جلسه‌ی ۷

شبکه های عصبی

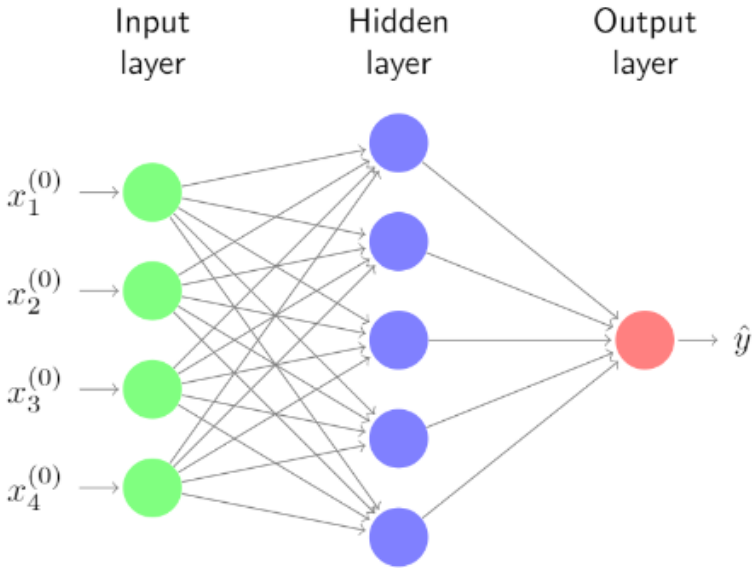
اهداف این جلسه

شما در این جلسه یاد خواهید گرفت که :

- تصویرِ بهتری از شبکه‌های عصبی در ذهن خود ایجاد کنید.
- بر روی یک شبکه‌ی عصبی ساده، الگوریتمِ پس‌انتشار [۱] و پیش‌رو [۲] را پیاده‌سازی کنید.

در مسائلی که در ادامه مطرح می‌شوند، ما از یک شبکه عصبی بسیار ساده استفاده خواهیم کرد. این شبکه، همانطور که در شکل یک نشان داده شده است، دارای یک لایه‌ی پنهان [۳] با اندازه‌ی $K = 5$ به همراه لایه‌ی ورودی با اندازه $D = 4$ و $L = 1$ و لایه‌ی خروجی با سایز یک، می‌باشد

[۱] Backpropagation
[۲] Feed-Forward
[۳] Hidden layer



شکل ۱.۷: یک شبکه عصبیِ ساده

تمرین اول

در شبکه عصبیِ ساده‌سازی شده‌ی ما، ما یک تابعِ feed-forward مانند زیر، داریم:

$$x_j^{(1)} = \phi\left(z_j^{(1)}\right) = \phi\left(\sum_{i=1}^{D} w_{i,j}^{(1)} x_i^{(0)} + b_j^{(1)}\right) \tag{۱.۷}$$

،

$$\hat{y} = \phi\left(z_1^{(2)}\right) = \phi\left(\sum_{i=1}^{K} w_{i,1}^{(2)} x_i^{(1)} + b_1^{(2)}\right) \tag{۲.۷}$$

از رابطه‌ی ۱ و ۲ استفاده کنید و تابع متناظر در فایل ژوپیتر را تکمیل کنید. برای سادگی، در مسائلِ پیش‌رو، مقدار بایاس را برابر با صفر در نظر بگیرید و از `Sigmoid` به عنوان تابع فعال‌ساز [۱] استفاده کنید.

تمرین دوم

فرض کنید که ما از مربعات خطا به عنوان تابع هزینه خود استفاده می‌کنیم:

$$\mathcal{L} = \frac{1}{2}(\hat{y} - y)^2 \tag{۳.۷}$$

که در مثال ما، فقط یک نمونه وجود دارد و y نیز مقدار واقعی است و $\hat{y}$ هم مقدار پیش‌بینی‌شده توسط شبکه عصبی می‌باشد.

با توجه به وزن‌های $w_{i,1}^{(1)}$ و $w_{i,1}^{(2)}$ ، مشتقِ $\mathcal{L}(\boldsymbol{L})$ را بررسی کنید و تابع متناظر آن را در فایل ژوپیتر، پیاده‌سازی نمایید.

