بنام خدا

تمرین سری سوم یادگیری ماشین(شبکه‌های عصبی)

مریم رضوانی
شماره دانشجویی:۹۹۲۱۱۶۰۰۱۹

(1

الف) **گزینه A؛** تکنیک dropout یک روش منظم سازی است که تقریباً معادل آموزش تعداد زیادی شبکه عصبی با معماری های مختلف به صورت موازی است. در این روش، در هر مرحله آموزش، برخی از خروجی های لایه ها به صورت تصادفی صفر می شوند یا «حذف می شوند». این کار باعث می شود که شبکه به نورون های خاصی وابسته نشود و تمام نورون ها بتوانند بهتر تعمیم دهند.
Bagging: این تکنیک شامل آموزش چندین شبکه عصبی مستقل با داده‌های زیرمجموعه‌ای از داده‌های آموزشی است. سپس، پیشبینی‌های این شبکه‌ها با هم ترکیب می‌شوند تا یک پیش‌بینی نهایی ایجاد کنند. این تکنیک باعث می‌شود که شبکه‌ها از تنوع بیشتری برخوردار شوند و کمتر به نویز داده‌ها حساس باشند
بنابراین، می توان گفت که تکنیک dropout مشابه تکنیک bagging است، با این تفاوت که در dropout، فقط یک شبکه عصبی آموزش داده می‌شود که در هر مرحله، بخشی از نورون های آن حذف می‌شوند. این کار باعث می‌شود که شبکه عصبی مانند یک مجموعه از شبکه‌های عصبی با معماری‌های مختلف رفتار کند. این روش دارای مزایایی مانند سرعت بالا، حافظه کمتر و پیاده سازی آسان است

ب) تفاوت اصلی آنها در این است که batch gradient descent از تمام داده‌های آموزشی برای محاسبه گرادیان در هر مرحله استفاده می‌کند، در حالی که stochastic gradient descent فقط از یک نمونه یا یک زیرمجموعه از داده‌های آموزشی برای محاسبه گرادیان در هر مرحله استفاده می‌کند.

این تفاوت باعث می‌شود که batch gradient descent و stochastic gradient descent دارای مزایا و معایب مختلفی باشند. برخی از این مزایا و معایب عبارتند از:

- batch gradient descent دقیق‌تر و پایدارتر است، زیرا از تمام داده‌های آموزشی برای محاسبه گرادیان استفاده می‌کند و به سمت کمینه محلی یا سراسری تابع هزینه حرکت می‌کند. Stochastic gradient descent ناپایدارتر و نویزی تر است، زیرا ممکن است در هر مرحله به سمت های مختلفی حرکت کند.

- Stochastic gradient descent سریع تر و کم حافظه تر است، زیرا فقط نیاز به محاسبه گرادیان برای یک نمونه از داده های آموزشی در هر مرحله دارد و نیازی به ذخیره سازی تمام داده های آموزشی ندارد. batch gradient descent کندتر و حافظه برتر است، زیرا نیاز به محاسبه گرادیان برای تمام داده های آموزشی در هر مرحله دارد و نیاز به ذخیره سازی تمام داده های آموزشی دارد.

- Stochastic gradient descent مناسب تر برای بهینه سازی توابع هزینه غیر محدب است، زیرا می تواند از کمینه های محلی خارج شود و به سمت کمینه سراسری حرکت کند.batch gradient descent مناسب تر برای بهینه سازی توابع هزینه محدب است، زیرا می تواند به سرعت به کمینه محلی یا سراسری برسد.

ج)مزیت استفاده از چندین لایه در یک شبکه عصبی چندلایه این است که شبکه می‌تواند روابط پیچیده و غیرخطی بین ورودی‌ها و خروجی‌ها را یاد بگیرد و داده‌هایی را متمایز کند که به صورت خطی قابل تفکیک نیستند. برای مثال، شبکه عصبی چندلایه می‌تواند مسئله XOR را حل کند که پرسپترون تک لایه نمی‌تواند.

استفاده از چندین لایه در یک شبکه عصبی چندلایه می‌تواند به شبکه امکان دهد تا از ویژگی‌های سطح بالاتر و انتزاعی‌تری از داده‌ها استفاده کند. برای مثال، در یک شبکه عصبی چندلایه که برای شناسایی تصاویر آموزش داده شده است، لایه‌های اولیه می‌توانند ویژگی‌های ساده مانند لبه‌ها و رنگ‌ها را تشخیص دهند، لایه‌های میانی می‌توانند ویژگی‌های پیچیده‌تر مانند شکل‌ها و الگوها را تشخیص دهند، و لایه‌های آخر می‌توانند ویژگی‌های بسیار انتزاعی مانند چهره‌ها و اشیا را تشخیص دهند.
استفاده از چندین لایه در یک شبکه عصبی چندلایه میتواند به شبکه کمک کند تا از مشکلاتی مانند محو گرادیان و انفجار گرادیان جلوگیری کند.
استفاده از چندین لایه در یک شبکه عصبی چندلایه می‌تواند به شبکه امکان دهد تا از مزایای روش‌های مختلف یادگیری استفاده کند.
اما استفاده از چندین لایه در یک شبکه عصبی چندلایه ممکن است آموزش شبکه را سریع‌تر نکند. بلکه ممکن است آموزش شبکه را کندتر و پیچیده‌تر کند. زیرا با افزایش تعداد لایه‌ها، تعداد پارامترها و محاسبات شبکه نیز افزایش می‌یابد و نیاز به بیشترین داده‌ها و منابع محاسباتی دارد. بنابراین، باید تعادل مناسبی بین تعداد لایه‌ها و کیفیت شبکه ایجاد کرد.

