پروژه شبکه های عصبی

در این پروژه قصد داریم دانش شبکه عصبی خود را به کد تبدیل کنیم . برای این کار ما کدهایی را برای شما آماده کرده ایم که با کامل کردن ToDo های آن و استفاده از آن پروژه ی خود را کامل می کنید و به دانش خوبی از پیاده سازی های شبکه های عصبی می رسید . برای شروع کار با کلون یا فورک کردن این ریپازیتوری شروع کنید .
توجه کنید مطالبی که در ادامه با رنگ سبز نوشته شده اند امتیازی هستند .

بخش اول : پیاده سازی شبکه های عصبی با کتابخانه Pytorch

کتابخانه ی Pytorch یکی از محبوب ترین و پراستفاده ترین کتابخانه ها در حوزه شبکه های عصبی می باشد. در این بخش می خواهیم با استفاده از این کتابخانه دو نوع مدل متفاوت شبکه های عصبی را بر روی مجموعه دادگان MNIST که در واقع 10 کلاس بین 0 تا 9 دارد را برای تشخیص عدد در عکس آموزش دهیم.
(یک مثال کاربردی پیاده سازی شده با کتابخانه pytorch : لینک).
جهت نصب آسان این کتابخانه پیشنهاد می شود فایل های نوتبوک را در فضای ابری Google Colab اجرا کنید.

تسک اول : تشخیص عدد در دیتاست MNIST با شبکه های عصبی fully connected

فایل mnist-linear.ipynb را کامل کنید و دقت بالای 97 درصد کسب کنید.

تسک دوم : تشخیص عدد در دیتاست MNIST با شبکه های عصبی convolution

فایل mnist-cnn.ipynb را کامل کنید و دقت بالای 97 درصد کسب کنید.

تسک سوم : تشخیص تومور مغزی

این دیتاست شامل عکس هایی از مغز است که دارای چهار لیبل هستند (بدون تومور و سه نوع تومور متفاوت) . قرار است از یادگیری انتقالی[1] استفاده کنید که برای این کار از مدل معروف resnet 50 استفاده می کنید فایل Tumor_Project.ipynb در اختیار شما قرار گرفته است که با کامل کردن آن این تسک را انجام میدهید. انتظار می رود که در این تسک دقت بالای 73 درصد کسب کنید.

[1] Transfer learning

بخش دوم: نوشتن یک کتابخانه برای شبکه های عصبی

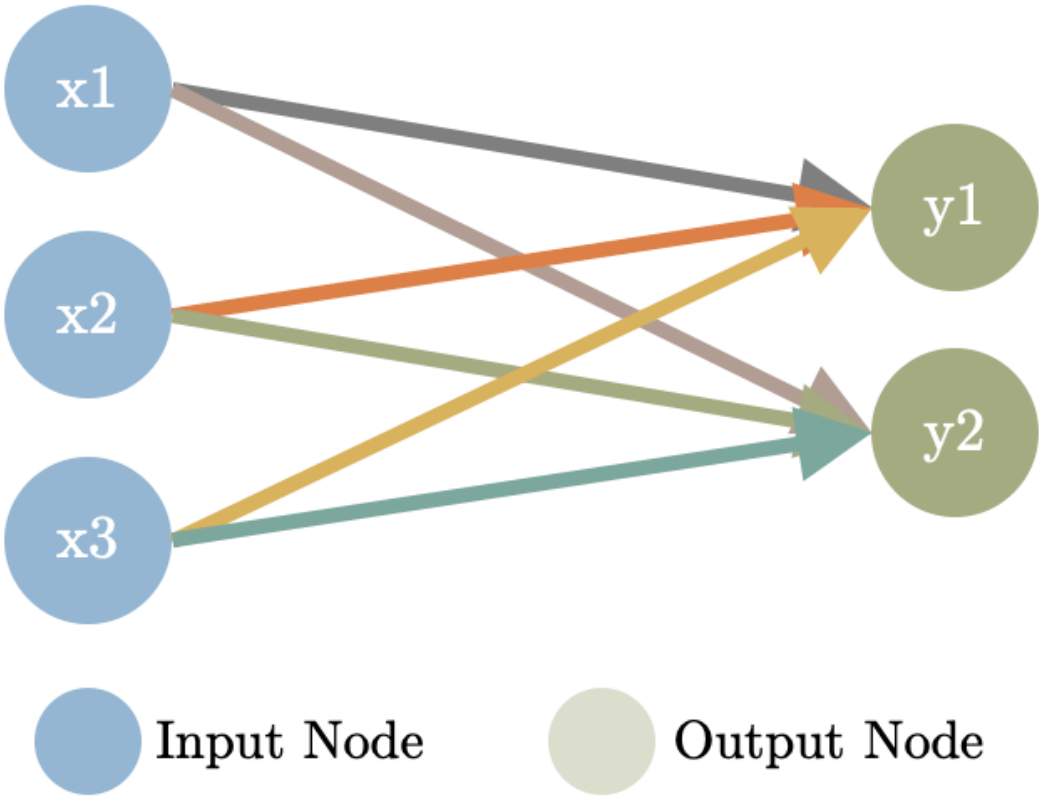
برای درک کامل شبکه های عصبی، بسنده کردن به کتابخانه های آماده همانند Pytorch کافی نیست و به همین علت در این بخش می خواهیم خودمان یک کتابخانه برای ساخت و آموزش شبکه های عصبی توسعه بدهیم.

