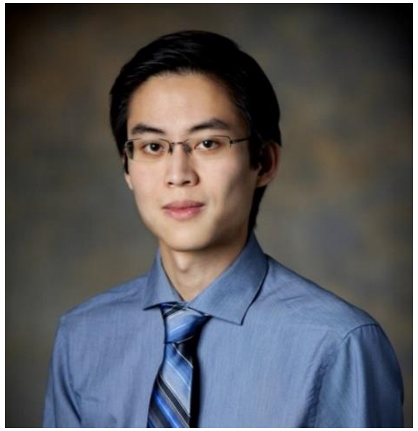
Bounding and Counting Linear Regions of Deep Neural Networks

Thiago Serra

Mitsubishi Electric Research Labs

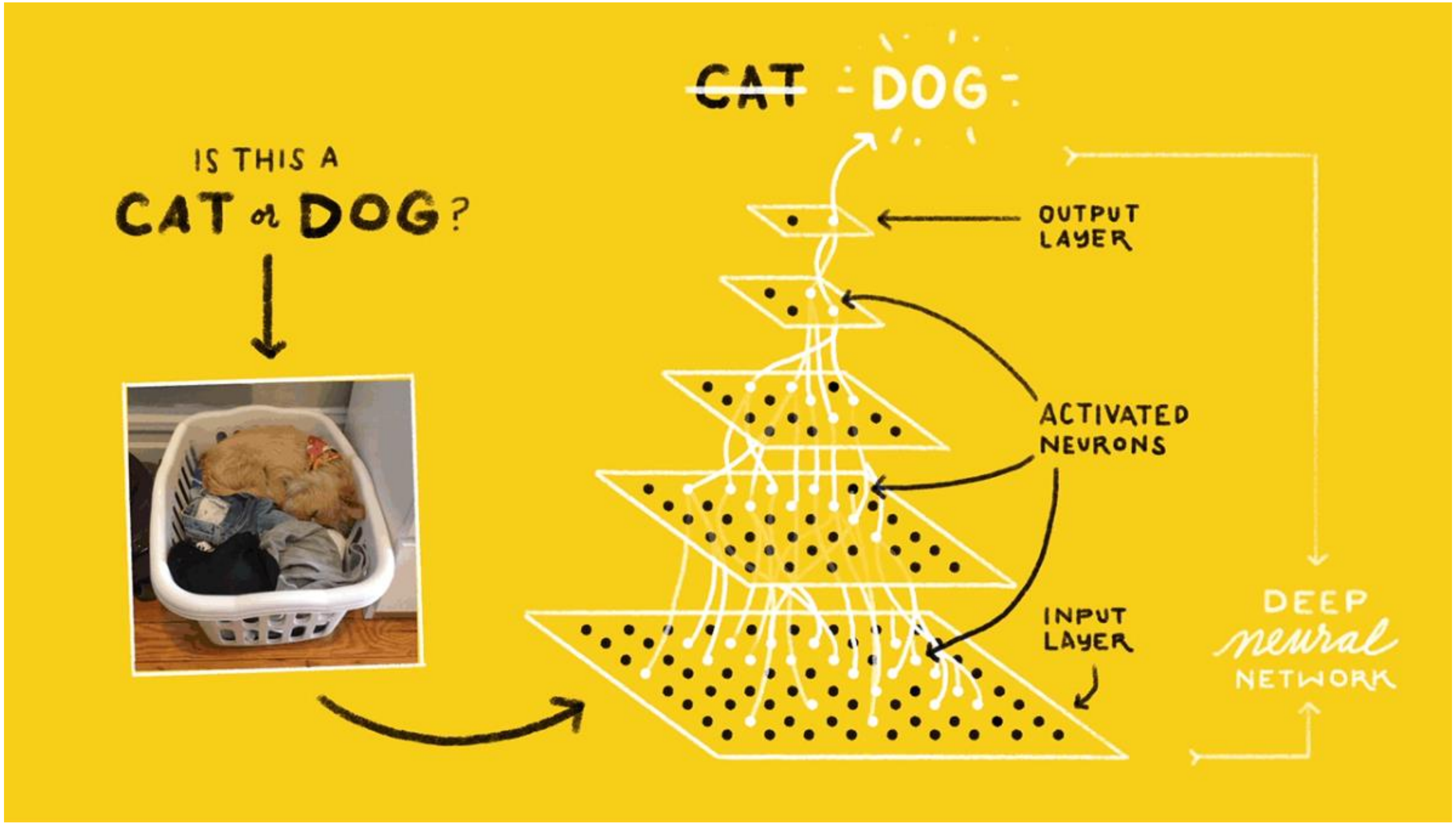
@thserra

Christian Tjandraatmadja
Google



Srikumar Ramalingam
The University of Utah

The Answer to Life, the Universe, and Everything



By Matteo Kofler
https://towardsdatascience.com/deep-learning-with-tensorflow-part-1-b19ce7803428

Or, Sometimes, Maybe Not…

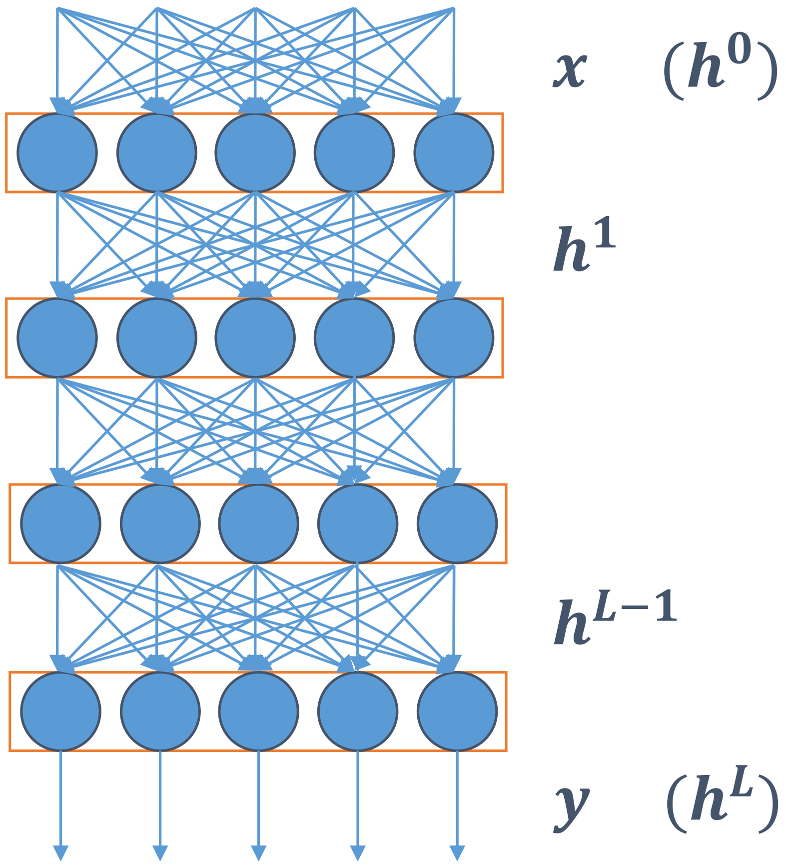
By Matt Hopkins
https://www.pedestrian.tv/tech/the-computer-from-hitchhikers-guide-to-the-galaxy-is-being-made-irl/

Notation

Notation:

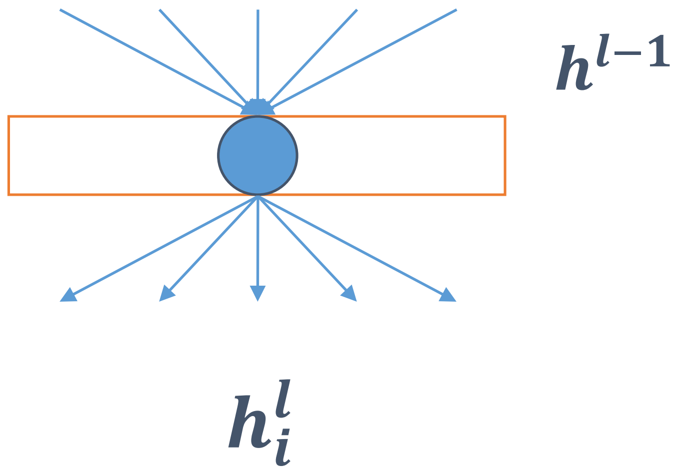
- Number of layers: L
- Width of layer l: n^l
- Output of layer l: $h^l \in \mathbb{R}^{n^l}$

- Input vector: $x\ (h^0)$
- Input dimension: n^0



The Scope of This Work

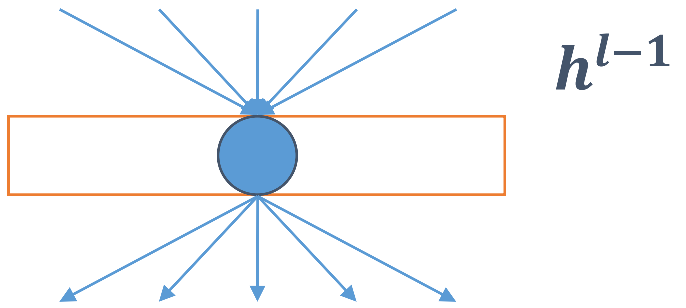
We study homogeneous DNNs with piecewise linear activations



The Scope of This Work

We study homogeneous DNNs with piecewise linear activations

- Rectifier Linear Unit (ReLU):

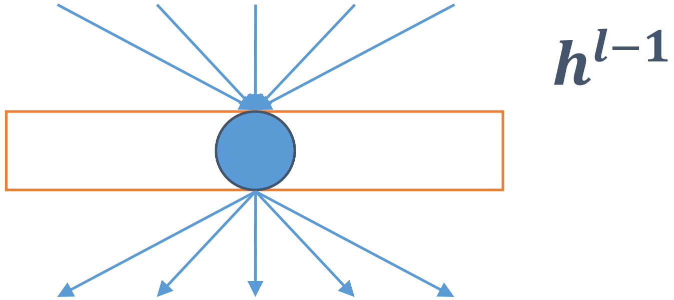


$$h_i^l = max\,\{\,0, W_i^l h^{l-1} + b_i^l\,\}$$

The Scope of This Work

We study homogeneous DNNs with piecewise linear activations

- Rectifier Linear Unit (ReLU):



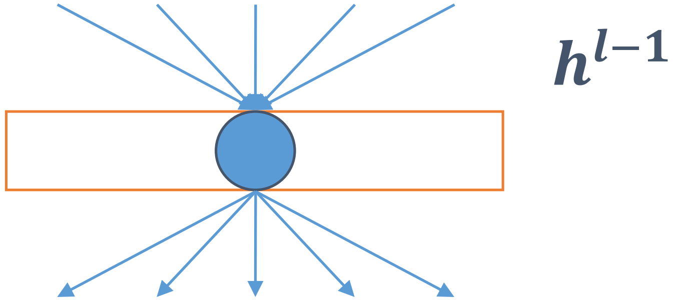
$$h_i^l = max\{\,0, W_i^l h^{l-1} + b_i^l\,\}$$

Inactive

The Scope of This Work

We study homogeneous DNNs with piecewise linear activations

- Rectifier Linear Unit (ReLU):

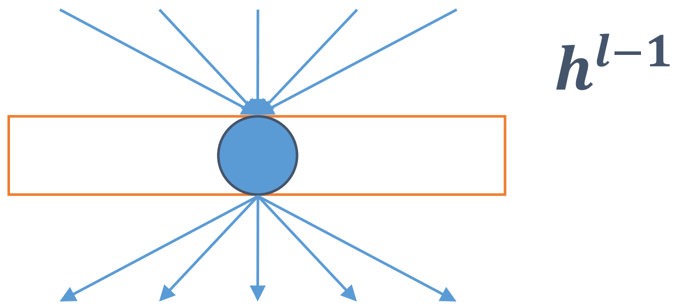


$$h_i^l = max\,\{\,\underbrace{0}_{\text{Inactive}},\, \underbrace{W_i^l h^{l-1} + b_i^l}_{\text{Active}}\,\}$$

The Scope of This Work

We study homogeneous DNNs with piecewise linear activations

- Rectifier Linear Unit (ReLU):

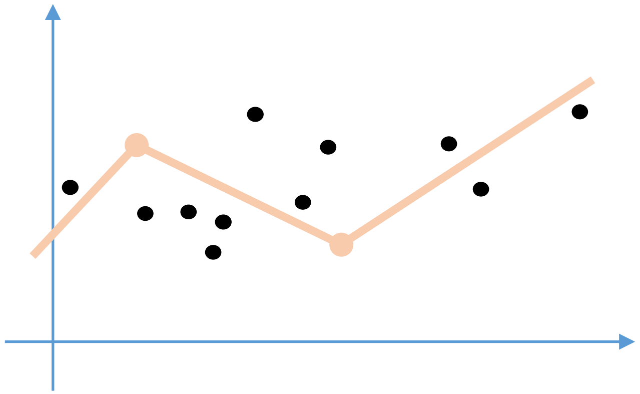


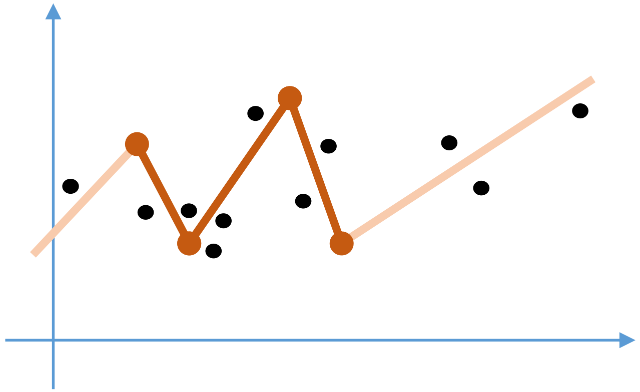
$$\boldsymbol{h_i^l = max\,\{\,\underbrace{0}_{\textbf{Inactive}},\ \underbrace{W_i^l h^{l-1} + b_i^l}_{\textbf{Active}}\,\}}$$

For piecewise linear activations, the DNN models a piecewise linear function

What Piecewise Linear Regression?

We study the number of "pieces", or linear regions, that can those DNNs can attain, both theoretically and empirically





What Piecewise Linear Regression?

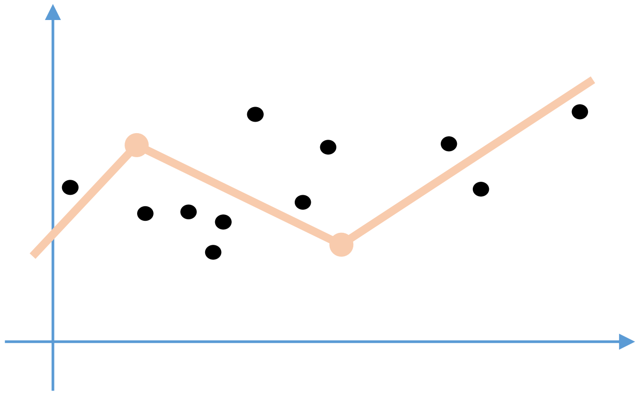
We study the number of "pieces", or linear regions, that can those DNNs can attain, both theoretically and empirically

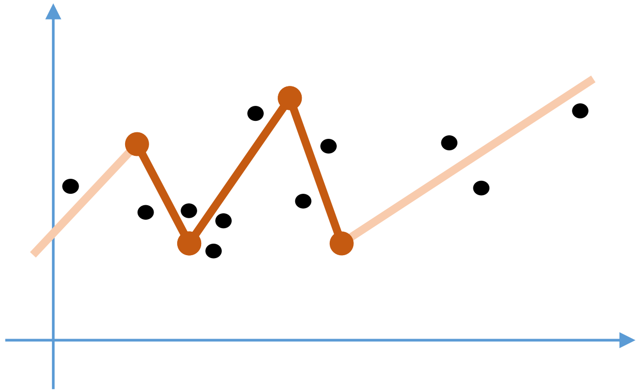
- Each linear region is mapped to the output by a single affine function

What Piecewise Linear Regression?

We study the number of "pieces", or <u>linear regions</u>, that can those DNNs can attain, both <u>theoretically</u> and <u>empirically</u>

- Each linear region is mapped to the output by a single affine function
- The configuration affects the <u>number</u> and <u>form</u> of the linear regions

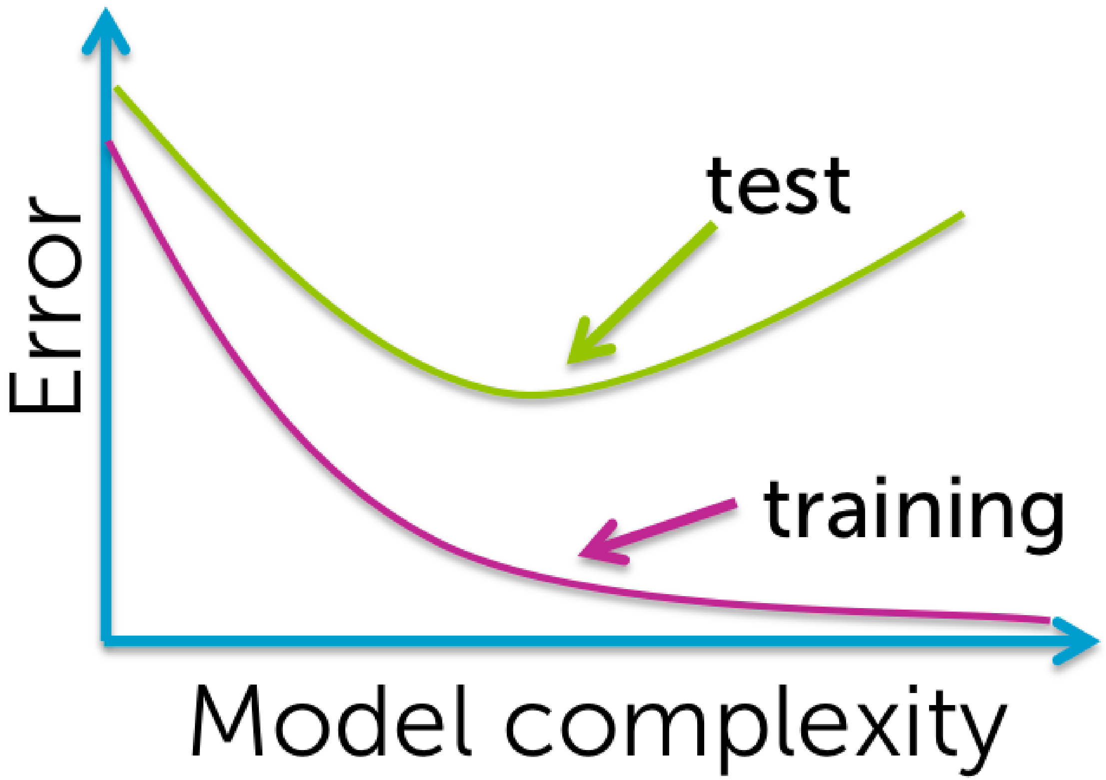




The Number of Regions Approach

Linear regions could be a proxy for model complexity

Pascanu et al. 2013, Montufar et al. 2014, Raghu et al. 2017, Montufar 2017, Arora et al. 2018



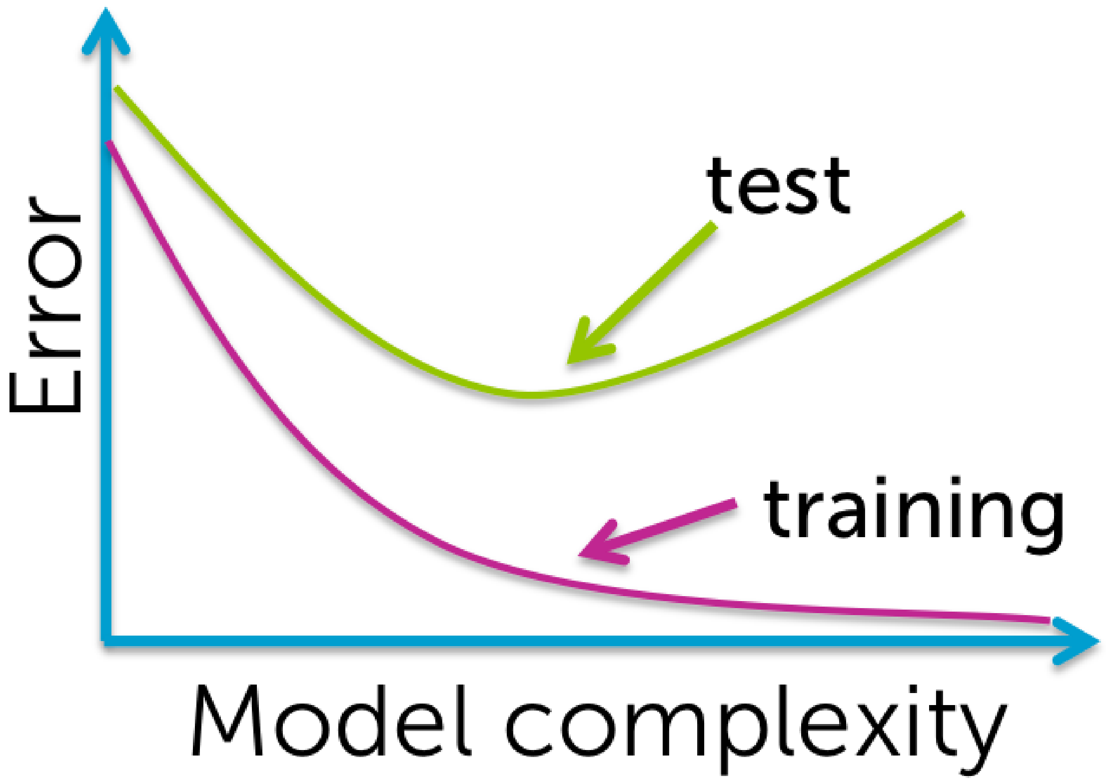
By Yurii
https://stats.stackexchange.com/questions/184103/why-the-error-on-a-training-set-is-decreasing-while-the-error-on-the-validation

The Number of Regions Approach

Linear regions could be a proxy for model complexity

Pascanu et al. 2013, Montufar et al. 2014, Raghu et al. 2017, Montufar 2017, Arora et al. 2018

- Enough capacity to fit well the training data well (**low training error**)



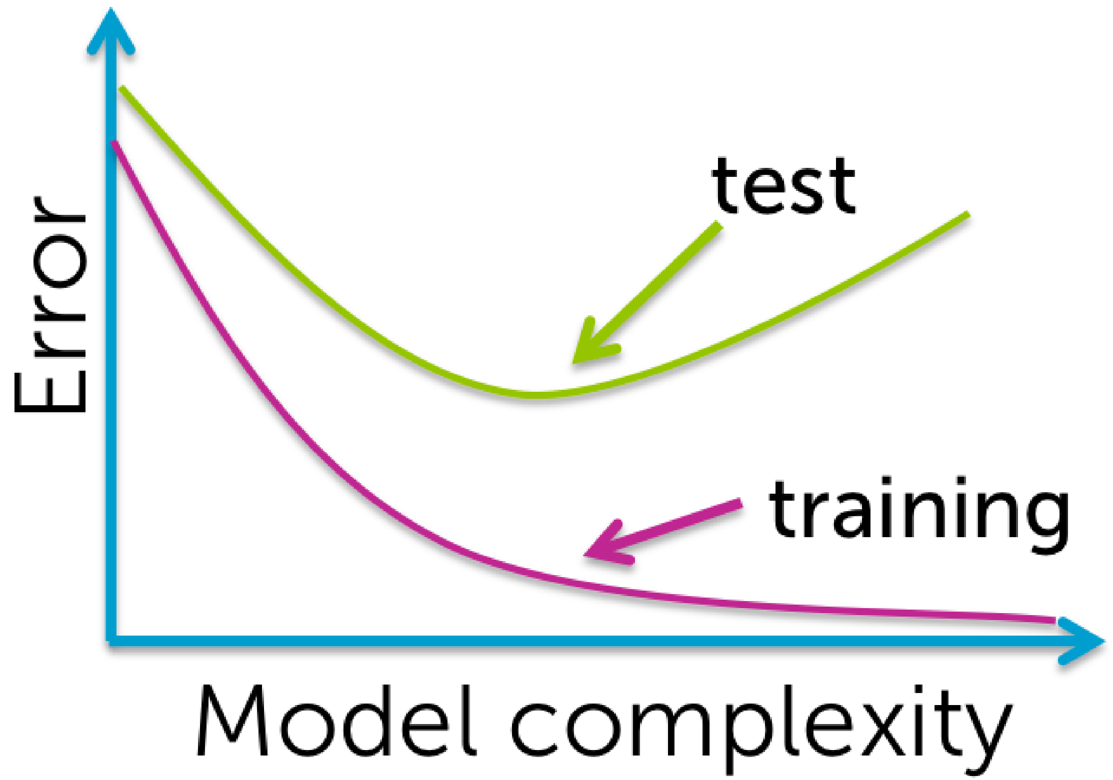
By Yurii
https://stats.stackexchange.com/questions/184103/why-the-error-on-a-training-set-is-decreasing-while-the-error-on-the-validation

