Local Optimal Trees Explaining Deep Models

XuehuaiShi[1] Yong Qi[2] Qianmu Li[3] Weibin Zhang[4]

Abstract

There are many reasons emphasize the importance of interpretability of a model while the interpretability and the role of human-simulation are oft-overlooked in deep models. Based on the brilliant achievements of previous researchers, we proposed a deep model explanation using local optimal decision trees. Specifically, we first cluster the data to get clusters by the input number of clusters, which is, similar instances should have similar explanations that is complied with local fidelity. We build deep models to fit the training set, using different decision trees to fit the deep model respectively via the result of a clustering algorithm. Inspired by reinforcement learning, we explore the parameters in the deep model with the target of smaller APL. Finally, we will get a deep model with a small APL without sacrificing predictive power.

Keywords: Deep Model Explanation, Tree Regularization, Local Optimal Tree

Introduction

Deep models have achieved great success in a variety of applications such as computer version (e.g. (Kalal Z, Mikolajczyk K, Matas J. 2012; Gall J, Yao A, Razavi N, et al. 2011)) and NLP (e.g. (Athiwaratkun B, Wilson A G. 2017)). However, the interpretability and the role of human-simulation are oft-overlooked in the field. There are many reasons emphasize the importance of interpretability of a model: (1) In many fields, such as medical diagnosis (e.g. (Das S, Guha D, Dutta B. 2016)) or risk evaluation (e.g. (Hao Z, Xu Z, Zhao H, et al. 2017)) etc., humans need to know the faith behind the model's behavior as the blind knowledge may cause catastrophic results; (2) Real-world data is always different between toy data or training data, only a human-simulatable model has the foundation that users can take actions or fine tuning based on it.

Creating an accurate yet interpretable deep model is an important and paramount question in many domains. To address the question, Ramprasaath R. Selvaraju (Selvaraju R, Cogswell M, Das A, et al 2016) produced a coarse localization map that gave the deep models visual explanations in an image classification task, and Hao Li & Zheng Xu et al.(Li H, Xu Z, Taylor G, et al. 2017) explored how network architecture/training parameters affect the loss landscape of minimizers to understand deep models, but in structured data processing tasks, these approaches would show less capability to promote the interpretability of a deep model. Thus Marco Tulio Ribeiro & Sameer Singh et al. (Ribeiro M T, Singh S, Guestrin C. 2016) proposed a method called LIME to explain any classifier, as the method gave lists of important features to explain the predictions of a classifier which is

not sufficient enough. Mike Wu et al. (Wu M, Hughes M C, Parbhoo S, et al. 2017) simulated deep models by trees as decision trees with a few nodes are easy for humans to simulate and thus understand and trust.

Based on the brilliant achievements of previous researchers, we try to approximate the decision boundary of deep models using decision trees (Breiman L. et al. 2017.) based on local fidelity. As we know the decision tree is more in line with the logic of human thinking, in this paper, we proposed a deep model explanation using local optimal decision trees. Specifically, we first cluster the data to get clusters by the input number of clusters, which is, similar instances should have similar explanations that is complied with local fidelity, and it is often impossible for an explanation to be completely faithful (Ribeiro M T, Singh S, Guestrin C. 2016). We build deep models to fit the training set, and then we use different decision trees to fit the deep model respectively via the result of a clustering algorithm. Then the problem is that can we form the association between the deep model and the APL (average decision path length of decision trees). Inspired by reinforcement learning, we assume each training loop of the deep model is an action, the action will give the mission a state, as the state has a direct relation with APL which gives the action a reward to behave itself. Finally, we will get an accurate deep model with a small APL.

The rest of this paper is organized as follows. We start with a brief review of related work. Then we formulate the problem and present the proposed approach. Experimental results are reported, followed by the conclusion of this work.

Related Work

Recurrent Neural Networks with Gated Recurrent Units.

A recurrent neural network (RNN) is produced by Chung J et al. (Chung J, Gulcehre C, Cho K H, et al. 2014), it is a new variant of LSTM (Long Short-Term Memory) (Hochreiter S, Schmidhuber J. 1997). It takes an arbitrary length sequence $x_n = [x_{t_1} \dots x_{t_n}]$ as the input, generating a hidden state sequence $h_n = [h_{t_1} \dots h_{t_n}]$ as the input in each layer of RNN. Formula 1 describes the details of a gated recurrent unit in each layer, it produces $\mathrm{h}_{nt_n} = f(\mathrm{x}_{nt_n}, \mathrm{h}_{nt_{n-1}})$ as the output for finding the nonlinear relation between the arbitrary length sequences x_n and h_n.

$$
\begin{aligned}
\text{output state:} \quad & h_{t_\mathrm{n}} = \left(1 - z_{t_\mathrm{n}}\right) h_{t_{\mathrm{n}-1}} + z_{t_\mathrm{n}} \widetilde{h_{t_\mathrm{n}}} \\
\text{candidate state:} \quad & \widetilde{h_{t_\mathrm{n}}} = \tanh(V_{t_\mathrm{n}}^{h} x_{t_\mathrm{n}} + U_{t_\mathrm{n}}^{h}(r_{t_\mathrm{n}} \odot h_{t_{\mathrm{n}-1}})) \\
\text{update gate:} \quad & z_{t_\mathrm{n}} = \sigma(V_{t_\mathrm{n}}^{z} x_{t_\mathrm{n}} + U_{t_\mathrm{n}}^{z} h_{t_{\mathrm{n}-1}}) \\
\text{reset gate:} \quad & r_{t_\mathrm{n}} = \sigma(V_{t_\mathrm{n}}^{r} x_{t_\mathrm{n}} + U_{t_\mathrm{n}}^{r} h_{t_{\mathrm{n}-1}})
\end{aligned}
\tag{1}
$$

