CS224N: Assignment #1

Kangwei Ling

February 14, 2017

1 Softmax

(a) eliminate the term c.

$$softmax(\boldsymbol{x}+\boldsymbol{c})_i = \frac{e^{\boldsymbol{x}_i+c}}{\sum_j e^{\boldsymbol{x}_j+c}} = \frac{e^c \cdot e^{\boldsymbol{x}_i}}{e^c \cdot \sum_j e^{\boldsymbol{x}_j}} = softmax(\boldsymbol{x})_i$$

(b) in `q1_softmax.py`.

2 Neural Network Basics

(a)

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

(b) As $\boldsymbol{y}$ is the one-hot label vector, $\sum_j y_j = 1$, and

$$\frac{\partial \hat{y}_j}{\partial \theta_i} = \begin{cases} -\hat{y}_j\hat{y}_i & \text{if } i \neq j \\ \hat{y}_j(1-\hat{y}_j) & \text{otherwise} \end{cases}$$

thus

$$\begin{aligned} \frac{\partial}{\partial \theta_i} CE(\boldsymbol{y}, \hat{\boldsymbol{y}}) &= -y_i(1-\hat{y}_i) - \sum_{j\neq i} y_j(-\hat{y}_i) \\ &= -y_i + y_i\hat{y}_i + \hat{y}_i \sum_{j\neq i} y_j \\ &= \hat{y}_i \sum_j y_j - y_i \\ &= \hat{y}_i - y_i \end{aligned}$$

Therefore, $\frac{\partial}{\partial \theta} CE(\boldsymbol{y}, \hat{\boldsymbol{y}}) = \hat{\boldsymbol{y}} - \boldsymbol{y}$.

(c) First find the gradients with respect to the hidden layer $\boldsymbol{h}$. Let $\boldsymbol{z_2} = \boldsymbol{h}\boldsymbol{W}_2 + \boldsymbol{b}_2, \boldsymbol{z_1} = \boldsymbol{x}\boldsymbol{W}_1 + \boldsymbol{b}_1, \boldsymbol{\delta}_2 = \frac{\partial(\cdot)}{\partial \boldsymbol{z_2}}, \boldsymbol{\delta}_1 = \frac{\partial(\cdot)}{\partial \boldsymbol{z_1}}$. ($(\cdot) = CE(\boldsymbol{y}, \hat{\boldsymbol{y}})$, $\circ$ means piecewise multiplication). thus

$$\begin{aligned}
\boldsymbol{\delta}_2 &= \hat{\boldsymbol{y}} - \boldsymbol{y} \\
\frac{\partial(\cdot)}{\partial \boldsymbol{h}} &= \boldsymbol{\delta}_2 \boldsymbol{W}_2^T \\
\boldsymbol{\delta}_1 &= \boldsymbol{\delta}_2 \boldsymbol{W}_2^T \circ \sigma'(\boldsymbol{z_1}) \\
\boldsymbol{\delta}_3 &= \frac{\partial(\cdot)}{\partial \boldsymbol{x}} = \boldsymbol{\delta}_1 \boldsymbol{W}_1^T
\end{aligned}$$

(d) $D_x H + H D_y + H + D_y$

(e) `q2_sigmoid.py`

(f) `q2_gradcheck.py`

(g) `q2_neural.py`

3 word2vec

(a) Let $\boldsymbol{y}, \hat{\boldsymbol{y}}$ be column vectors. All vectors are column vectors.

$$\frac{\partial J}{\partial \boldsymbol{v}_c} = \boldsymbol{U}(\hat{\boldsymbol{y}} - \boldsymbol{y})$$

(b)

$$\begin{aligned}
\frac{\partial J}{\partial \boldsymbol{u}_i} &= (\hat{y}_i - y_i)\boldsymbol{v}_c \\
\frac{\partial J}{\partial \boldsymbol{U}} &= \boldsymbol{v}_c (\hat{\boldsymbol{y}} - \boldsymbol{y})^T
\end{aligned}$$

(c) gradients with respect to $\boldsymbol{v}_c$:

$$\frac{\partial J}{\partial \boldsymbol{v}_c} = (\sigma(\boldsymbol{u}_o^T \boldsymbol{v}_c) - 1)\boldsymbol{u}_o + \sum_{k=1}^{K} (1 - \sigma(-\boldsymbol{u}_k^T \boldsymbol{v}_c))\boldsymbol{u}_k$$

gradients with respect to $\boldsymbol{U}$:

$$\frac{\partial J}{\partial \boldsymbol{u}_i} = \begin{cases} (\sigma(\boldsymbol{u}_o^T \boldsymbol{v}_c) - 1)\boldsymbol{v}_c & \text{if } i = o \\ (1 - \sigma(-\boldsymbol{u}_i^T \boldsymbol{v}_c))\boldsymbol{v}_c & \text{if } i \in \{1, \ldots, K\} \\ 0 & \text{otherwise} \end{cases}$$

