README

March 27, 2019

```
In [3]: from IPython.display import display, HTML

        display(HTML(data="""
        <style>
            div#notebook-container    { width: 40%; }
            div#menubar-container     { width: 65%; }
            div#maintoolbar-container { width: 99%; }
        </style>
        """))
```

```
<IPython.core.display.HTML object>
```

```
In [1]: from main import *
        from PlotBuilder import *
```

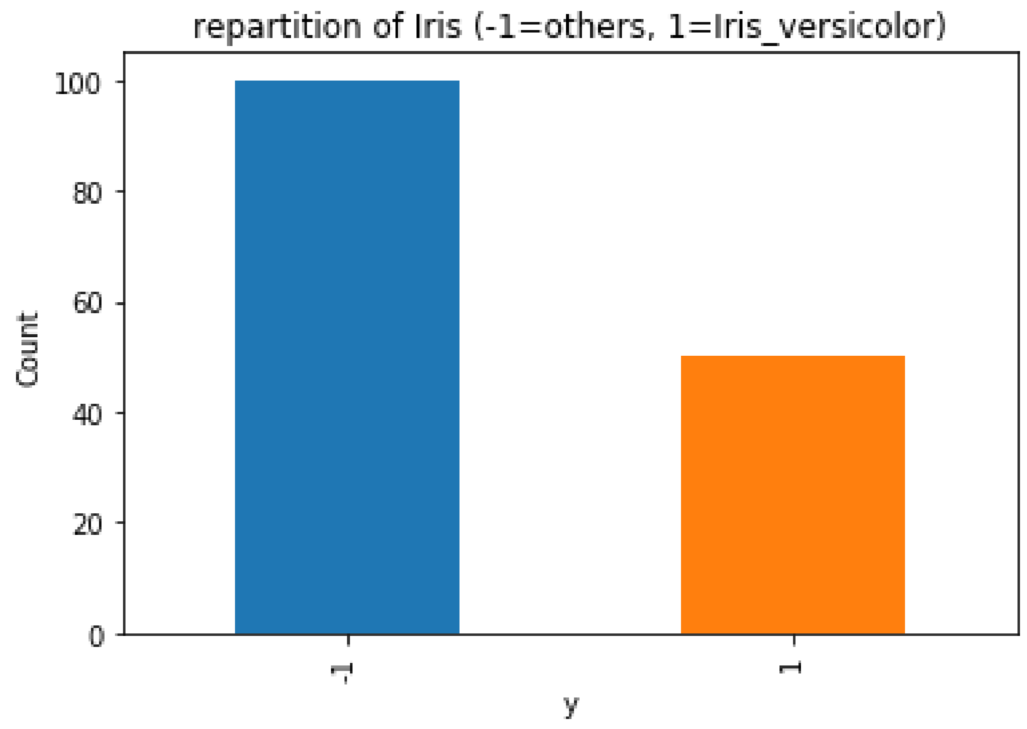
- 'yielding_pas' mean how many **rows** you learn between each testing process.

the data is readed in cycle.
during all testing process we add an other point on the plot graph.

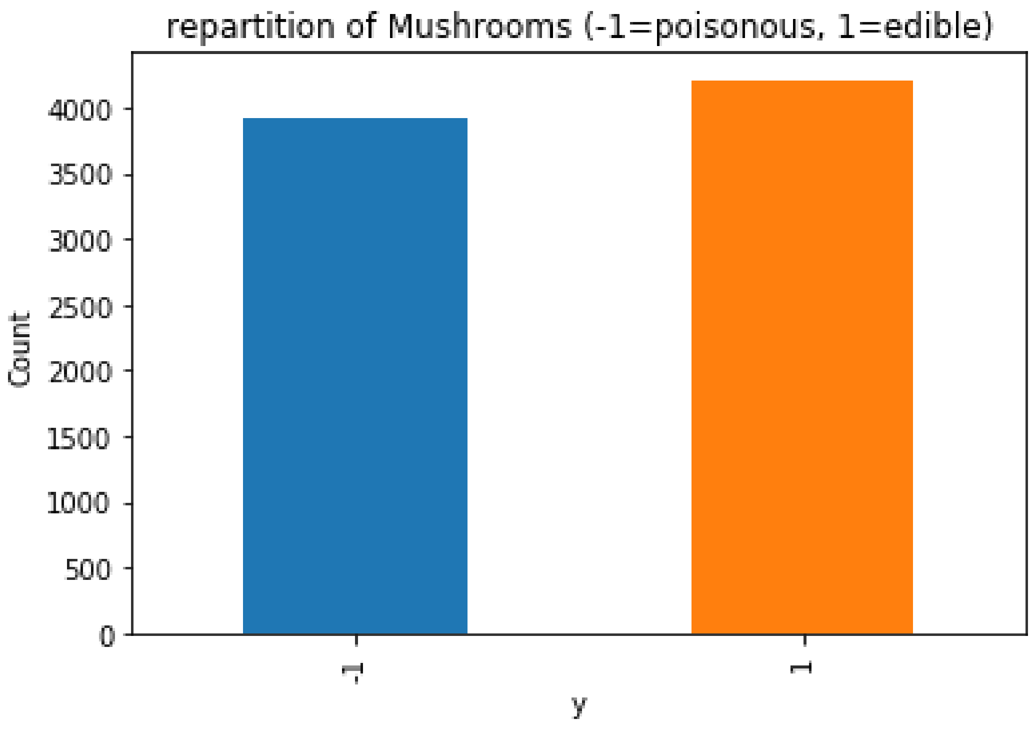
```
In [2]: MAX_PLOT_X_PERCEPTRON = 100
        MAX_PLOT_X = 4500

        iris_X, iris_y = Data.load_iris_data()
        mushroom_X, mushroom_y = Data.load_mushroom_data()
        spambase_X, spambase_y = Data.load_spambase_data()
```

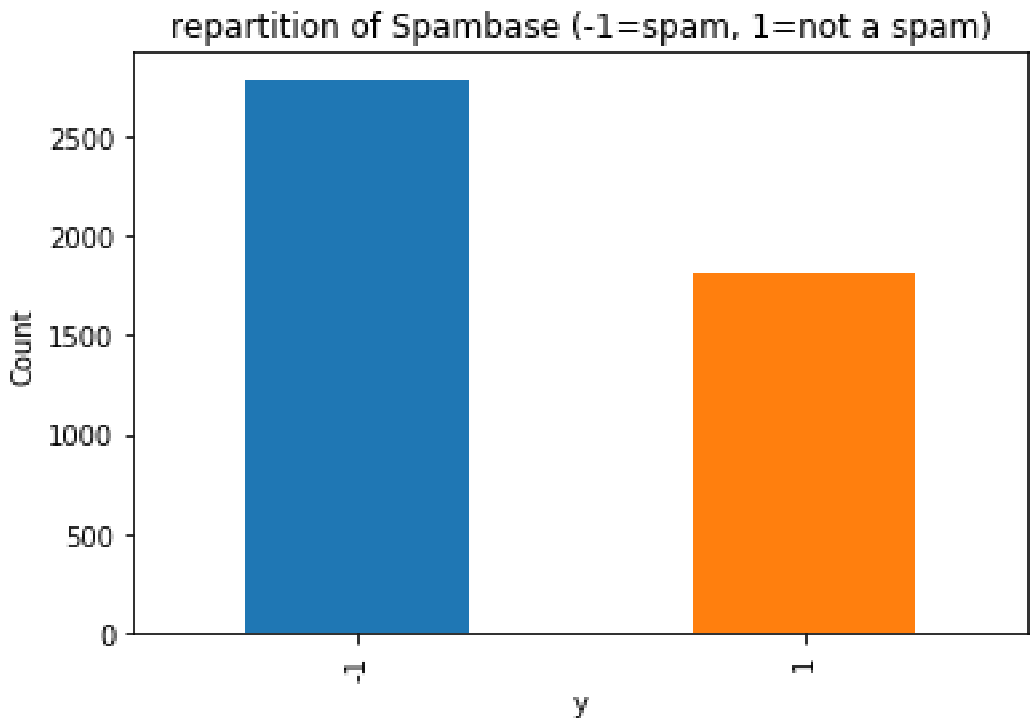
```
In [3]: Plot_builder.y_repartition(iris_y, title="repartition of Iris \
        (-1=others, 1=Iris_versicolor)")
```



```
In [4]: Plot_builder.y_repartition(mushroom_y, title="repartition of Mushrooms\
        (-1=poisonous, 1=edible)")
```

