数据拟合模型

袁雨　001139.01　PB20151804

2023 年 4 月 30 日

摘要

本文旨在探究如何使用神经网络完成对某昆虫的三类属性分类。

研究数据包括体长值、翼长值和相应的类别标签，分为训练集和测试集两部分，其中测试集包括从训练集中随机抽取的样本和新的未被训练的样本。此外，数据集还涉及误差（噪声）的影响。

通过构建神经网络，本文探究了该网络在多分类任务中的表现，并测试了不同的网络结构、不同的激活函数对结果的影响，分析了神经网络模型方法的影响因素及性能。并对不同数据集的结果进行了比较分析。

一、前言（问题的提出）

生物学家对某昆虫进行研究，发现该昆虫具有 3 个不同的属性，即可分为 3 类，依据的资料是体长和翼长。

数据格式为：每一行 (x,y,label)：x 为体长值，y 为翼长值，label 为所属类别 0/1/2。

数据集 1 中，insects-training.txt 为生物学家所鉴定的分类结果，作为训练集；insects-testing.txt 作为测试集，其中数据可分为两部分来进行测试；前 60 个为从训练数据中随机抽取的，后 150 个为不在训练数据集的新数据。

数据集 2 在测量昆虫体征时带有部分误差（噪声）。insects-2-training.txt 为生物学家所鉴定的分类结果，作为训练集；insects-2-testing.txt 作为测试集，其中数据可分为两部分来进行测试；前 60 个为从训练数据中随机抽取的，后 150 个为不在训练数据集的新数据。

二、相关工作

对于多分类问题，常见的算法有 KNN、SVM、决策树、朴素贝叶斯以及神经网络等。

在实际应用中，需要根据具体数据情况和模型性能选择合适的算法和超参数，并对模型进行训练和测试。

三、问题分析

该昆虫分类问题属于多分类问题，常见的算法有 KNN、SVM、决策树、朴素贝叶斯以及神经网络等。

其中，KNN 算法可以根据相邻数据点的标签进行分类预测；SVM 可以使用核函数将低维特征空间映射到高维空间进行分类；决策树采用分支结构不断选择最优特征进行分类预测；朴素贝叶斯使用贝叶斯公式计算概率进行分类；神经网络则通过训练自动提取特征进行分类预测。

本文使用神经网络进行昆虫分类。

四、建模的假设

每个昆虫样本的体长和翼长是区分其所属类别的主要因素，忽略了其他次要因素。

所有昆虫样本均可以被合理地归类到 3 个类别中，即不存在不可归类的“边缘”样本。

五、符号说明

表 1: **符号说明**

符号	说明
lr	学习率
$gamma$	学习率衰减因子
acc	准确率

六、数学模型建立

6.1 网络结构

```
class Net(nn.Module):
    def __init__(self)
        super(Net, self).init()
        self.fc1 = nn.Linear(2, 16)
        self.fc2 = nn.Linear(16, 32)
        self.fc3 = nn.Linear(32, 3)
        self.activation = nn.LeakyReLU()

    def forward(self, x):
        x = self.fc1(x)
        x = self.activation(x)
        x = self.fc2(x)
        x = self.activation(x)
        x = self.fc3(x)

        return x
```

定义一个三层前馈神经网络模型，输入为 2 维向量，输出为 3 维向量。该模型采用了线性全连接层和激活函数，其中包括了两个隐藏层和一个输出层。

具体来说，在初始化方法中，通过 nn.Linear 定义了三个全连接层，分别为输入层到第一层隐藏层的 fc1，第一层隐藏层到第二层隐藏层的 fc2，第二层隐藏层到输出层的 fc3。这三个全连接层的参数量分别为 16 * 2 + 16、32 * 16 + 32、3 * 32 + 3 。另外，初始化了一个激活函数。

在前向计算方法 forward 中，输入 x 经过 $fc1$ 后，经过激活函数得到第一层的输出，再经过 $fc2$ 和激活函数得到第二层的输出，最后经过 fc3 得到最终的输出。整个过程就是通过矩阵乘法以及激活函数来实现从输入到输出的映射过程。

