使用支持向量机（SVM）和不同损失函数的线性分类器——大作业1

学号　22336303　姓名　张西艾

实验内容

1.实现一个线性分类器，使用两种不同的损失函数：Hinge Loss （对应SVM的损失函数）；Cross-Entropy Loss（对应于逻辑回归的损失函数）

比较两种损失函数在分类任务中的表现。

2.使用多项式回归实现一个分类器，并比较不同参数量下的结果，从中体会什么是过拟合。

3.支持向量机SVM：选择两种核函数来训练SVM模型：线性核函数；高斯核函数（RBF）

线性分类器

比较两种损失函数（Hinge Loss 和 Cross-Entropy Loss）在线性分类模型中的表现。

hinge loss

代码见 `hinge_loss.py`

结果如下：

```
epoch: 0 SVM with Hinge Loss Accuracy(training): 0.16483516483516483
epoch: 50 SVM with Hinge Loss Accuracy(training): 0.9604395604395605
epoch: 100 SVM with Hinge Loss Accuracy(training): 0.9692307692307692
epoch: 150 SVM with Hinge Loss Accuracy(training): 0.978021978021978
epoch: 200 SVM with Hinge Loss Accuracy(training): 0.9824175824175824
epoch: 250 SVM with Hinge Loss Accuracy(training): 0.9846153846153847
epoch: 300 SVM with Hinge Loss Accuracy(training): 0.9846153846153847
epoch: 350 SVM with Hinge Loss Accuracy(training): 0.9824175824175824
epoch: 400 SVM with Hinge Loss Accuracy(training): 0.9824175824175824
epoch: 450 SVM with Hinge Loss Accuracy(training): 0.9846153846153847
Training set metrics - Accuracy: 0.9846153846153847, Precision: 0.9821428571428571, Recall: 0.9763313609467456, F1 Score: 0.9792284866468843
Testing set metrics - Accuracy: 0.9824561403508771, Precision: 0.9767441860465116, Recall: 0.9767441860465116, F1 Score: 0.9767441860465116
```

cross-entropy loss

代码见 `cross_entropy.py`

结果如下：

```
Epoch 0:Accuracy(training): 0.5077
Epoch 10:Accuracy(training): 0.9429
Epoch 20:Accuracy(training): 0.9538
Epoch 30:Accuracy(training): 0.9626
Epoch 40:Accuracy(training): 0.9648
Epoch 50:Accuracy(training): 0.9648
Epoch 60:Accuracy(training): 0.9670
Epoch 70:Accuracy(training): 0.9714
Epoch 80:Accuracy(training): 0.9736
Epoch 90:Accuracy(training): 0.9758
Epoch 100:Accuracy(training): 0.9758
Epoch 110:Accuracy(training): 0.9758
Epoch 120:Accuracy(training): 0.9758
Epoch 130:Accuracy(training): 0.9780
Epoch 140:Accuracy(training): 0.9780
Epoch 150:Accuracy(training): 0.9802
Epoch 160:Accuracy(training): 0.9802
Epoch 170:Accuracy(training): 0.9802
Epoch 180:Accuracy(training): 0.9824
Epoch 190:Accuracy(training): 0.9824
Epoch 199:Accuracy(training): 0.9824
Training set metrics - Accuracy: 0.9824, Precision: 0.9877, Recall: 0.9641, F1 Score: 0.9758
Testing set metrics - Accuracy: 0.9561, Precision: 0.9348, Recall: 0.9556, F1 Score: 0.9451
```

实验结果分析与讨论

Hinge Loss在训练集和测试集上都显示出较高的准确率、精确率、召回率和F1值，表明其在这项任务中表现更好。 Cross-Entropy Loss在训练集上表现良好，但在测试集上表

现较差，模型可能在训练集上过拟合。

Hinge Loss主要用于支持向量机（SVM）等最大间隔分类器，它对线性可分数据集特别有效。如果数据集确实是线性可分的，Hinge Loss可以更有效地找到决策边界，因为它专注于将不同类别的数据点正确分开，而不是预测概率。Hinge Loss对异常值和噪声不太敏感，因为它只关注那些位于间隔边界附近的数据点（即支持向量）。相比之下，交叉熵损失函数对所有数据点的预测误差都敏感，包括异常值。使用Hinge Loss时，SVM的正则化项直接作用于权重，这有助于防止过拟合。在交叉熵损失函数中，正则化通常需要单独添加，并且可能不如SVM中的正则化那样直接。

