《大数据机器学习》第 1 次实验

姓名：刘培源　　学号：2023214278

题目 1：

SVM 的实现。熟悉 SVM 的原理与优化求解 SVM 分类器的算法的过程。

答：

支持向量机（SVM）旨在找到一个超平面，最大化两类数据的边界间隔。线性可分时，超平面定义为：

$$w^T x + b = 0$$

其中，w 是权重向量，b 是偏置项。

SVM 的基础形式是通过以下方式优化来找到该超平面：

$$\min_{w,b} \frac{1}{2}\|w\|^2$$

$$s.t. \quad y_i(w^T x_i + b) \geq 1 \quad \forall i$$

当存在不可分数据时，通过松弛变量 ξ 和惩罚系数 C 进行调整：

$$\min_{w,b,\xi} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{m} \xi_i$$

$$s.t. \quad y_i(w^T x_i + b) \geq 1 - \xi_i \quad \forall i, \xi_i \geq 0$$

为处理非线性数据，SVM 可以使用核技巧，如 RBF 核，将数据映射到高维空间。

SMO 算法：Sequential Minimal Optimization (SMO) 算法是解决 SVM 优化问题的一种有效方法。其主要思路是每次只优化两个拉格朗日乘子 α。

1. 选择违反 KKT 条件最严重的乘子 α_1。2. 选择使得目标函数变化最大的 α_2 作为第二个乘子。3. 固定其它乘子，只优化 α_1 和 α_2。为此，引入边界 L 和 H，以保证约束条件。4. 根据新的乘子值更新阈值 b 和误差缓存。5. 重复上述步骤直到所有乘子满足 KKT 条件。

在代码实现中，binarySVM 类处理二分类问题，采用 SMO 算法更新乘子。a1_search 选择第一个乘子，a2_search 选择第二个乘子，然后基于 eta 更新这两个乘子。kkt_cond 用于检查 KKT 条件。另外，类还实现了线性和 RBF 核函数。

对于多分类问题，multiclassSVM 类实现了 “one-vs-one” 和 “one-vs-all” 策略。

题目 2：

MNIST 数据集分类。利用 MNIST 训练集训练、验证，在测试集测试，报告你在测试集上的准确率，并使用现成的库实例化的一个 SVM 分类器（python 下如 sklearn），并同样在 MNIST 上进行测试和训练，对比这个分类器与你设计的 SVM 分类器的结果，如果你设计的分类器效果相较于这个分类器的准确率较低的话，请分析一下原因。

答：

基于上面的分析，我们实现了两个类：binarySVM 用于二分类，multiclassSVM 采用 "one-vs-one" 或 "one-vs-all" 进行多分类任务。binarySVM 和 multiclassSVM 的实现伪代码如下：

Algorithm 1: 二分类 SVM 的 SMO 算法

Data: 训练集 (x, y), 松弛变量惩罚系数 C, 容忍度 KKT_tol, 核函数 kernel
Result: 拉格朗日乘子 a, 偏置 b, 支持向量 sv

初始化 a 为零向量;
初始化偏置 $b = 0$;
选择不满足 KKT 条件的 a_1;
while a_1 存在 **do**
    选择 a_2 以最大化 $|E1 - E2|$;
    根据 a_1 和 a_2 计算边界 L 和 H;
    if $L \neq H$ **then**
        计算 η;
        更新 a_1 和 a_2;
        更新 b;
    end if
    选择不满足 KKT 条件的新的 a_1;
end while
return a, b, 支持向量;

Algorithm 2: 多分类 SVM 的算法

Data: 训练集 (x, y), 类别数量 num_cls, 策略 strategy
Result: 训练好的多分类 SVM 模型

初始化一个空的二分类 SVM 列表 binarySVM;
if *strategy* 为 *ovo* **then**
    for 每一对不同的类别 (cls_a, cls_b) **do**
        训练一个二分类 SVM 模型用于区分 cls_a 和 cls_b;
        将该 SVM 模型加入 $binarySVM$ 列表;
    end for
end if
else if *strategy* 为 *ova* **then**
    for 每一个类别 cls **do**
        训练一个二分类 SVM 模型用于区分 cls 和其他类;
        将该 SVM 模型加入 $binarySVM$ 列表;
    end for
end if
return $binarySVM$;