具体而言，对于一个输入样本 x ，首先被送入到模型的输入层 fc1 中，通过矩阵乘法和偏置的偏移量相加，将输入 x 变换成了一个长度为 16 的向量；然后经过激活函数处理后，得到了第一隐藏层的输出结果；接下来，第一层的输出结果再次被送入到第二个全连接层 fc2 中，通过矩阵相乘和偏置的偏移量相加，将长度为 16 的向量变换为长度为 32 的向量；经过激活函数后得到第二隐藏层的输出；最终，第二层的输出结果又被送入到输出层 fc3 中，再次进行矩阵乘法和偏置偏移量相加操作，将长度为 32 的向量变换成长度为 3 的向量，即最终的输出结果。

可以使用以下公式来描述该神经网络：

$$\mathbf{y} = f^{(3)}\left(\mathbf{W}^{(3)} f^{(2)}\left(\mathbf{W}^{(2)} f^{(1)}\left(\mathbf{W}^{(1)}\mathbf{x} + \mathbf{b}^{(1)}\right) + \mathbf{b}^{(2)}\right) + \mathbf{b}^{(3)}\right),$$

其中，x 是一个 2 维的输入向量，y 则是一个 3 维的输出向量。$\mathbf{W}^{(l)}$ 和 $\mathbf{b}^{(l)}$ 分别是第 l 层神经元的权重矩阵和偏置向量。$f^{(l)}$ 是激活函数，用于将线性变换后的结果映射到非线性空间中。

训练过程中，在训练集中划分出验证集来评估模型的泛化能力，并通过指定的最大迭代次数和早停机制控制训练过程。在每个 epoch 中，将训练集上的输出结果与真实标签进行比较，计算损失函数并反向传播优化权重参数。使用网络框架封装的 optimizer 完成参数更新过程。同时，在验证集上进行评估，记录并更新最佳的验证集 loss 和准确率，以及此时的训练集 loss 和准确率。当连续 early_stop_patience 个 epoch 验证集 loss 都没有改善时，停止训练，返回最佳的模型参数状态字典、训练数据和验证数据上的损失列表，以及最佳训练集和验证集的输出标签。此外，使用学习率调度器 scheduler 更新学习率。

6.2 参数选择

6.2.1 激活函数

激活函数给神经元引入了非线性因素，使得神经网络可以任意逼近任何非线性函数。常用的有 Sigmoid、Softmax、ReLU、LeakyReLU 等。

Sigmoid 函数:

$$f(x) = \frac{1}{1 + e^{-x}},$$

Softmax 函数:

$$\text{softmax}\left(z_i\right) = \frac{exp(z_i)}{\sum_{j=1}^{k} exp(z_j)},$$

其中 z_i 表示第 i 个节点的输出，K 表示节点总数。

ReLU 函数:

$$f(x) = \begin{cases} 0, & x < 0 \\ x, & x \geq 0 \end{cases},$$

LeakyReLU 函数:

$$f(x) = \begin{cases} ax, & x < 0 \\ x, & x \geq 0 \end{cases},$$

其中 a 是一个小于 1 的超参数。

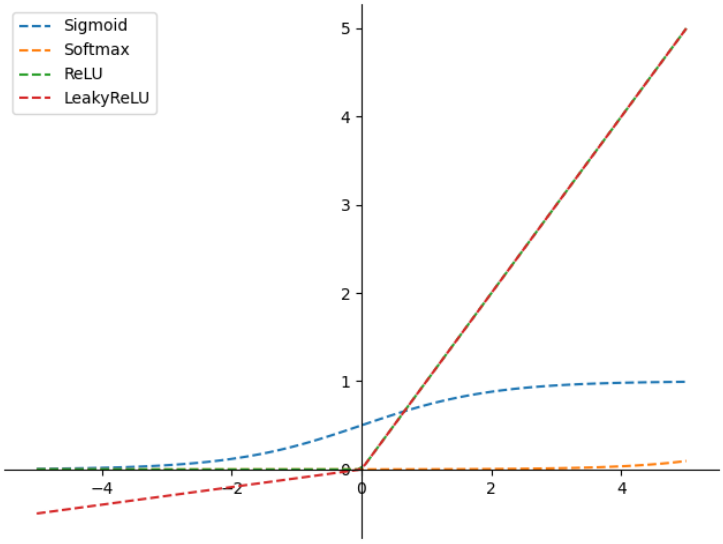

图 1: 激活函数图像

6.2.2 损失函数

使用交叉熵损失函数（CrossEntropyLoss），公式为:

$$L(y, f(x)) = -\sum_{i=1}^{c} \left(y_i \cdot \log\left(f(x)_i\right)\right),$$

其中，y 为真实值的 one-hot 编码，$f(x)$ 为模型输出的概率分布，log 运算为自然对数。

6.2.3 反向传播

使用反向传播来更新模型参数，以最小化损失函数。反向传播算法利用链式法则，能更有效地更新模型参数，提高神经网络训练的效率和准确性。

以对权重的梯度为例 [1]。设 $y = f(x)$ 表示输入 x 通过神经网络前向传播后的输出，$L(y, t)$ 表示预测值 y 与真实值 t 的损失函数，则:

$$\frac{\partial L}{\partial w_{ij}^{(l)}} = \frac{\partial L}{\partial y_j^{(l)}} \frac{\partial y_j^{(l)}}{\partial z_j^{(l)}} \frac{\partial z_j^{(l)}}{\partial w_{ij}^{(l)}},$$

其中，$w_{ij}^{(l)}$ 表示第 l 层的第 i 个神经元向第 $l+1$ 层的第 j 个神经元的连接权重，$y_j^{(l)}$ 表示第 l 层的第 j 个神经元的输出，$z_j^{(l)}$ 表示第 l 层的第 j 个神经元的加权输入，即 $z_j^{(l)} = \sum_i w_{ij}^{(l)} y_i^{(l-1)}$ 。

6.2.4 优化器

优化器（optimizer）使用 Adam 算法，其更新公式为：

$$
\begin{aligned}
v_t &= \beta_1 \cdot v_{t-1} + (1-\beta_1) \cdot g_t, \\
s_t &= \beta_2 \cdot s_{t-1} + (1-\beta_2) \cdot g_t^2, \\
\theta_t &= \theta_{t-1} - \alpha \cdot \frac{vt}{\sqrt{st} + \epsilon},
\end{aligned}
$$

其中，t 表示当前时间步，θ 表示待更新参数，v 为一阶矩估计，s 为二阶矩估计，g 为梯度，α 为学习率，β_1 和 β_2 为动量系数（一般设置为 0.9 和 0.999），ϵ 为平滑项（一般设置为 10^{-8}）。

6.2.5 学习率调度器

学习率调度器（scheduler）使用 StepLR 方法，其更新公式为：

$$
lr = lr_0 * \gamma^{\lfloor \frac{epoch}{step_size} \rfloor},
$$

其中，lr_0 为初始学习率，γ 为衰减因子 (一般设置为 0.1 或 0.5)，$step\_size$ 为衰减间隔 (即多少个 epoch 后进行一次学习率调整)。

6.2.6 评价指标

以预测的准确率作为评价指标。准确率的公式为:

$$
Accuracy = \frac{TP + TN}{TP + TN + FP + FN},
$$

其中，TP 为真正例数量，TN 为真负例数量，FP 为假正例数量，FN 为假负例数量。

七、实验分析

7.3 调参

下面展示在数据集 1 上的对网络结构、激活函数、学习率的调参过程。数据集 2 上的调参过程类似。

7.4 网络结构

表 2: **网络结构**

层数	训练集 acc	验证集 acc
2	93.61 %	90.00 %
3	98.33 %	97.78 %
4	96.39 %,	93.33 %

在一般情况下，神经网络的深度越深，其拟合能力就越强，但也会相应地增加模型的复杂度和计算成本。此外，若深度过深，可能会导致过拟合。

由实验可知，3 层网络已经足够。

7.5 激活函数

表 3: **激活函数**

激活函数	训练集 acc	验证集 acc
Sigmoid	99.44 %	96.67 %
Softmax	99.17 %	96.67 %
ReLU	99.44 %,	97.78 %
LeakyRelu	99.44 %	97.78 %

激活函数直接影响着神经网络的性能和训练效果。在实际应用中，常用的激活函数包括 Softmax、Sigmoid、ReLU 和 LeakyReLU 等。

对于 Sigmoid 函数，它将每个神经元的输出限制在 0 到 1 之间，适用于二分类问题。但是，当应用于多分类任务时，Sigmoid 函数不能将不同类别之间的差异完全刻画出来，可能导致模型的预测效果不佳。

