فرهاد امان ۹۹۳۱۰۰۶

۱-

a) مسئله‌ی یافتن خروجی XOR یک مسئله‌ی دسته‌بندی دودویی است. برای این مسئله، می‌توانیم نمودار داده‌ها را در نظر بگیریم. محور Y برای ورودی X1 و محور X برای ورودی X2 قرار دارد. خروجی XOR را می‌توان با استفاده از یک پرسپترون و رابطه‌ی زیر محاسبه کرد:

خروجی = w1 * X1 + w2 * X2 + b

در اینجا، خطی را در نظر می‌گیریم. اما به وضوح هیچ خطی وجود ندارد که دایره‌ها را از ضربدرها جدا کند. بنابراین، با استفاده از یک پرسپترون نمی‌توانیم این مسئله را حل کنیم.

از جدول ارزش‌های OR و AND می‌دانیم که می‌توانیم آنها را با استفاده از یک پرسپترون و به شکل خطی از یکدیگر جدا کنیم. بنابراین، می‌توانیم عملگر XOR را بر اساس عملگرهای AND و OR تعریف کنیم:

A XOR B = (A AND (NOT B)) OR (B AND (NOT A))

برای انجام این کار، به تعداد استفاده شده از عملگرهای AND و OR نیاز به پرسپترون داریم.

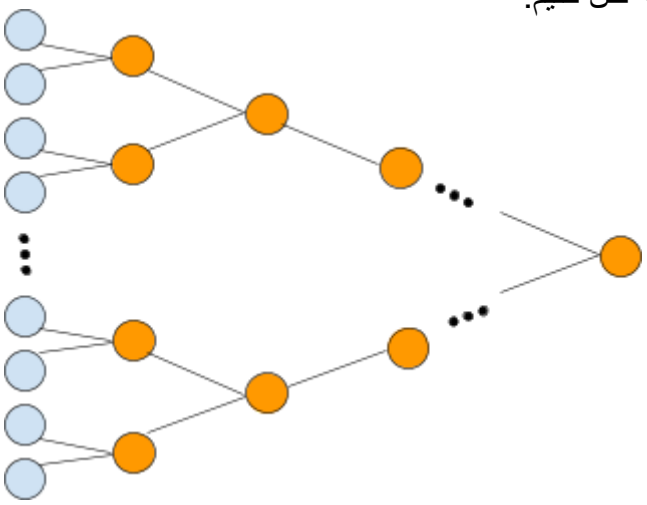
با استفاده از سه پرسپترون، می‌توانیم تابع XOR را ایجاد کنیم:

Y = f(w1 * h1 + w2 * h2)

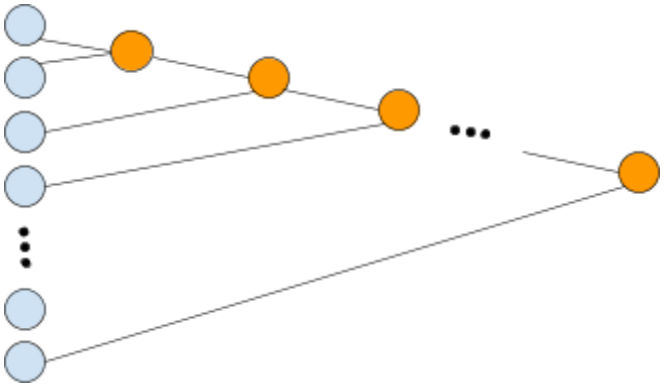
Y = f(w1 * f(wx11 * (1 - X1) + wx21 * X2) + w2 * f(wx12 * X1 + wx22 * (1 - X2)))

به این ترتیب، می‌توانیم مسئله‌ی XOR را با استفاده از عملگرهای AND و OR و پرسپترون‌های مربوطه حل کنیم.

(b



(c



(d

با فرض دو متغیر X1 و X2، فرم SOP برای آن‌ها به صورت زیر است:

X1X2 = X1X2' + X2X1'

حال فرض کنید برای سه متغیر با استفاده از پرانتز گذاری، قسمت پ را برابر با XOR بین X1 و X2 محاسبه کرده‌ایم و می‌خواهیم آن را جایگزین کنیم.

Z = X1X2 = X1X2' + X2X1'

(X1X2)X3 = ZX3 = ZX3' + X3Z' = (X1X2' + X2X1')X3' + X3(X1X2' + X2X1')'

= X1X2'X3' + X2X1'X3' + X3(X1X2' + X2X1')' = X1X2'X3' + X2X1'X3' +

X3(X1' + X2)(X2' + X1)

= X1X2'X3' + X2X1'X3' + X3X1'X2' + X3X1X2 + X3X1'X1 + X3X2X2'

= X1X2'X3' + X2X1'X3' + X3X1'X2' + X3X1X2

به این ترتیب، می‌توانیم با استفاده از پرانتز گذاری و معادله‌ی فرم SOP، محاسبه‌ی XOR بین X1 و X2 را برای سه متغیر X2 ،X1 و X3 به‌دست آوریم.

