به نام خدا

یادگیری عمیق و بهینه سازی محدب: جستاری به بهینه سازی شبکه‌های عصبی عمیق و بررسی دلایل قدرت آنها با وجود محدب نبودن

پروژه درس بهینه سازی در علوم داده

علیرضا گرگوری مطلق - ۹۸۱۰۲۱۷۶

سروش آتشی - ۹۸۱۰۴۶۰۳

استاد:

دکتر مجتبی تفاق

SHARIF
UNIVERSITY OF
TECHNOLOGY

نیم سال دوم تحصیلی ۱۴۰۱-۱۴۰۰

فهرست

مقدمه:

یادگیری عمیق شاخه ای جدیدتر و به قولی پخته تر از یادگیری ماشین است که از لایه های متعددی از الگوریتم ها برای پردازش داده استفاده میکند. لایه های یادگیری عمیق در واقع با الهام گیری از روند تفکر انسان ها در مغز به وجود آمده و در سال های اخیر انقلابی در هوش مصنوعی و جهان اطراف ما به وجود آورده اند.

تاریخچه معرفی یادگیری عمیق در اصل به سال هایی به دوری ۱۹۴۳ بازمیگردد اما این الگوریتم ها تا حدود سال ۲۰۱۰ کاربرد چندان چشم گیری از خود نشان نداده بودند و در حوزه هوش مصنوعی حساب چندانی روی آنها باز نمیشد. در دهه ۲۰۱۰ اما با پیشرفت تکنولوژی و افزایش سرعت محاسبات و به کاربردن GPUها در این زمینه، دانشمندان این حوزه قادر شدند با بهره گیری از لایه های عمیق کانوولوشنال به دقت اعجاب انگیزی در تشخیص و کلاس بندی عکس ها برسند و توانستند برای اولین بار قدرت واقعی این شبکه ها را به جهانیان ثابت کنند. به طور دقیق تر، شبکه کانوولوشنال AlexNet در سال ۲۰۱۲ توانست به دقت ۸۴.۷٪ بر روی دیتاست ImageNet دست پیدا کند که تا آن زمان کاملا بی سابقه بود و الگوریتم های دیگر حتی نتوانسته بودند به عددی نزدیک به این دقت برسند.

از آن به بعد یادگیری عمیق همواره مارا قادر ساخته تا عکس ها را با دقت بسیار بالایی دسته بندی کنیم، چهره ها را شناسایی کنیم، فایل های صوتی را به نوشته تبدیل کنیم و هزاران هزار کار شگفت انگیز و پیچیده دیگر را به سادگی و سرعت بسیار زیاد انجام دهیم. امروزه هوش مصنوعی کاملا بر یادگیری عمیق متکی است. بانک ها و شرکت های مالی امروزه میتوانند با تکیه بر یادگیری عمیق، معاملات بورسی و خرید سهام را به صورت اتوماتیک انجام دهند، ریسک ها را به حداقل برسانند و کلاه برداری های عظیم را به سادگی و به صورت اتوماتیک تشخیص داده و از آنها جلوگیری کنند. یادگیری عمیق را به نوعی میتوان انقلاب چهارم صنعتی دانست. تکنولوژی که به طور کامل زندگی مارا در دنیای امروز تحت تاثیر قرار داده و در آینده حتی میتواند نقش مهم تری نه تنها در صنعت و شرکت های عظیم، بلکه در زندگی روزمره ما داشته باشد.

شبکه های عصبی عمیق را با وجود ساختار بسیار غیر خطی و غیر محدبشان، میتوان به سادگی با استفاده از الگوریتم های درجه اولی مانند gradient descent آموزش داد. با این حال، ما هنوز کاملا مطمئن نیستیم این الگوریتم ها دقیقا به چه صورتی میتوانند با این دقت شبکه های عمیق را آموزش داده و به مدل هایی دست یابند که به راحتی در تمامی شرایط و روی داده های از پیش دیده نشده به ما جواب مطلوبی بدهند.

همانطور که پیش تر گفته شد، حل تابع هزینه ی شبکه های عصبی به هیچ وجه یک مسئله محدب نیست و تابع هزینه میتواند میلیون ها نقطه مینیمم محلی داشته باشد که کار ما را برای پیدا کردن مینیمم کلی (global minimum) بسیار دشوار میکنند. همچنان ما میدانیم که حل کردن مسئله های بهینه سازی غیر محدب میتوانند بسیار دشوار باشند و حتی در اکثر مواقع حل تحلیلی نداشته باشند. پس شبکه های عصبی چگونه توانسته اند به چنین پیشرفت و دقت خیره کننده ای دست پیدا کنند؟ شاید جواب این پرسش over parametrization باشد! در ادامه بیشتر به این مسئله میپردازیم. در نهایت نیز کاربرد مستقیمی از بهینه سازی محدب را در حالت‌هایی خاص برای شبکه‌های عصبی بیان میداریم.

Over-Parameterization و اثر آن بر شبکه های عصبی عمیق:

شبکه‌های عصبی عمیق غالبا با شمار زیادی از پارامترهای شبکه ایجاد میشوند که معمولا این تعداد از تعداد نمونه‌های آموزشی نیز بیشتر می‌باشد. با این وجود، این مدل‌های Over-Parameterize شده در خیلی از موارد دچار بیش‌برازش (overfitting) نشده و به خوبی بر روی داده‌های جدید تعمیم داده می شوند(generalization). این موضوع در دیدگاه نخست و با دانش ما از یادگیری ماشین کلاسیک کمی عجیب بنظر میرسد و *"چرا شبکه‌های عصبی Over-Parameterize شده دچار بیش برازش نمیشوند؟"* منجر به شکل گیری سوالی اساسی راجع به عملکرد و موفقیت شبکه‌های عصبی عمیق میشود.

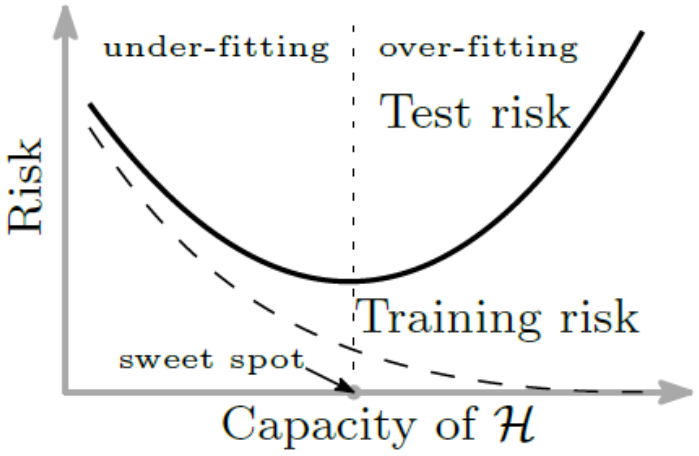
نظریه مرسوم یادگیری ماشین و تناقض برای شبکه‌های عصبی عمیق

نظریه مرسوم ماشین لرنینگ بیان میدارد که در انتخاب مدلی از خانواده‌ای از توابع $\mathcal{H}$ با ظرفیت N، ظرفیتی برای مدل برحسب tradeoff بین بایاس و واریانس انتخاب شود که تعادلی میان کم برازش(underfitting) و بیش برازش (overfitting) برقرار شود و این ظرفیت نقش کنترل بایاس-واریانس را داشته باشد(۱):

۱. اگر $\mathcal{H}$ بسیار مقدماتی باشد (مقدار N کوچک باشد)، تمام پیش‌بینی‌های H احتمالا به داده‌های آموزش underfit شده (به بیان دیگر ریسک تجربی (Empirical Risk) بالایی دارند) و در نتیجه بر روی داده های تست و جدید پیش بینی ضعیفی دارند. (بایاس زیاد، واریانس کم)