相比之下，Softmax 函数更适合多分类任务。它将神经网络的输出转化为概率分布，对多个类别进行分类。使用 Softmax 函数时，还需要注意是否存在样本的多标签问题。如果一个样本属于多个类别，就需要使用 Multi-Label Softmax 等变种。

除了上述两种常见的激活函数，ReLU 和 LeakyReLU 也被广泛应用于深度学习中。ReLU 函数在非负区间内输出输入值，因此训练速度较快。而 LeakyReLU 则可以缓解 ReLU 函数的问题，使得神经元输出非负时具有更好的鲁棒性。

实验可知可以选择 LeakyReLU 作为激活函数。

7.6 学习率

表 4: **学习率**

lr	gamma	训练集 acc	验证集 acc
0.01	0.1	90.56 %	85.56 %
0.01	0.5	95.56 %	91.11 %
0.03	0.1	94.44 %	93.33 %
0.03	0.5	95.56 %	92.22 %
0.05	0.1	95.28 %	94.44 %
0.05	0.5	95.00 %	94.44 %
0.1	0.1	98.33 %	97.78 %
0.1	0.5	95.28 %	93.33 %
0.3	0.1	94.17 %	95.56 %
0.3	0.5	95.28 %	96.67 %

学习率一开始要保持大些保证收敛速度，在收敛到最优点附近时要小些以避免来回振荡。故使用 stepLR 进行学习率衰减。

当学习率较小时，收敛较慢，适合较大的 gamma。当学习率较大时，容易发生震荡，适合较小的 gamma。实验可知 lr = 0.1，gamma=0.1 效果较好。

八、实验结果

8.7 最合适的一组超参数

网络深度：3 层;
激活函数：LeakyReLU ;
学习率：(0.1，0.1)。
其他参数见源代码。

8.8 训练集和验证集结果

8.8.1 数据集 1

训练集 loss: 0.041941, 验证集 loss: 0.073244;
训练集准确率:98.61 %, 验证集准确率:97.78 %。

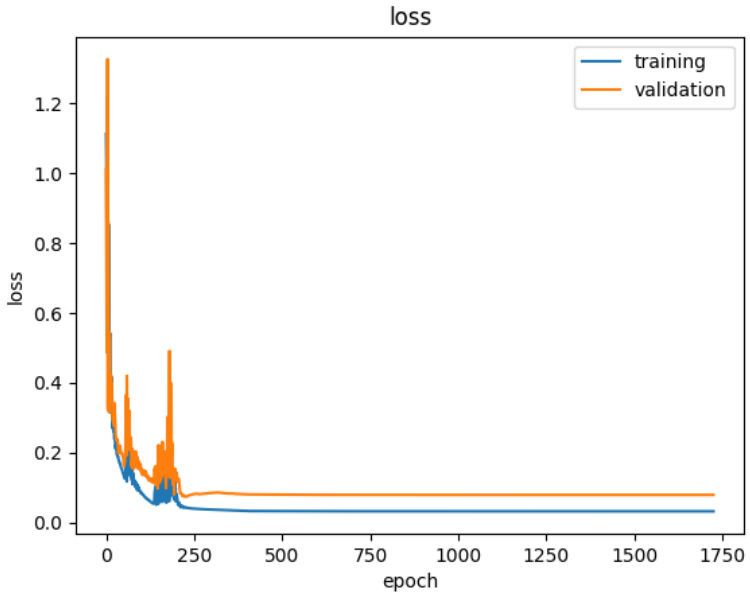

图 2: 数据集 1 损失

由损失图可见，训练前期 loss 发生了震荡。可能是因为数据集较小，随机划分的训练集与验证集数

据分布存在偏差，或者学习率较大等。故在调参时设置了较大的 early_stop_patience，防止训练不充分时就停止。

8.8.2 数据集 2

训练集 loss: 0.272564, 验证集 loss: 0.330230；
训练集准确率:88.57 %, 验证集准确率:83.70 %。

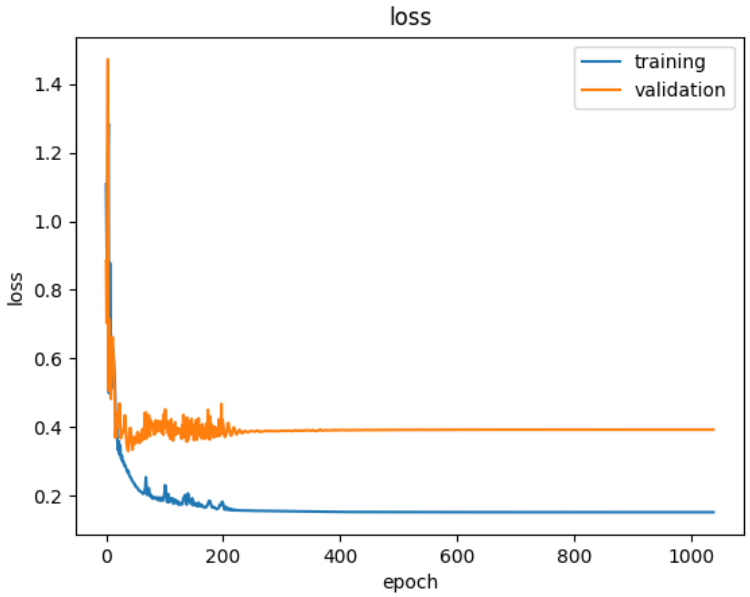

图 3: 数据集 2 损失

由损失图可见，对于噪声数据，训练集 loss 不断下降，验证集 loss 先下降后略有上升。可能是因为神经网络会将这些噪声特征也学习进去，但是这些噪声特征对于验证集来说是没有意义的，从而产生了过拟合。