The Number of Regions Approach

Linear regions could be a proxy for model complexity

Pascanu et al. 2013, Montufar et al. 2014, Raghu et al. 2017, Montufar 2017, Arora et al. 2018

- Enough capacity to fit well the training data well (**low training error**)
- Not so much that we single out the training points (**low test error**)



By Yurii
https://stats.stackexchange.com/questions/184103/why-the-error-on-a-training-set-is-decreasing-while-the-error-on-the-validation

Bounds on The Number of Linear Regions

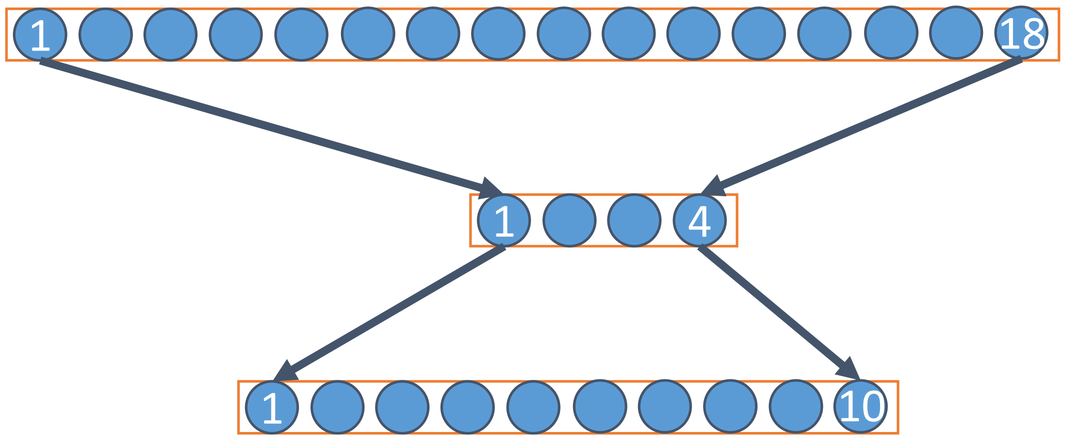
Negatives are important

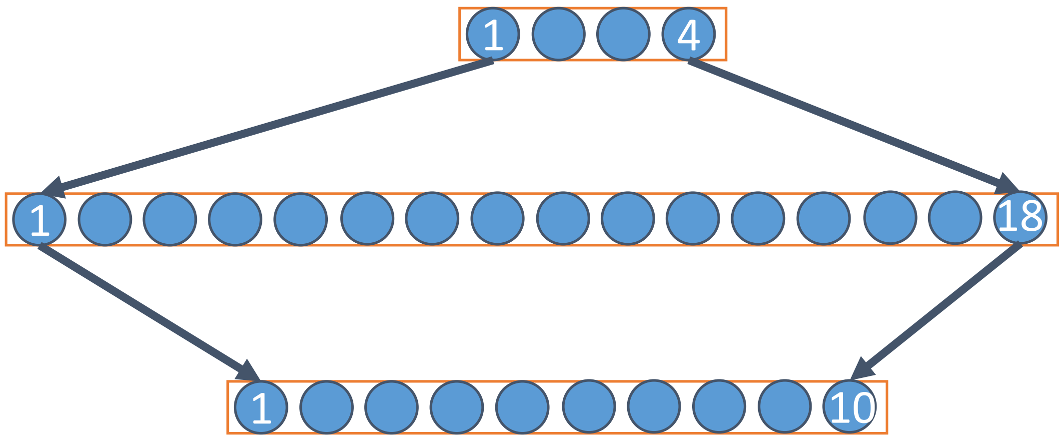
- Find limits to what functions can be approximated

Bounds on The Number of Linear Regions

Negatives are important

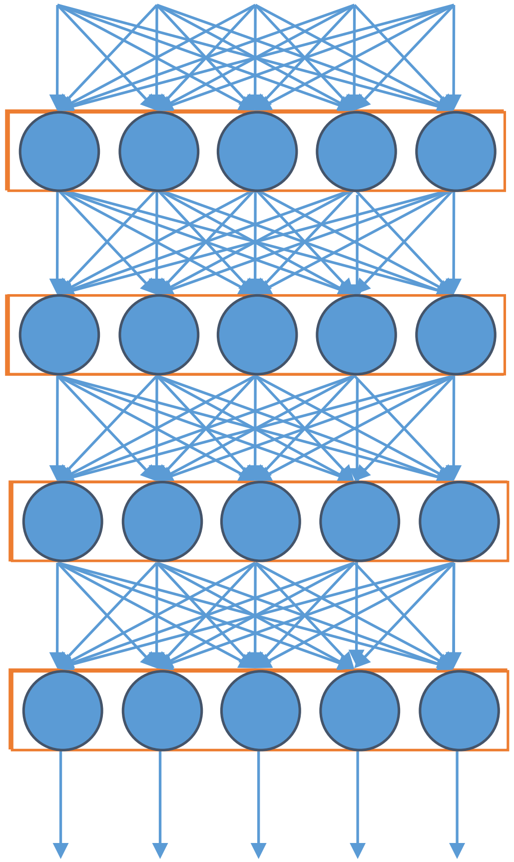
- Find limits to what functions can be approximated
- Comparison between different configurations





Activation Patterns and Linear Regions

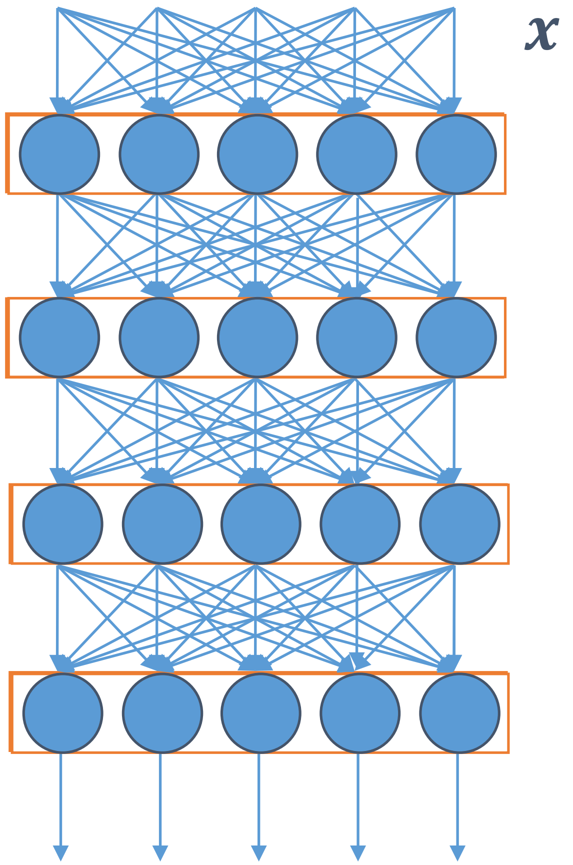
For ReLUs, we characterize these regions using the concept of activation patterns (Raghu et al., 2017; Montufar, 2017):



Activation Patterns and Linear Regions

For ReLUs, we characterize these regions using the concept of activation patterns (Raghu et al., 2017; Montufar, 2017):

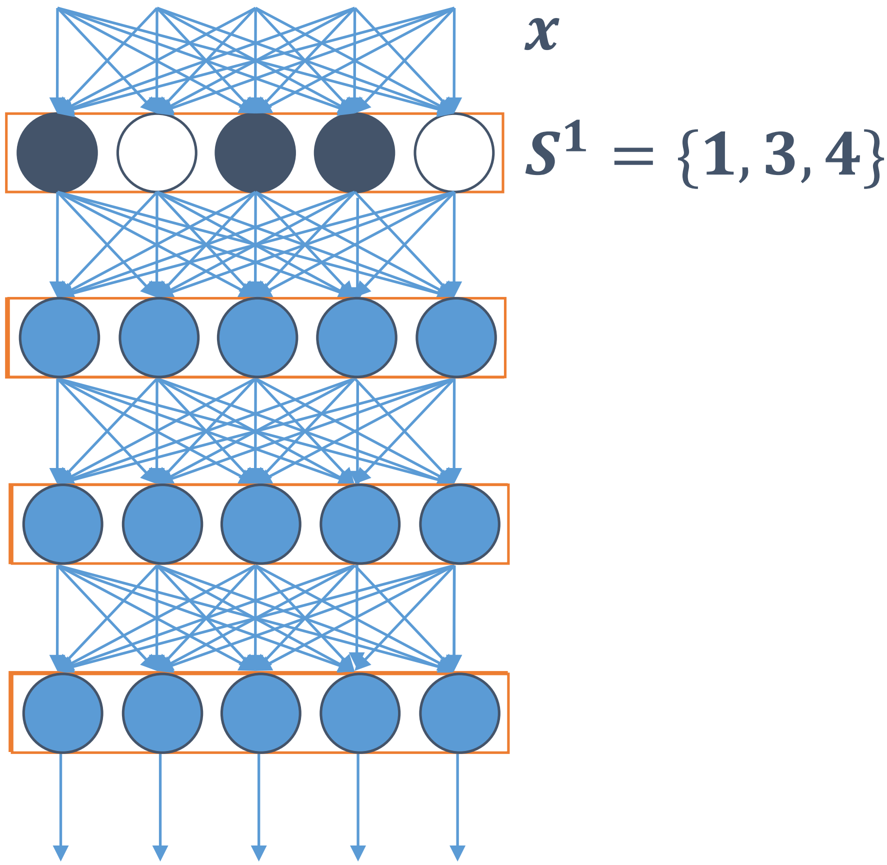
- For a given input $\boldsymbol{x}$



Activation Patterns and Linear Regions

For ReLUs, we characterize these regions using the concept of activation patterns (Raghu et al., 2017; Montufar, 2017):

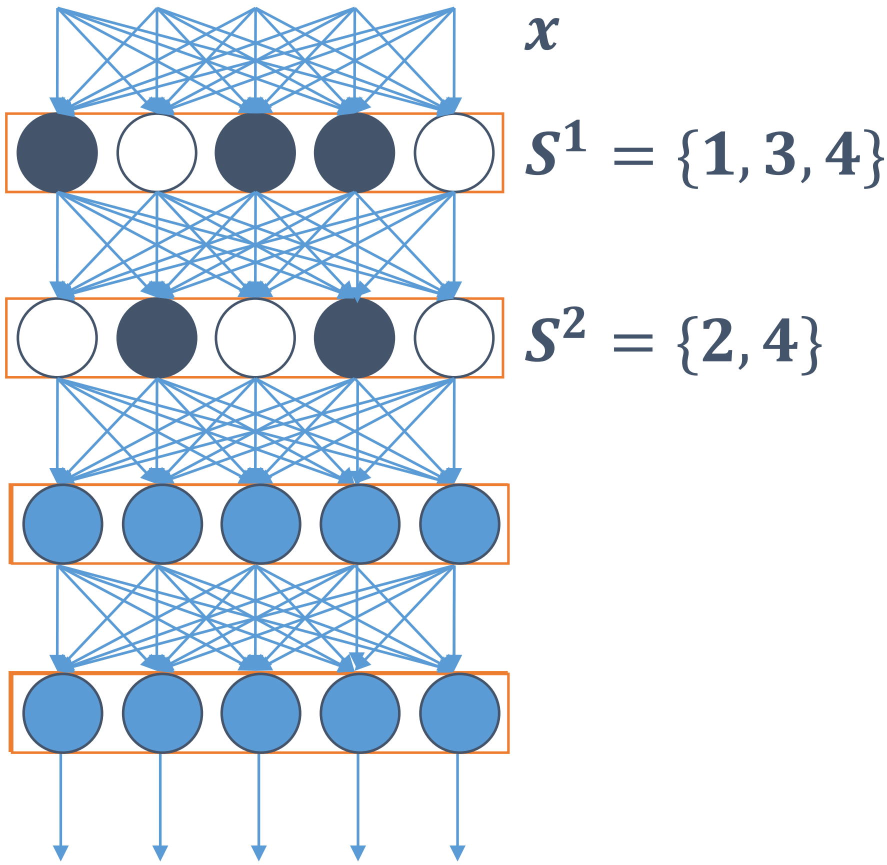
- For a given input $\boldsymbol{x}$
- There is an activation set $\boldsymbol{S^l} \subseteq \{\mathbf{1}, \mathbf{2}, \dots, \boldsymbol{n^l}\}$ for each layer l such that $\boldsymbol{i} \in \boldsymbol{S^l}$ iff $\mathbf{h_i^l} > \mathbf{0}$



Activation Patterns and Linear Regions

For ReLUs, we characterize these regions using the concept of activation patterns (Raghu et al., 2017; Montufar, 2017):

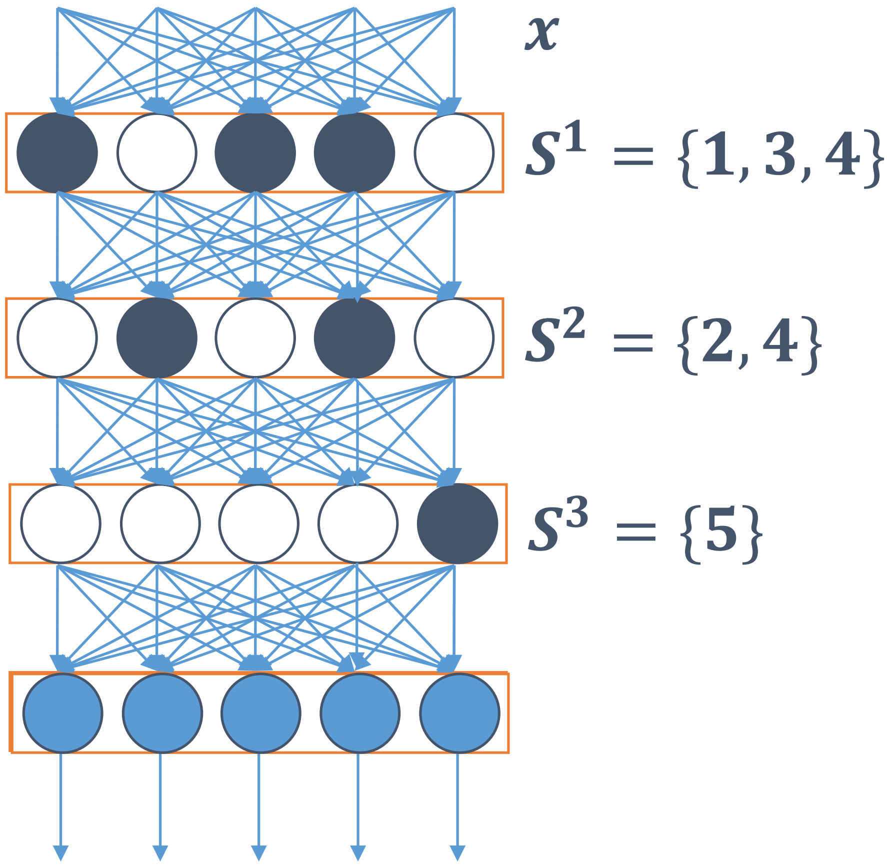
- For a given input $\boldsymbol{x}$
- There is an activation set $\boldsymbol{S}^{l} \subseteq \{1, 2, \dots, n^{l}\}$ for each layer l such that $\boldsymbol{i} \in \boldsymbol{S}^{l}$ iff $\mathbf{h}_{\mathbf{i}}^{\mathbf{l}} > \mathbf{0}$



Activation Patterns and Linear Regions

For ReLUs, we characterize these regions using the concept of activation patterns (Raghu et al., 2017; Montufar, 2017):

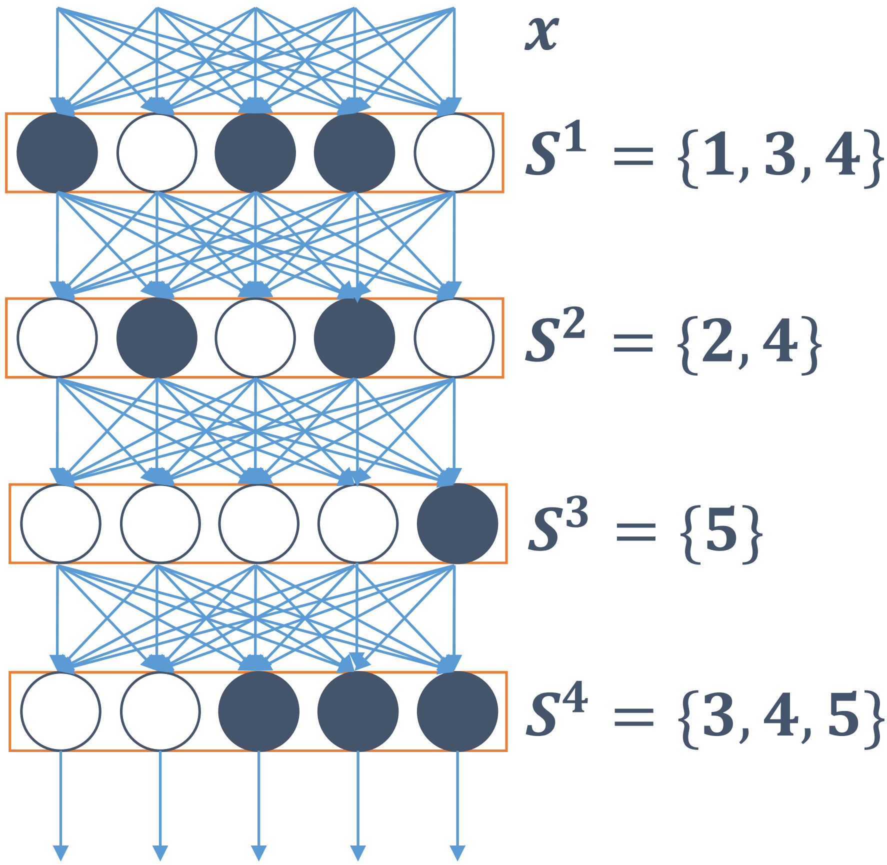
- For a given input $\boldsymbol{x}$
- There is an activation set $\boldsymbol{S^l} \subseteq \{\mathbf{1}, \mathbf{2}, \dots, \boldsymbol{n^l}\}$ for each layer l such that $\boldsymbol{i} \in \boldsymbol{S^l}$ iff $\mathbf{h_i^l} > \mathbf{0}$



Activation Patterns and Linear Regions

For ReLUs, we characterize these regions using the concept of activation patterns (Raghu et al., 2017; Montufar, 2017):

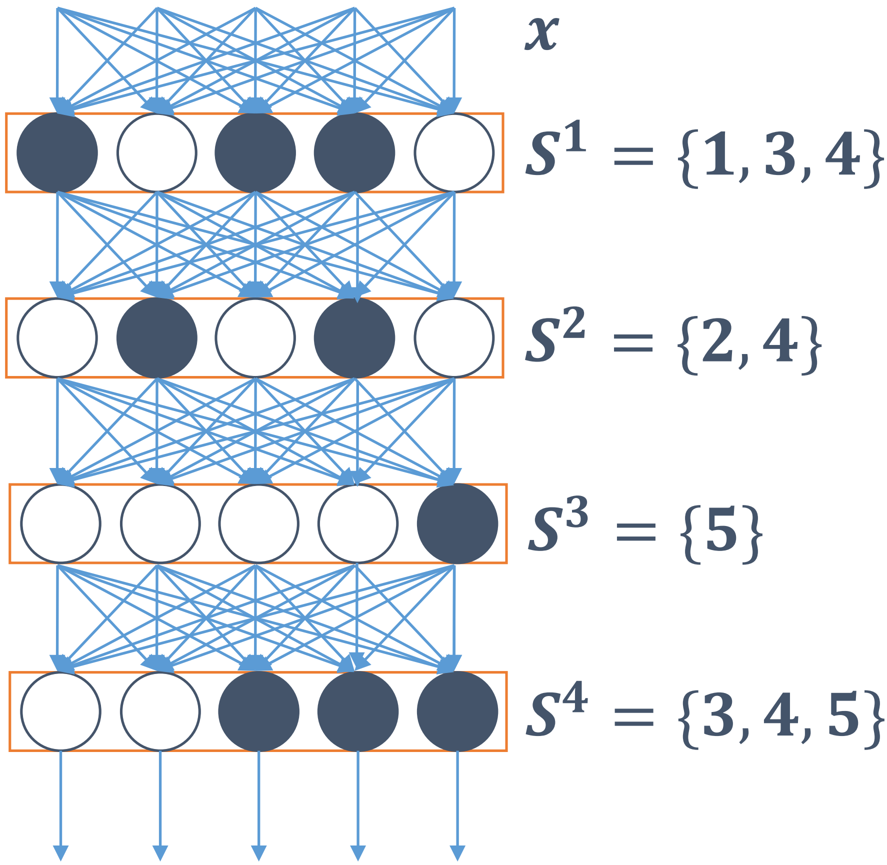
- For a given input $\boldsymbol{x}$
- There is an activation set $\boldsymbol{S^l} \subseteq \{\mathbf{1}, \mathbf{2}, \dots, \boldsymbol{n^l}\}$ for each layer l such that $\boldsymbol{i} \in \boldsymbol{S^l}$ iff $\mathbf{h_i^l} > \mathbf{0}$



Activation Patterns and Linear Regions

For ReLUs, we characterize these regions using the concept of activation patterns (Raghu et al., 2017; Montufar, 2017):

- For a given input $\boldsymbol{x}$
- There is an activation set $\boldsymbol{S^l} \subseteq \{\mathbf{1}, \mathbf{2}, \dots, \boldsymbol{n^l}\}$ for each layer l such that $\boldsymbol{i} \in \boldsymbol{S^l}$ iff $\mathbf{h_i^l} > \mathbf{0}$
- The activation pattern of $\boldsymbol{x}$ is $\boldsymbol{\mathcal{S}} = (\boldsymbol{S^1}, \dots, \boldsymbol{S^l})$

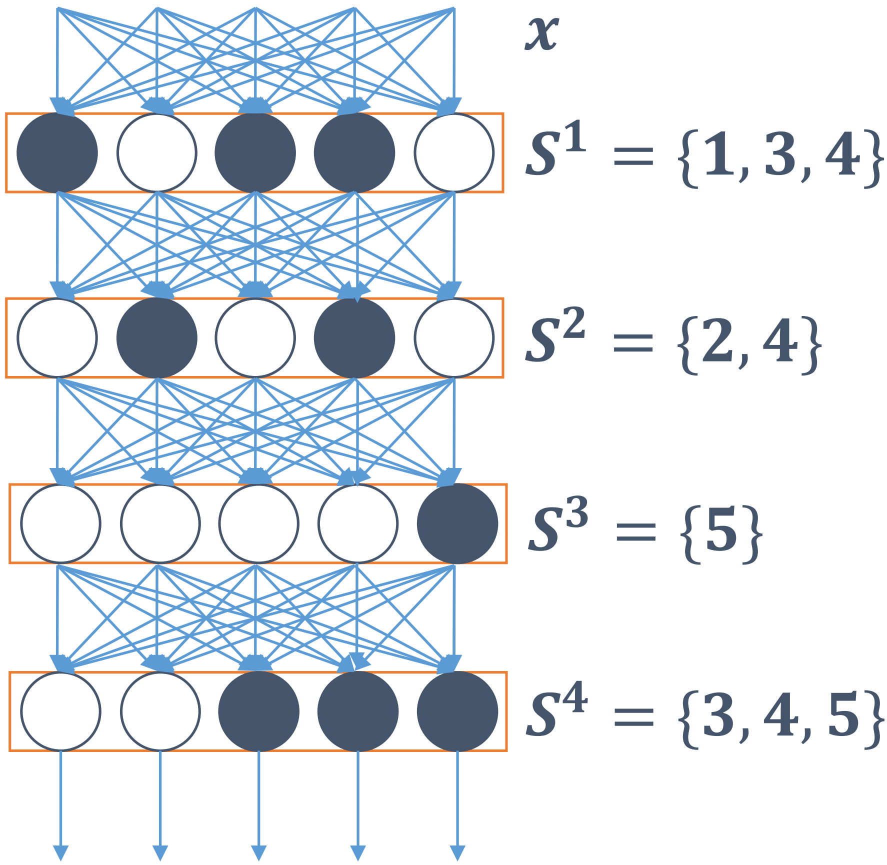


Activation Patterns and Linear Regions

For ReLUs, we characterize these regions using the concept of activation patterns (Raghu et al., 2017; Montufar, 2017):

- For a given input $\boldsymbol{x}$
- There is an activation set $\boldsymbol{S^l} \subseteq \{\mathbf{1}, \mathbf{2}, \dots, \boldsymbol{n^l}\}$ for each layer l such that $\boldsymbol{i} \in \boldsymbol{S^l}$ iff $\mathbf{h_i^l} > \mathbf{0}$
- The activation pattern of $\boldsymbol{x}$ is $\boldsymbol{\mathcal{S}} = (\boldsymbol{S^1}, \dots, \boldsymbol{S^l})$

A linear region is the set of all points with a same activation pattern



Bounds on Rectifier Networks

- Better theoretical limits to the number of regions

Bounding Deep Networks, Act 0

The number of activation patterns is a first upper bound (Montufar et al., 2014):

$$2^{n_1+\ \ldots + n_L}$$

Bounding Deep Networks, Act 0

The number of activation patterns is a first upper bound (Montufar et al., 2014):

$$\mathbf{2^{n_1 + \ldots + n_L}}$$

However, we cannot differentiate configurations with same number of units!

