Deep Learning Course

Assignment 3

Assignment Goals:

- Implementing and improving RNN based language models.
- Implementing and applying a Recurrent Neural Network on text classification problem.

In this assignment, you will implement RNN-based language models and compare extracted word representation from different models. You will also compare two different training methods for sequential data: Truncated Backpropagation Through Time **(TBTT)** and Backpropagation Through Time **(BTT)**. Also, you will be asked to apply Vanilla RNN to capture word representations and solve a text classification problem.

DataSets

You will use two datasets, an English Literature dataset for language model task (part 1 to 4) and the 20Newsgroups dataset for text classification (part 5).

Requirements

1. **(30 points) Implement a RNN based language model.**

   Implement the RNN based language model described by Mikolov et al., also called **Elman network**. The Elman network contains input, hidden and output layer and is trained by standard backpropagation (TBTT with $\tau = 1$) using the cross-entropy loss.

   - The input vector $x(t)$ at time t consists of the current word while using 1-of-N coding (thus its size is equal to the size of the vocabulary) $w(t)$ and a vector $s(t-1)$ which represents output values in the hidden layer from the previous time step $t-1$.
   $$x(t) = w(t) + s(t-1)$$
   - The hidden layer is a fully connected tanh layer with size 500.
   $$s_j(t) = f(\sum_i x_i(t) u_{ji})$$
   Here u is the parameter matrix of hidden layer, f is the tanh activation function.
   - The softmax output layer captures a valid probability distribution.
   $$y_k(t) = g(\sum_j s_j(t) v_{kj})$$
   Here v is the parameter matrix of output layer, g is the softmax function.
   - The model is trained with truncated backpropagation through time (TBTT) with $\tau = 1$: the weights of the network are updated based on the error vector computed only for the current time step.

   Train the language model on the given English Literature dataset, report the model cross-entropy loss on the train set. Visualize the cross-entropy loss during training using a curve line. Your curve line should demonstrate that the loss value converges.

   Use nltk.word_tokenize to tokenize the documents. For initialization, $s(0)$ can be set to a vector of small values. Note that we are not interested in the *dynamic model* mentioned in the original paper.

2. **(20 points) Train the Elman network with BTT.**

   TBTT has less computational cost and memory needs in comparison with **backpropagation through time algorithm (BTT)**. These benefits come at the cost of losing long term dependencies (reference). We will explore BTT in this part.

   Train your implemented Elman network with BTT, then compare the computational costs and performance of BTT and TBTT training. For training the Elman-type RNN with BTT, one option is to perform mini-batch gradient descent with exactly one sentence per mini-batch. (Hints: The input size will be (1, Sentence Length)). This means that BTT will use a complete sentence for updating the RNN weights (rather than just the previous word as in the TBTT part).

   - Split the document into sentences (you can use nltk.tokenize.sent_tokenize. The natural language toolkit (nltk) can be installed using the command 'pip install nltk').
   - For each sentence, perform one pass that computes the mean/sum loss for this sentence; then perform a gradient update for the whole sentence. (So the mini-batch size varies for the sentences with different lengths). You can truncate long sentences to fit the data in memory.
   - Report the model cross-entropy loss.Visualize the cross-entropy loss during training using a curve line. Your curve line should demonstrate that the loss value converges.

3. **(30 points) Improve your Elman network with GRU.**

   (a) Gated Recurrent Unit: It does not seem that simple recurrent neural networks can capture truly exploit context information with long dependencies, because of the problem of gradient vanishing and exploding. To solve this problem, gating mechanisms for recurrent neural networks were introduced. (15 points)

   Try to learn your last model (Elman + BTT) with the RNN unit replaced with a **Gated Recurrent Unit (GRU)**. Report the model cross-entropy loss. Visualize the cross-entropy loss during training using a curve line. Your curve line should demonstrate that the loss value converges. Compare your results in terms of cross-entropy loss with two other approaches (part 1 and 2).

   (b) Text generation: Use each model to generate 10 synthetic sentences of 15 words each. Discuss the quality of the sentences generated - do they look like proper English? Do they match the training set? (15 points)

   Text generation from a given language model can be done using the following iterative process:

   - Set sequence = [first_word], chosen randomly.
   - Select a new word based on the sequence so far, add this word to the sequence, and repeat. At each iteration, select the word with maximum probability given the sequence so far. The trained language model outputs this probability.

---

4. **(20 points) Implement a text classification model.**

   We are aiming to learn an RNN model that predicts document categories given its content (text classification). For this task, we will use the 20Newsgroups dataset. The 20Newsgroupst contains messages from twenty newsgroups. We selected four major categories (comp, politics, rec, and religion) comprising around 13k documents altogether. Your model should learn word representations to support the classification task. For solving this problem modify the **Elman network** architecture and simple RNN such that the last layer is a softmax layer with just 4 output neurons (one for each category).

   - Download the 20Newsgroups dataset, and use the below helper function data_loader() to read in the dataset.
   - Split the data into a training set (90%) and validation set (10%).
   - Implement your text classification model, and train the model on 20Newsgroups training set.
   - Report your accuracy results on the validation set. Try to achieve $\geq 80\%$ validation accuracy. (5 points)

Submission Notes

Please use Jupyter Notebook. The notebook should include the final code, results and your answers. You should submit your Notebook in (.pdf or .html) and .ipynb format. (penalty 10 points)

To reduce the parameters, you can merge all words that occur less often than a threshold into a special rare token (*_unk_*).

Instructions：

The university policy on academic dishonesty and plagiarism (cheating) will be taken very seriously in this course. Everything submitted should be your own writing or coding. You must not let other students copy your work. Spelling and grammar count.

Your assignments will be marked based on correctness, originality (the implementations and ideas are from yourself), and test performance.

Your Implementation

1. Implement a RNN based language model

2. Train the Elman network with BTT

3. Improve your Elman network with GRU

4. Implement a text classification model

```
"""This code is used to read all news and their labels"""
import os
import glob

def to_categories(name, cat=["politics","rec","comp","religion"]):
    for i in range(len(cat)):
        if str.find(name,cat[i])>-1:
            return(i)
    print("Unexpected folder: " + name) # print the folder name which does not in
    return("wth")

def data_loader(images_dir):
    categories = os.listdir(data_path)
    news = [] # news content
    groups = [] # category which it belong to

    for cat in categories:
        print("Category:"+cat)
        for the_new_path in glob.glob(data_path + '/' + cat + '/*'):
            news.append(open(the_new_path,encoding = "ISO-8859-1", mode ='r').rea
            groups.append(cat)

    return news, list(map(to_categories, groups))



data_path = "20Newsgroups_subsampled"
news, groups = data_loader(data_path)
```

Start coding or generate with AI.