دانشکده‌ی مهندسی کامپیوتر

یادگیری عمیق

نیم‌سال نخست ۱۴۰۱ـ۱۴۰۰

مدرس: حمید بیگی

پاسخنامه تمرین سری چهارم — یادگیری بازنمایی و مدل‌های توجه — زمان تحویل: ۱۳ دی‌ماه

مسئله‌ی ۱. Neural network Language Models(NNLM)

(آ) دو تفاوت اصلی و عمده بین NNLM و CBOW که ر درس آموخته‌اید را بیان کنید و برای هرکدام یک مثال (به زبان فارسی یا انگلیسی) بزنید.

۱ـ در روش CBOW با در نظر گرفتن پنجره‌ای از کلمات زمینه اطراف کلمه میانی(هدف)، سعی می‌شود که کلمه هدف پیش‌بینی شود، یعنی به کلمات هردو سمت کلمه میانی توجه می‌شود. اما در روش NNLM به کلمات زمینه سمت چپ(کلمات بعدی در زبان فارسی) کلمه هدف توجهی نمی‌کند(در زبان انگلیسی برعکس است). برای مثال در روش NNLM با توجه به دو جلمه زیر، بین کلمه تابستان و زمستان هیچ تفاوتی در نظر گرفته نمی‌شود:

می‌دانیم که آب و هوا در (زمستان) اغلب سرد و بارانی است.

می‌دانیم که آب و هوا در (تابستان) اغلب گرم و آفتابی است.

۲ـ در روش CBOW از جمع بردارهای کلمات زمینه استفاده می‌شود اما در NNLM بردارهای کلمات زمینه به صورت غیرخطی ترکیب می‌شوند. بنابراین برای مثال در روش NNLM بین کلمات زیر به عنوان کلمات زمینه تفاوت قائل می‌شویم اما در روش CBOW اینطور نیست: good to not ،not good to

(ب) پیچیدگی محاسباتی را در بدست آوردن خروجی شبکه NNLM با روش انتشار رو به جلو [1] برای یک عدد داده آموزشی محاسبه کنید. توجه کنید که برای گرفتن نمره کامل این بخش باید مرحله به مرحله پیچیدگی محاسباتی را حساب کنید و سپس تجمیع آن‌ها را بدست آورید.

۱ـ در ادغام بردار کلمات پیچیدگی محاسباتی برابر با $N \times D$ است.

۲ـ در محاسبه مقدار h پیچیدگی محاسباتی برابر با $N \times D \times H$ است.

۳ـ در محاسبه مقدار $\hat{y}$ از روی h پیچیدگی محاسباتی برابر با $H \times V$ است.

← در مجموع پیچیدگی برابر با $O(NDH + HV)$ است که چون $V \gg ND$ لذا پیچیدگی نهایی برابر است با $O(HV)$.

(ج) اگر بتوانیم تابع هزینه را تغییر دهیم، یک روش ارائه کنید تا بتوانیم پیچیدگی محاسباتی بخش قبل را کاهش دهیم. نام بردن روش به صورت مختصر کافی است و نیازی به توضیح نیست.

استفاده از Sampling Negative برای محاسبه Softmax و یا استفاده از Softmax Hierarchical.

(د) گرادیان J نسبت به x را بدست آورید.

اگر $z_1 = xW + b$ و $z_2 = hU + d$ باشد، داریم:

[1] Forward Propagation

$$\delta_1 = \frac{\partial CE}{\partial \hat{y}} = (\hat{y} - y)$$
$$\delta_2 = \frac{\partial CE}{\partial h} = \delta_1 \frac{\partial z_2}{\partial h} = U^T \delta_1$$
$$\delta_3 = \frac{\partial CE}{\partial z_1} = \delta_2 \frac{\partial h}{\partial z_1} = \delta_2 (1 - \tanh(z_1)^2)$$
$$\rightarrow \frac{\partial CE}{\partial x} = \delta_3 \frac{\partial z_1}{\partial x} = W^T \delta_3$$

مسئله‌ی ۲. Word2Vec and Glove

(آ) یکی دیگر از روش‌های یادگیری بردار کلمات، روش‌های وقوع همزمان مبتنی بر شمارش [۲] است. دو مزیت و دو عیب در استفاده از روش Word2Vec نسبت به این روش را بیان کنید.

