Neural Networks are Decision Trees - note

Yuki Sakishita

2022 年 11 月 12 日

Caglar Aytekin, "Neural Networks Are Decision Trees." (2022)[1] [*1] のレビュー

Abstract

- 区分線形な活性化関数（ReLU のような折れ線状の関数）の Neural Network（NN）が決定木と等価であることを示した
  - 解釈可能性の向上
  - （活性化関数が区分線形であれば）近似を含まない等価な表現が得られる
- ResNet, CNN, RNN へ拡張.
- 元の NN と，等価に構成した決定木を比較すると，空間容量が増える代わりに推論の計算量は小さくなる.

目次

[*1] ライセンス表記: Caglar Aytekin "Neural Networks Are Decision Trees." (2022) / CC BY-NC-SA 4.0 ↩

- Toy Model での実験
- 講評
- 補足：バイアス項の入れ方

1 Introduction

Newral Network（以下 NN）はブラックボックス性のため，信頼性の高い用途への妨げとなっており，説明可能性の研究が行われてきた．既存手法として saliency map・解釈可能な方法による近似・モデルの組み合わせなどが挙げられる．

saliency map（顕著性マップ）は NN が予測によく利用する入力空間の領域を図示したものである（例：図 1）．判断を指示する領域が健全であるかを確認するなどの目的には利用できるが，論理的な理由づけはできていない．

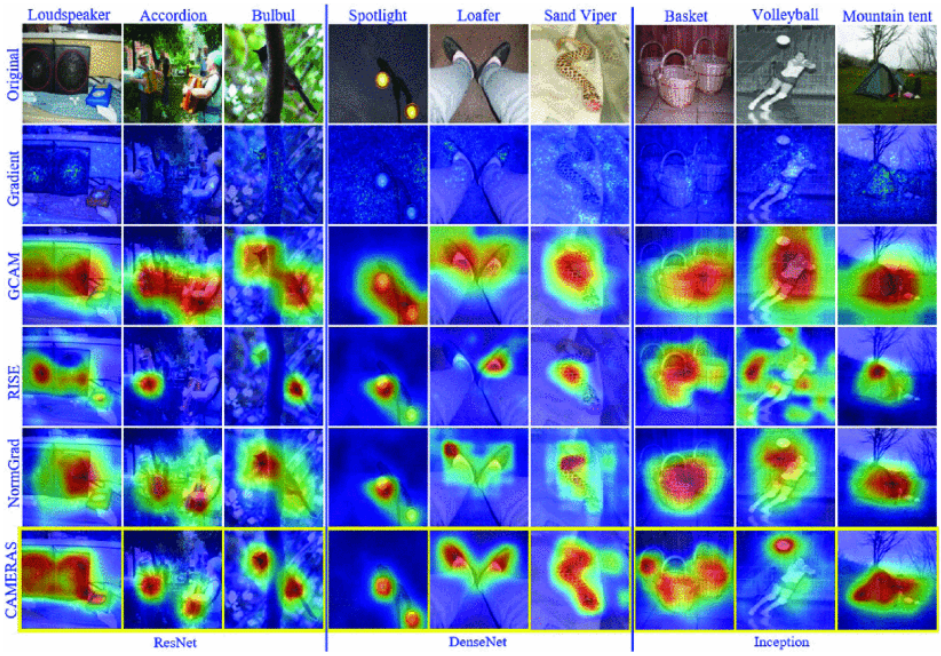

図 1　いくつかの手法・モデルによる saliency map の例．（Mohammad et al.(2021).[2] より）

NN を解釈可能なモデルで近似する方法は NN の近似であるため，元のモデルよりも性能が落ちる．

NN と決定木を組み合わせたモデルではバックボーンは NN のままなので完全な論理的推論はできず，決定の良し悪しを検証する手段を提供するにとどまる．

本論文では NN が近似を含まず等価な決定木の表現を持つことを示す．特にフィード

フォワード・ReLU の NN については先行研究に類似するところがあるが，一般の活性化関数でスキップ層・CNN・RNN の場合にも拡張できる．

2　Neural Network と等価な Decision Tree の構成

2.1　全結合 NN の場合

Notation:

学習済みの NN　$y = \mathrm{NN}(\boldsymbol{x}_0)$: 隠れ層 $n-1$，入力 $\boldsymbol{x}_0$ （m_{in} 次元），出力 y（m_{out} 次元）．
活性化関数 σ　区分線形関数．i.e. 場合分けされた領域ごとに 1 次方程式で表せるもの．
　　e.g. ReLU，leaky-ReLU など．区分けされた領域の数を k とする [*2] ．
各層の出力　$\boldsymbol{x}_i$（m_i 次元）
重み行列　W_i

