predFrame

Predicts single test point and computes angle and radius by predicted sinoids. Delivers several quality criteria.

Syntax

```
[fang, frad, fcos, fsin, fcov, s, ciang, cirad] = predFrame(Mdl, Xcos, Xsin)
```

Description

[fang, frad, fcos, fsin, fcov, s, ciang, cirad] = predFrame(Mdl, Xcos, Xsin)predicts sinoids by passed regression model and test frame of raw data matrix. Comutes angle and radius by predicted results. Several quality creteria are setup based on predictive variance.

Input Argurments

Mdl model struct.

Xcos matrix frame of cosine test data.

Xsin matrix frame of sine test data.

Output Argurments

fang computed angle by predicted cosine and sine results.

frad computed radius by predicted cosine and sine results.

fcos predictive mean result of cosine regression.

fsin predictive mean result of sine regression.

fcov predictive variance for both predictive means.

s resulting standard deviation by predictive variance and noise level.

ciang confidence interval of computed angle.

cirad confidence interval of computed radius.

Requirements

- Other m-files required: None
- Subfunctions: computeTransposeInverseProduct, sinoids2angles
- MAT-files required: None

See Also

- computeTransposeInverseProduct
- sinoids2angles
- initKernel
- initKernelParameters

```
function [fang, frad, fcos, fsin, fcov, s, ciang, cirad] = predFrame(Mdl, ...
    Xcos, Xsin)

    % adjust inputs if needed
    Xcos = Mdl.inputFun(Xcos);
    Xsin = Mdl.inputFun(Xsin);

    % compute covariance between observations and test point
    k = Mdl.kernelFun(Mdl.Xcos, Xcos, Mdl.Xsin, Xsin, Mdl.theta);

    % compute predictiv variance as the difference between test point covariance
    % which should be Mdl.theta(1) = s2f  product of the covariance between
    % observations and test points
    % compute the covariance of test point itself means distance is zero which
    % implies that result must be the variance s2f
    c1 = Mdl.kernelFun(Xcos, Xcos, Xsin, Xsin, Mdl.theta);
    % assert(c1 == Mdl.theta(1));

    % now add variance from additives
    fcov = c1 - computeTransposeInverseProduct(Mdl.L, k);

    % predict depending on model mean function
    switch Mdl.mean
        case 'zero'
             % compute the predictive means directly by covariance vector and
             % alpha weights, mean is zero
            fcos = k' * Mdl.AlphaCos;
            fsin = k' * Mdl.AlphaSin;
        case 'poly'
            % compute
            fcos = Mdl.meanFunCos(Xcos) + k' * Mdl.AlphaCos;
            fsin = Mdl.meanFunSin(Xsin) + k' * Mdl.AlphaSin;
        otherwise
            error('Unsupported mean function %s in prediction.', Mdl.mean);
    end

    % compute radius from sinoid results
    frad = sqrt(fcos^2 + fsin^2);

    % compute angle in rad from sinoid results
    fang = sinoids2angles(fsin, fcos, frad, true);

    % sigma of the normal distribution over fradius
    s = sqrt(fcov + Mdl.s2n);

    % 95% confidence interval over fradius
    ciang = [fang - asin(1.96 * s * sqrt(2)), fang + asin(1.96 * s * sqrt(2))];

    % 95% confidence interval over fradius
    cirad = [frad - 1.96 * s * sqrt(2), frad + 1.96 * s * sqrt(2)];
end
```

Published with MATLAB® R2020b