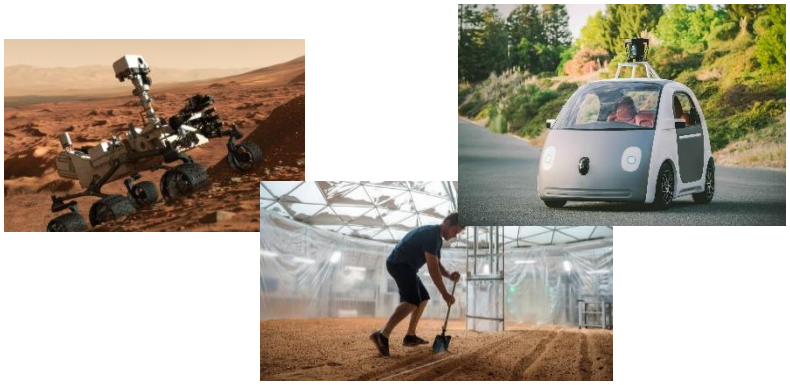
L0. Introduction

Motivation

- Machine Learning
- Artificial Intelligence
- Optimization
- Optimum Control
- Planning
- Markov Decision Process
- Influential Diagram
- Decision Tree
- Dynamic Control
- Game Theory
- Search
- Stochastic Programming
- Reinforcement Learning
- Bandit problem

⋮

Engineering is all about decision makings



What are the differences in these decision-making strategies

What are the common aspects in these decision-making strategies?

Data-driven decision making and control in engineering domain

Sensors (Hardware)
- Embedded computing
- Sensor-Actuator Network
- Wireless communication

Data analytics
- Feature extraction
- Machine learning
- Data mining

Decision making
- Mathematical programming
- Stochastic control
- Sequential decision making

Feedback

$x \rightarrow f(x; \theta_j) \rightarrow y$

input — output

θ_j

Environment

Target system

$\mathcal{D} = \{x^{1:n}, y^{1:n}, \theta^{1:n}\}$ — Data acquisition

$y = f(x; \theta)$ — Learning

$x^* | \theta = \underset{x}{\operatorname{argmax}} f(x; \theta)$ — Making decision

Motivation

engineers' creativity depends on the diversity of tools that he or she has

- Diversify tools for decision making
- Understand the usages of your tools
- Sharpen your tools
- Organize your tools

Core Ideas

What type of decision making framework will be used?

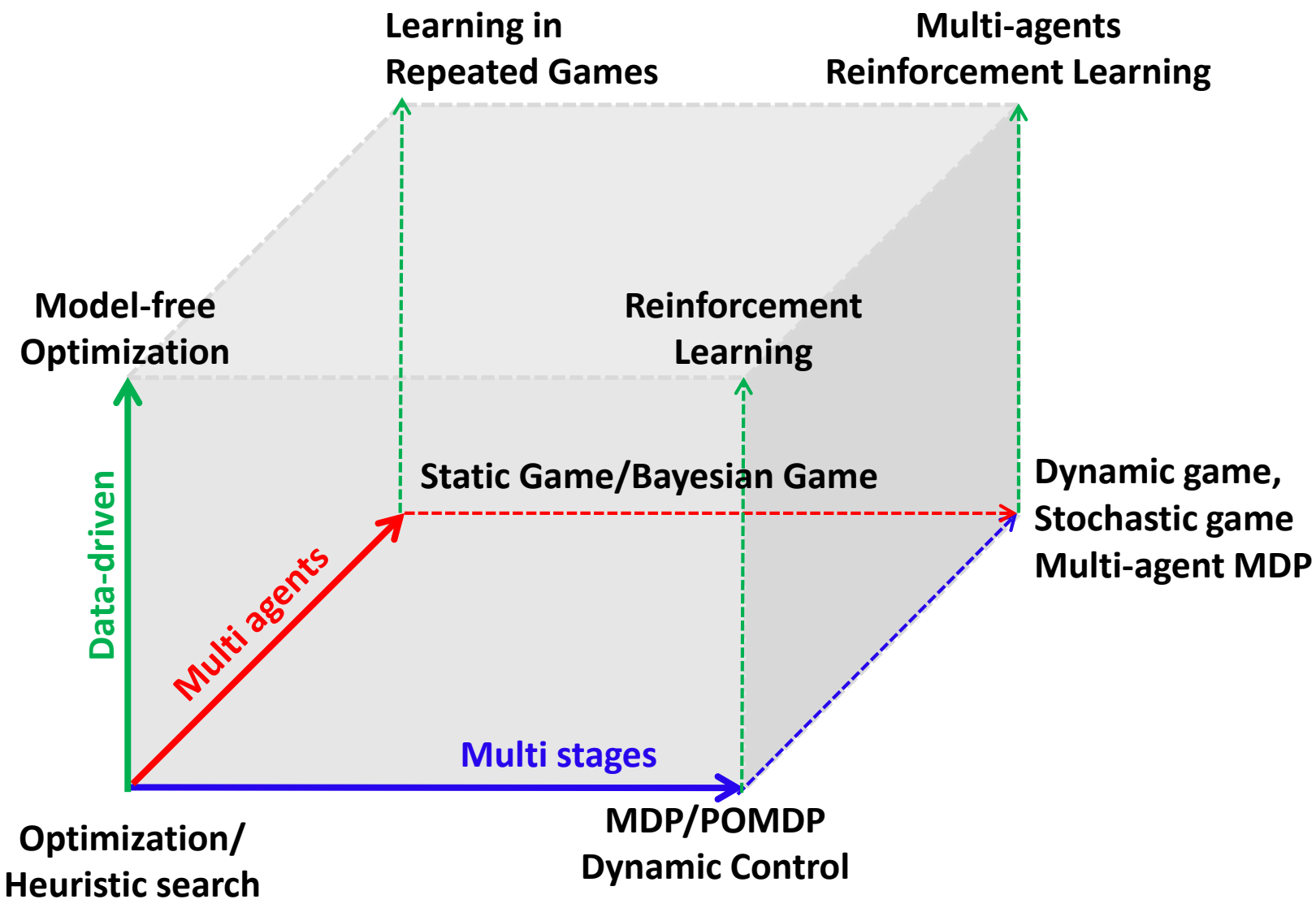
- Single stage or multi stages
- Single decision maker or many decision makers
- Model based or model-free

"Decision makings under uncertainties"

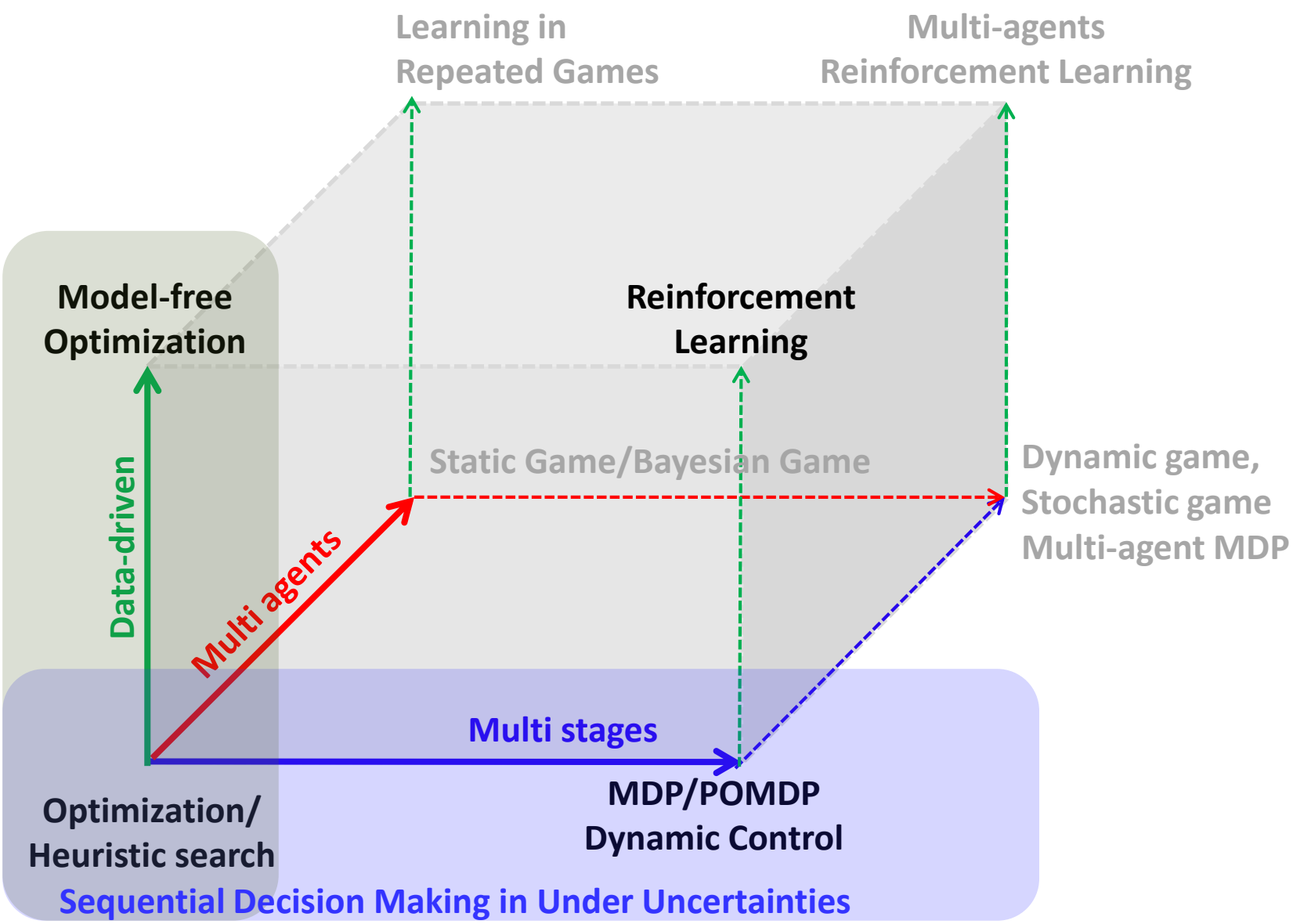
How to model uncertainties?

- **Epistemic Uncertainty** (systemic uncertainty) :
  Uncertainty arising through lack of knowledge
  - Model uncertainty
  - State uncertainty
- **Aleatoric uncertainty** (statistical uncertainty):
  Uncertainty arising through an underlying stochastic system

