

Machine Learning and MSC Nastran

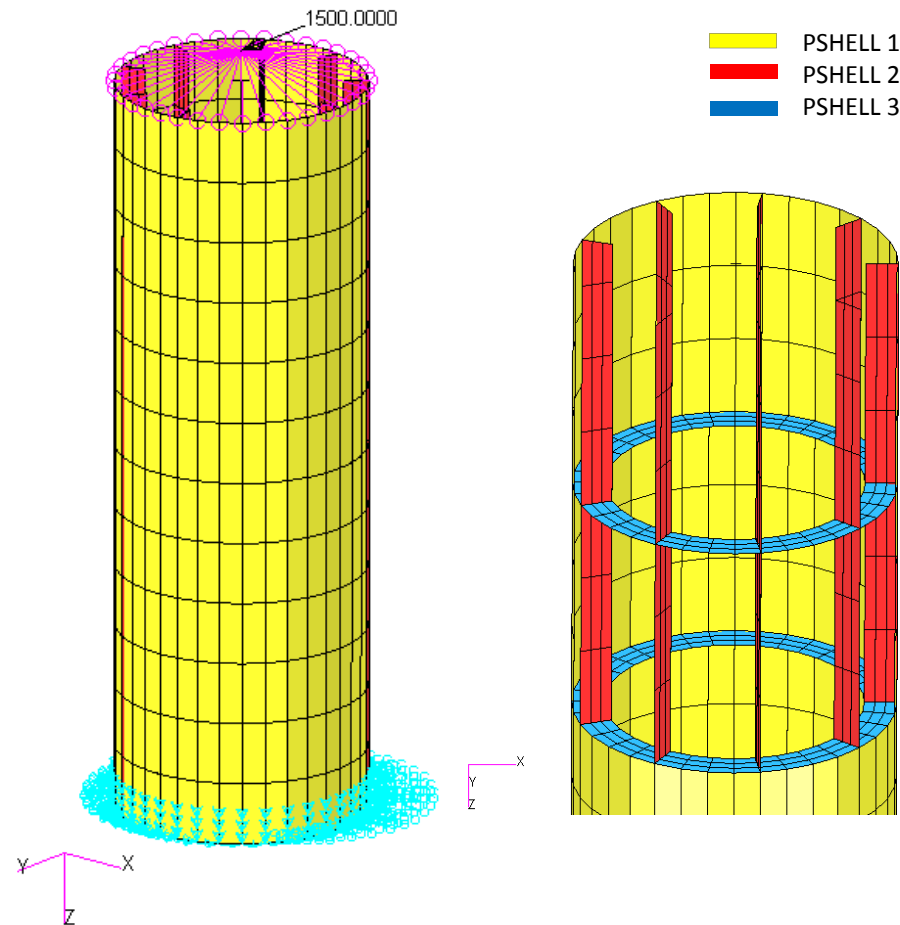
PRESENTED BY: CHRISTIAN APARICIO

AUGUST 7, 2022

Agenda

- Demo of Machine Learning with MSC Nastran
- Models and Types of Learning Algorithms in Machine Learning
- Goals
 - Sequential Design/Active Learning
 - Fit Model
 - MVN Conditioning Equations
 - Example 1
 - Example 2
 - Choose Next Point
 - Acquisition Functions
- Conclusion
- FAQ
- Common Terminology

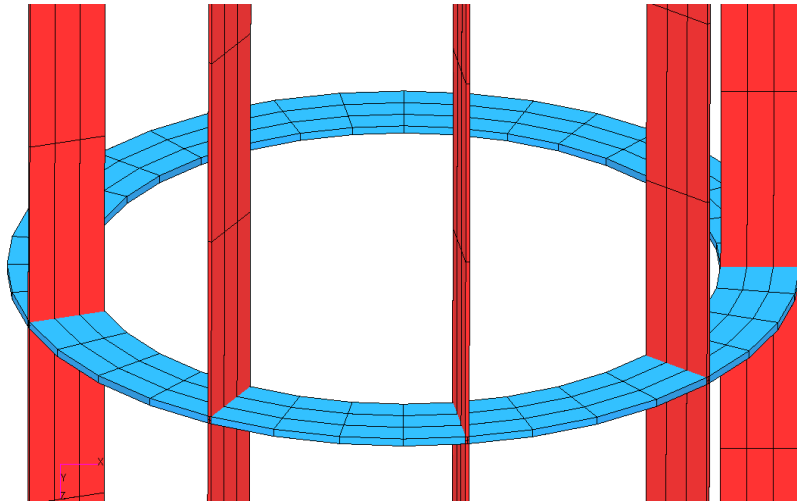
Demo



Goal: Use Machine Learning for Nonlinear Response Optimization

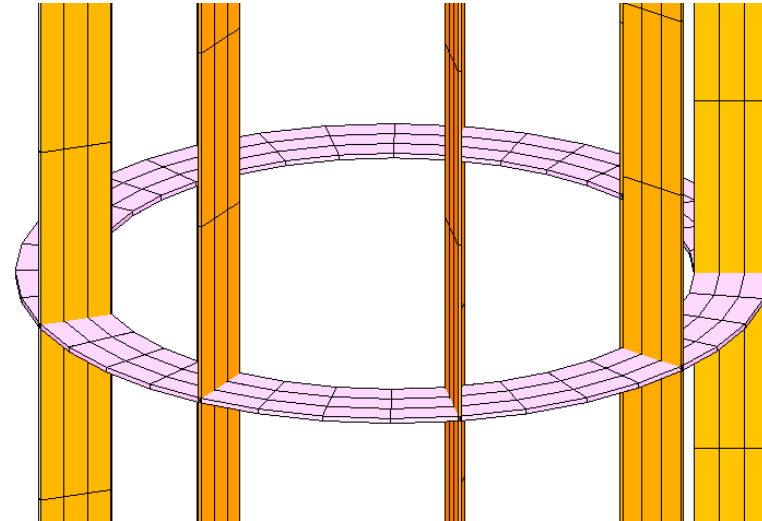
Before Optimization

- Weight: 552021.8
- Pcr: 2435.129 N



After Optimization

- Weight: ~533714.01
- Pcr: ~2000.0 N

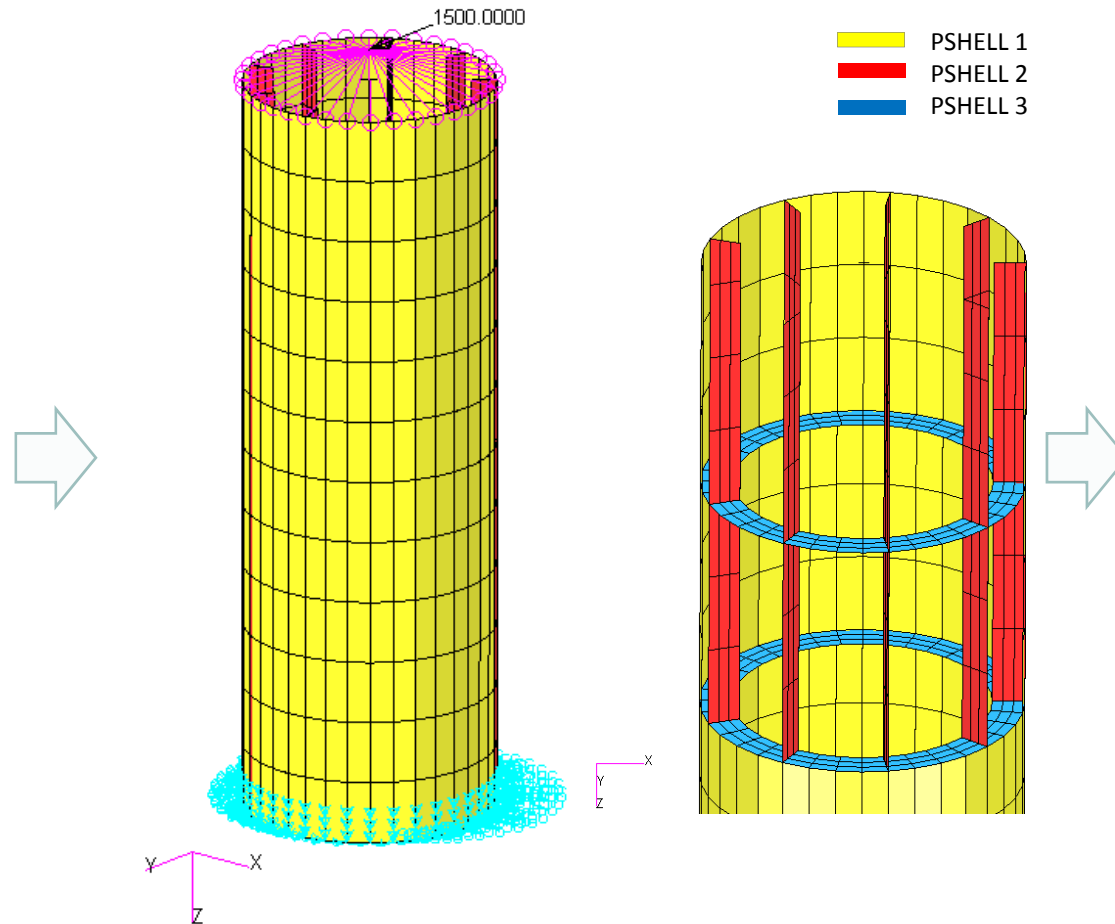


Optimization Problem Statement

X Inputs - Design Variables

x1: Thickness of PSHELL 2
x2: Thickness of PSHELL 3

$$1.0 < x_i < 5.$$



Y Outputs – Objective and Constraint Responses

Objective

r0: Minimize the weight

Constraints

r1: Critical buckling load

$$2000.0 < r1$$

Buckling load is based on nonlinear buckling (post-buckling) analysis with geometry nonlinearity and/or material nonlinearity

Machine Learning Results

- The entire process consists of 2 phases.
 - Phase A – Initial Training Data Acquisition
 - This phase involves evaluating the FE model at different sampling points and recovering the monitored responses for the objective and constraints. The recovered monitored responses are referred to as training data.
 - This training data is used to train the regression model at the start of phase B, the machine learning phase.
 - Phase B – Machine Learning
 - This phase involves the machine learning process. The regression models for the objective and constraints are used to determine the next sample point to evaluate.
 - After each sample evaluated, the regression models are updated with the latest training data.
- This example was initially configured for a 10 sample Latin Hypercube design. After the initial training data was required, machine learning was executed for 20 runs. A total of 30 runs were performed.

Session ID:
20118



Completed
successfully

Upload .csv File

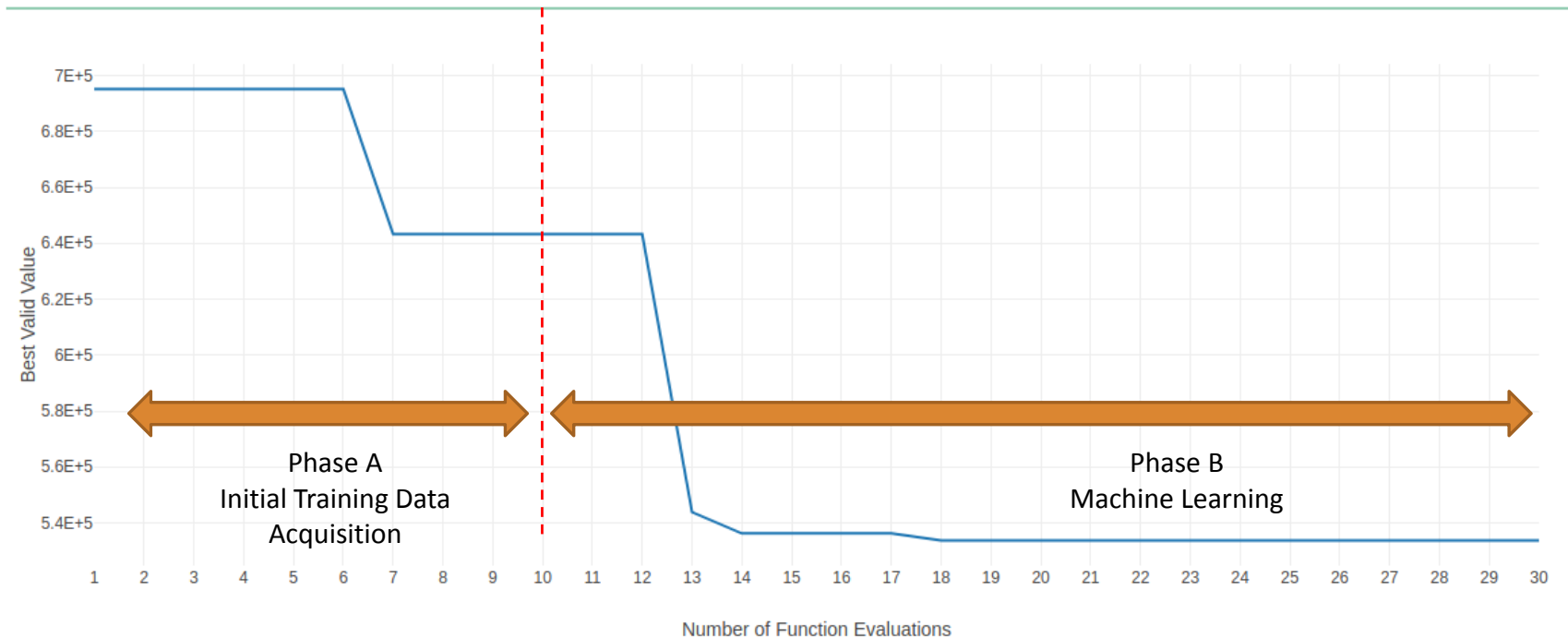
CSV Import

Select files

a_tmp_best_valid_value.csv

Import

Best Valid Value



Machine Learning Results

1. A bar chart displays the objective value after each sample. A green colored bar indicates the constraints are satisfied for that sample. A gray colored bar indicates the constraints are NOT satisfied for that sample.
2. The Status message indicates sample 18 is the best design. Your solution will be different.
3. Use the horizontal bar to locate sample 18 in the table.

The best feasible design yields an objective of $5.3371\text{E}+5$ with $x_1=1.0286$ and $x_2=1.3661$. Your solution might be different.

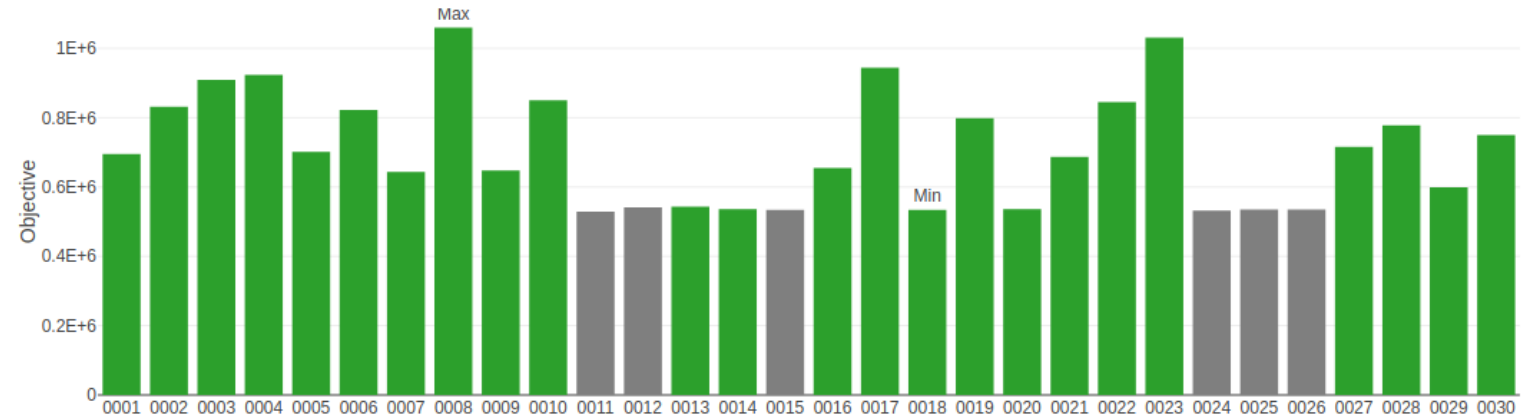
- Important! The machine learning process involves some randomization of data. A consequence of this randomization is that the number of samples needed to obtain the optimum may vary. For example, when you perform this example, you may obtain the optimum after a total of 25 runs, 33 runs, 38 runs, or some other number of runs.

