机器学习导论 习题三

221300066, 季千🄵, qkjiai@smail.nju.edu.cn

2024 年 5 月 17 日

作业提交注意事项

1. 作业所需的 LaTeX 及 Python 环境配置要求请参考: [Link];
2. 请在 LaTeX 模板中第一页填写个人的学号、姓名、邮箱;
3. 本次作业需提交的文件与对应的命名方式为:
   (a) 作答后的 LaTeX 代码 — `HW3.tex`;
   (b) 由 (a) 编译得到的 PDF 文件 — `HW3.pdf`;
   (c) 第三题模型代码 — `p3_models.py`;
   (d) 第四题模型代码 — `p4_models.py`;
   (e) 第四题训练代码 — `p4_trainer.py`.

   请将以上文件**打包为 学号_姓名.zip** (例如 221300001_张三.zip) 后提交;
3. 若多次提交作业, 则在命名 .zip 文件时加上版本号, 例如 221300001_张三_v1.zip" (批改时以版本号最高的文件为准);
4. 本次作业提交截止时间为 **5 月 17 日 23:59:59**. 未按照要求提交作业, 提交作业格式不正确, **作业命名不规范**, 将会被扣除部分作业分数; 除特殊原因 (如因病缓交, 需出示医院假条) 逾期未交作业, 本次作业记 0 分; **如发现抄袭, 抄袭和被抄袭双方成绩全部取消**;
5. 学习过程中, 允许参考 ChatGPT 等生成式语言模型的生成结果, 但必须在可信的信息源处核实信息的真实性; **不允许直接使用模型的生成结果作为作业的回答内容**, 否则将视为作业非本人完成并取消成绩;
6. 本次作业提交地址为 [Link], 请大家预留时间提前上交, 以防在临近截止日期时, 因网络等原因无法按时提交作业.

1 [25pts] Principal Component Analysis

主成分分析是一种经典且常用的数据降维方法. 请仔细阅读学习《机器学习》第十章 10.3 节, 并根据图 10.5 中的算法内容, 完成对如下 6 组样本数据的主成分分析.

$$\mathbf{X} = \begin{bmatrix} 2 & 3 & 3 & 4 & 5 & 7 \\ 2 & 4 & 5 & 5 & 6 & 8 \end{bmatrix}$$

(1) **[6pts]** 试求样本数据各维的均值、标准差.

(2) **[7pts]** 试求标准化后的样本矩阵 $\mathbf{X}_{\text{std}}$, 以及 $\mathbf{X}_{\text{std}}$ 对应的协方差矩阵.
(**Hint:** 相比中心化, 标准化还需要额外除以标准差.)

(3) **[7pts]** 试求协方差矩阵对应的特征值, 以及投影矩阵 $\mathbf{W}^{\star}$.

(4) **[5pts]** 如果选择重构阈值 $t = 95\%$, 试求 PCA 后样本 $\mathbf{X}_{\text{std}}$ 在新空间的坐标矩阵.

Solution. 此处用于写解答 (中英文均可)
(1) 样本数据各维的均值、标准差:

$$\mu_1 = \frac{1}{6}\sum_{i=0}^{6} x_{1i} = 4,$$

$$\mu_2 = \frac{1}{6}\sum_{i=0}^{6} x_{2i} = 5,$$

$$\sigma_1 = \sqrt{\frac{1}{6}\sum_{i=0}^{6}(x_{1i}-\mu_1)^2} = \sqrt{\frac{8}{3}} \approx 1.633,$$

$$\sigma_2 = \sqrt{\frac{1}{6}\sum_{i=0}^{6}(x_{2i}-\mu_2)^2} = \sqrt{\frac{10}{3}} \approx 1.826.$$

(2) 标准化后的样本矩阵 $\mathbf{X}_{\text{std}} \leftarrow \frac{\boldsymbol{x}_i - \mu}{\sigma} =$

$$\begin{bmatrix} -\frac{\sqrt{6}}{2} & -\frac{\sqrt{6}}{4} & -\frac{\sqrt{6}}{4} & 0 & \frac{\sqrt{6}}{4} & \frac{3\sqrt{6}}{4} \\ -\frac{3\sqrt{30}}{10} & -\frac{\sqrt{30}}{10} & 0 & 0 & \frac{\sqrt{30}}{10} & \frac{3\sqrt{30}}{10} \end{bmatrix}$$

