

HSI 2020: Advanced Computational Chemistry

Applied Machine Learning for Chemistry

Aug 26: 10:30~12:00 (90min)

Aug 26: 13:00~14:30 (90min)

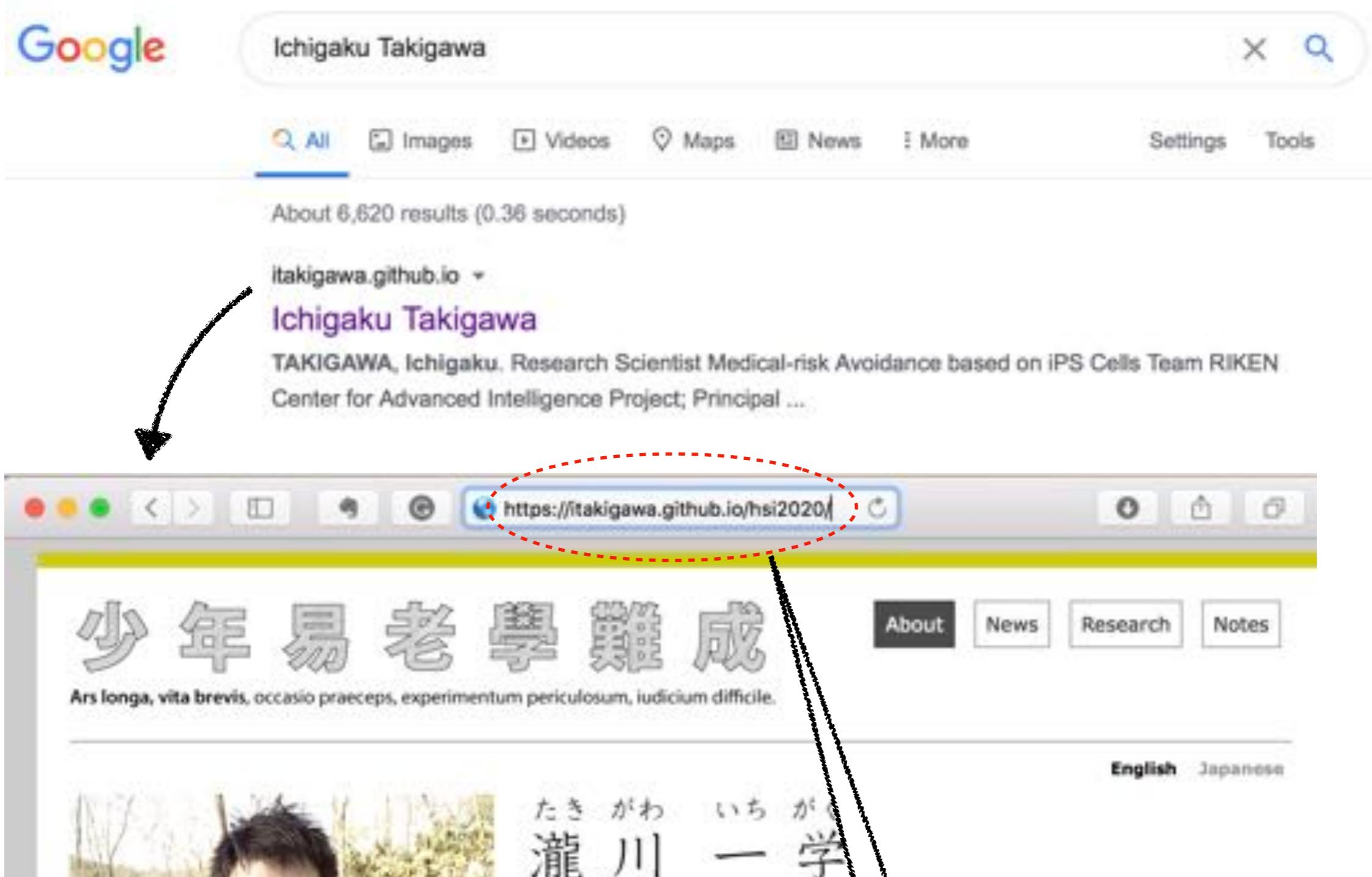
Ichigaku Takigawa

ichigaku.takigawa@riken.jp

<https://itakigawa.github.io>



You can get this slide PDFs and more.



https://itakigawa.github.io/hsi2020/

Brief Bio: Ichigaku TAKIGAWA



Computer Scientist (Machine Learning)

10 years@**Hokkaido U**
(1995~2004)

Statistical Machine Learning & Signal Processing
(Dept. Engineering)

7 years@**Kyoto U**
(2005~2011)

Bioinformatics (Institute for Chemical Research)
Chemoinformatics (Dept. Pharmaceutical Sciences)

7 years @**Hokkaido U**
(2012~2018)

Machine Learning (Dept. Information Science and
Technology)
JST Presto (Advanced Materials Informatics Group)

? years@**RIKEN**
(2019~)

Medical-risk Avoidance based on iPS Cells Team (**RIKEN**
Center for Advanced Intelligence Project)
Institute for Chemical Reaction Design and Discovery
(Hokkaido U)

Aug 26: 10:30~12:00 (90min)

1. What is "machine learning"?
2. Why does it matter to chemists?
3. Let's try it in your browser (with no setup!)

Aug 26: 13:00~14:30 (90min)

4. Five things all beginners should know
 - "The quality of your inputs decide the quality of your output"
 - Training / validation / test data
 - Tuning hyperparameters
 - Identification and design of input variables (or "descriptors")
 - "Correlation does not imply causation"
5. Standard pipeline and deep learning
6. Current efforts and future directions

Aug 26: 10:30~12:00 (90min)

1. What is "machine learning"?
2. Why does it matter to chemists?
3. Let's try it in your browser (with no setup!)

Aug 26: 13:00~14:30 (90min)

4. Five things all beginners should know
 - "The quality of your inputs decide the quality of your output"
 - Training / validation / test data
 - Tuning hyperparameters
 - Identification and design of input variables (or "descriptors")
 - "Correlation does not imply causation"
5. Standard pipeline and deep learning
6. Current efforts and future directions

"machine learning" is a new way of programming

Consider when you need to write a code for a "Rock paper scissors" robot.