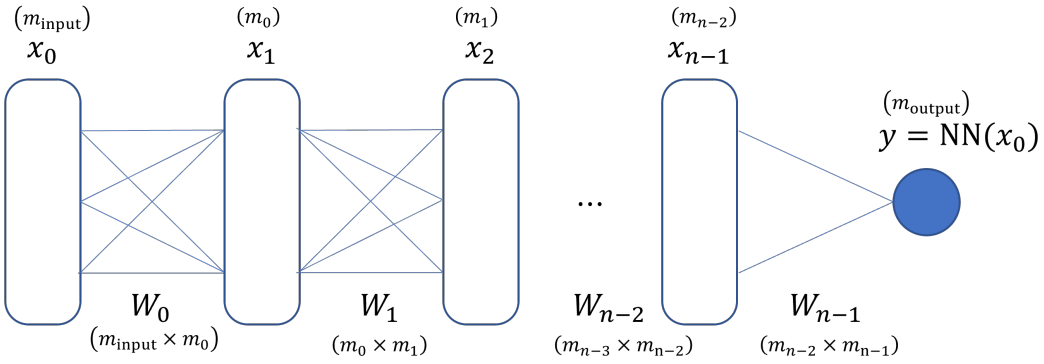

図 2　全結合 NN：モデル構造

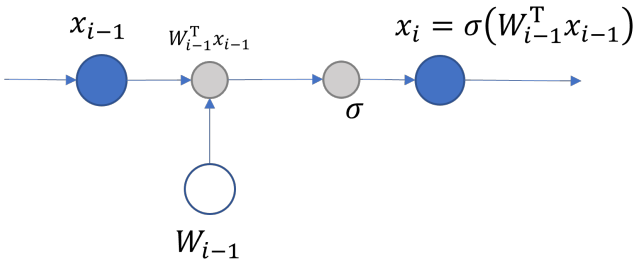

図 3　全結合 NN：層の中身

[*2] ReLU なら $k = 2$ ↩

全結合 NN の各層の順伝播は

$$\boldsymbol{x}_i = \sigma(W_{i-1}^{\mathsf{T}} \boldsymbol{x}_{i-1})$$

のようになり，出力は

$$\begin{aligned} &\mathrm{NN}(\boldsymbol{x}_0) = W_{n-1}^{\mathsf{T}} \boldsymbol{x}_{n-1} \\ &= W_{n-1}^{\mathsf{T}} \sigma(W_{n-2}^{\mathsf{T}} \sigma(\cdots W_1^{\mathsf{T}} \sigma(W_0^{\mathsf{T}} \boldsymbol{x}_0) \cdots)) \end{aligned} \tag{1}$$

と表せる．バイアス項は 1 を最後の次元に追加すれば線形で書けるので省略する．

区分線形である σ は各領域でスカラー倍とみなせるので，要素積を $\odot$ として

$$W_i^{\mathsf{T}} \sigma(W_{i-1}^{\mathsf{T}} \boldsymbol{x}_{i-1}) = W_i^{\mathsf{T}} \big(\boldsymbol{a}_{i-1} \odot \big(W_{i-1}^{\mathsf{T}} \boldsymbol{x}_{i-1}\big)\big) \tag{2}$$

と書ける．$\boldsymbol{a}_i$ の第 j 要素は $W_{i-1}^{\mathsf{T}} \boldsymbol{x}_{i-1}$ の第 j 要素が該当する領域での σ の傾き． [*3]

成分計算を考えると，以下のように書き換えられる．

$$(\mathrm{rhs}) = (W_i \odot \boldsymbol{a}_{i-1})^{\mathsf{T}} \big(W_{i-1}^{\mathsf{T}} \boldsymbol{x}_{i-1}\big) \tag{3}$$

ここでの $\odot$ は列方向ごとの要素積.

これを Eq. (1) に適用すると，

$$\mathrm{NN}(\boldsymbol{x}_0) = (W_{n-1} \odot \boldsymbol{a}_{n-2})^{\mathsf{T}} (W_{n-2} \odot \boldsymbol{a}_{n-3})^{\mathsf{T}} \cdots (W_1 \odot \boldsymbol{a}_0)^{\mathsf{T}} W_0^{\mathsf{T}} \boldsymbol{x}_0 \tag{4}$$

ここで有効重み行列 ${}_{\boldsymbol{c}_i}\hat{W}_i$ を次のように定義する [*4]

$$\begin{aligned} {}_{\boldsymbol{c}_i}\hat{W}_i^{\mathsf{T}} \boldsymbol{x}_0 &= W_i^{\mathsf{T}} \boldsymbol{x}_i \\ {}_{\boldsymbol{c}_i}\hat{W}_i^{\mathsf{T}} &= (W_i \odot \boldsymbol{a}_{i-1})^{\mathsf{T}} (W_{i-1} \odot \boldsymbol{a}_{i-2})^{\mathsf{T}} \cdots (W_1 \odot \boldsymbol{a}_0)^{\mathsf{T}} W_0^{\mathsf{T}} \end{aligned} \tag{5}$$

ここで $\boldsymbol{c}_i$ は第 i 層までの分類ベクトルで，$\boldsymbol{a}_0, \ldots, \boldsymbol{a}_{i-1}$ を繋げたものである．

$$\boldsymbol{c}_i = \boldsymbol{a}_0 \parallel \boldsymbol{a}_1 \parallel \cdots \parallel \boldsymbol{a}_{i-1}$$

($i = 0$ では 0 次元ベクトル.)