基于 multiclassSVM，我们训练了 MNIST[1]数据集，并在其测试集上进行了测试。训练

[1]http://yann.lecun.com/exdb/mnist/

代码如下：

```
classifier = multiclassSVM(10,
                           display=True,
                           C=10,
                           KKT_tol=0.1,
                           epochs=1000,
                           kernel_type="linear")
classifier.fit(tr_img.reshape((-1, 28**2))[:2000], tr_lbl[:2000])

y_pred = classifier.pred(te_img.reshape((-1, 28**2)))
acc = np.mean(y_pred==te_lbl.astype(np.uint8))
print(f"多分类SVM的推理准确率为：{acc:.2%}")
```

超参数选取如表1所示：

超参数	数值	含义
C	10	松弛变量惩罚系数
KKT_tol	0.1	KKT 条件的容忍度
epochs	1000	最大 epoch 的数目，-1 代表训练直到收敛
kernel_type	linear	核函数类型

Table 1: 超参数选取（linear 核函数）

由于本地计算资源与计算时间的限制，只选取了训练集中的前 2000 张图片，结果如下：

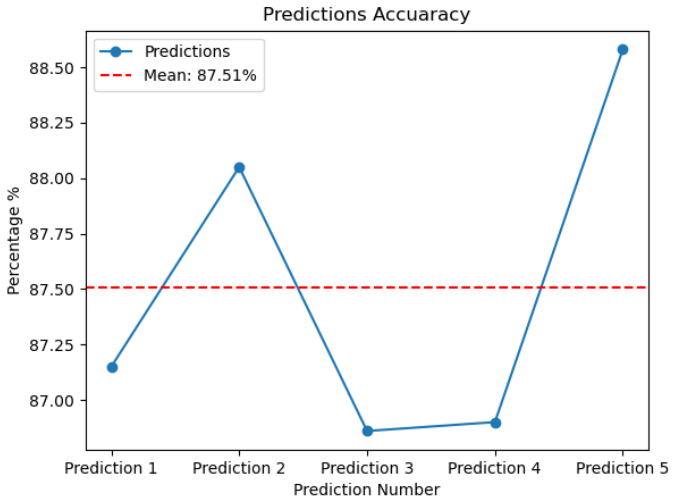

Figure 1: 5 次实验的准确率（linear 核函数）

从图1可以看出，5 次实验的平均预测准确率为 87.51%。

接下来，实现基于 sklearn 库中 svm 对于 MNIST 数据集的分类。为了确保对比的公平性，同样只选取了训练集中的前 2000 张图片，以及迭代了 1000 次，代码如下：

```
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score

clf = SVC(kernel='linear',
          C=1,
          max_iter=1000,
          decision_function_shape='ovo')
clf.fit(tr_img.reshape((-1, 28**2))[:2000], tr_lbl[:2000])

y_pred = clf.predict(te_img.reshape((-1, 28**2)))

accuracy = accuracy_score(te_lbl, y_pred)
print(f"基于sklearn多分类SVM的推理准确率为：{accuracy * 100:.2f}%")
```

基于 sklearn 的 svm 的平均预测结果为 89.16%，略高于我的 multiclassSVM 的实现。原因分析如下：我实现的 binarySVM 的 SMO 算法中对于违反 KKT 条件的 alpha1 的选取，是采用遇到的第一个，而不是看完所有的 alpha，选取违反程度最大的。同时，我选取完之后，会把不符合 KKT 条件的列表，进行随机的打乱，这有可能会影响收敛的速度。

题目 3：

对 SVM 进行超参数的搜索。对 SVM 分类器进行超参数的搜索 (比如收敛终止条件，学习率等)，选取你的最优的超参数组合下的 SVM 分类器的优化结果。

答：

我实现了一个 KFold 的参数搜索方法。在 epoch 为-1（即搜索直到所有 alpha 均满足 KKT 条件）的条件下，搜索 C 值在 $\{0.01, 10, 100\}$ 和 KKT_tol 值在 $\{0.01, 0.1, 0.5\}$ 之间的组合，发现当 $(C, \text{KKT\_tol}) = (0.01, 0.01)$ 时，模型表现最好，平均为 89.56%。(注意到此值已经可以与 sklearn 的 svm 实现相媲美。)

Bonus1：构建你的使用 kernel 方法的 SVM 分类器，以及相关的训练优化流程，利用 MNIST 训练集训练、验证，在测试集测试，报告你在测试集上的准确率。

答：

在我的 binarySVM 的实现中已经有对于 RBF 的支持，所以我只需要调用 multiclassSVM 设置 kernel_type 为 rbf，并且设置合适的 rbf 的 gamma 值即可，代码如下：

```
classifier = multiclassSVM(10,
                           display=True,
                           C=10,
                           KKT_tol=1e-2,
                           epochs=-1,
                           kernel_type="rbf",
                           gamma_rbf=0.01)
classifier.fit(tr_img.reshape((-1, 28**2))[:2000], tr_lbl[:2000])

y_pred = classifier.pred(te_img.reshape((-1, 28**2)))
acc = np.mean(y_pred==te_lbl.astype(np.uint8))
print(f"多分类SVM(RBF核)的推理准确率为：{acc:.2%}")
```

超参数选取如表2所示：

超参数	数值	含义
C	10	松弛变量惩罚系数
KKT_tol	0.01	KKT 条件的容忍度
epochs	-1	最大 epoch 的数目，-1 代表训练直到收敛
kernel_type	rbf	核函数类型
gamma_rbf	0.01	rbf 核函数的 gamma 值

Table 2: 超参数选取（RBF 核函数）

由于本地计算资源与计算时间的限制，只选取了训练集中的前 2000 张图片，结果如下：

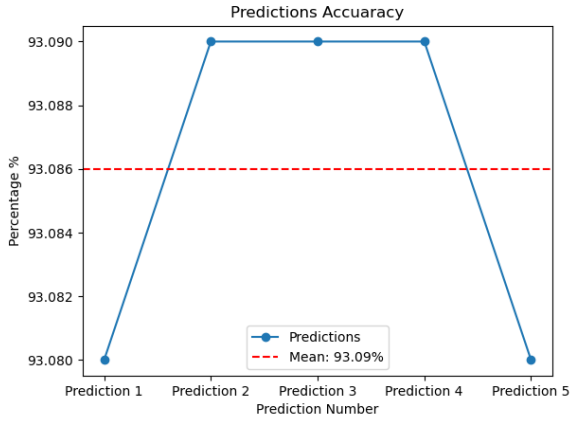

Figure 2: 5 次实验的准确率（RBF 核函数）

从图2可以看出，5 次实验的平均预测准确率为 93.09%。我还进一步验证了 sklearn 中 rbf kernel 的准确率，在 C 和 epochs 一致的情况下，准确率为 93.44%。与我的实现几乎一致。

Bonus2： 对比一下未使用 kernel 方法的 SVM 分类器和不同 kernel 方法下的 SVM 分类器，分析一下不同方法间的区别.(比如可视化降维后的数据在不同方法下的分类界面)

答：

为了更好的格式化多个核函数，我采用了 sklearn 的 svm 的实现，对比了四种不同的核函数 Linear, Polynomial, RBF 和 Sigmoid 核函数。具体来说，我先通过 PCA 主成分分析将训练的图片从 28×28 维，降维成 2 维，方便可视化。随后选取了训练集中前 200 个样本，分别用四个不同的核函数训练，并且可视化，代码如下：

```
def plot_svm_kernels(X, y):
    x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1
    y_min, y_max = X[:, 1].min() - 1, X[:, 1].max() + 1
    xx, yy = np.meshgrid(np.arange(x_min, x_max, .02),
                         np.arange(y_min, y_max, .02))

    titles = ['Linear', 'RBF', 'Polynomial', 'Sigmoid']

    plt.figure(figsize=(10, 10))

    for i, kernel in enumerate(('linear', 'rbf', 'poly', 'sigmoid')):
        clf = svm.SVC(kernel = kernel, gamma = 2)
        clf.fit(X, y)
        plt.subplot(2, 2, i + 1)
        plt.subplots_adjust(wspace = 0.4, hspace = 0.4)

        Z = clf.predict(np.c_[xx.ravel(), yy.ravel()])
        Z = Z.reshape(xx.shape)
        plt.contourf(xx, yy, Z, alpha=0.8)

        plt.scatter(X[:,0], X[:,1], c=y, edgecolors='k', marker='o')
        plt.xlabel('Feature 1')
        plt.ylabel('Feature 2')
        plt.title(titles[i])

    plt.show()

plot_svm_kernels(X_sample, y_sample)
```

