Lecture 10:
Modeling Data Surrogate Based Modeling, Hybrid Piecewise Polynomial Modeling, and Random Forest

COSC 526: Introduction to Data Mining
Spring 2020

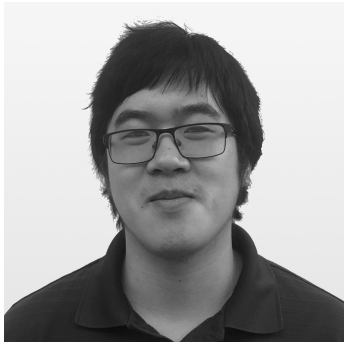
THE UNIVERSITY OF TENNESSEE KNOXVILLE

BIG ORANGE. BIG IDEAS.®

Instructor:

Michela Taufer

GRA:



Nigel Tan

Experts:

Leobardo Valera

Mike Wyatt

Assignment 10

Assignment 10

- Last assignment, we performed some early stages of data downscaling using a common and powerful Python package for data analysis: Pandas.
- This week, we execute a fully developed workflow to reproduce scientific experiments using powerful machine learning methods (i.e., KNN, Surrogate-based Model, HYPPO, and Random Forest).

Setup of the SOMOSPIE workflow

This process consists of two major steps:

- creating a new virtual machine in Jetstream from a custom VM image
- cloning the SOMOSPIE GitHub repository on your new VM

Setup of the SOMOSPIE workflow

This process consists of two major steps:

- creating a new virtual machine in Jetstream from a custom VM image
- cloning the SOMOSPIE GitHub repository on your new VM

Setup of the SOMOSPIE workflow

Your assignment is to:

- Setup and run the full SOMOSPIE workflow
- Use SOMOSPIE to partially reproduce a published study on one region with multiple machine learning methods
- Use SOMOSPIE to downscale soil moisture using one method across multiple regions
- Use Pandas to analyze the reported accuracies ($R^2$and RMSE) of the predictions across multiple regions and months

You will report the results of your experiments within this Notebook.

Do not push a copy of the SOMOSPIE repo in your class repo!

SOMOSPIE Setup

Step 1: Create a new VM

- In Jetstream (https://use.jetstream-cloud.org), launch an m1.medium Virtual Machine (VM) using the image titled "Ubuntu 18.04 for SOMOSPIE" (version 1.1). Consult the JetstreamGuide.ipynb for instructions on launching a VM.

SOMOSPIE Setup

Step 2: Clone the repo

- **Inside your new virtual machine**, clone the SOMOSPIE repository:

```
git clone –recursive https://github.com/TauferLab/SOMOSPIE
```

SOMOSPIE Setup

Step 3: Set up environment

- Before opening the SOMOSPIE notebook, we need to update the .bashrc:

```
cd SOMOSPIE
make bash
source ~/.bashrc
```

SOMOSPIE Setup

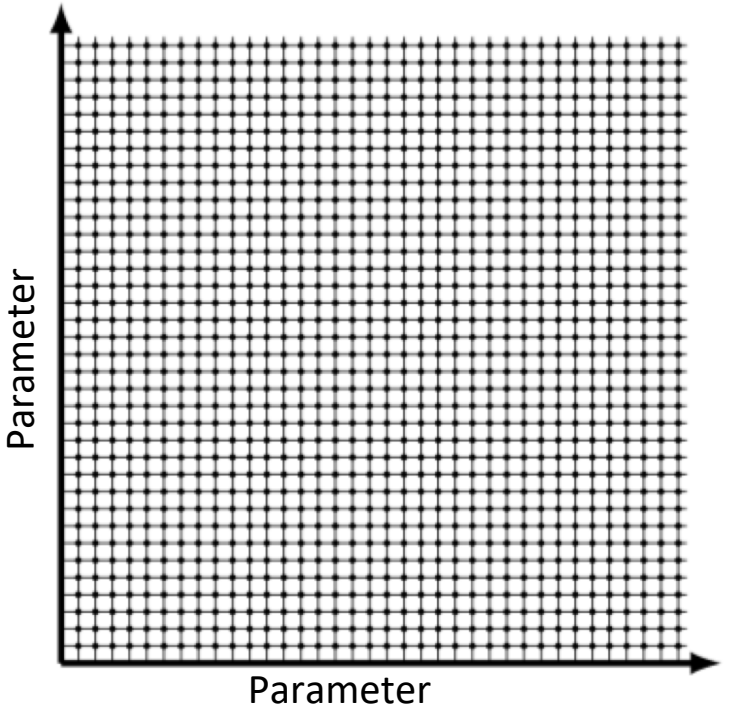
Step 4: Load data and run simple test

- Now open the notebook SOMOSPIE.ipynb and in the *Cell* menu select *Run All*

IMPORTANT: The first time you run it may take 45 minutes. The reason is because you are loading the data from public datasets.

Exhaustive Search

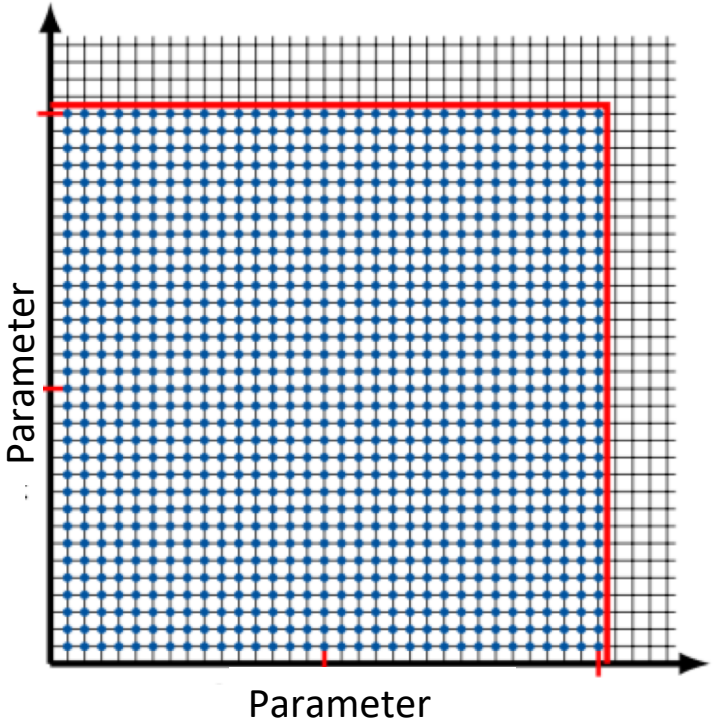
Exhaustive Search



Step 1: Reduce to finite search space

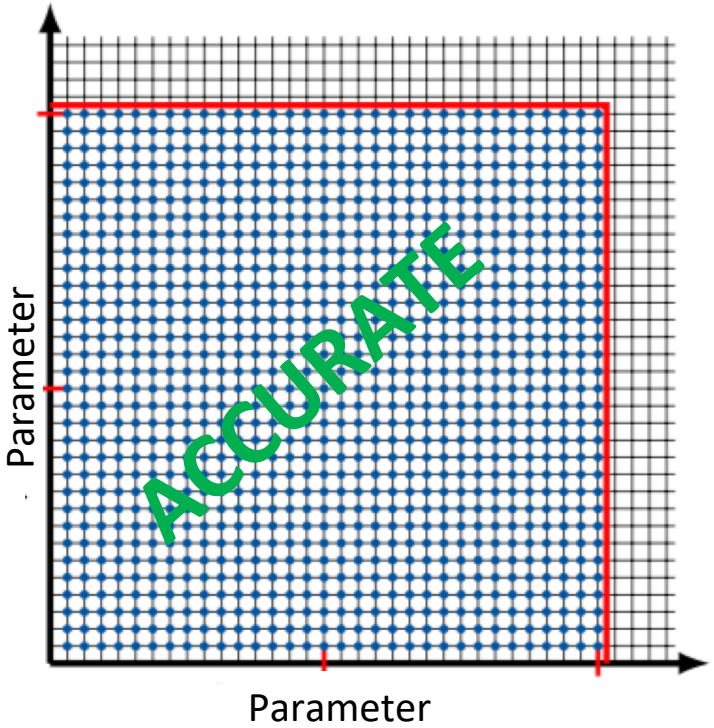
P. K. Lakkimsetti, A Framework for Automatic Optimization of MapReduce Programs Based on Job Parameter Configurations, M. Thesis, Kansas State University, 2011.

Exhaustive Search



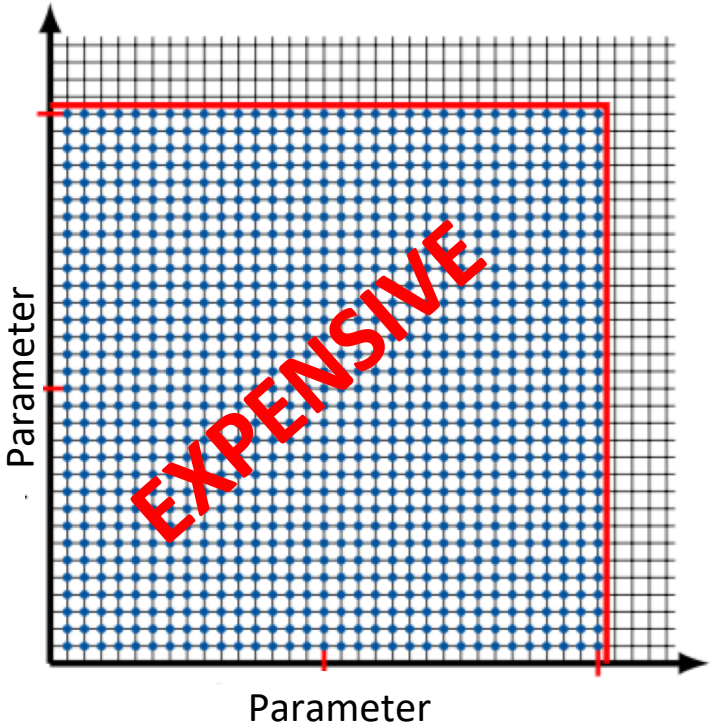
Step 2: Sample all these points

Exhaustive Search



Step 2: Sample all these points

Exhaustive Search

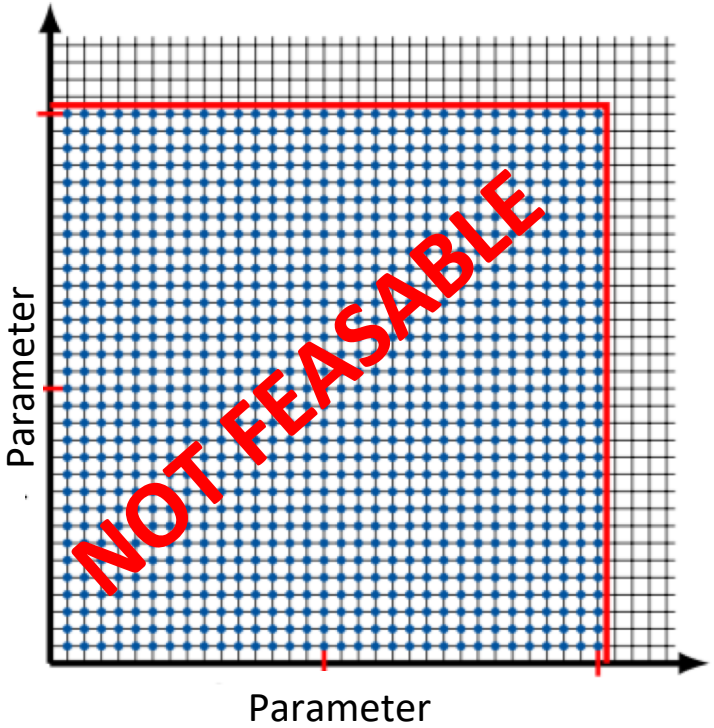


Step 2: Sample all these points **EXPENSIVE**

- Sample the entire population of the United States to leaner what each individual eats

P. K. Lakkimsetti, A Framework for Automatic Optimization of MapReduce Programs Based on Job Parameter Configurations, M. Thesis, Kansas State University, 2011.

Exhaustive Search



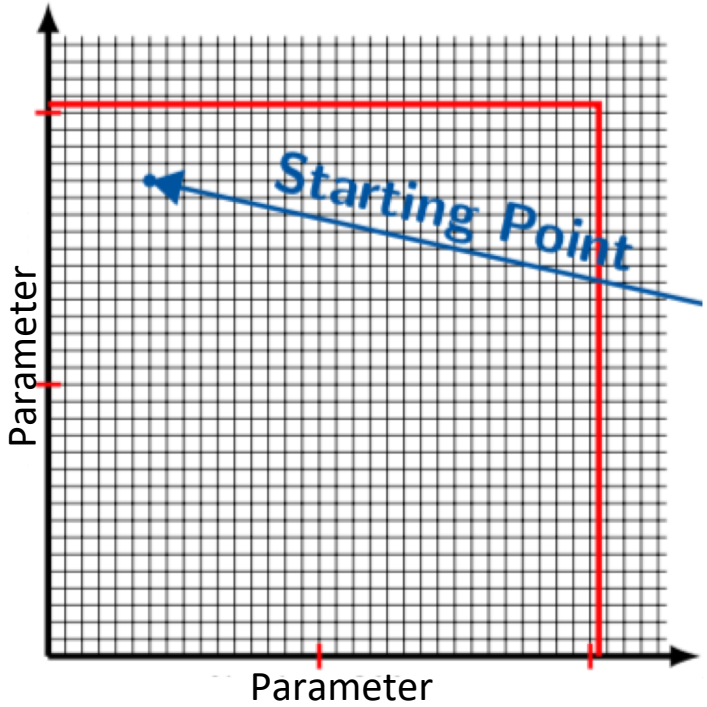
Step 2: Sample all these points **NOT FEASABLE**

- Satellites do not collect all the points across the globe

P. K. Lakkimsetti, A Framework for Automatic Optimization of MapReduce Programs Based on Job Parameter Configurations, M. Thesis, Kansas State University, 2011.

Local Search

Local Searching Algorithms



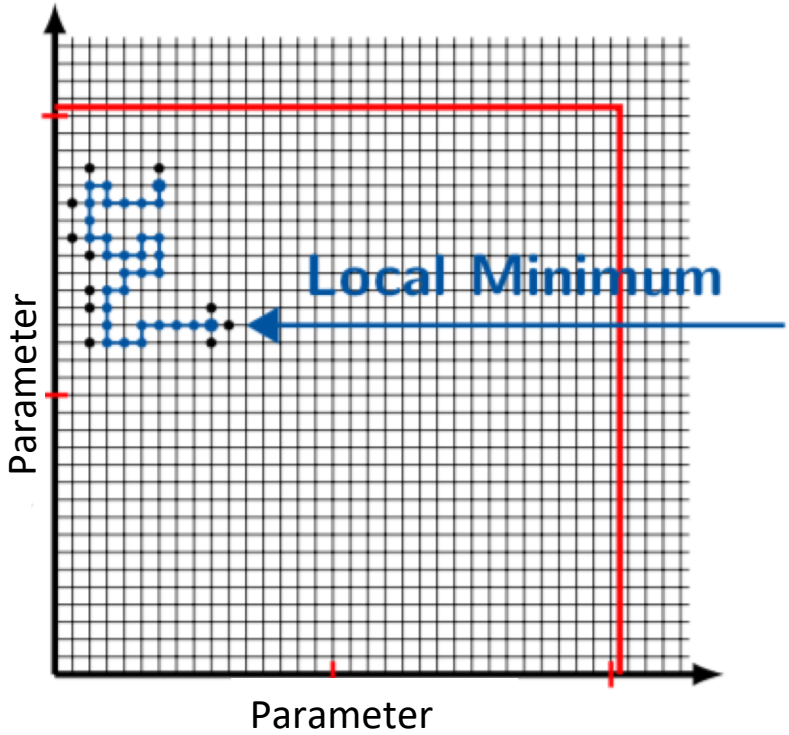
Methods: Grid Hill [1] and Simulated Annealing [2]

Step 1: Reduce to finite search space

Step 2: Randomly choose starting point and sample neighbors

[1] K. Wang, X. Lin, W. Tang, Predator-An Experience Guided Configuration Optimizer..., IEEE CloudCom 2012.
[2] D. Wu, A Profiling and Performance Analysis Based Self-tuning System for Optimization..., M. Thesis, Vanderbilt, 2013.

