^{I(i)} x_i + \sum_{k=1}^{d} \max\{(w_i^k - w_i^{I(i)})L_i, (w_i^k - w^{I(i)})U_i\} z_k\right) + \sum_{k=1}^{d} b^k z_k \quad \forall I : [\eta] \to [d]$$

$$y \ge w^k \cdot x + b^k \quad \forall k \in [d]$$

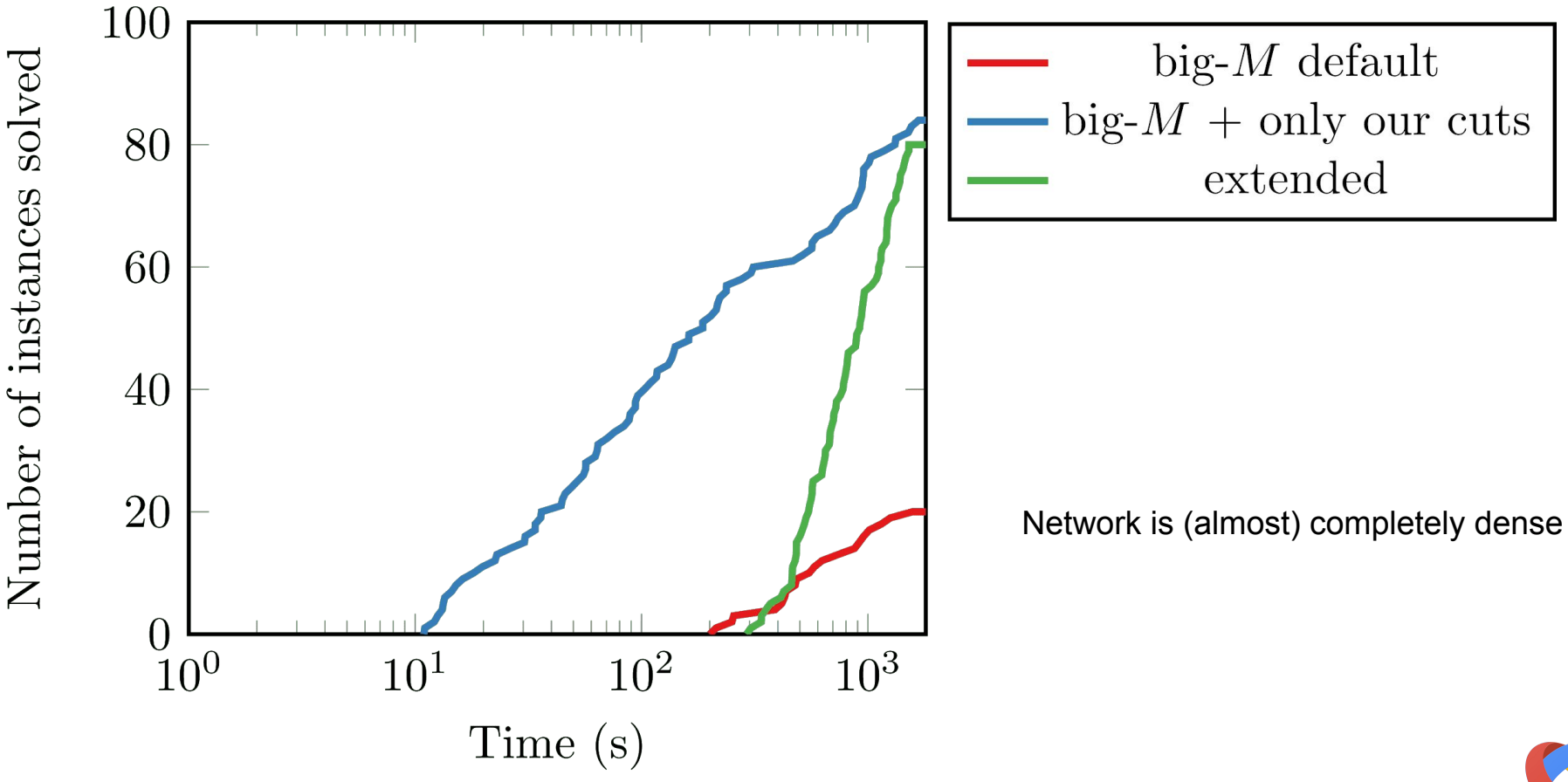
$$(x, z) \in D \times \Delta^d$$

$$z \in \{0,1\}^d.$$

- The most violated inequality can be identified in time linear in O(dn) time.