برای هر n متغیر، می‌توان فرم SOP را نوشت و خروجی را در دو مرحله محاسبه کرد. در مرحله اول، محصول‌ها (product) محاسبه می‌شوند و در مرحله دوم، یک جمع (summation) بین تمام خروجی‌های قبلی انجام می‌شود. محاسبه محصول‌ها را به عنوان لایهٔ مخفی شبکه عصبی در نظر می‌گیریم و محاسبه جمع را نیز به عنوان لایهٔ خروجی در نظر می‌گیریم.

در جبر بولی، برای بررسی درستی عبارت در صورتی که k متغیر از n متغیر 1 باشند، به تعداد جملات C(n, r) نیاز داریم. در هر جمله، r تا از n متغیر را فرض می‌کنیم برابر با 1 و n-r تا را فرض می‌کنیم برابر با صفر و نقیض آن‌ها را می‌گذاریم.

تفسیر XOR بین n متغیر این است که تعداد فردی از آنها 1 باشند. بنابراین، در فرم SOP آن، مجموع جملات C(n, 1) + C(n, 3) + ... + C(n, n) وجود دارد. هر جمله یک پرسپترون است.

C(n, 1) + C(n, 3) + ... + C(n, n) = 2n - 1

۲-

(a

لایه پنهان اول:

A1 = tanh(0.5 * 4) = 0.964

A2 = tanh(0.5 * 4) = 0.964

A3 = tanh(0.5 * 4) = 0.964

A4 = tanh(0.5 * 4) = 0.964

لایه پنهان دوم:

D1 = tanh(0.964 * 0.5 * 4) = 0.958

D2 = tanh(0.964 * 0.5 * 4) = 0.958

D3 = tanh(0.964 * 0.5 * 4) = 0.958

لایه پنهان سوم:

E1 = swish(0.958 * 3 * 0.5 + 0.964 * 0.5, 1) = 1.674
E2 = swish(0.958 * 3 * 0.5, 1) = 1.161

خروجی نهایی:

Y = sigmoid(1.161 * 0.5 + 1.674 * 0.5)
Y = sigmoid(0.580 + 0.837)
Y = sigmoid(1.418) = 0.8050517369790207926706

(b

C = 1/2(y - 1)^2
Cy = -1 + y = -0.19495
d sigmoid(x)/dx = x(1 - x)
d swish(x, 1)/dx = x + x(1 - x)
YZ4 = Z4(1 - Z4) = 0.80505 * 0.19495 = 0.156944
Z4E1 = 0.5
E1Z3 = Z3 + Z3 * (Z3)(1 - Z3) = 0.87212 + 1.919877 * 0.87212 * 0.12788 = 1.0862
Z3WA2E1 = 0.958575
CWA2E1 = -0.19495 * 0.156944 * 0.5 * 1.0862 * 0.958575 = -0.01592
WA2E1 = WA2E1 - CWA2E1 = 0.5 - 0.1 * (-0.01592) = 0.501592

CWA2D1 = CY * (YZ4 * Z4E1 * E1Z3 * Z3D1 * D1Z2 * Z2WA2D1 + YZ4 * Z4E2 * E2Z3 * Z3D1 * D1Z2 * Z2WA2D1) - 0.0013
CWA2D2 = CY * (YZ4 * Z4E1 * E1Z3 * Z3D2 * D2Z2 * Z2WA2D1 + YZ4 * Z4E2 * E2Z3 * Z3D2 * D2Z2 * Z2WA2D1) - 0.0013
CWA2D3 = CY * (YZ4 * Z4E1 * E1Z3 * Z3D3 * D3Z2 * Z2WA2D1 + YZ4 * Z4E2 * E2Z3 * Z3D3 * D3Z2 * Z2WA2D1) - 0.0013
WA2D1 = WA2D1 - CWA2D1 = 0.5 - 0.1 * (-0.0013) = 0.50013
WA2D2 = WA2D2 - CWA2D2 = 0.5 - 0.1 * (-0.0013) = 0.50013
WA2D3 = WA2D3 - CWA2D3 = 0.5 - 0.1 * (-0.0013) = 0.50013

۳-

(a

Batch gradient descent الگوریتمی است که برای یافتن حداقل تابع هزینه در زمینه گرادیان نزولی استفاده می شود. این شامل استفاده از کل مجموعه داده آموزشی در هر مرحله برای به روز رسانی پارامترها است. با این حال، زمانی که مجموعه داده آموزشی بزرگ است، این الگوریتم می‌تواند از نظر محاسباتی گران باشد زیرا برای هر به‌روزرسانی پارامتر باید محاسبات روی کل مجموعه داده انجام شود. از سوی دیگر کل مجموعه داده را در نظر می گیرد و تابع هزینه را برای همه m نمونه ها در هر تکرار محاسبه می کند، که منجر به منحنی تابع هزینه هموارتر و تضمین یک راه حل بهینه می شود.

