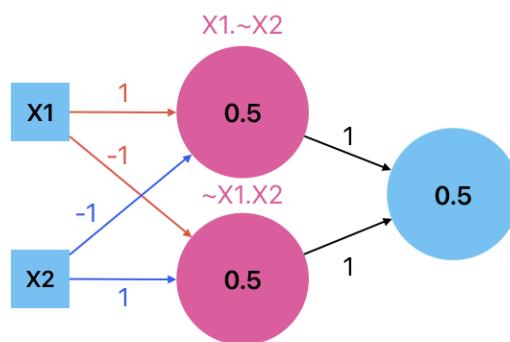


پاسخ سوال ۱:

بخش A: خیر. می‌دانیم که یک پرسپترون تنها قادر به جداسازی داده‌هایی است که به شکل خطی قابل جداسازی هستند. بنابراین از آنجا که تابع XOR را (حتی برای دو متغیر) نمی‌توان به شکل خطی جدا کرد نمی‌توانیم از یک پرسپترون برای این کار استفاده کنیم.

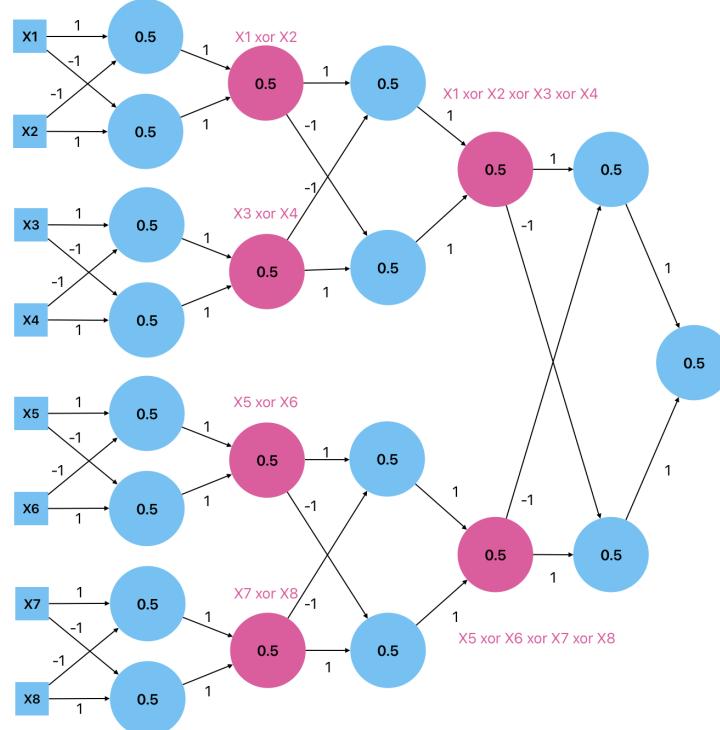
بخش B: شبکه مورد نیاز برای تابع دو متغیره XOR به شکل زیر است. در بالای هر یک از نورون‌های میانی مشخص شده است که آن نورون کدام یک از term های سازنده XOR را در فرم SOP آن بازنمایی می‌کند. برای هر یک از ورودی‌ها اگر نقیض آن در term مورد نظر ظاهر شده باشد وزن -۱ و اگر خودش در آن term ظاهر شده باشد وزن ۱ را اختصاص می‌دهیم. بنابراین هر یک از term ها در صورتی فعال می‌شود که مقدار ورودی ای که وزن آن ۱ است برابر با ۱ و مقدار ورودی ای که وزن آن -۱ است برابر با ۰ باشد. در این صورت مقدار جمع ورودی هر یک از نورون‌های میانی برابر با ۱ خواهد بود و در سایر حالات مقداری کمتر از ۱. پس می‌توانیم آستانه این نورون‌ها را ۰.۵ در نظر بگیریم تا صرفا در حالتی که ورودی‌ها مطلوب باشند مقدار خروجی نورون برابر با ۱ شود و در غیر این صورت خروجی صفر شود.

در نهایت، نورون لایه خروجی وظیفه or کردن term های ساخته شده در لایه میانی را داراست. تنها حالتی که این نورون باید خروجی صفر داشته باشد این است که هر دو نورون لایه میانی خروجی صفر بدهند. به همین دلیل با توجه به اینکه وزن‌های ورودی به لایه آخر را ۱ در نظر گرفته ایم اگر آستانه را برابر با ۰.۵ قرار دهیم نورون خروجی می‌تواند نتیجه or دو نورون لایه میانی را به دست آورد چرا که با ۱ شدن خروجی هر یک از نورون‌های میانی مقدار ورودی به نورون خروجی از ۱ بزرگتر می‌شود و آستانه را رد می‌کند.



حال با استفاده از شبکه ساخته شده می‌توانیم به حل سوال پردازیم. در ادامه به شبکه ساخته شده در بالا به اختصار گیت XOR می‌گوییم.

یک روش که می‌توانیم با استفاده از این شبکه، یک شبکه بزرگتر برای حل مسئله چند متغیره XOR بسازیم این است که ورودی‌ها را در یک ساختار درخت مانند با هم XOR کنیم و پیش برویم. به عنوان مثال شبکه مورد نظر برای ۸ ورودی به شکل زیر خواهد بود. همانطور که می‌بینید در مرحله اول از ۴ گیت XOR، در مرحله دوم از ۲ گیت XOR و در مرحله آخر از یک گیت XOR استفاده شده است تا حاصل عملیات XOR در یک ساختار درخت مانند به دست آید.



حال می‌خواهیم تعداد نورون‌های این شبکه را محاسبه کنیم. برای راحتی محاسبات می‌توانیم فرض کنیم تعداد ورودی‌ها توانی از ۲ است. در این صورت تعداد گیت‌های XOR در شبکه ساخته شده برابر است با:

$$n/2 + n/4 + n/8 + \dots + 1 = n - 1$$

از آنجا که هر گیت XOR دارای ۳ نورون است، تعداد نورون‌های این شبکه برابر است با 3 برابر تعداد نورون‌های گیت XOR:

$$3(n - 1) = 3n - 3 = \Theta(n)$$

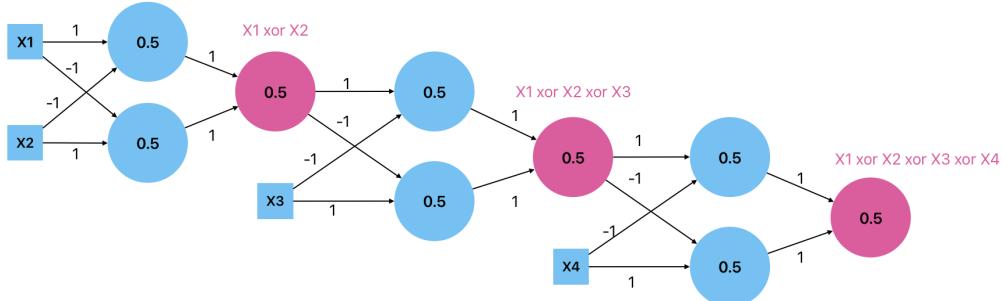
بنابراین ثابت شد که تعداد نورون‌های شبکه از مرتبه $\Theta(n)$ است.

حال عمق شبکه را بررسی می‌کنیم. با توجه به ساختار درختی شبکه، ماکسیمم تعداد گیت‌های XOR ای که لازم است پشت سر هم قرار دهیم تا به انتهای شبکه برسیم برابر با سقف $\log_2 n$ خواهد بود. از آنجا که

عمق شبکه گیت XOR برابر با ۲ بود، عمق این شبکه برابر با 2 برابر سقف $\log_2 n$ خواهد بود که $\Theta(\log_2 n)$ است.

بنابراین شبکه ساخته شده نیازمندی‌های سوال را برطرف می‌سازد.

بخش C: اگر به جای اینکه از یک ساختار درخت مانند استفاده کنیم، ابتدا حاصل XOR دو متغیر اول را محاسبه کنیم و سپس مقدار حاصل را با متغیر سوم XOR کنیم و به همین شکل ادامه دهیم، شبکه ای مانند شکل زیر حاصل می‌شود (برای ۴ ورودی):



برای ساخت چنین شبکه‌ای مانند شبکه قبل به $1 - n$ گیت XOR نیاز داریم. بنابراین تعداد نورون‌های این شبکه نیز همانند شبکه قبل برابر با $3 - 3n$ خواهد بود که با خواسته مسئله سازگار است.

همانطور که مشاهده می‌شود در این حالت $1 - n$ گیت XOR پشت سر هم قرار می‌گیرند. با توجه به اینکه هر گیت XOR عمقی برابر با ۲ دارد، عمق این شبکه برابر است با:

$$2(n - 1) = 2n - 2 = \Theta(n)$$

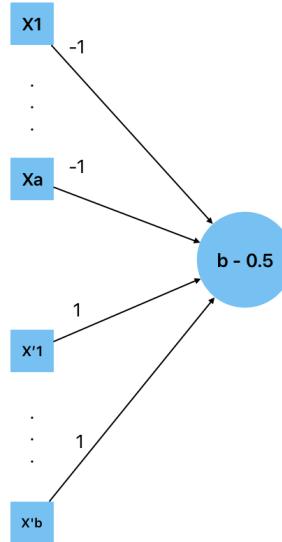
بنابراین شبکه ساخته شده خواسته‌های مسئله را برآورده می‌کند.

