بسم الله الرحمن الرحیم

گزارش پروژه پایانی

استاد راهنما: دکتر سرکار خانم طباطبایی

موضوع:

Movie recommendation system with

combining DNN and autoencoder

تهیه کننده: محمد احسان کریمی نوری

تاریخ: تابستان ۱۴۰۲

فهرست

1.مقدمه

این روزها، مردم به جای اتلاف وقت بر روی مشکل «بعدش چه باید ببینم»، ترجیح می‌دهند توصیه‌هایی را بر اساس تاریخچه‌های قبلی/لیست فیلم‌های تماشا شده او دریافت کنند. با رواج جامعه اطلاعاتی، دسترسی ما به اطلاعات بیشتر از آنچه تاکنون در تمام تاریخ بشر داشته ایم، آسان تر است. با این حال، طبق نظریه تصمیم گیری، اطلاعات بیش از حد می تواند منجر به تصمیمات بدتر شود، نه اینکه مردم را خوشحال کند و اطمینان حاصل کند که آنها به آنچه می خواهند می رسند. در واقع، اضافه بار اطلاعات، مصنوع انقلاب دیجیتال و موبایل، روز به روز جدی تر می شود. در مواجهه با منابع عظیم شبکه، کاربران با تصمیم گیری های دشواری با مشکل اضافه بار اطلاعات مواجه می شوند که می تواند باعث احساس واقعی اضطراب، خستگی ذهنی، ناتوانی و سرگیجه شود. با ارائه آنها به مردم کمک می کنیم تا بتوانند فیلم های مشابه را بر اساس اولویت پیدا کنند.

در حال حاضر برای آنها، این باعث صرفه جویی در زمان زیادی می شود که آنها برای انتخاب خودشان تلف می کردند. برای توصیه، این واقعیت را در نظر می گیریم که آیا برخی افراد همان فیلم را تماشا کرده اند و بعد از آن، بقیه افراد چه فیلم هایی را تماشا می کنند، تا بتوانیم پیش بینی کنیم که او ممکن است به برخی از فیلم هایی که دیگران پس از تماشای فیلم او تماشا کرده اند، علاقه مند باشد. با این کار لیستی از فیلم های مشابه قبلی را در اختیار کاربر قرار می دهیم. یادگیری عمیق به عنوان یک رویکرد مهم در زمینه های مختلف از جمله تجزیه و تحلیل داده ها در وب سایت و حوزه های مرتبط با سلامت بازی کرده است.

امروزه سیستم‌های توصیه سیستم‌های فیلتر اطلاعاتی هستند که از هوش مصنوعی یا الگوریتم‌های هوش مصنوعی استفاده می‌کنند که می‌توانند با فیلتر کردن Big Data با کارایی بالا مشکل اضافه بار اطلاعات را تا حد زیادی کاهش دهند تا محتوا و خدمات شخصی‌سازی شده را در اختیار کاربران قرار دهند که می‌تواند تا حد زیادی مشکل اضافه بار اطلاعات را کاهش دهد.

یک سیستم توصیه نه تنها یک الگوریتم فانتزی است، بلکه در مورد درک داده ها و کاربران شما نیز هست. سیستم های توصیه برای اولین بار در اواسط دهه ۱۹۹۰ به وجود آمدند و از آن زمان تاکنون کانون تحقیقات بوده اند. در حال حاضر بسیاری از پلتفرم ها و وب سایت ها شاهد کاربرد گسترده انواع مختلف سیستم های توصیه هستند. وب سایت های تجارت الکترونیک از سیستم های توصیه برای پیشنهاد محصولات و خدمات شخصی شده از مقالات، کتاب ها، موسیقی و فیلم ها استفاده می کنند. به طور معمول، دو نوع داده مورد استفاده در یک سیستم توصیه معمولی وجود دارد، به عنوان مثال، رتبه‌بندی یا رفتارهای خرید که منعکس‌کننده اطلاعات قابل انتساب در مورد اقلام، کاربران، و کلمات کلیدی یا نمایه‌های بافتی هستند که تعاملات کاربر–مورد را ثبت می‌کنند.

برای سرویس‌های پخش فیلم مانند Netflix، x، Hulu، Amazon Prime، و سایر سیستم‌های توصیه برای کمک به کاربران برای کشف محتوای جدید برای لذت بردن از اهمیت هستند.

1.1 هدف(Purpose)

این بخش سیستم‌های توصیه فیلم را به عنوان الگوریتم‌ها و تکنیک‌هایی تعریف می‌کند که پیشنهادات فیلم شخصی‌سازی شده را به کاربران ارائه می‌کنند. توضیح می‌دهد که چگونه می‌توان با کاهش بار اطلاعات و هدایت کاربران به سمت محتوایی که احتمالاً از آن لذت می‌برند، تجربه کاربر را افزایش داد.

هدف از این پروژه ساخت نرم افزاری است که بتوان از آن برای توصیه فیلم به کاربر استفاده کرد. مردم خیلی رنج می برند تا تصمیم بگیرند که بعداً چه چیزی را تماشا کنند، زمان زیادی برای تصمیم گیری صرف می شود. بعدی خود را برای تماشا انتخاب کنید. بر اساس علاقه قبلی آنها، مدل ما به آنها فیلم ها را توصیه می کند.

یادگیری عمیق، که اساساً فقط شبکه های عصبی مصنوعی عمیق است، می تواند مرزهای تصمیم گیری پیچیده را برای طبقه بندی یا رگرسیون های غیرخطی پیچیده بیاموزد. با قرار دادن تعداد زیادی لایه پنهان در این شبکه‌ها، شبکه‌های عصبی عمیق می‌توانند توابع پیچیده را با یادگیری استخراج بسیاری از ویژگی‌های سطح پایین از داده‌ها و ترکیب آنها در ترکیب‌های غیرخطی مفید بیاموزند.

هدف سند پیشنهادی انجام کاری است که منجر به توسعه رویکردی برای پیش‌بینی با استفاده از یادگیری عمیق می‌شود و هدف پیشنهادی با تقسیم کار به اهداف زیر محقق می‌شود:

۱. به دست آوردن یک مجموعه داده خوب (مجموعه داده ها).

۲. استفاده از روش های یادگیری عمیق برای به دست آوردن دقت بالا / نتایج بهتر.

۳. اعتبار طرح را با استفاده از یادگیری عمیق و بررسی متقاطع نتایج.

مجموعه داده های مورد استفاده: مجموعه داده های مورد استفاده برای این نوت بوک، مجموعه داده های رتبه بندی M۱ از Movie Lens است. این شامل ۱ میلیون رتبه بندی فیلم از ۷۱۲۰ فیلم و ۱۴۰۲۵ کاربر است. این مجموعه داده شامل:

شناسه فیلم: به هر فیلم یک شناسه منحصر به فرد به نام شناسه فیلم داده می شود، این شناسه می تواند برای اهداف شناسایی هر فیلمی استفاده شود. همچنین، هر روشی که قرار است پیاده‌سازی کنیم، تنها با گرفتن مرجع شناسه فیلم، پیاده‌سازی می‌شود، زیرا Movie ID شناسایی هر فیلمی است و برای هر فیلم منحصر به فرد است.

شناسه کاربری: هر کاربر شناسه منحصربه‌فرد خود را دارد تا بتوانیم کاربران را با شناسه آنها ارجاع دهیم، اگر در دو تست یک شناسه کاربری ذکر شده باشد، یعنی هر دو کاربر یکسان هستند.

رتبه بندی: امتیاز یک فیلم از ۱۰ امتیاز است، امتیاز به عوامل مختلفی مانند بازخورد ارائه شده توسط کاربر، بودجه و سود یک فیلم خاص بستگی دارد. بر اساس رتبه بندی، ما تصمیم می گیریم که فیلم در باکس آفیس موفق شود یا شکست بخورد. علاوه بر این، مجموعه داده ای از فیلم ها شامل عنوان فیلم، برچسب ها و ژانرها است.

عنوان: عنوان نام فیلم است، در برخی موارد ممکن است عنوان یکسان باشد، اما اگر فیلم توسط شناسه فیلم انجام شود، شناسایی می شود. عنوان فیلم می تواند توضیح دهد که «فیلم همه چیز درباره چیست» تا حد زیادی.

ژانرها: نوع فیلم را توضیح می دهد که چه عاشقانه، اکشن، درام، هیجان انگیز یا ماجراجویی و ... نوع فیلم مشابهی را که باید پیشنهاد کنیم تا ژانر نقش مهمی در سیستم توصیه ایفا کند.

برچسب‌ها: استفاده از تگ‌های تهیه‌کننده یا کارگردان نشان می‌دهد که چیزی که در مورد این فیلم خاص است به عنوان مثال – نام هر بازیگری می‌تواند در برچسب‌ها باشد یا ژانری می‌تواند در برچسب‌ها باشد، هر نقل قولی نیز می‌تواند یک برچسب باشد.

1.2 طرح در آینده(Scope)

سیستم توصیه فیلم مورد بحث در این سند برای ارائه توصیه های شخصی فیلم به کاربران بر اساس ترجیحات، سابقه مشاهده و سایر عوامل مرتبط طراحی شده است. هدف این سیستم افزایش تجربه کاربر با پیشنهاد فیلم‌هایی است که احتمالاً مورد علاقه کاربر هستند، بنابراین تعامل و رضایت کاربر را افزایش می‌دهد.

در آینده، تکنیک‌های یادگیری عمیق به طور گسترده برای حل سایر برنامه‌های چالش‌برانگیز، مانند طبقه‌بندی ویدئو، تجزیه و تحلیل شبکه‌های اجتماعی، طبقه‌بندی تصاویر در سطح کاملاً جدید، و استنتاج‌های منطقی استفاده خواهند شد. همچنین، ما ممکن است طرح خود را به‌عنوان محصول یا خدمات ارائه کنیم. به شرکتی که به یک سیستم توصیه موثر نیاز دارد.

2. مروری بر سیستم های توصیه فیلم

2.1 اهداف سیستم(System objectives)

هدف سیستم توصیه فیلم دستیابی به اهداف زیر است:

– ارائه توصیه های شخصی فیلم به کاربران

– افزایش تعامل و رضایت کاربر

– بهبود فرآیند کشف و انتخاب فیلم

– افزایش تجربه کاربر در پلت فرم

2.2 اجزای سیستم(System components)

سیستم توصیه فیلم از اجزای کلیدی زیر تشکیل شده است:

– نمایه کاربر: هر کاربر یک نمایه دارد که حاوی اطلاعاتی در مورد ترجیحات، سابقه مشاهده و سایر داده های مرتبط است.

– پایگاه داده فیلم: پایگاه داده ای جامع که حاوی اطلاعاتی درباره فیلم ها از جمله ابرداده ها مانند ژانر، سال انتشار، کارگردان، بازیگران و رتبه بندی کاربران است.

- Recommendation Engine: جزء اصلی سیستم که مسئول تولید توصیه های فیلم بر اساس داده ها و ترجیحات کاربر است.

- رابط کاربری: رابطی که از طریق آن کاربران با سیستم برای مرور و انتخاب فیلم ها تعامل دارند.

2.3 اجزای شبکه عصبی(Neural Network components)

- **Inputs:** ورودی‌های معماری شبکه ما دو بردار n بعدی هستند که n تعداد فیلم‌های موجود در مجموعه داده فیلم، مانند پایگاه داده MovieLens است. یک بردار نمایه کاربر خاصی را رمزگذاری می‌کند و هر بعد نشان‌دهنده امتیازی است که کاربر برای یک فیلم خاص داده است (یا یک صفر برای نشان دادن اینکه هیچ امتیازی داده نشده است). بردار دیگر یک رمزگذاری تک داغ از یک فیلم خاص است (به عنوان مثال، یک بردار با یک بعد "گرم" منفرد روی ۱ تنظیم شده و همه مقادیر دیگر روی صفر تنظیم شده است). این دو بردار درخواست می کنند که شبکه یک امتیاز برای یک کاربر خاص برای یک فیلم خاص پیش بینی کند.

یکی از مزیت‌های این قالب ورودی این است که می‌توانیم بدون یک رتبه‌بندی از یک نمایه کاربر شناخته شده کار کنیم و از رتبه‌بندی شناخته شده برای آیتم پنهان‌شده به عنوان مثال برچسب‌گذاری شده استفاده کنیم. در نتیجه، با وجود اینکه ما فقط ۲۷۰۰۰۰ پروفایل کاربری داریم، هر یک از ۲۶۰۰۰۰۰۰ رتبه بندی فردی یک نمونه قطار است.

- **Hidden layers:** راه‌های مختلفی برای ساختار یک شبکه عصبی پیش‌خور ساده وجود دارد. ما با تعدادی از لایه های استاندارد کاملا متصل شروع می کنیم. با این حال، ما همچنین با ساختارهای جایگزین، مانند ResNets، که در حال حاضر نتایج پیشرفته‌ای را در زمینه‌های دیگر مانند تشخیص تصویر به دست می‌آوریم، آزمایش می‌کنیم.

- **Output:** دو احتمال اصلی برای خروجی شبکه ما وجود دارد. اولین مورد این است که این مشکل را به عنوان یک مشکل طبقه بندی در نظر بگیریم، با پنج کلاس مختلف که نشان دهنده پنج رتبه بندی شروع است که در داده ها وجود دارد. تحت این معماری، ما پنج خروجی شبکه خود را به عنوان احتمالات لاگ غیر عادی در نظر می گیریم و از آنتروپی متقاطع به عنوان تابع ضرر استفاده می کنیم.

با معماری اصلی شبکه عصبی که در بالا معرفی شد، معماری یادگیری عمیق را که به عنوان جایگزینی برای رویکرد همسایگی مبتنی بر کاربر پیشنهاد شده است، توصیف می‌کنیم. ابتدا ابعاد ورودی و خروجی شبکه عصبی را در نظر می گیریم. به منظور به حداکثر رساندن مقدار داده‌های آموزشی که می‌توانیم به شبکه تغذیه کنیم، یک مثال آموزشی را نمایه کاربر در نظر می‌گیریم (یعنی یک ردیف از ماتریس مورد کاربر R) با یک رتبه‌بندی در نظر گرفته نشده است. از دست دادن شبکه در آن مثال آموزشی باید با توجه به رتبه‌بندی تک‌تک‌شده محاسبه شود. نتیجه این امر این است که هر رتبه بندی فردی در مجموعه آموزشی به جای هر کاربر، با یک مثال آموزشی مطابقت دارد.

از آنجایی که ما به آنچه اساساً یک رگرسیون است علاقه مندیم، ما انتخاب می کنیم که از ریشه میانگین مربعات خطا (RMSE) با توجه به رتبه بندی های شناخته شده به عنوان تابع ضرر استفاده کنیم. در مقایسه با میانگین خطای مطلق، ریشه میانگین مربعات خطا به شدت پیش بینی هایی را که دورتر هستند جریمه می کند. ما معتقدیم که این در زمینه سیستم توصیه‌کننده خوب است، زیرا پیش‌بینی رتبه‌بندی بالا برای آیتمی که کاربر از آن لذت نمی‌برد، به طور قابل‌توجهی بر کیفیت توصیه‌ها تأثیر می‌گذارد. از سوی دیگر، خطاهای کوچکتر در پیش‌بینی احتمالاً منجر به توصیه‌هایی می‌شود که هنوز مفید هستند – شاید رگرسیون دقیقاً درست نباشد، اما حداقل بالاترین رتبه پیش‌بینی‌شده احتمالاً به کاربر مربوط است.

2.4 اهمیت توصیه های شخصی

این بخش به اهمیت توصیه‌های فیلم شخصی‌شده عمیق‌تر می‌پردازد. این بحث می‌کند که چگونه پیشنهادات متناسب با سلیقه کاربران می‌تواند تعامل آن‌ها با پلتفرم را افزایش دهد و منجر به رضایت بیشتر کاربر، استفاده طولانی‌مدت و بهبود حفظ مشتری شود. همچنین تأکید می‌کند که چگونه توصیه‌های شخصی‌شده، کشف محتوای جدید و مرتبط را تسهیل می‌کنند.

3. عملکردها

I. User side Func.:

3.1 ثبت نام کاربر و ایجاد پروفایل(registration)

کاربران ملزم به ثبت نام در پلتفرم و ایجاد پروفایل هستند. در طی مراحل ثبت نام، کاربران اطلاعات اولیه مانند نام، سن و جنسیت را ارائه می دهند. آنها همچنین می توانند ترجیحات فیلم، ژانرهای مورد علاقه و بازیگران خود را مشخص کنند.

3.2 انتخاب و مشاهده فیلم(Movie selection)

کاربران می توانند با استفاده از فیلترهای مختلف مانند ژانر، سال انتشار و رتبه بندی، پایگاه داده فیلم را مرور کنند. آنها می توانند اطلاعات دقیق درباره هر فیلم، از جمله خلاصه، بازیگران و رتبه بندی کاربران را مشاهده کنند. کاربران می توانند فیلمی را برای تماشا انتخاب کنند، به آن امتیاز دهند و بازخورد ارائه کنند.

3.3 توصیه های فیلم(Movie recommendation)

موتور توصیه پروفایل های کاربر، تاریخچه مشاهده و تنظیمات برگزیده را تجزیه و تحلیل می کند تا توصیه های فیلم شخصی سازی شود. توصیه ها بر اساس فیلتر مشارکتی، فیلتر مبتنی بر محتوا یا رویکردهای ترکیبی است. کاربران می توانند توصیه هایی را در پلتفرم یا از طریق اعلان های ایمیل شخصی دریافت کنند.

3.4 سیستم رتبه بندی و بازخورد(System rating)

کاربران می توانند به فیلم هایی که تماشا کرده اند امتیاز بدهند و بازخورد ارائه کنند. این اطلاعات برای بهبود الگوریتم توصیه و ارائه پیشنهادات بهتر در آینده استفاده می شود. کاربران همچنین می توانند میانگین امتیاز و بازخورد سایر کاربران را برای کمک به فرآیند انتخاب فیلم خود مشاهده کنند.

II. Key Sys side Func.(اجزای کلیدی سیستم های توصیه فیلم):

3.1 جمع آوری داده ها و پیش پردازش(Data collection & preprocessing)

این بخش فرآیند جمع آوری و پیش پردازش داده ها را برای سیستم های توصیه فیلم توضیح می دهد. این منابع داده های مختلف، از جمله رتبه بندی کاربر، ابرداده فیلم و نمایه های کاربر را بررسی می کند. اهمیت تکنیک‌های تمیز کردن و عادی‌سازی داده‌ها، همراه با استراتژی‌هایی برای مدیریت داده‌های از دست رفته برجسته می‌شود.

3.2 استخراج و نمایش ویژگی(Feature extraction)

در اینجا، استخراج ویژگی های معنادار از ویژگی های فیلم مورد بحث قرار می گیرد. تکنیک‌های مختلف برای نمایش فیلم‌ها و کاربران در سیستم توصیه توضیح داده شده‌اند، از جمله نمایش‌های برداری، جاسازی‌ها و مهندسی ویژگی.

3.3 الگوریتم ها و تکنیک های توصیه(Recommendation algorithms)

این زیربخش به بررسی الگوریتم‌ها و تکنیک‌های توصیه محبوب مورد استفاده در سیستم‌های توصیه فیلم می‌پردازد. الگوریتم‌های فیلتر مبتنی بر محتوا مانند TF-IDF و شباهت کسینوس به تفصیل مورد بحث قرار گرفته‌اند. تکنیک‌های فیلتر مشترک، از جمله روش‌های فاکتورسازی مبتنی بر کاربر، مبتنی بر آیتم و ماتریس نیز توضیح داده شده‌اند. علاوه بر این، رویکردهای توصیه ترکیبی و چارچوب‌های الگوریتمی معرفی شده‌اند که بر توانایی آن‌ها در ترکیب نقاط قوت تکنیک‌های فیلتر مبتنی بر محتوا و مشارکتی تأکید می‌کنند.

3.4 معیارهای ارزیابی برای ارزیابی اثربخشی توصیه ها(Evaluation & assessing)

برای ارزیابی اثربخشی سیستم های توصیه، معیارهای ارزیابی ضروری است. این زیربخش معیارهای رایج مورد استفاده مانند دقت، یادآوری، امتیاز F1 و دقت میانگین را ارائه می‌کند. همچنین تکنیک‌های ارزیابی آفلاین، مانند روش نگه‌داری و اعتبارسنجی متقابل، و همچنین رویکردهای ارزیابی آنلاین، از جمله تست A/B و مطالعات کاربر را مورد بحث قرار می‌دهد.

۴. الگوریتم های توصیه

در این قسمت انواع مختلف سیستم های توصیه فیلم به تفصیل توضیح داده شده است. فیلترینگ مبتنی بر محتوا مورد بحث قرار می‌گیرد و نحوه استفاده از ویژگی‌های فیلم مانند ژانر، کارگردان و بازیگران برای ارائه توصیه‌ها را برجسته می‌کند. فیلتر مشارکتی نیز مورد بررسی قرار می‌گیرد و بر اتکای آن به رفتار کاربر و ترجیحات برای تولید پیشنهادات تأکید می‌کند. سیستم‌های ترکیبی، که رویکردهای فیلتر مبتنی بر محتوا و مشارکتی را ترکیب می‌کنند، معرفی شده‌اند و مزایای خود را در ارائه توصیه‌های دقیق و متنوع نشان می‌دهند.

