Foundations of Deep Learning - CentraleSupelec - Spring 2019 - Assignment 2

Ayush Kumar Rai – ayush.rai2512@student-cs.fr

January 18, 2019

By turning in this assignment, I declare that all of this is my own work.

I. Monolingual Embeddings

This part involves only coding questions therefore please refer to the Jupyter Notebook **nlp_project.ipynb.** Please read the comments in the code.

II. MultiLingual Word Embeddings

Solution: Question 1

In order to find a mapping between source and target embedding we can solve the following problem

$$\underset{W \in O_d(\mathbb{R})}{\operatorname{argmin}} \|WX - Y\|_F$$

$$Or$$

$$\underset{W \in O_d(\mathbb{R})}{\operatorname{argmin}} \|WX - Y\|_F^2$$

$$\underset{W \in O_d(\mathbb{R})}{\operatorname{argmin}} (WX - Y)(WX - Y)$$

$$\underset{W \in O_d(\mathbb{R})}{\operatorname{argmin}} \|X\|_F^2 + \|Y\|_F^2 - 2\langle WX, Y\rangle$$

$$\underset{W \in O_d(\mathbb{R})}{\operatorname{argmin}} -2\langle WX, Y\rangle$$

$$\underset{W \in O_d(\mathbb{R})}{\operatorname{argmax}} \langle WX, Y \rangle$$

$$\underset{W \in O_d(\mathbb{R})}{\operatorname{argmax}}\, trace(WXY^T)$$

From the Von Neumann theorem we can state : For any $m \times n$ real-valued matrices F and G, let $\sigma_1(F) \geqslant \sigma_2(F) \geqslant ... \geqslant 0$ and $\sigma_1(G) \geqslant \sigma_2(G) \geqslant ... \geqslant 0$ be the descending singular values of F and G respectively. Then

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

We now introduce two Single Value Decompositions

$$SVD(W) = U_w \Sigma_w V_w$$

$$SVD(YX^T) = U \Sigma V$$

Then we can rewrite the quantity that we want to maximize for any $W \in O_d(\mathbb{R})$ by

$$trace(WXY^T) = trace(U_w \Sigma_w V_w (U \Sigma V)^T)$$

$$trace(WXY^T) = trace(U_w \Sigma_w V_w V \Sigma U^T)$$

$$trace(WXY^T) = trace(\hat{U} \Sigma_w \hat{V} \Sigma)$$

with $\hat{U} = U^T U_w$ and $\hat{V} = V V_w$

$$trace(WXY^T) \leqslant trace(\Sigma_w \Sigma)$$

by applying the Von Neumann theorem to $F = W$ and $G = XY^T$

Hence the quantity is maximized for $\hat{U} = U^T U_w = I$ and $\hat{V} = V V_w = I$. Hence $V_w = V^T$ and $U_w = U$. Therefore we have $W = U \Sigma_w V^T$. Finally using the fact that $W \in O_d(\mathbb{R})$ we have $W^T W = I$.

That leads to $U \Sigma^2 U^T = I$ which implies that $\Sigma^2 = \Sigma = I$. Therefore finally we obtain $W = UV^T$.

III. Sentence Classification using BoV

Solution: Question 1

Using the average of word vectors, the training accuracy obtained is **0.477** while the dev set accuracy is **0.4069**

Using the weighted average of word vectors, the training accuracy obtained is **0.4637** while the dev set accuracy is **0.3941**.

