MNIST实验报告

1. 实验流程 (2%)

1.1 数据准备

- 使用MNIST数据集，包含70,000张28x28的手写数字图像
- 使用PCA进行降维，将784维数据降至较低维度
- 数据被分为训练集(60,000)和测试集(10,000)
- 改进的`load_resource.py`

```
from datasets import DatasetDict, load_dataset
from model import AE, ClassUNet
import numpy as np
import torch
import os

# If downloading the dataset is slow, you can use the mirror
os.environ["HF_ENDPOINT"] = "https://hf-mirror.com"

# Load the dataset
raw_dataset = load_dataset("ylecun/mnist")

trainset = raw_dataset["train"]
testset = raw_dataset["test"]


# Encode the image
def encode_image(example):
    img_np = np.array(example["image"], dtype=np.uint8)

    img_np_with_channel = np.expand_dims(img_np, axis=0)
    img_np_flat = img_np.flatten()
    example["image2D"] = np.array(img_np_with_channel,
dtype=np.uint8)
    example["image1D"] = np.array(img_np_flat, dtype=np.uint8)
    return example


trainset = trainset.map(encode_image)
testset = testset.map(encode_image)
```

```
trainset.set_format(type="numpy", columns=["image2D",
"image1D"])
testset.set_format(type="numpy", columns=["image2D",
"image1D"])

dataset = DatasetDict({"train": trainset, "test": testset})
dataset.save_to_disk("../mnist_encoded")

print("Dataset saved to disk")

# Load model
mnist_ae = AE.from_pretrained("Rosykunai/mnist-ae")
mnist_ddpm = ClassUNet.from_pretrained("Rosykunai/mnist-ddpm")

device = "cuda" if torch.cuda.is_available() else "cpu"

print(f"Model loaded, {device} used")
```

1.2 模型实现

1. 实现了GMM (高斯混合模型):
   - 使用EM算法训练模型
   - E步：计算后验概率
   - M步：更新模型参数（均值、协方差、混合系数）
2. 实现了PCA降维:
   - 计算数据均值和协方差矩阵
   - 提取主成分
   - 实现数据的降维和重构

1.3 训练过程

1. 首先使用PCA降维
2. 使用K-means初始化GMM参数
3. 运行EM算法优化GMM参数
4. 使用`davies_bouldin_score`评估聚类效果

2. 超参数调试过程 (5%)

