**Documentation of Synthetic Data (Corr) for project JiaXing**

**Preamble**

The data synthesised/transformed from real projects are designed for testing the effectiveness and efficiency of each of the JiaXing tuning projects (YangZhou etc).

They are stored in .json file format, with ‘num\_arg\_vals’ being a list that captures the number of values in each dimension (and also how many dimensions are in this synthetic data), ‘synthetic data’ storing the synthesised randomly generated tuning scores, ‘theoretical data’ storing the synthesised mean of randomly generated tuning scores. The final object in the json is a dictionary under ‘max’, which contains the synthesised maximum score ‘synth\_max’, theoretical (mean of randomly generated) maximum score ‘theor\_max’, synthesised maximum coordinates ‘synth\_max\_coord’ and theoretical maximum coordinates ‘theo\_max\_coord’.

**Documentation**

This dataset contains 6 batch of data

Batch 1: Synthesised data with each type of characteristics captured (2688 sets; 20.7GB)

Batch 2: Synthesised data with 3 particular type of mean characteristics - with random inversion (816; 8.23GB)

Batch 3: Synthesised data with randomly chosen characteristics (60; 1.85GB)

Batch 4: Real accuracy/r2 data from past projects (50; 5.1MB)

Batch 7: Synthesised data like 3, but with larger variances (120; 4.87GB)

Batch 1 2 and 3

General concepts:

Each point in the synthetic data is generated by random generator according to:

1. Generate a random vector from ~N(**Mu**, **V**)
2. The sum of the vector is added to 0.5 to gain this observation

\*note observations may go above 1 or subzero

**Mu** (mean vector)

For each dimension (each individual hyperparameter), it is assumed with rising values of actual hyperparameter values, the ‘mean’ of the contribution to the score.

Each dimension thus can be generated with a trend/pattern:

‘take\_off’: first few values contribute 0, then gradually increasing by 0.01

‘flat’: all values contribute 0

‘v\_shape’: values first decrease, then increase

‘down\_flat\_up’: values first decrease, then contribute 0, then increase again

‘up\_flat\_up’: values first increase, then contribute 0, then increase again

‘flat\_up\_flat’: values first contribute 0, then increase, then contribute 0

‘flat\_up\_down\_flat’: values first contribute 0, then increase, then decrease, then contribute 0

Note: all steps in trends move in magnitude of 0.01. (either -0.01, 0 or +0.01)

Line chart

Description automatically generated with medium confidenceNote: when in action, all these points of change are randomly generated, within a reasonable space (i.e. won’t take\_off from the leftmost point.

**V** (covariance matrix)

The standard deviations are either

0.0005 or 0.0025

(1/20 or 1/4 of the typical gaps in the means); whilst the correlations can be anything in

{-0.67, -0.33, 0, 0.33, 0.67, rand }.

To generate the random matrix, of course the correlations must be multiplied by both variances to get a covariance.

Batch 1:

D\_VAL = {2: (5, 7, 9, 13),

3: (5, 7, 9, 13),

4: (5, 7, 9),

5: (5, 7, 9, 'rand'),

7: (5, 'rand')}

for d in (2, 3, 4, 5, 7):

for d\_val in D\_VAL[d]:

for pattern in ['take\_off', 'v\_shape', 'down\_flat\_up', 'up\_flat\_up', 'flat\_up\_flat', 'flat\_up\_down\_flat', 'rand']:

if pattern == 'flat\_up\_down\_flat' and (d\_val == 5 or d\_val == 'rand'):

continue

for sd in (0.001, 0.005):

for corr in (-0.67, -0.33, 0, 0.33, 0.67, 'rand'):

for rep in (1, 2):

Note: each combination repeated two times

Note: whilst the sd was intended to be 0.001 and 0.005, post-generation validation demonstrated that it is in fact 0.0005 and 0.0025

Note: when d\_val = 5, or there is chance of <=7 in d\_val = ‘rand’, will skip ‘flat\_up\_down\_flat’ as hyperparameter mean pattern as the generating mechanism of ‘flat\_up\_down\_flat’ requires there to be at least 7 values in the dimension

Note: when pattern is not rand, all dimensions utilise the same pattern

Note: when d\_val is not rand, all dimensions have the same values

Note: when correlation is not rand, all dimensions are correlated to each other by ‘corr’

Batch 2:

-uses a slightly different mechanism to ‘v\_shape’, ‘down\_flat\_up’ and ‘flat\_up\_down\_flat’ in which the shape could be inverted (whereas batch 1 for simplicity sake only has the described direction as the patterns’ names).

D\_VAL = {2: (7, 9, 13),

3: (7, 9, 13,),

4: (7, 9),

5: (7, 9, 'rand'),

7: ('rand',)}

i = 0

for d in (2, 3, 4, 5, 7):

for d\_val in D\_VAL[d]:

for pattern in ['v\_shape', 'down\_flat\_up', 'flat\_up\_down\_flat']:

if pattern == 'flat\_up\_down\_flat' and (d\_val == 'rand'):

continue

for sd in (0.001, 0.005):

for corr in (-0.67, -0.33, 0, 0.33, 0.67, 'rand'):

for rep in (1, 2):

Note: each combination repeated two times

Note: all other notes same as batch 1

Batch 3:

D\_VAL = {2: ('rand',),

3: ('rand',),

4: ('rand',),

5: ('rand',),

6: ('rand',),

7: ('rand',)}

for d in (2, 3, 4, 5, 6, 7):

for d\_val in D\_VAL[d]:

for pattern in ['rand',]:

for sd in (0.005,):

for corr in ('rand',):

for rep in (1, 2, 3, 4, 5, 6, 7, 8, 9, 10):

Note: each combination repeated ten times

Note: all other notes same as batch 1

Batch 4

-comes from projects including ML Asmt 2; ADS Asmt 1; ADS Asmt 2 and New Brownlow Predictor project

-converted into same format as synthesised data using *‘./Batch 4 5 6 Creator/synthetic\_data\_generator\_batch4.ipynb’*

Batch 4

-accuracy, r2

Batch 7

Batch 7:

D\_VAL = {2: ('rand',),

3: ('rand',),

4: ('rand',),

5: ('rand',),

6: ('rand',),

7: ('rand',)}

for d in (2, 3, 4, 5, 6, 7):

for d\_val in D\_VAL[d]:

for pattern in ['rand',]:

for sd in (0.01, 0.02):

for corr in ('rand',):

for rep in (1, 2, 3, 4, 5, 6, 7, 8, 9, 10):

Note: each combination repeated ten times

Note: all other notes same as batch 1