Practical 1.1

An overview on Torch's Tensors

Torch (I)

- http://torch.ch
- Torch is an easy to use and efficient scientific computing framework which leverage LuaJIT, and an underlying C/CUDA implementation
  - a powerful N-dimensional array
  - amazing interface to C, via LuaJIT
  - neural network, and energy-based models
  - Fast and efficient GPU support
  - Embeddable, with ports to iOS, Android and FPGA back-ends

Torch (II)

- http://torch.ch
- Torch goals:
  - Maximum flexibility
  - Highest speed
  - Extreme simplicity

Overview (I)

- Generic help
- `? torch.<item>`
- `torch.type()`
- `torch.Tensor()`
- `#` operator, `:dim()` and `:size(dim)`
- `:apply()`
- `Tensor`s types (Byte, Char, Short…)
  - `torch.setdefaulttensortype()`

Overview (II)

- `Tensor`s and `Storage`
- `:resize()`
- `Tensor`'s type mismatch error
- The `=` between `Tensor`s, and `:clone()`
- Vectors: 1D `Tensor`s, and `*`
- Matrices: 2D `Tensor`s, and `*`
  - Row and column vectors
- Slicing with the `[{ … }]` operator

Overview (III)

- `Tensor`s constructors
  - `torch.range()`
  - `torch.linspace()`
  - `torch.logspace()`
  - `torch.zeros()`
  - `torch.ones()`
  - `torch.eye()`
  - `torch.rand()`
  - `torch.randn()`
- Cast of `Tensor`s

Overview (III)

- Charts with `gnuplot`
  - `gnuplot.plot()`
  - `gnuplot.hist()`
- Element wise multiplication with `torch.cmul()`
- Transposition with `:t()` and `:transpose()`
- Concatenation with `torch.cat()`
- `+` and `*` over `:add()` and `:mul()`
- `:resize()`, `:reshape()` and `:view()`