ABU 量化系统 简介（版本 0.1）

- author = 'BBFamily'
- email = 'bbfamily@126.com'
- weixin = 'aaaabbbuu'

第八部分 解决方案B ‘非均衡胜负收益’带来的必然’非均衡胜负比例‘，目标由’因子‘的能力解决一部分，’模式识别‘提升关键的一部分

`python import ZEnv import ZLog import ZCommonUtil %matplotlib inline`

`python from UmpMain import UmpMainClass from UmpJump import UmpJumpClass from MlFiterJumpPd import MlFiterJumpPdClass`

`python fn = './data/cache/golden_n6_test_abu' key = 'golden_n6_test_abu' orders_pd_test = ZCommonUtil.load_hdf5(fn, key) orders_pd_test.shape`

```
(4837, 31)
```

`python fn = './data/cache/golden_n6_train_abu' key = 'golden_n6_train_abu' orders_pd_train = ZCommonUtil.load_hdf5(fn, key) orders_pd_train.shape`

```
(42538, 31)
```

使用全量测试集数据

`python fn = './data/cache/orders_pd_ump_hit_predict_abu' key = 'orders_pd_ump_hit_predict_abu' orders_pd_ump = ZCommonUtil.load_hdf5(fn, key) orders_pd_ump.shape`

```
(47374, 39)
```

jump ump 辅助裁决

`python ump_jump = UmpJumpClass(orders_pd_ump, MlFiterJumpPdClass,` `dd_threshold=21)`

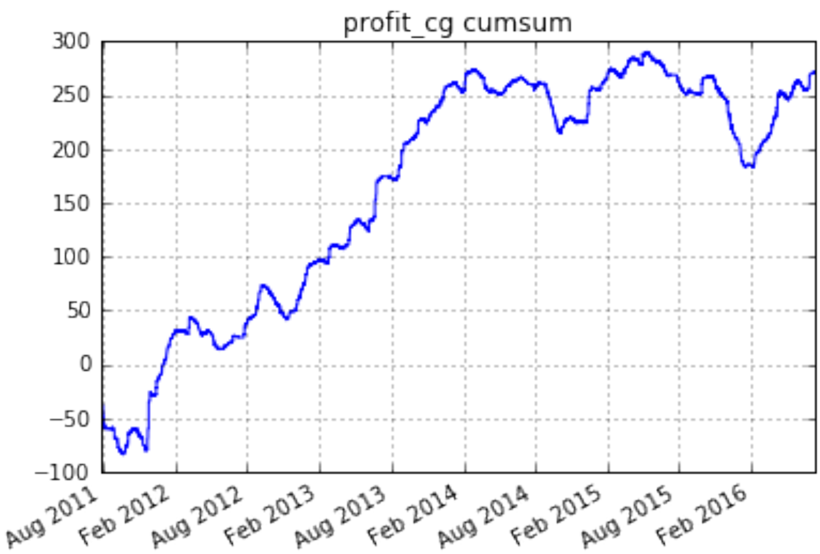
`python ZLog.info(ump_jump.fiter.df.shape) ump_jump.fiter.df.head()`

```
(5035, 3)
```

	result	jump_power	diff_days
2015-07-28	1	-1.382554	6
2015-07-28	0	-1.084962	21
2015-07-28	0	-3.415892	5
2015-07-28	0	-3.236708	4
2015-07-28	0	-1.307611	13

`python ump_jump.show_general()`

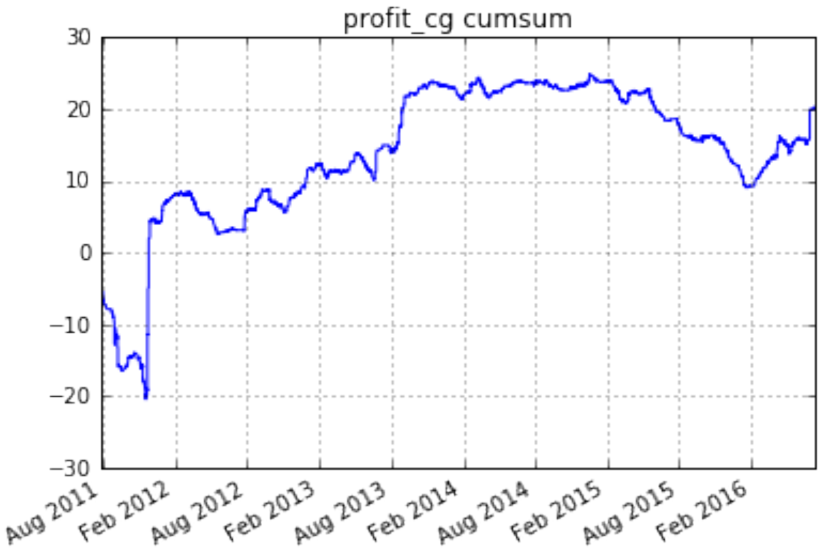
```
all fit order = (44906, 39)
win rate = 0.500757137131
profit_cg.sum() = 272.117613217
win mean = 0.0743788075658 loss_mean = -0.0626291433739
```



跳空监控区，收益小于整个区间

`python ump_jump.show_general(use_fiter=True)`

```
all fit order = (5035, 39)
win rate = 0.498907646475
profit_cg.sum() = 20.2331512267
win mean = 0.0673137719871 loss_mean = -0.0591414557032
```