(3) 协方差矩阵

$$\frac{1}{5}\mathbf{X}\mathbf{X}^T = \frac{1}{5}\begin{bmatrix} 6 & \frac{51\sqrt{5}}{20} \\ \frac{51\sqrt{5}}{20} & 6 \end{bmatrix} = \begin{bmatrix} 1.2 & \frac{51\sqrt{5}}{100} \\ \frac{51\sqrt{5}}{100} & 1.2 \end{bmatrix}$$

故，计算特征值：$\mathbf{X}\mathbf{X}^T\hat{\boldsymbol{w}} = \lambda\hat{\boldsymbol{w}}$，解得：$\lambda_1 = \frac{51\sqrt{5}}{100} \approx 2.340$, $\lambda_2 = 1.2 - \frac{51\sqrt{5}}{100} \approx 0.057$

$$\mathbf{W}^{*} = \begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix}$$

(4) 由于重构阈值为 0.95，而 $\frac{\lambda_1}{\lambda_1+\lambda_2} \approx 0.98 > 0.95$，所以选取 λ_1 对应的特征向量作为投影向量，所以有：$\mathbf{X}'_{\text{std}} = \hat{\boldsymbol{w}}_1^T\mathbf{X}_{\text{std}} =$

$$\begin{bmatrix} -\frac{\sqrt{6}}{2} - \frac{3\sqrt{30}}{10} & -\frac{\sqrt{6}}{4} - \frac{\sqrt{30}}{10} & -\frac{\sqrt{6}}{4} & 0 & \frac{\sqrt{6}}{4} + \frac{\sqrt{30}}{10} & \frac{3\sqrt{6}}{4} + \frac{3\sqrt{30}}{10} \end{bmatrix}$$

2 [25pts] Support Vector Machines

核函数是 SVM 中常用的工具, 其在机器学习中有着广泛的应用与研究. 请仔细阅读学习《机器学习》第六章, 并回答如下问题.

(1) **[6pts]** 试判断下图 ① 到 ⑥ 中哪些为支持向量.

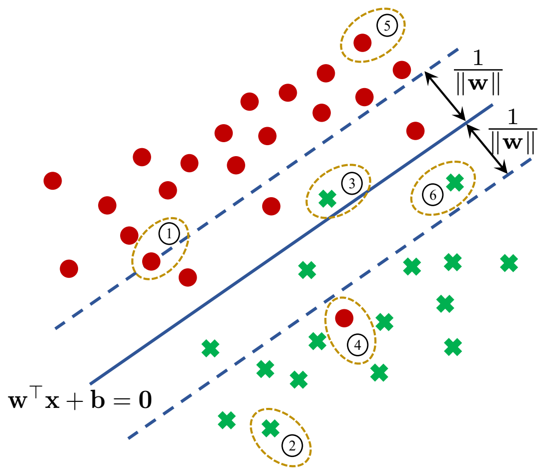

图 1: 分离超平面示意图

(2) **[5pts]** 试判断 $\kappa(\boldsymbol{x},\mathbf{z}) = (\langle\boldsymbol{x},\mathbf{z}\rangle + 1)^2$ 是否为核函数, 并给出证明或反例.

(3) **[5pts]** 试判断 $\kappa(\boldsymbol{x},\mathbf{z}) = (\langle\boldsymbol{x},\mathbf{z}\rangle - 1)^2$ 是否为核函数, 并给出证明或反例.

(4) **[9pts]** 试证明: 若 κ_1 和 κ_2 为核函数, 则两者的直积

$$\kappa_1 \otimes \kappa_2(\boldsymbol{x},\mathbf{z}) = \kappa_1(\boldsymbol{x},\mathbf{z})\kappa_2(\boldsymbol{x},\mathbf{z})$$

也是核函数. 即证明《机器学习》(6.26) 成立.

(**Hint:** 利用核函数与核矩阵的等价性.)