Status

1

THE LATEST OPTIMAL SOLUTION IS: **SAMPLE # 18 (MIN)** SAMPLE # 8 (MAX)
 OBJECTIVE = $5.3371\text{E}+5$ (MIN), $1.0599\text{E}+6$ (MAX)
 MAXIMUM CONSTRAINT VALUE = $-8.2112\text{E}-3$, $-1.0731\text{E}+0$ (FEASIBLE DESIGNS)

Objective for Each Sample



Data for Each Sample

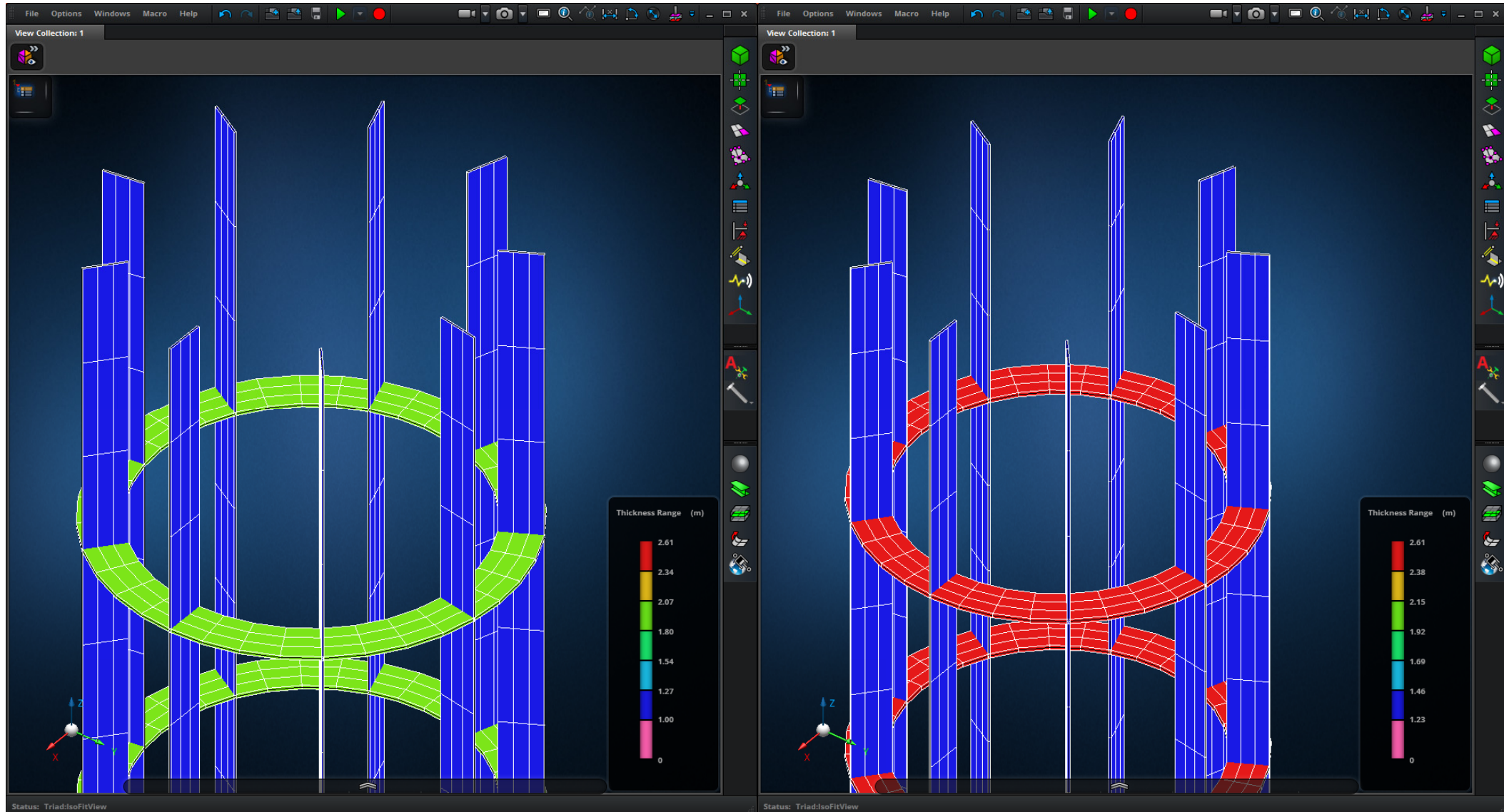
Sample

3

Item	le 0013	Sample 0014	Sample 0015	Sample 0016	Sample 0017	Sample 0018	Sample 0019	Sample 0020	Sample 0021	Sample 0022	Sam
Extrema (Max/Min)						Min					
Objective	E+5	$5.3620\text{E}+5$	$5.3360\text{E}+5$	$6.5471\text{E}+5$	$9.4400\text{E}+5$	$5.3371\text{E}+5$	$7.9829\text{E}+5$	$5.3575\text{E}+5$	$6.8654\text{E}+5$	$8.4526\text{E}+5$	1.031
Normalized Constraint	3E-3	$-5.8229\text{E}-2$	$3.2651\text{E}-2$	$-6.4504\text{E}-1$	$-1.0914\text{E}+0$	$-8.2112\text{E}-3$	$-1.1351\text{E}+0$	$-1.0750\text{E}-2$	$-8.7054\text{E}-1$	$-1.1222\text{E}+0$	-1.07
X1	E+0	$1.0000\text{E}+0$	$1.0812\text{E}+0$	$1.2062\text{E}+0$	$3.9655\text{E}+0$	$1.0286\text{E}+0$	$3.3280\text{E}+0$	$1.0584\text{E}+0$	$2.3455\text{E}+0$	$3.4666\text{E}+0$	4.735
X2	E+0	$1.5314\text{E}+0$	$1.1946\text{E}+0$	$4.3823\text{E}+0$	$4.1242\text{E}+0$	$1.3661\text{E}+0$	$1.8472\text{E}+0$	$1.3311\text{E}+0$	$1.6804\text{E}+0$	$2.7952\text{E}+0$	4.237

2

New dimensions, post machine learning, displayed in MSC Apex



Agenda

- Demo of Machine Learning with MSC Nastran
- Models and Types of Learning Algorithms in Machine Learning
- Goals
 - Sequential Design/Active Learning
 - Fit Model
 - MVN Conditioning Equations
 - Example 1
 - Example 2
 - Choose Next Point
 - Acquisition Functions
- Conclusion
- FAQ
- Common Terminology

Models and Types of Learning Algorithms in Machine Learning

Models

- Artificial neural networks
- Decision trees
- Support vector machines
- Regression analysis
 - Linear regression
 - Polynomial regression
 - Gaussian Process regression
- Bayesian networks
- Genetic algorithms

Focus of this presentation

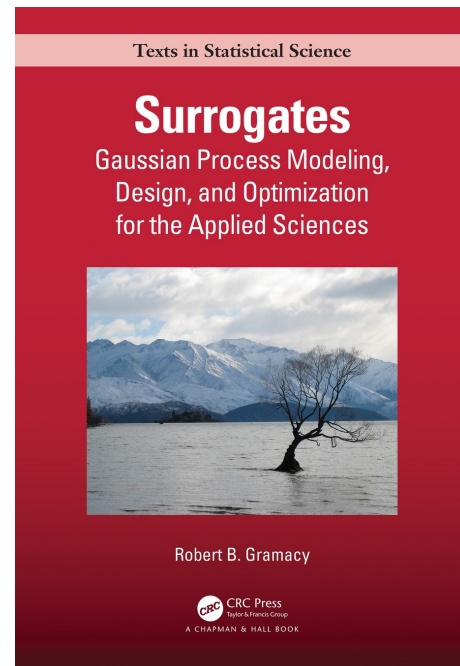
Types of learning algorithms

- Supervised learning
 - Active learning/Sequential Design
 - Bayesian Optimization
 - Classification
 - Regressions
- Unsupervised learning
- Semi-supervised learning
- Reinforcement learning
- Self learning
- Feature learning
- Sparse dictionary learning
- Anomaly detection
- Robot learning
- Association rules

Acknowledgement

The majority of the content in this presentation is sourced from the following reference

- Gramacy, Robert B. *Surrogates: Gaussian Process Modeling, Design, and Optimization for the Applied Sciences*. CRC Press, Taylor and Francis Group, 2020.



This book uses *surrogate model* to refer to the function that makes predictions but this presentation uses *prediction model* to mean the same thing

Free E-Book Available At:

<https://bobby.gramacy.com/surrogates/>

Goals

Sequential Design/Active Learning

Goal - Learn how to:

1. Fit model
 - MVN Conditioning Equations
2. Choose next design point
 - Acquisition Functions (Criteria)

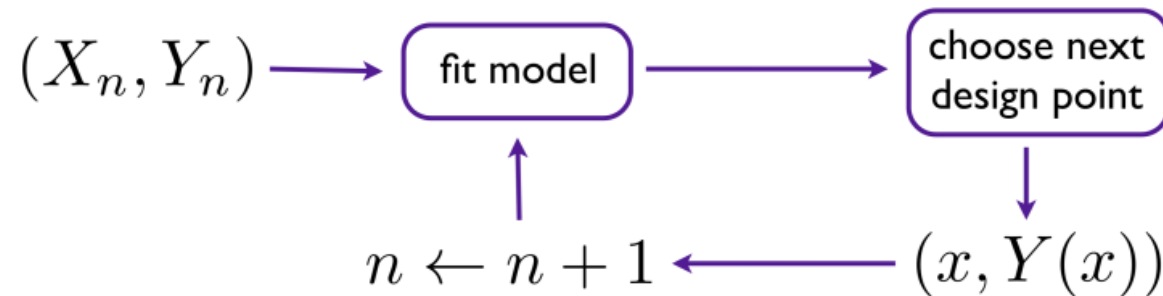


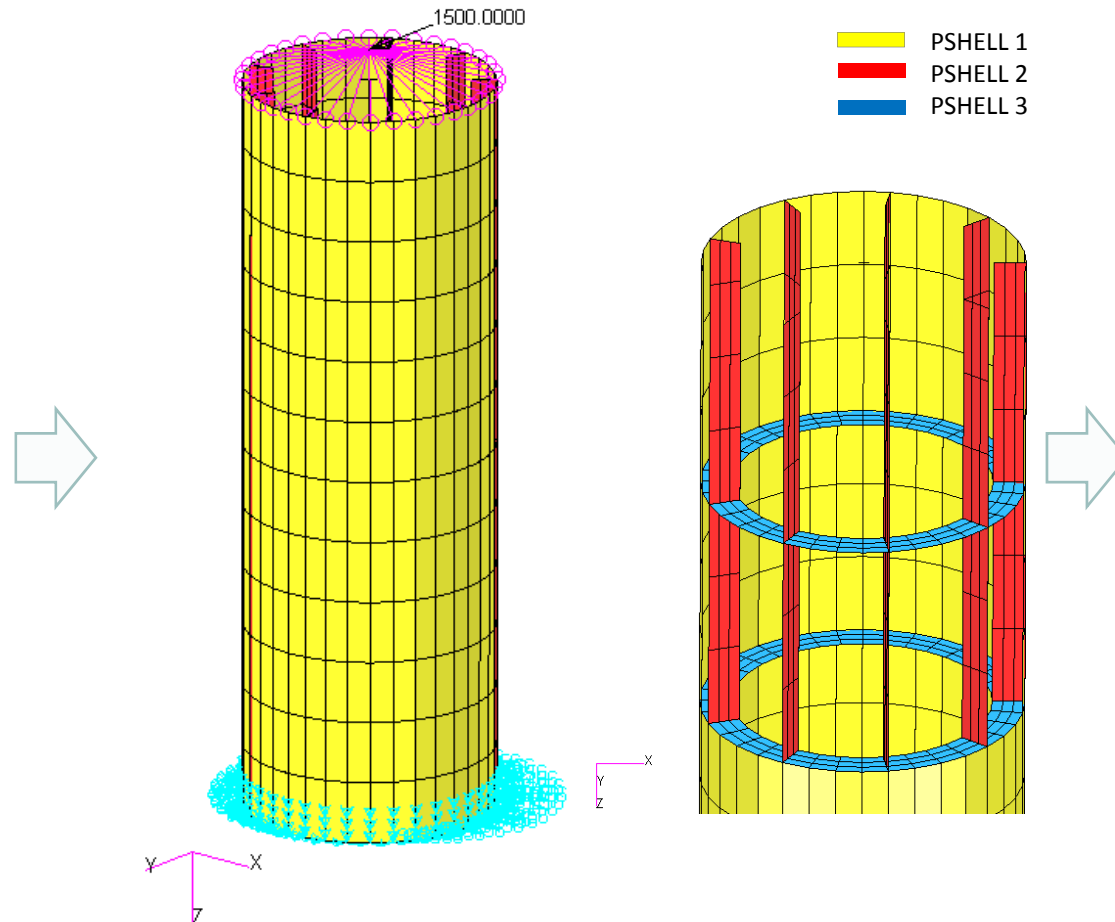
FIGURE 6.8: Diagram of sequential design/active learning/design augmentation.

Optimization Problem Statement

x Inputs - Design Variables

x1: Thickness of PSHELL 2
x2: Thickness of PSHELL 3

$$1.0 < x_i < 5.$$



y Outputs – Objective and Constraint Responses

Objective

r0: Minimize the weight

Constraints

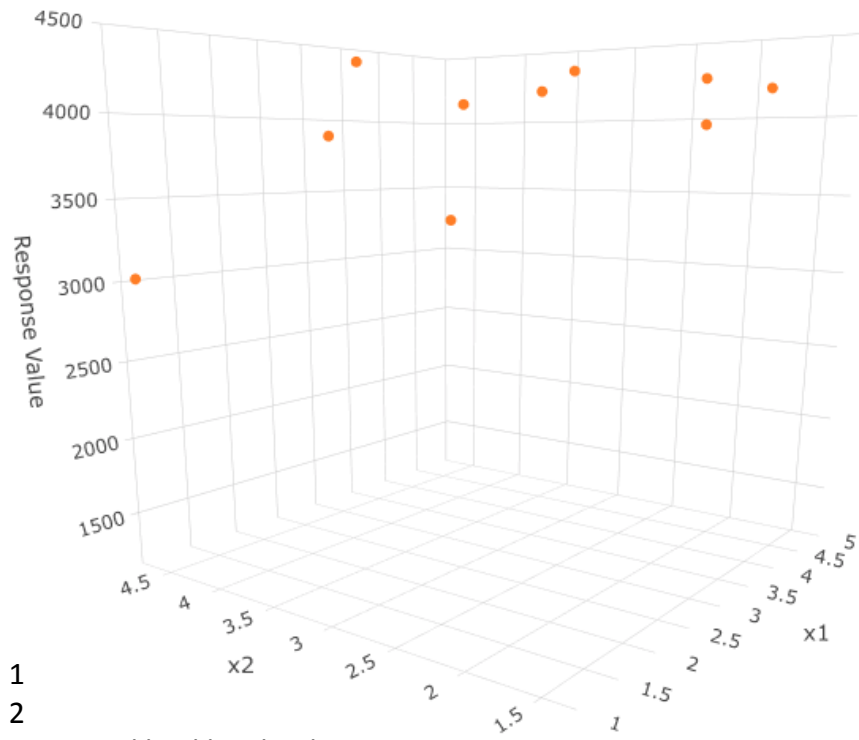
r1: Critical buckling load

$$2000.0 < r1$$

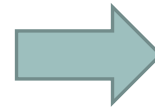
Buckling load is based on nonlinear buckling (post-buckling) analysis with geometry nonlinearity and/or material nonlinearity