Local Searching Algorithms



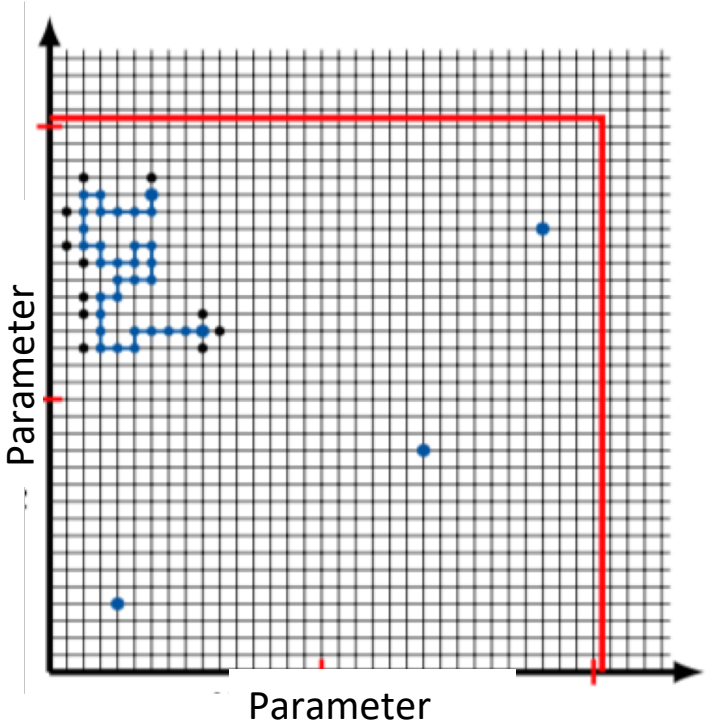
Methods: Grid Hill [1] and Simulated Annealing [2]

Step 1: Reduce to finite search space

Step 2: Randomly choose starting point and sample neighbors

[1] K. Wang, X. Lin, W. Tang, Predator-An Experience Guided Configuration Optimizer..., IEEE CloudCom 2012.
[2] D. Wu, A Profiling and Performance Analysis Based Self-tuning System for Optimization..., M. Thesis, Vanderbilt, 2013.

Local Searching Algorithms

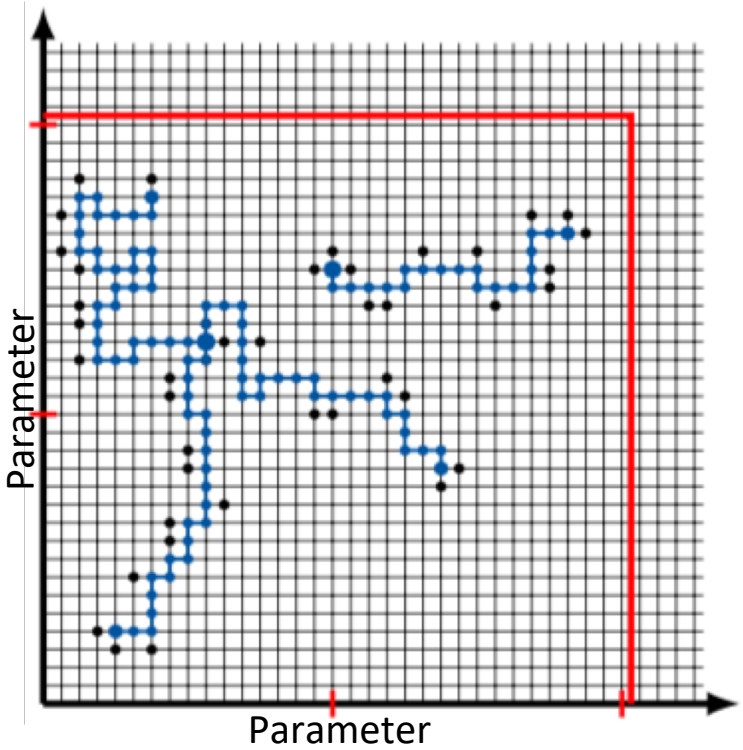


Methods: Grid Hill [1] and Simulated Annealing [2]

Step 3: Repeat Step 2 with new starting point(s)

[1] K. Wang, X. Lin, W. Tang, Predator-An Experience Guided Configuration Optimizer..., IEEE CloudCom 2012.
[2] D. Wu, A Profiling and Performance Analysis Based Self-tuning System for Optimization..., M. Thesis, Vanderbilt, 2013.

Local Searching Algorithms

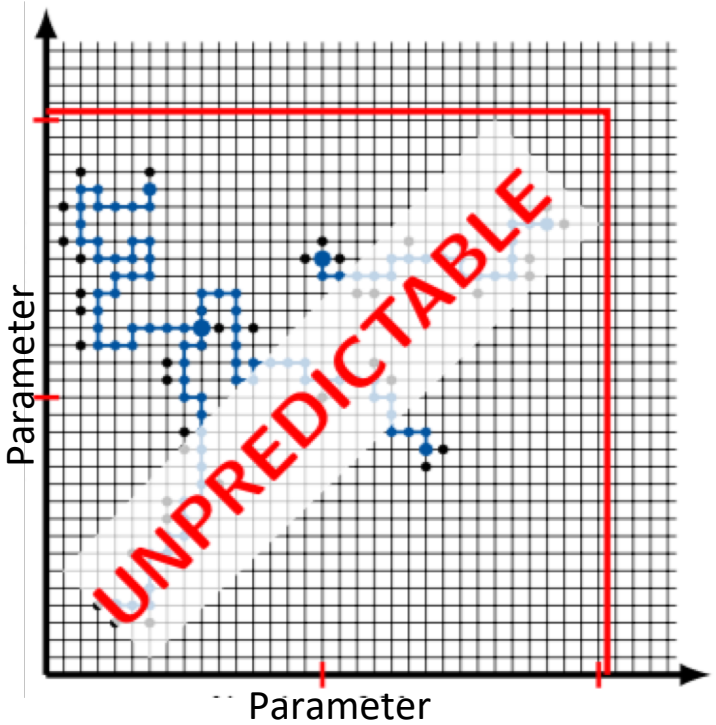


Methods: Grid Hill [1] and Simulated Annealing [2]

Step 3: Repeat Step 2 with new starting point(s)

[1] K. Wang, X. Lin, W. Tang, Predator-An Experience Guided Configuration Optimizer..., IEEE CloudCom 2012.
[2] D. Wu, A Profiling and Performance Analysis Based Self-tuning System for Optimization..., M. Thesis, Vanderbilt, 2013.

Local Searching Algorithms



Methods: Grid Hill [1] and Simulated Annealing [2]

Step 3: Repeat Step 2 with new starting point(s)

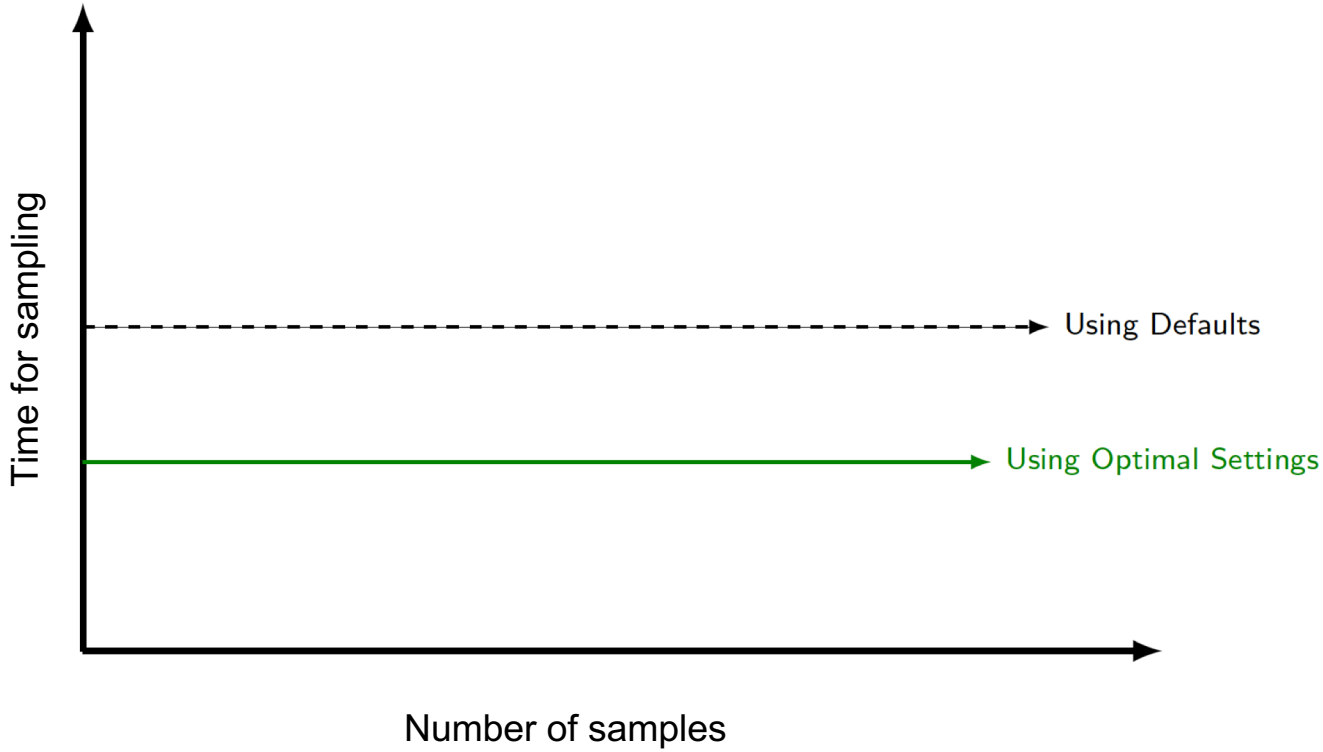
[1] K. Wang, X. Lin, W. Tang, Predator-An Experience Guided Configuration Optimizer..., IEEE CloudCom 2012.
[2] D. Wu, A Profiling and Performance Analysis Based Self-tuning System for Optimization..., M. Thesis, Vanderbilt, 2013.

Exhaustive and Local Searches

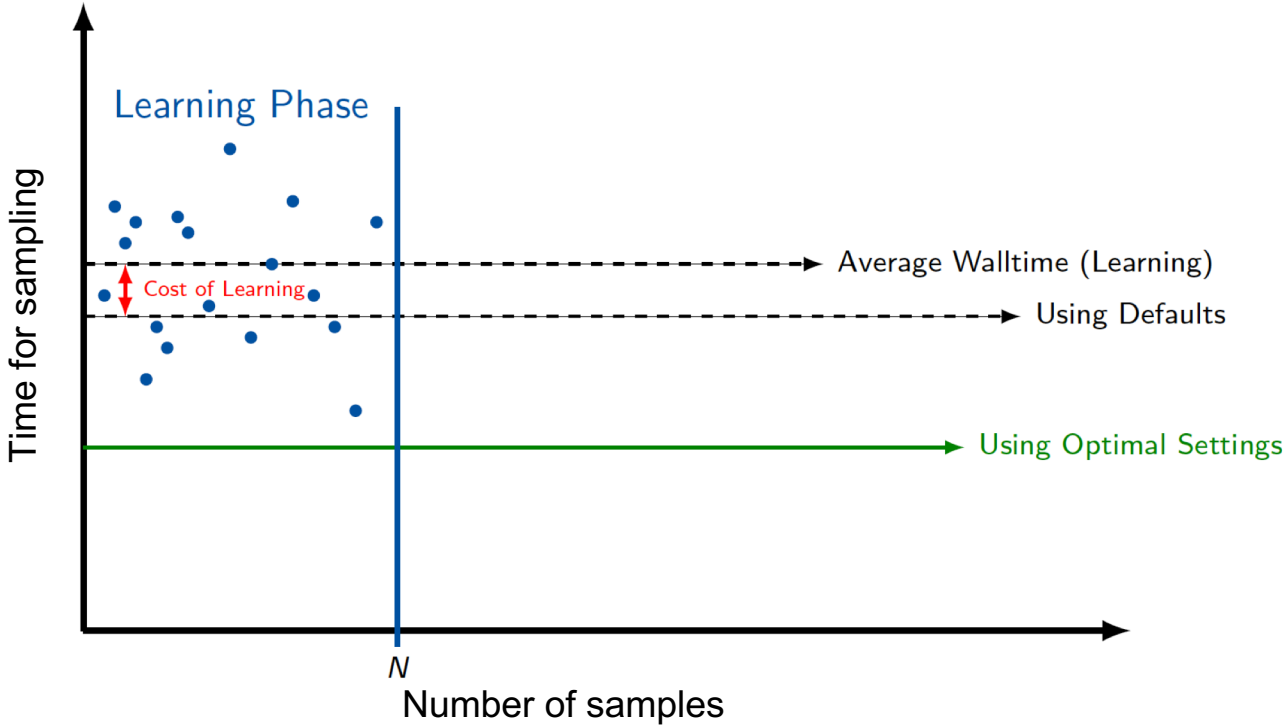
- Exhaustive sampling is too expensive
- Local search algorithms (LSAs) sample fewer points but are unpredictable

