دانشگاه صنعتی شریف

دانشکده‌ی مهندسی کامپیوتر

گزارش پروژه پایانی هوش مصنوعی

عنوان:

شبکه‌های عصبی

نویسنده :

علی قربان‌پور

استاد :

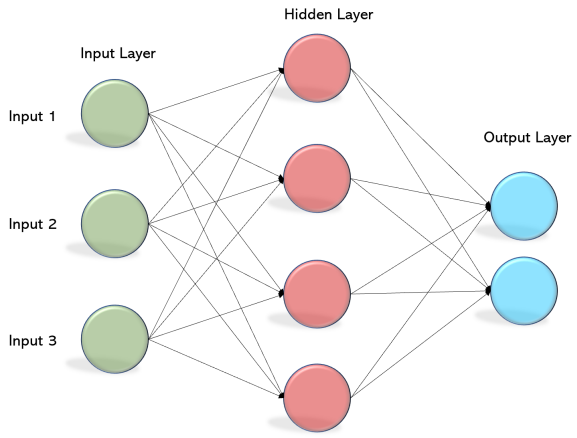
دکتر آرش عبدی هجران‌دوست

نیم‌سال دوم سال تحصیلی ۱۳۹۸ ـ ۱۳۹۹

چکیده

شبکه‌های عصبی یکی از الگوریتم‌های بسیار پرکاربرد در حوزه‌ی داده‌کاوی، یادگیری ماشین و هوش مصنوعی هستند. این الگوریتم‌ها در پردازشِ زبانِ طبیعی، شناساییِ تصاویر مانند شناساییِ چهره، تشخیصِ صحتِ امضا یا تشخیص تقلب در تراکنش‌های بانکی کاربردهای فراوانی دارند. در این پروژه همانطور که از عنوانش بر می‌آید قصد داریم به کمک پیاده‌سازی شبکه‌های عصبی و بهره‌گیری از توانایی یادگیری این شبکه‌ها به حل و بحث برخی مسائل رایج در این حوزه بپردازیم. برای پیاده‌سازی‌ها از پرسپترون‌های چندلایه ۱ استفاده می‌کنیم که دقت بالاتری در مقایسه با شبکه‌های عصبی عادی دارند. مفاهیمی که در این پژوهش بررسی می‌کنیم در دو دسته‌ی کلی مسائل رگرسیون و پردازش تصویر تقسیم‌بندی می‌شوند.

کلمات کلیدی: شبکه‌های عصبی ـ پردازش تصویر ـ رگرسیون ـ Python - Tensorflow



شکل ۱: Multilayer Perceptron

فهرست مطالب

مقدمه

مسائلی که امروزه و در دنیای صنعت طرح و بررسی می‌شوند عموماً مسائل بزرگ و پیچیده‌ای هستند که در پیاده‌سازی عملیاتی به بخش‌های کوچک‌تر تقسیم می‌شوند و توسط تیم اجرایی به افراد دخیل در پروژه تخصیص داده می‌شوند. این مسائل کوچک‌تری که افراد با آن‌ها درگیر هستند عموماً از پایه‌های ریاضیاتی برخوردارند و دانش فنی افراد کمک حال آن‌ها در پیاده‌سازی این مسائل است. تفاوت مواجه با مسائل در دنیای کار و فضای آکادمیک از موضوعاتی‌ست که آگاهی درست نسبت به آن کمک حال افراد در تصمیم‌گیری‌های فردی می‌تواند باشد. پروژه‌ای که در این درس توسز استاد مربوطه طرح و ارائه شد با خوانشی که من از پروژه داشته‌ام تا حد بسیار خوبی پنجره‌ای بود به فضای کار صنعتی و شکل واقعی پیاده‌سازی‌هایی که در دنیای واقع به وقوع می‌پیوندد. دلیل این گفته را هم اینطور بیان می‌کنم که در این پروژه با کتاب‌خانه‌های مشهور حوزه‌ی یادگیری ماشین که در صنعت کاربرد دارند آشنا شدیم، از الگوریتم‌هایی که شاید جزییات دقیق آن‌ها را ندانیم استفاده کردیم. این در حالی‌ست که در فضای آکادمیک کمتر به این شیوه‌ی جعبه‌ی سیاه به مسائل نگاه می‌کردیم و باید جزییات دقیق آن را تحلیل و بررسی می‌کردیم. این نوع نگاه صنعتی مزایا و معایب خود را به همراه دارد. مهم‌ترین مزیت این شیوه بالا بردن توانایی افراد در بررسی و یافتن خروجی بهینه است بی آنکه درگیر جزییات تئوری شوند. این توانایی رسیدن به خروجی مقبول با آگاهی از عدم دانایی کامل نسبت به برخی بخش‌ها از مهارت‌ها بسیار باارزش است که در پیاده‌سازی پروژه این درس آن را تجربه کردم. اما شاید اندکی دور شدن از جزییات دقیق تئوری و ترجیح دریافت خروجی مورد انتظار به پیاده‌سازی دقیق و گام‌به‌گام، گاهی فهم عمیق مسائل را به فراموشی بسپارد و ارزش این فهم از یاد برود. نباید فراموش کرد که تمامی این مسائل وقتی به بهترین نحو قابل استفاده هستند و در مسائل مختلف قابل تطبیق می‌شوند که آن فهم عمیق تئوری در پس آن وجود داشته باشد.

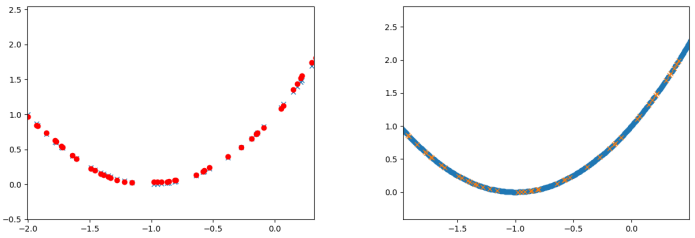
پروژه‌ای که گزارش آن در پیش روی شماست از گام‌های حساب‌شده و دقیقی پیروی می‌کند. درگام نخست با پیاده‌سازی رگرسیون توسط شبکه‌های عصبی آشنا می‌شویم. این که چطور بتوانیم یک تابع را تخمین بزنیم از اولین مسائلی‌ست که در حوزه یادگیری ماشین با آن برخورد خواهیم داشت. در گام دوم و سوم با مفهوم نویز و شکل کارکرد آن در داده‌ها آشنا می‌شویم و داده‌هایی با ابعاد بالاتر را بررسی می‌کنیم. سپس وارد بخش فنی‌تر پروژه می‌شویم. در بخش‌های ۵ و ۶ با پایگاه

داده MNIST که یک پایگاه داده تصویری در موضوع اعداد دست‌نوشته است آشنا می‌شویم. آموزش و آزمایش شبکه های پیچیده‌تر را بر روی این دیتاست اجرا می‌کنیم و در نهایت با استفاده از شبکه‌ی طراحی‌شده به ایجاد یک فیلتر نویز می‌پردازیم. در بخش پایانی نیز با نگاهی اجمالی به گام‌های پیموده شده به بحث و بررسی پیرامون تعیین بهینه‌ی پارامترهای دخیل می‌پردازیم. در نهایت جمع‌بندی پروژه به همراه منابع مورد استفاده نیز ذکر شده‌اند. در اکثر پیاده‌سازی‌ها از منابع معتبر اینترنتی استفاده شده است که به صورت دقیق در جای جای متن ارجاع داده شده است. پیاده‌سازی‌های مورد نظر با توجه به نیازهای پروژه تغییر داده شده است. در صورتی که می‌خواهید مطالب را به صورت جامع‌تر دنبال کنید می‌توانید به مراجع ذکر شده مراجعه کنید.