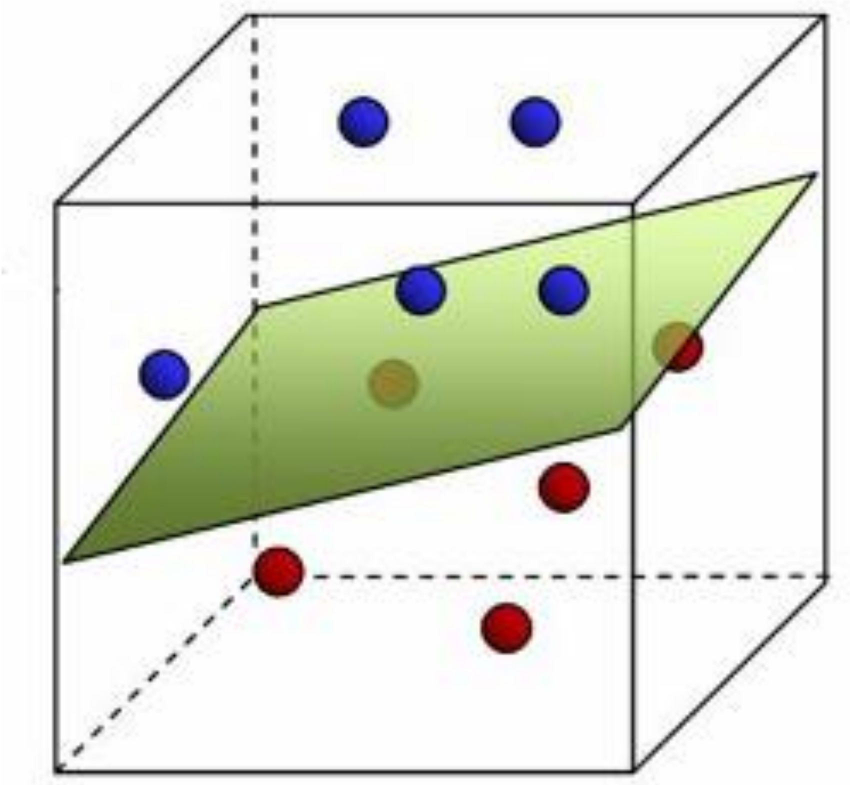
Building Blocks to Bound Linear Regions

For each unit *i* in layer *l*, $\boldsymbol{W}_{\boldsymbol{i}}^{\boldsymbol{l}}$ and $\boldsymbol{b}_{\boldsymbol{i}}^{\boldsymbol{l}}$ define an activation hyperplane on $\boldsymbol{h}^{\boldsymbol{l-1}}$:

Building Blocks to Bound Linear Regions

For each unit *i* in layer *l*, $\boldsymbol{W}_i^l$ and $\boldsymbol{b}_i^l$ define an activation hyperplane on $\boldsymbol{h}^{l-1}$:

$$\boldsymbol{W}_i^l \boldsymbol{h}^{l-1} + \boldsymbol{b}_i^l = \mathbf{0}$$



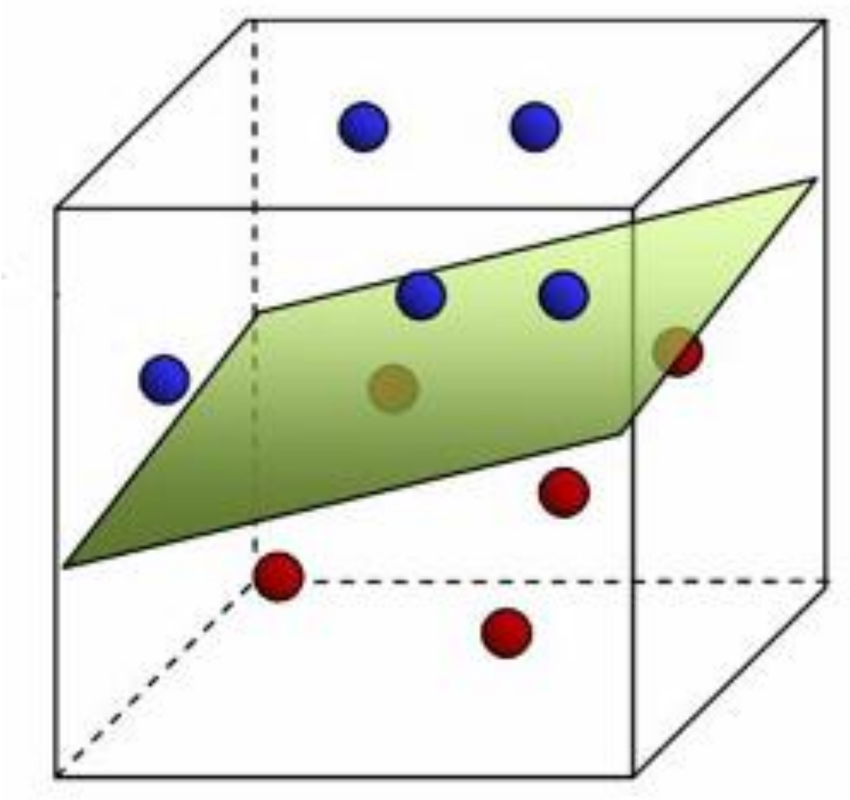
By Arvind Narayanan
https://33bits.wordpress.com/2010/12/20/the-unsung-success-of-can-spam/

Building Blocks to Bound Linear Regions

For each unit *i* in layer *l*, $\boldsymbol{W_i^l}$ and $\boldsymbol{b_i^l}$ define an <u>activation hyperplane</u> on $\boldsymbol{h^{l-1}}$:

$$\boldsymbol{W_i^l h^{l-1} + b_i^l = 0}$$

Active points:
$$\boldsymbol{W_i^l h^{l-1} + b_i^l > 0}$$



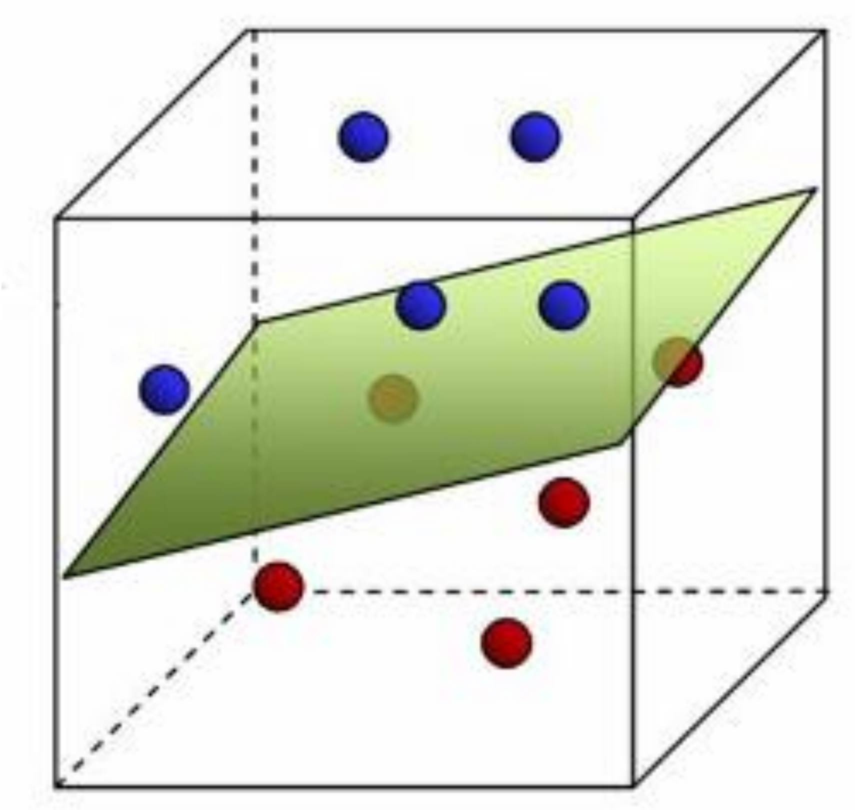
By Arvind Narayanan
https://33bits.wordpress.com/2010/12/20/the-unsung-success-of-can-spam/

Building Blocks to Bound Linear Regions

For each unit *i* in layer *l*, $\boldsymbol{W}_i^l$ and $\boldsymbol{b}_i^l$ define an <u>activation hyperplane</u> on $\boldsymbol{h}^{l-1}$:

$$\boldsymbol{W}_i^l \boldsymbol{h}^{l-1} + \boldsymbol{b}_i^l = 0$$

Active points:
$$\boldsymbol{W}_i^l \boldsymbol{h}^{l-1} + \boldsymbol{b}_i^l > 0$$



Inactive points:
$$\boldsymbol{W}_i^l \boldsymbol{h}^{l-1} + \boldsymbol{b}_i^l \leq 0$$

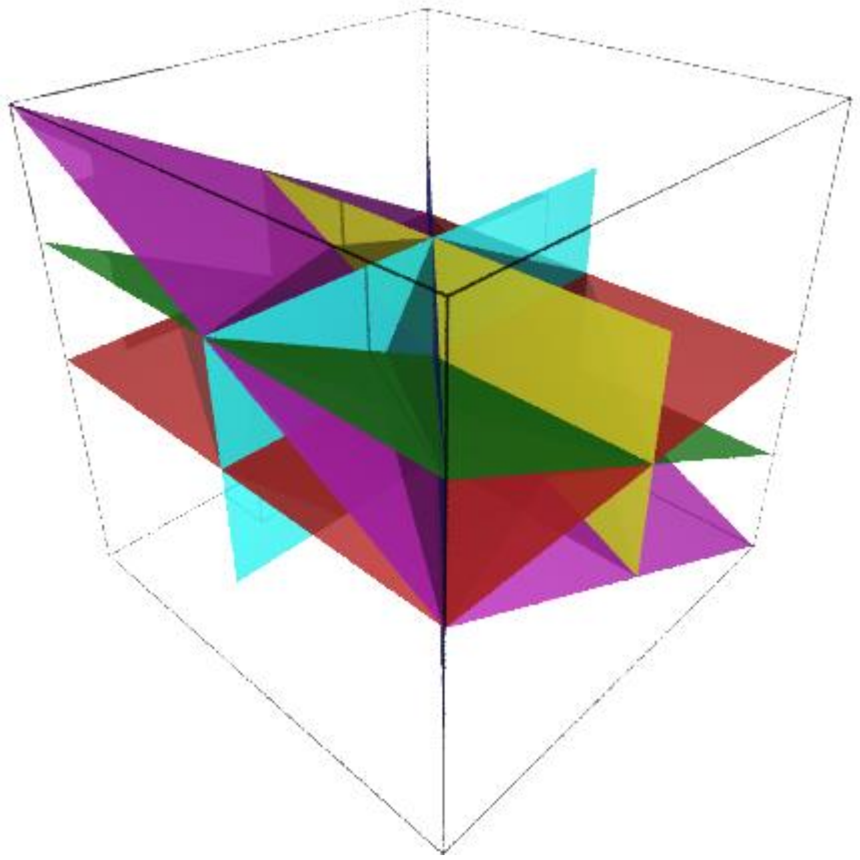
By Arvind Narayanan
https://33bits.wordpress.com/2010/12/20/the-unsung-success-of-can-spam/

Building Blocks to Bound Linear Regions

We can use the theory of hyperplane arrangements on the layers (Zaslavsky, 1975):

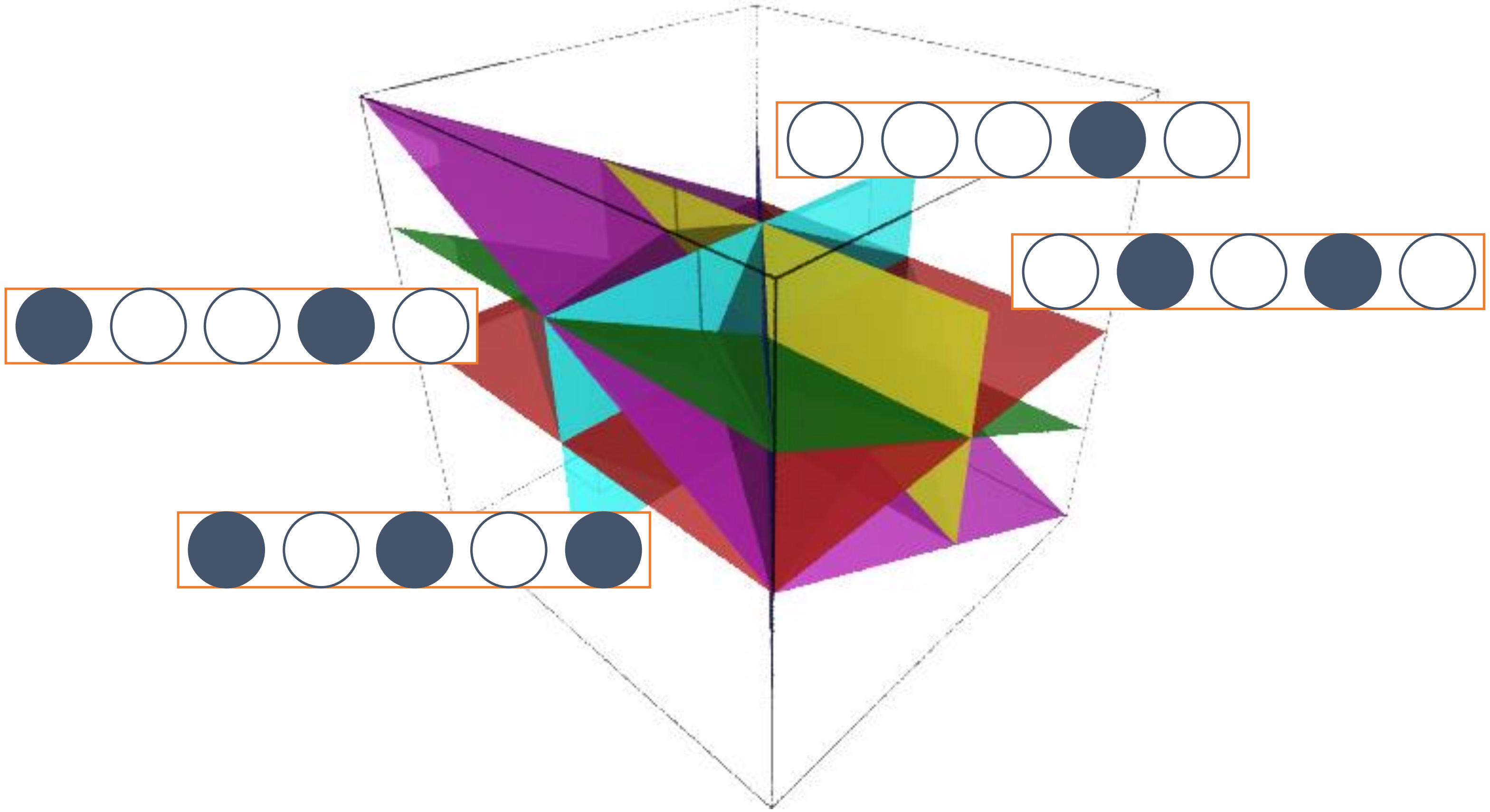
- The number of full-dimensional regions defined by $\boldsymbol{n}$ hyperplanes in $\mathbb{R}^d$ is

$$\sum_{i=0}^{d} \binom{n}{i}$$



The Effect of a Single Layer

Each full-dimensional polyhedron defined by the arrangement of activation hyperplanes of a given layer corresponds to a distinct activation set



Bounding Shallow Networks

The number of regions of a shallow network is at most

$$\sum_{i=0}^{n_0} \binom{n_1}{i}$$

Bounding Shallow Networks

The number of regions of a shallow network is at most

$$\sum_{i=0}^{n_0} \binom{n_1}{i}$$

With 4 hyperplanes in 2 dimensions, we have:

$$\binom{4}{0} + \binom{4}{1} + \binom{4}{2} = 1 + 4 + 6 = 11$$

Bounding Shallow Networks

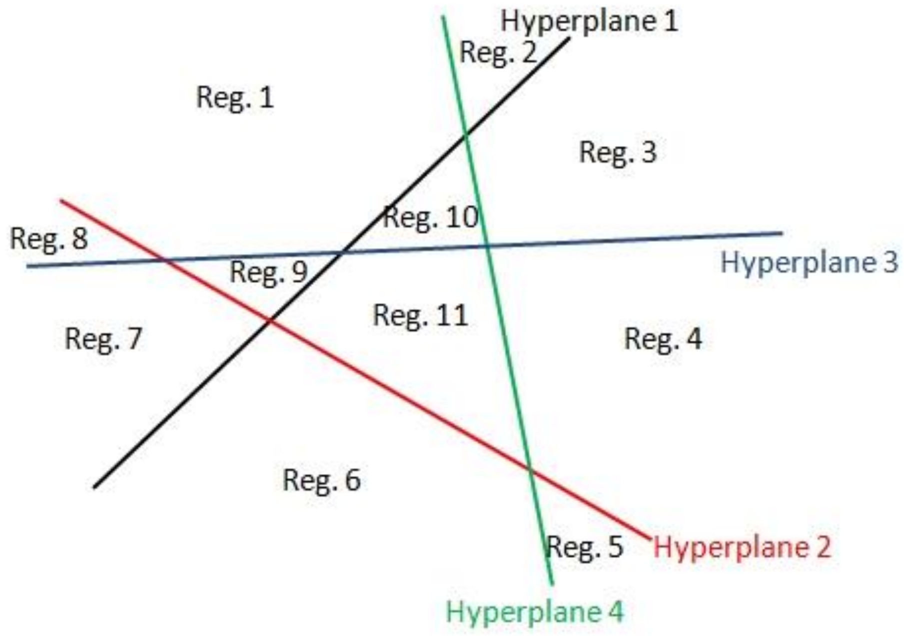
The number of regions of a shallow network is at most

$$\sum_{i=0}^{n_0} \binom{n_1}{i}$$

With 4 hyperplanes in 2 dimensions, we have:

$$\binom{4}{0} + \binom{4}{1} + \binom{4}{2} = 1 + 4 + 6 = 11$$

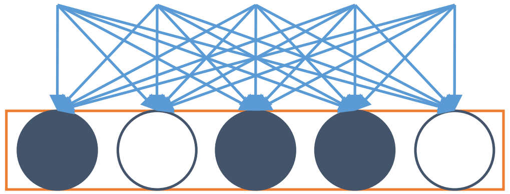
<ins>We can always reach that bound</ins>



From https://onionesquereality.wordpress.com/2012/11/23/maximum-number-of-regions-in-arrangement-of-hyperplanes/

Bounding Deep Networks, Act 1

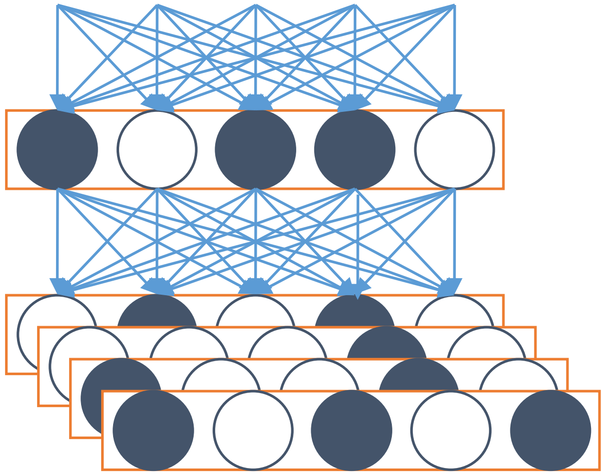
We can generalize the previous idea to multiple layers:



Bounding Deep Networks, Act 1

We can generalize the previous idea to multiple layers:

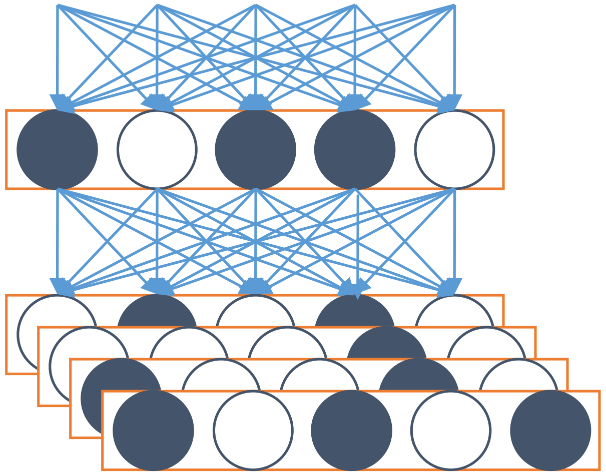
- Each LR in layer l can be potentially combined with all LRs in the subsequent layers



Bounding Deep Networks, Act 1

We can generalize the previous idea to multiple layers:

- Each LR in layer $\boldsymbol{l}$ can be potentially combined with all LRs in the subsequent layers



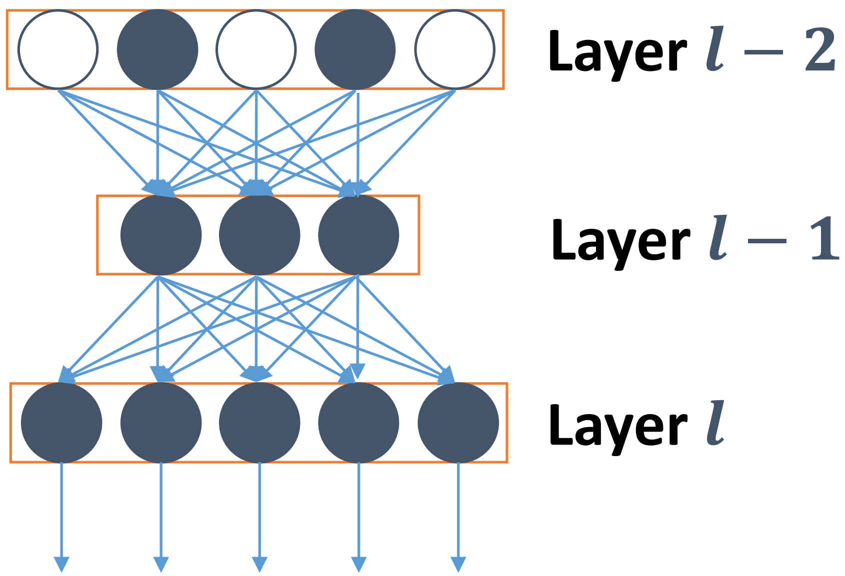
Implicit in **Raghu et al.** (2017):
For a rectifier DNN, there are at most

$$\prod_{l=1}^{L} \sum_{j=0}^{n_{l-1}} \binom{n_l}{j}$$

linear regions.