The prediction of a single GRU sequence is produced as follows. As described by Formula 1, the transition function f produces h_{t_n} from the previous state $h_{t_{\mathrm{n}-1}}$ and the current input x_{t_n}. There are several internal nodes in a layer of the gated recurrent unit: candidate state gates $\tilde{\mathrm{h}}$, update gates z and reset gates r, they all have the same

cardinality as h. Reset gates r are used for forgetting past state vectors via the logistic sigmoid nonlinearity $\sigma()$. Update gates z are used for either updating the state vector. The structure of the gated recurrent unit is diagrammed in Fig. 1. The prediction formulation is detailed in Formula 2, y_{t_n} is the prediction at time t_n:

$$\hat{y}_{t_n} = \sigma(W^T h_{t_n}) \quad (2)$$

$W \in R^K$ are the parameters of the output layer. The entire parameters of the GRU-RNN are described as $W = (w, U, V)$. The loss function of the model is detailed in Formula 3:

$$min_W \lambda \Psi(W) + \sum_{n=1}^{N} \sum_{t=t_1}^{T_n} loss(y_{nt}, \hat{y}_{nt_n}(x_n, W)) \quad (3)$$

Where $\Psi(W)$ represents the regularization cost of the parameters W.

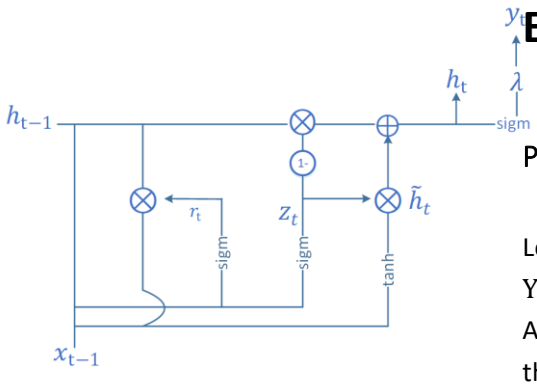


Figure 1: Diagram of gated recurrent unit (GRU). Used for each time step our neural time-series model.

Tree Regularization for Deep Models.

The tree regularization function $\Omega(\mathrm{W})$ (Wu M, Hughes M C, Parbhoo S, et al. 2017) is defined to penalize the models whose predictions are not simulatable. It has two stages: first, find a decision tree which can fit deep model's predictions $\hat{y}_n$ well with given input x_n. Second, measure the complexity of decision tree as $\Omega(\mathrm{W})$. Alg. 1 defines the cost function $\Omega(\mathrm{W})$.

The other contribution of the Tree Regularization is introducing and training a surrogate model to fit the regularization function $\Omega(\mathrm{W})$ with the parameters of the deep model we try to simulate.

Algorithm 1 Average-Path-Length Cost Function $\Omega(\mathrm{W})$

Require:

$\hat{y}$(.,W): binary prediction function, with parameters W

D = $\{x_n\}_{n=1}^N$: dataset with N examples

1: **Function** $\Omega(\mathrm{W})$

2: tree $\leftarrow$ TrainTree($\{x_n, \hat{y}(x_n, \mathrm{W})\}$)

3: **return** $\frac{1}{N}\sum_n PathLength(tree, x_n)$

Local Optimal Tree Explaining Deep Models

Problem Formulation

Let $\mathrm{X} = \mathrm{R}^{t \times n}$ denote the input space and $\mathrm{Y} = \mathrm{R}^n$ be the n-dimensional output space. Assuming $\mathcal{D} = \{(x_i, y_i) | 1 \leq \mathrm{i} \leq \mathrm{m}\}$ denote the training set, x_i is the feature tensor for i-th instance and y_i is the label vector. And we get a deep model's prediction function $\hat{y}(x_i, \mathrm{W})$ with parameters W. The problem is defined as:

Definition 1. *Given the training set $\mathcal{D}$, the deep model's prediction function $\hat{y}(x_i, W)$, the task is to get the decision process with the decision tree from the input space to the deep model's prediction.*

$$\mathbb{f} : \text{decision process} \leftarrow \mathbb{k}(x_i, \hat{y}(x_i, \mathrm{W})) \quad (4)$$

For an instance $x \in X$, the function $\mathbb{f}$ gives the decision process of the deep model's

prediction.

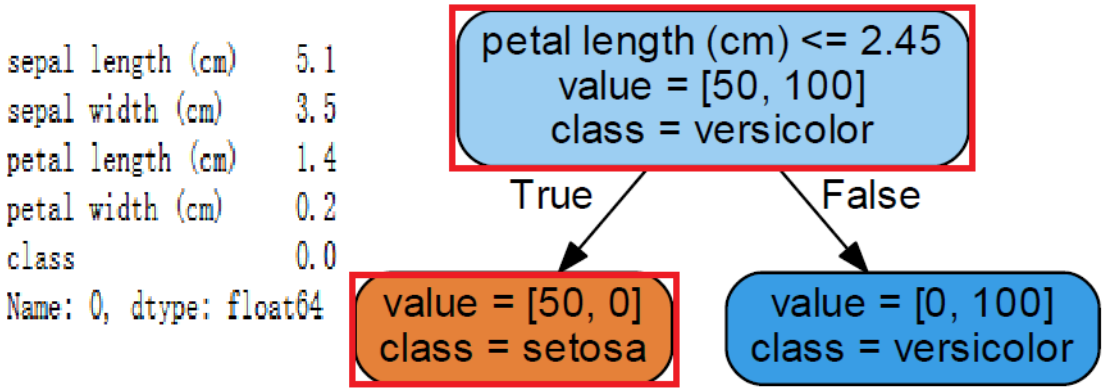


Figure 2: Explaining a single prediction of a GRU classifier trying to classify the type of the iris. The red frames in the decision tree on the right side of the figure give the decision-making process for the example on the left side.