بخش D: در این بخش همانطور که گفته شده است لازم است تا از فرم SOP تابع XOR استفاده کنیم. می‌دانیم اگر تعداد ورودی‌هایی که a هستند فرد باشد خروجی XOR برابر با 1 است. حال سوال اینجاست که چه تعداد term می‌توان با n متغیر ساخت که تعداد ورودی‌هایی که a هستند در آنها فرد باشد. با توجه به تقارنی که این مسئله نسبت به زوجیت خواسته شده دارد، این تعداد برابر است با نصف تمام term‌هایی که

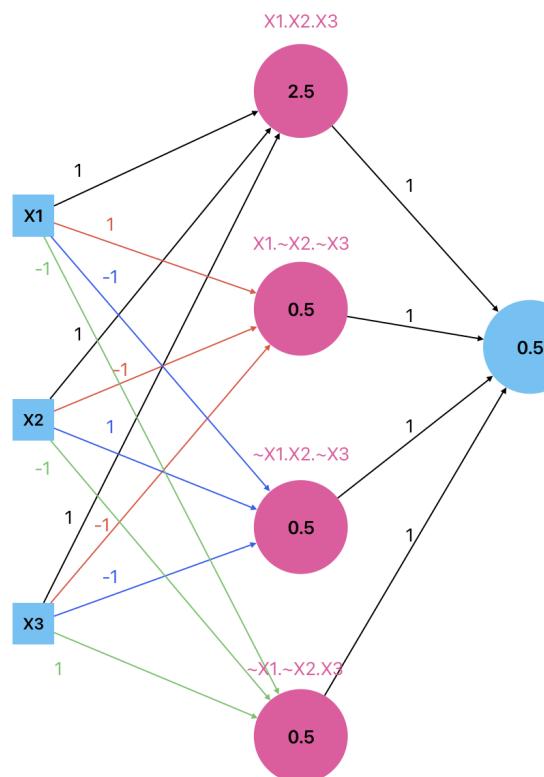
$$2^n / 2 = 2^{n-1}$$

حال کافی است نشان دهیم که هر term را می‌توان با یک نورون میانی هندل کرد. فرض کنید در یک term تعداد متغیرهایی که به صورت نقیض ظاهر شده اند برابر با a و تعداد متغیرهایی که بدون نقیض ظاهر شده اند برابر با b باشد. در این صورت اگر به متغیرهایی که در دسته اول هستند وزن -1 و به متغیرهای دسته دوم وزن 1 را اختصاص دهیم و آستانه نورون مورد نظر را برابر با $0.5 - b$ قرار دهیم، نورون ساخته شده در صورت

مطابقت ورودی‌ها با term موردنظر مقدار ۱ را خروجی می‌دهد. دلیل این اتفاق این است که اگر ورودی‌ها با حالت مطلوب فرق داشته باشند، جمع ورودی‌های وارد به نورون یک عدد صحیح کوچکتر از b می‌شود و از حد آستانه کمتر می‌شود.



حال که می‌توانیم هر term را با یک نورون هندل کنیم می‌توان گفت که شبکه عصبی مورد نظر می‌تواند با داشتن n^{n-1} نورون در لایه میانی این مسئله را حل کند. در شکل پایین یک نمونه از این شبکه را برای سه متغیر مشاهده می‌کنید. در لایه آخر شبکه نیز مشابه قبل عملیات or بین term‌ها صورت می‌گیرد.



پاسخ سوال 2:

Date:

Subject:

$$A_i = \tanh\left(\sum_{j=0}^3 X_j V_{ji} + b_i\right) = \tanh(2) = 0.96$$

$$D_i = \tanh\left(\sum_{j=1}^4 A_j U_{ji} + c_i\right) = \tanh(2 \tanh(2)) = 0.96$$

$$E_1 = \text{swish}(K_{11}D_1 + K_{21}D_2 + K_{31}D_3 + g_1 + A_2 w)$$

$$= \text{swish}\left(1.5 \tanh(2 \tanh(2)) + \frac{\tanh(2)}{2}\right) = 1.67$$

$$E_2 = \text{swish}(K_{12}D_1 + K_{22}D_2 + K_{32}D_3 + g_2)$$

$$= \text{swish}(1.5 \tanh(2 \tanh(2))) = 1.16$$

$$Y = \text{sigmoid}(w_0 + w_1 E_1 + w_2 E_2) = 0.94$$

$$\text{Loss} = \frac{1}{2} (Y - \hat{Y})^2 = L = 0.002$$

$$\frac{\partial L}{\partial Y} = Y - \hat{Y} = -0.06$$

$$\begin{cases} Y = \text{sigmoid}(x) = \sigma(x) \\ \rightarrow \frac{\partial Y}{\partial x} = \sigma(x)(1 - \sigma(x)) = Y(1 - Y) \end{cases}$$

$$\frac{\partial L}{\partial E_1} = \frac{\partial L}{\partial Y} \times \frac{\partial Y}{\partial E_1} = \frac{\partial L}{\partial Y} (Y(1 - Y) w_1) = -0.0017$$

$$K_{ian SHAD} \rightarrow \frac{\partial L}{\partial E_2} = \frac{\partial L}{\partial Y} (Y(1 - Y) w_2) = -0.0017$$

Date:

Subject:

$$E_1 = \text{swish}(e_1), E_2 = \text{swish}(e_2)$$

$$\begin{aligned} Y &= \text{swish}(x) = \text{sw}(x) = x \text{ sigmoid}(x) = x \sigma(x) \\ \rightarrow \frac{\partial Y}{\partial x} &= \sigma(x) + x \sigma(x)(1 - \sigma(x)) = \sigma(x) + Y - Y \sigma(x) \end{aligned}$$

$$\begin{aligned} \frac{\partial L}{\partial w} &= \frac{\partial L}{\partial E_1} \cdot \frac{\partial E_1}{\partial w} = \frac{\partial L}{\partial E_1} [E_1 + \sigma(e_1) - E_1 \sigma(e_1)] A_2 \\ &= -0.0018 \end{aligned}$$

$$\begin{aligned} \frac{\partial L}{\partial D_i} &= \frac{\partial L}{\partial E_1} \frac{\partial E_1}{\partial D_i} + \frac{\partial L}{\partial E_2} \frac{\partial E_2}{\partial D_i} \\ &= \frac{\partial L}{\partial E_1} [E_1 + \sigma(e_1) - E_1 \sigma(e_1)] K_{i1} \\ &\quad + \frac{\partial L}{\partial E_2} [E_2 + \sigma(e_2) - E_2 \sigma(e_2)] K_{i2} \end{aligned}$$

$$\rightarrow \frac{\partial L}{\partial D_1} = -0.0018 \quad \frac{\partial L}{\partial D_2} = -0.0018 \quad \frac{\partial L}{\partial D_3} = -0.0018$$

$$D_i = \tanh(\dots + A_2 u_{2i}) = \tanh(d_i)$$

$$Y = \tanh(x) \rightarrow \frac{\partial Y}{\partial x} = 1 - \tanh^2(x) = 1 - Y^2$$

$$\frac{\partial L}{\partial u_{2i}} = \frac{\partial L}{\partial D_i} \frac{\partial D_i}{\partial u_{2i}} = \frac{\partial L}{\partial D_i} (1 - D_i^2) A_2$$

$$= -0.0018 \times (1 - 0.96^2) \times 0.96 = -0.00014$$

$$w = w - \alpha \frac{\partial L}{\partial w} = 0.50018$$

$$u_{2i} = u_{2i} - \alpha \frac{\partial L}{\partial u_{2i}} = 0.500014$$

پاسخ سوال 3:

(a)

هر دو الگوریتم batch gradient descent و stochastic gradient descent روش هایی برای یافتن مجموعه پارامترهایی هستند loss function را کمینه کنند. (که این کار با ارزیابی پارامترها در برابر داده ها و سپس انجام یک سری تنظیمات حاصل میشود). در Batch Gradient Descent، تمام داده های آموزشی برای انجام یک مرحله از الگوریتم گرادیان کاهشی در نظر گرفته می شود. به این معنی که میانگین گرادیان تمام داده های آموزشی را میگیریم و سپس از آن گرادیان میانگین برای بهروزرسانی پارامترهایمان استفاده میکنیم. از سوی دیگر، stochastic gradient descent ، گرادیان را تنها با استفاده از یک نمونه آموزشی(هر نمونه در واقع یک سطر از داده هاست) در هر تکرار از الگوریتم گرادیان کاهشی محاسبه میکند. سایر تفاوت ها و کاربرد هر یک از ۲ الگوریتم در شرایط مختلف در جدول زیر قابل مشاهده هستند:

Batch Gradient Descent	Stochastic Gradient Descent
الگوریتم کند و از نظر محاسباتی پرهزینه است	سریعتر و از نظر محاسباتی کم هزینه تر از Batch GD است.
همگرایی کند است.	همگرایی سریعتر
دقیق تر است زیرا گرادیان را با استفاده از کل مجموعه داده آموزشی محاسبه می کند.	می تواند دقیق تر داشته باشد زیرا گرادیان را با استفاده از زیرمجموعه ای از مثال ها محاسبه می کند، که می تواند نویز و واریانس بیشتری را در برآورد گرادیان ایجاد کند.

کاربرد هر یک:

برای بهینه سازی توابع غیر محدب مناسب تر است زیرا می تواند از حداقل های محلی فرار کند و حداقل جهانی را پیدا کند. از طرف دیگر، BGD ممکن است در مینیمم های محلی گیر کند. BGD از نظر تئوری تابع خطای SGD را بهتر از SGD به حداقل می رساند. با این حال، زمانی که مجموعه داده بزرگ شود، SGD خیلی سریعتر همگرا می شود. این بدان معناست که BGD برای مجموعه داده های کوچک ترجیح داده می شود در حالی که SGD برای مجموعه های بزرگتر ترجیح داده می شود. با این حال، در عمل، SGD برای اکثر کاربردها استفاده می شود، زیرا عملکرد خطای SGD را به اندازه کافی به حداقل می رساند در حالی که برای مجموعه داده های

بزرگ بسیار سریع‌تر و کارآمدتر از حافظه است.

