湖南大學

HUNAN UNIVERSITY

工程优化

论文题目：	基于粒子群算法优化支持向量机 PSO-SVM 对不同复杂度的数据集进行分类
学生姓名：	田静
学生学号：	S230200183
学院名称：	机械与运载工程学院
课程老师：	王琥

目录

1 序言 3
2 PSO-SVM 算法 4
2.1 支持向量机 4
2.2 粒子群优化算法 6
2.3 PSO-SVM 建模 7
3 基于 PSO-SVM 对不同复杂度的数据集进行分类 8
3.1 基于 PSO-SVM 对湖南大学 2024 年研究生录取数据集分类 9
3.1.1 湖南大学 2024 年研究生录取数据集 9
3.1.2 分类结果 9
3.2 基于 PSO-SVM 对心脏病诊断数据集分类 10
3.2.1 心脏病诊断数据集 10
3.2.2 分类结果 10
4 结论 11
致谢 12
参考文献 12

基于粒子群算法优化支持向量机 PSO-SVM 对不同复杂度的数据集进行分类

摘要：为解决 SVM（支持向量机）性能在很大程度上依赖其参数选择，本文通过在 SVM 中使用 PSO（粒子群优化）算法进行优化。首先，介绍 SVM 以及 PSO 优化算法原理，通过 Python 编写其优化算法；然后，通过 SVM 以及 PSO-SVM 优化算法对比较简单的湖南大学 2024 年研究生录取数据集以及复杂的心脏病诊断数据集进行分类。结果表明，对于简单的分类问题，SVM 可以达到与 PSO-SVM 相同效果。而对于比较复杂的分类问题，SVM-PSO 优化算法的预测精度高达 82.47%，相比 SVM 算法预测精度提高了 36.63%。综上所述，在 SVM 中使用 PSO 进行参数优化是一种有效且实用的方法。它不仅能够自动化地搜索最优参数组合，还能够适应复杂问题、提高模型性能，并具有良好的灵活性和可扩展性。

关键词：SVM；PSO-SVM；湖南大学 2024 年研究生录取数据集；葡萄糖数据集

Optimization of Support Vector Machine PSO-SVM for Classifying Data Sets of Different Complexity Based on Particle Swarm Optimization Algorithm

Abstract：To address the issue that SVM (Support Vector Machine) performance heavily relies on its parameter selection, this paper optimizes SVM using PSO (Particle Swarm Optimization) algorithm. Firstly, introduce the principles of SVM and PSO optimization algorithms, and write their optimization algorithms in Python; Then, SVM and PSO-SVM optimization algorithms were used to classify the relatively simple 2024 graduate admission dataset of Hunan University and the complex heart disease diagnosis dataset. The results indicate that SVM can achieve the same effect as PSO-SVM for simple classification problems. For more complex classification problems, the prediction accuracy of the SVM-PSO optimization algorithm is as high as 82.47%, which is 36.63% higher than that of the SVM algorithm. In summary, using PSO for parameter optimization in SVM is an effective and practical method. It can not only automatically search for the optimal parameter combination, but also adapt to complex problems, improve model performance, and has good flexibility and scalability.

Key words：SVM;PSO-SVM;Hunan University 2024 Graduate Admission Dataset; Glucose dataset

1 序言

在当今大数据时代，随着信息技术的飞速发展，数据量呈现出爆炸式增长。如何从海量数据中提取有用信息，实现高效、准确的分类预测，成为机器学习领域的重要研究课题。支持向量机（Support Vector Machine, SVM）作为一种经典的分类算法，因其在高维空间中的良好分类能力和强大的泛化性能，在众多领域得到了广泛应用[1]。然而，随着数据集规模的增大和复杂度的提升，SVM 的训练时间显著增加，且对参数的敏感度较高，容易陷入局部最优，从而限制了其在实际应用中的性能[2-4]。

为了克服 SVM 的这些局限性，研究者们开始探索将优化算法与 SVM 相结合的方法，以提高其分类性能。粒子群优化算法（Particle Swarm Optimization, PSO）作为一种模拟鸟群觅食行为的进化计算技术，因其简单易实现、收敛速度快、全局搜索能力强等优点，在优化问题中展现出巨大潜力[5]。PSO 通过群体中个体之间的协作和信息共享来寻找最优解，与 SVM 相结合，可以优化 SVM 的参数，提高分类预测的精度和效率[6-12]。

