

# LST勉強会 第2回

NPBDAAの起動

# Step1 google ドライブからDAAをダウンロード



ドライブを検索



ドライブ

マイ... > 創発... > 勉強会 > 2015... > LST勉強会



新規

マイドライブ

共有アイテム

Google フォト

最近使用したアイテム

スター付き

ゴミ箱

名前 ↑

オーナー

最終更新



第一回

中島諒

2015/07/20



NPBDAA.zip



中島諒

2015/07/17



PYHSMM.zip



中島諒

2015/07/17

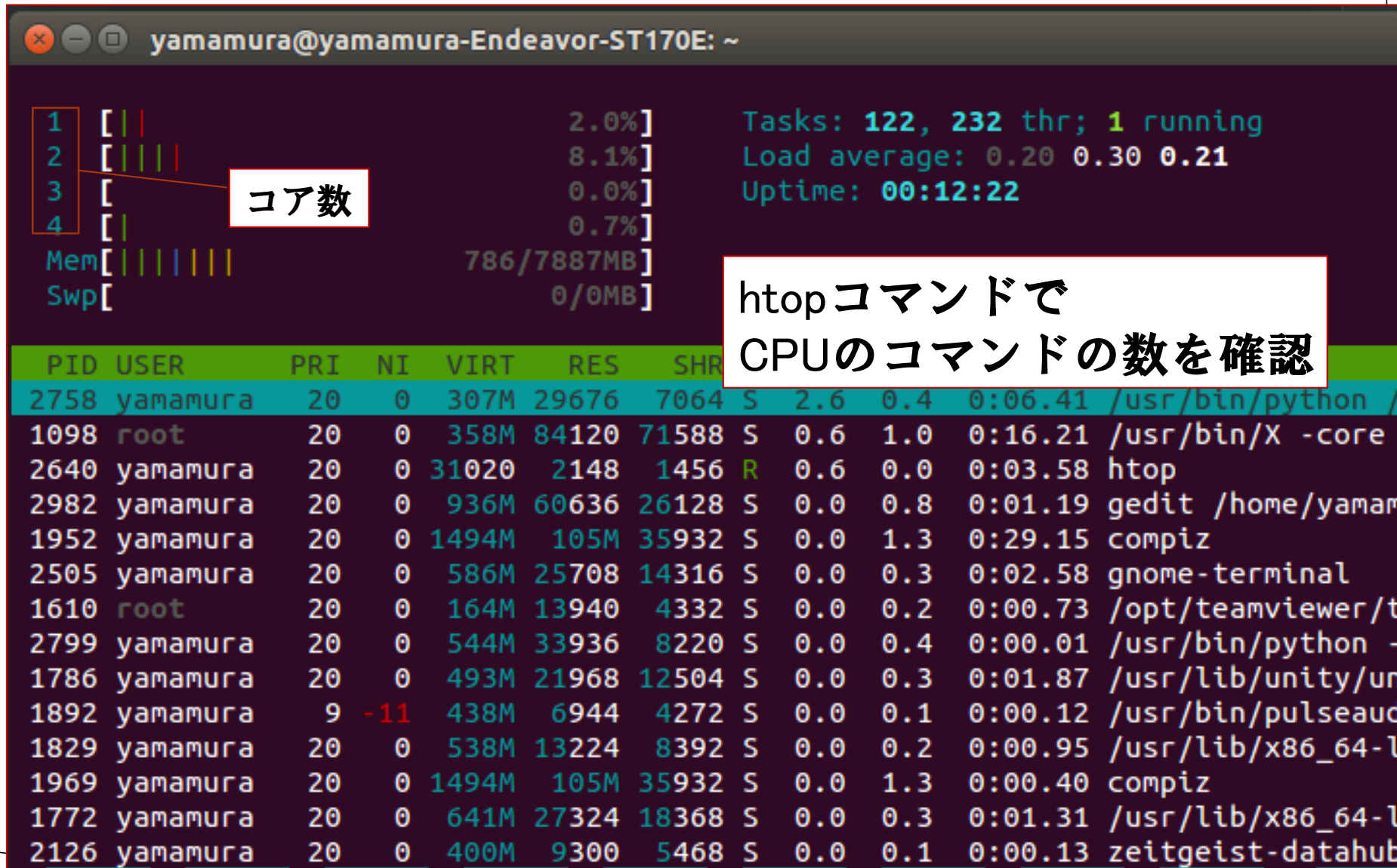
このフォルダの中身を  
作業フォルダに移動

## Step2 カレントディレクトリの移動

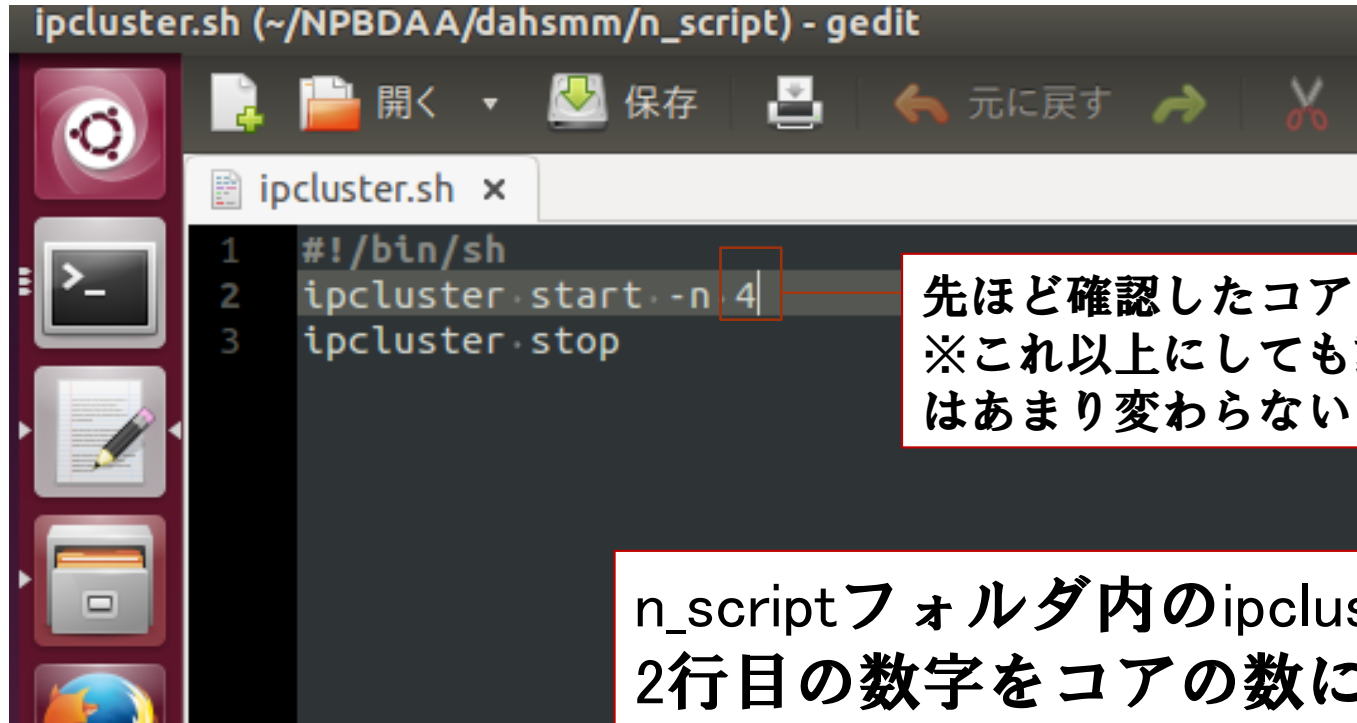
```
yamamura@yamamura-Endeavor-ST170E: ~/NPBDAA/dahsmm/n_script  
yamamura@yamamura-Endeavor-ST170E:~$ cd NPBDAA/dahsmm/n_script/  
yamamura@yamamura-Endeavor-ST170E:~/NPBDAA/dahsmm/n_script$
```



# Step3 コアの設定



# Step3 コアの数の設定



```
ipcluster.sh (~/NPBDAA/dahsmm/n_script) - gedit
1  #!/bin/sh
2  ipcluster.start -n 4
3  ipcluster.stop
```

先ほど確認したコアの数に書き換える  
※これ以上にしても動くがパフォーマンスはあまり変わらない

n\_scriptフォルダ内のipcluster.shを開き  
2行目の数字をコアの数に書き換える

# Step3 コアの数の設定

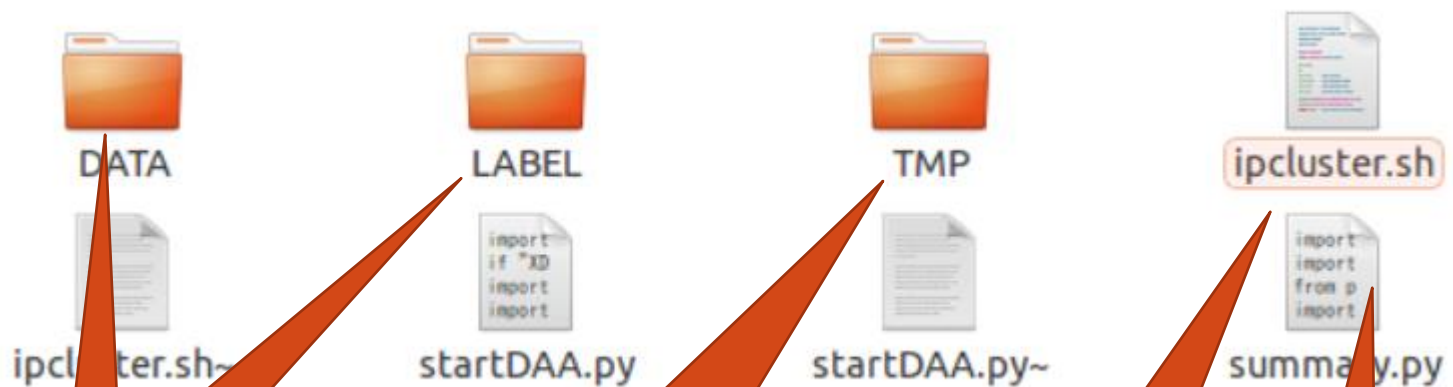
```
yamamura@yamamura-Endeavor-ST170E:~/NPBDAA/dahsmm/n_script$ sh ipcluster.sh
2015-07-29 15:37:22.793 [IPClusterStart] Using existing profile dir: u'/home/yamamura/.config/ipython/profile_default'
2015-07-29 15:37:22.842 [IPClusterStart] Removing pid file: /home/yamamura/.config/ipython/profile_default/pid/ipcluster.pid
2015-07-29 15:37:22.842 [IPClusterStart] Starting ipcluster with [daemon=False]
2015-07-29 15:37:22.843 [IPClusterStart] Creating pid file: /home/yamamura/.config/ipython/profile_default/pid/ipcluster.pid
2015-07-29 15:37:22.843 [IPClusterStart] Starting Controller with LocalControllerLauncher
2015-07-29 15:37:23.843 [IPClusterStart] Starting 4 Engines with LocalEngineSetLauncher
```

sh コマンドでipcluster.shを起動

```
1  [|||||] 38.6%
2  [|||||] 38.7%
3  [|||||] 40.0%
4  [|||||] 34.9%
Mem[|||||] 728/7887MB
Swp[|||||] 0/0MB
```

