به نام خدا

دانشگاه صنعتی شریف

دانشکده مهندسی برق

گزارش پروژه‌ی کارشناسی

بررسی و آموزش شبکه‌های عصبی اسپایکی احتمالاتی

نگارش:

علی فتحی

استاد راهنما:

دکتر صابر صالح

بهار ۹۸

فهرست

راهنمای اشکال

قدردانی

برای این پروژه، از راهنمایی‌های استاد راهنمای خود، دکتر صالح، کمال قدردانی را دارم؛ که موضوع مورد بحث را برای کار پیشنهاد داده و مراجع و منابع مناسبی در این راستا در اختیار من قرار دادند.

همچنین از اساتید درس پروژه‌ی کارشناسی خود، دکتر فردمنش در درس پروژه ۱ و دکتر بهروزی در درس پروژه‌ی ۲، برای راهنمایی‌هایشان در زمینه‌ی ارائه و نگارش مناسب درمورد پروژه، کمال تشکر را دارم.

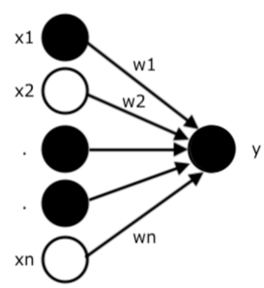
۰ چکیده

در این مقاله، به بررسی یک نوع خاص از شبکه‌های عمیق، شبکه‌های باینری احتمالاتی یا PNNها می‌پردازیم. دو ویژگی باینری بودن و احتمالاتی بودن، ویژگی‌هایی هستند که در شبکه‌های اسپایکی (در واقع دقیق‌تر، شبکه‌های مبتنی بر اسپایک‌ها) توجیه بیولوژیکی دارند و بر خلاف شبکه‌های عصبی عمیق امروزی، از این دو ویژگی صرفنظر نشده است. ابتدا بر اساس منبع [۱]، ساختار شبکه و مدل نورون‌ها را در این روش معرفی می‌کنیم، سپس روابط حاکم بر شبکه را بدست می‌آوریم. در این قسمت روابط مفهومی برای آموزش شبکه با کمک ایده‌ی پس انتشار خطا[1] را نیز معرفی می‌کنیم که بستر را برای استفاده از الگوریتم نزول گرادیان[2] مهیا می‌کند، که روش اصلی آموزش ضرایب شبکه‌های امروزی است. در ادامه، به تحلیل ریاضی شبکه می‌پردازیم و نشان می‌دهیم می‌توان با تعریف پارامترهای مناسب، روابط حاکم بر شبکه را با روابطی خطی توصیف کرد. سپس روش آموزش مطرح شده را در قبل را در این قسمت دقیق‌تر می‌کنیم. در پایان نیز تعابیر مختلفی از روابط بدست آمده از شبکه ارائه می‌دهیم.

واژه‌های کلیدی: شبکه‌های عصبی عمیق، نورون احتمالاتی، شبکه‌های خطی، پس انتشار خطا، نزول گرادیان، قاعده احتمال بیز

۱ مقدمه (مرور ادبیات)

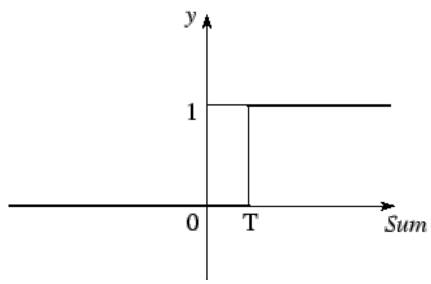
قدمت شبکه‌های عصبی باینری[3] به میانه‌ی قرن بیستم میلادی برمی‌گردد، که با معرفی نورون باینری (ترشولدی) توسط مک‌کلوچ[4] و پیتز[5] در سال ۱۳۴۳ آغاز شده و با کارهای هب در سال ۱۹۴۹ و پرسپترون[6] رزنبلات[7] در ۱۹۶۲ ادامه می‌یابد. در تمام این کارها، مدل نورون همان مدل باینری ترشولد[8]ی است (شکل ۱.۱) :



شکل ۱.۱ – نورون باینری

[1] Error Backpropagation
[2] Gradient Descent
[3] Binary Neural Networks
[4] Warren McCulloch
[5] Walter Pitts
[6] Perceptron
[7] Frank Rosenblatt
[8] Threshold

مطابق شواهد بیولوژیکی، اسپایک زدن هر نورون را اسپایک زدن نورون‌هایی تعیین می‌کند که به آن متصل است. رفتار آن به صورت زیر (شکل ۱.۲) است:

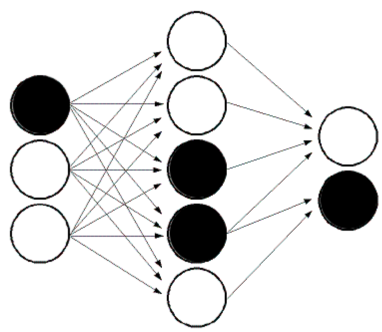


$$sum = \sum w_i x_i \qquad (1.1)$$

شکل ۱.۲ – رفتار ترشولدی نورون

که با صرفنظر از آثار پسماندی نورون، پتانسیل نورون بصورت تابعیت خطی وزن‌دار از نورون‌های متصلش بدست می‌آید (رابطه ۱.۱) و اگر این جمع از ترشولد بیشتر باشد، به اسپایک زدن این نورون نیز می‌انجامد، که در اینجا به معنی ۱ شدن آن است (همانطور که مشخص است، رفتار ترشولدی یک رفتار غیرخطی است).

در بیشتر این کارها، ساختار شبکه مستقیم و بدون فیدبک فرض می‌شود و ما هم در این مقاله، تنها روی ساختارهای این چنینی تمرکز می‌کنیم.



شکل ۱.۳ – ساختار شبکه‌ی باینری

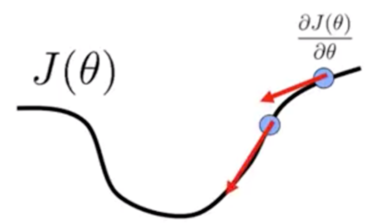
شبکه، مجموعه‌ای از نورون ها است که طبق ساختار مشخصی قرار گرفته‌اند و با ضرایبی به سایر نورون‌ها متصل اند. معمولا نورون‌ها را به صورت لایه‌ای قرار می‌دهیم؛ یعنی یک شبکه متشکل از تعدادی لایه است که در هر لایه، تعدادی نورون قرار دارد که تنها از نورون‌های لایه‌ی قبل از خود تاثیر می‌پذیرند، یا به عبارتی دیگر تنها با ضرایبی به نورون‌های لایه‌ی قبل از خود متصلند (شکل ۱.۳).

این شبکه‌های باینری، دشواری‌های زیادی برای آموزش ضرایب به منظور انجام یک عملکرد مشخص دارند. در مسئله‌ی دسته‌بندی[9]، موفقیت شبکه‌های عصبی با ورود روش‌هاش بهینه سازی حاصل شد که البته این موفقیت مدیون پیشرفت پردازنده‌ها و کامپیوترها

---

[9] Classification

است. مبنای این روش، ایجاد تغییراتی در شبکه است به صورتی که روابط آن مشتق پذیر شده و بتوان از الگوریتم نزول گرادیان[۱۰] برای آموزش ضرایب استفاده کرد. این روش، این گونه است که یک تابع هزینه، مبتنی بر خروجی مطلوب، برای شبکه تعریف می‌کنیم به شکلی که وقتی کمینه شود که ضرایب به صورت بهینه تنظیم شده باشند. سپس مرحله به مرحله، با شروع از ضرایب اولیه حرکت در خلاف راستای گرادیان تابع هزینه در فضای ضرایب حرکت می‌کنیم تا به نقطه کمینه‌ی آن برسیم (شکل ۱.۴):



شکل ۱.۴ – الگوریتم نزول گرادیان

که لازمه‌ی استفاده‌پذیر بودن این روش، دانستن مشتق تابع هزینه نسبت به همه‌ی ضرایب شبکه است که در شبکه‌های عصبی عمیق امروزی[۱۱]، این کار با پس انتشار خطا[۱۲] ممکن می‌شود. مشتق‌پذیر بودن این معادلات هم بخاطر استفاده از توابع فعال‌ساز پیوسته (مانند سیگموید[۱۳]) به جای تابع فعال‌ساز ترشولدی است.

گروه دیگری از شبکه‌های عصبی، شبکه‌های عصبی احتمالاتی[۱۴] هستند که در این مقاله هم تمرکز ما بر روی گروه خاصی از این شبکه‌ها است که در مقاله‌ی [1] معرفی شده است (انواع معمول PNN ها، شبکه بولتزمن و شبکه‌های Generative هستند که در اینجا به آنها کاری نداریم). لازم به ذکر است رفتار احتمالاتی نورون‌های مغز انسان، توجیه بیولوژیکی دارد و شواهد کافی هم نداریم که یک نورون با ورودی‌های یکسان رفتار مشابهی در شرایط مختلف از خود نشان دهد. به عنوان یک استدلال نادقیق، می‌توان از این واقعیت استفاده کرد که انسان در مواجه با شرایط یکسانی هر بار به چیزهای مختلفی فکر می‌کند و عکس‌العمل‌های متفاوتی نشان می‌دهد (مثلا یک مداد، می‌تواند ذهن انسان را به سمت‌های مختلفی ببرد که قابل پیش‌بینی نیستند).