基于上述背景，本文提出了基于粒子群算法优化支持向量机（PSO-SVM）的数据分类方法，并研究了该方法在不同复杂度的数据集上的分类性能。PSO-SVM 通过将 PSO 的全局搜索能力与 SVM 的分类能力相结合，通过迭代优化 SVM 的超参数，如权重矩阵和偏置项，从而找到最优的分类模型[13-17]。本文旨在通过实验验证 PSO-SVM 在不同复杂度的数据集上的分类效果，并探讨其在实际应用中的潜力和优势。

本文首先介绍了 SVM 和 PSO 的基本原理，阐述了 PSO-SVM 算法的实现步骤和核心思想。随后，选取了多个具有不同复杂度的数据集，包括小规模数据集、大规模数据集以及高维特征空间的数据集，进行实验验证。实验过程中，我们比较了传统 SVM 和 PSO-SVM 在分类准确率方面的性能，并通过统计分析和可视化展示结果。

实验结果表明，PSO-SVM 在不同复杂度的数据集上均表现出良好的分类性能，尤其在处理大规模数据集和高维特征空间的数据时，其优势更加明显。PSO-SVM 通过优化 SVM 的参数，可以显著提高对于复杂问题分类的准确率，从而在实际应用中具有更高的效率和实用性。

本文的研究成果不仅丰富了机器学习算法的理论体系，也为实际应用提供了有效的技术支持。在未来的研究中，我们将进一步探索 PSO-SVM 算法的优化方法，提高其在实际应用中的性能和稳定性，并拓展其在更多领域的应用。相信随着机器学习和优化算法的不断发展，PSO-SVM 在数据分类预测等领域的应用前景将会更加广阔。

2 PSO-SVM 算法

2.1 支持向量机

SVM 是一种基于统计学理论的有监督机器学习方法，常用于分类和回归。通过搜索最小结构风险，可以提高 SVM 分类器的泛化能力和最小化经验风险。因此，它在样本量较小的情况下也能展现出良好的分类性能。SVM 的核心思想是找到一个可以分离不同类别数据的超平面，并最大化支持向量到超平面的间距。

假设存在数据集 $D=\left\{\left(x_1,y_1\right),\left(x_2,y_2\right),\cdots,\left(x_m,x_m\right)\right\}$，$y_i\in\left\{-1,+1\right\}$；其中 x_i 是输入数据，是一个 N 维的向量；y_i 是类标签。在样本空间中，超平面的定义如下：

$$\omega^T x+b=0 \tag{1}$$

其中，$\omega=\left(\omega_1;\omega_2;\cdots;\omega_d\right)$ 是确定超平面的法向因子；b 是位移项用于确定超平面与原点间的距离。显然，超平面由 ω 和 b 共同确定。

假设超平面 $\left(\omega,b\right)$ 可以直接将数据分类，则有：

$$\begin{cases}\omega^T x_i+b\geq+1, y_i=+1\\ \omega^T x_i+b\leq-1, y_i=-1\end{cases} \tag{2}$$

能够使式(2)中等号成立的输入数据称作 “支持向量”。 当两个异类支持向量到超平面的距离和最大时，该超平面即是所搜索的目标平面。而这个距离被称为 “最大边距”， 定义如下：

$$\begin{aligned} &\min \frac{1}{2}\|\omega\|^2 \\ &s.t. \quad y_i\left(\omega^T x_i + b\right) \geq 1, i = 1, ..., n \end{aligned} \tag{3}$$

这意味着 SVM 的基本工作原理就是找到能够满足约束条件的ω，b，使得 “最大边距” 最大化。

使用拉格朗日乘子法，以及引入核函数，式(3)的对偶问题表示如下：

$$\begin{aligned} &\max_{\alpha} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j k\left(x_i, x_j\right) \\ &s.t. \quad \sum_{i=1}^{n} \alpha_i y_i = 0, \alpha_i \geq 0, i = 1, ..., n \end{aligned} \tag{4}$$

此时超平面模型表示成：

$$f\left(x\right) = \sum_{i}^{m} \alpha_i y_i k\left(x, x_i\right) + b \tag{5}$$

其中函数$k\left(\cdot,\cdot\right)$表示 “核函数”， 用以实现将样本从原始特征空间映射到更高维的特征空间。核函数的种类如下所示：

- 径向基函数$\left(RBF\right)$或者高斯核：$k\left(x_i, x_j\right) = \exp\left(-\frac{\|x_i - x_j\|}{2\sigma^2}\right)$。
- d 次多项式核：$k\left(x_i, x_j\right) = \left(\left\langle x_i, x_j \right\rangle + C\right)^d$
- Sigmoid 核：$k\left(x_i, x_j\right) = \tanh\left(\beta x_i^T x_j + \theta\right)$。

