SFU SIMON FRASER UNIVERSITY
ENGAGING THE WORLD

Understanding Deep Neural Networks from the Perspective of Piecewise Linear Property

A Thesis Seminar for the Degree of Doctor of Philosophy

Xia Hu

School of Computing Science

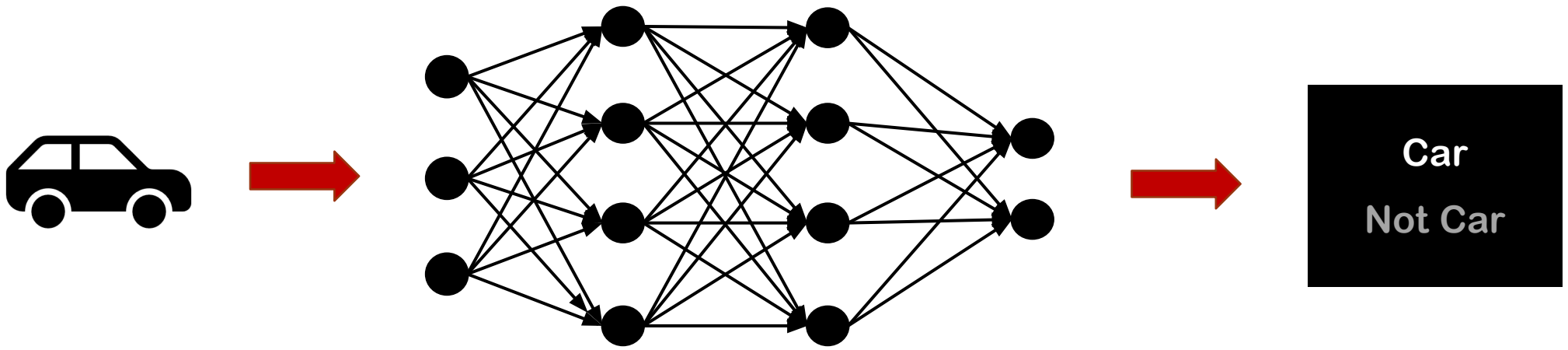
Simon Fraser University

May 6, 2021

Outline

- Understanding Deep Neural Networks
- Understanding by Piecewise Linear Property
- Our Focus
- Conclusion and Future Works

Deep Neural Networks



Input

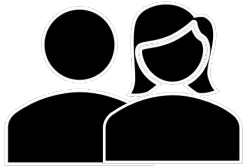
Feature extraction + classification

Output

Wide Application of Deep Neural Networks

Health care

Financial prediction



Facial recognition

Self-driving

Customer service

Deep Neural Networks Are Black-Boxes

- Deep neural networks (DNNs) are always considered **black-boxes**

Image source: XKCD

- ❑ Why deep models generalize well?
- ❑ Why small adversarial perturbations can easily fake a deep model?
- ❑ Why the model comes out prediction result $f(x)$ given an input x?

Why Understanding DNNs is Critical?

- Open the black-box and understand how deep neural networks work
  - Theoretical understanding
    - Theoretical foundation of deep learning
  - Requirements from practical application
    - Interpretations of predictions
    - Human-understandable

Perspectives of Understanding

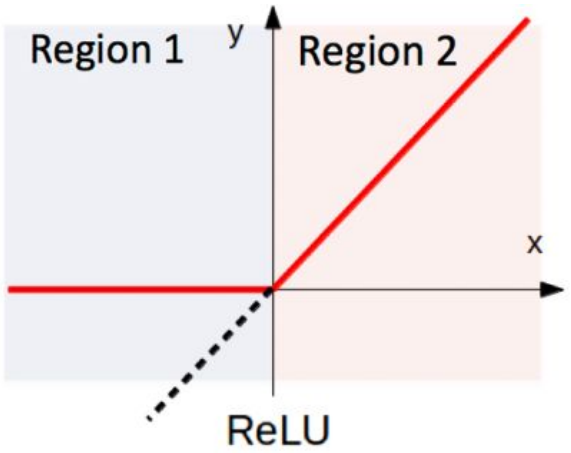
- Interpretation & Visualization
  - Why the model comes out prediction result $f(x)$ given an input x?
- Complexity
  - How complex the function represented by a deep model is?
- Robustness
  - Why small adversarial perturbations can easily fake a deep model?

Outline

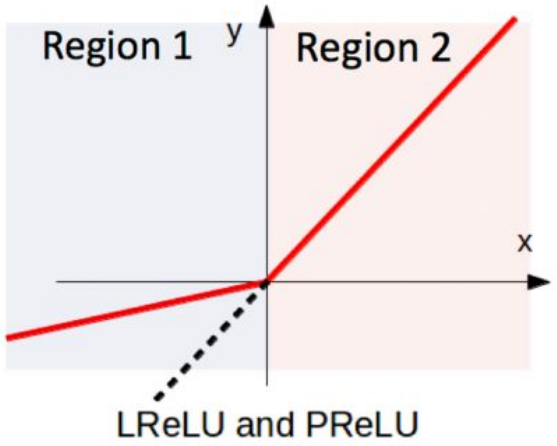
- Understanding Deep Neural Networks
- Understanding by Piecewise Linear Property
  - Piecewise Linear Property
  - Understanding by Piecewise Linear Property
- Our Focus
- Conclusion and Future Works