Eq. (5) で index をずらしたものを比較すれば

$${}_{\boldsymbol{a}_0 \parallel \cdots \parallel \boldsymbol{a}_{i-2} \parallel \boldsymbol{a}_{i-1}}\hat{W}_i^{\mathsf{T}} = (W_i \odot \boldsymbol{a}_{i-1})^{\mathsf{T}} \, {}_{\boldsymbol{a}_0 \parallel \cdots \parallel \boldsymbol{a}_{i-2}}\hat{W}_{i-1}^{\mathsf{T}}$$

[*3] e.g. ReLU なら $x < 0$ で 0，$x \geq 0$ で 1. ↩

[*4] 元論文では大文字と小文字の C が混じっているため見ずらいが，$\boldsymbol{c}_i$ は左下付き添字. ↩

と書けることがわかる．すなわち，${}_{\boldsymbol{c}_{i-1}}\hat{W}_{i-1}$ が定まれば，k^{m_i} 通りの $\boldsymbol{a}_{i-1}$ について ${}_{\boldsymbol{c}_{i-1}\| \boldsymbol{a}_{i-1}}\hat{W}_i$ が得られる．$\boldsymbol{a}_i$ の各次元（ノード）ごとに処理すれば，各階層で k 分岐する決定木が得られ，最終的な値 ${}_{\boldsymbol{c}_{n-1}}\hat{W}_{n-1}^{\mathsf{T}}\boldsymbol{x}_0$ [*5] を出力とする NN の表現が得られた．

また，入力 $\boldsymbol{x}_0$ が与えられれば各層の前の出力 $W_{i-1}^{\mathsf{T}}\boldsymbol{x}_{i-1} = {}_{\boldsymbol{c}_{i-1}}\hat{W}_{i-1}^{\mathsf{T}}\boldsymbol{x}_0$ が定まるので，$\boldsymbol{a}_{i-1}$ がどの値を取るかが定まる．よって各分岐の真偽が定まるので決定木の出力も得られる．

例として図 4 に示す深さ 2，幅 2 で活性化関数が ReLU

$$\sigma(x) = \begin{cases} 0 & (x \leq 0) \\ x & (x > 0) \end{cases}$$

の全結合 NN を考えると，最初の階層では $\boldsymbol{a}_0$ の第 0 成分が 0 か 1 か，すなわち ${}_{()}\hat{W}_0\boldsymbol{x}_0$ の第 1 成分が負か正かによって分岐している．次の階層では同様に第 1 成分により分岐している． [*6]

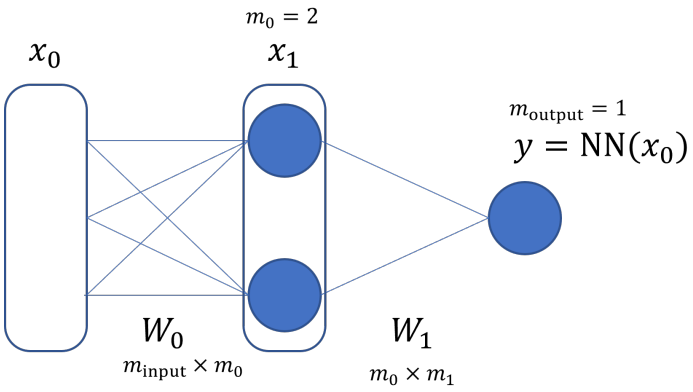

図 4　2 層 NN

3 階層目では $\boldsymbol{c}_1 = \boldsymbol{a}_0 = (0,0),(0,1),(1,0),(1,1)$ についてそれぞれ ${}_{\boldsymbol{c}_1}\hat{W}_1\boldsymbol{x}_0$ の正負で分岐し，最終的な出力 ${}_{\boldsymbol{c}_2}\hat{W}_2\boldsymbol{x}_0$ が葉，すなわち決定木の出力になっている．

この決定木は葉の数が $k^{\sum_i m_i}$ と膨大な分岐を持つが，論理的に偽な分岐（$x > 0$ で真を取ったあとに $x < -1$ の分岐がある場合など）を削除すれば関数形を保ったまま枝刈りをすることができるのでいくらかましになる．

[*5] $\boldsymbol{x}_0$ の領域内での 1 次関数である ↩

[*6] 図 5 では「${}_{\boldsymbol{c}_i}\hat{W}_i^{\mathsf{T}}\boldsymbol{x}_i$ の第 j 成分」が ${}_{\boldsymbol{c}_i}\hat{W}_{ij}^{\mathsf{T}}\boldsymbol{x}_i$ と表記されているが，先に行列の要素を取っては行列積が計算できないので正しくは $\left({}_{\boldsymbol{c}_i}\hat{W}_i^{\mathsf{T}}\boldsymbol{x}_i\right)_j$ であろう． ↩

2.2 Skip Connection を持つ（Residual Net）場合

以下の Residual NN を考える.

$$
\begin{aligned}
{}_{\mathrm{r}}\boldsymbol{x}_0 &= W_0^{\mathsf{T}}\boldsymbol{x}_0 \\
{}_{\mathrm{r}}\boldsymbol{x}_i &= {}_{\mathrm{r}}\boldsymbol{x}_{i-1} + W_i^{\mathsf{T}}\sigma({}_{\mathrm{r}}\boldsymbol{x}_{i-1})
\end{aligned}
\tag{6}
$$

全結合の場合と同様にすれば

$$
{}_{\mathrm{r}}\boldsymbol{x}_i = \left(I + (W_i \odot \boldsymbol{a}_{i-1})^{\mathsf{T}}\right) {}_{\mathrm{r}}\boldsymbol{x}_{i-1} \tag{7}
$$

