2 Understanding word2vec

2.a

assume $x \in R^d$ and let $\vec{c} \in d$ be a vector with constant entries.

$$softmax(\vec{x}+\vec{c})_i = \frac{\exp{(x_i+c)}}{\sum_{j=1}^{d}\exp{(x_j+c)}} = \frac{\exp x_i \exp c}{\exp(c)\sum_{j=1}^{d}\exp{(x_j)}} = \frac{\exp x_i}{\sum_{j=1}^{d}\exp{(x_j)}} = softmax(\vec{x})$$

2.b

let y be one-hot vector.

$$-\sum_{w\in W} y_w \log(\hat{y_w}) = -\sum_{w\neq o} 0 * \log(\hat{y_w}) + (-\log(\hat{y_o})) = log(\hat{y_o})$$

2.c

$$J(\vec{V_c}, o, U) = -log(\frac{\exp(\vec{\mu_o^T}\vec{V_c})}{\sum_{w\in W}\exp(\vec{\mu_w^T}\vec{V_c}}) = log(\sum_{w\in W}\exp(\vec{\mu_w^T}\vec{V_c}) - \vec{\mu_o^T}\vec{V_c}$$

$$\nabla_{\vec{v_c}(J)} = -\mu_o + \frac{1}{\sum_{w\in W}\exp(\vec{\mu_w^T}\vec{V_c})}\sum_{r\in W}\vec{\mu_r}\exp(\vec{\mu_r^T}\vec{V_c}) = \sum_{r\in W}\vec{\mu_r}\frac{\exp(\vec{\mu_r^T}\vec{V_c})}{\sum_{w\in W}\exp(\vec{\mu_w^T}\vec{V_c})} =$$

$$\sum_{r\in W} P(o=r|C=c)\vec{\mu_r} - \vec{\mu_o} = \sum_{r\in W}(\hat{y})\vec{\mu}_r - \vec{\mu}_0 =$$

$$E[\vec{\mu}]_{\vec{\mu}\sim P(O|C=c)} - \vec{\mu}_0$$

2.d

for $w = 0$:

$$\nabla_{\vec{\mu}_w} = \vec{V_c}\exp(\vec{\mu_w^T}\vec{V_c})\frac{1}{\sum_{r\in W}\exp(\vec{\mu_r^T}\vec{V_c})} - \vec{V_c} = \vec{V_c}[(\hat{Y}_w) - 1]$$

for $w \neq 0$:

$$\nabla_{\vec{\mu}_w} = \vec{V_c}(\hat{\vec{Y}})_w$$

2.e

since 6 is applied element-wise, the entries of the derivative vector will be the derivative of σx_i w.r.t for each entry i. thus, will calculate the derivative for single element x_i:

$$\frac{\partial\sigma(x_i)}{\partial x_i} = -\frac{1}{(1+\exp(-x_i))^2}(-\exp^{-x_i}) = -\frac{1+(-\exp^{-x_i})}{(1+\exp(-x_i))^2} - \frac{-1}{(1+\exp(-x_i))^2} =$$

$$\sigma(x_i) - \sigma^2(x_i) = \sigma(x_i)(1-\sigma(x_i)) = \sigma(x_i)\sigma(-x_i)$$

2.f

$$\nabla_{\vec{\mu}_o}[J_{neg}] = -[\frac{1}{\sigma(\vec{\mu}_o^T V_c)}\sigma(\vec{\mu}_o^T V_c)\sigma(-\vec{\mu}_o^T V_c)V_c] - [\sum_{k=1}^{K}\frac{1}{\sigma(\vec{\mu}_k^T V_c)}\sigma(\vec{\mu}_k^T V_c)\sigma(-\vec{\mu}_k^T V_c)\vec{O}_{o\notin(\mu_1,..,\mu)} =$$
$$= -\sigma(-\vec{\mu}_o^T V_c)V_c$$

for $j \in (1, 2...K)$:

$$\nabla_{\vec{\mu}_j}[J_{neg}] = -[\frac{1}{\sigma(\vec{\mu}_o^T V_c)}\sigma(\vec{\mu}_o^T V_c)\sigma(-\vec{\mu}_o^T V_c)\vec{O}] - [\sum_{k=1}^{K}\frac{1}{\sigma(\vec{\mu}_k^T V_c)}\sigma(\vec{\mu}_k^T V_c)\sigma(-\vec{\mu}_k^T V_c)frac\partial[-\mu_k^T\vec{V}_c]\partial\vec{\mu}_j) =$$
$$= +\sigma(-\vec{\mu}_j^T V_c)V_c$$

$$\nabla_{\vec{V}_c}[J_{neg}] = -[\frac{1}{\sigma(\vec{\mu}_o^T V_c)}\sigma(\vec{\mu}_o^T V_c)\sigma(-\vec{\mu}_o^T V_c)V_c] - [\sum_{k=1}^{K}\frac{1}{\sigma(\vec{\mu}_k^T V_c)}\sigma(\vec{\mu}_k^T V_c)\sigma(-\vec{\mu}_k^T V_c)\vec{\mu}_k =$$
$$\sum_{k=1}^{K}\sigma(\vec{\mu}_k^T\vec{V}_c)\vec{\mu}_k - \sigma(\vec{\mu}_o^T\vec{V}_c)\vec{\mu}_o$$

In naïve soft-max one has to update the parameters - going through the entire corpus in each iteration, whereas here, we only consider k "additional" vecotrs.

2.g.i

$$\frac{\partial J_skip-gram}{\partial U} = \sum_{-M<=j<=M;j\neq o}\frac{\partial(\vec{V}_c, W_{t+j}, U}{\partial U}$$

2.g.ii

$$\frac{\partial J_skip-gram}{\partial \vec{V}_c} = \sum_{-M<=j<=M;j\neq o}\frac{\partial(\vec{V}_c, W_{t+j}, U}{\partial \vec{V}_c}$$

