GENETIC_ALGORITHM

June 14, 2023

#GENETIC ALGORITHM

[2]:
```
pip install pygad
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-
wheels/public/simple/
Collecting pygad
  Downloading pygad-3.0.1-py3-none-any.whl (67 kB)
                             68.0/68.0 kB
2.8 MB/s eta 0:00:00
Requirement already satisfied: cloudpickle in
/usr/local/lib/python3.10/dist-packages (from pygad) (2.2.1)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.10/dist-
packages (from pygad) (3.7.1)
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages
(from pygad) (1.22.4)
Requirement already satisfied: contourpy>=1.0.1 in
/usr/local/lib/python3.10/dist-packages (from matplotlib->pygad) (1.0.7)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-
packages (from matplotlib->pygad) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in
/usr/local/lib/python3.10/dist-packages (from matplotlib->pygad) (4.39.3)
Requirement already satisfied: kiwisolver>=1.0.1 in
/usr/local/lib/python3.10/dist-packages (from matplotlib->pygad) (1.4.4)
Requirement already satisfied: packaging>=20.0 in
/usr/local/lib/python3.10/dist-packages (from matplotlib->pygad) (23.1)
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.10/dist-
packages (from matplotlib->pygad) (8.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in
/usr/local/lib/python3.10/dist-packages (from matplotlib->pygad) (3.0.9)
Requirement already satisfied: python-dateutil>=2.7 in
/usr/local/lib/python3.10/dist-packages (from matplotlib->pygad) (2.8.2)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-
packages (from python-dateutil>=2.7->matplotlib->pygad) (1.16.0)
Installing collected packages: pygad
Successfully installed pygad-3.0.1
```

[22]:
```
import numpy
import matplotlib.pyplot
import pygad
```

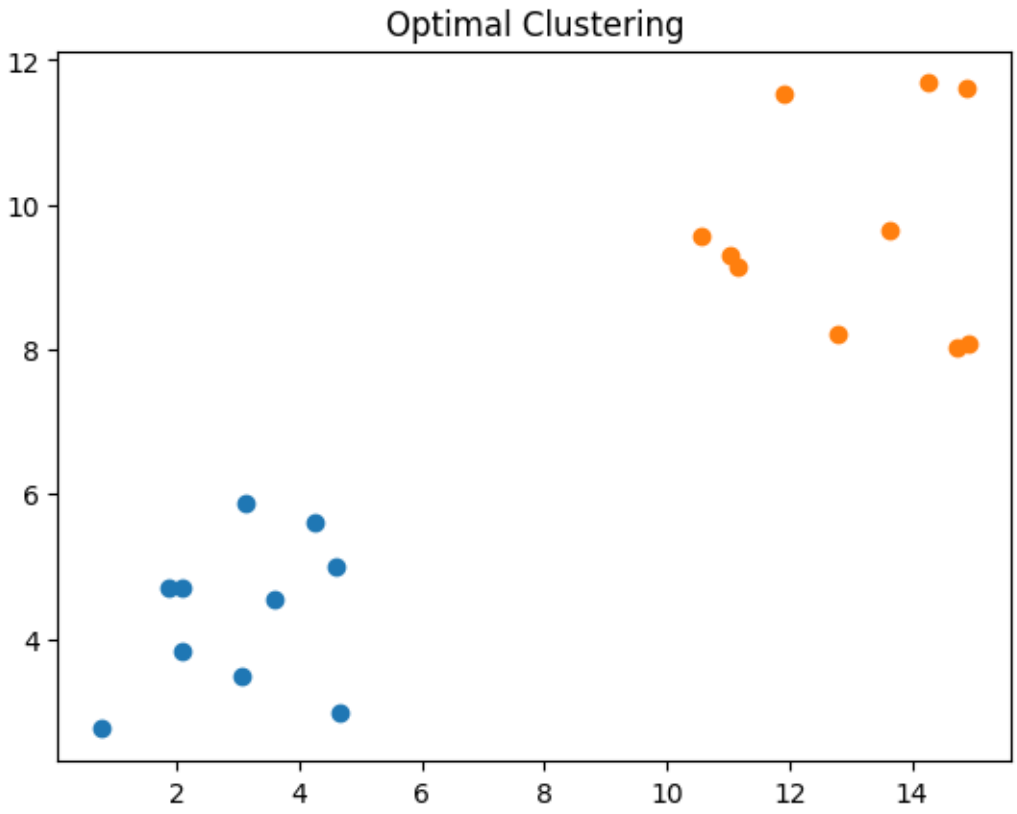
[23]:
```
cluster1_num_samples = 10
cluster1_x1_start = 0
cluster1_x1_end = 5
cluster1_x2_start = 2
cluster1_x2_end = 6
cluster1_x1 = numpy.random.random(size=(cluster1_num_samples))
cluster1_x1 = cluster1_x1 * (cluster1_x1_end - cluster1_x1_start) +␣
 ↪cluster1_x1_start
cluster1_x2 = numpy.random.random(size=(cluster1_num_samples))
cluster1_x2 = cluster1_x2 * (cluster1_x2_end - cluster1_x2_start) +␣
 ↪cluster1_x2_start
cluster2_num_samples = 10
cluster2_x1_start = 10
cluster2_x1_end = 15
cluster2_x2_start = 8
cluster2_x2_end = 12
cluster2_x1 = numpy.random.random(size=(cluster2_num_samples))
cluster2_x1 = cluster2_x1 * (cluster2_x1_end - cluster2_x1_start) +␣
 ↪cluster2_x1_start
cluster2_x2 = numpy.random.random(size=(cluster2_num_samples))
cluster2_x2 = cluster2_x2 * (cluster2_x2_end - cluster2_x2_start) +␣
 ↪cluster2_x2_start
```