۴.۱ توضیح رویکرد فیلترینگ مشارکتی

۴.۱.۱ فیلتر مشارکتی (Collaborative Filtering)

فیلترینگ مشارکتی یک الگوریتم توصیه محبوب است که از رفتار و ترجیحات کاربران مشابه برای تولید توصیه‌ها استفاده می‌کند. کاربرانی را که اولویت‌های فیلم مشابهی دارند شناسایی می‌کند و فیلم‌هایی را پیشنهاد می‌کند که این کاربران مشابه آن‌ها را دوست داشته‌اند یا به آن‌ها امتیاز بالایی داده‌اند. فیلتر مشارکتی می‌تواند بر اساس شباهت کاربر–کاربر یا شباهت مورد–اقلام باشد.

I. فیلتر مشارکتی مبتنی بر کاربر:

فیلتر مشارکتی مبتنی بر کاربر بر یافتن کاربران مشابه بر اساس تعاملات گذشته آنها و توصیه مواردی که آن کاربران مشابه دوست داشته اند یا با آنها تعامل داشته اند، تمرکز دارد. این شامل مراحل زیر است:

آ. اندازه‌گیری شباهت: شباهت بین کاربران با استفاده از تکنیک‌های مختلفی مانند شباهت کسینوس، ضریب همبستگی پیرسون یا شباهت جاکارد محاسبه می‌شود. این معیارها شباهت بین کاربران را بر اساس رتبه بندی یا تعاملات آنها کمیت می کند.

ب انتخاب نزدیکترین همسایه: هنگامی که شباهت بین کاربران محاسبه شد، نزدیکترین همسایگان یا مشابه ترین کاربران به کاربر هدف انتخاب می شوند. تعداد همسایگانی که باید در نظر گرفته شوند را می توان با استفاده از آستانه های از پیش تعریف شده یا روش های آماری تعیین کرد.

ج. تولید توصیه: توصیه ها با تجمیع ترجیحات نزدیکترین همسایگان ایجاد می شوند. این را می توان با محاسبه میانگین وزنی رتبه بندی آنها یا استفاده از سایر تکنیک های تجمیع انجام داد.

II. فیلتر مشارکتی مبتنی بر آیتم:

فیلتر مشارکتی مبتنی بر آیتم بر یافتن موارد مشابه بر اساس تعاملات گذشته آنها و توصیه موارد مشابه با مواردی که کاربر قبلاً دوست داشته یا با آنها تعامل داشته است، تمرکز دارد. مراحل مربوط به فیلترینگ مشارکتی مبتنی بر آیتم به شرح زیر است:

آ. محاسبه شباهت آیتم ها:

تشابه بین آیتم ها با استفاده از تکنیک هایی مانند شباهت کسینوس یا ضریب همبستگی پیرسون اندازه گیری می شود. این معیارها شباهت بین موارد را بر اساس رتبه‌بندی یا تعامل کاربرانی که با هر دو مورد تعامل داشته‌اند، نشان می‌دهد.

ب انتخاب بیشترین موارد مشابه:

مشابه ترین موارد با مواردی که کاربر با آنها تعامل داشته انتخاب می شود. این می تواند بر اساس آستانه های از پیش تعریف شده یا روش های آماری باشد.

ج. تولید توصیه:

توصیه ها با در نظر گرفتن مواردی که مشابه مواردی هستند که کاربر قبلاً دوست داشته یا با آنها تعامل داشته است، ایجاد می شود. این رویکرد فرض می کند که اگر کاربر یک مورد را دوست داشته باشد یا با آن تعامل داشته باشد، احتمالاً به موارد مشابه علاقه مند است.

۴.۱.۲ مزایا و محدودیت های سیستم های فیلتر مشارکتی

مزایای سیستم های فیلتر مشترک، از جمله دقت و توانایی آنها برای ارائه توصیه های متنوع، بررسی می شود. چالش‌های ناشی از مشکل شروع سرد و پراکندگی داده‌ها در فیلتر مشترک، همراه با استراتژی‌های کاهش بالقوه نیز بررسی می‌شوند.

۴.۱.۳ پیشرفت ها و ارتقاها ها در توصیه های فیلتر مشارکتی

پیشرفت‌ها و پیشرفت‌ها در توصیه‌های فیلتر مشارکتی بر رسیدگی به چندین چالش و بهبود دقت توصیه‌ها متمرکز شده است. در اینجا برخی از پیشرفت‌های کلیدی در فیلتر مشارکتی وجود دارد: مانند مقیاس‌پذیری، پراکندگی داده‌ها و مشکل شروع سرد. تکنیک هایی مانند فاکتورسازی ماتریس، رویکردهای مبتنی بر همسایگی، مدل های ترکیبی، فیلترسازی آگاه از زمینه، یادگیری عمیق و راه حل هایی برای مشکل شروع سرد به طور قابل توجهی دقت و اثربخشی سیستم های توصیه مبتنی بر فیلتر مشارکتی را بهبود بخشیده است. این پیشرفت ها همچنان به تکامل و اصلاح سیستم های توصیه فیلم کمک می کند.

آ. فاکتورسازی ماتریسی:

تکنیک های فاکتورسازی ماتریس، مانند تجزیه ارزش منفرد (SVD) و فاکتورسازی ماتریس غیر منفی (NMF)، به طور گسترده ای برای بهبود فیلتر مشترک استفاده شده است. این تکنیک‌ها ماتریس تعامل کاربر–مورد را به ماتریس‌های با ابعاد پایین‌تر تجزیه می‌کنند و عوامل پنهانی را که نشان‌دهنده ترجیحات کاربر و ویژگی‌های آیتم هستند، می‌گیرند. با استفاده از این عوامل پنهان، روش‌های فاکتورسازی ماتریسی می‌توانند رتبه‌بندی‌های گمشده را پیش‌بینی کنند و توصیه‌های شخصی‌سازی شده را ایجاد کنند.

ب یادگیری عمیق در فیلتر مشارکتی:

تکنیک‌های یادگیری عمیق، مانند شبکه‌های عصبی عمیق و شبکه‌های عصبی تکراری، برای فیلتر کردن مشارکتی برای ثبت الگوها و وابستگی‌های پیچیده در تعاملات کاربر–مورد استفاده شده‌اند. این مدل‌ها می‌توانند به طور خودکار نمایش سلسله مراتبی ترجیحات کاربر و ویژگی‌های آیتم را بیاموزند که منجر به بهبود دقت توصیه‌ها می‌شود. رویکردهای فیلتر مشارکتی مبتنی بر یادگیری عمیق نتایج امیدوارکننده‌ای را در گرفتن ترجیحات طولانی مدت کاربر، مدیریت داده‌های پراکنده و مدل‌سازی رفتارهای متوالی کاربر نشان داده‌اند.

۴.۲ توضیح رویکرد فیلترینگ مبتنی بر محتوا (Content-based Filtering)

۴.۲.۱ فیلترینگ مبتنی بر محتوا

فیلتر مبتنی بر محتوا فیلم‌ها را به کاربران بر اساس ویژگی‌ها و ویژگی‌های فیلم‌هایی که قبلاً دوست داشته‌اند یا امتیاز بالایی داده‌اند توصیه می‌کند. محتوای و ابرداده فیلم‌ها، مانند ژانر، کارگردان، بازیگران و خلاصه داستان را تجزیه و تحلیل می‌کند تا شباهت‌های بین فیلم‌ها را شناسایی کند و بر اساس این شباهت‌ها توصیه‌هایی ارائه کند.

۴.۲.۲ مزایا و محدودیت های سیستم های مبتنی بر محتوا

در اینجا، مزایای سیستم‌های توصیه مبتنی بر محتوا، از جمله توانایی آنها در ارائه توصیه‌های بی‌نظیر و استحکام آنها در برابر پراکندگی داده‌ها، مورد بحث قرار می‌گیرد. همچنین آنها به هیچ اطلاعاتی در مورد سایر کاربران نیاز ندارند

محدودیت‌های سیستم‌های مبتنی بر محتوا، مانند تنوع محدود و اتکای بیش از حد به نمایه‌های کاربر، نیز بررسی می‌شوند. علاوه بر این، آنها نمی توانند علایق در حال تغییر کاربر را در طول زمان جلب کنند

۴.۲.۳ پیشرفت ها و پیشرفت ها در توصیه های مبتنی بر محتوا

در سال های اخیر، پیشرفت ها و پیشرفت های زیادی در توصیه های مبتنی بر محتوا صورت گرفته است. برخی از این موارد شامل انواع جدیدی از اطلاعات جانبی (مانند ابرداده یا محتوای تولید شده توسط کاربر) برای توصیه یا رویکردهای الگوریتمی جدید برای پردازش اطلاعات موجود است. برخی از این موارد عبارتند از **یادگیری عمیق**(DL)، رویکردهای مبتنی بر فرامسیر، و رویکردهای مرتبط با داده های الگوریتمی

۴.۳ توضیح رویکرد سیستم توصیه ترکیبی (Hybrid Filtering)

۴.۳.۱ رویکردهای ترکیبی

رویکردهای ترکیبی فیلتر مشارکتی و فیلتر مبتنی بر محتوا را برای استفاده از نقاط قوت هر دو الگوریتم ترکیب می‌کنند. هدف آنها ارائه توصیه‌های دقیق‌تر و متنوع‌تر با ترکیب منابع متعدد داده‌ها و اطلاعات است.

۴.۳.۲ مزایای رویکردهای ترکیبی

در اینجا، مزایای سیستم های توصیه ترکیبی با جزئیات بیشتری مورد بحث قرار می گیرد. موضوعات تحت پوشش عبارتند از بهبود دقت توصیه به دست آمده از طریق ترکیبی از تکنیک ها، توانایی پرداختن به محدودیت های رویکردهای توصیه فردی، و افزایش رضایت کاربر ناشی از توصیه های متنوع و شخصی.

5. کار های مشابه

متداول ترین روش انجام فیلترینگ مشترک، استفاده از رویکرد k–نزدیکترین همسایه بین کاربران است. با این تکنیک، ابتدا با یک ماتریس کاربر–مورد R شروع می‌شود، جایی که $R_{i,j}$ رتبه‌بندی کاربر i را برای آیتم j می‌دهد و مقدار ۰ نشان می‌دهد که رتبه‌بندی خاصی وجود ندارد. از R یک ماتریس شباهت کاربر–کاربر S محاسبه می شود، که در آن $S_{i,j}$ شباهت بین کاربر i و کاربر j است که می تواند با $R \cdot RT$ محاسبه شود. توجه داشته باشید که استفاده از سایر معیارهای فاصله، مانند معیار تشابه همبستگی یا شباهت کسینوس، برای پر کردن S نیز موثر است. هنگامی که S محاسبه شد، می‌توانیم رتبه‌بندی کاربر i را برای مورد j با محاسبه $RTj \cdot Si$ پیش‌بینی کنیم، که اساساً میانگین رتبه‌بندی سایر کاربران را برای مورد j با وزن شباهت آنها به کاربر i محاسبه می‌کند.

همچنین می‌توانیم از k شبیه‌ترین کاربران به کاربر i برای پیش‌بینی امتیاز مورد j استفاده کنیم. از نظر تجربی، این عملکرد بهتر از میانگین وزنی روی همه کاربران است، اگرچه برای محاسبه k نزدیک‌ترین همسایه‌ها، مقداری کار اضافی در زمان آزمایش لازم است. این رویکرد بر این فرض تکیه دارد که اگر دو کاربر به یک مورد مشابه امتیاز دهند، احتمالاً به موارد دیگر نیز امتیاز مشابهی می دهند. در مقیاس، ساختارهای داده مانند درختان توپ و درختان k-d (یک درخت پارتیشن فضایی دودویی در ابعاد k) می توانند برای محاسبه موثرتر همسایگان محلی بین پروفایل های کاربر استفاده شوند.

یک رویکرد جایگزین k–نزدیکترین همسایه در عوض شباهت بین جفت آیتم ها (در مقابل کاربران) را با این ایده که کاربرانی که یک آیتم خاص را دوست دارند، موارد مشابه را دوست خواهند داشت محاسبه می کند. با این رویکرد ما یک ماتریس شباهت مورد به مورد I را به صورت $R \cdot RT$ محاسبه می کنیم. مانند قبل، می‌توانیم از معیارهای مشابه دیگری نیز برای پر کردن I استفاده کنیم. برای پیش‌بینی رتبه‌بندی کاربر i در مورد j، می‌توانیم $Ri \cdot Ij$ را محاسبه کنیم که میانگین رتبه‌بندی‌های ارائه‌شده توسط کاربر i را با وزن شباهت نشان می‌دهد. آن موارد به مورد j. از آنجایی که تعداد کاربران بسیار بیشتر از موارد در یک سیستم توصیه‌گر است، فیلتر مشترک کاربر–کاربر می‌تواند عملکرد بیشتری داشته باشد.

یکی دیگر از روش‌های رایج برای انجام فیلترینگ مشارکتی، فاکتورسازی ماتریس است. با این تکنیک، یک ماتریس کاربر– آیتم به دو ماتریس با بعد داخلی که نشان دهنده برخی عوامل پنهان با استفاده از تکنیک‌هایی مانند تجزیه ارزش منفرد (SVD) است، فاکتور می‌شود. فاکتوربندی حاصل، هم کاربران و هم اقلام را از نظر عوامل پنهان نشان می‌دهد، به گونه‌ای که می‌توان از آنها برای توصیه موارد جدید استفاده کرد. همانند رویکردهای همسایگی مورد–آیتم، آزمایش‌های اولیه ما روی رتبه‌بندی فیلم نشان می‌دهد که رویکردهای همسایگی کاربر–کاربر نسبت به فاکتورسازی ماتریس برتری دارند.

یادگیری عمیق بسیاری از زمینه های علوم کامپیوتر از جمله پردازش زبان طبیعی را متحول کرده است. با وجود این، یادگیری عمیق در حوزه سیستم های توصیه گر نسبتاً جدید است و توجه زیادی به آن نشده است. یک مدل یادگیری عمیق مشارکتی (CDL) پیشنهاد کنید که به طور مشترک عمیق را انجام دهد. یادگیری بازنمایی برای اطلاعات محتوا و فیلتر مشارکتی برای ماتریس رتبه‌بندی. CDL با ما متفاوت است، زیرا اولی به اطلاعات محتوا متکی است، در حالی که ما نه. یک سیستم توصیه یادگیری عمیق را با توجه به تاریخچه مرور وب و جستجوهای ارائه شده توسط کاربران معرفی کنید. آنها شباهت بین کاربران و آیتم های ترجیحی آنها را با نگاشت کاربران و آیتم ها در یک فضای پنهان به حداکثر می رساند. محدودیتی که بر این رویکرد اعمال می‌شود این است که تاریخچه مرور و جستجوهای کاربران مورد نیاز است که همیشه در دسترس نیستند. یک مدل شبکه عصبی عمیق ایجاد کنید که ویژگی‌های محتوای آیتم‌ها را برای پیش‌بینی رتبه‌بندی آیتم‌های شروع سرد استخراج می‌کند. سیستم توصیه ما متفاوت است، زیرا ما با محتوای کاربر سروکار نداریم.

همانطور که در شکل ۱ نشان داده شده است، انواع بسیاری از الگوریتم ها در سیستم های مختلف توصیه فیلم در دو دهه گذشته آزمایش و آزمایش شده اند.

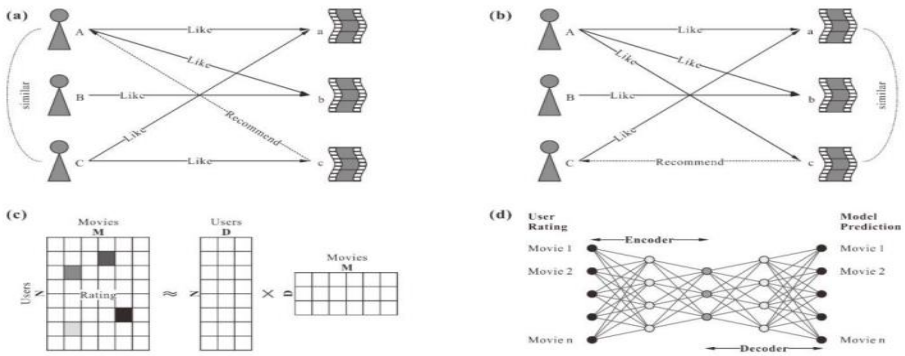


Figure 1. Typical recommendation methods. **(a)** User-based collaborative filtering, **(b)** Item-based collaborative filtering, **(c)** Model-based collaborative filtering, **(d)** Deep-learning neural networks.

۶. معماری DNN

۱) *Autoencoder*: یکی از مدل های یادگیری عمیق موجود، مدل شبکه عصبی عمیق (DNN) است. DNN یک مدل پرسپترون چند لایه (MLP) با لایه های پنهان بسیاری است. منحصر به فرد بودن DNN به دلیل تعداد بیشتر واحدهای پنهان و تکنیک های اولیه سازی پارامترهای بهتر است. یک مدل DNN با تعداد زیادی واحد پنهان می تواند قدرت مدل سازی بهتری داشته باشد. اگرچه پارامترهای آموخته شده مدل DNN یک بهینه محلی است که به داده های آموزشی بیشتر و قدرت محاسباتی بیشتری نیاز دارد، اما می تواند بسیار بهتر از مدل هایی با واحدهای پنهان کمتر عمل کند. Deep Auto Encoder نوع خاصی از DNN است. (شکل ۱ را برای نمونه Autoencoder ببینید.)

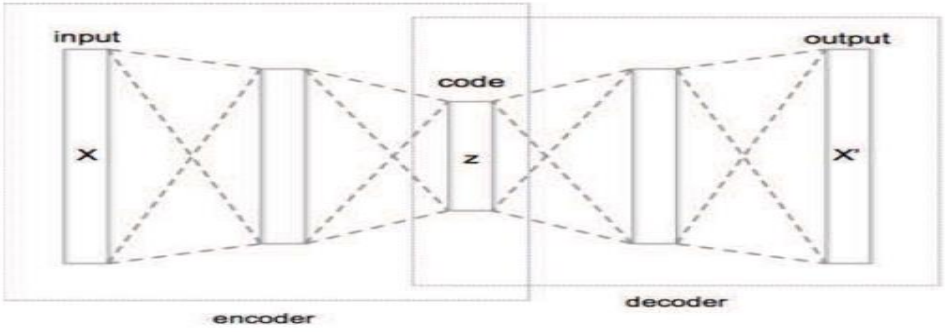


Figure 1. An autoencoder with three fully-connected hidden layers

رمزگذار خودکار(Autoencoder) یک شبکه عصبی است که آموزش داده شده است تا ورودی خود را در خروجی خود کپی کند، با هدف معمول کاهش ابعاد، یعنی فرآیند کاهش تعداد متغیرهای تصادفی مورد بررسی. دارای یک تابع رمزگذار برای ایجاد یک لایه پنهان (یا چندین لایه پنهان) است که حاوی کدی برای توصیف ورودی است. یک رمزگشا وجود دارد که یک بازسازی ورودی از لایه پنهان ایجاد می کند.
رمزگذار خودکار می تواند با داشتن یک لایه مخفی کوچکتر از لایه ورودی مفید واقع شود و آن را وادار کند تا با یادگیری همبستگی در داده ها، نمایشی فشرده از داده ها در لایه پنهان ایجاد کند. این رمزگذار خودکار نوعی یادگیری بدون نظارت است، به این معنی که یک رمزگذار خودکار فقط به داده‌های بدون برچسب نیاز دارد، که مجموعه‌ای از داده‌های ورودی است نه جفت‌های ورودی–خروجی. از طریق یک الگوریتم یادگیری

بدون نظارت، برای بازسازی های خطی، رمزگذار خودکار سعی می کند تابعی را یاد بگیرد تا اختلاف میانگین مربع ریشه را به حداقل برساند. برای محاسبه ریشه میانگین مربعات خطا (RMSE) یک مدل یادگیری ماشینی، می توانیم عملکرد مدل را اندازه گیری کنیم. RMSE به صورت تعریف شده است :

$$RMSE = \frac{1}{m}\sum_i (\hat{y} - y)_i^2, \quad \hat{y} = \mathbf{w}^T\mathbf{x} \qquad (1)$$

که در آن $w \in R^n$ بردار پارامترها است، $x \in R^n$ برداری است که برای پیش بینی مقدار اسکالر $y \in R$ استفاده می شود و $\hat{y}$ مقداری است که یک مدل یادگیری ماشینی پیش بینی می کند که مقدار اسکالر $y \in R$ باید چقدر باشد. توجه داشته باشید که RMSE زمانی که $\hat{y} = y$ به ۰ کاهش می یابد و با افزایش فاصله اقلیدسی بین مقادیر پیش بینی شده و مقادیر هدف، خطا افزایش می یابد.

