Mingxiang Chen

CS 224N

Assignment 1

Prob 1.a.

For each term in the output.

$$\text{softmax}(x+c)_i = \frac{e^{x_i+c}}{\sum_j e^{x_j+c}} = \frac{e^c e^{x_i}}{e^c \sum_j e^{x_j}} = \frac{e^{x_i}}{\sum_j e^{x_j}} = \text{softmax}(x)_i.$$

Prob 2.a.

$$\frac{d}{dx}\left(\frac{1}{1+e^{-x}}\right) = -(1+e^{-x})^{-2}\cdot e^{-x}(-1) = \frac{e^{-x}}{(1+e^{-x})^2}$$

$$= \frac{e^{-x}}{1+e^{-x}}\cdot\frac{1}{1+e^{-x}} = (1-\sigma(x))\,\sigma(x).$$

Prob 2.b.

Since only the k^{th} element in y_i is 1, we can conclude that:

$$\text{gradience} = \frac{-d\sum_i y_i \log\left(\frac{e^{x_i}}{\sum_j e^j}\right)}{d\theta} = \frac{d(-\log(\hat{y}_k))}{d\theta}.$$

for each term x_i in θ:

$$\text{grad}_i = \frac{-d\log\left(\frac{e^{x_k}}{e^{x_k}+\sum_{j\neq k} e^{x_j}}\right)}{dx_k}$$

$$= -\frac{e^{x_k}+\sum_{j\neq k} x_j}{e^{x_k}}\cdot\frac{e^{x_k}\sum_{j\neq k} e^{x_j}}{(e^{x_k}+\sum_{j\neq k} e^{x_j})^2}$$

$$= \frac{\sum_{j\neq k} e^{x_j}}{\sum_j e^{x_j}} = \frac{e^{x_k}}{\sum_j e^{x_j}} - 1$$

$$\Rightarrow \text{gradiance} = \frac{e^{\theta_i}}{\sum_j e^{\theta_j}} - 1 = \hat{y} - y.$$

Prob 2.c

Using the chain rule, we get:

$$\frac{dJ}{dx} = \frac{dJ}{dh} \cdot \frac{dh}{d(xW_1+b_1)} \cdot \frac{d(xW_1+b_1)}{dx}$$

$$\frac{dJ}{dh} = \frac{dJ}{d(hW_2+b_2)} \cdot \frac{d(hW_2+b_2)}{dh}$$

According to prob 2.b. $\frac{dJ}{d(hW_2+b_2)} = \hat{y} - y$.

$$\Rightarrow \frac{dJ}{dh} = (\hat{y}-y) \cdot W_2^T \quad \text{--------} \quad ①.$$

According to prob 2.a. $\frac{dh}{d(xW_1+b_1)} = \sigma(1-\sigma) = h(1-h)$. ---- ②.

$$\Rightarrow \frac{\partial J}{\partial x} = (\hat{y}-y) \cdot W_2^T \cdot h(1-h) \cdot W_1^T$$

Prob 2.d.

$$\# = \theta_1 + \theta_2 + b_1 + b_2$$

$$= D_x H + D_y H + H + D_y.$$

Prob 3.a.

$$\frac{\partial J}{\partial v_c} = \frac{\partial CE(y,\hat{y})}{\partial(\overset{}{\cancel{u_o^T v_c}}\ \theta)} \cdot \frac{\partial(\overset{\theta}{\cancel{u_o^T v_c}})}{\partial v_c} = (\hat{y}-y)\,u = \sum_{i=1}^{w} p(o|c)\,u_i - u_o .$$

where θ is the input of softmax.

Prob 3.b.

$$\frac{\partial J}{\partial u} = \frac{\partial J}{\partial(\theta)} \cdot \frac{\partial(\theta)}{\partial u} = (\hat{y}-y)\,v_c$$

for u_o: $\dfrac{\partial J}{\partial u} = (p(o|c)-1)\,v_c$

for other u_w: $\dfrac{\partial J}{\partial u} = p(w|c)\,v_c$

Prob 3.c

$$\frac{\partial J}{\partial v_c} = \frac{\partial\left(-\log(\sigma(u_o^T v_c))\right)}{\partial v_c} + \frac{\partial\left(-\sum_{k=1}^{K}\log\left(\sigma(-u_k^T v_c)\right)\right)}{\partial v_c}.$$

$$= -\frac{1}{\sigma}\cdot\sigma(1-\sigma)\cdot u_o - \sum_{k=1}^{K}\frac{1}{\sigma}(\sigma-1)\sigma\cdot u_k\cdot(-1)$$

$$= (\sigma(u_o^T v_c)-1)\,u_o + \sum_{k=1}^{K} -(\sigma(-u_k^T v_c)-1)\cdot u_k .$$

$$\frac{\partial J}{\partial u} = -\frac{1}{\sigma}\cdot\sigma(1-\sigma)\cdot v_c + \sum_{k=1}^{K}(\sigma(-u_k^T v_c)-1)\cdot v_c(-1)$$

$$= (\sigma(u_o^T v_c)-1)\,v_c + \sum_{k=1}^{K} -(\sigma(-u_k^T v_c)-1)\cdot v_c .$$

