هوش مصنوعی
تمرین شماره‌ی x

مدرسین: **دکتر فدایی و دکتر یعقوب‌زاده**

طراحان: پویا صادقی، مجید فریدفر، متین بذرافشان

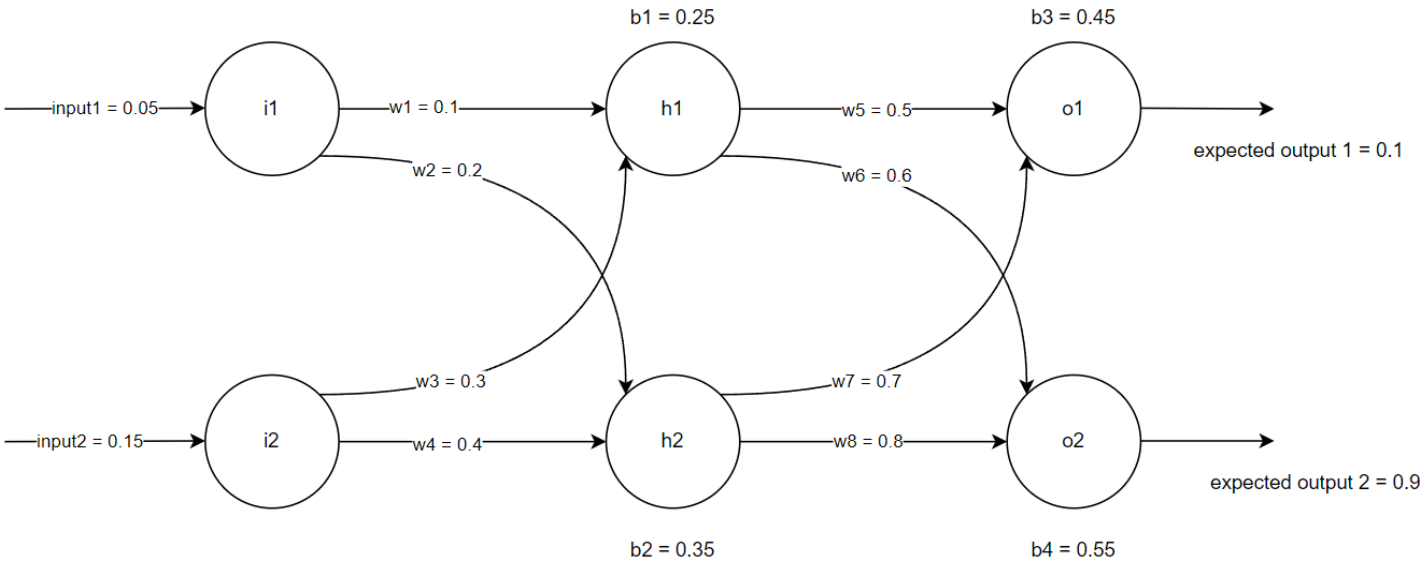
مهلت تحویل: ۰۹ خرداد ۱۴۰۳، ساعت ۲۳:۵۹

بخش کتبی

بهینه‌سازی

سوال اول

شبکه عصبی زیر را در نظر بگیرید:



اگر تابع فعال‌ساز هر نورون آن sigmoid باشد و دارای تابع هزینه $E = \frac{1}{2}\sum_i (target_i - output_i)^2$ باشد. به سوالات زیر پاسخ دهید.

الف) خروجی نورون‌های o1 و o2 را به‌دست آورید.

ب) هزینه کلی[1] را محاسبه کنید.

ج) با استفاده از backward propagation و قواعد مشتق زنجیره‌ای، مقدار جدید w5 را محاسبه کنید. نرخ آموزش[2] را 0.5 در نظر بگیرید.

شبکه‌های عصبی

سوال اول

به [این سایت] که در درس نیز به آن اشاره شد، مراجعه کنید. از قسمت دادگان Circle را انتخاب کنید و دیگر بخش‌ها را بدون تغییر باقی بگذارید. حال به سوال‌های زیر پاسخ دهید. لازم است بدانید در این مدل، لایه خروجی مقادیر را به بازه 1- تا 1 اسکیل می‌کند و خروجی مثبت را برچسب آبی و خروجی منفی را برچسب نارنجی می‌زند.

الف) فقط فیچرهای X_1^2 و X_2^2 را انتخاب کنید. لایه‌های پنهان را حذف کنید و شبکه را آموزش دهید. به‌نظرتان چرا شبکه به درستی کار می‌کند؟

ب) فقط فیچرهای X_1 و X_2 را انتخاب کنید. یک لایه پنهان با 3 نورون را به شبکه اضافه کنید و تابع فعال‌ساز ReLU را انتخاب کنید و شبکه را آموزش دهید. با استفاده از وزن‌ها و بایاس‌های به‌دست آمده، مرز تصمیم هر نورون لایه پنهان را رسم کنید. با استفاده از مرزهای تصمیم و وزن‌های لایه خروجی، علامت بایاس لایه خروجی را پیدا کنید. سپس با استفاده از آن و دیگر اطلاعات شبکه، مرز تصمیم شبکه را به صورت ریاضی توجیه کنید.(لازم به محاسبه معادلات دقیق مرزها نیست، کافی است به طور تقریبی مرز تصمیم شبکه را اثبات کنید.)

پ) فقط فیچرهای X_1 و X_2 را انتخاب کنید. یک لایه پنهان با 3 نورون را به شبکه اضافه کنید، آموزش را با دو تابع فعال‌ساز Tanh و Sigmoid انجام دهید. تفاوت مرز تصمیم شبکه در این دو حالت را با حالت ReLU توجیه کنید.

نکته 1: آموزش را به اندازه‌ای ادامه دهید که مقادیر وزن‌ها ثابت شده شوند.
نکته 2: با توجه به یکسان نبودن این فرآیند در هر بار اجرا، تصاویر و عددهای مشاهده شده را حتما گزارش کنید.