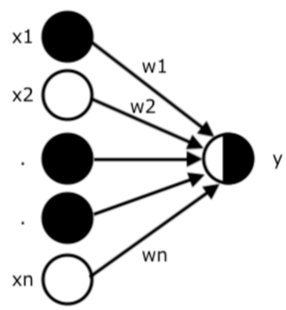
در ابتدا به بیان مدل و ساختار شبکه‌ی مدنظر می‌پردازیم (بخش ۲). سپس برای آن روابط تحلیلی ریاضی بدست آورده و نیز روشی برای آموزش آن بر پایه‌ی روش بهینه‌سازی Gradient Descent پیشنهاد می‌دهیم (بخش ۳ و ۴).

10 Gradient Descent
11 Deep Neural Networks (DNNs)
12 Error Backpropagation
13 Sigmoid
14 Probabilistic Neural Networks (PNNs)

۲ معرفی مدل شبکه‌ی احتمالاتی ([۱])

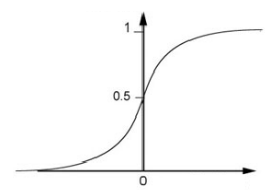
۲.۱ معرفی نورون احتمالاتی

یک نورون احتمالاتی، رفتاری به صورت زیر (شکل ۲.۱) دارد:



$$pot_y = \sum w_i x_i \qquad (2.1)$$

$$p_y = \sigma(pot_y) \qquad (2.2)$$



شکل ۲.۱ – نورون احتمالاتی و رفتار آن

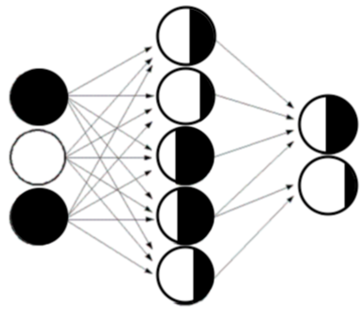
رفتار آن بدین صورت است که پتانسیل یک نورون، مشابه شبکه‌ی باینری ساده، با ترکیب خطی فعالیت‌های نورون‌های متصل به آن از لایه‌ی قبل ضربدر ضریبشان مشخص شده (رابطه ۲.۱) و بسته به این پتانسیل، یک احتمال برای اسپایک زدن این نورون به شکل سیگموید بدست می‌آید (رابطه‌ی ۲.۲). شکل ۲.۱ سمت راست، تابع فعالیت به شکل سیگموید را نشان می‌دهد و احتمال اسپایک زدن نورون را نشان می‌دهد. هر چه پتانسیل نورون بیشتر باشد، احتمال اسپایک زدن به ۱ نزدیک‌تر است و هرچه پتانسیل منفی‌تر باشد (یعنی ضرایب منفی بوده و بازدارنده عمل کرده‌اند) این احتمال به صفر نزدیک‌تر است.
لازم به ذکر است در این مدل، فرض می‌کنیم شبکه، رفتار ترتیبی و سنکرون دارد؛ یعنی ابتدا نورون‌های لایه‌ی قبل در یک لحظه‌ی زمانی اسپایک می‌زنند (یا نمی‌زنند) و سپس در لحظه‌ی بعدی، فعالیت نورون‌های لایه‌ی بعدی با پتانسیل القا شده در آنها مشخص می‌شود.

وجود تابع سیگموید در اینجا به دو دلیل دارای اهمیت است. اول آنکه مشابه DNN ها، که رفتار مشتق‌پذیر خود را مدیون این توابع هستند، در اینجا نیز این تابع مشتق‌پذیر دقیقاً برای عمل فعال‌سازی ظاهر شده است، و گویا شبیه همان ایده‌ی نرم‌سازی تابع ترشولدی اینجا اتفاق افتاده است. زیرا در حالت حدی که احتمال اسپایک زدن از تابع ترشولدی پیروی کند، پتانسیل نورون اگر از ترشولد کمتر باشد، با احتمال صفر اسپایک می‌زند که یعنی به صورت یقینی اسپایک نمی‌زند و اگر از ترشولد بیشتر باشد، با احتمال ۱ اسپایک زده که به معنی یقینی ۱ بودن خروجی آن است و شبکه به شبکه‌ی باینری ساده تبدیل می‌شود؛ و ما رفتار نرم برای نورون قائل شده‌ایم بدون آنکه لازم باشد تعبیر پیوسته بودن از خروجی آن داشته باشیم که بر خلاف واقعیت بیولوژیکی اسپایک است.
قابل ذکر است که خود تابع سیگموید هم ذات احتمالاتی دارد، چرا که در مدل گاز کامل بولتزمان[15]، احتمال حالت‌های ذرات گاز با این تابع مشخص می‌شود پس تابع سیگموید در سیستم‌های بس‌ذره‌ای توجیه فیزیکی نیز دارد.

[15] Boltzmann Ideal Gas Model

۲.۲ ساختار شبکه

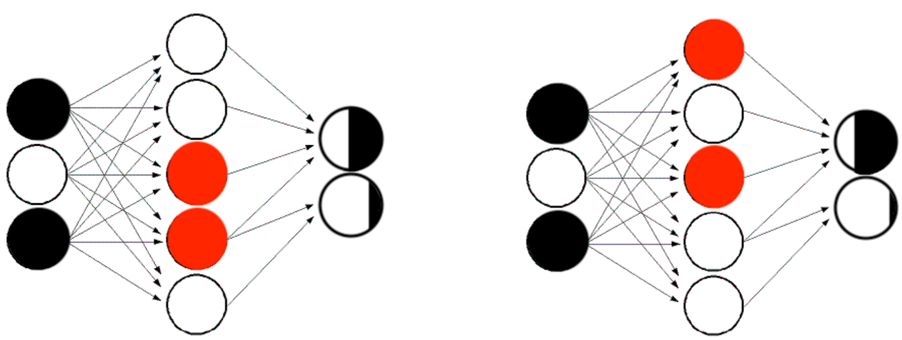
شکل کلی شبکه به صورت زیر (شکل ۲.۲) است:



شکل ۲.۲ – ساختار شبکه

که از لحاظ ساختار مشابه شبکه‌های عصبی عادی یا شبکه‌های باینری است. یک ورودی یقینی به شبکه داده می‌شود (تصویر، مختصات یا هرچیز دیگری در نقش تحریک‌های مغزی). این ورودی، پتانسیل‌هایی را در نورون‌های لایه‌ی بعد بوجود می‌آورد که احتمال اسپایک زدن آنها را مشخص می‌کند (و به طور نمادین در شکل ۲.۲ ترسیم شده، که هرچه قسمت سیاه یک نورون بیشتر باشد، یعنی احتمال اسپایک زدن بیشتری دارد). در لحظه‌ی بعد از اعمال ورودی، نورون‌های لایه‌ی بعد متناسب با احتمالشان، اسپایک می‌زنند. سپس مجددا اسپایک زدن یا نزدن های این لایه، رفتار لایه‌ی بعد (در شکل ۲.۲ لایه‌ی آخر) را مشخص می‌کنند و در لحظه‌ی زمانی بعد هم نورون‌های این لایه متناسب با احتمال‌های ایجاد شده روی خود اسپایک می‌زنند و این فرآیند تا انتهای شبکه ادامه می‌یابد.

اما بخاطر رفتار احتمالاتی شبکه، به ازای یک ورودی مشخص هم اتفاقات مختلفی می‌تواند در شبکه رخ دهد:



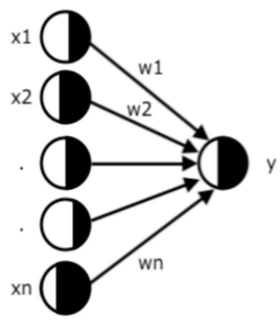
شکل ۲.۳ – دو حالت ممکن از رفتار شبکه به ازای یک ورودی یکسان

برای مثال شکل بالا (شکل ۲.۳)، دو اتفاق مختلف در لایه‌ی دوم شبکه را به ازای ورودی داده شده‌ی مشخص نشان می‌دهد، که در هر کدام نتیجه و احتمال‌ها در لایه‌ی خروجی متفاوت است. این وابستگی احتمالات لایه‌ها به اتفاقات لایه‌ی قبلشان، کار تحلیل کامل رفتار شبکه از ابتدا تا انتها را دشوار می‌کند. در ادامه (بخش ۳) به استخراج این روابط می‌پردازیم.

۳ تحلیل مقدماتی رفتار شبکه و آموزش ضرایب

۳.۱ تحلیل رفتار احتمالاتی نورون

برای تحلیل رفتار شبکه، لازم است رفتار نورون‌ها را در حالتی بررسی کنیم که از اتفاقات لایه‌ی قبل خود تنها به صورت احتمالاتی مطلع است (شکل ۳.۱):



شکل ۳.۱ – نورون احتمالاتی با ورودی‌های نایقینی

برای تحلیل روابط، ابتدا یک قضیه معرفی می‌کنیم:

قضیه‌ی برنولی بودن احتمال اسپایک زدن: احتمال اسپایک زدن یک نورون در شبکه به طور کلی (بدون دانستن بقیه‌ی اتفاقات شبکه در لایه‌های قبل) از توزیع برنولی معرفی می‌کند.

می‌دانیم هر متغیر تصادفی دو مقداره‌ای توزیع برنولی دارد، اما چیزی که در قضیه‌ی قبل روی آن تاکید می‌کنیم، جدا کردن قید وابسته بودنِ احتمال اسپایک زدن یک نورون به اتفاقاتِ یقینی لایه‌ی قبل خود است. یعنی هر نورون، مشروط بر آن که در لحظه‌ی قبل چه اتفاقات یقینی‌ای در لایه‌ی قبل از خود افتاده احتمال متفاوتی برای اسپایک زدن دارد اما با حذف کردن اثر متغیرهای مشروط (در واقع امید ریاضی گرفتن روی تمام اتفاقات لایه‌ی قبل) این احتمال یک عدد مشخص بدست می‌آید.