この時コアが全て動いていたら  
OK

# Step3.5 NPBDAAの起動に必要なファイル



真ラベルデータ  
フォルダ (60デー  
タ)

音声  
MFCC→DSAE3次  
元データ (60デー  
タ)

データ読み込み再利用  
用オブジェクト保存  
フォルダ

NPB-DAA実行スクリプ  
ト

ipcluster用 (並列  
処理)

結果図, 単語リストなど出  
力ファイル生成スクリプト

# Step4 NPBDAAの起動

```
startDAA.py x
19 LETTER_N
20 WORD_N
21 DATA_N
22 model_hypparams
23 word_model_params
24 obs_hypparams
25 dur_hypparams
26 filename''.split('\n')
27 #-----multi-process-function-----
28 def multi_dump_object(data,model,f,count):
29     ...print f, ".dumping..."
30     ...fp=open("TMP/"+f+".dump", 'w')
31     ...pickle.dump(HSMMState(data,model), fp)
32     ...fp.close()
33     ...count.value+=count.value+1
34     ...print f, 'dump finish-->count:', count.v
35 #-----main-----
36 def main(result_dir=None):
37     ...if result_dir==None:
38     ...result_dir=get_result_dir(__file_
39     ...os.mkdir(result_dir)
40     ...param_path=os.path.join(result_dir, 'parameter.json')
41     ...fig_title_path=os.path.join(result_dir, 'fig_title.json')
42 #initialize model.params#
43 ...#O(T*L_max*W_max^2*d_max^3)
44 ...ITER_N=5
45 ...LETTER_N=7
46 ...WORD_N=7
47 ...DATA_N=60
48 ...obs_dim=3
49 ...model_hypparams={'state_dim':WORD_N, 'alpha':10.0, 'gamma':10.0}
50 ...word_model_params={'letter_type':LETTER_N, 'rho':10}
51 ...obs_hypparams={'mu_0':np.zeros(obs_dim), 'sigma_0':np.eye(obs_dim), 'kappa_0':0.01,
52 ...dur_hypparams={'alpha_0':50.0, 'beta_0':10.0}
53 ...length_dist=pyhsmm.distributions.PoissonDuration(alpha_0=30, beta_0=10, lambda=3)
54 #setting#
```

StartDAA.pyを開き

44行目のITER\_Nの数字を100から5に変更  
(ITER\_Nはイテレーションの回数

今回は試験的に起動するため小さく設定)



# Step4 NPBDAAの起動

```
yamamura@yamamura-Endeavor-ST170E: ~/NPBDAA/dahsmm/n_script
[(2, (0, 10)), (2, (10, 20))]
[(2, (0, 8)), (1, (8, 20))]
[(2, (0, 10)), (1, (10, 21)), (0, (21, 30))]
[(2, (0, 10)), (2, (10, 20)), (0, (20, 30))]
[(2, (0, 10)), (0, (10, 19))]
[(2, (0, 10)), (0, (10, 20))]
[(4, (0, 27)), (2, (27, 32))]
[(4, (0, 26)), (1, (26, 32))]
[(0, (0, 21))]
[(0, (0, 21))]
[(6, (0, 25))]
[(6, (0, 24))]
[(6, (0, 28)), (1, (28, 36))]
[(0, (0, 11)), (6, (11, 26)), (1, (26, 36))]
[(6, (0, 20))]
[(6, (0, 20))]
[(0, (0, 9)), (0, (9, 21))]
[(0, (0, 20))]
.
22.12sec avg, 110.62sec total
-----estimation process completed!!-----
yamamura@yamamura-Endeavor-ST170E: ~/NPBDAA/dahsmm/n_script$
```

N\_scriptフォルダ内の  
startDAA.pyを起動  
左図のような画面が出ればOK

結果はstartDAA\_result\_N に保存  
(Nは数字)

データあたりの推定された単語境界  
の表示 (21フレームのうち, 0~9  
に1単語, 9~21に一単語)

トータルの時間と  
1イテレーションごとの時間

※Keyerror がでた場合はもう一度起動しなおすとうまくいく

# Step4. 5 出力ファイル説明

["aioi\_aioi", "aioi\_aioi2", "aioi\_ao", ..., "uo\_ie",  
"uo\_ie2", "uo\_uo", "uo\_uo2"]

データ名と順番を保存するファイル

モデルの尤度をイ  
テレーション毎に  
保存するファイル

単語境界をオブジェクト  
保存するファイル（機械  
言語でかかれていますので  
見てもわからない）

["LETTER\_N": 7,  
"model\_hypparams":  
{ "state\_dim": 7, "gamma":  
10.0, "alpha": 10.0},  
"ITER\_N": 100, "DATA\_N":  
60, ..., "word\_model\_params":  
{ "letter\_type": 7, "rho": 10}]  
設定したパラメータ,  
実験条件などを保存し  
ておくファイル

sample\_letters\_\*.txt,  
sample\_states\_\*.txtは\*番目の  
データにおいてイテレー  
ション毎に推定されたフ  
レームごとの音素,単語ラベ  
ルが保存されている

fig_title.json	sample_letters_28.txt	sample_states_0.txt	sample_states_32.txt	state_ranges_3.txt	state_ranges_35.txt
loglikelihood.txt	sample_letters_29.txt	sample_states_1.txt	sample_states_33.txt	state_ranges_4.txt	state_ranges_36.txt
parameter.json	sample_letters_30.txt	sample_states_2.txt	sample_states_34.txt	state_ranges_5.txt	state_ranges_37.txt
sample_letters_0.txt	sample_letters_31.txt	sample_states_3.txt	sample_states_35.txt	state_ranges_6.txt	state_ranges_38.txt
sample_letters_1.txt	sample_letters_32.txt	sample_states_4.txt	sample_states_36.txt	state_ranges_7.txt	state_ranges_39.txt
sample_letters_2.txt	sample_letters_33.txt	sample_states_5.txt	sample_states_37.txt	state_ranges_8.txt	state_ranges_40.txt
sample_letters_3.txt	sample_letters_34.txt	sample_states_6.txt	sample_states_38.txt	state_ranges_9.txt	state_ranges_41.txt
sample_letters_4.txt	sample_letters_35.txt	sample_states_7.txt	sample_states_39.txt	state_ranges_10.txt	state_ranges_42.txt
sample_letters_5.txt	sample_letters_36.txt	sample_states_8.txt	sample_states_40.txt	state_ranges_11.txt	state_ranges_43.txt
sample_letters_6.txt	sample_letters_37.txt	sample_states_9.txt	sample_states_41.txt	state_ranges_12.txt	state_ranges_44.txt
sample_letters_7.txt	sample_letters_38.txt	sample_states_10.txt	sample_states_42.txt	state_ranges_13.txt	state_ranges_45.txt
sample_letters_8.txt	sample_letters_39.txt	sample_states_11.txt	sample_states_43.txt	state_ranges_14.txt	state_ranges_46.txt
sample_letters_9.txt	sample_letters_40.txt	sample_states_12.txt	sample_states_44.txt	state_ranges_15.txt	state_ranges_47.txt
sample_letters_10.txt	sample_letters_41.txt	sample_states_13.txt	sample_states_45.txt	state_ranges_16.txt	state_ranges_48.txt
sample_letters_11.txt	sample_letters_42.txt	sample_states_14.txt	sample_states_46.txt	state_ranges_17.txt	state_ranges_49.txt
sample_letters_12.txt	sample_letters_43.txt	sample_states_15.txt	sample_states_47.txt	state_ranges_18.txt	state_ranges_50.txt
sample_letters_13.txt	sample_letters_44.txt	sample_states_16.txt	sample_states_48.txt	state_ranges_19.txt	state_ranges_51.txt
sample_letters_14.txt	sample_letters_45.txt	sample_states_17.txt	sample_states_49.txt	state_ranges_20.txt	state_ranges_52.txt
sample_letters_15.txt	sample_letters_46.txt	sample_states_18.txt	sample_states_50.txt	state_ranges_21.txt	state_ranges_53.txt
sample_letters_16.txt	sample_letters_47.txt	sample_states_19.txt	sample_states_51.txt	state_ranges_22.txt	state_ranges_54.txt
sample_letters_17.txt	sample_letters_48.txt	sample_states_20.txt	sample_states_52.txt	state_ranges_23.txt	state_ranges_55.txt
sample_letters_18.txt	sample_letters_49.txt	sample_states_21.txt	sample_states_53.txt	state_ranges_24.txt	state_ranges_56.txt
sample_letters_19.txt	sample_letters_50.txt	sample_states_22.txt	sample_states_54.txt	state_ranges_25.txt	state_ranges_57.txt
sample_letters_20.txt	sample_letters_51.txt	sample_states_23.txt	sample_states_55.txt	state_ranges_26.txt	state_ranges_58.txt
sample_letters_21.txt	sample_letters_52.txt	sample_states_24.txt	sample_states_56.txt	state_ranges_27.txt	state_ranges_59.txt
sample_letters_22.txt	sample_letters_53.txt	sample_states_25.txt	sample_states_57.txt	state_ranges_28.txt	
sample_letters_23.txt	sample_letters_54.txt	sample_states_26.txt	sample_states_58.txt	state_ranges_29.txt	
sample_letters_24.txt	sample_letters_55.txt	sample_states_27.txt	sample_states_59.txt	state_ranges_30.txt	
sample_letters_25.txt	sample_letters_56.txt	sample_states_28.txt	sample_word_list.txt	state_ranges_31.txt	
sample_letters_26.txt	sample_letters_57.txt	sample_states_29.txt	state_ranges_0.txt	state_ranges_32.txt	
sample_letters_27.txt	sample_letters_58.txt	sample_states_30.txt	state_ranges_1.txt	state_ranges_33.txt	
	sample_letters_59.txt	sample_states_31.txt	state_ranges_2.txt	state_ranges_34.txt	

