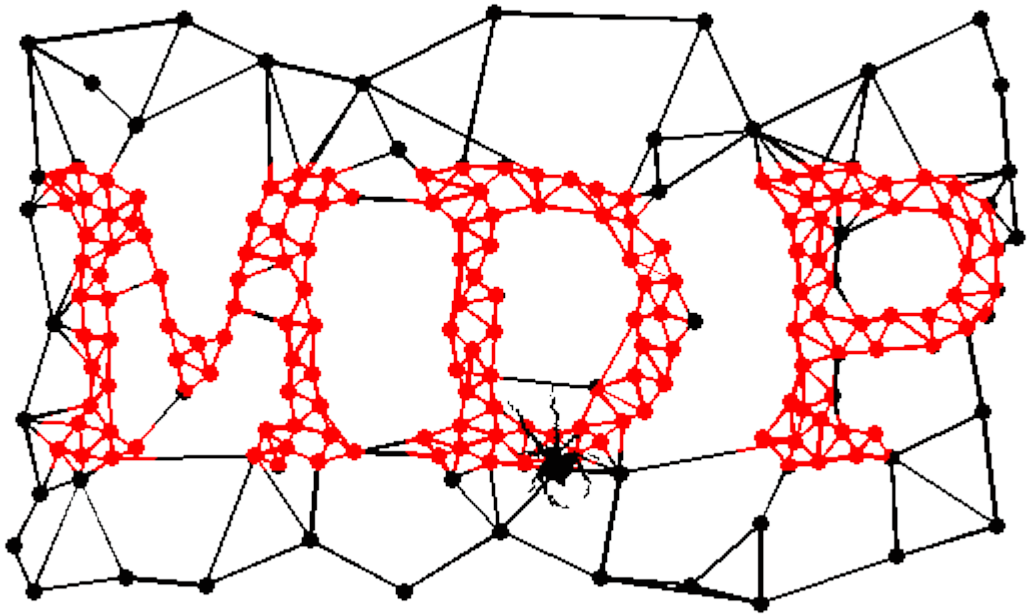
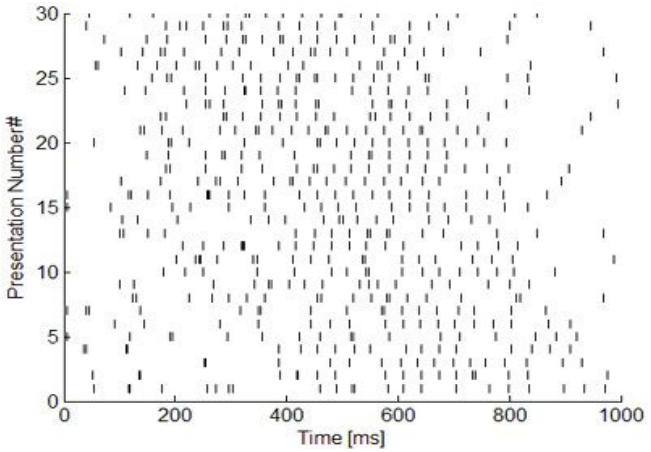
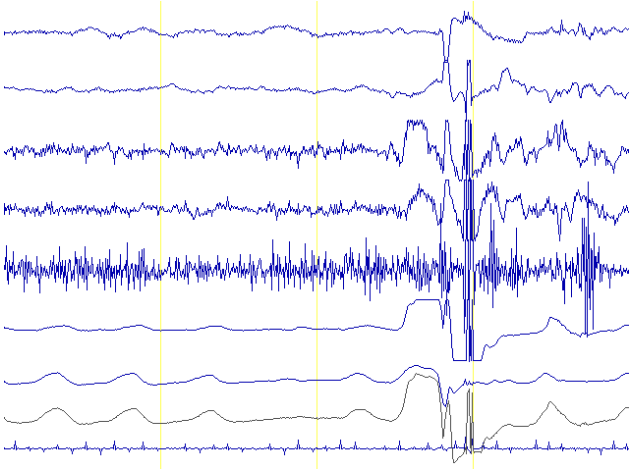
Modular Toolkit for Data Processing

