Glove C++ User Guide

Version 1.0

October 1, 2015

Author: Jonathan A. Cox

Approved for Unclassified Unlimited Release (UUR), SAND No: SAND2015-8080 O

Contents

Introduction

GloVe C++ is an implementation of the GloVe algorithm for learning word vectors from a corpus. The details of this algorithm are described by Pennington, Socher and Manning:

> Pennington, Jeffrey, Richard Socher, and Christopher D. Manning. "Glove: Global vectors for word representation." Proceedings of the Empiricial Methods in Natural Language Processing (EMNLP 2014) 12 (2014): 1532-1543.

This implementation takes a modern, C++ approach. In particular, it leverages the Intel Thread Building Blocks (TBB) library for efficient parallelization when building a vocabulary and the co-occurrence matrix from a corpus. The TBB library also provides an efficient, thread safe hashmap for storing this data. In addition, GloVe C++ provides a vectorized *cost function* which uses the sparse matrix features in the Armadillo library to parallelize calculation of the gradient and cost/lost during gradient descent.

While this approach may or may not be faster than an efficient, multi-threaded implementation in C, it is nevertheless simpler to read, understand and modify. A vectorized implementation is also potentially capable of greater performance with a suitable matrix library, by exploiting modern CPU architectures.

Feature Requests or Bugs

If you have a feature request, or if you find a bug, you can contact the author at either:
jacox@sandia.gov or joncox@alum.mit.edu.

Compiling the Source Code

To compile GloVe, you will need the following libraries and compiler installed on your system:

- Armadillo 5.4 or greater
- Intel Thread Building Blocks 4.2 or greater
- Boost 1.55 or greater
- G++ compiler 4.9 or greater.

Simply run the `make` command from within the `Release` directory to compile GloVe.

Operation

Using GloVe C++ is divided into three stages:

1. Building a vocabulary from a corpus of text.
2. Building the co-occurrence matrix from a corpus of text.
3. Learning the word vectors from the co-occurrence matrix.

Corpus Format

GloVe C++ expects the corpus to be divided into multiple plain text files contained within a single directory. It reads in all files within the specified directory and assigns each one to a separate thread for processing. Tokenization of the words in the corpus is performed based on the following standard characters:

```
,.'\"@:!?$-_/%+&*;<>=()1234567890
```

This list can be modified by editing the definition for `TOKEN_DELIMETERS` near the top of VocabBuilder.hpp.

1. Building a Vocabulary

Before building the co-occurrence matrix, GloVe C++ requires that a dictionary of words be computed from the corpus. This vocabulary is assumed to be a subset of all of the words in the corpus. Please keep in mind that the vectorized training in GloVe C++ requires that large vocabularies be divided into subsets for computation. In other words, if the vocabulary exceeds 20,000 words, it will be divided into equal chunks for training. So the vocabulary size should ideally be a multiple of 10,000 words (e.g. 50,000; 100,000; or 120,000). If an unusual size is chosen, the GloVe may either fail or to train or divide the matrix into very small chunks, causing slow training. Nevertheless, GloVe will attempt to divide the matrix into the largest suitable chunks.

To build the vocabulary, use the following commands, where <vocab_size> is an integer specifying the size of the vocabulary, and /path/to/corpus_directory is the path to the directory containing the text files to use for training:

```
./GloVe –v  –s <vocab_size> -c /path/to/corpus_directory
```

GloVe will write out the vocabulary list to a CSV file named `vocabList.csv`.

2. Building the co-occurrence Matrix

Next, instruct GloVe to load the vocabulary list and build up the co-occurrence matrix using:

```
./GloVe –x –c /path/to/corpus_directory
```

This will write out a sparse matrix representation of the co-occurrence matrix to `X.arma`. Saving this matrix can take a little while with the present version of Armadillo.

Note, if you wish to inspect this matrix, you can print the contents by adding the line `cout << X;` to `CoMat::writeX()` in CoMat.cpp, and piping the output of GloVe to a file with ./Glove > X.txt.

3. Training the Word Vectors

The next step is to train the word vectors from the co-occurrence matrix contained in `X.arma`. To do so, the following, where `–e` and `–xfile` arguments are optional:

```
./GloVe –t [-e <learning_rate>] [--xfile /path/to/X.arma]
```

For example: `./Glove –t –e 0.01 –x /home/user/X.arma`

GloVe will attempt to break the co-occurrence matrix up into smaller sub-matrices for training. If you chose an unusual size for the vocabulary, such as 117,452 or 48,733, GloVe won't be able to break up the matrix into suitable, equal sized sub-matrices, and will report an error.

Moreover, you may need to set the learning rate and the momentum with the –e and –m arguments. Suggested initial values for learning rate are 0.01 and 0.8 for momentum.