



# Scalable Rule-Based Representation Learning for Interpretable Classification

Zhuo Wang<sup>1</sup>, Wei Zhang<sup>3\*</sup>, Ning Liu<sup>4</sup>, Jianyong Wang<sup>1,2\*</sup>

<sup>1</sup> Department of Computer Science and Technology, Tsinghua University

<sup>2</sup> Jiangsu Collaborative Innovation Center for Language Ability, Jiangsu Normal University

<sup>3</sup> School of Computer Science and Technology, East China Normal University

<sup>4</sup> School of Software, Shandong University

wang-z18@mails.tsinghua.edu.cn, {zhangwei.thu2011, victorliucs}@gmail.com,  
jianyong@tsinghua.edu.cn

# Background & Motivation

- Rule-based models still play an important role in domains demanding high model interpretability, such as medicine, finance, and politics.

- Transparent inner structures
- Good model expressivity



Medicine

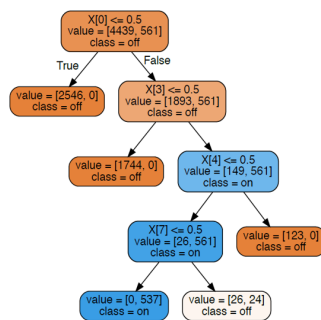


Finance



Politics

- However, conventional rule-based models are hard to optimize, especially on large data sets, due to their discrete parameters and structures, which limits their application scope.



Decision Tree

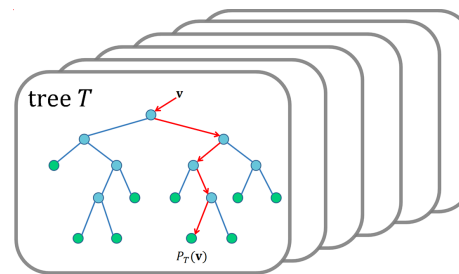
if male and adult then survival probability 21% (19%–23%)  
 else if 3rd class then survival probability 44% (38%–51%)  
 else if 1st class then survival probability 96% (92%–99%)  
 else survival probability 88% (82%–94%)

Rule List

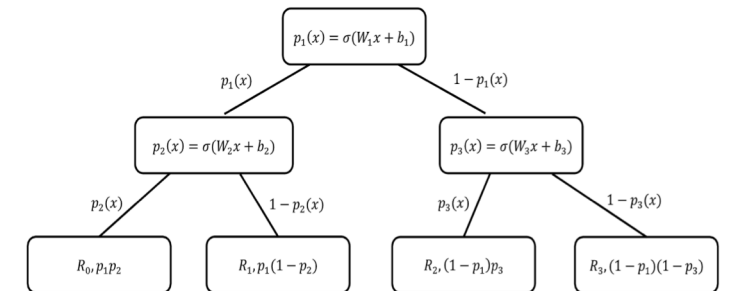
if ( sunny and hot ) or ( cloudy and hot )  
 or ( sunny and thirsty and bored )  
 or ( bored and tired ) or ( thirty and tired )  
 or ( code running ) or ( friends away and bored )  
 or ( sunny and want to swim ) or ( just feel like it )  
 then go to beach  
 else work

Rule Set

Conventional Rule-based Models



Ensemble Methods



Soft/fuzzy Rules

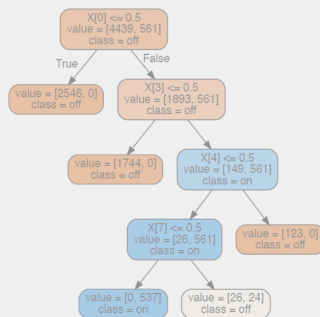
# Background & Motivation

- Rule-based models still play an important role in domains demanding high model interpretability, such as medicine, finance, and politics.
  - Transparent inner structures
  - Good model expressivity



How to improve the scalability of rule-based models while keeping their interpretability?

- How on limits their application scope.



Decision Tree

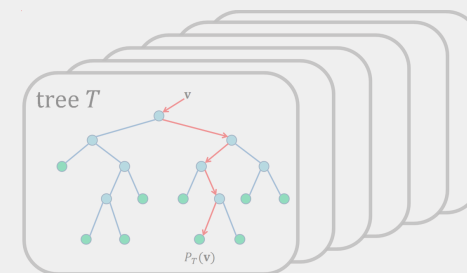
if male and adult then survival probability 21% (19%–23%)  
else if 3rd class then survival probability 44% (38%–51%)  
else if 1st class then survival probability 96% (92%–99%)  
else survival probability 88% (82%–94%)

Rule List

if ( sunny and hot ) or ( cloudy and hot )  
or ( sunny and thirsty and bored )  
or ( bored and tired ) or ( thirty and tired )  
or ( code running ) or ( friends away and bored )  
or ( sunny and want to swim ) or ( just feel like it )  
then go to beach  
else work

Rule Set

## Conventional Rule-based Models



## Ensemble Methods

# Rule-based Representation Learner

A Rule-based Representation Learner (RRL) is a hierarchical model consisting of three different types of layers. Each layer in RRL has trainable edges connected with its previous layer.

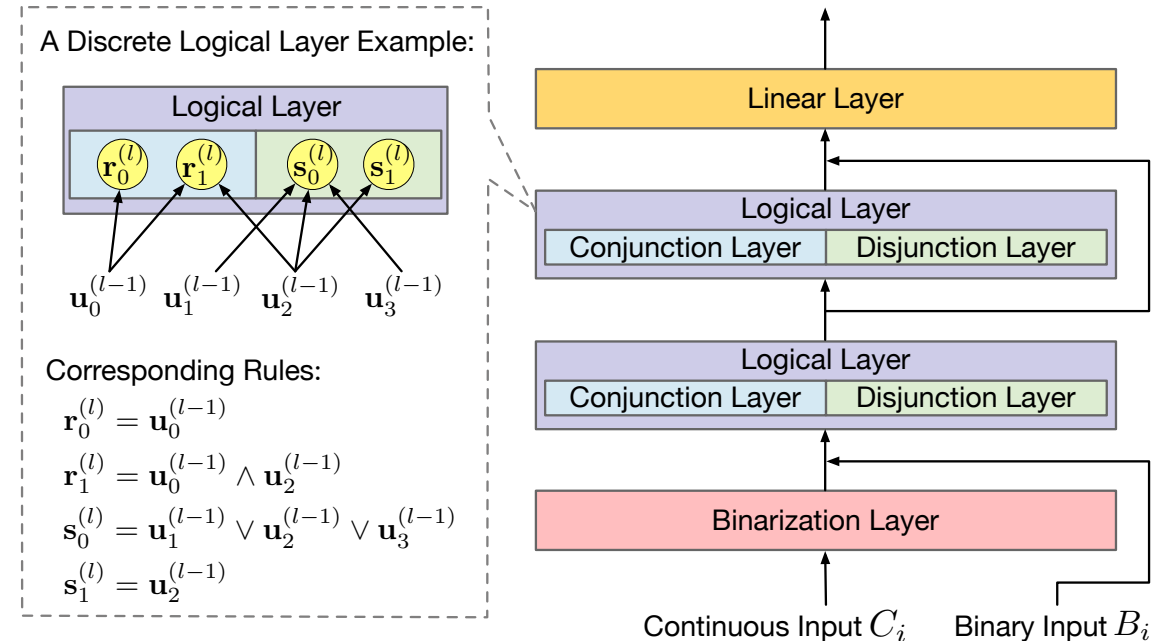


Figure: A Rule-based Representation Learner example. The dashed box shows an example of a discrete logical layer and its corresponding rules.



# Rule-based Representation Learner

A Rule-based Representation Learner (RRL) is a hierarchical model consisting of three different types of layers. Each layer in RRL has trainable edges connected with its previous layer.