---

[1] Total Loss
[2] Learning Rate

شبکه‌های عصبی پیچشی[3]

سوال اول

حداقل دو دلیل برای ترجیح استفاده از لایه‌های پیچشی و ادغام به جای لایه‌های کاملا متصل برای استخراج ویژگی‌های یک عکس، بیاورید.

سوال دوم

همانطور که می‌دانید، پیچش[4] را می‌توان برای بردارهای تک بعدی نیز انجام داد. برای این‌کار علاوه بر ورودی، فیلتر نیز باید تک بعدی باشد. فرض کنید فیلتر f به ابعاد 3 × 1 را به سیگنال

[1, 4, 0, -2, 3]

را بدون حاشیه‌گذاری[5] و با گام[6] 1 اعمال کرده‌ایم. اگر خروجی

[-2, -2, 11]

باشد، فیلتر f را به‌دست آورید.

بخش عملی

مقدمه

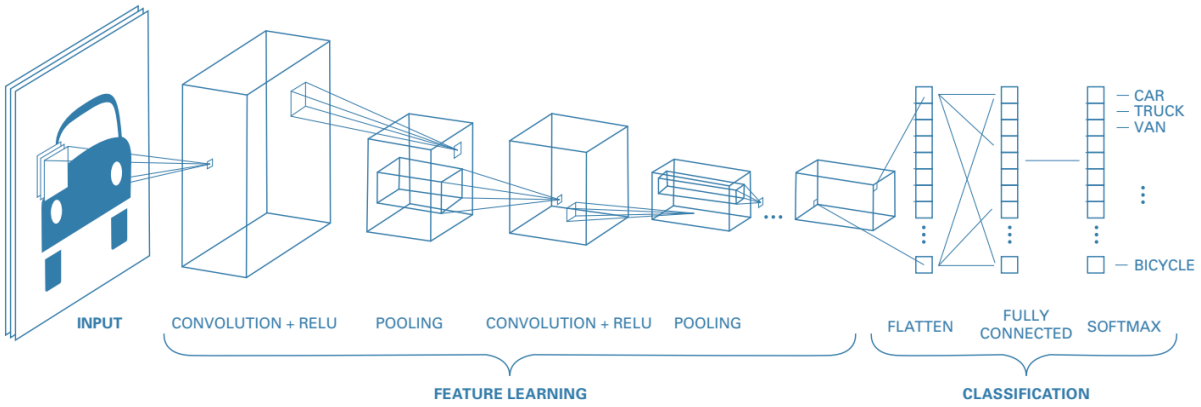
شبکه‌های عصبی پیچشی - Convolutional Neural Networks (CNNs)

شبکه‌های عصبی پیچشی (CNN) نوعی مدل یادگیری عمیق هستند که به طور گسترده و به خصوص برای پردازش تصویر و ویدیو استفاده می‌شوند. این مدل‌ها برای شناسایی موثر ویژگی‌های دارای همبستگی[7] مکانی با استفاده از مفهوم پیچش طراحی شده‌اند. CNN ها معمولا از چندین لایه شامل

---

[3] Convolutional Neural Networks
[4] Convolution
[5] Padding
[6] Stride
[7] Correlation

لایه‌های پیچش، لایه‌های ادغام[8] و لایه‌های کاملا متصل[9] تشکیل می‌شوند. لایه‌های پیچش، فیلترهایی را روی ورودی اعمال می‌کنند و ویژگی‌ها را در مکان‌های مختلف ورودی استخراج می‌کنند. لایه‌های ادغام ابعاد فضایی ویژگی‌ها را کاهش می‌دهند و لایه‌های کاملاً متصل بر اساس ویژگی‌های استخراج شده، عملیات طبقه‌بندی را انجام می‌دهند. این معماری سلسله مراتبی، CNN ها را قادر می‌سازد تا الگوهای پیچیده را به طور خودکار یاد بگیرند. این قابلیت مدل‌های پیچشی، آنها را در کارهایی مانند طبقه‌بندی تصویر، تشخیص اشیا و بخش‌بندی[10] تصویر بسیار پرقدرت می‌سازد.



شکل ۱. ساختار کلی یک شبکه‌ی پیچشی

جاسازی کلمه - Word Embedding

جاسازی کلمه، تکنیکی در پردازش زبان طبیعی (NLP) است که کلمات را به بردارهای عددی تبدیل می‌کند. این بردارها معنی و بافت کلمات را به تصویر می‌کشند. هدف اصلی جاسازی کلمه، تبدیل فضاهای ویژگی با ابعاد بالا از کلمات به بردارهای ویژگی با ابعاد پایین و در عین حال حفظ شباهت متنی در پیکره است. این بردارها می توانند به عنوان ویژگی های ورودی برای مدل های یادگیری ماشین برای بهبود عملکرد آنها در وظایف مختلف NLP استفاده شوند.