د) تفاوت بین یک Feedforward Network سنتی و یک Recurrent Network در جهت انتقال داده‌ها، حافظه نورون‌ها، پیچیدگی محاسباتی و مناسب بودن برای مسائل مختلف است.
یک Recurrent Network یک نوع شبکه عصبی مصنوعی است که در آن ورودی‌ها به صورت دنباله‌ای از مقادیر متغیر در طول زمان انتقال می‌یابند. (وابستگی زمانی لحاظ می‌شود.) این شبکه از یک لایه ورودی، یک یا چند لایه پنهان و یک لایه خروجی تشکیل شده است. هر لایه شامل چندین نورون است که با وزن‌ها و بایاس‌هایی به لایه بعدی متصل هستند. هر نورون یک تابع فعال‌سازی را روی جمع وزن‌دار ورودی‌های خود و خروجی‌های خود از مرحله قبلی اعمال می‌کند و **خروجی خود را به لایه بعدی و به خودش در مرحله بعدی می‌فرستد.** این شبکه می‌تواند برای یادگیری روابط بین دنباله‌های ورودی و خروجی در مسائل مختلف مانند ترجمه زبان، تبدیل گفتار به متن و کنترل رباتیک استفاده شود.
یک Feedforward Network سنتی یک نوع شبکه عصبی مصنوعی است که در آن داده‌ها یا ورودی‌ها فقط در یک جهت انتقال می‌یابند. این شبکه از یک لایه ورودی، یک یا چند لایه پنهان و یک لایه خروجی تشکیل شده است. هر لایه شامل چندین نورون است که با وزن‌ها و بایاس‌هایی به لایه بعدی متصل هستند. هر نورون یک تابع فعال‌سازی را روی جمع وزن‌دار ورودیهای خود اعمال می‌کند و **خروجی خود را به لایه بعدی می‌فرستد.** این شبکه می‌تواند برای یادگیری روابط بین ورودی‌ها و خروجی‌ها در مسائل مختلف مانند شناسایی الگو، تشخیص گفتار و تشخیص حروف استفاده شود.

ه) نرخ یادگیری مقداری است که نشان می‌دهد که چقدر وزن‌ها و بایاس‌های شبکه در هر تکرار به‌روزرسانی می‌شوند. نرخ یادگیری باید به اندازه کافی بزرگ باشد تا شبکه بتواند به سرعت یاد بگیرد و به نقطه کمینه تابع هزینه برسد. اما اگر نرخ یادگیری بیش از حد بزرگ باشد، ممکن است شبکه ناپایدار شود و از نقطه کمینه عبور کند. برعکس، اگر نرخ یادگیری بیش از حد کوچک باشد، ممکن است شبکه به آهستگی یاد بگیرد و در نقاط کمینه محلی گیر کند. بنابراین، انتخاب نرخ یادگیری مناسب برای بهینه‌سازی شبکه عصبی بسیار مهم است.
ممنتم یک پارامتر است که نشان می‌دهد که چقدر شبکه از گرادیان‌های قبلی برای به‌روزرسانی وزن‌ها و بایاس‌ها استفاده می‌کند. ممنتم باعث می‌شود که شبکه در جهتی که گرادیان‌ها همگرا هستند، سرعت بیشتری داشته باشد و از نوسانات ناخواسته جلوگیری کند. ممنتم می‌تواند به شبکه کمک کند تا از نقاط کمینه محلی خارج شود و به نقطه کمینه سراسری

نزدیک‌تر شود. ممنتم باید به اندازه کافی بزرگ باشد تا شبکه بتواند از گرادیان‌های قبلی استفاده کند و از نقاط کمینه محلی عبور کند. اما اگر ممنتم بیش از حد بزرگ باشد، ممکن است شبکه نتواند در نقطه کمینه متوقف شود و از آن فراتر رود. بنابراین، انتخاب ممنتم مناسب برای بهینه‌سازی شبکه عصبی بسیار مهم است.

و) تفاوت بین multiclass classification و multiple classification در تعداد کلاس‌هایی که هر نمونه می‌تواند به آنها تعلق داشته باشد، است. بعنوان مثال: در multiclass classification، هر نمونه به یک و تنها یک کلاس از چندین کلاس موجود اختصاص می‌یابد. برای مثال، اگر داده‌ها شامل سه کلاس گل‌های آیریس باشند، هر نمونه فقط می‌تواند به یکی از کلاس‌های versicolor ،setosa یا virginica تعلق داشته باشد و در multiple classification، هر نمونه می‌تواند به چندین کلاس از چندین کلاس موجود اختصاص یابد. برای مثال، اگر داده‌ها شامل چندین کلاس برچسب مانند رنگ، شکل، اندازه و جنس باشند، هر نمونه می‌تواند به چندین برچسب متفاوت تعلق داشته باشد.

