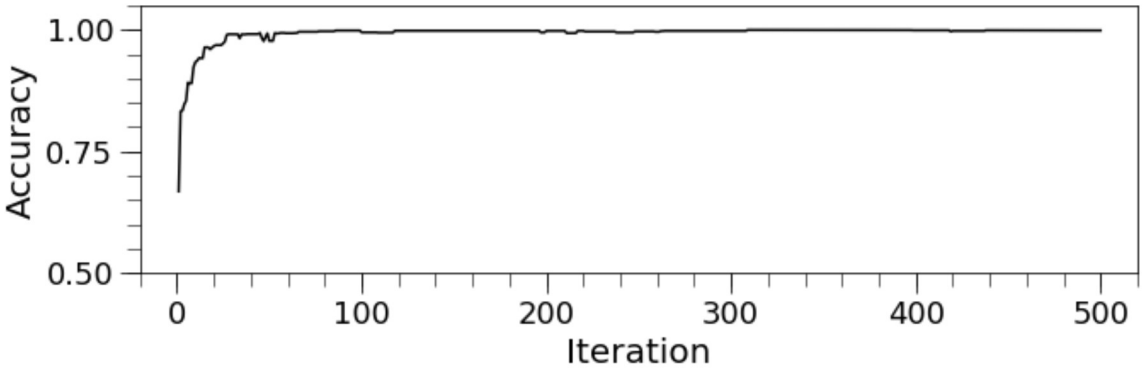
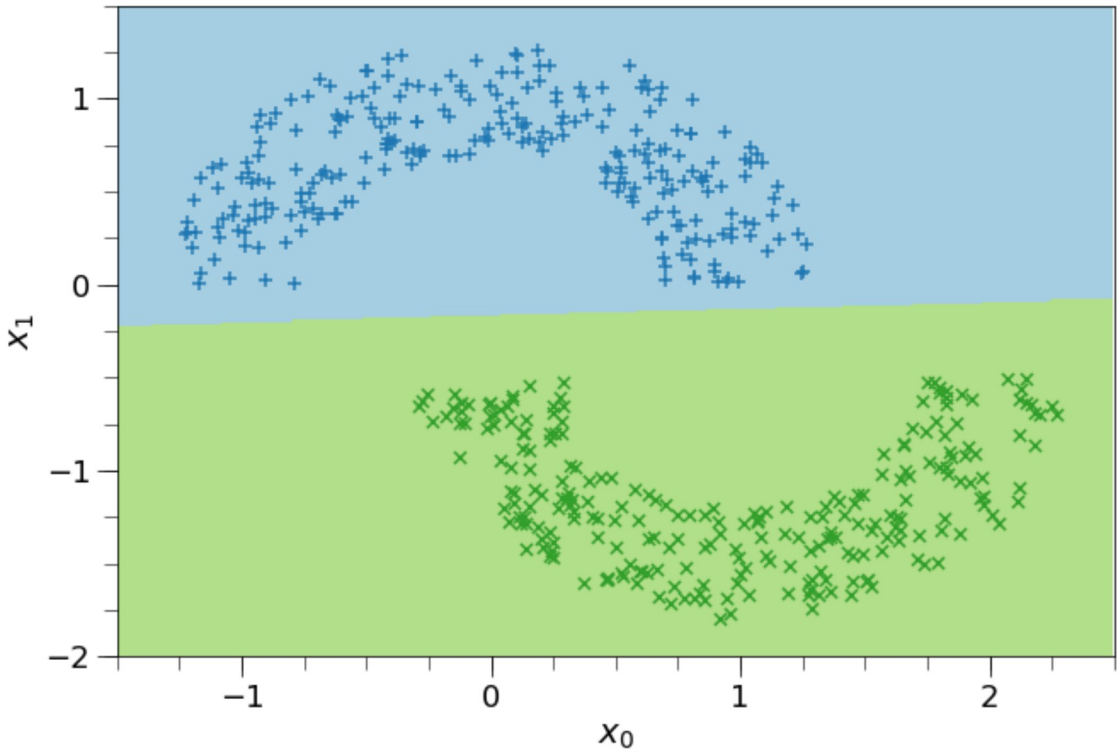
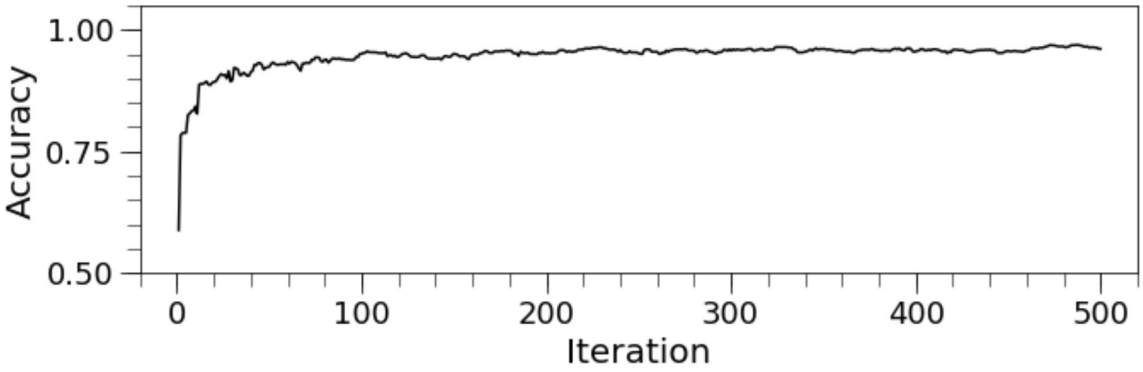
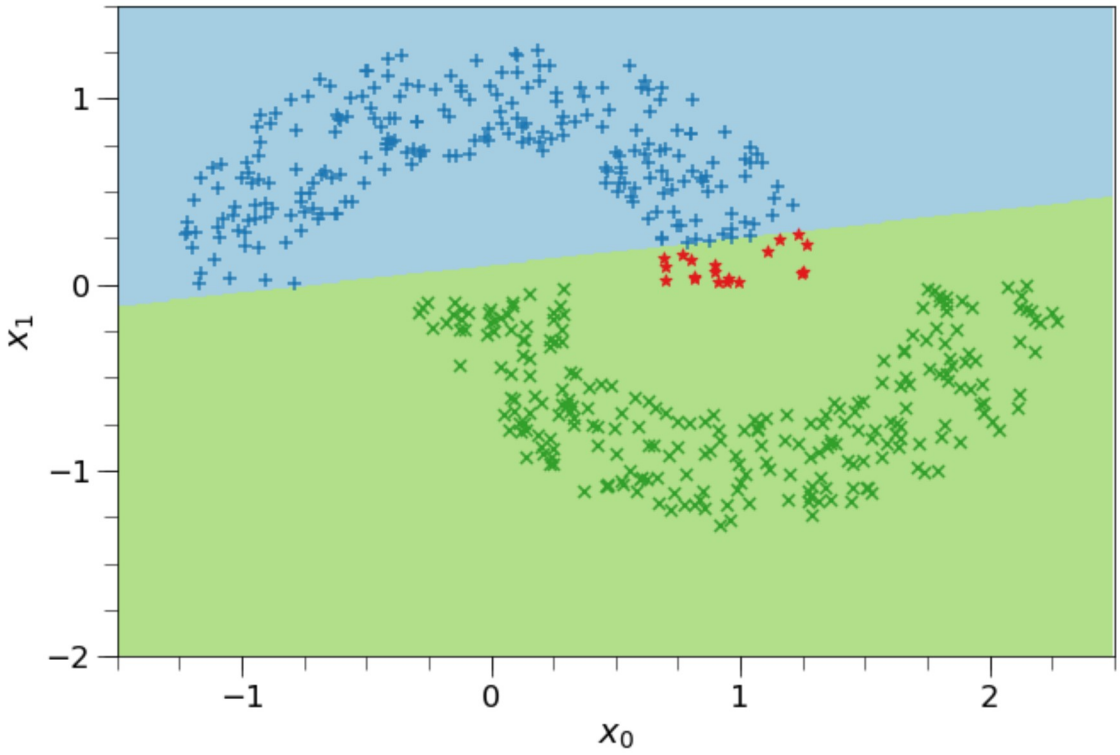
In [5]:
```
# -*- coding: utf-8 -*-
"""
Created on Mon Oct 21 19:17:00 2019

@author: jorge
"""

import numpy as np
from requiredFunctions.train_Perceptron import PerceptronClassifier
from requiredFunctions.doubleMoon import doubleMoon
import matplotlib as mpl
import matplotlib.pyplot as plt
import matplotlib.ticker as mtick

height = 10
width = 10

mpl.rcParams['figure.figsize'] = (width, height)
mpl.rcParams['font.size'] = 20
mpl.rcParams['figure.titlesize'] = 'small'
mpl.rcParams['legend.fontsize'] = 'small'
mpl.rcParams['xtick.major.size'] = 12
mpl.rcParams['xtick.minor.size'] = 8
mpl.rcParams['xtick.labelsize'] = 18
mpl.rcParams['ytick.major.size'] = 12
mpl.rcParams['ytick.minor.size'] = 8
mpl.rcParams['ytick.labelsize'] = 18

N = 500
r = 1
w = 0.6
d_range  = [0.5, 0, -0.5]
trials = 30
```

