Student_1_1plus1_ES

April 14, 2021

1 1+1 Evolution Strategies

Please note, the adaptive 1+1 ES are optional.

[ ]:
```
%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
```

[ ]:
```
# 1+1 Evolution Strategy

# Parameters
n_evolutions = 50
init_population = [2.5, 2.8]
sig = 0.6
population = np.random.normal(init_population, sig)
```

```
population_log = [population]

# Parameters for adaption
length_log = 5
log = [0]*length_log
c = 0.817

# 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)


for episode in range(n_evolutions):

    # Todo: Implement 1+1 update strategy

    # Todo: Generate a random sample
    sample = 0

    # Todo: Calculate loss (function) for new sample and old population


    # Todo: Compare loss from the sample to the loss of the old population.
    #       Store in better if the new loss is smaller than the former


    # Todo: Update population
    if better:
        ax.plot(sample[0], sample[1], ".g")
    else:
        ax.plot(sample[0], sample[1], ".b")

    # Additional code here
 ↪=======================================================

    # Todo: Implement the presented update strategy for sigma.
```

```
    #       This exericse comes with further instructions to make it a little␣
↪more challenging.
    #       Please feel free to come back to this exercise after completing the␣
↪other notebooks.




    #␣
↪===========================================================================

    # Plot population
    population_log.append(population)
    fig.canvas.draw()

population_log = np.array(population_log)

ax.plot(population_log[:,0], population_log[:,1], "-g", label="Intermediate␣
↪updates")
ax.plot(population_log[0,0], population_log[0,1], "xg", label="Start")
ax.plot(population_log[-1,0], population_log[-1,1], "*g", label="End")
plt.legend()
```

[ ]:

```

```