Natalia Rutecka

NLP - report of practical 2

Exercise 1

a) Using the fact that y is a one-hot vector with a 1 for the true outside word o, and 0 everywhere else:

$$-\sum_{w \in Vocab} y_w log(\hat{y_w}) = -\sum_{w \in Vocab: y_w=1} y_w log(\hat{y_w}) = -log(\hat{y}_o)$$

b)
$\frac{\partial}{\partial v_c} J_{naive-softmax}(v_c, o, U) =$
$-\frac{\partial}{\partial v_c} log(\frac{exp(u_o^T v_c)}{\sum_{w \in Vocab} exp(u_w^T v_c)}) =$
$-u_o^T + \frac{\partial}{\partial v_c} log(\sum_{w \in Vocab} exp(u_w^T v_c)) =$
$-u_o^T + \frac{1}{\sum_{w \in Vocab} exp(u_w^T v_c)} \sum_{w \in Vocab} exp(u_w^T v_c) \cdot u_w^T =$
$-u_o^T + \sum_{w \in Vocab} \hat{y}_w \cdot u_w^T =$
$-y^T U^T + \hat{y}^T U^T$

c) if $x = o$:
$\frac{\partial}{\partial u_x} J_{naive-softmax}(v_c, o, U) =$
$-\frac{\partial}{\partial u_x} log(\frac{exp(u_o^T v_c)}{\sum_{w \in Vocab} exp(u_w^T v_c)}) =$
$-v_c + \frac{\partial}{\partial u_x} log(\sum_{w \in Vocab} exp(u_w^T v_c)) =$
$-v_c + \frac{1}{\sum_{w \in Vocab} exp(u_w^T v_c)} \cdot exp(u_x^T v_c) \cdot v_c =$
$-v_c + \hat{y}_x^T \cdot v_c$

if $x \neq o$:
$\frac{\partial}{\partial u_x} J_{naive-softmax}(v_c, o, U) =$
$-\frac{\partial}{\partial u_x} log(\frac{exp(u_o^T v_c)}{\sum_{w \in Vocab} exp(u_w^T v_c)}) =$
$\frac{\partial}{\partial u_x} log(\sum_{w \in Vocab} exp(u_w^T v_c)) =$
$\frac{1}{\sum_{w \in Vocab} exp(u_w^T v_c)} \cdot exp(u_x^T v_c) \cdot v_c =$
$\hat{y}_x^T \cdot v_c$

In general $(\hat{y} - y) \cdot v_c^T$ is a matrix in which i-th row contains the partial derivative of $J_{naive-softmax}$ with respect to u_{w_i} where w_i is the i-th word in vocabulary.

d) $\frac{\partial}{\partial x} \frac{e^x}{e^x+1} = \frac{e^x(e^x+1)-e^{2x}}{(e^x+1)^2} = \sigma(x)(1-\sigma(x))$

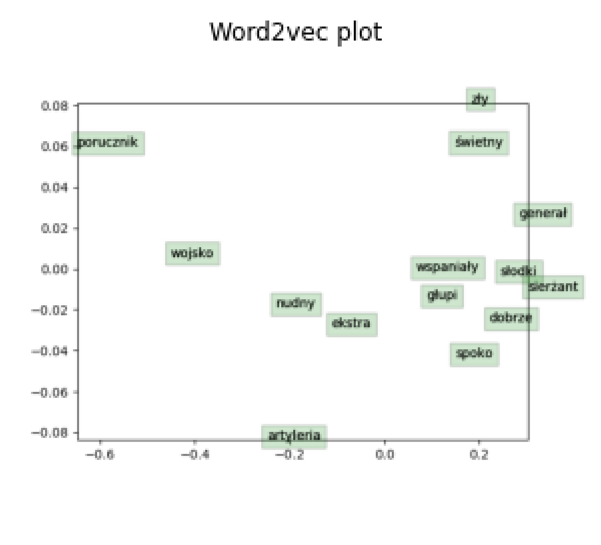
e)
$\frac{\partial}{\partial v_c} - log(\sigma(u_o^T v_c)) - \sum_{k=1..K} log(\sigma(-u_k^T v_c)) =$
$= -\frac{1}{\sigma(u_o^T v_c)} \cdot \sigma(u_o^T v_c) \cdot (1 - \sigma(u_o^T v_c)) \cdot u_o^T - \sum_{k=1..K} \frac{1}{\sigma(-u_k^T v_c)} \cdot \sigma(-u_k^T v_c) \cdot (1 - \sigma(-u_k^T v_c) \cdot (-u_k^T)) =$
$= (\sigma(u_o^T v_c) - 1) \cdot u_o^T - \sum_{k=1..K} (\sigma(-u_k^T v_c) - 1) \cdot u_k^T$
$\frac{\partial}{\partial u_o} - log(\sigma(u_o^T v_c)) - \sum_{k=1..K} log(\sigma(-u_k^T v_c)) = -\frac{1}{\sigma(u_o^T v_c)} \cdot \sigma(u_o^T v_c) \cdot (1 - \sigma(u_o^T v_c)) \cdot v_c = (\sigma(u_o^T v_c) - 1) \cdot v_c$
$\frac{\partial}{\partial u_w} - log(\sigma(u_o^T v_c)) - \sum_{k=1..K} log(\sigma(-u_k^T v_c)) = \frac{1}{\sigma(-u_w^T v_c)} \cdot \sigma(-u_w^T v_c) \cdot (1 - \sigma(-u_w^T v_c)) \cdot -v_c = (\sigma(-u_w^T v_c) - 1) \cdot v_c$

