分段线性神经网络的精确一致解释:封闭解析法

作者: Lingyang Chu, Xia Hu, Juhua Hu, Lanjun Wang, Jian Pei
译者: 王伟哲 王宏 崔亚楠 刘赫然

摘要

由深度神经网络驱动的强大的智能机器越来越多地被部署为黑盒子，以便在诸如金融和医疗领域等对风险敏感领域中做出种种决策。为了减少潜在的风险并建立用户信任，解释这些机器做出其决策的过程至关重要。现存解释它们的方法主要是依靠分析隐神经元、预训练模仿者模型抑或是局部解释法。然而，这些方法都不能保证其解释的正确性和一致性。本文中，我们提出了*OpenBox*这一简洁优美的解析式方法来为分段线性神经网络(PLNN)做出准确而一致的解释。主要思想是首先将PLNN变形为数学上等价的一组线性分类器集合，然后通过主导其预测值的特征来解释每个线性分类器。我们进一步应用*OpenBox*证明了其有效提高了在非负稀疏约束的下对PLNNs的可解释性。基于生成数据和实际数据集的大量实验也明确证明了其解释准确性和一致性。

关键字

深度神经网络; 精确一致的解释; 解析式方法

1 引入

越来越多的机器学习系统正在重要领域做出重大决策，例如临床医疗，自动驾驶，刑事司法和军事决策[15]. 随着机器决策影响力的逐步上升，对机器学习系统给出一个明确解释而非部署做出黑盒决策机器的需求越来越强烈[17]。准确可靠地解释机器学习模型是许多重要任务的关键，例如识别故障模型[1]，与人类用户建立信任[35]，发现新知识[34]以及避免不公平问题[45]。

机器学习模型的解释问题已经被研究了几十年。如Logistic回归和支持向量机(SVM)的传统模型，从实践和理论的角度都有了很好的解释 [4]。强大的非负稀疏约束方法也被开发出来，以通过稀疏特征选择来增强传统模型的可解释性[21, 27]。然而，由于深度神经网络的复杂网络结构，现代深度模型的解释问题仍然是一个具有挑战性的领域，有待进一步探索。

正如在第2节中提到的，现有研究以三种主要方式解释深度神经网络。隐神经元分析法[9, 29, 44] 分析并可视化被神经网络中的隐神经元学习到的特征；模仿者模型法[2, 3, 7, 20]建立透明可观测模型来模拟深度神经网络的分类函数；局部解释法[11, 37, 39, 41]研究在局部扰动下输入实例的预测值，据此来寻找能影响其决策的特征。所有这些方法都对深入了解深度模型的机制有所裨益，不过没法保证它们得出的解释的确就是深度神经网络的确切行为。根据Ghorbani[13]的研究，大多数解释都十分脆弱且欠缺一致性，因为具有相同预测结果的两个具有明显不可区分的实例可以轻易地人为给出两种明显不同的解释。

我们能否为预训练的深度神经网络给出一种精确而一致的解释呢？本文中，我们给出了肯定的回答，我们找到了一种优雅的解析式方法来解释分段线性神经网络族。这里提到的**分段线性神经网络(PLNN)** [18]是一种采用了诸如MaxOut[16]和ReLU函数族[14, 19, 31]的分段线性激活函数的神经网络。PLNN的广泛应用[26]和大量实例检验[25]要求对这种神经网络的整体行为进行精确一致的解读，于是在此我们做出了以下技术贡献:

首先我们证明了PLNN在数学上等价于一组局部线性分类器，每个分类器都在凸包的输入空间上分类一组实例。第二点是我们提出了一种名为*OpenBox*的方法，通过封闭地计算出其局部线性分类器的等价集来为PLNN提供准确的解释。第三，我们通过其局部线性分类器的决策特征来解释每个实例的分类结果。因为这些相同的局部线性分类器作用于凸包中全部的实例，因此在每个凸包上的解释都是一致的。第四，我们用*OpenBox*来研究了非负稀疏约束对PLNN可解释性的影响。发现在这些条件下训练出的PLNN能选择出有意义的特征，这些特征极大地提升了模型的可解释性。最后基于生成数据和现实数据，我们进行了大量的实验来验证该方法的有效性。

本文其他内容组织如下：在第2节回顾了相关工作，在第3节我们建模了问题，将*OpenBox*的提出放在了第4节。在第5节展示了实验结果并在第6节得出了本文的结论。

2 相关工作成果

如何解释深度神经网络的整体机制是一个迫切而具有挑战性问题。

2.1 隐神经元分析法

隐神经元分析法[9, 29, 44]通过可视化，恢复映射或者标记隐层神经元学习到的特征来解释预训练的神经网络。Yosinski 等人[44]可视化了卷积神经网络(ConvNet)隐层神经元的实时激活状态，并提出了一种正则化优化方法来生成质量更优的图像。Erhan 等人[10]提出了一种激活最大化方法和一种单元取样方法来可视化被隐层神经元学到的特征。Cao 等人[5]通过一种推断隐层神经元的激活状态的反馈循环(feedback loop)来可视化神经网络对其目标对象的注意力。Li 等人[28] 通过分析自然语言处理神经模型中隐藏神经元的输出来可视化从句的组成性。为了理解隐层神经元学习到的特征，Mahendran 等人[29]提出了一种反向映射(revert-map)所学特征的通用框架，该框架可以用从图片中提取的的特征来重建图片。Dosovitskiy 等人[9]完成了同 Mahendran 等人[29]相同的任务，但他们训练的是非池增强卷积神经网络(Un-pooling-convolutional neural network)。Zhou 等人[46]通过基于对人类可理解语义概念的最佳匹配(best aligned)来标记每个隐层神经元，以此解释卷积神经网络，但获得具有所有人类语义概念的准确和完整标签的完美数据集是十分困难的。

隐神经元分析法为每个隐藏神经元的属性提供了有用的定性的见解，但定性分析每个神经元并不能为整个神经网络的全局机制给出多少值得关心的和定量的解释[12]。

2.2 模仿者模型方法

通过模仿神经网络的分类函数，模仿者模型方法[2,3,7,20]构建了一个易于解释并有着很高分类精度的透明模型。Ba等人[2]提出了一种模型压缩方法，用于使用由一个或多个深度神经网络标记过的训练实例来训练浅层模拟网络(Mimic Shallow Neural Network). Hinton等人[20]提出了一种知识蒸馏方法，通过训练相对较小的网络来模拟原始大型网络的预测概率，从而提炼出大型神经网络的知识。为了提高知识蒸馏的可解释性，Frosst和Hinton[12]通过训练软决策树来模拟深度神经网络的预测概率，扩展了蒸馏方法[20]。Che等人[7]提出了一种模仿学习方法来学习可解释的显型特征。Wu等人[42] 提出了一种树正则化的方法，该方法使用二叉决策树来模拟和正则化深度时序序列模型的分类函数。Zhu等人[48] 在深度特征嵌入网络之上构建了透明森林模型，但仍然难以解释深度特征嵌入网络。

通过模仿者模型方法构建的模仿模型比深度神经网络更容易解释。但由于模仿者模型复杂度的降低，无法保证具有较大VC维[18,24,40]的深度神经网络可以成功通过更简单的浅层模型模拟。因此模仿模型的解释与目标深度神经网络的实际整体机制之间总是存在差距。

2.3 局部解释方法

局部解释方法[11, 37, 39, 41] 通过分析其局部扰动的预测值来计算和可视化输入实例的重要特征。Simonyan等人[38]通过计算相应输入图像分类值的梯度，为每类图像生成一张有代表性的图像和一张该类的显著图。Ribeiro等人[35]提出了*LIME*来解释任意分类器的预测，该方法通过在输入实例周围的局部区域学习出一个可解释的模型来实现。Zhou等人[47]提出的CAM使用CNNs中的全局均值池化来识别每类图像有鉴别性的图像区域。Selvaraju等人[36]通过Grad-CAM泛化了CAM，Grad-CAM通过将特定类别的梯度流入CNN的最终卷积层来识别图像的重要区域。Koh等人[23]使用影响函数来追踪模型的预测并辨别对预测值有最大影响的训练实例。

局部解释方法为每个输入实例提供了富有洞见的一对一解释。然而，它对某些视角下无法区分的实例的解释可能会不一致[13]，且不影响预测结果的输入实例经过了一些简单变换[22]之后，也可能会改变局部解释法的解释。

3 问题描述

对于包含L层神经元的PLNN网络$\mathcal{N}$，网络$\mathcal{N}$的第l层记为$\mathcal{L}_l$。因此$\mathcal{L}_1$为其**输入层**，$\mathcal{L}_L$为其**输出层**。对于其他层$\mathcal{L}_l$且$l \subseteq \{2, \dots, L-1\}$为**隐藏层**。令$n_l$表示层$\mathcal{L}_l$中的神经元个数，整个神经网络中隐层的神经元总数为$\sum_{l=2}^{L-1} n_l$。