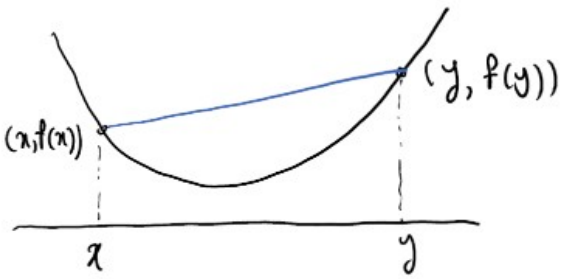
ز)**محو گرادیان:** همانطور که الگوریتم پس انتشار از لایه خروجی به سمت لایه ورودی به سمت پایین (یا به عقب) پیش می رود، گرادیان ها اغلب کوچکتر و کوچکتر می شوند و به صفر نزدیک می شوند که در نهایت وزن لایه های اولیه یا پایین تقریباً بدون تغییر باقی می ماند. در نتیجه، گرادیان کاهشی هرگز به حد مطلوب همگرا نمی شود. این به عنوان مشکل محوگرادیان(*Vanish gradient*) شناخته می شود.
انفجارگرادیان:برعکس، در برخی موارد، با پیشرفت الگوریتم پس انتشار، گرادیان ها بزرگتر و بزرگتر می شوند. این به نوبه خود باعث به‌روز رسانی وزن بسیار بزرگ می شود و باعث واگرایی گرادیان کاهشی می شود. این به عنوان مشکل گرادیان انفجاری(*Exploding gradient*) شناخته می شود.

ی) عبارت درست است؛ زیرا در یک مسأله دسته‌بندی چند کلاسه، لایه خروجی شبکه باید به اندازه کلاس‌های موجود در داده‌ها نورون داشته باشد.هر نورون در لایه خروجی می‌تواند احتمال تعلق یک نمونه به یک کلاس را محاسبه کند. و به ما بستگی دارد که مسأله را چند کلاسه طراحی کنیم برای هر کلاس یک نورون در لایه خروجی قرار می‌دهیم.

2) تعریف تابع محدب:

Definition 1. *A function $f : \mathbb{R}^n \to \mathbb{R}$ is convex if its domain is a convex set and for all x, y in its domain, and all $\lambda \in [0, 1]$, we have*

$$f(\lambda x + (1 - \lambda)y) \leq \lambda f(x) + (1 - \lambda)f(y).$$



مجموعه محدب مجموعه ای است که در آن هر پاره خطی که دو نقطه از مجموعه را به هم متصل می کند نیز به طور کامل در مجموعه قرار می گیرد. این بدان معنی است که مجموعه های محدب ساده هستند و ساختار مشخصی دارند.

یک سطح تصمیم خطی را در نظر بگیرید که با یک ابر صفحه تعریف شده با معادله $w^T x + w0 = 0$، که در آن w بردار وزن و w0 عبارت بایاس است، در نظر بگیرید. فرض کنید x1 و x2 دو نقطه در یک کلاس باشند، و اجازه دهید x3 یک میانگین وزنی x1 و x2 باشد:

$x3 = \lambda x1 + (1 - \lambda)x2$، where $0 \leq \lambda \leq 1$

با جایگزینی x3 در معادله سطح تصمیم، می‌بینیم:

$w^T x3 + w0 = \lambda w^T x1 + (1 - \lambda)w^T x2 + w0 = 0$

از آنجایی که $w^T x1$ و $w^T x2$ هر دو برابر 0 هستند، داریم:

$\lambda w^T x1 + (1 - \lambda)w^T x2 = 0$

این به این معنی است که x3 نیز روی سطح تصمیم قرار دارد، و بنابراین، هر نقطه دیگری در پاره خطی که x1 و x2 را به هم متصل می‌کند نیز باید روی سطح تصمیم قرار گیرد. این ثابت می کند که سطوح تصمیم گیری خطی محدب هستند.

شبکه های عصبی و جداسازی ناحیه محدب نورون ها، واحدهای محاسباتی اساسی شبکه های عصبی، همچنین می‌توانند برای جداسازی مناطق محدب در فضای ورودی استفاده شوند. این به این دلیل است که نورون‌ها می‌توانند انواع توابع غیرخطی را پیاده‌سازی کنند که می‌توانند فضای ورودی را به فضایی با ابعاد بالاتر که در آن سطح تصمیم‌گیری خطی است، ترسیم کنند.

نورون ها می توانند نواحی محدب را با استفاده از سطوح تصمیم خطی یا غیرخطی که فضای ورودی را به مناطق مختلف تقسیم می کنند، جدا کنند. نورون ها می توانند سطوح تصمیم گیری خطی یا غیرخطی را با استفاده از توابع فعال سازی مختلف ایجاد کنند که مجموع وزنی ورودی های آنها را به خروجی تبدیل می کند. به عنوان مثال، یک نورون با یک تابع فعال سازی خطی می تواند یک سطح تصمیم گیری خطی، مانند یک ابر صفحه، که محدب است ایجاد کند. یک نورون با تابع فعال‌سازی سیگموئید می‌تواند یک سطح تصمیم‌گیری غیرخطی، مانند منحنی سیگموئید، که همچنین محدب است، ایجاد کند. با این حال، یک نورون با تابع فعال‌سازی درجه دوم می‌تواند یک سطح تصمیم غیرخطی، مانند سهمی، که محدب نیست ایجاد کند.

برای نشان دادن این موضوع، یک شبکه عصبی ساده با یک نورون و یک تابع فعال‌سازی سیگموئید را در نظر بگیرید. نورون را می توان با معادله نشان داد:

$f(x) = \sigma(wx + b)$، که σ تابع سیگموئید و w و b وزن‌ها و بایاس نورون هستند.

