Factor Graph Toolbox

A quick overview of the most important features

Felix Vollmer

The IME Factor Graph Toolbox

Python implementation of a generic (Forney-style) factor graph toolbox.

Current features include:

- Multivariate, linear Gaussian Message Passing
- Sigma-point methods for treatment of nonlinear transformations
- Expectation Maximization (EM) for parameter estimation
- Kalman filtering and smoothing

Install The Factor Graph Toolbox

1. Download toolbox and extract it to some folder
2. Open or create a new project in PyCharm. (not a single file)
3. Open terminal (right click on project folder, Open in terminal or Alt + F12). This should handle virtual enviroments
4. Install factor graph package with

   ```
   pip install -e "path/to/toolbox"
   ```

   where "path/to/toolbox" folder contains the setup.py. This also installs any missing package.
5. Done. A restart of PyCharm may be useful to recognize the new packages

Quick Test Of The Installation

Create `hello_world.py` in current project
(right click on project folder, New > Python File)

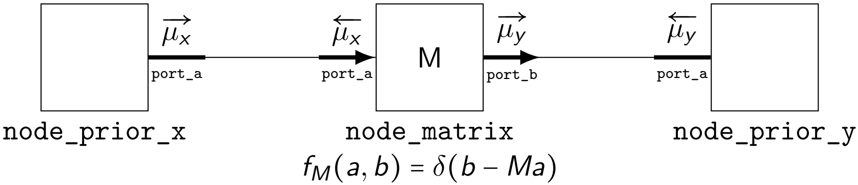
```
from ime_fgs.messages import GaussianMeanCovMessage

distribution = GaussianMeanCovMessage([[1]],[[0]])
print(distribution)
```

and run the file with with a right click on the file contents, Run 'hello_world' or Ctrl + Shift + F10. (should run without errors)

Pitfalls:

- PyCharm may need some time to index the new files. While indexing the auto completion, etc. is unavailable.
- Automatic handling of virtual environments might not work correctly with multiple projects. Either change the virtual environment by hand or only open a single project.

Essential Components of the factor graph toolbox



- Nodes (`node_prior_x`, `node_matrix`, `node_prior_y`)
- NodePorts (`node_prior_x.port_a`, `node_matrix.port_a`, ...)
  - Belong to a node
  - Connect with other ports to build a graph
- Messages ($\overleftarrow{\mu}_x$, $\overrightarrow{\mu}_x$, $\overleftarrow{\mu}_y$, $\overrightarrow{\mu}_y$)
  - Belong to a port

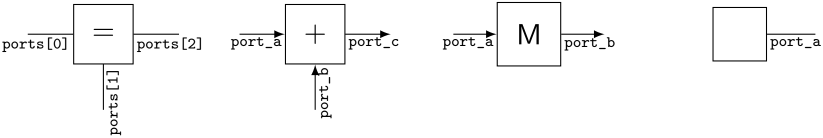
    `node_prior_x.port_a.out_msg` $= \overrightarrow{\mu}_x$

    `node_matrix.port_a.out_msg` $= \overleftarrow{\mu}_x$

    ...

Nodes

`basic_nodes.py`: most common nodes, including



`EqualityNode`

$f_=(x_0, x_1, x_2)$

$=\delta(x_0 - x_1)\delta(x_0 - x_2)$

`AdditionNode`

$f_+(a, b, c)$

$=\delta(a + b - c)$

`MatrixNode`

$f_M(a, b)$

$=\delta(b - Ma)$

`PriorNode`

$\mathcal{N}(\mathbf{m}, \mathbf{V})$

Messages

`messages.py`: Contains different messages including

- Multivariate gaussian normal distribution in different parameterizations
  - Mean $\mathbf{m}$, variance $\mathbf{V}$: `GaussianMeanCovarianceMessage`
  - Weighted mean ($\mathbf{Wm} = \xi$), information matrix $\mathbf{W} = \mathbf{V}^{-1}$: `GaussianWeightedMeanInformationMessage`

