The `pumadyn` datasets

Zoubin Ghahramani

April 29, 1996

1 Introduction

The `pumadyn` datasets are a family of datasets synthetically generated from a realistic simulation of the dynamics of a Puma 560 robot arm. The tasks associated with these datasets consist of predicting the angular acceleration of one of the links of the robot arm, given the angular positions, velocities, torques, and in some cases, other dynamic parameters of the robot arm. There are 8 different `pumadyn` datasets, named `pumadyn-`xyz, consisting of all combinations of the following three binary attributes

x: low (8) vs. high (32) input dimensionality

y: highly nonlinear (n) vs. fairly linear (f)

z: medium noise (m) vs. high noise (h)

For example, dataset `pumadyn-32nh` has 32 inputs, high nonlinearity, and high noise. The `pumadyn` datasets were generated by Zoubin Ghahramani (zoubin@cs.toronto.edu) in Matlab using the Matlab Robotics Toolbox (Release 3). The Matlab Robotics Toolbox was written by Peter I. Corke and is available freely from The MathWorks or from the author.

Whereas the input dimensionality of an generated dataset is easy to control, the nonlinearity and amount of noise are generally not. To guide us in producing datasets with approximately controlled amounts of nonlinearity and noise we defined the following two measures:

- Index of nonlinearity:

$$I_{NL} = \frac{E_L}{E_C},$$

where E_L is the squared error loss of the best linear fit on noise-free data,[1] and E_C is the squared error for the corresponding constant fit. Note that E_C is the variance of the target, and therefore I_{NL} is one minus the proportion variance explained by a linear model. Datasets with $I_{NL} < 0.05$ were classified as having *low* nonlinearity, and $I_{NL} > 0.2$, *high* nonlinearity.

- Index of unpredictability:

$$I_U = \frac{E_N}{E_C},$$

  where E_N is the expected squared loss on noisy data of the best nonlinear fit, i.e. the expected value of the function. In practice, we estimated E_N as follows. For each input in the data set, we used the simulator and desired noise model to generate multiple targets. The average variance of these targets was taken as a measure of E_N. Data with $I_U < 0.01$ were classified as having *low* noise, $0.01 \leq I_U < 0.05$ as *medium* noise, and $I_U \geq 0.2$ as *high* noise.

2 Datasets with 8 inputs

The forward dynamics of a revolute robot arm are described by the function

$$\ddot{\theta} = f(\theta, \dot{\theta}, \tau, \Phi), \tag{1}$$

where θ, $\dot{\theta}$, and τ are the vectors of joint angles, velocities, and torques, and Φ is the vector of parameters describing the kinematics and dynamics of the arm. For the six degree of freedom Puma robot arm, θ, $\dot{\theta}$ and $\ddot{\theta}$ are 6-dimensional vectors. Eight dimensional input data were generated by fixing $\theta_4 = \theta_5 = \theta_6 = \dot{\theta}_4 = \dot{\theta}_5 = \dot{\theta}_6 = \tau_3 = \tau_4 = \tau_5 = \tau_6 = 0$ and sampling the other variables uniformly from the ranges given in Table 1.

The range parameter β was used to control the nonlinearity attribute of the dataset. To control the unpredictability of the dataset, zero-mean Gaussian noise with standard deviation γ was added to each element of the

[1] Although noise-free data is trivial to obtain from deterministic simulators, for inherently stochastic simulators it is necessary to average the output of many runs starting from the same input.

name	description	range
θ_1	ang pos of link 1	$\beta[-\pi/2, \pi/2]$
θ_2	ang pos of link 2	$\beta[-\pi/2, \pi/2]$
θ_3	ang pos of link 3	$\beta[-\pi/2, \pi/2]$
$\dot{\theta}_1$	ang vel of link 1	$\beta[-\pi/2, \pi/2]$
$\dot{\theta}_2$	ang vel of link 2	$\beta[-\pi/2, \pi/2]$
$\dot{\theta}_3$	ang vel of link 3	$\beta[-\pi/2, \pi/2]$
τ_1	torque at joint 1	$\beta[-1/2, 1/2]$
τ_2	torque at joint 2	$\beta[-1/2, 1/2]$

Table 1: Input descriptions and sampling ranges.