Cost of Sampling

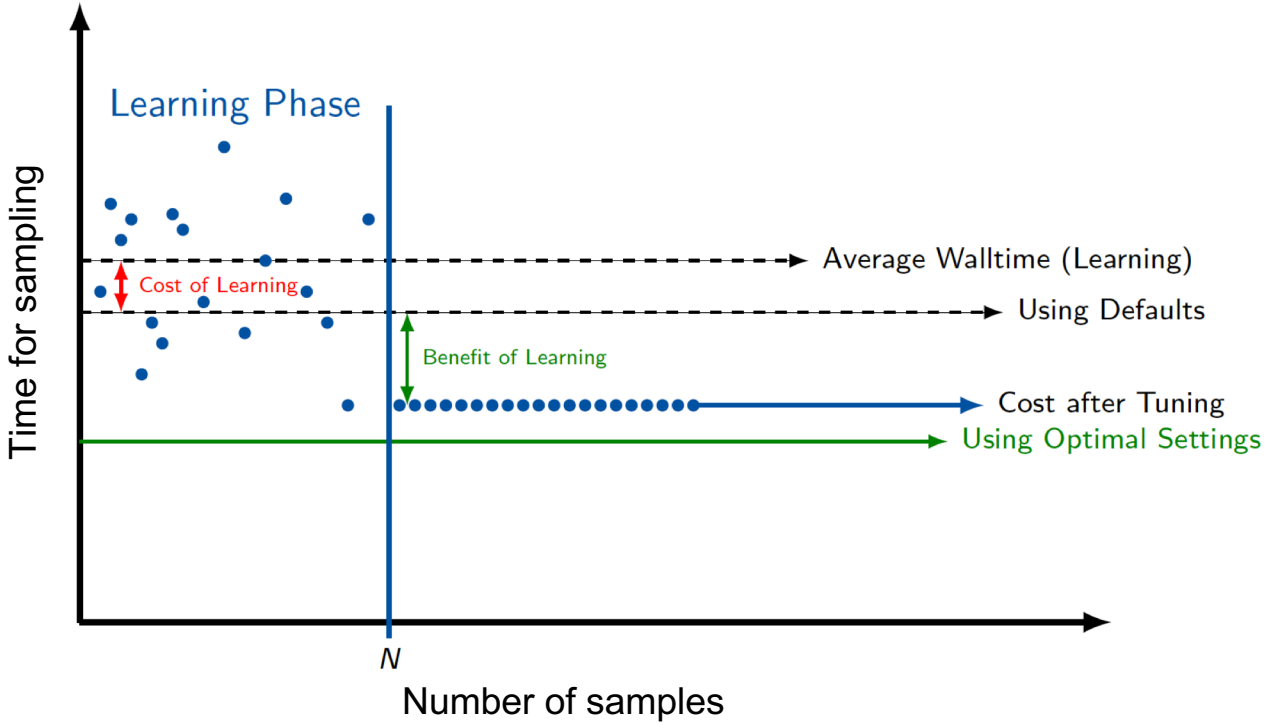
Amortizing the Cost of Sampling



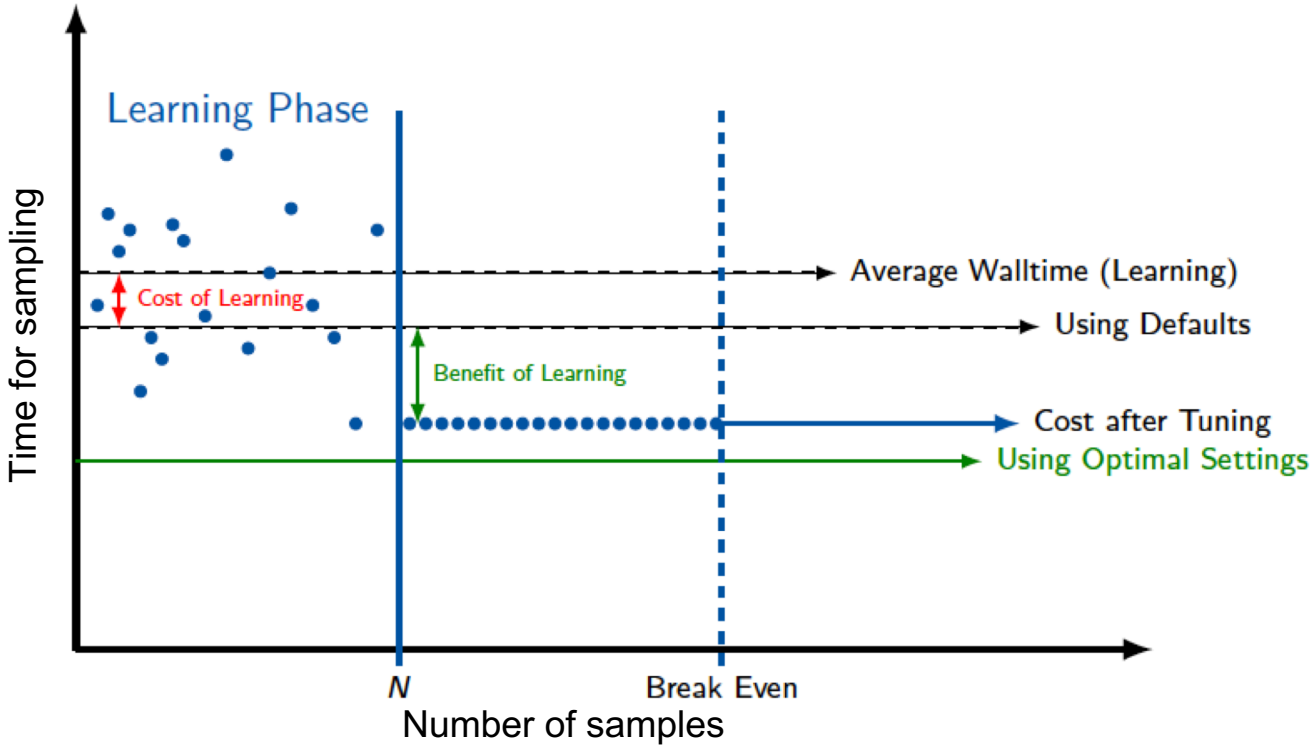
Amortizing the Cost of Sampling



Amortizing the Cost of Sampling



Amortizing the Cost of Sampling

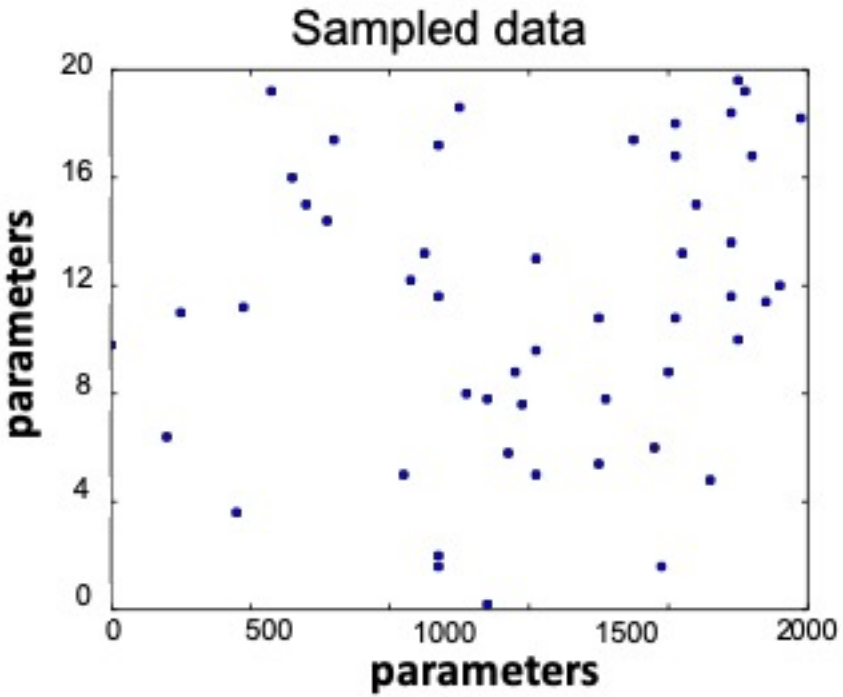


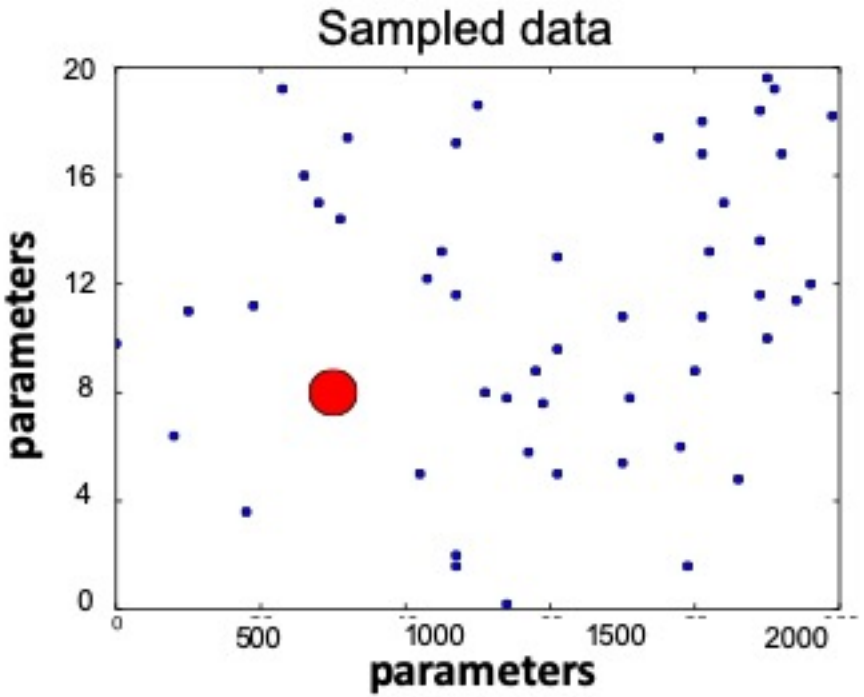
Amortizing the Cost of Sampling

- How do we focus the learning phase to maximize the return on our investment?

k Nearest Neighbors or kNN

k Nearest Neighbors Model

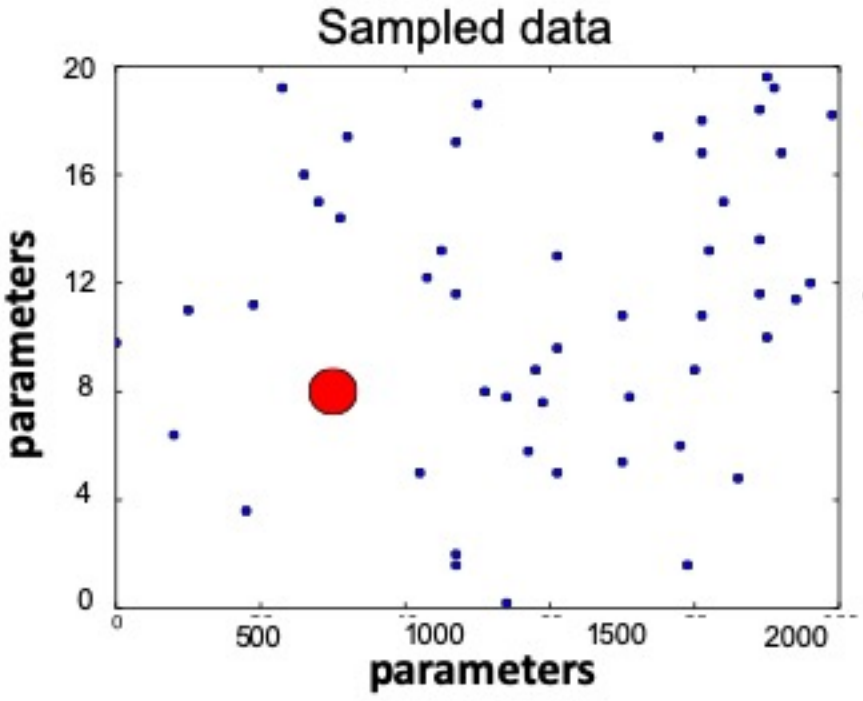


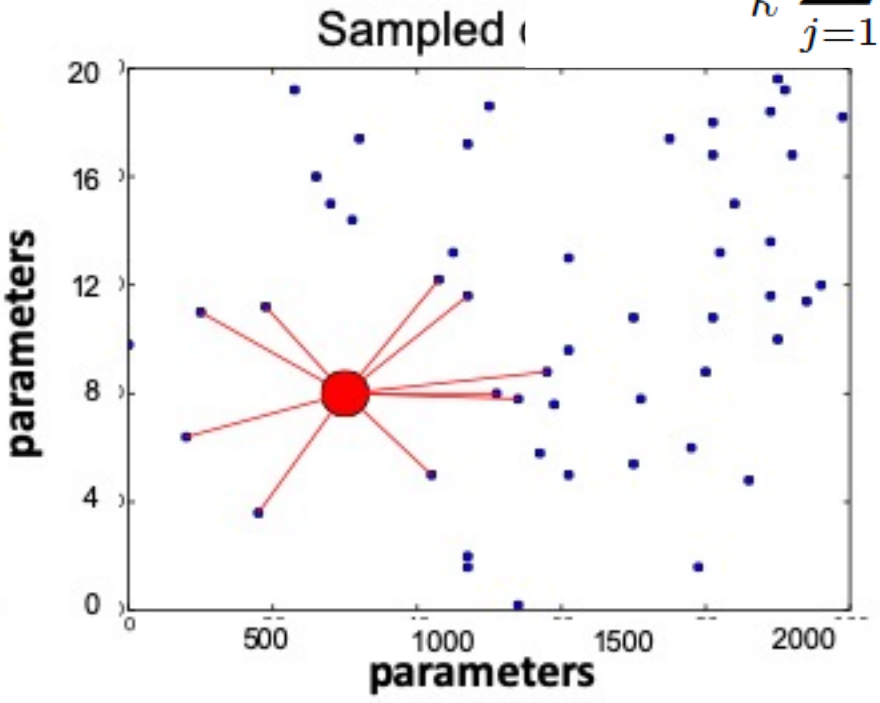


k Nearest Neighbors Model

Polynomial with degree zero

$$z(\vec{x}) := \frac{1}{k}\sum_{j=1}^{k} z_{i_j}$$





kNN

KNN:

→ Use **local data**

→ Compute k and distance kernel using cross validation automatically

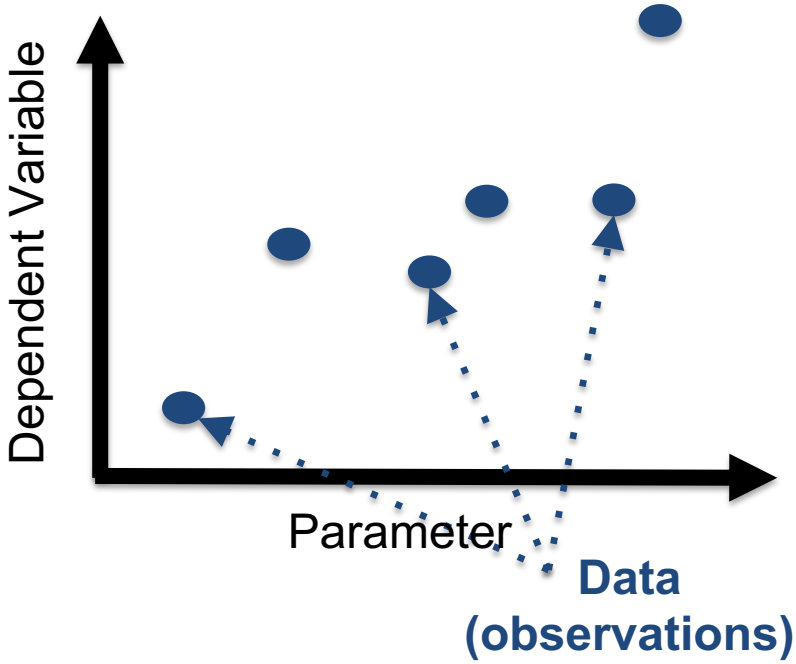
→ Compute weighted means with the kernel (***many values***)

Surrogate-Based Modeling

Using Surrogate-Based Modeling

- The surrogate model may predict optimal configurations that were never sampled in the learning phase
- We can explicitly determine the number of points required to build a surrogate model

