**import** numpy **as** np

**import** matplotlib.pyplot **as** plt

**class** PSO(object):

**def** \_\_init\_\_(self, population\_size, max\_steps):

self.w = 0.6 *# 惯性权重*

self.c1 = self.c2 = 2

self.population\_size = population\_size *# 粒子群数量*

self.dim = 2 *# 搜索空间的维度*

self.max\_steps = max\_steps *# 迭代次数*

self.x\_bound = [-10, 10] *# 解空间范围*

self.x = np.random.uniform(self.x\_bound[0], self.x\_bound[1],

(self.population\_size, self.dim)) *# 初始化粒子群位置*

self.v = np.random.rand(self.population\_size, self.dim) *# 初始化粒子群速度*

fitness = self.calculate\_fitness(self.x)

self.p = self.x *# 个体的最佳位置*

self.pg = self.x[np.argmin(fitness)] *# 全局最佳位置*

self.individual\_best\_fitness = fitness *# 个体的最优适应度*

self.global\_best\_fitness = np.max(fitness) *# 全局最佳适应度*

**def** calculate\_fitness(self, x):

**return** np.sum(np.square(x), axis=1)

**def** evolve(self):

fig = plt.figure()

**for** step **in** range(self.max\_steps):

r1 = np.random.rand(self.population\_size, self.dim)

r2 = np.random.rand(self.population\_size, self.dim)

*# 更新速度和权重*

self.v = self.w\*self.v+self.c1\*r1\*(self.p-self.x)+self.c2\*r2\*(self.pg-self.x)

self.x = self.v + self.x

plt.clf()

plt.scatter(self.x[:, 0], self.x[:, 1], s=30, color=**'k'**)

plt.xlim(self.x\_bound[0], self.x\_bound[1])

plt.ylim(self.x\_bound[0], self.x\_bound[1])

plt.pause(0.01)

fitness = self.calculate\_fitness(self.x)

*# 需要更新的个体*

update\_id = np.greater(self.individual\_best\_fitness, fitness)

self.p[update\_id] = self.x[update\_id]

self.individual\_best\_fitness[update\_id] = fitness[update\_id]

*# 新一代出现了更小的fitness，所以更新全局最优fitness和位置*

**if** np.min(fitness) < self.global\_best\_fitness:

self.pg = self.x[np.argmin(fitness)]

self.global\_best\_fitness = np.min(fitness)

print(**'best fitness: %.5f, mean fitness: %.5f'** % (self.global\_best\_fitness, np.mean(fitness)))

pso = PSO(100, 100)

pso.evolve()

plt.show()