A probability density function generator based on neural networks

Chi-Hua Chen, Fangying Song, Feng-Jang Hwang, Ling Wu

摘要

为了生成用于拟合实际数据的概率分布的概率密度函数 (PDF)，本研究提出了一种深度学习方法，该方法包括两个阶段：(1) 用于估计累积分布函数 (CDF) 的训练阶段和 (2) 用于预测相应的 PDF 的执行阶段。公共概率分布的 CDF 可以用作所提出的深度学习模型的隐藏层中的激活函数，用于学习实际累积概率，并且可以使用所训练的深度学习模式的微分方程来估计 PDF。进行了单分布和混合分布的数值实验，以评估所提出方法的性能。实验结果表明，该方法可以准确地估计 CDF 和 PDF 的值。

关键词：概率密度函数；累积密度函数；神经网络

1 引言

为了解释趋势或现象，统计工具和概率模型通常用于分析实际数据。各种常见的概率分布 (例如指数分布 (ED)、正态分布 (ND)、对数正态 (LD)、伽马分布 (GD) 等) 被广泛用作关于实际数据分布的假设。然而，实际数据的 CDF 和 PDF 可能是复杂且不规则的，因此简单的假设会导致严重的错误。这项研究提出了一种深度学习方法[1]，该方法开发了一个神经网络来学习实际数据的 CDF 并拟合相应的 PDF。在训练阶段，收集实际数据的 CDF，并训练所提出的具有设计的激活函数的深度神经网络模型以学习 CDF。在执行阶段，对训练的深度神经网络模型中的 CDF 进行微分，以估计各种基于概率的应用的实际数据的 PDF。

2 相关工作和文献综述

论文的主要工作包括：

（1）多个概率函数的组合作为所提出的深度神经网络模型中的激活函数，用于学习 CDF。

（2）推导并评估所提出的基于梯度下降的深度神经网络模型中每个神经元的权重更新。

（3）对相应的训练深度神经网络模型执行 CDF 微分，获得各种基于概率的应用中的实际数据的 PDF。

2.1 传统激活函数

近年来，包括线性函数 (即方程 (1) 中的 a_L)[2]、校正线性单元 (ReLU) 函数 (即方程 (2) 中的 a_R)[3]和 Sigmoid 函数 (即方程 (3) 中的 a_S)[4]的激活函数已广泛应用于神经网络模型中。线性函数的图形是一条直线，因此线性函数不能表示曲线或双曲线。ReLU 函数采用输入的正部分，输入的负部分表示为零，以改进线性函数。Sigmoid 函数可用于表示非线性问题分析的曲线或双曲线。

$$a_L(z) = z \tag{1}$$

$$a_R(z) = \begin{cases} z, & if z > 0 \\ 0, & otherwise \end{cases} \tag{2}$$

$$a_S(z) = \frac{1}{1 + e^{-z}} \tag{3}$$

2.2 神经网络

用于估计概率分布的神经网络模型如图 1所示。此神经网络示意图具有一个输入层、两个隐藏层和一个输出层。输入层中包括一个参数 (即参数 x)，输出层中包括另一个参数。每个隐藏层有 n 个神经元；例如，参数 $g_{k,j}$，j 表示第 k 层中的第 j 个神经元 (如等式 (4) 所示)，参数 $w_{k,i,j}$ 表示第 (k-1) 层中的第 i 个神经元与第 k 层中的第 j 个神经元之间的权重值。表示神经元激活功能的函数 a() 可以是线性函数、ReLU 函数或 Sigmoid 函数。根据神经网络输出的值 (即参数 F) 分析最小二乘误差，以优化构建的神经网络。在神经网络中，参数和权重可以通过梯度公式进行更新。

