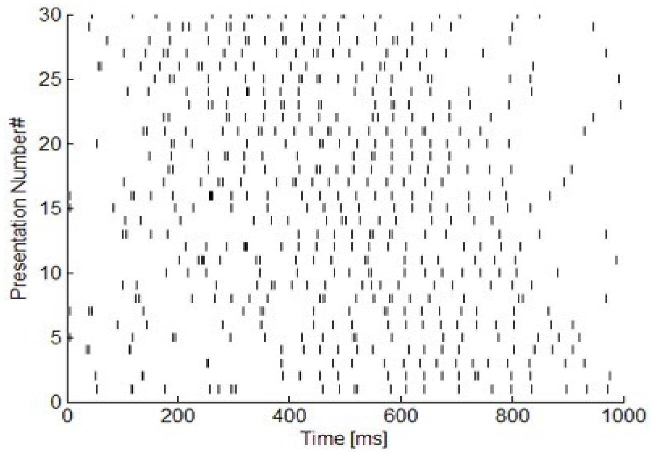
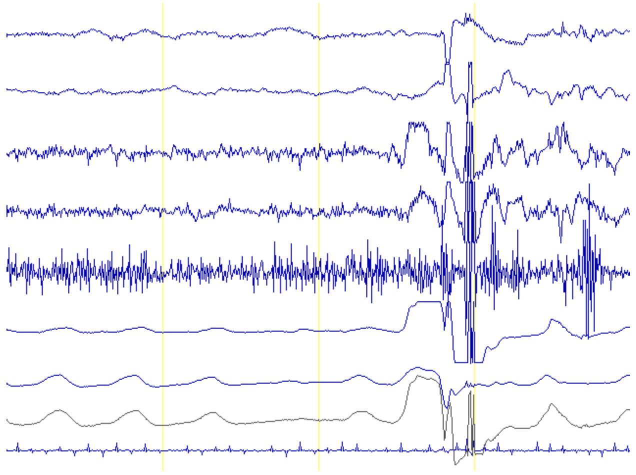
Modular Toolkit for Data Processing

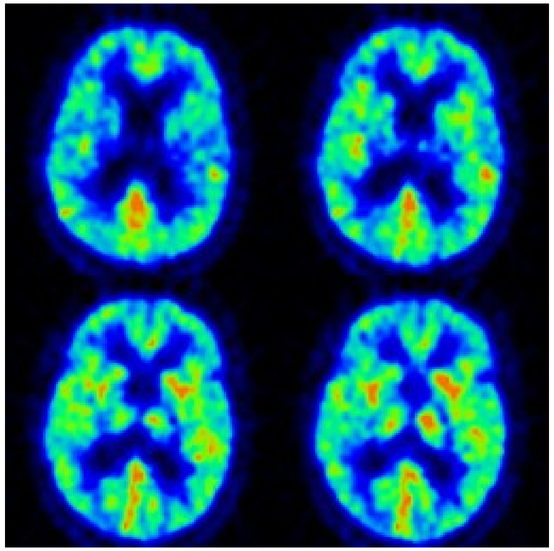
03.07.2006

Pietro Berkes & Tiziano Zito

Data Processing in Neuroscience







MDP Main Features

- Data processing units (nodes)
- Data processing flows
- Static typing
- Several algorithms
- Easy to use and to extend
- Extensive documentation

MDP Building Blocks: Node

Data processing unit:

- Numpy dtype
- Input and output dimensions
- Training (batch, online, block-mode)

```
>>> node = mdp.PCANode(output_dim=10, dtype='f')
>>> for x in train_stream:
...         node.train(x)
...
>>> node.stop_training()
>>> out = node.execute(data)
>>> # helper function for one-shot train and exec
>>> out = pca(data)
```

MDP Building Blocks: Node

Some implemented nodes:

- Principal Component Analysis
- Independent Component Analysis
- Slow Feature Analysis
- Growing Neural Gas Network
- Fisher Discriminant Analysis
- Gaussian Classifiers
- Factor Analysis

