سوال ۱-

آ.
- **دندریت:** دندریت‌ها در یک سلول عصبی، وظیفه انتشار محرک الکتروشیمیایی دریافت شده از دیگر نورون‌ها را به جسم سلولی (سوما) دارند. در واقع دندریت‌ها، معادل بردار ورودی (حاوی مقادیر ضرب شده در وزن‌ها) یک نورون در یک پرسپترون است.
- **سوما:** سوما یا جسم سلولی، به نوعی هسته اصلی یک نورون است و بخش پردازشی خاصی ندارد. از آنجا که ورودی‌های دندریت‌ها به این بخش وارد می‌شوند، می‌توان به نوعی آن را معادل جمع‌کننده (عملگر سیگما) مقادیر ورودی (پس از ضرب در وزن‌ها) در یک نورون یک پرسپترون در نظر گرفت.
- **آکسون:** آکسون در سلول عصبی وظیفه هدایت پالس الکتریکی را دارد و اطلاعات را منتقل می‌کند. در واقع می‌توان آکسون را شبیه به تابع فعال‌سازی و خروجی نورون در یک پرسپترون دانست.
- **سیناپس:** سیناپس، که بخشی از یک نورون نیست، ساختاری است که اجازه انتقال یک سیگنال الکتریکی یا شیمیایی را به نورون بعدی می‌دهد. در واقع در سیناپس، مشابه عمل وزن‌ها در یک شبکه پرسپترون، سیگنال می‌تواند تضعیف یا تقویت شود.

ب.
- **یادگیری با نظارت**[1]: در این نوع یادگیری، هدف یادگیری تابعی است که ورودی را به خروجی نگاشت می‌کند. این یادگیری به وسیله زوج‌های ورودی و خروجی نمونه و دارای برچسب[2] انجام می‌شود. از آنجا که شبکه‌های عصبی برای بازنمایی توابع غیرخطی و پیچیده، بسیار مناسب هستند، نقش بسزایی در یادگیری با نظارت دارند.
- **یادگیری بدون نظارت**[3]: در یادگیری بدون نظارت، داده دارای برچسب وجود ندارد. در نتیجه، الگوریتم یادگیری الگوهای موجود در بین داده‌ها را پیدا می‌کند. نمونه‌های این نوع یادگیری خوشه‌بندی و کاهش بعد هستند. شبکه‌های auto-encoder و استفاده از مقادیر فعال‌سازی لایه‌های مخفی آن‌ها برای خوشه‌بندی و کاهش بعد (استخراج ویژگی‌ها)، در این زمینه پرکاربرد هستند. همچنین معماری‌های شبکه عصبی مانند SOM نیز هستند که می‌توانند به طور مستقیم برای خوشه‌بندی استفاده شوند.

[1] Supervised Learning
[2] Label
[3] Unsupervised Learning

- **یادگیری تقویتی**[۴]: در یادگیری عمیق، هدف انجام اعمالی توسط یک عامل هوشمند در یک محیط است تا پاداش[۵] دریافتی را حداکثر کند. در این مسائل، با بزرگ شدن محیط و اعمال (افزایش تعداد حالات[۶])، امکان ذخیره‌سازی بهترین عمل در هر حالت به شکل‌های ساده مانند جدول وجود ندارد. در نتیجه می‌توان از شبکه‌های عصبی (مانند DQN) برای بازنمایی توابع غیرخطی و پیچیده استفاده کرد.

سوال ۲-

آ. در صورتی که از تابع فعالیت **غیرخطی** استفاده نکنیم، مستقل از تعداد لایه‌های درونی، می‌توان خروجی را برحسب ترکیب خطی ورودی نوشت. در نتیجه افزدون لایه‌های میانی تنها بار محاسباتی ما را افزایش می‌دهد و بر روی کارایی تاثیری ندارد. به همین دلیل، از توابع فعال‌سازی غیرخطی در هر لایه استفاده می‌شود تا بتوان با افزودن تعداد لایه‌ها توابع پیچیده‌تری را بازنمایی کرد.

ب. افزودن بایاس به نوعی امکان شیفت دادن تابع فعالیت را فراهم می‌کند. این کار به فیت شدن بهتر بر روی داده‌ها کمک می‌کند.