Piecewise Linear Activation Function

$$y = \max\{0, x\}$$



$$y = \max\{\alpha x, x\}$$



$$y = \max\{f_1(x), f_2(x), f_3(x), \dots\}$$

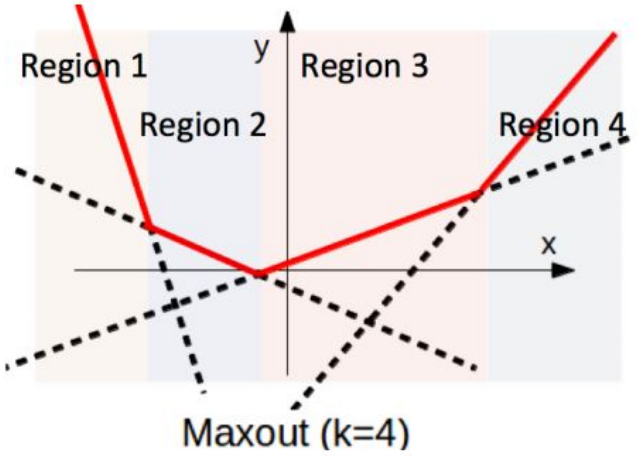
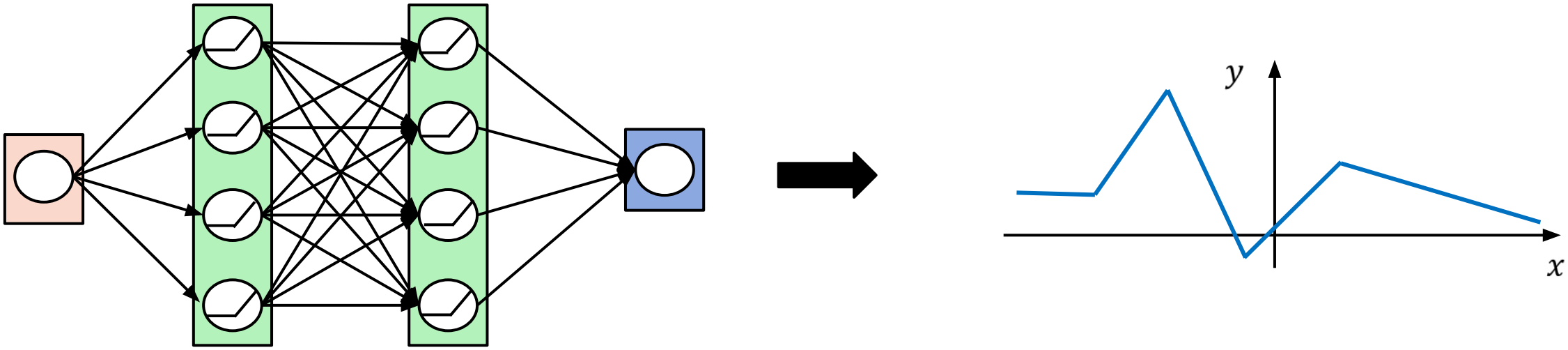


Image source: "On the Importance of Normalisation Layers in Deep Learning with Piecewise Linear Activation Units" Liao and Carneiro. 2016

Piecewise Linear Neural Network

- The function represented by a neural network with piecewise linear activation functions is piecewise linear



Piecewise Linear Property

- A deep neural network with piecewise linear activation functions divides the input space into a finite number of linear regions
  - "A ***linear region*** of a piecewise linear function $f: R^d \rightarrow R^c$ is a maximal connected subset of the input space R^d on which f is linear." [Montufar et al. 2014]
  - Linear regions are **convex polytopes**
  - Each linear region represents a **local linear function**.

Piecewise Linear Property

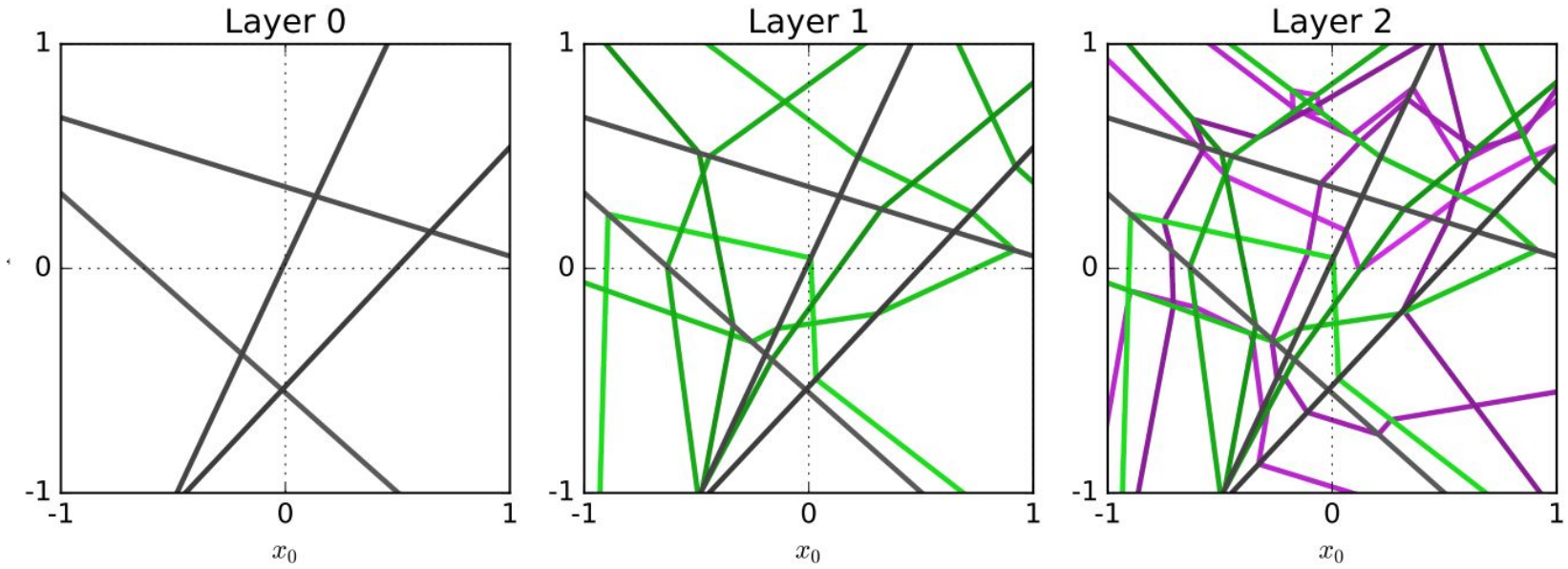


Image source: “On the Expressive Power of Deep Neural Networks” Raghu et al. 2016

Outline

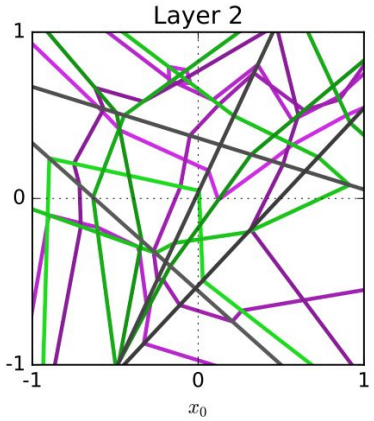
- Understanding Deep Neural Networks
- Understanding by Piecewise Linear Property
  - Piecewise Linear Property
  - Understanding by Piecewise Linear Property
- Our Focus
- Conclusion and Future Works

Piecewise linear property can be used to investigate the understanding of deep neural networks.

Piecewise linear property can be used to investigate the understanding of deep neural networks.

