توصیف پارامترهای یادگیری Word2vec

چکیده

مدل یا application با نام word2vec توسط Mikolov و همکارانش ارائه شده است. این مدل در سال‌های اخیر توجه زیادی را به خود جلب کرده است. نمایش برداری کلمات توسط مدل یادگیری word2vec برای ارائه معنای مفاهیم امکان‌پذیر است که در انواع وظایف از نوع NLP کاربرد دارد. بنا به افزایش تعداد تحقیقات و آزمایشات موجود در این زمینه، نبود یک توصیف فراگیر و جامع برای پارامترهای فرآیند یادگیری مدل‌های نهفته کلمات به‌صورت جزئی، توجه مرا به خود جلب کرد. بنابراین به‌منظور جلوگیری از انجام تحقیقات غیرتخصصی در شبکه‌های عصبی به‌دلیل عدم فهم مکانیزم کارایی چنین مدل‌هایی، این مقاله ارائه شده است.

این مقاله، جزئیات و توصیف پارامترهای مدل‌های word2vec را نشان می‌دهد، که شامل مدل‌های پیوسته bag-of-word (CBOW) و مدل‌های skip-gram (SG)، همانند تکنیک‌های بهینه‌سازی پیشرفته، از جمله softmax سلسله مراتبی و نمونه منفی است. تفسیر معادلات شیب نیز در کنار مشتقات ریاضی ارائه شده است.

در پیوست، یک بررسی بر روی اصول اولیه شبکه عصبی و انتشار به عقب ارائه شده است. من هم یک نسخه‌ی نمایشی تعاملی، به‌منظور تسهیل درک بصری از مدل ارائه کرده‌ام.

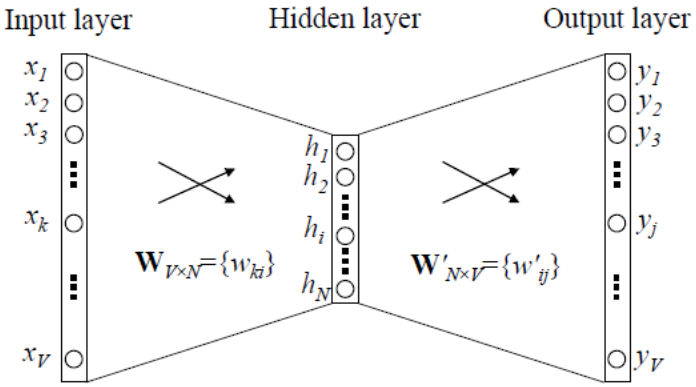
1. مدل Bag-of-Word متوالی

1.1. متن One-word

ما از ساده‌ترین نسخه از مدل Bag-of-Word متوالی (CBOW) که توسط Mikolov و همکارانش در سال2013 معرفی شده است شروع می‌کنیم. فرض می‌کنیم که تنها یک کلمه از هر متن در نظر گرفته شده است، بدین معناست

که مدل یک کلمه‌ی هدف داده شده از یک متن را پیش‌بینی خواهد کرد، که مثل یک مدل بیگرام است. برای خوانندگانی که به شبکه‌های عصبی ناآشنا هستند، توصیه می‌شود که برای یک بررسی سریع از مفاهیم و اصطلاحات مهم قبل از هر اقدامی به ضمیمه A مراجعه کنند.

شکل 1 مدل شبکه تحت تعریف ساده‌ی متن را نشان می‌دهد. در تنظیمات ما، اندازه واژگان V است و اندازه لایه پنهان N است. واحدهای مجاور لایه‌ها به‌طور کامل متصل هستند.



شکل 1: مدل CBOW ساده با تنها یک کلمه در متن

ورودی یک بردار کدگذاری one-hot است، که به معنی یک ورودی کلمه‌ی متن است، تنها یک خروجی از واحدهای V یک خواهد بود و بقیه واحدها صفر هستند.

وزن بین لایه‌ی ورودی و لایه‌ی خروجی می‌تواند توسط ماتریس $V \times N$، W، نمایش داده شود. هر سطر از ماتریس W یک بردار N بعدی با نمایش v_w، کلمه مرتبط با لایه ورودی است. با توجه به متن (یک کلمه)، با فرض $x_k = 1$ و $x_{k'} = 0$ برای $k' \neq k$ پس

$$\mathbf{h} = \mathbf{x}^T\mathbf{W} = \mathbf{W}_{(k,\cdot)} := \mathbf{v}_{w_I}, \qquad (1)$$

که اساسا kامین سطر از W را به h کپی می‌کند. v_{wI} نمایش بردار ورودی w_I است. این بدین معنی است که تابع لینک (فعال‌ساز) از واحدهای لایه پنهان خطی ساده است (به‌عنوان مثال، عبور مستقیم مجموع وزن آن از ورودی‌ها به لایه‌ی بعدی).

از لایه‌های پنهان به لایه‌ی خروجی است، یک ماتریس وزن مختلف $W' = \{w_{ij}\}$ وجود دارد که یک ماتریس $N \times V$ است. با استفاده از این وزن‌ها می‌توانیم مقدار u_j را برای هر کلمه از واژگان محاسبه کنیم،

$$u_j = {v'_{w_j}}^T \cdot h, \qquad (2)$$

که v'_{wj}، jامین سطر از ماتریس W' است. بنابراین ما از softmax که یک مدل طبقه‌بندی است، برای به دست آوردن توزیع خلفی کلمات، که یک توزیع چند جمله‌ای است استفاده می‌کنیم.

$$p(w_j|w_I) = y_j = \frac{\exp(u_j)}{\sum_{j'=1}^{V} \exp(u_{j'})}, \qquad (3)$$

که در آن y_j خروجی jامین واحد در لایه‌ی خروجی است. با جایگزینی (1) و (2) در (3)، به‌دست می‌آوریم،

$$p(w_j|w_I) = \frac{\exp\left({v'_{w_O}}^T v_{w_I}\right)}{\sum_{j'=1}^{V} \exp\left({v'_{w'_j}}^T v_{w_I}\right)} \qquad (4)$$

توجه داشته باشید که v_m و v'_m دو نمایش متفاوت از کلمه w هستند. v_m از سطرهای W حاصل شده است که ماتریس وزن ورودی⟵پنهان است و v'_m از ستون‌های W' حاصل شده است، که ماتریس پنهان⟵خروجی است. در تجزیه‌وتحلیل، v_m به‌عنوان "بردار ورودی " و v'_m به‌عنوان "بردار خروجی" از کلمه‌ی W است.

به‌روزرسانی معادله برای وزن‌های پنهان⟵ خروجی

اکنون اجازه دهید تا از معادله‌ی به‌روز شده‌ی وزن برای این مدل مشتق بگیریم. اگرچه رقابت واقعی غیرعملی است (در زیر توضیح داده شده است)، اما درحال انجام مشتق برای به‌دست آوردن بینش در مدل اصلی بدون هیچ حقه‌ای هستیم. برای بررسی اصول اولیه پس از انتشار، ضمیمه A را مطاالعه کنید.

هدف آموزش (برای یک نمونه آموزش) به حداکثر رساندن (4)، احتمال شرطی مشاهده کلمه خروجی واقعی w_O (دلالت بر شاخص آن در لایه خروجی به عنوان j^*) با توجه به وزن و ورودی متن کلمه w_I است.

$$\max p(w_O|w_I) = \max y_{j^*} \quad (5)$$
$$= \max \log y_{j^*} \quad (6)$$
$$= u_{j^*} - \log \sum_{j'=1}^{V} \exp(u_{j'}) := -E, \quad (7)$$

که در آن $E = -\log p(w_O|w_I)$ تابع خسارت ما (ما می‌خواهیم E را حداقل کنیم) و j^* اندیس کلمه خروجی واقعی در لایه‌ی خروجی است. توجه داشته باشید که این تابع خسارت می‌تواند به‌عنوان یک مورد خاص از اندازه‌گیری متقابل آنتروپی بین دو توزیع احتمالاتی باشد.

اکنون اجازه دهید تا از معادله‌ی به‌روزشده از وزن بین لایه‌های پنهان و خروجی مشتق بگیریم. با توجه به مشتق E با توجه به jامین واحد ورودی u_j، معادله زیر را بدست می‌آوریم،

$$\frac{\partial E}{\partial u_j} = y_j - t_j := e_j \quad (8)$$

که در آن $t_j = 1(j = j^*)$، به‌عنوان مثال، t_j فقط زمانی که jامین واحد، کلمه خروجی واقعی است، 1 می‌شود، در غیراینصور $j\ t_j = 0$ است. توجه داشته باشید که این مشتق به‌سادگی خطای پیش‌بینی e_j از لایه‌ی خروجی است.

