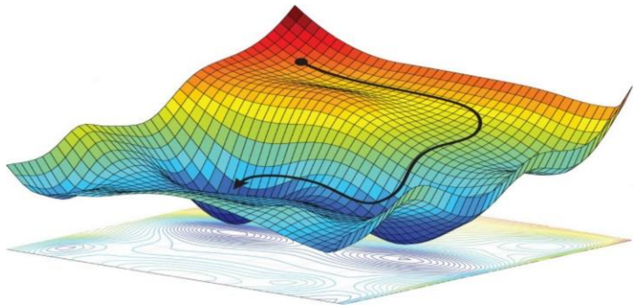
System Intelligence Laboratory

WONIK-KAIST FTC 기술 전수 세미나:
Decision-Making with Data-Driven Model

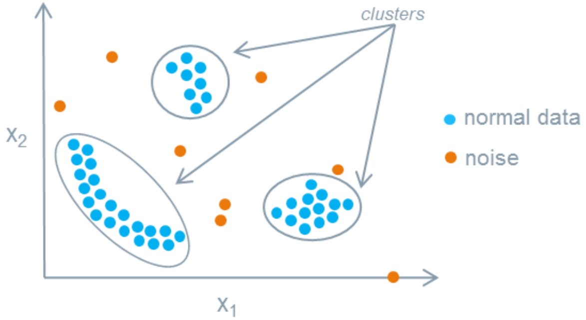
Dept. of Industrial & Systems engineering, KAIST
Chihyeon Song, Haewon Jung, Jinkyoo Park

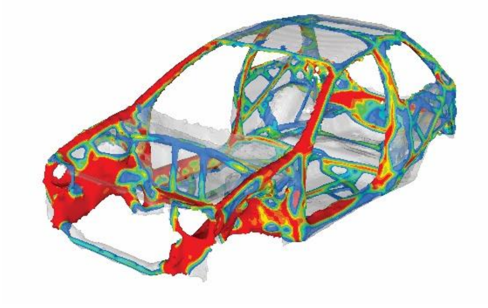
Decision-Making

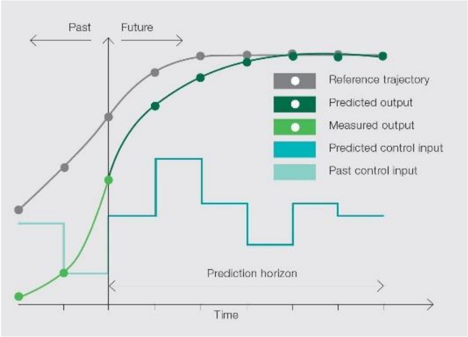
- Decision-making: What we really want to do

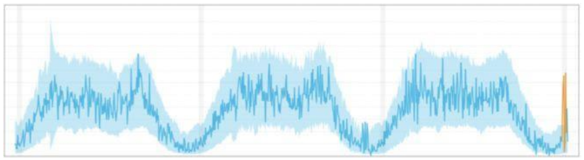


$$\min_{u_1,..,u_T} \sum_{t=1}^{T} c(x_t, u_t)$$
$$s.t. \quad x_{t+1} = f(x_t, u_t)$$



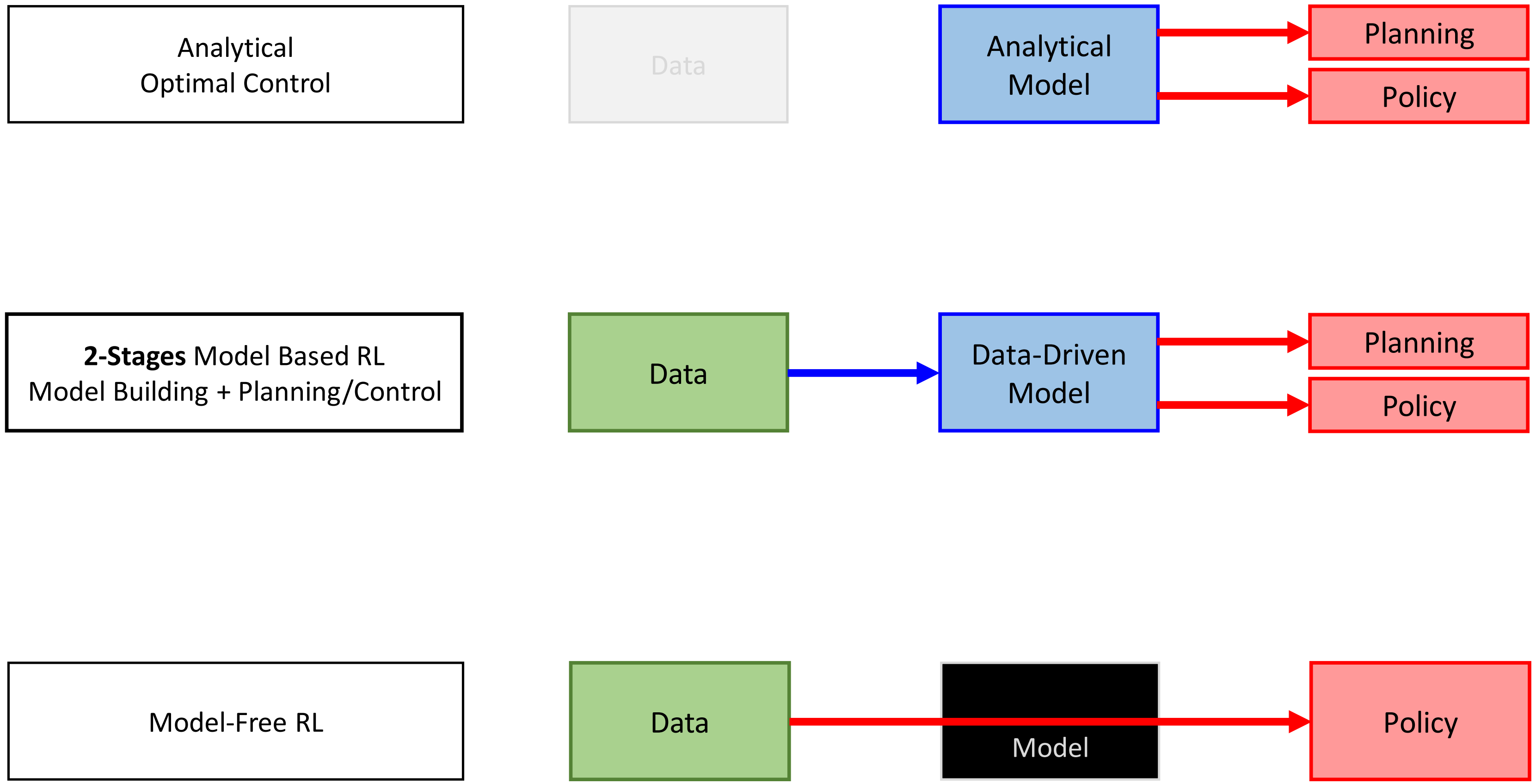






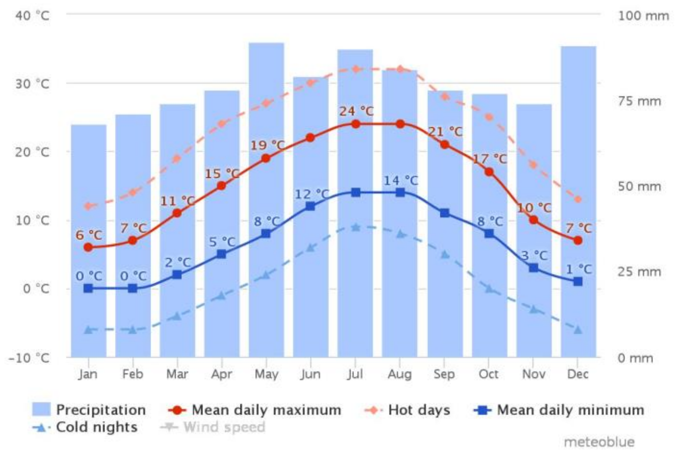
Model-based optimizations **Optimal controls** **Anomaly detections**

