دانشکده مهندسی کامپیوتر

دکتر محمدرضا محمدی

بهار ۱۴۰۰

تمرین سری دهم

یادگیری عمیق

مجتبی نافذ 96431335

مهلت تحویل : ۸ خرداد ۱۴۰۰ ساعت ۲۳:۵۹:۵۹

---

۱. فرض کنید خروجی یک لایه به ازای N داده به صورت زیر باشد (ستون‌ها D و ردیف‌ها N می‌باشند). نتیجه batch normalization و layer normalization را محاسبه کنید (برای محاسبات مجاز هستید از توابع پایه numpy استفاده کنید).

29	39	75	38	35
60	31	49	27	96
48	26	13	29	90
65	97	62	34	11
46	71	76	40	87

کد ضمیمه شده است.

۲. در کدامیک از لایه‌های batch normalization، instance normalization، layer normalization و group normalization نیاز است تا میانگین و واریانس داده‌های آموزشی برای زمان تست ذخیره شوند؟

فقط در batch normalization این نیاز وجود دارد چون در بقیه ی حالت ها فقط روی یک داده از batch محاسبات انجام می شود و ما به راحتی در زمان تست از داده ی تست استفاده می کنیم و آن ها را محاسبه می کنیم..

۳. در این سوال هدف آشنایی بیشتر با مدل Word2Vec است. در این کد ابتدا میتوانید مدل word2vec از مدل از پیش آموزش داده شده load کنید. توجه کنید که مدل آموزش دیده در زبان انگلیسی است و در قسمت های بعد باید کلمات انگلیسی مورد نظر را انتخاب کنید.

الف) در این قسمت یک کلمه انتخاب کنید. پس از اجرای سلول قسمت A کلمات مشابه با کلمه مورد نظرتان را ارزیابی کنید که مدل تا چه میزان خوب عمل می‌کند و چه اشتباهاتی وجود دارد؟

ب) در قسمت B ابتدا یک کلمه انتخاب کنید. دو کلمه دیگر نیز انتخاب کنید به طوری که فاصله یکی از آن ها تا کلمه اول کم و دیگری زیاد باشد. آیا انتظاری که برای دور بودن و نزدیک بودن کلمات داشتید برآورده شد؟ اگه نه فکر می‌کنید مشکل چیست؟

پ) در قسمت C سه کلمه انتخاب کنید. هدف این است دنبال کلماتی بگردید که به کلمات اول و دوم شبیه و متفاوت از کلمه سوم باشد. همچنین کلماتی که به کلمات اول و سوم شبیه و متفاوت از کلمه سوم باشد. نتایج را ارزیابی کنید.

ت) نتایج حاصل از قسمت D را تحلیل کنید (توجه داشته باشید که Word2Vec یکی از روش‌های موجود برای یافتن word embedding است و قرار نیست کاملا مطابق با پیش‌بینی ما عمل کند).

الف) در کل مدل خوبی است. اما خوب اشکالاتی هم وارد است. مثلا:

```
word is 'iran'
most similar words are  [('tehran', 0.8018953204154968), ('iranian',
0.7916760444641113), ('syria', 0.682746171951294), ('nuclear',
0.6627756357192993), ('iranians', 0.654109537601471), ('iraq',
0.645396888256073), ('ahmadinejad', 0.6219896078109741), ('enrichment',
0.6144343614578247), ('libya', 0.6124347448348999), ('arabia',
0.6014441251754761)]
```

برای همین نمونه خیلی جالب است که در مورد ایران چه چیزهایی را فهمیده.

از لغات کاملا مشخص است کلمات سیاسی و اخبار روز را به عنوان داده ی training آموزش دیده. و جملات آن تعمیم دهی خوبی ندارد. و حول یک موضوع است.

غنی سازی، سوریه ، هسته ای ، احمدی نژاد، ... و مشکلی که در این نمونه دارد این است در واقع کلمات در context خود را cluster کرده است اما لزوما این کلاسترینگ به دلیل مترادف معنایی بودن نیست و فقط نشان میدهد این کلمات در جمله ها با هم آمده اند.(دلیل اصلی کمبود ورودی بوده و مدل واقعا خوب یادگرفته است.)

ب) خیر برآورده نشد.

Distance 'word' and 'world' is 0.7419539988040924
Distance 'word' and 'homework' is 0.797638937830925

انتظار می رفت که word, world خیلی به هم نزدیک باشند و word, homework خیلی از هم دور باشند.اما همانطور که میبینیم خیلی تفاوت زیاد و فاحشی ندارند.

خوب اولا این نشان دهنده یادگرفتن مفاهیم است و عالی به نظر میرسد.

اما علت آن این است که ما در word2vec به کاراکتر ها توجهی نداریم و کلماتی که در جملات مشترکی با هم به کار رفته اند برای ما ارزش بالایی دارند و در این جا هم چون با هم به کار نرفته اند پس مشابه نیستند.

(زمانی که با هم به کار نروند embedding متفاوت خواهند گرفت.

```
Distance 'father' and 'brother' is 0.1406240463256836
Distance 'father' and 'homework' is 0.8055841624736786
```

پ) مثال:

```
[('[('son', 0.7970596551895142), ('uncle', 0.7218263149261475),
('grandfather', 0.720313310623169), ('nephew', 0.7136063575744629),
('grandson', 0.7050833702087402), ('his', 0.6235296726226807), ('himself',
0.6094051003456116), ('sons', 0.6067705154418945), ('elder',
0.5954800844192505), ('cousin', 0.5950334072113037)]
mother', 0.8504593968391418), ('daughter', 0.7459856867790222),')]
('wife', 0.7339885234832764), ('grandmother', 0.7117711305618286), ('she',
0.7019714713096619), ('her', 0.7003562450408936), ('daughters',
0.6821584701538086), ('husband', 0.6506712436676025), ('aunt',
[(0.6474298238754272), ('niece', 0.6314456462860107
```

به نظر من خیلی خوب این مورد را جواب داد

و دقیقا توانست جنسیت را تشخیص دهد البته کمی مشکل ریز دارد که cousine را هم در مذکر قرار داده، اما خیلی مشکل بزرگی نیست.