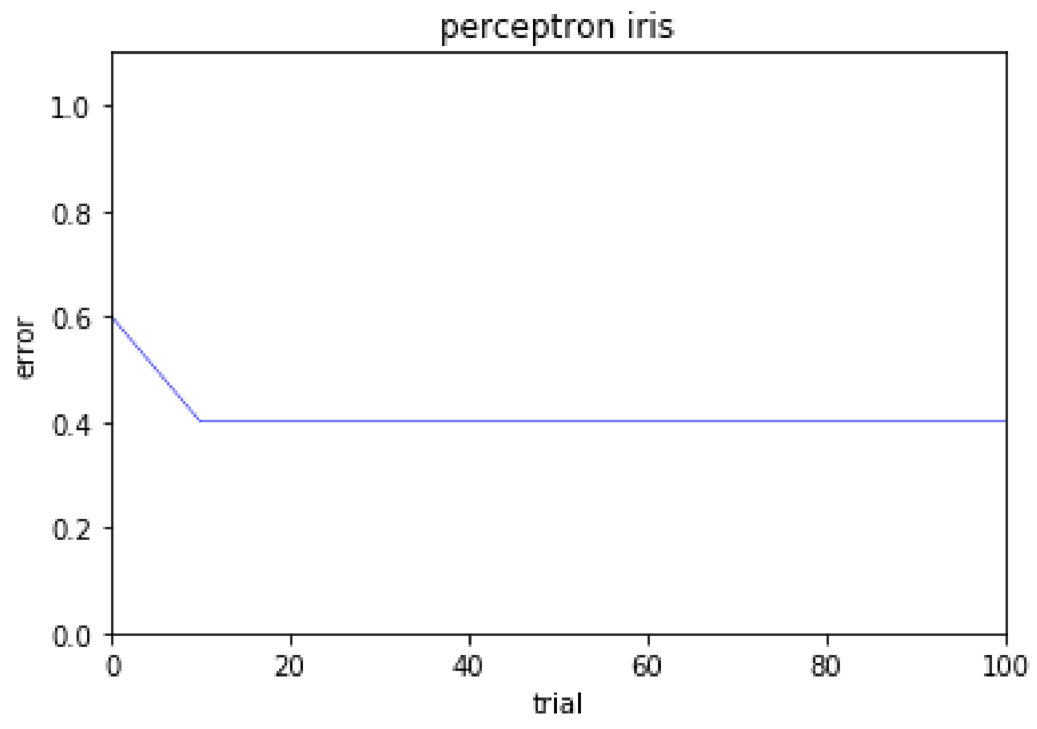


```
In [5]: Plot_builder.y_repartition(spambase_y, title="repartition of Spambase \
        (-1=spam, 1=not a spam)")
```