Propagating Dimensions through Width

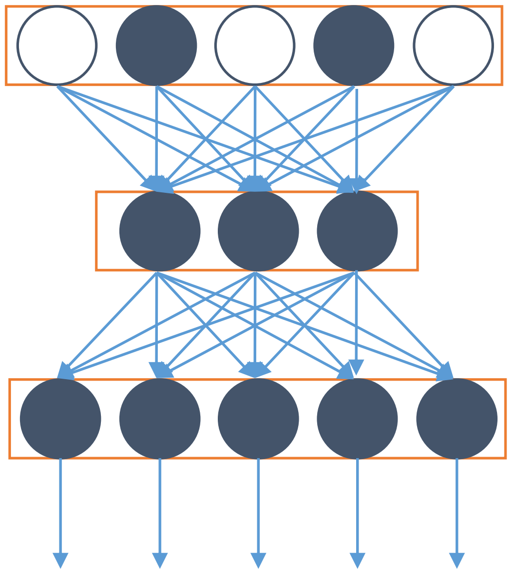
A layer with small width restricts the dimension of hyperplane arrangements in subsequent layers



Propagating Dimensions through Width

A layer with small width restricts the dimension of hyperplane arrangements in subsequent layers

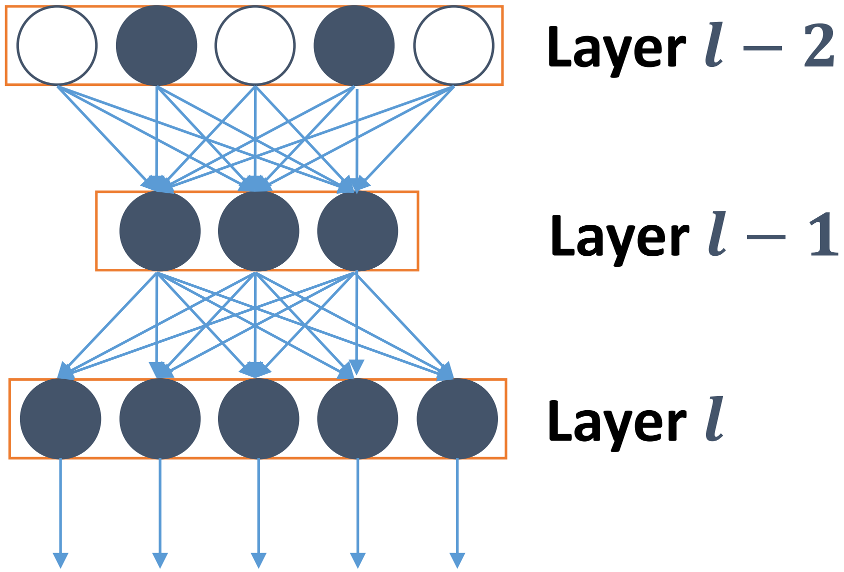
- For layer l, we have 5 hyperplanes in dimension 3



Propagating Dimensions through Width

A layer with small width restricts the dimension of hyperplane arrangements in subsequent layers

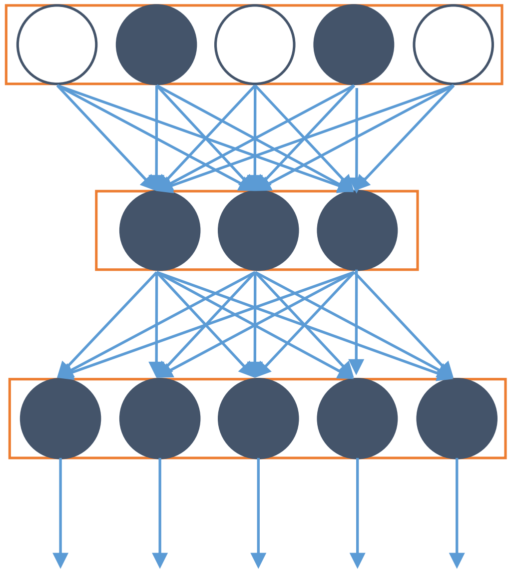
- For layer l, we have 5 hyperplanes in dimension 3
- In fact, the output $\mathbf{h}^{l}$ is contained in a 3D region



Propagating Dimensions through Width

A layer with small width restricts the dimension of hyperplane arrangements in subsequent layers

- For layer $\boldsymbol{l}$, we have 5 hyperplanes in dimension 3
- In fact, the output $\mathbf{h}^{\mathbf{l}}$ is contained in a 3D region



More generality, the maximum dimension of the arrangement in layer $\boldsymbol{l}$ is

$$\boldsymbol{d_{l-1} = min\{n_0, n_1, \ldots, n_{l-1}\}}$$

Consequence to the Upper Bound, Act 2

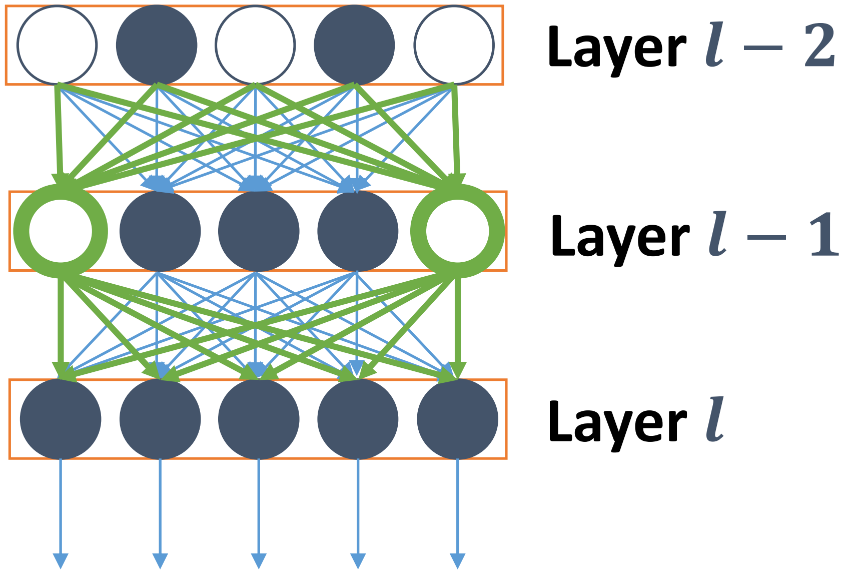
Montufar **(2017)**: For a rectifier DNN, there are at most

$$\prod_{l=1}^{L} \sum_{j=0}^{d_l} \binom{n_l}{j}$$

linear regions.

Refining Dimensions through Activation Patterns

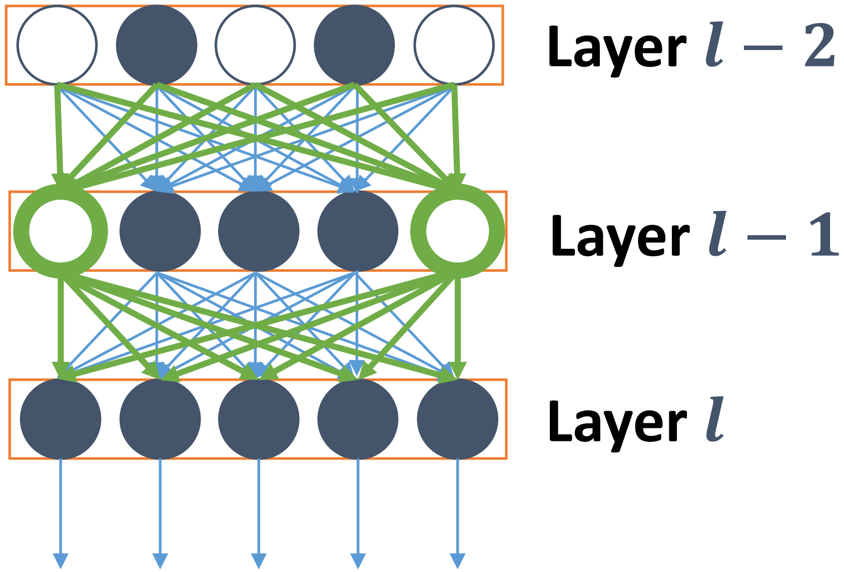
In the last example, nothing changes if layer $l-1$ has extra inactive units



Refining Dimensions through Activation Patterns

In the last example, nothing changes if layer $\boldsymbol{l-1}$ has extra <u>inactive</u> units

In fact, we could make stronger statements:

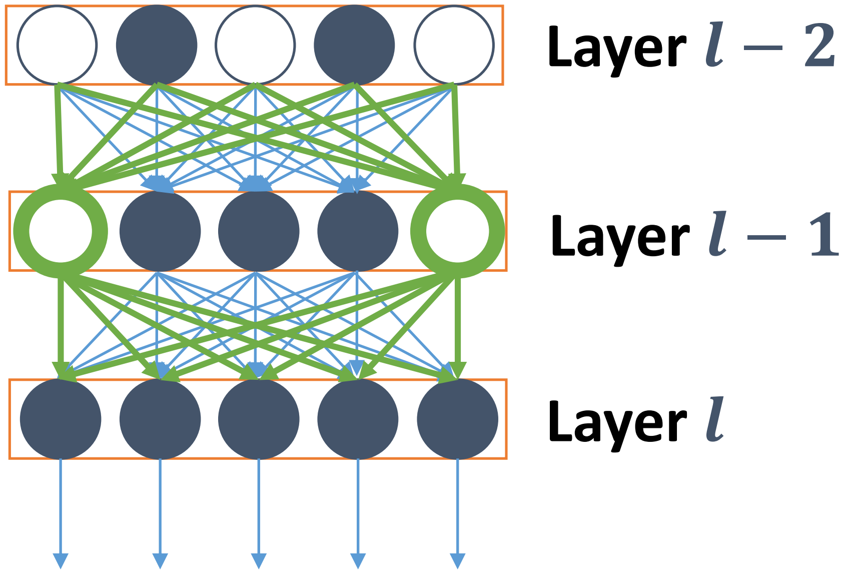


Refining Dimensions through Activation Patterns

In the last example, nothing changes if layer $\boldsymbol{l-1}$ has extra inactive units

In fact, we could make stronger statements:

- Given $\boldsymbol{S}^{\boldsymbol{l-2}}$, the arrangement in layer $\boldsymbol{l-1}$ consists of 5 hyperplanes in dimension 2

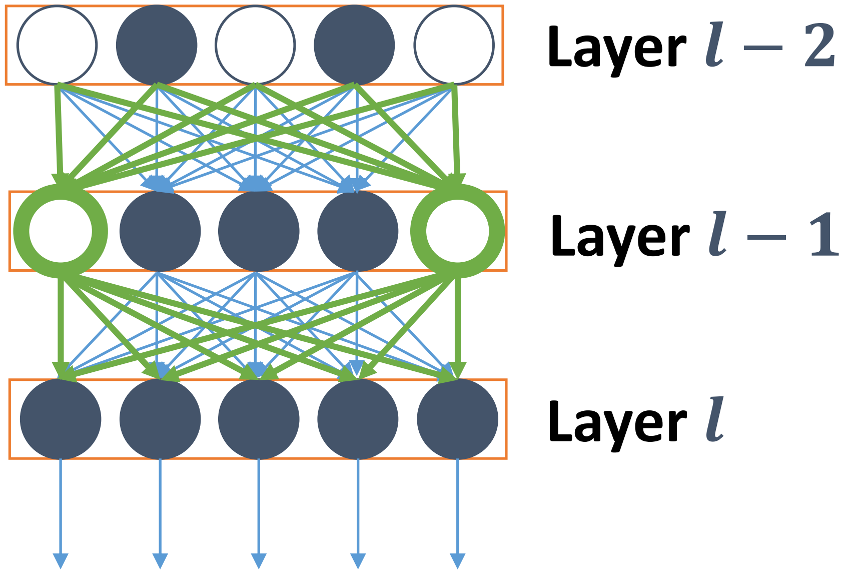


Refining Dimensions through Activation Patterns

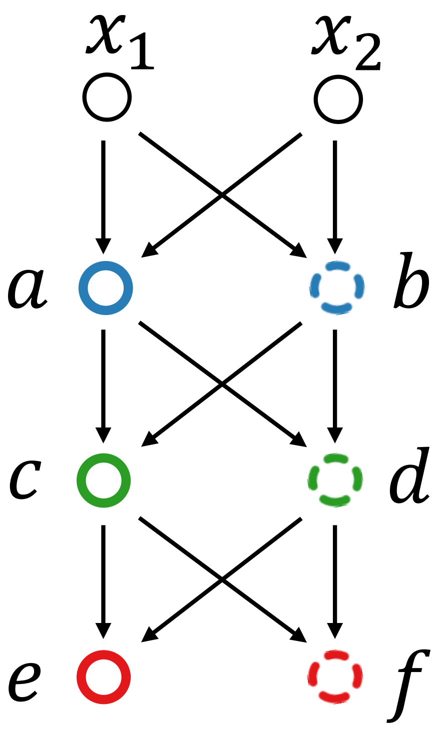
In the last example, nothing changes if layer $l-1$ has extra inactive units

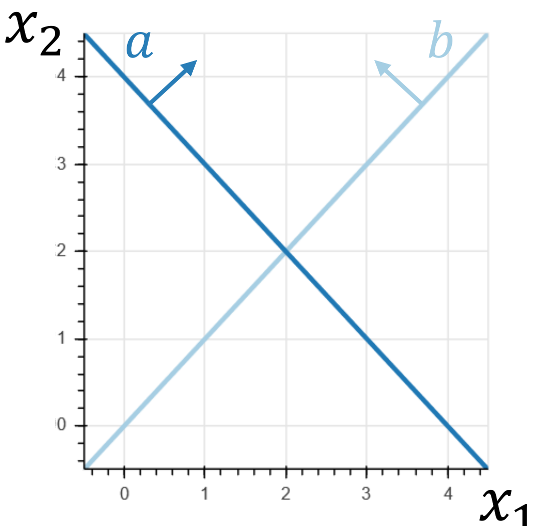
In fact, we could make stronger statements:

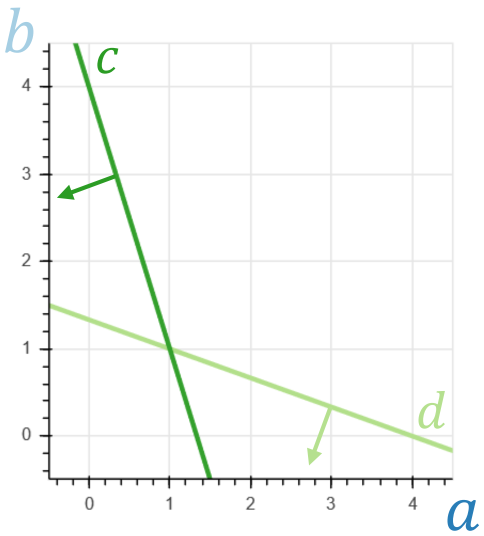
- Given S^{l-2}, the arrangement in layer $l-1$ consists of 5 hyperplanes in dimension 2
- Hence, for that S^{l-2}, outputs $\mathbf{h}^{l-1}$ and $\mathbf{h}^{l}$ are both contained in 2D regions

