دانشکده مهندسی کامپیوتر

گزارش درس مبانی هوش محاسباتی

عنوان گزارش:

تمرین سوم درس هوش محاسباتی (شبکه عصبی)

ارائه‌دهندگان:

فرزانه آقازاده **زینب جنتی** **فاطمه نجفی**

استاد درس:

دکتر حسین کارشناس

دستیاران درس:

رضا برزگر

علی شاه زمانی

آرمان خلیلی

نیم سال دوم ۱۴۰۳-۱۴۰۴

فهرست

سوال اول:

هر نورون در یک شبکه عصبی مصنوعی حامل اطلاعات مشخصی است. این اطلاعات به‌صورت عددی و از طریق وزن‌های مختلف به نورون وارد می‌شود. این وزن‌ها میزان اهمیت هر ورودی را تعیین می‌کنند. در واقع، نورون‌ها در شبکه عصبی به‌طور تجمعی اطلاعات را پردازش کرده و مدل یادگیری را قادر می‌سازند تا به‌تدریج ویژگی‌های پیچیده‌تری را شناسایی کرده و پیش‌بینی‌های دقیقی انجام دهد. هر نورون شامل ورودی ها, وزن ها, توابع جمع, توابع فعالسازی و خروجی است.

ورودی‌ها (Inputs):

ورودی‌ها اطلاعاتی هستند که نورون از منابع مختلف دریافت می‌کند. این ورودی‌ها ممکن است شامل داده‌هایی مانند ویژگی‌های تصویر (برای مثال پیکسل‌ها)، ویژگی‌های متنی (مثل کلمات) یا داده‌های عددی باشند. در شبکه‌های عصبی، این ورودی‌ها معمولاً از نورون‌های قبلی در لایه‌های مختلف شبکه یا داده‌های ورودی اولیه به شبکه می‌آیند.

وزن‌ها (Weights):

در یک نورون، هر ورودی با یک عدد به نام وزن ضرب می‌شود.وزن‌ها اهمیت یا تأثیر هر ورودی را تعیین می‌کنند. به عبارت ساده‌تر، وزن‌ها نشان می‌دهند که یک ورودی چقدر برای نورون مهم است. به عنوان مثال، اگر وزن یک ورودی خیلی بزرگ باشد، این ورودی تأثیر زیادی در تصمیم‌گیری نورون خواهد داشت و بالعکس اگر وزن آن کم باشد، ورودی تأثیر کمتری خواهد داشت. وزن‌ها معمولاً در طول فرآیند آموزش شبکه عصبی تغییر می‌کنند تا شبکه بهترین نتیجه را بدست آورد. اهمیت وزن‌ها در شبکه عصبی به‌طور خودکار و از طریق فرآیند آموزش تعیین می‌شود. در واقع، شبکه عصبی به‌طور خودکار یاد می‌گیرد که کدام ویژگی‌ها یا ورودی‌ها برای پیش‌بینی نهایی مهم‌تر هستند و این را با تنظیم وزن‌ها در طول زمان انجام می‌دهد. این فرآیند از طریق روش‌هایی مانند پیش‌خور (Forward Propagation) و پس‌خور (Backpropagation) انجام می‌شود.

ویژگی‌های بنیادی شبکه‌های عصبی که براساس آن‌ها جواب این سوال را دادیم را این‌طور می‌شود بیان کرد:

یادگیری مبتنی بر وزن‌ها: در فرآیند آموزش، وزن‌های بین نورون‌ها به‌گونه‌ای تنظیم می‌شوند که شبکه بتواند الگوهای موجود در داده‌ها را شناسایی کند. این فرآیند به شبکه امکان می‌دهد تا اطلاعات را به‌صورت مؤثر ذخیره و پردازش کند.

پردازش توزیع‌شده: اطلاعات در سراسر شبکه توزیع می‌شود و هر نورون نقش خاصی در پردازش اطلاعات دارد. این ویژگی باعث می‌شود که شبکه بتواند به‌صورت موازی و مؤثر اطلاعات را پردازش کند.

