DEEP NETWORKS LEARN FEATURES FROM LOCAL DISCONTINUITIES IN THE LABEL FUNCTION

Presenter:
Harish G Ramaswamy
IIT Madras

Authors: Prithaj Banerjee, Harish G. Ramaswamy,
Yadav Mahesh Lorik, Chandrashekar Lakshminarayanan

Setup

- Setting : Supervised classification
- Data : (x,y) Tuples
- Model : Function from X to Y
- Learning Algorithm : Converts training data to model
- Feature: Typically a function from X to Reals

Outline

1. Feature Learning

2. Oblique Decision Trees (ODT)

3. Deep Linearly Gated Networks (DLGN)

4. Decision Tree Construction via DLGN

5. New Narratives

Feature Learning Hypothesis

Model features *change during training to better "align" with* ***data features***

- Linear/Kernel models with fixed features do NOT have feature learning abilities.

- Feature learning enables better generalisation with true structure discovery

Demonstrating Feature Learning in DNNs

- Requirements on data:
  - Admits a natural notion of “features”.
  - DNNs outperform Kernel/Linear methods.

- Requirements on architecture/model:
  - Admits a natural notion of “features”.
  - Amenable to be decomposed into simple components.
  - Perform comparably to classic deep net architectures on benchmarks.
  - Outperform linear/kernel methods on benchmarks

Demonstrating Feature Learning in DNNs

- Requirements on data:
  - Admits a natural notion of “features”.
  - DNNs outperform Kernel/Linear methods.

- Requirements on architecture/model:
  - Admits a natural notion of “features”.
    - Amenable to be decomposed into simple components.
    - Perform comparably to classic deep net architectures on benchmarks.
    - Outperform linear/kernel methods on benchmarks

Demonstrating Feature Learning in DNNs

- Additional Requirements on data and architecture:
  - Data and model features must be comparable.
  - Value and role of depth must be clear.
  - Provide intuition on how loss gradients w.r.t. parameters enable feature discovery.

Outline

1. Feature Learning

2. Oblique Decision Trees (ODT)

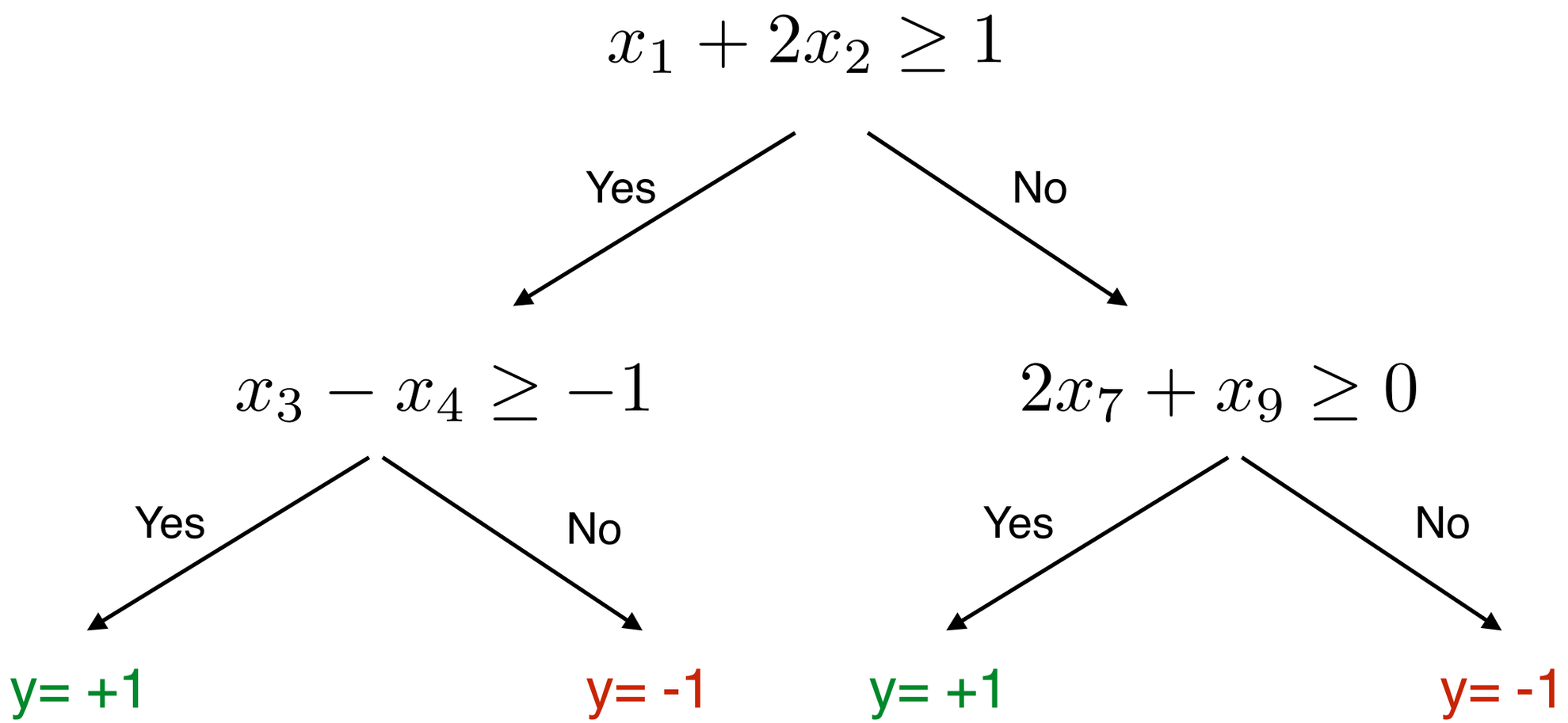
3. Deep Linearly Gated Networks (DLGN)