سپس نسبت به w_{ij} برای به‌دست آوردن شیب در وزن‌های پنهان⟵خروجی مشتق می‌گیریم.

$$\frac{\partial E}{\partial w'_{ij}} = \frac{\partial E}{\partial u_j} \cdot \frac{u_j}{\partial w'_{ij}} = e_j \cdot h_i \quad (9)$$

بنابراین، با استفاده از گرادیان نزولی تصادفی، معادله‌ی به‌روز شده‌ی وزن برای وزن‌های پنهان⟵خروجی را به‌دست می‌آوریم:

$$w'^{(\text{new})}_{ij} = w'^{(\text{old})}_{ij} - \eta \cdot e_j \cdot h_i. \quad (10)$$

یا

$$\mathbf{v}'_{w_j}{}^{(\text{new})} = \mathbf{v}'_{w_j}{}^{(\text{old})} - \eta \cdot e_j \cdot \mathbf{h} \qquad \text{for } j = 1, 2, \cdots, V. \tag{11}$$

که در آن $\eta > 0$ میزان یادگیری، $e_j = y_j - t_j$ و h_i، iامین واحد در لایه مخفی است. v_{wj} بردار خروجی از w_j است. توجه داشته باشید که معادله‌ی به‌روز شده نتیجه می‌دهد که ما مجبور به رفتن از طریق هر کلمه‌ی ممکن در واژگان، بررسی احتمال خروجی آن y_j، و مقایسه‌ی y_j با خروجی مورد انتظار آن t_j (0 یا 1) هستیم. اگر $y_j > t_j$، بنابراین نسبت بردار پنهان h (به‌عنوان مثال V_{wI}) را از V_{wO} کم می‌کنیم، بنابراین V_{wO} را دورتر از V_{wI} می‌سازیم؛ اگر $y_j < t_j$، مقدار h را به V_{wO} اضافه می‌کنیم، بنابراین V_{wO} را نزدیک به V_{wI} می‌سازیم. اگر y_j بسیار نزدیک به t_j باشد، با توجه به معادله‌ی به‌روز شده، تغییر بسیار کمی به وزن‌ها اعمال خواهد شد. دوباره توجه داشته باشید، که v_w (بردار ورودی) و $v_{w'}$ (بردار خروجی) دو نمایش مختلف بردار از کلمه w هستند.

به‌روزرسانی معادله برای وزن‌های ورودی←پنهان

پس از بدست آوردن معادلات به‌روزشده برای W'، ما هم اکنون می‌توانیم به W حرکت کنیم. بنابراین از E بر روی خروجی لایه‌های پنهان مشتق می‌گیریم و رابطه‌ی زیر را به دست می‌آوریم

$$\frac{\partial E}{\partial h_i} = \sum_{j=1}^{V} \frac{\partial E}{\partial u_j} \cdot \frac{\partial u_j}{\partial h_i} = \sum_{j=1}^{V} e_j \cdot w'_{ij} := \text{EH}_i \tag{12}$$

که در آن h_i خروجی iامین واحد از لایه‌های پنهان است. u_j در (2) تعریف شده است، ورودی شبکه از jامین واحد در لایه خروجی؛ و $e_j = y_j - t_j$ خطای پیش‌بینی jامین کلمه در لایه خروجی است. EH، یک بردار N بعدی، مجموع بردار خروجی همه‌ی کلمات در واژگان است که، توسط خطای پیش‌بینی وزن‌دار شده‌اند.

سپس ما باید مشتق E را نسبت به W بگیریم. ابتدا به یاد می‌آوریم که لایه‌های پنهان محاسبات خطی بر روی مقادیر لایه ورودی انجام می‌دهند. با بسط نماد بردار در (1) داریم

$$h_i = \sum_{k=1}^{V} x_k \cdot w_{ki} \tag{13}$$

حالا می‌توانیم مشتق E را نسبت به W به‌دست بیاوریم

$$\frac{\partial E}{\partial w_{ki}} = \frac{\partial E}{\partial h_i} \cdot \frac{\partial h_i}{\partial w_{ki}} = \mathrm{EH}_i \cdot x_k \qquad (14)$$

$$\frac{\partial E}{\partial \mathbf{W}} = \mathbf{x} \cdot \mathrm{EH} \qquad (15)$$

که از آن یک ماتریس $V \times N$ به دست می‌آوریم. از آنجا که تنها یک جزء از x غیر صفر است، تنها یک ردیف از $\frac{\partial E}{\partial W}$ غیرصفر است، و مقدار سطر آن EH، یک بردار Nبعدی، است. معادله‌ی به‌روز شده از W را به‌صورت رابطه‌ی زیر به‌دست می‌آوریم

$$\mathbf{v}_{w_I}^{(\mathrm{new})} = \mathbf{v}_{w_I}^{(\mathrm{old})} - \eta \cdot \mathrm{EH} \qquad (16)$$

که در آن v_{wI} یک ردیف از W، "بردار ورودی" از تنها کلمه متن و تنها سطر از W که مشتق آن غیرصفر است. تمام سطرهای دیگر W بعد از تکرار به‌دلیل این‌که مشتقات آنها صفر هستند بدون تغییر باقی خواهند ماند.

به‌طورمستقیم، بردار EH، مجموع بردار خروجی از تمام کلمات در واژگان وزن‌دار توسط خطای پیش‌بینی $e_j = y_j$ t_j است، ما می‌توانیم (16) را مانند اضافه کردن بخشی از هر بردار خروجی در واژگان به بردار ورودی از کلمه‌ی متن درک کنیم. اگر در لایه خروجی، احتمال این‌که یک کلمه‌ی w_j، کلمه خروجی دست بالا باشد $y_j > t_j$، سپس بردار ورودی از کلمه متن w_I تمایل به حرکت دورتر از بردار خروجی w_j دارد؛ در مقابل، اگر احتمال این‌که w_j خروجی کلمه در نظر گرفته شده باشد $y_j < t_j$، بردار ورودی w_I تمایل به حرکت نزدیک به بردار خروجی w_j دارد؛ اگر احتمال w_j نسبتا دقیق پیش‌بینی شده باشد، پس از آن تاثیر کمی در حرکت از بردار ورودی w_I دارد. حرکت بردار ورودی از w_I توسط خطای پیش‌بینی همه‌ی بردارها در واژگان تعیین می‌شود؛ خطای پیش‌بینی بزرگتر، توجه بیشتر یک کلمه بر روی حرکت در بردار ورودی از کلمه‌ی متن را اعمال می‌کند.

همانطور که ما مکررا پارامترهای مدل را با حرکت در طول جفت کلمه‌ی متن هدف تولید شده از یک مجموعه آموزشی به‌روز می‌کنیم، اثرات بر روی بردار تجمع می‌یابد. ما می‌توانیم تصور کنیم که بردار خروجی یک کلمه w، با بردار ورودی همسایگان عقب و جلو کشیده می‌شود، به‌عنوان مثال اگر رشته فیزیکی بین بردار w و بردار همسایگان آن

وجود داشته باشد. به‌طورمشابه، یک مسیر ورودی می‌تواند توسط کشیدن بسیاری از بردارهای خروجی در نظر گرفته شود. این تفسیر می‌تواند گرانش و یا طرح نمودار نیرو را برای ما یادآوری کند. طول تعادل هر یک از رشته‌های خیالی مربوط به قدرت بین جفت کلمات در ارتباط به‌عنوان نرخ یادگیری بیان می‌شود. پس از تکرار زیاد، موقعیت نسبی بردار ورودی و خروجی در نهایت ثبات خواهد یافت.

1.2 متن چند کلمه‌ای

شکل 2 مدل CBOW را با یک تنظیم متن چندکلمه‌ای نشان می‌دهد. در هنگام محاسبه‌ی خروجی لایه پنهان، به جای کپی کردن مستقیم بردار ورودی از متن ورودی کلمه، مدل CBOW به‌طور متوسط بردار کلمات متن ورودی را گرفته و از محصول ماتریس وزن‌دار ورودی←پنهان و بردار متوسط به‌عنوان خروجی استفاده می‌کند.

$$\mathbf{h} = \frac{1}{C}\mathbf{W}\cdot(\mathbf{x}_1+\mathbf{x}_2+\cdots+\mathbf{x}_C) \quad (17)$$

$$= \frac{1}{C}\cdot(\mathbf{v}_{w_1}+\mathbf{v}_{w_2}+\cdots+\mathbf{v}_{w_C}) \quad (18)$$

که در آن C تعداد کلمات موجود در متن، w_Cو ... و w_1 کلمات موجود در متن و v_w بردار ورودی از کلمه‌ی w است.