تابع فعال‌سازی غیرخطی: استفاده از توابع فعال‌سازی غیرخطی مانند ReLU یا سیگموئید به نورون‌ها امکان می‌دهد تا روابط پیچیده و غیرخطی بین ورودی‌ها و خروجی‌ها را مدل‌سازی کنند. این ویژگی برای حل مسائل پیچیده ضروری است.

قابلیت تعمیم: شبکه‌های عصبی قادرند از داده‌های آموزشی یاد بگیرند و دانش حاصل را به داده‌های جدید تعمیم دهند. این قابلیت به‌واسطه ساختار و عملکرد نورون‌ها در شبکه حاصل می‌شود.

در مجموع، هر نورون در شبکه عصبی نقش حیاتی در پردازش و انتقال اطلاعات دارد و عملکرد آن مبتنی بر ویژگی‌های بنیادی مذکور است.

سوال دوم:

در شـبکه‌های عصـبی مصـنوعی، «دانش» به‌صـورت الگوهای عددی در وزن‌ها و بایاس‌های بین نورون‌ها ذخیره می‌شـود. این پارامترها از طریق فرآیند آموزش و بهینه‌سـازی (مانند الگوریتم‌های گرادیان کاهشـی) تنظیم می‌شوند تا شـبکه بتواند روابط پیچیده بین ورودی‌ها و خروجی‌ها را مدل‌سازی کند.

شکل‌گیری دانش در شبکه‌های عصبی

دانش در شبکه‌های عصبی به‌صورت زیر شکل می‌گیرد:

- بازنمایی (Representation):
  شبکه به‌صورت توابع ریاضی وزن‌دار ورودی را به خروجی نگاشت می‌دهد.
- یادگیری (Learning):
  با استفاده از یک تابع هزینه(Loss Function) ، خطای پیش‌بینی محاسبه و وزن‌ها به‌روزرسانی می‌شوند.
- تعمیم (Generalization):
  شبکه باید دانش خود را از داده‌های آموزش به داده‌های جدید تعمیم دهد.

برای فرمول‌بندی کردن دو شبکه عصبی معادل، اول تعریف کنیم که کدام دو شبکه عصبی با یکدیگر معادل‌اند:
دو شبکه عصبی را می‌توان به‌صورت رسمی معادل دانست اگر برای تمام ورودی‌های ممکن، خروجی‌های یکسانی تولید کنند.
یعنی:
برای تمام x در دامنه X ، اگر $F_1(x) = F_2(x)$ باشد، آنگاه شبکه‌های F_1 و F_2 معادل هستند.

در عمل، به‌دلیل پیچیدگی‌های محاسباتی، اغلب از مفهوم «معادل تقریبی» استفاده می‌شود، به‌طوری‌که خروجی‌های دو شبکه برای ورودی‌های مختلف در محدوده‌ای از خطا با یکدیگر نزدیک باشند.

کاربردهای معادل‌سازی شبکه‌های عصبی

بررسی معادل بودن شبکه‌های عصبی در موارد زیر اهمیت دارد:

- **فشرده‌سازی مدل‌ها**: برای کاهش اندازه مدل‌ها بدون از دست دادن دقت.
- **انتقال دانش**: در فرآیندهایی مانند تقطیر دانش، که دانش از یک مدل بزرگ به مدل کوچکتر منتقل می‌شود.
- **تأیید صحت مدل‌ها**: برای اطمینان از اینکه تغییرات در ساختار یا پارامترهای شبکه باعث تغییر رفتار ناخواسته نمی‌شود.

سوال سوم:

توانایی شبکه‌های عصبی در یادگیری، به‌خاطر سپردن و تعمیم، از ویژگی‌های بنیادی آن‌ها ناشی می‌شود که به‌طور خلاصه در ادامه توضیح داده می‌شوند:

۱ .یادگیری(Learning)

شبکه‌های عصبی از طریق فرآیند آموزش، با استفاده از الگوریتم‌هایی مانند گرادیان کاهشی، وزن‌ها و بایاس‌های خود را تنظیم می‌کنند تا بتوانند الگوهای موجود در داده‌ها را شناسایی کنند. این فرآیند به شبکه امکان می‌دهد تا اطلاعات را به‌صورت مؤثر ذخیره و پردازش کند.

