Project 2 Deep Learning

Alejandro de la Concha (alexdavidhalo@gmail.com)

1 Multilingual word embeddings

The goal is to find a mapping W that will map a source word space (e.g French) to a target word space (e.g English), s.t. the mapped source words are close to their translations in the target space. For this, we need a dictionary of "ancho points". Here, we will use the identical character strings in both languages. We can show that the solution of $\text{argmin}_{W \in O_d(\mathbf{R})} ||WX - Y||_F^2$ has a closed form.

1.1 Question

Using the orthogonality and the properties of the trace, prove that, for X and Y two matrices:

$$W^* = \text{argmin}_{W \in O_d(\mathbf{R})} ||WX - Y||_F^2 = UV^T, U\Sigma V^T = SVD(XY^T)$$

First, lets remember that the singular value decomposition of a dxn matrix $A = U\Sigma V^\top$ is such that:

U is a dxd orthogonal matrix whose column vectors are know as the left-singular vectors.

Σ is a dxn diagonal matrix with positive entries known as singular values.

V is a nxn orthogonal matrix whose columns are known as the rigth-singular vectors.

Minimizing $\|WX - Y\|_F$ is equivalent to minimize $\|WX - Y\|_F^2$. I will develop this quantity and by taking advantage of the properties of the transpose and the orthogonality of W we get.

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

Then our minimization problem is equivalent to the maximization problem $\max_{W \in O(R^d)} tr(X^\top W^\top Y)$. By using the property that $tr(AB) = tr(BA)$ if A is an dxn matrix and B is an nxd matrix we have:

$$\begin{aligned}
tr(X^\top W^\top Y) &= tr(W^\top Y X^\top) \\
&= tr(W^\top U\Sigma V^\top) \\
&= tr(V^\top W^\top U\Sigma)
\end{aligned}$$

Let's call $S = V^\top W^\top U$. As by definition S is the product of orthogonal matrix for all $W \in O(R^d)$, it is an orthogonal matrix. Then:

$$\begin{aligned} tr(V^\top W^\top U\Sigma) &= tr(S\Sigma) \\ &= \sum_i^n S_{i,i}\sigma_i \end{aligned}$$

As the entries of Σ are positive and S is orthogonal, in order to maximize the last sum, the entries $S_{i,i}$ should be equal to one. Then $S = I$. From this observation, we can conclude that:

$$V^\top W^\top U = I$$

Then:

$$W = UV^\top$$

2 Sentence classification with BoV

In this section and the following, we give you the train, dev and test sets of the Stanford Sentiment Treebank fine-grained sentiment analysis task. It consists of input sentences that you have to classify into 5 classes. For the test set, we only provide you with the input samples, not the ground-truth labels. You will have to produce your predictions using your best model, and send it to us. We will evaluate ourselves the quality of your predictions. You are asked to use scikit-learn to learn a logistic regression on top of bag-of-words embeddings on the SST task.

2.1 Question

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

In the question of monolingual embeddings, I was asked to create sentence embeddings and find the closest sentences to a given one. I got more satisfying results using the wheighted embeddings with IDF. For that reason, I decided to use the same approach in this classification task. I used a logistic regression with a L_2 penalized term wit parameter $C = 0.5$. With this model, I got an accuracy of 46.79% in the train set and 41.05% in the development set. As the difference between both is not very big, it seems that the model does not overfit.

Accuracy	**Train**	**Development**
With IDF	0.467965	0.410536

3 Deep Learning models for classification

3.1 Question

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

In a multiclass classification problem the cross entropy is normally used as loss function:

$$-\frac{1}{N}\sum_{n=1}^{N}\sum_{k=1}^{K} y_{n,k}\log(p_{n,k})$$

where N is the number of observations, K is the number of classes, $y_{n,k}$ is a variable which takes the value of 1 if observation n is in the class k and 0 otherwise and $p_{n,k}$ is the estimated probability by the model that the observation n belongs to class k. In this exercise $K = 5$ and $n = 8544$.

I got good results with this loss function. Nevertheless, there are many more options as the the hinge loss which is present in SVM's and can be applied to multiple class classification too.