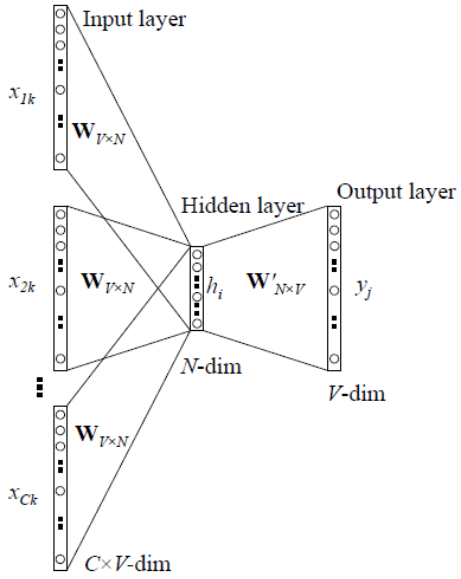
تابع خسارت به‌صورت رابطه‌ی زیر بیان می‌شود

$$E = = -\log p(w_O|w_{I,1},\cdots,w_{I,C}) \quad (19)$$

$$= -u_{j^*}+\log\sum_{j'=1}^{V}\exp(u_{j'}) \quad (20)$$

$$= -\mathbf{v}'_{w_O}{}^T\cdot\mathbf{h}+\log\sum_{j'=1}^{V}\exp(\mathbf{v}'_{w_j}{}^T\cdot\mathbf{h}) \quad (21)$$

که همانند (7) است، هدف از مدل یک کلمه مستقل از متن، جز این است که h همانگونه که در (18) به جای (1) تعریف شده است، متفاوت است.



شکل 2: مدل Bag-of-word متوالی

معادله‌ی به‌روزشده برای وزن‌های پنهان←خروجی همانند مدل یک کلمه از متن باقی می‌ماند (11). ما آن را در اینجا کپی می‌کنیم:

$$\mathbf{v}'^{(\text{new})}_{w_j} = \mathbf{v}'^{(\text{old})}_{w_j} - \eta \cdot e_j \cdot \mathbf{h} \qquad \text{for } j = 1, 2, \cdots, V. \tag{22}$$

توجه داشته باشید که ما نیاز داریم این را به هر عنصر از ماتریس وزن پنهان←خروجی برای هر نمونه آموزش به‌کار ببریم.

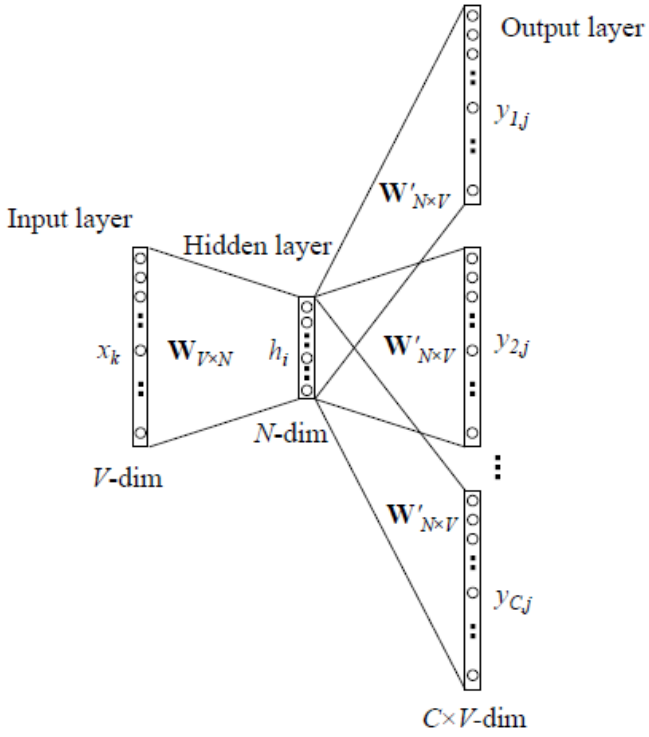
معادله‌ی به‌روزشده برای وزن ورودی←پنهان مشابه (16) است، جز این‌که درحال حاضر ما نیاز به استفاده از معادله زیر برای هر کلمه‌ی $w_{I.c}$ در متن داریم:

$$\mathbf{v}^{(\text{new})}_{w_{I,c}} = \mathbf{v}^{(\text{old})}_{w_{I,c}} - \frac{1}{C} \cdot \eta \cdot \text{EH} \qquad \text{for } c = 1, 2, \cdots, C. \tag{23}$$

که در آن $v_{w_{I.c}}$؛ بردار ورودی cامین کلمه در متن ورودی است. η یک نرخ یادگیری مثبت است و $EH = \frac{\partial E}{\partial h_i}$ توسط (12) داده شده است. درک بصری از این معادله‌ی به‌روزشده همان است که در (16) بیان شده است.

2. مدل Skip-Gram

مدل Skip-Gram توسط Mikolov و همکارانش معرفی شده است(2013). شکل 3 مدل Skip-Gram را نشان می‌دهد. این مدل در مقابل مدل CBOW قرار دارد. کلمه‌ی مورد نظر درحال حاضر در لایه‌ی ورودی و کلمات متن در لایه‌ی خروجی قرار دارند.



شکل 3: مدل Skip-Gram

ما هنوز هم از v_{w_I} برای نشان دادن بردار ورودی از تنها کلمه‌ی بر روی لایه ورودی استفاده می‌کنیم، بنابراین تعریف یکسانی از خروجی‌های h لایه پنهان همانند (1) داریم، که به معنی این است که h به‌سادگی یک سطر از ماتریس وزن ورودی←پنهان، W، همراه با ورودی کلمه w_I را کپی می‌کند. تعریف h را به‌صورت زیر کپی می‌کنیم:

$$\mathbf{h} = \mathbf{W}_{(k,\cdot)} := \mathbf{v}_{w_I}, \qquad (24)$$

در لایه‌ی خروجی، به‌جای خارج‌سازی یک توزیع چندجمله‌ای، توزیع چندجمله‌ای c را خارج می‌کنیم. هر یک از خروجی‌های محاسبه شده از ماتریس پنهان←خروجی یکسانی استفاده می‌کند:

$$p(w_{c,j} = w_{O,c}|w_I) = y_{c,j} = \frac{\exp(u_{c,j})}{\sum_{j'=1}^{V} \exp(u_{j'})} \quad (25)$$

که در آن $w_{c.j}$، jامین کلمه در پانل cام از لایه خروجی است. $w_{O.c}$، cامین کلمه‌ی واقعی در کلمات خروجی متن است. w_I تنها کلمه ورودی است. $y_{c.j}$؛ خروجی jامین واحد در پانل cام از لایه خروجی است؛ $u_{c.j}$ ورودی خالص jامین واحد در پانل cام لایه خروجی است. از آنجا که پانل لایه خروجی وزن‌های یکسانی را به اشتراک می‌گذارد، در نتیجه

$$u_{c,j} = u_j = {\mathbf{v}'_{w_j}}^T \cdot \mathbf{h}, \text{ for } c = 1, 2, \cdots, C \quad (26)$$

که در آن v'_{wj} بردار خروجی jامین کلمه در واژگان است، w_j، و همچنین v'_{wj} از یک ستون ماتریس وزن پنهان←خروجی، W' گرفته شده است.

مشتق از پارامتر به‌روز شده‌ی معادلات، متفاوت از مدل متن یک کلمه‌ای نیست. تابع خسارت به‌صورت زیر تغییر می‌یابد:

$$E = -\log p(w_{O,1}, w_{O,2}, \cdots, w_{O,C}|w_I) \quad (27)$$

$$= -\log \prod_{c=1}^{C} \frac{\exp(u_{c,j_c^*})}{\sum_{j'=1}^{V} \exp(u_{j'})} \quad (28)$$

$$= -\sum_{c=1}^{C} u_{j_c^*} + C \cdot \log \sum_{j'=1}^{V} \exp(u_{j'}) \quad (29)$$

که در آن j_c^* شاخص c امین کلمه خروجی واقعی در واژگان است.

