机器学习与数据挖掘 作业1实验报告

SVM模型的一般理论

SVM 的核心思想是通过寻找一个超平面将数据点分割为不同的类别，以最大化类别之间的边界。SVM 主要有以下几个关键理论点：
SVM 模型的目标是找到一个超平面使样本分成两类，并且间隔最大。
1. 在二分类问题中，超平面是一个线性决策边界，其方程为:$w \cdot x+b=0$其中，w是超平面的法向量，b是偏置。
2. 最大间隔：SVM 通过最大化数据点到超平面的距离来选择最佳超平面。这种最大化间隔的策略可以使得模型具有更好的泛化能力，从而减少过拟合。
3. 正则化项：为了避免过拟合，SVM 引入了正则化项，通常是权重的平方和，用来控制模型复杂度.

采用不同核函数的模型和性能比较及分析

1.线性核函数： 线性核函数假设数据是线性可分的，即通过一个超平面即可将数据分开。在实际应用中，线性核通常适用于数据具有明显的线性分布的情况。线性核的计算复杂度较低，训练速度较快，但对于复杂的非线性数据，可能无法得到较好的分类效果。
2.高斯核函数： 而高斯核函数则能处理更复杂的非线性问题。它将原始数据映射到更高维的特征空间，从而能够找到更复杂的超平面进行数据分类。高斯核的优点是可以处理高度非线性的数据，但它需要调整的超参数较多，且计算复杂度较高。

在本实验中，使用线性核和高斯核的SVM模型进行了比较。
代码如下：

```
# --- 使用现成的SVM软件包训练模型 ---
# 使用线性核函数训练 SVM
start_time = time.time()  # 记录训练开始时间
svm_linear = SVC(C=1, kernel="linear", gamma="auto")  # 创建线性核的SVM模型
svm_linear.fit(X_train, y_train)  # 在训练集上训练SVM模型
y_pred_linear = svm_linear.predict(X_test)  # 在测试集上进行预测
accuracy_linear = accuracy_score(y_test, y_pred_linear)  # 计算并输出模型的准确率
linear_train_time = time.time() - start_time  # 计算训练时长

#使用高斯核训练 SVM
start_time = time.time()  # 记录训练开始时间
svm_rbf = SVC(C=1, kernel="rbf", gamma="auto")  # 创建高斯核（RBF）SVM模型
svm_rbf.fit(X_train, y_train)  # 在训练集上训练SVM模型
y_pred_rbf = svm_rbf.predict(X_test)  # 在测试集上进行预测
accuracy_rbf = accuracy_score(y_test, y_pred_rbf)  # 计算并输出模型的准确率
rbf_train_time = time.time() - start_time  # 计算训练时长
```

运行结果如下：

```
[Running] python -u "e:\robotstudy\rstudy\test.py"
线性核函数准确率: 0.9995 (Training time: 0.5697 seconds)
高斯核函数准确率: 0.9972 (Training time: 2.4679 seconds)
```

从运行结果可以看出，高斯核的SVM的训练时间和计算成本比线性核的SVM高，且可能因为数据集不是很复杂，故线性核函数的准确率要高于高斯核。

采用 hinge loss 的线性分类模型和 SVM 模型之间的关系

Hinge Loss 是 SVM 中用于分类任务的主要损失函数。SVM 通过最小化 hinge loss 来优化模型，目标是让正确分类的样本尽可能远离超平面。SVM 中的 hinge loss 定义为：$L(y,f(x))=\max(0,1-y\cdot f(x))$其中$y$是真实标签（-1 或 1），$f(x)$是预测值，表示样本$x$到超平面的距离。SVM 就是通过优化hinge loss来寻找最佳的分类超平面，并且通过正则化来控制模型复杂度。

采用 hinge loss 线性分类模型和 cross-entropy loss 线性分类模型比较

Hinge Loss：

Hinge Loss 主要用于SVM中，是SVM中的核心损失函数。它通过最大化数据点与分割超平面之间的间隔来训练模型。Hinge Loss的目标是使样本的预测边际大于1，同时对误分类样本施加惩罚。

其工作原理：

1.分类间隔最大化：目标是找到一个分割超平面，使得不同类别的样本点尽量远离这个平面，即使得每个样本点$f(x)$与其对应标签y的乘积大于等于1。若$y \cdot (x)$≥1,则表示该样本正确分类并且离决策超平有一定的距离，损失为0；否则损失为 1−$y \cdot f(x)$。