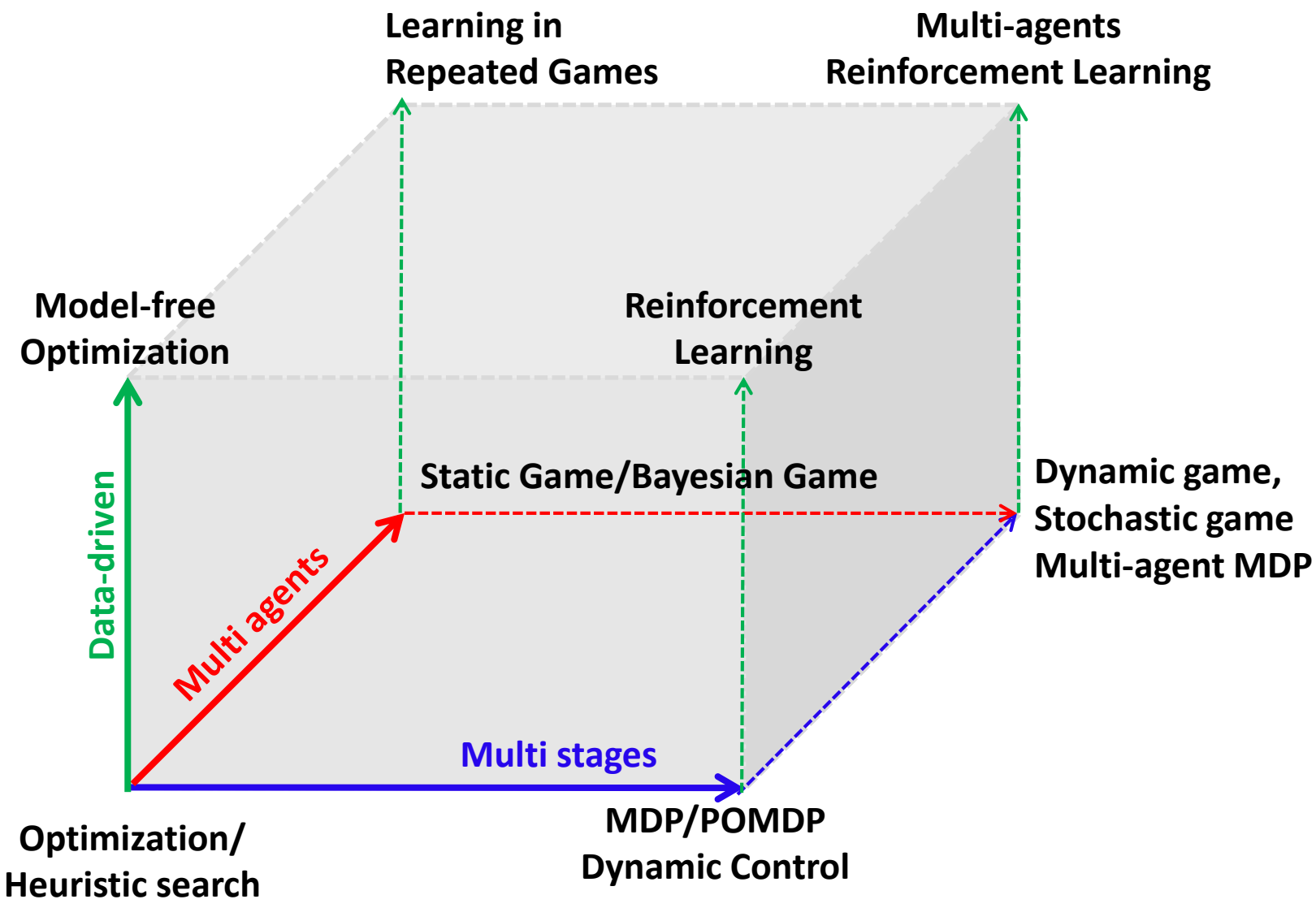
Classified decision making methods



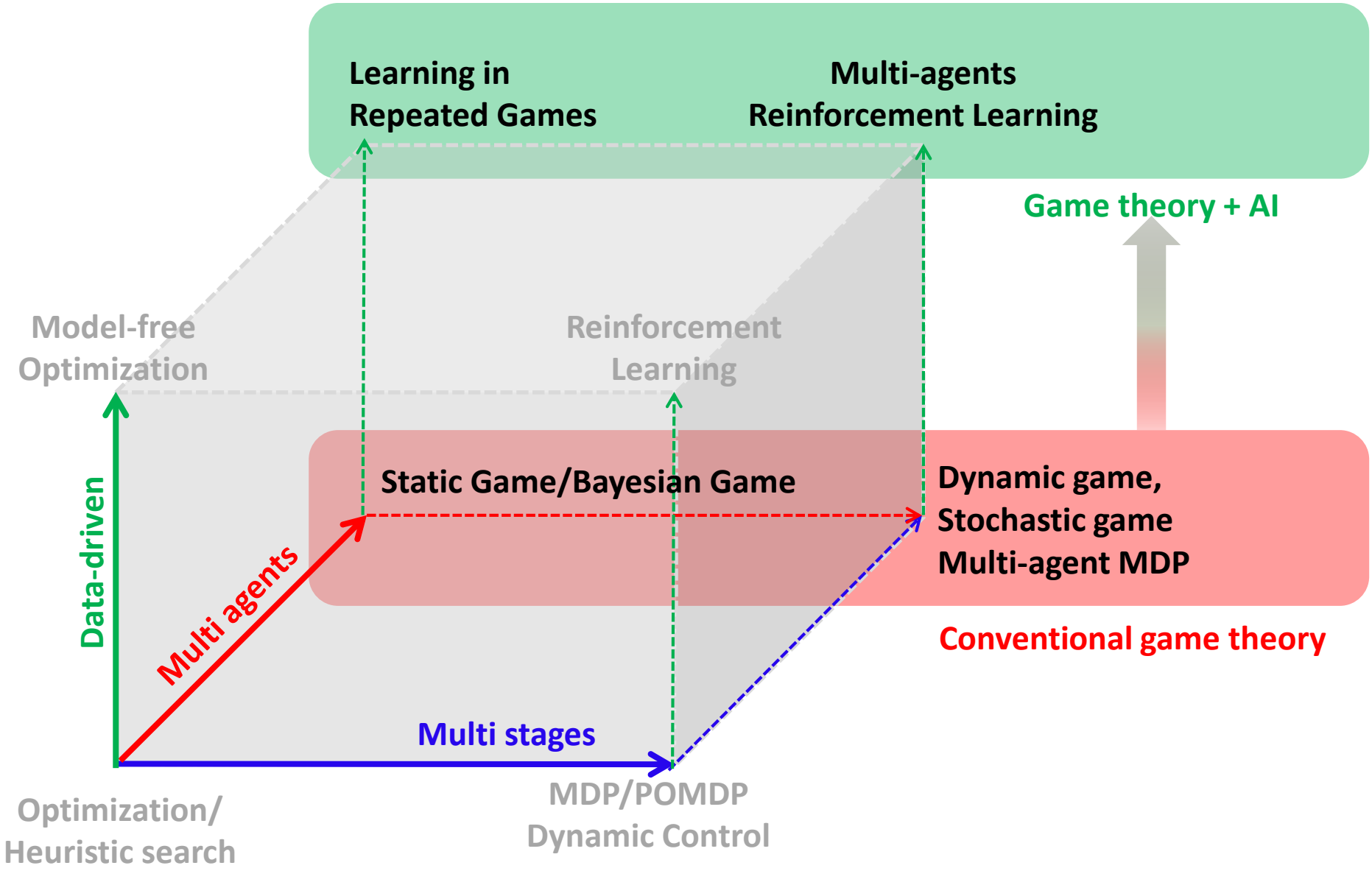
Classified decision making methods



Classified decision making methods



Classified decision making methods

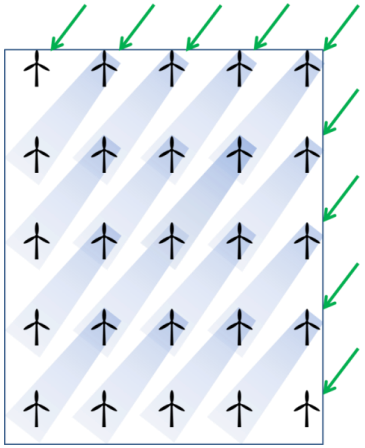


Problem solving

Real-world task

Modeling

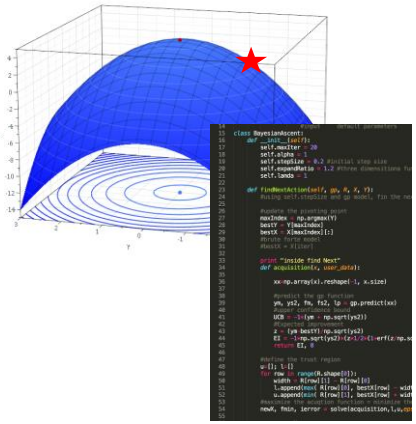
Formal task (model)



$$\underset{\boldsymbol{x}}{\text{maximize}} \sum_{i=1}^{N} P_i(\boldsymbol{x}; \theta, U)$$

Solving

Algorithm (program)



Validation

Are we building the right model?

Verification

Does the algorithm capture all the essential aspects of the model?

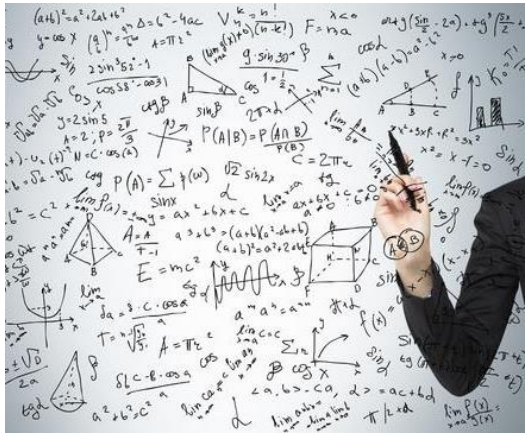
Is this solution is good for the target system?

Data can help model more realistically and derive more accurate solution!

Key elements of this course

Data analytics

- Bayesian Statistics
- Machine learning
- Bayesian Network



Modeling

- Optimization
- Markov Decision Process
- Game Theory

Decision Making

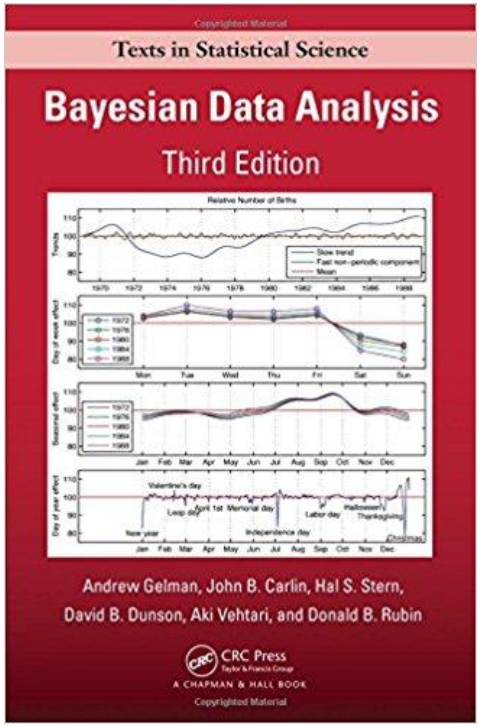
- Mathematical Programming
- Dynamic Programming
- Reinforcement Learning

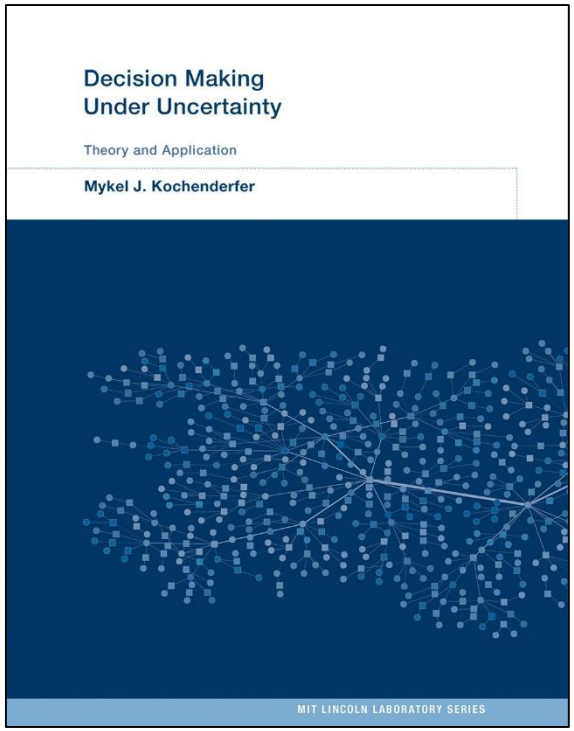
Course Objectives

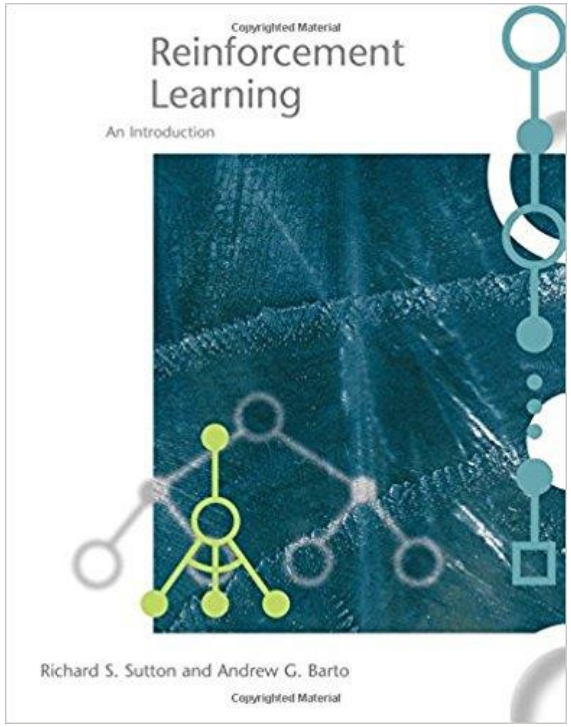
Upon successful completion of the course, you are able to

- *understand* various mathematical models describing decision making problems.
- *formulate* real-world decision making problems in a mathematical form.
- *implement* key algorithms and approaches to solving various decision making problems.
- *Interpret* the results of decision-making problems.

Books (optional)

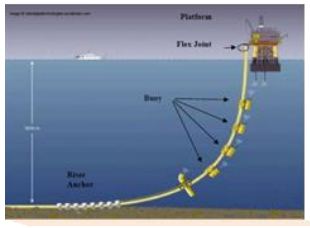






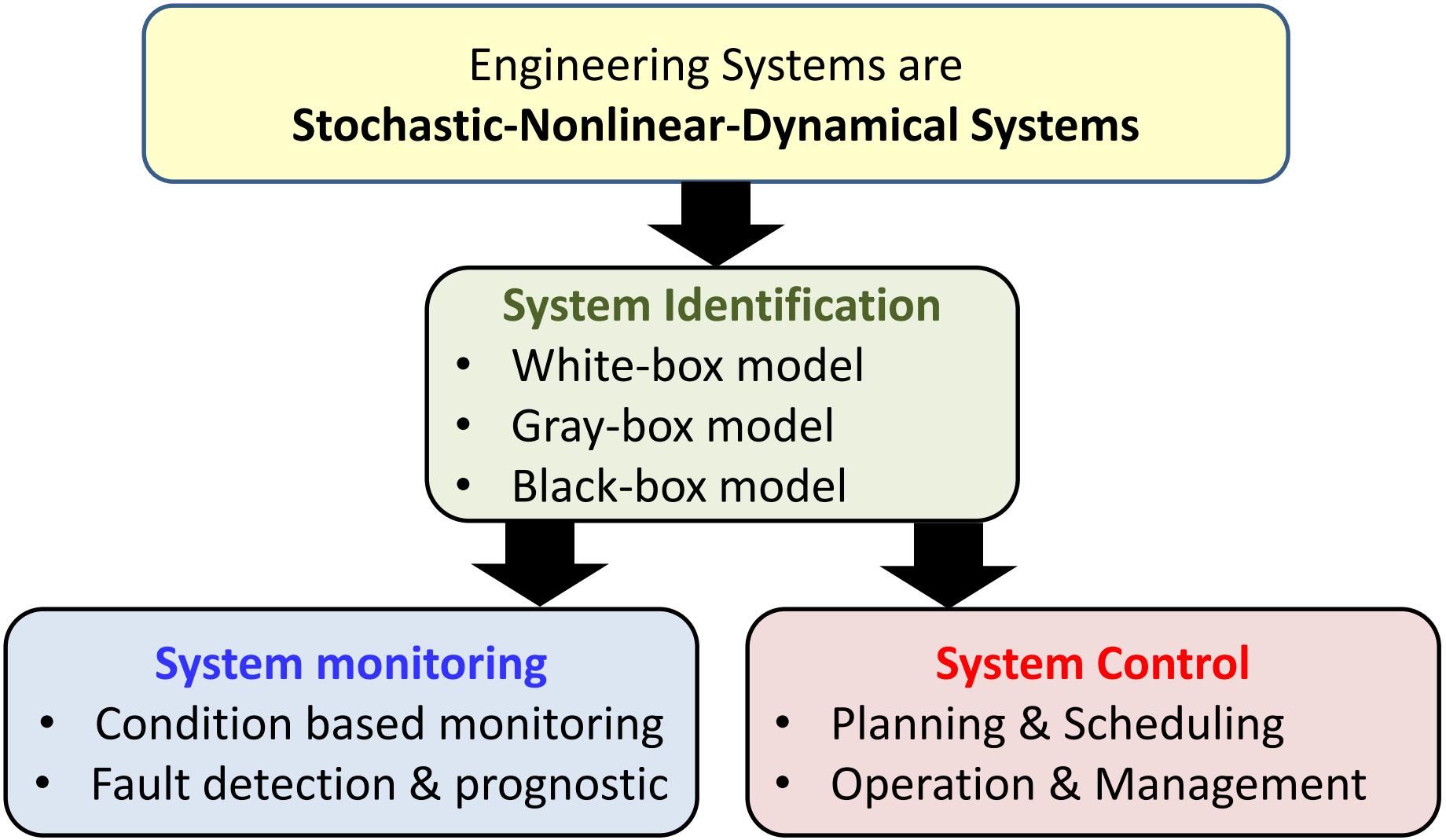
Why machine learning is important in engineering fields

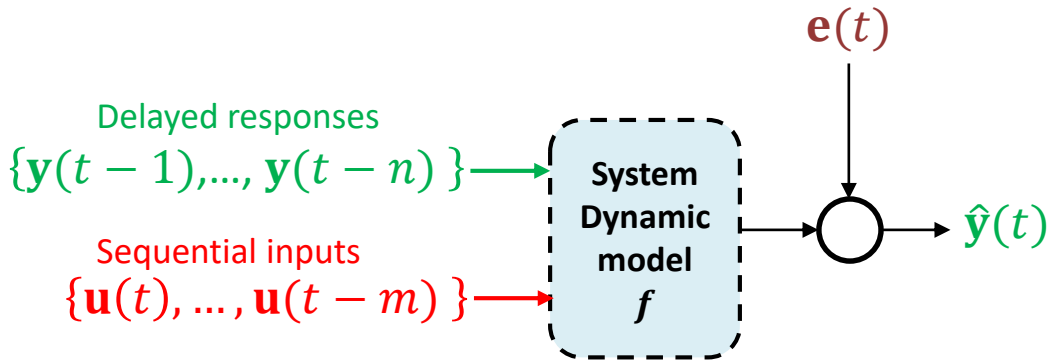
Engineering Systems are
Stochastic-Nonlinear-Dynamical Systems