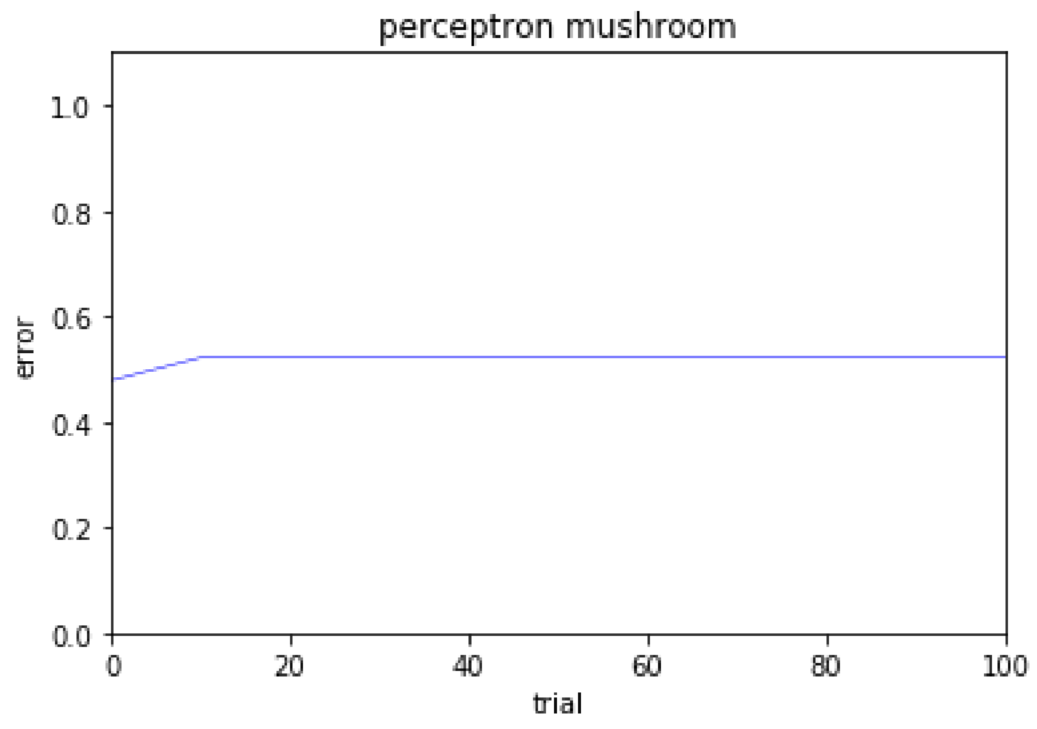


1 Perceptron

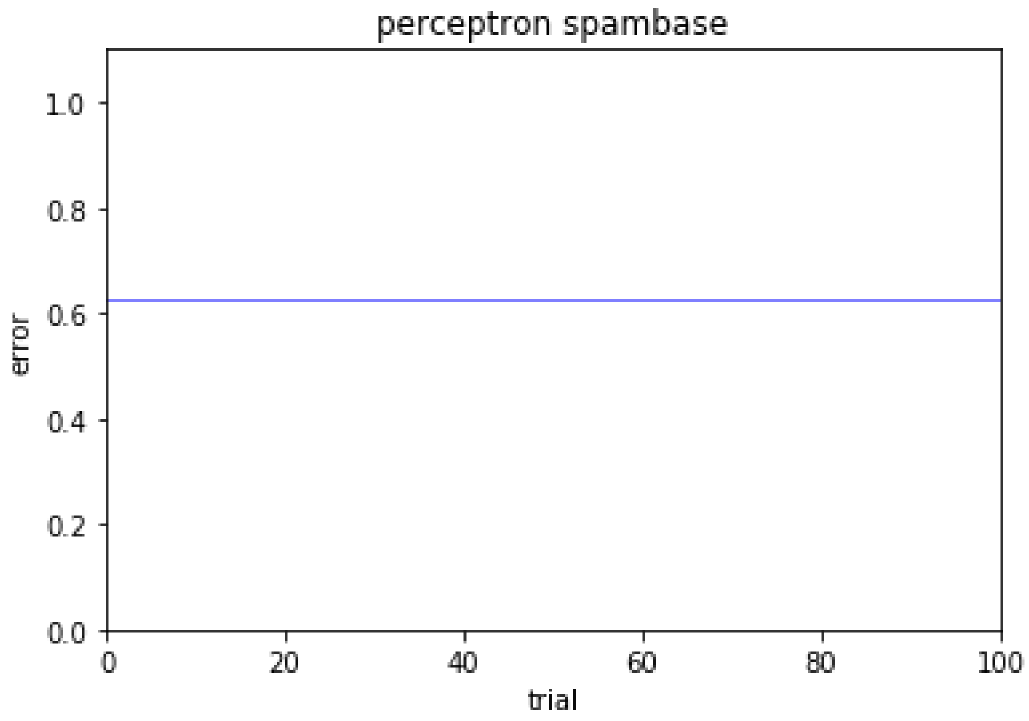
```
In [6]: perceptron = Algo(iris_X, iris_y, algo=Perceptron, yielding_pas=10)
        perceptron.do_all(nb_iterations=MAX_PLOT_X_PERCEPTRON,name="perceptron iris",
                          max_x=MAX_PLOT_X_PERCEPTRON)
```



```
In [7]: perceptron = Algo(mushroom_X, mushroom_y, algo=Perceptron, yielding_pas=10)
        perceptron.do_all(nb_iterations=MAX_PLOT_X_PERCEPTRON,
                          name="perceptron mushroom", max_x=MAX_PLOT_X_PERCEPTRON)
```

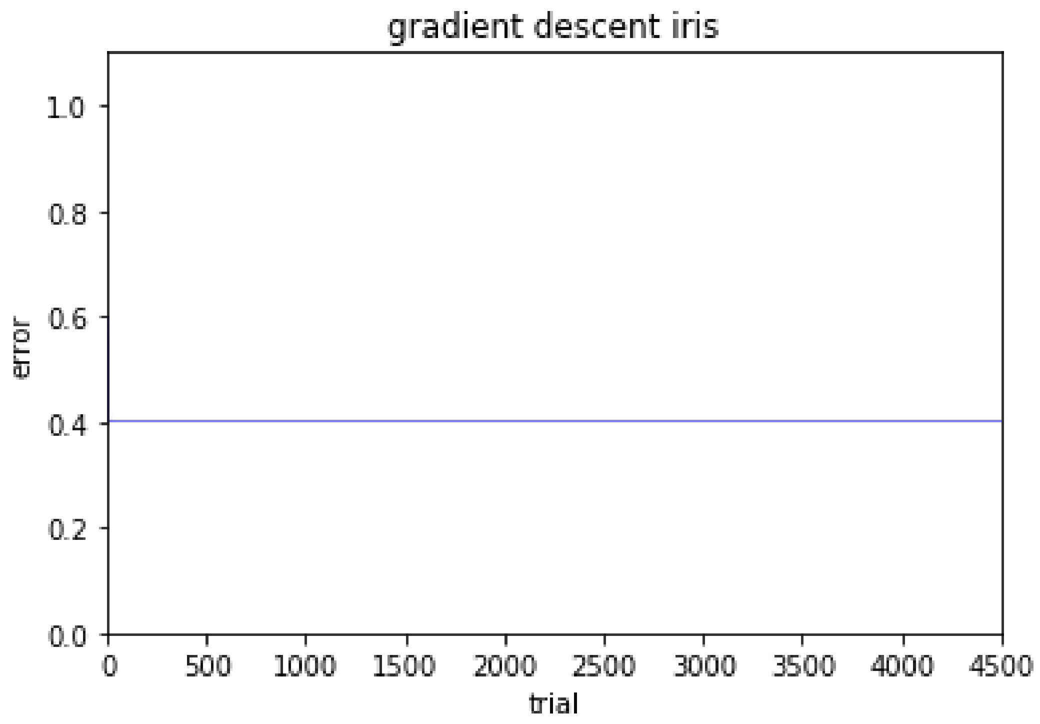


```
In [8]: perceptron = Algo(spambase_X, spambase_y, algo=Perceptron, yielding_pas=10)
        perceptron.do_all(nb_iterations=MAX_PLOT_X_PERCEPTRON,
                          name="perceptron spambase", max_x=MAX_PLOT_X_PERCEPTRON)
```

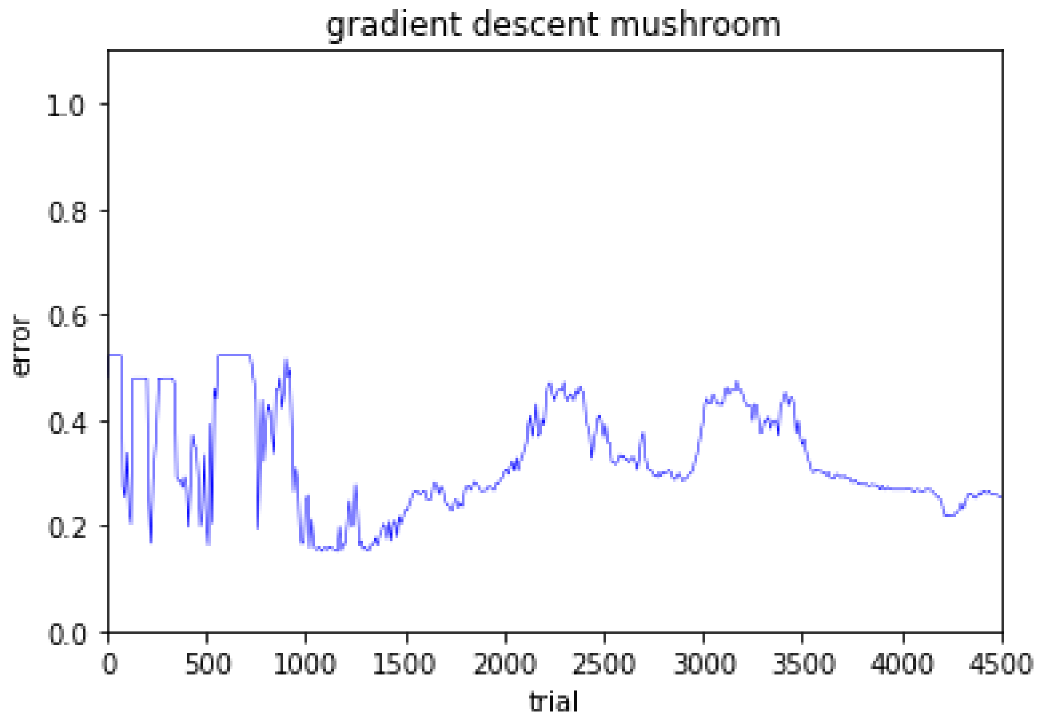


2 Descent gradient

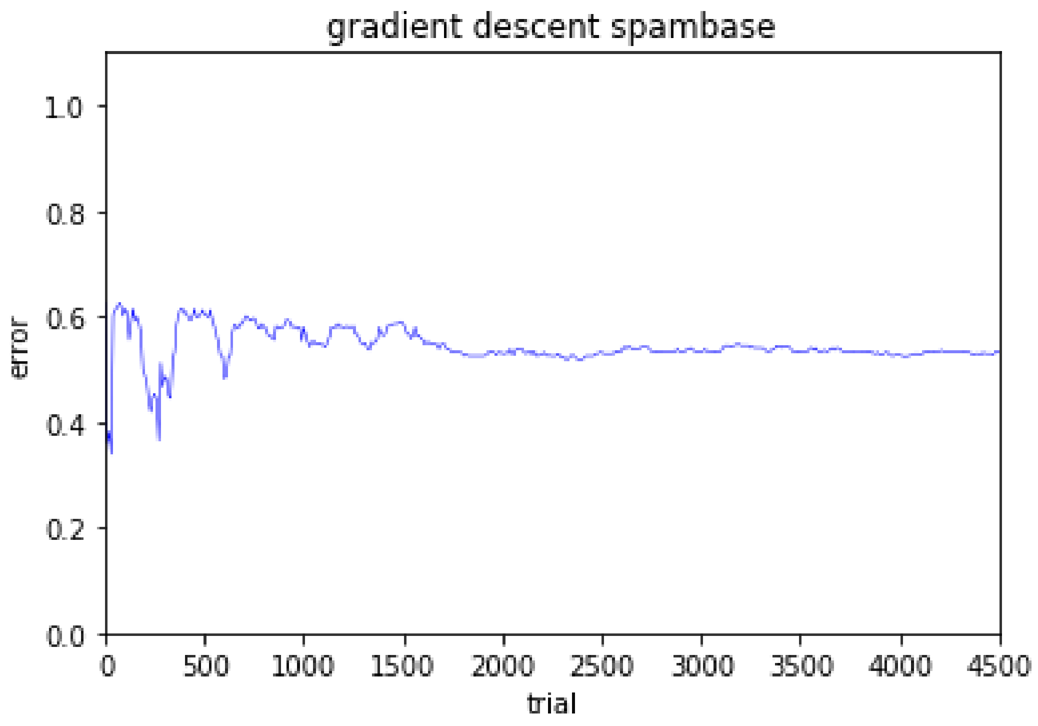
```
In [9]: gradient_descent = Algo(iris_X, iris_y, algo=GradientDescent,
                                yielding_pas=10, gradient_eta=2)
        gradient_descent.do_all(nb_iterations=MAX_PLOT_X,
                                name="gradient descent iris", max_x=MAX_PLOT_X)
```



```
In [10]: gradient_descent = Algo(mushroom_X, mushroom_y, algo=GradientDescent,
                                 yielding_pas=10, gradient_eta=2)
         gradient_descent.do_all(nb_iterations=MAX_PLOT_X,
                                 name="gradient descent mushroom", max_x=MAX_PLOT_X)
```