"machine learning" is a new way of programming

Your first task would be to write a code for computers to recognize the hand shapes among rock, paper, or scissors.



"machine learning" is a new way of programming

But you'll instantly recognize this task is really
really hard... we need to consider many variations
and nuisances... but human can do this easily.



"machine learning"

Concerned with the question of how to construct computer programs that automatically improve with experience.

————— Tom Mitchell

Tasks below would need experience rather than a single principle.

- Learning to recognize spoken words, handwritten characters, etc
- Learning to recognize who is who by seeing faces
- Learning to walk, speak, swim, ski, etc.
- Learning to drive an autonomous vehicle
- Learning to play world-class go, chess, shogi, etc.

"machine learning"

"Machine" means **computer programs**

"Learning" means to **automatically improve with experience**

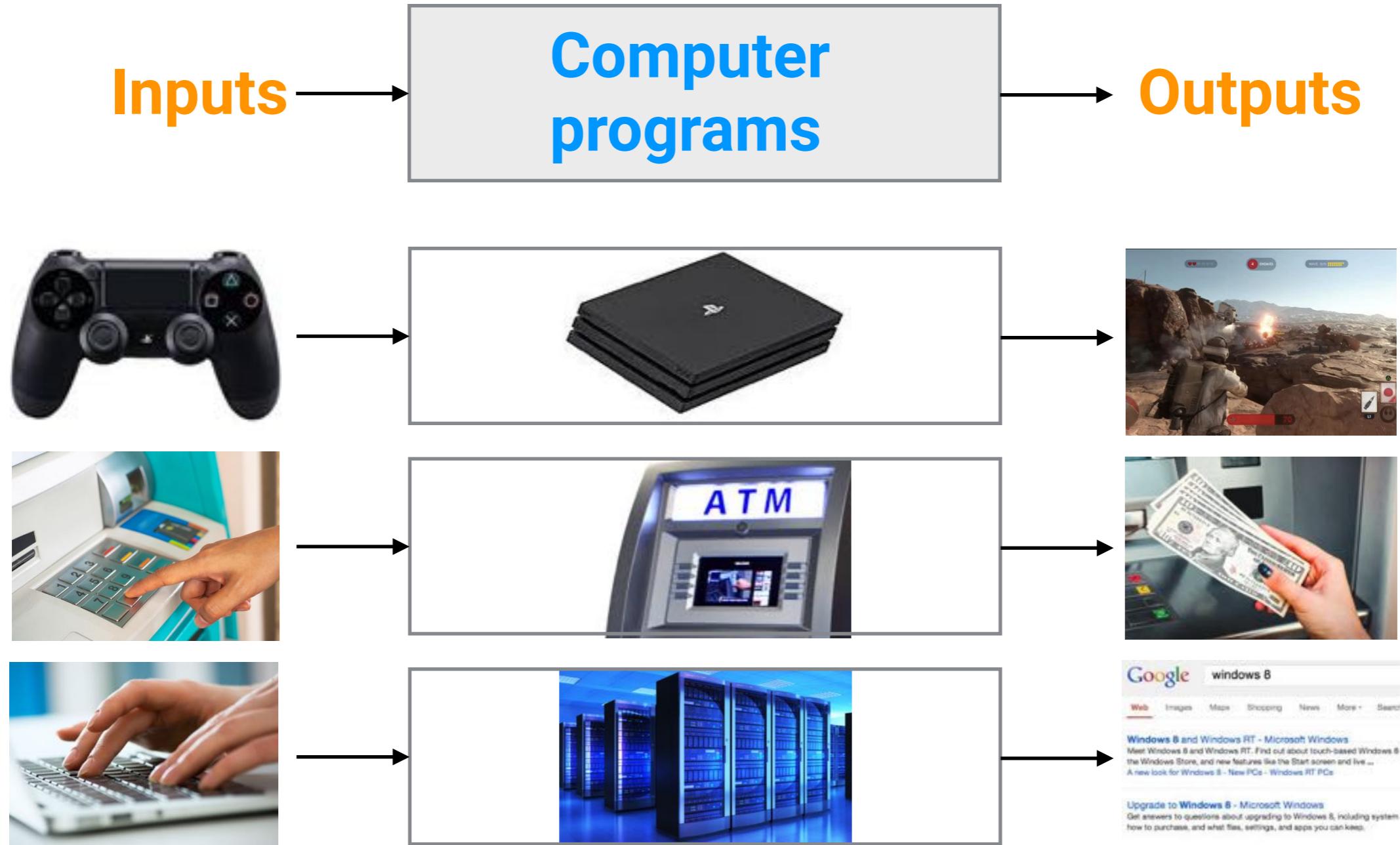
By machine learning, let's get **computer programs** to
automatically improve with experience

But what exactly means....

- Computer programs? 🤔
- Automatically improve with experience? 🤔

Computer programs

Any computer programs process inputs to get outputs.



Computer programs

Any computer programs process inputs to get outputs.



- We need to **explicitly** know the complete procedure to get outputs from inputs. 😱
- We **manually** code the computer program using computer programming languages. 😵

But often we don't know how to do it 😞



Consider how we can construct **computer programs** that can

- Learn to recognize spoken words, handwritten characters, etc
- Learn to recognize who is who
- Learn to walk, speak, swim, ski, etc.
- Learn to drive an autonomous vehicle
- Learn to play world-class go, chess, shogi, etc.