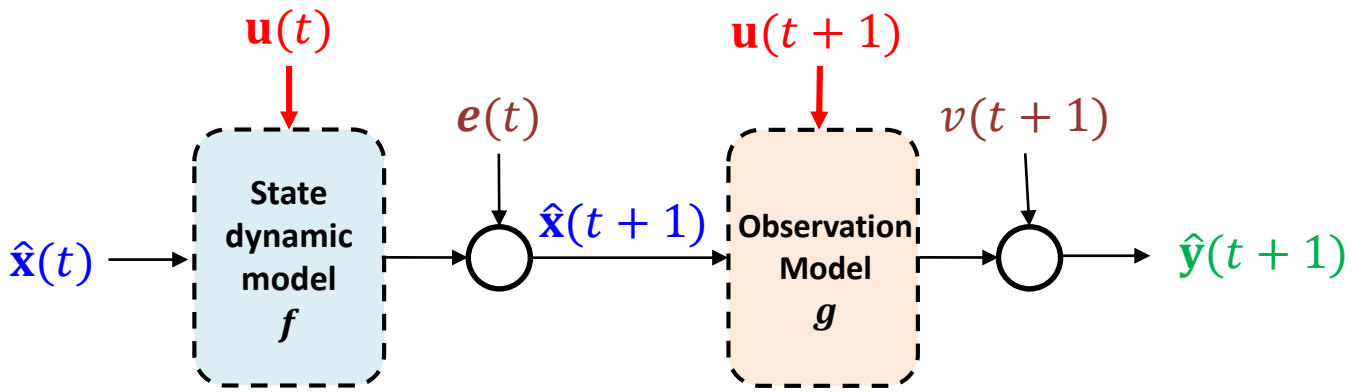

Why machine learning is important in engineering fields





NARX (nonlinear autoregressive model with exogenous input)

$$\hat{\mathbf{y}}(t) = f(\mathbf{y}(t-1), \ldots, \mathbf{y}(t-n), \mathbf{u}(t), \ldots, \mathbf{u}(t-m)) + \boldsymbol{e}(t)$$



State-space model

- State dynamic model

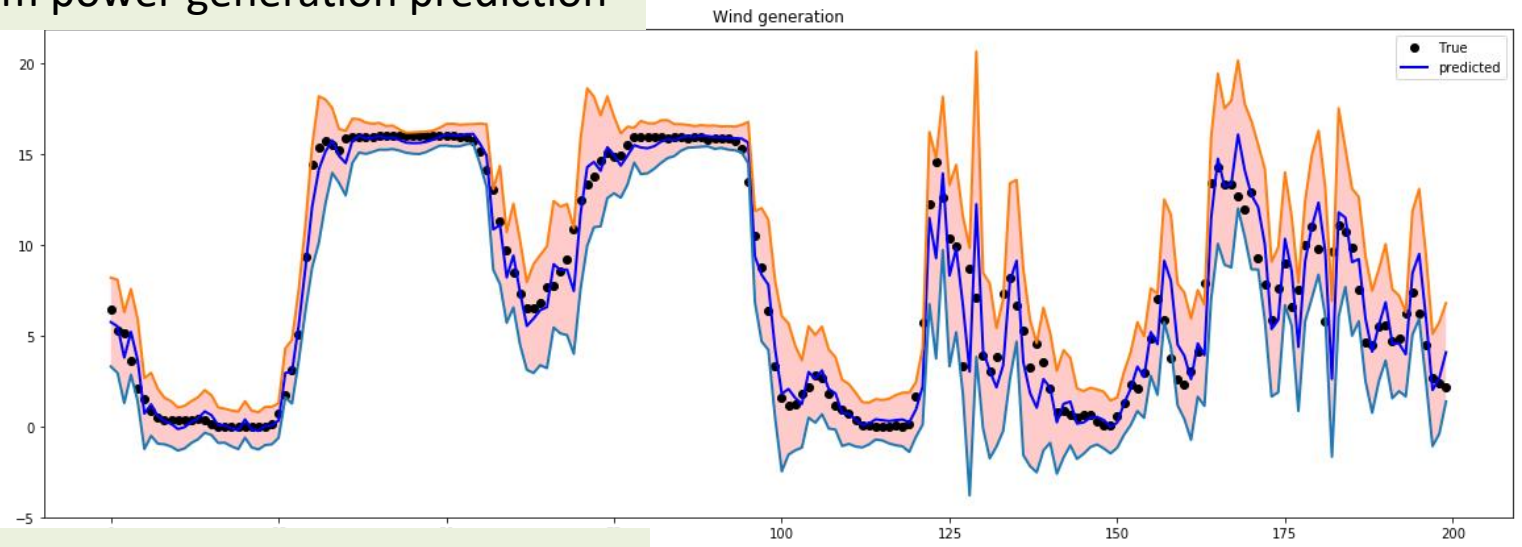
$$\hat{\mathbf{x}}(t+1) = f\big(\mathbf{x}(t), \mathbf{u}(t)\big) + \boldsymbol{e}(t)$$

- Observation model

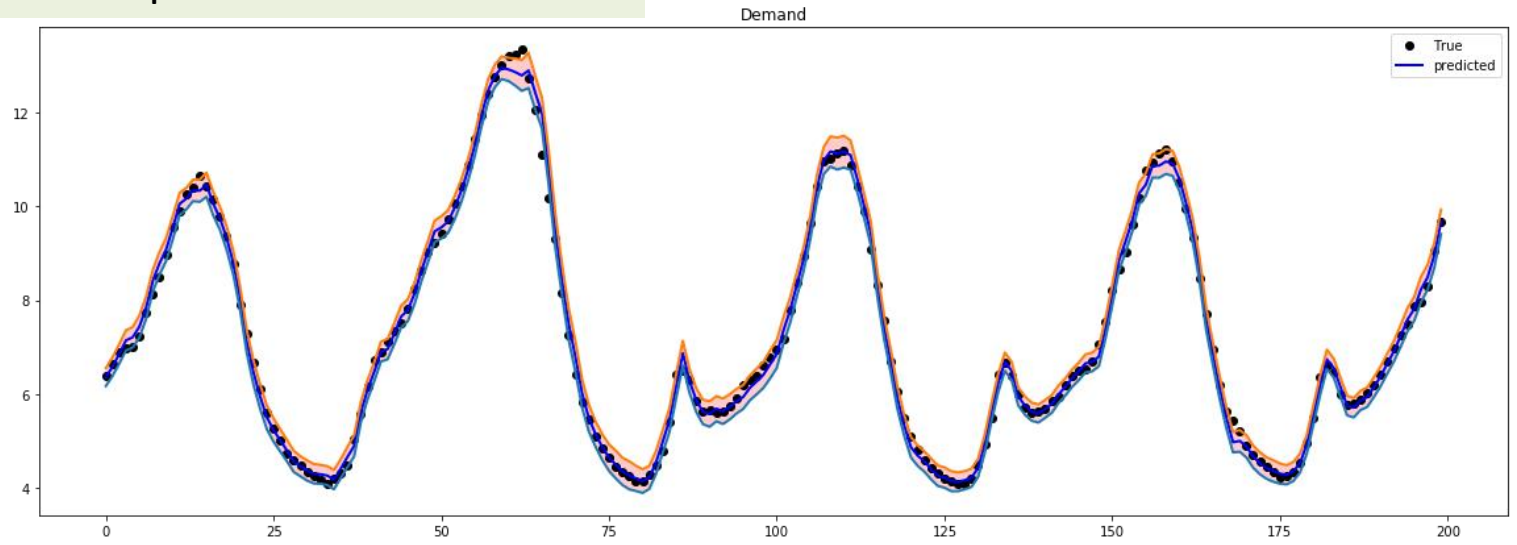
$$\hat{\mathbf{y}}(t+1) = g\big(\mathbf{x}(t+1), \mathbf{u}(t+1)\big) + \boldsymbol{v}(t+1)$$

Systems Identification

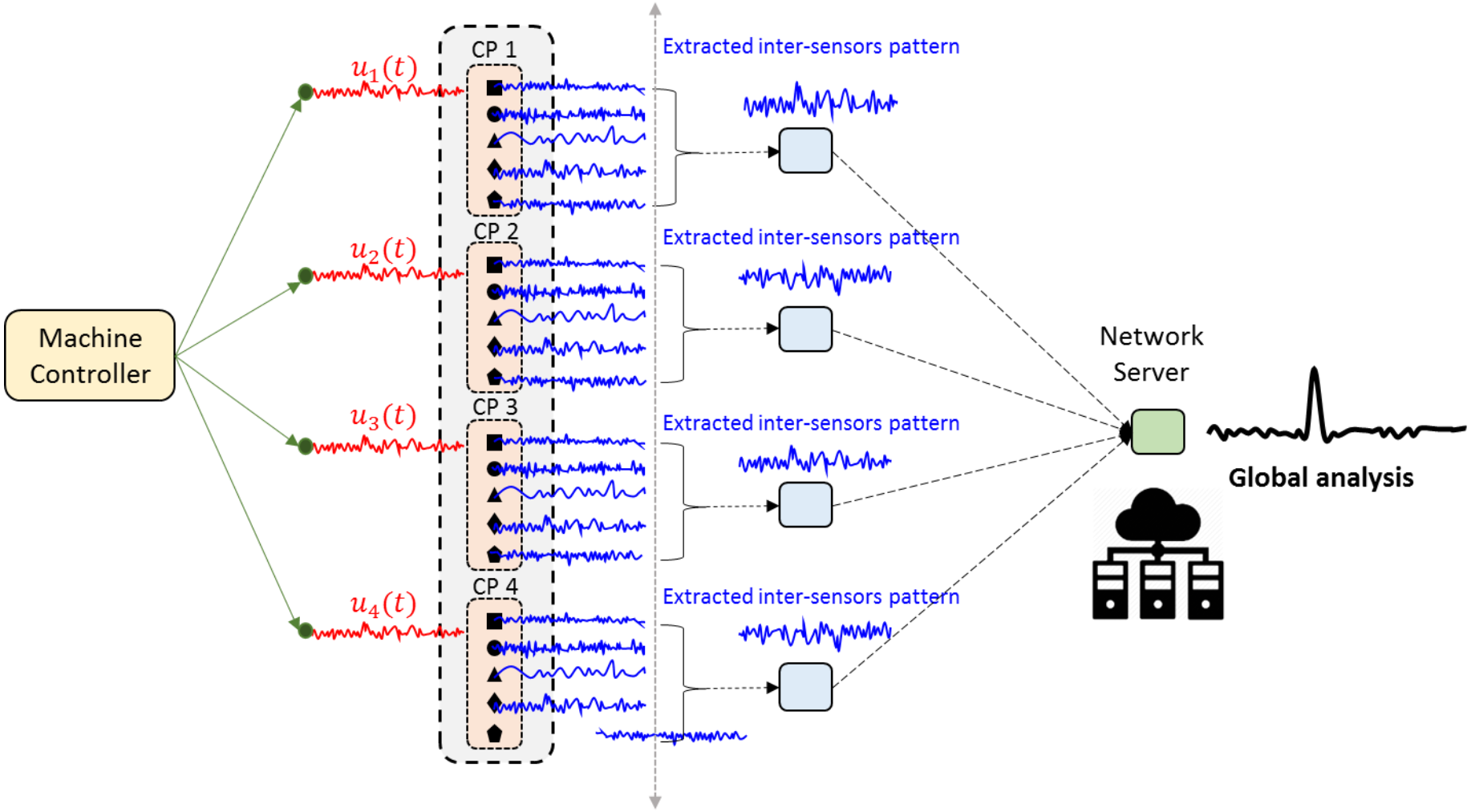
Wind farm power generation prediction



Energy demand prediction



Systems Monitoring



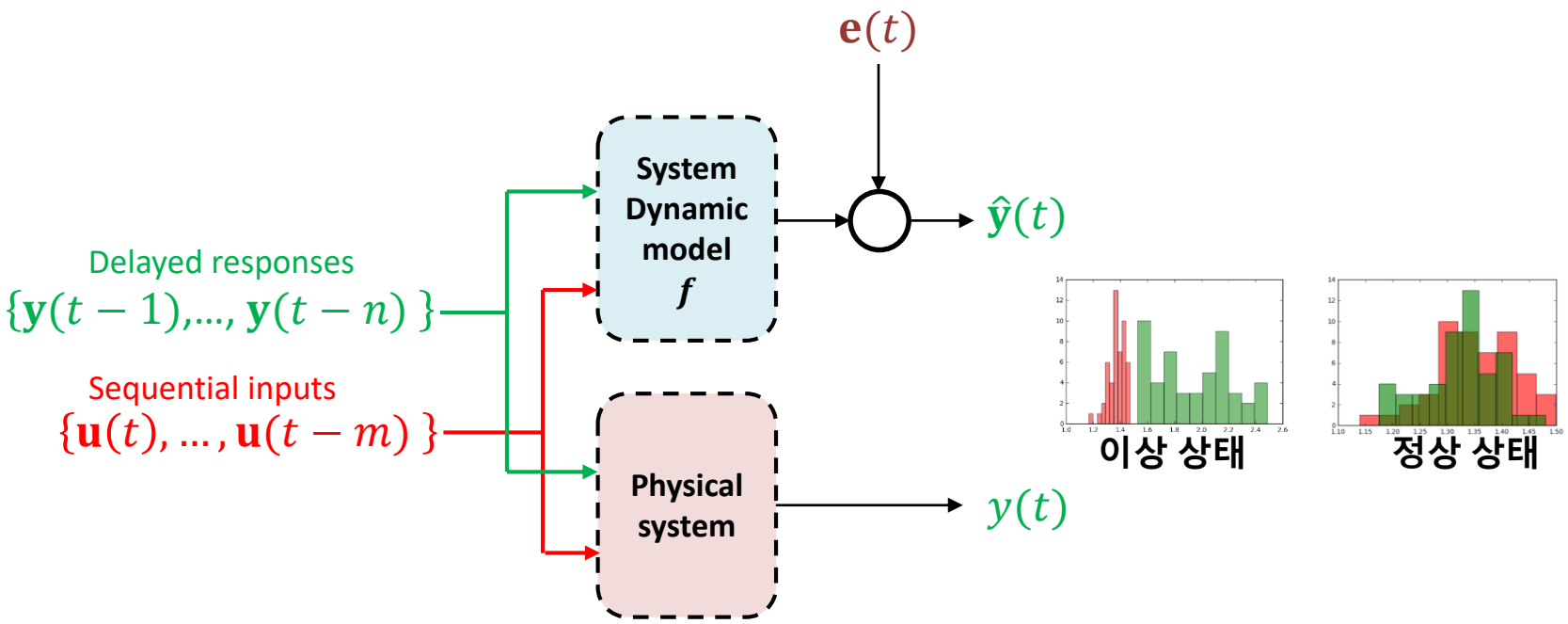
IoT Hierarchical *Data processing* :