ما از E با توجه به ورودی خالص هر واحد در هر پانل از لایه خروجی، $u_{c.j}$ مشتق می‌گیریم و به‌دست می‌آوریم

$$\frac{\partial E}{\partial u_{c,j}} = y_{c,j} - t_{c,j} := e_{c,j} \quad (30)$$

که خطای پیشگویی در واحد، همان معادله‌ی (8) است. برای سادگی نماد، یک بردار V بعدی $EI = \{EI_1 . 000 . EI_V\}$ به‌عنوان مجموع خطاهای پیش‌بینی شده در تمام کلمات متن تعریف می‌کنیم:

$$\mathrm{EI}_j = \sum_{c=1}^{C} e_{c,j} \quad (31)$$

سپس، مشتق E را با توجه به ماتریس پنهان←خروجی W' گرفته و به‌دست می‌آوریم

$$\frac{\partial E}{\partial w'_{ij}} = \sum_{c=1}^{C} \frac{\partial E}{\partial u_{c,j}} \cdot \frac{\partial u_{c,j}}{\partial w'_{ij}} = \mathrm{EI}_j \cdot h_i \qquad (32)$$

بنابراین معادله‌ی به‌روزشده برای ماتریس پنهان←خروجی W' به‌دست می‌آید،

$$w'^{(\mathrm{new})}_{ij} = w'^{(\mathrm{old})}_{ij} - \eta \cdot \mathrm{EI}_j \cdot h_i \qquad (33)$$

or

$$\mathbf{v}'^{(\mathrm{new})}_{w_j} = \mathbf{v}'^{(\mathrm{old})}_{w_j} - \eta \cdot \mathrm{EI}_j \cdot \mathbf{h} \qquad \text{for } j = 1, 2, \cdots, V. \qquad (34)$$

درک بصری از این معادله‌ی به‌روزشده همان (11) است، به جز این‌که خطای پیش‌گویی است در تمام کلمات متن در لایه خروجی خلاصه شده است. توجه داشته باشید که ما نیاز به درخواست این معادله‌ی به‌روزشده برای هر عنصر از ماتریس‌های پنهان←خروجی برای هر نمونه آموزش هستیم.

مشتق معادله‌ی به‌روزشده برای ماتریس ورودی←پنهان با (12) و (16)، به جز در نظر گرفتن خطای پیش‌بینی e_j که با EI_j جایگزین شده است، یکسان است. به‌طورمستقیم به معادله‌ی به‌روزشده‌ی زیر می‌رسیم:

$$\mathbf{v}^{(\mathrm{new})}_{w_I} = \mathbf{v}^{(\mathrm{old})}_{w_I} - \eta \cdot \mathrm{EH} \qquad (35)$$

که در آن EH یک بردار Nبعدی است، هر جزء آن به‌صورت زیر تعریف شده است:

$$\mathrm{EH}_i = \sum_{j=1}^{V} \mathrm{EI}_j \cdot w'_{ij}. \qquad (36)$$

درک بصری از (35) همانند (16) است.

3. بهینه‌سازی بازده محاسباتی

تاکنون مدل‌هایی که بحث کردیم (مدل "بیگرام"، CBOW و skip-gram) هر دو در فرم اصلی خود، بدون هیچ‌گونه بهینه‌سازی ترفندهای بازده بودن بودند.

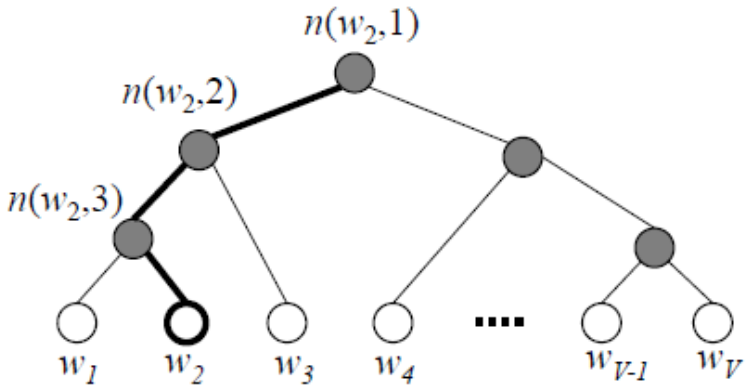
برای همه‌ی این مدل‌ها، دو نمایش از بردار برای هر کلمه در واژگان وجود دارد: بردار ورودی v_m، و بردار خروجی $v_{\dot{m}}$. آموزش بردارهای ورودی ارزان اس ؛ اما یادگیری بردار خروجی بسیار گران است. از معادلات به‌روز شده‌ی (22) و (33)، می‌توانیم دریابیم که، به‌منظور به‌روزرسانی $v_{\dot{m}}$، برای هر نمونه آموزش، نیاز به تکرار از طریق هر کلمه w_j در واژگان، محاسبه خالص ورودی u_j، احتمال پیش‌بینی y_j (یا $y_{c.j}$؛ برای skip-gram)، خطای پیش‌بینی e_j (یا EI_j برای skip-gram)، و سرانجام استفاده از خطای پیش‌بینی خود برای به‌روزرسانی بردار خروجی خود v_j داریم.

انجام چنین محاسباتی برای همه‌ی کلمات برای هر نمونه آموزش بسیار گران است، و بنا به مقیاس واژگان بزرگ و یا شرکتهای آموزش بزرگ غیرعملی است. برای حل این مشکل، یک بینش برای محدود کردن تعداد بردار خروجی در هر آموزش باید به‌روز شود. یکی از این روش‌های زیبا برای دستیابی به این softmax سلسله مراتبی است. دیگر رویکرد که از طریق نمونه‌برداری به‌دست می‌آید در بخش بعدی مورد بحث قرار خواهد گرفت.

هر دو ترفند بهینه‌سازی تنها در محاسبه‌ی به‌روزرسانی برای بردار خروجی هستند. در مشتقات ما، سه مقدار را مورد توجه قرار می‌دهیم: (1) E، تابع هدف جدید؛ (2) $\frac{\partial E}{\partial v_{\dot{w}}}$، معادله‌ی جدید به‌روزشده برای بردار خروجی؛ و (3) $\frac{\partial E}{\partial h}$ مجموع وزنی از پیش‌بینی خطاها برای به‌روزرسانی بردارهای ورودی است.

3.1 Softmax سلسله مراتبی

softmax سلسله مراتبی یک راه موثر برای محاسبه softmax (مورن و بنگیو، 2005، منیح و هینتون، 2009). مدل از یک درخت دودویی برای نمایش تمام کلمات در واژگان استفاده می‌کند. کلمات V باید واحدهای برگ درخت باشند. می‌توان ثابت کرد که N-1 واحد درونی وجود دارد. برای هر واحد برگ، یک مسیر منحصر به فرد از ریشه وجود دارد و این مسیر برای برآورد احتمال از کلمه ارائه شده توسط واحد برگ استفاده می‌شود. شکل 4 یک درخت را نشان می‌دهد.



شکل 4: یک درخت دودویی نمونه برای مدل softmax سلسله مراتبی است. واحدهای سفید کلمات در واژگان هستند، و واحد تاریک واحد درونی هستند. به‌عنوان مثال یک مسیر از ریشه تا w_2 هایلایت شده است. همان‌طور که در مثال نشان داده شده است، طول مسیر $L(w_2) = 4$. $n(w.j)$، jامین واحد در مسیر از ریشه تا کلمه W است.

در مدل softmax سلسله مراتبی، هیچ نمایشی برای بردار خروجی کلمات وجود ندارد. در عوض، هر یک از واحدهای V-1 درونی است یک بردار خروجی $v_{n(w.j)}$ دارد و احتمال این‌که یک کلمه، کلمه‌ی خروجی باشد به‌صورت تعریف شده است

$$p(w = w_O) = \prod_{j=1}^{L(w)-1} \sigma\left(\llbracket n(w, j+1) = \mathrm{ch}(n(w, j)) \rrbracket \cdot \mathbf{v'_{n(w,j)}}^T \mathbf{h}\right) \qquad (37)$$

که در آن $ch(n)$ فرزند سمت چپ واحد n است. $v_{n(w.j)}$ نمایش برداری ("بردار خروجی") از واحد داخلی $n(w.j)$ است، h مقدار خروجی لایه‌های پنهان (در مدل skip-gram است $h = v_{w_I}$؛ و در CBOW، $h = \frac{1}{C}\sum_{c=1}^{C} v_{w_I}$)، $\llbracket x \rrbracket$ یک تابع خاص است که به‌صورت زیر تعریف می‌شود

$$\llbracket x \rrbracket = \begin{cases} 1 & \text{if } x \text{ is true;} \\ -1 & \text{otherwise.} \end{cases} \qquad (38)$$

اجازه دهید تا به‌طورمستقیم، معادله را از طریق یک مثال درک کنیم. بنا به شکل 4، فرض کنید می‌خواهیم احتمال این‌که w_2 کلمه خروجی باشد محاسبه کنیم. این احتمال را به‌عنوان احتمال یک پیاده‌روی تصادفی با شروع از پایان

ریشه در واحد سطح برگ در سوال تعریف می‌کنیم. در هر واحد داخلی (از جمله واحد ریشه)، ما نیاز به اختصاص احتمال رفتن به سمت چپ و رفتن به راست داریم. احتمال رفتن به سمت چپ را در یک واحد n داخلی به‌صورت زیر تعریف می‌کنیم

$$p(n, \text{left}) = \sigma\left({\mathbf{v}'_n}^T \cdot \mathbf{h}\right) \tag{39}$$