4. Decision Tree Construction via DLGN

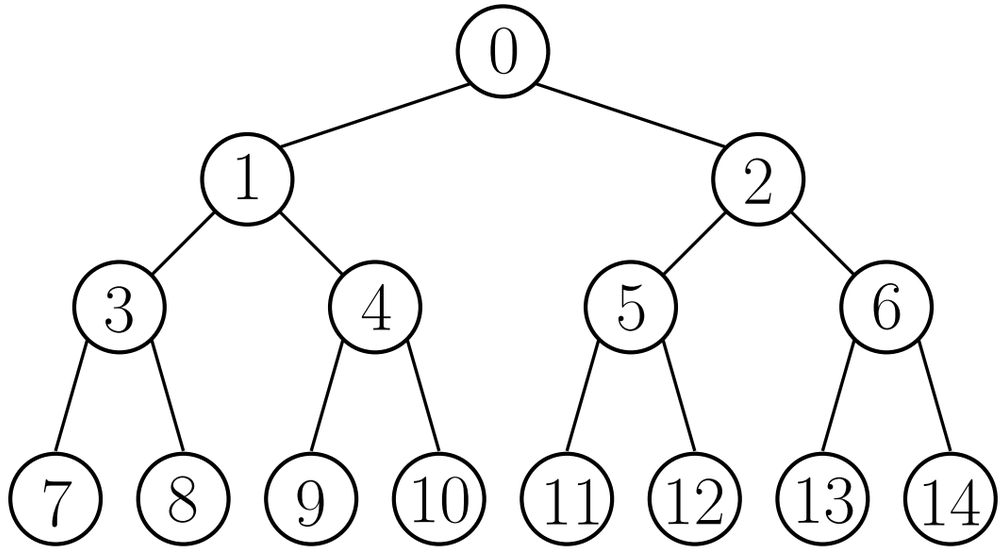
5. New Narratives

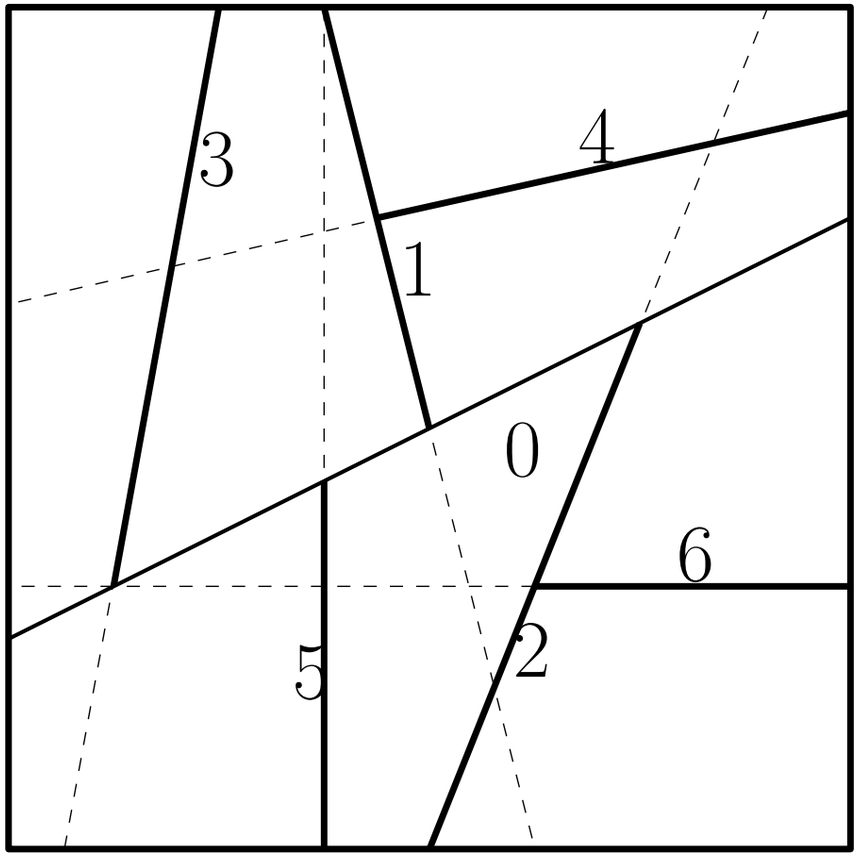
Oblique Decision Tree Labelling Functions

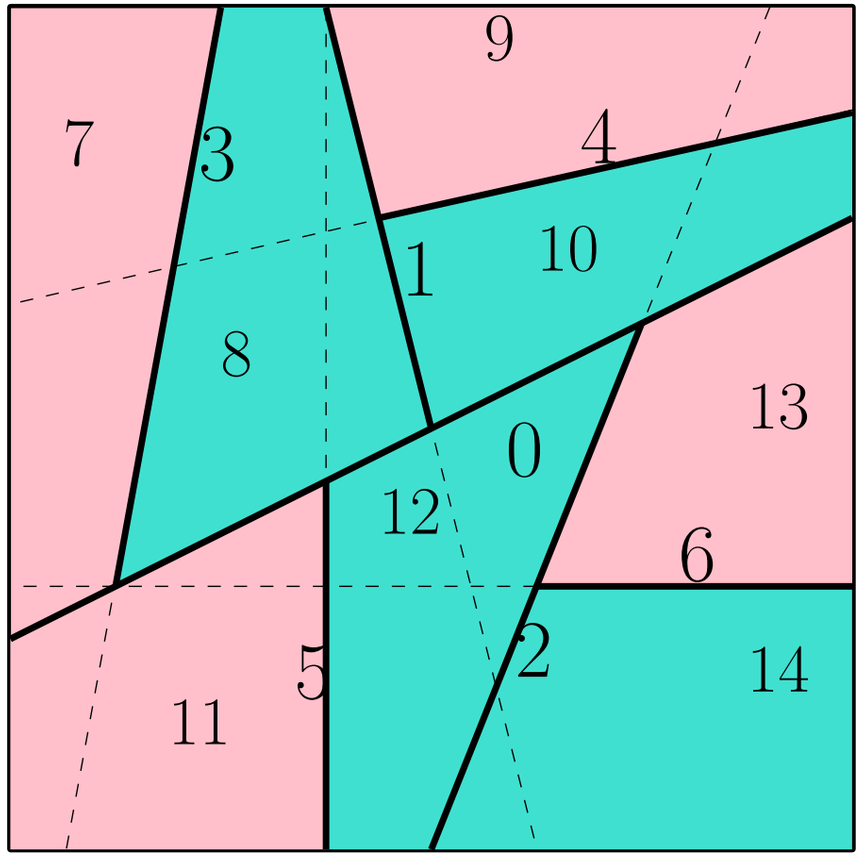
Let the true labelling function be an ODT e.g.



