Assignment 1: Word Vector

Aviv Jan, Tal Dvir

January 26, 2024

1 Understanding word2vec

a)

$$Softmax(x_i + c) = \frac{exp(x_i + c)}{\sum_j exp(x_j + c)} = \frac{exp(x_i) * exp(c)}{\sum_j exp(x_j * exp(c)} = \frac{exp(x_i)}{\sum_j exp(x_j)} = Softmax(x_i)$$

b)

$$-\sum_{w \in W} y_w \log(\hat{y}_w) = -\sum_{o \neq w \in W} y_w \log(\hat{y}_w) - y_o \log(\hat{y}_o) = \log(\hat{y}_o)$$

Because y is a one hot vector with a 1 only for the outside word o.

c)

$$\text{Loss} = -\log\left(\frac{e^{(u_o^T v_c)}}{\sum_{t \in W} e^{(u_t^T v_c)}}\right) = -u_o^T v_c + \log\left(\sum_{t \in W} e^{(u_t^T v_c)}\right)$$

Case $w \neq 0$:

$$\nabla v_c(\text{Loss}) = \frac{1}{\sum_{t \in W} e^{(u_t^T v_c)}} \sum_{t \in W} e^{(u_t^T v_c) * v_c} = \sum_{t \in W} p(t|c) * v_c = E_{x \sim p(t|c)}[v_c]$$

Case $w = 0$:

$$\nabla v_c(\text{Loss}) = -v_c + E_{x \sim p(t|c)}[v_c] = E_{x \sim p(t|c)}[v_c] - v_c$$

Explanation: In both lines the derivative of the second part is identical, therefore the difference between the cases boils down into the derivative of the first part. (In the first case its zeros out and in the second not).

d)

(i)

$$\frac{\partial J_{skip-gram}(c, w_t - m, ..., w_t + m, V, U)}{\partial U} = \frac{\partial \sum_{\substack{-m\leq j\leq m \\ j\neq 0}} J(c, w_t + j, V, U)}{\partial U} = \frac{\sum_{\substack{-m\leq j\leq m \\ j\neq 0}} \partial J(c, w_t + j, V, U)}{\partial U}$$

(ii)

$$\frac{\partial J_{skip-gram}(c, w_t - m, ..., w_t + m, V, U)}{\partial V_c} = \frac{\partial \sum_{\substack{-m\leq j\leq m \\ j\neq 0}} J(c, w_t + j, V, U)}{\partial v_c} = \frac{\sum_{\substack{-m\leq j\leq m \\ j\neq 0}} \partial J(c, w_t + j, V, U)}{\partial v_c}$$

(iii)

Assume $w \neq c$:

$$\frac{\partial J_{skip-gram}(c, w_t - m, ..., w_t + m, V, U)}{\partial v_w} = 0$$

Explanation:J doesn't depend on the V_w, when $w \neq c$ because we use v only for the center word, and in this context, the only sent the word c.

e)

The splitting of each token representation into two parts is crucial for a few reasons:

1. This separation allows the model to learn more robust representations. When a word is in the center position, its representation is used to predict surrounding words. Conversely, when it's an output token, it is being predicted based on the center words. This dual perspective enriches the learning process, allowing the model to understand both how a word predicts its context and how it is predicted by its context.

2. This split helps in capturing more accurate semantic relationships. For instance, the center word representation can become specialized in capturing the essence of the word itself, while the output word representation gets better at understanding how the word interacts with other words in different contexts.

3. If the same representation were used for both the center and output tokens, the inner product between a word and itself would be high. This is because the inner product of any vector with itself is always large, reflecting a high degree of similarity.

f)

The intuition of the algo:

1. we can think of words as people. Just like you can tell a lot about a person by who they hang out with, you can tell a lot about a word by the words it's often used with. Word2vec learns to understand words by the company they keep.

2. Words that are similar in meaning tend to appear in similar contexts. For example, "happy" and "joyful" often appear with words like "smile" or "celebration". so if we imagine each word as a point in space. Words with similar "friends" are placed closer together. As word2vec learns from more text, it keeps adjusting the positions of these points. Words that keep showing up in similar contexts move closer to each other.

For better intuition we think of it like knowing a person - In order to really know him you need to ask him about himself and ask others about him (usually a polite thing to do). The same in words - you want to see how they behave in a full context(sentence) and how they behave in different contexts

2 Optimizing word2vec

a)

Let Σ denote the vocabulary. As indicated in the provided hint, $\#(c, o)$ represents the count of co-occurrences of words c and o in the given corpus.

