به نام خدا

دانشگاه تهران

پردیس دانشکده‌های فنی

دانشکده برق و کامپیوتر

هوش مصنوعی

پروژه پنجم (فاز اول)

پیاده‌سازی و آموزش شبکه‌های عصبی Feed Forward

مهلت تحویل: چهارشنبه 10 دی

طراح: آرش هاتفی

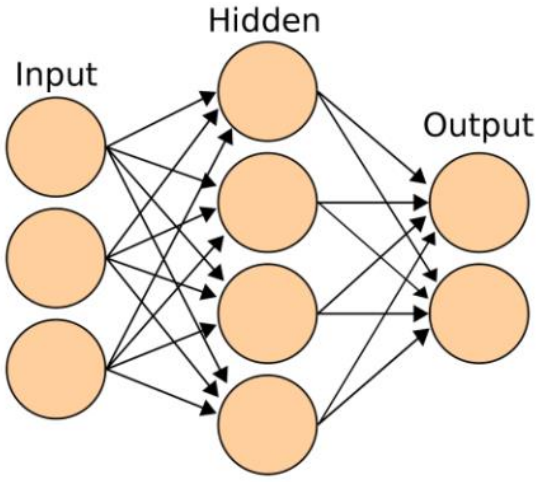
پاییز 99

مقدمه

در فاز اول پروژه پنجم به پیاده‌سازی شبکه‌های عصبی چندلایه جهت طبقه‌بندی تصاویر می‌پردازیم.

شبکه‌ی عصبی Feed Forward

در شبکه‌های عصبی Feed Forward که در درس نیز با آن آشنا شدید، هر تصویر ابتدا مسطح شده و به‌صورت بردار به‌عنوان ورودی شبکه داده می‌شود. هر درایه این بردار (معادل با یک پیکسل تصویر) یک ویژگی برای آن محسوب می‌شود. شبکه قرار است بر اساس این ویژگی‌ها و با ساختن ترکیبات غیرخطی از آن‌ها، وزن اتصالات بین لایه‌هایش را طوری تنظیم کند، که خروجی آن ضمن داشتن کمترین خطا، کلاس تصویر ورودی متناظر را به‌درستی پیش‌بینی کند.



تعریف مسئله

در این تمرین، در ابتدا به پیاده‌سازی یک شبکه‌ی عصبی Feed Forward از پایه و با استفاده از کتابخانه NumPy می‌پردازید. جهت تسریع این فرایند، یک Notebook ناقص از پیاده‌سازی شبکه نیز در اختیار شما قرار می‌گیرید که لازم است از آن استفاده نمایید. در گام دوم، به کمک کدهای بخش اول، چند شبکه‌ی عصبی را روی‌داده‌های Fashion Mnist آموزش خواهید داد و تأثیر چندی از پارامترها را در فرایند یادگیری بررسی خواهید کرد.

مجموعه داده‌ها

در این تمرین شما با یک مجموعه داده Fashion Mnist شامل تصاویری با سایز 28 در 28 پیکسل از 10 لباس مختلف، کارخواهید کرد. فایل زیپ‌شده داده‌ها در صفحه درس قرار داده شده است. این فایل حاوی 4 فایل csv زیر است:

- فایل TrainData.csv: این فایل شامل داده‌های آموزش است. هر سطر از این فایل یک بردار از یک تصویر مسطح به‌اندازه‌ی 784 (28*28) است.
- فایل TrainLabel.csv: این فایل شامل برچسب داده‌های آموزش است. کلاس 10 داده‌ی موجود با اعداد 0 تا 9 نمایش داده‌شده‌اند.
- فایل TestData.csv: این فایل شامل داده‌های تست است. هر سطر از این فایل یک بردار از یک تصویر مسطح به‌اندازه‌ی 784 (28*28) است.
- فایل TestLabel.csv: این فایل شامل برچسب داده‌های تست است. کلاس 10 داده‌ی موجود با اعداد 0 تا 9 نمایش داده‌شده‌اند.

مرحله اول: بررسی و پیش‌پردازش داده‌ها

در این مرحله، سه گام زیر را دنبال کنید:

گام 1: داده‌های Mnist به‌صورت بردارهای مسطح در اختیار شما قرارگرفته‌اند. از میان مجموعه داده‌های آموزش، یک داده از هر کلاس به‌دلخواه انتخاب کنید و نمایش دهید.

گام 2: فراوانی داده‌های هر کلاس در میان داده‌های تست و آموزش را روی یک نمودار میله‌ای ترسیم کنید.

