

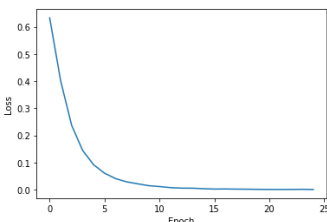
Assignment 2 Summary

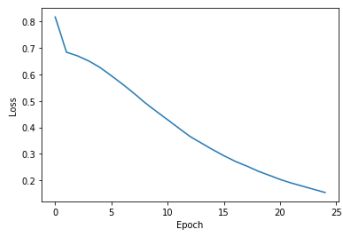
Method Description: In this task, I chose the Stanford Sentiment Treebank (SST-2) dataset to construct a model in Pytorch for text classification. SST-2, provides fine-grained sentiment labels for 215,154 phrases in parse trees of 11,855 sentences and introduces classification which is based on positive and negative emotions. For this assignment, I used a continuous bag-of-words (CBoW) neural network with embeddings close to those used in (Mikolov et al., 2013) [1].

Implementation: We are given a corpus of text from movie reviews for this text classification assignment. Each text in the dataset has been assigned a positive, negative, or neutral label. For the data preprocessing, I followed the Pytorch.org tutorial [2]. Model used in this assignment is based on the formula $f(x) = \text{sign}(w^t x + w_0)$ to predict y , where sign is the activation function, w^t is the learned weights, x is the input and w_0 is a bias vector. In this $y \in \{-1, 1\}$ is for the positive and negative sentiment inputs.

Continuous Bag-of-Words (CBoW) Model: In this assignment, I used CBoW model, which predicts the target word based on the four future and four history words at the input. For example, {"the", "cat", "over", "the", "puddle"}, these are the context words to predict the target word as "jumped" [3]. The training complexity is given by $Q = N \times D + D \times \log_2(V)$. Unlike BoW this model uses continuous distributed words in context.[1] Where, projectetion layer (P) has $N \times D$ dimensionality and the number of output units that must be evaluated can be reduced to about $\log_2(V)$ with binary tree representations of the vocabulary (V)[1]. For model implementation, I used Pytorch, which was trained for 10 epochs with learning rates of 1e-3 and 25 epochs with learning rates of 1e-4 both at a size of 10. The Adam optimizer and the negative log likelihood loss function were used in the model. For writing code I referred to different resources, among all, following are the ones that helped me the most, word embeddings by Pytorch[4], Word2Vec in Pytorch - Continuous Bag of Words[5] and sentiment classification[6] by Shayne O'brien.

Accuracy: For word embedding I used FastText embedding which is trained on english wikipedia for 294 languages and has 300 dimensions which obtained using skip-gram model [7]. FastText embeddings are augmented with sub-word detail that can be used to handle misspelled and out-of-vocabulary terms [8]. The findings obtained from the task are summarized below.

<i>Model</i>	<i>Epoch</i>	<i>Learning rate</i>	<i>Accuracy</i>	<i>Curve</i>
CBoW	10	1e-3	78.78	

CBoW	25	1e-4	82.04	
------	----	------	-------	--

References:

- [1] T. Mikolov, K. Chen, G. Corrado, J. Dean. “Efficient estimation of word representations in vector space.” arXiv preprint arXiv:1301.3781. 2013
- [2] Source code for torchtext.datasets.sst. Accessed on 2017 Pytorch contribution[online]. Available: https://text-docs.readthedocs.io/en/latest/_modules/torchtext/datasets/sst.html
- [3] Deep Learning for NLP. Accessed in spring 2016 [online]. Available: http://cs224d.stanford.edu/lecture_notes/notes1.pdf
- [4] Word Embeddings:Encoding Lexical Semantics. Accessed on 2021 [online]. Available: https://pytorch.org/tutorials/beginner/nlp/word_embeddings_tutorial.html
- [5] Word2Vec in Pytorch - Continuous Bag of Words and Skipgrams. Accessed on September 9, 2018 [online]. Available: <https://srijithr.gitlab.io/post/word2vec/>
- [6] Shayne O’Brien, “Sentiments Classification”. Accessed on Oct 7th, 2018 [online]. Available: <https://github.com/shayneobrien/sentiment-classification>
- [7] Bojanowski, P., Grave, E., Joulin, A., & Mikolov, T. (2017). Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5, 135-146.
- [8] FastText embedding with subword information. Accessed in 2018 [online]. Available: <https://www.kaggle.com/vsmolyakov/fasttext>