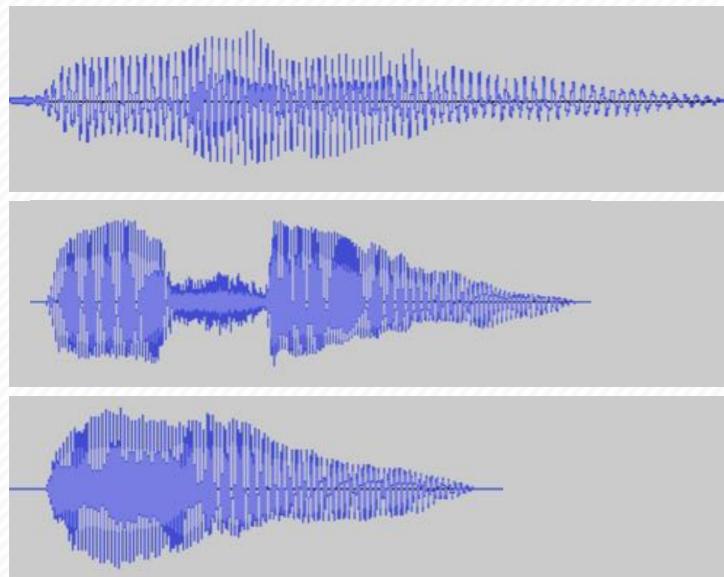
So we give “training data” to teach programs

“automatically improve with experience (given data)”



“training data” = examples of input-output pairs

Inputs



Outputs

“hello”

“machine”

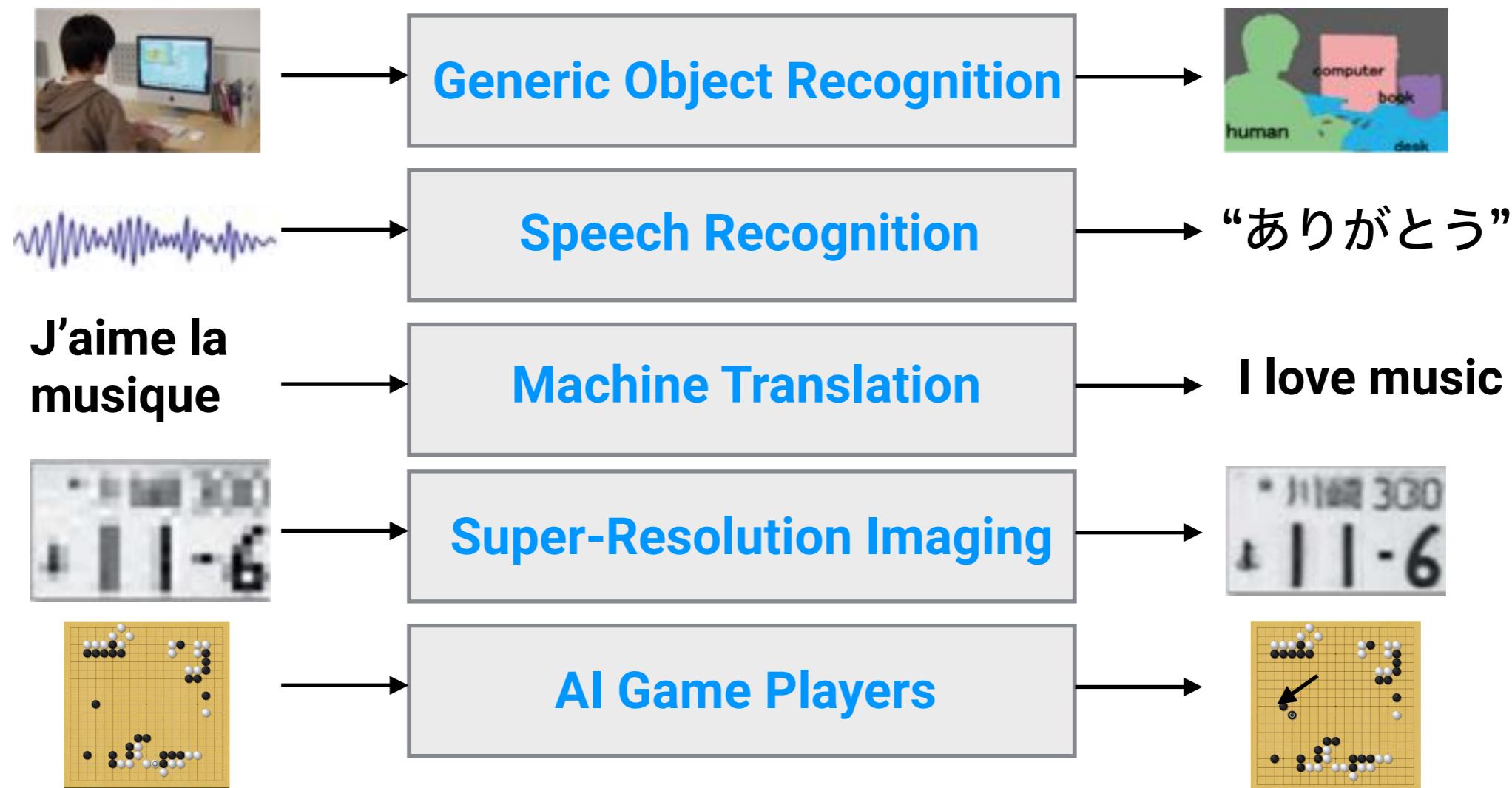
“learning”

:

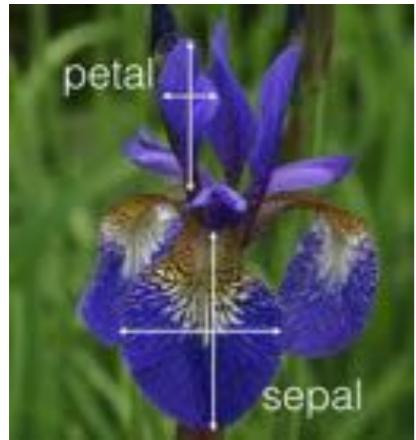
:

(Supervised) machine learning

Machine learning is a way to construct a computer program directly by a given (large) collection of input-output examples without being explicitly programmed.



