easydata

Unknown Author

January 5, 2014

In [56]:
```
%pylab inline
import sys
import glob
import os
import random

import numpy as np
import numpy.linalg as linalg
import numpy.random as rnd
from mpl_toolkits.mplot3d.axes3d import Axes3D
```

```
Populating the interactive namespace from numpy and matplotlib

WARNING: pylab import has clobbered these variables: ['f', 'random']
`%pylab --no-import-all` prevents importing * from pylab and numpy
```

Part I

Easydata GPLVM tests

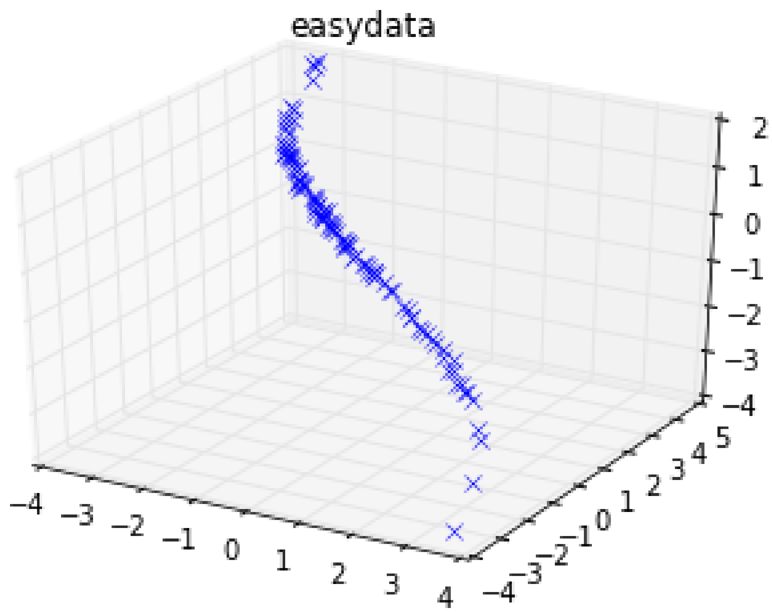
1 Generating the data

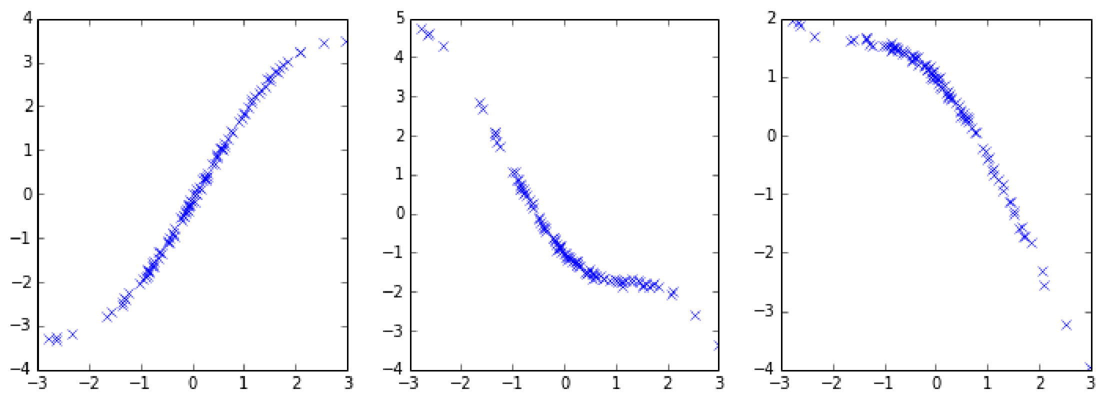
In [57]:
```
sys.path.append('../tools/')
sys.path.append('..')
import easy_dataset

Y, Xt = easy_dataset.gen_easydata(100, 1, 3)
```

After generating the data, plot in 3D and then each dimension as a function of the latent variable X:

In [58]:
```
fig = plt.figure()
ax = fig.gca(projection='3d')
ax.plot(Y[:, 0], Y[:, 1], Y[:, 2], 'x')
#ax.view_init(elev=60, azim=300)
ax.set_title('easydata')

fig, ax = plt.subplots(1, 3, figsize=(12, 4), dpi=180)
ax[0].plot(Xt, Y[:, 0], 'x')
ax[1].plot(Xt, Y[:, 1], 'x')
ax[2].plot(Xt, Y[:, 2], 'x')
```

Out [58]: [<matplotlib.lines.Line2D at 0x8709c10>]





2 Initialisation with PCA

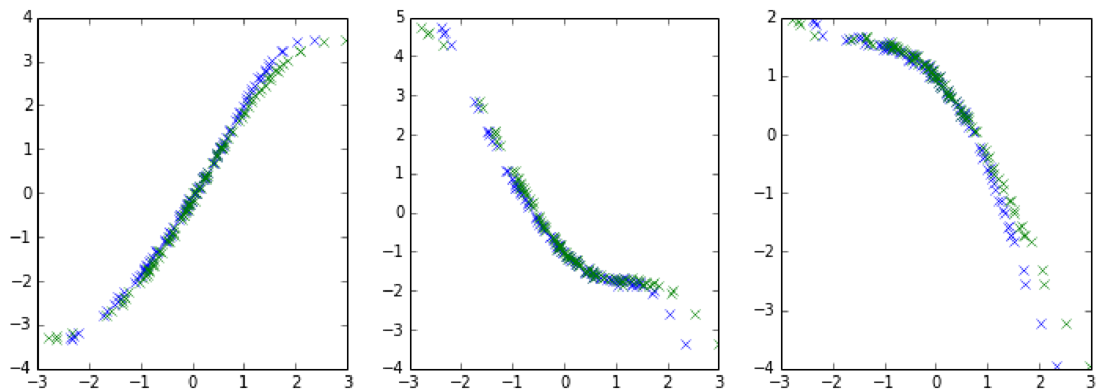
In [59]:
```
def PCA(Y, input_dim):
Z = numpy.linalg.svd(Y - Y.mean(axis=0), full_matrices=False)
[X, W] = [Z[0][:, 0:input_dim], numpy.dot(numpy.diag(Z[1]), Z[2]).T[:, 0:
v = X.std(axis=0)
X /= v;
W *= v;
return X, W

X, W = PCA(Y, 2)
```