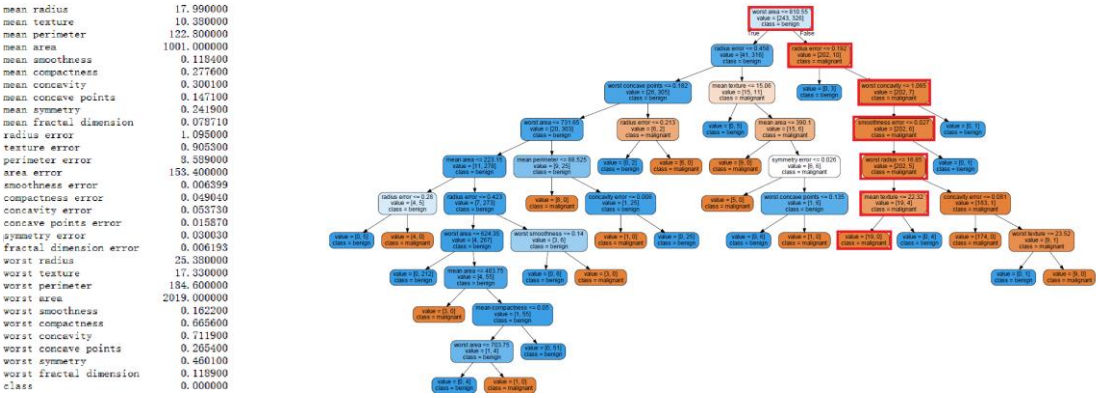
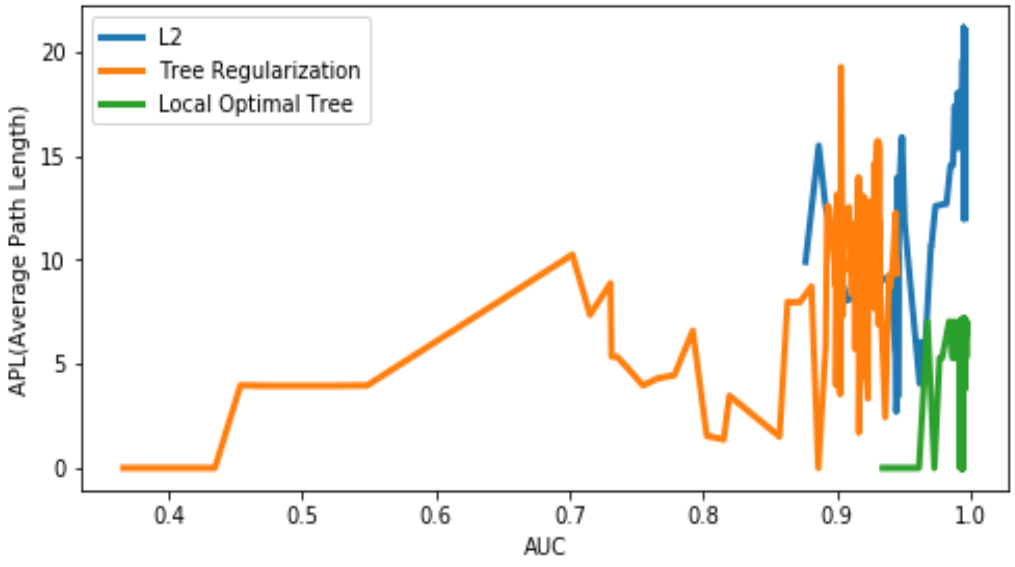
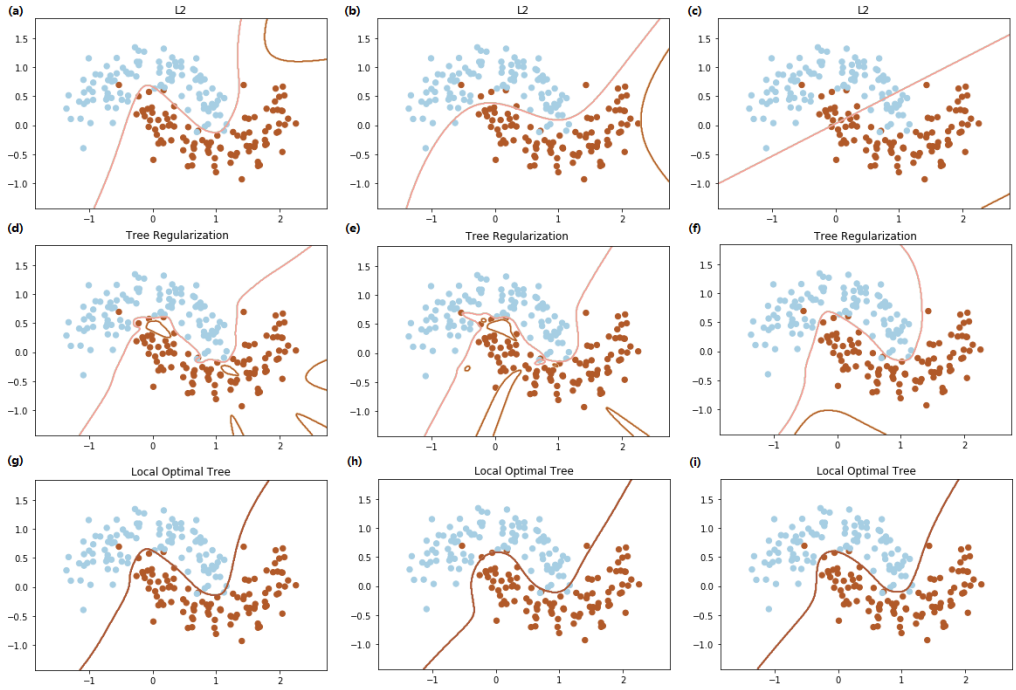


Figure 3: The other example of our target, explaining a single prediction of a GRU classifier trying to classify the type of the breast cancer example is benign or malignant. The red frames in the decision tree on the right side of the figure give the decision-making process for the example on the left side.