پ. به طور کلی افزایش تعداد نورون‌ها در هر لایه و افزایش تعداد لایه باعث پیچیده‌تر شدن شبکه عصبی می‌شود. یکی اثرات واضح این کار، افزایش بار محاسباتی است. در طرف دیگر، پیچیدگی بیشتر شبکه عصبی، امکان بازنمایی توابع پیچیده‌تر توسط آن و کاهش بایاس (که منجر به underfitting می‌شود) را می‌دهد. اما باید توجه کرد، افزایش عمق می‌تواند روند یادگیری را (به خصوص در لایه‌های ابتدایی) بسیار کُند کند. در واقع افزایش تعداد لایه‌ها می‌تواند باعث نزدیک به صفر شدن گرادیان[۷] یا بسیار بزرگ شدن آن[۸] شود که هر دوی آن ها یادگیری را مختل می‌کند.

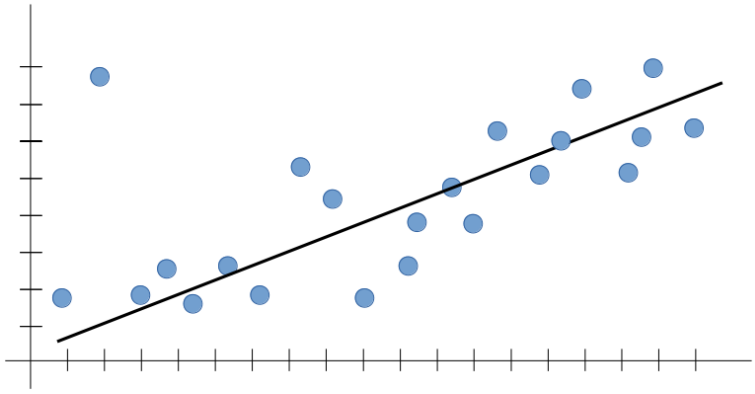
ت. با استفاده از ضریب یادگیری می‌توان اندازه گام را تعیین نمود. معمولا در ابتدای یادگیری می‌توان از اندازه قدم‌های بزرگ‌تری استفاده کرد و با نزدیک شدن به نقطه کمینه، بر اساس یک قاعده مشخص نرخ یادگیری (و در نتیجه اندازه گام) را کاهش داد. به عنوان مثال، در صورتی که با برداشتن چند قدم، مشاهده کنیم که مقدار تابع هزینه کاهش پیدا نکرده است، می‌توان نرخ

[4] Reinforcement Learning
[5] Reward
[6] States
[7] Vanishing gradient
[8] Exploding gradient

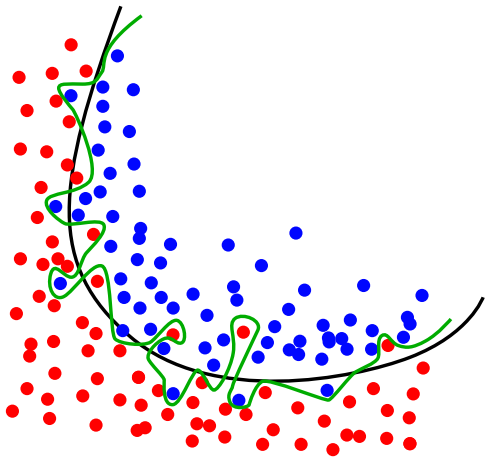
یادگیری را کاهش داد (مثلا در ۰/۹ ضرب کنیم).

سوال ۳-

آ. آندرفیتینگ زمانی رخ می‌دهد که یک مدل نتواند به میزان کافی بر روی ساختار داده‌ها یادگیری داشته باشد. به عنوان مثال در شکل زیر، مدل خطی نمی‌تواند پیش‌بینی مناسبی بر روی داده‌ها داشته باشد.