2.惩罚误分类：如果样本被误分类，或者距离分隔超平面太近（即$f(x)$离边界过近），就会产生损失。

hinge loss线性分类模型更适用于高维数据，其在高维空间下通常能找到更好的决策边界，尤其在特征非常多时效果好。

其损失函数代码如下：

```python
# Hinge Loss 损失函数（SVM模型）
def hinge_loss(W, b, X, y, reg=0.1):
    # 计算 SVM 模型的损失（Hinge Loss）
    margin = y * (np.dot(X, W) + b)  # margin 是每个样本点的 margin，y 是标签，W 是权重，b 是偏置项
    # 计算 hinge loss 和正则化项，reg 是正则化强度
    loss = np.mean(np.maximum(0, 1 - margin)) + 0.5 * reg * np.sum(W ** 2)
    return loss
```

Cross-Entropy Loss：

Cross-Entropy Loss 通常用于逻辑回归或神经网络中，目标是使模型输出的概率分布尽可能接近真实标签的概率分布。

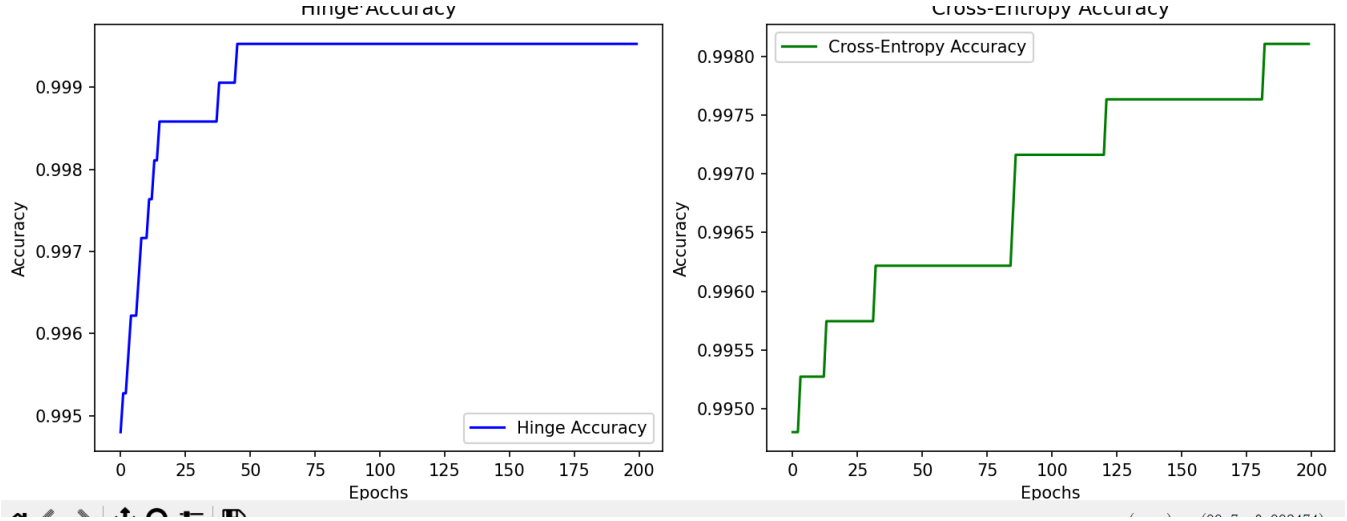
其对损失的定义是：Cross-Entropy Loss损失度量的是预测的概率分布与真实标签分布之间的差异。若模型预测的概率与真实标签相近，损失较小；若预测偏差较大，则损失较大。

该损失函数如下：

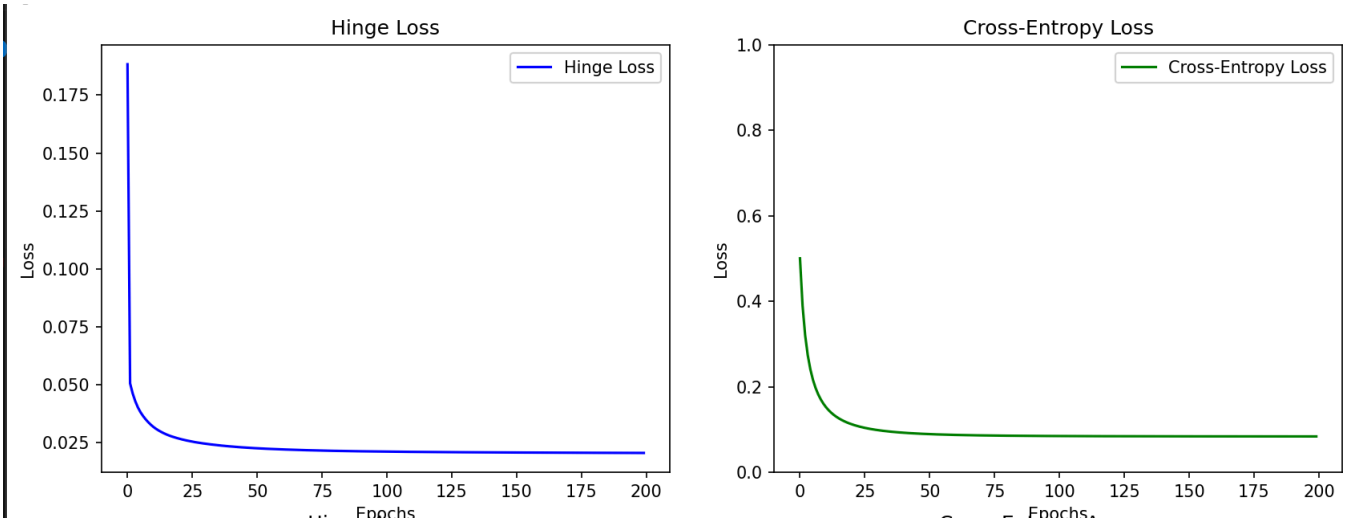
```python
# Cross-Entropy Loss 损失函数（逻辑回归）
def cross_entropy_loss(W, b, X, y, reg=0.1):
    # 计算逻辑回归模型的损失（Cross-Entropy Loss）
    n_samples = X.shape[0]
    linear_output = np.dot(X, W) + b  # 计算每个样本的线性输出
    predictions = 1 / (1 + np.exp(-linear_output))  # 使用 Sigmoid 函数计算预测的概率
    # 计算 cross-entropy 损失和正则化项，reg 是正则化强度
    loss = -np.mean(y * np.log(predictions) + (1 - y) * np.log(1 - predictions)) + 0.5 * reg * np.sum(W ** 2)
    return loss
```