(1) Prediction Quality and Complexity as AUC vs APL



(2) Decision Boundaries with Respective Approaches

Figure 4: *2D Making-moons task:* (1) Each approach's AUC and APL matrics, with λ in L2 is 300, strength in Tree Regularization is 1000 and strength in Local Optimal Tree is 0.4 (strength in Local Optimal Tree ranges between 0 and 1, the smaller strength would get stronger shift for reducing APL). In the Fig. 4(1), our proposed approach (Local Optimal Tree) produces models with smaller APL than L2 and Tree Regularization, and Tree Regularization produces lower AUC than L2 and the proposed approach while its APL regime between 5 and 10. (2) Decision boundaries have qualitatively different shapes for approaches. (a-c): Decision boundaries with L2, each λ in (a-c) is 0.1, 10 and 300. (d-f): Decision boundaries with Tree Regularization, each $\mathit{strength}$ in (d-f) is 0.1, 10 and 1000. (g-i): Decision boundaries with Local Optimal Tree, each $\mathit{strength}$ in (d-f) is 0.9, 0.7 and 0.4. In the Fig. 4(2), the latter two approaches are more robust than L2.

The Proposed Approach

In the original tree regularization for deep models, we are given a training data set $\mathcal{D} = \{x_n, y_n\}_{n=1}^{N}$, the algorithm would get the cost function $\Omega(\mathrm{W})$ to punish the model whose average decision path length is too complex. We make some enhancements to the tree regularization.

First, we will train a deep model with parameters W and training set $\mathcal{D}$ as usual. The next step is to get the explanation for the deep model. As we know, there are several criterions for explanations, including interpretable and local fidelity. The ideal explanation is completely faithful, but it is always difficult to achieve. To achieve the local fidelity, we first cluster the dataset $\mathcal{D}$ to get clusters by the input number of clusters c, we use c different decision trees to simulate the deep model's prediction boundary. Next, we measure the complexity of all these c trees as the output of $\mathrm{APL}(\mathrm{W}, \mathcal{D})$ just like the previous tree regularization. We also measure the complexity of the trees as the APL (average decision path length) --- the average number of edges that would be passed to make a prediction. The calculation of $\mathrm{APL}(\mathrm{W}, \mathcal{D})$ is detailed in Alg. 2. It requires three

subroutines, $GetClusters, TrainTree,$ and $PathLength$. $GetClusters$ splits the dataset $\mathcal{D}$ into c clusters, in this paper, we use K-means algorithm. $TrainTree$ trains c different decision trees to simulate the deep model's prediction boundary with different clusters. $PathLength$ calculates the number of edges for a specific input to the output of the deep model.

Algorithm 2 Complexity of Decision Trees $\mathcal{APL}(\mathrm{W}, \mathcal{D})$

Require:

c: the input number to cluster dataset $\mathcal{D}$ into c clusters

$f(.,\mathrm{W})$: deep model's prediction function, with parameters W

$\mathcal{D} = \{x_n\}_{n=1}^{N}$: dataset with N examples

1: **Function** $\mathrm{APL}(\mathrm{W})$

2: Trees = []

3: $\{\mathcal{D}_1, \ldots, \mathcal{D}_c\} \leftarrow GetClusters(\mathcal{D})$

4: for i in range(c):

5: $tree_i \leftarrow TrainTree(\{\mathcal{D}_i, f(\mathcal{D}_i, \mathrm{W})\})$

6: Trees. Append($tree_i$)

7: **return** $\frac{1}{N}\sum_n PathLength(\mathrm{Trees}, \mathcal{D})$

Accuracy-Interpretability Trade-off

We define the deep model as $m \in \mathcal{M}$, where $\mathcal{M}$ is the model space. Let $\Omega(m)$ denote the complexity of the model m, $\ell(m, \mathcal{D}_i)$ denote the measure of how unfaithful m is in approximating $\mathcal{D}_i$ in the local fidelity. The bigger $\Omega(m)$ is, the model m is more complex; the smaller $\ell(m, \mathcal{D}_i)$ is, the model m is more accurate. Finally, we want to strike a balance between the complexity and accuracy of a model m, the target function produced by the proposed approach is obtained by following:

$$\xi(f, \mathcal{D}) = \min_{f \in \mathcal{F}} \{\ell(m, \mathcal{D}_i) + \Omega(m)\} \quad (5)$$

In this paper, we use GRUs as the deep model example we want to explain. The other key contribution of our work is introducing and training the GRUs with its APL. Inspired by the knowledge of enforcement learning, we get the target by following. Assuming each epoch of the GRUs training is an action α, after the action, GRUs would achieve better performance on training set $\mathcal{D}$ ($\ell(m, \mathcal{D}_i)$ become smaller) and maybe the GRUs becomes more complex ($\Omega(m)$ become bigger), we define it as a new state s. Then the GRUs would get a reward r by the target function $\xi(m, \mathcal{D})$, which can shift the parameters of GRUs by the function $shift(r, s, \mathrm{W})$. Unlike enforcement learning, the proposed approach only has one action --- training the GRUs to get better performance on the training set, we just shift the parameters of GRUs to get smaller $\Omega(m)$ without sacrificing its fidelity too much during each epoch. Our true approach to explaining deep models is detailed in Alg. 3.

