investigateKernelParameters

Sweep kernel parameters against inner tuning criteria which is built by the logarithmic likelihoods for cosine and sine fit on training datasets.

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

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.');
```

Create GPR Model for Investigation

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

Sweep Title with Model Parameters

```
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, ...
    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);
```

Execute Parameter Sweep with Constant Noise

```
nEval = 300;
disp('Sweep kernel parameters with constant noise ...');
sweepKernelWithConstNoise(Mdl1, nEval, titleStr, PathVariables)
```

Execute Parameter Sweep with Constant Variance

```
nEval = 300;
disp('Sweep kernel parameters with constant variance ...');
sweepKernelWithConstVariance(Mdl1, nEval, titleStr, PathVariables)
```

Execute Parameter Sweep with Constant Lengthscale

```
nEval = 300;
disp('Sweep kernel parameters with constant lengthscale ...');
sweepKernelWithConstLengthscale(Mdl1, nEval, titleStr, PathVariables)
```

Sweep Kernel Parameters vs. Likelihood Criteria with Constant Noise

```
function sweepKernelWithConstNoise(Mdl, nEval, titleStr, PathVariables)

    % create sweep parameters for sweeping theta to given modle
    s2f = linspace(Mdl.s2fBounds(1) * 0.1, Mdl.s2fBounds(2) * 10, nEval);
    sl = linspace(Mdl.slBounds(1) * 0.1, Mdl.slBounds(2) * 10, nEval);
    [sl, s2f] = meshgrid(sl, s2f);

    % allocate memory for inner tuning criteria, combined likelihoods for cosine
    % and sine fit on trainings data
    RLI = zeros(nEval, nEval);

    % run sweep in multiprocess pool to gain speed
    parfor i = 1:nEval
        for j = 1:nEval
            % compute sweep with tuning criteria of inner GPR optimization of
            % tuning GPR kernel parameters
            RLI(i,j) = computeTuneCriteria([s2f(i,j) sl(i,j)], Mdl);
        end
    end

    % plot results in countour plot
    fig = figure('Name', 'Sweep Kernel Parameters with Constant Noise', ...
```

```
        'Units', 'normalize', 'OuterPosition', [0 0 1 1]);

    % plot sweep with log axis
    contourf(s1, s2f, RLI, linspace(min(RLI, [], 'all') + 1, 1, 10), ...
        'LineWidth', 1.5);
    set(gca, 'YScale', 'log')
    set(gca, 'XScale', 'log')
    hold on;
    grid on;

    % plot bounds origin model parameters
    p1 = yline(Mdl.s2fBounds(1), 'k-.', 'LineWidth', 2.5);
    yline(Mdl.s2fBounds(2), 'k-.', 'LineWidth', 2.5);
    yline(Mdl.theta(1), 'k', 'LineWidth', 2.5);
    xline(Mdl.s1Bounds(1), 'k-.', 'LineWidth', 2.5);
    xline(Mdl.s1Bounds(2), 'k-.', 'LineWidth', 2.5);
    xline(Mdl.theta(2), 'k', 'LineWidth', 2.5);

    % plot fmincon search area
    p2 = patch( ...
        [Mdl.s1Bounds(1), Mdl.s1Bounds(2), ...
         Mdl.s1Bounds(2), Mdl.s1Bounds(1)], ...
        [Mdl.s2fBounds(1) Mdl.s2fBounds(1), ...
         Mdl.s2fBounds(2) Mdl.s2fBounds(2)],...
        [0.8 0.8 0.8], 'FaceAlpha', 0.7);

    % plot argmin fmincon result
    p3 = scatter(Mdl.theta(2), Mdl.theta(1), 60, [0.8 0.8 0.8], ...
        'filled', 'MarkerEdgeColor', 'k', 'LineWidth', 1.5);

    % labels, titles, legends
    xlabel('$\sigma_l$')
    ylabel('$\sigma_f^2$')
    title(titleStr);
    stStr = "$\sigma_f^2,\sigma_l|\sigma_n^2 = " + ...
        "\arg\min\tilde{R}_\mathcal{LI}" + ...
        "(\sigma_f^2,\sigma_l|\sigma_n^2)$ f. $\sigma_n^2 = const.$";
    subtitle(stStr);
    legend([p1, p2, p3], ...
        {"Parameter Bounds", "Search Area", ...
         sprintf("fmincon $\\tilde{R}_\\mathcal{LI}(%1.2f,%1.2f|%1.2e)=%1.2f$",...
            Mdl.theta, Mdl.s2n, -(Mdl.LMLcos + Mdl.LMLsin))}, ...
        'Location', 'South')

    cb = colorbar;
    cb.TickLabelInterpreter = 'latex';
    cb.Label.Interpreter = 'latex';
    cb.Label.FontSize = 24;
    cbStr = "$\tilde{R}_\mathcal{LI}(\sigma_f^2,\sigma_l|\sigma_n^2)$";
    cb.Label.String = cbStr;