Fit Model

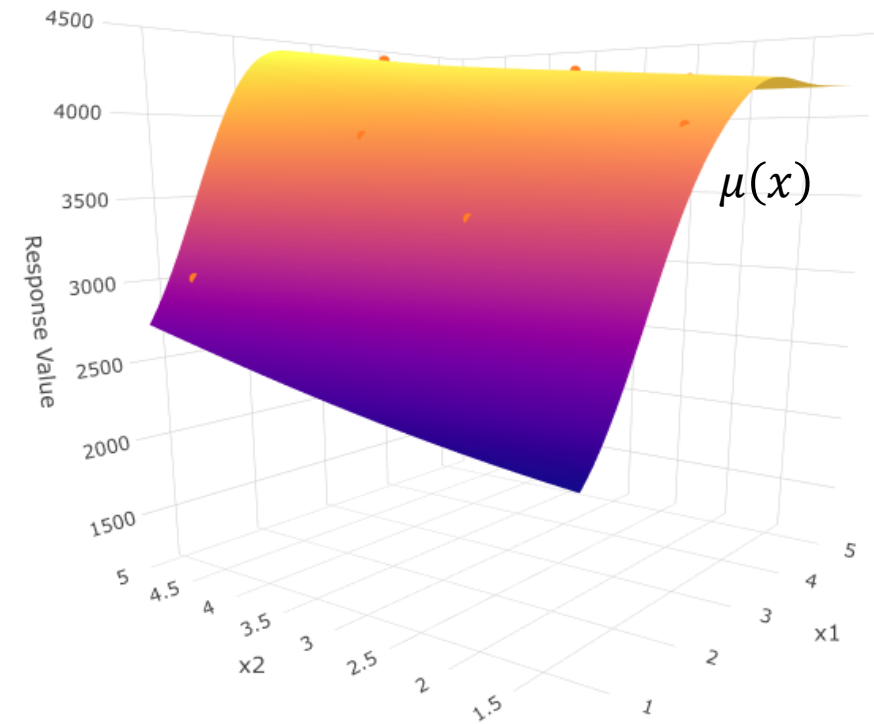
Initial Training Data



x1: Thickness 1
x2: Thickness 2
Response Value: Critical buckling load

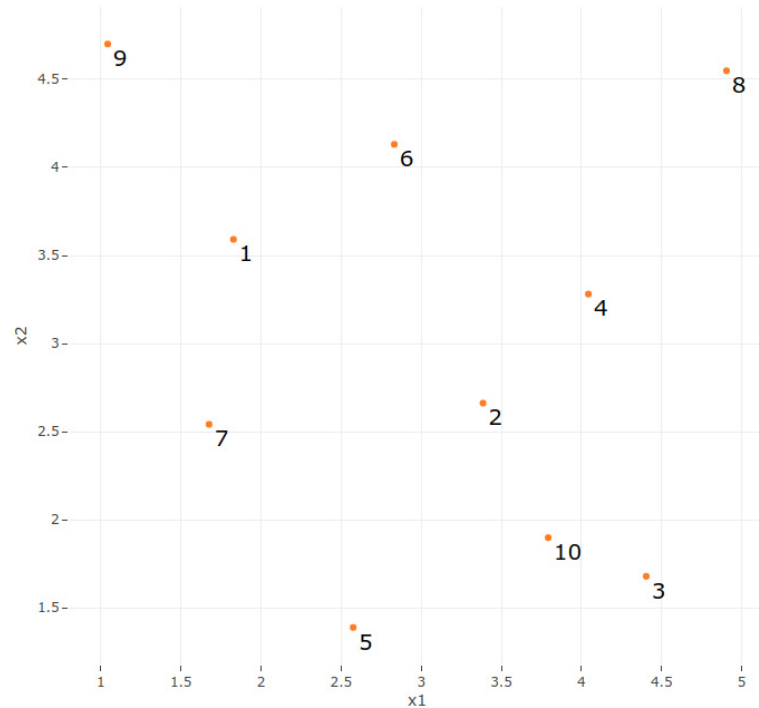


Initial Prediction Model
(surrogate model, meta model, emulator)

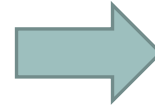


Fit Model, Continued

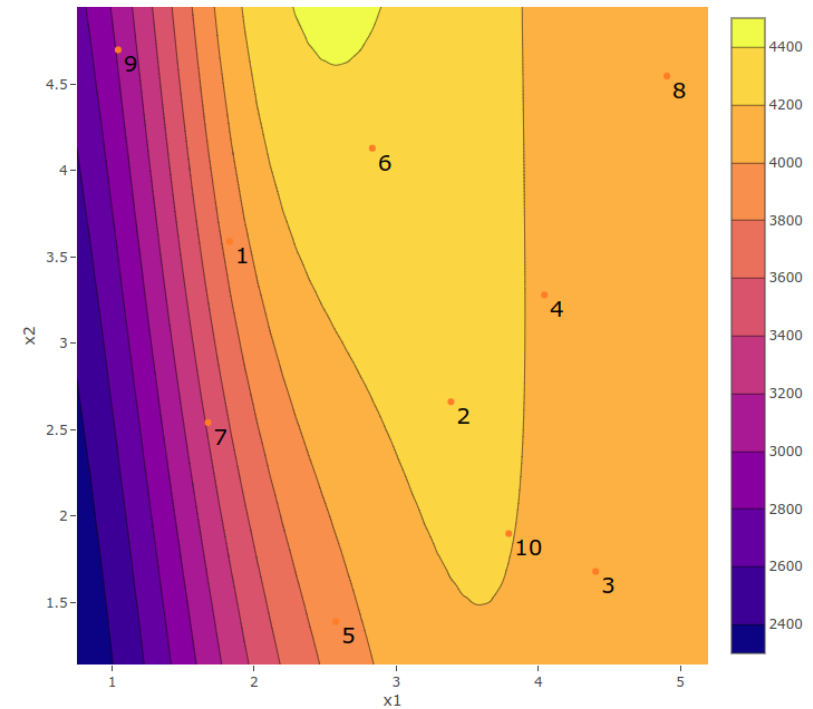
Initial Training Data



x1: Thickness 1
x2: Thickness 2
Response Value: Critical buckling load

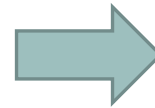
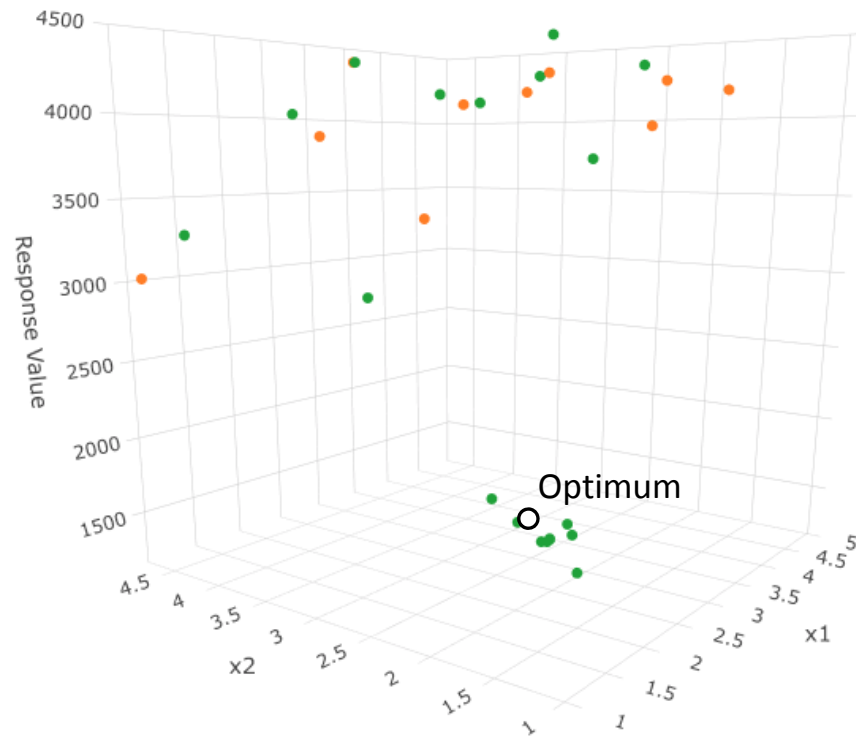


Initial Prediction Model
(surrogate model, meta model, emulator)

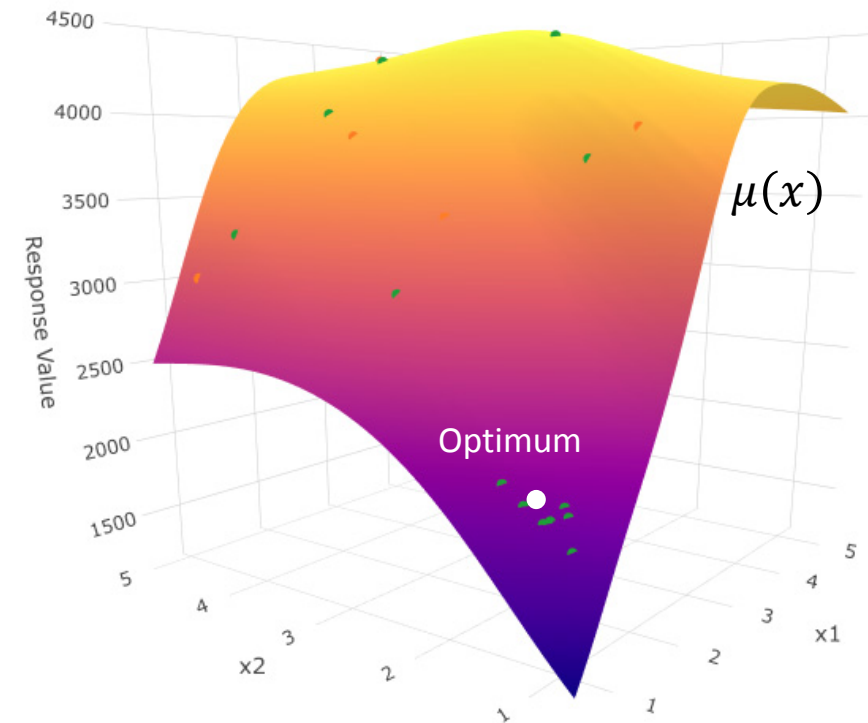


Choose next design point

Updated Training Data

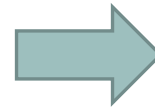
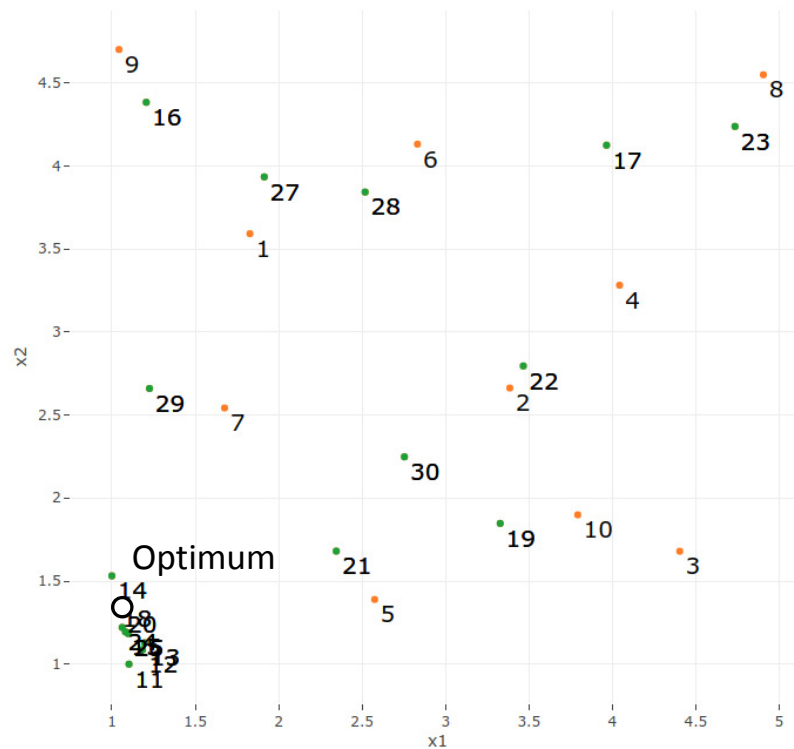


Updated Prediction Model
(surrogate model, meta model, emulator)

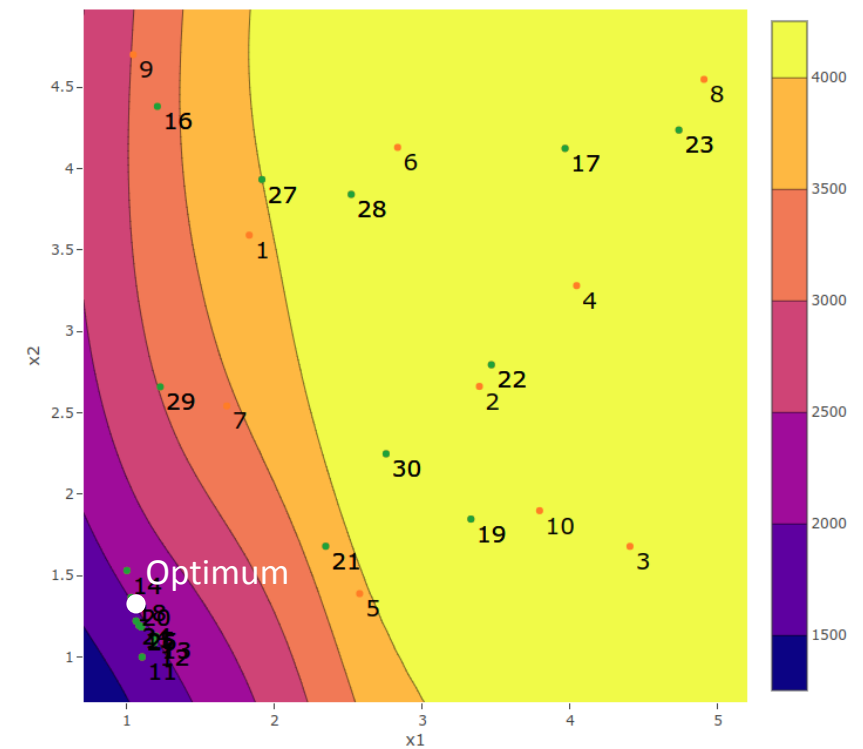


Choose next design point, Continued

Updated Training Data

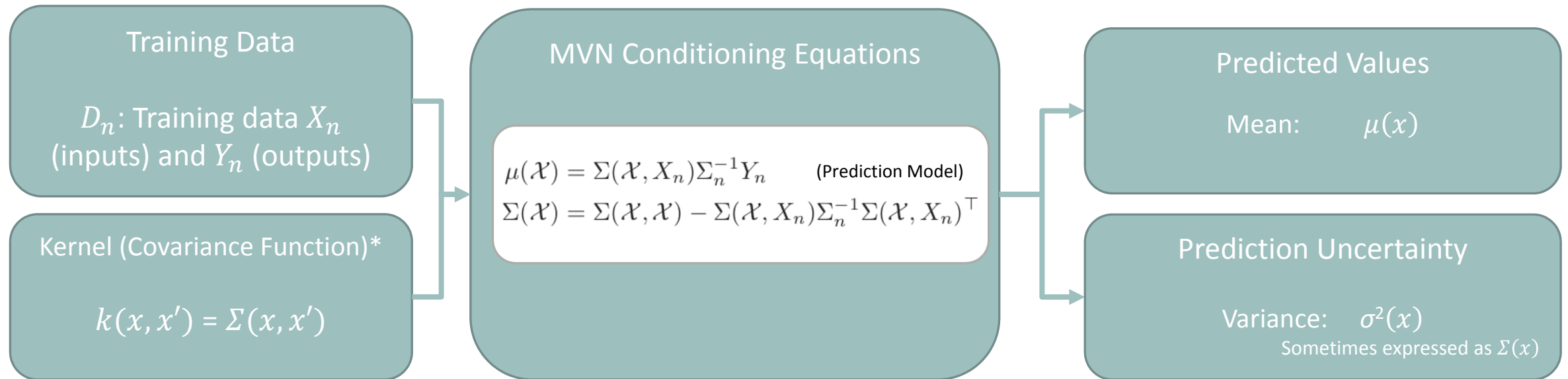


Updated Prediction Model
(surrogate model, meta model, emulator)



Fit Model

Gaussian Process Regression Overview



* Hyperparameter optimization is part of the procedure but not covered in this presentation

Multivariate Normal (MVN) Conditioning Equations

The following must be calculated: Covariance Matrix, Mean and Variance

Covariance Matrix

$$\Sigma = \begin{pmatrix} \Sigma(\chi, \chi) & \Sigma(\chi, X_n) \\ \Sigma(X_n, \chi) & \Sigma_n = \Sigma(X_n, X_n) \end{pmatrix}$$

X_n : Training locations
 χ : Testing (predictive) locations

Apply the covariance function $\Sigma(x, x')$ (kernel $k(x, x')$)

- $\Sigma(\chi, \chi)$: Covariance between testing (predictive) locations and themselves
- $\Sigma(\chi, X_n)$: Covariance between testing (predictive) and training locations
- $\Sigma(X_n, \chi)$: Covariance between training and testing (predictive) locations, which is the transpose of $\Sigma(\chi, X_n)$
- $\Sigma_n = \Sigma(X_n, X_n)$: Covariance between training locations and themselves

MVN Conditioning Equations (Mean and Variance)

Also referred to as “Gaussian process regression,” “kriging” or “kriging equations”

mean $\mu(\chi) = \Sigma(\chi, X_n) \Sigma_n^{-1} Y_n$ Prediction Model (Vary χ to make predictions)

and variance $\Sigma(\chi) = \Sigma(\chi, \chi) - \Sigma(\chi, X_n) \Sigma_n^{-1} \Sigma(X_n, \chi)^\top$ Prediction Uncertainty