Alg. 3 requires four subroutines: $m(., \mathrm{W})$, $\mathcal{APL}(\mathrm{W}, \mathcal{D})$, $\xi(m, \mathcal{D})$ and $shift(r, s_i, \mathrm{W})$. $m(., \mathrm{W})$ is the deep model's prediction function. $\mathcal{APL}(\mathrm{W}, \mathcal{D})$ gets the complexity of decision tree with parameters W of the deep model, it would be used for the target function that can shift the parameters W of deep model $\xi(m, \mathcal{D})$. $shift(r, s_i, \mathrm{W}_{i-1})$ is function that shift W as the feedback of the reward r. In this paper, we propose an MLP (multi-layer perception)

approximate the relationship between W and $\mathcal{APL}(\mathrm{W},\mathcal{D})$, shifting the parameters W to reduce $\xi(m,\mathcal{D})$ with the input number $strength$.

Algorithm 3 Local Optimal Tree Explaining Deep Models: $k(x_i, y_i)$

Require:

m(., W): deep model's prediction function, with parameters W

$\mathcal{APL}(\mathrm{W},\mathcal{D})$: get the complexity of decision tree with parameters W of the deep model

$\xi(m,\mathcal{D}) = \min_{f\in\mathcal{F}}\{\ell(m(.,\mathrm{W}),\mathcal{D}) + \mathcal{APL}(\mathrm{W})\}$: the target function that used for parameters W shifting

$shift(r, s_i)$: the function to shift original parameters of the deep model into new parameters

$strength$: the input number measure the strength to reduce $\xi(m,\mathcal{D})$

$\mathcal{D} = \{x_n\}_{n=1}^{N}$: dataset with N examples

1. **Function** $k(x_i, y_i)$:
2. Initialize: State:s_0, W_0
3. While no significant advancement in GRUs:
4. Action: $a_i \leftarrow m(\mathcal{D}, \mathrm{W}_{i-1})$
5. State: $s_i \leftarrow s_{i-1} \times a_i$
6. Reward: $r \longleftarrow \xi(\hat{y}, \mathcal{D})$
7. $\mathrm{W}_i \leftarrow shift(r, s_i, \mathrm{W}_{i-1}) \times strength$

Experiments

Tasks

Synthetic Task: Signal-and-noise HMM

We test the proposed approach in a toy dataset, which has $N = 100$ sequences with $T = 20$ time steps, and each time step has a data vector of 14 binary features x_{nt} and 1 output label y_{nt}. The data comes from two separate HMM processes. First, a "signal" HMM generates the first 7 data dimensions from 5 well-separated states. Second, an independent "noise" HMM generates the remaining 7 data dimensions from a different set of 5 states. Each timestep's output label y_{nt} is produced by a rule involving both the signal data and the signal hidden state: the target is 1 at the timestep t only if both the first signal state is active and the first observation is turned on. We deliberately designed the generation process so that neither logistic regression with x as features nor an RNN model that makes predictions from hidden states alone can perfectly separate this data.

Real-World Tasks

- **Wafer:** The series was produced by R. Olszewski (Olszewski R T.2001). The two labels of the dataset are normal and abnormal, representing the wafer is qualified or not. The dataset was collected by a series of sensors with a series of tools during the process of one wafer. A collection of inline process control measurement details was produced by the database.
- **Yoga:** This series was produced by Wei L (Wei L, Keogh E. 2006). Each record in the dataset was converted from the image that captured the yoga pose of the actor, the record was one-dimensional. There are two actors posing the yoga poses in

the dataset, as the two classes of the dataset were male and female, and our target is distinguishing the classes by each one-dimensional series in the dataset.

- **PhalangesOutlinesCorrect:** This dataset was formatted by Luke Davis (Davis L M. 2013), which was converted from the record by Cao F (Cao F et al. 2000). The dataset extract the hand outlines and the outlines of three bones of the middle finger. At first, their target for converting the image to the one-dimensional series data are bone outline detection and bone age prediction.

Results

The experiments comparing different approaches with GRU are detailed below. Beside the parameters of respective approaches, the hyper-parameters of the GRUs are the same.