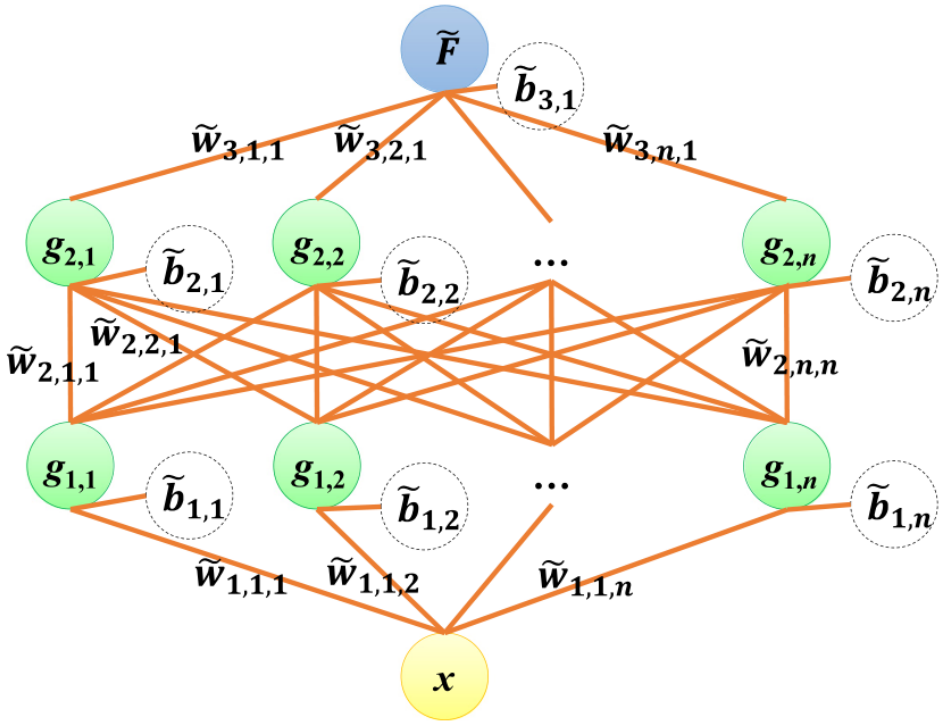

图 1: 估计概率分布的神经网络方法

$$\begin{aligned} & g_{k,j} = a(z_{k,j}), \\ & where \quad z_{k,j} = \sum_{i=1}^{n} \tilde{w}_{k,i,j} \times g_{k-1,j} + \tilde{b}_{k,j} \end{aligned} \tag{4}$$

当用神经网络学习数据集的 CDF 时，计算出的 CDF 的值被记录为参数 F。因此，可利用已训练好的神经网络，对 CDF 进行微分来估计数据集的 PDF(如等式 (5) 所示)。激活函数可以设置为线性函数、ReLU 函数或 Sigmoid 函数[5]，但是进行微分推导后会发现：如若使用线性函数、ReLU 函数作为激活函数，会导致对 CDF 微分后得到的 PDF 的计算公式，最后会缺少参数 x，其仅包括常数。本论文将使用多种概率函数的组合作为激活函数来建立神经网络模型。

$$\begin{aligned}\frac{\partial \tilde{F}}{\partial x} &= \frac{\partial \tilde{F}}{\partial g_{3,1}}\frac{\partial g_{3,1}}{\partial z_{3,1}}\sum_{i=1}^{n}\frac{\partial z_{3,1}}{\partial g_{2,i}}\frac{\partial g_{2,i}}{\partial z_{2,i}}\sum_{j=1}^{n}\frac{\partial z_{2,i}}{\partial g_{1,j}}\frac{\partial g_{1,j}}{\partial z_{1,j}}\frac{\partial z_{1,j}}{\partial x} \\ &= 1 \times \frac{\partial g_{3,1}}{\partial z_{3,1}} \times \sum_{i=1}^{n}\tilde{w}_{3,i,1} \times \frac{\partial g_{2,i}}{\partial z_{2,i}} \times \sum_{j=1}^{n}\tilde{w}_{2,j,i} \times \frac{\partial g_{1,j}}{\partial z_{1,j}}\tilde{w}_{1,1,j}\end{aligned} \tag{5}$$

3 本文方法

3.1 神经网络结构概述

在训练阶段，收集分析实际分布的数据，以生成输出累积概率。本文提出的神经网络构建了保留输入层、单层隐藏层和输出层的模型，根据输出的累积概率来微分生成 CDF(如图 2所示)。输入层包含随机变量 x，其实际 CDF 由 F 表示，输出层包含随机变量 x 的估计 CDF，由 $\tilde{F}$ 表示。用常见概率分布的累积分布函数作为隐藏层中的激活函数：例如 ED、ND 和 LD 的累积分布函数 (分别由 f_1、f_2 和 f_3 表示) 可用作激活函数 (分别如等式 (6)-(8) 所示，其中 $\tilde{\lambda}$、$\tilde{\mu}_1$、$\tilde{s}_1$、$\tilde{\mu}_2$、$\tilde{s}_2$ 均为参数)。

此外，具有不同激活功能的神经元的数量可以扩展到 n 个，并且 CDF 可以根据神经网络中的权重集 (即$\tilde{w}_1$，$\tilde{w}_2$，⋯，$\tilde{w}_n$) 通过等式 (9) 来估计。

