CMPT 413/825 Natural language processing
Homework 2, Fall 2020

Due: October 15th, 2020

Instructions Answers are to be submitted by 11:59pm of the due date as a PDF file. Please submit your answers using either Gradescope (preferred) or Coursys. You do not need to submit your answers using both platforms, just one.

- **Gradescope**: Go to https://www.gradescope.ca/, select **HW2** and submit your PDF. Please make sure to assign the pages of your submission to the corresponding questions.
- **Coursys**: Go to Coursys, select the **Homework 2** activity and submit your answer as **answer.pdf**.

This assignment is to be done individually.

1 Vector Semantics (10 pts)

(a) Explain what the term "word vector" mean. What properties would we want a vector space of words to have? (2 pts)

(b) Explain what the "distributional hypothesis" refers to and how it can be used for learning representations of each individual word in a large text corpus. (2 pts)

(c) Give two advantages of representing words using dense vectors. (2 pts)

(d) Word2vec is a way of learning semantic vector spaces, where each word is represented using one vector. Give two limitations of these semantic vector spaces. (2 pts)

(e) Given word vectors, explain how you can use the word vectors to obtain representations for sentences. (2 pts)

2 Term Frequency Representations (12 pts)

Suppose we have some documents and we are interested in a few words like "amazing", "perfect", "nice" and "good". The table specifies raw counts, count(t, d), of each word t for each document d. The document frequency (df) is given for each word in the column "df". Assume that the total number of documents is given by $N = 2500$.

Words	**doc 1**	**doc 2**	**doc 3**	**df**
amazing	20	42	14	165
perfect	31	3	0	65
nice	10	0	0	219
good	14	20	43	350

The inverse document frequency of a term (idf_t) can be calculated as:

$$\text{idf}_t = \log_{10}(\frac{N}{\text{df}_t})$$

The term frequency of term t in document d ($\text{tf}_{t,d}$) can be calculated as:

$$\text{tf}_{t,d} = \begin{cases} 1 + \log_{10} \text{count}(t,d) & \text{if count}(t,d) > 0 \\ 0 & \text{otherwise} \end{cases}$$

(a) Compute the tf-idf weights of these four words for the three documents in the table. (6 pts)

(b) Take the documents as vectors of the tf-idf weights and calculate the cosine similarity between document 2 and 3. (2 pts)

(c) Why do we prefer tf-idf representation over raw counts of words? (2 pts)

(d) What are two limitations of the tf-idf representation? (2 pts)

3 PPMI (18 pts)

Following table shows the word-word co-occurrence matrix for a text corpus. Here each cell contains the count of times the (row) word occurs in the context of the (column) word.

	toast	**tasty**	**bone**	**muscle**
butter	30	9	0	0
cheese	54	39	0	0
human	0	0	29	27
skeleton	0	0	43	14

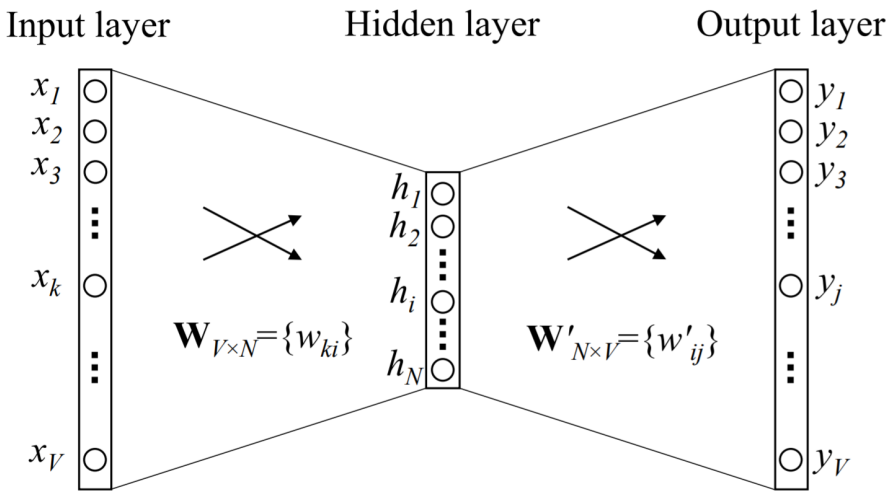
(a) Compute the PPMI weights for the given co-occurrence matrix directly from the raw co-occurrence counts. (7 pts)

(b) Use Add-1 smoothing (Laplace smoothing) to the raw co-occurrence counts and again compute the PPMI weights. (7 pts)

(c) Did you notice any changes between (a) and (b)? Explain your answer. (4 pts)

4 Learning Word Vectors (23 pts)

In this problem, we will look at how the word2vec model can be implemented using a neural network. Consider the following neural network architecture.



As a simplification, assume that we only consider contexts consisting of one word, which means that the network predicts one target word given one context word as input. In this problem, the vocabulary size is V and the hidden layer size is N. The layers on adjacent units are fully connected. The input to the neural network $\mathbf{x}$ is a one-hot vector. This means that out of V units, only one of them will be 1 and all other units are 0 (only one of the x_i is set to 1).