Example 1

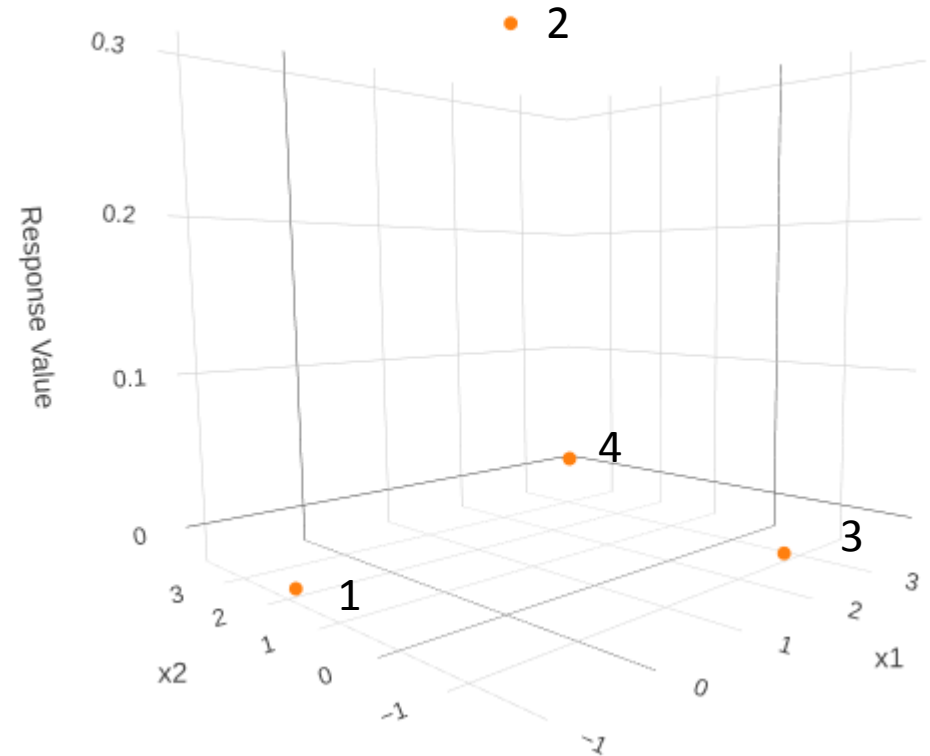
Example 1

Suppose a black box function was executed at 4 different samples (x_1 , x_2 combinations)

With limited data (x and y), what does the response surface look like?

Training Data

Sample	x_1	x_2	y
1	-1.03	1.76	-1.56E-02
2	.49	.49	3.04E-01
3	1.77	-1.77	3.38E-03
4	3.62	3.76	5.43E-12



Training Data and Testing (Predictive) Locations

Suppose you have the following training data (X_n and Y_n) and testing locations (χ)

- X_n : The training design consists of 4 points
- χ : The test design (locations to make predictions) consists of 2 points

$$X = \begin{bmatrix} \chi \\ X_n \end{bmatrix} = \begin{bmatrix} .35 & .69 \\ .65 & .46 \\ -1.03 & 1.76 \\ .49 & .49 \\ 1.77 & -1.77 \\ 3.62 & 3.76 \end{bmatrix}$$

$$-\begin{bmatrix} y^* \\ Y_n \end{bmatrix} = \begin{bmatrix} ? \\ ? \\ -1.56e-02 \\ 3.04e-01 \\ 3.38e-03 \\ 5.43e-12 \end{bmatrix}$$

The goal is make predictions (y^*) for points in χ

Note

- X_n : inputs of the training data
- Y_n : outputs of the training data
- χ or x : inputs of the testing data (predictive locations, i.e. points to make predictions)
- y^* : predicted outputs
- D_n : Training data X_n and Y_n

X : upper case of Greek letter chi (pronounced kai in English)
 χ : lower case of Greek letter chi

Calculation of the Covariance Matrix

1. Select a covariance (kernel) function

- Many covariance functions (kernels) exist: Radial Basis Function (RBF), Matern 5/2, 3/2, Exponential, ...
- For this example, a form of the RBF covariance function is used. This covariance function is described as the “inverse exponentiated squared Euclidean distance”

$$k(x, x') = \Sigma(x, x') = \exp\{-\|x - x'\|^2\} = e^{-\|x-x'\|^2}$$

2. Calculate D (Distance Matrix)

$$D = \|X - X\|^2 \quad \text{“Norm between } X \text{ and } X, \text{ squared”}$$

3. Calculate Σ (Covariance Matrix)

$$\Sigma = e^{-D}$$

Calculation of D

$D =$

$$\sqrt{(.35 - .35)^2 + (.69 - .69)^2}^2 = 0$$

$$\sqrt{(.35 - .65)^2 + (.69 - .46)^2}^2 = .1429$$

$$\sqrt{(.35 - -1.03)^2 + (.69 - 1.76)^2}^2 = 3.0493$$

$$\sqrt{(.35 - .49)^2 + (.69 - .49)^2}^2 = .0596$$

$$\sqrt{(.35 - 1.77)^2 + (.69 - -1.77)^2}^2 = 8.068$$

$$\sqrt{(.35 - 3.62)^2 + (.69 - 3.76)^2}^2 = 20.1178$$

$$.1429$$

$$\sqrt{(.65 - .65)^2 + (.46 - .46)^2}^2 = 0$$

$$\sqrt{(.65 - -1.03)^2 + (.46 - 1.76)^2}^2 = 4.5124$$

$$\sqrt{(.65 - .49)^2 + (.46 - .49)^2}^2 = .0265$$

$$\sqrt{(.65 - 1.77)^2 + (.46 - -1.77)^2}^2 = 6.2273$$

$$\sqrt{(.65 - 3.62)^2 + (.46 - 3.76)^2}^2 = 19.7109$$

$$3.0493$$

$$4.5124$$

$$\sqrt{(-1.03 - -1.03)^2 + (1.76 - 1.76)^2}^2 = 0$$

$$\sqrt{(-1.03 - .49)^2 + (1.76 - .49)^2}^2 = 3.9233$$

$$\sqrt{(-1.03 - 1.77)^2 + (1.76 - -1.77)^2}^2 = 20.3009$$

$$\sqrt{(-1.03 - 3.62)^2 + (1.76 - 3.76)^2}^2 = 25.6225$$

$$.0596$$

$$.0265$$

$$3.9233$$

$$\sqrt{(.49 - .49)^2 + (.49 - .49)^2}^2 = 0$$

$$\sqrt{(.49 - 1.77)^2 + (.49 - -1.77)^2}^2 = 6.746$$

$$\sqrt{(.49 - 3.62)^2 + (.49 - 3.76)^2}^2 = 20.4898$$

$$8.068$$

$$6.2273$$

$$20.3009$$

$$6.746$$

$$\sqrt{(1.77 - 1.77)^2 + (-1.77 - -1.77)^2}^2 = 0$$

$$\sqrt{(1.77 - 3.62)^2 + (-1.77 - 3.76)^2}^2 = 34.0034$$

$$20.1178$$

$$19.7109$$

$$25.6225$$

$$20.4898$$

$$34.0034$$

$$\sqrt{(3.62 - 3.62)^2 + (3.76 - 3.76)^2}^2 = 0$$

Calculation of Σ

$$\Sigma = \begin{bmatrix} e^0 = 1 & e^{-.1429} = .8668 & e^{-3.0493} = .0474 & e^{-.0596} = .9421 & e^{-8.068} = .0003 & e^{-20.1178} = 1.832\text{e-}9 \\ .8668 & e^0 = 1 & e^{-4.5124} = .0110 & e^{-.0265} = .9738 & e^{-6.2273} = .0020 & e^{-19.7109} = 2.8\text{e-}9 \\ .0474 & .0110 & e^0 = 1 & e^{-3.9233} = .0198 & e^{-20.3009} = 1.5\text{e-}9 & e^{-25.6225} = 7.5\text{e-}12 \\ .9421 & .9738 & .0198 & e^0 = 1 & e^{-6.746} = .0012 & e^{-20.4898} = 1.263\text{e-}9 \\ .0003 & .0020 & 1.5\text{e-}9 & .0012 & e^0 = 1 & e^{-34.0034} = 1.7\text{e-}15 \\ 1.832\text{e-}9 & 2.8\text{e-}9 & 7.5\text{e-}12 & 1.263\text{e-}9 & 1.7\text{e-}15 & e^0 = 1 \end{bmatrix}$$

Calculation of Σ

$\Sigma =$

$$\Sigma(\chi, \chi)$$

$e^0 = 1$	$e^{-1.429} = .8668$
$.8668$	$e^0 = 1$

$$\Sigma(\chi, X_n)$$

$e^{-3.0493} = .0474$	$e^{-0.596} = .9421$	$e^{-8.068} = .0003$	$e^{-20.1178} = 1.832e-9$
$e^{-4.5124} = .0110$	$e^{-0.265} = .9738$	$e^{-6.7273} = .0020$	$e^{-19.7109} = 2.8e-9$

$$\Sigma(X_n, \chi)$$

$.0474$	$.0110$
$.9421$	$.9738$
$.0003$	$.0020$
$1.832e-9$	$2.8e-9$

$$\Sigma_n = \Sigma(X_n, X_n)$$

$e^0 = 1$	$e^{-3.9233} = .0198$	$e^{-20.3009} = 1.5e-9$	$e^{-25.6225} = 7.5e-12$
$.0198$	$e^0 = 1$	$e^{-6.745} = .0012$	$e^{-20.4898} = 1.263e-9$
$1.5e-9$	$.0012$	$e^0 = 1$	$e^{-34.0034} = 1.7e-15$
$7.5e-12$	$1.263e-9$	$1.7e-15$	$e^0 = 1$

Since Σ is symmetric, note that $\Sigma(X_n, \chi) = \Sigma(\chi, X_n)^T$

Calculation of Predictive Quantities

The MVN conditioning equations are used to determine the predictive quantities mean and variance

mean $\mu(\mathcal{X}) = \Sigma(\mathcal{X}, X_n) \Sigma_n^{-1} Y_n$

$$\mu(\chi) = y * = \begin{pmatrix} 0.2849657 \\ 0.2954011 \end{pmatrix} \quad \text{Predicted values for locations in } \chi$$

and variance $\Sigma(\mathcal{X}) = \Sigma(\mathcal{X}, \mathcal{X}) - \Sigma(\mathcal{X}, X_n) \Sigma_n^{-1} \Sigma(\mathcal{X}, X_n)^\top$

$$\Sigma(\chi) = \begin{pmatrix} 0.11154162 & -0.05042265 \\ -0.05042265 & 0.05155061 \end{pmatrix} \quad \text{Prediction Uncertainty}$$

The diagonal terms are the variances at prediction points 1 and 2

$$\sigma^2(\chi) = \begin{pmatrix} 0.11154162 \\ 0.05155061 \end{pmatrix}$$

R

Code to replicate this example in R

```
library(plgp)

eps = sqrt(.Machine$double.eps)

# Training points
X = rbind(c(-1.03, 1.76), c(.49, .49), c(1.77, -1.77), c(3.62, 3.76))

# The goal is to fit this function:  $y(x) = x_1 * \exp(-x_1^2 - x_2^2)$ 
y = X[,1] * exp(-X[,1]^2 - X[,2]^2)

# Test points
XX = rbind(c(.35, .69), c(.65, .46))
XX

# Sigma 22 (Sigma) and its inverse (Si)
# #####
# Distance among the Training Data
D = distance(X)
Sigma = exp(-D)
Si = solve(Sigma)

# Sigma 11
# #####
# Distance among the Testing Data
DXX = distance(XX)
SXX = exp(-DXX)

# Sigma 12 and Sigma 21 (Transpose of Sigma 12)
# #####
# Distance between training and testing data
DX = distance(XX, X)
SX = exp(-DX)

# Calculate the predictive mean and predictive variance
# #####
# Predictive mean
mup = SX %*% Si %*% y
mup

# Predictive variance
Sigmap = SXX - SX %*% Si %*% t(SX)
Sigmap
```

Output

```
> # #####
> # Predictive mean
> mup = SX %*% Si %*% y
> mup
      [,1]
[1,] 0.2849657
[2,] 0.2954011
>
> # Predictive variance
> Sigmap = SXX - SX %*% Si %*% t(SX)
> Sigmap
      [,1]      [,2]
[1,] 0.11154162 -0.05042265
[2,] -0.05042265 0.05155061
> |
```

Mean

Variance

R

Code to replicate this example in R with Plots

```
library(plgp)
library(lhs)

eps = sqrt(.Machine$double.eps)

# Training Data
# #####
# Training points
number_of_sample_points = 4
X = rbind(c(-1.03,1.76), c(.49,.49), c(1.77,-1.77), c(3.62,3.76))

# Observed values
# The goal is to fit this function:  $y(x) = x_1 * \exp(-x_1^2 - x_2^2)$ 
y = X[,1] * exp(-X[,1]^2 - X[,2]^2)

# Testing Data
# #####
# Test points
number_of_test_points_per_axis = 40
xx = seq(-2, 4, length=number_of_test_points_per_axis)
XX = expand.grid(xx, xx)

# Sigma 22 (Sigma) and its inverse (Si)
# #####
# Distance among the Training Data
D = distance(X)
Sigma = exp(-D) + diag(eps, nrow(X))
Si = solve(Sigma)

# Sigma 11
# #####
# Distance among the Testing Data
```

```
DXX = distance(XX)
SXX = exp(-DXX)

# Sigma 12 and Sigma 21 (Transpose of Sigma 12)
# #####
# Distance between training and testing data
DX = distance(XX, X)
SX = exp(-DX)

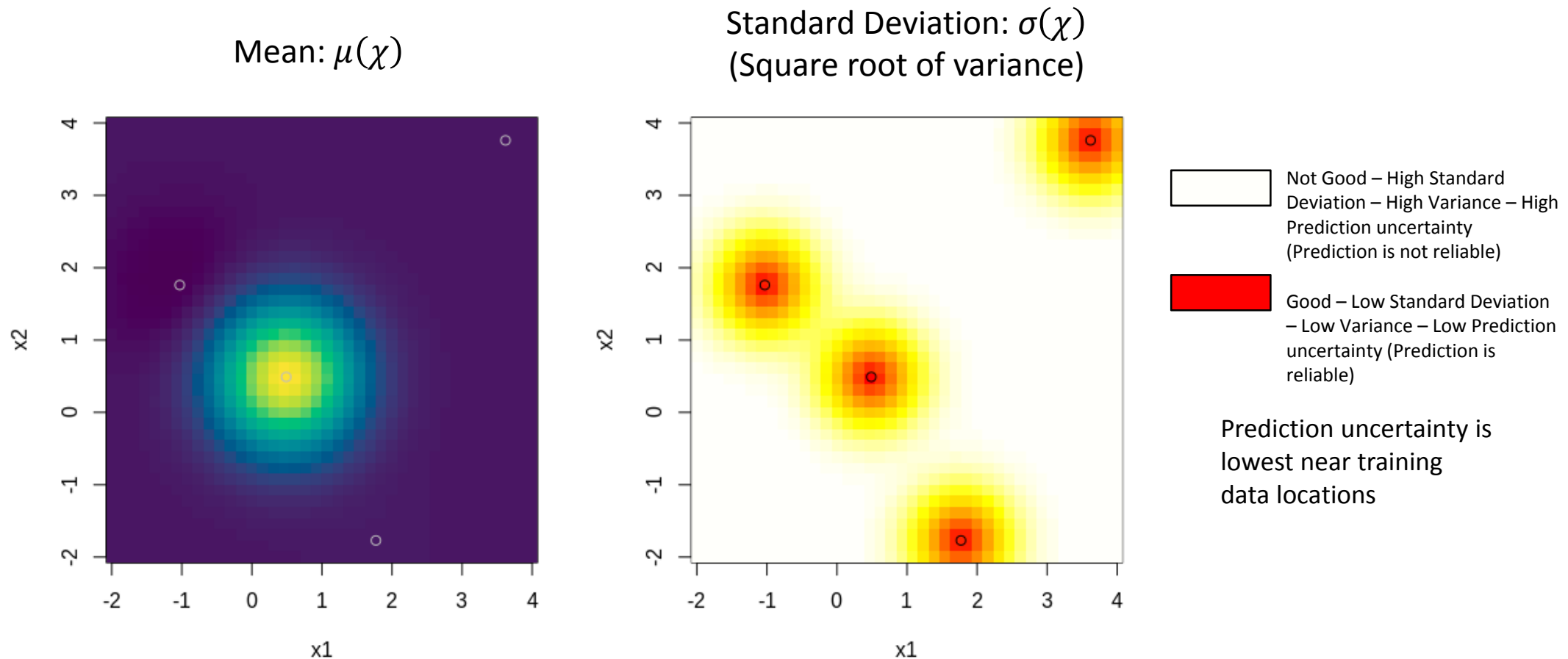
# Calculate the predictive mean and predictive variance
# #####
mup = SX %*% Si %*% y
Sigmap = SXX - SX %*% Si %*% t(SX)

# Predictive standard deviation
diag(Sigmap)
sdp = sqrt(diag(Sigmap))

# Figure 5.5
par(mfrow=c(1, 2))
cols_a = hcl.colors(128, palette = "viridis")
cols_b = heat.colors(128)
image(xx, xx, matrix(mup, ncol=length(xx)), xlab='x1', ylab='x2', col=cols_a)
points(X[,1], X[,2])
image(xx, xx, matrix(sdp, ncol=length(xx)), xlab='x1', ylab='x2', col=cols_b)
points(X[,1], X[,2])

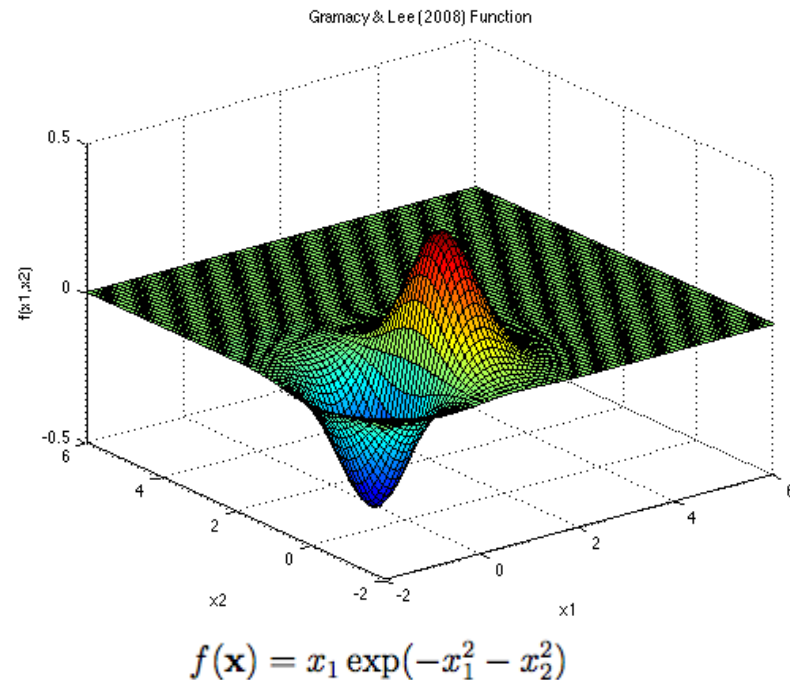
# Figure 5.6
persp(xx, xx, matrix(mup, ncol=number_of_test_points_per_axis), theta=-30, phi=30,
xlab='x1', ylab='x2', zlab='y', zlim = c(-.5,.5))
```