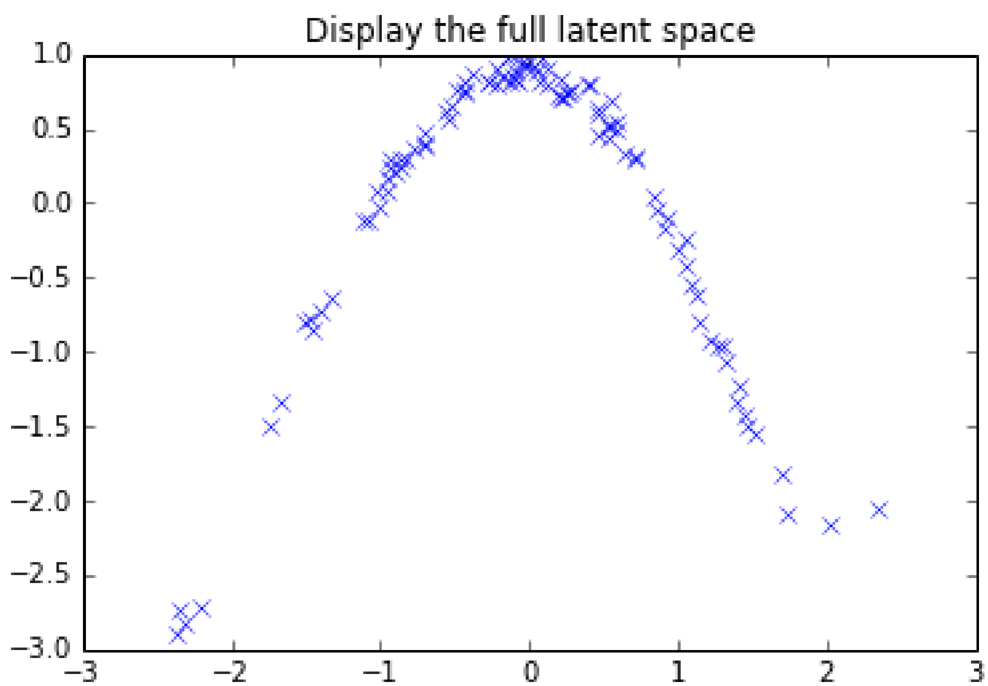
The main principle component of PCA manages to recover the correct latent coordinates (with perhaps a horizontal flip). In any case, the ordering is correctly found, which should give a very good initialisation for the GPLVM.

In [60]:
```
print('Display the first latent coordinate vs. the observed coordinates.')
fig, ax = plt.subplots(1, 3, figsize=(12, 4), dpi=180)
ax[0].plot(X[:, 0], Y[:, 0], 'x')
ax[0].plot(Xt, Y[:, 0], 'x')
ax[1].plot(X[:, 0], Y[:, 1], 'x')
ax[1].plot(Xt, Y[:, 1], 'x')
ax[2].plot(X[:, 0], Y[:, 2], 'x')
ax[2].plot(Xt, Y[:, 2], 'x')

plt.figure()
plt.plot(X[:, 0], X[:, 1], 'x')
plt.title('Display the full latent space')
```

```
Display the first latent coordinate vs. the observed coordinates.
```

Out [60]:
```
<matplotlib.text.Text at 0x8209b50>
```





We can also find the marginal likelihood of the model:

In [61]:
```
import MLtools
# Requires optimisation over beta, so maybe later.
```

3 GPy GPLVM results

We now run the GPy Bayesian GPLVM to see if sensible results are obtained. Sheffield seem to fix the noise while performing optimisation over (presumably) X, ard, Z etc.

In [62]:
```
# Parameters to adjust
Q = 2
num_inducing = 10
```

In [63]:
```
import GPy
np.random.seed(0)

# Normalise data
Yn = Y - Y.mean(0)
Yn /= Yn.std(0)

# Set up model
rbf_comp = GPy.kern.rbf(Q, ARD=True)
kern = rbf_comp + GPy.kern.bias(Q, np.exp(-2)) + GPy.kern.white(Q, np.exp
m = GPy.models.BayesianGPLVM(Y, Q, kernel=kern, num_inducing=10)
m['.*lengt'] = 1.                # ???
m['noise'] = Yn.var() / 100
m.ensure_default_constraints()

m.constrain_fixed('noise')
m.optimize('scg', messages=1, max_f_eval=100, gtol=.05)
print m.log_likelihood()
m.constrain_positive('noise')
m.optimize('scg', messages=1, max_f_eval=50, gtol=.05)
print m.log_likelihood()
```

```
Y is not zero mean, centering it locally (GPy.util.linalg.PCA)
Warning: re-constraining these parameters
noise_variance
 I      F              Scale          |g|
 0001   1.174933e+04   1.000000e+00   1.298781e+08   0002
8.510244e+03   5.000000e-01   9.630844e+06   0003   5.237181e+03
2.500000e-01   1.953195e+06   0004   2.763031e+03   1.250000e-01
4.560727e+05   0005   2.231165e+03   6.250000e-02   9.662234e+04   0006
1.877544e+03   3.125000e-02   6.718029e+04   0007   1.691982e+03
1.562500e-02   4.705385e+04   0008   1.189139e+03   7.812500e-03
2.267972e+04   0009   1.090526e+03   3.906250e-03   5.912366e+04   0010
1.019498e+03   1.953125e-03   9.675996e+03   0011   8.173122e+02
9.765625e-04   4.087968e+03   0012   7.818935e+02   4.882812e-04
6.351934e+03   0013   7.049787e+02   2.441406e-04   1.354300e+03   0014
6.720841e+02   1.220703e-04   2.297052e+03   0015   6.520418e+02
6.103516e-05   1.029606e+03   0016   6.346825e+02   3.051758e-05
1.542317e+03   0017   6.205158e+02   1.525879e-05   8.401982e+02   0018
6.076508e+02   7.629395e-06   1.301209e+03   0019   5.957174e+02
3.814697e-06   6.935461e+02   0020   5.842914e+02   1.907349e-06
1.280522e+03   0021   5.735990e+02   9.536743e-07   5.925474e+02   0022
5.631451e+02   4.768372e-07   1.250134e+03   0023   5.535387e+02
2.384186e-07   5.209937e+02   0024   5.441374e+02   1.192093e-07
```