[۱] Function Activation

جلسه‌ی ۸

منظم سازی

اهداف این جلسه

مدلهای یادگیری عمیق، ظرفیت و انعطاف زیادی بر روی انواع مجموعه داده‌ها دارند، اما یک مشکل فراگیر، مسئله‌ی **بیش‌برازش** [۱] است. این مشکل یک مشکل بسیار جدی هنگام آزمونِ مدل‌های آموزش داده شده است. اگر مجموعه آموزشی، به اندازه کافی بزرگ نباشد، نتایج روی داده‌های آموزشی خوب خواهد بود اما شبکه‌ی آموزش داده شده، بر روی داده های جدیدی که تا کنون آن را ندیده است، اصلا خوب عمل نخواهد کرد. در این جلسه قصد داریم با استفاده از منظم‌سازی [۲] ، این مشکل را برطرف سازیم.

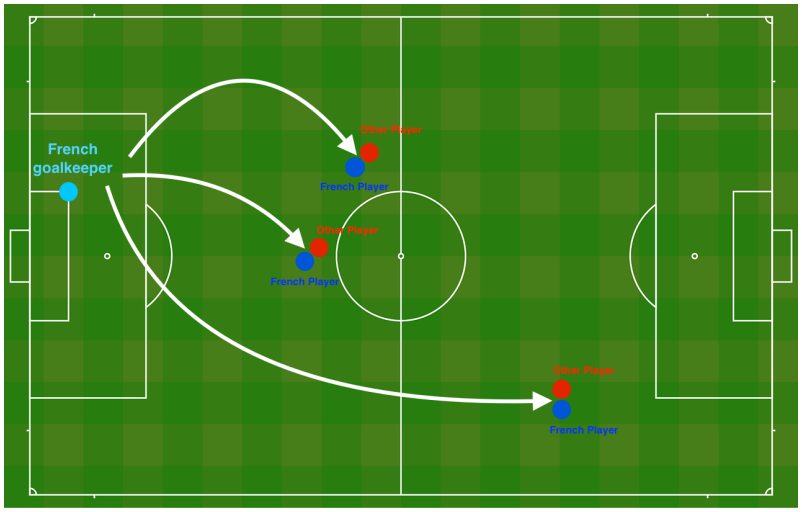
[۱] Overfitting
[۲] Regularization

۱.۸ پکیج‌ها

در فایل ژوپیتر این جلسه، در قسمت پکیج‌ها، کد مربوط به فراخوانی کتابخانه‌های آماده و کتابخانه‌های حاوی توابعی که شما به آن‌ها نیاز پیدا خواهید کرد، نوشته شده است.

۲.۸ شرح مسئله

فرض کنید شما تحت عنوان متخصص هوش مصنوعی در تیم فوتبالِ فرانسه انتخاب شده اید. وظیفه شما پیشنهاد موقعیت‌هایی است که دروازه‌بان باید توپ را به سمت آن‌ها بفرستد تا بازیکنان بتوانند به توپ با سر، ضربه بزنند.



شکل ۱.۸ : زمین فوتبال

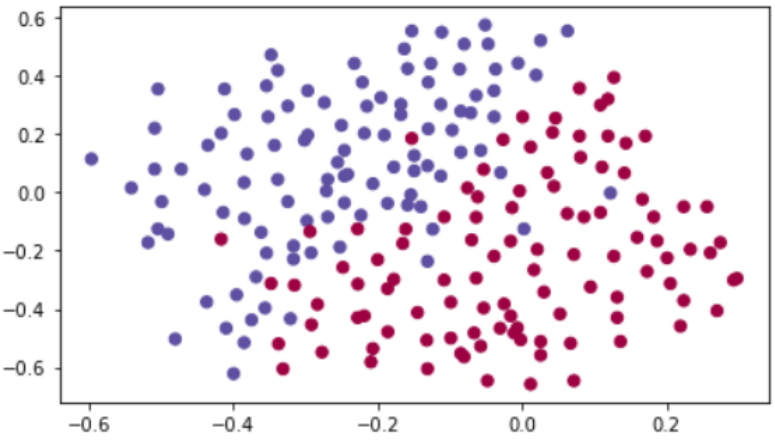
مجموعه داده شما شامل یک صفحه دو بُعدی از ده بازی قبلی است. که در بخش بعدی آن را مشاهده خواهید کرد.

۳.۸ بارگذاری مجموعه داده

در این بخش، قطعه کد مربوط به بارگذاری داده‌ها با استفاده از یکی از توابع آماده، موجود است. با اجرای آن، مجموعه داده را بصورت مصور، همانند شکلِ (۲) مشاهده خواهید کرد.