طبق این قضیه و توضیحات بالا، احتمال اسپایک زدن یک نورون در شبکه از رابطه‌ی شرطی زیر (رابطه ۳.۱) بدست می‌آید (دقت شود که اسپایک زدن نورون خروجی، در لحظه‌ی زمان بعد از ورودی صورت می‌گیرد و اندیس زمان را به دلیل سادگی حذف کرده‌ایم):

$$\mathrm{p_y} = \sum_{\mathbf{x}\in \mathrm{X_s}} \mathbb{P}(\mathrm{X} = \mathbf{x}).\, \mathbb{P}\big(\mathrm{y} = 1 \mid \mathrm{X} = \mathbf{x}\big) \tag{3.1}$$

که X_s مجموعه‌ی زیر است:

$$\mathrm{X_s} = \{0,1\}^n = \{[00\ldots0],[00\ldots1],\ldots,[11\ldots0],[11\ldots.1]\} \tag{3.2}$$

و همه حالات (جایگشت‌های) ممکن برای اسپایک زدن نورون‌های ورودی را نشان می‌دهد.

همانطور که نورون احتمالاتی را تعریف کردیم، برای خروجی شرطی داریم:

$$\mathbb{P}\big(\mathrm{y}=1 \mid \mathrm{X}=\mathbf{x}\big)=\sigma(\mathbf{w}^{\mathrm{T}}.\mathbf{x}) \tag{3.3}$$

پس رابطه‌ی احتمال اسپایک زدن یک نورون به صورت زیر در می‌آید.

$$\mathrm{p_y}=\textstyle\sum_{\mathbf{x}\in \mathrm{X_s}} \mathbb{P}(\mathbf{x}).\,\sigma(\mathbf{w}^{\mathrm{T}}.\mathbf{x}). \tag{3.4}$$

درباره‌ی $\mathbb{P}(\boldsymbol{x})$ در بخش ۳.۲ و ۴.۱ مفصل صحبت می‌شود، اما برای روشن‌تر شدن موضوع در اینجا آن را برای حالتی که اسپایک زدن‌های نورون‌های ورودی از هم مستقل باشند محاسبه می‌کنیم. با فرض اینکه نورون i ام با احتمال p_i اسپایک می‌زند و x_i ها از هم مستقل اند، داریم:

$$\mathbb{P}(\mathbf{x})=\mathrm{q_1 q_2 \dots q_n} \tag{3.5}$$

$$q_i=\begin{cases} p_i & x_i=1\\ 1-p_i & x_i=0\end{cases} \tag{3.6}$$

که این رابطه نشان می‌دهد برای هر جایگشت x احتمال متفاوتی برای اتفاق افتادن وجود دارد. برای مثال:

$$\mathbb{P}([00\dots0])=(1-\mathrm{p_1})(1-\mathrm{p_2})\dots(1-\mathrm{p_n}) \tag{3.7}$$

و:

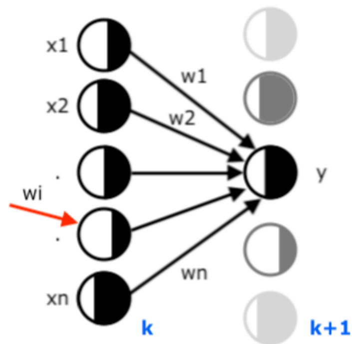
$$\mathbb{P}([11\dots1])=p_1p_2\dots p_n. \tag{3.8}$$

در حالت کلی که احتمال اسپایک زدن‌های نورون‌های یک لایه از هم مستقل نیستند، لازم است توزیع مشترک[۱۶] احتمالات هر لایه را بدانیم که در ادامه (بخش ۳.۲) روابط آن به تفصیل مورد بحث قرار می‌گیرند.

[16] Joint Distribution

۳.۲ تحلیل روابط بین لایه‌های شبکه

طبق توضیح ارائه شده در بخش ۳.۱، برای محاسبه‌ی $\mathbb{P}(\boldsymbol{x})$ نیاز به توزیع مشترک اسپایک زدن نورون‌های یک لایه داریم. دو لایه‌ی متوالی دلخواه در شبکه‌ی عمیق را درنظر می‌گیریم (لایه‌ی اول n نورون و لایه‌ی دوم m نورون دارد):



شکل ۳.۲ – دو لایه‌ی متوالی شبکه

روابط احتمالاتی کامل که تعمیمی از رابطه‌ی ۳.۱ (مربوط به تک نورون) است به صورت زیر است:

$$\mathbb{P}(\mathrm{Y}=\mathbf{y}) = \sum_{\mathbf{x}\in \mathrm{X_s}} \mathbb{P}(\mathrm{X}=\mathbf{x}).\,\mathbb{P}\big(\mathrm{Y}=\mathbf{y}\mid \mathrm{X}=\mathbf{x}\big) =$$

$$= \sum_{\mathbf{x}\in \mathrm{X_s}} \mathbb{P}(\mathrm{X}=\mathbf{x}).\prod_{\mathrm{j}=1}^{\mathrm{m}} \mathbb{P}\big(\mathrm{Y_j}=\mathbf{y_j}\mid \mathrm{X}=\mathbf{x}\big) =$$

$$= \textstyle\sum_{\mathbf{x}\in \mathrm{X_s}} \mathbb{P}(\mathrm{X}=\mathbf{x}).\prod_{\mathrm{j}=1}^{\mathrm{m}} \mathrm{f}_{\mathrm{y_j}}(\mathbf{w_j}^{\mathrm{T}}.\,\mathbf{x}) \tag{3.9}$$

که از رابطه‌ی استقلال زیر (شبکه‌ی ما یک شبکه‌ی بیز[۱۷] است) استفاده شده است:

$$\mathrm{Y}_i \perp\!\!\!\perp \mathrm{Y_j} \mid \mathbf{X} \tag{3.10}$$

[17] Bayesian Network

یعنی رفتار نورون‌های هر لایه با دانستن اطلاعات نورون‌های لایه‌ی قبل، از هم مستقل اند زیرا نورون‌های لایه‌ی قبل common cause هستند؛ پس احتمال‌های مشروط در هم ضرب می‌شوند. f از رابطه‌ی زیر (رابطه ۳.۱۱) بدست می‌آید:

$$f_{y_j}(\mathbf{w_j}^T.\mathbf{x}) = \begin{cases} p_{\mathbf{y_j}}(\mathbf{x}) & y_j = 1 \\ 1 - p_{\mathbf{y_j}}(\mathbf{x}) & y_j = 0 \end{cases} \tag{3.11}$$

که در آن:

$$p_{\mathbf{y_j}}(\mathbf{x}) = \mathbb{P}(y_j = 1 \mid X = \mathbf{x}) = \sigma(\mathbf{w_j}^T.\mathbf{x}) \tag{3.12}$$

دقت شود که هر نورون لایه‌ی دوم، با ضریب مخصوص خود، w_j ، به نورون‌های لایه‌ی قبل مربوط می‌شود (مشابه شبکه‌های DNN).

۳.۳ درباره‌ی بار محاسباتی زیاد

با معادلات معرفی شده (در راس آن رابطه ۳.۹)، احتمال اسپایک زدن همه‌ی نورون‌های شبکه بدست می‌آید اما این محاسبه بار بیشتری از اجرای شبکه و بدست آوردن اسپایک‌های یقینی دارد. در جمع‌ها، تمام جایگشت‌های ممکن اسپایک زدن ورودی ظاهر شده است که متناظر با تمامی مقادیر توزیع احتمال مشترک است. در اجرای یقینی شبکه، در هر لایه، لازم است برای هر نورون در لایه‌ی دوم (به تعداد n نورون) پتانسیل آن ناشی از اثر نورون‌های لایه‌ی اول (به تعداد m نورون) را محاسبه کنیم؛ که یعنی O(mn) محاسبه لازم داریم. اما در محاسبات دقیق احتمالی معرفی شده، باید تمام مقادیر توزیع احتمال مشترک لایه‌ی دوم را بدست آوریم که برای هر مقدار، جمع روی تمام حالات توزیع احتمال مشترک لایه اول لازم است که محاسباتی به اندازه $O(2^m.2^n)$ لازم دارد؛ یعنی تعداد محاسبات خطی بر حسب نورون لایه‌ها به تعداد نمایی افزایش پیدا کرده‌است و این بار محاسباتی نمایی می‌تواند محدود کننده باشد. همین جا یک تفاوت بین این شبکه و شبکه‌های DNN مشخص می‌شود؛ با اینکه در هر دو شبکه پتانسیل نورون‌ها با ترکیب خطی وزن‌دار نورون‌های لایه قبلش مشخص شده و مقدار این پتانسیل در هر دو شبکه وارد تابع سیگموید می‌شود، بار محاسباتی کاملاً متفاوتی لازم دارند. زیرا در DNN همه‌ی مقادیر یقینی است و حالت یک لایه، با دانستن مقادیر نورون‌های آن کاملاً مشخص می‌شود اما در شبکه‌ی احتمالاتی معرفی شده، دانستن مقادیر توزیع احتمال مشترک است که حالت هر لایه را مشخص می‌کند و این یعنی تفاوت نمایی در بردار نمایش دهنده‌ی هر لایه.

۳.۴ مقدمه‌ای درباره‌ی آموزش شبکه

(الگوریتم Gradient Descent و ایده‌ی Error Backpropagation)

در بخش‌های ۳.۱ تا ۳.۳، روابط محاسبات Forward شبکه را بدست آورده ایم. در این بخش خطا را برای آموزش شبکه معرفی کرده و نحوه‌ی بروزرسانی ضرایب را نشان می‌دهیم.