به همراه این گزارش پروژه صورت دقیق کدهای پیاده‌سازی شده و همچنین دیتاست‌های مورد بحث نیز ارسال شده است. در طراحی گام‌های مختلف سعی شده است به نحوی پیاده‌سازی شود تا اجرا گرفتن کدها هر چه آسان‌تر باشد. با این اوصاف اگر در اجرای گام‌های پروژه ابهام یا ایرادی وجود داشت می‌توانید از طریق آدرس الکترونیکی Aliiiqbp@gmail.com مشکل پیش آمده را با بنده درمیان بگذارید تا بتوانم مشکل احتمالی را رفع کنم. به امید آنکه از مطالعه‌ی این گزارش بهره و لذت کافی را ببرید.

۱ بخش اول

در گام اول می‌خواهیم با مسئله‌ی تخمین تابع کار را آغاز کنیم. برای پیاده‌سازی باید یک شبکه‌ی MLP [۱] طراحی کنیم. سپس مجموعه‌ی نقاطی را با استفاده از تابع ارائه شده تولید و آن‌ها را به دو دسته‌ی داده‌های آموزش [۲] و آزمایش [۳] تقسیم کنیم. داده‌های آموزش را به شبکه‌ی طراحی‌شده بدهیم تا این شبکه رفتار تابع را یاد بگیرد. سپس از نقاط دسته‌ی دوم به عنوان معیاری برای سنجش دقت یادگیری شبکه استفاده می‌کنیم. این مسائل که در آن‌ها هدف، یادگیری تابع است مسائل Regression نام دارند. مثالی از خروجی اولیه این شبکه را در شکل ۱ـ۱ مشاهده می‌کنید.



شکل ۱ـ۱: تخمین تابع $y = x^2$lr

بعد از پیاده‌سازی [۱] [۲] این شبکه و طی کردن مراحل فوق مشاهده می‌کنیم که پارامترهای زیادی در حل و بحث این مسئله درگیر هستند. مواردی مانند میزان پیچیدگی تابع اولیه، چگونگی تقسیم نقاط اولیه به دو گروه، تعداد لایه‌های شبکه‌ی طراحی‌شده و ... . در ادامه می‌خواهیم با تغییر پارامترهای موجود در مسئله، به بحث پیرامون تاثیر این مقادیر بر بهینگی حل مسئله بپردازیم. برای سنجش کارکرد پیاده‌سازی از سه معیار مختلف استفاده می‌کنیم. نخست زمان اجرای برنامه، سپس مقدار خطای تخمین و در نهایت میزان دقت شبکه که با استفاده از تابع score شبکه ارائه شده است. حال به بررسی این پارامترها در بخش‌های جداگانه می‌پردازیم.

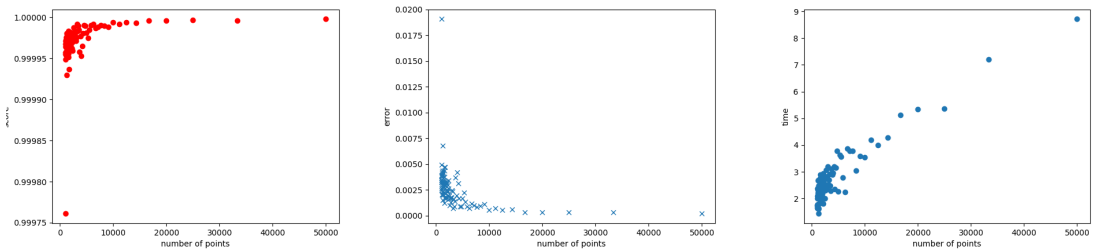
[1] multilayer perceptron
[2] train
[3] test

۱ـ۱ نقاط آموزش و آزمایش

نقاط ورودی مجموعه‌ای از اعداد هستند که بر روی آن‌ها آموزش و آزمایش را انجام می‌دهیم. این که در یک دامنه مشخص چه مقدار نقطه برای این کار انتخاب کنیم و به چه نسبتی این نقاط را به دو گروه آموزش و آزمایش تقسیم کنیم پارامتری موثر در خروجی برنامه است.

۱ـ۱ـ۱ تعداد نقاط

در ابتدا به بررسی تاثیر تعداد نقاط ورودی برای یادگیری و سنجش شبکه در حالی که سایر پارامترها ثابت هستند می‌پردازیم. به شکل ۱ـ۲ دقت کنید.



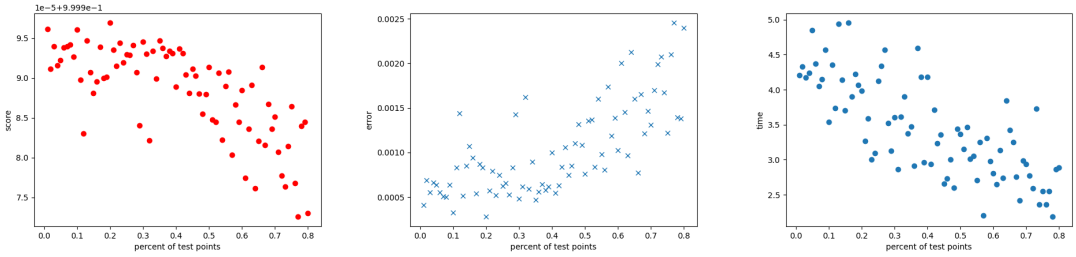
شکل ۱ـ۲: بررسی روند تغییرات معیارهای ارزیابی بر حسب تعداد نقاط ورودی

به وضوح می‌توان دید که با افزایش تعداد نقاط انتخابی به منظور آموزش و آزمایش شبکه زمان و دقت اجرا به صورت نمایی افزایش و خطای محاسبه نیز به همان صورت نمایی کاهش می‌یابد.

۱ـ۱ـ۲ تقسیم نقاط

حال در این بخش تاثیر نسبت تقسیم نقاط به دو گروه آموزش و آزمایش را بررسی می‌کنیم. به این شکل که تعداد ۱۰۰۰۰ نقطه را برای این شبکه آماده می‌کنیم. به ترتیب از ٪۱ الی ٪۵۰ نقاط را به دو دسته‌ی آزمایش و آموزش تقسیم می‌کنیم. تاثیر این نسبت تقسیم را در معیارهای خروجی را در شکل ۱ـ۳ مشاهده می‌کنید.