More typical cases with tabular data



Inputs

x

ML model

Outputs

y

	A	B	C	D	E	F
1	ID	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
2	8	5	3.4	1.5	0.2	setosa
3	51	7	3.2	4.7	1.4	versicolor
4	36	5	3.2	1.2	0.2	setosa
5	15	5.8	4	1.2	0.2	setosa
6	60	5.2	2.7	3.9	1.4	versicolor
7	88	6.3	2.3	4.4	1.3	versicolor
8	126	7.2	3.2	6	1.8	virginica
9	32	5.4	3.4	1.5	0.4	setosa
10	13	4.8	3	1.4	0.1	setosa
11	146	6.7	3	5.2	2.3	virginica
12	5	5	3.6	1.4	0.2	setosa
13	105	6.5	3	5.8	2.2	virginica
14	133	6.4	2.8	5.6	2.2	virginica
15	92	6.1	3	4.6	1.4	versicolor
16	59	6.6	2.9	4.6	1.3	versicolor



Setosa



Versicolor

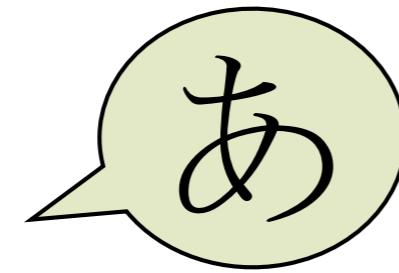


Virginica

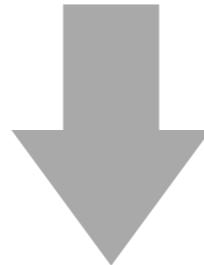
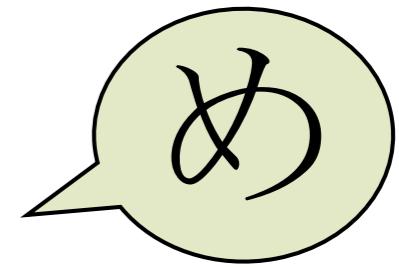
Underlying principle: Use of statistical trends

We see common patterns (empirical rules) emerge from observing many examples, which we cannot recognize when we see only a few.