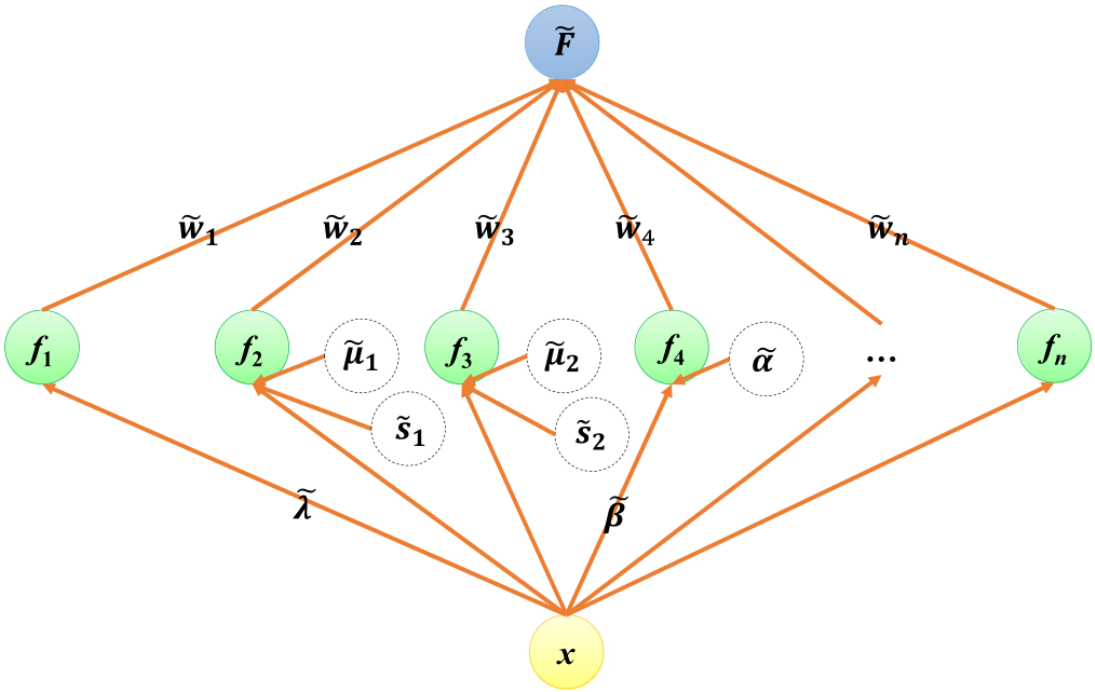

图 2: 文中所提出学习模型的结构

$$f_1 = 1 - e^{-\tilde{\lambda}x} \tag{6}$$

$$f_2 = \frac{1}{2}\left[1 + \text{erf}\left(\frac{x - \tilde{\mu}_1}{\tilde{s}_1\sqrt{2}}\right)\right] \tag{7}$$

$$f_3 = \frac{1}{2}\left[1 + \text{erf}\left(\frac{\ln(x) - \tilde{\mu}_2}{\tilde{s}_2\sqrt{2}}\right)\right] \tag{8}$$

$$\tilde{F} = \frac{\sum_{i=1}^{n} \tilde{w}_i f_i}{\sum_{i=1}^{n} \tilde{w}_i} \tag{9}$$

3.2 损失函数和梯度计算

所提出的深度学习模型的损失函数 E 由等式 (10) 定义，根据利用神经网络模型估计的 CDF(即 $\tilde{F}$) 和数据集中的实际 CDF(即 F) 来最小化估计误差。每个要学习的参数都使用梯度下降。计算损失函数 E 相对于每个参数的偏微分 (如等式 (11)-(16))，第 (k+1) 次迭代中的参数可以根据第 k 次迭代中的偏微分进行更新。可以增加隐藏层的数量来估计相对复杂的 CDF。

$$E = \frac{1}{2}(\tilde{F} - F)^2 = \frac{1}{2}\sigma^2 \tag{10}$$

$$\frac{\partial E}{\partial \tilde{w}_j} = \frac{\partial E}{\partial \sigma}\frac{\partial \sigma}{\partial \tilde{F}}\frac{\partial \tilde{F}}{\partial \tilde{w}_j} = \sigma \times \frac{\sum_{i=1}^{n}\left[\tilde{w}_i\left(f_j - f_i\right)\right]}{\left(\sum_{i=1}^{n}\tilde{w}_i\right)^2} \tag{11}$$

$$\frac{\partial E}{\partial \tilde{\lambda}} = \frac{\partial E}{\partial \sigma}\frac{\partial \sigma}{\partial \tilde{F}}\frac{\partial \tilde{F}}{\partial f_1}\frac{\partial f_1}{\partial \tilde{\lambda}} = \sigma \times \frac{\tilde{w}_1}{\sum_{i=1}^{n}\tilde{w}_i} \times \left(xe^{-\tilde{\lambda}x}\right) \tag{12}$$

$$\frac{\partial E}{\partial \tilde{\mu}_1} = \frac{\partial E}{\partial \sigma}\frac{\partial \sigma}{\partial \tilde{F}}\frac{\partial \tilde{F}}{\partial f_2}\frac{\partial f_2}{\partial \tilde{\mu}_1} = \sigma \times \frac{\tilde{w}_2}{\sum_{i=1}^{n}\tilde{w}_i} \times \left(\frac{-1}{\sqrt{2\pi\tilde{s}_1^2}}e^{-\frac{\left(x-\tilde{\mu}_1\right)^2}{2\tilde{s}_1^2}}\right) \tag{13}$$

$$\frac{\partial E}{\partial \tilde{s}_1} = \frac{\partial E}{\partial \sigma}\frac{\partial \sigma}{\partial \tilde{F}}\frac{\partial \tilde{F}}{\partial f_2}\frac{\partial f_2}{\partial \tilde{s}_1} = \sigma \times \frac{\tilde{w}_2}{\sum_{i=1}^{n}\tilde{w}_i} \times \left(\frac{-\left(x-\tilde{\mu}_1\right)}{\sqrt{2\pi\tilde{s}_1^2}}e^{-\frac{\left(x-\tilde{\mu}_1\right)^2}{2\tilde{s}_1^2}}\right) \tag{14}$$

