UNIVERSITÄT
DES
SAARLANDES

Neural Networks: Theory and Implementation

WS 19/20

Prof. Dr Dietrich Klakow
Project 02
Date of Submission: 28-02-2020

Smita Bhattacharya
Matrikelnummer: 2581485
Tutorial Group: Thursday

Task 1

Dataset Pre-processing

In this task, the dataset used is AG_NEWS. The inbuilt torchtext TextClassificationDataset is used. The train and test dataset is obtained from torchtext Dataset. Since there is no separate validation data, the training dataset is a divide at the ration of 0.95 to obtain train data and validation data. Then a top-level method is defined that generates the batch which is then provided as the collate function to the data loader. The data loader loads the data in batches parallelly.

Model Architecture

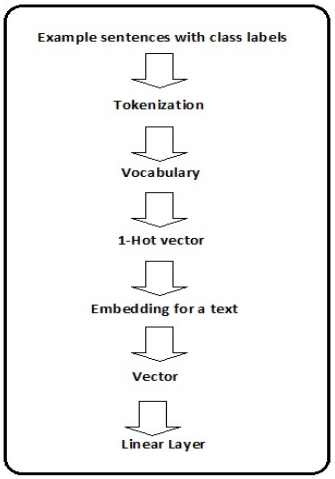


Figure 1.1 Model Architecture of Linear Neural Network model

The model architecture is as shown in Figure 1.1. The model consists of an embedding layer, a fully connected linear layer. The n-grams of the input are obtained and appended at the end of the sentence. This is then passed through an embedding layer to obtain the word embedding. This is then passed to a linear layer to obtain the output.

Training and Evaluation

The optimizer used is stochastic gradient descent. During each forward pass, the SGD optimizer updates the model parameters by the amount defined by the learning rate for each

parameter. The loss function used is binary cross-entropy. The model accuracy is computed as the percentage of correct predictions. The gradients are set to zero. The batches are then fed to the model. The loss computed is averaged over all the data points in the batch. The gradients are computed using loss.backward(). The parameters are then updated using the optimizer.step(). The loss and accuracy are accumulated and averaged across the epoch. The model is trained for multiple epochs. An epoch is completed when all the data points in training and validation data are passed through the model. The model obtained after all the epochs are finished is then used to compute the test accuracy by passing the test data through the model.

Results and Conclusions

The test accuracy obtained at the end of 5 epochs is approximately 89%. The validation accuracy improved with each epoch. A linear model does not take into consideration the context. This provides the scope for further improvement.

As we are using simple fully connected networks we are considering the current state and predicting the output, which in other sense is not considering the sequence of the word appearance or temporal relation of the words. So we can use different models such as LSTM where the sequence of the word matters not just the words. We can increase the depth of our neural network and try. Because usually in the initial layers only primitive features are learnt. So we can try to increase the depth of the network and test whether the model learns complex features. We can reduce the step size and increase the number of epochs for the current network to see whether it gives more validation and test accuracy.

Task 2

Dataset Pre-processing

In this task, the dataset used is AG_NEWS. In order to have flexibility around the format of data required by the model, a custom dataset is created using the raw data with the help of pandas instead of using the inbuilt torchtext TextClassificationDataset. The compressed data file is downloaded from the official site. It is then extracted to obtain train and test CSV files. Using pandas, comma-separated values are processed to obtain text, title and the respective label. It is then used to create train data and test data. Since there is no separate validation data, the training dataset is a divide at the ration of 0.95 to obtain train data and validation data. Then the iterators are created for the train, validation and test data. Iterators split the data into batches of a specified batch size such that within batch the data points are of similar size which results in minimal padding.

Model Architecture

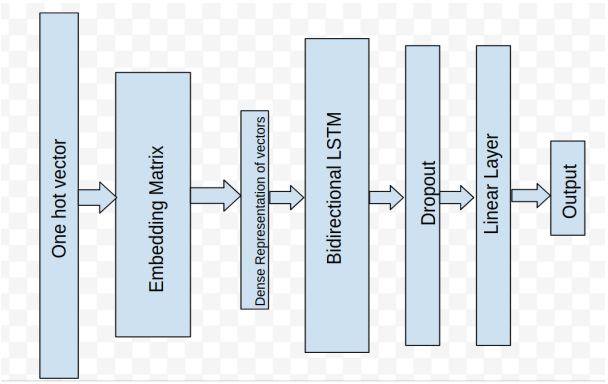


Figure 2.1 Bi-direction Multi-layer LSTM for Text Classification

The model architecture is as shown in the Figure 2.1. The first layer is an embedding layer. A pre-trained glove embedding is used. The word embedding is initialized with pre-trained vectors. The glove embeddings are such that semantically closer words are also closer in the vector space. Hence the embedding has already learnt some relationship among the words and need not learn these vectors from scratch. The glove embedding used is “glove.6B.100d”. It consists of vectors of dimension 100 obtained from the training of 6 billion tokens.