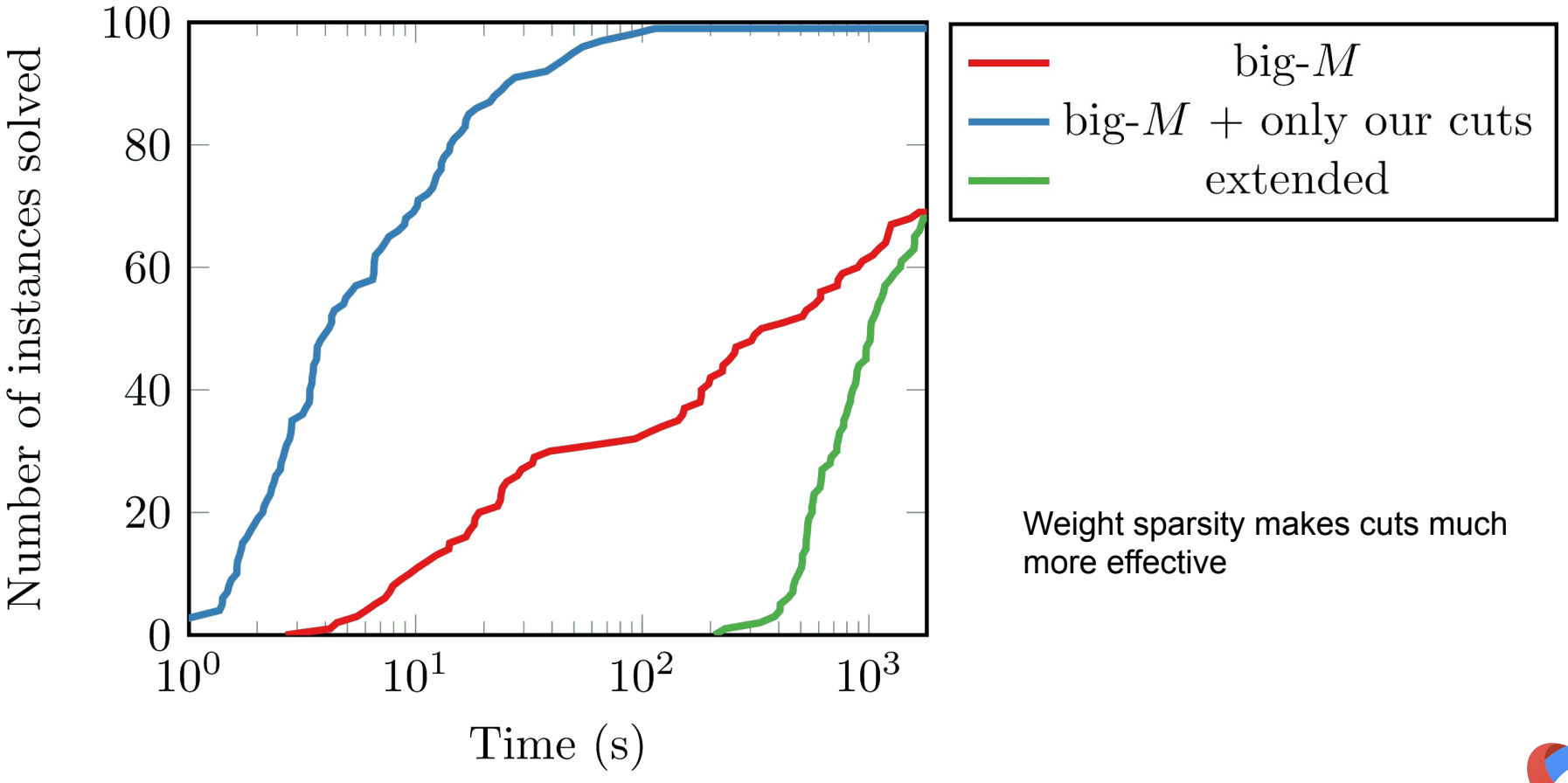
Computational Results

Network 1: Small network with standard training



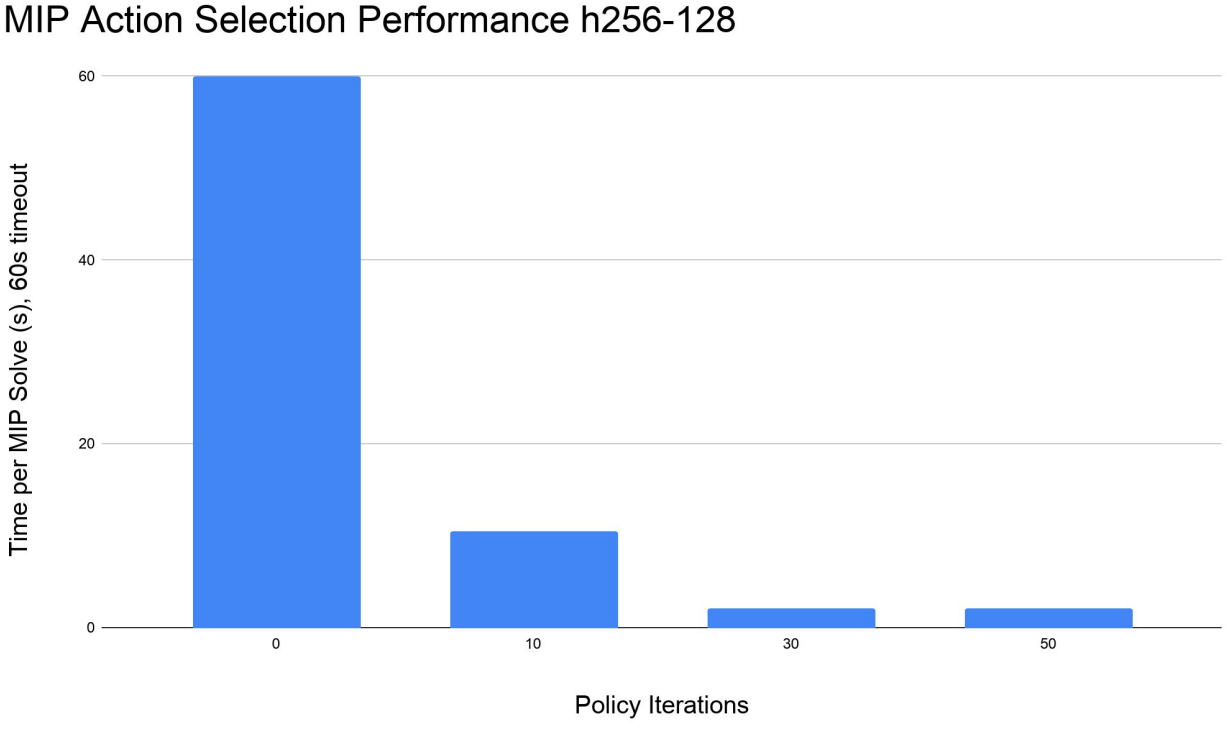
Network is (almost) completely dense

Network 2: Small network with L1 regularization (Xiao et al. 2019)



Weight sparsity makes cuts much more effective

Q-learning: Are optimal actions tractable?