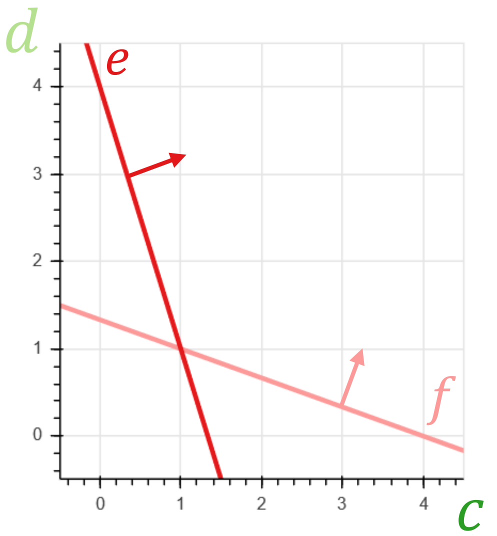


How This Looks in Practice

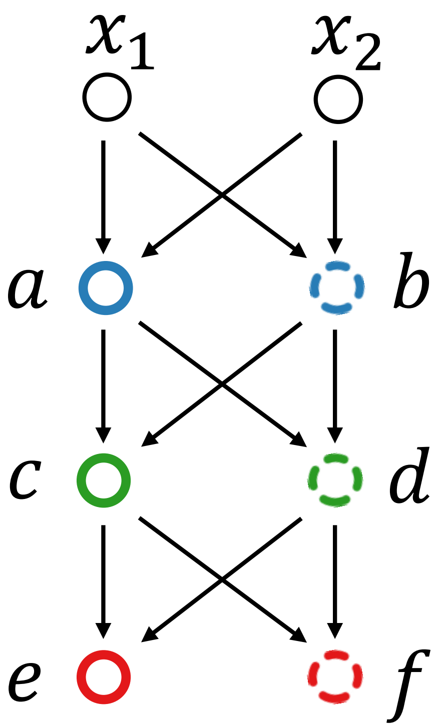


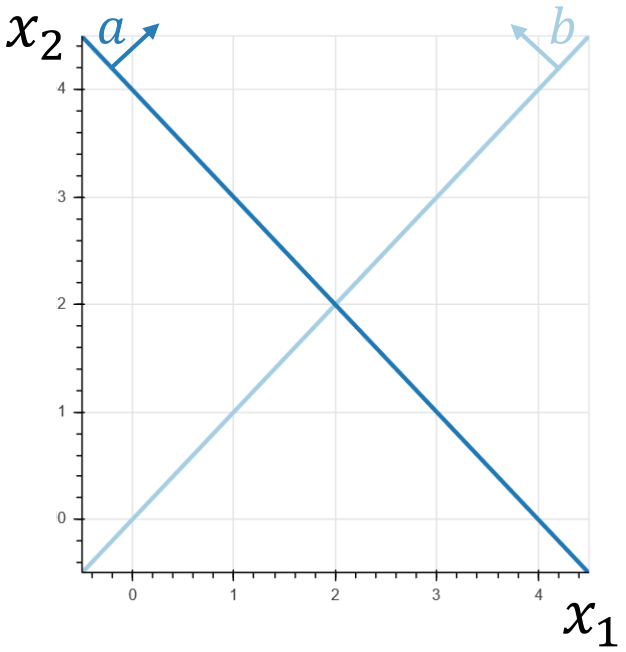


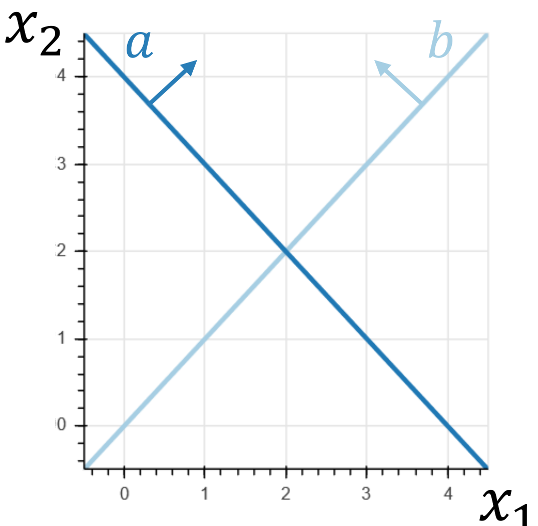


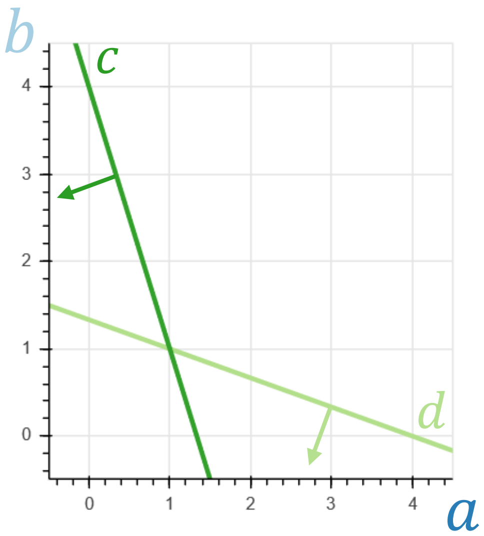


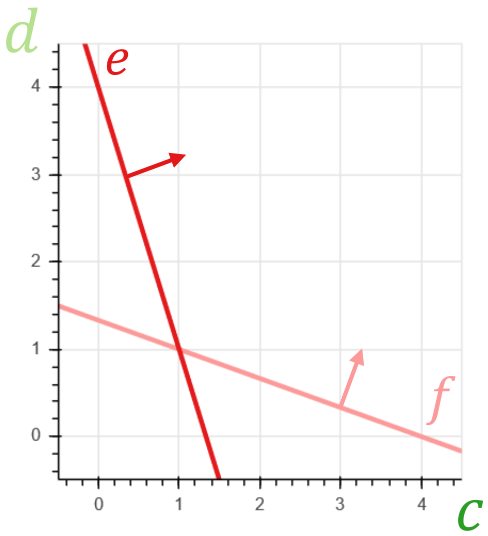
How This Looks in Practice



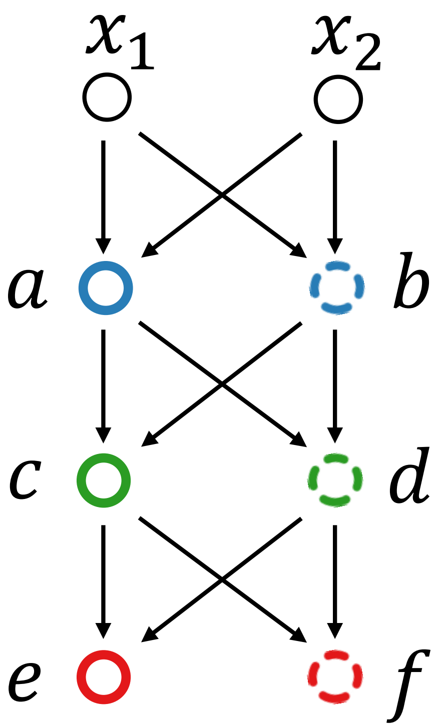


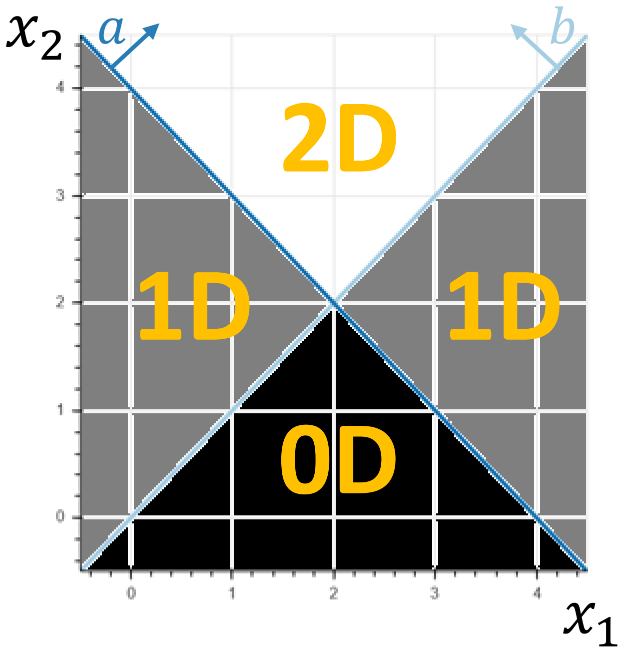


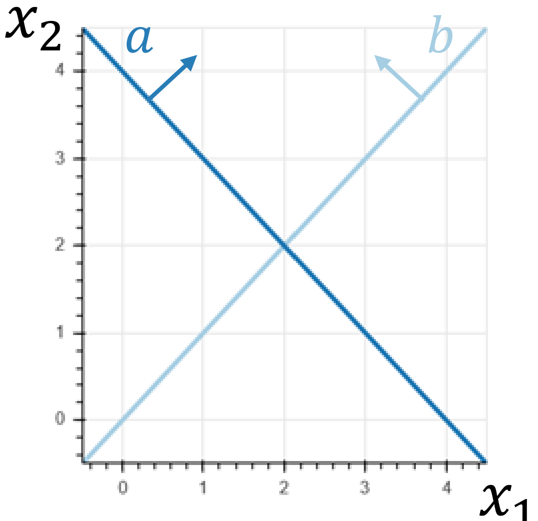


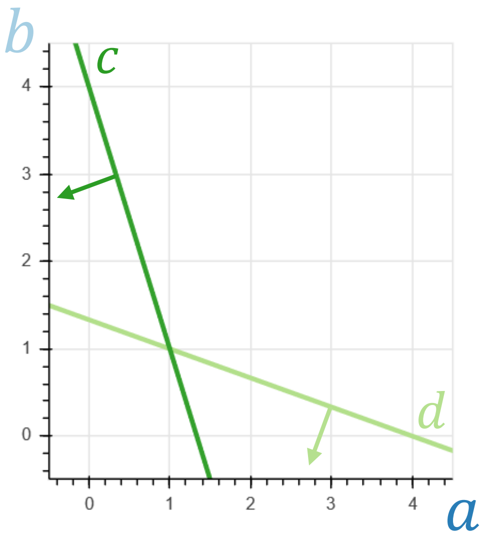


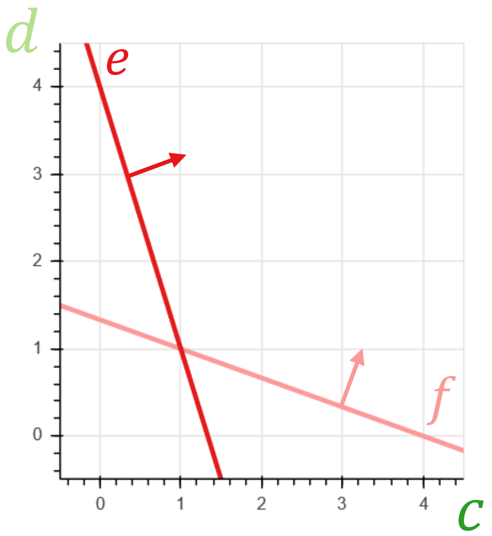
How This Looks in Practice



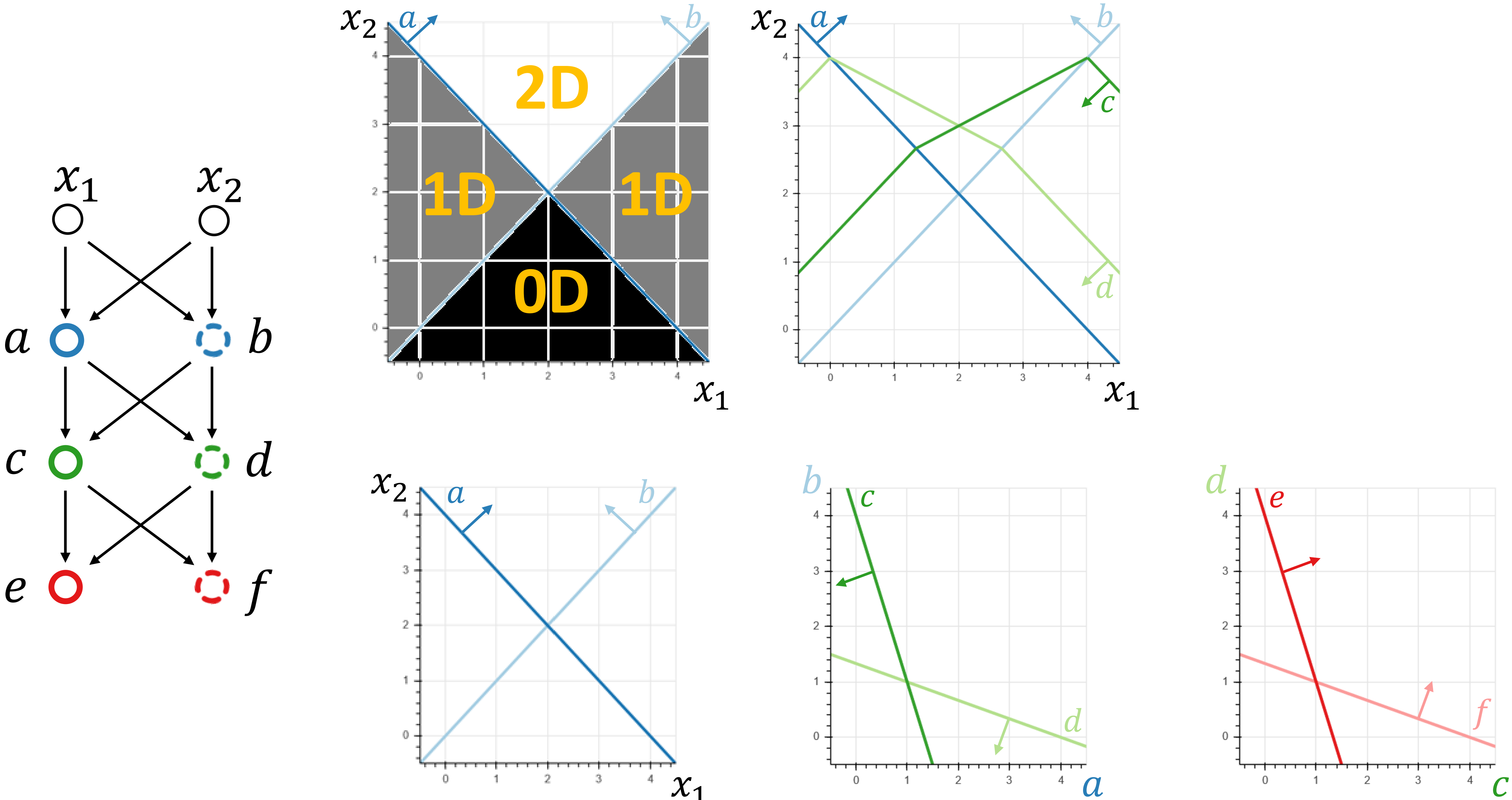




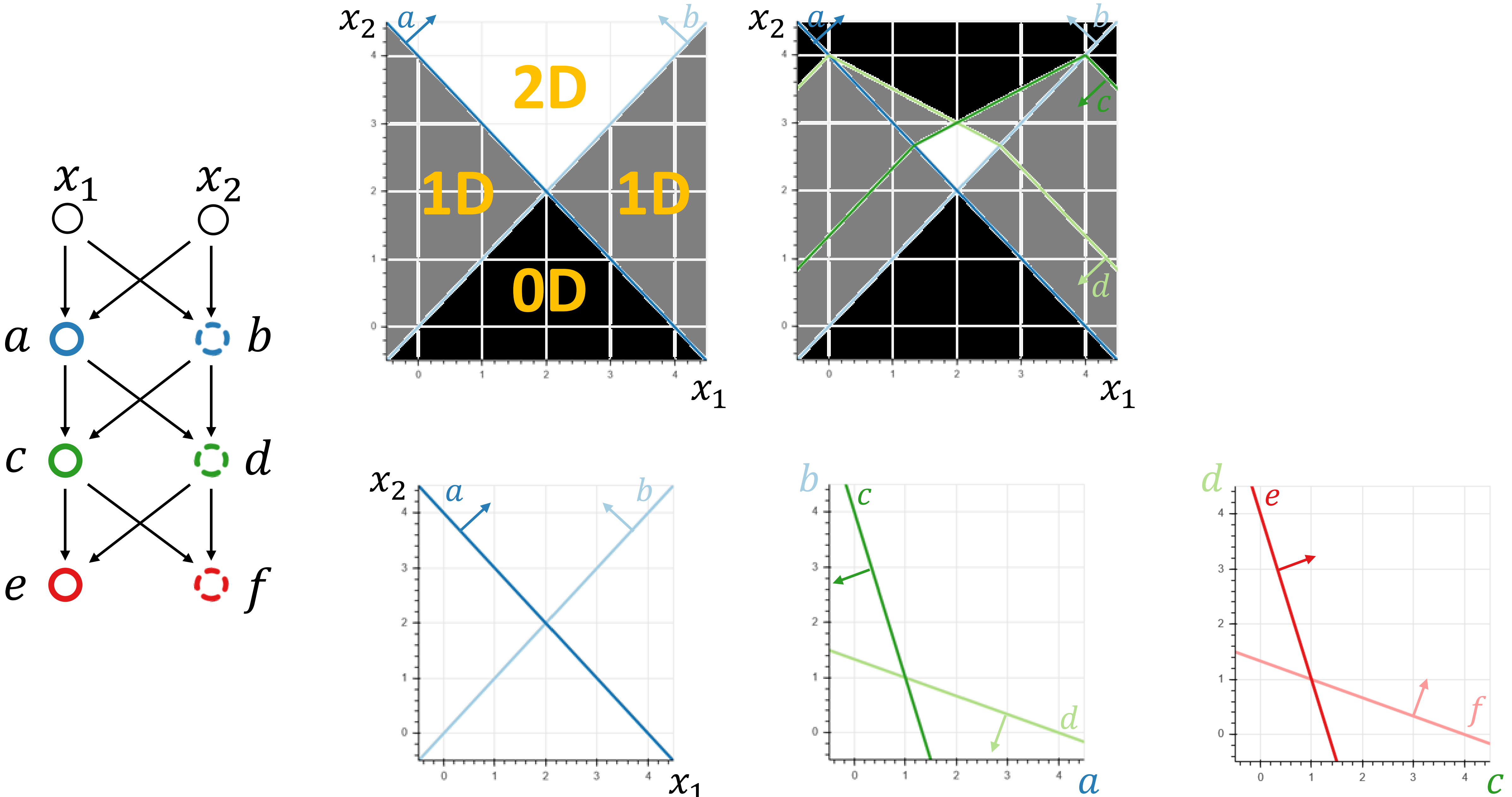




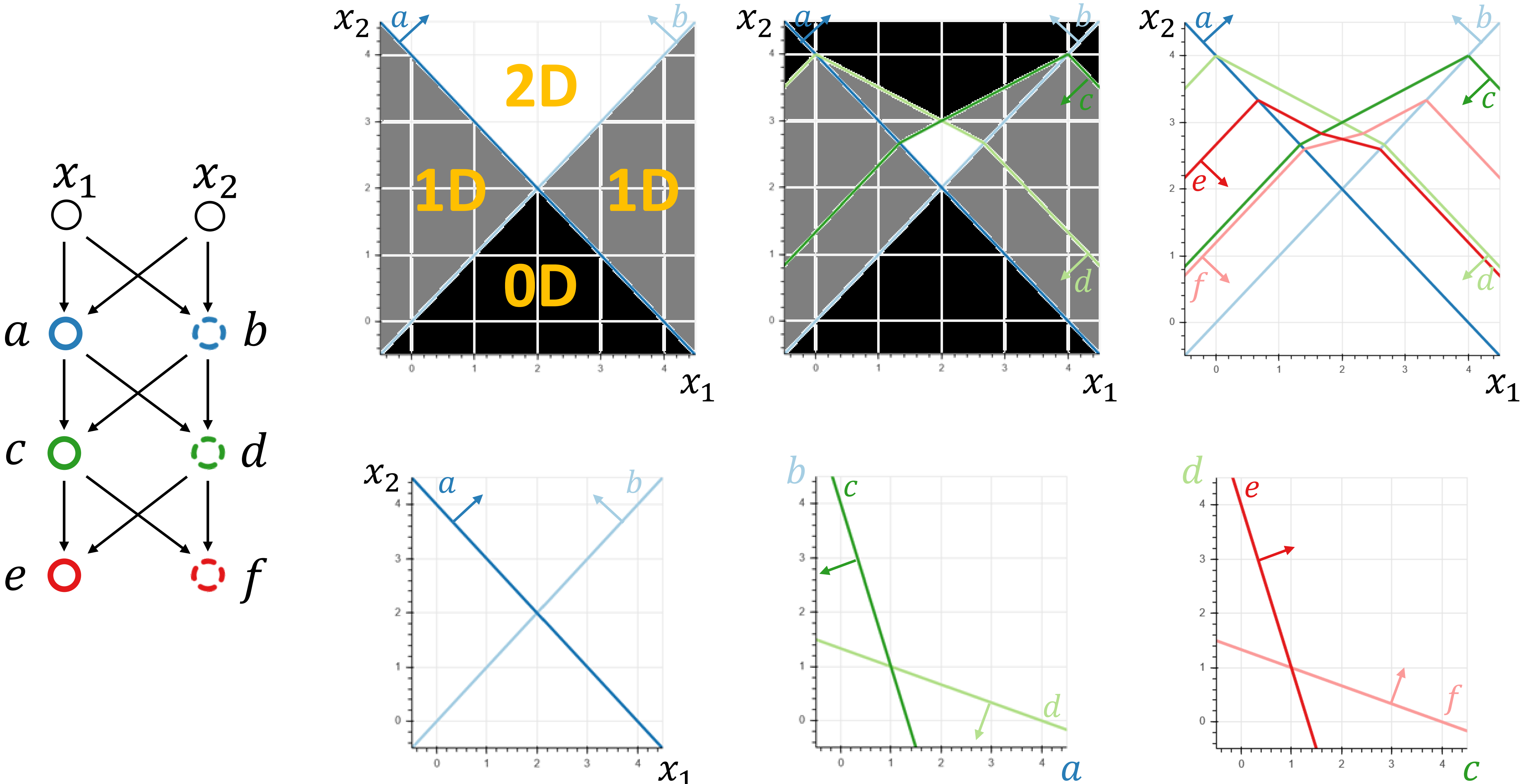
How This Looks in Practice



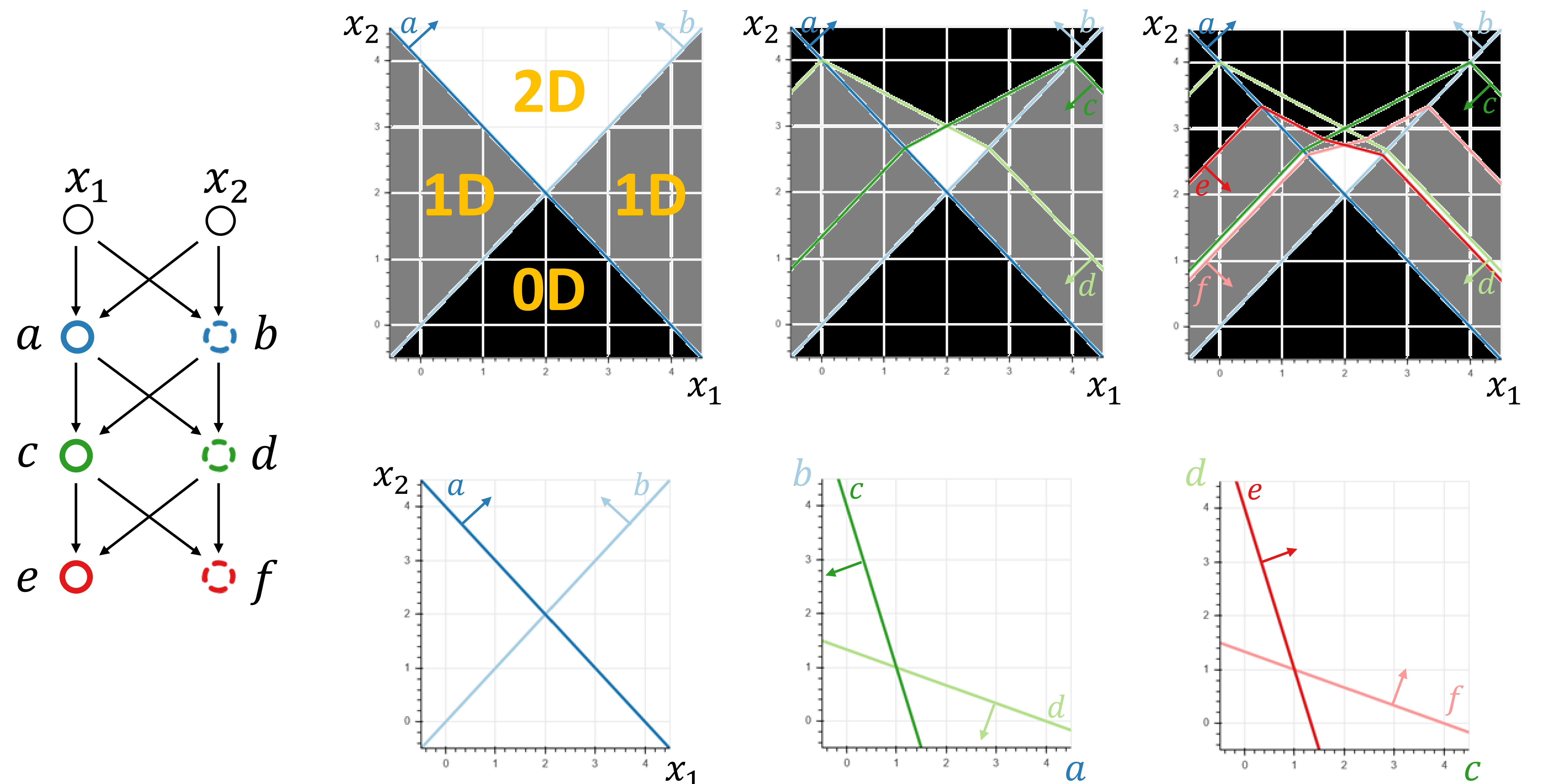
How This Looks in Practice



How This Looks in Practice



How This Looks in Practice



Bounding Deep Networks, Act 3

Theorem 1 **(S., Tjandraatmadja, Ramalingam 2018a)**: For a rectifier DNN, there are at most

$$\sum_{(j_1,\ldots,j_L)\in J}\prod_{l=1}^{L}\binom{n_l}{j_l}$$

linear regions, where

$J = \{(j_1, \ldots, j_L) \in \mathbb{Z}^L : 0 \le j_l \le \min\{n_0, n_1 - j_1, \ldots, n_{l-1} - j_{l-1}, n_l\}\ \forall l = 1, \ldots, L\}$.

Bounding Deep Networks, Act 3

Theorem 1 **(S., Tjandraatmadja, Ramalingam 2018a)**: For a rectifier DNN, there are at most

$$\sum_{(j_1,\ldots,j_L)\in J} \prod_{l=1}^{L} \binom{n_l}{j_l}$$

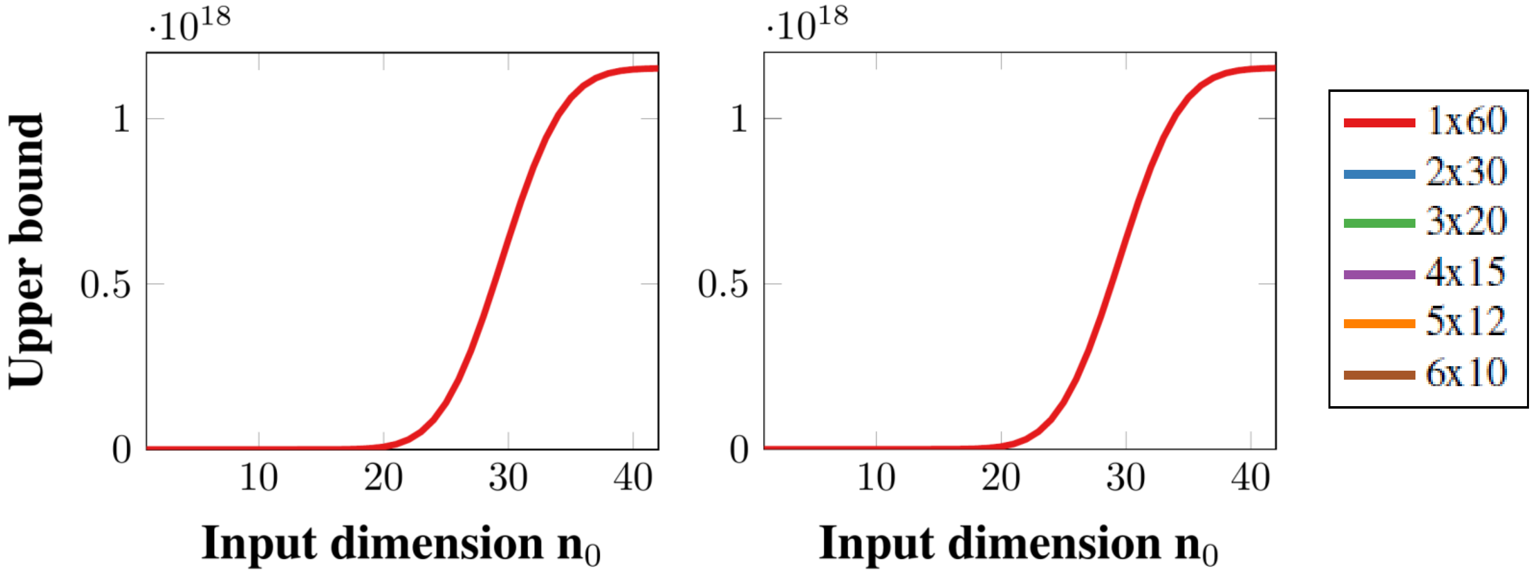
linear regions, where

$J = \{(j_1, \ldots, j_L) \in \mathbb{Z}^L : 0 \leq j_l \leq \min\{n_0, n_1 - j_1, \ldots, n_{l-1} - j_{l-1}, n_l\} \ \forall l = 1, \ldots, L\}$.

