# پروژه پنجم – شروع کار با Tensorflow

هدف این پروژه آشنایی با کتابخانه تنسورفلو و کار با توابع مختلف این کتابخانه است.

## مرحله صفر: اضافه کردن کتابخانه های لازم

محتویات پوشه Requirements را در مسیر پروژه اضافه کنید.

- نحوه پیاده سازی فرمول زیر را به کمک توابع تنسور فلو مشاهده کنید.

$$loss = \mathcal{L}(\hat{y}, y) = (\hat{y}^{(i)} - y^{(i)})^2$$

- توابع ضرب و session ها را در تنسور فلو تمرین کنید. (کد های آماده را مطالعه و بررسی کنید و امکانات آن ها را با تغییر پارامترها مشاهده کنید.)
- نحوه کار با placeholder را تمرین کنید. (کد آماده را مطالعه و بررسی کنید و امکانات آن ها را با تغییر پارامترها مشاهده کنید.)

## مرحله اول: پیاده سازی معادله خطی

تا اینجا تعدادی از توابع و عملکرد های تنسور فلو را آموختید.

**تمرین اول: تابع خطی $Y=WX+b$ را با آموخته های خود پیاده سازی کنید.**
(مقدار W را با ابعاد (۴,۳) و مقدار X را با ابعاد (۳,۱) و مقدار b را در ابعاد (۴,۱) تعریف کنید. برای اینکار از تابع constant تنسور فلو استفاده کنید و مقادیر آنها را از تابع np.random.randn() به صورت اولیه انتخاب کنید. سپس مقدار آن را با ایجاد یک session محاسبه کنید.)

## مرحله دوم: پیاده سازی تابع سیگموید

**تمرین دوم: تابع سیگموید را با توابع کتابخانه تنسورفلو پیاده سازی کنید.**
(برای اینکار ابتدا یک placeholder ایجاد کنید و پس از استفاده از تابع سیگموید تنسورفلو، session مربوطه را ایجاد و اجرا کنید. در مورد کتابخانه feed_dict مطالعه کنید و از آن کمک بگیرید.)

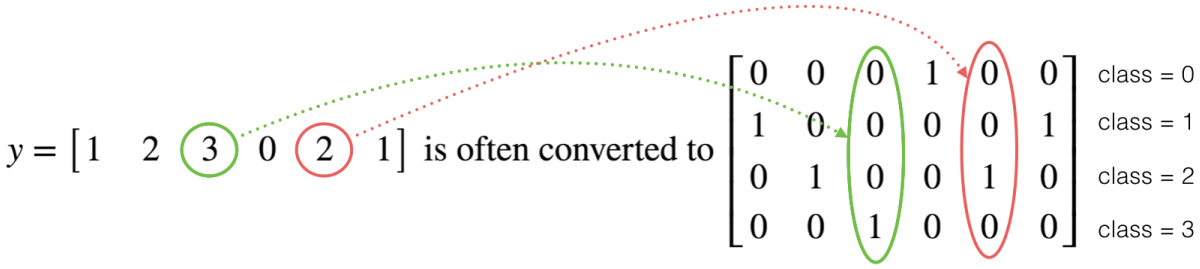
## مرحله سوم: پیاده سازی تابع هزینه

**تمرین سوم: با توجه به آنچه تا کنون آموخته اید، و با کمک تابع tf.nn.sigmoid_cross_entropy_with_logits تابع هزینه را طبق رابطه زیر پیاده سازی کنید.**

$$-\frac{1}{m}\sum_{i=1}^{m}\left(y^{(i)}\log\sigma(z^{[2](i)})+(1-y^{(i)})\log(1-\sigma(z^{[2](i)}))\right)$$

## مرحله چهارم: رمزگذاری One-Hot Encoding

در یادگیری عمیق بارها احتیاج دارید که شماره کلاسهای خروجی یک شبکه عصبی را به صورت one-hot encoding پیاده سازی کنید. این کد گذاری به صورت زیر تبدیل بردارها را به ماتریس انجام میدهد:




توجه کنید که مقادیر بردار اولیه نشان دهنده اندیس مقدار ۱ در ماتریس نهایی هستند.