۲. از طرفی اگر $\mathcal{H}$ بسیار پیچیده باشد (مقدار N بزرگ باشد)، به حداقل رساندن ریسک تجربی ممکن است به الگوهای جعلی یا نویز در داده‌های آموزش بیش از حد برازش داشته باشد و منجر به دقت کم بر روی داده‌های تست و جدید شود. (بایاس کم، واریانس زیاد)

بنابراین این دیدگاه کلاسیک توجه خود را به یافتن نقطه‌ای بهینه (sweet spot) میان underfitting و overfitting معطوف میدارد. انتخاب ظرفیت مدل میتواند به صورت صریح (برای مثال انتخاب نوع مدل و ساختار آن در شبکه‌های عصبی) یا ضمنی (برای مثال استفاده ازعملکرد توقف زودهنگام در آموزش مدل) باشد. هنگامی که تعادلی مناسب بین این دو پدیده رخ داد، بیان میشود که مدل h_N به خوبی بر روی داده‌های تست و تمام جمعیت P تعمیم داده میشود. صحبت‌های بالا در منحنی متداول U شکل خطای تست خلاصه می شود که در نمودار ۱ آورده شده است و از این نمودار بارها به طور گسترده ای به عنوان معیاری جهت انتخاب مدل استفاده شده است و حتی تصور می شود جنبه هایی از نحوه تصمیم‌گیری انسان را توصیف کند(۲).



نمودار ۱ – U-Shape Risk Curve

به طور دقیقتر، دیدگاه مرسوم ماشین لرنینگ عنوان میکند که در فضای فرضیه(hypothesis space) مدل‌های ما با نام H، و دارای m نمونه آموزشی .i.i.d، فاصله یا شکاف بین خطای تجربی (Empirical error) و خطای تعمیم دادن (Generalization error) معمولا کران پایینی دارد و اغلب با $O(\sqrt{|H|/m})$ محدوده شده است که $|H|$ پیچیدگی فضای فرضیه مدل میباشد. برای مثال با درنظر گرفتن تمام فضای فرضیه‌ها در شبکه‌های عصبی عمیق، $|H|$ متناظر عمق × عرض مدل میباشد (تعداد لایه‌ها × تعداد نورون‌های هر لایه)؛ بنابراین در

شبکه‌های عصبی عمیق مطابق این دیدگاه انتظار داریم که صورت کسر در کران ذکرشده افزایش پیدا کرده و در نتیجه خطای عمومیت دهی بیشتری داشته باشیم(۳).

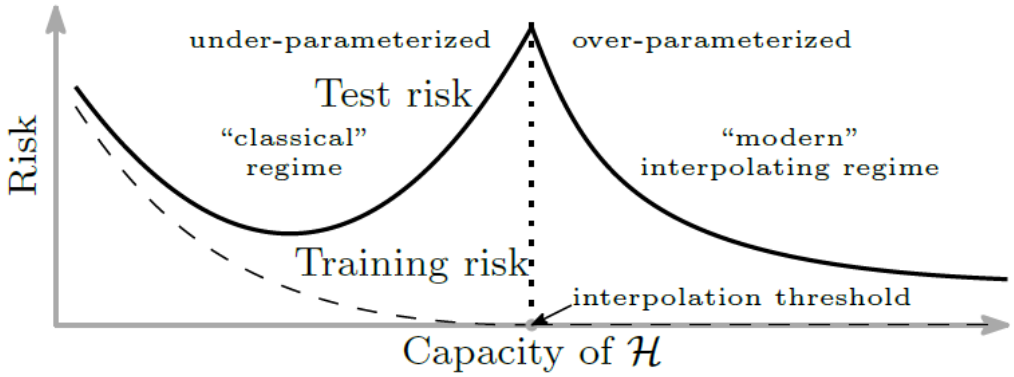
با وجود دلایل بالا اما، در کاربرد به طور معمول از روش‌های یادگیری ماشینی مدرن مانند شبکه‌های عصبی عمیق و سایر پیش بینی کننده های غیر خطی استفاده می‌شود که خطای آموزش بسیار کم یا صفر دارند. با وجود ظرفیت بالای این کلاس توابع و تناسب تقریباً کامل با داده های آموزشی، این پیش بینی کننده ها اغلب جهت پیش بینی داده های جدید نیز بسیار دقیق هستند. ما روزانه شبکه‌های عصبی بزرگ و عمیقی را میبینیم که دارای میلیون‌ها یا میلیاردها پارامتر می‌باشند اما بر روی تعداد به مراتب کمتری از نمونه‌ها آموزش داده میشوند که به خوبی بر داده‌های تست عمومیت داده میشوند.

ما دارای شواهد تجربی بسیار و در عین حال شواهد تئوری کمی میباشیم که این مدل‌های Over-Parameterize شده به طور معمول به خوبی بهینه سازی شده و با وجود نامحدب بودن تابع هزینه آنها و در نتیجه وجود کمینه‌های محلی و نقاط زینی برای آنها، عملکرد بسیار خوبی داشته و کاربردی ترین و پیشرفته ترین مدل های یادگیری ماشین محسوب میشوند.

پیش از ادامه بحث برای رفع ابهام و مشخص شدن موضوع اصلی به تعریف over-parameterization میپردازیم:

over-parameterization به حالتی اطلاق میشود که در آن تعداد پارامترهای مدل بیشتر از تعداد نمونه‌های آموزشی آن است.

در این مقاله هدف اصلی این کار، الگویی است که نشان می‌دهد چگونه عملکرد داده‌های دیده نشده به ظرفیت مدل بستگی دارد و مکانیسم و دلایل زیربنای به وجود آمدن آن را تبیین میکند. این نوع وابستگی، که به صورت تجربی برای کلاس‌های مهمی از مدلها از جمله شبکه‌های عصبی و بر روی دیتاست های متعددی مشاهده شده است باعث بوجود آمدن نمودار جدیدی تحت عنوان "Double Descent Curve" می شود که میتواند در بسیاری از مدلها جایگزین نمودار کلاسیک "U-Shape" شود. این منحنی جدید در نمودار ۲ نمایش داده شده است (۴):



نمودار ۲ – Double Descent Curve

در ابتدا به بررسی وجود این نمودار در مدل‌های مشخصی میپردازیم و پیاده سازی خودمان را در یکی از این حالت ها همراه با نتایج آن بیان میداریم؛ سپس ادامه بحث را به بررسی دلایل بوجود آمدن این نمودار و عملکرد مناسب شبکه‌های عصبی عمیق با وجود محدب نبودن تابع هزینه آن ها اختصاص میدهیم:

منحنی Double Descent؛ نموداری جدید برای انتخاب و بررسی مدل‌ها (۴)

در این بخش ما درباره Double Descent Curve در زمینه شبکه‌های عصبی صحبت میکنیم و پیاده سازی الگوریتم مقاله را انجام میدهیم:

ویژگی‌های فوریه رندوم (Random Fourier Features)

در ابتدا ما یک مدل معروف از کلاس مدل‌های پارامتریک غیرخطی به نام random fourier features(RFF) را که میتوان آنها را به صورت یک شبکه عصبی دو لایه که وزن های لایه اول در آن‌ها ثابت است نشان داد را در نظر میگیریم. خانواده مدل های RFF با $\mathcal{H}_N$ پارامتر شامل تابع های $h: \mathbb{R}^d \to \mathbb{C}$ به فرم

$$h(x) = \sum_{k=1}^{N} a_k \emptyset(x; v_k)$$

که در آن $\emptyset(x; v_k) = e^{i<v_k, x>}$ است. در این معادله بردارهای $v_1, \dots, v_N$ به صورت مستقل از توزیع نرمال استاندارد در $\mathbb{R}^d$ نمونه‌برداری میشوند.