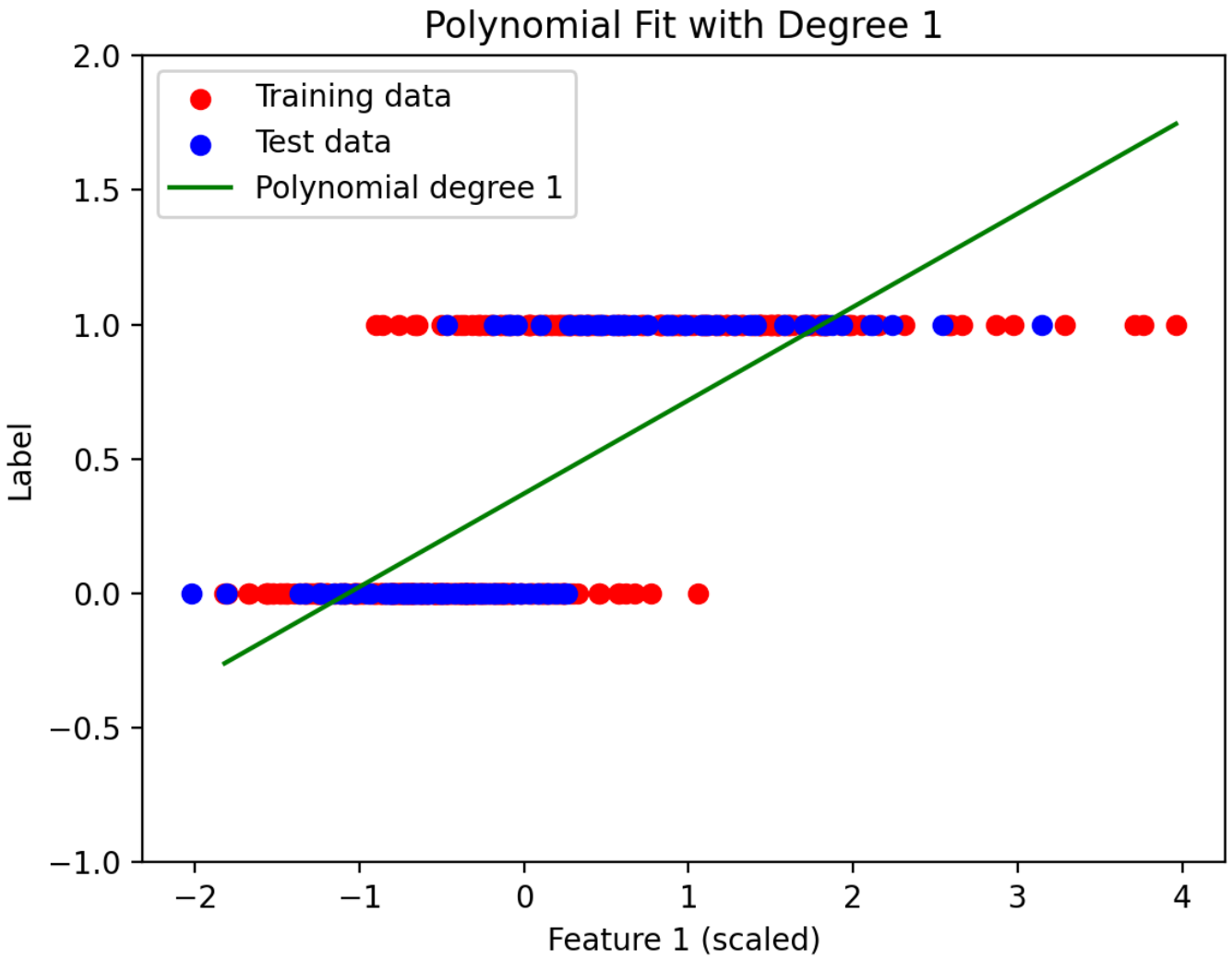
多项式回归实现分类器

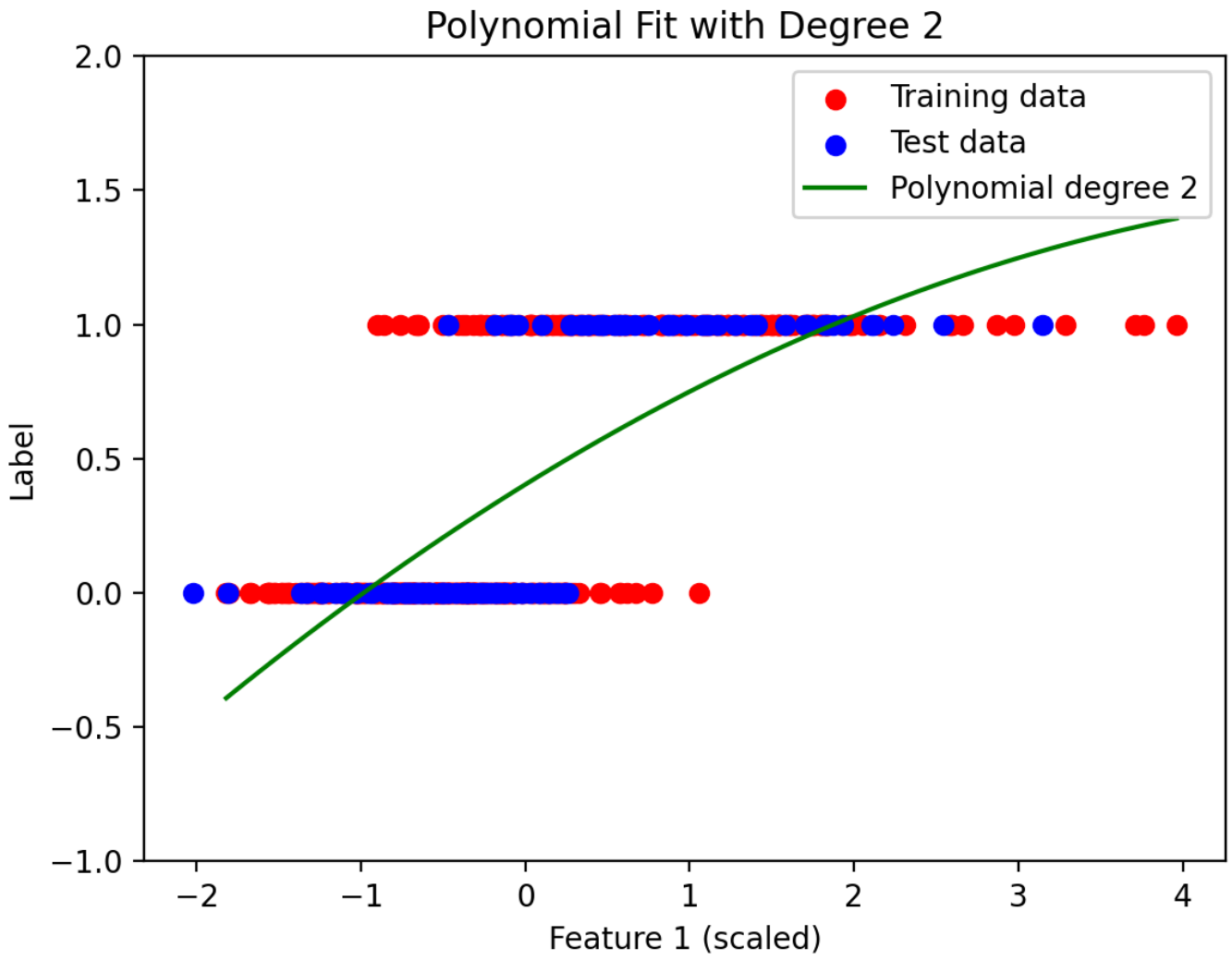
代码见 `polyfit.py`

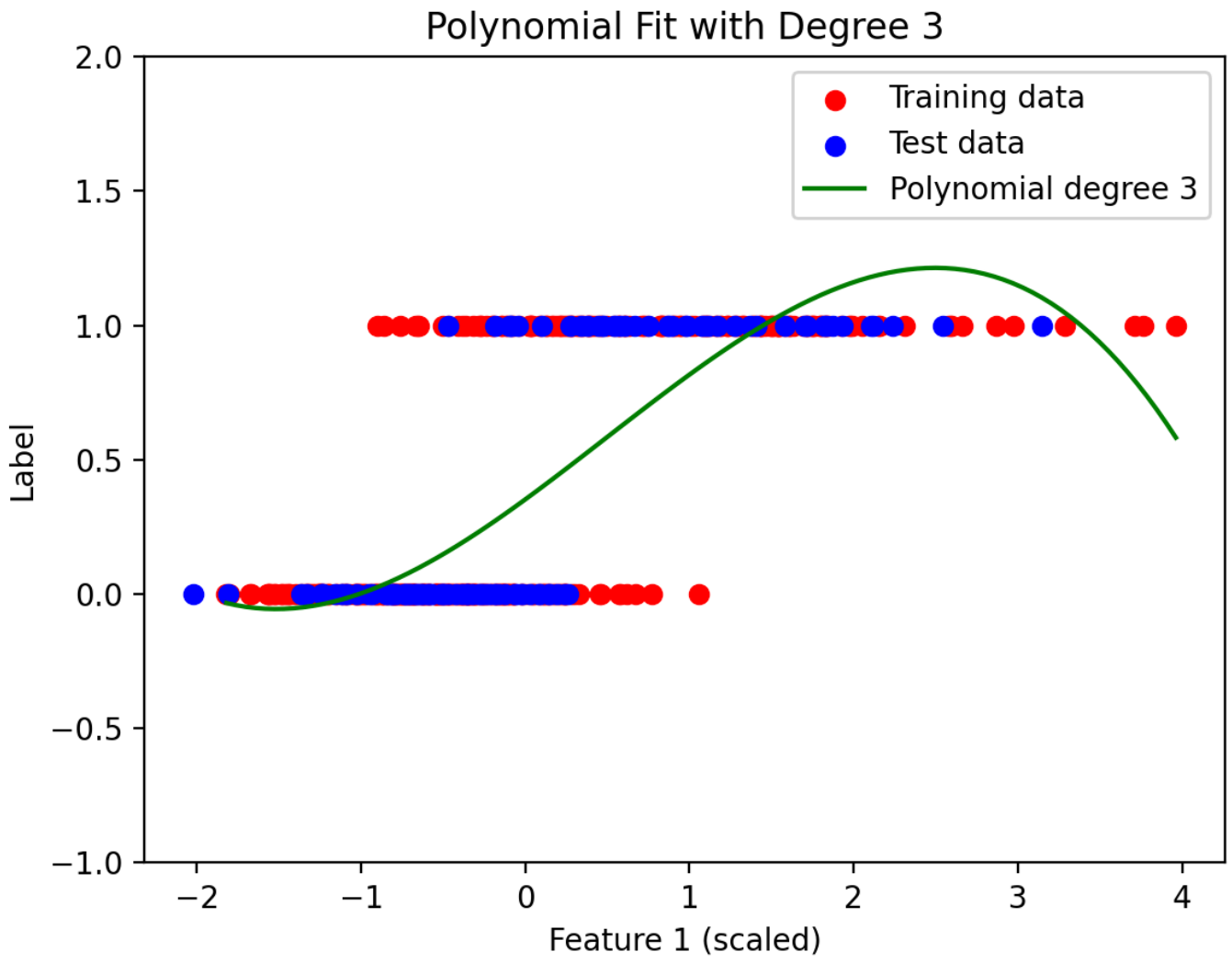