- Binarization Layer
  - Discretize continuous features end-to-end

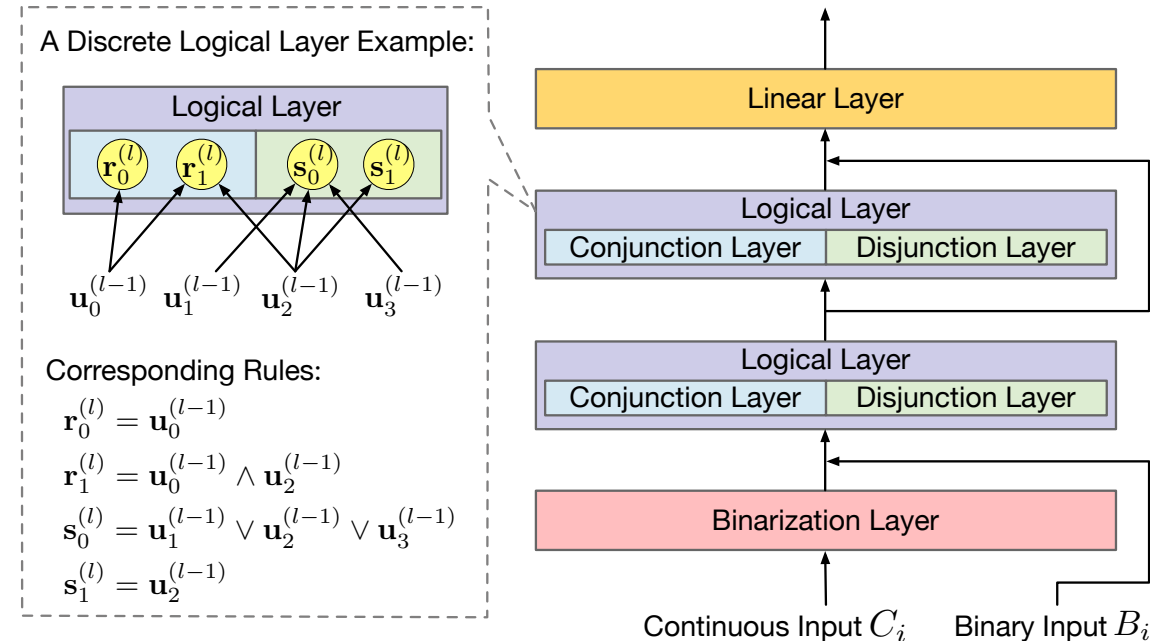


Figure: A Rule-based Representation Learner example. The dashed box shows an example of a discrete logical layer and its corresponding rules.

# Rule-based Representation Learner

A Rule-based Representation Learner (RRL) is a hierarchical model consisting of three different types of layers. Each layer in RRL has trainable edges connected with its previous layer.

- Binarization Layer
  - Discretize continuous features end-to-end
- Logical Layer
  - Learn data representation using logical rules automatically
  - Stacked logical layers can learn rules in more complex forms, e.g., CNF or DNF rules

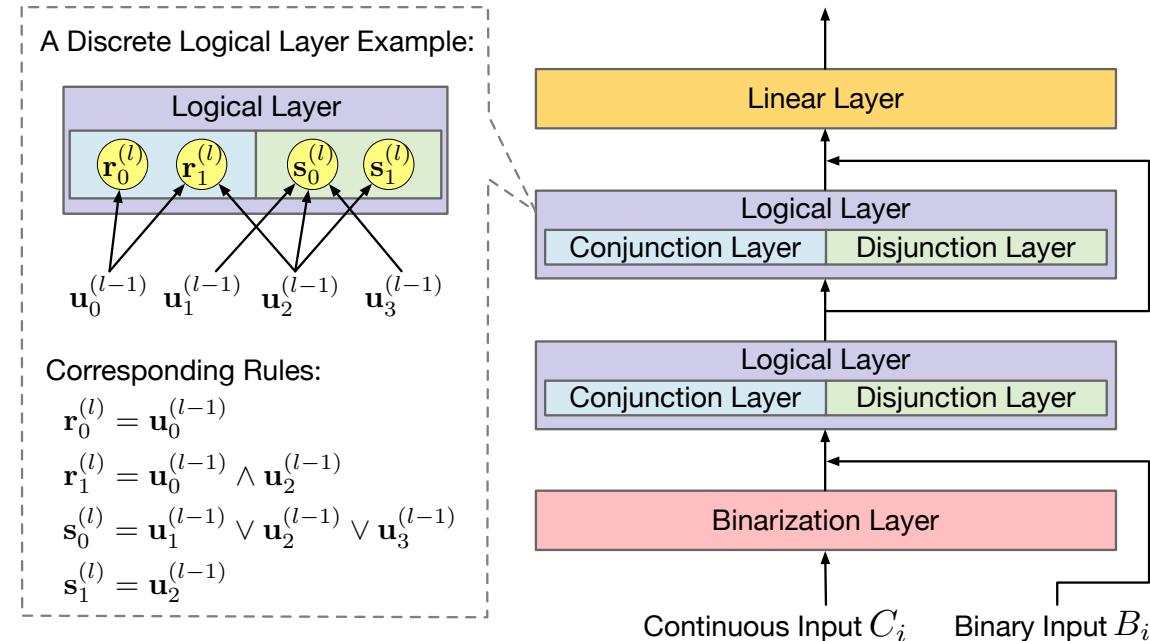


Figure: A Rule-based Representation Learner example. The dashed box shows an example of a discrete logical layer and its corresponding rules.

# Rule-based Representation Learner

A Rule-based Representation Learner (RRL) is a hierarchical model consisting of three different types of layers. Each layer in RRL has trainable edges connected with its previous layer.

- Binarization Layer
  - Discretize continuous features end-to-end
- Logical Layer
  - Learn data representation using logical rules automatically
  - Stacked logical layers can learn rules in more complex forms, e.g., CNF or DNF rules
- Linear Layer
  - Learn the linear part of the data
  - Evaluate rule importance using weights

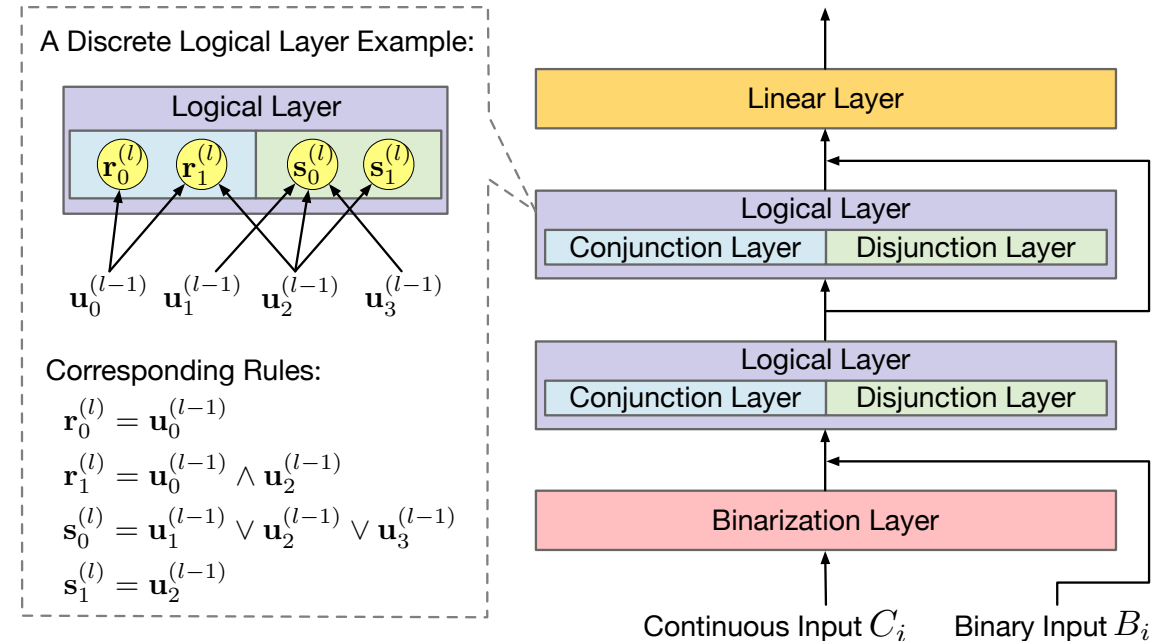


Figure: A Rule-based Representation Learner example. The dashed box shows an example of a discrete logical layer and its corresponding rules.

# Rule-based Representation Learner

A Rule-based Representation Learner (RRL) is a hierarchical model consisting of three different types of layers. Each layer in RRL has trainable edges connected with its previous layer.

