stgp class provides the spatial-temporial Gaussianc process.

It uses the exponential kernel for the input. It is mainly used to make predictions given observations of Cartesian product coordinates of space and time.

Required python library: pytorch, numpy,

Initialization:

* space\_coordinates, the space coordinate, must be a matrix of [number of space\_coordinates x (lat,long,elevation)] in UTM or any other coordinate using meter as unit
* time\_coordinates, must be a matrix of [number of time\_coordinates x 1] in hour formate
* stData, the real (**calibrated**) pm25 values. It must be a full matrix. If there are missing value, use interpolation the fill the missing position or exclude the row/column if the miss values are too many to provide valid interpolations (remember to modify space\_coordinates/ time\_coordinates accordingly if doing so)
* must be a matrix of [space\_coordinates.size(0) x time\_coordinates.size(0)]
* latlong\_length\_scale=4300., length scale of latitude and lontitude. Default value 4300
* elevation\_length\_scale=30., length scale of elevation. Default value 30
* time\_length\_scale= 0.25, length scale of time. Default value 0.25
* noise\_variance = 0.1 , noise variance, this is the variance for all sensor noise. Since the
* signal\_variance = 1, signal variance, this it he variance for the underlying pm2.5. signal\_variance is fixed to 1, and thus noise\_variance is the variance relative to 1, which could be also understood as the ratio of noise variance to the signal variance.

Methods:

* yPred, yVar = forward(self, test\_space\_coordinates, test\_time\_coordinates), return predictions of the given spatial coordinates and time\_coordinates. yPred is the predictive values and yVar is the variance (uncertainty) of the predictions.
* Train\_bfgs(self, niteration, lr=0.001). Optimize hyperparameters (latlong\_length\_scale, time\_length\_scale, noise\_variance, and signal\_variance) using L-BFGS-B with given learning rate and number of iterations. BE WARE of overfitting. The optimized hyperparameters can be visited as properties of the stgp class.
* Train\_adam(self, niteration, lr=0.001). Optimize hyperparameters (latlong\_length\_scale, time\_length\_scale, noise\_variance, and signal\_variance) using adam with given learning rate and number of iterations.

Tips:

* a wider spatial/temporal window of initialization data can provides a more stable training process. The limitation is the uniform length scale and a long computational time.
* Users can use a smaller window of initialization and different length scale to give predictions more flexibility.
* Remember the prediction (forward method) should use spatial-temporal coordinates that are covered by the initialization data. Otherwise the results will show large uncertainty.