结果如下：

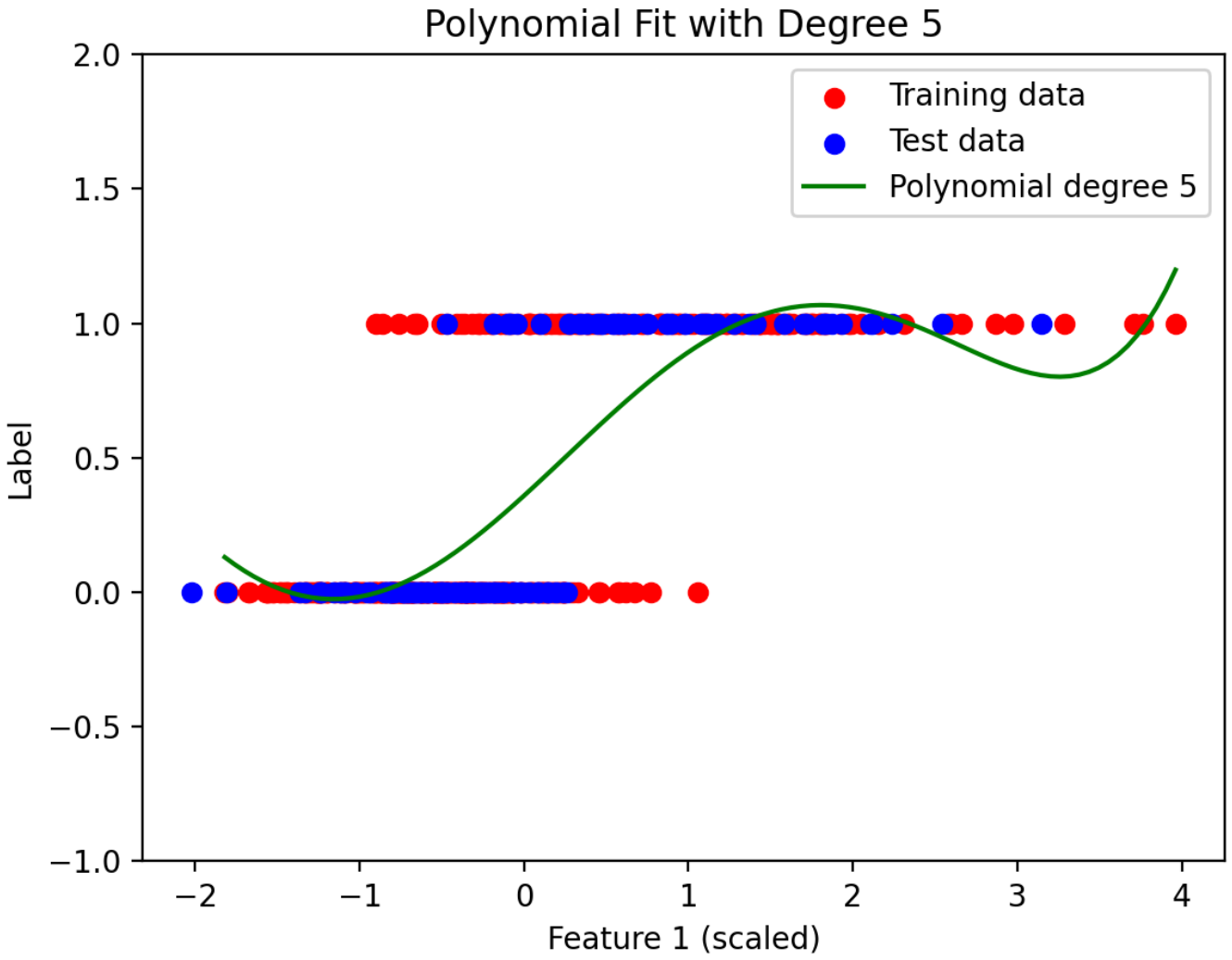
```
Degree 1: Accuracy = 0.9211
Degree 2: Accuracy = 0.9298
Degree 3: Accuracy = 0.9298
Degree 5: Accuracy = 0.9298
Degree 10: Accuracy = 0.9211
Degree 20: Accuracy = 0.9123
Degree 25: Accuracy = 0.9035
```