This bound is tight when $n_0 = 1$

Insights from the New Upper Bound

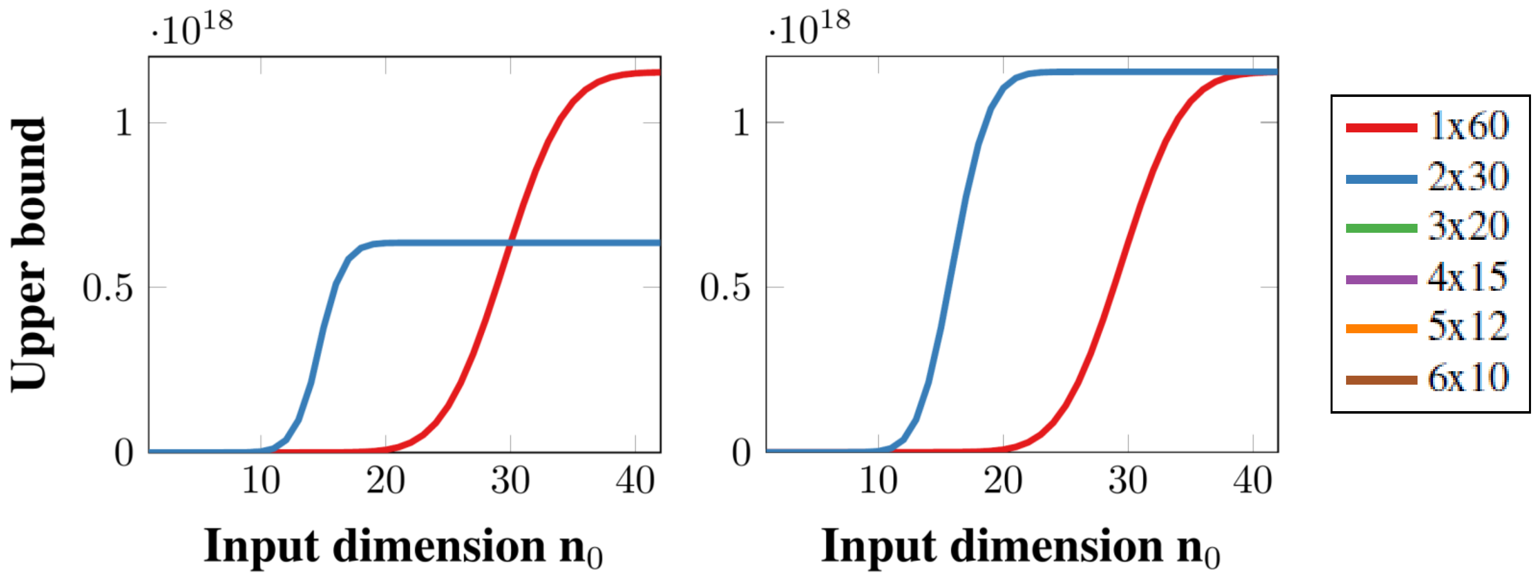
We uniformly distribute 60 units in 1 to 6 layers and vary input dimension



(a) Theorem 1 (b) Montúfar (2017)

Insights from the New Upper Bound

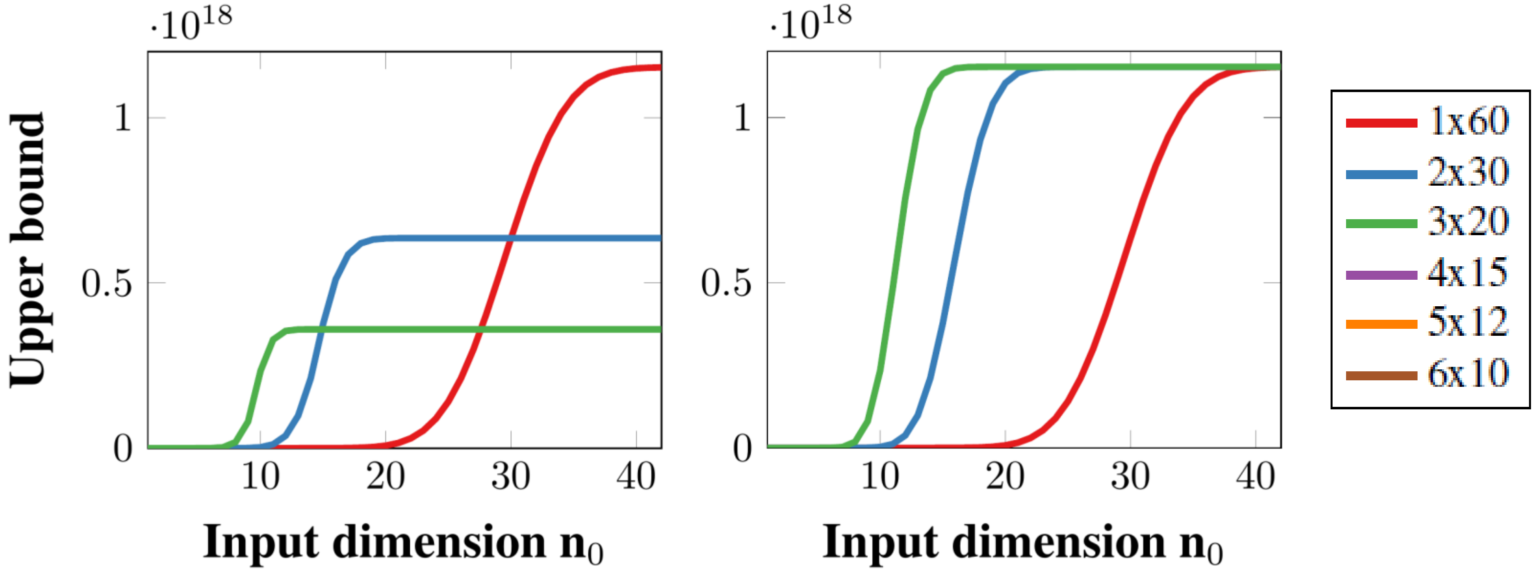
We uniformly distribute 60 units in 1 to 6 layers and vary input dimension



(a) Theorem 1 (b) Montúfar (2017)

Insights from the New Upper Bound

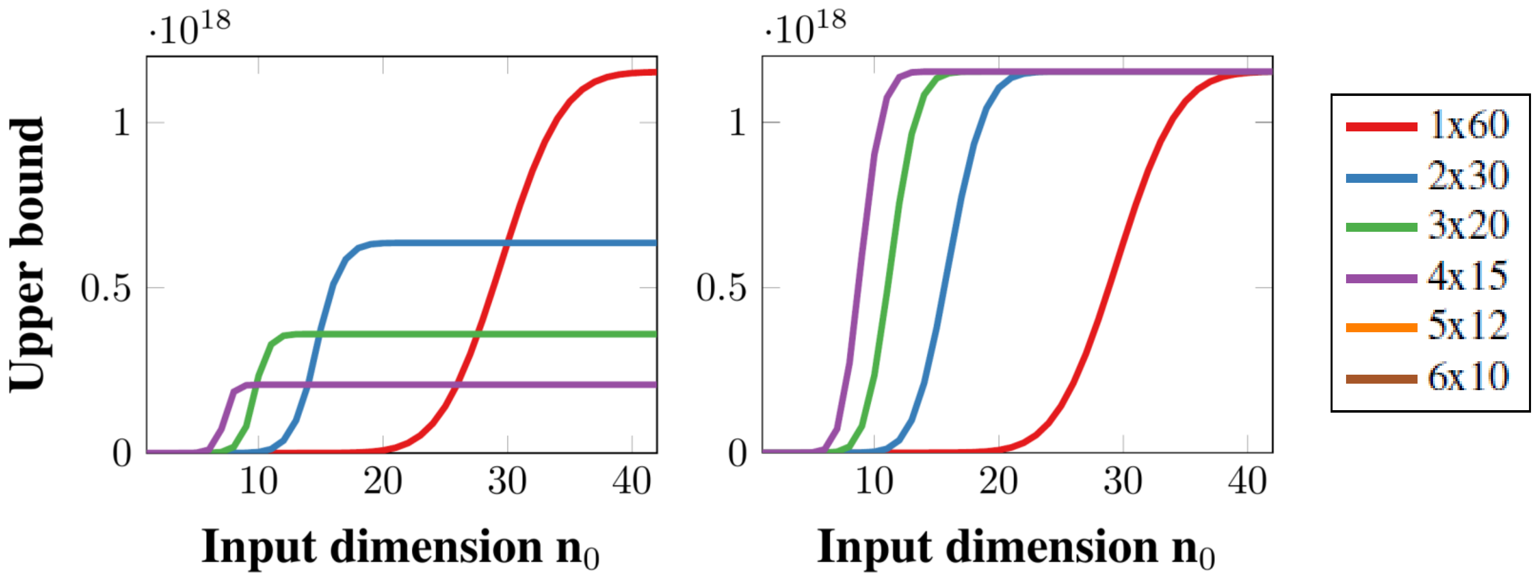
We uniformly distribute 60 units in 1 to 6 layers and vary input dimension



(a) Theorem 1 (b) Montúfar (2017)

Insights from the New Upper Bound

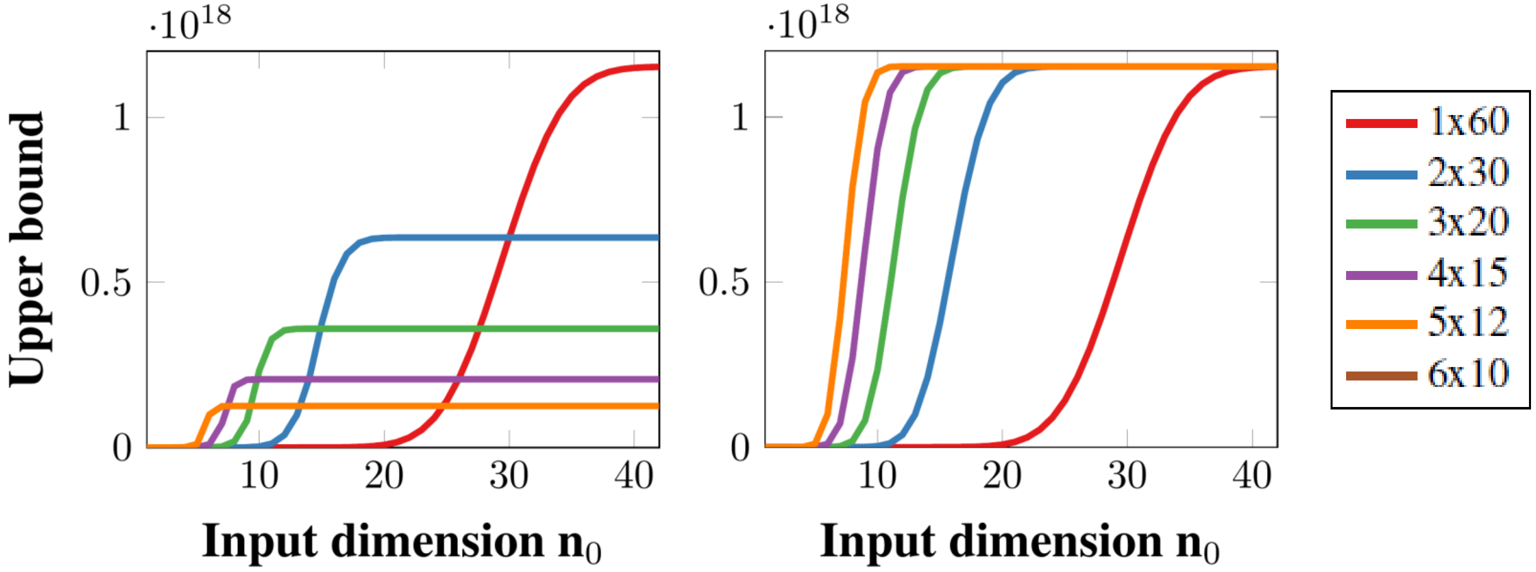
We uniformly distribute 60 units in 1 to 6 layers and vary input dimension



(a) Theorem 1 (b) Montúfar (2017)

Insights from the New Upper Bound

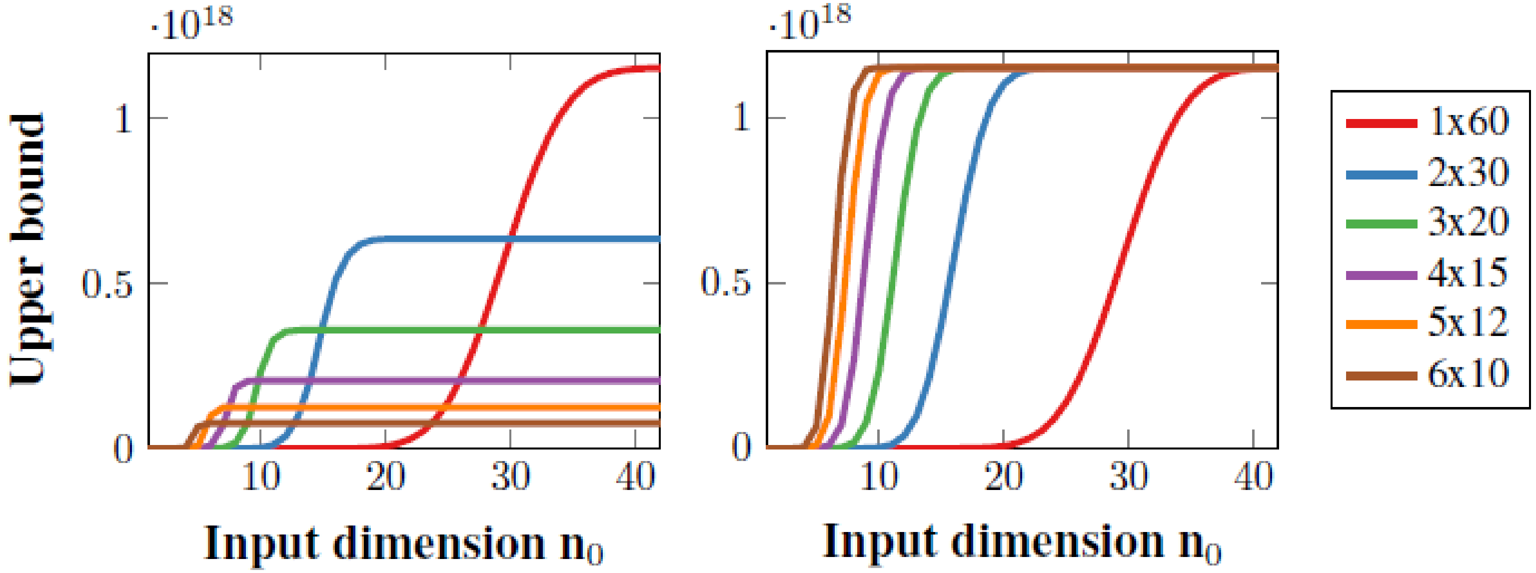
We uniformly distribute 60 units in 1 to 6 layers and vary input dimension



(a) Theorem 1 (b) Montúfar (2017)

Insights from the New Upper Bound

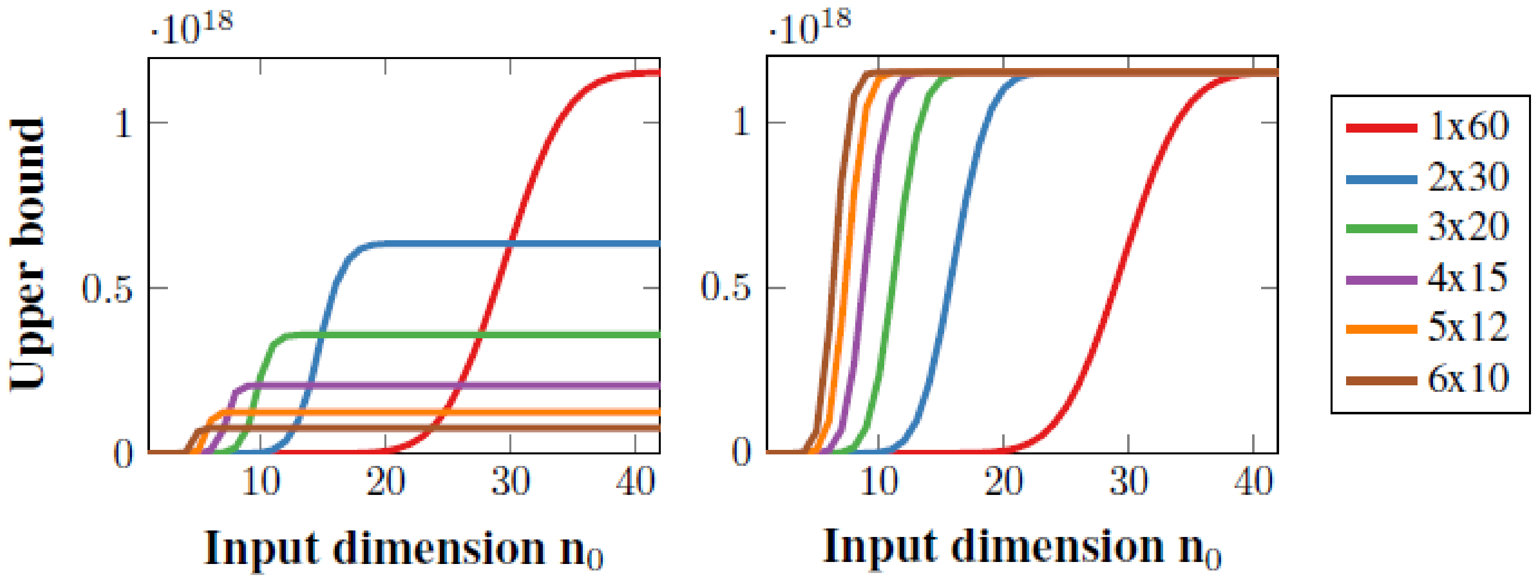
We uniformly distribute 60 units in 1 to 6 layers and vary input dimension



(a) Theorem 1 (b) Montúfar (2017)

Insights from the New Upper Bound

We uniformly distribute 60 units in 1 to 6 layers and vary input dimension

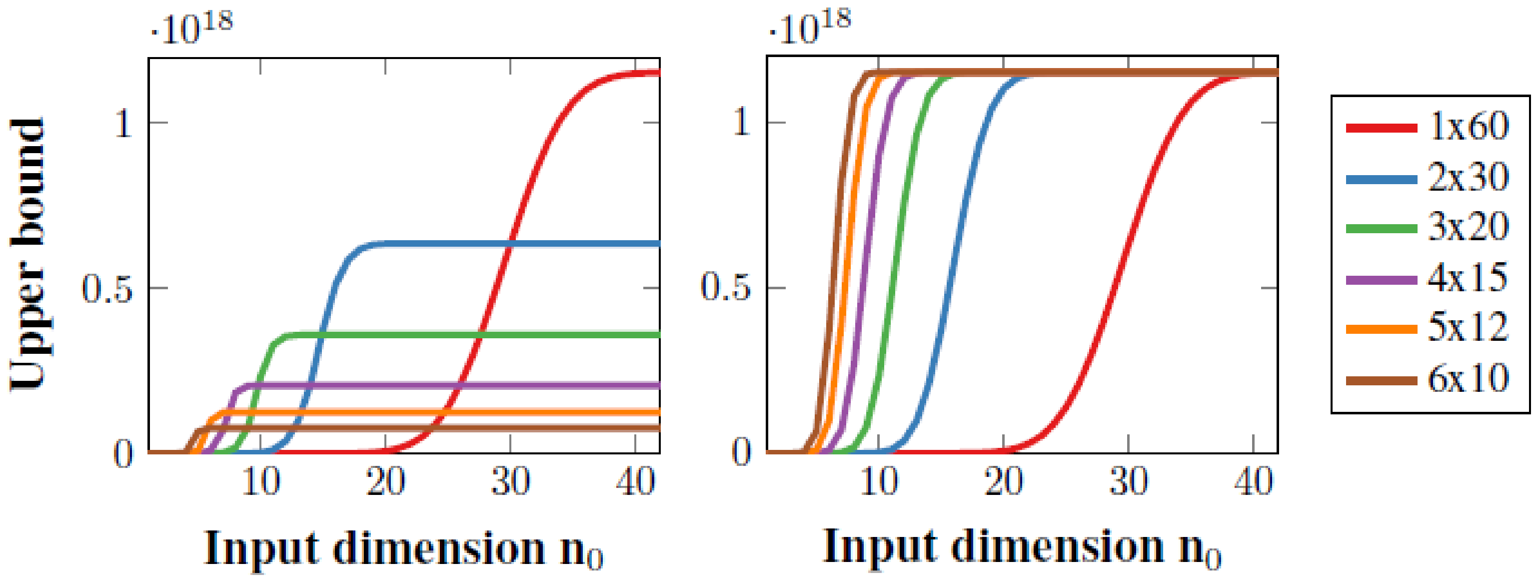


(a) Theorem 1 (b) Montúfar (2017)

When the <u>input dimension is very large</u>, shallow networks have more LRs

Insights from the New Upper Bound

We uniformly distribute 60 units in 1 to 6 layers and vary input dimension



(a) Theorem 1 (b) Montúfar (2017)

When the <u>input dimension is very large</u>, shallow networks have more LRs

For a <u>fixed input dimension</u>, there is a depth that maximizes the bound

Exact Counting on Rectifier Networks

- MILP-based procedure to enumerate linear regions

Linear Regions and Polyhedra

For ReLUs, given a pattern $\boldsymbol{S}$, we can first represented the linear region in the lifted space $\boldsymbol{x}, \boldsymbol{h}^1, \dots, \boldsymbol{h}^{L-1}, \boldsymbol{y}$:

$$\boldsymbol{h}_i^l = \boldsymbol{W}_i^l \boldsymbol{h}^{l-1} + \boldsymbol{b}_i^l > \boldsymbol{0} \quad \forall\, \boldsymbol{i} \in \boldsymbol{S}^l, \boldsymbol{l} \in \{\boldsymbol{1}, \dots, \boldsymbol{L}\}$$

$$\begin{aligned} &\boldsymbol{W}_i^l \boldsymbol{h}^{l-1} + \boldsymbol{b}_i^l \le \boldsymbol{0} && \forall\, \boldsymbol{i} \notin \boldsymbol{S}^l, \boldsymbol{l} \in \{\boldsymbol{1}, \dots, \boldsymbol{L}\} \\ &\boldsymbol{h}_i^l = \boldsymbol{0} && \forall\, \boldsymbol{i} \notin \boldsymbol{S}^l, \boldsymbol{l} \in \{\boldsymbol{1}, \dots, \boldsymbol{L}\} \end{aligned}$$

Linear Regions and Polyhedra

If we slightly relax the definition of active units (**borders overlap**), each linear region corresponds to a polyhedron in $\boldsymbol{x}, \boldsymbol{h}^1, \ldots, \boldsymbol{h}^{L-1}, \boldsymbol{y}$:

$$\boldsymbol{h}_i^l = \boldsymbol{W}_i^l \boldsymbol{h}^{l-1} + \boldsymbol{b}_i^l \geq \boldsymbol{0} \quad \forall\, \boldsymbol{i} \in \boldsymbol{S}^l, \boldsymbol{l} \in \{\boldsymbol{1}, \ldots, \boldsymbol{L}\}$$

$$\boldsymbol{W}_i^l \boldsymbol{h}^{l-1} + \boldsymbol{b}_i^l \leq \boldsymbol{0} \quad \forall\, \boldsymbol{i} \notin \boldsymbol{S}^l, \boldsymbol{l} \in \{\boldsymbol{1}, \ldots, \boldsymbol{L}\}$$

$$\boldsymbol{h}_i^l = \boldsymbol{0} \quad \forall\, \boldsymbol{i} \notin \boldsymbol{S}^l, \boldsymbol{l} \in \{\boldsymbol{1}, \ldots, \boldsymbol{L}\}$$