3.2 Question

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

When trying different combinations of hyper-parameters, I realized that the proposed model overfitted the data in most cases even if I fixed the dropout rate at high values.The evolution of the model accuracy throughout the number of epochs suggest that the number of parameters to estimate it too big with respect to the number of observations.

The model that leaded me to the lowest overfitting is made of a layer aiming to learn the word embedding of dimension 50, a LSTM network of 15 hidden units and a output layer with the classifier to the 5 existing classes. I used a dorpout rate of 0.3. The number of parameters of this architecture is of 895,990 as it can be seen in Figure 1.

The accuracy of the algorithm is 38.06% over the dev set, which is inferior to the results found in the logistic regression.

Accuracy	**Train**	**Development**
LSTM	0.654962	0.380563

The figure bellow shows the evolution of the accuracy and the loss during training. I tried different optimizers and I kept ADAM. I used a batch size of 64 and 5 epochs. During the training , we can see how the accuracy and the loss function diverge from each other, a clear syntom of overfitting.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_3 (Embedding)      (None, None, 50)          891950
_________________________________________________________________
lstm_3 (LSTM)                (None, 15)                3960
_________________________________________________________________
dropout_3 (Dropout)          (None, 15)                0
_________________________________________________________________
dense_3 (Dense)              (None, 5)                 80
=================================================================
Total params: 895,990
Trainable params: 895,990
Non-trainable params: 0
_________________________________________________________________
None
```

Figure 1: Number of parameters for the first Neural network model.

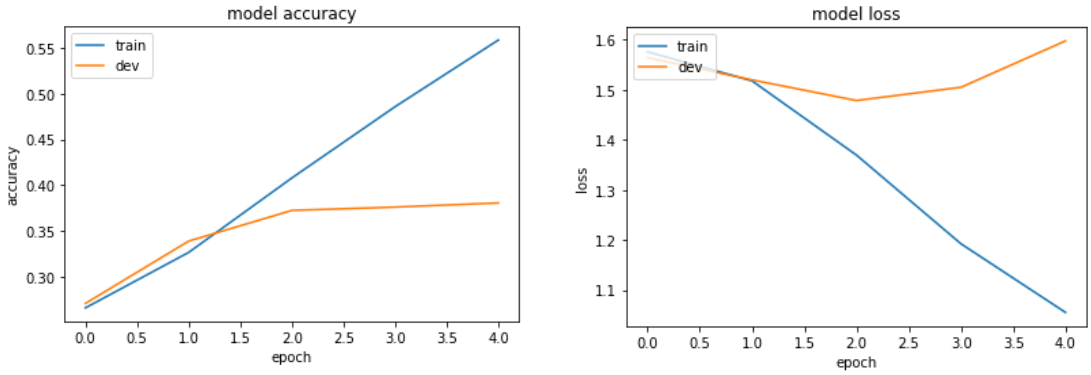

Figure 2: Evolution of loss and accuracy in the train and dev set through epochs of the LSTM model

3.3 Question

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