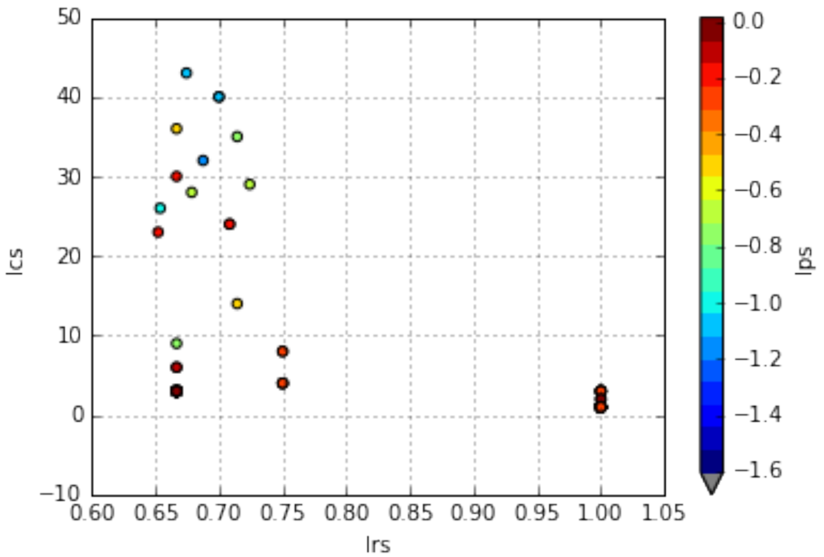
make a joke

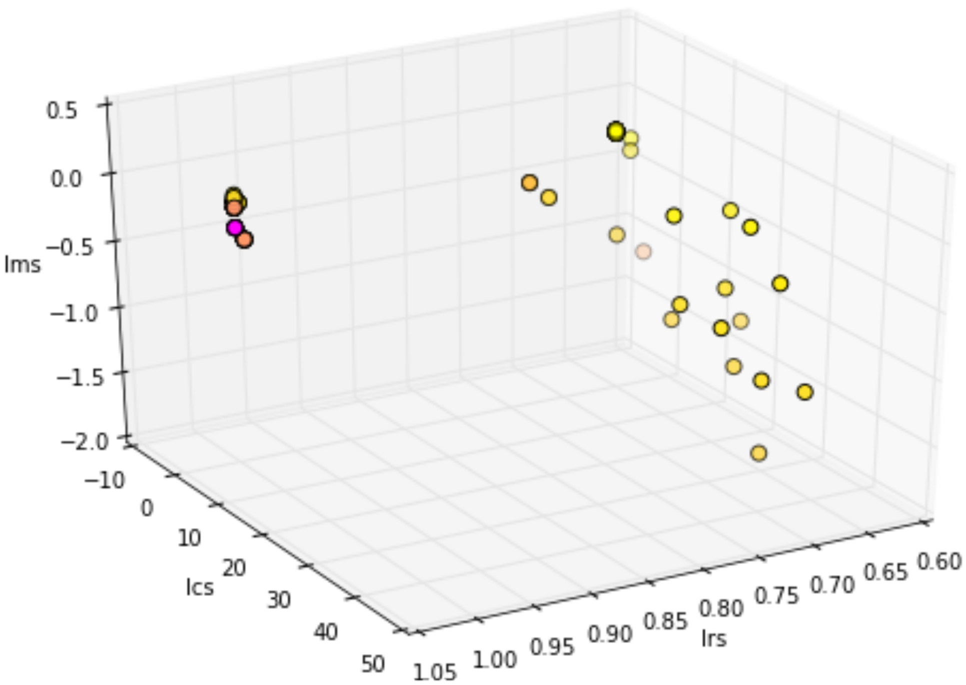
`python ump_jump.fiter().estimator.svc()` `ump_jump.fiter().train_test_split_xy()`

```
(5035, 2)
(4531, 2)
(504, 2)
accuracy = 0.56
precision_score = 0.57
recall_score = 0.54
             precision    recall  f1-score   support

       loss       0.55      0.59      0.57       247
        win       0.57      0.54      0.56       257

avg / total       0.56      0.56      0.56       504



Confusion Matrix  [[145 102]
 [119 138]]
          Predicted
         |  0  |  1  |
         |-----|-----|
       0 | 145 | 102 |
Actual   |-----|-----|
       1 | 119 | 138 |
         |-----|-----|
```

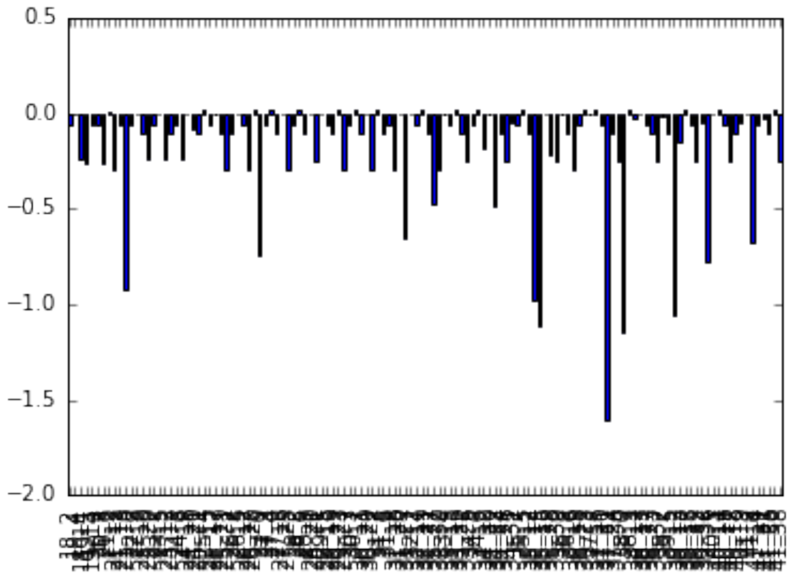
p_ncs的选择要依据数据集的大小

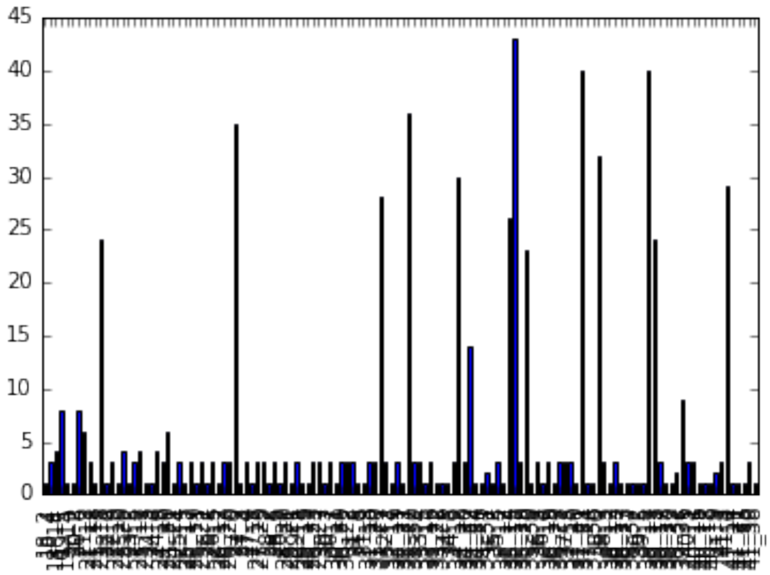
`python import numpy as np`
`ump_jump.gmm_component_filter(p_ncs=np.arange(18, 42), threshold=0.65)`





lms
lcs
lrs





	lcs	lrs	lps	lms
18_2	1.0	1.000000	-0.059441	-0.059441
18_4	3.0	0.666667	-0.003412	-0.001137
18_14	4.0	0.750000	-0.233595	-0.058399
19_5	8.0	0.750000	-0.265865	-0.033233
19_14	1.0	1.000000	-0.059441	-0.059441
20_2	1.0	1.000000	-0.059441	-0.059441
20_8	8.0	0.750000	-0.265865	-0.033233
21_2	6.0	0.666667	0.011840	0.001973
21_12	3.0	1.000000	-0.294978	-0.098326
21_13	1.0	1.000000	-0.059441	-0.059441
21_18	24.0	0.708333	-0.921051	-0.038377
22_2	1.0	1.000000	-0.059441	-0.059441
22_9	3.0	0.666667	-0.003412	-0.001137
22_10	1.0	1.000000	-0.101029	-0.101029
22_20	4.0	0.750000	-0.233595	-0.058399
23_2	1.0	1.000000	-0.059441	-0.059441
23_5	3.0	0.666667	-0.003412	-0.001137

23_12	4.0	0.750000	-0.233595	-0.058399
23_13	1.0	1.000000	-0.101029	-0.101029
24_3	1.0	1.000000	-0.059441	-0.059441
24_6	4.0	0.750000	-0.233595	-0.058399
24_18	3.0	0.666667	-0.003412	-0.001137
24_20	6.0	0.666667	-0.077712	-0.012952
24_21	1.0	1.000000	-0.101029	-0.101029
25_4	3.0	0.666667	0.015253	0.005084
25_15	1.0	1.000000	-0.059441	-0.059441
25_17	3.0	0.666667	-0.003412	-0.001137
25_19	1.0	1.000000	-0.101029	-0.101029
25_21	3.0	1.000000	-0.294978	-0.098326
26_2	1.0	1.000000	-0.101029	-0.101029
...	...	...	...	...
37_36	1.0	1.000000	-0.248848	-0.248848
38_0	32.0	0.687500	-1.140307	-0.035635
38_3	3.0	0.666667	0.015253	0.005084
38_12	1.0	1.000000	-0.028857	-0.028857
38_13	3.0	0.666667	-0.003412	-0.001137
38_17	1.0	1.000000	-0.059441	-0.059441
38_23	1.0	1.000000	-0.101029	-0.101029
38_37	1.0	1.000000	-0.248848	-0.248848
39_2	1.0	1.000000	-0.009235	-0.009235
39_5	1.0	1.000000	-0.101029	-0.101029
39_12	40.0	0.700000	-1.052118	-0.026303
39_13	24.0	0.708333	-0.150335	-0.006264
39_17	3.0	0.666667	0.015253	0.005084
39_18	1.0	1.000000	-0.059441	-0.059441

39_22	1.0	1.000000	-0.248848	-0.248848
39_32	2.0	1.000000	-0.046130	-0.023065
39_36	9.0	0.666667	-0.780621	-0.086736
40_3	3.0	0.666667	-0.003412	-0.001137
40_11	3.0	0.666667	0.015253	0.005084
40_16	1.0	1.000000	-0.059441	-0.059441
40_17	1.0	1.000000	-0.248848	-0.248848
40_18	1.0	1.000000	-0.101029	-0.101029
40_19	2.0	1.000000	-0.046130	-0.023065
41_3	3.0	0.666667	-0.003412	-0.001137
41_8	29.0	0.724138	-0.679913	-0.023445
41_14	1.0	1.000000	-0.059441	-0.059441
41_21	1.0	1.000000	-0.028857	-0.028857
41_25	1.0	1.000000	-0.101029	-0.101029
41_30	3.0	0.666667	0.015253	0.005084
41_38	1.0	1.000000	-0.248848	-0.248848