MDP Building Blocks: Flow

Data processing sequence:

- Automatic training and execution
- Automatic sanity checks
- Use of iterators to receive input data

```
>>> flow = Flow([WitheningNode(output_dim=35),
...              PolynomialExpansionNode(3),
...              FDANode(output_dim=9),
...              GaussianClassifierNode()])
...
>>> flow.train([DataIterator(train_digits), ...])
>>> guess = flow.execute(DataIterator(test_digits))
```

MDP: Framework for Developers

Write your own nodes:

- Implement _train and _execute
- Integrate with existing library

```
>>> class MyNode(Node):
...      def _train(self, x):
...          train_code()
...      def _execute(self, x):
...          return execute_code()
...      def _get_supported_dtypes(self):
...          return ['f', 'd', 'i']
...
>>> flow = Flow([FastICANode(), MyNode()])
>>> flow.train([generatorICA(), generatorMyNode()])
>>> out = flow(data)
```

MDP: Additional Features

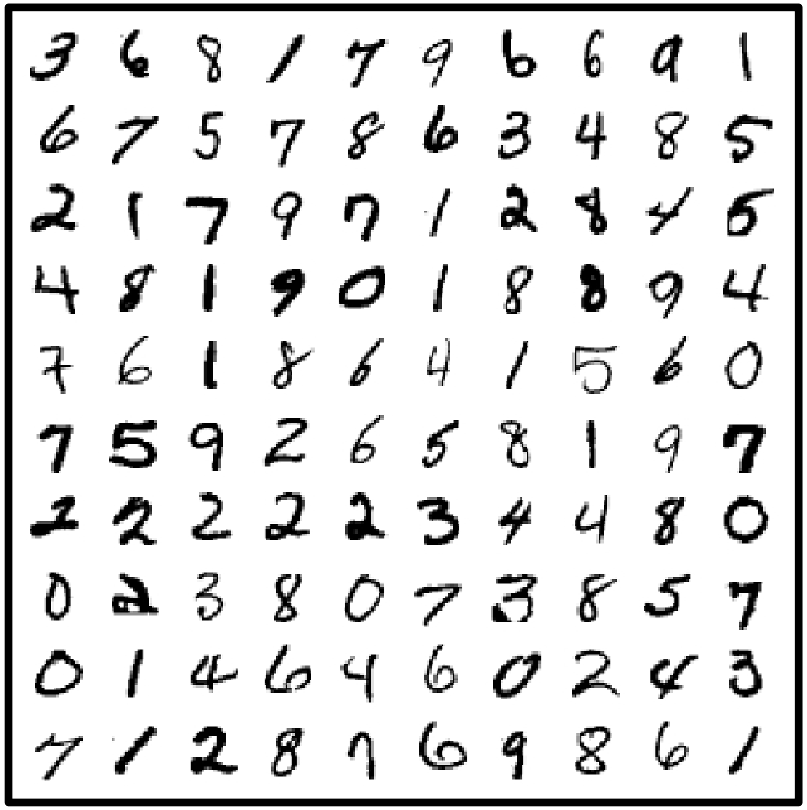
- Flows are container types
- Checkpoint functions
- Optional crash recovery
- Invert nodes and flows
- Lightweight graph module

New in version 2.0:

- Supervised nodes
- Multiple training phases
- Properties simplify subclassing
- Converted to the new numpy
- Several utilities
- Extended documentation

