Project information

Project name: **Toolbox for managing the training neural networks**
Project author: Pyry Takala

This document is based on the combination of:

- The original project introduction PDF by the PO
- Material delivered by the PO via email
- Notes made during Skype discussions with the PO
- Information accumulated from the Internet

Introduction

Deep learning is one of the newest trends in machine learning. Instead of specifying features that a machine should learn, neural networks can learn these features from data. Recent breakthroughs of deep learning include state-of-the-art image classification algorithms, computers playing Atari-games above human-level and flexible tools for analyzing natural language. Training of neural networks is often challenging, with many practical difficulties:

- how do we know what is happening inside a neural network?
- Where is the network making errors?
- How is the training of the network proceeding?
- How do we manage a queue of different experiments?
- Why is training the network too slow?

State of the art

Deep learning research is conducted by modelling neural networks and other constructs using specific deep learning libraries. Such libraries include:

- Theano – A Python library that allows you to define, optimize, and evaluate mathematical expressions involving multi-dimensional arrays efficiently.
  - Lasagne – A lightweight library to build and train neural networks in Theano.
  - Pylearn2 – A machine learning library built on top of Theano.
  - Blocks – A framework to facilitate building and managing neural network models using Theano.
- Torch – A scientific computing framework with wide support for machine learning algorithms. See tutorial.
- There are lots of software for deep learning: Deep learning portal

Deep learning research involves a lot of computing which is why the deep learning researchers at Aalto currently utilize a few different methods to meet their computational needs:

- BYOM – *Bring your own machine* is the first way to test simple computations. However, the lack of computing power and preinstalled and configured tools as well as interference with other tasks required of the machine make this method an inpopular one.
- gpus – A group of servers with powerful GPUs maintained by Aalto CS Department (Simo Tuomisto, 3rd floor). These servers named `gpu1`, `gpu2`, and so on are accessible by SSH by anyone with an Aalto IT account. The department's sysadmins maintain the machines and install generally useful software on request. Researchers use `virtualenv` to manage their own libraries. Certain user directories that are automounted in Aalto work desktops are also mounted on these servers which facilitates file management.
- Triton – A computer cluster managed by Aalto for use by all Aalto researchers. It utilizes the Slurm queuing and task management system to distribute computing resources to researchers. In practice, researchers access a gateway server using SSH and then add their scripts to the queue with terminal commands (almost `slurm OPTIONS FILES`). A nasty aspect for deep learning researchers is that they might have little knowledge of when and how their experiment is progressing. Similar to the case with `gpu` some user directories are also available here.
- CSC – Another cluster managed by the Finnish IT center for science. From Aalto researchers' point of view, it is basically similar to `triton` but harder to access and with no filesystem shares.
- amazon – A commercial choice for cloud computation. When Pyry was doing research at Amazon he realized their systems were not fancier than those at the Aalto CS Department. One could hire servers like the `gpus` start an SSH session in one and start coding.

In addition to advanced cluster computing environments and schedulers such as Slurm, OGE or Jobman researchers use simpler job scheduler and experiment manager tools created or under development in deep learning labs around the world. However, most of them are sort of script like solutions to serve immediate needs. A better tool could be built to replace them:

- LadderNet – Looked a bit complicated at first glance.
- soteloplot – A script that sends email with plots
- sacred – Seemed a bit complicated at first glance.
- Checkpoint – Allegedly a pretty feature complete tool.
- Jobman – A tool to facilitate launching concurrent experiments

Project goals

A good toolbox for deep learning would let a researcher easily specify experiments, manage a queue of experiments, and automatically monitor networks during training. During train-time, a diagnostics toolbox can perform various analyses on the training log and network parameters to detect possible problems early on. Notifications can be sent to a researcher so that expensive computing time is not wasted on an experiment that is unlikely to give good results. Naturally, the tool should not create a huge computational overhead, and usability should be good. Majority of the work involves creating a tool that helps a user define network inputs, manage an experiment queue, and visualize intermediary and final values of the network. The final product will be a tool for training neural networks that should be agnostic of the deep learning framework (e.g. frameworks *Torch* or *Theano* could both be used), and can benefit any neural networks researchers. Requirement gathering could be done from various deep learning users at the university, and potentially students could talk to some researchers abroad as well.

It is also important to be able to monitor host resources (GPU memory, RAM, disk-space). Ideally it should check already before starting whether there is enough diskspace available. Also consider profiling (python, theano, etc).

