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 will be used for scientific work in the field of math and computer science.

2.2 Target groups

Mathematican and computer scientists who are working with sparse linear systems.

2.3 Operating conditions

- Use in the field of scientific work
- Office environment

3 Product enviroment

3.1 Software

- The product will run on Windows 10 and Linux distributions
- The labeling of the matricies and training of the neural network will be done with Linux

3.2 Hardware

- The product will run on a workstation computer
- The labeling of the matricies and training of the neural network will be done on a server with multiple GPUs

3.3 Orgware

A Documentation for the user will be generated.

4 Functional requirements

4.1 Artificial matrix generator

4.2 Matrix label

4.3 Matrix train and test

4.4 Matrix classification

5 Product data

- 500-1000 matrices for training, validation and test

6 Nonfunctional requirements

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

8.1.1 User

The User safes the sparse linear matrix in any desired filepath. Afterwards the user starts our program. The command open <filepath> loads the matrix in the program. That is when the neural network will classify the matrix and determine the fastest iterative solver/preconditioner combination for solving the matrix. This combination will be printed to the command line. (**Optional:** After determining the fastest combination the program will solve the matrix with this combination. The solved matrix will be safed in a directory the user specifies.)

8.2 Use cases

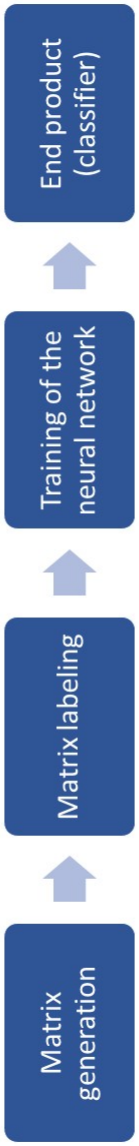
8.3 Object models

8.4 dynamic models

8.5 Command line options

8.6 Use of the classifier

8.7 Workflow



9 Glossar