记$\boldsymbol{u}_i^{(l)}$为层$\mathcal{L}_l$的第i个神经元，$b_i^{(l)}$为其偏置，$a_i^{(l)}$为其输出，$z_i^{(l)}$为该神经元输入的全部加权和。对$\mathcal{L}_l$中的全部神经元n_l，记其偏置为向量$b^{(l-1)} = \left[b_1^{(l-1)}, \dots, b_{n_l}^{(l-1)}\right]^T$，其输出为向量$a^{(l-1)} = \left[a_1^{(l-1)}, \dots, a_{n_l}^{(l-1)}\right]^T$，输入为向量$z^{(l-1)} = \left[z_1^{(l-1)}, \dots, z_{n_l}^{(l-1)}\right]^T$。相邻层中的神经元通过带权边连接。记$\mathcal{L}_{l+1}$上第$i$个神经元和$\mathcal{L}_l$上第$j$个神经元之间的边上的权值为$W_{ij}^{(l)}$，因此$W^{(l)}$是一个规模为$n_{l+1}$乘$n_l$的矩阵。对于我们用

$$z^{(l+1)} = W^{(l)} a^{(l)} + b^{(l)} \tag{1}$$

来计算$z^{(l+1)}$。

记$f: \mathbb{R} \to \mathbb{R}$为网络$\mathcal{N}$中每个隐层神经元的分段线性激活函数。对于$l \subseteq \{2, \dots, L-1\}$我们有$a_i^{(l)} = f\left(z_i^{(l)}\right)$。我们拓展$f$的定义使其以适用于各个元素的方式作用于向量，形式为$f\left(z^{(l)}\right) = \left[f\left(z_1^{(l)}\right), \dots, f\left(z_{n_l}^{(l)}\right)\right]^T$。然后对于所有$l \subseteq \{2, \dots, L-1\}$上的神经元输出$a^{(l)}$都可以用

$$a^{(l)} = f\left(z^{(l)}\right) \tag{2}$$

来计算。

网络$\mathcal{N}$的**输入实例**记为$x \in \chi$，其中$\chi \in \mathrm{R}^d$且为一个d维的输入空间，x在下文被简称为实例。

记x_i为x的第i维。输入层$\mathcal{L}_1$包含$n_1 = d$个神经元，对所有$i \subseteq \{1, \dots, d\}$有$a_i^{(1)} = x_i$。

网络$\mathcal{N}$的**输出实例**为$\mathrm{a}^{(L)} \in \mathcal{Y}$，其中$\mathcal{Y} \in \mathrm{R}^{n_L}$且为一个$n_L$维的输出空间，输出层$\mathcal{L}_L$采用$softmax$函数计算输出，为$a^{(L)} = softmax(\mathrm{z}^{(L)})$。

PLNN等效于一个分类函数$F: \chi \to \mathcal{Y}$，其作用是将输入$x \in \chi$映射到输出$\mathrm{a}^{(L)} \in \mathcal{Y}$。该函数即是人称分段线性函数[30, 33]的$F(\cdot)$。然而由于PLNN网络的复杂性，$F(\cdot)$的全局行为非常难以理解，因此PLNN常被视为黑箱。

如何以人类可理解的方式解释PLNN的整体行为是一个有趣的问题，近年来引起了很多关注。

遵循解释机器学习模型的原则方法[4]，我们将PLNN网络$\mathcal{N}$的**解释**视作定义$\mathcal{N}$的决策边界的决策特征。如果它能清晰地给出其解释(比如，决策特征)的解析式，我们则称该模型为**可解释的**。

*定义*3.1. 有给定状态和参数不变的PLNN网络$\mathcal{N}$，可通过计算得出满足以下条件的可解释模型$\mathcal{M}$来解释$\mathcal{N}$的整体行为。

准确性：$\mathcal{M}$数学上等价于$\mathcal{N}$，因此由$\mathcal{M}$得出的解释能原本忠实地描述$\mathcal{N}$的确切行为。

一致性：$\mathcal{M}$能为同类相近实例给出相近的解释。

4 *OPENBOX*方法

在本节中，我们描述了*OpenBox*方法，该方法通过激活函数的分段线性解析式计算解释模型$\mathcal{M}$，以此得出对PLNN的精确且一致的解释。

我们首先定义PLNN $\mathcal{N}$的状态，它规定了$\mathcal{N}$中每个隐藏神经元的激活状态。然后，我们将说明如何解释固定实例上的分类结果。最后，通过计算在数学上等价于$\mathcal{N}$的解释模型$\mathcal{M}$来说明如何解释$\mathcal{N}$的整体行为。

4.1 PLNN的状态

对隐神经元 $\boldsymbol{u}_i^{(l)}$，其分段线性激活函数$f(z_i^{(l)})$有如下形式:

$$f\left(z_i^{(l)}\right)=\begin{cases} r_1 z_i^{(l)}+t_1, & if\ z_i^{(l)}\in I_1 \\ r_2 z_i^{(l)}+t_2, & if\ z_i^{(l)}\in I_2 \\ \vdots & \\ r_k z_i^{(l)}+t_k, & if\ z_i^{(l)}\in I_k \end{cases} \tag{3}$$

其中k为大于1的整数，$f\left(z_i^{(l)}\right)$包含k个线性函数，$\{r_1,\dots,r_k\}$为**斜率**，$\{t_1,\dots,t_k\}$为**截距**，$\{I_1,\dots,I_k\}$为一组分割$\mathbb{R}$的**实数区间**。

给定一个PLNN网络$\mathcal{N}$，实例$x\in\chi$决定了$z_i^{(l)}$的值，且进一步决定了线性函数$f\left(z_i^{(l)}\right)$的函数值。通过判断代入线性函数$f\left(z_i^{(l)}\right)$的哪一个分段，我们能将每个隐神经元的激活状态编码为k个**状态**，每个状态唯一对应于$f\left(z_i^{(l)}\right)$这k个线性函数中相应的一个函数分段。记 $c_i^{(l)}\in\{1,\dots,k\}$为$\boldsymbol{u}_i^{(l)}$的状态，有$z_i^{(l)}\in I_q$当且仅当$c_i^{(l)}=q\ (q\in\{1,\dots,k\})$。因为输入$z_i^{(l)}$在神经元之间各不相同，不用的隐神经元的状态也就千差万别。

记向量$c^{(l)}=\left[c_1^{(l)},\dots,c_{n_l}^{(l)}\right]$为层$\mathcal{L}_l$所有隐神经元的状态，$\mathcal{N}$的**状态**为一个$N$维的向量，又记$C=\left[c^{(2)},\dots,c^{(L-1)}\right]$为$\mathcal{N}$中所有隐神经元的状态。

一个PLNN的固定实例x唯一决定了其状态C。把实例$x\in\chi$映射为状态$C\in\{1,\dots,k\}^T$的函数记为 $conf:\chi\to\{1,\dots,k\}^T$。对每个神经元$\boldsymbol{u}_i^{(l)}$，分别记$r_i^{(l)}$和$t_i^{(l)}$为线性函数对应状态$c_i^{(l)}$的斜率和截距，$r_i^{(l)}$和$t_i^{(l)}$由$c_i^{(l)}$唯一确定，那么有$r_i^{(l)}=r_q$且有$t_i^{(l)}=t_q$当且仅当$c_i^{(l)}=q\ (q\in\{1,\dots,k\})$。

对层$\mathcal{L}_l$的所有隐神经元，可将斜率和截距写作$r^{(l)} = \left[r_1^{(l)}, \dots, r_{n_l}^{(l)}\right]^T$和$t^{(l)} = \left[t_1^{(l)}, \dots, t_{n_l}^{(l)}\right]^T$，然后隐层$\mathcal{L}_l$所有神经元的激活函数可重写为：

$$f\left(z^{(l)}\right) = r^{(l)} \circ z^{(l)} + t^{(l)} \tag{4}$$

其中$r^{(l)} \circ z^{(l)}$为$r^{(l)}$和$z^{(l)}$之间的元素积(Hadamard product)。

自此，我们就可以开始解释在固定实例上的分类结果了。

表 1：本文常用记法

记法	意义
$\boldsymbol{u}_i^{(l)}$	层$\mathcal{L}_l$的第i个神经元
n_l	层$\mathcal{L}_l$中的神经元个数
N	网络$\mathcal{N}$中隐层神经元总数
$z_i^{(l)}$	层$\mathcal{L}_l$中第i个神经元的输入
$c_i^{(l)}$	层$\mathcal{L}_l$中第i个神经元的状态
C_h	PLNN网络$\mathcal{N}$的第h个状态
P_h	由C_h决定的第h个凸包(convex polytope)
$F_h(\cdot)$	由C_h决定的第h个线性分类器
Q_h	定义P_h的一组线性不等式

4.2 单独实例上的精确解释

给定恒定PLNN网络$\mathcal{N}$，由如下步骤推导其解析式$F(x)$来解释固定实例$x \in \chi$的分类结果。对式(2)和式(4)，其中所有$l \in \{2, \dots, L-1\}$，有：

$$a^{(l)} = f\left(z^{(l)}\right) = r^{(l)} \circ z^{(l)} + t^{(l)}$$

将$a^{(l)}$代入式(1)，可将$z^{(l+1)}$重写为如下形式：

$$z^{(l+1)} = W^{(l)}\left(r^{(l)} \circ z^{(l)} + t^{(l)}\right) + b^{(l)} = \widetilde{W}^{(l)} z^{(l)} + \tilde{b}^{(l)} \tag{5}$$