(d) For skip-gram:

$$\frac{\partial J_{skip-gram}}{\partial \boldsymbol{U}} = \sum_{-m \le j \le m, j \neq 0} \frac{\partial F(\boldsymbol{w}_{c+j}, \boldsymbol{v}_c)}{\partial \boldsymbol{U}}$$
$$\frac{\partial J_{skip-gram}}{\partial \boldsymbol{v}_c} = \sum_{-m \le j \le m, j \neq 0} \frac{\partial F(\boldsymbol{w}_{c+j}, \boldsymbol{v}_c)}{\partial \boldsymbol{v}_c}$$
$$\frac{\partial J_{skip-gram}}{\partial \boldsymbol{v}_i} = \boldsymbol{0} \quad \text{for } i \neq c$$

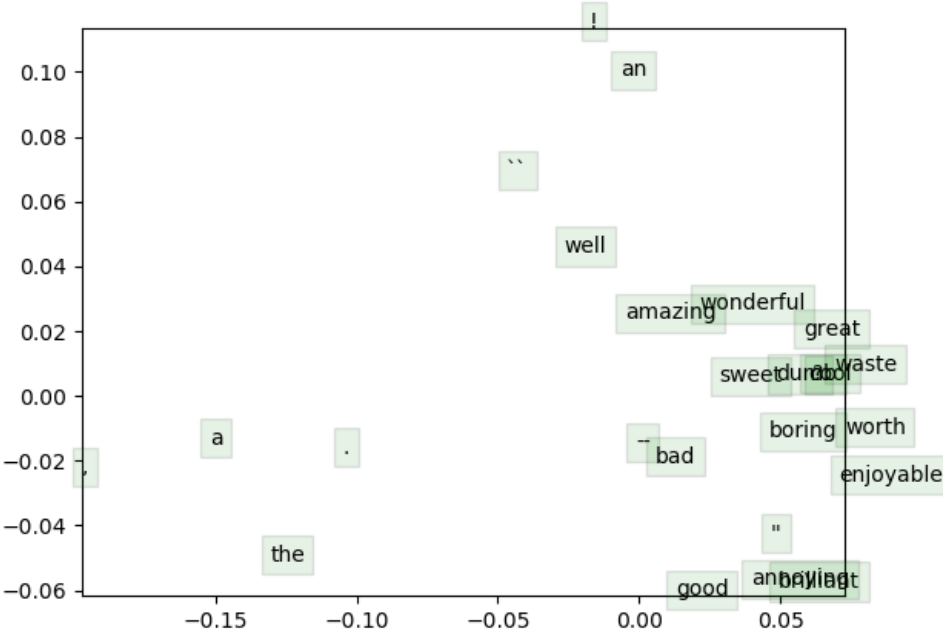
For CBOW:

$$\frac{\partial J_{CBOW}}{\partial \boldsymbol{U}} = \frac{\partial F(\boldsymbol{w}_c, \hat{\boldsymbol{v}})}{\partial \boldsymbol{U}}$$
$$\frac{\partial J_{CBOW}}{\partial \boldsymbol{v}_j} = \frac{\partial F(\boldsymbol{w}_c, \hat{\boldsymbol{v}})}{\partial \hat{\boldsymbol{v}}} \quad \text{for } j \in \{c-m, ..., c-1, c+1, ..., c+m\}$$
$$\frac{\partial J_{CBOW}}{\partial \boldsymbol{v}_j} = \boldsymbol{0} \quad \text{for } j \notin \{c-m, ..., c-1, c+1, ..., c+m\}$$

(e) `q3_word2vec.py`

(f) `q3_sgd.py`

(g)



(h) `q3_word2vec.py`

4 Sentiment Analysis

(a) `q4_sentiment.py`

(b) To prevent out model from overfitting.