在现实中，很难找到一个合适的核函数使得训练样本能够在特征空间中线性分离。因此，需要加入松弛变量ξ_i来放松线性 SVM 的约束，其中ξ_i表示第i个训练模式与对应的边距超平面之间的距离，并且它应该最小化。最终 SVM 的优化目标定义如下。

$$\begin{aligned} &\min \frac{1}{2}\|\omega\|^2 + C\sum_{i=1}^{m} \xi_i \\ &s.t. \quad y_i\left(\omega^T k\left(x_i, x_j\right) + b\right) \geq 1 - \xi_i \\ &\xi_i \geq 0, i = 1, ..., m \end{aligned} \tag{6}$$

式中，C 是一个大于 0 的常数，表示为对错误样本的惩罚程度。

2.2 粒子群优化算法

PSO 是粒子群优化算法（Particle Swarm Optimization）的英文缩写，是一种基于种群的随机优化技术，由 Eberhart 和 Kennedy 于 1995 年提出。粒子群算法是模仿昆虫、兽群、鸟群和鱼群等的群集行为，这些群体按照一种合作的方法寻找食物，群体中的每个成员通过学习它自身的经验和其他成员的经验来不断的改变其搜索方式[18-19]。PSO 由于操作简单、收敛速度快、并没有许多参数的调节，因此，被广泛应用于函数优化、神经网络训练、模糊系统控制以及其他遗传算法的应用领域。

粒子群优化算法的基本思想是通过群体中个体之间的协作和信息共享来寻找最优解。用一种粒子模拟种群个体，每个粒子可视为 N 维搜索空间中的一个搜索个体，粒子的当前位置即为对应优化问题的一个候选解，粒子的飞行过程即为该个体的搜索过程。粒子的飞行速度可根据粒子历史最优位置和种群历史最优位置进行动态调整。粒子仅有两个属性：速度和位置，速度代表移动的快慢，位置代表移动的方向。每个粒子单独搜寻的最优解叫做个体极值，粒子群中的最优个体极值作为当前全局的最优解。不断迭代，更新速度和位置。最终得到满足条件的最优解。流程如下：

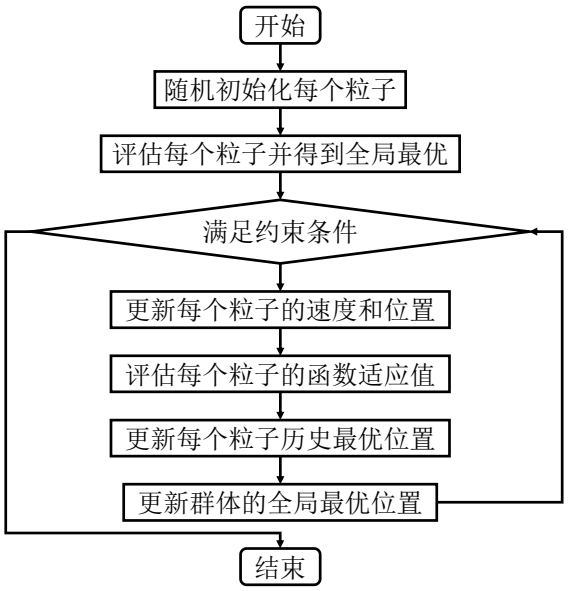


图 1 PSO 算法流程图

PSO 算法的步骤为:

（1）初始化所有粒子，即给它们的速度和位置赋值，并将个体的历史最优 pBest 设为当前位置，群体中的最优个体作为当前的 gBest。

（2）在每一代的进化中，计算各个粒子的适应度函数值。

（3）如果当前适应度函数值优于历史最优值，则更新 pBest。

（4）如果当前适应度函数值优于全局历史最优值，则更新 gBest。

（5）对每个粒子 i 的第 d 维的速度和位置分别按照如下公式进行更新。其中 w 为惯量权重，一般初始化为 0.9，随着进化过程线性递减到 0.4；c1 和 c2 是加速系数，传统上取固定值 2.0；rand1 和 rand2 是两个[0，1]区间上的随机数。

2.3 PSO-SVM 建模

在使用 SVM 进行分类时，需要优化 SVM 模型中惩罚系数 C 以及所选定的核函数的参数。将采用 PSO 算法优化 SVM 的超参数，使 SVM 模型在预测上达到理想的分类效果[20-22]。

