Manual

PaREnDataSources

Introduction

The PaREnDataSources module is a spin-off from the ExCoLib. Check out the full folder from the svn in order to get it all.

PaREnDataSources generates various artificial problems. The PaREn Translator class has been removed from the module. Therefore the functions return multiple lists and arrays instead. PaREnDataSources and the full PaREn ExCoLib should not be installed in parallel or there can exist conflicts, due to the same names.

PaREnDataSources

Installation

Download PaREnTranslatorPlugins (Separated) and invoke

```
python setup.py install
```

This will use distutils, in order to make PaREnDataSources globally visible. Through this global visibility you can always invoke:

import PaREnDataSources

from PaREnDataSources import *

General Description

This module contains a set of functions for generation of artificial data. The currently supplied list is by no means complete. More dataset generators will follow in the future.

Functions

def WhiteGaussian(**int** Count, **int** Dimensions)

The WhiteGaussian function returns a gaussian distribution consisting of >>Count<< vectors in >>Dimensions<< sitting at the zero coordinate. The result is a 2D Numpy Array where vectors are rows. The vectors must be transformed with some affinity transformation, in order to obtain densities with other covariance matrices or positions.

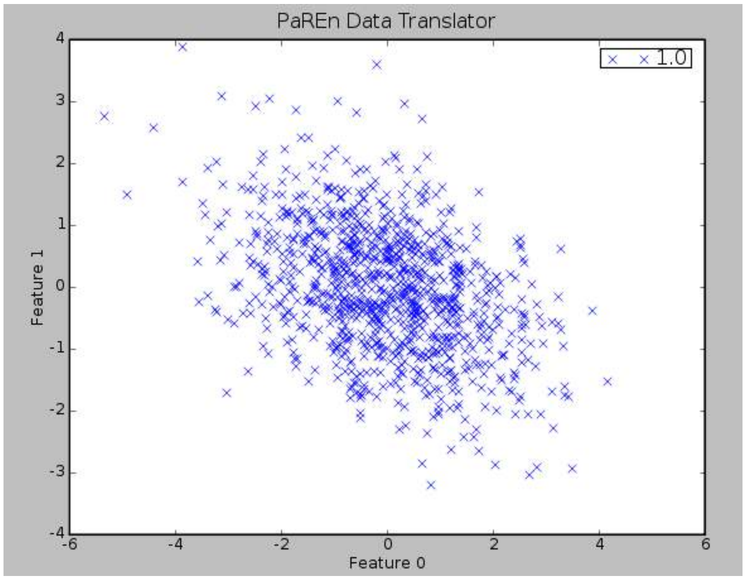
def GaussianDistribution(

int Count, **int** Dimensions,
ndarray(1xDimensions) mu,
ndarray(Dimensions x Dimensions) epsilon)

This function performs the right transformation on the WhiteGaussian, in order to obtain a density where cov() on the vectors will return >>epsilon<< and mean() will return >>mu<<.

Example:

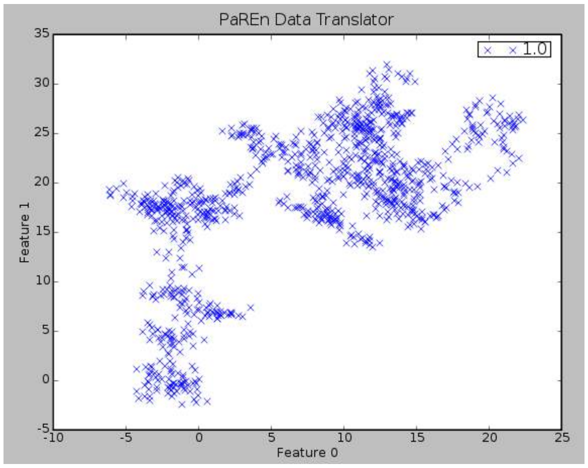
```
C = GaussianDistribution( 1000, 2, zeros(2), array( [ [ 2, 0.5], [0.5, 1.0] ] ) )
```



def ComplexDistribution(

int Count,
int Dimensions,
ndarray(1xDimensions) StartVec,
int Weirdness):

This functions generates strange distributions. The stronger is weirdness, the less gaussian the distribution becomes. For a value of Weirdness=0.0 the distribution is point like. For Weirdness > 0.0 and Weirdness<0.5 the distribution is roughly uniform. Weirdness factors around 1.0 to 3.0 generate gaussian densities. Weirdness > 3.0 will generate strangely shaped distributions like below. This function provides estimated probabilities using a special kernel method.