The objective function $J(\theta)$ is defined as the logarithm of the likelihood function $L(\theta)$:

$$J(\theta) = \log(L(\theta)) = \sum_{c\in\Sigma}\sum_{o\in\Sigma} \#(c, o) \log(p_\theta(c, o))$$

We are informed that $\theta^* = \arg\max_\theta L(\theta)$. Given the monotonic nature of the logarithm, it follows that:

$$\theta^* = \arg\max_\theta L(\theta) = \arg\max_\theta \log(L(\theta)) = \arg\max_\theta J(\theta)$$

Next, we consider specific center and context words, denoted as $\overline{c}$ and $\overline{o}$ respectively. The aim is to find the parameter $p_\theta(\overline{o}|\overline{c})$ that maximizes $J(\theta)$, subject to the probability constraint:

$$\sum_{o\in\Sigma} p\theta(o|\overline{c}) = 1$$

With the second hint, utilizing Lagrange multipliers, we derive the Lagrangian:

$$L(\theta_{\overline{c},\overline{o}}, \lambda) = \sum_{o\in\Sigma} \#(o, \overline{c}) \log(p_\theta(o|\overline{c})) + \lambda\left(1 - \sum_{o'\in\Sigma} p_\theta(o'|\overline{c})\right)$$

Deriving the Lagrangian with respect to $p_\theta(\overline{o}|\overline{c})$ gives:

$$\frac{\partial L(\theta_{\overline{c},\overline{o}})}{\partial p_\theta(\overline{o}|\overline{c})} = \frac{\#(\overline{c}, \overline{o})}{p_\theta(\overline{o}|\overline{c})} - \lambda$$

Setting this derivative to zero, we find:

$$p_\theta(\overline{o}|\overline{c}) = \frac{\#(\overline{c}, \overline{o})}{\lambda}$$

By further deriving the Lagrangian with respect to λ and equating to zero, we obtain:

$$1 = \sum_{o^{‘}\in\Sigma} p_\theta(o^{‘}|\overline{c})$$

Substituting the previous expression, we arrive at:

$$\lambda = \sum_{o^{‘}\in\Sigma} \#(\overline{c}, o^{‘})$$

Finally, substituting this result back into the first equation yields:

$$p_\theta(\overline{o}|\overline{c}) = \frac{\#(\overline{c}, \overline{o})}{\sum_{o^{‘}\in\Sigma} \#(\overline{c}, o^{‘})}$$

b)

We aim to demonstrate that achieving the optimum solution for a corpus over a vocabulary with no more than four words is impossible. Consider the corpus aa, aa, aa, ab, ab, ac. The resulting probabilities for this corpus are:

$$p(a|a) = \frac{1}{2}, \quad p(b|a) = \frac{1}{3}, \quad p(c|a) = \frac{1}{6}.$$

Let V be $\{\text{"a"}, \text{"b"}, \text{"c"}, \text{"d"}\}$ and the corpus is $\{\text{aa}, \text{aa}, \text{aa}, \text{ab}, \text{ab}, \text{ac}\}$. According to the result in part (a), we have:

$$p_{\theta^*} = \frac{\#(\overline{c}, \overline{o})}{\sum_{o^{‘} \in \Sigma} \#(o^{‘}, \overline{c})}$$

For $\overline{c} = "d"$, the optimal solution becomes:

$$p_{\theta^*}(o|\,"d") = \frac{0}{0} = 0 \quad , \quad (for all o \in V)$$

However, we know that for all c and for all o the probability distribution induced by the skip-gram model:

$$p_\theta = \frac{e^{(u_o^T v_c)}}{\sum_{w \in \sum} e^{(u_w^T v_c)}} > 0$$

Therefore, in our scenario, achieving the optimum is impossible.

3 Paraphrase Detection(theoretical)

a)

Initially, let's observe that for any values of x_1 and x_2, the following holds true:

$$\text{relu}(x_1)^\top \text{relu}(x_2) \geq 0$$

Consequently, for any x_1 and x_2, we derive the inequality:

$$\sigma(\text{relu}(x_1)^\top \text{relu}(x_2)) \geq \frac{1}{2}$$

Thus, for any pair of sentences x_1 and x_2, the model will predict them as paraphrases, resulting in an accuracy of $\frac{1}{4}$ on the given dataset (given the 1:3 ratio of positive to negative examples).

b)

An uncomplicated solution for the aforementioned issue could involve eliminating the ReLU function from the neural network.

This adjustment simplifies the model by exclusively utilizing the dot product of the two vectors without the ReLU non-linearity. By doing so, the dot product can now yield negative values, expanding the range of possible outcomes. This modification enables the sigmoid function to generate values within the range $(0, 1)$ for pairs of sentences.

Formally stated, the updated model is as follows:

$$p(\text{the pair is a paraphrase}|x_1, x_2) = \sigma(x_1^\top x_2)$$

This modification enhances the model's capacity to capture a broader spectrum of relationships between sentence pairs.