مفهوم جاسازی کلمات در طول زمان تکامل یافته است و روش‌های پایه بسیاری در توسعه آن سهیم بوده‌اند. در اینجا یک بررسی اجمالی بر روی روش‌های پیشین انجام می‌دهیم:

---

[8] Pooling

[9] Fully-Connected

[10] Segmentation

- One-Hot Encoding: یکی از اولین روش‌ها برای نمایش کلمات به‌عنوان بردار بود، که در آن هر کلمه به‌عنوان یک بردار باینری با یک 1 (که بیان‌گر کلمه مدنظر بود) و همه عناصر دیگر با 0 نمایش داده می‌شد. به عنوان اولین روش نمایش کلمات با بردار، مزیت‌های بر روش‌های غیر برداری مثل ترتیبی نبودن و مستقل بودن کلمات (عمود بودن) را ارائه می‌کرد. با این حال، این رویکرد دارای محدودیت‌هایی مانند ابعاد بالا و ناتوانی در گرفتن روابط معنایی بین کلمات بود.
از معایب این روش، ابعاد بالای بردارها می‌باشد. برای مثال، اگر ۵۰۰۰ کلمه داشه باشیم (که در واقعیت معمولا بیشتر از این مقدار می‌باشد.) هر بردار طول ۵۰۰۰ خواهد داشت که از نظر مصرف منابع بسیار پرهزینه می‌باشد. همچنین روابط معنایی بین کلمات در نظر گرفته نمی‌شود. به عنوان مثال کلمات apple و orange مربوط به دسته میوه‌ها می‌باشند و انتظار می‌رود در فضای برداری، به هم نزدیک‌تر باشند و هر دو نسبت به ماشین که یک وسیله است، فاصله بیشتری داشته باشند، که در این نمایش، این‌گونه نمی‌باشد و تمامی بردارها بر هم عمودند.
- Sparse Representations: برای پرداختن به محدودیت‌های رمزگذاری one-hot، نمایش‌های پراکنده[11] معرفی شدند. این نمایش ها از بردار با تعداد کمی از عناصر غیر صفر (بیان‌گر یک ویژگی) برای نمایش هر کلمه استفاده می کردند. در حالی که نمایش های پراکنده یک پیشرفت بود، آنها همچنان محدودیت هایی مانند عدم توانایی در نظر گرفتن روابط بین کلمات را دارا بودند.
- Dense Representations: گذار به نمایش‌های متراکم، جهشی قابل توجه در جاسازی کلمات را نشان داد. نمایش‌های متراکم از بردارهایی با بسیاری از عناصر غیر صفر برای نمایش هر کلمه استفاده می‌کردند که امکان گرفتن روابط غنی‌تر بین کلمات را فراهم می‌کرد.
- Word2Vec و GloVe: توسعه Word2Vec و GloVe نقطه عطف قابل توجهی در جاسازی کلمات است. این الگوریتم‌ها از نمایش‌های متراکم استفاده می‌کردند و می‌توانستند روابط معنایی بین کلمات، مانند قیاس‌ها و مترادف‌ها را ثبت کنند. در این پروژه ما از جاسازی از پیش آموزش دیده Word2Vec استفاده می‌کنیم.

به طور کلی، نیاکان و پایه‌های جاسازی کلمه را می‌توان روش‌های One-Hot Encoding و Sparse Representations و Latent Semantic Analysis-LSA و Harris's Distributional Hypothesis دانست.

ولی چرا جاسازی کلمه به طور گسترده در NLP استفاده می شود؟

1. گرفتن روابط معنایی: جاسازی کلمات می تواند روابط معنایی بین کلمات مانند مترادف ها، متضادها و قیاس ها را به تصویر بکشد.

[11] Sparse

2. کاربرد در یادگیری ماشین: خروجی عددی جاسازی کلمات، الگوریتم‌های یادگیری ماشین را قادر می‌سازد تا داده‌های متنی را پردازش و تجزیه و تحلیل کنند.
3. بهبود عملکرد: نشان داده شده است که جاسازی کلمات عملکرد وظایف مختلف NLP مانند مدل سازی زبان، طبقه بندی متن و تجزیه و تحلیل احساسات را بهبود می بخشد.

روش‌های دیگری نیز همچون Latent Dirichlet Allocation-LDA و Non-Negative Matrix Factorization-NMF و Deep learning-based methods وجود دارند که می‌توانید درباره آنها مطالعه کنید. در ادامه ما به جاسازی کلمه w2v و glove میپردازیم و با کمک کتابخانه gensim، آزمایش‌هایی بر روی آن‌ها انجام خواهیم داد.

در جایگذاری کلمات، در نمایش بردارهای فشرده در فضای برداری، می‌توان مشاهده کرد کلماتی که معانی نزدیک دارند، به یکدیگر نزدیک و از کلمات به معانی متفاوت دور هستند:

می‌توانیم کلمات معنایی مشابه به هر کلمه را استخراج کنیم. به عنوان مثال:

```
model = KeyedVectors.load_word2vec_format(input_file, binary=False,
no_header=True)
semantically_similar_words = {
    "computer": model.most_similar("computer", topn=5),
    "football": model.most_similar("football", topn=5),
    "ocean": model.most_similar("ocean", topn=5),
    "music": model.most_similar("music", topn=5),
}
```

خروجی قطعه کد بالا:

```
Similar words to computer:
      (computer) <-> computers        : 0.835772
      (computer) <-> software         : 0.782846
      (computer) <-> technology       : 0.690766
      (computer) <-> pc               : 0.664798
      (computer) <-> systems          : 0.658432
Similar words to football:
```

