导入所需的库

In [11]:

```
import gzip
import numpy as np

from sklearn.datasets import fetch_openml
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score,recall_score,f1_score
```

1. 读取 MNIST 数据：加载并处理手写数字图像数据及其对应的标签。

特征缩放：对图像数据进行归一化处理，缩放到[0,1]范围。

load_images函数用于读取MNIST数据集中的图片数据并进行预处理。 首先使用gzip库中的gzip.open函数打开指定的文件，并按大端字节序读取前4个字节，将其解析为magic_number，表示该文件的格式标识。接着依次读取文件中图片的数量、每张图片的行数和列数，并将读取到的数据解析为数字格式(uint8)。 接下来，读取剩余的数据，并使用np.frombuffer函数将其转换为numpy数组images。最后，将images数组进行形状变换，将每张图片展平成一个1维数组，并返回像素值范围缩放为[0,1]的images数组，以便于后续的机器学习算法使用。

In [2]:

```
def load_images(filename):
    with gzip.open(filename, 'rb') as f:
        magic_number = int.from_bytes(f.read(4), 'big')
        num_images = int.from_bytes(f.read(4), 'big')
        num_rows = int.from_bytes(f.read(4), 'big')
        num_cols = int.from_bytes(f.read(4), 'big')
        images = np.frombuffer(f.read(), dtype=np.uint8)
        images = images.reshape(num_images, num_rows * num_cols)
        # 像素值范围缩放
        return images/255.0
```

读取标签数据

load_labels函数用于读取MNIST数据集中的标签数据。 首先使用gzip库中的gzip.open函数打开指定的文件，并按大端字节序读取前4个字节，将其解析为magic_number，表示该文件的格式标识。接着读取文件中标签的数量，并将其解析为数字格式(uint8)。最后，读取剩余的数据，并使用np.frombuffer函数将其转换为numpy数组labels，并返回labels数组，其中每个元素表示对应图片的数字标签。

In [3]:

```
def load_labels(filename):
    with gzip.open(filename, 'rb') as f:
        magic_number = int.from_bytes(f.read(4), 'big')
        num_labels = int.from_bytes(f.read(4), 'big')
        labels = np.frombuffer(f.read(), dtype=np.uint8)
        return labels
```

2. 划分训练集和测试集：将图像数据划分为训练集和测试集

In [7]:

```
# 加载训练集数据和标签
X_train = load_images('train-images-idx3-ubyte.gz')
y_train = load_labels('train-labels-idx1-ubyte.gz')
```

In [8]:

```
# 加载测试集数据和标签
X_test = load_images('t10k-images-idx3-ubyte.gz')
y_test = load_labels('t10k-labels-idx1-ubyte.gz')
```

In [16]:

```
print('X_train shape:', X_train.shape)
print('y_train shape:', y_train.shape)
print('X_test shape:', X_test.shape)
print('y_test shape:', y_test.shape)
```

```
X_train shape: (60000, 784)
y_train shape: (60000,)
X_test shape: (10000, 784)
y_test shape: (10000,)
```

3. 构建支持向量机模型：核函数分别选择线性核、多项式核、sigmoid和rbf训练模型。

In [ ]:

```
# 核函数为linear
svm1 = SVC(kernel='linear', C=1.0)
svm1.fit(X_train, y_train)
```

In [ ]:

```
# 核函数为poly
svm2 = SVC(kernel='poly', C=1.0)
svm2.fit(X_train, y_train)
```

In [ ]:

```
# 核函数为sigmoid
svm3 = SVC(kernel='sigmoid', C=1.0)
svm3.fit(X_train, y_train)
```

In [ ]:

```
# 核函数为rbf
svm4 = SVC(kernel='rbf', C=1.0)
svm4.fit(X_train, y_train)
```

4. 模型评估：在测试集上进行预测，计算模型的准确率、召回率、F 1 值以评估模型性能。

In [23]:

```
# 在测试集上进行预测并计算准确率、召回率及F1
y_pred = svm4.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
recall = recall_score(y_test, y_pred,average='macro')
f1 = f1_score(y_test, y_pred,average='macro')
print('Accuracy:', accuracy)
print('Recall:', recall)
print('F1:', f1)
```

```
Accuracy: 0.9792
Recall: 0.9790919842945065
F1: 0.9791298259748042
```

结论：由实验结果可以得出，当选择核函数为rbf时，效果最好，准确率为0.9792，召回率为0.9790919842945065，F1值为0.9791298259748042。