Predictive Quantities Mean and Standard Deviation



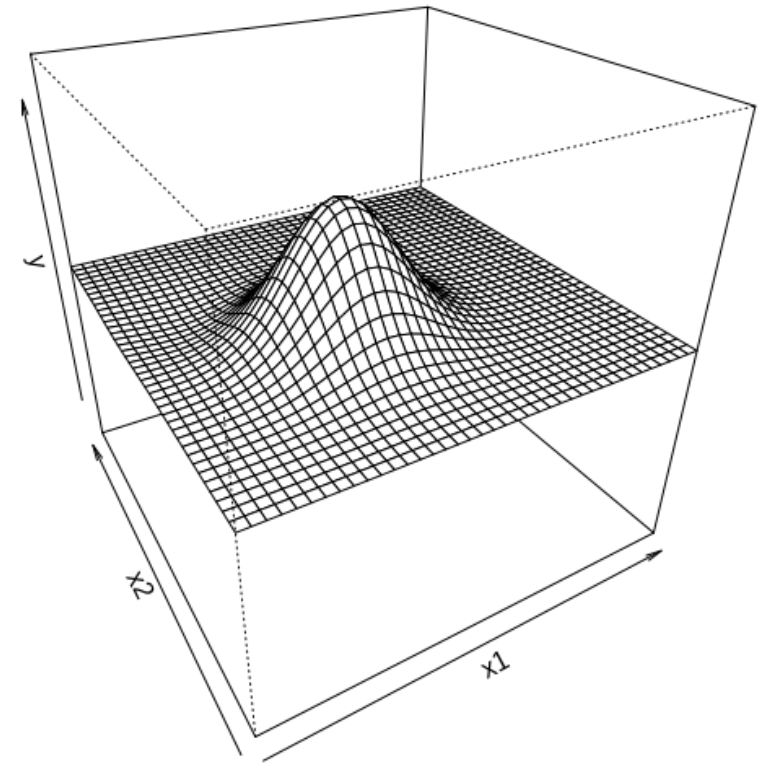
Comparison of True Function and Prediction Model

True Function



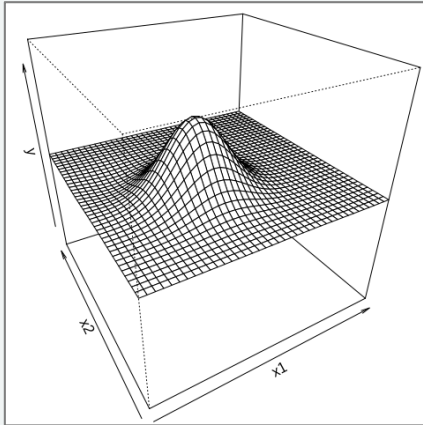
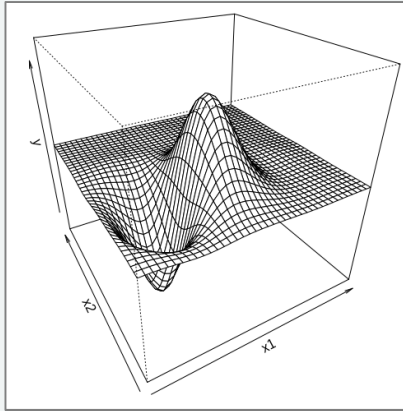
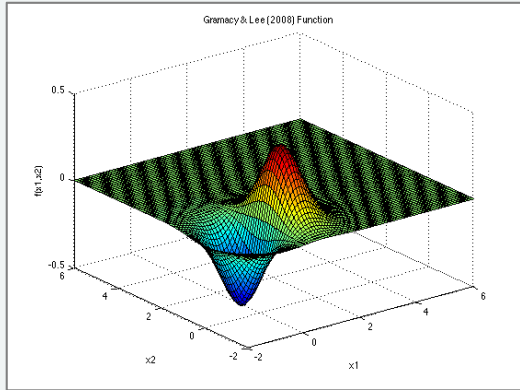
Source: <https://www.sfu.ca/~ssurjano/grlee08.html>

Prediction Model ($\mu(\chi)$)



Example 2

Example 2

	Example 1	Example 2	True Function
X_n : The training design	4 Points	40 Points	
	<p>Predicted Model ($\mu(\chi)$) 4 Training Points</p> 	<p>Predicted Model ($\mu(\chi)$) 40 Training Points</p> 	<p>True Function ($f(x)$)</p> 

R

Code to replicate this example in R with more training data

```
library(plgp)
library(lhs)

eps = sqrt(.Machine$double.eps)

# Training Data
# #####
# Training points
number_of_sample_points = 40
X = randomLHS(number_of_sample_points, 2)
X[,1] = (X[,1] - .5) * 6 + 1
X[,2] = (X[,2] - .5) * 6 + 1

# Observed values
# The goal is to fit this function:  $y(x) = x_1 * \exp(-x_1^2 - x_2^2)$ 
y = X[,1] * exp(-X[,1]^2 - X[,2]^2)

# Testing Data
# #####
# Test points
number_of_test_points_per_axis = 40
xx = seq(-2, 4, length=number_of_test_points_per_axis)
XX = expand.grid(xx, xx)

# Sigma 22 (Sigma) and its inverse (Si)
# #####
# Distance among the Training Data
D = distance(X)
Sigma = exp(-D) + diag(eps, nrow(X))
Si = solve(Sigma)

# Sigma 11

# #####
# Distance among the Testing Data
DXX = distance(XX)
SXX = exp(-DXX)

# Sigma 12 and Sigma 21 (Transpose of Sigma 12)
# #####
# Distance between training and testing data
DX = distance(XX, X)
SX = exp(-DX)

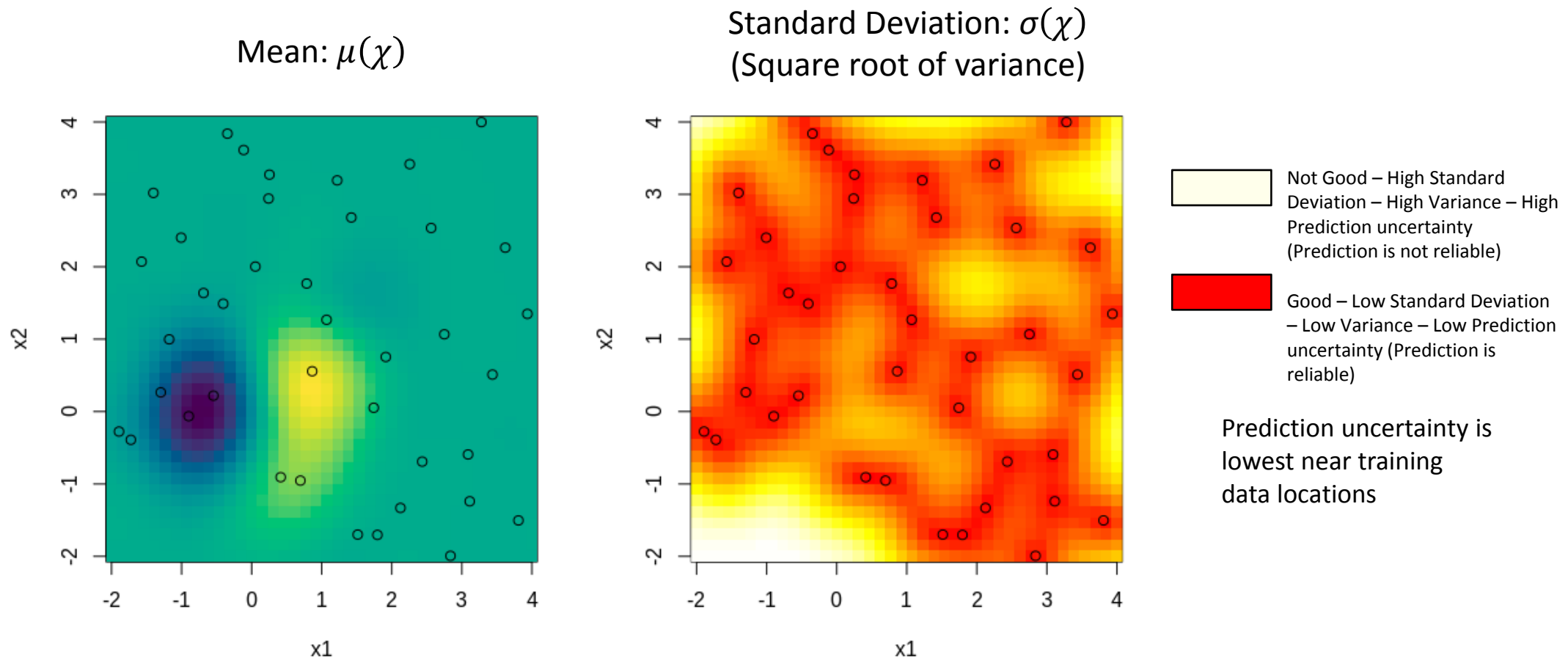
# Calculate the predictive mean and predictive variance
# #####
mup = SX %*% Si %*% y
Sigmap = SXX - SX %*% Si %*% t(SX)

# Predictive standard deviation
diag(Sigmap)
sdp = sqrt(diag(Sigmap))

# Figure 5.5
par(mfrow=c(1, 2))
cols_a = hcl.colors(128, palette = "viridis")
cols_b = heat.colors(128)
image(xx, xx, matrix(mup, ncol=length(xx)), xlab='x1', ylab='x2', col=cols_a)
points(X[,1], X[,2])
image(xx, xx, matrix(sdp, ncol=length(xx)), xlab='x1', ylab='x2', col=cols_b)
points(X[,1], X[,2])

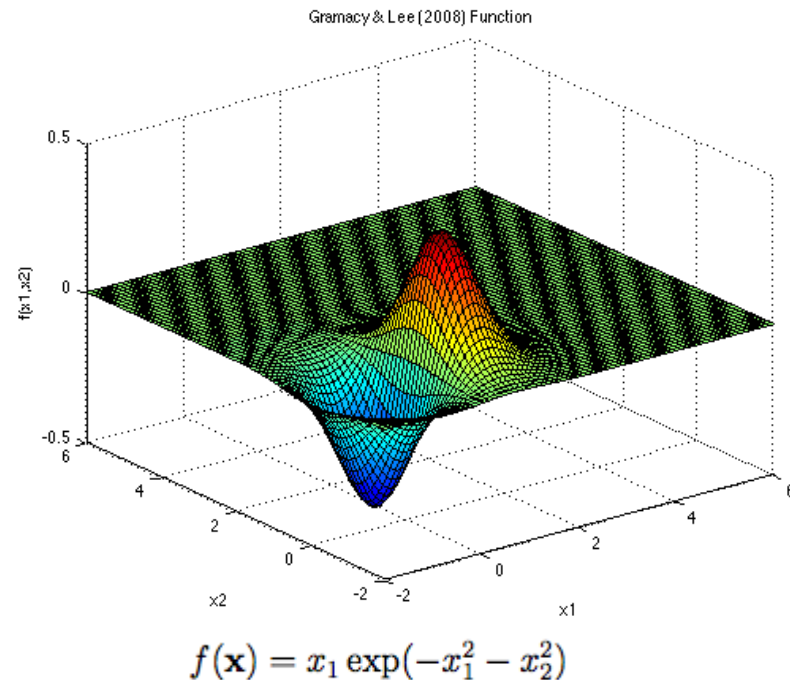
# Figure 5.6
persp(xx, xx, matrix(mup, ncol=number_of_test_points_per_axis), theta=-30, phi=30,
xlab='x1', ylab='x2', zlab='y', zlim = c(-.5,.5))
```

Predictive Quantities Mean and Standard Deviation



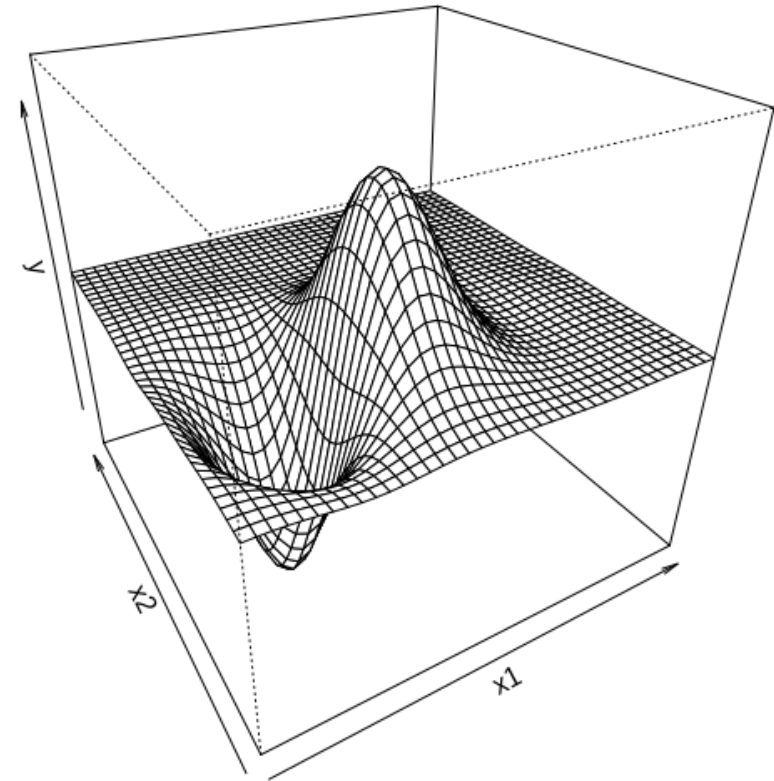
Comparison of True Function and Prediction Model

