Language Modeling using GRU

Davide Dalla Stella (223727)
University of Trento
Via Sommarive, 9, 38123 Povo,Trento TN
davide.dallastella@studenti.unitn.it

Abstract

This document focuses on the implementation of the gating mechanism GRU (Gated Recurrent Unit) in a recurrent neural network. The network is trained using the Penn Treebank dataset and tests were done by changing hyperparameters trying to improve performance.

1 Introduction

Recurrent Neural Networks (RNN) are artificial neural networks that include a loop between interconnected neurons. Usually the output values of a higher layer are used as input for a lower layer and the interconnection between them allows the use of a layer as state's memory (fig: 1). The data used in these networks are sequential data or time series data and given the nature of them these networks are usually used for Natural Language Processing (NLP), language translation or speech recognition. A difference from other classical Neural Networks is that RNN shares parameters across each layer of the network. While feedforward networks have different weights across each node, recurrent neural networks share the same weight parameter within each layer of the network. The main problem of a RNN is that it only exploit recent information to solve present task. When a RNN tries to capture long term dependencies there is the problem of vanishing/exploding gradient. With the vanishing gradient problem the gradient gets smaller as it backpropagated more time steps back, in simple terms it consists in loss of information when a RNN has a long sequence of terms and it can not store them. The problem of the exploding gradient is like the vanishing one but in this case the gradient become larger as it is backpropagated more time steps back.
To solve these problems different versions of RNN have been created like LSTM or GRU.

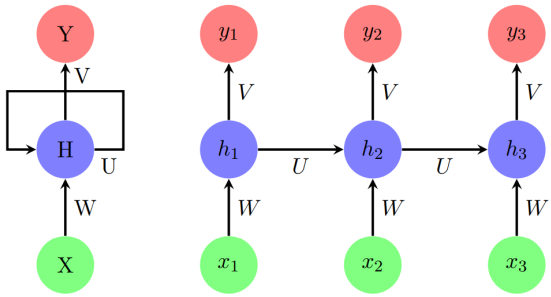

Figure 1: Recurrent Neural Network.

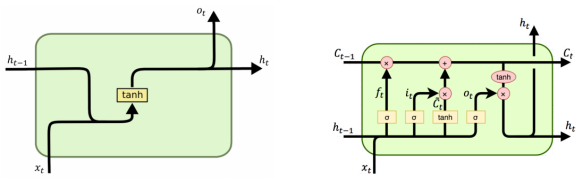

Figure 2: RNN Cell and LSTM Cell.

Long Short Term Memory (LSTM) manage context in order to remove information no longer needed and add necessary information for decisions that need to be take later. A cell of a LSTM consists in three "gates" (seen in fig. 2) necessary to control the information flow between units: forget gate, input gate and output gate. Gated Recurrent Unit (GRU) is just a simplification of LSTM since it uses 2 gates instead of 3: instead of the forget gate and the input gate it uses only an update gate, the second is the reset gate that control how much past information to forget. Furthermore it uses less training parameters and less memory. For these reasons it has lower accuracy if compared to LSTM but the execution and the training are faster.

2 Problem Statement

In Natural Language Processing language models are used to predict the probability of words analyzing bodies of text data and they can be used to

generate new sentences.
There are several types of models:

- **Unigram**: The simplest one, each word's probability is evaluated independently:

$$P(w_1, w_2, w_3) = P(w_1)P(w_2)P(w_3)$$

- **N-gram**: More sophisticated than unigram,it creates a probability distribution for a sequence of n:

$$P(w_i|w_1, \ldots, w_{i-1}) \approx$$

$$\approx P(w_i|w_{i-(n-1)}, \ldots, w_{i-1})$$

- **Bidirectional**: More complex than N-gram, it analyzes text in both directions: backwards and forwards:

$$P(w_1, w_2, \ldots, w_n) =$$

$$= \prod_{k=1}^{N} p(t_k|t_1, t_2, \ldots, t_{k-1})$$

- **Exponential**: The text is evaluated using an equation that combines feature functions and n-gram:

$$P(w_i|h_i) = \frac{1}{Z(h_i)} exp(\sum_j \lambda_j f_j(h_i w_i))$$

The goodness measurement of a model is **perplexity**. To know what the perplexity is it's necessary to calculate the normalized inverse probability of the test set.

$$\text{PPL(W)} \quad = P(w_1, w_2, w_3, \ldots, w_n)^{-\frac{1}{N}} =$$

$$= \sqrt[N]{\frac{1}{P(w_1, w_2, \ldots, w_n)}}$$

3 Data Analysis

The dataset used is called Penn Treebank which contains 10.000 unique words without numbers, punctuations or capital letters. The dataset composition is heterogenous, it contains sentences from different domains (like journals articles) and in this way the network is forced to learn each sentence separately.

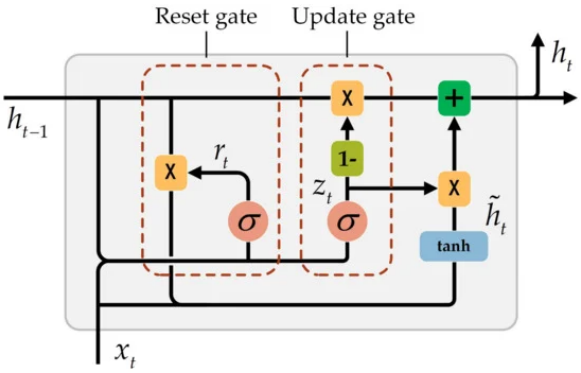

Figure 3: GRU cell architecture.