A Disjunctive Program

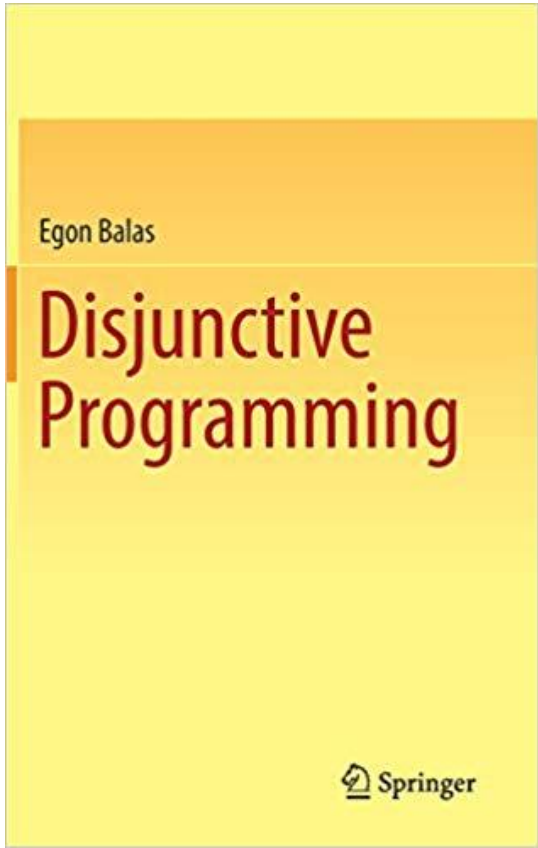
The union of the polyhedra corresponding to the sets of activation patterns is a disjunctive program, which can be translated to a MILP formulation

$$\bigvee_{(\mathcal{S}^1,\dots,\mathcal{S}^L)\in\mathcal{S}} \left(\begin{array}{ll} h_i^l = W_i^l h^{l-1} + b_i^l \geq 0 & \forall\, i \in S^l, l \in \{1,\dots,L\} \\ W_i^l h^{l-1} + b_i^l \leq 0 & \forall\, i \notin S^l, l \in \{1,\dots,L\} \\ h_i^l = 0 & \forall\, i \notin S^l, l \in \{1,\dots,L\} \end{array} \right)$$

A Disjunctive Program

The union of the polyhedra corresponding to the sets of activation patterns is a disjunctive program, which can be translated to a MILP formulation

$$\bigvee_{(S^1,\ldots,S^L)\in\mathcal{S}} \begin{array}{ll} h_i^l = W_i^l h^{l-1} + b_i^l \geq 0 & \forall\, i \in S^l, l \in \{1,\ldots,L\} \\ W_i^l h^{l-1} + b_i^l \leq 0 & \forall\, i \notin S^l, l \in \{1,\ldots,L\} \\ h_i^l = 0 & \forall\, i \notin S^l, l \in \{1,\ldots,L\} \end{array}$$

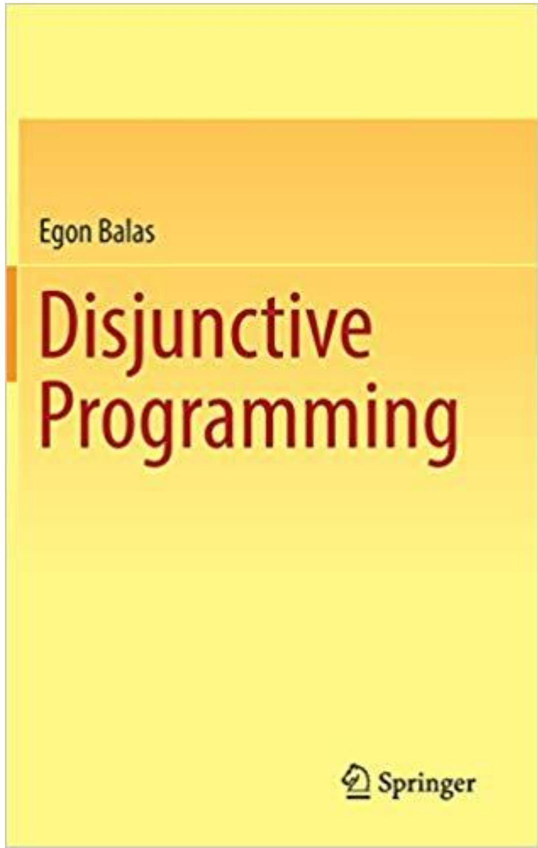


A Disjunctive Program

The union of the polyhedra corresponding to the sets of activation patterns is a disjunctive program, which can be translated to a MILP formulation

$$\bigvee_{(\mathcal{S}^1,\ldots,\mathcal{S}^L)\in\mathcal{S}} \begin{array}{ll} \boldsymbol{h}_i^l = \boldsymbol{W}_i^l \boldsymbol{h}^{l-1} + \boldsymbol{b}_i^l \geq 0 & \forall\, \boldsymbol{i} \in \boldsymbol{S}^l, \boldsymbol{l} \in \{1,\ldots,\boldsymbol{L}\} \\ \boldsymbol{W}_i^l \boldsymbol{h}^{l-1} + \boldsymbol{b}_i^l \leq 0 & \forall\, \boldsymbol{i} \notin \boldsymbol{S}^l, \boldsymbol{l} \in \{1,\ldots,\boldsymbol{L}\} \\ \boldsymbol{h}_i^l = 0 & \forall\, \boldsymbol{i} \notin \boldsymbol{S}^l, \boldsymbol{l} \in \{1,\ldots,\boldsymbol{L}\} \end{array}$$

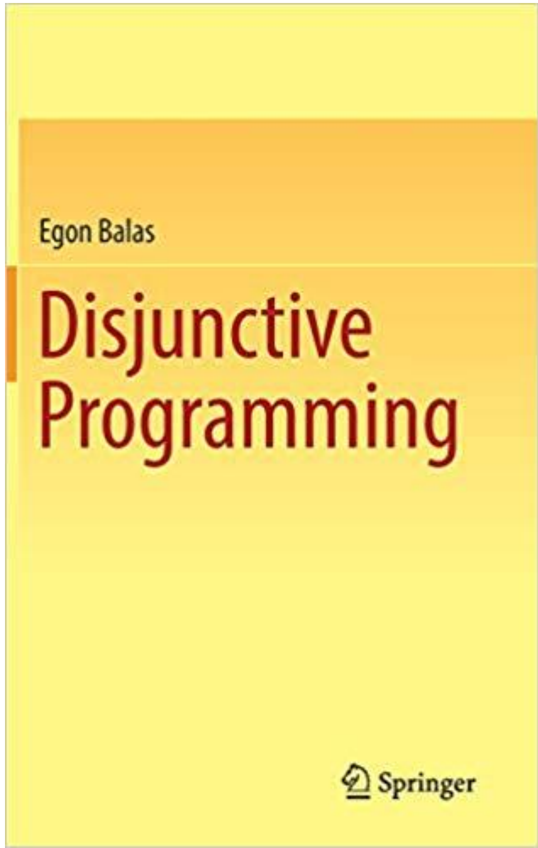
We obtain the polyhedron in $\boldsymbol{x}$ by Fourier-Motzkin elimination



A Disjunctive Program

The union of the polyhedra corresponding to the sets of activation patterns is a disjunctive program, which can be translated to a MILP formulation

$$\bigvee_{(\boldsymbol{S}^1, \dots, \boldsymbol{S}^L) \in \mathcal{S}} \begin{array}{ll} \boldsymbol{h}_i^l = \boldsymbol{W}_i^l \boldsymbol{h}^{l-1} + \boldsymbol{b}_i^l \geq \mathbf{0} & \forall\, \boldsymbol{i} \in \boldsymbol{S}^l, \boldsymbol{l} \in \{\mathbf{1}, \dots, \boldsymbol{L}\} \\ \boldsymbol{W}_i^l \boldsymbol{h}^{l-1} + \boldsymbol{b}_i^l \leq \mathbf{0} & \forall\, \boldsymbol{i} \notin \boldsymbol{S}^l, \boldsymbol{l} \in \{\mathbf{1}, \dots, \boldsymbol{L}\} \\ \boldsymbol{h}_i^l = \mathbf{0} & \forall\, \boldsymbol{i} \notin \boldsymbol{S}^l, \boldsymbol{l} \in \{\mathbf{1}, \dots, \boldsymbol{L}\} \end{array}$$



We obtain the polyhedron in $\boldsymbol{x}$ by Fourier-Motzkin elimination

We find all linear regions using a mixed-integer formulation

Mapping Inputs to Outputs on Units

The following constraints represent a ReLU $\boldsymbol{i}$ in layer $\boldsymbol{l}$:

$$\boldsymbol{W_i^l h^{l-1}} + \boldsymbol{b_i^l} = \boldsymbol{g_i^l}$$

Mapping Inputs to Outputs on Units

The following constraints represent a ReLU $\boldsymbol{i}$ in layer $\boldsymbol{l}$:

$$\boldsymbol{W_i^l h^{l-1} + b_i^l = g_i^l}$$
$$\boldsymbol{g_i^l = h_i^l - \overline{h}_i^l}$$

- $\boldsymbol{\overline{h}_i^l}$ is the output of a fictitious complementary unit

Mapping Inputs to Outputs on Units

The following constraints represent a ReLU $\boldsymbol{i}$ in layer $\boldsymbol{l}$:

$$\boldsymbol{W}_i^l \boldsymbol{h}^{l-1} + \boldsymbol{b}_i^l = \boldsymbol{g}_i^l$$
$$\boldsymbol{g}_i^l = \boldsymbol{h}_i^l - \overline{\boldsymbol{h}}_i^l$$
$$\boldsymbol{h}_i^l \geq \boldsymbol{0}$$
$$\overline{\boldsymbol{h}}_i^l \geq \boldsymbol{0}$$

- $\overline{\boldsymbol{h}}_i^l$ is the output of a fictitious complementary unit

Mapping Inputs to Outputs on Units

The following constraints represent a ReLU $\boldsymbol{i}$ in layer $\boldsymbol{l}$:

$$\boldsymbol{W_i^l h^{l-1} + b_i^l = g_i^l}$$
$$\boldsymbol{g_i^l = h_i^l - \overline{h}_i^l}$$
$$\boldsymbol{h_i^l \geq 0}$$
$$\boldsymbol{\overline{h}_i^l \geq 0}$$
$$\boldsymbol{z_i^l \in \{0, 1\}}$$

- $\boldsymbol{\overline{h}_i^l}$ is the output of a fictitious complementary unit
- $\boldsymbol{z_i^l}$ is a binary variable modeling if the neuron is active

Mapping Inputs to Outputs on Units

The following constraints represent a ReLU $\boldsymbol{i}$ in layer $\boldsymbol{l}$:

$$\boldsymbol{W_i^l h^{l-1} + b_i^l = g_i^l}$$
$$\boldsymbol{g_i^l = h_i^l - \overline{h}_i^l}$$
$$\boldsymbol{h_i^l \geq 0}$$
$$\boldsymbol{\overline{h}_i^l \geq 0}$$
$$\boldsymbol{z_i^l \in \{0, 1\}}$$
$$\boldsymbol{h_i^l \leq H_i^l\, z_i^l}$$
$$\boldsymbol{\overline{h}_i^l \leq \overline{H}_i^l (1 - z_i^l)}$$

- $\boldsymbol{\overline{h}_i^l}$ is the output of a fictitious complementary unit
- $\boldsymbol{z_i^l}$ is a binary variable modeling if the neuron is active
- $\boldsymbol{H_i^l}$ and $\boldsymbol{\overline{H}_i^l}$ are sufficiently large and positive constants (bounded inputs)

Counting LRs as Integer Solutions

The number of LRs of a rectifier DNN corresponds to the number of solutions on $\boldsymbol{z}$ with positive value for the following mixed-integer program:

$$\boldsymbol{max\ f}$$

$$\begin{array}{lll} \text{s.t.} & (\boldsymbol{previous\ constraints}) & \text{for each neuron } i \text{ in layer } l \\ & \boldsymbol{f} \leq \boldsymbol{h}_i^l + \left(\mathbf{1} - \boldsymbol{z}_i^l\right)\boldsymbol{M} & \text{for each neuron } i \text{ in layer } l \\ & \boldsymbol{x} \in \boldsymbol{X} & \end{array}$$

Counting LRs as Integer Solutions

The number of LRs of a rectifier DNN corresponds to the number of solutions on $\boldsymbol{z}$ with positive value for the following mixed-integer program:

$$\max f$$

$$\begin{aligned}
\text{s.t.} \quad & (\textit{previous constraints}) && \text{for each neuron } i \text{ in layer } l \\
& f \le h_i^l + (1 - z_i^l)M && \text{for each neuron } i \text{ in layer } l \\
& x \in X &&
\end{aligned}$$

Similar mixed-integer formulations proposed around the time:

C.-H. Cheng et al. (2017), Fischetti and Jo (2017)

Computational Results

- How theoretical and empirical numbers compare
- How these numbers mean in practice

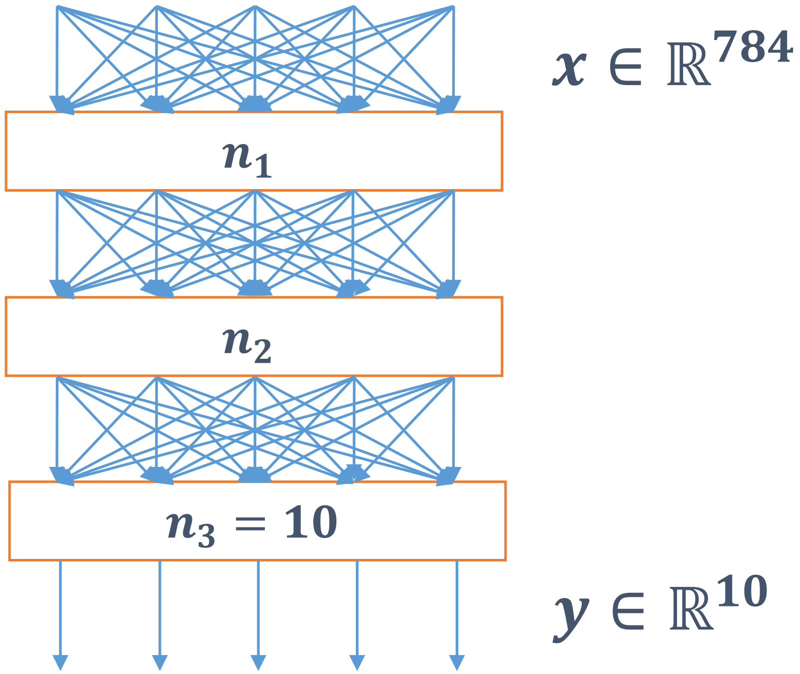
Setup

We trained rectifier networks on the MNIST benchmark

Setup

We trained rectifier networks on the MNIST benchmark

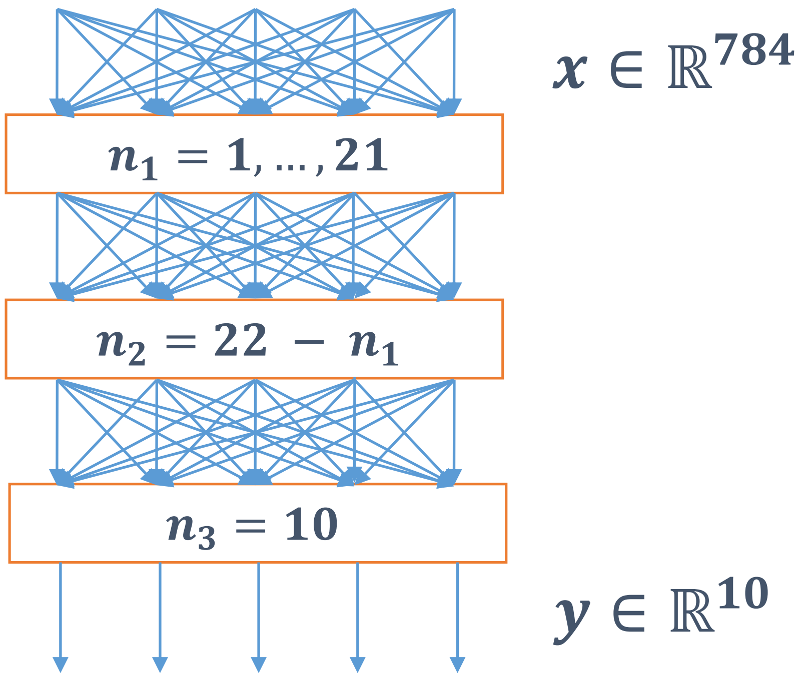
- Input is 28x28, final layer has 10 units (one per digit)



Setup

We trained rectifier networks on the MNIST benchmark

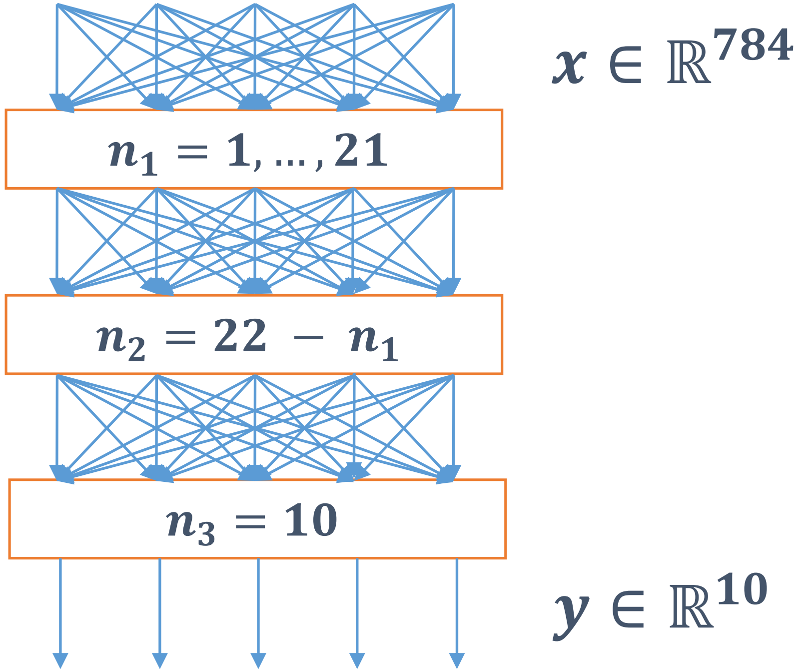
- Input is 28x28, final layer has 10 units (one per digit)
- Two other layers share 22 units



Setup

We trained rectifier networks on the MNIST benchmark

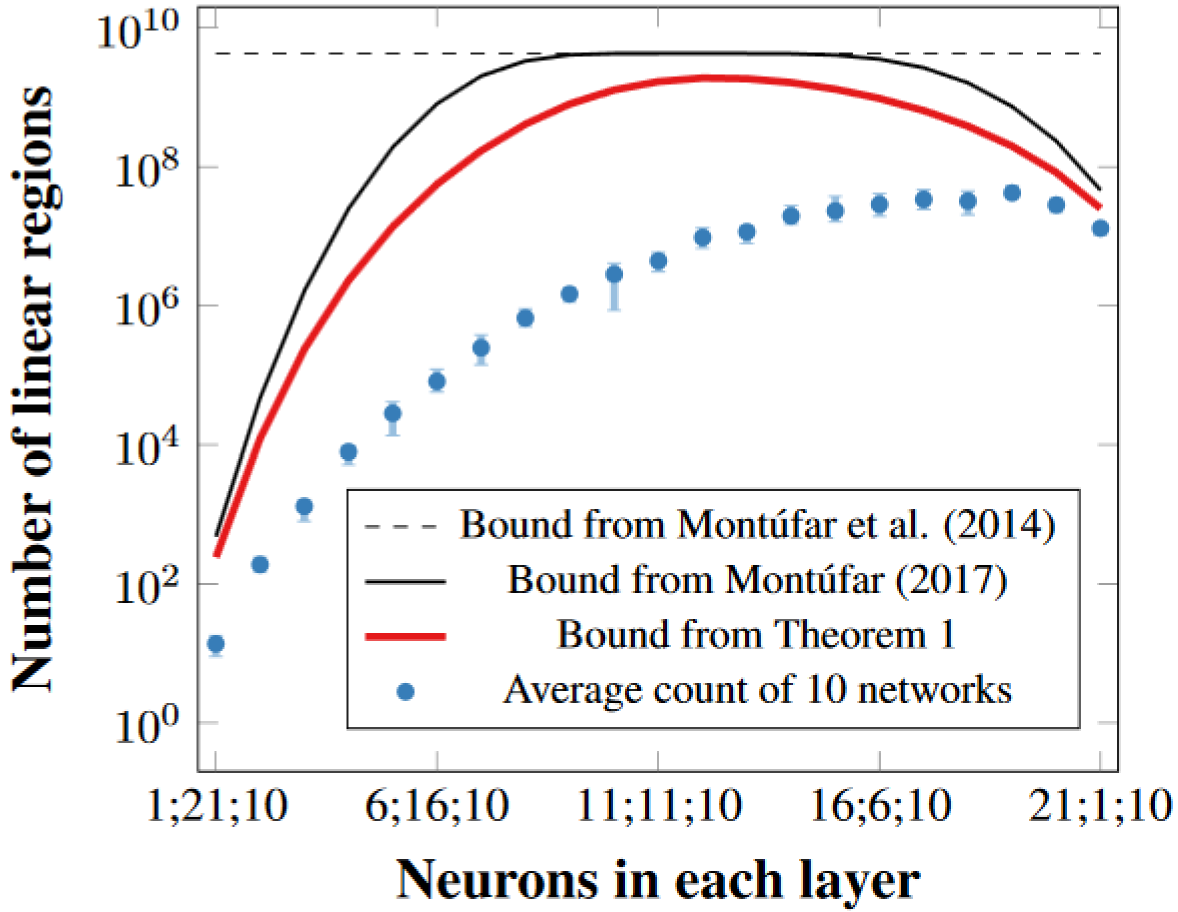
- Input is 28x28, final layer has 10 units (one per digit)
- Two other layers share 22 units
- For each possible configuration, 10 networks were trained and counted



Bounding vs. Counting Results

S., Tjandraatmadja, Ramalingam 2018a

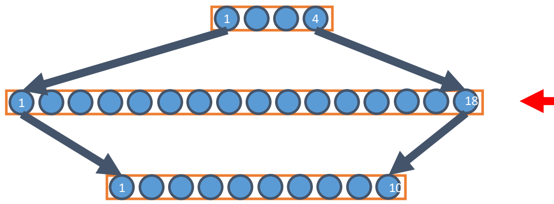
Comparison of bounds with average of 10 networks and min-max bars