در مقابل، (SGD) به‌جای در نظر گرفتن تمام نمونه‌های آموزشی، به‌طور تصادفی یک نقطه داده واحد را در هر مرحله انتخاب می‌کند و پارامترها را بر اساس مقادیر خروجی به‌روزرسانی می‌کند. این روش سریعتر از BGD است و برای مجموعه های تمرینی بزرگ مناسب است. از آنجایی که در هر مرحله فقط یک نمونه در نظر گرفته می شود، تابع هزینه نوسان می کند و ممکن است به صورت یکنواخت کاهش نیابد. با این حال، در دراز مدت، می توان کاهش تابع هزینه و همگرایی را مشاهده کرد. SGD یک راه حل بهینه را تضمین نمی کند، اما راه حل مناسب و قابل قبولی را ارائه می دهد که عدم بهینه بودن آن را جبران می کند.

به طور خلاصه، زمانی که مجموعه داده آموزشی بسیار بزرگ است، از شیب نزولی تصادفی برای همگرایی سریعتر استفاده می شود. با این حال، هنگامی که راه حل بهینه مهم است، و علاوه بر این، زمانی که کاهش صاف و ملایم در تابع هزینه مورد نظر است، BGD ترجیح داده می شود. در الگوریتم min batch، ما هر دو الگوریتم دسته ای و تصادفی را ترکیب می کنیم تا از مزایای هر دو بهره مند شویم. در این رویکرد، محاسبات بر روی کل مجموعه داده آموزشی در هر مرحله انجام نمی شود. در عوض، زیر مجموعه ای از داده های آموزشی استفاده می شود. به طور کلی، با این روش، پارامترها به طور مکرر به روز می‌شوند، پیاده‌سازی برداری شده می تواند برای محاسبات سریعتر استفاده شود، و منحنی تابع هزینه نوسانات کمتری را نشان می دهد و به آرامی به سمت همگرایی حرکت می کند.

(b

با افزایش مقدار λ، اثرگذاری پنالتی در تابع هزینه اهمیت بیشتری پیدا می‌کند و مدل تشویق می‌شود که مقادیر پارامترهای کوچکتری داشته باشد. به عبارت دیگر، با اعمال مدل روی داده‌ها، مدل بیشتر مورد پنالتی قرار می‌گیرد و جریمه می‌شود، که در نتیجه مدل ترغیب می‌شود تا منظم‌سازی (regularization) بیشتری داشته باشد. این منظم‌سازی بیشتر از حالت بیش‌برازش (overfitting) جلوگیری می‌کند که در آن مدل با داده‌های آموزشی عملکرد خوبی دارد، اما با داده‌های واقعی به خوبی عمل نمی‌کند. از طرفی، افزایش بیش از حد مقدار λ باعث می‌شود که مدل کمتر به مقادیر ورودی توجه کند و اثر پنالتی بسیار بالا رفته و ساختار و پیچیدگی داده را در نظر نگیرد. بنابراین، پیدا کردن اندازه مناسب برای مقدار λ اهمیت زیادی در ساخت یک مدل مناسب دارد.

در منظم‌سازی L1، پنالتی برابر با جمع مقادیر مطلق پارامترها است. با این روش، می‌خواهیم پنالتی را کاهش دهیم و برخی از پارامترها را به صفر برسانیم. به عبارت دیگر، با منظم‌سازی L1 می‌توانیم

پارامترهای غیرضروری را حذف کنیم و فقط پارامترهای مهم و تأثیرگذار را نگه داریم. بنابراین، منظم‌سازی L1 معمولاً زمانی استفاده می‌شود که می‌خواهیم ویژگی‌های مهم‌تر مدل را شناسایی کنیم.
در منظم‌سازی L2، پنالتی برابر با جمع مقادیر مربعی پارامترها است و هدف آن کاهش اندازه تمامی پارامترها است، حتی در صورت وجود همبستگی بین آن‌ها.