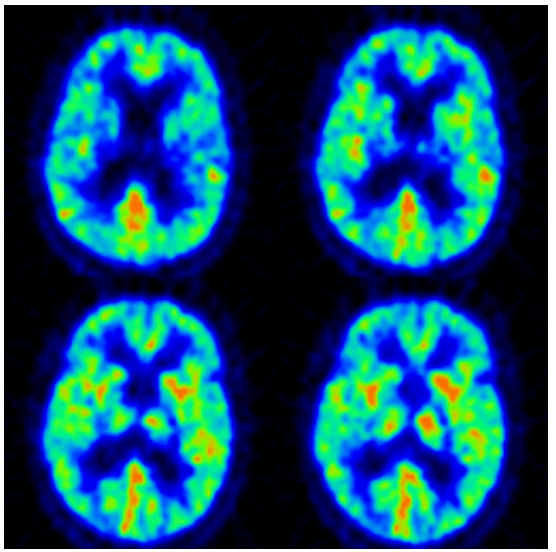


28.06.2005
Pietro Berkes & Tiziano Zito
Institute for Theoretical Biology
Berlin, Germany

Data Processing in Neuroscience







Wish List

- Collection of standard algorithms
- Efficient in RAM and CPU usage
- No limits for data set size
- No limits for data set dimensionality
- Combine many data processing units
- Easily extensible framework

Python versus Matlab®

- Python is good
- OO, reference calls
- OS: community assistance
- OS: reproducible results
- No numerical extension in stdlib

Python Numerical Extensions

- Numeric, Numarray, SciPy
- F2PY (used in symeig)
- Scientific visualization software?
- Small community, *lazy* developers

Modular toolkit for Data Processing

Python library to perform data processing:

- **Data processing units (nodes)**
- **Data processing flows**
- **Static typing design**
- **Many standard algorithms**
- **Easy to use and to extend**
- **Documentation and demos**

MDP Building Blocks: Node

Data processing unit:

- **Typecode**
- **Input and output dimensions**
- **Training (batch, online, block-mode)**

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

MDP Building Blocks: Node

Already implemented nodes:

- **Principal Component Analysis**
- **Independent Component Analysis**
- **Slow Feature Analysis**
- **Growing Neural Gas Network**
- **Polynomial Expansion**
- **Time Frames**
- **Hit Parades**
- **Noise**

To be added soon:

- **Fisher Discriminant Analysis**
- **Gaussian Classifiers**

MDP Building Blocks: Flow

Data processing sequence:

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

```
>>> flow = SimpleFlow([EtaComputerNode(),
...                    PCANode(output_dim=10),
...                    PolynomialExpansionNode(5),
...                    SFANode(),
...                    EtaComputerNode(),
...                    VisualizationNode()])
...
>>> def generator(seed, cycles):
...     set_random_seed(seed)
...     for i in range(cycles):
...           x = produce_stuff()
...           yield x
...
>>> flow.train([None, generator(seed, cycles), ...])
>>> out = flow.execute(data)
```

MDP: framework for developers

Write your own nodes:

- **Implement `train` and `execute`**
- **Integrated with existing library**

```
>>> class MyNode(SignalNode):
...       def train(self, x):
...           train_code()
...       def execute(self, x):
...           execute_code()
...       def get_supported_typecodes(self):
...           return ['f', 'd', 'i']
...
>>> flow = SimpleFlow([PCANode(), MyNode()])
>>> flow.train([generatorPCA(), generatorMyNode()])
>>> out = flow.execute(data)
```