با توجه به نمودارها می‌توان مشاهده کرد که افزایش درصد نقاط آزمایش منجر به کاهش درصد نقاط آموزش می‌شود که در نتیجه‌ی این تغییر، دقت و زمان اجرا کاهش و خطای تخمین افزایش می‌یابد.

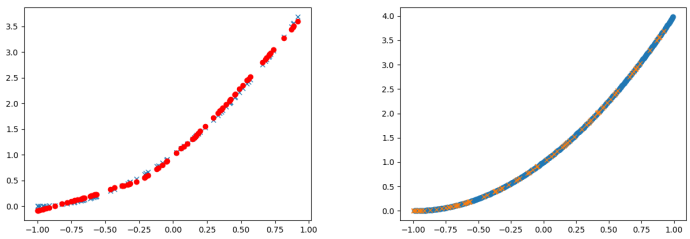


شکل ۱ـ۳: بررسی روند تغییرات معیارهای ارزیابی بر حسب درصد تخصیص نقاط به گروه آزمایش.

۱ـ۲ پیچیدگی تابع

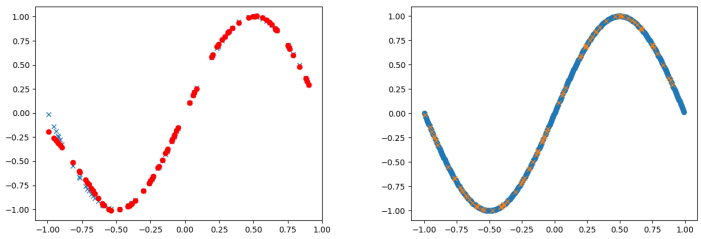
پارامتر بعدی پیچیدگی تابع مورد بحث است. بدین شکل که توابع خطی و چندجمله‌ای را در مراتب پایین‌تر سختی و توابع سینوسی و به طور کلی توابعی که نقاط انهنای زیادی دارند را در مراتب بالاتر سختی در نظر می‌گیریم. با استفاده از چند تابع به بررسی تاثیر این پیچیدگی بر معیارهای خروجی می‌پردازیم.

$$y_۱(x) = x^۲ + ۲x + ۱ \qquad (۱ـ۱)$$



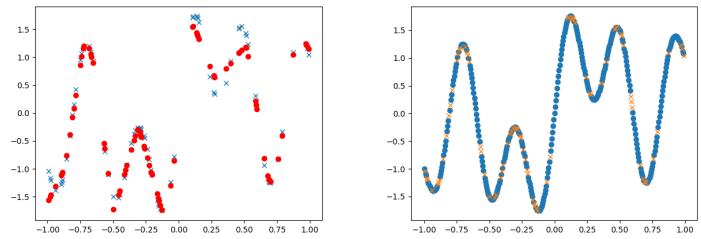
شکل ۱ـ۴: تخمین تابع شماره یک

$$y_۲(x) = sin(\pi x) \qquad (۱ـ۲)$$



شکل ۱ـ۵: تخمین تابع شماره دو

$$y_۱(x) = sin(۲\pi x) + sin(۵\pi x) + x \qquad (۱ـ۳)$$



شکل ۱ـ۶: تخمین تابع شماره سه

حال که تصاویر تخمین‌های توابع فوق توسط شبکه‌ی عصبی چندلایه را مشاهده کردید، به میانگین معیارهای خروجی برای هر یک از توابع فوق نیز نگاهی بیندازید. اعداد جدول زیر حاصل اجرای پیاده‌سازی برای هر یک از توابع ارائه شده فوق و گرفتن میانگین آن‌ها می‌باشد.

$function$	Time	Error	Score
y_1	1.94994	0.00040	0.99997
y_2	2.58053	0.00021	0.99956
y_3	12.50928	0.21474	0.94358

همانطور که در جدول فوق مشاهده می‌کنید، پیچیده‌تر شدن تابع ورودی به شکل چشم‌گیری در زمان اجرا و میزان خطای محاسبه شده [۴] تاثیر دارد.

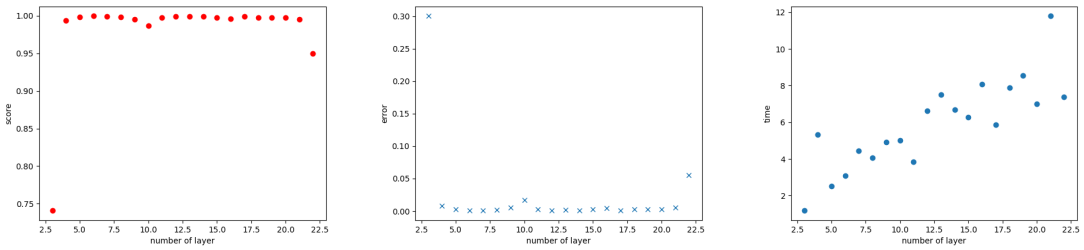
[4] mean squared error

۱ـ۳ پارامترهای شبکه

تا به این جا به بررسی پارامترهایی پرداختیم که از شبکه مستقل بودند و به داده‌ها و شیوه‌ی تقسیم‌بندی و تولید آن‌ها مربوط بودند. در این بخش به بررسی ویژگی‌های خود شبکه‌ی عصبی چندلایه مانند تعداد لایه‌های شبکه، تعداد نورون‌های هر لایه و تعداد چرخش شبکه برای یادگیری می‌پردازیم.

۱ـ۳ـ۱ تعداد لایه

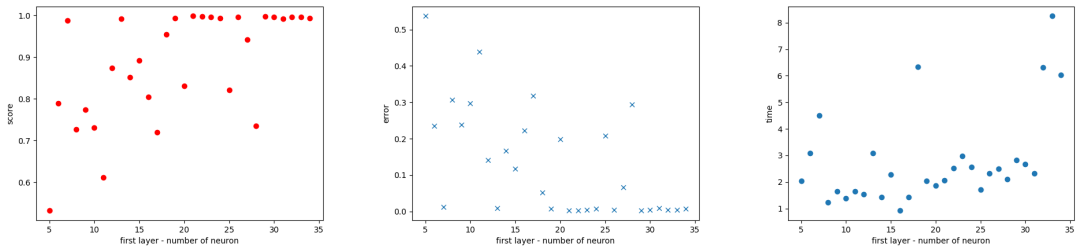
حداقل تعداد لایه برای تشکیل شبکه‌ی multilayer perceptron ۳ لایه است که یک لایه‌ی پنهان دارد. در این بخش با آزمودن تعداد لایه‌های مختلف مشاهده می‌کنیم به طور عمومی با افزایش این لایه‌ها تا حدی کارکرد شبکه بهبود پیدا می‌کند و از جایی به بعد منجر به اضافه شدن سربار محاسباتی می‌شود که مطلوب نیست. بنابرای برای هر تابعی و با توجه به تعداد نقاط و سایر پارامترها، مرزی وجود دارد که در آن بازه از لحاظ تعداد لایه‌های شبکه بهترین مقدار است و با کاهش یا افزایش این تعداد کارکرد شبکه به نحوی کاهش پیدا می‌کند. این کاهش کارکرد می‌تواند در افزایش زمان محاسبه باشد یا منجر به بیش‌برازش نیز شود. در مقیاس محاسبات با cpu خیلی نتوانستم اجراهایی با ابعاد بالا پیاده‌سازی کنم و فلذا شاید به خوبی در نمودارها این مهم نمایش داده نشود. ۱ـ۷