فرق این دو الگوریتم با الگوریتم mini-batch gradient descent می‌باشد: از آنجایی که در SGD ما در هر زمان فقط از یک نمونه آموزشی استفاده می‌کنیم، نمی‌توانیم پیاده‌سازی برداری الگوریتم را برای این الگوریتم انجام دهیم. این می‌تواند محاسبات را کند کند. برای مقابله با این مشکل، ترکیبی از Batch Gradient Descent و mini-batch gradient descent است. بنابراین نه ما از همه مجموعه داده به طور همزمان استفاده می‌کنیم و نه از یک نمونه واحد در یک مرحله از الگوریتم استفاده می‌کنیم. ما از مجموعه‌ای از تعداد ثابت نمونه‌های آموزشی استفاده می‌کنیم که کمتر از مجموعه داده واقعی است و آن را یک دسته کوچک می‌نامیم. انجام این کار به ما کمک می‌کند تا به مزایای هر دو نوع قبلی که دیدیم دست پیدا کنیم. بنابراین، پس از ایجاد mini batch ها با اندازه ثابت، مراحل زیر را در یک epoch انجام می‌دهیم:

1. یک mini batch انتخاب کنید
2. آن را به شبکه عصبی بدهید
3. میانگین گرادیان mini batch را محاسبه کنید
4. از گرادیان میانگینی که در مرحله 3 محاسبه کردیم برای به روز رسانی وزن‌ها استفاده کنید
5. مراحل 1 تا 4 را برای mini batch هایی که ایجاد کردیم تکرار کنید

(b)

همانطور که ما هایپر پارامتر لامبدا را افزایش می‌دهیم، وزن‌ها شروع به کوچک شدن می‌کنند. این را می‌توان با معادله به روز رسانی وزن‌ها در گرادیان کاهشی (با تنظیم L2) که $w = w(1 - \alpha * \lambda / m) - \alpha * dLoss / dw$ است تأیید کرد. بنابراین، با افزایش λ به یک مقدار بسیار بالا، وزن‌ها به 0 نزدیک‌تر می‌شوند. این منجر به مدلی می‌شود که خیلی ساده است و در نهایت به داده‌ها underfit می‌شود و در نتیجه عملکرد مدل را کاهش می‌دهد. پس اگر مقدار لامبدا شما بیش از حد بالا باشد، مدل شما ساده خواهد بود و به اندازه کافی در مورد داده‌های آموزشی یاد نمی‌گیرد تا بتواند پیش‌بینی‌های مفیدی انجام دهد. اگر مقدار لامبدا شما خیلی کم باشد، مدل شما پیچیده‌تر می‌شود و شما در معرض خطر بیش برآذش (overfitting) داده‌هایتان هستید.

L1 and L2 Regularization Methods

تفاوت اصلی بین این تکنیک‌ها این است که L1 ضریب ویژگی‌های کم اهمیت‌تر را به صفر کاهش می‌دهد، بنابراین برخی از ویژگی‌ها را به طور کلی حذف می‌کند. بنابراین، در صورتی که تعداد زیادی ویژگی داشته

باشیم، L1 برای feature selection خوب عمل می کند. از سوی دیگر، L2 زمانی مفید است که ویژگی های collinear/codependent داشته باشید. (یک مثال از ویژگی های وابسته جنسیت و باردار هستند، زیرا فقط زنان می توانند باردار باشند.)

(c)

Dropout تکنیکی است که در آن نورون های انتخابی تصادفی در طول تمرین نادیده گرفته می شوند. در این روش تعدادی از نورون ها فقط برای یک عبور از شبکه به جلو و عقب (feed forward and back) حذف می شوند - به این معنی که وزن آنها به طور مصنوعی برای آن عبور صفر می شود و خطای آنها نیز . میشود و متعاقبا وزن های آن نورونها به روز نمی شود(Dropout) در واقع نورون ها را از کل شبکه حذف نمی کند- بلکه فقط برای آن اپیاک خاص غیرفعال میشوند). همچنین به عنوان شکلی از منظم سازی عمل می کند، زیرا تا حدودی مدل را به دلیل پیچیدگی اش جریمه می کند. برخلاف منظم سازی Dropout و L1 و L2، این روش به تغییرتابع هزینه متکی نیست. در عوض، خود شبکه را اصلاح می کنیم. باعث افزایش مقاومت(robustness) مدل و همچنین حذف هر گونه وابستگی ساده بین نورون ها میشود. برای درک این مساله و اینکه چرا از یک مدل ساده تر بجای استفاده از تکنیک dropout استفاده نمیکنیم مثال زیر را در نظر بگیرید: تصور کنید از شما می خواهم که برای من یک فنجان چای درست کنید - ممکن است همیشه از دست راست خود برای ریختن آب، از چشم چپ برای اندازه گیری سطح آب و سپس دوباره از دست راست برای هم زدن چای با قاشق استفاده کنید. این بدان معناست که دست چپ و چشم راست شما استفاده کمی دارند. با استفاده از dropout به عنوان مثال مجبور میشوید دست راست خود را از پشت بیندید - و بنابراین مجبور شوید از دست چپ خود استفاده کنید) که سابقا مورد استفاده کمتری بود). حالا بعد از تهییه 20 فنجان چای، با یک چشم یا یک دست بسته، در استفاده از همه چیز بهتر عمل خواهید کرد))). شاید بعدا مجبور شوید در یک آشپزخانه کوچک چای درست کنید، جایی که فقط می توان از کتری با بازوی چپ خود استفاده کرد... بنابراین با آن روش شما در مقایسه با حالتی که مدل ساده تری از اول ارائه کنید، نسبت به داده های دیده نشده قوی تر شده اید. از طرفی پیچیدگی مدل را کاملا از دست نداده اید بلکه از overfitting نیز جلوگیری کرده اید.

(d)

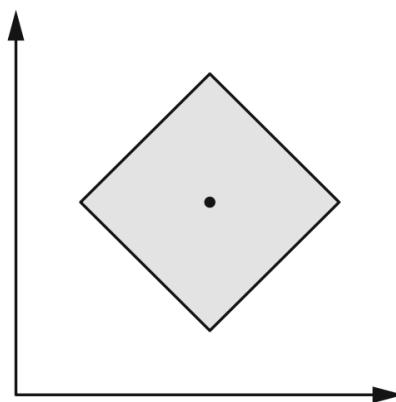
با توجه به شکل داده شده، میتوان دید که نمودار X نسبت به بقیه با شیب تند تری همگرا شده است پس میتوان گفت که نرخ یادگیری آن از همه بزرگتر است. بین y, z ، میتوان دید که برای z همگرایی اتفاق افتاده اما y هنوز به همگرایی نرسیده و احتمالا در ادامه همگرا میشود. پس y نرخ یادگیری کوچکتری نسبت به z دارد. به صورت کلی میتوان نوشت: $y < z < X$

پاسخ سوال ۴:

بخش A: به ازای $k=1$ این تابع برابر است با:

$$d_k(x, y) = \left(\sum_{i=1}^n |x_i - y_i|^k \right)^{\frac{1}{k}} = \sum_{i=1}^n |x_i - y_i| = |x - y|$$

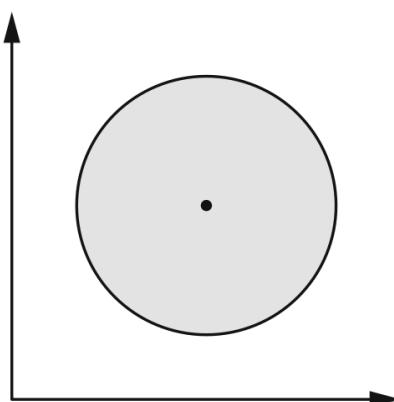
بنابراین در این حالت این تابع برابر با تابع فاصله منهتن (norm-1) است. در این حالت مکان هندسی نقاطی که فاصله یکسانی نسبت به مرکز ناحیه خواهند داشت به شکل یک لوزی خواهد بود.



به ازای $k=2$ این تابع برابر است با:

$$d_k(x, y) = \left(\sum_{i=1}^n |x_i - y_i|^k \right)^{\frac{1}{k}} = \sqrt{\sum_{i=1}^n (x_i - y_i)^2} = \|x - y\|$$

بنابراین در این حالت این تابع برابر با تابع فاصله اقلیدسی (norm-2) است. در این حالت مکان هندسی نقاطی که فاصله یکسانی نسبت به مرکز ناحیه خواهند داشت به شکل یک دایره خواهد بود.



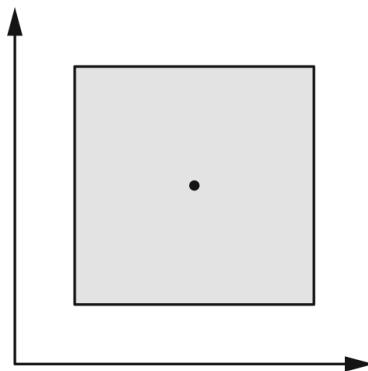
بخش B: اگر بخواهیم به شکل شهودی به این مسئله نگاه کنیم هر چه k بزرگتر شود میزان تاثیر $|x_i - y_i|$ ای که بزرگتر از بقیه است بیشتر و بیشتر می‌شود و در حالت حدی و زمانی که k به بینهایت میل می‌کند

تنها همان $|x_i - y_i|$ است که باقی می‌ماند. بنابراین می‌توان گفت که در این حالت مقدار تابع فاصله برابر

است با:

$$d_k(x, y) = \max_i\{|x_i - y_i|\}$$

بنابراین در این صورت تنها بعدی تاثیرگذار است که دو بردار در آن بعد بیشترین فاصله را نسبت به هم دارند. به این ترتیب مکان هندسی نقاطی که در این حالت فاصله یکسانی از مرکز ناحیه دارند به شکل یک مربع می‌باشد.



اثبات اینکه چرا این فاصله در بین نهایت بردارها می‌شود از شما خواسته نشده اما برای دانشجویانی که علاقه مند هستند در ادامه آورده شده است.