$$\frac{\partial E}{\partial \tilde{\mu}_2} = \frac{\partial E}{\partial \sigma}\frac{\partial \sigma}{\partial \tilde{F}}\frac{\partial \tilde{F}}{\partial f_3}\frac{\partial f_3}{\partial \tilde{\mu}_2} = \sigma \times \frac{\tilde{w}_3}{\sum_{i=1}^{n}\tilde{w}_i} \times \left(\frac{-1}{\sqrt{2\pi\tilde{s}_2^2}}e^{-\frac{\left(\ln(x)-\tilde{\mu}_2\right)^2}{2\tilde{s}_2^2}}\right) \tag{15}$$

$$\frac{\partial E}{\partial \tilde{s}_2} = \frac{\partial E}{\partial \sigma}\frac{\partial \sigma}{\partial \tilde{F}}\frac{\partial \tilde{F}}{\partial f_3}\frac{\partial f_3}{\partial \tilde{s}_2} = \sigma \times \frac{\tilde{w}_3}{\sum_{i=1}^{n}\tilde{w}_i} \times \left(\frac{-\left(\ln(x)-\tilde{\mu}_2\right)}{\sqrt{2\pi\tilde{s}_2^2}}e^{-\frac{\left(\ln(x)-\tilde{\mu}_2\right)^2}{2\tilde{s}_2^2}}\right) \tag{16}$$

神经网络在经历过训练阶段后，利用等式 (17) 描述的训练深度学习模型的微分方程，将 $\tilde{F}$ 对 x 进行偏微分，以计算获得用神经网络估计的 CDF 求得的 PDF(即 $\tilde{P}$)。

$$\begin{aligned}\tilde{P} = \frac{\partial \tilde{F}}{\partial x} &= \sum_{i=1}^{n}\left(\tilde{w}_i \times \frac{\partial f_i}{\partial x}\right) \\ &= \tilde{w}_1 \times \tilde{\lambda}e^{-\tilde{\lambda}x} + \tilde{w}_2 \times \frac{1}{\sqrt{2\pi\tilde{s}_1^2}}e^{-\frac{\left(x-\tilde{\mu}_1\right)^2}{2\tilde{s}_1^2}} \\ &\quad + \tilde{w}_3 \times \frac{1}{x\tilde{s}_2\sqrt{2\pi}}e^{-\frac{\left(\ln(x)-\tilde{\mu}_2\right)^2}{2\tilde{s}_2^2}} + \sum_{i=4}^{n}\left(\tilde{w}_i \times \frac{\partial f_i}{\partial x}\right)\end{aligned} \tag{17}$$

3.3 验证其有效性

在验证有效性部分，论文中设置了不同的测试用例，包括单个 ED、单个 ND、单个 LD、单个 GD 以及具有不同参数的单个 ND 和单个 GD 的两个混合分布 (MD)。为了评估所提出的方法，根据 $\tilde{F}$ 和 F，估计的 CDF(即 M_F) 的平均绝对百分比误差 (MAPE) 定义为等式 (18)，根据 $\tilde{P}$ 和 P，估计的 PDF(即 M_P) 的 MAPE 定义为等式 (19)。

（1）参数 λ=0.5 的单个指数分布；

（2）参数 σ=1 和 μ=0.5 的单个正态分布；

（3）参数 σ=1 和 μ=0.5 的单个对数正态分布；

（4）参数 α=0.1 和 β=0.5 的单个伽马分布；

（5）参数 σ=1、μ=0.5、α=0.1 和 η=0.5 的单个正态分布和单个伽马分布的混合分布；

（6）参数 σ=1、μ=0.5、α=2 和 β=10 的单个正态分布和单个伽马分布的混合分布。

$$M_F = \sum_{k=1}^{m} \frac{\left| F_k - \tilde{F}_k \right|}{F_k} \times \frac{100\%}{m} \tag{18}$$

$$M_P = \sum_{k=1}^{m} \frac{\left| P_k - \tilde{P}_k \right|}{P_k} \times \frac{100\%}{m} \tag{19}$$

在验证有效性的实验中，设置样本数 m 的值分别为 2、5、10、20、30 和 40。使用 t 折交叉验证来评估所提出的方法并验证过拟合问题，t 值设置为 5。样本随机分布在五个组中。在交叉验证评估中，选择一组作为验证数据，其他四组作为训练数据。计算了在每种情况下 MAPE 的值的大小，MAPE 样本数量较大时 MAPE 的收敛速度更快。当 m 值为 40 时，每种情况下 M_F 和 M_P 的值都约为 0。样本数量太小 (例如 m ＝ 2 和 m ＝ 5) 时，在估算复杂分布时的误差会较大。

为了比较使用神经网络估计的 CDF 和 PDF 的误差情况，论文还选择了分别基于线性函数、ReLU 函数和 Sigmoid 函数的神经网路。样本数量 m 都设置为 40，迭代 300 次。在基于线性函数和 ReLU 函数的神经网络中，由于训练后的神经网络模型进行微分后缺少参数 x，所以 MAPE 值很大。虽然使用 Sigmoid 函数作为激活函数的神经网络 MAPE 值较小，但它需要进行更多次迭代。而使用论文所提出的方法可以准确地估计 CDF 和 PDF，MAPE 值不会超过 0.4%。