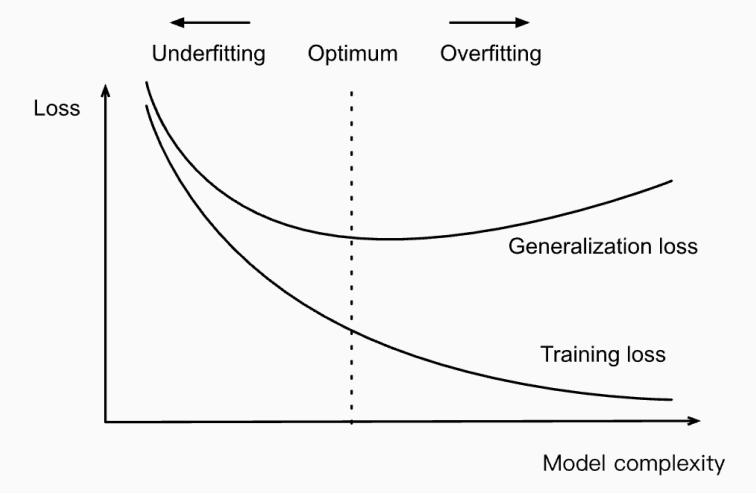


در طرف مقابل، اورفیتینگ زمانی رخ می‌دهد که مدل بسیار به داده‌های خاصی (داده‌های آموزش) وابسته شود و در مواجهه با نمونه‌های جدید نتواند به خوبی آن‌ها را پیش‌بینی کند. در واقع مدل عمومیت نخواهد داشت. در شکل زیر، خط سبز رنگ نشان‌دهنده یک مدل اورفیت شده است که عمومیت مناسبی ندارد. در حالی که خط سیاه رنگ چنین نیست و اگر چه خطای بیشتری روی داده‌های آموزشی دارد، اما در مواجهه با نمونه‌های جدید دقت بالاتری دارد.

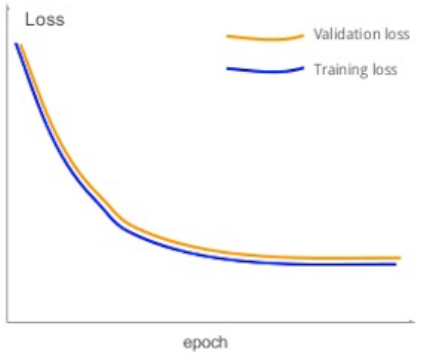


هر دوی این اتفاقات، دقت پیش‌بینی را کاهش می‌دهد.

ب. برای این کار ابتدا داده‌ها را به دو مجموعه train و test تقسیم می‌کنیم. همانطور که در شکل زیر می‌بینیم، در صورتی که میزان loss هر دو مجموعه train و test عدد بالایی بود، دچار آندرفیتینگ هستیم. در طرف مقابل، در صورتی که loss مجموعه train کاهش پیدا کند، اما مجموعه test افزایش پیدا کند (بین این دو فاصله بیفتد) دچار اورفیتینگ هستیم و مدل عمومیت ندارد.



زمانی که هر دو نمودار loss کاهش داشته باشند (شکل زیر)، یعنی این دو مشکل وجود ندارد.



پ. برای رفع مشکل آندرفیتینگ، یکی از راه‌ها آموزش دادن مدل به مدل طولانی‌تر است. اما باید توجه کرد این کار لزوما نمی‌تواند مشکل را رفع کند؛ به عنوان مثال مدل خطی در مورد آ. امکان بازنمایی با دقت بالای داده‌های مسئله را ندارد). در نتیجه می‌توان از مدل‌های پیچیده‌تر (مثلا مدل چندجمله‌ای به جای خطی یا استفاده از شبکه عصبی عمیق‌تر) استفاده کرد.

ت.
- **روش regularization**: در این روش در کنار تابع loss مورد استفاده (مثلا cross-entropy)، قدر مطلق (L1 regularization) یا مجذور (L2 regularization) وزن‌ها را با آن جمع می‌کنیم؛ یعنی:

$$\mathcal{L}_{\mathcal{R}} = \mathcal{L} + \frac{\lambda}{2}\sum_{l=1}^{L}||W^{[l]}||^2$$

که در آن λ پارامتر regularization است و نسبت اهمیت بخش cross-entropy و regularization را مشخص می‌کند. با این کار در واقع شبکه را برای بزرگ شدن وزن‌های آن جریمه می‌کنیم و در نتیجه وزن‌های شبکه مقادیر کوچک‌تری به خود می‌گیرند. از آنجا که زمانی که اورفیتینگ رخ می‌دهد، برخی از وزن‌های شبکه عصبی مقادیر بزرگی به خود می‌گیرند و تاثیر بالایی بر روی خروجی می‌گذارند، انجام این کار از اورفیتینگ جلوگیری می‌کند.