    % save and close
%     fPath = fullfile(PathVariables.saveImagesPath, 'Sweep_Kernel_Const_Noise');
%     print(fig, fPath, '-dsvg');
%     close(fig);
end
```

Sweep Kernel Parameters vs. Likelihood Criteria with Constant Variance

```
function sweepKernelWithConstVariance(Mdl, nEval, titleStr, PathVariables)
```

```
% kepp s2n origin to plot later
s2nOrigin = Mdl.s2n;

% create sweep parameters for sweeping lengthscale and noise to given modle
s2n = linspace(Mdl.s2nBounds(1) * 0.1, Mdl.s2nBounds(2) * 10, nEval);
sl = linspace(Mdl.slBounds(1) * 0.1, Mdl.slBounds(2) * 10, nEval);
s2f = Mdl.theta(1);

% allocate memory for inner tuning criteria, combined likelihoods for cosine
% and sine fit on trainings data
RLI = zeros(nEval, nEval);

% run sweep in multiprocess pool to gain speed
for i = 1:nEval
    % assign struct values to compute corresponding lenght scale row wise
    % due to parfor struct issue
    Mdl.s2n = s2n(i);
    parfor j = 1:nEval
        % compute sweep with tuning criteria of inner GPR optimization of
        % tuning GPR kernel parameters, variance is set to 1
        RLI(i,j) = computeTuneCriteria([s2f sl(j)], Mdl);
    end
end

% generate grid on vectors to plot results
[sl, s2n] = meshgrid(sl, s2n);

% plot results in countour plot
fig = figure('Name', 'Sweep Kernel Parameters with Constant Variance', ...
    'Units', 'normalize', 'OuterPosition', [0 0 1 1]);

% plot sweep with log axis
contourf(sl, s2n, RLI, linspace(min(RLI, [], 'all') + 1, 1, 10), ...
    'LineWidth', 1.5);
set(gca, 'YScale', 'log')
set(gca, 'XScale', 'log')
hold on;
grid on;

% plot bounds origin model parameters
p1 = yline(Mdl.s2nBounds(1), 'k-.', 'LineWidth', 2.5);
yline(Mdl.s2nBounds(2), 'k-.', 'LineWidth', 2.5);
yline(s2nOrigin, 'k', 'LineWidth', 2.5);
xline(Mdl.slBounds(1), 'k-.', 'LineWidth', 2.5);
xline(Mdl.slBounds(2), 'k-.', 'LineWidth', 2.5);
xline(Mdl.theta(2), 'k', 'LineWidth', 2.5);

% plot fmincon search area
p2 = patch( ...
    [Mdl.slBounds(1), Mdl.slBounds(2), ...
     Mdl.slBounds(2), Mdl.slBounds(1)], ...
    [Mdl.s2nBounds(1) Mdl.s2nBounds(1), ...
     Mdl.s2nBounds(2) Mdl.s2nBounds(2)],...
    [0.8 0.8 0.8], 'FaceAlpha', 0.7);

% plot argmin fmincon result
p3 = scatter(Mdl.theta(2), s2nOrigin, 60, [0.8 0.8 0.8], ...
    'filled', 'MarkerEdgeColor', 'k', 'LineWidth', 1.5);

% labels, titles, legends
xlabel('$\sigma_l$')
ylabel('$\sigma_n^2$')
```

```
    title(titleStr);
    stStr = "$\sigma_f^2,\sigma_l|\sigma_n^2 = " + ...
        "\arg\min\tilde{R}_\mathcal{LI}" + ...
        "(\sigma_f^2,\sigma_l|\sigma_n^2)$ f. $\sigma_f^2 = const.$";
    subtitle(stStr);
    legend([p1, p2, p3], ...
        {"Parameter Bounds", "Search Area", ...
         sprintf("fmincon $\\tilde{R}_\\mathcal{LI}(%1.2f,%1.2f|%1.2e)=%1.2f$",...
            Mdl.theta, s2nOrigin, -(Mdl.LMLcos + Mdl.LMLsin))}, ...
        'Location', 'South')

    cb = colorbar;
    cb.TickLabelInterpreter = 'latex';
    cb.Label.Interpreter = 'latex';
    cb.Label.FontSize = 24;
    cbStr = "$\tilde{R}_\mathcal{LI}(\sigma_f^2,\sigma_l|\sigma_n^2)$";
    cb.Label.String = cbStr;