تابع سیگموئید به صورت زیر تعریف می شود:

$\sigma(x) = 1 / (1 + e^{(-x)})$

تابع سیگموئید مقدار ورودی x را به مقداری بین 0 و 1تقسیم می‌کند. این بدان معناست که نورون می‌تواند فضای ورودی را به دو ناحیه جدا کند: یکی که $f(x) < 0.5$ و دیگری $f(x) > 0.5$. اگر بردار وزن w با دقت انتخاب شود، سطح تصمیم تشکیل شده توسط نورون می‌تواند محدب باشد. این را می‌توان با آموزش نورون با مجموعه ای از نقاط داده که نشان دهنده دو کلاسی است که باید از هم جدا شوند به دست آورد. فرآیند آموزش، وزن‌های w نورون را تنظیم می‌کند تا خطا بین برچسب‌های کلاس پیش‌بینی‌شده و برچسب‌های کلاس واقعی نقاط داده به حداقل برسد. به طور خلاصه، سطوح تصمیم گیری محدب هستند و می‌توان از نورون ها برای تقریب آن‌ها استفاده کرد. این باعث می‌شود نورون‌ها برای کارهایی مانند طبقه‌بندی و خوشه‌بندی، که اغلب شامل جداسازی داده‌ها به کلاس‌های مختلف است، مناسب باشند.

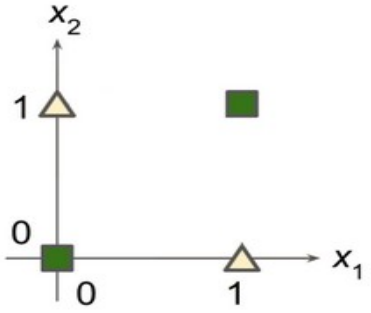
(3

می‌دانیم که مسأله XOR با پرسپترون تک لایه حل نمی‌شود. برای حل آن ابتدا جدول آن را می‌سازیم و نمودار آن را رسم می کنیم:

جدول XOR

Input 1	Input 2	Output
0	0	0
0	1	1
1	0	1
1	1	0

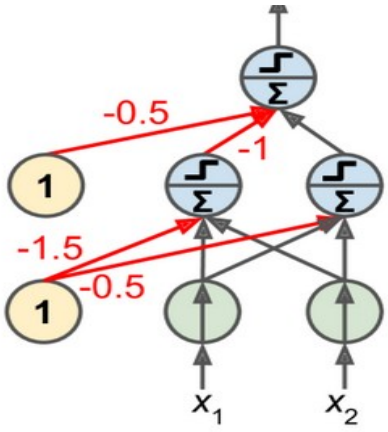
نمودار XOR



طبق نمودار رسم شده می‌بینیم که داده‌ها بصورت خطی قابل تفکیک نیست و برای جداسازی آن‌ها نیاز به(MLP)پرسپترون چندلایه داریم؛ برای حل این مشکل، یک لایه اضافی به پرسپترون خود اضافه می کنیم. ما این لایه اضافی را لایه پنهان می نامیم.
لایه اول ورودی های ما دارای ۲ نورون یعنی X1,X2 هستند که یک نورون اضافی بایاس هم در هر مرحله داریم.
لایه دوم که لایه پنهان نامیده می‌شود شامل ۲ نورون Y1,Y2 است که برای تبدیل مسأله XORبه یک تابع خطی اضافه شده اند.که کار AND و OR که با تابع خطی قابل حل است را انجام می دهند.
پس در این مسأله ما ۳ لایه (ورودی(۳نورون)،پنهان (۳نورون)و خروجی(۱ نورون)) داریم .

$$y_j = f\left(\sum_{i=0}^{m} x_i w_{i,j}\right)$$

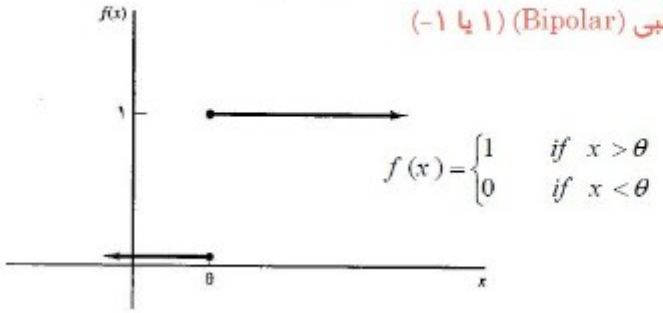
معادله ۱: معادله خروجی پرسپترون



شکل ۴: یک شبکه عصبی mlp که می‌تواند مسئله XOR را حل کند

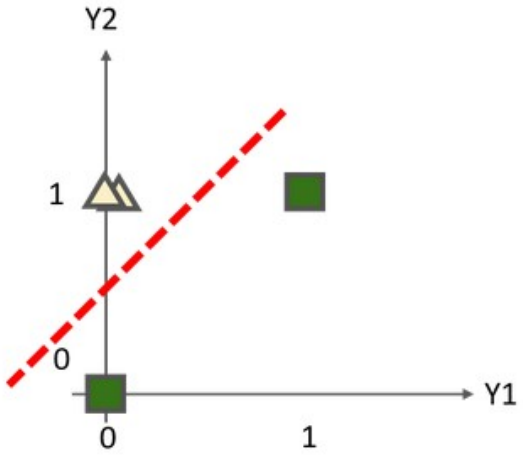
اتصالات خاکستری در شکل وزن w=1 دارند. Y1 یک گیت AND و Y2یک گیت OR است.محاسبه شکل بالا را در زیر می‌توان مشاهده کرد. که **خروجی در هر مرحله توسط تابع Step مشخص می شود.**