将其中$\tilde{b}^{(l)} = W^{(l)} t^{(l)} + b^{(l)}$和$\widetilde{W}^{(l)} = W^{(l)} \circ r^{(l)}$定义为增强的元素积操作，因此其第$i$行第$j$列上的元素即为$\widetilde{W}_{ij}^{(l)} = W_{ij}^{(l)} r_{ij}^{(l)}$。

不断将式(5)代入自式，可在$l \in \{2, \dots, L-1\}$上得到$z^{(l+1)}$：

$$z^{(l+1)} = \prod_{h=0}^{l-2} \widetilde{W}^{(l-h)} z^{(2)} + \sum_{h=2}^{l} \prod_{q=0}^{l-h-1} \widetilde{W}^{(l-q)} \tilde{b}^{(h)}$$

再将$z^{(2)} = W^{(1)} a^{(1)} + b^{(1)}$和$a^{(1)} = x$带入上式，对所有$l \in \{2, \dots, L-1\}$，可将$z^{(l+1)}$重写为如下形式：

$$\begin{aligned} z^{(l+1)} &= \prod_{h=0}^{l-2} \widetilde{W}^{(l-h)} W^{(1)} x + \prod_{h=0}^{l-2} \widetilde{W}^{(l-h)} b^{(1)} + \sum_{h=2}^{l} \prod_{q=0}^{l-h-1} \widetilde{W}^{(l-q)} \tilde{b}^{(h)} \\ &= \widehat{W}^{(1:l)} x + \hat{b}^{(1:l)} \end{aligned} \tag{6}$$

其中$\widehat{W}^{(1:l)}=\prod_{h=0}^{l-2}\widehat{W}^{(l-h)}W^{(1)}$为$x$的系数矩阵，$\hat{b}^{(1:l)}$是余项的和。其中的记号$(1:l)$表示$\widehat{W}^{(1:l)}x+\hat{b}^{(1:l)}$等价于PLNN中层$\mathcal{L}_1$到层$\mathcal{L}_l$的前向传播过程。

因为$\mathcal{N}$在一个输入$x\in\chi$的输出为$F(x)=\mathrm{a}^{(L)}=softmax(\mathrm{z}^{(L)})$，$F(x)$的解析式为：

$$F(x)=softmax\left(\widehat{W}^{(1:L-1)}x+\hat{b}^{(1:L-1)}\right) \tag{7}$$

对给定的PLNN网络$\mathcal{N}$和固定实例x，$\widehat{W}^{(1:L-1)}$和$\hat{b}^{(1:L-1)}$都是由给定的状态$C=conf(x)$唯一确定的固定参数。因此对于恒定实例x，$F(x)$即是由$\widehat{W}^{(1:L-1)}x+\hat{b}^{(1:L-1)}$定义了明确决策边界的线性分类器。

受到如Logistic回归和线性SVM [4]等传统线性分类器广泛使用的解释方法的启发，我们通过$F(x)$的决策特征来解释恒定实例x上的预测。具体来说，$\widehat{W}^{(1:L-1)}$的第i行的元素就是第i类实例的决策特征。

式(7)给出了解释一个固定实例上分类结果的一个直观方式。不过，单独解释每个实例的分类结果和对PLNN $\mathcal{N}$整体行为的理解还相差甚远。下一节，我们描述了如何通过计算在数学上等价于$\mathcal{N}$的解释模型$\mathcal{M}$来解释$\mathcal{N}$的整体行为。

4.3 PLNN全局的精确解释

一个由N个神经元组成的恒定PLNN网络$\mathcal{N}$最多有k^N种状态。用$C_h\in C$表示PLNN网络$\mathcal{N}$的第h个状态，其中$C\subseteq\{1,\dots,k\}^N$为$\mathcal{N}$的状态集合。回想一下，每个实例$x\in\chi$唯一地确定状态$conf(x)\in C$。由于$C$的容量$|C|$最多为$k^N$，但$\chi$中的实例数量可以任意大，显然$C$中的至少有一个状态被$\chi$中多个实例所共有。

记$P_h=\left\{x\in\chi\,\middle|\,conf(x)=C_h\right\}$为有相同状态$C_h$的一组实例。定理4.1中我们证明了，对任意状态$C_h\in C$有$P_h$为$\chi$的凸包。

定理4. 1：给定有N个隐神经元的恒定PLNN网络$\mathcal{N}$，$\forall C_h\in C$，$P_h=\left\{x\in\chi\,\middle|\,conf(x)=C_h\right\}$为$\chi$中的一个凸包。

证明：我们可以通过证明$conf(x)=C_{\mathrm{h}}$等价于一个关于x的有限线性不等式集来证明该定理。

当$l=2$，有$z^{(2)}=W^{(1)}x+b^{(1)}$。对于$l\in\{3,\dots,L-1\}$，都服从式(6)这一关于x的线性函数$z^{(l)}=\widehat{W}^{(1:l-1)}x+\hat{b}^{(1:l-1)}$，因为当$C_h$恒定的时候，$\widehat{W}^{(1:l-1)}$和$\hat{b}^{(1:l-1)}$为常数参数。

综上所述，对于一个固定的C_h，$z^{(l)}$为在$l\in\{2,\dots,L-1\}$上关于x的线性函数。

可以看到，因为$conf(x)=c_h$等价于容量$2N$的关于x的线性不等式集，因此P_h为一个凸包。有如下证明：

前面提到有$z_i^{(l)}\in I_q$当且仅当$c_i^{(l)}=q\ (q\in\{1,\dots,k\})$，记双射函数$\psi:\{1,\dots,k\}\to\{I_1,\dots,I_k\}$将状态$c_i^{(l)}$映射到在$\{I_1,\dots,I_k\}$里的一个实数区间上，则有$\psi\left(c_i^{(l)}\right)=I_q$当且仅当$c_i^{(l)}=q\ (q\in\{1,\dots,k\})$。那么$conf(x)=c_h$等价于一组约束，该约束记为$Q_h=\left\{z_i^{(l)}\in\psi\left(c_i^{(l)}\right)\,\middle|\,i\in\{1,\dots,n_l\},\ l\in\{2,\dots,L-1\}\right\}$。因为$z_i^{(l)}$为关于$x$的线性函数且$\psi\left(c_i^{(l)}\right)$为实数区间，$Q_h$中的每个约束$z_i^{(l)}\in\psi\left(c_i^{(l)}\right)$都等价于两个关于$x$的线性不等式。因此，$conf(x)=c_h$等价于一个关于$x$的容量$2N$的线性不等式集，意即$Q_h$为一个凸包。

由定理4.1可知，所有共享相同状态C_h的实例组成了唯一凸包P_h，而P_h由Q_h中的$2N$个线性不等式显式确定。因为C_h也确定了式(7)中固定实例的线性分类器，所以所有在凸包P_h中的实例都有由C_h确定的相同线性分类器。

记P_h中全部实例共有的线性分类器为$F_h(\cdot)$，则可以将$\mathcal{N}$翻译为**局部线性分类器集**(以下简称**LLCs**)，其中每个LLC都是作用于凸包P_h中全部实例的一个线性分类器$F_h(\cdot)$。记数组$\left(F_h(x), P_h\right)$为第h个LLC，恒定PLNN网络$\mathcal{N}$即等价于一组LLC，再记$\mathcal{M} = \left\{\left(F_h(x), P_h\right) \middle| C_h \in C\right\}$，则$\mathcal{M}$就是我们最终用来解读$\mathcal{N}$的解释模型。

对于恒定PLNN网络$\mathcal{N}$，如果其N个隐神经元的状态都相互独立，PLNN网络$\mathcal{N}$有个k^N状态，那么$\mathcal{M}$就包含了k^N个LLC。由于PLNN是层级结构(hierarchical structure)的，因此其层$\mathcal{L}_l$的隐神经元状态和其前层$\mathcal{L}_q (q < 1)$的神经元状态密切相关。那么C中元素的个数就远小于k^N且$\mathcal{M}$中的LLC个数也远小于k^N，我们会在表3和5.4节中再谈这个问题。

实际上，我们不需要一次算出$\mathcal{M}$中的全部LLC，而只需要计算$\mathcal{M}$的活动子集——即实际用于对可用实例集进行分类的LLCs。当使用新LLC对新出现的实例进行了分类时，再把这个LLC加入$\mathcal{M}$就可以了。

算法1总结了*OpenBox*的具体做法，该算法计算实际用来分类训练集(记作D_{train})的活动LLCs，然后把把这些LLCs作为$\mathcal{M}$。

算法 1: *OpenBox* (N, D_{train})

输入：$\mathcal{N}$:= 恒定PLNN，$D_{train} \in \chi$为用于训练$\mathcal{N}$的训练实例
输出：$\mathcal{M}$:= 一个活动LLCs的集合

1:初始化：$\mathcal{M} = \emptyset, C = \emptyset$
2: **for** each $x \in D_{train}$ **do**
3: 通过$C_h \leftarrow conf(x)$计算状态
4: **if** $C_h \notin C$ **then**
5: $C \leftarrow C \cup C_h$且$\mathcal{M} \leftarrow \mathcal{M} \cup \left(F_h(x), P_h\right)$
6: **end if**
7: **end for**
8: **return** $\mathcal{M}$