فرض کنید M بزرگتر از سایر $|x_1 - y_1|$ ها باشد. به عبارتی:

$$M = \max_i\{|x_i - y_i|\}$$

در این صورت مقدار فاصله برابر است با:

$$d_k(x, y) = \left(\sum_{i=1}^n |x_i - y_i|^k \right)^{\frac{1}{k}} = M \left(\frac{\sum_{i=1}^n |x_i - y_i|^k}{M^k} \right)^{\frac{1}{k}} = M \left(\left(\frac{\sum_{i=1}^n |x_i - y_i|}{M} \right)^k \right)^{\frac{1}{k}}$$

از آنجا که به ازای هر ا مقدار M بزرگتر یا مساوی با $|x_i - y_i|$ است:

$$M \left(\left(\frac{\sum_{i=1}^n |x_i - y_i|}{M} \right)^k \right)^{\frac{1}{k}} \leq M n^{\frac{1}{k}}$$

همچنین از آنجا که به ازای یک مقدار مشخص از ا مقدار M برابر با $|x_i - y_i|$ می‌شود:

$$M \left(\left(\frac{\sum_{i=1}^n |x_i - y_i|}{M} \right)^k \right)^{\frac{1}{k}} \geq M (1)^{\frac{1}{k}}$$

حال حد کران‌های به دست آمده برای این تابع را به دست می‌آوریم:

$$\lim_{k \rightarrow \infty} M n^{\frac{1}{k}} = M$$

$$\lim_{k \rightarrow \infty} M(1)^{\frac{1}{k}} = M$$

بنابراین طبق قضیه فشردگی می‌توان گفت که حد تابع فاصله مینکوفسکی نیز در بی‌نهایت برابر با M است

که M برابر است با:

$$M = \max_i \{|x_i - y_i|\}$$

پاسخ سوال ۵:

اگر ضریب منظم‌سازی برابر با λ باشد، با استفاده از روش شبه معکوس^۱ نتیجه نهایی به شکل زیر است:

$$W = (G^T G + \lambda I_{L \times L})^{-1} G^T Y$$

$$G_{m \times L}, W_{L \times 2}, Y_{M \times 2}$$

اثبات اینکه چرا در حالتی که چند خروجی داریم می‌توانیم همچنان از روش شبه معکوس استفاده کنیم و همچنین اینکه ضریب منظم‌سازی چگونه وارد معادله می‌شود از شما خواسته نشده است اما در تصاویر زیر می‌توانید اثبات آن را مشاهده کنید.

نحوه محاسبه W در مدل $Y = XW + \epsilon$ با λ ضریب منظم‌سازی:

$$\min \left\{ \frac{1}{2} \|XW - Y\|^2 + \lambda \|W\|^2 \right\}$$

$$\frac{1}{2} [\|XW - Y\|^2 + \lambda \|W\|^2] = \frac{1}{2} (XW - Y)^T (XW - Y) + \frac{1}{2} \lambda \|W\|^2$$

$$= \frac{1}{2} (W^T X^T X W - W^T X^T Y - Y^T X W + Y^T Y) + \frac{1}{2} \lambda W^T W$$

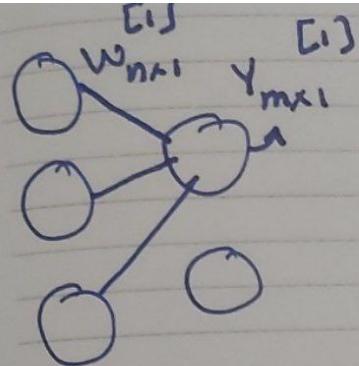
چون $X^T X$ ماتریس مربع متمایز است، $W^T X^T Y$ و $Y^T X W$ متریس 0 است.

$$= \frac{1}{2} (W^T X^T X W - 2Y^T X W + Y^T Y) + \frac{1}{2} \lambda W^T W$$

با $W \approx$ مسُقیّت $\Rightarrow W^T X^T X - Y^T X + \lambda W^T = 0 \Rightarrow X^T X W - X^T Y + \lambda W = 0$

$$\Rightarrow X^T X W + \lambda W = X^T Y \Rightarrow (X^T X + \lambda I) W = X^T Y \Rightarrow W = (X^T X + \lambda I)^{-1} X^T Y$$

¹ Pseudo Inverse



۲ حالت دو چند حریض را درم (۶۲)

با درجہ بہ قسمت قبل درم،

$$w^{[1]} = (X^T X + \lambda I)^{-1} X^T Y^{[1]}$$

$$w^{[2]} = (X^T X + \lambda I)^{-1} X^T Y^{[2]}$$

بایان از ماتریس \$w\$، \$Y\$ بے عمل نہ باشد:

$$W = \begin{bmatrix} I & | & | \\ w^{[1]} & | & w^{[2]} \\ | & | & | \end{bmatrix}, \quad Y = \begin{bmatrix} I & | & | \\ Y^{[1]} & | & Y^{[2]} \\ | & | & | \end{bmatrix}$$

$$W = (X^T X + \lambda I)^{-1} X^T Y$$

13

14

پاسخ سوال 6:

input				kernel		
$X:$						
x_{11}	x_{12}	x_{13}	x_{14}	k_{11}	k_{12}	k_{13}
x_{21}	x_{22}	x_{23}	x_{24}	k_{21}	k_{22}	k_{23}
x_{31}	x_{32}	x_{33}	x_{34}	k_{31}	k_{32}	k_{33}
x_{41}	x_{42}	x_{43}	x_{44}			

$$X_{ij} = X[i \text{ to } i+2, j \text{ to } j+2] \rightarrow 3 \times 3$$

$$\text{input: } 4 \times 4, \text{ kernel: } 3 \times 3 \rightarrow \text{output: } 2 \times 2 : O$$

$$\text{output}_{ij} = X_{ij} \oplus K$$

$$O_{11} = x_{11}k_{11} + x_{12}k_{12} + x_{13}k_{13} + x_{21}k_{21} + x_{22}k_{22} \\ + x_{23}k_{23} + x_{31}k_{31} + x_{32}k_{32} + x_{33}k_{33}$$

$$O_{12} = \dots, O_{21} = \dots, O_{22} = \dots$$

$$O = \begin{bmatrix} 3 & 0 \\ 1 & 1 \end{bmatrix}$$

b)

$$y: \text{actual value} = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$$

$$\text{Loss} = \frac{1}{4} ((O_{11} - Y_{11})^2 + (O_{12} - Y_{12})^2 + (O_{21} - Y_{21})^2 + (O_{22} - Y_{22})^2)$$

$$\frac{\partial \text{Loss}}{\partial O_{11}} = \frac{1}{2} (O_{11} - Y_{11})$$

$$\frac{\partial O_{11}}{\partial K_{11}} = x_{11}$$

$$\rightarrow \frac{\partial \text{Loss}}{\partial K_{11}} = \frac{\partial \text{Loss}}{\partial O_{11}} \frac{\partial O_{11}}{\partial K_{11}} + \frac{\partial \text{Loss}}{\partial O_{12}} \frac{\partial O_{12}}{\partial K_{11}} \\ + \frac{\partial \text{Loss}}{\partial O_{21}} \frac{\partial O_{21}}{\partial K_{11}} + \frac{\partial \text{Loss}}{\partial O_{22}} \frac{\partial O_{22}}{\partial K_{11}}$$

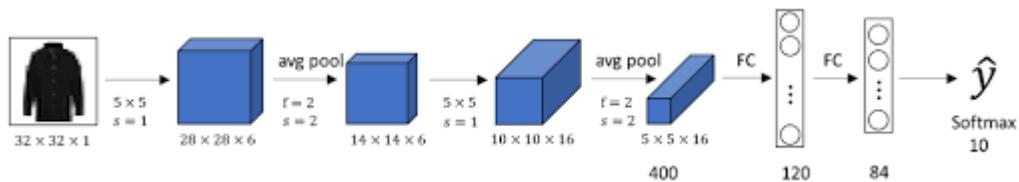
$$\frac{\partial \text{loss}}{\partial k} = \begin{bmatrix} 1.0 & 2.5 & 4.0 \\ 0.5 & 2.0 & 3.5 \\ 1.5 & 3.0 & 4.5 \end{bmatrix}$$

$$k = k - 0.1 * \frac{\partial loss}{\partial k}$$

$$k = \begin{bmatrix} 0.90 & -1.25 & 0.60 \\ -1.05 & 0.80 & -1.35 \\ 0.85 & -1.30 & 0.55 \end{bmatrix}$$

C)

برای استفاده از لایه‌ی کانولوشنی به جای لایه FC باید جوری عمل کنیم که خروجی ما بشود
 $1 * 1 * 1 * c$ پس داریم به جای اولین لایه FC که ورودی آن $16 * 5 * 5$ است لایه کانولوشنی قرار میدهیم



با ساختار $120 * 5 * 5, s = 1, C_{out} = 1$ و در ادامه به جای لایه بعدی دوباره لایه ای به شکل
 $1 * 1 * 1, s = 1, C_{out} = 10$ و به جای لایه بعدی داریم $1 * 1, s = 1, C_{out} = 84$
 من شود $10 * 1 * 1$ که softmax را در بعد آخر آن محاسبه میکنیم.