## Step4. 5 出力ファイル説明

["aioi\_aioi", "aioi\_aioi2", "aioi\_ao", ..., "uo\_ie",  
"uo\_ie2", "uo\_uo", "uo\_uo2"]

## データ名と順番を保存するファイル

モデルの尤度をイ  
テレーション毎に  
保存するファイル

単語境界をオブジェクト  
保存するファイル（機械  
言語でかかっているので  
見てもわからない）

```
{ "LETTER_N": 7,  
  "model_hypparams":  
    { "state_dim": 7, "gamma":  
      10.0, "alpha": 10.0},  
  "ITER_N": 100, "DATA_N":  
60,..., "word_model_params":  
[ "letter_type": 7, "rho": 10] }
```

設定したパラメータ,  
実験条件などを保存し  
ておくファイル

sample\_letters\_\*.txt.  
sample\_states\_\*.txtは\*番目の  
データにおいてイテレー  
ション毎に推定されたフ  
レームごとの音素,単語ラベ  
ルが保存されている

[illegible][illegible]

## Step5 結果を図示する

startDAA\_result\_のフォルダに移動し  
summary.pyを起動(summary.pyはn\_scriptフォルダにあるので注意)

```
yamamura@yamamura-Endeavor-ST170E:~/NPBDAA/dahsmm/n_script$ cd startDAA_result_  
startDAA_result_1/ startDAA_result_2/  
yamamura@yamamura-Endeavor-ST170E:~/NPBDAA/dahsmm/n_script$ cd startDAA_result_  
startDAA_result_1/ startDAA_result_2/  
yamamura@yamamura-Endeavor-ST170E:~/NPBDAA/dahsmm/n_script$ cd startDAA_result_2  
yamamura@yamamura-Endeavor-ST170E:~/NPBDAA/dahsmm/n_script/startDAA_result_2$ ipython ../sum
```

一番数字が  
新しいフォルダを選択

summary.pyを起動

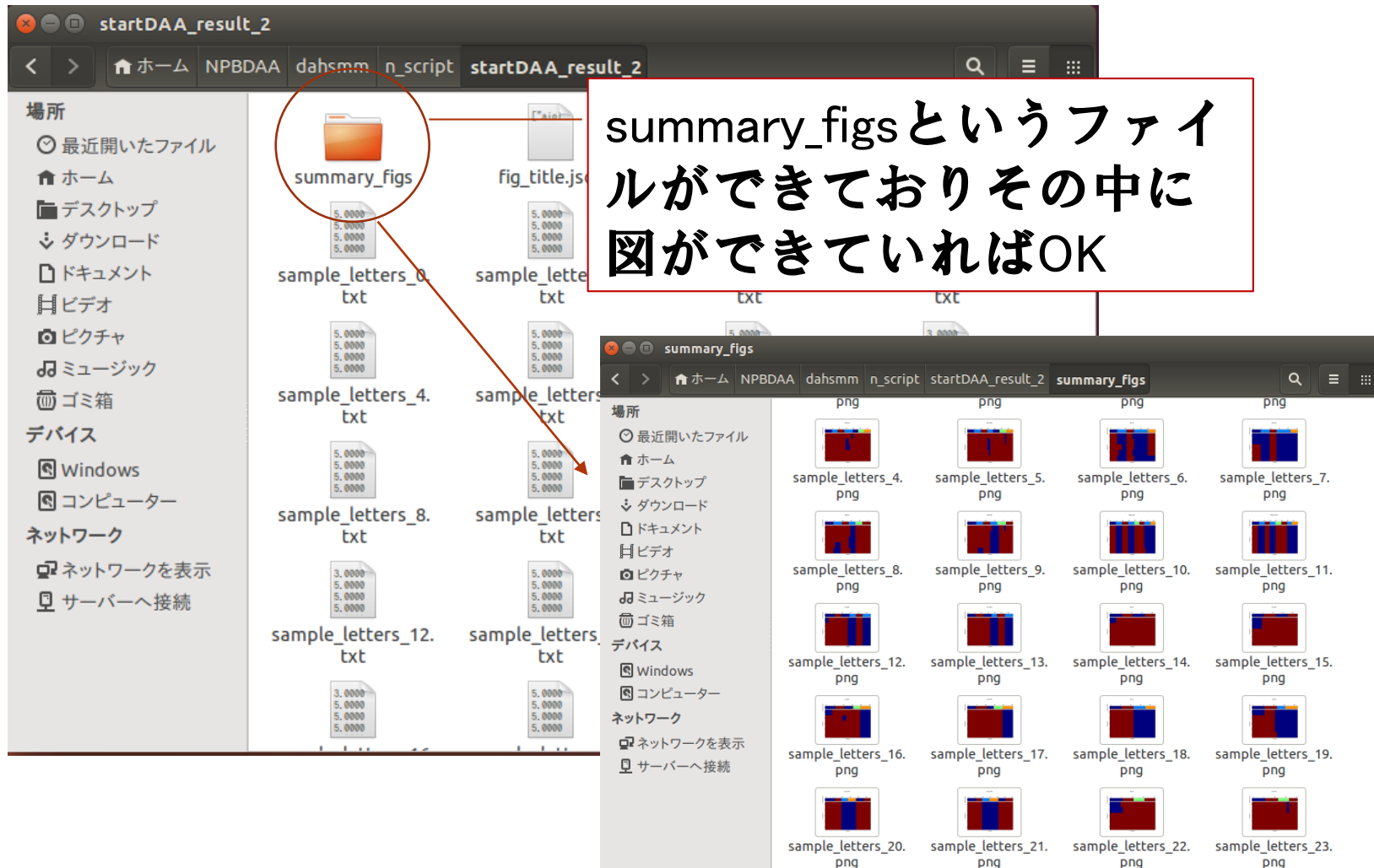
# Step5 結果を図示する

```
yamamura@yamamura-Endeavor-ST170E: ~/NPBDAA/dahsmm/n_script/startDAA_result_2
ie_ao plot finish-->count: 43
ie_aioi2 plot finish-->count: 44
ie_aue plot finish-->count: 45
ie_uo2 plot finish-->count: 46
ie_ie_uo plot finish-->count: 47
ie_ie2 plot finish-->count: 48
uo_ao2 plot finish-->count: 49
uo_uo2 plot finish-->count: 50
uo_ie2 plot finish-->count: 51
uo_uo plot finish-->count: 52
uo_ao plot finish-->count: 53
uo_ie plot finish-->count: 54
uo_aioi2 plot finish-->count: 55
ie_ie_uo2 plot finish-->count: 56
uo_aue_ie plot finish-->count: 57
uo_aue2 plot finish-->count: 58
uo_aue plot finish-->count: 59
uo_aue_ie2 plot finish-->count: 60
-----plot process completed!!-----
-----
```

こんな表示が出ればOK



# Step5 結果を図示する



# Step5. 5 出力ファイル説明

aRIs\_l.txt  
aRIs\_s.txt  
confused\_matrix\_l.csv  
confused\_matrix\_s.csv  
maxLk\_adjusted\_rand\_index\_l.txt  
maxLk\_adjusted\_rand\_index\_s.txt  
PERandWER.txt  
sample\_letters\_0.png

sample\_letters\_25.png  
sample\_letters\_26.png  
sample\_letters\_57.png  
sample\_letters\_58.png  
sample\_states\_29.png  
sample\_states\_30.png  
sample\_states\_31.png  
sample\_states\_32.png  
sample\_states\_33.png  
sample\_states\_34.png  
sample\_states\_35.png  
sample\_states\_36.png

これは中島用なので考えなくてよいです

state\_boundary\_1.png  
state\_boundary\_2.png  
state\_boundary\_3.png  
state\_boundary\_4.png  
state\_boundary\_5.png  
state\_boundary\_6.png  
state\_boundary\_7.png  
state\_boundary\_8.png

state\_boundary\_33.png  
state\_boundary\_34.png  
state\_boundary\_35.png  
state\_boundary\_36.png  
state\_boundary\_37.png  
state\_boundary\_38.png  
state\_boundary\_39.png  
state\_boundary\_40.png