运行结果如下：(学习率为0.01，epoch为200)

hinge loss和cross-entropy loss的准确率图像：



hinge loss和cross-entropy loss的损失曲线：




```
Hinge Loss Model Accuracy: 0.9995 (Training time: 1.7903 seconds)
Cross-Entropy Loss Model Accuracy: 0.9981 (Training time: 1.2246 seconds)

[Done] exited with code=0 in 98.878 seconds
```

可以看出，相比之下hinge loss的训练时间要比cross-entropy loss长，且hinge loss的准确率要比cross-entropy loss要高，且hinge loss的损失与cross-entropy loss的损失均随着时间逐渐收敛，而hinge loss收敛速度更快。

训练过程（包括初始化方法、超参数选择、训练技巧等）

初始化方法： 在实现梯度下降法时，模型参数W和b初始化为零
超参数选择： 1.学习率：在代码中，学习率被设定为 0.01。 若学习率调为0.001，可以看出训练时间变长且准确率下降，hinge loss的与cross-entropy loss的损失收敛速度也没有学习率为0.01时下降得快.

学习率为0.001时：

```
Hinge Loss Model Accuracy: 0.9986 (Training time: 3.2414 seconds)
Cross-Entropy Loss Model Accuracy: 0.9957 (Training time: 1.4795 seconds)

[Done] exited with code=0 in 80.252 seconds
```

2.正则化参数λ:正则化参数控制模型的复杂度，防止过拟合。较大的λ会抑制权重的增大，从而简化模型。故在代码中选择了 0.5 的正则化系数。
训练技巧： 采用了批量梯度下降来更新模型参数，即每次梯度更新都涉及到整个数据集.

实验结果、分析及讨论

使用现成的 SVM 库进行训练时，线性核和高斯核的 SVM 模型分别达到了不同的准确率。

```
[Running] python -u "e:\robotstudy\rstudy\test.py"
线性核函数准确率: 0.9995 (Training time: 0.5697 seconds)
高斯核函数准确率: 0.9972 (Training time: 2.4679 seconds)
```

手动实现的线性SVM（基于hinge loss）和逻辑回归（基于cross-entropy loss）模型运行结果表明，SVM 模型通常在数据集的分类任务中提供较好的准确性，尤其是在使用高斯核时。

```
Hinge Loss Model Accuracy: 0.9995 (Training time: 1.7903 seconds)
Cross-Entropy Loss Model Accuracy: 0.9981 (Training time: 1.2246 seconds)

[Done] exited with code=0 in 98.878 seconds
```

分析：
高斯核模型的准确性通常优于线性核模型，因为高斯核可以处理更复杂的非线性数据集。而线性核的优势在于其计算效率较高，适用于简单的线性数据。但由于本实验使用的数据较简单，故线

性核函数的准确率要高于高斯核。
Hinge loss 和 Cross-Entropy loss：这两种损失函数分别代表了两种不同的分类策略：Hinge loss 适用于硬分类的 SVM 模型，而 Cross-Entropy loss 则适用于概率输出的模型（如逻辑回归）。它们的主要区别在于处理分类问题的方式不同：Hinge loss 强调“决策边界”，而 Cross-Entropy loss 更侧重于概率预测。
从运行结果可以看出，对于本实验的数据，更适合用Hinge loss来分类。