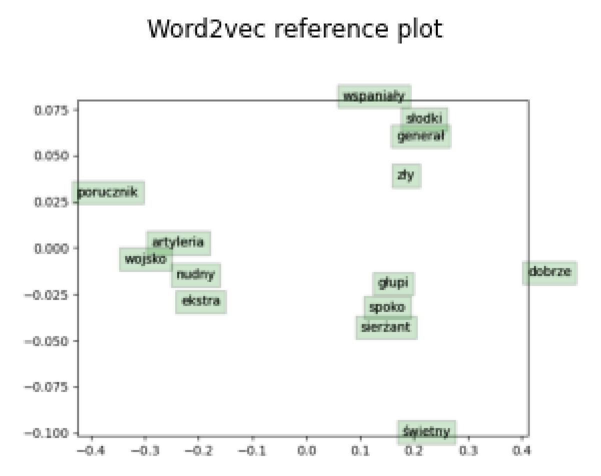
Negative sampling loss is much more efficient to compute than naive-softmax loss, because the formula uses only K sampled words, whereas naive softmax loss uses all words from the vocabulary.

f)
$\frac{\partial}{\partial U} J_{skip-gram}(v_c, w_{t-m}, \ldots, w_{t+m}, U) = \sum_{-m \leq j \leq m, j \neq 0} \frac{\partial}{\partial U} J_{skip-gram}(v_c, w_{t_j}, U)$
$\frac{\partial}{\partial v_c} J_{skip-gram}(v_c, w_{t-m}, \ldots, w_{t+m}, U) = \sum_{-m \leq j \leq m, j \neq 0} \frac{\partial}{\partial v_c} J_{skip-gram}(v_c, w_{t_j}, U)$
$\frac{\partial}{\partial v_w} J_{skip-gram}(v_c, w_{t-m}, \ldots, w_{t+m}, U) = 0$

Exercise 2

c) The plot of the word vectors is depicted below:





The plot is significantly different to the reference plot in terms of pairwise distances between word vectors, however both embeddings have the following properties.

The words connected to the military don't form a cluster, which seems counterintuive. Adjectives with opposite meaning tend to be close in the space, eg. "nudny" and "ekstra". Similarly, adjectives and adverbs with similar meaning tend to cluster together as well, eg. "wspaniały" and "słodki".

Exercise 3

a) Using m means that in each step of the optimiser, the size of the step depends not only on the current value of the gradient but also on the values of the gradient in previous steps, which has a smoothing effect on the loss curve. In general it makes the training more stable and helps the optimiser to get out of a local minimum if the momentum, i.e. the moving average of the gradients in the past, is large enough.

b) The model parameters that will get larger updates are the ones that, apart from having large gradients and momentum, did not consistently have large gradients in the previous steps. This modification prevents large oscillations in the directions where they might occur, which helps with finding local minima quicker.

c) The L2 regularisation helps to prevent overfitting, because it forces the model to not assign high values to single parameters by introducing a penalty term equal to the L2 norm of the parameter vector. L2 regularisation is helpful in low-data settings because then the models are most prone to overfitting.

When using L2 loss with Adam, the penalty term is added to the gradient before the calculation of the adaptive learning rates, which makes the shrinking effect of the L2 regularisation depend on the v vectors. The idea of AdamW optimiser is to decouple the weight decay from the optimisation steps taken with regard to the loss function, i.e. to use a regularisation factor directly in the update rule.

d) Q1: During evaluation we usually want to get a deterministic result of the neural network. The purpose of the dropout is to train multiple parts of the network to correctly recognise the same pattern. Thanks to that, using all neurons during evaluation acts as averaging the predictions of the well-trained parts, which has a regularising effect.