Deep Learning for Natural Language Processing - Project

LAURES Benoît

January 11, 2019

1 Monolingual embeddings

Nothing to add

2 Multilingual word embeddings

Let X and Y 2 real matrices such that $U\Sigma V^T = SVD(YX^T)$.

$$\begin{aligned}
\min_{W \epsilon\ O_d(\mathbb{R})} \|WX - Y\|_F &= \min_{W \epsilon\ O_d(\mathbb{R})} \|WX - Y\|_F^2 \\
&= \min_{W \epsilon\ O_d(\mathbb{R})} \|WX\|_F^2 + \|Y\|_F^2 - 2\langle WX, Y\rangle \\
&= \min_{W \epsilon\ O_d(\mathbb{R})} \|X\|_F^2 + \|Y\|_F^2 - 2\langle WX, Y\rangle \\
&= \max_{W \epsilon\ O_d(\mathbb{R})} \langle WX, Y\rangle
\end{aligned}$$

We use Frobenius inner product and associated norm for that. Let's remind that U, V are orthogonal matrix, so they preserve the norm.

$$\begin{aligned}
\min_{W \epsilon\ O_d(\mathbb{R})} \|WX - Y\|_F &= \min_{W \epsilon\ O_d(\mathbb{R})} \|WX - Y\|_F^2 \\
&= tr(YX^TW^T) \\
&= tr(U\Sigma V^TW^T) \\
&= tr(\Sigma V^TW^TU) \\
&= tr(\Sigma Z) \\
&= \sum_{i=1}^{d} \Sigma_{ii} Z_{ii} \\
&\leqslant \sum_{i=1}^{d} \Sigma_{ii}
\end{aligned}$$

with $Z = V^TW^TU$ and because Z is an orthogonal matrix by product so the maximum is achieved when $Z = I$.

Thus $\underset{\mathbf{W}\epsilon\ \mathbf{O_d}(\mathbb{R})}{\mathbf{argmin}} \|\mathbf{WX} - \mathbf{Y}\|_{\mathbf{F}} = \mathbf{W}^* = \mathbf{UV^T}$

3 Sentence classification with BoV

Using the logistic regression model with $C = 1.623776739188721$, I got:

Average of word vectors:

- Training accuracy = 0.485
- Dev accuracy = 0.433

Weighted-average (IDF only computed on the training set):

- Training accuracy = 0.485
- Dev accuracy = 0.426

4 Deep Learning models for classification

4.1 Loss

I used the loss **sparse categorical crossentropy** because we want to measure the probability error in discrete classification tasks in which the classes are mutually exclusive (each entry is in exactly one class) and targets are integers (not one-hot encoded).
Let's note B the minibatch, the loss is :

$$\frac{-1}{|B|} \sum_{(\mathbf{x},y)\epsilon B} log\left(\frac{exp(s_y)}{exp(s_k)}\right)$$

4.2 Learning curve

This plot (Fig 1) suggests that the model deeply overfits the training set, however I didn't manage to limit this overfit (even by adding noise to training data, increasing dropout or adding a BN layer).

4.3 Another encoder

I got inspired by this Keras blog but modified a bit the architecture of the network to improve the performance. To make it work, you'll also have to download the GloVe embeddings vectors.

Also, because CUDA (for GPU computations) doesn't run properly on Jupyter Notebook, I had to write the code for this question in the file `question43.py`.

I use Conv 1D + LSTM + Dense layers. Conv 1D is useful to take care of neighboring words.
You'll find the learning curve below (Fig 2).