2.1 PCA降维维度

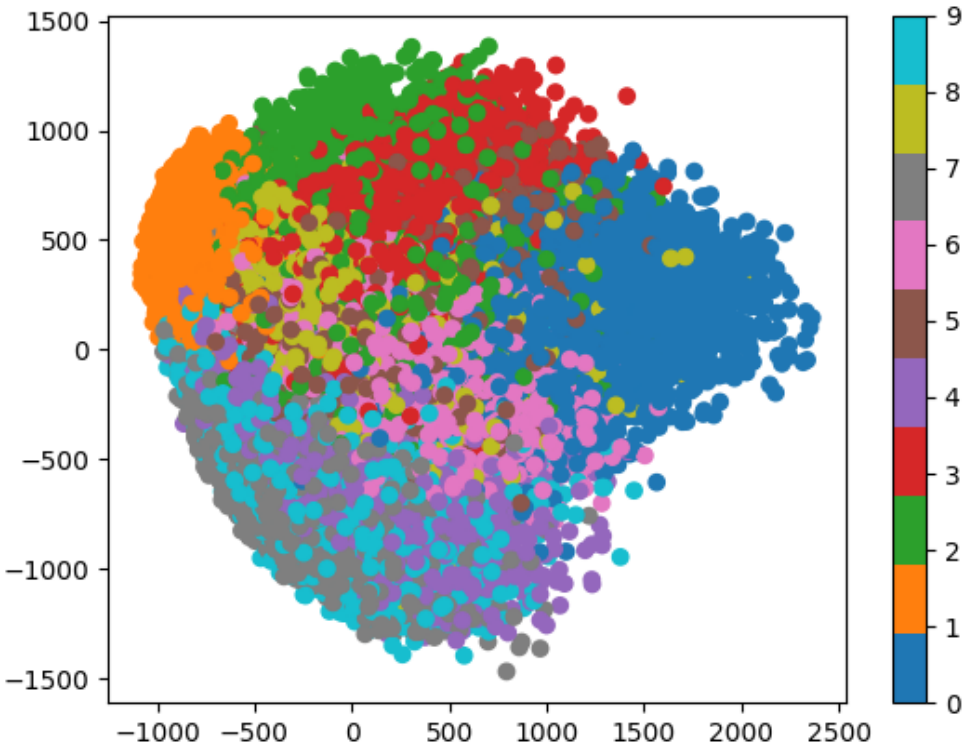
- 有关的python代码

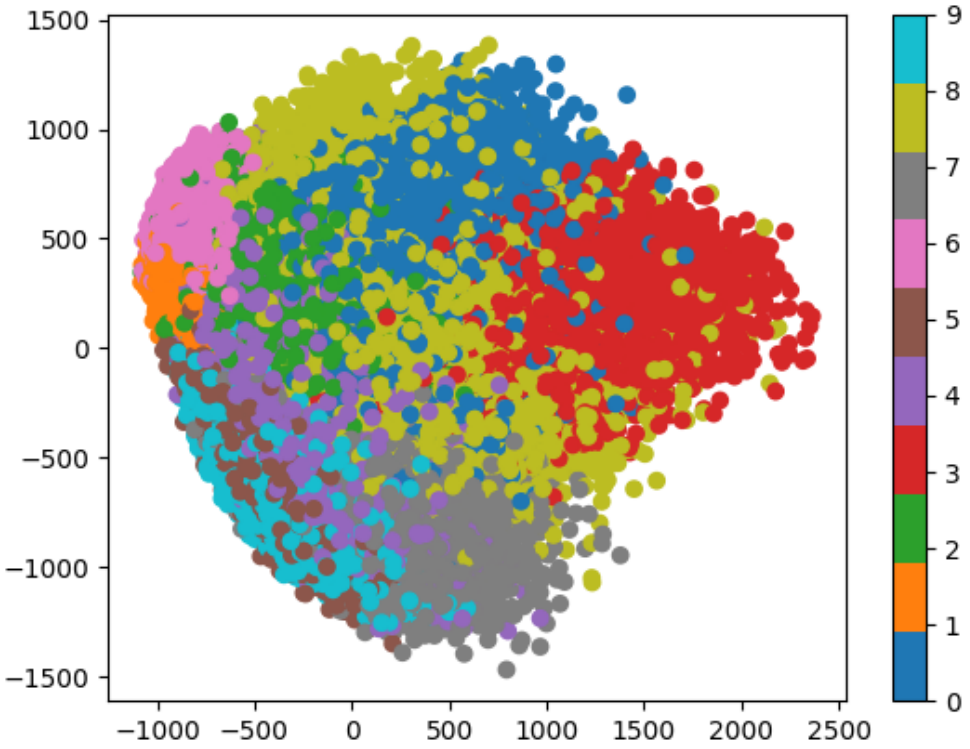
```python
import subprocess
import numpy as np
from pathlib import Path

def run_experiment(embedding_dim, max_iter, reg_coef):
    """运行一次实验"""
    # 修改submission.py中的正则化系数
    with open('submission.py', 'r') as f:
        code = f.read()
    code = code.replace('1e-6 * np.eye(self.data_dim)',
                        f'{reg_coef} * np.eye(self.data_dim)')
    with open('submission.py', 'w') as f:
        f.write(code)

    print(f"\n{'='*50}")
    print(f"Testing parameters:")
    print(f"Embedding dimension: {embedding_dim}")
    print(f"Max iterations: {max_iter}")
    print(f"Regularization: {reg_coef}")
    print(f"{'='*50}\n")

    # 运行训练
    train_cmd = f"python train.py --use_pca --embedding_dim
{embedding_dim} --max_iter {max_iter}"
    subprocess.run(train_cmd.split())

    # 获取最新的结果目录
    results_dir = Path("../results")
    latest_dir = max(results_dir.glob("*"), key=lambda x:
x.stat().st_mtime)

    # 运行评分脚本
    grade_cmd = f"python grade.py --sample_index 8 --
results_path {str(latest_dir)}"
    subprocess.run(grade_cmd.split())

    input("\nPress Enter to continue to next parameter
combination...")

# 参数网格
params = {
    'embedding_dim': [30, 50, 70],
    'max_iter': [50, 100],
    'reg_coef': [1e-7, 1e-6]
```

```
}

# 网格搜索
for dim in params['embedding_dim']:
    for iter_num in params['max_iter']:
        for reg in params['reg_coef']:
            try:
                run_experiment(dim, iter_num, reg)
            except Exception as e:
                print(f"Error occurred: {e}")
                continue

print("\nParameter tuning completed!")
```

