In [90]:

```
import numpy as np
from sklearn.datasets import make_blobs
from sklearn.metrics import accuracy_score
!pip install h5py
from utilities import *
from tqdm import tqdm
import matplotlib.pyplot as plt
```

Le package h5py est une interface Pythonic au format de données binaires HDF5. HDF5 vous permet de stocker d'énormes quantités de données numériques et de manipuler facilement ces données à partir de NumPy. Par exemple, vous pouvez découper en ensembles de données de plusieurs téraoctets stockés sur le disque, comme s'il s'agissait de véritables tableaux NumPy.

```
Requirement already satisfied: h5py in /usr/local/anaconda3/lib/python3.9/site-packages (3.2.1)
Requirement already satisfied: numpy>=1.19.0 in /usr/local/anaconda3/lib/python3.9/site-packages (from h5py) (1.20.
3)
```

In [91]:

```
x_train, y_train, x_test, y_test = load_data()
```

In [92]:

```
print(x_train.shape)
print(y_train.shape)
```

```
(1000, 64, 64)
(1000, 1)
```

dataset train: 1000 images de 64 by 64 pixels

dataset train : 1000 labels (un pour chaque image)

In [93]:

```
print(np.unique(y_train, return_counts=True))
```

```
(array([0., 1.]), array([500, 500]))
```

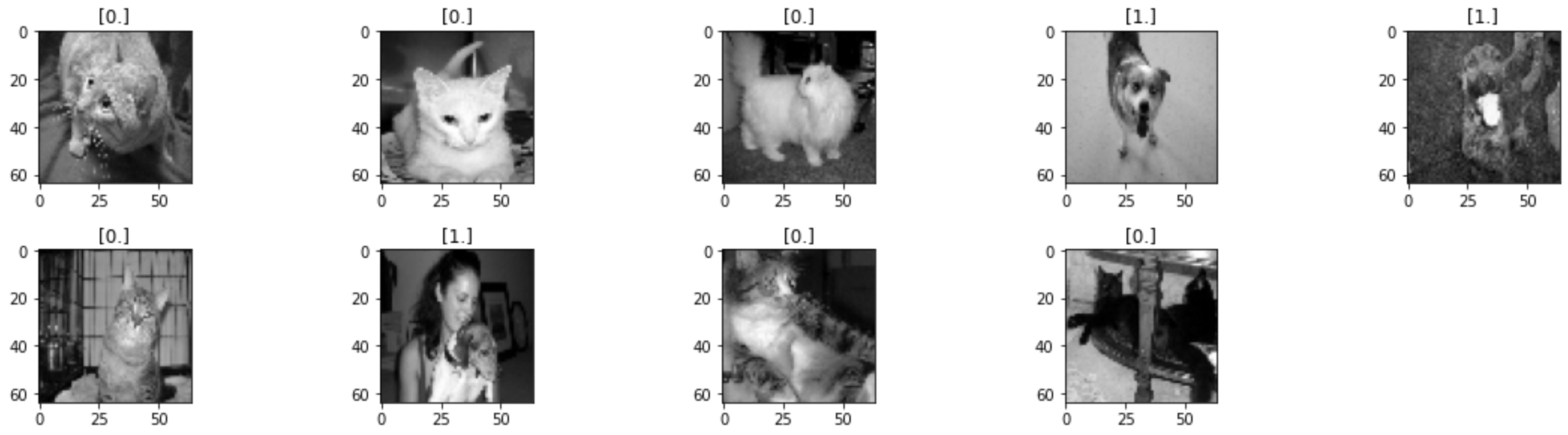
In [94]:

```
print(x_test.shape)
print(y_test.shape)
print(np.unique(y_test,return_counts=True))
```

```
(200, 64, 64)
(200, 1)
(array([0., 1.]), array([100, 100]))
```

In [95]:
```
plt.figure(figsize=(16,8))
for i in range(1,10):
    plt.subplot(4,5,i)
    plt.imshow(x_train[i],cmap='gray')
    plt.title (y_train[i])
    plt.tight_layout()
plt.show()
```

```
/usr/local/anaconda3/lib/python3.9/site-packages/matplotlib/text.py:1215: FutureWarning: elementwise comparison fail
ed; returning scalar instead, but in the future will perform elementwise comparison
  if s != self._text:
```



In [96]:
```
x_train_reshape=x_train.reshape(x_train.shape[0],-1)/x_train.max()
```

In [97]:
```
print(x_train_reshape.shape)
```

```
(1000, 4096)
```

In [98]:
```
x_train_reshape.max()
```

Out[98]:
```
1.0
```

In [99]:
```
x_test_reshape=x_test.reshape(x_test.shape[0],-1)/x_train.max()
```

In [100…
```
print(x_test_reshape.shape)
```

```
(200, 4096)
```

In [101…
```
x_test_reshape.max()
```

Out[101…
```
1.0
```

In [102…
```
print(y.shape)
```

```
(100,)
```

In [103…
```
def initialisation (X):
    W=np.random.randn(X.shape[1],1)
    b=np.random.randn(1)
    return (W, b)
```