Solution. 此处用于写解答 (中英文均可)

(1) 支持向量为: ①

(2) 证明 $\kappa(\boldsymbol{x},\mathbf{z}) = (\langle\boldsymbol{x},\mathbf{z}\rangle + 1)^2$ 是核函数:

由于 $\langle x_i, x_j\rangle$ 为线性核函数，所以其核矩阵:

$$\mathbf{K}_0 = \begin{bmatrix} \langle\boldsymbol{x}_1,\boldsymbol{x}_1\rangle & \langle\boldsymbol{x}_1,\boldsymbol{x}_2\rangle & \cdots & \langle\boldsymbol{x}_1,\boldsymbol{x}_m\rangle \\ \langle\boldsymbol{x}_2,\boldsymbol{x}_1\rangle & \langle\boldsymbol{x}_2,\boldsymbol{x}_2\rangle & \cdots & \langle\boldsymbol{x}_2,\boldsymbol{x}_m\rangle \\ \vdots & \vdots & \ddots & \vdots \\ \langle\boldsymbol{x}_m,\boldsymbol{x}_1\rangle & \langle\boldsymbol{x}_m,\boldsymbol{x}_2\rangle & \cdots & \langle\boldsymbol{x}_m,\boldsymbol{x}_m\rangle \end{bmatrix} \succeq 0$$

即：对于所有的 $y \in R^m$，以及所有的 $D = \{\boldsymbol{x}_1,\boldsymbol{x}_2,\ldots,\boldsymbol{x}_m\}$ 有 $y^\top\mathbf{K}_0 y \geq 0$. 而对于

$\kappa(\boldsymbol{x},\mathbf{z})=(\langle\boldsymbol{x},\mathbf{z}\rangle+1)^2$, 其核矩阵为:

$$\mathbf{K}=\begin{bmatrix}\langle\boldsymbol{x}_1,\boldsymbol{x}_1\rangle^2+2\langle\boldsymbol{x}_1,\boldsymbol{x}_1\rangle+1 & \langle\boldsymbol{x}_1,\boldsymbol{x}_2\rangle^2+2\langle\boldsymbol{x}_1,\boldsymbol{x}_2\rangle+1 & \cdots & \langle\boldsymbol{x}_1,\boldsymbol{x}_m\rangle^2+2\langle\boldsymbol{x}_1,\boldsymbol{x}_m\rangle+1\\ \langle\boldsymbol{x}_2,\boldsymbol{x}_1\rangle^2+2\langle\boldsymbol{x}_2,\boldsymbol{x}_1\rangle+1 & \langle x_2,x_2\rangle^2+2\langle\boldsymbol{x}_2,\boldsymbol{x}_2\rangle+1 & \cdots & \langle\boldsymbol{x}_2,\boldsymbol{x}_m\rangle^2+2\langle\boldsymbol{x}_2,\boldsymbol{x}_m\rangle+1\\ \vdots & \vdots & \ddots & \vdots\\ \langle\boldsymbol{x}_m,\boldsymbol{x}_1\rangle^2+2\langle\boldsymbol{x}_m,\boldsymbol{x}_1\rangle+1 & \langle\boldsymbol{x}_m,\boldsymbol{x}_2\rangle^2+2\langle\boldsymbol{x}_m,\boldsymbol{x}_2\rangle+1 & \cdots & \langle\boldsymbol{x}_m,\boldsymbol{x}_m\rangle^2+2\langle\boldsymbol{x}_m,\boldsymbol{x}_m\rangle+1\end{bmatrix}$$

$$=\begin{bmatrix}\langle\boldsymbol{x}_1,\boldsymbol{x}_1\rangle^2 & \langle\boldsymbol{x}_1,\boldsymbol{x}_2\rangle^2 & \cdots & \langle\boldsymbol{x}_1,\boldsymbol{x}_m\rangle^2\\ \langle\boldsymbol{x}_2,\boldsymbol{x}_1\rangle^2 & \langle\boldsymbol{x}_2,\boldsymbol{x}_2\rangle^2 & \cdots & \langle\boldsymbol{x}_2,\boldsymbol{x}_m\rangle^2\\ \vdots & \vdots & \ddots & \vdots\\ \langle\boldsymbol{x}_m,\boldsymbol{x}_1\rangle^2 & \langle\boldsymbol{x}_m,\boldsymbol{x}_2\rangle^2 & \cdots & \langle\boldsymbol{x}_m,\boldsymbol{x}_m\rangle^2\end{bmatrix}+\mathbf{K}_0+\mathbf{1}$$