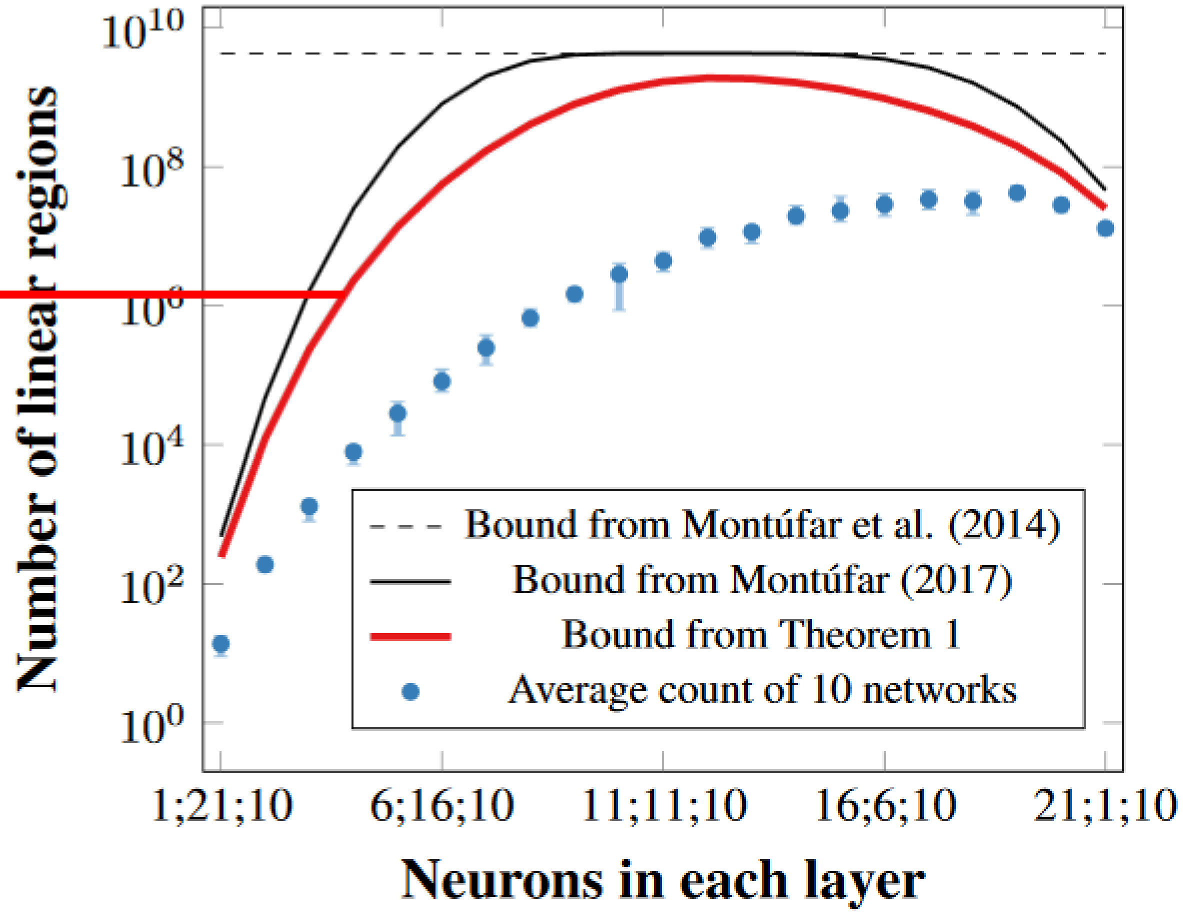


Bounding vs. Counting Results

S., Tjandraatmadja, Ramalingam 2018a

Comparison of bounds with average of 10 networks and min-max bars

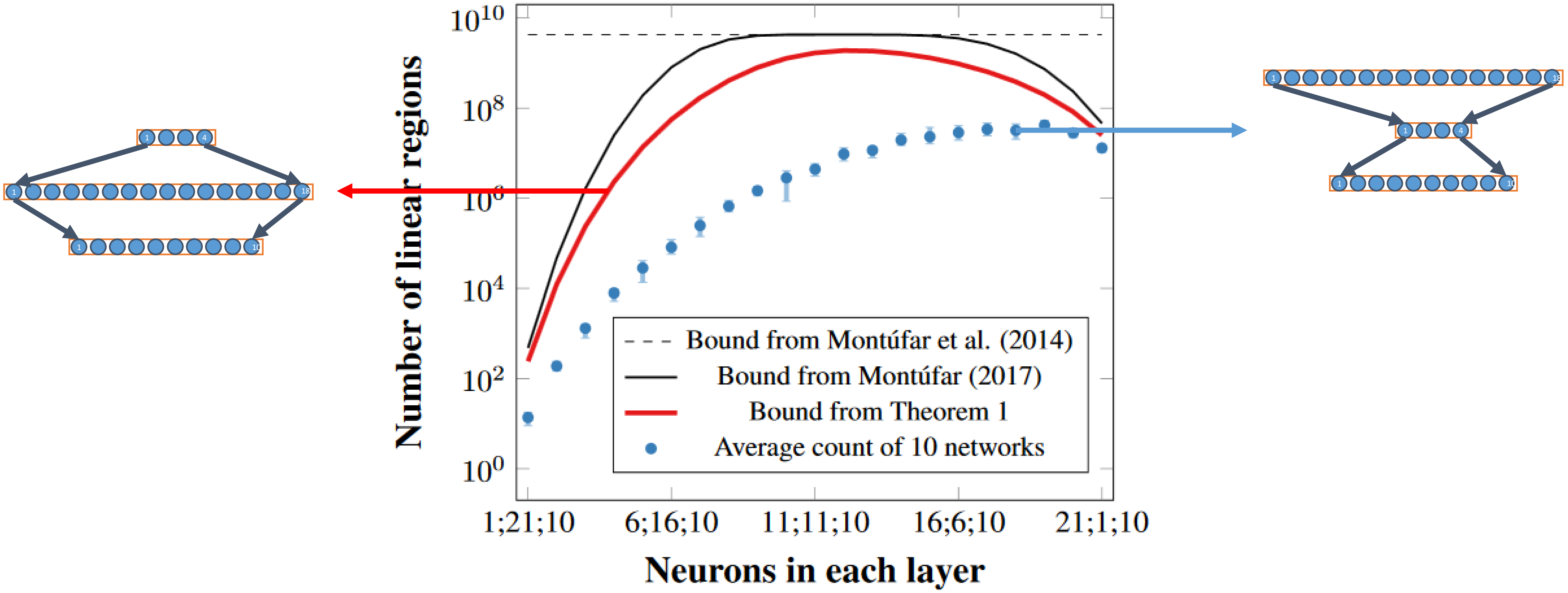




Bounding vs. Counting Results

S., Tjandraatmadja, Ramalingam 2018a

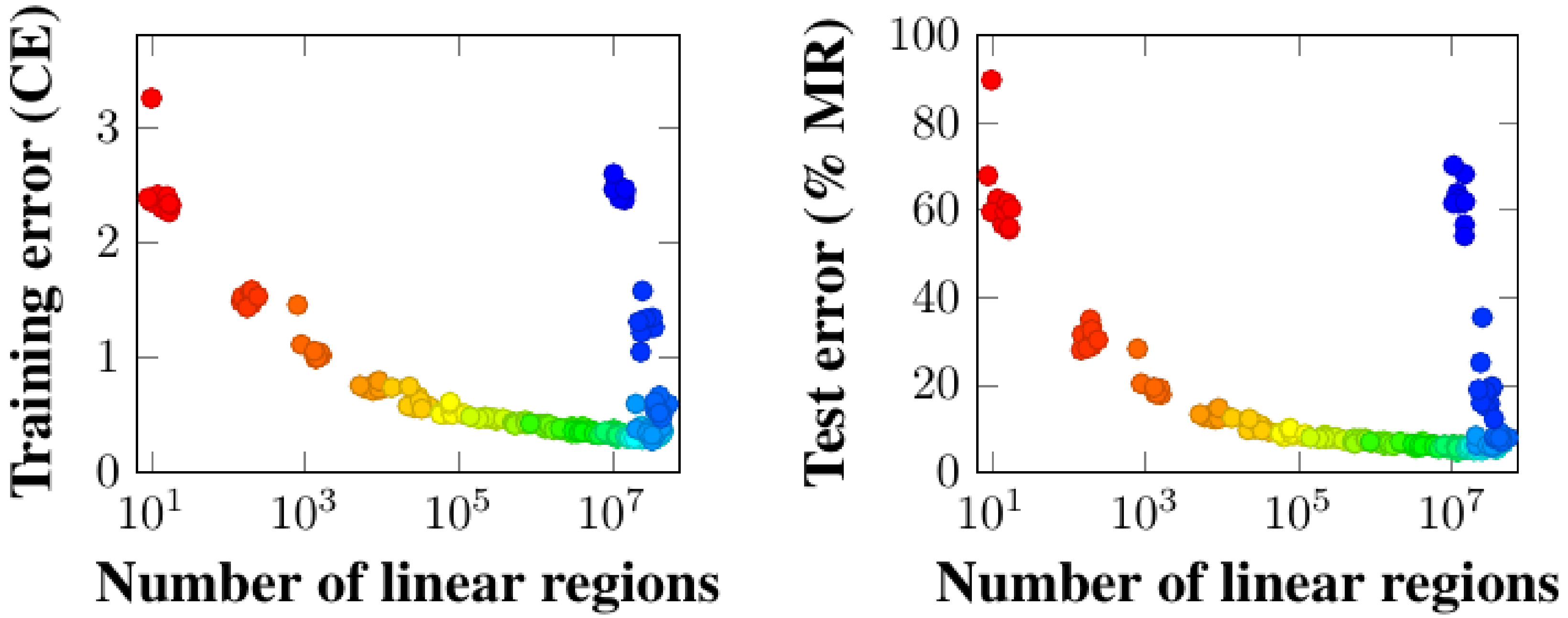
Comparison of bounds with average of 10 networks and min-max bars



Linear Regions and Accuracy

S., Tjandraatmadja, Ramalingam 2018a

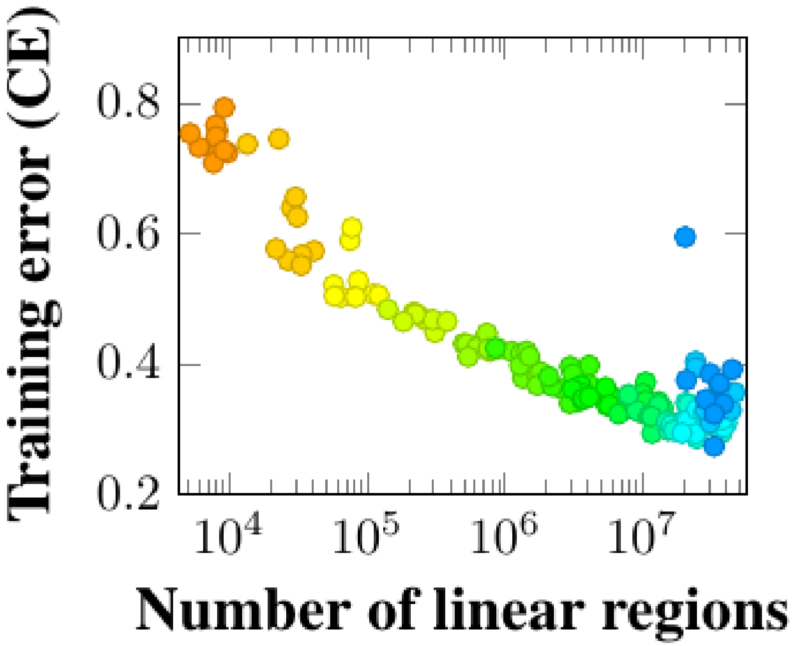
Plot with all points in heat scale by width, from 1,21,10 to 21,1,10

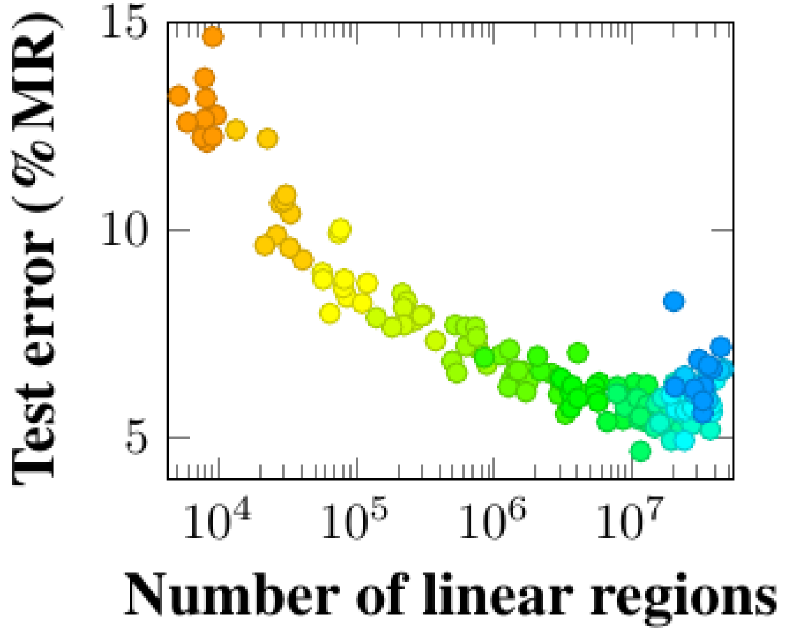


Linear Regions and Accuracy

S., Tjandraatmadja, Ramalingam 2018a

Same plot, but configurations are limited from 4,18,10 to 18,4,10





Towards Faster Methods to Measure Expressiveness

- **SAT-inspired probabilistic lower bounds**

Sampling with XOR Constraints

XOR constraints on Boolean variables, and parity constraints on 0—1 variables, have good sampling properties to splitting arbitrary solution sets

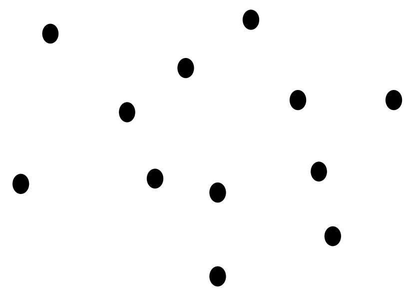
$$XOR(x_1, x_2, x_3) \leftrightarrow (x_1 + x_2 + x_3)\ MOD\ 2 = 1$$

Sampling with XOR Constraints

XOR constraints on Boolean variables, and parity constraints on 0—1 variables, have good sampling properties to splitting arbitrary solution sets

$$XOR(x_1, x_2, x_3) \;\leftrightarrow\; (x_1 + x_2 + x_3)\; MOD\; 2 = 1$$

- After adding $\boldsymbol{r}$ of such constraints multiple times, we may compute the probability of a lower bound of $\mathbf{2}^{r}$ if the resulting set is more often feasible

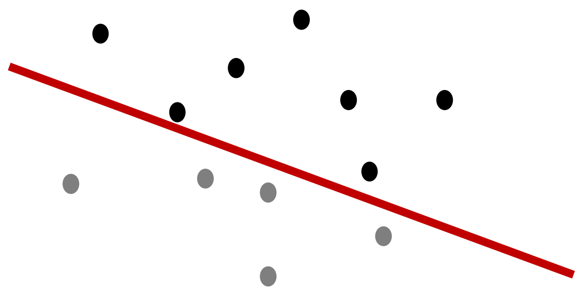


Sampling with XOR Constraints

XOR constraints on Boolean variables, and parity constraints on 0—1 variables, have good sampling properties to splitting arbitrary solution sets

$$XOR(x_1, x_2, x_3) \;\leftrightarrow\; (x_1 + x_2 + x_3)\, MOD\ 2 = 1$$

- After adding ***r*** of such constraints multiple times, we may compute the probability of a lower bound of $\mathbf{2^r}$ if the resulting set is more often feasible

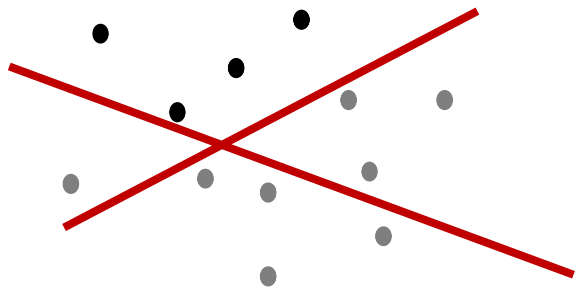


Sampling with XOR Constraints

XOR constraints on Boolean variables, and parity constraints on 0—1 variables, have good sampling properties to splitting arbitrary solution sets

$$XOR(x_1, x_2, x_3) \;\leftrightarrow\; (x_1 + x_2 + x_3)\, MOD\ 2 = 1$$

- After adding ***r*** of such constraints multiple times, we may compute the probability of a lower bound of $\mathbf{2^r}$ if the resulting set is more often feasible

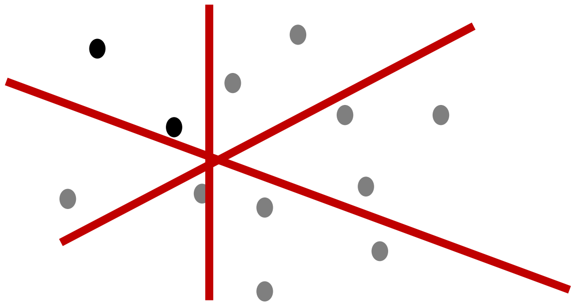


Sampling with XOR Constraints

XOR constraints on Boolean variables, and parity constraints on 0—1 variables, have good sampling properties to splitting arbitrary solution sets

$$XOR(x_1, x_2, x_3) \leftrightarrow (x_1 + x_2 + x_3)\ MOD\ 2 = 1$$

- After adding ***r*** of such constraints multiple times, we may compute the probability of a lower bound of $\mathbf{2}^{r}$ if the resulting set is more often feasible

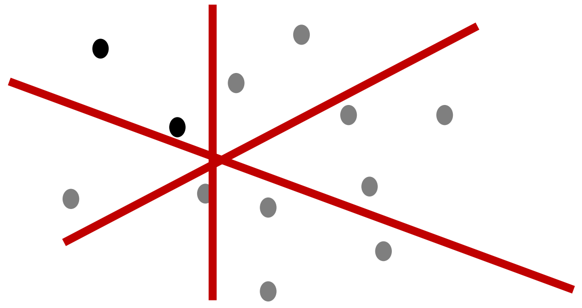


Sampling with XOR Constraints

XOR constraints on Boolean variables, and parity constraints on 0—1 variables, have good sampling properties to splitting arbitrary solution sets

$$XOR(x_1, x_2, x_3) \;\leftrightarrow\; (x_1 + x_2 + x_3)\; MOD\; 2 = 1$$

- After adding ***r*** of such constraints multiple times, we may compute the probability of a lower bound of $\mathbf{2^r}$ if the resulting set is more often feasible

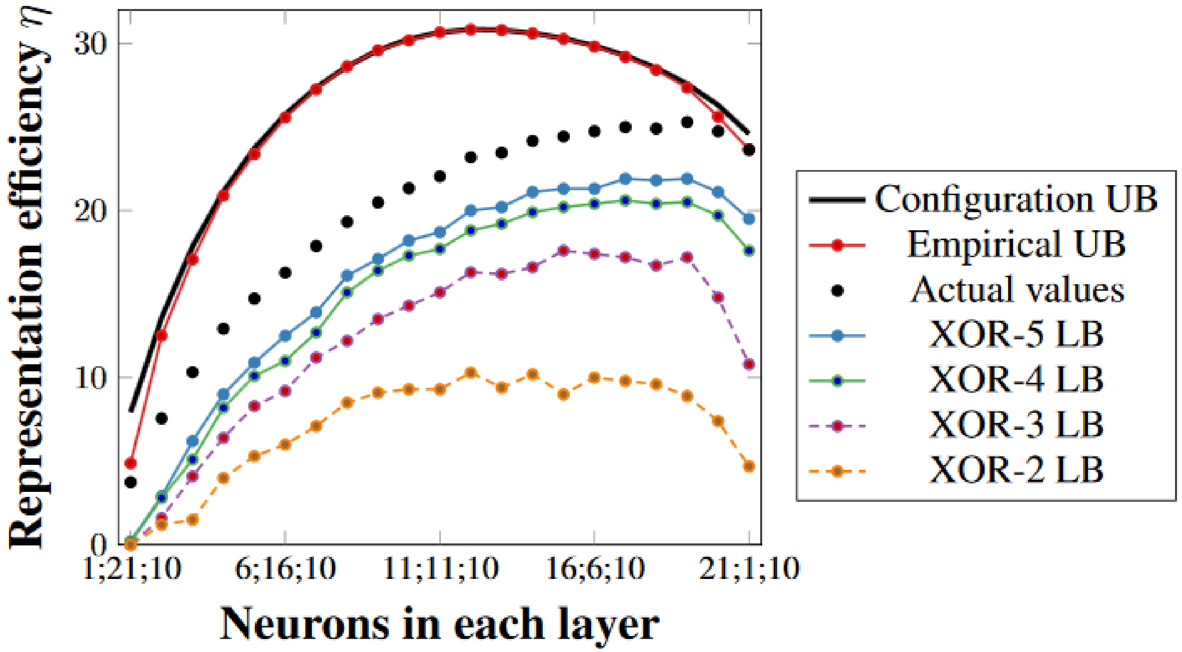


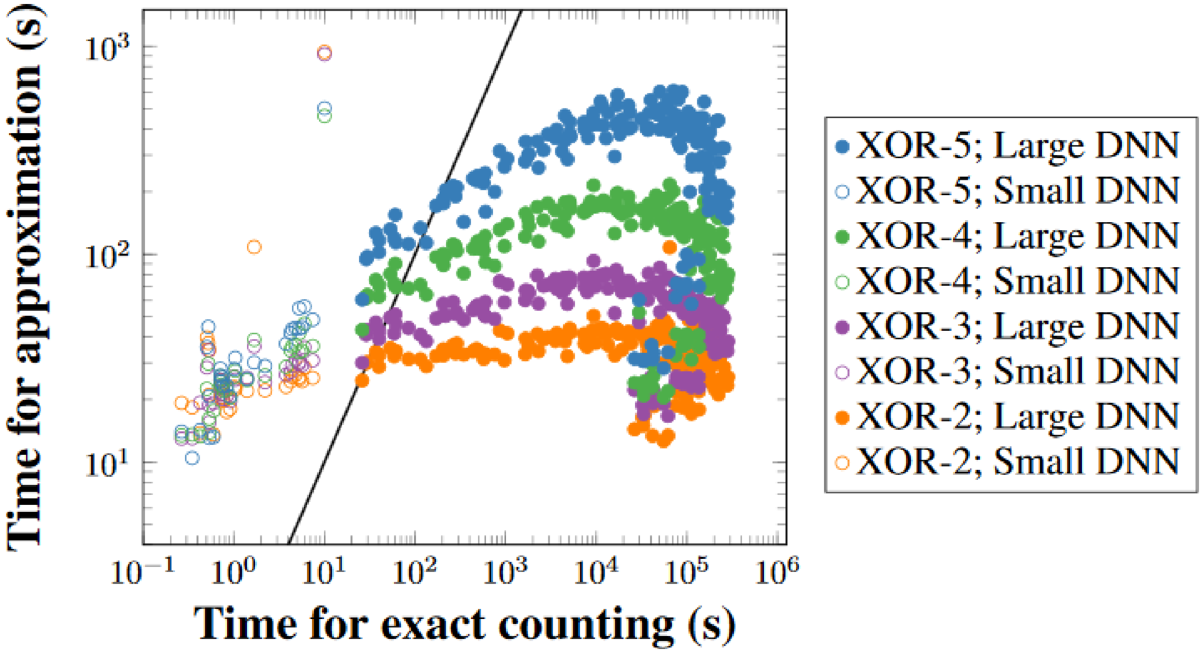
- Upper bounds require sufficiently large XORs, but we do not need them

Empirical Bounding Results

S., Ramalingam 2018b

Comparison of bound with coefficients and approximate counting





Summary

Conclusion

Bounds on linear regions

- We discovered tighter bounds that are maximized at particular depths
- The ReLU bound is precise for input of size 1
- Shallow networks can define more linear regions

Conclusion

Bounds on linear regions

- We discovered tighter bounds that are maximized at particular depths
- The ReLU bound is precise for input of size 1
- Shallow networks can define more linear regions

Counting linear regions

- We proposed an MILP-based method
- We developed SAT-inspired probabilistic lower bounds

Conclusion

Bounds on linear regions

- We discovered tighter bounds that are maximized at particular depths
- The ReLU bound is precise for input of size 1
- Shallow networks can define more linear regions

Counting linear regions

- We proposed an MILP-based method
- We developed SAT-inspired probabilistic lower bounds

What does the number of linear regions tells us?

- We can compare similar configurations through the number of regions
- The shape may also be important

Future Work

Practical uses for the characterization by linear regions:

- Compress neural nets without loss

Future Work

Practical uses for the characterization by linear regions:

- Compress neural nets without loss

Two-way exchange with integer programming:

- Counting and approximating solution sets
- Application to postoptimality analysis

Future Work

Practical uses for the characterization by linear regions:

- Compress neural nets without loss

Two-way exchange with integer programming:

- Counting and approximating solution sets
- Application to postoptimality analysis

New research directions:

- Understand other types of architectures
- Connect geometry with data

Thank you!

S., Tjandraatmadja, Ramalingam 2018a; **ICML 2018** (arXiv: 1711.02114)

S., Ramalingam 2018b; Submitted (arXiv: 1810.03370)

ThiagoSerra.com