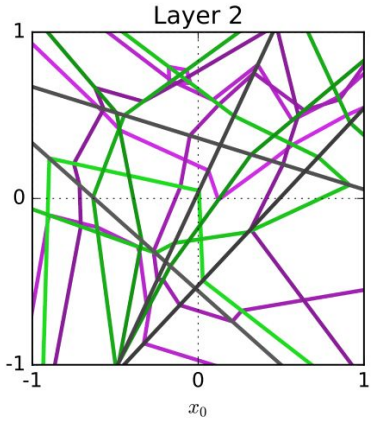
- The application of piecewise linear activation functions (especially ReLU) becomes more and more common.
- Linear regions are with a finite number ($\leq 2^m$)
- The deep neural network corresponds to a simple linear function within each linear region.

Examples



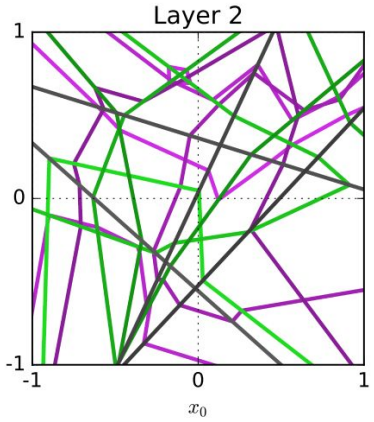
- **Interpretation**:
  - Space division
  - Regional linearity: $f(x) = wx + b, w \in R^d, b \in R$

Examples



- **Interpretation**:
  - Space division
  - Regional linearity: $f(x) = wx + b, w \in R^d, b \in R$
- **Complexity**: the finite number of linear regions demonstrate the model nonlinearity/complexity [Raghu et.al, 2016; Novak et.al, 2018]

Examples



- **Interpretation**:
  - Space division
  - Regional linearity: $f(x) = wx + b, w \in R^d, b \in R$
- **Complexity**: the finite number of linear regions demonstrate the model nonlinearity/complexity [Raghu et.al, 2016; Novak et.al, 2018]
- **Robustness**: the volume of linear regions demonstrates model sensitivity/robustness [Croce et.al, 2019]

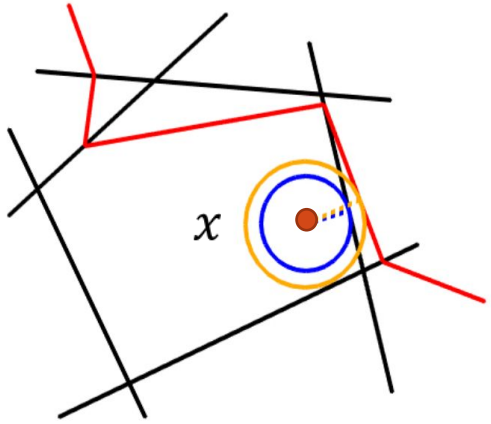


Image source: [1]"On the Expressive Power of Deep Neural Networks" Raghu et al. 2016; [2] "Provable Robustness of ReLU networks via Maximization of Linear Regions" Croce et al. 2019

Our Focus

We investigate two understanding problems using piecewise linear property

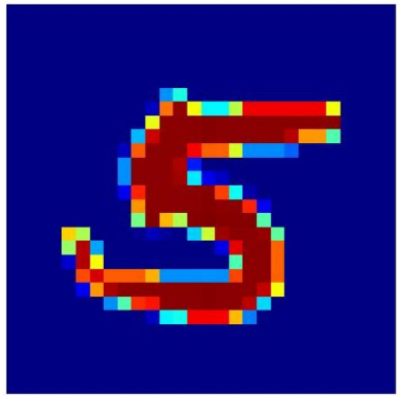
- Model interpretation
- Model complexity

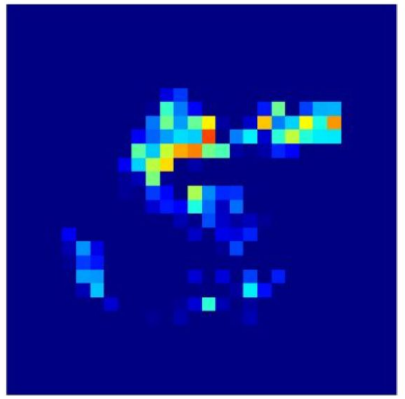
Outline

- Understanding Deep Neural Networks
- Understanding by Piecewise Linear Property
- Our Focus
  - Model Interpretation
  - Model Complexity
- Conclusion and Future Works

Model Interpretation

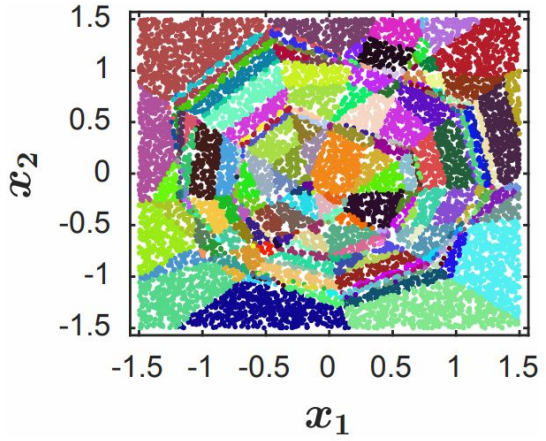
- Interpretation is to interpret the behavior of deep learning models in human-understandable way
  - Local interpretation [Simonyan et.al, 2013]
  - Mimic learning [Ba et.al, 2014]
  - Hidden neuron analysis [Yosinski et.al, 2015]

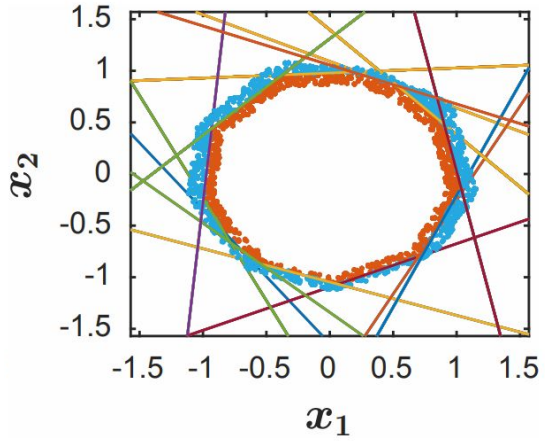




Openbox

- Problem: interpret deep neural network with piecewise linear activation functions
- Compute the exact linear function within each linear region
  - A specific linear region is represented by a configuration: $c = \{c_{11}, c_{12}, \dots, c_{1m_1}, c_{21}, \dots, c_{Lm_L}\}$
  - Given x, the exact linear function and the region boundary features of $c(x)$ are computable.
- We analyze the piecewise linear function represented by deep neural networks with piecewise linear activation functions.