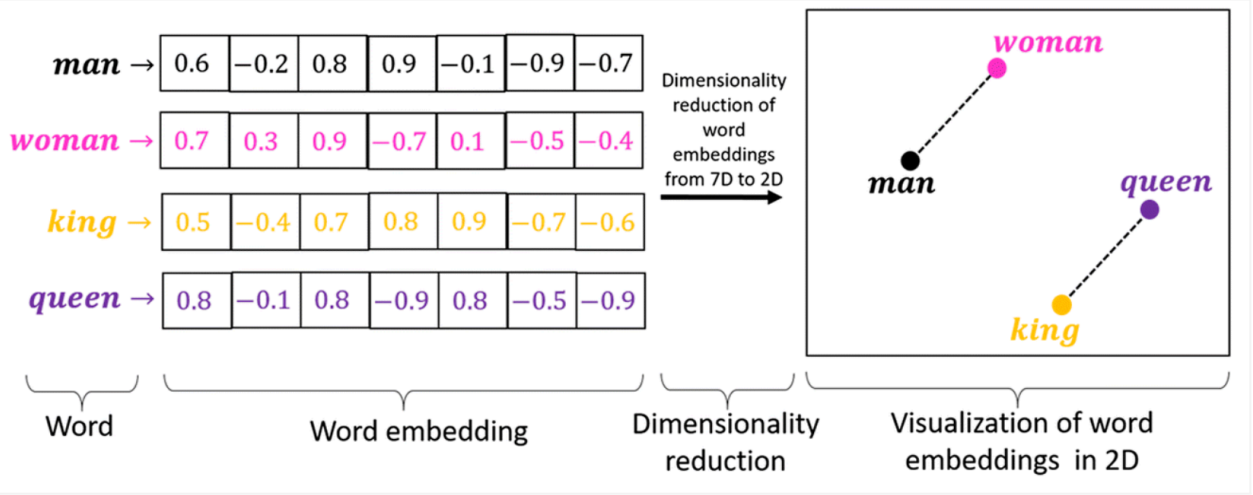
```
      (football) <-> soccer          : 0.810518
      (football) <-> basketball      : 0.789614
      (football) <-> league          : 0.716710
      (football) <-> baseball        : 0.702313
      (football) <-> rugby           : 0.700695
Similar words to ocean:
      (ocean) <-> sea              : 0.727745
      (ocean) <-> waters           : 0.724155
      (ocean) <-> coast            : 0.697292
      (ocean) <-> atlantic         : 0.691838
      (ocean) <-> seas             : 0.681202
Similar words to music:
      (music) <-> musical          : 0.733881
      (music) <-> songs            : 0.725357
      (music) <-> pop              : 0.690601
      (music) <-> musicians        : 0.687654
      (music) <-> recording        : 0.684866
```

مشاهده می‌شود کلماتی که از نظر معنایی با هم رابطه دارند، در اینجا نیز مشابه و نزدیک قرار گرفته‌اند.

قیاس[12] در تعبیه کلمه به انجام عملیات جبری بر روی بردارها برای گرفتن شبیه ترین کلمات به جاسازی حاصل اشاره دارد. یکی از معروف ترین نمونه ها قیاس $king - man + woman = queen$ است.

در gensim پارامتر positive نمایانگر جمع و negative نمایانگر تفریق است. به عنوان مثال در قیاس مطرح شده:

```
model.most_similar(positive=["king","queen"], negative=["woman"],
topn=3)
```



چند نمونه قیاس دیگر:

```
Analogy: king - man + woman:
```

[12] Analogy

```
(1)   queen (0.697868)
(2)   princess (0.608175)
(3)   monarch (0.588975)
Analogy: king - queen + woman:
(1)   man (0.717850)
(2)   person (0.605025)
(3)   father (0.599216)
Analogy: paris - france + italy:
(1)   rome (0.758541)
(2)   milan (0.685050)
(3)   italian (0.664783)
Analogy: dog - puppy + cat:
(1)   dogs (0.627271)
(2)   cats (0.592487)
(3)   horse (0.556448)
Analogy: car - road + ship:
(1)   vessel (0.646534)
(2)   ships (0.602428)
(3)   boat (0.591753)
```

توضیح مسئله

در این تمرین، کار با داده‌های متنی را خواهید دید. در ابتدا به پیش‌پردازش داده‌های متنی می‌پردازید. در بخش بعدی با نحوه استفاده از مدل word2vec آشنا می‌شوید و جملات را آماده ورود به شبکه عصبی می‌کنید. در بخش دوم برای طبقه‌بندی جملات، با استفاده از کتابخانه‌ی PyTorch یک شبکه عصبی پیچشی پیاده‌سازی می‌کنید. در بخش سوم اثر اندازه پنجره متن[13] در یادگیری شبکه را مشاهده خواهید کرد. در بخش چهارم هم به بررسی تاثیر روش‌های منظم‌سازی[14] در فرایند آموزش خواهید پرداخت.

برای راحتی استفاده از کتابخانه‌ها و تسریع فرآیند آموزش، می‌توانید از سرویس **Google Colab** استفاده کنید. توجه کنید که آموزش شبکه‌های عمیق (به خصوص شبکه‌های پیچشی) روی GPU بسیار سریع‌تر از CPU می‌باشد. در نتیجه می‌توانید Runtime Type سرویس Colab را روی GPU قرار دهید. دقت داشته باشید که مدت زمان استفاده از GPU سرویس Colab به ازای هر اکانت، محدود می‌باشد. در نتیجه مراحل پیش‌پردازش و تعریف مدل خود را روی CPU انجام داده و فقط حین آموزش مدل از GPU استفاده کنید تا مدت زمان استفاده از GPU اختصاصی شما به پایان نرسد. همچنین استفاده از کارت

[13] Context Window

[14] Regularization

گرافیک کامپیوتر خودتان به وسیله CUDA و cuDNN برای کارت‌های گرافیک پشتیبانی شده ممکن است؛ البته این روش مشکلات و محدودیت‌هایی نیز به همراه دارد.

معرفی مجموعه داده

مجموعه دادگان مورد استفاده در این تمرین، یک مجموعه داده متنی به منظور تشخیص افکار خودکشی در شبکه‌های اجتماعی است. این مجموعه داده، ملقب به «twitter-suicidal-data» مجموعه‌ای از پست‌های توییتر[15] است که با برچسب‌هایی نشان‌دهنده افکار خودکشی یا محتوای فاقد افکار خودکشی است. این مجموعه داده شامل توییت‌هایی است که هر توییت به‌عنوان یک قطعه متن نمایش داده می‌شود. همچنین، هر توییت با یک مقدار قصد[16] برچسب گذاری شده است: 1 (خودکشی) یا 0 (غیر خودکشی). شایان ذکر است که این مجموعه داده و مجموعه دادگان مشابه در حوزه پردازش زبان طبیعی[17] و یادگیری عمیق به صورت ویژه‌ای مهم می‌باشند، زیرا به یک مسئله اجتماعی مهم می پردازند: پیشگیری از خودکشی. افکار خودکشی یک نگرانی رو به رشد است و تشخیص زودهنگام می تواند گامی مهم در جلوگیری از اقدام به خودکشی باشد. پلتفرم‌های رسانه‌های اجتماعی مانند توییتر به وسیله‌ای ضروری برای بیان احساسات و افکار مردم تبدیل شده‌اند. این بدان معناست که رسانه‌های اجتماعی می‌توانند بینش‌های ارزشمندی در مورد سلامت روانی افراد ارائه دهند.

