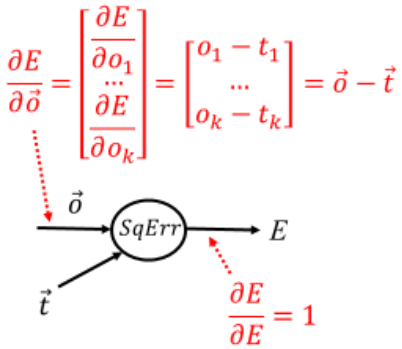
Exercises on text classification with Multi-Layer Perceptrons (MLPs)

Ion Androutsopoulos, 2025–26

Submit as a group of 2–3 members (unless specified otherwise in the lectures) a report (max. 10 pages, PDF format) for exercises 4 and 5. Include in your report all the required information, especially experimental results. Do not include code in the report, but include a link to a Colab notebook containing your code. Make sure to divide fairly the work of your group to its members and describe in your report the contribution of each member. The contribution of each member will also be checked during the oral examination of your submission. For delayed submissions, one point will be subtracted per day of delay.

1. Show that without activation functions, a multi-layer neural network is equivalent to applying a linear transformation to the input, i.e., the output can be written as $\vec{o} = W\vec{x} + b$, where W is a weights matrix, $b \in \mathbb{R}$ is a bias term, and $\vec{x} \epsilon \mathbb{R}^n$ is the input feature vector.

2. Confirm the computation of $\frac{\partial E}{\partial \vec{o}}$ in the computation graph of slide 19.



Answer: The gradient that we need to compute is:

$$\frac{\partial E}{\partial \vec{o}} = \begin{bmatrix} \frac{\partial E}{\partial o_1} \\ \dots \\ \frac{\partial E}{\partial o_i} \\ \dots \\ \frac{\partial E}{\partial o_k} \end{bmatrix}$$

Let us consider separately a single derivative $\frac{\partial E}{\partial o_i}$ (a single element of the gradient):

$$\frac{\partial E}{\partial o_i} = \frac{\partial}{\partial o_i} \sum_{j=1}^{k} \frac{1}{2} (t_j - o_j)^2 = \frac{\partial}{\partial o_i} \frac{1}{2} (t_i - o_i)^2 = \frac{1}{2} \cdot 2 \cdot (t_i - o_i) \cdot \frac{\partial}{\partial o_i} (t_i - o_i)$$
$$= (t_i - o_i) \cdot (-1) = (o_i - t_i)$$

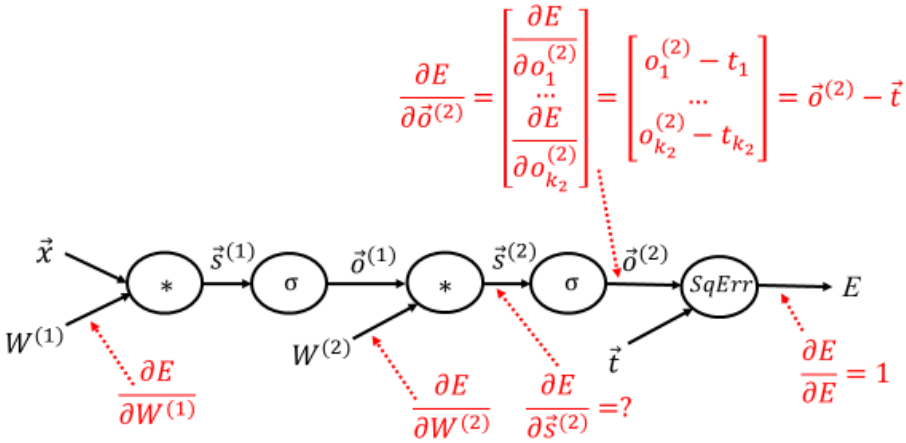
Hence:

$$\frac{\partial E}{\partial \vec{o}} = \begin{bmatrix} \frac{\partial E}{\partial o_1} \\ \cdots \\ \frac{\partial E}{\partial o_i} \\ \cdots \\ \frac{\partial E}{\partial o_k} \end{bmatrix} = \begin{bmatrix} o_1 - t_1 \\ \cdots \\ o_i - t_i \\ \cdots \\ o_k - t_k \end{bmatrix} = \vec{o} - \vec{t}$$