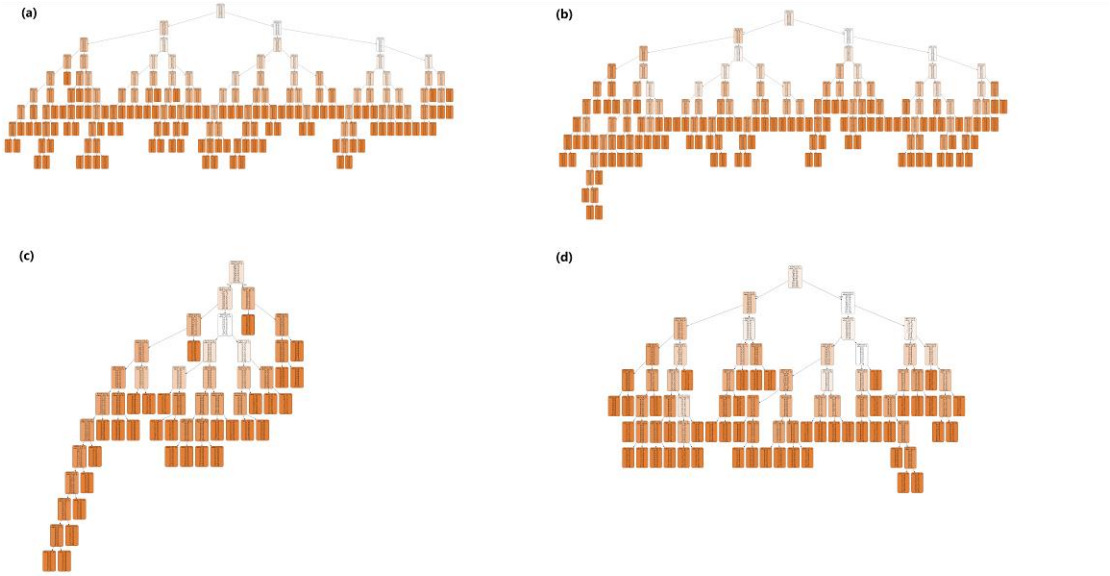


Figure 5: *Signal-and-Noise HMM Task*: (a) GRU trained with L2 objective, (b) GRU trained with tree regularization, (c)-(d) GRU trained with our proposed approach with 2 clusters which gave us two decision trees for different clusters. Obviously, (c)-(d) can mimic the prediction of the GRU model with the lower APL (average decision path length).

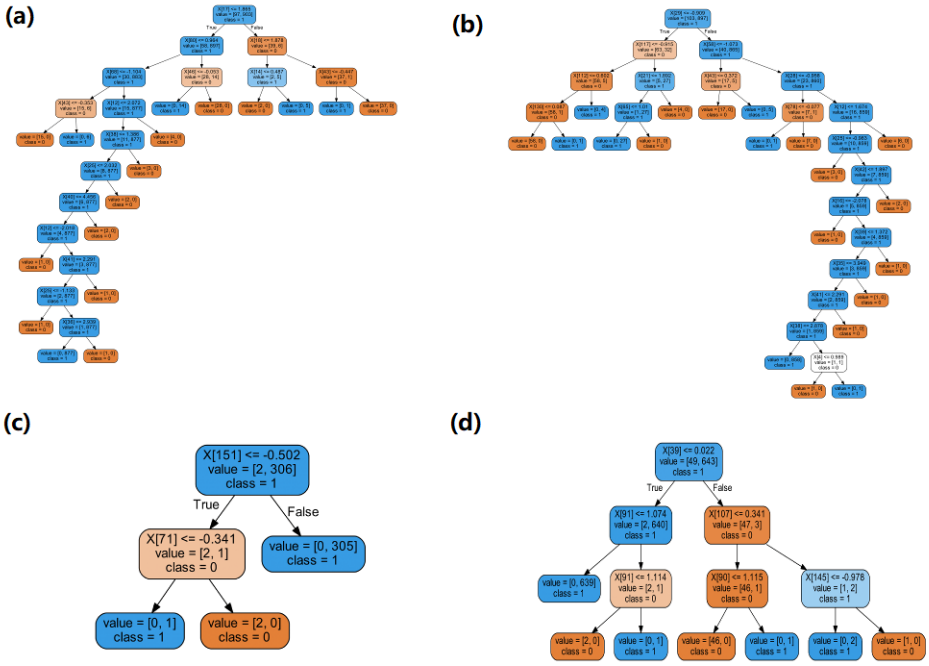


Figure 6: *Wafer Task*: (a) GRU trained with L2 objective, (b) GRU trained with tree regularization, (c)-(d) GRU trained with our proposed approach with 2 clusters which gave us two decision trees for different clusters. The figure shows that (c)-(d) can mimic the prediction of the GRU model with the lower APL as well.