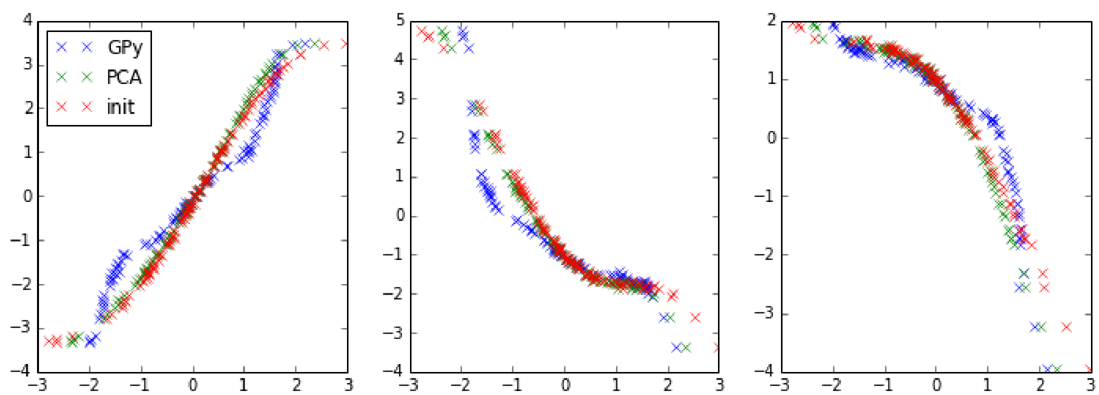
```
1.167765e+03  0025   5.355547e+02   5.960464e-08   4.693805e+02  0026
5.271719e+02   2.980232e-08   1.067871e+03  0027   5.193209e+02
1.490116e-08   4.217984e+02  0028   5.116299e+02   7.450581e-09
1.020822e+03  0029   5.044355e+02   3.725290e-09   3.826234e+02  0030
4.973630e+02   1.862645e-09   9.761591e+02  0031   4.908192e+02
9.313226e-10   3.516363e+02  0032   4.843352e+02   4.656613e-10
9.260817e+02  0033   4.784200e+02   2.328306e-10   3.269997e+02  0034
4.725837e+02   1.164153e-10   8.512031e+02  0035   4.673215e+02
5.820766e-11   3.099890e+02  0036   4.621396e+02   2.910383e-11
7.615319e+02  0037   4.574083e+02   1.455192e-11   2.948496e+02  0038
4.527497e+02   7.275958e-12   6.912431e+02  0039   4.484305e+02
3.637979e-12   2.795322e+02  0040   4.441511e+02   1.818989e-12
6.482094e+02  0041   4.402076e+02   9.094947e-13   2.666168e+02  0042
4.363096e+02   4.547474e-13   5.971419e+02  0043   4.327213e+02
2.273737e-13   2.567749e+02  0044   4.291861e+02   1.136868e-13
5.432539e+02  0045   4.259268e+02   5.684342e-14   2.498691e+02  0046
4.227272e+02   2.842171e-14   4.886610e+02  0047   4.197307e+02
1.421085e-14   2.423644e+02  0048   4.167954e+02   7.105427e-15
4.482699e+02  0049   4.140479e+02   3.552714e-15   2.378178e+02  0050
4.113469e+02   1.776357e-15   4.110117e+02  0051   4.088007e+02
8.881784e-16   2.321850e+02  0052   4.063028e+02   4.440892e-16
3.789586e+02  0053   4.039352e+02   2.220446e-16   2.269751e+02  0054
4.016158e+02   1.110223e-16   3.517152e+02  0055   3.994100e+02
5.551115e-17   2.221002e+02  0056   3.972495e+02   2.775558e-17
3.296582e+02  0057   3.951882e+02   1.387779e-17   2.158140e+02  0058
3.931732e+02   6.938894e-18   3.132113e+02  0059   3.912533e+02
3.469447e-18   2.099576e+02  0060   3.893824e+02   1.734723e-18
2.976827e+02  0061   3.875997e+02   8.673617e-19   2.048292e+02  0062
3.858605e+02   4.336809e-19   2.852357e+02  0063   3.841947e+02
2.168404e-19   1.982345e+02  0064   3.825630e+02   1.084202e-19
2.774875e+02  0065   3.809901e+02   5.421011e-20   1.905933e+02  0066
3.794469e+02   2.710505e-20   2.701763e+02  0067   3.779492e+02
1.355253e-20   1.829693e+02  0068   3.764740e+02   6.776264e-21
2.665296e+02  0069   3.750419e+02   3.388132e-21   1.762272e+02  0070
3.736287e+02   1.694066e-21   2.614184e+02  0071   3.722527e+02
8.470329e-22   1.692279e+02  0072   3.708918e+02   4.235165e-22
2.584774e+02  0073   3.695625e+02   2.117582e-22   1.615718e+02  0074
3.682476e+02   1.058791e-22   2.568330e+02  0075   3.669639e+02
5.293956e-23   1.549819e+02  0076   3.656925e+02   2.646978e-23
2.548263e+02  0077   3.644507e+02   1.323489e-23   1.486778e+02  0078
3.632199e+02   6.617445e-24   2.529492e+02  0079   3.620155e+02
3.308722e-24   1.424502e+02  0080   3.608177e+02   1.654361e-24
2.536346e+02  0081   3.596465e+02   8.271806e-25   1.363869e+02  0082
3.584832e+02   4.135903e-25   2.525861e+02  0083   3.573450e+02
2.067952e-25   1.309668e+02  0084   3.562147e+02   1.033976e-25
2.513823e+02  0085   3.551070e+02   5.169879e-26   1.258113e+02  0086
3.540057e+02   2.584939e-26   2.511799e+02  0087   3.529255e+02
1.292470e-26   1.207975e+02  0088   3.518473e+02   6.462349e-27
2.529810e+02  0089   3.507899e+02   3.231174e-27   1.157883e+02  0090
3.497362e+02   1.615587e-27   2.534917e+02  0091   3.487034e+02
8.077936e-28   1.113576e+02  0092   3.476741e+02   4.038968e-28
2.535066e+02  0093   3.466651e+02   2.019484e-28   1.072111e+02  0094
3.456569e+02   1.009742e-28   2.544931e+02  0095   3.446694e+02
5.048710e-29   1.031857e+02  0096   3.436812e+02   2.524355e-29
2.554610e+02  0097   3.427084e+02   1.262177e-29   9.899800e+01  0098
```