The initial layer of the network transforms the input $\mathbf{x} \in \mathbb{R}^V$ using the weight matrix $\mathbf{W} \in \mathbb{R}^{V\times N}$ into

$$\mathbf{h} = \mathbf{W}^\top \mathbf{x}.$$

The second layer of the network uses a second weight matrix $\mathbf{W}' \in \mathbb{R}^{N\times V}$ to obtain a vector of scores for all words in the vocabulary:

$$\mathbf{u} = \mathbf{W}'^\top \mathbf{h}.$$

The score for each word can be written as:

$$u_j = \mathbf{w}'^\top_j \mathbf{h}$$

where $\mathbf{w}'_j$ is the jth column of W'. Then we can get the multinomial distribution of the words by applying a softmax:

$$p(\text{word}_j|\mathbf{x}) = y_j = \text{softmax}(\mathbf{u})_j$$

(a) Give the training objective for this model for one training sample (t, c) with target word t and context word c. Assume that we are maximizing the log likelihood. (5 pts)

(b) Find the update equation for w'_{ij}. Assume that we use SGD to optimize the above objective. (8 pts)

(c) How would you obtain word embeddings using this model (after you have trained it)? (4 pts)

(d) Describe how would you extend this model to obtain the full word2vec CBOW model? Can it be extended to obtain the word2vec Skip-gram model? (6 pts)

5 Error Backpropagation (12 pts)

Suppose we have the following neural network for the binary sentiment classification task. The output is either 1 (for Positive sentiment) or 0 (for Negative sentiment). Given the network, we will derive error derivatives using back-propagation. For your answers, please follow the notation given in the network.

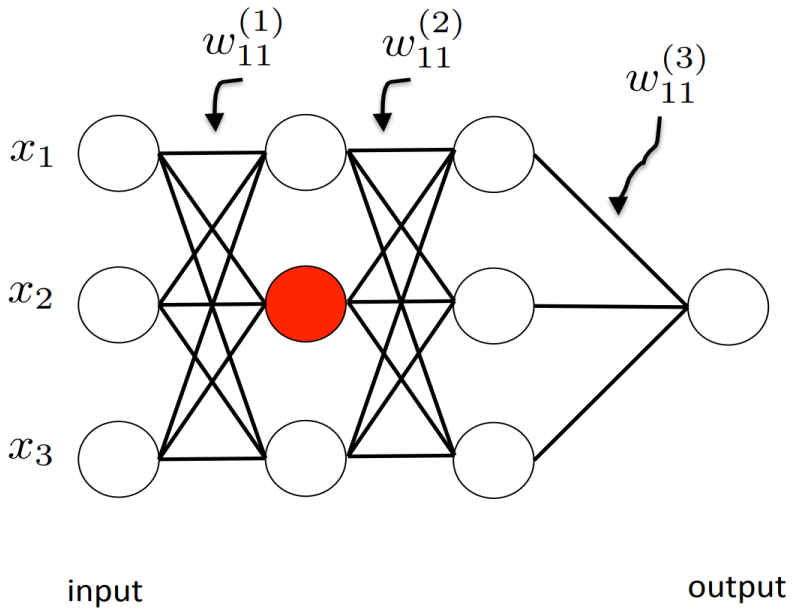
We will use $a_i^{(j)}$ to refer to the input to the activation function of the ith unit of the jth layer, and $z_i^{(j)}$ the output of the ith unit of the jth layer, and $w_{ik}^{(j)}$ is the weight associated with the output of kth unit of the jth layer to the ith unit of the $j+1$th layer.

As an example, the red node has as input to the activation function:

$$a_2^{(2)} = w_{21}^{(1)} x_1 + w_{22}^{(1)} x_2 + w_{23}^{(1)} x_3$$

and its output would be:

$$z_2^{(2)} = h(a_2^{(2)})$$



Assume that the activation functions for the hidden layers is ReLU. For the final output node, assume the activation function is a sigmoid function.

$$\text{ReLU}(x) = \max(0, x) \tag{1}$$

$$\text{sigmoid}(x) = \frac{1}{1 + e^{-x}} \tag{2}$$

Also, assume this network is trained using to minimize the cross entropy error. Let $\text{CE}_n(\mathbf{w})$ be the cross entropy loss for the nth training sample $(\mathbf{x}_n, y_n)$

$$E_n = \text{CE}_n(\mathbf{w}) = -y_n \log(\hat{y}_n) - (1 - y_n) \log(1 - \hat{y}_n) = -y_n \log g(\mathbf{x}_n, \mathbf{w}) - (1 - y_n) \log(1 - g(\mathbf{x}_n, \mathbf{w})) \tag{3}$$

where $\hat{y}_n = g(\mathbf{x}_n, \mathbf{w})$ is the predicted output of the network, and y_n is the ground-truth (0 or 1). Note that $g(\mathbf{x}_n, \mathbf{w})$ is the function represented by the neural network with input $\mathbf{x}_n$ and parameters $\mathbf{w}$. Let's use the backpropagation algorithm to compute the derivative of E_n.

