CS 224N **Name: Xin Li**
Homework #1 **SUNet ID: 06121602**

Answer to Question 1: Softmax

(1.a)

$$\begin{aligned} softmax(x+c) &= \frac{e^{x_i+c}}{\sum_j e^{x_j+c}} \\ &= \frac{e^c \cdot e^{x_i}}{e^c \cdot \sum_j e^{x_j}} \\ &= \frac{e^{x_i}}{\sum_j e^{x_j}} \\ &= softmax(x) \end{aligned}$$

(1.b) q1_softmax.py is submitted.

Answer to Question 2: Neural Network Basics

(2.a)

$$\begin{aligned}
\sigma'(\boldsymbol{x}) &= -(1+e^{-\boldsymbol{x}})^{-2}(-e^{-\boldsymbol{x}}) \\
&= \frac{e^{-\boldsymbol{x}}}{(1+e^{-\boldsymbol{x}})^2} \\
&= -\frac{1}{(1+e^{-\boldsymbol{x}})^2} + \frac{1}{1+e^{-\boldsymbol{x}}} \\
&= \sigma(\boldsymbol{x}) - \sigma(\boldsymbol{x})^2
\end{aligned}$$

(2.b)

$$\begin{aligned}
\frac{d(CE(\boldsymbol{y}, \hat{\boldsymbol{y}}))}{d\theta} &= \frac{d(-\sum\limits_i y_i log(\hat{y}_i))}{d\theta} \\
&= \frac{d(-\sum\limits_i y_i log(\frac{e^{\theta_i}}{\sum\limits_j e^{\theta_j}}))}{d\theta} \\
&= \frac{d(-\theta_k + log \sum\limits_i e^{\theta_i})}{d\theta} \\
&= -\boldsymbol{e_k} + \frac{d(log \sum\limits_i e^{\theta_i})}{d\theta} \\
&= -\boldsymbol{y} + \frac{1}{\sum\limits_i e^{\theta_i}} \cdot \frac{d \sum\limits_j e^{\theta_j}}{d\theta} \\
&= -\boldsymbol{y} + \frac{e^{\theta_i}}{\sum\limits_j e^{\theta_j}} \\
&= -\boldsymbol{y} + \hat{\boldsymbol{y}}
\end{aligned}$$

(2.c)

$$
\begin{aligned}
\frac{\partial \boldsymbol{h}}{\partial \boldsymbol{x}} &= \frac{\partial(sigmoid(\boldsymbol{x}\boldsymbol{W_1} + \boldsymbol{b_1}))}{\partial \boldsymbol{x}}, \qquad Let \quad \theta_1 = \boldsymbol{x}\boldsymbol{W_1} + \boldsymbol{b_1} \\
&= \frac{\partial(sigmoid(\theta_1))}{\partial \theta_1} \cdot \frac{\partial \theta_1}{\partial \boldsymbol{x}} \\
&= \sigma'(\theta_1) \cdot \frac{\partial(\boldsymbol{x}\boldsymbol{W_1} + \boldsymbol{b_1})}{\partial \boldsymbol{x}} \\
&= \sigma'(\boldsymbol{x}\boldsymbol{W_1} + \boldsymbol{b_1}) \cdot \boldsymbol{W_1^T} \\
\frac{d(CE(\boldsymbol{y}, \hat{\boldsymbol{y}}))}{d\boldsymbol{x}} &= \frac{d(CE(\boldsymbol{y}, \hat{\boldsymbol{y}}))}{d\theta_2} \cdot \frac{d\theta_2}{\boldsymbol{x}}, \qquad Let \quad \theta_2 = \boldsymbol{h}\boldsymbol{W_2} + \boldsymbol{b_2} \\
&= (\hat{\boldsymbol{y}} - \boldsymbol{y}) \cdot \frac{d\theta_2}{\boldsymbol{x}} \\
&= (\hat{\boldsymbol{y}} - \boldsymbol{y}) \cdot \frac{d(\boldsymbol{h}\boldsymbol{W_2} + \boldsymbol{b_2})}{\boldsymbol{x}} \\
&= (\hat{\boldsymbol{y}} - \boldsymbol{y}) \cdot (\boldsymbol{W_2^T} \cdot \frac{\partial \boldsymbol{h}}{\partial \boldsymbol{x}} + \boldsymbol{h}\frac{\partial \boldsymbol{W_2}}{\partial \boldsymbol{x}} + \frac{\partial \boldsymbol{b_2}}{\partial \boldsymbol{x}}) \\
&= (\hat{\boldsymbol{y}} - \boldsymbol{y}) \cdot (\boldsymbol{W_2^T} \cdot \frac{\partial \boldsymbol{h}}{\partial \boldsymbol{x}}) \\
&= (\hat{\boldsymbol{y}} - \boldsymbol{y}) \cdot \boldsymbol{W_2^T} \cdot \sigma'(\boldsymbol{x}\boldsymbol{W_1} + \boldsymbol{b_1}) \cdot \boldsymbol{W_1^T}
\end{aligned}
$$