معمولاً، خروجی مطلوب یک شبکه به صورت یقینی است (یعنی در این مقاله به استفاده از خاصیت و توانایی‌های نایقینی این شبکه کاری نداریم و از آن برای کاربردهای معمول یقینی مانند دسته بندی استفاده می‌کنیم). تابع هزینه‌ی شبکه را به شکلی تعریف می‌کنیم که اگر نورون خروجی مطلوب است ۱ باشد، احتمال اسپایک زدن آن تا جای ممکن به ۱ نزدیک باشد و اگر خروجی مطلوبِ یک نورون صفر است، احتمال اسپایک زدن آن هم تا جای ممکن کم باشد. احتمال اسپایک زدن خود عددی بین ۰ و ۱ است چون این احتمال، خروجی تابع لوجستیک یا سیگموید است پس تابع‌های هزینه‌ی لوجستیکی برای این شبکه‌ها مناسب اند.

برای مثال دو شکل زیر قابل استفاده اند (y بیانگر لایه‌ی خروجی است):

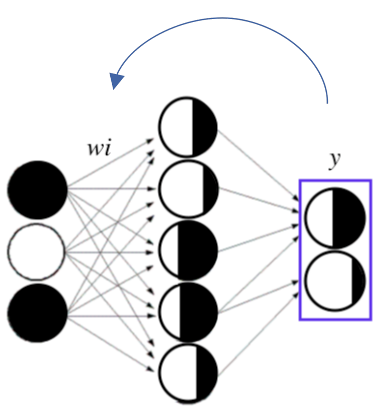
$$\text{Loss} = \text{Loss}(\mathbf{p_y}) = \sum_y (\text{p}_\text{y} - \text{y}_\text{ideal})^2 \tag{3.13}$$

و یا:

$$\text{Loss} = \text{Loss}(\mathbf{p_y}) = -\sum_y [\text{y}_\text{ideal}\, log(\text{p}_\text{y}) + (1 - \text{y}_\text{ideal})\, log(1 - \text{p}_\text{y})] \tag{3.14}$$

که جمع روی تمام نورون‌های لایه‌ی خروجی حساب می‌شود.

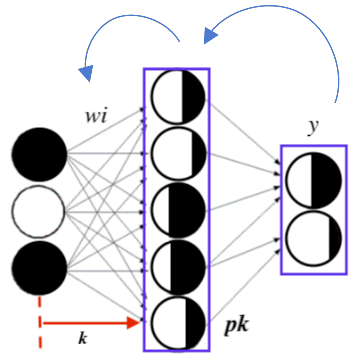
وابستگی تابع هزینه به خروجی نورون های لایه‌ی خروجی صریح و واضح است پس گرادیان تابع هزینه هم نسبت به مقادیر نورون‌های این لایه به سادگی محاسبه می‌شود. برای به‌روزرسانی ضرایب، به مشتق تابع هزینه نسبت به پارامترهای شبکه نیاز داریم.



$$\frac{\partial Loss}{\partial w_i} = \ ?$$

شکل ۳.۳ – لازم داشتن مشتق تابع هزینه نسبت به ضرایب شبکه

برای محاسبه‌ی این مشتق، مشابه ایده‌ی شبکه‌های عمیق عمل می‌کنیم. یعنی مرحله به مرحله، مشتق تابع هزینه نسبت به نورون‌های لایه‌های پیشین را به کمک مشتق تابع هزینه نسبت به نورون‌های لایه‌های پسین محاسبه می‌کنیم و در طی این عقب رفتن مشتق تابع هزینه نسبت به ضرایب پیشین هر لایه نیز قابل محاسبه می‌شود (شکل ۳.۴):



شکل ۳.۴ – شکستن مشتق به مشتقات زنجیره‌ای

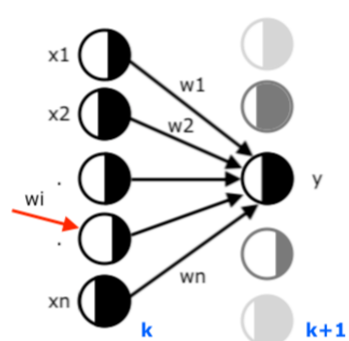
$$\mathbf{p_y} = \mathrm{f}(\mathbf{p_k}) \quad (3.15)$$

$$\frac{\partial \mathrm{Loss}}{\partial \mathrm{w_i}} = \frac{\partial \mathrm{Loss}}{\partial \mathbf{p_k}} \cdot \frac{\partial \mathbf{p_k}}{\partial \mathrm{w_i}} \quad (3.16)$$

که منظور از $\boldsymbol{p_k}$ ، بردار احتمال اسپایک زدن نورون‌های لایه‌ی k ام است. همانطور که گفته شد، جمله‌ی اول رابطه‌ی ۳.۱۶ بصورت زنجیره‌ای به عقب محاسبه می‌شود:

$$\frac{\partial \mathrm{Loss}}{\partial \mathbf{p_k}} = \frac{\partial \mathrm{Loss}}{\partial \mathbf{p_{k+1}}} \cdot \frac{\partial \mathbf{p_{k+1}}}{\partial \mathbf{p_k}} \quad (3.17)$$

که با دانستن مشتق تابع هزینه نسبت به نورون‌های لایه خروجی، می‌توان با رابطه‌ی بالا به صورت زنجیره‌ای عقب آمد و مشتق تابع هزینه نسبت به نورون‌های همه‌ی لایه‌ها را محاسبه کرد. جمله‌ی دوم رابطه‌ی ۳.۱۷ به صورت زیر محاسبه می‌شود:



شکل ۳.۵ – دو لایه‌ی متوالی در شبکه

$$p_y = \sum_{\boldsymbol{x}\in X_s} \mathbb{P}(\boldsymbol{x}).\,\sigma(\boldsymbol{w}^T.\,\boldsymbol{x}) \qquad (3.18)$$

$$\frac{\partial p_y}{\partial p_{x_i}} = \sum_{\boldsymbol{x}\in X_s} \frac{\partial \mathbb{P}(\boldsymbol{x})}{\partial p_{x_i}}.\,\sigma(\boldsymbol{w}^T.\,\boldsymbol{x}) \qquad (3.19)$$

اما در این محاسبه یک نکته وجود دارد. همانطور که اشاره شد، برای توصیف یک لایه، بردار احتمالات نورون‌های آن ($\boldsymbol{p_k}$) کافی نیست و اطلاع از تمام توزیع احتمال مشترک لازم است؛ پس رابطه‌ی بالا وقتی معنی‌دار است که مشتقات را نسبت به مولفه‌های توزیع مشترک بگیریم. این عدم دقت را در بخش‌های بعدی، با معرفی تحلیل خطی دقیق شبکه رفع می‌کنیم و در این قسمت تنها به شهود شبکه درباره‌ی ممکن بودن تعریف پس‌انتشار خطا در آن و استفاده از الگوریتم نزول گرادیان بسنده می‌کنیم.

طبق توضیح بالا، رابطه‌ی ۳.۱۶ وقتی دقیق است که مشتق نسبت به یک مولفه از توزیع احتمال مشترک محاسبه شده باشد. اما عدم پافشاری روی دقت (که در بخش‌های بعدی رفع می‌شود) فرض می‌کنیم رابطه برای مقادیر تک نورون‌ها هم برقرار است. جمله‌ی اول رابطه‌ی ۳.۱۶ را بسط دادیم، و حال بسط جمله‌ی دوم آن را ادامه می‌دهیم. در رابطه‌ی ۳.۴، احتمال اسپایک زدن نورون را بدست آوردیم (رابطه‌ی ۳.۲۰ تکرار رابطه‌ی ۳.۴ است):

$$p_{y_j} = \sum_{\boldsymbol{x}\in X_s} \mathbb{P}(\boldsymbol{x}).\,\sigma(\boldsymbol{w_j^T}.\,\boldsymbol{x}) \qquad (3.20)$$

پس گرادیان آن نسبت به ضرایب مربوطه‌اش از رابطه‌ی زیر بدست می‌آید:

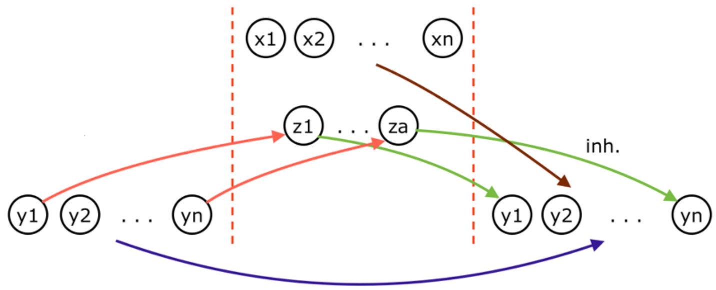
$$\frac{\partial p_{y_j}}{\partial \boldsymbol{w_j}} = \sum_{\boldsymbol{x}\in X_s} \boldsymbol{x}.\,[\mathbb{P}(\boldsymbol{x}).\,\sigma'\big(\boldsymbol{w_j^T}.\,\boldsymbol{x}\big)] \qquad (3.21)$$

و این ضریب روی بقیه‌ی نورون‌های این لایه اثری ندارد و فقط مقادیر نورون متناظر با خودش را تعیین می‌کند.
بدین ترتیب کلیاتی از روابط آموزش شبکه را در این بخش معرفی کردیم. روابط دقیق آن در بخش ۴ استخراج می‌شوند.