به جای پیدا کردن ضریب های مختلط $\{a_k\}_{k=1}^{N}$، ما ماتریس دیزاین(design) را به صورت زیر تعریف میکنیم:

$$\mathbf{\Phi} = \begin{bmatrix} \cos(v_1^\top x_1) & \sin(v_1^\top x_1) & cos(v_2^\top x_1) & \sin(v_2^\top x_1) & \cdots & \cos(v_N^\top x_1) & \sin(v_N^\top x_1) \\ \cos(v_1^\top x_2) & \sin(v_1^\top x_2) & cos(v_2^\top x_2) & \sin(v_2^\top x_2) & \cdots & \cos(v_N^\top x_2) & \sin(v_N^\top x_2) \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ \cos(v_1^\top x_n) & \sin(v_1^\top x_n) & cos(v_2^\top x_n) & \sin(v_2^\top x_n) & \cdots & \cos(v_N^\top x_n) & \sin(v_N^\top x_n) \end{bmatrix}$$

این کار به ما ماتریس $\mathbf{\Phi} \in \mathbb{R}^{n\times 2N}$ را میدهد که در آن طبق مقاله، n تعداد داده‌های آموزش و N تعداد ویژگی رندوم فوریه میباشد. توجه داشته باشیم که به دلیل اینکه هر ویژگی در نحوه نمایش ما شامل یک سینوس و یک کسینوس است، تعداد کل تابع‌های پایه 2N خواهد بود. فرکانس‌های ویژگی‌ها نیز همانطور که (۴) توضیح داده است، باید از یک چگالی نرمال استاندارد سمپل شوند.

به همین صورت ما میتوانیم خروجی هر تابع را که با h(x) نشان میدهیم به صورت زیر با ضرب ماتریسی محاسبه کنیم:

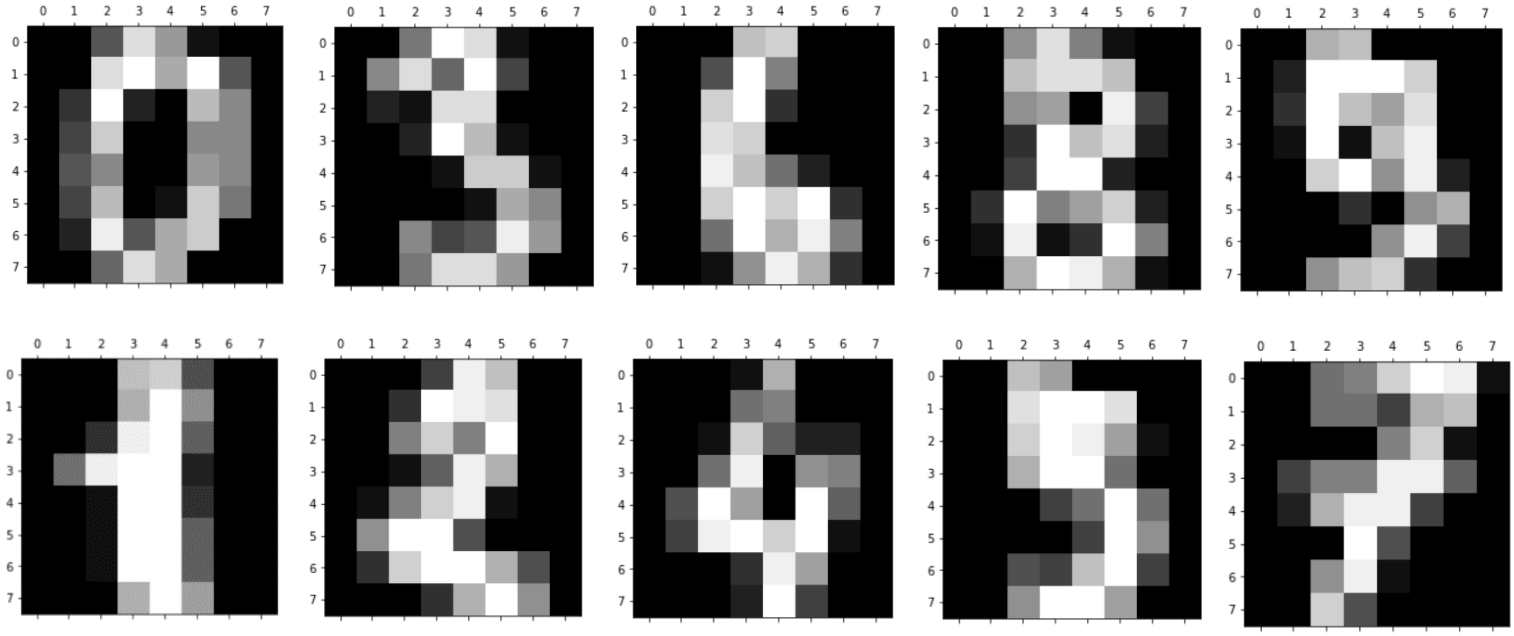
$$h(x) = a^T \mathbf{\Phi}(x)$$

پروسه یادگیری ما با استفاده از $\mathcal{H}_N$ بدین ترتیب خواهد بود: با استفاده از داده های$(x_1, y_1), (x_2, y_2), \dots, (x_N, y_N)$ که دارای بعد $\mathbb{R}^d \times \mathbb{R}$ هستند، ما $h_{n,N} \in H_N$ را با استفاده از ERM و مربع خطاها به دست می آوریم. این به این معناست که که ما تابع ریسک تجربی (Empirical Risk) هدف $\sum_{i=1}^{N}(h(x_i) - y_i)^2$ را روی تمامی توابع $h \in \mathcal{H}_N$ کمینه میکنیم.

هنگامی که مینیمایزر یکتا نیست(که در حالت $2N > n$ همواره برقرار است) ما مینیمایزری را انتخاب میکنیم که ضرایب $(a_1, a_2, \dots, a_N)$ آن دارای نرم l_2 کمینه باشند.

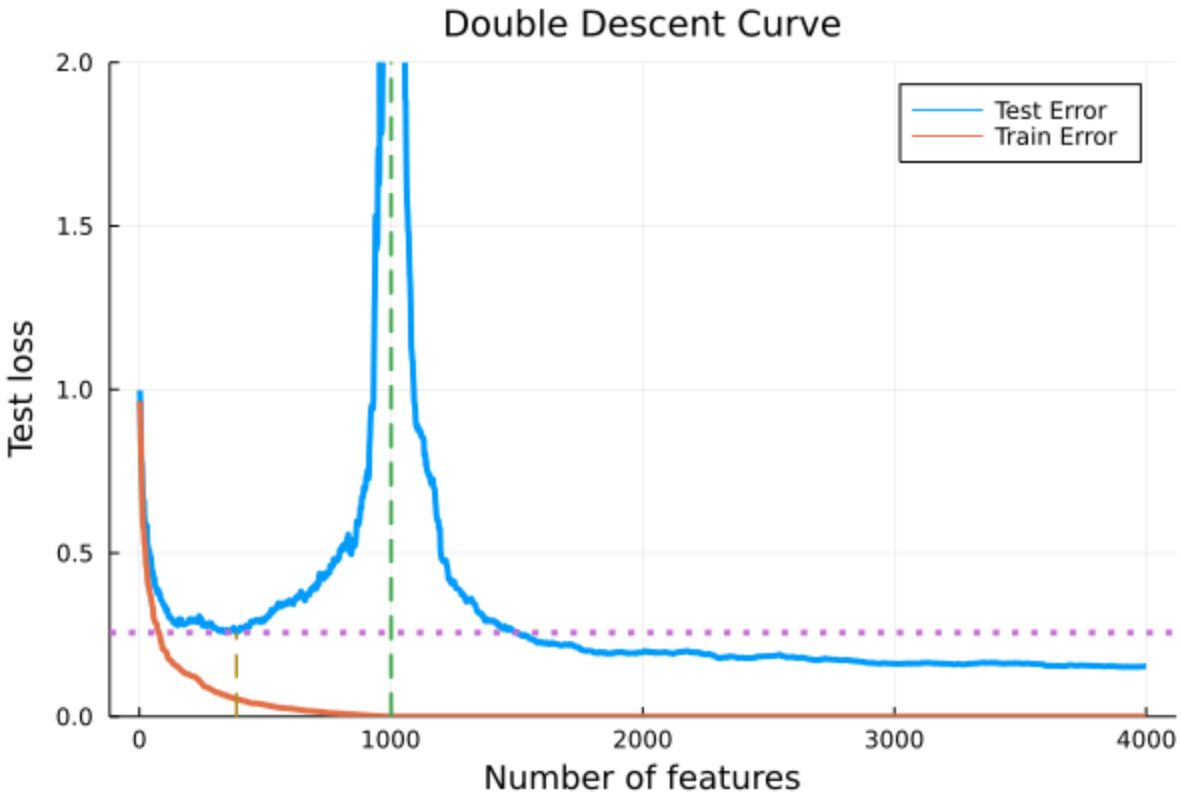
همانطور که پیشتر اشاره شد، ما داریم ریسک تجربی را مینیمایز میکنیم. بنابراین جواب مدل، پاسخ کمترین مربعات برای $2N \leq n$ خواهد بود. همچنین برای $2N > n$ که در دسته مدل‌های Over-Parameterize قرار میگیرند و جواب آن یکتا نخواهد بود، ما جوابی را در نظر میگیریم که نرم l_2 آن کمترین مقدار ممکن باشد که در این صورت نیز به جواب یکتایی خواهیم رسید. در جولیا با استفاده از دستور بک اسلش "/"، تمام شرایط ذکر شده به صورت خودکار لحاظ میشوند و ما همواره به جواب یکتایی میرسیم.