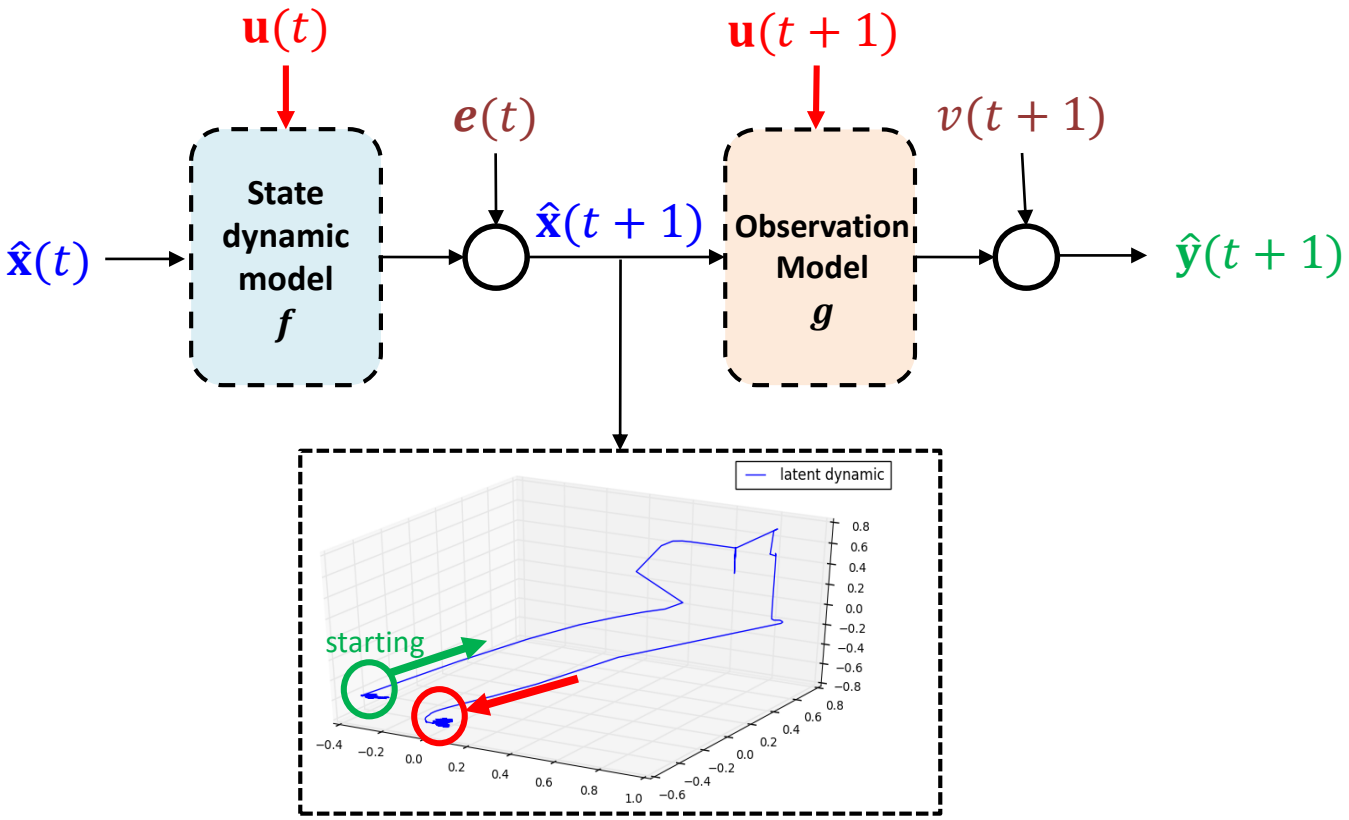
Local monitoring sensor → Component monitoring module → Machine monitoring module

AI Hierarchical *machine intelligence* :

Change point detection → Anomaly detection → Machine diagnosis & prognosis

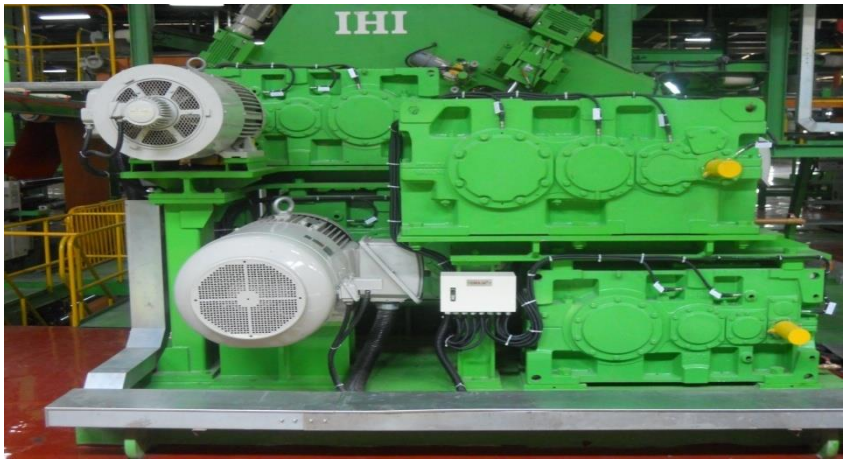
Systems Monitoring

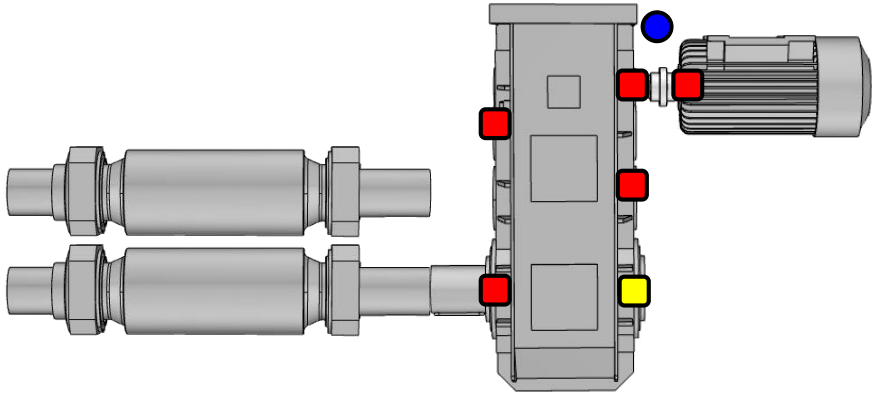


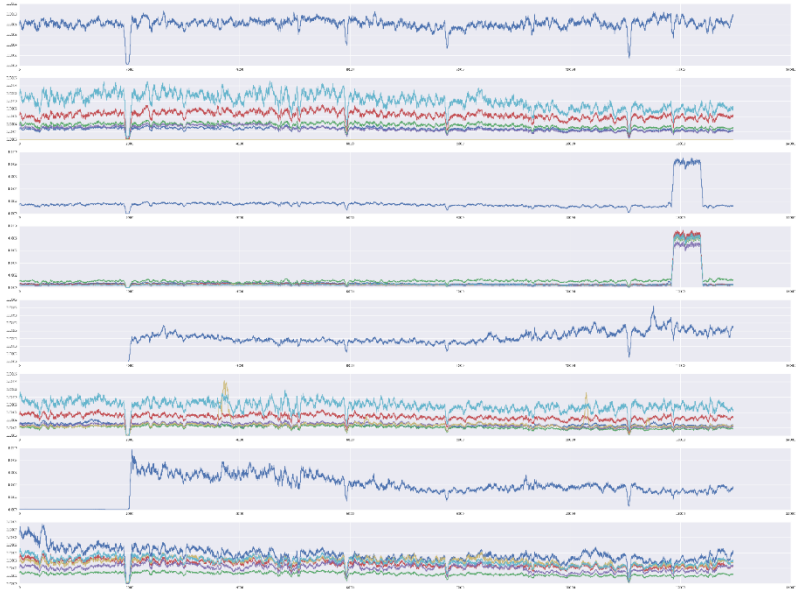


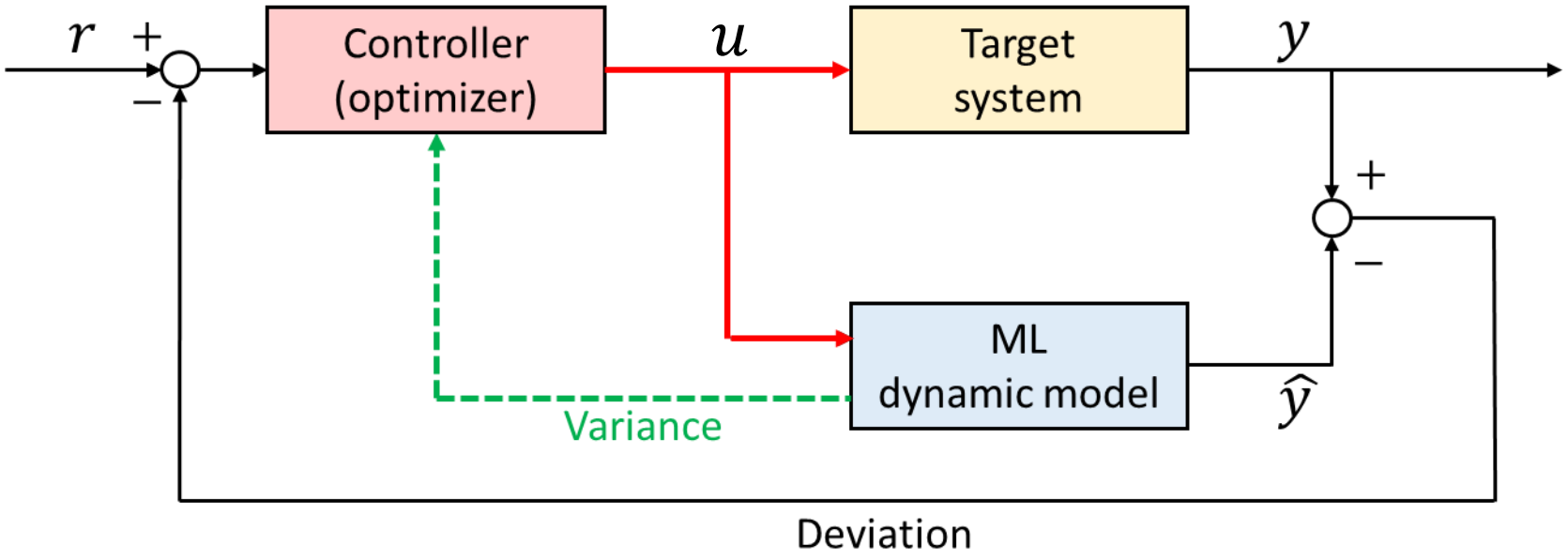
The trajectory of latent state vector

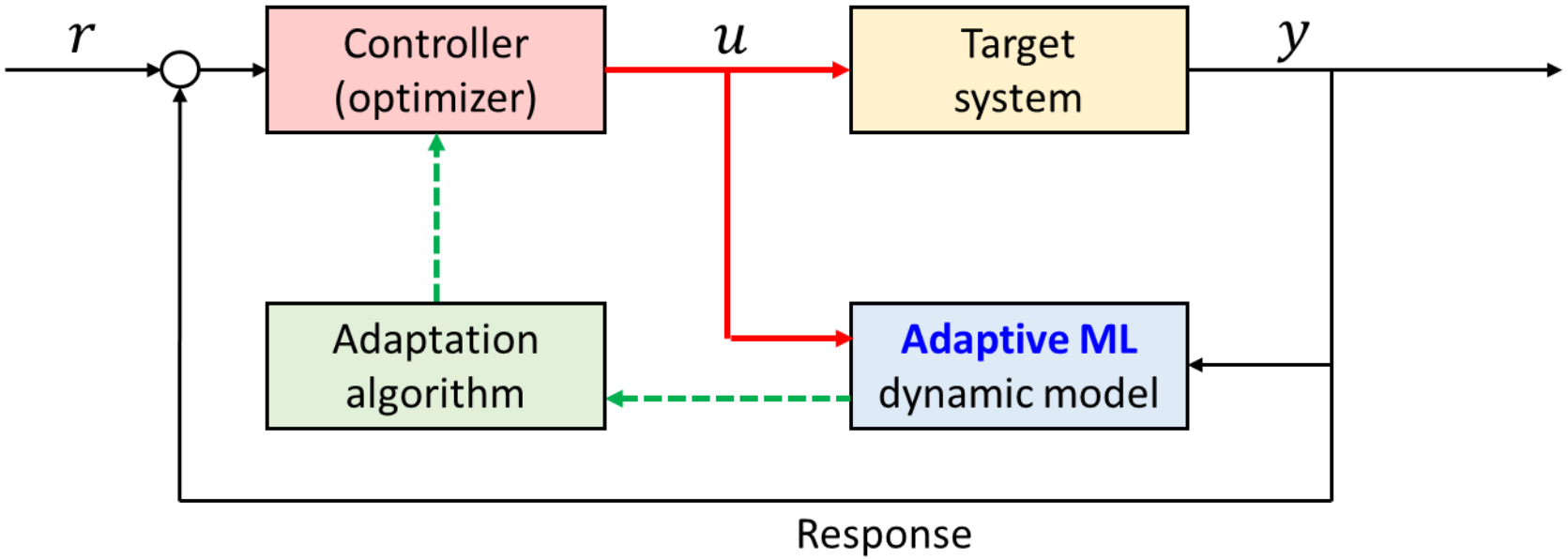
Systems Monitoring



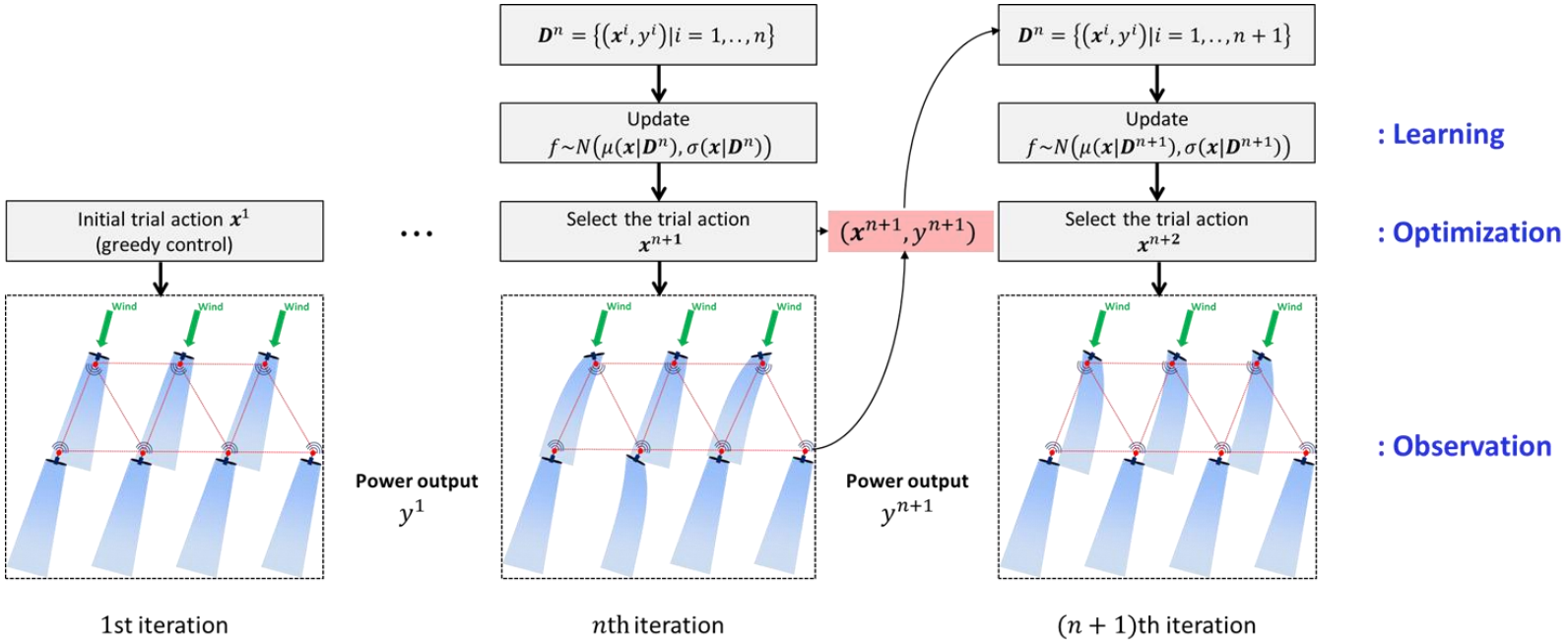


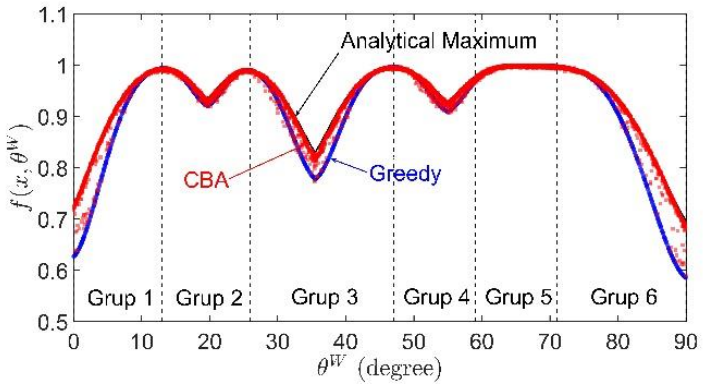


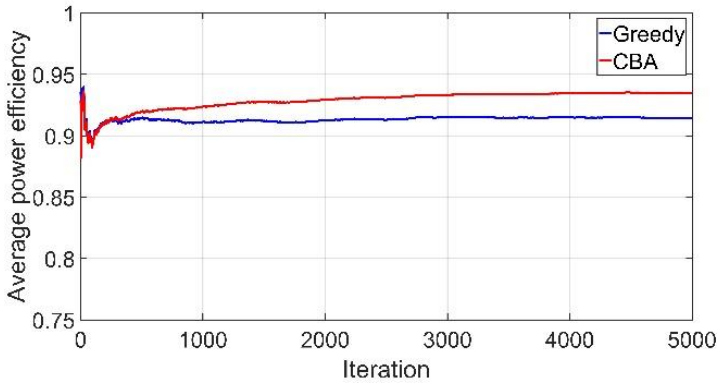

r
+
−
Controller
(optimizer)
u
Target
system
y
+
−
ML
dynamic model
ŷ
Variance
Deviation