که توسط هر دو نمایش بردار واحد داخلی و مقدار خروجی لایه پنهان (که توسط نمایش بردار ورودی کلمه تعیین می‌شود (بازدیدکنندگان)) بیان می‌شود. ظاهرا احتمال رفتن به راست در واحد n به‌صورت زیر است

$$p(n, \text{right}) = 1 - \sigma\left({\mathbf{v}'_n}^T \cdot \mathbf{h}\right) = \sigma\left(-{\mathbf{v}'_n}^T \cdot \mathbf{h}\right) \tag{40}$$

با دنبال کردن مسیر ریشه تا w_2در شکل 4، می‌توانیم احتمال این‌که w_2 کلمه خروجی باشد محاسبه کنیم

$$\begin{aligned} p(w_2 = w_O) &= p\left(n(w_2, 1), \text{left}\right) \cdot p\left(n(w_2, 2), \text{left}\right) \cdot p\left(n(w_2, 3), \text{right}\right) && (41) \\ &= \sigma\left({\mathbf{v}'_{n(w_2,1)}}^T \mathbf{h}\right) \cdot \sigma\left({\mathbf{v}'_{n(w_2,2)}}^T \mathbf{h}\right) \cdot \sigma\left(-{\mathbf{v}'_{n(w_2,3)}}^T \mathbf{h}\right) && (42) \end{aligned}$$

که دقیقا همان چیزی است که در (37) داده شده است. بنابراین بررسی ان نباید سخت باشد

$$\sum_{i=1}^{V} p(w_i = w_O) = 1 \tag{43}$$

ساخت softmax سلسله مراتبی یک توزیع چندجمله‌ای به خوبی تعریف شده در میان همه‌ی کلمات است.

درحال‌حاضر از معادله‌ی به‌روزشده‌ی پارامتر برای نمایش بردار از واحد درونی مشتق می‌گیریم. برای سادگی، ابتدا به مدل اولین کلمه متن نگاه می‌کنیم. بسط معادلات به‌روز شده به مدل CBOW و skip-gram آسان است.

برای سادگی نماد، نماد کوتاه شده‌ی بدون ابهام زیر را معرفی می‌کنیم:

$$[\![\cdot]\!] := [\![n(w, j+1) = \text{ch}(n(w, j))]\!] \tag{44}$$

$$\mathbf{v}'_j := \mathbf{v}'_{n_{w,j}} \tag{45}$$

برای یک نمونه آموزش، تابع خطا به‌صورت زیر تعریف می‌شود

$$E = -\log p(w = w_O | w_I) = -\sum_{j=1}^{L(w)-1} \log \sigma([\![\cdot]\!] {\mathbf{v}'_j}^T \mathbf{h}) \tag{46}$$

با مشتق گرفتن از E با توجه به $v_j h$ به دست می‌آوریم

$$\frac{\partial E}{\partial \mathbf{v}'_j \mathbf{h}} = \left(\sigma([\![\cdot]\!]\mathbf{v}'^T_j \mathbf{h}) - 1\right)[\![\cdot]\!] \quad (47)$$

$$= \begin{cases} \sigma(\mathbf{v}'^T_j \mathbf{h}) - 1 & ([\![\cdot]\!] = 1) \\ \sigma(\mathbf{v}'^T_j \mathbf{h}) & ([\![\cdot]\!] = -1) \end{cases} \quad (48)$$

$$= \sigma(\mathbf{v}'^T_j \mathbf{h}) - t_j \quad (49)$$

که در آن $t_j = 1$ اگر $\|0\| = 1$ و در غیراین‌صورت $t_j = 0$.

سپس مشتق E را با توجه به نمایش بردار واحد داخلی $n(w.j)$ گرفته و به دست می‌آوریم

$$\frac{\partial E}{\partial \mathbf{v}'_j} = \frac{\partial E}{\partial \mathbf{v}'_j \mathbf{h}} \cdot \frac{\partial \mathbf{v}'_j \mathbf{h}}{\partial \mathbf{v}'_j} = \left(\sigma(\mathbf{v}'^T_j \mathbf{h}) - t_j\right) \cdot \mathbf{h} \quad (50)$$

که نتیجه در معادله‌ی به‌روزشده‌ی زیر است:

$$\mathbf{v}'^{(\text{new})}_j = \mathbf{v}'^{(\text{old})}_j - \eta \left(\sigma(\mathbf{v}'^T_j \mathbf{h}) - t_j\right) \cdot \mathbf{h} \quad (51)$$

که باید در $j = \{1.2.000.L(w) - 1\}$ به‌کار برده شود. همچنین می‌توانیم $\sigma\left(v_j^T h\right) - t_j$ را به‌عنوان خطای پیشگویی برای واحد داخلی $n(w.j)$ درک کنیم. "وظیفه " هر واحد داخلی پیش‌بینی آن است که آیا آن فرزند سمت چپ یا راست را در راه رفتن تصادفی دنبال می‌کند. $t_j = 1$ بدین معنی است که باید فرزند سمت چپ را دنبال کند. $t_j = 0$ بدین معنی است که باید فرزند سمت راست را دنبال کند. $\sigma\left(v_j^T h\right)$ نتیجه‌ی پیش‌بینی است. برای یک نمونه آموزش، اگر پیش‌بینی واحد داخلی بسیار نزدیک به حقیقت زمین باشد، نمایش بردار آن v_j بسیار آهسته حرکت خواهد کرد. در غیراین‌صورت v_j در جهت مناسب با حرکت به‌منظور کاهش خطای پیش‌بینی برای این مثال (نزدیک‌تر یا دورتر از h)، حرکت خواهد کرد. این معادله‌ی به‌روزشده می‌تواند برای هر دو مدل CBOW و skip-gram استفاده شود. هنگامی که برای مدل skip-gram استفاده می‌شود، نیاز به تکرار روش به‌روزرسانی برای هر یک از کلمات C در متن خروجی داریم.

به‌منظور کاهش خطا برای یادگیری وزن ورودی← پنهان، مشتقات نسبی E با توجه به خروجی لایه‌های پنهان گرفته می‌شود و رابطه‌ی زیر بدست می‌آید

$$\frac{\partial E}{\partial \mathbf{h}} = \sum_{j=1}^{L(w)-1} \frac{\partial E}{\partial \mathbf{v}'_j \mathbf{h}} \cdot \frac{\partial \mathbf{v}'_j \mathbf{h}}{\partial \mathbf{h}} \quad (52)$$

$$= \sum_{j=1}^{L(w)-1} \left(\sigma({\mathbf{v}'_j}^T \mathbf{h}) - t_j\right) \cdot \mathbf{v}'_j \quad (53)$$

$$:= \mathrm{EH} \quad (54)$$

که می‌تواند به‌طورمستقیم با (23) برای به‌دست آوردن معادله به‌روزشده برای بردار ورودی CBOW جایگزین شود. برای مدل skip-gram ، ما نیاز به محاسبه ارزش EH برای هر کلمه در متن skip-gram و جمع مقادیر EH با (35) برای به‌دست آوردن معادله‌ی به‌روزشده برای بردار ورودی داریم.

از معادلات به‌روز شده، می‌توانیم ببینیم که پیچیدگی محاسباتی در آموزش در هر کلمه متن از O (V) به O(log(v)) کاهش می‌یابد که یک بهبود بزرگ در سرعت است. تقریبا هنوز تعداد یکسانی از پارامترها داریم (بردار V-1 برای واحدهای داخلی در مقایسه با بردار خروجی اصل V برای کلمات).

3.2 نمونه‌گیری منفی

ایده‌ی نمونه‌گیری منفی ساده‌تر از softmax سلسله مراتبی است: به منظور مقابله با داشتن بردار خروجی بیش از حد که نیاز به به‌روزرسانی در هر تکرار دارند، ما تنها یک نمونه از آنها را به‌روزرسانی می‌کنیم.

ظاهرا کلمه خروجی (به‌عنوان مثال، حقیقت زمین، و یا نمونه مثبت) باید در نمونه ما نگه داشته شود و به‌روز شود و ما نیاز به نمونه‌گیری چند کلمه به‌عنوان نمونه منفی هستیم. یک توزیع احتمالاتی برای فرآیند نمونه‌برداری مورد نیاز است و می‌توان آن‌را به دلخواه انتخاب کرد. این توزیع را با نام توزیع صدا می‌نامیم و با $P_n(w)$ نشان می‌دهیم.

