Sai Satya Charan Malladi
charanm@umich.edu

Project 4

AEROSP 567
Dec 16, 2021

Code

Pre-Process Data

```
% Sai Satya Charan Malladi
% AEROSP 567 Fall 21
% Final Project

% data_loader.m
% file to pre-process data

%% Begin

% load the walking data of subject 02
load('02_02_moc.mat')
data_dense = wsMoc.Dof;

% sample every 4 frames to generate sparse data
sample_rate = 4;
data_sparse = data_dense(:,1:sample_rate:length(data_dense));

% subtract mean from the sparse data
data_sparse = data_sparse - mean(data_sparse,2);
save('data_walking_sparse.mat','data_sparse')


% load the running data of subject 02
load('02_03_moc.mat')
data_dense = wsMoc.Dof;

% sample every 4 frames to generate sparse data
sample_rate = 4;
data_sparse = data_dense(:,1:sample_rate:length(data_dense));

% subtract mean from the sparse data
data_sparse = data_sparse - mean(data_sparse,2);
save('data_running_sparse.mat','data_sparse')


% load the jumping data of subject 02
load('02_04_moc.mat')
data_dense = wsMoc.Dof;

% sample every 4 frames to generate sparse data
sample_rate = 4;
data_sparse = data_dense(:,1:sample_rate:length(data_dense));

% subtract mean from the sparse data
data_sparse = data_sparse - mean(data_sparse,2);
save('data_jumping_sparse.mat','data_sparse')
```

RBF Kernel

```
function val = rbf_kernel(x,xp,beta1,beta2,beta3)
% rbf kernel

    val =  beta1*exp(-beta2/2*norm(x-xp)^2);
    if x == xp
        val = val + beta3;
    end

end
```

Linear+RBF kernel

```
function val = linear_rbf_kernel(x,xp,alpha1,alpha2,alpha3,alpha4)
% linear + rbf kernel

    val =  alpha1*exp(-alpha2/2*norm(x-xp)^2) + alpha3*x*xp';
    if x == xp
        val = val + alpha4;
    end

end
```

GPLVM

```
function cost = gplvm_objective(z,Y,Xdim,Ydim,W,betalen,kernel_Y)
%

% unpack z
X = reshape(z(1:end-betalen,1),Xdim);
beta = z(end-betalen+1:end);

%
time_steps = Xdim(1);
obs_dim = Ydim(2);
K_Y = zeros(time_steps);

for ii = 1:time_steps
    for jj = 1:time_steps
        K_Y(ii,jj) = kernel_Y(X(ii,:),X(jj,:),beta(1),beta(2),beta(3));
    end
end

% K_Y = K_Y + beta(3)*eye(time_steps);
det_term = obs_dim/2*logdet(K_Y);
trace_term = 1/2*trace(K_Y\Y*Y');
log_likelihood = det_term + trace_term + sum(log(beta));

cost = (log_likelihood);

```

```
end
```

GPDM

```
function cost = gpdm_objective(z,Y,Xdim,Ydim,W,alphalen,betalen,...
                                    kernel_X,kernel_Y)
%

% unpack z
X = reshape(z(1:end-(alphalen+betalen),1),Xdim);
alpha = z(end-(alphalen+betalen)+1:  end-betalen);
beta = z(end-betalen+1:end);

%
Xout = X(2:end,:);
time_steps = Xdim(1);
lnt_dim = Xdim(2);
obs_dim = Ydim(2);

K_Y = zeros(time_steps);
K_X = zeros(time_steps-1);

for ii = 1:time_steps
    for jj = 1:time_steps
        K_Y(ii,jj) = kernel_Y(X(ii,:),X(jj,:),beta(1),beta(2),beta(3));
        if ii ≠ time_steps && jj ≠ time_steps
            K_X(ii,jj) = ...
                kernel_X(X(ii,:),X(jj,:),alpha(1),alpha(2),alpha(3),alpha(4));
        end
    end
end

% K_Y = K_Y + beta(3)*eye(time_steps);
det_term = obs_dim/2*logdet(K_Y);
trace_term = 1/2*trace(K_Y\Y*W^2*Y');
log_likelihood = det_term + trace_term;

% K_X = K_X + alpha(4)*eye(time_steps-1);
det_term = lnt_dim/2*logdet(K_X);
trace_term = 1/2*trace(K_X\Xout*Xout');
log_prior =  det_term + trace_term + sum(log(alpha)) + sum(log(beta));

cost = (log_prior+log_likelihood);

end
```