r
Controller
(optimizer)
u
Target
system
y
Adaptation
algorithm
Adaptive ML
dynamic model
Response

Systems Control







Systems Control

에너지 생산

신재생 에너지 시스템의 생산 가능 전력 예측과 제어

에너지 분배

- 전압 및 송전선 전송 용량 제어
- 전력품질 개선 가능
- 화석연료에 대한 의존도 축소

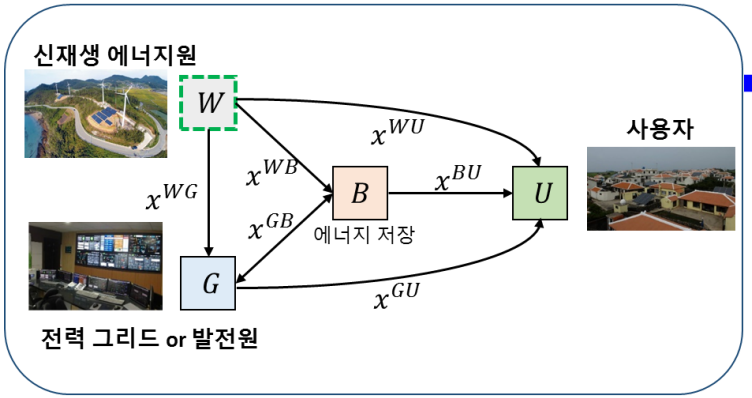
에너지 소비

빅데이터를 이용한 수용가 전력 수요 예측 및 제어 기법

에너지 분배를 통한
에너지 생산과 소비의 불균형 감소

에너지 생산 정보

에너지 수요 정보



$$\min_{x_1,\dots,x_T} E\left[\sum_{t=1}^{T} P_t^G(x_t^{GB}+x_t^{GU}) - P_t^W x_t^{WG}\right]$$

P_t^W: 재생 에너지 가격
P_t^G: 에너지 생산가격
U_t: 에너지 수요
E_t: 에너지 생산량

$$\text{s.t.}\quad x_t^{WG}+x_t^{WB}+x_t^{GU}\le E_t$$
$$x_t^{BU}+x_t^{GU}+x_t^{WU}\le U_t$$
$$B_{t+1}=\rho^T B_t+\rho^C(x_t^{WG}+x_t^{WB})-\rho^D x_t^{BU}$$
$$B_t\le 0.9C_B$$

For $t=1,\dots,T-1$

- Stochastic Programming
- Model Predictive Control
- Dynamic Programming
- Reinforcement Learning

Course schedule

1. Bayesian Modeling and Inference (3 weeks)

- Probability distributions
- Prior, Likelihood, and Posterior
- Conjugate models
- Hierarchical Models
- Elements of Computational Bayesian Statistics

2. Single-agent, single-stage decision-making (3 weeks)

- Bayesian regression
- Bayesian classification
- Bayesian Network
- Influential Diagram

3. Multi-stages decision-making (6 weeks)

- Bandit problem
- Bayesian Optimization
- Markov Decision Process (MDP)
- Dynamic Programming
- Reinforcement Learning
  - Monte Carlo Methods
  - Temporal Difference Methods

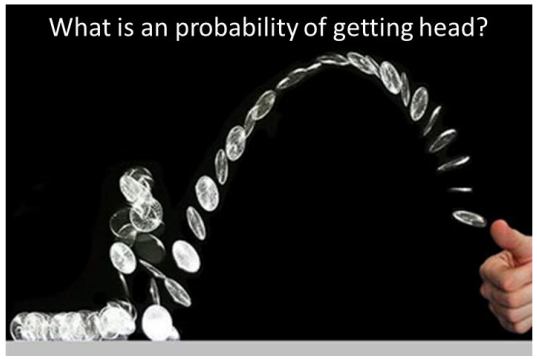
4. Multi-agents decision-making (3 weeks)

- Basics of game theory
- Bayesian Game (with uncertainty about other agents)

Main subject

1. Bayesian Modeling and Inference (3 weeks)

- Probability distributions
- **Prior, Likelihood, and Posterior**
- Conjugate models
- Hierarchical Models
- Elements of Computational Bayesian Statistics

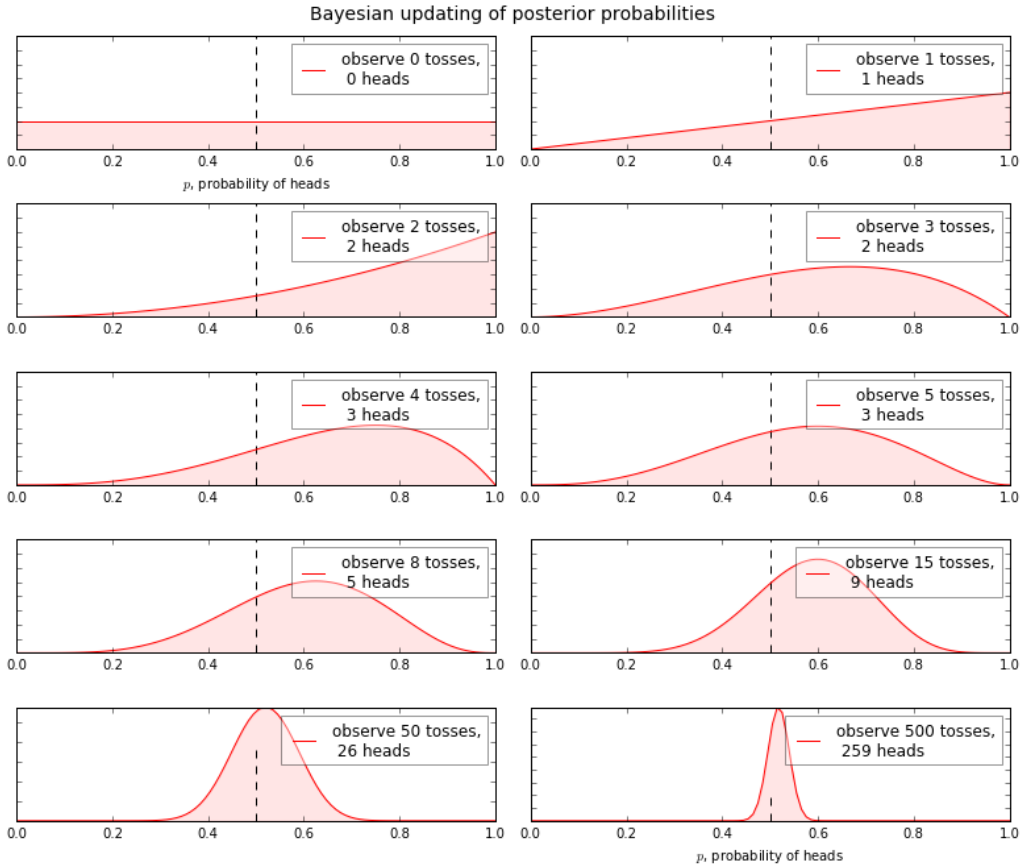


$$p(\theta|y) = \frac{p(y|\theta)p(\theta)}{p(y)}$$

$$= \frac{p(y|\theta)p(\theta)}{\int_\theta p(y|\theta)p(\theta)d\theta}$$

$$\propto p(y|\theta)p(\theta)$$

Posterior ∝ Likelihood X Prior

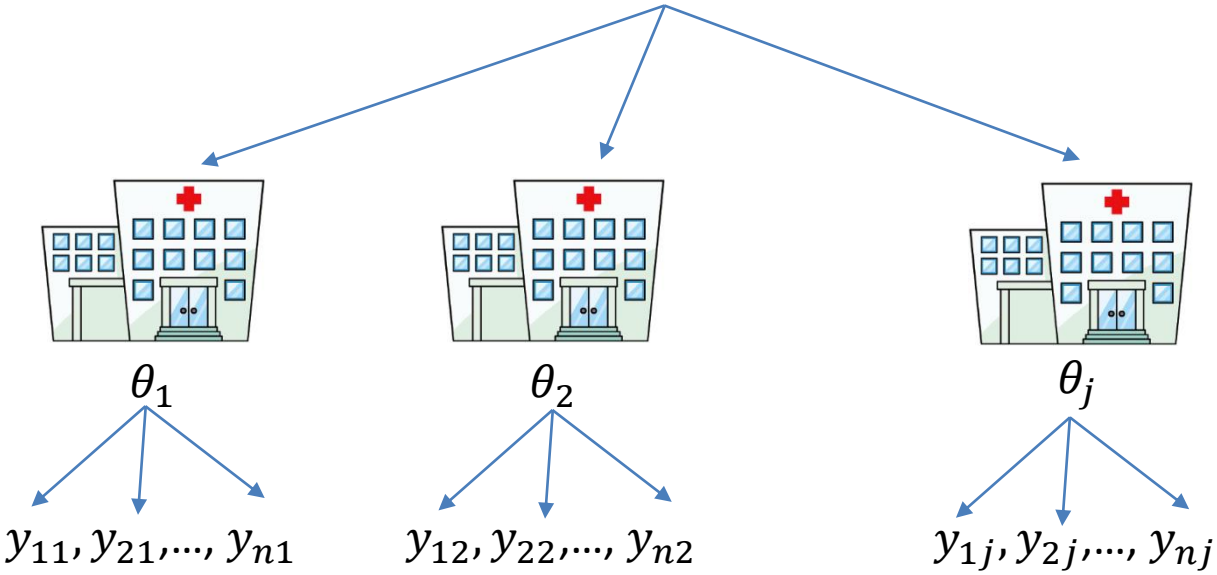


Main subject

1. Bayesian Modeling and Inference (3 weeks)

- Probability distributions
- Prior, Likelihood, and Posterior
- Conjugate models
- **Hierarchical Models**

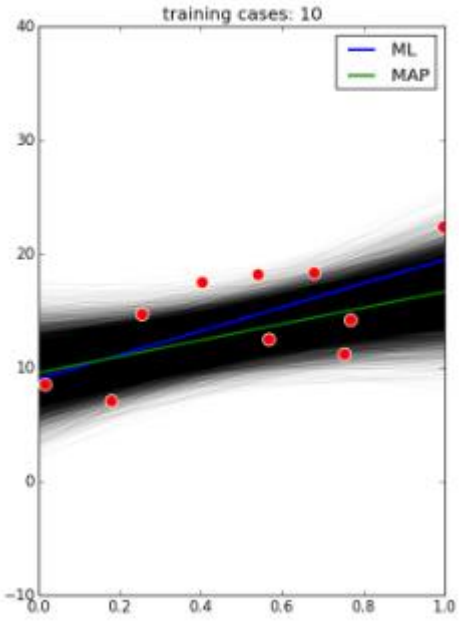
Survival probability of cardiac patients $\theta_j \sim$ population distribution

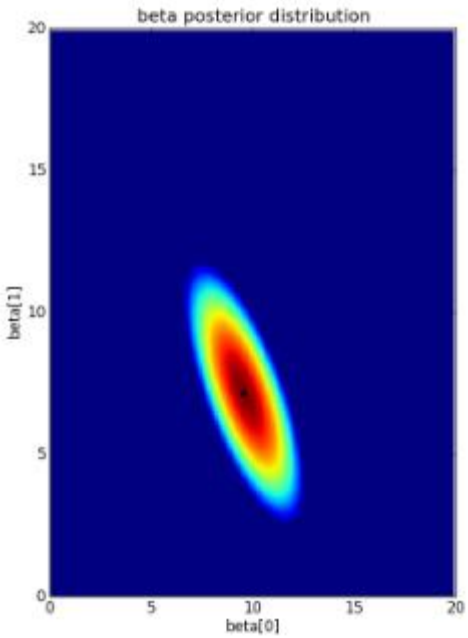


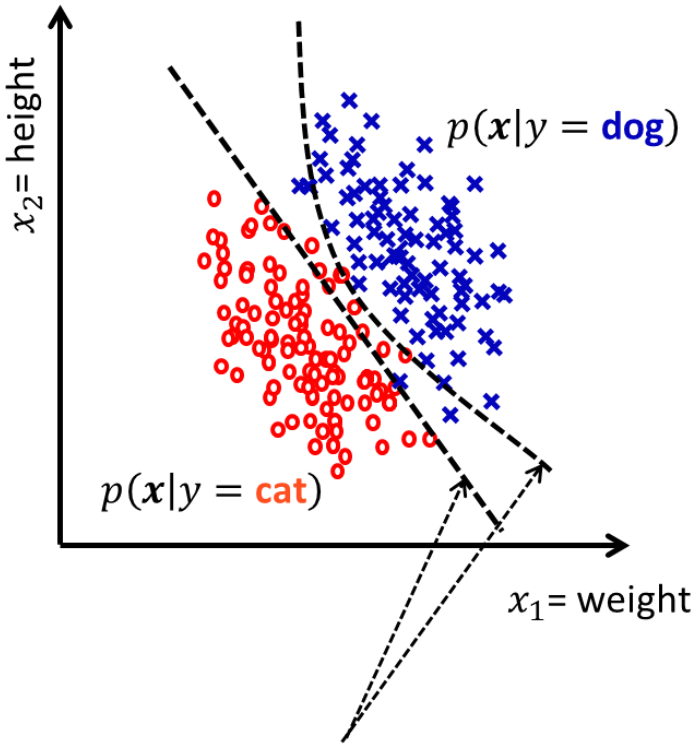
Main subject

2. Single-agent, single-stage decision-making (3 weeks)

- **Bayesian regression**
- **Bayesian classification**
- Bayesian Network
- Influential Diagram