算法1的时间成本包括步骤3中计算$conf(x)$的时间T_{conf}和步骤5中计算LLC$\left(F_h(x), P_h\right)$的时间T_{LLC}。因为T_{conf}和T_{LLC}主要是由矩阵(向量)乘法的时间开销组成，所以我们通过标量乘法规模来评估算法1的时间成本。第一步，因为$conf(x)$是从层$\mathcal{L}_1$向层$\mathcal{L}_{L-1}$的前向传播来算得的，则$T_{conf} = \sum_{l=2}^{L-1} n_l n_{l-1}$。第二步，由于$\left(F_h(x), P_h\right)$由数组$\mathcal{G} = \left\{\left(\widehat{W}^{(1:l)}, \widehat{b}^{(1:l)}\right) \middle| l \in \{1, \dots, L-1\}\right\}$确

定，T_{LLC}即是计算$\mathcal{G}$的时间。对于给定$(\widehat{W}^{(1:l)}, \hat{b}^{(1:l)})$，我们可以通过将$z^{(l+1)} = \widehat{W}^{(1:l)}x + \hat{b}^{(1:l)}$即式(6)代入式(5)得到$(\widehat{W}^{(1:l)}, \hat{b}^{(1:l)})$，其时间开销为$n_{l+1}n_l(n_1+1)$。 由于$\widehat{W}^{(1:1)} = W^{(1)}$且$\hat{b}^{(1:1)} = b^{(1)}$，那迭代计算$\mathcal{G}$可得总时间开销为$T_{LLC} = \sum_{l=2}^{L-1} n_{l+1}n_l(n_1+1)$。

算法1的最坏情况是每个实例$x \in D_{train}$都有唯一的状态$conf(x)$，记$|D_{train}|$为训练集实例个数，算法1在最坏情况下的时间开销即为$|D_{train}|(T_{conf} + T_{LLC})$。因为$n_l, l \in \{2, \dots, L-1\}$为常数且$n_1 = d$是输入$x \in \mathbb{R}^d$的个数，则算法1的时间复杂度即为$O(|D_{train}|d)$。

现在，我们来介绍如何解释实例$x \in P_h$ $h \in \{1, \dots, |C|\}$的分类结果。首先，我们使用$F_h(x)$(第4.2节)的决策特征来解释x的分类结果。然后我们使用**包体边界特征(PBF)**来说明为什么x包含在P_h中时PBF是包体边界的决策特征。更具体地说， Q_h中的线性不等式$z_i^{(l)} \in \psi(z_i^{(l)})$定义了$P_h$的包体边界。从式(6)可知$z_i^{(l)}$是关于$x$的线性函数，而PBF是$z_i^{(l)}$中$x$的系数。

我们还发现Q_h中的一些线性不等式是多余的，因为其超平面与P_h不相交。为了简化对包体边界的解读，我们通过Caron方法[6]去掉了这些多余的不等式，并着重研究那些非冗余的PBF。

*OpenBox*有如下三个优点。第一个是其解释是精确的，因为$\mathcal{M}$的LLC集在数学上等价于$\mathcal{N}$的分类函数$F(\cdot)$。第二个是我们的解释在分类上有一致性，这是由于同一凸包中的所有实例都由完全相同的LLC分类，因此解释与给定的凸包一致。 最后一个是由于算法1的时间复杂度低，所以这一的解释方法易于计算。

5 实验

我们在本节评估了*OpenBox*的性能，并将其与现有最先进的方法LIME[35]进行了比较。具体来说，我们解决了以下问题：(1)LLC的图象是什么样的？(2)LIME和OpenBox产生的解释是否精确且一致？(3)LLC的决策特征是否易于理解？非负和稀疏约束是否能提高这些特征的可解释性？(4)如何解释LLCs的PBF？(5)*OpenBox*的解释用在改进和调试PLNN模型之后效果如何？

表2展示了我们使用的这六种模型的细节。这里的PLNN和PLNN-NS采用了相同的网络结构并使用了ReLU [14]这一广泛使用的激活函数，其网络结构如表3所示。我们采用了Chorowski等人[8]给出的非负稀疏约束来训练PLNN-NS。由于我们的目标是全面研究*OpenBox*的解释效果而非实现最好的分类性能，所以使用了相对简单的PLNN和PLNNNS网络结构，这些简单的网络结构仍然足以提供比Logistic回归(LR)好得多的分类性能。 我们将LR、LR-F、LR-NS和LR-NSF的决策特征作为基准来与LLC的决策特征进行比较。

LIME的Python代码由其作者本人公开而其他的方法和模型使用Matlab实现。在Matlab中使用了深度学习工具箱[32]来训练PLNN和PLNN-NS。所有实验均在搭载Core-i7-3370 CPU(3.40 GHz)，16GB内存和运行Windows7操作系统的5400rpm机械硬盘硬盘的PC上进行。

我们使用了如下数据集，表4中给出了数据集的具体信息。

人工生成(SYN)数据集：如图1(a)所示，该数据集包含从欧式空间中二维矩形上均匀采样的20,000个实例。红色点和蓝色点分别是正例和负例。由于我们仅仅使用SYN来可视化PLNN的LLC且不在SYN上进行测试，因此我们把所有SYN实例作为训练数据。

FMNIST-1和FMNIST-2数据集：这两个数据集各包含两种来自于Fashion MNIST数据集[43]中的图像。FMNIST-1由短靴和背包的图像构成而FMNIST-2由外套和套头衫的图像构成。FMNIST-1和FMNIST-2中的所有图像都是28×28像素的灰度图，因此我们将784像素值压缩为784维特征向量来表示图像。Fashion MNIST数据集可在这里下载。

表 2：要解读的模型

LR表示Logistic回归；NS表示非负和稀疏约束；Flip表示模型在带有翻转标签的实例上进行训练

模型	PLNN	PLNN-NS	LR	LR-F	LR-NS	LR-NSF
NS	×	√	×	×	√	√
Flip	×	×	×	√	×	√

表 3：PLNN和PLNN-NS的网络结构$(n_1, n_2, \dots, n_l)$和状态量$|c|$

相邻层中的神经元被初始化为全连接；$k = 2$为ReLU函数分段的个数，N为隐神经元的数量

数据集	神经元个数 $(n_1, n_2, \dots, n_l)$	PLNN		PLNN-NS	
		$\|c\|$	k^N	$\|c\|$	k^N
SYN	(2,4,16,2,2)	226	2^{22}	41	2^{22}
FMNIST-1	(784,8,2,2)	78	2^{10}	3	2^{10}
FMNIST-2	(784,8,2,2)	23	2^{10}	18	2^{10}

表 4：数据集的详细描述

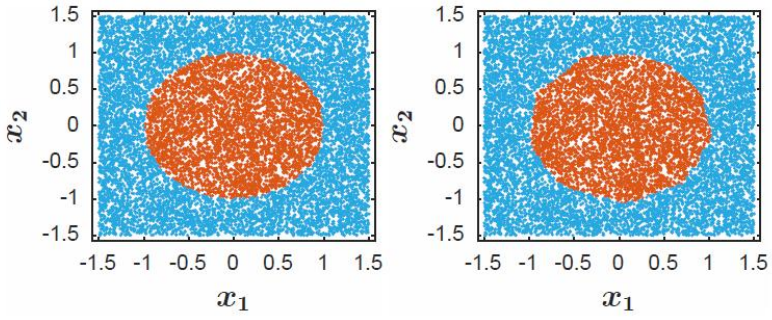
数据集	训练数据		测试数据	
	正例数量	负例数量	正例数量	负例数量
SYN	6961	13039	N/A	N/A
FMNIST-1	4000	4000	3000	3000
FMNIST-2	4000	4000	3000	3000

5.1 LLCs看上去是什么样的?

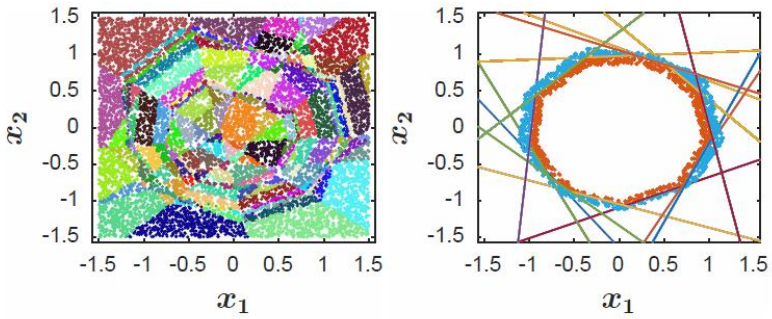
我们可视化了在SYN数据集上训练的PLNN的LLC来佐证了定理4.1。

图1(a)-(b)分别给出SYN的训练实例和PLNN在训练实例上得出的预测结果。由于所有实例都用在了训练里，因此预测准确率为99.9%。图1(c)中，我们以相同的颜色标注了所有状态相同的实例。显然，这个凸包包含了全部状态相同的实例，这证明了定理4.1的正确性。

图 1：在SYN上训练的PLNN的LLCs



(a) training data of SYN (b) prediction results of PLNN



(c) convex polytopes (d) LLCs

图1(d)展示了带有正负实例且覆盖PLNN决策边界的凸包的LLC。如图所示，实线表示LLC的决策边界，它表征正负实例之间的差异并构成PLNN的全局决策边界，所以不覆盖PLNN边界的凸包只包含一种实例。这些凸包的LLC表征了对应分类的实例的共有特征。在一节要分析的是：LLC集是否会得出与PLNN完全一致的预测结果，以及如何得到易于理解的有丰富内涵的决策特征。

5.2 *OpenBox*的解释是否精确一致?

精确一致的解释显然更适合人类思维。在本小节中，我们系统地研究了LIME和*OpenBox*在FMNIST-1和FMNIST-2上的解释的准确性和一致性。由于LIME给一整天都不够处理完所有实例，因此对于FMNIST-1和FMNIST-2中这两个数据集，我们从测试集中均匀采样600个实例，并对采样的实例进行如下实验：

首先来分析**解释的精确性**，我们比较了由*LIME*的局部解释模型和*OpenBox*在PLNN上的LLCs得出的预测结果，这里对实例的预测是将实例分类为正例的概率。

图 2：LIME，*OpenBox*和PLNN的预测
所有方案的预测均由分次独立计算得出，按降序对PLNN的预测结果进行排序

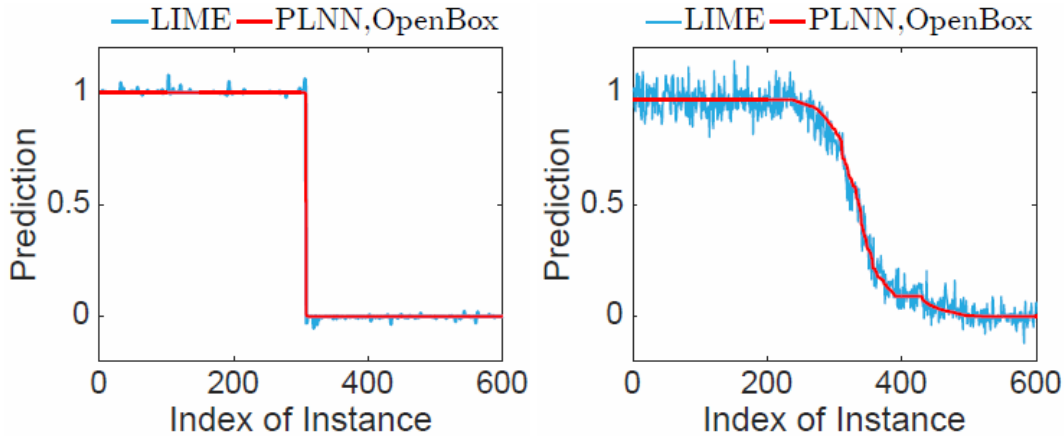
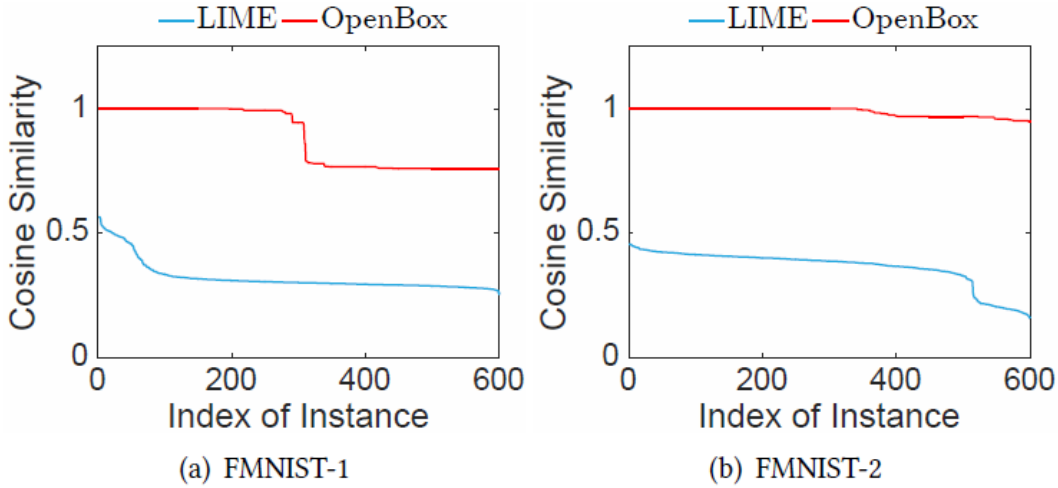


图 3：每个实例的决策特征与其最近邻之间的余弦相似性
LIME和*OpenBox*的预测结果按余弦相似度降序排列



对于图2，由于LIME不保证PLNN的局部预测的没有逼近误差，因此LIME的预测与在FMNIST-1上PLNN得出的结果并不完全相同，而且与FMNIST-2上的PLNN结果有极大的差异。差异之所以在FMNIST-2上更为显著，是因为FMNIST-2中的图像更难以区分，这使得PLNN的决策边界更复杂，更难以逼近。还可以看到LIME的预测超出了[0,1]区间，这是因为LIME的可解释模型的输出本来就不是概率。这导致了LIME计算得出的解释可能不能如实地描述PLNN的确切行为。相比之下，由于*OpenBox*得出的LLC集在数学上等价于PLNN的$F(\cdot)$，因此无论是哪个实例，*OpenBox*的预测结果都与PLNN的完全相同，由此可知LLC的决策特征准确地描述了PLNN的整体行为。

接下来通过分析相似实例的解释之间的相似性，可以研究LIME和*OpenBox*的**解释一致性**。

通常有一致性的解释方法在实例相似时会给出相近的解释。对于实例x，记x'为x在欧氏距离下的最近邻，记$\gamma, \gamma' \in \mathbb{R}^d$分别表示$x$和$x'$的分类的决策特征。可以通过$\gamma$和$\gamma'$之间的余弦相似度来衡量解释的一致性，较大的余弦相似度意味着更好的解释一致性。

如图3所示，因为*OpenBox*始终对同一凸包中的所有实例给出相同的解释，所以*OpenBox*的余弦相似度在大约50%的实例上都是1。由于最近邻x和x'可能在同一个凸包内，因此*OpenBox*的余弦相似度不会在所有实例上恒为1。对比来看，由于LIME根据每个实例的唯一局部扰动算

得独立解释结果，因此LIME的余弦相似度在所有情况下都明显低于*OpenBox*，这证明了*OpenBox*优越的解释一致性。

总结：*OpenBox*的解释是精确的并且比LIME的解释更具一致性。

5.3 LLCs的决策特征以及非负稀疏约束的影响

除了精确性和一致性之外，良好的解释还应具有强语义含义，以便人脑轻易地理解智能机器的“想法”。本小节中，首先展示了LLC的决策特征的含义，然后研究了非负稀疏约束在提高决策特征可解释性方面的作用。我们用*OpenBox*计算出了PLNN和PLNN-NS的决策特征，并将 LR、LR-F、LR-NS和LR-NSF的决策特征用作基准。表5展示了这些模型的准确率。

表 5：模型的训练和测试精度

数据集	FMNIST-1		FMNIST-2	
准确率	训练	测试	训练	测试
LR	0.998	0.997	0.847	0.839
LR-F	0.998	0.997	0.847	0.839
PLNN	**1.000**	**0.999**	**0.907**	**0.868**
LR-NS	0.772	0.776	0.711	0.698
LR-NSF	0.989	0.989	0.782	0.791
PLNN-NS	**1.000**	**0.999**	**0.894**	**0.867**

图 4：FMNIST-1上所有模型的决策特征
(a)- (e)和(f)-(j)分别展示了这些模型上短靴和背包的均值图像和决策特征
PLNN和PLNN-NS模型展示的是LLC的决策特征，其凸包包含大多数实例

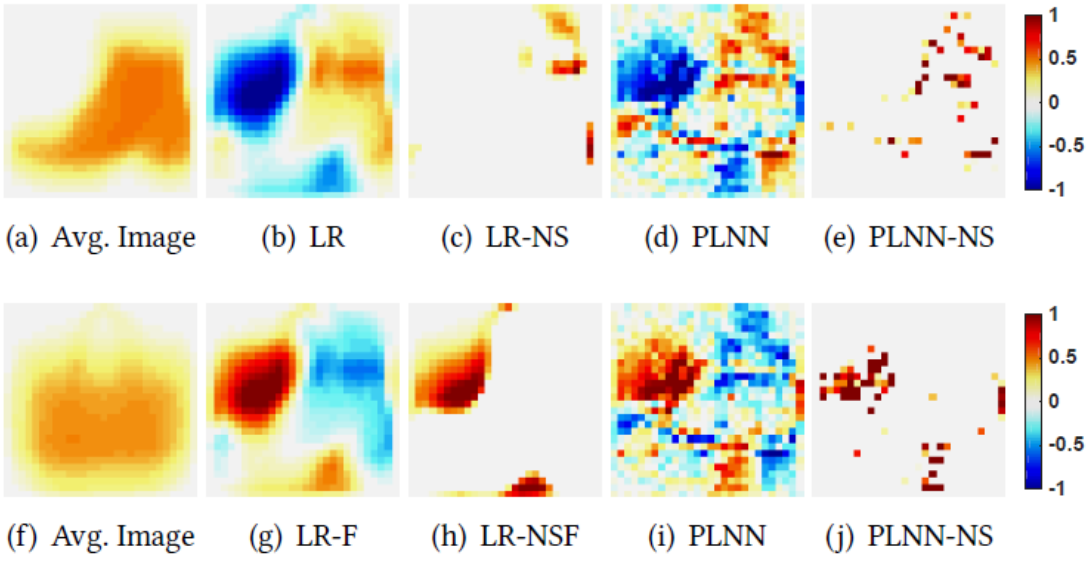


图4展示了以上所有模型在FMNIST-1上的决策特征。有趣的是，PLNN的决策特征与LR和LR-F的决策特征一样容易理解。所有这些特征都清晰地标识出了有含义的图块，比如短靴的脚踝和脚跟，背包的左上角。仔细考察均值图像会发现，正是这些决策特征描述了短靴和背包之间的区别。