Attention

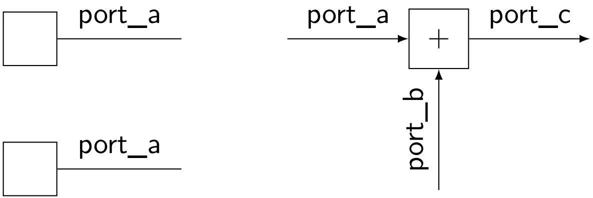
Not all operations are available for all parameterizations.
(call `message.convert(…)` for manual conversion)

Toolbox Workflow

1. Instantiate all nodes
2. Create graph by connecting ports with `port.connect(...)`
3. Provide prior messages
4. Propagate messages through graph by calling `port.update()` and passing an optional type argument to convert messages

A Simple Usage Example

Create nodes

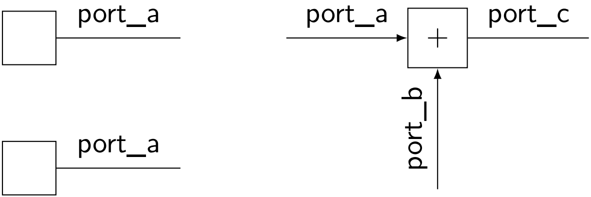


Code

```
addition_node = AdditionNode()
prior_node_1 = PriorNode()
prior_node_2 = PriorNode()
```

A Simple Usage Example

Connect nodes by connecting ports



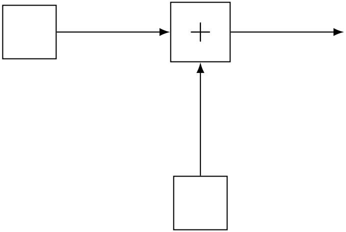
Code

```
prior_node_1.port_a.connect(addition_node.port_a)
prior_node_2.port_a.connect(addition_node.port_b)
```

Call `connect()` once for every edge

A Simple Usage Example

Connect nodes by connecting ports



Code

```
prior_node_1.port_a.connect(addition_node.port_a)
prior_node_2.port_a.connect(addition_node.port_b)
```

Call `connect()` once for every edge

A Simple Usage Example

Create messages

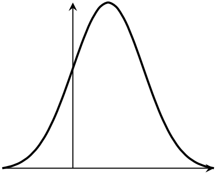


Figure: Distribution d1

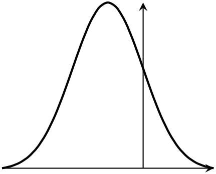


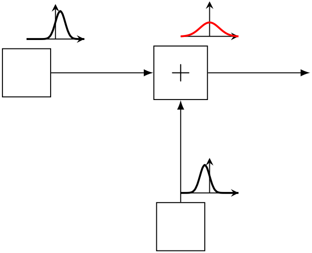
Figure: Distribution d2

Code

```
d1 = GaussianMeanCovMessage([[1]],[[ 1]])
d2 = GaussianMeanCovMessage([[-1]], [[1]])
```

A Simple Usage Example

Propagate messages by calling `.update()` on ports

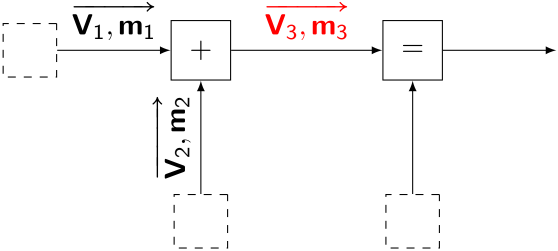


Code

```
prior_node_1.update_prior(d1)
prior_node_2.update_prior(d2)

addition_node.port_c.update()
print(addition_node.port_c.out_msg)
```

Message Conversion Example

There is no automatic conversion


$\overrightarrow{\mathbf{V}_1, \mathbf{m}_1}$
+
$\overrightarrow{\mathbf{V}_3, \mathbf{m}_3}$
=
$\overrightarrow{\mathbf{V}_2, \mathbf{m}_2}$

