به نام خدا

گزارش تمرین اول

Word2Vec

NLPDL4001

محمد رشیدخان

[m.rashidkhan99@gmail.com](mailto:m.rashidkhan99@gmail.com)

استاد: دکتر قاسمی

پردیس فارابی دانشگاه تهران

بهار 1401

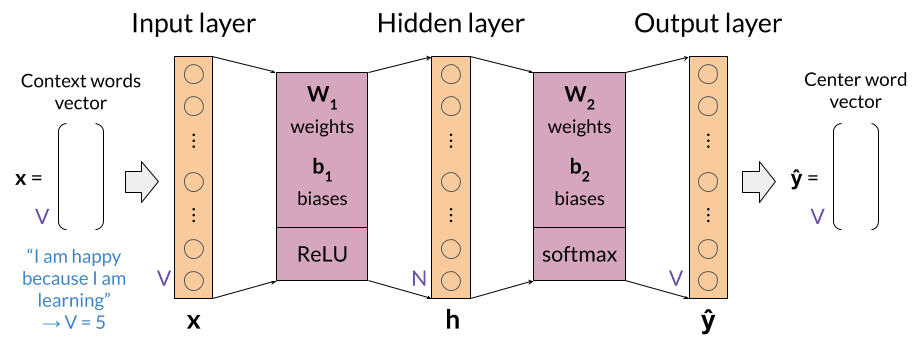
در این تمرین میخواهیم الگوریتم Word2Vec که بردار کلمات را در یک فضای n بعدی به ما میدهد پیاده­سازی کنیم.

طبق مقاله Mikolov et al 2013 ، 2 روش برای این کار پیشنهاد شده است.

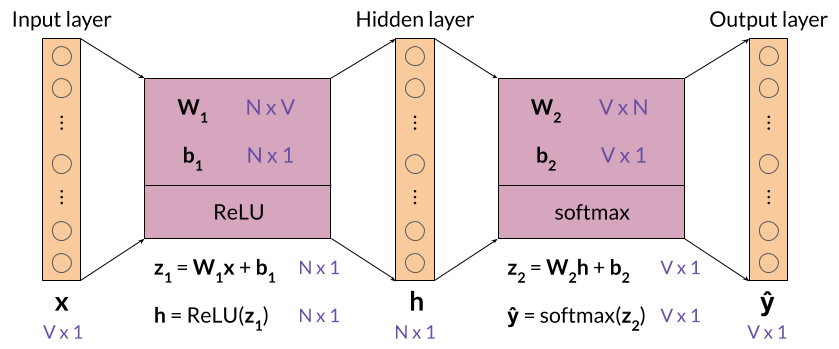
روش اول CBOW (Continuous Bag Of Words) است. روش دوم Skipgram – Negative sampling است که ما با هر 2 روش مدل train کردیم. (البته با کتابخانه­های مختلف مثل Tensorflow – pytorch )

CBOW (Continuous Bag Of Words)

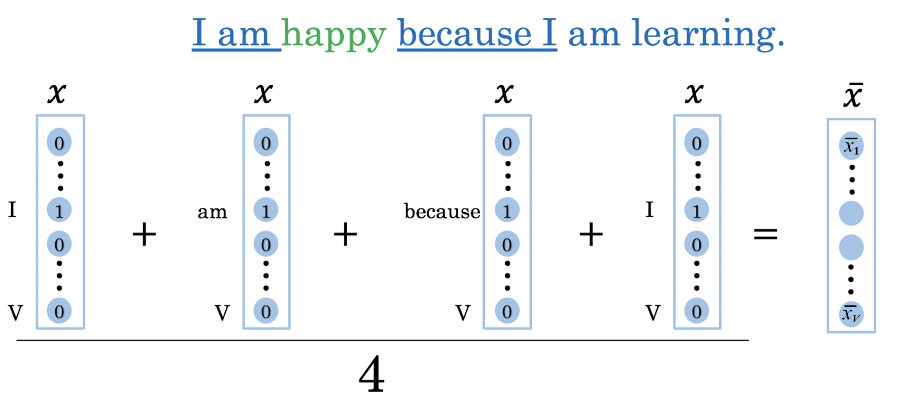
در این روش هدف ما این است که با داشتن تعدادی کلمه در جمله که به انها context words گفته میشود کلمه بینشان را که center word می­گوییم پیش­بینی کنیم. ورودی­ها طبق شکل زیر به شبکه عصبی داده میشوند و خروجی احتمال کلمات است که مدل پیش­بینی کرده است و باید به این نکته توجه داشت که در اینجا خروجی ما word embeddings نیست و در واقع محصول جانبی ما در این شبکه عصبی بردار کلمات میباشد که همان وزن لایه­ها میباشد.

