

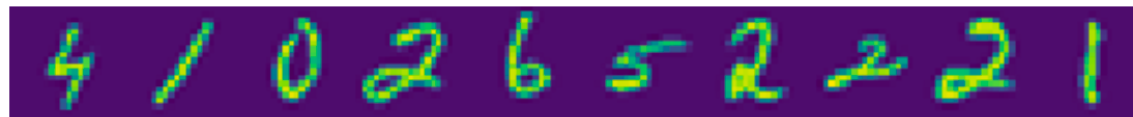
Práctica 3:

regresión logística multiclase y redes neuronales

```
from scipy.io import loadmat
```

```
data = loadmat('ex3data1.mat')  
# se pueden consultar las claves con data.keys()  
y = data['y']  
X = data['X']  
# almacena los datos leídos en X, y
```

```
# Selecciona aleatoriamente 10 ejemplos y los pinta  
sample = np.random.choice(X.shape[0], 10)  
plt.imshow(X[sample, :].reshape(-1, 20).T)  
plt.axis('off')
```



4 1 0 2 6 5 2 2 2 1

```
def ejemplo_error():  
    data = loadmat('ex3data1.mat')  
    y = data['y'] # (5000, 1)  
    X = data['X'] # (5000, 400)  
    m = np.shape(X)[0]  
    X1s = np.hstack([np.ones([m, 1]), X]) # (5000, 401)  
    theta = np.zeros(X1s.shape[1]) # (401, )  
    H = sigmoid(np.matmul(X1s, theta)) # (5000, )  
    error = H - y # (5000, 5000)  
    error = H - np.ravel(y) # (5000)
```

```
def oneVsAll(X, y, n_labels, reg):
```

```
    """
```

```
    oneVsAll entrena varios clasificadores por regresión logística con término  
    de regularización 'reg' y devuelve el resultado en una matriz, donde  
    la fila i-ésima corresponde al clasificador de la etiqueta i-ésima
```

```
    """
```

log(0) : NAN

```
def sigmoid(x):  
    ## cuidado con x > 50 devuelve 1  
    ##  
    s = 1 / (1 + np.exp(-x))  
  
    return s
```

```
def costReg(theta, reg, XX, Y):  
    m = Y.size  
    h = sigmoid(XX.dot(theta))  
  
    ... np.log(1 - h) ...
```

```
def costReg(theta, reg, XX, Y):  
    m = Y.size  
    h = sigmoid(XX.dot(theta))  
  
    ... np.log(1 - h + 1e-6) ...
```