四种不同核函数预测结果可视化如图3：

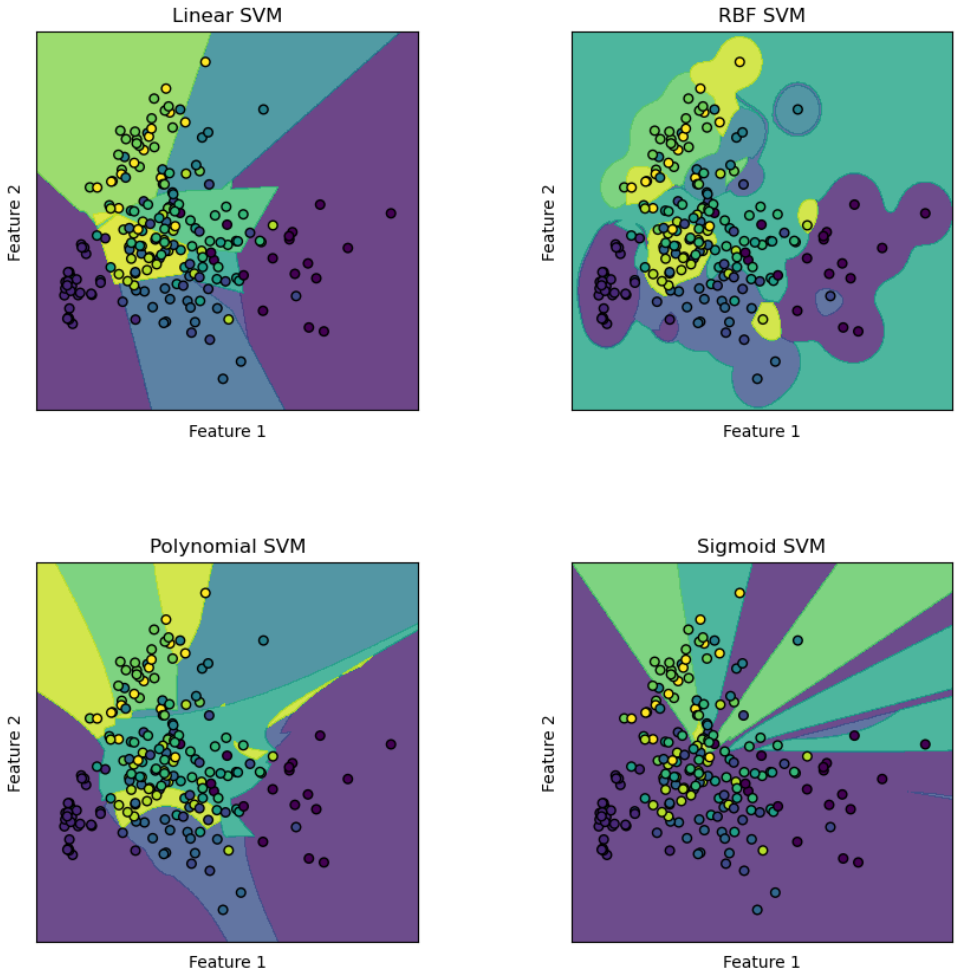

Figure 3: 四种不同核函数预测结果（200 个样本）

分析

1. Linear 核是 SVM 在高维空间中的原始形式，不涉及任何映射。当数据是线性可分的，或者问题的维度很高而样本数量相对较少时，线性核是很好的选择。
   (a) 公式：$K(u, v) = u^T v$
   (b) 优点：计算效率高。
   (c) 缺点：对非线性问题的建模能力不足。
2. Polynomial 将输入数据映射到高维空间，使得非线性数据变得线性可分。
   (a) 公式：$K(u, v) = (gamma * u^T v + r)^d$
   (b) 优点：可以捕捉数据中的多项式型模式。
   (c) 缺点：当度数 d 很高时，可能会导致过拟合；计算量也相对较大。
3. RBF 又叫高斯核，可以将每一个样本点映射到一个无穷维的空间。

(a) 公式：$K(u,v) = e^{-gamma*||u-v||^2}$

(b) 优点：非常适用于大部分非线性问题。

(c) 缺点：对 gamma 很敏感。

4. Sigmoid 核与神经网络的双曲正切激活函数类似

(a) 公式：$K(u,v) = \tanh(gamma * u^T v + r)$

(b) 优点：可以模拟神经网络的学习方式。

(c) 缺点：在某些条件下，Sigmoid 核不是正定的，可能导致 SVM 没有解。