It will be more efficient because here we are only consider $(K+1)$ terms rather than W terms.

3.d For skip-gram:

$$\frac{\partial J}{\partial v_c} = \sum_{-m \le j \le m,\, j \ne 0} \frac{\partial F(w_{c+j}, v_c)}{\partial v_c} = \sum_{-m \le j \le m,\, j \ne 0} \frac{\partial CE(w_{c+j}, v_c)}{\partial v_c}.$$

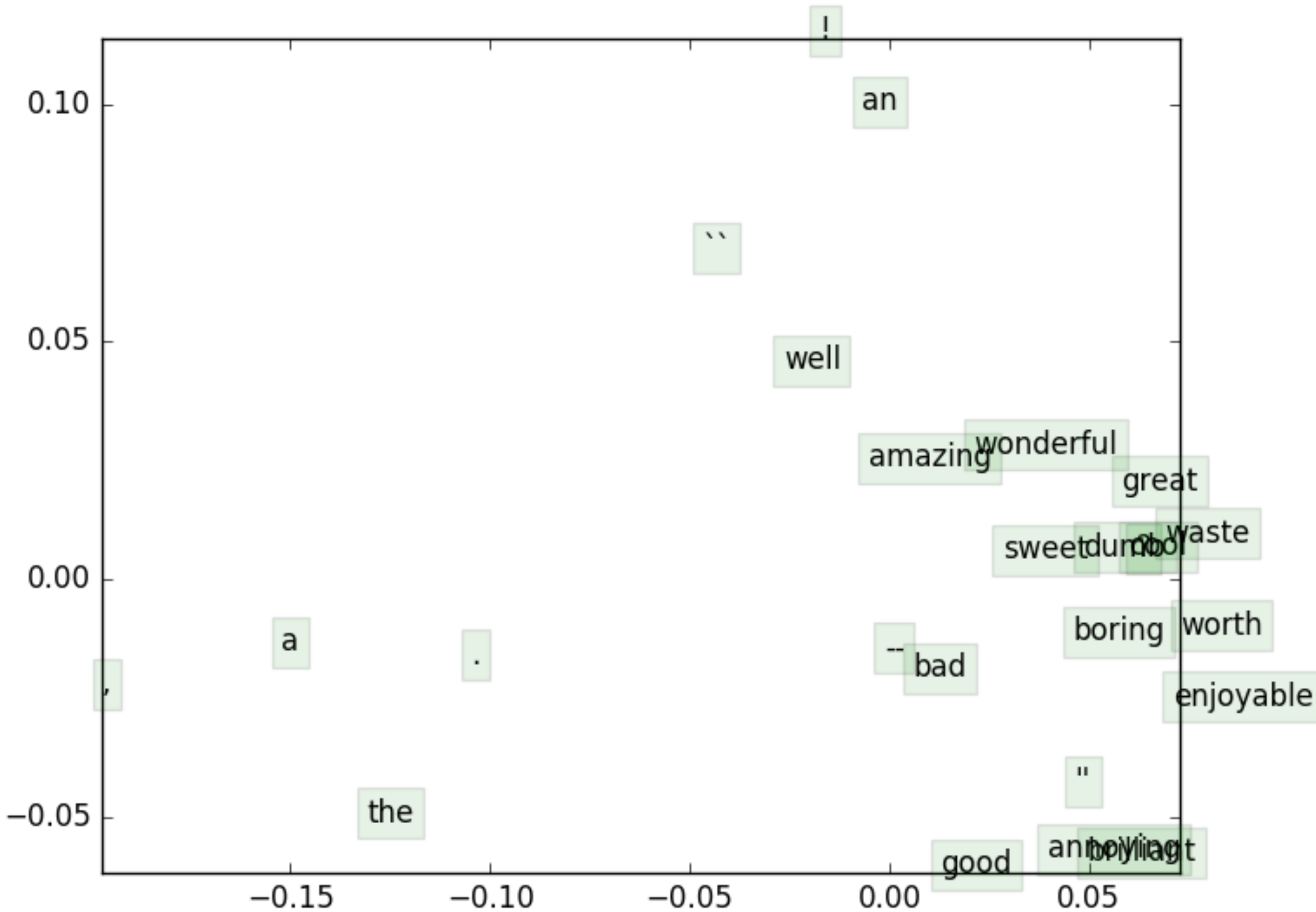
$$\frac{\partial J}{\partial u_k} = \sum_{-m \le j \le m,\, j \ne 0} \frac{\partial F(w_{c+j}, v_c)}{\partial u_k} = \sum_{-m \le j \le m,\, j \ne 0} \frac{\partial CE}{\partial u_k}.$$