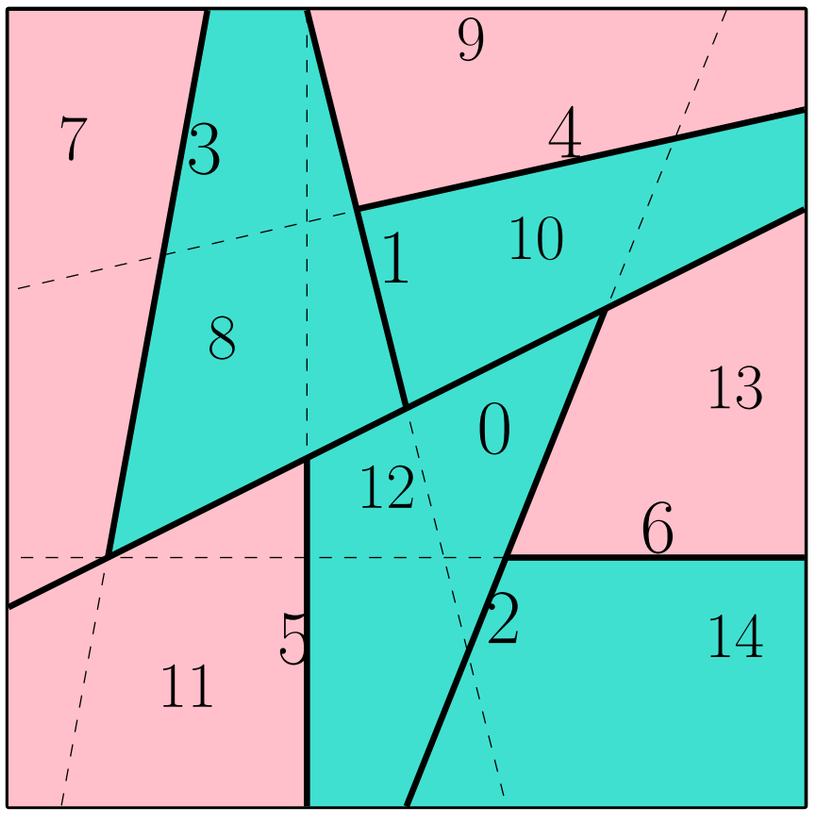
Features in an ODT Labelled Dataset

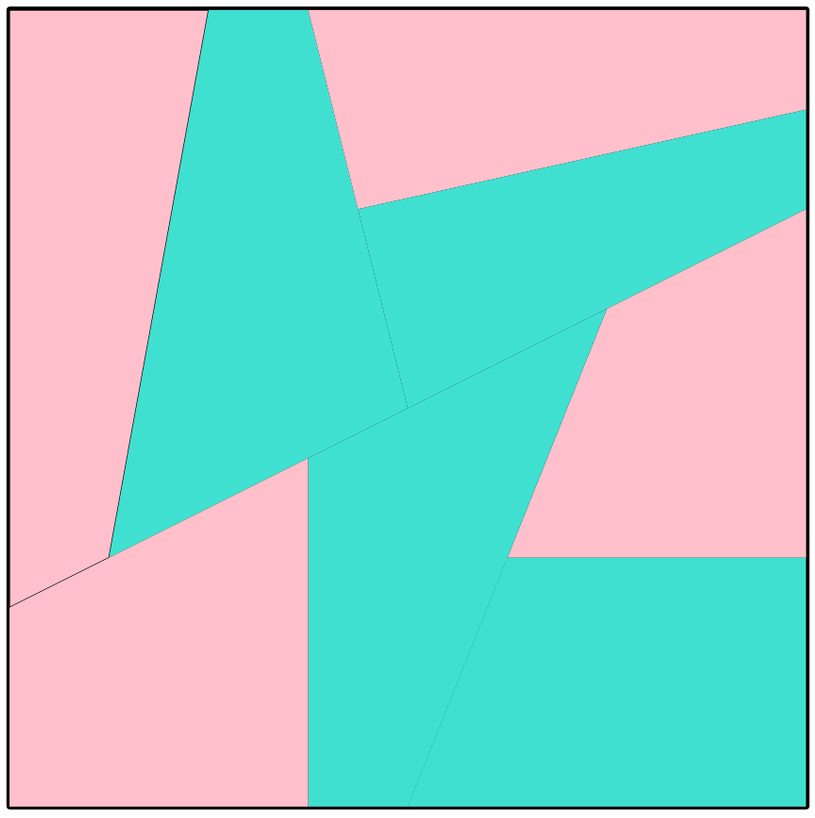




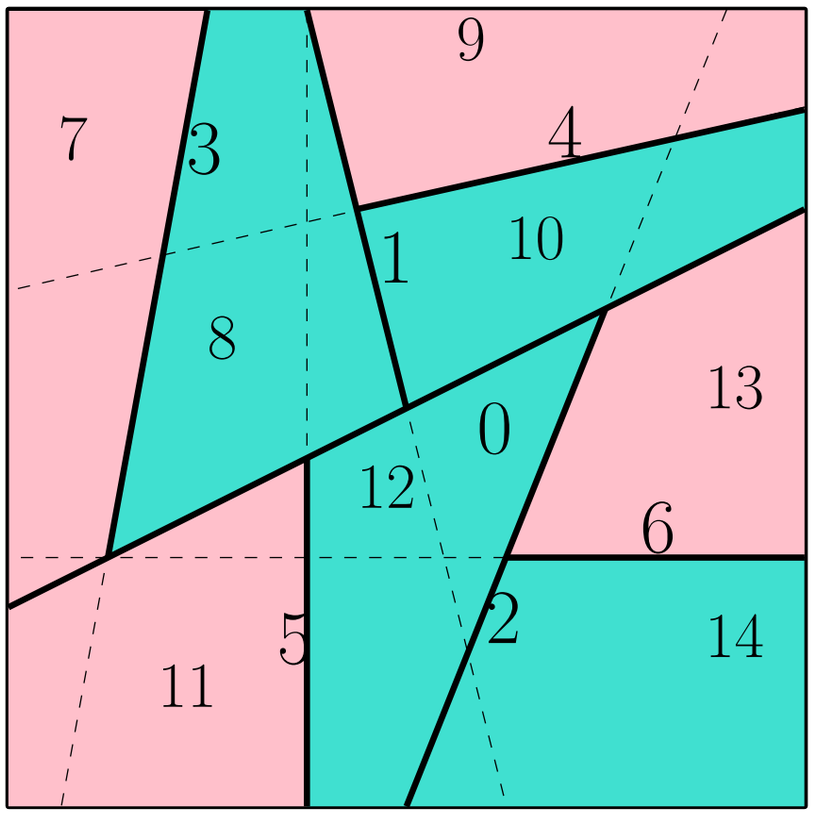


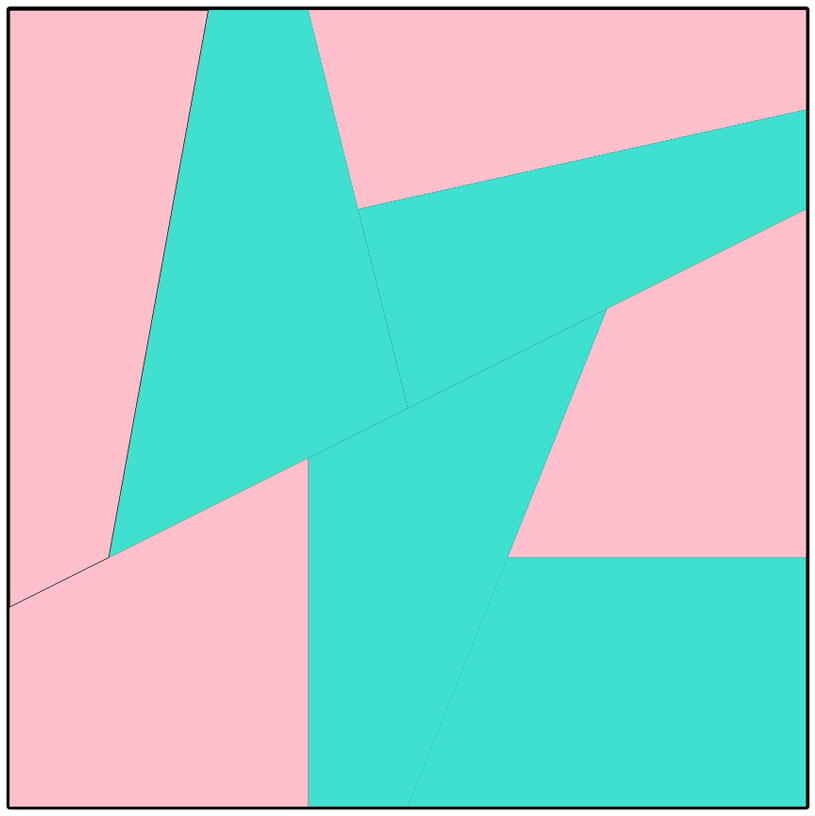
Features in an ODT Labelled Data





Features in an ODT Labelled Data





The data features correspond to the hyperplanes/halfspaces of the ODT internal nodes.

Outline

1. Feature Learning

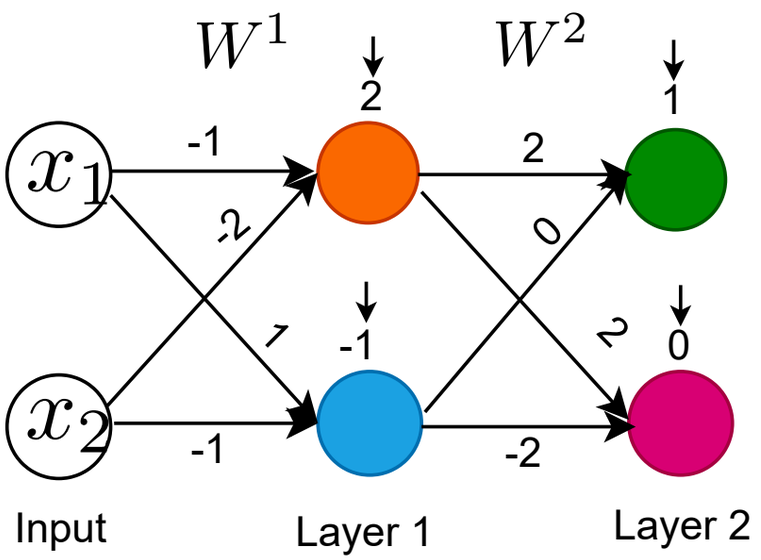
2. Oblique Decision Trees (ODT)