۴ تحلیل دقیق رفتار شبکه و آموزش ضرایب

۴.۱ تحلیل دقیق شبکه

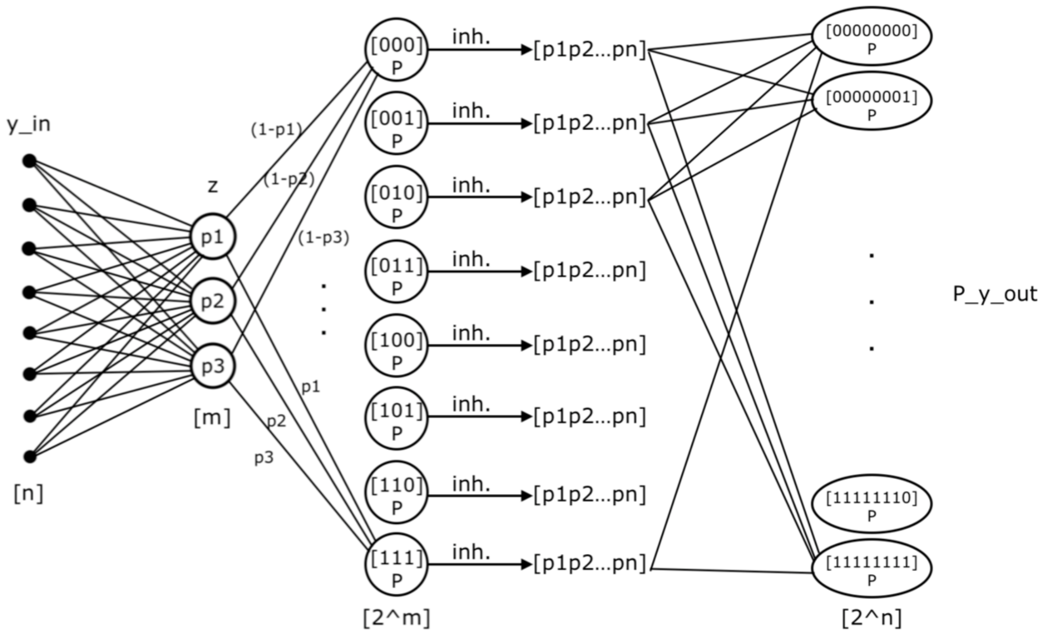
شبکه‌ی زیر را برای مثال درنظر بگیرید:



شکل ۴.۱ – یک شبکه‌ی نمونه (Winner Takes All)

این شبکه در حقیقت شبکه‌ی Winner Takes All مربوط به مرجع [۲] است اما بحث آتی برای هر شبکه‌ای صادق است و از این ساختار تنها به عنوان یک ساختار نمونه برای معرفی روابط استفاده می‌کنیم. خروجی این شبکه (y_out)، با سه ورودی x ، y_in و z تعیین می‌شود. خود z نیز با y_in مشخص می‌شود. برای ساده‌تر شدن تحلیل شبکه و نشان دادن روابط آن، فرض می‌کنیم خروجی (y_out) تنها با z تعیین می‌شود.

می‌خواهیم به جای نورون‌های هر لایه، مقادیر توزیع احتمال مشترک را به عنوان نورون‌های فرضی جدیدی در شبکه قرار دهیم و از این به بعد به این کار، گستراندن می‌گوییم. شبکه را می‌توان آن را به صورت زیر (شکل ۴.۲) گستراند:



شکل ۴.۲ – شبکه‌ی گسترده‌شده

که به جای آنکه هر نورون را نمایش دهیم، هر ترکیب توزیع احتمال مشترک را نمایش می‌دهیم. چگونگی این گستراندن در لایه‌ی ابتدایی به خوبی نمایش داده شده است. یک ورودی یقینی دلخواه (واضحاً ورودی شبکه همیشه یقینی است)، که در شکل با y_in نشان داده شده، احتمال اسپایک زدن نورون‌های لایه دوم (z) را مشخص می‌کند (که در شکل با p1 تا pm (m=3) مشخص شده اند). پس احتمال رخ دادن تمام حالات اسپایک زدن نورون‌های z را داریم، که در لایه‌ی بعدی متناظر با ۰۰۰ تا ۱۱۱ نمایش داده شده است. متناظر با هر کدام، یا هر اتفاق در لایه‌ی z ، یک احتمال برای اسپایک زدن نورون‌های لایه‌ی بعد به دست می‌آید که در شکل مقابلِ هر اتفاق در لایه‌ی z با p1 تا pn نشان داده شده است. هر کدام از این اتفاقات، مانند آنچه در لایه‌ی ابتدایی توصیف شد، با احتمالات متفاوتی می‌تواند به هر اتفاقی در لایه‌ی بعد بیانجامد و خروجی گسترده شده‌ی نهایی (لایه‌ی انتهایی در شکل) ترکیب وزن دار به احتمال هر کدام از حالات لایه‌ی میانی است.

پس روابط زیر برقرار است (که همان رابطه‌ی قبل یا رابطه‌ی ۳.۹ است که دوباره آورده شده و به شکل ماتریسی توصیف شده است):

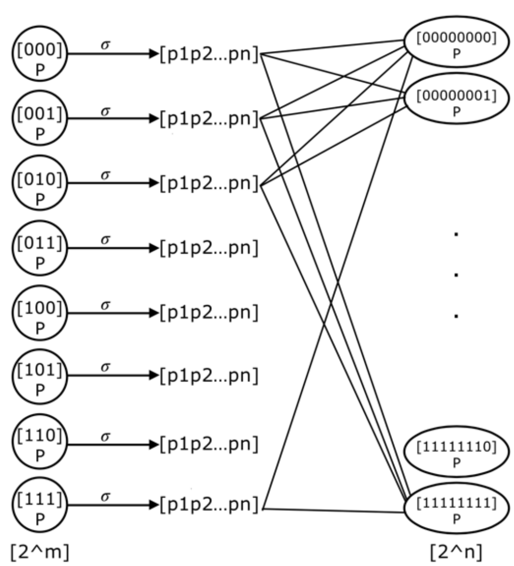
$$\mathbb{P}(Y = \boldsymbol{y}) = \sum_{\boldsymbol{x}\in X_s} \mathbb{P}(X = \boldsymbol{x}).\prod_{j=1}^{n} f_{y_j}(\boldsymbol{w_j}^T.\boldsymbol{x}) \qquad (4.1)$$

$$\Rightarrow \overrightarrow{\mathbb{P}(Y)} = \mathrm{W}.\overrightarrow{\mathbb{P}(X)} \qquad (4.2)$$

که در آن:

$$\overrightarrow{\mathbb{P}_m(X)} = \begin{pmatrix} \mathbb{P}(X{=}000\ldots0) \\ \mathbb{P}(X{=}000\ldots1) \\ \ldots \\ \mathbb{P}(X = 111\ldots1) \end{pmatrix},\ \ \overrightarrow{\mathbb{P}_n(Y)} = \begin{pmatrix} \mathbb{P}(Y{=}000\ldots0) \\ \mathbb{P}(Y{=}000\ldots1) \\ \ldots \\ \mathbb{P}(Y = 111\ldots1) \end{pmatrix} \qquad (4.3)$$

و رابطه‌ی ۴.۲ رابطه‌ی خطی است! پس می‌توان یک شبکه‌ی کامل و دلخواه را با روابط خطی به صورت زیر توصیف کرد که رابطه‌ی کل شبکه را در ادامه می‌آوریم. طبق معادله‌ی بالا، ضریب لایه (W) به صورت زیر است (روابط ۴.۴ و ۴.۵):



شکل ۴.۳ – دو لایه‌ی متوالی در شبکه‌ی گسترده شده

$$W_{2^n \times 2^m} = \begin{pmatrix} \prod_{j=1}^{n}(1-\sigma(\boldsymbol{w_j}^T.\boldsymbol{x_0})) & \cdots & \prod_{j=1}^{n}(1-\sigma(\boldsymbol{w_j}^T.\boldsymbol{x_{2^m}})) \\ \vdots & \ddots & \vdots \\ \prod_{j=1}^{n}\sigma(\boldsymbol{w_j}^T.\boldsymbol{x_0}) & \cdots & \prod_{j=1}^{n}\sigma(\boldsymbol{w_j}^T.\boldsymbol{x_{2^m}}) \end{pmatrix} \quad (4.4)$$

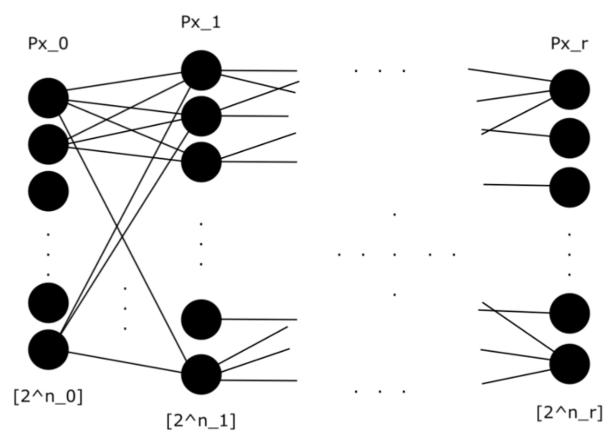
$$[W_{2^n \times 2^m}]_{(a,b)} = \prod_{j=1}^{n} f_{a_j}\left(\boldsymbol{w_j}^T.\boldsymbol{x}_b\right) \quad (4.5)$$

$$\begin{aligned} &\boldsymbol{x}_b = bin(b) \\ &a_j = bin(a)[j] \\ &f_{a_j}(\boldsymbol{w_j}^T.\boldsymbol{x}) = \begin{cases} \sigma(\boldsymbol{w_j}^T.\boldsymbol{x}) & y_j = 1 \\ 1-\sigma(\boldsymbol{w_j}^T.\boldsymbol{x}) & y_j = 0 \end{cases} \end{aligned}$$

که $W_{2^n \times 2^m}$ به طور موثر n×m پارامتر آزاد دارد، که همان $\boldsymbol{w_j}$ ها هستند ($\boldsymbol{w_1}$ تا $\boldsymbol{w_n}$ که هر کدام بردار m پارامتری است).