For CBOW

$$\frac{\partial J}{\partial v_k} = \sum_{\substack{-m \le j \le m,\, j \ne 0 \\ \text{when } v_{c+j} = v_k}} \frac{\partial F}{\partial \hat{v}} \quad \left(\hat{v} = \sum_{\substack{-m \le j \le m \\ j \ne 0}} v_{c+j} \right).$$

$$\frac{\partial J}{\partial u_k} = \frac{\partial F(w, \hat{v})}{\partial u_k}.$$

Prob 3.g



Most of the words are on the bottom right corner. Some very useful words and notations such as “a”, “the” is rather far away to the majority.

Prob 4.b

Use of regularization is to avoid overfitting.

Prob 4.c

```
def chooseBestModel(results):
    """Choose the best model based on parameter tuning on the dev set

    Arguments:
    results -- A list of python dictionaries of the following format:
        {
            "reg": regularization,
            "clf": classifier,
            "train": trainAccuracy,
            "dev": devAccuracy,
            "test": testAccuracy
        }

    Returns:
    Your chosen result dictionary.
    """
    bestResult = results[0]

    ### YOUR CODE HERE
    for i in range(len(results)):
        currDict = results[i]
        if currDict["test"] > bestResult["test"]:
            bestResult = currDict
    ### END YOUR CODE

    return bestResult
```

Prob 4.d

The result based on my trainin:

Reg		Train	Dev	Test
0.00E+00	31.016	32.516	30.407	
0.00E+00	31.016	32.516	30.407	
1.00E+00	28.897	29.609	27.149	
1.00E+01	27.247	25.522	23.077	
1.00E+02	27.247	25.522	23.032	
1.00E+03	27.247	25.522	23.032	
1.00E+04	27.247	25.522	23.032	
1.00E+05	27.247	25.522	23.032	
1.00E+06	27.247	25.522	23.032	
1.00E+07	27.247	25.522	23.032	
1.00E+08	27.247	25.522	23.032	
1.00E+09	27.247	25.522	23.032	

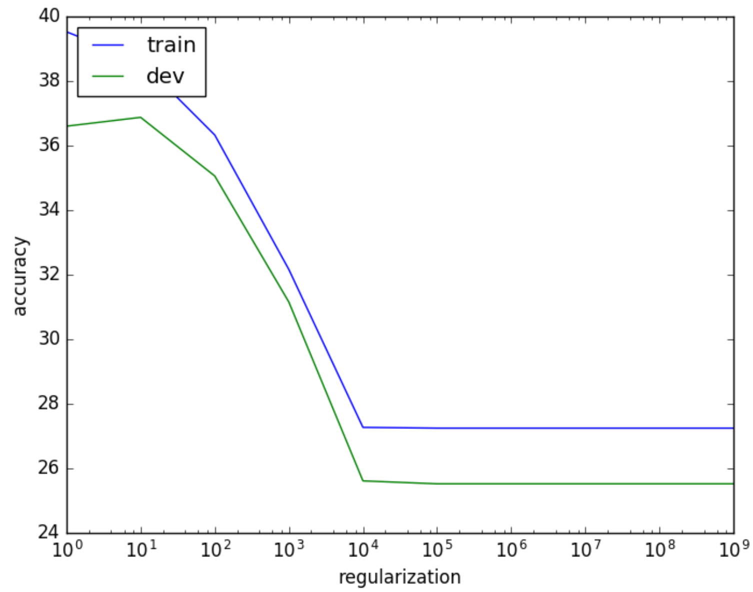
The result based on pre-trained:

0.00E+00	39.923	36.421	37.059
0.00E+00	39.923	36.421	37.059
1.00E+00	39.525	36.603	37.330
1.00E+01	38.624	36.876	37.692
1.00E+02	36.330	35.059	35.701
1.00E+03	32.163	31.153	30.588
1.00E+04	27.271	25.613	23.122

