# CS 224N: Assignment #1

Saran Ahluwalia

2019-01-09

**Question 1(a)**

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

**Question 2(a)**

$$\begin{aligned}
\frac{d}{dx}\sigma(x) &= \frac{d}{dx}\frac{1}{1+e^x} = \frac{-1}{(1+e^x)^2}\frac{d}{dx}(1+e^x) = \frac{-e^x}{(1+e^x)^2} && (1)\\
&= \frac{1}{1+e^x}\frac{-e^x}{1+e^x} = \sigma(x)\frac{(1+e^x)-(1+e^x)-e^x}{1+e^x} && (2)\\
&= \sigma(x)\left(1-\frac{1}{1+e^x}\right) = \sigma(x)(1-\sigma(x)) && (3)
\end{aligned}$$

**Question 2(b)**

$$\begin{aligned}
\forall k : \frac{\partial}{\partial\theta_k} log(\hat{y_i}) &= \frac{\partial}{\partial\theta_k} log\left(\frac{e^{\theta_i}}{\sum_j e^{\theta_j}}\right)\\
&= \frac{\partial}{\partial\theta_k} log(e^{\theta_i}) - \frac{\partial}{\partial\theta_k} log\left(\sum_j e^{\theta_j}\right)\\
&= \frac{\partial\theta_i}{\partial\theta_k} - \frac{1}{\sum_j e^{\theta_j}}\frac{\partial}{\partial\theta_k}\sum_j e^{\theta_j}\\
&= \mathbb{1}\{i=k\} - \frac{e^{\theta_k}}{\sum_j e^{\theta_j}} = \mathbb{1}\{i=k\} - \hat{y_k}
\end{aligned}$$

$$\begin{aligned}
\frac{\partial}{\partial\theta_k} CE(y,\hat{y}) &= -\sum_i y_i \frac{\partial}{\partial\theta_k} log(\hat{y_i}) = -y_k(1-\hat{y_k}) + \sum_{i\neq k} y_i\hat{y_k}\\
&= -y_k + y_k\hat{y_k} + \sum_{i\neq k} y_i\hat{y_k} = \hat{y_k}\sum_i y_i - y_k
\end{aligned}$$

Since $y$ is a one-hot label vector, $\sum_i y_i = 1$, hence we can conclude that

$$\frac{\partial}{\partial \theta_k} CE(y, \hat{y}) = \hat{y_k} - y_k$$

...or, in vector form

$$\frac{\partial}{\partial \theta} CE(y, \hat{y}) = \hat{y} - y \tag{4}$$

Note that $\theta$, the gradient, $y$ and $\hat{y}$ can be either row or column vectors as long as they all have the same shape.

**Question 2(c)** Let us first define some variables:

$$z = x\boldsymbol{W_1} + \boldsymbol{b_1} \tag{5}$$
$$h = sigmoid(z) \tag{6}$$
$$\theta = h\boldsymbol{W_2} + \boldsymbol{b_2} \tag{7}$$
$$\hat{y} = softmax(\theta) \tag{8}$$
$$J = CE(y, \hat{y}) \tag{9}$$

For consistency with the class programming assignment, we assume $x$, $h$, and $y$ are all row vectors having dimensions $(1 \times D_x)$ $(1 \times H)$ $(1 \times D_y)$ respectively. Since $\boldsymbol{b_1}$ and $z$ are of the same shape as $h$, they are all $(1 \times H)$ vectors. Since $\boldsymbol{b_2}$, $\theta$ and $\hat{y}$ are of the same shape as $y$, they are $(1 \times D_y)$ vectors. For (5) to be a valid equation, $\boldsymbol{W_1}$ must be a $(D_x \times H)$ matrix. Similarly, because of (7), $\boldsymbol{W_2}$ must be a $(H \times D_y)$ matrix.

Table 1: Variable and matrix dimensions.

| $x$ | $\boldsymbol{W_1}$ | $h$, $z$, $\boldsymbol{b_1}$ | $\boldsymbol{W_2}$ | $y$, $\hat{y}$, $\theta$, $\boldsymbol{b_2}$ |
|---|---|---|---|---|
| $(1 \times D_x)$ | $(D_x \times H)$ | $(1 \times H)$ | $(H \times D_y)$ | $(1 \times D_y)$ |

Using chain rule and simple equation partial derivative formulas we can calculate the following composite derivative:

$$\frac{\partial J}{\partial x} = \frac{\partial J}{\partial \theta}\frac{\partial \theta}{\partial h}\frac{\partial h}{\partial z}\frac{\partial z}{\partial x} = (\hat{y} - y)\boldsymbol{W_2}^\top diag\{\sigma'(z)\}\boldsymbol{W_1}^\top \tag{10}$$