Ways to Make a Decision

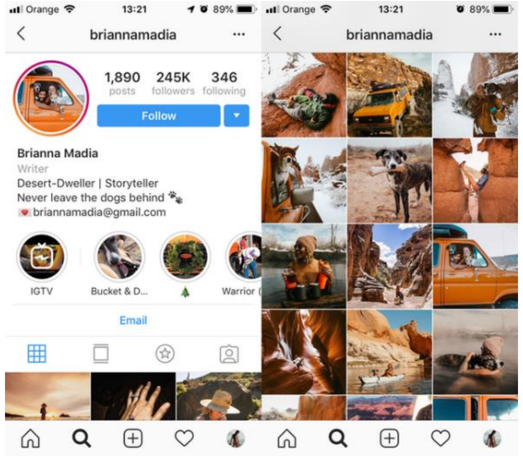


Decision-Making with Data-Driven Models

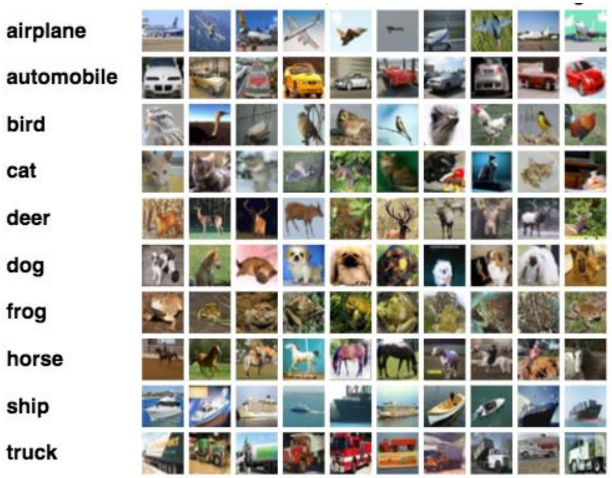
- Data: discrete value that contains information



Weather



SNS

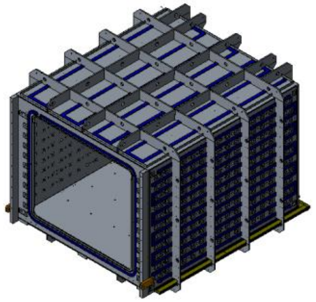


Image

Decision-Making with Data-Driven Models

- Model: Theoretical representation of a system
  - Will be used in decision-making instead of the actual system

Z-set (Work set)



Glass TC

Complex Systems

Modeling

Input x

Model

Output $\hat{y}$

Model

$\hat{y} = \hat{f}(x; \theta)$

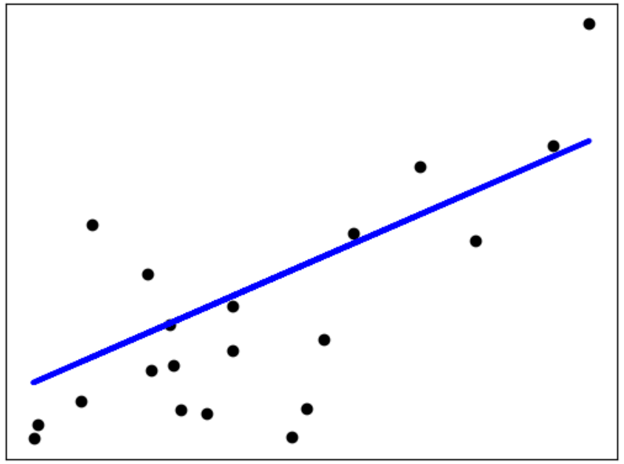
ex: ($\hat{y} = 2x^3 + 5$)

$\hat{f}(\cdot)$: model class

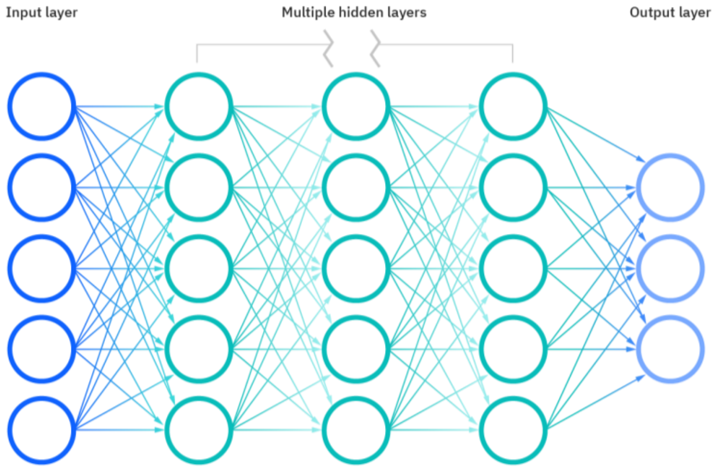
θ: model parameters

Decision-Making with <u>Data-Driven Models</u>

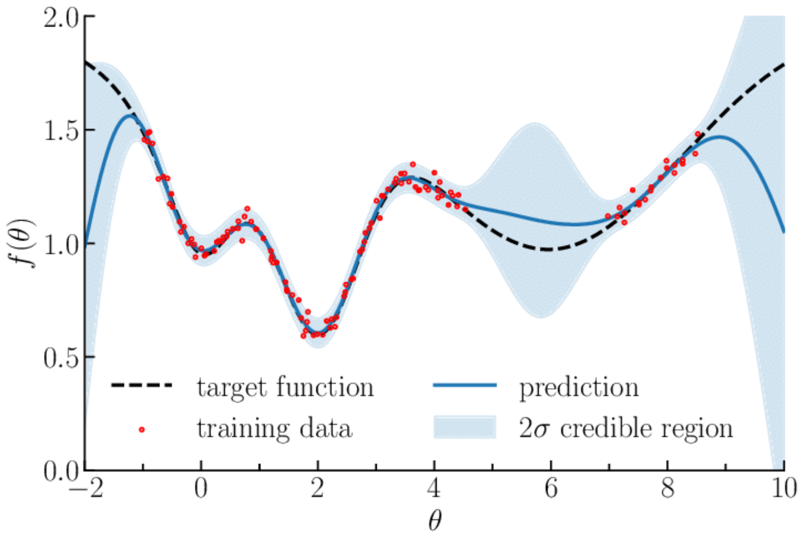
- Data-driven model: Model parameters are *learned* from the data
  - Collect the information from the data in terms of modeling



Linear model



Deep neural networks



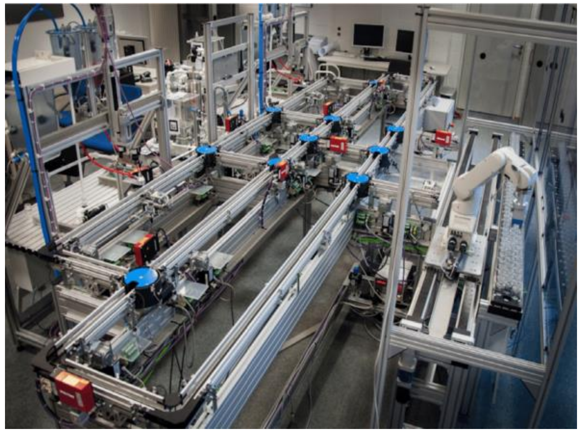
Gaussian process

Why Data-Driven Decision-Making?

- Complexity of modern systems
  - Data-driven approach doesn't require any background knowledge of the system

$$\frac{\partial}{\partial x}\left(k\frac{\partial T}{\partial x}\right)+\frac{\partial}{\partial y}\left(k\frac{\partial T}{\partial y}\right)+\frac{\partial}{\partial z}\left(k\frac{\partial T}{\partial z}\right)+q_V=\rho c_p\frac{\partial T}{\partial t}$$

where
k is the materials conductivity [W.m^{-1}.K^{-1}]
q_V is the rate at which energy is generated per unit volume of the medium [W.m^{-3}]
ρ is the density [kg.m^{3}]
c_p is the specific heat capacity [J.kg^{-1}.K^{-1}]

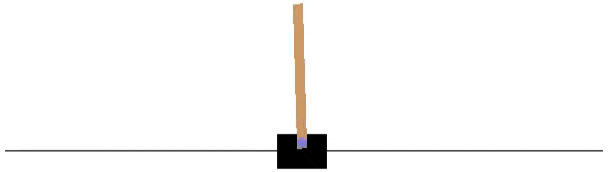


Scientific Knowledge **Modern Complex Systems**

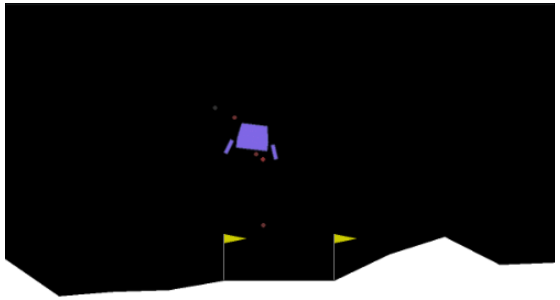
Why Data-Driven Model?