۲) *پرسپترون چندلایه (Multilayer Perceptron)*: در ابتدا، معماری سیستم توصیه‌گر ما شامل ورودی از ردیف ماتریس کاربر–مورد R با رتبه‌بندی برخی از آیتم‌های j به همراه یک پرس و جوی رمزگذاری‌شده یک‌طرفه است که نشان می‌دهد شبکه باید پیش‌بینی کند. امتیاز برای کاربر i در مورد j. متأسفانه، ثابت شده است که آموزش این معماری دشوار است، زیرا شبکه باید یاد بگیرد که نه تنها نمایه های کاربر، بلکه تعامل بین آن پروفایل ها و ورودی های پرس و جو را نیز درک کند. با توجه به ریشه میانگین مربعات خطا در داده های آموزشی، ما هرگز با این معماری به ضرر کمتر از ۱.۲ نرسیدیم.

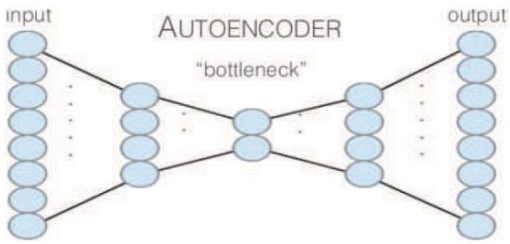


Figure 2. Overview of the network architecture of our recommender

در عوض، ما از مفهوم رمزگذار خودکار برای طراحی معماری شبکه عصبی خود الهام می گیریم (شکل ۲ را ببینید). این معماری ساده یک ورودی دریافت می کند و آن را به تعدادی از لایه های پنهان کاملاً متصل که شامل یک «گلوگاه» است متصل می کند. این گلوگاه یک لایه پنهان است که ابعاد بسیار کوچکتری نسبت به ورودی دارد. سپس خروجی شبکه مجدداً گسترش می یابد تا ابعادی مشابه ورودی داشته باشد. سپس شبکه برای یادگیری تابع هویت آموزش داده می شود، با این ایده که برای اینکه شبکه بتواند تابع هویت را از طریق گلوگاه محاسبه کند، باید یک نمایش متراکم از ورودی را بیاموزد. بنابراین، رمزگذار خودکار را می توان به عنوان چیزی شبیه به تکنیک کاهش ابعاد در نظر گرفت. همچنین می‌توان امیدوار بود که لایه گلوگاه چیز مفیدی در رابطه با ساختار زیربنایی ورودی یاد بگیرد. به عنوان مثال، یک نورون در لایه گلوگاه ممکن است چیزی مرتبط با ژانر یک فیلم یا گروه بندی فیلم های مشابه را نشان دهد. توجه داشته باشید که ما علاقه ای به یادگیری محاسبه یک تابع هویت نداریم – در نهایت، هدف ما پیش‌بینی رتبه‌بندی‌های گمشده است، نه بازتولید صفرها در بردارهای ورودی. در نتیجه، در حالی که معماری شبکه نهایی ما شبیه یک رمزگذار خودکار با لایه‌های پنهان گلوگاه و ابعاد منطبق بر ورودی و خروجی است، شبکه در واقع با استفاده از یک تابع ضرر برای رگرسیون (یعنی RMSE) با هدف یادگیری پیش‌بینی رتبه‌بندی‌های گمشده آموزش داده می‌شود. به طور خاص، نمونه‌های آموزشی برای شبکه، نمایه‌های کاربر با یک رتبه‌بندی پنهان است، و خروجی رتبه‌بندی‌های پیش‌بینی‌شده برای همه فیلم‌های مجموعه داده است. در حالی که انتظار می‌رود شبکه رتبه‌بندی هر فیلم را بر اساس نمایه کاربر پیش‌بینی کند، ما فقط برای رتبه‌بندی پنهان شده پاسخ داریم. در نتیجه، ما فقط هنگام یادگیری از مثال آموزشی، ضرر را برای رتبه از دست رفته منتشر می کنیم.

۳) سیستم توصیه‌کننده یادگیری عمیق(The Deep Learning Recommender System): رتبه‌بندی‌های خودداری این نتیجه ناگوار را به همراه دارد که مدل یادگیری عمیق ما فقط می‌تواند رتبه‌بندی‌هایی را برای فیلم‌هایی مشابه آنچه کاربر تماشا کرده است یاد بگیرد، زیرا عملکرد ضرر مستقیماً تحت تأثیر خروجی غیرمرتبط قرار نمی‌گیرد. فیلم ها. به دلیل لایه تنگنا، مدل باید تا حدی تعمیم یابد، اما این مدل ممکن است برای فیلم هایی که به شدت متفاوت از فیلم هایی است که کاربر واقعاً رتبه بندی کرده است، مشکل داشته باشد. در حالی که کاربران فیلم‌ها را تماشا می‌کنند و امتیاز پایینی دارند، بیشتر اوقات به بیش از چند صد مورد امتیاز نمی‌دهند و از تماشای فیلم‌های کاملاً غیر مرتبط اجتناب می‌کنند، بنابراین ممکن است پیش‌بینی رتبه‌بندی برای فیلم‌های کاملاً نامرتبط برای مدل دشوار باشد. برای مقاصد تابع ضرر ما، که ریشه میانگین مربعات خطا در رتبه‌بندی‌های شناخته شده است، این واقعیت که شبکه ما ممکن است نحوه خروجی رتبه‌بندی فیلم‌های کاملاً نامرتبط را یاد نگیرد، به نظر نمی‌رسد که روی از دست دادن تست تأثیر بگذارد، احتمالاً به این دلیل که فیلم‌های موجود در آزمایش داده‌ها به اندازه‌ای مرتبط هستند که الگوهای آموخته‌شده از داده‌های آموزشی به رتبه‌بندی‌های داده‌های آزمون تعمیم می‌یابند. البته، ممکن است بر رتبه‌بندی تأثیر بگذارد، بنابراین می‌تواند مطلوب باشد که یک اصطلاح تنظیم به ضرر اضافه شود. با استفاده از این طرح اولیه، ما چندین گونه از این معماری را با استفاده از تعداد لایه های مختلف و اندازه های مختلف برای لایه گلوگاه آزمایش کرده ایم. جالب‌ترین پارامتر اندازه کوچکترین لایه گلوگاه بود و پس از آزمایش مقادیر مختلف، در نهایت بر روی اندازه گلوگاه ۵۱۲ قرار گرفتیم. از آنجا با تعداد مختلفی از لایه‌های کاملاً متصل آزمایش کردیم، همیشه از توان ۲ برای افزایش و افزایش استفاده کردیم. کاهش ابعاد توپولوژی شبکه نهایی دارای هفت لایه پنهان کاملاً متصل با ابعاد [۴۰۹۶، ۲۰۴۸، ۱۰۲۴، ۵۱۲، ۱۰۲۴، ۲۰۴۸، ۴۰۹۶] است. هر لایه از یک واحد خطی اصلاح شده ۴ به عنوان تابع فعال سازی غیر خطی استفاده می کند. وزن‌های اتصال لایه‌های پنهان با استفاده از مقداردهی اولیه خاویر با بایاس‌ها روی صفر تنظیم شدند.

۴) خوشه بندی(Clustering): ایده استفاده از کوچکترین لایه گلوگاه در شبکه را به عنوان نوعی خوشه بندی طبیعی در نظر گرفته ایم. با فشار دادن ورودی به چنین فضای ابعادی کوچکی، مدل باید لزوماً چیزی در مورد ساختار زیربنایی داده های ورودی بیاموزد. فرضیه این بود که با تثبیت یک نورون منفرد در لایه گلوگاه و صفر کردن نورون‌های باقیمانده در لایه گلوگاه، و سپس بهینه‌سازی فضای ورودی برای این فعال‌سازی خاص، می‌توانیم با نمایش فیلم‌هایی که هر خوشه را تحریک می‌کنند، آن ساختار را تجسم کنیم. برای مثال، ما انتظار داریم که ممکن است یک نورون یا مجموعه کوچکی از نورون‌ها وجود داشته باشد که ژانرهای مختلف فیلم یا سبک‌های مختلف فیلم‌شناسی را تحریک کند. جدول I مثالی از چنین "خوشه ای" را از بهینه سازی ورودی برای راه اندازی یک نورون گلوگاه ارائه می دهد. این فیلم ها مضمون مشترکی دارند. بدیهی است که برای اینکه این شبکه بتواند رتبه بندی فیلم ها را به طور دقیق پیش بینی کند، باید نوعی ساختار را یاد بگیرد. با این حال، این ساختار بیش از حد انتظار در سراسر لایه گلوگاه توزیع شده است. یکی از راه‌حل‌های بالقوه برای این مشکل، اضافه کردن یک اصطلاح منظم‌سازی به ضرر است که باعث ایجاد پراکندگی در لایه گلوگاه می‌شود.

Table I
A CLUSTER WHEN OPTIMIZING THE INPUT TO TRIGGER A SINGLE BOTTLENECK NEURON

Jules and Jim (Jules et Jim) (1961)
Frankenstein Must Be Destroyed (1969)
Lolita (1962)
Lawnmower Man, The (1992)
First Knight (1995)
Urban Legends: Final Cut (2000)
Fair Game (1995)
Guinevere (1999)
Paradine Case, The (1947)
400 Blows The (Les quatre cents coups) (1959)

۷. چالش ها و پیشرفت های پیاده سازی در سیستم های توصیه فیلم

۷.۱ مشکل شروع سرد و پراکندگی داده ها(Cold Start Problem and Data Sparsity)

مشکل شروع سرد زمانی رخ می دهد که یک سیستم توصیه فاقد اطلاعات کافی کاربر یا آیتم برای پیش بینی دقیق باشد. به همین دلیل است که به چالش ایجاد توصیه‌های دقیق برای کاربران جدید یا فیلم‌های جدید با داده‌های محدود اشاره دارد. سیستم توصیه باید موقعیت‌هایی را مدیریت کند که داده‌های کاربر یا داده‌های فیلم کافی برای پیش‌بینی دقیق وجود ندارد.

برای مقابله با این چالش، می توان از توصیه های مبتنی بر محتوا استفاده کرد. به عنوان مثال، یک سیستم توصیه فیلم می تواند از ویژگی های فیلم مانند ژانر، کارگردان و بازیگران برای ارائه پیشنهادات اولیه به کاربران جدید استفاده کند. علاوه بر این، رویکردهای ترکیبی که تکنیک‌های فیلتر مبتنی بر محتوا و مشارکتی را ترکیب می‌کنند، می‌توانند با استفاده از داده‌های محتوا و رفتار کاربر به کاهش مشکل شروع سرد کمک کنند. سیستم توصیه فیلم نتفلیکس از چنین رویکردهای ترکیبی برای ارائه توصیه هایی حتی برای کاربران جدید با داده های محدود استفاده می کند.

با توجه به پراکندگی داده ها، تکنیک های فیلتر مشارکتی می توانند با اعمال نفوذ بر رفتار و ترجیحات کاربران مشابه، بر این چالش غلبه کنند. به عنوان مثال، اگر یک کاربر تنظیمات ترجیحی فیلم مشابهی را با سایر کاربران نشان دهد، الگوریتم‌های فیلتر مشترک می‌توانند فیلم‌هایی را توصیه کنند که کاربران مشابه از آنها لذت می‌برند. این رویکرد به ویژه زمانی مؤثر است که همپوشانی محدودی در رتبه‌بندی کاربران وجود داشته باشد. Amazon Prime Video از فیلتر مشترک برای ایجاد توصیه‌هایی بر اساس رتبه‌بندی و رفتار کاربران، حتی در سناریوهایی با داده‌های پراکنده استفاده می‌کند.

۷.۲ مقیاس پذیری و توصیه های بلادرنگ(Scalability and Real-Time Recommendations)

همانطور که پایگاه کاربر و پایگاه داده فیلم رشد می کند، سیستم توصیه باید بتواند حجم زیادی از داده ها را مدیریت کند و توصیه های بلادرنگ را بدون کاهش عملکرد قابل توجه ارائه دهد. سیستم‌های پیشنهادی هنگام برخورد با پایگاه‌های کاربری بزرگ و کاتالوگ‌های فیلم گسترده با چالش‌های مقیاس‌پذیری مواجه هستند. مقیاس پذیری عاملی حیاتی برای اطمینان از اثربخشی و پاسخگویی سیستم است.

برای پرداختن به این، از روش‌های محاسباتی توزیع شده و پردازش موازی استفاده می‌شود. برای مثال، پلتفرم‌هایی مانند YouTube و Netflix از سیستم‌های توزیع‌شده برای مدیریت میلیون‌ها کاربر و کتابخانه‌های وسیع محتوا استفاده می‌کنند و از توصیه‌های سریع و مقیاس‌پذیر اطمینان می‌دهند. توصیه‌های بلادرنگ با به‌روزرسانی مداوم تنظیمات برگزیده کاربر و ترکیب فعالیت‌های اخیر به دست می‌آیند. به عنوان مثال، Spotify بر اساس عادات گوش دادن فعلی کاربران، توصیه‌های بلادرنگ ارائه می‌کند و با تغییر اولویت‌ها، توصیه‌ها را به صورت پویا تنظیم می‌کند.

۷.۳ حریم خصوصی و امنیت داده ها(Privacy and Data Security)

سیستم توصیه فیلم داده‌های کاربر را جمع‌آوری و تجزیه و تحلیل می‌کند تا توصیه‌هایی ایجاد کند. اجرای اقدامات امنیتی مناسب برای محافظت از حریم خصوصی کاربر و اطمینان از اینکه داده های کاربر به طور ایمن مدیریت می شوند ضروری است. رعایت مقررات حفاظت از داده ها باید در اولویت باشد.

برای رفع این مشکل، پروتکل های رمزگذاری قوی داده و احراز هویت کاربر را برای محافظت از اطلاعات شخصی اجرا کنید. علاوه بر این، گستره جمع‌آوری داده‌ها را فقط به آنچه برای ایجاد توصیه‌ها ضروری است محدود کنید و به کاربران این امکان را بدهید که در هر زمانی داده‌های خود را انصراف دهند یا حذف کنند. ممیزی های منظم و به روز رسانی اقدامات امنیتی نیز باید برای اطمینان از حفاظت مداوم از اطلاعات کاربر اجرا شود.

۷.۴ ترکیب عوامل زمینه و زمانی(Incorporating Context and Temporal Factors)

عوامل زمینه ای و زمانی نقش حیاتی در توصیه های فیلم دارند. به عنوان مثال، یک سیستم توصیه می تواند زمان روز، شرایط آب و هوایی یا مکان کاربر را برای پیشنهاد فیلم های مناسب در نظر بگیرد. اگر کاربر در عصر به پلتفرم دسترسی داشته باشد، سیستم ممکن است فیلم های آرامش بخش یا سرگرم کننده را توصیه کند. Netflix با شخصی‌سازی توصیه‌ها بر اساس زمان روز و الگوهای مشاهده تاریخی کاربر، زمینه را در نظر می‌گیرد.

مثال دیگر گنجاندن پویایی های زمانی در توصیه ها است. با تجزیه و تحلیل رفتار کاربر در طول زمان، سیستم های توصیه می توانند با تغییر تنظیمات سازگار شوند. الگوریتم توصیه نتفلیکس تاریخچه بازدید کاربر را تجزیه و تحلیل می کند و به تدریج توصیه ها را بر اساس سلیقه در حال تکامل آنها به روز می کند. این تضمین می کند که سیستم پیشنهادات فیلم به روز و مرتبط را ارائه می دهد.

۷.۵ یادگیری عمیق و پیشرفت های یادگیری ماشین(Deep Learning and Machine Learning Advancements)

تکنیک های یادگیری عمیق و یادگیری ماشین انقلابی در سیستم های توصیه فیلم ایجاد کرده است. شبکه‌های عصبی، مانند شبکه‌های عصبی کانولوشن (CNN) و شبکه‌های عصبی تکراری (RNN)، برای ثبت الگوها و وابستگی‌های پیچیده در داده‌های فیلم استفاده شده‌اند.

به عنوان مثال، نتفلیکس از مدل های یادگیری عمیق برای بهبود توصیه های فیلم خود استفاده کرد. این شرکت الگوریتمی مبتنی بر ماشین‌های محدود بولتزمن (RBM) برای یادگیری نمایش ویژگی‌های فیلم‌ها و کاربران معرفی کرد. این امکان پیش بینی های دقیق تری را با ثبت روابط پیچیده بین فیلم ها و ترجیحات کاربران فراهم می کرد.

مثال دیگر استفاده از رمزگذارهای خودکار در سیستم های توصیه است. رمزگذارهای خودکار می توانند نمایش های فشرده رتبه بندی یا ویژگی های فیلم را بیاموزند. با بازسازی ورودی، آنها می توانند الگوهای زیربنایی را شناسایی کرده و توصیه هایی ایجاد کنند. رمزگذارهای خودکار با موفقیت در سیستم های توصیه فیلم، مانند سیستم هایی که توسط Spotify و Last.fm استفاده می شوند، استفاده شده اند.

۸. ارزیابی و معیارها برای سیستم های توصیه

۸.۱ معیارهای ارزیابی رایج(Commonly Used Evaluation Metrics)

برای ارزیابی عملکرد سیستم های توصیه، معمولاً از چندین معیار استفاده می شود. دقت، یادآوری، امتیاز F1، و میانگین دقت میانگین (MAP) به طور گسترده برای اندازه گیری اثربخشی توصیه استفاده می شود. این معیارها توانایی سیستم را برای ارائه پیشنهادهای مرتبط و دقیق فیلم ارزیابی می کند. برای مثال، دقت نسبت فیلم‌های توصیه‌شده درست را از همه فیلم‌های توصیه‌شده اندازه‌گیری می‌کند، در حالی که فراخوان نسبت فیلم‌های به‌درستی توصیه‌شده را اندازه‌گیری می‌کند.

از همه فیلم های مرتبط امتیاز F1 دقت و یادآوری را در یک متریک واحد ترکیب می کند. MAP میانگین دقت را در سطوح مختلف فراخوان محاسبه می کند و به ویژه در هنگام برخورد با توصیه های رتبه بندی شده مفید است.

محققان از این معیارهای ارزیابی در مطالعات توصیه فیلم استفاده کرده اند. به عنوان مثال، مطالعه ای بر روی الگوریتم های توصیه فیلم، دقت و یادآوری رویکردهای فیلترینگ مبتنی بر محتوا و مشارکتی را مقایسه کرد. نتایج ارزیابی نشان داد که فیلتر مشارکتی به دقت بالاتری دست یافت، در حالی که فیلتر مبتنی بر محتوا یادآوری بالاتری را نشان داد.

۸.۲ ارزیابی آفلاین در مقابل ارزیابی آنلاین(Offline Evaluation vs. Online Evaluation)

سیستم های توصیه را می توان با استفاده از روش های آفلاین و آنلاین ارزیابی کرد. ارزیابی آفلاین شامل استفاده از داده های تاریخی برای ارزیابی کیفیت توصیه می شود. این رویکرد امکان ارزیابی سریع و مقرون به صرفه را فراهم می کند، اما تعاملات کاربر را در زمان واقعی ثبت نمی کند. از سوی دیگر، ارزیابی آنلاین شامل انجام آزمایش‌هایی با کاربران واقعی، جمع‌آوری بازخورد و اندازه‌گیری تأثیر توصیه‌ها بر رفتار کاربر است.

یک تکنیک ارزیابی آنلاین معروف، تست A/B است که در آن دو یا چند الگوریتم توصیه با استفاده از کاربران واقعی مقایسه می‌شوند. به عنوان مثال، یک پلت فرم استریم ممکن است به طور تصادفی کاربران را به گروه‌های پیشنهادی مختلف اختصاص دهد و هر گروه را با الگوریتم‌های متفاوتی ارائه دهد. با اندازه‌گیری معیارهای تعامل، رضایت و تعامل کاربر، این پلتفرم می‌تواند عملکرد رویکردهای مختلف توصیه را ارزیابی کند و تصمیم‌های مبتنی بر داده اتخاذ کند.

۸.۳ چالش ها و ملاحظات در ارزیابی سیستم های توصیه(Challenges and Considerations in Evaluating Recommendation Systems)

ارزیابی سیستم های توصیه چالش های متعددی را به همراه دارد. یکی از چالش‌ها فقدان داده‌های حقیقت پایه است، زیرا ترجیحات کاربر ذهنی هستند و دائماً در حال تغییر هستند. برای پرداختن به این موضوع، محققان استفاده از داده‌های بازخورد ضمنی، مانند کلیک‌ها، زمان تماشا یا سابقه خرید را به عنوان یک پروکسی برای تنظیمات برگزیده کاربر پیشنهاد کرده‌اند.

معیارهای تنوع نیز در ارزیابی سیستم های توصیه بسیار مهم هستند. توصیه صرف فیلم های محبوب یا مشابه ممکن است منجر به عدم تنوع در پیشنهادها شود. معیارهایی مانند پوشش کاتالوگ و تازگی برای اندازه گیری تنوع توصیه ها استفاده می شود. برای مثال، معیار پوشش کاتالوگ ارزیابی می‌کند که سیستم چقدر فیلم‌هایی را از ژانرها و دسته‌های مختلف توصیه می‌کند.

