Práctica 4: entrenamiento de redes neuronales

one_hot y

```
data = loadmat('ex4data1.mat')
y = data['y'].ravel() # (5000, 1) --> (5000,)
X = data['X']

m = len(y)
input_size = X.shape[1]
num_labels = 10

y = (y - 1)
y_onehot = np.zeros((m, num_labels))  # 5000 x 10

for i in range(m):
    y_onehot[i][y[i]] = 1
```

Back prop

```
m = X.shape[0]
...

a1, z2, a2, z3, h = forward_propagate(X, theta1, theta2)
...

for t in range(m):
    a1t = a1[t, :]  # (1, 401)
    a2t = a2[t, :]  # (1, 26)
    ht = h[t, :]  # (1, 10)
    yt = y[t]  # (1, 10)

    d3t = ht - yt  # (1, 10)
    d2t = np.dot(theta2.T, d3t) * (a2t * (1 - a2t))  # (1, 26)

    delta1 = delta1 + np.dot(d2t[1:, np.newaxis], a1t[np.newaxis, :])
    delta2 = delta2 + np.dot(d3t[:, np.newaxis], a2t[np.newaxis, :])
```

Coste

```
# compute the cost
J = 0
for i in range(m):
    J += np.sum(-y[i] * np.log(h[i]) - (1 - y[i]) * np.log(1 - h[i]))
J = J / m

# add the cost regularization term
J += (reg / (2 * m)) * (np.sum(theta1[:, 1:] ** 2) +
                        np.sum(theta2[:, 1:] ** 2)   )
```