PLNN的决策特征抓住了短靴和背包之间的区别，比LR和LR-F的决策特征包含了更多细节。这是因为PLNN的LLC仅包含凸包内的实例子集之间的差异，但LR和LR-F却包含了短靴和

背包的所有实例之间的整体差异。因为短靴和背包的实例很好区分，PLNN、LR和LR-F的精度尚能一较高下。PLNN因为包含比LR和LR-F更细致的特征，所以当实例难以区分时(如图5所示)也能达到显著更佳的准确率。

图 5：FMNIST-2上所有模型的决策特征
(a)- (e)和(f)-(j)分别展示了这些模型上外套和套头衫的均值图像和决策特征
PLNN和PLNN-NS模型展示的是LLC的决策特征，其凸包包含大多数实例

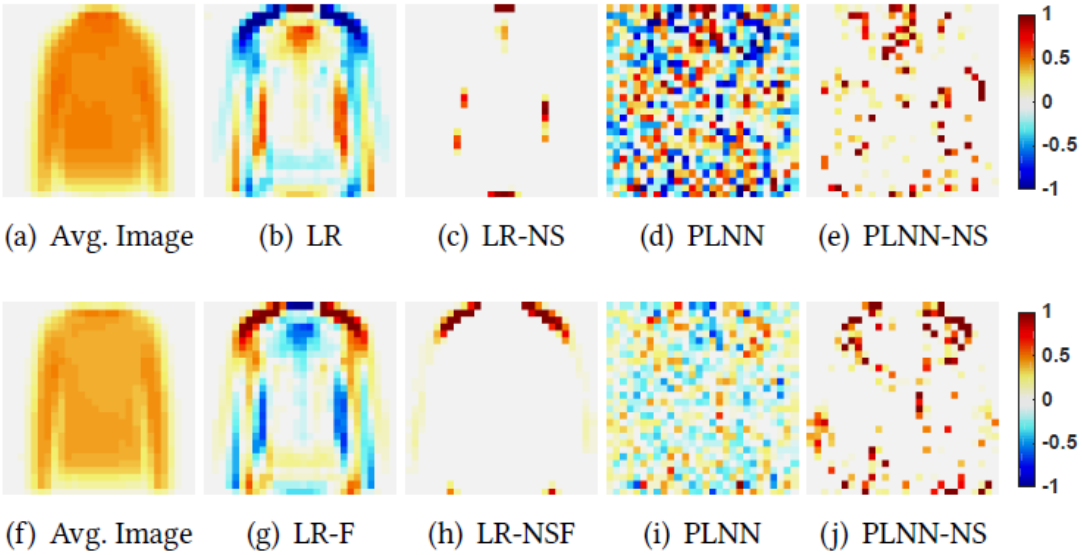


图5展示了这几个模型在FMNIST-2上的决策特征。图中LR和LR-F包含了具有强语义含义的决策特征，比如外套的衣领、胸口以及套头衫的肩部。然而这些特征太笼统而无法准确区分外套和套头衫，因此LR和LR-F没能实现较高精度。可喜的是，同样因为PLNN的决策特征比LR和LR-F包含了更多的细节，因而保持了较高准确率。

PLNN的高准确率是以杂乱的决策特征为代价的，这些特征可能难以理解。幸运的是，在PLNN上应用非负稀疏约束有效地提高了决策特征的可解释性且不会影响分类准确性。

在图4和图5中，PLNN-NS的决策特征标记出了与LR-NS和LR-NSF相似的部分，并且比PLNN的决策特征更易于理解。特别是在图5中，PLNN-NS的决策特征清晰标记出了外套的衣领和胸口和套头衫的肩部，这比PLNN的杂乱特征好理解多了。这些结果证明了在选择有意义的特征时非负稀疏约束十分有效。 此外，PLNN-NS的决策特征比LR-NS和LR-NSF包含更多细节，因此PLNN-NS可以达到与PLNN相当的准确度，并在FMNIST-2上明显优于LR-NS和LR-NSF。

总结：LLC的决策特征易于理解，非负稀疏约束在提高LLC的决策特征的可解释性方面非常有效。

5.4 LLCs的PBFs是否易于理解?

包体边界(PB)的**包体边界特征(PBF)**解释了为什么实例包含在LLC的凸包中。本小节中，我们研究了PBF的语义含义。受空间限制，我们仅使用FMNIST-1和FMINST-2训练的PLNN-NS模型作为需要解释的目标模型。PLNN-NS的LLC由*OpenBox*得出。

前文提到，PB 由线性不等式$z_i^{(l)} \in \psi(z_i^{(l)})$定义，PBF 是$z_i^{(l)}$中$x$的系数。由于激活函数是 ReLU，$z_i^{(l)} \in \psi(z_i^{(l)})$不是$z_i^{(l)} > 0$，就是$z_i^{(l)} \leq 0$。因为 PBF 值在 PLNN-NS 上保持非负，所以对凸包P_k，

若$z_i^{(l)} > 0$,则$P_{\not h}$中的图片强相关于$z_i^{(l)}$的 PBF;如果$z_i^{(l)} \leq 0$,则$P_{\not h}$中的图像不强相关于$z_i^{(l)}$的 PBF。

图 6：(a)-(d)表示FMNIST-1上PLNN-NS的PBF; (e)-(h)表示FMNIST-2上PLNN-NS的PBF

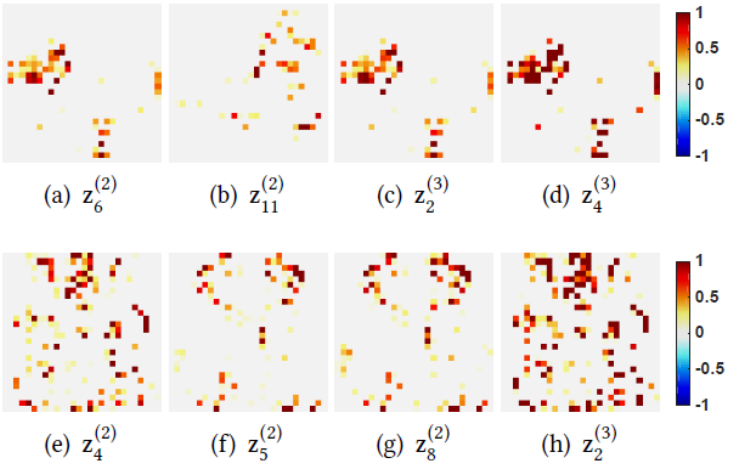


表 6：前3大凸包(CP)的PB包含了FMNIST-1中大多数实例
“/”表示无效线性不等式；准确率为每个CP上LLC的训练准确率

CP	$z_6^{(2)}$	$z_{11}^{(2)}$	$z_2^{(3)}$	$z_4^{(3)}$	短靴数量	背包数量	准确率
1	/	> 0	> 0	/	3991	3997	0.999
2	⩽ 0	> 0	/	⩽ 0	9	0	1.000
3	/	⩽ 0	/	> 0	0	3	1.000

表 7：前3大凸包(CP)的PB包含了FMNIST-2中大多数实例
准确率为每个CP上LLC的训练准确率

CP	$z_4^{(2)}$	$z_5^{(2)}$	$z_8^{(2)}$	$z_2^{(3)}$	外套数量	套头衫数量	准确率
1	> 0	> 0	> 0	> 0	3932	3942	0.894
2	> 0	⩽ 0	> 0	> 0	32	10	0.905
3	> 0	⩽ 0	⩽ 0	> 0	18	0	0.944

表 6、表 7 及图 6 中的数据证明了上述对 PB 和 PBF 的分析。以表 6 中的第一个凸包为例，其 PB 为$z_{11}^{(2)} > 0$和$z_2^{(3)} > 0$，分别对应图 6(b)-(c)中 PBF 的短靴和背包的特征。因此，凸包既包含了短靴的图片特征和也包含了背包的图片特征。其余实验结果也表明，凸包的 PBF 易于理解且能准确描述每个凸包中的图片。

我们还能发现图 6 中的 PBF 看起来类似于图 4 和图 5 中 PLNN-NS 的决策特征。这表明 PLNN-NS 的不同神经元学到的特征之间呈强相关，这可能是由网络的层次结构导致的。 由于神经元之间的强相关性， C中的状态数量——如表 3 所示——远少于k^N。