True Function

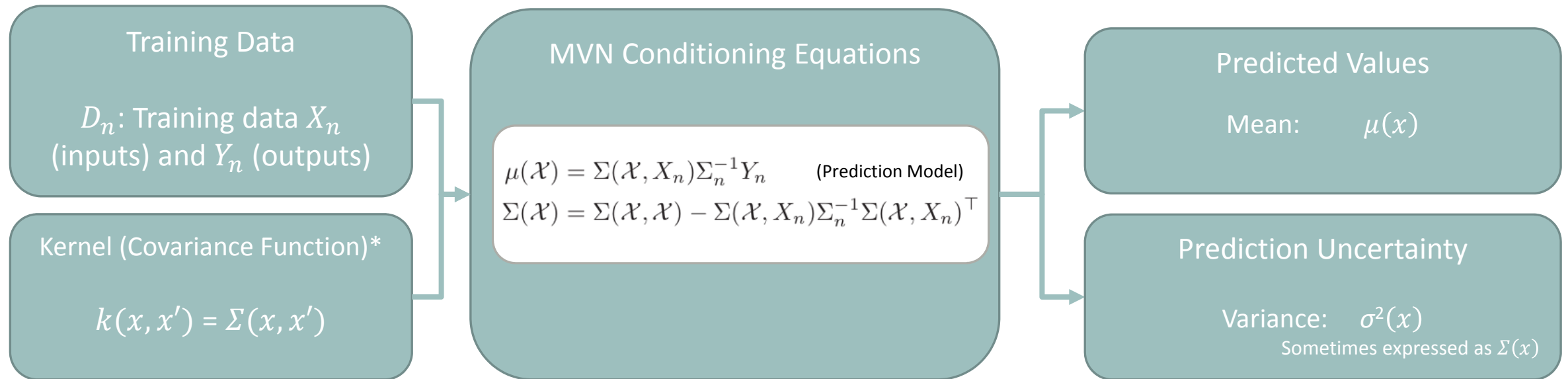


Source: <https://www.sfu.ca/~ssurjano/grlee08.html>

Prediction Model ($\mu(\chi)$)



Gaussian Process Regression Overview



* Hyperparameter optimization is part of the procedure but not covered in this presentation

Goals

Sequential Design/Active Learning

Goal - Learn how to:

1. Fit model
 - MVN Conditioning Equations
2. Choose next design point
 - Acquisition Functions (Criteria)

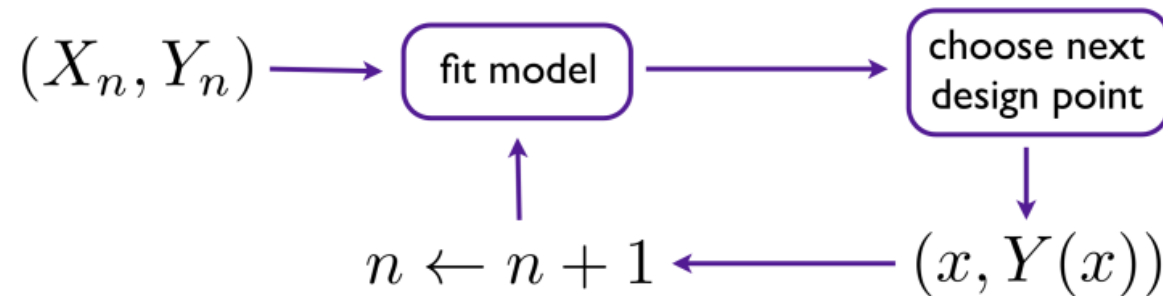


FIGURE 6.8: Diagram of sequential design/active learning/design augmentation.

Explanation of Sequential Design/Active Learning

1. The black box function (MSC Nastran) is evaluated at different samples (X_n). The outputs (Y_n), or responses, are collected.

2. The training data (X_n, Y_n) is used in the MVN equations to yield the mean $\mu(\chi)$ and variance functions $\sigma^2(\chi)$.

3. Acquisition functions are used to determine a new point x that may yield a better sample.

(X_n, Y_n)

fit model

choose next
design point

5. Since a new training point ($x, Y(x)$) is available, the model is updated and the process is repeated.

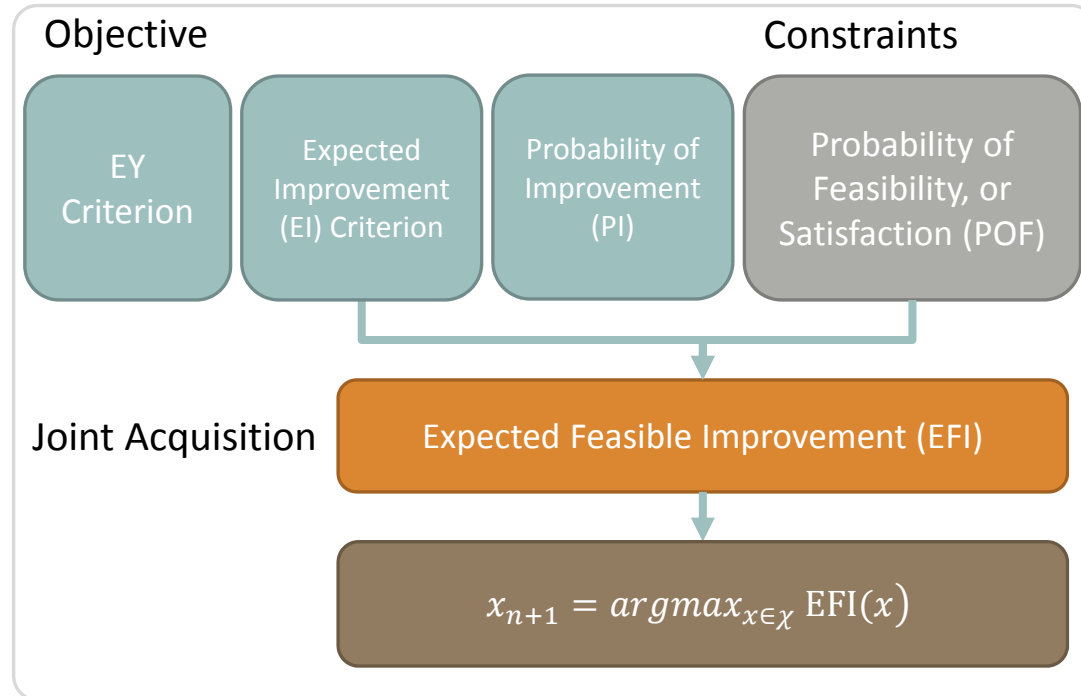
$n \leftarrow n + 1$

$(x, Y(x))$

4. $Y(x)$: The black box function (MSC Nastran) is evaluated at this new x .

Acquisition Functions

Acquisition Functions



Bayesian
Optimization

Bayesian Optimization

“Suppose we have a function $f : \mathcal{X} \rightarrow \mathbb{R}$ that we wish to minimize on some domain $X \subseteq \mathcal{X}$. That is, we wish to find

$$x^* = \arg \min_{x \in X} f(x).$$

In numerical analysis, this problem is typically called (global) *optimization* and has been the subject of decades of study. We draw a distinction between global optimization, where we seek the absolute optimum in X , and local optimization, where we seek to find a local optimum in the neighborhood of a given initial point x_0 .

A common approach to optimization problems is to make some assumptions about f . For example, when the objective function f is known to be convex and the domain X is also convex, the problem is known as *convex optimization* and has been widely studied. Convex optimization is a common tool used across machine learning.

If an exact functional form for f is not available (that is, f behaves as a ‘black box’), what can we do? Bayesian optimization proceeds by maintaining a probabilistic belief about f and designing a so called *acquisition function* to determine where to evaluate the function next. Bayesian optimization is particularly well-suited to global optimization problems where f is an *expensive* black-box function; for example, evaluating f might require running an expensive simulation. Bayesian optimization has recently become popular for training expensive machine-learning models whose behavior depend in a complicated way on their parameters (e.g., convolutional neural networks). This is an example of the ‘AutoML’ paradigm.”

Garnett, R. (2015, March 16). *Lecture 12: Bayesian Optimization*. Lecture presented at CSE 515T: Bayesian Methods in Machine Learning – Spring 2015 in Washington University in St. Louis, St. Louis.

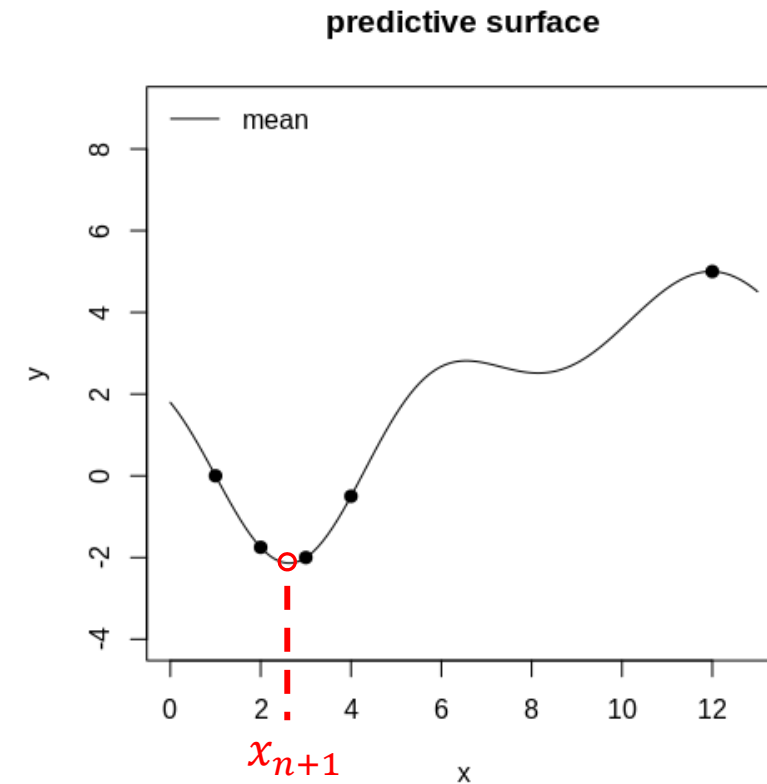
Acquisition Functions

Objective

EY
Criterion

EY Criterion

$$x_{n+1} = \operatorname{argmin}_{x \in \mathcal{X}} \mu(x)$$



Acquisition Functions

Objective

EY
Criterion

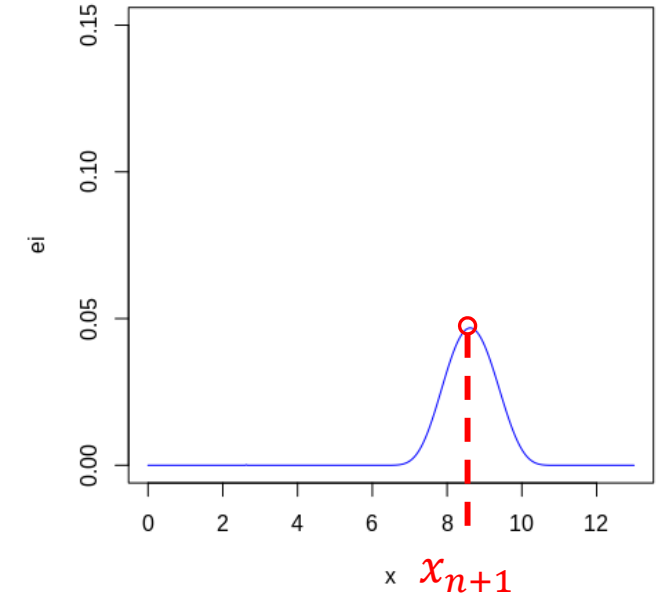
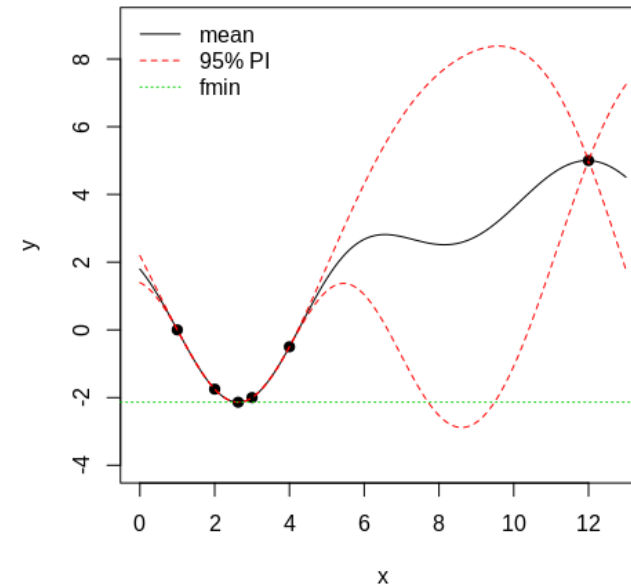
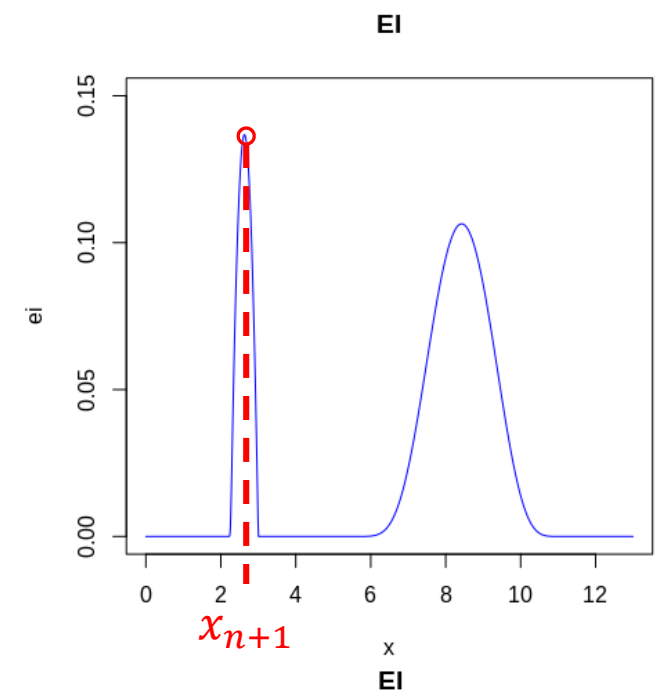
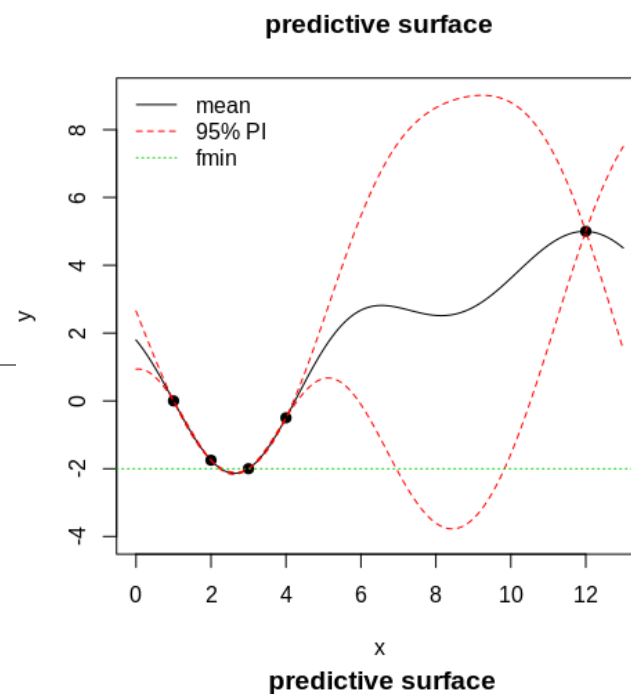
Expected
Improvement
(EI) Criterion

Expected Improvement (EI) Criterion

$$\text{EI}(x) = (f_{\min}^n - \mu_n(x)) \Phi\left(\frac{f_{\min}^n - \mu_n(x)}{\sigma_n(x)}\right) + \sigma_n(x) \phi\left(\frac{f_{\min}^n - \mu_n(x)}{\sigma_n(x)}\right)$$

= exploitation + exploration

$$x_{n+1} = \operatorname{argmax}_{x \in \mathcal{X}} \text{EI}(x)$$



Acquisition Functions

Objective



Expected Improvement (EI) Criterion

$$\text{EI}(x) = (f_{\min}^n - \mu_n(x)) \Phi\left(\frac{f_{\min}^n - \mu_n(x)}{\sigma_n(x)}\right) + \sigma_n(x) \phi\left(\frac{f_{\min}^n - \mu_n(x)}{\sigma_n(x)}\right)$$

= exploitation + exploration

$$x_{n+1} = \operatorname{argmax}_{x \in \mathcal{X}} \text{EI}(x)$$

- n : number of function evaluations
- f_{\min}^n : Best observed objective value
- $\mu(x)$: Mean function from the MVN conditioning equations
- $\sigma^2(x)$: Variance function from the MVN conditioning equations
- Φ : cumulative distribution function (CDF) of the n -dimensional multivariate normal distribution (Uppercase Phi)
- ϕ : probability density function (PDF) of the n -dimensional multivariate normal distribution (Lowercase Phi)

Acquisition Functions

Objective

EY
Criterion

Expected
Improvement
(EI) Criterion

Expected Improvement (EI) Criterion

$$\text{EI}(x) = (f_{\min}^n - \mu_n(x)) \Phi\left(\frac{f_{\min}^n - \mu_n(x)}{\sigma_n(x)}\right) + \sigma_n(x) \phi\left(\frac{f_{\min}^n - \mu_n(x)}{\sigma_n(x)}\right)$$

= exploitation + exploration

$$x_{n+1} = \operatorname{argmax}_{x \in \mathcal{X}} \text{EI}(x)$$

More about Φ and ϕ

▼ Multivariate Normal Distribution

The multivariate normal distribution is a generalization of the univariate normal distribution to two or more variables. It has two parameters, a mean vector μ and a covariance matrix Σ , that are analogous to the mean and variance parameters of a univariate normal distribution. The diagonal elements of Σ contain the variances for each variable, and the off-diagonal elements of Σ contain the covariances between variables.

