تمرین ۲ یادگیری ماشین برای بیوانفورماتیک

مهدی کافی ۹۹۲۱۰۷۵۳

①

الف) در ساده ترین حالت، محاسبه کانولوشن دو تصویر هر یک با ابعاد $N \times N$، مرتبه زمانی $O(N^2 \times N^2)$ برابر با $O(N^4)$ خواهد داشت.

ب) اگر بخواهیم دو چندجمله ای را در یکدیگر ضرب کنیم می توانیم آرایه ضرایب یکی از چندجمله ای ها را بر روی دیگری حرکت دهیم تا نتیجه محاسبه شود به این کار کانولوشن گفته می شود. روش دیگر این است که برای این محاسبه ابتدا چندجمله ای ها را به فضای دیگری ببریم و سپس ضرب را به صورت elementwise بین درایه های هر لیست انجام دهیم. به این فضا فضای فوریه است که می توان به صورت روبرو نشان داد: $FFT(f) \circ FFT(g)$ سپس باید دوباره این حاصل را به فضای اولیه برگردانیم که با عکس تبدیل فوریه امکان پذیر است. بنابراین برای ضرب چندجمله ای f در چندجمله ای g می توانیم از روش روبرو استفاده کنیم $FFT^{-1}(FFT(f) \circ FFT(g))$. این روش مرتبه زمانی $O(n \log n)$ دارد اگر n چندجمله ای با درجه بالاتر، درجه n داشته باشد. حال می توانیم از این روش برای کانولوشن در فضای دو بعدی استفاده کنیم. برای این منظور به این صورت عمل می کنیم که دو ماتریس را از حالت دو بعدی با عملیات flatten به آرایه های یک بعدی تبدیل می کنیم و در فضای یک بعدی کانولوشن را انجام می دهیم و نتیجه را به حالت دو بعدی برمی گردانیم. در ابتدا باید هر دو ماتریس را با صفر padding کنیم تا مقادیر کرنل در ردیف مناسب در تصویر قرار بگیرند. بنابراین اگر کرنل $H \times W$ باشد کافی است

s.a.m

اگر هر دوی تصویر و کرنل را به اندازه سطر و ستون نهایی Convolution، یعنی $N+M-1$، padding کنیم، هر دو را flatten کنیم، با FFT، کانولوشن را انجام دهیم و حاصل را به حالت دو بعدی برگردانیم. اگر کرنل $k\times k$ باشد و k یک عدد فرد باشد، عنصر وسط این کرنل بر روی هر پیکسل تصویر قرار می‌گیرد و تصویری با ابعاد تصویر اولیه تولید می‌کند. می‌توانیم این کار را با cut off کردن $\frac{k-1}{2}$ ردیف و ستون اول و آخر انجام دهیم. در نتیجه ابعاد نهایی به $N = N+k-1-2\left(\frac{k-1}{2}\right)$ تبدیل می‌شود. زمان اجرای این الگوریتم $O(MN(\log N+\log M)) = O(MN\log MN)$ خواهد بود. حال در اینجا که دو تصویر هر دو $N\times N$ هستند، بنابراین زمان الگوریتم $O(N^2\log N)$ خواهد بود. در حالتی شبیه به اینجا که ابعاد کرنل بزرگ است این روش به ما کمک می‌کند.

(۲) آ) اگر $n=d$ باشد، بی‌نهایت پاسخ برای صفر کردن تابع هزینه وجود دارد.

ب) اگر به دنبال مینیمم کردن هزینه با regularization term باشیم خواهیم داشت:

$$\min J(w) + \lambda \|w\|_2^2 = \min \frac{1}{2}\|Xw - Y\|_2 + \lambda \|w\|_2^2$$

اگر λ صفر باشد، بی‌نهایت جواب خواهیم داشت و اگر بزرگتر از صفر باشد، یک minimizer یکتا خواهیم داشت. این دانشی که به عنوان prior اضافه کردیم می‌خواهد که پاسخ با کمترین نرم را به دست آورد، زیرا می‌دانیم پاسخ با نرم کمتر پاسخی در مساله مورد فضای حل مساله است. حال می‌توانیم w را به صورت زیر بنویسیم:

$$w = X^T \beta + v$$
$$\text{s.t. } v^T x_i = 0 \text{ for all } i$$

$$\Rightarrow \min_{\beta, v} \frac{1}{2} \sum_{i=1}^{n} loss(\beta^T X x_i, y_i) + \lambda \|X^T \beta\|_2^2 + \lambda \|v\|_2^2$$

