PROGRAMMING IN PYTHON I

ML Modules in Python

Andreas Schörgenhumer

Institute for Machine Learning

JOHANNES KEPLER
UNIVERSITY LINZ

Institute for
Machine Learning

Copyright Statement

Contact

Andreas Schörgenhumer

Institute for Machine Learning
Johannes Kepler University
Altenberger Str. 69
A-4040 Linz

E-Mail: schoergenhumer@ml.jku.at
Write mails only for personal questions
Institute ML Homepage

MACHINE LEARNING IN PYTHON

Motivation

- Python is a go-to language for Machine Learning (ML) and Deep Learning (DL)
  - Implementation of research and production code is possible
  - Convenient usage supports quick implementation of ideas
  - There are modules that allow for fast execution on dedicated hardware

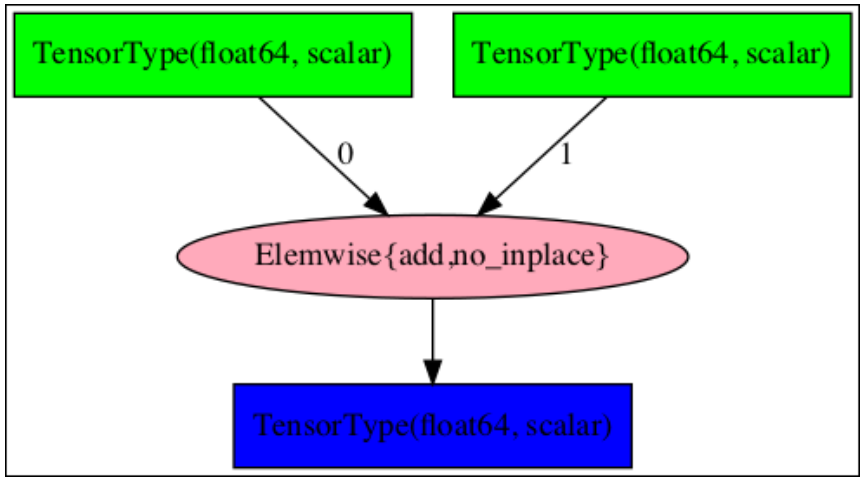
Motivation

- Python is a go-to language for Machine Learning (ML) and Deep Learning (DL)
  - Implementation of research and production code is possible
  - Convenient usage supports quick implementation of ideas
  - There are modules that allow for fast execution on dedicated hardware
- Now we will tap into modules that allow us to write optimized Python code for ML
  - Usage of dedicated hardware (CPU, GPU, TPUs)
  - Automatic differentiation (e.g., for training of DL networks)
  - Convenience functions for data loading, preprocessing, training, evaluation, ...
  - TensorFlow, **PyTorch**, ...

COMPUTATIONAL GRAPHS

Computational Graphs

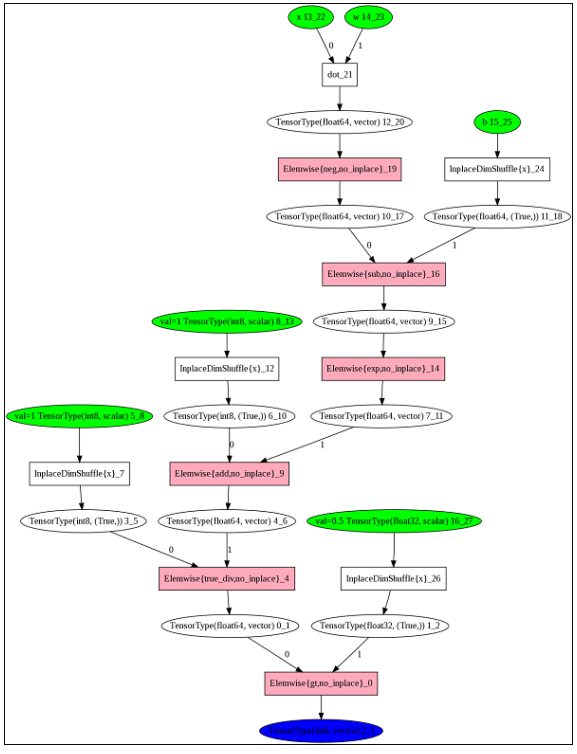
- A large part of the “magic” provided by frameworks such as TensorFlow/PyTorch is based on **computational graphs**
  - Symbolic graph of computations
  - Defines a pipeline of computations
  - Nodes in the graph can represent functions and placeholders for the data (=**tensors**)



Scalar addition as computational graph. Source: https://www.tutorialspoint.com/theano/theano_computational_graph.htm

Computational Graphs: Benefits

- Compilation of graph with optimization for dedicated devices or data types
- Automatic differentiation



More complex computational graph. Source: https://www.tutorialspoint.com/theano/theano_computational_graph.htm

AUTOMATIC DIFFERENTIATION

Automatic Differentiation

- When training artificial neural networks (NNs), we typically rely on gradient-based methods to update the weights
  - E.g.: Compute gradient of loss w.r.t. NN parameters to change parameters such that loss decreases
  - For deeper NNs (multiple layers), this relies heavily on the chain rule for differentiation

Automatic Differentiation

- When training artificial neural networks (NNs), we typically rely on gradient-based methods to update the weights
  - E.g.: Compute gradient of loss w.r.t. NN parameters to change parameters such that loss decreases
  - For deeper NNs (multiple layers), this relies heavily on the chain rule for differentiation
- Not having to implement the chain rule formulas by hand for every NN makes our lives significantly easier
  - Combined with modular layer design, this makes NN design and training almost plug-and-play

Automatic Differentiation

- When training artificial neural networks (NNs), we typically rely on gradient-based methods to update the weights
  - E.g.: Compute gradient of loss w.r.t. NN parameters to change parameters such that loss decreases
  - For deeper NNs (multiple layers), this relies heavily on the chain rule for differentiation
- Not having to implement the chain rule formulas by hand for every NN makes our lives significantly easier
  - Combined with modular layer design, this makes NN design and training almost plug-and-play
- Computational graph contains information about functions used to compute result and allows for automatic differentiation (This makes us really happy!)

PYTHON MODULES

Python Modules

- Computational graph created using Python code
  - After creation, data can be fed into the graph to compute output and gradients for updates

Python Modules

- Computational graph created using Python code
  - After creation, data can be fed into the graph to compute output and gradients for updates
- **PyTorch** (`https://pytorch.org`)
  - NumPy-like code to dynamically create graph (done on-the-fly, no explicit PyTorch commands required)
  - Computational graph is evaluated automatically when result is used

Python Modules

- Computational graph created using Python code
  - After creation, data can be fed into the graph to compute output and gradients for updates
- **PyTorch** (https://pytorch.org)
  - NumPy-like code to dynamically create graph (done on-the-fly, no explicit PyTorch commands required)
  - Computational graph is evaluated automatically when result is used
- TensorFlow (https://www.tensorflow.org)
  - Explicit creation of static computational graph using TensorFlow commands in Python code
  - Evaluation of graph explicitly via TensorFlow commands

Python Modules

- Computational graph created using Python code
  - After creation, data can be fed into the graph to compute output and gradients for updates
- **PyTorch** (https://pytorch.org)
  - NumPy-like code to dynamically create graph (done on-the-fly, no explicit PyTorch commands required)
  - Computational graph is evaluated automatically when result is used
- TensorFlow (https://www.tensorflow.org)
  - Explicit creation of static computational graph using TensorFlow commands in Python code
  - Evaluation of graph explicitly via TensorFlow commands
- PyTorch and TensorFlow both provide many convenience functions for ML