(c
در روش Dropout، هر نورون در شبکه با احتمال p به طور مستقل از سایر نورون‌ها حذف می‌شود. این حذف شامل تمام اتصالات رو به جلو و عقب با آن نورون است، که باعث ایجاد یک شبکه جدید از شبکه اصلی می‌شود. با احتمال p، خروجی هر نورون در یک متغیر باینری ضرب می‌شود، که به طور تصادفی مقدار 0 یا 1 را با احتمال p و 1-p می‌گیرد. نرخ انصراف p معمولاً بر اساس شکل و اندازه شبکه در بازه‌ای بین 0.2 تا 0.5 تنظیم می‌شود.
با حذف تصادفی نورون‌ها در طول آموزش، شبکه مجبور می‌شود به جای تکیه بیش از حد بر یک ویژگی یا نورون خاص، ویژگی‌های قوی‌تر و قابل تعمیم‌پذیری را بیاموزد. این روش می‌تواند به جلوگیری از بیش‌برازش (overfitting) و بهبود قابلیت تعمیم شبکه با داده‌های جدید کمک کند.
در صورتی که از ابتدا نورون‌ها به صورت کم در نظر گرفته شوند، با undefitting مواجه می‌شویم. به عبارت دیگر، ساختار پیچیده داده و ویژگی‌های آن به طور مناسب و کامل در نظر گرفته نمی‌شوند.

(d
نرخ یادگیری کنترل کننده‌ای برای سرعت تطبیق مدل با مسئله است. نرخ‌های یادگیری کوچکتر به علت تغییرات کوچکتر در وزن‌ها در هر به‌روزرسانی، نیاز به تعداد بیشتری ایپاک دارند. نرخ‌های یادگیری بزرگتر باعث تغییرات سریعتر در وزن‌ها می‌شوند.
استفاده از نرخ یادگیری بسیار بالا می‌تواند منجر به همگرایی سریع مدل به یک راه‌حل غیربهینه شود. همچنین، استفاده از نرخ یادگیری خیلی کوچک باعث کند شدن فرایند همگرایی می‌شود.
در نمودار، X نرخ یادگیری بسیار بالا دارد و تغییرات سریعی در ابتدا دارد که در نقطه‌ای غیربهینه همگرا می‌شود. Y نرخ یادگیری بسیار پایین دارد و تغییرات کمی دارد و هنوز به همگرایی نرسیده است. Z نرخ یادگیری بهینه و خوبی دارد که بین X و Y قرار دارد. تغییرات آن نه به طور خیلی تند است تا در نقطه‌ای غیربهینه گیر نکند و نه به طور خیلی کند است که نیاز به تعداد بیشتری ایپاک برای همگرایی داشته باشیم.
ترتیب نرخ‌های یادگیری: $X > Z > Y$

۴-
(a
در فرمول فاصله میان دو نقطه X و Y در فضای دو بعدی با استفاده از معیار dk(x, y)، دو حالت را بررسی می‌کنیم:

1. وقتی k برابر 1 است (k=1):

در این حالت فاصله منهتن بین دو نقطه مورد نظر محاسبه می‌شود.

فرمول محاسبه فاصله بین X و Y برای این حالت به صورت زیر است:

$$d1(x, y) = (i=1 \text{ to } n) |xi - yi|$$

در اینجا، n تعداد ابعاد فضا است. برای فضای دو بعدی (n=2)، داریم:

$$d1(x, y) = |x1 - y1| + |x2 - y2|$$

اگر فاصله بین دو نقطه در این فرمول برابر با 1 باشد، یعنی مقدار جمع فاصله هر بعد از مبدا برابر با 1 است.

2. وقتی k برابر 2 است (k=2):

در این حالت فاصله اقلیدسی بین دو نقطه محاسبه می‌شود.

فرمول محاسبه فاصله بین X و Y برای این حالت به صورت زیر است:

$$d2(x, y) = ((i=1 \text{ to } n) |xi - yi|^2)^{(1/2)}$$

برای فضای دو بعدی (n=2)، داریم:

$$d2(x, y) = ((x1 - y1)^2 + (x2 - y2)^2)^{(1/2)}$$

اگر فاصله بین دو نقطه در این فرمول برابر با 1 باشد، یعنی مقدار جمع مربعات فاصله هر بعد از مبدا برابر با 1 است.

این فرمول معادل معادله دایره‌ای با شعاع 1 و مرکز مبدا است. در فضای دو بعدی، نواحی تحت پوشش این فرمول یک دایره و در فضای سه بعدی یک کره است.

با استفاده از این فرمول‌ها، می‌توانیم ناحیه تحت پوشش شبکه را در نظر بگیریم و مشخص کنیم که در هر حالت (k=1 و k=2) نقاطی که فاصله‌شان از مبدا 1 است، چگونه توصیف می‌شوند.

(b

در فرمول فاصله بین دو نقطه X و Y در فضای دو بعدی با استفاده از معیار dk(x, y)، حالت k=∞ را بررسی می‌کنیم.

وقتی k به سمت بینهایت میل کند (k=∞)، فاصله Chebyshev بین دو نقطه محاسبه می‌شود.

فرمول محاسبه فاصله بین X و Y برای این حالت به صورت زیر است:

$$d\infty(x, y) = (i=1 \text{ to } n) |xi - yi|\infty = maxi(|xi - yi|)$$

در اینجا، n تعداد ابعاد فضا است. برای فضای دو بعدی (n=2)، داریم:

$$d\infty(x, y) = \max(|x1 - y1|, |x2 - y2|)$$

اگر فاصله بین دو نقطه در این فرمول برابر با 1 باشد، یعنی حداکثر مقدار فاصله در هر بعد از مبدا برابر با 1 است.

