Arnaud Massenet
arnaudyoh@gmail.com

Lab session # 1

Deep Learning MVA 2020

22/02/20

1 Monolingual embeddings

See the **nlp_project.ipynb**.

2 Multilingual word embeddings

Question 1: Values of W^*

We want to find $W^* = \text{argmin}_{W \in O_d(\mathcal{R})} ||WX - Y||_F$. For notation purpose we will assume that $W \in O_d(\mathcal{R})$. First we observe that,

$$\begin{aligned} ||WX - Y||^2 &=< WX, WX > -2 < WX, Y > +||Y||^2 \\ &=< W^T WX, X > -2 < WX, Y > +||Y||^2 \\ &= ||X||^2 - 2 < WX, Y > +||Y||^2, \quad \text{since } W^T W = I_d \end{aligned}$$

Thus $\text{argmin}_W \, ||WX - Y||_F = \text{argmax}_W \, 2 < WX, Y >= \text{argmax}_W \, trace(WXY^T)$.

We introduce the singular value decomposition (SVD) of W and XY^T.

$$\begin{aligned} SVD(W) &= U_w \Sigma_w V_w^T \\ SVD(YX^T) &= U \Sigma V^T \end{aligned}$$

Now we can go back to our trace:

$$\begin{aligned} trace(WYX^T) &= trace(U_w \Sigma_w V_w^T (U \Sigma V^T)^T) \\ &= trace(U_w \Sigma_w V_w^T V \Sigma U^T) \\ &= trace(U^T U_w \Sigma_w V_w^T V \Sigma \\ &\leq trace(\Sigma_w \Sigma) \text{By Von Neumann theorem with } F = XY^T \text{ and } G = W \end{aligned}$$

Recall that the Von Neumann theorem states that, for real valued mxn matrices F and G, and σ_i the function return the i^{th} the largest singular value of a matrix:

$$trace(F^T G) \leq \sum_{i=1} \sigma_i(F) \sigma_i(G)$$

Thus, $trace(WYX^T)$ is maximized when $U^T U_w = V_w V = I_d$. By orthogonality of V, V_w, U and U_w, that would mean that $U^T = U_w$ and $V_w = V$. As such $W = U \Sigma_w V$ and since $W^T W = I_d$ $\Sigma_w = I_d$. Thus the solution to our minimization problem is $W = UVT$

3 Sentence classification with BoW

Question 1: Comparing average and weighted average

In table 1 we have the results for the best model for both average word vector and weighted average. "Best model" is defined as the model with the highest validation score (on dev set). Interestingly the weighted average did not improve the result. In fact, it helps to the improve the train accuracy by the validation accuracy essentially stays the same.

		Accuracy (%)
Train	Avg	45.72
	W Avg	47.00
Dev	Avg	40.42
	W Avg	40.69

Table 1: Accuracy on train and dev set with using average of vector vs. weighted average

4 Deep Learning models for classification

Question 1: Loss definition

Since we are working on a multi-label classification problem, we naturally turned to the categorical cross-entropy loss function. Based on our 5 labels, we can write the loss as

$$H(y, \hat{y}) = -\frac{1}{N}\sum_{i=1}^{N}\sum_{c=1}^{5}\hat{y}^c \log(y_i^c)$$

where $\hat{y}$ and y are the one-hot encoded prediction and truth vectors and N is the number of samples.

Question 2: Plots w.r.t. Epochs

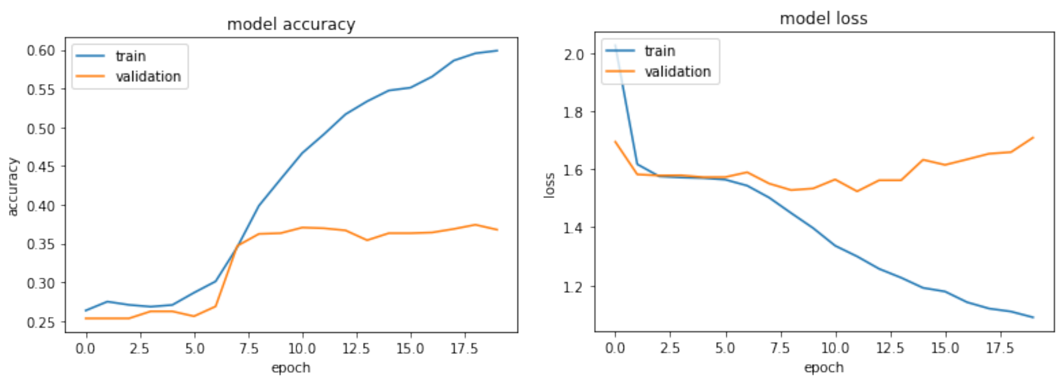

Figure 1: Accuracy and Loss on the Train and Validation set

In Fig. 2 we have the results from the LSTM model. We can see that the model has a tendency to overfit in less than 6 epochs despite the high dropout probabilities that we used (p=0.7).

Question 3: Using another encoder

We replaced the trainable embedding with the word2vec embedding using a non-trainable embedding layer. We added an extra dense layer before the output layer in order to add more non-linearity and we used dropout with probability 0.6 in the LSTM and after the extra layer. We thought of using a bidirectional LSTM, but it only led to more overfitting.

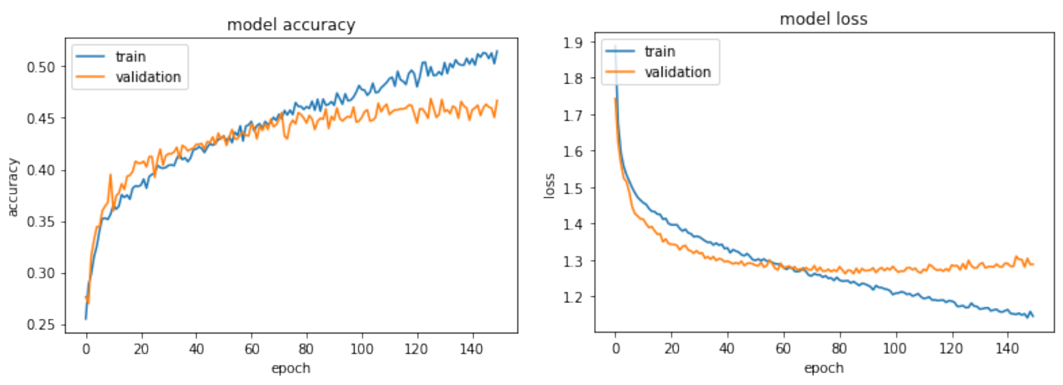

Figure 2: Accuracy and Loss on the Train and Validation set on our model

Observe that while there a tendency for overfitting after 80 epochs, our new model generalizes much better overall. After 100 epochs, we reached an accuracy of 48.71% for the train set and 45.03% for the validation (dev) set.