```
% example file to illustrate the use of multi-task Gaussian Process models
%
% by Robert Duerichen
% 31/01/2014

path_gpml = 'E:\OneDrive - hnu.edu.cn\tools\matlabcourse\GPML_matlab\gpml-matlab-v4.
2-2018-06-11';                    % please insert here path of GPML Toolbox

% add folders of MTGP and GPML Toolbox
if ~isunix  % windows system
    addpath(genpath('..\'));
    addpath(genpath(path_gpml));
else        % linux system
    addpath(genpath('../'));
    addpath(genpath(path_gpml));
end

clear; %close all

% please select one of the example cases for shifting the feature space: (1-3)
MTGP_case = 1;
% case1: 3 signals which are phase shifted to each other, modelled by a
%           normal MTGP with k_t = SE covariance function
%
% case2: 3 signals which are phase shifted to each other, modelled by a
%           MTGP which considers a shift between the tasks with k_t = SE covariance
function
%
% case3: Demonstration of an example with two tasks and two dimensions,
%           representing e.g. x and y of a spatial coordinate
%           Even though different labels are known from each task, MTGP
%           can estimate the offset in both dimensions (e.g. template
%           matching)
%
seed=0; rng(seed);

if MTGP_case <= 2
    % generate synthetic data for case 1 and 2
    t = (0:0.05:5)'; % 101×1
    phase_shift_t = pi/4;    pi/4 ≈ 0.7854
```

```
        f = 1;
        translation = 0;
101×1   z1 = sin(2*pi*f*t)+randn(length(t),1)./50;
        z2 = sin(2*pi*f*t+phase_shift_t)+randn(length(t),1)./50;
        z3 = sin(2*pi*f*t+2*(phase_shift_t))+randn(length(t),1)./100;
    else
        % generate synthetic data for case 3
        translation = [0.15 -0.26];
        x = (-0.5:0.1:0.5)'; % 11×1
        y = (-0.5:0.1:0.5)'; % 11×1    y = (-0.4:0.05:0.6)';    % 21×1
        f = 1;
        x_mat = repmat(x,[1 length(y)]); % 11×11 % 11×21
        y_mat = repmat(y,[1 length(x)]); % 11×11 % 11×21
        y_mat = y_mat';  [y_mat, x_mat] = meshgrid(y, x);
        z1 = sin(2*pi*f*x_mat).*sin(2*pi*f*y_mat)+randn(length(x),length(y))/50; % 11×11% 11×21
        z2 = sin(2*pi*f*x_mat).*sin(2*pi*f*y_mat)+randn(length(x),length(y))/50; % 11×11% 11×21
    end
                                              subplot(2,1,1);
                                              plot3(x_mat, y_mat, z1);
    %% initial parameter                      subplot(2,1,2);
    switch MTGP_case                          plot3(x_mat, y_mat, z2);
        case 1
            opt.training_data{1} = 1:100;   % index of know training points of signal 1
            opt.training_data{2} = 1:50;    % index of know training points of signal 2
            opt.training_data{3} = 1:50;    % index of know training points of signal 3
            opt.cov_func = 1;               % select cov. function
            opt.start = 1;                  % start index for prediction
            opt.end = 100;                  % end index for prediction
            z = [z1 z2 z3];                 % 101×3
        case 2
            opt.training_data{1} = 1:100;   % index of know training points of signal 1
            opt.training_data{2} = 1:50;    % index of know training points of signal 2
            opt.training_data{3} = 1:50;    % index of know training points of signal 3
            opt.cov_func = 2;               % select cov. function
            opt.start = 1;                  % start index for prediction