The following table verifies dimensionality correctness of the above equation:

| $\frac{\partial J}{\partial x}$ | $(\hat{y} - y)$ | $\boldsymbol{W_2}^\top$ | $diag\{\sigma'(z)\}$ | $\boldsymbol{W_1}^\top$ |
|---|---|---|---|---|
| $(1 \times D_x)$ | $(1 \times D_y)$ | $(D_y \times H)$ | $(H \times H)$ | $(H \times D_x)$ |

Furthermore, forgoing matrix product associativity, using $\odot$ to denote Hadamard product and taking into account (3), (11) can be optimized to:

$$\frac{\partial J}{\partial x} = (((\hat{y} - y)\boldsymbol{W_2}^\top) \odot (1 - z) \odot z)\boldsymbol{W_1}^\top \tag{11}$$

**Question 2(d)** Assuming dimesionalities of $\boldsymbol{W_1}$, $\boldsymbol{b_1}$, $\boldsymbol{W_2}$ and $\boldsymbol{b_2}$ (see Table 1) the total number of parameters is

$$D_xH + H + HD_y + D_y$$

**Questions 3(a) and 3(b)** For consistency with programming assignments we assume $u_i$ and $v_i$ are $(1 \times D)$ row vectors. Let:

$$U = \begin{pmatrix} — & u_1 & — \\ — & u_2 & — \\ & \vdots & \\ — & u_W & — \end{pmatrix} \qquad \in \mathbb{R}^{(W \times D)} \tag{12}$$

$$\theta = (u_1 \cdot v_c, u_2 \cdot v_c ... u_W \cdot v_c) = v_c U^\top = (U v_c^\top)^\top \qquad \in \mathbb{R}^{(1 \times W)} \tag{13}$$

$$\tag{14}$$

$$\hat{y} = softmax(\theta) \qquad \in \mathbb{R}^{(1 \times W)} \tag{15}$$

$$\tag{16}$$

$$y(o) \text{ is a one-hot label row vector} \qquad \in \mathbb{R}^{(1 \times W)} \tag{17}$$

Then, assuming $J_{softmax-CE}(o, v_c, U) = CE(y, \hat{y})$ and using (4), we can calculate the following derivatives:

$$\frac{\partial J}{\partial v_c} = \frac{\partial J}{\partial \theta} \frac{\partial \theta}{\partial v_c} = (\hat{y} - y)U \tag{18}$$

$$\frac{\partial J}{\partial U} = \frac{\partial J}{\partial \theta} \frac{\partial \theta}{\partial U} = (\hat{y} - y)^\top v_c \tag{19}$$

The following table verifies dimensionality correctness of the above equations:

| $\frac{\partial J}{\partial v_c}$ | $(\hat{y} - y)$ | $U$ | $\frac{\partial J}{\partial U}$ | $(\hat{y} - y)^\top$ | $v_c$ |
|---|---|---|---|---|---|
| $(1 \times D)$ | $(1 \times W)$ | $(W \times D)$ | $(W \times D)$ | $(W \times 1)$ | $(1 \times D)$ |

**Question 3(c)** For consistency with programming assignment we assume $u_i$ and $v_i$ are $(1 \times D)$ row vectors. Let us first define:

$$U^{(K)} = \begin{pmatrix} — & u_o & — \\ — & u_1 & — \\ — & u_2 & — \\ & \vdots & \\ — & u_K & — \end{pmatrix} \qquad \in \mathbb{R}^{(K+1) \times D}$$

$$\delta = (\sigma(u_o \cdot v_c) - 1, \sigma(u_1 \cdot v_c), \sigma(u_2 \cdot v_c) ... \sigma(u_K \cdot v_c)) = (U^{(K)} v_c^\top)^\top - (1, 0...0) \qquad \in \mathbb{R}^{1 \times (K+1)}$$

Also, let's observe the following relationship:

$$\frac{\partial}{\partial a} log(\sigma(a \cdot b)) = \frac{1}{\sigma(a \cdot b)} \frac{\partial}{\partial a} \sigma(a \cdot b) = \frac{\sigma(a \cdot b)(1 - \sigma(a \cdot b))}{\sigma(a \cdot b)} \frac{\partial}{\partial a}(a \cdot b) = (1 - \sigma(a \cdot b))b$$

From the above and the definition of $J_{neg.s.}$:

$$J_{neg.s.}(0, v_c, U) = -log(\sigma(u_o \cdot v_c)) - \sum_{k=1}^{K} log(\sigma(-u_k \cdot v_c)) \tag{20}$$

follows:

$$\begin{aligned} \frac{\partial J_{neg.s.}}{\partial v_c} &= -(1 - \sigma(u_o \cdot v_c))u_o - \sum_{k=1}^{K} (1 - \sigma(-u_k \cdot v_c))(-u_k) \\ &= (\sigma(u_o \cdot v_c) - 1)u_o + \sum_{k=1}^{K} \sigma(u_k \cdot v_c)u_k \qquad (21) \\ &= -u_o + \sum_{k=o}^{K} \sigma(u_k \cdot v_c)u_k = \delta U^{(K)} \end{aligned}$$