## すべてのデータを通して推定された単語リスト

```
90 iter89:: 0:(1, 2, 6, 1, 6, 0, 6) 1:(1, 0, 1, 2) 2:(6, 1, 6, 0, 6) 3:(1,) 4:(6,) 5:(6, 2, 0) 6:(1, 2, 6)
91 iter90:: 0:(1, 2, 6, 1, 6, 0, 6) 1:(1, 0, 1, 2) 2:(6, 1, 6, 0, 6) 3:(1,) 4:(6,) 5:(6, 2, 0) 6:(1, 2, 6)
92 iter91:: 0:(1, 2, 6, 1, 6, 0, 6) 1:(1, 0, 1, 2) 2:(6, 1, 6, 0, 6) 3:(1,) 4:(6,) 5:(6, 2, 0) 6:(1, 2, 6)
93 iter92:: 0:(1, 2, 6, 1, 6, 0, 6) 1:(1, 0, 1, 2) 2:(6, 1, 6, 0, 6) 3:(1,) 4:(6,) 5:(6, 2, 0) 6:(1, 2, 6)
94 iter93:: 0:(1, 2, 6, 1, 6, 0, 6) 1:(1, 0, 1, 2) 2:(6, 1, 6, 0, 6) 3:(1,) 4:(6,) 5:(6, 2, 0) 6:(1, 2, 6)
95 iter94:: 0:(1, 2, 6, 1, 6, 0, 6) 1:(1, 0, 1, 2) 2:(6, 1, 6, 0, 6) 3:(1,) 4:(6,) 5:(6, 2, 0) 6:(1, 2, 3, 6)
96 iter95:: 0:(1, 2, 6, 1, 6, 0, 6) 1:(1, 0, 1, 2) 2:(6, 1, 6, 0, 6) 3:(1,) 4:(6,) 5:(6, 2, 0) 6:(1, 2, 3, 6, 3)
97 iter96:: 0:(1, 2, 6, 1, 6, 0, 6) 1:(1, 0, 1, 2) 2:(6, 1, 6, 0, 6) 3:(1,) 4:(6,) 5:(6, 2, 0) 6:(1, 2, 3, 6, 3)
98 iter97:: 0:(1, 2, 6, 1, 6, 0, 6) 1:(1, 0, 1, 2) 2:(6, 1, 6, 0, 6) 3:(1,) 4:(6,) 5:(6, 2, 0) 6:(1, 2, 3, 6, 3)
99 iter98:: 0:(1, 2, 6, 1, 6, 0, 6) 1:(1, 0, 1, 2) 2:(6, 1, 6, 0, 6) 3:(1,) 4:(6,) 5:(6, 2, 0) 6:(1, 2, 3, 6, 3)
100 iter99:: 0:(1, 2, 6, 1, 6, 0, 6) 1:(1, 0, 1, 2) 2:(6, 1, 6, 0, 6) 3:(1,) 4:(6,) 5:(6, 2, 0) 6:(1, 2, 3, 6, 3)
```

イテレーション

単語ラベル

単語の音素ラベル列

state\_boundary\_41.png  
state\_boundary\_42.png  
state\_boundary\_43.png  
state\_boundary\_44.png  
state\_boundary\_45.png  
state\_boundary\_46.png  
state\_boundary\_47.png  
state\_boundary\_48.png  
state\_boundary\_49.png  
state\_boundary\_50.png  
state\_boundary\_51.png  
state\_boundary\_52.png  
state\_boundary\_53.png  
state\_boundary\_54.png  
state\_boundary\_55.png  
state\_boundary\_56.png  
state\_boundary\_57.png  
state\_boundary\_58.png  
state\_boundary\_59.png  
WordList.txt

sample\_letters\_24.png  
sample\_letters\_56.png  
sample\_states\_28.png  
state\_boundary\_0.png  
state\_boundary\_32.png

# Step5. 5 出力ファイル説明

aRIs\_l.txt  
aRIs\_s.txt  
confused\_matrix\_l.csv  
confused\_matrix\_s.csv  
maxLk\_adjusted\_rand\_index\_l.txt  
maxLk\_adjusted\_rand\_index\_s.txt  
PERandWER.txt  
sample\_letters\_0.png  
sample\_letters\_1.png  
sample\_letters\_2.png  
sample\_letters\_3.png  
sample\_letters\_4.png  
sample\_letters\_5.png  
sample\_letters\_6.png  
sample\_letters\_7.png  
sample\_letters\_8.png  
sample\_letters\_9.png  
sample\_letters\_10.png  
sample\_letters\_11.png  
sample\_letters\_12.png  
sample\_letters\_13.png  
sample\_letters\_14.png  
sample\_letters\_15.png  
sample\_letters\_16.png  
sample\_letters\_17.png  
sample\_letters\_18.png  
sample\_letters\_19.png  
sample\_letters\_20.png  
sample\_letters\_21.png  
sample\_letters\_22.png  
sample\_letters\_23.png  
sample\_letters\_24.png

sample\_letters\_25.png  
sample\_letters\_26.png  
sample\_letters\_27.png  
sample\_letters\_28.png  
sample\_letters\_29.png  
sample\_letters\_30.png  
sample\_letters\_31.png  
sample\_letters\_32.png  
sample\_letters\_33.png  
sample\_letters\_34.png  
sample\_letters\_35.png  
sample\_letters\_36.png  
sample\_letters\_37.png  
sample\_letters\_38.png  
sample\_letters\_39.png  
sample\_letters\_40.png  
sample\_letters\_41.png  
sample\_letters\_42.png  
sample\_letters\_43.png  
sample\_letters\_44.png  
sample\_letters\_45.png  
sample\_letters\_46.png  
sample\_letters\_47.png  
sample\_letters\_48.png  
sample\_letters\_49.png  
sample\_letters\_50.png  
sample\_letters\_51.png  
sample\_letters\_52.png  
sample\_letters\_53.png  
sample\_letters\_54.png  
sample\_letters\_55.png  
sample\_letters\_56.png  
sample\_letters\_57.png  
sample\_letters\_58.png  
sample\_letters\_59.png  
sample\_letters\_60.png  
sample\_letters\_61.png  
sample\_letters\_62.png  
sample\_letters\_63.png  
sample\_letters\_64.png  
sample\_letters\_65.png  
sample\_letters\_66.png  
sample\_letters\_67.png  
sample\_letters\_68.png  
sample\_letters\_69.png  
sample\_letters\_70.png  
sample\_letters\_71.png  
sample\_letters\_72.png  
sample\_letters\_73.png  
sample\_letters\_74.png  
sample\_letters\_75.png  
sample\_letters\_76.png  
sample\_letters\_77.png  
sample\_letters\_78.png  
sample\_letters\_79.png  
sample\_letters\_80.png  
sample\_letters\_81.png  
sample\_letters\_82.png  
sample\_letters\_83.png  
sample\_letters\_84.png  
sample\_letters\_85.png  
sample\_letters\_86.png  
sample\_letters\_87.png  
sample\_letters\_88.png  
sample\_letters\_89.png  
sample\_letters\_90.png  
sample\_letters\_91.png  
sample\_letters\_92.png  
sample\_letters\_93.png  
sample\_letters\_94.png  
sample\_letters\_95.png  
sample\_letters\_96.png  
sample\_letters\_97.png  
sample\_letters\_98.png  
sample\_letters\_99.png

sample\_states\_1.png  
sample\_states\_2.png  
sample\_states\_3.png  
sample\_states\_4.png  
sample\_states\_5.png  
sample\_states\_6.png  
sample\_states\_7.png  
sample\_states\_8.png  
sample\_states\_9.png  
sample\_states\_10.png  
sample\_states\_11.png  
sample\_states\_12.png  
sample\_states\_13.png  
sample\_states\_14.png  
sample\_states\_15.png  
sample\_states\_16.png  
sample\_states\_17.png  
sample\_states\_18.png  
sample\_states\_19.png  
sample\_states\_20.png  
sample\_states\_21.png  
sample\_states\_22.png  
sample\_states\_23.png  
sample\_states\_24.png  
sample\_states\_25.png  
sample\_states\_26.png  
sample\_states\_27.png  
sample\_states\_28.png  
sample\_states\_29.png  
sample\_states\_30.png  
sample\_states\_31.png  
sample\_states\_32.png  
sample\_states\_33.png  
sample\_states\_34.png  
sample\_states\_35.png  
sample\_states\_36.png  
sample\_states\_37.png  
sample\_states\_38.png  
sample\_states\_39.png  
sample\_states\_40.png  
sample\_states\_41.png  
sample\_states\_42.png  
sample\_states\_43.png  
sample\_states\_44.png  
sample\_states\_45.png  
sample\_states\_46.png  
sample\_states\_47.png  
sample\_states\_48.png  
sample\_states\_49.png  
sample\_states\_50.png  
sample\_states\_51.png  
sample\_states\_52.png  
sample\_states\_53.png  
sample\_states\_54.png  
sample\_states\_55.png  
sample\_states\_56.png  
sample\_states\_57.png  
sample\_states\_58.png  
sample\_states\_59.png  
sample\_states\_60.png  
sample\_states\_61.png  
sample\_states\_62.png  
sample\_states\_63.png  
sample\_states\_64.png  
sample\_states\_65.png  
sample\_states\_66.png  
sample\_states\_67.png  
sample\_states\_68.png  
sample\_states\_69.png  
sample\_states\_70.png  
sample\_states\_71.png  
sample\_states\_72.png  
sample\_states\_73.png  
sample\_states\_74.png  
sample\_states\_75.png  
sample\_states\_76.png  
sample\_states\_77.png  
sample\_states\_78.png  
sample\_states\_79.png  
sample\_states\_80.png  
sample\_states\_81.png  
sample\_states\_82.png  
sample\_states\_83.png  
sample\_states\_84.png  
sample\_states\_85.png  
sample\_states\_86.png  
sample\_states\_87.png  
sample\_states\_88.png  
sample\_states\_89.png  
sample\_states\_90.png  
sample\_states\_91.png  
sample\_states\_92.png  
sample\_states\_93.png  
sample\_states\_94.png  
sample\_states\_95.png  
sample\_states\_96.png  
sample\_states\_97.png  
sample\_states\_98.png  
sample\_states\_99.png