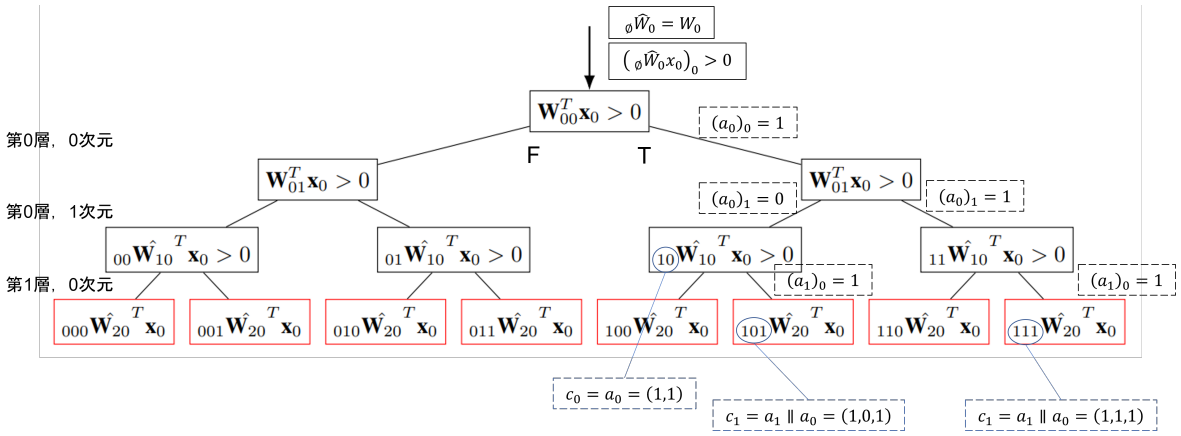

図 5　図 4 の NN の決定木 (論文 [1]Figure1 より)

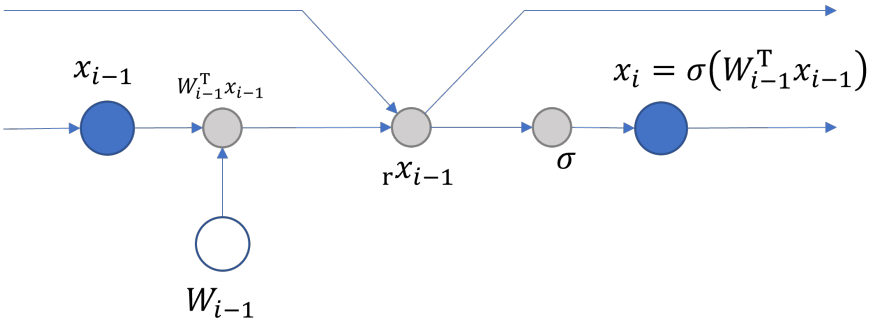

図 6　ResNet：層の中身

これを繰り返し適用すると，

$$
\begin{aligned}
{}_{\mathrm{r}}\boldsymbol{x}_i &= {}^{\mathrm{r}}_{\boldsymbol{c}_i}\hat{W}_i^{\mathsf{T}}\boldsymbol{x}_0 \\
{}^{\mathrm{r}}_{\boldsymbol{c}_i}\hat{W}_i^{\mathsf{T}} &= \Big(I + (W_i \odot \boldsymbol{a}_{i-1})^{\mathsf{T}}\Big)\Big(I + (W_{i-1} \odot \boldsymbol{a}_{i-2})^{\mathsf{T}}\Big)\cdots\Big(I + (W_1 \odot \boldsymbol{a}_0)^{\mathsf{T}}\Big)W_0^{\mathsf{T}}
\end{aligned}
\tag{8}
$$

のように Eq. (5) と同様の形が得られ，決定木に変換できることがわかる．

2.3 CNN の場合

畳み込みカーネルをサイズ $C_{i+1} \times C_i \times M_i \times N_i$ のテンソル $\boldsymbol{K}_i$, 各層の入力をサイズ $C_i \times H_i \times W_i$ のテンソル $\boldsymbol{F}_i$ としたとき，Convolutional NN の関数は

$$
\begin{aligned}
\mathrm{CNN}(\boldsymbol{F}_0) &= \boldsymbol{K}_{n-1} * \sigma(\boldsymbol{K}_{n-2} * \sigma(\cdots\sigma(\boldsymbol{K}_0 * \boldsymbol{F}_0)\cdots)) \\
\boldsymbol{F}_i &= \sigma(\boldsymbol{K}_{i-1} * \sigma(\cdots\sigma(\boldsymbol{K}_0 * \boldsymbol{F}_0)\cdots))
\end{aligned}
\tag{9}
$$

全結合 NN の Eq. (1) の行列積が畳み込み積になっただけなので同様に変形することができる．

$$
\boldsymbol{K}_i * \sigma(\boldsymbol{K}_{i-1} * \boldsymbol{F}_{i-1}) = (\boldsymbol{K}_i \odot \boldsymbol{a}_{i-1}) * (\boldsymbol{K}_{i-1} * \boldsymbol{F}_{i-1}) \tag{10}
$$