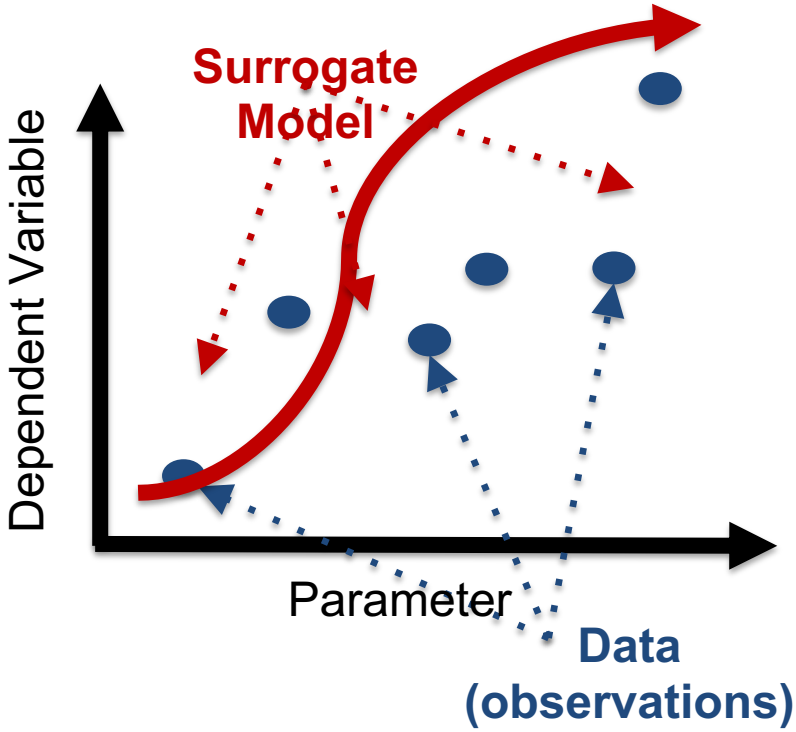
Surrogate-Based Modeling



- Data → **all** sampled data to create a **single global model**
- Model → fit a **polynomial** to data (continuous and differentiable)
- Best for→ finding underlying **global trends** when they exist

Image source: Johnston, R., Taufer

Surrogate-Based Modeling

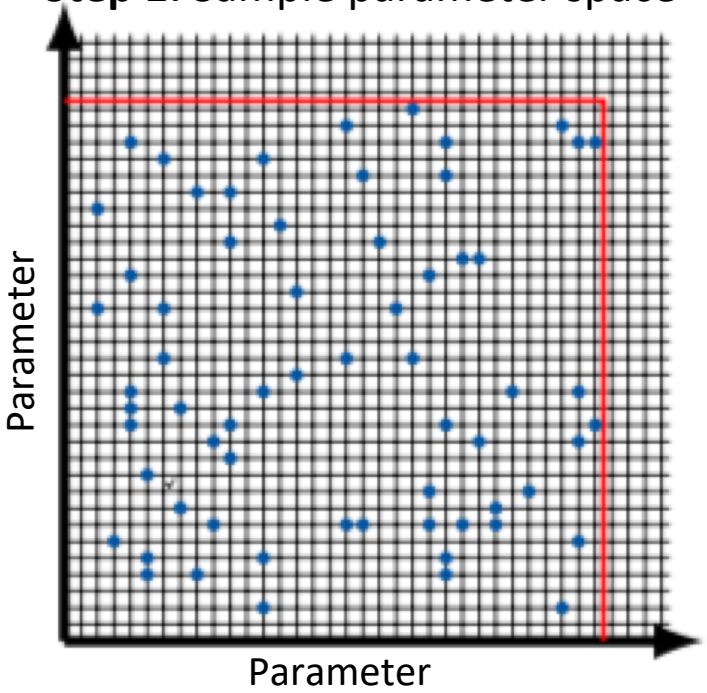


- Data → **all** sampled data to create a **single global model**
- Model → fit a **polynomial** to data (continuous and differentiable)
- Best for→ finding underlying **global trends** when they exist

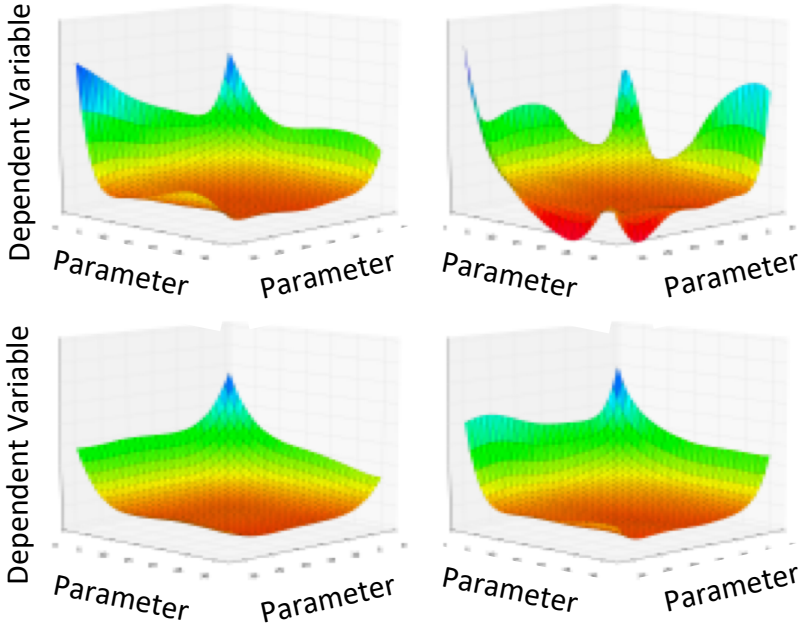
Image source: Johnston, R., Taufer

Surrogate-Based Modeling

Step 1: Sample parameter space

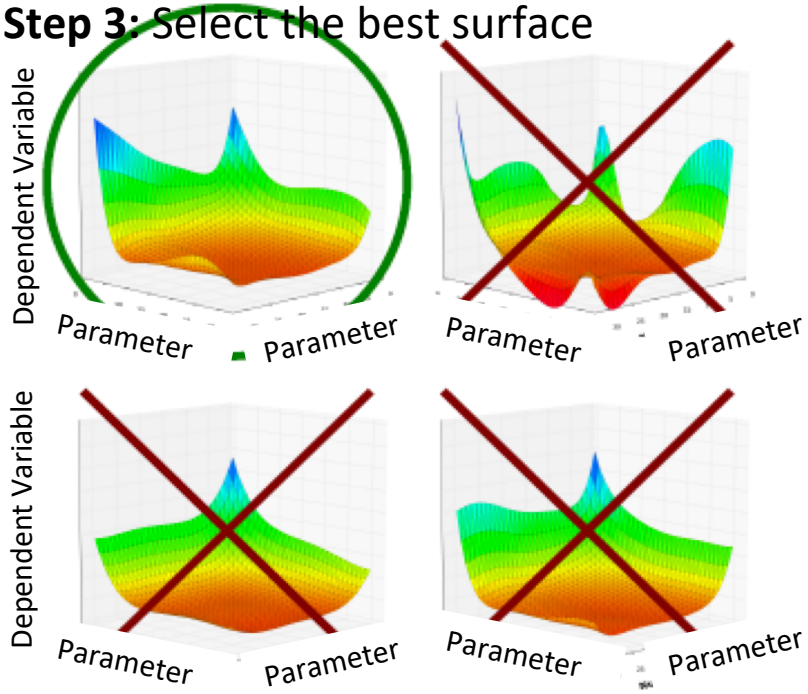


Step 2: Build candidate surfaces

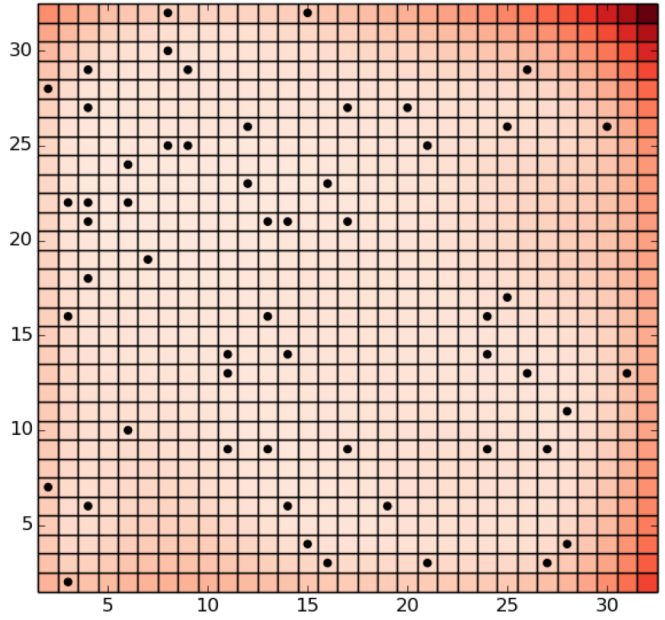


Surrogate-Based Modeling

Step 3: Select the best surface

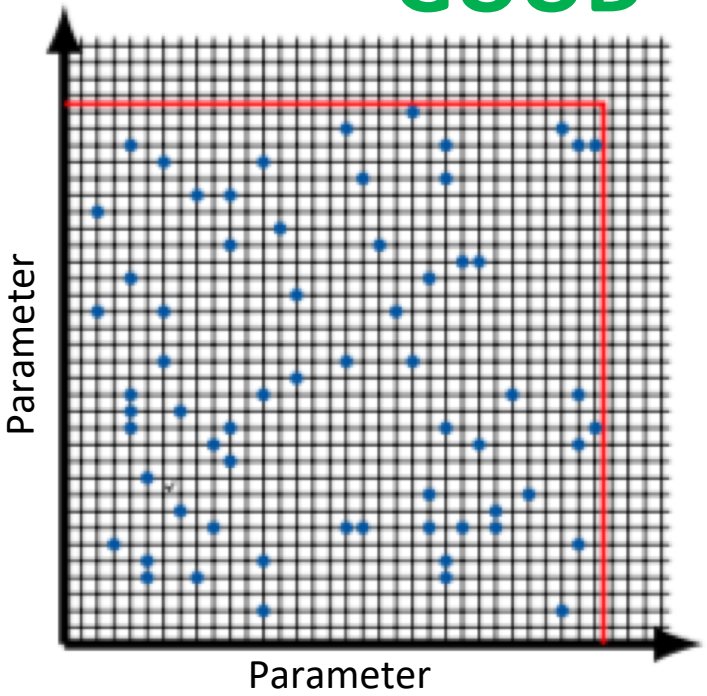


Step 4: Apply confidence interval

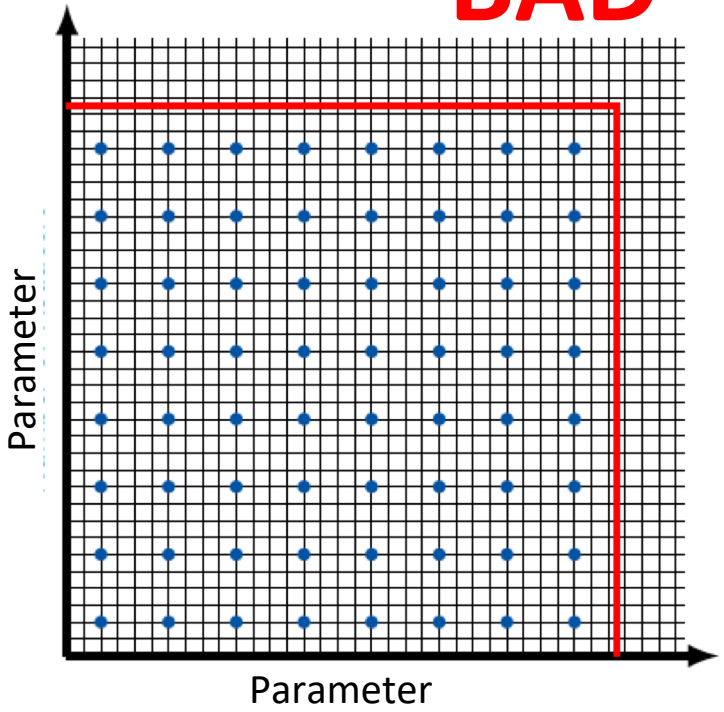


Step 1: Random sampling

GOOD



BAD



Step 2: Building candidate surfaces

What kind of surfaces do we build?

- We represent our **surface** by a multivariate **polynomials**
- The candidate surfaces look like: **Z = B X**

- $z_1(x, y) = \beta_1 + \beta_2 x + \beta_3 y$ Degree 1
- $z_2(x, y) = \beta_1 + \beta_2 x + \beta_3 y + \beta_4 x^2 + \beta_5 xy + \beta_6 y^2$ Degree 2
- $z_3(x, y) = z_2(x, y) + \beta_7 x^3 + \beta_8 x^2 y + \beta_9 xy^2 + \beta_{10} y^3$ Degree 3

Step 2: Building candidate surfaces

Why build a polynomial surface?

- Polynomials are easy to describe and represent in memory
- Polynomials can generate quite complex surfaces
- Polynomials easily generalize to any number of variables

Step 2: Building candidate surfaces

- We construct best fit polynomial surfaces of degree *d* for each reasonable value of d, i.e. d = {1, 2, …, M} where M is small enough that the matrix X^TX is invertible

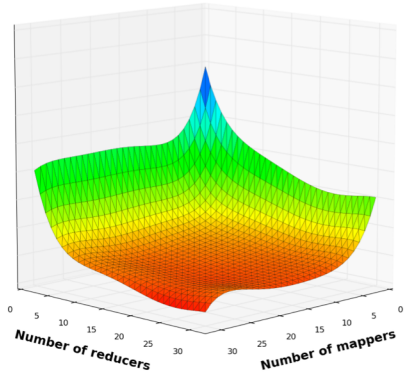
Step 2: Building candidate surfaces

How many points do we need to sample?