state\_boundary\_1.png  
state\_boundary\_2.png  
state\_boundary\_3.png  
state\_boundary\_4.png  
state\_boundary\_5.png  
state\_boundary\_6.png  
state\_boundary\_7.png  
state\_boundary\_8.png  
state\_boundary\_9.png  
state\_boundary\_10.png  
state\_boundary\_11.png  
state\_boundary\_12.png  
state\_boundary\_13.png  
state\_boundary\_14.png  
state\_boundary\_15.png  
state\_boundary\_16.png  
state\_boundary\_17.png  
state\_boundary\_18.png  
state\_boundary\_19.png  
state\_boundary\_20.png  
state\_boundary\_21.png  
state\_boundary\_22.png  
state\_boundary\_23.png  
state\_boundary\_24.png  
state\_boundary\_25.png  
state\_boundary\_26.png  
state\_boundary\_27.png  
state\_boundary\_28.png  
state\_boundary\_29.png  
state\_boundary\_30.png  
state\_boundary\_31.png  
state\_boundary\_32.png  
state\_boundary\_33.png  
state\_boundary\_34.png  
state\_boundary\_35.png  
state\_boundary\_36.png  
state\_boundary\_37.png  
state\_boundary\_38.png  
state\_boundary\_39.png  
state\_boundary\_40.png  
state\_boundary\_41.png  
state\_boundary\_42.png  
state\_boundary\_43.png  
state\_boundary\_44.png  
state\_boundary\_45.png  
state\_boundary\_46.png  
state\_boundary\_47.png  
state\_boundary\_48.png  
state\_boundary\_49.png  
state\_boundary\_50.png  
state\_boundary\_51.png  
state\_boundary\_52.png  
state\_boundary\_53.png  
state\_boundary\_54.png  
state\_boundary\_55.png  
state\_boundary\_56.png  
state\_boundary\_57.png  
state\_boundary\_58.png  
state\_boundary\_59.png  
state\_boundary\_60.png  
state\_boundary\_61.png  
state\_boundary\_62.png  
state\_boundary\_63.png  
state\_boundary\_64.png  
state\_boundary\_65.png  
state\_boundary\_66.png  
state\_boundary\_67.png  
state\_boundary\_68.png  
state\_boundary\_69.png  
state\_boundary\_70.png  
state\_boundary\_71.png  
state\_boundary\_72.png  
state\_boundary\_73.png  
state\_boundary\_74.png  
state\_boundary\_75.png  
state\_boundary\_76.png  
state\_boundary\_77.png  
state\_boundary\_78.png  
state\_boundary\_79.png  
state\_boundary\_80.png  
state\_boundary\_81.png  
state\_boundary\_82.png  
state\_boundary\_83.png  
state\_boundary\_84.png  
state\_boundary\_85.png  
state\_boundary\_86.png  
state\_boundary\_87.png  
state\_boundary\_88.png  
state\_boundary\_89.png  
state\_boundary\_90.png  
state\_boundary\_91.png  
state\_boundary\_92.png  
state\_boundary\_93.png  
state\_boundary\_94.png  
state\_boundary\_95.png  
state\_boundary\_96.png  
state\_boundary\_97.png  
state\_boundary\_98.png  
state\_boundary\_99.png

Based sample\_letters\_0.txt

フレームごとの真音素ラベル

イテレーション毎におけるフレームごとの推定音素ラベル

aiou\_aioi

Truth Label

Iteration

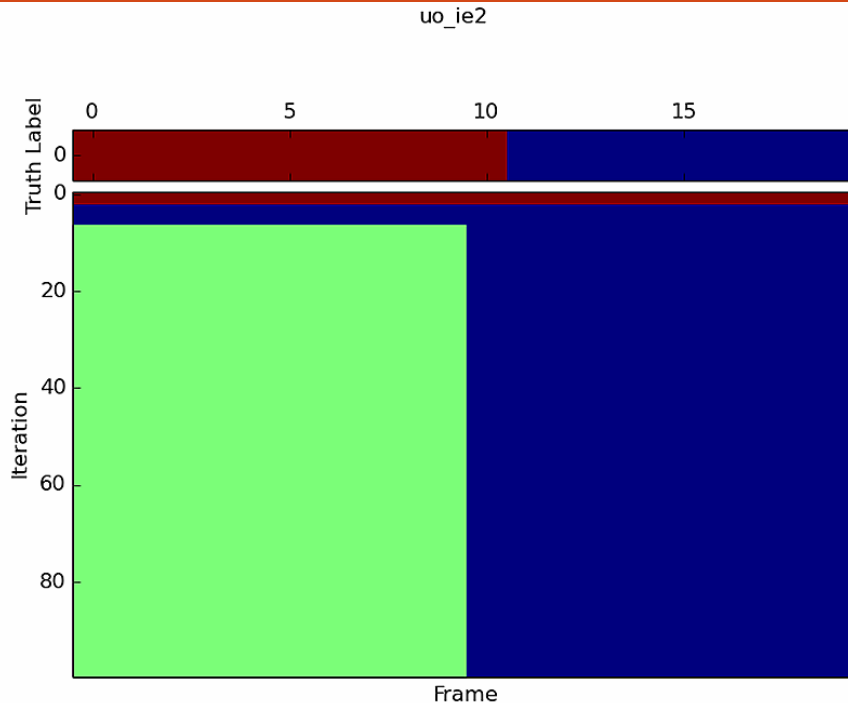
Frame



# Step5. 5 出力ファイル説明

フレームごとの  
真単語ラベル

イテレーション  
毎における  
フレームごとの  
推定単語ラベル



Based sample\_states\_57.txt

# Step5. 5 出力ファイル説明

aRIs\_l.txt  
aRIs\_s.txt  
confused\_matrix\_l.csv  
confused\_matrix\_s.csv  
maxLk\_adjusted\_rand\_index\_l.txt  
maxLk\_adjusted\_rand\_index\_s.txt  
PERandWER.txt  
sample\_letters\_0.png  
sample\_letters\_1.png

sample\_letters\_25.png  
sample\_letters\_26.png  
sample\_letters\_27.png

sample\_letters\_57.png  
sample\_letters\_58.png  
sample\_letters\_59.png

sample\_states\_29.png  
sample\_states\_30.png  
sample\_states\_31.png

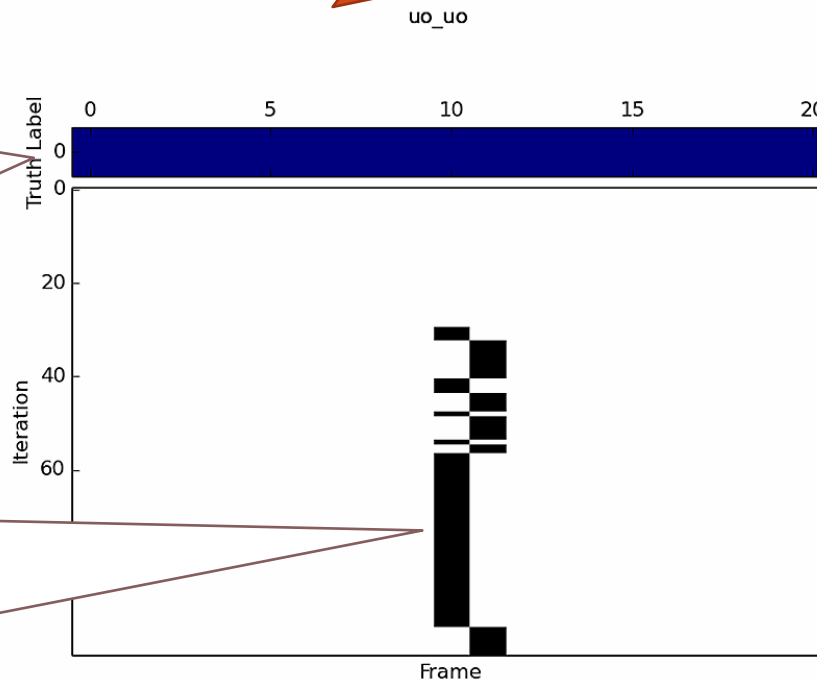
state\_boundary\_1.png  
state\_boundary\_2.png  
state\_boundary\_3.png  
state\_boundary\_4.png  
state\_boundary\_5.png  
state\_boundary\_6.png  
state\_boundary\_7.png  
state\_boundary\_8.png  
state\_boundary\_9.png

state\_boundary\_33.png  
state\_boundary\_34.png  
state\_boundary\_35.png  
state\_boundary\_36.png  
state\_boundary\_37.png  
state\_boundary\_38.png  
state\_boundary\_39.png  
state\_boundary\_40.png  
state\_boundary\_41.png

Based state\_ranges\_58.txt

フレームごとの  
真単語ラベル

イテレーション  
毎における  
単語境界が何  
フレーム目に  
存在するか



state\_boundary\_42.png  
state\_boundary\_43.png  
state\_boundary\_44.png  
state\_boundary\_45.png  
state\_boundary\_46.png  
state\_boundary\_47.png  
state\_boundary\_48.png  
state\_boundary\_49.png  
state\_boundary\_50.png  
state\_boundary\_51.png  
state\_boundary\_52.png  
state\_boundary\_53.png  
state\_boundary\_54.png  
state\_boundary\_55.png  
state\_boundary\_56.png  
state\_boundary\_57.png  
state\_boundary\_58.png  
state\_boundary\_59.png  
WordList.txt

sample\_letters\_24.png

sample\_letters\_56.png

sample\_states\_28.png

state\_boundary\_0.png

state\_boundary\_32.png

# Step6 startDAA.py 説明

```
1 import os
2 if "XDG_CACHE_HOME" in os.environ: del os.environ["XDG_CACHE_HOME"]
3 import pyhsmm
4 import pyhsmm.parallel
5 from pyhsmm import parallel
6 from pyhsmm.basic.pybasicbayes.distributions import Gaussian as Gaussian
7 from pyhsmm.util.text import progprint_xrange
8 from dahsmm.hsmm import DAHSMM
9 from dahsmm.word import LLHSMM
10 from dahsmm.util import get_result_dir, save_parameters, save_fig_title, Result
11 from dahsmm.states import HSMMState
12 import numpy as np
13 import glob
14 import multiprocessing
15 import pickle
16 import time
17 #-----save paras-----
18 SAVE_PARAMS = ['ITER_N',
19 LETTER_N
20 WORD_N
21 DATA_N
22 model_hypparams
23 word_model_params
24 obs_hypparams
25 dur_hypparams
26 filename''.split('\n')]
```

- 各種インポート

- 保存するパラメータ

- parameter.json  
に保存される

# Step6 startDAA.py 説明

```
36 def main(result_dir=None):
37     if result_dir == None:
38         result_dir = get_result_dir(__file__)
39         os.mkdir(result_dir)
40         param_path = os.path.join(result_dir, 'parameter.json')
41         fig_title_path = os.path.join(result_dir, 'fig_title.json')
42     #initialize model params#
43     #O(T*L_max*W_max^2*d_max^3)
44     ITER_N = 100
45     LETTER_N = 7
46     WORD_N = 7
47     DATA_N = 60
48     obs_dim = 3
49     model_hypparams = {'state_dim': WORD_N, 'alpha': 10.0, 'gamma': 10.0}
50     word_model_params = {'letter_type': LETTER_N, 'rho': 10}
51     obs_hypparams = {'mu_0': np.zeros(obs_dim), 'sigma_0': np.eye(obs_dim), 'kappa_0': 0.01, 'nu_0': obs_dim + 5}
52     dur_hypparams = {'alpha_0': 200.0, 'beta_0': 10.0}
53     length_dist = pyhsmm.distributions.PoissonDuration(alpha_0=30, beta_0=10, Lmbda=3)
```