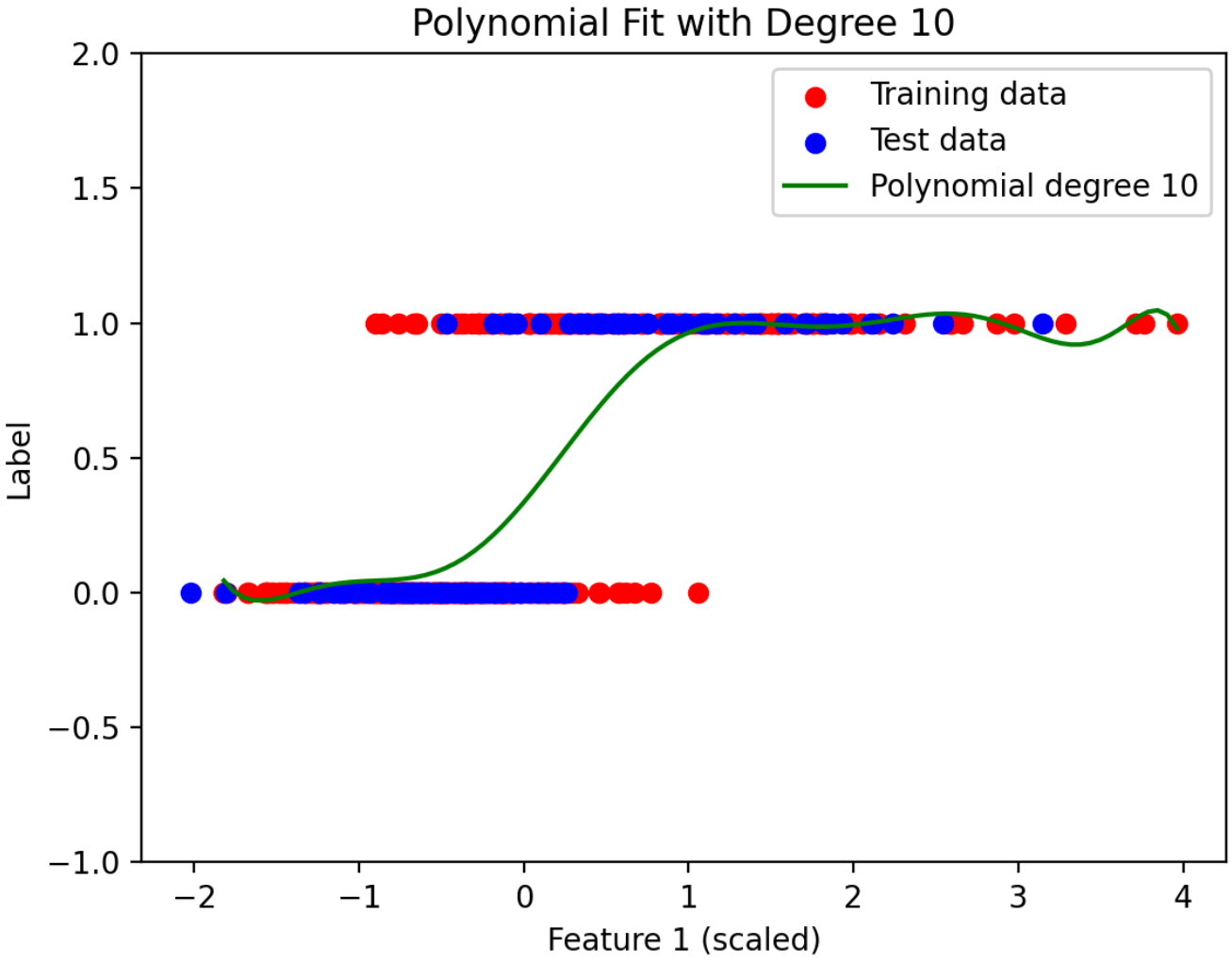
不同参数量下的拟合曲线如图所示：

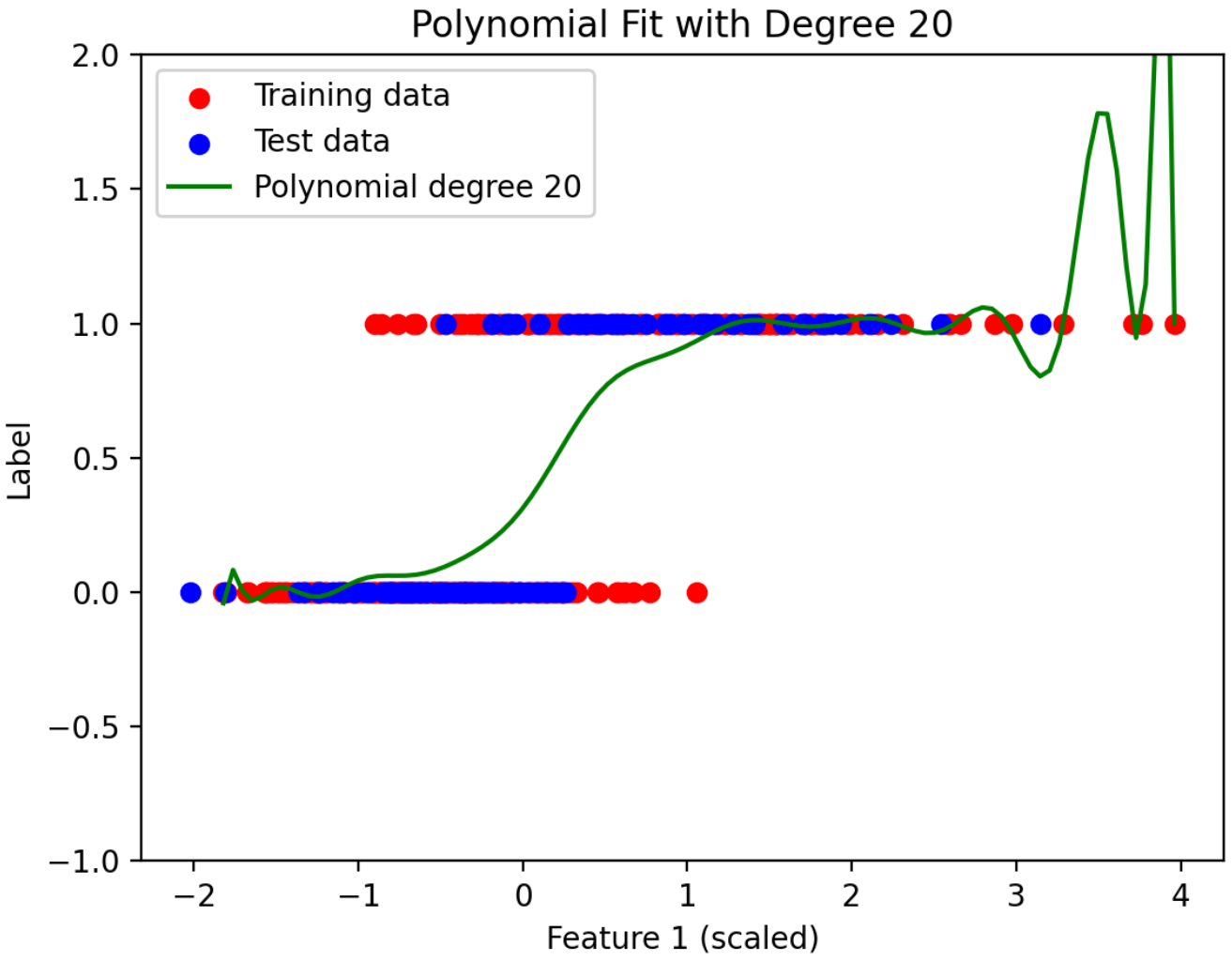

Polynomial Fit with Degree 1
Training data
Test data
Polynomial degree 1
Label
Feature 1 (scaled)
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−2
−1
0
1
2
3
4