- **تابع پله‌ای دودویی (Step Function)**
  - تابع آستانه (Threshold Function) یا تابع هویساید (Heaviside Function)
  - خروجی = سیگنال دودویی (۱ یا ۰) یا دوقطبی (Bipolar) (۱ یا ۱-)



X1	X2	Y1	Y2	Output
0	0	Step(-1.5*1 + 0*1 + 0*1) = 0	Step(-0.5*1 + 0*1 + 0*1) = 0	Step((-0.5*1 - 1*0 + 1*0) = 0
0	1	Step(-1.5*1 + 0*1 + 1*1) = 0	Step(-0.5*1 + 0*1 + 1*1) = 1	Step((-0.5*1 - 1*0 + 1*1) = 1
1	0	Step(-1.5*1 + 1*1 + 0*1) = 0	Step(-0.5*1 + 1*1 + 0*1) = 1	Step((-0.5*1 - 1*0 + 1*0) = 1
1	1	Step(-1.5*1 + 1*1 + 1*1) = 1	Step(-0.5*1 + 1*1 + 1*1) = 1	Step(-0.5*1 -1*1 + 1*1) = 0

یعنی دو نورون Y1و Y2 خروجی ها را به فضای جدیدی بردند، طوریکه تفکیک داده‌ها راحت است.
در شکل زیر تفکیک داده‌ها مطابق خروجی جدید مشاهده می‌شود:



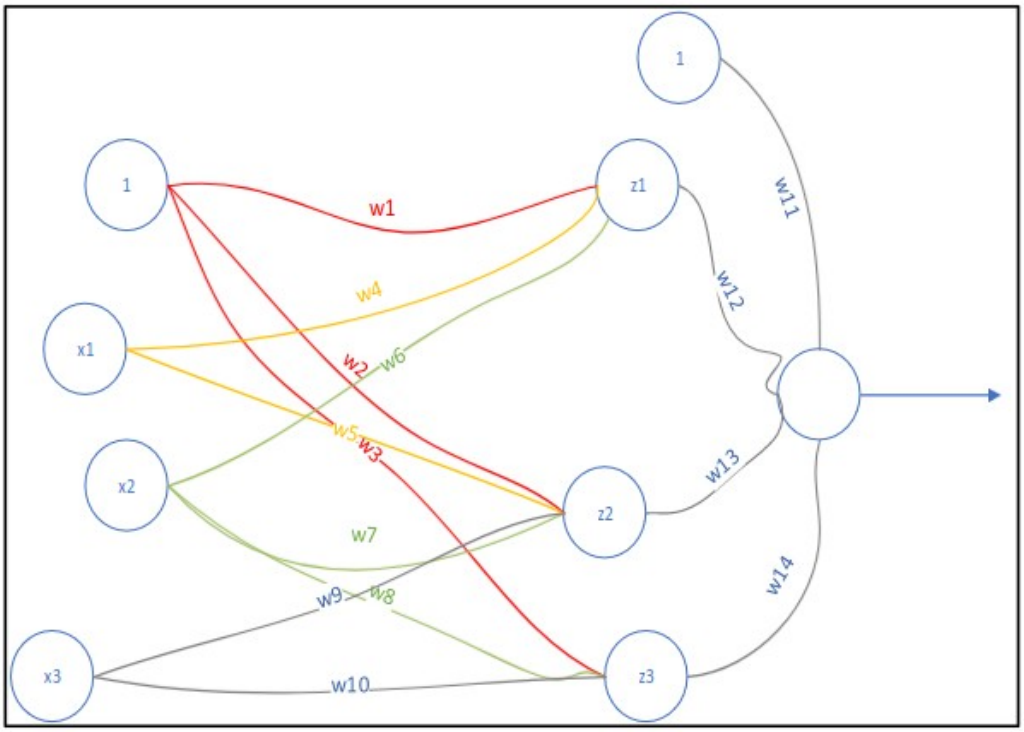
شکل ۵: تغییر ورودی‌ها توسط دو نورون Y1 و Y2