            opt.end = 100;                  % end index for prediction
            z = [z1 z2 z3];
        case 3
            opt.training_data{1} = 1:2:length(x_mat(:));    % index of know training↙
    points of signal 1  % 1×61  % 1×116
            opt.training_data{2} = 20:2:length(x_mat(:));   % index of know training↙
                        % 1×51  % 1×106
```

```
points of signal 2
        opt.cov_func = 3;                    % select cov. function
        opt.start = 1;                       % start index for prediction
        opt.end = length(y_mat(:));          % end index for prediction
                                             % opt.end = 121
        z = [z1(:) z2(:)];                   % z: 121×2      % z: 231×2
        t = [x_mat(:) y_mat(:)];             % t: 121×2      % t: 231×2
end

opt.init_num_opt = 200;          % number of optimization steps
opt.show = 1;                    % show result plot
~~opt.random = 1;rng('shuffle'); % if 1 - hyp for correlation and SE will set randomly~~
opt.num_rep = 1;                 % number of trails for selecting hyp

% init values for hyperparameter ~~(if opt.random ~= 1)~~
opt.se_hyp = 1;
opt.cc_hyp = [1/1 0/1 0 0];      % assumes that all signals are dependent MTGP_covCC_chol_nD
                                 dependent:
                                 1
                                 1 0
                                 0 0 1
                                 independent:
                                 1
                                 0 1
                                 0 0 1
opt.noise_lik = 0.1;
opt.shift_hyp = [0 0];

num_dim = size(opt.training_data,2);3   case3: num_dim = 2
num_cc_hyp = sum(1:size(opt.training_data,2)); sum(1:3) = 6 sum(1:2) = 3

%% generate training and test data with labels
% also ~~substract~~ mean % y_train_mean(cnt_dim) = mean(y(opt.training_data{cnt_dim},cnt_dim));
100×2
61×3
116×3  x_train = [t(opt.training_data{1},:)-repmat(translation,[length(opt.training_data{1}),1]), ...
                                                                 100
                                                    0 %100×1
        ones(length(opt.training_data{1}),1)];
                  100
61×1
116×1
       B = repmat(A, n) returns an array containing n copies of A in the row and column dimensions. The size of B is size(A)*n when A is a matrix.
100×1 z_train = z(opt.training_data{1},1);

100×2 x_test = [t(opt.start:opt.end,:) ones(opt.end-opt.start+1,1)];    第1组训练样本
100×1 z_test = z(opt.start:opt.end,1);
121×3; 121×1
231×3; 231×1 % 11×21=231                                                  串联所有组的训练样本
for cnt_dim = 2:num_dim num_dim = 3 %不同数据的组数
        x_train = [x_train(:,:);...
             t(opt.training_data{cnt_dim},:) ...
             ones(length(opt.training_data{cnt_dim}),1)*cnt_dim];

        z_train = [z_train; z(opt.training_data{cnt_dim},cnt_dim)];
                                                    z % 101×3
                                                    opt.training_data{2} = 1:50;
                                                    opt.training_data{3} = 1:50;
```

cnt_dim = 2
x_train(:, :) 100×2
x_train % 150×2
z_train % 150×2
cnt_dim = 2
x_train(:, :) 61×3 116×3
x_train % 112×3 222×3
z_train % 112×1 222×1
x_test % 242×3 462×3
z_test % 242×1 462×1

```
        x_test = [x_test(:,:);...