گام 3: مقدار هر یک از پیکسل‌های موجود در تصاویر بین 0 تا 255 است. در ابتدا بردار متناظر این تصاویر را Normalized کنید. (همه‌ی اعداد را بر 255 تقسیم کنید تا مقدار درایه‌های بردارها بین 0 تا 1 شود.) در ادامه از این بردارها در فرایند آموزش استفاده نمایید. مزیت این کار را شرح دهید.

مرحله دوم: تکمیل بخش‌های ناقص شبکه عصبی

یک فایل Notebook شامل کدهای ناقص موردنیاز برای پیاده‌سازی شبکه عصبی Feed Forward آپلود شده و در این قسمت با تکمیل بخش‌های مختلف این فایل، نهایتاً یک کلاس FeedForwardNN خواهید داشت که به کمک آن می‌توانید شبکه‌های عصبی Feed Forward با معماری‌های مختلف پیاده کنید و آموزش دهید. پارامترهای شبکه‌ی موردنظر از طریق روش Stochastic Gradient Discent در طی فرایند آموزش به‌روزرسانی خواهد شد. بخش‌های حذف‌شده از کد که لازم است آن‌ها را کامل کنید، با #TODO مشخص‌شده‌اند.

در ادامه به معرفی مختصر بخش‌های این NoteBook می‌پردازیم:

بخش 1: کلاس Dataloader

از این کلاس جهت آماده‌سازی داده‌های ورودی استفاده می‌شود. این کلاس در Constructor خود، برداری از داده‌ها (data)، برچسب‌های متناظر (labels)، اندازه‌ی batch موردنظر (batch_size) و بُر خوردن یا نخوردن داده‌ها (shuffle) را می‌گیرد. همچنین این کلاس شامل متدها زیر است:

- __onehot جهت ساخت بردارهای onehot از برچسب‌های ورودی. این تابع برچسب‌ها و تعداد کلاس‌ها را به‌عنوان ورودی دریافت می‌نماید.
- __shuffle_dataset جهت بُر زدن هم‌زمان داده‌ها و برچسب‌هایشان
- __iter__ جهت دریافت batch ها

بخش 2: توابع فعال‌ساز (Activation Functions)

در این بخش، 4 تابع فعال‌ساز مرسوم مورداستفاده در شبکه‌های عصبی (Relu / LeakyRelu / Sigmoid / Softmax) و تابع فعال‌ساز همانی یا Identical (برای مدل‌سازی زمانی که قصد استفاده از توابع فعال‌ساز را در یک‌لایه نداریم) در قالب کلاس‌های مجزا پیاده‌سازی خواهند شد. (برای مطالعه بیشتر در مورد این توابع می‌توانید از محتوای این لینک استفاده نمایید.) هر کلاس شامل دو متد اصلی زیر است:

- __val جهت دریافت مقدار تابع به ازای یک رودی خاص
- derivative برای محاسبه‌ی مشتق تابع به ازای یک ورودی خاص

سایر متدهای موجود جهت آسان‌تر کردن کاربری کلاس‌ها است.

به‌عنوان نمونه در این بخش، کلاس Identical به‌طور کامل پیاده‌سازی شده است.

** توجه: برای جلوگیری از overflow در محاسبه‌ی Softmax به ازای ورودی‌های بزرگ، از نسخه‌ی پایدار این تابع (Stable Softmax) استفاده کنید. (در نسخه‌ی پایدار قبل از محاسبه‌ی Softmax برای یک بردار، همه‌ی عناصر بردار را از یک مقدار ثابت کم می‌کنیم تا ماکسیمم درایه‌ی بردار کاهش یابد و overflow اتفاق نیفتد. این مقدار ثابت می‌تواند بزرگ‌ترین درایه‌ی بردار یا هر مقدار دیگر باشد.)

بخش 3: توابع هزینه (Loss Functions)

در این بخش به پیاده‌سازی تابع هزینه‌ی Cross Entropy به همراه Softmax خواهید پرداخت. به‌مانند قسمت قبل، کلاس مربوط به این توابع هزینه نیز دارای دو متد اصلی هستند:

- __val__ جهت دریافت مقدار تابع به ازای یک رودی خاص
- derivative برای محاسبه‌ی مشتق تابع به ازای یک ورودی خاص

سایر متدهای موجود جهت آسان‌تر کردن کاربری کلاس است.