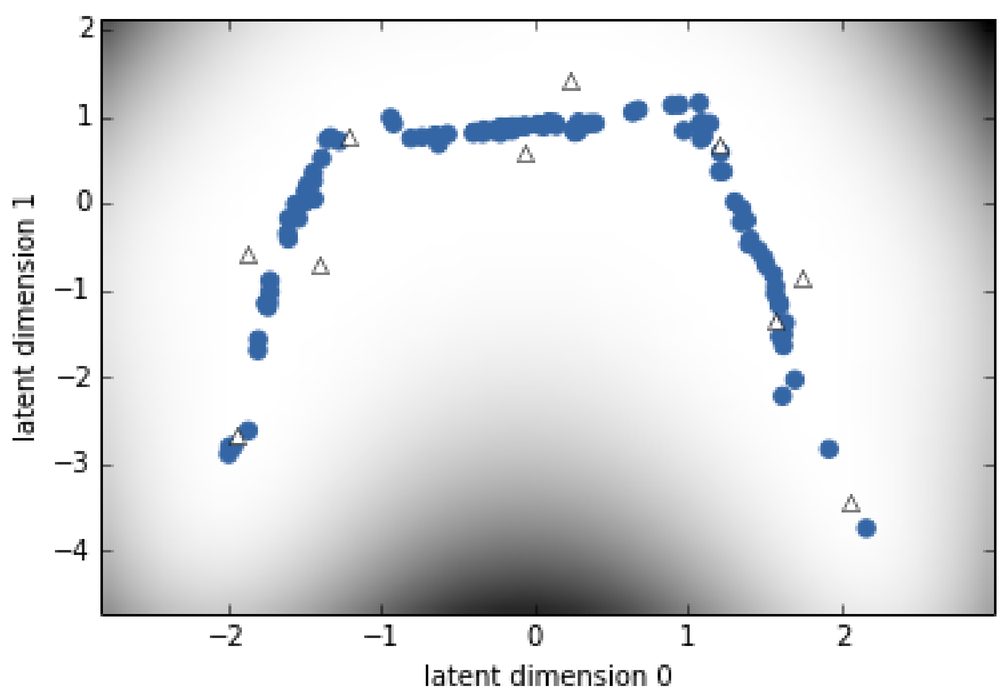
```
3.417366e+02   6.310887e-30   2.575918e+02  0098   3.417366e+02
3.155444e-30   9.524294e+01
-341.736608333
Warning: re-constraining these parameters
noise_variance
 I      F               Scale          |g|
 0001   3.308121e+02   1.000000e+00   3.848568e+03  0002
3.294127e+02   5.000000e-01   1.677208e+02  0003   3.281057e+02
2.500000e-01   1.816257e+02  0004   3.268982e+02   1.250000e-01
1.985234e+02  0005   3.256702e+02   6.250000e-02   1.286248e+02  0006
3.244673e+02   3.125000e-02   2.341516e+02  0007   3.230594e+02
1.562500e-02   9.514105e+01  0008   3.216197e+02   7.812500e-03
3.380565e+02  0009   3.193485e+02   3.906250e-03   7.212349e+01  0010
3.169905e+02   1.953125e-03   6.597005e+02  0011   3.044907e+02
9.765625e-04   5.268098e+01  0012   2.977472e+02   9.765625e-04
2.379833e+03  0013   2.970431e+02   4.882812e-04   9.812382e+01  0014
2.964119e+02   2.441406e-04   8.062892e+01  0015   2.958579e+02
1.220703e-04   9.498714e+01  0016   2.953221e+02   6.103516e-05
5.650727e+01  0017   2.948218e+02   3.051758e-05   1.136131e+02  0018
2.943218e+02   1.525879e-05   4.226404e+01  0019   2.938390e+02
7.629395e-06   1.309860e+02  0020   2.933192e+02   3.814697e-06
3.409743e+01  0021   2.928236e+02   1.907349e-06   1.435280e+02  0022
2.923247e+02   9.536743e-07   2.874499e+01  0023   2.918306e+02
4.768372e-07   1.507646e+02  0024   2.913021e+02   2.384186e-07
2.407615e+01  0025   2.907958e+02   1.192093e-07   1.620948e+02  0026
2.902300e+02   5.960464e-08   2.036665e+01  0027   2.896641e+02
2.980232e-08   1.917216e+02  0028   2.889716e+02   1.490116e-08
1.716894e+01  0029   2.882168e+02   7.450581e-09   2.742414e+02  0030
2.869010e+02   3.725290e-09   1.423708e+01  0031   2.857577e+02
1.862645e-09   4.576162e+02  0032   2.812669e+02   9.313226e-10
1.166612e+01  0033   2.789885e+02   4.656613e-10   5.410575e+02  0034
2.788599e+02   2.328306e-10   3.759074e+01  0035   2.787696e+02
1.164153e-10   1.591055e+01  0036   2.786810e+02   5.820766e-11
2.596733e+01  0037   2.785938e+02   2.910383e-11   1.582399e+01  0038
2.785069e+02   1.455192e-11   2.562052e+01  0039   2.784209e+02
7.275958e-12   1.568173e+01  0040   2.783351e+02   3.637979e-12
2.550357e+01  0041   2.782503e+02   1.818989e-12   1.556529e+01  0042
2.781655e+02   9.094947e-13   2.545256e+01  0043   2.780813e+02
4.547474e-13   1.535089e+01  0044   2.779976e+02   2.273737e-13
2.522939e+01  0045   2.779146e+02   1.136868e-13   1.530872e+01  0046
2.779146e+02   5.684342e-14   2.488127e+01  0047   2.779146e+02
2.273737e-13   2.488127e+01  0048   2.779146e+02   9.094947e-13
2.488127e+01  0048   2.779146e+02   3.637979e-12   2.488127e+01
-277.914632628
```

