电子科技大学 计算机 学院

标 准 实 验 报 告

（实验）课程名称 人工智能综合实验 I

学生姓名：刘洋岑　　学 号：2020080601018

指导教师：姬艳丽

实验地点：主楼 A2-412

实验时间：2022.3.16

电子科技大学教务处制表

电 子 科 技 大 学

实 验 报 告

一、实验项目名称：SVM 实现分类任务

二、实验内容：

实验一：编程实现 SVM(SMO)分类器模型，在 Iris 数据集上验证 SVM 分类器。
（1）实现 kernel 不同取值时分类实验，分析实验结果。
（2）采用 欧几里得核函数，高斯核函数等 kernel 计算验证 SMO 分类结果，并分析讨论。
实验二：在 MNIST 数据集上实现 SVM 分类。
（1）实现 kernel 不同取值时分类实验，分析实验结果。
（2）分别使用 PCA 以及 CNN 中的 pooling 层进行 downsample 对于数据集进行降维处理。

三、实验算法设计：

实验一

自己手写实现 SMO，在 IRIS 数据集进行不同核函数的实验。其中还进行了一个圆形分布的数据集的实验，可以更加直观的展示核函数的作用（测试核函数效用，**此处引用李沐轩(2020080602022)提供的他自行生成的“circle”测试数据**）

实验二

使用自己手写的 SMO，以及 sklearn 中的 SMO 来进行实验，对于 MNIST 数据集使用了降采样方法，得到了 PCA 的碎石图，进行了数值分析。

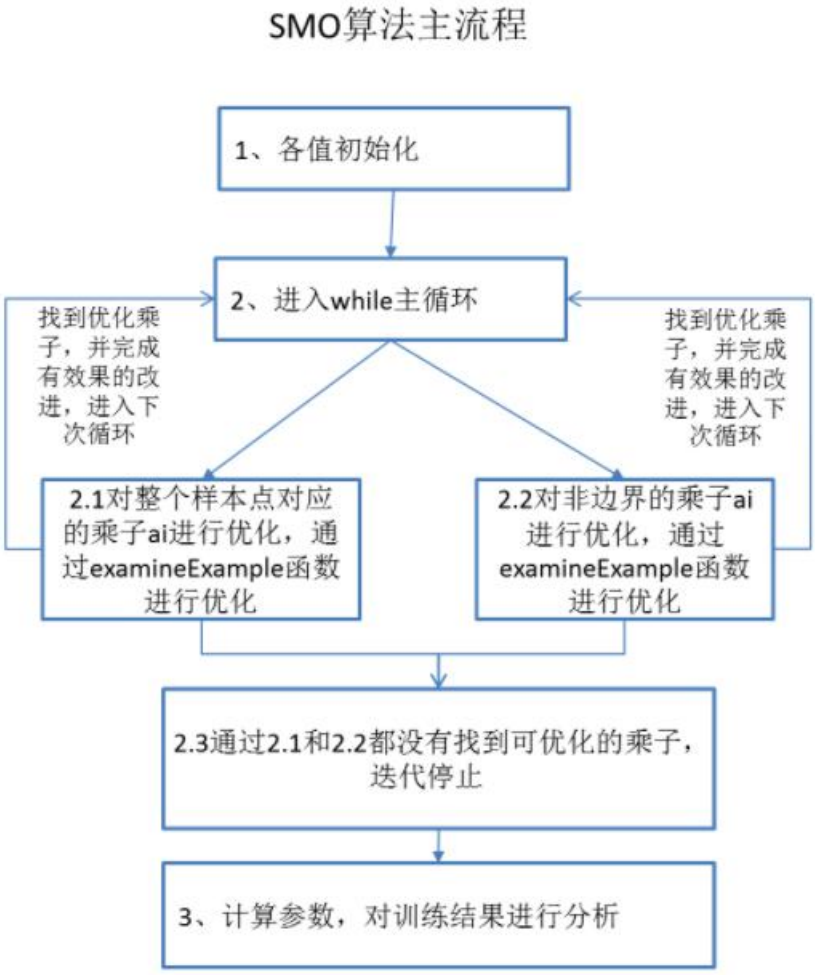
注意：本报告中部分未加说明引用的图片带有 CSDN 水印，因为我就是博客作者（不是抄袭）