پاسخ سوال 7

بخش اول)

(الف) مفهوم تابع همسایگی در نقشه‌های خودسازماندهی (SOM) نحوه تعامل نورون‌های همسایه و به روز رسانی وزن‌های خود را در طول فرآیند یادگیری تعیین می‌کند. تابع همسایگی میزان تاثیر نورون‌های مجاور را توسط داده‌های ورودی مشخص می‌کند. تابع همسایگی معمولاً یک تابع است که تأثیر بیشتری را به نورون‌های نزدیک به نورون برنده (نورونی با وزن‌های مشابه ورودی) اختصاص می‌دهد و به تدریج تأثیر نورون‌های دورتر را کاهش می‌دهد. در طول آموزش، هنگامی که ورودی به SOM داده می‌شود، نورون برنده (که بهترین واحد تطبیق یا BMU نیز نامیده می‌شود) بر اساس شباهت آن به ورودی تعیین می‌شود. تابع همسایگی یک همسایگی را در اطراف BMU تعریف می‌کند و نورون‌های درون این همسایگی وزن‌های خود را

طوری تنظیم می‌کنند که شبیه‌تر به ورودی شوند. نورون‌های نزدیک‌تر به BMU بیشتر از آن‌هایی که دورتر هستند، به روزرسانی می‌شوند و تشکیل خوش‌ها را ترویج می‌کنند و همچنین ساختار توپولوژیکی فضای ورودی را نیز حفظ می‌کنند.

تأثیر تابع همسایگی در طول زمان با پیشرفت یادگیری کاهش می‌یابد و نرخ یادگیری کاهشی را شبیه‌سازی می‌کند. این به SOM اجازه می‌دهد تا به سمت یک نمایش پایدار از داده‌های ورودی همگرا شود.

(ب) میزان یادگیری در SOM میزان به روز رسانی وزن را در طول تمرین تعیین می‌کند. این نرخ کنترل می‌کند که نورون‌ها با چه سرعتی وزن خود را با داده‌های ورودی تطبیق بدھند. نرخ یادگیری بالا منجر به به روزرسانی‌های وزن قابل توجه‌تر می‌شود، و باعث می‌شود SOM سریع‌تر همگرا شود اما به طور بالقوه، بیش از حد راه حل‌های بهینه را پشت سر² می‌گذارد. نرخ یادگیری پایین منجر به به روزرسانی‌های وزن کوچک‌تر می‌شود که منجر به همگرایی کنترل اما به طور بالقوه، تنظیم دقیق مدل می‌شود.

در طول آموزش، نرخ یادگیری معمولاً³ به مقدار بالاتری، مقداردهی اولیه می‌شود و به تدریج در طول زمان کاهش می‌یابد. به روز رسانی نرخ یادگیری اغلب توسط یک تابع کاهش³ یا یک برنامه از پیش تعیین شده کنترل می‌شود. توابع کاهش نرخ یادگیری که معمولاً⁴ مورد استفاده قرار می‌گیرند شامل کاهش زمانی خطی، نمایی یا معکوس⁴ هستند. برنامه کاهش تضمین می‌کند که نرخ یادگیری به تدریج کاهش می‌یابد تا به SOM اجازه دهد تا به آرامی همگرا شود.

(ج) مقداردهی اولیه وزن‌ها در SOM گام مهمی در فرآیند یادگیری است. وزن نورون‌های SOM قبل از شروع آموزش مقداردهی اولیه می‌شود و انتخاب روش اولیه می‌تواند بر همگرایی و عملکرد مدل تأثیر بگذارد.

یکی از رویکردهای رایج برای مقداردهی اولیه وزن، تخصیص تصادفی مقادیر اولیه به وزن‌ها در یک محدوده کوچک است. این به نورون‌ها اجازه می‌دهد تا بازنمایی‌های اولیه متنوعی داشته باشند، که برای SOM برای یادگیری خوش‌ها و الگوهای مختلف در فضای ورودی ضروری است. یک دیگر

² Overshoot

³ Decaying Function

⁴ Inverse Time Decay

از تکنیک های اولیه سازی استفاده از یک الگوریتم خوش بندی بر روی داده های ورودی برای تعیین وزن اولیه است.

اهمیت مقداردهی اولیه وزن در ارائه نقطه شروع برای فرآیند یادگیری نهفته است. اگر وزن ها بد مقدار دهی اولیه شوند، من تواند منجر به همگرایی کنترلر با یک راه حل غیربهینه شود. مقداردهی اولیه وزن ها به طور مناسب تضمین می کند که SOM با یک نقطه آغازی معقول شروع می شود و بهتر می تواند توزیع داده های اساسی مجموعه داده را یادگیری کند.

(d) اندازه شبکه SOM، که با تعداد نورون های شبکه تعریف می شود، بر عملکرد خوشبندی و پیچیدگی محاسباتی SOM تأثیر دارد.

از نظر عملکرد خوشبندی، اندازه شبکه SOM بزرگتر می تواند جزئیات دقیق تری را ثبت کند و وضوح بالاتری را در نمایش داده های ورودی ارائه دهد. من تواند منجر به خوش های تمایزتر و حفظ بهتر توپولوژی ورودی بشود. با این حال، یک شبکه بیش از حد بزرگ ممکن است منجر به بیش از حد برآذش⁵ شود، به خصوص در حالتی که داده های ورودی محدود باشند. از سوی دیگر، یک شبکه کوچکتر ممکن است منجر به نمایش های ساده شده غیر مفید و کاهش عملکرد خوش بندی شود.

از نظر پیچیدگی محاسباتی، یک شبکه SOM بزرگتر به منابع محاسباتی و زمان آموزش بیشتری نیاز دارد. تعداد نورون ها در شبکه به طور مستقیم بر تعداد به روز رسانی وزن و محاسبات انجام شده در طول تمرین تأثیر می گذارد. با افزایش اندازه شبکه، پیچیدگی محاسباتی افزایش می یابد و روند آموزش را کنترلر و منابع فشرده تر می کند.

ایجاد تعادل بین اندازه شبکه، منابع محاسباتی و پیچیدگی داده های ورودی برای دستیابی به یک تعادل خوب بین عملکرد خوش بندی و کارایی محاسباتی در SOM مهم است.

⁵ Overfitting

بخش دوم

Iteration 1

$$\omega_{ij} \in [0, 1, 0, 1, 0, 1, 0, 1, 0, 1]$$

d_{ij} is neuron j if n_i is true

$$d_{11} = \sqrt{0,1 + 0,09 + 0,14 + 0,01} = \sqrt{0,44}$$

$$d_{14} = \sqrt{0,14 + 0,06 + 0 + 0,12} = \sqrt{0,32}$$

$$d_{12} = \sqrt{0 + 0,05 + 0,02 + 0} = \sqrt{0,07}$$

$$d_{15} = \sqrt{0,12 + 0,09 + 0,01 + 0,14} = \sqrt{0,44}$$

$$d_{13} = \sqrt{0,01 + 0,01 + 0,14 + 0,01} = \sqrt{0,32}$$

$$d_{18} = \sqrt{0,14 + 0,12 + 0,05 + 0,09} = \sqrt{0,44}$$

$$d_{16} = \sqrt{0,05 + 0 + 0,14 + 0,09} = \sqrt{0,44}$$

$$d_{19} = \sqrt{0,14 + 0,14 + 0,09 + 0,05} = \sqrt{0,49}$$

$$d_{17} = \sqrt{0,09 + 0,01 + 0,14 + 0,14} = \sqrt{0,44}$$

min \rightarrow BMU of Neuron 1

نیز اخراج مجموعه

$$\omega_{(i,j)} = \omega_{(i,j)} + \alpha \times h_{(i,j),t} \times (X - \omega_{(i,j)})$$

$$-\frac{\partial \omega_{(i,j)}}{\partial \text{sigma}(t)} e^{-\frac{X}{\text{sigma}(t)}}$$

1 2 3

$$h_{(i,j),t} = e^{-\frac{t}{\text{sigma}(t)}}$$

4 5 6

$$\text{sigma}(t) = \text{sigma}_0 \times e^{-\frac{t}{\tau}}$$

7 8 9

$$d = \sqrt{1+1} = \sqrt{2}$$

$t=1$, BMU of Neuron 1 \rightarrow نیز 5, 4, 2, 1 نیز
می شوند (با وجود بیش از یک نیز)

$$\text{sigma}(1) = e^{-\frac{1}{\tau}}$$

$$\text{Neuron 1: } [0,11, 0,14, 0,14, 0,14, 0,14] = [0,1, 0,1, 0,1, 0,1, 0,1] + 0,1 \times \underbrace{e^{-\frac{1}{\tau}}}_{\text{sigma}(1)} ([0,1, 0,1, 0,1, 0,1, 0,1])$$

$$\text{Neuron 2: } [0,1, 0,1, 0,1, 0,1, 0,1, 0,1] = [0,1, 0,1, 0,1, 0,1, 0,1, 0,1] + 0,1 \times \underbrace{e^{-\frac{1}{\tau}}}_{\text{sigma}(1)} ([0,1, 0,1, 0,1, 0,1, 0,1, 0,1])$$

$$\hookrightarrow h = e^{-\frac{(1)^2}{\text{sigma}(1)}} = e^{\frac{1}{\text{sigma}(1)}} = e^{-\frac{1}{\text{sigma}(1)}} = 0,14$$

$$\text{Neuron 9: } [0.3972, 0.43079, 0.1058] = [0.2, 0.4, 0.1, 0.1] + 0.1 \times 0.14 \times [-0.2, 0, -0.1, 0.1], \quad 10$$

$$\text{Neuron 5: } [0.8928, -0.4918, 0.11113] = [0.2, 0.4, 0.1, 0.1] + 0.1 \times 0.14 \times [-0.4, -0.1, -0.1, 0.4], \quad 11$$