**تمرین چهارم: از تابع one_hot از کتابخانه تنسورفلو کمک بگیرید و رمزگذاری one-hot encoding را پیاده سازی کنید.**

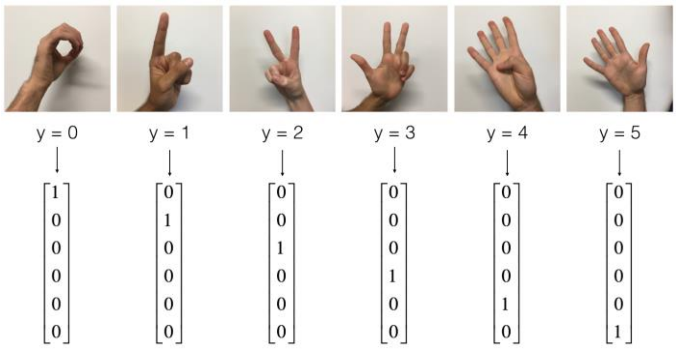
**تمرین پنجم: تابع ones() را برای مقدار دهی یک برای پارامترها پیاده سازی کنید. فقط از کتابخانه تنسورفلو استفاده کنید.**

## مرحله پنجم: ساخت اولین شبکه عصبی با استفاده از تنسورفلو

**تمرین ششم: پیاده سازی شبکه عصبی را به صورت مرحله به مرحله با استفاده از تنسورفلو و توضیحات زیر انجام دهید.**

- **معرفی دیتاست**

در این مرحله داده های SIGN دسته بندی میشوند. این دیتاست شامل تصاویر ۶۴*۶۴ شامل علامت های دست از صفر تا ۵ است. برای آموزش از ۱۰۸۰ داده و برای تست از ۱۲۰ داده استفاده میشود.




- **ایجاد placeholder**

برای مقادیر X و Y، که مقادیر ویژگی ها و برچسب داده ها هستند، placeholder ایجاد کنید.

- **مقداردهی اولیه پارامترها**

مقادیر وزن ها را با استفاده از ابعاد زیر و توابع get_variable و xavier_initializer و مقدار seed=1 مقدار دهی کنید. مقادیر بایاس های شبکه را با صفر مقدار دهی کنید. (فقط از تنسورفلو استفاده کنید.)

  - W1 : [25, 12288]
  - b1 : [25, 1]
  - W2 : [12, 25]
  - b2 : [12, 1]
  - W3 : [6, 12]
  - b3 : [6, 1]

- **انتشار رو به جلو (Forward Propagation)**

شبکه را با ساختار زیر ایجاد کنید و مراحل انتشار رو به جلو را پیاده سازی کنید.

LINEAR -> RELU -> LINEAR -> RELU -> LINEAR -> SOFTMAX

- **محاسبه هزینه**

محاسبه هزینه را همانند آنچه پیش از این پیاده سازی کردید انجام دهید. برای بهینه سازی آن از تابع زیر کمک بگیرید.

tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits(….)

- **پس انتشار (Back Propagation)**

با کمک گرفتن از فریم ورک هایی نظیر تنسورفلو، عملیات پس انتشار در شبکه های عصبی به راحتی پیاده سازی میشود، با کمک گرفتن از توابع زیر عملیات پس انتشار را پیاده سازی کنید.

```
optimizer = tf.train.GradientDescentOptimizer(learning_rate = learning_rate).minimize(cost)

_ , c = sess.run([optimizer, cost], feed_dict={X: minibatch_X, Y: minibatch_Y})
```

- **مدلسازی نهایی و آموزش مدل**

در نهایت اجزای مختلف شبکه که تا اینجا پیاده سازی کردید در کنار هم قرار دهید، شبکه را کامل کرده و آن را آموزش دهید.