            t(opt.start:opt.end,:) ones(opt.end-opt.start+1,1)*cnt_dim];
        z_test = [z_test; z(opt.start:opt.end,cnt_dim)];
end

%% init covariance functions and hyperparameters
switch opt.cov_func % feval(covfunc{:}, feval('MTGP_covCC_chol_nD')); feval('MTGP_covCC_chol_nD')
    case 1
        disp('Covariance Function: K = CC(l) x (SE_U(t))');
        covfunc = {'MTGP_covProd',{'MTGP_covCC_chol_nD','MTGP_covSEisoU'}};
        hyp.cov(1:num_cc_hyp) = opt.cc_hyp(1:num_cc_hyp);
        hyp.cov(num_cc_hyp+1) = log(sqrt(opt.se_hyp));

    case 2
        disp('Covariance Function: K = CC(l) x (SE_U_shift(t))');
        covfunc = {'MTGP_covProd',{'MTGP_covCC_chol_nD','MTGP_covSEisoU_shift'}};

        hyp.cov(1:num_cc_hyp) = opt.cc_hyp(1:num_cc_hyp);
        hyp.cov(num_cc_hyp+1) = log(opt.se_hyp);
        hyp.cov(num_cc_hyp+2) = opt.shift_hyp(1);
        hyp.cov(num_cc_hyp+3) = opt.shift_hyp(2);
    case 3
        disp('Covariance Function: K = CC(l) x (SE_U_shift_nD(t))');
        covfunc = {'MTGP_covProd',{'MTGP_covCC_chol_nD','MTGP_covSEisoU_shift_nD'}};

        hyp.cov(1:num_cc_hyp) = opt.cc_hyp(1:num_cc_hyp);
        hyp.cov(num_cc_hyp+1) = log(opt.se_hyp);
        hyp.cov(num_cc_hyp+2) = opt.shift_hyp(1);
        hyp.cov(num_cc_hyp+3) = opt.shift_hyp(2);
end

% likelihood function
likfunc = @likGauss;
hyp.lik = log(opt.noise_lik);

% optimize hyperparameters
[results.hyp] = minimize(hyp, @MTGP, -opt.init_num_opt, @MTGP_infExact, [], covfunc,↙
likfunc, x_train,z_train);
results.nlml = MTGP(results.hyp, @MTGP_infExact, [], covfunc, likfunc, x_train,↙
                          ln p(z|x)
```

```
z_train);

%% perform prediction
[results.m, results.s2, fmu, fs2, results.p] = MTGP(results.hyp, @MTGP_infExact, [],
covfunc, likfunc, x_train, z_train, x_test, z_test);
```

300×1
462×1

```
% reshape of results
results.m = reshape(results.m,[opt.end-opt.start+1 num_dim]);
results.s2 = reshape(results.s2,[opt.end-opt.start+1 num_dim]);
results.p = exp(reshape(results.p,[opt.end-opt.start+1 num_dim]));
```

100×3
231×2

```
%% compute RMSE for training and test data for each dimension
for cnt_dim = 1:num_dim
    index_test_data = opt.start:opt.end;

    index_test_data(ismember(index_test_data,opt.training_data{cnt_dim})) = [];

    results.rmse_test(cnt_dim) = rms(results.m(index_test_data-opt.start+1,cnt_dim)-
...
        z(index_test_data,cnt_dim));

    results.rmse_train(cnt_dim) = rms(results.m(opt.training_data{cnt_dim}-opt.
start+1,cnt_dim)-...
        z(opt.training_data{cnt_dim},cnt_dim));
        z(opt.training_data{cnt_dim}+opt.start-1, cnt_dim)
end

% compute resulting K_f matrix
vec_dim = 1:num_dim;
L = zeros(num_dim,num_dim);
for cnt_dim = 1:num_dim
    L(cnt_dim,1:vec_dim(cnt_dim)) = [results.hyp.cov(sum(vec_dim(1:cnt_dim-1))+1:sum
(vec_dim(1:cnt_dim-1))+vec_dim(cnt_dim))];
end

results.K_f =  L*L';

% print results on console:
disp('Estimated cross correlation covariance K_f:');
results.K_f
```

```
if opt.cov_func == 2
    disp('Estimated time shift: ');
    disp(['\Delta t_1: ',num2str(results.hyp.cov(end-1))]);
    disp(['\Delta t_2: ',num2str(results.hyp.cov(end))]);
elseif opt.cov_func == 3
    disp('Estimated shift: ');
    disp([' dim1: ',num2str(results.hyp.cov(end-1)), ' (true shift: ',num2str(-↙
translation(1)),')']);
    disp([' dim2: ',num2str(results.hyp.cov(end)), ' (true shift: ',num2str(-↙
translation(2)),')']);
end

%% plot basic results
if opt.show == 1
    if opt.cov_func <= 2
    h=figure('Position',[1 1 1400 800]);
        for cnt_dim = 1:num_dim
            % plot prediction results
            subplot(2,num_dim,cnt_dim);

            min_val = min(results.m(:,cnt_dim))-abs(min(results.s2(:,cnt_dim)));
            max_val = max(results.m(:,cnt_dim))+abs(max(results.s2(:,cnt_dim)));

            hTlines = plot([t(opt.training_data{cnt_dim}) t(opt.training_data↙
{cnt_dim})]',...