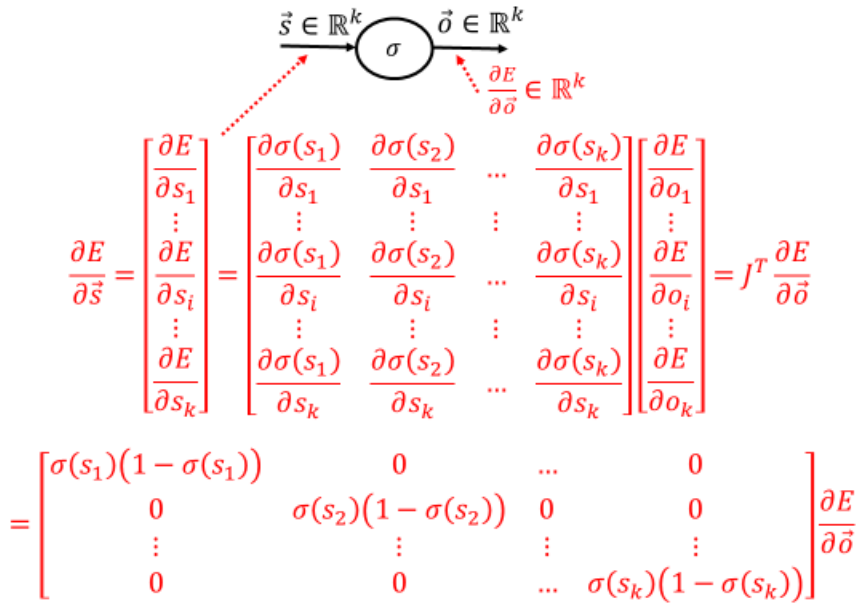
Note: We do not need to compute $\frac{\partial E}{\partial \vec{t}}$, because we do not update $\vec{t}$ (the correct prediction).

3. (i) Compute the gradient $\frac{\partial E}{\partial \vec{o}^{(2)}}$ in the network with the following computation graph.

Answer: The gradient $\frac{\partial E}{\partial \vec{o}^{(2)}}$ is computed as in Exercise 2.



(ii) Show that for a sigmoid node $\sigma(\vec{s}) = \vec{o}$, $\frac{\partial E}{\partial \vec{s}}$ can be computed as follows, where J is the Jacobian matrix.[1]



[1] See https://en.wikipedia.org/wiki/Jacobian_matrix_and_determinant.

Answer: The gradient that we need to compute is:

$$\frac{\partial E}{\partial \vec{s}} = \begin{bmatrix} \frac{\partial E}{\partial s_1} \\ \vdots \\ \frac{\partial E}{\partial s_i} \\ \vdots \\ \frac{\partial E}{\partial s_k} \end{bmatrix}$$

Let us consider separately a single derivative $\frac{\partial E}{\partial s_i}$ *(a single element of the gradient). By the chain rule of derivatives, we obtain:*

$$\frac{\partial E}{\partial s_i} = \sum_{j=1}^{k} \frac{\partial E}{\partial o_j} \frac{\partial o_j}{\partial s_i}$$

However, each s_i *affects only* $o_i = \sigma(s_i)$*. It does not affect any other* $o_j = \sigma(s_j)$*, for* $j \neq i$*. Hence,* $\frac{\partial o_j}{\partial s_i} = 0$ *for* $j \neq i$*, and we obtain:*

$$\frac{\partial E}{\partial s_i} = \frac{\partial E}{\partial o_i} \frac{\partial o_i}{\partial s_i} = \frac{\partial E}{\partial o_i} \frac{\partial \sigma(s_i)}{\partial s_i} = \frac{\partial E}{\partial o_i} \sigma(s_i)\big(1 - \sigma(s_i)\big)$$

where we have use the property of the sigmoid that $\frac{d\sigma(x)}{dx} = \sigma(x)\big(1 - \sigma(x)\big)$*.*