The next layer in the model is a Bi-directional LSTM. LSTM is a variant of the RNN model. LSTM in addition to hidden states also uses cell states. The cell states can be considered as the memory of the LSTM. The changes made to cell states is controlled by multiple gates. In a bi-directional LSTM, one LSTM processes the words from the beginning and another LSTM processes the words from the end. The output so obtained is the function of last hidden states of the forward and backward LSTM. The model consists of of a multi-layer

LSTM. In a multi-layer LSTM, the hidden states from the first layer are used as the input in the next layer. The output is the function of last hidden states of the highest layer.

Further, the model has two linear layers which take the input from the LSTM layer and provides the output. Since the model consists of many layers, it consists of many parameters. Hence, there is a possibility of overfitting resulting in low validation and test errors. Hence, there is a necessity of applying regularization. The regularization method applied in the current model is dropout. In the dropout method, neurons in a particular layer to which dropout is applied are dropped randomly with the probability defined by the hyperparameter.

Training and Evaluation

The optimizer used is Adam (Adaptive Moment Estimation). During each forward pass, the Adam updates the model parameters by adapting the learning rate for each parameter. An initial learning rate is provided. The loss function used is a cross-entropy. The model accuracy is computed as percentage of correct predictions. The model is first put into training mode using model.train(). This turns on the batch normalization and regularization. The gradients are then set to zero. The batches are then fed to the model one by one. The loss computed is averaged over all the data points in the batch. The gradients are computed using loss.backward(). The parameters are then updated using the optimizer.step(). The loss and accuracy are accumulated and averaged across the epoch. The hyperparameters in the model are learning rate, dropout, the number of layers, hidden layer dimension are the important ones. The model is trained for multiple epochs. An epoch is completed when all the data points in training and validation data are passed through the model. For the evaluation, the model is put evaluation mode using model.eval(). This turns off the dropout. The model obtained after all the epochs are finished is then used to compute the test accuracy by passing the test data through the model.

Results and Conclusions

The test accuracy of the model obtained at the end of 10 epochs is approximately 91%. Although the validation accuracy is highest in the second epoch, it would be too early to stop training. Hence ten epochs have been used which was decided by trying a different number of epochs and choosing the number of epochs that resulted in stable better accuracy. Comparing this model to the simple linear model in task 1, the test accuracy has improved from approximately 89% to 91%. This can be accounted for the efficiency of LSTM on sequential data. The LSTM considers the larger context. And since bi-directional LSTM is used the context in the future is also considered for current apart from the context in the past. Using the regularization has improved the accuracy of the model on the unseen data.

Task 3

Our unique approach: We are taking two regular independent neural network models and Can we combine two different model predictions using probability theories and come up with predictions which will have the influence of both the models.

As a simple neural network and LSTM both have its advantages, so can we combine the probabilities and get the advantages of both the models to obtain higher accuracy. This is the intuition behind our approach.

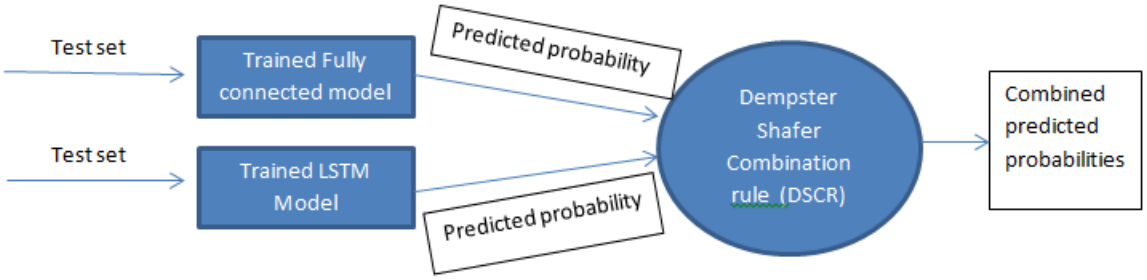
Model Architecture

We are using the existing two models and training them independently on the same data set. Then we are trying to combine the predictions of the two models using the Dempster Shafer combination rule (DSCR) to get a weighted prediction which will have the effect of both the models.