四、算法及创新：

代码量稍显庞大，写成了仓库形式，所以这里给出链接：

https://github.com/Randle-Github/Machine-Learning-Experiment/tree/main/experiment4and5

SMO：流程转载自 https://www.csdn.net/tags/MtTaEgxsOTc1OTkwLWJsb2cO0O0O.html



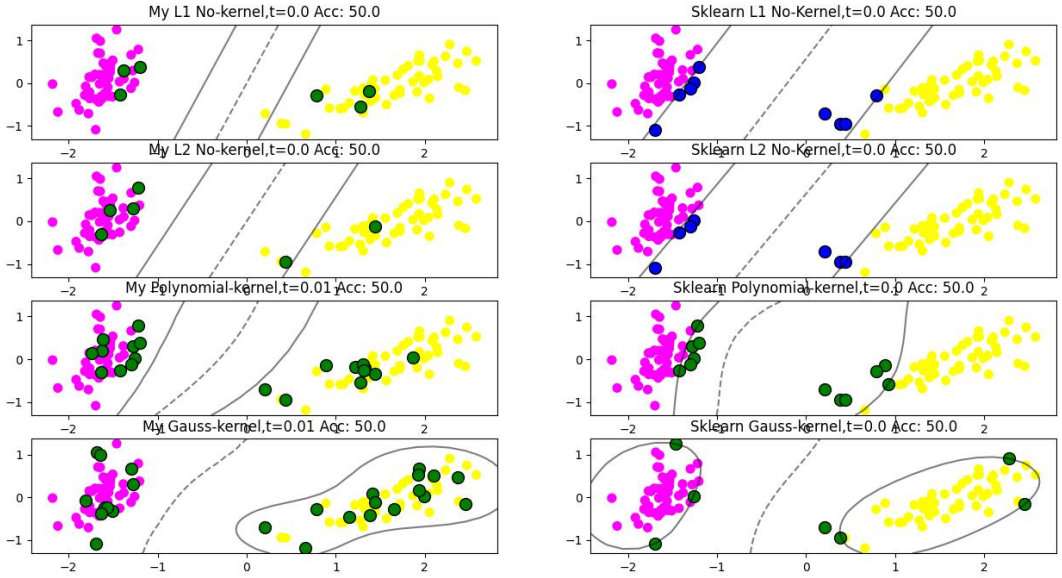
个人创新：

除了基本的手写 SMO 任务，在对于 MNIST 数据集进行处理时，使用了两种降维方式进行实验，加入了平均 pooling 的方式，以及使用了 PCA 来进行实验。并且在可视化方面做了一个较为复杂的对比实验。

五、实验数据及结果分析：

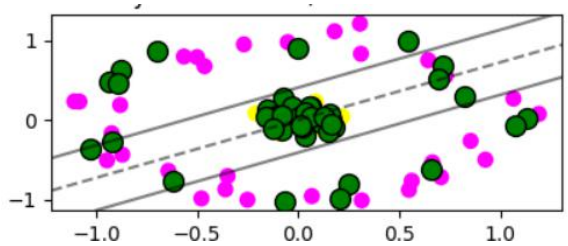
实验一：

手写(与 sklearn 对比）实现 SMO，在 Iris 数据集进行。由于 Iris 有 3 个类别，我只取了前 2 个类别。为了可视化，我将 Iris 的 4-dim 改成了 2-dim（使用 PCA 降维方法）。我将软间隔和超平面以及分类数据都进行了可视化。