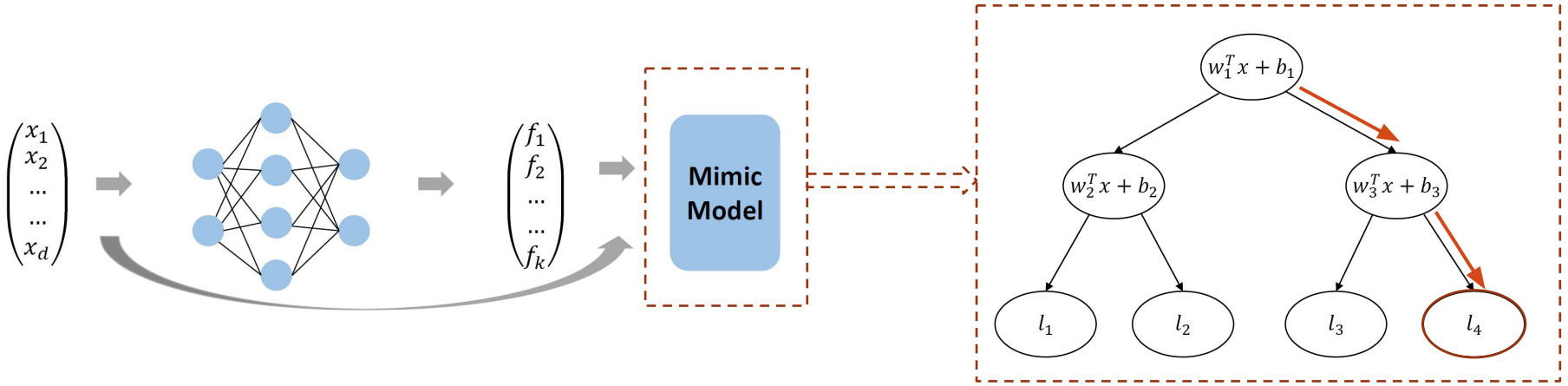


Oblique Model Tree

- Problem: provide interpretations to deep models regardless of model structure
- Mimic an oblique model tree with approximately functional equivalent from the deep model.
  - Functional equivalent: $\forall x, f(x) = g(x)$
  - Oblique Model Tree

$$S = \bigcup_{t \in T} S_t\,,\ S_t \cap S_{t'} = \emptyset\ for\ t \neq t'$$

- The oblique model tree is a piecewise linear approximation of the deep model.



Our Interpretation Study

- OPENBOX
  - Closed-form interpretation
  - Piecewise linear neural network
  - More efficient
  - Model-view, a large number of linear regions

Piecewise linear neural network, local interpretation.

- OMT
  - Approximation, functional equivalent
  - Any “black-box” model
  - Time cost to build
  - Rule extraction

For models that are not very complex, it can perform rule extraction and local interpretation well. (The tree is not too deep)

Outline

- Piecewise Linear Property
- Understanding Deep Neural Networks
- Our Focus
  - Model Interpretation
  - Model Complexity
- Conclusion and Future Works

Model complexity analyzes how complicated functions can be expressed by a given deep learning model

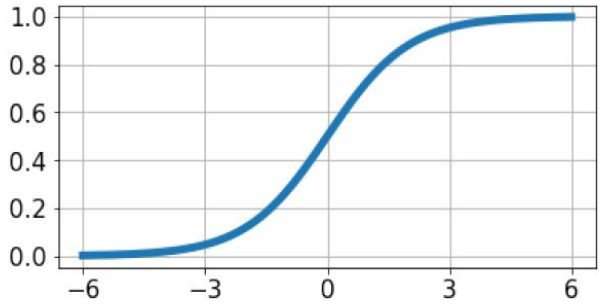
- Expressive capacity
- Effective complexity

Example

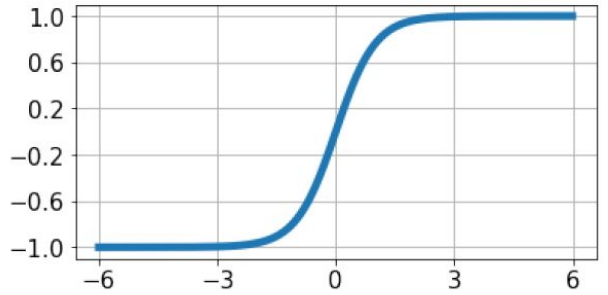
- Expressive capacity of this quadratic function is "quadratic"
  - $y = ax^2 + bx + c$
- With parameter values {0, 1, 0}, the effective complexity becomes linear
  - $y = 0x^2 + 1x + 0$

Effective Complexity Measure

- Piecewise linear neural networks
  - number of linear regions [Novak et.al., 2018]
  - trajectory length [Raghu et.al., 2017]
  - Volume of boundaries of linear regions [Hanin and Rolnick, 2019]
- What about non-piecewise linear?



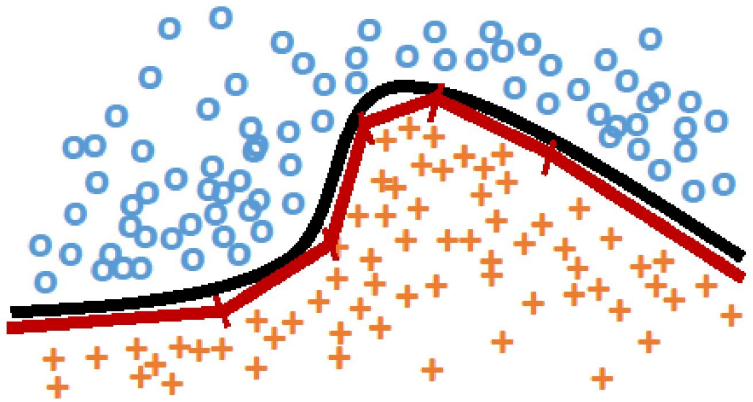
(a) Sigmoid



(b) Tanh

Model Complexity Of Curve DNNs

- Learn a piecewise linear approximation of DNNs with smooth curve activation functions
- Use the number of linear regions to represent model complexity