با استفاده از این فرمول، می‌توانیم ناحیه تحت پوشش شبکه را در نظر بگیریم و مشخص کنیم که در حالت k=∞، نقاطی که فاصله Chebyshev آن‌ها از مبدا برابر با 1 است، چگونه توصیف می‌شوند. در فضای دو بعدی، نواحی تحت پوشش این فرمول به شکل یک hypercube است، به این صورت

که اگر مختصات یک نقطه به ترتیب x و y باشند، نقاطی که بر روی خطوط شعاعی از ابتدای هر چارچوب اولویت (شامل خطوط $x=\pm1$ و $y=\pm1$) قرار دارند واقع در فاصله 1 از مبدا قرار می‌گیرند.

۵-

ماتریس G با ابعاد M*L است (هر مقدار Gi برای هر M ورودی) و Gij برابر است با
$e^{(-|xi - xj|^2)}$.
ماتریس Y داده‌های خروجی مطلوب برای هر M است و بنابراین ابعاد آن M*2 است.
مقدار L2 regularization برابر است با
$L2(W) = 1/2 * \Sigma(i=1 \text{ to } m) ||Yi - GiW||^2 + 2||W||^2$.
حال می‌خواهیم تابع L2 را نسبت به ماتریس وزن W مینیمم کنیم. بنابراین با مشتق‌گیری جزئی از تابع L2 نسبت به W، می‌توانیم به رابطه‌ی زیر برسیم:
$\nabla L2(W) = 1/2 * \Sigma(-2G^T_i(Yi - GiW)) + 2W = 0$
با جمع کردن تمامی جملات در رابطه بالا، داریم:
$\Sigma(-2G^T(Y - GW)) + MW = 0$
که در آن M ماتریس وزن W را نشان می‌دهد. با جابه‌جایی ماتریس وزن W به سمت راست، به رابطه زیر می‌رسیم:
$G^T(Y - GW) + MW = 0$
با معکوس گرفتن ماتریس $(GTG + I)$، داریم:
$(GTG + I)^{(-1)}G^T(Y - GW) = W$
بنابراین، می‌توانیم برای ماتریس وزن W به رابطه زیر برسیم:
$W = (GTG + I)^{(-1)}G^TY$

۶-

(a

با توجه به ورودی 4x4 و کرنل 3x3، خروجی یعنی ماتریس Y به شکل زیر محاسبه می‌شود:
$Y(1,1) = k11*g(1,1) + k12*g(1,2) + k13*g(1,3) + k21*g(2,1) + k22*g(2,2) + k23*g(2,3) + k31*g(3,1) + k32*g(3,2) + k33*g(3,3)$
$Y(1,1) = 1*(-2) + 3*(-4) + (-4)*(-3) + 0*1 + 1*3 + (-2)*(-2) + 1*(-1) + (-2)*2 + 3*(-2) = 3$
به طور مشابه، سایر عناصر خروجی به صورت زیر محاسبه می‌شوند:
$Y(1,2) = 2*(-2) + (-3)*(-4) + 3*(-3) + (-1)*1 + 2*3 + (-2)*(-2) + 2*(-1) + (-3)*2 + 0*(-2) = 0$
$Y(2,1) = 0*(-2) + (-1)*(-4) + 2*(-3) + (-1)*1 + 2*3 + (-3)*(-2) + 1*(-1) + (-2)*2 + 3*(-2) = 1$

Y(2,2) = 0*(-2) + 3*(-4) + 1*(-3) + (-1)*1 + 1*3 + (-2)*(-2) + 2*(-1) + (-3)*2 + 0*(-2) = 1

بنابراین، ماتریس خروجی Y به شکل زیر است:

3 0
1 1

(b

MSE = C = 1/2 * [(y11 - y~11)^2 + (y12 - y~12)^2 + (y21 - y~21)^2 + (y22 - y~22)^2]
C = 1/2 * [(1 - 1)^2 + (0 - 0)^2 + (0 - 1)^2 + (1 - 1)^2]
Cy11 = y11 - y~11 = 1 - 1 = 0
Cy12 = y12 - y~12 = 0 - 0 = 0
Cy21 = y21 - y~21 = 0 - 1 = -1
Cy22 = y22 - y~22 = 1 - 1 = 0

Ck11 = Cy11 * y11k11 + Cy12 * y12k11 + Cy21 * y21k11 + Cy22 * y22k11 = 0 * 1 + 0 * 2 + (-1) * 0 + 0 * 1 = 0
k11 = 1 - 0.1 * 0 = 1