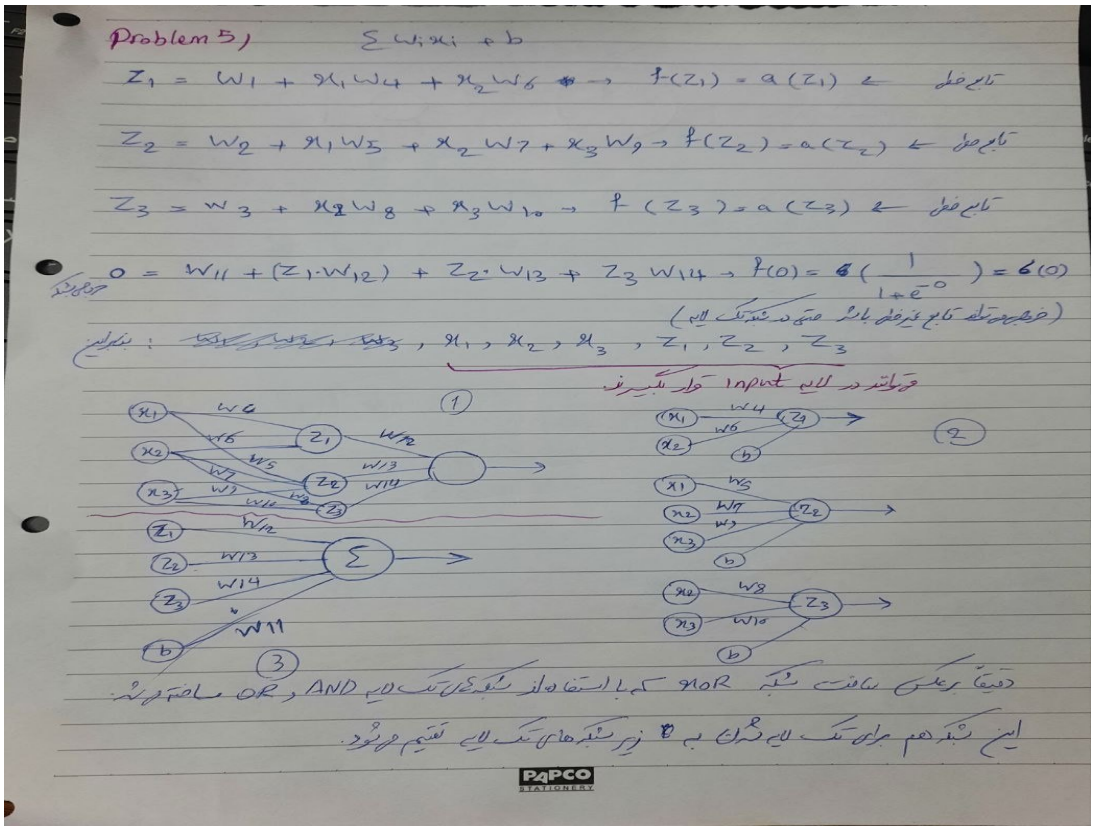
(4

برای پاسخ به این سوال، باید به این نکته توجه کنیم که گرادیان تابع خطا نسبت به وزن‌ها، به صورت جزئی از مشتق‌های خطا نسبت به خروجی‌های هر لایه تشکیل شده است. **اگر همه وزن‌های اولیه یکسان باشند، خروجی‌های هر لایه نیز یکسان خواهند بود. این باعث می‌شود که مشتق‌های خطا نسبت به خروجی‌ها نیز یکسان باشند. در نتیجه،**

گرادیان تابع خطا نسبت به وزن‌ها نیز یکسان خواهند بود. این یعنی همه وزن‌ها با یک مقدار یکسان به‌روزرسانی می‌شوند و هیچ تفاوتی بین آن‌ها ایجاد نمی‌شود. این باعث می‌شود که شبکه عصبی قادر به یادگیری ویژگی‌های مختلف ورودی‌ها نباشد و در نتیجه همگرا نشود.

(5





شبکه عصبی با هر تعداد لایه، بدون توابع غیرخطی، شبیه یک شبکه عصبی تک لایه است.
بله می‌توان این شبکه چندلایه را به یک تک لایه تبدیل نمود. به صورت زیر عمل می‌کنیم:

با توجه به مفروضات گفته شده، میتوان این شبکه را با یک شبکه تک لایه جایگزین کرد. برای این کار، باید از رابطه زیر استفاده کنیم:

$$f(z) = sigmoid(z) = 1/1 + e^{-z}$$

که در آن sigmoid تابع سیگموید و z خروجی هر نورون است. اگر فرض کنیم که ورودی شبکه یک بردار x باشد و وزن های بین لایه ورودی و لایه پنهان را با W_1 و بایاسهای لایه پنهان را با b_1 نشان دهیم، می توان خروجی لایه پنهان را به شکل زیر محاسبه کرد:

W_1= w4,w5,w6,w7,w8,w9,w10
b_1= w1,w2,,w3
z_1=z1,z2,z3

$$z\_1 = f(W\_1\ x + b\_1$$

که در آن z_1 یک بردار سه بعدی است که هر عنصر آن نشان دهنده خروجی یک نورون پنهان است. سپس، اگر وزن های بین لایه پنهان و لایه خروجی را با W_2 و بایاس های لایه خروجی را با b_2 نشان دهیم، می توان خروجی شبکه را به شکل زیر محاسبه کرد:

W_2=w12,w13,w14
b_2=w11

$$y = sigmoid(W\_2\ z\_1 + b\_2)$$

که در آن y یک بردار یک بعدی است که نشان دهنده خروجی شبکه است. حال، اگر بخواهیم این شبکه را با یک شبکه تک لایه جایگزین کنیم، باید از رابطه زیر استفاده کنیم:

$$y = sigmoid(W\ x + b)$$

که در آن W و b وزن ها و بایاس های شبکه تک لایه هستند. برای محاسبه این وزن ها و بایاس ها، باید از رابطه زیر استفاده کنیم:

$$W = W\_2\ f(W\_1)$$
$$b = W\_2\ f(b\_1) + b\_2$$

که در آن f(W_1) و f(b_1) بردارهایی هستند که هر عنصر آن ها برابر با مقدار تابع سیگموید روی هر عنصر ماتریس W_1 و بردار b_1 است. برای مثال، اگر فرض کنیم که وزن ها و بایاس های شبکه با یک لایه پنهان به شکل زیر باشند:

W_1 = [[0.1, -0.2, 0.3], [0.4, -0.5, 0.6], [0.7, -0.8, 0.9
b_1 = [0.1, 0.2, 0.3
,W_2 = [[0.1, 0.2, 0.3]
b_2 = [0.1