- **روش dropout:** در این روش، با انتخاب یک پارامتر بین صفر و یک، در آموزش شبکه عصبی، برای هر نورون در هر اجرای backprop، یک عدد تصادفی بین صفر و یک انتخاب می‌شود و بسته به کوچک‌تر یا بزرگ‌تر بودن آن از پارامتر انتخابی، تاثیر یا عدم تاثیرگذاری آن نورون در آن مرحله مشخص می‌شود. این کار به نوعی باعث داشتن یک شبکه عصبی ساده‌تر در هر مرحله و همچنین پخش شدن مقادیر وزن‌ها روی نورون‌های مختلف می‌شود. در نتیجه از اورفیتینگ جلوگیری می‌شود.

سوال ۴-

آ. طبق قاعده مشتق زنجیره‌ای داریم (مقدار هر یک از نورون‌ها پیش از اعمال تابع فعال‌سازی با حرف کوچک نشان داده شده است):

$$\frac{\partial \mathcal{L}}{\partial u_0} = \frac{\partial \mathcal{L}}{\partial O_1}\cdot\frac{\partial O_1}{\partial o_1}\cdot\frac{\partial o_1}{\partial B_0}\cdot\frac{\partial B_0}{\partial b_0}\cdot\frac{\partial b_0}{\partial A_1}\cdot\frac{\partial A_1}{\partial a_1}\cdot\frac{\partial a_1}{\partial u_0}$$

مشتق تابع $\mathrm{g}(x) = \sigma(x)$ برابر است با:

$$\frac{\mathrm{d}\sigma(x)}{\mathrm{d}x} = \frac{e^{-x}}{(1+e^{-x})^2} = \frac{1}{1+e^{-x}}\cdot\frac{e^{-x}}{1+e^{-x}} = \sigma(x)\cdot\big(1-\sigma(x)\big)$$

حال هر یک از مشتق‌های جزیی را به طور جداگانه محاسبه می‌کنیم.

$$\frac{\partial \mathcal{L}}{\partial O_1} = 2(O1 - y_t)$$

$$\frac{\partial O_1}{\partial o_1} = \sigma(o_1)\left(1-\sigma(o_1)\right)$$

$$\frac{\partial o_1}{\partial B_0} = t_0$$

$$\frac{\partial B_0}{\partial b_0} = \sigma(b_0)\left(1-\sigma(b_0)\right)$$

$$\frac{\partial b_0}{\partial A_1} = v_1$$

$$\frac{\partial A_1}{\partial a_1} = \sigma(a_1)\left(1-\sigma(a_1)\right)$$

$$\frac{\partial a_1}{\partial u_0} = X_0$$

پس در نهایت با جایگذاری مشتق‌ها، خواهیم داشت:

$$\frac{\partial \mathcal{L}}{\partial u_0} = 2(O1-y_t)\cdot\sigma(o_1)\left(1-\sigma(o_1)\right)\cdot t_0\cdot\sigma(b_0)\left(1-\sigma(b_0)\right)\cdot v_1\cdot\sigma(a_1)\left(1-\sigma(a_1)\right)\cdot X_0$$

ب. با انجام دادن محاسبات به شکل برداری، به ترتیب هر لایه را حساب می‌کنیم (مقدار هر یک از نورون‌ها پیش از اعمال تابع فعال‌سازی با حرف کوچک نشان داده شده است).

$$a = \begin{bmatrix} a_0 \\ a_1 \end{bmatrix} = \begin{bmatrix} b_0 & w_0 & w_1 & w_2 \\ b_1 & u_0 & u_1 & u_2 \end{bmatrix} \begin{bmatrix} 1 \\ X_0 \\ X_1 \\ X_2 \end{bmatrix} = \begin{bmatrix} -0.3 & 0.1 & 0.5 & 0.7 \\ 0.5 & 0.6 & 0.9 & 0.1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0.5 \\ 1.2 \end{bmatrix}$$

$$A = \sigma(a) = \sigma\left(\begin{bmatrix} 0.5 \\ 1.2 \end{bmatrix}\right) = \begin{bmatrix} 0.62 \\ 0.76 \end{bmatrix}$$