In [64]:
```
fig, ax = plt.subplots(1, 3, figsize=(12, 4), dpi=180)
ax[0].plot(m.X[:, 0], Y[:, 0], 'x', label='GPy')
ax[0].plot(X[:, 0], Y[:, 0], 'x', label='PCA')
ax[0].plot(Xt, Y[:, 0], 'x', label='init')
ax[0].legend(loc=2)

ax[1].plot(m.X[:, 0], Y[:, 1], 'x', label='GPy')
ax[1].plot(X[:, 0], Y[:, 1], 'x', label='PCA')
ax[1].plot(Xt, Y[:, 1], 'x', label='init')

ax[2].plot(m.X[:, 0], Y[:, 2], 'x', label='GPy')
```

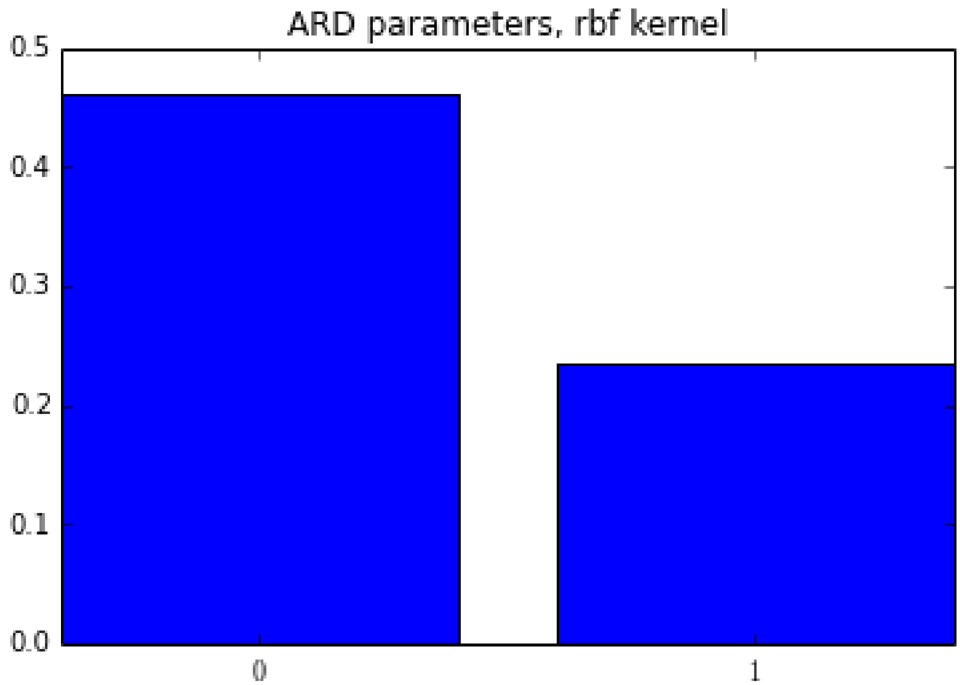
```
ax[2].plot(X[:, 0], Y[:, 2], 'x', label='PCA')
ax[2].plot(Xt, Y[:, 2], 'x', label='init')

#fig, (latent_axes, sense_axes) = plt.subplots(1, 2)
#plt.sca(latent_axes)
plt.figure()
m.plot_latent()

plt.figure()

kern.plot_ARD()
```

Out [64]: <matplotlib.axes.AxesSubplot at 0x9a95b50>





<matplotlib.figure.Figure at 0x974f6d0>



4 Parallel GPLVM results

This is a bit more difficult, as we need to copy the input data to a bunch of files.

In [65]:
```
P = 4
path = '../easydata/'

# First delete all current inputs & embeddings
filelist = glob.glob(path + "/inputs/*")
filelist.extend(glob.glob(path + "/embeddings/*"))
for f in filelist:
    os.remove(f)

# Open files for writing the divided dataset into
f = []
for p in xrange(1, P + 1):
    name = path + 'inputs/easy_' + str(p)
    f.append(open(name, 'w'))

# Divide up dataset
for y in Y:
    x_str = ",".join(np.char.mod('%f', y))
    randf = random.choice(f)
    randf.write(x_str)
    randf.write('\n')

for fi in f:
    fi.close()
```

Now set up the options and call the actual script.

