Assignment 1

1 Softmax

(a) Proof:

$$\begin{aligned}\text{softmax}(\boldsymbol{x}+c)_i &= \frac{e^{x_i+c}}{\sum_j e^{x_j+c}} \\ &= \frac{e^{x_i}e^c}{\sum_j \left(e^{x_j}e^c\right)} \\ &= \frac{e^{x_i}}{\sum_j e^{x_j}} \\ &= \text{softmax}(\boldsymbol{x})_i\end{aligned}$$

(b) See file q1_softmax.py

2 Neural Network Basic

(a) We have:

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

(b) Because $\boldsymbol{y}$ is a one-hot label, we can assume that all elements in it is zero, except that the kth element is one. Therefore, by using the chain rule, we have:

$$\begin{aligned}\frac{\partial CE}{\partial \boldsymbol{\theta}_i} &= \frac{\partial CE}{\partial \hat{\boldsymbol{y}}}\frac{\partial \hat{\boldsymbol{y}}}{\partial \boldsymbol{\theta}_i} \\ &= -\frac{1}{\hat{\boldsymbol{y}}_k}\frac{\partial \hat{\boldsymbol{y}}}{\partial \boldsymbol{\theta}_i}\end{aligned}$$

For $i = k$, we have:

$$\frac{\partial \hat{\boldsymbol{y}}}{\partial \boldsymbol{\theta}_k} = -\frac{e^{x_k}\sum_{j\neq k} e^{x_j}}{(\sum_j e^{x_j})^2}$$

For $i \neq k$, we have:

$$\frac{\partial \hat{\boldsymbol{y}}}{\partial \boldsymbol{\theta}_i} = \frac{e^{x_k}e^{x_i}}{(\sum_j e^{x_j})^2}$$

Therefore, we have:

$$\frac{\partial CE}{\partial \boldsymbol{\theta}} = \hat{\boldsymbol{y}} - \boldsymbol{y}$$

(c) By using the result from the above question, we know that:

$$\frac{\partial CE}{\partial \boldsymbol{h}} = (\hat{\boldsymbol{y}} - \boldsymbol{y})\boldsymbol{W}_2{}^T$$

Let $\boldsymbol{O_1} = \boldsymbol{x}\boldsymbol{W}_1 + \boldsymbol{b}_1$, and by using the result of question (a), we have:

$$\frac{\partial CE}{\partial \boldsymbol{O}_1} = (\hat{\boldsymbol{y}} - \boldsymbol{y})\boldsymbol{W}_2{}^T * \boldsymbol{h}(1 - \boldsymbol{h})$$

Therefore, we have:

$$\frac{\partial CE}{\partial \boldsymbol{x}} = \Big((\hat{\boldsymbol{y}} - \boldsymbol{y})\boldsymbol{W}_2{}^T * \boldsymbol{h}(1 - \boldsymbol{h})\Big)W_1^T$$

(d) The dimension of W_1 is $D_x \times H$ and the dimension of W_2 is $H \times D_y$. Therefore, the total number of parameters are $D_x \times H + H \times D_y + H + D_y$

(e) See file q2_sigmoid.py

(f) See file q2_gradcheck.py

(g) See file q2_neural.py

3 word2vec

(a) Let $\boldsymbol{U} = [\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \ldots, \boldsymbol{\mu}_v]$ where each column represents an "output" word vector for some word in the vocabulary. The gradient of loss with respect to $\boldsymbol{v}_c$ is given by:

$$\begin{aligned}
\frac{\partial CE}{\partial \boldsymbol{v}_c} &= -\frac{1}{\hat{y}_o}\frac{\partial \hat{y}_o}{\partial \boldsymbol{v}_c} \\
&= \frac{1}{\hat{y}_o}\frac{\left(\sum_{w=1}^{V}\left((\boldsymbol{\mu}_w - \boldsymbol{\mu}_o)\exp(\boldsymbol{\mu}_w^T\boldsymbol{v}_c)\right)\right)\exp(\boldsymbol{\mu}_o^T\boldsymbol{v}_c)}{\left(\sum_{w=1}^{V}\exp(\boldsymbol{\mu}_w^T\boldsymbol{v}_c)\right)^2} \\
&= \frac{\sum_{w=1}^{V}\left((\boldsymbol{\mu}_w - \boldsymbol{\mu}_o)\exp(\boldsymbol{\mu}_w^T\boldsymbol{v}_c)\right)}{\sum_{w=1}^{V}\exp(\boldsymbol{\mu}_w^T\boldsymbol{v}_c)}
\end{aligned}$$

