**گزارش پروژه اول دوره آموزش شبکه عصبی و یادگیری عمیق**

**رگرسیون لجستیک**

**(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 آرایه های train\_set\_x\_orig و test\_set\_x\_orig گرفته شد و در نتیجه متغیرهای خواسته شده به صورت زیر در کد مقداردهی شد:

m\_train = 209

m\_test = 50

num\_px = 64

**توضیح تمرین دوم:**

در این تمرین برای تغییر ابعاد ماتریس های تصاویرtrain و test از اندازه (3، 64، 64) به (1، 12288) از تابع reshape استفاده شد. در نهایت برای پیاده سازی آسان تر در استفاده در مدل از عملگر ترانهاده استفاده شد. برای داده های train به شکل زیر:

train\_set\_x\_flatten = train\_set\_x\_orig.reshape(m\_train, num\_px\*num\_px\*3).T

در نتیجه ابعاد train\_set\_x\_flatten به صورت (209، 12288) شد.

**مرحله دوم: معماری کلی و آموزش شبکه عصبی**

**توضیح تمرین سوم**

در تابع sigmoid برای محاسبه مقدار سیگموئید آرگومان ورودی، دستور زیر نوشته شد:

s = 1 / (1 + np.exp(-x))

**توضیح تمرین چهارم**

برای مقدار دهی وزن ها و بایاس را با مقدار صفر، از آنجایی که w یک بردار و b یک عدد ثابت هست به شکل زیر مقداردهی اولیه شدند:

w = np.zeros((dim, 1))

b = 0

**توضیح تمرین پنجم**

در تابع propagate در انتشار رو به جلو (Forward Propagation) طبق فرمول هایی که داشتیم، کد تابع فعالساز و تابع هزینه قرار داده شد. در انتشار رو به عقب (Backward Propagation) هم طبق فرمول ها کد dw و db قرار داده شد.

Forward Propagation:

A = sigmoid(np.dot(w.T, X) + b)

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

Backward Propagation:

dw = 1 / m \* np.dot(X, (A - Y).T)

db = 1 / m \* np.sum(A - Y)

**توضیح تمرین ششم**

در این قسمت داخل حلقه for ، برای محاسبه مقادیر grads و cost تابع propagate را که در تمرین پنجم پیاده سازی کردیم، فراخوانی میکنیم:

grads, cost = propagate(w, b, X, Y)

همچنین برای به روزرسانی آرایه وزن ها (w) و و مقدار ثابت بایاس (b) نیز از فرمولی که داشتیم استفاده میکنیم. (مقادیر dw و db از grades گرفته شدند.)

w = w - learning\_rate \* dw

b = b - learning\_rate \* db

**توضیح تمرین هفتم**

در تابع predict برای محاسبه پیش بینی احتمالات خروجی مدل (یعنی A) از تابع سیگموئید و فرمول آن استفاده میکنیم:

A = sigmoid(np.dot(w.T, X) + b)

سپس برای تبدیل احتمالات به مقادیر پیش بینی واقعی خروجی مدل (گربه هست = 1، گربه نیست = 0) و با توجه به حد آستانه تصمیم گیری، داخل حلقه for این خط کد را مینویسیم:

Y\_prediction[0][i] = 1 if A[0][i] > 0.5 else 0

**توضیح تمرین هشتم**

در این مرحله اجزای شبکه عصبی را که تا اینجا پیاده سازی کردیم، در تابع model در کنار هم قرار می دهیم:

w, b = initialize\_with\_zeros(num\_px \* num\_px \* 3)

parameters, grads, costs = optimize(w, b, X\_train, Y\_train, num\_iterations, learning\_rate, print\_cost)

Y\_prediction\_test = predict(w, b, X\_test)

Y\_prediction\_train = predict(w, b, X\_train)

در گام بعد، مدل را با مجموعه دیتای train وtest صدا زدیم. در نتیجه به ازای هر 100 تکرار مقدار تابع هزینه را در هر propagate نمایش داد و در نهایت دقت را به شکل زیر خروجی داد:

train accuracy: 99.04306220095694 %

test accuracy: 70.0 %

در کد قسمت بعد، با مشخص کردن index های خاص از مجموعه دیتای test میتوانستیم تصویر هر نمونه و درستی یا نادرستی پیش بینی مدلمان را ببینیم. به طور مثال با index = 1 تصویر یک گربه به همراه پیغام زیر نمایش داده شد که این یعنی پیش بینی مدل برای این نمونه درست بوده.

y = 1, you predicted that it is a "cat" picture.

توضیح: کد این قسمت یک اشکال ریز داشت: اینکه در دستور print باید نوع مقدار d["Y\_prediction\_test"][0,index] را از float به int تغییر می دادیم. یعنی:

int(d["Y\_prediction\_test"][0,index]

با اجرای کد قسمت آخر هم (یعنی Plot learning curve (with costs)) ، منحنی یادگیری با مقادیر هزینه در هر 100 تکرار نمایش داده شد که مشخص می کند که در هر 100 تکرار propagate مقدار تابع هزینه در حال کاهش و رو به صفر هست و در نتیجه مقادیر w و b و نیز پیش بینی خروجی مسئله به سمت بهینه تر شدن هستند.