- Binarization Layer
  - Discretize continuous features end-to-end
- Logical Layer
  - Learn data representation using logical rules automatically
  - Stacked logical layers can learn rules in more complex forms, e.g., CNF or DNF rules
- Linear Layer
  - Learn the linear part of the data
  - Evaluate rule importance using weights
- Skip connection
  - Skip the unnecessary layer automatically

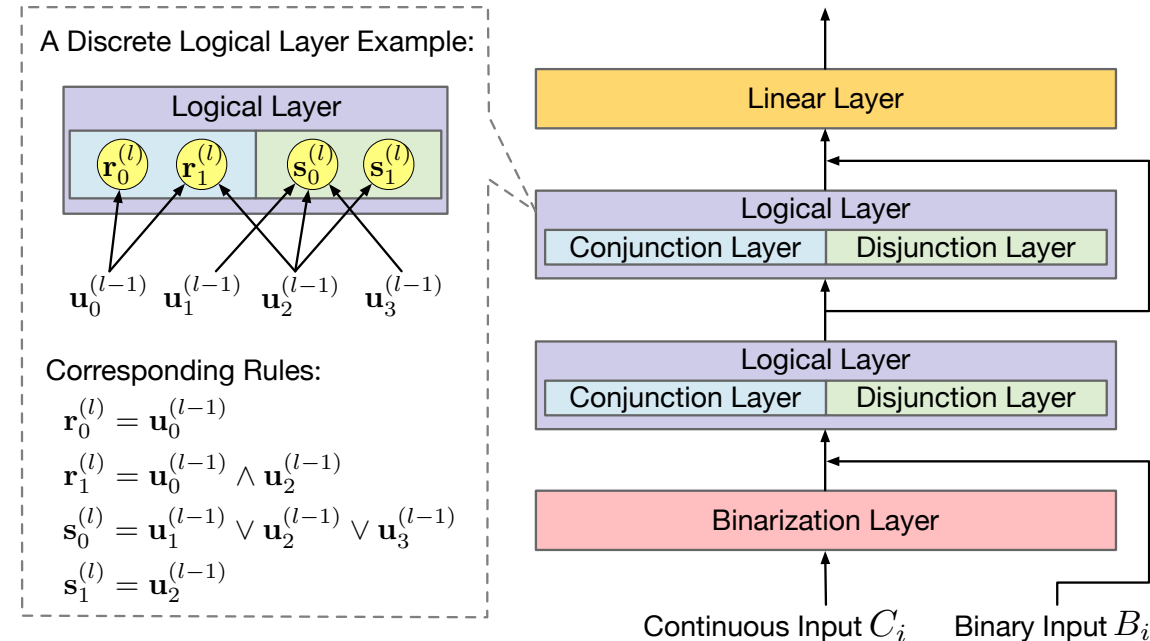
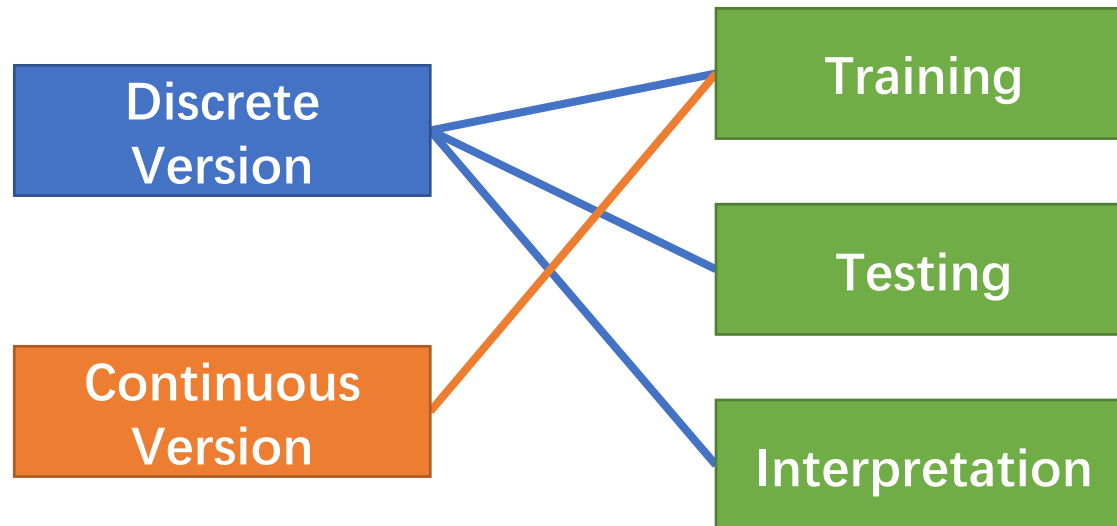


Figure: A Rule-based Representation Learner example. The dashed box shows an example of a discrete logical layer and its corresponding rules.

# Logical Layer

---

- One logical layer consists of one conjunction layer and one disjunction layer.
- One node corresponds to one logical operator.
  - Nodes in conjunction layer: **AND**
  - Nodes in disjunction layer: **OR**
- To learn data representations using logical rules automatically, logical layers are designed to have a discrete version and a continuous version. These two versions share the same parameters.



# Logical Layer

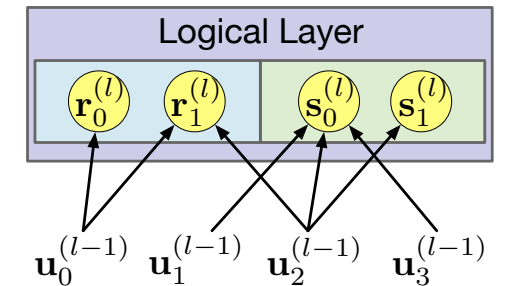
- Discrete Version

- Edge connections indicate which variables will involve the logical operation.
- Let  $r_i^{(l)}$  and  $s_i^{(l)}$  denote the  $i$ -th node in conjunction and disjunction layer, respectively.  $W^{(l,0)}$  and  $W^{(l,1)}$  are the adjacency matrices.

$$\mathbf{r}_i^{(l)} = \bigwedge_{W_{i,j}^{(l,0)}=1} \mathbf{u}_j^{(l-1)}, \quad \mathbf{s}_i^{(l)} = \bigvee_{W_{i,j}^{(l,1)}=1} \mathbf{u}_j^{(l-1)}$$

- Although the discrete logical layers have good interpretability, they are hard to train for their discrete parameters and non-differentiable structures.

A Discrete Logical Layer Example:



Corresponding Rules:

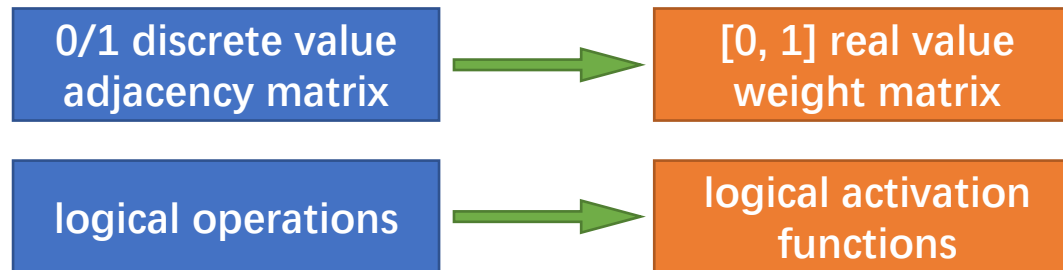
$$\begin{aligned} \mathbf{r}_0^{(l)} &= \mathbf{u}_0^{(l-1)} \\ \mathbf{r}_1^{(l)} &= \mathbf{u}_0^{(l-1)} \wedge \mathbf{u}_2^{(l-1)} \\ \mathbf{s}_0^{(l)} &= \mathbf{u}_1^{(l-1)} \vee \mathbf{u}_2^{(l-1)} \vee \mathbf{u}_3^{(l-1)} \\ \mathbf{s}_1^{(l)} &= \mathbf{u}_2^{(l-1)} \end{aligned}$$

# Logical Layer

---

- Continuous Version

- The continuous version is differentiable, and when we discretize the parameters of a continuous logical layer, we can obtain its corresponding discrete logical layer.
- To achieve this, it need to:



- The logical activation functions proposed by Payani and Fekri(2019) are:

$$\hat{\mathbf{r}}_i^{(l)} = \text{Conj}(\hat{\mathbf{u}}^{(l-1)}, \hat{W}_i^{(l,0)}) = \prod_{j=1}^n F_c(\hat{\mathbf{u}}_j^{(l-1)}, \hat{W}_{i,j}^{(l,0)})$$

$$\hat{\mathbf{s}}_i^{(l)} = \text{Disj}(\hat{\mathbf{u}}^{(l-1)}, \hat{W}_i^{(l,1)}) = 1 - \prod_{j=1}^n (1 - F_d(\hat{\mathbf{u}}_j^{(l-1)}, \hat{W}_{i,j}^{(l,1)}))$$

# Improved Logical Activation Functions

---

- However, these logical activation functions suffer from the serious vanishing gradient problem.
- Using the multiplications to simulate the logical operations is the main reason for vanishing gradients.

$$\left| \frac{\partial \hat{\mathbf{r}}_i^{(l)}}{\partial \hat{W}_{i,j}^{(l,0)}} = (\hat{\mathbf{u}}_j^{(l-1)} - 1) \cdot \prod_{k \neq j} F_c(\hat{\mathbf{u}}_k^{(l-1)}, \hat{W}_{i,k}^{(l,0)}) \right.$$

$$F_c(\cdot) \in [0, 1]$$

$$\left| \frac{\partial \hat{\mathbf{s}}_i^{(l)}}{\partial \hat{W}_{i,j}^{(l,1)}} = \hat{\mathbf{u}}_j^{(l-1)} \cdot \prod_{k \neq j} (1 - F_d(\hat{\mathbf{u}}_k^{(l-1)}, \hat{W}_{i,k}^{(l,1)})) \right.$$

$$F_d(\cdot) \in [0, 1]$$



# Improved Logical Activation Functions

---

- However, these logical activation functions suffer from the serious vanishing gradient problem.
- Using the multiplications to simulate the logical operations is the main reason for vanishing gradients.

$$\left| \frac{\partial \hat{\mathbf{r}}_i^{(l)}}{\partial \hat{W}_{i,j}^{(l,0)}} = (\hat{\mathbf{u}}_j^{(l-1)} - 1) \cdot \prod_{k \neq j} F_c(\hat{\mathbf{u}}_k^{(l-1)}, \hat{W}_{i,k}^{(l,0)}) \right.$$

$$F_c(\cdot) \in [0, 1]$$

$$\left| \frac{\partial \hat{\mathbf{s}}_i^{(l)}}{\partial \hat{W}_{i,j}^{(l,1)}} = \hat{\mathbf{u}}_j^{(l-1)} \cdot \prod_{k \neq j} (1 - F_d(\hat{\mathbf{u}}_k^{(l-1)}, \hat{W}_{i,k}^{(l,1)})) \right.$$


$$F_d(\cdot) \in [0, 1]$$

$$0.9^{100} \approx 0$$

# Improved Logical Activation Functions

---

- One straightforward idea is to convert multiplications into additions using logarithm.

$$\log\left(\prod_{j=1}^n F_c(\mathbf{h}_j, W_{i,j})\right) = \sum_{j=1}^n \log(F_c(\mathbf{h}_j, W_{i,j}))$$


- However, after taking the logarithm, the logical activation functions cannot keep the characteristics of logical operations any more.

# Improved Logical Activation Functions

- We need a projection function  $g(x)$  to fix it, and three conditions must be satisfied:

- (i)  $g(0) = e^0$  (ii)  $\lim_{x \rightarrow -\infty} g(x) = \lim_{x \rightarrow -\infty} e^x$  (iii)  $\lim_{x \rightarrow -\infty} \frac{e^x}{g(x)} = 0$

- We choose  $g(x) = \frac{-1}{-1+x}$ , and the improvement can be summarized as  $\mathbb{P}(v) = \frac{-1}{-1+\log(v)}$

- Now, the improved logical activation functions are:

$$\text{Conj}_+(\mathbf{h}, W_i) = \mathbb{P}\left(\prod_{j=1}^n (F_c(\mathbf{h}_j, W_{i,j}) + \epsilon)\right)$$

$$\text{Disj}_+(\mathbf{h}, W_i) = 1 - \mathbb{P}\left(\prod_{j=1}^n (1 - F_d(\mathbf{h}_j, W_{i,j}) + \epsilon)\right)$$

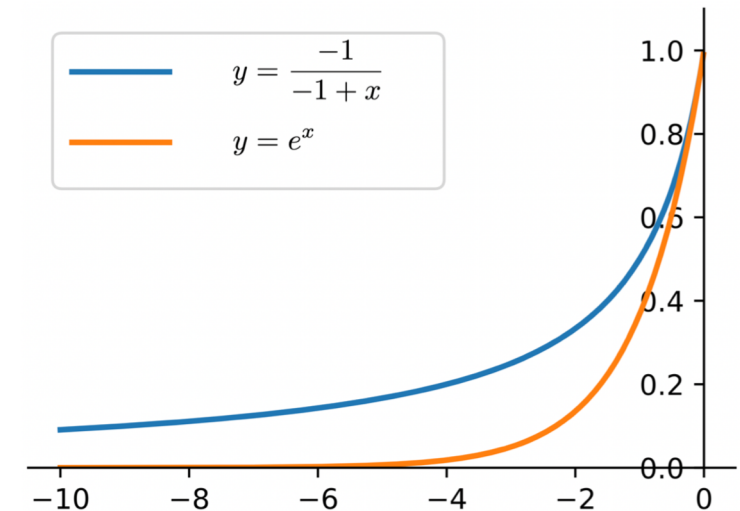


Figure:  $y = \frac{-1}{-1+x}$  and  $y = e^x$

# Binarization Layer

---

- By combining one binarization layer and one logical layer, we can automatically choose the appropriate bins for feature discretization (binarization), i.e., binarizing features in an end-to-end way.
- For the  $j$ -th continuous feature to be binarized, there are  $k$  lower bounds  $(\mathcal{T}_{j,1}, \mathcal{T}_{j,2}, \dots, \mathcal{T}_{j,k})$  and  $k$  upper bounds  $(\mathcal{H}_{j,1}, \mathcal{H}_{j,2}, \dots, \mathcal{H}_{j,k})$ .
  - The output is  $Q_j$ , and  $q(x) = 1_{x>0}$
  - $Q_j = [q(c_j - \mathcal{T}_{j,1}), \dots, q(c_j - \mathcal{T}_{j,k}), q(\mathcal{H}_{j,1} - c_j), \dots, q(\mathcal{H}_{j,k} - c_j)]$

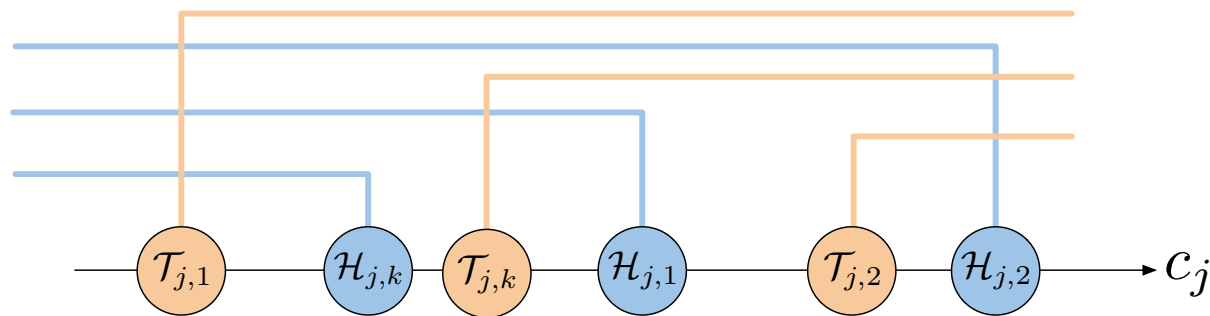


Figure: The visualization of  $k$  lower and  $k$  upper bounds.