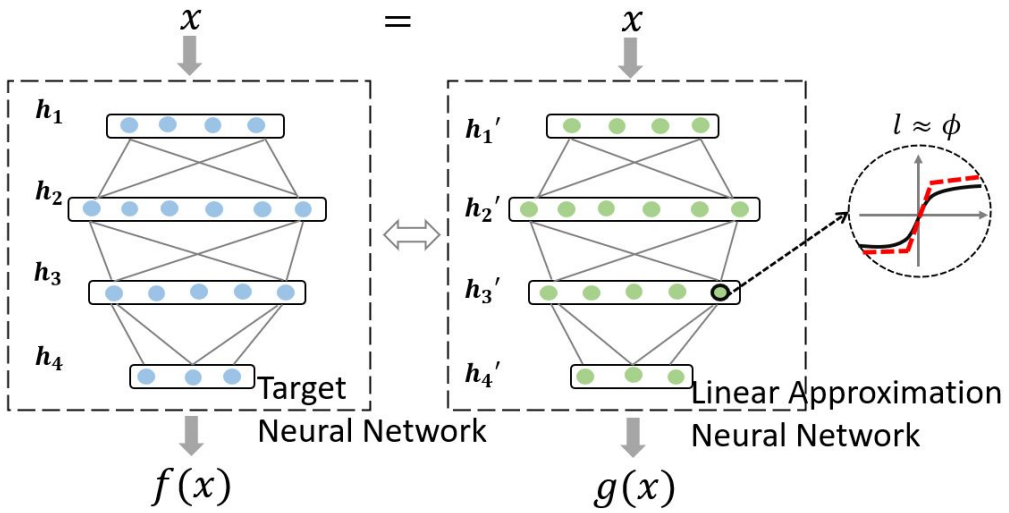


TWO CHALLENGES OF APPROXIMATION

- Guarantee approximation degree.
- Construction principle of approximations.

LANN

- We provide a piecewise linear approximation g to deep neural networks f with curve activation functions (e.g., Sigmoid, Tanh)
  - Individual piecewise linear approximation to each hidden neuron



- ✔ $h_i(z) = \phi(V_i z + b_i)$
- ✔ $f(x) = V_o h_L(h_{L-1}(\cdots(h_1(x)))) + b_o$
- ✔ $h_i'(z) = \ell_i(V_i z + b_i)$
- ✔ $g(x) = V_o h_L'(h_{L-1}'(\ldots(h_1'(x)))) + b_o$

Degree of Approximation

- Define degree of approximation

$$\mathcal{E}(g;f) = \mathbb{E}[\ \frac{1}{c}\sum |g(x) - f(x)|\ ] \quad (1)$$

- Approximation of every neuron brings error: $e_{ij} = |l_{ij} - \phi_{ij}|$.
- We analyze approximation error propagation through layers
  - Approximation error = accumulation of neuronal approximation error

$$\mathcal{E}(g;f) = \sum_{i,j} \underbrace{\frac{1}{c}\sum(\ |V_o|\prod_{q=L}^{i+1}\mathbb{E}[|J_q|]\ )_{*,j}}_{w_{i,j}^{(e)}}(\mathbb{E}[e_{i,j}] + \mathbb{E}[|\hat{\epsilon}_{i,j}|]) \quad (2)$$

- Build a LANN g with minimal number of linear regions s.t. $\mathcal{E}(g;f) \leq \lambda$

Complexity Measure by LANN

- Complexity of f can be represented by g when $\mathcal{E}(g;f)$ is small.
- The number of linear regions of the approximation g is upper bounded by

$$\prod_{i=1}^{L}(\sum_{j=1}^{m_i} k_{i,j} - m_i + 1)^d$$

- Model complexity

$$C(f)_\lambda = d\sum_{i=1}^{L}\log(\sum_{j=1}^{m_i} k_{i,j} - m_i + 1)$$

Discussion of LANN

- Provide an effective complexity measure to DNNs with curve activation.
- Bridge the gap
  - Interpretations and other understanding studies

Outline

- Understanding Deep Neural Networks
- Understanding by Piecewise Linear Property
- Our Focus
- Conclusion and Future Works

Summary

- Piecewise linear property
- Understanding deep neural networks from the perspective of piecewise linear property

Model Interpretation

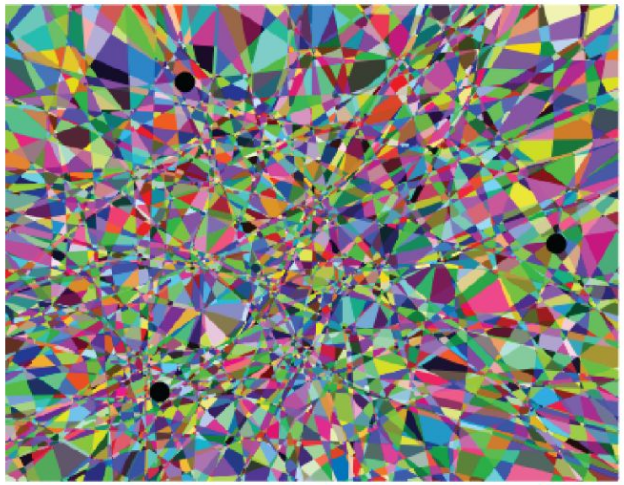
- OMT
  - Tree-based approximation
- OPENBOX
  - Analyze region division, linear regions

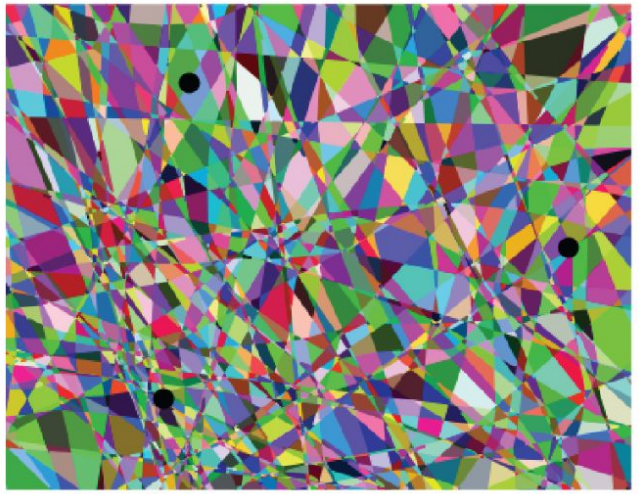
Model Complexity

- LANN
  - Piecewise linear approximation

Future Work

- Constrain model complexity by reducing the number of linear regions
  - Better interpretation
  - Prevent overfitting