- In the real-world, we often have limited amount of data

- With an appropriate model, more data-efficient than model-free approach
  - More practical approach to the real-world systems

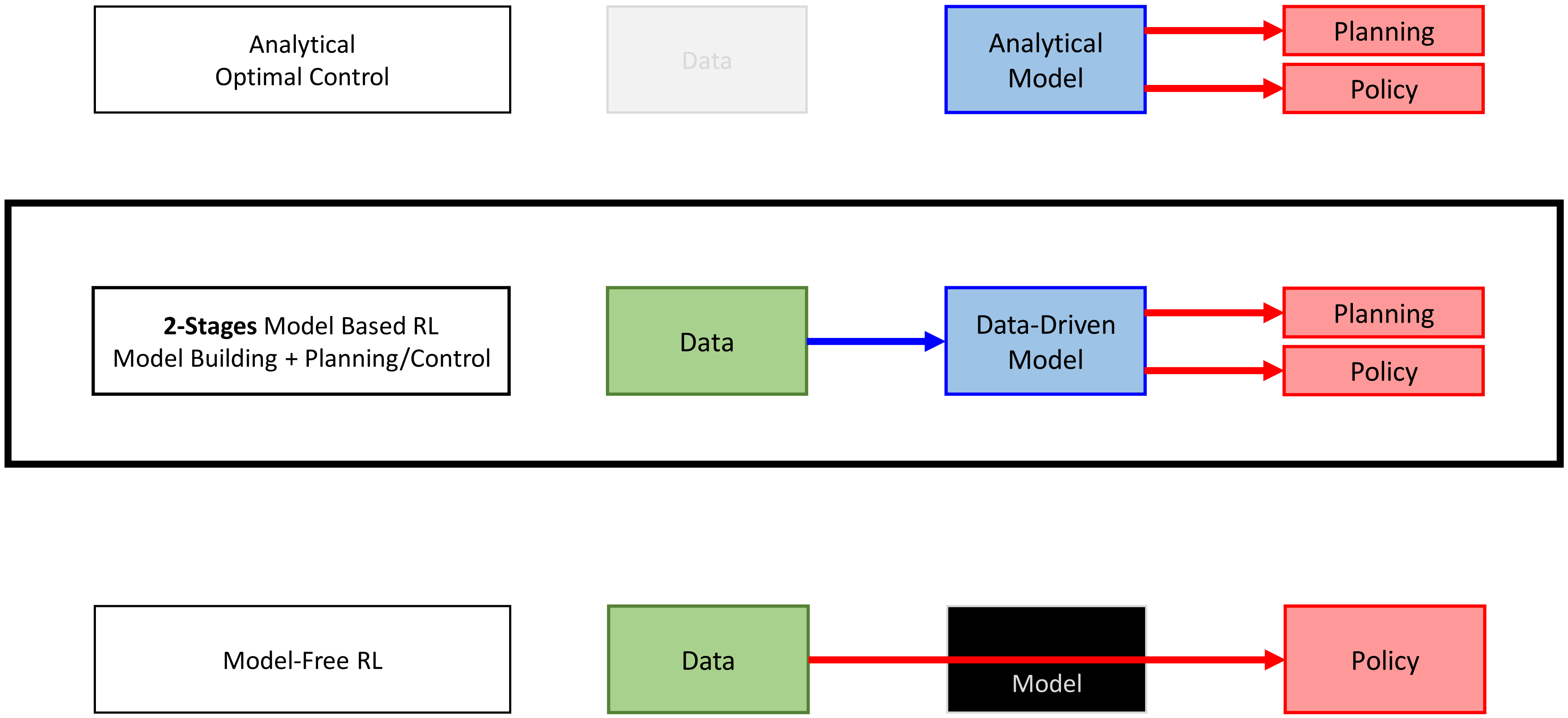


CartPole



LunarLander

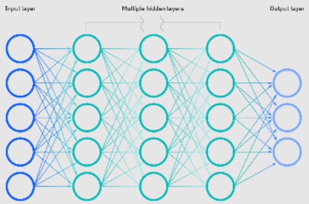
How to Make a Decision

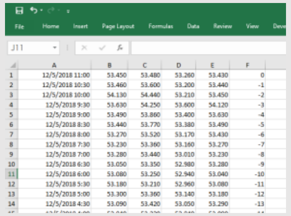


Two Big Questions

Q1. How we build data-driven model?

- Formulate System
- Model Selection
- Data Preprocessing
- Learn Model





	A	B	C	D	E	F
1	12/5/2018 11:00	53.450	53.480	53.260	53.430	0
2	12/5/2018 10:30	53.460	53.600	53.200	53.440	-1
3	12/5/2018 10:00	54.130	54.440	53.210	53.450	-2
4	12/5/2018 9:30	53.630	54.250	53.600	54.120	-3
5	12/5/2018 9:00	53.490	53.860	53.400	53.630	-4
6	12/5/2018 8:30	53.440	53.770	53.380	53.490	-5
7	12/5/2018 8:00	53.270	53.520	53.170	53.430	-6
8	12/5/2018 7:30	53.230	53.360	53.160	53.270	-7
9	12/5/2018 7:00	53.280	53.440	53.010	53.230	-8
10	12/5/2018 6:30	53.050	53.350	52.980	53.280	-9
11	12/5/2018 6:00	53.080	53.250	52.940	53.040	-10
12	12/5/2018 5:30	53.180	53.210	52.960	53.080	-11
13	12/5/2018 5:00	53.300	53.360	53.140	53.180	-12
14	12/5/2018 4:30	53.090	53.420	53.050	53.290	-13

Q2. How we make a decision with model?

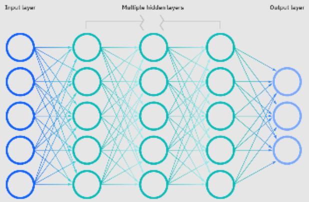
- Define Decision-Making
- Formulate Optimization Problem
- Solve Control Optimization
- Validate Control Performance

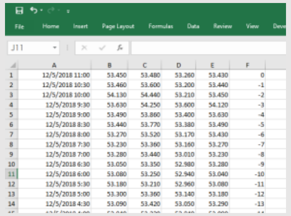
$$\min_x f(x)$$
$$s.t.\ g_i(x) \le 0\ \forall\, i = 1, \dots, I$$
$$h_j(x) = 0\ \forall j = 1, \dots, J$$

Two Big Questions

Q1. How we build data-driven model?

- Formulate System
- Model Selection
- Data Preprocessing
- Learn Model





Day 1

Q2. How we make a decision with model?

- Define Decision-Making
- Formulate Optimization Problem
- Solve Control Optimization
- Validate Control Performance

$$\min_{x} f(x)$$
$$s.t.\ g_i(x) \le 0\ \forall i = 1, \dots, I$$
$$h_j(x) = 0\ \forall j = 1, \dots, J$$

Day 2

Q1. How We Build Data-Driven Model?

Q1. How We Build Data-Driven Model?

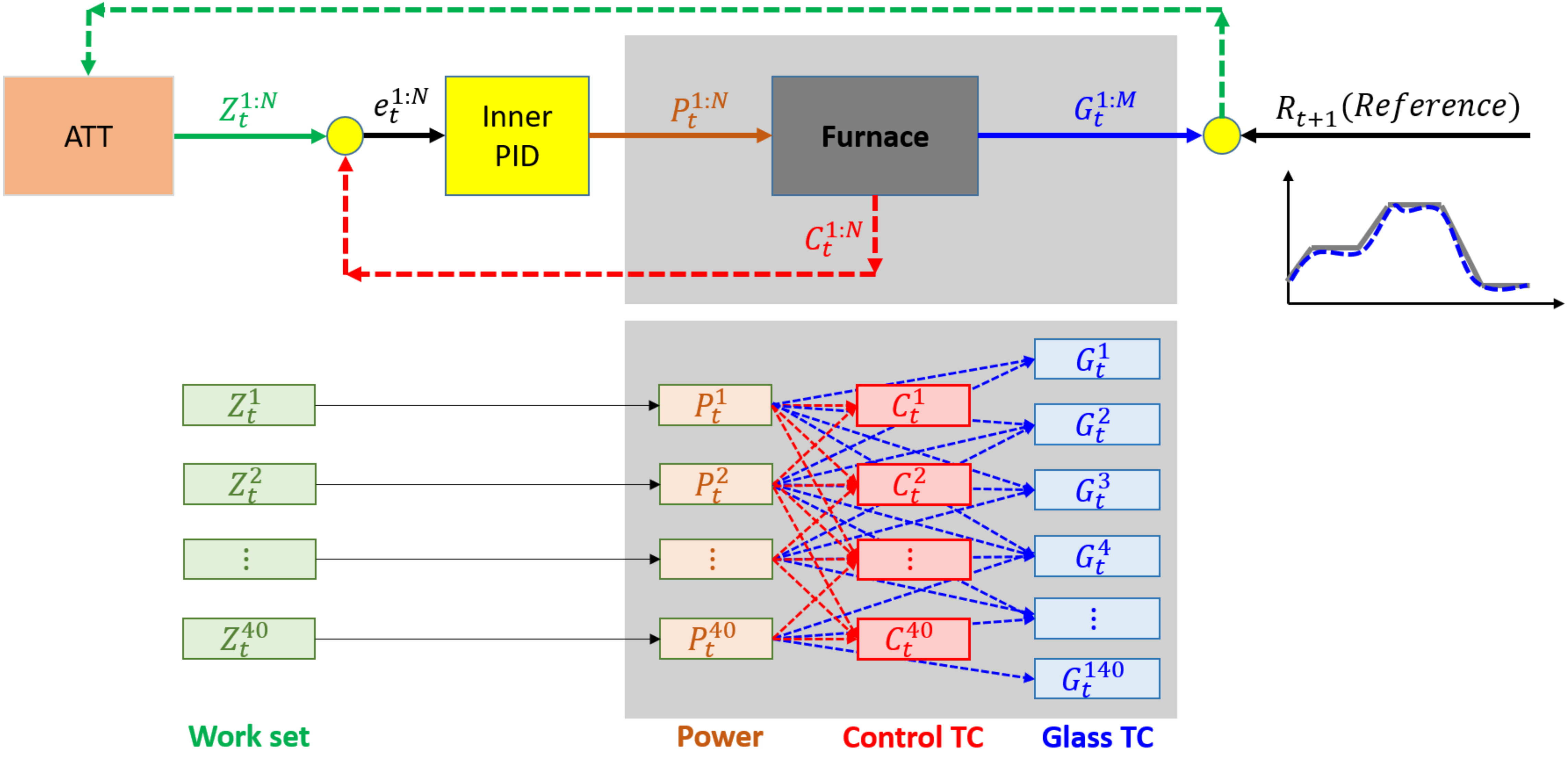
- Step 1. Formulate into a mathematical system

