Karlsruher Institut für Technologie

Functional Specification Document (FSD)

Numerical Linear Algebra meets Machine Learning

Fabian Koffer

Simon Hanselmann

Yannick Funk

Dennis Leon Grötzinger

Anna Katharina Ricker

Supervisors

Hartwig Anzt Makus Götz

13. November 2018

1 Success Criteria

Goal is the delivery of a consistent software stack that allows for employing neural networks for the linear system classification. The ecosystem should allow to train a neural network on selecting a suitable iterative solver depending on the linear system characteristics.

1.1 Mandatory Requirements

- A software that supports the described work-flow design including the embedding of external components.
- The software must be cross-platform compatible and support at least a Linux and the Windows operating system.
- The software must be usable via a command-line interface (CLI).
- A data exchange format design that allows to store matrices and annotate them with additional meta-data, including labels.
- An extensible design for multiple entities that are able to generate matrices in the proposed exchange format.
- There need to be two actual realizations of these entities, which:
  - allow to generate artificial noise with uniform and gaussian noise as well as
  - can fetch test matrices from the Suite Sparse matrix collection.
- A dataset of at least 500 matrices in the envisioned data format and generated by the above two entities. There smallest share of matrices of a given entity must be no less than 30% of the total number of contained matrices.
- An extensible design that allows to solve the matrices using a configurable set of iterative solver algorithms using a newly developed binding to the Ginkgo linear algebra library.
- A readily implemented and trained neural network of the ResNet architecture. It must be able to predict for a given matrix (in arbitrary format), which of the iterative solver algorithms is the most suitable.

- An entity that allows to store and load the trained neural network.
- The software must include entities for training and re-training a neural network from scratch, respectively from a previously stored state.
- The software must be able to show the predicted algorithm and its associated suitability probability on the standard output.
- Realization of a sustainable and quality-assured software development process. This includes a software design document, in-code documentation, unit testing and a continuous integration (CI).

1.2 Optional Requirements

- Scalability of the work-flow including matrix generation, training, prediction in that multiple processors may be used in parallel.
- The software must be able to utilize GPU accelerators for the training and prediction capabilities of the neural network.
- The system must support at least five iterative solver algorithms.
- A web interface to the software that is able to select a single, a set or all matrices of an uploadable file for prediction by the neural network. The web interface may also be able to visualize the contained matrices, annotated labels as well as prediction results.

1.3 demarcation criteria

2 Product use

2.1 Scope of application

the software should be used for scientific work in the field of math and computer science

2.2 Target groups

mathematican and computer scientists for classification of matrices

2.3 Operating conditions

training and test data need to be labeled

3 Product enviroment

3.1 Software

- Python
- Gingko
- pytest
- Keras
- ssget

3.2 Hardware

Scc server will be used to train the program. Linx and Windows PCs will be used for programming and testing.

3.3 Orgware

the documentation will be supported by doxygen

3.4 Product interface

the interface will be the comand line

4 Functional requirements

4.1 Matrix Generation

- /F10/ Generation of sparse matrices by a given sparsity level and size and metric
- /F15/ Generation of a given amount of matrices
- /F30/ Putting noise on the matrices
- /F40/ Saving the generated matrices in a given directory

4.2 Matrix Labeling

- /F50/ Choice of data origin(local or ssget tool)
- /F60/ Integration of the ssget tool
- /F70/ Determination of the best solving algorithm by time(fix algorithms)
- /F71/ optional: Determination of the best solving algorithm by time with costum algorithms
- /F75/ Labeling the matrix with the determined best algorithm
- /F90/ Creating a grayscale sparsity pattern image of the labeled matrix
- /F100/ Saving the labeled matrix with its sparsity pattern image in a given directory

4.3 DNN Training and Testing

- /F110/ Input of matrix files from a given directory
- /F120/ Randomization of the matrix files order
- /F125/ Separation of the matrix files into a training and testing dataset
- /F130/ Existence of a neural network to train

- /F140/ Training of the neural network by a given training dataset(/F125/)
- /F141/ Testing of the neural network by a given testing dataset(/F125/)
- /F150/ Printing the accuracy(/loss) during the training and testing process of the neural network
- /F151/ optional: creating of accuracy histogramms
- /F160/ Saving the neural network in its current state on a given directory

4.4 Matrix Classification

- /F170/ Input of a matrix to classify
- /F175/ Creating a grayscale sparsity pattern image of the input matrix
- /F180/ Loading a trained neural network from a given directory
- /F200/ Classification of the given pattern image by the neural network
- /F201/ Printing the classification output

5 Product data

- 500-1000 matrices for training, validation and test

6 Nonfunctional requirements

- /NF10/ All matrices are squared
- /NF20/ All fixed algorithms are used within the labeling processes
- /NF30/ The grayscale sparsity pattern all have the same size before training
- /NF40/ The NN ouputs a distinct prediction

- /NF50/ The NN is saved after every iteration

7 Global test cases

7.1 the following function sequences must be checked

7.2 the following data consistencies must be checked

8 System models

8.1 Scenarios

8.2 Use cases

8.3 Object models

8.4 dynamic models

8.5 Command line options

9 Glossar