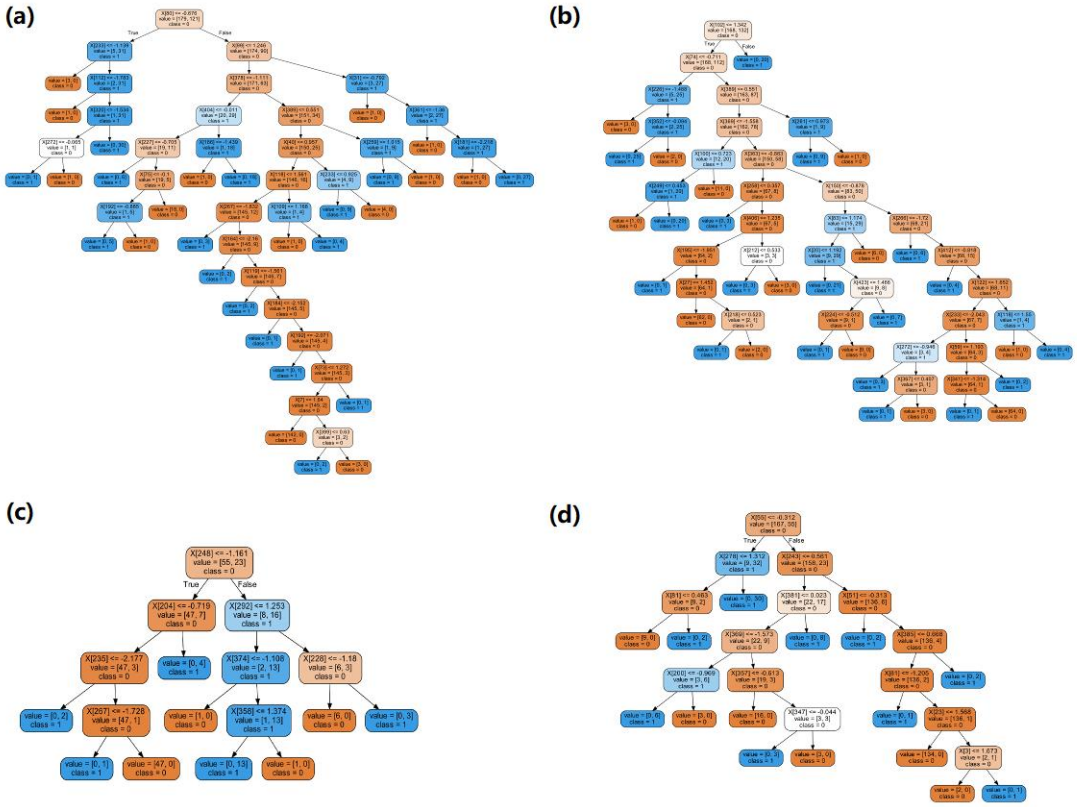


Figure 7: *Yoga Task*: (a) GRU trained with L2 objective, (b) GRU trained with tree regularization, (c)-(d) GRU trained with our proposed approach with 2 clusters which gave us two decision trees for different clusters. The figure shows that (c)-(d) can mimic the prediction of the GRU model with the lower APL.

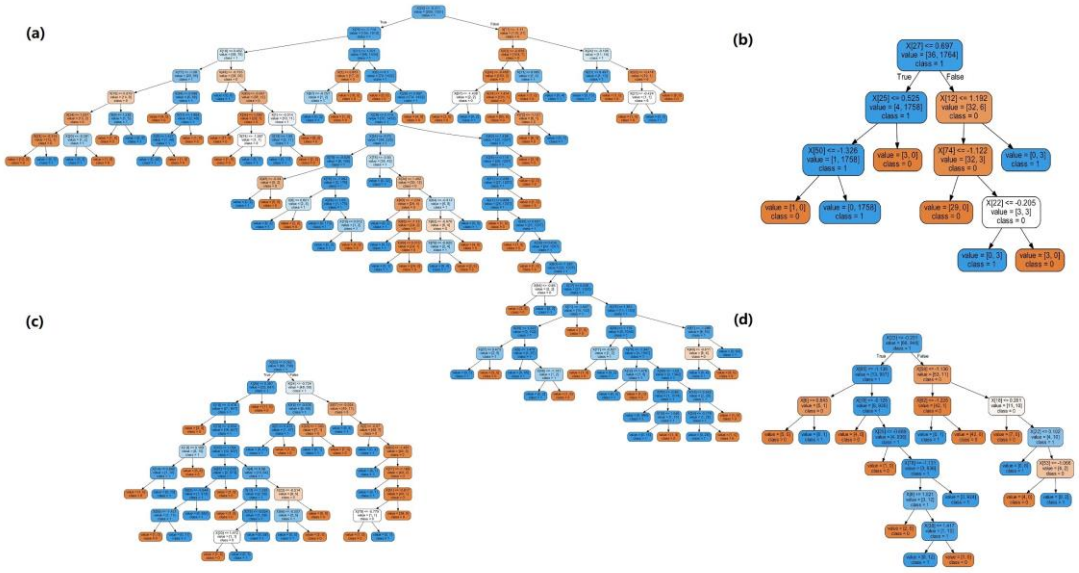


Figure 8: *PhalangesOutlinesCorrect Task*: (a) GRU trained with L2 objective, (b) GRU trained with tree regularization, (c)-(d) GRU trained with our proposed approach with 2 clusters which gave us two decision trees for different clusters. The figure shows that (c)-(d) can mimic the prediction of the GRU model with the lower APL.

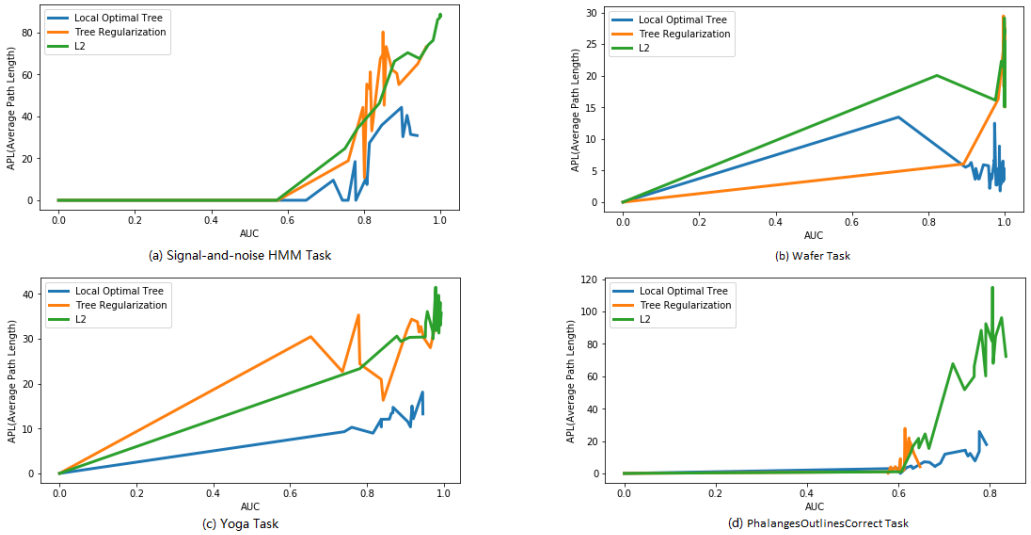


Figure 9: Fitness curves for the GRU-L2, GRU-TreeRegularization, and GRU-LocalOptimalTree(our proposed approach). Each line represents the prediction quality (AUC) vs. complexity (APL). Besides the parameters of respective approach, the parameters of the GRU models are the same.

Dataset	Fidelity
Signal-and-noise HMM	0.88
Wafer	0.91
Yoga	0.88
PhalangesOutlinesCorrect	0.80
BirdChicken	0.81
Gun-Point	0.90

Table 1: Fidelity of the mimic result of the

decision trees to the predictions of our trained GRU recurrent neural networks. Fidelity is defined as the percentage of test examples on which the prediction made by tree agree with the deep model (Craven and Shavilk 1996).