علاوه بر این، ملاحظات انصاف در ارزیابی حیاتی است. سوگیری ممکن است در توصیه‌ها ایجاد شود و بر فیلم‌ها یا گروه‌های کاربری که کمتر نمایش داده شده‌اند تأثیر بگذارد. محققان روش‌ها و معیارهای ارزیابی مبتنی بر عدالت را برای ارزیابی عادلانه بودن سیستم‌های توصیه پیشنهاد کرده‌اند. این تضمین می‌کند که توصیه‌ها بی‌طرفانه هستند و منافع مختلف کاربران را برآورده می‌کنند.

۹. تجربه کاربر و ملاحظات اخلاقی

۹.۱ اهمیت تجربه کاربر در سیستم های توصیه(Importance of User Experience in Recommendation Systems)

تجربه کاربری یک جنبه حیاتی از سیستم های توصیه فیلم است. توصیه‌های دقیق و شخصی‌شده، رضایت و تعامل کاربر را افزایش می‌دهد و منجر به استفاده طولانی‌مدت و حفظ مشتری می‌شود. برای مثال، صفحات اصلی شخصی‌سازی‌شده و فهرست‌های پخش انتخاب‌شده، مانند آنچه توسط Amazon Prime Video ارائه می‌شود، با ارائه پیشنهادهای فیلم متناسب بر اساس اولویت‌های فردی و سابقه مشاهده، تجربه کاربر را بهبود می‌بخشد.

علاوه بر این، توصیه های مؤثر، کشف محتوای جدید و مرتبط را بهبود می بخشد. سیستم های توصیه با پیشنهاد فیلم هایی که مطابق با سلیقه کاربران هستند، در زمان و تلاش کاربران در جستجوی گزینه های مناسب صرفه جویی می کنند. این تجربه کلی کاربر را بهبود می‌بخشد و تضمین می‌کند که کاربران فیلم‌هایی را پیدا می‌کنند که واقعاً از آنها لذت می‌برند.

۹.۲ رسیدگی به مسائل مربوط به حریم خصوصی و امنیت داده ها(Addressing Issues of Privacy and Data Security)

سیستم‌های توصیه فیلم حجم زیادی از داده‌های کاربر را مدیریت می‌کنند و نگرانی‌هایی را در مورد حفظ حریم خصوصی و امنیت داده‌ها ایجاد می‌کنند. پلتفرم‌ها باید اعتماد کاربران را با اجرای اقدامات حفظ حریم خصوصی قوی در اولویت قرار دهند. اطمینان از رضایت کاربر، ناشناس کردن داده ها، و ارائه سیاست های شفاف استفاده از داده بسیار مهم است. پلتفرم‌هایی مانند Netflix و Amazon Prime Video از مقررات حفظ حریم خصوصی مانند مقررات حفاظت از داده‌های عمومی (GDPR) برای محافظت از اطلاعات کاربر پیروی می‌کنند.

امنیت داده ها به همان اندازه مهم است. سیستم های توصیه باید از پروتکل های ذخیره و دسترسی ایمن داده استفاده کنند تا از داده های کاربر در برابر دسترسی غیرمجاز محافظت کنند. تکنیک های رمزگذاری و کنترل های دسترسی دقیق به حفظ محرمانه بودن و یکپارچگی داده های کاربر کمک می کند.

۹.۳ برخورد با سوگیری های الگوریتمی و تنوع در توصیه ها(Dealing with Algorithmic Biases and Diversity in Recommendations)

سوگیری های الگوریتمی می تواند در توصیه های فیلم ظاهر شود و بر نمایش و تنوع تأثیر بگذارد. برای مثال، اگر یک سیستم توصیه در درجه اول فیلم‌هایی را پیشنهاد می‌کند که با ترجیحات اکثریت همخوانی دارند، ممکن است فیلم‌هایی را که به گروه‌های کم‌نمایش داده می‌شوند نادیده بگیرد.

پلتفرم‌ها با اجرای الگوریتم‌های آگاه از انصاف و در نظر گرفتن تنوع به طور فعال به این موضوع می‌پردازند.

معیارها در سیستم های توصیه آنها. هدف آنها ارائه پیشنهادهای عادلانه و متنوع فیلم است که ژانرها، فرهنگ ها و دیدگاه های مختلف را در بر می گیرد.

10.مدل معماری اجرا شده

۱. سیستم توصیه گر پیشنهادی

در این بخش، ما در اینجا معماری سیستم سیستم توصیه فیلم خود را بر اساس فناوری یادگیری عمیق و تجزیه و تحلیل داده های چندوجهی پیشنهاد می کنیم. این مدال ویژگی‌های ویژگی‌های کاربران و آیتم‌های موجود در مجموعه داده را بررسی می‌کند و آنها را در سیستم توصیه ادغام می‌کند، داده‌های امتیازدهی را ترکیب می‌کند، شبکه عصبی ساخته‌شده را آموزش می‌دهد، و در نهایت، امتیاز کاربر از فیلم را با دقت بیشتری پیش‌بینی می‌کند، که به طور قابل توجهی بهبود یافته است. در مقایسه با الگوریتم فیلتر مشترک سنتی.

۱۰.۱. چارچوب مدل پیشنهادی

مدل فرآیند کلی سیستم توصیه فیلم ما با یادگیری عمیق و داده های چندوجهی در شکل ۲ نشان داده شده است. ورودی شبکه مجموعه داده ای است که حاوی اطلاعات چندوجهی کاربران و فیلم ها است. خروجی یک لیست برتر از فیلم های توصیه شده برای کاربر است. ابتدا، پارامترهای کاربران و فیلم‌ها به ماتریس‌های تک مقداری تبدیل می‌شوند که حاوی مقادیر تکی غیر صفر هستند. دوم، CNN با چندین لایه از فیلترهای درهم پیچیده برای بهبود طبقه بندی سطح داده ها آموزش دیده است. سپس، سیستم توصیه با یک مدل آموزش دیده از ویژگی های تصفیه شده برای یافتن روابط بالقوه بین کاربران و فیلم ها بر اساس معیارهای شباهت استفاده می کند. شباهت های محتوا از طریق چندین مرحله از جمله حذف افزونگی، تخصیص امتیازات، عادی سازی و فیلتر کردن بیشتر اصلاح می شوند. در نهایت، بر اساس تئوری شباهت، فیلم های top-N برای کاربر توصیه می شود. توانایی منحصر به فرد در پردازش و پیوند دادن اطلاعات بر اساس روش های مختلف. بنابراین، یادگیری عمیق چندوجهی راه را برای ارائه بهتر از انواع مختلف داده‌های بدون ساختار استخراج می‌کند [۳۰]. اخیراً، برخی از افراد پیشنهاد کرده‌اند که ترکیب داده‌های چندوجهی مانند ویژگی‌های صدا، متن و تصویر، عملکرد سیستم‌های توصیه فیلم را بیشتر بهبود می‌بخشد [۲۱،۲۹–۳۲]. ۳. سیستم پیشنهادی پیشنهادی در این بخش، ما در اینجا معماری سیستم سیستم توصیه فیلم خود را بر اساس فناوری یادگیری عمیق و تجزیه و تحلیل داده های چندوجهی پیشنهاد می کنیم. این مدال ویژگی‌های ویژگی‌های کاربران و آیتم‌های موجود در مجموعه داده را بررسی می‌کند و آنها را در سیستم توصیه ادغام می‌کند، داده‌های امتیازدهی را ترکیب می‌کند، شبکه عصبی ساخته‌شده را آموزش می‌دهد، و در نهایت، امتیاز کاربر از فیلم را با دقت بیشتری پیش‌بینی می‌کند، که به طور قابل توجهی بهبود یافته است. در مقایسه با الگوریتم فیلتر مشترک سنتی.

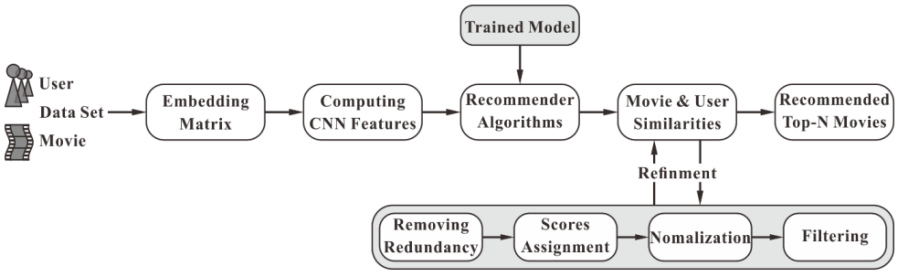


Figure 2. The framework of the movie recommendation system with deep learning.

۱۰.۲. استخراج ویژگی

در این کار حاضر، ما از CNN برای استخراج ویژگی‌های پنهان کاربران و فیلم‌ها از مجموعه داده‌های MovieLens استفاده می‌کنیم. CNN گونه ای از شبکه های عصبی پیشخور با سه بخش اصلی است، یعنی یک لایه کانولوشن که ویژگی های مختلف داده های ورودی را جدا و شناسایی می کند، یک لایه ادغام که داده ها را با انتخاب ویژگی های محلی نماینده از لایه کانولوشن قبلی متراکم می کند، و یک لایه کاملاً متصل. لایه. لایه کاملاً متصل ورودی را در یک ماتریس وزن ضرب می کند و سپس یک بردار بایاس اضافه می کند. در مقایسه با شبکه‌های عصبی سنتی، CNNها ممکن است حاوی صدها لایه پنهان باشند که عموماً برای کشف الگوها یا ویژگی‌های پیچیده در داده‌های پیچیده بدون مدل ریاضی مشخص مناسب هستند [۶،۳۳]. مدل CNN که در سیستم توصیه فیلم ما به کار می رود، یک نوع کوچک از معماری CNN است که توسط کولوبرت و همکاران پیشنهاد شده است. [۳۳]. از چهار بخش شامل لایه ورودی، لایه جمع‌آوری، لایه کانولوشن و لایه خروجی تشکیل شده است (شکل ۳). لایه ورودی داده های خام را به یک ماتریس عددی متراکم ۳۲×۳۲ تبدیل می کند که نشان دهنده داده های کانولوشن بعدی است. سه لایه پیچیدگی برای استخراج ویژگی‌های متنی داده‌های ورودی از مجموعه داده MovieLens استفاده می‌شود که به‌صورت شش لایه با ابعاد ۳۲ × ۳۲ (۶ @ ۳۲ × ۳۲)، ۱۶ لایه با ابعاد ۱۰ × ۱۰ (۱۶ @ ۱۰ × ۱۰) طراحی شده‌اند. و به ترتیب ۱۲۰ لایه. دو لایه نظرسنجی، که به عنوان ۶ @ ۱۴ × ۱۴ و ۱۶ @ ۵ × ۵ تنظیم شده است، برای استخراج ویژگی های نماینده از لایه های کانولوشن استفاده شد. در نهایت، لایه خروجی ۱۰ توصیه برتر را برای برنامه ایجاد می کند. در عمل، چهار پارامتر (شناسه فیلم، نوع، عنوان و پوستر) فیلم‌ها و چهار پارامتر (شناسه کاربر، جنسیت، سن و حرفه) کاربران را به‌عنوان داده‌های ورودی برای تولید ماتریس‌های اولیه برای فرآیندهای استخراج ویژگی بعدی انتخاب کردیم (شکل ۴).

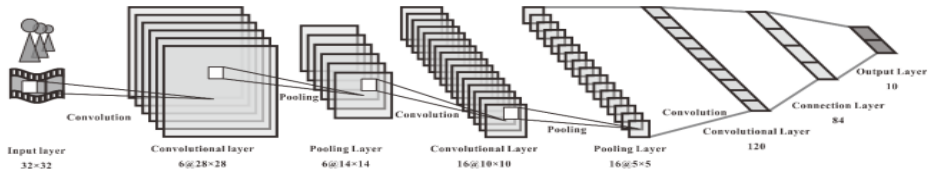


Figure 3. The CNN architecture in the proposed movie recommendation system.

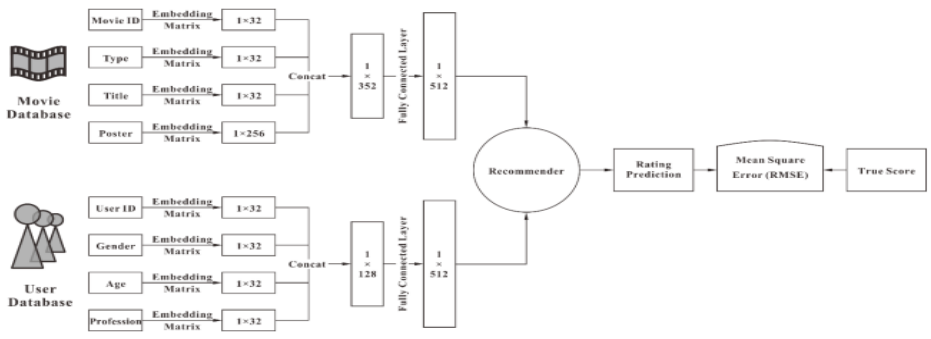


Figure 4. Score prediction model of the movie recommendation system based on neural network.

الف. منظم سازی

منظم سازی در یادگیری عمیق، و به طور کلی در یادگیری ماشین، مفهوم مهمی است که مشکل بیش از حد برازش را حل می کند. اجرای منظم سازی در حین آموزش یک مدل خوب بسیار مهم است، زیرا این تکنیکی است که در تلاش برای حل مشکل اضافه برازش استفاده می شود. همانطور که قبلا ذکر شد، منظم سازی تلاشی برای اصلاح بیش از حد برازش مدل با معرفی اطلاعات اضافی به تابع هزینه است. در زمینه رگرسیون خطی حداقل مربعات، عبارت منظم سازی به تابع هزینه رگرسیون خطی حداقل مربعات استاندارد J اضافه می شود که در زیر تعریف شده است.

$$J(\Theta) = \frac{1}{2}m\ [\sum_{i=1}^{m}(h_\Theta(x^i) - y^i)^2 + \lambda \sum_{j=1}^{n} \Theta_j^2\ ] \qquad (2)$$

که در آن Θ مقادیر پارامتر است، m تعداد مثال‌های آموزشی با n ویژگی مختلف، hΘ(xi) مقدار hΘ تخمین‌گر برای مثال آموزشی i، yi مقدار برچسب‌گذاری شده واقعی مثال آموزشی i، و λ منظم‌سازی است. ثابت. در بحث منظم‌سازی، از منظم‌سازی L2 استفاده کرده‌ایم، در حالی که منظم‌سازی L1 یکی دیگر از این استراتژی‌ها برای کنترل بیش‌برازش است. این دو قانونمندسازی هدف یکسانی دارند اما از چند جنبه کلیدی با هم تفاوت دارند. توجه داشته باشید که در معادله ۲،

$$\lambda \sum_{j=1}^{n} \Theta_j^2$$

عبارت منظم سازی L2 است، در حالی که در L1، همان عبارت منظم سازی به عنوان نوشته می شود:

$$\lambda \sum_{j=1}^{n} |\Theta_j| \qquad (4)$$

از این رو، تفاوت بین L1 و L2 در این است که L2 از مجموع مجذور پارامترها استفاده می کند، در حالی که L1 مجموع قدر مطلق پارامترها است. در اصل، منظم‌سازی L1 برخی از پارامترهای مرتبط با یک ویژگی معین را به صفر کاهش می‌دهد، در حالی که منظم‌سازی L2 پارامترهای ویژگی را صفر نمی‌کند، بلکه فقط به کاهش مقدار یک Θ معین ادامه می‌دهد.

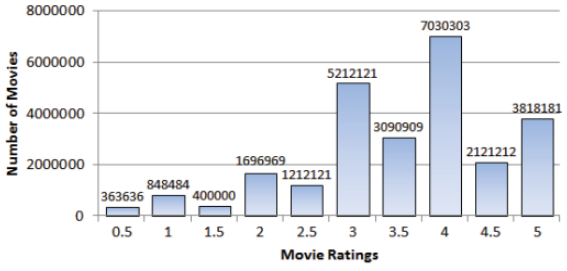


Figure 3. Distribution of ratings in the full MovieLens dataset

11.تئوری کد پروژه:

1. داده ها و فرضیات اولیه

۱ .1 مقدمه

مدل‌های فیلتر مشارکتی (CF) با هدف بهره‌برداری از اطلاعات مربوط به ترجیحات کاربران برای موارد (مانند رتبه‌بندی ستاره‌ها) برای ارائه توصیه‌های شخصی‌سازی شده است. با توجه به چالش نتفلیکس، مجموعه ای از مدل های مختلف CF پیشنهاد شده است، با انتخاب های محبوب فاکتورسازی ماتریس و مدل های همسایگی. این مقاله AutoRec، یک مدل CF جدید بر اساس پارادایم رمزگذار خودکار را پیشنهاد می‌کند. علاقه ما به این پارادایم از موفقیت‌های اخیر مدل‌های شبکه عصبی (عمیق) برای وظایف بینایی و گفتاری ناشی می‌شود. ما استدلال می‌کنیم که AutoRec دارای مزایای نمایشی و محاسباتی نسبت به رویکردهای عصبی موجود در CF است، و به‌طور تجربی نشان می‌دهد که از روش‌های پیشرفته فعلی بهتر عمل می‌کند.

۱ .2 مرور مدل Autorec

در فیلتر مشارکتی مبتنی بر رتبه‌بندی، ما m کاربر، n آیتم و یک ماتریس رتبه‌بندی کاربر مورد مشاهده شده $R \in R^{m \times n}$ داریم. هر کاربر $u \in U = \{1 \ldots m\}$ را توسط یک بردار نیمه مشاهده شده(partially observed) می توان نشان داد: $r^{(u)} = (R_{u1}, \ldots, R_{un}) \ni R^n$.

به طور مشابه، هر آیتم $i \in I = \{1 \ldots n\}$ توسط یک بردار نیمه مشاهده شده قابل نمایش است :

$r^{(i)} = (R_{1i}, \ldots R_{mi}) \in R^m$

هدف ما در این کار طراحی یک آیتم محور (مبتنی بر کاربر) است. رمزگذار خودکار(autoencoder) که می تواند به عنوان ورودی هر کدام نیمه مشاهده شود: $r^{(u)}$ ($r^{(i)}$، آن را در یک فضای لیتنت (پنهان) با ابعاد کم پخش کند و سپس $r^{(u)}$ ($r^{(i)}$ در فضای خروجی برای پیش‌بینی رتبه‌بندی‌های گمشده برای اهداف توصیه بازسازی کند. به طور رسمی، با توجه به مجموعه S از بردارها در R^d و مقداری $k \in N_+$ رمزگذار خودکار حل می کند:

$$\min_{\theta} \sum_{\mathbf{r} \in \mathbf{S}} ||\mathbf{r} - h(\mathbf{r}; \theta)||_2^2, \qquad (1)$$

Figure 1: Item-based AutoRec model. We use plate notation to indicate that there are n copies of the neural network (one for each item), where $\mathbf{W}$ and $\mathbf{V}$ are tied across all copies.

where $h(\mathbf{r}; \theta)$ is the *reconstruction* of input $\mathbf{r} \in \mathbb{R}^d$,

$$h(\mathbf{r}; \theta) = f\left(\mathbf{W} \cdot g(\mathbf{V}\mathbf{r} + \boldsymbol{\mu}) + \mathbf{b}\right)$$

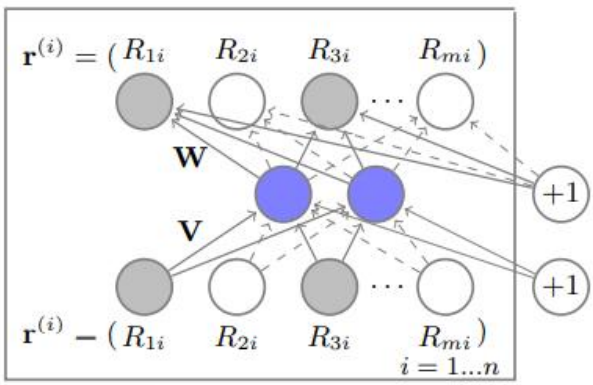


Fig1

مدل AutoRec مبتنی بر آیتم، که در شکل ۱ نشان داده شده است، یک رمزگذار خودکار را مطابق با معادله ۱ به مجموعه بردارها اعمال می کند.$\{r^{(i)}\}_{i=1}^{n}$ ، با دو تغییر مهم اول، ما این واقعیت را حساب می کنیم که هر $r^{(i)}$ تنها در مدت به‌روزرسانی backpropagation آن دسته از وزن‌هایی که با ورودی‌های نیمه مشاهده شده مرتبط هستند، دیده می‌شود، همانطور که در روش‌های فاکتورسازی ماتریس و RBM رایج است.