مجموعه داده‌های متنی استخراج‌شده از شبکه‌های اجتماعی، مانند مجموعه داده مورد بحث، با ماهیت نویزی و بدون ساختار مشخص می‌شوند. آنها اغلب حاوی اختصارات، لحن عامیانه و زبان غیر رسمی هستند که می تواند تجزیه و تحلیل و استخراج اطلاعات معنی‌دار را چالش برانگیز کنند. علاوه بر این، پست‌های رسانه‌های اجتماعی می‌توانند کوتاه باشند و شناسایی الگوها یا شاخص‌های واضح افکار خودکشی را دشوار می‌کنند. در این تمرین می‌خواهیم از قدرت شبکه‌های عصبی پیچشی برای غلبه بر این چالش‌ها استفاده کنیم. این مجموعه دادگاه را از این لینک می‌توانید دریافت کنید.

بخش صفر: آماده کردن داده

استفاده مجموعه دادگان استخراج شده از شبکه‌های اجتماعی، معمولا با چالش‌های زیر همراه است:

- متن نویزی: توییت ها اغلب کوتاه، غیررسمی و حاوی اختصارات، عامیانه و غلط املایی هستند.
- وجود ابهام در متن: درک کانتکست ضروری است. برخی از توییت‌ها ممکن است از عبارات مربوط به خودکشی بدون بیان قصد واقعی استفاده کنند. در متن‌های شبکه‌های اجتماعی، بیان کنایی و استعاری بسیار به کار برده می‌شود.

---

[15] شبکه اجتماعی که امروزه به نام "X" شناخته می‌شود.

[16] Intention

[17] Natural Language Processing

در این قسمت پیش‌پردازش‌های مورد نیاز برای داده‌های متنی بیان می‌شود. در ابتدا تلاش می‌کنیم متن را تمیز کنیم و بخش‌هایی از متن که به ما اطلاعات مفیدی نمی‌دهند را حذف کنیم، چرا که مدل‌های یادگیری عمیق ساده امکان دارد ویژگی‌های سطحی را به عنوان میانبر شناسایی کرده و یادگیری مناسبی را انجام ندهند.

مراحل پیش پردازش متن به صورت زیر هستند. یک تابع بنویسید که با گرفتن یک متن ورودی (یک توییت) این پردازش‌ها را بر روی آن انجام دهد و سپس یک لیست از توکن[18]های ایجاد شده را خروجی دهد:

- تبدیل متن به فرمت lowercase
- حذف علائم نگارشی (punctuations)
- حذف اعداد
- حذف لینک‌ها (ممکن هست url ها به فرمت‌هایی با شروع http, https, www دیده شوند)
- حذف ارجاع به سایر کاربران شبکه اجتماعی (در شبکه توییتر این ارجاعات به فرمت $< username > @$ می‌باشند.)
- حذف ایموجی‌ها[19] و جایگذاری آن‌ها با اسپیس
- یکسان سازی وایت‌اسپیس‌ها (ممکن است کاربران شبکه‌های اجتماعی چند وایت‌اسپیس به صورت مکرر استفاده کنند و یا در اثر پیش‌پردازش‌های قبلی چندین اسپیس مکرر ایجاد شود که باید به یک اسپیس تبدیل شوند.)
- حذف عبارات غیر کلمه (فقط کلمات نگه‌داری شوند و ترم های غیر کلمه حذف شوند.)
- توکنایز[20] کردن و ریشه‌یابی توکن‌ها (بردن توکن‌ها به فرمت ریشه)
- حذف کلمات توقف[21]
- توکن‌های باقی مانده را به صورت لیستی از رشته‌ها خروجی دهید.

همچنین به سوالات زیر پاسخ دهید:

- درباره مزایا و معایب تبدیل متن به فرمت lowercase به صورت خلاصه توضیح دهید و بگویید در نهایت چرا این پردازش را انجام می‌دهیم؟
- درباره حذف اعداد در پردازش‌های بالا تحقیق کنید و مزایا و معایب این پردازش را نام ببرید.
- در شبکه اجتماعی توییتر، قابلیت هشتگ‌گذاری داریم. توضیح دهید چرا این عبارات را حذف نکردیم و نگه‌داشتن آنها چه تاثیری بر عملکرد مدل دارد؟

---

[18] Token

[19] برای این منظور می‌توانید از کتابخانه emoji و فراخوانی demojize استفاده کنید.

[20] Tokenize

[21] Stop Words

- بر روی چند نمونه‌ی دلخواه از دیتاست داده شده، تابع پیاده‌سازی شده را اعمال کنید و متن توییت را قبل و بعد پیش‌پردازش نمایش داده و مقایسه کنید.
- پس از تمیز کردن مجموعه داده و انجام مراحل ذکر شده، نمودار توزیع تعداد توکن‌های هر نمونه را برای برچسب 1 و 0 و همچنین در سطح کل مجموعه دادگان ترسیم کنید. برای نمایش بهتر از نمودار جعبه ای استفاده کنید و یا مقادیر کمینه، بیشینه و میانگین را چاپ کنید.