(b) For $\boldsymbol{\mu}_k$ where $k \neq o$, we have:

$$\begin{aligned}
\frac{\partial CE}{\partial \boldsymbol{\mu}_k} &= \frac{1}{\hat{y}_o}\frac{\exp(\boldsymbol{\mu}_o^T\boldsymbol{v}_c)\exp(\boldsymbol{\mu}_k^T\boldsymbol{v}_c)\boldsymbol{v}_c}{\left(\sum_{w=1}^{V}\exp(\boldsymbol{\mu}_w^T\boldsymbol{v}_c)\right)^2} \\
&= \frac{\exp(\boldsymbol{\mu}_k^T\boldsymbol{v}_c)\boldsymbol{v}_c}{\sum_{w=1}^{V}\exp(\boldsymbol{\mu}_w^T\boldsymbol{v}_c)}
\end{aligned}$$

For $\boldsymbol{\mu}_o$, we have:

$$\begin{aligned}
\frac{\partial CE}{\partial \boldsymbol{\mu}_o} &= -\frac{1}{\hat{y}_o}\frac{\boldsymbol{v}_c\exp(\boldsymbol{\mu}_o^T\boldsymbol{v}_c)\sum_{w=1,\ w\neq o}^{V}\exp(\boldsymbol{\mu}_w^T\boldsymbol{v}_c)}{\left(\sum_{w=1}^{V}\exp(\boldsymbol{\mu}_w^T\boldsymbol{v}_c)\right)^2} \\
&= -\frac{\boldsymbol{v}_c\sum_{w=1,\ w\neq o}^{V}\exp(\boldsymbol{\mu}_w^T\boldsymbol{v}_c)}{\sum_{w=1}^{V}\exp(\boldsymbol{\mu}_w^T\boldsymbol{v}_c)}
\end{aligned}$$

(c) The gradient of negative sampling loss with respect to $\boldsymbol{v}_c$ is given by:

$$\frac{\partial J_{NS}}{\partial \boldsymbol{v}_c} = (\sigma(\boldsymbol{\mu}_o^T\boldsymbol{v}_c) - 1)\boldsymbol{\mu}_0 - \sum_{k=1}^{K}\left((\sigma(-\boldsymbol{\mu}_k^T\boldsymbol{v}_c) - 1)\boldsymbol{\mu}_k\right)$$

The gradient of loss with respect to $\boldsymbol{\mu}_k$ where $k \neq o$ is given by:

$$\frac{\partial J_{NS}}{\partial \boldsymbol{\mu}_k} = (1 - \sigma(-\boldsymbol{\mu}_k^T \boldsymbol{v}_c))\boldsymbol{v}_c$$

And the gradient of loss with respect to $\boldsymbol{\mu}_o$ is given by:

$$\frac{\partial J_{NS}}{\partial \boldsymbol{\mu}_o} = (\sigma(\boldsymbol{\mu}_o^T \boldsymbol{v}_c) - 1)\boldsymbol{v}_c$$

This cost function is much more efficient compared to softmax loss because in softmax loss we have to iterate through all words in vocabulary to calculate the inner product with the word vector of the center word, while in the negative sampling loss function, we just have to pick k random words and calculate the inner product, which gives us approximately $\frac{V-K}{V}$ speed up.