Weirdness 10, 1000 Samples

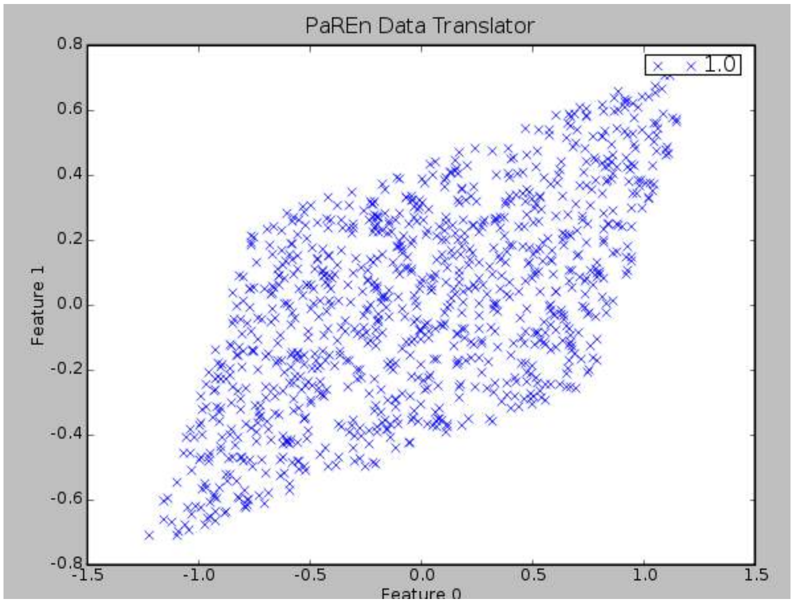
def IndependentDistribution(

int Count,
int Dimensions,
ndarray(1xDimensions) TranslationVector,
ndarray(Dimensions x Dimensions) TransformationMatrix)

This function creates uniformly distributed samples. The samples are generated within a -1/+1 range around >>TranslationVector<<. The form can be distorted via TransformationMatrix.

Example:

```
C[1] = IndependentDistribution( 1000, 2, zeros(2), array( [ [ 2, 0.5], [0.5, 1.0] ] ) )
```



---

[1] *C is a tuple of (Vector Matrix, Label List, Probabilities Matrix, Header List)*

def **Mixture**(**[[]]** Descriptions, **int** Dims, **int** Normalize=1)

The Mixture function creates more complicated datasets formed of gaussian, complex and independent distributions and allows to specify multiple densities to one class label.

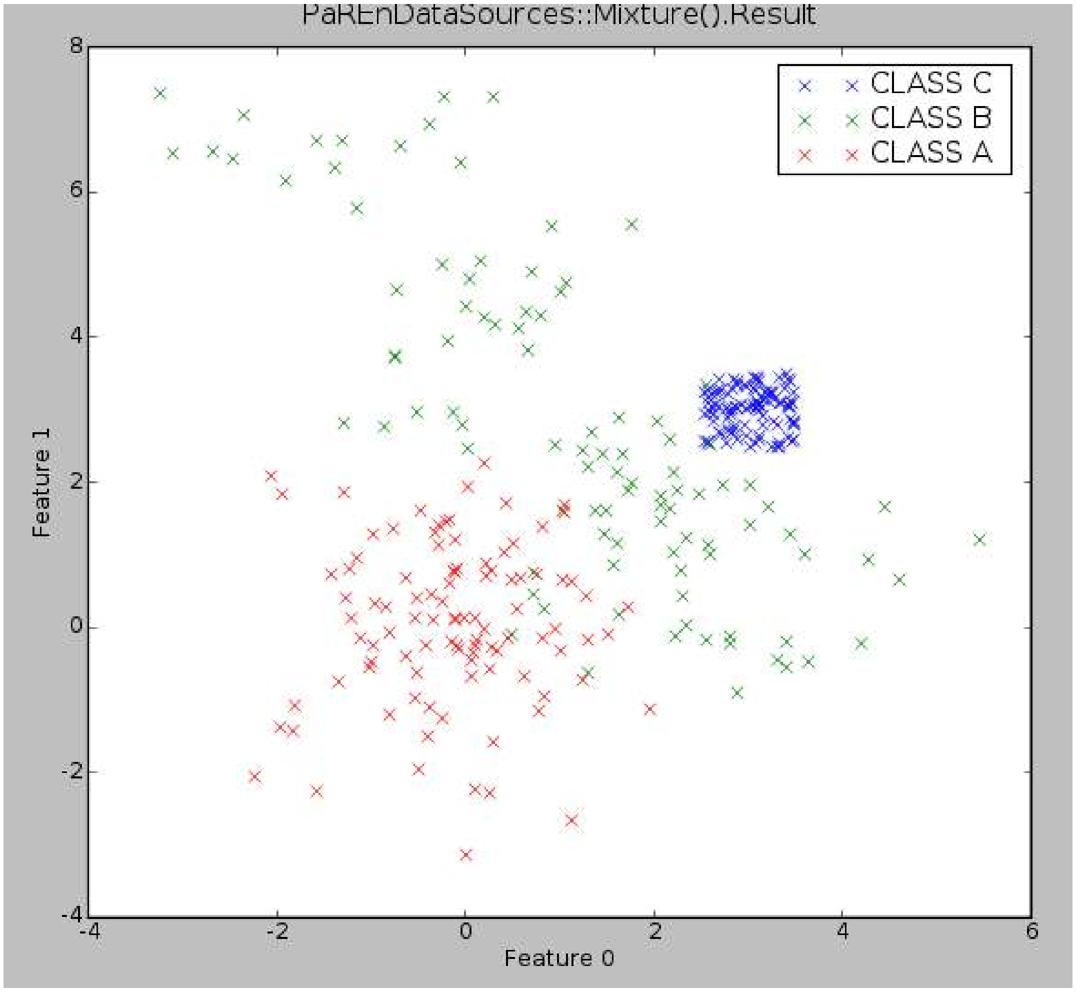
The >>Descriptions<< parameter is a list of lists. Each list (E) contains a set of parameters that describes a density and which label this density belongs to. E is built up like the following:

E = [LABEL, DENSITY_TYPE, SAMPLE_COUNT, VECTOR, OPTIONAL_PARAMETER]

D = [E_1, E_2, E_3, ... E_n]

Example:

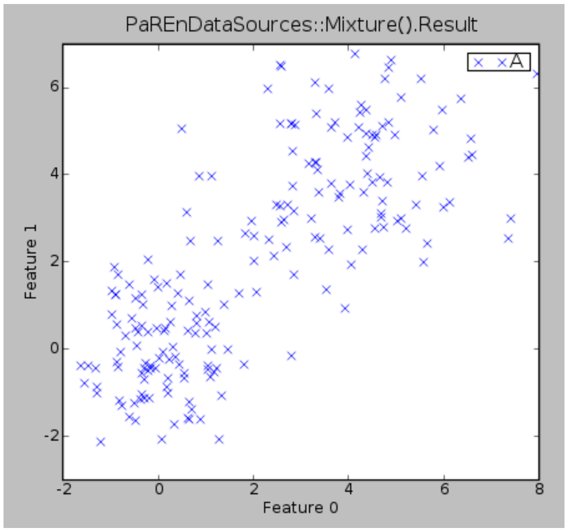
```
Description = [ [ “Class A”, “Gaussian Distribution”, 100, array( [ [ 0 ], [ 0 ] ] ), eye( 2 ) ],
                [ “Class B”, “Weird Distribution”, 100, array( [ [ 10 ], [ -3 ] ] ), 5 ],
                [ “Class C”, “Independen Distribution”, 100, array( [ [ 3 ], [ 3 ] ] ), eye( 2 ) ] ];
C[2] = Mixture( Description, 2 )
```

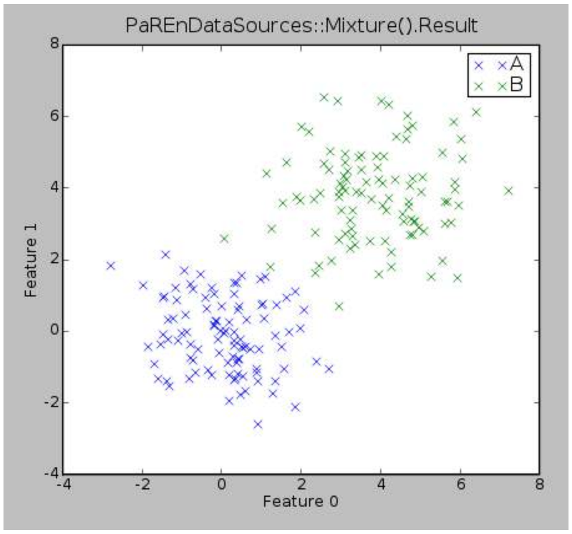


---

[2] *C is a tuple of (Vector Matrix, Label List, Probabilities Matrix, Header List)*

Every entry can belong to a different label, but they also can have all one label:





Left: `C = Mixture( [["A", "gauss", 100, zeros(2), eye(2)], ["A", "gauss", 100, 4*ones(2), 2*eye(2)]], 2 )`