ただし，ここでの $\boldsymbol{a}_{i-1}$ はベクトルではなく $\boldsymbol{K}_{i-1} * \boldsymbol{F}_{i-1}$ のサイズのテンソルである．変形を続けると，

$$
\begin{aligned}
{}_{\boldsymbol{c}_{i-1}}\hat{\boldsymbol{K}}_i &= (\boldsymbol{K}_i \odot \boldsymbol{a}_{i-1}) * \cdots * (\boldsymbol{K}_1 \odot \boldsymbol{a}_0) * \boldsymbol{K}_0 \\
{}_{\boldsymbol{c}_{i-1}}\hat{\boldsymbol{K}}_i * \boldsymbol{F}_0 &= \boldsymbol{K}_i * \boldsymbol{F}_i
\end{aligned}
\tag{11}
$$

が得られ [*7]，これまでと同様に等価な決定木を得ることができる．

2.4 RNN の場合

Recurrent NN はフィードフォワードに展開できるため，これまでと同様に決定木で表現できる．

以下で定義される RNN を考える．

$\boldsymbol{x}^{(t)}$: 入力
$\boldsymbol{o}^{(t)}$: 出力
$\boldsymbol{h}^{(t)}$: 再帰層

[*7] 元論文では 2 行目の $\boldsymbol{F}.$ が $\boldsymbol{x}.$ になっている．誤植か． ↩

W, U, V:　重み行列

$$\begin{aligned}\boldsymbol{h}^{(t)} &= \sigma(W^{\mathsf{T}}\boldsymbol{h}^{(t-1)} + U^{\mathsf{T}}\boldsymbol{x}^{(t)}) \\ \boldsymbol{o}^{(t)} &= V^{\mathsf{T}}\boldsymbol{h}^{(t)}\end{aligned} \tag{12}$$

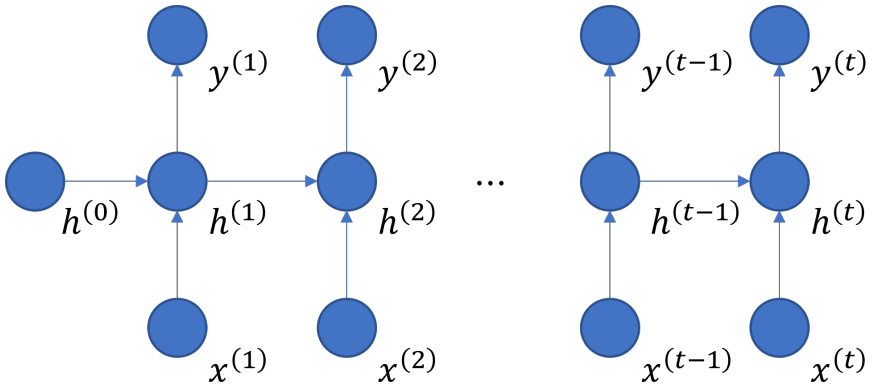

図 7　RNN：モデル構造

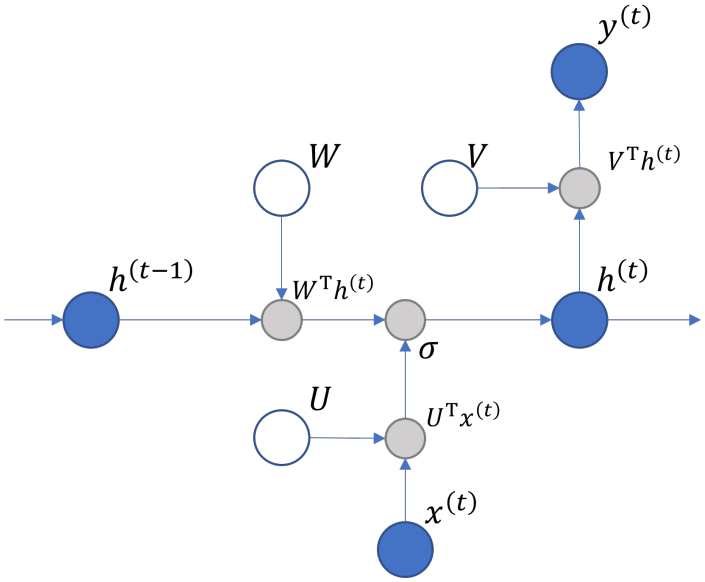

図 8　RNN：層の中身

活性化関数をスカラーに変換して,

$$\boldsymbol{h}^{(t)} = \boldsymbol{a}^{(t)} \odot \left(W^{\mathsf{T}}\boldsymbol{h}^{(t-1)} + U^{\mathsf{T}}\boldsymbol{x}^{(t)}\right) \tag{13}$$