从表7我们能惊讶地发现，FMNIST-2上的第1大凸包内含超过98%的训练实例。在这些实例上，LLC的训练准确率远高于LR-NS和LR-NSF的训练准确率。这意味着第1大凸包中的训练实例比FMNIST-2中的训练实例更容易线性分离。这样看来，PLNN-NS的做法就像一个“分而治之”的策略：留下一小部分妨碍分类准确率的实例不管，这样大多数实例都可以被LLC更好地分开。比如表7中的第2大和第3大凸包，第1大凸包留存的实例被分在它们的凸包中，但凸包中

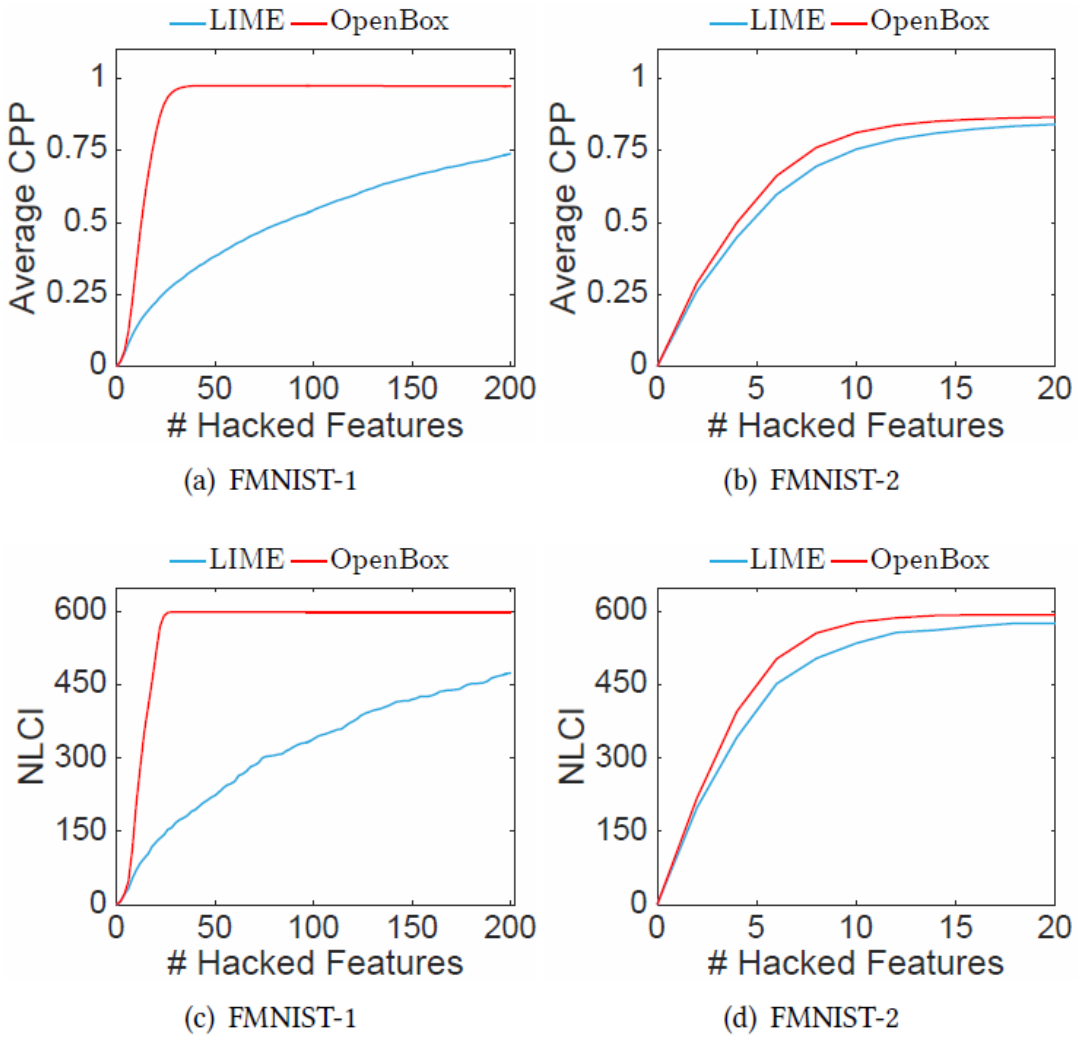
相应的LLC也达到了非常高的准确率，表6指出FMNIST-1上也有类似现象。不过由于FMNIST-1中的实例易于线性分离，因此PLNN-NS的训练准确率略高于LR-NS和LR-NSF。

5.5 能用*OpenBox*来破解模型吗?

了解智能机器的“想法”给了我们“破解”它的机会。这里是在尽可能不修改x的特征的情况下，调整目标模型来大幅改变其在实例x上的预测结果。我们通常通过改变权重最大的决策特征来大幅改变预测结果，但对目标模型更精确的解释能让我们更加切实地探索其重要的决策特征，因此要求尽量不改变实例特征而尽可能地改变预测结果。基于这个想法，我们使用LIME和*OpenBox*来hack这个PLNN-NS，并通过比较改变相同数量的决策特征时PLNN-NS预测结果变化来比较LIME和*OpenBox*得出的解释的质量。

对于实例$x \in \chi$，记$\gamma \in \mathbb{R}^d$为分类x的决策特征。我们通过将x中的一些权值最高的决策特征值置零来破解PLNN-NS，以使x上的PLNN-NS的预测结果大幅变化。给出以下两项指标评估预测的变化：第一个是**预测概率变化(CPP)**，它是将x分类为正例的概率的变化绝对值。第二个是**标签更改实例量(NLCI)**，这是在模型调整后其预测标签发生变化的实例个数。同样由于LIME的低效率，我们以5.2节中给出方式的采样数据集并进行评估。

图 7：LIME和*OpenBox*的调整性能；(a)-(b)表示平均CPP；(c)-(d)表示NLCI。



如图7，两个数据集上*OpenBox*的平均CPP和NLCI一致优于LIME。这说明*OpenBox*给出的解释在用来调整目标模型时比LIME更高效。有趣的是*OpenBox*的优势在FMNIST-1上比在FMNIST-2上更大。这是因为——如图2(a)所示——FMNIST-1中大多数实例的预测概率不是1.0就是0.0，这几乎没能为LIME提供多少梯度信息，使得它无法准地逼近PLNN-NS的分类函数。在这种情况下，LIME算得的决策特征无法描述目标模型的确切行为。

总结：*OpenBox*为目标模型给出了精确一致的解释，因此它能比LIME的更好地调整和破解模型。

5.6 我们能否使用*OpenBox*来为模型排错?

智能机器并不完美且偶尔会给出无效预测结果。当发生此类问题时，我们可以用*OpenBox*来解释实例被误分类的原因。

图 8：(a)外套(CO)，(d)套头衫(PU)，(g)短靴(AB)，和(j)背包(BG)的误分类的图像 (a)，(d)，(g)和(j)是原图；对于其他几张子图，标签给出了分为相应种类的预测概率; 图片标注了给出相应分类预测的决策特征依据

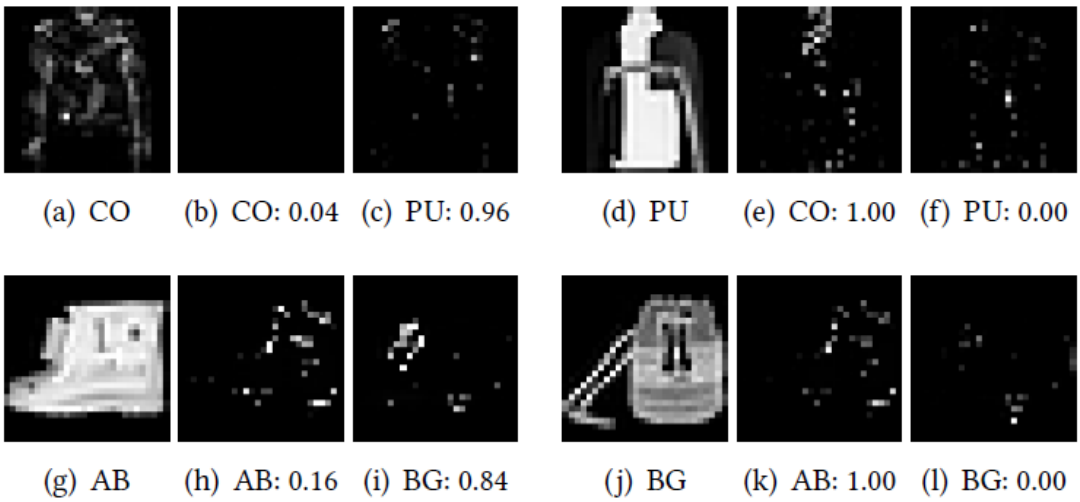


图8展示了一些由PLNN-NS以高概率误分类的图片。在图8(a)-(c)中原图是外套，然而由于布料上的分散的马赛克图案比套头衫更多地匹配了套头衫的特征，因此原图以高概率被分类为套头衫。在图8(d)-(f)中，原图是套头衫但被误分类为外套，因为领口和胸口匹配了外套的典型特征，而肩膀和袖子这里太暗于是错过了大多数套头衫的重要特征。同样图8(g)中的短靴因为标注了太多左上的棱角而被误分类为背包。图8(j)中的背包被误分类为短靴，是因为它匹配了短靴的脚踝和脚后跟这些特征却错过了左上的棱角这一背包的典型特征。

总结：*OpenBox*能准确地解释误分类，在已解读模型行为异常时对其排错有潜在帮助。

6 结论与今后的工作

在本文中，我们解决了解释PLNN这一充满挑战性的问题。通过研究隐神经元的状态和PLNN的结构，我们证明了PLNN在数学上等价于一组LLC，可以通过我们提出的*OpenBox*方法高效地算出这组LLC。大量实验表明，LLC的决策特征和PBF为PLNN的整体行为提供了精确一致的解释。

这种解释在改进和调试PLNN模型方面非常有效。未来的工作中，我们将深化现有的成果以解释如sigmoid函数和tanh函数这样使用了平滑激活函数的更一般神经网络。

参考文献

[1] Aishwarya Agrawal, Dhruv Batra, and Devi Parikh. 2016. Analyzing the behavior of visual question answering models. arXiv:1606.07356 (2016).