کلاس Tensor (فایل tensors.py):

در این بخش با کامل کردن ToDo ها ، یک کلاس Tensor خواهیم داشت که همانند کلاس Tensor در کتابخانه ی Pytorch از آن برای تمام عملیات های ریاضی خود استفاده خواهیم کرد. با کامل کردن ToDo های این بخش ، عملیات های ریاضی و همچنین نحوه ی ذخیره ی گرادیان ها برای هر آبجکت Tensor کامل می شود. توضیحات کمک کننده ای برای شما داخل کد وجود دارد.

لایه ها (پوشه ی layers):

لایه Linear : این لایه همان لایه ی مرسوم است که پرسپترون[2] های یک لایه را به پرسپترون های لایه بعد متصل میکند . بعد از رجوع به کد آن شروع به انجام ToDo هایش کنید توضیحات کمک کننده ای برای شما داخل کد وجود دارد.



[2] perceptron

بخش سوم : بهینه ساز ها[3] (پوشه ی optim):

۳ - ۱ : گرادیان کاهشی تصادفی[4] : گرادیان کاهشی جز اولین الگوریتم های بهینه سازی شبکه های عصبی بوده است در این بخش از شما خواسته شده است که بعد از رجوع به کد آن شروع به انجام ToDo هایش کنید توضیحات کمک کننده ای برای شما داخل کد وجود دارد.

۳ - ۲ : امتیازی: بهینه ساز های momentum ، adam و rmsprop امتیازی می باشد و برای کسب نمره ی امتیازی آنها لازم است که در تحویل نشان داده شود با استفاده از این ها task2.py که در ادامه گفته می شود با موفقیت آموزش دیده است.

توابع هزینه (پوشه ی losses) :

۴ - ۱ :کمترین مربعات خطا[5] : این تابع بیشترین کاربردش در زمان پیش بینی مقادیر پیوسته است اما در تسک های کلاسیفیکیشن نیز کاربرد دارد . در این بخش از شما خواسته شده است که بعد از رجوع به کد آن شروع به انجام ToDo هایش کنید توضیحات کمک کننده ای برای شما داخل کد وجود دارد.

۴ - ۲ : کراس آنتروپی[6] : این تابع یکی از محبوب ترین تابع هزینه برای تسک های کلاسیفیکیشن است در این بخش از شما خواسته شده است که بعد از رجوع به کد آن شروع به انجام ToDo هایش کنید توضیحات کمک کننده ای برای شما داخل کد وجود دارد.

اکنون زمان استفاده از این ماژول ها و ابزار ها کنار هم برای پیاده سازی مدل شبکه عصبی خود است . برای این کار از شما میخواهیم به فایل های task4.py و task5.py رفته و ToDo های این فایل را انجام دهید . همچنین توضیحات کمک کننده ای برای شما داخل کد وجود دارد.

تسک چهارم:

در این تسک فرض می شود که ما 100 نمونه داده ی تصادفی که هر داده 3 فیچر دارد، را در یک w که در کد coef می باشد ضرب میکنیم و حاصل را با یک عدد جمع کنیم. سپس می خواهیم صرفا با داشتن 100 نمونه ی ورودی و خروجی حاصل ، مقدار w و b را با استفاده از الگوریتم گرادیان کاهشی بیابیم. در صورتیکه کد صحیح اجرا شده باشد مقدار w و b برابر با مقدار coef و عدد جمع شده خواهد بود. برای اینکار فایل task4.py را کامل کنید.

---

[3] optimizers
[4] Stochastic Gradient Descent
[5] Mean squared error
[6] Categorical Cross entropy

تسک پنجم:

مشابه تسک چهارم ، این بار این کار را با استفاده از تعریف یک لایه ی Linear و با استفاده از کلاس بهینه ساز می خواهیم انجام دهیم. برای اینکار فایل task5.py را کامل کنید.

توابع فعالسازی[7] (پوشه ی activations):

در این بخش شما باید چهار تابع متداول sigmoid , tanh, LeakyRelu , relu را پیاده سازی کنید . بعد از رجوع به کد آن شروع به انجام ToDo هایش کنید توضیحات کمک کننده ای برای شما داخل کد وجود دارد.
پیاده سازی Softmax امتیازی می باشد.

تسک ششم (امتیازی):

اکنون با استفاده از توابع و کلاس هایی که دارید مانند تسک اول اما فقط با استفاده از کتابخانه ی خودتان یک مدل تعریف کنید و آن را بر روی مجموعه دادگان mnist آموزش دهید. فایل mnist-task.py را کامل کنید.

نکات مربوط به پروژه:

1. **پروژه تحویل آنلاین دارد.**
2. **مهلت آپلود پروژه 29 آذر 1402 ساعت 23:59 می باشد.**
3. **پس از کامل کردن پروژه آن را در قالب یک فایل zip در سامانه کورسز آپلود کنید.**
4. **برای تسک تشخیص تومور مغزی یک چکپوینت از مدل آموزش دیده ی خود آماده کنید (نیازی به آپلود چکپوینت نیست) که موقع ارائه بتوانید از آن بدون اینکه از ابتدا آموزش دهید، خروجی بگیرید و دقت آن را نمایش دهید.**
5. **در صورت هرگونه پرسش و ابهام درباره ی پروژه می توانید به آیدی های تلگرام @amirrezarajabb و @msajadcr7 پیام دهید.**

[7] Activation functions