3. Deep Linearly Gated Networks (DLGN)

4. Decision Tree Construction via DLGN

5. New Narratives

Deep Linearly Gated Networks

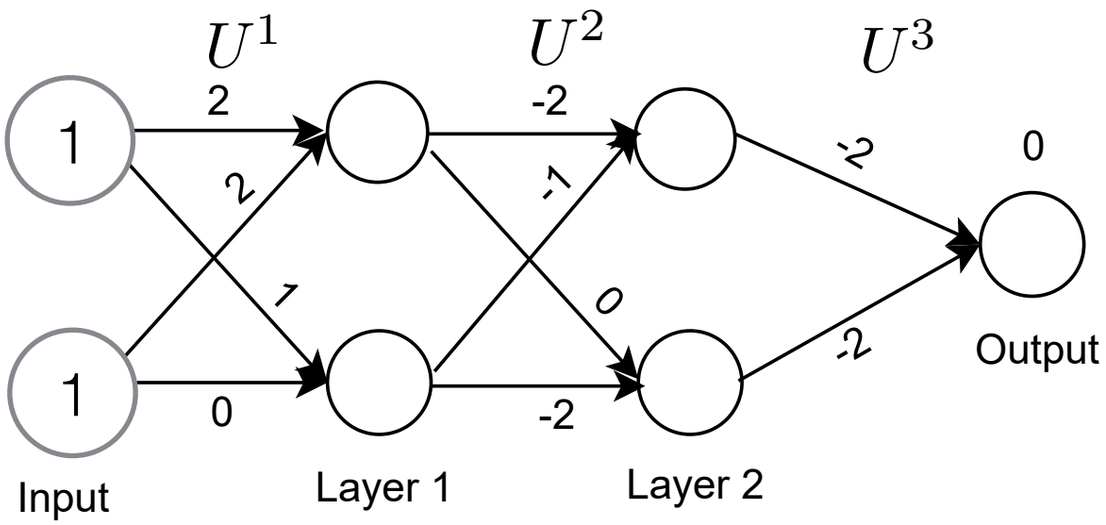


Gating network

$$\boldsymbol{\eta}^1(\mathbf{x}) = W^1\mathbf{x} + \mathbf{b}^1$$

$$\boldsymbol{\eta}^2(\mathbf{x}) = W^2\boldsymbol{\eta}^1(\mathbf{x}) + \mathbf{b}^2$$

Deep Linearly Gated Networks



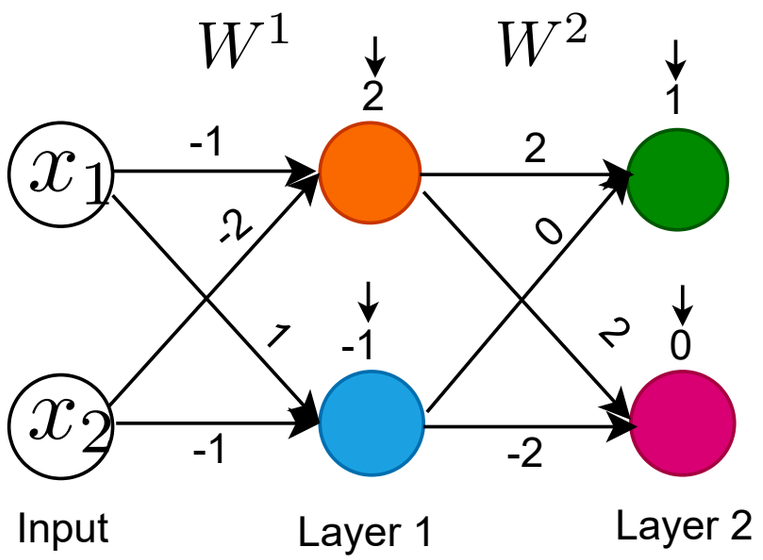
$$\mathbf{h}^1(\mathbf{x}) = \sigma(\boldsymbol{\eta}^1(\mathbf{x})) \circ (U^1\mathbf{1})$$

$$\mathbf{h}^2(\mathbf{x}) = \sigma(\boldsymbol{\eta}^2(\mathbf{x})) \circ (U^2\mathbf{h}^1(\mathbf{x}))$$

$$\widehat{y}(\mathbf{x}) = U^3\mathbf{h}^2(\mathbf{x})$$

Value network

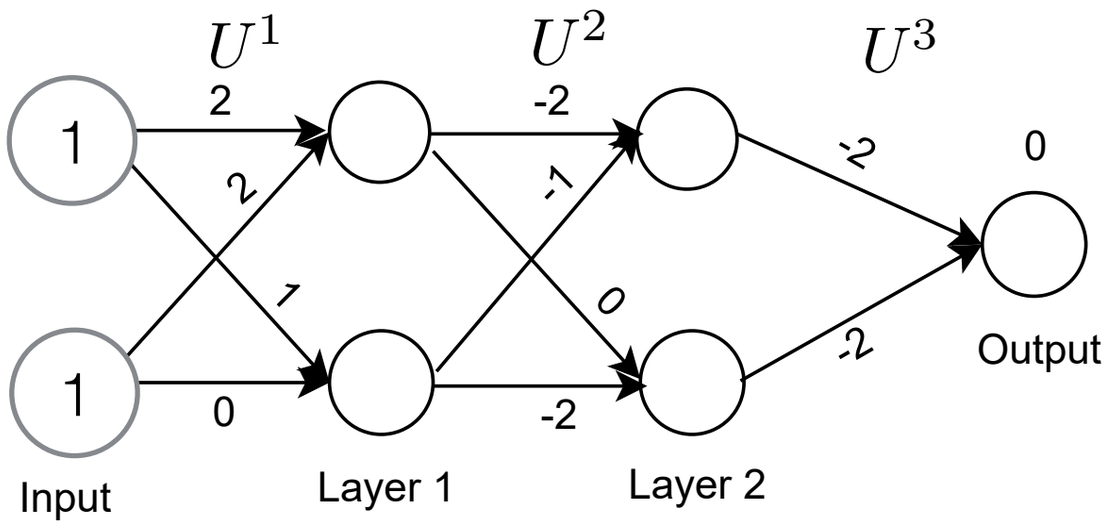
Deep Linearly Gated Networks



Gating network

$$\boldsymbol{\eta}^1(\mathbf{x}) = W^1\mathbf{x} + \mathbf{b}^1$$

$$\boldsymbol{\eta}^2(\mathbf{x}) = W^2\boldsymbol{\eta}^1(\mathbf{x}) + \mathbf{b}^2$$



$$\mathbf{h}^1(\mathbf{x}) = \sigma(\boldsymbol{\eta}^1(\mathbf{x})) \circ (U^1\mathbf{1})$$

$$\mathbf{h}^2(\mathbf{x}) = \sigma(\boldsymbol{\eta}^2(\mathbf{x})) \circ (U^2\mathbf{h}^1(\mathbf{x}))$$

$$\widehat{y}(\mathbf{x}) = U^3\mathbf{h}^2(\mathbf{x})$$

Value network

DLGNs : Performance on Benchmark Data

Table 1: ReLU network(R) and DLGN(D) test accuracy on CIFAR10 with a simple 5 layer convolutional architecture and also ResNet34 / ResNet110 architectures.