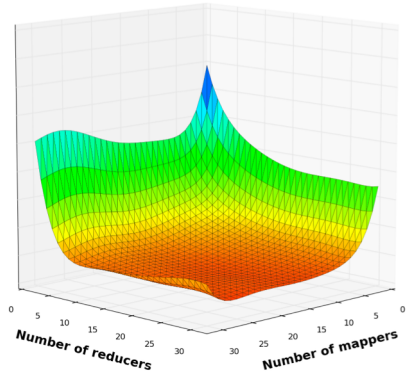
- To build the surface, we determine the coefficients B by solving the matrix equation:

  $X\,B = Z \rightarrow \mathbf{X^T X\,B = X^T Z}$

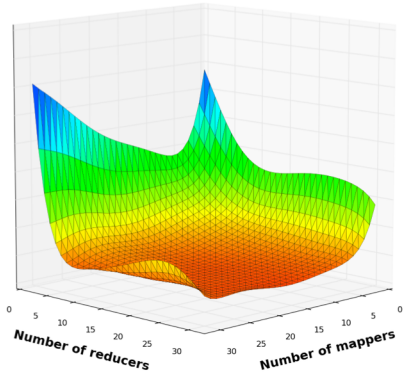
- If $\mathbf{X^T X}$ is not invertible, then there is not a unique solution for B
- If the number of samples taken is smaller than the number of terms in our polynomial, then $X^T X$ is not invertible
- To build a surface of degree d with *v* variables we need at least $\binom{d+v}{v}$



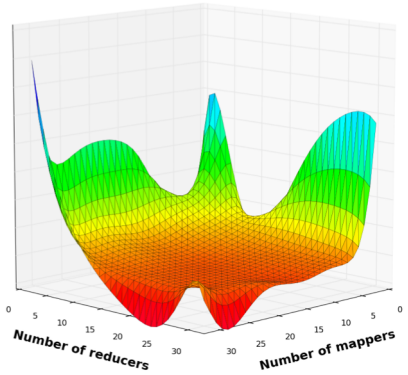
Degree 5



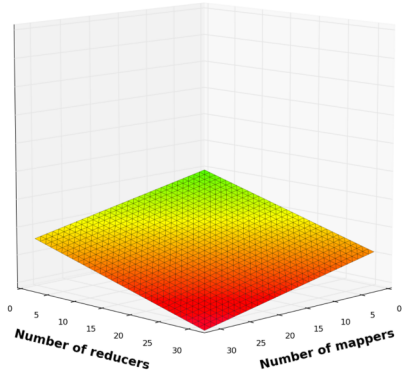
Degree 6



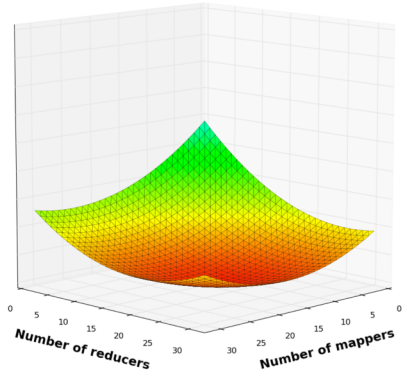
Degree 7



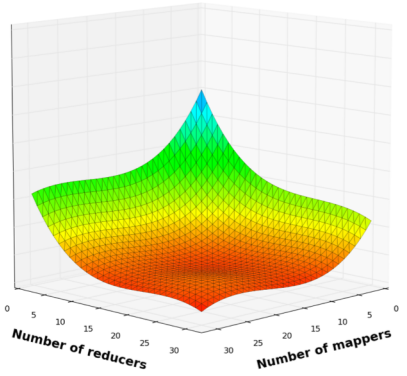
Degree 8



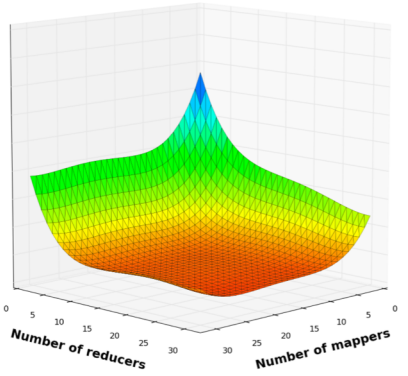
Degree 1



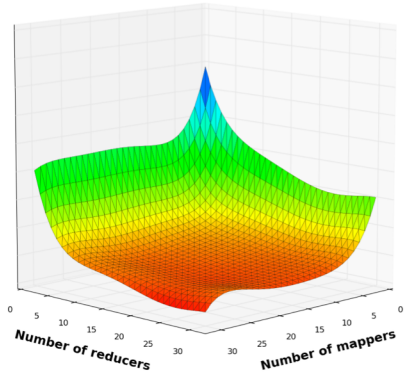
Degree 2



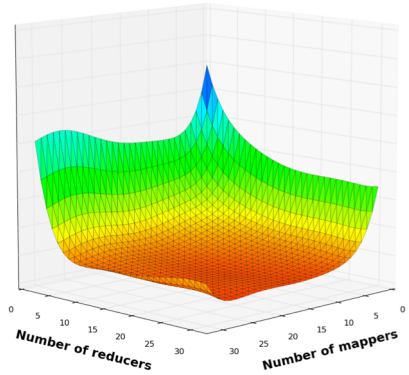
Degree 3



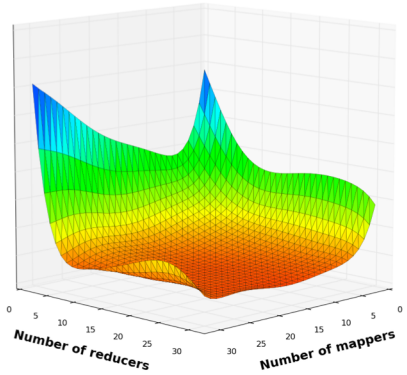
Degree 4



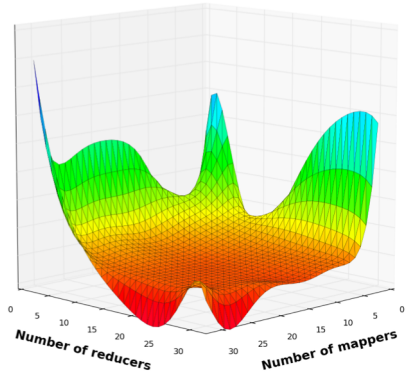
Degree 5



Degree 6

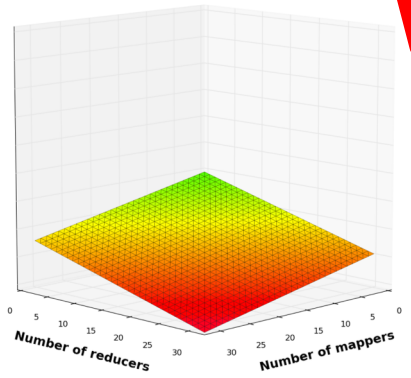


Degree 7

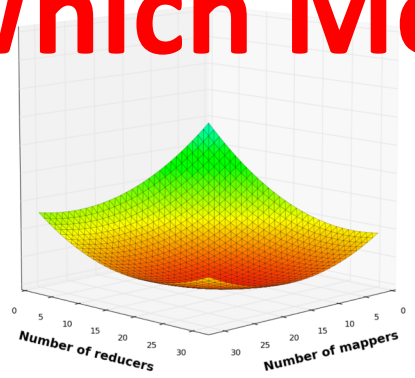


Degree 8

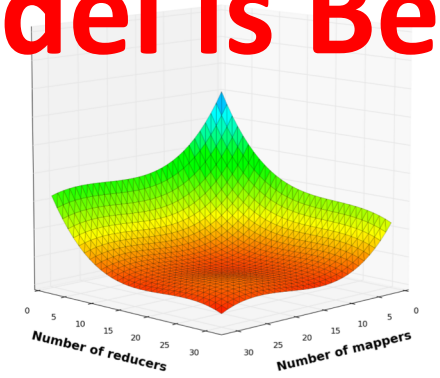
Which Model is Best?



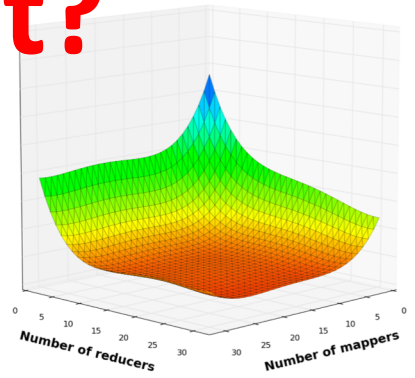
Degree 1



Degree 2



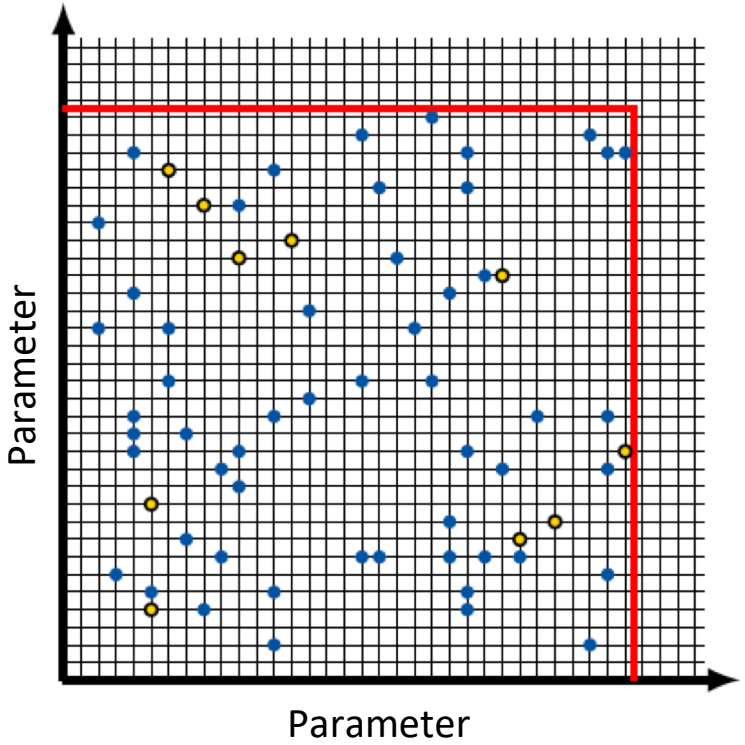
Degree 3



Degree 4

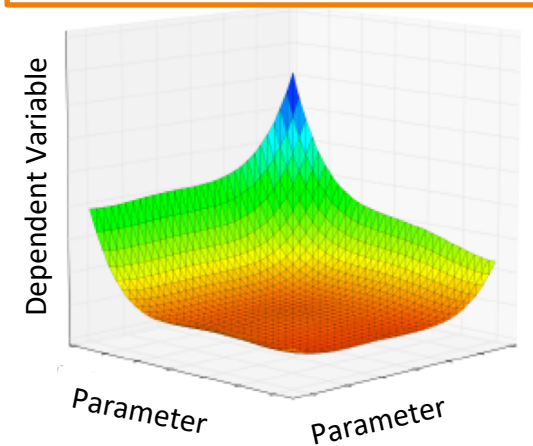
Step 3: Selecting the best surface using k-fold cross validation

- **Partition samples into k sets of equal size**
- **k -1 sets are used for learning; one is use for testing**
- Build best fit polynomial surface from learning set
- Use polynomial surface to predict unknown variables in testing set
- Use points in the testing set to compute SSE



Partition 1

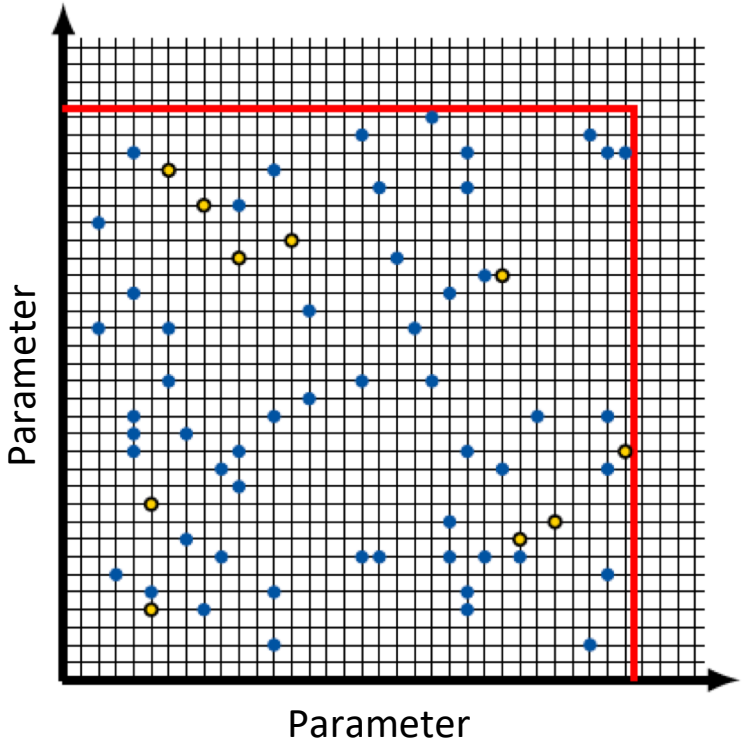
- **Learning Sets(k -1)**
- **Testing Set (1)**



Degree 4 Surface

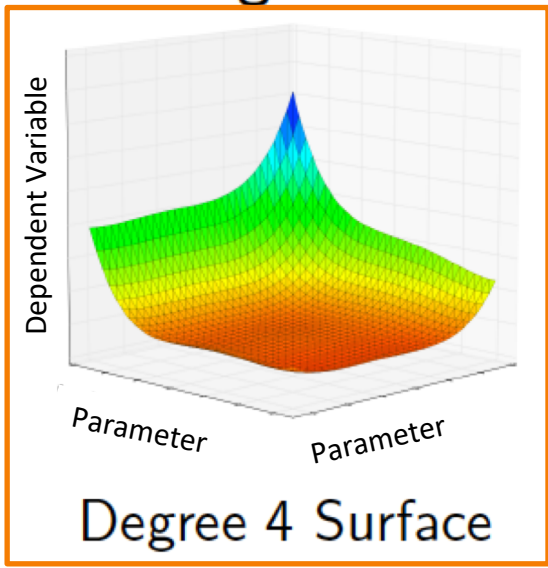
Step 3: Selecting the best surface using k-fold cross validation

- Partition samples into k sets of equal size
- k -1 sets are used for learning; one is use for testing
- **Build best fit polynomial surface from learning set**
- Use polynomial surface to predict unknown variables in testing set
- Use points in the testing set to compute SSE



Partition 1

- Learning Sets
- Testing Set



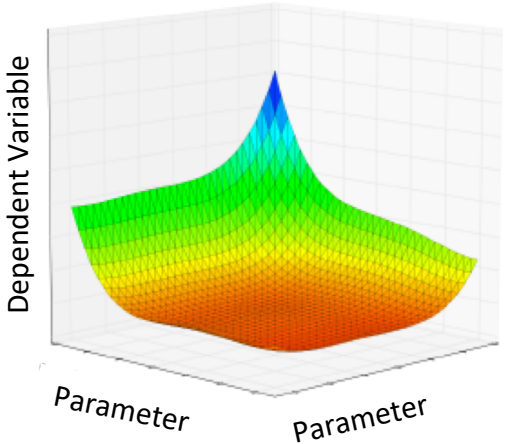
Step 3: Selecting the best surface using k-fold cross validation

- Partition samples into k sets of equal size
- k -1 sets are used for learning; one is use for testing
- Build best fit polynomial surface from learning set
- **Use polynomial surface to predict unknown variables in testing set**
- **Use points in the testing set to compute SSE**

Compute SSE:
Observed variables
in testing points
vs
predicted variable
(using surface)
in testing pints

Partition 1

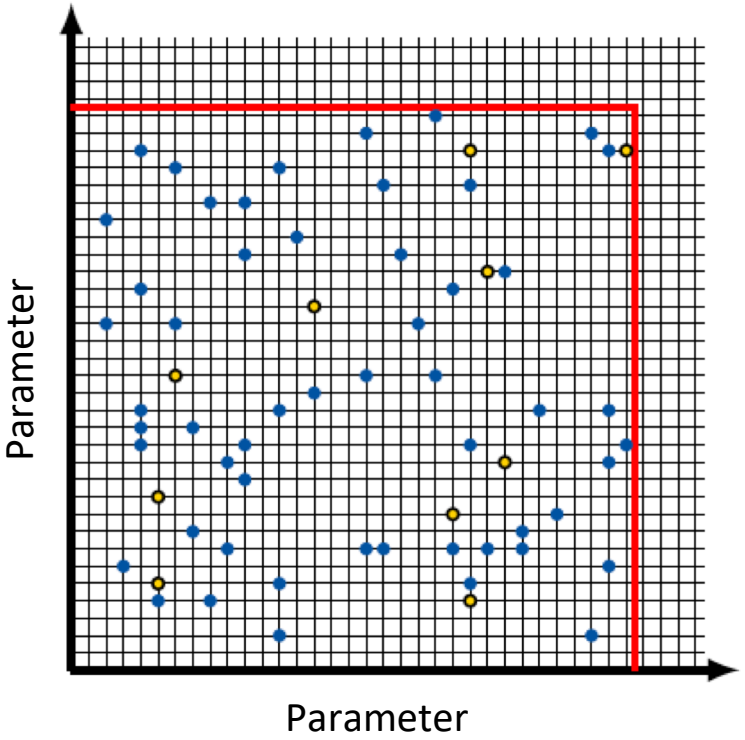
- **Learning Sets**
- **Testing Set**



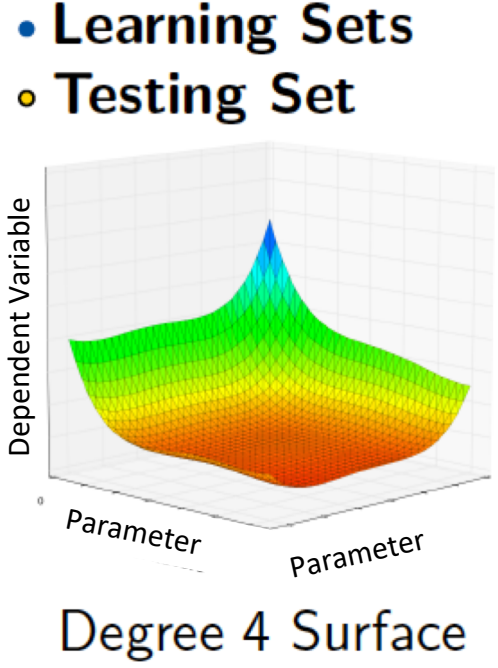
Degree 4 Surface

Step 3: Selecting the best surface using k-fold cross validation

- Partition samples into k sets of equal size
- k -1 sets are used for learning; one is use for testing
- Build best fit polynomial surface from learning set
- Use polynomial surface to predict unknown variables in testing set
- Use points in the testing set to compute SSE