Ck12 = 0 * 2 + 0 * 3 + (-1) * 1 + 0 * 2 = -1
k12 = -1 - 0.1 * (-1) = -0.9
Ck13 = 0 * 3 + 0 * 3 + (-1) * 2 + 0 * 2 = -2
k13 = 1 - 0.1 * (-2) = 1.2
Ck21 = 0 * 1 + 0 * 2 + (-1) * 1 + 0 * 2 = -1
k21 = -1 - 0.1 * (-1) = -0.9
Ck22 = 0 * 2 + 0 * 3 + (-1) * 2 + 0 * 3 = -2
k22 = 1 - 0.1 * (-2) = 1.2
Ck23 = 0 * 2 + 0 * 2 + (-1) * 3 + 0 * 2 = -3
k23 = -1 - 0.1 * (-3) = -0.7
Ck31 = 0 * 1 + 0 * 2 + (-1) * 1 + 0 * 2 = -1
k31 = 1 - 0.1 * (-1) = 1.1
Ck32 = 0 * 2 + 0 * 3 + (-1) * 2 + 0 * 3 = -2
k32 = -1 - 0.1 * (-2) = -0.8
Ck33 = 0 * 3 + 0 * 0 + (-1) * 3 + 0 * 0 = -3
k33 = 1 - 0.1 * (-3) = 1.3

(c

$$1 \times 1 \times 120 \dashrightarrow 1 \times 1 \times 84 \dashrightarrow 1 \times 1 \times 10$$

-۷

(a

SOM یک الگوریتم رقابتی است. در هر دور تکرار، وزن‌های نورون‌ها با بردار ورودی مقایسه می‌شوند و نورونی که بیشترین شباهت را دارد، برنده می‌شود و به عنوان واحد تطبیق بهتر یا BMU شناخته می‌شود. سپس نورون BMU قادر به یادگیری است و می‌تواند وزن‌های خود را به‌روزرسانی کند تا با ورودی بیشتری تطابق داشته باشد. اما این فرآیند یادگیری تنها به BMU محدود نیست؛ نورون‌های همسایه BMU نیز به‌روزرسانی می‌شوند. تابع همسایگی، تابعی است که نرخ تغییر همسایگان BMU را بر اساس فاصله آن‌ها از BMU تعیین می‌کند. به‌روزرسانی همسایگان دو دلیل اصلی را دارد. اولاً، رابطه توپولوژیکی بین نورون‌ها حفظ می‌شود، به این معنی که نورون‌های نزدیک به هم پاسخ نسبتاً مشابهی به یک ورودی دارند. ثانیاً، SOM در کلاستربندی استفاده می‌شود و به‌روزرسانی همسایگان باعث می‌شود ترانزیشن بین کلاسترها به‌طور نرم صورت گیرد. تابع به‌روزرسانی همسایگی معمولاً کاهشی است، به این معنی که اندازه تغییرات نورون‌های همسایه با افزایش فاصله آن‌ها از نورون برنده کاهش می‌یابد.

(b

نرخ یادگیری، میزان تغییرات در وزن‌ها برای نزدیک شدن به بردار ورودی را تعیین می‌کند. نرخ یادگیری بالا به این معناست که تغییرات سریعتر اتفاق می‌افتد و به سرعت به همگرایی می‌رسیم، اما احتمالاً در مینیمم محلی گیر می‌کنیم و به جواب بهینه نمی‌رسیم. از طرفی، اگر نرخ یادگیری کم باشد، همگرایی به طور قابل توجهی طولانی می‌شود و ممکن است دارای هزینه محاسباتی بالایی باشد. به طور معمول، نرخ یادگیری در ابتدا بزرگتر قرار داده می‌شود تا تغییرات سریعتری اتفاق بیافتد، سپس با گذشت تعداد تکرارها، از نرخ یادگیری کاسته می‌شود و تأثیر داده‌های جدید کمتر می‌شود. این باعث می‌شود بهترین تعمیم‌بندی (generalization) را نسبت به داده‌های جدید داشته باشیم. این کاهش در نرخ یادگیری از بروز overfitting جلوگیری می‌کند.

(c

در ابتدا، به تمام وزن‌ها عددهای تصادفی و کوچک اختصاص داده می‌شود. اگر توزیع این اعداد تصادفی مناسب نباشد، SOM قادر به صحیح درک کردن تمام الگوهای ورودی نمی‌باشد و ممکن است در مینیمم‌های محلی گیر کند. اگر وزن‌ها به درستی توزیع نشوند، می‌تواند منجر به خوشه‌بندی تحت تأثیر انحراف (bias) یا واحد مرده (dead unit) شود.
همچنین، معمولاً وزن‌های اولیه نرمال‌سازی می‌شوند. با نرمال‌سازی وزن‌های اولیه، تمام وزن‌ها به یک اندازه و مرتبط با هم قرار می‌گیرند، که جلوی تسلط یک نورون با وزن بالا بر شبکه را می‌گیرد.
همچنین، نرمال‌سازی باعث می‌شود وزن‌های اولیه در محدوده‌ی معقولی قرار بگیرند و از مشکلاتی مانند اشباع یا از بین رفتن گرادیان‌ها در طول یادگیری جلوگیری شود.