شکل ۱ـ۷: افزایش تعداد لایه‌ی شبکه

۱ـ۳ـ۲ تعداد نورون

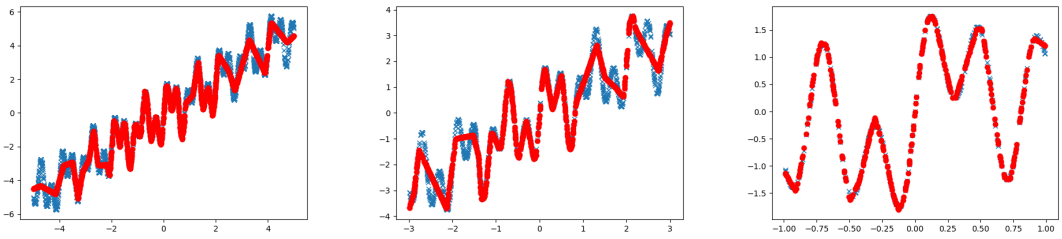
برای بررسی تعداد تاثیر تعداد نورن‌ها بر کارایی شبکه، یک تابع را در نظر گرفتم و با ثابت نگاه داشتن سایر پارامترها و افزایش تعداد نورون هر لایه به بررسی پارامترهای خروجی می‌پردازیم. شبکه در ابتدا به شکل لایه‌هایی با تعداد به ترتیب ۵، ۱۰، ۱۰ و ۵ نورون بود. در هر مرحله تعداد نورون‌ها تمام لایه‌ها را یک واحد افزایش داده و نمودار پارامترهای خروجی بر حسب تعداد نورون لایه‌ی اول را رسم کردم. تصاویر این نمودارها را در تصویر ۱ـ۸ مشاهده می‌کنید.



شکل ۱ـ۸: افزایش تعداد نورون‌های لایه‌ها

۱ـ۴ وسعت دامنه

مورد نهایی که در این بخش بررسی می‌شود وسعت دامنه‌ی داده‌ها، مخصوصاً در توابع پیچیده‌تر است. با بزرگ شدن دامنه کارکرد شبکه به شکل چشم‌گیری کاهش پیدا می‌کند به نحوی که عملاً در نقاط دور از مرکز بازه کارایی خود را از دست می‌دهد. این خطا گاهی آنقدر فاحش می‌شود که دیگر نمی‌توان از این شبکه به عنوان تخمین‌زننده آن تابع نام برد. چندین خروجی برای دامنه‌های مختلف از کم تا زیاد را در شکل ۱ـ۱ مشاهده می‌کنید.



شکل ۱ـ۹: افزایش تعداد نورون‌های لایه‌ها

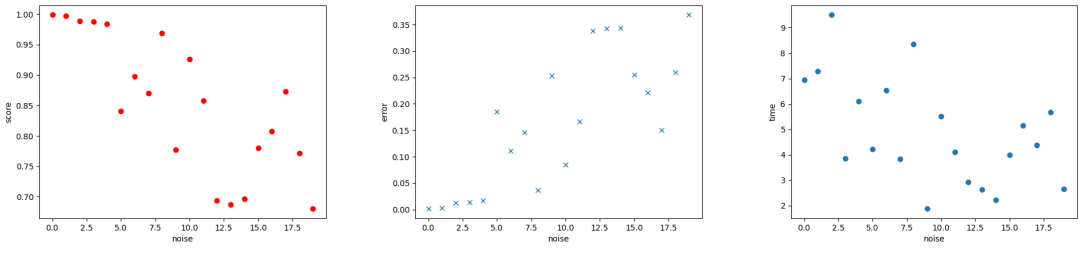
در جدول زیر رابطه‌ی بین پارامترهای سنجش شبکه و ابعاد دامنه را مشاهده می‌کنید. شما هم مثل من از تغییرات گسترده‌ی این پارامترها به نسبت تغییر اندک دامنه شگفت‌زده شده‌اید؟ [۵]

Domain	Time	Error	Score
$[-1, 1]$	7.25970	0.00206	0.99820
$[-3, 3]$	16.33532	0.23520	0.93837
$[-5, 5]$	66.48407	0.36147	0.94825

[۵] باید شگفت‌زده شوید :)

۲ بخش دوم

در این بخش به بررسی اثر نویز بر کارکرد شبکه می‌پردازیم. برای این کار با افزودن نویزی که به صورت نرمال در بین داده‌ها پخش شده است و تغییر پارامتر شدت نویز، کارکرد شبکه را بررسی می‌کنیم [۱] .



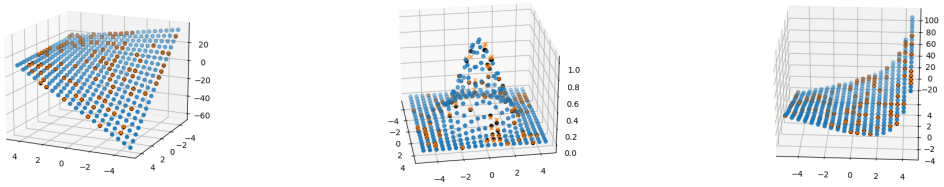
شکل ۲ ـ ۱: تاثیر افزایش نویز بر پارامترهای خروجی

همانطور که در نمودارها فوق مشخص است تاثیر نویز بر زمان اجرا چندان قابل ذکر نیست. اما همین عامل تاثیر عیانی بر خطای تخمین شبکه می‌گذارد که با افزایش این نویز، مقدار خطا افزایش می‌یابد. البته موضوعی که قابل ذکر است این است که درصورتی که تعداد نقاط زیاد باشد و تابع مورد نظر چندان پیچیده نباشد، تا حد اندکی نویز می‌تواند گاهی به بهبود کارکرد شبکه کمک کند. اما اگر بعنوان برآیند، بخواهیم موضوع نویز را جمع‌بندی کنیم این‌گونه است که رفع نویز در داده‌های ورودی شبکه در مجموع می تواند به بهبود کارکرد شبکه کمک کند و در دقت این شبکه اثری چشم‌گیر دارد.