در نقطه در این شکل، متناظر مکانی از زمین فوتبال است که یک بازیکن، به توپ در آن نقطه، ضربه زده است.

- اگر نقطه آبی باشد، یعنی بازیکن فرانسوی با سر به توپ ضربه زده است
- اگر نقطه قرمز باشد، یعنی بازیکن تیم روبه‌رو به توپ با سر ضربه زده است.



شکل ۲.۸: تصویر مجموعه داده

هدف شما ، استفاده از یک مدل یادگیری عمیق، برای یافتن موقعیت‌هایی در زمین است که دروازه‌بان توپ را باید به آن موقعیت‌ها پرتاب کند.

بررسی مجموعه داده: همانطور که دیدید، این مجموعه داده کمی نویزی است اما به نظر می رسد یک خط مورب برای جدا کردن نیمه بالا سمت چپ (آبی) از نیمه راست پایین (قرمز)، به خوبی کار می کند.
شما در این جلسه در ابتدا، یک مدل منظم‌سازی-نشده را امتحان خواهید کرد. سپس یاد خواهید گرفت که آن مدل را منظم‌سازی کنید و سپس تصمیم خواهید گرفت که از کدام مدل قرار است استفاده کنید.

۴.۸ مدلِ منظم‌سازی‌نشده

در فایل ژوپیتر این جلسه، در بخش چهارم، شما از شبکه عصبی‌ای که از قبل برای شما پیاده‌سازی شده است، استفاده خواهید کرد. این مدل می‌تواند به طریق زیر، استفاده شود:

- در حالتِ *منظم‌سازی*: با مقداردهیِ غیر صفر به متغیرِ `lambd` این کار صورت خواهد گرفت. ما از `lambd` بجای `lambda` استفاده می‌کنیم زیرا `lambda` یک کلمه‌ی رزرو شده در پایتون است.
- در حالتِ *dropout* : با مقداردهیِ `keep_prob` به مقداری کوچکتر از یک.

شما در ابتدا از مدلی که منظم‌سازی ندارد، استفاده خواهید کرد، سپس موارد ذیل را پیاده‌سازی خواهید نمود:

- L2 regularization : توابعِ `compute_cost_with_regularization()` و `backward_propagation_with_regularization()`

- Dropout: توابعِ

`forward_propagation_with_dropout()`

و `backward_propagation_with_dropout()`

در هر قسمت ، این مدل را با ورودی های صحیح اجرا می کنید تا توابعی را که پیاده کرده اید فراخوانی کند. برای آشنایی بیشتر با مدل به کدی که در سلول اولِ بخش چهارمِ فایل ژوپیتر این جلسه نوشته شده است، دقت کنید.
پس از اجرای سلول اول، شما در سلول دوم، مدل خود را بدون منظم‌سازی، آموزش خواهید داد و دقت آن را بر روی مجموعه‌های آموزشی و آزمون، خواهید دید.
دقتِ آموزش در این حالت، ۹۴/۸% و دقت آزمون، ۹۱/۵% است. این مدل، مدلِ پایه‌ی ماست و شما تاثیر منظم‌سازی را بر روی این مدل در بخشهای آتی، مشاهده خواهید نمود.
در سلول سوم این بخش، در فایل ژوپیتر، کد مربوط به مصوّرسازیِ مرز تصمیمِ مدل، آمده است، آن را اجرا کنید و تصویر را ببینید.
همانطور که می‌بینید، مدلِ منظم‌سازی‌نشده، به وضوح، بر روی مجموعه آموزشی، بیش‌برازش شده است و نقاط نویز را برازش کرده است. حال برای کاهش بیش‌برازش، دو تکنیک را بررسی می‌کنیم.

۵.۸ منظم‌سازیِ L2

راهِ حل استاندارد برای جلوگیری از بیش‌برازش، **منظم‌سازیِ L2** نامیده می‌شود. تمرکز این روش بر روی تغییرِ مناسبِ تابع هزینه از حالتِ

$$J = -\frac{1}{m}\sum_{i=1}^{m}\left(y^{(i)}\log\left(a^{[L](i)}\right) + \left(1-y^{(i)}\right)\log\left(1-a^{[L](i)}\right)\right) \tag{۱.۸}$$

به حالتِ

$$J_{\text{regularized}} = \underbrace{-\frac{1}{m}\sum_{i=1}^{m}\left(y^{(i)}\log\left(a^{[L](i)}\right) + \left(1-y^{(i)}\right)\log\left(1-a^{[L](i)}\right)\right)}_{\text{cost cross-entropy}} + \underbrace{\frac{1}{m}\frac{\lambda}{2}\sum_{l}\sum_{k}\sum_{j}W_{k,j}^{[l]2}}_{\text{cost regularization L۲}} \tag{۲.۸}$$

است. حال بیایید تابع هزینه را تغییر دهیم و عواقب آن را مشاهده کنیم.