Therefore:

$$\frac{\partial E}{\partial \vec{s}} = \begin{bmatrix} \frac{\partial E}{\partial s_1} \\ \vdots \\ \frac{\partial E}{\partial s_i} \\ \vdots \\ \frac{\partial E}{\partial s_k} \end{bmatrix} = \begin{bmatrix} \frac{\partial E}{\partial o_1} \frac{\partial \sigma(s_1)}{\partial s_1} \\ \vdots \\ \frac{\partial E}{\partial o_i} \frac{\partial \sigma(s_i)}{\partial s_i} \\ \vdots \\ \frac{\partial E}{\partial o_k} \frac{\partial \sigma(s_k)}{\partial s_k} \end{bmatrix} = \begin{bmatrix} \frac{\partial E}{\partial o_1} \sigma(s_1)\big(1 - \sigma(s_1)\big) \\ \vdots \\ \frac{\partial E}{\partial o_i} \sigma(s_i)\big(1 - \sigma(s_i)\big) \\ \vdots \\ \frac{\partial E}{\partial o_k} \sigma(s_k)\big(1 - \sigma(s_k)\big) \end{bmatrix}$$

The latter can also be written as:

$$\frac{\partial E}{\partial \vec{s}} = \begin{bmatrix} \frac{\partial \sigma(s_1)}{\partial s_1} & 0 & \dots & 0 \\ 0 & \frac{\partial \sigma(s_2)}{\partial s_2} & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \dots & \frac{\partial \sigma(s_k)}{\partial s_k} \end{bmatrix} \begin{bmatrix} \frac{\partial E}{\partial o_1} \\ \frac{\partial E}{\partial o_2} \\ \vdots \\ \frac{\partial E}{\partial o_k} \end{bmatrix} =$$

$$= \begin{bmatrix} \sigma(s_1)\big(1 - \sigma(s_1)\big) & 0 & \dots & 0 \\ 0 & \sigma(s_2)\big(1 - \sigma(s_2)\big) & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \dots & \sigma(s_k)\big(1 - \sigma(s_k)\big) \end{bmatrix} \frac{\partial E}{\partial \vec{o}}$$

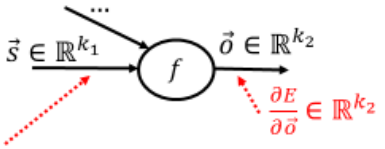
More generally, it can be written as:

$$\frac{\partial E}{\partial \vec{s}} = \begin{bmatrix} \frac{\partial \sigma(s_1)}{\partial s_1} & \frac{\partial \sigma(s_2)}{\partial s_1} & \dots & \frac{\partial \sigma(s_k)}{\partial s_1} \\ \frac{\partial \sigma(s_1)}{\partial s_2} & \frac{\partial \sigma(s_2)}{\partial s_2} & \dots & \frac{\partial \sigma(s_k)}{\partial s_2} \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\partial \sigma(s_1)}{\partial s_k} & \frac{\partial \sigma(s_2)}{\partial s_k} & \dots & \frac{\partial \sigma(s_k)}{\partial s_k} \end{bmatrix} \begin{bmatrix} \frac{\partial E}{\partial o_1} \\ \frac{\partial E}{\partial o_2} \\ \vdots \\ \frac{\partial E}{\partial o_k} \end{bmatrix} = J^T \frac{\partial E}{\partial \vec{o}}$$

where J is the Jacobian matrix:

$$J = \begin{bmatrix} \frac{\partial \sigma(s_1)}{\partial s_1} & \frac{\partial \sigma(s_1)}{\partial s_2} & \dots & \frac{\partial \sigma(s_1)}{\partial s_k} \\ \frac{\partial \sigma(s_2)}{\partial s_1} & \frac{\partial \sigma(s_2)}{\partial s_2} & \dots & \frac{\partial \sigma(s_2)}{\partial s_k} \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\partial \sigma(s_k)}{\partial s_1} & \frac{\partial \sigma(s_k)}{\partial s_2} & \dots & \frac{\partial \sigma(s_k)}{\partial s_k} \end{bmatrix}$$