将上式中的第一个矩阵设为 $\mathbf{K}_1$, 显然其等价为核函数 $\langle\boldsymbol{x}_i,\boldsymbol{x}_j\rangle^2$ 的核矩阵, 其为多项式核, 故显然 $\mathbf{K}_1$ 为半正定矩阵, 而对于全 1 方阵 $\mathbf{1}$, 也有 $y^\top\mathbf{1}y=\sum_i\sum_j y_iy_j=(\sum_i y_i)^2\geq 0$, 显然 $\mathbf{1}$ 也为半正定矩阵, 所以半正定矩阵之和矩阵 $\mathbf{K}$ 也为半正定矩阵, 所以 $\kappa(\boldsymbol{x},\mathbf{z})=(\langle\boldsymbol{x},\mathbf{z}\rangle+1)^2$ 为核函数

(3) 举反例如下: 令 $D=\{[1,0],[0,1]\}$, 对于 $\kappa(\boldsymbol{x},\mathbf{z})=(\langle\boldsymbol{x},\mathbf{z}\rangle-1)^2$, 其核矩阵为

$$\mathbf{K}=\begin{bmatrix}0 & 1\\ 1 & 0\end{bmatrix}$$

$|K|=-1$, 显然 K 不是半正定矩阵, 故 $\kappa(\boldsymbol{x},\mathbf{z})=(\langle\boldsymbol{x},\mathbf{z}\rangle-1)^2$ 不是核函数

(4) 若 κ_1 和 κ_2 为核函数, 假设两者的核矩阵分别为 $\mathbf{K}_1$ 和 $\mathbf{K}_2$ 则两者的直积 $\kappa_1\otimes\kappa_2(\boldsymbol{x},\mathbf{z})=\kappa_1(\boldsymbol{x},\mathbf{z})\kappa_2(\boldsymbol{x},\mathbf{z})$, 其有核矩阵 $\mathbf{K}=\mathbf{K}_1\odot\mathbf{K}_2$, 其中 $\odot$ 表示两矩阵对应元素相乘运算. 由于核矩阵都是实对称矩阵, 所以将 $\mathbf{K}_2$ 正交分解化可得:

$$\mathbf{K}_2=\mathbf{PAP}^\top$$

其中 $\mathbf{A}$ 为对角线是 $\mathbf{K}_2$ 特征值构成的对角阵, 而 $\mathbf{P}$ 为所对应的特征向量 $\boldsymbol{x}_i$ 组成的正交矩阵, 所以有:

$$\mathbf{K}_2=\mathbf{PAP}^T=[\boldsymbol{x}_1,\boldsymbol{x}_2,\ldots,\boldsymbol{x}_m]\mathbf{A}[\boldsymbol{x}_1,\boldsymbol{x}_2,\ldots,\boldsymbol{x}_m]^T=\sum_{i=0}^m\lambda_i\boldsymbol{x}_i\boldsymbol{x}_i^T$$

而显然:

$$\mathbf{K}_1\odot(\boldsymbol{x}_i\boldsymbol{x}_i^T)=\mathbf{D}_i\mathbf{K}_1\mathbf{D}_i^T$$

其中 $\mathbf{D}_i$ 是对角线为 $\boldsymbol{x}_i$ 的对角阵, 上式基于对角阵左乘和右乘分别对应, 按行和列乘上相应对角线元素. 所以有

$$\mathbf{K}=\mathbf{K}_1\odot\mathbf{K}_2=\mathbf{K}_1\odot\sum_{i=0}^m\lambda_i\boldsymbol{x}_i\boldsymbol{x}_i^T=\sum_{i=0}^m\lambda_i\mathbf{D}_i\mathbf{K}_1\mathbf{D}_i^\top$$