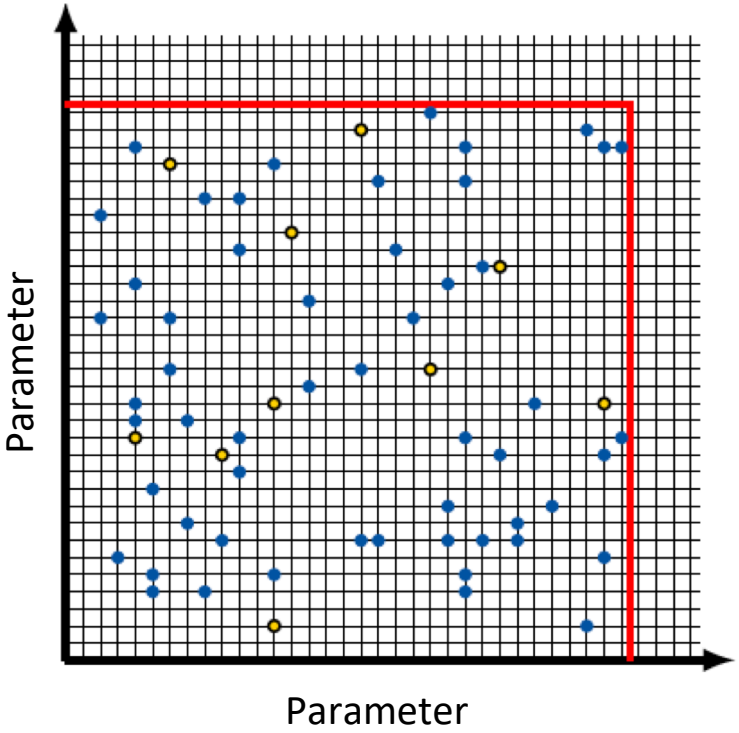


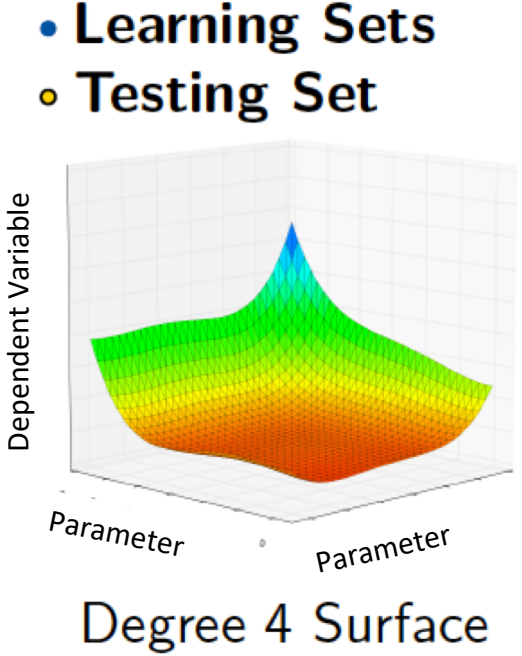
Partition 2



Step 3: Selecting the best surface using k-fold cross validation

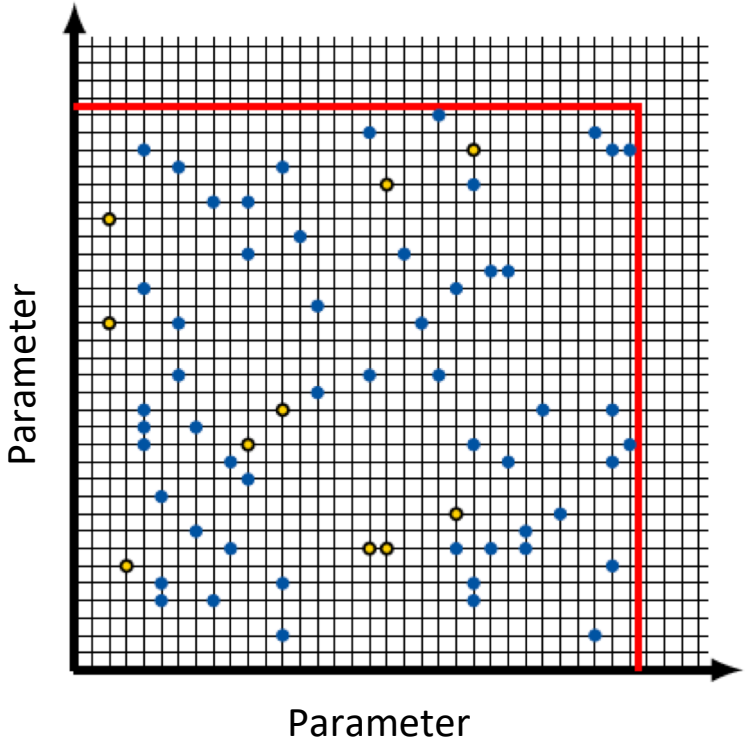
- Partition samples into k sets of equal size
- k -1 sets are used for learning; one is use for testing
- Build best fit polynomial surface from learning set
- Use polynomial surface to predict unknown variables in testing set
- Use points in the testing set to compute SSE



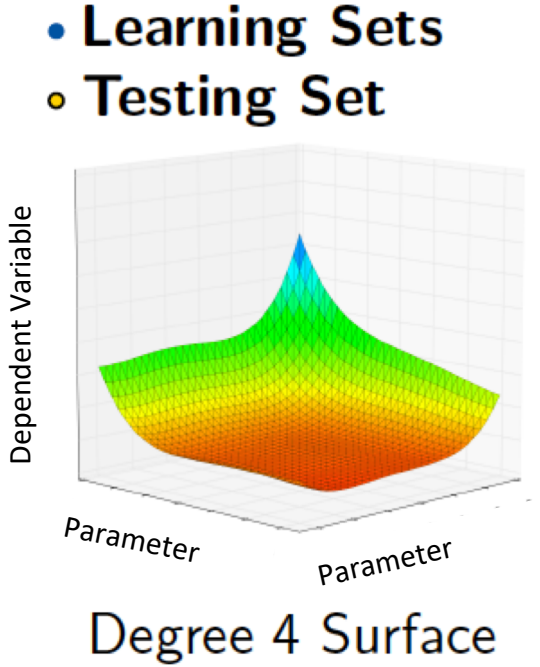


Step 3: Selecting the best surface using k-fold cross validation

- Partition samples into k sets of equal size
- k -1 sets are used for learning; one is use for testing
- Build best fit polynomial surface from learning set
- Use polynomial surface to predict unknown variables in testing set
- Use points in the testing set to compute SSE

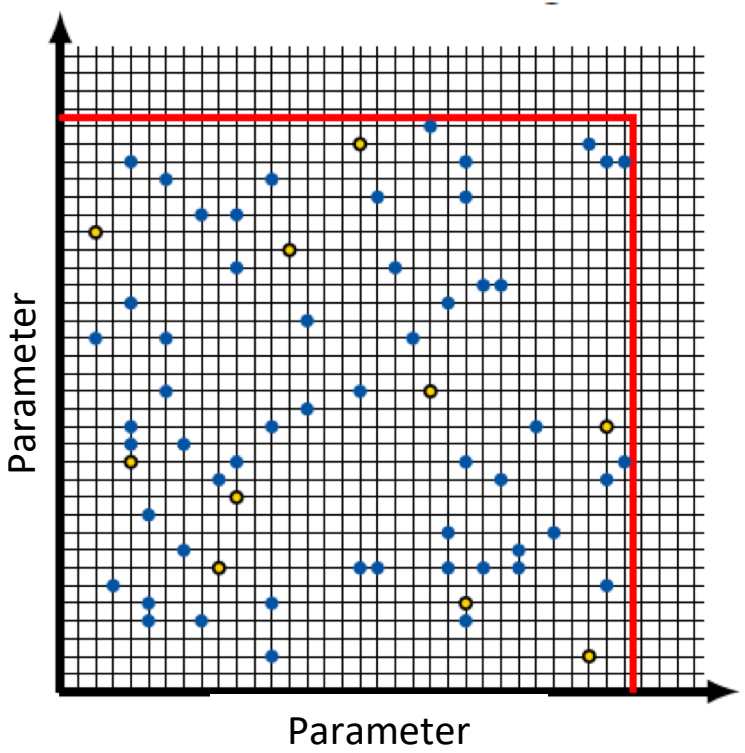


Partition 4

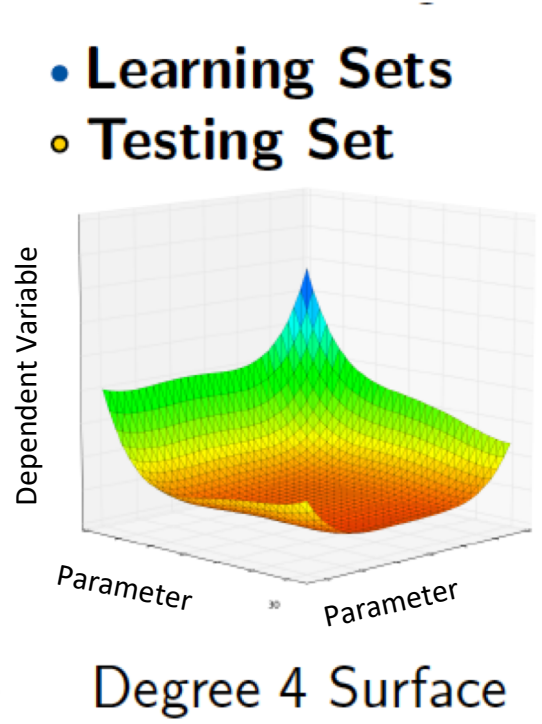


Step 3: Selecting the best surface using k-fold cross validation

- Partition samples into k sets of equal size
- k -1 sets are used for learning; one is use for testing
- Build best fit polynomial surface from learning set
- Use polynomial surface to predict unknown variables in testing set
- Use points in the testing set to compute SSE

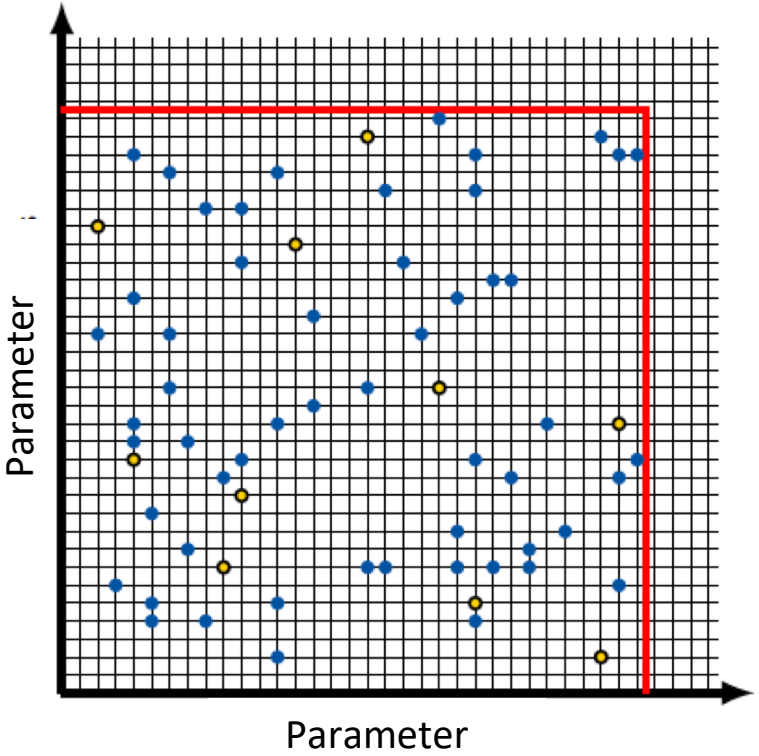


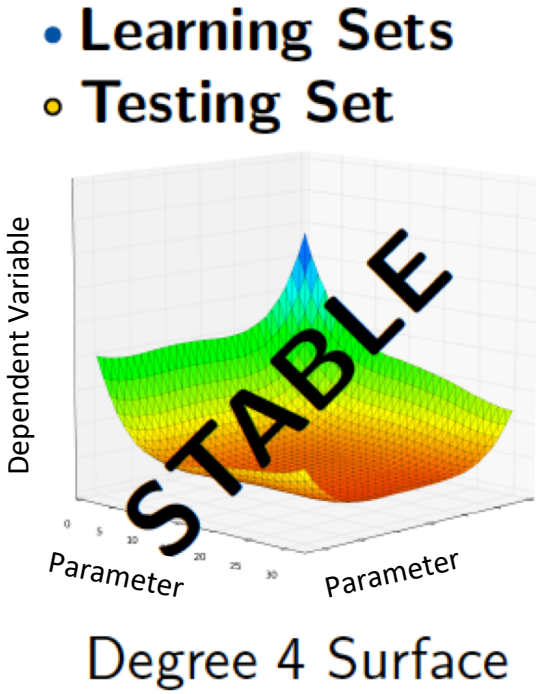
Partition 5



Step 3: Selecting the best surface using k-fold cross validation

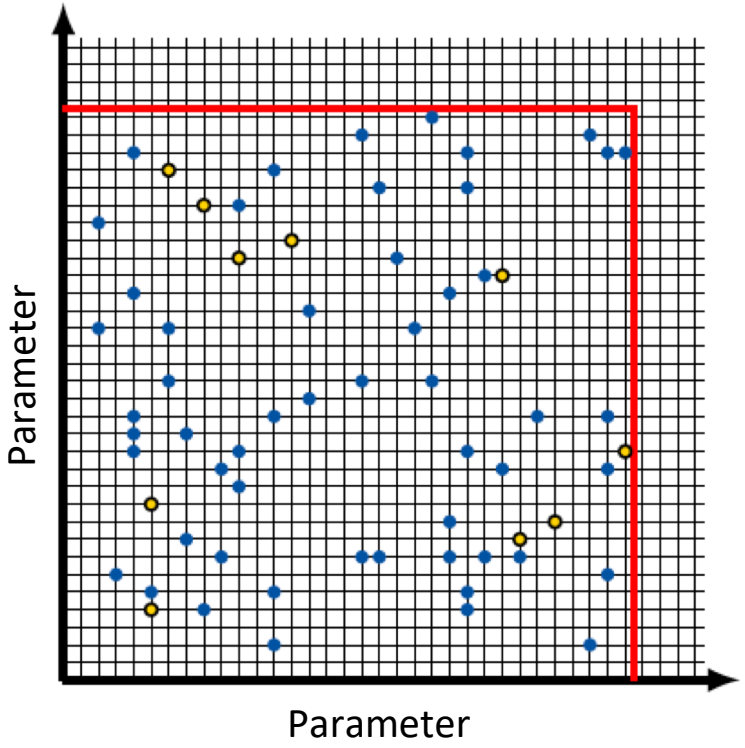
- Partition samples into k sets of equal size
- k -1 sets are used for learning; one is use for testing
- Build best fit polynomial surface from learning set
- Use polynomial surface to predict unknown variables in testing set
- Use points in the testing set to compute SSE