در vec2word، به‌جای استفاده از فرم نمونه منفی است که توزیع چندجمله‌ای خوبی تولید می‌کند، نویسندگان استدلال می‌کنند که هدف آموزش ساده قادر به تولید کلمه با کیفیت بالا است:

$$E = -\log \sigma({\mathbf{v}'_{w_O}}^T \mathbf{h}) - \sum_{w_i \in \mathcal{W}_{\text{neg}}} \log \sigma(-{\mathbf{v}'_{w_i}}^T \mathbf{h}) \tag{55}$$

که در آن w_O خروجی کلمه (به‌عنوان مثال، نمونه مثبت) است، و v_{wO} بردار خروجی آن است. h مقدار خروجی لایه‌های پنهان است: $h = \frac{1}{C}\sum_{c=1}^{C} v_{w_c}$ در مدل CBOW و $h = v_{w_I}$ در مدل skip-gram است؛ $W_{neg} =$ $\{w_i | i = 1.000.K\}$ مجموعه ای از کلمات است که بر اساس $P_n(w)$ نمونه می‌باشد و در نتیجه آنها نمونه منفی هستند.

برای به دست آوردن معادلات به‌روزشده‌ی بردار کلمه تحت نمونه منفی، ابتدا مشتق E با توجه به ورودی خالص واحد خروجی w_j به‌دست می‌آوریم:

$$\frac{\partial E}{\partial {\mathbf{v}'_{w_j}}^T \mathbf{h}} = \begin{cases} \sigma({\mathbf{v}'_{w_j}}^T \mathbf{h}) - 1 & \text{if } w_j = w_O \\ \sigma({\mathbf{v}'_{w_j}}^T \mathbf{h}) & \text{if } w_j \in \mathcal{W}_{\text{neg}} \end{cases} \tag{56}$$

$$= \sigma({\mathbf{v}'_{w_j}}^T \mathbf{h}) - t_j \tag{57}$$

که در آن t_j یک برچسب کلمه‌ی w_j است. $t = 1$ وقتی که w_jیک نمونه مثبت است؛ درغیراین‌صورت $t = 0$ است. سپس مشتق E را با توجه به بردار خروجی کلمه w_j بدست می‌آوریم

$$\frac{\partial E}{\partial \mathbf{v}'_{w_j}} = \frac{\partial E}{\partial {\mathbf{v}'_{w_j}}^T \mathbf{h}} \cdot \frac{\partial {\mathbf{v}'_{w_j}}^T \mathbf{h}}{\partial \mathbf{v}'_{w_j}} = \left(\sigma({\mathbf{v}'_{w_j}}^T \mathbf{h}) - t_j\right) \mathbf{h} \tag{58}$$

که نتیجه در معادله‌ی به‌روزشده‌ی زیر برای بردار خروجی آن است:

$$\mathbf{v}'^{(\text{new})}_{w_j} = \mathbf{v}'^{(\text{old})}_{w_j} - \eta \left(\sigma({\mathbf{v}'_{w_j}}^T \mathbf{h}) - t_j\right) \mathbf{h} \tag{59}$$

که تنها نیاز دارد تا در $w_j \epsilon \{w_O\} \cup W_{neg}$ به جای هر کلمه در واژگان به‌کار برده شود. این نشان می‌دهد که چرا ممکن است یک مقدار قابل توجهی تلاش محاسباتی را در هر تکرار ذخیره کند.

درک بصری از معادله‌ی به‌روزشده‌ی فوق باید همانند (11) باشد. این معادله می‌تواند برای هر دو مدل CBOW و skip-gram استفاده شود. برای مدل skip-gram، این معادله را برای متن یک کلمه‌ای در یک زمان به‌کار می‌بریم.

خطا در لایه‌های پنهان است و در نتیجه بردارهای ورودی از کلمات را به‌روز می‌کند، بنابراین نیاز به مشتق‌گیری E با توجه به خروجی لایه مخفی داریم،

$$\frac{\partial E}{\partial \mathbf{h}} = \sum_{w_j \in \{w_O\} \cup \mathcal{W}_{\text{neg}}} \frac{\partial E}{\partial {\mathbf{v}'_{w_j}}^T \mathbf{h}} \cdot \frac{\partial {\mathbf{v}'_{w_j}}^T \mathbf{h}}{\partial \mathbf{h}} \tag{60}$$

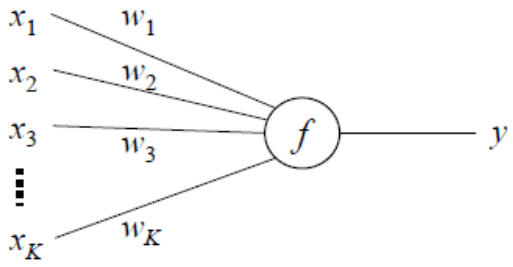
$$= \sum_{w_j \in \{w_O\} \cup \mathcal{W}_{\text{neg}}} \left(\sigma({\mathbf{v}'_{w_j}}^T \mathbf{h}) - t_j\right) \mathbf{v}'_{w_j} := \text{EH} \tag{61}$$

با متصل کردن EH به (23) معادله‌ی به‌روزشده بردارهای ورودی را از مدل CBOW بدست می‌آوریم. برای مدل skip-gram، نیاز به محاسبه ارزش EH برای هر کلمه در متن skip-gram و جمع مقادیر EH با (35) برای به‌دست آوردن معادله‌ی به‌روز بردار ورودی داریم.

A. مبانی بازگشت انتشار

A.1 یادگیری الگوریتم‌ها برای یک واحد

شکل 5 یک نورون مصنوعی (واحد) را نشان می‌دهد. $\{x_1 . 000 . x_K\}$ مقادیر ورودی هستند. $\{w_1 . 000 . w_K\}$ وزن‌ها هستند؛ y یک خروجی اسکالر است. و f تابع لینک (به‌نام تابع فعال/ تصمیم/ انتقال نیز یاد می‌شود).



شکل 5: یک نرون مصنوعی

واحد به روش زیر کار می‌کند:

$$y = f(u), \tag{62}$$

که در آن u عددی اسکالر است، که ورودی (یا "ورودی جدید ") از نورون است. u به‌صورت زیر تعریف می‌شود

$$u = \sum_{i=0}^{K} w_i x_i. \qquad (63)$$

با استفاده از نماد بردار، می‌توانیم بنویسیم

$$u = \mathbf{w}^T \mathbf{x} \qquad (64)$$

توجه داشته باشید که در اینجا ما از اصطلاح مورب در u چشم‌پوشی می‌کنیم. به‌منظور داشتن اصطلاح مورب، به‌سادگی می‌توانید

بعد ورودی را (به‌عنوان مثال، x_0)که ثابت 1 است اضافه کنید.

ظاهرا، توابع لینک‌های مختلف نتیجه‌ی رفتارهای متمایز نورون است. در اینجا در مورد دو مثال انتخاب از توابع لینک بحث می‌کنیم.

انتخاب اولین مثال از f (u)، تابع مرحله واحد است (تابع مرحله Heaviside):

$$f(u) = \begin{cases} 1 & \text{if } u > 0 \\ 0 & \text{otherwise} \end{cases} \qquad (65)$$

یک نرون با این تابع اتصال یک پرسپترون نامیده می‌شود. الگوریتم یادگیری برای پرسپترون الگوریتم پرسپترون است. معادله‌ی به‌روزشده‌ی آن به‌صورت زیر تعریف می‌شود:

$$\mathbf{w}^{(\text{new})} = \mathbf{w}^{(\text{old})} - \eta \cdot (y - t) \cdot \mathbf{x} \qquad (66)$$

که در آن t برچسب (استاندارد طلایی) و η نرخ یادگیری($\eta > 0$) است. توجه داشته باشید که پرسپترون یک طبقه‌بندی خطی است، که به‌معنی ظرفیت توضیحات آن می‌تواند بسیار محدود باشد. اگر ما می‌خواهیم به توابع پیچیده‌تری دست یابیم، نیاز به استفاده از یک مدل غیرخطی داریم.

انتخاب دومین مثال از f (u) تابع لجستیک (یک نوع از رایج‌ترین تابع sigmoid است)، که به‌صورت زیر تعریف می‌شود

$$\sigma(u) = \frac{1}{1 + e^{-u}} \qquad (67)$$

تابع لجستیک دو خاصیت خوب اولیه دارد: (1) خروجی y همیشه بین 0 و 1 است و (2) برخلاف یک تابع مرحله واحد $\sigma(u)$ ساده و مشتق‌پذیر است، ساخت مشتق معادله‌ی به‌روزشده بسیار آسان است.