۱.۵.۸ تمرین اول: compute_cost_with_regularization

تابعِ `compute_cost_with_regularization()` را پیاده‌سازی کنید. این تابع، هزینه را با استفاده از عبارتِ (۲) محاسبه خواهد کرد. برای محاسبه‌ی $\sum_k \sum_j W_{k,j}^{[l]2}$ از :

```
np.sum(np.square(W1))
```

استفاده کنید. نکته آن که شما باید این کار را برای $W^{[1]}$ ، $W^{[2]}$ و $W^{[3]}$ انجام دهید و سپس سه عبارت را جمع کرده و در $\frac{1}{m}\frac{\lambda}{2}$ ضرب کنید.

پر واضح است که بدلیل اینکه هزینه را عوض کردید، باید انتشارِ پس‌رو [1] را نیز تغییر دهید. تمام گرادیان‌ها باید با توجه به هزینه جدید، محاسبه شوند.

۲.۵.۸ تمرین دوم: backward_propagation_with_regularization

تغییرات لازم را در مرحله‌ی انتشارِ پس‌رو برای مناسب شدن آن برای منظم‌سازی، اعمال کنید. تغییرات باید فقط dW۱ ، dW۲ و dW۳ را تحت‌الشعاع قرار دهد. برای هرکدام، لازم است که گرادیانِ عبارتِ مربوط به منظم‌سازی ($\frac{d}{dW}\left(\frac{1}{2}\frac{\lambda}{m}W^2\right) = \frac{\lambda}{m}W$) را اضافه کنید.

در سلول سوم کدهای این تمرین، مدل را با استفاده از منظم‌سازیِ L2 ($\lambda = 0.7$) اجرا خواهید کرد. تابعِ `model()` ،

`compute_cost_with_regulariztion` را بجای `compute_cost` و

`backard_propagation_with_regularization` را بجای `backward_propagation` اجرا خواهد کرد.

پس از اجرا، دقت شما به ۹۳% افزایش پیدا خواهد کرد. در این حالت شما مجموعه آموزشی را بیش‌برازش نمی‌کنید. سپس در سلول چهارم کدهای این قسمت، مرزهای تصمیم را مصورسازی خواهید کرد. همانطور که مشاهده خواهید نمود :

[1] backward-propagation

- مقدارِ λ ابَرپارامتری است که شما می‌توانید با استفاده از یک مجموعه dev ، آن را تنظیم کنید.
- منظم‌سازیِ L2 مرزهای تصمیم شما را نَرم‌تر می‌کند.اگر λ زیاد بزرگ نباشد، ممکن است باعث "زیاد نرم شدن" شود که در مدلهای با بایاس زیاد، اتفاق می‌افتد.

منظم‌سازیِ L2 چطور کار می‌کند؟

منظم‌سازیِ L2 بر این فرض استوار است که مدل شما با وزن‌های کوچکتر، نسبت به مدل با وزن‌های بزرگتر، مدل ساده‌تری است. بنابراین ، با مجازات مقادیر مربع وزنها در تابع هزینه ، همه وزنها را به سمت گرفتن مقادیر کوچک‌تر، هدایت می‌کنید. در این حالت برای تابع هزینه، داشتن وزن‌های بزرگ، بسیار پرخرج است. این کار شما را به سمت یک مدلِ نَرم تر، که خروجی نسبت به تغییرات ورودی آرام‌تر تغییر می‌یابد، هدایت خواهد کرد.

تا این‌جا باید به خاطر داشته باشید که :

پیامدهای منظم‌سازیِ L2 بر روی:

- تابع هزینه : یک عبارتِ منظم‌سازی به هزینه اضافه می‌شود
- تابعِ انتشارِ پس‌رو: عبارات اضافه‌ای در گرادیانها با توجه به وزنِ ماتریس‌ها اضافه می‌شود.
- وزن‌ها به سمت مقادیر کمتر هدایت می‌شوند.