（1）对于 RBF 核函数需要优化两个参数，分别是 C、gamma；对于多项式核函数需要优化 4 个参数，分别是 C、gamma、r 和 d；对于 sigmoid 核函数，需要优化三个参数，分别是 C、gamma、r。随机初始化粒子的初速度和初始位置。

（2）计算适应度值。在实验中，将训练好的模型在测试集的分类准确率作为ＰＳＯ的适应度值，并将每个个体的初始适应度值作为个体的最优适应值，将种群初始适应值中最优的适应度值作为全局最优适应值。

（3）在给定的约束内，按照 PSO 算法的步骤(5)更新粒子的速度和位置。

（4）根据更新的位置计算适应度值。

（5） 更新个体最优和全局最优。如果当前个体适应度值优于之前个体最优，则将当前适应度值设为最优个体适应值，对应的位置设为个体最优位置。如果当前粒子适应度值要优于全局最优适应值，则将当前粒子适应度值设为全局最优适应度值，对应的位置设为全局最优位置。

（6）终止判断。如果满足终止条件，算法停止并返回最优的参数和准确率。否则重复过程（3）到（5）。

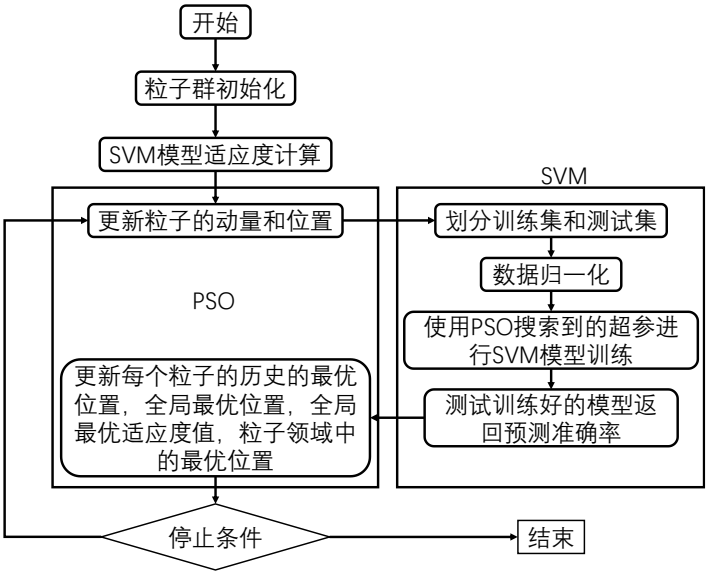


图 2 PSO-SVM 算法流程图

3　基于 PSO-SVM 对不同复杂度的数据集进行分类

上面我们对粒子群算法 PSO-SVM 算法进行了原理介绍，接下来，我们将对湖南大学 2024 年研究生录取数据集分以及克利夫兰数据库中的心脏病数据集进行分类，采用的分类算法为支持向量机（Support Vector Machine,SVM）。通过 PSO 优化 SVM 的重要超参数 c 和 g，寻找使模型效果最好的超参数 c 和 g。从随机解出发，通过迭代寻找最优解，通过适应度来评价解的质量。PSO 初始化为一群随机粒子，然后通过迭代找到最优解。所有的粒子具有位置(particle_position_vector)和速度(velocity_vector)两个属性。在每一次迭代中，粒子通过粒子本身所找到的最优解 pbest 和整个种群目前找到的最优解全局极值 gbest 来更新。接下来，我们通过代码，查看整个过程。

第一步：导入相应的包以及被测数据集合。

```
import numpy as np
import random
from sklearn.svm import SVC
from sklearn.metrics import confusion_matrix
from utils import plot
from utils import data_handle_v1, data_handle_v2
from config import args, kernel, data_src, data_path
from sklearn.model_selection import cross_val_score
import matplotlib.pyplot as plt
```

第二步，初始化模型参数。

```
# 数据源
data_src = '1'
if data_src == '1':
    data_path = 'data/Hunan_University.dat'
elif data_src == '2':
    data_path = 'data/heart_disease.dat'

# 粒子群算法参数配置
class args:
    W = 0.5 # 惯性权重
    c1 = 0.2 # 局部学习因子
    c2 = 0.5 # 全局学习因子
    n_iterations = 10 # 迭代步数
    n_particles = 100 # 粒子数

# SVM配置
kernel = 'rbf' # ["linear","poly","rbf","sigmoid"]
```