توجه داشته باشید که $\sigma(u)$ دارای دو ویژگی زیر است که می‌تواند بسیار مناسب باشد و در مشتقات زیردنباله استفاده خواهد شد:

$$\sigma(-u) = 1 - \sigma(u) \tag{68}$$

$$\frac{d\sigma(u)}{du} = \sigma(u)\sigma(-u) \tag{69}$$

ما از گرادیان نزولی تصادفی به‌عنوان الگوریتم یادگیری این مدل استفاده می‌کنیم. به‌منظور مشتق‌گیری معادله‌ی به‌روزشده، ما نیاز به تعریف تابع خطا، به‌عنوان مثال، هدف آموزش داریم. تابع هدف زیر به نظر مناسب می‌رسد:

$$E = \frac{1}{2}(t - y)^2 \tag{70}$$

از E با توجه به w_i مشتق می‌گیریم،

$$\frac{\partial E}{\partial w_i} = \frac{\partial E}{\partial y} \cdot \frac{\partial y}{\partial u} \cdot \frac{\partial u}{\partial w_i} \tag{71}$$

$$= (y - t) \cdot y(1 - y) \cdot x_i \tag{72}$$

به‌طوری که $\frac{\partial y}{\partial u} = y(1 - y)$ زیرا $y = f(u) = \sigma(u)$ و (68) و (69). بنا به مشتق می‌توانیم گرادیان نزولی تصادفی را اعمال کنیم:

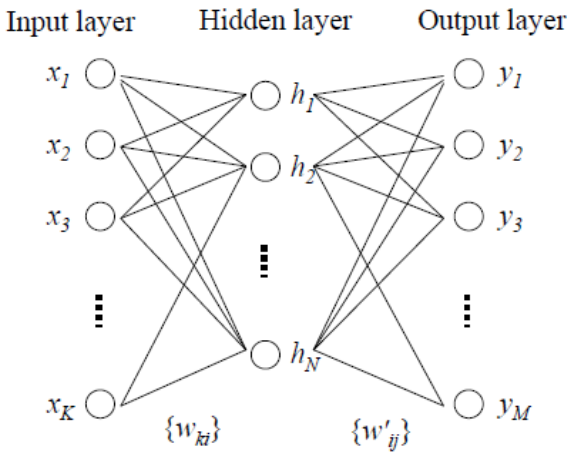
$$\mathbf{w}^{(\text{new})} = \mathbf{w}^{(\text{old})} - \eta \cdot (y - t) \cdot y(1 - y) \cdot \mathbf{x}. \tag{73}$$

A.2 پس انتشار با شبکه‌ی چند لایه

شکل 6 یک شبکه عصبی چند لایه با لایه‌ی ورودی $\{x_k\} = \{x_1 . 000 . x_K\}$؛ لایه‌ی پنهان $\{h_i\} = \{h_1 . 000 . h_N\}$، و یک لایه خروجی $\{y_j\} = \{y . 000 . y_M\}$ را به وضوح نشان می‌دهد. بنابراین از K؛ i؛ j به‌عنوان

اندیس برای واحدهای ورودی، پنهان، و لایه خروجی استفاده می‌کنیم. از u_i و u_j برای نشان دادن ورودی خالص واحدهای لایه‌ی پنهان و واحدهای لایه خروجی استفاده می‌کنیم.

ما می‌خواهیم از معادله‌ی به‌روزشده برای یادگیری وزن‌های w_{ki} بین ورودی و لایه‌های پنهان و w_{ij} بین لایه‌های پنهان و خروجی مشتق بگیریم. ما فرض می‌کنیم که تمام واحدهای محاسبه (به‌عنوان مثال، واحدها در لایه پنهان و لایه خروجی) از تابع لجستیک $\sigma(u)$ به‌عنوان تابع لینک استفاده می‌کند.



شکل 6: شبکه‌های عصبی چند لایه با یک لایه پنهان

بنابراین، برای یک واحد h_i در لایه مخفی، خروجی آن به‌صورت زیر تعریف می‌شود

$$h_i = \sigma(u_i) = \sigma\left(\sum_{k=1}^{K} w_{ki} x_k\right). \qquad (74)$$

به‌طور مشابه، برای یک واحد y_j در لایه خروجی، خروجی خود را به‌صورت زیر تعریف می‌کند

$$y_j = \sigma(u'_j) = \sigma\left(\sum_{i=1}^{N} w'_{ij} h_i\right). \qquad (75)$$

ما از تابع مجموع مربعات خطا داده شده در زیر استفاده می‌کنیم

$$E(\mathbf{x}, \mathbf{t}, \mathbf{W}, \mathbf{W}') = \frac{1}{2}\sum_{j=1}^{M}(y_j - t_j)^2, \qquad (76)$$

که در آن $W = \{w_{ki}\}$، یک ماتریس وزن $K \times N$(ورودی-پنهان)، و ، $W' = \{w_{ij}\}$یک ماتریس وزن $N \times M$(پنهان و خروجی) است. $t = \{t_1 . 000 . t_M\}$، یک بردار M -بعدی است، که برچسب استاندارد طلایی از خروجی است.

برای به‌دست آوردن معادلات به‌روزشده برای w_{ki} و w_{ij} نیاز به مشتق‌گیری از تابع خطا E با توجه به وزن داریم. برای مشتق‌گیری ساده، ما شروع به محاسبه مشتق برای-سمت راست‌ترین لایه (به‌عنوان مثال، لایه خروجی)، و سپس حرکت به سمت چپ می‌کنیم. برای هر لایه، محاسبات را به سه مرحله تقسیم می‌کنیم، محاسبه مشتق خطا با توجه به خروجی، ورودی خالص و وزن. این فرایند در زیر نشان داده شده است.

ما از لایه خروجی شروع می‌کنیم. اولین گام محاسبه مشتق از خطا w.r.t. خروجی است:

$$\frac{\partial E}{\partial y_j} = y_j - t_j. \qquad (77)$$

دومین گام برای محاسبه‌ی مشتق خطا با توجه به ورودی خالص از لایه‌ی خروجی است. توجه داشته باشید که در هنگام گرفتن مشتقات با توجه به چیزی، ما نیاز به حفظ ثابت هر چیز دیگری داریم. همچنین توجه داشته باشید که این مقدار بسیار مهم است زیرا مجددا و چندین بار در محاسبات بعدی مورد استفاده قرار خواهد گرفت. برای سادگی آن را به‌عنوان EI_j نشان می‌دهیم:

$$\frac{\partial E}{\partial u'_j} = \frac{\partial E}{\partial y_j} \cdot \frac{\partial y_j}{\partial u'_j} = (y_j - t_j) \cdot y_j(1 - y_j) := \mathrm{EI}'_j \qquad (78)$$

گام سوم، محاسبه‌ی مشتق خطا با توجه به وزن بین لایه‌های پنهان و لایه خروجی است.

$$\frac{\partial E}{\partial w'_{ij}} = \frac{\partial E}{\partial u'_j} \cdot \frac{\partial u'_j}{\partial w'_{ij}} = \mathrm{EI}'_j \cdot h_i \qquad (79)$$

تاکنون، معادله‌ی به‌روزشده برای وزن‌های بین لایه‌های پنهان و لایه خروجی را به دست آوردیم.

$$w'^{(\mathrm{new})}_{ij} = w'^{(\mathrm{old})}_{ij} - \eta \cdot \frac{\partial E}{\partial w'_{ij}} \qquad (80)$$

$$= w'^{(\mathrm{old})}_{ij} - \eta \cdot \mathrm{EI}'_j \cdot h_i. \qquad (81)$$

که در آن $\eta > 0$ نرخ یادگیری است.

ما می‌توانیم همین سه مرحله را برای به‌دست آوردن معادله‌ی به‌روزشده برای وزن‌های از لایه قبلی تکرار کنیم، که در اصل ایده‌ی پس از انتشار است.