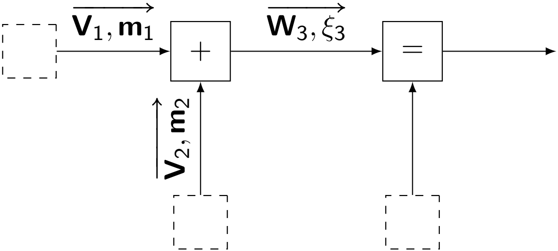
Factor graph types

- Input message types determine the calculation rules
- No implicit conversion

⇒ Convert `out_msg` by passing type argument to `.update()`

Message Conversion Example

There is no automatic conversion



$\overrightarrow{\mathbf{V}_1, \mathbf{m}_1}$
+
$\overrightarrow{\mathbf{W}_3, \xi_3}$
=
$\overrightarrow{\mathbf{V}_2, \mathbf{m}_2}$

Code: Propagate message with type conversion

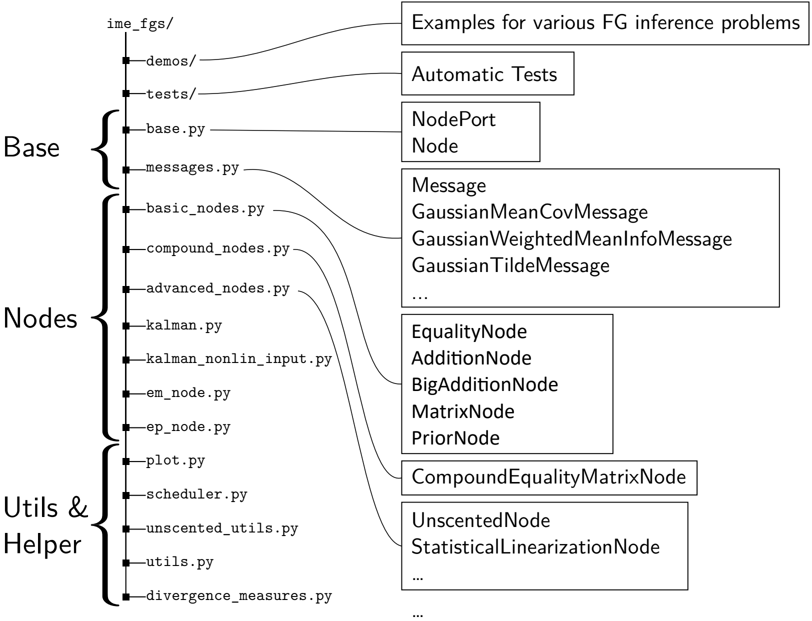
```
addition_node.port_c.update(  ↙
   GaussianWeightedMeanInformationMessage)
```

Complete example code

Code

```python
from ime_fgs.basic_nodes import AdditionNode, PriorNode
from ime_fgs.messages import GaussianMeanCovMessage

# create nodes
addition_node = AdditionNode()
prior_node_1 = PriorNode()
prior_node_2 = PriorNode()

# connect nodes
prior_node_1.port_a.connect(addition_node.port_a)
prior_node_2.port_a.connect(addition_node.port_b)

# create messages
d1 = GaussianMeanCovMessage([[1]], [[1]])
d2 = GaussianMeanCovMessage([[-1]], [[1]])

# pass messages through graph
prior_node_1.update_prior(d1)
prior_node_2.update_prior(d2)
addition_node.port_c.update()

# print result
print(addition_node.port_c.out_msg)
```

Overview of Files in Toolbox



```
ime_fgs/
├─demos/
├─tests/
├─base.py
├─messages.py
├─basic_nodes.py
├─compound_nodes.py
├─advanced_nodes.py
├─kalman.py
├─kalman_nonlin_input.py
├─em_node.py
├─ep_node.py
├─plot.py
├─scheduler.py
├─unscented_utils.py
├─utils.py
└─divergence_measures.py
```

Base

Nodes

Utils & Helper

Examples for various FG inference problems

Automatic Tests

NodePort
Node

Message
GaussianMeanCovMessage
GaussianWeightedMeanInfoMessage
GaussianTildeMessage
...

EqualityNode
AdditionNode
BigAdditionNode
MatrixNode
PriorNode

CompoundEqualityMatrixNode

UnscentedNode
StatisticalLinearizationNode
...

...