由于半正定矩阵特征值 $\lambda_i\geq 0$, 所以, 对于所有的 $y\in R^m$, 有:

$$y\mathbf{K}y^T=\sum_{i=0}^m\lambda_i\mathbf{D}_iy\mathbf{K}_1(\mathbf{D}_iy)^T\geq 0$$

所以 $\mathbf{K}$ 为半正定矩阵, 所以 $\kappa_1\otimes\kappa_2(\boldsymbol{x},\mathbf{z})=\kappa_1(\boldsymbol{x},\mathbf{z})\kappa_2(\boldsymbol{x},\mathbf{z})$ 为核函数。

3 [30pts] Basics of Neural Networks

多层前馈神经网络可以被用作分类模型. 在本题中, 我们先回顾前馈神经网络的一些基本概念, 再利用 Python 实现一个简单的前馈神经网络以进行分类任务.

[基础原理] 首先, 考虑一个多层前馈神经网络, 规定网络的输入层是第 0 层, 输入为 $\mathbf{x} \in \mathbb{R}^d$. 网络有 M 个隐层, 第 h 个隐层的神经元个数为 N_h, 输入为 $\mathbf{z}_h \in \mathbb{R}^{N_{h-1}}$, 输出为 $\mathbf{a}_h \in \mathbb{R}^{N_h}$, 权重矩阵为 $\mathbf{W}_h \in \mathbb{R}^{N_{h-1} \times N_h}$, 偏置参数为 $\mathbf{b}_h \in \mathbb{R}^{N_h}$. 网络的输出层是第 $M+1$ 层, 神经元个数为 C, 权重矩阵为 $W_{M+1} \in \mathbb{R}^{N_M \times C}$, 偏置参数为 $\mathbf{b}_{M+1} \in \mathbb{R}^C$, 输出为 $\mathbf{y} \in \mathbb{R}^C$. 网络隐层和输出层的激活函数均为 f, 网络训练时的损失函数为 $\mathcal{L}$, 且 f 与 $\mathcal{L}$ 均可微.

(1) **[5pts]** 请根据前向传播原理, 给出 $\mathbf{z}_h, \mathbf{a}_h$ $(1 \le h \le M)$ 及 $\mathbf{y}$ 的具体数学表示.

(2) **[5pts]** 结合 (1) 的表示形式, 谈谈为何要在神经网络中引入 (非线性) 激活函数 f?

[编程实践] 下面, 我们针对一个特征数 $d = 2$, 类别数为 2 的分类数据集, 实现一个结构为 "2-2-1" 的简单神经网络, 即: 输入层有 2 个神经元; 隐层仅一层, 包含 2 个神经元; 输出层有 1 个神经元; 所有层均使用 Sigmoid 作为激活函数. 此外, 我们使用 BP 算法进行神经网络的训练. **关于本题的细节介绍及具体要求, 请见附件: `p3_编程题说明`.** 请参考编程题说明文档与附件中的代码模板, 完成下面的任务.

(3) **[15pts]** 基于 `p3_models.py`, 补全缺失代码, 实现神经网络分类器的训练与预测功能.

(4) **[5pts]** 参考《机器学习》及第一次作业中对超参数调节流程的介绍, 为 (1) 中模型设置合适的超参数 (即: 学习率与迭代轮数). **请将选择的超参数设置为调用模型时的默认参数, 并在解答区域简要介绍你的超参数调节流程.**
(提示: 可以从数据集划分方法, 评估方法, 候选超参数生成方法等角度说明).

Solution. 此处用于写解答 (中英文均可)