The probability density function (pdf) of the d -dimensional multivariate normal distribution is

$$y = f(x, \mu, \Sigma) = \frac{1}{\sqrt{|\Sigma|(2\pi)^d}} \exp\left(-\frac{1}{2}(x-\mu)' \Sigma^{-1}(x-\mu)\right)$$

where x and μ are 1-by- d vectors and Σ is a d -by- d symmetric, positive definite matrix. Only `mvnrnd` allows positive semi-definite Σ matrices, which can be singular. The pdf cannot have the same form when Σ is singular.

The multivariate normal cumulative distribution function (cdf) evaluated at x is the probability that a random vector v , distributed as multivariate normal, lies within the semi-infinite rectangle with upper limits defined by x :

$$\Pr\{v(1) \leq x(1), v(2) \leq x(2), \dots, v(d) \leq x(d)\}.$$

Although the multivariate normal cdf does not have a closed form, `mvncdf` can compute cdf values numerically.

Source: <https://www.mathworks.com/help/stats/mvnpdf.html>

Acquisition Functions

Objective



When EI does not yield a good solution, use PI.

For example, for optimization problems with 10 or more parameters, EI performs poorly but PI performs well.

Probability of Improvement (PI) Criterion

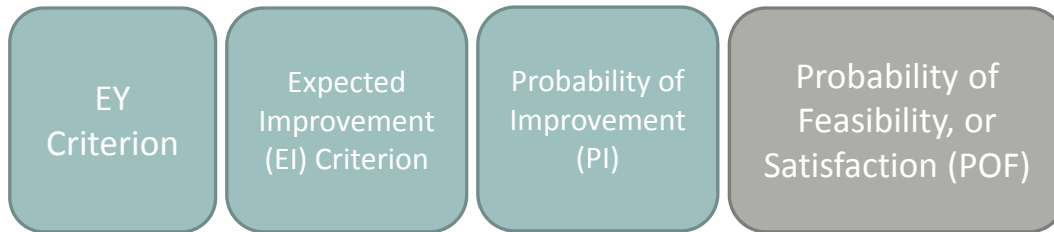
$$PI(x) = \Phi \left(\frac{f_{\min}^n - \mu_n(x)}{\sigma_n(x)} \right)$$

= exploitation

$$x_{n+1} = \operatorname{argmax}_{x \in \mathcal{X}} PI(x)$$

Acquisition Functions

Objective



Constraints

Probability of Feasibility, or Satisfaction (POF)

- Suppose you have constraints of this form
$$c(x) \leq 0$$
- Probability of satisfaction (feasibility) (POS or POF)

$$\begin{aligned} pof &= \prod_{j=1}^m p_n^{(j)}(x) \\ &= p_n^{(1)}(x) \cdot p_n^{(2)}(x) \cdots p_n^{(m)}(x) \end{aligned}$$

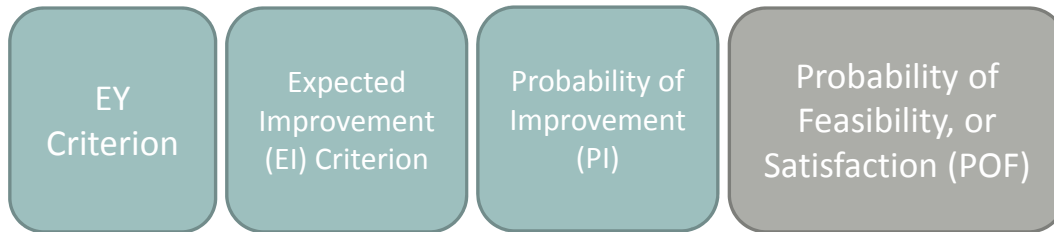
where

$$p_n^{(j)}(x) = \Phi \left(-\frac{\mu_n^{(j)}(x)}{\sigma_n^{(j)}(x)} \right)$$

- n : number of function evaluations
- m : Number of normalized constraints
- j : The j th normalized constraint
- $p_n^{(j)}(x)$: Probability of satisfaction (feasibility) for normalized constraint j
- Π : Product of a sequence (Greek uppercase letter pi), similar to addition of a sequence (Σ) but you perform the product operation instead
- $\mu(x)$: Mean function from the MVN conditioning equations
- $\sigma^2(x)$: Variance function from the MVN conditioning equations
- Φ : cumulative distribution function (CDF) of the n -dimensional multivariate normal distribution

Acquisition Functions

Objective



Constraints

Probability of Feasibility, or Satisfaction (POF)

- Suppose you have constraints of this form

$$c(x) \leq 0$$

- Probability of satisfaction (feasibility) (POS or POF)

$$\begin{aligned} pof &= \prod_{j=1}^m p_n^{(j)}(x) \\ &= p_n^{(1)}(x) \cdot p_n^{(2)}(x) \cdots p_n^{(m)}(x) \end{aligned}$$

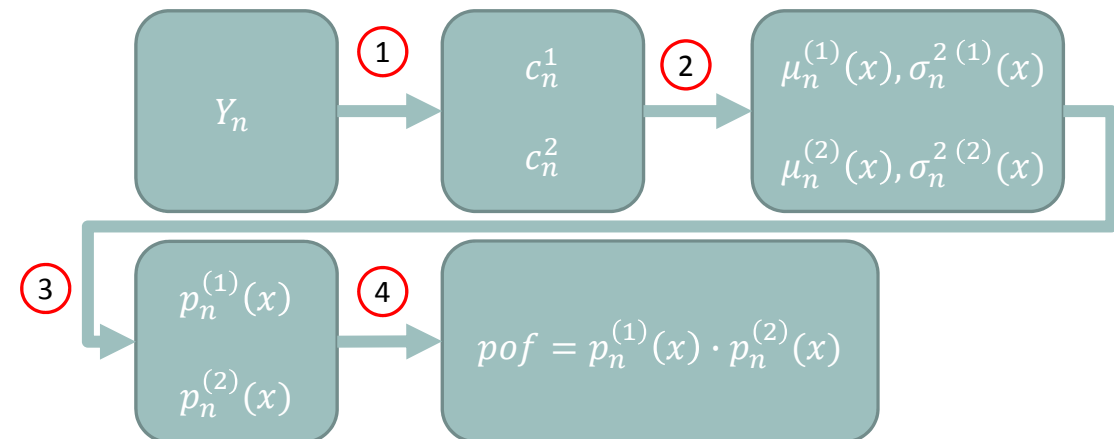
where

$$p_n^{(j)}(x) = \Phi \left(-\frac{\mu_n^{(j)}(x)}{\sigma_n^{(j)}(x)} \right)$$

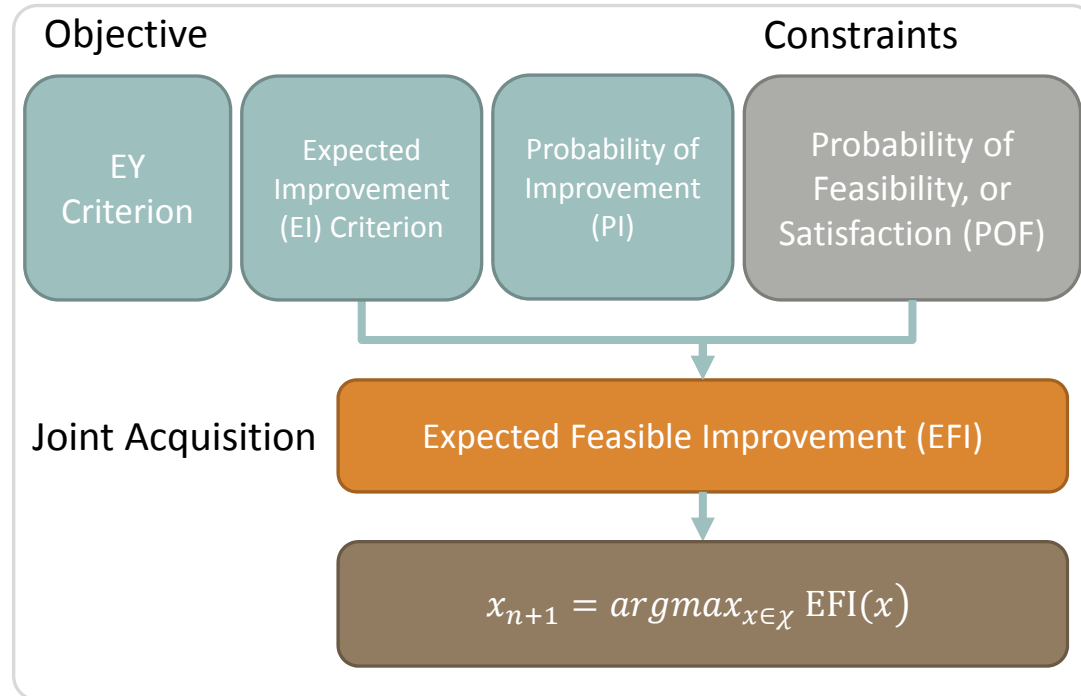
Suppose you have this constraint,

$$-15,000 < \sigma_{Stress} < 20,000$$

1. The training data is used to create normalized constraints for the lower and upper bounds
2. You need to calculate the mean and variance for both the lower and upper bounds
3. Then you calculate the probability for all normalized constraints
4. And take the product



Acquisition Functions



Expected Feasible Improvement (EFI)

$$\operatorname{EFI}(x) = \operatorname{EI}(x) \cdot \operatorname{POF}(x)$$

$$x_{n+1} = \operatorname{argmax}_{x \in \mathcal{X}} \operatorname{EFI}(x)$$

When calculating $\operatorname{EI}(x)$

- f_{min}^n : Best observed objective value of only the valid designs

Goals

Sequential Design/Active Learning

Goal - Learn how to:

1. Fit model
 - MVN Conditioning Equations
2. Choose next design point
 - Acquisition Functions (Criteria)

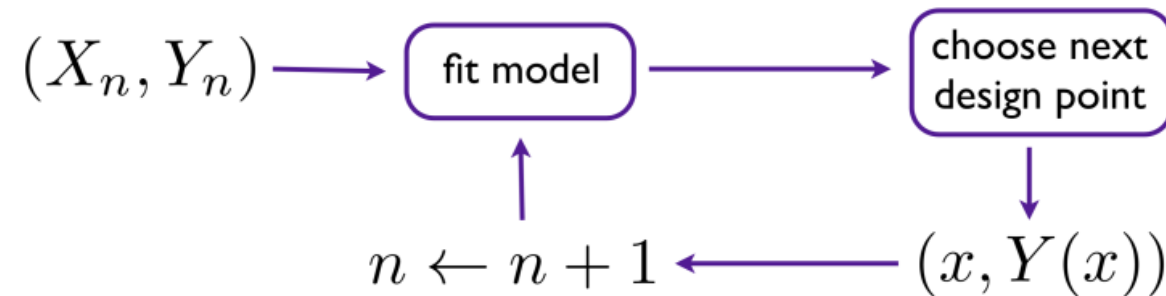


FIGURE 6.8: Diagram of sequential design/active learning/design augmentation.

Settings in the Machine Learning Web App

Explanation of Sequential Design/Active Learning

The following can be controlled:

- `n_iter`
- `acquisition_function_objective`

1. The black box function (MSC Nastran) is evaluated at different samples (X_n). The outputs (Y_n), or responses, are collected.

2. The training data (X_n, Y_n) is used in the MVN equations to yield the mean $\mu(\chi)$ and variance functions $\sigma^2(\chi)$.

3. Acquisition functions are used to determine a new point x that may yield a better sample.

(X_n, Y_n)

fit model

choose next
design point

`n_iter`

`acquisition_function_objective`

$n \leftarrow n + 1$

$(x, Y(x))$

5. Since a new training point $(x, Y(x))$ is available, the model is updated and the process is repeated.

4. $Y(x)$: The black box function (MSC Nastran) is evaluated at this new x .

Settings

The Machine Learning Web App allows you to control the following:

- `n_iter`: This is the number of machine learning iterations. The total number of MSC Nastran runs is the sum of number of samples and `n_iter`. (Default = 20)
- `acquisition_function_objective`: This specifies the criterion used to improve the objective
 - Options:
 - Expected Improvement
 - Probability of Improvement

Nastran SOL 200 Web App - Machine Learning

ParametersSamplesResponsesDownloadResults

Settings

Procedure

Machine Learning

Advanced Settings

Setting	Description	Configure
Bayesian Optimization		
<code>n_iter</code>	This is the number of machine learning iterations. The total number of MSC Nastran runs is the sum of number of samples and <code>n_iter</code> . (Default = 20)	20
Acquisition Function		
<code>acquisition_function_objective</code>	Acquisition function to use for the objective (Default = Expected Improvement)	Expected Improvement

Settings

n_iter

n_iter: This is the number of machine learning iterations. The total number of MSC Nastran runs is the sum of number of samples and n_iter. (Default = 20)

- For example, if n_iter=50, the machine learning process will run until 50 points are evaluated. The following happens in 1 iteration:
 - The model is fitted
 - The next design point is chosen by finding x_{n+1} that maximizing the acquisition function
 - Run MSC Nastran at x_{n+1}
 - Repeat
- This is similar to the DESMAX option on the DOPTPRM entry when using MSC Nastran SOL 200

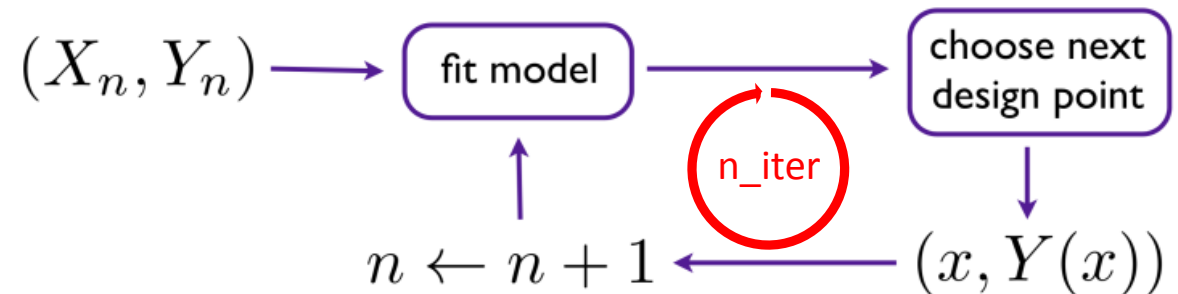


FIGURE 6.8: Diagram of sequential design/active learning/design augmentation.