۶.۸ Dropout

dropout یک تکنیک منظم‌سازیِ پرکاربرد است که مختص یادگیری عمیق می‌باشد. در این تکنیک، به صورت تصادفی، برخی از نورون‌ها را در هر تکرار، *خاموش* می‌کنیم. هنگامی که این کار صورت می‌گیرد و برخی نورونها را خاموش می‌کنید، شما در واقع مدل خود را تغییر می‌دهید. ایده‌ی پشت این کار این است که در هر تکرار، شما یک مدل متفاوت که یک زیرمجموعه از نورونهای شما را شامل می‌شود، آموزش می‌دهید. با dropout از آنجایی که هر نورون ممکن است در لحظه قطع شود، نورون های دیگر یاد میگیرند که به فعال شدن نورون های دیگر حساسیت کمتری داشته باشند.

۱.۶.۸ انتشارِ پیش‌رو همراه با Dropout

تمرین سوم - forward_propagation_with_dropout

انتشارِ پیش‌رو [۱] با dropout را پیاده‌سازی کنید. شما یک شبکه عصبیِ سه لایه را استفاده و به لایه پنهانِ اول و دوم، dropout را اضافه خواهید کرد. ما dropout را به لایه ورودی یا لایه خروجی، اعمال نمی‌کنیم.

دستورالعمل در این قسمت شما باید برخی از نورونها را در لایه‌های یک و دو خاموش کنید، برای این‌کار، ۴ مرحله را باید طی کنید.

۱. ابتدا یک ماتریس تصادفیِ $D^{[1]} = \left[d^{[1](1)}d^{[1](2)}\dots d^{[1](m)}\right]$ را که $d^{[1]}$ متغیری هم شکل با $a^{[1]}$ و دارای مقادیر تصادفی بین صفر تا یک است را میسازیم و ابعادِ $D^{[1]}$ ، مشابهِ $A^{[1]}$ است.

۲. هر ورودیِ $D^{[1]}$ را با احتمالِ `keep_prob` برابر یک، و در غیر این صورت، برابر صفر قرار می‌دهیم.

راهنمایی: اگر $\text{keep\_prob} = 0.8$ قرار دهیم، به معنی آن است که ما می‌خواهیم حدود ۸۰ درصد از نورون‌ها نگه داریم و حدود ۲۰ درصد آن‌ها را خاموش کنیم. ما می‌خواهیم برداری شامل ۱-ها و صفر-ها درست کنیم که حدود ۸۰ درصد آن مقدارِ ۱ و الباقی، دارای مقدارِ صفر باشند. کدِ پایتون :

```
X = (X < keep_prob).astype(int)
```

ز لحاظ مفهومی مانند قطعه کد شرطیِ زیر است: (برای سادگی، یک آرایه یک بعدی در نظر گرفته شده است):

```
for i,v in enumerate(x):
    if v < keep_prob:
        x[i] = 1
    else: # v >= keep_prob
        x[i] = 0
```

نکته آن که عبارتِ $\text{X} = (\text{X} < \text{keep\_prob}).\text{astype(int)}$ با آرایه‌های چند بُعدی کار می‌کند و خروجی آن، هم-بُعدِ ورودی آن است.

همچنین این نکته را در نظر داشته باشید که بدون استفاده از `.astype(int)` ، نتیجه آرایه ای از بولین‌های `True` و `False` خواهد بود که پایتون آنها را در صورت ضرب با اعداد، بصورت خودکار به صفر و یک تبدیل خواهد کرد.

[۱] forward propagation

۳. $A^{[1]}$ را برابرِ $A^{[1]} * D^{[1]}$ قرار دهید. (در این مرحله شما دارید تعدادی از نورون‌ها را خاموش می‌کنید) شما می‌توانید به $D^{[1]}$ به عنوان ماسکی نگاه کنید که با ضرب شدن در یک ماتریس دیگر، تعدادی از مقادیر را نادیده می‌گیرد.

۴. $A^{[1]}$ را بر `keep_prob` تقسیم کنید. با انجام این کار، شما اطمینان حاصل خواهید کرد که نتیجه‌ی هزینه، همان نتیجه‌ی مورد انتظار پیش از اعمالِ dropout را خواهد داشت. (از این تکنیک به نامِ dropout وارونه، یاد می‌شود.)