In [70]:
```
options = {}
options['input'] = path + '/inputs/'
options['embeddings'] = path + '/embeddings/'
options['parallel'] = 'local'
options['iterations'] = 10
options['statistics'] = path + '/tmp'
options['tmp'] = path + '/tmp'
options['M'] = num_inducing
options['Q'] = Q
options['D'] = 3
options['fixed_embeddings'] = False
options['keep'] = True
options['load'] = False
options['fixed_beta'] = False
options['init'] = 'PCA'

#filelist = (glob.glob(path + "/embeddings/*"))
#for f in filelist:
#    os.remove(f)

import parallel_GPLVM
parallel_GPLVM.main(options)
```

```
Creating ../easydata//embeddings//easy_1.embedding.npy with 24 points
Creating ../easydata//embeddings//easy_1.variance.npy with 24 points
Creating ../easydata//embeddings//easy_2.embedding.npy with 30 points
Creating ../easydata//embeddings//easy_2.variance.npy with 30 points
Creating ../easydata//embeddings//easy_3.embedding.npy with 23 points
Creating ../easydata//embeddings//easy_3.variance.npy with 23 points
Creating ../easydata//embeddings//easy_4.embedding.npy with 23 points
Creating ../easydata//embeddings//easy_4.variance.npy with 23 points
Dispatching statistics Map-Reduce...
Done! statistics Map-Reduce took  0  seconds
Calculating global statistics...
Done! global statistics took  0  seconds
 I    F               Scale          |g|
Starting optimisation for 10 iterations
Dispatching statistics Map-Reduce...
Done! statistics Map-Reduce took  0  seconds
Calculating global statistics...
Done! global statistics took  0  seconds
Dispatching statistics Map-Reduce...
Done! statistics Map-Reduce took  0  seconds
Calculating global statistics...
Done! global statistics took  0  seconds

 01   1.553467e+03   1.000000e+00   8.230386e+06

Calling local optimisation...
Dispatching embeddings Map-Reduce to run in background...
Waiting for embeddings Map-Reduce to finish...
Done! embeddings Map-Reduce took  0  seconds
Dispatching statistics Map-Reduce...
Done! statistics Map-Reduce took  0  seconds
Calculating global statistics...
Done! global statistics took  0  seconds
Dispatching statistics Map-Reduce...
Done! statistics Map-Reduce took  0  seconds
```

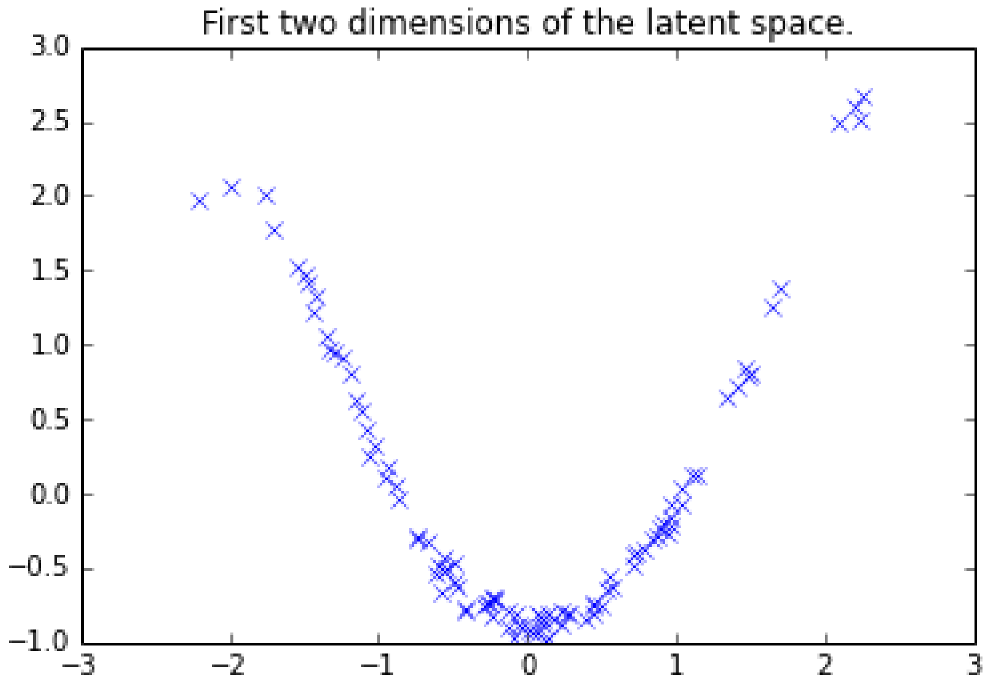
```
Calculating global statistics...
Done! global statistics took  0  seconds

 02   1.537431e+03   4.000000e+00   2.585053e+06

Calling local optimisation...
Dispatching embeddings Map-Reduce to run in background...
Waiting for embeddings Map-Reduce to finish...
Done! embeddings Map-Reduce took  0  seconds
Dispatching statistics Map-Reduce...
Done! statistics Map-Reduce took  0  seconds
Calculating global statistics...
Done! global statistics took  0  seconds
Dispatching statistics Map-Reduce...
Done! statistics Map-Reduce took  0  seconds
Calculating global statistics...
Done! global statistics took  0  seconds

 03   1.521132e+03   2.000000e+00   2.563256e+06

 03   1.521132e+03   2.000000e+00   2.563256e+06

Final global_statistics
{'alpha': array([[ 0.9968579,  0.9963545]]), 'beta': array([[
1.96194126]]), 'Z': array([[ 0.3315234 , -0.83912678],
       [-0.88165158,  0.20712083],
       [-0.28739926, -0.8759714 ],
       [-2.29176749,  2.10228504],
       [ 1.50356233,  0.76280535],
       [-0.30459867, -0.54154114],
       [-1.75789666,  1.96844668],
       [ 0.23797668, -0.74670968],
       [ 0.96231138, -0.13567684],
       [-1.04163384,  0.59424671]]), 'sf2': array([[ 0.95894112]])}
Dispatching statistics Map-Reduce...
Done! statistics Map-Reduce took  0  seconds
Calculating global statistics...
Done! global statistics took  0  seconds
final F=-1521.13183714
```