پس رابطه‌ی کل شبکه به صورت زیر (رابطه‌ی ۴.۶) درمی‌آید:

$$\overrightarrow{\mathbb{P}(X_r)} = W_{2^{n_r} \times 2^{n_{r-1}}} \ldots W_{2^{n_1} \times 2^{n_0}}.\overrightarrow{\mathbb{P}(X_0)} \quad (4.6)$$



شکل ۴.۴ – یک شبکه‌ی گسترده‌شده

که به دلیل آنکه ورودی شبکه یک بردار یقینی است، ورودی شبکه‌ی گسترده شده در احتمالات مشترک به صورت زیر است:

$$\overrightarrow{\mathbb{P}(X_0 = x_{in})} = \mathbb{I}[X_0 = x_{in}] \tag{4.7}$$

یعنی به احتمال ۱، همان نورون متناظر با ورودی اتفاق افتاده فعال است و بقیه کاملا خاموش بوده و احتمال صفر را نشان می‌دهند.

پس ورودی شبکه‌ی گسترده شده، در عمل تنها با ضرب شدن در یک ضریب خروجی احتمال‌های لایه‌ی خروجی را بدست می‌دهد:

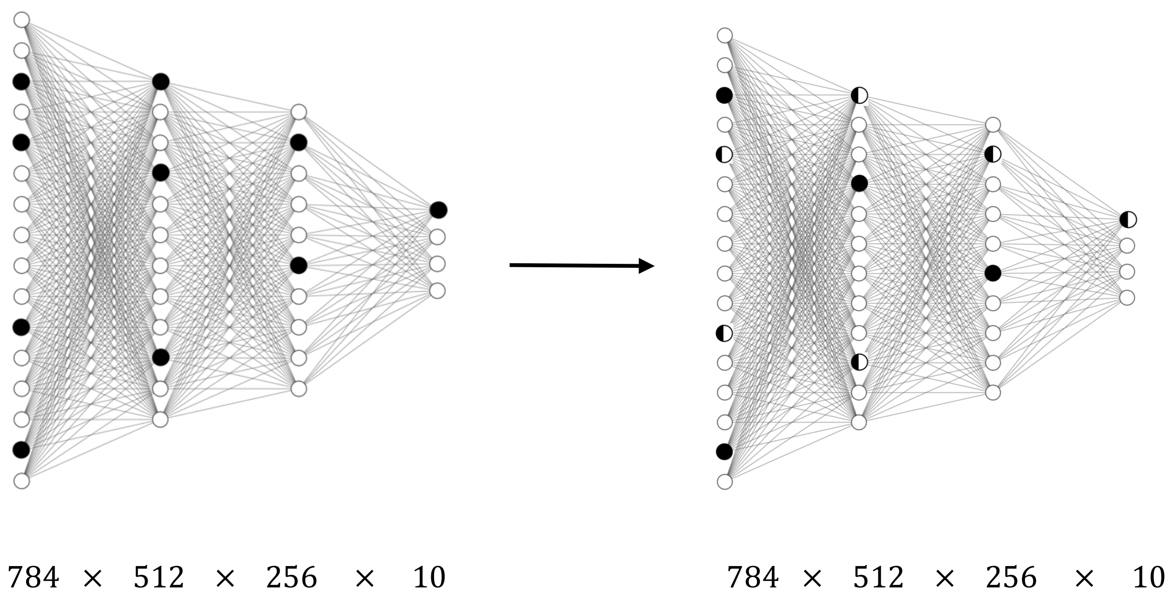
$$\overrightarrow{\mathbb{P}(X_r)} = W^{total}_{2^{n_r}\times 2^{n_0}}.\overrightarrow{\mathbb{P}(X_0)} \tag{4.8}$$

در نگاه اول، نگران بزرگی درمورد این شبکه‌ی PNN به وجود می‌آید. حس امروزی نسبت به شبکه‌های عمیق امروزی، این است که موفقیتشان بخاطر فعال‌سازهای غیر خطی استفاده شده در آنها (سیگموید، تانژانت هیپربولیک، ...) است و شبکه‌های خطی که مشابه رابطه‌ی قبل، در واقع به یک لایه خطی قابل فروکاست هستند، قدرت شبکه‌های غیر خطی را نداشته و معمولا موفقیتی در دسته‌بندی داده‌های ساده را نیز ندارند. پس این نگرانی مطرح می‌شود که آیا این شبکه‌ی PNN قدرت پردازشی کافی را دارد؟ در بخش بعد (بخش ۴.۲)، با آزمایش کردن عملی این شبکه نشان می‌دهیم شبکه قدرت پردازش کافی را دارد.

۴.۲ آموزش ساده‌انگارانه‌ی شبکه‌ی عصبی احتمالاتی

(Learning Naïve Transfer)

قدرت پردازش شبکه را به ساده‌ترین شکل ممکن آزمایش می‌کنیم؛ بدین صورت که ابتدا یک شبکه‌ی عمیق معمولی با همان ساختار شبکه‌ی احتمالاتی را آموزش می‌دهیم، و ضرایب آموزش دیده شده‌ی آن را عیناً انتقال می‌دهیم. برای مثال، شبکه‌ی زیر در شکل ۴.۵ را درنظر می‌گیریم و می‌خواهیم آن را روی مجموعه داده‌ی اعداد دست‌نویس (MNIST) آموزش دهیم:



شکل ۴.۵ – سمت چپ: یک شبکه‌ی عمیق عادی (DNN) برای دسته‌بندی MNIST
و سمت راست: شبکه‌ی احتمالاتی با همان ساختار برای انجام عملی مشابه

(توجه شود شبکه‌ی سمت راست در شکل ۴.۵ شبکه احتمالاتی اصلی است نه شبکه‌ی احتمالاتی گسترده شده.)

برای آزمایش عملکرد شبکه، تصویر ورودی را در لایه‌ی ورودی قرار می‌دهیم. درنتیجه‌ی ضرایب، پتانسیل اسپایک زدن لایه‌ی بعد مشخص شده که احتمال اسپایک زدن آن لایه را تعیین می‌کند. با این احتمال‌ها نمونه گیری می‌کنیم و مقادیر مشخصی برای اسپایک زدن یا نزدن نورون‌های این لایه مشخص می‌شود. همین کار را تا لایه‌ی آخر ادامه می‌دهیم و در پایان، نورون‌های لایه‌ی خروجی مقادیر صفر یا یک را متناظر با اسپایک زدن یا اسپایک نزدن اختیار کرده‌اند، که مطلوب این است که نورون خروجی متناظر با دیجیت تصویر ورودی فعال بوده و بقیه غیر فعال باشند. به دلیل ذات احتمالاتی شبکه، ۵۰ بار این نمونه‌گیریِ از ابتدا تا انتها را انجام می‌دهیم و این بین این ۵۰ آزمایش رای اکثریت می‌گیریم و نورونی که بیشتر از همه در این آزمایش‌ها فعالَ بوده را به عنوان خروجی شبکه به ازای این تصویر اعلام می‌کنیم.

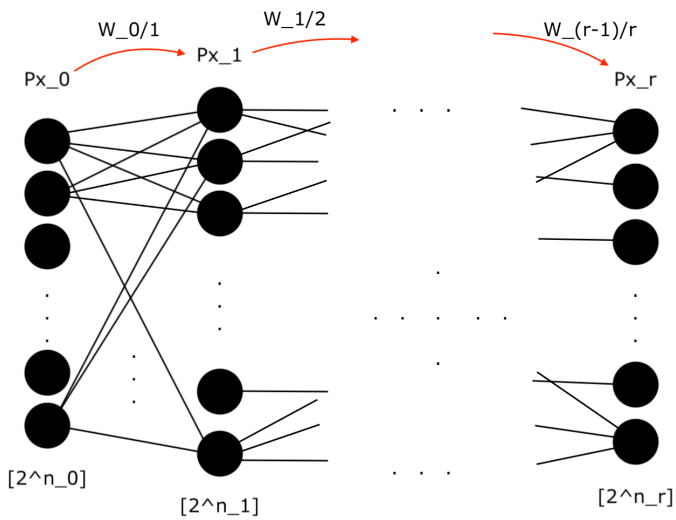
شبکه‌ی عمیق معمولی (سمت چپ) را با دقیق Train برابر ۱۰۰ درصد و دقت Test برابر ۹۸.۲۷ درصد آموزش داده‌ایم.

نتیجه‌ی شبکه‌ی احتمالاتی قابل توجه است؛ دقت Test برابر ۹۸.۱۵ درصد.

پس نتیجه می‌گیریم حتی با این آموزش که منحصراً برای شبکه‌ی احتمالاتی طراحی نشده هم شبکه نتیجه‌ای بسیار خوب نزدیک به شبکه‌ی عمیق معمولی می‌دهد پس این شبکه، قدرت پردازشی کافی را داراست.