第三步，设置适应度值，输出分类精度，并返回混淆矩阵错误结果分类情况。

```
def fitness_function(position,data):
    X_train, X_test, y_train, y_test = data
    svclassifier = SVC(kernel=kernel, gamma = position[0], C = position[1] )
    svclassifier.fit(X_train, y_train)
    y_train_pred = svclassifier.predict(X_train)
    y_test_pred = svclassifier.predict(X_test)
    return confusion_matrix(y_train,y_train_pred)[0][1] + confusion_matrix(y_train,y_train_pred)[1][0],
           confusion_matrix(y_test,y_test_pred)[0][1] + confusion_matrix(y_test,y_test_pred)[1][0]
```

第四步，初始化粒子位置(particle_position_vector)和粒子速度（velocity_vector)。

```
# 初始化参数
particle_position_vector = np.array([np.array([random.random() * 10, random.random() * 10]) for _ in range(args.n_particles)])
pbest_position = particle_position_vector
pbest_fitness_value = np.array([float('inf') for _ in range(args.n_particles)])
gbest_fitness_value = np.array([float('inf'), float('inf')])
gbest_position = np.array([float('inf'), float('inf')])
velocity_vector = ([np.array([0, 0]) for _ in range(args.n_particles)])
```

第五步，开始迭代。

```
iteration = 0
while iteration < args.n_iterations:
    plot(particle_position_vector,iteration)
    for i in range(args.n_particles):
        fitness_cadidate = fitness_function(particle_position_vector[i], data)
        print("error of particle-", i, "is (training, test)", fitness_cadidate, " At (gamma, c): ",
              particle_position_vector[i])

        if (pbest_fitness_value[i] > fitness_cadidate[1]):
            pbest_fitness_value[i] = fitness_cadidate[1]
            pbest_position[i] = particle_position_vector[i]

        if (gbest_fitness_value[1] > fitness_cadidate[1]):
            gbest_fitness_value = fitness_cadidate
            gbest_position = particle_position_vector[i]
        elif (gbest_fitness_value[1] == fitness_cadidate[1] and gbest_fitness_value[0] > fitness_cadidate[0]):
            gbest_fitness_value = fitness_cadidate
            gbest_position = particle_position_vector[i]
    iteration = iteration + 1
    for i in range(args.n_particles):
        new_velocity = (args.W * velocity_vector[i]) + (args.c1 * random.random()) * (
                    pbest_position[i] - particle_position_vector[i]) + (args.c2 * random.random()) * (
                           gbest_position - particle_position_vector[i])
        new_position = new_velocity + particle_position_vector[i]
        particle_position_vector[i] = new_position
```

第六步，输出最终的分类结果。

```
print("error of particle-", i, "is (training, test)", fitness_cadidate, " At (gamma, c): ",
      particle_position_vector[i])
```

第七步，我们将 PSO 得到的最优参数组合带入 SVM，进行验证。

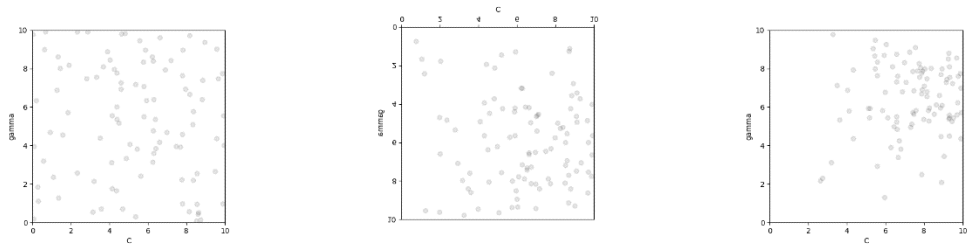
```
svclassifier_ = SVC(kernel='rbf',gamma=gbest_position[0],C=gbest_position[1])
X_train, X_test, y_train, y_test = data
svclassifier_.fit(X_train,y_train)
score = cross_val_score(svclassifier_,X_train,y_train,cv=10).mean()
print("验证后的结果为：",score)
```

3.1 基于 PSO-SVM 对湖南大学 2024 年研究生录取数据集分类

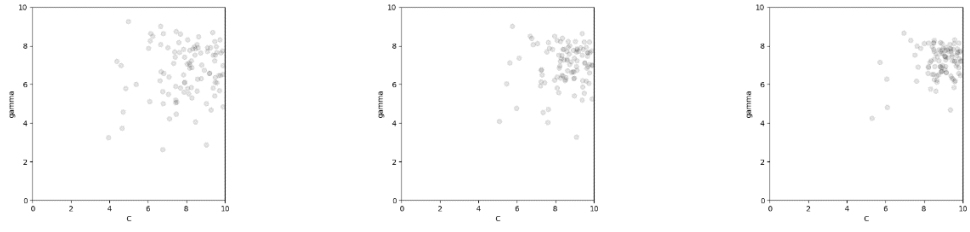
3.1.1 湖南大学 2024 年研究生录取数据集

本文数据来源于湖南大学机械与运载工程学院官网，记录的是湖南大学 2024 年研究生初试各学科和复试面试成绩。共有 301 个样本，选择了报考专业(x1) ，政治成绩(x2) ，外语成绩(x3) ，数学成绩(x4) ，专业课成绩(x5) ，复试面试成绩(x6) 6 个指标作为影响研究生是否录取的体系指标。