128 rows × 4 columns

`python ump_jump.nts['25_15']`

	result	jump_power	diff_days	ind	ss	profit
2014-01-27	0	-85.133358	14	2062	15	-0.059441

跳空能量向下两倍以上，10天左右的认为不应买入

`python ii = ump_jump.nts['39_12'] ii`

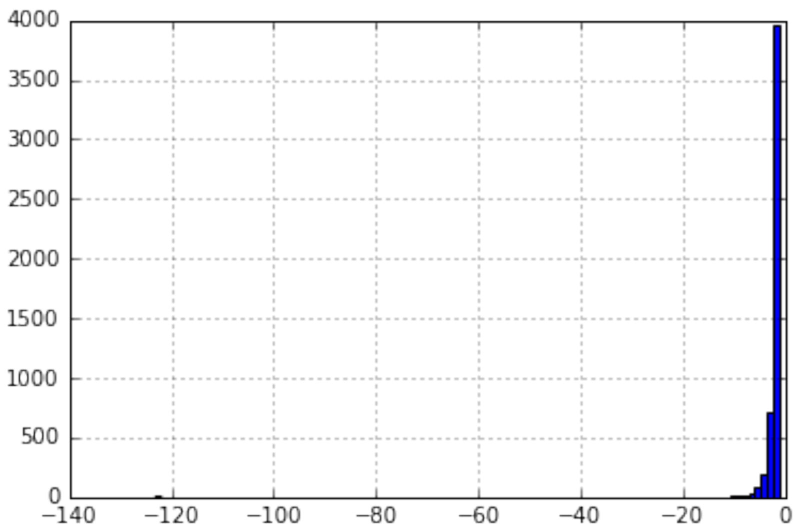
	result	jump_power	diff_days	ind	ss	profit
2015-08-10	0	-2.706233	11	41	12	-0.176608
2015-10-01	0	-2.702703	9	143	12	-0.053364
2015-10-23	1	-2.418221	10	170	12	0.062206
2015-11-16	1	-3.402683	11	249	12	0.113208

2015-11-17	0	-3.153450	10	262	12	-0.032323
2015-11-25	0	-3.251595	12	283	12	-0.128647
2016-01-04	0	-3.497082	10	354	12	-0.013350
2016-04-15	0	-2.571623	9	578	12	-0.007959
2016-05-16	0	-3.490188	10	672	12	-0.017408
2014-10-03	0	-2.728997	10	1016	12	-0.003032
2014-10-08	1	-3.083089	12	1020	12	0.049996
2014-12-05	1	-2.483176	10	1122	12	0.060298
2014-12-30	0	-2.631553	10	1193	12	-0.027207
2015-05-19	0	-2.951475	11	1493	12	-0.044630
2015-06-18	0	-2.629061	9	1580	12	-0.026883
2013-08-27	1	-3.107009	12	1762	12	0.096847
2013-12-02	1	-2.795049	10	1960	12	0.031864
2013-12-11	0	-2.657454	11	1989	12	-0.017341
2014-03-12	0	-3.151759	12	2141	12	-0.030650
2014-05-14	0	-3.117394	11	2277	12	-0.000439
2014-06-23	1	-2.646682	11	2336	12	0.040926
2012-08-24	1	-3.501264	10	2500	12	0.025974
2012-09-20	0	-2.593575	9	2575	12	-0.010300
2012-10-22	0	-2.853089	10	2690	12	-0.010843
2012-12-03	0	-2.946918	11	2829	12	-0.134032
2013-03-14	0	-2.873770	9	3152	12	-0.080194
2013-03-22	0	-2.482930	10	3169	12	-0.026621
2013-05-23	0	-2.795248	9	3342	12	-0.046241
2011-08-30	0	-3.409071	11	3633	12	-0.086524
2011-09-16	0	-2.776829	9	3785	12	-0.172645
2011-10-07	0	-2.599565	9	3838	12	-0.160468
2011-10-28	0	-2.456538	10	3871	12	-0.067081

2011-11-14	1	-3.023274	10	3972	12	0.023094
2011-11-14	0	-2.891566	10	3973	12	-0.095696
2011-11-14	0	-3.050215	10	3974	12	-0.037638
2011-11-22	1	-3.099930	12	4243	12	0.035945
2011-12-23	1	-2.819676	10	4509	12	0.016492
2012-04-02	0	-2.681171	9	4859	12	-0.124722
2012-04-10	1	-2.707060	11	4872	12	0.034586
2012-05-16	0	-3.198386	11	4962	12	-0.010707

`python import scipy.stats as scs import MlFiterBinsCs ZLog.info(scs.normaltest(ump_jump.fiter.df.jump_power)) MlFiterBinsCs.show_orders_hist(ump_jump.fiter.df, s_list = ['jump_power'])`

```
NormaltestResult(statistic=11612.386867948926, pvalue=0.0)
```



```
jump_power show hist and qcuts
(-1.0692, -1.000217]    504
(-1.243, -1.149]        504
(-1.669, -1.499]        504
(-2.304, -1.932]        504
[-123.0916, -3.0491]    504
(-1.149, -1.0692]       503
(-1.366, -1.243]        503
(-1.499, -1.366]        503
(-1.932, -1.669]        503
(-3.0491, -2.304]       503
Name: jump_power, dtype: int64
```

jump power选取llps没有执行最优化求解，因为数据量不够，按照0， 0， 0.65来办

`python llps = ump_jump.cprs[(ump_jump.cprs['lps'] <= 0) & (ump_jump.cprs['lms'] <= 0 ) & (ump_jump.cprs['lrs'] >=0.65) & ( (ump_jump.cprs['lcs'] >= 20) | (ump_jump.cprs['lrs'] == 1) )] llps`

	lcs	lrs	lps	lms
18_2	1.0	1.000000	-0.059441	-0.059441
19_14	1.0	1.000000	-0.059441	-0.059441
20_2	1.0	1.000000	-0.059441	-0.059441
21_12	3.0	1.000000	-0.294978	-0.098326
21_13	1.0	1.000000	-0.059441	-0.059441
21_18	24.0	0.708333	-0.921051	-0.038377
22_2	1.0	1.000000	-0.059441	-0.059441
22_10	1.0	1.000000	-0.101029	-0.101029
23_2	1.0	1.000000	-0.059441	-0.059441
23_13	1.0	1.000000	-0.101029	-0.101029
24_3	1.0	1.000000	-0.059441	-0.059441
24_21	1.0	1.000000	-0.101029	-0.101029
25_15	1.0	1.000000	-0.059441	-0.059441
25_19	1.0	1.000000	-0.101029	-0.101029
25_21	3.0	1.000000	-0.294978	-0.098326
26_2	1.0	1.000000	-0.101029	-0.101029