結果を保存するファイルのパス&フォルダ生成

- パラメータ設定部分（一回目のpyhsmmと基本は同じ
  - dahsmm部分
    - 最大単語数
    - 潜在単語の遷移確率ハイパーパラメータ
    - 単語の文字列長分布仮定（ポアソン分布を仮定）

# Step6 startDAA.py 説明

```
#setting#
obs_dists = [Gaussian(**obs_hypparams) for state in range(LETTER_N)]
dur_dists = [pyhsmm.distributions.PoissonDuration(**dur_hypparams) for state in range(LETTER_N)]
letter_hsmm = LLHSM(
    init_state_concentration = 10,
    alpha = 10.0,
    gamma = 10.0,
    dur_dists = dur_dists,
    obs_dists = obs_dists
)
model = DAHSM(
    model_hypparams,
    letter_hsmm,
    length_dist,
    obs_dists,
    dur_dists
)
```

pyhsmmと同じ

- hsmmオブジェクト生成
- dahsmmオブジェクト生成

# Step6 startDAA.py 説明

```
71 #dump_object in multi engines#
72 """
73 print "-----dump process start-----"
74 count = multiprocessing.Value('i', 0)
75 datatxt_names = glob.glob('./DATA/*.txt')
76 datatxt_names.sort()
77 for f in datatxt_names:
78     input_mat = np.loadtxt(f)
79     f = f.replace("./DATA/", "")
80     f = f.replace(".txt", "")
81     pr = multiprocessing.Process(target=multi_dump_object, args=(input_mat,model,f,count))
82     pr.start()
83 while (1):
84     if count.value > 59:
85         time.sleep(1)
86         print "-----dump process completed!!-----"
87         break
88 """
89 #add_data in one engine#
90 """
91 filename = []
92 for f in glob.glob('./DATA/*.txt'):
93     model.add_data(np.loadtxt(f))
94     f = f.replace("./DATA/", "")
95     filename.append(f.replace(".txt", ""))
96 """
```

```
27 #-----multi process function-----
28 def multi_dump_object(data,model,f,count):
29     print f, " dumping..."
30     fp = open("TMP/"+f+".dump", 'w')
31     pickle.dump(HSMMState(data,model), fp)
32     fp.close()
33     count.value = count.value + 1
34     print f, 'dump finish-->count:',count.value
```

- DATAフォルダ上の入力データ読み込みを並列処理  
multiprocessing.Process()
- 読み込みの逐次処理  
バージョン

データをHSMMオブジェクトに読み込みそれをTMPフォルダに保存する関数  
(これを並列処理)



# Step6 startDAA.py 説明

```
97 #add_data in multi engines#
98 print "-----add_data process start-----"
99 filename = []
100 datadmp_names = glob.glob('./TMP/*.dump')
101 datadmp_names.sort()
102 for f in datadmp_names:
103     print f, " loading..."
104     fp = open(f)
105     f = f.replace("./TMP/", "")
106     f = f.replace(".dump", "")
107     filename.append(f)
108     obj = pickle.load(fp)
109     model.states_list.append(obj)
110     if model.parallel:
111         parallel.add_data(model.states_list[-1].data)
112     fp.close()
113 print "-----add_data process completed!--"
```

- TMPフォルダ上の保存したオブジェクトファイル (\*.dump) を読み込み, DAHSMMオブジェクトに追加

一度TMP上に保存したら, 使いまわすことができる.  
(前ページの部分をコメントアウトする)

```
114 #save params&charm#
115 save_fig_title(fig_title_path, SAVE_PARAMS, locals())
116 save_parameters(param_path, SAVE_PARAMS, locals())
117 obs_hypparams['sigma_0']=np.eye(obs_dim)
118 obs_hypparams['mu_0']=np.zeros(obs_dim)
```

fig\_title.json (データ名, 順番), parameter.json (各種パラメータ) を保存する

# Step6 startDAA.py 説明

```
119 #estimation&result_write#
120 print "-----estimation process start-----"
121 result = Result(result_dir, DATA_N)
122 loglikelihood = []
123 for idx in xrange(ITER_N, perline = 10): }
124     model.resample_model()
125     loglikelihood.append(result.save_loglikelihood(model))
126     result.save(model)
127     result.write_loglikelihood(loglikelihood)
128 print "-----estimation process completed!--"
```

pyhsmmと同じ

- 推定部分

- startDAA\_result\_\*上にイテレーション毎でリサルトファイルを保存, 更新 (sample\_letters, sample\_states, state\_boundary)
- サンプリング
- startDAA\_result\_\*上にイテレーション毎の尤度を保存



# Step6 startDAA.py 説明

```
55 class Result(object):
56     def __init__(self, dirname, data_n):
57         self.dir = dirname
58         self.states = [[] for _ in range(data_n)]
59         self.state_ranges = [[] for _ in range(data_n)]
60         self.letters = [[] for _ in range(data_n)]
61         self.word_list = []
62
63     def dir_path(self, name):
64         return os.path.join(self.dir, name)
65
66     def save(self, model):
67         self.save_sampled_states(model)
68         self.save_word_list(model)
69         self.save_state_ranges(model)
70         self.write()
71
72     #loglikelihood_nakashima-----
73     def save_loglikelihood(self, model):
74         ll=[]
75         for k in model.states_list:
76             b=k.betal[0]
77             c=np.logaddexp.reduce(b)
78             ll.append(c)
79         reloglid = np.logaddexp.reduce(ll)
80         print reloglid
81         return reloglid
82
83     def write_loglikelihood(self, LogLikelihood):
84         savetxt(self.dir_path('loglikelihood.txt'), loglikelihood, '%
85         .2f')
86
87     def save_sampled_states(self, model):
88         for i, sampled in enumerate(model.states_list):
89             self.states[i].append(sampled.stateseq)
90             self.letters[i].append(sampled.letterseq)
91
92     def save_state_ranges(self, model):
93         for i, sample in enumerate(model.states_list):
94             print sample.state_ranges
95             self.state_ranges[i].append(copy.copy(sample.state_ranges
96             ))
97
98     def save_word_list(self, model):
99         self.word_list.append(copy.copy(model.word_list))
100
101     def write(self):
102         for idx, (states, letters) in enumerate(zip(self.states, self
103         .letters)):
104             savetxt(self.dir_path('sample_states_%d.txt' % idx), self
105             .states[idx])
106             savetxt(self.dir_path('sample_letters_%d.txt' % idx),
107             self.letters[idx])
108
109         with open(self.dir_path('sample_word_list.txt'), 'w') as f:
110             pickle.dump(self.word_list, f)
111
112         for idx, ranges in enumerate(self.state_ranges):
113             with open(self.dir_path('state_ranges_%d.txt' % idx), 'w'
114             ) as f:
115                 pickle.dump(ranges, f)
```

Result class 部分 (dahsmm/util.py)

# Step7 summary.py 説明

```
26 #-----main function-----#
27 def main():
28     #result_file make#
29     parser = argparse.ArgumentParser()
30     parser.add_argument('directory')
31     #opts = parser.parse_args()
32     figs_dir = 'summary_figs'
33     os.path.exists(figs_dir) or os.mkdir(figs_dir)
34     summary = Summary()
35     #evaluation_result save#
36     with pushd(figs_dir):
37         #gen confused matrix
38         summary.letter_confused_matrix()
39         summary.state_confused_matrix()
40         #gen PER and WER
41         summary.cu1PER()
42         summary.cu1WER()
43         #gen adjusted rand index
44         summary.a_rand_index(summary.sample_letters,summary.input_data,'l')
45         summary.a_rand_index(summary.sample_states,summary.input_data2,'s')
46         #gen word list
47         with open('WordList.txt','w') as f:
48             for num, key in enumerate(summary.word_list):
49                 f.write("iter%d::: " % num)
50                 for num2, key2 in enumerate(key):
51                     f.write("%d:" % num2 + str(key2) + " ")
52                 f.write("\n")
53     #multi plot sample states and letters#
54     print("-----plot process start-----")
55     count = multiprocessing.Value('i', 0)
56     for idx in range(summary.data_size):
57         pr = multiprocessing.Process(target=multi_plot_object, args=(summary,idx,count))
58         pr.start()
59         time.sleep(0.1) #charm!!!!(koreganaito roop karanukenai)
60     while (1):
61         if count.value > 59:
62             time.sleep(1)
63             print("-----plot process completed!!-----")
64             break
65 #=====summary(main process?) class=====
```

summary\_figs  
フォルダ作成,  
Summaryクラ  
ス生成

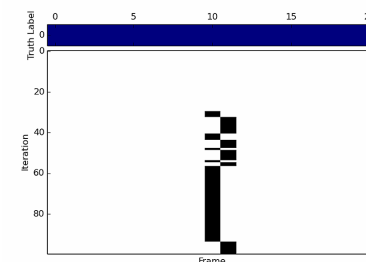
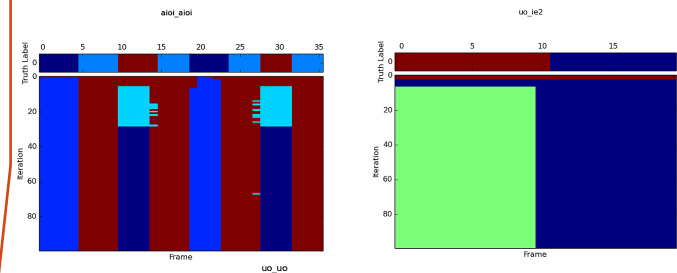
summary\_figs  
フォルダ移動

中島用

推定単語リ  
スト作成  
WordList.txt

## グラフ作成処理を並列処理 multiprocessing.Process()

```
14 #-----multi process function-----
15 def multi_plot_object(summary,idx,count):
16     print summary.fig_title[idx], " plotting..."
17     summary.plot_states(idx)
18     plt.savefig('sample_states_%d.png' % idx)
19     summary.plot_state_boundaries(idx)
20     plt.savefig('state_boundary_%d.png' % idx)
21     summary.plot_letters(idx)
22     plt.savefig('sample_letters_%d.png' % idx)
23     plt.clf()
24     count.value = count.value + 1
25     print summary.fig_title[idx], 'plot finish-->count:',count.value
```