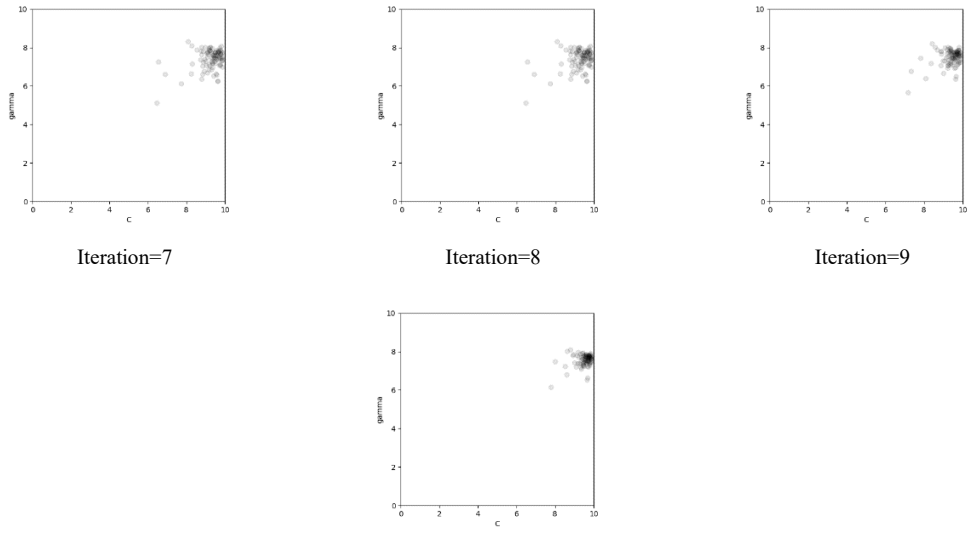
3.1.2 分类结果



Iteration=1　　Iteration=2　　Iteration=3



Iteration=4　　Iteration=5　　Iteration=6



Iteration=7　　Iteration=8　　Iteration=9

Iteration=10

图 3 基于 PSO-SVM 对参数 C、gamma 优化(湖南大学研究生录取数据集)

表 1 PSO-SVM 和 SVM 分类准确率对比表(湖南大学研究生录取数据集)

核函数	PSO-SVM分类准确率	SVM分类准确率
RBF	0.9249	0.9249

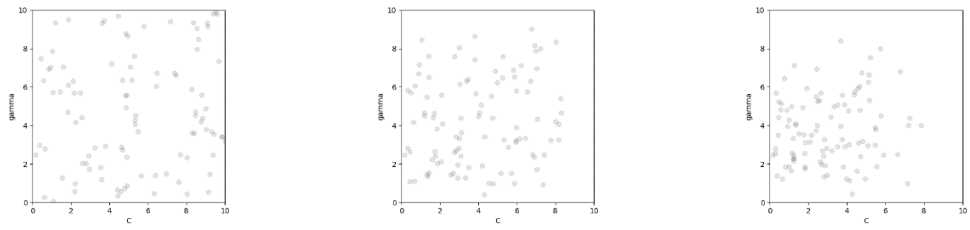
通过 SVM 以及 PSO-SVM 优化算法对比较简单的湖南大学 2024 年研究生录取数据集进行分类。结果表明，对于简单的分类问题，SVM 可以达到与 PSO-SVM 相同效果。

3.2 基于 PSO-SVM 对心脏病诊断数据集分类

3.2.1 心脏病诊断数据集

本文数据来源于克利夫兰数据库，记录的是影响心脏病诊断的 13 种因素。共有 270 个样本，选择了年龄(x1) ，性别(x2) ，胸痛类型 (x3) ，患者入院时的静息血压(x4) ，血清胆固醇水平(x5) ，空腹血糖(x6) ，静息心电图结果 (x7) ，达到的最大心率 (x8) ，运动引起的心绞痛(x9) ，运动相对于休息引起的 ST 抑制(x10) ，最高运动 ST 段的斜率 (x11) ，荧光显色的主要血管数目 (x12) ，一种称为地中海贫血的血液疾病 (x13) 等 13 个指标作为影响心脏病诊断的体系指标。