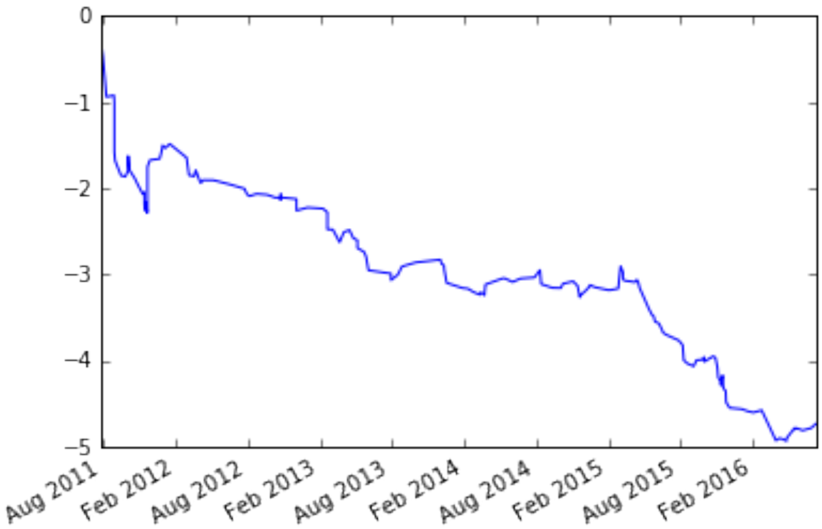
26_12	1.0	1.000000	-0.059441	-0.059441
26_17	3.0	1.000000	-0.294978	-0.098326
27_0	35.0	0.714286	-0.745038	-0.021287
27_3	1.0	1.000000	-0.059441	-0.059441
27_9	1.0	1.000000	-0.101029	-0.101029
27_23	3.0	1.000000	-0.294978	-0.098326
28_2	1.0	1.000000	-0.059441	-0.059441
28_9	1.0	1.000000	-0.101029	-0.101029
28_21	1.0	1.000000	-0.248848	-0.248848
29_15	1.0	1.000000	-0.059441	-0.059441
29_19	1.0	1.000000	-0.101029	-0.101029
29_27	3.0	1.000000	-0.294978	-0.098326
30_3	1.0	1.000000	-0.059441	-0.059441
30_11	1.0	1.000000	-0.101029	-0.101029
...	...	...	...	...
35_28	23.0	0.652174	-0.215931	-0.009388
35_30	1.0	1.000000	-0.248848	-0.248848
36_8	1.0	1.000000	-0.101029	-0.101029
36_16	3.0	1.000000	-0.294978	-0.098326
36_18	1.0	1.000000	-0.059441	-0.059441
37_20	1.0	1.000000	-0.059441	-0.059441
37_21	40.0	0.700000	-1.602485	-0.040062
37_24	1.0	1.000000	-0.101029	-0.101029
37_36	1.0	1.000000	-0.248848	-0.248848
38_0	32.0	0.687500	-1.140307	-0.035635
38_12	1.0	1.000000	-0.028857	-0.028857
38_17	1.0	1.000000	-0.059441	-0.059441
38_23	1.0	1.000000	-0.101029	-0.101029

38_37	1.0	1.000000	-0.248848	-0.248848
39_2	1.0	1.000000	-0.009235	-0.009235
39_5	1.0	1.000000	-0.101029	-0.101029
39_12	40.0	0.700000	-1.052118	-0.026303
39_13	24.0	0.708333	-0.150335	-0.006264
39_18	1.0	1.000000	-0.059441	-0.059441
39_22	1.0	1.000000	-0.248848	-0.248848
39_32	2.0	1.000000	-0.046130	-0.023065
40_16	1.0	1.000000	-0.059441	-0.059441
40_17	1.0	1.000000	-0.248848	-0.248848
40_18	1.0	1.000000	-0.101029	-0.101029
40_19	2.0	1.000000	-0.046130	-0.023065
41_8	29.0	0.724138	-0.679913	-0.023445
41_14	1.0	1.000000	-0.059441	-0.059441
41_21	1.0	1.000000	-0.028857	-0.028857
41_25	1.0	1.000000	-0.101029	-0.101029
41_38	1.0	1.000000	-0.248848	-0.248848

81 rows × 4 columns

`python ump_jump.choose_cprs_component(llps)`

```
nts_pd.shape = (182, 6)
nts_pd loss rate = 0.681318681319
improved rate = 0.0131082423039
predict win rate = 0.512015888779
```

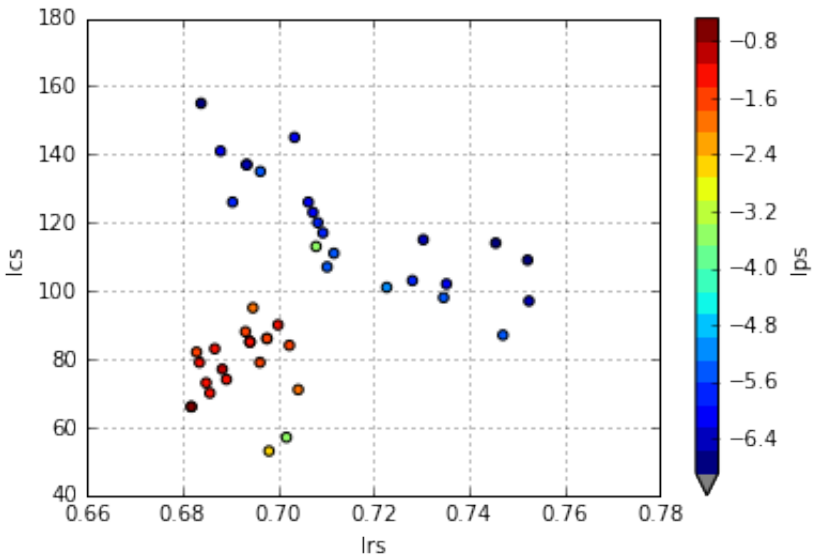


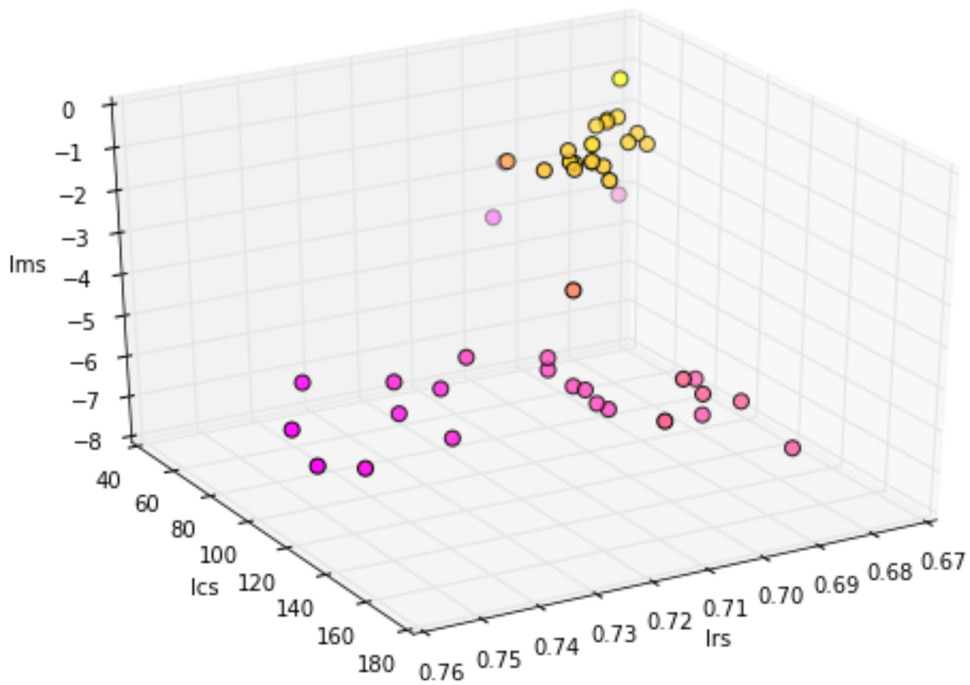
`python ump_jump.dump_clf(llps)`

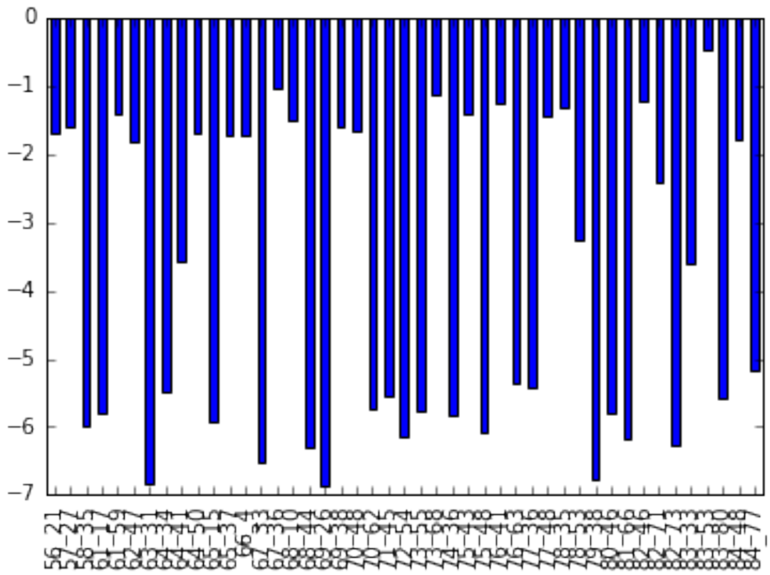
`python from MlFiterDegPd import MlFiterDegPdClass`