8.9 测试集结果

8.9.1 数据集 1

前 60 个准确率: 100.00 %，后 150 个准确率: 99.33 %。
可视化如图 4-7，可见绝大部分数据都预测正确了，模型效果较好。

8.9.2 数据集 2

前 60 个准确率: 95.00 %，后 150 个准确率: 94.00 %。
可视化如图 8-11，可见分类错误的数据主要处于类别的边界处。

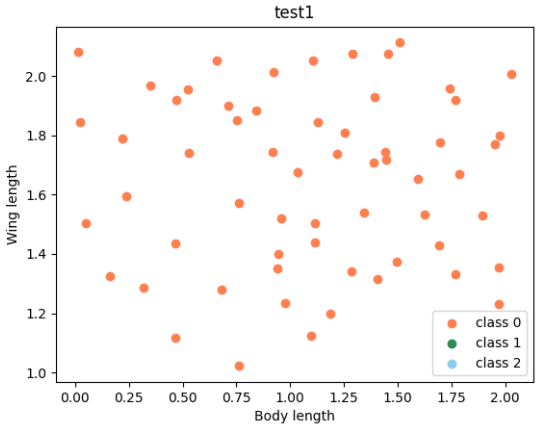

图 4: 数据集 1 前 60 个原数据

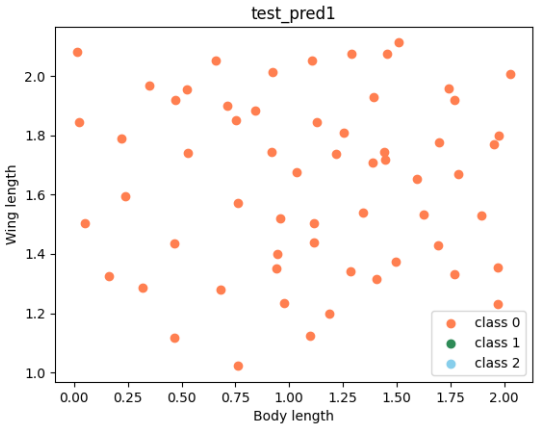

图 5: 数据集 1 前 60 个预测结果

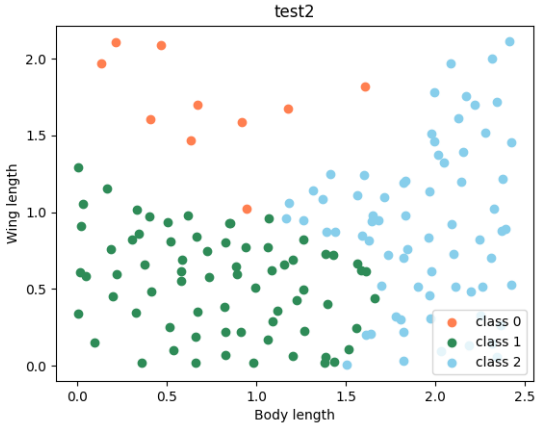

图 6: 数据集 1 后 150 个原数据

图 7: 数据集 1 后 150 个预测结果

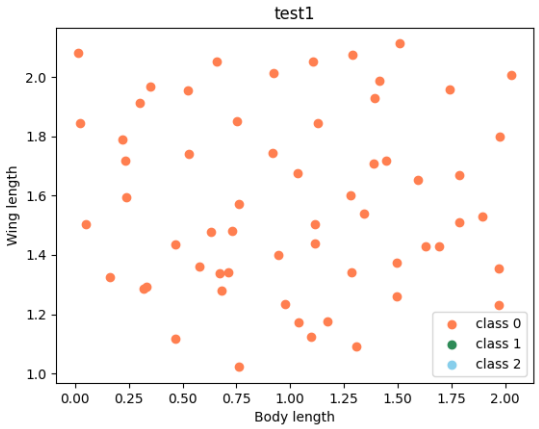

图 8: 数据集 2 前 60 个原数据

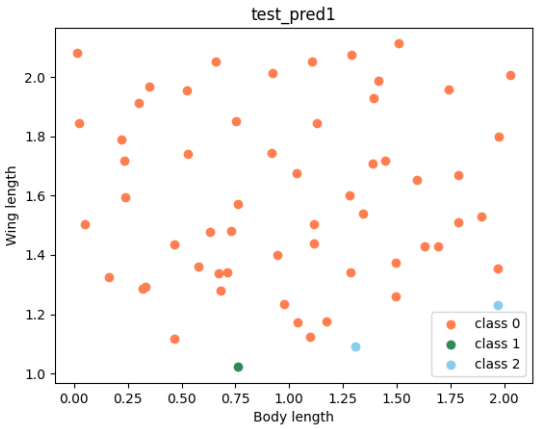

图 9: 数据集 2 前 60 个预测结果

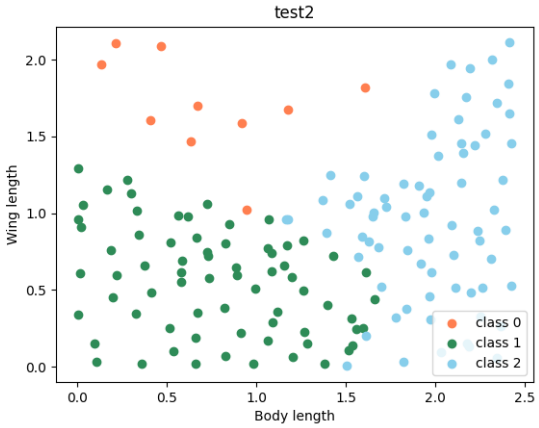

图 10: 数据集 2 后 150 个原数据

图 11: 数据集 2 后 150 个预测结果

九、结论与问题

本文使用前馈神经网络完成了对昆虫的三类属性分类。并测试了不同的网络结构、不同的激活函数对结果的影响，分析了神经网络模型方法的影响因素及性能。并对不同数据集的结果进行了比较分析。

由实验结果可知，模型在数据集 1 上的表现较好，在数据集 2 上的表现较差。当训练集含有噪声时，神经网络往往会将这些噪声特征也学习进去，从而导致训练集上的 loss 不断下降。但是，这些噪声特征对于验证集与测试集来说是没有意义的，loss 可能会出现先下降后上升的情况，即产生了过拟合。

此外，数据集 1 和数据集 2 的前 60 个数据的准确率均高于后 150 个。因为前者来自训练集，后者是模型没见过的数据。当模型在训练集上获得了足够高的准确率后，如果继续训练，可能会过度拟合训练数据，使得模型对新数据的泛化能力较差。

在今后的学习中，可以尝试一些缓解模型过拟合的方法，比如：

1. 数据清洗：通过去除噪声或异常值等不规则、无效的数据，可以减少噪声对模型训练的影响，提高模型的鲁棒性。

2. 数据增强：通过对训练数据进行旋转、平移、裁剪、加噪声等变换，生成更多的样本数据，增加模型的可泛化性；

3. 正则化：如 L1、L2 正则化等；

4. 使用 dropout 等技术。

参考文献

[1] 周志华, 机器学习. 北京: 清华大学出版社, 2016.