3.2.2 分类结果



Iteration=1　　Iteration=2　　Iteration=3

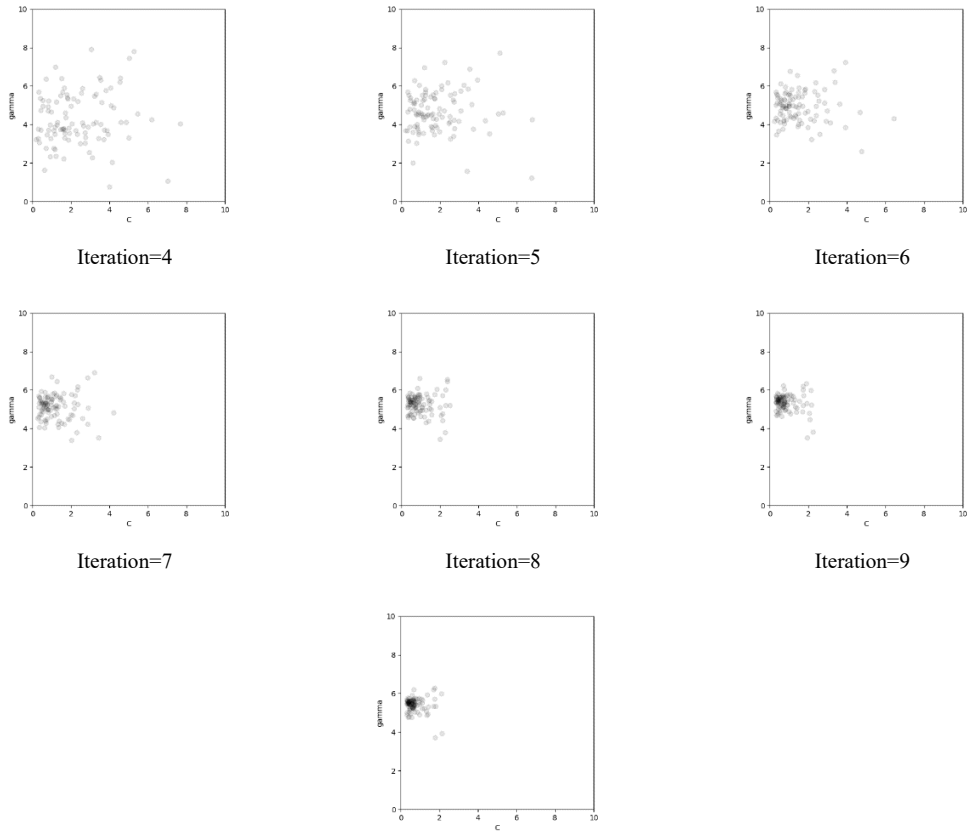


图 4 基于 PSO-SVM 对参数 C、gamma 优化（心脏病诊断数据集）

表 2 PSO-SVM 和 SVM 分类准确率对比表(心脏病诊断数据集)

核函数	PSO-SVM分类准确率	SVM分类准确率
RBF	0.8247	0.4584

通过 SVM 以及 PSO-SVM 优化算法对复杂的心脏病诊断数据集进行分类。结果表明，对于比较复杂的分类问题，SVM-PSO 优化算法的预测精度高达 82.47%，相比 SVM 算法预测精度提高了 36.63%。

4 结论

本文研究了基于 PSO-SVM 对不同复杂度的数据集进行分类，文中给出了模型的详细设计步骤，并给出了 Python 代码设计。最后的实验表明，将 PSO 算法引入到 SVM 模型参数的选取中，并将其应用到复杂问题的分类是可行的，对提高 SVM 模型的分类准确率是有效的，此模型也可以推广到其他分类预测中。

致谢

我由衷地感谢王琥老师对于工程优化课程的精彩讲解。王老师以其深厚的学术功底、丰富的实践经验以及独特的教学风格，为我们开启了一扇通往工程优化领域知识宝库的大门。在他的引领下，我不仅系统地学习了工程优化的基本原理与方法，还深入探索了多个前沿领域，其中支持向量机（SVM）作为一种强大的机器学习算法，更是让我受益匪浅。

通过王老师的悉心讲解和案例分析，我初步掌握了 SVM 的基本概念、核心原理以及它在分类与回归问题中的应用。我了解到，SVM 通过寻找一个最优的超平面来最大化不同类别数据之间的间隔，从而实现高效且准确的分类。这一独特的视角让我对机器学习的本质有了更深刻的理解，也让我认识到 SVM 在解决复杂工程优化问题中的巨大潜力。