これを繰り返し適用すると,

$$
\begin{aligned}
\boldsymbol{h}^{(t)} &= \boldsymbol{a}^{(t)} \odot \left(W^{\mathsf{T}} \left(\boldsymbol{a}^{(t)} \odot \left(W^{\mathsf{T}} \boldsymbol{h}^{(t-2)} + U^{\mathsf{T}} \boldsymbol{x}^{(t-1)} \right) \right) + U^{\mathsf{T}} \boldsymbol{x}^{(t)} \right) \\
&= \boldsymbol{a}^{(t)} \odot \left(W^{\mathsf{T}} \left(\boldsymbol{a}^{(t-1)} \odot W^{\mathsf{T}} \boldsymbol{h}^{(t-2)} \right) \right) \\
&\quad + \boldsymbol{a}^{(t)} \odot \left(W^{\mathsf{T}} \left(\boldsymbol{a}^{(t-1)} \odot U^{\mathsf{T}} \boldsymbol{x}^{(t-1)} \right) \right) + \boldsymbol{a}^{(t)} \odot U^{\mathsf{T}} \boldsymbol{x}^{(t)} \\
&= \boldsymbol{a}^{(t)} \odot \left(\left(W \odot \boldsymbol{a}^{(t-1)} \right)^{\mathsf{T}} W^{\mathsf{T}} \boldsymbol{h}^{(t-2)} \right) \\
&\quad + \boldsymbol{a}^{(t)} \odot \left(\left(W \odot \boldsymbol{a}^{(t-1)} \right)^{\mathsf{T}} U^{\mathsf{T}} \boldsymbol{x}^{(t-1)} \right) + \boldsymbol{a}^{(t)} \odot U^{\mathsf{T}} \boldsymbol{x}^{(t)} \\
&= \cdots \\
&= \boldsymbol{a}^{(t)} \odot \left(W \odot \boldsymbol{a}^{(t-1)} \right)^{\mathsf{T}} \left(W \odot \boldsymbol{a}^{(t-2)} \right)^{\mathsf{T}} \cdots \left(W \odot \boldsymbol{a}^{(1)} \right)^{\mathsf{T}} W^{\mathsf{T}} \boldsymbol{h}^{(0)} \\
&\quad + \boldsymbol{a}^{(t)} \odot \left\{ \left(W \odot \boldsymbol{a}^{(t-1)} \right)^{\mathsf{T}} \cdots \left(W \odot \boldsymbol{a}^{(1)} \right)^{\mathsf{T}} U^{\mathsf{T}} \boldsymbol{x}^{(1)} \right. \\
&\qquad + \left(W \odot \boldsymbol{a}^{(t-2)} \right)^{\mathsf{T}} \cdots \left(W \odot \boldsymbol{a}^{(1)} \right)^{\mathsf{T}} U^{\mathsf{T}} \boldsymbol{x}^{(2)} \\
&\qquad + \cdots \\
&\qquad + \left(W \odot \boldsymbol{a}^{(2)} \right)^{\mathsf{T}} \left(W \odot \boldsymbol{a}^{(1)} \right)^{\mathsf{T}} U^{\mathsf{T}} \boldsymbol{x}^{(t-2)} \\
&\qquad \left. + \left(W \odot \boldsymbol{a}^{(1)} \right)^{\mathsf{T}} U^{\mathsf{T}} \boldsymbol{x}^{(t-1)} + I U^{\mathsf{T}} \boldsymbol{x}^{(t)} \right\}
\end{aligned}
$$

すなわち

$$
\begin{aligned}
\boldsymbol{h}^{(t)} = \boldsymbol{a}^{(t)} \odot & \left(\coprod_{j=t-1}^{1} \left(W \odot \boldsymbol{a}^{(j)} \right)^{\mathsf{T}} \right) W^{\mathsf{T}} \boldsymbol{h}^{(0)} \\
& + \boldsymbol{a}^{(t)} \odot \sum_{i=1}^{t} \left(\coprod_{j=t-1}^{i} \left(W \odot \boldsymbol{a}^{(j)} \right)^{\mathsf{T}} \right) U^{\mathsf{T}} \boldsymbol{x}^{(i)}
\end{aligned}
\tag{14}
$$

[*8] ．ここで $\coprod_{j=t-1}^{i}$ は $t-1$ から i までの降順総積，要素がない場合は I を表す.

[*8] 元論文では transpose の位置が違うが，次元が合わないためおそらく誤植. ↩

総積の部分を ${}_{\boldsymbol{c}_i}\hat{W}_i$ にまとめて,

$$\begin{aligned}\boldsymbol{h}^{(t)} &= \boldsymbol{a}^{(t)} \odot {}_{\boldsymbol{c}_1}\hat{W}_1^{\mathsf{T}} W^{\mathsf{T}} \boldsymbol{h}^{(0)} + \boldsymbol{a}^{(t)} \odot \sum_{i=1}^{t} {}_{\boldsymbol{c}_i}\hat{W}_i^{\mathsf{T}} U^{\mathsf{T}} \boldsymbol{h}^{(i)} \\ {}_{\boldsymbol{c}_i}\hat{W}_i &= \coprod_{j=t-1}^{i} \left(W \odot \boldsymbol{a}^{(j)}\right)^{\mathsf{T}}\end{aligned} \tag{15}$$

[*9].

Eq.(12) に代入して,

$$\boldsymbol{o}^{(t)} = V^{\mathsf{T}}\boldsymbol{a}^{(t)} \odot {}_{\boldsymbol{c}_1}\hat{W}_1^{\mathsf{T}} W^{\mathsf{T}} \boldsymbol{h}^{(0)} + V^{\mathsf{T}}\boldsymbol{a}^{(t)} \odot \sum_{i=1}^{t} {}_{\boldsymbol{c}_i}\hat{W}_i^{\mathsf{T}} U^{\mathsf{T}} \boldsymbol{h}^{(i)} \tag{16}$$

$$\begin{aligned}\boldsymbol{o}^{(t)} &= {}_{\boldsymbol{c}_1}\hat{\boldsymbol{Z}}_1^{\mathsf{T}} W^{\mathsf{T}} \boldsymbol{h}^{(0)} + \sum_{i=1}^{t} {}_{\boldsymbol{c}_i}\hat{\boldsymbol{Z}}_i U^{\mathsf{T}} \boldsymbol{x}^{(i)} \\ {}_{\boldsymbol{c}_i}\hat{\boldsymbol{Z}}_i^{\mathsf{T}} &= V^{\mathsf{T}}\boldsymbol{a}^{(t)} \odot {}_{\boldsymbol{c}_i}\hat{W}_i^{\mathsf{T}}\end{aligned} \tag{17}$$