2.g.iii

$$\frac{\partial J_skip-gram}{\partial \vec{V}_w} = 0$$

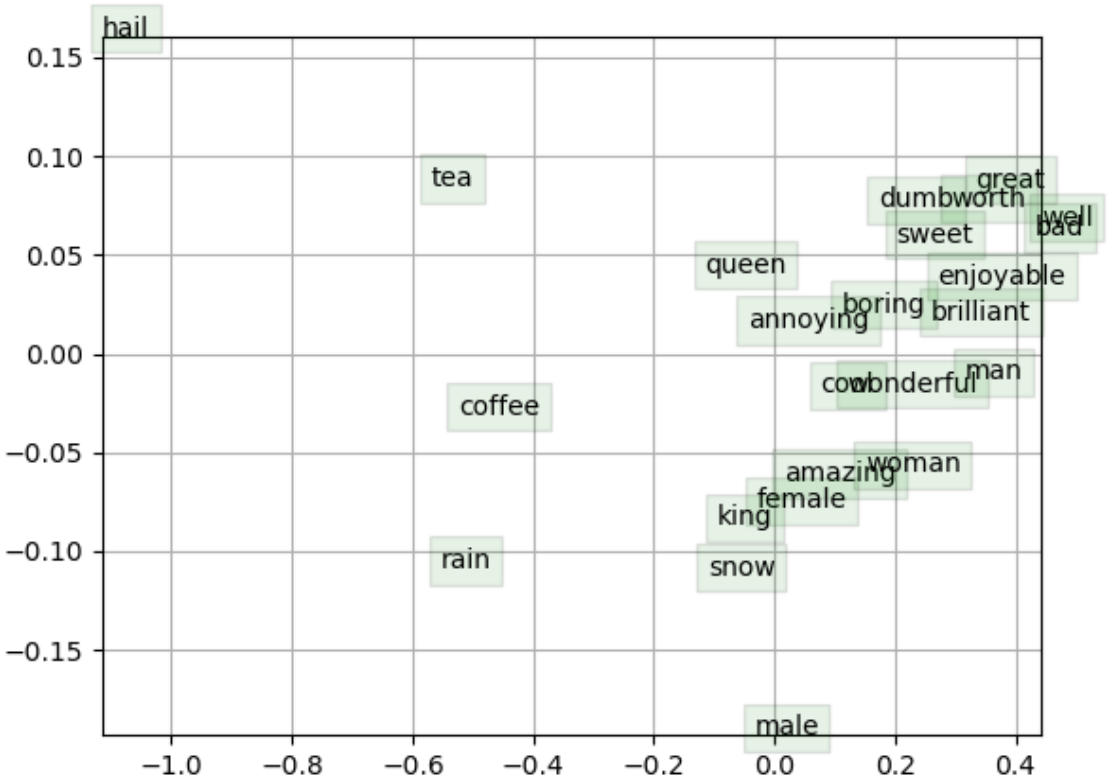


Figure 1: Word Vetors SVD plot

3 Implementing word2vec

3.e

In figure 1 we can see several intersting phenomena, regarding the clusters, trends and algebric operations.

3.e.i clusters

The most disticnt cluster, appears on the top right corner of the plot. It includes words such as ("great", "sweet", "dumb") and others, all can be seen in figure 1. all words in this cluster are adjectives. when looking at the left cluster, which is less distinctive, since words are further away from each other, we can see nouns (tea, coffee, rain, hail). The reasons why these words are not clustered as well could be several: their appearnces on the text might be too little, or biased (women drink tea, men drink coffee), or the sub dimension we are looking at ommits their relationship. In the middle, we can see a sub-cluster when looking at x values - king, queen, male, female. These words are considered as nouns, but they somewhat describe an object (king as a title for a man). We might be enforcing here what we wish to see, but it makes sense in the (x,y) coordinate system and according to the word2vec article.

3.e.ii trends

When looking at x-axis trends, we see the change from nouns (hail, tea, coffee) to adjectives (great, well, bad). In the middle we see titles (king, queen, etc.). For some reason, the word snow appears closer to king and female, than to rain. This of course doesn't support our conclusions, but we suspect it to be an outlier, brought-forth by biased text or the specific dim-reduction applied in SVD. On the y-axis, we couldn't see any clear trends.

3.e.iii algebric operations

Like word2vec suggested, we tried to perform some algerbric operations. Below is an evaluation of word-coordinates using the grid we added to the plot. lets take a look at the words:

$$king : (x = -0.10, y = -0.09)$$

$$male : (x = -0.03, y = -0.19)$$

$$female : (x = -0.03, y = -0.08)$$

$$queen : (x = -0.12, y = 0.04)$$

$$result = king - male + female = (-0.1, 0.02)$$

when using L_2 as distance function, annoted d,

$$d(result, queen) = 0.02$$

thus makes it the closest vector to results.

(4.4) Boring-solution:

- formulate as an optimization problem:

$$\max_{\theta} \left[\sum_{(c,o)\in D} \#(c,o) \cdot \log\left[P_\theta(o|c)\right] \right]$$

subject to:

$$\forall c: \sum_{o'\in D} P_\theta(o'|c) = 1 \iff$$

$$\iff \left[1 - \sum_{o'\in D} P_\theta(o'|c)\right] = 0$$

$$\Rightarrow \quad \bullet \; L(\theta, \lambda) = \sum_{(c,o)\in D} \#(c,o) \cdot \log\left[P_\theta(o|c)\right] + \sum_{c\in D} \lambda_c \left[1 - \sum_{o'\in D} P_\theta(o'|c)\right]$$

$$\Rightarrow \quad \forall (c,o): \quad \frac{\partial L}{\partial P_\theta(o|c)} = \frac{\#(c,o)}{P_\theta(o|c)} - \lambda_c = 0 \quad \text{iff}$$

$$P_\theta(o|c) = \frac{\#(c,o)}{\lambda_c}.$$

⇓

①

$$\Downarrow$$

$\Rightarrow$ From our constraint:

$$\forall c: \sum_{o' \in D} p_\theta(o'|c) = 1 \quad \text{iff} \quad \sum_{o' \in D} \left[\frac{\#(c,o')}{\lambda_c} \right] = 1$$

$$\text{iff} \quad \boxed{\lambda_c = \sum_{o' \in D} \#(c,o')}$$

$\Rightarrow \forall (c,o) \in D:$

$$\boxed{p_{\theta_{opt}}(o|c) = \frac{\#(c,o)}{\sum_{o' \in D} \#(c,o')}}$$

ניתן לראות כי $P_{\theta_{opt}}(o|c) \geq 0$ ולכן מקיים אילוץ

וכן - אילוץ זה נדרש שמכבים כאלו הגיוניים.