- Step 2. Select the model class

- Step 3. Prepare the data

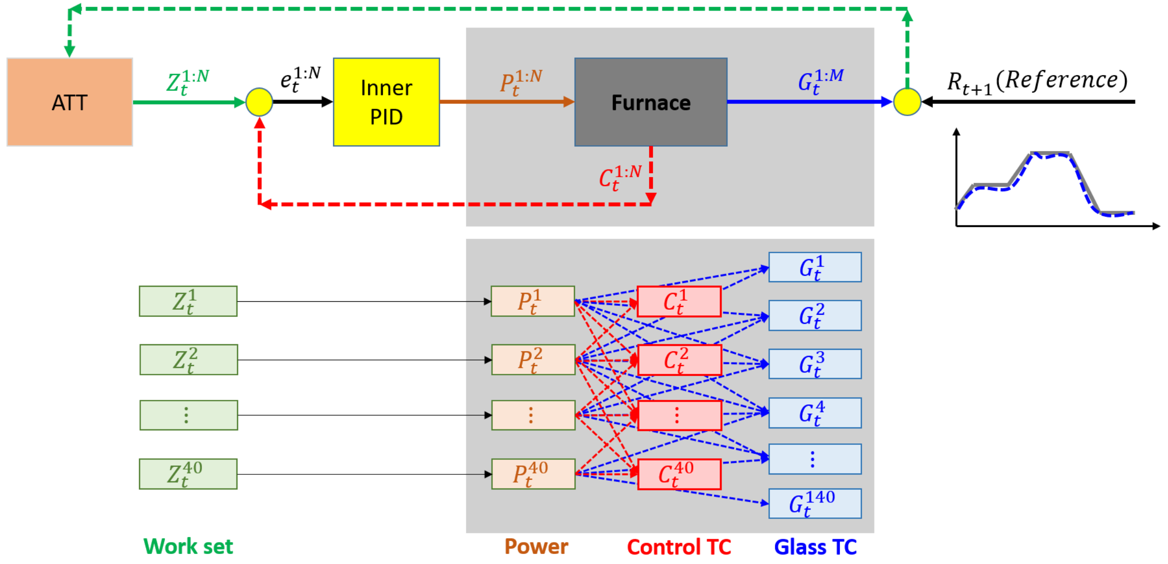
- Step 4. Learn (Train) the model

Step 1. Formulate Into Mathematical System



Step 1. Formulate Into Mathematical System

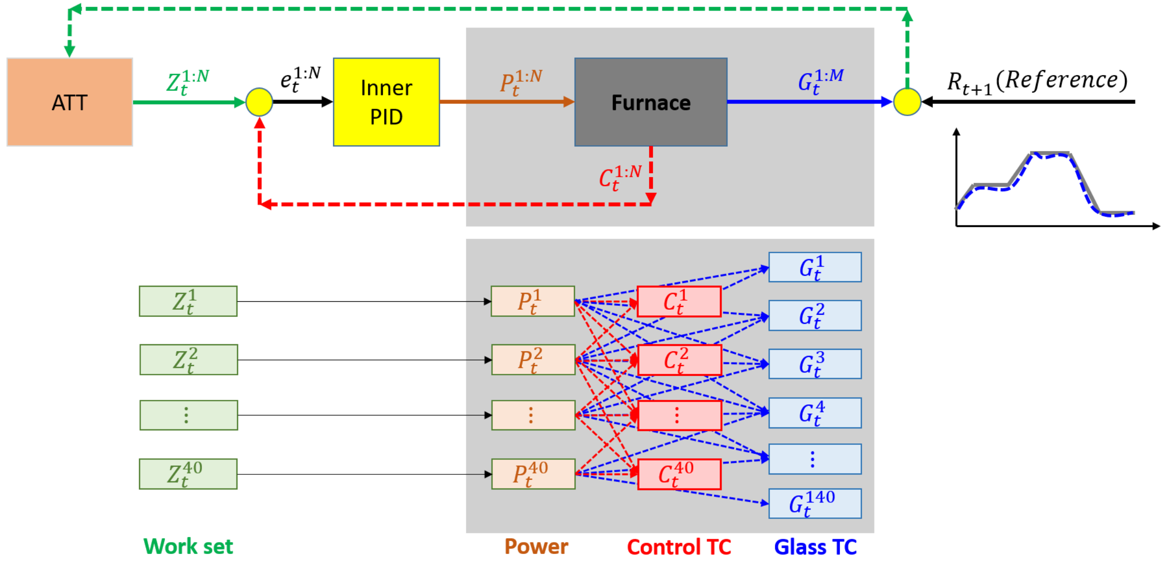
- Control Input u_t: What we can control to the system
  - Workset
- Observation o_t: What we can observe from the system
  - Power, Control TC, (Glass TC)
- State x_t: What we care in the system
  - Glass TC
  - NOT always observable



Step 1. Formulate Into Mathematical System

- Assume Glass TC is observable

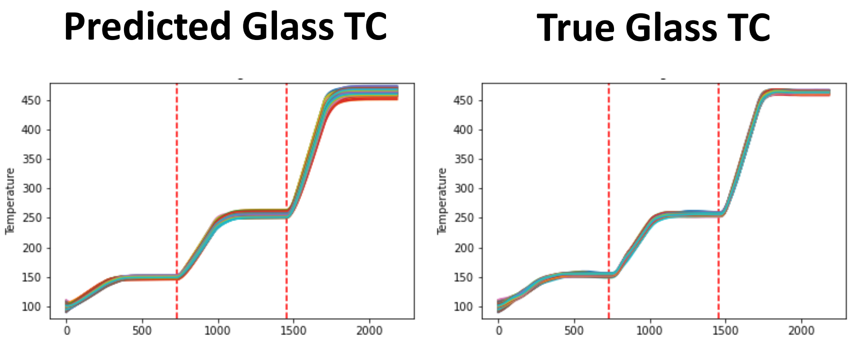
- Model: What we want to know in the system
  - Interaction between future Glass TC and other variables
  - $\hat{\boldsymbol{x}}_{t+1} = \hat{\boldsymbol{f}}(\boldsymbol{x}_t, \boldsymbol{u}_t; \boldsymbol{\theta})$, predicted Glass TC
  - Let true system is $x_{t+1} = f(x_t, u_t)$



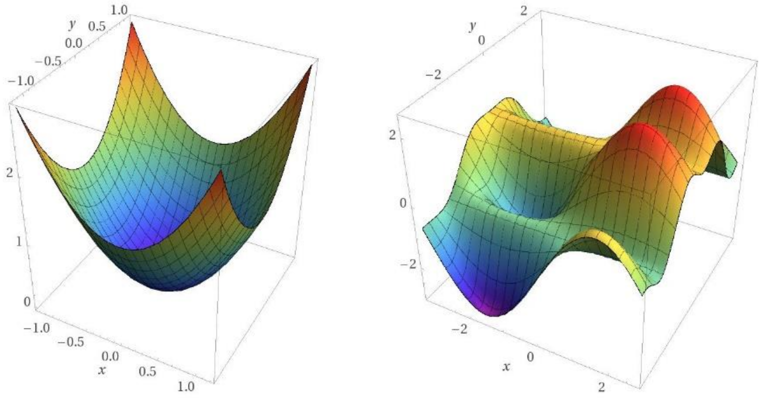
Step 2. Select Model Class

- Many different model classes
  - Linear, Deep neural network, Graph neural network, Gaussian process, etc.