4 复现细节

4.1 与已有开源代码对比

本论文没有任何相关开源代码，也未引用他人代码。

本次复现在阅读论文的基础上复现了大部分实验，包括论文中神经网络的搭建、三种情况的验证测试以及与其他神经网络的性能比较。论文所提出的神经网络在作者另一篇论文 [] 中有所使用。在搭建好神经网络后，生成三组论文中提到的数据集：指数分布、正态分布、伽马分布。首先是第一部分的实验，利用 5 折交叉验证分别对网络进行训练以及验证。所得结果和论文中提到的相匹配。根据利用该神经网络回归生成的概率密度函数和原概率密度函数的图像进行比较，不论是走势还是数值的误差率都很低。其次是第二部分的实验，将同样的数据集利用激活函数均为 Sigmoid 函数的神经网络进

行回归拟合，其生成的概率密度函数的图像虽然走势和原概率密度函数的类似，但是有较大的误差，数值的误差更大。

4.2 实验环境搭建

软件版本：Spyder(Python 3.9) 64bit

.py 文件直接在 python 编译器中使用即可

4.3 部分代码

复现的代码主要分为两大部分。第一部分是原论文神经网络的实现，包括生成数据集、在神经网络构建好后用数据集进行测试验证；第二部分是论文提出的方法与其他方法的比较情况，由于在论文中使用线性函数和 ReLU 函数作为激活函数时的误差较大，复现内容仅和将 Sigmoid 函数作为激活函数的情况进行了比较。报告将在本小节对复现的部分代码进行汇报。

4.3.1 数据集

论文复现一共选取了三种分布的数据集：

（1）参数 λ=0.5 的单个指数分布；

（2）参数 σ=1 和 μ=0.5 的单个正态分布；

（3）参数 α=1 和 β=2 的单个伽马分布。

根据对论文的了解，其误差函数的定义是均方误差，本质是回归神经网络。由于没有论文原本的数据集，所以自己生成了相应分布的数据。原论文中所提到，对每组数据集的个数分别为 2、5、10、20、30、40 进行了实验，当数据个数为 40 个的时候收敛小果子最好，因此在复现中直接选择生成包含 40 组数据的数据集。生成数据集的代码具体如下图 3所示。

```
#指数分布
x1 = np.linspace(expon.ppf(0.01,loc=0, scale=1/0.5),expon.ppf(0.99,loc=0, scale=1/0.5), 40)
F1 = expon.cdf(x1, 0.5)
P1 = expon.pdf(x1, 0.5)
#正态分布
data_ND = np.random.normal(0.5, 1, 40)
x2 = np.linspace(norm.ppf(0.6,loc=np.mean(data_ND), scale=np.std(data_ND)),
                 norm.ppf(0.99,loc=np.mean(data_ND), scale=np.std(data_ND)),
                 len(data_ND))
F2 = norm.cdf(x2 ,loc=np.mean(data_ND), scale=np.std(data_ND))
P2 = norm.pdf(x2, loc=np.mean(data_ND), scale=np.std(data_ND))
#伽马分布
x3 = np.arange(0.01,10,0.25)
F3 = gamma.cdf(x3,1,scale=2)
P3 = gamma.pdf(x3,1,scale=2)
```

图 3: 数据集生成代码

4.3.2 激活函数的定义

寻常的神经网络使用的常见的激活函数是线性函数、ReLU 函数或者 Sigmoid 函数。在本论文中，神经网络主要是用来回归拟合累积分布函数并求得相应的概率密度，所以提出了将激活函数换为常见的分布的累积分布函数，例如指数分布、正态分布、对数正态分布、伽马分布等分布的累积分布函数，可以根据要估计的分布的复杂情况来选择隐藏层中激活函数的单元个数。本次复现的数据集并不复杂，所以只选择了指数分布、正态分布、对数正态分布三者作为隐藏层的神经元，其转化为代码如下图 4所示。三个激活函数的参数 x 均表示神经网络输入层 (即数据集的随机变量 x) 的值，lambda_ 是指数分布的参数，m_1、s_1 分别是正态分布的均值和标准差，mu_2、s_2 分别是对数正态分布的均值和标准差。每个激活函数都将该神经元的输出返回，以便于后续训练神经网络的时候进行输出计算。每

个参数都需要进行学习，都采用梯度下降的方法。

```
#定义三个激活函数
def f1_ED(x,lambda_):
    return 1 - np.exp(-x * lambda_)

def f2_ND(x,mu_1,s_1):
    layer_2 = (1 + math.erf((x-mu_1)/(math.sqrt(2)*s_1)))/2
    return layer_2

def f3_LD(x,mu_2,s_2):
    layer_3 = (1 + math.erf((math.log(x,math.e)-mu_2)/(math.sqrt(2)*s_2)))/2
    return layer_3
```