Main

```
% Sai Satya Charan Malladi
% AEROSP 567 Fall 21
% Final Project

% main.m
% GPLVM and GDPM on the data

clc; clear all; close all;

%% Begin

X_pca = cell(3,1);
X_gpdm = cell(3,1);
alpha_gpdm = cell(3,1);
beta_gpdm = cell(3,1);
X_gplvm = cell(3,1);
beta_gplvm = cell(3,1);

% % kernels
kernel_X = @(x,xp,alpha1,alpha2,alpha3,alpha4) ...
    linear_rbf_kernel(x,xp,alpha1,alpha2,alpha3,alpha4);
kernel_Y = @(x,xp,beta1,beta2,beta3) rbf_kernel(x,xp,beta1,beta2,beta3);

for ii = 1:3
    %%%% load data
    switch ii
        case 1
            load('data_walking_sparse');
            motion = 'walking';
        case 2
            load('data_running_sparse');
            motion = 'running';
        case 3
            load('data_jumping_sparse');
            motion = 'jumping';
    end

    %%%% PCA Initialization
    latent_dim = 3;
    coeff_pca = pca(data_sparse);
    Y = data_sparse';
    Ydim = size(Y);
    vararr = var(Y);
    vararr(vararr == 0) = 1e-15;
    W = diag(1./sqrt(vararr));
    % W = diag(sqrt(vararr));

    X_pca{ii} = coeff_pca(:,1:3);
    X0 = X_pca{ii};
    Xdim = size(X0);

    % initialize
    alpha0 = [0.9; 1; 0.1; 1/exp(1)];
    alphalen = length(alpha0);
    beta0 = [1; 1; 1/exp(1)];
    betalen = length(beta0);

    %%%% GDPM
    % inital guess
    z0 = [X0(:); alpha0; beta0];
    % test
    test_gdpm = ...
       gpdm_objective(z0,Y,Xdim,Ydim,W,alphalen,betalen,kernel_X,kernel_Y);
    options = ...
       optimoptions('fminunc','Algorithm','quasi-newton','Display','iter','MaxFunEvals',15
%     options = optimoptions('fminunc','Display','iter','MaxFunEvals',15e4);
    z_gpdm = fminunc(@(z) gpdm_objective(z,Y,Xdim,Ydim,W,alphalen,betalen,...
                                                kernel_X,kernel_Y),z0,options);
    X_gpdm{ii} = reshape(z_gpdm(1:end-(alphalen+betalen),1),Xdim);
    alpha_gpdm{ii} = z_gpdm(end-(alphalen+betalen)+1:  end-betalen);
    beta_gpdm{ii} = z_gpdm(end-betalen+1:end);

    %%%% GPLVM
    % inital guess
    z0 = [X0(:); beta0];
    % test
    test_gplvm = gplvm_objective(z0,Y,Xdim,Ydim,W,betalen,kernel_Y);
    options = ...
       optimoptions('fminunc','Algorithm','quasi-newton','Display','iter','MaxFunEvals',15
%     options = optimoptions('fminunc','Display','iter','MaxFunEvals',15e4);
    z_gplvm = fminunc(@(z) ...
       gplvm_objective(z,Y,Xdim,Ydim,W,betalen,kernel_Y),z0,options);
    X_gplvm{ii} = reshape(z_gplvm(1:end-betalen,1),Xdim);
    beta_gplvm{ii} = z_gplvm(end-betalen+1:end);

end


% %% plot setup
load('optim_result.mat')
figa = figure('Position', get(0, 'Screensize'));
figatile = tiledlayout(2,3,'TileSpacing','tight','Padding','tight');

% plot
for ii = 1:3
    switch ii
        case 1
            motion = 'walking';
        case 2
            motion = 'running';
        case 3
            motion = 'jumping';
    end

    figure(figa)
    nexttile(ii)
    plot3(X_gpdm{ii}(:,1), X_gpdm{ii}(:,2), ...
       X_gpdm{ii}(:,3),'ro-','LineWidth',2,'DisplayName','GPDM')
%     hold on
%     plot3(X_gplvm{ii}(:,1), X_gplvm{ii}(:,2), ...
   X_gplvm{ii}(:,3),'bo-','LineWidth',2,'DisplayName','GPLVM')
%     plot3(X_pca{ii}(:,1), X_pca{ii}(:,2), ...
