Project 2 - Deep Learning for Natural Language Processing

Yonatan DELORO

January 12, 2019

1 Monolingual embeddings

No question. CF. code

2 Multilingual word embeddings

- To prove the assertion, let us rewrite more simply the problem of finding W minimizing $||WX-F||_F$ under the constraint $W \in O_d(\mathbb{R})$. First of all, it is clearly equivalent to minimizing $f(W) := ||WX - F||_F^2$ under the same constraint.

  Using the definition of the Frobenius norm and denoting Tr the Trace operator, we can rewrite the objective function as below :

  $$\begin{aligned} f(W) &= Tr((WX - Y)(WX - Y)^T) \\ &= Tr(WXX^TW^T) - Tr(WXY^T) - Tr(YX^TW^T) + cste \end{aligned}$$

  Using the property of the trace ($Tr(AB) = Tr(BA)$), we have :

  $$Tr(WXX^TW^T) = Tr(XX^TW^TW) = Tr(XX^T)$$

  since $W^TW = I_d$ for any $W \in O_d(\mathbb{R})$.

  Thus,

  $$\begin{aligned} f(W) &= -Tr(WXY^T) - Tr(YX^TW^T) + cste \\ &= -Tr((WXY^T)^T) - Tr(YX^TW^T) + cste \\ &= -Tr((2.YX^T)W^T) + cste \end{aligned}$$

  Therefore, the original problem is equivalent to maximizing $Tr(ZW^T)$ under the constraint $W \in O_d(\mathbb{R})$, with $Z = 2.YX^T$.

- Let $U\Sigma V^T$ be the SVD decomposition of YX^T and introduce $R = Diag(\sqrt{\sigma_1}, ..., \sqrt{\sigma_d})$ the square root of Σ (denoting σ_i the singular values of YX^T).

  Applying the Cauchy Swartz inequality, we get for any $W \in O_d(\mathbb{R})$:

  $$\begin{aligned} Tr(ZW^T) = Tr(UR^2V^TW) &= Tr((UR)(RV^TW^T)) \\ &\leq ||UR||_F||RV^TW^T||_F \\ &= ||R||_F||R||_F = Tr(RR^T) = Tr(R^2) = Tr(\Sigma) \end{aligned}$$

  ($V^TW^T \in O_d(\mathbb{R})$ since $V^T, W^T \in O_d(\mathbb{R})$ and $O_d(\mathbb{R})$ is a multiplicative group)

  It is also clear to see that the maximum is reached when $W^* = UV^T$ as shown below :

  $$Tr(ZW^{*T}) = Tr(UR^2V^TVU^T) = Tr(R^2V^TVU^TU) = Tr(R^2)$$

  (as $U, V \in O_d(\mathbb{R})$)

Therefore, we proved that :

$$\mathbf{argmin_{W \in O_d(\mathbb{R})} ||WX - F||_F = UV^T \text{ with } U\Sigma V^T = SVD(YX^T)}$$

3 Sentence classification with BoV

Question

Note : Classification error was computed as 1 minus the classification accuracy.

	Train (classification error, cross-entropy loss)	Dev (classification error, cross-entropy loss)
Average of word vectors	(52.0%,1.20)	(57.3%, 1.29)
Idf-Weighted Average	(53.0%, 1.22)	(57.3%, 1.32)

Table 1: Results for sentence classification using Logistic Regression on top of bag-of-words embeddings, with l2-regularization (regularization strength of 2 for simple average of word vectors, and of 90 for Idf-weighted average, values found using grid-search to minimize the cross-entropy loss on the dev set).

Using Idf-weighted average did not really improve the results, maybe would we see an improvement for larger set of sentences (better estimations of words scarcities). However, performing a Multi-Class SVM on the top of the embeddings instead of a Logistic Regression improved a bit the classification score on the train and on the dev set.

	Train (classification error)	Dev (classification error)
Idf-Weighted Average	43.6%	56.5%

Table 2: Results for sentence classification using Multi-Class SVM on top of bag-of-words embeddings (penalty for outlier/misclassified point equal to 6 minimized the dev error in our grid search)

4 Deep Learning models for classification

Question 1

I used a categorical cross-entropy loss, which is adequate for the task (network outputs vectors encoding probabilities for each sentence to belong to each class, and labels to predict being hot-encoded).