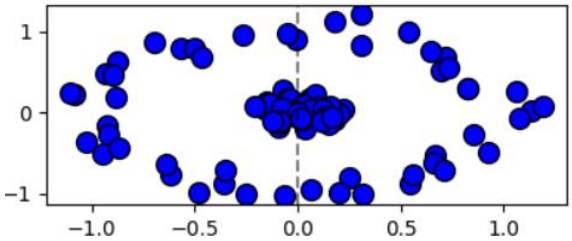


非常神奇的是，所有的结果都是 0.5，应该是我参数调的不好。对于参数是后面研究的内容，所以这里没有去调试了。可视化效果可以看出，由于数据集过于简单，加入核函数其实用处不大。感觉在 Iris 上面做研究意义不大，后面做 mnist 多做了一些实验。

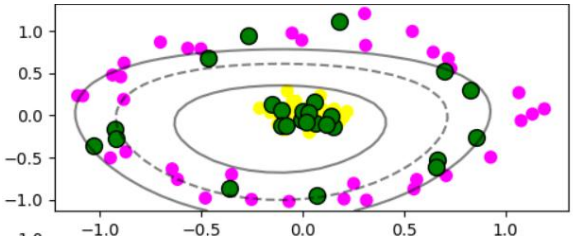
My_SMO,无核,L1,acc=0.53



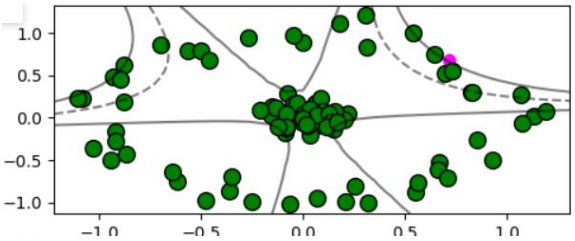
sklearn,无核,L1,acc=0.53



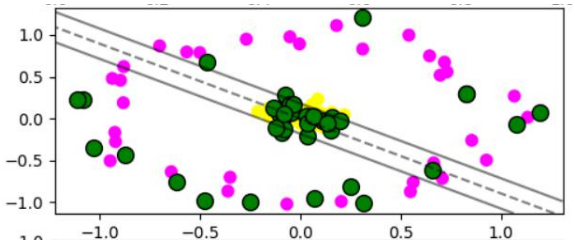
My_SMO,Poly,acc=0.50



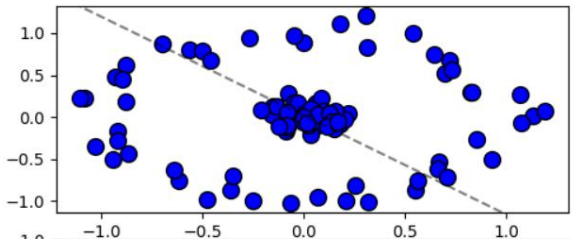
sklearn,Poly,acc=0.50



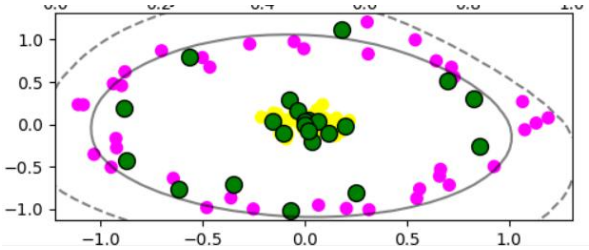
My_SMO,无核，L2,acc=0.5



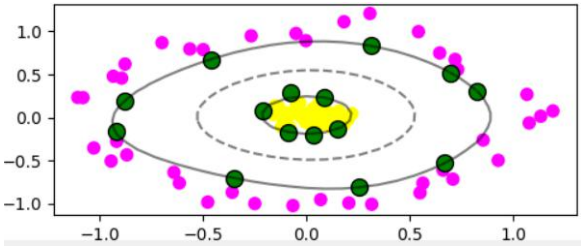
sklearn,无核，L2,acc=0.5



My_SMO,Gauss,acc=0.62



sklearn,Gauss,acc=1.00



实验二：

Mnist 数据集是一个人类手写数字识别的数据集，含有 60000 张训练集和 10000 张测试集。我这里主要测试的是降维模式对于结果的影响（核函数的影响在上文已经研究过了）。每一个样本使用了 28*28 的 size（有的版本是 32*32），包含一个通道的灰度值（0 到 255）。