# Step7 summary.py 説明 (Summary class)

```
65 #=====summary(main process?) class=====#
66 class Summary(object):
67 #-----init paras-----#
68 def __init__(self, dirpath = '.'):
69     with open('parameter.json') as f:
70         params = self.params = json.load(f)
71     with open('fig_title.json') as f2:
72         fig_title = self.fig_title = json.load(f2)
73     with open('sample_word_list.txt') as f3:
74         self.word_list = pickle.load(f3)
75     self.data_size = params['DATA_N']
76     self.input_data = [np.loadtxt("../LABEL/"+i+".lab") for i in fig_title]
77     self.input_data2 = [np.loadtxt("../LABEL/"+i+".lab2") for i in fig_title]
78     self.sample_states = [np.loadtxt('sample_states_%d.txt'%i) for i in range(params['DATA_N'])]
79     self.sample_letters = [np.loadtxt('sample_letters_%d.txt'%i) for i in range(params['DATA_N'])]
80     self.state_ranges = []
81     for i in range(params['DATA_N']):
82         with open('state_ranges_%d.txt'%i) as f:
83             self.state_ranges.append(pickle.load(f))
84     llist = np.loadtxt("loglikelihood.txt").tolist()
85     self.maxlikelihood = (max(llist), llist.index(max(llist)))
86     #manipulation part
87     self.l_label_dic={}
88     self.s_label_dic={}
89     #manipulation part end
```

中島用

各種ファイル読み込み

parameter.json : パラメータ)

fig\_title.json : データ名, 順番

LABEL/\*lab

(2) : 真ラベル

sample\_letters\_\*.txt

sample\_states\_\*.txt

state\_ranges\_\*.txt

loglikelihood.txt : 尤

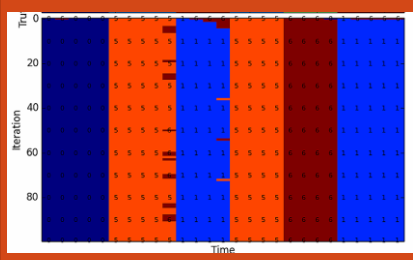
度 (イテレーションの中で最大だった尤度判定用)

```

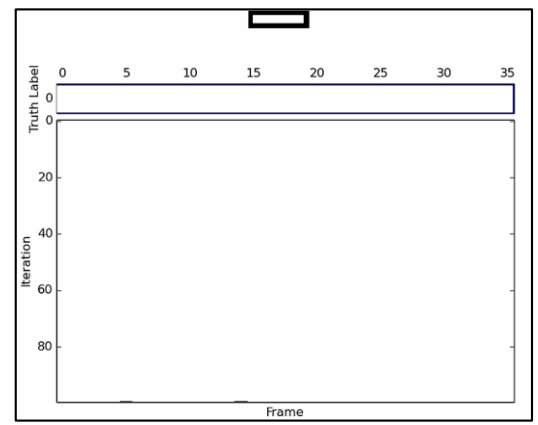
90 #-----write_result_graph-----#
91 #base_graph function#
92 def _plot_discreate_sequence(self, true_data, title, sample_data, label='u', ploptops={}):
93     ax = plt.subplot2grid((10,1), (1,0))
94     plt.sca(ax)
95     ax.matshow([true_data], aspect='auto')
96     plt.ylabel('Truth Label')
97     #label matrix
98     ax = plt.subplot2grid((10,1), (2,0), rowspan=8)
99     plt.suptitle(title)
100    plt.sca(ax)
101    ax.matshow(sample_data, aspect='auto', **ploptops)
102    #write per 10 iterations(max_likelihood) label
103    """
104    for i in range(label.shape[0]):
105        for j in range(label.shape[1]):
106            if i%10==0 or i==99 or i==self.maxlikelihood[1]:
107                if i==self.maxlikelihood[1]:
108                    ax.text(j, i+1.5, int(label[i][j]), ha='center', va='bottom', color='red', fontsize=8)
109                else:
110                    ax.text(j, i+1.5, int(label[i][j]), ha='center', va='bottom', color='black', fontsize=8)
111            ax.text(j, i+1.5, int(label[i][j]), ha='center', va='bottom', color='black', fontsize=8)
112    """
113    #write x&y label
114    plt.xlabel('Frame')
115    plt.ylabel('Iteration')
116    plt.xticks(())