۲ .حافظه(Memory)

شبکه‌های عصبی بازگشتی (RNN) و به‌ویژه مدل‌های حافظه بلند-کوتاه‌مدت(LSTM) ، برای پردازش داده‌های ترتیبی طراحی شده‌اند و قادر به حفظ اطلاعات در طول زمان هستند. این ساختارها به شبکه امکان می‌دهند تا وابستگی‌های زمانی را در داده‌ها شناسایی و حفظ کنند.

۳ .تعمیم(Generalization)

تعمیم به توانایی شبکه در اعمال دانش آموخته‌شده به داده‌های جدید و نادیده اشاره دارد. این قابلیت از ویژگی‌هایی مانند ساختار لایه‌ای شبکه، استفاده از توابع فعال‌سازی غیرخطی و تنظیم مناسب وزن‌ها ناشی می‌شود. تحقیقات نشان داده‌اند که شبکه‌های عصبی با ساختار مناسب و آموزش صحیح می‌توانند به‌خوبی تعمیم دهند.

به طور کلی توانایی‌های یادگیری، حافظه و تعمیم در شبکه‌های عصبی از ساختار و ویژگی‌های بنیادی آن‌ها ناشی می‌شود. با طراحی مناسب شبکه و استفاده از تکنیک‌های آموزشی مؤثر، می‌توان این توانایی‌ها را بهینه کرد و شبکه‌هایی با عملکرد بالا ایجاد نمود.

سوال چهارم:

الف) تابع درجه دوم

$$f(x) = ax^2 + \mathrm{b}x + c$$

در اینجا، x ورودی نورون است و a، b و c پارامترهای قابل یادگیری هستند. این تابع می‌تواند به‌عنوان تابع فعال‌سازی یا به‌عنوان بخشی از ساختار نورون استفاده شود. برای مثال، در مدل‌های نورونی خاص، خروجی نورون با استفاده از ترکیب غیرخطی ورودی‌ها محاسبه می‌شود:

$$f(x,y) = Ax^2 + By^2 + Cxy + Dx + Ey + F$$

که در آن x و y ورودی‌های نورون هستند و A تا F پارامترهای قابل یادگیری می‌باشند.

ب) تابع کروی

فرمول:

$$f(x) = ||x||^2 = \sum_{i=1}^{n} x_i^2$$

در این حالت، ورودی‌ها به‌صورت بردار در نظر گرفته می‌شوند و خروجی برابر با مربع طول بردار (نورم اقلیدسی) است.

- **کاربرد**: مسائل تشخیص الگو، به‌ویژه در فضاهای ویژگی با ساختار شعاعی.
- **ویژگی**: نگاشت ویژگی‌ها به تابعی از فاصله از مرکز، بدون توجه به جهت.

ج) تابع چندجمله‌ای

فرمول:

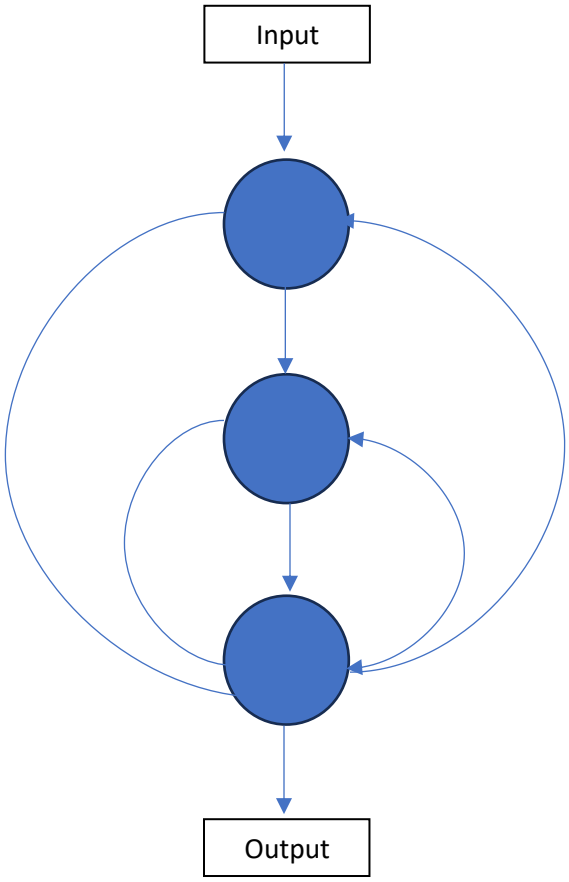
$$f(x) = \sum_{k=0}^{r} a_k x^k$$

این تابع می‌تواند درجات مختلفی از پیچیدگی را مدل‌سازی کند.