# Binarization Layer

- By combining one binarization layer and one logical layer, we can automatically choose the appropriate bins for feature discretization (binarization), i.e., binarizing features in an end-to-end way.
- For the  $j$ -th continuous feature to be binarized, there are  $k$  lower bounds  $(\mathcal{T}_{j,1}, \mathcal{T}_{j,2}, \dots, \mathcal{T}_{j,k})$  and  $k$  upper bounds  $(\mathcal{H}_{j,1}, \mathcal{H}_{j,2}, \dots, \mathcal{H}_{j,k})$ .
  - The output is  $Q_j$ , and  $q(x) = 1_{x>0}$
  - $Q_j = [q(c_j - \mathcal{T}_{j,1}), \dots, q(c_j - \mathcal{T}_{j,k}), q(\mathcal{H}_{j,1} - c_j), \dots, q(\mathcal{H}_{j,k} - c_j)]$

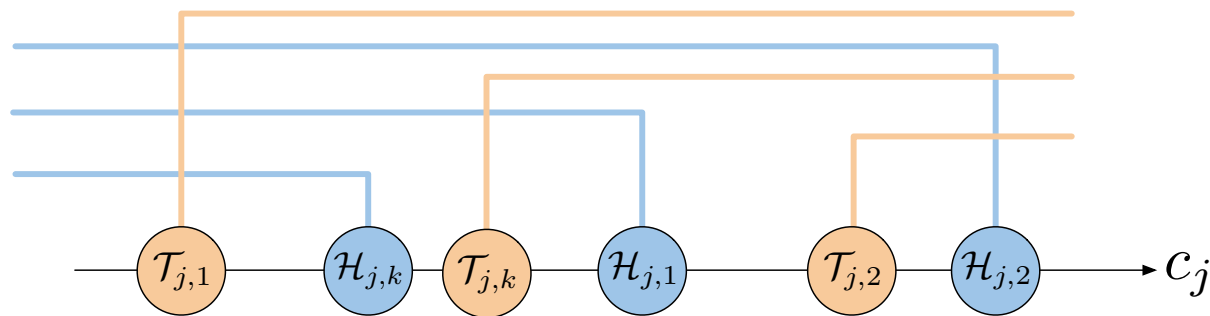
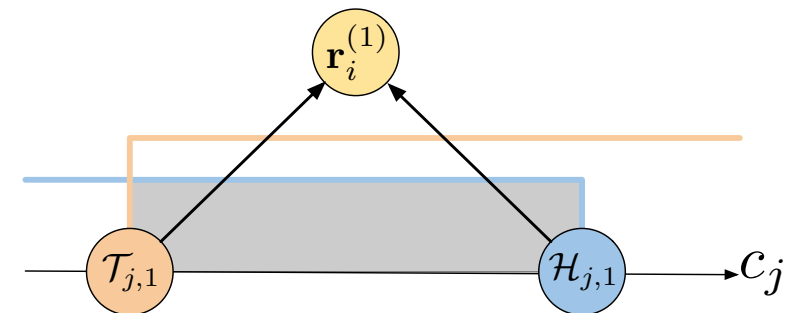


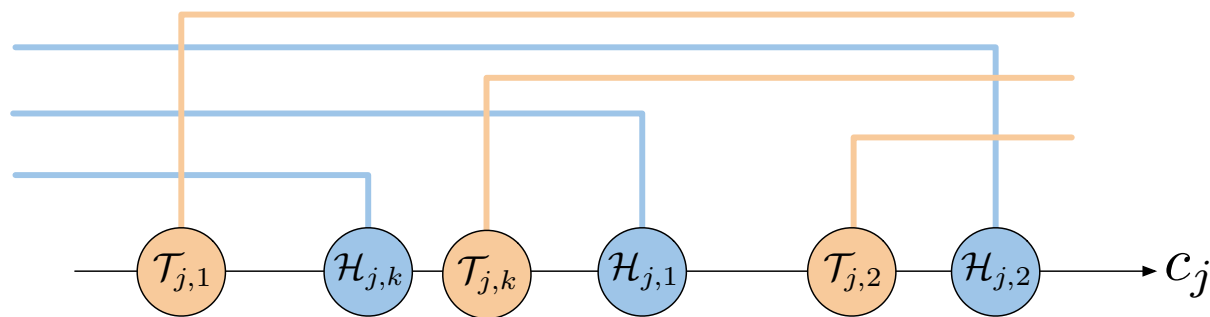
Figure: The visualization of  $k$  lower and  $k$  upper bounds.



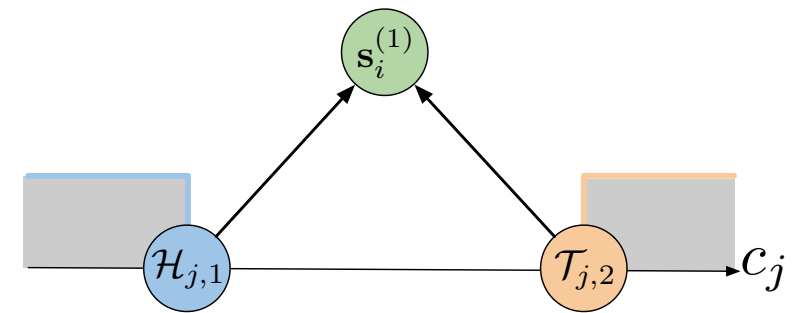
The bounds selected by a conjunction node.

# Binarization Layer

- By combining one binarization layer and one logical layer, we can automatically choose the appropriate bins for feature discretization (binarization), i.e., binarizing features in an end-to-end way.
- For the  $j$ -th continuous feature to be binarized, there are  $k$  lower bounds  $(\mathcal{T}_{j,1}, \mathcal{T}_{j,2}, \dots, \mathcal{T}_{j,k})$  and  $k$  upper bounds  $(\mathcal{H}_{j,1}, \mathcal{H}_{j,2}, \dots, \mathcal{H}_{j,k})$ .
  - The output is  $Q_j$ , and  $q(x) = 1_{x>0}$
  - $Q_j = [q(c_j - \mathcal{T}_{j,1}), \dots, q(c_j - \mathcal{T}_{j,k}), q(\mathcal{H}_{j,1} - c_j), \dots, q(\mathcal{H}_{j,k} - c_j)]$



The visualization of  $k$  lower and  $k$  upper bounds.



The bounds selected by a disjunction node.

# Gradient Grafting

- Although RRL can be differentiable with the continuous logical layers, it is challenging to search for a discrete solution in a continuous space.
- The gradients of RRL at discrete points have no useful information in most cases.
- Inspired by plant grafting, we propose a new training method, called Gradient Grafting, that can effectively train RRL.

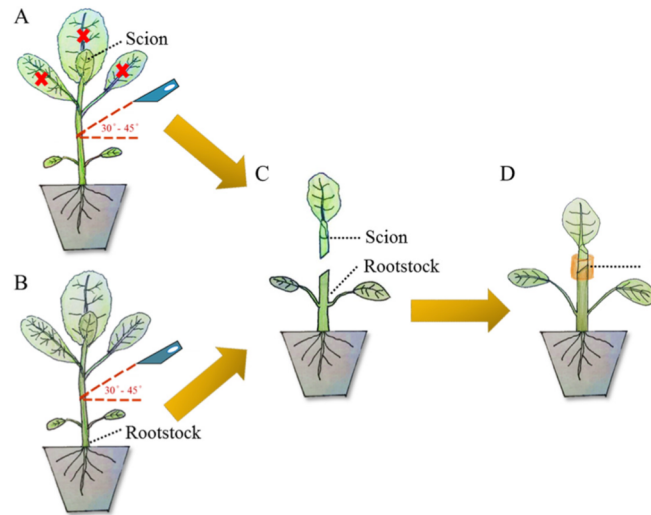


Figure: A plant grafting example (Chen et al., 2019).