* در اصل بعدها به دست آوران β MU نامیده شد

$$w^T = [0.2, 0.4, 0.1, 0.1] \rightarrow \beta MUs \text{ Neuron 2} \quad 12$$

$$d_{11} = 0.002$$

$$d_{12} = 1.039$$

$$d_{13} = 0.015 \rightarrow \min$$

$$d_{14} = 1.039$$

$$d_{15} = 0.0048$$

$$d_{16} = 0.721$$

$$d_{17} = 0.0207$$

$$d_{18} = 0.114$$

$$n(i, j, t) = e^{-\frac{d(i, j)}{\tau_{\text{sigma}}(t)}} \quad \begin{cases} 1 & \text{if } i = j \\ e^{0.5} & \text{if } i = 1, 2, \dots, 6 \\ 0 & \text{otherwise} \end{cases} \quad \text{Neuron 2} \quad 13$$

$$\text{sigma}(y) = e^{-\frac{y}{\tau}} \rightarrow \text{sigma}(y) = e^{-\frac{y}{\tau}} \quad \begin{cases} 1 & \text{if } y = 0 \\ e^{-\frac{y}{\tau}} & \text{if } y = 1, 2, \dots, 6 \\ 0 & \text{otherwise} \end{cases} \quad \text{Neuron 1, ..., 6 + 2} \quad 14$$

$$\text{Neuron 1: } [0.11, 0.392, 0.43079, 0.1058] = [0.11, 0.39, 0.43, 0.11] + 0.1 \times [0.29, -0.13, 0.44, -0.11] \quad 15$$

$$\text{Neuron 2: } [0.29, 0.411, 0.43079, 0.1058] = [0.11, 0.39, 0.43, 0.11] + 0.1 \times [0.29, -0.13, 0.44, -0.11] \quad 16$$

$$\text{Neuron 3: } [0.11, 0.392, 0.43079, 0.1058]$$

$$\text{Neuron 4: } [0.29, 0.411, 0.43079, 0.1058]$$

$$\text{Neuron 5: } [0.11, 0.392, 0.43079, 0.1058]$$

$$\text{Neuron 6: } [0.29, 0.411, 0.43079, 0.1058]$$

8. $\alpha \in [0, 0, 0, 0, t, 0, 0]$

$t = 1$

9. $d_{min} = 0.11^{\circ}$. $\sin \theta \rightarrow \text{BMU} \approx \text{Neuron 9}$

میانگین بین ۵، ۶، ۷، ۸، ۹ نهاده شد.

10. $h(i, j, t) = \begin{cases} 1 & \text{Neuron 9} \\ e^{-\frac{1}{(i-9)^2}} & \leq 0.0004 \end{cases}$

0.0004

11. $\text{Signal}(3)_{SC}^{-1} = 0.14$

12. Neuron 9: $[0, 1V9, 0, 210, 0, 8, 0, 299]$

13. Neuron 5: $[0, 872, 0, 2V4, 0, 118, 0, 228]$

14. Neuron 6: $[0, 9, 0, 2V4, 0, 118, 0, 218]$

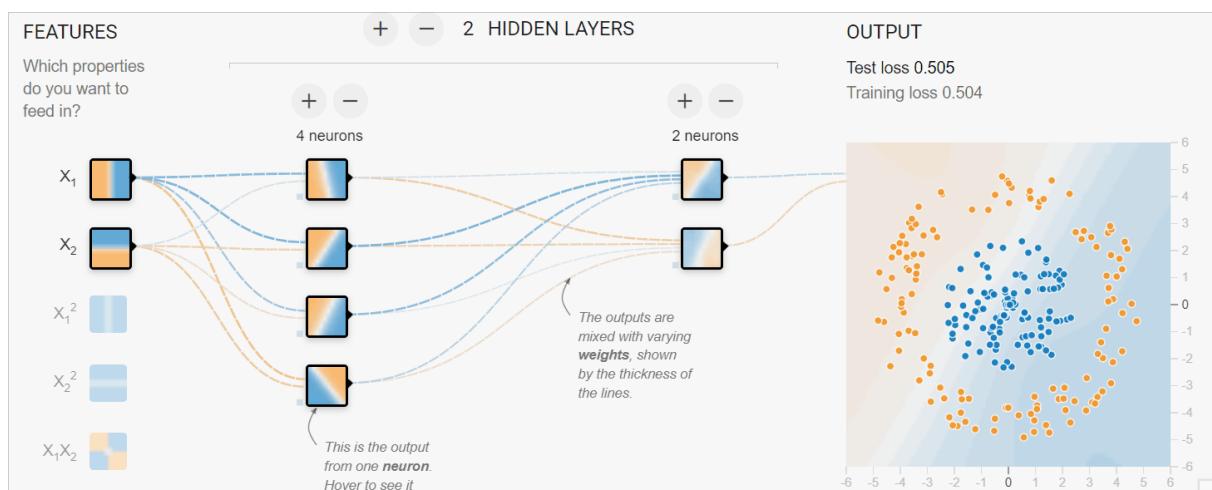
15. Neuron 8: $[0, V98, 0, 101, 0, 304, 0, 204]$

پاسخ سوال 8:

شبکه‌های عصبی در واقع به دلیل ماهیت جعبه سیاه خود شناخته شده‌اند، اما روش‌های مختلفی وجود دارد که می‌تواند به کسب بینش در مورد عملکرد درونی آنها کمک کند. باید برخی از این تکنیک‌ها را بررسی کنیم:

1. تکنیک‌های تجسم⁶: هدف روش‌های تجسم ارائه بازنمایی بصری از آنچه شبکه عصبی آموخته است. این تکنیک‌ها می‌توانند به درک ویژگی‌ها و الگوهای آموخته شده کمک کنند. برخی از تکنیک‌های متداول تجسم عبارتند از:

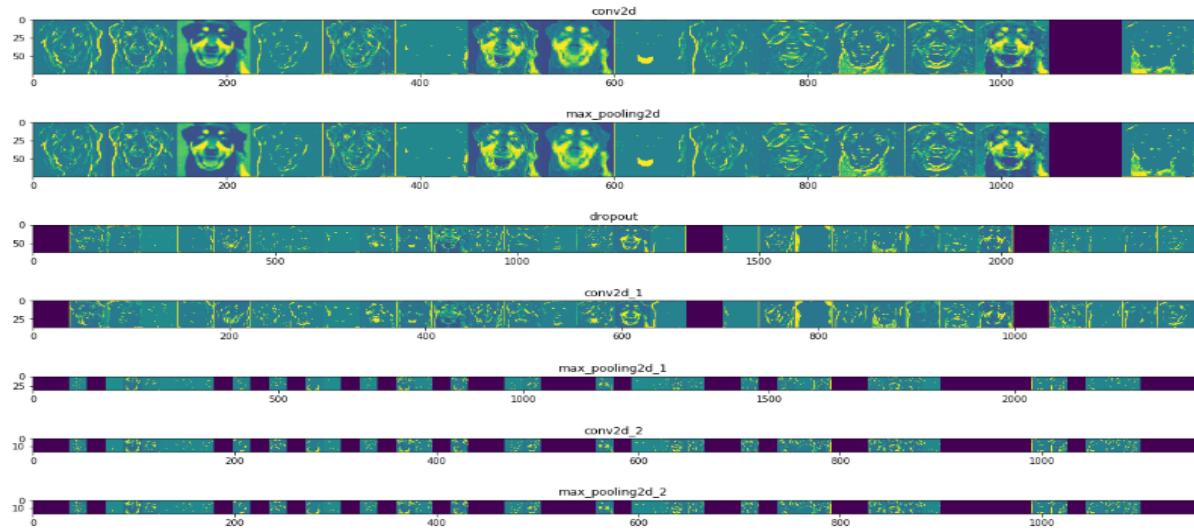
- تجسم فعال سازی⁷: این شامل تجسم فعال شدن تک تک نورون‌ها یا گروه‌های عصبی برای درک پاسخ آنها به ورودی‌های مختلف است. این کمک می‌کند تا مشخص شود کدام بخش از داده‌های ورودی برای نورون‌های خاص مرتبط است.



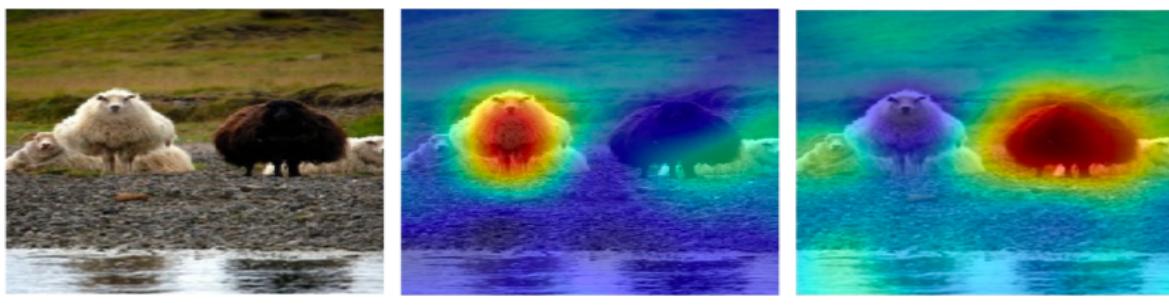
- Filter Visualization: شامل تولید تصاویری است که فعال شدن فیلترهای خاص را در شبکه به حداقل می‌رساند. با تجسم این فیلترها، محققان می‌توانند بینش‌هایی را در مورد انواع ویژگی‌هایی که شبکه شناسایی می‌کند به دست آورند.

⁶ Visualization Techniques

⁷ Activation Visualization



- تجسم نقشه حرارتی⁸: نقشه های حرارتی را می توان برای برجسته کردن مناطقی از یک تصویر ورودی که بیشترین سهم را در خروجی شبکه دارد، ایجاد کرد. این درک از مناطقی که در فرآیند تصمیم گیری شبکه تاثیرگذار هستند را فراهم می کند.



