机器学习

作业五

一. (30 points) 降维

1. 基于 numpy 和 fetch_lfw_people 数据集实现主成分分析（PCA）算法，不可以调用 sklearn 库，完成下面代码并且可视化前 5 个主成分所对应的特征脸 (15 points)

```
from sklearn.datasets import fetch_lfw_people
import matplotlib.pyplot as plt
# 1. 加载 LFW 数据集
lfw_people = fetch_lfw_people(min_faces_per_person=100, resize=0.4)
# 2. 获取数据
X = lfw_people.data

# Todo1: 写出PCA函数
def PCA1(X, n_components=5):

    return eigenfaces

eigenfaces = PCA(X)

# Todo2: 可视化5个主成分对应的特征脸
```

2. 根据局部线性嵌入 (Locally Linear Embedding, LLE) 的算法流程，尝试编写 LLE 代码，可以基于 sklearn 实现，并在瑞士卷数据集上进行实验降到 2 维空间。提交代码和展示多个在不同参数下的可视化的实验结果。请分析使用 LLE 时可能遇到哪些挑战 (15 points) [提示：瑞士卷数据集可以用 sklearn 的 make_swiss_roll(n_samples=3000, random_state=0) 生成 3000 个样本]

二. (40 points) 半监督

1. 在本题中使用朴素贝叶斯模型和 SST2 数据集进行半监督 EM 算法的实践，代码前面部分如下，请补充完后续代码，只保留 10% 的标注数据，置信度设为 0.7，训练 5 轮，给出训练后模型在验证集上的分类结果 (15 points)

```python
import numpy as np
import random
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score
from datasets import load_dataset, load_from_disk # SST-2 数据集

# 设置随机种子，确保结果可复现
random.seed(42)
np.random.seed(42)

# 加载 SST-2 数据集
datasets = load_dataset('glue', 'sst2')
train_data = datasets['train']
valid_data = datasets['validation']

# 提取文本和标签
train_texts = [example['sentence'] for example in train_data]
train_labels = [example['label'] for example in train_data]
valid_texts = [example['sentence'] for example in valid_data]
valid_labels = [example['label'] for example in valid_data]

# 划分标注数据和未标注数据
labeled_size = int(0.1 * len(train_texts))  # 仅保留10%的标注数据
indices = np.arange(len(train_texts))
np.random.shuffle(indices)

labeled_indices = indices[:labeled_size]
unlabeled_indices = indices[labeled_size:]

labeled_texts = [train_texts[i] for i in labeled_indices]
labeled_labels = [train_labels[i] for i in labeled_indices]
unlabeled_texts = [train_texts[i] for i in unlabeled_indices]

# 向量化文本数据
vectorizer = CountVectorizer()
X_labeled = vectorizer.fit_transform(labeled_texts)
X_unlabeled = vectorizer.transform(unlabeled_texts)
X_valid = vectorizer.transform(valid_texts)

# 半监督EM算法 - 使用朴素贝叶斯模型
```

```
model = MultinomialNB()
model.fit(X_labeled, labeled_labels)
```

Listing 1: 半监督 EM 算法的实践

2. 伪标签的置信度大小对模型的训练结果会有一定的影响，通常会有固定置信度和动态设置置信度两种方式，请你完成这两种方式，并统计不同方式下每次迭代中伪标签的错误率，并分析这两种方式的优劣 (10 points)

3. 请设计新的策略来提升半监督算法 (7 points)

4. 修改代码，设置不同的迭代次数（如 3 次、5 次、15 次）。在验证集上分析：不同迭代次数下，模型性能如何变化？分析为什么在过多迭代的情况下，模型性能可能下降？(8 points)

三. (30 points) 强化学习

在本问题中，你将思考如何通过在马尔可夫决策过程（MDP）中连续做决策来最大化奖励，并深入了解贝尔曼方程——解决和理解 MDP 的核心方程。

考虑经典的网格世界 MDP，即一个 4 * 3 的网格，其中单元格 (2, 2) 无法到达。智能体从单元格 (1, 1) 开始，并在环境中导航。在这个世界中，智能体每个格子里可以采取四个动作：上、下、左、右。格子用 (水平，垂直) 来索引；也就是说，单元格 (4, 1) 位于右下角。世界的转移概率如下：如果智能体在当前位置采取一个动作，它将以 0.8 的概率移动到动作的方向所在的格子，并以 0.1 的概率滑到动作的相对右或左的方向。如果动作（或滑动方向）指向一个没有可通过的格子（即边界或 (2, 2) 格子的墙壁），那么该动作将保持智能体处于当前格子。例如，如果智能体在 (3, 1) 位置，并采取向上的动作，它将以 0.8 的概率移动到 (3, 2)，以 0.1 的概率移动到 (2, 1)，以 0.1 的概率移动到 (4, 1)。如果智能体在 (1, 3) 位置并采取右移动作，它将以 0.8 的概率移动到 (2, 3)，以 0.1 的概率移动到 (1, 2)，以 0.1 的概率停留在 (1, 3)。当智能体到达定义的奖励状态时（在 (4, 2) 和 (4, 3) 单元格），智能体将获得相应的奖励，并且本次回合结束。

回顾计算 MDP 中每个状态的最优价值, $V^*(s)$ 的贝尔曼方程，其中我们有一组动作 A，一组状态 S，每个状态的奖励值 $R(s)$，我们的世界的转移动态 $P(s'|s,a)$，以及折扣因子 γ：

$$V^*(s) = R(s) + \gamma \max_{a \in A} \sum_{s' \in S} P(s'|s,a) V^*(s')$$

最后，我们将策略表示为 $\pi(s) = a$ 其中策略 π 指定了在给定状态下采取的行动。

(a) 考虑一个智能体从单元格 (1, 1) 开始，在第 1 步和第 2 步分别采取向上和向上的动作。计算在每个时间步内，根据这一动作序列，智能体可以到达哪些单元格，以及到达这些单元格的概率。(8 points)

(b) 考虑当前没有奖励值的所有状态的奖励函数 $R(s)$（即除了 (4, 2) 和 (4, 3) 以外的每个单元格）。定义在以下奖励值下，智能体的最优策略：(i.) $R(s) = 0$, (ii.) $R(s) = -2.0$, and (iii.) $R(s) = 1.0$. 你可以假设折扣因子接近 1，例如 0.9999。画出网格世界并标出在每个状态下应采取的动作可能会对你有所帮助（记住，策略是在 MDP 中对所有状态进行定义的!）(10 points)

注意: 你不需要算法上计算最优策略。你必须列出每种情况的完整策略，但只需要提供直观的理由。

(c) 有时, MDP 的奖励函数形式为 $R(s,a)$ 它依赖于所采取的动作, 或者奖励函数形式为 $R(s,a,s')$，它还依赖于结果状态。写出这两种形式的最优价值函数的贝尔曼方程。(12 points)