دوم، ما پارامترهای آموخته شده را منظم می کنیم تا از تطبیق بیش از حد در رتبه های مشاهده شده جلوگیری کنیم. به طور رسمی، تابع هدف برای مدل AutoRec مبتنی بر آیتم (I-AutoRec) برای قدرت منظم‌سازی $\lambda > 0$ است.

$$\min_{\theta} \sum_{i=1}^{n} ||\mathbf{r}^{(i)} - h(\mathbf{r}^{(i)}; \theta))||_{\mathcal{O}}^2 + \frac{\lambda}{2} \cdot (||\mathbf{W}||_F^2 + ||\mathbf{V}||_F^2), \quad (2)$$

where $||\cdot||_{\mathcal{O}}^2$ means that we only consider the contribution of observed ratings.

AutoRec مبتنی بر کاربر (U-AutoRec) از کار با $\{r^{(u)}\}_{u=1}^{m}$ مشتق می شود.

در مجموع، I-AutoRec به تقریب پارامترهای $2mk + m + k$ نیاز دارد. با توجه به پارامترهای آموخته شده $\hat{\theta}$، امتیاز پیش بینی شده I-AutoRec، برای کاربر u و آیتم i بدین شکل است:

$$\hat{R}_{ui} = (h(\mathbf{r}^{(i)}; \hat{\theta}))_u. \quad (3)$$

شکل ۱ مدلی را نشان می دهد، با گره های سایه دار مربوط به امتیاز های مشاهده شده، و اتصالات ثابت و محکم مربوط به وزن هایی که برای ورودی $r^{(i)}$ به روز می شوند.

۲.جدول مقایسه متد ها با روش Autorec

	ML-1M	ML-10M
U-RBM	0.881	0.823
I-RBM	0.854	0.825
U-AutoRec	0.874	0.867
I-AutoRec	**0.831**	**0.782**

(a)

$f(\cdot)$	$g(\cdot)$	RMSE
Identity	Identity	0.872
Sigmoid	Identity	0.852
Identity	Sigmoid	**0.831**
Sigmoid	Sigmoid	0.836

(b)

	ML-1M	ML-10M	Netflix
BiasedMF	0.845	0.803	0.844
I-RBM	0.854	0.825	-
U-RBM	0.881	0.823	0.845
LLORMA	0.833	**0.782**	0.834
I-AutoRec	**0.831**	**0.782**	**0.823**

(c)

Table 1: (a) Comparison of the RMSE of I/U-AutoRec and RBM models. (b) RMSE for I-AutoRec with choices of linear and nonlinear activation functions, Movielens 1M dataset. (c) Comparison of I-AutoRec with baselines on MovieLens and Netflix datasets. We remark that I-RBM did not converge after one week of training. LLORMA's performance is taken from [2].

۳.مقایسه کلی بین Autorec با RBM ها

AutoRec با روشهای موجود CF متمایز است. در مقایسه با مدل CF مبتنی بر (RBM-CF) RBM، چندین تفاوت وجود دارد. اول، RBM-CF یک مدل مولد احتمالی مبتنی بر ماشین‌های محدود بولتزمن (RBM)پیشنهاد می‌کند، در حالی که AutoRec یک مدل متمایز مبتنی بر رمزگذارهای خودکار است. دوم، RBM-CF پارامترها را با به حداکثر رساندن احتمال ورود تخمین می‌زند، در حالی که AutoRec مستقیماً RMSE، عملکرد متعارف در وظایف پیش‌بینی رتبه‌بندی را به حداقل می‌رساند. سوم، آموزش RBM-CF نیاز به استفاده از واگرایی کنتراست(contrastive divergenc) دارد، در حالی که آموزش AutoRec نیاز به انتشار پس‌انداز نسبتاً سریع‌تر مبتنی بر گرادیان(gradient-based backpropagation) دارد. در نهایت، RBM-CF فقط برای رتبه‌بندی‌های گسسته قابل استفاده است و مجموعه‌ای از پارامترها را برای هر مقدار رتبه‌بندی تخمین می‌زند. برای رتبه‌بندی r ممکن، این به پارامترهای nkr یا (mkr) برای RBM مبتنی بر کاربر (آیتم) اشاره دارد. AutoRec نسبت به r آگنوستیک است و از این رو به پارامترهای کمتری نیاز دارد. پارامترهای کمتر AutoRec را قادر می‌سازد تا حافظه کمتری داشته باشد و کمتر مستعد بیش از حد برازش باشد. در مقایسه با رویکردهای فاکتورسازی ماتریسی (MF)، که هم کاربران و هم موارد را در یک فضای پنهان مشترک جاسازی می‌کند، مدل AutoRec مبتنی بر آیتم فقط موارد را در فضای پنهان جاسازی می‌کند. علاوه بر این، در حالی که MF یک نمایش نهفته خطی را می آموزد، AutoRec می تواند یک نمایش نهفته غیرخطی را از طریق تابع فعال سازی g(·) یاد بگیرد.

۴.سوءالات متداول و اساسی

کدام بهتر است کدگذاری خودکار مبتنی بر آیتم یا کاربر، با RBM یا AutoRec ؟

جدول a۱ روش های مبتنی بر آیتم (I-) را برای RBM و AutoRec نشان می دهد که عموماً عملکرد بهتری دارند. این احتمال وجود دارد زیرا میانگین تعداد رتبه‌بندی‌ها در هر مورد بسیار بیشتر از رتبه‌بندی‌های هر کاربر است. واریانس زیاد در تعداد رتبه‌بندی‌های کاربر منجر به پیش‌بینی کمتر قابل اعتماد برای روش‌های مبتنی بر کاربر می‌شود. I-AutoRec از همه انواع RBM بهتر عمل می کند.

چگونه عملکرد AutoRec با توابع فعال سازی خطی و غیرخطی f(·)، g(·) متفاوت است؟

جدول b۱ نشان می دهد که غیرخطی بودن در لایه پنهان (از طریق g(·)) برای عملکرد خوب I-AutoRec حیاتی است، که نشان دهنده آن است که مزیت بالقوه نسبت به روش های MF جایگزینی سیگموئیدها با واحدهای خطی اصلاح شده (ReLU) عملکرد بدتری داشت. تمام آزمایش‌های AutoRec دیگر از توابع هویت f(·) و g(·) sigmoid استفاده می‌کنند.

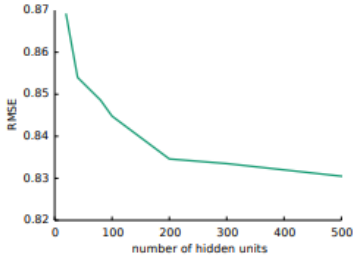


Figure 2: RMSE of I-AutoRec on Movielens 1M as the number of hidden units k varies.

چگونه عملکرد AutoRec با تعداد واحدهای پنهان متفاوت است؟

در شکل ۲، عملکرد مدل AutoRec را با توجه به اینکه تعداد واحدهای پنهان متفاوت است، ارزیابی می کنیم. توجه می کنیم که عملکرد به طور پیوسته با تعداد واحدهای پنهان افزایش می یابد، اما با کاهش بازده. تمام آزمایش‌های AutoRec دیگر از k = 500 استفاده می‌کنند.

AutoRec چگونه در برابر همه خطوط پایه عمل می کند؟

جدول c۱ نشان می دهد که AutoRec به طور مداوم از همه خطوط پایه بهتر عمل می کند، به جز نتایج قابل مقایسه با LLORMA در Movielens 10M. عملکرد رقابتی با LLORMA مورد توجه است، زیرا دومی شامل وزن دهی ۵۰ مدل مختلف فاکتورسازی ماتریس محلی است، در حالی که AutoRec تنها از یک نمایش نهفته از طریق رمزگذار خودکار شبکه عصبی استفاده می کند.

آیا پسوندهای عمیق AutoRec کمک می کند؟

ما یک نسخه عمیق از I-AutoRec را با سه لایه پنهان (۵۰۰، ۲۵۰، ۵۰۰) واحد توسعه دادیم که هر یک دارای یک فعال سازی سیگموئید است. ما از پیش‌آموزشی حریصانه استفاده کردیم و سپس با نزول گرادیان به‌خوبی تنظیم کردیم. در Movielens 1M، RMSE از ۰/۸۳۱ به ۰/۸۲۷ کاهش می‌یابد که نشان‌دهنده پتانسیل بهبود بیشتر از طریق AutoRec عمیق است.

12. بخش عملی کد پروژه

که در دو بخش کد ارائه می شود ۱-preprocess و ۲-autorec(autoencoder)

۱.۲.پیش پردازش و تبدیل آن به ماتریس کوچکتر

معرفی داده ها:

برای سیستم توصیه گر از مجموعه داده MovieLens 20M استفاده شده است. این مجموعه داده شامل ۲۰ میلیون رتبه بندی و ۴۶۵۰۰۰ برنامه تگ اعمال شده بر روی ۲۷۰۰۰ فیلم توسط ۱۳۸۰۰۰ کاربر. شامل داده های ژنوم تگ با ۱۲ میلیون امتیاز مرتبط در ۱۱۰۰ تگ است.

از این داده ها می توان فهمید که داده ها برای پردازش و ذخیره بسیار سنگین بوده! و برای همین بعد از اجرای اولیه برای رسم نمودار ها مجبور شده داده ها را یک پیش پردازش دیگر بر اساس کد زیر کرده تا برای پیش بینی و نتیجه گیری اجرای بهتری را مشاهده کنیم.

کتابخانه ها ضروری:

```
import pickle
import numpy as np
import pandas as pd
from collections import Counter
```

در کتابخانه بالا دو مورد بسیار اساسی و ضروری هستند به نام های pickle و collection که از

Pickle یک ماژول در پایتون است که راهی برای سریال‌سازی و سریال‌زدایی object پایتون ارائه می‌دهد. این به ما این امکان را می دهد که object ها پیچیده مانند مدل های یادگیری ماشین را به یک جریان بایت تبدیل کنیم که می تواند در یک فایل ذخیره شود یا از طریق شبکه منتقل شود. Pickling زمانی مفید است که بخواهیم وضعیت یک object را ذخیره کنیم و بعداً بدون نیاز به آموزش مجدد مدل از آن دوباره استفاده کنیم.

و Collections یک ماژول در پایتون است که کانتینر ها datatype خاص را ارائه می دهد. یکی از کلاس های رایج در ماژول Collections ، Counter است که یک دیکشنری ساب کلاس است که اشیاء قابل هش را شمارش می کند. برای ذخیره عناصر به عنوان کلیدهای دیکشنری و شمارش آنها به عنوان مقادیر دیکشنری استفاده می شود. شمارنده مخصوصاً زمانی مفید است که بخواهیم فراوانی عناصر را در یک لیست یا سایر موارد تکرار شونده محاسبه کنیم.

خواندن دیتا و کوچک کردن آن:

```
# Load in the data
# https://www.kaggle.com/grouplens/movielens-20m-dataset
df = pd.read_csv("C://Users//Admin//Documents//GitHub//edit_rating.csv")
print("original dataframe size:", len(df))

N = df.userId.max() + 1 # number of users
M = df.movie_idx.max() + 1 # number of movies

user_ids_count = Counter(df.userId)
movie_ids_count = Counter(df.movie_idx)

# number of users and movies we would like to keep
n = 10000
m = 2000

user_ids = [u for u, c in user_ids_count.most_common(n)]
movie_ids = [m for m, c in movie_ids_count.most_common(m)]

# make a copy, otherwise ids won't be overwritten
df_small = df[df.userId.isin(user_ids) & df.movie_idx.isin(movie_ids)].copy()
```

```
original dataframe size: 20000263
```

در قطعه کد نشان داده شده pickle برای ذخیره یا بارگیری بالقوه مدل‌های یادگیری ماشین وارد می‌شود، در حالی که collections.Counter برای شمارش تعداد شناسه‌های کاربر و شناسه‌های فیلم در مجموعه داده استفاده می‌شود. هدف از شمارش رخدادها انتخاب زیرمجموعه ای از کاربران و فیلم ها بر اساس فراوانی آنهاست، همانطور که با متغیرهای 'n' و 'm' نشان داده شده است. زیر مجموعه حاصل از دیتافریم در df_small برای پردازش بیشتر ذخیره می شود.

ابتدا، کد فایل CSV واقع در آدرس نشان داده شده را با استفاده از تابع pd.read_csv() از کتابخانه pandas می خواند. داده های بارگذاری شده در یک شی DataFrame pandas به نام df ذخیره می شود.

در ادامه، کد حداکثر شناسه کاربر (N) و حداکثر شاخص فیلم (M) موجود در DataFrame را محاسبه می کند. این مقادیر برای تعیین تعداد کل کاربران و فیلم ها در مجموعه داده استفاده می شود. سپس کد دو شیء Counter به نام‌های user_ids_count و movie_ids_count ایجاد می‌کند که به ترتیب تعداد هر شناسه کاربر و فهرست فیلم را در DataFrame شمارش می‌کنند. شناسه های کاربری انتخاب شده در لیستی به نام user_ids و شاخص های فیلم انتخاب شده در لیستی به نام movie_ids ذخیره می شوند.

در نهایت، کد یک DataFrame جدید به نام df_small با اعمال دو شرط برای DataFrame df اصلی ایجاد می کند. فقط ردیف هایی را انتخاب می کند که شناسه کاربری در لیست user_ids و فهرست فیلم در لیست movie_ids قرار دارد. از متد copy() برای ایجاد یک کپی از سطرهای انتخاب شده استفاده می شود تا تغییرات ایجاد شده در df_small روی DataFrame اصلی تاثیری نداشته باشد.

منطق کلی این کد بارگذاری مجموعه داده امتیازدهی فیلم، تعیین فعال ترین کاربران و فیلم های محبوب بر اساس فراوانی رتبه بندی آنها و ایجاد زیرمجموعه کوچکتری از مجموعه داده است که فقط شامل کاربران و فیلم های انتخاب شده است. این زیرمجموعه کوچکتر را می توان برای تحلیل بیشتر یا کارهای مدلسازی، کاهش نیازهای محاسباتی و تمرکز بر مرتبط ترین داده ها استفاده کرد.

مرتب سازی و اصلاح فرم داده ها به حالت اولیه

```python
# need to remake user ids and movie ids since they are no longer sequential
new_user_id_map = {}
i = 0
for old in user_ids:
  new_user_id_map[old] = i
  i += 1
print("i:", i)

new_movie_id_map = {}
j = 0
for old in movie_ids:
  new_movie_id_map[old] = j
  j += 1
print("j:", j)

print("Setting new ids")
df_small.loc[:, 'userId'] = df_small.apply(lambda row: new_user_id_map[row.userId], axis=1)
df_small.loc[:, 'movie_idx'] = df_small.apply(lambda row: new_movie_id_map[row.movie_idx], axis=1)
# df_small.drop(columns=['userId', 'movie_idx'])
# df_small.rename(index=str, columns={'new_userId': 'userId', 'new_movie_idx': 'movie_idx'})
print("max user id:", df_small.userId.max())
print("max movie id:", df_small.movie_idx.max())

print("small dataframe size:", len(df_small))
df_small.to_csv('C://Users//Admin//Documents//GitHub//small_rating.csv', index=False)

```

```
i: 10000
j: 2000
Setting new ids
max user id: 9999
max movie id: 1999
small dataframe size: 5392025
```

کد با مقداردهی اولیه یک دیکشنری خالی new_user_id_map و یک شمارنده i برای پیگیری شناسه های کاربر جدید شروع می شود. سپس، از طریق هر شناسه کاربری قدیمی در لیست user_ids حلقه می زند. برای هر شناسه کاربری قدیمی، یک شناسه کاربری جدید i در فرهنگ لغت new_user_id_map اختصاص می‌دهد و i را ۱ واحد افزایش می‌دهد. این فرآیند برای همه شناسه‌های کاربر در مجموعه داده تکرار می‌شود. به طور مشابه، کد یک دیکشنری خالی new_movie_id_map و یک شمارنده j را برای پیگیری شناسه های فیلم جدید مقداردهی اولیه می کند. از طریق هر شناسه فیلم قدیمی در لیست movie_ids

حلقه می‌زند و یک شناسه فیلم جدید j را در فرهنگ لغت new_movie_id_map اختصاص می‌دهد، و برای هر تکرار j را ۱ واحد افزایش می‌دهد.

پس از ایجاد نگاشت ها، کد برای به روز رسانی شناسه های کاربر و شناسه فیلم در دیتافریم df_small اقدام می کند. از تابع apply همراه با یک تابع lambda برای جایگزینی هر شناسه کاربری قدیمی در ستون 'userId' با شناسه جدید مربوطه آن از دیکشنری new_user_id_map استفاده می کند. به طور مشابه، هر شناسه فیلم قدیمی در ستون 'movie_idx' را با شناسه جدید مربوطه از فرهنگ لغت new_movie_id_map جایگزین می‌کند. در مرحله بعد، کد حداکثر شناسه کاربر و شناسه فیلم را در دیتافریم به روز شده df_small چاپ می کند تا بررسی کند که آیا نگاشت مجدد موفقیت آمیز بوده است یا خیر.

در نهایت، کد اندازه df_small dataframe را چاپ می کند و آن را در یک فایل CSV به نام 'small_rating.csv' در دایرکتوری مشخص شده ذخیره می کند.

هدف پشت این کار ایجاد نگاشت های جدید بین شناسه های کاربری قدیمی و شناسه فیلم و شناسه های جدید مربوط به آنهاست. این کار با استفاده از دو دیکشنری new_user_id_map و new_movie_id_map انجام می شود که نگاشت ها را ذخیره می کند.

و همچنین برای اطمینان از اینکه شناسه های کاربر و شناسه های فیلم در مجموعه داده متوالی و سازگار هستند. این می تواند برای کارهای مختلفی مانند ایندکس گذاری، تجزیه و تحلیل داده ها و مدل سازی مهم باشد. با نگاشت مجدد شناسه ها، کد تضمین می کند که مجموعه داده به درستی ساختار یافته و برای پردازش بیشتر آماده است.

۱.۲.پیش پردازش و تبدیل آن به ماتریس اسپارس

یک جنبه مهم از داده ها این حقیقت است که داده ها امتیاز دهی بسیار خلوت هستند(sparse). این بدین دلیل است که هر کاربر

فقط به بخش کوچکی از فیلم ها امتیاز داده است.بنابراین ، اگر ماتریس کامل با ۱۰۰۰۰۰ سطر مربوط به کاربران ساخته شود و ۹۰۰۰ستون مربوط به فیلم ها وجود داشته باشد، بیشتر مقادیر "Nan" هستند.

خواندن دیتا و کتابخانه ها:

```
from __future__ import print_function, division
from builtins import range, input

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.utils import shuffle
from scipy.sparse import lil_matrix, csr_matrix, save_npz, load_npz

# load in the data
df = pd.read_csv("C:\\Users\\Admin\\Documents\\GitHub\\edit_rating.csv")

```

- **نکته**:دو خط اول برای سازگار بودن با نسخه ها قدیمی تر پایتون نوشته شدند.(2.x)

Pandas,numpy: این دو کتابخانه معروف برای خواندن داده و انجام محاسبات روی آنها استفاده می شوند.

matplotlib: این کتابخانه معمولا برای تجسم داده ها در پایتون استفاده می شود. این طیف گسترده ای از توابع رسم و یک رابط مانند MATLAB برای ایجاد انواع مختلف چارتها، نمودارها و پلات کردن نمودارها گسسته را فراهم می کند. که در کد، matplotlib برای رسم منحنی خطا و میانگین مربعات خطا استفاده شده است.

sklearn.utils: این ماژول از کتابخانه scikit-learn توابع مختلف کاربردی را برای دستکاری و پیش پردازش داده ها ارائه می دهد. در کد، تابع shuffle از sklearn.utils برای به هم زدن مجموعه داده ها استفاده می شود.

scipy.sparse: این ماژول از کتابخانه SciPy قابلیت کار با ماتریس های اسپارس را ارائه می دهد که ماتریس هایی با تعداد زیادی عنصر صفر هستند. ماتریس های اسپارس در مواقع وجود مجموعه داده های بزرگ از نظر حافظه کارآمدتر هستند. در کد، از توابع lil_matrix، csr_matrix، save_npz و load_npz از scipy.sparse برای مدیریت و ذخیره ماتریس های اسپارس استفاده می شود.

این کتابخانه ها طیف وسیعی از قابلیت ها را برای تجزیه و تحلیل داده ها، تجسم، یادگیری ماشینی و محاسبات علمی ارائه می دهند. با استفاده از این کتابخانه ها، ما می توانیم وظایفی مانند دستکاری داده ها، تجزیه و تحلیل داده های اکتشافی، آموزش و ارزیابی مدل و تجسم نتایج را انجام دهیم.

بارگذاری داده: در نهایت فایل به صورت جدول csv فایل را که به حالت csv ادیت شده را میخواند و در df ذخیره می کند.