اگر برحسب v، مینیمم‌سازی انجام دهیم تنها پاسخ برای آن $v=0$ است. در نتیجه پاسخ w به صورت زیر درمی‌آید:

$$w = X^T \beta$$

نشان داده شد که پاسخ w در ابعاد تعداد داده ورودی است و کوچکتر یا مساوی n است. به این نتیجه‌گیری representer theorem گفته می‌شود.

ج) می‌توانیم این مسئله را به صورت روبرو بنویسیم:

$$\min_{w} \|w\|_2^2 \quad \text{subject to} \quad y = Xw$$

* می‌توانیم مسئله بالا را به صورت یک مسئله بدون محدودیت بنویسیم با استفاده از روش لاگرانژ ضرایب $\lim_{\lambda \to \infty}$

و دلیل اینکه λ را به سمت بی‌نهایت می‌بریم آن است که می‌خواهیم خطا (error) صفر بشود:

$$\min_{w} \left(\|w\|_2^2 + \lambda \|y - Xw\|_2^2 \right)$$

این یک مسئله Convex است. بنابراین گرادیان آن صفر خواهد شد در حالت کمینه:

$$2w - 2\lambda X^T (y - Xw) = 0 \Rightarrow w (I + \lambda X^T X) = \lambda X^T y$$

$$\xrightarrow{\lambda \to \infty} w^* = (X^T X)^{-1} X^T y \xrightarrow{w = X^T \alpha} w^* = (X^T X)^{-1} X^T X X^T \alpha = X^T \alpha = w$$

* نشان داده شد که اگر $w = X^T \alpha$ باشد کمینه نرم اقلیدسی برای حالتی که $y = Xw$ باشد همان $w^* = X^T \alpha$ است که توسط SGD به دست آمده است.

۳) اگر فرض کنیم یک شبکه عصبی داریم با m لایه مخفی که همگی تابع فعال‌ساز خطی داشته باشند مدل ما در حقیقت تنها یک مدل linear regression

خواهد بود. به طور مثال به ریاضیات زیر می‌پردازیم که شبکه‌ای با یک لایه مخفی و تابع فعال‌ساز خطی را می‌توان به یک شبکه بدون لایه مخفی تبدیل

کرد و می‌توان به سادگی این روش را برای هر شبکه‌ای با m لایه مخفی با تابع فعال‌ساز خطی تعمیم داد.

$$\left.\begin{aligned} f^1 &= mx+n \\ f^2 &= px+q \end{aligned}\right\} \text{activation functions}$$

$$a^1 = f^1(z^1) = f^1(w^1x^1+b^1) = m[w^1x^1+b^1]+n = mw^1x^1+mb^1+n$$

$$a^2 = f^2(z^2) = f^2(w^2a^1+b^2) = f^2(w^2(mw^1x^1+mb^1+n)+b^2)$$

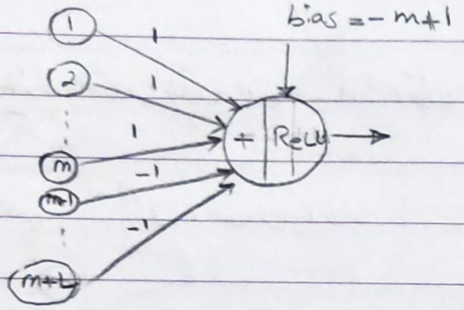
$$= f^2(w^2mw^1x^1+mb^1w^2+w^2n+b^2) = p[w^2mw^1x^1+mb^1w^2+w^2n+b^2]+q$$

$$= \underbrace{pw^2mw^1}_{W}x^1 \underbrace{+pmb^1w^2+pw^2n+pb^2+q}_{B} = Wx^1+B$$

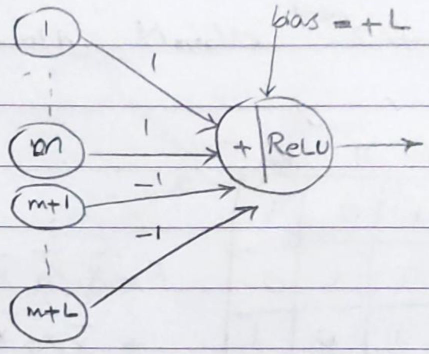
مدلی با تابع فعال‌ساز خطی و بدون لایه مخفی به دست آمد.