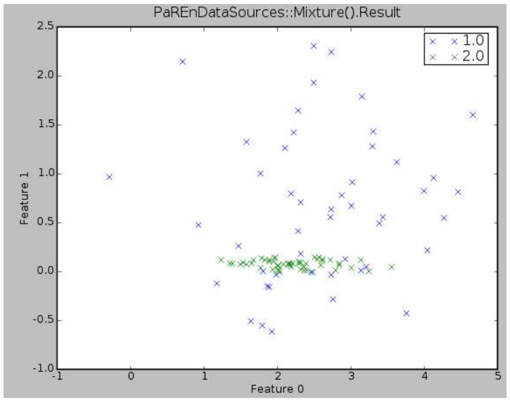
Right: `C = Mixture( [["A", "gauss", 100, zeros(2), eye(2)], ["B", "gauss", 100, 4*ones(2), 2*eye(2)]], 2 )`

```
def QuicklyGenerateProblem(
        int Classes=2,
        int Dimensions=2,
        int Samples=100,
        float Easy=3,
        float(0.0     1.0)Spacing=0.5 )
```

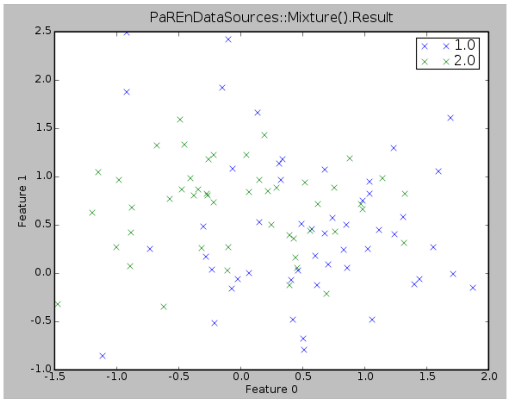
The QuicklyGenerateProblem function creates ad hoc problems of specified kind for your algorithm. You can undespecify your wishes, as there are many defaults.

Example:

`C[3] = QuicklyGenerateProblem()`



Try 1



Try 2

---

[3] *C is a tuple of (Vector Matrix, Label List, Probabilities Matrix, Header List)*

def ErrorProblem2SphericGaussians([4]

float Error,
int Dimensions = 2,
int Samples = 100,
float Tolerance = 0.0001)

This function generates a two class two Gaussian densities dataset, where the covariance is the identity matrix for each and the Bayes error is as specified by >>Errror<<. The algorithm is described in ... svn ... link. ...

def ErrorProblem2Gaussians([5]

float Error,
ndarray(Dimensions x Dimensions) Cov1,
ndarray(Dimensions x Dimensions) Cov2,
int Dimensions = 2,
int Samples = 100,
float Tolerance = 0.0001)

This function generates a two class two Gaussian densities dataset, where the covariance is user specified for each and the Bayes error is as specified by >>Errror<<. The matrices >>Cov1<< and >>Cov2<< must match >>Dimensions<<. The algorithm is described in ... svn ... link. ...

def CreateRawEncoderProblem(

int Dims = 2,
int Samples = 100,
str Mode = "input", (or in , out , in/out)
float Noise = 0.0,
int NoiseUnits = 0)

This function generates a raw encoder problem. It returns two Numpy Arrays:

Example:

```
array1, array2 = CreateRawEncoderProblem( 5, 1000, “input and output”, 0.1, 1 )
```

Mode specifies where noise parameters are applied. If mode is input then only array1 will be affected by noise parameters. If mode is output then only array2 will be affected by noise parameters. If mode is in/out then both arrays will be parametrized by the noise parameters.

---

[4] *retruns is a tuple of (Vector Matrix, Label List, Probabilities Matrix, Header List)*

[5] *retruns is a tuple of (Vector Matrix, Label List, Probabilities Matrix, Header List)*

>>Noise<< is a floating point value, that controls how much random value is added to the encoder dataset.

>>NoiseUnits<< controls how many noise-only nodes are added.

