# demoGPRModule

This script demonstrates the use of gaussianProcessRegression module. The demonstartions shows single steps of use from initialization to optimization. For use generate a training and testdataset with corresponding position. So the sensor model is built on none diverging coordinates.

## Requirements

- Other m-files required: gaussianProcessRegression module files
- Subfunctions: none
- MAT-files required: data/config.mat, corresponding Training and Test dataset

## See Also

- gaussianProcessRegression
- initGPR
- tuneGPR
- optimGPR.html
- generateConfigMat

Created on February 25. 2021 by Tobias Wulf. Copyright Tobias Wulf 2021.

## Start Script, Load Config and Read in Datasets

```
clc;
disp('Start GPR module demonstration ...');
clearvars;
close all;

disp('Load config ...');
load config.mat PathVariables GPROptions;

disp('Search for datasets ...');
TrainFiles = dir(fullfile(PathVariables.trainingDataPath, 'Training*.mat'));
TestFiles = dir(fullfile(PathVariables.testDataPath, 'Test*.mat'));
assert(~isempty(TrainFiles), 'No training datasets found.');
assert(~isempty(TestFiles), 'No test datasets found.');

disp('Load first found datasets ...');
try
    TrainDS = load(fullfile(TrainFiles(1).folder, TrainFiles(1).name));
    TestDS = load(fullfile(TestFiles(1).folder, TestFiles(1).name));

catch ME
    rethrow(ME)
end

disp('Check dataset coordinates corresponds ...');
assert(all(TrainDS.Data.X == TestDS.Data.X, 'all'), 'Wrong X grid.');
assert(all(TrainDS.Data.Y == TestDS.Data.Y, 'all'), 'Wrong Y grid.');
assert(all(TrainDS.Data.Z == TestDS.Data.Z, 'all'), 'Wrong Z grid.');
```

## Compute Means of Test Dataset as Comparing Root

Compute mean sinoids and angular error by raw dataset values. offcos0 - cosine offset offsin0 - sine offset fcos0 - offset free and normed mean cosine from raw test values fsin0 - offset free and normed mean sine from raw test values frad0 - radius by mean sinoids fang0 - angles by mean sinoids compute with angle function in predDS AAED0 - Absolute Angular Error in Degrees by mean sinoids

```
offcos0 = mean2(TestDS.Data.Vcos);
offsin0 = mean2(TestDS.Data.Vsin);
fcos0 = zeros(TestDS.Info.UseOptions.nAngles, 1);
fsin0 = zeros(TestDS.Info.UseOptions.nAngles, 1);
for n = 1:TestDS.Info.UseOptions.nAngles
    fcos0(n) = mean2(TestDS.Data.Vcos(:,:,n));
    fsin0(n) = mean2(TestDS.Data.Vsin(:,:,n));
end
fcos0 = fcos0 - offcos0;
fsin0 = fsin0 - offsin0;
frad0 = sqrt(fcos0.^2 + fsin0.^2);
fang0 = sinoids2angles(fsin0, fcos0, frad0);
AAED0 = abs(TestDS.Data.angles' - fang0 * 180 / pi);
```

## Create GPR Model for Demonstartion

Create three GPR Modles by the same base configuration to compare bare initilized modle with and optimized generated modle with same root of configuration. Mdl1 - optimized modle generated by configuration settings enable free free tuning of variance and length scale by changing the theta(1) to not equal 1 and widining the parameter bounds.

```
disp('Create GPR modles ...');
Mdl1 = optimGPR(TrainDS, TestDS, GPROptions, 0);
```

## Prediction on Test Dataset

Predict sinoids and angles on test dataset for each created GPR modle. fang1 - computed angle by predicted sinoids frad1 - computed radius by predicted sinoids fcos1 - predicted cosine fsin1 - predicted sine fcov1 - predictive variance s1 - standard deviation of prediction ciang1 - 95% confidence interval for angles cirad1 - 95% confidence interval for radius

```
[fang1, frad1, fcos1, fsin1, fcov1, s1, ciang1, cirad1] = predDS(Mdl1, TestDS);
```

## Compute Losses and Errors on Test Dataset

Compute the loss and error on test dataset for each created GPR modle. AAED1 - Absolute Angular Error in Degrees SLLA1 - Squared Log Loss Angular SLLR1 - Squared Log Loss Radius SEA1 - Squared Error Angular SER1 - Squared Error Radius SEC1 - Squared Error Cosine SES1 - Squared Error Sine

```
[AAED1, SLLA1, SLLR1, SEA1, SER1, SEC1, SES1] = lossDS(Mdl1, TestDS);
```

