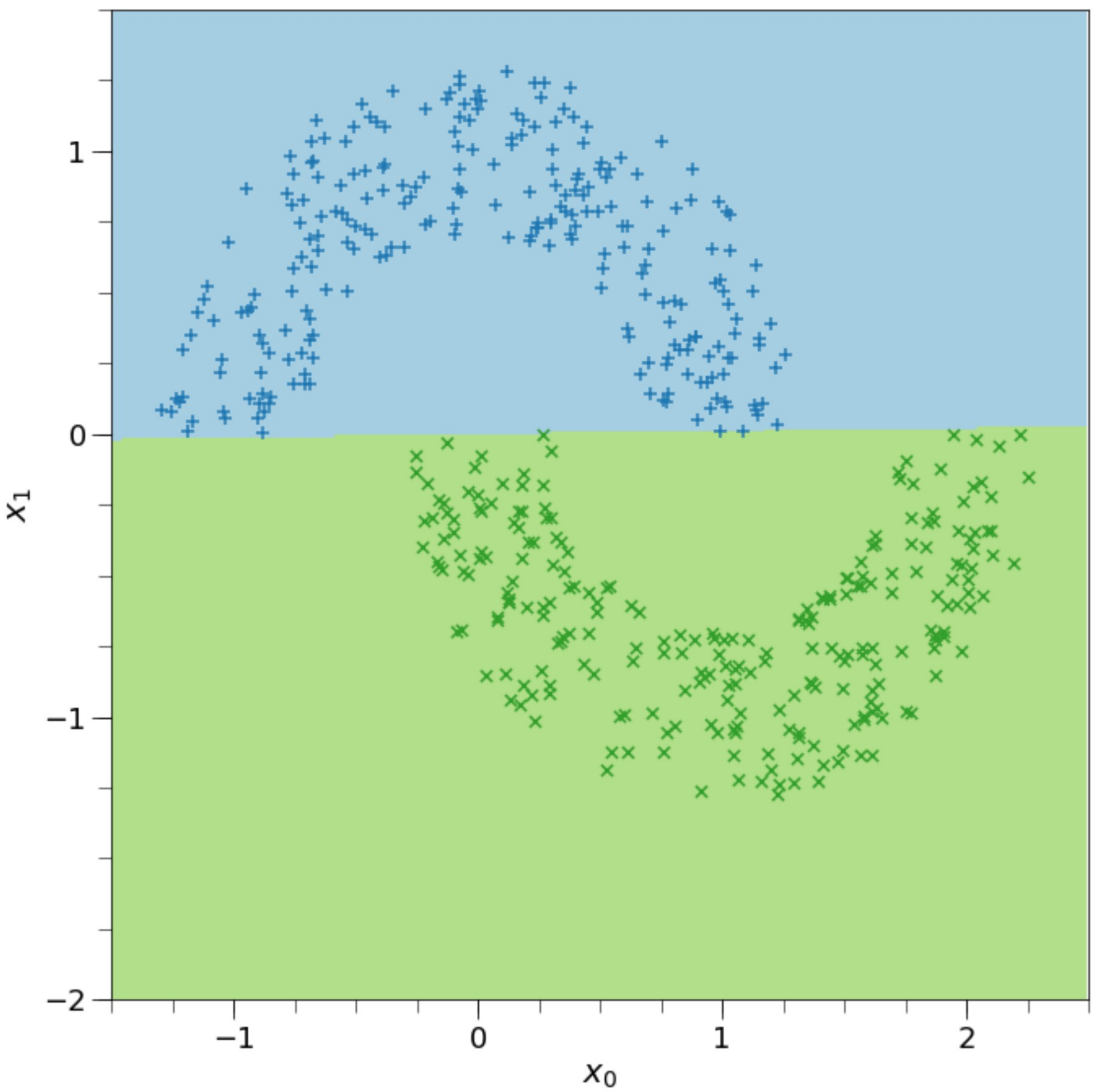
In [5]:

```
# -*- coding: utf-8 -*-
"""
Created on Sat Oct 26 16:47:55 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
max_epochs=100
```

In [6]:

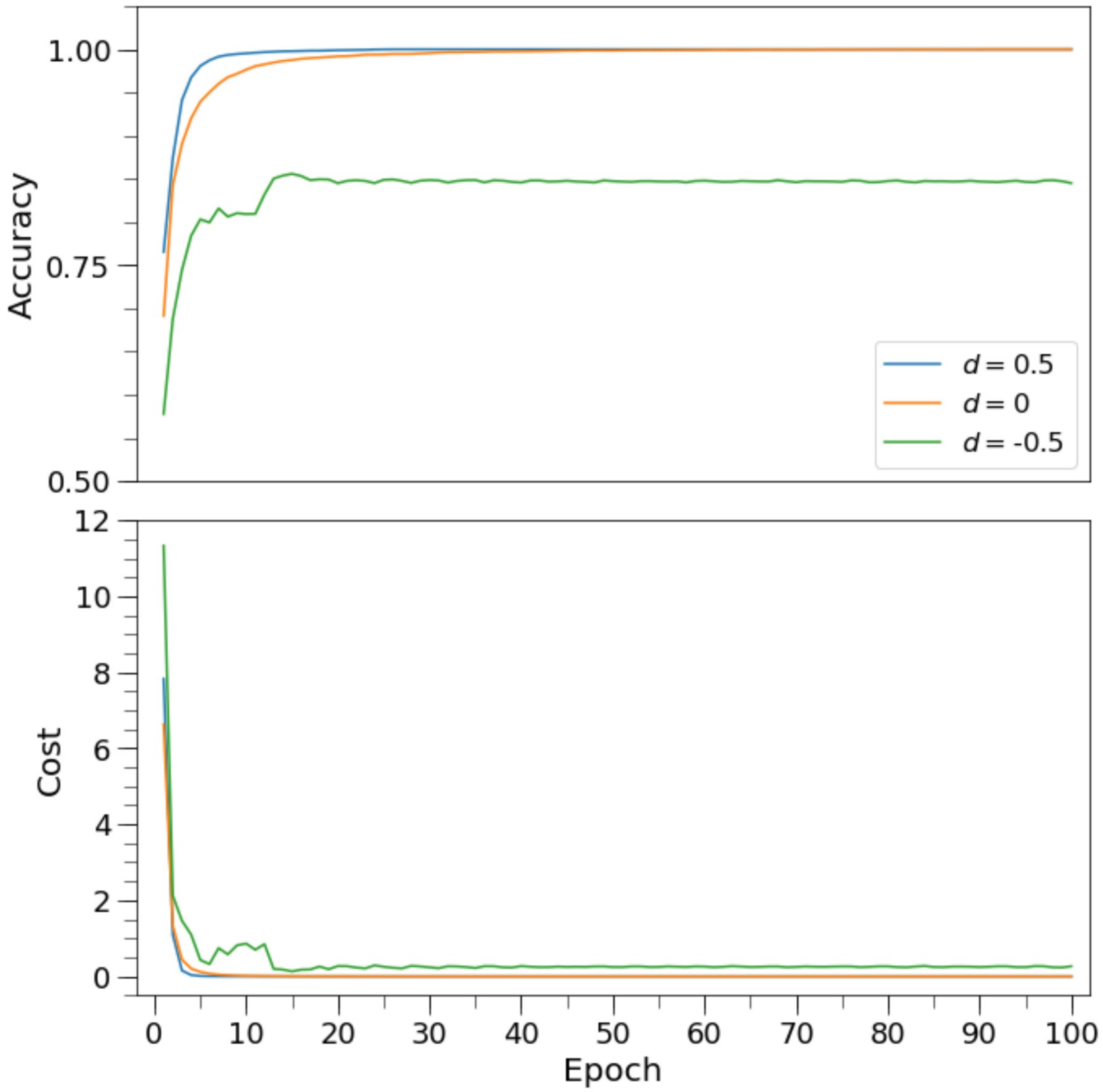
```
acc_per_d = np.zeros((len(d_range),max_epochs))
cost_per_d = np.zeros((len(d_range),max_epochs))
for n, d in enumerate(d_range):
    trial_acc = np.zeros((trials,max_epochs))
    trial_cost = np.zeros((trials,max_epochs))
    for i in range(trials):
        data = doubleMoon(N, w, r, d, seed=0)
        x_train, y_train = data[:,:2], data[:,2]

        perceptron = PerceptronClassifier()
        perceptron.fit_Batch(x_train, y_train, threshold=0, max_epochs=max_epochs)
        trial_acc[i,:] = perceptron.accuracy_log
        trial_cost[i,:] = perceptron.cost_log

    # Find average iteration error and plot
    acc_avg = trial_acc.mean(axis=0)
    cost_avg = trial_cost.mean(axis=0)
    acc_per_d[n] = acc_avg
    cost_per_d[n] = cost_avg
    print('Final Acc:', acc_avg[-1], 'Final Cost:', cost_avg[-1])
    if d == 0:
        # 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)))
        fig, ax = plt.subplots()
        ax.contourf(xx0, xx1, cc, cmap=cmap_contour)
        ax.scatter(x_train[:,0][blue_ind], x_train[:,1][blue_ind], 50,
                   c=[cmap_scatter(0)], marker='+')
        ax.scatter(x_train[:,0][green_ind], x_train[:,1][green_ind], 50,
                   c=[cmap_scatter(1)], marker='x')
        ax.scatter(x_train[:,0][y_wrong], x_train[:,1][y_wrong], 50,
                   c=[cmap_scatter(2)], marker='*')
        ax.set_xlim(-1.5, 2.5)
        ax.set_ylim(-2, 1.5)
        ax.xaxis.set_major_locator(mtick.MultipleLocator(1))
        ax.xaxis.set_minor_locator(mtick.MultipleLocator(0.25))
        ax.yaxis.set_major_locator(mtick.MultipleLocator(1))
        ax.yaxis.set_minor_locator(mtick.MultipleLocator(0.25))
        ax.set_xlabel(r'$x_0$')
        ax.set_ylabel(r'$x_1$')
        fig.tight_layout()
        #plt.savefig('../prob3a.eps', dpi=500)
```

```
Final Acc: 1.0 Final Cost: 0.0
Final Acc: 1.0 Final Cost: 0.0
Final Acc: 0.8450666666666664 Final Cost: 0.2641594616990176
