Student_2_Naive_ES

April 14, 2021

1 First steps in population based optimization

[ ]:

```
%matplotlib notebook
from math import sin, cos, sqrt, pi
from matplotlib import cm
import numpy as np
import matplotlib.pyplot as plt
```

[ ]:

```
def loss_function(x, a=10):
    dummy = a * len(x)
    for ii in range(len(x)):
        dummy += x[ii] ** 2 - a * cos(2 * pi * x[ii])
    return dummy
```

[ ]:

```
def plot_loss(ax):

    x = np.linspace(-5, 5, 200)
    y = np.linspace(-5, 5, 200)
    X, Y = np.meshgrid(x, y)

    Z = np.zeros_like(X)
    for ii in range(X.shape[0]):
        for jj in range(X.shape[1]):
            Z[ii][jj] = loss_function([X[ii][jj], Y[ii][jj]])
    img = ax.contour(X, Y, Z, levels=30, cmap=cm.coolwarm)
    plt.colorbar(img, ax=ax)

    return ax
```

[ ]:

```
# Simple Evolution Strategy
n_evolutions = 30
n_population = 50
start = [3.5, 3.8]
init_population = [start] * n_population
sig_0 = 0.4
population = np.random.normal(init_population, sig_0)
centers = [start]
```

```
# Inits for plot
fig = plt.figure()
ax = fig.gca()

axes = plt.gca()
axes.set_xlim([-5, 5])
axes.set_ylim([-5, 5])

plt.ion()
fig.show()
fig.canvas.draw()

plot_loss(ax)
fig.canvas.draw()

for episode in range(n_evolutions):
    # Population Update Strategy
    scores = []
    for ii in range(n_population):
        ax.plot(population[ii,0], population[ii,1], ".b")

    # Todo: evaluate the loss for every member of the population and save the value in score
    for ii in range(n_population):
        scores.append(function(population[ii, :]))

    # Todo: Save the id of the "best" member of the population in best_id
    best_id = np.argmin(scores)

    # Todo: Append the best member of the population to centers
    centers.append(list(population[best_id, 0:2]))

    # Todo: Update the population initializing a new population around the best member of the former.
    #       Keep sigma constant with respect to the initialization
    tmp_population = [
        [population[best_id, 0], population[best_id, 1]]
    ] * n_population
    population = np.random.normal(tmp_population, sig_0)

    # Plot population
    fig.canvas.draw()

centers = np.array(centers)

plt.plot(centers[:,0], centers[:,1], ".-g", label="Centers")
plt.plot(centers[0,0], centers[0,1], "or", label="Start")
plt.plot(centers[-1,0], centers[-1,1], "xr", label="End")
plt.legend()
```

[ ]:

```

```