الف) با استفاده از شبکه perceptron تک نورونی می‌توانیم گیت‌های AND ، OR و NOT را مانند زیر بسازیم و سپس با داشتن این گیت‌های پایه می‌توانیم هر رابطه منطقی را به فرم SOP درآورده و با شبکه‌ای با یک لایه مخفی، رابطه را نمایش داد.

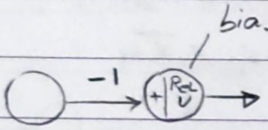
AND



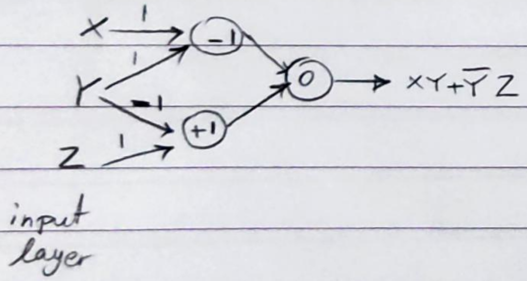
OR



NOT



حال هر تابعی که داشته باشیم می‌توانیم با تبدیل کردن به فرم SOP و با استفاده از AND و OR های ساخته شده، شبکه مورد نیاز را بسازیم. به طور مثال اگر پس از ساده کردن رابطه مورد نظر به رابطه $XY+\bar{Y}Z$ برسیم، می‌توانیم آن را به صورت زیر نشان دهیم:



ب) اگر تابع XOR برای ۴ متغیر را نشان دهیم توسط جدول کارنو، جدولی شبیه شکل زیر می‌شود، می‌دانیم این جدول ساده نمی‌شود

و هر خانه‌ای که ۱ دارد نیاز به یک نورون در لایه مخفی دارد که همان AND متغیرهای آن است و در لایه آخر یک نورون برای OR

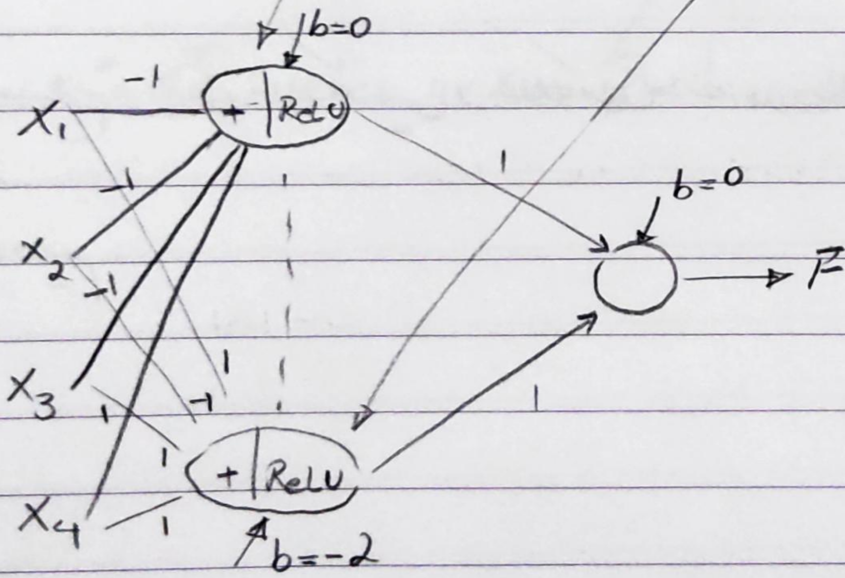
داریم. همانطور که مشخص است از 2^4 حالت مختلف ترتیب متغیرها در این جدول نصف آنها یعنی $2^{(4-1)}$ خانه دقیقا ۱ هستند