در این پیاده سازی ما از دیتاست MNIST استفاده میکنیم. این دیتاست کمی با دیتاست اصلی تفاوت داشته و شامل ۱۷۹۷ سمپل با اندازه 8×8 میباشد. دیتاست اصلی دارای ۱۰ رقم است. ما با توجه به محدودیت توانایی محاسباتی و کوچک بودن بعد ویژگی‌ها هدف را از تشخیص دادن ۱۰ کلاس عدد، به تشخیص اعداد زاویه دار از اعدادی که از خم تشکیل شده اند تغییر میدهیم. بنابراین هدف ما دسته بندی اعداد ۰، ۳، ۶، ۸، ۹ از اعداد ۱، ۲، ۴، ۵، ۷ خواهد بود. به عبارتی میتوان مسئله را دارای دو گروه مثبت و منفی در نظر گرفت. در این پیاده سازی از ۱۰۰۰ داده را برای اموزش و ۷۹۷ داده را برای تست استفاده شده است. نمونه‌ای از ورودی‌ها در نمودار ۳ آورده شده است:



نمودار ۳ – داده‌های یادگیری برای دو کلاس خم‌دار (بالا) و زاویه‌دار (پایین)

حال مدل را به ازای تعداد ویژگی‌های مختلفی اجرا میکنیم و میزان خطای آموزش و تست را ذخیره کرده و در نمودار ۴ آن را رسم کرده‌ایم. توضیحات کاملتر از نحوه اجرای کد و فرایند آموزش مدل در فایل نوت‌بوک "$Project\_DoubleDescentCurve.ipynb$" به همراه دیتاست مورد استفاده قرار گرفته در فایل‌های ارسالی موجود است.



نمودار ۴ – میزان خطای مدل بر حسب تعداد پارامترهای آن

همانطور که میبینیم، شکل متداول U-شکل در ناحیه under-parameterized معتبر است. همینطور با رسیدن به آستانه درون‌یابی 2N=n=1000، ارور به صورت تصاعدی زیاد میشود و گویا مدل تمام ورودی ها را حفظ میکند! اما در ناحیه Over-Parameterized شرایط به کل فرق میکنند. در این ناحیه تعداد پارامتر ها از تعداد داده ها بیشتر میشود و ارور همواره با اضافه کردن تعداد پارامتر های بیشتر کاهش پیدا میکند تا جایی که حتی مقدار ارور از ناحیه اولیه نیز کمتر میشود. این دقیقا چیزی است که ما قصد داشتیم نشان دهیم؛ در حالی که اضافه کردن پارامتر های بیشتر در ناحیه under-parametrized به عنوان یک تهدید ظاهر میشود، این کار در ناحیه over-parametrized میتواند یک فرصت طلایی باشد. این کار اما ممکن است بر هزینه محاسباتی ما بیفزاید.

شواهد نشان داده شده میتوانند کاندیدی باشند برای توضیح اینکه چرا شبکه های عصبی عمیق با میلیون ها یا حتی میلیاردها داده میتوانند به این خوبی بر روی داده های از پیش دیده نشده تعمیم یابند و در اکثر مواقع حتی با خطای آموزش صفر بیش‌برازیده(Overfit) نمیشوند.

بررسی دلایل پدیده Double Descent و عدم بیش‌برازش در Modern Interpolation Regime

برای عدم بیش‌برازش شدن مدل‌های Over-Parameterize شده تاکنون تئوری مشخصی توسعه داده نشده است و حدس‌ها و گمان‌های زیادی درباره این موضوع زده میشود؛ اما به طور کلی بحث‌ها در این موضوع را می‌توان به ۲ دلیل اصلی تقسیم‌بندی نمود:

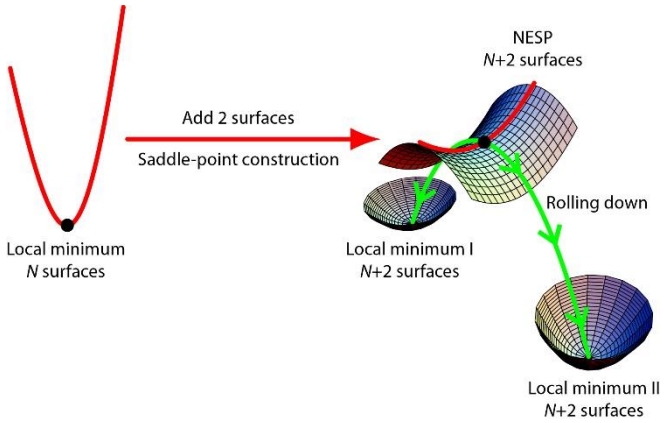
۱. مقداردهی اولیه و نقاط زینی:

میدانیم که روش های بهینه سازی مرتبه اول و دوم هنگام عملکرد بر روی توابع نامحدب به شدت به نقطه آغازین الگوریتم حساس میباشند و نقطه آغازین نامناسب میتواند منجر به رسیدن به کمینه محلی با هزینه زیاد شود و از نقطه بهینه کلی نیز بسیار عملکرد بدتری داشته باشد.

با افزایش ظرفیت مدل در واقع ما فضای فرضیه‌ها (Hypothesis Space) را گسترش میدهیم و این موضوع منجر به افزایش بعد مسئله میشود؛ پس اکنون در بعد بالاتری در حال بررسی مدل هستیم و بسیاری از پارامترها میتوانند اضافی(redundant) باشند.(برای مثال پارامترهای شبکه عصبی به شدت redundant میباشند). حال در این ابعاد بالا میتوان ادعا نمود که الگوریتم‌های بهینه سازی مرتبه اول (first order) همانند SGD(Stochastic Gradient Descent) تاثیرپذیری به مراتب کمتری نسبت به نقطه آغازین دارند؛ در واقع این زائد بودن پارامترها به نوعی فرصت است؛ مقداردهی اولیه نه‌چندان مناسب بعضی پارامترها که میتواند باعث آسیب به مدل شود توسط مقداردهی اولیه مناسب برای سایر پارامترها جبران شود. پس، از این دیدگاه میتوان به عنوان یکی از دلایل احتمالی موفقیت شبکه‌های عصبی عمیق نام برد(۵).