あ あ あ あ あ
あ あ あ あ あ

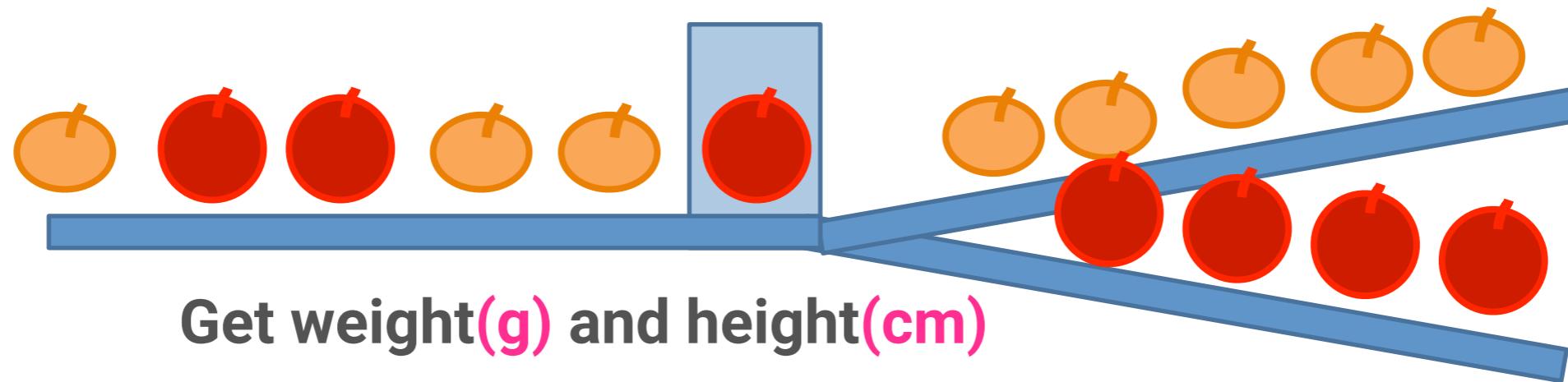


め め め め め
め め め め め

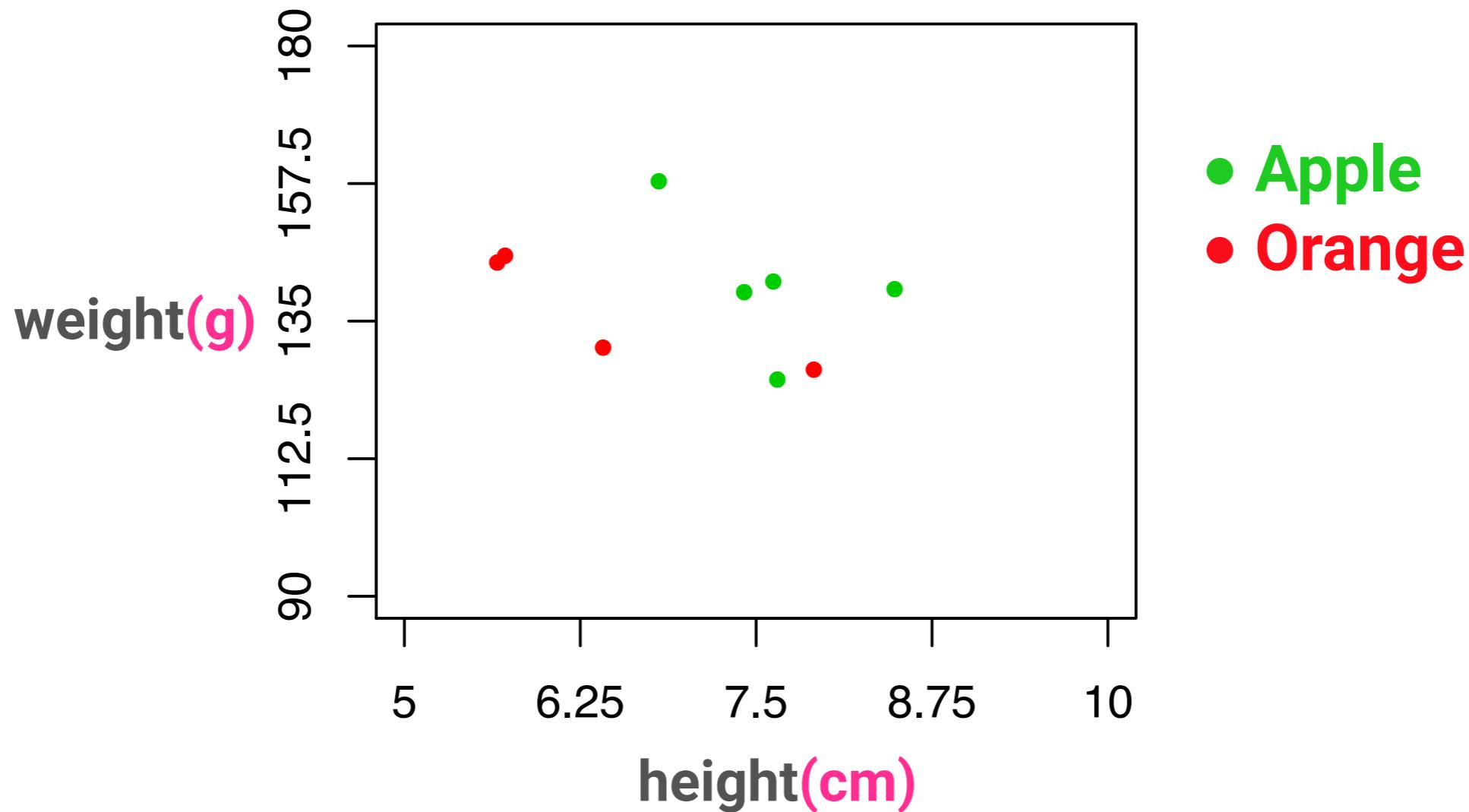
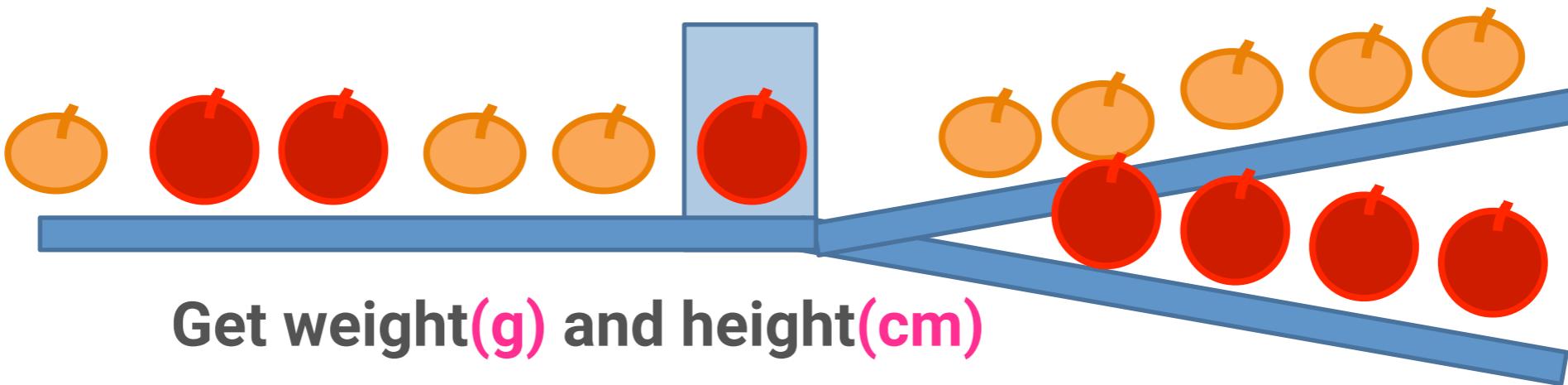


Characterize the difference between “あ” and “め”
not by explicit rules, but by implicit statistical rules directly defined by many observations.