** توجه: تابع هزینه‌ی Cross Entropy در درون خود باید شامل تابع فعال‌ساز Softmax نیز باشد. این به این معنا است که **لازم نیست** که در هنگام استفاده از این تابع هزینه، لایه‌ی آخر شبکه دارای تابع هزینه‌ی Softmax باشد. همچنین، مشتق این دو تابع نیز باید به صورت یکجا گرفته شود و استفاده شود. علت این شیوه‌ی پیاده‌سازی به ساده‌تر شدن فرم مشتق حاصل از قرار گرفتن این دو تابع در پشت‌هم و افزایش پایداری محاسبات شبکه مربوط است. برای اطلاع بیشتر از این موضوع می‌توانید محتوای این لینک را مشاهده نمایید.

بخش 4: کلاس Layer

همان‌گونه که از اسم این کلاس مشخص است، ازآن‌جهت ایجاد کردن هر یک از لایه‌های شبکه‌ی عصبی استفاده می‌شود. این کلاس در constructor خود آرگومان‌های زیر را دریافت می‌نماید:

- اندازه‌ی بردار ورودی به لایه (input_size)
- اندازه‌ی خروجی لایه (output_size)
- تابع فعال‌ساز لایه (activation) که به‌صورت پیش‌فرض identical (یا بدون activation function) است.
- شیوه‌ی وزن دهی اولیه (شامل 3 متد وزن دهی uniform یا normal) و پارامترهای مرتبط با آن. (شیوه‌ی وزن دهی مطلوب مخاطب به‌وسیله‌ی یک‌رشته حرفی در ورودی مشخص می‌شود.)

همچنین، این کلاس دارای متدهای اصلی زیر است:

- متد forward جهت محاسبه‌ی خروجی لایه به ازای یک ورودی خاص. در هر بار صدا شدن این متد، مقادیر ورودی لایه، ورودی تابع فعال‌ساز، مشتق تابع فعال‌ساز نسبت به ورودی و خروجی لایه ذخیره‌سازی می‌شوند تا از آن‌ها در فرایند backpropagation جهت آپدیت کردن وزن‌های شبکه استفاده شود.
- متد update_weights جهت به‌روزرسانی وزن‌های لایه با توجه به جریان گرادیان. این تابع در ورودی خود گرادیان محاسبه‌شده از لایه‌های بعدی و learning Rate را دریافت می‌نماید.
- متدهای __uniform_weight و __normal_weight جهت وزن دهی اولیه با توزیع‌های uniform و normal

سایر متدهای موجود جهت آسان‌تر کردن کاربری کلاس است و نیازی به کامل کردن آن‌ها ندارید.

بخش 5: کلاس FeedForwardNN

از این کلاس برای پیاده‌سازی شبکه‌های feed forward با معماری دلخواه استفاده می‌نماییم. این کلاس در constructor خود سایز ورودی شبکه را دریافت می‌نماید. این کلاس دارای دودسته متد است: از متدهای دسته‌ی اول برای ساختن شبکه و از متدهای دسته‌ی دوم برای آموزش شبکه استفاده می‌شود.

متدهای دسته‌ی اول به شرح زیرند:

- متد add_layer جهت تشکیل معماری شبکه عصبی استفاده می‌شود. از این متد می‌توان برای اضافه کردن یک‌لایه‌ی جدید به انتهای شبکه. این متد، پارامترهای موردنیاز جهت ساخت لایه‌ی جدید ازجمله تعداد نورون‌ها، تابع فعال‌ساز و .. را به‌عنوان آرگومان ورودی دریافت می‌کنید.
- متد forward از این متد جهت محاسبه‌ی خروجی شبکه ایجادشده به ازای یک ورودی دلخواه استفاده می‌شود.
- متد set_training_param جهت تعیین پارامترهای آموزش ازجمله Loss Function و Leanring Rate

متدهای دسته‌ی دوم به شرح زیر هستند:

- متد fit: این متد جهت آموزش شبکه استفاده می‌شود. این متد در ورودی خود، تعداد ایپاک‌های آموزش، dataloader داده‌های آموزش و dataloader داده‌های تست (در صورت تمایل) را دریافت می‌نماید و فرایند آموزش را انجام می‌دهد. درصورتی‌که print_result فعال باشد، بعد از هر epoch آموزش، مقادیر دقت شبکه چاپ می‌شود. خروجی این متد هم یک log از فرایند آموزش شبکه در قالب یک dictionary است.
- متد train__: جهت آموزش شبکه بر روی یک dataloader از داده‌های train برای یک ایپاک استفاده می‌شود.
- متد test__: جهت تست شبکه بر روی یک dataloader از داده‌های test برای یک ایپاک استفاده می‌شود.
- متد train_on_batch__: جهت آموزش شبکه روی‌داده‌های یک batch از داده‌های train استفاده می‌شود.
- متد test_on_batch__: جهت تست شبکه روی‌داده‌های یک batch از داده‌های test استفاده می‌شود.
- متد update_waights__: از این متد جهت آپدیت کردن وزن لایه‌ها به توجه به ورودی و خروجی ذخیره‌شده در آن‌ها استفاده می‌شود. این متد، مقدار واقعی شبکه به ازای یک batch از داده‌ی آموزش و مقدار مورد انتظار در خروجی را دریافت می‌کند.
- متد get_labels__: جهت دریافت برچسب بردارهای خروجی استفاده می‌شود.
- متد compute_accuracy__: جهت محاسبه‌ی accuracy شبکه با توجه به یک خروجی واقعی و مقدار مورد انتظار آن در خروجی

مرحله سوم: طبقه‌بندی داده‌ها

در این بخش به پیاده‌سازی و آموزش شبکه‌های عصبی Feed Forward با پارامترهای مختلف به کمک کلاس FeedForwardNN می‌پردازیم. در بخش Training Sample از NoteBook ناقص، یک مثال از شیوه‌ی استفاده از کلاس FeedForwardNN جهت پیاده‌سازی شبکه‌های عصبی آوردی شده است. انتظار می‌رود که با نوشتن کدهایی مشابه تأثیر پارامترهای زیر را در آموزش شبکه و دقت نهایی آن بررسی کنید و در گزارش نهایی مطرح نمایید:

1- تأثیر Learning Rate
2- تأثیر Batch Size
3- تأثیر تعداد Epoch آموزش
4- تأثیر استفاده از Activation Function های مختلف

برای بررسی موارد بالا مطابق زیر عمل کنید:

- **گام 1:** یک شبکه‌ی عصبی با حداقل دولایه‌ی پنهان طراحی کنید و آن را با پارامترهای زیر آموزش دهید:

Batch Size	32
Number of Training Epochs	30
Activation Function	Relu
Loss Function	Cross Entropy
Weight Initialization	Uniform or Normal

در این مرحله سعی کنید تا مقدار Learning Rate را به‌گونه‌ای انتخاب کنید که مناسب‌ترین دقت را از شبکه دریافت نمایید. توجه کنید که وزن‌دهی اولیه تاثیر بسزایی در همگرایی شبکه دارد. همچنین، در مراحل آینده پارامترهای آموزش را به‌طور پیش‌فرض مانند آنچه در این گام استفاده کردید در نظر بگیرید مگر آنکه **صراحتاً** خلاف آن از شما خواسته‌شده باشد.

- **گام 2:** انتظار دارید بزرگ یا کوچک کردن مقدار Learning Rate نسبت به مقدار بهینه‌ی آن چه تأثیری در فرایند آموزش بگذارد؟ این موضوع را با 10 برابر کردن و 0.1 کردن مقدار Learning Rate به‌دست‌آمده از قسمت 1 آزمایش کنید و نتایج را گزارش نمایید.

- **گام 3:** عملکرد شبکه‌ی طراحی‌شده در گام 1 را به کمک Activation Functionهای Leaky Relu، Tanh و Sigmoid بسنجید و نتایج را مقایسه نمایید. برتری Leaky Relu نسبت به Relu چیست و چرا معمولاً tanh و sigmoid عملکرد مناسبی برای این دست شبکه‌ها ندارند؟

** **توجه:** در ادامه مراحل، از Activation Function با بهترین نتیجه در لایه‌های شبکه استفاده نمایید.

- **گام 4:** عملکرد شبکه را به ازای Batch Size 16 و 128 نیز بسنجید و نتایج را توضیح دهید. فلسفه‌ی استفاده از batch در فرایند آموزش چیست و چگونه batch size های خیلی کوچک می‌توانند آموزش شبکه را مختل نمایند؟

** **توجه:** در ادامه مراحل، بهترین Batch size بررسی‌شده استفاده نمایید.

- **گام 5:** اصولاً چرا لازم است تا شبکه‌های عصبی برای چند Epoch آموزش داده شوند؟ آموزش شبکه‌های عصبی معمولاً تا زمانی ادامه پیدا می‌کند که overfitting شروع شود. سپس، شبکه‌ای با بالاترین دقت روی‌داده‌های تست به‌عنوان شبکه‌های نهایی انتخاب می‌شود. (این فرایند اصطلاحاً Early Stoppnig نامیده می‌شود.) در گام نهایی این بخش شبکه را تا جایی آموزش دهید که فرایند overfitting شروع شود. در این حالت، نمودار دقت و هزینه روی‌داده‌های تست و آموزش را رسم نمایید و با توجه به آن فرایند overfitting روش شبکه‌های عصبی را شرح دهید.