بخش اول: ساختن بردارهای ویژگی[22]

برای دادن ورودی به شبکه عصبی، باید هر ورودی را به یک بردار عددی تبدیل کنیم. ابتدا مدل Word2Vec داده شده را بارگیری کنید. این مدل حاوی یک دیکشنری است که هر کلمه را به یک بردار عددی 300 تایی نگاشت می‌کند. حال باید برای هر ورودی، بردار مختص خودش را بسازیم. در این بخش از روش الحاق[23] استفاده می‌کنیم، روش‌های دیگری مانند روش تجمیع[24] نیز وجود دارند که مورد توجه این پروژه نیستند. در روش الحاق، برای هر جمله یک بردار دو بعدی (ماتریس) در نظر گرفته می‌شود. برای مقداردهی این ماتریس، به ترتیب و به ازای هر توکن در جمله، بردار نظیر آن در word2vec در ماتریس مدنظر قرار داده می‌شود. دقت داشته باشید ابعاد ماتریس ساخته‌شده باید برای همه‌ی ورودی‌ها یکسان باشد، به همین علت پیش از شروع، باید سایز ماتریس را تعریف کنید. در این مرحله، می‌توانید بعد نشان‌دهنده تعداد توکن ماتریس را به صورت دلخواه مقدار دهی کنید. پیشنهاد می‌شود آن را برابر 64 قرار دهید. در **بخش سوم**، به بررسی این موضوع خواهید پرداخت که تغییر این مقدار چه تاثیری در عملکرد مدل می‌گذارد. هم‌چنین هنگام مقدار دهی ماتریس دو بعدی مربوط به کلمات، هرجا که به یک توکنی رسیدید که در دیکشنری word2vec موجود نبود، از یک بردار صفر به طولی برابر با بردارهای word2vec (در اینجا، 300×1) استفاده کنید.

- توضیح دهید چه روش‌هایی برای برخورد با کلمات ناموجود در دیکشنری ذکر شده وجود دارد و مزایا و معایب هر یک را نام ببرید.

حاشیه‌گذاری

حاشیه‌گذاری یا پدینگ، یک روش برای تصحیح اندازه یک بردار می‌باشد. معمولا ابعاد بردارها در در طول شبکه دچار تغییراتی می‌شوند، از طرفی اندازه ورودی هر لایه ثابت می‌باشد و در طول استفاده از شبکه نمی‌توان آن را تغییر داد. به همین علت حاشیه‌گذاری یک روش کارآمد برای رسیدگی این موضوع می‌باشد. حاشیه‌گذاری انواع مختلفی دارد، اما از پرکاربردترین آن‌ها می‌توان به حاشیه‌گذاری صفر[25] اشاره کرد. در این روش به اطراف بردار مقادیر صفر اضافه می‌شود تا ابعاد یک بردار به اندازه مورد نیاز برسد.

---

[22] Feature Vectors
[23] Concatenate
[24] Aggregate
[25] Zero Padding

به این منظور در طول ساخت بردار دو بعدی مربوط به جملات، اگر جمله کوتاه بود و توکن‌های آن پیش از رسیدن به انتهای ماتریس (که سایز آن را از پیش تعیین کرده‌اید) تمام شد، در ادامه بردارهای صفر اضافه کنید تا طول ماتریس به طول مشخص شده برسد. همچنین اگر تعداد توکن‌های یک جمله بیشتر از مقدار مشخص‌شده بود، ادامه‌ی توکن‌ها را حذف[26] کنید.

[26] truncate

بخش دوم: طبقه‌بندی با استفاده از یک شبکه‌ی CNN

در بخش دوم این پروژه، می‌خواهیم از یک شبکه پیچشی برای طبقه‌بندی بردارهای ساخته شده استفاده کنیم.

همانطور که توضیح داده شد، شبکه‌های پیچشی، با استفاده از لایه‌های پیچشی و لایه‌های ادغام قادرند اطلاعات مربوط به الگوها و ویژگی‌های مختلف در ورودی را استخراج کنند. لایه پیچشی با استفاده از عملیات پیچش، فیلترها را بر روی بردارها اعمال کرده و بردارهای جدید را ایجاد می‌کند که شامل ویژگی‌های محلی است. این لایه‌ها به صورت مکرر در سراسر شبکه استفاده می‌شوند تا ویژگی‌های سطح بالاتر را استخراج کنند.

لایه‌های ادغام به منظور کاهش ابعاد بردار و حذف اطلاعات کم اهمیت‌تر، استفاده می‌شوند. این لایه‌ها با استفاده از معیارهایی مانند حداکثرگیری[27] یا میانگین‌گیری[28] ، به کاهش ابعاد ویژگی‌های استخراج شده می‌پردازند.

در نهایت لایه‌های کاملا متصل، با استفاده از ویژگی‌های استخراج شده توسط لایه‌های پیچشی و ادغام، تصمیم‌گیری نهایی را برای طبقه‌بندی انجام می‌دهند. این لایه‌ها مشابه لایه‌های معمولی در شبکه‌های عصبی عمل می‌کنند و خروجی نهایی را تولید می‌کنند که شامل احتمال‌های مربوط به تعلق بردار ورودی به هر کلاس است.

برای شبکه‌ی عصبی، معماری‌های مختلفی را می‌توانید در نظر بگیرید. در ادامه یک شبکه‌ی پیچشی نمونه، توضیح داده شده است که می‌توانید آن را پیاده‌سازی کنید. اما توجه کنید الزامی برای استفاده از این معماری خاص برای این بخش وجود ندارد و می‌توانید مدل را تغییر داده و شبکه‌ی پیچشی خود را تعریف کنید.

