Deep learning with PyTorch

Elements of Machine Learning

Jens Petersen

with content from Deep Learning with PyTorch,
Eli Stevens and Luca Antiga

About this lecture

- Covering content from Deep learning with PyTorch, Eli Stevens and Luca Antiga, chapters 1-5
- further reading at `http://pytorch.org/docs` and `https://pytorch.org/tutorials/beginner/deep_learning_60min_blitz.html`

We assume familarity with Python, NumPy...

Deep learning with PyTorch

What is PyTorch

- Python library which
  - facilitates deep learning
  - allows easy use of both GPUs and CPUs (switching requires nothing more than a function call)

Why PyTorch as opposed to TensorFlow?

- (+) More pythonic and PyTorch tensors are similar to NumPy arrays, so may be easier to learn, use, extend, and debug (opinions are like ...)
- (+) Dynamic graph as opposed to static graph (TensorFlow 2.0 also has dynamic graphs)
- Adopted more in research and less in industry

Immediate versus deferred execution (Dynamic/Static)

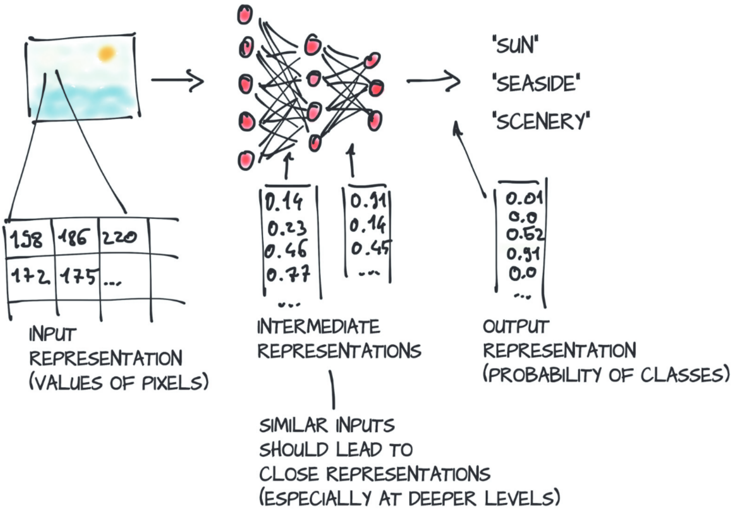
Immediate execution / dynamic computation graph / eager mode

- When problems arise, interpreter and debugger have direct access to Python objects involved.

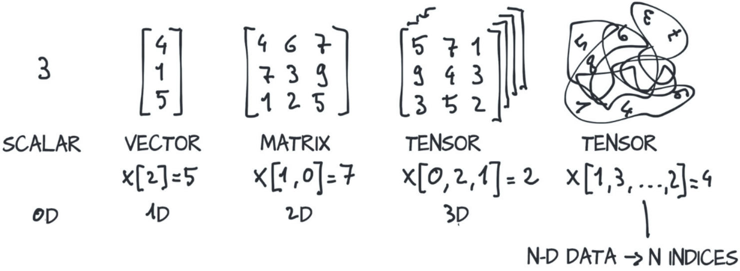
Deferred execution / static computation graph

- Code can be compiled to machine code (performance)

