بسمه تعالی

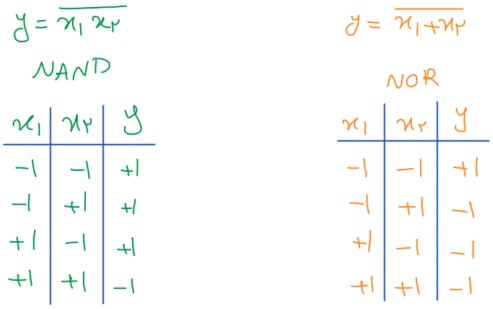
پاسخ سری چهارم تمرینات درس یادگیری ماشین

امیرحسین رمضانی بناب (۹۹۲۱۰۲۹۴)

۱ سوال ۱

۱.۱ آ

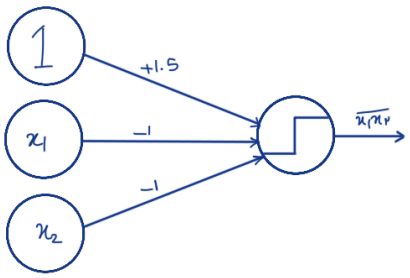
ابتدا جدول توابع NAND و NOR را رسم می‌کنیم:



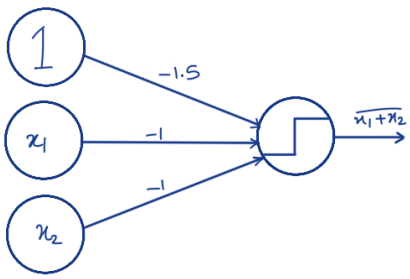
x_1	x_2	y
-1	-1	$+1$
-1	$+1$	$+1$
$+1$	-1	$+1$
$+1$	$+1$	-1

x_1	x_2	y
-1	-1	$+1$
-1	$+1$	-1
$+1$	-1	-1
$+1$	$+1$	-1

حال یک MLA برای NAND طراحی می‌کنیم. خروجی زمانی -1 است که هم $x_1 = +1$ و هم $x_2 = +1$ و در غیر اینصورت خروجی $+1$ است. برای رسیدن به چنین نتیجه‌ای کافی است وزن بایاس را عدد بزرگی مثل 1.5 قرار دهیم و وزن دو یال دیگر را -1 قرار دهیم تا خروجی تنها زمانی -1 شود که $x_1 = +1$ و $x_2 = +1$.



حال یک MLA برای NOR طراحی می‌کنیم. خروجی زمانی $+1$ است که هم $x_1 = -1$ و هم $x_2 = -1$ و در غیر اینصورت خروجی -1 است. برای رسیدن به چنین نتیجه‌ای کافی است وزن بایاس را عدد کوچکی مثل -1.5 قرار دهیم و وزن دو یال دیگر را -1 قرار دهیم تا خروجی تنها زمانی $+1$ شود که $x_1 = -1$ و $x_2 = -1$.

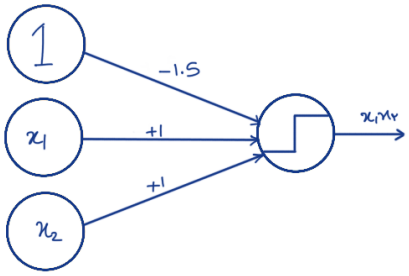


۲.۱ ب

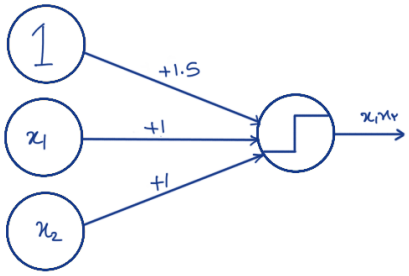
عبارت خواسته شده را به شکل زیر ساده می‌کنیم:

$$(x > y \wedge y < z) \vee (x < y \wedge y > z)$$
$$\rightleftarrows (sign(x-y) = +1 \wedge sign(z-y) = +1) \vee (sign(x-y) = -1 \wedge sign(z-y) = -1)$$
$$\rightleftarrows (sign(x-y) \wedge sign(z-y)) \vee (\overline{sign(x-y)} \wedge \overline{sign(z-y)})$$

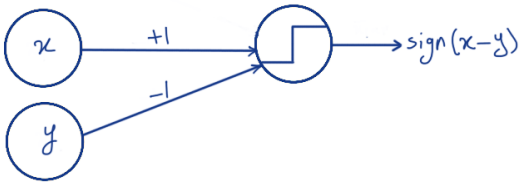
حال یک پرسپترون برای تابع AND به شکل زیر داریم:



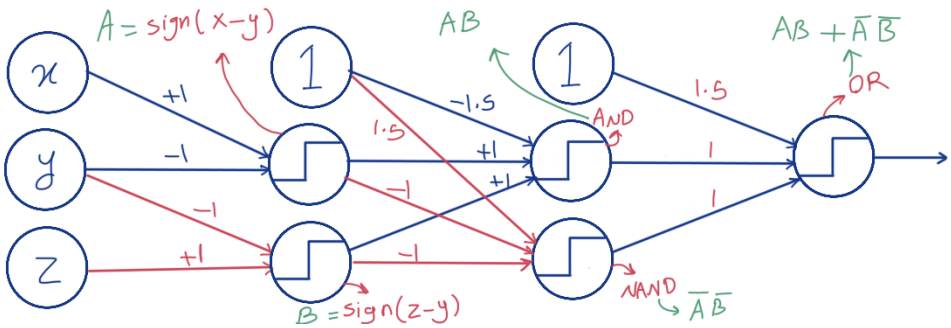
و یک پرسپترون برای تابع OR به شکل زیر داریم:



همچنین برای توابع علامت نیز به سادگی می‌توان یک پرسپترون ساخت. به عنوان مثال داریم:



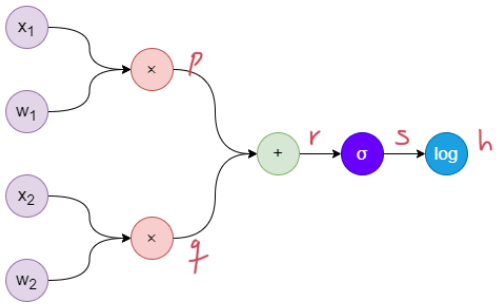
با ترکیب توابع بالا نتیجه حاصل خواهد شد :



۲ سوال ۲

۱.۲ آ

ابتدا نام‌گذاری‌های زیر را روی شکل انجام داده‌ایم:



از روی شکل مقدار این متغیرها مشخص است :

$$p = w_1 x_1 = 2 \quad , \frac{\partial p}{\partial x_1} = w_1 \quad , \frac{\partial p}{w_1} = x_1$$

$$q = w_2 x_2 = -3 \quad , \frac{\partial q}{\partial x_2} = w_2 \quad , \frac{\partial q}{w_2} = x_2$$

$$r = p + q = -1 \quad , \frac{\partial r}{\partial p} = 1 \quad , \frac{\partial r}{q} = 1$$

$$s = \sigma(r) = \frac{1}{1+e} \quad , \frac{\partial s}{\partial r} = \sigma'(r) = \frac{e^{-r}}{(e^{-r}+1)^2}$$

$$h = ln(s) = ln(\frac{1}{1+e}) \quad , \frac{\partial h}{\partial s} = \frac{1}{s}$$

پس داریم :

$$\frac{\partial h}{\partial h} = 1$$

$$\frac{\partial h}{\partial s} = \frac{1}{s} = e + 1$$

$$\frac{\partial h}{\partial r} = \frac{\partial h}{\partial s}\frac{\partial s}{\partial r} = \frac{1}{s}\frac{e^{-r}}{(e^{-r}+1)^2} = (e^{-r}+1)\frac{e^{-r}}{(e^{-r}+1)^2} = \frac{e^{-r}}{e^{-r}+1} = \frac{1}{1+e^{r}} = \frac{e}{e+1}$$

$$\frac{\partial h}{\partial p} = \frac{\partial h}{\partial r}\frac{\partial r}{\partial p} = \frac{1}{1+e^{r}} = \frac{1}{1+e^{p+q}} = \frac{e}{e+1}$$

$$\frac{\partial h}{\partial q} = \frac{\partial h}{\partial r}\frac{\partial r}{\partial q} = \frac{1}{1+e^{r}} = \frac{1}{1+e^{p+q}} = \frac{e}{e+1}$$

$$\frac{\partial h}{\partial x_1} = \frac{\partial h}{\partial p}\frac{\partial p}{\partial x_1} = \frac{w_1}{1+e^{p+q}} = \frac{e}{e+1}$$

$$\frac{\partial h}{\partial w_1} = \frac{\partial h}{\partial p}\frac{\partial p}{\partial w_1} = \frac{x_1}{1+e^{p+q}} = \frac{2e}{e+1}$$

$$\frac{\partial h}{\partial x_2} = \frac{\partial h}{\partial p}\frac{\partial p}{\partial x_2} = \frac{w_2}{1+e^{p+q}} = \frac{3e}{e+1}$$

$$\frac{\partial h}{\partial w_2} = \frac{\partial h}{\partial p}\frac{\partial p}{\partial w_2} = \frac{x_2}{1+e^{p+q}} = \frac{-e}{e+1}$$

۲.۲ ب

۱ ۱.۲.۲

در این روش خطا بر اساس کل دیتاست آموزش به شکل یکجا محاسبه می‌گردد. یعنی در هر قدم بروزرسانی وزن‌ها داریم

$$E_{in}(h) = \sum_{n=1}^{N} e(h(x_n), y_n)$$

به دلیل اینکه برای هر مرحله‌ی بهینه‌سازی w نیاز به یک‌بار محاسبه‌ی خطا روی کل دیتاست آموزش داریم، هزینه محاسباتی این روش بسیار بالاست. ولی از سمت دیگر به دلیل اینکه بین خطای تمام داده‌ها میانگین می‌گیرد، عملا نویز در داده‌ها اثر خود را از دست می‌دهد و دقت افزایش می‌یابد.ولی این افزایش دقت لزوما به معنی خوب بود این روش نمی‌باشد. زیرا در صورتی که فضای وزن‌ها دارای مینیمم‌های محلی زیادی باشد، ممکن است درون یکی از آن‌ها که دور از مینیمم سراسری است، گرفتار شود. همچنین به دلیل اینکه وزن‌ها را فقط بعد از دیدن کل دیتاست آموزش تغییر می‌دهد، در صورتی که اندازه دیتاست بزرگ باشد، سرعت همگرایی آن کاهش خواهد یافت. هر چند در هر مرحله، به سمت نقطه‌ی بهینه نزدیک‌تر خواهد شد.

۲ ۲.۲.۲

در این روش در هر گام خطا را بر اساس یکی از داده‌ها محاسبه می‌کنیم و بر اساس آن اقدام به بروز رسانی مدل می‌کنیم.

$$E_{in}(h) = e(h(x_n), y_n)$$

امیدریاضی این مقدار($\mathbb{E}[e(h(x_n), y_n)]$) را می‌توان توسط تخمین‌گر ناریب زیر تخمین زد

$$\mathbb{E}[E_{in}(h)] = \sum_{n=1}^{N} e(h(x_n), y_n)$$

بنابراین با تعداد تکرارهای به اندازه کافی بزرگ، پاسخ این روش به روش قبلی همگرا می‌شود. به دلیل اینکه در هر تکرار فقط به یک داده نیاز است، هزینه محاسباتی آن بسیار کمتر از روش قبلی است. از طرف دیگر به دلیل وجود نویز درون داده‌ها، ممکن است در برخی از مراحل بهینه‌سازی، خطا افزایش یابد ولی روند کلی آن به سمت نقطه‌ی بهینه همگرا خواهد شد. محاسبه‌ی خطا توسط یک داده هرچند ممکن است دقت آن را کاهش دهد، ولی به دلیل اعمال نویز، می‌تواند موجبات فرار از بهینه‌های محلی را فراهم آورد. از آنجایی که تعداد تکرارهای این روش بسیار بیشتر از روش قبلی است، می‌تواند روی یک دیتاست بزرگ، سریعتر از روش اول همگرا شود.

۳ ۳.۲.۲

در این روش، دیتاست آموزش را به چند بخش مشخص تقسیم کرده و در هر مرجله بردار وزن را بر اساس میانگین خطای روی یک دسته، بروز رسانی می‌کنیم. این روش به نوعی سعی می‌کند از مزایای هر دو روش بالا استفاده کند. به دلیل اینکه در روش قبلی داده‌ها به ترتیب به شبکه وارد می‌شدند، پیاده سازی برداری آن ممکن نبود و از امکانات سخت افزاری مانند GPU به اندازه کافی استفاده نمی‌شد. با این روش، این امکان فراهم می‌شود و می‌توان از پردازش موازی استفاده نمود. پس به دلیل امکان پردازش موازی ، هزینه محاسباتی آن، می‌تواند نزدیک به روش قبلی باشد. همچنین از آنجایی که عمل میانگین گیری خطاها انجام می‌شود، دقت خوبی برای محاسبه خطا داریم ولی همچنان نمودار خطا به شکل نوسانی نزول می‌کند. این نوسان باعث می‌شود از مزیت روش اول مبنی بر کاهش نویز و از مزیت روش دوم مبنی بر فرار از نقاط بهینه محلی استفاده کنیم. سرعت همگرایی آن در حالت کلی، به دلیل بروزرسانی مداوم پارامترها، می‌تواند بهتر از روش اول باشد و به دلیل دقت مناسب محاسبه‌ی خطا، می‌تواند از روش دوم نیز سریعتر همگرا شود. همه‌ی موارد گفته شده در صورتی است که اندازه‌ی بخش‌های تقسیم‌شده، به درستی انتخاب شده باشد.

۳ سوال ۳

۱.۳ آ

ابتدا E را به شکل زیر بازنویسی می‌کنیم

$$\begin{aligned} E &= (y - \sum_i w_i x_i)^2 + \lambda \sum_i w_i^2 \\ &= \underbrace{(y - \mathbf{x}w)^T (y - \mathbf{x}w)}_{E_m(w)} + \underbrace{\lambda w^T w}_{E_l(w)} \end{aligned}$$

حال اگر گرادیان E را نسبت به w محاسبه کنیم، داریم

$$\begin{aligned} \nabla E(w) &= \nabla E_m(w) + \nabla E_l(w) \\ &= \nabla E_m(w) + 2\lambda w \end{aligned}$$

از طرفی در روش گرادیان نزولی، وزن‌ها به شکل زیر بروز می‌شدند.

$$\begin{aligned} w^{t+1} &= w^t - \eta \nabla E(w^t) \\ &= w^t - \eta(\nabla E_m(w^t) + 2\lambda w^t) \\ &= w^t - \eta \nabla E_m(w^t) - 2\eta\lambda w^t \end{aligned}$$

پس

$$\begin{aligned} w^{t+1} &= w^t - \eta \nabla E_m(w^t) - 2\eta\lambda w^t \\ &= \underbrace{w^t(1 - 2\eta\lambda)}_{\text{weight decay}} - \eta \nabla E_m(w^t) \end{aligned}$$

اگر پارامترهای λ (نرخ یادگیری) و η (نرخ منظم‌سازی) مقادیر کوچکی داشته باشند، عبارت $1 - 2\eta\lambda$ مقداری کمتر از 1 دارد و این مورد باعث می‌شود موقعی که می‌خواهیم از w^t به w^{t+1} برسیم ابتدا آن را در یک عدد کوچک‌تر از 1 ضرب می‌کنیم(weight decay) و سپس در جهت عکس گرادیان نزول می‌کنیم. به این دلیل، به این روش weight decay می‌گویند.

۲.۳ ب

۱.۲.۳ معرفی نوتیشن

فرض کنیم A و B دو بردار به شکل زیر باشند:

$$\begin{aligned} a &=< a_1, a_2, ..., a_n > \\ b &=< b_1, b_2, ..., b_n > \end{aligned}$$

تعریف می‌کنیم

$$|a| =< |a_1|, |a_2|, ..., |a_n| > \qquad (*)$$

و

$$\frac{a}{b} =< \frac{a_1}{b_1}, \frac{a_2}{b_2}, ..., \frac{a_n}{b_n} > \qquad (**)$$

همچنین $\mathbf{1}$ نمایانگر یک بردار ستونی به اندازه‌ی n است که تمام عناصر آن برابر 1 هستند.

۲.۲.۳ حل

ابتدا E را به شکل زیر بازنویسی می‌کنیم

$$\begin{aligned} E &= (y - \sum_i w_i x_i)^2 + \lambda \sum_i |w_i| \\ &\overset{*}{=} \underbrace{(y - \mathbf{x}w)^T (y - \mathbf{x}w)}_{E_m(w)} + \underbrace{\lambda \mathbf{1}_d^T |w|}_{E_l(w)} \end{aligned}$$

حال اگر گرادیان E را نسبت به w محاسبه کنیم، داریم

$$\begin{aligned} \nabla E(w) &= \nabla E_m(w) + \nabla E_l(w) \\ &\overset{**}{=} \nabla E_m(w) + \lambda \frac{w}{|w|} \end{aligned}$$

از طرفی در روش گرادیان نزولی، وزن‌ها به شکل زیر بروز می‌شدند.

$$\begin{aligned} w^{t+1} &= w^t - \eta \nabla E(w^t) \\ &= w^t - \eta(\nabla E_m(w^t) + \lambda \frac{w^t}{|w^t|}) \\ &= w^t - \eta \nabla E_m(w^t) - \lambda \frac{w^t}{|w^t|} \end{aligned}$$

پس اگر بخواهیم به شیوه برداری نمایش دهیم داریم

$$\begin{aligned} w^{t+1} &= w^t - \eta \nabla E_m(w^t) - \lambda \frac{w^t}{|w^t|} \\ &=< w_1^t - \lambda \frac{w_1^t}{|w_1^t|}, w_2^t - \lambda \frac{w_2^t}{|w_2^t|}, ..., w_d^t - \lambda \frac{w_d^t}{|w_d^t|} > -\eta \nabla E_m(w^t) \end{aligned}$$

همانطور که مشخص است تفاوت این روش با روش قبلی این است که قبل از حرکت در جهت نزول گرادیان، به جای اینکه کل بردار وزن در یک عدد ثابت ضرب شود، تغییر دیگری در آن ایجاد شده است. به این شکل که اگر وزنی مثبت بوده باشد، عدد λ از آن کسر شده است و اگر منفی باشد، عدد λ به آن اضافه شده است.
در روش اول وزن‌هایی که اندازه‌ی بزرگ‌تر از ۱ دارند، مشارکت بالایی در خطا خواهند داشت و روش اول سعی خواهد کرد که این وزن‌ها را به اندازه‌ی ۱ نزدیک نماید. همچنین وزن‌هایی که اندازه‌ی کوچکتر از ۱ داشته باشند به دلیل اینکه به توان ۲ می‌رسند، مشارکت کمتری در خطا خواهند داشت و روش اول کمتر سعی می‌کند که این وزن‌ها را تغییر دهد. اما روش دوم، به دلیل اینکه فقط اندازه (و نه توان دوم) را لحاظ می‌کند، حتی وزن‌های کوچک را نیز در خطا مشارکت می‌دهد و سعی می‌کند آن‌ها را کاهش داده و به صفر نزدیک‌تر کند. در نتیجه برخی از ویژگی‌ها وزنشان را از دست می‌دهند و به این ترتیب عملا feature selection انجام می‌شود. به شکلی که ویژگی‌هایی که اهمیت پایین تری دارند، در محاسبه‌ی خروجی لحاظ نمی‌شوند و این مورد باعث بهبود عملکرد مدل خواهد شد.

۴ سوال ۴

۱.۴ آ

Validation Set دادگانی است که از آن به عنوان یک ابزار برای انتخاب مدل استفاده می‌شود.
به این شکل که در انتخاب پارامترهای مدل مانند پارامتر Regularization یا انتخاب تعداد Epoch ها برای Early Stopping یا انتخاب یک مدل بین چندین مدل کاندید، می‌توانیم از این دادگان استفاده کنیم که دقت مدل روی آن‌ها مستقل از دقت مدل روی دادگان آموزشی است. اما به خاطر اینکه در هنگام انتخاب مدل نهایی، دادگان Validation را دخالت می‌دهیم، برای گزارش دقت مدل نباید از آن دادگان استفاده کنیم. زیرا مدل به این دادگان بایاس شده و Generalization کاهش می‌یابد و دقت E_{out} مناسبی نخواهد داشت. بدین منظور از دادگان تست استفاده می‌شود تا گزارش ارزیابی مدل را روی مجموعه‌ی دادگانی مستقل از دادگان آموزش و Validation انجام دهیم تا یک Unbiased Estimator برای E_{out} داشته باشیم.

۲.۴ ب

بله. اگر فرض کنیم مجموعه‌ی Validtion Set برابر $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), ..., (\mathbf{x}_K, y_K)$ باشد و همه‌ی این داده‌ها به شکل $i.i.d$ باشند و از دیتاست آموزش مستقل باشند، خطای مدل روی این مجموعه برابر است با

$$E_{val} = \frac{1}{K}\sum_{k=1}^{K} e(h(\mathbf{x}_k), y_k)$$

حال اگر امیدریاضی آن را محاسب کنیم، داریم

$$\begin{aligned}\mathbb{E}[E_{val}] &= \frac{1}{K} \times K\mathbb{E}[e(h(\mathbf{x}_k), y_k)] \\ &= \mathbb{E}[e(h(\mathbf{x}_k), y_k)] \\ &= E_{out}\end{aligned}$$

پس E_{val} یک تخمین‌گر ناریب برای E_{out} است. از آنجایی که با همین استدلال خطای روی Test Set نیز یک تخمین‌گر ناریب برای E_{out} است. پس E_{val} یک تخمین‌گر ناریب برای خطای روی Test Set است.
اگر K تعداد فولدها و N تعداد کل داده‌های اعتبارسنجی باشد، در KFold Cross Validation خطای Validation به شکل زیر تعریف می‌شود.

$$E_{val} = \frac{1}{K}\sum_{k=1}^{K}[\frac{1}{N/K}\sum_{i\in fold_k} e(g_k^-(\mathbf{x}_i), y_i))]$$

در حالت کلی این تخمین‌گر، یک تخمین‌گر Unbiased برای E_{out} نیست ولی اگر به تعداد کافی فولد داشته باشیم که $g_k^- \approx g$ در آن صورت میتوانیم داشته باشیم

$$\begin{aligned}E_{val} &\approx \frac{1}{K}\sum_{k=1}^{K}[\frac{1}{N/K}\sum_{i\in fold_k} e(g(\mathbf{x}_i), y_i))] \\ &= \frac{1}{N}\sum_{i=1}^{N} e(g(\mathbf{x}_i), y_i))\end{aligned}$$

از آنجایی که دیدم

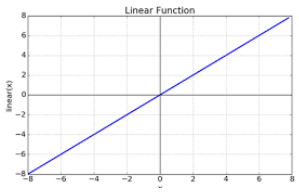
$$\mathbb{E}[E_{val}] = E_{out}$$

پس این تخمین‌گر یک تخمین‌گر تقریبا ناریب می‌باشد.

۵ سوال ۵

۱.۵ مدل اولیه

- تعداد لایه‌های نهان : 1
  - تعداد نورون : 5
  - تابع فعال ساز لایه‌ی نهان : خطی



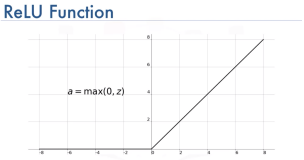
- تعداد Epoch آموزش برای هر λ : 2
- نرخ یادگیری : 0.01
- تعداد Fold ها : 10
- پارامترهای λ : $\{0.001, 0.0005, 0.0001\}$

پس از آموزش مدل با پارامترهای مشخص‌شده، دقت 10-Fold مدل برابر 81.84 درصد با $\lambda = 0.0001$ به دست آمد.
با آموزش مدل روی کل دادگان آموزش، دقت آن روی دادگان تست برابر 82.25 درصد شد.

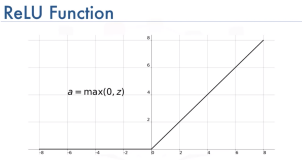
۲.۵ مدل بهبودیافته

در این بخش ابتدا تعداد لایه‌های نهان را از یک به دو لایه افزایش دادیم. بدین ترتیب مدل می‌تواند پترن‌ها پیچیده‌تری را روی دادگان تشخیص دهد و فضای فرضیه‌ی بزرگتری داشته باشد که در نهایت می‌تواند منجر به افزایش دقت مدل شود. از طرف دیگر، تعداد نورون‌ها در هر لایه به مقدار بسیار زیادی افزایش یافت. اعمال این تغییرات به دلیل بزرگ‌کردن بسیار زیاد فضای حالت می‌توانست منجر به Overfit شدن مدل شود که برای جلوگیری از این مورد، به جای تابع فعالسازی خطی از تابع فعالسازی Relu استفاده کردیم. این تابع فعالسازی چون سیگنال‌های کمتر از ۰ را به ۰ مپ می‌کند، باعث می‌شود که کمی از پیچیدگی مدل کاسته شود و از Overfit شدن آن جلوگیری شود.
همچنین به دلیل پیچیده شدن مدل باید تعداد Epoch ها افزایش می‌یافت تا بتوانیم به دقت مناسبی دست یابیم.
با تغییرات اعمال شده، مدل نهایی به شکل زیر است

- تعداد لایه‌های نهان : 2
  - لایه‌ی اول
    - تعداد نورون : 100
    - تابع فعال ساز لایه‌ی نهان : تابع Relu



  - لایه‌ی دوم
    - تعداد نورون : 200
    - تابع فعال ساز لایه‌ی نهان : تابع Relu



- تعداد Epoch آموزش : 10
- نرخ یادگیری : 0.01

در نهایت دقت مدل بهبودیافته روی دادگان تست برابر 97.11 درصد شد که بهبود قابل توجهی را نشان می‌دهد.