Homework 2

Βησσαρίων Μουτάφης sdi1800119@di.uoa.gr
1115201800119

1. Problem 1

Let $softmax(\theta_i) = \sigma(\theta_i) = \frac{e^{\theta_i}}{\sum_{k=1}^{k} e^{\theta_k}}$, $J = CE(y, \hat{y}) = -\sum_i y_i \log \hat{y}_i$, where θ are the logits, y is the true labels, $\hat{y}$ is the predicted labels. We are asked to define

$$\frac{\partial J}{\partial \theta} = \begin{pmatrix} \frac{\partial J}{\partial \theta_1} \\ . \\ . \\ . \\ \frac{\partial J}{\partial \theta_n} \end{pmatrix}$$

1.1 Softmax derivation

Let us first define the quantity

$$\frac{\partial \sigma(\theta_i)}{\partial \theta_j}$$

- **If** $i == j$, then it is know to us that

$$\frac{\partial \sigma(\theta_i)}{\partial \theta_j} = \sigma(\theta_j)(1 - \sigma(\theta_j))$$

- **If** $i \neq j$, we have:

$$\frac{\partial \sigma(\theta_i)}{\partial \theta_j} = \frac{\partial}{\partial \theta_j} \left(\frac{e^{\theta_i}}{\sum_{k=1}^{k} e^{\theta_k}} \right) = \frac{0 - e^{\theta_i} e^{\theta_j}}{(\sum_{k=1}^{k} e^{\theta_k})^2} = -\sigma(\theta_i)\sigma(\theta_j)$$

1.2 Cross Entropy Derivation

Now we can proceed by using the knowledge of the above section, to derive loss function in respect of 1 of the output logits and then generalize this formula to acquire the overall derivative.

Let us start:

$$\frac{\partial J}{\partial \theta_j} = \left(-\sum_i y_i \frac{\partial}{\partial \theta_j} \log \sigma(\theta_i) \right)$$

$$= - \sum_{i \neq j} y_i \frac{1}{\hat{y}_j}(-\hat{y}_j \hat{y}_i) - \frac{y_j}{\hat{y}_j} \hat{y}_j (1 - \hat{y}_j)$$

$$= - \sum_{i \neq j} y_i(-\hat{y}_i) - y_j(1 - \hat{y}_j)$$

$$= \sum_i y_i \hat{y}_j - y_j$$

But we know that for a given y that $\sum_i y_i = 1$ so we can conclude to the following formula

$$\frac{\partial J}{\partial \theta_j} = \hat{y}_j - y_j$$

and by generalizing on initial formula, we get

$$\frac{\partial J}{\partial \theta} = \begin{pmatrix} \hat{y}_1 - y_1 \\ . \\ . \\ . \\ \hat{y}_n - y_n \end{pmatrix} = \hat{y} - y$$

2. Problem 2

Given the assignment's graph let us define

$$z = m \cdot x + b$$

$$\text{y} = \text{ReLU(z)} = \text{a} \cdot max(0, z)$$
$$\text{J} = \text{MSE(y, y}^*\text{)} = (y - y^*)^2$$

First let's define the function $I(x > 0) = 1$ when $x > 0$ and $I(x > 0) = 0$ when $x \leq 0$. We also can estimate the following derivatives
$\frac{\partial J}{\partial y} = 2(y - y^*)$, $\frac{\partial J}{\partial y^*} = -(y - y^*)$, $\frac{\partial y}{\partial z} = a \cdot I(x > 0)$, $\frac{\partial z}{\partial x} = m$, $\frac{\partial z}{\partial m} = x$, $\frac{\partial z}{\partial b} = 1$ Now we have to apply the chain rule to acquire the derivatives:

$$\frac{\partial J}{\partial x} = \frac{\partial J}{\partial y} \cdot \frac{\partial y}{\partial z} \cdot \frac{\partial z}{\partial x} = I(x > 0) \cdot 2ma(y - y^*)$$

$$\frac{\partial J}{\partial m} = \frac{\partial J}{\partial y} \cdot \frac{\partial y}{\partial z} \cdot \frac{\partial z}{\partial m} = I(x > 0) \cdot 2ax(y - y^*)$$

$$\frac{\partial J}{\partial b} = \frac{\partial J}{\partial y} \cdot \frac{\partial y}{\partial z} \cdot \frac{\partial z}{\partial b} = I(x > 0) \cdot 2a(y - y^*)$$

$$\frac{\partial J}{\partial y^*} = \frac{\partial}{\partial y^*}(y - y^*)^2 = -2(y - y^*)$$

3. Sentiment Analysis with pytorch

In this section we will comment on our jupyter notebook where we trained and tested 4 models, 2 of them using the CountVectorizer's interpretation of tweets and the other 2 using GLoVe's embeddings aggregations to interpret a tweet.
NOTE: Connect a GPU to the notebook to run faster. The attached notebook supports gpu usage.