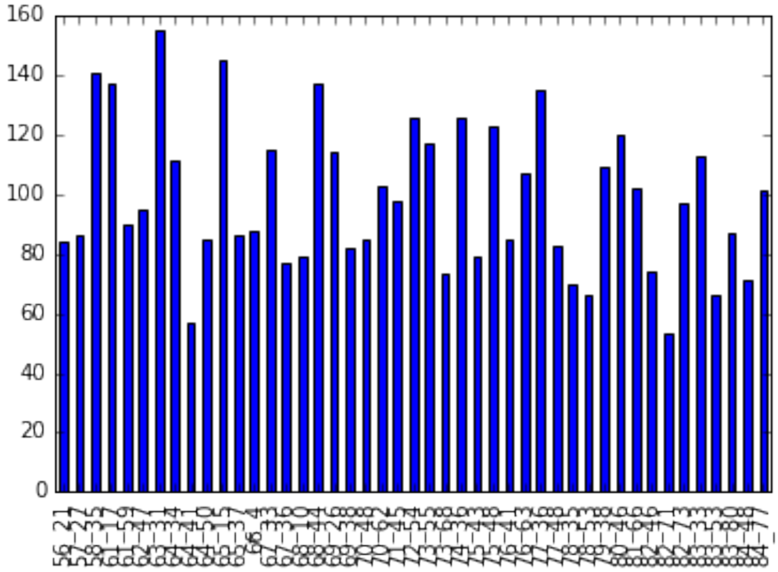
deg ump 主裁，只使用全局最优llps

`python ump_deg = UmpMainClass(orders_pd_ump, MlFiterDegPdClass) ump_deg.gmm_component_filter(p_ncs=np.arange(18, 85), threshold=0.68)`




Ims
Ics
Irs
0
−1
−2
−3
−4
−5
−6
−7
−8
40
60
80
100
120
140
160
180
0.76
0.75
0.74
0.73
0.72
0.71
0.70
0.69
0.68
0.67


0
−1
−2
−3
−4
−5
−6
−7

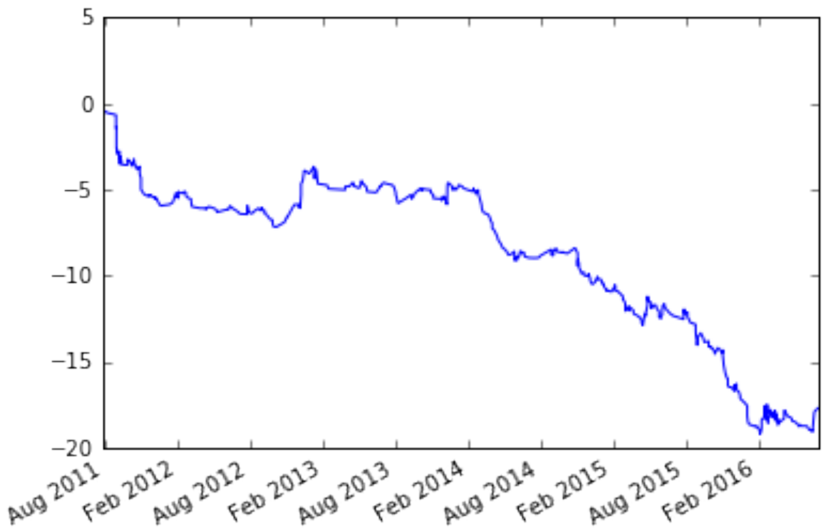


	lcs	lrs	lps	lms
56_21	84.0	0.702381	-1.699820	-0.020236
57_27	86.0	0.697674	-1.591000	-0.018500
58_35	141.0	0.687943	-5.994469	-0.042514
61_17	137.0	0.693431	-5.788397	-0.042251
61_59	90.0	0.700000	-1.401181	-0.015569
62_47	95.0	0.694737	-1.830953	-0.019273
63_31	155.0	0.683871	-6.826563	-0.044042
64_34	111.0	0.711712	-5.487420	-0.049436
64_41	57.0	0.701754	-3.558004	-0.062421
64_50	85.0	0.694118	-1.677678	-0.019737
65_15	145.0	0.703448	-5.934523	-0.040928
65_37	86.0	0.697674	-1.734857	-0.020173
66_4	88.0	0.693182	-1.715398	-0.019493
67_33	115.0	0.730435	-6.528349	-0.056768
67_36	77.0	0.688312	-1.016318	-0.013199
68_10	79.0	0.696203	-1.512461	-0.019145
68_44	137.0	0.693431	-6.297579	-0.045968

69_26	114.0	0.745614	-6.872898	-0.060289
69_38	82.0	0.682927	-1.584954	-0.019329
70_48	85.0	0.694118	-1.657861	-0.019504
70_62	103.0	0.728155	-5.746673	-0.055793
71_45	98.0	0.734694	-5.554473	-0.056678
72_54	126.0	0.706349	-6.144978	-0.048770
73_55	117.0	0.709402	-5.770077	-0.049317
73_68	73.0	0.684932	-1.130658	-0.015488
74_36	126.0	0.690476	-5.826468	-0.046242
75_43	79.0	0.683544	-1.393574	-0.017640
75_48	123.0	0.707317	-6.067599	-0.049330
76_41	85.0	0.694118	-1.257220	-0.014791
76_63	107.0	0.710280	-5.348693	-0.049988
77_36	135.0	0.696296	-5.407719	-0.040057
77_48	83.0	0.686747	-1.429252	-0.017220
78_35	70.0	0.685714	-1.308801	-0.018697
78_53	66.0	0.681818	-3.242988	-0.049136
79_38	109.0	0.752294	-6.785169	-0.062249
80_46	120.0	0.708333	-5.798165	-0.048318
81_66	102.0	0.735294	-6.190534	-0.060692
82_46	74.0	0.689189	-1.214181	-0.016408
82_71	53.0	0.698113	-2.419266	-0.045647
82_73	97.0	0.752577	-6.261241	-0.064549
83_33	113.0	0.707965	-3.611449	-0.031960
83_53	66.0	0.681818	-0.477639	-0.007237
83_80	87.0	0.747126	-5.564710	-0.063962
84_48	71.0	0.704225	-1.782087	-0.025100
84_77	101.0	0.722772	-5.183489	-0.051322

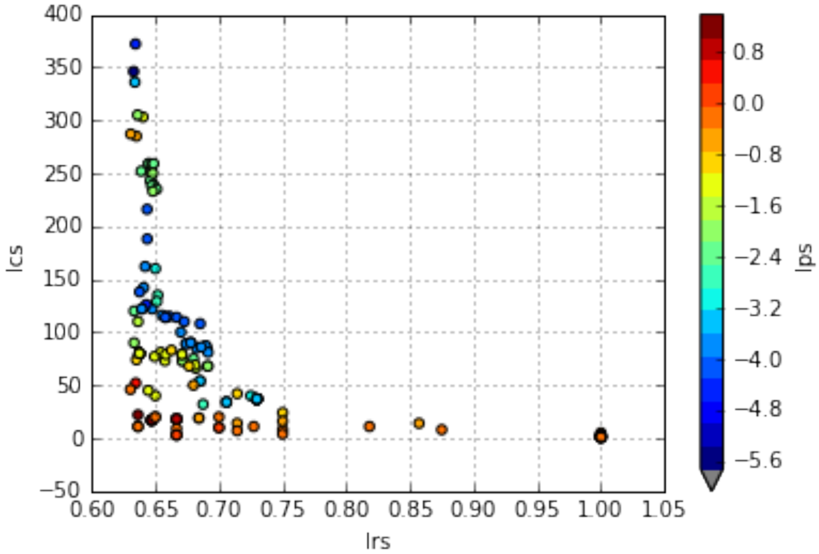
`python brust_min = ump_deg.brust_min() llps = ump_deg.cprs[(ump_deg.cprs['lps'] <= brust_min[0]) & (ump_deg.cprs['lms'] <= brust_min[1] ) & (ump_deg.cprs['lrs'] >= brust_min[2])] ump_deg.choose_cprs_component(llps) ump_deg.dump_clf(llps)`