[1] https://stackoverflow.com/questions/46385212/adding-noise-to-numpy-array

۳ بخش سوم

در گام سوم می‌خواهیم به همان شکل که در بخش‌های پیشین تابعی را تخمین زدیم، خروجی یک تابع را یاد بگیریم. با این تفاوت که در این بخش تابع مولد نقاط آموزش و آزمایش از مرتبه‌ی متغییری بالاتر از ۲ است. بنابراین با پیاده‌سازی و تخمین توابع چندمتغییره سر و کار داریم [۱] . در شکل ۳ـ۱ یک مثال از صفحه‌ای در فضای سه‌بعدی مشاهده می کنید [۳] .



شکل ۳ـ۱: نمودار تخمین توابع چندمتغییره

به خوبی مشاهده کردید که چگونه شبکه‌ی مورد استفاده در بخش‌های پیشین کارایی بنسبت قابل قبولی نیز در این بخش داشت. البته موردی که در اجراهای مختلف به آن برخوردم افزایش زمان اجرا در توابع چندمتغییره و افزایش نسبی خطا بود که با توجه به افزایش تعداد متغییرها کاملا قابل پیش‌بینی بود.

۳ـ۱ بررسی پارامترها

در بخش یک به طور مفصل راجع‌به تاثیر پارامترها صحبت کردیم و روند تغییرات به طور عمده مشابه یکدیگر هست در توابع یک یا چندمتغییره و شدت تغییر متفاوت است. بنابراین در اینجا خیلی از مباحث را بازگویی نمی‌کنیم و به ذکر یک

مثال و گزارش عددی بسنده می‌کنیم.

۳ـ۱ـ۱ تعداد متغییرها

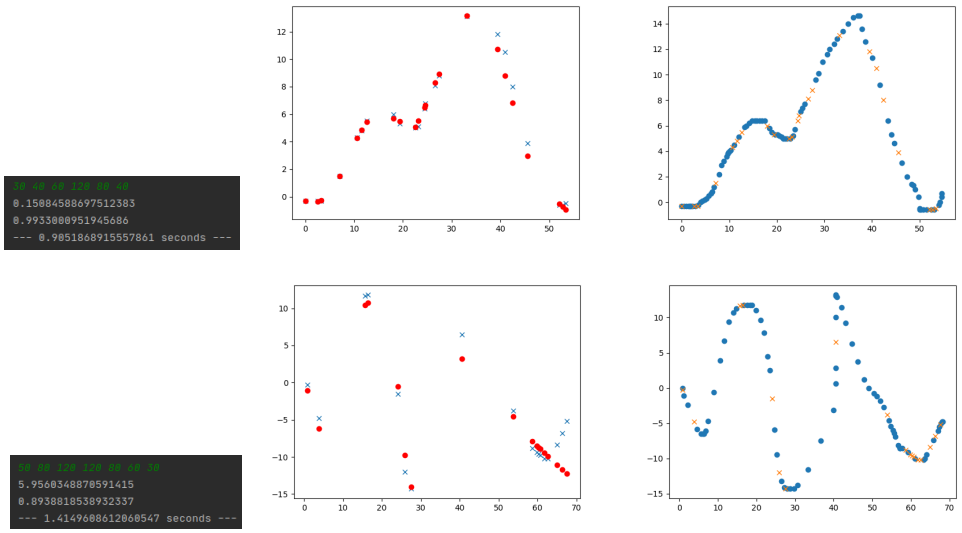
با افزایش تعداد نقاط در یک بازه‌ی ثابت و با ثابت نگاه داشتن سایر پارامترها مشاهده می‌کنیم که زمان اجرا و خطای تخمین به شکل چشم‌گیری افزایش می یابد و سرعت رشد این تغییرات در مقایسه با توابع یک متغییره شدیدتر است. در جدول زیر زمان اجرا، خطای تخمین و امتیاز شبکه در یادگیری تابع را مشاهده می‌کنید. روند سریع تغییرات پارامترهای ارزیابی شبکه در برابر تغییرات اندک تعداد نقاط ورودی را مشاهده می‌کنید.

$\#pints$	Time	Error	Score
10^2	1.59467	0.94604	0.99565
10^4	6.43488	0.20553	0.99942
10^6	22.20664	0.01399	0.99996

همچنین روند تغییرات پارامترهای ارزیابی با تغییر نسبت داده‌های آموزش و آزمایش نیز به شکل مشابه تغییر می‌کند.

۴ بخش چهارم

در این بخش هدف پیاده‌سازی شبکه‌ای است که بتواند تابعی نامشخص را تخمین بزند. تفاوت این بخش با بخش‌های پیشین در این است که نقاطی که از یک فرمول سرراست پیروی نمی‌کنند و شکلی ناهمگون دارند دارای نقاط پرشیب هستند و تشخیص رفتار آن‌ها در نقاط مختلف به دلیل عدم پیروی این نقاط از یک قاعده‌ی مشخص سخت‌تر از بخش‌های پیشین است [۱] . در ادامه در تصویر ۴ـ۱ برخی از این اجراها را مشاهده می‌کنید.



شکل ۴ـ۱: تخمین مجموعه نقاطی که از یک تابع مشخص پیروی نمی‌کنند.

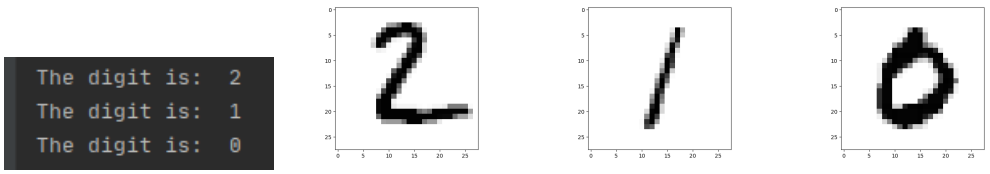
همانطور که در تصاویر مشاهده کردید، تخمین نقاطی که از تابع مشخصی پیروی نمی‌کنند پیچیده‌تر از حالت عادی است. در این موارد هم زمان آموزش شبکه طولانی‌تر است. هم آنکه خطای بدست آمده بیشتر از موارد عادی است. برای آنکه کارکرد شبکه را بهتر کنیم هم می‌توانیم تعداد لایه‌ها و نورون‌ها را افزایش دهیم، هم آنکه از بهینه‌ساز مناسب در شبکه بهره برداریم.

[1] https://stackoverflow.com/questions/48665201/keras-regression-to-approximate-function-goal-loss-1e-7

۵ بخش پنجم

در بخش پنجم به دسته‌بندی داده‌ها می‌پردازیم. برای این کار به دو دلیل از دیتاست MNIST استفاده کردیم [۴] [۵] . نخست آن که این پایگاه داده دارای پیاده‌سازی‌های متعدد و فراوان در اینترنت هست که با جست‌وجو به راحتی می توان به نحوه‌ی کار با این پایگاه داده پی برد. دوم آنکه پایگاه داده تصویری همیشه برای شخص من جذابیتی فرای سایر انواع داده داشته، زیرا نتایج حاصله و تست و ارزیابی را می‌توانم عیناً مشاهده کنم و این ویژگی برایم بسیار لذت‌بخش است. در ابتدا یک اجرای ساده با تعداد داده‌های آموزش و آزمایش پایین انجام می‌دهیم. خروجی این کار را در تصویر ۵ـ۱ مشاهده می‌کنید. سپس به بررسی پارامترهای دخیل در این موضوع و چگونگی تاثیر آن‌ها بر کارکرد شبکه می‌پردازیم.