d)
اندازه شبکه SOM با تعداد نورون‌های موجود در شبکه مشخص می‌شود. به طور کلی، یک شبکه با grid بزرگتر نیاز به منابع محاسباتی بیشتری دارد. این به دلیل دو عامل است: اولاً تعداد نورون‌های بیشتر که نیاز به به‌روزرسانی وزن‌های خود دارند، و دوماً فضای حافظه بیشتری که برای نگهداری وزن‌ها مورد نیاز است. همچنین، برای رسیدن به همگرایی، نیاز به تعداد بیشتری iteration داریم. اما از طرفی، تعداد نورون‌های بیشتر قادر به درک ساختارهای پیچیده‌تری از ورودی‌ها می‌باشند و می‌توانند کارایی بیشتری داشته باشند (با این حال، احتمال overfitting نیز بالا می‌رود). به طور کلی، تعداد نورون‌ها وابسته به نیازهای ماست. اگر ساختار ورودی پیچیده است، نیاز به نورون‌های بیشتری داریم و در عین حال باید مراقب overfitting نشدن و هزینه محاسباتی بالای نورون‌های بیشتر باشیم.

۸-
آموزش مقابله با دشمنان یا "Adversarial Training" یک تکنیک استفاده شده برای بهبود قابلیت اطمینان شبکه‌های عصبی در برابر مثال‌های خصمانه است. مثال‌های دشمنانه ورودی‌هایی هستند که به طور قصدی طراحی شده‌اند تا باعث اشتباه شدن شبکه در تشخیص برچسب داده شوند.
ایده پشت آموزش مقابله با دشمنان این است که یک شبکه عصبی را بر روی ترکیبی از مثال‌های سالم و مثال‌های دشمنانه آموزش دهیم. در طول آموزش، شبکه با مثال‌های سالم و دشمنانه مواجه می‌شود و مجبور است به درستی تشخیص دهد. این کار باعث می‌شود که شبکه عصبی به مرور زمان قابلیت اطمینان بیشتری در برابر مثال‌های خصمانه پیدا کند.
فرآیند تولید مثال‌های دشمنانه شامل تغییرات کوچک و نامحسوس بر روی مثال سالم است که به منظور فریب دادن شبکه عصبی طراحی شده‌اند. این کار معمولاً با استفاده از الگوریتم‌هایی مانند روش نشانه گرادیان سریع (FGSM) یا روش کاهش گرادیان پروژه‌ای (PGD) انجام می‌شود.
در آموزش مقابله با دشمنان، شبکه عصبی روی هر دو نوع مثال، یعنی مثال‌های سالم و مثال‌های دشمنانه، آموزش داده می‌شود. هدف این است که تابع هزینه روی هر دو نوع مثال به صورت همزمان کمینه شود. این باعث می‌شود که شبکه به درستی تشخیص دادن هر دو نوع مثال بپردازد.
پیش‌پردازش ورودی:
این روش مربوط به کنترل و تغییر ورودی‌ها قبل از وارد کردن آن داده‌ها به شبکه عصبی است. هدف آن افزایش کیفیت ورودی و بهبود عملکرد و همگرایی مدل است.
تکنیک‌های استفاده شده در این کار:
نرمال‌سازی: نرمال سازی داده‌های ورودی به یک محدوده مشخص، مثلاً بین 0 و 1، تا از غلبه و تسلط بعضی ویژگی‌ها در فرایند یادگیری جلوگیری شود.
استانداردسازی: داده‌های ورودی را به نحوی تغییر دهیم که میانگین آنها 0 و واریانس آنها برابر با یک شود و اینگونه می‌توان گفت که همه ویژگی‌ها به طور یکسان در یادگیری مشارکت دارند.
تغییر مقیاس ویژگی‌ها: تغییر دادن مقیاس ویژگی‌ها برای داشتن محدوده یا توزیع مشابه، که از تسلط برخی ویژگی‌ها جلوگیری می‌کند.
از محدودیت‌های پیش‌پردازش می‌توان به از دست دادن اطلاعات در برخی روش‌ها مانند مقیاس‌بندی ویژگی اشاره کرد. این از دست دادن اطلاعات ممکن است روی توانایی مدل تأثیر بگذارد. همچنین