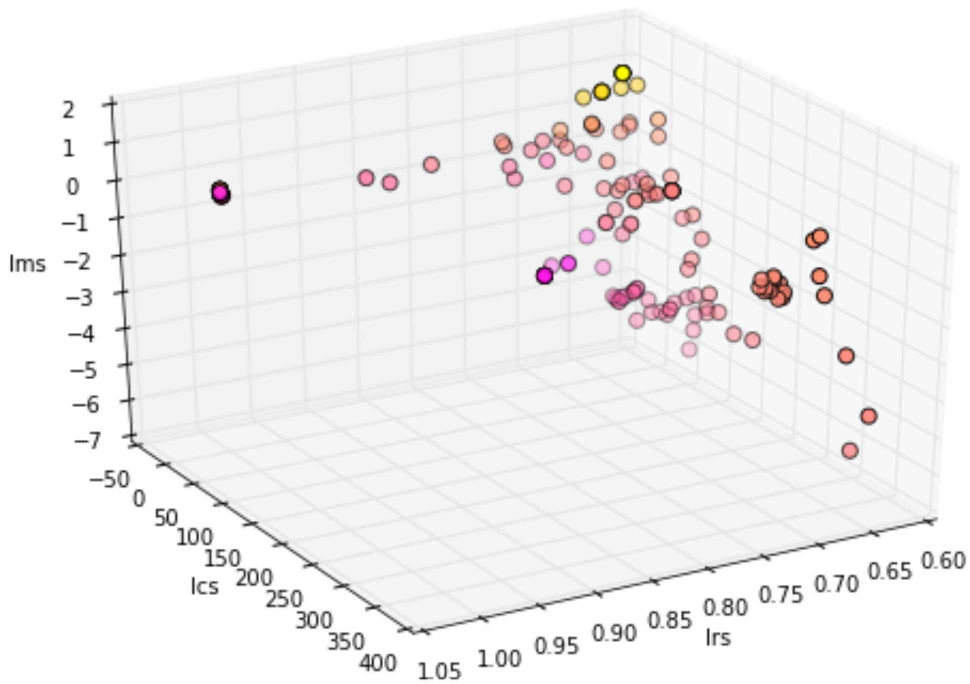
```
nts_pd.shape = (513, 7)
nts_pd loss rate = 0.690058479532
improved rate = 0.00434240413308
predict win rate = 0.505099541264
```

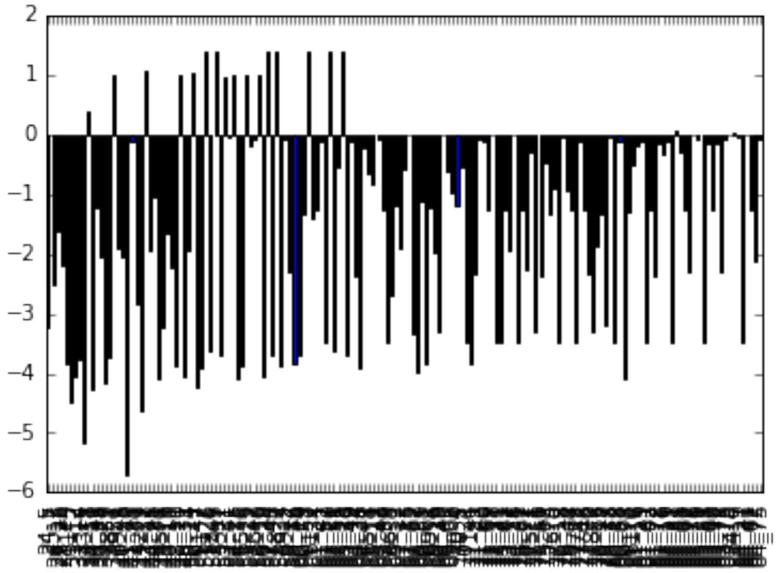


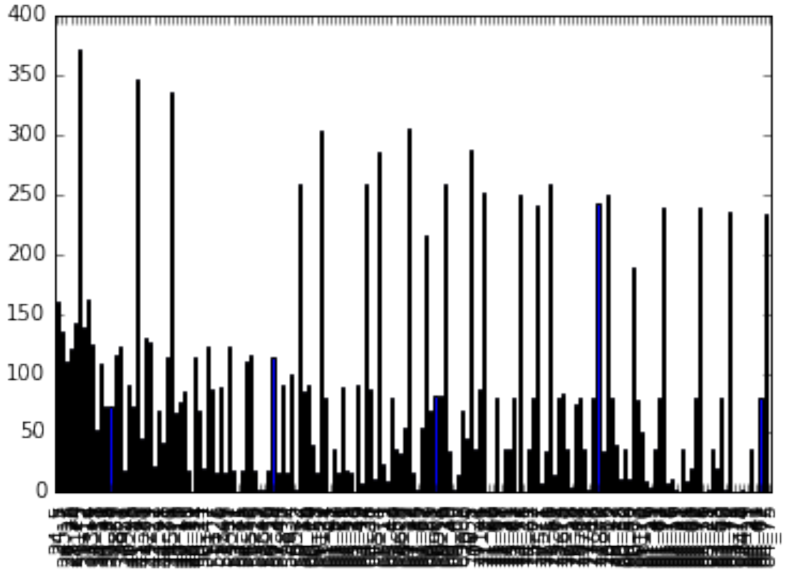
wave ump第二主裁

`python from MlFiterWavePd import MlFiterWavePdClass ump_wave = UmpMainClass(orders_pd_ump, MlFiterWavePdClass) ump_wave.gmm_component_filter(p_ncs=np.arange(18, 85), threshold=0.63)`


400
350
300
250
200
150
100
50
0
−50
lcs
0.60
0.65
0.70
0.75
0.80
0.85
0.90
0.95
1.00
1.05
lrs
0.8
0.0
−0.8
−1.6
−2.4
−3.2
−4.0
−4.8
−5.6
lps


2
1
0
−1
−2
−3
−4
−5
−6
−7
lms
−50
0
50
100
150
200
250
300
350
400
lcs
1.05
1.00
0.95
0.90
0.85
0.80
0.75
0.70
0.65
0.60
lrs





	lcs	lrs	lps	lms
24_5	160.0	0.650000	-3.223928	-0.020150
25_7	135.0	0.651852	-2.515814	-0.018636
26_15	110.0	0.636364	-1.624467	-0.014768
27_26	120.0	0.633333	-2.197939	-0.018316
28_24	142.0	0.640845	-3.833662	-0.026998
31_5	372.0	0.634409	-4.477736	-0.012037
31_12	138.0	0.637681	-4.040783	-0.029281

32_14	162.0	0.641975	-3.757361	-0.023194
33_5	124.0	0.645161	-5.155671	-0.041578
33_18	52.0	0.634615	0.401146	0.007714
34_11	108.0	0.685185	-4.270846	-0.039545
35_23	73.0	0.657534	-1.210312	-0.016580
36_28	73.0	0.671233	-2.032845	-0.027847
37_25	115.0	0.660870	-4.175807	-0.036311
38_7	122.0	0.647541	-3.728374	-0.030560
38_21	18.0	0.666667	1.009932	0.056107
39_9	90.0	0.633333	-1.914981	-0.021278
40_25	73.0	0.671233	-2.032845	-0.027847
41_10	346.0	0.632948	-5.712077	-0.016509
41_23	46.0	0.630435	-0.125099	-0.002720
41_31	129.0	0.651163	-2.841463	-0.022027
42_9	126.0	0.642857	-4.609681	-0.036585
42_22	22.0	0.636364	1.070885	0.048677
43_11	68.0	0.691176	-1.932722	-0.028422
44_28	42.0	0.714286	-1.051805	-0.025043
44_31	114.0	0.666667	-4.074704	-0.035743
45_6	336.0	0.633929	-3.224770	-0.009598
45_13	66.0	0.681818	-1.651001	-0.025015
46_10	75.0	0.680000	-2.241460	-0.029886
47_11	84.0	0.690476	-3.868414	-0.046053
...	...	...	...	...
81_0	50.0	0.680000	-0.499477	-0.009990
81_16	10.0	0.700000	-0.190936	-0.019094
81_19	5.0	1.000000	-0.117092	-0.023418
81_21	37.0	0.729730	-3.473791	-0.093886