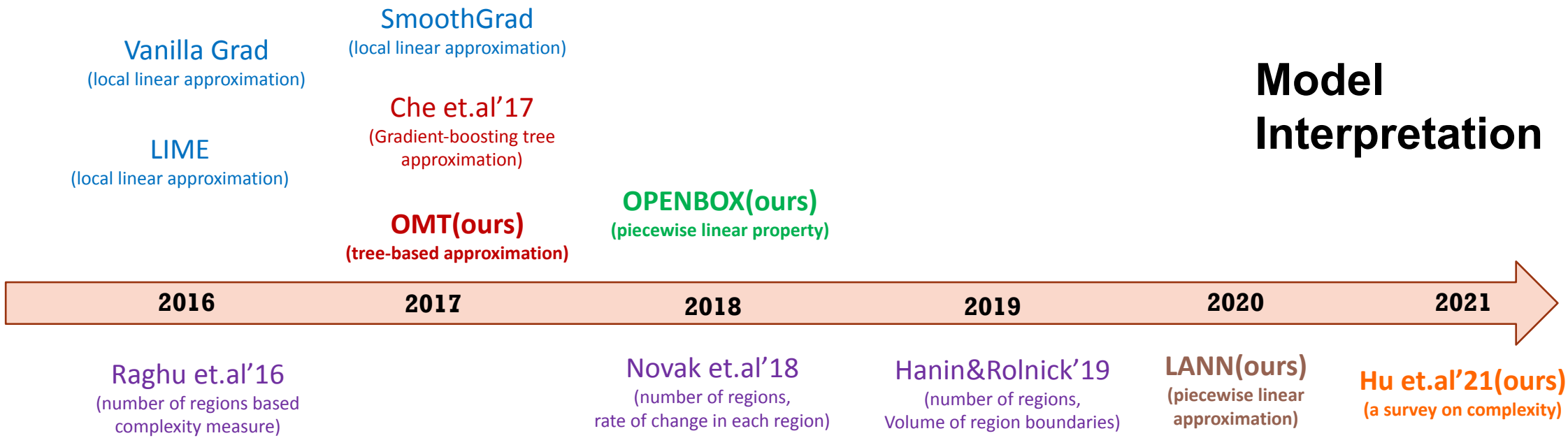


Future Work

- Measure effective complexity by estimating the amount of information in a deep neural network
  - Compression ratio
  - Model pruning

THANK YOU!

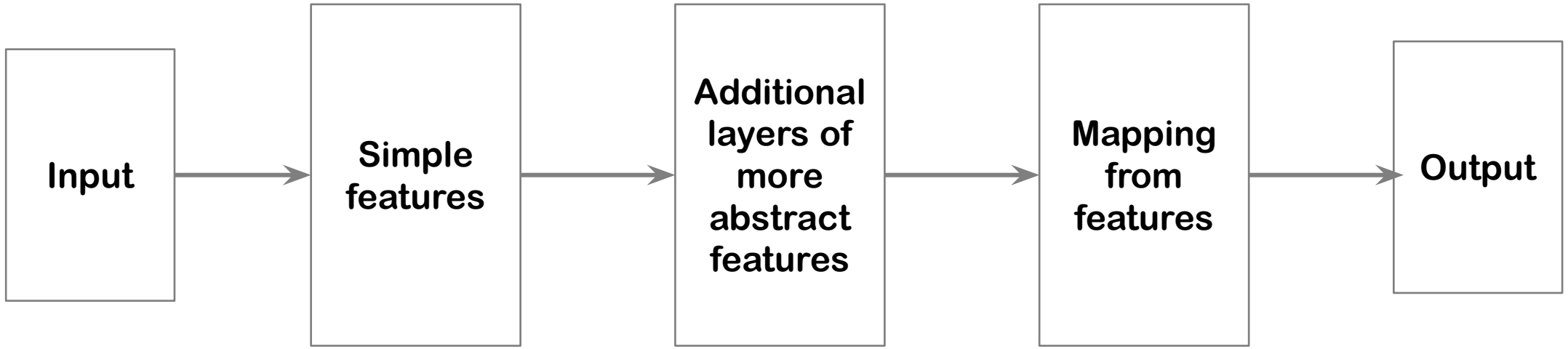
Timeline


Vanilla Grad
(local linear approximation)
LIME
(local linear approximation)
SmoothGrad
(local linear approximation)
Che et.al'17
(Gradient-boosting tree approximation)
OMT(ours)
(tree-based approximation)
OPENBOX(ours)
(piecewise linear property)
Model Interpretation
2016
2017
2018
2019
2020
2021
Raghu et.al'16
(number of regions based complexity measure)
Novak et.al'18
(number of regions, rate of change in each region)
Hanin&Rolnick'19
(number of regions, Volume of region boundaries)
LANN(ours)
(piecewise linear approximation)
Hu et.al'21(ours)
(a survey on complexity)
Model Complexity

Local Interpretation

- Local interpretation ⇒ prediction behavior
  - Given a model f, an input x
  - Interpretation $I\left(x, f(x)\right)$
    - $\{w_1, w_2, \ldots, w_d\}$
    - A mask [1, 0 , …, 1]

Deep Neural Networks



Oblique Model Tree

- Internal node p:

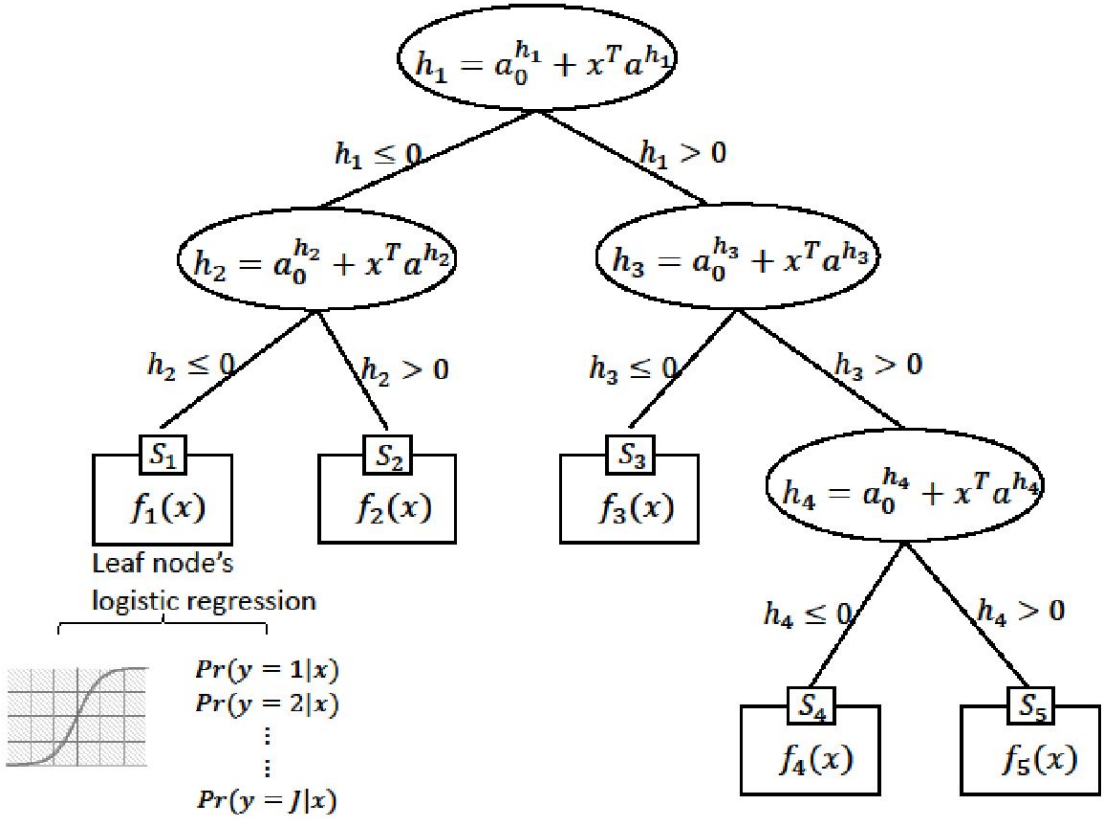
$$h_p(x) = a_o + \sum_{i=1}^{d} a_i x_i$$