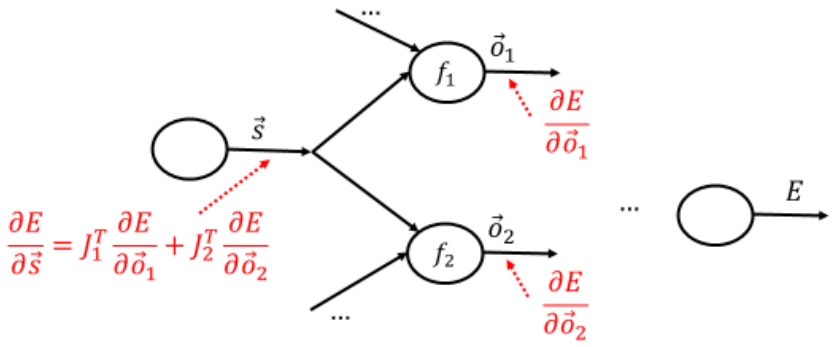
The latter applies more generally. For a node that computes $f(\vec{s}, \dots) = \vec{o}$, we can compute $\frac{\partial E}{\partial \vec{s}}$ as follows (provided that $\vec{s}$ is fed only to the f node):



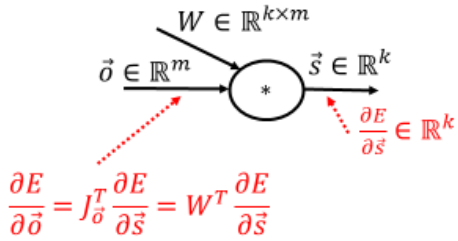
$$\frac{\partial E}{\partial \vec{s}} = \begin{bmatrix} \frac{\partial E}{\partial s_1} \\ \vdots \\ \frac{\partial E}{\partial s_i} \\ \vdots \\ \frac{\partial E}{\partial s_{k_1}} \end{bmatrix} = \begin{bmatrix} \frac{\partial o_1}{\partial s_1} & \frac{\partial o_2}{\partial s_1} & \dots & \frac{\partial o_{k_2}}{\partial s_1} \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\partial o_1}{\partial s_i} & \frac{\partial o_2}{\partial s_i} & \dots & \frac{\partial o_{k_2}}{\partial s_i} \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\partial o_1}{\partial s_{k_1}} & \frac{\partial o_2}{\partial s_{k_1}} & \dots & \frac{\partial o_{k_2}}{\partial s_{k_1}} \end{bmatrix} \begin{bmatrix} \frac{\partial E}{\partial o_1} \\ \vdots \\ \frac{\partial E}{\partial o_i} \\ \vdots \\ \frac{\partial E}{\partial o_{k_2}} \end{bmatrix} = J^T \frac{\partial E}{\partial \vec{o}}$$

(Check that this is also true for $\frac{\partial E}{\partial \vec{o}}$ in exercise 2.)