Mnist 数据集数字 1 的展示。

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 124 253 255 63 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 96 244 251 253 62 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 127 251 251 253 62 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 68 236 251 211 31 8 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 60 228 251 251 94 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 155 253 253 189 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 20 253 251 235 66 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 32 205 253 251 126 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 104 251 253 184 15 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 80 240 251 193 23 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 32 253 253 253 159 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 151 251 251 251 39 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 48 221 251 251 172 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 234 251 251 196 12 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 253 251 251 89 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 159 255 253 253 31 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 48 228 253 247 140 8 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 64 251 253 220 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 64 251 253 220 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

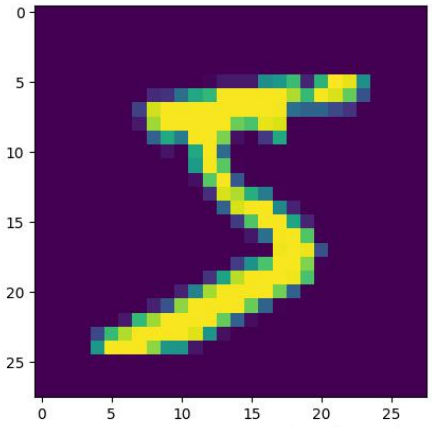
0 0 0 0 0 0 0 0 0 24 193 253 220 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

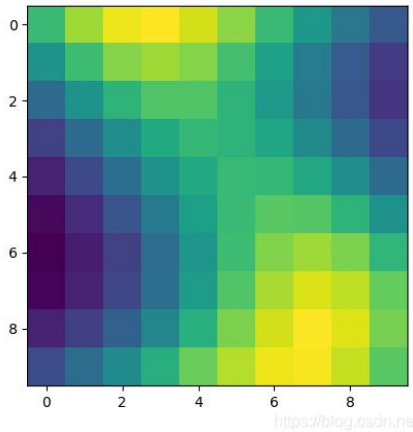
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

下面是一个 5 的可视化展示：



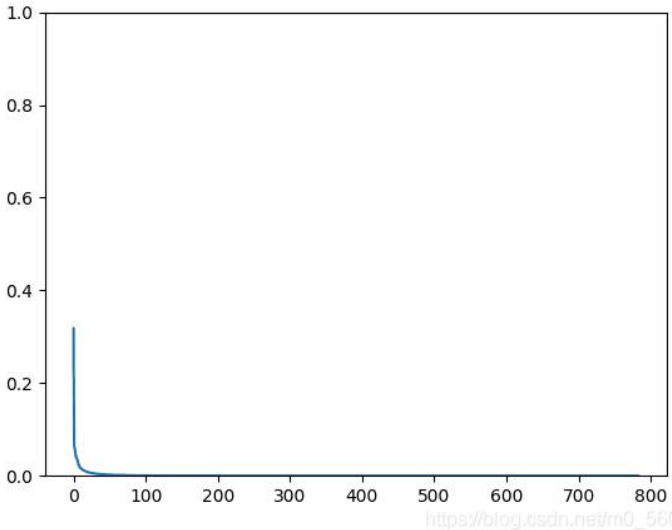
然而 SVM 是一种计算资源耗费并不小的方法，随着 dimension 递增，复杂度将会非常严重，所以在后面是使用了后文当中我主要研究的是降维方式对于数据集最后准确率的影响，也简单的使用了多种核函数来进行实验，但是使用了两种不同的降维方法：pooling 和 PCA。