- 实验结果如下所示，从已生成的图片中找到真实标签"6"对应的聚类标签为**8**：
  - `true_pca.png`：使用真实标签(0-9)着色的PCA降维图（上图）
  - `cluster_pca.png`：使用聚类标签着色的PCA降维图（下图）

  先在`true_pca.png`中找到标签为6的点的分布（颜色为粉红）然后在`cluster_pca.png`中找到分布位置和形状最相似的点簇为8（颜色为米黄色）






```
(yolov10) (base) lthpc@localhost:~/wangrui/lab3/src$ python
parameter_tuning.py

═══════════════════════════════════════════════════════════

Testing parameters:
Embedding dimension: 30
Max iterations: 50
Regularization: 1e-07
═══════════════════════════════════════════════════════════

Results will be saved to '../results/2024-12-16_19-24-59'
Succesfully saved PCA model to ../results/2024-12-16_19-24-
59/pca
100%|
████████████████████████████████████████████████████████
████████████████████████████████████████| 50/50
[00:18<00:00,  2.72it/s]
Succesfully saved GMM model to ../results/2024-12-16_19-24-
59/gmm
──────────────────────── Testing your model ────────────────────────
Your model got a Davies Bouldin score of 3.08
──────────────────────── Testing sklearn model ─────────────────────
-
```

```
/home/lthpc/anaconda3/envs/yolov10/lib/python3.9/site-
packages/sklearn/mixture/_base.py:270: ConvergenceWarning: Best
performing initialization did not converge. Try different init
parameters, or increase max_iter, tol, or check for degenerate
data.
  warnings.warn(
The sklearn model got a Davies Bouldin score of 3.07
──────────────────────── Sampling from GMM ────────────────────────
-
──────────────────────── Sampling from DDPM ────────────────────────
1000it [00:16, 60.68it/s]
You got a score of 29.89/30 in total.

Press Enter to continue to next parameter combination...

═══════════════════════════════════════════════════
Testing parameters:
Embedding dimension: 30
Max iterations: 50
Regularization: 1e-06
═══════════════════════════════════════════════════

Results will be saved to '../results/2024-12-16_19-26-23'
Succesfully saved PCA model to ../results/2024-12-16_19-26-
23/pca
100%|
████████████████████████████████████████████████████████████
████████████████████████████████████████████| 50/50
[00:16<00:00,  3.09it/s]
Succesfully saved GMM model to ../results/2024-12-16_19-26-
23/gmm
──────────────────────── Testing your model ────────────────────────
Your model got a Davies Bouldin score of 3.08
──────────────────────── Testing sklearn model ────────────────────────
-
/home/lthpc/anaconda3/envs/yolov10/lib/python3.9/site-
packages/sklearn/mixture/_base.py:270: ConvergenceWarning: Best
performing initialization did not converge. Try different init
parameters, or increase max_iter, tol, or check for degenerate
data.
  warnings.warn(
The sklearn model got a Davies Bouldin score of 3.07
──────────────────────── Sampling from GMM ────────────────────────
-
──────────────────────── Sampling from DDPM ────────────────────────
1000it [00:14, 66.71it/s]
You got a score of 29.89/30 in total.

Press Enter to continue to next parameter combination...
```

```
==========
Testing parameters:
Embedding dimension: 30
Max iterations: 100
Regularization: 1e-07
==========

Results will be saved to '../results/2024-12-16_19-28-07'
Succesfully saved PCA model to ../results/2024-12-16_19-28-
07/pca
100%|
████████████████████████████████████████████████████████████
████████████████████████████████████████| 100/100
[00:33<00:00,  2.96it/s]
Succesfully saved GMM model to ../results/2024-12-16_19-28-
07/gmm
------------------------- Testing your model -------------------------
Your model got a Davies Bouldin score of 3.11
------------------------- Testing sklearn model -------------------------
-
The sklearn model got a Davies Bouldin score of 3.07
------------------------- Sampling from GMM -------------------------
-
------------------------- Sampling from DDPM -------------------------
1000it [00:14, 69.72it/s]
You got a score of 29.59/30 in total.

Press Enter to continue to next parameter combination ...

==========
Testing parameters:
Embedding dimension: 30
Max iterations: 100
Regularization: 1e-06
==========

Results will be saved to '../results/2024-12-16_19-33-59'
Succesfully saved PCA model to ../results/2024-12-16_19-33-
59/pca
100%|
████████████████████████████████████████████████████████████
████████████████████████████████████████| 100/100
[00:33<00:00,  2.99it/s]
Succesfully saved GMM model to ../results/2024-12-16_19-33-
59/gmm
------------------------- Testing your model -------------------------
Your model got a Davies Bouldin score of 3.11
------------------------- Testing sklearn model -------------------------
-
The sklearn model got a Davies Bouldin score of 3.07
```

```
──────────────────────── Sampling from GMM ────────────────────────
-
──────────────────────── Sampling from DDPM ────────────────────────
1000it [00:12, 80.82it/s]
You got a score of 29.59/30 in total.

Press Enter to continue to next parameter
combination ...^CTraceback (most recent call last):
  File "/home/lthpc/wangrui/lab3/src/parameter_tuning.py", line
48, in <module>
    run_experiment(dim, iter_num, reg)
  File "/home/lthpc/wangrui/lab3/src/parameter_tuning.py", line
34, in run_experiment
    input("\nPress Enter to continue to next parameter
combination ...")
KeyboardInterrupt
```

