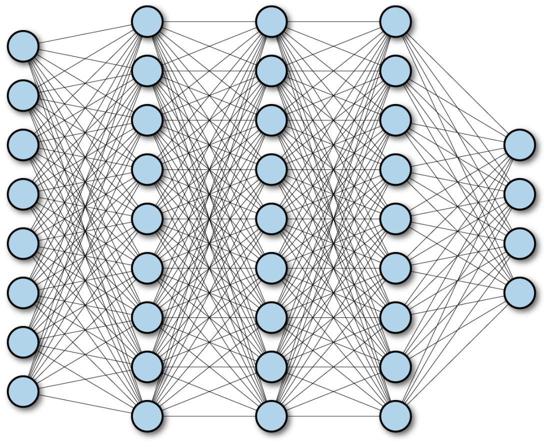
Advanced Methods in Text Analytics
Exercise 2: Deep Learning Basics and Word Embeddings

Daniel Ruffinelli

FSS 2025

1 Fully-Connected Neural Networks

A fully-connected neural network is a compositional (possibly non-linear) function that, in general, transforms an input vector $\boldsymbol{x}$ into a vector of outputs $\boldsymbol{y}$. Each node in such a network is a unit (often referred to as an artificial neuron) that holds a numerical value. Each edge in the network also holds a numerical value. These units are combined to create *layers*, usually an *input* layer (leftmost in image below), an *output* layer (rightmost in image below) and any number of *hidden* layers in between input and output layers.



(a) How many units should we have in the input layer? And in the output layer?

(b) What is the difference between the values in nodes and those in edges?

(c) What is the *general* operation performed by each unit (node) in the network? What is the input to this operation? Give a formal expression for it.

(d) What is the main difference between different types of artificial neurons? Give two examples of different types of neurons.

(e) Denote by $\boldsymbol{X} \in \mathbb{R}^{N \times d}$ the input matrix constructed by stacking N input vectors $\boldsymbol{x}_i \in \mathbb{R}^d$ as rows. Similarly, denote by $\boldsymbol{H}_1$ the matrix that contains the corresponding representations computed by the first hidden layer, which has p units. What is the size of $\boldsymbol{H}_1$? Give a formal expression for the operation that results in $\boldsymbol{H}_1$ given input $\boldsymbol{X}$.

2 Backpropagation

This task is based on a previous task written by Goran Glavas.

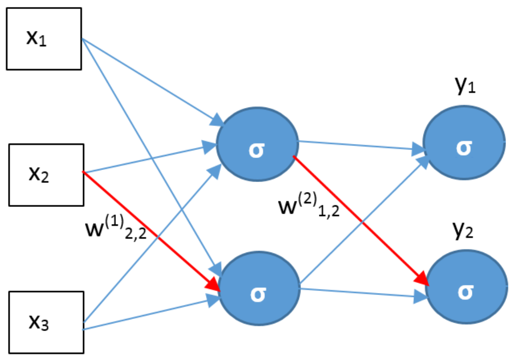
With backpropagation, we can get gradient information that is useful to update the parameters of a multi-layer neural network during training. We start by computing the relevant gradients of the output layer and then continue backwards by computing the gradients of previous layers based on the gradients we computed before.

(a) What are the *relevant gradients* we care about during training?

(b) The update rules of gradient-based optimizers require that we compute gradients of the functions we are optimizing. For example, when backpropagating through a layer with sigmoid activations, we will need the gradient of the sigmoid function. Show that this gradient is given by:

$$\sigma(x)' = \sigma(x) \cdot (1 - \sigma(x)), \text{where } \sigma(x) = \frac{1}{1 + e^{-x}}$$

(c) You are given the toy feed-forward neural network shown in the figure below. This network is used to predict one of two classes given three numeric input features: x_1, x_2, and x_3. The network has one hidden layer and one output layer, each consisting of two neurons. The activation function on both layers is the sigmoid function.