(c) `q4_sentiment.py`

(d)

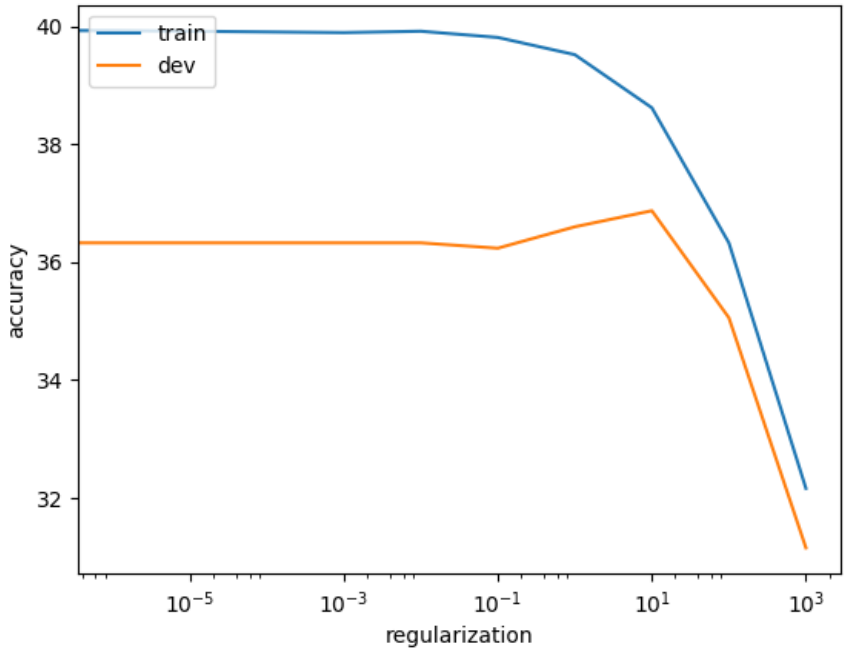
Best Accuracies	word vectors from problem 3	pretrained
train	31.121	39.934
dev	32.698	36.876
test	30.271	37.69

Table 1: best accuracies

The model using pretrained GloVe vectors outperforms for the following reasons:

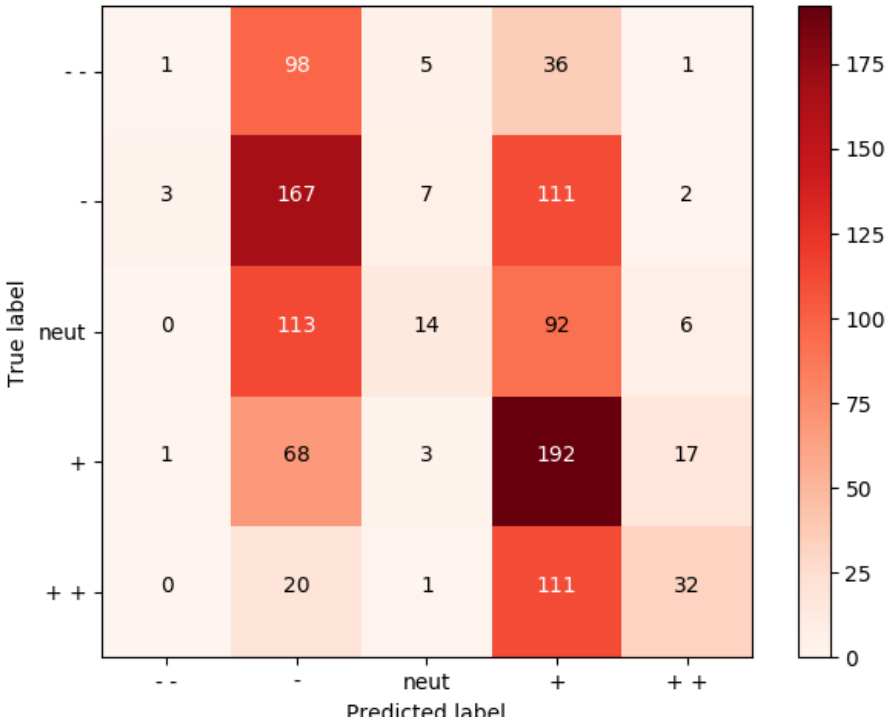
- The word vectors are trained on massive data (wikipedia), thus hold more information
- GloVe's vectors use different dimension from word2vec.
- GloVe represents words better than word2vec.

(e)



Regularization helped tackle overfitting, but regularization with too big weight harm the performance.

(f)



The mispredictions of the model mostly happened near neutral zone. For strong sentiment sentence, the prediction is quite accurate.

(g)

- Some cold/negative words in the sentence cause the whole sentence to be predicted as negative, while the sentence has a positive context.(Or negative meaning get overrun with positive words)

  nothing 's at stake , just a twisty double-cross you can smell a mile away – still , the derivative nine queens is lots of fun .

  This sentence is with positive label(3), yet was predicted as negative(1).

- Some negation words are not exhibiting their negating effect.

  scores no points for originality , wit , or intelligence .

  True label(0), predicted label(3).

- Names hurt performance.

  it takes a strange kind of laziness to waste the talents of robert forster , anne meara , eugene levy , and reginald veljohnson all in the same movie .

  True label(0), predicted label(3).