In [104…
```
def model(X,W,b):
    Z=X.dot(W)+b
    A=1/(1+np.exp(-Z))
    return A
```

In [105…
```
def logLoss(A,y):
    epsilon=1e-15
    return 1/len(y)*np.sum(-y*np.log(A+epsilon)-(1-y)*np.log(1-A+epsilon))
```

In [106…
```
def gradients (A,X,y):
    dW=1/len(y)*np.dot(X.T,A-y)
    db=1/len(y)*sum(A-y)
    return (dW,db)
```

In [107…

```
def update(dW,db, W, b, learning_rate):
    W=W-learning_rate*dW
    b=b-learning_rate*db
    return (W,b)
```

In [108…

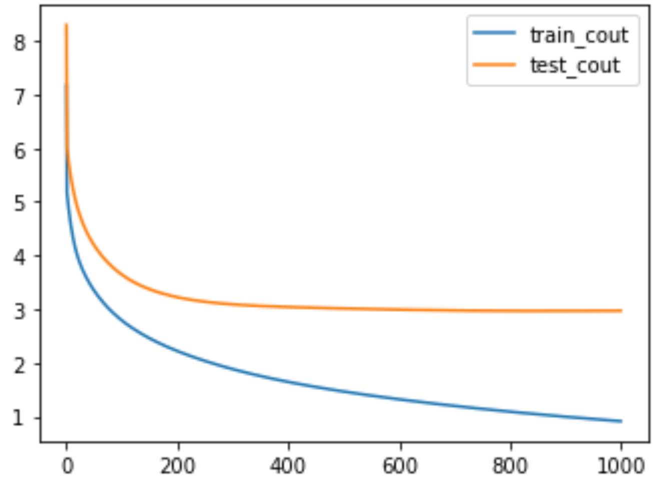
```
def prediction(X,W,b):
    A=model(X,W,b)
    #print(A)
    return A>=0.5
```

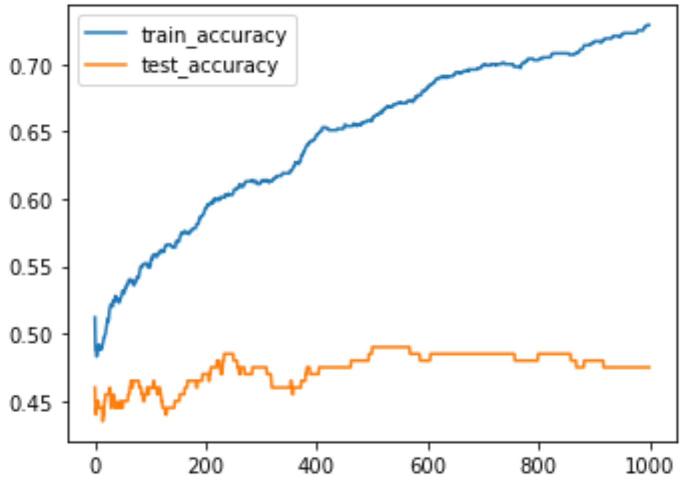
In [119…

```
def artificialNeuron(x_train,y_train,x_test,y_test, learning_rate=0.01, n_iter=10000):
    #initialisation
    W,b=initialisation(x_train)
    train_cout=[]
    train_accuracy=[]
    test_cout=[]
    test_accuracy=[]

    cout=[]
    accuracy=[]
    for i in tqdm(range(n_iter)):
        # activation
        A=model(x_train,W,b)
        if i%10==0:
        #train cout/accuracy
            train_cout.append(logLoss(A,y_train))
            #precision
            y_pred=prediction(x_train,W,b)
            train_accuracy.append(accuracy_score(y_train,y_pred))
        #test cout/accuracy
            A_test=model(x_test, W,b)
            test_cout.append(logLoss(A_test,y_test))
            #precision
            y_pred=prediction(x_test,W,b)
            test_accuracy.append(accuracy_score(y_test,y_pred))

        #mise à jour
        dW,db=gradients(A,x_train,y_train)
        W,b= update(dW,db,W,b,learning_rate)

    plt.figure(figsize=(12,4))
    plt.subplot(1,2,1)
    plt.plot(train_cout, label='train_cout')
    plt.plot(test_cout, label='test_cout')
    plt.legend()
    plt.subplot(1,2,2)
    plt.plot(train_accuracy, label='train_accuracy')
    plt.plot(test_accuracy, label='test_accuracy')
    plt.legend()
    plt.show()
    return (W,b)
```

```
In [120…  W,b=artificialNeuron(x_train_reshape,y_train,x_test_reshape, y_test)
```

```
100%|██████████| 10000/10000 [00:33<00:00, 298.07it/s]
```





```
In [ ]:
```