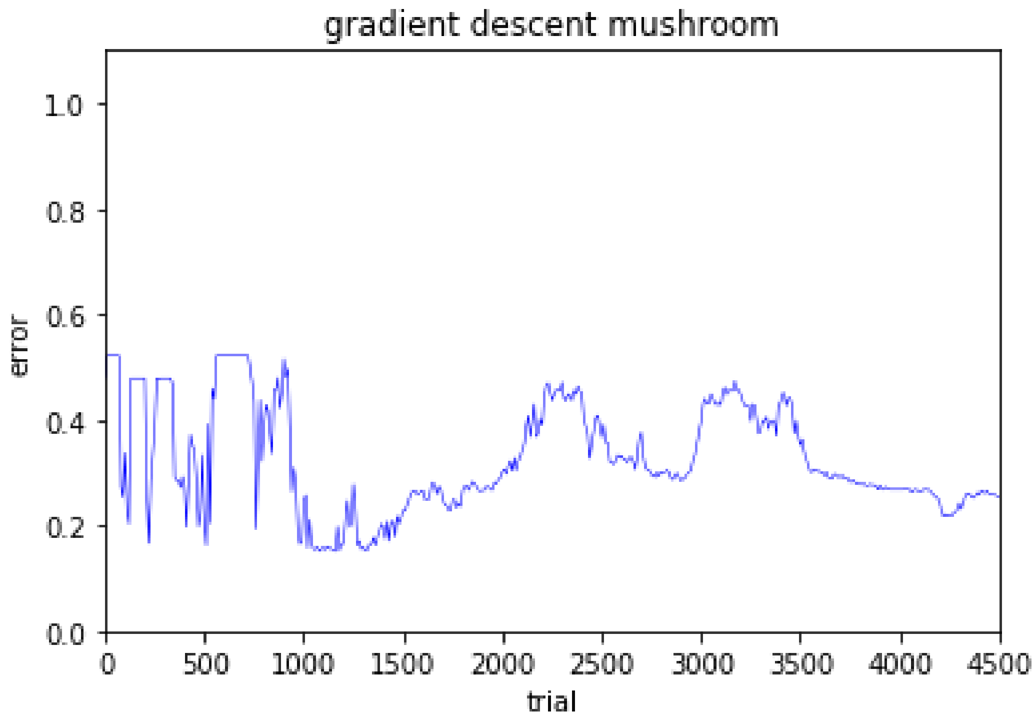


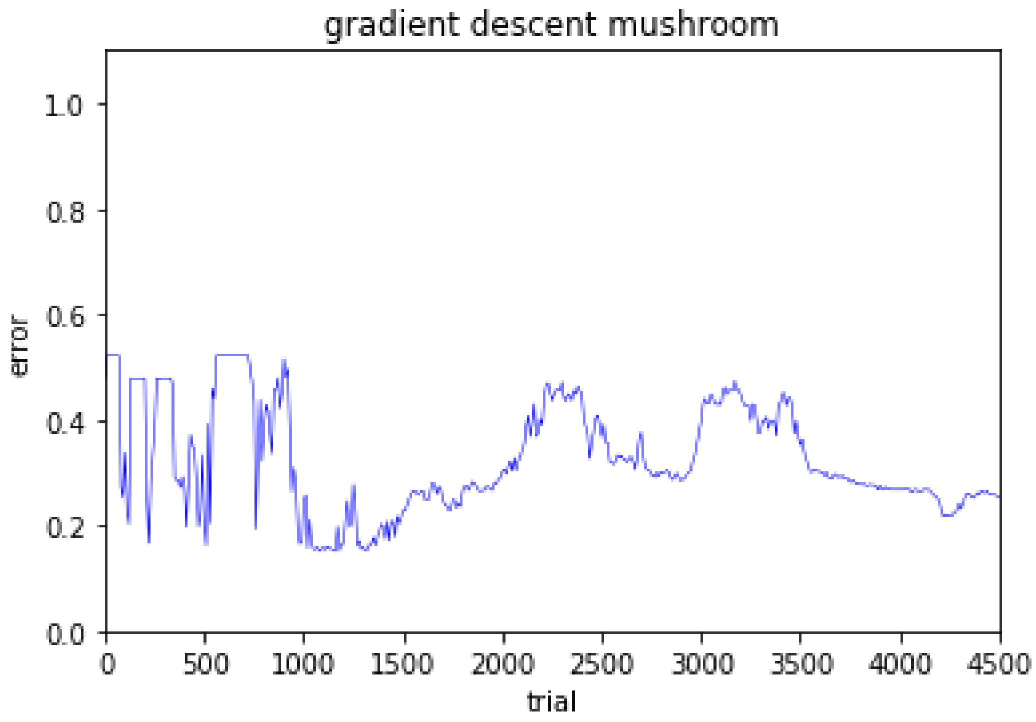
```
In [11]: gradient_descent = Algo(spambase_X, spambase_y, algo=GradientDescent,
                                 yielding_pas=10, gradient_eta=2)
         gradient_descent.do_all(nb_iterations=MAX_PLOT_X,
                                 name="gradient descent spambase", max_x=MAX_PLOT_X)
```