3.1 FeedForward Model class

This is the class, we will use to create a NN with variable number of hidden layers with different sizes. The model also realizes the ideas of Batch Normalization after input layer and after each activation. This feature in combination with some Dropout layers will secure us from the dangers of overfitting. Also we use pytorch-dataloaders to load out train-test data so we also apply batching while training in order to find the perfect data batch to train the models. Our models seem to train better with higher data batches since with batches of 32 to 128, the loss is decreases realatively slow and also the training is too long with no important performance boost.

3.2 FeedForward with Embeddings class

This template class is a derived class from the standard Feed-forward model we discessed in the last section, with only one difference, the user must provide us with a trained embedding layer. The latter will be a pytorch EmbeddingsLayer where the weights will be set according to some pretrained embeddings, in our case GLoVe embeddings and some random vectors for all the words in the train corpus where $w \notin Corpus(GLoVe)$. In this manner we will also use a *sentece to Embeddings Transformer* that our class provides the user with, in order to create an acceptable input from our raw tweets.

3.3 Models Evaluation

Among the 4 new models we can clearly see that model 2 wembeddings is the best perfoming one in terms of the metrics we used to measure models' predictive power. Let us now explore the reasons behind this.

First we can take a look at the metrics table (check notebook at section *Evaluation*). We can clearly notice that the "anti-vax" class is the lowest in terms of accurate predictions and this is because the models don't have enough anti-vax records so the models aren't so

familiar with this class features. Also we notice that neutral predictions have best results in terms of all metrics so we could deduce that this is the most dominant class in the dataset.

While metrics, provides us with much useful information, we can derive more conclusions by watching the Loss vs Epochs graphs and the ROC Curves. In the Loss vs Epoch graphs we can notice that only 2 out of 4 models have smooth loss curves, meaning there was always loss decrease while training and possibly they can be scaled to be better models. Nevertheless the second BoW NN, seems to be slightly overfitting after 90 iterations, so we can deduce that, however the learning algorithm, this kind of BoW techniques have a performance threshold, in regards of our NL sentiment analysis problem. By noticing the Embeddings models we can watch a slight decrease in error but no overfitting overall. This means that our embeddings space possess higher explaining power than BoW vector space, but our models are not appropriate for this kind of analysis. This is a logical deduction if one consider that our NL analysis does not take under consideration the time context of words, something that only an RNN could do.

Finally, by watching the ROC Curves we can realize the True Positives-False Negatives rate, that reassures as of our results: The "neutral" class is the most separable since its closer to the upper-left section of the graph, while "pro" and "anti-vax" are less separable due to the lack of training examples and the inherit label instability of the dataset. We can also notice that the "AUC" scores are greater that 0.8 for "neutral" class in all 4 models, when in the other 2 classes the scores beat this threshold, only in *Softmax* and second BoW NN models. This would ensure as that the second BoW neural network is the best of the 4 algorithms and by studying the ROC-Curve we can conclude that is nearly the same as the Softmax model, in terms of performance and separability.

Conclusion: Best model seems to be the Model 2 without Embeddings.

3.4 Softmax vs Best NN

After comparing softmax and NN models we notice that they are nearly the same with the only difference being their test loss scores where the softmax model clearly beats our NN, with a loss of 0.7 where the NN model overfits and results to a test loss of 0.72.

3.5 Notes about embeddings models

- Most models prefered the SGD optimizer with large batches of 256 or 512 datapoints per iteration.
- The learning rates where tuned after some experimenting with values in $(0.00001, 0.01)$
- The Loss function was set to Cross Entropy Loss Since this was appropriate for logits of multiclass classification
- The embeddings class was incorporated only as a passing stage of models input. The embeddings with best results was the ones of 200 features.
- The fact that Embedding models undefit with high error is reasonable since there is no learning mechanism to inspect the word time context and grammar order.

4. References

- ROC-Curves
- Classification Report
- How to load word Embeddings
- What are word embeddings