[24]:
```
c1 = numpy.array([cluster1_x1, cluster1_x2]).T
c2 = numpy.array([cluster2_x1, cluster2_x2]).T
data = numpy.concatenate((c1, c2), axis=0)
data
```

[24]:
```
array([[ 4.66938982,  2.9762822 ],
       [ 2.07543695,  3.83859086],
       [ 3.13531723,  5.87199961],
       [ 3.58970955,  4.54453056],
       [ 3.07629042,  3.49616418],
       [ 2.08771405,  4.71284543],
       [ 1.88474681,  4.71970647],
       [ 4.59947746,  4.99412735],
       [ 0.77009612,  2.75535743],
       [ 4.26303624,  5.6139186 ],
       [11.02428498,  9.29808227],
       [14.91307711,  8.08093762],
       [14.73543368,  8.02161854],
       [10.56723057,  9.56139784],
       [14.2527715 , 11.67962258],
```

```
[14.87836598, 11.61164421],
[13.63883571,  9.65565323],
[12.79816527,  8.22107239],
[11.89374054, 11.54060237],
[11.15415657,  9.15332527]])
```

[25]:
```
matplotlib.pyplot.scatter(cluster1_x1, cluster1_x2)
matplotlib.pyplot.scatter(cluster2_x1, cluster2_x2)
matplotlib.pyplot.title("Optimal Clustering")
matplotlib.pyplot.show()
```



[53]:
```
def euclidean_distance(X, Y):
  return numpy.sqrt(numpy.sum(numpy.power(X - Y, 2),axis=1))
```

[54]:
```
def cluster_data(solution, solution_idx):
  global num_cluster, data
  feature_vector_length = data.shape[1]
  cluster_centers = []
  all_clusters_dists = []
  clusters = []
  clusters_sum_dist = []
```

```
  for clust_idx in range(num_clusters):
    cluster_centers.append(solution[feature_vector_length*clust_idx:
↪feature_vector_length*(clust_idx+1)])
    cluster_center_dists = euclidean_distance(data, cluster_centers[clust_idx])
    all_clusters_dists.append(numpy.array(cluster_center_dists))

  cluster_centers = numpy.array(cluster_centers)
  all_clusters_dists = numpy.array(all_clusters_dists)
  cluster_indices = numpy.argmin(all_clusters_dists, axis=0)
  for clust_idx in range(num_clusters):
    clusters.append(numpy.where(cluster_indices == clust_idx)[0])

    if len(clusters[clust_idx]) == 0:
      clusters_sum_dist.append(0)
    else:
      clusters_sum_dist.append(numpy.
↪sum(all_clusters_dists[clust_idx,clusters[clust_idx]]))
  clusters_sum_dist = numpy.array(clusters_sum_dist)
  return cluster_centers, all_clusters_dists, cluster_indices, clusters,␣
↪clusters_sum_dist
```

[55]:
```
def fitness_func(ga_instance,solution, solution_idx):
  _, _, _, _, clusters_sum_dist = cluster_data(solution, solution_idx)
  fitness = 1.0 / (numpy.sum(clusters_sum_dist) + 0.00000001)
  return fitness
```

[56]:
```
num_clusters = 2
num_genes = num_clusters * data.shape[1]
ga_instance = pygad.GA(num_generations=100,
                       sol_per_pop=10,
                       num_parents_mating=5,
                       init_range_low=-6,
                       init_range_high=20,
                       keep_parents=2,
                       num_genes=num_genes,
                       fitness_func=fitness_func,
                       suppress_warnings=True)
ga_instance.run()
```

[58]:
```
best_solution, best_solution_fitness, best_solution_idx = ga_instance.
  ↪best_solution()
print("Best solution is {bs}".format(bs=best_solution))
print("Fitness of the best solution is {bsf}".format(bsf=best_solution_fitness))
print("Best solution found after {gen} generations".format(gen=ga_instance.
  ↪best_solution_generation))
```

```
Best solution is [ 2.93437306  4.4450975  13.10073844  9.60153479]
Fitness of the best solution is 0.028412177948481575
Best solution found after 29 generations
```

[61]:
```
cluster_centers, all_clusters_dists, cluster_indices, clusters,␣
 ↪clusters_sum_dist= cluster_data(best_solution,best_solution_idx)
```

[63]:
```
for cluster_idx in range(num_clusters):
 cluster_x = data[clusters[cluster_idx], 0]
 cluster_y = data[clusters[cluster_idx], 1]
 matplotlib.pyplot.scatter(cluster_x, cluster_y)
 matplotlib.pyplot.scatter(cluster_centers[cluster_idx, 0],␣
 ↪cluster_centers[cluster_idx,1],linewidth=5)
matplotlib.pyplot.title("Clustering using PyGAD")
matplotlib.pyplot.show()
```