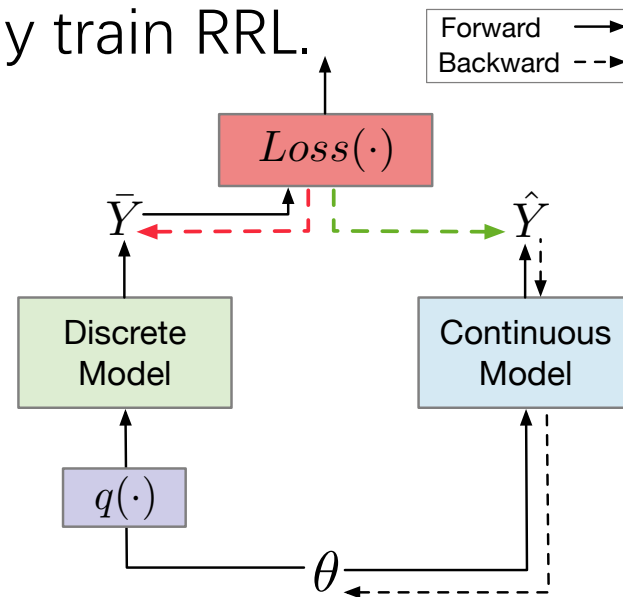


Figure: A simplified computation graph of Gradient Grafting. Arrows with solid lines represent forward pass while arrows with dashed lines represent backpropagation. The green arrow denotes the grafted gradient, a copy of the gradient represented by the red arrow.

$$\theta^{t+1} = \theta^t - \eta \frac{\partial \mathcal{L}(\bar{Y})}{\partial \bar{Y}} \cdot \frac{\partial \hat{Y}}{\partial \theta^t}$$

Scion
Rootstock

# Model Interpretation

- After training with Gradient Grafting, the discrete RRL can be used for testing and interpretation.
- RRL is easy to interpret, for we can simply consider it as:
  - A feature learner
  - A linear classifier
- It is easy to obtain a trade-off between the classification accuracy and model complexity of RRL.
  - The number of logical layers
  - The number of nodes in each logical layer
  - The L1/L2 regularization

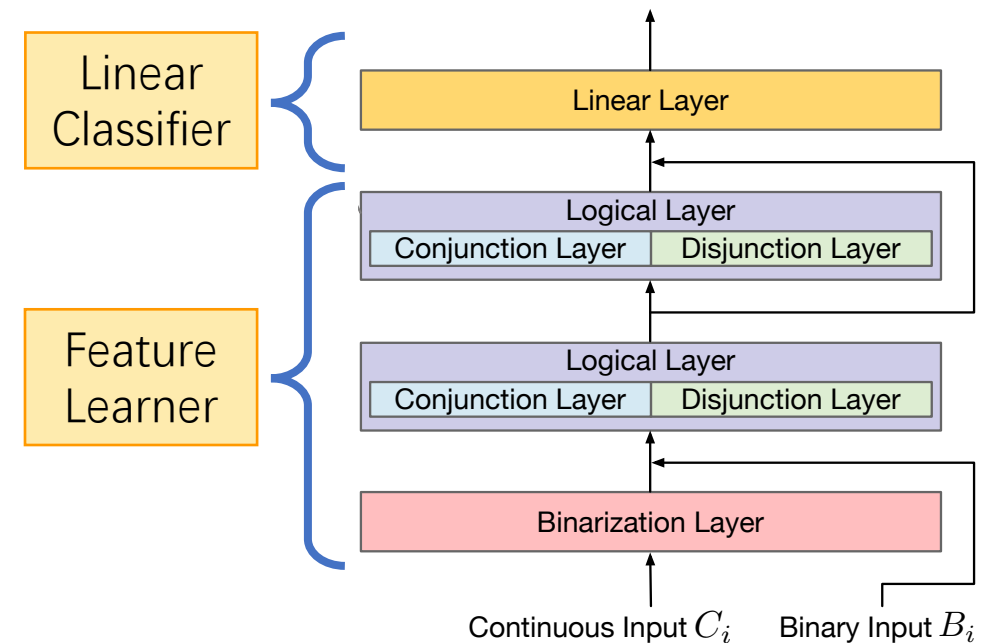


Figure: RRL can be simply considered as a feature learner and a linear classifier.



# Experiments

- We conduct experiments to evaluate the proposed model and answer the following questions:
  - How are the classification performance and model complexity of RRL?
  - How is the convergence of Gradient Grafting compared to other methods?
  - How is the scalability of the improved logical activation functions?

- Data set

- We took nine small and four large public datasets to conduct our experiments.
- Together they show the data diversity, ranging
  - from 178 to 102944 instances
  - from 2 to 26 classes
  - from 4 to 4714 original features

Table: Data sets properties.

Dataset	#instances	#classes	#features	feature type	density
adult	32561	2	14	mixed	-
bank-marketing	45211	2	16	mixed	-
banknote	1372	2	4	continuous	-
chess	28056	18	6	discrete	0.150
connect-4	67557	3	42	discrete	0.333
letRecog	20000	26	16	continuous	-
magic04	19020	2	10	continuous	-
tic-tac-toe	958	2	9	discrete	0.333
wine	178	3	13	continuous	-
activity	10299	6	561	continuous	-
dota2	102944	2	116	discrete	0.087
facebook	22470	4	4714	discrete	0.003
fashion	70000	4	784	continuous	-

# Classification Performance

- We compare the classification F1 score (Macro) of RRL with six interpretable models and five complex models.

Table: 5-fold cross validated F1 score (%) of comparing models on all 13 datasets.

Dataset	RRL	C4.5	CART	SBRL	CORESLS	CRS	LR	SVM	PLNN(MLP)	RF	LightGBM	XGBoost
adult	80.72	77.77	77.06	79.88	70.56	<b>80.95</b>	78.43	63.63	73.55	79.22	80.36	80.64
bank-marketing	<b>76.32*</b>	71.24	71.38	72.67	66.86	73.34	69.81	66.78	72.40	72.67	75.28	74.71
banknote	<b>100.00*</b>	98.45	97.85	94.44	98.49	94.93	98.82	<b>100.00</b>	<b>100.00</b>	99.40	99.48	99.55
chess	78.83	79.90	79.15	26.44	24.86	80.21	33.06	79.58	77.85	75.00	80.58	<b>80.66</b>
connect-4	<b>71.23*</b>	61.66	61.24	48.54	51.72	65.88	49.87	69.85	64.55	62.72	70.53	70.65
letRecog	96.15*	88.20	87.62	64.32	61.13	84.96	72.05	95.57	92.34	<b>96.59</b>	96.51	96.38
magic04	86.33*	82.44	81.20	82.52	77.37	80.87	75.72	79.43	83.07	86.48	86.67	<b>86.69</b>
tic-tac-toe	<b>100.00</b>	91.70	94.21	98.39	98.92	99.77	98.12	98.07	98.26	98.37	99.89	99.89
wine	98.23	95.48	94.39	95.84	97.43	97.78	95.16	96.05	76.07	98.31	<b>98.44</b>	97.78
activity	98.17	94.24	93.35	11.34	51.61	5.05	98.47	98.67	98.27	97.80	<b>99.41</b>	99.38
dota2	<b>60.12*</b>	52.08	51.91	34.83	46.21	56.31	59.34	57.76	59.46	57.39	58.81	58.53
facebook	<b>90.27*</b>	80.76	81.50	31.16	34.93	11.38	88.62	87.20	89.43	87.49	85.87	88.90
fashion	89.01*	80.49	79.61	47.38	38.06	66.92	84.53	84.46	89.36	88.35	<b>89.91</b>	89.82
<b>AvgRank</b>	2.77	8.23	8.92	9.31	9.92	7.08	7.92	6.77	5.77	5.38	2.77	<b>2.69</b>

# Classification Performance

- We compare the classification F1 score (Macro) of RRL with six interpretable models and five complex models.

Table: 5-fold cross validated F1 score (%) of comparing models on all 13 datasets.