MDP: a Real Life Example

Handwritten digit recognition



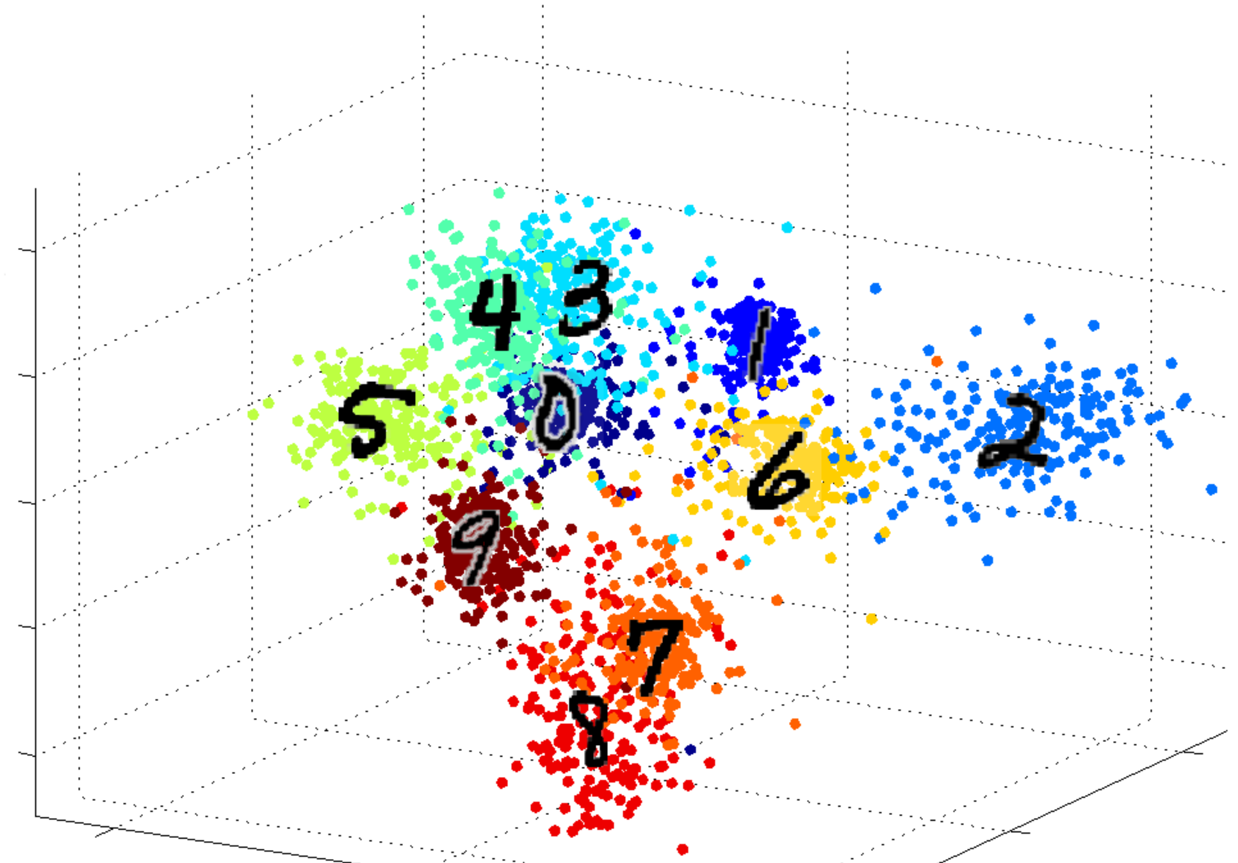
Pietro Berkes
Handwritten digit recognition with Nonlinear Fisher Discriminant Analysis
ICANN 2005

MDP: a Real Life Example

```
>>> flow = Flow([WhiteningNode(output_dim=35),
...                 PolynomialExpansionNode(3),
...                 FDANode(output_dim=9),
...                 GaussianClassifierNode()])
...
>>> class DataIterator(object):
...     def __init__(self, database, sup = False):
...         self.db = database
...         self.sup = sup
...     def __iter__(self):
...         for label, digits in progressinfo(self.db):
...             if self.sup: yield (label, digits)
...             else: yield digits
...
>>> flow.train([DataIterator(train_digits), ...])
[========================67%==========>...................]
>>> guess_labels = flow(DataIterator(test_digits))
89% [01:23:12]-[03:24:11]
>>> error_rate = check_error(guess_labels, known_labels)
>>> visualize_feature_space(DataIterator(test_digits))
```