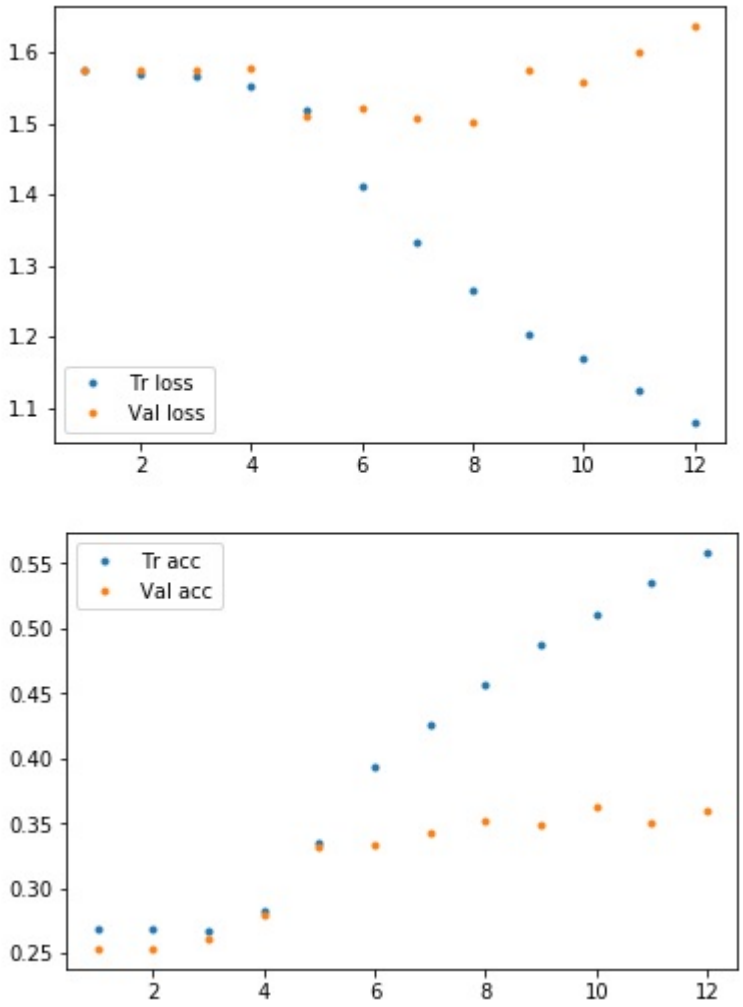

Figure 1: Learning curve for LSTM

curve 2.png

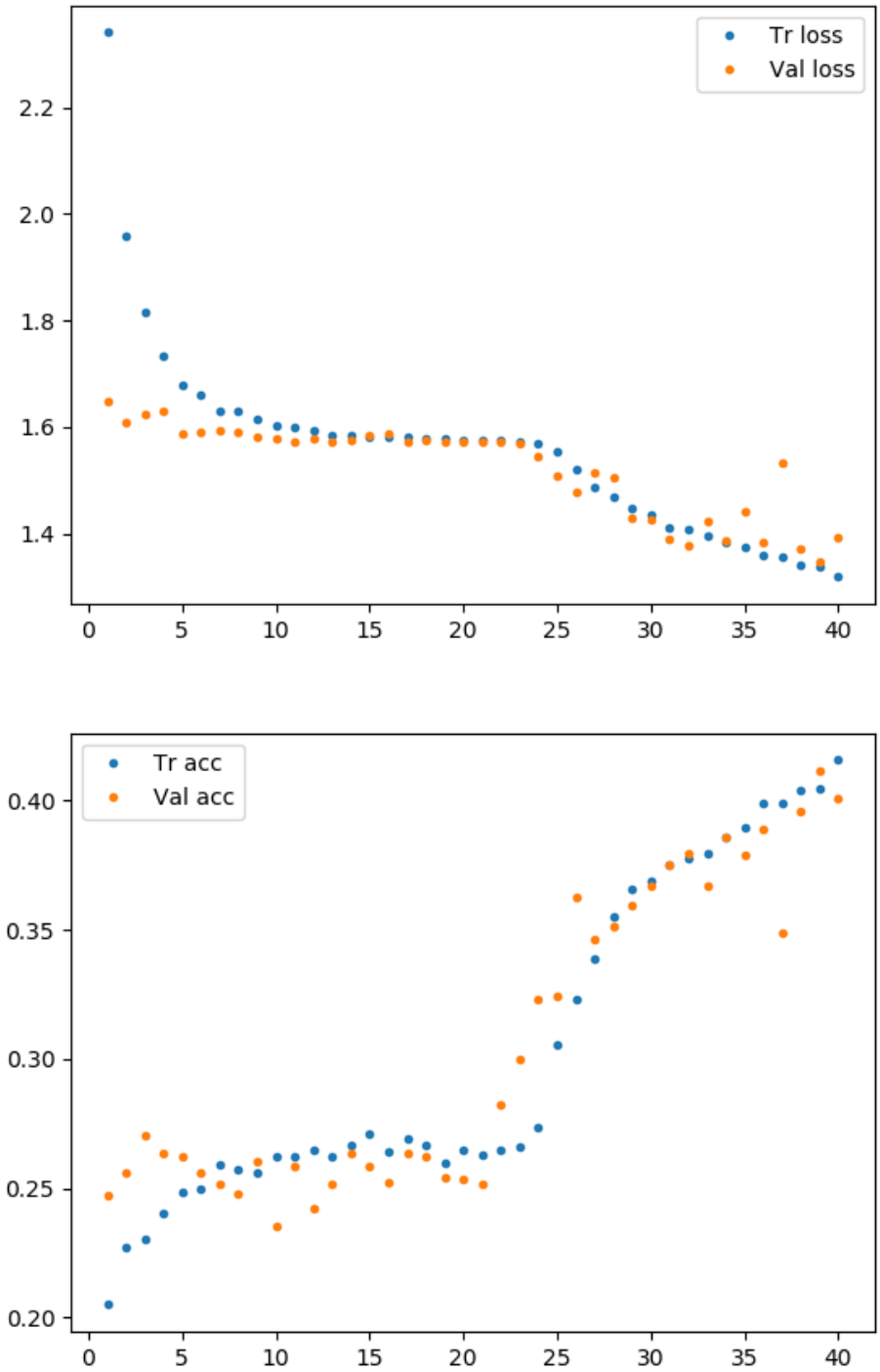

Figure 2: Learning curve for the other encoder