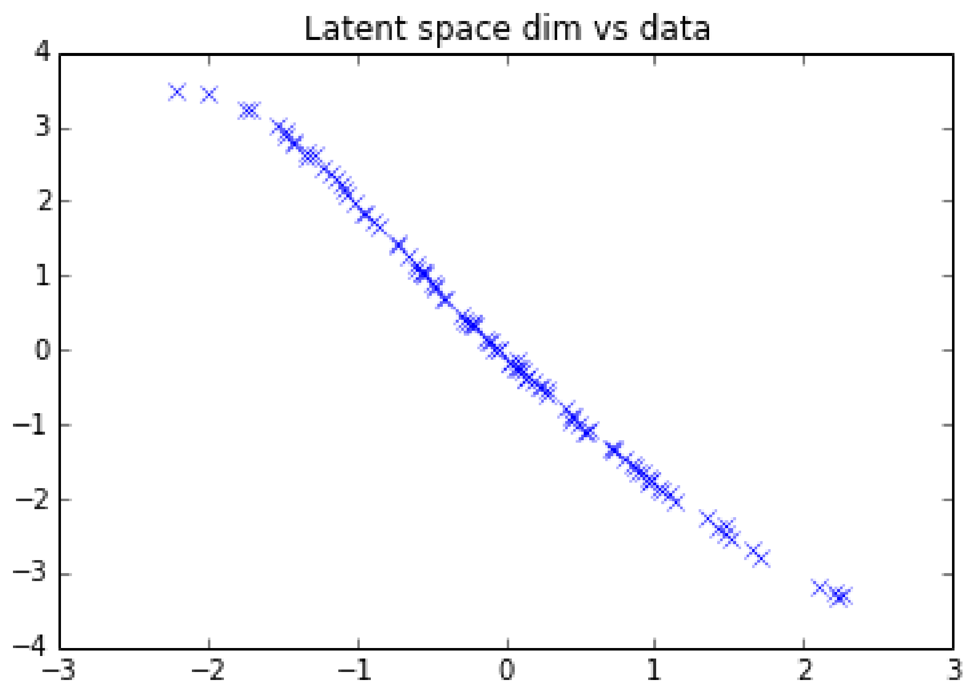
In [75]:

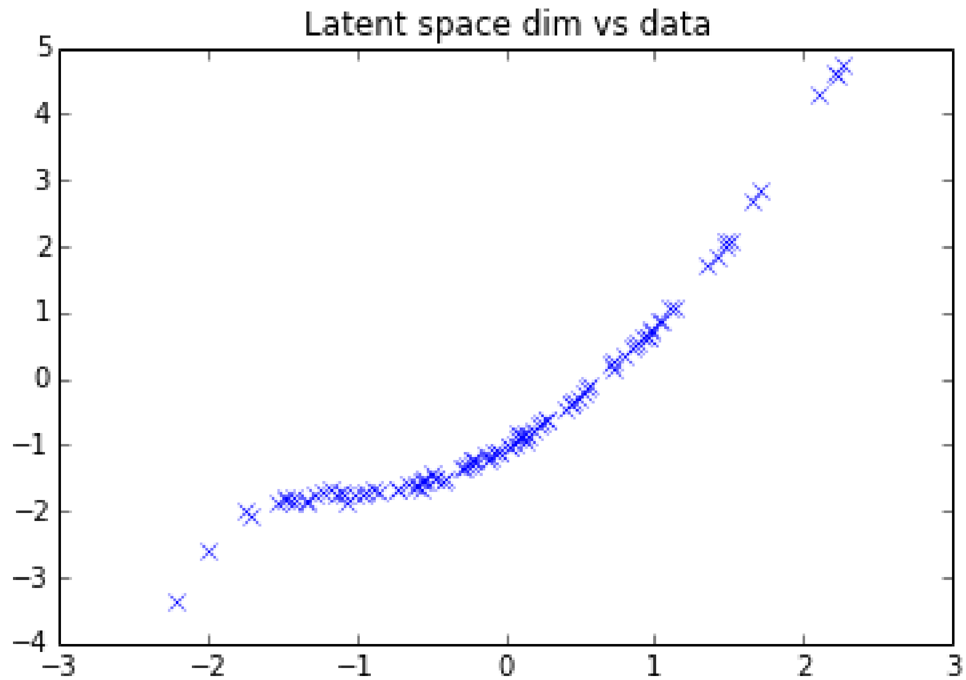
```
reload(show_embeddings)
import show_embeddings
class empty:
    pass
disp_opt = empty()
disp_opt.verbose = True
disp_opt.dimension = [0, 1]
disp_opt.output_dimension = [0, 1, 2]
disp_opt.plot2d = True
disp_opt.plot3d = False
args = [path]
show_embeddings.run(disp_opt, args)
```

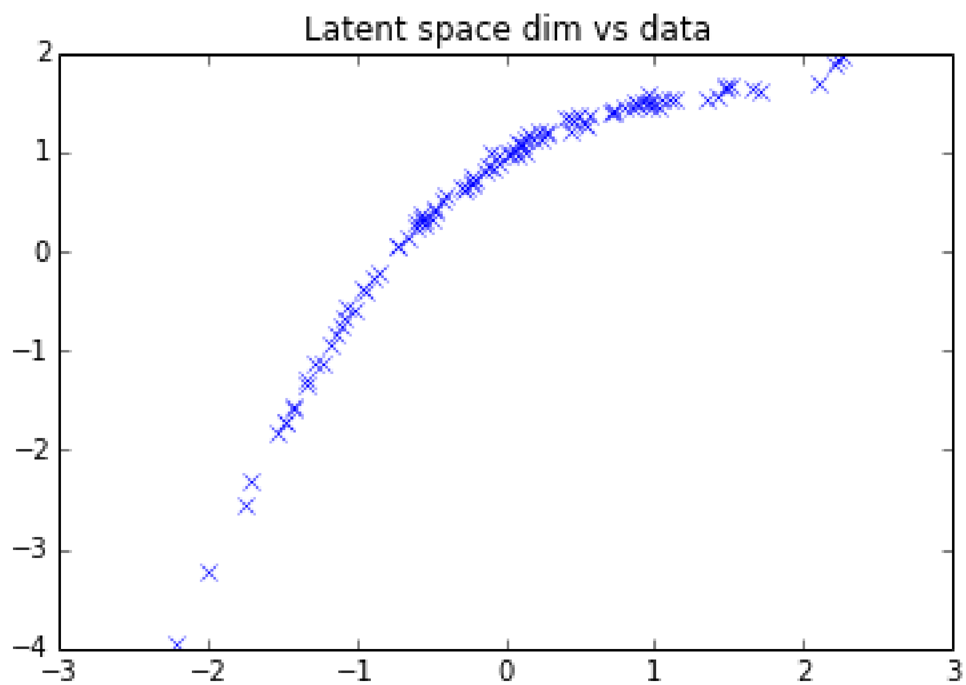
```
Displaying X in '../easydata/'...
alpha: [ 0.9968579  0.9963545]
beta : 1.96194126426
```