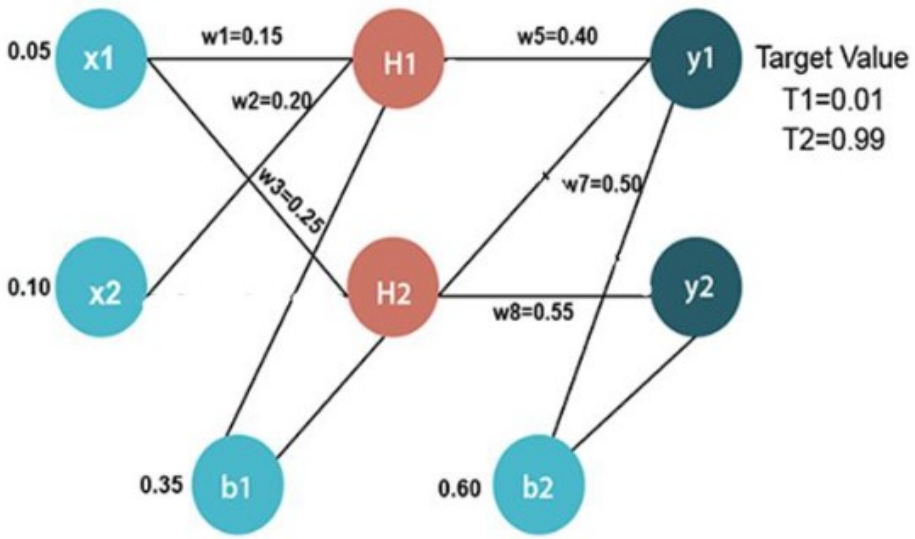
می توان وزن ها و بایاس های شبکه تک لایه را به شکل زیر محاسبه کرد:

f(W_1) = [[0.524, 0.450, 0.574], [0.598, 0.378, 0.645], [0.668, 0.310, 0.710
f(b_1) = [0.524, 0.549, 0.574
,W = W_2 f(W_1) = [[0.186, 0.133, 0.201]
b = W_2 f(b_1) + b_2 = [0.214

بنابراین، معماری پیشنهادی برای شبکه تک لایه به شکل زیر است:

y = sigmoid([[0.186, 0.133, 0.201] x + [0.214])

(6



$x_1 = 0.05$ $w_1 = 0.15$ $w_5 = 0.40$ (6

$x_2 = 0.10$ $w_2 = 0.20$ $w_7 = 0.50$ Target Value

$b_1 = 0.35$ $w_3 = 0.25$ $w_8 = 0.55$ $T_1 = 0.01$

$b_2 = 0.60$ $T_2 = 0.99$

Problem 6) activation function is Sigmoid $= \frac{1}{1+e^{x}}$

$H_1 = x_1 w_1 + x_2 w_2 + b_1$

out $H_1 = \frac{1}{1+e^{-H_1}}$

Forward pass

(a1) $\Leftarrow H_1 = x_1 w_1 + x_2 w_2 + b_1$

$= (0.05 \times 0.15) + (0.1 \times 0.2) + 0.35 = 1377 \rightarrow \sigma(H_1) = 0.59 = a_1$

(a2) $\Leftarrow H_2 = x_1 w_3 + x_2 w_4 + b_1 \rightarrow$

$= (0.5 \times 0.25) + (0.1 \times 0.3) + 0.35 = 0.362 \rightarrow \sigma(H_2) = 0.58 = a_2$

$y_1 = a_1 w_5 + a_2 w_7 + b_2$

$= (0.59 \times 0.4) + (0.58 \times 0.5) + 0.6 = 1.12 \rightarrow \sigma(y_1) = 0.75 = (z_1)$

$y_2 = (a_1 \times w_6) + (a_2 w_8) + b_2 =$

$+ (0.58 \times 0.55) + 0.6 = 0.919 \rightarrow \sigma(y_2) = 0.71 = (z_2)$

Loss: خود تابع

$E_{total} = \sum \frac{1}{2}(t - o)^2 = \frac{1}{2}(T_1 - z_1)^2 + \frac{1}{2}(T_2 - z_2)^2$

(t: target, o: output)

$E_{total} = \frac{1}{2}(0.1 - 0.75)^2 + \frac{1}{2}(0.99 - 0.71)^2 =$ ~~0.273 + 0.392~~ $= 0.312$

$E_{total} = \frac{1}{2}(0.547) + \frac{1}{2}(0.078) = 0.273 + 0.0392 = 0.312$

۱۳
مهر
سه شنبه
5 October 2021
۲۸ صفر ۱۴۴۳

محاسبه خطای بین انتظار:

$$E = \frac{1}{2}(t_i - o_i)^2$$

$$E_1 = \frac{1}{2}(t_1 - y_1)^2 = \frac{1}{2}(0.1 - 0.775)^2 = \frac{0.547}{2} = 0.273$$

$$E_2 = \frac{1}{2}(t_2 - y_2)^2 = \frac{1}{2}(0.99 - 0.771)^2 = \frac{0.078}{2} = 0.0392$$

$$\frac{\partial E_1}{\partial w_{ji}} = \frac{\partial E}{\partial o_j} \times \frac{\partial o_j}{\partial net_j} \times \frac{\partial net_j}{\partial w_{ji}}$$

$$-(t_i - o_j) \quad o_j(1-o_j) \quad x_{ji}$$

Back Propagation:

Consider w_5

Error at $w_5 = \frac{\partial E_{total}}{\partial w_5}$

$$\frac{\partial E_{total}}{\partial w_5} = \frac{\partial E_{total}}{\partial z_1} \times \frac{\partial z_1}{\partial y_1} \times \frac{\partial y_1}{\partial w_5}$$

$$-(T_1 - o_1) \quad o_1(1-o_1) \quad x_{ji}$$

$$\frac{\partial E_{total}}{\partial z_1} = -(T_1 - o_1) = -(T_1 - z_1)$$

خروجی z_1

$$\rightarrow -(T_1 - z_1) \cdot z_1(1 - z_1) \cdot a_1$$

① $= -0.174 \times 0.118 \times 0.159 = -0.008 \rightarrow \frac{\partial E_{total}}{\partial w_5} = 0.082$ میزان تغییر وزن w_5

updating w_5: $w_5 = w_5 - \eta \times \frac{\partial E_{total}}{\partial w_5} = 0.14 - 0.1(0.82) = 0.391 \simeq 0.14$

in the same way w_7, w_8

$$\frac{\partial E_{total}}{\partial w_7} = \frac{\partial E_{total}}{\partial z_1} \times \frac{\partial z_1}{\partial y_1} \times \frac{\partial y_1}{\partial w_7} = -0.174 \times 0.118 \times 0.158 = -0.0077$$

$$w_7 = w_7 - \eta \frac{\partial E_{total}}{\partial w_7} = 0.15 - 0.1(-0.77) = 0.149 = 0.15$$

$$w_8 = \frac{\partial E_{total}}{\partial w_8} = \frac{\partial E}{\partial z_2} \times \frac{\partial z_2}{\partial y_2} \times \frac{\partial y_2}{\partial w_8} =$$

$$= -(T_2 - z_2) \cdot z_2(1 - z_2) \cdot a_2 = 0.128 \cdot 0.12 \cdot 0.58 = 0.32$$

$$w_8 = w_8 - \eta \frac{\partial E_{total}}{\partial w_8} = 0.55 - 0.1(0.032) = 0.54$$

Now at Hidden layer updating w_1, w_2, w_3

$$\frac{\partial E_{total}}{\partial w_1} = \frac{\partial E_{total}}{\partial a_1} \times \frac{\partial a_1}{\partial H_1} \times \frac{\partial H_1}{\partial w_1} \quad w_1 = 0.5$$

$$\frac{\partial E_{total}}{\partial a_1} = \frac{\partial E_1}{\partial a_1} + \frac{\partial E_2}{\partial a_2}$$

$$\frac{\partial E_1}{\partial a_1} = \frac{\partial E_1}{\partial y_1} \times \frac{\partial y_1}{\partial a_1} \rightarrow w_5 = 1.4$$

$$\frac{\partial E_1}{\partial a_1} = 1.13 \times 1.4 = 1.55$$

$$\frac{\partial E_1}{\partial y_1} = \frac{\partial E_1}{\partial z_1} \times \frac{\partial z_1}{\partial y_1} = 1.74 \times 1.18 = 1.13$$

$$\frac{\partial E_2}{\partial a_1} = \frac{\partial E_2}{\partial y_2} \times \frac{\partial y_2}{\partial a_2} = -0.19$$

$$\frac{\partial E_{total}}{\partial a_1} = 0.55 + (-0.19) = 0.36$$

$$* \frac{\partial a_1}{\partial H_1} = a_1(1-a_1) = 1.59(1-1.59) = 0.24$$

$$H_1 = w_1 x_1 + w_2 x_2 + b_1$$

$$\frac{\partial H_1}{\partial w_1} = x_1 = 0.5$$

$$\frac{\partial E_{total}}{\partial w_1} = 0.36 \times 0.24 \times 0.5 = 0.00043$$

updating $w_1 = w_1 - \eta \frac{\partial E_{total}}{\partial w_1} = 0.15 - 0.1(0.004) = 0.14$

$$w_2 = w_2 - \eta \frac{\partial E_{total}}{\partial w_2} = 0.2 - 0.1(0.008) = 0.199$$

$0.36 \times 0.24 \times 0.1 = 0.008$

$$\frac{\partial E_{total}}{\partial w_3} = \frac{\partial E_{total}}{\partial a_2} \times \frac{\partial a_2}{\partial H_2} \times \frac{\partial H_2}{\partial w_3} \quad w_3 = .125$$

$$\frac{\partial E_{total}}{\partial a_2} = \frac{\partial E_1}{\partial a_2} + \frac{\partial E_2}{\partial a_2} \quad \ldots$$

$$\frac{\partial E_1}{\partial a_2} = \frac{\partial E_1}{\partial y_2} \times \frac{\partial y_2}{\partial a_2} =$$

$$\frac{\partial E_2}{\partial a_2} = \frac{\partial E_2}{\partial y_2} \times \frac{\partial y_2}{\partial a_2} =$$

$$\frac{\partial a_2}{\partial H_2} = a_2(1-a_2) = .158(1-.158) = .12$$

$$\frac{\partial E_{total}}{\partial w_3} = .141 \times .12 \times .125 = .01$$

updating $w_3 = w_3 - \eta\left(\frac{\partial E_{total}}{\partial w_3}\right) = .125 - .1(.1) = .1249$