总之，王琥老师的工程优化课程不仅为我提供了宝贵的知识财富，更在思维方式、学习态度等方面给予了我深远的影响。对于支持向量机（SVM）的初步认识与了解，只是我学习旅程中的一个起点，我相信在未来的日子里，我会继续深入学习，不断探索，将所学知识应用于更广泛的领域，为实现个人价值和社会贡献而不懈努力。再次感谢王琥老师的辛勤付出与无私奉献！

参考文献

[1]曹嘉嘉；严圆；陈益；尹玲；张斐.基于 PSO 优化的 SVM 在心脏病分类上的应用[J].东莞理工学院学报,2022,第 29 卷(3): 50-56.

[2]陈伟伟，高润霖，刘力生，朱曼璐，王文，王拥军，吴兆苏，李惠君，顾东风，杨跃进，郑哲，蒋立新，胡盛寿.《中国心血管病报告 2017》概要[J].中国循环杂志,2018,(1): 1-8.

[3]李学永.中国成人的心血管健康状况[J].中国循证心血管医学杂志,2015,7(03):306.

[4]王成武，郭志恒，晏峻峰.改进的支持向量机在心脏病预测中的研究[J].计算机技术与发展,2022,第 32 卷(3): 175-179.

[5]辛瑞昊，董哲原，苗冯博，王甜甜，李英瑞，冯欣.基于机器学习的心脏病预测模型研究[J].吉林化工学院学报,2022,第 39 卷(9): 27-32.

[6]王如锴，练继建，苑希民，田福昌，陈隆吉，马文豪.基于 PSO-SVM 的鹤盛溪流域山洪风险评价[J].水资源保护,2024,第 40 卷(2): 46-54.

[7]张伏，张方圆，崔夏华，王新月，曹炜桦，张亚坤，付三玲.高光谱成像结合 PSO-SVM 的银杏果种类鉴别[J].光谱学与光谱分析,2024,第 44 卷(3): 859-864.

[8]李学永.中国成人的心血管健康状况[J].中国循证心血管医学杂志,2015,7(03):306.

[9]杜一平.基于贝叶斯优化的心脏病诊断模型[J].吕梁学院学报,2020,10(02):31-33.

[10]蒋美艳,张辉.机器学习算法对心脏病预测效能的研究[J].中国医学物理学杂志,2024,41(07):905-909.

[11]曹嘉嘉,严圆,陈益,等.基于 PSO 优化的 SVM 在心脏病分类上的应用[J].东莞理工学院学报,2022,29(03):50-56.

[12]王成武,郭志恒,晏峻峰.改进的支持向量机在心脏病预测中的研究[J].计算机技术与发展,2022,32(03):175-179.

[13]吴冠雄.基于 PSO-SVM 的大跨 PC 连续刚构桥施工期可靠度分析[J].湖南交通科技,2024,50(02):132-136.

[14]曾庆栋,陈光辉,李文鑫,等.基于粒子群-支持向量机算法的激光诱导击穿光谱钢铁快速检测与分类[J].光谱学与光谱分析,2024,44(06):1559-1565.

[15]王伟峰,杨博,甘梅,等.基于 PSO-SVM 的矿井输送带火灾危险程度预测模型研究[J].煤炭技术,2024,43(05):212-216.

[16]朱宗玖,王宁.基于 PSO-SVM 的Φ-OTDR 系统模式识别研究[J].科学技术与工程,2024,24(12):5023-5029.

[17]刘惊.基于改进粒子群算法和支持向量结合的辅助发声训练系统研究[J].自动化与仪器仪表,2024,(04):176-184.

[18]许煜辰,王友仁,常烁.基于优化 SVM 的 BUCK 电路故障诊断方法[J].机械制造与自动化,2024,53(02):220-273.

[19]赵国彦,邹景煜,王猛.基于混沌粒子群改进支持向量机对露天矿边坡稳定性的分类预测[J].矿冶工程,2024,44(02):8-12.

[20]朱嘉骏,李英,唐志勇.基于 PSO-SVM 的带钢表面缺陷检测[J].长春理工大学学报(自然科学版),2024,47(01):42-51.

[21]刘庆.基于 IPSO-SVM 的双塔斜拉桥线形预测研究[J].西部交通科技,2024,(01):107-110.

[22]栾阔.基于 PSO-SVM 的机械零件表面缺陷智能分类应用[J].西安文理学院学报(自然科学版),2024,27(01):14-39.