                [ min_val max_val]','Color',[0.85 0.85 0.5]);
            hTGroup = hggroup;
            set(hTlines,'Parent',hTGroup);
            set(get(get(hTGroup,'Annotation'),'LegendInformation'),...
                'IconDisplayStyle','on');
            hold on;
            f = [results.m(:,cnt_dim)+2*sqrt(results.s2(:,cnt_dim)); flip(results.m↙
(:,cnt_dim)-2*sqrt(results.s2(:,cnt_dim)),1)];
            fill([t(opt.start:opt.end); flip(t(opt.start:opt.end),1)], f, [0.7 0.7↙
0.7],'EdgeColor','none')

            % plot mean org signal
            plot(t(opt.start:opt.end),z(opt.start:opt.end,cnt_dim),'b');
```

```
            % plot mean predicted signal
            plot(t(opt.start:opt.end),results.m(1:opt.end-opt.start+1,cnt_dim),'r');

            if cnt_dim == 1
                legend('training data','95% conf. int.','org. values','pred. ↙
values','Orientation','horizontal','Location',[0.45 0.49 0.15 0.04]);
            end
            axis tight

            title(['S',num2str(cnt_dim),': RMSE_{train}: ',num2str(results. ↙
rmse_train(cnt_dim)),...
                ' - RMSE_{test}: ',num2str(results.rmse_test(cnt_dim))]);
            xlabel('time [s]');
            ylabel('amplitude y [mm]');


            subplot(2,num_dim,num_dim+cnt_dim);
            min_val = min(results.p(:,cnt_dim));
            max_val = max(results.p(:,cnt_dim));
            plot([t(opt.training_data{cnt_dim}) t(opt.training_data{cnt_dim})]',...
                [ min_val max_val]','Color',[0.85 0.85 0.5]);
            hold on
            plot(t(opt.start:opt.end),results.p(1:opt.end-opt.start+1,cnt_dim));

            title('probability p');
            axis tight
            xlabel('time [s]');
            ylabel('p');
        end
    else
        figure
        subplot(2,1,1);
        xlim([-0.8 0.8]);
        ylim([-0.8 0.8]);
        zlim([-1.1 1.1]);
        hold on
        plot3(x_train(x_train(:,3)==1,1),x_train(x_train(:,3)==1,2),z_train(x_train↙
(:,3)==1,1),'k*','Markersize',8,'Linewidth',2);
        surface(x-translation(1),y-translation(2),z1);
        xlabel('Dim 1');
```

```
        ylabel('Dim 2');
        title('Task 1');
        hold on;
        legend('training points task 1')
        subplot(2,1,2);

        xlim([-0.8 0.8]);
        ylim([-0.8 0.8]);
        zlim([-1.1 1.1]);
        hold on
        plot3(x_train(x_train(:,3)==2,1)',x_train(x_train(:,3)==2,2)',z_train↙
(x_train(:,3)==2,1)','k*','Markersize',10,'Linewidth',2);
        surface(x,y,z2');
        xlabel('Dim 1');
        ylabel('Dim 2');
        title(['Task 2 - shifted by dim1: ',num2str(results.hyp.cov(end-1)),' (true:↙
',num2str(-translation(1)),') / dim2: ',num2str(results.hyp.cov(end)),' (true: ',↙
num2str(-translation(2)),')']);
        legend('training points task 2')
    end
end
```