(i) Derive the update rule for the weights marked in red in the image, i.e., for $w^{(1)}_{2,2}$ and $w^{(2)}_{1,2}$. Assume you are training your model with the following square error loss:

$$L = -\frac{1}{2}\frac{1}{N}\sum_{e=1}^{N}\sum_{l=1}^{M}(t_{e,l} - y_{e,l})^2.$$

where N is size of the training set, M is number of labels, $t_{e,l}$ the target label l of example e and $y_{e,l}$ the corresponding prediction made by the model. Note that in this toy setting, $M = N = 2$. Using square error loss for a classification task is not a common setting, but it is useful for us to construct a simple example to study backpropagation.

(ii) You are given the toy dataset consisting of the following two training examples: $([-1, 0, 2], [0, 1])$ and $([2, 1, 1], [1, 0])$. These examples are of the form $(\boldsymbol{x}, \boldsymbol{y})$, where $\boldsymbol{x}$ is input features, and $\boldsymbol{y}$ is corresponding label. Assuming that all weights are initialized to $w = 1$, carry out one update iteration for denoted weights $w^{(1)}_{2,2}$ and $w^{(2)}_{1,2}$, using the two training examples for supervision. Use learning rate $\psi = 1$ and the square error loss given above.

3 Skip-Gram Basics

This task is based on a previous task written by Goran Glavas.

Consider the sentence

$$\textit{The } \underset{c_1}{\textit{quick}} \ \underset{c_2}{\textit{brown}} \ \underset{w}{\textit{fox}} \ \underset{c_3}{\textit{jumped}} \ \underset{c_4}{\textit{over}} \textit{ the lazy dog},$$

where w marks the *center word* and c_i the *context words.* In this example, we have a *context window* of size 2, meaning we use the 2 previous and 2 subsequent words around w as context.

In the lecture, we saw how examples for the skip-gram model are constructed and used to learn a logistic regression model for the task of predicting whether two given words are in the same context, as done by the authors of skip-gram. Formally, they used the following binary classification task to learn word embeddings: $p(\text{True}|w,c) = \sigma(l_{w,c})$ and $p(\text{False}|w,c) = 1-p(\text{True}|w,c)$, where $l_{w,c}$ is the logit score (i.e. unnormalized prediction) produced by a logistic regression model given words w and c. Given this task, sentence and context window, some positive examples of the form (source word, target word) would be (*fox, quick*), (*fox, brown*), etc.

However, in their original work on the skip-gram approach, the authors learned word embeddings using a different task: predicting context word c_j given center word w_i in position i of a given sequence of words. We refer to this task as the *original task* throughout this exercise.

(a) How many classes are there in the original task proposed for skip-gram? And how could we compute probabilities for each class given logit scores from some classifier? Give a formal expression for this original task, including the way probabilities are computed from logits.

(b) Generate all positive training examples for the original skip-gram task using the given sentence above as the entire training set. Use a window size of 2.

(c) How is the skip-gram model parameterized? What is the role of each parameter? Assume your vocabulary consists of 101.425 words and you want to learn word representations of 300 dimensions each. What is the number of parameters in the skip-gram approach in this setting?

(d) Assume that word *Frodo* is the 79th word in your vocabulary. How do we obtain a single (final) representation of the word *Frodo* after our model has finished training? What decision do we need to make to obtain such a final representation?

4 Skip-Gram Training

This task is based on a previous task written by Goran Glavas.

We now focus on how to train models using the skip-gram approach. Assume throughout that we use dot products to compute logit scores, i.e. $l_{w_i,c_j} = \boldsymbol{w}_i^T \boldsymbol{w}'_j$, and that we use the softmax function to obtain probabilities from logit scores

(a) To train a skip-gram model, we find the parameters $\boldsymbol{\theta}$, i.e. word embedings, that maximize the likelihood of our training data, i.e. we perform MLE. Given text corpus T, give the expression for the dataset likelihood $L(\boldsymbol{\theta}|T)$ as determined by a skip-gram model with a context window of size L. Give the corresponding negative log-likelihood expression under the empirical risk minimization framework. As in the lecture, assume independence between words and between training examples (i.i.d. assumption).