مرحله چهارم: ترسیم داده‌های با بعد کاهش‌یافته

در این بخش قصد داریم به کمک یک شبکه‌ی عصبی Feed Forward، بعد داده‌های ورودی را از 784 به 2 کاهش دهیم و داده‌های با بعد کاهش‌یافته را در فضای دوبعدی ترسیم کنیم. برای این کار، شبکه‌ای با معماری زیر پیاده‌سازی کنید:

- لایه‌ی آخر شبکه (لایه خروجی) 10 نورون داشته باشد (به تعداد کلاس‌ها)
- لایه‌ی یکی مانده به آخر شبکه 2 نورون داشته باشد. (جهت ترسیم خروجی آن در فضای دوبعدی)
- لایه نخست شبکه 784 نورون داشته باشد. (به تعداد فضای ویژگی داده‌های ورودی)
- تعداد نورون‌های سایر لایه‌های شبکه از سمت ورودی به سمت خروجی کاهشی باشد.

بخش نخست شبکه‌ی فوق (شامل تمامی لایه‌ها به‌جز لایه‌ی آخر) را اصطلاحاً شبکه‌ی انکار (Encoder) و وظیفه‌ی آن کاهش بعد ویژگی داده‌های ورودی است. درنهایت، داده‌های کاهش‌یافته در بخش دوم شبکه (لایه آخر) طبقه‌بندی می‌شوند.

قصد داریم تا داده‌های کاش یافته‌ی خروجی لایه‌ی یکی مانده به آخر شبکه‌ی فوق را در فضای دوبعدی ترسیم نماییم. برای این کار، مراحل زیر را دنبال کنید:

1- لازم است تا ابتداً یک متد جدید جهت گرفتن خروجی لایه‌ی یکی مانده به کلاس شبکه اضافه شود.
2- در گام دوم شبکه‌ای با معماری مطرح‌شده طراحی کنید و آموزش دهید و با پارامترهای موردنظرتان آموزش دهید تا زمانی که به‌دقت مناسبی برسید.
3- نهایتاً خروجی لایه‌ی دوم شبکه‌ی آموزش داده‌شده را به ازای داده‌های آموزش و تست ترسیم کنید. نقاط مربوط به هر کلاس را با یک رنگ بخصوص نمایش دهید.

درنهایت، با توجه به نمودارهای به درست آمده، نتیجه را تفسیر کنید و عملکرد شبکه را شرح دهید. با توجه به تصویر به‌دقت آمده، به نظر شما شبکه‌تان در تفکیک کدام دسته از داده‌ها از هم دچار مشکل می‌شود و کدام داده‌ها را به‌راحتی از هم تفکیک می‌کند؟

نکات پایانی

- استفاده از Jupyter Notebook برای انجام این پروژه **الزامی** است.
- معیار سنجش دقت شبکه‌ها در این تمرین تنها Accuracy است. لازم است تا حداقل یکی از شبکه‌های طراحی‌شده در مرحله‌ی سوم و شبکه‌ی طراحی‌شده در بخش چهارم دارای دقت بالاتر از 75 درصد روی‌داده‌های تست و آموزش باشد.
- برای انجام پروژه، تنها مجاز به استفاده از کتابخانه‌ی Numpy و سایر کتابخانه‌های مورداستفاده برای ترسیم نمودارها (مانند matplotlib یا seaborn) هستید.
- نتایج و گزارش خود را در یک فایل فشرده با عنوان AI_CA5_<#SID>.zip تحویل دهید. محتویات پوشه باید شامل Jupyter Notebook و فایل گزارش شما به‌صورت PDF باشد. درصورتی‌که گزارش خود را نیز در Jupyter Notebook می‌نویسید ارسال فایل PDF الزامی نیست و می‌توانید فایل Notebook را به‌صورت HTML ارسال نمایید.
- درصورتی‌که سؤالی در مورد پروژه داشتید بهتر است در فروم درس مطرح کنید تا بقیه از آن استفاده کنند؛ در غیر این صورت توسط ایمیل با طراحان در ارتباط باشید.
- هدف از تمرین، یادگیری شماست. لطفاً تمرین را خودتان انجام دهید.