واین نشانگر خوب بودن یادگیری است.(embedding ها خوب هستند)

ت)

```
print(wv_from_bin.most_similar(positive=['many', 'cow'],
                               negative=['one']))
```

خروجی:

```
[('cows', 0.6353424787521362), ('cattle', 0.5631539225578308), ('mad',
0.5619195699691772), ('sheep', 0.5586581230163574), ('pigs',
0.557633638381958), ('bovine', 0.5567370653152466), ('herds',
0.5566994547843933), ('beef', 0.5285123586654663), ('chickens',
0.5182628631591797), ('bse', 0.5022904276847839)]
```

```
print(wv_from_bin.most_similar(positive=['young', 'cow'],
negative=['old']))
```

خروجی:

```
[('mad', 0.6016051769256592), ('cows', 0.5750541687011719), ('sheep', 0.5211982131004333), ('pigs',
0.517691969871521), ('herd', 0.5136926174163818), ('bovine', 0.5008134245872498), ('cattle',
0.48994070291519165), ('infected', 0.48399820923805237), ('spongiform', 0.4800351858139038),
('animals', 0.4769159257411957)]
```

همانطور که می بینیم cow در هر دو، حرف اول را میزند و همه ی کلمات ارتباط معنایی خاصی با آن دارند.

و many , one یا young , old تاثیر چندانی ندارد

در واقع many , one کلمات هم راستایی دارند ولی چون many , one کاملا ارتباط معنایی دارند پس کلمات مشابه آنها تقریبا یکسان است.

و هم دیگر را خنثی می کنند و فقط cow مهم می شود.

پس مدل ما به many , one به علت ارتباط معنایی، embedding مشابهی نسبت می دهد و آن ها چون embedding شان در یک حوالی است در نتیجه کلمات مشابه آن ها هم یکسان است و در صورت استفاده از negative, positive هم را خنثی می کنند(young , old هم همچنین)

۴. در این سوال می‌خواهیم یک مثال ساده از یادگیری ویژگی‌های بصری با استفاده از رویکرد یادگیری خودنظارتی را پیاده‌سازی کنیم. مراحل زیر را بر روی مجموعه داده CIFAR10 انجام دهید. برای حل این تمرین یک شبکه با قابلیت یادگیری بالا با استفاده از لایه‌های کانولوشنی و دیگر لایه‌های خوانده شده طراحی کنید و تمام مراحل زیر را با استفاده از آن انجام دهید. در این آزمایش، از داده‌های آموزشی هر کلاس تنها ۲۰ داده را دارای برچسب نگه می‌داریم و باقی داده‌ها را بدون برچسب استفاده خواهیم کرد. به عبارت دیگر، در مجموع ۲۰۰ داده آموزشی دارای برچسب و ۴۹۸۰۰ داده آموزشی بدون برچسب برای آموزش مدل خواهیم داشت و ۱۰۰۰۰ داده تست دارای برچسب برای ارزیابی مدل خواهیم داشت (در این کد، نحوه آماده‌سازی داده‌ها مشخص شده است).

الف) مدل خود را تنها با استفاده از داده‌های آموزشی دارای برچسب آموزش دهید و بر روی داده‌های تست ارزیابی کنید.

ب) با استفاده از داده‌های آموزشی بدون برچسب، مسئله تشخیص زاویه تصویر را حل کنید. سپس، لایه انتهایی شبکه را حذف کرده و بجای آن یک لایه دارای ۱۰ نورون برای دسته‌بندی قرار دهید و مدل خود را با این وزن‌های اولیه و با استفاده از داده‌های آموزشی دارای برچسب آموزش دهید (با نرخ آموزش کوچکتر) و ارزیابی کنید.

پ) مدل خود را به گونه‌ای تغییر دهید که دارای دو خروجی باشد (یک خروجی برای دسته‌بندی زاویه و یک خروجی برای دسته‌بندی ۱۰ کلاسه). سپس، مدل خود را با تمام ۵۰۰۰۰ داده آموزشی آموزش دهید (۴۹۸۰۰ نمونه از داده‌ها دارای برچسب نیستند و بنابراین برای این داده‌ها خروجی مطلوب دسته‌بند ۱۰ کلاسه را برابر با بردار صفر قرار دهید تا اثری روی تابع ضرر آن نداشته باشند). مدل آموزش دیده را بر روی داده‌های تست ارزیابی و با نتایج قبل مقایسه کنید. در این حالت، میزان اثر هر تابع ضرر باید به درستی تنظیم شود (با توجه به کم بودن داده‌های دارای برچسب، اثر آنها در مجموع کم خواهد بود). چند ضریب مختلف برای تابع ضرر تخمین زاویه را امتحان کنید و نتایج خود را با دقت تحلیل کنید.

* برای تعریف یک مدل با چند خروجی می‌توانید از مدل functional در keras استفاده کنید. همچنین، برای تعیین وزن هر کدام از توابع ضرر می‌توانید از loss_weights در هنگام compile مدل استفاده کنید. برای راهنمایی بیشتر می‌توانید از این لینک کمک بگیرید (البته توجه داشته باشید که در مسئله ما، فقط ورودی دو مسئله مشترک نیست بلکه بخش عمده شبکه CNN برای دو مسئله مشترک است).