(4,4) Full-solution:

- Define 2 probability distributions over $\mathcal{D}$:

$\forall (c,o) \in \mathcal{D}$: $P[(c,o)] = \frac{\#(c,o)}{|\mathcal{D}|}$; $Q[(c,o)] = \frac{P_\vartheta(o|c) \cdot \#(c)}{|\mathcal{D}|}$

$\#(c) \triangleq \sum_{o'} \#(c,o')$

- Re-formulate opt problem:

$$\arg\min_\vartheta [J(\vartheta)] = \arg\min_\vartheta \left[- \sum_{(c,o)\in\mathcal{D}} \#(c,o) \cdot \log [P_\vartheta[o|c]] \right] =$$

$$= \arg\min_\vartheta \left[- \sum_{(c,o)\in\mathcal{D}} \left[\frac{\#(c,o)}{b} \right] \cdot \left[\log [P_\vartheta(o|c)] + \log[\#(c)] - \log[|\mathcal{D}|] \right] \right] =$$

$$= \arg\min_\vartheta \left[- \sum_{(c,o)\in\mathcal{D}} \left[\frac{\#(c,o)}{|\mathcal{D}|} \right] \cdot \log \left[\frac{P_\vartheta(o|c) \cdot \#(c)}{|\mathcal{D}|} \right] \right] =$$

$$= \arg\min_\vartheta [+ H(P,Q)] = \arg\min_\vartheta [H(P) + KL(P\|Q)] =$$

($H(P,Q)$: cross-entropy! $H(P)$: entropy! $KL(P\|Q)$: KL-divergence!)

$$= \arg\min_\vartheta [KL[P\|Q]] = ?$$

⇓

- As $KL \geq 0 \Rightarrow$ setting: $q = p$ minimizes the objective. $\Rightarrow$

$$p \equiv q \text{ iff } \forall (c,o) \quad \frac{\#(c,o)}{|D|} = \frac{p_\theta(o|c) \cdot \#(c)}{|D|}$$

$$\text{iff } \frac{\#(c,o)}{\#(c)} = p_\theta(o|c) \text{ iff } \boxed{p_\theta(o|c) = \frac{\#(c,o)}{\max_{o'} \#(c,o')}}$$

Q.e.d.

(b)

Let $V = \{'a', 'b', 'c', 'd'\}$ be the vocabulary, and $C = \{"a\ a"; "b\ b"; "c\ c"; "d\ d"\}$ be the corpus.

→ If we use a window size of $(M=1)$, it is clear that each word will appear in D - both as the center words and as an outside-word. Also, no word from V occurs next to a different word from V in the set of tuples D. [Except maybe for the "blank" word],

⇒ Therefore, $P_1(O|C) = \frac{\#(C,O)}{\sum_{O'} (C,O')}$ is a well defined probability for all words in the vocabulary, yet, it results, (for example)

$$P_1('a'=O \mid C='b') = \frac{\#('a','b')}{1} = \frac{0}{1} = \boxed{0}$$

Whereas, $P_2(O|C) = \frac{e^{u_O^T \cdot v_C}}{\sum_{w \in V} e^{u_w^T \cdot v_C}} > 0$

⇒ Specifically: $P_1('a'=O \mid C='b') = 0 \neq P_2(O='a' \mid C='b')$

$\Rightarrow$ In this instance - the two probabilities differ $\rightarrow$ $P_2 \not\equiv P_1 \Rightarrow P_2$ cannot solve the optimization problem [AS shown in (a)].

q.e.d.

(5) Paraphrase-detection:

(a)
- X_1, X_2 := the input pair of sentences
- $\underline{X}_1, \underline{X}_2 \triangleq$ a vector representation for the pair of sentences obtained using some neural network, where $\underline{X}_i \in \mathbb{R}^d$.

$$P(\text{pair is a paraphrase} \mid X_1, X_2) = \sigma(\text{relu}^T(\underline{X}_1) \cdot \text{relu}(\underline{X}_2))$$

$\Rightarrow$ note that $\forall \underline{X}, \underline{Y} \in \mathbb{R}^d : \text{relu}^T(\underline{X}) \cdot \text{relu}(\underline{Y}) \geq 0$

$\Rightarrow \sigma(\text{relu}^T(\underline{X}_1) \cdot \text{relu}(\underline{X}_2)) = 1$

due to the behavior of the sigmoid function!