(2.d)
from the inputs x to the hidden layer: $(D_x + 1)H$
from the hidden layer to the outputs y: $(H + 1)D_y$
$\therefore$ there are $((D_x + D_y + 1)H + D_y)$ parameters.

(2.e) q2_sidmoid.py is submitted.

(2.f) q2_gradcheck.py is submitted.

(2.g) q2_neural.py is submitted.

Answer to Question 3: word2vec

(3.a) $U = [\boldsymbol{u_1}, \boldsymbol{u_2}, \cdots, \boldsymbol{u_w}]$, $\boldsymbol{y}$ and $\hat{\boldsymbol{y}}$ are both column vectors

$$
\begin{aligned}
Let: f_o &= \boldsymbol{u_o}^T \boldsymbol{v_c}, f_w = \boldsymbol{u_w}^T \boldsymbol{v_c} \\
p(\boldsymbol{o}|\boldsymbol{c}) &= \frac{exp(\boldsymbol{u_o}^T \boldsymbol{v_c})}{\sum\limits_{w=1}^{W} exp(\boldsymbol{u_w}^T \boldsymbol{v_c})} \\
&= \frac{exp(f_o)}{\sum\limits_{w=1}^{W} exp(f_w)} \\
&= softmax(f_o) \\
\frac{\partial J}{\partial \boldsymbol{v_c}} &= \sum_{w=1}^{W} \frac{\partial(-log(softmax(f_o)))}{\partial \boldsymbol{v_c}} \\
&= \sum_{w=1}^{W} \frac{\partial(-log(softmax(f_o)))}{\partial f_w} \cdot \frac{\partial f_w}{\partial \boldsymbol{v_c}} \\
&= \sum_{w=1}^{W} \delta_w \cdot \frac{\partial \boldsymbol{u_w}^T \boldsymbol{v_c}}{\partial \boldsymbol{v_c}} \\
&= \sum_{w=1}^{W} \delta_w \boldsymbol{u_w} \\
&= U^T \delta \\
&= U^T (\hat{\boldsymbol{y}} - \boldsymbol{y})
\end{aligned}
$$

(3.b) Similar to (3.a), we can get:

$$
\begin{aligned}
\frac{\partial J}{\partial \boldsymbol{u_w}} &= \frac{\partial(-log(softmax(f_o)))}{\partial \boldsymbol{u_w}} \\
&= \frac{\partial(-log(softmax(f_o)))}{\partial f_w} \cdot \frac{\partial f_w}{\partial \boldsymbol{u_w}} \\
&= \delta_w \cdot \frac{\partial \boldsymbol{u_w}^T \boldsymbol{v_c}}{\partial \boldsymbol{u_w}} \\
&= \delta_w \boldsymbol{v_c} \\
\frac{\partial J}{\partial U} &= \boldsymbol{v_c} \delta^T \\
&= \boldsymbol{v_c} (\hat{\boldsymbol{y}} - \boldsymbol{y})^T
\end{aligned}
$$