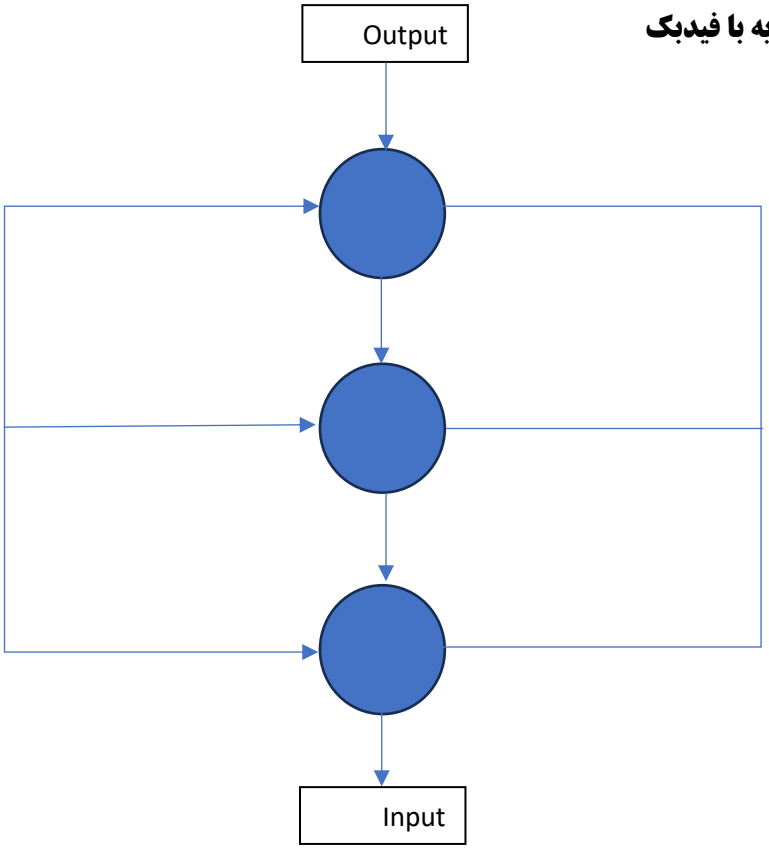
- **کاربرد**: مناسب برای نگاشت‌های بسیار غیرخطی.
- **ویژگی**: با افزایش درجه‌ی r، توانایی مدل در یادگیری رفتارهای پیچیده‌تر بیشتر می‌شود، ولی به طور همزمان ریسک بیش‌برازش (Overfitting) نیز افزایش می‌یابد.

سوال پنجم:

الف) شبکه عصبی تک‌نورونی با فیدبک به خود



ب) شبکه عصبی تک‌لایه با فیدبک



سوال ششم:

الف) پرسپترون تک‌نورونی

ورودی‌هایی که داریم: $P1 = \begin{bmatrix} -1 \\ 1 \end{bmatrix}, P2 = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$

می‌خواهیم پرسپترون طوری طراحی شود که:

- خروجی P1 برابر شود با ۱. (مثلا کلاس ۱)
- خروجی P2 برابر شود با ۰. (مثلا کلاس ۰)

فرض کنیم تابع فعال‌سازی پرسپترون یک step function باشد:

$$f(x) = \begin{Bmatrix} 1 & x \geq 0 \\ 0 & x < 0 \end{Bmatrix}$$

وزن w = [-1, 1] و بایاس b = 0 را انتخاب می‌کنیم.

در این صورت، برای P1: $w\ .\ P1 + b = (-1)(-1) + (1)(1) + 0 => f(2) = 1$

و برای P2 داریم: $w\ .P2 + b = (-1)(1) + (1)(-1) + 0 = \ -2 => f(-2) = 0$

شبکه با این وزن و این بایاس به درستی الگوهای متفاوت را تشخیص می‌دهد و تمایز قائل می‌شود.