Layer	Name	Input Size	Output Size	Activation Function	Kernel Size
1	Conv1D	300	64	ReLU	3
1	Conv1D	300	64	ReLU	5
1	Conv1D	300	64	ReLU	7
2	Conv1D	64	128	ReLU	3
2	Conv1D	64	128	ReLU	5
2	Conv1D	64	128	ReLU	7

[27] Max Pooling

[28] Average Pooling

3	Flatten				
4	Linear	3 * 128	128		
5	Linear	128	2		

هم‌چنین بعد از سطح دوم لایه‌های کانولوشن، از لایه بیشینه محلی[29] استفاده کرده و ابعاد ویژگی‌های استخراج شده را کاهش دهید. نحوه عملکرد آن به این صورت که به جای نگه‌داری تمام مقادیر به دست آمده از پیچش kernel ها روی ورودی به ازای هر فیلتر در لایه‌های کانولوشنی، فقط بزرگ‌ترین مقدار را در نظر می‌گیرد و با لایه بعدی می‌رساند و نوعی نقش فیلتر را بازی می‌کند که از انتقال سیگنال‌های ضعیف‌تر به لایه‌های بعد جلوگیری می‌کند.

در درس با بهینه‌ساز[30] SGD[31] آشنا شدید. این روش بهینه‌سازی با استفاده از روش گرادیان کاهشی به بهینه‌سازی وزن‌ها و پارامترهای شبکه‌ی عصبی می‌پردازد تا هزینه کلی کاهش یافته و دقت افزایش یابد. بهینه‌سازهای قدرتمندتر دیگری نیز وجود دارند. برای مثال یکی از معروف‌ترین آن‌ها، بهینه‌ساز **Adam** می‌باشد.

- به طور مختصر نحوه کار بهینه‌ساز Adam و تفاوت آن با بهینه‌ساز SGD را توضیح دهید.
- از تابع هزینه[32] Cross Entropy استفاده کنید. همچنین دلیل استفاده از این تابع هزینه را با توجه به ماهیت مسئله بگویید.

حالا داده‌ها را به دو قسمت آموزش[33] و آزمون[34] تقسیم کنید.

- نسبت این تقسیم‌بندی و علت استفاده شما از این نسبت را گزارش کنید.

سپس، مدل خود را با استفاده از پارامترهای گفته‌شده بسازید و آن را آموزش دهید و عملکرد شبکه را روی داده‌های آزمون بررسی کنید.

- تاثیر اندازه‌ی کرنل در لایه‌های کانولوشن چیست و چگونه در استخراج ویژگی‌های ورودی تاثیرگذار است؟ زیاد یا کم بودن آن به چه معناست؟

---

[29] Max Pooling
[30] Optimizer
[31] Stochastic Gradient Descent
[32] Loss Function
[33] Train
[34] Test

نکته 1: دقت کنید که در شبکه‌ی عصبی، نیاز است که پس از اتمام لایه‌های پیچشی، ویژگی‌های استخراج شده Flatten شده و وارد لایه‌های Linear برای طبقه‌بندی شوند.

- بنظر شما چرا خروجی کانولوشن را کاهش ندادیم و این کاهش را به واسطه لایه‌های Feed Forward انجام دادیم و این لایه چه مزایایی می‌تواند نسبت به روش‌های جایگزین داشته باشد؟ درباره‌ی دلیل این امر تحقیق کرده و نتایج را بیان کنید.

نکته 2: در تمامی مراحل تمرین، پس از اتمام تمامی epoch ها باید نمودار مقدار Loss و Accuracy برحسب epoch را رسم کنید.
نکته 3: در تمامی مراحل تمرین، پس از اتمام آموزش، باید معیارهای Precision, Recall, F1 برای داده‌های آموزش و آزمون گزارش شود .
نکته 4: در طول آموزش مدل خود، از batch size ها و نرخ‌های اولیه آموزش[35] مختلف استفاده کنید و تاثیر مقدار این پارامترها را گزارش کنید.
نکته 5: برای پیاده‌سازی شبکه عصبی، از لایه‌های آماده در کتابخانه‌ی PyTorch استفاده کنید. پیاده‌سازی شبکه عصبی از ابتدا اصلا مورد انتظار نیست. هم‌چنین در نظر داشته باشید که استفاده از کتابخانه‌ی دیگری برای پیاده‌سازی شبکه عصبی **مجاز نیست**.

[35] Initial Learning Rate

بخش سوم: تاثیر اندازه پنجره‌ی متن در یادگیری مدل

در این بخش، می‌خواهیم اثر پنجره متن یا Context Window را بررسی کنیم. پنجره‌های بزرگ‌تر باعث می‌شوند ما بتوانیم متنی‌های طولانی‌تری را به مدل ورودی بدهیم و پنجره کوچک‌تر از اندازه متن ورودی، باعث می‌شود بخشی از متن ورودی دور ریخته شود و اینگونه اطلاعات زیادی را از دست خواهیم داد. بنظر می‌رسد انتخاب پنجره‌های بزرگ‌تر یا انتخاب پنجره به اندازه طولانی‌ترین داده ورودی انتخاب مناسبی باشد؛ اما معمولا این اتفاق رخ نمی‌دهد و پنجره‌های ورودی متن در اندازه‌های کوچک‌تری ارائه می‌شوند. واقعیت این است که استفاده از پنجره‌های بزرگ چالش‌های بسیاری در زمان آموزش و همچنین زمان استفاده مدل خواهد داشت. یکی از این چالش‌ها افزایش تعداد پارامتر مدل است که مدل را سنگین تر می‌کند و استفاده از منابع را افزایش می‌دهد، اما چالش‌های موجود به همین موارد ختم نمی‌شوند.