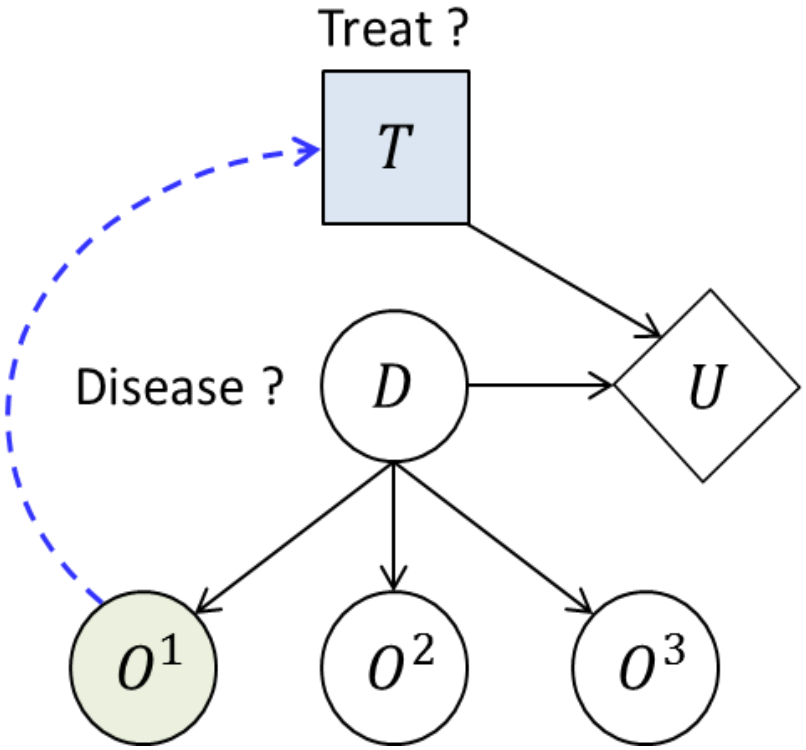
Decision boundary

$$P(y = \text{dog}|\boldsymbol{x}) = P(y = \text{cat}|\boldsymbol{x})$$

Main subject

2. Single-agent, single-stage decision-making (3 weeks)

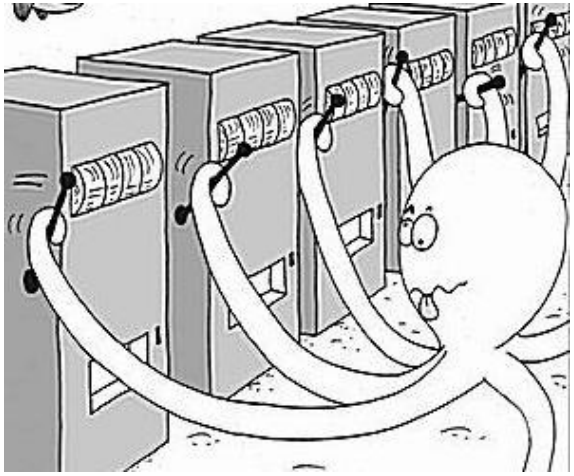
- Bayesian regression
- Bayesian classification
- **Bayesian Network**
- **Influential Diagram**



Main subject

3. Multi-stages decision-making (6 weeks)

- **Bandit problem**
- Bayesian Optimization
- Markov Decision Process (MDP)
- Dynamic Programming
- Reinforcement Learning
  - Monte Carlo Methods
  - Temporal Difference Methods



Facing with N slot machines with different payoff distribution,

→ **Devise a strategy to maximize payoff**

→ **Find the best bandit as quickly as possible**

Acquiring new information (***exploration***)

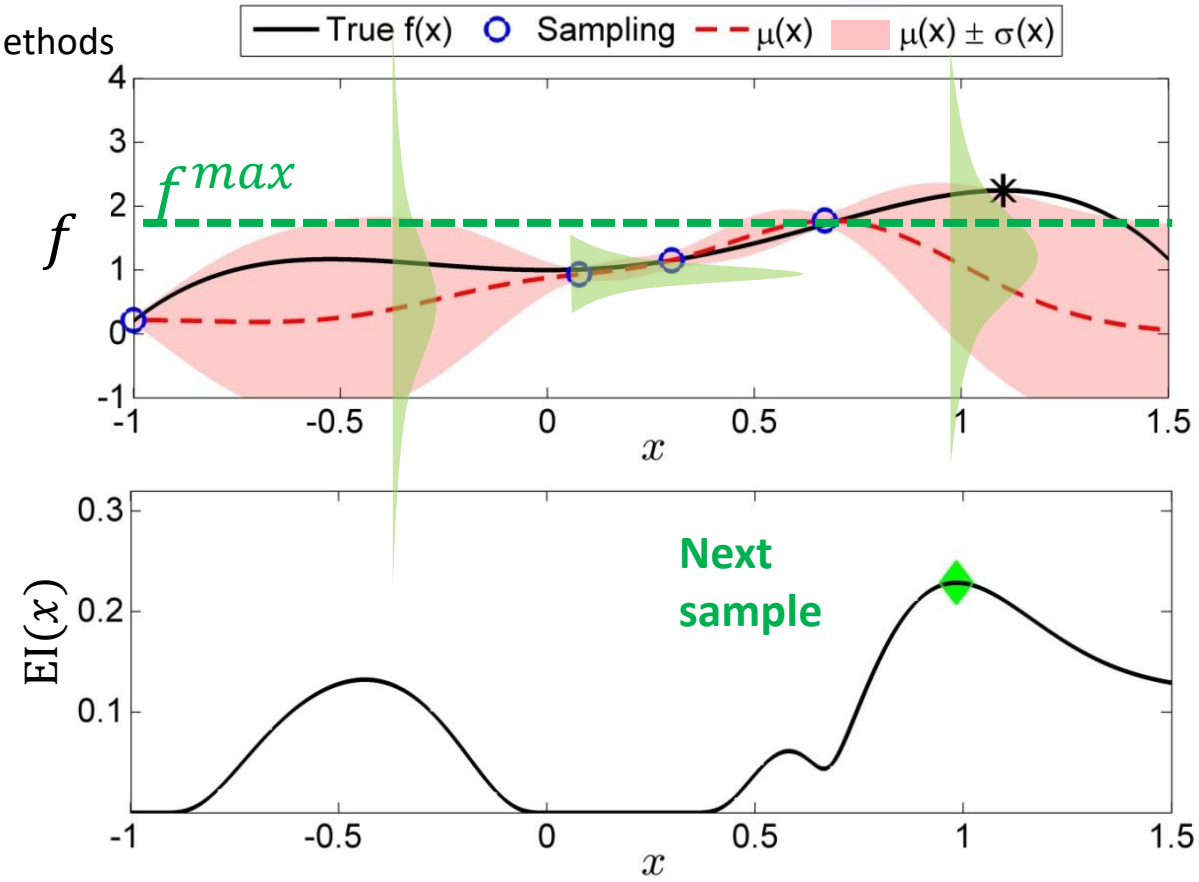
trade-off

capitalizing on the information available so far (***exploitation***)

Main subject

3. Multi-stages decision-making (6 weeks)

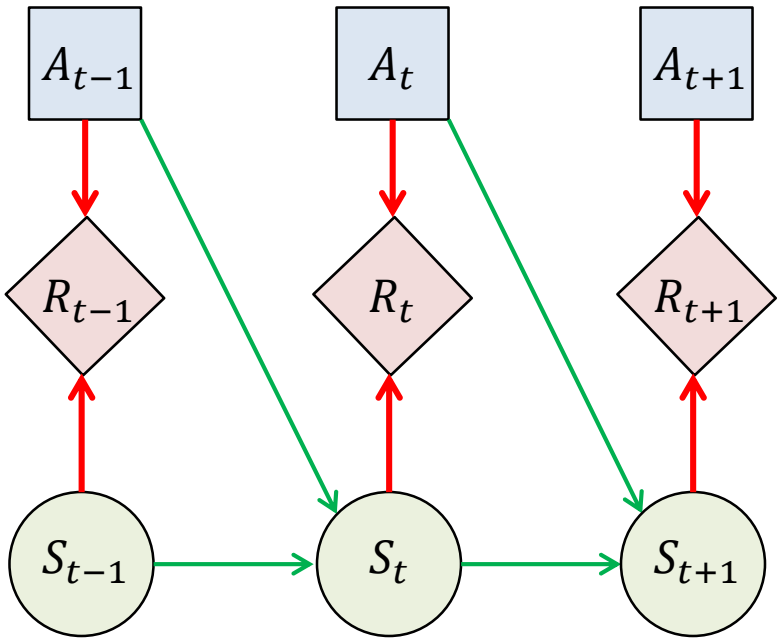
- Bandit problem
- **Bayesian Optimization**
- Markov Decision Process (MDP)
- Dynamic Programming
- Reinforcement Learning
  - Monte Carlo Methods
  - Temporal Difference Methods



Main subject

3. Multi-stages decision-making (6 weeks)

- Bandit problem
- Bayesian Optimization
- **Markov Decision Process (MDP)**
- Dynamic Programming
- Reinforcement Learning
  - Monte Carlo Methods
  - Temporal Difference Methods



A_t : action taken at time t

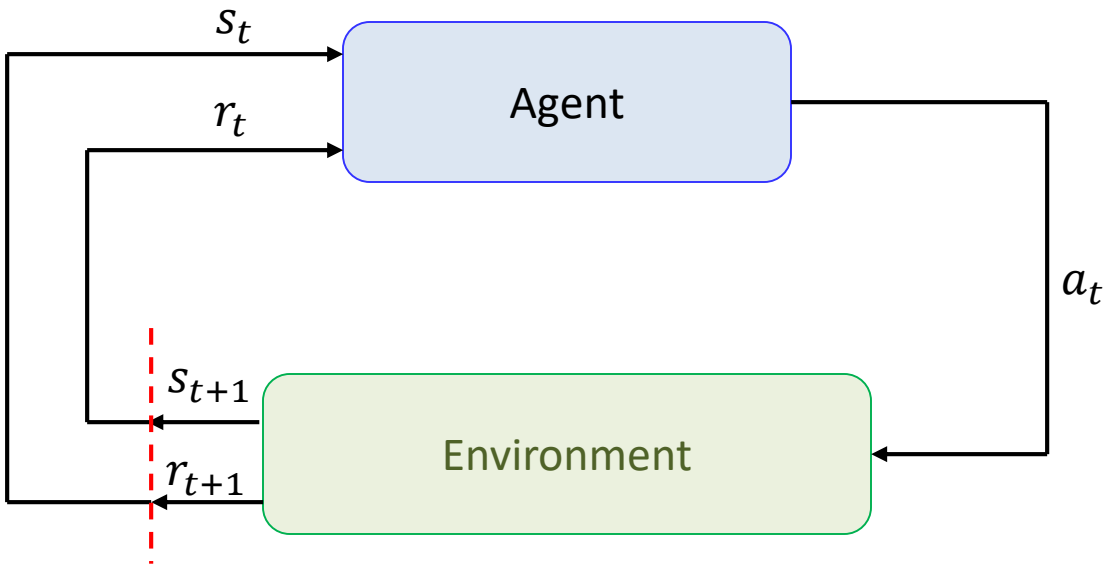
R_t : reward received at time t

S_t : state at time t

Main subject

3. Multi-stages decision-making (6 weeks)

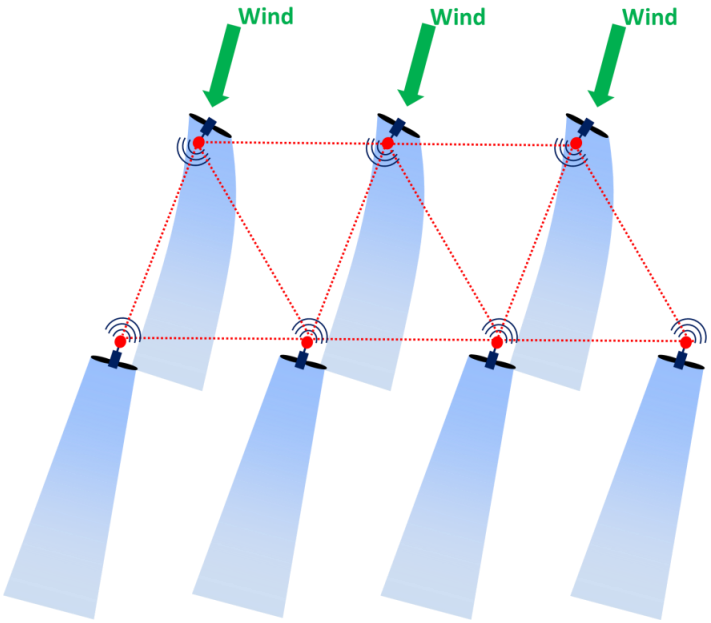
- Bandit problem
- Bayesian Optimization
- Markov Decision Process (MDP)
- Dynamic Programming
- **Reinforcement Learning**
  - Monte Carlo Methods
  - Temporal Difference Methods

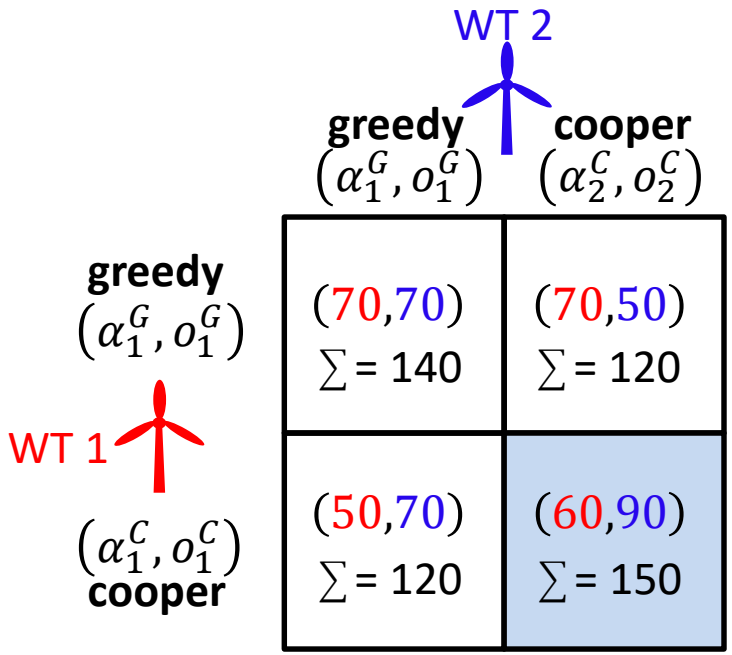


Main subject

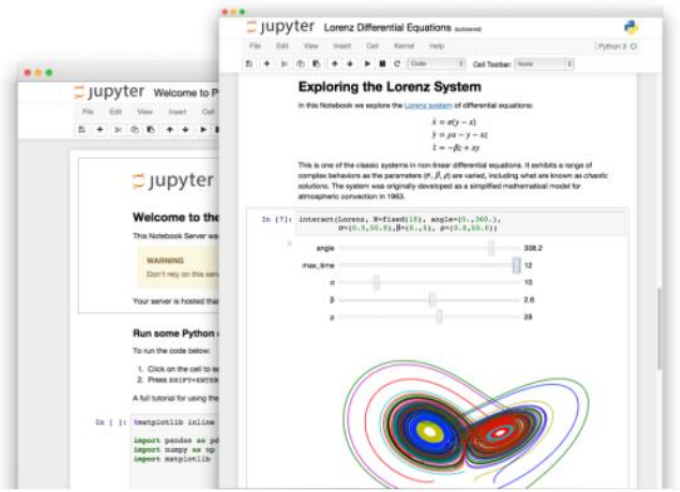
4. Multi-agents decision-making (3 weeks)

- **Basics of game theory**
- Bayesian Game (with uncertainty about other agents)





Computation tool



The Jupyter Notebook

The Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and explanatory text. Uses include: data cleaning and transformation, numerical simulation, statistical modeling, machine learning and much more.