4 GRU Description

As already mentioned, GRU is a simpler version of LSTM. To solve the gradient problems it uses only two gates: update gate and reset gate. These gates are trained to filter any irrelevant information and essentially they are two vectors which contains only 0 and 1 which will be multiplied with the input data and/or hidden state. These values are necessary to chose which data should be passed or not. The structure of a GRU Cell (it can be seen in fig): 3.

- **Reset Gate**: it's used to decide how much of the past information forget. The formula to calculate it is:

$$r_t = \sigma(W_{ir}x + b_{ir} + W_{ir}h + b_{hr})$$

- **Update Gate**: This gate is essential to delete the problem of the vanishing gradient: The update gate helps the model to determine how much information from previous time steps needs to be passed along to the future and thanks to this the model can decide to copy all the information from the past. The formula is the same as that for the reset gate:

$$z_t = \sigma(W_{iz}x + b_{iz} + W_{hz}h + b_{hz})$$

- **Current Memory Content**: This memory content uses the reset gate to store the relevant information from the previous state:

$$\tilde{h}_t = \tanh(W_{in}x + b_{in} + r * (W_{hn}h + b_{hn}))$$

- **Final Memory at current time step**: Once the outputs of the two gates have been calculated and the information of the previous

state has been stored is possible to proceed with the calculation of the new state which contains the new information for the current unit.

$$h_t = (1 - z_t) * n_t + z_t * \tilde{h}_t$$

To resolve the problem of the exploding gradient it's necessary the gradient clipping. With this solution a threshold value on the gradient is sets in order to limit it. This way, the direction of the gradient remains unaffected and only the magnitude of the gradient is changed.

5 Results

The network is composed by an embedding layer, a GRU cell with a variable number of layers and a linear decoder layer. To isolate sentences and help the network to recognize the start and the end of them, the tags "begin of sentence" (<bos>), "end of sentence" (<eos>) and "padding" (<pad>) were inserted inside them. In order to avoid overfitting in this work the dropout was applied. This technique randomly set some neurons to zero in the forward pass and the choice of units to drop is random, determined by a probability p. An interesting test that has been done was to apply the dropout to both recurring and non-recurring cells and then only to non-recurring ones (tab. 1); in the second case the performance were better. Other tests were performed trying to change some parameters, for example the clipping was increased to lessen the gradient, the learning rate was increased, the number of GRU layers has been modified or the dropout has been decreased in order to increase the number of connected neurons. Two of these results obtained are shown in the tables 2 and 3. Is possible to see in tab. 2 that decreasing the dropout the perplexity get larger and in tab. 3 there is an improvement using only a GRU layer. A test was also made increasing the number of layers to 3 but worse performances were obtained. These results obtained are not very good, the lower perplexity score is 199.34 but maybe implementing a bi-directional GRU or a simple LSTM the results would be better.

embedding size	512	
hidden size	512	
number of GRU layers	2	
clip	0.35	
epochs	25	
dropout	0.5	
learning rate	0.001	
	Both cells	Non-recurrent cell
loss	**5.45**	**5.41**
perplexity	**232.76**	**222.73**

Table 1: Dropout to recurrent and non-recurrent cells

embedding size	512
hidden size	512
number of GRU layers	2
clip	0.35
epochs	25
dropout	0.25
learning rate	0.001
loss	**5.47**
perplexity	**236.28**

Table 2: Lower dropout

embedding size	512
hidden size	512
number of GRU layers	1
clip	0.35
epochs	25
dropout	0.5
learning rate	0.001
loss	**5.30**
perplexity	**199.34**

Table 3: 1 GRU layer

References

Koehrsen W. (Nov. 5 2018), Recurrent Neural Networks by Example in Python:
$https://towardsdatascience.com/recurrent-neural-networks-by-example-in-python-ffd204f99470$

Simeon K (Dec. 16 2017), Understanding Gru Networks:
$https://towardsdatascience.com/understanding-gru-networks-2ef37df6c9be$

Pengpeng L. et al. (Oct. 26, 2020), Bidirectional Gated Recurrent Unit Neural Network for Chinese Address Element Segmentation:
$https://www.mdpi.com/2220-9964/9/11/635?type=check\_update\&version=2$

Stepanov A. E. (2020), slides on Sequence Labeling with Deep Neural Networks material:
$University\ of\ Trento : Natural\ Language\ Understanding\ course$

Ricci E. (2020), Slides on Regularization:
$University\ of\ Trento : Deep\ Learning\ course$

Ghandi M. (Nov. 25 2018), Evaluation of Language Models through Perplexity and Shannon Visualization Method:
$https://towardsdatascience.com/evaluation-of-language-models-through-perplexity-and-shannon-visualization-method-9148fbe10bd0$

Zaremba W. et al. (Feb. 19, 2015), Recurrent Neural Network Regularization:
$https://arxiv.org/abs/1409.2329$

Penn Treebank Dataset:
$https://deepai.org/dataset/penn-treebank$

PyTorch GruCell:
$https://pytorch.org/docs/stable/generated/torch.nn.GRUCell.html$

Exponential language model:
$https://natural-language-understanding.fandom.com/wiki/Exponential\_language\_model$

Choi Y. (Aug. 27 2019) Pytorch Model:
$https://github.com/yunjey/pytorch-tutorial/blob/master/tutorials/02-intermediate/language\_model/main.py$

Loye G. (Jul. 22 2019) Gated Recurrent Unit (GRU) With PyTorch:
$https://blog.floydhub.com/gru-with-pytorch/$