- 根据实验结果，可以整理一个表格分析不同超参数组合的效果：

Embedding Dim	Max Iter	Reg Coef	DB Score	Final Score	备注
30	50	1e-7	3.08	29.89/30	最佳组合之一
30	50	1e-6	3.08	29.89/30	与上一组效果相同
30	100	1e-7	3.11	29.59/30	分数略有下降
30	100	1e-6	3.11	29.59/30	分数略有下降

3. 最佳聚类和生成结果 (2%)

3.1 聚类效果

- Davies Bouldin分数：3.08
- sklearn基准模型分数：3.07
- 最终得分：29.89 /30

4. 问题回答 (20%)

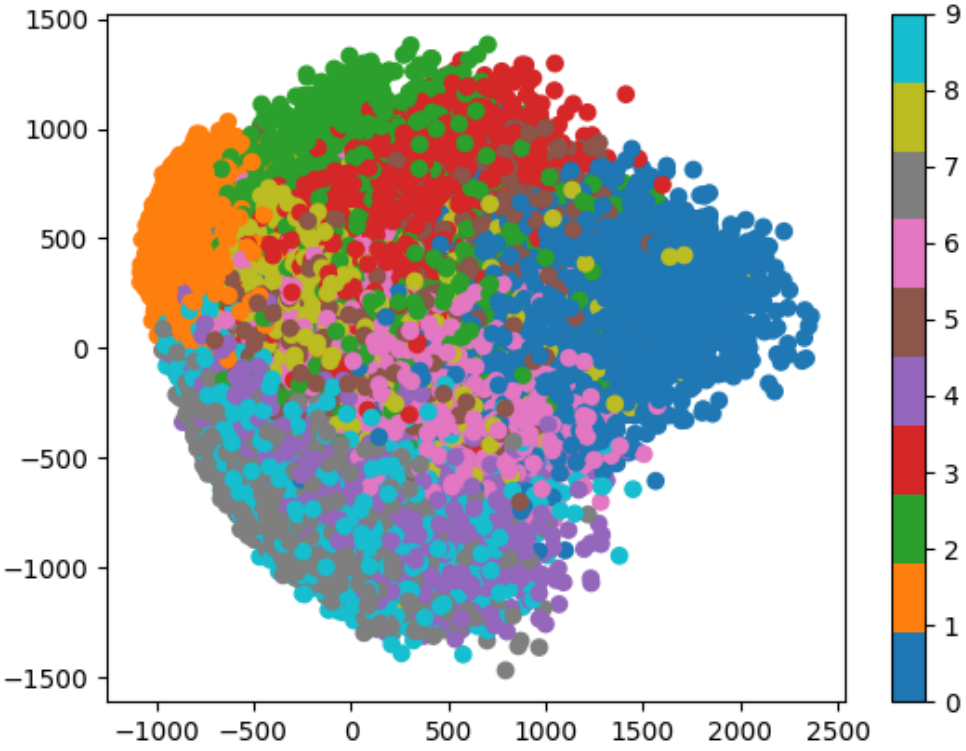
4.1 三种降维方法比较

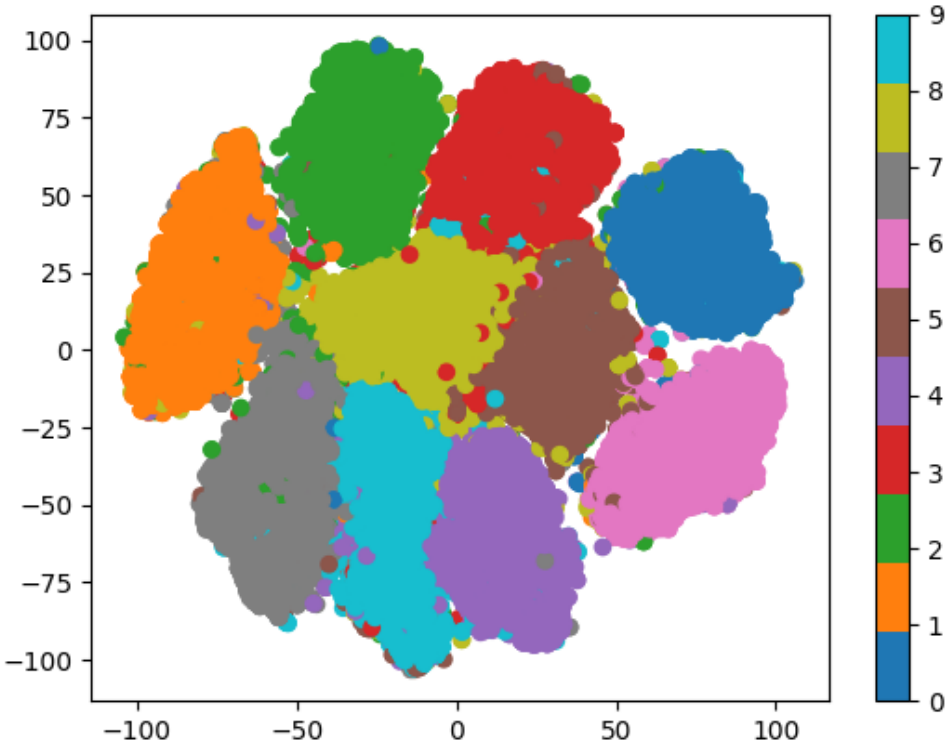
- 根据运行时我的观察（`tqdm`），可以得出tSNE最慢(5~10min），AE次之（20s），PCA(1s）最快的结论