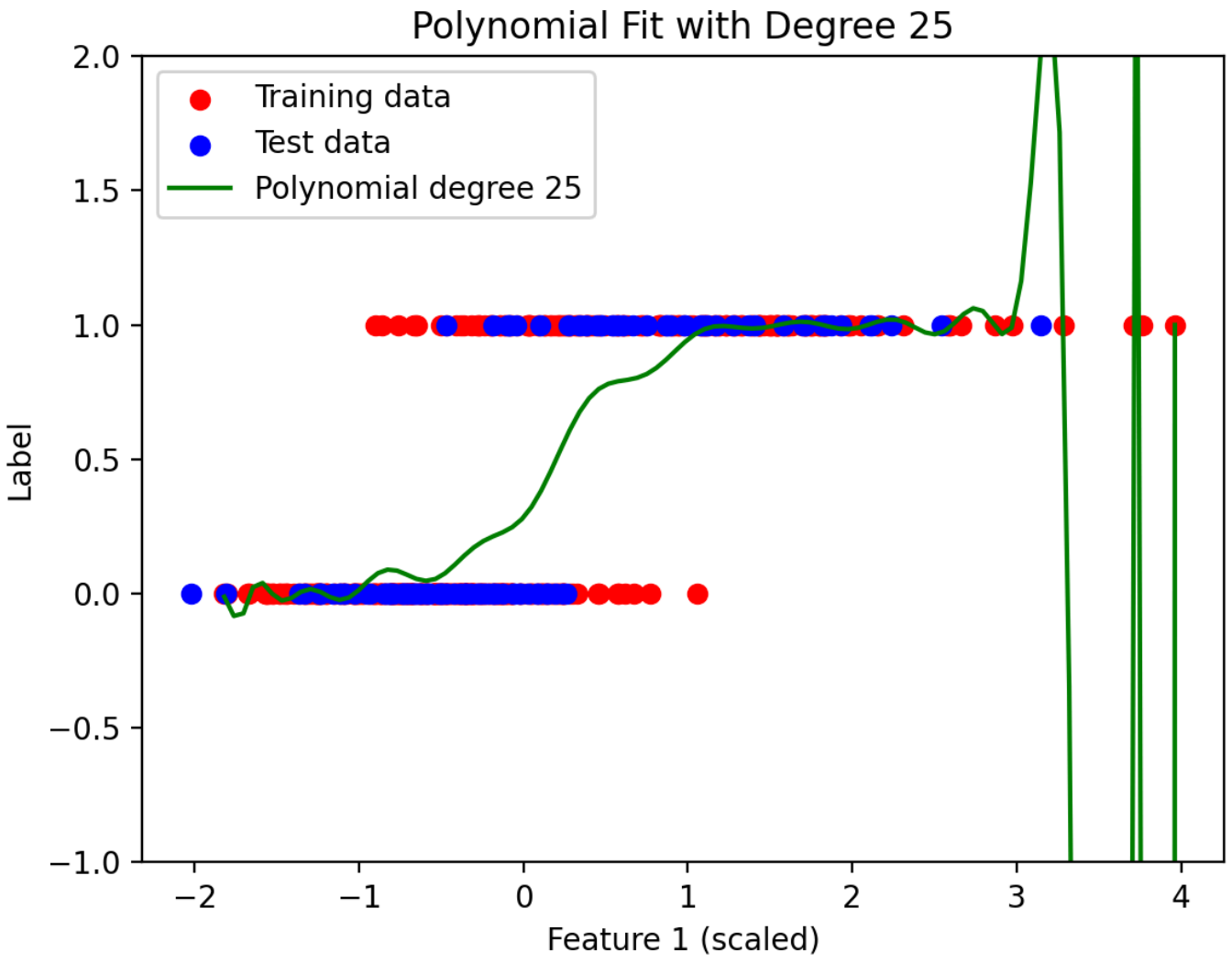

Polynomial Fit with Degree 2
Training data
Test data
Polynomial degree 2
Label
Feature 1 (scaled)
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−2
−1
0
1
2
3
4


Polynomial Fit with Degree 3
Training data
Test data
Polynomial degree 3
Label
Feature 1 (scaled)
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−2
−1
0
1
2
3
4


Polynomial Fit with Degree 5
Training data
Test data
Polynomial degree 5
Label
Feature 1 (scaled)
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−2
−1
0
1
2
3
4


Polynomial Fit with Degree 10
Training data
Test data
Polynomial degree 10
Label
Feature 1 (scaled)
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−2
−1
0
1
2
3
4


Polynomial Fit with Degree 20
Training data
Test data
Polynomial degree 20
Label
Feature 1 (scaled)
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−2
−1
0
1
2
3
4



过拟合现象分析

过拟合是指模型在训练数据上表现得很好，但是在新的、未见过的数据上表现不佳的现象。这通常是因为模型过于复杂，捕捉到了训练数据中的噪声和细节，而没有抓住数据的潜在分布。

1阶多项式（线性模型）：准确率为0.9211。

2阶多项式：准确率提高到0.9298，表明增加模型复杂度可以提高拟合质量。

3阶和5阶多项式：准确率保持在0.9298，与2阶多项式相同，表明进一步提高模型复杂度并没有带来更多的好处。