۴.۳ آموزش شبکه با روابط خطی

در این بخش، تقریبا همان روابط بخش آموزش مقدماتی ۳.۴ را تکرار می‌کنیم اما با دید دیگر؛ زیرا بعد از آنکه روابط را به صورت خطی نوشتیم، درک روابط قبل ساده تر می‌شود. شبکه‌ی زیر (شکل ۴.۶) را درنظر می‌گیریم:



شکل ۴.۶ – یک شبکه‌ی گسترده شده

و دو لایه‌ی متوالیِ آن با رابطه‌ی زیر (رابطه‌ی ۴.۹) به هم مربوط می‌شوند:

$$\overrightarrow{\mathbb{P}(X_{\iota+1})} = W_{2^{n_{i+1}}\times 2^{n_i}}.\overrightarrow{\mathbb{P}(X_\iota)} \tag{4.9}$$

و تابع هزینه‌ی شبکه از روی ورودی تعیین می‌شود؛ لذا گرادیان تابع هزینه نسبت به احتمالات مشترک (لایه خروجی در شبکه‌ی گسترده شده) مشخص است:

$$Loss = Loss\left(\overrightarrow{\mathbb{P}(X_r)}\right) \tag{4.10}$$

$$\Rightarrow \frac{\partial Loss}{\partial \overrightarrow{\mathbb{P}(X_r)}} = \nabla_{\overrightarrow{\mathbb{P}(X_r)}} Loss^T = [\frac{\partial Loss}{\partial \mathbb{P}_0}, \dots, \frac{\partial Loss}{\partial \mathbb{P}_{2^{n_r}-1}}] \tag{4.11}$$

که منظور از $[\mathbb{P}_0, \dots, \mathbb{P}_{2^{n_r}-1}]$ در رابطه‌ی ۴.۱۱، مولفه‌های بردار $\overrightarrow{\mathbb{P}(X_r)}$ است.

فرض می‌کنیم گرادیان تابع هزینه نسبت به بردار احتمالات لایه i+۱ ام، $\overrightarrow{\mathbb{P}(X_{\iota+1})}$ را داشته باشیم. آنگاه گرادیان تابع هزینه نسبت به بردار احتمالات لایه i ام به صورت زیر (رابطه‌ی ۴.۱۲) محاسبه می‌شود:

$$\nabla_{\overrightarrow{\mathbb{P}(X_\iota)}} Loss^T = \frac{\partial Loss}{\partial \overrightarrow{\mathbb{P}(X_\iota)}} = \frac{\partial Loss}{\partial \overrightarrow{\mathbb{P}(X_{\iota+1})}} . \frac{\partial \overrightarrow{\mathbb{P}(X_{\iota+1})}}{\partial \overrightarrow{\mathbb{P}(X_\iota)}} = \nabla_{\overrightarrow{\mathbb{P}(X_{\iota+1})}} Loss^T . W_{2^{n_{i+1}} \times 2^{n_i}} \qquad (4.12)$$

یا محاسبه‌ی مستقیم آن از گرادیان نسبت به لایه‌ی خروجی:

$$\nabla_{\overrightarrow{\mathbb{P}(X_\iota)}} Loss^T = \nabla_{\overrightarrow{\mathbb{P}(X_r)}} Loss^T . W_{2^{n_r} \times 2^{n_{r-1}}} . \dots . W_{2^{n_{i+1}} \times 2^{n_i}} \qquad (4.13)$$

که انی گرادیان، مثل هر گرادیان دیگری مانند یک بردار هموردا (تانسور مرتبه یک هموردا) در فضا تبدیل می‌شود (درباره‌ی تعبیر هر لایه به عنوان یا فضای برداری در ادامه صحبت می‌شود).

حال باید مشتق تابع هزینه نسبت به یک ضریب خاص را پیدا کنیم. پس از محاسبه‌ی بالا در رابطه‌ی ۴.۱۲ و ۴.۱۳، کافیست مشتق تابع هزینه نسبت به یک ضریب را به گرادیان تابع هزینه نسبت به بردار احتمالات لایه‌ی بعدی آن ضریب مربوط کنیم. برای یک ضریب بین لایه‌ی i ام و i-1 ام داریم:

$$\overrightarrow{\mathbb{P}(X_\iota)} = W_{2^{n_i} \times 2^{n_{i-1}}} . \overrightarrow{\mathbb{P}(X_{\iota-1})} \qquad (4.14)$$

پس رابطه‌ی دیفرانسیلی برای این لایه به صورت زیر (رابطه‌ی ۴.۱۵) است:

$$\overrightarrow{\Delta\mathbb{P}(X_\iota)} = \Delta W_{2^{n_i} \times 2^{n_{i-1}}} . \overrightarrow{\mathbb{P}(X_{\iota-1})} \qquad (4.15)$$

که ΔW در آرگومان‌هایی که ضریب مدنظر وجود ندارد صفر است (البته می‌توان با همین رابطه‌ی بالا، از همه‌ی ضرایب شبکه توامان مشتق گرفت).

پس رابطه‌ی دیفرانسیلی برای تابع هزینه بصورت زیر (رابطه‌ی ۴.۱۶) است:

$$\Delta Loss = \nabla_{\overrightarrow{\mathbb{P}(X_\iota)}} Loss^T . \overrightarrow{\Delta\mathbb{P}(X_\iota)} = \nabla_{\overrightarrow{\mathbb{P}(X_\iota)}} Loss^T . \Delta W_{2^{n_i} \times 2^{n_{i-1}}} . \overrightarrow{\mathbb{P}(X_{\iota-1})} \qquad (4.16)$$

و فرمی به شکل رابطه‌ی رایلی دارد:

$$\Delta Loss = u^T A v \qquad (4.17)$$

که u در واقع جمله‌ی پس انتشار خطا تا این لایه، و v جمله‌ی فورارد است. به جایگذاریِ کامل v و u داریم:

$$\Delta Loss = \left(\nabla_{\overrightarrow{\mathbb{P}(X_r)}} Loss^T\right).W_{2^{n_r}\times 2^{n_{r-1}}}.\dots.W_{2^{n_{i+1}}\times 2^{n_i}}.\Delta W_{2^{n_i}\times 2^{n_{i-1}}}.W_{2^{n_{i-1}}\times 2^{n_{i-2}}}.\dots.W_{2^{n_1}\times 2^{n_0}}.\overrightarrow{\mathbb{P}(X_0)} \tag{4.18}$$

حال، رابطه‌ی ΔW را بیشتر شفاف می‌کنیم. همانطور که بدست آمد، W به صورت زیر است:

$$W_{2^n\times 2^m} = \begin{pmatrix} \prod_{j=1}^{n}(1-\sigma(\boldsymbol{w_j}^T.\boldsymbol{x_0})) & \cdots & \prod_{j=1}^{n}(1-\sigma(\boldsymbol{w_j}^T.\boldsymbol{x_{2^m}})) \\ \vdots & \ddots & \vdots \\ \prod_{j=1}^{n}\sigma(\boldsymbol{w_j}^T.\boldsymbol{x_0}) & \cdots & \prod_{j=1}^{n}\sigma(\boldsymbol{w_j}^T.\boldsymbol{x_{2^m}}) \end{pmatrix} \tag{4.19}$$

برای واضح‌تر شدن عملیات ریاضی در ادامه، عملگر $\odot$ را بین دو ماتریس هم اندازه به عنوان ضرب مولفه به مولفه تعریف می‌کنیم، یعنی تحت این عملگر مولفه‌های دو ماتریس در هم متناظراً ضرب شده و ماتریس هم اندازه‌ی دیگری نتیجه بدهند.

بنابراین برای ماتریس ضرایب می‌توان نوشت:

$$W_{2^n\times 2^m} = W^{(1)}_{2^n\times 2^m}\odot W^{(2)}_{2^n\times 2^m}\odot\dots\odot W^{(n)}_{2^n\times 2^m} \tag{4.20}$$

که هر کدام از آنها به صورت زیر است:

$$W^{(j)}_{2^n\times 2^m} = \begin{pmatrix} (1-\sigma(\boldsymbol{w_j}^T.\boldsymbol{x_0})) & \cdots & (1-\sigma(\boldsymbol{w_j}^T.\boldsymbol{x_{2^m}})) \\ \vdots & \ddots & \vdots \\ \sigma(\boldsymbol{w_j}^T.\boldsymbol{x_0}) & \cdots & \sigma(\boldsymbol{w_j}^T.\boldsymbol{x_{2^m}}) \end{pmatrix} \tag{4.21}$$

و در واقع اثر احتمال نورون j ام را در ماتریس کل نشان می‌دهد، و سطر k ام آن همان اسپایک زدن یا نزدن این نورون را تعیین می‌کند:

$$f_{y_{jk}}(\boldsymbol{w_j}^T.\boldsymbol{x}) = \begin{cases} \sigma(\boldsymbol{w_j}^T.\boldsymbol{x}) & y_{jk}=1 \\ 1-\sigma(\boldsymbol{w_j}^T.\boldsymbol{x}) & y_{jk}=0 \end{cases} \tag{4.22}$$

و برای مثال، ماتریس اول که نورون اول را نشان می‌دهد، اگر آنرا در قرارداد نمایشمان MSB گرفته باشیم، در نصف اولیه سطرها احتمال اسپایک نزدن (1-p) و در نصف دوم سطرها احتمال اسپایک زدن (p) را نشان می‌دهد. نورون n ام نیز با قرارداد LSB ، یکی درمیان احتمال اسپایک زدن و نزدن خود را نشان می‌دهد.