Conv5(R)	Conv5(D)	Res34(R)	Res34(D)	Res110(R)	Res110(D)
72.17	72.2	91	86	94	89

DLGNs : Performance on Benchmark Data

Table 3: Test accuracy on tabular datasets

Dataset char.			DLGN		Non tree algo						Tree algo			
Data	n	d	**dlgn**	**dlgnv**	relu	disnn	gln	svl	svm	sdt	cart	rf	zan	tao
Adult	29000	14	85.7	85.0	83.2	84.6	83.3	81.1	84.2	84.8	83.0	**86.1**	84.5	85.1
Bank	27000	16	**91.6**	91.3	89.6	91.2	90.4	89.9	90.7	91.3	90.3	91.3	91.1	90.9
Card	18000	23	**81.6**	81.2	78.4	80.9	80.4	80.6	81.1	81.3	77.6	81.2	81.3	80.6
Telesc	11000	10	**88.2**	87.2	87.7	87.7	85.6	79.4	87.1	86.4	82.9	88.1	87.0	84.9
Rice	2000	7	**92.6**	**92.6**	92.3	92.5	92.3	91.2	91.4	**92.6**	91.9	91.8	92.2	90.1
Stat	100	20	**77.6**	72.6	72.2	76.5	74.0	71.1	71.7	75.2	65.7	77.4	75.3	68.0
Spam	3000	57	94.4	94.0	94.1	94.0	92.1	92.3	93.1	93.4	89.4	**94.8**	93.1	91.2
Gyro	19000	8	98.8	98.3	98.4	98.4	98.3	98.3	98.1	98.4	98.7	**99.1**	98.6	98.6
Swar	14000	2400	**100**	**100**	**100**	87.5	99.6	**100**	**100**	**100**	99.9	**100**	**100**	99.9
Credit	12000	10	76.2	75.4	75.5	75.9	70.2	70.4	74.5	74.6	76.1	**78.2**	75.4	76.2
Elec	27000	7	82.8	80.6	82.7	79.6	77.5	73.6	78.5	75.8	86.3	**87.6**	76.5	76.4
Cover	396000	10	**94.7**	93.3	92.9	81.1	78.0	78.3	80.4	77.4	91.6	94.2	78.6	84.9
Pol	7000	26	**98.9**	98.0	98.2	97.9	93.8	87.8	97.7	98.2	96.6	97.4	97.6	95.4
House	9000	16	87.8	86.7	87.4	86.5	85.6	83.0	87.2	86.3	84.0	**88.0**	85.3	85.1
Mini	51000	50	**93.6**	93.4	93.2	91.0	88.6	89.3	89.3	88.9	89.9	93.1	91.7	88.8
Diab	50000	7	**60.6**	**60.6**	59.4	60.5	60.2	58.0	60.3	**60.6**	60.3	**60.6**	60.4	60.3
Jannis	40000	54	**78.6**	78.0	75.0	77.2	73.8	73.9	77.4	78.4	74.6	**78.6**	75.1	74.1
Bior	2000	419	76.2	74.3	76.9	73.6	74.3	73.5	77.5	76.9	70.5	**78.8**	72.5	72.9
Calif	14000	8	88.8	88.1	87.2	86.6	82.1	84.7	86.6	85.4	84.5	**89.0**	85.3	85.5
Heloc	7000	22	**71.9**	71.6	66.2	71.3	70.2	70.7	71.4	71.3	68.6	71.1	71.2	68.4

DLGNs : A Path Decomposition View

Path $\pi = (i_1, \ldots, i_L)$ is a sequence of hidden nodes.

$$\widehat{y}(\mathbf{x}) = \sum_{\pi \in \Pi} g_\pi f_\pi(\mathbf{x})$$

DLGNs : A Path Decomposition View

Path $\pi = (i_1, \ldots, i_L)$ is a sequence of hidden nodes.

$$\widehat{y}(\mathbf{x}) = \sum_{\pi \in \Pi} g_\pi f_\pi(\mathbf{x})$$

m nodes per layer with L layers imply m^L paths.

$$f_\pi(\mathbf{x}) = \prod_{\ell=1}^{L} \mathbf{1}\left(\boldsymbol{\eta}_{i_\ell}^{\ell}(\mathbf{x}) \geq 0\right)$$

In practice, this indicator is replaced by a sigmoid with temperature beta.

$$\boldsymbol{\eta}^{\ell}(\mathbf{x}) = W^{\ell}\boldsymbol{\eta}^{\ell-1}(\mathbf{x}) = V^{\ell}\mathbf{x}$$

DLGNs : A Path Decomposition View

Path $\pi = (i_1, \ldots, i_L)$ is a sequence of hidden nodes.

$$\widehat{y}(\mathbf{x}) = \sum_{\pi \in \Pi} g_\pi f_\pi(\mathbf{x})$$

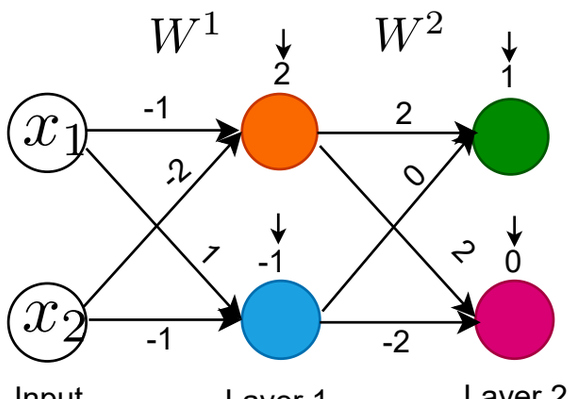
$$L$$

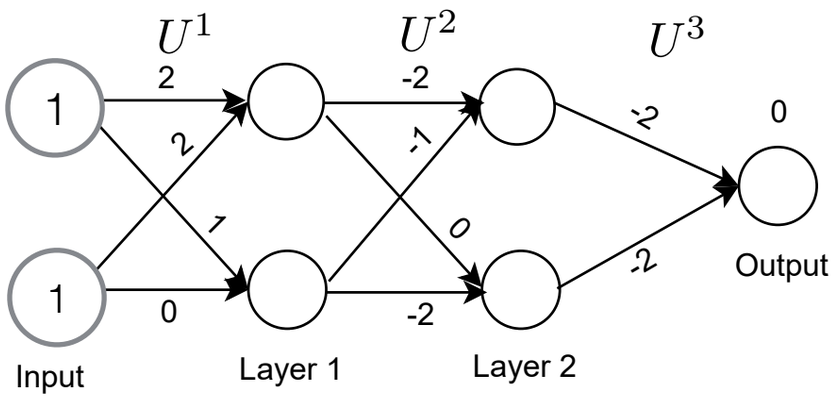
$$f_\pi(\mathbf{x}) = \prod_{\ell=1} \mathbf{1}\left(\boldsymbol{\eta}^\ell_{i_\ell}(\mathbf{x}) \geq 0\right)$$

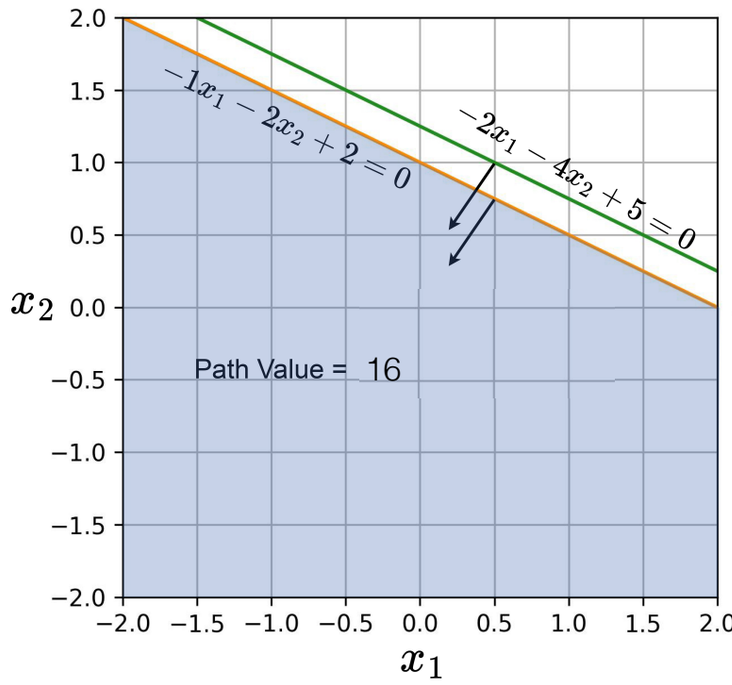
$$\boldsymbol{\eta}^\ell(\mathbf{x}) = W^\ell \boldsymbol{\eta}^{\ell-1}(\mathbf{x}) = V^\ell \mathbf{x}$$