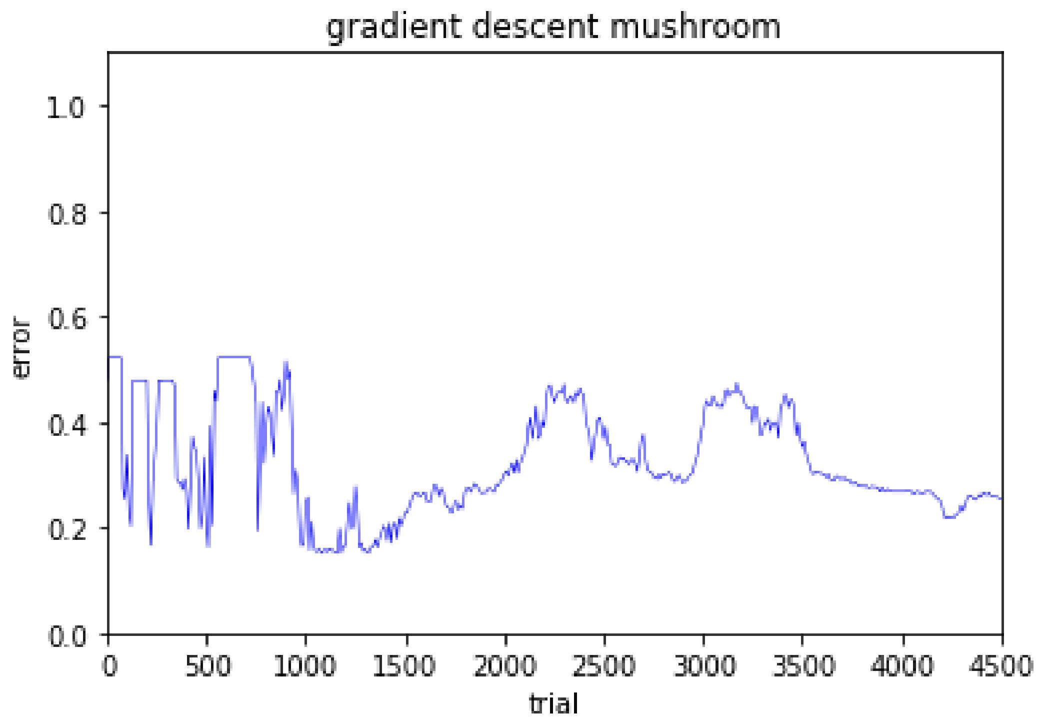


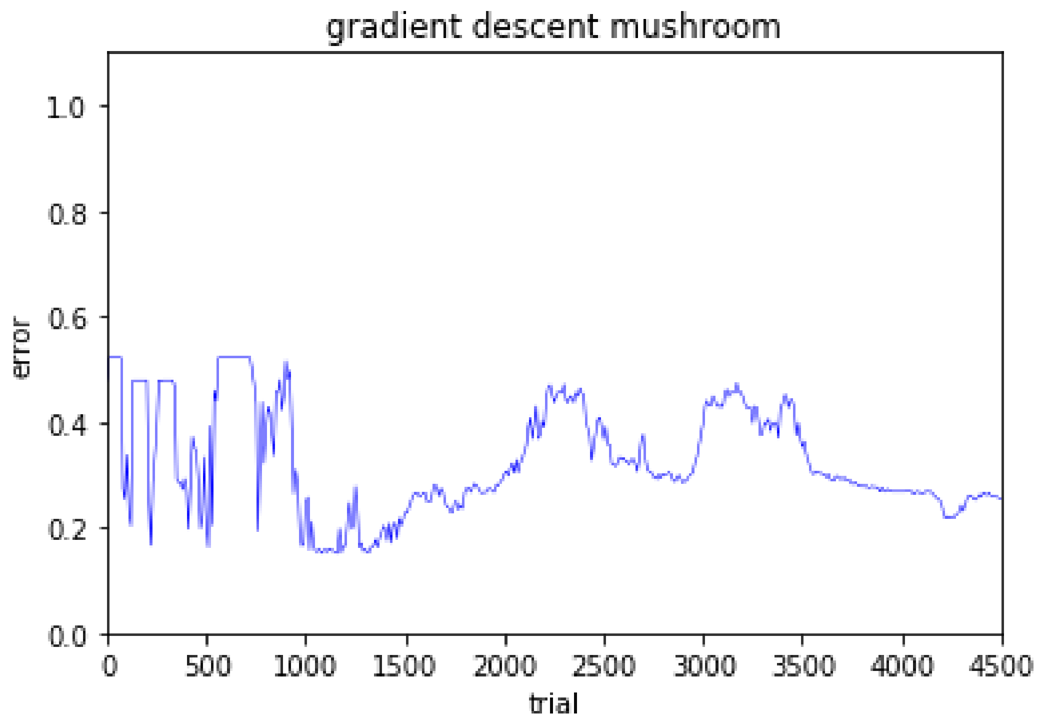
3 searching for best ETA

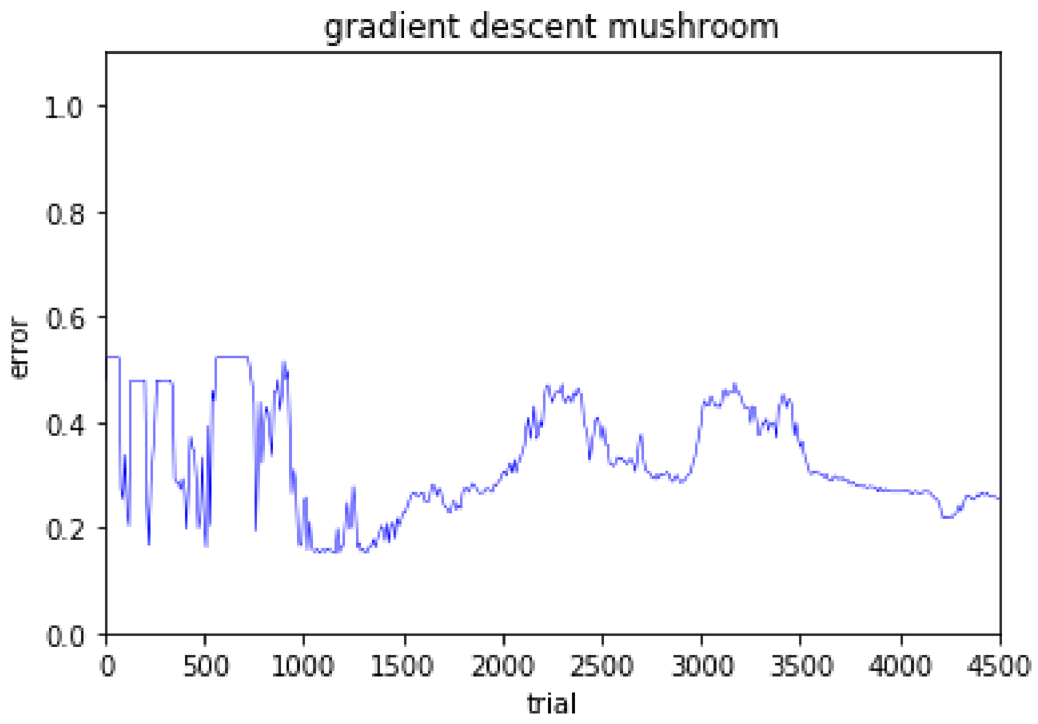
```
In [15]: for eta in range(100,1000,200):
             gradient_descent = Algo(mushroom_X, mushroom_y, algo=GradientDescent,
                                     yielding_pas=10, gradient_eta=eta)
             gradient_descent.do_all(nb_iterations=MAX_PLOT_X,
                                     name="gradient descent mushroom",
                                     max_x=MAX_PLOT_X)
```


gradient descent mushroom
error
trial
0.0
0.2
0.4
0.6
0.8
1.0
0
500
1000
1500
2000
2500
3000
3500
4000
4500


gradient descent mushroom
error
trial
0.0
0.2
0.4
0.6
0.8
1.0
0
500
1000
1500
2000
2500
3000
3500
4000
4500

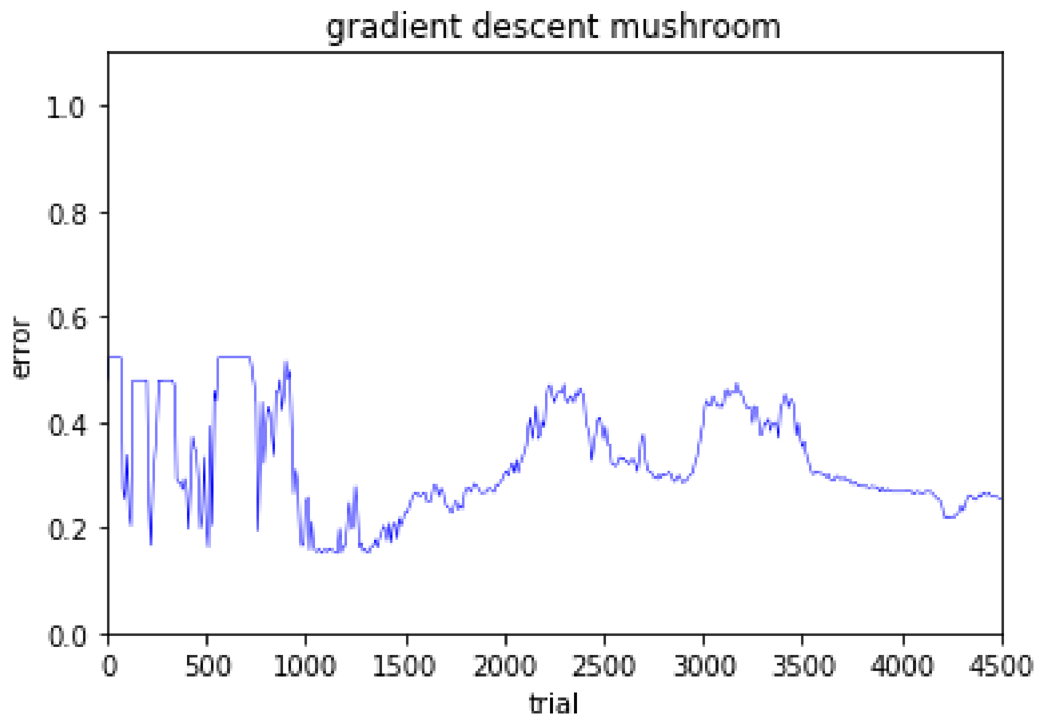

gradient descent mushroom
error
trial
0.0
0.2
0.4
0.6
0.8
1.0
0
500
1000
1500
2000
2500
3000
3500
4000
4500

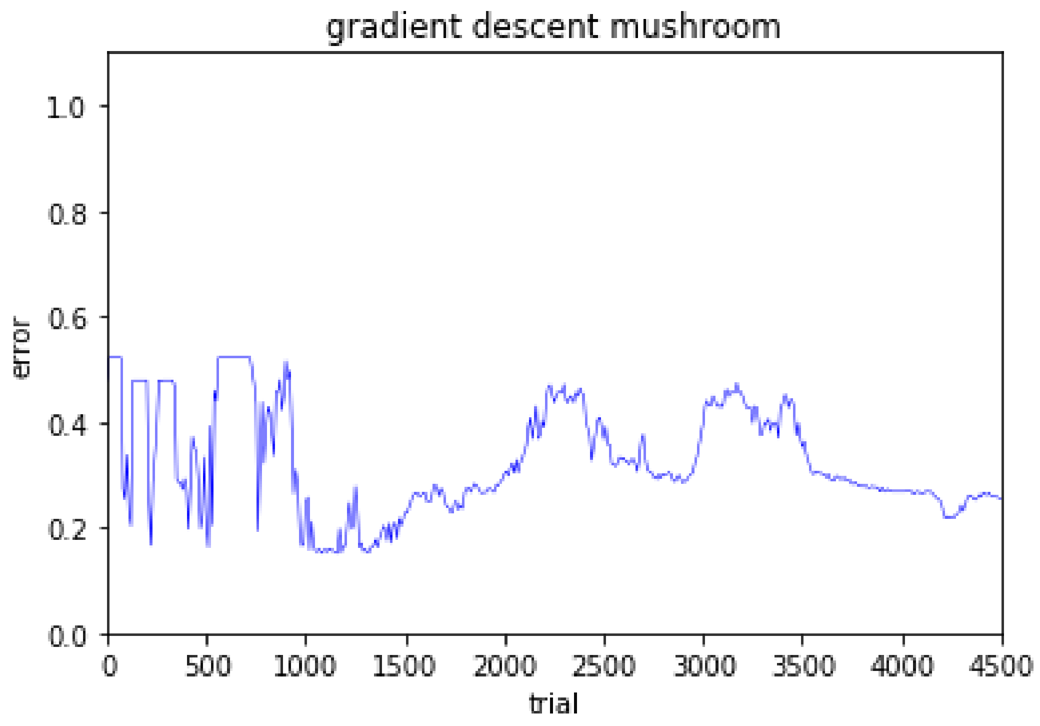

gradient descent mushroom
error
trial
0.0
0.2
0.4
0.6
0.8
1.0
0
500
1000
1500
2000
2500
3000
3500
4000
4500