1.Pooling

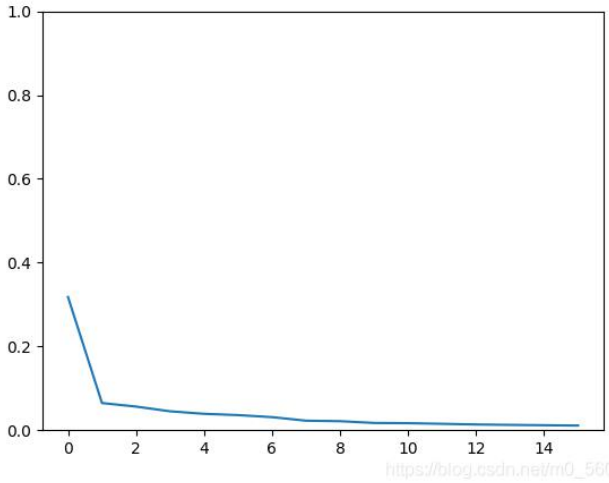


上图是我对于 5 进行一个简单的 average pooling 来做的结果。虽然已经肉眼难以辨认了，确实有很大的信息损失的问题，但是这个有一定的降噪功能，并且在本地机型可以进行测试。

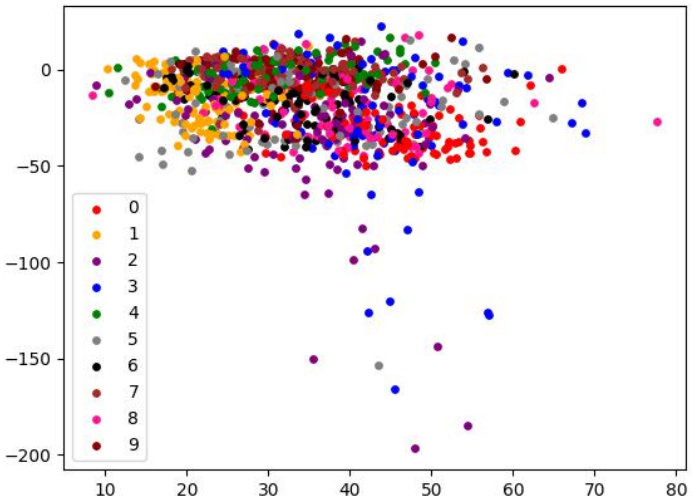
2.PCA



上图是我进行主成分分析之后的一个主成分碎石图（PCA 是我手写的，基于 SVD 方法，所以能够看到），提取前面的 15 个：



发现信息损失较为严重，最后和 pooling 使用了相同的方法，降到前 100 个主成分的维度。可视化 10 个类别之后的结果如下：



似乎是能有一点区分度的，但是类别比较多，损失太大（前两个特征加起来只有 20%）。我保留前 100 个特征可以保存 96%的信息，降低复杂度的同时还能去噪。

实验结果：

Gaussian, pooling 降维方法: 95.0%

Gaussian, PCA 降维方法: 96.7%

Polynomial, pooling 降维方法: 83.6%

Polynomial, PCA 降维方法: 87.2%

可以看出高斯核函数和 PCA 相比另外一种在 mnist 上更具优越性，并且效果几乎能够达到 100%，非常优秀。

六、总结及心得体会：

本周由于时间忙碌，本来想把上上次作业的人脸识别映射出来的数据集拿来用 SVM 做

的，当时用的 KNN，我认为 SVM 由于能够设置一个软间隔项，所以具备更好的优越性。但是时间问题没有实行。

本次感觉核函数的影响挺大的，在一共四个数据集上发现高斯核函数效果都最好，这点有待思考。

在遇到如 mnist 这种级别的数据集的时候，才 400 多维，基本就很难处理了。并且从 SMO 方法的原理来看，应该用硬件加速效果没有那么显著。所以在大数据时代，比如我最近一直在做视频的项目，那个一个样本就是 32 帧 3 个通道 224*224 的情况，使用 SVM 这种简单的机器学习确实很不切实际了。但是对于小数据集 SVM 或许仍然具备生存空间。

七、对本实验过程及方法、手段的改进建议：

改进暂无。

报告评分：

指导教师签字：