Settings

acquisition_function_objective

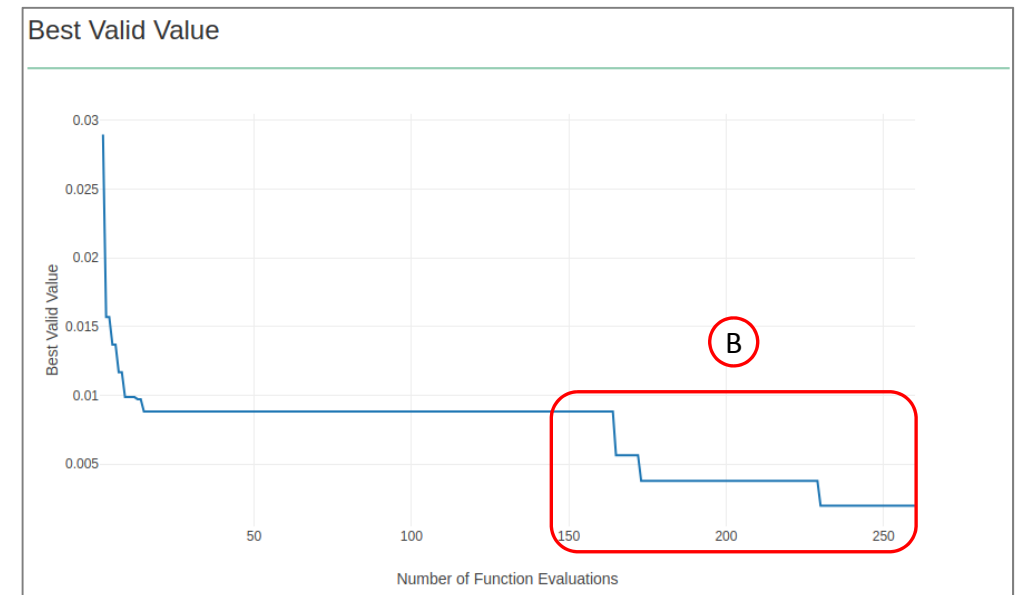
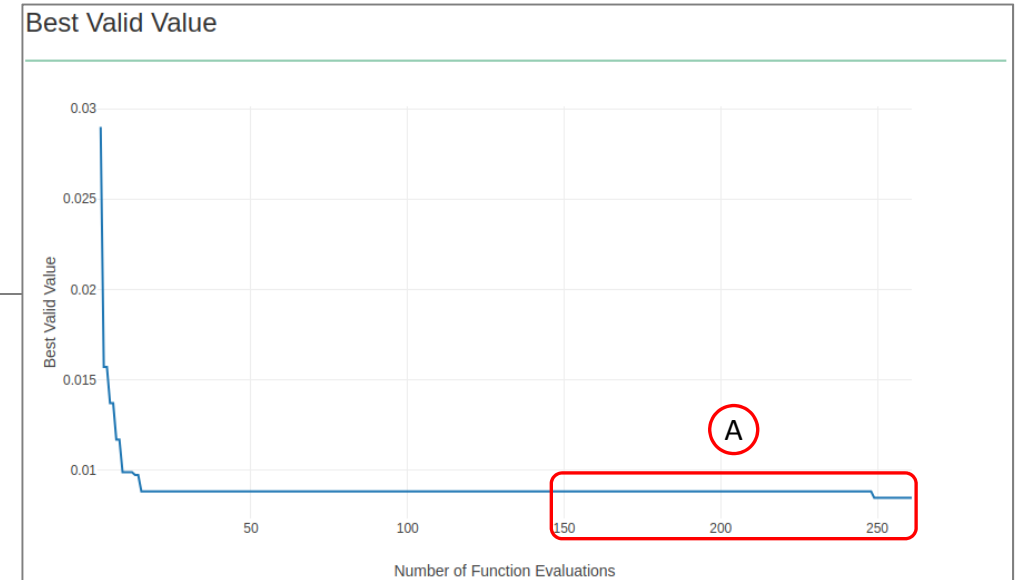
acquisition_function_objective: This specifies the criterion used to improve the objective

- Options:
 - Expected Improvement
 - Probability of Improvement

For problems with 10 or more parameters, the design space is significantly large. If the EI criterion is used, the design space is thoroughly searched for extrema but is time consuming as the design space increases. The PI is a good alternative.

Consider the optimization results of a 16 parameter optimization, shown in the images to the right.

- A. The EI criterion requires a lot of searches before a better design is found. This search is global.
- B. PI requires fewer searches. This search is local.



Conclusion

Models

- Artificial neural networks
- Decision trees
- Support vector machines
- Regression analysis
 - Linear regression
 - Polynomial regression
 - Gaussian process regression
- Bayesian networks
- Genetic algorithms

Focus of this
presentation

Types of learning algorithms

- Supervised learning
 - Active learning/Sequential Design
 - Bayesian Optimization
 - Classification
 - Regressions
- Unsupervised learning
- Semi-supervised learning
- Reinforcement learning
- Self learning
- Feature learning
- Sparse dictionary learning
- Anomaly detection
- Robot learning
- Association rules

Conclusion

Sequential Design/Active Learning

- Fit model: MVN Equations
- Choose next design point: Acquisition Functions

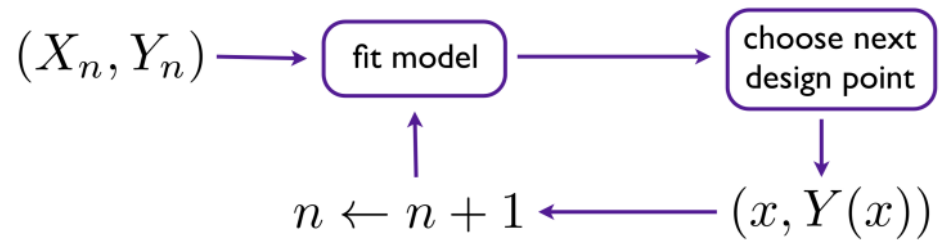
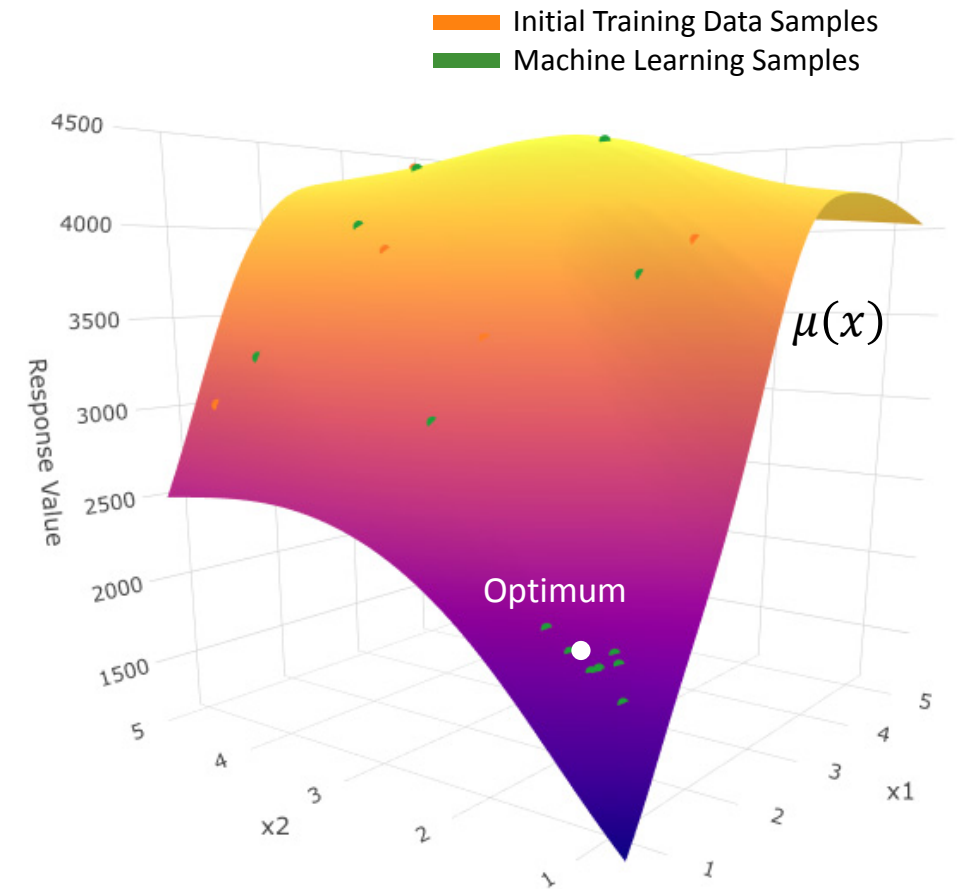


FIGURE 6.8: Diagram of sequential design/active learning/design augmentation.



FAQ

When do I use Machine Learning?

Gradient-Based Optimizer

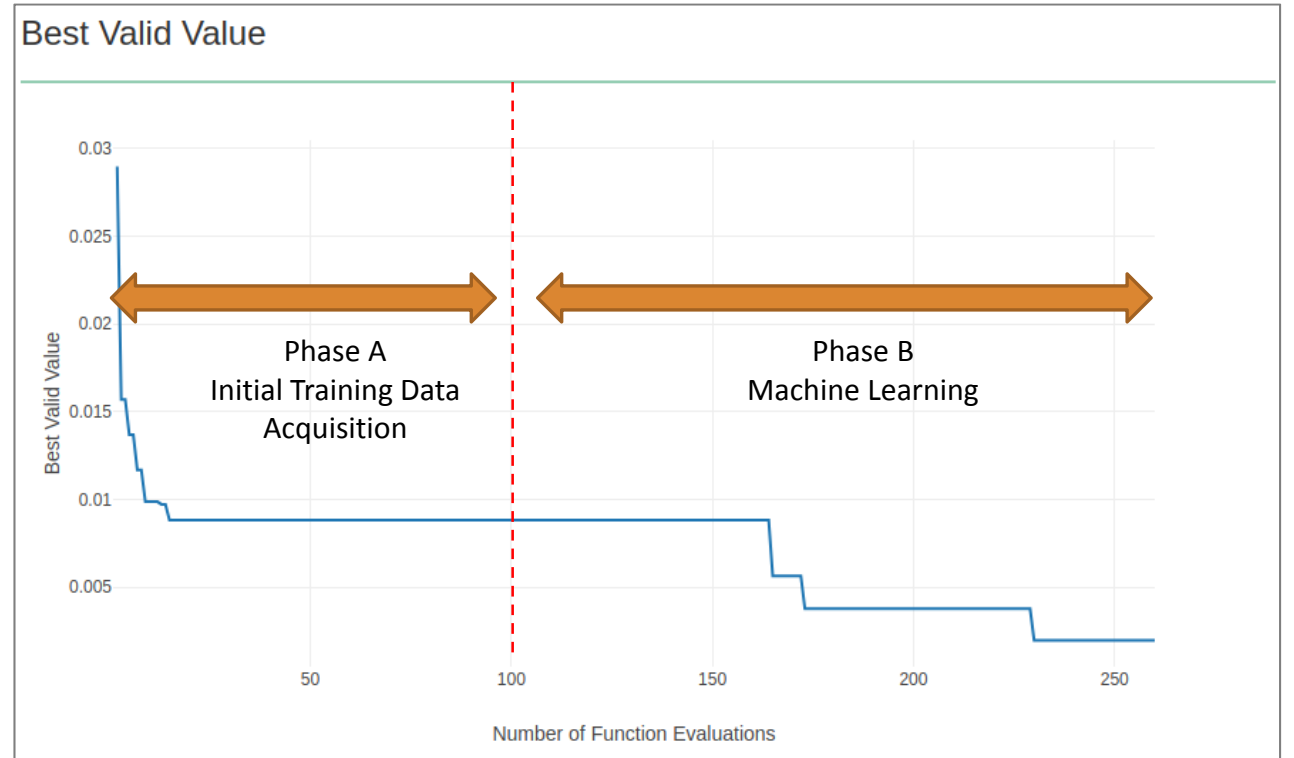
- SOL 200
 - Linear Responses
 - SOL 101, 103, 105, 107, 108, 110, 111, 112, 144 and 145
- Number of Variables
 - 1-1,000s

Machine Learning

- Bayesian Optimization
 - Little is known about the response function
 - Gradients are not available or expensive to compute
 - Nonlinear responses (SOL 400, 700)
 - Solver runs are time consuming
- Number of Parameters
 - 1-20 Parameters, Research
 - 1-16 Parameters, Tested

For the training data, how many runs per parameter?

5 runs per parameter (Tested for 1-16 parameters)

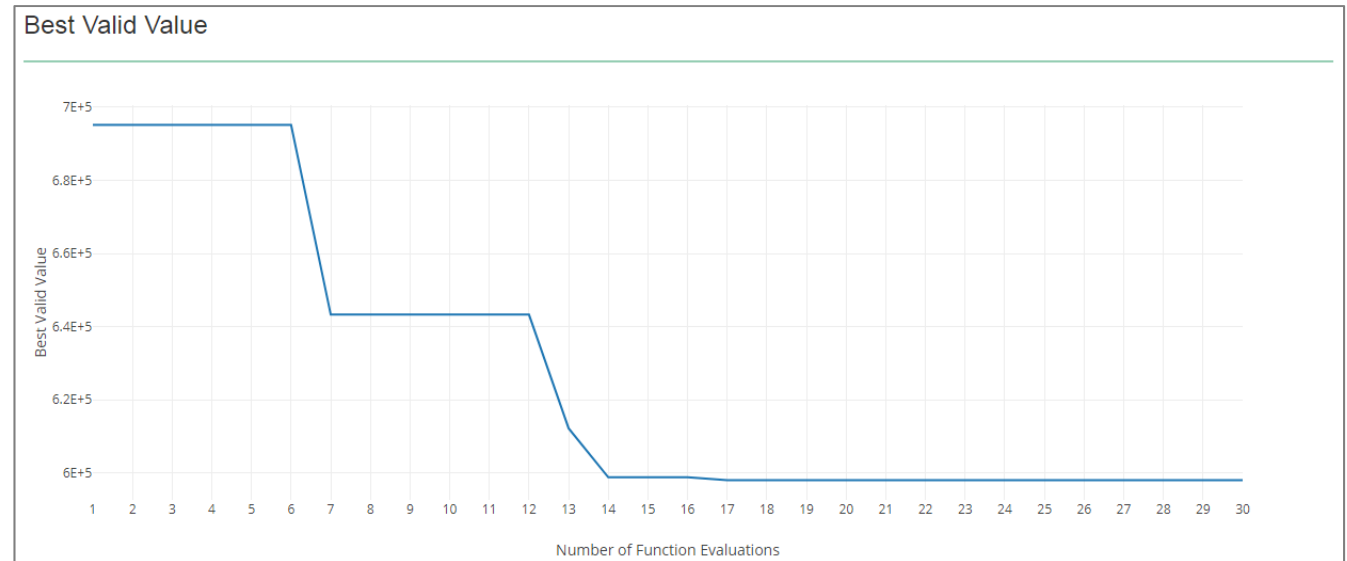


For higher dimensions, I do not reach the extrema (minimum/maximum). Why?

For higher dimensions, the design space is very large. Numerous function runs are required to search the large design space.

- Recommendations:
 1. Run machine learning with both EI and PI acquisition functions
 2. Perform a secondary machine learning where the design space is reduced to the neighborhood of the previous best optimum
 3. Run a gradient-based optimizer

“The optimal design is the last best feasible design”



Common Terminology

Common Terminology

Black Box Function: This is the FEA solver, such as MSC Nastran, CFD solver, or custom analysis code that generates outputs/responses.

Prediction Model: This model or function is used to predict the output of black box functions. Prediction model has many names in the machine learning world including emulator, surrogate model and meta model. In this work, the prediction model will correspond to the mean function $\mu(x)$.

Training Point: This is the x input and corresponding y output after running the black box function one time.

Training Data: This is all the x inputs and collected y outputs that will be used in the multivariate normal (MVN) conditioning equations that yields the mean function (prediction model) and variance function. Said another way, the training data is the x and y values that are used to train, or construct, the prediction model.

Testing Data: This is all the x inputs and collected y outputs (true y outputs). Predictions are made at these x inputs and should come close to matching the true y outputs.

Gaussian Process Model: Below is one form of the Gaussian Process (GP) model and is read as follows, “The output, $Y(x)$, given some observed outputs, D_n , is a multivariate normal distribution (\mathcal{N}) with mean function $\mu(x)$ and a variance function $\sigma^2(x)$.” The GP model is needed for stochastic outputs. Since we are interested only in deterministic outputs generated by FEA, MSC Nastran or CFD, we ignore \mathcal{N} and only need to determine the mean function and variance functions, or $\mu(x)$ and $\sigma^2(x)$, respectively.

$$Y(x) \mid D_n \sim \mathcal{N}(\mu(x), \sigma^2(x))$$

Gaussian Process Regression: This refers to the procedure used to obtain a Gaussian Process model. The procedure includes: kernel function selection, training data acquisition, kriging or applying the multivariate normal conditioning (MVN) expressions, which output the mean and variance functions. The mean function and variance functions are used in a final multivariate normal distribution (\mathcal{N}).

Thank You

christian @the-engineering-lab.com