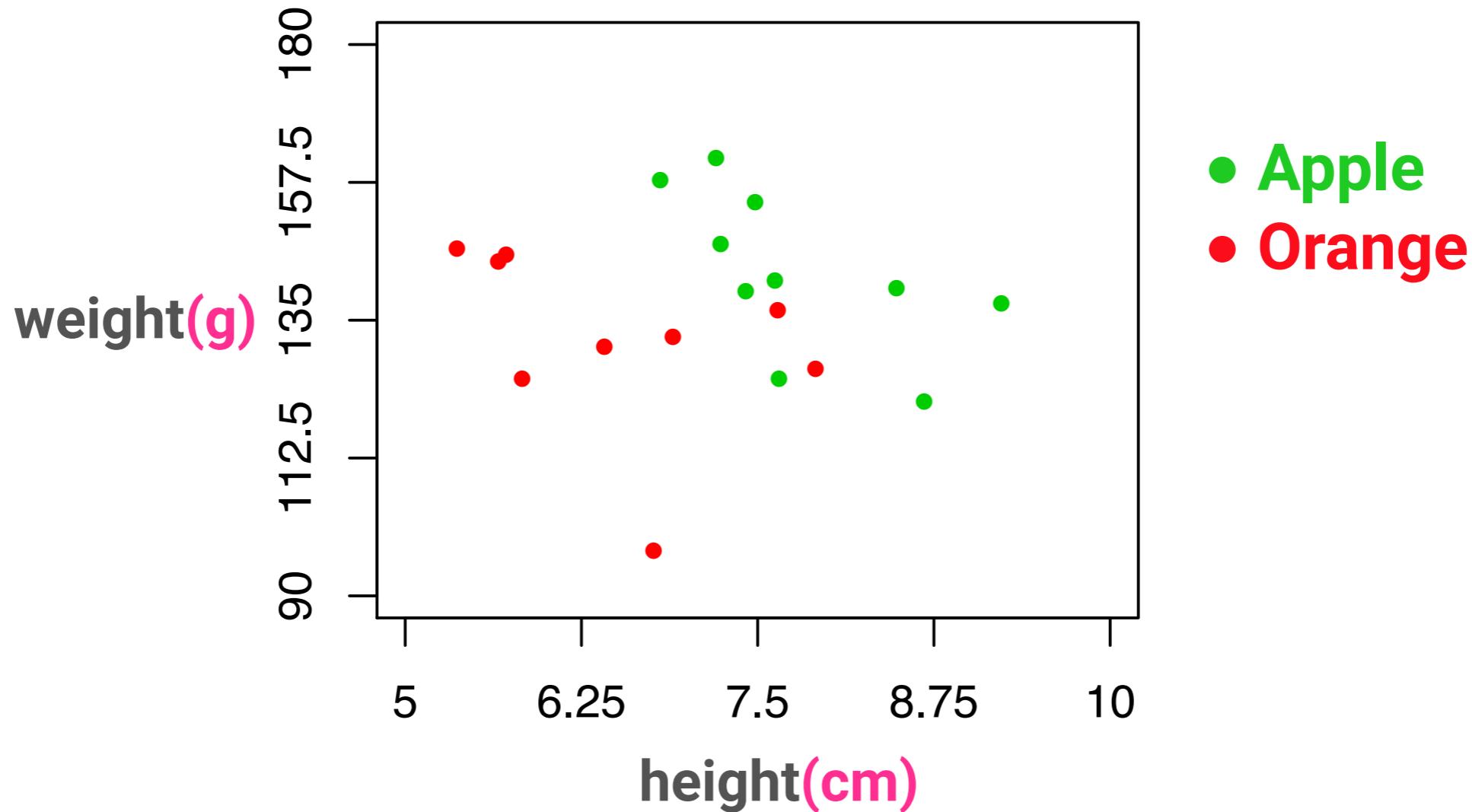
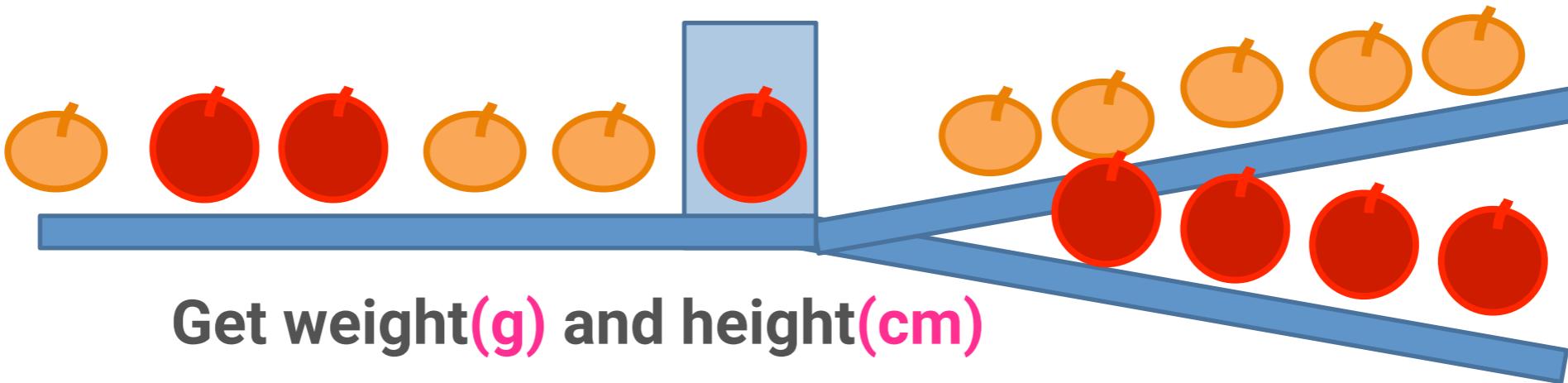
Patterns emerge from many examples?



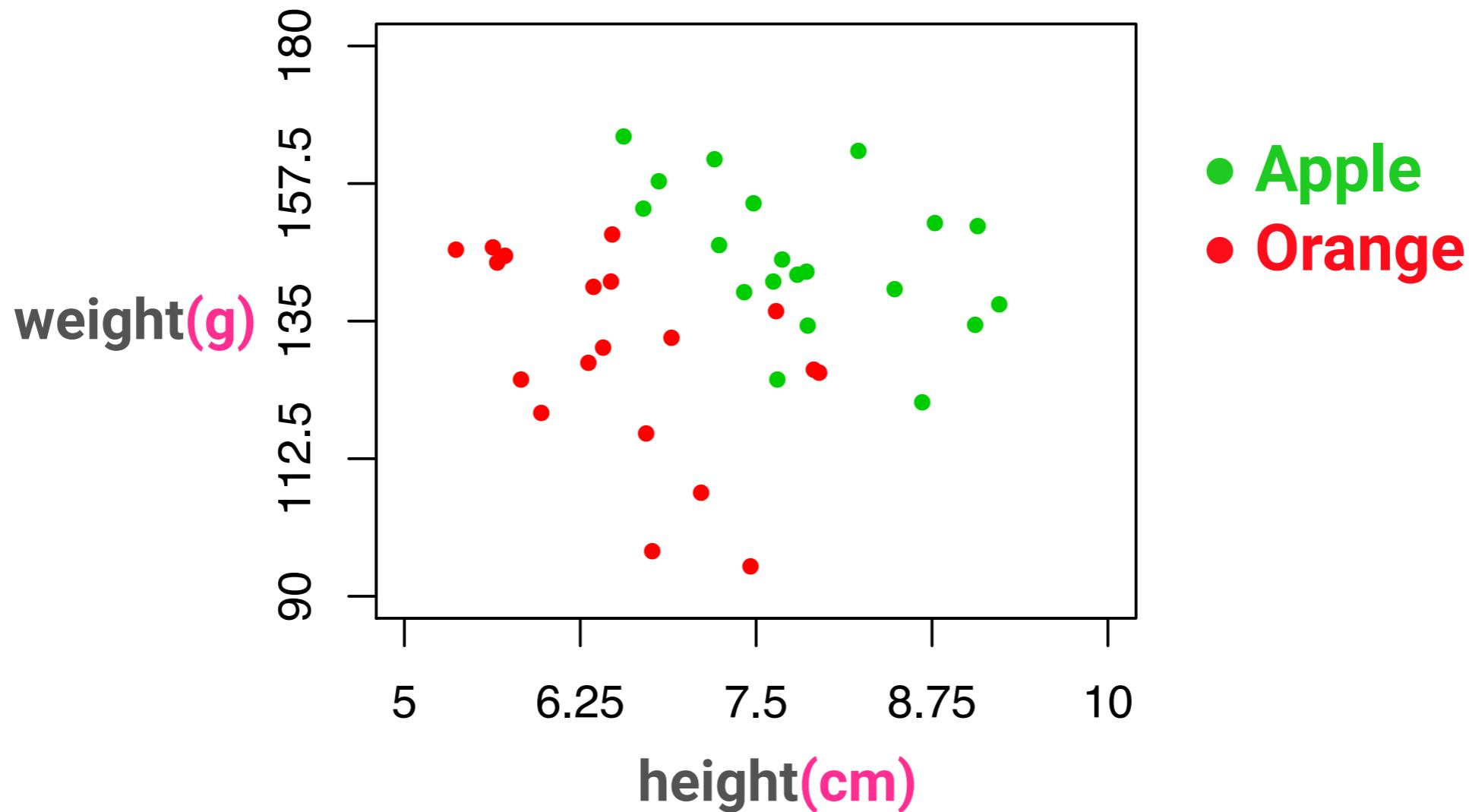
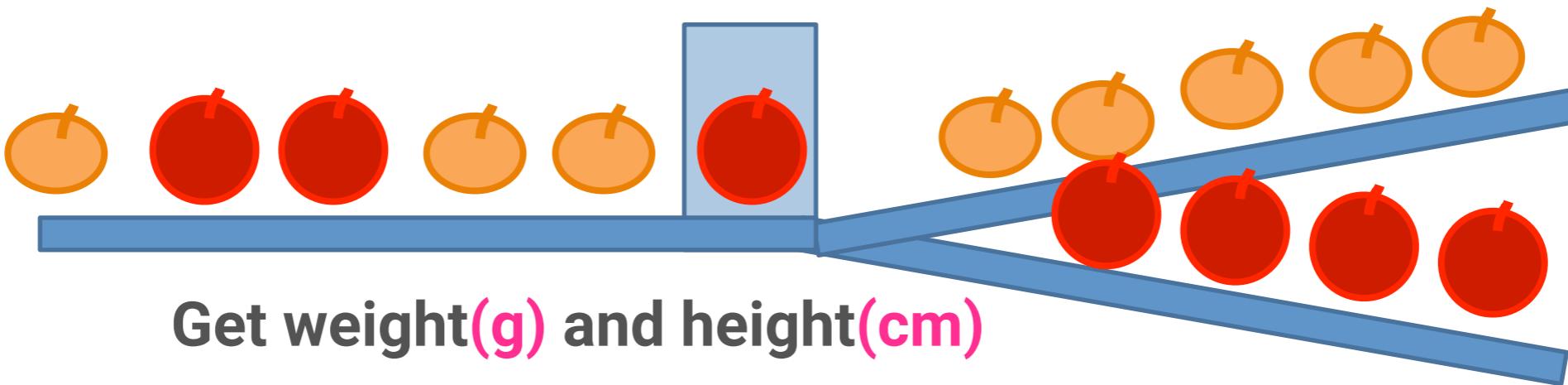
Patterns emerge from many examples?