Dataset	RRL	C4.5	CART	SBRL	CORESLS	CRS	LR	SVM	PLNN(MLP)	RF	LightGBM	XGBoost
adult	80.72	77.77	77.06	79.88	70.56	<b>80.95</b>	78.43	63.63	73.55	79.22	80.36	80.64
bank-marketing	<b>76.32*</b>	71.24	71.38	72.67	66.86	73.34	69.81	66.78	72.40	72.67	75.28	74.71
banknote	<b>100.00*</b>	98.45	97.85	94.44	98.49	94.93	98.82	<b>100.00</b>	<b>100.00</b>	99.40	99.48	99.55
chess	78.83	79.90	79.15	26.44	24.86	80.21	33.06	79.58	77.85	75.00	80.58	<b>80.66</b>
connect-4	<b>71.23*</b>	61.66	61.24	48.54	51.72	65.88	49.87	69.85	64.55	62.72	70.53	70.65
letRecog	96.15*	88.20	87.62	64.32	61.13	84.96	72.05	95.57	92.34	<b>96.59</b>	96.51	96.38
magic04	86.33*	82.44	81.20	82.52	77.37	80.87	75.72	79.43	83.07	86.48	86.67	<b>86.69</b>
tic-tac-toe	<b>100.00</b>	91.70	94.21	98.39	98.92	99.77	98.12	98.07	98.26	98.37	99.89	99.89
wine	98.23	95.48	94.39	95.84	97.43	97.78	95.16	96.05	76.07	98.31	<b>98.44</b>	97.78
activity	98.17	94.24	93.35	11.34	51.61	5.05	98.47	98.67	98.27	97.80	<b>99.41</b>	99.38
dota2	<b>60.12*</b>	52.08	51.91	34.83	46.21	56.31	59.34	57.76	59.46	57.39	58.81	58.53
facebook	<b>90.27*</b>	80.76	81.50	31.16	34.93	11.38	88.62	87.20	89.43	87.49	85.87	88.90
fashion	89.01*	80.49	79.61	47.38	38.06	66.92	84.53	84.46	89.36	88.35	<b>89.91</b>	89.82
<b>AvgRank</b>	<b>2.77</b>	8.23	8.92	9.31	9.92	7.08	7.92	6.77	5.77	5.38	2.77	<b>2.69</b>

RRL performs well on both small and large data sets.

# Classification Performance

- We compare the classification F1 score (Macro) of RRL with six interpretable models and five complex models.

Table: 5-fold cross validated F1 score (%) of comparing models on all 13 datasets.

Dataset	RRL	C4.5	CART	SBRL	CORESLS	CRS	LR	SVM	PLNN(MLP)	RF	LightGBM	XGBoost
adult	80.72	77.77	77.06	79.88	70.56	<b>80.95</b>	78.43	63.63	73.55	79.22	80.36	80.64
bank-marketing	<b>76.32*</b>	71.24	71.38	72.67	66.86	73.34	69.81	66.78	72.40	72.67	75.28	74.71
banknote	<b>100.00*</b>	98.45	97.85	94.44	98.49	94.93	98.82	<b>100.00</b>	<b>100.00</b>	99.40	99.48	99.55
chess	78.83	79.90	79.15	26.44	24.86	80.21	33.06	79.58	77.85	75.00	80.58	<b>80.66</b>
connect-4	<b>71.23*</b>	61.66	61.24	48.54	51.72	65.88	49.87	69.85	64.55	62.72	70.53	70.65
letRecog	96.15*	88.20	87.62	64.32	61.13	84.96	72.05	95.57	92.34	<b>96.59</b>	96.51	96.38
magic04	86.33*	82.44	81.20	82.52	77.37	80.87	75.72	79.43	83.07	86.48	86.67	<b>86.69</b>
tic-tac-toe	<b>100.00</b>	91.70	94.21	98.39	98.92	99.77	98.12	98.07	98.26	98.37	99.89	99.89
wine	98.23	95.48	94.39	95.84	97.43	97.78	95.16	96.05	76.07	98.31	<b>98.44</b>	97.78
activity	98.17	94.24	93.35	11.34	51.61	5.05	98.47	98.67	98.27	97.80	<b>99.41</b>	99.38
dota2	<b>60.12*</b>	52.08	51.91	34.83	46.21	56.31	59.34	57.76	59.46	57.39	58.81	58.53
facebook	<b>90.27*</b>	80.76	81.50	31.16	34.93	11.38	88.62	87.20	89.43	87.49	85.87	88.90
fashion	89.01*	80.49	79.61	47.38	38.06	66.92	84.53	84.46	89.36	88.35	<b>89.91</b>	89.82
<b>AvgRank</b>	2.77	8.23	8.92	9.31	9.92	7.08	7.92	6.77	5.77	5.38	2.77	<b>2.69</b>

RRL has a better performance than other interpretable models.



# Classification Performance

- We compare the classification F1 score (Macro) of RRL with six interpretable models and five complex models.

Table: 5-fold cross validated F1 score (%) of comparing models on all 13 datasets.

Dataset	RRL	C4.5	CART	SBRL	CORESLS	CRS	LR	SVM	PLNN(MLP)	RF	LightGBM	XGBoost
adult	80.72	77.77	77.06	79.88	70.56	<b>80.95</b>	78.43	63.63	73.55	79.22	80.36	80.64
bank-marketing	<b>76.32*</b>	71.24	71.38	72.67	66.86	73.34	69.81	66.78	72.40	72.67	75.28	74.71
banknote	<b>100.00*</b>	98.45	97.85	94.44	98.49	94.93	98.82	<b>100.00</b>	<b>100.00</b>	99.40	99.48	99.55
chess	78.83	79.90	79.15	26.44	24.86	80.21	33.06	79.58	77.85	75.00	80.58	<b>80.66</b>
connect-4	<b>71.23*</b>	61.66	61.24	48.54	51.72	65.88	49.87	69.85	64.55	62.72	70.53	70.65
letRecog	96.15*	88.20	87.62	64.32	61.13	84.96	72.05	95.57	92.34	<b>96.59</b>	96.51	96.38
magic04	86.33*	82.44	81.20	82.52	77.37	80.87	75.72	79.43	83.07	86.48	86.67	<b>86.69</b>
tic-tac-toe	<b>100.00</b>	91.70	94.21	98.39	98.92	99.77	98.12	98.07	98.26	98.37	99.89	99.89
wine	98.23	95.48	94.39	95.84	97.43	97.78	95.16	96.05	76.07	98.31	<b>98.44</b>	97.78
activity	98.17	94.24	93.35	11.34	51.61	5.05	98.47	98.67	98.27	97.80	<b>99.41</b>	99.38
dota2	<b>60.12*</b>	52.08	51.91	34.83	46.21	56.31	59.34	57.76	59.46	57.39	58.81	58.53
facebook	<b>90.27*</b>	80.76	81.50	31.16	34.93	11.38	88.62	87.20	89.43	87.49	85.87	88.90
fashion	89.01*	80.49	79.61	47.38	38.06	66.92	84.53	84.46	89.36	88.35	<b>89.91</b>	89.82
<b>AvgRank</b>	2.77	8.23	8.92	9.31	9.92	7.08	7.92	6.77	5.77	5.38	2.77	<b>2.69</b>

Only two complex models that use hundreds of estimators have comparable results with RRL.

# Model Complexity

- Interpretable models seek to keep low model complexity while ensuring high accuracy.
- To show the relationships between classification performance and model complexity of RRL and baselines, we draw scatter plots of F1 score against  $\log(\#\text{edges})$  for rule-based models.

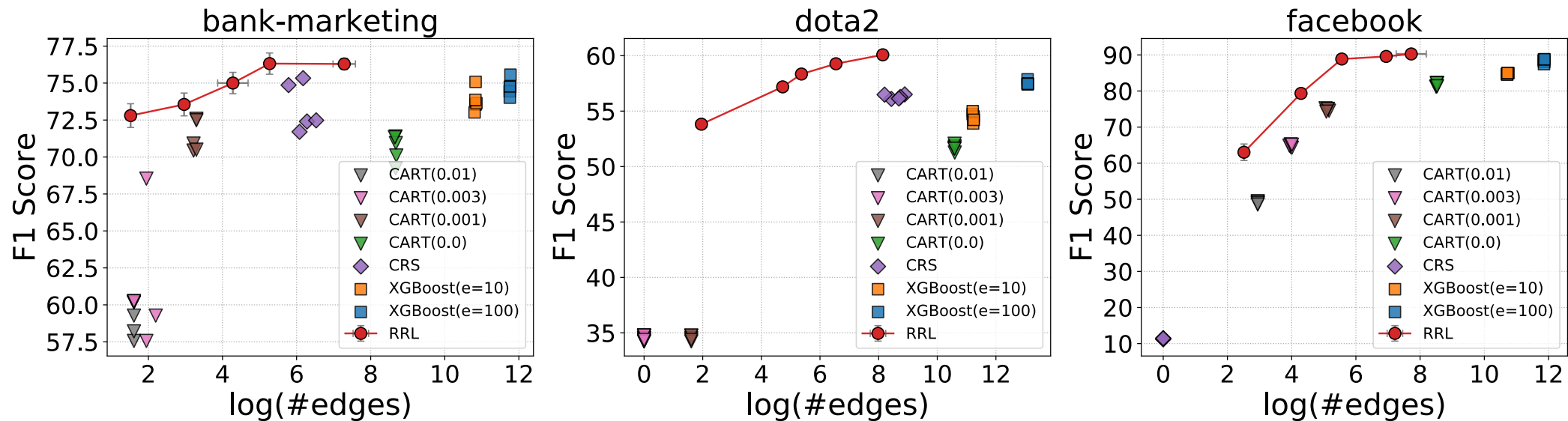


Figure: Scatter plot of F1 score against  $\log(\#\text{edges})$  for RRL and baselines on three datasets

# Ablation Study

- Training Method for Discrete Model
  - We compare Gradient Grafting with other representative gradient-based discrete model training methods, i.e., STE, ProxQuant, and RB, by training RRL with the same structure.
- Improved Logical Activation Functions
  - We also compare RRL with or without improved logical activation functions.