## Plot Area and Expand Model Results

Plot demo results in modle parameter view to show characteristics of covariance functions and modle generalization. Show full rotation on test dataset with angle error, predicted sinoids and confidence intervals.

```
% create general plot scalses and title
angles = TestDS.Data.angles';
ticks = Mdl1.Angles;
titleStr = "Kernel %s: $\\sigma_f = %1.2f$, $\\sigma_l = %1.2f$," + ...
    " $\\sigma_n^2 = %1.2e$, $N = %d$\n" +...
    "$%d \\times %d$ Sensor-Array, Posistion: $(%1.1f,%1.1f,-%1.1f)$ mm," + ...
    " Magnet Tilt: $%2.1f^\\circ$";
titleStr = sprintf(titleStr, ...
```

```matlab
    Mdl1.kernel, Mdl1.theta(1), Mdl1.theta(2), Mdl1.s2n, ...
    Mdl1.N, Mdl1.D, Mdl1.D, ...
    TestDS.Info.UseOptions.xPos, ...
    TestDS.Info.UseOptions.yPos, ...
    TestDS.Info.UseOptions.zPos, ...
    TestDS.Info.UseOptions.tilt);

% create figure for model view
figure('Name', Mdl1.kernel, 'Units', 'normalize', 'OuterPosition', [0 0 1 1]);
t=tiledlayout(2,2);
title(t, titleStr, 'Interpreter', 'latex', 'FontSize', 24);

% plot covariance slice for first covariance sample
nexttile;
p1 = plot(Mdl1.Ky, 'Color', [0.8 0.8 0.8]);
hold on;
p2 = yline(mean2(Mdl1.Ky), 'Color', '#0072BD', 'LineWidth', 2.5);
p3 = plot(1:Mdl1.N, Mdl1.Ky(1,:), 'kx-.');
xlim([1, Mdl1.N]);
ylim([min(Mdl1.Ky, [], 'all'), max(Mdl1.Ky, [], 'all')]);
legend([p1(1), p2, p3], {'$i \ne 0$','$\mu(K)$','$i = 0$'});
xlabel('$n$ Samples');
ylabel('Autocorrelation Coeff.');
title('a) $K$-Matrix $i$-th Row');

% plot covariance matrix
nexttile([2, 1]);
colormap('jet');
imagesc(Mdl1.Ky);
axis square;
colorbar;
xlabel('$j$');
ylabel('i');
title(sprintf('b) $K$-Matrix $%d \\times %d$ Samples', Mdl1.N, Mdl1.N))

% plot modle adjust as squared logarithmic loss for angles and radius
nexttile;
plot(angles, SLLA1, 'x-.');
hold on;
plot(angles, SLLR1, 'x-.');
xlim([0 360]);
legend({'$SLLA$' , '$SLLR$'});
xlabel('$\alpha$ in $^\circ$');
ylabel('$SLL$');
title(sprintf('c) $MSLLA = %1.2f$, $MSLLR = %1.2f$', mean(SLLA1), mean(SLLR1)));

% create figure for rotation results of test dataset
figure('Name', 'Rotation and Errors', 'Units', 'normalize', ...
    'OuterPosition', [0 0 1 1]);
t = tiledlayout(2,2);
title(t, titleStr, 'Interpreter', 'latex', 'FontSize', 24);

% plot circle with gpr reuslts and pure mean results
nexttile;
polarplot(fang0, frad0, 'LineWidth', 3.5);
hold on;
polarplot(fang1, frad1, 'LineWidth', 3.5);
polarscatter(Mdl1.Angles * pi / 180, 1.2 *ones(Mdl1.N, 1), 52, [0.8 0.8 0.8], ...
    'filled', 'MarkerEdgeColor', 'k', 'LineWidth', 1.5);
legend({'Mean', 'GPR', 'Ref. $\alpha$'});
rticklabels(["", "0.5", "1"]);
title('a) Rotation along $Z$-Axis in $^\circ$')
```

```matlab
% plot predicted, ideal and none treated sinoids
nexttile;
p1 = plot(angles, cosd(angles), 'k-.', 'LineWidth', 6.5);
hold on;
plot(angles, sind(angles), 'k-.', 'LineWidth', 6.5);
p2 = plot(angles, fcos0, 'Color', '#0072BD');
plot(angles, fsin0, 'Color', '#0072BD');
p3 = plot(angles, fcos1, 'Color', '#D95319');
plot(angles, fsin1, 'Color', '#D95319');
xlim([0 360]);
ylim([-1.1 1.1]);
xlabel('$\alpha$ in $^\circ$');
legend([p1 p2 p3], {'Ideal', 'Mean', 'GPR'})
title('b) Sine and Cosine')

% plot absolut angle errors of gpr and mean results
nexttile;
plot(angles, AAED0);
hold on;
plot(angles, AAED1);
yline(mean(AAED1), 'k-.', 'LineWidth', 3.5)
[~, idx] = max(AAED1);
scatter(angles(idx), max(AAED1), 52, [0.8 0.8 0.8], ...
    'filled', 'MarkerEdgeColor', 'k', 'LineWidth', 1.5)
ylim([0 4]);
xlim([0 360]);
xlabel('$\alpha$ in $^\circ$');
ylabel('$\epsilon_{abs}$ in $^\circ$');
legend({'Mean', 'GPR', '$\mu(\epsilon_{abs})$', '$\max\epsilon_{abs}$'});
tstr = 'c) $\\mu(\\epsilon_{abs}) = %1.2f$, $\\max\\epsilon_{abs} = %1.2f$';
tstr = sprintf(tstr, mean(AAED1), max(AAED1));
title(tstr);

% plot 95 percent confidence intervals for angles and radius
nexttile;
yyaxis left;
plot(angles, (ciang1-fang1) * 180/pi, '-', 'Color', '#0072BD');
ylabel('$CIA_{95\%} - \alpha$ in $^\circ$');
yyaxis right;
plot(angles, (cirad1-frad1), '-', 'Color', '#D95319');
ylabel('$CIR_{95\%} - r$');
xlim([0 360]);
xlabel('$\alpha$ in $^\circ$');
title('d) Centerd $95\%$ GPR Confidence Intervalls')
```

*Published with MATLAB® R2020b*