    % save and close
%     fPath = fullfile(PathVariables.saveImagesPath, 'Sweep_Kernel_Const_Var');
%     print(fig, fPath, '-dsvg');
%     close(fig);
end
```

Sweep Kernel Parameters vs. Likelihood Criteria with Constant Lengthscale

```
function sweepKernelWithConstLengthscale(Mdl, nEval, titleStr, PathVariables)

    % kepp s2n origin to plot later
    s2nOrigin = Mdl.s2n;

    % create sweep parameters for sweeping lengthscale and noise to given modle
    s2n = linspace(Mdl.s2nBounds(1) * 0.1, Mdl.s2nBounds(2) * 10, nEval);
    s2f = linspace(Mdl.s2fBounds(1) * 0.1, Mdl.s2fBounds(2) * 10, nEval);
    sl = Mdl.theta(2);

    % allocate memory for inner tuning criteria, combined likelihoods for cosine
    % and sine fit on trainings data
    RLI = zeros(nEval, nEval);

    % run sweep in multiprocess pool to gain speed
    for i = 1:nEval
        % assign struct values to compute corresponding lenght scale row wise
        % due to parfor struct issue
        Mdl.s2n = s2n(i);
        parfor j = 1:nEval
            % compute sweep with tuning criteria of inner GPR optimization of
            % tuning GPR kernel parameters, variance is set to 1
            RLI(i,j) = computeTuneCriteria([s2f(j) sl], Mdl);
        end
    end

    % generate grid on vectors to plot results
    [s2f, s2n] = meshgrid(s2f, s2n);

    % plot results in countour plot
    fig = figure('Name', 'Sweep Kernel Parameters with Constant Lenghtscale',...
        'Units', 'normalize', 'OuterPosition', [0 0 1 1]);

    % plot sweep with log axis
    contourf(s2f, s2n, RLI, linspace(min(RLI, [], 'all') + 1, 1, 10), ...
```

```
        'LineWidth', 1.5);
    set(gca, 'YScale', 'log')
    set(gca, 'XScale', 'log')
    hold on;
    grid on;

    % plot bounds origin model parameters
    p1 = yline(Mdl.s2nBounds(1), 'k-.', 'LineWidth', 2.5);
    yline(Mdl.s2nBounds(2), 'k-.', 'LineWidth', 2.5);
    yline(s2nOrigin, 'k', 'LineWidth', 2.5);
    xline(Mdl.s2fBounds(1), 'k-.', 'LineWidth', 2.5);
    xline(Mdl.s2fBounds(2), 'k-.', 'LineWidth', 2.5);
    xline(Mdl.theta(1), 'k', 'LineWidth', 2.5);

    % plot fmincon search area
    p2 = patch( ...
        [Mdl.s2fBounds(1), Mdl.s2fBounds(2), ...
         Mdl.s2fBounds(2), Mdl.s2fBounds(1)], ...
        [Mdl.s2nBounds(1) Mdl.s2nBounds(1), ...
         Mdl.s2nBounds(2) Mdl.s2nBounds(2)],...
        [0.8 0.8 0.8], 'FaceAlpha', 0.7);

    % plot argmin fmincon result
    p3 = scatter(Mdl.theta(1), s2nOrigin, 60, [0.8 0.8 0.8], ...
        'filled', 'MarkerEdgeColor', 'k', 'LineWidth', 1.5);

    % labels, titles, legends
    xlabel('$\sigma_f^2$')
    ylabel('$\sigma_n^2$')
    title(titleStr);
    stStr = "$\sigma_f^2,\sigma_l|\sigma_n^2 = " + ...
        "\arg\min\tilde{R}_\mathcal{LI}" + ...
        "(\sigma_f^2,\sigma_l|\sigma_n^2)$ f. $\sigma_l = const.$";
    subtitle(stStr);
    legend([p1, p2, p3], ...
        {"Parameter Bounds", "Search Area", ...
         sprintf("fmincon $\\tilde{R}_\\mathcal{LI}(%1.2f,%1.2f|%1.2e)=%1.2f$",...
            Mdl.theta, s2nOrigin, -(Mdl.LMLcos + Mdl.LMLsin))}, ...
        'Location', 'South')

    cb = colorbar;
    cb.TickLabelInterpreter = 'latex';
    cb.Label.Interpreter = 'latex';
    cb.Label.FontSize = 24;
    cbStr = "$\tilde{R}_\mathcal{LI}(\sigma_f^2,\sigma_l|\sigma_n^2)$";
    cb.Label.String = cbStr;

    % save and close
%     fPath = fullfile(PathVariables.saveImagesPath, 'Sweep_Kernel_Const_Len');
%     print(fig, fPath, '-dsvg');
%     close(fig);
end
```

Published with MATLAB® R2020b