10阶多项式：准确率下降到0.9211，与1阶多项式相同，表明过高的模型复杂度可能导致过拟合。

20阶和25阶多项式：准确率进一步下降到0.9123和0.9035，这表明模型在训练数据上捕捉到了噪声，而没有很好地泛化到新数据上。

从2阶到5阶多项式，增加模型复杂度并没有提高准确率。从10阶到25阶多项式，准确率开始下降，从拟合曲线上也可以看出模型开始过拟合。20阶和25阶多项式的准确率低于2阶多项式，这表明高阶模型在训练数据上的表现并不比简单模型好。

支持向量机SVM

SVM模型理论

支持向量机（SVM）是一种监督学习算法，主要用于分类问题。它的基本原理是通过找到一个最优的超平面来分隔不同的类别。

支持向量: 支持向量是距离决策边界最近的样本点，它们位于间隔边界上。这些点是最关键的数据点，因为它们决定了超平面的位置和方向。如果数据集中没有这些支持向量，超平面的位置可能会改变。

间隔最大化：SVM的一个核心目标是找到一个超平面，使得不同类别之间的间隔最大化。间隔是指从超平面到最近样本点的距离。最大化间隔的目的是提高分类器的泛化能力，即在未知数据上的表现。一个较大的间隔意味着模型对于训练数据的噪声和扰动更加鲁棒。

核函数：在现实世界中，许多数据集并不是线性可分的。核函数允许SVM在高维空间中寻找决策边界，而无需显式地计算数据点在高维空间中的坐标。核技巧通过将原始特征映射到高维空间，使得非线性问题转化为线性问题。常用的核函数包括：线性核，适用于线性可分的数据。多项式核：适用于需要曲线决策边界的情况。高斯径向基函数（RBF）核：适用于非常复杂的非线性问题。Sigmoid核：类似于神经网络中的激活函数。

训练过程

代码见 `SVM.py`

在 `scikit-learn`中，我们通过创建 `SVC`类的实例来初始化SVM模型。

SVM通过引入惩罚参数C来实现**正则化**，以避免模型过于复杂。`C` 参数是一个惩罚参数，它控制着模型对误分类点的惩罚强度。`C` 的值越大，模型对误分类的惩罚就越大，这可能导致模型在训练集上的拟合更好，但泛化能力变差，从而可能出现过拟合。相反，`C` 的值越小，模型对误分类的惩罚就越小，这可能导致模型在训练集上的拟合不足，但泛化能力更好，从而可能出现欠拟合。 `gamma` 参数定义了单个训练样本的影响范围，即高斯核的宽度。`gamma` 的值越大，高斯核的宽度就越小，支持向量的影响范围就越小，决策边界就越复杂，模型就越倾向于捕捉训练数据中的细微波动，可能导致过拟

