### 12. SVM + 粒子群优化案例：医学影像病灶识别

**问题背景**：某医院需用 SVM 识别肺部 CT 影像中的结节（良性 / 恶性），影像特征含 7 项（如结节大小、密度、边缘光滑度等），SVM 的核参数（\gamma）和惩罚系数（C）对识别精度影响极大，人工调参耗时且效果差。

**数据**：

* 1000 份 CT 影像的 7 项特征及诊断结果（1 = 恶性，0 = 良性）。

**要求**：用粒子群优化搜索最优C和\gamma，训练 SVM 模型，对比默认参数与优化后参数的识别准确率（尤其是恶性结节的检出率），分析参数对结果的影响。

### 12. SVM + 粒子群优化代码：医学影像病灶识别

|  |
| --- |
| import numpy as np  import pandas as pd  import matplotlib.pyplot as plt  from sklearn.svm import SVC  from sklearn.model\_selection import train\_test\_split, cross\_val\_score  from sklearn.preprocessing import StandardScaler  from sklearn.metrics import classification\_report, confusion\_matrix  # 1. 数据准备（1000份CT影像特征）  np.random.seed(42)  n\_samples = 1000  # 7项特征：结节大小、密度、边缘光滑度、位置深度、增强程度、形状不规则度、钙化程度  X = np.random.randn(n\_samples, 7)  # 模拟特征与标签的关系（恶性结节=1，良性=0）  weights = np.array([0.8, 1.2, -0.9, 0.5, 1.0, 1.5, -0.7])  logits = np.dot(X, weights) + np.random.randn(n\_samples) \* 0.5  y = (logits > 0).astype(int) # 恶性结节占比约50%  features = ['结节大小', '密度', '边缘光滑度', '位置深度', '增强程度', '形状不规则度', '钙化程度']  # 2. 数据划分与标准化  X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=42)  scaler = StandardScaler()  X\_train\_scaled = scaler.fit\_transform(X\_train)  X\_test\_scaled = scaler.transform(X\_test)  # 3. 粒子群优化SVM参数（C和gamma）  def pso\_optimize\_svm(X, y, pop\_size=20, iterations=50, c1=2, c2=2, w=0.5):  # 参数范围：C(1e-3~1e3)，gamma(1e-3~1e3)（取对数空间）  def to\_params(position):  c = 10 \*\* position[0]  gamma = 10 \*\* position[1]  return c, gamma    # 适应度函数（5折交叉验证准确率）  def fitness(position):  c, gamma = to\_params(position)  svm = SVC(C=c, gamma=gamma, kernel='rbf', class\_weight='balanced', random\_state=42)  scores = cross\_val\_score(svm, X, y, cv=5, scoring='accuracy')  return np.mean(scores)    # 初始化粒子  dim = 2 # 优化参数维度：C和gamma  positions = np.random.uniform(-3, 3, (pop\_size, dim)) # 对数空间  velocities = np.random.uniform(-0.1, 0.1, (pop\_size, dim))  pbest\_pos = positions.copy()  pbest\_val = np.array([fitness(pos) for pos in positions])  gbest\_idx = np.argmax(pbest\_val)  gbest\_pos = pbest\_pos[gbest\_idx]  gbest\_val = pbest\_val[gbest\_idx]    # 迭代优化  history = [gbest\_val]  for \_ in range(iterations):  for i in range(pop\_size):  # 更新速度和位置  r1, r2 = np.random.rand(2)  velocities[i] = w \* velocities[i] + c1\*r1\*(pbest\_pos[i] - positions[i]) + c2\*r2\*(gbest\_pos - positions[i])  positions[i] += velocities[i]  # 边界限制  positions[i] = np.clip(positions[i], -3, 3)    # 更新个体最优  current\_val = fitness(positions[i])  if current\_val > pbest\_val[i]:  pbest\_val[i] = current\_val  pbest\_pos[i] = positions[i].copy()    # 更新全局最优  current\_gbest\_idx = np.argmax(pbest\_val)  if pbest\_val[current\_gbest\_idx] > gbest\_val:  gbest\_val = pbest\_val[current\_gbest\_idx]  gbest\_pos = pbest\_pos[current\_gbest\_idx].copy()  history.append(gbest\_val)    return to\_params(gbest\_pos), history  # 运行PSO优化  best\_params, pso\_history = pso\_optimize\_svm(X\_train\_scaled, y\_train)  print(f"优化后的SVM参数：C={best\_params[0]:.4f}, gamma={best\_params[1]:.4f}")  # 4. 优化后的SVM模型评估  svm\_opt = SVC(C=best\_params[0], gamma=best\_params[1], kernel='rbf', class\_weight='balanced', random\_state=42)  svm\_opt.fit(X\_train\_scaled, y\_train)  y\_pred = svm\_opt.predict(X\_test\_scaled)  # 对比默认参数模型  svm\_default = SVC(kernel='rbf', class\_weight='balanced', random\_state=42)  svm\_default.fit(X\_train\_scaled, y\_train)  y\_pred\_default = svm\_default.predict(X\_test\_scaled)  # 5. 结果展示  print("\n优化后模型分类报告：")  print(classification\_report(y\_test, y\_pred, target\_names=['良性', '恶性']))  print("\n默认参数模型分类报告：")  print(classification\_report(y\_test, y\_pred\_default, target\_names=['良性', '恶性']))  # 可视化优化过程  </doubaocanvas> |