- There is no perfect answer to choose the model class



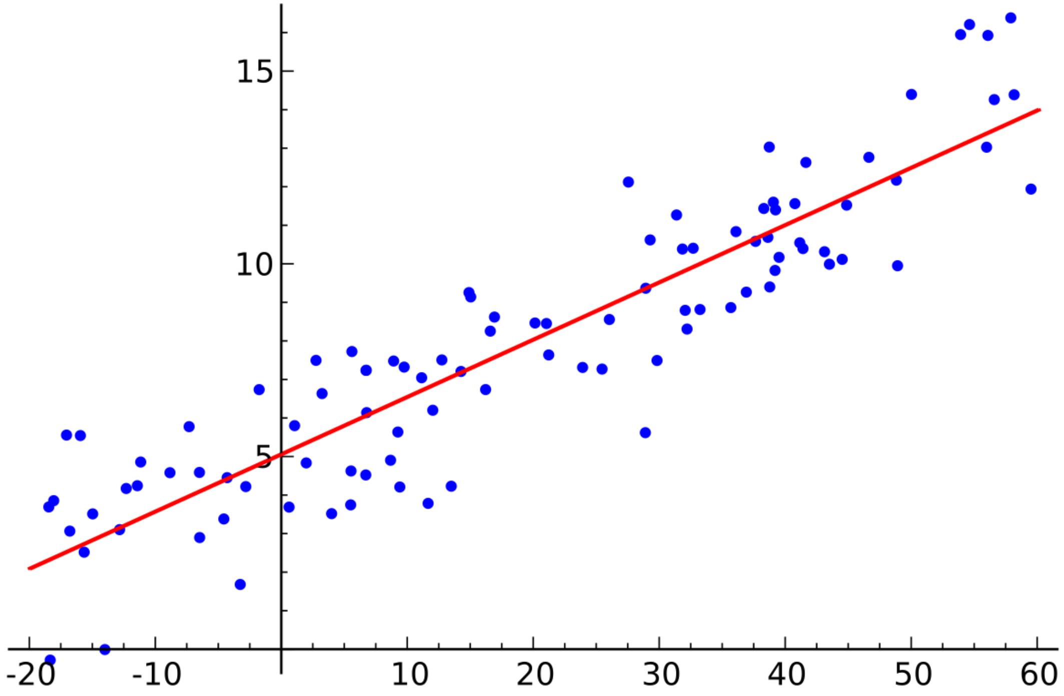
Model expressivity



Optimization solvability

Linear Model

- Simplest data-driven model
- Parameters: a weight matrix W and bias vector b



$$y = Wx + b$$

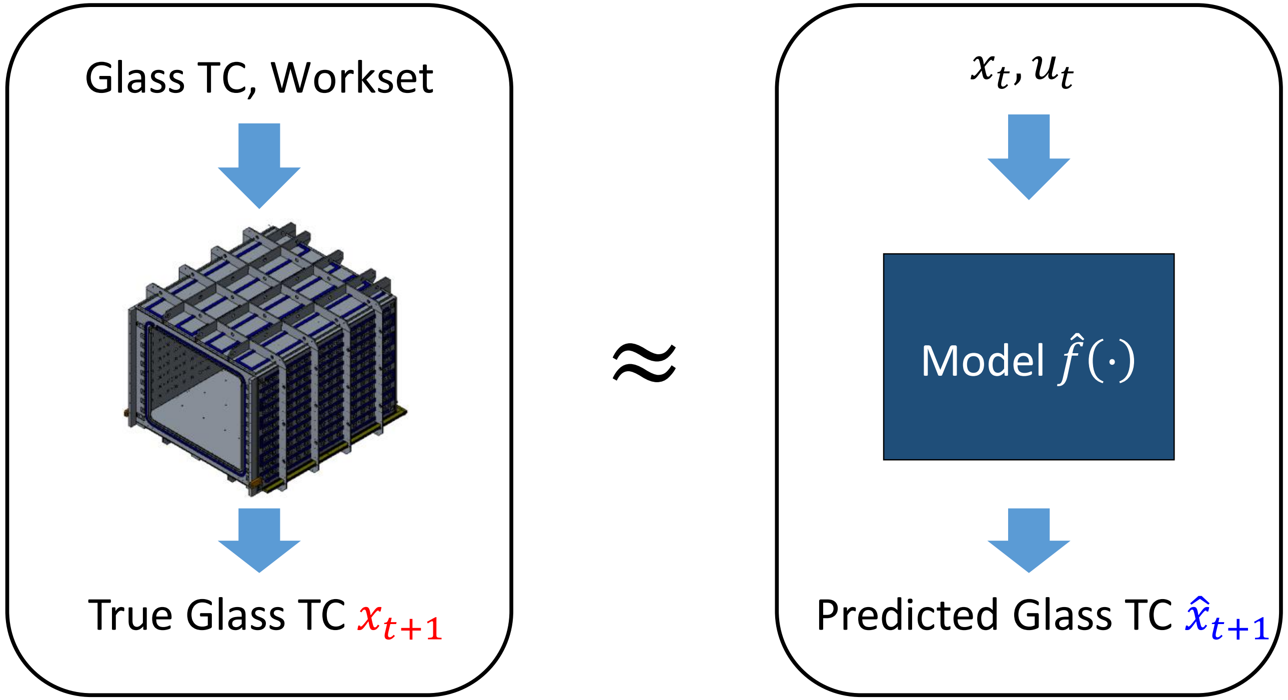
Step 3. Prepare the Data

- Let's skip for now

Step 4. Learn the Model

- Role of model: Substitute the true system in decision-making
  - Good model = Good approximation to the true system

- Make the predicted Glass TC as close as the true Glass TC



Step 4. Learn the Model

- Meaning of 'learning model'
  - Find the optimal model parameters θ from the data
  - $\hat{f}(x_t, u_t; \theta) \approx f(\cdot)$ as possible as we can

- Mathematically,

$$\min_{\theta} \frac{1}{|\mathcal{D}|} \sum_{(x_t, u_t, x_{t+1}) \in \mathcal{D}} (x_{t+1} - \hat{x}_{t+1})^2$$

$$\text{s.t. } \hat{x}_{t+1} = \hat{f}(x_t, u_t; \theta)$$

- $\mathcal{D}$: prepared dataset

- How can we solve this? Gradient Descent Algorithm!

General Description of Optimization Problem

Objective function $$\min_{x} f(x)$$

Constraints $$s.t.\ g_i(x) \leq 0\ \forall\ i = 1, \dots, I$$ Inequality constraints

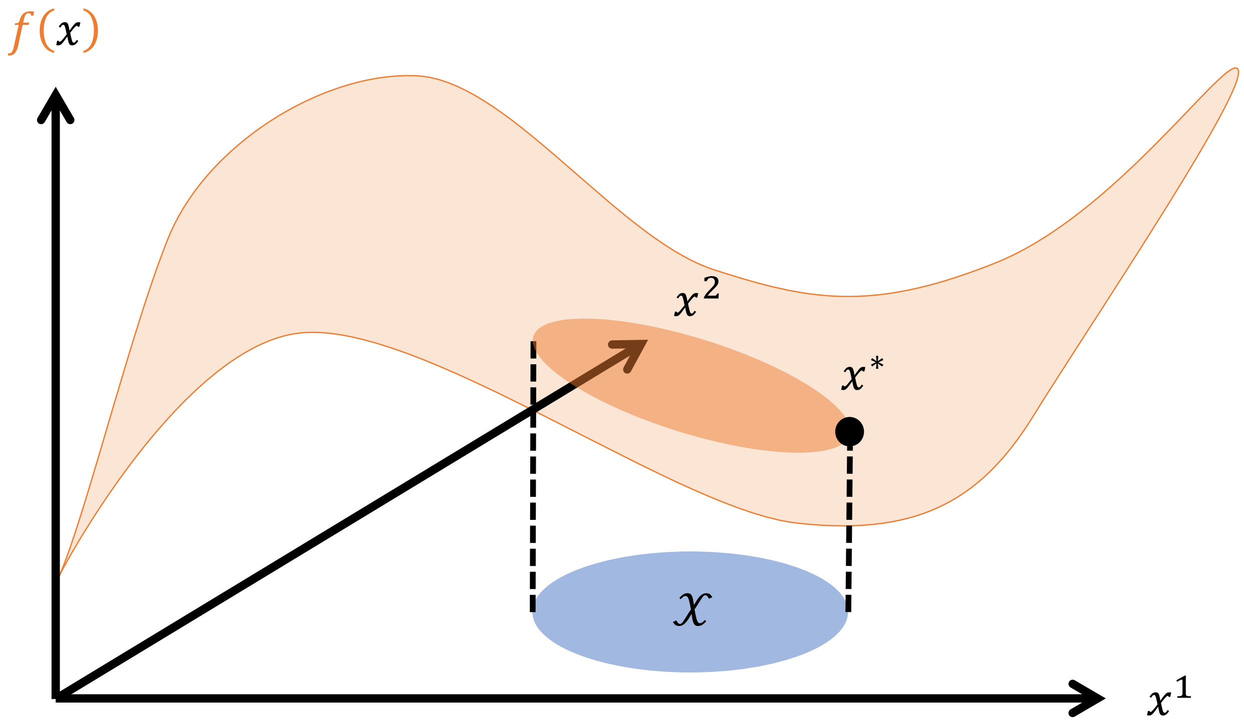
($s.t.$: Acronym of '**s**ubject **t**o') $$h_j(x) = 0\ \forall j = 1, \dots, J$$ Equality constraints