(1) 在神经网络中，每一层的输出都是下一层的输入。对于第 h 层（隐藏层）来说，它的输入 $\mathbf{z}_h$ 是上一层的输出 $\mathbf{a}_{h-1}$ 与权重矩阵 $\mathbf{W}_h$ 的乘积加上偏置向量 $\mathbf{b}_h$，即：

$$\mathbf{z}_h = \mathbf{W}_h \mathbf{a}_{h-1} + \mathbf{b}_h$$

然后通过激活函数 f，可以得到该层的输出 $\mathbf{a}_h$：

$$\mathbf{a}_h = f(\mathbf{z}_h)$$

输出层的计算方式与隐藏层类似，只不过输入是最后一个隐藏层的输出，即：

$$\mathbf{y} = f(\mathbf{W}_{M+1} \mathbf{a}_M + \mathbf{b}_{M+1})$$

(2) 引入非线性激活函数 f 的主要原因是线性模型的表达能力有限。如果神经网络只有线性操作（即没有激活函数或者激活函数是线性的），无论网络有多少层，其总体仍然是一个线性模型，这意味着它只能表示一些简单的模式，不能捕捉数据的复杂特征。而非线性激活函数可以帮助神经网络学习到数据中的非线性模式，从而大大提高了神经网络的表达能力

和拟合能力。

(3)

图 2: 训练与预测结果

(4) 超参数调节流程:

1. 数据集划分:

使用 train_test_split 方法从 sklearn.model_selection 进行数据划分,比如将数据集划分为 80% 的训练集和 20% 的验证集。

2. 评估方法

将 NeuralNetworkClassifier 包装成一个符合 GridSearchCV 要求的评估器。使用 GridSearchCV 来进行超参数的网格搜索,尝试所有可能的超参数组合,并通过交叉验证来评估每一组超参数的性能。

3. 超参数候选生成

对于 learning_rate 和 max_epoch,设置如下等候选值:

learning_rate: 0.001, 0.01, 0.1

max_epoch: 10, 50, 100, 75, 85, 55, 65, 95

4. 交叉验证与选择并对 model 进行更新

使用 GridSearchCV 进行交叉验证,确保模型的泛化能力,根据结果选择最优参数。

5. 对 model 进行更新

最优参数为 $learning\_rate = 0.01 max\_epoch = 85$

更新模型默认参数,以便在实际使用时能够直接应用最优参数。

4 [20(+5)pts] Neural Networks with PyTorch

在上一题的编程实践中, 我们使用 Python 实现了一个简单的神经网络分类器. 其中, 我们根据 BP 算法中神经网络参数梯度的数学定义, 手动实现了梯度计算及参数更新的流程. 然而, 在现实任务中, 我们往往利用深度学习框架来进行神经网络的开发及训练. 一些常用的框架例如: PyTorch, Tensorflow 或 JAX, 以及国产的 PaddlePaddle, MindSpore. 这类框架往往支持自动微分功能, 仅需定义神经网络的具体结果与前向传播过程, 即可在训练时自动计算参数的梯度, 进行参数更新. 此外, 我们可以使用由框架实现的更成熟的优化器 (如 Adam 等) 来提高模型的收敛速度, 或使用 GPU 加速以提高训练效率. 如果希望在今后的学习科研中应用神经网络, 了解至少一种框架的使用方式是极为有益的.

在本题中, 我们尝试使用 PyTorch 框架来进行神经网络的开发, 完成 FashionMNIST 数据集上的图像分类任务. 与上一题考察神经网络底层原理不同, 本题考察大家阅读文档, 搭建模型并解决实际任务的能力. **关于本题的细节介绍及具体要求, 请见附件: `p4_`编程题说明.** 请参考编程题说明文档与附件中的代码模板, 完成下面的任务.

(1) **[10pts]** 阅读文档, 配置 PyTorch 环境, 补全 `p4_models.py` 中神经网络的 `__init__` 与 `forward` 方法, 最终成功运行 `p4_main.py`. **请在解答区域附上运行 `p4_main.py` 后生成的 `plot.png`.**

(2) **[10pts]** 从 (1) 中生成的训练过程图片 `plot.png` 中可以看出: 模型明显出现了**过拟合**现象, 即训练一定轮次后, 训练集 loss 持续下降, 但测试集 loss 保持不变或转为上升. 请提出**至少两种**缓解过拟合的方法, 分别通过编程实现后, 在解答区域附上应用前后的训练过程图片, 并结合图片简要分析方法有效/无效的原因.
(提示: 可以考虑的方法包括但不限于: Dropout, 模型正则化, 数据增强等.)