بخش دوم: train test split

```
N = df.userId.max() + 1 # number of users
M = df.movie_idx.max() + 1 # number of movies

# split into train and test
df = shuffle(df)
cutoff = int(0.8*len(df))
df_train = df.iloc[:cutoff]
df_test = df.iloc[cutoff:]

A = lil_matrix((N, M))
print("Calling: update_train")
count = 0
def update_train(row):
  global count
  count += 1
  if count % 100000 == 0:
    print("processed: %.3f" % (float(count)/cutoff))

  i = int(row.userId)
  j = int(row.movie_idx)
  A[i,j] = row.rating
df_train.apply(update_train, axis=1)

```

در بخش دوم، در دو خط اول تعداد حداکثر فیلم ها و کاربران را مشخص کردیم و بطوریکه اول، بزرگترین ستون ID را یافته و بعد یکی به آن اضافه کرده تا تعداد کل به دست آِید.

تابع shuffle از ماژول sklearn.utils برای به هم زدن تصادفی ردیف های DataFrame df استفاده می شود. برای اطمینان از تصادفی بودن داده ها و جلوگیری احتمالی از هرگونه متمایل شدن به یک جهت خاص داده ها در ترتیب مجموعه داده ها انجام می شود.

سپس در قسمت تقسیم داده ها متغیر cutoff روی ۸۰ درصد طول DataFrame df تنظیم شده است. این مقدار نشان دهنده شاخصی است که در آن مجموعه داده به زیر مجموعه های آموزشی و آزمایشی تقسیم می شود.

(پس ۸۰ درصد داده هایمان را برای آموزش و ۲۰ درصد مابقی را برای تست قرار دادیم)

df_train = df.iloc[:cutoff]

df_train با انتخاب اولین تعداد سطرها از df ایجاد می شود. از این ردیف ها برای آموزش سیستم توصیه گر استفاده می شود.

[:cutoff]df_test = df.iloc

df_test با انتخاب سطرها از cutoff به بعد در df ایجاد می شود. از این ردیف ها برای آزمایش سیستم توصیه گر استفاده می شود.

A = lil_matrix ((N, M))

متغیر A به عنوان یک ماتریس اسپارس به اندازه N x M با استفاده از تابع lil_matrix از ماژول scipy.sparse مقداردهی اولیه می شود. یک ماتریس اسپارس روشی کارآمد از نظر حافظه برای ذخیره ماتریس های بزرگ با مقادیر عمدتاً صفر است.

در این مورد، A برای نشان دادن ماتریس امتیاز دهی استفاده می شود، که در آن هر سطر مربوط به یک کاربر و هر ستون مربوط به یک فیلم است.

:def update_train(row)

یک ردیف را به عنوان ورودی می گیرد و در هر دفعه صدا شدن متغیر سراسری count را یک واحد افزایش میدهد و اگر پروسه از 100,000 عبور کرد نمایش شماره updating را نشان میدهد.

و سپس تغییرات زیر را روی درایه ها می دهند:

rating.row = A[i,j]

(i شناسه کاربری را از ستون userId ورودی j فهرست فیلم را از ستون movie_idx ردیف ورودی استخراج می کند.)

این خط مقدار امتیازات را از ستون rating ردیف ورودی به موقعیت (i, j) در ماتریس A اختصاص می دهد. این مرحله ماتریس امتیازات را با امتیازات کاربران برای فیلم‌های مختلف پر می‌کند.

هدف کلی این قسمت کد این است که داده‌ها را از پیش پردازش کرده، آن‌ها را به مجموعه‌های آموزشی و آزمایشی تقسیم می‌کند و یک نمایش ماتریس اسپارس (A) برای ذخیره امتیازات کاربران برای فیلم‌های مختلف ایجاد می‌کند. تابع update_train برای پر کردن ماتریس امتیازات در هر ردیف از مجموعه داده آموزشی اعمال می‌شود.

```
# mask, to tell us which entries exist and which do not
A = A.tocsr()
mask = (A > 0)
save_npz("Atrain.npz", A)

# test ratings dictionary
A_test = lil_matrix((N, M))
print("Calling: update_test")
count = 0
def update_test(row):
  global count
  count += 1
  if count % 100000 == 0:
    print("processed: %.3f" % (float(count)/len(df_test)))

  i = int(row.userId)
  j = int(row.movie_idx)
  A_test[i,j] = row.rating
df_test.apply(update_test, axis=1)
A_test = A_test.tocsr()
mask_test = (A_test > 0)
save_npz("Atest.npz", A_test)
```

بخش سوم: تشخیص اسپارسیتی درایه ها ماتریس

اول آرایه A مان را با متد ()tocsr (Compressed Sparse Row)به شکل فشرده تر و بهینه تری در حافظه ذخیره می کنیم (زیرا بیشتر اطلاعات Nan هستند!)

mask = (A > 0)

ماسک به‌عنوان یک ماتریس بولی ایجاد می‌شود که نشان‌دهنده وجود امتیازات در A است. اگر عنصر مربوطه در A بزرگتر از 0 باشد (به عنوان مثال، اگر امتیازدهیی وجود داشته باشد) هر عنصر در ماتریس ماسک روی True تنظیم می‌شود.

از این ماتریس ماسک برای انتخاب ورودی های معتبر از A در طول محاسبات استفاده می کنیم.

سپس ماتریس A را که بصورت فشرده کرده بودیم، در قالب فایل فشرده NumPy .npz با استفاده از تابع save_npz() از ماژول scipy.sparse ذخیره می کنیم. این اجازه می دهد تا ماتریس امتیازدهی به طور موثر ذخیره شود و بعداً در صورت نیاز بارگذاری شود.

و بار دیگر یک ماتریس اسپارس دیگر A_test با استفاده از تابع lil_matrix از ماژول scipy.sparse مقداردهی اولیه می شود. این ماتریس برای ذخیره امتیازات برای مجموعه داده آزمایشی استفاده خواهد شد. در ادامه هم پیام موفقیت آمیز به عنوان "update-test" در خروجی نمایش داده می شود.

و همان مراحلی که برای قسمت آموزش (train) انجام دادیم را مشابهش انجام میدهیم.

این تابعی به نام update_test را تعریف می کنیم که یک ردیف را به عنوان ورودی می گیرد. برای هر ردیف در df_test با استفاده از df_test.apply() اعمال خواهد شد.

از Global count برای متغیر count در تابع update_test به متغیر شمارش سراسری تعریف شده خارج از تابع اشاره دارد. و از count += ۱ برای افزایش متغیر count را ۱ استفاده میکنیم.

تا اگر count ٪ ۱۰۰۰۰۰ == ۰: درست باشد و باقیمانده نداشته باشد، به این معنی است که ۱۰۰۰۰۰ ردیف پردازش شده است و پیام به روز رسانی پیشرفت را چاپ می کند.

سپس دو متغیر i,j را بدین صورت تعریف می کنیم:که i id کاربر را از ستون userId از ردیف ورودی و j فهرست فیلم را از ستون movie_idx از ردیف ورودی استخراج می کند.

A_test[i,j] = row.rating:

و سپس این خط مقدار امتیازات را از ستون rating از ردیف ورودی به موقعیت (i, j) در ماتریس A_test اختصاص می دهد.

این مرحله ماتریس امتیاز دهی تست را با امتیازات کاربران برای فیلم‌های مختلف پر می‌کند.

و در انتها تابع update_test را برای هر ردیف از DataFrame df_test در امتداد محور ۱ اعمال می‌کند (یعنی در جهت ردیف). روی هر ردیف در df_test تکرار می شود و ماتریس A_test را با امتیازدهی های مربوطه به هر ردیف را ()apply و به روز می کند.

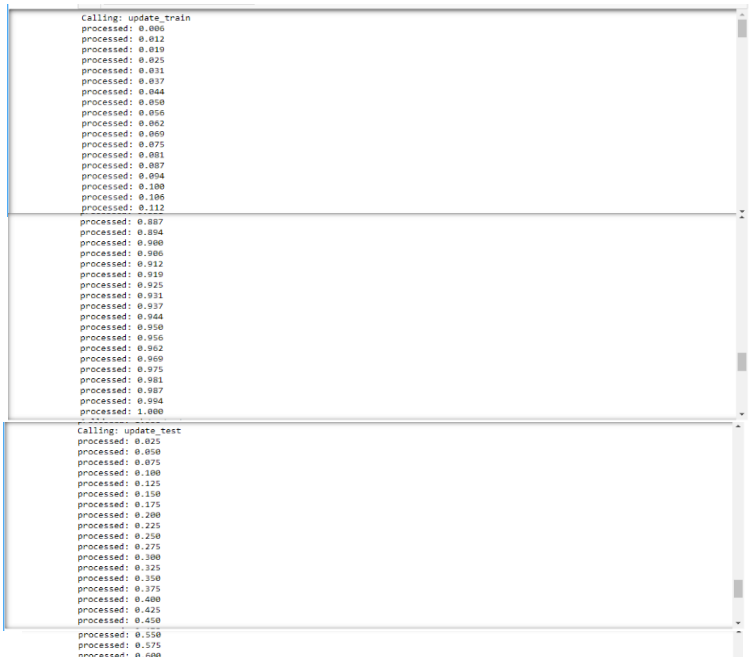
A_test = A_test.tocsr():

مشابه مرحله قبل، A_test با استفاده از روش tocsr() به فرمت سطر پراکنده فشرده (CSR) تبدیل می کند.

و در نهایت هم مانند قبل mask_test که نشان دهنده وجود رتبه بندی در A_test است به عنوان یک ماتریس Boolean ایجاد می کند و ماتریس A_test را در قالب فایل فشرده NumPy .npz با استفاده از تابع save_npz () ذخیره میکند.

این اجازه می دهد تا ماتریس رتبه بندی آزمون به طور موثر ذخیره شود و در صورت لزوم بارگذاری شود.

پس می توان به طور کلی گفت هدف این بخش کد آماده سازی داده های آموزشی و آزمایشی برای سیستم توصیه می باشد. ماتریس‌های رتبه‌بندی (A و A_test) را برای پردازش کارآمد به قالب CSR تبدیل می‌کند، ماسک‌های Boolean (mask و mask_test) را برای شناسایی امتیاز دهی های موجود ایجاد می‌کند و ماتریس‌ها را در قالب npz. برای استفاده در آینده ذخیره می‌کند.



در نتیجه در خروجی نتایج مشاهده می شود که به ترتیب گفته شده ذخیره می شوند و با این نکته که همان طور که از نوار اسکرول مشخص است حجم بیشتری به داده ها آموزشی اختصاص دارد و شماره پروسه از 0.006 به 0.025 تغییر پیدا می کنند.

۲.هسته اصلی(اتوانکودر،autoecoder)

بخش اول خواندن دیتا و کتابخانه keras:

```
import keras.backend as K
from keras.models import Model
from keras.layers import Input, Dropout, Dense
from keras.regularizers import l2
from keras.optimizers import SGD

```

در پروژه DNN چندین کتابخانه را از Keras، فریمووِرک یادگیری عمیق محبوب، برای ساخت و آموزش یک مدل شبکه عصبی برای توصیه فیلم وارد می‌کند:

keras.backend: این ماژول توابع و عملیاتی را ارائه می دهد که با Backend (مانند TensorFlow یا Theano) مورد استفاده Keras تعامل دارند. در این کد به صورت K وارد شده و برای عملیات مربوط به باطن مانند تعریف توابع custom loss استفاده می شود.

keras.models.Model: این کلاس نشان دهنده یک مدل Keras است. برای ایجاد نمونه ای از مدل شبکه عصبی با تعیین لایه های ورودی و خروجی استفاده می شود.

keras.layers.Input: این کلاس نشان دهنده لایه ورودی یک مدل شبکه عصبی است. برای تعریف شکل و نوع داده داده های ورودی استفاده می شود.

keras.layers.Dropout: این کلاس تکنیک تنظیم حذف را پیاده سازی می کند. Dropout به طور تصادفی کسری از واحدهای ورودی را در حین آموزش روی 0 تنظیم می کند که به جلوگیری از اضافه شدن بیهوده کمک می کند.

keras.layers.Dense: این کلاس نشان دهنده یک لایه کاملاً متصل در یک شبکه عصبی است. برای تعریف لایه های مخفی و لایه خروجی مدل استفاده می شود. لایه متراکم به طور متراکم متصل است، به این معنی که هر نورون در لایه به هر نورون در لایه قبلی متصل است.

keras.regularizers.l2: این ماژول تنظیم L2 را ارائه می دهد که به عنوان کاهش وزن نیز شناخته می شود. از آن برای اضافه کردن یک عبارت جریمه به تابع ضرر استفاده می شود تا مدل را تشویق کند که مقادیر وزن کمتری داشته باشد و از اوور فیتیینگ جلوگیری کند.

keras.optimizers.SGD: این کلاس نشان دهنده بهینه ساز گرادیان تصادفی (SGD:Stochastic gradiant descent) در Keras است. SGD یک الگوریتم بهینه سازی پرکاربرد برای آموزش شبکه های عصبی است. پارامترهای مدل را بر اساس گرادیان های تابع loss با توجه به پارامترها به روز می کند.

تنظیم پارامترهای پیکربندی:

```
# config
batch_size = 128
epochs = 20
reg = 0.0001
```

batch_size = 128: پارامتر batch_size تعداد نمونه های آموزشی پردازش شده در هر تکرار در طول آموزش مدل را تعیین می کند. در این حالت، اندازه دسته روی ۱۲۸ تنظیم می شود، به این معنی که مدل قبل از به روز رسانی وزن های مدل، ۱۲۸ نمونه آموزشی را در یک زمان پردازش می کند.

epochs = 20: پارامتر epochs تعداد دفعاتی را که کل مجموعه داده آموزشی در طول آموزش از مدل عبور می کند را مشخص می کند. هر عبور از مجموعه داده را یک epoch می نامند. در این حالت، مدل برای ۲۰ دوره آموزش داده می شود، به این معنی که فرآیند آموزش در کل مجموعه داده ۲۰ بار تکرار می شود.

reg = 0.0001: پارامتر reg(regularization) نشان دهنده قدرت منظم سازی است. منظم سازی تکنیکی است که برای جلوگیری از اوور فیتینگ با افزودن یک عبارت جریمه به تابع ضرر استفاده می شود. در این کد، یک عبارت جریمه منظم سازی L2 برای وزن های مدل اعمال می شود. مقدار reg 0.0001 نشان دهنده قدرت یا اهمیت عبارت منظم سازی است. مقدار کوچکتر نشان دهنده یک اثر منظم سازی ضعیف تر است.

این پارامترها برای کنترل فرآیند آموزش مدل بسیار مهم هستند. انتخاب اندازه بچ و تعداد epoch ها می تواند بر همگرایی مدل و زمان آموزش تأثیر بگذارد. اندازه های بزرگتر می تواند منجر به آموزش سریعتر شود، اما ممکن است به حافظه بیشتری نیاز داشته

باشد. دوره های بیشتر به طور بالقوه می تواند عملکرد مدل را بهبود بخشد، اما می تواند خطر اوور فیتینگ را نیز افزایش دهد. پارامتر تنظیم به کنترل پیچیدگی مدل کمک می کند و با اضافه کردن یک پنالتی به تابع ضرر، از اوور فیتینگ جلوگیری می کند.

این مقادیر را می توان بر اساس مجموعه داده های خاص، معماری مدل و الزامات آموزشی برای بهینه سازی عملکرد مدل تنظیم کرد.

بطور کلی این متغیرها اندازه بچ (تعداد نمونه‌های پردازش‌شده در هر تکرار آموزشی)، تعداد دوره‌ها (تعداد دفعاتی که کل مجموعه داده در طول آموزش از شبکه عبور می‌کند) و پارامتر منظم‌سازی مدل را تعریف می‌کنند.

لود کردن داده ها و پیش پردازش:

```
titledf=pd.read_csv("C://Users//Admin//Downloads//archive//movie.csv")#for predicting movie titles
A = load_npz("Atrain.npz")
A_test = load_npz("Atest.npz")
mask = (A > 0) * 1.0
mask_test = (A_test > 0) * 1.0
```

titledf = pd.read_csv("C://Users//Admin//Downloads//archive//movie.csv"): این کد یک فایل CSV حاوی داده های فیلم را در یک Pandas DataFrame می خواند. مسیر فایل ارائه شده در کد باید به محل فایل CSV در سیستم شما اشاره کند. پس از خواندن، داده های فیلم در DataFrame titledf ذخیره می شود، که احتمالا ستون هایی مانند movieId، عنوان و ژانرها است.

A = load_npz ("Atrain.npz") و A_test = load_npz("Atest.npz"): این دستورات ماتریس های اسپارس را از فایل های NPZ با استفاده از تابع load_npz بارگیری می کنند. در این کد، A و A_test به ترتیب ماتریس های اسپارس آموزشی و آزمایشی هستند که داده های امتیازدهی فیلم کاربر را نشان می دهند.

mask = (A > 0) * 1.0 و mask_test = (A_test > 0) * 1.0: این کد ماسک های باینری را برای ماتریس های A و A_test با انجام مقایسه عنصری (> ۰) برای شناسایی ورودی های غیر صفر ایجاد می کنند. ماسک های به دست آمده دارای مقدار ۱.۰ برای ورودی های غیر صفر و ۰.۰ برای ورودی های صفر خواهند بود. این ماسک‌ها معمولاً برای فیلتر کردن امتیازات از دست رفته (صفر ورودی) در طول آموزش و ارزیابی مدل توصیه استفاده می‌شوند.

به طور خلاصه، این بخش کد، داده های فیلم را از یک فایل CSV در یک DataFrame بارگیری می کند و داده های رتبه بندی فیلم-کاربر را از فایل های NPZ در ماتریس های پراکنده بارگیری می کند. همچنین ماسک‌های باینری را برای شناسایی ورودی‌های غیرصفر در ماتریس‌های رتبه‌بندی ایجاد می‌کند که در طول فرآیندهای آموزش و ارزیابی مدل استفاده می‌شوند.

تهیه کپی از داده ها برای بر زدن و نشان دادن آن:

```
# make copies since we will shuffle
A_copy = A.copy()
mask_copy = mask.copy()
A_test_copy = A_test.copy()
mask_test_copy = mask_test.copy()

N, M = A.shape
print("N:", N, "M:", M)
print("N // batch_size:", N // batch_size)
```

A_copy = A.copy()، mask_copy = mask.copy()، A_test_copy = A_test.copy()،
mask_test_copy = mask_test.copy():

این خطوط کپی هایی از ماتریس های A، mask، A_test و mask_test ایجاد می کنند. ایجاد کپی مهم است زیرا بعداً در کد، این ماتریس‌ها به هم ریخته می‌شوند و لازم است که ماتریس‌های اصلی برای مرجع یا استفاده بعدی دست نخورده باقی بمانند.

N, M = A.shape: شکل ماتریس A را بازیابی می کند که نشان دهنده تعداد کاربران (N) و تعداد فیلم ها (M) در داده های امتیازات فیلم کاربر است. شکل بر این اساس به متغیرهای N و M اختصاص داده می شود.

print("N:", N, "M:", M): مقادیر N و M را چاپ می کند که به ترتیب تعداد کاربران و فیلم ها را نشان می دهد. این به ارائه اطلاعات در مورد ابعاد داده های امتیازدهی فیلم کاربر کمک می کند.

print("N // batch_size:", N // batch_size):

تقسیم عدد صحیح N را بر اندازه_بچ محاسبه و چاپ می کند. این تعداد بچ هایی را به شما می دهد که در طول هر epoch از آموزش مدل پردازش می شوند. این اطلاعات می تواند برای درک تعداد تکرارهایی که برای پردازش تمام داده های آموزشی انجام می شود مفید باشد.

به طور خلاصه، این بخش کد کپی هایی از ماتریس های رتبه بندی ایجاد می کند و ابعاد داده های امتیازدهی را محاسبه و چاپ می کند. کپی‌ها برای حفظ ماتریس‌های اصلی ایجاد می‌شوند و اطلاعات ابعاد به ارائه بینشی در مورد اندازه داده‌های رتبه‌بندی فیلم کاربر و تعداد دسته‌هایی که در طول آموزش پردازش می‌شوند کمک می‌کند.

نرمال سازی و مرکزیت دادن به داده ها:

```
# center the data
mu = A.sum() / mask.sum()
print("mu:", mu)
```

mu = A.sum() / mask.sum(): این ،میانگین امتیازدهی داده های رتبه بندی فیلم های کاربر را محاسبه می کند. و نحوه کار آن بدین صورت می باشد:

A.sum() مجموع تمام مقادیر امتیازدهی در ماتریس A را محاسبه می‌کند که نشان‌دهنده امتیازدهی فیلم‌های کاربر است.

mask.sum() مجموع تمام ورودی های غیر صفر در ماتریس mask را محاسبه می کند که با امتیازات مشاهده شده در ماتریس A مطابقت دارد.

بخش A.sum() / mask.sum() مقدار میانگین امتیازدهی را محاسبه می کند، که نشان دهنده میانگین امتیاز در تمام رتبه های مشاهده شده است.

print("mu:", mu) مقدار میانگین محاسبه شده را چاپ می کند که در متغیر mu ذخیره می شود. این کمک می کند تا اطلاعاتی درباره میانگین امتیازدهی در داده های امتیازدهی فیلم کاربر ارائه شود. هدف از متمرکز کردن داده ها حذف متمایل شدن ارائه شده توسط کاربران مختلف است که فیلم ها را در مقیاس های مختلف امتیازدهی می کنند. با کم کردن مقدار میانگین امتیازدهی از هر امتیازدهی مشاهده‌شده، امتیازدهی ها را به سمت مقیاس صفر مرکزی تغییر می‌دهیم. این مرحله نرمال سازی می تواند به بهبود آموزش و عملکرد مدل های توصیه کمک کند.