Let us denote N the number of sentences, x_i the hot encoding of the i-th sentence, y_i its label, and $\hat{p_i}$ the network output prediction.
With these notations, the loss expresses as follows :

$$\mathcal{L}(\mathbf{x_i}, \mathbf{y_i}) = -\sum_{\mathbf{i=1}}^{\mathbf{N}} \sum_{\mathbf{k=1}}^{\mathbf{5}} \mathbf{1}_{\mathbf{y_i=k}} \log(\mathbf{\hat{p}_i})$$

If we want to express it in terms of the embeddings $\phi_\theta(x_i)$ obtained after the LSTM layer, we can introduce $(w_k)_{1 \leq k \leq 5}$ the columns of the weights matrix of the logistic regression done on the top of these embeddings (thanks to the dense layer and the softmax activation) :

$$P_w(y = k | \phi_\theta(x)) = \frac{\exp \phi_\theta(x)^T w_k}{\sum_{l=1}^{5} \exp \phi_\theta(x)^T w_l}$$

With these notations, the loss expresses as follows :

$$\mathcal{L}(x_i, y_i, \theta, w) = -\sum_{i=1}^{N} \sum_{k=1}^{5} 1_{y_i=k} \log P_w(y = k | \phi_\theta(x_i))$$

$$\mathcal{L}(\mathbf{x_i}, \mathbf{y_i}, \theta, \mathbf{w}) = -\sum_{\mathbf{i=1}}^{\mathbf{N}} \sum_{\mathbf{k=1}}^{\mathbf{5}} \mathbf{1}_{\mathbf{y_i=k}} \phi_\theta(\mathbf{x_i})^{\mathbf{T}} \mathbf{w_k} + \sum_{\mathbf{i=1}}^{\mathbf{N}} \log \left(\sum_{\mathbf{k=1}}^{\mathbf{5}} \exp \phi_\theta(\mathbf{x_i})^{\mathbf{T}} \mathbf{w_k} \right)$$

Question 2

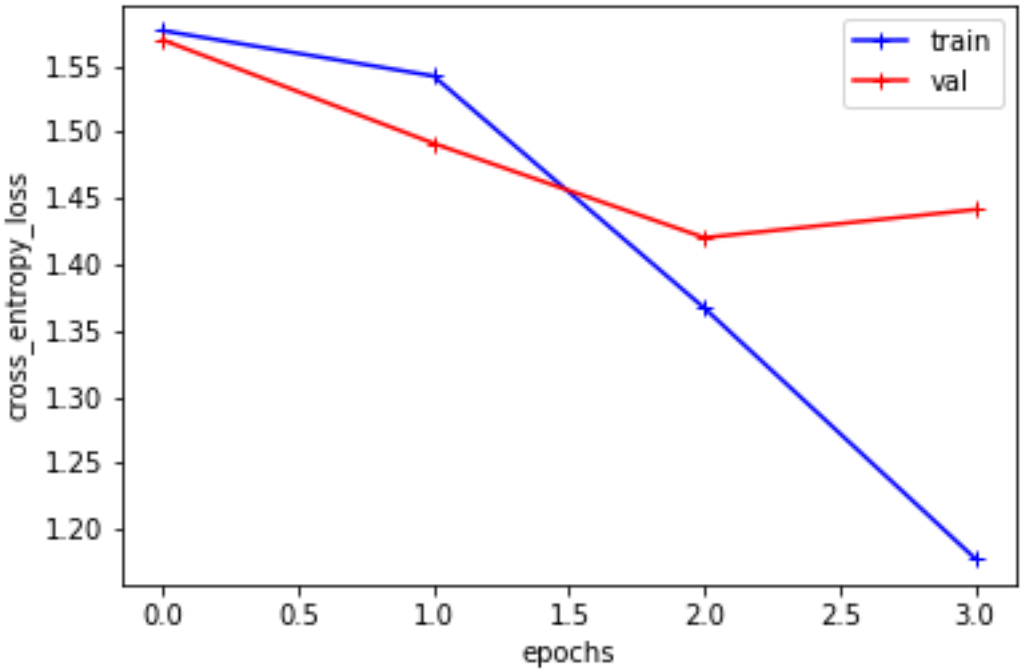

Figure 1: Results for sentence classification using Logistic Regression on the top of LSTM. We stopped the training before overfitting using keras early stopping implementation (the error on the dev set will increase even more after the fourth epoch). Increasing the dropout probabilities of the LSTM layer enabled to improve a bit performances, but we did not manage to find a solution enough powerful against overfitting to beat the performances of the BoW-based classifier.

Question 3

One could maybe try a Multi-Class SVM on the outputs of the LSTM Layer. However, I did not find time to investigate more on the topic with the many projects for January.... But this TP was very interesting, thanks !