تمرین ۳ درس یادگیری ماشین

آریا جلالی
۹۸۱۰۵۶۶۵

۱ Comparison between some models

۱.۱

درخت تصمیم در بسیاری از موارد روی داده‌ی آموزش overfit می‌کند و طبق Bias-variance tradeoff دارای بایاس کم و واریانس بالا هستند. طبق رابطه‌ی بایاس می‌توانیم بنویسیم

$$Bias = (f - E[f^{'}])^2$$

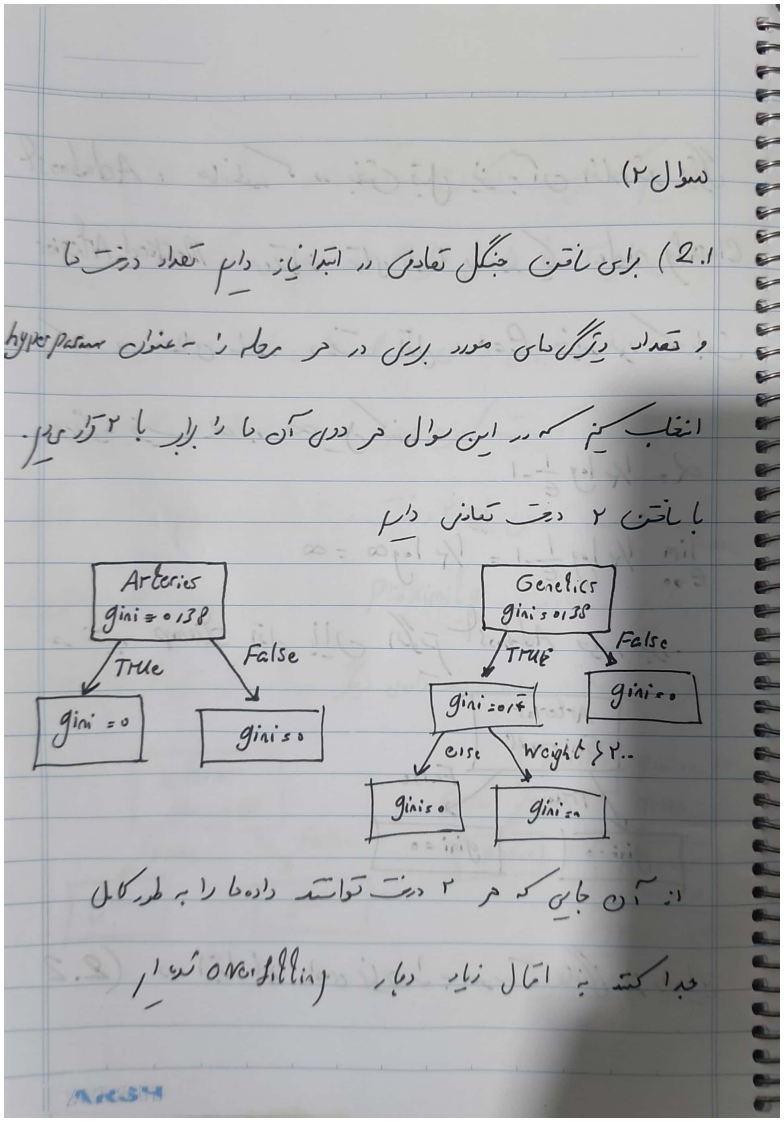
است. همانطور که مشخص است، اگر ما دفعات متعددی درخت تصمیم آموزش دهیم و میانگین آن‌ها را به عنوان classifier نهایی قرار دهیم، خطای ما به مراتب روی داده‌ی تست کاهش پیدا می‌کند.
یکی از روش‌هایی که باعث می‌شود واریانس ما کمتر شود (و خطا روی داده‌ی تست نیز کاهش پیدا کند) استفاده از تعدادی درخت تصمیم با correlation کم است. برای کم کردن هرچه بیشتر این ارتباط می‌توانیم featureهای متفاوت یا داده‌های متفاوت برای آموزش به هر درخت بدهیم و در نهایت بین نتایج بدست آمده رای‌گیری داشته باشیم.
در واقع استفاده از Random Forests با استفاده کردن از چندین درخت تصمیم جلوی overfit شدن که یکی از خصوصیات بارز درخت تصمیم است را می‌گیرد.

۲.۱

XGBoost در مقایسه با Random Forests عملکرد بهتری در دیتاست‌های unbalanced از خود نشان می‌دهد، زیرا اگر در گام اول نتواند به درستی نتیجه را پیشبینی کند، در گام‌های بعدی توجه و اهمیت بیشتری به آن کلاس می‌دهد.
XGBoost سرعت بالاتری نسبت به AdaBoost دارد و حساسیت کمتری نسبت به نویز موجود در دیتاست از خود نشان می‌دهد.
XGBoost در مقابل Gradient Boosting همانند بخش قبل سریعتر است و با استفاده از loss های $l1$ و $l2$ احتمال overfit شدن را می‌گیرد و قدرت generalization بیشتری از خود نسبت به Gradient Boosting نشان می‌دهد.

Fitting a model ۲

۱.۲

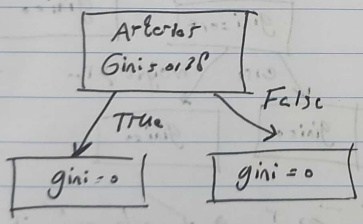


Adaboost : همانطور که در بخش قبل نیز به آن اشاره شد ویژگی Classify node می‌تواند تمای داده‌ها را در یک Mocked Arterior کند . خطای ما در درخت اول $e=0$ خواهد بود که باعث می‌شود ضریب α به ∞ میل کند.

$$\alpha = \frac{1}{2}\log\frac{1}{e}-1$$

$$\lim_{e\to 0} \frac{1}{2}\log\frac{1}{e}-1 = \frac{1}{2}\log\infty = \infty$$

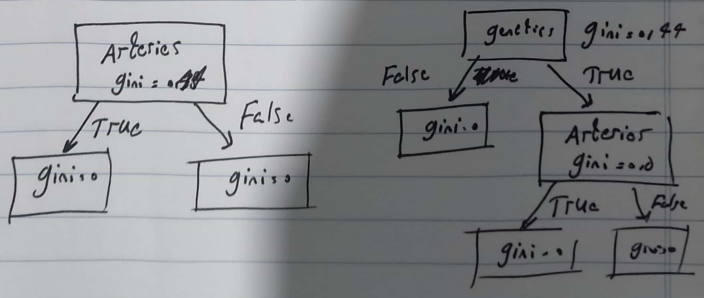
در نتیجه stump اول پایان الگوریتم Adaboost خواهد بود.



2.2) در ابتدا باید داده‌ی خود را به صورت عددی از داده‌های

قرار دهیم که برچسب یکسان با آن دارند. در این دیتاست تنها داده‌ی سطر ۳ با داده‌ی سطر ۴ در برچسب مشترک است. پس حدس اولیه برای داده‌ی سطر ۴ داده‌ی سطر ۳ خواهد بود.

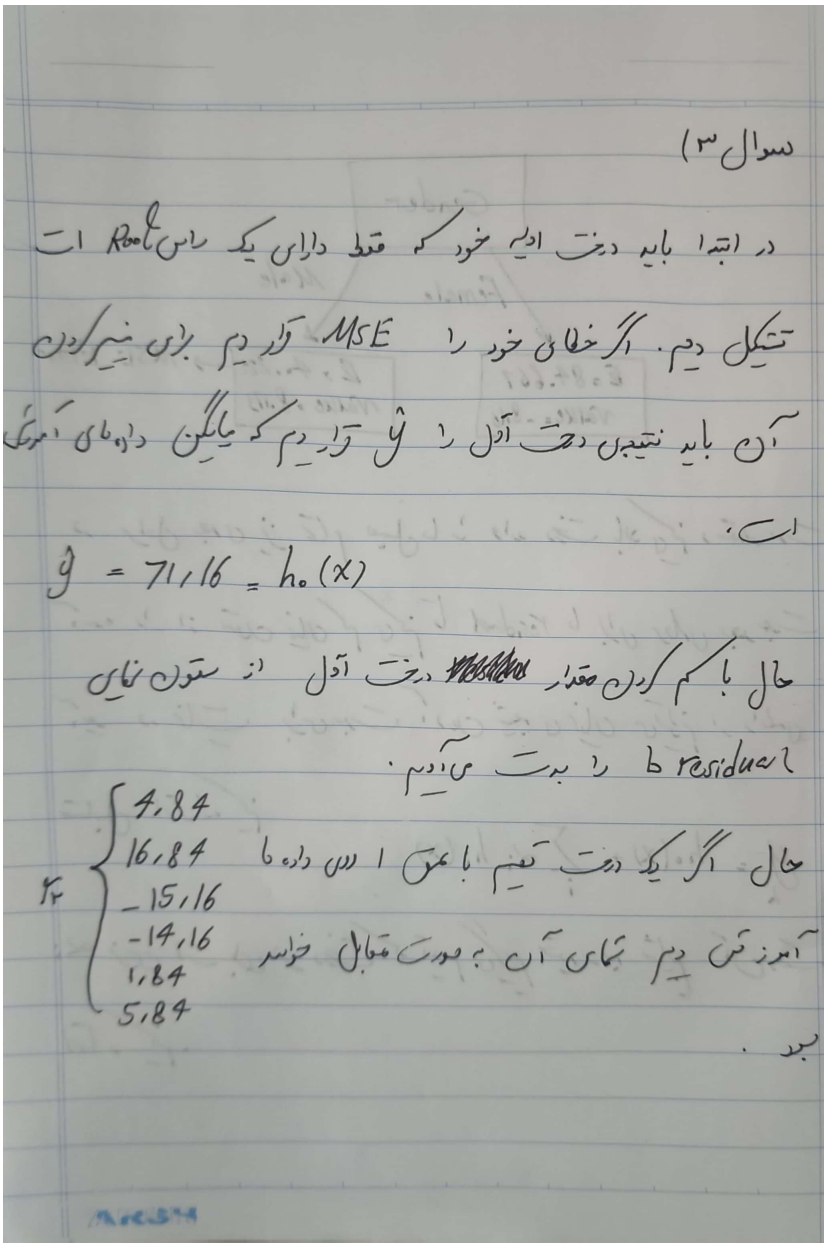
حال برای آپدیت کردن حدس اولیه با استفاده از ۳ داده‌ی اول یک درخت تصمیم با ۲ درخت و ۳ برگ رسم در هر مرحله می‌سازیم و ماتریس Proximity را تشکیل می‌دهیم. این ماتریس نشان می‌دهد هر داده با چند داده دیگر در یک برگ درخت قرار گرفته است.



در هر ۲ درخت داده‌ی سطح 4 تنها با داده‌ی سطح ۳ در یک برگ تزار می‌گیرد و آپدیت نمی‌شود و الگوریتم ما در این مرحله به پایان می‌رسد.

$x_4 = x_3 =$ Yes - 167 - No - Yes

Gradient Boost ۳



سوال ۳)

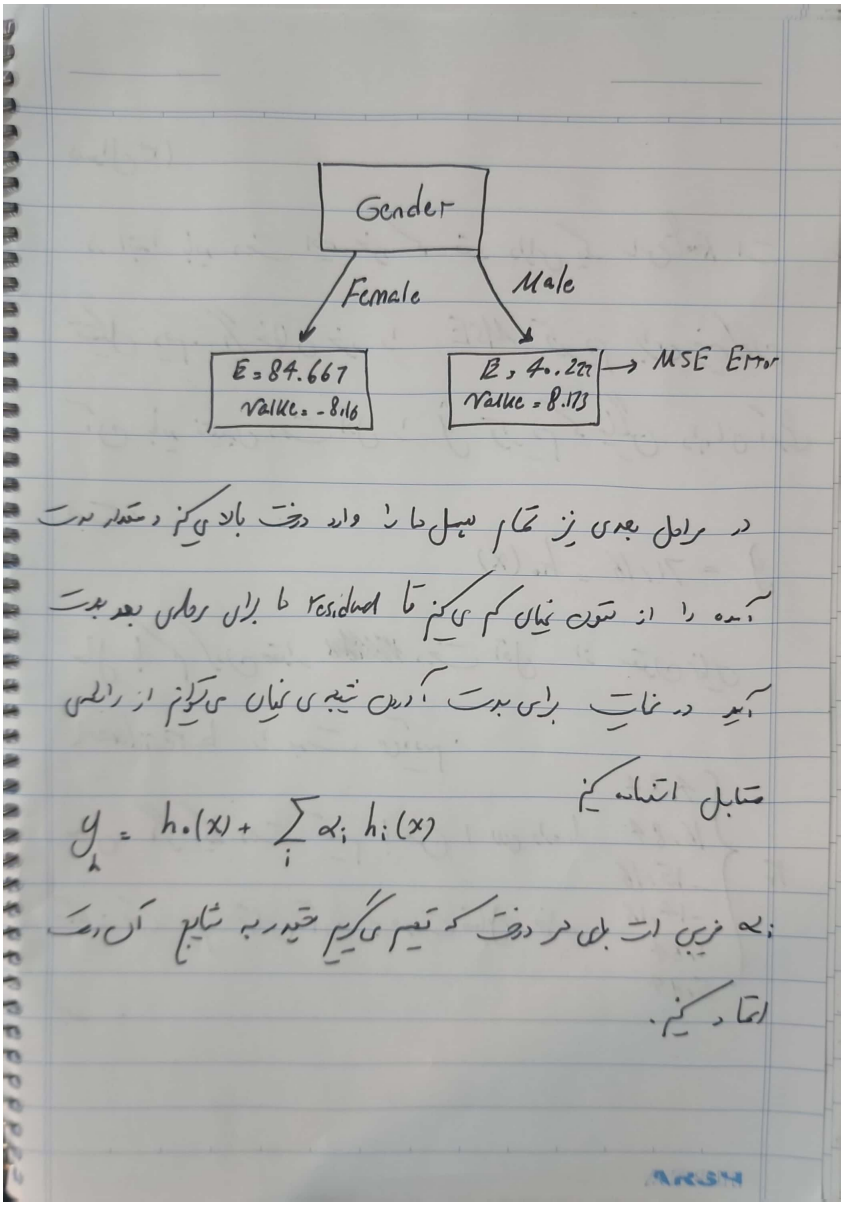
در ابتدا باید درخت اولیه خود که فقط دارای یک راس Root است تشکیل دهیم. اگر خطای خود را MSE در نظر بگیریم برای مینیمم کردن آن باید نتیجه درخت اول را $\hat{y}$ در نظر بگیریم که میانگین داده‌های آموزشی است.

$$\hat{y} = 71.16 = h_0(x)$$

حال با کم کردن مقدار درخت اول از ستون نمایی residual را بدست می‌آوریم.

$$r_2 \begin{cases} 4.84 \\ 16.84 \\ -15.16 \\ -14.16 \\ 1.84 \\ 5.84 \end{cases}$$

حال اگر یک درخت تصمیم با عمق ۱ روی داده‌ها آموزش دهیم نمای آن به صورت مقابل خواهد بود.



۴ Descriptive Questions

۱.۴

Exploding Gradient زمانی رخ می‌دهد که گرادیان اررورها به صورت نمایی بزرگ می‌شوند و این باعث می‌شود در مرحله‌ی آپدیت وزن مقادیر w_i بسیار افزایش یا کاهش پیدا کنند و مدل ما unstable شود و نتواند از داده‌ی آموزشی چیز معنی‌داری یاد بگیرد.
برای حل این مشکل می‌توانیم از روش Weight Initialization است. به این صورت که مقادیر اولیه وزن‌ها را به صورت رندوم انتخاب نمی‌کنیم و سعی می‌کنیم مقادیر داده شده از یک

توزیع نرمال با یک سری پارامتر خاص پیروی کنند تا وزن‌ها در بازه‌ی محدودی قرار بگیرند و مشکل Exploding Gradient با احتمال کمتری رخ دهد.
یکی دیگر از روش‌های حل این مشکل استفاده از Gradient Clipping است. در این روش اگر مقدار گرادیان بزرگتر از یک بازه‌ای باشد مقدار آن برابر با کران بالای بازه‌ی انتخاب شده قرار داده می‌شود و همین عمل برای گرادیان‌ها کوچکتر از بازه و کران پایین انجام می‌شود. با این روش گرادیان‌ها در محدوده‌ی خاصی قرار می‌گیرند و به صورت نامحدود رشد نمی‌کنند.

۲.۴

اضافه کردن لایه‌های بیشتر به مدل باعث پیچیده‌تر شدن آن می‌شود و می‌توانیم بایاس را کاهش دهیم. ولی اگر پیچیدگی مدل به مراتب بیشتر از حد نیاز برای حل سوال باشد، دچار overfitting می‌شویم و مدل ما واریانس بالایی خواهد داشت. از طرفی اضافه کردن لایه‌های زیاد باعث افزایش زمان train و به وجود آمدن مشکلات متعددی مانند Exploding Gradient و Vanishing Gradient می‌شود.

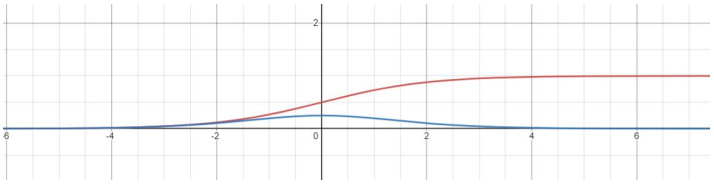
۳.۴

در صورتی که داده‌ی آموزشی ما از قبل به صورت تصادفی چیده نشده باشد، ممکن است ارتباطی بین داده‌های نزدیک به هم باشد و مدل ما سعی می‌کند با یاد گرفتن این ارتباط خطا را کاهش دهد. در صورتی که هدف ما یاد گرفتن توزیع آماری مدل به صورت کلی است. این bias در داده‌ی آموزشی باعث می‌شود مدل به جای یاد گرفتن ویژگی‌ها و روابط توصیف کننده‌ی توزیع داده، روابط بین داده‌ها را یاد بگیرد و دچار overfitting شدید روی داده‌ی آموزشی شود.

۴.۴

$$(1+e^{-x})\sigma = 1 \Rightarrow -e^{-x}\sigma + (1+e^{-x})\frac{d\sigma}{dx} = 0$$

$$\frac{d\sigma}{dx} = \sigma.\frac{e^{-x}}{(1+e^{-x})} = \sigma.\frac{(1+e^{-x})-1}{(1+e^{-x})} = \sigma.[1-\frac{1}{(1+e^{-x})}] = \sigma.(1-\sigma)$$



شکل ۱: شکل تابع و مشتق سیگموید

همانطور که از شکل بالا مشخص است در صورتی که مقدار وزن یا مقدار ورودی به تابع بسیار بزرگ باشد، گرادیان آن تقریبا برابر با ۰ خواهد بود و دچار مشکل Vanishing Gradient می‌شویم و مدل در تغییر وزن‌ها دچار مشکل می‌شود و فرایند Train به اتمام نمی‌رسد یا زمان بسیار زیادی طول خواهد کشید.
یکی از راه‌های حل این مشکل تغییر تابع activation به یک تابع دیگر مانند RELU است.

۵.۴

در این حالت ۲ مشکل ممکن است رخ دهد. اگر همانند صورت سوال فرض کنیم مقادیر بزرگتر از ۵.۰ برابر با کلاس ۱ قرار داده می‌شوند و مقادیر ۵.۰ کمتر برابر با کلاس ۰، مشکلی از لحاظ پیشبینی همواره یک کلاس نخواهیم داشت (در صورتی که مقادیر ۵.۰ و بزرگتر به عنوان کلاس ۰ پیشبینی می‌شدند به دلیل نامنفی بودن خروجی تابع RELU همواره پیشبینی مدل کلاس ۱ بود.) ولی مشکل اساسی این مدل این است که اگر اعضای کلاس ۱ به اشتباه به کلاس ۰ نسبت داده شوند به دلیل منفی بودن نتیجه و صفر بودن مشتق تابع $\sigma(RELU(x))$ در نقاط منفی گرادیان برابر با ۰ می‌بود و مدل از این پیشبینی اشتباه چیزی یاد نمی‌گرفت.

۶.۴

در صورتی که تابع activation نداشته باشیم، خروجی هر لایه را به صورت یک ضرب ماتریسی به صورت مقابل نشان داد

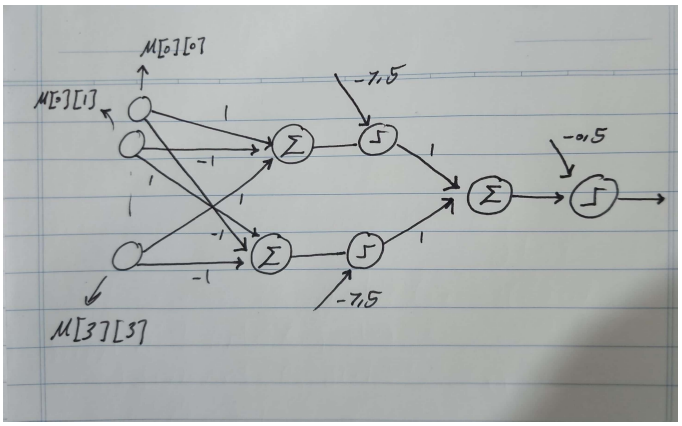
$$A^n = W^n(A^{n-1}) + b^{n-1} = W^n(W^{n-1}A^{n-2} + b^{n-1}) + b^{n-2} = \cdots = W^{'}(X) + b^0$$

$$W^{'}(X) = W^n(W^{n-1}(W^{n-2}(\dots) + b^{n-3}) + b^{n-2})$$

با ضرب متوالی ماتریس‌ها به این نتیجه می‌رسیم که خروجی هر لایه ترکیب خطی‌ای از ورودی‌ها است و مدل ما در واقع یک Regressor است.

۵ Intuitive Questions

۱.۵



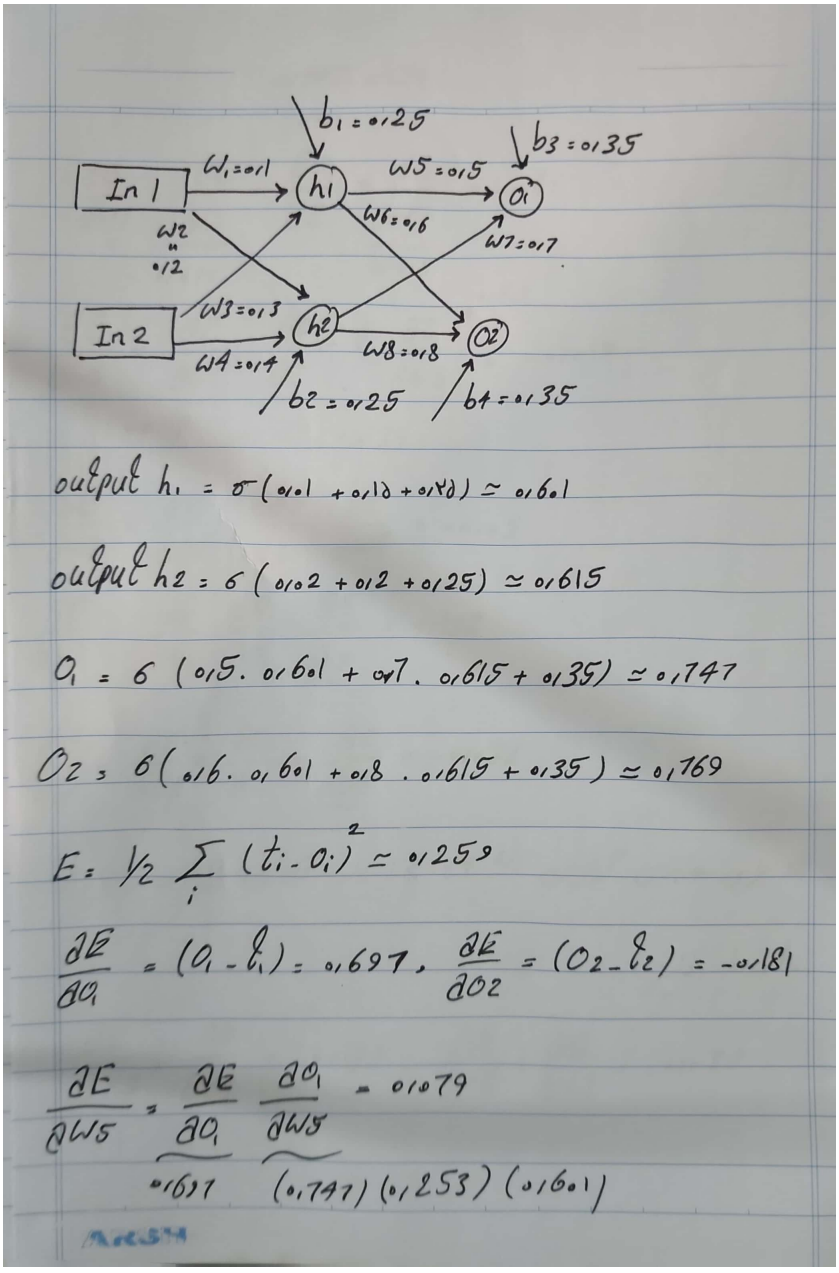
شکل ۲: مدل طراحی شده برای تشخیص شکل شطرنجی

در مدل بالا به پیکسل‌هایی که باید در حالت مدنظر ما ۱ باشند وزن ۱ و به پیکسل‌های ۰ وزن -۱ را می‌دهیم. در این صورت حداکثر مقدار جمع برابر با ۸ خواهد بود، زیرا اگر پیکسل دیگری روشن باشد حداقل یکی از خروجی کم می‌شود. در نهایت این مقدار را با بایاس -۵.۷ جمع می‌کنیم و اگر شکل صفحه همانند صفحه‌ی نشان داده شده باشد مقدار ورودی به تابع پله برابر با ۵.۰ خواهد بود و در غیر اینصورت مقداری منفی خواهد بود.
عکس همین کار برای حالتی که جای پیکسل‌های سیاه و سفید شود انجام شده است و در نهایت نتیجه‌ی خروجی‌ها باهم OR شده‌اند.
در نهایت اگر صفحه‌ی مدنظر تشخیص داده شود خروجی مدل ما برابر با ۱ و در غیر اینصورت برابر با ۰ خواهد بود.

۲.۵

MLP ها رابطه‌ی بین پیکسل‌های اطراف را در نظر نمی‌گیرند و هر پیکسل را مستقل از پیکسل‌های اطراف در نظر می‌گیرند (از دانش prior استفاده‌ای نمی‌شود.) از طرفی مدل‌های MLP حساس به مکان object در تصویر هستند و در اکثر اوقات دچار overfit روی داده‌ی آموزشی می‌شوند. این مدل‌ها دارای Invariance to translation نیستند و تنها داده‌های جدید را می‌توانند تشخیص دهند که در داده‌ی آموزشی نمونه‌ای از آن در همان نقطه وجود داشته باشد.
مشکل دیگر استفاده از MLP و در نظر نگرفتن رابطه‌ی بین پیکسل‌ها وجود پارامترهای فراوان در عکس‌ها است و این باعث کاهش سرعت یادگیری می‌شود. از طرفی منابع پردازشی زیادی برای آموزش نیاز است.

Computational Question ۶



output $h_1 = \sigma(0.1 + 0.15 + 0.4) \simeq 0.601$

output $h_2 = \sigma(0.2 + 0.2 + 0.25) \simeq 0.615$

$O_1 = \sigma(0.5 \cdot 0.601 + 0.7 \cdot 0.615 + 0.35) \simeq 0.747$

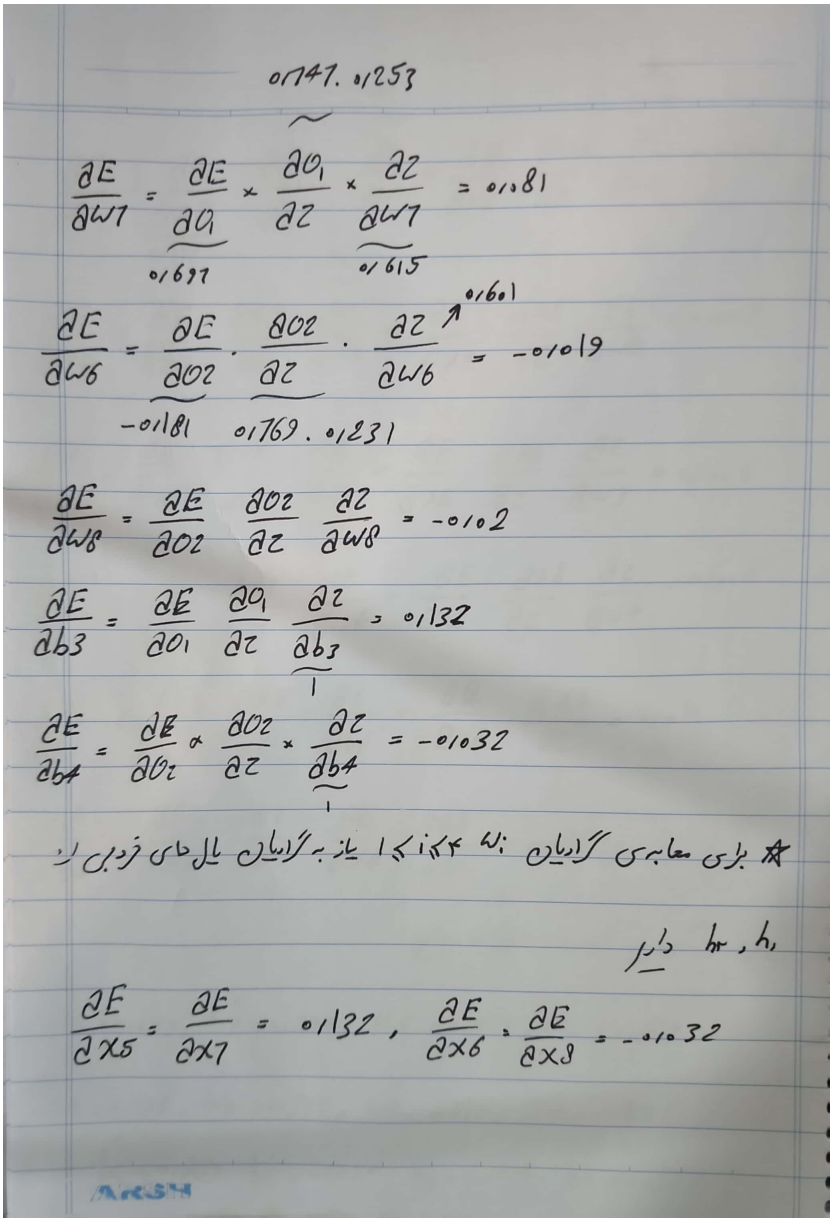
$O_2 = \sigma(0.6 \cdot 0.601 + 0.8 \cdot 0.615 + 0.35) \simeq 0.769$

$E = \frac{1}{2} \sum_i (t_i - O_i)^2 \simeq 0.259$

$\frac{\partial E}{\partial O_1} = (O_1 - t_1) = 0.697, \quad \frac{\partial E}{\partial O_2} = (O_2 - t_2) = -0.181$

$\frac{\partial E}{\partial w_5} = \frac{\partial E}{\partial O_1} \frac{\partial O_1}{\partial w_5} = 0.079$

$0.697 \quad (0.747)(0.253)(0.601)$



$$\frac{\partial E}{\partial w_1} = \frac{\partial E}{\partial x_5}\frac{\partial x_5}{\partial z}\frac{\partial z}{\partial w_1} + \frac{\partial E}{\partial x_6}\frac{\partial x_6}{\partial z}\frac{\partial z}{\partial w_1} = 0.001$$

$$\frac{\partial E}{\partial w_2} = \frac{\partial E}{\partial x_7}\frac{\partial x_7}{\partial z}\frac{\partial z}{\partial w_2} + \frac{\partial E}{\partial x_8}\frac{\partial x_8}{\partial z}\frac{\partial z}{\partial w_2} = 0.002$$

$$\frac{\partial E}{\partial w_3} = \frac{\partial E}{\partial x_5}\frac{\partial x_5}{\partial z}\frac{\partial z}{\partial w_3} + \frac{\partial E}{\partial x_6}\frac{\partial x_6}{\partial z}\frac{\partial z}{\partial w_3} = 0.009$$

$$\frac{\partial E}{\partial w_4} = \frac{\partial E}{\partial x_7}\frac{\partial x_7}{\partial z}\frac{\partial z}{\partial w_4} + \frac{\partial E}{\partial x_8}\frac{\partial x_8}{\partial z}\frac{\partial z}{\partial w_4} = 0.008$$

$$\frac{\partial E}{\partial b_1} = \frac{\partial E}{\partial x_5}\frac{\partial x_5}{\partial z}\frac{\partial z}{\partial b_1} + \frac{\partial E}{\partial x_6}\frac{\partial x_6}{\partial z} = 0.11$$

$$\frac{\partial E}{\partial b_2} = \frac{\partial E}{\partial x_7}\frac{\partial x_7}{\partial z} + \frac{\partial E}{\partial x_6}\frac{\partial x_6}{\partial z} = 0.16$$

پس از آپدیت کردن وزن ها داریم

$w_1' = 0.099$, $w_2' = 0.294$, $w_3' = 0.421$, $w_4' = 0.619$

$w_5' = 0.198$, $w_6' = 0.392$, $w_7' = 0.619$, $w_8' = 0.82$

$b_1' = 0.239$, $b_2' = 0.234$, $b_3' = 0.218$, $b_4' = 0.382$