- Leaf node t:

$$f_t(x) = \arg\max_i \{\Pr(y = i|x)\}$$

$$\Pr(y = i \mid x) = \frac{e^{l_i(x)}}{\sum_{k=1}^{c} e^{l_k(x)}}$$

$$l(x) = \begin{pmatrix} w^{(1)} \\ w^{(2)} \\ \vdots \\ w^{(c)} \end{pmatrix} \cdot \begin{pmatrix} 1 \\ x \end{pmatrix}$$



Oblique Model Tree

Algorithm 1: BuildingOMT

Require: The set of input instances X, a deep model $\mathcal{N}$
Ensure: A oblique model tree T

```
repeat
    Optimize logistic model on each child node using gradient descent
    Optimize oblique split using gradient descent
until ℒ converge
Build OMT on left and right children nodes
return T
```

Openbox

- Configuration $c = \{c_{11}, c_{12}, \dots, c_{1m_1}, c_{21}, \dots, c_{Lm_L}\}$
  - ReLU: $c_{ij} = 0 \; if \; output \; 0; c_{ij} = 1 \; if \; output \; x$
- Linear function of a linear region

$$softmax\left(\widehat{W}^{(1:L)}x + \hat{b}^{(1:L)}\right)$$

- Boundaries of a linear region

$$z^{(l)} = \widehat{W}^{(1:l)}x + \hat{b}^{(1:l)} \; for \; l = 1, \dots, L-1$$

Openbox

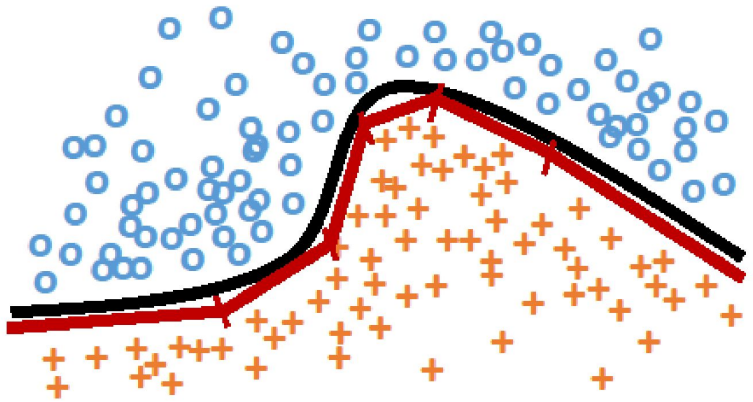
Algorithm 2: $OpenBox(\mathcal{N}, D_{train})$

Require: a fixed piecewise linear neural network $\mathcal{N}$, the training set D_{train}.
Ensure: the set of active local linear classifiers G .

1: **for** each $x \in D_{train}$ **do**
2: Compute the configuration $c(h) \leftarrow conf(x)$.
3: **if** $c(h) \notin \mathcal{C}$ **then**
4: $\mathcal{C} \leftarrow \mathcal{C} \cup c(h)$.
5: Compute the closed form of $f_h(x)$ and P_h.
6: $G \leftarrow G \cup (f_h(x), P_h)$.
7: **end if**
8: **end for**
9: **return** G

Model Complexity Of Curve DNNs

- Learn a piecewise linear approximation of DNNs with smooth curve activation functions
- Use the number of linear regions to represent model complexity

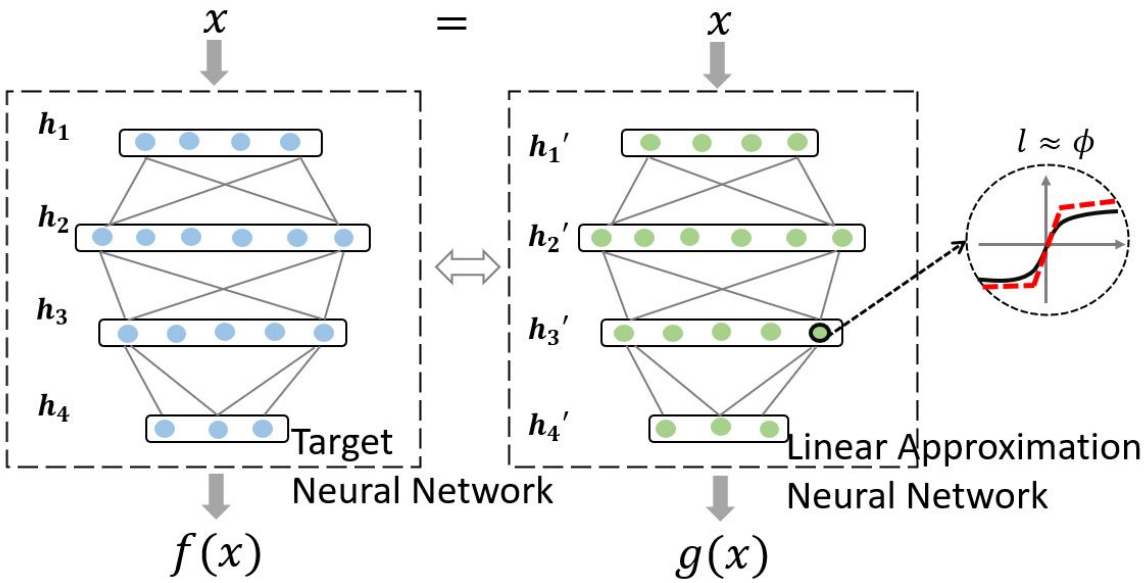


TWO CHALLENGES OF APPROXIMATION

- Guarantee approximation degree.
- Construction principle of approximations.