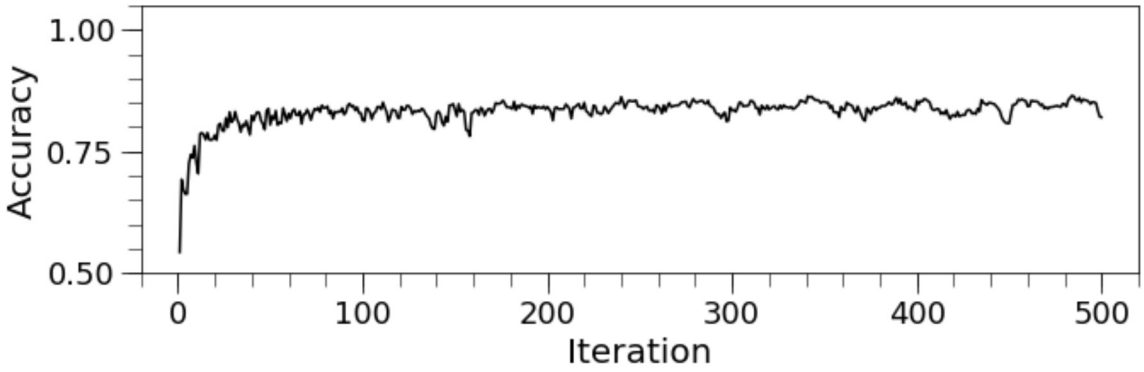
In [6]:
```
for d in d_range:
    trial_acc = np.zeros((trials, N))
    for i in range(trials):
        data = doubleMoon(N, w, r, d, seed=i)
        x_train, y_train = data[:,:2], data[:,2]

        perceptron = PerceptronClassifier()
        perceptron.fit_Online(x_train, y_train, seed=0, max_epochs=1)
        trial_acc[i] = perceptron.accuracy_log

    # Find average iteration error and plot
    acc_avg = trial_acc.mean(axis=0)
    iter_grid = np.arange(1, N+1, 1)

    fig = plt.figure()
    ax = [plt.subplot2grid((3,1), (0,0), colspan=1, rowspan = 1, fig=fig),
          plt.subplot2grid((3,1), (1,0), colspan=1, rowspan = 2, fig=fig)]

    ax[0].errorbar(iter_grid, acc_avg, color='black')
    ax[0].set_xlim(-20, 520)
    ax[0].set_ylim(0.5, 1.05)
    ax[0].xaxis.set_major_locator(mtick.MultipleLocator(100))
    ax[0].xaxis.set_minor_locator(mtick.MultipleLocator(20))
    ax[0].yaxis.set_major_locator(mtick.MultipleLocator(0.25))
    ax[0].yaxis.set_minor_locator(mtick.MultipleLocator(0.05))
    ax[0].set_xlabel('Iteration')
    ax[0].set_ylabel('Accuracy')

    # Predict for the data points and assign indeces of what's right and wrong
    y_pred = perceptron.predict(x_train)
    y_right = np.where(y_pred == y_train)[0]
    y_wrong = np.where(y_pred != y_train)[0]
    blue_ind = y_right[np.where(y_right<250)]
    green_ind = y_right[np.where(y_right>=250)]

    # Make mesh for decision regions and predict for the points within
    x0_min, x0_max = -1.5, 2.5
    x1_min, x1_max = -2, 1.5
    xx0, xx1 = np.meshgrid(np.arange(x0_min, x0_max, 0.01),
                           np.arange(x1_min, x1_max, 0.01))
    cc = perceptron.predict(np.c_[xx0.ravel(), xx1.ravel()]).reshape(xx0.shape)

    # Plotting of decision regions and data points
    cmap = plt.get_cmap('Paired')
    cmap_scatter = mpl.colors.ListedColormap(cmap((1, 3, 5)))
    cmap_contour = mpl.colors.ListedColormap(cmap((0, 2)))
    ax[1].contourf(xx0, xx1, cc, cmap=cmap_contour)
    ax[1].scatter(x_train[:,0][blue_ind], x_train[:,1][blue_ind], 50,
                  c=[cmap_scatter(0)], marker='+')
    ax[1].scatter(x_train[:,0][green_ind], x_train[:,1][green_ind], 50,
                  c=[cmap_scatter(1)], marker='x')
    ax[1].scatter(x_train[:,0][y_wrong], x_train[:,1][y_wrong], 50,
                  c=[cmap_scatter(2)], marker='*')
    ax[1].set_xlim(-1.5, 2.5)
    ax[1].set_ylim(-2, 1.5)
    ax[1].xaxis.set_major_locator(mtick.MultipleLocator(1))
    ax[1].xaxis.set_minor_locator(mtick.MultipleLocator(0.25))
    ax[1].yaxis.set_major_locator(mtick.MultipleLocator(1))
    ax[1].yaxis.set_minor_locator(mtick.MultipleLocator(0.25))
    ax[1].set_xlabel(r'$x_0$')
    ax[1].set_ylabel(r'$x_1$')
    fig.tight_layout(h_pad=0)
    #plt.savefig('../prob2b_' + str(d) + '.eps', dpi=500)
```