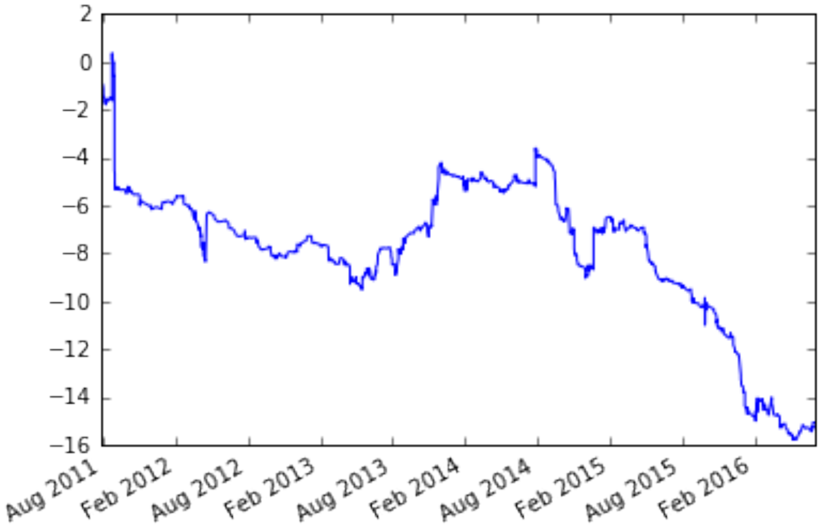
81_62	80.0	0.637500	-1.248253	-0.015603
81_72	239.0	0.648536	-2.384517	-0.009977
81_78	7.0	0.714286	-0.162419	-0.023203
82_16	11.0	0.727273	-0.310766	-0.028251
82_19	3.0	1.000000	-0.095641	-0.031880
82_21	37.0	0.729730	-3.473791	-0.093886
82_31	10.0	0.700000	0.076625	0.007663
82_58	20.0	0.700000	-0.292118	-0.014606
82_62	80.0	0.637500	-1.248253	-0.015603
82_72	238.0	0.647059	-2.299922	-0.009664
82_78	1.0	1.000000	-0.015716	-0.015716
83_19	2.0	1.000000	-0.069436	-0.034718
83_21	37.0	0.729730	-3.473791	-0.093886
83_58	20.0	0.650000	-0.132397	-0.006620
83_62	80.0	0.637500	-1.248253	-0.015603
83_70	3.0	1.000000	-0.135101	-0.045034
83_72	235.0	0.651064	-2.291549	-0.009751
83_75	1.0	1.000000	-0.086957	-0.086957
83_78	1.0	1.000000	-0.015716	-0.015716
84_4	3.0	0.666667	0.029706	0.009902
84_19	1.0	1.000000	-0.042480	-0.042480
84_21	37.0	0.729730	-3.473791	-0.093886
84_31	1.0	1.000000	-0.004204	-0.004204
84_62	80.0	0.637500	-1.248253	-0.015603
84_72	233.0	0.648069	-2.120741	-0.009102
84_75	1.0	1.000000	-0.086957	-0.086957

163 rows × 4 columns

`python brust_min = ump_wave.brust_min() llps =`

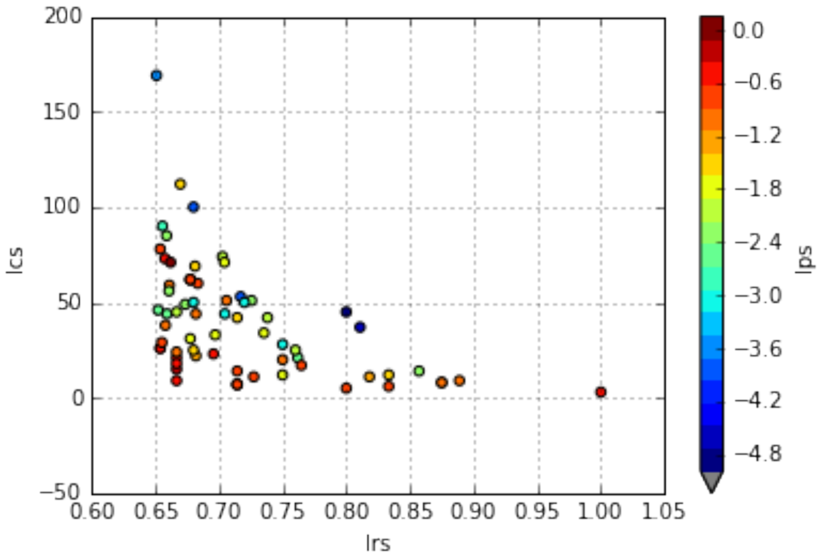
`ump_wave.cprs[(ump_wave.cprs['lps'] <= brust_min[0]) & (ump_wave.cprs['lms'] <= brust_min[1] ) & (ump_wave.cprs['lrs'] >= brust_min[2])] ump_wave.choose_cprs_component(llps) ump_wave.dump_clf(llps)`

```
nts_pd.shape = (1126, 7)
nts_pd loss rate = 0.639431616341
improved rate = 0.00699238409121
predict win rate = 0.507749521222
```

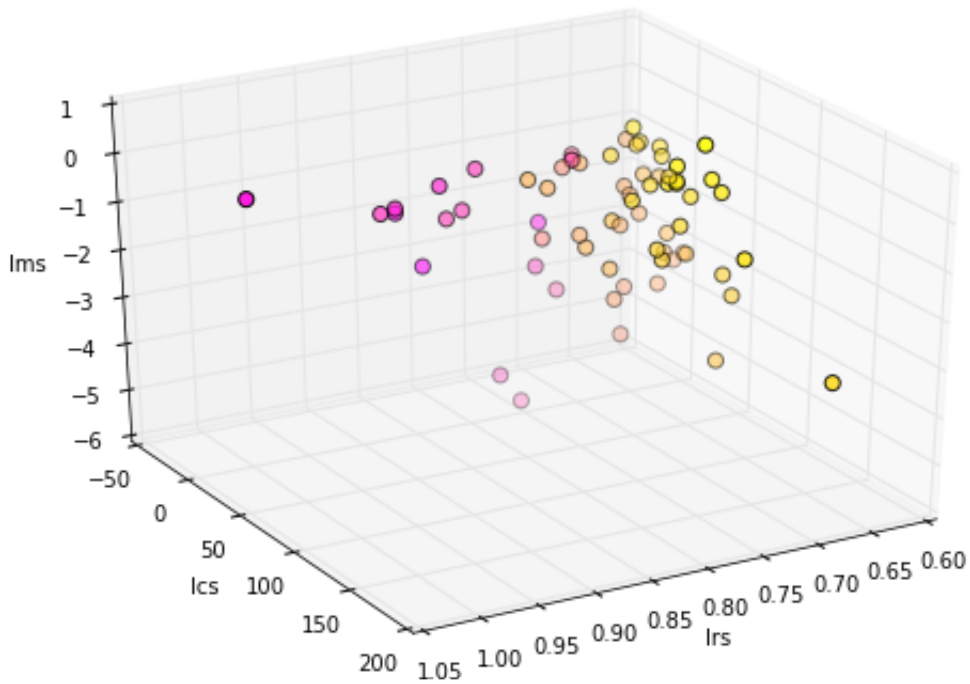


第三主裁

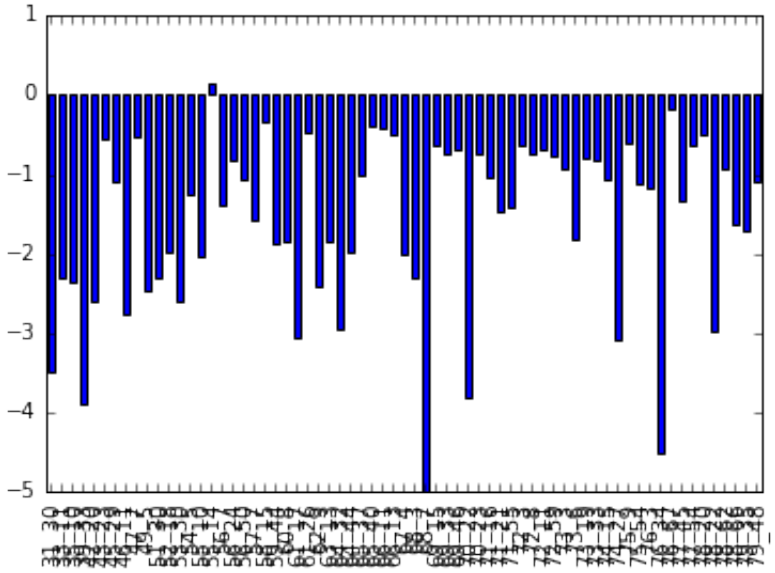
` python from MlFiterMainPd import MlFiterMainPdClass ump_main = UmpMainClass(orders_pd_ump, MlFiterMainPdClass) ump_main.gmm_component_filter(p_ncs=np.arange(18, 80), threshold=0.65)`