(d) In Skip-Gram, the gradient for word vector w_i where $t - m \leq i \leq t + m$ and $i \neq t$ is given by:

$$\frac{\partial J_{SG}}{\partial w_i} = \sum_{-m \leq j \leq m, j \neq 0} \frac{\partial F(w_j, \boldsymbol{v}_c)}{\partial w_i}$$

And the gradient for center word $\boldsymbol{v}_c$ is given by:

$$\frac{\partial J_{SG}}{\partial \boldsymbol{v}_c} = \sum_{-m \leq j \leq m, j \neq 0} \frac{\partial F(w_j, \boldsymbol{v}_c)}{\partial \boldsymbol{v}_c}$$

For CBOW, the gradient for w_t is given by:

$$\frac{\partial J_{CBOW}}{\partial w_t} = \frac{\partial F(w_t, \hat{\boldsymbol{v}})}{\partial w_t}$$

And the gradient for $\boldsymbol{v}_i$ where $t - m \leq i \leq t + m$ and $i \neq t$ is given by:

$$\frac{\partial J_{CBOW}}{\partial \boldsymbol{v}_i} = \frac{\partial F(w_t, \hat{\boldsymbol{v}})}{\partial \hat{\boldsymbol{v}}}$$

(e) See file q3_word2vec.py

(f) See file q3_sgd.py

(g) The plot is:

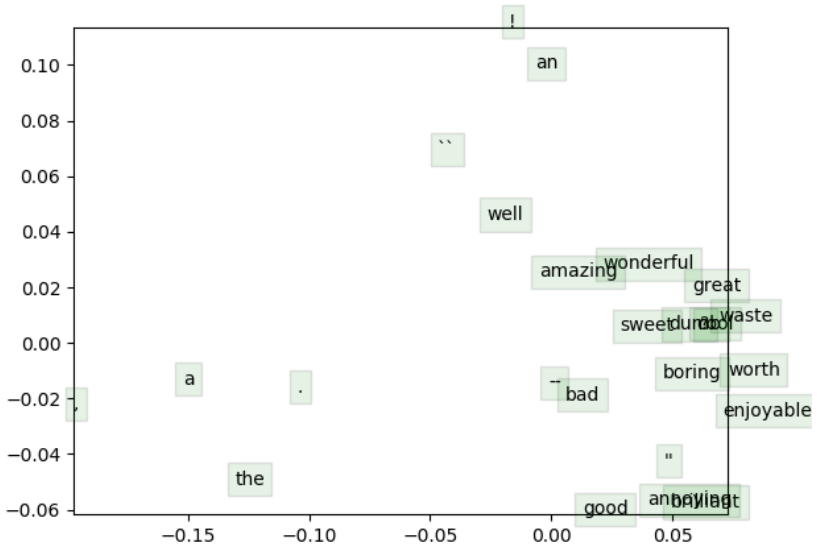

图 1: Visualization of Word Vectors

As we can see from the plot, all adjective words are clustered together at the bottom right corner of the plot, while article words like ”the” is on the other side of the plot.

4 Sentiment Analysis

(a) See file q4_sentiment.py

(b) The reason why we want to introduce regularization when doing classification is to prevent overfitting.

(c) See file q4_sentiment.py. My code for *chooseBestModel* is:

```
current_best_dev_acc = 0
for result in results:
    if result['dev'] > current_best_dev_acc:
        bestResult = result
        current_best_dev_acc = result['dev']
```

(d) The best results are:

	Train	Dev	Test
Our vectors	31.004%	32.698%	30.362%
Pretrained	39.934%	36.512%	37.014%

The pretrained vectors are better because:

1. These vectors are trained on a much larger data set.
2. These vectors are trained for a much longer time.
3. GloVe tends to work better than Skip-Gram and CBOW.

