TABLE OF CONTENTS

CHAPTER I

MATLAB: DIRECT PERCEPTION

1.1 batch_DP_learn.m

Batch learning with DP.

1.2 script_DP_learn.m

Script of DP learning.

1.3 DP_get_para.m

Get the parameters and bounds for optimization from the DP model.

1.4 DP_put_para.m

Put the optimized parameters back into the DP model.

1.5 DP_learn_gmm.m

The real function being called to learn a DP model for one affordance.

1.6 DP_prob.m

The posterior probability of the DP model applied on testing data.

CHAPTER II

MATLAB: BATCH LEARNING SCRIPTS

2.1 all_plot_error_bar.m

Function to select the best model after-training plot the

2.2 all_error_analysis.m

Script to generate the average error for later analysis, load the training results mat file from each directories.

2.3 batch_plot_errorbar.m

Script to plot the errorbar of LL, CLL and detection error on both the training and testing data.

2.4 batch_chain_dis_subset_EM.m

`CA-chain (Dis-subset)` Script to do a number of learning with the CA-chain model.

2.5 batch_chain_gen_EM.m

`CA-chain (Gen)` Script to do a number of learning with the CA-chain model.

2.6 batch_full_gen_EM.m

`CA-full (Gen)` Script to do a number of learning with the CA-full model.

2.7 batch_full_dis_subset_EM.m

`CA-full (Dis-EM)` Script to do a number of learning with the CA-full model.

CHAPTER III

MATLAB: LEARNING SCRIPTS

3.1 script_calc_error.m

`CA_chain ONLY:` Script to calculate CLL on training and testing data. Also calculates separate error and group error.

3.2 script_learn_CA_chain_gen_EM.m

`CA_chain ONLY:` Script to learn a CA-chain model **Generatively** through EM method, the performance is not as good as in discriminative training.

3.3 script_learn_CA_chain_dis_subset_EM.m

`CA_chain ONLY:` Script to learn CA-chain model with subset method, the one that we should use for **discriminative training**.

3.4 script_learn_CA_chain_dis_subset.m

`CA_chain ONLY:` Script to learn CA-chain model with subset method, the one that we should use for **discriminative training**.

3.5 script_learn_CA_full_gen_EM.m

`CA_FULL:` Sctipt to learn a CA-full model generatively through EM method.

3.6 script_learn_CA_full_dis_subset_EM.m

`CA_FULL:` Sctipt to learn a CA-full model discriminativelly through EM method.

3.7 script_run_CA_exp.m

The script to set up the experiment, then calls `learn_category_mixture()` and

`script_learn_CA_chain_gen_EM`

CHAPTER IV

MATLAB: FUNCTIONS FOR LEARNING AND TESTING WITH CA MODEL

4.1 get_CA_err.m

Calculate all kinds of errors from the CA model, works for both chain and full model.

4.2 test_script_singleexp.m

Do similar plotting as in `batch_plot_errorbar`, but only do it for one experiment.

4.3 CA_cond_LL.m

The CLL function value

```
[LL, Err] = CA_cond_LL(CA_model, type, train_X, train_C, train_A, condtype)
```

Parameters listed below:

```
CA_chain:
    pA_X_group_err: group error from argmax of all combinatorics
                    (train_A can't have -1 entries)
    pA_X_group: the real group CLL of P(A|X)
    pA_X_sep: the separation approximation group CLL of P(A|X)        (1)
CA_full:
    pA_X_group_err: no need
    pA_X_group: done
    pA_X_sep: done
```

4.4 CA_joint_LL.m

The LL function value

```
LL = CA_joint_LL(CA_model, type, train_X, train_C, train_A, jointtype)
```

```
CA_chain:
    pXCA: by setting CATENFORCE to 1
    pXA: P(X,A)
CA_full:                                                  (2)
    pXCA: by setting CATENFORCE to 1
    pXA: P(X,A)
```

4.5 CA_posterior.m

Calculate the posterior probability of C and A from the CA_model, if C or A is observed in training data, the posterior is set to 1.0, this is used only for training with EM, not for evaluating LL or CLL.

```
[post_pC,post_pA,pC_X]=CA_posterior(CA_model,type,train_X,train_C,train_A)

CA_chain:  set CATENFORCE to change the options
CA_full:   set CATENFORCE to change the options
```

4.6 count_CA_para.m

Count number of parameters in the CA model.

```
CA_chain:  enforced by C currently
CA_full: no use since we never directly optimize
```

4.7 fun_CA_cond_LL.m

Calculate CLL on training data returns the negative value of *pa_x_ll*. *alldata* is a struct holding all input data. This calls function `CA_cond_LL` using `CA_chain, pA_X_group` option.

```
LLF = fun_CA_cond_LL(X,alldata)
```

4.8 fun_CA_cond_LL_subset.m

Same as `fun_CA_cond_LL.m`, but this uses the subset optimization algorithm. The subset parameter is indicated by *alldata.ind* field. The function reconstruct the full parameter set and calls function `CA_cond_LL` with option of `CA_chain, alldata.condtype` can be group or sep.

```
LLF = fun_CA_cond_LL_subset(X0,alldata)
```

4.9 fun_CA_joint_LL.m

Calculate LL on training data returns the negative value of *xa_ll*. This calls function `CA_joint_LL` using `CA_chain, pXA` option.