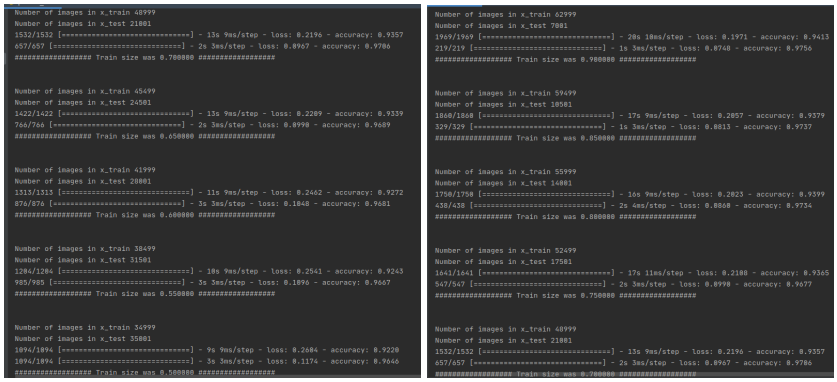
در مورد این پایگاه داده لازم است بدانید که دارای ۶۰،۰۰۰ داده‌ی تصویری به منظور آموزش و ۱۰،۰۰۰ داده برای آزمایش است. این داده‌ها از دست‌نوشته‌های دانش‌آموزان و کارمندان جمع‌آوری شده است. برای کار با این پایگاه داده که در کتاب‌خانه‌ی TensorFlow نیز موجود است، ابتدا آن را لود کرده و سپس به تخصیص داده‌های آموزش و آزمایش به متغییرهای مورد نظر می‌پردازیم. در گام‌های بعدی [۶] به آماده‌سازی این داده‌ها از نظر ابعاد و شکل جهت ورود به شبکه می‌پردازیم. در گام بعدی شبکه را طراحی و داده‌های آموزش را به آن می‌دهیم. در نهایت نیز این داده‌ها را مورد تست و آزمایش قرار می‌دهیم و نتایج را تحت عنوان دقت خروجی گزارش می کنیم. برای درک بهتر نیز چندین مورد آزمایش تصویری نیز انجام می‌دهیم و با مقادیر برچسب‌ها که در پایگاه داده است مقایسه می‌کنیم.



شکل ۵ـ۱: چندین تخمین بر روی دیتاست MNIST

۵ـ۱ نسبت داده‌ها

در بخش نخست بررسی دیتاست MNIST به مشاهده‌ی تاثیر نسبت داده‌های آموزش و آزمایش بر دقت تخمین می‌پردازیم. به این صورت که تعداد کل داده‌ها ۷۰،۰۰۰ تا است. نسبت داده‌های آموزش را از ۹۰ درصد تا ۵۰ درصد تغییر می‌دهیم تا روند تغییرات درصد دقت تخمین را مشاهده کنیم [۱] .



شکل ۵ـ۲: نتایج تغییر نسبت داده‌های آموزش و آزمایش بر دقت خروجی

همانطور که در خروجی‌های تصویر بالا مشاهده می‌کنید به طور عمومی دقت شبکه با افزایش نسبت داده‌های آموزش افزایش می‌یابد. البته این موارد و روند تغییرات را در بخش اول به تفصیل بحث و بررسی کردیم و در این بخش به ذکر مورد و نمایش نتایج اجراها بسنده می‌کنیم.

۵ـ۲ Optimizer

در شبکه‌های عصبی و برای ساخت و آموزش این شبکه‌های یک پاراامتر به نام optimizer وجود دارد. این پارامتر در حقیقت الگوریتمی است که شبکه با آن روش وزن یال‌های خود را به مرور تعیین می‌کند و به عبارتی شبکه آموزش داده می‌شود. بسته به نوع پیاده‌سازی شبکه و مهم‌تر از آن نوع داده‌های مورد بررسی، بهینه‌سازهای مختلف، کارایی‌های متفاوتی خواهند داشت. بدین ترتیب برای هر موضوعی، انتخاب بهینه‌سازهای مناسب جهت رسیدن به دقت دلخواه، از گام‌های اساسی است. در این بخش چندین بهینه‌ساز معروف را تست و ارزیابی می‌کنیم و مشاهده می‌کنیم این بهینه‌سازها بر روی دیتاست مورد نظر چقدر کارایی دارند [۷] .

[1] https://stackoverflow.com/questions/54316779/change-size-of-train-and-test-set-from-mnist-dataset

```
########## adam Optimizer ##########

1750/1750 [==============================] - 14s 8ms/step - loss: 0.2116 - accuracy: 0.9352
438/438 [==============================] - 1s 3ms/step - loss: 0.0905 - accuracy: 0.9744

########## Adagrad Optimizer ##########

1750/1750 [==============================] - 14s 8ms/step - loss: 1.1146 - accuracy: 0.7217
438/438 [==============================] - 1s 3ms/step - loss: 0.5135 - accuracy: 0.8710

########## sgd Optimizer ##########

1750/1750 [==============================] - 14s 8ms/step - loss: 0.5427 - accuracy: 0.8472
438/438 [==============================] - 1s 3ms/step - loss: 0.2544 - accuracy: 0.9248

########## AdaDelta Optimizer ##########

1750/1750 [==============================] - 15s 9ms/step - loss: 2.2009 - accuracy: 0.2999
438/438 [==============================] - 1s 3ms/step - loss: 2.0733 - accuracy: 0.5656
```

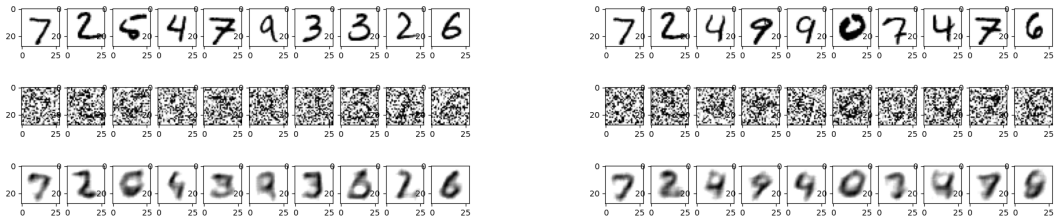
شکل ۵ـ۳: دقت خروجی با استفاده از بهینه‌سازهای مختلف

با مشاهده‌ی نتایج خروجی‌های حاصل از انواع بهینه‌ساز در شکل ۵ـ۳ به خوبی متوجه می‌شویم که نقش بهینه‌ساز تا چه اندازه مهم است و بر روی دقت خروجی اثر می گذارد. در این مثال دقت خروجی از حدود ۵۵ الی ۹۷ درصد متغییر بود که بازه‌ی بسیار بزرگی است.

