CS230: Experimental Neural Net Framework (ENNF)

Video presentation: https://www.youtube.com/watch?v=6Nui1k9gJLc

Student: Konstantin Burlachenko (bruziuz@stanford.edu, burlachenkok@gmail.com)
Mentor: Steven Ziqiu Chen (stevenzc@stanford.edu)

Motivation

There are a lot of materials describes why *Deep Learning* is very powerfull hammer in various engineering application. Original motivation of *Deep Learning* community was to solve important problems more then theoretical justification.

Unfortunately all this bring us to theory-practice gap in this area. One of them lie in understanding more of phenomen of non-convex optimization. I talked with *Boris Polyak* (world expert in math optimization) 1 year ago and he said for me that unfortunately non-convex optimization in terms of fast methods has only reach situation with a lot of confusion when method works or not.

Personally I hope in future we will have more powerfull tools developed in area of intersection of ***optimization community*** and ***machine learning community***.
And I want to be part of it in future.

From other side there is a situation that prevalent programming language in Machine Learning is **Python**, and not C/C++ which is de-facto language in which almost all things in this planet have been written when we talk about speed.

Both this thing thing bring me to conclusion to create own ***framework*** in C++ which I hope will evolve in future.

I want give ability to use *AI/ML* tools not only for prototyping languages like *Python* or *Matlab*, but have ability to use them from C/C++ production code.

What I have implemented

Relative to software

1. General framework for people who is interesting in creating their own math optimization solvers
2. Various need things to orgranize work - load data from the disk to dense matrices and vectors, generate data with various p.d.f.
3. All is written in compilable languages C/C++11 and optionally exploits SSE2
4. Demonstrate how possible to do things outside *TensorFlow*.

Software implementation goal was:

1. Implement system in C/C++
2. Be buildable for Windows OS
3. Use standard tools for project files (CMake), for unit-tests (Google Test)

Relative to math

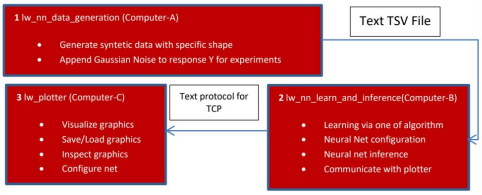
Solvers used during learning support various well-known first order methods including batched version of SGD, ADAM, RMSProp, Polyak heavy ball method.

Datasets

Dataset is synthetically generated via using *lw_nn_data_generation* program

Schema

Experimental Light Neural Net Framework consist of several parts:



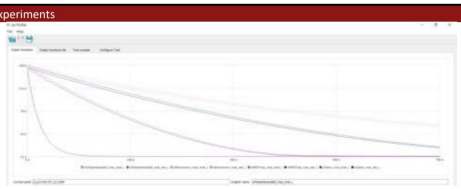
In fact there is a stack of technologies and very high level libraries will really not allow todo deep enhacements. Even interfaces is how Computer Science is evolving during last 70 years, but interfaces bring to misunderstanding limitations and possibility care un-obvious experiments.

Deep Neural Nets	
Numerical Optimization and Convex Optimization	Work with datastorages like SSD
Numerical Linear Algebra	
Basic Linear Algebra	
High level compilable languages like C++	
Assembly lanaguages allow tow work with hardware with some microcomands	
MicroElectronics	
Quantum Physics	

Inside different binaries different things have been implemented:

- optimization algorithms
- data loading
- forward and backward pass
- own BLAS library
- different weight initialization schemas,
- inititialize network with specific topology,
- network communication with plotter tool.

Experiments



Via using my framework it's possible to carry experiments and compare different optimization methods. Here in plot the following methods are shown
Polyak Heavy Ball method, RMSPRop method, ADAM method, Momentum method applied to learn parameters for neural net:
LINEAR -> RELU -> LINEAR -> RELU -> LINEAR -> RELU -> LINEAR
With configuration: 30, 5, 5, 3,1

Another experiment is measure learning speed of my framework with TF. Train set contain 30 features and 200 testing examples. Such configuration in my experience may arise in business applications in real life.

Method / Approach	CPU TensorFlow v1.13.1	Experimental Neural Net Framework
ADAM optimizer	1.115 seconds	0.306
SGD optimizer	1.105 seconds	0.298

Future Work if assume that I have more free time and References

1. Append support of Convolution Neural Nets
2. Append support of showing feature maps in plotter tool.
3. Append support of visualizing configured Neural Net via **graphviz**
4. Try improve YOLO: Real-Time Object Detection
5. Append NVIDIA CUDA support.

[1] CS230 class from Stanford University
http://cs230.stanford.edu/syllabus/
[2] Notes about Stochastics Gradient Descend from S.P.Boyd and J.C.Duchi
http://web.stanford.edu/class/ee364b/lectures/stoch_subgrad_notes.pdf