- به نظر شما افزایش اندازه پنجره‌ی متن، به طوری که از تمام جملات دیتاست بیشتر باشد، در یک شبکه عصبی پیچشی چه مزایا و معایبی می‌تواند داشته باشد.
- به این منظور مدل قبلی را طوری تغییر دهید که پنجره‌هایی به اندازه 196 داشته باشد.
- به‌ازای اندازه‌های گفته‌شده، شبکه‌های عصبی پیچشی جدیدی را بسازید و تمامی مراحل خواسته‌شده در بخش قبلی را انجام دهید.
- نتایج به‌دست آمده را با بخش قبلی مقایسه کنید. آیا مشاهدات شما مطابق انتظار بود؟ آیا مدل به درستی آموزش دیده است؟ عملکرد مدل چگونه دچار تغییر شده است؟ توضیح دهید.

بخش چهارم: تاثیر روش‌های منظم‌سازی[36] در فرآیند آموزش

روش‌های منظم‌سازی در فرآیند آموزش شبکه عصبی مورد استفاده قرار می‌گیرند تا از بیش‌برازش[37] جلوگیری کنند. دو روش معروف در این زمینه Dropout و Batch Normalization هستند که به ترتیب در ادامه توضیح داده خواهند شد:

1. **Dropout**: این روش یکی از تکنیک‌های مؤثر برای کاهش بیش‌برازش در شبکه‌های عصبی است. این روش در فرآیند آموزش، به صورت تصادفی برخی از نورون‌ها را در هر مرحله غیرفعال می‌کند. به این ترتیب، هر نورون مجبور است الگوهای مفید را در حضور نورون‌های دیگر یاد بگیرد و به این ترتیب میزان وابستگی نورون‌ها به هم و همچنین میزان وابستگی شبکه به هر نورون کاهش می‌یابد. این باعث می‌شود که شبکه‌ی عصبی توانایی تعمیم‌پذیری بهتری پیدا کند و کمتر دچار بیش‌برازش شود. Dropout می‌تواند به عنوان یک لایه Dropout با ضریب احتمال Dropout خاصی در ساختار شبکه اعمال شود.
2. **Batch Normalization**: این تکنیک، یک روش استانداردسازی ورودی‌های هر دسته (batch) در یک لایه است. در هر بچ از داده‌ها، میانگین و واریانس ورودی‌ها محاسبه شده و سپس ورودی‌ها با استفاده از میانگین و واریانس محاسبه شده استانداردسازی شده و وارد لایه‌ی بعدی می‌شوند. این روش باعث می‌شود که توزیع ورودی‌ها در هر لایه بهبود یابد و بیش‌برازش کاهش یابد. Batch Normalization نیز می‌تواند به عنوان یک لایه در ساختار شبکه اعمال شود.

با توجه به توضیحات داده شده، استفاده از Dropout و Batch Normalization در شبکه عصبی، می‌تواند تاثیر مثبتی بر عملکرد شبکه در فرآیند آموزش داشته باشد. این روش‌ها موجب کاهش بیش‌برازش، بهبود تعمیم‌پذیری و عملکرد مدل در مجموعه داده‌های تست می‌شوند. با اعمال این روش‌ها، شبکه عصبی قادر خواهد بود الگوهای مفید را یاد بگیرد و همچنین مقاومت بیشتری در برابر داده‌های نویزی و تغییرات کوچک در ورودی‌ها خواهد داشت.

در این بخش، باید از لایه‌های Dropout و Batch Normalization بین لایه‌های کاملا متصل در شبکه خود استفاده کنید.

- به شبکه‌ی تعریف شده‌ی خود در **بخش دوم**، لایه‌های ذکر شده را اضافه کنید. سعی کنید با چند بار آزمایش معماری بهینه را به‌دست آورید و دقت خود را افزایش دهید.
- نتایج روی داده‌ی تست را با نتایج قبلی مقایسه کنید. مشاهدات به‌دست آمده را تحلیل و گزارش کنید.

---

[36] Regularization

[37] overfitting

منابع

براش آشنایی با پایتورچ نیز می‌توانید از لینک‌های پایه و مثال مربوط به داکیومنت خود پایتورچ یا سایر آموزش‌های موجود در اینترنت استفاده کنید.

برای آموزش شیوه استفاده از Google Colab می‌توانید از این لینک یا این لینک استفاده نمایید.

در این لینک درباره‌ی Adam Optimizer مطالعه کنید.

در این لینک درباره‌ی روش‌های مختلف Regularization مطالعه کنید.

نکات پایانی

- دقت کنید که هدف پروژه **تحلیل نتایج** و تاثیر عوامل مختلف است؛ بنابراین از ابزارهای تحلیل داده بطور مثال نمودارها استفاده کنید و توضیحات مربوط به هر بخش از پروژه را بطور خلاصه و در عین حال مفید در گزارش خود ذکر کنید.
- نتایج و گزارش خود را در یک فایل فشرده با عنوان AI-CA5-<#SID>.zip تحویل دهید. محتویات پوشه باید شامل فایل jupyter-notebook، خروجی html و فایل‌های مورد نیاز برای اجرای آن باشد. **تحلیل و نمایش خروجی‌های خواسته شده بخشی از نمره این تمرین را تشکیل می‌دهد**. از نمایش درست خروجی‌های مورد نیاز در فایل html مطمئن شوید.
- توجه داشته باشید که علاوه بر ارسال فایل‌های پروژه، این پروژه به صورت حضوری نیز تحویل گرفته خواهد شد. بنابراین تمام بخش‌های پروژه باید قابلیت اجرای مجدد در زمان تحویل حضوری را داشته باشند. همچنین در صورت عدم حضور در تحویل حضوری نمره‌ای دریافت نخواهید کرد.
- در صورتی که سوالی در مورد پروژه داشتید بهتر است در فروم یا گروه درس مطرح کنید تا بقیه از آن استفاده کنند؛ در غیر این صورت با طراحان در ارتباط باشید.
- هدف از تمرین، یادگیری شماست. لطفا تمرین را خودتان انجام دهید.

موفق باشید.