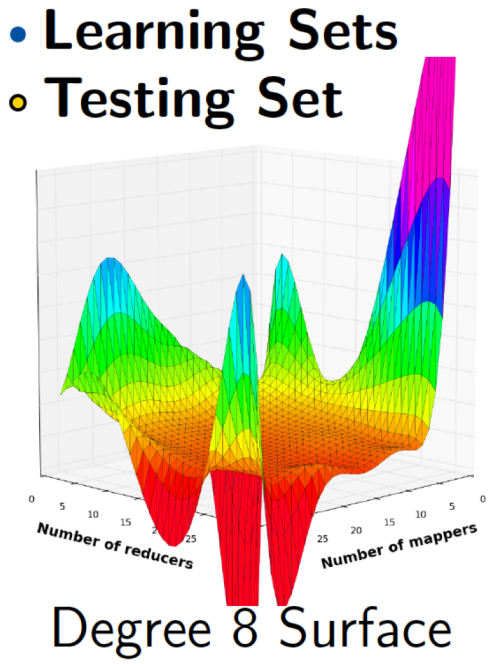




Step 3: Selecting the best surface using k-fold cross validation

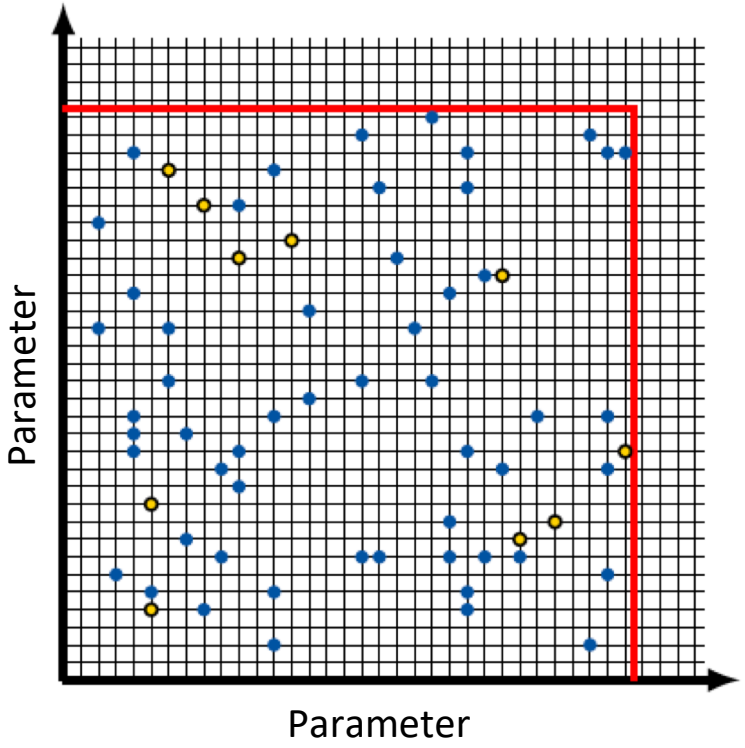
- Partition samples into k sets of equal size
- k -1 sets are used for learning; one is use for testing
- Build best fit polynomial surface from learning set
- Use polynomial surface to predict unknown variables in testing set
- Use points in the testing set to compute SSE

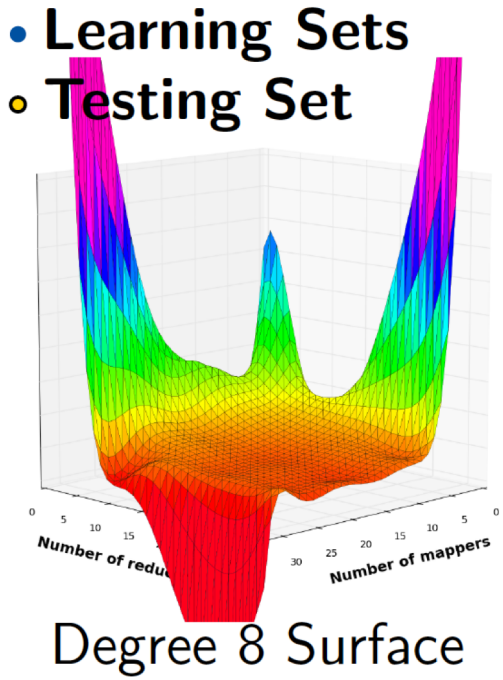




Step 3: Selecting the best surface using k-fold cross validation

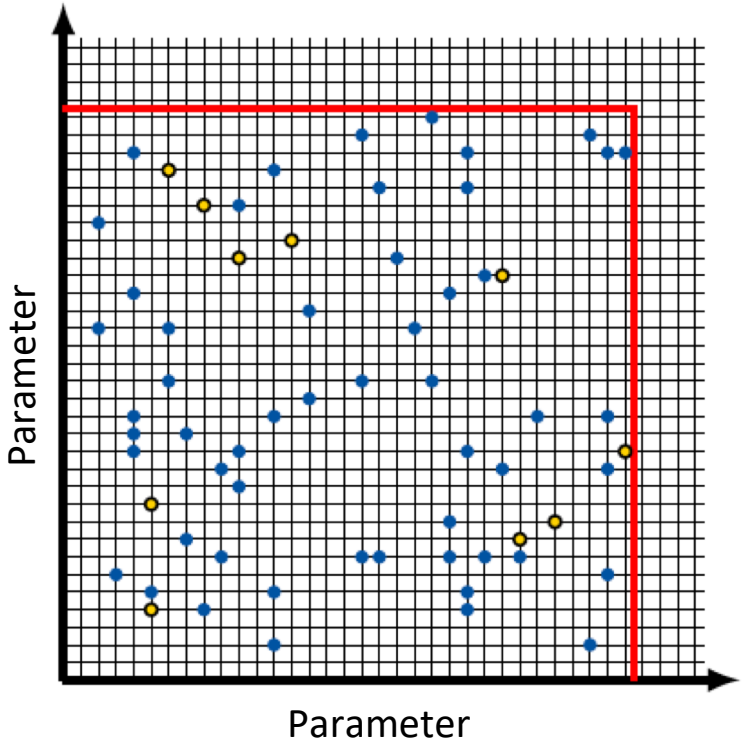
- Partition samples into k sets of equal size
- k -1 sets are used for learning; one is use for testing
- Build best fit polynomial surface from learning set
- Use polynomial surface to predict unknown variables in testing set
- Use points in the testing set to compute SSE



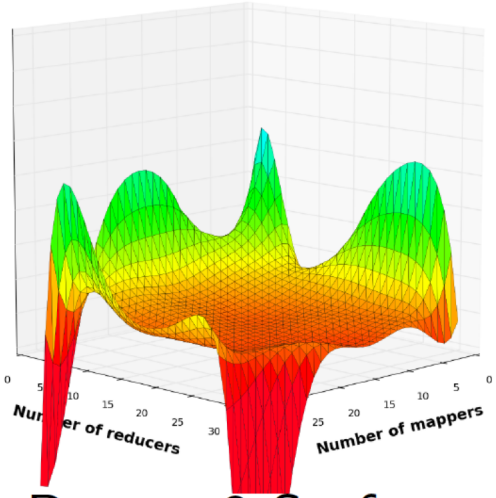


Step 3: Selecting the best surface using k-fold cross validation

- Partition samples into k sets of equal size
- k -1 sets are used for learning; one is use for testing
- Build best fit polynomial surface from learning set
- Use polynomial surface to predict unknown variables in testing set
- Use points in the testing set to compute SSE



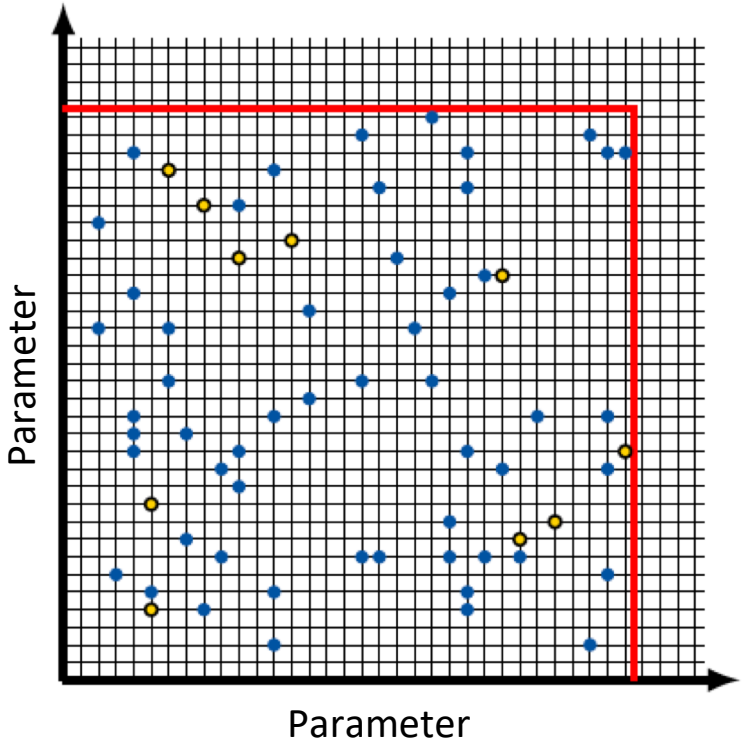
- Learning Sets
- Testing Set



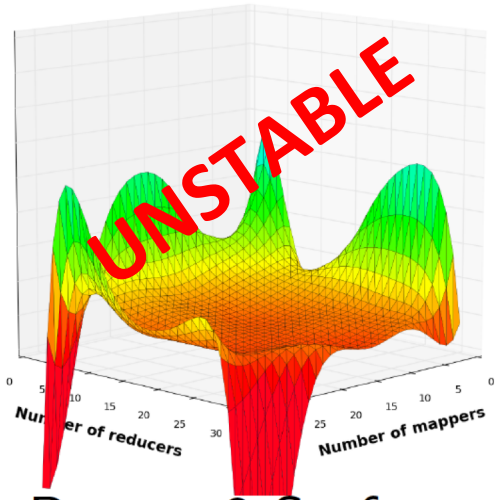
Degree 8 Surface

Step 3: Selecting the best surface using k-fold cross validation

- Partition samples into k sets of equal size
- k -1 sets are used for learning; one is use for testing
- Build best fit polynomial surface from learning set
- Use polynomial surface to predict unknown variables in testing set
- Use points in the testing set to compute SSE



- Learning Sets
- Testing Set

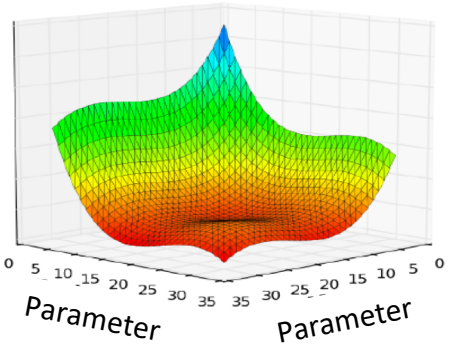


Degree 8 Surface

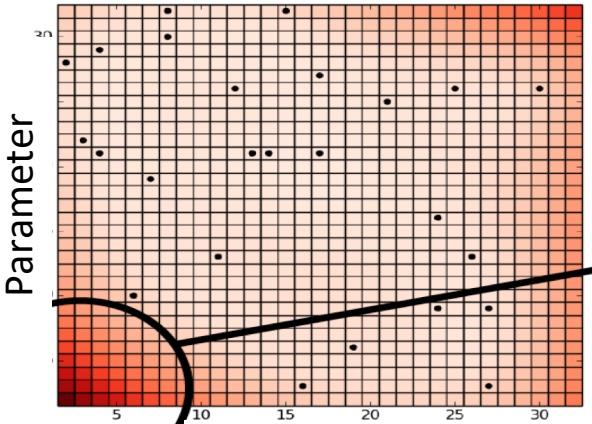
Step 4: Applying a confidence interval

The Model

$z = B\,x$

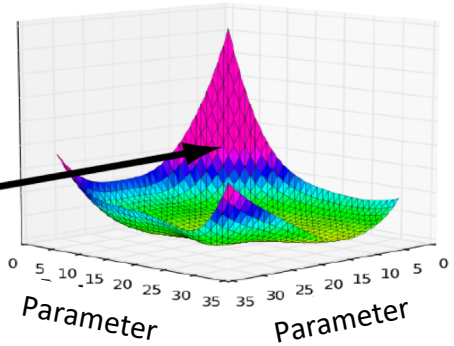


Width of Confidence Interval



Model + Confidence Interval

$z + e$



with *e* small for areas sampled more

KNN and SBM

KNN:

→ Use **local data**

→ Compute k and distance kernel using cross validation automatically

→ Compute weighted means with the kernel (***many values***)

Surrogate based model (SBM):

→ Use **all sampled data**

→ Use regression to generate one single polynomial model (***single polynomial model***)

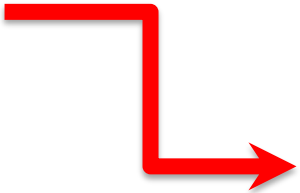
D. Rorabaugh, M. Guevara, R. Llamas, J. Kitson, R. Vargas, and **M. Taufer**. SOMOSPIE: A Modular SOil MOisture SPatial Inference Engine based on Data Driven Decisions. eScinece 2019.