(3.c)

$$\frac{\partial J_{neg-sample}(\boldsymbol{o}, \boldsymbol{v_c}, U)}{\partial \boldsymbol{v_c}} = \frac{\partial(-log(\sigma(\boldsymbol{u_o}^T \boldsymbol{v_c})) - \sum_{k=1}^{K} log(\sigma(-\boldsymbol{u_k}^T \boldsymbol{v_c})))}{\partial \boldsymbol{v_c}}$$
$$= -\frac{\sigma'(\boldsymbol{u_o}^T \boldsymbol{v_c})}{\sigma(\boldsymbol{u_o}^T \boldsymbol{v_c})}\boldsymbol{u_o} - \sum_{k=1}^{K} \frac{\sigma'(-\boldsymbol{u_k}^T \boldsymbol{v_c})}{\sigma(-\boldsymbol{u_k}^T \boldsymbol{v_c})}(-\boldsymbol{u_k})$$
$$= (-1 + \sigma(\boldsymbol{u_o}^T \boldsymbol{v_c}))\boldsymbol{u_o} + \sum_{k=1}^{K}(1 - \sigma(-\boldsymbol{u_k}^T \boldsymbol{v_c}))\boldsymbol{u_k}$$

$$\frac{\partial J_{neg-sample}(\boldsymbol{o}, \boldsymbol{v_c}, U)}{\partial \boldsymbol{u_o}} = \frac{\partial(-log(\sigma(\boldsymbol{u_o}^T \boldsymbol{v_c})) - \sum_{k=1}^{K} log(\sigma(-\boldsymbol{u_k}^T \boldsymbol{v_c})))}{\partial \boldsymbol{u_o}}$$
$$= (-1 + \sigma(\boldsymbol{u_o}^T \boldsymbol{v_c}))\boldsymbol{v_c}$$

$$\frac{\partial J_{neg-sample}(\boldsymbol{o}, \boldsymbol{v_c}, U)}{\partial \boldsymbol{u_k}} = \frac{\partial(-log(\sigma(\boldsymbol{u_o}^T \boldsymbol{v_c})) - \sum_{k=1}^{K} log(\sigma(-\boldsymbol{u_k}^T \boldsymbol{v_c})))}{\partial \boldsymbol{u_k}}$$
$$= (1 - \sigma(-\boldsymbol{u_k}^T \boldsymbol{v_c}))\boldsymbol{v_c}$$

This cost function sums over K elements, but the softmax-CE loss function sums over W elements.

(3.d)
For skip-gram:

$$\frac{\partial J_{skip-gram}(word_{c-m,\ldots,c+m})}{\partial \boldsymbol{v_k}} = \frac{\sum_{-m \le j \le m, m \ne 0} \partial F(\boldsymbol{w_{c+j}}, \boldsymbol{v_c})}{\partial \boldsymbol{v_k}}$$
$$= \begin{cases} \frac{\sum_{-m \le j \le m, m \ne 0} \partial F(\boldsymbol{w_{c+j}}, \boldsymbol{v_c})}{\partial \boldsymbol{v_c}}, & for \quad k = c \\ \boldsymbol{0}, & for \quad k \ne c \end{cases}$$

$$\frac{\partial J_{skip-gram}(word_{c-m,\ldots,c+m})}{\partial \boldsymbol{u_k}} = \frac{\sum_{-m \le j \le m, m \ne 0} \partial F(\boldsymbol{w_{c+j}}, \boldsymbol{v_c})}{\partial \boldsymbol{u_k}}$$

For the CBOW:

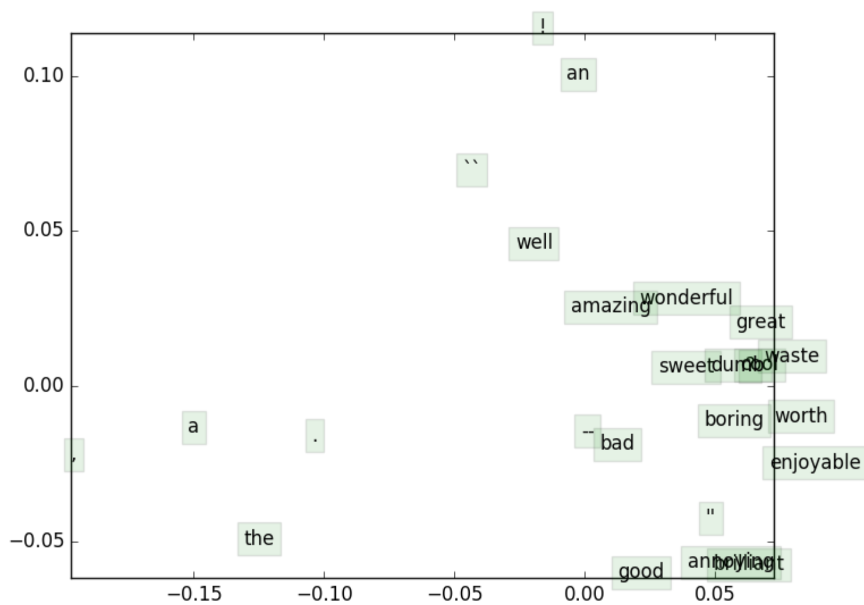
$$
\begin{aligned}
\frac{\partial J_{CBOW}(word_{c-m,\dots,c+m})}{\partial \boldsymbol{v_k}} &= \frac{\partial F(\boldsymbol{w_c}, \boldsymbol{\hat{v}})}{\partial \boldsymbol{v_k}} \\
&= \frac{\partial F(\boldsymbol{w_c}, \sum_{-m\le j\le m, j\ne 0} \boldsymbol{v_{c+j}})}{\partial \boldsymbol{v_k}} \\
&= \begin{cases} \frac{\partial F(\boldsymbol{w_c}, \boldsymbol{\hat{v}})}{\partial \boldsymbol{v_k}}, & for \quad k = c-m, \dots, c+m \\ \boldsymbol{0}, & otherwise \end{cases}
\end{aligned}
$$

$$
\frac{\partial J_{CBOW}(word_{c-m,\dots,c+m})}{\partial \boldsymbol{u_k}} = \frac{\partial F(\boldsymbol{w_c}, \boldsymbol{\hat{v}})}{\partial \boldsymbol{u_k}}
$$

(3.e) q3_word2vec.py is submitted.

(3.f) q3_sgd.py is submitted.

(3.g) q3_run.py is submitted.



Those words with similar meanings (like "wonderful" and "amazing" which are both adjectives) are located in a region. Punctuation and words like "the", "a", "an" which don't have any specific meanings are located separately, scattered all around the plane.

Answer to Question 4: Sentiment Analysis

(4.a) q4_sentiment.py is submitted.

(4.b) In order to avoid overfitting. Otherwise, the parameters can be extremely large to get high accuracy, which results in overfitting.

(4.c) q4_sentiment.py is submitted.
Best regularization value: 5.00E-04
Test accuracy (%): 37.013575

Code for chooseBestModel:

```python
def chooseBestModel(results):
    bestResult = None
    ### YOUR CODE HERE
    #print results
    maxDev = 0
    for r in results:
        if maxDev <= r['dev']:
            maxDev = r['dev']
            bestResult = r
    ### END YOUR CODE
    return bestResult
```

(4.d)
Results:

```
=== Recap ===
Reg             Train   Dev     Test
1.00E-05        39.923  36.512  37.014
3.00E-05        39.934  36.331  36.968
5.00E-05        39.923  36.331  36.923
7.00E-05        39.911  36.512  37.014
1.00E-04        39.946  36.421  36.968
3.00E-04        39.899  36.331  36.968
5.00E-04        39.876  36.512  37.014
7.00E-04        39.923  36.421  37.014
1.00E-03        39.958  36.421  36.968
3.00E-03        39.970  36.331  37.014
5.00E-03        39.958  36.240  37.104
7.00E-03        39.946  36.240  37.149
1.00E-02        39.923  36.240  37.195
3.00E-02        39.853  36.421  37.376
5.00E-02        39.923  36.421  37.511
7.00E-02        39.853  36.331  37.195

Best regularization value: 5.00E-04
Test accuracy (%): 37.013575
```