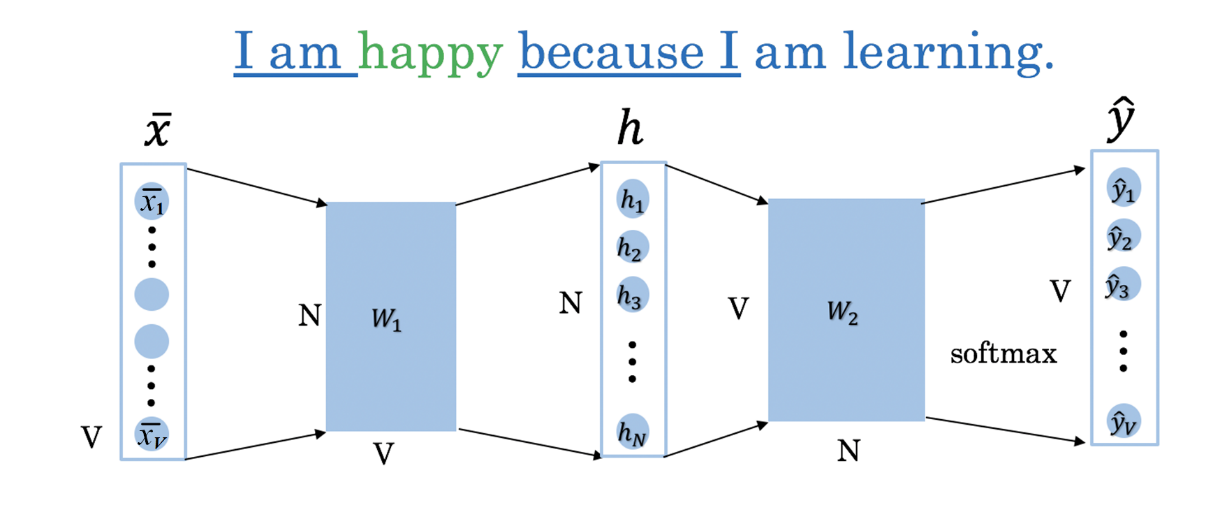


همانطور که در تصویر بالا مشخص است ما سه لایه ورودی، پنهان و خروجی داریم و 2 ماتریس برای وزن لایه­ها داریم که به ترتیب W1 و W2 هستند و همینطور بایاس برای هر لایه و در ورودی برداری از context word ها را داریم که طول بردار اندازه vocab است چون بردارها به صورت one-hot هستند. در لایه پنهان از تابع فعال­سازی Relu استفاده کردیم و در لایه خروجی نیز از تابع softmax برای احتمال­ها استفاده کردیم. خروجی لایه اخر نیز مانند لایه ورودی لایه اول میباشد و هم سایز vocab باید باشد. در شکل زیر ابعاد ورودی و خروجی لایه­ها را به خوبی میتوان مشاهده کرد.



در این معماری ما بردار context words را به صورت میانگین بردارهای تشکیل دهنده آن context در نظر گرفتیم که برای مثال شکل زیر بردار context را برای کلمات {I, am, because, I} نشان میدهد که هدف در اینجا تشخیص کلمه وسط یعنی happy میباشد.





پس ما در این معماری 2 نوع بردار داریم یکی برای center word و یکی هم برای context words که هر 2 به صورت جفتی به عنوان یک نمونه اموزش به مدل داده میشوند.

حال که یک کلیتی از ماجرا را گفتیم به سراغ بررسی معماری اخذ شده میرویم.

به صورت کلی مراحل زیر برای آموزش مدل باید طی شود:

* آماده سازی داده
  + پاکسازی و توکنایز کردن
  + بدست اوردن پنجره­های center word, context word
  + تبدیل کلمات به بردارها
  + ساخت مجموعه آموزش
* ساخت مدل
  + Forward propagation
  + Compute cost
  + Backpropagation
  + Gradient descent
* استخراج word embedding ها
* Visualization and evaluation

**آماده سازی داده**

در مرحله نخست باید دیتایی که به مدل میدهیم را اماده­سازی بکنیم و برای این منظور متن را اول پاکسازی میکنیم و سپس توکنایز میکنیم و بعد پنجره­های کلمات را با توجه به هایپرپارامترها میسازیم. حال این پنجره­ها را تبدیل به بردار میکنیم و بعد به صورت جفت (center word, context words) به عنوان یک نمونه اموزش ذخیره میکنیم. برای ساخت مجموعه آموزش نیز میتوانیم بر حسب مقدار پارامتر batch\_size که مثلا 1024 است batch هایی شامل 1024 نمونه تصادفی از کل جفت­ها انتخاب میکنیم و در هر epoch برای شبکه عصبی یک batch را میفرستیم.

**ساخت مدل**

در مرحله اول از این بخش forward propagation را طراحی میکنیم طبق شبکه مورد نظرمان که shallow هست و در این لایه ماتریس­های موردنظر را بدست میاوریم و خروجی مدل را بعد از اعمال softmax بدست میاوریم. سپس یک تابع هزینه بر اساس cross-entropy پیاده میکنیم که پیش­بینی مدل را بسنجیم و مشتق تابع هزینه را نسبت به پارامترهای شبکه عصبی حساب میکنیم و به وسیله backpropagation مقادیر پارامترها را به روش گرادیان کاهشی آپدیت میکنیم.

**استخراج word embedding ها**

کلا به 3 روش ما میتوانیم بردار کلمات را استخراج کنیم. در روش اول میتوانیم ماتریس W1 را به عنوان بردار کلمات در نظر بگیریم. در روش دوم به طور مشابه میتوانیم وزن لایه 2 را در نظر بگیریم یعنی W2 و در روش سوم میتوانیم میانگین این دو ماتریس را به عنوان بردار کلمات خود در نظر بگیریم.

**شبیه­سازی و ارزیابی**

در این بخش به ارزیابی و نمایش بردار کلمات در فضای 3 یا 2 بعدی میپردازیم که از الگوریتم PCA کمک میگیریم تا این بردارها را در 2 یا 3 بعد نمایش بدیم. برای نمایش 3 بعدی بردار کلمات از Tensorboard استفاده میکنیم همچنین از کتابخانه whatlies نیز برای شبیه­سازی استفاده میکنیم. برای نزدیکترین بردارهای همسایه نیز این قابلیت در Embedder projector وجود دارد با این حال تابع را از اول خودمان بر حسب فاصله اقلیدسی یا کسینوسی پیاده­سازی میکنیم.