**Gaussian Processes Procedure Comparison**

The Gaussian Processes was first trained with GPML MATLAB tool1 because of its wild range of kernel options. However, with heavy computational, GPML Gaussian Processes takes more than 30 minutes while the MATLAB self-included Gaussian Processes regression model only takes about 1~ minute. Yet, the negative log likelihoods obtained by MATLAB model is better.

**Gaussian Processes vs Linear Regression**

The running time for Gaussian processes is about 1~ minute without automatic hyper-parameter optimization, while training the linear regression for milestone one only takes a few seconds. This is mostly due to the computation of the covariance matrix.

Gaussian Processes seem to produce better . The training loss of linear regression is about 60 and 100 for motor score and total score fitting respectively (with cross validation). However, the training loss of Gaussian processes is around 30 (without cross validation) and 40 (with cross validation) for motor score. As for the Gaussian processes with total score, the training loss is around 45 (without cross validation) and 70 (with cross validiation).

**Prior distribution**

The initial value for the noise standard deviation for the Gaussian process model is the square root of the standard deviation of divided by the square root of 2.

**Kernel/Covariance Functions2:**

1. Squared Exponential kernel
2. Exponential kernel
3. Matern kernel ()

Note: the hyper parameters are not optimized.

**Training Output**

Training with Motor Score

Round 1 without cross validation

Training with squared exponential kernel

28-Mar-2018 16:37:40

28-Mar-2018 16:38:03

Negative log likelihood is -6739.7306

The training loss is 36.8477

...

Training with exponential kernel

28-Mar-2018 16:38:03

28-Mar-2018 16:38:26

Negative log likelihood is -6760.3959

The training loss is 22.4103

...

Training with matern 3/2 kernel

28-Mar-2018 16:38:26

28-Mar-2018 16:38:52

Negative log likelihood is -6744.0904

The training loss is 31.3263

Round 2 with 10-fold cross validation

Training with squared exponential kernel

28-Mar-2018 16:38:53

28-Mar-2018 16:39:18

Negative log likelihood is -6739.7306

The loss is 43.5607

...

Training with exponential kernel

28-Mar-2018 16:39:18

28-Mar-2018 16:39:55

Negative log likelihood is -6760.3959

The loss is 41.6602

...

Training with matern 3/2 kernel

28-Mar-2018 16:39:55

28-Mar-2018 16:40:27

Negative log likelihood is -6744.0904

The loss is 38.9444

Training with Motor Score

Round 1 without cross validation

Training with squared exponential kernel

28-Mar-2018 16:40:27

28-Mar-2018 16:40:57

Negative log likelihood is -6739.7306

The training loss is 58.0305

...

Training with exponential kernel

28-Mar-2018 16:40:57

28-Mar-2018 16:41:34

Negative log likelihood is -6760.3959

The training loss is 36.0366

...

Training with matern 3/2 kernel

28-Mar-2018 16:41:35

28-Mar-2018 16:42:07

Negative log likelihood is -6744.0904

The training loss is 49.0757

Round 2 with 10-fold cross validation

Training with squared exponential kernel

28-Mar-2018 16:42:08

28-Mar-2018 16:42:36

Negative log likelihood is -6739.7306

The loss is 70.9451

...

Training with exponential kernel

28-Mar-2018 16:42:36

28-Mar-2018 16:43:10

Negative log likelihood is -6760.3959

The loss is 70.1313

...

Training with matern 3/2 kernel

28-Mar-2018 16:43:10

28-Mar-2018 16:43:40

Negative log likelihood is -6744.0904

The loss is 67.1333

>>

**Kernel Comparison**

From the above training output, it is clear that exponential kernel function has the best performance in terms of negative log likelihood. Either with or without the cross-validation, the exponential kernel has the best log likelihood for training with both motor score and total score.

As for the loss function with training data (Squared Loss), Matern 3/2 kernel has the best performance when the model is trained with cross validation, followed by exponential kernel and then the squared exponential kernel. Without cross validation, exponential kernel has the best performance while the squared exponential kernel remains the third.

However, this means that the model with exponential kernel is better at fitting the training samples, which means that it is also more likely to over fit when trained without cross validation.

In conclusion, the exponential kernel has the best overall performance.