As I pointed out before, I though the main problem of the model proposed in the last question was the number of parameters to estimate, for this reason I decided to not make the network learn the embeddings, but use pretrained ones. I gave the network the pretrained embeddings used in the previous exercises. I know that this embeddings are high quality since they were trained using more data and they keep important features of each word as the previous exercises indicated, something I will hardly achieve myself given the number of observations. So I fixed them as the word embeddings of the model and I did not update them.

Then, as I see that regularization improved a lot the results in logistic regression and that the LSTM had good results for a small embedding dimension, I supposed not all dimension of the embedding space where informative for this particular case so I reduced the dimension of the embedding by using a fully connected layer of 50 units. The outputs of this layer were the imputs of a Bidirectional LSTM model with 50 hidden units since, in this case, it is key to keep track of the information given by the words in the entire phrase and how they interact, for example it is important that the

model know that it is not the same "incredibly boring" and "incredibly interesting" so I keep this structure with a dropout rate of 20%

During training, I got the best results with "ADAM", a number of epochs of 10, a batch size of 64. With this simple modifications I improved the previous model by around 11% and the logistic regression by almost 3% in the dev set.

Accuracy	**Train**	**Development**
LSTM with pretrainned embeddings	0.507490	0.439600

Addittionally, the figures bellow indicate a more consisting training where both accuracies and loss functions are not far away from each other and in the case of the accuracy it increases in both sets, so I hope there is not overfitting.

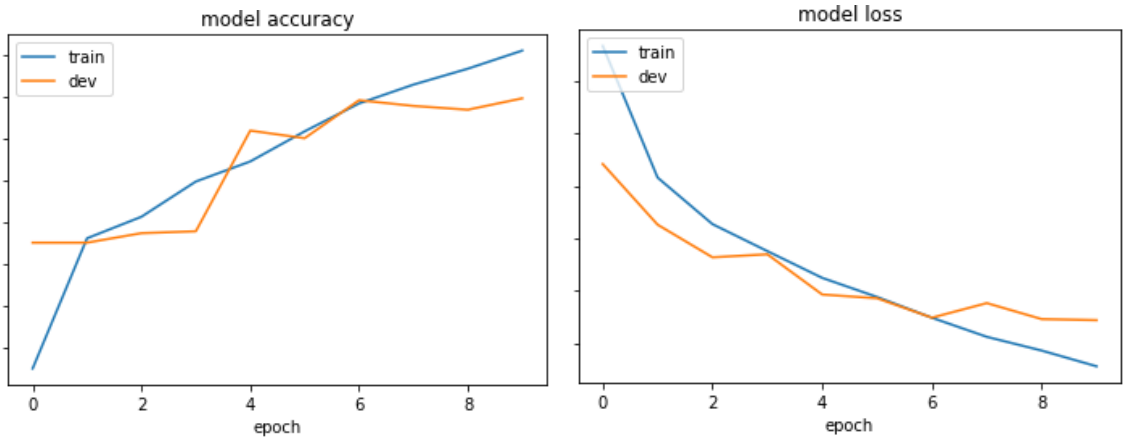

Figure 3: Evolution of loss and accuracy in the train and dev set through epochs of the LSTM model using pretrained embeddings

The attention mechanism is a technique that is ubiquitous in NLP. I tried to improve the previous model by adding this component. The training process remained the same. The new algorithm got an improvement of 1 % with respect to the previous algorithm.

Accuracy	**Train**	**Development**
LSTM with pretrainned embeddings and attention mechanism	0.51638	0.445049

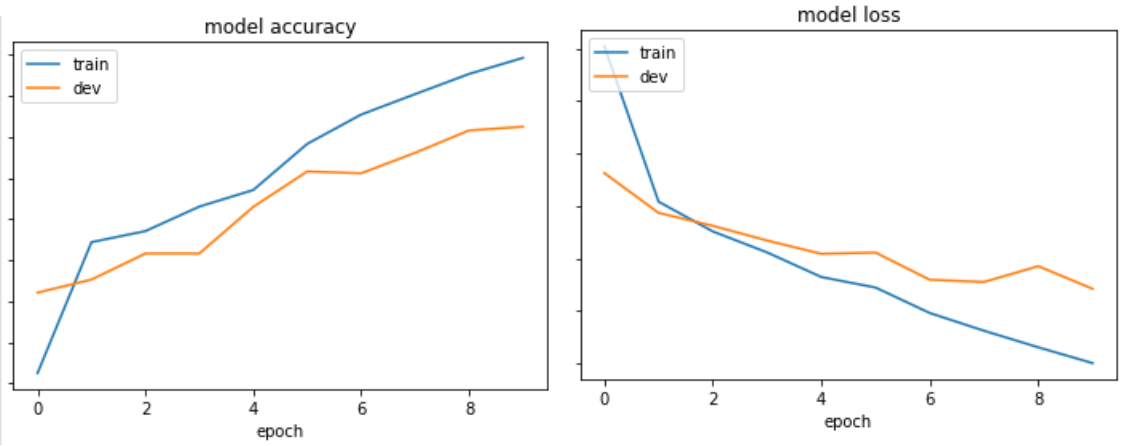

Figure 4: Evolution of loss and accuracy in the train and dev set through epochs of the LSTM model using pretrained embeddings and attention mechanism