مزایا: مقیاس‌پذیری با اندازه پایگاه‌داده متون (corpus) ـ توانایی بدست آوردن الگوهای پیچیده فراتر از شباهت کلمات به یکدیگر

معایب: سرعت کمتر ـ روش های مبتنی بر شمارش، استفاده بهینه‌تری از آمار می‌کنند و از لحاظ پیچیدگی محاسباتی بهینه ترند

(ب) دو نفر می‌خواهند از روش Word2Vec برای بدست آوردن تعبیه کلمات در یک Vocabulary یکسان با سایز V استفاده کنند. به طور دقیق‌تر نفر اول بردار کلمات زمینه u_w^A و بردار کلمه هدف v_w^A را برای هر $w \in V$ و نفر دوم بردار کلمات زمینه u_w^B و بردار کلمه هدف v_w^B را برای هر $w \in V$ می‌خواهند به وسیله این روش، بدست آورند. فرض کنید برای هر جفت کلمه $w, w' \in V$ ضرب داخلی بردار کلمات در مدل هر دو نفر یکسان باشد یعنی

$$(u_w^A)^T v_{w'}{}^A = (u_w^B)^T v_{w'}{}^B$$

آیا می‌توان نتیجه گرفت که برای هر کلمه $w \in V$ داریم $v_w^A = v_w^B$ ؟ چرا؟

خیر زیرا مدل Word2Vec برای کلماتی که زمینه یکسانی دارند، فقط برای ضرب داخلی بردارهای کلمات آن‌ها بهینه می‌شود. یعنی ممکن است شخصی همه بردارهای کلمات را به مقدار یکسانی دوران دهد و همچنان ضرب داخلی ثابت باقی بماند، و یا شخصی دیگر بردارهای کلمات زمینه را در عدد ثابت k و بردارهای کلمات هدف را در عدد ثابت $\frac{1}{k}$ ضرب کند، در حالی که بردارها متفاوت شده‌اند، ضرب داخلی همچنان ثابت است. پس یکسان بودن ضرب داخلی نشانگر یکسان بودن بردارها به صورت جدا نیست.

(ج) چرا استفاده از تابع Softmax برای Word2Vec کار درستی نیست؟ راهکار پیشنهادی شما برای حل این مشکل چیست؟

اگر از Softmax استفاده کنیم، مدل بسیار سخت آموزش داده می‌شود زیرا تعداد کتگوری‌ها (سایز Vocabulary) زیاد است. برای حل این مشکل همانطور که در مسئله ۱ گفته شد، از Negative Sampling و Hierechical Softmax می‌توانیم استفاده کنیم.

(د) مقدار حافظه مصرفی الگوریتم های Word2Vec و Glove را با ذکر دلیل مقایسه کنید.

چون که Glove بر روی ماتریس وقوع همزمان کلمات آموزش داده می‌شود، حافظه بسیار زیادی را نیاز دارد که در روش Word2Vec اینطور نیست.

[2] Cooccurence Count-based Methods

(ه) دو مورد از نواقص مشترک Word2Vec و Glove را نام ببرید.

۱ـ مشکل در یادگیری تعبیه کلمات خارج از دیکشنری

۲ـ مشکل در متفاوت دانستن جفت کلمات متضاد(از لحاظ معنی). برای مثال کلمه خوب و بد اغلب در فضای برداری در نزدیک یکدیگر قرار می‌گیرند که کارایی این روش‌هارا کاهش می‌دهد به‌خصوص در کاربردهایی نظیر تشخیص احساسات!

(و) در تابع هزینه زیر که متعلق به الگوریتم Glove است، اگر ماتریس های R و $\tilde{R}$ تعبیه‌های کلمات $\{r_i\}, \{\tilde{r}_j\}$ را در خود داشته باشند، نشان دهید که این تابع هزینه محدب [۳] نیست. برای حل این سوال، نیازی به اثبات ریاضی نیست و فقط کافی است توضیح منطقی‌ای ارائه کنید که محدب نبودن را تصدیق کند.

$$J(R, \tilde{R}) = \sum_{i,j} f(x_{ij})(r_i^T \tilde{r}_j - \log x_{ij})^2$$

راهنمایی: از مفاهیم Swap-inveriance و Permutation-invariance استفاده کنید.

این تابع هزینه محدب نیست زیرا:

راه حل اول: وقتی ابعاد بردارهای تعبیه داخل R و $\tilde{R}$ را به صورت یکسان و همزمان تغییر [۴] بدهیم، تابع هزینه ثابت می‌ماند. برای مثال اگر از بردارهای تعبیه تغییریافته میانگین بگیریم، تمام مقادیر در تمام ابعاد بردارهای تعبیه نتیجه، یکسان خواهد شد که قطعا هزینه بیشتری نسبت به حالت تغییر نیافته دارد.