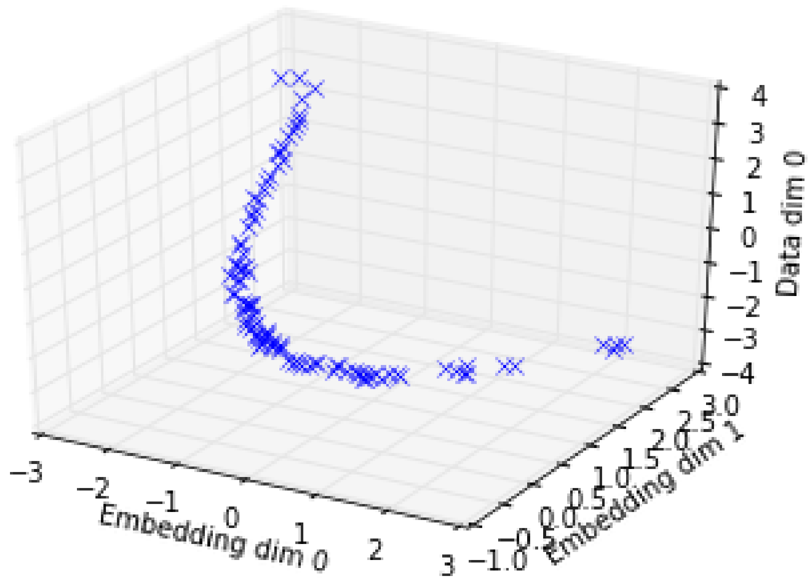
```
sf2  : 0.958941121247
```



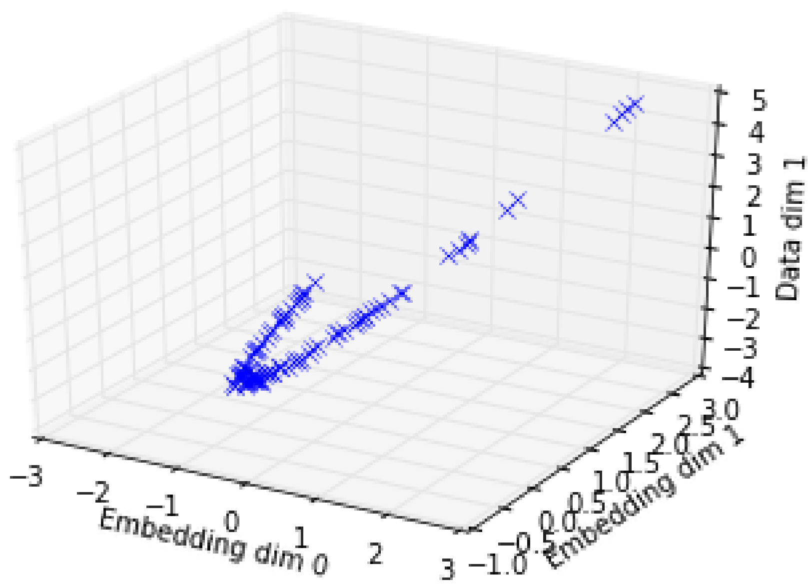



Latent space dim vs data
5
4
3
2
1
0
−1
−2
−3
−4
−3
−2
−1
0
1
2
3

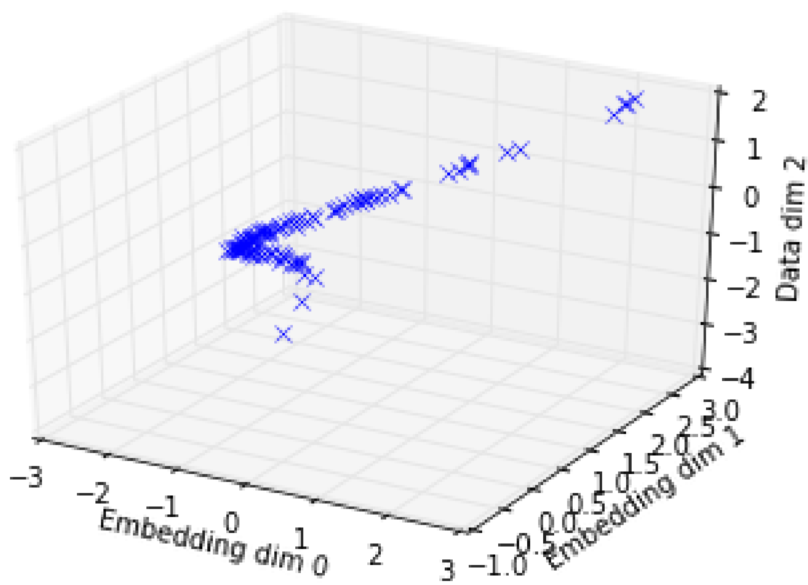

Latent space dim vs data
2
1
0
−1
−2
−3
−4
−3
−2
−1
0
1
2
3



Data dim 0
Embedding dim 0
Embedding dim 1



Data dim 1
Embedding dim 0
Embedding dim 1



Embedding dim 0
Embedding dim 1
Data dim 2
−3 −2 −1 0 1 2 3
−1.0 −0.5 0.0 0.5 1.0 1.5 2.0 2.5 3.0
−4 −3 −2 −1 0 1 2