Modular toolkit for Data Processing

a Python data processing framework

Pietro Berkes, Niko Wilbert, Tiziano Zito

NIPS 2008
Workshop on Machine Learning Open Source Software
December 12th, 2008

http://mdp-toolkit.sourceforge.net
Building blocks: Node

- dtype, dimensionality
- training: multiple phases, batch, online, chunks, supervised, unsupervised
- execution
- inversion

```python
>>> pca = PCANode(output_dim=0.9, dtype='float32')
>>> for x in data_stream:
...     pca.train(x)
>>> out = pca(x)
>>> rec = pca.invert(out)
```
Building blocks: Node

- PCA (standard, NIPALS)
- ICA (FastICA, CuBICA, JADE, TDSEP)
- Locally Linear Embedding
- Hessian Locally Linear Embedding
- Fisher Discriminant Analysis
- Slow Feature Analysis
- Independent Slow Feature Analysis
- Restricted Boltzmann Machine
- Growing Neural Gas
- Factor Analysis
- Gaussian Classifiers
- Polynomial Expansion
- Time Frames
- Hit Parades
- Noise
- ...

http://mdp-toolkit.sourceforge.net
Building blocks: Flow

- automatic: training, execution, inversion
- sanity checks
- Python containers
- feed on arrays or iterators
- crash recovery, checkpoints

```python
>>> flow = PCANode() + SFANode() + FastICANode()
>>> generator = (chunk for chunk in openfile)
>>> flow.train(generator())
>>> out = flow(x)
>>> rec = flow.invert(out)
>>> flow += HitParadeNode()
```
Building blocks: Network

- Layer
- Switchboard
- HTML display

```python
>>> layer = Layer([PCANode(), ICANode(), ...])
>>> switch = Switchboard(input_dim=N, ...
...       connections=[0, 3, 1, ...])
>>> net = switch + layer
>>> net.train(generator())
>>> out = net(x)
```
Speed: let's go parallel

- embarrassingly parallel problems
- scheduler
- multiple processors
- multiple machines
- automatic parallelization of serial flows
- abstract scheduler API
- support for Parallel Python

```python
>>> flow = PCANode() + SFANode()
>>> scheduler = ProcessScheduler(n_processes=8)
>>> pflow = make_flow_parallel(flow)
>>> pflow.train(data, scheduler)
```
Users

- ML and neuroscience (>12K downloads):
  - modeling, computer vision, pattern recognition,
  - electrophysiological data analysis, education, ...

- comprehensive documentation:
  - tutorial covering basic and advanced usage
  - public objects doc-strings
    - PEP8 compliant, commented, and pylint-clean code

- collection of efficient and well tested (350+ unit tests) algorithms

- minimal dependencies: Python + NumPy
Embedding MDP

- input and output just NumPy arrays
- API is stable and designed for straightforward embedding
- PyMCA: X-ray fluorescence mapping
- PyMVPA: ML framework for neuroimaging data analysis
- Chandler: personal organizer application
Developers

- framework to develop new supervised and unsupervised algorithms
- concentrate on the algorithm, MDP takes care of the details
- use MDP utilities in your nodes
- immediately integrate your nodes with the existing library
- contribute your nodes to MDP!

```python
>>> class MyNode(Node):
...     def _train(self, x):
...         ... training code ...
...     def _execute(self, x):
...         ... execution code ...
... >>> flow = PCANode() + MyNode()
```
Future perspectives

- Architecture:
  - feedback loops
  - Python 3

- Algorithms:
  - involve more external contributors
  - integrate widely used libraries
http://mdp-toolkit.sourceforge.net