و در لایه مخفی نیاز به نورون دارند. این مسئله را می‌توان تعمیم داد به اینصورت که برای هر N ورودی، 2^{N-1} خانه جدول کارنو ۱ می‌شوند

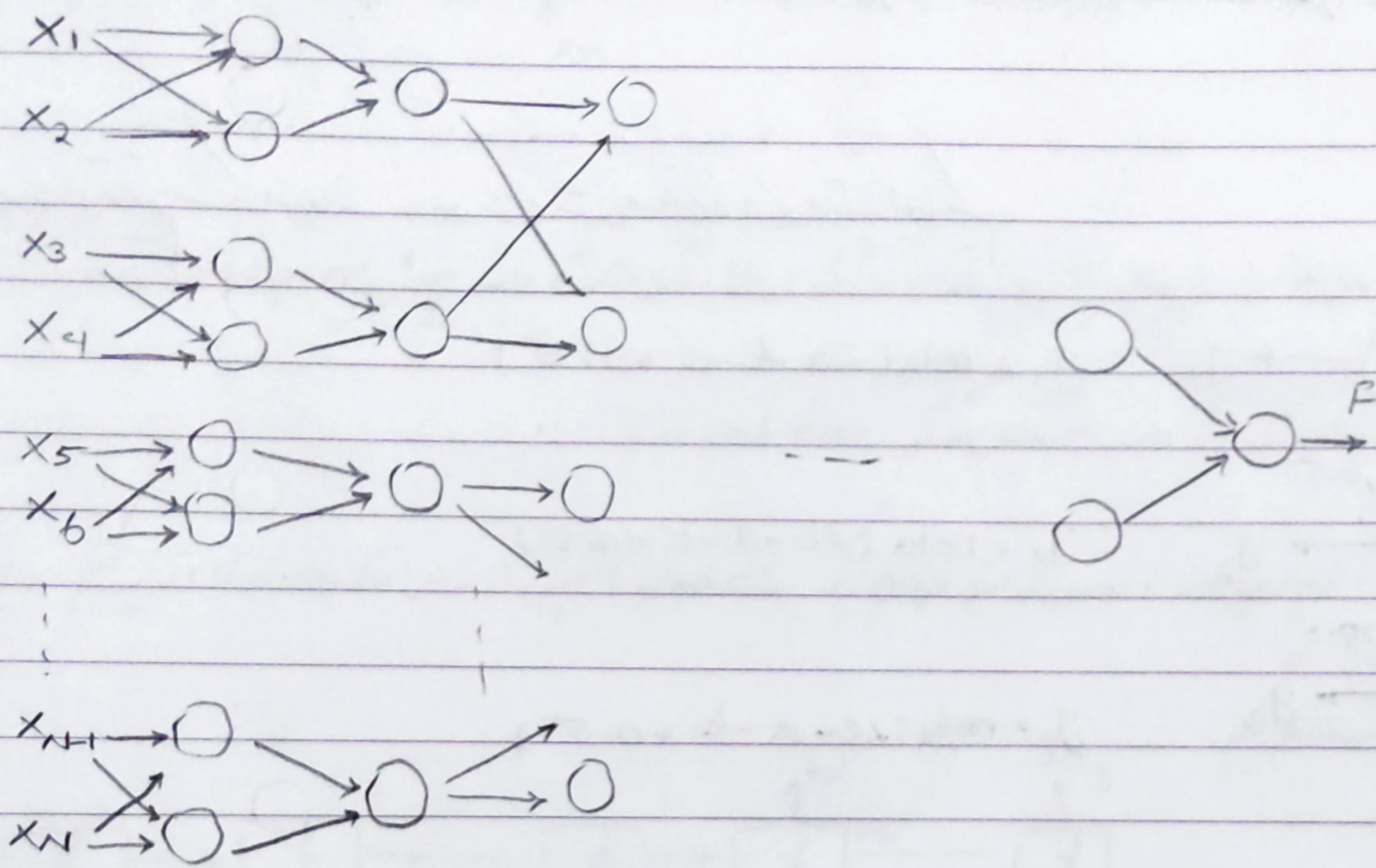
بنابراین لایه مخفی 2^{N-1} نورون نیاز دارد.