MDP: additional features

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

MDP: a real life example

Handwritten digit recognition

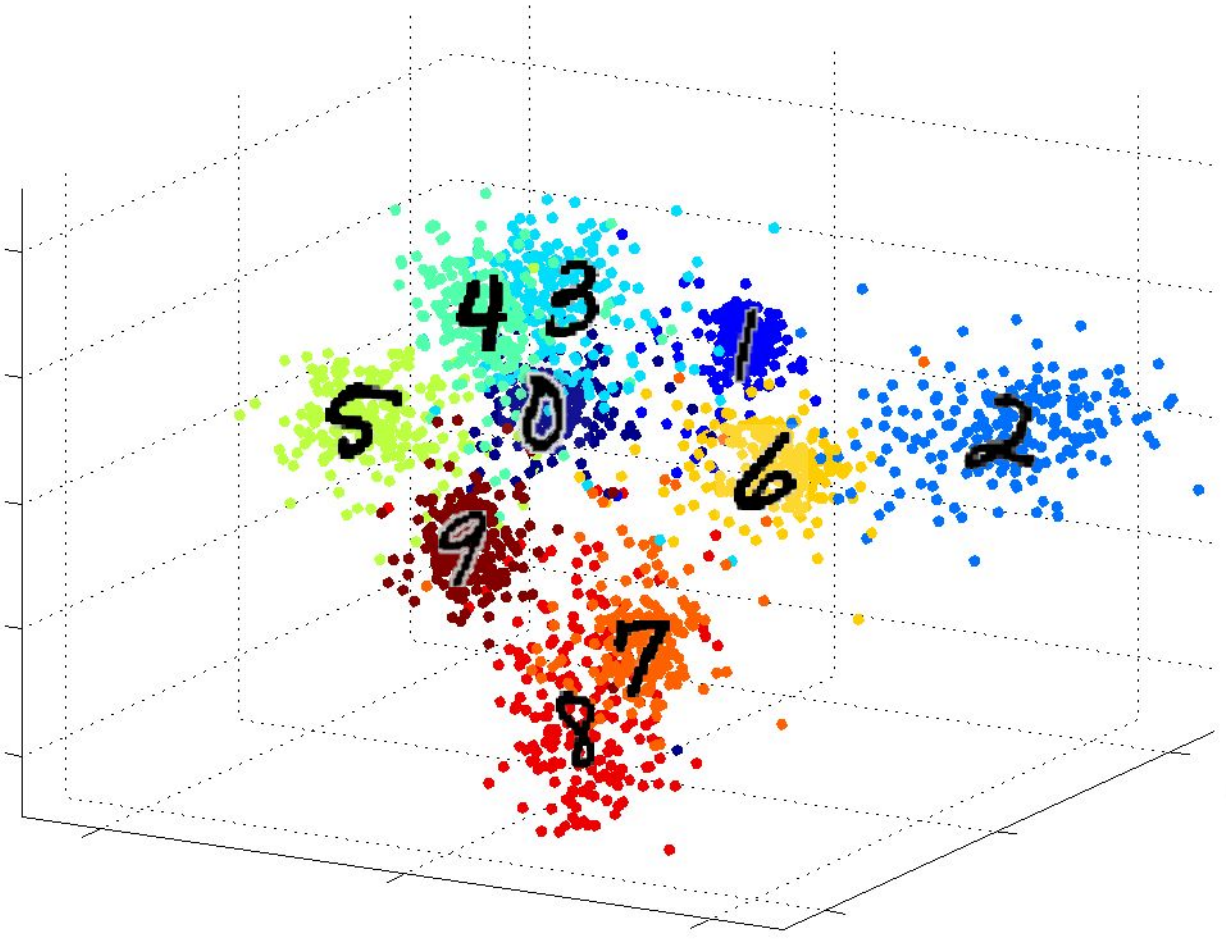
Pietro Berkes
Handwritten digit recognition with Nonlinear Fisher Discriminant Analysis
ICANN 2005

