باسمه تعالی

دانشگاه صنعتی شریف

دانشکده مهندسی برق

۲۵۶۴۵ ـ علوم اعصاب یادگیری، حافظه، شناخت ـ بهار ۹۹ ـ۱۳۹۸

تمرین سری ششم: شبکه‌های عصبی (قسمت عملی)

موعد تحویل: جمعه ۲۰ تیر، ساعت ۲۳:۵۵

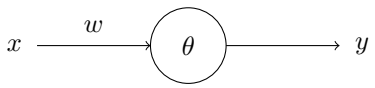
توضیحات

- در انتخاب زبان برنامه‌نویسی برای شبیه‌سازی‌های این تمرین آزاد هستید؛ امّا پیشنهاد می‌شود از یکی از دو محیط برنامه‌نویسی متلب یا پایتون استفاده نمایید.
- لازم است کُدهای مربوط به شبیه‌سازی‌های خود را به صورت کامل تحویل دهید. دقّت کنید که ضروری است این کدها بدون هیچ‌گونه ایراد و خطایی قابل اجرا باشند. بدیهی است در صورتی که کدهای شما با خطا مواجه شوند، نتایجی که به استناد از شبیه‌سازی گزارش کرده‌اید نیز قابل اعتنا نخواهند بود.
- علاوه بر کدها و نتایج شبیه‌سازی، لازم است یک گزارش به فرمت `pdf` نیز تحویل دهید. لازم است تمامی نتایجی از شبیه‌سازی که قصد ارائه‌ی آن را دارید، در فایل گزارش ذکر کنید. این نتایج می‌توانند شامل نمودارها، مقادیر عددی، و... باشند. به جز این موارد، پرسش‌های تئوری که در متن تمرین موجود است را نیز در همان گزارش پاسخ دهید. به علاوه، لازم است توضیح مختصری در مورد نحوه‌ی پیاده‌سازی الگوریتم‌ها و شبیه‌سازی‌ها نیز در گزارش ذکر کنید. در هر مورد دیگری نیز در صورتی که احساس می‌کنید ارائه‌ی توضیحات بیشتر مورد نیاز است، آن را در گزارش بیاورید. تنها موردی که نیازی نیست در گزارش ذکر شود، کدهای شبیه‌سازی است که فایل آن‌ها را جداگانه تحویل می‌دهید.
- تمامی فایل‌های تحویلی را در قالب یک فایل با فرمت `zip` یا `rar` ذخیره کرده و در سامانه‌ی CW آپلود کنید. نام‌گذاری این فایل حتماً به صورت `HW06_StudentNumber.zip/rar` باشد که در آن، به جای `Studen_Number` شماره‌ی دانشجویی خود را می‌نویسید.
- سؤالات خود در مورد تمرین را از طریق آدرس ایمیل `afsharrad.a@gmail.com` مطرح کنید.

در تمرین شماره‌ی ۵، به بررسی چگونگی پیاده‌سازی ساختاری و یادگیری وزن‌های یک شبکه‌ی عصبی مستقیم[1] (از کتاب Computational Intelligence; A Methodological Introduction) پرداختید. در این تمرین، هدف آن است که در چند نمونه‌ی خیلی ساده، آن چه که مطالعه کرده‌اید را پیاده‌سازی کنید.

۱ شبکه‌ی یک‌لایه‌ی تک‌ورودی ـ تک‌خروجی

شکل ۱ را در نظر بگیرید. این شکل ساده‌ترین شبکه‌ی عصبی ممکن است که می‌توان تصوّر کرد. این شبکه یک ورودی دارد و با استفاده از دو پارامتر w و θ خروجی را از روی ورودی تعیین می‌کند.



شکل ۱

در این جا می‌خواهیم به کمک این شبکه، عمل منطقی NOT را پیاده‌سازی کنیم (معادل با مثال قسمت ۶ـ۵ کتاب مرجع که در تمرین بخش‌هایی از آن را مطالعه کردید). برای پیاده‌سازی، نکات زیر را در نظر بگیرید:

- مطابق با مطالب گفته‌شده در کتاب، از تابع فعّال‌سازی زیر برای شبکه‌ی عصبی استفاده کنید:

$$f_{act}(x) = \frac{1}{1+e^{-x}}$$

- برای آن که دقیقاً بتوانید از روابط ریاضی موجود در کتاب مرجع استفاده کنید، می‌توانید پارامتر θ را نیز به صورت یکی از وزن‌ها در نظر بگیرید. در این حالت، باید یک ورودی ثابت با مقدار -1 نیز برای شبکه در نظر بگیرید. به این ترتیب، ورودی تابع فعّالسازی به صورت $wx - \theta$ در می‌آید که همان چیزی است که انتظار دارید.
- می‌توانید پارامتر نرخ یادگیری[2] را $\eta = 1$ در نظر بگیرید. همچنین فعلاً مقادیر اوّلیه‌ی پارامترها را $w_0 = \theta_0 = 3$ در نظر بگیرید.