(e) The plot is:

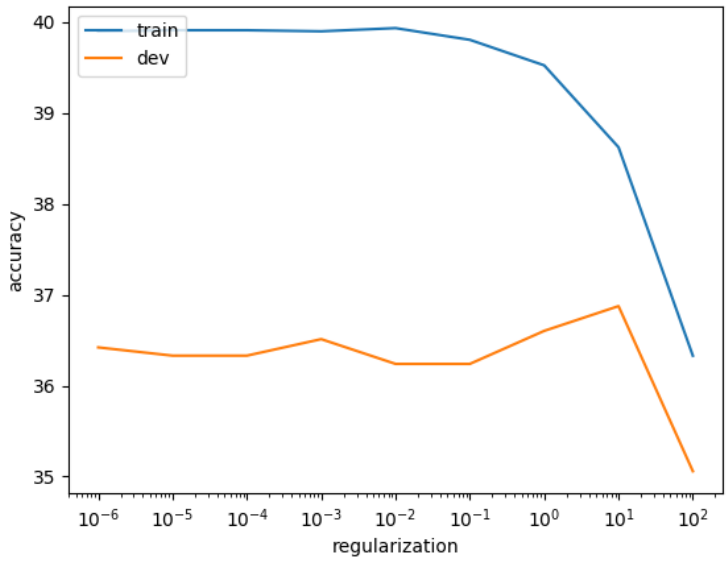

图 2: Reg Value with respect to Dev and Train Acc

We can see from above that when regularization value is small, there is a huge gap between training accuracy and dev accuracy, meaning that we have encountered overfitting. As the regularization value

increase, the gap becomes smaller, meaning that we have greatly reduced the overfitting. However, as the regularization value continues to increase, both the training accuracy and dev accuracy begin to fall, meaning that we may encounter underfitting.

(f) The plot is:

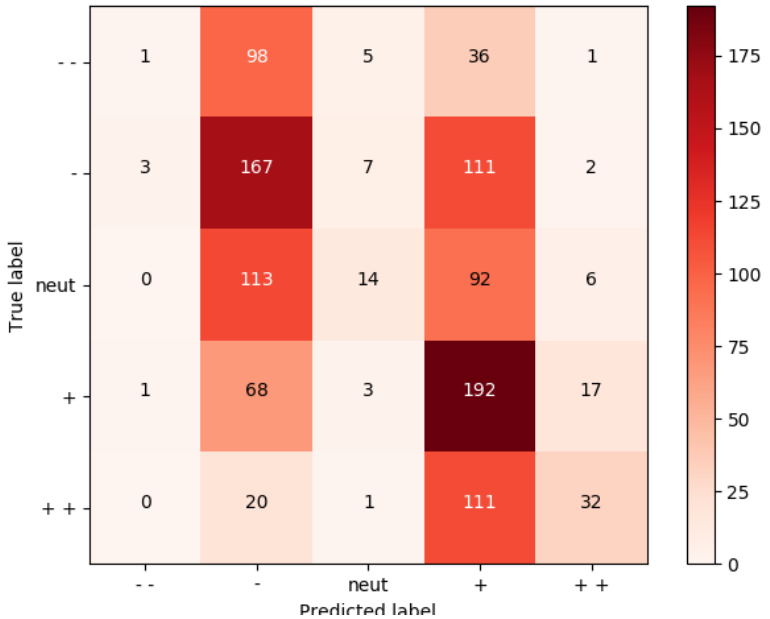

图 3: Confusion Matrix on the Dev Set (with Pretrained Vectors)

It is easy to see from above that our model tends to predict a sentence more positive than it should be. Also, our model using the pretrained vectors works quite well that it usually only makes small mistakes (like predicting a positive sentence to be very positive) rather than big mistakes (like predicting a very negative sentence to be very positive).

(g) For the following examples:

1. Comment: like mike is a winner for kids, and no doubt a winner for lil bow wow, who can now add movies to the list of things he does well. Its ground truth is very positive (4), however, we predict it as negative (1). We will need feature ”list of things he does well”.

2. Comment: this riveting world war ii moral suspense story deals with the shadow side of american culture: racial prejudice in its ugly and diverse forms. Its ground truth is negative (1), however we predict it as positive (3).

3. Comment: a painfully funny ode to bad behavior. Its ground truth is positive (3), while we predict it as vert negative (0). This maybe is because our model sees words like ”painfully” and ”bad”. The feature we need to classify it right is ”ode”.