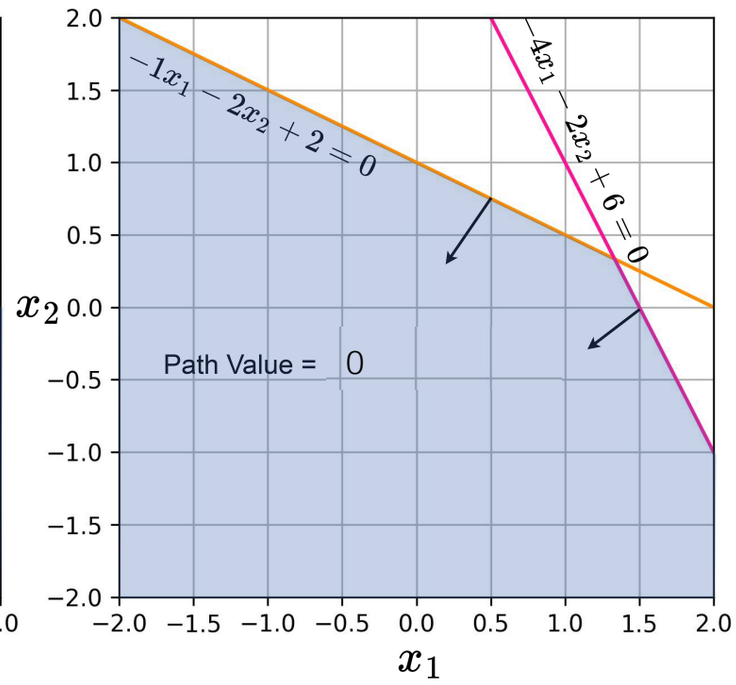
$$g_\pi = \mathbf{u}^1_{i_1} \left[\prod_{\ell=2}^{L} U^\ell_{i_\ell, i_{\ell-1}}\right] \mathbf{u}^{L+1}_{i_L}$$

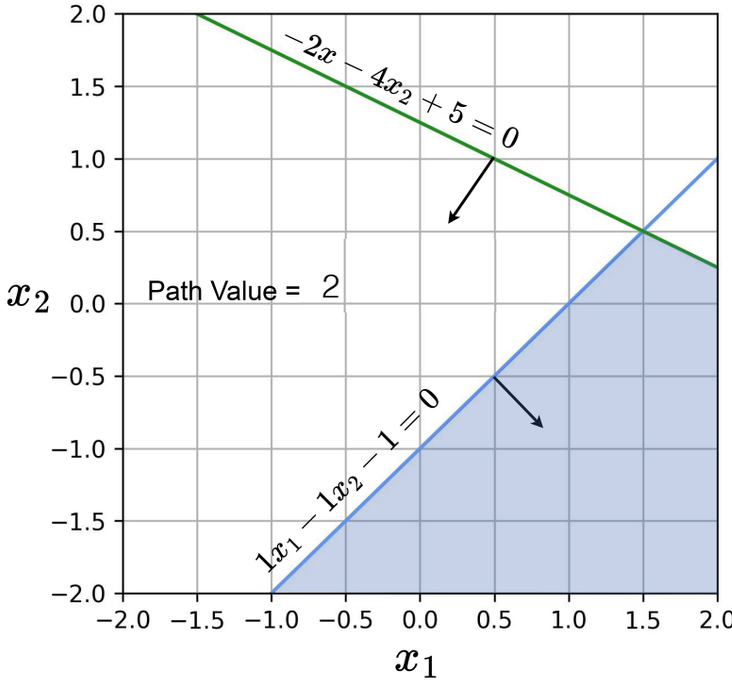
DLGNs : A Path Decomposition View

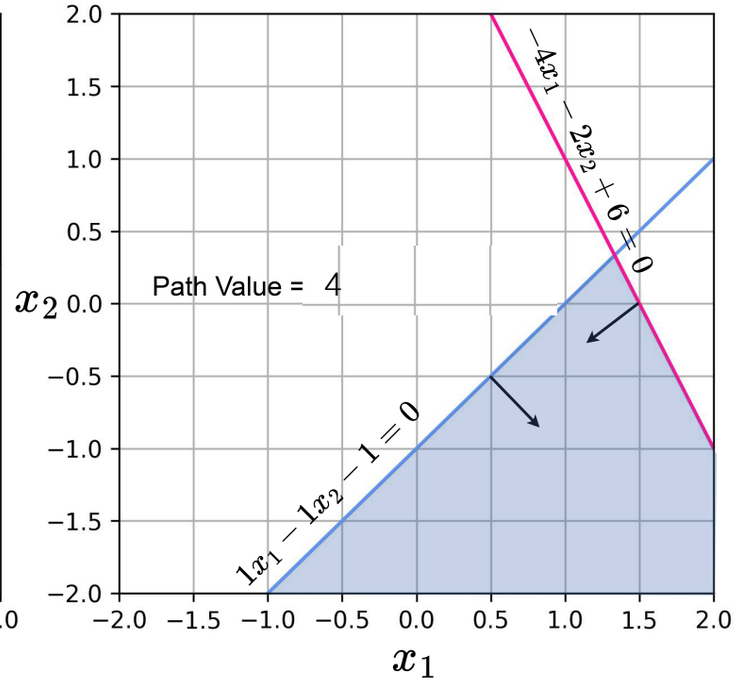








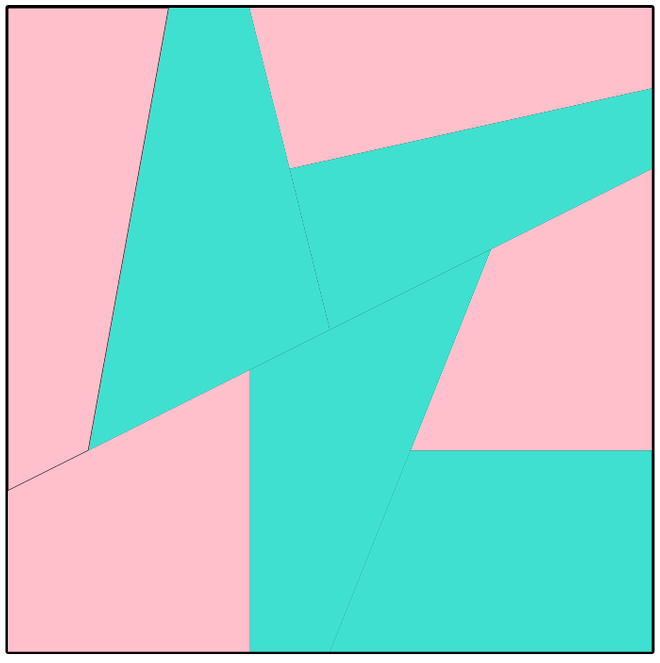


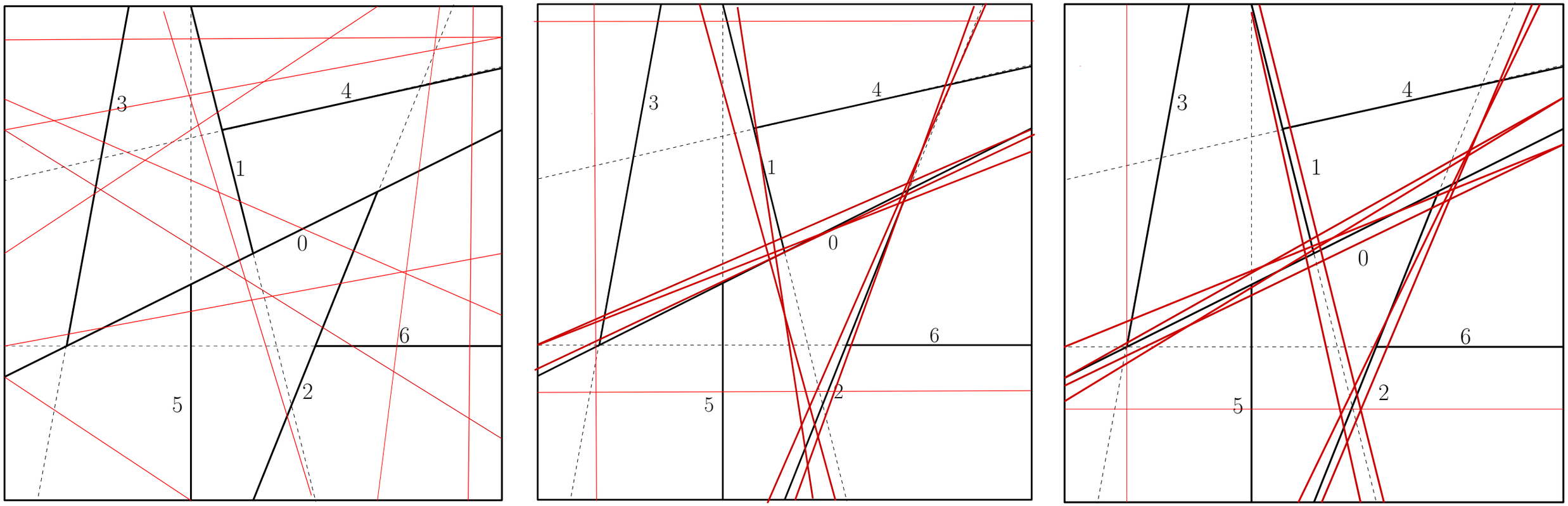


DLGN Path/Neuron Gating Functions

- DLGNs are a linear combination of (exponentially many) path gating functions.
- Each path gating function is a product of neuron gating functions.
- Each neuron gating function is a half-space.
- Neuron gating functions make a natural feature for DLGNs