۲.۶.۸ انتشارِ پس‌رو همراه با dropout

تمرین چهارم - backward__propagation__with__dropout

در این بخش قصد داریم انتشارِ رو به عقب همراه با dropout را پیاده‌سازی کنیم. مانند قبل، شما یک شبکه سه لایه را آموزش می‌دهید. dropout را به لایه پنهانِ اول و دوم و با استفاده از $D^{[1]}$ و $D^{[2]}$ که در cache ذخیره شده اند، اضافه کنید.

دستورالعمل: انتشارِ پس‌رو همراه با dropout کار آسانی است. شما باید دو مرحله را پیگیری کنید:

۱. شما قبلا در طی مرحله‌ی انتشارِ پیش‌رو تعدادی از نورون ها را خاموش کرده اید. در انتشارِ پس‌رو شما باید همان نورون ها را با اعمال کردنِ همان ماسکِ $D^{[1]}$ بر روی `dA1` خاموش کنید.

۲. حینِ انتشار پیش‌رو شما `A1` را بر `keep_prob` تقسیم کردید. در انتشارِ پس‌رو شما باید مجددا `dA1` بر `keep_prob` تقسیم کنید. (توجیه ریاضیاتی این است که چون $A^{[1]}$ را توسطِ `keep_prob` مقیاس‌بندی کردید، پس باید مشتق آن، $dA^{[1]}$ را نیز توسطِ `keep_prob` مقیاس‌بندی کنید.)

پس از تکمیل کردنِ کدها در سلول اول و دومِ این قسمت، مدل را با dropout (`keep_prob = 0.86`) اجرا کنید. این عدد به این معناست که در هر تکرار شما حدودِ ۱۴% از نورونها را خاموش خواهید کرد. تابعِ `model()` اکنون بجایِ `forward_propagation` ، تابعِ `forward_propagation_with_dropout` و به جای `backward_propagation` ، تابعِ `backward_propagation_with_dropout` را اجرا خواهد کرد. پس از اجرای سلول سوم در این بخش در فایل ژوپیتر این جلسه، مشاهده خواهید کرد که dropout به خوبی کار می‌کند. دقتِ آزمون به ۹۵% افزایش یافته است و مدل شما دیگر بیش‌برازش نشده است. برای دیدنِ مرز تصمیم، کدِ سلول چهارمِ این بخش در فایل ژوپیتر را اجرا کنید.

نکته:

دقت داشته باشید که شما باید dropout را فقط بر روی مجموعه آموزشی اعمال کنید

همچنین لازم است بدانید که چهارچوب [۱] های یادگیری عمیق، مانند تنسورفلو، کراس و ... همراه یک لایه‌ی dropout ارائه شده اند.

نکاتی که در مورد dropout باید به یاد داشته باشید :

- dropout یک روش برای منظم‌سازی می‌باشد.
- dropout فقط باید هنگام مرحله آموزش انجام بگیرد و نباید در حین مرحله آزمون، از این روش استفاده شود.
- dropout باید هم در مرحله انتشار پس‌رو و هم در انتشار پیش‌رو اعمال شود.
- حین مرحله آموزش، هر لایه‌ی dropout را بر keep_prob تقسیم کنید تا مقادیر فعالساز نسبت به قبل، تغییری نکنند. برای مثال اگر مقدارِ keep_prob برابر با ۰/۵ باشد آنگاه به طور متوسط، نیمی از گره‌ها خاموش خواهند شد. پس خروجی نصف می‌شود، زیرا فقط نیمه‌ی باقی مانده در حل مشارکت می‌کنند. تقسیم بر ۰/۵ یعنی ضرب در ۲ ، بنابراین خروجی اکنون همان مقدارِ مورد انتظار را خواهد داشت.

۷.۸ نتیجه‌گیری

در ادامه نتایج حاصل از سه مدلی که در این تمرین بررسی کردیم را مشاهده می‌کنید:

مدل	دقت آموزش	دقت آزمون
شبکه عصبی سه لایه، بدون منظم‌سازی	۹۵%	۹۱/۵%
شبکه عصبی سه لایه با منظم‌سازیِ L2	۹۴%	۹۳%
شبکه عصبی سه لایه همراه با dropout	۹۳%	۹۵%

همانطور که مشاهده می‌کنید، منظم‌سازی بر عملکرد مجموعه آموزشی، تاثیر منفی می‌گذارد. این بخاطر آن است که منظم‌سازی، شبکه را محدود کرده و از بیش‌برازش جلوگیری می‌کند. اما از آنجایی که در مجموعه آزمون، باعث نتایج بهتری می‌شود، در کل روش خوبی است.

[۱] framework