Dataset	β	γ	I_{NL}	I_U
`pumadyn-8nm`	1.2	0.12	0.455	0.029
`pumadyn-8nh`	1.2	0.40	0.564	0.253
`pumadyn-8fm`	0.6	0.12	0.034	0.041
`pumadyn-8fh`	0.6	0.40	0.033	0.300

Table 2: Empirically estimated indices of nonlinearity and unpredictability for the 8 input datasets.

θ, $\dot{\theta}$ and τ vectors, resulting in the noisy vectors θ', $\dot{\theta}'$ and τ'. The target was $\ddot{\theta}'_3 = f(\theta', \dot{\theta}', \tau', \Phi) + \epsilon$, where ϵ was also zero-mean Gaussian noise with standard deviation γ.

Each data point therefore consists of the inputs $[\theta_1\ \theta_2\ \theta_3\ \dot{\theta}_1\ \dot{\theta}_2\ \dot{\theta}_3\ \tau_1\ \tau_2]$ and the target $\ddot{\theta}'_3$. Note that there are three sources of uncertainty: (1) input noise (e.g. from observing θ_1 rather than θ'_1), (2) uncertainty from not observing some inputs (e.g. θ'_4), and (3) output noise from observing $\ddot{\theta}'_3$ rather than $f(\theta', \dot{\theta}', \tau', \Phi)$. Table 2 gives the approximate indices of nonlinearity and unpredictability for the four datasets with 8 inputs. Appendix A shows sample Matlab code for generating this data.

3 Datasets with 32 inputs

Data with 32 inputs were obtained from dynamic equation (1) by allowing more of the joint angles, velocities, torques, and some of the dynamic parameters of the Puma arm to vary. The inputs are described in Table 3; the goal was to predict the acceleration of link 6, $\ddot{\theta}_6$. The Δm, Δl and Δb inputs are multiplicative perturbations to the mass, length, and viscous friction parameters of the Puma arm. For example, since $m_2 = 17.4$ kg for the Puma, $\Delta m_1 = 0.8$ signifies that the mass of the simulated robot arm is 13.92 kg. Note that certain parameters of the Puma arm have a value of 0, for which the corresponding multiplicative perturbations are irrelevant inputs to the function (Table 4).

name	description	range
$\theta_1 \ldots \theta_6$	ang pos of links 1 to 6	$\beta[-\pi/2, \pi/2]$
$\dot{\theta}_1 \ldots \dot{\theta}_6$	ang vels of links 1 to 6	$\beta[-\pi/2, \pi/2]$
$\tau_1 \ldots \tau_5$	torques at joints 1 to 5	$\beta[-50, 50]$
$\Delta m_1 \ldots \Delta m_5$	change in mass of links 1 to 5	$[1 - \beta/2, 1 + \beta]$
$\Delta a_1 \ldots \Delta a_5$	change in length of links 1 to 5	$[1 - \beta/2, 1 + \beta]$
$\Delta b_1 \ldots \Delta b_5$	change in viscous friction of links 1 to 5	$[1 - \beta/2, 1 + \beta]$

Table 3: Input descriptions and sampling ranges for 32 input datasets.

parameter	link 1	link 2	link 3	link 4	link 5
mass (m)	0	17.4	4.8	0.82	0.34
length (a)	0	0.4318	0.0203	0	0
friction (b)	0.0015	0.0008	0.0014	0.0001	0.0001

Table 4: Relevant parameters of the Puma arm (Corke, 1996).

The range parameter β was used to control the nonlinearity of the dataset, and the noise parameter γ to control the unpredictability. As in the 8 input case, zero-mean Gaussian noise with standard deviation γ was added to each element of θ, $\dot{\theta}$ and τ, and to the target $\ddot{\theta}_6$. No noise was added to Δm, Δl, or Δb—they were assumed to be known parameters of the robot arm.

Table 5 documents the empirically estimated indices of nonlinearity and unpredictability for the datasets. Appendix B shows sample Matlab code for generating this data.