از طرفی دیگر میتوان اثری دیگر را در ابعاد بالا بررسی نمود. با افزایش تعداد پارامترهای مدل و قرار گرفتن در ابعاد بالاتر از بعد متغیرهای ورودی، نقاط کمینه محلی به نقاط زینی تبدیل میشوند (۶)! دلیل این موضوع نیز ساده است: به طور شهودی کمینه محلی نقطه‌ای است که در همسایگی به شعاع ε کمترین مقدار تابع را اتخاذ میکند؛ در حالیکه برای نقطه زینی فقط کمینه بودن در یک جهت کافی است و ممکن است در جهت‌های دیگر تابع مقدار کمتری داشته باشد. پس با افزایش ابعاد فضای تابع هزینه از طریق افزایش پارامترهای مدل، یک کمینه محلی احتمال به مراتب کمتری دارد که وضعیت خود را در ابعاد بالا

نیز حفظ کند؛ زیرا نیاز دارد که در تمامی جهات در فضای گسترش یافته، مینیمم بودن خود در همسایگی ε' را نگه دارد. پس افزایش ابعاد مسئله منجر به تبدیل شدن نقاط کمینه محلی به نقاط زینی میشود. تصویر ۵ باعث شهود بیشتری به این موضوع میشود؛ از طرفی، در صورتی که الگوریتم بهینه سازی مورد استفاده بتواند در نقاط زینی راحتتر خارج شود ما به هدف خود رسیدهایم؛ زیرا احتمال بیشتری برای رسیدن به نقطه بهینه گلوبال یا در مواردی رسیدن به نقطه مینیمم محلی بهتری خواهیم داشت. برای اینکه الگوریتمی به صورت مذکور عمل کند نیاز به بهروزرسانی گرادیان تابع حتی در نقاطی که شیب صفر میباشد خواهیم داشت؛ از جمله این الگوریتمها که به طور فزایندهای در آموزش شبکههای عصبی استفاده میشود SGD میباشد که در مقاله (۷) این موضوع به طور دقیقتر بررسی شده است.



نمودار ۵ – تبدیل شدن نقطه کمینه محلی به نقطه زینی در ابعاد بالا

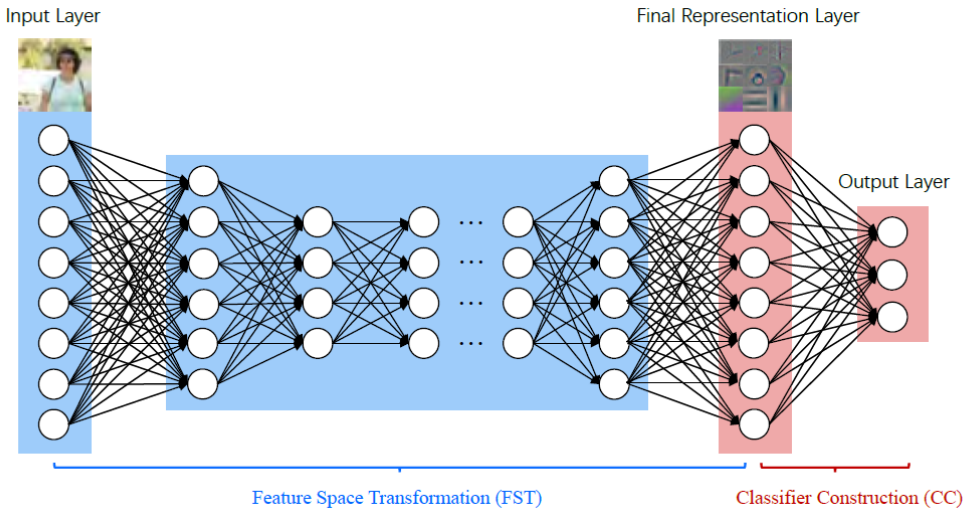
۲. یادگیری بازنمایی (Representation learning) (۳)

تمام نظریههای یادگیری ماشین کلاسیک بر مبنای آموزش یک یادگیرنده، یا به طور خاص، یک طبقه بند در وظایف طبقه بندی، از یک فضای ویژگی میباشد اما توجه کمی در مورد ایجاد و ساخت خود فضای ویژگی میکند. از این رو، نظریه یادگیری مرسوم کلاسیک می تواند مورد استفاده برای بررسی عمومیتیابی مدلها قرار گیرد اما هنگام اعمال آن برای یادگیری بازنمایی باید با دیدگاه متفاوتی به موضوع پرداخت.

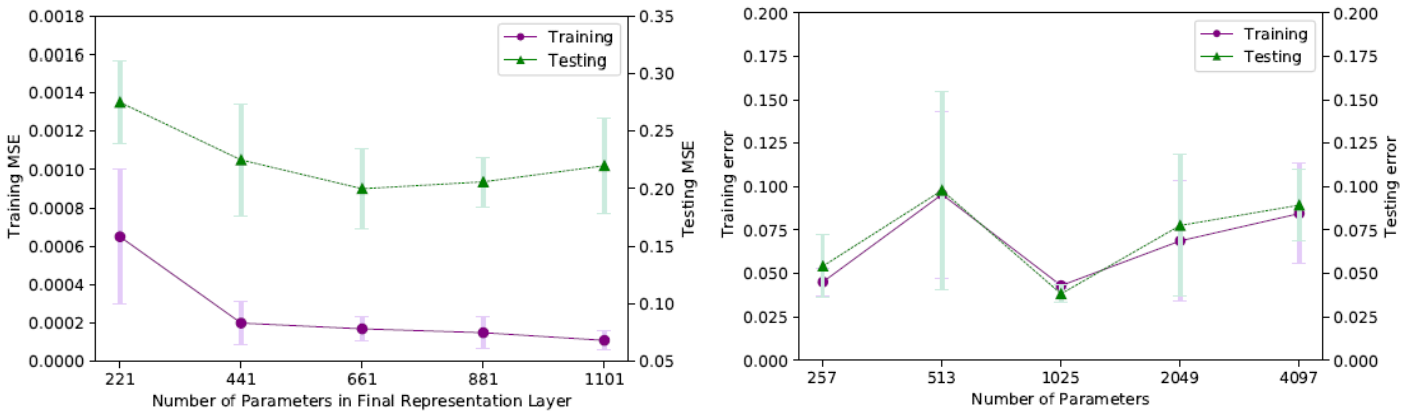
به طور خاص، میدانیم که شبکههای عصبی معروف به ساختن فضای ویژگیها و بدست آوردن ویژگیهای بهینه برای ساخت طبقه بند هستند؛ همانطور که در نمودار ۶-الف رسم شده است، یک شبکه عصبی عمیق را میتوان متشکل از ۲ بخش دانست؛ به طوری که بخش اول آن اختصاص به تبدیل فضای ویژگیها یا به اختصار FST (Feature Space Transformation) دارد که نمونههای خام ورودی را تبدیل به ویژگیهایی مناسب جهت آموزش طبقهبند استخراج میکند؛ بخش دوم این شبکه را نیز میتوان به ایجاد طبقهبند یا به اختصار CC (Classifier Construction) دانست که طبقه بند مناسب را با استفاده از خروجی نهایی FST آموزش میدهد.

برای بررسی این ساختار از بخش CC شروع میکنیم؛ وظیفه این بخش مانند تمام طبقه بند های مرسوم یادگیری ماشین، بدست آوردن مدلی بهینه جهت تشخیص کلاسها میباشد و تعداد پارامترهای آن وابسته به تعداد ویژگی‌های خروجی آخرین لایه FST دارد؛ بنابراین میتوان بیان داشت مطابق نظریه کلاسیک، با افزایش تعداد پارامترهای این بخش باز هم دچار بیش‌برازش خواهیم شد و نمودار متداول U-شکل را خواهیم داشت که در نمودار ۶-ب ترسیم شده است.