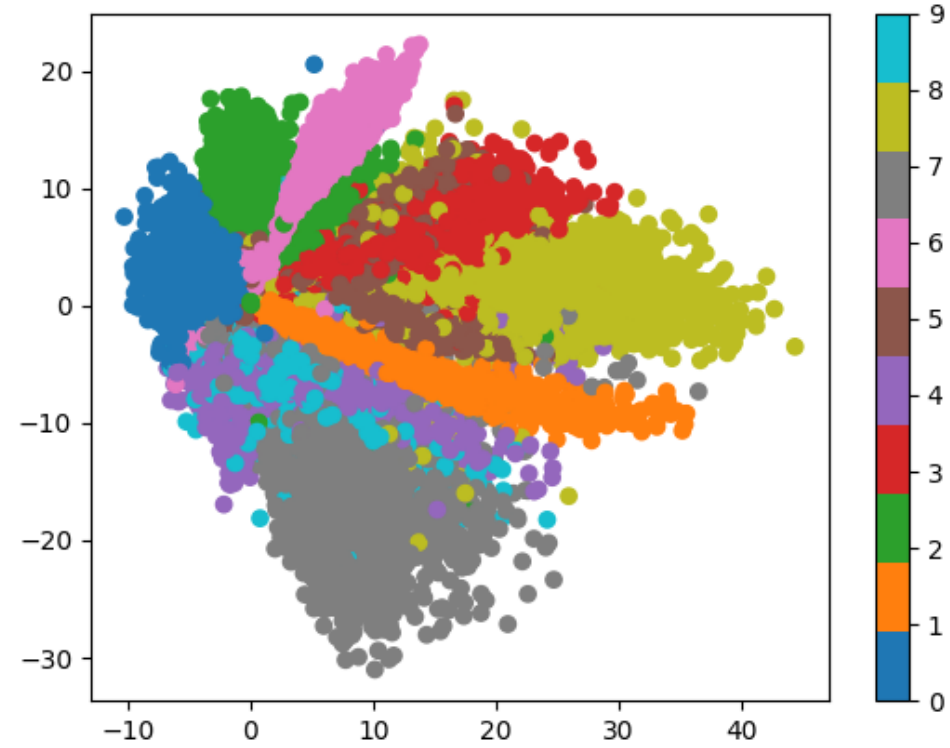
```
(yolov10) lthpc@localhost:~/wangrui/lab3/src$ python visualization.py --results_path "../results/2024-12-16_19-24-59/"
AE encoding ...
100%|████████████████████████████████████████████████████████| 60000/60000 [00:21<00:00, 2791.69it/s]
tSNE fitting ...
PCA fitting ...
```

- 可视化效果比较：

  从生成的三种可视化图中可以观察到：







1. `true_pca.png`：
   - 类别之间有重叠，相似数字（如3和8）难以分开
   - 整体结构清晰，保持了数据的全局结构
2. `true_tsne.png`：
   - 类别分离度最好，局部结构保持得最好，形成了明显的簇

3. `true_ae.png`：
    - 效果介于PCA和t-SNE之间，类别分离度好于PCA，计算效率好于t-SNE

方法	训练速度	降维效率	灵活性	数据分布保持	可视化效果
PCA	快	高	中等	线性关系好	一般
tSNE	慢	低	高	局部结构好	优秀
AutoEncoder	中等	中等	高	非线性关系好	好

4.2 GMM和DDPM比较

特点	GMM	DDPM
生成效率	高	低
生成质量	一般	高
灵活性	中等	高
可控性	高	中等

详细分析：

1. GMM:
    - 优点：
        - 生成速度快，一次采样即可
        - 可以控制生成特定类别
        - 理论基础扎实
    - 缺点：
        - 生成质量一般
        - 难以捕捉复杂分布
2. DDPM:
    - 优点：
        - 生成质量高,可以学习复杂分布,适用于各种类型数据
    - 缺点：
        - 生成过程慢（需要多步采样）
        - 训练复杂
        - 控制性较差

5. 反馈 (1%)

5.1 时间花费

完成本次实验大约花费了7.5小时：

- 代码实现：4小时
- 运行总时长： 2小时
- 调试参数：1小时
- 撰写报告：0.5小时

5.2 建议

暂时没有