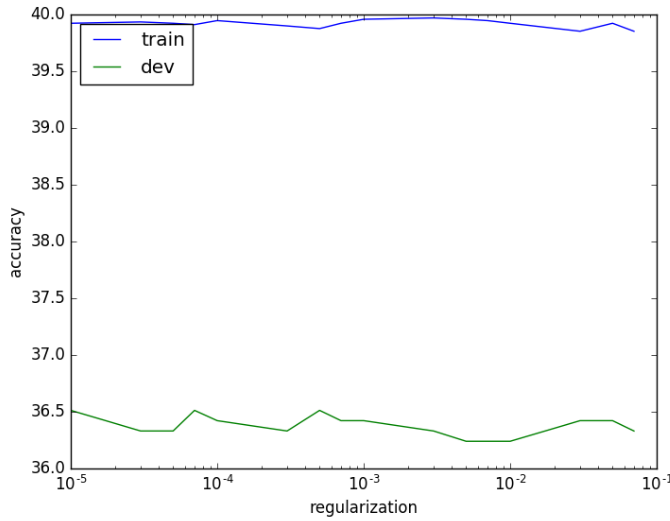
```
=== Recap ===
Reg             Train   Dev     Test
1.00E-05        31.051  32.607  30.452
3.00E-05        30.993  32.698  30.271
5.00E-05        31.016  32.516  30.498
7.00E-05        31.086  32.516  30.362
1.00E-04        31.039  32.516  30.452
3.00E-04        31.133  32.698  30.452
5.00E-04        31.145  32.607  30.362
7.00E-04        31.098  32.516  30.407
1.00E-03        31.121  32.607  30.362
3.00E-03        31.098  32.243  30.362
5.00E-03        30.911  32.425  30.452
7.00E-03        30.887  32.516  30.317
1.00E-02        30.946  32.334  29.910
3.00E-02        30.536  31.608  30.045
5.00E-02        30.337  31.880  30.136
7.00E-02        30.372  32.243  30.045

Best regularization value: 3.00E-04
Test accuracy (%): 30.452489
```

(i)For the pretrained vectors, the best train accuracy is 39.978%, the best dev accuracy is 36.512%, and the best test accuracy is 37.511%.

(ii)For my word vectors, the best train accuracy is 31.145%, the best dev accuracy is 32.698%, and the best test accuracy is 30.498%.

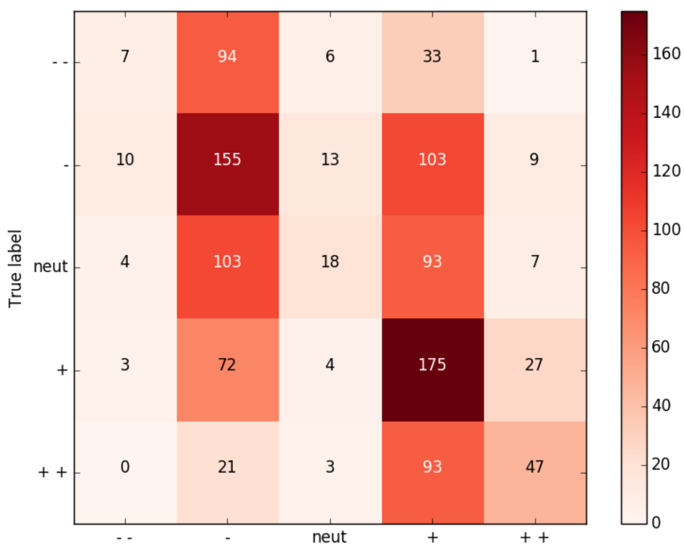
(iii)Reasons: The Wikipedia database is larger in size and it's more reliable. It's likely that the training set and the test set share similar features. GloVe also contributes to good performance.

(4.e)



Train accuracy is around 40% and development accuracy is around 36.5%. According to the plot, increasing or decreasing the regularization will not affect the accuracy much.

(4.f)



According to the diagonal elements, the prediction accuracy is relatively high for the positive(+) class and the negative(-) class. According to the off-diagonal elements, the model tends to label the most vectors as positive(+) or negative(-) instead of very negative(–), neutral or very positive(++).

(4.g)

True: 4 Predicted: 1

Text: an effectively creepy , fear-inducing -lrb- not fear-reducing -rrb- film from japanese director hideo nakata , who takes the superstitious curse on chain letters and actually applies it .

Reason: The text uses several negative adjectives to describe the director's talents, which can be confusing for the classifier. The classifier should be able to classify a polysemy according to the text.

True: 4 Predicted: 1

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

Reason: The text misspelled 'not' as 'n't'. The classifier should be able to find and correct minor misspellings.

True: 1 Predicted: 4

Text: directed in a paint-by-numbers manner .

Reason: The text uses a compound. The classifier should be able to separate and classify compounds.