حال به بررسی بخش FST در مدل میپردازیم: بیشترین تردید در مورد ناتوانی نظریه یادگیری کلاسیک در بررسی شبکه های عصبی عمیق از این واقعیت ناشی می شود که به نظر می رسد بیش‌برازشی در این ناحیه وجود ندارد حتی زمانی که تعداد پارامترهای شبکه (عرض ×عمق) بسیار بزرگ است.(۸) در واقع ما میخواهیم این موضوع را تبیین کنیم که پارامترهای بخش FST را نمیتوان ساده انگارنه به عنوان پارامترهای فضای ویژگی‌ها که در بیان نظریات یادگیری ماشین مرسوم استفاده میشود درنظر گرفت. در واقع بیان میدارد هنگام بیان این موضوع که Over-Parameterization از دیدگاه تئوری ماشین کلاسیک منجر به بیش برازش میشود، منظور از پارامترها، پارامترهای فضای ویژگیها یا همان بخش CC میباشد در حالیکه این تئوری برای پارامترهای ناحیه FST ادعایی نمیکند. نمونه‌ای از اثر افزایش پارامترهای بخش FST و عدم بیش‌برازش مدل در نمودار ۶-ج آورده شده است. البته باید درنظر گرفت که تاکنون اثر Over-Parameterization بخش FST تاکنون به ندرت مورد مطالعه مستقیم قرار گرفته است و نیازمند به بررسی‌های بیشتر برای پیشبینی اثر این پدیده وجود دارد.



(الف)



(ب) (ج)

نمودار ۶ – بررسی ارتباط یادگیری بازنمایی و Over-Parameterization

ارتباط مستقیم بهینه سازی محدب و شبکه‌های عصبی:

توضیحات بالا نباید خواننده مقاله را به این فکر وادارد که بهینه سازی محدب به طور مستقیم مورد استفاده در شبکه‌های عصبی قرار نمیگیرد؛ برای درک اهمیت این موضوع کافی است به این موضوع توجه کنیم که در کنفرانس NeuroIPS 2019 که یکی از معتبرترین کنفرانس‌های علمی دنیا در زمینه گسترش یادگیری عمیق میباشد، از تمام مقالات مورد قبول واقع شده ۳۲ مقاله به طور مستقیم مرتبط با بهینه سازی محدب بوده‌اند. این موضوع گواه آن است که میتوان امید داشت از تکنیک‌ها و روش‌های بهینه سازی محدب در آینده‌ای نه چندان دور در یادگیری عمیق نیز استفاده نمود.

بهینه سازی محدب و شبکه‌های عصبی ۲ لایه ReLU

با این وجود اکثر کارهایی که تاکنون برای بهینه سازی شبکه های عصبی انجام شده مربوط به شبکه‌های عصبی کم‌عمق (Shallow Neural Networks) می‌باشند که فقط از ۲ یا ۳ لایه تشکیل شده‌اند اما تعداد نورون‌های لایه پنهان بسیار زیاد است (۹و۱۰و۱۱)؛ این روش را میتوان بدین صورت نقد کرد که از طرفی میدانیم شبکه‌های عصبی کم‌عمق طبق قضیه تقریب جهانی (Universal Approximation Theory) (۱۲) با تعداد نورون‌های کافی و استفاده از تابع فعالسازی غیرخطی قابلیت تقریب زدن هر نوع تابعی را دارند؛ اما این موضوع از لحاظ محاسباتی به صرفه نیست و اتصالات فراوان بین دو لایه متوالی منجر به مشکلات محاسباتی میشود. از طرفی

نیز عدم استفاده از لایه‌های بیشتر منجر به عدم یادگیری ویژگی‌های بهینه جهت استخراج آنها در لایه‌های پنهان میشود و در کل این روش‌ها چندان به صرفه نمیباشند. هرچند امید است بتوان با گسترش درک ما از شبکه‌های عمیق نتایج بهینه سازی محدب در شبکه‌های کم‌عمق را به آنها نیز گسترش داد یا با تغییراتی به آنها تعمیم داد.

در ادامه نیز یک کاربرد خاص از بهینه سازی محدب در شبکه های عصبی کانوولوشنال ذکر شده است:

بهینه سازی محدب و CNN (۱۳)

مطالعات نشان میدهند عملکرد شبکه های عصبی عمیق میتوانند به میزان قابل توجهی با استفاده از تکنیک های بهینه سازی محدب بهبود یابد. در ابتدا ما میبینیم که زیر-مدل های به دست آمده با استفاده از تکنیک drop out را میتوان به صورت بهینه با مدلسازی به صورت یک مسئله محدب با هم ترکیب کرد. ما این ایده را برای مدل هایی که با drop out به دست نیامده اند نیز تعمیم میدهیم. در مقایسه با متدهای مرسوم، ما به بهبود ۰.۲۲٪ و ۰.۷۶٪ داده تست بر روی دیتاست CIFAR10 دست می یابیم.

شبکه های عصبی کانوولشنال (CNN) نوع خاصی از شبکه های عصبی عمیق هستند که به طور خاص در شاخه بینایی کامپیوتر استفاده میشوند. این نوع شبکه ها توانسته اند در سال های اخیر به عملکر خارق العاده ای در مسئله های بسیار چالش بر انگیز بینایی کامپیوتر دست پیدا کنند که این موضوع توجه عظیمی را به این نوع خاص از شبکه های عصبی جلب کرده است.

با وجود موفقیت های عظیم شبکه های عصبی عمیق، همواره انتقاد ها و نگرانی هایی نیز در رابطه با آنها وجود داشته است. یکی از مهم ترین آنها محدب نبودن این شبکه ها و تابع هزینه آنهاست. مسئله این است که مدل های محدب قدرت تعمیم پذیری کافی را ندارند و آموزش دادن مدل های غیر محدب نیز کاری بسیار پیچیده و دشوار است. اما تقریبا هیچ کدام از دقت های چشمگیر چندسال اخیر با بهینه سازی محدب صرف به دست نیامده اند. بنابراین میتوان گفت همواره trade-off بین مدل "آسان برای استفاده" و "قدرتمند" وجود دارد.

پس آیا میتوان از بهینه سازی محدب برای بهبود عملکرد شبکه های عمیق به شدت غیرمحدب بهره گرفت؟ در ادامه به پاسخ این سوال میپردازیم. ما شبکه های CNN را به اجزای کوچک و محلی محدبی تجزیه میکنیم و با استفاده از بهینه سازی محدب عملکرد آنها را بهبود میبخشیم: درواقع با تغییر دادن دو لایه آخر شبکه با استفاده از ترکیب خطی زیر-مدل های متعددی که به دست آورده ایم سعی در بهتر شدن عملکرد شبکه می‌کنیم.

پیش از ادامه دادن لازم است کمی با معماری شبکه های عصبی کانوولوشنال بیشتر آشنا شویم. این نوع از شبکه ها از همبستگی و مجاورت ویژگی های یک تصویر در بلوک های همسایه هم استفاده میکنند و سعی میکنند نوعی الگوی خاص محلی بین نورون های مجاور همدیگر پیدا کنند. برای یک لایه پنهان خاص m، واحد های پنهان آن به یک زیرمجموعه محلی واحدها در m-1 لایه قبلی متصلند. همچنین، هر یک از فیلتر های تنک(sparse) h_i در تمام میدان بصری تکثیر میشود. واحدهای تکثیر شده همگی پارامترهای یکسانی دارند (بردار وزن و بایاس آنها یکسان میباشد).