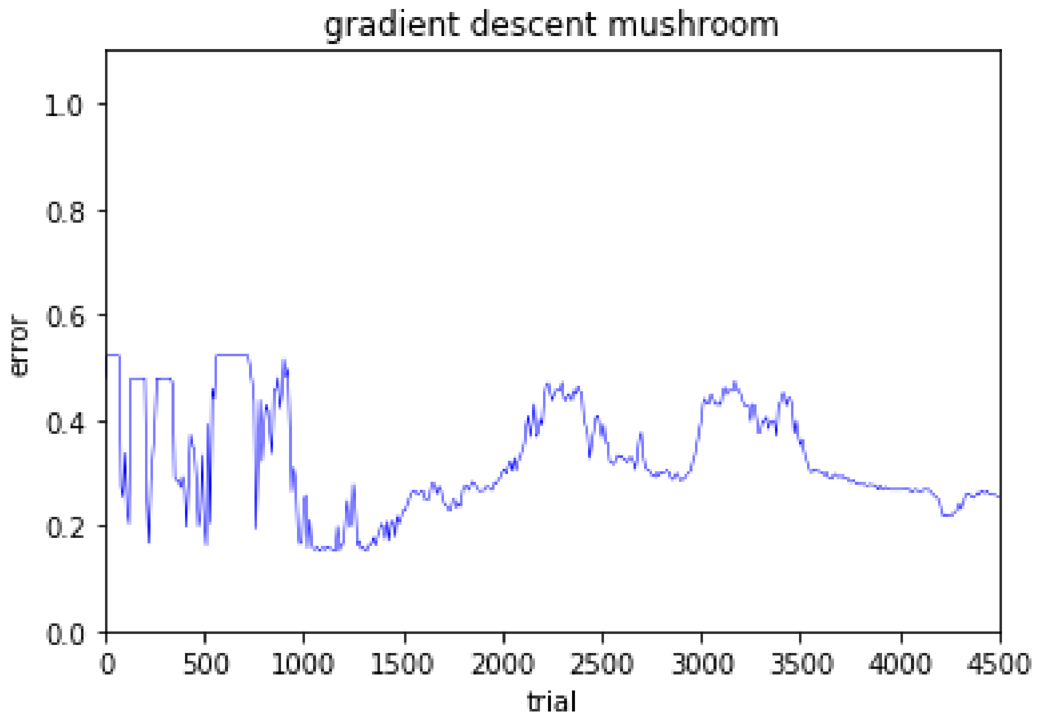


3.1 notting change

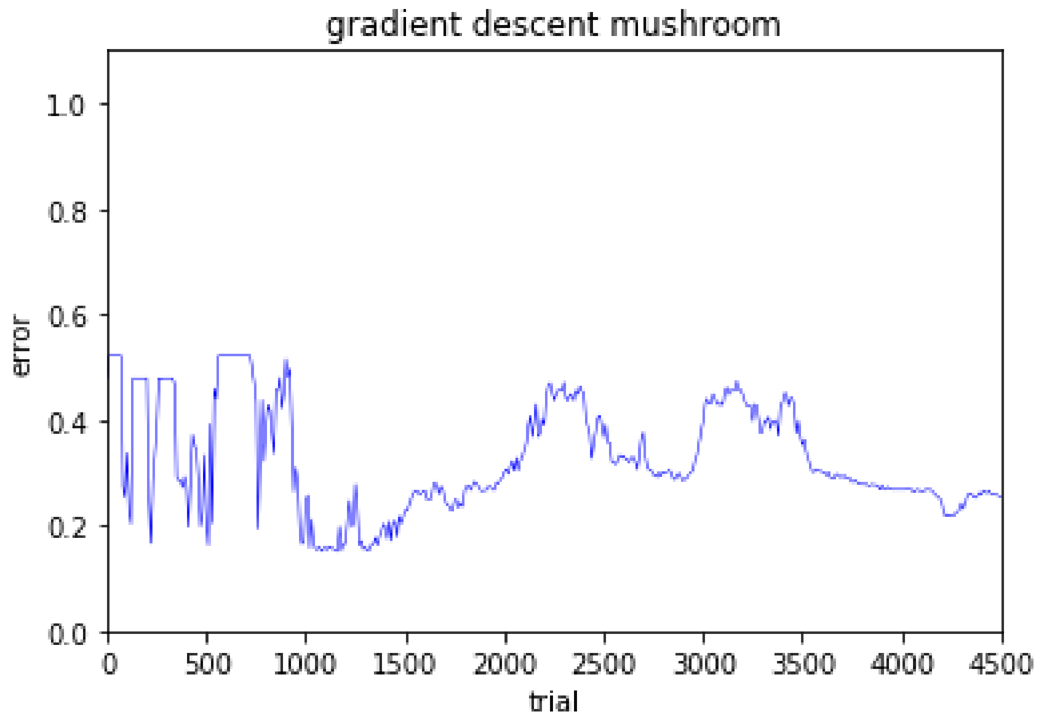
```
In [18]: for eta in [-1000,-500,-100]:
             gradient_descent = Algo(mushroom_X, mushroom_y,
                                     algo=GradientDescent, yielding_pas=10,
                                     gradient_eta=eta)
             gradient_descent.do_all(nb_iterations=MAX_PLOT_X,
                                     name="gradient descent mushroom",
                                     max_x=MAX_PLOT_X)
```