```

## グラフにラベル印字



## グラフの共通デザイン設定 (軸ラベルや図位置など)



## 各種グラフ作成関数

- sample\_letters\*.png
- sample\_states\*.png
- state\_boundary\*.png

Summaryクラス  
内の以降の関数  
は中島用なので  
必要ないです  
(消しても良い)

```

140 #-----compute adjusted rand index-----#
141 def _a_rand_index(self, sample_data, true_data, char):
142     #true index
143     true_idx = range(len(sample_data[0]))
144     #sample index
145     sample_idx = range(len(sample_data[0]))
146     #true index
147     true_idx = range(len(sample_data[0]))
148     #sample index
149     sample_idx = range(len(sample_data[0]))
150     #true index
151     true_idx = range(len(sample_data[0]))
152     #sample index
153     sample_idx = range(len(sample_data[0]))
154     #true index
155     true_idx = range(len(sample_data[0]))
156     #sample index
157     sample_idx = range(len(sample_data[0]))
158     #true index
159     true_idx = range(len(sample_data[0]))
160     #sample index
161     sample_idx = range(len(sample_data[0]))
162     #true index
163     true_idx = range(len(sample_data[0]))
164     #sample index
165     sample_idx = range(len(sample_data[0]))
166     #true index
167     true_idx = range(len(sample_data[0]))
168     #sample index
169     sample_idx = range(len(sample_data[0]))
170     #true index
171     true_idx = range(len(sample_data[0]))
172     #sample index
173     sample_idx = range(len(sample_data[0]))
174     #true index
175     true_idx = range(len(sample_data[0]))
176     #sample index
177     sample_idx = range(len(sample_data[0]))
178     #true index
179     true_idx = range(len(sample_data[0]))
180     #sample index
181     sample_idx = range(len(sample_data[0]))
182     #true index
183     true_idx = range(len(sample_data[0]))
184     #sample index
185     sample_idx = range(len(sample_data[0]))
186     #true index
187     true_idx = range(len(sample_data[0]))
188     #sample index
189     sample_idx = range(len(sample_data[0]))
190     #true index
191     true_idx = range(len(sample_data[0]))
192     #sample index
193     sample_idx = range(len(sample_data[0]))
194     #true index
195     true_idx = range(len(sample_data[0]))
196     #sample index
197     sample_idx = range(len(sample_data[0]))
198     #true index
199     true_idx = range(len(sample_data[0]))
200     #sample index
201     sample_idx = range(len(sample_data[0]))
202     #true index
203     true_idx = range(len(sample_data[0]))
204     #sample index
205     sample_idx = range(len(sample_data[0]))
206     #true index
207     true_idx = range(len(sample_data[0]))
208     #sample index
209     sample_idx = range(len(sample_data[0]))
210     #true index
211     true_idx = range(len(sample_data[0]))
212     #sample index
213     sample_idx = range(len(sample_data[0]))
214     #true index
215     true_idx = range(len(sample_data[0]))
216     #sample index
217     sample_idx = range(len(sample_data[0]))
218     #true index
219     true_idx = range(len(sample_data[0]))
220     #sample index
221     sample_idx = range(len(sample_data[0]))
222     #true index
223     true_idx = range(len(sample_data[0]))
224     #sample index
225     sample_idx = range(len(sample_data[0]))
226     #true index
227     true_idx = range(len(sample_data[0]))
228     #sample index
229     sample_idx = range(len(sample_data[0]))
230     #true index
231     true_idx = range(len(sample_data[0]))
232     #sample index
233     sample_idx = range(len(sample_data[0]))
234     #true index
235     true_idx = range(len(sample_data[0]))
236     #sample index
237     sample_idx = range(len(sample_data[0]))
238     #true index
239     true_idx = range(len(sample_data[0]))
240     #sample index
241     sample_idx = range(len(sample_data[0]))
242     #true index
243     true_idx = range(len(sample_data[0]))
244     #sample index
245     sample_idx = range(len(sample_data[0]))
246     #true index
247     true_idx = range(len(sample_data[0]))
248     #sample index
249     sample_idx = range(len(sample_data[0]))
250     #true index
251     true_idx = range(len(sample_data[0]))
252     #sample index
253     sample_idx = range(len(sample_data[0]))
254     #true index
255     true_idx = range(len(sample_data[0]))
256     #sample index
257     sample_idx = range(len(sample_data[0]))
258     #true index
259     true_idx = range(len(sample_data[0]))
260     #sample index
261     sample_idx = range(len(sample_data[0]))
262     #true index
263     true_idx = range(len(sample_data[0]))
264     #sample index
265     sample_idx = range(len(sample_data[0]))
266     #true index
267     true_idx = range(len(sample_data[0]))
268     #sample index
269     sample_idx = range(len(sample_data[0]))
270     #true index
271     true_idx = range(len(sample_data[0]))
272     #sample index
273     sample_idx = range(len(sample_data[0]))
274     #true index
275     true_idx = range(len(sample_data[0]))
276     #sample index
277     sample_idx = range(len(sample_data[0]))
278     #true index
279     true_idx = range(len(sample_data[0]))
280     #sample index
281     sample_idx = range(len(sample_data[0]))
282     #true index
283     true_idx = range(len(sample_data[0]))
284     #sample index
285     sample_idx = range(len(sample_data[0]))
286     #true index
287     true_idx = range(len(sample_data[0]))
288     #sample index
289     sample_idx = range(len(sample_data[0]))
290     #true index
291     true_idx = range(len(sample_data[0]))
292     #sample index
293     sample_idx = range(len(sample_data[0]))
294     #true index
295     true_idx = range(len(sample_data[0]))
296     #sample index
297     sample_idx = range(len(sample_data[0]))
298     #true index
299     true_idx = range(len(sample_data[0]))
300     #sample index
301     sample_idx = range(len(sample_data[0]))
302     #true index
303     true_idx = range(len(sample_data[0]))
304     #sample index
305     sample_idx = range(len(sample_data[0]))
306     #true index
307     true_idx = range(len(sample_data[0]))
308     #sample index
309     sample_idx = range(len(sample_data[0]))
310     #true index
311     true_idx = range(len(sample_data[0]))
312     #sample index
313     sample_idx = range(len(sample_data[0]))
314     #true index
315     true_idx = range(len(sample_data[0]))
316     #sample index
317     sample_idx = range(len(sample_data[0]))
318     #true index
319     true_idx = range(len(sample_data[0]))
320     #sample index
321     sample_idx = range(len(sample_data[0]))
322     #true index
323     true_idx = range(len(sample_data[0]))
324     #sample index
325     sample_idx = range(len(sample_data[0]))
326     #true index
327     true_idx = range(len(sample_data[0]))
328     #sample index
329     sample_idx = range(len(sample_data[0]))
330     #true index
331     true_idx = range(len(sample_data[0]))
332     #sample index
333     sample_idx = range(len(sample_data[0]))
334     #true index
335     true_idx = range(len(sample_data[0]))
336     #sample index
337     sample_idx = range(len(sample_data[0]))
338     #true index
339     true_idx = range(len(sample_data[0]))
340     #sample index
341     sample_idx = range(len(sample_data[0]))
342     #true index
343     true_idx = range(len(sample_data[0]))
344     #sample index
345     sample_idx = range(len(sample_data[0]))
346     #true index
347     true_idx = range(len(sample_data[0]))
348     #sample index
349     sample_idx = range(len(sample_data[0]))
350     #true index
351     true_idx = range(len(sample_data[0]))
352     #sample index
353     sample_idx = range(len(sample_data[0]))
354     #true index
355     true_idx = range(len(sample_data[0]))
356     #sample index
357     sample_idx = range(len(sample_data[0]))
358     #true index
359     true_idx = range(len(sample_data[0]))
360     #sample index
361     sample_idx = range(len(sample_data[0]))
362     #true index
363     true_idx = range(len(sample_data[0]))
364     #sample index
365     sample_idx = range(len(sample_data[0]))
366     #true index
367     true_idx = range(len(sample_data[0]))
368     #sample index
369     sample_idx = range(len(sample_data[0]))
370     #true index
371     true_idx = range(len(sample_data[0]))
372     #sample index
373     sample_idx = range(len(sample_data[0]))
374     #true index
375     true_idx = range(len(sample_data[0]))
376     #sample index
377     sample_idx = range(len(sample_data[0]))
378     #true index
379     true_idx = range(len(sample_data[0]))
380     #sample index
381     sample_idx = range(len(sample_data[0]))
382     #true index
383     true_idx = range(len(sample_data[0]))
384     #sample index
385     sample_idx = range(len(sample_data[0]))
386     #true index
387     true_idx = range(len(sample_data[0]))
388     #sample index
389     sample_idx = range(len(sample_data[0]))
390     #true index
391     true_idx = range(len(sample_data[0]))
392     #sample index
393     sample_idx = range(len(sample_data[0]))
394     #true index
395     true_idx = range(len(sample_data[0]))
396     #sample index
397     sample_idx = range(len(sample_data[0]))
398     #true index
399     true_idx = range(len(sample_data[0]))
400     #sample index
401     sample_idx = range(len(sample_data[0]))
402     #true index
403     true_idx = range(len(sample_data[0]))
404     #sample index
405     sample_idx = range(len(sample_data[0]))
406     #true index
407     true_idx = range(len(sample_data[0]))
408     #sample index
409     sample_idx = range(len(sample_data[0]))
410     #true index
411     true_idx = range(len(sample_data[0]))
412     #sample index
413     sample_idx = range(len(sample_data[0]))
414     #true index
415     true_idx = range(len(sample_data[0]))
416     #sample index
417     sample_idx = range(len(sample_data[0]))
418     #true index
419     true_idx = range(len(sample_data[0]))
420     #sample index
421     sample_idx = range(len(sample_data[0]))
422     #true index
423     true_idx = range(len(sample_data[0]))
424     #sample index
425     sample_idx = range(len(sample_data[0]))
426     #true index
427     true_idx = range(len(sample_data[0]))
428     #sample index
429     sample_idx = range(len(sample_data[0]))
430     #true index
431     true_idx = range(len(sample_data[0]))
432     #sample index
433     sample_idx = range(len(sample_data[0]))
434     #true index
435     true_idx = range(len(sample_data[0]))
436     #sample index
437     sample_idx = range(len(sample_data[0]))
438     #true index
439     true_idx = range(len(sample_data[0]))
440     #sample index
441     sample_idx = range(len(sample_data[0]))
442     #true index
443     true_idx = range(len(sample_data[0]))
444     #sample index
445     sample_idx = range(len(sample_data[0]))
446     #true index
447     true_idx = range(len(sample_data[0]))
448     #sample index
449     sample_idx = range(len(sample_data[0]))
450     #true index
451     true_idx = range(len(sample_data[0]))
452     #sample index
453     sample_idx = range(len(sample_data[0]))
454     #true index
455     true_idx = range(len(sample_data[0]))
456     #sample index
457     sample_idx = range(len(sample_data[0]))
458     #true index
459     true_idx = range(len(sample_data[0]))
460     #sample index
461     sample_idx = range(len(sample_data[0]))
462     #true index
463     true_idx = range(len(sample_data[0]))
464     #sample index
465     sample_idx = range(len(sample_data[0]))
466     #true index
467     true_idx = range(len(sample_data[0]))
468     #sample index
469     sample_idx = range(len(sample_data[0]))
470     #true index
471     true_idx = range(len(sample_data[0]))
472     #sample index
473     sample_idx = range(len(sample_data[0]))
474     #true index
475     true_idx = range(len(sample_data[0]))
476     #sample index
477     sample_idx = range(len(sample_data[0]))
478     #true index
479     true_idx = range(len(sample_data[0]))
480     #sample index
481     sample_idx = range(len(sample_data[0]))
482     #true index
483     true_idx = range(len(sample_data[0]))
484     #sample index
485     sample_idx = range(len(sample_data[0]))
486     #true index
487     true_idx = range(len(sample_data[0]))
488     #sample index
489     sample_idx = range(len(sample_data[0]))
490     #true index
491     true_idx = range(len(sample_data[0]))
492     #sample index
493     sample_idx = range(len(sample_data[0]))
494     #true index
495     true_idx = range(len(sample_data[0]))
496     #sample index
497     sample_idx = range(len(sample_data[0]))
498     #true index
499     true_idx = range(len(sample_data[0]))
500     #sample index
501     sample_idx = range(len(sample_data[0]))
502     #true index
503     true_idx = range(len(sample_data[0]))
504     #sample index
505     sample_idx = range(len(sample_data[0]))
506     #true index
507     true_idx = range(len(sample_data[0]))
508     #sample index
509     sample_idx = range(len(sample_data[0]))
510     #true index
511     true_idx = range(len(sample_data[0]))
512     #sample index
513     sample_idx = range(len(sample_data[0]))
514     #true index
515     true_idx = range(len(sample_data[0]))
516     #sample index
517     sample_idx = range(len(sample_data[0]))
518     #true index
519     true_idx = range(len(sample_data[0]))
520     #sample index
521     sample_idx = range(len(sample_data[0]))
522     #true index
523     true_idx = range(len(sample_data[0]))
524     #sample index
525     sample_idx = range(len(sample_data[0]))
526     #true index
527     true_idx = range(len(sample_data[0]))
528     #sample index
529     sample_idx = range(len(sample_data[0]))
530     #true index
531     true_idx = range(len(sample_data[0]))
532     #sample index
533     sample_idx = range(len(sample_data[0]))
534     #true index
535     true_idx = range(len(sample_data[0]))
536     #sample index
537     sample_idx = range(len(sample_data[0]))
538     #true index
539     true_idx = range(len(sample_data[0]))
540     #sample index
541     sample_idx = range(len(sample_data[0]))
542     #true index
543     true_idx = range(len(sample_data[0]))
544     #sample index
545     sample_idx = range(len(sample_data[0]))
546     #true index
547     true_idx = range(len(sample_data[0]))
548     #sample index
549     sample_idx = range(len(sample_data[0]))
550     #true index
551     true_idx = range(len(sample_data[0]))
552     #sample index
553     sample_idx = range(len(sample_data[0]))
554     #true index
555     true_idx = range(len(sample_data[0]))
556     #sample index
557     sample_idx = range(len(sample_data[0]))
558     #true index
559     true_idx = range(len(sample_data[0]))
560     #sample index
561     sample_idx = range(len(sample_data[0]))
562     #true index
563     true_idx = range(len(sample_data[0]))
564     #sample index
565     sample_idx = range(len(sample_data[0]))
566     #true index
567     true_idx = range(len(sample_data[0]))
568     #sample index
569     sample_idx = range(len(sample_data[0]))
569

```

```

117 #plot letter_result graph#
118 def _plot_letters(self, idx):
119     self._plot_discreate_sequence(
120         self.input_data[idx],
121         self.fig_title[idx],
122         self.sample_letters[idx],
123         Label=self.sample_letters[idx]
124     )
125 #plot state_result graph#
126 def _plot_states(self, idx):
127     self._plot_discreate_sequence(
128         self.input_data2[idx],
129         self.fig_title[idx],
130         self.sample_states[idx],
131         Label=self.sample_states[idx]
132     )
133 #plot boundary graph#
134 def _plot_label_boundary(self, true_data, title, sample_data, label='u'):
135     boundaries = [[stop for state, (start, stop) in r for r in sample_data]
136                  size = boundaries[0][-1]
137     data = np.zeros((len(sample_data), size))
138     for i, b in enumerate(boundaries):
139         for x in b[:-1]:
140             data[i, x] = 1.0
141     self._plot_discreate_sequence(true_data, title, data, label, ploptops={'cmap': 'Greys'})
142 def _plot_state_boundaries(self, idx):
143     self._plot_label_boundary(
144         self.input_data2[idx],
145         self.fig_title[idx],
146         self.state_ranges[idx],
147         Label=self.sample_states[idx]
148     )
149

```