PROGRAMMING IN PYTHON II

TorchScript

Michael Widrich
Institute for Machine Learning

JOHANNES KEPLER
UNIVERSITY LINZ

Institute for
Machine Learning

Outline

MOTIVATION

Motivation

- PyTorch allows us to:
  - Optimize Python code using a dynamic graph and just-in-time (JIT) compiler
  - Keep the flexibility of Python code while developing ML solutions
- Downside: Flexibility of Python still restricts the performance (e.g. Python for-loops)

Motivation

- PyTorch allows us to:
  - Optimize Python code using a dynamic graph and just-in-time (JIT) compiler
  - Keep the flexibility of Python code while developing ML solutions
- Downside: Flexibility of Python still restricts the performance (e.g. Python for-loops)
- Modules like `numba` and `Cython` (Programming in Python I) allow us to optimize Python code
  - Either restrict set of Python commands available or combine Python and C code

TORCHSCRIPT

TorchScript (1)

- TorchScript allows us to write Python/PyTorch for
  - Better optimization and performance
  - Deployment in Python and high-performance environments like C++ code
- Idea:
  - Develop ML algorithms and solutions using flexible Python/PyTorch code, optionally with restricted vocabulary
  - Export trained model to production environment (e.g. C++)

TorchScript (2)

- `TorchScript` creates serializable and optimizable models from PyTorch code
  - Restrictions on set of Python commands allows for better optimization
  - TorchScript program can be saved from a Python process and loaded in a process where there is no Python dependency
  - Not restricted by Python performance and multi-threading limitations
- TorchScript is an intermediate representation of a PyTorch model
  - Subclass of `torch.nn.Model` and can be used as such in Python code

SCRIPT AND TRACE

Script vs. trace

- 2 options to turn PyTorch code into `TorchScript` program

Script vs. trace

- 2 options to turn PyTorch code into `TorchScript` program
- Scripting
  - Compile Python function or `torch.nn.Model`
  - Python commands like if-else statements, loops, and variables will be compiled as `TorchScript` program
  - Consequence: Restricted set of Python and PyTorch commands

Script vs. trace

- 2 options to turn PyTorch code into `TorchScript` program
- Scripting
  - Compile Python function or `torch.nn.Model`
  - Python commands like if-else statements, loops, and variables will be compiled as `TorchScript` program
  - Consequence: Restricted set of Python and PyTorch commands
- Tracing
  - Tracing records the execution of a Python/PyTorch function given an example input
  - `TorchScript` program will be created/optimized from this record
  - Conditionals on data in tensors becomes static
  - Consequence: Conditional Python code (if-else statements or loops) will always follow same path as example input

Scripting a function

- Scripting functions is done via `torch.jit.script`
  - As function for functions and modules
  - As decorator for functions and `TorchScript Classes`

```
1 import torch
2 @torch.jit.script
3 def foo(x, y):
4     return x + y
```

```
1 import torch
2 def foo(x, y):
3     return x + y
4 foo_s = torch.jit.script(foo)
```

TorchScript language

- Scripting a PyTorch program means writing a `TorchScript` program directly
- `TorchScript` language has to be used
  - Statically typed subset of Python
- Any valid `TorchScript` function is also a valid Python function
  - Can be debugged as Python code before scripting
- Scripting creates `ScriptFunction` or `ScriptModule` objects
- TorchScript language reference: `https://pytorch.org/docs/stable/jit_language_reference.html`

TorchScript language

- Tracing a PyTorch program means recording the operations of a Python function given some input
  - `torch.jit.trace`: record function to `ScriptFunction`
  - `torch.jit.trace_module`: record `nn.Module.forward` or `nn.Module` to `ScriptModule`
- No data-dependencies are recorded!
  - Control-flow decision based on data are ignored, only flow recorded when using the given input is followed

```
1 import torch
2 def foo(x, y):
3     return x + y
4 traced_input = (torch.rand(3), torch.rand(3))
5 foo_s = torch.jit.trace(foo, traced_input)
```

Scripting or tracing?

- Depending on the use-case, scripting or tracing might be better suited

Scripting or tracing?

- Depending on the use-case, scripting or tracing might be better suited
- Use tracing if:
  - Control-flow can be be static and is independent of input
    - Sequence length for loop in RNN is always constant
    - Conditions used for building architecture should remain static after tracing

Scripting or tracing?

- Depending on the use-case, scripting or tracing might be better suited
- Use tracing if:
  - Control-flow can be be static and is independent of input
    - Sequence length for loop in RNN is always constant
    - Conditions used for building architecture should remain static after tracing
- Use scripting if:
  - `TorchScript` language can be used
  - Control-flow needs to be dynamic or dependent on input
    - Sequence length for loop in RNN is variable

SCRIPTFUNCTION AND SCRIPTMODULE

ScriptFunction and ScriptModule (1)

- Scripting and tracing produce `ScriptFunction` or `ScriptModule` objects
  - Subclasses of `torch.nn.Module`
- Scripting and tracing can be combined
  - `ScriptFunction` and `ScriptModule` can contain other `ScriptFunction` or `ScriptModule` objects

ScriptFunction and ScriptModule (2)

- TorchScript modules can be saved to/loaded from disk in an archive format
  - Freestanding representation of the model, including code, parameters, attributes, debug information
- In Python:
  - `foo_s.save('filename.pt')`: save module
  - `torch.jit.load('filename.pt')`: load module
  - `foo_s.graph`: low-level representation of TorchScript code
  - `foo_s.code`: Python-syntax interpretation of TorchScipt code

FURTHER READING

Scripting a function

- Official documentation: `https://pytorch.org/docs/stable/jit.html`
- Tutorial: `https://pytorch.org/tutorials/beginner/Intro_to_TorchScript_tutorial.html`
- Tutorial for export to C++: `https://pytorch.org/tutorials/advanced/cpp_export.html`
- Parallel code execution: `https://pytorch.org/docs/stable/generated/torch.jit.fork.html`
- Known issues: `https://pytorch.org/docs/stable/jit.html#known-issues`