图 4: 激活函数定义代码

4.3.3 神经网络训练

论文中提到使用 K 折交叉验证来训练验证神经网络的有效性，并将 K 值设置为 5，在复现代码中直接将数据集进行划分成五部分，其中四组作为训练集，另外一组作为测试集。原论文使用的随机划分，由于数据集在生成的时候是使用的随机生成，所以在进行划分时没使用随机划分，而是分层划分。论文中的实验的迭代次数包括 50、100、150、200、250、300，在三组实验中迭代次数为 150 时基本上已经收敛好，但是为了保证实验结果的精准性，复现代码中采用迭代 300 次对神经网络进行训练。其数据集划分和迭代次数的代码体现如下图 5所示。

```
kf = KFold(n_splits = 5, shuffle=True, random_state=0)    # 5折
for train_index, test_index in kf.split(x3,F3,P3):     #将数据划分为k折
    train_x = x3[train_index]   # 选取的训练集数据下标
    train_F = F3[train_index]
    train_P = P3[train_index]
    test_x = x3[test_index]     # 选取的测试集数据下标
    test_F = F3[test_index]
    test_P = P3[test_index]
    #五折交叉验证的训练部分
    for k in range(300):
        CDF = []
        PDF = []
        for index in range(len(train_x)):
            #print(index)
            x = train_x[index]
            F = train_F[index]
```

图 5: 数据集划分代码

在训练神经网络的代码部分，由于数据量较小，迭代次数也不高，所以是使用正向传播进行参数更新。由于代码篇幅较长，不适合粘贴在报告中，在报告中进对其主要步骤进行概述，详细代码见所提交的源码。其主要步骤包括如下几个方面：

（1）根据数据集的随机变量 x 和各个参数的初始值，分别计算每个激活函数的输出；

（2）根据激活函数的输出和神经网络权值的初始值，利用向量的点乘计算出神经网络的输出 (即估计的随机变量 x 对应的累计分布函数的值 $\tilde{F}$，再根据概率密度的微分计算公式计算出随机变量 x 所对应的概率密度的值 $\tilde{P}$；

（3）根据数据集对应的真实值和神经网络的估计值定义出估算误差；

（4）根据估算误差计算各个参数包括神经网络的权重、各个激活函数需要学习的参数的梯度，再用较小的学习率对参数进行一次更新，完成一次正向传播。

4.3.4 测试验证

对论文中的测试验证部分，复现主要是将 K 折交叉验证的测试集在已经过训练的神经网络上进行测试，计算出测试集的估计 CDP 和 PDF 的值，并根据等式 (18) 和等式 (19) 计算其平均绝对百分比

误差大小。验证测试部分的代码如下图 6所示。

```
    error_F = 0
    error_P = 0
    for i in range(len(test_x)):
        x = test_x[i]
        F = test_F[i]
        P = test_P[i]
        hidden_layer1 = f1_ED(x,lambda_)
        hidden_layer2 = f2_ND(x,mu_1,s_1)
        hidden_layer3 = f3_LD(x,mu_2,s_2)
        hidden_layer = np.array([hidden_layer1,hidden_layer2,hidden_layer3])
        synapse_sum = np.sum(np.fromiter(synapse, float))
        F_ = (np.dot(hidden_layer, synapse))/(synapse_sum)
        P_ = synapse[0]*(lambda_*np.exp(-lambda_*x))
        + synapse[1]*(1/(math.sqrt(2*math.pi*(s_1**2))))*(np.exp(-((x-mu_1)**2)/(2*(s_1**2))))
        + synapse[2]*(1/(x*s_2*math.sqrt(2*math.pi)))*(np.exp(-(math.log(x,math.e)-mu_2)**2/(2*(s_2**2))))
        error_F = error_F + (abs(F - F_))/F
        error_P = error_F + (abs(P - P_))/P
    #error_F = (abs(F - F_))/F
    #error_P = (abs(P - P_))/P
    M_f = M_f + error_F/len(test_x)/100
    M_p = M_p + error_P/len(test_x)/100
print(M_f/5)
print(M_p/5)
```

图 6: 验证测试代码

5 实验结果分析

5.1 指数分布

复现实验中拟合了 λ=0.5 的指数分布，数据集包含 40 组数据，在神经网路中梯度下降迭代 300 次。经过神经网络拟合出来的 PDF 图像和真实数据的图像如下图 7所示。左侧为真实数据的图像，右侧为复现拟合的图像。

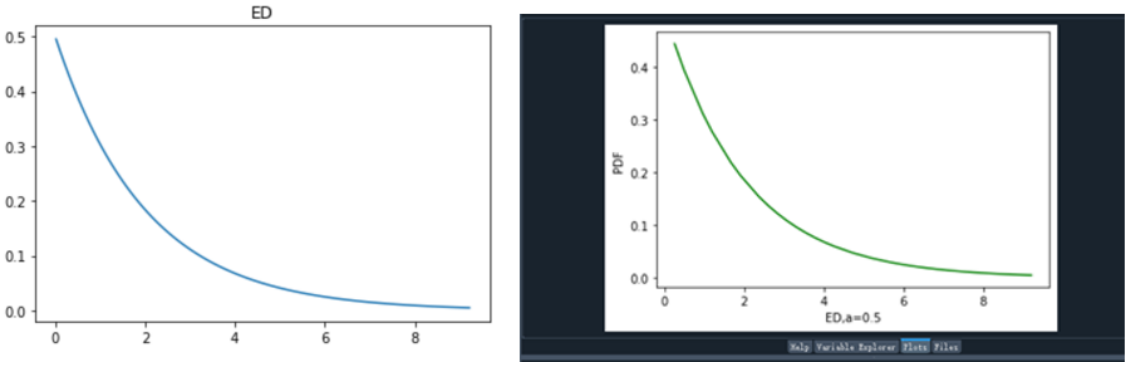

图 7: 指数分布 PDF 图像对比

在复现构建的神经网络中进行了五次实验，计算了每次实验的 CDP 和 PDF 的 MAPE 值，和原论文的相比较，虽然相差较小，但是其拟合效果从数据结果上来看没有原论文的效果好。五次实验和原论文的数据如下表 1所示。