If $\vec{s}$ is fed to two (or more) nodes f_1, f_2, we have to add the gradients for $\frac{\partial E}{\partial \vec{s}}$ that we get from f_1, f_2:



(iii) Show that for a matrix-vector multiplication node $W\vec{o} = \vec{s}$, $\frac{\partial E}{\partial \vec{o}}$ can be computed as follows:



Answer:

$$\vec{s} = \begin{bmatrix} s_1 \\ s_2 \\ s_3 \\ \dots \\ s_k \end{bmatrix} = W\vec{o} = \begin{bmatrix} w_{1,1} & w_{2,1} & \dots & w_{m,1} \\ w_{1,2} & w_{2,2} & \dots & w_{m,2} \\ w_{1,3} & w_{2,3} & \dots & w_{m,3} \\ \dots & \dots & \dots & \dots \\ w_{1,k} & w_{2,k} & \dots & w_{m,k} \end{bmatrix} \begin{bmatrix} o_1 \\ o_2 \\ o_3 \\ \dots \\ o_m \end{bmatrix} = \begin{bmatrix} w_{1,1}o_1 + w_{2,1}o_2 + \cdots + w_{m,1}o_m \\ w_{1,2}o_1 + w_{2,2}o_2 + \cdots + w_{m,2}o_m \\ w_{1,3}o_1 + w_{2,3}o_2 + \cdots + w_{m,3}o_m \\ \dots \\ w_{1,k}o_1 + w_{2,k}o_2 + \cdots + w_{m,k}o_m \end{bmatrix}$$

The gradient that we need to compute is:

$$\frac{\partial E}{\partial \vec{o}} = \begin{bmatrix} \frac{\partial E}{\partial o_1} \\ \vdots \\ \frac{\partial E}{\partial o_i} \\ \vdots \\ \frac{\partial E}{\partial o_m} \end{bmatrix}$$

Let us consider separately a single derivative $\frac{\partial E}{\partial o_i}$ *(a single element of the gradient). By the chain rule of derivatives, we obtain:*

$$\frac{\partial E}{\partial o_i} = \sum_{j=1}^{k} \frac{\partial E}{\partial s_j} \frac{\partial s_j}{\partial o_i}$$

According to the equations for $\vec{s} = W\vec{o}$ *above:*

$$s_j = w_{1,j}o_1 + w_{2,j}o_2 + \cdots + w_{i,j}o_i + \cdots + w_{m,j}o_m$$

Hence:

$$\frac{\partial s_j}{\partial o_i} = w_{i,j}$$

Therefore:

$$\frac{\partial E}{\partial o_i} = \sum_{j=1}^{k} \frac{\partial E}{\partial s_j} \frac{\partial s_j}{\partial o_i} = \sum_{j=1}^{k} \frac{\partial E}{\partial s_j} w_{i,j}$$

which can also be written as:

$$\frac{\partial E}{\partial o_i} = \begin{bmatrix} \frac{\partial s_1}{\partial o_i} & \frac{\partial s_2}{\partial o_i} & \dots & \frac{\partial s_k}{\partial o_i} \end{bmatrix} \begin{bmatrix} \frac{\partial E}{\partial s_1} \\ \frac{\partial E}{\partial s_2} \\ \vdots \\ \frac{\partial E}{\partial s_k} \end{bmatrix} = \begin{bmatrix} w_{i,1} & w_{i,2} & \dots & w_{i,k} \end{bmatrix} \begin{bmatrix} \frac{\partial E}{\partial s_1} \\ \frac{\partial E}{\partial s_2} \\ \vdots \\ \frac{\partial E}{\partial s_k} \end{bmatrix}$$

Hence, for the overall gradient:

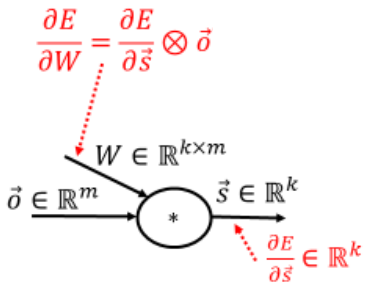
$$\frac{\partial E}{\partial \vec{o}} = \begin{bmatrix} \frac{\partial E}{\partial o_1} \\ \frac{\partial E}{\partial o_2} \\ \vdots \\ \frac{\partial E}{\partial o_i} \\ \vdots \\ \frac{\partial E}{\partial o_m} \end{bmatrix} = \begin{bmatrix} \frac{\partial s_1}{\partial o_1} & \frac{\partial s_2}{\partial o_1} & \dots & \frac{\partial s_k}{\partial o_1} \\ \frac{\partial s_1}{\partial o_2} & \frac{\partial s_2}{\partial o_2} & \dots & \frac{\partial s_k}{\partial o_2} \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\partial s_1}{\partial o_i} & \frac{\partial s_2}{\partial o_i} & \dots & \frac{\partial s_k}{\partial o_i} \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\partial s_1}{\partial o_m} & \frac{\partial s_2}{\partial o_m} & \dots & \frac{\partial s_k}{\partial o_m} \end{bmatrix} \begin{bmatrix} \frac{\partial E}{\partial s_1} \\ \frac{\partial E}{\partial s_2} \\ \vdots \\ \frac{\partial E}{\partial s_k} \end{bmatrix} = \begin{bmatrix} w_{1,1} & w_{1,2} & \dots & w_{1,k} \\ w_{2,1} & w_{2,2} & \dots & w_{2,k} \\ \vdots & \vdots & \vdots & \vdots \\ w_{i,1} & w_{i,2} & \dots & w_{i,k} \\ \vdots & \vdots & \vdots & \vdots \\ w_{m,1} & w_{m,2} & \dots & w_{m,k} \end{bmatrix} \begin{bmatrix} \frac{\partial E}{\partial s_1} \\ \frac{\partial E}{\partial s_2} \\ \vdots \\ \frac{\partial E}{\partial s_k} \end{bmatrix} =$$

$$J_{\vec{o}}^{T} \frac{\partial E}{\partial \vec{s}} = W^{T} \frac{\partial E}{\partial \vec{s}}$$

Note: We prefer to use matrix operators, which can be efficiently computed using highly optimized algorithms and GPUs, rather than relying on our own for-loops (e.g., in our own Python scripts) to compute individual elements of matrices, which is much slower.

(iv) Show that for a matrix-vector multiplication node $W\vec{o} = \vec{s}$, $\frac{\partial E}{\partial W} = \frac{\partial E}{\partial \vec{s}} \otimes \vec{o}$, where $\otimes$ denotes the outer product.[2]

[2] See https://en.wikipedia.org/wiki/Matrix_multiplication#Outer_product.



Answer: Recall that we use the following notation for the elements of W:

$$W = \begin{bmatrix} w_{1,1} & w_{2,1} & \dots & w_{m,1} \\ w_{1,2} & w_{2,2} & \dots & w_{m,2} \\ w_{1,3} & w_{2,3} & \dots & w_{m,3} \\ \dots & \dots & \dots & \dots \\ w_{1,k} & w_{2,k} & \dots & w_{m,k} \end{bmatrix}$$

The gradient that we need to compute is:

$$\frac{\partial E}{\partial W} = \begin{bmatrix} \frac{\partial E}{\partial w_{1,1}} & \frac{\partial E}{\partial w_{2,1}} & \dots & \frac{\partial E}{\partial w_{m,1}} \\ \frac{\partial E}{\partial w_{1,2}} & \frac{\partial E}{\partial w_{2,2}} & \dots & \frac{\partial E}{\partial w_{m,2}} \\ \frac{\partial E}{\partial w_{1,3}} & \frac{\partial E}{\partial w_{2,3}} & \dots & \frac{\partial E}{\partial w_{m,3}} \\ \dots & \dots & \dots & \dots \\ \frac{\partial E}{\partial w_{1,k}} & \frac{\partial E}{\partial w_{2,k}} & \dots & \frac{\partial E}{\partial w_{m,k}} \end{bmatrix}$$

Let us consider separately a single derivative $\frac{\partial E}{\partial w_{i,j}}$ (a single element of the gradient). By the chain rule of derivatives, we obtain:

$$\frac{\partial E}{\partial w_{i,j}} = \sum_{l=1}^{k} \frac{\partial E}{\partial s_l} \frac{\partial s_l}{\partial w_{i,j}}$$

