Deep Learning for Natural Language Processing

Léo Alberge

January 11, 2019

Multilingual word embeddings

Using the orthogonality and the properties of the trace, prove that:

for X and Y two matrices:

$$W^* = \underset{W \in \mathbf{O^d}}{\text{argmin}} \, ||WX - Y||_F$$

We have that:

$$\begin{aligned} ||WX - Y||_F &= \text{tr}((WX - Y)^T (WX - Y)) \\ &= \text{tr}(X^T W^T W X) + tr(Y^T Y) - 2tr(X^T W^T Y) \\ &= \text{tr}(X^T X) + \text{tr}(Y^T Y) - 2\text{tr}(X^T W^T Y) \\ &= \text{tr}(X^T X) + \text{tr}(Y^T Y) - 2\text{tr}(Y X^T W) \end{aligned}$$

since $W \in \mathbf{O^d}$ and with the trace properties. We can then ignore the 2 terms without W and we are left with minimizing $-2\text{tr}(YX^T W)$ over $\mathbf{O^d}$. We can use the SVD decomposition of $YX^T = U^T$.

$$-2\text{tr}(YX^T W) = -2\text{tr}(U\Sigma V^T W)$$

and the property of the trace for orthogonal matrices: For $L \in \mathbf{O^d}$ and $\Sigma \in \mathbf{S_{n+}}$, we have $\text{tr}(L\Sigma) \leq \text{tr}(\Sigma)$. We apply this property and we have that:

$$-2\text{tr}(YX^T W) \geq -2\text{tr}(\Sigma)$$

with equality when $W = UV^T$. It shows that $W^* = UV^T$ achieves the minimum.

Sentence classification with BoV

What is your training and dev errors using either the average of word vectors or the weighted-average?

- Average of word vectors best scores:

Training Error	Validation Error
0.512	0.445

- Weighted average of word vectors(IDF) scores:

Training Error	Validation Error
0.462	0.428

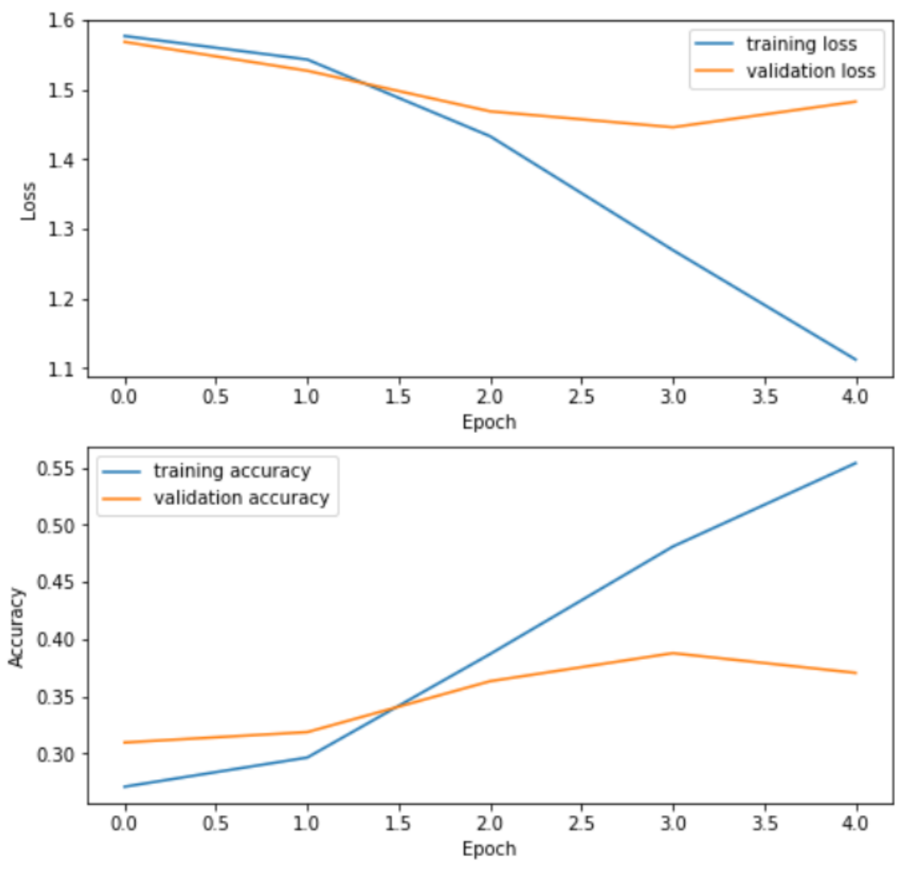

Figure 1: Evolution of train/validation results w.r.t the number of epochs

Deep Learning models for classification

Which loss did you use? Write the mathematical expression of the loss you used for the 5-class classification

I have used the categorical cross-entropy loss for this 5-classes classification problem. We note $k = 1..5$ ($K = 5$) the different classes, y the one-hot encoded label vector and p_k denotes the predicted probability of class k. For one sample, the loss writes:

$$L = -\sum_{k=1}^{K} y_k \log(p_k)$$

Plot the evolution of train/validation results w.r.t the number of epochs.

See Figure 1.

Be creative: use another encoder. Make it work! What are your motivations for using this other model?

For this part, I decided to use a similar model to the previous question but to use a pre-trained embedding layer instead of learning this layer like in the previous section. The embedding layer number of parameters is very high which makes it hard to train with the amount of data we have. Using a pre-trained embedding should help having already good embeddings inputs for the LSTM.