در این حالت، در هر ماتریس تنها ضرایب مربوط به یک نورون در لایه‌ی ثانویه ظاهر شده اند پس اگر مشتق به ضرایب یکی از نورون‌ها را بخواهیم، تنها کافی است که مشتق ماتریس مربوطه اش را نسبت به آن ضریب محاسبه کنیم، و مشتق ماتریس کلی لایه با همین ضرب مولفه به مولفه به شکل مشابهی بدست می‌آید:

$$\Delta W_{2^n\times 2^m} = W^{(1)}_{2^n\times 2^m}\odot W^{(2)}_{2^n\times 2^m}\odot\dots\odot[\Delta W^{(j)}_{2^n\times 2^m}]\odot\dots\odot W^{(n)}_{2^n\times 2^m} \tag{4.23}$$

پس کافیست مشتق ماتریس $W^{(j)}_{2^n \times 2^m}$ نسبت به $\boldsymbol{w_j}$ ها را محاسبه کنیم.

این کار را برای مولفه k ام آن (1< k <m) انجام می‌دهیم:

$$\frac{\partial W^{(j)}_{2^n \times 2^m}}{\partial \boldsymbol{w}^{(k)}_j} = \begin{pmatrix} -\sigma'\left(\boldsymbol{w_j}^T . \boldsymbol{x_0}\right) . \boldsymbol{x}^{(k)}_0 & \cdots & -\sigma'\left(\boldsymbol{w_j}^T . \boldsymbol{x_{2^m}}\right) . \boldsymbol{x}^{(k)}_{2^m} \\ \vdots & \ddots & \vdots \\ \sigma'\left(\boldsymbol{w_j}^T . \boldsymbol{x_0}\right) . \boldsymbol{x}^{(k)}_0 & \cdots & \sigma'\left(\boldsymbol{w_j}^T . \boldsymbol{x_{2^m}}\right) . \boldsymbol{x}^{(k)}_{2^m} \end{pmatrix} \tag{4.24}$$

$$\left[\frac{\partial W^{(j)}_{2^n \times 2^m}}{\partial \boldsymbol{w}^{(k)}_j}\right]_{ab} = (-1)^{1-bin(a)_j} . \sigma'\left(\boldsymbol{w_j}^T . \boldsymbol{x_b}\right) . \boldsymbol{x}^{(k)}_b \tag{4.25}$$

$$(\boldsymbol{x_b} = bin(b))$$

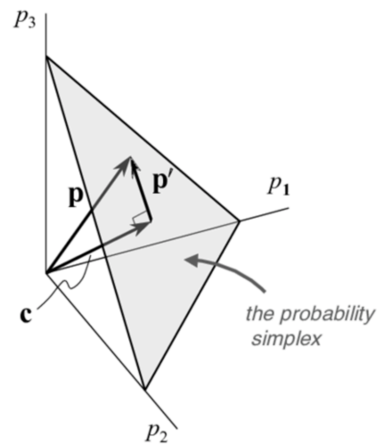
نکته‌ی قابل توجه در این ماتریس (رابطه‌ی ۴.۲۴) این است که ستون‌هایی که مولفه k ام نمایش باینری آنها صفر است، در این ماتریس صفر هستند.

با این روابط، گرادیان‌های لازم برای استفاده از الگوریتم نزول گرادیان بدست آمده و این الگوریتم قابل اجرا است.

۵ تعابیری از روابط شبکه

۵.۱ تعبیر فضای بردارهای احتمال

بردار هر لایه در شبکه‌ی گسترده شده، یک بردار احتمال روی فضای پیشامد نورون‌های آن است و ویژگی‌های یک بردار احتمالاتی (مانند مثبت بودن ۱ بودن جمع) را دارد.



شکل ۵.۱ – ابرصفحه‌ی احتمال در فضای نورون‌ها

ماتریس ضرایب بین هر دو لایه در شبکه‌ی گسترده شده نیز، مانند یک ماتریس تبدیل انتقال است که بردار را از فضای لایه‌ی اولیه به فضای لایه‌ی ثانویه می‌برد:

$$\overrightarrow{\mathbb{P}(Y)} = \mathrm{W}.\overrightarrow{\mathbb{P}(X)} \qquad (5.1)$$

که به دلیلی آنکه این تبدیل خواص احتمالاتی را حفظ می‌کند، و خودش جمع هر ستونش برابر ۱ است، به نوعی یک ماتریس احتمال گذار است. با این تعبیر، شبکه یک حالت اولیه در ورودی را گرفته و پس از گذار دادن‌هایی در لایه‌هایش، آنرا به حالتی در خروجی می‌برد که احتمال بودن در یک حالت نهایی در شبکه‌ی اصلی، با عدد شبکه‌ی گسترده شده‌اش مشخص می‌شود.

۵.۲ درباره‌ی خطی بودن

در دنیای احتمال، خطی بودن امری عجیب نیست. خطی بودن شبکه‌ی ما از رابطه‌ی احتمال شرطی نتیجه می‌شود که برای دو لایه‌ی متوالی داریم:

$$W_{2^{n_i}\times 2^{n_j}} = \mathbb{P}\left(X_i = \boldsymbol{x_i} | X_j = \boldsymbol{x_j}\right) \tag{5.2}$$

و برای کل شبکه نیز یک فاکتورگیری ساده‌ی بیز برقرا است:

$$\mathbb{P}(X_0, X_1, \dots, X_r) = \mathbb{P}(X_r|X_{r-1}). \dots . \mathbb{P}(X_1|X_0). \mathbb{P}(X_0) \tag{5.3}$$

و رابطه‌ی برداری ما، که تمام مقادیر $\mathbb{P}(X_i)$ را در یک بردار قرار داده‌ایم، با ضرب ماتریسی عمل جمع زدن روی مقادیر احتمال را انجام می‌دهد و تنها بردار مقادیر $\mathbb{P}(X_r)$ باقی می‌ماند.

۶ نتیجه‌گیری و کارهای آتی

در این مقاله، شبکه‌ای متشکل از نورون‌های احتمالاتی را شرح دادیم، بررسی کردیم، روابط ریاضی حاکم بر آن را به طور کامل بدست آوردیم، و نشان دادیم که این روابط در فضای احتمال‌های مشترک، خطی هستند. سپس روابط آموزش برای این شبکه را بدست آوردیم که برای آن، از ایده‌ی پس انتشار خطا استفاده کردیم و دیدیم که مشتق پذیر بودن روابط شبکه نسبت به ضرایب، این امکان را به ما می‌دهد که آموزش ضرایب را با الگوریتم نزول گرادیان انجام دهیم. در انتها نیز صحبت کوتاهی در مورد شهود برداری این روابط انجام دادیم. همچنین با یک آزمایش به صورت سرانگشتی (تحت عنوان آموزش ساده انگارانه) نشان دادیم که خطی بودن روابط شبکه، از قدرت پردازشی آن نمی‌کاهد.

روابط حاکم بر شبکه‌ی معرفی شده، در ماتریس‌هایی به ابعاد نمایی نسبت به تعداد نورون لایه‌ها، ولی به صورت خطی ظاهر شدند. استخراج این روابط این امکان را به داد که روابطی برای آموزش ضرایب آن انتخاب کنیم، لذا این شبکه می‌تواند به عنوان جایگزینی قابل آموزش برای شبکه‌های عمیق فعلی مورد استفاده قرار گیرد؛ در حالیکه دیدیم آموزش ساده‌انگارانه که زحمتی به اندازه‌ی آموزش شبکه‌های متداول دارد نیز پاسخ مطلوبی در اختیار ما قرار می‌دهد.
اما این شبکه می‌تواند قابلیت‌های دیگری نیز داشته باشد. مثلا می‌توان بدون آن که زمان زیادی برای آموزش دقیق آن صرف کرد، آن را اندکی آموزش داد و با احتمال مثبت و قابل قبولی خروجی مطلوب را مشاهده کرد. پس شبکه‌ای داریم که قابلیت‌های دیگری نیز برای ما فراهم می‌کند.

در این نوشته، به تحلیل ریاضی توان پردازشی شبکه نپرداختیم. یکی از مواردی که نیاز به دقیق‌تر شدن دارد، همین مورد است و دقیق‌تر کردن اثر گسترده‌کردن شبکه و رابطه‌ی تبدیل خطی آن؛ یا به عبارات دیگر دقیق‌تر پیدا کردن شکل تابعیت[18] شبکه با توجه به آن که رابطه‌ی خطی و صریحی برای آن پیدا کرده‌ایم. شباهت این شبکه و شبکه‌های متداول (که این شباهت را مناسب بودن یادگیری ساده‌انگارانه به ما نشان داد) هم انگیزه‌ای به ما می‌دهد که پیدا کردن شکل تابعیت این شبکه، حدودی از شکل تابعیت شبکه‌های متداول امروزی نیز ارائه دهد.
زمینه‌ی دیگر قابل کار، آموزش شبکه است. انتظار می‌رود به دلیل خطی بودن شکل روابط در شبکه‌ی گسترده شده، بتوان ایده‌ها و الگوریتم‌های بهتری برای آموزش این شبکه ارائه داد. در آموزشی که در این نوشته به آن اشاره کردیم (بک پروپگیشن) زمان آموزش زیاد است و فرم نمایی دارد ($O(r.2^m.2^n)$) که زمان مناسبی نیست. بهبود آن نیز جز زمینه‌های قابل کار است.
همچنین همانطور که اشاره شد، تمرکز روی کاربردهای دیگر این شبکه، مانند طراحی ایده‌هایی برای استفاده از قابلیت احتمالاتی بودن و چند نوع خروجی دادن آن نیز موضوع قابل کار دیگری در این زمینه است.

[18] Landscape

۷ منابع و مراجع

[1] A Basic Compositional Model for Spiking Neural Networks; Nancy Lynch, Cameron Musco; 2018

[2] Computational Tradeoffs in Biological Neural Networks: Self-Stabilizing Winner-Take-All Networks; Nancy Lynch, Cameron Musco, Merav Parter; 2016

Sharif University of Technology

Department of
Electrical Engineering

Spring 1398