$x_1x_2 \backslash x_3x_4$	00	01	11	10
00	0	1	0	1
01	1	0	1	0
11	0	1	0	1
10	1	0	1	0

$$F = \bar{x}_1\bar{x}_2\bar{x}_3x_4 + \bar{x}_1\bar{x}_2x_3\bar{x}_4 + \bar{x}_1x_2\bar{x}_3\bar{x}_4 + \bar{x}_1x_2x_3x_4 + x_1x_2\bar{x}_3x_4 + x_1x_2x_3\bar{x}_4 + x_1\bar{x}_2\bar{x}_3\bar{x}_4 + x_1\bar{x}_2x_3x_4$$



ج: $F = x_1 \oplus x_2 \oplus \cdots \oplus x_N = (((x_1 \oplus x_2) \oplus (x_3 \oplus x_4)) \oplus ((x_5 \oplus x_6) \oplus (x_7 \oplus x_8))) \oplus \cdots$



* به ازای هر XOR ، دو لایه نیاز داریم بنابراین $2\log_2 N$ لایه خواهیم داشت ولی تعداد نورون ها به $3(N-1)$ کاهش می یابد که از درجه خطی نسبت به تعداد ورودی است.

$\frac{1}{1+e^{-x}}$ Sigmoid .

مزایا: ۱) مشتق‌پذیر است و در هر نقطه می‌توانیم مشتق را به دست آوریم.

۲) این تابع monotonic است و نسبیت اعضای مجموعه ورودی را حفظ می‌کند.

۳) گرادیان smooth دارد و باعث می‌شود تغییرات ناگهانی نداشته باشیم.

۴) مقدار خروجی آن بین ۰ و ۱ است بنابراین در مواقعی که نیاز داریم احتمال محاسبه کنیم، بسیار کارآمد است.

۵) مقدار تابع در بازه‌های بالای ۲ و پایین‌تر از ۲- به ترتیب بسیار به ۱ و ۰ نزدیک می‌شود و به ما اجازه تخمین روشنی را می‌دهد.

معایب: ۱) در نقاط ابتدایی و انتهایی این تابع مقدار گرادیان بسیار کوچک می‌شود که باعث می‌شود فرآیند یادگیری بسیار کند شود و اگر مقدار گرادیان صفر شود کلاً یادگیری نخواهیم داشت و در نتیجه عملکرد بدی می‌بینیم. این مشکل vanishing gradient نامیده می‌شود.

۲) خروجی این تابع zero-centered نیست

۳) از نظر محاسباتی سنگین است.

$\frac{e^x - e^{-x}}{e^x + e^{-x}}$ tanh .

مزایا : این تابع از نظر ساختار بسیار شبیه به تابع Sigmoid است. ولی تفاوت هایی نیز دارد، یکی اینکه مقدار خروجی آن در بازه (1 ,1-) است، zero centered است. تفاوت دیگر این است که گرادیان با شیب تندتری نسبت به Sigmoid دارد و بنابراین گرادیان می‌تواند مقادیر بیشتری بگیرد و همینطور برای یادگیری سریع مناسب تر از Sigmoid است.

معایب :
1) این تابع هم دچار مشکل vanishing gradient می‌شود زیرا که در نقاط انتهایی، گرادیانش نزدیک به صفر دارد.

2) این تابع از نظر محاسباتی، تابعی سنگین است.

$\begin{cases} 0 & x<0 \\ x & x>0 \end{cases}$ ReLU .

مزایا : 1) این تابع از نظر پیچیدگی محاسباتی، بسیار سبک‌تر از ۲ تابع قبلی است و به دلیل صفر بودن خروجی به ازای ورودی منفی باعث می‌شود که فعال سازی بسیار سبک تر شده و شبکه سریع‌تر عمل کند. و به همین دلیل برای شبکه‌های چندلایه بیشتر ترجیح داده می‌شود.

2) اگرچه به نظر می‌رسد که در بازه مثبت ویژگی‌های تابع خطی را داشته باشد ولی ReLU یک تابع خطی نیست و یک تخمین‌گر خوب است.