where $x = \begin{bmatrix} x^1 \\ \vdots \\ x^n \end{bmatrix} \in \mathbb{R}^n$, $f, g_i, h_j : \mathbb{R}^n \longrightarrow \mathbb{R}\ \forall\ i = 1, \dots, I, j = 1, \dots, J$

- **Optimal solution** x^* is the minimizers of the optimization problem. Possibly exist **multiple** optimal solutions
- **Optimal value** $f(x^*)$ is the minimal (maximal) function value
- **Feasible solutions (search space)** $\mathcal{X} = \{x \in \mathbb{R}^n | g_i(x) \leq 0\ \forall\ i = 1, \dots, I, h_j(x) = 0\ \forall j = 1, \dots, J\}$

- If objective or constraints are not scalar valued, then we can't solve the problem with optimization techniques.
  - such setting requires different solution concepts → Game Theory

Visual Understanding of Optimization Problem



$$\mathcal{X} = \{x \in \mathbb{R}^n | g_i(x) \le 0 \; \forall \, i = 1, \ldots, I, h_j(x) = 0 \; \forall j = 1, \ldots, J\}$$

- Naïve approach: “Trial-and-error”
  - Plug in every solution in $\mathcal{X}$ and Find the solution that is corresponds to the minimal function value.

Can we Do Better?

- If the objective or constraint not differentiable,
  - We can come up with heuristics:
    - Simulated Annealing, Genetic algorithms.
  - If the optimization problem is combinatorics, use Branch and Bound (B&B) or MILP etc.

- What if the objective and constraint is differentiable?
  - Use Gradient / Hessians to solve the optimization problem efficiently.

Sidewalk: Extremum and Optimum

Recall our memory of high school mathematic class

문제) 함수 $f(x) = (x-3)^2 + 2$ 일 때, 함수의 최솟값을 구하면?

① 5　② 4　③ 3　④ 2　⑤ 1

해설) $\frac{df(x)}{dx} = 2(x-3)$ 이며, 극댓값 정리, $\frac{df(x)}{dx} = 0$ 인 극값에서, 최대/최소값이 존재, 를 활용해서 $2(x-3) = 0$ 를 만족하는 극값을 찾으면, $x = 3$.

극점 (Extreme point = local optima) 는 gradient $\frac{df(x)}{dx}$ 가 0 이 되는 지점!

Example 1) $f(x) = ax^2 + bx + c$ for $a > 0$, then $x^* = -\frac{b}{2a}$

Example 2) $f(x) = \sin x$, then $x^* = \left(2k + \frac{3}{2}\right)\pi \ \forall \ k \in \mathbb{Z}$

How to Solve Unconstrained Optimization Problem?

$$\min_{x} \; f(x)$$

- What if $f(x)$ is complicate so we can't find analytical form of optimal solution?
- Use "Gradient descent <u>algorithm</u>" to iteratively find the optimal solution x^*

Taylor Approximation

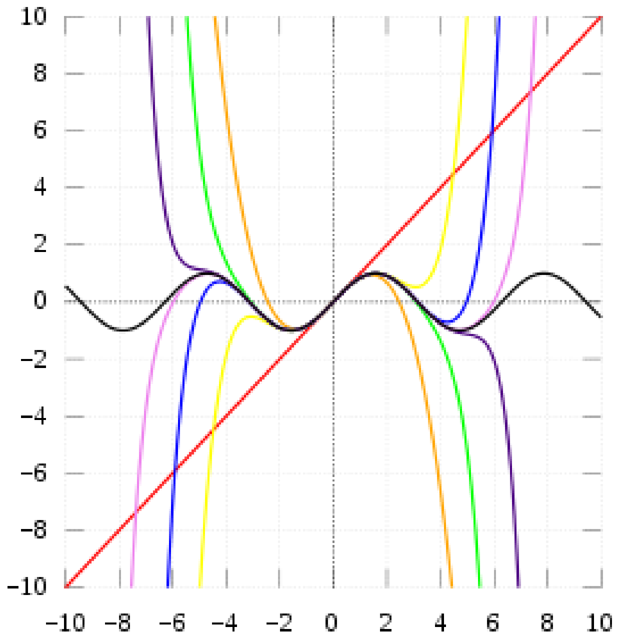
- Taylor approximation is a local approximation of function.

$$f(x) = \sum_{n=0}^{\infty} \frac{f^n(a)}{n!}(x-a)^n$$ Taylor series

$$= \sum_{n=0}^{T} \frac{f^n(a)}{n!}(x-a)^n$$ T order Taylor approximation

$f^n(a)$: n^{th} derivative of f at a

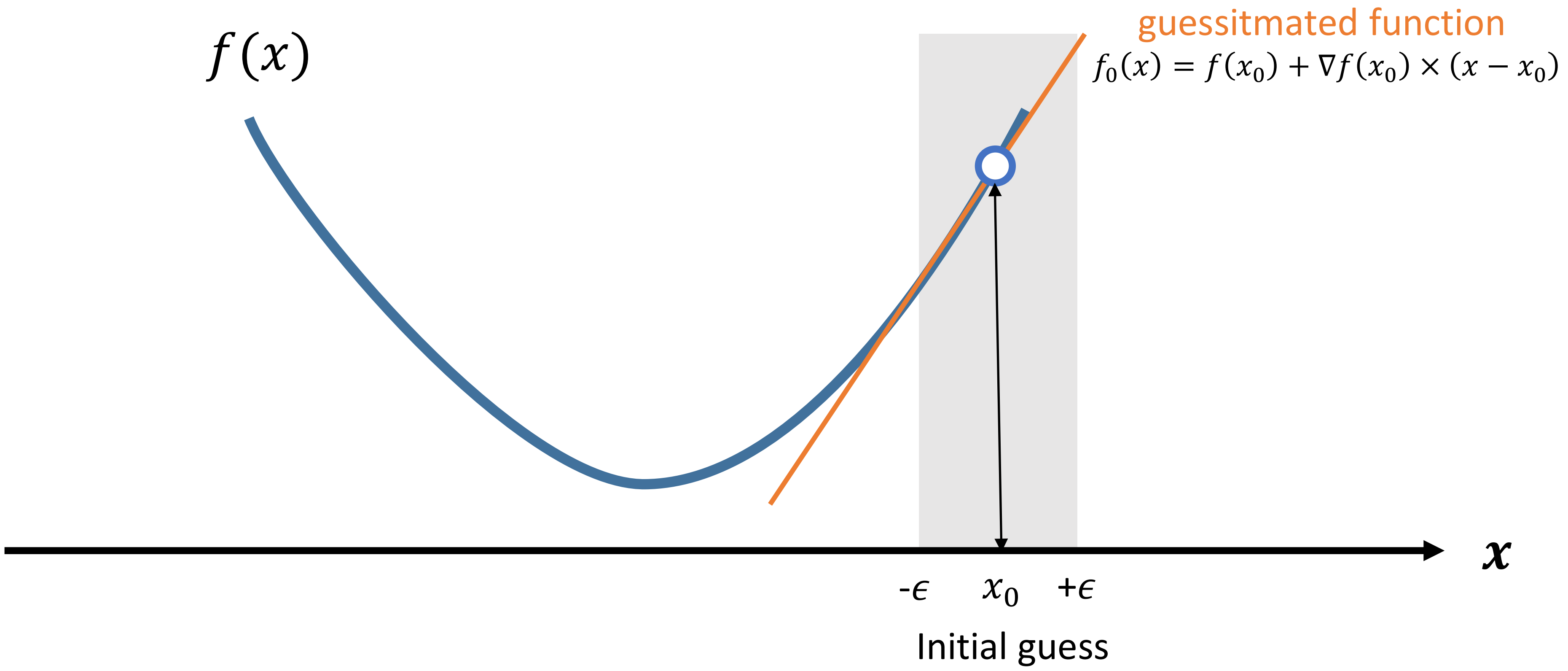
This approximation is valid when $|x-a| < \epsilon$ .i.e., local approximation



Taylor approximations of function

Gradient Descent Algorithm

***Utilize 1st order local approximation of function to find the next good point!**



- Since Taylor approximation becomes more inaccurate as we move far from x_0
- Trust the approximated function in locale only! i.e., Move our guess slightly
- Set next good point as $x_1 \leftarrow x_0 + \eta \nabla f(x_0)$. step size (learning rate) η is sufficiently small number.