برای انتخاب صحیح بهینه‌ساز باید به دو مورد توجه کرد. نخست ساختار شبکه‌ی طراحی‌شده و دیگری نوع داده‌های ورودی و هدف از پیاده‌سازی است. با جست‌وجوی مناسب می‌توان متناسب با نیاز خود بهینه‌ساز قابل قبول را پیدا و از آن استفاده کرد.

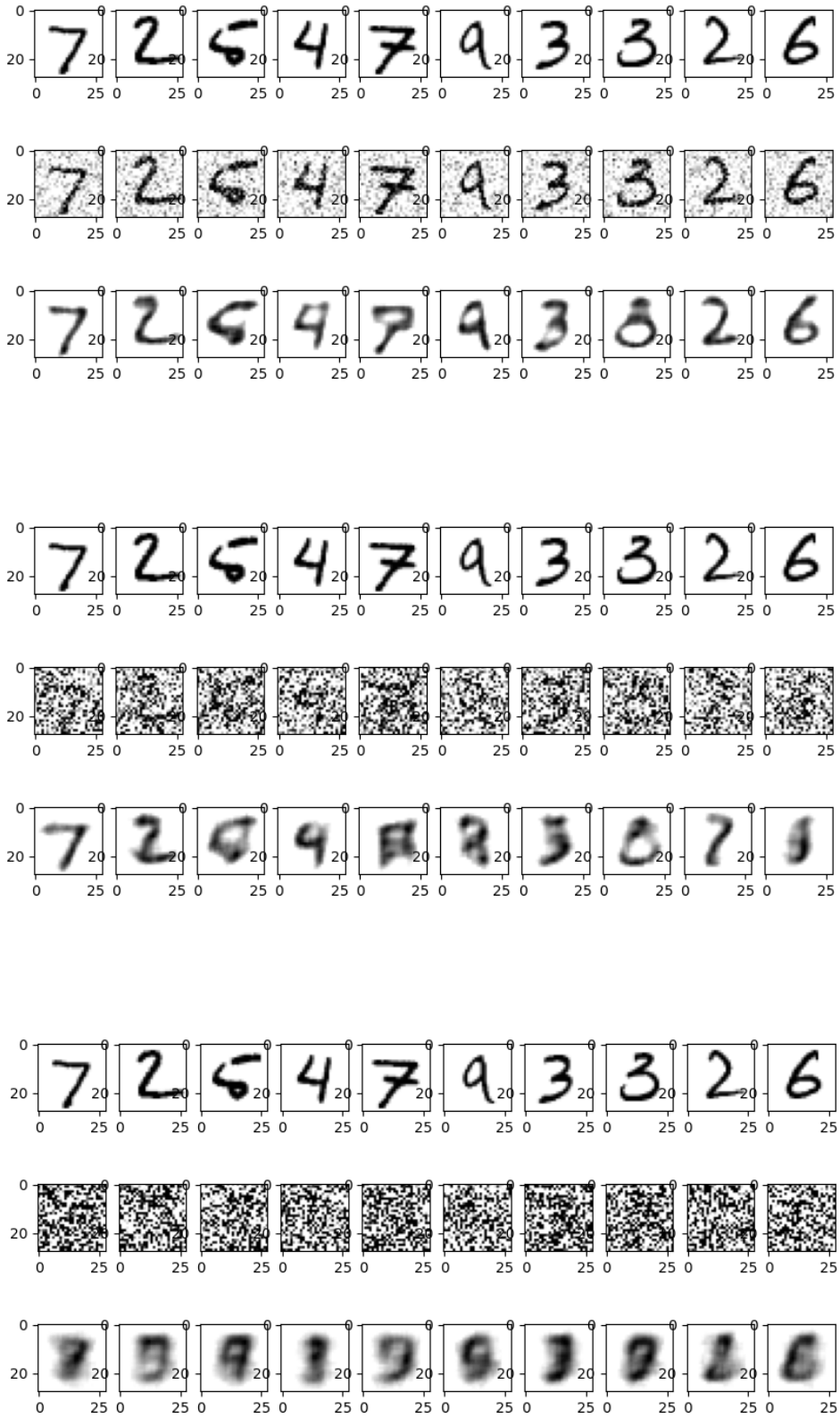
۶ بخش ششم

در این بخش می‌خواهیم به کمک شبکه‌ی طراحی شده در بخش پیشین از تصاویر نویز را رفع کنیم. حال به چه شکل این کار را انجام دهیم؟ به این صورت که تعدادی از تصاویر را انتخاب کرده و به آن‌ها نویز اضافه می‌کنیم. سپس این عکس‌های دارای نویز را به ورودی شبکه می‌دهیم. خروجی مورد انتظار هر ورودی را عکس‌های بدون نویز متناظر قرار می دهیم. بدین ترتیب شبکه یاد می‌گیرد که چگونه از یک تصویر با نویز به تصویر بدون نویز برسد. حال که شبکه آموزش دید، ورودی‌های دیگری به همراه نویز به شبکه می‌دهیم و می‌بینیم که تصویر خروجی را چگونه فیلتر می‌کند. بدین ترتیب با استفاده از شبکه‌ی عصبی یک فیلتر رفع نویز ساخته‌ایم. برخی از اجراها را در تصاویر زیر مشاهده می‌کنید.



شکل ۶ـ۱: فیلتر نویز ساخته‌شده با استفاده از شبکه‌های عصبی چندلایه

همانطور که در تصاویر بالا مشاهده می‌کنید این تصاویر تا حد خوبی از نویز خالی شدند. البته ساختار شبکه را چندان تغییر ندادم و از همان شبکه‌ی بخش ۵ استفاده کردم و صرفاً ابعاد ورودی و خروجی را کنترل کردم. با این حال می‌توان با استفاده از بهینه‌سازهای مختلف به نتایج بهتری دست یافت. برای ارزیابی بصری کارایی شبکه در ادامه نویز را از میزان اندک تا میزان زیاد تغییر می‌دهیم و کارایی شبکه را مشاهده می‌کنیم.



شکل ۶ـ۲: کارایی شبکه در نویزها به صورت افزایشی

۷ بخش هفتم

نوعی از شبکه‌های عصبی شبکه‌های تکاملی [۱] هستند. این شبکه‌ها به این دلیل که با فیلترهای مختلف اشکال متفاوت را تشخیص می‌دهند و از کنار هم قرار دادن این مجموعه‌ها به درک درستی از تصویر می‌رسند دقت بسیار بالایی دارند. همچنین با توجه به نیاز اندک به پیش‌پردازش و آنکه به مثابه مغز انسان کارکرد دارند برای داده‌های حجیم و پیچیده کارایی بهتری دارند. بنابراین برای داده‌های بزرگ این نوع شبکه‌های عصبی مناسب هستند.

