NLP HW1

Gefen Keinan Amit Mualem May Shushan

March 2019

1 Basics

a. The Softmax function is: $\sigma(x)_j = \frac{e^{z_j}}{\sum_{K=1}^{k} e^{z_k}}$
We need to prove foreach dimension :
$\sigma(x)_j = \sigma(x+c)j$
$\sigma(x+c)_j = \frac{e^{z_j+c}}{\sum_{K=1}^{k} e^{z_k+c}} = \frac{e^c e^{z_j}}{e^c \sum_{K=1}^{k} e^{z_k}} = \frac{e^{z_j}}{\sum_{K=1}^{k} e^{z_k}} = \sigma(x)_j$

c.

$\frac{d}{dx}\sigma(x) = \frac{d}{dx}[\frac{1}{1+e^{-x}}] = -(1+e^{-x})^{-2}(-e^{-x}) = \frac{1}{(1+e^{-x})}\frac{e^{-x}}{(1+e^{-x})} = \frac{1}{(1+e^{-x})}(1-\frac{1}{(1+e^{-x})}) =$
$\sigma(x)(1-\sigma(x))$

2 Word2Vec

a.
$\hat{y_o} = p(o|c) = \frac{exp(u_o^T v_c)}{\sum_{w=1}^{W} exp(u_w^T v_c)}$

the Cross Entropy defined as:

$CE(y,\hat{y}) = -\sum_i y_i * log(\hat{y_i}) =$

$$-\sum_i^{|W|} y_i * log(\frac{exp(u_i^T v_c)}{\sum_{w=1}^{W} exp(u_w^T v_c)}) = -\sum_i^{|W|} y_i[u_i^T v_c - log(\sum_{w=1}^{W} exp(u_w^T v_c))]$$

we derive the gradients with respect to v_c, also its one hot vector with only one index 1 the rest are zeroes:

$$\frac{d}{d_{v_c}}CE = -[u_i - \frac{\sum_{w=1}^{W} exp(u_w^T v_c)u_w}{\sum_{x=1}^{W} exp(u_x^T v_c)}] = \sum_{w=1}^{W}(\frac{exp(u_x^T v_c)}{\sum_{x=1}^{W} exp(u_x^T v_c)}u_w) - u_i = \sum_{w=1}^{W}(\hat{y_w}u_w) - u_i = U[\hat{y} - y]$$

b.

$\frac{d}{d_U}CE = c_c(\frac{exp(u_o^T v_c)}{\sum_{j=1}^{|V|} exp(u_j^T v_c)} - \delta_j^o) = v_c(\hat{y} - y)^T$

c.

$J(u_o, v_c, U) = -\log(\sigma(u_o^\top v_c)) - \sum_{k=1}^{K} \log(\sigma(-u_k^\top v_c))$

we already proved the gradient of the sigmoid function in 1.c

$\frac{d}{d_{v_c}J} = -\frac{1}{\sigma(u_o^T v_c)}\sigma(u_o^T v_c)(1 - \sigma(u_o^T v_c))u_o - \sum_{k=1}^{K} \frac{1}{\sigma}\sigma(1 - \sigma(-u_k^T v_c)) - u_k = (\sigma(u_o^T v_c) - 1)u_o - \sum_{k=1}^{K}(\sigma(-u_k^T v_c) - 1)u_k$

$\frac{d}{d_{u_o}J} = -\frac{1}{\sigma}\sigma(1 - \sigma(u_o^T v_c))v_c = (\sigma(u_o^T v_c) - 1)v_c$

$\frac{d}{d_{u_k}J} = \frac{1}{\sigma}\sigma(1 - \sigma(-u_k^T v_c))v_c = (-\sigma(-u_k^T v_c) + 1)v_c$

To k in 1...K

d.

We already know :

$$J_{\texttt{skip-gram}} = \sum_{-m \le j \le m, j \ne 0} F(w_{c+j}, v_c) \quad (1)$$

and we know the derivative : $\frac{\partial F(w_i, \hat{v})}{\partial U}$ or $\frac{\partial F(w_i, \hat{v})}{\partial \hat{v}}$
and

$$\frac{\partial J_{\texttt{skip-gram(word}_{c-m....c+m})}}{\partial v_c} = \sum_{-m \le j \le m, j \ne 0} \frac{\partial F(w_{c+j}, v_c)}{\partial v_c} \quad (2)$$

$$\frac{\partial J_{\texttt{skip-gram(word}_{c-m....c+m})}}{\partial U} = \sum_{-m \le j \le m, j \ne 0} \frac{\partial F(w_{c+j}, v_c)}{\partial U} \quad (3)$$

$$\frac{\partial J_{\texttt{skip-gram(word}_{c-m....c+m})}}{\partial v_j} = 0, j \ne c \quad (4)$$

h.

