**گزارش پروژه سوم (فصل پنجم) دوره آموزش شبکه عصبی و یادگیری عمیق**

**ایجاد شبکه عصبی قدم به قدم**

**(ملیحه حسن زاده)**

**مرحله صفر: اضافه کردن کتابخانه های مورد نیاز**

فایل های موجود در پوشه Requirements را در مسیر پروژه قرار گرفت

**مرحله اول : مقدار دهی اولیه**

**توضیح تمرین اول:**

در تابع initialize\_parameters مقداردهی اولیه پارامترهای w و b به شکل زیر انجام شد:

W1 = np.random.randn(n\_h, n\_x) \* 0.01

b1 = np.zeros((n\_h, 1))

W2 = np.random.randn(n\_y, n\_h) \* 0.01

b2 = np.zeros((n\_y, 1))

در تابع initialize\_parameters\_deep هم به شکل زیر:

for l in range(1, L):

parameters['W' + str(l)] = np.random.randn(layer\_dims[l], layer\_dims[l - 1]) \* 0.01

parameters['b' + str(l)] = np.zeros((layer\_dims[l], 1))

**مرحله دوم: انتشار رو به جلو**

در تابع linear\_forward بخش خطی (Z) به شکل زیر محاسبه شد:

Z = np.dot(W , A) + b

در تابع linear\_activation\_forward مقادیر فعالساز (ReLU و Sigmoid) به شکل زیر پیاده سازی شدند:

if activation == "sigmoid":

Z, linear\_cache = linear\_forward(A\_prev, W, b)

A, activation\_cache = sigmoid(Z)

elif activation == "relu":

Z, linear\_cache = linear\_forward(A\_prev, W, b)

A, activation\_cache = relu(Z)

و در نهایت در تابع L\_model\_forward پیاده سازی forward propagation برای L لایه به شکل زیر انجام شد:

for l in range(1, L):

A\_prev = A

A, cache = linear\_activation\_forward(A\_prev, parameters['W' + str(l)], parameters['b' + str(l)] , "relu")

caches.append(cache)

AL, cache = linear\_activation\_forward(A , parameters['W' + str(L)], parameters['b' + str(L)] , "sigmoid")

caches.append(cache)

مرحله سوم: محاسبه خطا

تابع هزینه بر اساس فرمول آن، به صورت زیر پیاده سازی شد:

cost = (-1 / m) \* np.sum(np.multiply(Y, np.log(AL)) + np.multiply((1 - Y) , np.log(1 - AL)))

مرحله چهارم: پس انتشار

در تابع linear\_backward مشتقات dW، db و dA\_prev بر اساس فرمول ها به صورت زیر پیاده سازی شدند:

dW = (1 / m) \* np.dot(dZ, A\_prev.T)

db = (1 / m) \* np.sum(dZ, axis = 1, keepdims = True)

dA\_prev = np.dot(W.T, dZ)

در تابع linear\_activation\_backward مقادیر dZ و سپس dA\_prev, dW, db برای هر دو تابع فعالساز relu و sigmoid به صورت زیر پیاده سازی شدند:

if activation == "relu":

dZ = relu\_backward(dA, activation\_cache)

dA\_prev, dW, db = linear\_backward(dZ, linear\_cache)

elif activation == "sigmoid":

dZ = sigmoid\_backward(dA, activation\_cache)

dA\_prev, dW, db = linear\_backward(dZ, linear\_cache)

در نهایت تابع L\_model\_backward، نیز به شکل زیر تکمیل شد:

dAL = - (np.divide(Y, AL) - np.divide(1 - Y, 1 - AL))

current\_cache = caches[L - 1]

grads["dA" + str(L-1)], grads["dW" + str(L)], grads["db" + str(L)] = linear\_activation\_backward(dAL, current\_cache, "sigmoid")

for l in reversed(range(L-1)):

current\_cache = caches[l]

dA\_prev\_temp, dW\_temp, db\_temp = linear\_activation\_backward (grads["dA" + str(l + 1)], current\_cache, "relu")

grads["dA" + str(l)] = dA\_prev\_temp

grads["dW" + str(l + 1)] = dW\_temp

grads["db" + str(l + 1)] = db\_temp

مرحله پنجم: به روز رسانی وزن ها

در تابع update\_parameters پارامترهای W و b به شکل زیر آپدیت شدند:

for l in range(L):

parameters["W" + str(l+1)] = parameters["W" + str(l+1)] - np.multiply(learning\_rate, grads["dW" + str(l + 1)])

parameters["b" + str(l+1)] = parameters["b" + str(l+1)] - np.multiply(learning\_rate, grads["db" + str(l + 1)])