合。相反，`gamma` 的值越小，高斯核的宽度就越大，支持向量的影响范围就越广，决策边界就越平滑，模型就越倾向于捕捉训练数据中的一般趋势，这有助于提高模型的泛化能力。 在GridSearchCV类中，交叉验证是内置的，这意味着在寻找最佳超参数的过程中，它会对每一组超参数组合进行交叉验证。在GridSearchCV中，可以通过cv参数来指定交叉验证的折数。在实际训练中，我们设置cv=5，使用**5折交叉验证**来评估每一组超参数的性能。scoring='accuracy'表示使用准确率作为评估标准。

在支持向量机（SVM）的训练过程中，通常使用的是优化算法如**序列最小优化算法SMO**，而不是梯度下降。SVM的优化问题是一个凸二次规划问题，而SMO算法是专门针对这类问题设计的一种有效算法。SMO算法将SVM的复杂优化问题分解为多个二分类问题，然后逐一解决，从而找到原始问题的解。虽然梯度下降是一种广泛使用的优化算法，尤其在深度学习和大规模优化问题中，但它并不适用于SVM的优化问题。SVM的优化目标是找到一个最大化间隔的超平面，而这个目标函数不是光滑的（因为存在间隔最大化的合页损失函数），这使得梯度下降难以直接应用。

结果比较

线性核的结果如下：

线性核最佳参数： {'C': 0.1}

线性核最优模型的训练集分类报告：

	precision	recall	f1-score	support
B	0.98	0.99	0.99	249
M	0.99	0.97	0.98	149
accuracy			0.98	398
macro avg	0.98	0.98	0.98	398
weighted avg	0.98	0.98	0.98	398

线性核最优模型的测试集分类报告：

	precision	recall	f1-score	support
B	0.98	0.99	0.99	108
M	0.98	0.97	0.98	63
accuracy			0.98	171
macro avg	0.98	0.98	0.98	171
weighted avg	0.98	0.98	0.98	171

线性核最优模型的训练集准确率：0.9824120603015075
线性核最优模型的训练集精确率：0.9863013698630136
线性核最优模型的训练集召回率：0.9664429530201343
线性核最优模型的训练集F1值：0.9762711864406781
线性核最优模型的测试集准确率：0.9824561403508771
线性核最优模型的测试集精确率：0.9838709677419355
线性核最优模型的测试集召回率：0.9682539682539683
线性核最优模型的测试集F1值：0.976

高斯核的结果如下：

高斯核最佳参数： {'C': 100, 'gamma': 'scale'}

高斯核最优模型的训练集分类报告：

	precision	recall	f1-score	support
B	1.00	1.00	1.00	249
M	1.00	1.00	1.00	149
accuracy			1.00	398
macro avg	1.00	1.00	1.00	398
weighted avg	1.00	1.00	1.00	398

高斯核最优模型的测试集分类报告：

	precision	recall	f1-score	support
B	0.98	0.94	0.96	108
M	0.91	0.97	0.94	63
accuracy			0.95	171
macro avg	0.95	0.96	0.95	171
weighted avg	0.95	0.95	0.95	171

高斯核最优模型的训练集准确率：1.0
高斯核最优模型的训练集精确率：1.0
高斯核最优模型的训练集召回率：1.0
高斯核最优模型的训练集F1值：1.0
高斯核最优模型的测试集准确率：0.9532163742690059
高斯核最优模型的测试集精确率：0.9104477611940298
高斯核最优模型的测试集召回率：0.9682539682539683
高斯核最优模型的测试集F1值：0.9384615384615386

训练集性能：高斯核在训练集上达到了完美的准确率、精确率、召回率和F1值，表明它在训练数据上完美地拟合了所有数据点。线性核在训练集上的性能也非常出色，但略低于高斯核。

测试集性能：线性核在测试集上的性能非常稳定，准确率、精确率、召回率和F1值都很高，与训练集性能相当接近。高斯核在测试集上的性能显著下降，准确率降低到0.9532，精确率降低到0.9104，尽管召回率仍然很高。

高斯核在训练集上完美拟合，但在测试集上性能下降，这表明可能存在过拟合。过拟合意味着模型在训练数据上表现太好，以至于它学习到了训练数据中的噪声和细节，而不是潜在的数据分布。线性核在训练集和测试集上的性能更加均衡，表明它具有更好的泛化能力。

线性核函数通常具有更好的泛化能力。它不会像高斯核那样对训练数据中的噪声和异常值过于敏感，因此在测试集上的表现更为稳定。高斯核函数通过映射到高维空间来处理非线性问题，这可能导致过拟合，尤其是在样本数量有限的情况下。线性核函数由于其简单性，不太可能在训练数据上过拟合。