According to the equations for $\vec{s} = W\vec{o}$ in part (iii) of the exercise:

$$s_l = w_{1,l}o_1 + w_{2,l}o_2 + \dots + w_{i,l}o_i + \dots + w_{m,l}o_m$$

Hence:

$$\frac{\partial s_l}{\partial w_{i,j}} = 0, \text{for } l \neq j$$

and:

$$\frac{\partial E}{\partial w_{i,j}} = \sum_{l=1}^{k} \frac{\partial E}{\partial s_l} \frac{\partial s_l}{\partial w_{i,j}} = \frac{\partial E}{\partial s_j} \frac{\partial s_j}{\partial w_{i,j}}$$

Given that:

$$s_j = w_{1,j}o_1 + w_{2,j}o_2 + \dots + w_{i,j}o_i + \dots + w_{m,j}o_m$$

we obtain:

$$\frac{\partial s_j}{\partial w_{i,j}} = o_i$$

Hence:

$$\frac{\partial E}{\partial w_{i,j}} = \frac{\partial E}{\partial s_j}\frac{\partial s_j}{\partial w_{i,j}} = \frac{\partial E}{\partial s_j} o_i$$

Going back to the overall gradient:

$$\frac{\partial E}{\partial W} = \begin{bmatrix} \frac{\partial E}{\partial w_{1,1}} & \frac{\partial E}{\partial w_{2,1}} & \cdots & \frac{\partial E}{\partial w_{m,1}} \\ \frac{\partial E}{\partial w_{1,2}} & \frac{\partial E}{\partial w_{2,2}} & \cdots & \frac{\partial E}{\partial w_{m,2}} \\ \frac{\partial E}{\partial w_{1,3}} & \frac{\partial E}{\partial w_{2,1}} & \cdots & \frac{\partial E}{\partial w_{m,3}} \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\partial E}{\partial w_{1,k}} & \frac{\partial E}{\partial w_{2,k}} & \cdots & \frac{\partial E}{\partial w_{m,k}} \end{bmatrix} = \begin{bmatrix} \frac{\partial E}{\partial s_1} o_1 & \frac{\partial E}{\partial s_1} o_2 & \cdots & \frac{\partial E}{\partial s_1} o_m \\ \frac{\partial E}{\partial s_2} o_1 & \frac{\partial E}{\partial s_2} o_2 & \cdots & \frac{\partial E}{\partial s_2} o_m \\ \frac{\partial E}{\partial s_3} o_1 & \frac{\partial E}{\partial s_3} o_2 & \cdots & \frac{\partial E}{\partial s_3} o_m \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\partial E}{\partial s_k} o_1 & \frac{\partial E}{\partial s_k} o_2 & \cdots & \frac{\partial E}{\partial w_{s_k}} o_m \end{bmatrix} =$$

$$= \begin{bmatrix} \frac{\partial E}{\partial s_1} \\ \frac{\partial E}{\partial s_2} \\ \frac{\partial E}{\partial s_3} \\ \vdots \\ \frac{\partial E}{\partial s_k} \end{bmatrix} [o_1 \quad o_2 \quad \cdots \quad o_m] = \frac{\partial E}{\partial \vec{s}} \otimes \vec{o}$$

→ **4.** Repeat exercise 11 of Part 2 (text classification with mostly linear classifiers), now using an MLP classifier implemented (by you) in PyTorch.[3] You may use different features in the MLP classifier than the ones you used in exercise 11 of Part 2. Tune the hyper-parameters (e.g., number of hidden layers, dropout probability) on the development subset of your dataset. Monitor the performance of the MLP on the development subset during training to decide how many epochs to use. Include experimental results of a baseline majority classifier, as well as experimental results of your best classifier from exercise 11 of Part 2, now treated as a second baseline. Include in your report:

- Curves showing the loss on training and development data as a function of epochs (slide 42).
- Precision, recall, F1, precision-recall AUC scores, for each class and classifier, separately for the training, development, and test subsets, as in exercise 11 of Part 2.
- Macro-averaged precision, recall, F1, precision-recall AUC scores (averaging the corresponding scores of the previous bullet over the classes), for each classifier, separately for the training, development, and test subsets, as in exercise 11 of Part 2.
- A short description of the methods and datasets you used, including statistics about the datasets (e.g., average document length, number of training/dev/test documents, vocabulary size) and a description of the preprocessing steps that you performed.