In addition to the above, these requirements have been explicitly mentioned by deep learning researchers:

- Sends customizable email notification (validation/test results, training curve/crash report)
- Queue and schedule lots of experiments with slightly different settings to a cluster or just your local network
- Should be lightweight (depend only on the python standard lib)
- Easy resume capability (load parameters and hyperparameters from disk and continue after a crash/failure)
- Autocheck the code repo's git version and other facts
- Record the history of "variables of interest" into a json file (validation costs & validation error rates after each epoch)
- The experiment manager should as simple as possible
  - Should not rely on one toolchain
  - Jobman feels oppressive. NoSQL setup should work better
- Command line is king; programmatic access from Python is nice to have

Tools and technology

The toolbox could be written e.g. as an interactive web tool (e.g. with JavaScript) that could be always run on a server that manages the experiments.

Requirements for the students

The toolbox could be implemented for instance as a hybrid of JavaScript and Python. Some other language can be also considered if it appears more suitable for the task. Students participating in this project will need to be willing to read a little bit about neural networks, but a detailed view is not required. The interface of the tool should be created in English.

Legal Issues

Potentially, the resulting code could be released under an open-source license after the project. Signing the non-disclosure agreement (NDA) included in the Aalto's contract template is not required.

Client

Aalto University's Deep learning and Bayesian modeling group conducts research in the field of neural networks. Recent projects include for instance *DeepBeat* a neural network that generates rap, an image-processing framework network *Ladder* that learns to recognize images from very small training datasets, and an ongoing project of financial predictions.

The toolbox would ideally be used by all researchers at Aalto and also researchers at other universities and companies. The student will have a good opportunity to get to know the field of machine learning and deep learning during the project.

Client representative

- Name: Pyry Takala
- Role: Researcher
- Organization: Aalto University

Additional notes

About deep learning

- Its a reborn topic.
- Less than 10 researchers do neural networks research at Aalto. Namely Pyry Takala, Antti Rasmus, Mathias Berglund, Jelena Lukatina.
- Some top companies like Google, Microsoft, Siri, Amazon (Echo) have started to research into the field. Notable are also Montreal LISA lab, Google DeepMind.

- The project could be useful to many deep learning labs and people in other fields running time consuming experiments (e.g. physics simulations).

Materials

- Specify experiments: rnn_experiments.xlsx
- Manage queue
  - rnn_experiments.xlsx
  - experiment1.slurm
- Monitor experiments
  - notification (e.g. email) if X
  - saving & loading requirements
- Analyze (visualise) running experiments
  - error analysis – a plot of error by number of weight updates
  - time analysis
  - weight norm analysis (e.g. per layer)
  - Analyze (visualise) ready experiments
  - Recurrent neural networks – A long post with lots of interesting content.
  - error analysis per input feature
- Training log examples
  - log.txt
  - See Monitoring experiments in Pylearn2
- Other deep learning resources
  - Colah – a blog with demos and visualizations

Triton

- Triton Wiki
- Scientific computing in practice

Triton Networking

The cluster has two internal networks: Infiniband for MPI and Lustre filesystem and Gigabit Ethernet for everything else like NFS for /home directories and ssh.

The internal networks are unaccessible from outside. The only host available for user access is the front-end triton.aalto.fi.

All compute nodes and front-end are connected to DDN SFA10k storage system: large disk arrays with the Lustre filesystem on top of it cross-mounted under /triton directory. The system provides about 430TB of disk space available to end-user.

Usage

- Running programs on Triton

Existing tools

SLURM

- Slurm at llnl.gov
- Wikipedia: Slurm Workload Manager

Simple Linux Utility for Resource Management (SLURM) is an open source, fault-tolerant, and highly scalable cluster management and job scheduling system for large and small Linux clusters.

Usage

- Interactive use
- Summary of CLI usage

Notes

- salloc obtain job allocation
- srun obtain a job allocation and execute an app
- sbatch [options] script [args...]
- sbatch submits a batch script to SLURM. By default redirects output to "slurm-%j.out" ("%j" is the job allocation number).

