Assignment 2

1 Understanding word2vec [15 points]

(a) According to writeup,

$$\mathbf{J}_{\text{naive}-\text{softmax}}(\mathbf{v}_c, o, \mathbf{U}) = -\log P(O=o|C=c) = -\log \hat{y}_o = -\sum_{w\in\text{Vocab}} y_w \log \hat{y}_w.$$

Note that

1. scalar $\hat{y}_o = P(O=o|C=c) = \frac{\exp(\mathbf{u}_o^T\mathbf{v}_c)}{\sum_{w\in\text{Vocab}}\exp(\mathbf{u}_w^T\mathbf{v}_c)}$, and
2. ground-truth $\mathbf{y}$ is an one-hot vector where only $y_o = 1$. □

(b) Use the chain rule,

$$\begin{aligned}
\frac{\partial \mathbf{J}}{\partial \mathbf{v}_c} &= -\frac{\partial}{\partial \mathbf{v}_c}\log P(O=o|C=c) \\
&= -\frac{1}{P(O=o|C=c)} \cdot \frac{\partial P(O=o|C=c)}{\partial \mathbf{v}_c} \\
&= -\frac{\sum_{w\in\text{Vocab}}\exp\left(\mathbf{u}_w^T\mathbf{v}_c\right)}{\exp\left(\mathbf{u}_o^T\mathbf{v}_c\right)} \\
&\cdot \frac{\exp\left(\mathbf{u}_o^T\mathbf{v}_c\right)\left[\mathbf{u}_o\sum_{w\in\text{Vocab}}\exp\left(\mathbf{u}_w^T v_c\right) - \sum_{w\in\text{Vocab}}\mathbf{u}_w\exp\left(\mathbf{u}_w^T\mathbf{v}_c\right)\right]}{\left[\sum_{w\in\text{Vocab}}\exp\left(\mathbf{u}_w^T\mathbf{v}_c\right)\right]^2} \\
&= \frac{\sum_{w\in\text{Vocab}}\mathbf{u}_w\exp\left(\mathbf{u}_w^T\mathbf{v}_c\right)}{\sum_{w\in\text{Vocab}}\exp\left(\mathbf{u}_w^T\mathbf{v}_c\right)} - \mathbf{u}_o.
\end{aligned}$$

Since $\mathbf{U} = \left[\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_{|\text{Vocab}|}\right]$, we have

$$\frac{\sum_{w\in\text{Vocab}}\mathbf{u}_w\exp\left(\mathbf{u}_w^T\mathbf{v}_c\right)}{\sum_{w\in\text{Vocab}}\exp\left(\mathbf{u}_w^T\mathbf{v}_c\right)} = \mathbf{U}\hat{\mathbf{y}},$$

and

$$\mathbf{u}_o = \mathbf{U}\mathbf{y}$$

Therefore, the derivative is $\mathbf{U}\left(\hat{\mathbf{y}} - \mathbf{y}\right)$, where $\mathbf{U} \in \mathbb{R}^{d\times|\text{Vocab}|}$, and $\hat{\mathbf{y}}, \mathbf{y} \in \mathbb{R}^{|\text{Vocab}|}$.

The derivative equals to zero when

1. $\hat{\mathbf{y}} = \mathbf{y}$ (impossible due to the nature of softmax), or
2. $\hat{\mathbf{y}} - \mathbf{y} \in \ker\left(\mathbf{U}\right)$ (possible since d is usually much less than $|\text{Vocab}|$).

Assume vector $\mathbf{v}_c$ is randomly initialized and $\mathbf{U}$ remains fixed, then gradient descent updates $\mathbf{v}_c$ by pulling it towards the ground-truth context vector $\mathbf{U}\mathbf{y}$ and repelling it from the (false) predicted distribution $\mathbf{U}\hat{\mathbf{y}}$:

$$\mathbf{v}_c := \mathbf{v}_c - \alpha\mathbf{U}(\hat{\mathbf{y}} - \mathbf{y}).$$

(c) L2 normalization takes away information about magnitudes of vector. If we sum word vectors to get representation of the phrase, the normalization will change the representation and thus may affect following classification results.

(d) Similar to derivation in (b), we have

$$\frac{\partial \mathbf{J}}{\partial \mathbf{u}_w} = -\frac{1}{P(O=o|C=c)} \cdot \frac{\partial P(O=o|C=c)}{\partial \mathbf{u}_w}.$$

Case 1: $\mathbf{u}_o \neq \mathbf{u}_w$. In this case,

$$\frac{\partial \mathbf{J}}{\partial \mathbf{u}_w} = \frac{\exp\left(\mathbf{u}_w^T \mathbf{v}_c\right)}{\sum_{w \in \text{Vocab}} \exp\left(\mathbf{u}_w^T \mathbf{v}_c\right)} \cdot \mathbf{v}_c = \hat{y}_w \mathbf{v}_c.$$

Case 2: $\mathbf{u}_o = \mathbf{u}_w$. In this case,

$$\frac{\partial \mathbf{J}}{\partial \mathbf{u}_w} = \left[\frac{\exp\left(\mathbf{u}_w^T \mathbf{v}_c\right)}{\sum_{w \in \text{Vocab}} \exp\left(\mathbf{u}_w^T \mathbf{v}_c\right)} - 1\right] \cdot \mathbf{v}_c = (\hat{y}_w - 1)\mathbf{v}_c.$$

(e) The derivation has been formulated in (d).

2 Machine Learning & Neural Networks [8 points]

(a) *Momentum* makes gradient updates largely depend on previvous updates. In essence it's an exponential moving average. *Adaptive learning rates* adjust learning rate according to previous gradient magnitudes. Parameters with smaller gradients tend to have larger learning rates. This mechanism may help keep gradient updates in an appropriate range.

(b) $\gamma = \frac{1}{1-p_{\text{drop}}}$, by noting that

$$\begin{aligned} \mathbb{E}\left[\mathbf{h}_{\text{drop}}\right]_i &= \mathbb{E}\left[\gamma \mathbf{d} \odot \mathbf{h}\right]_i \\ &= \gamma \mathbb{E}\left[d_i\right] \mathbb{E}\left[h_i\right] \\ &= \gamma\left(1 - p_{\text{drop}}\right) h_i. \end{aligned}$$

Dropout is applied during training to prevent overfitting. By randomly dropping units, it prevents neurons from co-adapting too much (relying on specific other neurons) and forces the network to learn more robust features. It also acts as an approximate method of training an ensemble of different neural network architectures.

Dropout is not applied during evaluation because we want to use the full capacity of the network to make the best possible predictions. We want a deterministic output, not a stochastic one. Using the full network acts as an approximation of averaging the predictions from the ensemble of all thinned networks formed during training.