Fewer nodes than tree-regularized models. Across four tasks, GRU with the Local Optimal Tree generates smaller interpretable decision trees that can mimic the predictions of GRU. In Fig. 8(a), the Local Optimal Tree achieves lower APL with the same quality while its fitness has a little lower performance than the GRUs with tree regularization or simple L2, but it can reproduce much smaller interpretable trees (much lower APL) in the signal-and-noise HMM task. In the Wafer task, all three approaches can fit the dataset well, the proposed approach can get better APL while having the same quality in Fig. 8(b). In the Yoga task, our proposed approach has the familiar result as the Wafer task, a better APL with the same quality, as shown in Fig 8(c). In the PhalangesOutlinesCorrect task, our proposed approach and simple L2 can fit the dataset better than the tree regularization while the proposed approach can get the better trees.

Less running times that tree-regularized models. Our proposed approach takes 4365 seconds per epoch on the PhalangesOutlinesCorrect task (exampling the longest training time) while the GRU with tree regularization takes 3986 seconds per epoch. Our proposed approach fit the dataset with less epoch---20 while the GRU with tree regularization takes 28 epochs. Because the weights *W* in the deep models with tree regularization have the indirect relation with APL and the *W* may not be as relevant in characterizing the current decision function of GRU. As our proposed approach fix it with the hyper-parameter $\mathit{strength}$ for preventing the *W* shift away.

More robust and less parameter-tuning than tree-regularized models. The proposed approach has two hyper-parameters c and $\mathit{strength}$, c is the number of clusters for the dataset, $\mathit{strength}$ is the parameter that defines the strength of weights in deep models shifting for reducing APL during each epoch, the default $c = 0$, and $\mathit{strength} = 0.8$. A better c would nearly average the dataset. The GRU with Local Optimal Tree would perform well with the default c and $\mathit{strength}$. In the deep models with tree regularization, the weights *W* in the deep models have the indirect relation with APL and the *W* may not be as relevant in characterizing the current decision function of GRU, our proposed approach fix the first problem by tuning the weights of deep models directly and fix the second problem with the fixed window for shifting *W* in Alg. 3.

Conclusion

We have introduced a novel technique for approximating the decision boundary of deep models using decision trees based on local fidelity. Specifically, we first cluster the data to get clusters by the input number c of clusters, which is, similar instances should have similar explanations that is complied with local fidelity, and it is often impossible for an explanation to be completely faithful (Ribeiro M T et al. 2016). We build deep models to fit the training set, then we use different decision trees to fit the deep model respectively via the result of a clustering algorithm. Inspired by reinforcement learning, we assume each training loop of the deep model is an action, the action will give the mission a state, as the state has a direct relation with APL which gives the action a reward to behave itself. We introduced the Alg. 3 for getting an accurate

deep model with a small APL. The results of experiments show it performs well with our proposed approach. Our future work could continue to explore and increase the stability of the explanation to the learned deep models.

Reference

Kalal Z, Mikolajczyk K, Matas J. Tracking-learning-detection[J]. IEEE transactions on pattern analysis and machine intelligence, 2012, 34(7): 1409-1422.

Gall J, Yao A, Razavi N, et al. Hough forests for object detection, tracking, and action recognition[J]. IEEE transactions on pattern analysis and machine intelligence, 2011, 33(11): 2188-2202.

Athiwaratkun B, Wilson A G. Multimodal word distributions[J]. arXiv preprint arXiv:1704.08424, 2017.

Das S, Guha D, Dutta B. Medical diagnosis with the aid of using fuzzy logic and intuitionistic fuzzy logic[J]. Applied Intelligence, 2016, 45(3):1-18.

Hao Z, Xu Z, Zhao H, et al. Probabilistic dual hesitant fuzzy set and its application in risk evaluation[J]. Knowledge-Based Systems, 2017, 127(C):16-28.

Selvaraju R, Cogswell M, Das A, et al. Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization[J]. 2016.

Li H, Xu Z, Taylor G, et al. Visualizing the Loss Landscape of Neural Nets[J]. arXiv preprint arXiv:1712.09913, 2017.

Ribeiro M T, Singh S, Guestrin C. Why should i trust you?: Explaining the predictions of any classifier[C]//Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM, 2016: 1135-1144.

Wu M, Hughes M C, Parbhoo S, et al. Beyond Sparsity: Tree Regularization of Deep Models for Interpretability[J]. 2017.

Chung J, Gulcehre C, Cho K H, et al. Empirical evaluation of gated recurrent neural networks on sequence modeling[J]. arXiv preprint arXiv:1412.3555, 2014.

Hochreiter S, Schmidhuber J. Long short-term memory[J]. Neural computation, 1997, 9(8): 1735-1780.

Breiman L. Classification and regression trees[M]. Routledge, 2017.

Olszewski R T. Generalized feature extraction for structural pattern recognition in time-series data[R]. CARNEGIE-MELLON UNIV PITTSBURGH PA SCHOOL OF COMPUTER SCIENCE, 2001.

Wei L, Keogh E. Semi-supervised time series classification[C]//Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2006: 748-753.

Davis L M. Predictive modelling of bone ageing[D]. University of East Anglia, 2013.

Cao F, Huang H K, Pietka E, et al. Digital hand atlas and web-based bone age assessment: system design and implementation[J]. Computerized medical imaging and graphics, 2000, 24(5): 297-307.

Craven M, Shavlik J W. Extracting tree-structured representations of trained networks[C]//Advances in neural information processing systems. 1996: 24-30.