ب) شبکه‌ی همینگ

در مرحله‌ی اول، وزن‌ها را می‌سازیم:

$$W = \begin{bmatrix} P1^T \\ P2^£ \end{bmatrix} = \begin{bmatrix} -1 & 1 \\ 1 & -1 \end{bmatrix}$$

برای ورودی جدید x، شبکه‌ی خروجی به شکل زیر محاسبه می‌شود:

$$y = W\ .\ x$$

- اگر x = P1، در نتیجه $y = [2, -2]^T$ که منجر به تشخیص P1 می‌شود.
- اگر x = P2، در نتیجه $y = [-2, 2]^T$ که منجر به تشخیص P2 می‌شود.

این سیستم به صورت رقابتی عمل کرده و نزدیک‌ترین بردار را تشخیص می‌دهد.

ج) شبکه‌ی هاپفیلد

فرض کنیم که دو بردار دو بعدی روبه‌رو را داریم: $P1 = \begin{bmatrix} -1 \\ 1 \end{bmatrix}, P2 = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$

با استفاده از قانون Hebbian وزن‌ها را اینگونه تعریف می‌کنیم:

$$W = P1\ .\ P1^T + P2\ .\ P2^T = \begin{bmatrix} 2 & -2 \\ -2 & 2 \end{bmatrix}$$

وزن‌های قطری صفر می‌شوند: $W = \begin{bmatrix} 0 & -2 \\ -2 & 0 \end{bmatrix}$

اگر شبکه با حتی یک ورودی نویزی مانند $x = [-1, 1]^T$ مقداردهی شود، طی به‌روزرسانی وضعیت نورون‌ها، به سمت یکی از الگوهای P1 یا P2 همگرا می‌شود.

سوال هفتم:

ما در این مسئله ۶ بردار ورودی داریم که به دو کلاس تقسیم شدن:

- کلاس مثبت (t = 1)
  - $P1 = \begin{bmatrix} -1 \\ 1 \end{bmatrix}$
  - $P2 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$
  - $P3 = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$
- کلاس منفی (t = 0)
  - $P4 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$
  - $P5 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$
  - $P6 = \begin{bmatrix} \frac{1}{\varepsilon} \\ \frac{1}{\varepsilon} \end{bmatrix}$

هدف این است که یک پرسپترون تک‌لایه طراحی کنیم که بتواند این دو کلاس را به صورت خطی از هم جدا کند.

الف) طراحی پرسپترون

برای طبقه‌بندی این داده‌ها از یک **پرسپترون تک‌لایه** استفاده می‌کنیم که تصمیم‌گیری رو بر اساس فرمول زیر انجام می‌دهد:

$$y = hardlim(w^T x + b)$$

- $w = [w1, w2]^T$ برابر است با وزن‌ها
- b که برابر است با بایاس
- hadlim(z) = 1 اگر $z \geq 0$، در غیر این صورت مقدارش برابر با صفر است.

این الگوریتم ساده‌ی پرسپترون است:

۱. مقداردهی اولیه:

$$w = [0, 0]^T, b = 0$$

۲. برای هر نمونه (p_i, t_i):

- خروجی را محاسبه می‌کنیم:

$$\hat{t} = handlim(w^T p_i + b)$$

- ارور را محاسبه می‌کنیم:

$$e = t_i - \hat{t}$$

- وزن و بایاس را به‌روزرسانی می‌کنیم:

$$e \cdot \alpha + b = p_i, \quad b \cdot e \cdot \alpha + w = w$$

- این فرآیند رو ادامه می‌دیم تا همه نمونه‌ها به‌درستی طبقه‌بندی بشن. یعنی e برای همه صفر بشه.

ب) بررسی اثر ε بر عملکرد

الگوی $P_6 = \left[\frac{1}{\varepsilon}, \frac{1}{\varepsilon}\right]$ باعث می‌شـود که با افزایش ε، این نقطه به مبدا نزدیک‌تر شـود. وقتی ε بزرگ‌تر می‌شـود (مثلا ۶ یا ۱۲)، این نقطه ممکن است وارد ناحیه‌ی کلاس مثبت شود و طبقه‌بندی صحیح سخت شود.