```



In [7]:

```
epoch_grid = np.arange(1, 100+1, 1)

fig, ax = plt.subplots(nrows=2, sharex=True)
for n in range(len(d_range)):
    ax[0].plot(epoch_grid, acc_per_d[n], label=r'$d =$' + str(d_range[n]))
ax[0].set_xlim(-2, 102)
ax[0].set_ylim(0.5, 1.05)
ax[0].tick_params(axis='x', which='both', bottom=False)
ax[0].yaxis.set_major_locator(mtick.MultipleLocator(0.25))
ax[0].yaxis.set_minor_locator(mtick.MultipleLocator(0.05))
ax[0].set_ylabel('Accuracy')
ax[0].legend()

for n in range(len(d_range)):
    ax[1].plot(epoch_grid, cost_per_d[n], label=r'$d =$' + str(d_range[n]))
ax[1].set_xlim(-2, 102)
ax[1].set_ylim(-0.5, np.ceil(cost_per_d.max()))
ax[1].xaxis.set_major_locator(mtick.MultipleLocator(10))
ax[1].xaxis.set_minor_locator(mtick.MultipleLocator(5))
ax[1].yaxis.set_major_locator(mtick.MultipleLocator(2))
ax[1].yaxis.set_minor_locator(mtick.MultipleLocator(0.5))
ax[1].set_xlabel('Epoch')
ax[1].set_ylabel('Cost')
fig.tight_layout(h_pad=0)
#plt.savefig('../prob3bcd.eps', dpi=500)
```


Accuracy
$d = 0.5$
$d = 0$
$d = -0.5$
Cost
Epoch