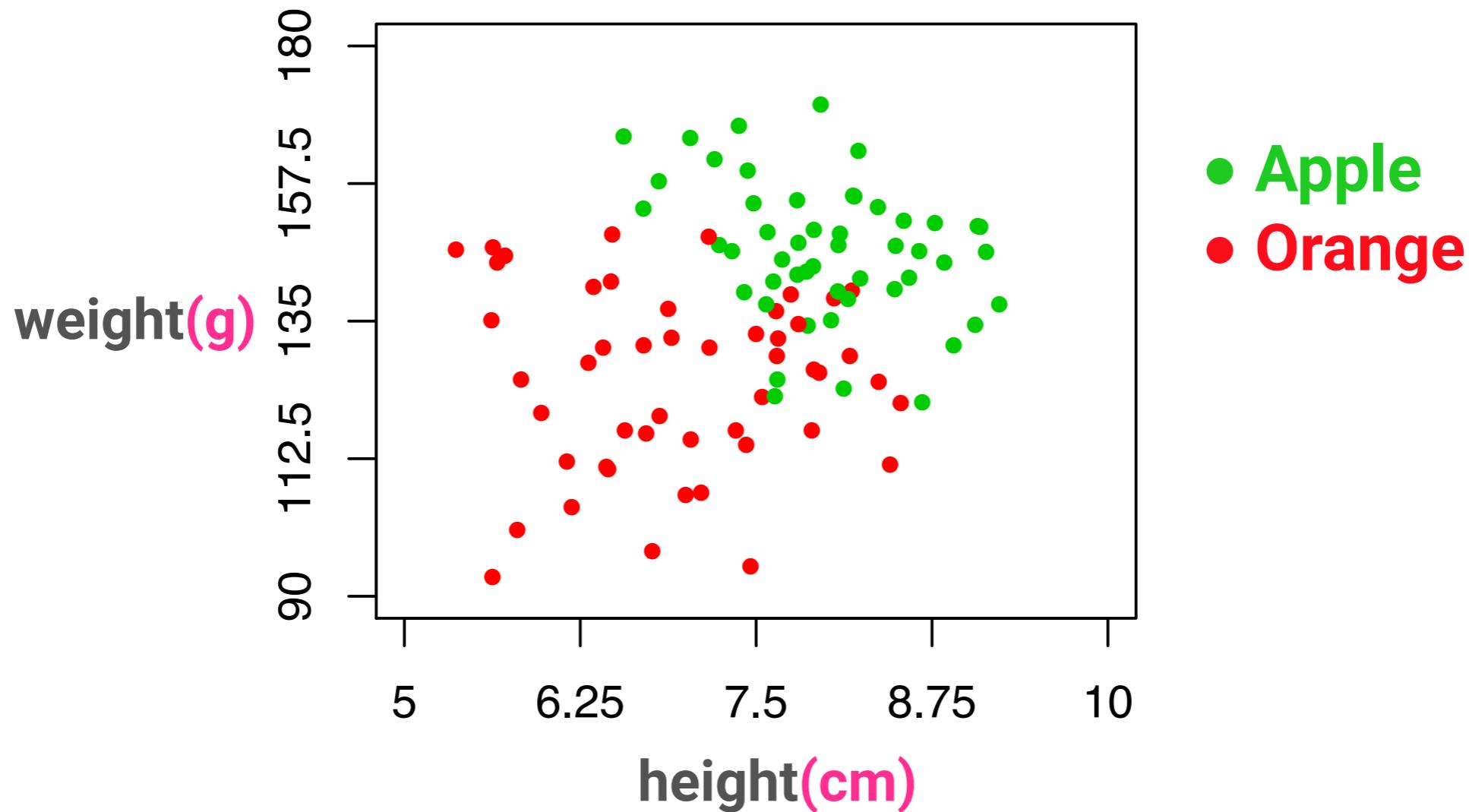
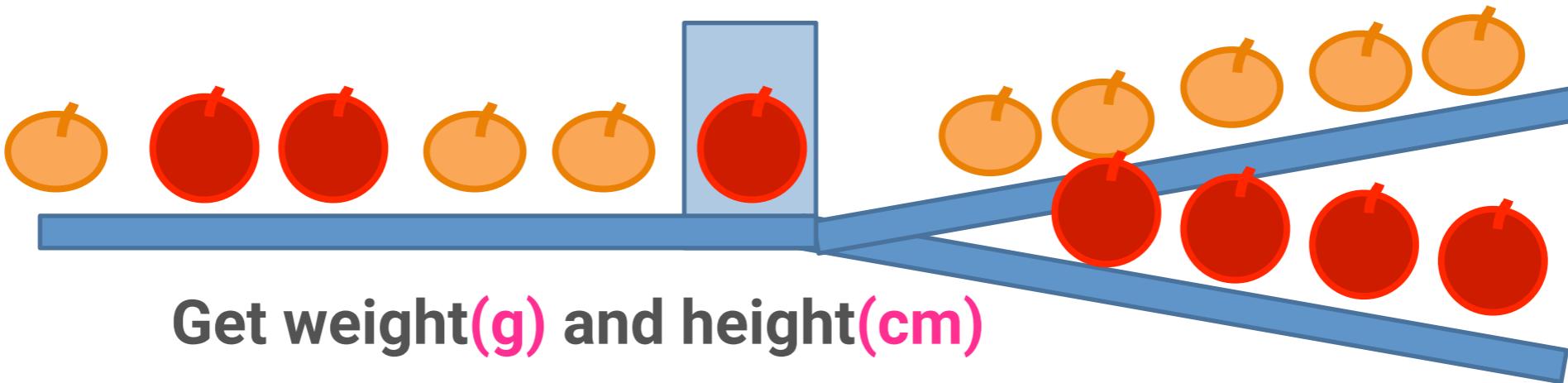
Patterns emerge from many examples?