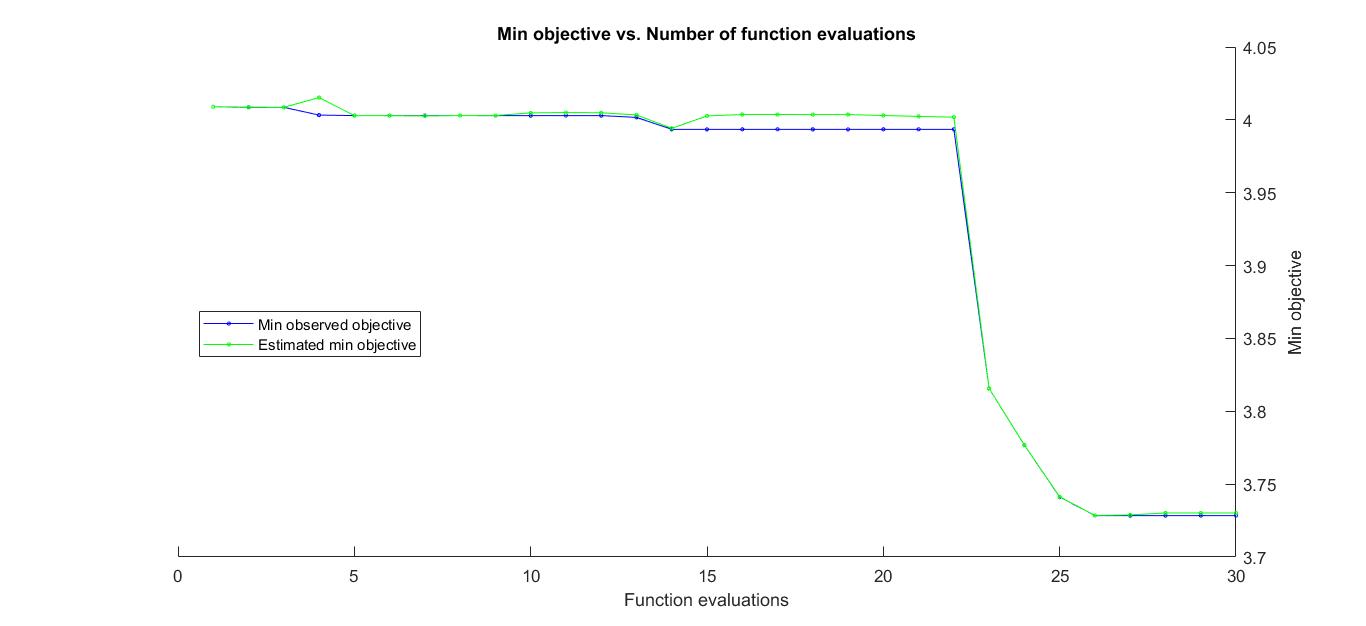
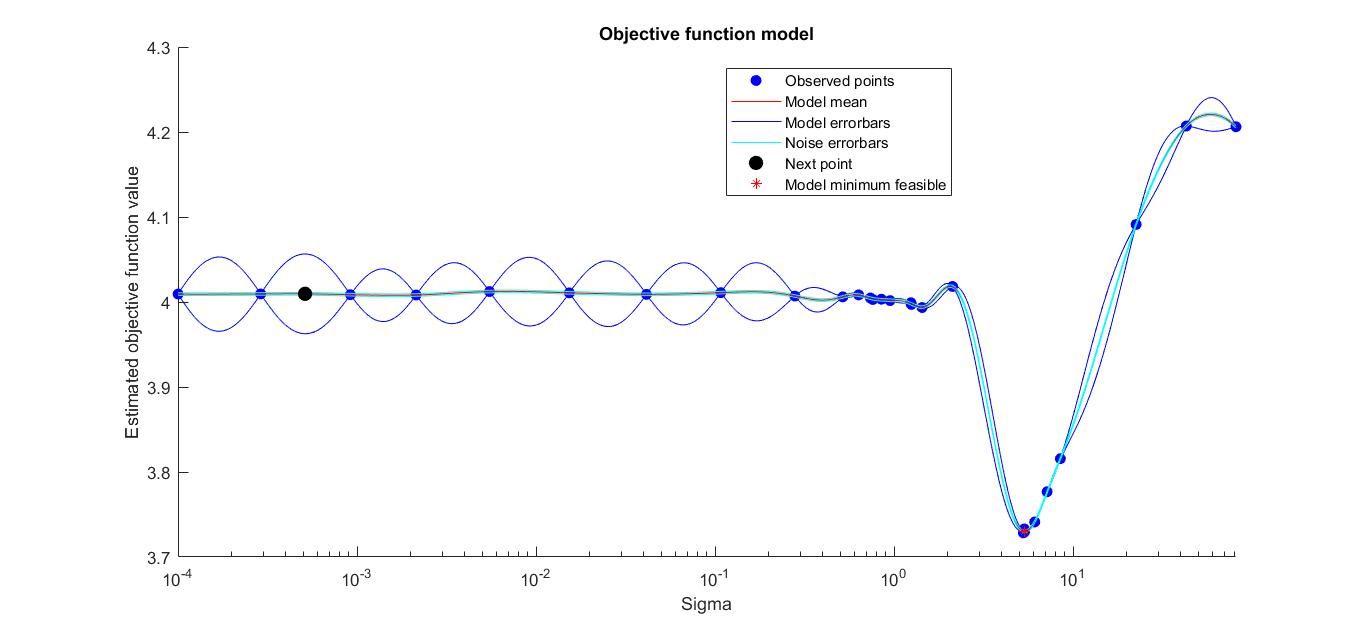
**Optimization**

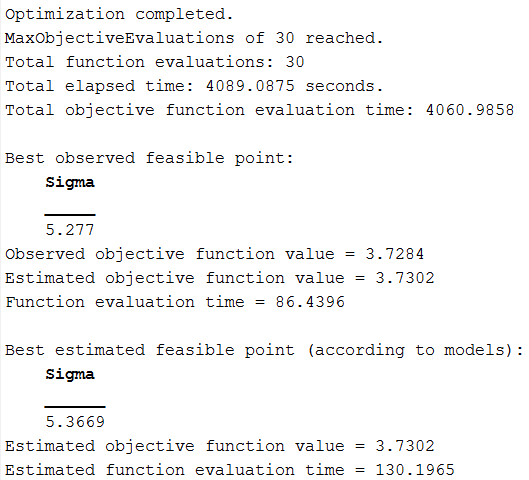
Since the hyper parameters are not optimized, potentially each model with different kernel function may perform better with optimal hyper parameters. The optimal hyper parameters can be found by cross validation with MATLAB built-in name pairs

'OptimizeHyperparameters','auto','HyperparameterOptimizationOptions',...

struct('AcquisitionFunctionName','expected-improvement-plus')

Below is the output from the exponential kernel model hyper parameter optimization.





The data label standard deviation is around 9.1293 and the optimized estimation is around 5.3669