表 1: 指数分布的五次实验 MAPE 值

	复现 1	复现 2	复现 3	复现 4	复现 5	论文
CDF	0.38%	0.59%	0.53%	0.89%	0.37%	接近 0%
PDF	0.16%	0.07%	0.08%	0.03%	0.05%	接近 0%

5.2 正态分布

复现实验中拟合了 σ=1 和 μ=0.5 的正态分布，数据集包含 40 组数据，在神经网路中梯度下降迭代 300 次。经过神经网络拟合出来的 PDF 图像和真实数据的图像如下图 8所示。左侧为真实数据的图像，右侧为复现拟合的图像。由于每次实验的实验数据是随机生成的，所以生成的图像不会每次都一样，稳定度较差，报告中截取了第一次实验的。细看对比图象会发现其走势还是有所区别，我认为其

主要原因是在原论文提出的方法中，当激活函数包含对数正态分布时，其输入层的随机变量 x 的值不能为负数，所以在选取实验数据集的时候会有所偏差。由于论文设计该神经网络的主要目的就是应用于交通研究中，其车流量为负数，所以没有考虑到随机变量为 x 的情况。

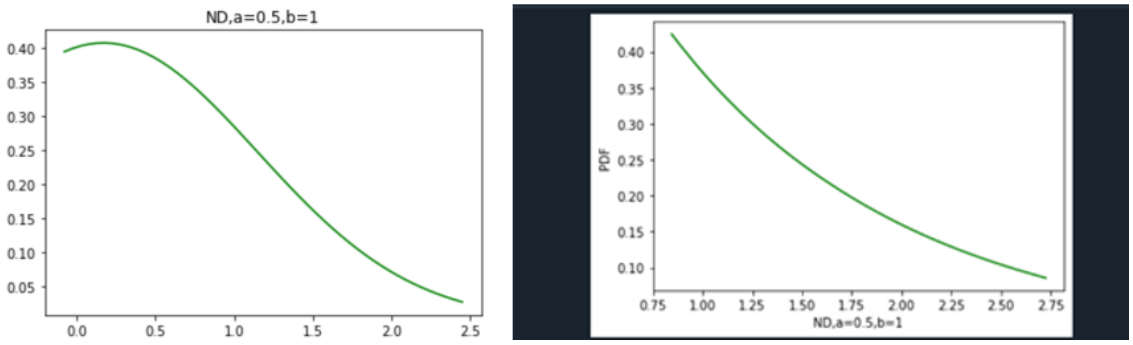

图 8: 正态分布 PDF 图像对比

在复现构建的神经网络中进行了五次实验，计算了每次实验的 CDP 和 PDF 的 MAPE 值，和原论文的相比较，CDF 的偏差比较大，PDF 的 MAPE 值和原论文差不多。五次实验和原论文的数据如下表 2所示。

表 2: 正态分布的五次实验 MAPE 值

	复现 1	复现 2	复现 3	复现 4	复现 5	论文
CDF	1.29%	0.84%	1.03%	1.06%	1.36%	接近 0%
PDF	0.22%	0.03%	0.04%	0.05%	0.07%	接近 0%

5.3 伽马分布

复现实验中拟合了 α=1 和 β=2 的伽马分布，数据集包含 40 组数据，在神经网路中梯度下降迭代 300 次。经过神经网络拟合出来的 PDF 图像和真实数据的图像如下图 9所示。左侧为真实数据的图像，右侧为复现拟合的图像。其 PDF 图像的走势大致相同，数值上偏差较小。

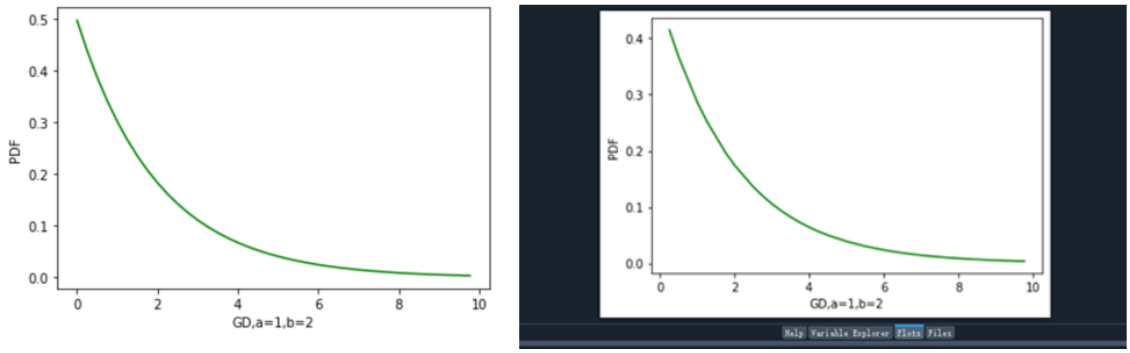

图 9: 伽马分布 PDF 图像对比

在神经网络中进行了五次实验，计算了每次实验的 CDP 和 PDF 的 MAPE 值。在伽马分布的实验中，CDF 比 PDF 的误差要小，PDF 的误差不稳定，有时仅有 0.08%，有时却能高达 1.70%。五次实验和原论文的数据如下表 3所示。

