CS224n HW1

Xinyu Tan

July 19, 2017

1 Softmax

(a)

Take a look at any element i,

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

Therefore, we have $\text{softmax}(\boldsymbol{x}+c) = \text{softmax}(\boldsymbol{x})$

(b)

Note the first case illustrate the broadcasting principle (dimension match) in numpy:

```
if len(x.shape) > 1:
    #matrix
    x = x - np.max(x, axis=1, keepdims=True)
    x = np.exp(x) / np.sum(np.exp(x), axis=1, keepdims=True)
else:
    # vector
    x = x - x.max() # normalize
    x = np.exp(x)/np.sum(np.exp(x))
```

2 Neural Network Basics

(a)

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

(b)

First, we have

$$\hat{y}_i = \frac{e^{\theta_i}}{\sum_j e^{\theta_j}}$$

For one-hot encoding, only k-th element in $\boldsymbol{y}$ is *one*, so we have

$$
\begin{aligned}
CE(\boldsymbol{y}, \hat{\boldsymbol{y}}) &= -\log \hat{y}_k = -\log \frac{e^{\theta_k}}{\sum_j e^{\theta_j}} \\
&= -\theta_k + \log \sum_j e^{\theta_j}
\end{aligned}
$$

Therefore,

$$
\begin{aligned}
\frac{\partial CE(\boldsymbol{y}, \hat{\boldsymbol{y}})}{\partial \theta_k} &= -1 + \frac{e^{\theta_k}}{\sum_j e^{\theta_j}} \\
\frac{\partial CE(\boldsymbol{y}, \hat{\boldsymbol{y}})}{\partial \theta_i} &= \frac{e^{\theta_i}}{\sum_j e^{\theta_j}}, \forall i \neq k
\end{aligned}
$$

Put them altogether,

$$
\frac{\partial CE(\boldsymbol{y}, \hat{\boldsymbol{y}})}{\partial \boldsymbol{\theta}} = -\boldsymbol{y} + \mathrm{softmax}(\boldsymbol{\theta}) = \hat{\boldsymbol{y}} - \boldsymbol{y}
$$

(c)

We have

$$
\begin{aligned}
\hat{\boldsymbol{y}} &= \mathrm{softmax}(\boldsymbol{h}\boldsymbol{W}_2 + \boldsymbol{b}_2) \\
&= \mathrm{softmax}\left(\mathrm{sigmoid}(\boldsymbol{x}\boldsymbol{W}_1 + \boldsymbol{b}_1) + \boldsymbol{b}_2\right)
\end{aligned}
$$

Denote $\boldsymbol{a}_2 = \boldsymbol{h}\boldsymbol{W}_2 + \boldsymbol{b}_2$ and $\boldsymbol{a}_1 = \boldsymbol{x}\boldsymbol{W}_1 + \boldsymbol{b}_1$, then

$$
\begin{aligned}
\frac{\partial CE(\boldsymbol{y}, \hat{\boldsymbol{y}})}{\partial \boldsymbol{x}} &= \frac{\partial CE(\boldsymbol{y}, \hat{\boldsymbol{y}})}{\partial \boldsymbol{a}_2} \frac{\partial \boldsymbol{a}_2}{\partial \boldsymbol{x}} \\
&= (\hat{\boldsymbol{y}} - \boldsymbol{y}) \frac{\partial \boldsymbol{a}_2}{\partial \boldsymbol{h}} \frac{\partial \boldsymbol{h}}{\partial \boldsymbol{x}} \\
&= (\hat{\boldsymbol{y}} - \boldsymbol{y}) \boldsymbol{W}_2^T \sigma'(\boldsymbol{a}_1) \boldsymbol{W}_1^T
\end{aligned}
$$

(d)

There are in total $D_x H + H + H D_y + Dy$ parameters.

3 word2vec

(a)

For a word $\boldsymbol{o}$, the loss function is

$$
\begin{aligned}
L = -y_o \log \hat{y}_o &= -\log \frac{\exp(\boldsymbol{u}_o^T \boldsymbol{v}_c)}{\sum_{w=1}^W \exp(\boldsymbol{u}_w^T \boldsymbol{v}_c)} \\
&= -\boldsymbol{u}_o^T \boldsymbol{v}_c + \log \sum_{w=1}^W \exp(\boldsymbol{u}_w^T \boldsymbol{v}_c)
\end{aligned}
$$

Therefore, derivative with respect to $\boldsymbol{v}_c$ is:

$$\begin{aligned}\frac{\partial L}{\partial \boldsymbol{v}_c} &= -\boldsymbol{u}_o + \frac{1}{\sum_{l=1}^{W}\exp(\boldsymbol{u}_l^T\boldsymbol{v}_c)}\sum_{w=1}^{W}\exp(\boldsymbol{u}_w^T\boldsymbol{v}_c)\boldsymbol{u}_w \\ &= -\boldsymbol{u}_o + \sum_{w=1}^{W} p(w|c)\boldsymbol{u}_w\end{aligned}$$

where

$$p(w|c) = \frac{\exp(\boldsymbol{u}_w^T\boldsymbol{v}_c)}{\sum_{l=1}^{W}\exp(\boldsymbol{u}_l^T\boldsymbol{v}_c)}$$

.