Hybrid Piecewise Polynomial Modeling

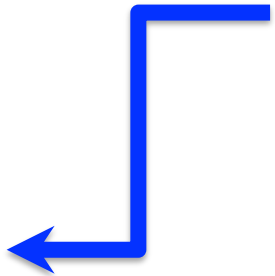
HYbrid Piecewise POlynomial modeling (HYPPO)

k Nearest Neighbors
- Use **local** data
- Compute the average (*many simple local models*)

Surrogate-Based Modeling
- Use **all** sampled data
- Construct **one polynomial** (*single complex global model*)



Hybrid modeling ? HYPPO
- Use **local** data
- Construct many **polynomials** (*many complex local models*)



KNN, SBM, and HYPPO

KNN:

→ Use **local data**

→ Compute k and distance kernel using cross validation automatically

→ Compute weighted means with the kernel (***many values***)

Surrogate based model (SBM):

→ Use **all sampled data**

→ Use regression to generate one single polynomial model (***single polynomial model***)

HYPPO (Hybrid Piecewise Polynomial Modeling):

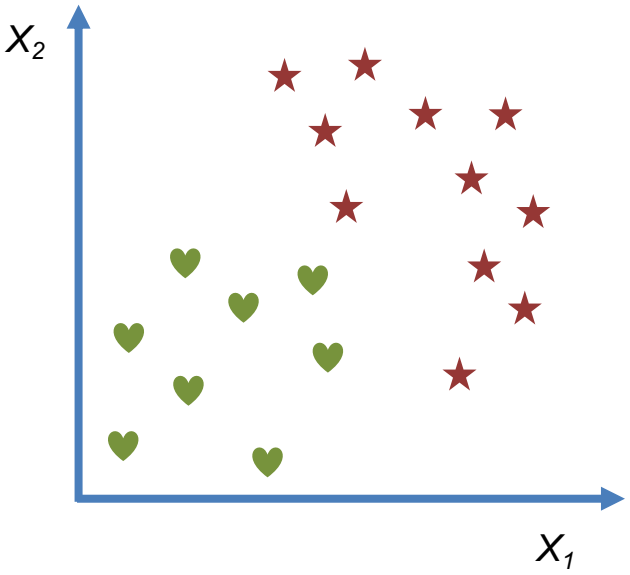
→ Use **local data**

→ Determine local polynomial degree using cross validation

→ Use regression to generate local polynomial model (***many* polynomial models**)

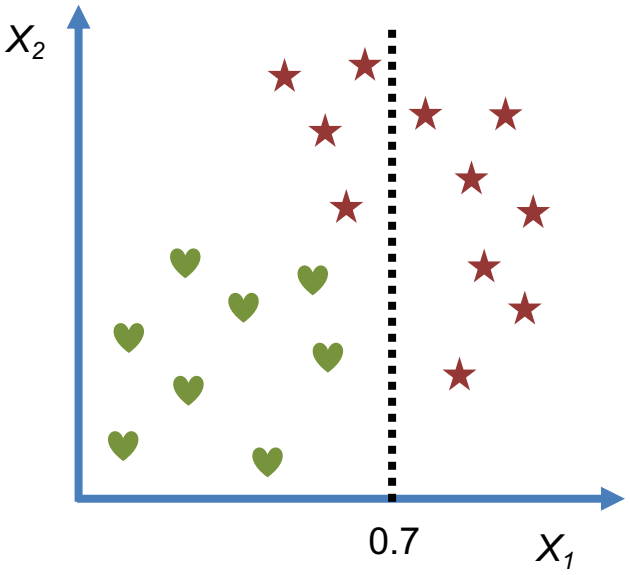
Random Forest

Divide and Conquer



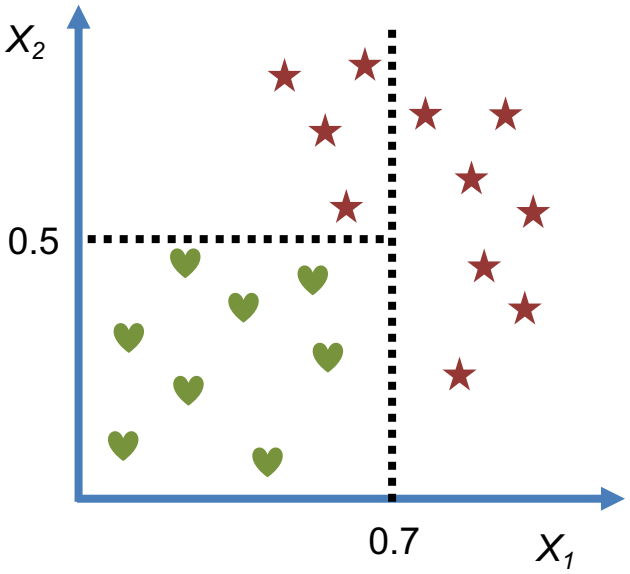
- Information gain is used to decide how to divide the dataset for classification
- The data is recursively divided until subdivision are all one class

Divide and Conquer



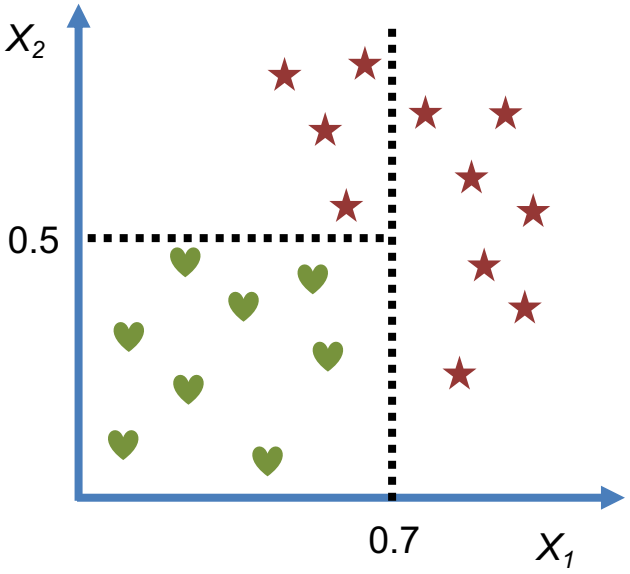
- Information gain is used to decide how to divide the dataset for classification
- The data is recursively divided until subdivision are all one class

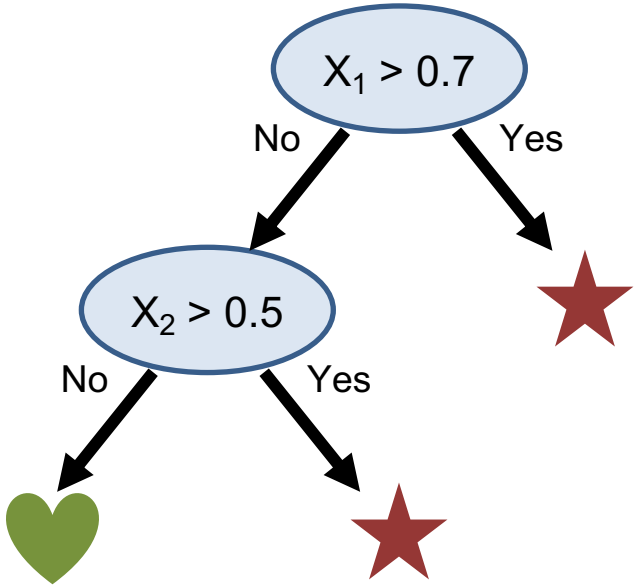
Divide and Conquer



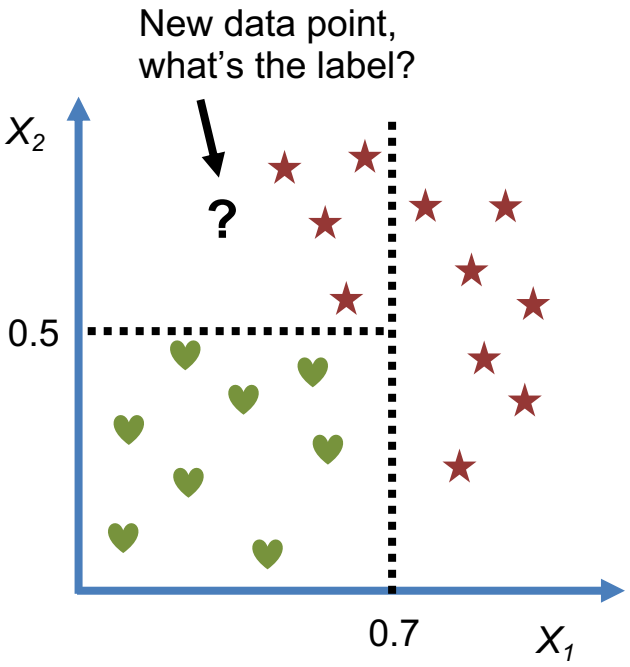
- Information gain is used to decide how to divide the dataset for classification
- The data is recursively divided until subdivision are all one class

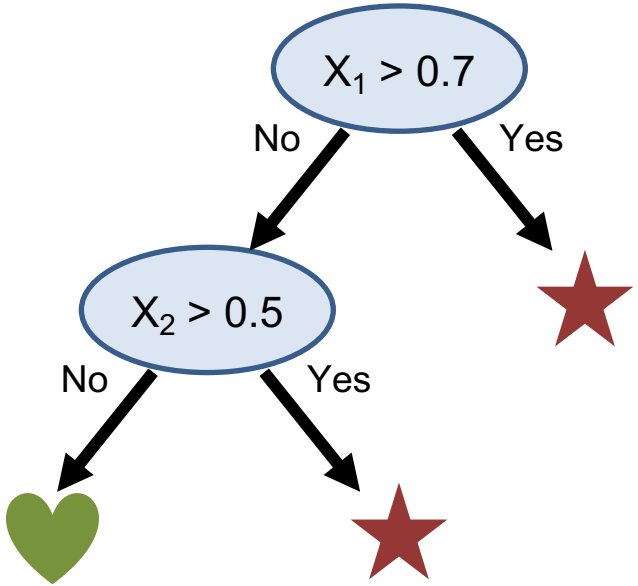
Divide and Conquer



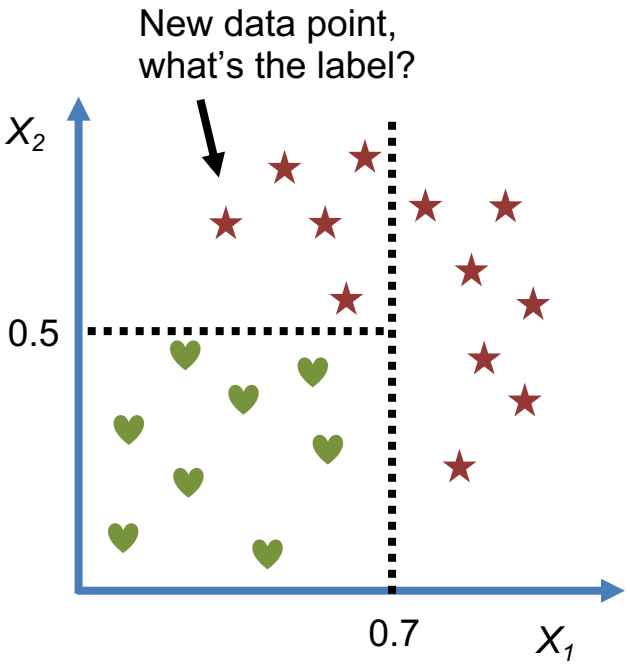


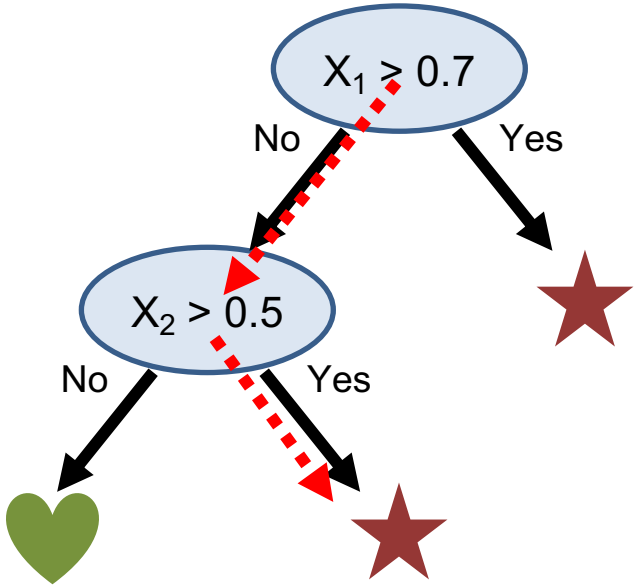
Divide and Conquer





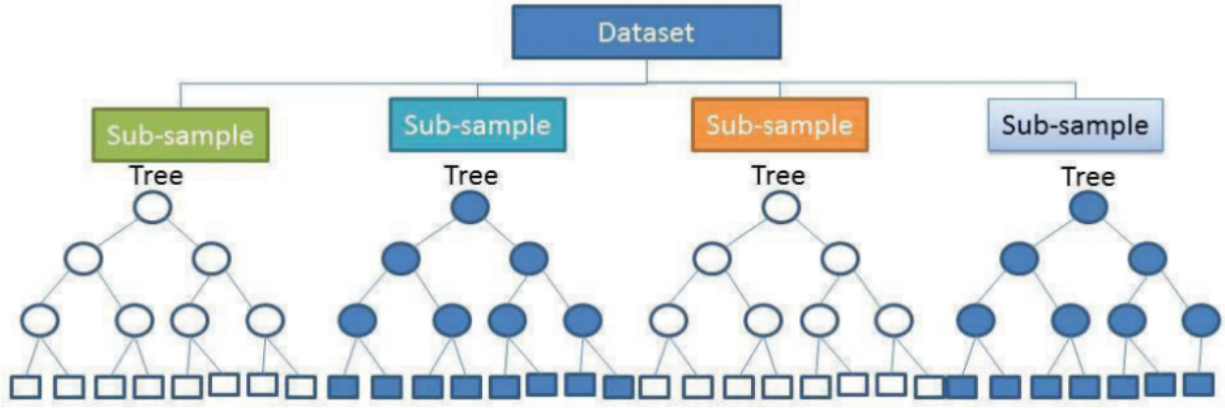
Divide and Conquer





Random Forests: Ensembles of Trees

- The prediction of many trees is better than the prediction of any one tree



- Each tree is built with a random sample of data rows and data features
- Each tree makes a prediction and the most popular prediction is returned

Assignment 10

Project

Define your project (March 13)

- Which dataset will you be using? How are you obtaining the data? (We will provide you the appropriate data from NHANES or Medicaid, if you choose to use either of these datasets.)
- What is (are) the scientific question(s) that you want to answer? Be as specific as possible.
- What is your strategy to answer the question(s)? Define a set of steps that, if implemented with your code, will allow you to answer the question(s). Be as specific as possible.
- What is the tentative title of your project?

Create a new notebook with your solution (March 27)

- Write down the steps of your solution in distinct text cells; add one or multiple cells (as needed) to hold your code for each step. You can leave these software cells empty for the moment. Expand the text cells describing your solution.
- Add visualization cells that allow you to visualize results. You can leave these software cells empty for the moment.
- Add software to the code cells that upload data from source and pre-process data.
- Push your notebook into your GitHub repository as frequently as needed.

Finalize software and run tests within your notebook (April 3)

- Add the software that implements the method (or methods) to analyze your data.
- Add visualization cells that allow you to visualize results
- Push your notebook into your GitHub repository as frequently as needed.

Build a set of 15 slides that describe your work and get feedback (April 10)

- Build a set of ppt slides (use template provided) that summarize your work; use text slides to tell the story of your project and figures with the key results of your work.
- Make sure your slides include: motivation and problem definition, related work and background, your methodology (e.g., with flowcharts and code sections), your results, summary, and conclusions.

Create your poster and get feedback (April 17)

- Copy and paste your slides into the poster template.
- Shuffle as needed, extend and fill gaps, embellish fonts and text, enlarge text and figures to make them readable.
- **Submit your poster in GitHub for printing (April 20)**
- **Present your poster at the annual EECS COSC 562 poster session (April 23)**

Reading

Reading

- T. Johnston, M. Alsulmi, P. Cicotti and M. Taufer. Performance Tuning of MapReduce Jobs Using Surrogate-Based Modeling. In *Proceedings of the International Conference on Computational Science (ICCS)*, pp. 49 – 59. Reykjavik, Iceland. June 1 – 3, 2015.
- T. Johnston, C. Zannin, and M. Taufer. HYPPO: A Hybrid, Piecewise Polynomial Modeling Technique for Non-Smooth Surfaces. In *Proceedings of the 28th IEEEE International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD)*, pp. 1 – 8. Los Angeles, CA, USA. October 26 – 28, 2016.
- Breiman, L. Random Forests. *Machine Learning* **45,** 5–32 (2001). https://doi.org/10.1023/A:1010933404324