$\Rightarrow$ therefore, prediction will always be positive

$\Rightarrow$ accuracy of $\frac{1}{4}$.

(4)

↓

(1)

(b) An easy fix would be - to use

a different activation for example

the composition: (sigmoid)∘(tanh[inner-product])

It is also possible to consider - introducing

a biasing - parameter $\varepsilon > 0$ s.t.

$$\text{Prediction} = \text{sigmoid}[\text{relu}^T(\underline{x}_1)\cdot\text{relu}(\underline{x}_2) - \varepsilon] \Rightarrow$$

⇒ This would allow us to keep the benefits

of using sparse vectors while enabling

us classifying negative examples.

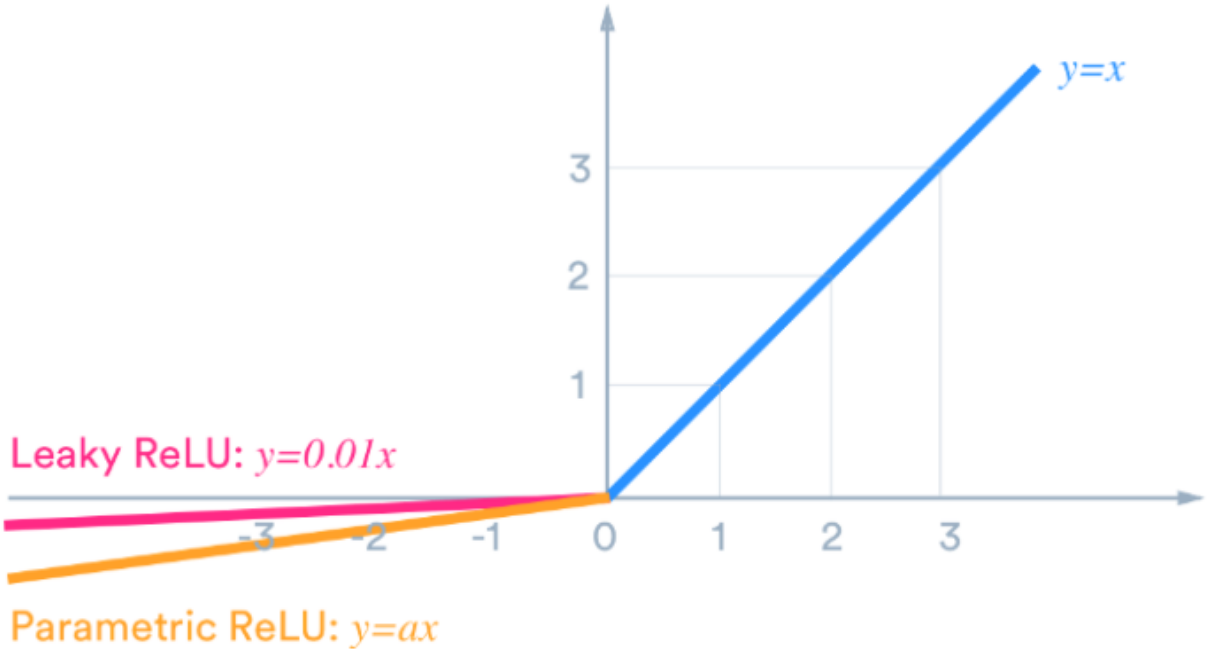
[The parameter ε should be learned as-well].

- The simplest solution of all is to

remove the inner-activation altogether:

⇒ (sigmoid) ∘ [inner-Product].

Using tanh might result in squeezing our values toward the edges, that in turn will cause the PDF to be very decisive – values will be very close to either 0 or 1. And alternative solution, will be replacing ReLU with leaky ReLU, which holds representation for negative values as well, thus will solve the problem presented, while keeping the values spread as possible.



Leaky ReLU