Dataset	β	γ	I_{NL}	I_U
pumadyn-32nm	1.5	0.005	0.760	0.028
pumadyn-32nh	1.5	0.020	0.773	0.278
pumadyn-32fm	0.7	0.003	0.038	0.025
pumadyn-32fh	0.7	0.020	0.035	0.538

Table 5: Empirically estimated indices of nonlinearity and unpredictability for the 32 input datasets.

A Matlab code for pumadyn-8

```
% Generate 8 input Data from Forward Dynamics of Puma 560
% Using the Robotics Toolbox (Corke, 1996) available at
% http://www.dem.csiro.au/dmt/programs/autom/pic/matlab.html

puma560;                % read parameters of the Puma arm
N=8192;                 % number of data points
beta=1.2;               % nonlinearity parameter beta
gamma=0.12;             % unpredictability parameter gamma

X=zeros(N,8);           % inputs
Y=zeros(N,1);           % outputs

for i=1:N;
  thetan=pi*[(rand(3,1)-0.5)*beta; zeros(3,1)];      % generate positions
  thetadn=pi*[(rand(3,1)-0.5)*beta; zeros(3,1)];     % generate velocities
  torquen=[(rand(2,1)-0.5)*beta; zeros(4,1)];        % generate torques

  theta=thetan+randn(6,1)*gamma;                     % add Gaussian noise
  thetad=thetadn+randn(6,1)*gamma;                   % to inputs
  torque=torquen+randn(6,1)*gamma;
  thetadd=accel(p560,theta,thetad,torque);           % compute forward dynamics

  X(i,:)=[thetan(1:3)' thetadn(1:3)' torquen(1:2)']; % assemble input vector
  Y(i,:)=thetadd(3)+gamma*randn;                     % add noise to target
end;
Dat=[X Y];                                           % assemble data matrix
```

B Matlab code for pumadyn-32

```
% Generate 32 input Data from Forward Dynamics of Puma 560
% Using the Robotics Toolbox (Corke, 1996) available at
% http://www.dem.csiro.au/dmt/programs/autom/pic/matlab.html

puma560;                % read parameters of the Puma arm
N=8192;                 % number of data points
beta=1.5;               % nonlinearity parameter beta
gamma=0.02;             % unpredictability parameter gamma
X=zeros(N,32);          % inputs
Y=zeros(N,1);           % outputs

for i=1:N;
  thetan=beta*pi*(rand(6,1)-0.5);            % generate positions
  thetadn=beta*pi*(rand(6,1)-0.5);           % generate velocities
  torquen=beta*[(rand(5,1)-0.5); 0]*100;     % generate torques
  dmass=[1+beta*(rand(5,1)-1/3)*(3/2); 1];   % generate change in mass
  dlen=[1+beta*(rand(5,1)-1/3)*(3/2); 1];    % generate change in length
  dB=[1+beta*(rand(5,1)-1/3)*(3/2); 1];      % generate change in friction
  dinert=dmass;                              % scale inertia by mass

  dyn=p560;                                  % modify all dynamic
  dyn(:,6)=dyn(:,6).*dmass;                  % parameters accordingly
  dyn(:,2)=dyn(:,2).*dlen;
  dyn(:,18)=dyn(:,18).*dB;      dyn(:,10)=dyn(:,10).*dinert;
  dyn(:,11)=dyn(:,11).*dinert;  dyn(:,12)=dyn(:,12).*dinert;

  theta=thetan+randn(6,1)*gamma;             % add noise to inputs
  thetad=thetadn+randn(6,1)*gamma;
  torque=torquen+[randn(5,1)*gamma; 0];
  thetadd = accel(dyn,theta,thetad,torque); % compute forward dynamics

  X(i,:)=[thetan(1:6)' thetadn(1:6)' torquen(1:5)' ...
                         dmass(1:5)' dlen(1:5)' dB(1:5)'];
  Y(i,:)=thetadd(6)+randn*gamma;             % add noise to output
end;
Dat=[X Y];                                   % assemble data matrix
```