Feature Discovery/Learning in DLGNs





A 3 Layer DLGN's hyperplanes after training

Feature Discovery/Learning in DLGNs

Table 1: Number of DLGN hyperplanes (after training) within a given distance of the label function ODT hyperplanes. The 15 ODT internal nodes are numbered 0 to 14, with 0 as the root. The distance of the closest (initial and trained) DLGN hyperplane to all the ODT hyperplanes is given in the last 2 rows. The DLGN has 4 hidden layers with 20 neurons in each layer, i.e. $L = 4, m = 20$.

Distance	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0.1	4	2	2	1	1	1	1	1	1	1	1	1	1	1	1
0.2	12	3	5	2	1	1	1	1	1	1	1	1	1	1	1
0.3	13	3	5	2	1	1	1	2	1	1	1	1	1	2	1
Closest Dist.(init)	1.26	1.22	1.21	1.25	1.23	1.24	1.23	1.17	1.27	1.28	1.17	1.23	1.27	1.22	1.26
Closest Dist.(final)	0.05	0.07	0.06	0.07	0.1	0.06	0.05	0.06	0.09	0.08	0.07	0.07	0.06	0.09	0.06

Feature Learning Conjecture

DLGN Ansatz

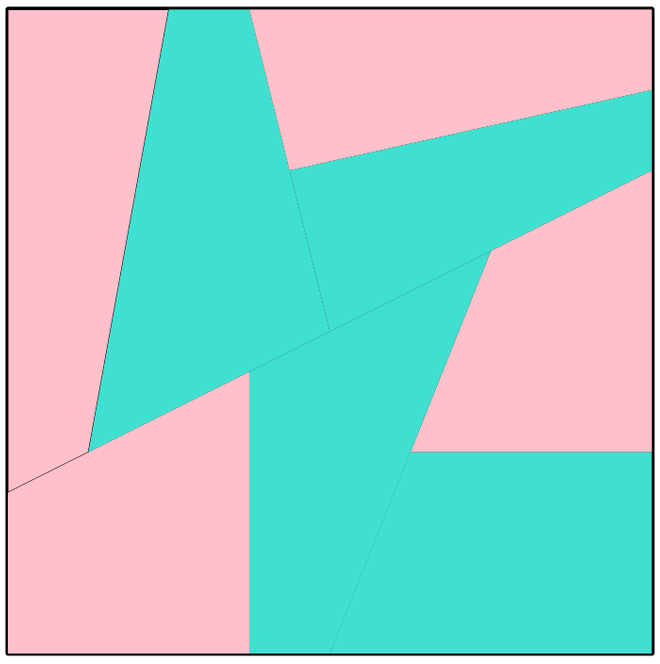
DLGN Hyperplanes move towards hyperplanes of maximum discontinuity in the labelling function.

Can we abstract further to get a Deep Neural Network Ansatz?

Outline

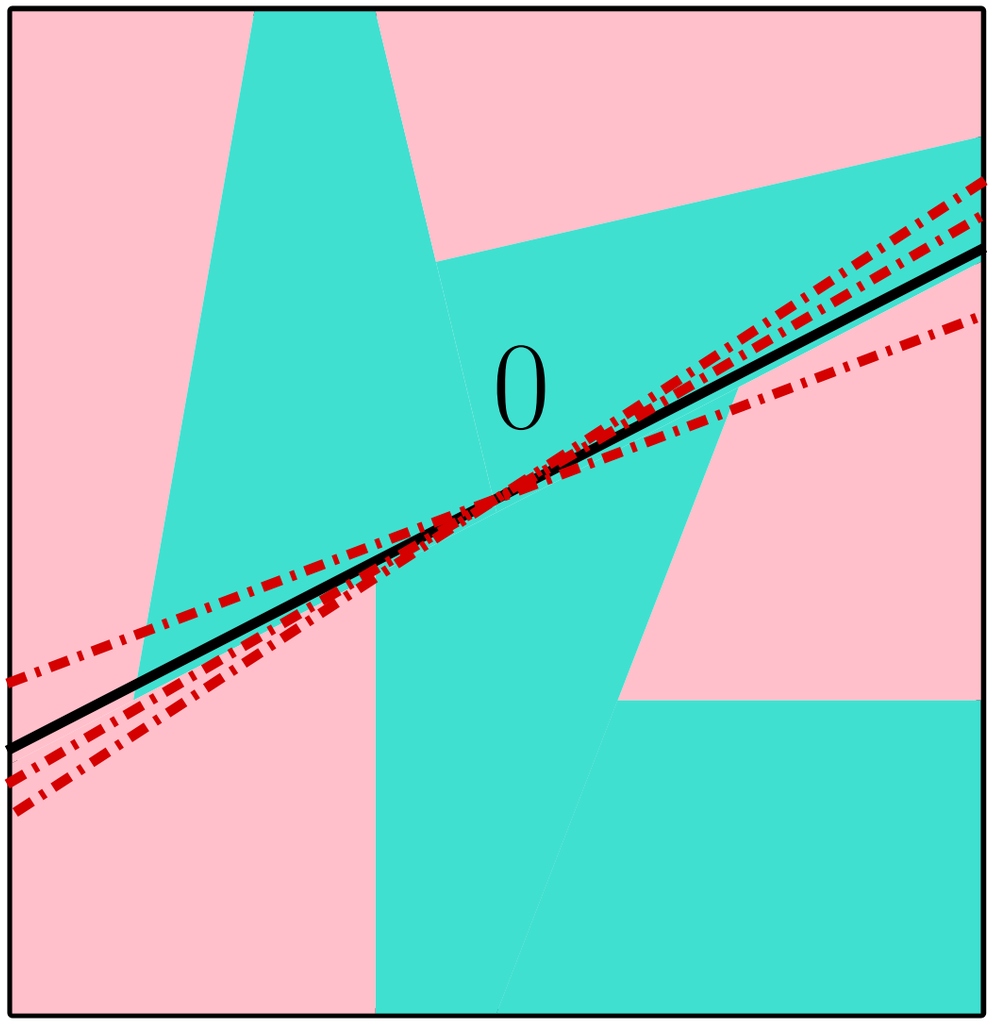
1. Feature Learning
2. Oblique Decision Trees (ODT)
3. Deep Linearly Gated Networks (DLGN)
4. Decision Tree Construction via DLGN
5. New Narratives