*Utilizing 2nd order local approximation of function can be also used → Newton methods

Gradient Descent Algorithm

(Vanilla) Gradient Descent

1) Initialize an arbitrary $x = x_0 \in \mathbb{R}^n$, step size $\eta > 0$, tolerance $\epsilon > 0$
2) For $t = 0,1,\dots,$

   2-1) Compute $\nabla f(x_t) = \begin{bmatrix} \frac{\partial f}{\partial x^1}(x_t) \\ \vdots \\ \frac{\partial f}{\partial x^n}(x_t) \end{bmatrix}$

   2-2) Update $x_{t+1} \leftarrow x_t - \eta \nabla f(x_t)$

   2-3) break if $|x_{t+1} - x_t| < \epsilon$

Example of Gradient Descent Algorithm

$$\min_x f(x) = (x-3)^2 + 1$$

- Start with $x = x_0 = 5$, $\eta = 0.1$, $\epsilon = 0.01$
- For $t = 0$,

  Compute $\nabla f(x_0) = 2(x_0 - 3) = 4$

  Update $x = x_1 = x_0 - \eta \nabla f(x_0) = 5 - 0.1 * 4 = 4.6$

  Since $|5 - 4.6| > \epsilon$, continue for loop

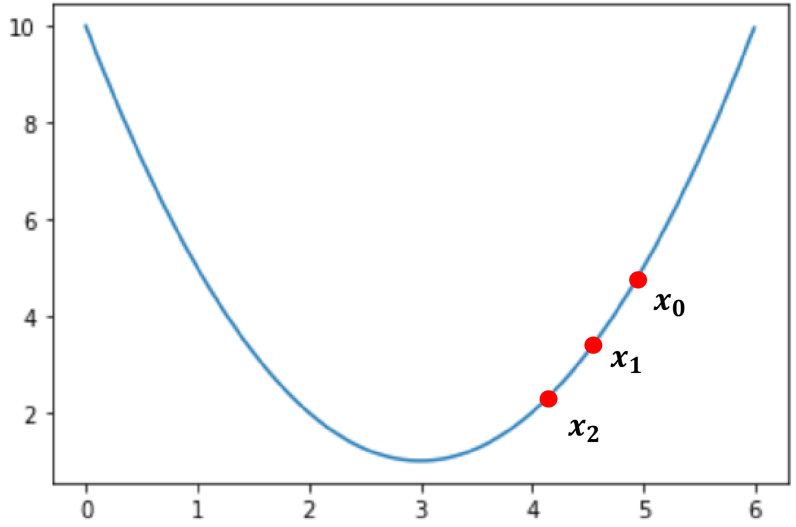
- For $t = 1$,

  Compute $\nabla f(x_1) = 2(x_1 - 3) = 3.2$

  Update $x = x_2 = x_1 - \eta \nabla f(x_1) = 4.6 - 0.1 * 3.2 = 4.28$

  Since $|4.6 - 4.28| > \epsilon$, continue for loop

- And so forth…



Limitations of Gradient Descent Algorithm

- If $f(x)$ is a convex function, GD find the optimal solution x^*
- If $f(x)$ is not convex, GD does not guarantee the optimality of converged solution.
- As a remedy, several GD variants are developed and works well in practice.
  - Stochastic Gradient Descent (SGD)
  - Momentum
  - AdaGrad
  - AdaDelta
  - RMSprop
  - Adam

Code Exercise!

github.com/song970407/WONIK-KAIST-Day1/settings

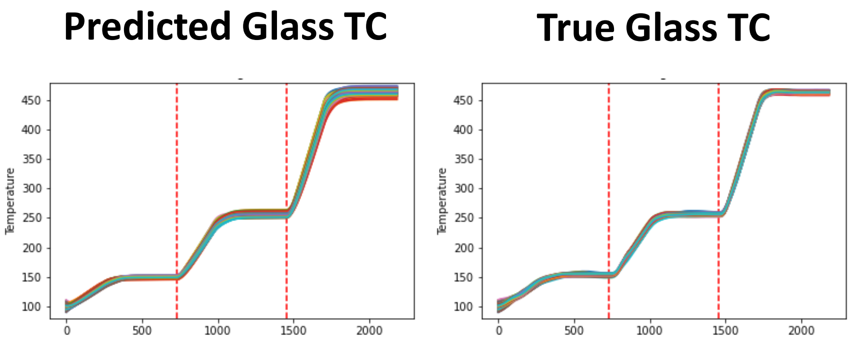
Real-World is Different…

- Lots of variables
  - In case of WONIK, 40 control inputs and 140 state variables
- More complicate true system
  - Linear model may be insufficient to approximate the true system
- Different data scale
  - Bad for training
- Introduce more techniques to apply in the real-world problem

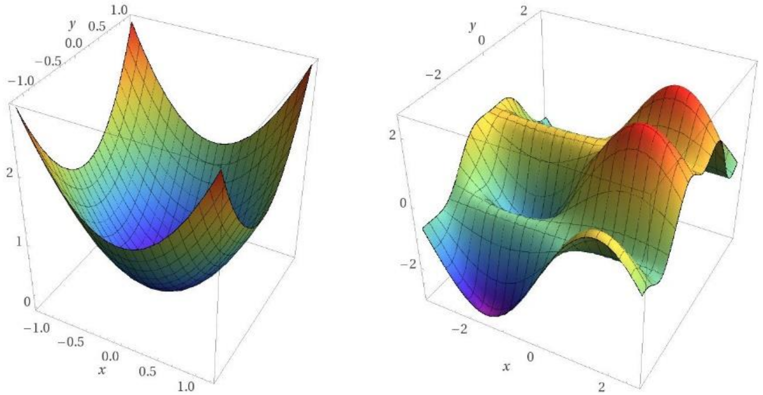
Step 2. Select Model Class

- Many different model classes
  - Linear, Deep neural network, Graph neural network, Gaussian process, etc