Language of choice

The Notebook has support for over 40 programming languages, including those popular in Data Science such as Python, R, Julia and Scala.

Share notebooks

Notebooks can be shared with others using email, Dropbox, GitHub and the Jupyter Notebook Viewer.

Interactive widgets

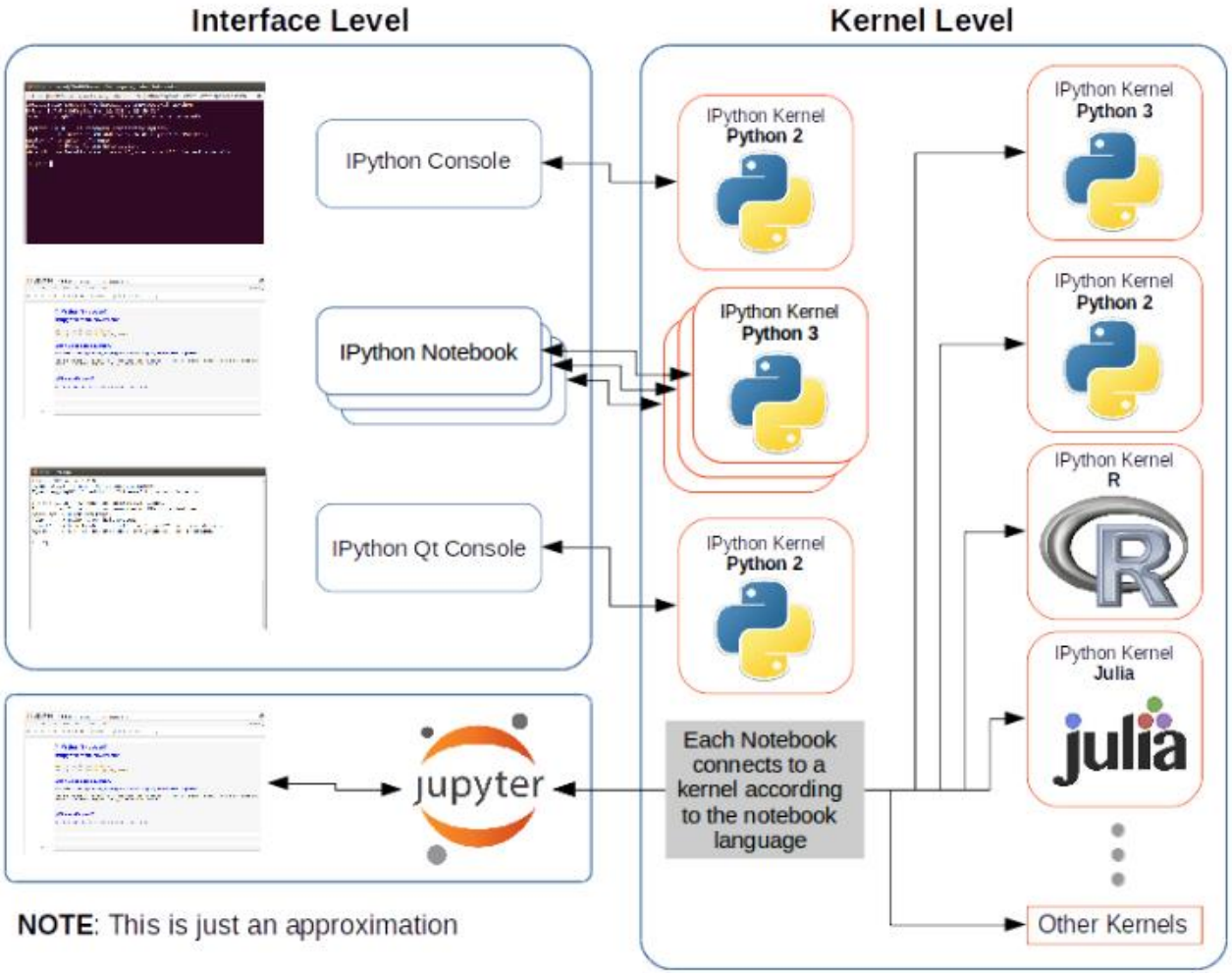
Code can produce rich output such as images, videos, LaTeX, and JavaScript. Interactive widgets can be used to manipulate and visualize data in realtime.

Big data integration

Leverage big data tools, such as Apache Spark, from Python, R and Scala. Explore that same data with pandas, scikit-learn, ggplot2, dplyr, etc.

- It combines complier, ide, text editor, graphic editor…

Computation tool



$$Julia + Python + R = jupyter$$

- Jupyter is the latest version of Ipython and now it includes more than 10 kernels (i.e., python, R, Julia,…)

Computation tool

Basic scientific computing modulus

NumPy

Base N-dimensional array package

SciPy library

Fundamental library for scientific computing

Matplotlib

Comprehensive 2D Plotting

IPython

Enhanced Interactive Console

Sympy

Symbolic mathematics

pandas

Data structures & analysis

Application oriented modules

PyMC3 is a Python package for Bayesian statistical modeling and Probabilistic Machine Learning which focuses on advanced Markov chain Monte Carlo and variational fitting algorithms. Its flexibility and extensibility make it applicable to a large suite of problems.

Edward

A library for probabilistic modeling, inference, and criticism.

Edward is a Python library for probabilistic modeling, inference, and criticism. It is a testbed for fast experimentation and research with probabilistic models, ranging from classical hierarchical models on small data sets to complex deep probabilistic models on large data sets. Edward fuses three fields: Bayesian statistics and machine learning, deep learning, and probabilistic programming.

It supports **modeling** with

GPy

The Gaussian processes framework in Python.

Computation tool

Text regions:

Import Python modulus:

Code for plotting:

Result plot

jupyter nbviewer

JUPYTER FAQ

In Depth: Linear Regression

Just as naive Bayes (discussed earlier in In Depth: Naive Bayes Classification) is a good starting point for classification tasks, linear regression models are a good starting point for regression tasks. Such models are popular because they can be fit very quickly, and are very interpretable. You are probably familiar with the simplest form of a linear regression model (i.e., fitting a straight line to data) but such models can be extended to model more complicated data behavior.

In this section we will start with a quick intuitive walk-through of the mathematics behind this well-known problem, before seeing how before moving on to see how linear models can be generalized to account for more complicated patterns in data.

We begin with the standard imports:

In [1]:
```
%matplotlib inline
import matplotlib.pyplot as plt
import seaborn as sns; sns.set()
import numpy as np
```

Simple Linear Regression

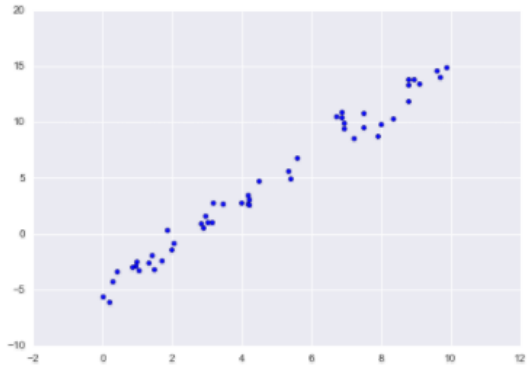
We will start with the most familiar linear regression, a straight-line fit to data. A straight-line fit is a model of the form

$$y = ax + b$$

where a is commonly known as the *slope*, and b is commonly known as the *intercept*.

Consider the following data, which is scattered about a line with a slope of 2 and an intercept of -5:

In [2]:
```
rng = np.random.RandomState(1)
x = 10 * rng.rand(50)
y = 2 * x - 5 + rng.randn(50)
plt.scatter(x, y);
```



We can use Scikit-Learn's `LinearRegression` estimator to fit this data and construct the best-fit line:

In [3]:
```
from sklearn.linear_model import LinearRegression
model = LinearRegression(fit_intercept=True)

model.fit(x[:, np.newaxis], y)
```

Computation tool

Prerequisite: Python

While Jupyter runs code in many programming languages, **Python** is a requirement (Python 3.3 or greater, or Python 2.7) for installing the Jupyter Notebook.

We recommend using the Anaconda distribution to install Python and Jupyter. We'll go through its installation in the next section.

Installing Jupyter using Anaconda and conda

For new users, we **highly recommend** installing Anaconda. Anaconda conveniently installs Python, the Jupyter Notebook, and other commonly used packages for scientific computing and data science.

We recommend using the Anaconda distribution to install Python and Jupyter. We'll go through its installation in the next section.

- Download Anaconda. We recommend downloading Anaconda's latest Python 3 version (currently Python 3.5).
- Install the version of Anaconda which you downloaded, following the instructions on the download page.
- Congratulations, you have installed Jupyter Notebook. To run the notebook:

```
jupyter notebook
```

See Running the Notebook for more details.

Alternative for experienced Python users: Installing Jupyter with pip

As an existing Python user, you may wish to install Jupyter using Python's package manager, pip, instead of Anaconda.

First, ensure that you have the latest pip; older versions may have trouble with some dependencies:

```
pip3 install --upgrade pip
```

Then install the Jupyter Notebook using:

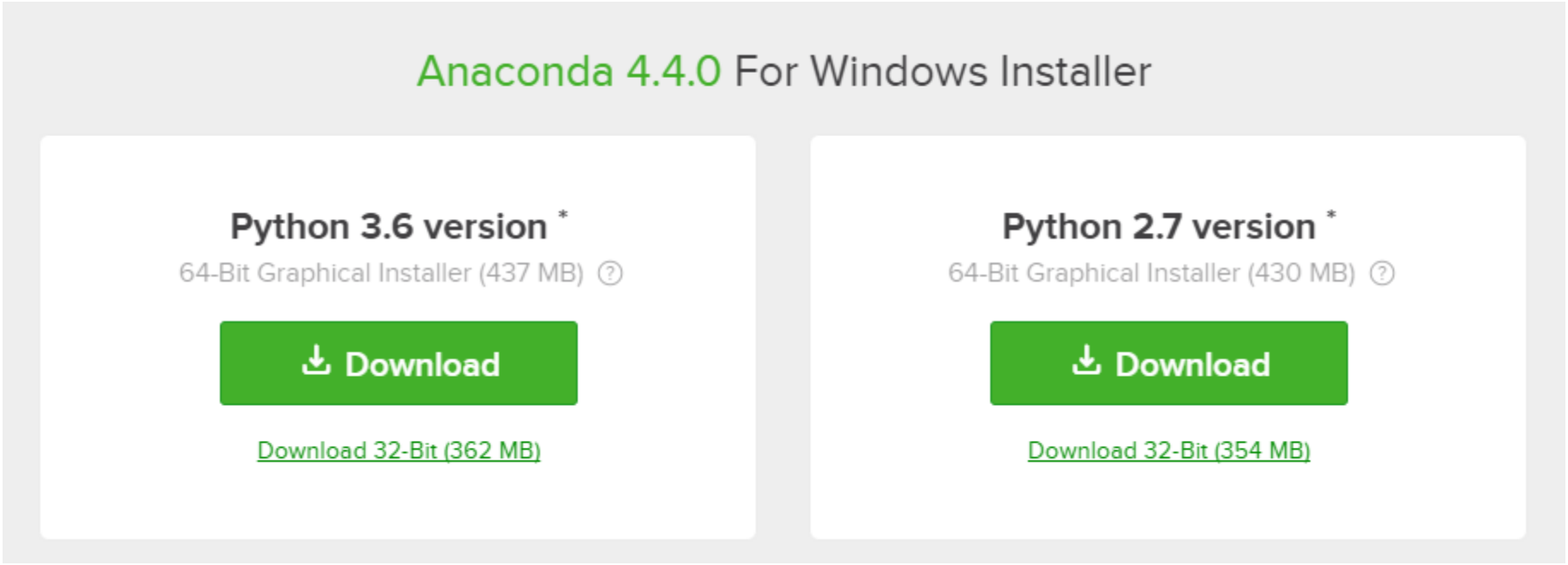
```
pip3 install jupyter
```

(Use pip if using legacy Python 2.)

Congratulations. You have installed Jupyter Notebook.

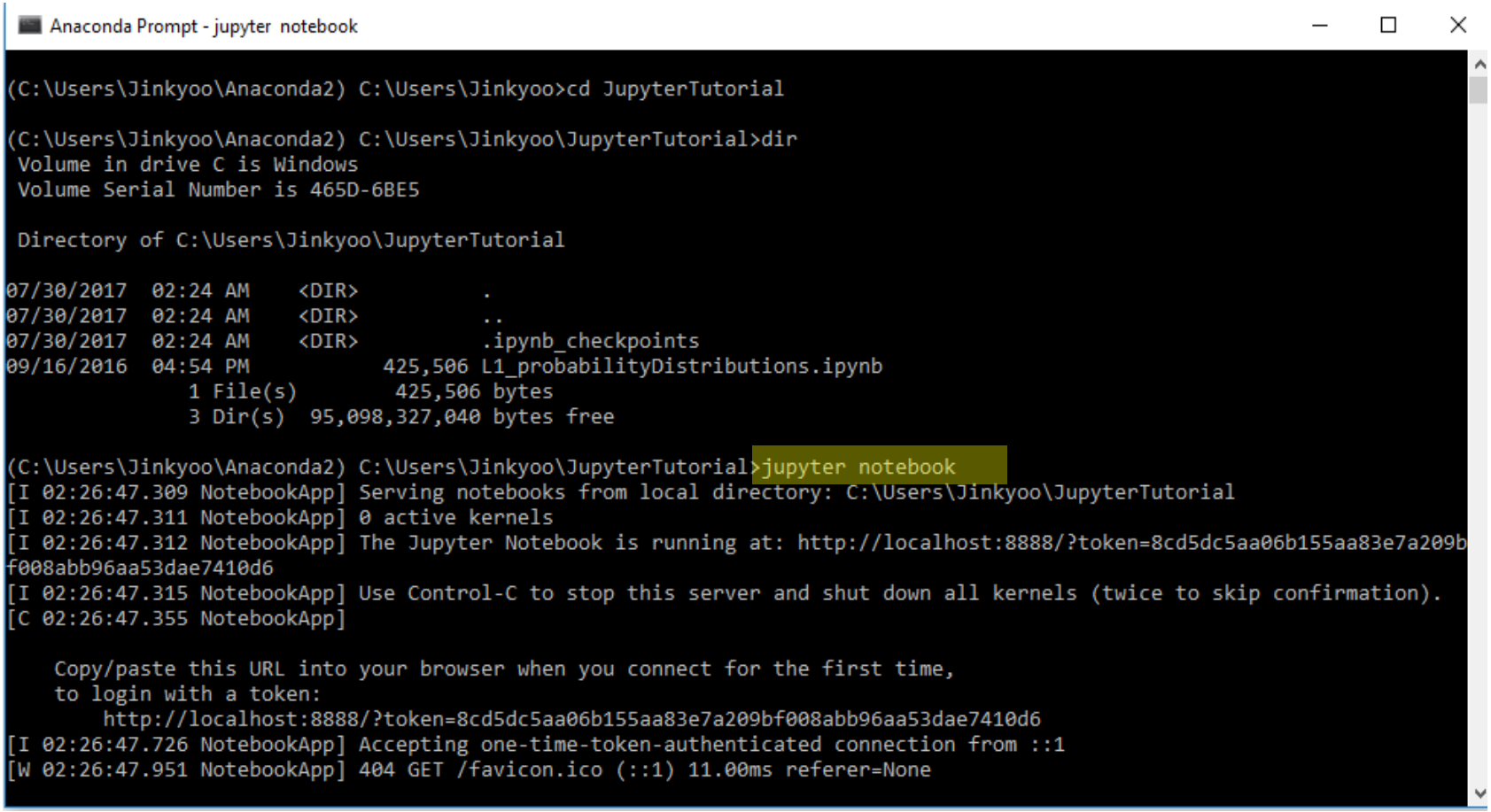
See Running the Notebook for more details.