LANN

- LANN = Linear Approximation Neural Network
  - Piecewise linear approximation function $l_{ij} \approx \phi_{ij}$
  - Individual approximation for each neuron



✔ $h_i(z) = \phi(V_i z + b_i)$

✔ $f(x) = V_o h_L(h_{L-1}(\cdots(h_1(x)))) + b_o$

✔ $h_i'(z) = \ell_i(V_i z + b_i)$

✔ $g(x) = V_o h_L'(h_{L-1}'(\ldots(h_1'(x)))) + b_o$

Degree of approximation

- Define approximation error as

$$\mathcal{E}(g;f) = \mathbb{E}[\ \frac{1}{c}\sum |g(x) - f(x)|\ ] \quad (1)$$

- Approximation of every neuron brings error: $e_{ij} = |l_{ij} - \ \phi_{ij}|$.
- We analyze approximation error propagation through layers

$$\mathcal{E}(g;f) = \frac{1}{c}\sum(|V_o|\ \mathbb{E}[|r_L|])$$

$$\mathbb{E}[|r_i|] \leq \mathbb{E}[e_i] + \mathbb{E}[|J_i|]\mathbb{E}[|r_{i-1}|] + \mathbb{E}[|\hat{\epsilon_i}|] \quad (2)$$

- Approximation error = accumulation of neuronal approximation error

$$\mathcal{E}(g;f) = \sum_{i,j} \underbrace{\frac{1}{c}\sum(\ |V_o|\prod_{q=L}^{i+1}\mathbb{E}[|J_q|]\ )_{*,j}}_{w_{i,j}^{(e)}}(\mathbb{E}[e_{i,j}] + \mathbb{E}[|\hat{\epsilon}_{i,j}|]) \quad (3)$$

How to build an LANN?

- Build a LANN to λ approximation degree:

$$\mathcal{E}(g; f) \leq \lambda$$

- Minimize the number of linear regions of the approximated g
- A l_{ij} with k linear subfunctions contributes $k - 1$ hyperplanes to input space splitting.
- Minimize the number of hyperplanes contributing to input space splitting.

How to build an LANN?

- Main idea of the algorithm:
  1) Initialize g to set every l_{ij} to be a linear function
  2) Find neuron $\{i^*, j^*\}$ s.t.

$$\{i^*, j^*\} = \arg\max(\mathbb{E}[e_{i,j}] - \mathbb{E}[e_{i,j}]_{+p^*_{i,j}})$$

  - ✓ p^*_{ij} corresponds to subfunction whose insertion decreases $\mathbb{E}[e_{i,j}]$ most;
  - ✓ The Subscript $+p^*_{ij}$ demonstrates recomputing $\mathbb{E}[e_{i,j}]$ after adding p^*_{ij} to l_{ij}.

  3) Add the new subfunction represented by p^*_{ij} to $l_{\{i^*,j^*\}}$;
  4) Repeat 2), 3) until $\mathcal{E}(g; f) \leq \lambda$.

Recall the two challenges

- Guarantee approximation degree.
  - Analyze approximation error propagation;
  - Build LANNs under required approximation degree.
- Construction principle of approximations.
  - Minimize the number of linear regions of the approximation function.

Complexity measure by lanns

- The (minimal) number of linear regions can be a representation of model complexity.
- The number of linear regions of a LANN is bounded by

$$\prod_{i=1}^{L}\left(\sum_{j=1}^{m_i} k_{i,j} - m_i + 1\right)^d \quad (1)$$

- Model complexity measure by

$$C(f)_\lambda = d\sum_{i=1}^{L}\log\left(\sum_{j=1}^{m_i} k_{i,j} - m_i + 1\right) \quad (2)$$

- ✓ λ: the approximation degree
- ✓ d: input dimension
- ✓ m_i: width of layer i
- ✓ $k_{i,j}$: number of subfunctions of $l_{i,j}$

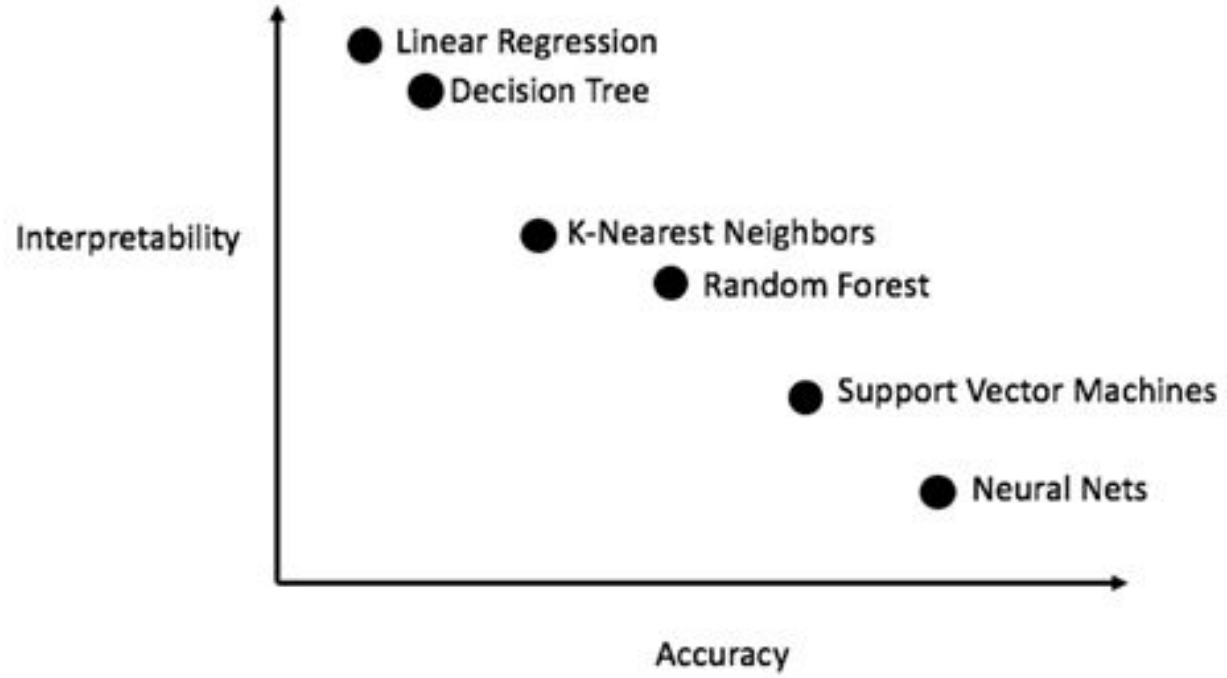


Image source: towardsdatascience.com