For simplicity, for the ReLU, you only need to consider the case when the input to the ReLU is positive (so $a_i^{(j)} > 0$ when it is input to a ReLU). As a short hand, let $\delta_k^{(j)} = \frac{\partial E_n}{\partial a_k^{(j)}}$ be the partial derivative of the cross entropy loss with respect to $a_k^{(j)}$.

(a) Consider the output layer (layer 4). Calculate $\frac{\partial E_n}{\partial a_1^{(4)}}$. Note that $a_1^{(4)}$ is the input to the activation function of the output node. Using our short hand, $\frac{\partial E_n}{\partial a_1^{(4)}} = \delta_1^{(4)}$. (2 pts)

(b) Use the result from (a) to calculate $\frac{\partial E_n}{\partial w_{12}^{(3)}}$. (2 pts)

(c) Consider the second to last layer of nodes (layer 3). Write an expression for $\frac{\partial E_n}{\partial a_1^{(3)}}$. Use $\delta_1^{(4)}$ in this expression. (2 pts)

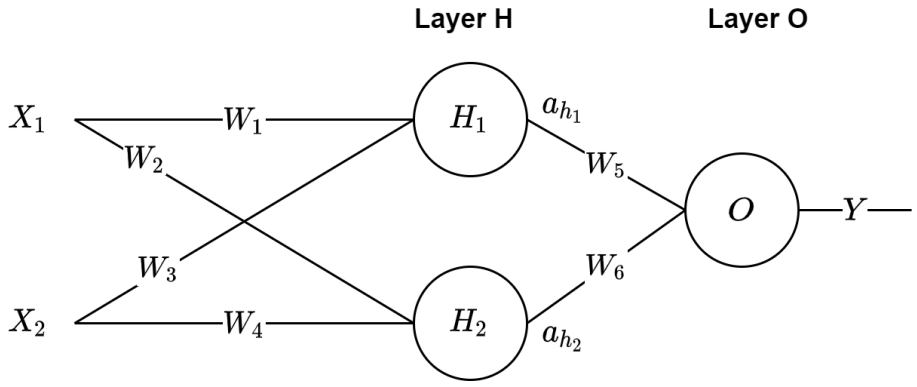
(d) Use the result from (c) to calculate $\frac{\partial E_n}{\partial w_{11}^{(2)}}$. (2 pts)

(e) Consider the weights connecting from the inputs. Write an expression for $\frac{\partial E_n}{\partial a_1^{(2)}}$. Use the set of $\delta_k^{(3)}$ in this expression. (2 pts)

(f) Use result from (e) to calculate $\frac{\partial E_n}{\partial w_{11}^{(1)}}$. (2 pts)

6 Feed-Forward Neural Networks (25pt)

Consider the following neural network, consisting of inputs X_1 and X_2. The network consists of one hidden layer H containing two units H_1 and H_2, and the output layer O contains a single unit O_1. The output Y is the final output of the network.



In this problem, we will consider the effects of different activation functions in the layers of the network. Given an activation function ϕ, the output of each unit can be written as $\phi(\mathbf{w}^\top \mathbf{z})$ where $\mathbf{w}$ is the weight vector and $\mathbf{z}$ is the input vector. For instance, the output of the hidden unit H_1 can be defined as:

$$a_{h_1} = \phi(\mathbf{w}_{h_1}^\top \mathbf{x}) \text{ where } \mathbf{w}_{h_1} = \begin{pmatrix} W_1 \\ W_3 \end{pmatrix} \text{ and } \mathbf{x} = \begin{pmatrix} X_1 \\ X_2 \end{pmatrix}$$

Similarly we can define the output of H_2 and O_1 (i.e., a_{h_2} and Y).

(a) Assume that the network is using *linear* activation functions for all the units in layers H and O. The output of the hidden unit H_1 can then be written as: $a_{h_1} = C \times (\mathbf{w}_{h_1}^\top \mathbf{x})$ for some fixed constant C. Now suppose we want to remove H_1 from the hidden layer. How can we redesign the neural network with new weights expressed in terms of the old weights and the constant C? (5 pts)

(b) Show that a neural network with arbitrary number of hidden layers with arbitrary number of nodes and linear activations can be written as a neural network without any hidden layers. (5 pts)

(c) Another common activation function is a threshold or step function, where the activation $t(x) = 1$ if $x > 0$ and $t(x) = 0$ otherwise. Let the hidden units use sigmoid activation functions and let the output unit use a threshold activation function. Find weights such that the output of the given network is the Exclusive NOR (XNOR) of the inputs X_1 and X_2 where both X_1 and X_2 are binary variables. As a reminder, the XNOR of X_1 and X_2 is 1 if $X_1 = X_2$ and 0 otherwise. Keep in mind that there is no bias term for these units. (5 pts)

(d) Why is the ReLU (Rectified Linear Unit) a popular activation function in neural networks? Give two properties of this function. (5 pts)

(e) We want the neural network to learn the logistic regression function using the same architecture as in the figure. Each unit must use a sigmoid, linear, or a threshold activation function. Choose activation functions for all the units in the network in both layers H and O so that the network resembles a logistic regression classifier. (5 pts)