Simlarly:

$$\begin{aligned} \frac{\partial J_{neg.s.}}{\partial u_o} &= -(1 - \sigma(u_o \cdot v_c))v_c = v_c(\sigma(u_o \cdot v_c) - 1) \\ \frac{\partial J_{neg.s.}}{\partial u_k} &= +(1 - \sigma(-u_k \cdot v_c))v_c = v_c \sigma(u_k \cdot v_c) \end{aligned}$$

or in matrix form:

$$\frac{\partial J_{neg.s.}}{\partial U^{(K)}} = \delta^{\top} v_c \tag{22}$$

Note that since $U^{(K)}$ is a sub-matrix of the whole output word matrix $U$. Therefore $\frac{\partial J_{neg.s.}}{\partial U}$ is a sparse matrix with zeros everywhere except in corresponding rows from $\frac{\partial J_{neg.s.}}{\partial U^{(K)}}$.

The cost of computing (20) (21) and (22) is $O(KD)$. On the other hand, the cost of computing $J_{softmax-CE}$ is $O(WD)$, which makes it roughly $W/D$ more expensive. Since W runs in thousands or tens of thousands while K runs in tens, negative sampling is approximately two to three orders of magnitude more efficient.

**Question 3(c)** Since every each member of skip-gram cost sum is dependent on its all output words in the dictionary as well as on the center predicted word:

$$\frac{\partial J_{skip-gram}}{\partial U} = \sum_{-m\leq j\leq m; j\neq 0} \frac{\partial F(w_{c+j}, v_c)}{\partial U}$$
$$\frac{\partial J_{skip-gram}}{\partial v_c} = \sum_{-m\leq j\leq m; j\neq 0} \frac{\partial F(w_{c+j}, v_c)}{\partial v_c}$$

COWB's single member is additionally dependent on all constituents of $\hat{v}$:

$$\frac{\partial J_{CBOW}}{\partial U} = \frac{\partial F(w_c, \hat{v})}{\partial U}$$
$$\frac{\partial J_{CBOW}}{\partial v_i} = \frac{\partial F(w_c, \hat{v})}{\partial \hat{v}} \frac{\partial \hat{v}}{\partial v_i} = \left\{ \begin{array}{cc} \frac{\partial F(w_c,\hat{v})}{\partial \hat{v}} & \forall i \in \{w_{c-m}...w_{c-1}, w_{c+1}...w_{c+m}\} \\ 0 & \text{otherwise} \end{array} \right\}$$

**Question 3(g)** The training result visualization is shown on Figure 1. It appears a number of words expressing human satisfaction or degree of enjoyment are clustering together. On the other hand, we can see that both negative and positive sentiments appear mixed. It is possible that this is a limitation of 2D projection and the the words corresponding to the opposite sentiments but appearing together in this plot, e.g. great and waste, are more distant in other dimensions.

**Question 4(b)** Regularization is one of the methods used to prevent overfitting of the model that is to prevent the model from memorizing the training data set rather than learning its general features that would be applicable to more data. Although there are other methods, regularization is a simple, quantifiable parameter that imposes penalty on mode's numeric complexity, i.e. overall magnitude of its parameters.

**Question 4(c)**

```
def chooseBestModel(results):

    ### YOUR CODE HERE
    bestResult = results[0]

    for model in results[1:]:
        if model["dev"] > bestResult["dev"]:
            bestResult = model

    ### END YOUR CODE

    return bestResult
```