1.00E+05	27.247	25.522	23.032
1.00E+06	27.247	25.522	23.032
1.00E+07	27.247	25.522	23.032
1.00E+08	27.247	25.522	23.032
1.00E+09	27.247	25.522	23.032

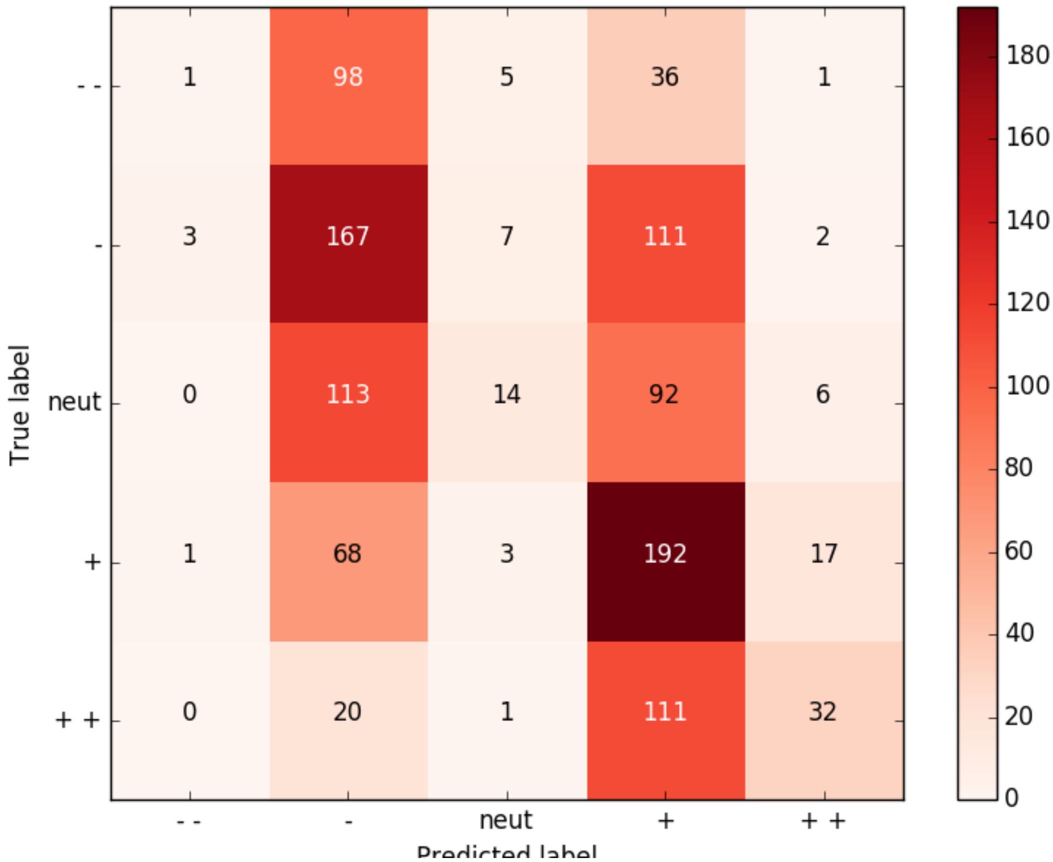
In most cases the training accuracy is better than dev accuracy, and the dev accuracy is better than test accuracy. The pre-trained result is better than mine because it is using GloVe vectors (which may be better than word2vec) with a bigger dataset (Wikipedia data). The pre-trained data may also trained for a longer period.

Prob 4.e



The training accuracy is always decreasing when the regularization factor is increasing until it reached a platform. However, the dev accuracy increased a little bit at first, but then also decreased.

Prob 4.f



From the diagram, we can see that the algorithm tends to giving "+" or "—". Overall it gives a pretty acceptable result on these two labels, but not very good on "— — ", "++", and neutual.

Prob 4.g

Example 1.

3 1 whether you like rap music or loathe it , you ca n't deny either the tragic loss of two young men in the prime of their talent or the power of this movie .

Here may be the program mistakenly only gives a large weight on strong words like "deny", "tragic", but missed words like "talent".

Example 2.

4 1 the movie is n't just hilarious : it 's witty and inventive , too , and in hindsight , it is n't even all that dumb .

Although, words like "dumb" is a strong signal of bad sentiment, but the algorithm may missed considering "isn't".

Example 3.

0 1 it 's like watching a nightmare made flesh .

The length of the sentence could also be taken into account.