حال گام‌های زیر را طی کنید و نتایج را در گزارش خود ذکر کنید:

۱. برای آن که شبکه‌ی شما، کار مورد نظر را (که در این جا، عمل NOT است) یاد بگیرد، در ابتدا به یک دیتاست نیاز دارید. برای این منظور، $N = 100$ عدد تصادفی در بازه‌ی $[-2, 2]$ تولید کنید. خروجی تابع NOT را برای هر عدد به صورت زیر تعریف کنید:

$$\mathrm{NOT}(x) = \begin{cases} 0 & x \geq 0 \\ 1 & x < 0 \end{cases}$$

۲. همان طور که به خاطر دارید، پس از آن که هر یک از ۱۰۰ نمونه‌ی موجود در دیتاست را به شبکه‌ی عصبی بدهید، می‌توانید یک بار پارامترهای شبکه را به‌روزرسانی کنید. (تفاوت یادگیری دسته‌ای[3] و برخط[4] را به خاطر دارید؟)
در این جا از یادگیری برخط استفاده می‌کنیم و پس از آن که هر یک از ورودی‌ها را به شبکه می‌دهیم، با استفاده از روابط ریاضی که در تمرین ۵ بررسی کردید، یک بار پارامترهای شبکه را به‌روزرسانی می‌کنیم. بنابراین پس از ۱۰۰ بار به‌روزرسانی پارامترها، شبکه یک بار همه‌ی ورودی‌ها را دیده است. در این مرحله و پس از ۱۰۰ بار به روزرسانی وزن‌ها، خطا را از رابطه‌ی زیر محاسبه کنید:

$$e = \sum_{i=1}^{N} (y_i^* - y_i)^2$$

[1] feed-forward neural network
[2] learning rate
[3] batch learning
[4] online learning

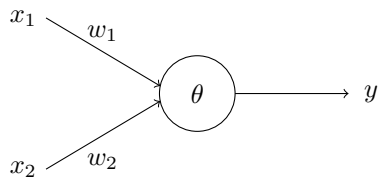
که در آن، y_i خروجی شبکه به ورودی x_i و y_i^*، خروجی مطلوب شما به همان ورودی است.
طبعاً هر قدر مقدار e به صفر نزدیک‌تر باشد، شبکه بهتر کار مورد نظر را یاد گرفته است. مقدار e را محاسبه کنید و در گزارش خود ذکر کنید. همچنین خروجی شبکه‌ی خود را با پارامترهای فعلی، بر حسب x در بازه‌ی $[-2, 2]$ رسم کنید و در گزارش ذکر کنید. آیا شکل به‌دست‌آمده با چیزی که انتظار دارید شباهت دارد؟ در حالت ایده‌آل انتظار دارید این نمودار به چه شکل در بیاید؟

۳. تا به این جا دیتاست خود را یک بار به شبکه عرضه کرده‌ایم و پارامترهای آن را به‌روزرسانی نموده‌ایم. اصولاً در فرآیند یادگیری شبکه‌های عصبی، این کار را بارها و بارها تکرار می‌کنند. به هر بار عرضه‌کردن کل دیتاست به شبکه و به‌روزرسانی وزن‌ها، یک epoch (بخوانید «ایپاک») می‌گویند. حال فرآیند یادگیری را برای ۱۰۰۰ epoch انجام دهید و در پایان هر epoch، مقادیر w، θ، و e را ذخیره کنید. در نهایت، مقادیر این سه پارامتر را بر حسب شماره‌ی epoch در سه نمودار مجزّا رسم کنید. آیا مقدار e با افزایش تعداد epochها به سمت صفر رفته است؟ مقادیر w و θ به سمت چه اعدادی همگرا شده‌اند؟ در این حالت نیز نمودار خروجی شبکه‌ی خود را بر حسب x در حالتی که x در بازه‌ی $[-2, 2]$ است رسم کنید. آیا نمودار به شکل مطلوب نزدیک شده است؟
راهنمایی: در صورتی که کد شما درست باشد، احتمالاً در این مرحله باید به نتایج درست رسیده باشید و خطای شما به صفر میل کرده باشد. اگر پاسخ درست نیست، می‌توانید افزایش تعداد epochها را امتحان کنید ولی اگر در آن صورت نیز شبکه به جواب درست نرسید، باید اشتباه را در نحوه‌ی پیاده‌سازی شبکه و به‌روزرسانی وزن‌ها جست‌وجو کنید.