MDP: a real life example

Feature Space

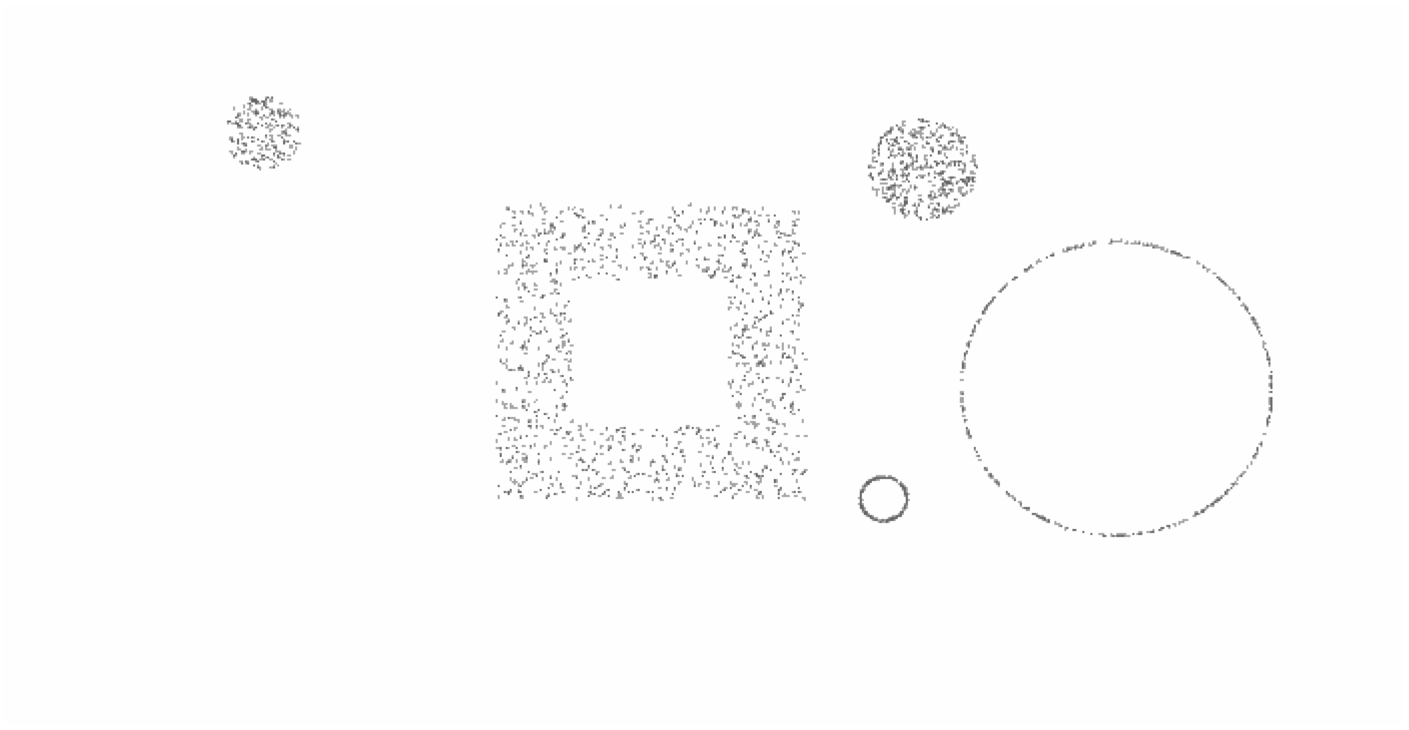


MDP: future perspectives

API and internal structure are now stable:

- **Extend algorithms library**
- **Acyclic graphs**
- **Support SciPy**
- **User feedback and contributions**

The End



http://mdp-toolkit.sourceforge.net