معایب : هنگامیکه مقدار ورودی به صفر و مقادیر منفی نزدیک می‌شود، مقدار گرادیان صفر شده و آموزش صورت نمی‌گیرد. ولی مشکل vanishing gradient ندارد.

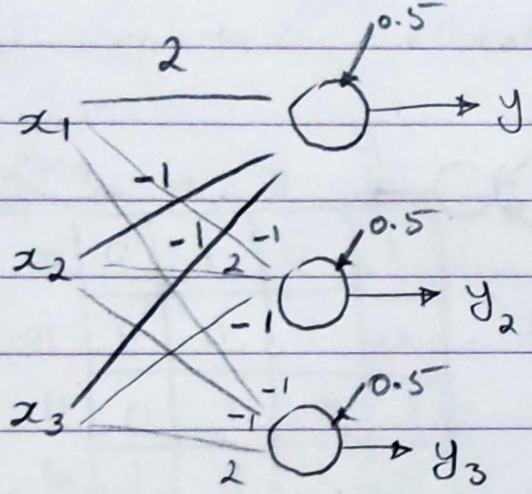
s.a.m

۶) اگر ۳ عدد صحیح داشته باشیم می‌توانیم به طور مثال رابطه‌ای مانند زیر را برای آن بنویسیم:

اگر بدانیم:

$$\left.\begin{array}{l} a \geq b \\ a \geq c \end{array}\right\} \Rightarrow a + a \geq b + c \Rightarrow 2a - b - c \geq 0 \Rightarrow 2a - b - c + 0.5 \geq 0.5 > 0$$

بنابراین می‌توانیم شبکه‌ای مانند روبرو برای شناسایی مقدار بیشینه ایجاد کنیم:



$$y_1 = \mathrm{relu}(2a - b - c + 0.5)$$

$$y_2 = \mathrm{relu}(2b - a - c + 0.5)$$

$$y_3 = \mathrm{relu}(2c - a - b + 0.5)$$

7

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

ΔC_0 در اینجا به معنای گرادیان تابع هزینه بدون regularization term است نسبت به وزن‌ها.

L2 regularization $\Rightarrow C = C_0 + \frac{\lambda}{2} \|w\|_2^2$

$$\frac{\partial C}{\partial w} = \frac{\partial C_0}{\partial w} + 2\frac{\lambda w}{2} \xrightarrow{\Delta C = \partial C / \partial w} w = w - \alpha \Delta C = w - \alpha(\Delta C_0 + \lambda w)$$

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

با جایگذاری $\alpha\lambda = \lambda'$ ، رابطه آپدیت کردن وزن‌های بالا به صورت زیر در می‌آید:

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

که این رابطه دقیقاً برابر با رابطه weight decay است. اثبات شد که در SGD ، L_2 Decay و weight Decay برابر هستند ولی در روش‌هایی مانند Adam این دو یکسان نیستند.

Guided Backpropagation: این روش جزئیات اصلی تصویر را مشخص می‌کند. روش کار آن این است که فرض می‌کند
نورون‌ها مانند detector هایی کار می‌کنند که فیچرهای خاصی از تصویر را شناسایی می‌کنند. بنابراین هنگام back Propagation
مقادیر منفی گرادیان را برابر صفر قرار می‌دهد تا پیکسل‌هایی که در تصویر مهم هستند را هایلایت کند.

Guided Grad CAM: در ابتدا باید بدانیم روشی داریم به نام class Activation Maps که عمل مشخص کننده class
است و نقاط مهم مشخص کننده کلاس را مشخص می‌کند ولی فقط روی شبکه‌هایی کار می‌کند که دقیقاً قبل از لایه prediction
از global average pooling استفاده می‌کنند. روش دیگری داریم به نام Grad CAM که مانند CAM،
class discriminative است ولی پیکسل‌های مهم تصویر را نمایش نمی‌دهد و مزیتش نسبت به CAM این است که بر روی
هر شبکه‌ای قابل اعمال است؛ حال Guided Grad CAM ویژگی‌های Grad CAM و
Guided BackProp. را با یکدیگر ترکیب کرده و هم بخش‌های مرتبط با تخمین کلاس را نشان می‌دهد و هم پیکسل‌های
مهم را در حین back propagation از لایه‌های ReLU، هایلایت می‌کند.