اگر پرسپترون نتواند با هیچ خطی، همه‌ی داده‌ها را از هم جدا کند -> الگوها خطی جداپذیر نیستند.

در واقع، **تغییر ε** داره یک نوع تست برای "قدرت تعمیم" شبکه انجام می‌دهد. اگر داده‌ها خطی جداپذیر نباشند، پرسپترون نمی‌تواند همه رو درست تفکیک کند. این موضوع نشان می‌دهد که پرسپترون فقط وقتی جواب قطعی و درست می‌دهد که داده‌ها را بتوان با یک خط ساده از هم جدا کرد.

پس در نتیجه، متوجه می‌شویم که پرسپترون تنها زمانی به جواب می‌رسد که الگوها به صورت خطی جداپذیر باشند. همچنین، تغییر ε روی موقعیت P_6 اثر می‌گذارد و ممکن است باعث شود الگوریتم دیگر نتواند با خطی ساده کلاس‌ها را جدا کند.

سوال هشتم:

الف) رسم هر بردار به صورت یک تصویر (ماتریس ۳ * ۳):

هر الگو یک بردار ۹ تایی هست. اون رو باید تبدیل کنیم به ماتریس ۳×۳ (هر سطر ۳ عدد). در اینجا، مقدار ۱ به منزله پیکسل سفید و مقدار ۱- به منزله پیکسل سیاه در نظر گرفته می‌شود:

1. P1:
   1 1 1
   -1 -1 -1
   -1 -1 -1
2. P2:
   -1 -1 -1
   1 1 1
   -1 -1 -1
3. P3:
   -1 -1 -1
   -1 -1 -1
   1 1 1
4. P4:
   1 -1 -1
   1 -1 -1
   1 -1 -1
5. P5:
   -1 1 -1
   1 -1 1
   -1 1 -1
6. P6:
   -1 -1 1
   -1 -1 1
   -1 -1 1

ب) آموزش یک پرسپترون

ب) ما شبکه‌ای می‌خواهیم بسازیم که بتواند تشخیص بدهد کدام الگو با بقیه فرق می‌کند. برای مثال، فرض کنیم:

الگوهای P1، P2، P3 متعلق به کلاس A (خروجی = ۱) هستن.

الگوهای P4، P5، P6 متعلق به کلاس B (خروجی = ۱-) هستن.

ساختار شبکه و پارامترها:

تعداد نورون‌های ورودی: ۹ (چون هر تصویر ۹ پیکسل داره)

وزن‌ها: w_1 تا w_9 ⟵ همه رو صفر مقداردهی اولیه می‌کنیم.

بایاس: $b = 0$ (ابتدا صفر)

الگوریتم آموزش پرسپترون به نحوه‌ی زیر برای هر الگو عمل می‌کند:

محاسبه‌ی خروجی نرون:

$$net = w\,.\,p + b$$

محاسبه‌ی خطا:

$$e = target - output$$

به‌روزرسانی وزن‌ها:

$$w_{new} = w + \alpha\,.\,p\,.\,e$$

به‌روزرسانی بایاس:

$$b_{new} = b + \alpha\,.\,e$$

برای آموزش یک پرسپترون جهت شناسایی الگوی P1 ، مراحل زیر را دنبال می‌کنیم:

تعریف الگوها و برچسب‌ها:

الگوی P1 با برچسب ۱ (مثبت) و سایر الگوها با برچسب ۱− (منفی) مشخص می‌شوند.

مقداردهی اولیه به وزن‌ها و بایاس:

وزن‌ها و بایاس به صورت تصادفی یا صفر مقداردهی می‌شوند.

فرآیند آموزش:

برای هر الگو، خروجی پرسپترون محاسبه می‌شود.

اگر خروجی با برچسب واقعی مطابقت نداشته باشد، وزن‌ها و بایاس به‌روزرسانی می‌شوند.

تکرار مراحل تا همگرایی:

فرآیند آموزش تا زمانی که همه الگوها به درستی طبقه‌بندی شوند، تکرار می‌شود.