Examples:

```
A,B = CreateRawEncoderProblem( 3, 10 )
A:            B:
+---+---+---+ +---+---+---+
|1.0|0.0|0.0| |1.0|0.0|0.0|
|0.0|1.0|0.0| |0.0|1.0|0.0|
|0.0|0.0|1.0| |0.0|0.0|1.0|
|1.0|0.0|0.0| |1.0|0.0|0.0|
|0.0|1.0|0.0| |0.0|1.0|0.0|
|0.0|0.0|1.0| |0.0|0.0|1.0|
|1.0|0.0|0.0| |1.0|0.0|0.0|
|0.0|1.0|0.0| |0.0|1.0|0.0|
|0.0|0.0|1.0| |0.0|0.0|1.0|
|1.0|0.0|0.0| |1.0|0.0|0.0|
+---+---+---+ +---+---+---+
```

```
A,B = CreateRawEncoderProblem( 3, 10, Mode=”in”, Noise=0.5 )
A:                                                B:
+--------------+---------------+---------------+  +---+---+---+
|1.34489508978 |0.0655556880806| 0.45494119438 |  |1.0|0.0|0.0|
|0.319214803314| 1.03947855831 |0.0839070174238|  |0.0|1.0|0.0|
|0.129328179365|0.0648677289401| 1.20803935353 |  |0.0|0.0|1.0|
|1.03264143184 | 0.212919683338| 0.46376974767 |  |1.0|0.0|0.0|
|0.349685986479| 1.13581725967 | 0.145952643732|  |0.0|1.0|0.0|
|0.496091617656| 0.365446700636| 1.40265205017 |  |0.0|0.0|1.0|
| 1.3348465234 | 0.266344493772| 0.390886226823|  |1.0|0.0|0.0|
|0.301720089697| 1.47063020742 | 0.381966592744|  |0.0|1.0|0.0|
|0.174359293167| 0.471520613969| 1.05809272028 |  |0.0|0.0|1.0|
|1.38831139689 | 0.198323500777| 0.329593570954|  |1.0|0.0|0.0|
+--------------+---------------+---------------+  +---+---+---+
A,B = CreateRawEncoderProblem( 3, 10, Mode=”out”, Noise=0.5 )
A:            B:
+---+---+---+ +--------------+--------------+--------------+
|1.0|0.0|0.0| |1.34489508978 |0.0655556880806| 0.45494119438 |
|0.0|1.0|0.0| |0.319214803314| 1.03947855831 |0.0839070174238|
|0.0|0.0|1.0| |0.129328179365|0.0648677289401| 1.20803935353 |
|1.0|0.0|0.0| |1.03264143184 | 0.212919683338| 0.46376974767 |
|0.0|1.0|0.0| |0.349685986479| 1.13581725967 | 0.145952643732|
|0.0|0.0|1.0| |0.496091617656| 0.365446700636| 1.40265205017 |
|1.0|0.0|0.0| | 1.3348465234 | 0.266344493772| 0.390886226823|
|0.0|1.0|0.0| |0.301720089697| 1.47063020742 | 0.381966592744|
|0.0|0.0|1.0| |0.174359293167| 0.471520613969| 1.05809272028 |
|1.0|0.0|0.0| |1.38831139689 | 0.198323500777| 0.329593570954|
+---+---+---+ +--------------+--------------+--------------+
```

```
A,B = CreateRawEncoderProblem( 3, 10, Mode=”out”, Noise=0.1, NoiseUnits=2 )
A:
+---+---+---+
|1.0|0.0|0.0|
|0.0|1.0|0.0|
|0.0|0.0|1.0|
|1.0|0.0|0.0|
|0.0|1.0|0.0|
|0.0|0.0|1.0|
|1.0|0.0|0.0|
|0.0|1.0|0.0|
|0.0|0.0|1.0|
|1.0|0.0|0.0|
+---+---+---+
B:
+----------------+---------------+----------------+---------------+---------------+
| 1.04058994783  | 0.077770737664|0.0845909717234 |0.0465844219446|0.0792101958286|
|0.0215343157484 | 1.08871112068 |0.0139484807802 |0.0950639989274|0.0280484573487|
| 0.094413327519 |0.0387981333766| 1.08714990891  |0.0680018475687|0.0379175410037|
| 1.03520451117  |0.0418488654474| 0.060844080744 |0.0470502237221|0.0541516866494|
|0.0997343692416 | 1.04073489485 |0.00379138353052|0.0818611447894|0.0545989602734|
|0.00140694623189|0.0224240200133| 1.04167675775  |0.0269606400482|0.0453919936546|
| 1.09438408905  |0.0682346795393|0.0589501384258 |0.0451808449257|0.0384955144281|
|0.0897448336309 | 1.07636400688 |0.0176064123395 |0.0513936367328|0.0494465373316|
|0.0229403677908 |0.0199076531124| 1.06947235653  |0.0887747111166| 0.089102010419|
| 1.05661448932  |0.0155217078461|0.0392747754913 |0.0951182738576|0.0802722964425|
+----------------+---------------+----------------+---------------+---------------+
```