Building ODTs



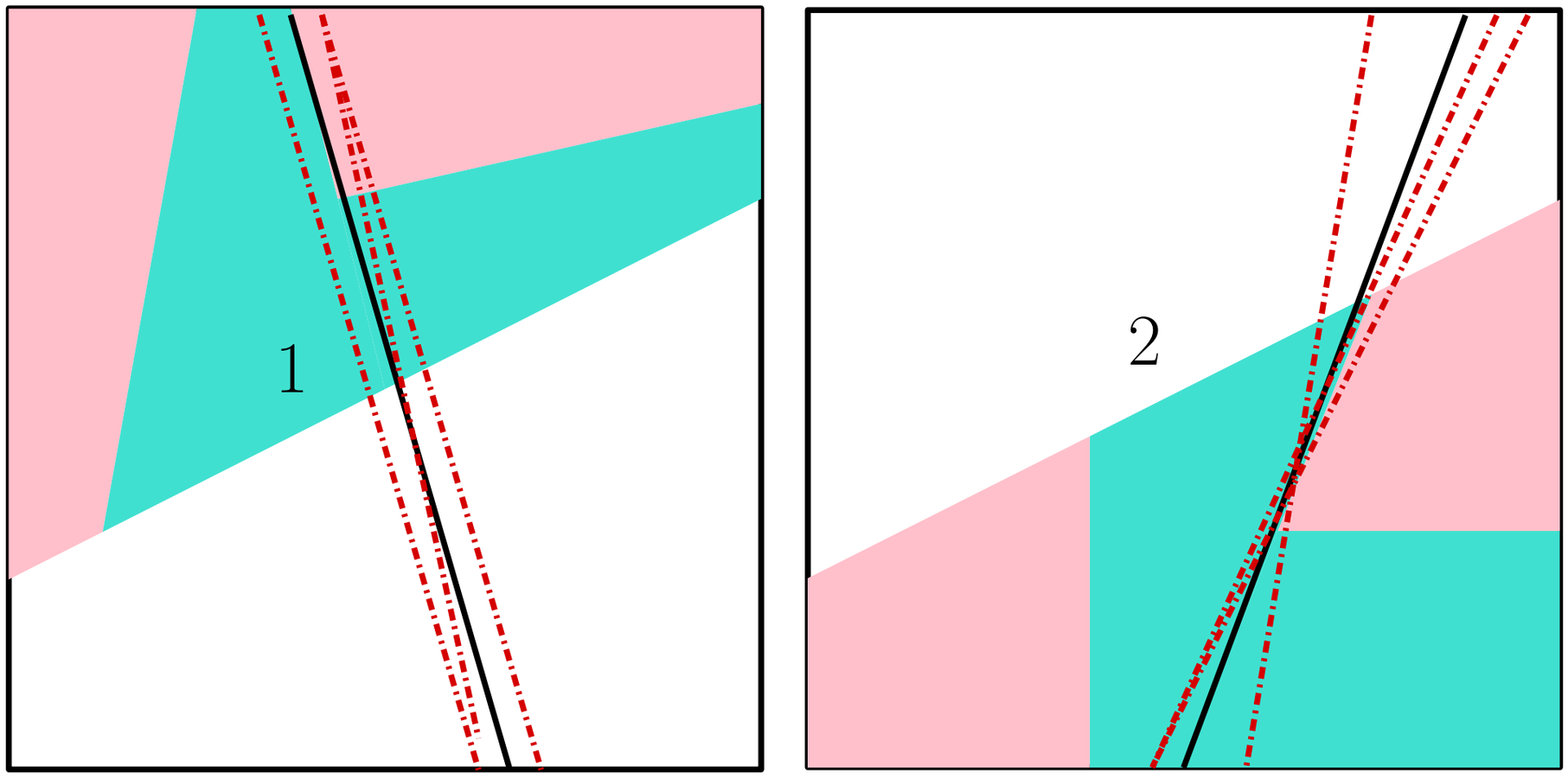
- Goal: Show control on the features of a learned DLGN
- Can we construct a decision tree from the parameters of a learned DLGN? (With no extra fine-tuning etc).
- Exploit the tendency of the DLGN hyperplanes/halfspaces to cluster around discontinuities in the true label function.

Building ODTs through DLGNs



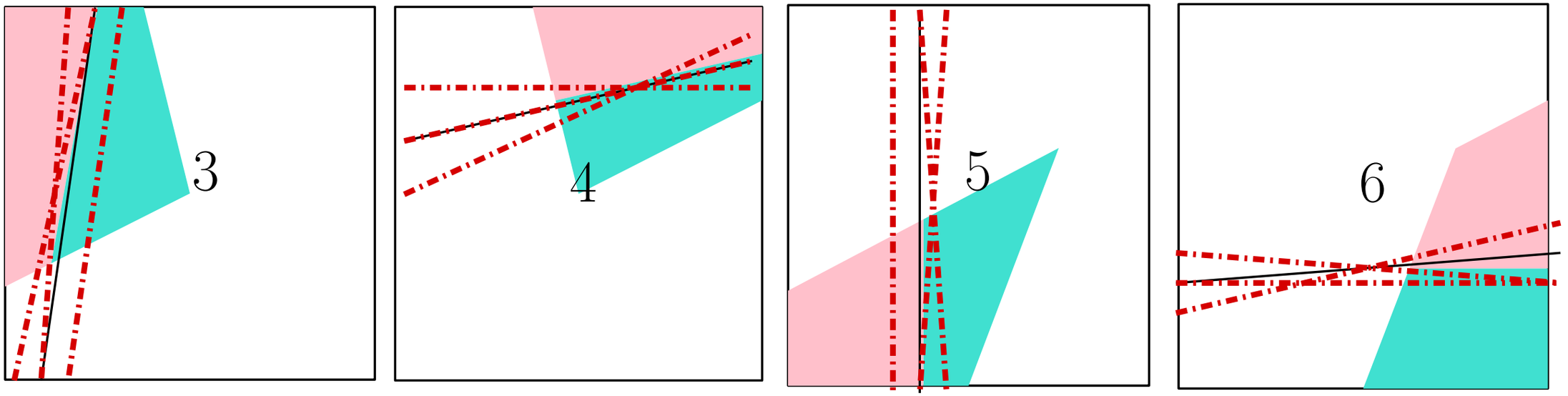
1. Cluster the hyperplanes in a learned DLGN model.
2. Pick the largest cluster.
3. Divide the data into two halves based on this hyperplane.

Building ODTs through DLGNs



4. Repeat the process over by training DLGNs on both halves.

Building ODTs through DLGNs



5. Stop when the data is pure.
6. Construct an ODT with the hyperplanes corresponding to the largest cluster centres.

DLGN Test Accuracy on ODT Labeled Data

Dataset char.			DLGN			Non tree algo						Tree algo				
Data	n	d	**dlgn**	**dlgns**	**dlgnv**	relu	disnn	gln	svl	svm	sdt	cart	rf	zan	tao	**dlgnd**
SDI	20000	20	96.1	97.4	**98.4**	90.6	89.8	66.8	63.4	76.5	68.9	57.3	65.9	84.9	60.3	93.1
SDII	30000	100	94.3	90.3	**95.8**	82.3	74.8	61.3	61.9	66.8	62.2	54.5	58.8	64.7	56.1	88.2
SDIII	50000	500	**65.1**	63.6	62.2	62.1	61.6	59.6	60.7	62.3	61.9	51.5	54.6	61.3	52.7	62.5

DLGN variants

Decision tree
Via DLGN

Outline

1. Feature Learning
2. Oblique Decision Trees (ODT)
3. Deep Linearly Gated Networks (DLGN)
4. Decision Tree Construction via DLGN
5. New Narratives

Feature Learning Narrative

$$\widehat{y}_{\ell,i}(\mathbf{x}) = \sigma(\beta V_i^{\ell}\mathbf{x}) \sum_{\pi \in \Pi_{\ell,i}} g_\pi \prod_{k \in \{1,\ldots,L\}\setminus\{\ell\}} \sigma(\beta V_{\pi_k}^{k}\mathbf{x})$$

Component of output thru neuron i in layer ℓ

Halfspace corr. to neuron i in layer ℓ

Paths through neuron i in layer ℓ

Halfspace corr. to layer k neuron in path π

The half space of a given neuron is influenced by intersection of other layer neuron half spaces!

Thus, half spaces that are only locally useful, can still be learned, as the other layers set the context.

Potential Applications

- The setup of feature learning with DLGNs is intricate and complex enough to study several mysteries in deep learning in an alternate fashion. Such as:
  - Why does pruning work?
  - What role does depth serve?
  - Why does model distillation work?