Computation tool

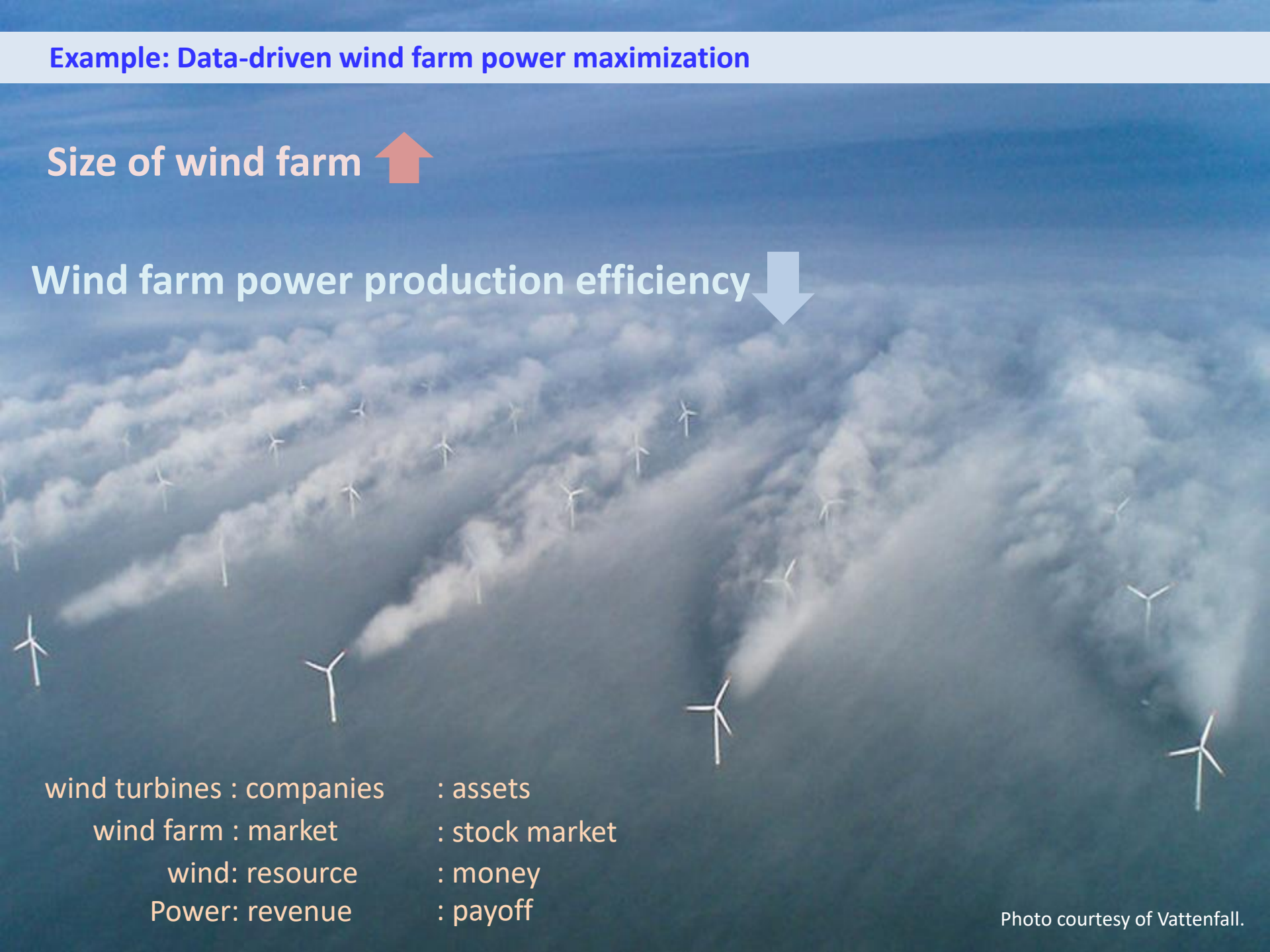


- If Anaconda is used, most scientific computing python modules (i.e., Scipy, Numpy, Matplotlib, Scikit, will be automatically installed)

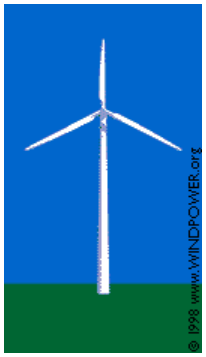
Computation tool



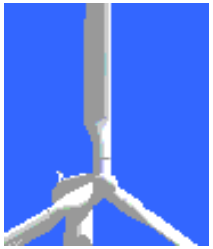
Example: Data-driven wind farm power maximization



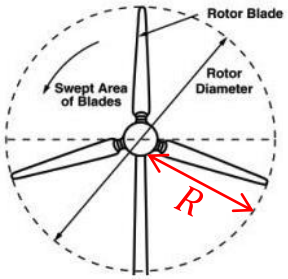
Example: Data-driven wind farm power maximization



<www.windpowr.org>



<www.windpowr.org>

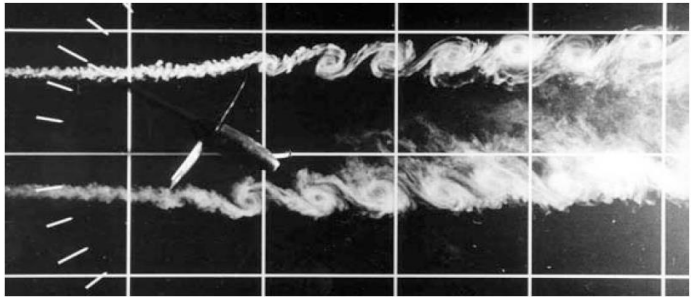


Yaw angle

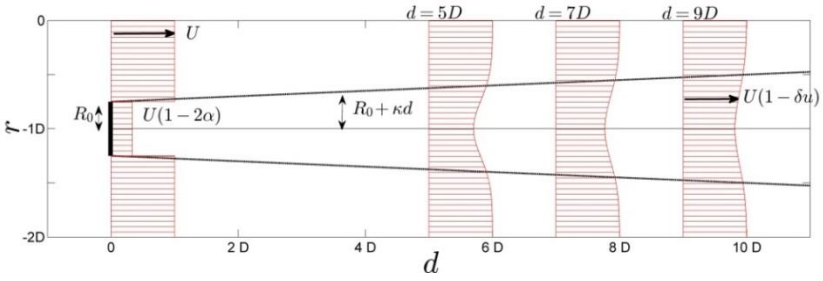
Blade pitch angle

Tip speed ratio

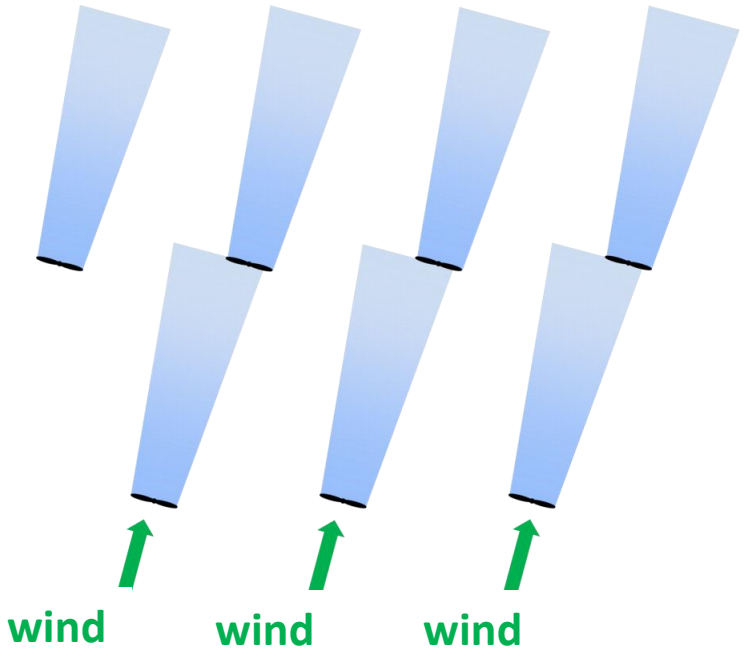
Control Inputs



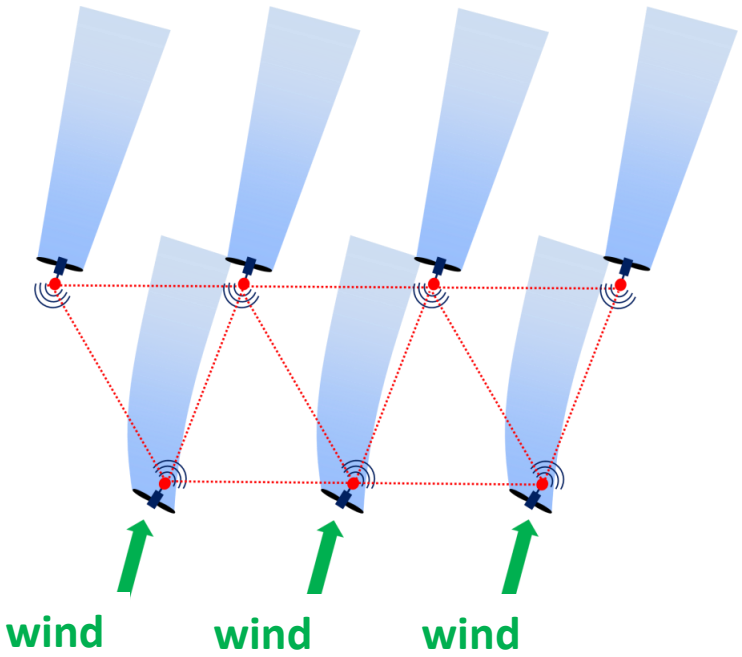
Flow visualization of wake deflection. Conducted at the Royal institute of Technology at 1987, Dahlberg (2003)



Example: Data-driven wind farm power maximization



•sensors



Greedy strategy:

$$\sum_{i=1}^{N} \underset{x_i}{\text{maximize}}\ P_i(\boldsymbol{x}; U, \theta)$$

$\leq$

Cooperative strategy:

$$\underset{\boldsymbol{x}}{\text{maximize}} \sum_{i=1}^{N} P_i(\boldsymbol{x}; U, \theta)$$

$P_i(\boldsymbol{x}; U, \theta)$:Power of wind turbine i

$\boldsymbol{x} = (x_1, \ldots, x_N)$: Control inputs for N wind turbines

Example: Data-driven wind farm power maximization

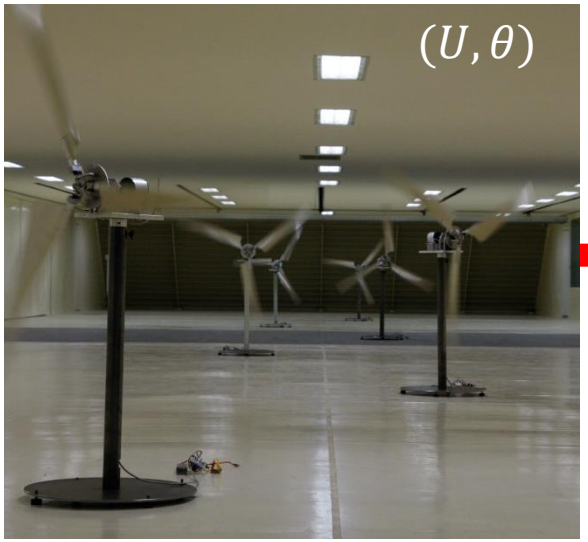
(U, θ) : **Fixed wind condition (Fixed context)**

Input (control actions)

$$\boldsymbol{x}^1 = (x_1^1, \dots, x_N^1)$$

$$\boldsymbol{x}^2 = (x_1^2, \dots, x_N^2)$$

$$\vdots$$



Output (power measurements)

$$y^1 = \sum_{i=1}^{N} P_i(\boldsymbol{x}^1; \theta, U) + \epsilon^1$$

$$y^2 = \sum_{i=1}^{N} P_i(\boldsymbol{x}^2; \theta, U) + \epsilon^2$$

$$\vdots$$

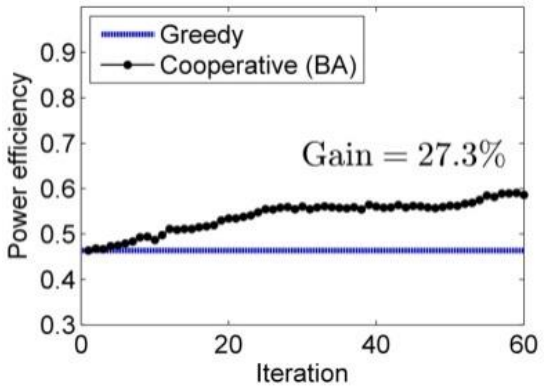
(y = total wind farm power)

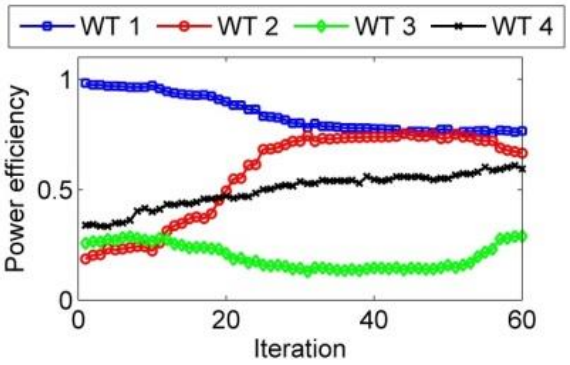
subject to $\boldsymbol{x}^l \leq \boldsymbol{x} \leq \boldsymbol{x}^u$

$$\underset{\boldsymbol{x}}{\text{maximize}} \; f(\boldsymbol{x}; U, \theta) = \sum_{i=1}^{N} P_i(\boldsymbol{x}; \theta, U)$$

Without knowing $f(\boldsymbol{x}; U, \theta)$

Example: Data-driven wind farm power maximization





Greedy control (initial)

Cooperative control (after convergence)