از طرفی در طول پیاده‌سازی و بررسی اجراهای متعددی که گرفته شد موضوعی مهم به خوبی برای من نمایان شد. این مهم که پارامترهای به نسبت زیادی در هر پیاده‌سازی، اجرا و تست شبکه‌های عصبی وجود دارند. مقادیر این پارامترها با توجه به نوع هر مسئله و ابعاد داده‌های ورودی قابل تغییر است و یافتن مقادیر بهینه برای این پارامترها موضوع بسیار مهمی است که می‌تواند به طرز چشم‌گیری در دقت شبکه و زمان اجرای آن تاثیر داشته باشند. همان طور که در بخش ۵ مشاهده کردیم، با تغییر الگوریتم بهینه‌ساز، یا در بخش ۴ با تغییر تعداد لایه‌های نهان و نورون‌ها، دقت شبکه چیزی در حدود ۵۰ درصد جابجا می‌شود.

بدین ترتیب می‌توان تعیین بهینه‌ی پارامترهای شبکه‌های عصبی را مهم‌ترین گام در آموزش و ارزیابی مدل ارائه شده دانست. چاره چیست؟ ایده‌ای که می‌توان آن را بررسی کرد استفاده از الگوریتم‌های تکاملی است. بدین شکل که تمامی پارامترهای دخیل در پیاده‌سازی را به مثابه یک کروموزم در نظر گرفته که هر پارامتر یک ژن از ژنوم‌های آن است. حال با اجراهای متعدد و استفاده از تکنیک‌های موجود در الگوریتم ژنتیک به پاسخی بهینه برای تعیین پارامترها دست یافت.

همچنین ایده‌ی دیگری نیز که وجود دارد استفاده از الگوریتم‌های جست‌وجوی آگاهانه است. برای مثال الگوریتم‌های ذوب فلزات یا جست‌وجوی محلی نیز می‌تواند در این موضوع کارا باشند. برای ارزیابی نقاط نیز می‌توان یک یا چند خروجی زمان اجرا یا دقت تخمین را مد نظر قرار داد و بهینگی هر نقطه را با یک یا چندین مورد از این ارزیاب‌ها سنجید.

اما کماکان معظل اجراهای پیاپی و زمان اجرا ممکن است باقی مانده باشد. برای این کار دو رویکرد متفاوت وجود دارد. نخست آنکه از محاسبه‌کننده‌های GPU به جای CPU استفاده کرد. این واحدها به طور موازی برنامه‌ها را اجرا می‌کنند و برای

[1] CNN

پردازش‌های سنگین بسیار مناسب هستند. پردازش‌هایی از قبیل کارهای گرافیکی یا پیاده‌سازی‌های پیچیده‌ی نرم‌افزاری. دومین مورد هم آن که از نمونه‌هایی کوچکتر از دیتاست اصلی استفاده کنیم و برداشتی رندوم از آن‌ها را به منظور یافتن پارامترهای بهینه به کار ببندیم. در هر مرحله که به پارامترهای بهینه نزدیک شدیم، مقدار دیتاست دخیل در محاسبات را افزایش دهیم تا در نهایت با مجموعه‌ی کامل، آموزش و آزمایش را به اتمام برسانیم.

۸ جمع‌بندی

شبکه‌های عصبی، ابزاری قدرتمند در حوزه‌ی یادگیری ماشین هستند. درک درست از کارکرد برداری ماتریس‌ها و هم‌چنین دانش آمار و احتمال می‌تواند کمک حال ما در فهم عمیق‌تر این عنوان باشد. آنچه در کارکرد عملیاتی و نه آموزشی آکادمیک صورت می‌گیرد شامل دو رویه مهم است. نخست انتخاب مدل مناسب برای حل مسئله‌ی پیش رو. سپس یافتن پارامترهای مناسب که در هر مدلی دخیل هستند. این دو گام اساسی، چارچوب کلی پیاده‌سازی‌های شبکه‌های عصبی را تشکیل می‌دهد. در بخش ۵ دیدیم که انتخاب درست بهینه‌ساز تا چه میزان بر کارکرد شبکه تاثیر می‌گذارد. در بخش‌های ابتدایی به خوبی تاثیر هر یک از پارامترها بر کارکرد سیستم را دیدیم. این تاثیر گاهی به حدی چشم‌گیر است که می‌تواند یک مسئله‌ی قابل حل در زمان خطی را به یک مسئله‌ی لاینحل تبدیل کند. در بخش پایانی ایده‌هایی برای انتخاب درست پارامترهای دخیل در شبکه صورت گرفت. موارد ذکر شده بدیهتاً تمامی موارد موجود نبوده و نخواهد بود.

در سال‌های اخیر (موکداً دهه‌ی اخیر) روند دانش هوش مصنوعی با سرعت چشم‌گیری پیشرفت داشته. این مهم نه صرفا در حوزه‌ی دانشگاهی که در ساختار صنعت نیز مشهود بوده است. از بودجه‌های کلان آکادمیک در دانشگاه‌های بزرگ جهان و پوزیشن‌ها متعدد موجود در این فیلد، این موضوع را می‌توان دریافت.

شکل ۸ــ۱: آیا ربات‌ها قدرت جهان را به دست خواهند گرفت :)

سپاس

از استاد بزرگوار، جناب آقای دکتر آرش عبدی هجران‌دوست که با کمک‌ها و راهنمایی‌های بی‌دریغشان، تعریف این پروژه و تدریس این واحد درسی بنده را در انجام این مهم یاری داده‌اند، تشکر و قدردانی می‌کنم.

مراجع

[1] J. Henschel. *Approximate Function With Neural Network*, 2019 March 27 (accessed August 1, 2020). https://blog.cubieserver.de/2019/approximate-function-with-neural-network/.

[2] J. Brownlee. *Neural Networks are Function Approximation Algorithms*, 2020 March 18 (accessed July 28, 2020). https://machinelearningmastery.com/neural-networks-are-function-approximators/.

[3] A. Murphy. *Matplotlib 3D Plot – A Helpful Illustrated Guide*, 2020 (accessed July 28, 2020). https://blog.finxter.com/matplotlib-3d-plot/.

[4] L. M. Z. S. Z. Y. L. R. F. Wan. *Neural Network*, 2013 (accessed July 28, 2020). https://www.python-course.eu/neural_network_mnist.php.

[5] O. G. Yalçın. *Image Classification in 10 Minutes with MNIST Dataset*, 2018 August 20 (accessed August 8, 2020). https://towardsdatascience.com/image-classification-in-10-minutes-with-mnist-dataset-54c35b77a38d.

[6] J. Brownlee. *How to Develop a CNN for MNIST Handwritten Digit Classification*, 2019 May 8 (accessed July 31, 2020). https://machinelearningmastery.com/how-to-develop-a-convolutional-neural-network-from-scratch-for-mnist-handwritten-digit-c

[7] Chengwei. *Quick Notes on How to choose Optimizer In Keras*, 2018 (accessed August 8, 2020). https://www.dlology.com/blog/quick-notes-on-how-to-choose-optimizer-in-keras/.