In [38]:

```
import sys
sys.path.append('C:/Users/jeff8')
datapath = 'C:/Users/jeff8/mlrefined_datasets/nonlinear_superlearn_datasets/'
import autograd.numpy as np
import matplotlib.pyplot as plt
from mlrefined_libraries import multilayer_perceptron_library as multi
```

In [39]:

```
def transform(a,w):

    for W in w:

        a = W[0] + np.dot(a.T , W[1:])
        a = activation(a). T

    return a
```

In [40]:

```
def model(x, theta):

    f = transform(x, theta[0])
    a = theta[1][0] + np.dot(f.T, theta[1][1:])

    return a.T
```

In [41]:

```
def network_initial (layer_sizes, scale):

    weights = []

    for k in range(len(layer_sizes)-1):

        U_k = layer_sizes[k]
        U_k_plus_1 = layer_sizes[k+1]

        weight = scale* np. random. randn(U_k +1, U_k_plus_1)
        weights. append(weight)

    theta_init = [weights[:1], weights[-1]]

    return theta_init
```

In [42]:

```
def bce_loss(output, label):

    y = softmax(outout)

    if label == 0:
        return -np.log(y[0])

    elif label == 1:
        return -np.log(y[1])
```

In [43]:

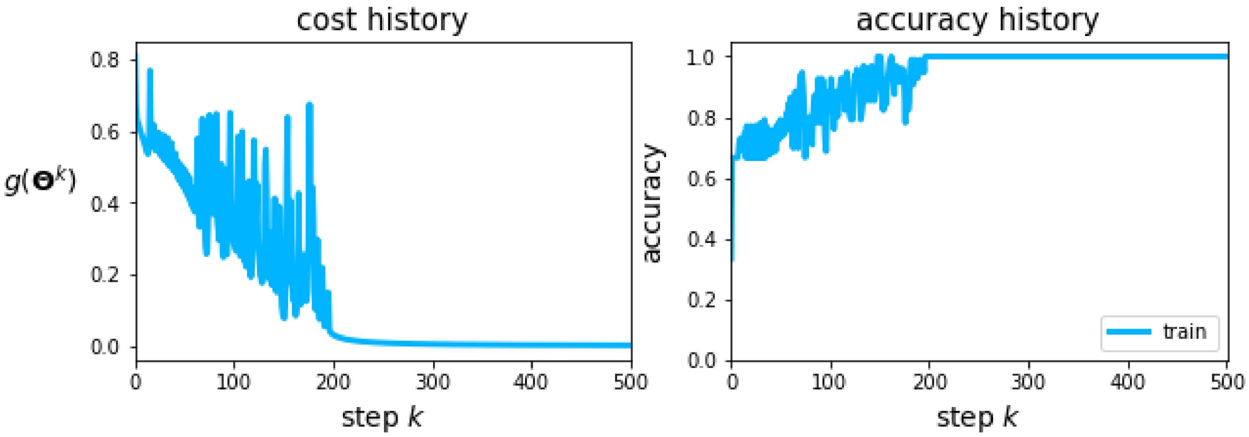
```
data1 = multi.nonlinear_classification_visualizer. Visualizer(datapath + '2_eggs.csv')
x1 = data1.x.T
```

```
y1 = data1.y[np.newaxis,:]
layer_sizes = [2, 10, 10, 10, 1]
w = network_initial([2, 10, 10, 10, 10, 2], 1)

two_class = multi.basic_lib.super_setup.Setup(x1,y1)
two_class.preprocessing_steps(normalizer = 'standard')
two_class.make_train_val_split(train_portion = 1)
two_class.choose_cost(name = 'softmax')

layer_sizes = [10, 10, 10, 10]

two_class.choose_features(features_name = 'softwax', layer_sizes = layer_sizes, activat
two_class.fit(max_its = 500, alpha_choice = 1, verbose = False)
two_class.show_histories()
```

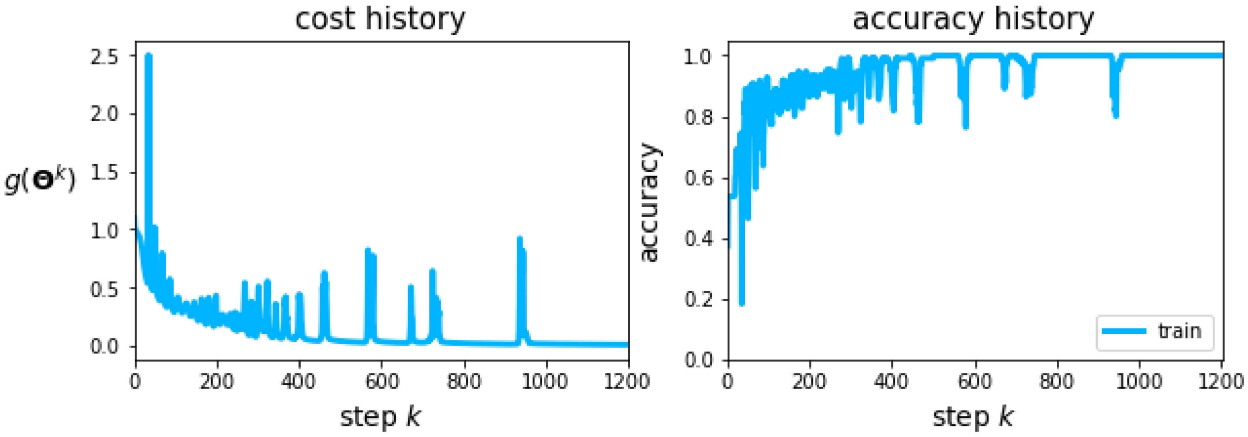


In [46]:

```
data2 = multi.nonlinear_classification_visualizer.Visualizer(datapath + '3_layercake_da
x2 = data2.x.T
y2 = data2.y[np.newaxis,:]

w = network_initial([2, 12, 5, 3], 1)
multi_class = multi.basic_lib.super_setup.Setup(x2,y2)
multi_class.preprocessing_steps(normalizer = 'standard')
multi_class.make_train_val_split(train_portion = 1)
multi_class.choose_cost(name = 'multiclass_softmax')

layer_sizes = [12,5]
multi_class.choose_features(feature_name = 'softmax', layer_sizes = layer_sizes, activa
multi_class.fit(max_its = 1200, alpha_choice = 1, verbose = False)
multi_class.show_histories()
```


cost history
$g(\mathbf{\Theta}^k)$
2.5
2.0
1.5
1.0
0.5
0.0
0
200
400
600
800
1000
1200
step k
accuracy history
accuracy
1.0
0.8
0.6
0.4
0.2
0.0
train
0
200
400
600
800
1000
1200
step k

In [ ]: