تمرین سری چهارم

امیرحسین محمدی

۹۹۲۰۱۰۸۱

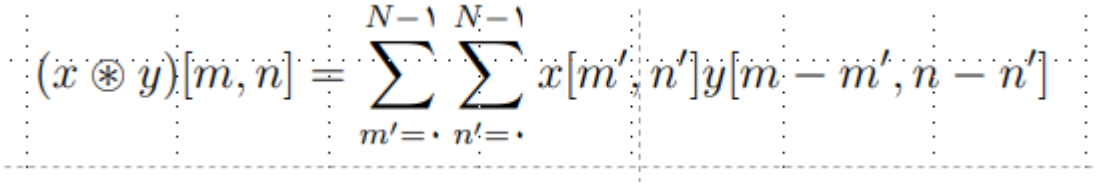
❖ لازم به ذکر این است این تمرین با همفکری با آقایان سعید درویشی، مجتبی زمانی و پوریا لقایی انجام شده است.

۱. (۵ + ۱۰ نمره) می‌خواهیم حاصل کانولوشن دو تصویر x و y، هر یک به ابعاد $N \times N$ را محاسبه کنیم؛ به این صورت که

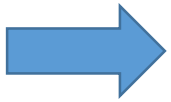
$$(x \circledast y)[m,n] = \sum_{m'=۰}^{N-۱} \sum_{n'=۰}^{N-۱} x[m',n']y[m-m',n-n']$$

(آ) (۵ نمره) مطلوب است محاسبه‌ی مرتبه زمانی محاسبه‌ی این مقدار.

(ب) (۱۰ نمره امتیازی) با معرفی محاسبات میانی، راه‌کاری پیشنهاد کنید که بتوان با آن مقدار فوق را در مرتبه‌زمانی $O(N^۲log(N))$ محاسبه‌کرد و جایگیری راه‌کار خود در این مرتبه‌زمانی را اثبات کنید. بعضا از فرمولیشن‌های این چنینی برای کاهش هزینه‌های محاسباتی شبکه‌های کانولوشنال استفاده می‌شود. (راهنمایی: از تبدیل فوریه استفاده‌کنید)



$$(x \circledast y)[m,n] = \sum_{m'=0}^{N-1} \sum_{n'=0}^{N-1} x[m',n']y[m-m',n-n']$$



مرتبه ی زمانی N^4

- برای هر پیکسل باید این عمیات انجام شود، فرض کنیم N پیکسل در طول داریم و N پیکسل در عرض، بنابراین به ازای هر پیکسل رابطه ی بالا را داریم (ضرب عنصر به عنصر و جمع یک فیلتر و سیگنال)، همچنین می دانیم برای پیاده سازی این عملیات به دو تا حلقه نیاز داریم که مرتبه زمانی آن N^2 می شود. بنابراین اگر بخواهیم روی کل تصاویر این عملیات انجام شود باید دو حلقه ی دیگر نیز اضافه کنیم که در نهایت مرتبه ی زمانی ما N^4 می شود.

قسمت ب

داریم

Convolution theorem

$$r(x) = \{g * h\}(x) = \mathcal{F}^{-1}\{G \cdot H\}, \quad \text{where } \cdot \text{ denotes point-wise multiplication} \quad \textbf{(Eq.1b)}$$

- بنابراین با استفاده از این رابطه می توانیم از این ایده استفاده کنیم که ابتدا عملیات کانوولوشن را به فرم فوریه تبدیل کنیم (چون در فرم فوریه در فضای فرکانس کار می کنیم میتوان عملیات را به صورت یک ضرب و جمع ساده انجام داد) و سپس مجددا از محاسبات به فرم فوریه معکوس بگیریم و نتیجه کانوولوشن را محاسبه کنیم.

- برای محاسبه به فرم فوریه می توانیم از رابطه ی مقابل استفاده کنیم

$$F(u,v) = \sum_{x=0}^{M-1} e^{-j2\pi ux/M} \sum_{y=0}^{N-1} f(x,y) e^{-j2\pi vy/N}$$
$$= \sum_{x=0}^{M-1} F(x,v) e^{-j2\pi ux/M}$$

$$MNf^*(x,y) = \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} F^*(u,v) e^{-j2\pi(ux/M + vy/N)}$$

$$f(x,y) = \frac{1}{MN} \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} F(u,v) e^{j2\pi(ux/M + vy/N)}$$

- همچنین با محاسبه ی این رابطه و تقسیم طرفین بر MN می توانیم معکوس فرم فوریه را محاسبه کنیم.

$$MNf^*(x,y) = \sum_{u=0}^{M-1}\sum_{v=0}^{N-1} F^*(u,v)e^{-j2\pi(ux/M + vy/N)}$$

- برای پیاده سازی این رابطه و محاسبه ی آن به صورت عادی، با مرتبه ی زمانی معادل $(MN)^2$ مواجه هستیم. که در مسئله ی ما N^4 خواهد بود. همونطور که مشخص است این عملیات بسیار پر هزینه است.

بنابراین با توجه به مشکل گفته شده می توانیم، از ایده ی FAST FOURIER TRANSFORM استفاده کنیم، زیرا تبدیل فوریه یک تبدیل جدا پذیر است بنابراین با استفاده از این ویژگی می توانیم مرتبه ی زمای را کاهش دهیم. در واقع می توان ادعا کرد که بدون کشف تبدیل سریع فوریه (FTT) که محاسبات را به صورت چشم گیری کاهش می دهد ($MNlog_2\ MN$ و در مسئله ی ما $N^2log_2\ N$)، روش های دیگر که مبتی بر محاسبات ساده ی کانوولوشن است ارزش عملی کمی دارند.

الگوریتمی که برای اثبات این مسئله انتخاب می شود، اصطلاحاً روش دو برابر سازی متوالی (successive-doubling) است. این الگوریتم خاص فرض می کند که تعداد نمونه ها، ۲ به توان یک عدد صیح (p) است (این مسئله به صورت کلی الزامی نیست). ما بر اساس قضایا و رابطه ی زیر می دانیم که DFTهای دو بعدی را می توان با successive passes های تبدیل های (transform) یک بعدی جداسازی و پیاده سازی کرد:

$$F(u,v) = \sum_{x=0}^{M-1} e^{-j2\pi ux/M} \sum_{y=0}^{N-1} f(x,y)e^{-j2\pi vy/N}$$

$$= \sum_{x=0}^{M-1} F(x,v)e^{-j2\pi ux/M}$$

$$F(x,v) = \sum_{y=0}^{N-1} f(x,y)e^{-j2\pi vy/N}$$

طبق قضایا مرسوم است که معادله ی صفحه ی قبل را به صورت زیر باز نویسی می کنند:

$$F(u) = \sum_{x=0}^{M-1} f(x) e^{-j2\pi ux/M} \qquad u = 0, 1, 2, \ldots, M-1$$

$$F(u) = \sum_{x=0}^{M-1} f(x) W_M^{ux} \qquad u = 0, 1, 2, \ldots, M-1$$

$$W_M = e^{-j2\pi/M}$$

جهت تجزیه به دو بخش جملات فرد و زوج

با جایگذاری M در رابطه ی مقابل داریم

Pیک عدد صحیح و مثبت است، بنابراین می توان نتیجه گرفت M زوج است

فرض می کنیم $M = 2^p$

$$M = 2K$$

$$F(u) = \sum_{x=0}^{2K-1} f(x) W_{2K}^{ux} = \sum_{x=0}^{K-1} f(2x) W_{2K}^{u(2x)} + \sum_{x=0}^{K-1} f(2x+1) W_{2K}^{u(2x+1)}$$

همچنین با توجه به رابطه ی مقابل

$$W_M = e^{-j2\pi/M}$$

خواهیم داشت

$$W_{2K}^{2ux} = W_K^{ux}$$

$$F(u) = \sum_{x=0}^{K-1} f(2x) W_K^{ux} + \sum_{x=0}^{K-1} f(2x+1) W_K^{ux} W_{2K}^{u}$$

بنابراین رابطه ی بالا را می توان به صورت مقابل باز نویسی کرد

$$F(u) = \sum_{x=0}^{K-1} f(2x) W_K^{ux} + \sum_{x=0}^{K-1} f(2x+1) W_K^{ux} W_{2K}^{u}$$

بنابراین می توان جملات فرد و زوج رو جدا کرد و نوشت:

$$F_{even}(u) = \sum_{x=0}^{K-1} f(2x) W_K^{ux}$$

$$F_{odd}(u) = \sum_{x=0}^{K-1} f(2x+1) W_K^{ux}$$

$$u = 0, 1, 2, \ldots, K-1$$

می توان با توجه به روابط بالا نوشت

$$F(u) = F_{even}(u) + F_{odd}(u) W_{2K}^{u}$$

همچنین با توجه به:

$$W_M^{u+K} = W_K^{u}$$

$$W_{2K}^{u+K} = -W_{2K}^{u}$$

داریم:

$$F(u+K) = F_{even}(u) - F_{odd}(u) W_{2K}^{u}$$

از رابطه ی بدست آمده می توان نتیجه گرفت که برای محاسبه ی DFT (تبدیل فوریه گسسته) به ازای M نقطه، می توان عبارت و محسابات را به دو قسمت (جملات زوج و فرد) تقسیم کرد. به طوریکه برای محسابه ی نیمه ی اول F(u) می توانیم با تبدیل M/2 نقطه این محاسبه را انجام داد و نتایج مربوط به F_{odd} و F_{even} را محاسبه کرده و در رابطه ی بالا قرار می دهیم. همچنین نیمه بعدی با استفاده از رابطه ی F(u+k) بدون هیچ تبدیل اضافی محاسبه می شود. جهت محاسبه ی مرتبه ی زمانی در ابتدا دو متغیر m(p) و a(p) را فرض می کنیم که m(p) تعداد ضرب های complex است و a(p) تعداد جمع هاست.

طبق الگوریتم successive-doubling اگر تعداد نمونه ها را 2^P فرض کنیم، اگر p=1 باشد آنگاه تعداد نمونه ها ۲ خواهد بود و برای تبدیل دو نمونه نیاز به محاسبه ی F(0) و F(1) خواهد بود برای محاسبه ی F(0) باید $F_{even}(0)$ و $F_{odd}(0)$ را محاسبه کرد و چون k=1 است طبق روابط بلا نیازی به هیچ ضرب و جمعی برای محاسبه ی $F_{even}(0)$ و $F_{odd}(0)$ نخواهد بود. فقط یک ضرب بین $F_{odd}(0)$ و W_2^0 خواهیم داشت و یک جمع در F(0) خواهیم داشت. سپس برای محاسبه F(1) یک جمع دیگر نیاز داریم که در نهایت داریم m(1)=1 و a(1)=2. همچنین اگر p=2 باشد، آنگاه k=2 و محاسبه ی تبدیل چهار نقطه به دو بخش تقسیم خواهد شد. نیمه ی اول F(u) به دو تبدیل دو نقطه نیاز دارد. تبدیل دو نقطه به m(1) ضرب و a(1) احتیاج دارد. بنابریان برای محاسبه ی دو رابطه به 2m(1) ضرب و 2a(1) جمع احتیاج داریم. همچنین دو تا ضرب و جمع اضافی نیز برای محاسبه ی F(0) و F(1) مورد نیاز است. به دلیل اینکه $F_{odd}(u)W_{2k}^{u}$ در محاسبات قبل برای u={0,1} انجام شده است بنابراین فقط برای F(2) و F(3) نیاز به دو جمع اضافه تر نیز داریم. بنابراین به صورت کلی m(2)=2m(1)+2 عملیات ضرب و a(2)=2a(1)+4 عملیات جمع داریم. بنابراین با یک روند بازگشتی مواجه هستیم.

بنابراین با توجه به این روند بازگشتی می‌توان یک فرمول عمومی برای تعداد ضرب ها و جمع ها به ازای p های گوناگون حساب کرد:

$$\mathrm{m}(p) = 2\mathrm{m}(p-1) + 2^{p-1} \qquad p \geq 1$$

$$\mathrm{a}(p) = 2\mathrm{a}(p-1) + 2^{p} \qquad p \geq 1$$

بنابراین برای M های گوناگون ما مرتبه ی محاسباتی زیر مواجه خواهیم بود:

$$\mathrm{m}(p) = \frac{1}{2} M \log_2 M$$

$$\mathrm{a}(n) = M \log_2 M$$

بنابراین برای M نمونه یا نقطه ما مرتبه زمانی $Mlog_2^M$ مواجه هستیم. در صورتی که در حالت عادی و بدون FFT با مرتبه ی زمانی M^2 مواجه هستیم.

بنابراین برای محاسبه ی یک تصویر به ابعاد N در N در حالت عادی (همانور که گفته شد) با مرتبه ی زمانی N^4 مواجه هستیم و در حالت FFT طبق روابط بدست آمده ($Mlog_2^M$) با مرتبه ی زمانی $N^2log_2^N$ مواجه هستیم.

Ref: Digital.Image.Processing.4th.Edition, صفحه ی ۳۰۳, google

۲. (۲۵ نمره) می‌خواهیم بر دادگان $S = \{(x_i, y_i) | x_i \in R^d, y_i \in \{۰, ۱\}\}$ و $|S| = n$ دسته‌بند خطی $(\hat{y} = w^T x)$ به منظور کمینه‌سازی تابع هزینه ی $J(w) = \frac{۱}{۲}||Xw - Y||_۲$ را بیابیم که در آن ماتریسی X بوده که سطرهایش‌را بردارهای x_i تشکیل داده‌اند و Y برداری ستونی باشد که در آن $Y(i) = y_i$ باشد.

(آ) (۷ نمره) بر حسب اندازه‌ی n، در چه شرایطی حتما پاسخی برای صفرکردن تابع هزینه وجود دارد؟

(ب) (۸ نمره) تصور کنید W مجموعه‌ی بردارهای wای باشند که J را کمینه‌می‌کنند، نشان‌دهید $dim(W)$ کوچک‌تر یا مساوی n است.

(ج) (۱۰ نمره) فرض‌کنید برای حل این مسئله از الگوریتم بهینه‌سازی Stochastic Gradient Descent بهره‌گیریم. نشان‌دهید پاسخ‌نهایی به دست آمده از این الگوریتم در صورتی‌که $dim(W) > ۰$ و $rank(X) = n$ و بردار $w_۰ = ۰$ باشد، wای بوده که از میان فضای n بعدی جواب کمترین نرم اقلیدسی را دارد. (این مسئله به SGD's Implicit Regularization مشهور است.)

الف

$$S = \{(x_i, y_i) | x_i \in R^d, y_i \in \{۰, ۱\}\} \text{ و } |S| = n$$

طبق فرضیات مسئله داریم:

$$X=\begin{bmatrix} & \\ & \end{bmatrix}_{n\times d} \quad W=\begin{bmatrix} \\ \end{bmatrix}_{d\times 1} \quad Y=\begin{bmatrix} \\ \end{bmatrix}_{n\times 1} \quad Y^- = \begin{bmatrix} \\ \end{bmatrix}_{n\times 1}$$

می خواهیم تابع هزینه را صفر کنیم

$$J(w) = \frac{۱}{۲}||Xw - Y||_۲$$

بنابراین باید حاصل ضرب X در W (مثلا متغیر Y^-)برابر با Y باشد تا معادله ی مربوطه صفر شود

$$X=\begin{bmatrix} \dots x^1 \dots \\ \dots x^2 \dots \\ \dots x^3 \dots \\ \dots \\ \dots \\ \dots \\ \dots x^n \dots \end{bmatrix}_{n\times d} \quad W=\begin{bmatrix} W_0 \\ W_1 \\ W_2 \\ \dots \\ \dots \\ \dots \\ W_d \end{bmatrix}_{d\times 1}$$

برای محاسبه ی حاصل ضرب X در W داریم:

$$Y^- = X \times W = \begin{bmatrix} \dots x^1 \dots \\ \dots x^2 \dots \\ \dots x^3 \dots \\ \dots \\ \dots \\ \dots \\ \dots x^n \dots \end{bmatrix}_{n\times d} \times \begin{bmatrix} \dots W_0 \dots \\ \dots W_1 \dots \\ \dots W_2 \dots \\ \dots \\ \dots \\ \dots \\ \dots W_n \dots \end{bmatrix}_{d\times 1} = \begin{bmatrix} W_0 + x_1^1 W_1 + x_2^1 W_2 + \cdots + x_d^1 W_d \\ W_0 + x_1^2 W_1 + x_2^2 W_2 + \cdots + x_d^2 W_d \\ W_0 + x_1^3 W_1 + x_2^3 W_2 + \cdots + x_d^3 W_d \\ \dots \\ \dots \\ \dots \\ W_0 + x_1^n W_1 + x_2^n W_2 + \cdots + x_d^n W_d \end{bmatrix}_{n\times 1}$$

طبق فرضیات سوال می خواهیم:

$$Y^- - Y = 0$$

$$\begin{bmatrix} W_0 + x_1^1 W_1 + x_2^1 W_2 + \cdots + x_d^1 W_d \\ W_0 + x_1^2 W_1 + x_2^2 W_2 + \cdots + x_d^2 W_d \\ W_0 + x_1^3 W_1 + x_2^3 W_2 + \cdots + x_d^3 W_d \\ \dots \\ \dots \\ \dots \\ W_0 + x_1^n W_1 + x_2^n W_2 + \cdots + x_d^n W_d \end{bmatrix}_{n\times 1} - Y = 0$$

$$\begin{bmatrix} W_0 + x_1^1 W_1 + x_2^1 W_2 + \cdots + x_d^1 W_d \\ W_0 + x_1^2 W_1 + x_2^2 W_2 + \cdots + x_d^2 W_d \\ W_0 + x_1^3 W_1 + x_2^3 W_2 + \cdots + x_d^3 W_d \\ \cdots \\ \cdots \\ \cdots \\ W_0 + x_1^n W_1 + x_2^n W_2 + \cdots + x_d^n W_d \end{bmatrix}_{n\times 1} - Y = 0 \Rightarrow \begin{bmatrix} W_0 + x_1^1 W_1 + x_2^1 W_2 + \cdots + x_d^1 W_d \\ W_0 + x_1^2 W_1 + x_2^2 W_2 + \cdots + x_d^2 W_d \\ W_0 + x_1^3 W_1 + x_2^3 W_2 + \cdots + x_d^3 W_d \\ \cdots \\ \cdots \\ \cdots \\ W_0 + x_1^n W_1 + x_2^n W_2 + \cdots + x_d^n W_d \end{bmatrix}_{n\times 1} - \begin{bmatrix} Y_1 \\ Y_2 \\ Y_3 \\ \cdots \\ \cdots \\ \cdots \\ Y_n \end{bmatrix} = 0$$

$$Y^- - Y = XW - Y = \begin{cases} W_0 + x_1^1 W_1 + x_2^1 W_2 + \cdots + x_d^1 W_d - Y_1 = 0 \\ W_0 + x_1^2 W_1 + x_2^2 W_2 + \cdots + x_d^2 W_d - Y_2 = 0 \\ W_0 + x_1^3 W_1 + x_2^3 W_2 + \cdots + x_d^3 W_d - Y_3 = 0 \\ \cdots \\ \cdots \\ \cdots \\ W_0 + x_1^n W_1 + x_2^n W_2 + \cdots + x_d^n W_d - Y_n = 0 \end{cases}$$

بنابراین ما به یک مجموعه معادلات رسیدیم که در این معادلات ضرایب w مجهولات مسئله ما هستند، برای حل این معادلات (n) می دانیم، اگر تعداد این معادلات از تعداد مجهولات (d) بیشتر شود بی شمار جواب خواهیم داشت، اگر تعداد این معادلات (n) کمتر از تعداد مجهولات (n) شود آنگاه هیچ جوابی نخواهیم داشت، در نهایت اگر تعداد معادلات برابر با تعداد مجهولات شود تنها یک جواب خواهیم داشت.

ب

طبق قضیه ی **The representer theorem and kernel methods** داریم:

طبق این قضیه ما می دانیم که فضای مسئله را ما به روش خطی می توانیم به فضای داده ها محدود کنیم یعنی حتی اگر فضای مسئله (d) بسیار بزرگ و یا بی نهایت باشد بازهم می توانیم به یک فضای n بعدی محدود کنیم.این مسئله بر می گردد به این قضیه که پاسخ های موجود در فضای فضای داده ها برای مسائل بهینه سازی، بهینه هستند. بنابراین ما در ادامه اثبات خواهیم کرد که بهترین مدل برای یک مسئله مبتنی بر فضای داده های آموزشی است.

طبق قضیه ی ERM (empirical risk minimization) داریم:

$$\text{minimize} \frac{1}{n} \sum_{i=1}^{n} loss(w^T x_i, y_i) + \lambda \|w\|^2$$

همانطور که می دانیم جمله منظم ساز به جهت افزودن برخی اطلاعات قبلی به یک مسئله بهینه سازی برای منحصر به فرد کردن راه حل بهینه است. در واقع اطلاعات قبلی به ما کمک می کند که W ای با نرم کوچکتر را ترجیح دهیم و این مسئله باعث می شود که ما پاسخ های ساده تری را از فضای مسئله (ویژگی ها) انتخاب کنیم. بنابراین جمله منظم ساز به ما اجازه می دهد تا نرم پاسخ بهینه را تنظیم کنیم.

- اگر مقدار $\lambda=0$ باشد آنگاه ما بی شمار W خواهیم داشت که مسئله ERM را مینیمم کند.
- اگر مقدار $\lambda>0$ باشد آنگاه ما یک W یکتا خواهیم داشت که مسئله ERM را مینیمم کند.

$$\text{minimize} \frac{1}{n} \sum_{i=1}^{n} loss(w^T x_i, y_i) + \lambda \|w\|^2$$

با توجه با توجه به اینکه W یک فرم خطی دارد داریم: $w = X^T \beta + v$

با جای گذاری در معادله ی بالا خواهیم داشت

$$\text{minimize}_{\beta,v} \frac{1}{n} \sum_{i=1}^{n} loss(\beta^T X x_i, y_i) + \lambda \|X^T \beta\|^2 + \lambda \|v\|^2 .$$

حال می توانیم رابطه ی بالا را نسبت به V مینیمم کنیم. برای اینکار می توانیم مقدار V=0 در نظر بگیریم. بنابریان خواهیم داشت:

$$w = X^T \beta .$$

این مسئله را تئوری representer در یادگیری ماشین می نامند. به خاطر اینکه تابع هزینه فقط به $f(x_i) = w^T x_i$ بستگی دارد و هزینه با تابع ||W|| افزایش می یابد. بنابراین فضای مسئله ERM مبتنی به فضای داده ها می شود.

تابع کرنل مقابل را تعریف می کنیم:

$$K = XX^T$$

با جایگذاری تابع کرنل تعریف شده در معادله ی زیر خواهیم داشت:

$$\text{minimize}_{\beta,v} \frac{1}{n} \sum_{i=1}^{n} loss(\beta^T X x_i, y_i) + \lambda \|X^T \beta\|^2 \quad \Rightarrow \quad \text{minimize}_{\beta} \frac{1}{n} \sum_{i=1}^{n} loss(e_i^T K \beta, y_i) + \lambda \beta^T K \beta$$

در این رابطه e_i بردار بایاس اقلیدسی استاندارد است

همانطور که از رابطه ی بالا مشخص است برای حل مسائل یادگیری ماشین ما از پارامتر های موجود در K فقط استفاده می کنیم، بنابراین طبق تئوری representer (تکنیک کرنل) اغلب مسائل یادگیری ماشین به یک فضای سرچ n بعدی کاهش می یابد (حتی اگر فضای مسئله خیییللییی بزرگ باشد). بنابراین کل فضای مسئله بهینه سازی به ضرب نقطه ای بین دو نقطه (dot product) محدود می شود(طبق کرنل بالا). بنابراین می توان نتیجه گرفت مقدار ابعاد جواب در بدترین حالت برابر با n است و یا حتی کمتر است.

$$\dim(W) \leq n$$

ج

طبق معادله ی SGD داریم: $w_{t+1} = w_t - \alpha e_t x_t$

جاییکه e_t گرادیان هزینه در current prediction است. اگر $W_0 = 0$ باشد آنگاه W_t همیشه در فضای داده های مسئله خواهد بود. این مسئله ثابت می کند که حتی اگر وزن های عمومی در ابعاد بالایی قرار داشته باشند، SGDدر فضایی با ابعاد حداکثر n، تعداد نقاط داده جستجو می کند.

فرض می کنیم یک loss نامنفی داریم به طوریکه $\frac{\partial loss(z,y)}{\partial z} = 0$ اگر و فقط اگر $y = z$ باشد. و این شرط برای square loss برقرار است نه برای hinge و logistic losses. برای چنین lossهایی در حالت بهینه داریم:

1. $Xw = y$ → زیرا ما loss صفر نیز داریم

2. $w = X^T v$ → زیرا ما در فضای داده هستیم

با این فرض که نمونه های ما از نظر خطی مستقل هستند ، می توانیم این معادلات را با هم ترکیب کنیم و به عبارت زیر برسیم:

$$w = X^T (X X^T)^{-1} y$$

در واقع وقتی ما از stochastic gradient descent استفاده کنیم ، به یک راه حل بسیار خاص همگرایی می شویم، حتی اگر ما در حال جستجو در یک فضای nبعدی هستیم ، به یک نقطه منحصر به فرد در این فضا همگرایی می شویم.

$$w = X^T(XX^T)^{-1}y$$

این W در واقع پاسخ مینیمم نرم اقلیدسی تابع $Wx = y$ است. به عبارت دیگر ، از بین تمام توابع پیش بینی خطی که از داده های آموزشی بدست می آید، SGD راه حلی را با حداقل نرم اقلیدسی انتخاب می کند.

برای اثبات این موضوع فرض کنید داریم:

$$\hat{w} = X^T\alpha + v$$

که در آن V یک بردارorthogonal است

بنابراین داریم:

$$X\hat{w} = XX^T\alpha + Xv = XX^T\alpha$$

این به این معنی است که α کاملا معین است، بنابراین داریم:

$$\hat{w} = X^T(XX^T)^{-1}y + v.$$

بنابراین برای مینیمم کردن سمت راست نشان می دهد که v باید برابر با صفر باشد

بنابراین داریم

$$\|\hat{w}\|^2 = \|X^T(XX^T)^{-1}y\|^2 + \|v\|^2$$

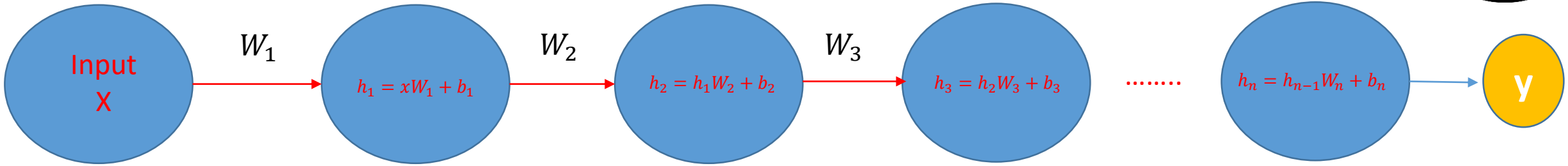
$$V = 0$$

بنابراین می توان ادعا کرد که W بدست آمده از میان فضای n بعدی کمترین نرم اقلیدسی را دارد.

Ref: http://Mlstory.org optimazation بخش

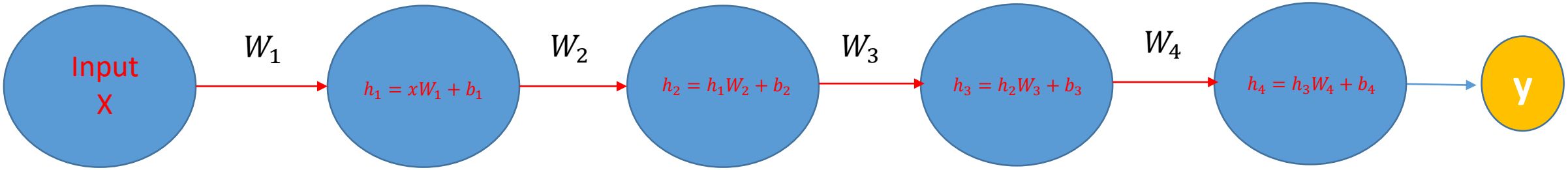
۳. (۱۵ نمره) در حالت کلی نشان‌دهید خروجی هر شبکه‌ی عصبی با توابع فعال‌سازی خطی (و m لایه‌ی مخفی) را می‌توان با شبکه‌ای عصبی دیگر بدون لایه‌ی مخفی تشکیل‌داد.

فرض می کنیم که شبکه ی زیر دارای تعدادی توابع فعالیت خطی در لایه های مختلفش است:

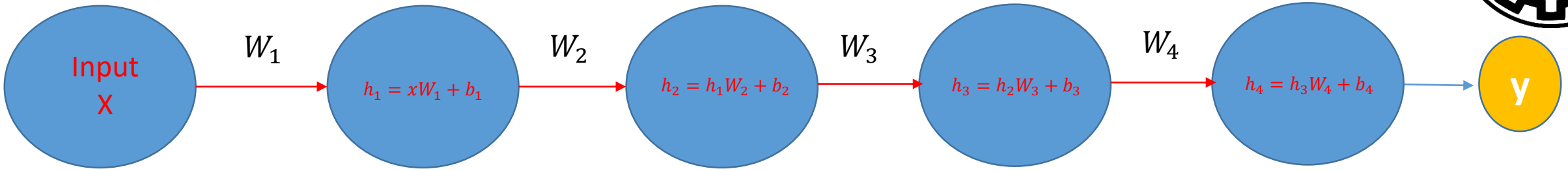


همانظور که مشاهده می شود در شبکه های عصبی با توابع فعالیت خطی و در مرحله ی feed forward ، هر لایه ی مخفی برابر است با یک شبکه ی عصبی بدون لایه ی مخفی

اگر شبکه ی زیر را داشته باشیم:



اگر شبکه ی زیر را داشته باشیم:



می توان نوشت:

$$y = h_3W_4 + b_4$$
$$= (h_2W_3 + b_3)W_4 + b_4 = h_2W_3W_4 + \mathrm{b_3W_4} + \mathrm{b_4}$$
$$= (h_1W_2 + b_2)W_3W_4 + \mathrm{b_3W_4} + \mathrm{b_4} = h_1W_2W_3W_4 + b_2W_3W_4 + \mathrm{b_3W_4} + \mathrm{b_4}$$
$$= (xW_1 + b_1)W_2W_3W_4 + b_2W_3W_4 + \mathrm{b_3W_4} + \mathrm{b_4} = xW_1W_2W_3W_4 + b_1W_2W_3W_4 + b_2W_3W_4 + \mathrm{b_3W_4} + \mathrm{b_4}$$

بنابراین همانطور که در رابطه ی زیر می بینید می توان یک شبکه با لایه های مخفی را را که از توابع فعالیت خطی تشکیل شده است با یک شبکه بدون لایه ی مخفی پیاده سازی کرد. زیرا می توان چندین transformation خطی را با یک transformation خطی مدل کرد و ترکیب bias ها را هم با یک bias مدل کرد.

$$y = xW_1W_2W_3W_4 + b_1W_2W_3W_4 + b_2W_3W_4 + \mathrm{b_3W_4} + \mathrm{b_4} \Rightarrow Y = W' + b'$$

$$y = xW_1W_2W_3W_4 + b_1W_2W_3W_4 + b_2W_3W_4 + \mathrm{b}_3\mathrm{W}_4 + \mathrm{b}_4 \Rightarrow Y = W' + b'$$

بنابریان هر چه قدر یک شبکه تعداد لایه های مخفی بیشتری داشته باشید و توابع فعالیت خطی استفاده کند بازهم می توان با یک شبکه بدون لایه ی مخفی مدل کرد. بنابراین افزایش تعداد لایه های مخفی شبکه با توابع فعالیت مخفی منجر به افزایش قدرت شبکه های عصبی نمی شود.

برای یک شبکه با n لایه ی و توابع فعالیت خطی ⇒ $y = xW_1W_2W_3 \dots W_n + b_1W_2W_3 \dots W_n + b_2W_{3\dots}W_n + \cdots + \mathrm{b}_{\mathrm{n}-1}\mathrm{W}_\mathrm{n} + \mathrm{b}_\mathrm{n}$

Ref: https://stackoverflow.com/questions/9782071/why-must-a-nonlinear-activation-function-be-used-in-a-backpropagation-neural-net

۴. (۱۵ نمره) می‌دانید که هر رابطه‌ی منطقی را می‌توان به کمک شیوه‌های گوناگون چون جدول کارنو به قالب جمعی از ضرب‌ها یا Sum of Products نوشت، با توجه به این موضوع به سوالات زیر پاسخ دهید.

(آ) (۵ نمره) نشان‌دهید که چگونه می‌توان هر رابطه‌ی منطقی را با یک شبکه‌ی عصبی ۲ لایه (یک لایه‌ی مخفی) و تابع فعال‌ساز $ReLU$ نمایش‌داد.

(ب) (۵ نمره) یک شبکه‌ی عصبی دو لایه (یک لایه‌ی مخفی) با تعداد پارامترهایی از مرتبه‌ی ۲^n برای نمایش تابع XOR با n ورودی پیشنهاد دهید.

(ج) (۵ نمره) می‌توان با افزایش عمق شبکه، تعداد پارامترهای لازم برای نمایش تابع هدف را کاهش‌داد. بدین منظور یک شبکه‌عصبی برای نمایش تابع XOR با n ورودی و تعداد پارامترهایی از مرتبه‌ی n و لایه‌هایی از مرتبه‌ی $log\ n$ پیشنهاد دهید.

الف

می دانیم که عبارت های بولی را می توانیم با جدول درستی مدل کنیم همانطور که از جدول مقابل مشخص است:

جدول درستی در واقع نشان می دهد که چه ترکیبی از ورودی ها منجر به خروجی یک می شود:

می دانیم که تابع ReLU خروجی صفر را خنثی می کند بنابراین می توانیم شبکه را بر اساس خروجی های یک بسازیم همانطور که در زیر می‌بینیم:

X1	X2	X3	X4	X5	Y
0	0	1	1	0	1
0	1	0	1	1	1
0	1	1	0	0	1
1	0	0	0	1	1
1	0	1	1	1	1
1	1	0	0	1	1

جدول درستی

X1	X2	X3	X4	X5	Y
0	0	1	1	0	1
0	1	0	1	1	1
0	1	1	0	0	1
1	0	0	0	1	1
1	0	1	1	1	1
1	1	0	0	1	1
1	1	1	1	0	0
0	0	0	0	0	0
1	1	1	0	0	0
1	1	0	1	1	0

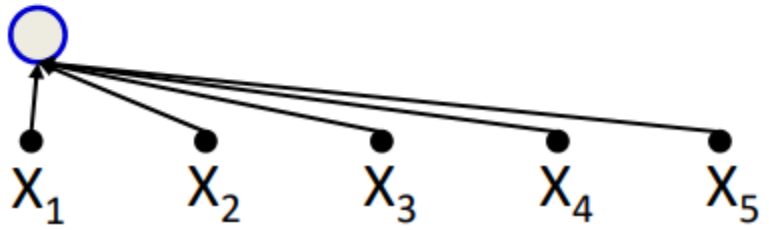
X1	X2	X3	X4	X5	Y
0	0	1	1	0	1
0	1	0	1	1	1
0	1	1	0	0	1
1	0	0	0	1	1
1	0	1	1	1	1
1	1	0	0	1	1

$$Y = \bar{X}_1\bar{X}_2X_3X_4\bar{X}_5 + \bar{X}_1X_2\bar{X}_3X_4X_5 + \bar{X}_1X_2X_3\bar{X}_4\bar{X}_5 + X_1\bar{X}_2\bar{X}_3\bar{X}_4X_5 + X_1\bar{X}_2X_3X_4X_5 + X_1X_2\bar{X}_3\bar{X}_4X_5$$

هرکدام از سطرها را می توان توسط یک نورون و AND مدل کرد(برای X ها یک و برای X^- منفی یک در نظر می گیریم):

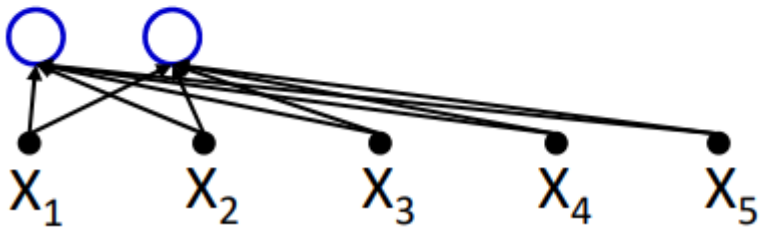
1

$$Y = \bar{X}_1\bar{X}_2X_3X_4\bar{X}_5 + \bar{X}_1X_2\bar{X}_3X_4X_5 + \bar{X}_1X_2X_3\bar{X}_4\bar{X}_5 + X_1\bar{X}_2\bar{X}_3\bar{X}_4X_5 + X_1\bar{X}_2X_3X_4X_5 + X_1X_2\bar{X}_3\bar{X}_4X_5$$



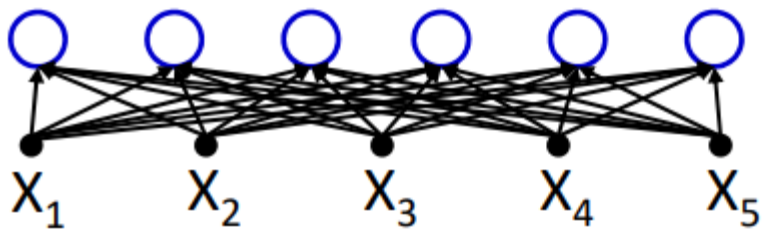
2

$$Y = \bar{X}_1\bar{X}_2X_3X_4\bar{X}_5 + \bar{X}_1X_2\bar{X}_3X_4X_5 + \bar{X}_1X_2X_3\bar{X}_4\bar{X}_5 + X_1\bar{X}_2\bar{X}_3\bar{X}_4X_5 + X_1\bar{X}_2X_3X_4X_5 + X_1X_2\bar{X}_3\bar{X}_4X_5$$

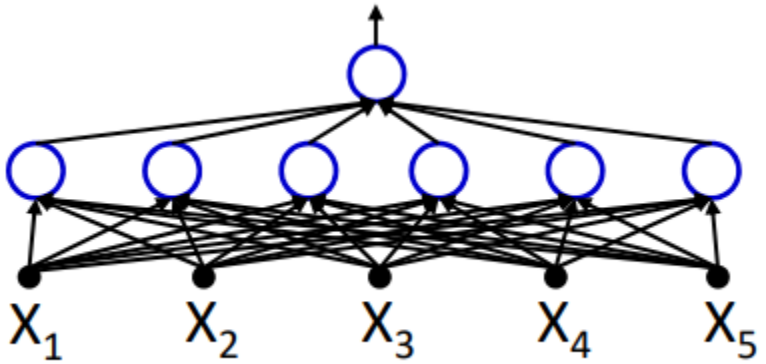


و در نهایت

$$Y = \bar{X}_1\bar{X}_2X_3X_4\bar{X}_5 + \bar{X}_1X_2\bar{X}_3X_4X_5 + \bar{X}_1X_2X_3\bar{X}_4\bar{X}_5 + X_1\bar{X}_2\bar{X}_3\bar{X}_4X_5 + X_1\bar{X}_2X_3X_4X_5 + X_1X_2\bar{X}_3\bar{X}_4X_5$$



همچنین با استفاده از نورون OR در آخرین لایه می توانیم شبکه روابط بین سطرها را به طور کامل مدل کنیم:

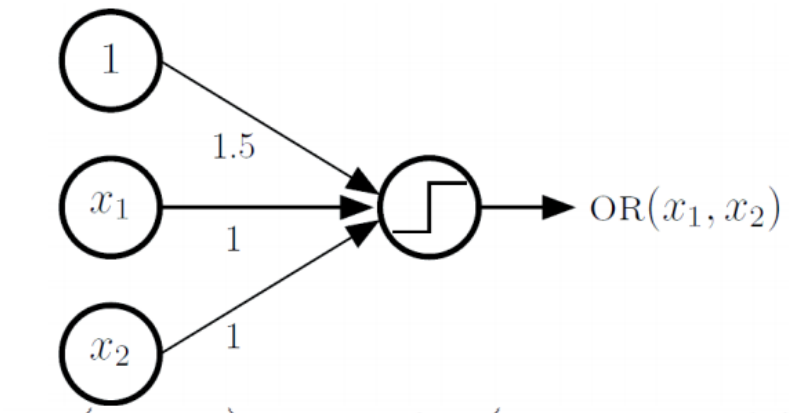


بنابراین دیدیم که هر رابطه ی منطقی را با استفاده از یک شبکه ی دو لایه (یک لایه مخفی) می توان مدل کرد.

ب

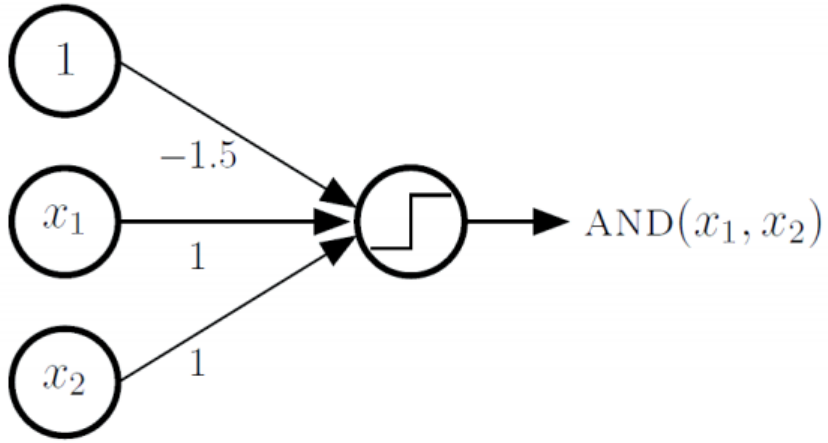
می دانیم که عملیات XOR را می توان با دو عملیات OR و AND انجام داد، همانطور که در شکل زیر می بینیم ما برای محاسبه AND و OR شبکه های را می توانیم طراحی کنیم:

OR



$$\mathrm{OR}(x_1, x_2) = \mathrm{sign}(x_1 + x_2 + 1.5)$$

AND

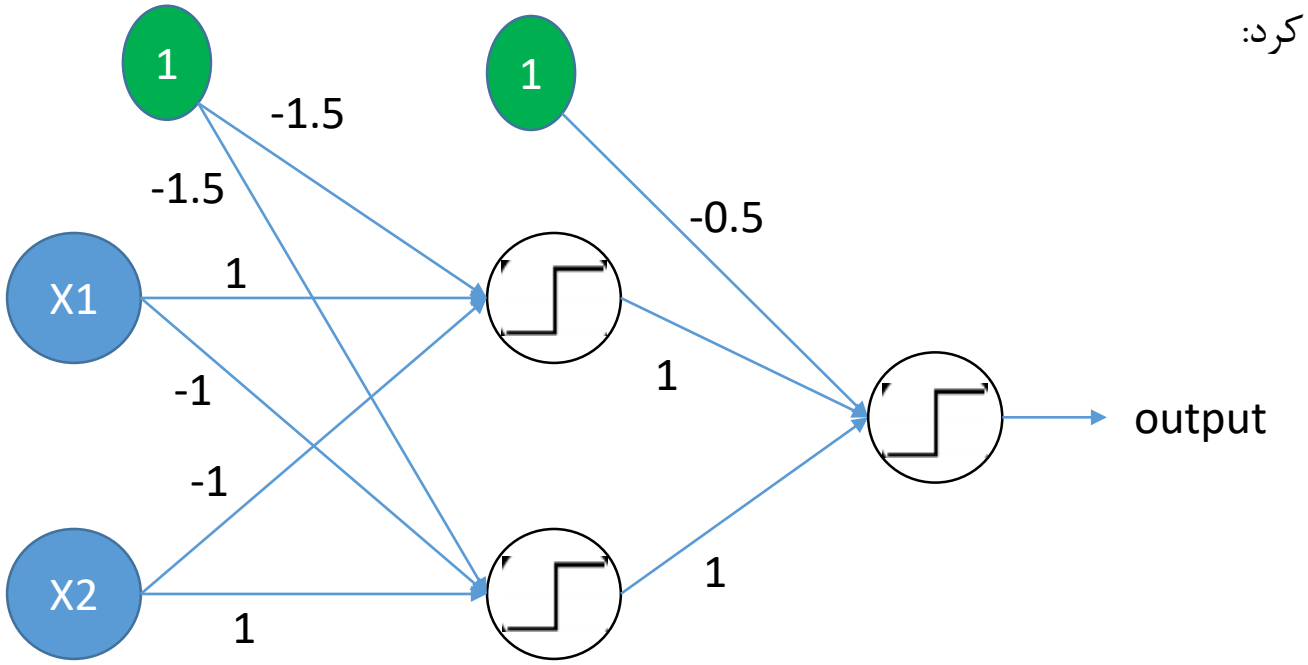


$$\mathrm{AND}(x_1, x_2) = \mathrm{sign}(x_1 + x_2 - 1.5)$$

همانطور که گفته شد می توانیم عملیات XOR را با ترکیب عملیات OR وAND انجام دهیم و فرم SOP آنرا بنویسیم:

$$SOP = XOR(X_1, X_2) = OR(AND(X_1, X_2^-), AND(X_1^-, X_2\ ))$$

بنابراین XOR را می توان به صورت زیر مدل کرد:



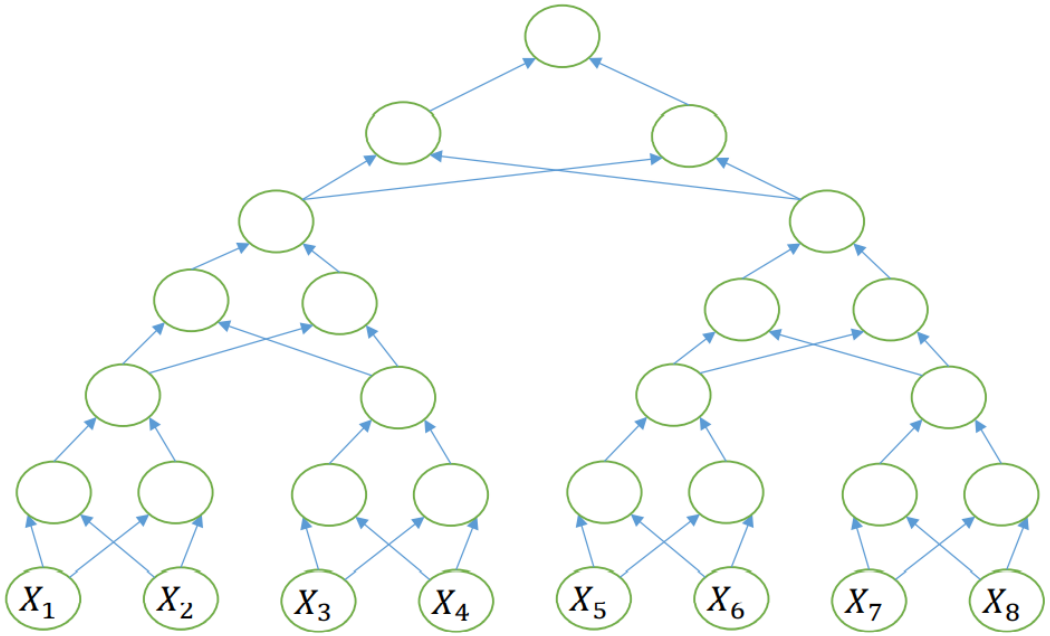
ج

$$f = X_1 \oplus X_2 \oplus \cdots \oplus X_N$$

می دانیم تابع XOR قابل شکسته شدن و توزیع پذیر است بنابریان می توانیم تابع مقابل را به صورت زیر بازنویسی کنیم:

- $f = \left((X_1 \oplus X_2) \oplus (X_3 \oplus X_4)\right) \oplus \left((X_4 \oplus X_5) \oplus (X_6 \oplus X_7)\right)$

بنابراین می توان شبکه ای به صورت زیر طراحی کرد:



همانطور از شبکه ی رو به رو مشخص است شبکه به یک درخت باینرس است بنابراین تعداد پارامترهایی به مرتبه n و تعداد لایه هایی به مرتبه $\log^n$ دارند.

بخش MLP

Ref: اسلایدهای دکتر سلیمانی

Neural Networks: What can a network represent

۵. (۱۵ نمره) مزایا و معایب استفاده از هر یک از توابع فعال سازی sigmoid و tanh و ReLU را بیان کنید و این توابع را از نظر هزینه محاسباتی و موضوع vanishing gradient بررسی کنید.

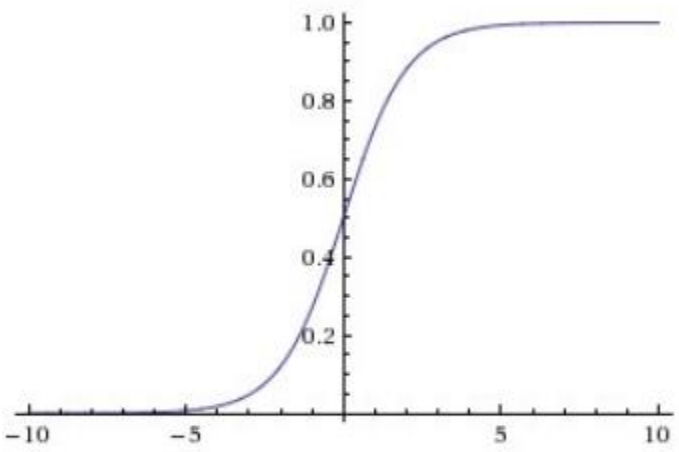
مزایا

- نگاشت اعداد به بازه ی [0,1]
- آموزش آسان روی مجموعه داده های کوچک
- درک و استفاده ی آسان
- در همه ی نقاط مشتق پذیر است بنابراین سازگار با عملیات Backpropagation است.
- خود تابع Monotonic است.
- این تابع nonlinear است بنابراین برای active hidden layerها مناسب است.

معایب

- مشتق تابع Monotonic نیست.
- نورون های اشباع شده، گرادیان را از بین می برند.
- خروجی های تابع سیگموید دارای میانگین صفر نیستند.
- توان رسانی، عمل نسبتاً پر هزینه ای است.
- اگر در یک دسته بند به عنوان آخرین activation استفاده شود حمع همه ی کلاس های لزوما یک نمی شود.

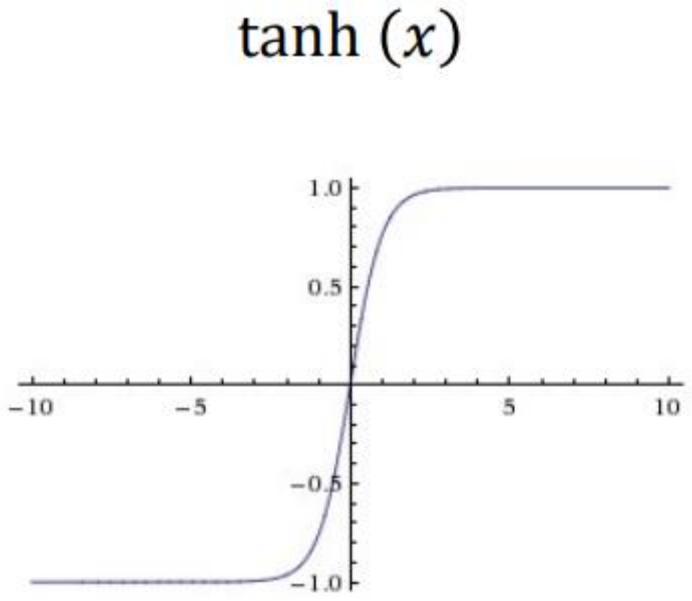
$$\sigma(x) = 1/(1 + e^{-x})$$



تابع سیگموید

مزایا

- نگاشت اعداد به بازه [-1,1]
- دارای میانگین صفر
- گرادیان در Tanh قوی تر از سیگموید است.
- در مقایسه با سیگموئید ، راه حل بهینه تری ارائه می دهد.
- مشتق پذیر است.
- این تابع Monotonic است.



تانژانت هایپربولیک

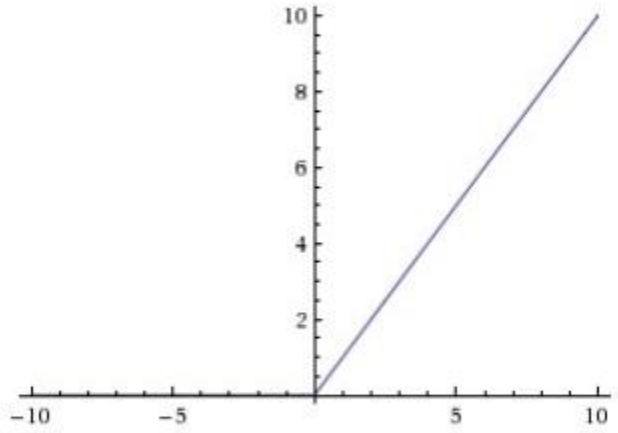
معایب

- به صورت معمول برای دسته بندی بین دو کلاس مورد استفاده قرار می گیرد.
- هنوز نورون های اشباع شده، گرادیان را از بین می برند.
- مشتق این تابع Monotonic است.

مزایا

- اشباع نمی شود (در نواحی مثبت) گرادیان از بین نمی رود.
- از نظر محاسباتی بسیار کارا است.
- در عمل، بسیار سریع تر از توابع سیگموید و تانژانت هایپربولیک همگرا می شود. (مثلا ۶ برابر).
- در واقع از نظر biologically قابل قبول تر ازسیگموئید.
- Activation را کم و کارآمد می کند.

$$\max(0, x)$$



ReLU

معایب

- دارای میانگین صفر نیست.
- بخش Dead ReLU (نرون مرده) که در واقع هیچگاه active نمی شود و به روز رسانی نمی شود.

Ref: اسلایدهای دکتر سلیمانی، اسلایدهای سید ناصر رضوی، گوگل، یک مقاله

Training Neural Networks
Practical Issues

۶. (۱۵ نمره) قصد داریم شبکه ای طراحی کنیم که با ورودی گرفتن سه عدد صحیح، تعیین کند که کدام یک بزرگ‌تر است. اگر x_1 ، x_2 و x_3 سه ورودی و y_1 ، y_2 و y_3 سه خروجی شبکه‌باشند، ضمن طراحی شبکه‌ای با عمق و عرض دلخواه و تابع فعال سازی ReLU برای این کار، وزن‌ها و بایاس‌های شبکه را به گونه‌ای تعیین کنید که اگر x_i بزرگ‌ترین عدد ورودی باشد، مقدار y_i عددی مثبت و در غیر این صورت برابر صفر شود. همچنین در حالتی که بیش از یک عدد بیشینه باشد انتظار داریم مقدار خروجی متناظر با هر بیشینه مقداری مثبت شود.

برای تبدیل عملیات ماکسیمم بین دو عدد به فرم عملیات ضرب و جمع ساده داریم:

$$\frac{a+b}{2}+\frac{|a-b|}{2} \Rightarrow \max(a=2,b=3)=\frac{2+3}{2}+\frac{|2-3|}{2}=3$$

با توجه به فرضیات صورت سوال، برای بدست آوردن نتیجه(بزرگترین عدد از بین سه عدد) قرار بر این شده که عدد بزرگتر به شکل یک عدد مثبت و دیگر اعداد در خروجی شوند.

برای تابع relu داریم:

$$\max(0,x)$$

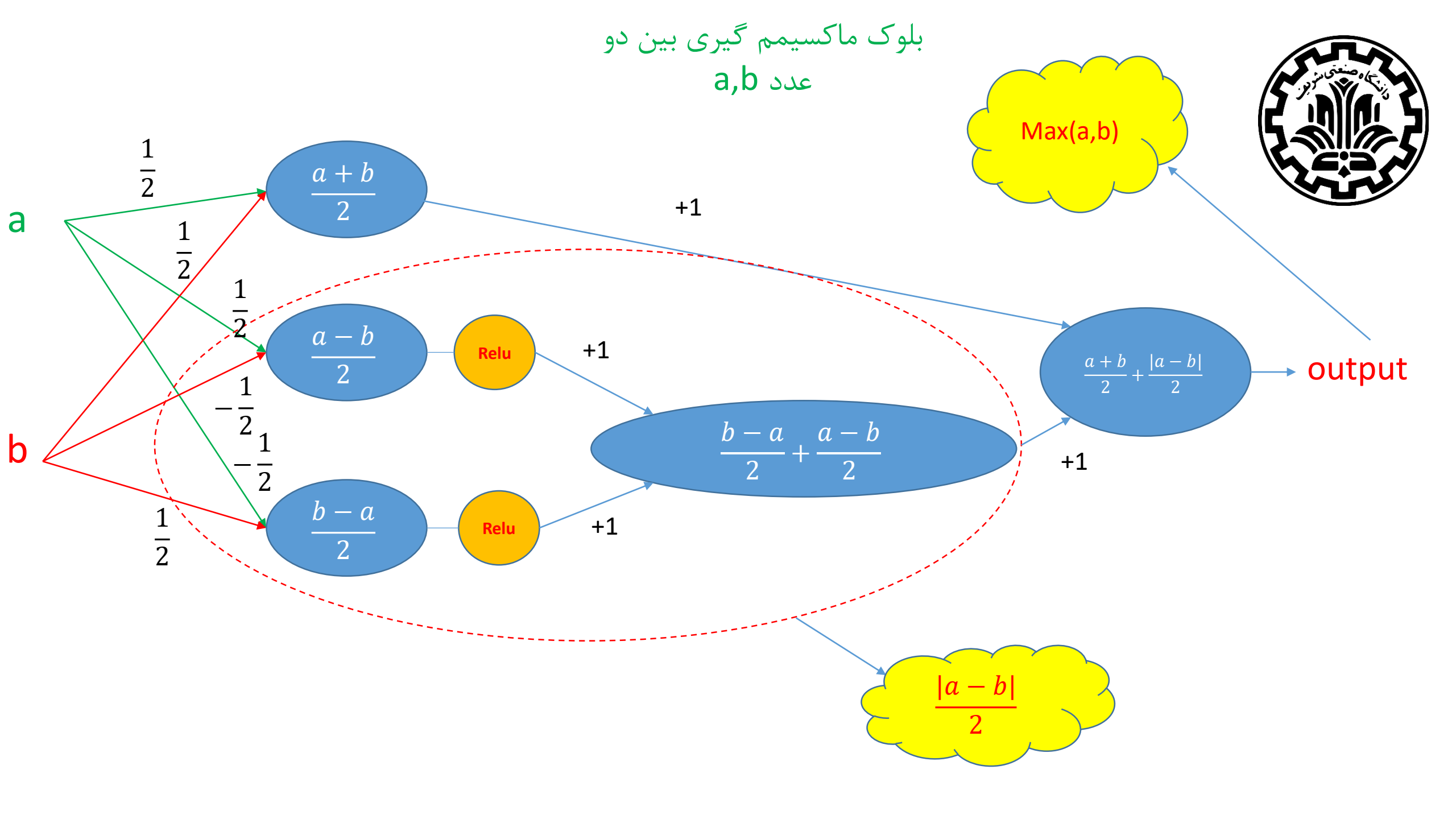
برای بخش تابع RELU می توانیم اینطور در نظر بگیریم که اگر مقدار ماکسیمم سه عدد را از هر یک از اعداد کم کنیم عدد ماکسیمم صفر می شود و دو عدد دیگر منفی بنابراین اگر دو عدد منفی از تابع RELU عبور کنند نتیجه صفر خواهد شد اما برای بزرگترین عدد میخواهیم یک عدد مثبت به عنوان خروجی تولید شود برای همین علاوه بر کم کردن بزرگترین عدد از سه عددی مقدار ناچیز به تمامی اعداد اضافه می کنم که باعث می شود اعداد منفی، منفی باقی بمانند و عدد صفر یک عدد بزرگتر از صفر شود و در این صورت خروجی RELU نیز یک عدد مثبت خواهد شد.

a-max(a,b,c)+λ

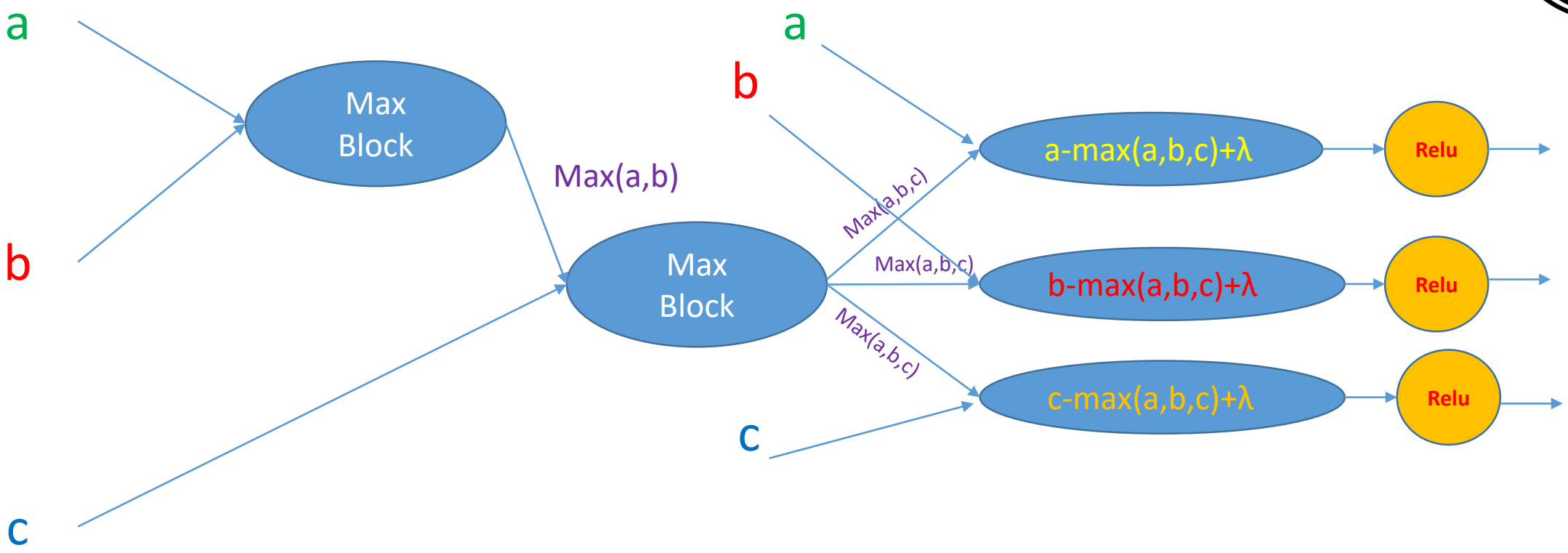
b-max(a,b,c)+λ

c-max(a,b,c)+λ

بلوک ماکسیمم گیری بین دو عدد a,b



پیاده سازی عملیات ماکسیمم گیری بین سه متغیر a، b وc که از بلوک مربطو به عملیان ماکس بین دو متغیر استفاده کردیم:



بنابراین با توجه به شبکه ای که طراحی کردیم به ازای بزرگترین عدد یک مثبت بر می گردد و به ازای دو عدد دیگر صفر بر می گردد.

همکاری با دوستان

۷. (۱۰ نمره) توضیح‌دهید که چگونه می‌توان L_2 Decay را تحت بهینه‌سازی SGD برابر با Weight Decay تصور کرد. (راهنمایی)

$L2\ Regularization$ و *Weight Decay* در عمل برابر نیستند ولی می توانند آن ها را با استفاده از *SGD* و تنظیم مجدد پارامترهای weight decay بر اساس learning rate برابر در نظر گرفت. معادله weight decay در شبکه های عصبی در زیر آورده شده است که λ عامل decay است.

$$w = (1\text{-}\lambda)w - \alpha\Delta C_0$$

همچنین برای معادله ی L2 Regularization داریم:

$$C = C_0 + \frac{\lambda}{2}\|w\|_2^2$$

بنابراین ما می خواهیم معادله ی بالا را به فرم معادله ی weight decay تبدیل کنیم برای این منظور ابتدا گرادیان معادله L2 Regularization نسبت به W محاسبه می کنیم:

$$\frac{\partial C}{\partial w} = \frac{\partial C_0}{\partial w} + 2\frac{\lambda w}{2}$$

همچنین داریم:

$$\Delta C == \frac{\partial C}{\partial w}$$

پس از محاسبه ی گرادیان معادله ی L2 Regularization نسبت به W و جایگذاری در معادله ی Weight decay داریم:

$$w = w - \alpha \Delta C$$
$$w = w - \alpha(\Delta C_0 + \lambda w)$$
$$w = w - \alpha \Delta C_0 - \alpha \lambda w$$
$$w = (1 - \alpha\lambda)w - \alpha \Delta C_0$$

معادله‌ی Weight Decay

معادله ی L2 Regularization

$$w = (1-\lambda)w - \alpha \Delta C_0$$

$$w = (1 - \alpha\lambda)w - \alpha \Delta C_0$$

همانطور که میبینید تفاوت بین معادله ی L2 Regularization و Weight Decay در ضریب α ضرب شده در λ است. بنابراین با جایگذاری در رابطه ی زیر در معادله یL2 Regularization داریم:

$$\lambda = \frac{\lambda'}{\alpha} \Rightarrow w = (1 - \lambda')w - \alpha \Delta C_0$$

بنابراین ثابت شد که تحت SGD، weight Decay و R2 Regularization برابر است.

Ref:https://towardsdatascience.com/weight-decay-l2-regularization-90a9e17713cd

۸. (۱۵ نمره امتیازی) امروزه شبکه‌های کانولوشنال برای انواع و اقسام مسائل مربوط به بینایی ماشین به صورت همه‌جانبه استفاده شده و همه‌جا وجود دارند. این در حالیست که درک دقیقی از آنکه چگونه در مورد تصاویر ورودی خود تصمیم گیری‌کرده و برای تخمین‌های خود به کدام بخش از تصویر توجه می‌کنند وجود ندارد. هرچند مصورسازی فیچرهای تشخیص‌داده‌شده در لایه‌ی نخستین آنها ممکن است، اعمال این‌کار برای لایه‌های میانی شبکه سود چندانی نداشته و به ادراک عمل‌کرد آنها کمک نمی‌کند. بدین منظور ابزارهای گوناگونی برای بررسی عمل‌کرد درونی این شبکه‌ها پیشنهادشده و به کار گرفته می‌شود. به انتخاب خود از میان مقالات زیر دو ابزار را انتخاب کرده و راه‌کار پیشنهادی آنها را به صورت خلاصه توصیف‌کنید.

Deconvolution

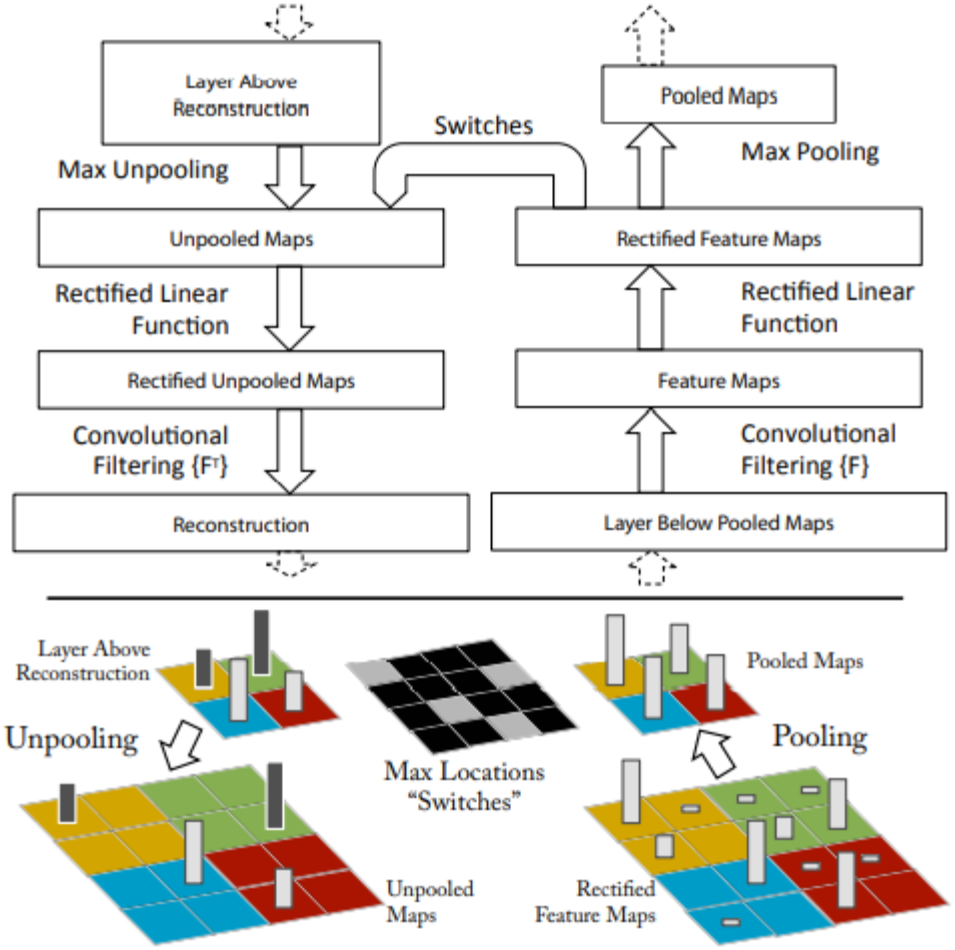
- مقاله اول

شبکه های کانوولوشنی در سالهای اخیر رشد قابل توجهی در حل مسائل گوناگون داشته اند، به طوری که در دسته بندی مجموعه ی دادگان ImageNet ، این شبکه ها کارآیی قابل ملاحظه ای از خود نشان داده اند. اما پیوسته یکی از مسائل مطرح شده در حوزه ی شبکه های کانوولوشنی می توان به عدم فهم دقیق جزئیات ساختاری شبکه (ویژگی هایی که کمک کننده هستند و کارآیی دارند) اشاره کرد. همچنین این مشکل همواره وجود داشته است که دقیقا نمیدانیم با تغییر چه بخشی ممکن است کارآیی بهبود یابد. در این مقاله به بررسی و حل این مشکل پرداخته شده است. در این مقاله به معرفی یک تکنیک Visualization با استفاده deconvolution می پردازیم که دید بهتری نسبت به ویژگی های ساختاری شبکه و لایه های آن و operation های شبکه می دهد. تکنیک های Visualization به ما کمک می کند که مدل ها و معماری هایی را طراحی کنیم که نسبت از به بنچ مارک های موجود بر روی مجموعه دادگان مختلف از جمله ImageNet از پیشرفت قابل توجهی برخوردار باشند. در ابتدا برای بهبود عملکرد شبکه های کانوولوشنی ایده های ابتدایی از جمله افزایش مجموعه ی دادگان، استفاده از GPU برای پیاده سازی و اجرای شبکه های پیچیده تر و در نهایت معرفی مدل های تنظیم (Regularization) قدرتمند تر مثل dropout ارائه شد و تا حدی در بسیاری از موارد، عملکرد شبکه ها (افزایش دقت و کاهش خطا) را بهبود بخشید. اما این روش و تکنیک ها برای بسیاری از محققان قابل قبول نبود زیرا مبتنی بر آزمون و خطا بود. بنابراین در این مقاله به معرفی یک تکنیک Visualization پرداخته شده است که تاثیر و اهمیت ویژگی های مختلف (به صورت تکی) ورودی ها در لایه های مختلف را فاش می کند. همچنین این مسئه امکان بررسی و مشاهده ی ویژگی های را در طول مرحله ی آموزش و تشخیص مشکلات بالقوه مدل فراهم می کند. Visualization برای به دست آوردن شهود در مورد شبکه به عنوان یک مسئله مهم در بسیاری در کارها مورد بررسی قرار گرفته است، اما اغلب به لایه اول محدود می شود که در آن صرفا Projection به فضای پیکسل امکان پذیر است و این مسئله در لایه های بالاتر امکان پذیر نیست و به راحتی نمی توان آن را تفسیر کرد. این این مقاله بر این اساس است که فرض می کنیم که یک مدل داریم (یک شبکه کانوولوشنی و یک آموزش نظارت شده) که تصاویر دو بعدی را به به عنوان ورودی دریافت می کند و درنهایت لیبل مربوط به آن را مشخص می کند. در تکنیک معرفی شده در این مقاله، هر لایه در شبکه شامل یک لایه کاونولوشنی است که ورودی های لایه قبل را به عنوان ورودی دریافت می کند و با فیلتر های موجود کانوولو می کند. نتیجه کانوولوشن به یک تابع فعالیت داده می شود مثل (ReLU) و در نهایت به صورت اختیاری می تواند از یک لایه Max pooling عبور کند و یا یک عملیات نرمال کردن نیز داشته باشیم.

Deconvolution

حال می توانیم با استفاده از ایده و تکنیک جدید Deconvolution ویژگی های موجود در شبکه را Visualization کنیم به این صورت که نتایج شبکه که حاصل از عملیات کانوولوشن است و حاصل لایه ی ورودی تا لایه های کانوولوشنی و لایه های فعالیت و.... است را به صورت معکوس پیمایش کنیم به طوری که مجددا بتوانیم ورودی های مسئله را تولید کنیم (در واقعا ما در Deconvolution به دنبال Mapping از نتایج حاصل از لایه های مختلف به فضای ورودی یعنی پیکس های تصویر هستیم). می توان این گونه تصور که عملیات deconvolution همان convolution است اما این بار از تمامی اجزا و ساختار موجود در کانوولوشن به صورت معکوس استفاده می شود (زیرا ما به جای نگاشت تصویر به ویژگی ها به دنبال نگاشت ویژگی های به تصویر هستیم). بنابراین با توجه به ایده مطرح شده ما باید معماری کانوولوشنی طراحی کنیم که بتواند با ایجاد و طراحی یک مسیر مجددا به ورود های مسئله (مثلا پیکسل ها) بر گردد. برای شروع ورودی ما به یک شبکه کانوولوشنی داده می شود و پس ار عبور از لایه ها و محاسبات کانوولوشنی ویژگی ها استخراج می شود. برای ارزیابی convolutional activation تمامی activationهای موجود در لایه ها رو به صورت صفر تنظیم می کنیم و ویژگی های بدست اومده رو برای نگاشت به ورودی به معماری deconvolution که در کنار آن طراحی شده ارسال می کنیم. سپس به صورت متوالی عملیات های ۱) unpooling ۲)Rectification ۳)Filtering را انجام می دهمیم تا بتوانیم activity که زیر لایه activation که انتخاب کریم را بازسازی کنیم. و این کار تا زمانی که تکرار می کنیم که فضای پیکسل های ورودی را به صورت کامل بازسازی کنیم. عملیات unpooling قابل معکوس کردن نیست اما می توان این عملیات را به صورت تقریبی انجام داد(با استفاده از ثبت لوکیشنmaxima در هر ناحیه pooling با استفاده از یک سری متغیر switch). در مرحله ی deconvolution، عملیات unpooling از این متغیرهای سوئیچ استفاده می کند که لایه های باسازی شده بالایی را در لوکیشن های مربوط به آنها قرار دهد. در لایه ی Rectification همانند حالت convolutionما مقادیر بازسازی شده را از یک relu non-linearity عبور می دهیم. در Convolution فیلترها با ویژگی های مپ شده ی لایه ی قبلی کانوولو می شوند اما برای معکوس این عملیات در deconvolution ما ورژن transpose این فیلترها را در rectified maps ها کانوولو می کنیم نه خروجی لایه ی زیرین (به بیان دیگر به صورت افقی و عموری هر فیلتر را flip می کنیم). همه این موارد به لطف ایجاد یک سوییچ (مسیر) در کنار لایه ی max pooling در معماری convolution میسر شده است همانطور که در شکل زیر می بینیم.

Deconvolution



Guided Backpropagation

بسیاری از شبکه‌های عصبی کانولوشنِ مورد استفاده بر اساس یک ساختار معین ساخته شده اند (مثلا شبکه های مورد استفاده در تشخیص تصاویر): معماری این شبکه ها به این صورت که پس از یک سری لایه ی کانوولوشنی، لایه هایی مثل max pooling و drop out به یک لایه ی fully connected منتهی می شوند. در این مقاله به پیاده سازی یک روش جدید برای جایگزینی لایه max pooling با یک لایه کانولوشنی با استفاده از stride معرفی شده است.

Table 1: The three base networks used for classification on CIFAR-10 and CIFAR-100.

Model		
A	B	C
Input 32 × 32 RGB image		
5 × 5 conv. 96 RcLU	5 × 5 conv. 96 RcLU 1 × 1 conv. 96 ReLU	3 × 3 conv. 96 RcLU 3 × 3 conv. 96 ReLU
3 × 3 max-pooling stride 2		
5 × 5 conv. 192 ReLU	5 × 5 conv. 192 ReLU 1 × 1 conv. 192 ReLU	3 × 3 conv. 192 ReLU 3 × 3 conv. 192 ReLU
3 × 3 max-pooling stride 2		
3 × 3 conv. 192 ReLU		
1 × 1 conv. 192 ReLU		
1 × 1 conv. 10 ReLU		
global averaging over 6 × 6 spatial dimensions		
10 or 100-way softmax		

Guided Backpropagation

در این مقاله اثبات شده است که این مسئله بدون از دست دادن دقت میسر شده است. بنابراین کل ساختار ایجاد شده در این مقاله بر اساس ساختار کانوولوشن است. این معماری از چندین نظر با معماری های دیگری شبکه های عصبی متفاوت است که عبارت اند از:

۱) در این شبکه برای کاهش ابعاد به جای لایه ی pooling از لایه کانوولوشن با stride=2 استفاده شده است. استفاده از لایه ی pooling به منظور کاهش بعد و پارامترهای شبکه انجام می گیرد. برای کاهش ابعاد از دو روش استفاده می‌شود :

- حذف لایه ی pooling و استفاده از لایه ی کانوولوشنی قبلی برای ایفای نقش لایه pooling با استفاده از افزایش Stride کانوولوشن لایه‌ی قبل
- لایه کانوولشنی با stride بزرگتر از یک را جایگزین لایه pooling کنیم.

۲) استفاده از لایه‌های کانولوشن کوچک (با $k < 5$) که می‌تواند تعداد پارامترهای شبکه را تا حد زیادی کاهش دهد و به عنوان regularization مورد استفاده قرار می گیرد.

CIFAR-10 classification error

Model	Error (%)	# parameters
without data augmentation		
Model A	12.47%	≈ 0.9 M
Strided-CNN-A	13.46%	≈ 0.9 M
ConvPool-CNN-A	**10.21%**	≈ 1.28 M
ALL-CNN-A	10.30%	≈ 1.28 M
Model B	10.20%	≈ 1 M
Strided-CNN-B	10.98%	≈ 1 M
ConvPool-CNN-B	9.33%	≈ 1.35 M
ALL-CNN-B	**9.10%**	≈ 1.35 M
Model C	9.74%	≈ 1.3 M
Strided-CNN-C	10.19%	≈ 1.3 M
ConvPool-CNN-C	9.31%	≈ 1.4 M
ALL-CNN-C	**9.08%**	≈ 1.4 M