ここで出力は分類ベクトル $\boldsymbol{c}_i$ だけによるから，これまでと同様，決定木に変換できる．

RNN の場合，通常活性化関数は tanh が用いられることに注意．tanh は連続的に変化するため有限の大きさの決定木を構築するには区分線形関数で近似する必要がある．

2.4.1 連続な活性化関数

これまでに見たように，等価に構成される決定木は，活性化関数の区分数 k と NN のノード数 $d = \sum_i m_i$ に対し，k^d 個の分類を持つ．連続な活性化関数の場合は無限の区分を持つ区分線形関数と考えることができるので，無限の幅の決定木が対応することになる．実用上は有限の区分線形関数での近似を考えることになる．

連続活性化関数を量子化することでノード数を小さくすることができるかもしれない[*10].

[*9] この前後は transpose がついたりつかなかったりしている．これで正しいはず． ↩

[*10] これ以上特に深掘りはされていない． ↩

3　Toy Model での実験

3.1　$y = x^2$ の回帰

深さ 3，幅 2，活性化関数 leaky-ReLU($\alpha = 0.3$):

$$\sigma(x) = \begin{cases} \alpha x & (x \leq 0) \\ x & (x > 0) \end{cases}$$

（最終層は活性化なし），バイアス項付きの全結合 NN で $y = x^2$ を fit したモデルを変換した．

アルゴリズムに従い決定木をかくと図 9 の通り，2^4 通りの分類を持つ 2 分木になる．

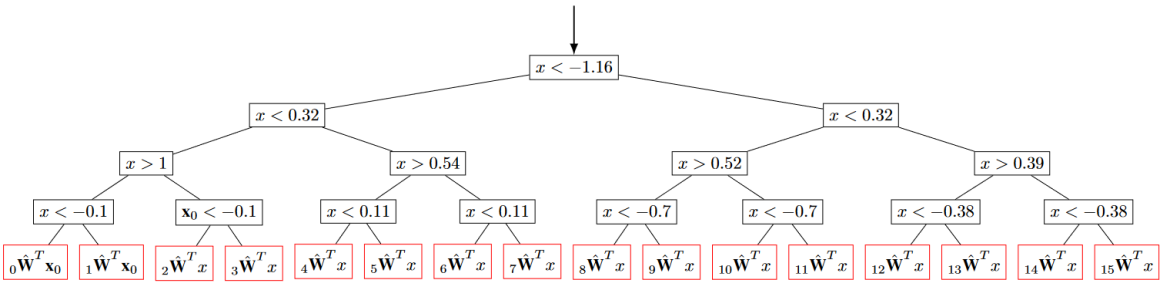

図 9　$y = x^2$ の回帰 NN に対応する決定木．${}_{\boldsymbol{c}_i}\hat{W}_i\boldsymbol{x}_0$ は展開され x について整理されている．（論文 [1]Figure2 より）

この分岐は冗長であるため，論理的に整理すると図 10 のようになり，関数が領域ごとに 1 次関数の折れ線となっていることがわかる．

この決定木と元の NN の出力をプロットすると図 11 のように一致することがわかる．

決定木に変換したことで，決定境界が非対称であることからもわかるようにモデルが対称性を学習していないことや，$x < -1.16, x > 1$ の領域には境界がないためこの領域ではモデルの推定が精度を保たないことなどが明らかになる．

3.2　半月型の分類

half-moon dataset（非線形分類のベンチマークによく用いられるデータセット）を深さ 3，幅 2，活性化関数 leaky-ReLU($\alpha = 0.3$)（最終層は sigmoid）の全結合 NN で学習

したモデルを同様に決定木に変換した．入力が 2 次元なので分岐は 2 次元平面の直線分割になる．

図 12 のように決定木に従い 16 個ほどの領域に分割され，それぞれの領域で 0/1 に分類されている．決定木の利点として，明示的な分類境界を得られる他，非有界な領域は不適切な外挿になっている可能性があること，領域にサンプル点が含まれているかどうかでその領域の予測に信用がおけるかどうかを判断できるといったことが挙げられている．

