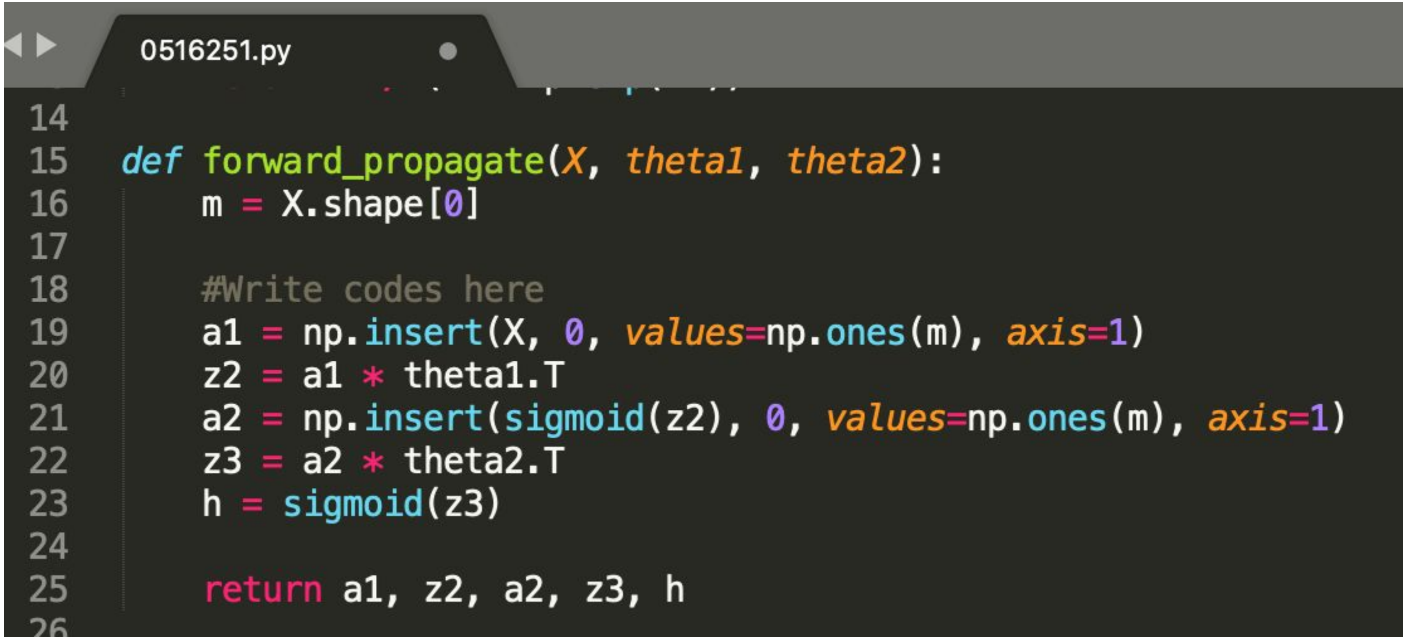
Machine Learning

Project 4

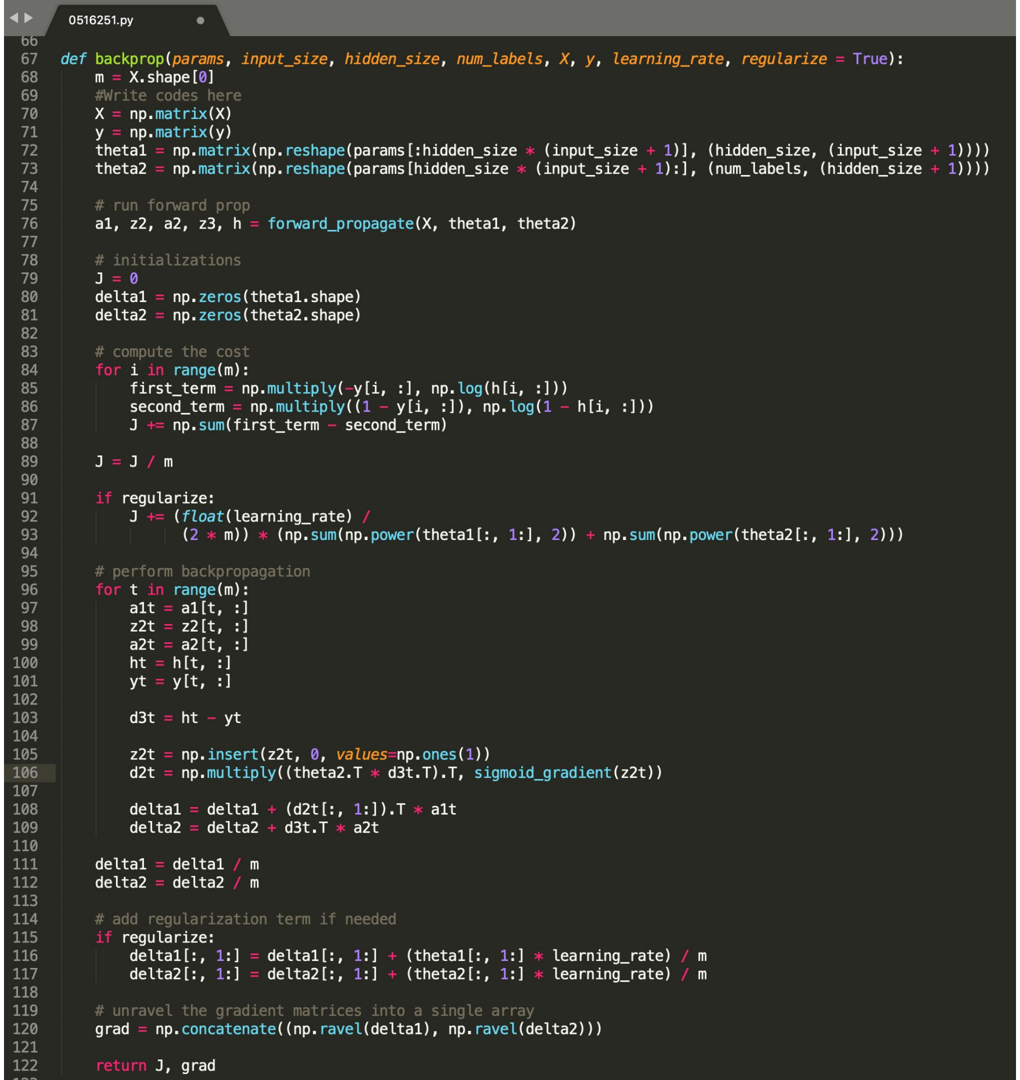
費蓋德 0516251

Forward-propagate:



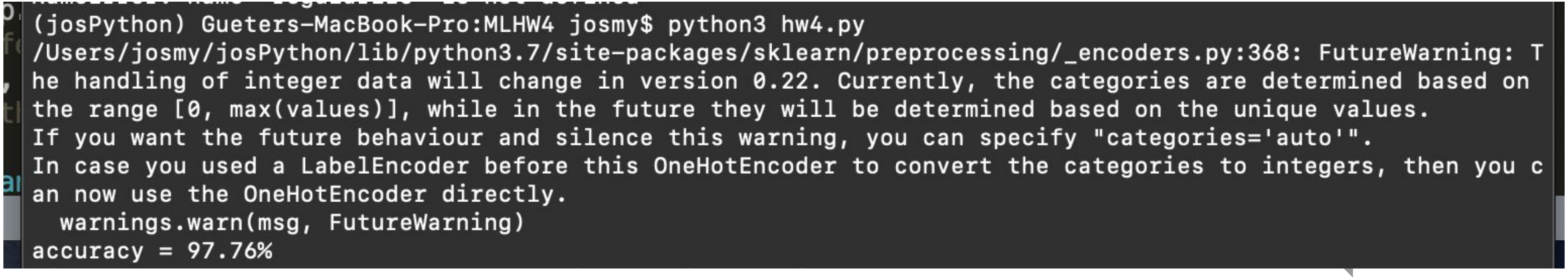
```python

def forward_propagate(X, theta1, theta2):
    m = X.shape[0]

    #Write codes here
    a1 = np.insert(X, 0, values=np.ones(m), axis=1)
    z2 = a1 * theta1.T
    a2 = np.insert(sigmoid(z2), 0, values=np.ones(m), axis=1)
    z3 = a2 * theta2.T
    h = sigmoid(z3)

    return a1, z2, a2, z3, h
```

Back-propagate:



```
def backprop(params, input_size, hidden_size, num_labels, X, y, learning_rate, regularize = True):
    m = X.shape[0]
    #Write codes here
    X = np.matrix(X)
    y = np.matrix(y)
    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))))

    # run forward prop
    a1, z2, a2, z3, h = forward_propagate(X, theta1, theta2)

    # initializations
    J = 0
    delta1 = np.zeros(theta1.shape)
    delta2 = np.zeros(theta2.shape)

    # compute the cost
    for i in range(m):
        first_term = np.multiply(-y[i, :], np.log(h[i, :]))
        second_term = np.multiply((1 - y[i, :]), np.log(1 - h[i, :]))
        J += np.sum(first_term - second_term)

    J = J / m

    if regularize:
        J += (float(learning_rate) /
              (2 * m)) * (np.sum(np.power(theta1[:, 1:], 2)) + np.sum(np.power(theta2[:, 1:], 2)))

    # perform backpropagation
    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

    # add regularization term if needed
    if regularize:
        delta1[:, 1:] = delta1[:, 1:] + (theta1[:, 1:] * learning_rate) / m
        delta2[:, 1:] = delta2[:, 1:] + (theta2[:, 1:] * learning_rate) / m

    # unravel the gradient matrices into a single array
    grad = np.concatenate((np.ravel(delta1), np.ravel(delta2)))

    return J, grad
```

Accuracy



```
(josPython) Gueters-MacBook-Pro:MLHW4 josmy$ python3 hw4.py
/Users/josmy/josPython/lib/python3.7/site-packages/sklearn/preprocessing/_encoders.py:368: FutureWarning: T
he handling of integer data will change in version 0.22. Currently, the categories are determined based on
the range [0, max(values)], while in the future they will be determined based on the unique values.
If you want the future behaviour and silence this warning, you can specify "categories='auto'".
In case you used a LabelEncoder before this OneHotEncoder to convert the categories to integers, then you c
an now use the OneHotEncoder directly.
  warnings.warn(msg, FutureWarning)
accuracy = 97.76%
```