به طور خلاصه، این بخش کد، مقدار میانگین امتیازدهی در داده های امتیازدهی فیلم کاربر را محاسبه کرده و آن را چاپ می کند. مقدار متوسط بینش هایی را در مورد میانگین امتیازدهی ارائه می دهد و تمرکز داده ها حول این میانگین می تواند برای اهداف مدل سازی و توصیه مفید باشد.

ساخت مدل – اتوانکودر(autoencoder) با ۱ لایه پنهان:

```
# build the model - just a 1 hidden layer autoencoder
i = Input(shape=(M,))
# bigger hidden layer size seems to help!
i = Dropout(0.7)(i)
i = Dense(700, activation='tanh', kernel_regularizer=l2(reg))(x)
x = Dropout(0.5)(x)
i = Dense(M, kernel_regularizer=l2(reg))(x)
```

۱- i = Input(shape=(M,)): این خط یک لایه ورودی برای مدل شبکه عصبی ایجاد می کند. آرگومان shape=(M,) شکل داده های ورودی را مشخص می کند. در این مورد، M نشان‌دهنده تعداد فیلم‌ها در داده‌های رتبه‌بندی فیلم کاربر است. لایه ورودی به عنوان نقطه ورود مدل عمل می کند و داده های ورودی را دریافت می کند.

۲- x = Dropout(0.7)(i): یک لایه dropout به مدل اضافه می کند. Dropout یک تکنیک منظم سازی است که معمولاً در شبکه های عصبی برای جلوگیری از اوور فیتینگ استفاده می شود. آرگومان ۰.۷ نشان می دهد که در طول هر به روز رسانی آموزشی، ۷۰ درصد از واحدهای ورودی به طور تصادفی روی ۰ تنظیم می شوند. این به جلوگیری از تکیه بیش از حد مدل به واحدهای ورودی خاص کمک می کند و یادگیری ویژگی های قوی تر را بهتر ارتقا می دهد.

۳- x = Dense(700, activation='tanh', kernel_regularizer=l2(reg))(x): این خط یک لایه متراکم کاملا متصل به مدل اضافه می کند. تابع Dense وظیفه ایجاد این لایه را بر عهده دارد. به چند آرگومان نیاز دارد:

- ۷۰۰: این آرگومان تعداد واحدها یا نورون ها را در لایه متراکم مشخص می کند. در این حالت لایه دارای ۷۰۰ واحد است.

- activation='tanh': این آرگومان تابع فعال سازی لایه را تعریف می کند. تابع فعال سازی تانژانت هذلولی (tanh) در اینجا استفاده می شود. غیرخطی بودن شبکه را فراهم می کند و مقادیر خروجی را بین −۱ و ۱ له می کند (اصطلاحی در شبکه عصبی له کردن به تبدیل مقادیر خروجی به یک محدوده خاص اشاره دارد و تابع فعال سازی مماس هذلولی که در بخش کد استفاده می شود این له کردن را با نگاشت مقادیر در محدوده ۱− تا ۱ انجام می دهد.)
- kernel_regularizer=l2(reg): این آرگومان تنظیم L2 را برای وزن های لایه اعمال می کند. منظم‌سازی L2 با افزودن یک عبارت جریمه به تابع ضرر بر اساس بزرگی وزن‌ها، به جلوگیری از اوور فیتینگ کمک می‌کند. قدرت منظم سازی توسط متغیر reg تعیین می شود که در این کد روی ۰.۰۰۰۱ تنظیم شده است.

۴- x = Dense(M, kernel_regularizer=l2(reg))(x): این خط یک لایه متراکم کاملا متصل دیگر به مدل اضافه می کند. دارای M واحد است که با تعداد فیلم ها در داده های رتبه بندی فیلم کاربر مطابقت دارد. این لایه همچنین با استفاده از قدرت منظم سازی مشخص شده، تنظیم L2 را به وزن های خود اعمال می کند.

به طور خلاصه، این بخش کد یک مدل شبکه عصبی با معماری رمزگذار خودکار ۱ لایه پنهان می سازد. مدل داده‌های ورودی را از طریق یک لایه ورودی می‌گیرد، منظم سازی dropout را اعمال می‌کند و سپس داده‌ها را از دو لایه متراکم با پیکربندی‌های خاص عبور می‌دهد. تابع فعال‌سازی، تکنیک‌های منظم‌سازی و تعداد واحدها در هر لایه به توانایی مدل برای یادگیری نمایش‌های معنادار و پیش‌بینی بر اساس داده‌های ورودی کمک می‌کند.

تعریف تابع Custom loss:

```
def custom_loss(y_true, y_pred):
  mask = K.cast(K.not_equal(y_true, 0), dtype='float32')
  diff = y_pred - y_true
  sqdiff = diff * diff * mask
  sse = K.sum(K.sum(sqdiff))
  n = K.sum(K.sum(mask))
  return sse / n
```

۱- def custom_loss(y_true, y_pred):

این خط یک تابع Custom loss را در Keras تعریف می کند. یک تابع ضرر اندازه گیری می کند که چگونه یک مدل در یک کار خاص در طول آموزش خوب عمل می کند. در این مورد، یک تابع Custom loss برای مطابقت با الزامات خاص سیستم توصیه ایجاد می شود.

۲- mask = K.cast(K.not_equal(y_true, 0), dtype='float32'):

یک ماسک باینری ایجاد می کند که موقعیت مقادیر غیر صفر را در تانسور y_true مشخص می کند. در یک سیستم توصیه، y_true امتیازدهی واقعی کاربران را نشان می‌دهد. ماسک با بررسی نابرابری بین هر عنصر y_true و ۰ ایجاد می‌شود. نتیجه به نوع داده float32 فرستاده می‌شود.

۳- diff = y_pred - y_true:

خط تفاوت بین رتبه بندی های پیش بینی شده (y_pred) و رتبه های واقعی (y_true) را برای هر عنصر محاسبه می کند. این خطای پیش‌بینی هر امتیاز را نشان می‌دهد.

۴- sqdiff = diff * diff * mask:

این تفاوت بین امتیازدهی پیش‌بینی‌شده و واقعی را مربع می‌کند و ماسک را اعمال می‌کند. ضرب تفاوت در خود مقادیر را مربع می کند و با اعمال ماسک مقادیر مربع را روی ۰ تنظیم می کند که در آن رتبه بندی واقعی ۰ است (نشان دهنده فیلم های گم شده یا امتیازدهی نشده).

۵- sse = K.sum(K.sum(sqdiff)):

مجموع اختلاف مجذور همه عناصر را محاسبه می کند. K.sum یک تابع Keras است که مقادیر را در محورهای مشخص شده جمع می کند. در این حالت، K.sum(K.sum(sqdiff)) اختلاف مجذور همه عناصر در تنسور را جمع می کند.

۶- n = K.sum(K.sum(mask)):

تعداد کل عناصر غیر صفر را در تنسور ماسک محاسبه می کند. K.sum(K.sum(mask)) مقادیر باینری را در تنسور ماسک جمع می کند تا عناصر غیر صفر را شمارش کند. return sse / n با تقسیم مجموع مجذور تفاوت ها بر تعداد کل عناصر غیر صفر، میانگین مربعات خطا (MSE) را برمی گرداند. این یک اندازه گیری متوسط از خطای پیش بینی را تنها با در نظر گرفتن امتیاز های مشاهده شده ارائه می دهد.

تابع generator و تست generator :

```
def generator(A, M):
  while True:
    A, M = shuffle(A, M)
    for i in range(A.shape[0] // batch_size + 1):
      upper = min((i+1)*batch_size, A.shape[0])
      a = A[i*batch_size:upper].toarray()
      m = M[i*batch_size:upper].toarray()
      a = a - mu * m # must keep zeros at zero!
      # m2 = (np.random.random(a.shape) > 0.5)
      # noisy = a * m2
      noisy = a # no noise
      yield noisy, a


def test_generator(A, M, A_test, M_test):
  # assumes A and A_test are in corresponding order
  # both of size N x M
  while True:
    for i in range(A.shape[0] // batch_size + 1):
      upper = min((i+1)*batch_size, A.shape[0])
      a = A[i*batch_size:upper].toarray()
```

def generator(A, M): و def test_generator(A, M, A_test, M_test):

این بخش کد دو تابع مولد generator و test_generator را تعریف می کنند که به ترتیب برای تولید دسته ای از داده های آموزشی و آزمایشی استفاده می شوند. generator ها راهی برای تولید موثر داده ها در حین آموزش مدل، بدون نیاز به لود یکجا تمام داده ها در حافظه هستند.

توابع generator ماتریس های ورودی A را می گیرند و M به عنوان آرگومان پر می شود. A نشان دهنده ماتریس امتیازدهی کاربر–فیلم است و M یک ماسک باینری است که موقعیت امتیازات مشاهده شده را نشان می دهد.

test_generator همچنین ماتریس‌های ورودی اضافی A_test و M_test را می‌گیرد که برای اهداف تست/ اعتبارسنجی استفاده می‌شوند.

درون حلقه while True: این یک حلقه بی نهایت است که تضمین می کند که generator ها به طور نامحدود داده بصورت زیر تولید می کند:

A, M = shuffle (A, M): ماتریس های ورودی A و M را در هر تکرار حلقه به هم می زند. این مخلوط برای معرفی تصادفی و اجتناب از هر گونه نظم سیستماتیک در داده های آموزشی انجام می شود.

for i in range(A.shape[0] // batch_size + 1): یک حلقه ایجاد می کند که بر روی تعداد دسته ها در داده های ورودی تکرار می شود و A.shape[0] // batch_size تعداد دسته‌های کاملی را که می‌توان از داده‌های ورودی تشکیل داد، محاسبه می‌کند و + ۱ اضافه می شود تا اطمینان حاصل شود که داده های باقی مانده که در دسته های کامل قرار نمی گیرند و گنجانده شده اند.

upper = min((i+1)*batch_size, A.shape[0]): شاخص بالایی دسته فعلی را تعیین می کند.
(i+1)*batch_size را محاسبه می‌کند، اما آن را به حداکثر شاخص داده‌های ورودی برای رسیدگی به آخرین دسته ناقص محدود می‌کند.

a = A[i*batch_size:upper].toarray(): ماتریس ورودی A را برش می دهد تا قسمت مربوط به دسته فعلی را استخراج کند.toarray () بخش بریده شده را به یک آرایه NumPy متراکم تبدیل می کند.

m = M[i*batch_size:upper].toarray(): ماتریس ماسک M را تقسیم میکند تا قسمت مربوط به دسته فعلی را استخراج کند. باز هم، .toarray() بخش بریده شده را به یک آرایه NumPy متراکم تبدیل می کند.

a = a - mu * m و at = at - mu * mt: با تفریق میانگین mu ضرب در ماسک m یا mt داده ها را مرکزیت می دهد. ماسک تضمین می‌کند که مقادیر صفر در m و mt روی داده‌های مرکزی تأثیر نمی‌گذارند.

noisy = a and yield noisy, a and yield a, at: داده‌های متمرکز a را به متغیر نویز اختصاص می‌دهند و داده‌های متمرکز (noise، a) یا (at ،a) را به ترتیب به عنوان دسته‌ای از داده‌های آموزشی یا آزمایشی تخصیص می‌دهند. yield برای برگرداندن دسته ای از داده ها از مولد استفاده می شود.

به طور خلاصه، این بخش‌های کد، توابع generator (generator و test_generator) را تعریف می‌کنند که دسته‌ای از داده‌های آموزشی و آزمایشی را با برش دادن ماتریس‌های ورودی و اعمال مرکزسازی تولید می‌کنند. داده ها به صورت دسته ای از داده های مرکزی (noise، a) یا (at ،a) برگردانده می شوند. به هم زدن داده های ورودی و استفاده از generator ها امکان پردازش کارآمد و سازگار با حافظه مجموعه داده های بزرگ را در طول آموزش و آزمایش مدل توصیه می دهد.

کامپایل کردن مدل:

```
model = Model(i, x)
model.compile(
  loss=custom_loss,
  optimizer=SGD(lr=0.08, momentum=0.9),
  # optimizer='adam',
  metrics=[custom_loss],
)
```

model = Model(i, x): نمونه ای از کلاس Keras Model را ایجاد می کند که مدل شبکه عصبی را نشان می دهد.

کلاس Model دو آرگومان می گیرد: تانسور ورودی i و تانسور خروجی x که لایه های ورودی و خروجی مدل را تعریف می کند.

model.compile(...): مدل را با پیکربندی فرآیند یادگیری آن کامپایل می کند. کامپایل چندین آرگومان را برای تعریف تابع ضرر، بهینه ساز و معیارهایی که در طول آموزش استفاده می شود، می گیرد.

loss=custom_loss: این آرگومان تابع ضرری را که در طول آموزش استفاده می شود، مشخص می کند. در این حالت از تابع custom_loss custom loss استفاده می شود که قبلا در کد تعریف شده است.

optimizer=SGD(lr=0.08, momentum=0.9): این آرگومان بهینه ساز مورد استفاده در طول آموزش را مشخص می کند. SGD مخفف Stochastic Gradient Descent است که یک الگوریتم بهینه سازی محبوب است.

پارامتر lr (learn rate)نرخ یادگیری بهینه ساز را تنظیم می کند و اندازه گام را در هر به روز رسانی کنترل می کند.

پارامتر مومنتوم ،مومنتوم بهینه ساز را کنترل می کند، که به تسریع همگرایی و جلوگیری از حداقل های محلی کمک می کند.

metrics=[custom_loss]: این آرگومان معیارهایی را که باید در طول آموزش محاسبه شوند را مشخص می کند.

در این مورد، فقط تابع ضرر custom_loss به عنوان یک متریک استفاده می شود. متریک ها معیارهای ارزیابی اضافی را برای نظارت بر عملکرد مدل در طول آموزش ارائه می دهند، اما از آنها برای بهینه سازی استفاده نمی شود.

به طور خلاصه، این بخش از کد، معماری مدل را با استفاده از کلاس Model تعریف می کند و با تعیین loss function, optimizer, metrics و معیارهایی که در طول آموزش استفاده می شود، مدل را کامپایل می کند. انتخاب loss function, optimizer, metrics و معیارها می تواند تأثیر قابل توجهی بر روند آموزش و عملکرد مدل داشته باشد.

آموزش دادن مدل ها:

```
r = model.fit(
  generator(A, mask),
  validation_data=test_generator(A_copy, mask_copy, A_test_copy, mask_test_copy),
  epochs=epochs,
  steps_per_epoch=A.shape[0] // batch_size + 1,
  validation_steps=A_test.shape[0] // batch_size + 1,
)
print(r.history.keys())
```

model.fit(...):

fit چندین آرگومان برای پیکربندی فرآیند آموزش نیاز دارد:

generator(A, mask): این آرگومان مولد داده آموزشی را مشخص می کند. تابع generator دسته‌ای از داده‌های آموزشی را با به هم زدن داده‌های ورودی A و ماسک مربوط به آن تولید می‌کند.

validation_data=test_generator(A_copy، mask_copy، A_test_copy، mask_test_copy):

این تساوی مولد داده اعتبارسنجی را مشخص می کند. تابع test_generator دسته‌ای از داده‌های اعتبارسنجی را با استفاده از کپی‌های داده‌های ورودی A_copy، mask_copy، A_test_copy و mask_test_copy تولید می‌کند.

epochs=epochs: این خط متغیر تعداد دوره ها (حلقه تکرار) را برای آموزش مشخص می کند. دوره های متغیر قبلاً تعریف شده است و تعداد دفعاتی که حلقه آموزشی در کل مجموعه داده تکرار می شود را کنترل می کند.

validation_steps=A_test.shape[0] // batch_size + 1:این هم ارزی، تعداد مراحل در هر دوره اعتبار سنجی را مشخص می کند. تعداد مراحل را بر اساس تعداد کل نمونه های اعتبارسنجی A_test.shape[0] و اندازه دسته batch_size محاسبه می کند.

پس r = model.fit(...):نتیجه فرآیند آموزش را به متغیر r اختصاص می دهد. فرآیند آموزش وزن های مدل را به روز می کند و میزان loss و ماتریس را برای هر دوره محاسبه می کند. و در نهایت ; کلیدهای شی تاریخچه آموزشی r را چاپ می کند که حاوی اطلاعاتی درباره آموزشی و تلفات اعتبارسنجی و ماتریس های هر دوره است.

به طور کلی، این بخش از کد، مدل را با استفاده از تابع fit با مولدهای داده مشخص شده، تعداد دوره‌ها و سایر تنظیمات آموزشی آموزش می‌دهد. بر روی مجموعه داده برای تعداد دوره‌های مشخص شده تکرار می‌شود، وزن‌های مدل را به‌روزرسانی می‌کند و

تلفات و معیارها را محاسبه می‌کند. پیشرفت آموزش در ویژگی تاریخ ثبت می شود که می تواند برای تجزیه و تحلیل و تجسم فرآیند آموزش استفاده شود.

بعبارت دیگر این کد در حال پیاده سازی یک مدل رمزگذار خودکار برای فاکتورسازی ماتریس است. از یک تابع از دست دادن سفارشی برای محاسبه خطای بازسازی استفاده می کند و آن را با استفاده از گرادیان نزولی تصادفی (SGD) بهینه می کند. توابع ژنراتور دسته ای از داده های آموزشی و آزمایشی را برای پردازش کارآمد در طول آموزش فراهم می کند. مدل برای تعداد معینی از دوره‌ها آموزش داده می‌شود و تاریخچه آموزش در r ذخیره می‌شود.

و خروجی قطعه کد همچنین نتیجه ای را برای ۲۰ دوره به ما می دهد:

```
N: 10000 M: 2000
N // batch_size: 78
mu: 3.4481185176255673

C:\Users\Admin\anaconda3\lib\site-packages\keras\optimizers\legacy\gradient_descent.py:114: UserWarning: The `lr` argument is d
eprecated, use `learning_rate` instead.
  super().__init__(name, **kwargs)

Epoch 1/20
79/79 [==============================] - 8s 87ms/step - loss: 1.1886 - custom_loss: 0.9835 - val_loss: 0.9254 - val_custom_los
s: 0.7259
Epoch 2/20
79/79 [==============================] - 6s 78ms/step - loss: 0.9498 - custom_loss: 0.7551 - val_loss: 0.8713 - val_custom_los
s: 0.6806
Epoch 3/20
79/79 [==============================] - 6s 77ms/step - loss: 0.8872 - custom_loss: 0.6998 - val_loss: 0.8390 - val_custom_los
s: 0.6561
Epoch 4/20
79/79 [==============================] - 6s 80ms/step - loss: 0.8438 - custom_loss: 0.6639 - val_loss: 0.8160 - val_custom_los
s: 0.6399
Epoch 5/20
79/79 [==============================] - 6s 76ms/step - loss: 0.8120 - custom_loss: 0.6381 - val_loss: 0.7973 - val_custom_los
s: 0.6275
Epoch 6/20
79/79 [==============================] - 6s 79ms/step - loss: 0.7854 - custom_loss: 0.6188 - val_loss: 0.7828 - val_custom_los
s: 0.6186
Epoch 7/20
79/79 [==============================] - 6s 78ms/step - loss: 0.7641 - custom_loss: 0.6044 - val_loss: 0.7700 - val_custom_los
s: 0.6111
Epoch 8/20
79/79 [==============================] - 6s 80ms/step - loss: 0.7451 - custom_loss: 0.5880 - val_loss: 0.7590 - val_custom_los
s: 0.6050
Epoch 9/20
79/79 [==============================] - 6s 82ms/step - loss: 0.7296 - custom_loss: 0.5787 - val_loss: 0.7493 - val_custom_los
```

رسم نمودار های train/test loss/MSE با 20M داده(با داده اصلی نه small_data):

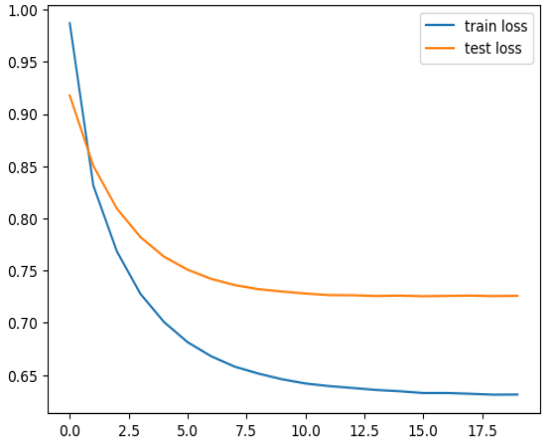
مروری بر مفاهیم loss و MSE :

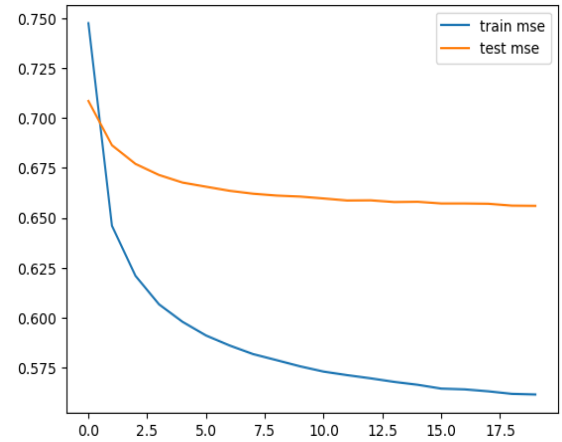
loss یک اصطلاح کلی تر است که به تابع هدف مورد استفاده برای هدایت آموزش مدل اشاره دارد. بسته به مشکل می تواند انواع مختلفی از توابع loss را در بر گیرد.