The result for KNN :

Words related to ”the”: [’the’, ’if’, ’that’, ’comedythriller’, ’or’, ’a’, ’.’, ’is’, ’derek’, ’bolt’, ’decide’]

Words related to ”unique”: [’unique’, ’1979’, ’puns’, ’ba’, ’realized’, ’succumb’, ’chabrolian’, ’dares’, ’regardless’, ’imaginative’, ’lunar’]

Words related to ”superb”: [’superb’, ’mine’, ’gold’, ’zingers’, ’moppets’, ’roussillon’, ’best’, ’ghoulish’, ’industry’, ’pool’, ’transporter’]

Words related to ”comedy”: [’comedy’, ’sensation’, ’observation’, ’fast’, ’first-timer’, ’singing’, ’cleaving’, ’longest’, ’cute’, ’mature’, ’often-funny’]

Words related to ”surprisingly”: [’surprisingly’, ’either’, ’hundred’, ’20-car’, ’philandering’, ’unusually’, ’protective’, ’dogs’, ’bollywood’, ’thinking’, ’soderbergh’]

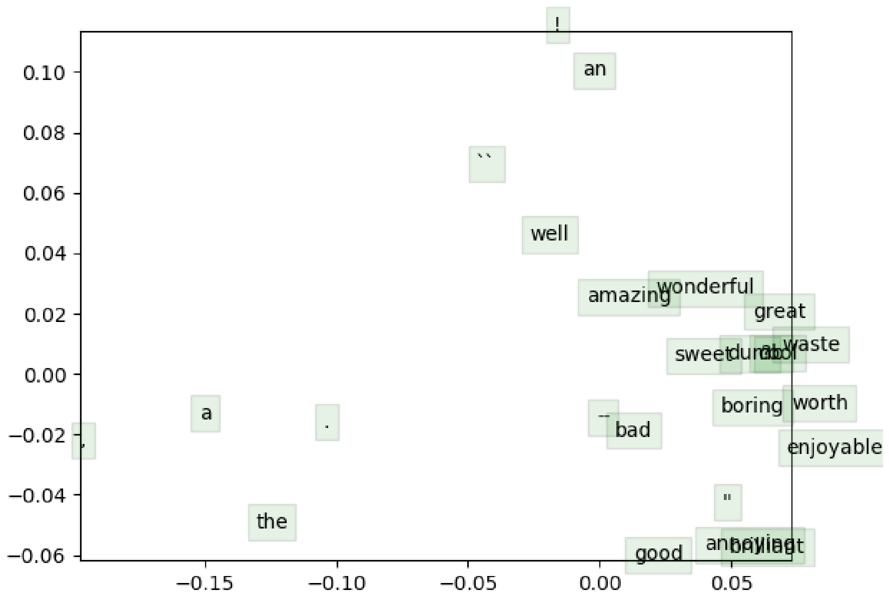

Figure 1: word vectors.

3 Analogies

a. We tried to find polysemous word such that both meaning will appear under the 10-top most similar words (according to cosine similarity). First we tried the examples words and then some other words we found in Wikipedia : "bank", "leaves", "Mole"... I think those words wasnt good for the mission because the data set has a lot of examples to one meaning of the word, for example : "bank" has only words matching the meaning of a financial institution, so i could guess the data has a lot of examples around a financial institution.

The word we found is : "mouse" it has two meanings:

1. a small rodent characteristically having a pointed snout, small rounded ears, a body-length scaly tail and a high breeding rate.

2. a hand-held pointing device that detects two-dimensional motion relative to a surface, which moves the cursor in accordance with its move.

we can see here words similar to both meanings:

('Logitech$_{M}X_{R}evolution', 0.6175230741500854),$
$('Razer_{M}amba', 0.5994571447372437),$
$('mice', 0.5896884799003601), -similar\, to\, option\, 1$
$('cordless_{l}aser', 0.5652030110359192),$
$('VX_{N}ano', 0.5619357824325562),$
$('Logitech_{M}X', 0.5604779720306396),$
$('keyboard_{a}rrow_{k}eys', 0.5545550584793091), -similar\, to\, option\, 2$
$('Logitech_{G}9x', 0.5538491606712341),$
$('NOTE_{T}O_{R}EADERS_{H}overing', 0.5520266890525818),$
$('Razer_{A}byssus', 0.5489388108253479)$

b.
The words we found are:
w1 = "female"
w2 = "women"
w3 = "male"

Synonyms female, women have cosine distance: 0.44437193870544434
Antonyms female, male have cosine distance: 0.15946662425994873

I think that female and male has smaller distance then female and women because word2vec is learning how "close" words are by the probability that word will appear next to w1 and w2, so i can guess there are more common words appear next to female and male then common words appear next to female and women.

c.

Analogies:
1.
boy:man girl:women positive=['boy', 'man'], negative=['girl']

The results:

```
('teenager', 0.5787034034729004),
('woman', 0.5065137147903442),
('youngster', 0.49353092908859253),
```

2.
bad:smart good:stupid
(positive=['bad', 'smart'], negative=['good'])

The results:

```
('dumb', 0.6246984601020813),
('stupid', 0.5432119965553284),
('boneheaded', 0.46688923239707947),
```

"boy" is the correct answer because real and barca are football groups that are

d.

'bad':'sad' 'good':'happy'

(positive=['bad', 'sad'], negative=['good']))

The results:

```
 ('saddening', 0.6041171550750732),
('distressing', 0.5857565402984619),
('unfortunate', 0.5792170763015747),
```

I wish to see "happy"