4.10 get_CA_para.m

Get parameters and bounds for the CA model

```
[X, A, B, Aeq, Beq, LB, UB] = get_CA_para(CA_chain, train_X)
```

```
CA_chain: implemented
CA_full: no need
```

4.11 put_CA_para.m

Put parameters in the CA model

```
CA_chain = put_CA_para(CA_chain,X)
```

```
CA_chain: implemented
CA_full: no need
```

4.12 learn_category_mixture.m

Learn the Gaussian mixture model. *weight* is used only in weighted GMM learning, otherwise it's empty.

```
[newmix, ll] = learn_category_mixture(gmmdata, weight, ...
    nmixture, maxiter, epsilon, prior_cov, cov_type, model)
```

4.13 learn_CA_chain_para.m

Input is the posterior on C and A. Output is CA-chain model with *probC* and *probA_C* learned. This handles when pC or pA has -1 terms, meaning no posterior info and the data is unknown in training. This only occurs in the first step of EM based training.

```
CA_chain = learn_CA_chain_para(CA_chain, pC, pA, epsilon)
```

CHAPTER V

MATLAB: FILES IN THE COMMON DIRECTORY

5.1 confusion.m

Show the confusion matrix of 7 objects, with captions

```
ret = confusion(d,cap1,cap2)
```

5.2 gmmem_weight.m

Gaussian Mixture Model training with weighted input

```
[mix, options, errlog] = gmmem_weight(mix, x, weight,options)
```

5.3 pca_visualize.m

Visualize the first 3 dimensions of projdata1 and projdata2

5.4 displine(star).m

Display a line separator

5.5 init_CA_model.m

Initialize the CA_model structure

```
CA_model = init_CA_model(type)
```

5.6 load_define.m

Load all the file data to catdef structure, unorganized, but if one file has more than maxnum data, sample to get maxnum data

```
catdef = load_define(fname, maxnum)
```

5.7 *rand_sample.m*

Given an index, sample a number of points, choose if need to sort the output index or not

```
sample_index = rand_sample(index, num, ifsort)
```

CHAPTER VI

MATLAB: FUNCTIONS OF DATA GENERATION AND PCA

6.1 modify_train_data.m

```
[train_A, train_C] = modify_train_data(train_A, train_C, type, opt1, opt2)
```

Modify the training data to hide some variables:

type: fixed or random

opt1: subnum for 'fixed' option

opt2: num of known affordance for 'fixed' option

6.2 generate_traintest_data.m

Generate training and testing data from PCA projected full data set.

```
[train_data, test_data] = generate_traintest_data(affdata,
        projdata1,projdata2,dim1,dim2,numtrain,numtest,ratio)
```

6.3 get_subdim.m

Get a sub dimension from the full data set. *objdata* is a cell array of 7 objects.

```
data = get_subdim(objdata,range)
```

6.4 pca_experiment.m

Do PCA separately for first 'dim1' dimensions and the rest, do PCA separately for first $dim1$ dimensions and the rest. The actual code for PCA is in `pca_projection()`.

```
[featuredim, projdata1,projdata2]=pca_experiement( ...
        objdata, totaldim, dim1, projdim1,projdim2)
```

6.5 pca_projection.m

Actual code performing PCA. *data* and *projdata* are both cell arrays containing data for 7 objects.

```
[projdata, avg, sigma, projV, V, d] = pca_projection(data, projdim)
```

6.6 check_train_AC.m

Check the from of training data C, A, make sure the values are in the correct range, return the number of missing C and A.

```
[unknown_C,unknown_A]=check_train_AC(train_C,train_A,train_X,nclass,nafford)
```

6.7 script_loadproject.m

First script to call to load the project.

CHAPTER VII

MATLAB: NOT USED

7.1 script_learn_CA_chain_dis_direct.m

`CA_chain ONLY:` Script to learn CA-chain model with direct search, very slow sleep.

7.2 CA_All_LL.m (no use)

Get the joint LL for CA-chain model, this calls `CA_joint_LL.m` with option 'pXA'

```
CA_chain = CA_All_LL(CA_chain, type, train_X, train_C, train_A)
```

7.3 script_plot_error.m

Plot CLL, LL, ect, not much use now.