گام اول را تکرار می‌کنیم و مشتق خطا را با توجه به خروجی لایه‌های پنهان محاسبه می‌کنیم. توجه داشته باشید که خروجی لایه‌های پنهان مربوط به تمام واحدها در لایه خروجی است.

$$\frac{\partial E}{\partial h_i} = \sum_{j=1}^{M} \frac{\partial E}{\partial u'_j} \frac{\partial u'_j}{\partial h_i} = \sum_{j=1}^{M} \mathrm{EI}'_j \cdot w'_{ij}. \qquad (82)$$

سپس مرحله دوم را برای محاسبه مشتق خطا در رابطه با ورودی خالص از لایه‌های پنهان تکرار می‌کنیم. این مقدار بسیار مهم است، و ما آن را با EI_i مشخص می‌کنیم.

$$\frac{\partial E}{\partial u_i} = \frac{\partial E}{\partial h_i} \cdot \frac{\partial h_i}{\partial u_i} = \sum_{j=1}^{M} \mathrm{EI}'_j \cdot w'_{ij} \cdot h_i(1 - h_i) := \mathrm{EI}_i \qquad (83)$$

سپس مرحله سوم را برای محاسبه‌ی مشتق خطا با توجه به وزن بین لایه ورودی و لایه‌های پنهان تکرار می‌کنیم.

$$\frac{\partial E}{\partial w_{ki}} = \frac{\partial E}{\partial u_i} \cdot \frac{\partial u_i}{\partial w_{ki}} = \mathrm{EI}_i \cdot x_k, \qquad (84)$$

در نهایت، می‌توانیم معادله‌ی به‌روزشده برای وزن‌ها بین لایه ورودی و لایههای پنهان را بدست آوریم.

$$w_{ki}^{(\mathrm{new})} = w_{ki}^{(\mathrm{old})} - \eta \cdot \mathrm{EI}_i \cdot x_k. \qquad (85)$$

از مثال بالا، می‌توانید ببینید که نتایج میانی (EI_i) در هنگام محاسبه‌ی مشتقات برای یک لایه را می‌توان برای لایه قبلی مورد استفاده مجدد قرار داد. تصور کنید لایه دیگری قبل از لایه ورودی وجود دارد، بنابراین EI_i می‌تواند برای ادامه محاسبه زنجیره‌ای مشتقات مورد استفاده مجدد قرار گیرد. مقایسه معادلات (78) و (83) را، ممکن است در (83) بیابیم عامل $\sum_{j=1}^{M} EI_j w_{ij}$ دقیقا شبیه "خطا" از لایه‌ی پنهان واحد h_i است. ممکن است این اصطلاح را به‌عنوان خطا از لایه بعدی تفسیر کنیم، و این انتشار ممکن است بیشتر به عقب برگردد در صورتی که شبکه لایه‌های پنهان زیادی داشته باشد.

B. wevi: بازرس بصری جاسازی کلمه

یک رابط بصری تعاملی، wevi (بازرس بصری جاسازی کلمه)، به‌صورت آنلاین برای نشان دادن سازوکار مدل‌های شرح داده شده در این مقاله در دسترس است. شکل 7 یک تصویر از wevi را نشان می‌دهد.

نسخه‌ی نمایشی به کاربر اجازه می‌دهد تا حرکت بردارهای ورودی و بردارهای خروجی را به‌عنوان نمونه آموزش به صورت بصری بررسی نماید. روند آموزش را می‌توان در در حالت دسته‌ای اجرا کرد (به‌عنوان مثال، مصرف 500 مورد آموزش در یک سطر)، که می‌تواند از الگوهای ماتریس وزن و بردار کلمه مربوطه آشکار شود. مولفه‌های اصلی تجزیه‌وتحلیل (PCA) برای تجسم بردارهای با ابعاد "بالا " در یک طرح پراکندگی 2 بعدی استفاده شده است. نسخه‌ی نمایشی هر دو مدل CBOW و skip-gram را پشتیبانی می‌کند.

پس از مدل آموزش، کاربر به‌صورت دستی می‌تواند یک یا چند واحد لایه ورودی را فعال کرده و بازرسی کند که واحد لایه‌های پنهان و واحد لایه خروجی فعال باشند. کاربر همچنین می‌تواند داده‌های آموزشی، اندازه لایه پنهان و نرخ آموزش را سفارشی کند. چند تا از مجموعه داده‌های آموزشی از پیش تعیین شده ارائه شده است، که می‌تواند نتایج مختلفی که به نظر جالب می‌رسد تولید کند، مانند استفاده از یک فرهنگ لغت اسباب‌بازی برای تکثیر کلمه معروف: پادشاه - ملکه = مرد- زن.

امید است که با تعامل با این نسخه‌ی نمایشی به سرعت نحوه‌ی عملکرد مدل را به‌دست آورید. این سیستم در http://bit.ly/wevi-online در دسترس است. کد منبع در http://github.com/ronxin/wevi در دسترس است.

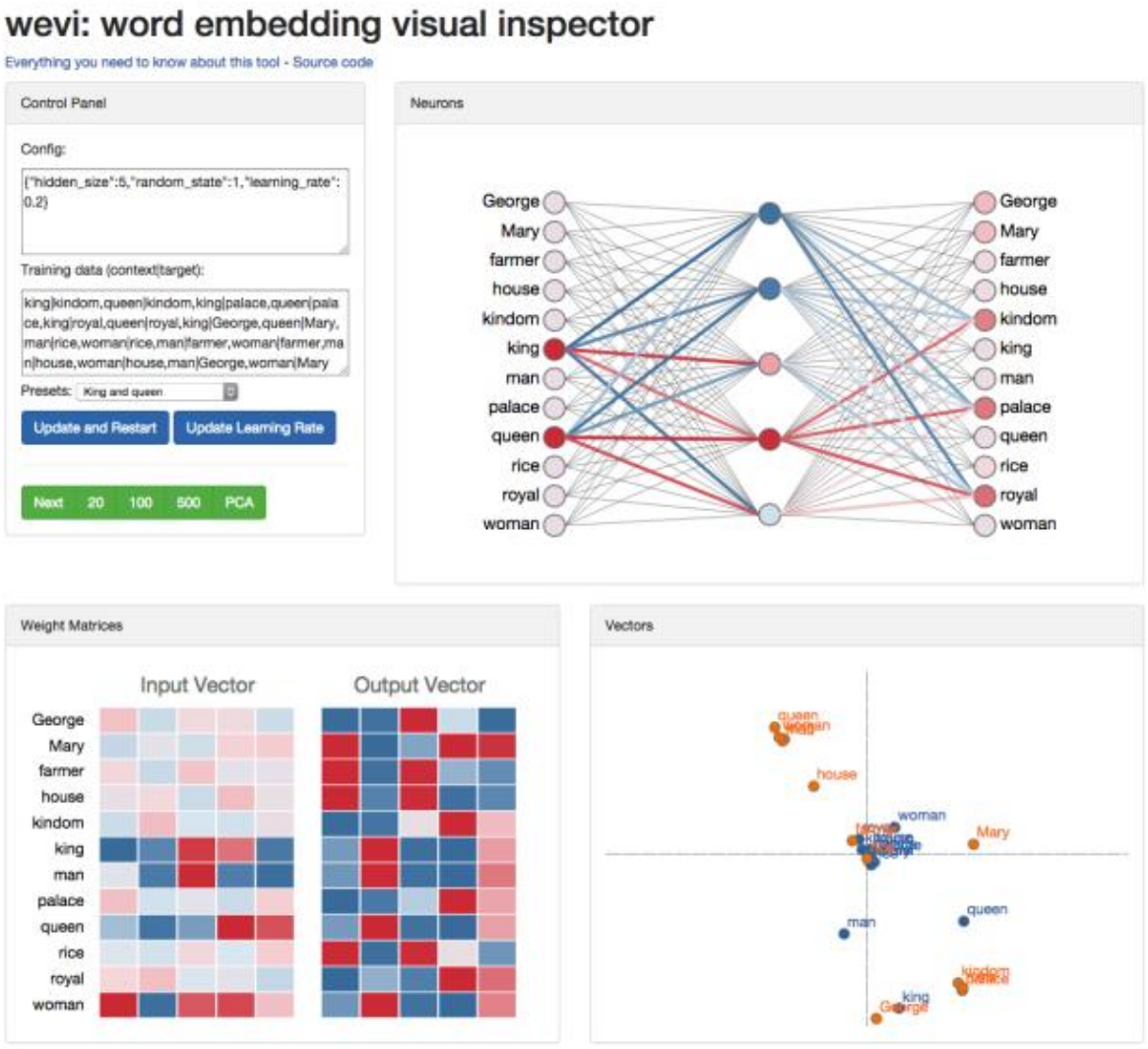


Figure 7: wevi screenshot (`http://bit.ly/wevi-online`)

References

Goldberg, Y. and Levy, O. (2014). word2vec explained: deriving mikolov et al.'s negative sampling word-embedding method. arXiv: 1402.3722[cs, stat]. arXiv: .1402.3722

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013a). E? Cient estimation of word representations in vector space. arXiv preprint arXiv: .1301.3781

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013b). Distributed representations of words and phrases and their compositionality. In Advances in Neural Information Processing Systems, pages3111-3119

Mnih, A. and Hinton, G. E. (2009). A scalable hierarchical distributed language model. In Koller, D., Schuurmans, D., Bengio, Y., and Bottou, L., editors, Advances in Neural Information Processing Systems 21, pages 1081 {1088. Curran Associates, Inc.

Morin, F. and Bengio, Y. (2005). Hierarchical probabilistic neural network language model. In AISTATS, volume 5, pages 246 {252. Citeseer.