MIP is very slow for early Policy Iterations

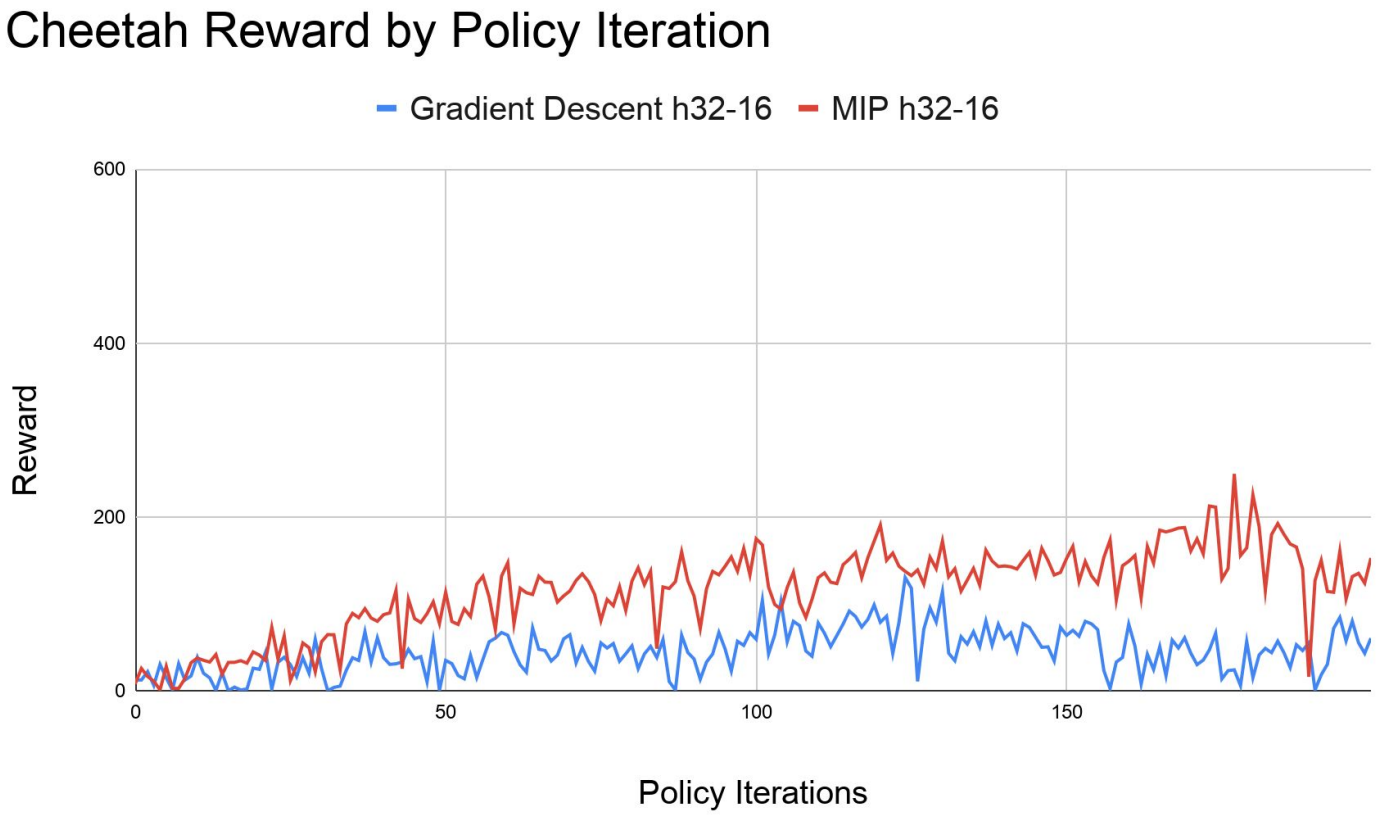
Active Index Local Search and Gradient Descent solves take ~100ms

Hybrid approach:

1. Run local search to produce decent solutions quickly
2. Feed into MIP with small time budget

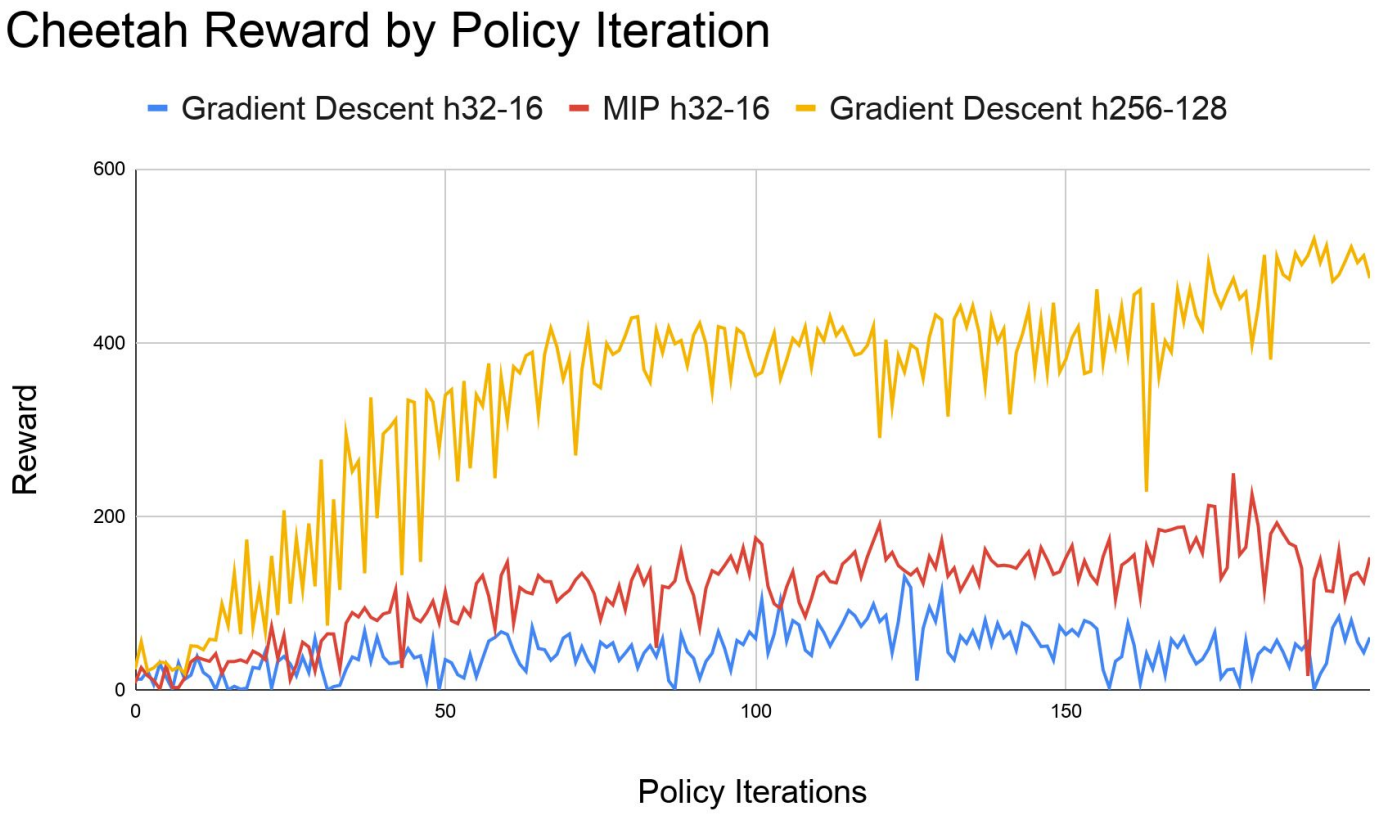
Ongoing joint work with Craig Boutilier, Martin Mladenov, and Moonkyung Ryu (Google Research)

Q-learning: Are optimal actions better?



Ongoing joint work with Craig Boutilier, Martin Mladenov, and Moonkyung Ryu (Google Research)

Q-learning: Are optimal actions better?



Ongoing joint work with Craig Boutilier, Martin Mladenov, and Moonkyung Ryu (Google Research)

Conclusion

- Strong MIP formulations for optimizing over trained neural networks
- Applications abound: verification, drug design, reinforcement learning, etc.
- Framework of independent interest:

  Recipes for strong formulations for the max of d affine functions

- Questions going forward:
  - Separation-based algorithm and implementation
  - How to train the network for optimization in a *principled* way
  - How best to formulate entire network (not just individual neurons)