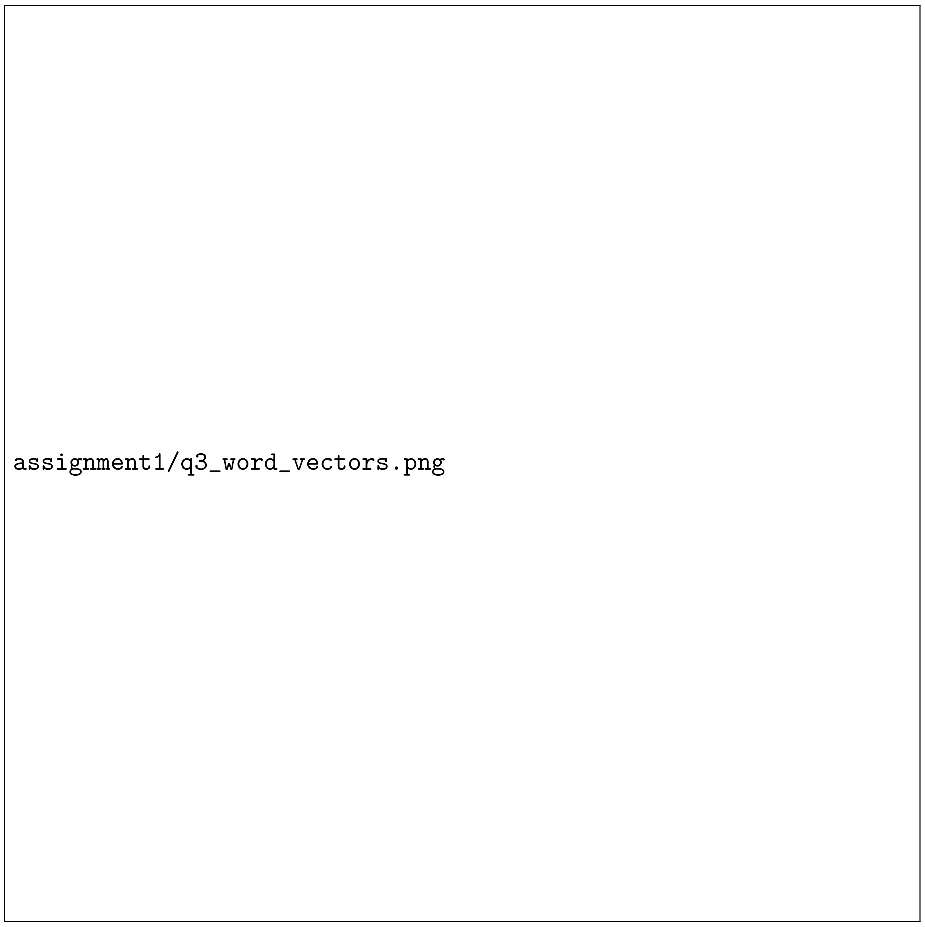

Figure 1: Training result visualization.

**Question 4(d)** The following table summarizes the best set of accuracies from 2 respective runs of `q4 sentiment.py`:

| Model | Train | Dev | Test |
|---|---|---|---|
| My word vectors from q3 | 30.489 | 31.244 | 30.136 |
| Pretrained GloVe vectors on Wikipedia | 39.162 | 37.148 | 38.054 |

Clearly GloVe model produced better results. The most likely reasons for it are (1) higher dimensionality of the model allowing for richer feature representation, (2) significantly greater amount of training data producing that allows capturing more detailed relationships and, perhaps, (3) the ability of GloVe model to minimize the effect of words and word pairs that occur much more often.

**Question 4(e)** The plot of the train and dev accuracy with respect to the regularization value generated by `q4_sentiment.py --pretrained` is shown on Figure 2.

`assignment1/q4_reg_v_acc.png`

Figure 2: Accuracy vs regularization parameter.

The plot clearly demonstrates the effect of regularization parameter on model behavior. Since this is a relatively simple sentence model with word relationship features we can see that the increased regularization has little effect on training until the parameter gets huge, which time model practically collapses. Also the simplicity of model leads to its slow generalization as the gap between train and development accuracy metrics stay almost constant for long time. Eventually, at lambda's sweet spot the development accuracy does rises and comes much closer to the training one giving us a warm and fuzzy feeling that the model reached its best.

`assignment1/q4_dev_conf.png`

Figure 3: Confusion matrix.

**Question 4(e)** The confusion matrix generated by `q4_sentiment.py --pretrained` is shown on Figure 3.

The biggest problem demonstrated by this matrix is that the model fails to be precise. It's generally correct in finding general mood of the sentence, however it's to shy to be precise– neither extremes, nor neutral buckets are adequately represented. It also appears the "modest" +/- demonstrate some kind of bell curve pattern pointing to mostly random (Gaussian?) distribution indicating unsophisticated nature of the sentence model.

**Question 4(f)** The following are the exemplary mistakes the model made:

```
3 1 and if you 're not nearly moved to tears by a couple of scenes ,
you 've got ice water in your veins .
```

A clearly positive sentence was mistakenly predicted as a very negative because it failed to capture the sarcasm of the reviewer

```
3 1 nothing 's at stake , just a twisty double-cross you can smell a
mile away -- still , the derivative nine queens is lots of fun .|
```

A similar misclassification, most likely caused by many negative words like "nothing", "twisty" or "double-cross", even thought a "lots of fun" at the end would defintely otweghted the decision if the feature model accounted for it.

```
0 3 ultimately feels empty and unsatisfying , like swallowing a
communion wafer without the wine .
```

An opposite mistake, quite a puzzling one because "unsatisfying" and "empty" should have put it in the grib bucket. A bug?