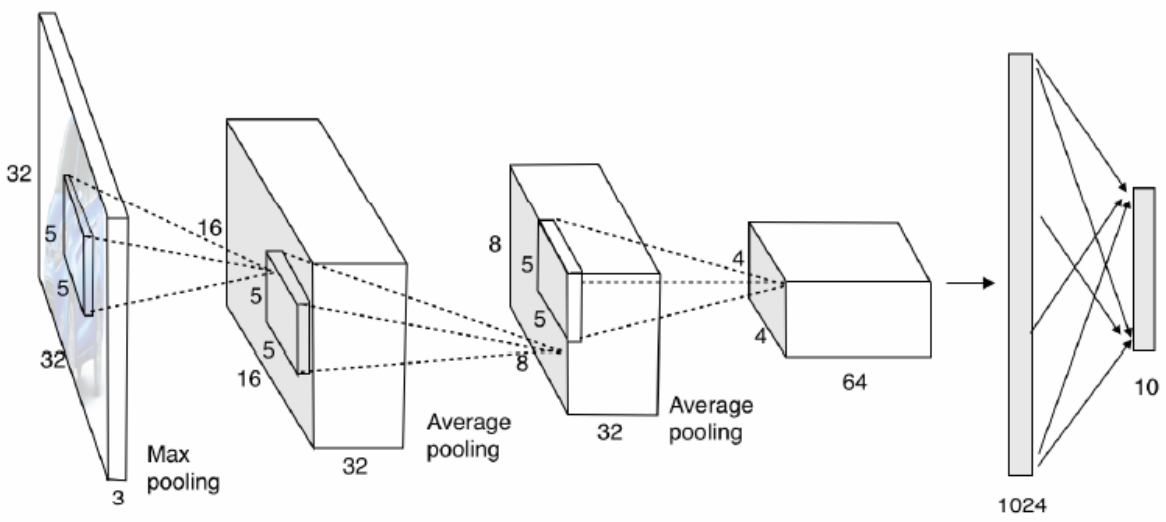
از منظر ریاضی، یک نگاشت ویژگی h^k با کانوالو کردن ورودی با یک فیلتر خطی، اضافه کردن یک ترم بایاس و سپس اعمال یک تابع فعال سازی غیر خطی به دست می آید.

$$h^k_{\{i,j\}} = f\left((W^k * x)_{i,j} + b_k\right)$$

b^k و W^k بایاس و وزن kاُمین نگاشت ویژگی هستند و تابع f، تابع فعال سازی غیرخطی میباشد. در آزمایش های انجام شده این مقاله، از تابع غیرخطی ReLU استفاده شده است. این تابع در استفاده های عملی نتایج بهتری نسبت به تابع متداول tanh داشته است.

نوع مهم دیگری از لایه ها، لایه pooling است. این لایه یک فرم غیر خطی down-sampling است. انواع مختلفی از pooling وجود دارند اما دو مورد از متداول ترین آنها max-pooling و average-pooling هستند. این لایه ها عکس ورودی را به تعدادی مربع متداخل یا جدا از هم تقسیم بندی میکنند و مقدار میانگین یا بیشینه هرکدام ازین مربع ها را خروجی میدهند. استفاده ازین لایه ها میتواند هزینه محاسباتی را برای لایه ها عمیق تر کاهش دهد و در مقابل چرخش یا انتقال نیز مقاوم باشد.

معماری کلی شبکه های CNN در نمودار ۷ نشان داده شده است. معماری زیر برای دیتاست CIFAR10 طراحی شده است. در این شبکه سه لایه کانوولوشنال و pooling به صورت یکی درمیان قرار گرفته اند. همچنین در لایه های pooling مربع ها با یکدیگر تداخل دارند. مربع ها سایز ۳×۳ داشته و با گام ۲ پیش میروند. خروجی این شبکه یک بردار احتمال از ۱۰ کلاس مختلف خواهد بود که درایه با بیشترین مقدار، کلاس عکس ورودی را مشخص میکند.



نمودار ۷ – معماری شبکه CNN مورد استفاده در مقاله

شبکه کانوولوشنال با استفاده از زیر-مدل ها

این بخش را با معرفی تکنیک موثری به نام dropout آغاز میکنیم. با الهام گرفتن از این تکنیک، ما مفهوم زیر-مدل را معرفی کرده و به بررسی بهبود شبکه ها با ترکیب کردن این زیر-مدل ها به صورت خطی میپردازیم. ما همچنین از بهینه سازی محدب برای پیدا کردن بهترین وزن ها در هنگام ترکیب خطی این زیر- مدل ها استفاده میکنیم.

Dropout تکنیکی نسبتا جدید است که برای جلوگیری از بیش‌برازیده شدن استفاده میشود. در dropout، ما با احتمال ۵۰٪ خروجی هر نورون پنهان را در فرایند آموزش شبکه برابر با صفر قرار میدهیم. بنابراین در هربار آموزش شبکه، شبکه ما معماری متفاوتی خواهد داشت که اینجا آن را m_k مینامیم. تمام این ساب-مدل ها وزن های مخصوص به خودشان را خواهند داشت. در هنگام تست کردن مدل، ما از تمام نورون ها استفاده میکنیم اما خروجی هریک را در ۰.۵ ضرب میکنیم. این کار معادل میانگین گرفتن بین تمام ساب-مدل هاست.

$$\lim_{n\to\infty}(P_{m_1}(Y=i)\cdot P_{m_2}(Y=i)\cdot\cdots P_{m_n}(Y=i))^{\frac{1}{n}} = \lim_{n\to\infty}\left(\frac{e^{(b_i+W_i\frac{1}{n}\sum_{k=1}^{n}h_{m_k})}}{(\sum_{j=1}^{d}e^{(nb_j+W_j\sum_{k=1}^{n}h_{m_k})})^{\frac{1}{n}}}\right)$$

$$= \frac{e^{(b_i+W_i\cdot h_{comb})}}{\lim_{n\to\infty}(\sum_{j=1}^{d}e^{(nb_j+W_j\sum_{k=1}^{n}h_{m_k})})^{\frac{1}{n}}}$$

$$P_{comb}(Y=i) = \frac{e^{b_i+W_i\cdot h_{comb}}}{\sum_{i=1}^{d}e^{b_j+W_j\cdot h_{comb}}}$$

در معادله بالا، h_{mk} نورون های لایه یکی مانده به آخر ساب مدل m_k است و h_{comb} مربوط به نورون های مدل به دست آمده از ترکیب ساب-مدل ها میباشد.

به نظر میرسد که توزیع پیش بینی مدل ترکیبی فقط یک تقریب از میانگین پیش بینی ساب-مدل ها میباشد. در آزمایش ما، بعد از لایه یکی مانده به آخر، dropout انجام شده و تعداد کلاس های دیتاست نیز ۱۰ میباشد. در ادامه سعی میکنیم از روش بهتری برای ترکیب ساب مدل-ها استفاده کنیم. به جای تقریب زدن، قصد داریم که دقیقا از ترکیب خطی n ساب مدل یک مدل جدید بسازیم.

$$P_{l.comb}(Y = i) = \sum_{k=1}^{n} l_k \times P_{m_k}(Y = i)$$

در فرمول بالا l_k وزن توضیع kاُمین مدل است. میتوانیم وزن های بهینه را با حل کردن مسئله بهینه سازی زیر به دست بیاوریم:

$$\begin{aligned} \min_{l} \quad & \textstyle\sum_{i=1}^{N} \|P_i \cdot l - y_i\|_2^2 \\ \text{s.t.} \quad & l \geq 0 \end{aligned}$$

که در آن N تعداد داده های آموزش، y_i برداری ستونی 10×1 لیبل داده iاُم و P_i نیز $10 \times n$ است که هر ستون آن شامل احتمالات پیش بینی شده هر ساب مدل است. مسئله بالا در واقع یک مسئله QP است که با کمینه کردن جمع مربعات نرم l_2 تفاوت های بین لیبل های پیش بینی شده و لیبل های اصلی، میتوانیم بردار وزن های نامنفی l را پیدا کنیم. اثبات QP بودن مسئله بهینه سازی بالا نیز به صورت زیر میباشد:

$$\begin{aligned} \sum_{i=1}^{N} \|P_i \cdot l - y_i\|_2^2 &= \sum_{i=1}^{N} (P_i \cdot l - y_i)^t (P_i \cdot l - y_i) \\ &= (P \cdot l - y)^t (P \cdot l - y) \\ &= l^t P^t P l - 2 y^t P l + y^t y \end{aligned}$$

که در آن $y = [y_1^T, y_2^T, \dots, y_N^T]^T$ برداری $10N \times 1$ شامل تمامی لیبل‌ها و $P = [P_1^T, P_2^T, \dots, P_N^T]^T$ ماتریسی $10N \times n$ شامل توزیع احتمال تمام نمونه‌ها می باشد. پس میتوان مسئله اصلی را به صورت یک مسئله Quadratic Programming به صورت زیر بیان نمود که با توجه به محدب بودن آن به سادگی قابل حل است:

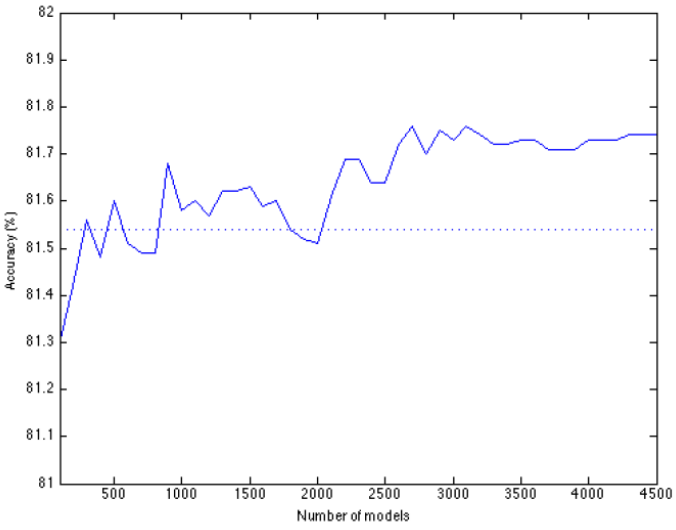
$$\begin{aligned} \min_{l} \quad & l^t P^t P l - 2 y^t P l + y^t y \\ \text{s.t.} \quad & l \geq 0 \end{aligned}$$

ترکیب ساب مدل ها بدون استفاده از Dropout

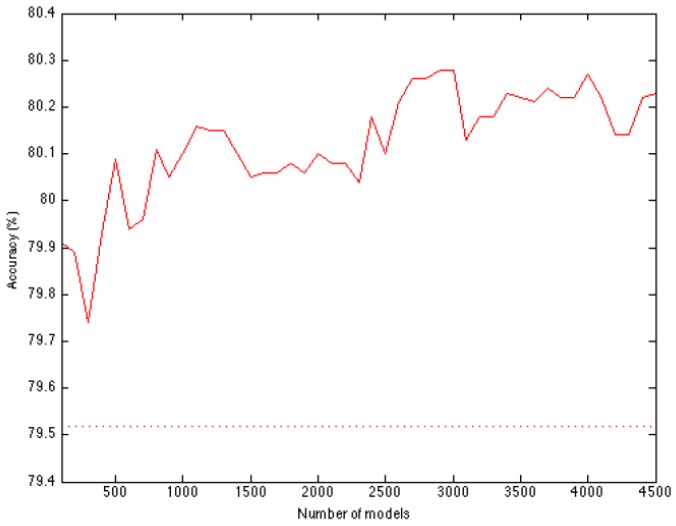
اکنون دوباره به CNN ها بدون استفاده از تکنیک dropout باز میگردیم. ما میتوانیم دقیقا از همان ایده بخش قبل برای ترکیب کردن ساب مدل ها برای بهبود کارایی شبکه CNN وقتی که بدون dropout آموزش دیده استفاده کنیم. مانند بخش قبل، مسئله ای که در اینجا باید برای به دست آوردن وزن های بهینه ساب مدل ها حل کنیم یک مسئله بهینه سازی محدب خواهد بود که میدانیم به سادگی قابل حل بوده و همیشه جواب تحلیلی خواهد داشت.

نتایج مدل جدید

نویسندگان مقاله نتایج زیر را در مقایسه با مدل‌های بدون بهینه سازی محدب ارائه میدهند که در نمودار ۸ آورده شده است:



الف – مقایسه دقت ساب-مدل نسبت به تعداد مدلها همراه با Droupout



ب – مقایسه دقت ساب-مدل نسبت به تعداد مدلها همراه بدون Droupout

نمودار ۸ – مقایسه عملکرد الگوریتم محدب زیر-مدل‌ها با شبکه CNN عادی

همانطور که مشاهده میشود در هر دو حالت الگوریتم پیشنهادی عملکرد بهتری نسبت به شبکه CNN عادی دارد که این بهبود در حالت بدون Dropout به وضوح اثر بیشتری داشته است.

جمع بندی:

در این پروژه ما سعی کردیم ابتدا خلاصه ای از کارکرد شبکه های عصبی عمیقی و تاثیر شگرف آنها در جهان امروز و در کاربردهای صنعتی و علمی ارائه کنیم. سپس موضوع بسیار مهم Over-Parameterization را به طور مفصل بررسی کردیم که میتواند شاهد بسیار مهمی برا این سوال باشند که "چرا شبکه های عصبی عمیق با وجود این تعداد پارامتر همچنان قابلیت تعمیم پذیری بالایی دارند و روی داده های از پیش دیده نشده میتوانند عملکرد بسیار فراتر از انتظار از خود نشان دهند." و با ۲ دلیل سعی در توجیه این پرسش داشتیم.

در ادامه پروژه به بررسی نقش بسیار مهمی که بهینه سازی محدب میتواند در بهبود کارایی و عملکرد شبکه های عصبی عمیق بازی کند پرداختیم. یکی از بزرگترین مشکل های شبکه های عصبی محدب نبودن آنهاست اما ما همچنان میتوانیم با به کارگیری متدهای قدرتمند بهینه سازی محدب عملکرد آنها را به صورت قابل توجهی بهبود ببخشیم.

در سالهای اخیر شبکه های عصبی عمیق به نوبه ای جای پای خود را به عنوان آینده هوش مصنوعی و بخش عظیمی از آینده پیش روی بشریت محکم کرده اند. این شاخه از هوش مصنوعی با وجود عملکرد چشمگیری که از خود نشان داده است، همچنان فاصله بسیار زیادی از بی نقص بودن دارد و همچنان فرصت های زیادی برای مطالعه و بهبود عملکرد آنها باقیست. یکی از مهم ترین ابزاری که میتواند به بی نقص شدن این ابزار قدرتمند کمک کند بهینه سازی محدب است. امید است که در آینده ای نه چندان دور شاهد پیشرفت هرچه بیشتر شبکه های عصبی عمیق به دست متدهای توانمند بهینه سازی محدب باشیم!

منابع:

1. Stuart Geman, Elie Bienenstock, and Ren Doursat: ***Neural networks and the bias/variance dilemma.***

2. Gerd Gigerenzer and Henry Brighton. Homo heuristicus: ***Why biased minds make better inferences.***

3. Zhi-Hua Zhou: ***Why Over-parameterization of Deep Neural Networks Does Not Overfit?***

4. Mikhail Belkin, Daniel Hsu, Siyuan Ma, and Soumik Mandal: ***Reconciling modern machine learning practice and the bias-variance trade-off***

5. David Lopez-Paz, Levent Sagun: ***EASING NON-CONVEX OPTIMIZATION WITH NEURAL NETWORKS***

6. Itay Safran, Gilad Yehudai, Ohad Shamir:***The Effects of Mild Over-parameterization on the Optimization Landscape of Shallow ReLU Neural Networks***

7. Robert Kleinberg, Yuanzhi Li, Yang Yuan: ***An Alternative View: When Does SGD Escape Local Minima?***

8. C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals: ***Understanding deep learning requires rethinking generalization.***

9. Tolga Ergen and Mert Pilanci: ***Convex Optimization for Shallow Neural Networks***

10. Mert Pilanci 1 Tolga Ergen: ***Neural Networks are Convex Regularizers***

11. Tolga Ergen & Mert Pilanci: ***IMPLICIT CONVEX REGULARIZERS OF CNN ARCHITECTURES: CONVEX OPTIMIZATION OF TWO- AND THREE-LAYER NETWORKS IN POLYNOMIAL TIME***

12. Hornik, Kurt; Stinchcombe, Maxwell; White, Halbert: ***Multilayer Feedforward Networks are Universal Approximators***

13. *Si Chen and Yufei Wang****: Convolutional Neural Networks and Convex Optimization***