راه حل دوم: اگر به صورت مستقیم R و $\tilde{R}$ را تغییر دهیم نیز تابع هزینه ثابت می‌ماند. برای مثال اگر میانگین هر دو را حساب کنیم یعنی $\frac{R+\tilde{R}}{2}$ ، برای هر دو ماتریس بردار تعبیه کلمات یکسانی خواهیم داشت که هزینه بیشتری نسبت به تابع هزینه اصلی دارد. دلیل این امر این است که بیشترین وقوع [۵] همیشه ضرب داخلی کلمات با خودشان است.

مسئله‌ی ۳. Transformers and Attention models

(آ) یکی از مراحل مقدماتی در پیش پردازش داده‌ها برای حوزه پردازش متن، توکن‌سازی [۶] از جمله‌های پایگاه داده است. یکی از چالش‌های این مرحله، این است که ترتیب کلمات در جمله با این‌کار از بین می‌رود. برای حل این مشکل رویکرد Positional Embedding مطرح می‌شود. به صورت کلی و کوتاه توضیح دهید که چگونه این رویکرد می‌تواند مشکل را برطرف کند.

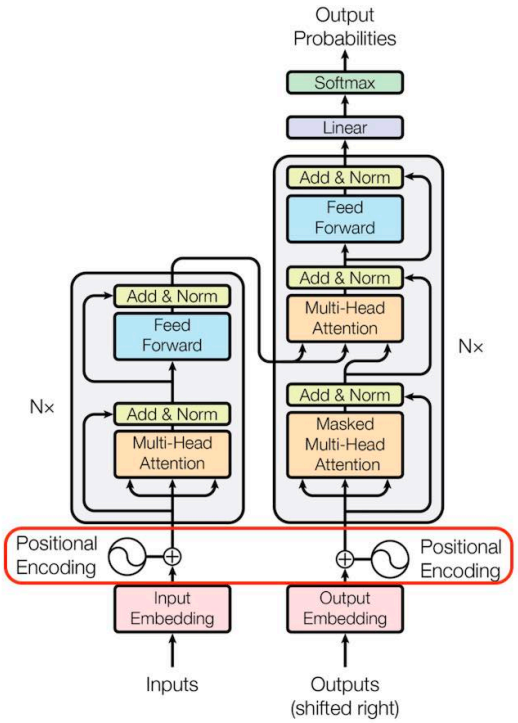
به صورت کلی در رویکرد Positional Embedding بردار تعبیه یک کلمه برابر است با تجمیع بردار کلمه ورودی و جایگاه آن در جمله به صورت یک مقدار. پس با استفاده از این رویکرد می‌توانیم مکان کلمه را در جمله حفظ کرده و از فراموشی آن جلوگیری کنیم. لذا یک کلمه در جایگاه مختلف در جمله قطعا بردار تعبیه متفاوتی خواهد داشت.

(ب) یکی از کاربردهای رویکرد گفته شده در قسمت قبل، استفاده از Transformer ها است. این معماری چگونه مشکل گفته شده(بهم خوردن ترتیب کلمات) را برطرف می‌کند؟

در Transformer ها تعبیه کلمه ورودی قبل از وارد شدن به Encoder با یک ماتریس که بیانگر مکان آن کلمه در جمله(یا دنباله ای از کلمات) است تجمیع شده و با این کار مکان کلمات حفظ می‌شود. در تصویر زیر می‌توانید معماری گفته شده را مشاهده کنید.

---

[3] Convex
[4] Permute
[5] Occurance
[6] Tokenization



(ج) محدودیت معماری Encoder-Decoder ای که از مکانیزم توجه استفاده نمی‌کند را در حوزه ترجمه ماشینی بیان کنید و سپس به صورت خلاصه توضیح دهید که چگونه مکانیزم توجه می‌تواند این مشکل را برطرف کند.

در معماری گفته شده، تمام مفهوم و معنی جمله ورودی(بردار مخفی) به شبکه باید در یک بردار ذخیره شود، لذا هرچه اندازه جمله ورودی بزرگ‌تر شود، حفظ اطلاعات سخت‌تر می‌شود و در نهایت ممکن است اطلاعاتی از دست رود. به این اتفاق، فراموشی می‌گویند. بنابراین بردار مخفی گفته شده یک گلوگاه محسوب می‌شود و در افزایش اندازه جمله ورودی محدودیت وجود دارد.

با استفاده از مکانیزم توجه می‌توانیم قسمت‌های مهم جمله ورودی را به شبکه بفهمانیم(با محاسبه امتیازی برای هر قسمت) تا به آن قسمت‌ها توجه بیشتری کند و محدودیت بزرگی جمله ورودی را برطرف کنیم.
از طرف دیگر به جای بردار مخفی می‌توانیم از ماتریس استفاده کنیم و محدودیت بزرگ شدن بردار را نیز برطرف کنیم.