MDP: a real life example

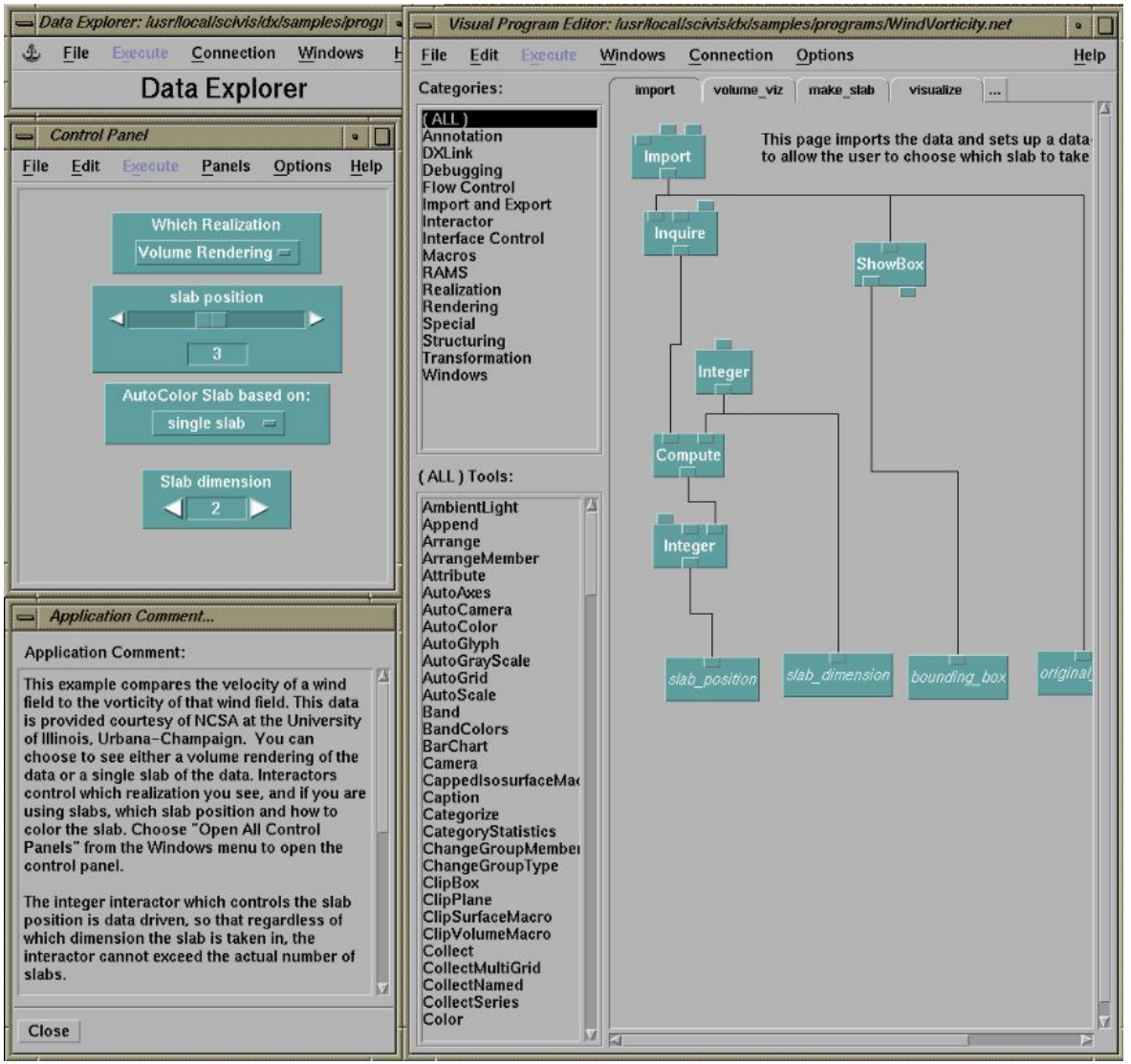
```
>>> flow = SimpleFlow([PCANode(output_dim=35),
...                    PolynomialExpansionNode(3),
...                    FDANode(output_dim=9),
...                    GaussClassifierNode()])
...
>>> def generator(database):
...     for label, digits in database.items():
...         yield digits
...
>>> flow.train([generator(train_digits), ...])
>>> guess_labels = flow.execute(generator(test_digits))
>>> error_rate = check_error(guess_labels, known_labels)
>>> visualize_feature_space(generator(test_digits))
```