[3] See http://pytorch.org/.

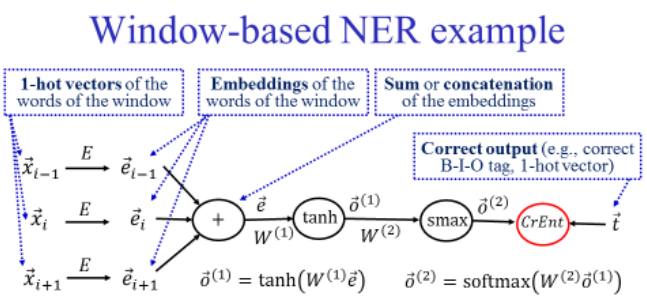
You may optionally wish to try ensembles. One possibility is to use k separate MLP classifiers, corresponding to your k best checkpoints (k best epochs in terms of development loss), and aggregate their decisions by majority voting. Another possibility is to use temporal averaging, i.e., use a single MLP classifier, whose weights are the average of the weights of the k best checkpoints.

→ **5.** Develop a part-of-speech (POS) tagger for one of the languages of the Universal Dependencies treebanks (http://universaldependencies.org/), using an MLP (implemented by you) operating on windows of words (slides 27–28). Consider only the words, sentences, and POS tags of the treebanks (not the dependencies or other annotations). Use PyTorch to implement the MLP. You may use any types of word features you prefer, but it is recommended to use pre-trained word embeddings. Make sure that you use separate training, development, and test subsets. Tune the hyper-parameters (e.g., number of hidden layers, dropout probability) on the development subset. Monitor the performance of the MLP on the development subset during training to decide how many epochs to use. Include experimental results of a baseline that tags each word with the most frequent tag it had in the training data; for words that were not encountered in the training data, the baseline should return the most frequent tag (over all words) of the training data. Include in your report:

- Curves showing the loss on training and development data as a function of epochs (slide 42).
- Precision, recall, F1, precision-recall AUC scores, for each class (tag) and classifier, separately for the training, development, and test subsets, as in exercise 11 of Part 2.
- Macro-averaged precision, recall, F1, precision-recall AUC scores (averaging the corresponding scores of the previous bullet over the classes), for each classifier, separately for the training, development, and test subsets, as in exercise 11 of Part 2.
- A short description of the methods and datasets you used, including statistics about the datasets (e.g., average sentence length, number of training/dev/test sentences and words, vocabulary size) and a description of the preprocessing steps.

You may optionally wish to try ensembles, as in exercise 4 above.

6. (a) We use the window-based neural network named entity recognizer (NER) of the slide on the right, with 300-dimensional word embeddings, to recognize three types of named entities (persons, organizations, locations). We use B-I-O tags (BPerson, IPerson, BOrganization etc., with a single O tag). The size of the vocabulary is $|V| = 100{,}000$. The "+" node concatenates the embeddings of the three words in the window. The hidden layer contains 500 neurons (with *tanh* activation functions). What are the dimensions of matrices $E, W^{(1)}, W^{(2)}$? Fully justify your answers.



7. [**optional**] Repeat exercise 3 of Part 1 (n-gram language models) now using an MLP language model, instead of an n-gram language model. The MLP takes as input the concatenation of the word embeddings of the n previous words, and outputs a probability distribution over the vocabulary as a prediction for the next word. Compare the results you obtained using the MLP language model to those you had obtained with the bigram and trigram language models of Part 1.