IV. Deep Learning Models for Classification

Solution: Question 1

For sentence classification task using LSTM, I used **Categorical CrossEntropy Loss** given by the following expression

$$\sum_{c=1}^{M} y_{o,c} \log(p_{o,c})$$

M : number of classes, which is 5 in this problem
log : natural log
y : binary indicator (0 or 1) if class label c is the correct classification for observation o
p : predicted probability observation o is of class c

Solution: Question 2

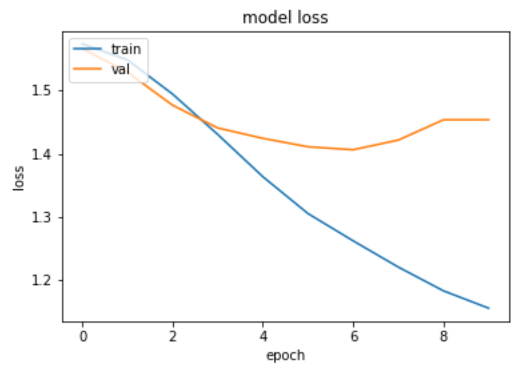

Figure 1: Loss Plot for LSTM based Network

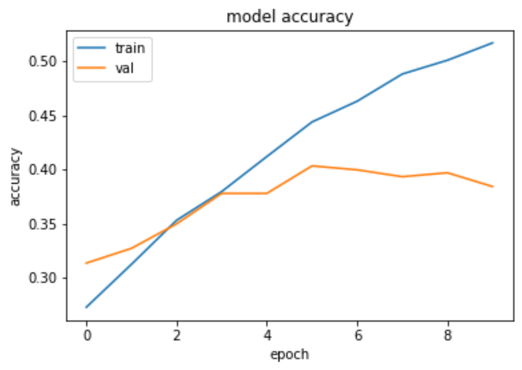

Figure 2: Accuracy Plot for LSTM based Network

It can be easily inferred from the above two plots that the LSTM based network is getting overfitted very easily for this problem. We can conclude that the variance of our model is high which is leading to over-fitting.

Solution: Question 3

Regarding the innovation part, I am using Conv1D layers instead of RNN and LSTM. Intuitively, both LSTM and 1D conv nets are more or less the same. The input shape for both are 3-D tensors, with the shape of LSTM being (batch, timesteps, features) and the shape of 1D conv nets being (batch, steps, channels). They are both used for tasks involving sequences like time series, NLP etc.

An important case where LSTMs are easier to use is with data of unknown lengths. For example, in sentence translation (e.g. translating Chinese to Icelandic) both the input and output sizes are dynamic. In this case, it is easier and more intuitive to use LSTMs than to try to use CNNs.

However in our problem of sentence classification, we are using sentence embedding of fixed length therefore Conv1D layer seems a good choice. Also 1D CNNs are faster to train and perform better for fixed length inputs.

Note: An important observation that I made is that LSTM based models are very easily getting overfitted than 1D CNNs. Following are the loss and accuracy curves for sentence classification problem.

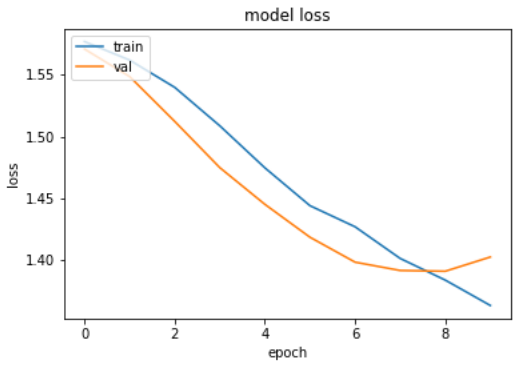

Figure 3: Loss Plot for 1D CNN based Network

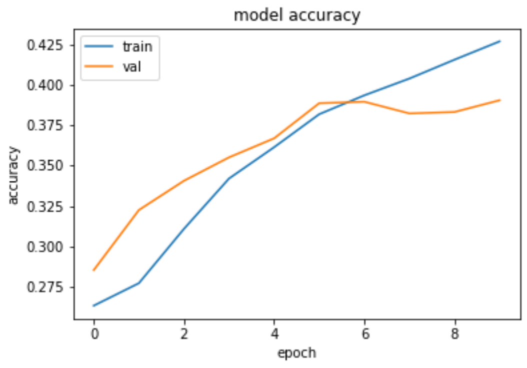

Figure 4: Accuracy Plot for 1D CNN based Network