Software Packages for Deep Learning

Chensong Zhang

with Zheng Li and Ronghong Fan

DL Seminar — April 27, 2017

Outline

Machine Learning

- ML gives computers the ability to learn without being explicitly programmed [Samuel 1959]
- ML explores the study and construction of algorithms that can learn from and make predictions on data
- Data mining, computational statistics, optimization, ...
- Fourth paradigm, big data, deep learning, artificial intelligence

General Tasks of ML

- Classification: Inputs are divided into two or more classes, and the learner must produce a model that assigns unseen inputs to one or more (multi-label classification) of these classes
- Clustering: Inputs are divided into groups. Unlike in classification, the groups are not known beforehand, making this typically an unsupervised task
- Regression: Similar to classification, but the outputs are continuous rather than discrete
- Other tasks: density estimation, dimensionality reduction, ...

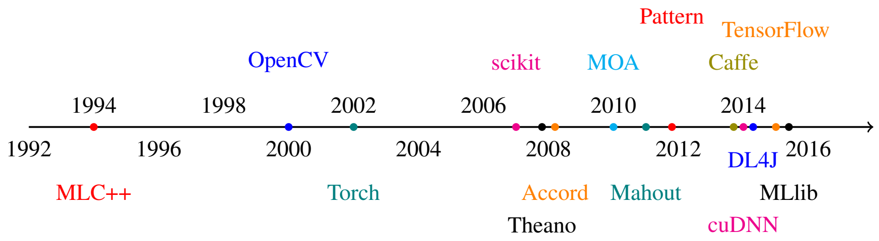
Packages for General Machine Learning

What is the purpose?

- Solving problems from practical applications (user interface)
- Developing algorithms and optimizing implementation (development)
- Theoretical analysis for machine learning

What do we want for a ML package?

- Easy for new tasks and new network structures (less steep learning curve)
- Easy for debugging (with good support and large community)
- Performance and scalability



Deep Learning

Comparison

Table: Framework Comparison: Basic information

Viewpoint	Torch	Caffe	TensorFlow	MXNet
Started	2002	2013	2015	2015
Main Developers	Facebook, Twitter, Google, ...	BVLC (Berkeley)	Google	DMLC
License	BSD	BSD	Apache	Apache
Core Languages	C/Lua	C++	C++ Python	C++ Python
Supported Interface	Lua	C++/Python Matlab	C++/Python R/Java/Go	C++/Python R/Julia/Scala

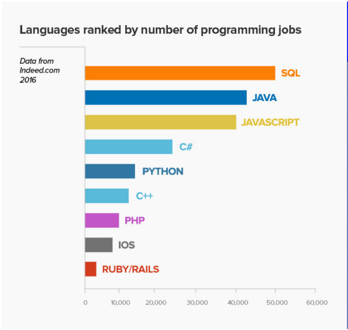
Comparison

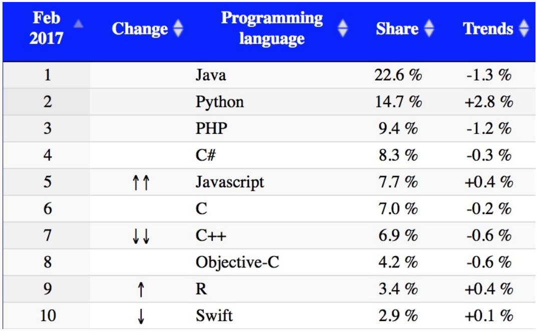
Table: Framework Comparision: Performance

Viewpoint	Torch	Caffe	TensorFlow	MXNet
Pretrained Models	Yes	Yes	No	Yes
Low-level Operators	Good	Good	Fairly good	Very few
High-level Support	Good	Good	Good	Good
Speed One-GPU	Great	Great	Not so good	Excellent
Memory Management	Great	Great	Not so good	Excellent
Parallel Support	Multi-GPU	Multi-GPU	Multi-GPU	Distributed

Python: A general-purpose programming language

- Created by Guido van Rossum in 1989 and first released in 1991
- Named after “the Monty Python” (British comedy group)
- An interpreted language—simple, clear, and readable
- Python has many excellent packages for machine learning
- The language of choice in introductory programming courses





Feb 2017	Change	Programming language	Share	Trends
1		Java	22.6 %	-1.3 %
2		Python	14.7 %	+2.8 %
3		PHP	9.4 %	-1.2 %
4		C#	8.3 %	-0.3 %
5	↑↑	Javascript	7.7 %	+0.4 %
6		C	7.0 %	-0.2 %
7	↓↓	C++	6.9 %	-0.6 %
8		Objective-C	4.2 %	-0.6 %
9	↑	R	3.4 %	+0.4 %
10	↓	Swift	2.9 %	+0.1 %

Python for Scientific Computing

Why Python for scientific computing?

- Strong introspection capabilities (???What does even mean???)
- Full modularity, supporting hierarchical packages
- Exception-based error handling
- Dynamic data types and automatic memory management

Why consider such a slow language for simulation?

- Good for proof-of-concept
- Implementation time versus execution time
- Code readability and maintenance — short code, fewer bugs
- Well-written Python code is “fast enough” for most computational tasks
- Time critical parts executed through compiled language or available packages

Built-in Data Structures

- Numeric types–int, float, complex, ex: a=1, b=1.0, c=1L, d=0xf, e=010, f=1+2j
- Sequence types–list, tuple, str, dict, ex: g=[3.14, True, ’Yes’, [1], (1L,)] + [False] + [None]*3, h=(3.14, True, ’Yes’, [1], ()), i=’Hello’ + ”,” + ”’world!”’, j={1: ’int’, ’pi’: 3.14}

Control Flow

- If-then-else
- For loop
- While loop

Functions and Modules

- Defining functions
- Using modules

Programming interface

Example 1

Programming interface

Example 1

Computational graph

Programming interface

Visualization

Example 1

Programming interface

Example 1

Numerical tests

Thank You!