**گزارش پروژه دوم (فصل چهارم) دوره آموزش شبکه عصبی و یادگیری عمیق**

**پیاده سازی شبکه عصبی با یک لایه مخفی**

**(ملیحه حسن زاده)**

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

فایل های موجود در پوشه Requirements در مسیر پروژه قرار گرفت

**مرحله اول : بارگذاری مجموعه دیتا**

**توضیح تمرین اول:**

با دستور shape ابعاد X و Y و تعداد نمونه های آموزشی را به دست می آوریم. به صورت زیر:

shape\_X = X.shape

shape\_Y = Y.shape

m = X.shape[1]

**مرحله دوم: مشاهده عملکرد لجستیک رگرسیون روی مجموعه داده ها**

**تفسیر دسته بندی رگرسیون لجستیک، آیا دسته بندی به درستی انجام شده است؟**

اجرای کد clf.fit(X.T, Y.T) با خطای زیر مواجه بود:

DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n\_samples, ), for example using ravel().

y = column\_or\_1d(y, warn=True)

که با کد زیر اصلاح گردید:

clf.fit(X.T, Y.ravel());

**تفسیر دسته بندی رگرسیون لجستیک:**

با توجه به نتیجه plot\_decision\_boundary برای دسته بندی نمونه ها و همچنین دقت 47% مشخص می شود که در اینجا مدل رگرسیون لجستیک به خوبی عمل نکرده و دسته بندی را با خطای تقریبا بالایی انجام داده که در بیشتر نقاط، کلاس گل موردنظر اشتباه تخمین زده شده و در نتیجه این مدل برای هدف ما مناسب نیست.

**مرحله سوم : استفاده از شبکه های عصبی**

**توضیح تمرین دوم:**

با دستور shape و آرگومان های ورودی X و Y ، اندازه های n\_x، n\_h و n\_yبه دست آمد و ساختار کلی شبکه عصبی تعریف شد. به صورت زیر:

n\_x = X.shape[0]

n\_h = 4

n\_y = Y.shape[0]

**توضیح تمرین سوم:**

به کمک توابع random.randn و zeros مقداردهی اولیه ماتریس وزن ها و بایاس با توجه به ابعادشان انجام شد. به شکل زیر:

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))

**توضیح تمرین چهارم:**

با استفاده از توابع np.dot، np.tanh و sigmoid بردارهای Z1، A1، Z2 و A2 را پیاده سازی کردیم و سپس با استفاده از تابع assert صحت ابعاد بردار خروجی (A2) را بررسی کردیم. در نهایت بردارهای Z1، A1، Z2 و A2 را در یک دیکشنری با نام cache ذخیره کردیم.

W1 = parameters["W1"]

b1 = parameters["b1"]

W2 = parameters["W2"]

b2 = parameters["b2"]

Z1 = np.dot(W1, X) + b1

A1 = np.tanh(Z1)

Z2 = np.dot(W2, A1) + b2

A2 = sigmoid(Z2)

assert(A2.shape == (1, X.shape[1]))

cache = {"Z1": Z1,

"A1": A1,

"Z2": Z2,

"A2": A2}

**بررسی تابع هزینه و کاربرد تابع squeeze**

هزینه cross-entropy به صورت زیر محاسبه شده است:

logprobs = np.multiply(np.log(A2), Y) + np.multiply(np.log(1 - A2), 1 - Y)

cost = (-1./ m)\* np.sum(logprobs)

سپس تابع squeeze ابعاد اضافی را حذف کرده و یک مقدار واحد (a scalar value) که همان مقدار محاسبه شده تابع هزینه هست را برمیگرداند.

cost = np.squeeze(cost)

و در نهایت با تابع assert و isinstance چک می شود که مقدار cost از نوع float باشد:

assert(isinstance(cost, float))

**توضیح تمرین پنجم:**

در تابع backward\_propagetion ، مقادیر موردنیاز بر اساس فرمول ها به شکل زیر پیاده سازی شدند:

dZ2 = A2 - Y

dW2 = (1 / m) \* np.dot(dZ2, A1.T)

db2 = (1 / m) \* np.sum(dZ2, axis = 1, keepdims = True)

dZ1 = np.dot(W2.T, dZ2) \* (1 - np.power(A1, 2))

dW1 = (1 / m) \* np.dot(dZ1, X.T)

db1 = (1 / m) \* np.sum(dZ1, axis = 1, keepdims = True)

**توضیح تمرین ششم:**

در تابع update\_parameters، آپدیت پارامترهای شبکه عصبی به شکل زیر پیاده سازی شدند:

W1 = W1 - learning\_rate \* dW1

b1 = b1 - learning\_rate \* db1

W2 = W2 - learning\_rate \* dW2

b2 = b2 - learning\_rate \* db2

**توضیح تمرین هفتم:**

تابع nn\_modelرا با استفاده از اجزایی که در مراحل قبل پیاده سازی کردیم، تکمیل کردیم:

parameters = initialize\_parameters(n\_x, n\_h, n\_y)

W1 = parameters["W1"]

b1 = parameters["b1"]

W2 = parameters["W2"]

b2 = parameters["b2"]

for i in range(0, num\_iterations):

A2, cache = forward\_propagation(X, parameters)

cost = compute\_cost(A2, Y, parameters)

grads = backward\_propagation(parameters, cache, X, Y)

parameters = update\_parameters(parameters, grads, learning\_rate = 1.2)

if print\_cost and i % 1000 == 0:

print ("Cost after iteration %i: %f" %(i, cost))

**توضیح تمرین هشتم:**

تابع predict طبق حد آستانه پیش بینی کلاس ها یعنی 0.5، به صورت زیر پیاده شد:

A2, cache = forward\_propagation(X, parameters)

predictions = (A2 > 0.5).astype(int)

**مرحله چهارم : آموزش شبکه عصبی**

در نهایت مدل با تابع nn\_model که از قبل نوشتیم، آموزش داده شد و پلات مرز تصمیم آن با تابع plot\_decision\_boundary رسم شد و مشاهده شد که در مقایسه با مدل رگرسیون لجستیک که در ابتدای پروژه (مرحله دوم) آن را بررسی کردیم دسته بندی خیلی بهتری انجام شده و به خوبی دو کلاس خروجی ما (گل ها) از هم تفکیک شدند.

همچنین می بینیم که هزینه در هر 1000 ایپوک کم و کمتر می شود

سپس با محاسبه دقت مشخص شد که دقت پیش بینی مدل از 47% (در مدل رگرسیون لجستیک) به 90% (در شبکه عصبی با 1 لایه پنهان) افزایش داشته است و این نشان دهنده برتری شبکه عصبی با 1 لایه پنهان نسبت به رگرسیون لجستیک می باشد.