We have used a simple fully connected neural network without any hidden layers where our input dimension is vocabulary size and output dimension is no. of classes.

We have used a unidirectional LSTM followed by a linear layer and input dimension is equal to vocabulary size and the output size is equal to no. of classes.

While testing, the predictions of both the models are combined using DSCR as shown in the figure below.



Training and Testing:

We train both the models independently on the same dataset in a similar way as we did for task 1 and 2.

But here while testing we combine the predictions of both models using DSCR.

Dempster Shafer theory for combining two model output probabilities:

Let us consider two trained models “a” and “b” for binary classification.

Let us assume for a test case we get the following predictions, where “True” value indicates the probability of the test case belonging to first class.

Model “a” : True: 0.7 False : 0.3

Model “b” : True 0.6 False 0.4

Let us say we know from history, that we can trust Model “a” 60% and model “b” 80%

So this will affect the above-mentioned predictions as follows:

Model “a” : True: 0.42 Uncertainty: 0.42 False : 0.18

Model “b”: True 0.48 Uncertainty:0.2 False 0.32

Now we need to combine these new probabilities as per the combination rule:

	Model a True (0.42)	Model a Uncertain(0.42)	Model a False (0.18)
Model b True (0.48)	True (0.2016)	True (0.2064)	Uncertain (0.0864)
Model b Uncertain (0.2)	True (0.084)	Uncertain (0.086)	False (0.036)
Model b False(0.32)	Uncertain (0.1344)	False(0.1376)	False (0.0576)

From the above table, Combined probability of being True is calculated as follows:

$$Combined\ true\ probability = Summation\ of\ True\ probabilities\ /\ (1 - Summation\ of\ false\ probabilities)$$

= 0.2016+0.2064+0.084 / (1- (0.1376+0.036+0.0576)) = 0.64

So our combined probability of being true is 0.64.

In our case we consider the predicted probabilities from the Fully connected neural networks and LSTM. We have assigned 60% reliability for a fully connected neural network model and 80% reliability for LSTM model, as we believe LSTM model analyses texts better.

In our case, we have 4 classes. As we are considering DSCR for the binary classifier, we consider one vs all methods and convert our 4 class classification into the binary classifier and find out the combined the probability of each class and then use these combined probabilities to compute the test accuracy.

As we have 1,2,3,4 classes.

Compute Combined Probability of first-class as, 1 vs (2,3,4)

Compute Combined Probability of second class as, 2 vs (1,3,4)

Compute Combined Probability of third class as 3 vs (2,1,4)

Compute Combined Probability of fourth class as 4 vs (1,2,3)

Results and conclusions

We obtained 88.1% accuracy for fully connected neural network and we obtained 89.2% accuracy for LSTM model independently. When we combined the probabilities of both the model using DSCR, we obtained accuracy of 87.9%, which is less than expected. Maybe our intuition did not really hold good for this case as we thought combining both the probabilities may result in better accuracy.

References

[1]Xiang Zhang, Junbo Zhao, Yann LeCun. “Character-level Convolutional Networks for Text Classification”. Advances in Neural Information Processing Systems 28 (NIPS 2015). Poster. Datasets. Code. Errata.

[2]Siwei Lai, Liheng Xu, Kang Liu, Jun Zhao.”Recurrent Convolutional Neural Networks for Text Classification”. Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence.

[3] [BOOK] “Classic works of the Dempster-Shafer theory of belief functions”, Yager, L Liu - 2008 - books.google.com

[4] “Application of Dempster–Shafer theory in fault diagnosis of induction motors using vibration and current signals”, Mechanical Systems and Signal Processing, Volume 20, Issue 2, February 2006, Pages 403-420

[5] R. R. Murphy, "Dempster-Shafer theory for sensor fusion in autonomous mobile robots," in IEEE Transactions on Robotics and Automation, vol. 14, no. 2, pp. 197-206, April 1998.