اثربخشی روش‌ها بسته به مجموعه داده‌ها و نوع مسئله می‌تواند متفاوت باشد. انتخاب مراحل پیش‌پردازش مناسب به دانش و تجربه نیاز دارد.
منظم‌سازی مدل:
"Overfitting" یعنی مدل در یادگیری روی داده‌های آموزشی عملکرد خوبی داشته باشد، اما در مواجه با داده‌های جدید خوب عمل نکند. روش‌های منظم‌سازی با اعمال محدودیت‌ها یا جریمه‌های اضافی بر روی پارامترهای مدل حین مرحله آموزش و یادگیری به کاهش overfitting کمک می‌کنند.
تعدادی از تکنیک‌های منظم‌سازی عبارتند از:
منظم‌سازی L1 و L2: در این منظم‌سازی‌ها یک عبارت جریمه به تابع هزینه اضافه می‌شود که مدل را تشویق می‌کند وزن‌های کوچک‌تری داشته باشد و نیز پیچیدگی مدل کم شود.
Dropout: به طور تصادفی اثر تعدادی از نورون‌ها را در طول تمرین صفر می‌کنیم و نورون‌ها انگار دور انداخته می‌شوند. این کار شبکه را تشویق می‌کند تا نمایش‌های قوی و قابل تعمیم بیشتری را یاد بگیرد.
توقف زودهنگام: مجموعه‌ای از اعتبارسنجی‌ها حین آموزش هستند که منجر به توقف فرآیند آموزش در زمانی می‌شوند که عملکرد شروع به بدتر شدن می‌کند. این کار با پیدا کردن نقطه‌ای که بهترین trade-off را میان آموزش و اعتبارسنجی دارد، از overfitting جلوگیری می‌کند.
البته محدودیت‌هایی نیز وجود دارد. روش‌های منظم‌سازی معمولاً شامل معرفی فرا پارامترهای اضافی به مدل هستند. انتخاب مقادیر مناسب برای این فرا پارامترها می‌تواند پیچیده باشد. برخی از روش‌ها مانند dropout ممکن است تفسیر کارکرد مدل را به علت رفتار تصادفی آن دشوار کنند. منظم‌سازی بیش از حد می‌تواند منجر به underfitting و اعمال منظم‌سازی ناصحیح می‌تواند باعث تغییر در عملکرد و کیفیت مدل شود.

-۱۰
در ارزیابی کیفیت خوشه‌بندی در SOM، می‌توان از معیارهای متنوعی استفاده کرد:
1. خطای کوانتیزاسیون (Quantization Error): این معیار، میانگین فاصله بین هر نقطه داده ورودی و مرکز خوشه در SOM را اندازه‌گیری می‌کند. هرچه این خطا کمتر باشد، خوشه‌بندی با کیفیت‌تری داریم. به عبارت دیگر، کاهش خطای کوانتیزاسیون به معنای بیشترین تطابق داده‌ها با نتایج خوشه‌بندی SOM است.
2. خطای توپوگرافیک (Topographic Error): این معیار نشان می‌دهد که نقشه ساخته شده توانسته است توپوگرافی داده‌ها را به خوبی نمایش دهد یا خیر. اگر خطای توپوگرافیک کم باشد، نشان‌دهنده یک نقشه کاملاً توپوگرافیک است که نقاط داده‌های همسایه در نقشه نزدیک به هم و نقاط داده‌های دور از هم در نقشه دور از هم قرار گرفته‌اند. اما اگر خطای توپوگرافیک بالا باشد، نشان‌دهنده یک نقشه ناهمسانی است که در آن خوشه‌ها در نقشه به طور ناهمگون و ناهمسان قرار گرفته‌اند و نقاط داده‌های همسایه در نقشه دور از هم و نقاط داده‌های دور از هم در نقشه نزدیک به هم قرار گرفته‌اند.
3. ضریب سیلوئت (Silhouette Coefficient): این معیار نسبت بین فاصله داخلی نقاط داده در یک خوشه و فاصله بین نقاط داده در خوشه‌های مجاور را محاسبه می‌کند. برای هر نقطه داده، مقدار ضریب

سیلوئت محاسبه می‌شود و میانگین این مقادیر برای تمام نقاط داده‌ها در خوشه‌بندی SOM به عنوان مقدار کلی ضریب سیلوئت استفاده می‌شود. مقدار بالاتر ضریب سیلوئت نشان‌دهنده خوشه‌بندی بهتر و جداسازی بهتر بین خوشه‌ها است.

4. بررسی بصری: بررسی بصری خوشه‌ها می‌تواند مفید باشد و می‌توان الگوهای مشهود را شناسایی و ارزیابی کرد. در این روش، نقشه SOM را مشاهده کرده و خوشه‌ها را به صورت بصری بررسی می‌کنیم.

علاوه بر اینها، بهبود دقت خوشه‌بندی می‌تواند از طریق اقداماتی مانند بهبود روش‌های مقداردهی اولیه، تعیین تعداد بهینه خوشه‌ها، مدیریت داده‌های با ابعاد بالا، تقویت SOM با تبدیلات غیرخطی و مدیریت داده‌های پرت انجام شود.