(a) Sheep - 26%, Cow - 17% (b) Importance map of 'sheep' (c) Importance map of 'cow'



(d) Bird - 100%, Person - 39% (e) Importance map of 'bird' (f) Importance map of 'person'

⁸ Heatmap Visualization

2. نقشه های ویژگی⁹: نقشه های ویژگی خروجی های میانی لایه های شبکه عصبی هستند. آنها ویژگی های آموخته شده را در سطوح مختلف انتزاع نشان می دهند. تجزیه و تحلیل این نقشه های ویژگی می تواند اطلاعات مهمی را در مورد آنچه شبکه آموخته است، آشکار کند. برخی از روش ها برای کاوش نقشه های ویژگی عبارتند از:

- حداکثر سازی فعال سازی¹⁰: با به حداکثر رساندن فعال سازی نورون های خاص در نقشه های ویژگی، محققان می توانند الگوهای ورودی ای را ایجاد کنند که آن نورون های خاص را به شدت فعال می کند. این به کشف انواع الگوهایی که یک نورون به آنها حساس است کمک می کند.

- نگاشت فعال سازی کلاس¹¹: نگاشت فعال سازی کلاس مناطقی را که بیشترین سهم را در پیش‌بینی شبکه برای یک کلاس خاص دارند، بر جسته می کند. اینکار با تجزیه و تحلیل نقشه های ویژگی لایه کانولوشنال نهایی، متمایزترین مناطق را در یک تصویر ورودی شناسایی می کند.

3. انتشار اتصال لایه ای¹²: تکنیک های انتشار اتصال لایه ای با هدف ردیابی تأثیر ویژگی های ورودی تکی از طریق لایه های شبکه انجام می شود. با شناسایی تأثیرگذارترین ویژگی ها، می توانیم بفهمیم که چگونه اطلاعات در شبکه جریان می یابد و تغییر می کند. Grad-CAM (نگاشت فعال سازی کلاس با وزن گرادیان) نمونه ای از این تکنیک است.

علیرغم مفید بودن، این تکنیک ها دارای محدودیت ها و چالش های خاصی هستند:

- تفسیرپذیری در مقابل عملکرد: برخی از روش های تفسیرپذیری، مانند تجسم ها یا نقشه های ویژگی، ممکن است همیشه با فرآیند تصمیمگیری واقعی شبکه هماهنگ نباشند. شبکه ها اغلب بازنمایی های پیچیده ای را می آموزند که تفسیر مستقیم آنها دشوار است. بنابراین، تفسیرپذیری می تواند به قیمت عملکرد تمام شود.

- پیچیدگی شبکه های عمیق: تفسیر شبکه های عصبی عمیق با لایه های متعدد و میلیون ها پارامتر می تواند چالش بزرگیز باشد. با افزایش پیچیدگی شبکه، درک عملکرد درونی آن دشوارتر می شود.

⁹ Feature Maps

¹⁰ Activation Maximization

¹¹ Class Activation Mapping

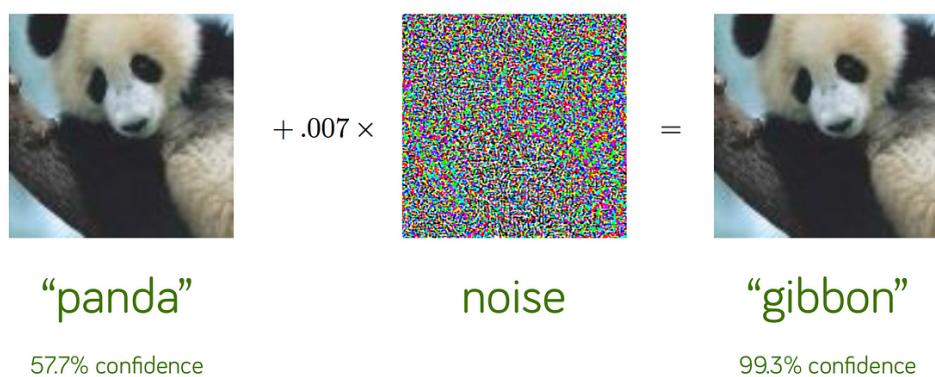
¹² Layered Connection Propagation

- تعمیم به برنامه‌های کاربردی دنیای واقعی: در حالی که روش‌های تفسیرپذیری بینش‌های را در مورد رفتار شبکه ارائه می‌دهند، تعمیم این بینش‌ها به برنامه‌های کاربردی دنیای واقعی می‌تواند غیر ضروری باشد. شبکه‌های عصبی بازنمایی‌های پیچیده را یاد می‌گیرند و تفسیر ممکن است همیشه به صورت یکپارچه به مجموعه داده‌ها یا سناریوهای مختلف منتقل نشود.
- کاربردهای واقعی این روش‌ها عبارتند از:
- تصویربرداری پزشکی¹³: تکنیک‌های تجسم و نقشه‌های ویژگی به رادیولوژیست‌ها و متخصصان پزشکی کمک می‌کنند تا فرآیند تصمیم‌گیری مدل‌های یادگیری عمیق را در وظایف تصویربرداری پزشکی مانند تشخیص تومور، تقسیم‌بندی اندام‌ها و تشخیص بیماری تفسیر و درک کنند.
 - رانندگی خودمختار¹⁴: با تفسیر پذیری و تجسم توجه سیستم‌های رانندگی خودمختار در مناطق مختلف یک تصویر ورودی، محققان می‌توانند بینشی در مورد آنچه که شبکه در هنگام تصمیم‌گیری روی آن تمرکز می‌کند، مانند تشخیص شی یا تشخیص خط به دست آورند.
 - پردازش زبان طبیعی¹⁵: روش‌های تفسیرپذیر برای درک فرآیند تصمیم‌گیری مدل‌های زبان در کارهایی مانند تحلیل احساسات، طبقه‌بندی متن و ترجمه ماشینی استفاده می‌شوند. آنها به شناسایی کلمات یا عبارات مهمی که به پیش‌بینی مدل کمک می‌کنند، کمک می‌کنند.
- در حالی که تفسیرپذیری یک حوزه فعال تحقیقاتی است، این تکنیک‌ها بینش‌های ارزشمندی در مورد شبکه‌های عصبی و کاربردهای واقعی آن‌ها ارائه می‌کنند و شکاف بین عملکرد و قابلیت تفسیر را پر می‌کنند.
- پاسخ سوال ۹:**
- حملات خصم‌مانه چالش مهمی برای شبکه‌های عصبی ایجاد می‌کند، اما چندین تکنیک وجود دارد که می‌توان برای دفاع در برابر آنها استفاده کرد. بیایید برخی از این تکنیک‌ها را بررسی کنیم:

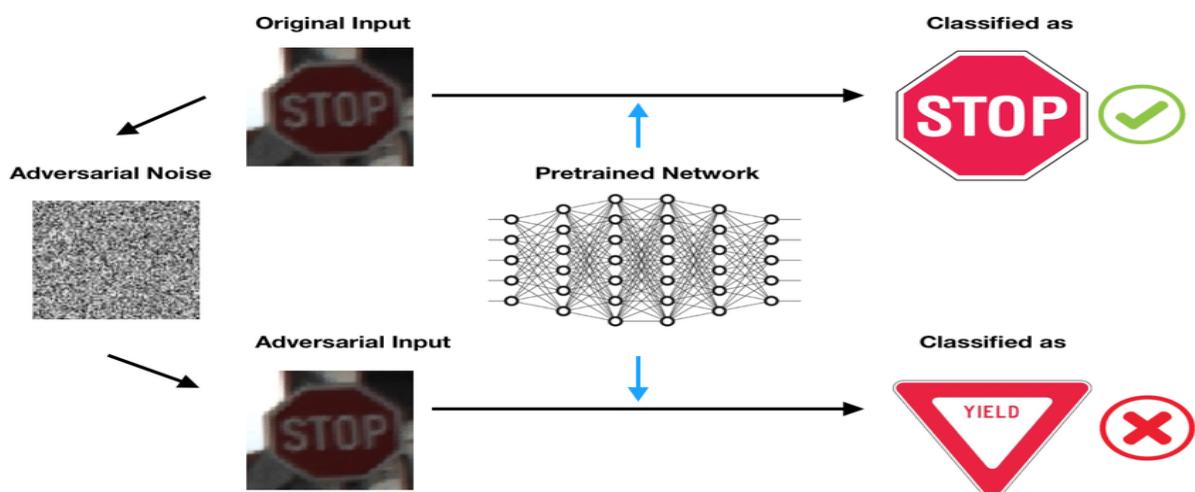
¹³ Medical Imaging

¹⁴ Autonomous Driving

¹⁵ Natural Language Processing



۱. آموزش دشمن^{۱۶}: آموزش دشمن، که به عنوان آموزش خصمانه یا تقطیر دفاعی^{۱۷} نیز شناخته می‌شود، شامل تقویت فرآیند آموزش با مثال‌های خصمانه^{۱۸} است. نمونه‌های متخاصم با اضافه کردن اغتشاشات و نویزهای کوچک به داده‌های ورودی برای گمراه کردن شبکه تولید می‌شوند. با گنجاندن این نمونه‌های متخاصم در مجموعه آموزشی، شبکه یاد می‌گیرد که در برابر چنین حملاتی قوی باشد. این تکنیک با قرار دادن آن در معرض طیف وسیع تری از تغییرات ورودی، تعمیم شبکه را بهبود می‌بخشد.



- اثربخش و محدودیت‌ها: آموزش دشمن می‌تواند به طور قابل توجهی استحکام شبکه‌های عصبی را در برابر حملات خصمانه بهبود بخشد. با این حال، ممکن است از

¹⁶ Adversary Training

¹⁷ Defensive Distillation

¹⁸ Adversarial Examples

تعمیم محدود¹⁹ رنج ببرد، به این معنی که شبکه در برابر مجموعه‌ای از حملات قوی می‌شود

اما در برابر حملات غیرقابل مشاهده یا پیچیده‌تر آسیب‌پذیر می‌ماند. علاوه بر این، آموزش

حریف می‌تواند هزینه‌های محاسباتی و زمان آموزش را افزایش دهد.