با انجام این مراحل، پرسپترون قادر خواهد بود الگوی P1 را از سایر الگوها تمییز دهد.

می‌خواهیم با استفاده از یک نورون پرسپترون، یاد بگیریم که الگوهای P1, P2, P3 رو از الگوهای P4, P5, P6 جدا کنیم.

خروجی مطلوب برای:

P1, P2, P3 = +1

P4, P5, P6 = -1

تنظیمات پرسپترون:

وزن‌ها:

$$w = [w1 + w2 + \cdots + w9] = [0, 0, \dots, 0]$$

بایاس: b = 0

نرخ یادگیری: $\alpha = 1$

تکرار اول (Epoch 1) :

P1 = [1 1 1, −1 −1 −1, −1 −1 −1]

target = +1

$$\boldsymbol{net = 0 \,.\, p + 0 = 0 => output = sign(0) = 0}$$

$$\boldsymbol{e = target - output => 1 - 0 = 1}$$

$$\boldsymbol{w = w + \alpha \,.\, e \,.\, p = 0 + 1 \,.\, 1 \,.\, p = p}$$

$$\boldsymbol{w = [1, 1, 1, -1, -1, -1, -1, -1, -1]}$$

$$\boldsymbol{b = b + \alpha \,.\, e = 0 + 1 \,.\, 1 = 1}$$

تکرار دوم (Epoch 2) :

P1 = [−1 −1 −1, 1 1 1, −1 −1 −1]

target = +1

$$\boldsymbol{net = w \,.\, p + b}$$

$$\boldsymbol{= 0 \xrightarrow{yields} (1)(-1) + (1)(-1) + (1)(-1) + (-1)(1) + (-1)(1) + (-1)(1)}$$
$$\boldsymbol{+ (-1)(-1) + (-1)(-1) + (-1)(-1) + 1 = -3 - 3 + 3 + 1 = -2}$$

$$\boldsymbol{e = target - output => 1 - (-1) = 2}$$

$$\boldsymbol{w = [1, 1, 1, -1, -1, -1, -1, -1, -1] + 2 \,.\, [-1, -1, -1, 1, 1, 1, -1, -1, -1] =}$$

$$\boldsymbol{[-1, -1, -1, 1, 1, 1, -3, -3, -3]}$$

$$\boldsymbol{b = 1 + 2 = 3}$$

این مراحل را برای هر الگو تکرار می‌کنیم تا وزن‌ها و بایاس نهایی را بدست بیاوریم.

سوال نهم:

اگر شبکه عصبی یک لایه‌ی پنهان داشته باشد و از توابع فعال سازی مثل سیگموید یا ReLU استفاده کند، هر نرون در آن لایه می‌تواند **یک خط تصمیم (مرز خطی)** در فضای دوبعدی تعریف کند و بنابراین، **فضا را به دو نیم‌صفحه تقسیم** کند.

در فضای دوبعدی($\mathbb{R}^2$) ، برای اینکه بخواهیم حداکثر **۹** ناحیه‌ی مجزا بسازیم، از رابطه‌ی زیر استفاده می‌کنیم:

$$R(n) = \frac{n^2 + n + 2}{2}$$

ما میخواهیم فضا به ۹ ناحیه‌ی جداگانه تبدیل شود، پس داریم:

$$\frac{n^2 + n + 2}{2} \geq 9 \Rightarrow n^2 + n - 16 \geq 0$$

این معادله درجه دو را حل می‌کنیم:

$$n = \frac{-1 \pm \sqrt{1 + 64}}{2} = \frac{-1 \pm \sqrt{65}}{2} \approx \frac{-1 \pm 8.06}{2} \approx 3.53$$

پس n برابر می‌شود با عدد ۴.

بنابراین، برای تقسیم فضای دوبعدی به ۹ ناحیه‌ی مجزا با استفاده از شبکه عصبی تک‌لایه، **حداقل ۴ نرون** در لایه پنهان لازم است.

منابع:

۱. https://amostofi.com/fa/what-is-a-neuron-in-a-neural-network/
۲. https://analium.com/blog/artificial-neural-network/
۳. https://cs231n.github.io/neural-networks-1/?utm_source=chatgpt.com
۴.