表 3: 伽马分布的五次实验 MAPE 值

	复现 1	复现 2	复现 3	复现 4	复现 5	论文
CDF	0.38%	0.46%	0.50%	0.51%	0.65%	接近 0%
PDF	0.15%	1.04%	0.99%	1.70%	0.08%	接近 0%

5.4 和 Sigmoid 比较

在原论文中，作者还将激活函数换成最为常用的线性函数、ReLU 函数和 Sigmoid 函数。用同样的数据集在更换激活函数后的神经网络上进行训练得出其 CDF 和 PDF 的 MAPE 值大小，并于论文所提出来的神经网络进行比较。在复现代码中，仅将激活函数全部更改为 Sigmoid 函数再进行训练，其在上述三种分布中，其拟合出来的 PDF 函数图像如下图 10所示。不论从走势还是从数据上来看，误差都比论文所提出的方法要大。

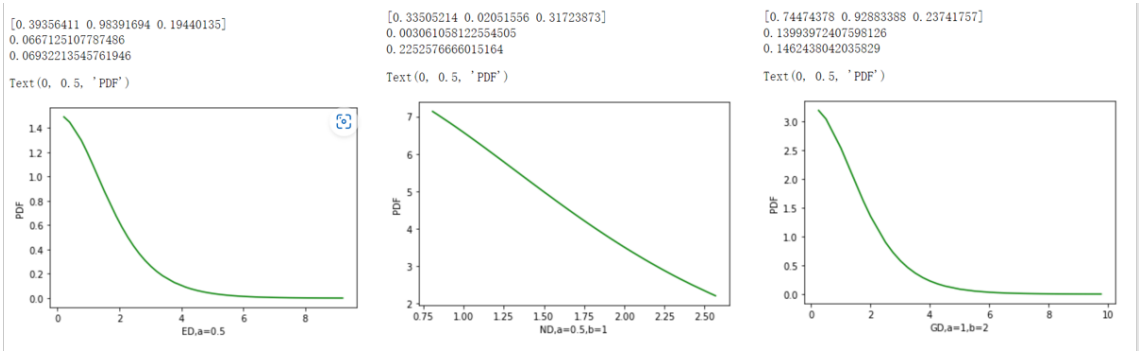

图 10: Sigmoid 激活函数的 PDF 拟合图像

6 总结与展望

对于论文中所提出的神经网络，其激活函数采用常见的分布的概率密度函数，数据集中数据较少的时候可以使用该神经网络来拟合其累积分布函数，再利用其累积分布函数进行微分得到概率密度函数。在复现中，仅实现了三种较为简单的分布情况，没有进行复杂分布的验证。论文创建此神经网络是运用在另一个交通相关的研究[6]中，在数据集中数据个数为 40 以上，迭代次数为 300 以上时能得到一个较为准确的结果。

但是在我看来，论文所提出的神经网络在计算误差函数时，是利用了已知随机变量所对应的累积分布函数的值，在实际情况下，拟合累积分布函数时是不知道该值的，已知的只有随机变量 x 出现的次数。假如已知 x 和 F 的值，其分布实际上已经出来了，我们未来的研究可以注重在仅知道随机变量 x 出现次数的情况下，拟合其概率密度函数。我们未来的工作也可以注重在多维随机变量的研究上。

参考文献

[1] SCHMIDHUBER J. Deep learning in neural networks: An overview[J]. Neural networks, 2015, 61: 85-117.

[2] GAO X, LI W, LOOMES M, et al. A fused deep learning architecture for viewpoint classification of echocardiography[J]. Information Fusion, 2017, 36: 103-113.

[3] LU W, SU H, YANG X, et al. Subsurface temperature estimation from remote sensing data using a clustering-neural network method[J]. Remote Sensing of Environment, 2019, 229: 213-222.

[4] KE X, ZOU J, NIU Y. End-to-end automatic image annotation based on deep CNN and multi-label data augmentation[J]. IEEE Transactions on Multimedia, 2019, 21(8): 2093-2106.

[5] WANG S, CAI J, LIN Q, et al. An overview of unsupervised deep feature representation for text categorization[J]. IEEE Transactions on Computational Social Systems, 2019, 6(3): 504-517.

[6] SUN Z, PAN J S, CHEN C H, et al. A Probability-Based Analytical Model Based on Deep Learning for Traffic Information Estimation[C]//2020 IEEE International Conference on Consumer Electronics-Taiwan (ICCE-Taiwan). 2020: 1-2.