Handling data in DL networks - PyTorch tensors

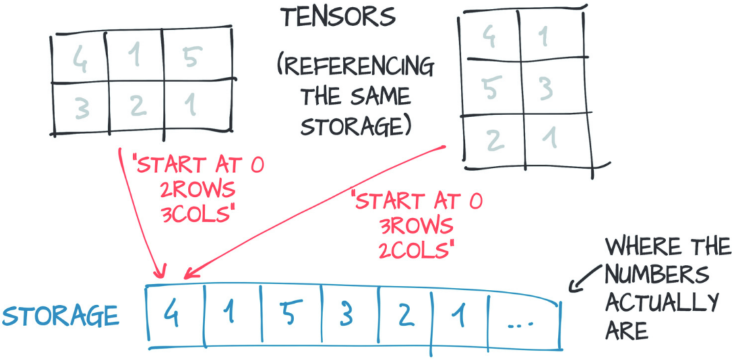


PyTorch tensors

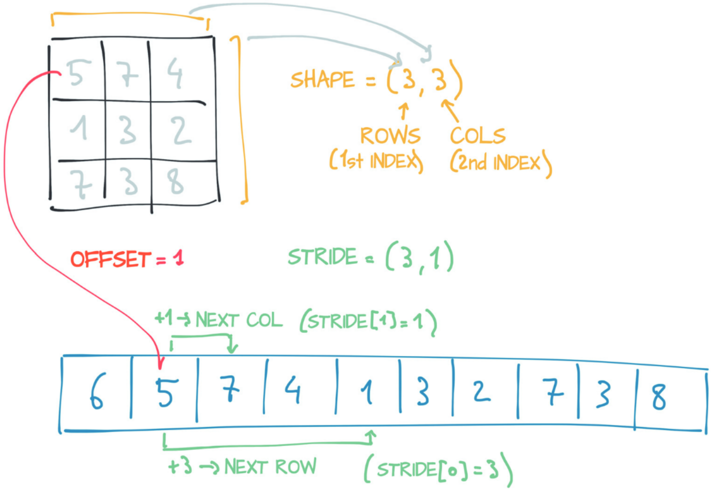


- Seamless interoperability with NumPy
    - Allows easy integration with existing scientific libraries (SciPy, Scikit-learn, Pandas...)

PyTorch tensors - views over a Storage instance

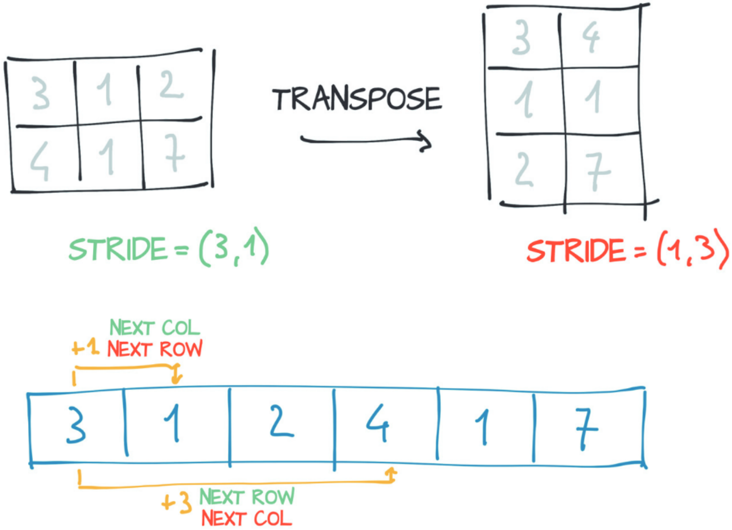


PyTorch tensors - views over a Storage instance



Open `1_tensors.slides.html`

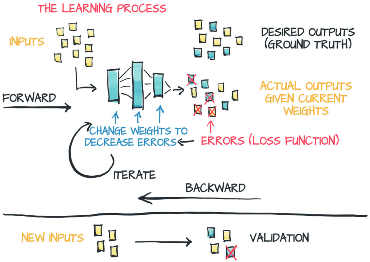
PyTorch tensors - transpose example



Learning with PyTorch

Given input and groundtruth output, iterate over

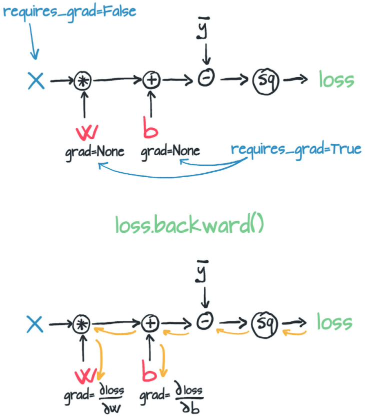
- forward pass: compute error given weights
- backward pass: compute how much we need to change each weight to decrease error.
- update weights accordingly



Learning with PyTorch

Given input and groundtruth output, iterate over

- forward pass: compute error given weights
- backward pass: compute gradients, or partial derivatives of the loss with respect to each parameter, w and b in this example.
  - In PyTorch, these are stored in the .grad attribute of each tensor
- update weights accordingly



Autograd

Gradients are computed automatically using Autograd.

- Graph (DAG) recording all operations that created the data as you execute operations
  - forward pass
  - requires_grad = True
- By tracing this graph from roots to leaves, gradients are computed using the chain rule
  - backward_pass
- Note the graph is created from scratch in every iteration
  - What you run is what you differentiate!

Open `2_autograd.slides.html`

PyTorch optimizers and training loop

PyTorch comes with optimizers, open `3_optimizers.slides.html` for an example of how to use them.

Neural networks in PyTorch

Critical to implementing neural networks are activation and loss functions.

Open `4_neural_networks.slides.html`

Convolutional neural networks in PyTorch

We will go through the example from the PyTorch 60 minutes tutorial `https://colab.research.google.com/drive/1B-NPM6i6U0sU_bFQzDmMN27ZqTpmgMwC`.

Questions?