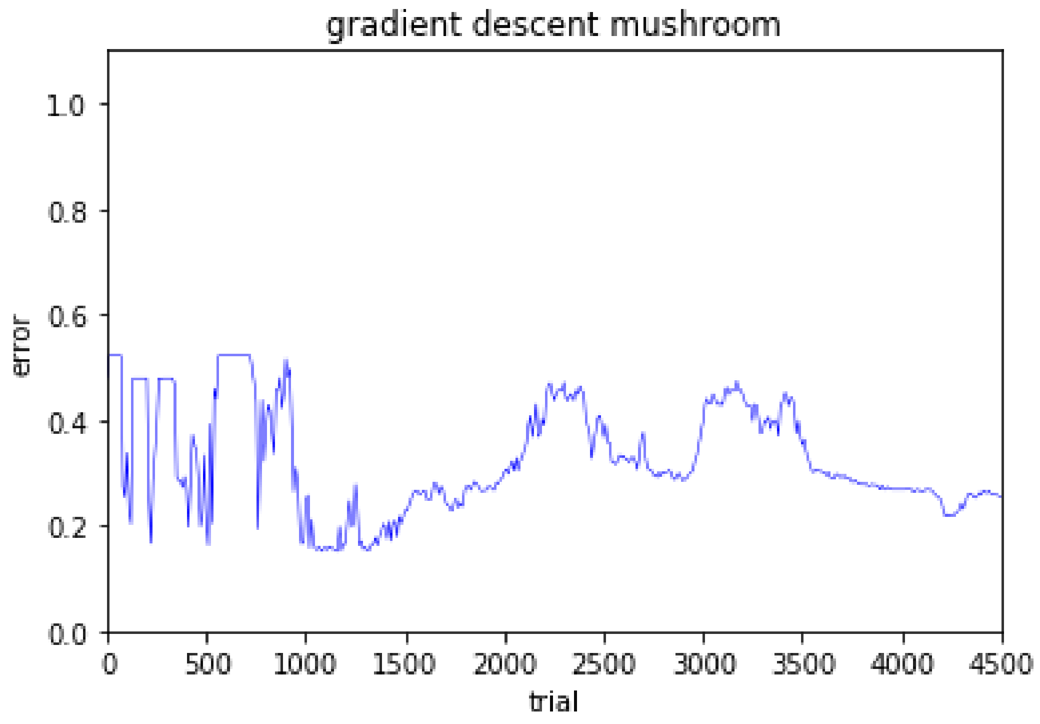

gradient descent mushroom
error
trial
0.0
0.2
0.4
0.6
0.8
1.0
0
500
1000
1500
2000
2500
3000
3500
4000
4500

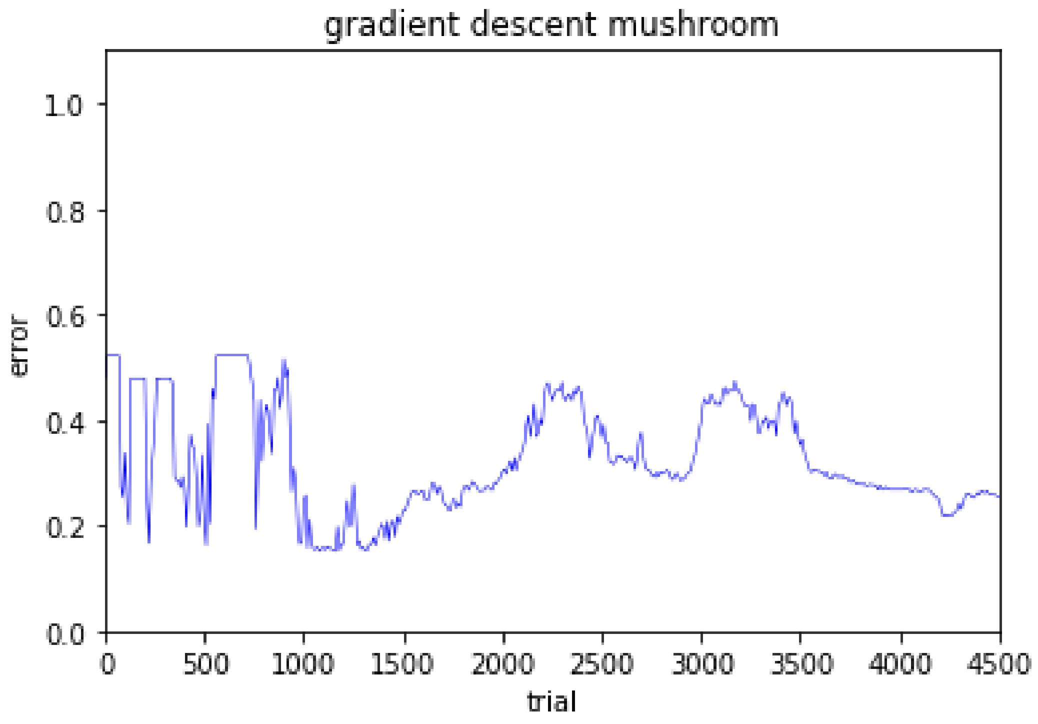

gradient descent mushroom
error
trial
0.0
0.2
0.4
0.6
0.8
1.0
0
500
1000
1500
2000
2500
3000
3500
4000
4500

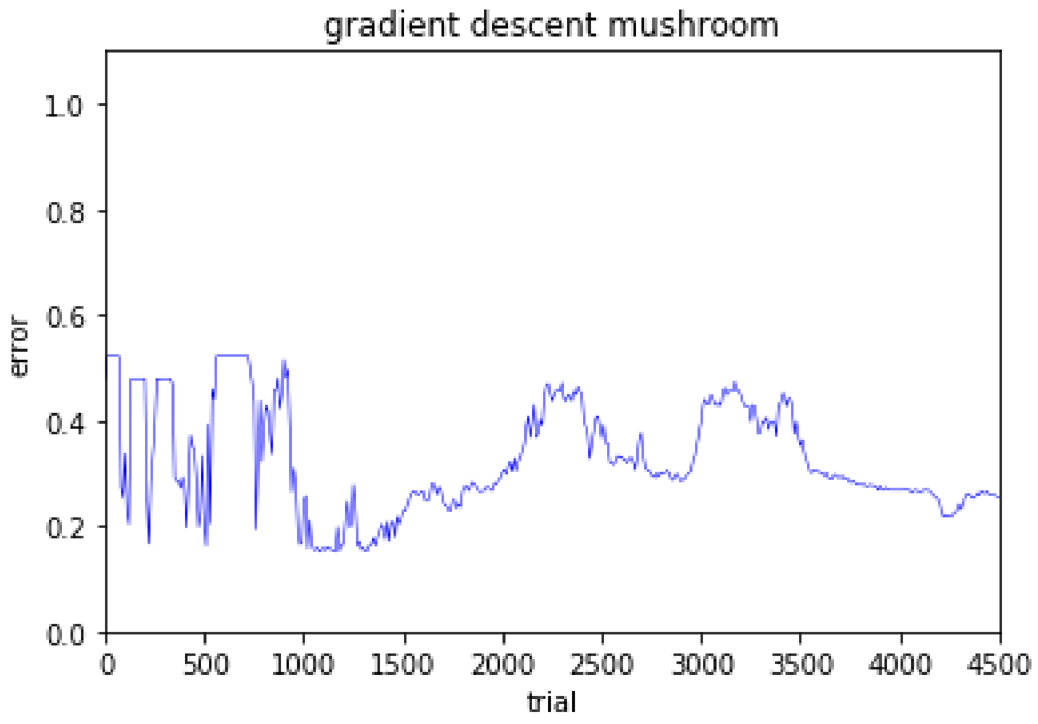


```
In [19]: for eta in [0.2, 0.8, -0.3, -0.7]:
             gradient_descent = Algo(mushroom_X, mushroom_y, algo=GradientDescent,
                                     yielding_pas=10, gradient_eta=eta)
             gradient_descent.do_all(nb_iterations=MAX_PLOT_X,
                                     name="gradient descent mushroom",
                                     max_x=MAX_PLOT_X)
```


gradient descent mushroom
error
1.0
0.8
0.6
0.4
0.2
0.0
0
500
1000
1500
2000
2500
3000
3500
4000
4500
trial


gradient descent mushroom
error
1.0
0.8
0.6
0.4
0.2
0.0
0
500
1000
1500
2000
2500
3000
3500
4000
4500
trial





Still the same.