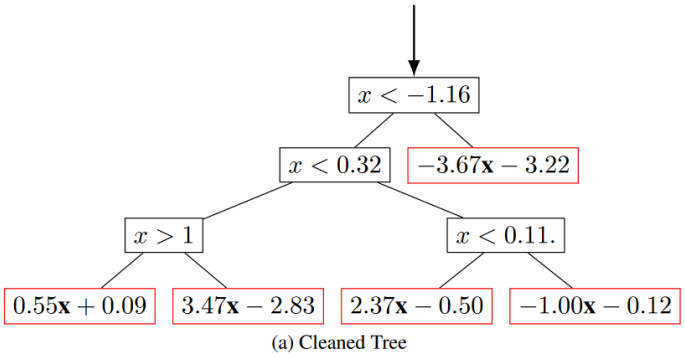

図 10　図 9 を整理した決定木．（論文 [1]Figure3(a) より）

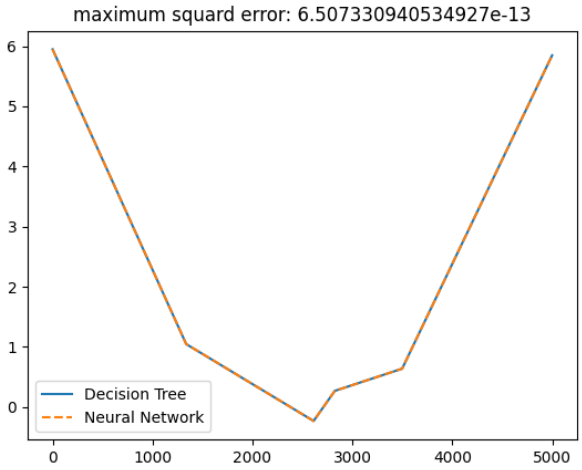

図 11　NN と変換された決定木の出力をプロットしたもの．https://github.com/CaglarAytekin/NN_DT のデータを元にプロット．

3.3　計算量の比較

決定木表現は記憶容量が大きくなる一方で計算量は削減される．表 1 は計算回数の期待値.

	$y = x^2$			Half-Moon		
	Param.	Comp.	Mult./Add.	Param.	Comp.	Mult./Add.
Tree	14	2.6	2	39	4.1	8.2
NN	13	4	16	15	5	25

表 1　トイモデルにおける計算量と記憶容量の比較.（論文 Table1 より）

4　講評

ReLU が傾きの場合分け関数であることを考えればほとんどそのままエンコードしているだけであるが，構成の手続きを与え，実際に決定木表現が得られたのは面白かった.

テーブルデータを学習した際に Random Forest などの決定木系の学習モデルと NN で学習してエンコードした決定木を比較したらどのような結果になるか興味が持てる．また，ディープモデルで実際に変換を行うのはかなり困難であろうが，Toy model よりは大

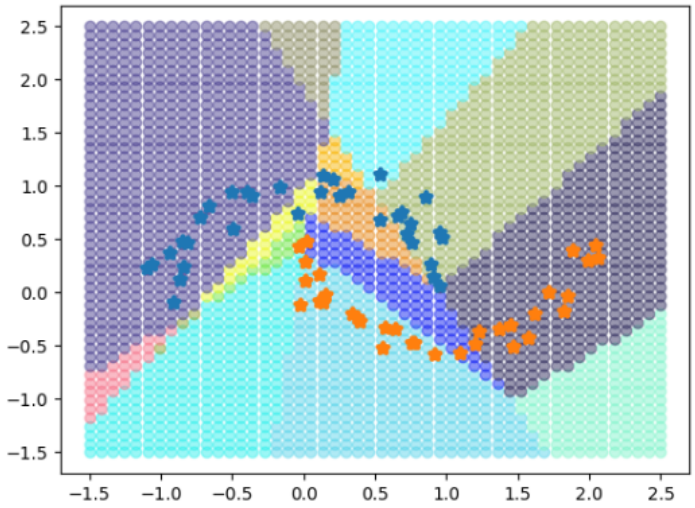

図 12　決定木による分類領域（論文 [1]Figure5 より）

きい数十次元ほどのモデルで計算量がどうなるのかも調べてみたい.

補足：バイアス項の入れ方

1 次関数はアフィン変換として扱えば行列積で書くことができる.

$$y = ax + b$$

$$\begin{pmatrix} y \\ 1 \end{pmatrix} = \begin{pmatrix} a & b \\ 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ 1 \end{pmatrix}$$

ベクトル演算でも同様

$$\boldsymbol{y} = A\boldsymbol{x} + \boldsymbol{b}$$

$$\begin{pmatrix} y_1 \\ \vdots \\ y_n \\ 1 \end{pmatrix} = \begin{pmatrix} a_{11} & \cdots & a_{1n} & b_1 \\ \vdots & \ddots & \vdots & \vdots \\ a_{n1} & \cdots & a_{nn} & b_n \\ 0 & \cdots & 0 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ \vdots \\ x_n \\ 1 \end{pmatrix}$$

$$\begin{pmatrix} \boldsymbol{y} \\ 1 \end{pmatrix} = \begin{pmatrix} A & \boldsymbol{b} \\ \boldsymbol{0}^\mathsf{T} & 1 \end{pmatrix} \begin{pmatrix} \boldsymbol{x} \\ 1 \end{pmatrix}$$

アフィン変換は乗法については線形変換と同様に扱える（加法は閉じない）. 本文の議論は乗法のみなので OK.

参考文献

[1] Aytekin, Caglar. “Neural Networks Are Decision Trees.” arXiv, October 25, 2022. https://doi.org/10.48550/arXiv.2210.05189.

[2] Jalwana, Mohammad A. A. K., Naveed Akhtar, Mohammed Bennamoun, and Ajmal Mian. “CAMERAS: Enhanced Resolution And Sanity Preserving Class Activation Mapping for Image Saliency.” In 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 16322–31, 2021. https://doi.org/10.1109/CVPR46437.2021.01606.