   X_pca{ii}(:,3),'ko-','LineWidth',2,'DisplayName','PCA')
```

```
    set(gca,'FontSize',20)
    set(gca,'TickLabelInterpreter','latex');
    xlabel('latent-dim-1','fontsize',20,'interpreter','latex')
    ylabel('latent-dim-2','fontsize',20,'interpreter','latex')
    zlabel('latent-dim-3','fontsize',20,'interpreter','latex')
    title(motion,'fontsize',25,'interpreter','latex')
    legend('location','best','fontsize',20,'interpreter','latex')
    grid on

    figure(figa)
    nexttile(ii+3)
%     plot3(X_gpdm{ii}(:,1), X_gpdm{ii}(:,2), ...
   X_gpdm{ii}(:,3),'ro-','LineWidth',2,'DisplayName','GPDM')
%     hold on
    plot3(X_gplvm{ii}(:,1), X_gplvm{ii}(:,2), ...
       X_gplvm{ii}(:,3),'bo-','LineWidth',2,'DisplayName','GPLVM')
%     plot3(X_pca{ii}(:,1), X_pca{ii}(:,2), ...
   X_pca{ii}(:,3),'ko-','LineWidth',2,'DisplayName','PCA')
    set(gca,'FontSize',20)
    set(gca,'TickLabelInterpreter','latex');
    xlabel('latent-dim-1','fontsize',20,'interpreter','latex')
    ylabel('latent-dim-2','fontsize',20,'interpreter','latex')
    zlabel('latent-dim-3','fontsize',20,'interpreter','latex')
    title(motion,'fontsize',25,'interpreter','latex')
    legend('location','best','fontsize',20,'interpreter','latex')
    grid on

 end


%% mean prediction sequences
load('optim_result.mat')

% just for walking
X = X_gpdm{1};
alpha = alpha_gpdm{1};

%
Xout = X(2:end,:);
Xdim = size(X);
time_steps = Xdim(1);
lnt_dim = Xdim(2);

% compute K_X
K_X = zeros(time_steps-1);
for ii = 1:time_steps
    for jj = 1:time_steps
        if ii ≠ time_steps && jj ≠ time_steps
            K_X(ii,jj) = ...
               kernel_X(X(ii,:),X(jj,:),alpha(1),alpha(2),alpha(3),alpha(4));
        end
    end
end


% sample sequences
```

```
num_sequences = 25;
sample_sequence = zeros(time_steps,lnt_dim,num_sequences);


for ii = 1:num_sequences
    %
    x_tt =  X(1,:);
    sample_sequence(1,:,ii) = x_tt;

    % get k_x
    for tt = 2:time_steps
        k_x = zeros(time_steps-1,1);
        for jj = 1:time_steps-1
            k_x(jj,1) = ...
               kernel_X(x_tt,X(jj,:),alpha(1),alpha(2),alpha(3),alpha(4));
        end

         % calculate mu and sigma
        mu_tt = Xout'/K_X*k_x;
        sigma2_tt = kernel_X(x_tt,x_tt,alpha(1),alpha(2),alpha(3),alpha(4)) ...
           - k_x'/K_X*k_x;

        % next state
        x_tt = (mu_tt + sqrt(sigma2_tt)*randn(lnt_dim,1))';
        sample_sequence(tt,:,ii) = x_tt;
    end
end

% plot
figb = figure('Position', get(0, 'Screensize'));

% plot
figure(figb)
plot3(X(:,1), X(:,2), X(:,3),'ro-','LineWidth',2,'DisplayName','GPDM')
hold on
for ii = 1:num_sequences
    plot3(sample_sequence(:,1,ii), sample_sequence(:,2,ii), ...
       sample_sequence(:,3,ii),'go-','LineWidth',1,'HandleVisibility','off')
end
set(gca,'FontSize',30)
set(gca,'TickLabelInterpreter','latex');
xlabel('latent-dim-1','fontsize',30,'interpreter','latex')
ylabel('latent-dim-2','fontsize',30,'interpreter','latex')
zlabel('latent-dim-3','fontsize',30,'interpreter','latex')
title('25 sample trajectories using mean-prediction ...
   [Walking]','fontsize',30,'interpreter','latex')
legend('location','best','fontsize',30,'interpreter','latex')
grid on
```