- سناریوی دنیای واقعی: آموزش دشمن در سیستم‌های رانندگی خودمختار برای افزایش

انعطاف‌پذیری آن‌ها در برابر حملات متخصص استفاده شده است. با آموزش مدل‌ها با

مثال‌های متخصص که آشفتگی‌ها را در علائم راهنمایی و رانندگی یا علامت‌گذاری جاده‌ها

شبیه‌سازی می‌کنند، مدل‌ها در برابر حملات احتمالی با هدف دستکاری سیستم ادراک

قوی‌تر می‌شوند.

2. پیش پردازش ورودی²⁰: تکنیک‌های پیش پردازش ورودی شامل اصلاح داده‌های ورودی قبل از

تغذیه آن‌ها به شبکه عصبی برای حذف یا کاهش تأثیر اغتشاشات متخصص است. نمونه‌هایی از

تکنیک‌های پیش پردازش ورودی عبارتند از:

- تقطیر دفاعی: تقطیر دفاعی شامل آموزش یک مدل جداگانه بر روی احتمالات نرم شده

خروجی²¹ داده شده توسط شبکه اصلی است. این تکنیک با وارد کردن عدم قطعیت بیشتر

در پیش‌بینی‌های شبکه، شبکه را در برابر حملات متخصص مقاوم‌تر می‌کند.

- تبديل ورودی: تکنیک‌های تبدیل ورودی، داده‌های ورودی را به گونه‌ای تغییر می‌دهند که

اثربخش اختلالات متخصص را کاهش دهد. این روش می‌تواند شامل تکنیک‌هایی مانند

اضافه کردن نویز، محو کردن تصویر یا استفاده از الگوریتم‌های حذف نویز برای حذف

اختلالات متخصص باشد.

- اثربخش و محدودیت‌ها: تکنیک‌های پیش‌پردازش ورودی می‌توانند لایه‌های دفاعی

بیشتری در برابر حملات خصم‌مانه فراهم کنند. با این حال، آنها ممکن است بی خطا نباشند

و گاهی اوقات می‌توانند عملکرد شبکه را در داده‌های تمیز کاهش دهند. حملات پیچیده

هنوز هم می‌توانند راه‌هایی برای دور زدن این تکنیک‌های پیش‌پردازش پیدا کنند.

¹⁹ Limited Generalization

²⁰ Input Preprocessing

²¹ Softened Output Probabilities

- سناریوی دنیای واقعی: تکنیک های پیش پردازش ورودی در سیستم های تشخیص چهره برای کاهش حملات خصم‌مانه بررسی شده است. با اعمال تغییرات در تصاویر چهره، مانند اضافه کردن نویز تصادفی یا اعمال فیلترهای تصویر، سیستم ها در برابر حملاتی که ویژگی های چهره را دستکاری می‌کنند یا اختلالات نامحسوس اضافه می‌کنند، قوی‌تر می‌شوند.
- 3. تنظیم مدل²²: تکنیک های تنظیم مدل شامل ایجاد تغییراتی در معماری شبکه یا فرآیند آموزش برای بهبود استحکام آن در برابر حملات متخصص است. برخی از تکنیک ها عبارتند از:
 - منظم سازی²³: تکنیک های منظم سازی، مانند منظم سازی L1 یا L2، می‌تواند به کاهش حساسیت مدل به اختلالات ورودی کوچک کمک کند. منظم سازی، شبکه را تشویق می‌کند تا بازنمایی‌های قوی‌تر و پایدارتر را بیاموزد.
 - درج لایه دفاعی²⁴: لایه‌های دفاعی، مانند لایه‌های تزریق نویز متخصص²⁵ یا لایه‌های پوشاننده گرادیان²⁶، می‌توانند به معماری شبکه اضافه شوند تا انعطاف‌پذیری آن در برابر حملات دشمن (متخصص) افزایش یابد. این لایه‌ها نویز یا ابهامات اضافی را به نمایش‌های داخلی شبکه وارد می‌کنند و سوءاستفاده از آسیب‌پذیری‌ها را برای مهاجمان سخت‌تر می‌کنند.
 - اثربخشی و محدودیت‌ها: تکنیک‌های تنظیم مدل می‌توانند استحکام شبکه را در برابر حملات متخصص بهبود بخشد، اما ممکن است از نظر عملکرد مدل یا پیچیدگی محاسباتی با یکدیگر معاوضه²⁷ داشته باشند. مهاجمان پیچیده هنوز هم می‌توانند راههایی برای دور زدن این دفاع‌ها پیدا کنند، زیرا مکانیسم‌های دفاعی ممکن است دارای نقاط ضعف قابل سوءاستفاده باشند.
 - سناریوی دنیای واقعی: تکنیک‌های تنظیم مدل در سیستم‌های رانندگی خودمختار برای افزایش انعطاف‌پذیری آن‌ها در برابر حملات خصم‌مانه بررسی شده‌اند. با ترکیب لایه‌های دفاعی یا تکنیک‌های منظم‌سازی، مدل‌ها در برابر حملات با هدف دستکاری سیستم ادراک یا معرفی سیگنال‌های ورودی گمراه‌کننده قوی‌تر می‌شوند.

²² Model Tuning

²³ Regularization

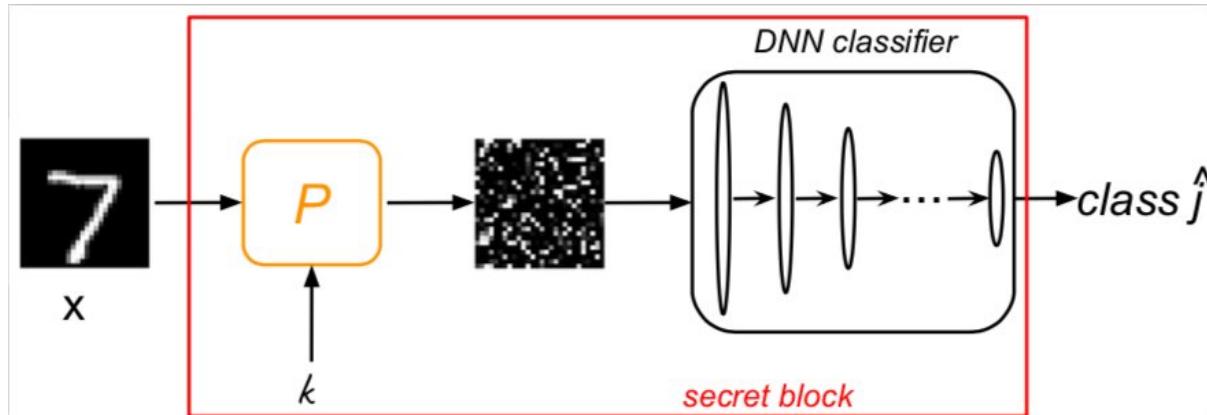
²⁴ Defensive Layer Insertion

²⁵ Adversarial Noise Injection Layers

²⁶ Gradient Masking Layers

²⁷ Trade-off

توجه به این نکته مهم است که هیچ تکنیک دفاعی، اینمی کاملی در برابر حملات متخاصل ایجاد نمی کند. حملات خصمانه به تکامل خود ادامه می دهند و مکانیسم های دفاعی جدیدی به طور مداوم برای مقابله با آنها در حال توسعه هستند. اثربخشی تکنیک های دفاعی به سناریوی حمله خاص و سطح پیچیدگی مهاجم بستگی دارد. بنابراین، ترکیبی از تکنیک های دفاعی متعدد و تحقیقات مداوم در مورد استحکام خصمانه برای کاهش تأثیر حملات خصمانه بر شبکه های عصبی، بسیار مهم خواهد بود.



پاسخ سوال 10:

کیفیت خوش بندی با توجه به تشکیل خوش بندی هر ورودی پایین است. (تعداد ورودی ها کم بوده است) برای بهبود دقت خوش بندی SOM، می توانید ویژگی های زیر را در نظر بگیرید:

1. افزایش اندازه SOM: افزایش اندازه شبکه SOM می تواند ظرفیت بیشتری برای نمایش فضای ورودی فراهم کند. یک شبکه بزرگتر امکان نمایش بالقوه بهتر الگوهای ورودی را فراهم می کند. با این حال، افزایش اندازه شبکه، پیچیدگی محاسباتی را نیز افزایش می دهد.
2. تنظیم پارامترهای یادگیری: میزان یادگیری و اندازه شعاع همسایگی از پارامترهای حیاتی در آموزش SOM هستند. آزمایش با مقادیر مختلف برای این پارامترها می تواند دقت خوش بندی را بهبود ببخشد. کاهش سرعت یادگیری می تواند منجر به همگرایی کندر اما پایدارتر شود در حالی که تنظیم اندازه همسایگی می تواند بر میزان تعامل بین نورون ها در طول فرآیند یادگیری تأثیر بگذارد.
3. افزایش تکرارهای آموزش: اجرای آموزش SOM برای تکرارهای بیشتر به شبکه اجازه می دهد تا نمایش الگوهای ورودی را اصلاح کند. افزایش تکرارهای آموزشی به طور بالقوه می تواند به دقت خوش بندی بهتر منجر شود.

4. عادی سازی داده های ورودی: اگر داده های ورودی دارای مقیاس ها یا واحدهای متفاوتی باشند، عادی سازی بردارهای ورودی می تواند به بهبود عملکرد خوشه بندی کمک کند. عادی سازی تضمین می کند که هر ویژگی به طور یکسان در فرآیند آموزش SOM مشارکت دارد و از بایاس ناشی از تفاوت در مقیاس ها جلوگیری می کند.

5. مهندسی ویژگی ها: گاهی اوقات، کیفیت خوشه بندی را می توان با انتخاب دقیق یا مهندسی ویژگی ها از داده های ورودی بهبود بخشید. شناسایی آموزنده ترین یا مرتبط ترین ویژگی ها و تمرکز بر آنها در طول آموزش می تواند به نتایج خوشه بندی بهتری منجر بشود.