۴. تا به این‌جا فرض کرده بودید که مقادیر اوّلیه‌ی پارامترهای شبکه به صورت $w_0 = \theta_0 = 3$ بوده‌اند. حال مقادیر اوّلیه را تغییر داده و به صورت $w_0 = \theta_0 = k$ در نظر بگیرید و از $k = 4$ شروع کرده و هر بار k را یک واحد زیاد کنید و قسمت قبل را تکرار کنید (نیاز نیست به ازای همه‌ی مقادیر k نتایج را در گزارش ذکر کنید و در این مرحله کافی است خودتان صرفاً این نتایج را مشاهده کنید). این کار را آن قدر ادامه دهید تا به اوّلین مقداری از k مثل k^* برسید که اگر شرایط اوّلیه را برابر با آن در نظر بگیرید، دیگر با ۱۰۰۰ epoch مقدار خطای شبکه به صفر میل نمی‌کند. در این حالت نمودار خطا و نمودار خروجی شبکه بر حسب ورودی را رسم کنید و در گزارش بیاورید. در ادامه با ثابت نگه داشتن $k = k^*$، تعداد epochها را آن‌قدر زیاد کنید تا دوباره خطا به صفر میل کند. مجدّداً نمودار خطا و نمودار خروجی بر حسب ورودی را رسم کرده و در گزارش ذکر کنید.
نتیجه‌ی این بخش آن است که هر چقدر مقداردهی اوّلیه‌ی پارامترها بدتر باشد، ممکن است نیاز داشته باشید تعداد epochهای بیشتری صبر کنید تا شبکه به حالت مطلوب همگرا شود.

۲ شبکه‌ی یک‌لایه‌ی دوورودی ـ تک‌خروجی

در این قسمت، شبکه‌ی عصبی شکل ۲ را در نظر بگیرید. این شبکه تنها اندکی از شبکه‌ی قبلی پیچیده‌تر است و سه پارامتر دارد.



شکل ۲

۱. یک دیتاست متشکّل از $N = 100$ زوج مرتّب (x_1, x_2) بسازید که هر کدام از x_iها به صورت تصادفی و با توزیع یکنواخت در بازه‌ی $[-2, 2]$ قرار داشته باشند.

۲. فرآیند یادگیری شبکه را به گونه‌ای طی کنید که تابع زیر را یاد بگیرد:

$$f(x_1, x_2) = \begin{cases} 1 & x_1 \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

مقادیر اوّلیه‌ی پارامترها را به صورت $w_1 = w_2 = \theta = 3$ در نظر بگیرد. تعداد epochها را به قدری زیاد کنید که خطا به صفر میل کند. نمودار خطا، و نیز هر سه پارامتر را در چهار نمودار مختلف بر حسب شماره‌ی epoch رسم کنید. (اگر شبکه‌ی شما درست کار کند، در نمودار خطا باید به وضوح مشاهده شود که خطا در epochهای به قدر کافی بزرگ، به صفر میل می‌کند.)
در حالتی که شبکه به درستی تابع مذکور را یاد گرفته است (خطا به صفر میل کرده است)، نمودار خروجی شبکه را بر حسب دو ورودی x_1 و x_2 در یک شکل سه‌بعدی رسم کنید. (محدوده‌ی x_1 و x_2 را در بازه‌ی $[-2, 2]$ در نظر بگیرید. همچنین برای رسم نمودار سه‌بعدی در متلب، می‌توانید از توابع `meshgrid` و `surf` استفاده کنید.) آیا این نمودار با آن چه انتظار دارید انطباق دارد؟

۳. به صورت شهودی و بدون در نظر گرفتن فرآیند یادگیری‌ای که در قسمت قبل طی کردید، (با توجّه به این که خروجی تابع f مستقل از یکی از دو ورودی است) به نظرتان سه پارامتر شبکه را باید چه مقادیری در نظر گرفت تا پاسخ آن درست باشد؟ مقادیر اوّلیه‌ی شبکه را برابر با پارامترهایی که حدس زده‌اید قرار دهید و قسمت قبل را تکرار کنید و نمودار خطا بر حسب تعداد epoch را رسم کنید. آیا تفاوت محسوسی با نمودار قسمت قبل مشاهده می‌کنید؟ آیا نتیجه با چیزی که منطقاً انتظار داشتید تطابق دارد؟

۴. قسمت ۲ را برای تابع زیر تکرار کنید:

$$\text{AND}(x_1, x_2) = \begin{cases} 1 & x_1 \geq 0\,,\ x_2 \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

آیا در این حالت هم شبکه‌ی عصبی می‌تواند خطا را به صفر میل دهد و تابع AND را دقیقاً یاد بگیرد؟ اگر پاسخ منفی است، علّت را تشریح کنید. فکر می‌کنید اگر به جای تابع فعّال‌سازی سیگموید[۵] از تابع فعّال‌سازی دیگری استفاده می‌شد، نتیجه‌ی بهتری حاصل می‌شد؟