(3) **[5pts]** (本题为附加题, 得分计入卷面分数, 但本次作业总得分不超过 100 分)
寻找最优的改进神经网络结构及训练方式的方法, 使模型在另一个未公开的测试集上取得尽可能高的分类准确率.
本题得分规则如下: 假设共有 N 名同学完成本题, 我们将这 N 名同学的模型测试集分类准确率由高到低排列, 对前 $K = \min(\lfloor N/10 \rfloor, 10)$ 名同学奖励附加题分数. 对于排列序号为 i 的同学 ($1 \le i \le K$), 得分为: $5 - \lfloor 5(i-1)/k \rfloor$.
(提示: 你可以自由尝试修改模型结构, 修改优化器超参数等方法.)

Solution. 此处用于写解答 (中英文均可)
(1)

从图片可以看出，test loss 显著降低，与 train loss 的差距明显变小，所以方法有效

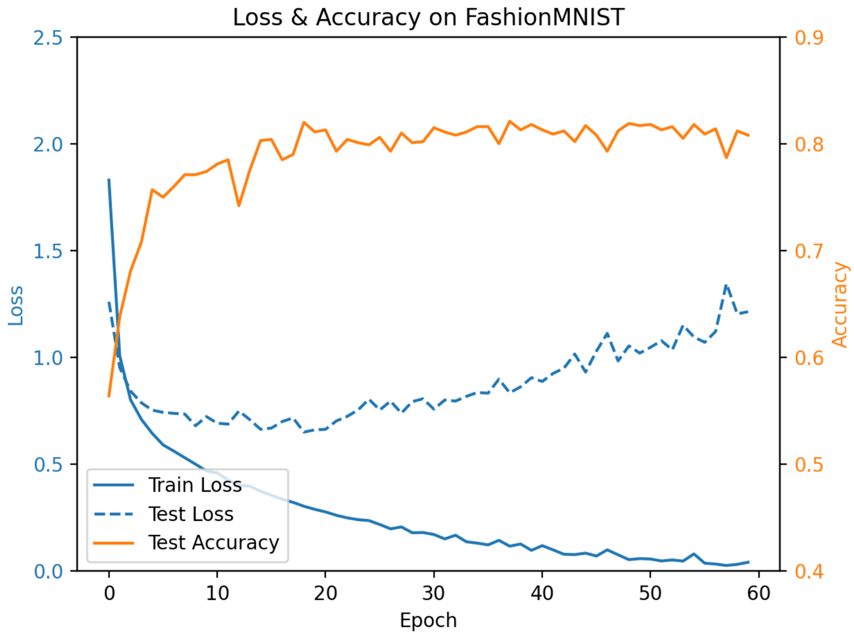

图 3: plot

(2) 方法一：添加 Dropout 层，实现为 BetterFashionClassifier1 类
在神经网络的每个隐藏层之后添加 Dropout 层，以一定概率随机丢弃部分神经元的输出，减少过拟合。