[2] Jimmy Ba and Rich Caruana. 2014. Do deep nets really need to be deep? . In NIPS. 2654 − 2662.

[3] Osbert Bastani, Carolyn Kim, and Hamsa Bastani. 2017. Interpreting Blackbox Models via Model Extraction. arXiv:1705.08504 (2017).

[4] C Bishop. 2007. Pattern Recognition and Machine Learning (Information Science and Statistics). Springer, New York (2007).

[5] C. Cao, X. Liu, Y Yang, Y. Yu, J. Wang, Z. Wang, Y. Huang, L. Wang, C. Huang, et al. 2015. Look and think twice: Capturing top-down visual attention with feedback convolutional neural networks. In ICCV. 2956 − 2964.

[6] RJ Caron, JF McDonald, and CM Ponic. 1989. A degenerate extreme point strategy for the classification of linear constraints as redundant or necessary. JOTA 62, 2(1989), 225 − 237.

[7] Z. Che, S. Purushotham, R. Khemani, and Y. Liu. 2015. Distilling knowledge from deep networks with applications to healthcare domain. arXiv:1512.03542 (2015).

[8] Jan Chorowski and Jacek M Zurada. 2015. Learning understandable neural networks with nonnegative weight constraints. TNNLS 26, 1 (2015), 62 − 69.

[9] Alexey Dosovitskiy and Thomas Brox. 2016. Inverting visual representations with convolutional networks. In CVPR. 4829 − 4837.

[10] D. Erhan, Yoshua Bengio, A. Courville, and P. Vincent. 2009. Visualizing higher layer features of a deep network. University of Montreal 1341 (2009), 3.

[11] Ruth Fong and Andrea Vedaldi. 2017. Interpretable Explanations of Black Boxes by Meaningful Perturbation. arXiv:1704.03296 (2017).

[12] Nicholas Frosst and Geoffrey Hinton. 2017. Distilling a Neural Network Into a Soft Decision Tree. arXiv:1711.09784 (2017).

[13] Amirata Ghorbani, Abubakar Abid, and James Zou. 2017. Interpretation of Neural Networks is Fragile. arXiv:1710.10547 (2017).

[14] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Deep sparse rectifier neural networks. In ICAIS. 315 − 323.

[15] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. Deep Learning. MIT Press. http://www.deeplearningbook.org.

[16] Ian J Goodfellow, DavidWarde-Farley, Mehdi Mirza, Aaron Courville, and Yoshua Bengio. 2013. Maxout networks. arXiv:1302.4389 (2013).

[17] B. Goodman and S. Flaxman. 2016. European Union regulations on algorithmic decision-making and a" right to explanation". arXiv:1606.08813 (2016).

[18] Nick Harvey, Chris Liaw, and Abbas Mehrabian. 2017. Nearly-tight VC-dimension bounds for piecewise linear neural networks. arXiv:1703.02930 (2017).

[19] K. He, X. Zhang, S. Ren, and J. Sun. 2015. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In ICCV. 1026 − 1034.

[20] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. arXiv:1503.02531 (2015).

[21] Patrik O Hoyer. 2002. Non-negative sparse coding. In WNNSP. 557 − 565.

[22] Pieter-Jan Kindermans, Sara Hooker, Julius Adebayo, Maximilian Alber, Kristof T Schütt, Sven D 鋒 ne, Dumitru Erhan, and Been Kim. 2017. The (Un) reliability of saliency methods. arXiv:1711.00867 (2017).
[23] Pang Wei Koh and Percy Liang. 2017. Understanding black-box predictions via influence functions. arXiv:1703.04730 (2017).
[24] Pascal Koiran and Eduardo D Sontag. 1996. Neural networks with quadratic VC dimension. In NIPS. 197 − 203.
[25] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In NIPS. 1097 − 1105.
[26] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. Nature 521, 7553 (2015), 436.
[27] Honglak Lee, Alexis Battle, Rajat Raina, and Andrew Y Ng. 2007. Efficient sparse coding algorithms. In NIPS. 801 − 808.
[28] Jiwei Li, Xinlei Chen, Eduard Hovy, and Dan Jurafsky. 2015. Visualizing and understanding neural models in NLP. arXiv:1506.01066 (2015).
[29] Aravindh Mahendran and Andrea Vedaldi. 2015. Understanding deep image representations by inverting them. In CVPR. 5188 − 5196.
[30] Guido F Montufar, Razvan Pascanu, Kyunghyun Cho, and Yoshua Bengio. 2014. On the number of linear regions of deep neural networks. In NIPS. 2924 − 2932.
[31] Vinod Nair and Geoffrey E Hinton. 2010. Rectified linear units improve restricted boltzmann machines. In ICML. 807 − 814.
[32] R. B. Palm. 2012. Prediction as a candidate for learning deep hierarchical models of data.
[33] Razvan Pascanu, Guido Montufar, and Yoshua Bengio. 2013. On the number of response regions of deep feed forward networks with piece-wise linear activations. arXiv:1312.6098 (2013).
[34] Nadeem N Rather, Chintan O Patel, and Sharib A Khan. 2017. Using Deep Learning Towards Biomedical Knowledge Discovery. IJMSC 3, 2 (2017), 1.
[35] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. Why should I trust you?: Explaining the predictions of any classifier. In KDD. ACM, 1135 − 1144.
[36] R. R Selvaraju, A. Das, R. Vedantam, M. Cogswell, D. Parikh, and D. Batra. 2016. Grad-cam: Why did you say that? visual explanations from deep networks via gradient-based localization. arXiv:1610.02391 (2016).
[37] A. Shrikumar, P. Greenside, and A. Kundaje. 2017. Learning important features through propagating activation differences. arXiv:1704.02685 (2017).
[38] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. 2013. Deep inside convolutional networks: Visualising image classification models and saliency maps. arXiv:1312.6034 (2013).
[39] D. Smilkov, N. Thorat, B. Kim, F. Viégas, and M. Wattenberg. 2017. SmoothGrad: removing noise by adding noise. arXiv:1706.03825 (2017).
[40] Eduardo D Sontag. 1998. VC dimension of neural networks. NATO ASI Series F Computer and Systems Sciences 168 (1998), 69 − 96.
[41] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. 2017. Axiomatic Attribution for Deep Networks. arXiv:1703.01365 (2017).
[42] M. Wu, M. C Hughes, S. Parbhoo, M. Zazzi, V. Roth, and F. Doshi-Velez. 2018. Beyond Sparsity: Tree Regularization of Deep Models for Interpretability. AAAI(2018).
[43] Han Xiao, Kashif Rasul, and Roland Vollgraf. 2017. Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms. arXiv:cs.LG/cs.LG/1708.07747
[44] J. Yosinski, J. Clune, A. Nguyen, T. Fuchs, and H. Lipson. 2015. Understanding neural networks through

deep visualization. arXiv:1506.06579 (2015).
[45] Rich Zemel, Yu Wu, Kevin Swersky, Toni Pitassi, and Cynthia Dwork. 2013. Learning fair representations. In ICML. 325 – 333.
[46] Bolei Zhou, David Bau, Aude Oliva, and Antonio Torralba. 2017. Interpreting Deep Visual Representations via Network Dissection. arXiv:1711.05611 (2017).
[47] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. 2016. Learning deep features for discriminative localization. In CVPR. 2921 – 2929.
[48] J. Zhu, Y. Shan, JC Mao,D. Yu, H. Rahmanian, and Y. Zhang. 2017. Deep embedding forest: Forest-based serving with deep embedding features. In KDD. 1703 – 1711.

译注术语表：

exact	精确，正确，确切
exactness	精确性
consistency	一致性
configuration	状态
convex polytope	凸包，凸多边形
polytope	包体，多边形
fixed	恒定，固定，常值
interpretation	解释，翻译
constant	常值的，恒定的，不变的
close form	解析式，封闭形式
hack	改进，破解
method	方法
dominated by	由…确定
define	定义了…
prediction	预测值
mimic model	模仿者模型
active	活动的，使用中的
overall	整体的
approximate	逼近，近似
decision	决策