[5] sigmoid

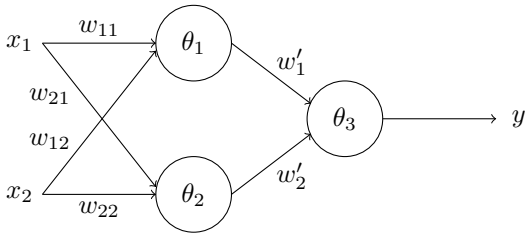
۵. قسمت ۲ را برای تابع زیر تکرار کنید:

$$\mathrm{XOR}(x_1, x_2) = \begin{cases} 1 & x_1 x_2 \leq 0 \\ 0 & \text{otherwise} \end{cases}$$

آیا در این حالت شبکه‌ی عصبی می‌تواند خطا را به صفر میل دهد؟ با توجّه به ماهیت تابع XOR آیا اصلاً امیدی وجود دارد که چنین شبکه‌ای بتواند آن را یاد بگیرد؟

۳ شبکه‌ی دولایه

این بار شبکه‌ی شکل ۳ را در نظر بگیرید.



شکل ۳

۱. یک دیتاست متشکّل از $N = 100$ زوج مرتّب (x_1, x_2) بسازید که هر کدام از x_iها به صورت تصادفی و با توزیع یکنواخت در بازه‌ی $[-2, 2]$ قرار داشته باشند.

۲. فرآیند یادگیری شبکه را به گونه‌ای طی کنید که تابع زیر را یاد بگیرد:

$$f(x_1, x_2) = \begin{cases} 1 & x_1 \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

مقادیر اوّلیه‌ی همه‌ی پارامترها (w_{ij}, w'_i, θ_i) را برابر با عدد ۳ در نظر بگیرید. تعداد epochها را به قدری زیاد کنید که خطا به صفر میل کند. نمودار خطا را بر حسب شماره‌ی epoch رسم کنید. (اگر شبکه‌ی شما درست کار کند، در نمودار خطا باید به وضوح مشاهده شود که خطا در epochهای به قدر کافی بزرگ، به صفر میل می‌کند.) نمودار رسم‌شده را با نمودار قسمت ۲ در بخش قبل (شبکه‌ی عصبی یک‌لایه با دو ورودی) مقایسه کنید. آیا تفاوتی مشاهده می‌شود؟
در حالتی که شبکه به درستی تابع مذکور را یاد گرفته است (خطا به صفر میل کرده است)، نمودار خروجی شبکه را بر حسب دو ورودی x_1 و x_2 در یک شکل سه‌بعدی رسم کنید. (محدوده‌ی x_1 و x_2 را در بازه‌ی $[-2, 2]$ در نظر بگیرید.) آیا این نمودار با آن چه انتظار دارید انطباق دارد؟ آیا این نمودار با شکل متناظر در قسمت ۲ بخش قبل تفاوتی دارد؟

۳. این بار با همان شرایط اوّلیه، سعی کنید تابع زیر را به شبکه یاد دهید:

$$\text{XOR}(x_1, x_2) = \begin{cases} 1 & x_1 x_2 \leq 0 \\ 0 & \text{otherwise} \end{cases}$$

چه تفاوتی میان این شبکه و شبکه‌ای که در بخش قبلی بود، باعث می‌شود تا حدّاقل از نظر تئوری امیدوار باشیم که بتواند تابع XOR را یاد بگیرد؟ آیا در عمل هم این اتّفاق می‌افتد؟

۴. فرض کنید تابع فعّال‌سازی مورد استفاده در شبکه‌ی شکل ۳، به جای سیگموید تابع زیر باشد:

$$f_{act}(x) = \begin{cases} 1 & x \geq 0 \\ 0 & x < 0 \end{cases}$$

در این حالت، به صورت تئوری و بدون انجام شبیه‌سازی، ۹ پارامتر موجود در شکل ۳ را طوری تعیین کنید که این شبکه به درستی تابع XOR را پیاده‌سازی کند. (در این بخش فرض کنید ورودی‌ها صرفاً صفر یا یک هستند.)

۵. پارامترهایی که در قسمت قبل یافتید را به عنوان شرایط اوّلیه در فرآیند یادگیری شبکه‌ی عصبی در نظر بگیرید و به کمک آن‌ها، همان شبکه‌ی قبلی را (با تابع فعّال‌سازی سیگموید) برای یادگیری XOR آموزش دهید و نمودار خطا و

نیز خروجی شبکه بر حسب x_1 و x_2 را رسم کنید. نتایج را با قسمت ۳ مقایسه کنید. آیا پیشرفتی حاصل شده است؟ چرا؟