- Important: Balance between Model expressivity and Optimization solvability

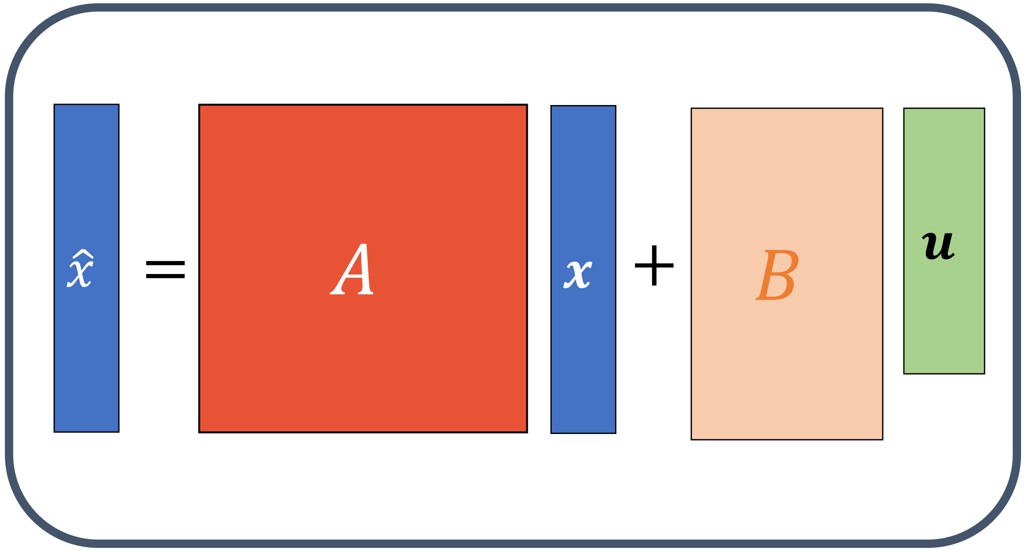


Model expressivity

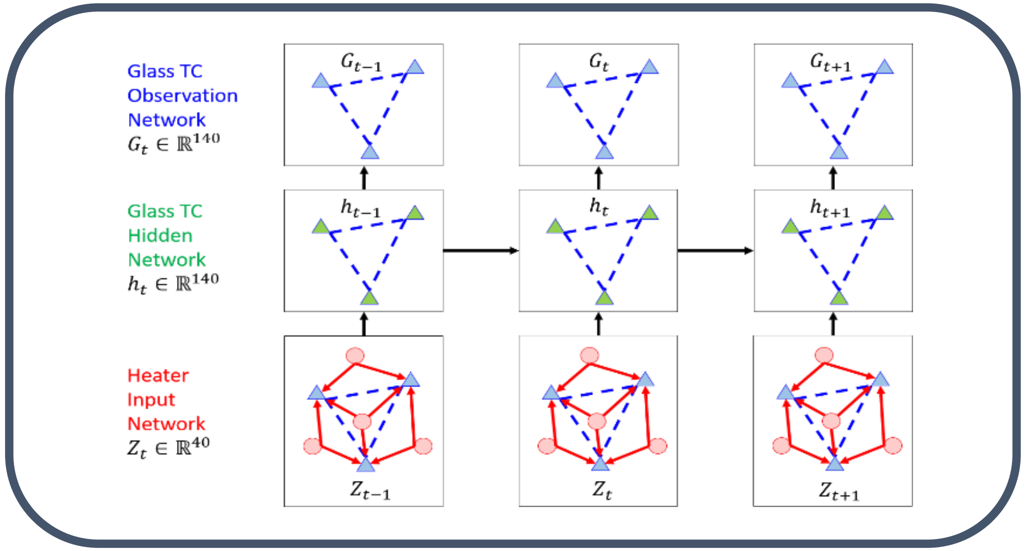


Optimization solvability

(Recap) Step 2. Select Model Class



Linear Model

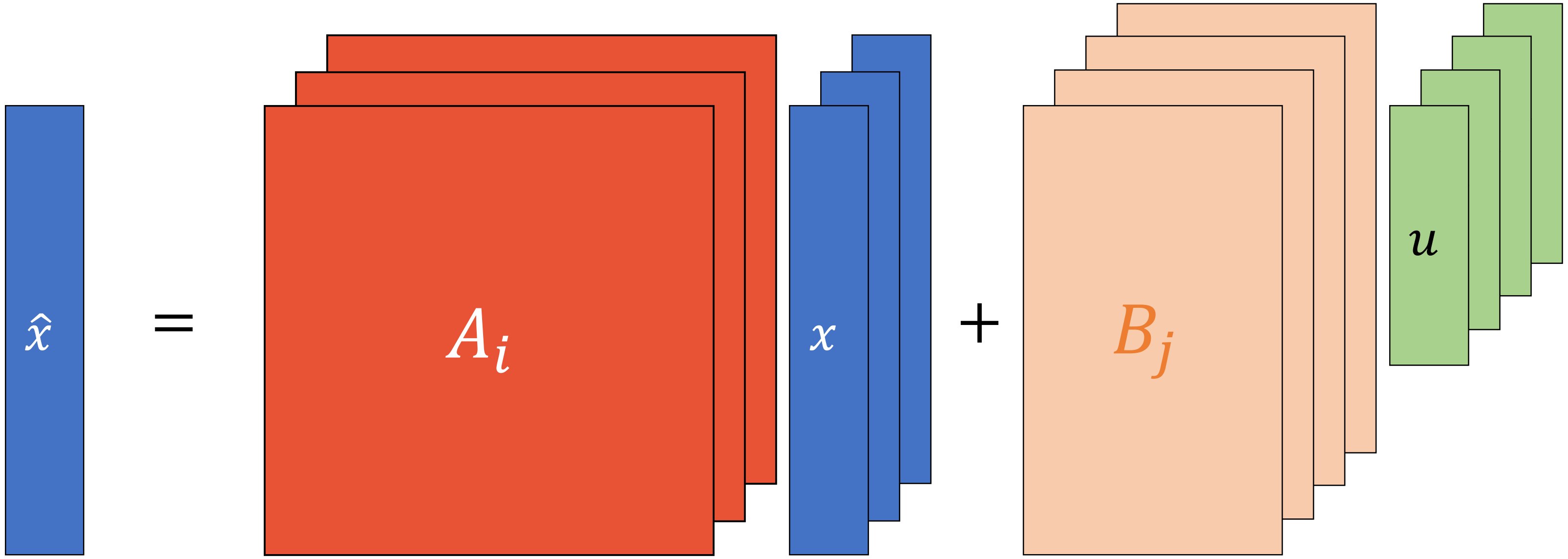


Non-Linear Model

	Linear Model	Non-Linear Model
Model Expressivity	↓	↑
Optimization Solvability	↑	↓
Example	Linear / Multistep Linear	NN, GNN

Multistep Linear Model

Predicted Glass TC | Past Glass TC | Past Workset



$$\hat{x}_{t+1} = \sum_{i=0}^{I-1} A_i x_{t-i} + \sum_{j=0}^{J-1} B_j u_{t-j}$$

$$A_i \in \mathbb{R}^{140\times 140}$$
$$B_j \in \mathbb{R}^{140\times 40}$$

- 현재의 Glass TC와 Work set 뿐만 아니라 과거 값도 사용 → Furnace의 관성 모델링 가능
- **Model expressivity** 와 **Optimization solvability** 를 적절히 밸런싱

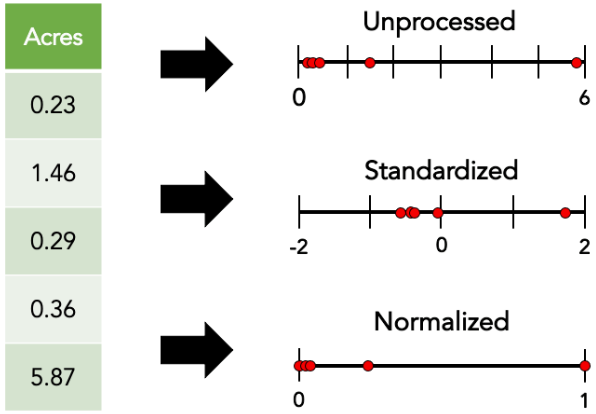
Step 3. Prepare the Data (a.k.a. Data Preprocessing)

- Make sure that data is ready to be used

- How?
  - **Missing value**, outlier
  - Categorical variable
  - Variable reduction
  - **Feature scaling**

Table (1): A hypothetical example of numerical data

	Column 1	Column 2	Column 3
Row 1	26	22	12
Row 2	Green	8	7
Row 3	84	60	-



Hyperparameter Tuning

- Model parameter vs Hyperparameter
  - Model parameter: can be estimated from the data
  - Hyperparameter: cannot be estimated from the data

- Select the best hyperparameter by comparing the model performance
  - The devil is here…

- Learning rate, epoch, state/action order, etc

Trick 1: Multistep Prediction

- Not only predict $\hat{x}_{t+1}$ but also $\hat{x}_{t+2}, \hat{x}_{t+3}, \ldots$
  - Important: Use x_t and a sequence of future control input $(u_t, u_{t+1}, \ldots)$
  - $\hat{x}_{t+1} = \hat{f}(x_t, u_t; \theta), \hat{x}_{t+2} = \hat{f}(\hat{x}_{t+1}, u_{t+1}; \theta), \ldots$
  - Much harder than one step prediction

- Furnace does not change dramatically within just 5 seconds, which means $x_t \approx x_{t+1}$
  - The learned model can be a trivial model, which is $\hat{f}(x_t, u_t) = x_t$
  - By minimizing MSE between $(x_{t+1}, x_{t+2}, \ldots)$ and $(\hat{x}_{t+1}, \hat{x}_{t+2}, \ldots)$, we can obtain more reasonable model

Trick 2: Good Model Parameter Initialization

- Initial model parameter is super important
  - Stable & fast training

- Naïve idea: Just return the current state $\hat{x}_{t+1} = \hat{f}(x_t, u_t; \theta) = x_t$
  - Initialize A, B as $A_0 = I$, otherwise $A_i = B_j = 0$.
  - $\hat{x}_{t+1} = \sum_{i=0}^{I-1} A_i x_{t-i} + \sum_{j=0}^{J-1} B_j u_{t-j} = A_0 x_t + 0 = x_t$

Dive into the Real Code