(b)

Similarly,

$$\frac{\partial L}{\partial \boldsymbol{u}_w} = \begin{cases} -(1-\hat{y}_o)\boldsymbol{v}_c(& w = o \\ \hat{y}_w\boldsymbol{v}_c & w \neq o \end{cases}$$

(c)

Given

$$J_{\text{neg-sample}}(\boldsymbol{o}, \boldsymbol{v}_c, \boldsymbol{U}) = -\log\sigma(\boldsymbol{u}_o^T\boldsymbol{v}_c) - \sum_{k=1}^{K}\log\sigma(-\boldsymbol{u}_o^T\boldsymbol{v}_c)$$

we have

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

and

$$\frac{\partial J}{\partial \boldsymbol{u}_w} = \begin{cases} -\sigma(-\boldsymbol{u}_w^T\boldsymbol{v}_c)\boldsymbol{v}_c & w = o \\ \sigma(\boldsymbol{u}_w^T\boldsymbol{v}_c)\boldsymbol{v}_c & w \neq o \end{cases}$$

(d)

Let's denote $\boldsymbol{U}$ the matrix that aggregates all the output word vectors.

1. Skip-gram:

$$\frac{\partial J_{\text{skip-gram}}}{\partial \boldsymbol{U}} = \sum_{j\neq 0}\frac{\partial F(\boldsymbol{u}_{c+j}, \boldsymbol{v}_c)}{\partial \boldsymbol{U}}$$

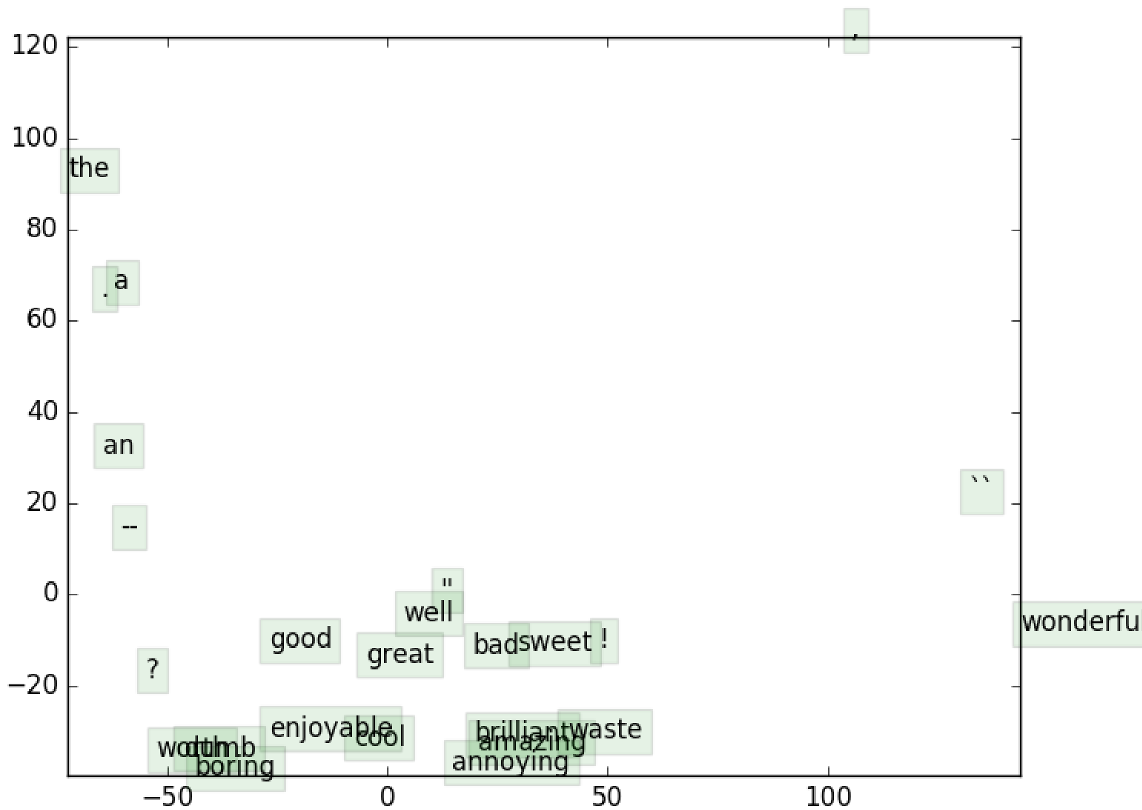
$$\frac{\partial J_{\text{skip-gram}}}{\partial \boldsymbol{v}_c} = \sum_{j\neq 0}\frac{\partial F(\boldsymbol{u}_{c+j}, \boldsymbol{v}_c)}{\partial \boldsymbol{v}_c}$$

2. CBOW:

$$\frac{\partial J_{\text{CBOW}}}{\partial \boldsymbol{U}} = \frac{\partial F(\boldsymbol{u}_c, \hat{\boldsymbol{v}})}{\partial \boldsymbol{U}}$$

$$\frac{\partial J_{\text{CBOW}}}{\partial \boldsymbol{v}_j} = \frac{\partial F(\boldsymbol{u}_c, \hat{\boldsymbol{v}})}{\partial \hat{\boldsymbol{v}}}$$

(g)



Summary: "the", "a", and "an" are close to each other; most adjectives are clustered at the bottom of the graph; "wonderful" somehow is far from its similar words.

4 Sentiment Analysis

(b)

The main reason to introduce regularization is to avoid overfitting.

(c)

Code for choosing the best model:

```
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 = None

    ### YOUR CODE HERE
    test_acc = []
    for res in results:
        test_acc.append(res["dev"])
    ### END YOUR CODE
    best_index = test_acc.index(max(test_acc))
    bestResult = results[best_index]
    return bestResult
```