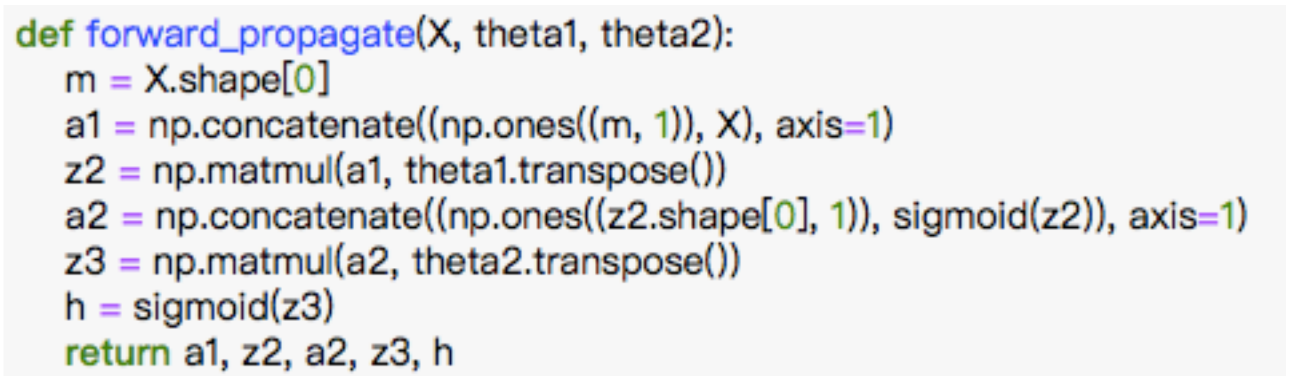
MLHW4

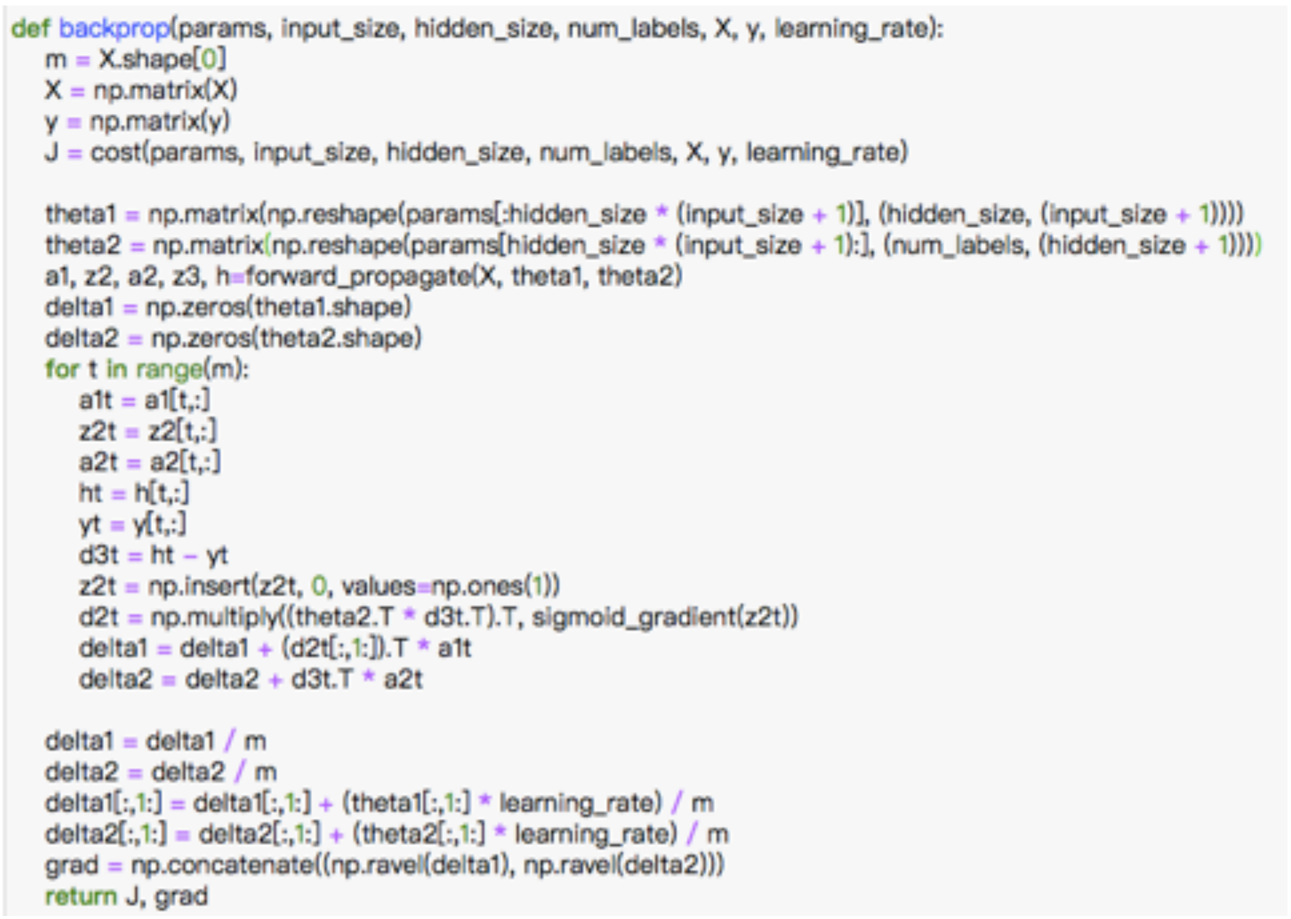
0516218 軒轅照雯

1. Screenshot of Forward-propagate code



```
def forward_propagate(X, theta1, theta2):
    m = X.shape[0]
    a1 = np.concatenate((np.ones((m, 1)), X), axis=1)
    z2 = np.matmul(a1, theta1.transpose())
    a2 = np.concatenate((np.ones((z2.shape[0], 1)), sigmoid(z2)), axis=1)
    z3 = np.matmul(a2, theta2.transpose())
    h = sigmoid(z3)
    return a1, z2, a2, z3, h
```

2. Screenshot of Back-propagate code



```
def backprop(params, input_size, hidden_size, num_labels, X, y, learning_rate):
    m = X.shape[0]
    X = np.matrix(X)
    y = np.matrix(y)
    J = cost(params, input_size, hidden_size, num_labels, X, y, learning_rate)

    theta1 = np.matrix(np.reshape(params[:hidden_size * (input_size + 1)], (hidden_size, (input_size + 1))))
    theta2 = np.matrix(np.reshape(params[hidden_size * (input_size + 1):], (num_labels, (hidden_size + 1))))
    a1, z2, a2, z3, h=forward_propagate(X, theta1, theta2)
    delta1 = np.zeros(theta1.shape)
    delta2 = np.zeros(theta2.shape)
    for t in range(m):
        a1t = a1[t,:]
        z2t = z2[t,:]
        a2t = a2[t,:]
        ht = h[t,:]
        yt = y[t,:]
        d3t = ht - yt
        z2t = np.insert(z2t, 0, values=np.ones(1))
        d2t = np.multiply((theta2.T * d3t.T).T, sigmoid_gradient(z2t))
        delta1 = delta1 + (d2t[:,1:]).T * a1t
        delta2 = delta2 + d3t.T * a2t

    delta1 = delta1 / m
    delta2 = delta2 / m
    delta1[:,1:] = delta1[:,1:] + (theta1[:,1:] * learning_rate) / m
    delta2[:,1:] = delta2[:,1:] + (theta2[:,1:] * learning_rate) / m
    grad = np.concatenate((np.ravel(delta1), np.ravel(delta2)))
    return J, grad
```

3. accuracy

```
accuracy = 97.72%
```