$$b = \begin{bmatrix} b_0 \end{bmatrix} = \begin{bmatrix} b_2 & v_0 & v_1 \end{bmatrix} \begin{bmatrix} 1 \\ A_0 \\ A_1 \end{bmatrix} = \begin{bmatrix} -0.4 & 0.2 & 0.6 \end{bmatrix} \begin{bmatrix} 1 \\ 0.62 \\ 0.76 \end{bmatrix} = \begin{bmatrix} 0.18 \end{bmatrix}$$

$$B = \sigma(b) = \sigma\left(\begin{bmatrix} 0.18 \end{bmatrix}\right) = \begin{bmatrix} 0.54 \end{bmatrix}$$

$$o = \begin{bmatrix} o_0 \end{bmatrix} = \begin{bmatrix} b_3 & t_0 \end{bmatrix} \begin{bmatrix} 1 \\ B_0 \end{bmatrix} = \begin{bmatrix} 0.2 & 0.8 \end{bmatrix} \begin{bmatrix} 1 \\ 0.54 \end{bmatrix} = \begin{bmatrix} 0.63 \end{bmatrix}$$

$$O_1 = \sigma\left(0.63\right) = 0.65$$

میزان خطا (تابع cost) برابر است با:

$$\text{cost} = (0.65 - 0)^2 = 0.42$$

سوال ۵-

آ. بر خلاف FCNN که تمام نورون‌ها در یک لایه به تمام نورون‌ها در لایه بعدی متصل هستند، در شبکه‌های پیچشی، تنها بخش کوچک و نزدیکی از هر لایه، به لایه بعد متصل می‌شود (اتصال محلی).

همچنین از آنجا که هر فیلتر (پارامترها) بر روی کل نورون‌های لایه قبل جابجا می‌شود، این فیلتر ویژگی‌ها را از تمام قسمت‌های لایه قبل استخراج می‌کند (به اشتراک‌گذاری پارامترها).

هر دوی این ویژگی‌ها، یعنی اتصال بخش‌های کوچکی از لایه‌های مجاور و به اشتراک‌گذاری ویژگی‌های یاد گرفته شده (مثلا درک خطوط، اشکال و ...)، باعث کارایی مناسب CNN برای داده‌های عکسی می‌شوند.

ب. طبق فرمول زیر، طول و عرض خروجی برابر است با:

$$n' = \left\lfloor \frac{n + 2p - f}{s} \right\rfloor + 1 = \left\lfloor \frac{64 + 2 \times 2 - 5}{3} \right\rfloor + 1 = 22$$

از آنجا که ۶ فیلتر داریم، پس ابعاد خروجی برابر $22 \times 22 \times 6$ است.

پ. هر فیلتر با احتساب بایاس، $5 \times 5 \times 3 + 1 = 76$ پارامتر دارد. پس در مجموع $76 \times 6 = 456$ پارامتر داریم.

سوال ۶-

آ. در صورتی که مدل، کلاس داده ورودی را به درستی و با اختلاف بیش از یک نسبت به بقیه کلاس‌ها پیش‌بینی کند، این تابع مقدار صفر را برمی‌گرداند. از آنجا که در این تابع ورودی‌ها با صفر max شده‌اند، پس کمترین مقدار این تابع **صفر** خواهد بود. بیشینه مقدار این تابع اما مشخص نیست و هر مقدار نامنفی را می‌تواند اختیار کند.

ب. در این صورت عبارت $s_j - s_{yi} + 1$ برای تمام نورون‌های خروجی تقریبا برابر $0 - 0 + 1 = 1$ می‌شود و خروجی max نیز تقریبا برابر یک می‌شود. در نتیجه با توجه به اینکه $N - 1$ (به دلیل جمع نکردن کلاس ground-truth) جمع انجام می‌شود، مقدار $\mathcal{L}_i$ تقریبا برابر $N - 1$ خواهد شد.

پ. در این صورت یک واحد به تمامی $\mathcal{L}_i$ها و همچنین تابع cost اضافه می‌شود. در نتیجه مقدار کمینه تابع به جای صفر، برابر یک خواهد بود و تاثیری بر روی بهینه‌سازی این تابع و یادگیری پارامترها نخواهد داشت.