(b) Recall that computing the negative log-likelihood is equivalent to using cross-entropy loss. Consider the following toy vocabulary V={*Daenerys*, *Frodo*, *ring*, *dragon*, *Picard*, *ship*}. Assume you are training a skip-gram model over a large corpus T with this vocabulary, and that the current values of the parameters of your model are the following:

$$\mathbf{W} = \begin{bmatrix} 0.54 & 0.25 & 0.07 \\ 0.91 & 0.74 & 0.27 \\ 0.58 & 0.00 & 0.05 \\ 0.16 & 0.77 & 0.31 \\ 0.10 & 0.54 & 0.32 \\ 0.59 & 0.99 & 0.06 \end{bmatrix}, \mathbf{W'} = \begin{bmatrix} 0.19 & 0.64 & 0.98 \\ 0.17 & 0.42 & 0.47 \\ 0.55 & 0.24 & 0.92 \\ 0.89 & 0.58 & 0.13 \\ 0.31 & 0.73 & 0.48 \\ 0.49 & 0.20 & 0.54 \end{bmatrix}$$

Compute the value of the cross-entropy loss corresponding to the single training instance (*Frodo* (source word), *ring* (target word)) when used as positive example.

(c) Assume we train using stochastic gradient-descent, i.e. we update our relevant parameters during training after computing the loss for a single positive example as done in (b). Which parameters are updated after every single example? Why?

5 Word Embeddings as Matrix Factorization

As discussed in Task 1, we obtain the final representation of a given word from its corresponding representations in $\boldsymbol{W}$ and $\boldsymbol{W}'$. Let us now consider the perspective that $\boldsymbol{W}$ and $\boldsymbol{W}'$ are factors of another matrix $\boldsymbol{M}$, i.e. $\boldsymbol{M} = \boldsymbol{W}\boldsymbol{W}'^T$. If we can derive some meaning for $\boldsymbol{M}$, we may be able to interpret what we are learning to factorize with the skip-gram approach.

As in the lecture, we now take the perspective of skip-gram as the binary classification task $p(\text{True}|w, c) = \sigma(\boldsymbol{w}_w^t \boldsymbol{w}_c)$, where w is a source word, c a target (context) word, and $\boldsymbol{w}_w, \boldsymbol{w}_c$ their corresponding word representations. Note that in this setting, entry m_{ij} in $\boldsymbol{M}$ is given by $\boldsymbol{w}_i^T \boldsymbol{w}_j$.

(a) We saw in the lecture that the skip-gram training objective for *a single training example* (w, c) using the cross-entropy loss is the following:

$$L(w, c) = \log \sigma(\boldsymbol{w}_w^t \boldsymbol{w}_c) + \sum_{i=1}^{k} \log(-\boldsymbol{w}_w^t \boldsymbol{w}_i), \tag{1}$$

where k are the number of negative examples sampled per positive tuple (w, c) in the training set (we omit the negation of the loss for brevity). Rewrite Equation 1 so that it explicitly shows the *sampling* of negatives. To this end, use the expectation operator and assume negative examples $\boldsymbol{w}_i$ are drawn from the uniform distribution P over vocabulary V. That is, the probability of using word c to construct a negative is $\frac{\#(c)}{|V|}$, where $\#(c)$ is the number of times word c appears in the training corpus.

(b) Generalize your training objective derived in the previous question to include all training examples (w, c), i.e. the overall training objective L using the entire corpus.

(c) During training, we optimize the objective L. To see what this optimization process tells us about m_{wc}, show that the optimal value for $\boldsymbol{w}_w^t \boldsymbol{w}_c$ is given by the following expression:

$$\boldsymbol{w}_w^t \boldsymbol{w}_c = \ln\left(\frac{\#(w,c)|V|}{\#(w)\#(c)}\right) - \ln k \tag{2}$$

To get there, derive an optimal value for $\boldsymbol{w}_w^t \boldsymbol{w}_c$ by computing the gradient of L w.r.t *a single training example* (w, c), setting it to zero, and deriving a value for $\boldsymbol{w}_w^t \boldsymbol{w}_c$.
Hint: only two terms in L are relevant for this gradient computation.