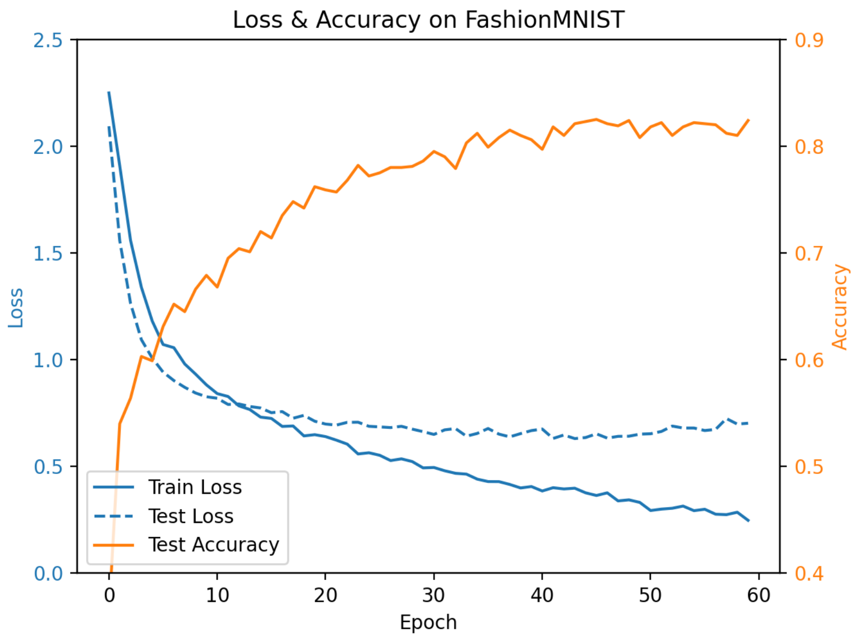

图 4: Dropout 层 plot

从图片可以看出，test loss 显著降低，与 train loss 的差距明显变小，所以方法有效

原因：Dropout 可以随机丢弃神经元的输出，迫使网络学习更加鲁棒的特征表示，减少过拟合。

方法二：在损失函数中添加 L2 正则化项，惩罚模型参数的大小，防止过拟合。实现为 BetterFashionClassifier2 类

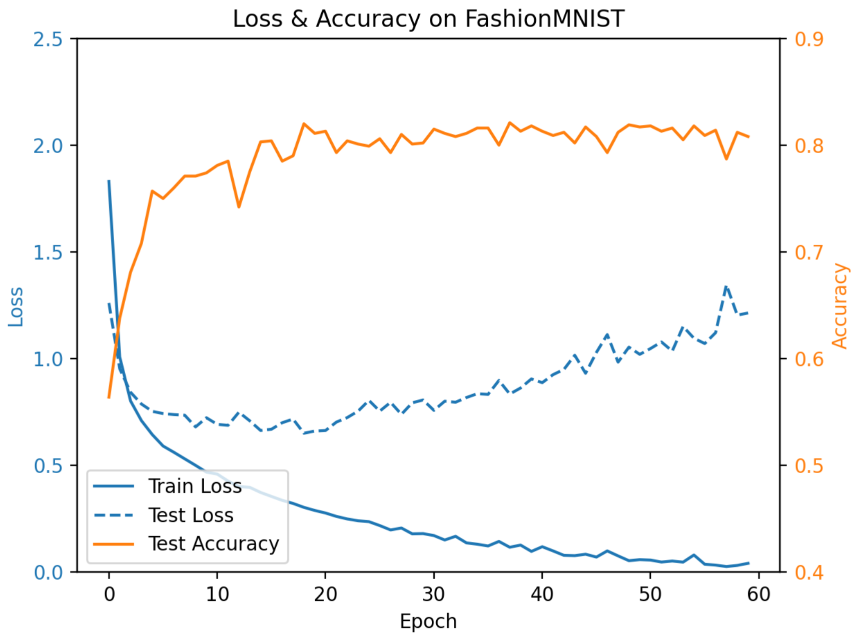

图 5: L2 正则化 plot

从图片可以看出，test loss 与 train loss 没有发生显著变化，所以方法无效

原因：正则化系数可能过大，可能会导致模型欠拟合，影响了模型的拟合能力。

(3) 解答如下：

BetterFashionClassifier 类使用了卷积层、池化层、全连接层等组件，用于实现更复杂的神经网络结构。具体来说：

- 神经网络包括两个卷积层（Conv2d）、两个最大池化层（MaxPool2d）、两个全连接层（Linear）以及激活函数（ReLU）。
- 在 forward 方法中，定义了神经网络的前向传播过程，按照卷积、激活、池化、全连接的顺序进行计算。

BetterTrainer 类包括了训练步骤 train_step 和测试方法 test。

- 训练过程中使用交叉熵损失函数（CrossEntropyLoss）和 Adam 优化器（Adam）进行模型优化。
- 在测试过程中，计算模型在测试集上的分类准确率。

这样设计的神经网络结构和训练方法可以帮助提高模型的分类效果，通过卷积和池化层提取更丰富的特征信息，同时使用交叉熵损失函数和 Adam 优化器进行训练，有助于提高模型的收敛速度和准确率。

Acknowledgments

允许与其他同样未完成作业的同学讨论作业的内容, 但需在此注明并加以致谢; 如在作业过程中, 参考了互联网上的资料, 且对完成作业有帮助的, 亦需注明并致谢.