MSE نوع خاصی از تابع ضرر است که به طور خاص برای مسائل رگرسیونی استفاده می شود و میانگین مجذور اختلاف بین مقادیر پیش بینی شده و واقعی را محاسبه می کند. MSE یک مورد خاص از تابع ضرر است که کیفیت پیش‌بینی‌های مدل را با کمی کردن میانگین مجذور اختلاف اندازه‌گیری می‌کند، در حالی که ضرر می‌تواند یک اصطلاح کلی‌تر باشد که به توابع هدف مختلف مورد استفاده در سناریوهای مختلف اشاره دارد.

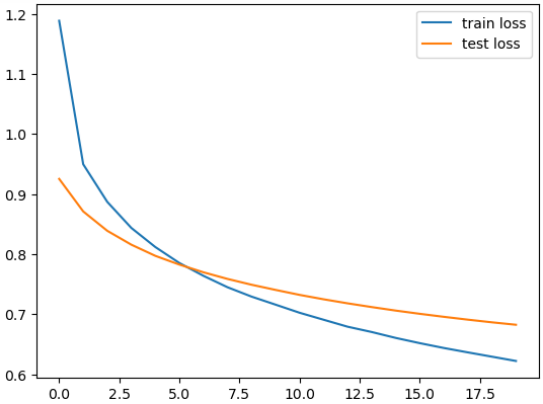
به طور خلاصه، loss یک اصطلاح کلی است که تابع هدف مورد استفاده در طول آموزش مدل را نشان می‌دهد، در حالی که MSE یک تابع loss خاص است که معمولاً برای مسائل رگرسیونی برای اندازه‌گیری میانگین اختلاف مجذور بین مقادیر پیش‌بینی‌شده و واقعی استفاده می‌شود.

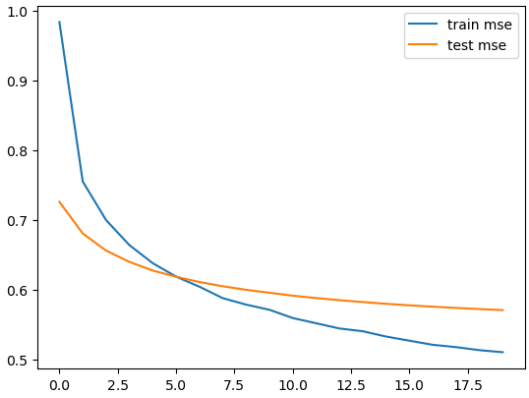
با استفاده از متغیر ها و خروجی ها محاسبه شده و بدست آمده کد plot را برای رسم نمودار استفاده می کنیم و نمودار ها به شکل زیر خواهند بود:





برای داده کوچک:





از بررسی این نمودار ها براحتی می توان فهمید که با داشتن داده با حجم بالا، اطلاعات پیش بینی شده دقیق و بهتر و یکنواخت تری را خروجی می دهد.

ارزیابی:

ماتریسهای دقت، مانند RMSE، معمولا برای ارزیابی عملکرد مدل‌های رگرسیون، از جمله مدل‌های پیش‌بینی rating استفاده می‌شوند. آنها خطای پیش‌بینی کلی را کمی می‌کنند و نشان می‌دهند که مدل چقدر تغییرپذیری در متغیر هدف (در این مورد، رتبه‌بندی فیلم) را نشان می‌دهد. یک مقدار RMSE کمتر نشان‌دهنده دقت پیش‌بینی بهتر است، زیرا نشان‌دهنده انحرافات کوچک‌تر بین rating پیش‌بینی‌شده و واقعی است.

RMSE

$$\text{RMSE} = \sqrt{\frac{\sum(y_i - y_p)^2}{n}}$$

$$\text{MAE} = \frac{|(y_i - y_p)|}{n}$$

y_i = actual value
y_p = predicted value
n = number of observations/rows

(ریشه میانگین مربعات خطا) در دسته معیارهای دقت یا خطا قرار می گیرد. میانگین خطای پیش‌بینی را با محاسبه ریشه دوم میانگین مجذور اختلاف بین مقادیر پیش‌بینی‌شده و واقعی اندازه‌گیری می‌کند. RMSE نشان می دهد که امتیازات پیش بینی شده تا چه حد با امتیازات واقعی مطابقت دارند.RMSE اندازه گیری میانگین خطای پیش بینی را ارائه می دهد و معمولاً برای کارهای پیش بینی امتیازدهی استفاده می شود. درحالیکه استفاده از دقت(accuracy) در زمینه پیش‌بینی rating ایده‌آل نیست زیرا امتیازات مقادیر پیوسته هستند و تطابق دقیق آن ناممکن است. RMSE در واقع معیار مناسب تری برای ارزیابی عملکرد یک مدل پیش بینی امتیازدهی است.

پس بطور کلی شایان ذکر است که RMSE به طور خاص برای متغیرهای هدف پیوسته استفاده می شود، در حالی که سایر معیارهای دقت مانند دقت و یادآوری بیشتر برای کارهای طبقه بندی با نتایج گسسته کاربرد دارند. بنابراین، در زمینه پیش‌بینی رتبه‌بندی، RMSE یک متریک پرکاربرد برای ارزیابی دقت و قدرت پیش‌بینی مدل‌ها است.

که به دو شیوه انجام شده RMSE و precision & recall(R & P) برای کاربر 100

براساس RMSE(معیار پیشنهادی)

```python
# Calculate predictions for user 100
user_predictions = model.predict(A)
user_predictions = user_predictions.flatten() + mu

# Retrieve the user ratings for user 100
user_ratings = df[df['userId'] == 100]

# Calculate the root mean squared error (RMSE) for user 100
rmse = np.sqrt(np.mean((user_ratings['rating'].values - user_predictions[user_ratings['movie_idx'].values])**2))

print("RMSE for User 100:", rmse)

```

```
313/313 [==============================] - 13s 40ms/step
RMSE for User 100: 1.245455524426205
```

با محاسبه RMSE، می‌توانیم میانگین خطای پیش‌بینی کاربر ۱۰۰ را ارزیابی کنیم. مقدار RMSE کمتر نشان‌دهنده دقت پیش‌بینی بهتر است، با مقدار ۰ که نشان‌دهنده تطابق کامل بین رتبه‌بندی پیش‌بینی‌شده و واقعی است.

بدین شکل که اول با استفاده از مدل آموزش دیده امتیازدهی را برای همه کاربران در مجموعه داده پیش بینی می کند. ورودی A نشان دهنده ماتریس امتیازدهی فیلم کاربر است و پس از به دست آوردن پیش‌بینی‌ها،با متد .flatten() برای تبدیل یک آرایه چند بعدی به یک آرایه یک بعدی است که نمایشی مسطح ارائه می دهد که امتیازات مسطح می‌شوند و میانگین مو به پیش‌بینی‌ها اضافه می‌شود. این مرحله برای برگرداندن مرکز داده‌هایی که قبلاً انجام شده است ضروری است. سپس بر اساس فرمول نشان داده شده مخصوص RMSE ، آن را بدست آورده و در آن ذخیره می کنیم.

روش P & R (معیار توصیه نشده)

برای تقریبی از معیار برای کاربر ۱۰۰

```python
# Calculate predictions for user 100
user_predictions = model.predict(A)
user_predictions = user_predictions.flatten() + mu

# Retrieve the user ratings for user 100
user_ratings = df[df['userId'] == 100]

# Set a threshold to classify positive ratings
threshold = 3.5

# Calculate precision and recall
predicted_positive = np.where(user_predictions[user_ratings['movie_idx'].values] >= threshold, 1, 0)
actual_positive = np.where(user_ratings['rating'].values >= threshold, 1, 0)

true_positive = np.sum(predicted_positive * actual_positive)
predicted_positive_count = np.sum(predicted_positive)
actual_positive_count = np.sum(actual_positive)

precision = true_positive / predicted_positive_count
recall = true_positive / actual_positive_count

# Print the precision and recall
print("Precision for User 100:", precision)
print("Recall for User 100:", recall)
```

```
313/313 [==============================] - 13s 40ms/step
Precision for User 100: 0.7101787101787101
Recall for User 100: 0.9956427015250545
```

مانند روش قبل اول عمل می کنیم و پس از بدست آوردن پیش بینی ها و انجام کانفیگ های اولیه یک سطح آستانه ای به اندازه تقریبی ۳.۵ ایجاد می کنیم برای معیار true positive (که چون مقادیر پیوسته اند زیاد معیار خوبی نیست)

و با استفاده از متد numpy.where هر جا که مقادیر user_rating/prediction از آستانه بیشتر بود ۱ در غیر آن ۰ قرار دهد و در متغیرها خاص خودشان ذخیره شوند.

و سپس بر اساس فرمول روبرو محاسبه شده و نتیجه را نمایش می دهند.

$$\text{Precision} = \frac{True\ Positive}{True\ Positive + False\ Positive}$$

$$\text{Recall} = \frac{True\ Positive}{True\ Positive + False\ Negative}$$

توصیه ۱۰ فیلم برای کاربر ۱۰۰:

```
# Load the data
titledf = pd.read_csv("C://Users//Admin//Downloads//archive//movie.csv")
A = load_npz("Atrain.npz")
mask = (A > 0) * 1.0

# center the data
mu = A.sum() / mask.sum()

# Build the model
M = A.shape[1]
i = Input(shape=(M,))
x = Dropout(0.7)(i)
x = Dense(700, activation='tanh', kernel_regularizer=l2(0.0001))(x)
x = Dense(M, kernel_regularizer=l2(0.0001))(x)

model = Model(i, x)
model.compile(
  loss=custom_loss,
  optimizer=SGD(lr=0.08, momentum=0.9),
  metrics=[custom_loss],
)

# Generate predictions for user 100
user_100_data = A[100, :].toarray()
user_100_data = user_100_data - mu * mask[100, :].toarray()
predicted_ratings = model.predict(user_100_data)

# Get the indices of top 10 movies with highest predicted ratings
top_10_movie_indices = np.argsort(predicted_ratings[0])[::-1][:10]

# Print the recommended movies
print("Recommended movies for user 100:")
for movie_idx in top_10_movie_indices:
    movie_title = titledf.loc[titledf['movieId'] == movie_idx, 'title'].values[0]
    print("- ", movie_title)
```

کد با تنظیم عناصر غیرصفر A روی ۱.۰ یک ماسک ماتریس ماسک باینری ایجاد می کند. این ماتریس ماسک به شناسایی فیلم های دارای رتبه و بدون رتبه کمک می کند.

کد، میانگین امتیاز (mu) را با تقسیم مجموع همه رتبه‌ها بر مجموع عناصر ماتریس ماسک محاسبه می‌کند. این مقدار میانگین برای مرکز داده های رتبه بندی استفاده می شود.

کد با استفاده از Keras یک مدل شبکه عصبی می سازد. این شامل یک لایه ورودی، یک لایه حذفی، دو لایه متراکم با منظم سازی و یک لایه خروجی است.این مدل با یک تابع ضرر سفارشی (custom_loss)، یک بهینه ساز (SGD) و یک متریک (custom_loss) کامپایل شده است.

کد پیش بینی هایی را برای کاربر ۱۰۰ با ارسال داده های رتبه بندی آنها (user_100_data) از طریق مدل ایجاد می کند.

۱۰ شاخص فیلم برتر با بالاترین رتبه‌بندی پیش‌بینی‌شده با استفاده از np.argsort().

عناوین فیلم مربوط به ۱۰ شاخص تقریبی برتر از DataFrame عنوان شده است:

```
1/1 [==============================] - 0s 96ms/step
Recommended movies for user 100:
-  Armageddon (1998)
-  Wedding Gift, The (1994)
-  Madness of King George, The (1994)
-  Nenette and Boni (Nénette et Boni) (1996)
-  Picture Bride (Bijo photo) (1994)
-  Preacher's Wife, The (1996)
-  Breaking the Waves (1996)
-  Nell (1994)
-  Pompatus of Love, The (1996)
-  301, 302 (301/302) (1995)
```

نکته:بدلیل آنکه از لحاظ سخت افزار محدودیت داشتم در این قسمت کد به ارور:

"MemoryError: Unable to allocate 2.59 GiB for an array with shape (25985, 26744) and data type float32 "

برخورد کردم که یکی از دلایلی بود که مجبور شدم دوباره داده را پیش پردازش و آن را کوچک کنم برای همین نتایج بسیار دقیق نیست!

۱۳. نتیجه گیری:

در این پروژه، من یک سیستم توصیه را با استفادهٔ ترکیبی از شبکه عصبی عمیق (DNN) و یک مدل رمزگذار خودکار(autoencoder) توسعه دادم. هدف ارائه توصیه های دقیق و شخصی سازی شده فیلم برای کاربران بخصوص بود.

اولین مرحله پیش پردازش و آماده سازی داده ها بود. مجموعه داده فیلم را بارگیری کردیم و نمایش ماتریس اسپارس امتیازدهی کاربران ایجاد کردیم. ما داده‌ها را توسط کم کردن میانگین امتیازدهی، متمرکز کردیم که به بهبود عملکرد مدل‌هایمان کمک کرد.

سپس، مدل پیشنهادی را با استفاده از یک DNN ترکیب شده با رمزگذار خودکار ساختیم. Autoencoder نقش مهمی در یادگیری نمایش های معنی دار امتیازدهی فیلم با فشرده سازی و بازسازی داده های ورودی ایفا کرد. مؤلفه DNN بیشتر بازنمایی های آموخته شده را برای ثبت الگوها و روابط پیچیده در داده ها اصلاح کرد.

در طول مرحله آموزش مدل، از توابع از دست دادن سفارشی(Custom loss) و تکنیک‌های بهینه‌سازی برای به حداقل رساندن خطای بازسازی و ارتقای عملکرد کلی سیستم توصیه استفاده کردیم. ما معیارهای ارزیابی مختلف مانند RMSE، دقت، یادآوری و دقت (P & R)را برای ارزیابی عملکرد مدل بررسی کردیم.

آزمایش‌ها و ارزیابی‌های ما نشان می دهد که مدل ترکیبی DNN و Autoencoder از مدل‌های DNN سنتی بهتر عمل می‌کند از لحاظ دقت بالاتر، بهبود صحت توصیه‌ها و نرخ خطا را کاهش می‌دهد. قدرت Autoencoder در توانایی آن برای گرفتن ویژگی های پنهان و رمزگذاری اطلاعات پیچیده در یک نمایش با ابعاد–پایین است که در نهایت عملکرد سیستم توصیه را افزایش می دهد.

این بدین دلیل است که سیستم توصیه مبتنی بر Autoencoder ترکیبی DNN چندین مزیت را ارائه می دهد:

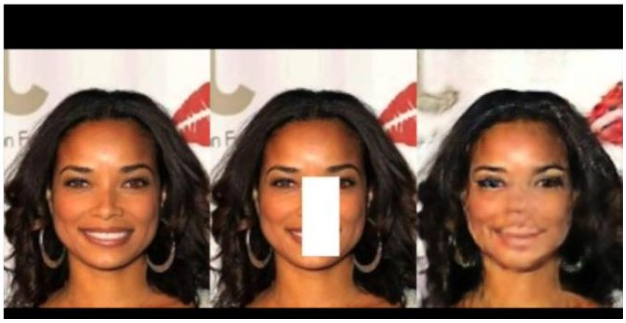
- کاهش ابعاد و یادگیری ویژگی: رمزگذارهای خودکار به ویژه برای کاهش ابعاد و وظایف یادگیری ویژگی موثر هستند. آنها می توانند ساختار زیرلایه ای داده های با ابعاد بالا را با یادگیری نمایش های فشرده در لایه های مخفی ثبت و ضبط کند. این کار آنها را قادر می سازد تا به طور مؤثر داده های ورودی را رمزگذاری و رمزگشایی کنند و ویژگی های مهم را حفظ کنند و در عین حال نویز و اطلاعات نامربوط را دور بیندازند. از سوی دیگر، DNN ها عمدتاً برای وظایف یادگیری نظارت شده استفاده می شوند و ممکن است به طور واضح نمایش فشرده ای از داده ها را یاد نگیرند.

- بازسازی و حذف نویز: رمزگذارهای خودکار برای بازسازی داده های ورودی از یک نمایش فشرده طراحی شده اند. این باعث می شود آنها برای کارهایی مانند حذف نویز تصویر، نقاشی داخلی و بازسازی مناسب باشند. با یادگیری ایجاد

بازسازی های تمیز و دقیق، رمزگذارهای خودکار به طور موثر نویز را فیلتر کرده و کیفیت خروجی را بهبود می بخشند. از سوی دیگر، DNN ها ممکن است به طور خاص برای کارهای بازسازی طراحی نشده باشند و ممکن است در سناریوهای حذف نویز یا بازسازی به خوبی عمل نکنند.

Denoising Autoencoders

Original Noisy Input Output



(مانند شکل روبرو)

- کارایی محاسباتی: رمزگذارهای خودکار می توانند در مقایسه با DNN ها از نظر محاسباتی کارآمدتر باشند زیرا تعداد پارامترهای آنها کاهش یافته است. از آنجایی که رمزگذارهای خودکار یک نمایش فشرده از ورودی را یاد می گیرند، تعداد پارامترها در لایه های پنهان معمولاً بسیار کمتر از DNN است. این منجر به آموزش سریع‌تر و زمان‌های استنتاج می‌شود و رمزگذارهای خودکار را برای مجموعه داده‌های مقیاس بزرگ کاربردی‌تر می‌کند.

توجه به این نکته مهم است که عملکرد و اثربخشی رمزگذارهای خودکار در مقابل DNN ها به وظایف خاص، مجموعه داده ها و طراحی معماری بستگی دارد. در حالی که رمزگذارهای خودکار در سناریوهای خاصی عالی هستند، DNN ها نقاط قوت خود را دارند و برای کارهای یادگیری نظارت شده با داده های برچسب گذاری شده فراوان مناسب تر هستند.

در حالی که نتایج امیدوارکننده هستند، هنوز جا برای بهبود بیشتر وجود دارد. تنظیم دقیق معماری مدل، کاوش در تکنیک‌های مختلف تنظیم، و ترکیب اطلاعات زمینه‌ای می‌تواند عملکرد سیستم توصیه را بهبود بخشد و محدودیت‌های بالقوه را برطرف کند.

در نتیجه، ترکیب DNN و Autoencoder یک رویکرد قدرتمند برای توسعه سیستم های توصیه دقیق و شخصی شده نشان داده است. نتایج پروژه ما اثربخشی این رویکرد را در ارائه توصیه‌های فیلم با کیفیت بالا برای کاربر ۱۰۰ نشان می‌دهد. از آنجایی که زمینه سیستم‌های توصیه به پیشرفت ادامه می‌دهد، ادغام تکنیک‌های یادگیری عمیق، مانند ترکیب رمزگذار خودکار با DNN، نوید بزرگی برای ارائه توصیه های دقیق تر و مرتبط تر در حوزه های مختلف می دهد.

منابع:

سایتها:

- https://www.mygreatlearning.com/blog/masterclass-on-movie-recommendation-system/
- https://towardsdatascience.com/tensorflow-for-recommendation-model-deep-learning-d9d4e826ea0b
- https://www.sciencedirect.com/science/article/abs/pii/S0957417421001366

کلاس ها:

- The Ultimate Beginners Guide to Python Recommender Systems_Udemy
- https://deeplearningcourses.com/ Movie recommendation system with deep learning

مقالات:

- Jeffrey Lund_Movie Recommendations Using the Deep Learning Approach
- Lakshit Sama, Hua Wang_Movie Recommendation System Using Deep Learning

- ZiXi Yao_Review of Movie Recommender Systems Based on Deep Learning
- Yongheng Mu_Multimodal Movie Recommendation System Using Deep Learning

- Suvash Sedhain†_AutoRec: Autoencoders Meet Collaborative Filtering

- Video Recommendations Based on Visual Features Extracted with Deep Assoc. Prof. Dr Mehdi Elahi_Learning

https://chat.openai.com (بابت ساختار بندی و جهت دهی منابع و سایت ها پر سود)

در نهایت کمال تشکر را از استاد راهنمای محترم سرکار خانم دکتر طباطبایی را دارم

-محمد احسان کریمی نوری