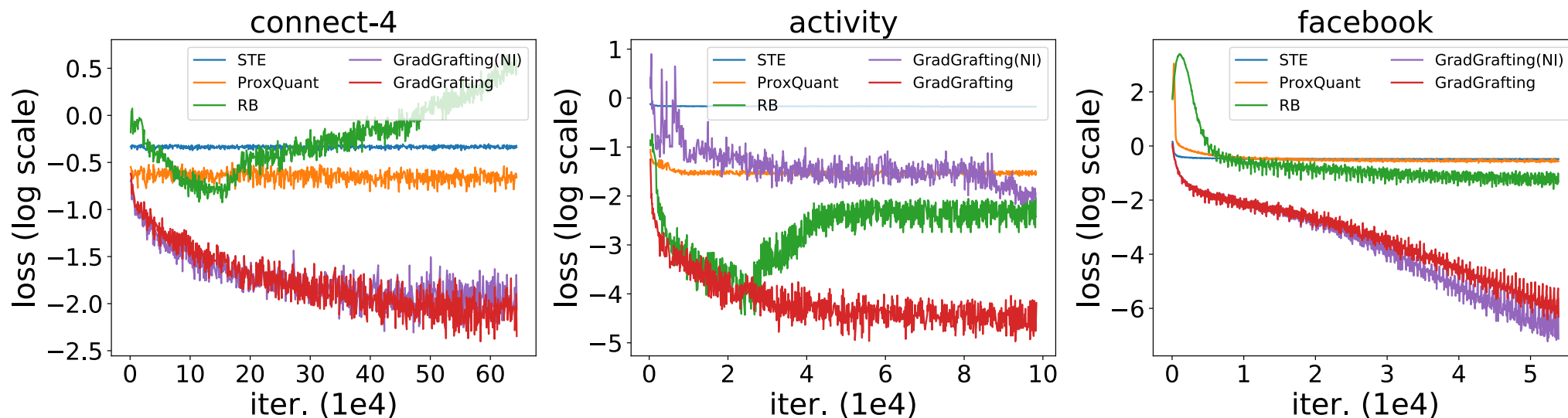


Figure: Training loss of three compared discrete model training methods and Gradient Grafting with or without improved logical activation functions on three data sets.

# Case Study

- Weight Distribution
  - We show the distribution of weights in the linear layer of the RRLs trained on the *bank-marketing* data set.
- Learned Rules
  - The learned rules, with high weights, of one RRL trained on the bank-marketing data set.
  - Provide intuition for image classification tasks by visualizations of learned rules.

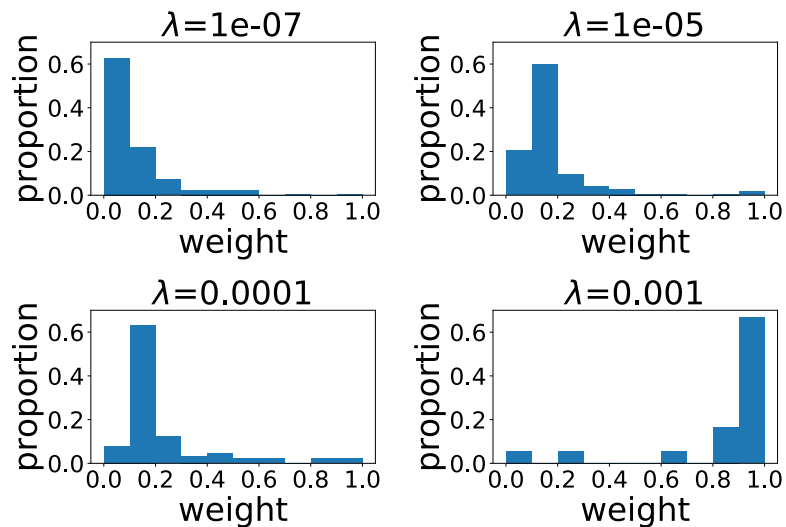


Figure: The distribution of weights in the linear layer of RRLs with the same model structure but different  $\lambda$ .

Weight	Rule
0.995	$-122.5 < \text{balance} < 2606.1 \wedge \text{marital} = \text{married} \wedge \text{campaign} < 5 \wedge \text{poutcome} = \text{success} \wedge \text{previous} > 0$
0.753	$1757.2 < \text{balance} < 7016.7 \wedge \text{marital} = \text{married} \wedge \text{contact} = \text{telephone} \wedge 6 < \text{day} < 27 \wedge \text{previous} < 5$
0.733	$\text{age} > 36 \wedge \text{balance} < 7016.7 \wedge \text{marital} = \text{married} \wedge \text{campaign} < 5 \wedge \text{pdays} < 104 \wedge \text{poutcome} = \text{success}$
0.731	$36 < \text{age} < 60 \wedge \text{balance} > -122.5 \wedge \text{campaign} < 7 \wedge \text{day} > 22 \wedge \text{pdays} > 304 \wedge \text{previous} > 0$
0.728	$\text{age} > 28 \wedge -669.1 < \text{balance} < 5813.7 \wedge \text{campaign} < 6 \wedge \text{pdays} > 304 \wedge 0 < \text{previous} < 6$

Figure: Logical rules obtained from RRL trained on the *bank-marketing* data set.

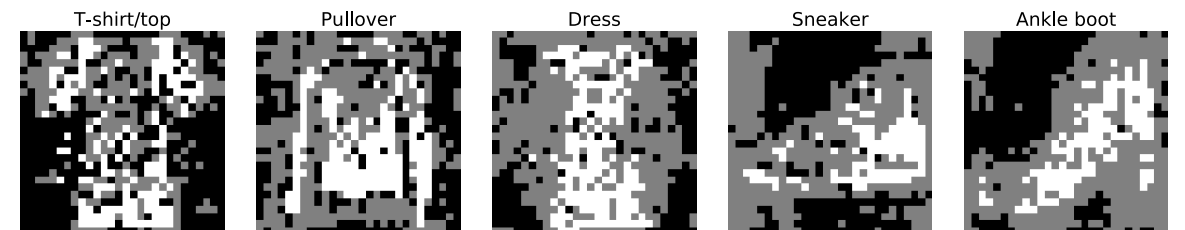


Figure: Decision mode for the *fashion* data set summarized from rules of RRL.



# Conclusion & Future Work

---

- We propose a new scalable classifier, named Rule-based Representation Learner (RRL), that can automatically learn interpretable rules for data representation and classification.
- For the particularity of RRL, we propose a new gradient-based discrete model training method, i.e., Gradient Grafting, that directly optimizes the discrete model.
- We also propose an improved design of logical activation functions to increase the scalability of RRL and make RRL capable of discretizing the continuous features end-to-end.
- Our experimental results show that RRL enjoys both high classification performance and low model complexity on data sets with different scales.
- For the current design of RRL is limited to structured data, we will extend RRL to suit more unstructured data as future work.

# Thanks for your Listening!

E-mail: wang-z18@mails.tsinghua.edu.cn

Code: <https://github.com/12wang3/rrl>