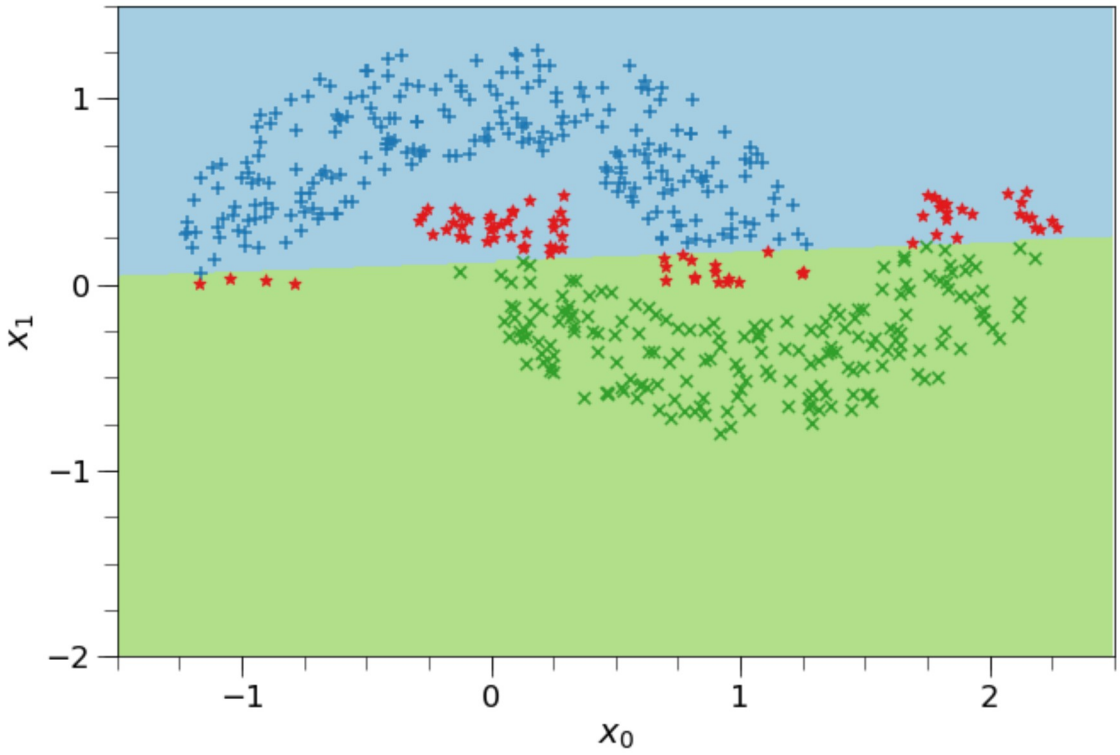

1.00
0.75
0.50
Accuracy
0
100
200
300
400
500
Iteration


1
0
−1
−2
x_1
−1
0
1
2
x_0


1.00
0.75
0.50
Accuracy
0
100
200
300
400
500
Iteration


1
0
−1
−2
x_1
−1
0
1
2
x_0


1.00
0.75
0.50
Accuracy
0
100
200
300
400
500
Iteration


1
0
−1
−2
x_1
−1
0
1
2
x_0