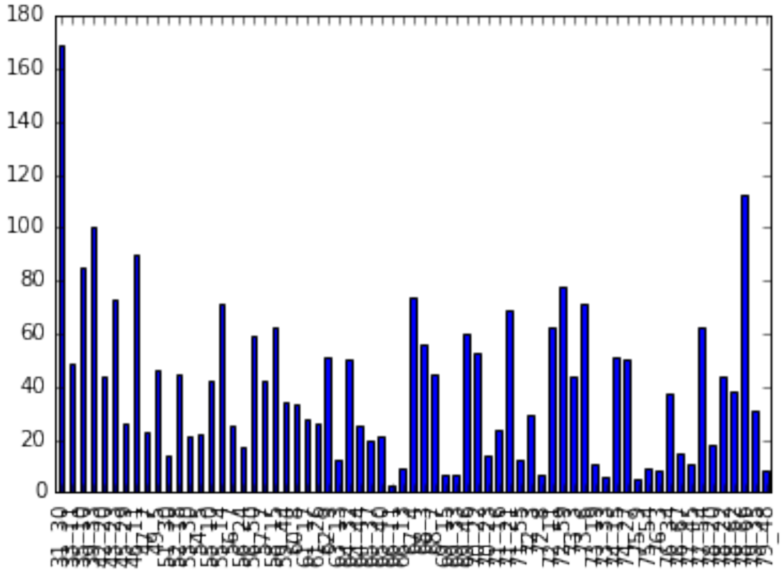


lcs
lrs
lps



lms
lcs
lrs





	lcs	lrs	lps	lms
31_30	169.0	0.650888	-3.488626	-0.020643
33_11	49.0	0.673469	-2.314873	-0.047242
35_10	85.0	0.658824	-2.364326	-0.027816
39_33	100.0	0.680000	-3.877226	-0.038772
42_20	44.0	0.659091	-2.604120	-0.059185
43_23	73.0	0.657534	-0.548995	-0.007520
45_29	26.0	0.653846	-1.105545	-0.042521

46_11	90.0	0.655556	-2.752587	-0.030584
47_7	23.0	0.695652	-0.536203	-0.023313
49_5	46.0	0.652174	-2.461245	-0.053505
51_30	14.0	0.857143	-2.291159	-0.163654
52_18	45.0	0.666667	-1.992503	-0.044278
53_30	21.0	0.761905	-2.606842	-0.124135
54_5	22.0	0.681818	-1.242849	-0.056493
55_10	42.0	0.738095	-2.028529	-0.048298
55_14	71.0	0.661972	0.153408	0.002161
56_7	25.0	0.680000	-1.394074	-0.055763
56_24	17.0	0.764706	-0.833099	-0.049006
56_50	59.0	0.661017	-1.069024	-0.018119
57_7	42.0	0.714286	-1.574196	-0.037481
58_15	62.0	0.677419	-0.331972	-0.005354
59_44	34.0	0.735294	-1.886792	-0.055494
60_8	33.0	0.696970	-1.858589	-0.056321
61_17	28.0	0.750000	-3.059575	-0.109271
61_26	26.0	0.653846	-0.486007	-0.018693
62_9	51.0	0.725490	-2.412638	-0.047307
63_13	12.0	0.750000	-1.847714	-0.153976
64_32	50.0	0.680000	-2.936657	-0.058733
64_44	25.0	0.760000	-1.977574	-0.079103
65_37	20.0	0.750000	-1.026872	-0.051344
...	...	...	...	...
69_33	7.0	0.714286	-0.757312	-0.108187
69_46	60.0	0.683333	-0.692344	-0.011539
70_22	53.0	0.716981	-3.815061	-0.071982
70_23	14.0	0.714286	-0.735310	-0.052522

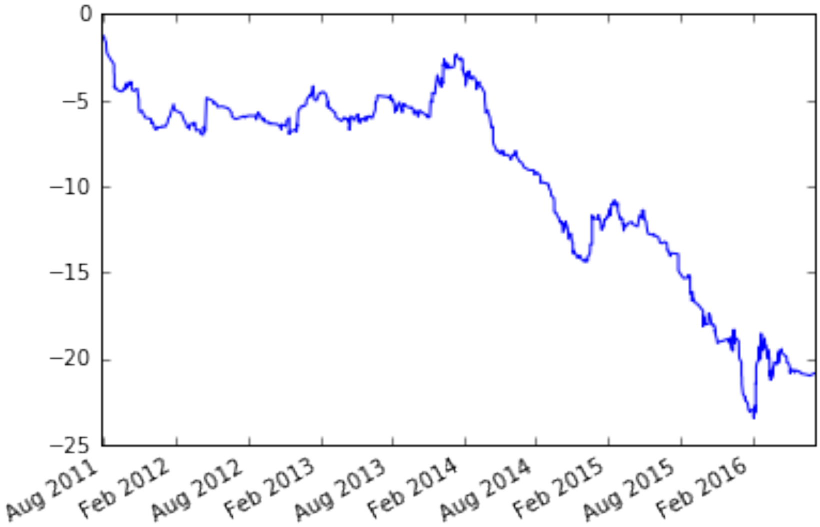
71_16	24.0	0.666667	-1.032935	-0.043039
71_21	69.0	0.681159	-1.465262	-0.021236
71_55	12.0	0.833333	-1.425944	-0.118829
72_3	29.0	0.655172	-0.642314	-0.022149
72_8	7.0	0.714286	-0.757312	-0.108187
72_11	62.0	0.677419	-0.690655	-0.011140
72_59	78.0	0.653846	-0.769406	-0.009864
73_3	44.0	0.681818	-0.936858	-0.021292
73_6	71.0	0.704225	-1.829152	-0.025763
73_19	11.0	0.727273	-0.811367	-0.073761
73_33	6.0	0.833333	-0.820737	-0.136790
74_15	51.0	0.705882	-1.070951	-0.020999
74_27	50.0	0.720000	-3.096087	-0.061922
75_9	5.0	0.800000	-0.617400	-0.123480
75_54	9.0	0.888889	-1.126203	-0.125134
76_3	8.0	0.875000	-1.182426	-0.147803
76_34	37.0	0.810811	-4.498345	-0.121577
76_67	15.0	0.666667	-0.173521	-0.011568
77_45	11.0	0.818182	-1.324581	-0.120416
77_54	62.0	0.677419	-0.645419	-0.010410
78_20	18.0	0.666667	-0.495147	-0.027508
78_22	44.0	0.704545	-2.978029	-0.067682
78_62	38.0	0.657895	-0.941355	-0.024773
78_66	112.0	0.669643	-1.630110	-0.014555
79_35	31.0	0.677419	-1.715791	-0.055348
79_48	8.0	0.875000	-1.089348	-0.136168

67 rows × 4 columns

`python brust_min = ump_main.brust_min() llps =`

```
ump_main.cprs[(ump_main.cprs['lps'] <= brust_min[0]) &
(ump_main.cprs['lms'] <= brust_min[1] ) & (ump_main.cprs['lrs'] >=
brust_min[2])] ump_main.choose_cprs_component(llps)
ump_main.dump_clf(llps)
```

```
nts_pd.shape = (804, 7)
nts_pd loss rate = 0.661691542289
improved rate = 0.00578987217744
predict win rate = 0.506547009308
```




```python
```