Example

```
#!/bin/bash
#SBATCH -J LM_miuc            # Specify job name
#SBATCH -o miuc_%j.out        # Specify output file name
#SBATCH -p gpu                # Request a specific partition for the resource allocation
#SBATCH --gres=gpu:2          # Generic resources required by each node
#SBATCH --mem-per-cpu=16000   # Memory required per allocated CPU.
#SBATCH -t 1-23:59:59         # Time limit
```

```
module load python-env
cd ../owlet
echo "Starting the run"

# launch experiment
python lang_experiment.py
```

Technical overview As a cluster resource manager, Slurm has three key functions:

- Allocates exclusive and/or non-exclusive access to resources (compute nodes) to users for some duration of time so they can perform work
- Provides a framework for starting, executing, and monitoring work (normally a parallel job) on the set of allocated nodes
- Arbitrates contention for resources by managing a queue of pending work.

Co-workability

- Our app could be a simple command line tool that would be installed to all the hosts where experiments are done
- The `SBATCH` script (or the python program launched by it) could utilize our tool to provide progress information (text, values, files) to a central server that we would also develop which would then provide the information to the researchers.

Example

```
#!/bin/bash
#SBATCH -J LM_miuc            # Specify job name
#SBATCH -o %j.out             # Specify output file name
module load python-env
# Send initial start notice along with system environment information
neronet --server neronet.cs.hut.fi --started exp01.py
# Launch the experiment
python exp01.py --iter=0-500
# Send progress information along with log and data files
neronet --server neronet.cs.hut.fi --progress exp01.py --log *.out --data *.json
# Continue experiment
python lang_experiment.py --iter=500-1000
# Send end state information along with log and data files
neronet --server neronet.cs.hut.fi --finished exp01.py --log *.out --data *.json
```

Oracle Grid Engine

OGE (previously known as SGE, Sun Grid Engine) is a grid computing computer cluster software system similar to Slurm. It manages grid computing resources and accepting, scheduling, dispatching, and managing the execution of large numbers of standalone, parallel or interactive user jobs.

A practical tutorial is available online.

Example

```
# Submit a job
qsub -V -b n -cwd runJob.sh
```

Jobman

Jobman aims to facilitate the process of launching many concurrent jobs, by automatically handling how parameters are passed to your programs and how results are stored for further analysis. While Jobman is application-agnostic, it is particularly well suited for machine learning when performing model selection (handling of hyperparameters, storing and analyzing results).

Notes

- requires python 2.7 to work on client side
- requires postgreSQL on server side
- client(s) running connects to server to retrieve a job to execute
- jobman does it's thing and outputs files that an be synced with rsync
- jobman updates database that job has been completed

Related information

Testing

Robot Framework is a generic test automation framework for acceptance testing and acceptance test-driven development (ATDD). It has easy-to-use tabular test data syntax and it utilizes the keyword-driven testing approach. Its testing capabilities can be extended by test libraries implemented either with Python or Java, and users can create new higher-level keywords from existing ones using the same syntax that is used for creating test cases.

Scrum & Backlog tools

- http://scrumfortrello.com/

- http://www.tommasonervegna.com/blog/2014/1/9/10-effective-tips-for-using-trello-as-an-agile-scrum-project-management-tool
- https://civicactions.com/blog/2012/oct/10/five_tips_for_using_trello_for_scrum
- https://trello.com/b/Nr3RvsY1/sprint-template
- http://www.screenful.me/blog/how-to-keep-your-product-backlog-clean-and-lean
- http://blog.flowdock.com/2013/08/16/trello-integration-added-to-flowdock/

Floobits & Sublime

- Provides realtime simultaneous file editing collaboration cababilities.
- Can be setup with a repo such that changes are easy to reflect back in GitHub.

See http://awan1.github.io/subl-floo-tutorial.html

Meeting notes

First meeting with PO

Written by Joona & Teemu

Questions we discussed

- importance of queue management functions
- technical details

Things we clarified

- requirements by the PO
- the current workflow

Decisions made

- Command prompt applications, no web UI
- Two different applications: Server and client
- Client application contacts server. The client application doesn't have to be running all the time.
- Server application contacts the clusters to run the experiments, manages the experiment queues and collects information on running experiments. The server also saves the information and notifies the client if the experiments go wrong. In an ideal situation the server application is always running.

- Queue management can use either jobman or slurm
- Server makes a .csv document about the past experiments and their outputs and the client can download it.
- Server can tell the information on the available GPU:s etc.
- Config:
- Defines the ID of the experiment (author, subject, name, group name, git commit ID)
- Define the variables that must be extracted and sent to the server
- The Preconditions (minimum available disk space, expected max time, minimum RAM)
- The files that must be sent to the cluster

Things yet to research/decide

- Simo
- Possibly in the future a web interface and user login

Roadmap

- TBD