MDP: a real life example

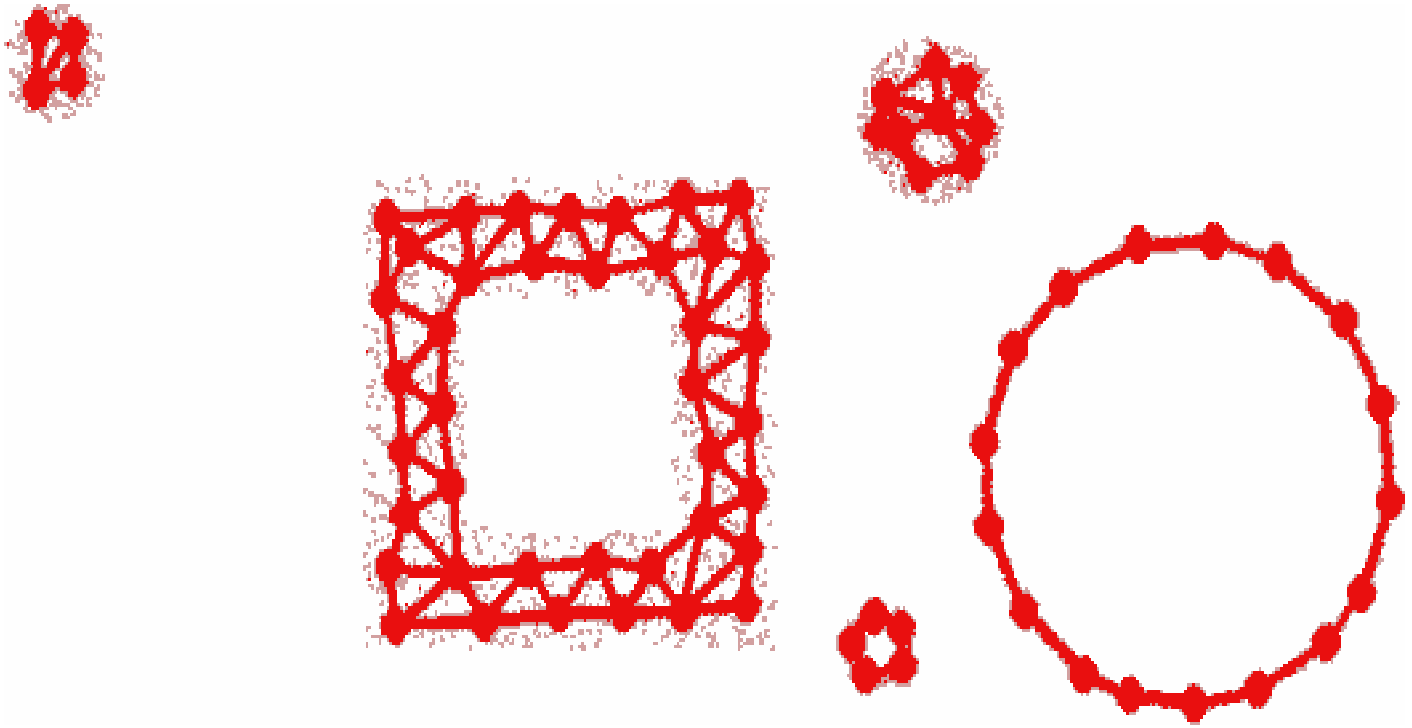
Feature Space



MDP: future perspectives



The End



http://mdp-toolkit.sourceforge.net