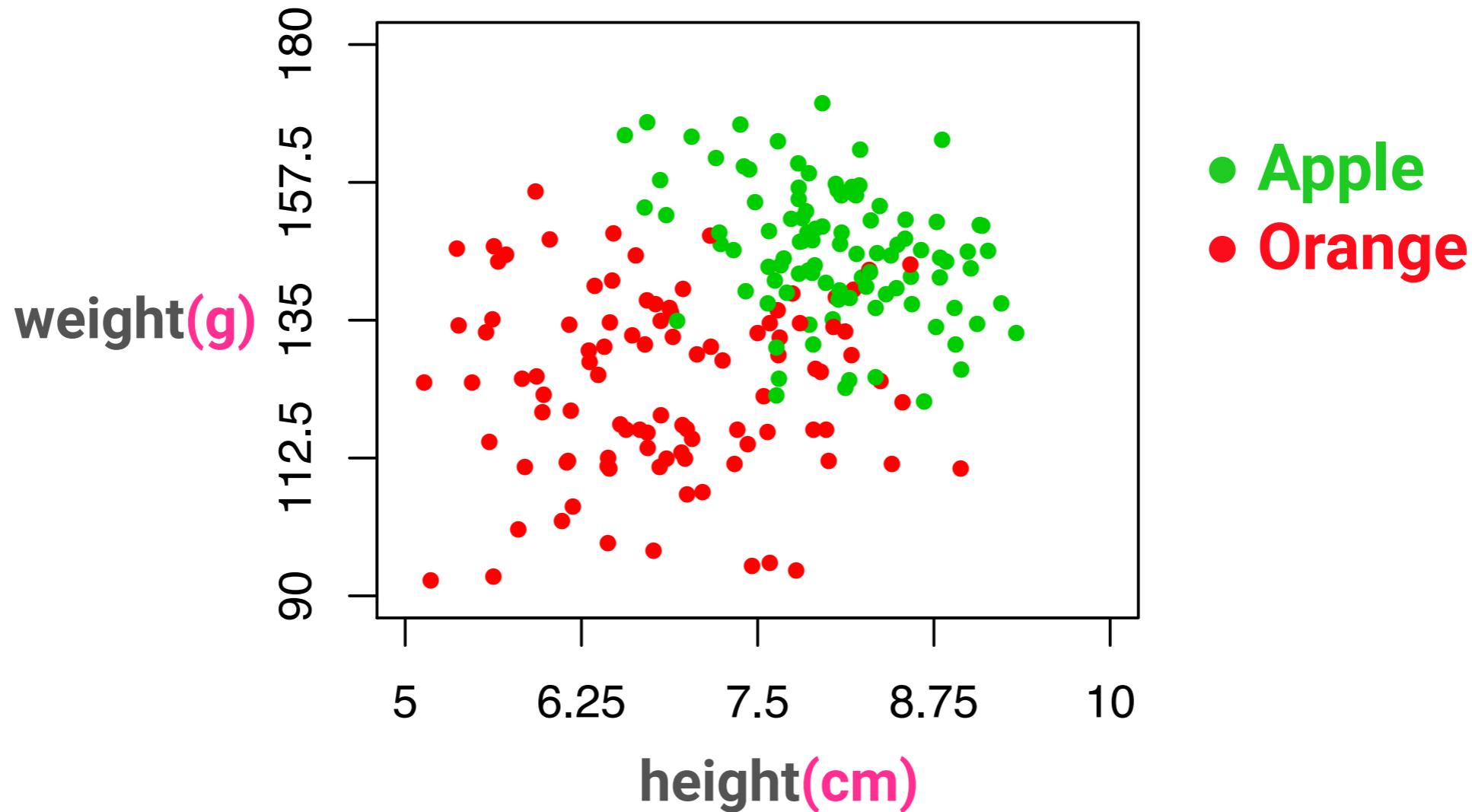
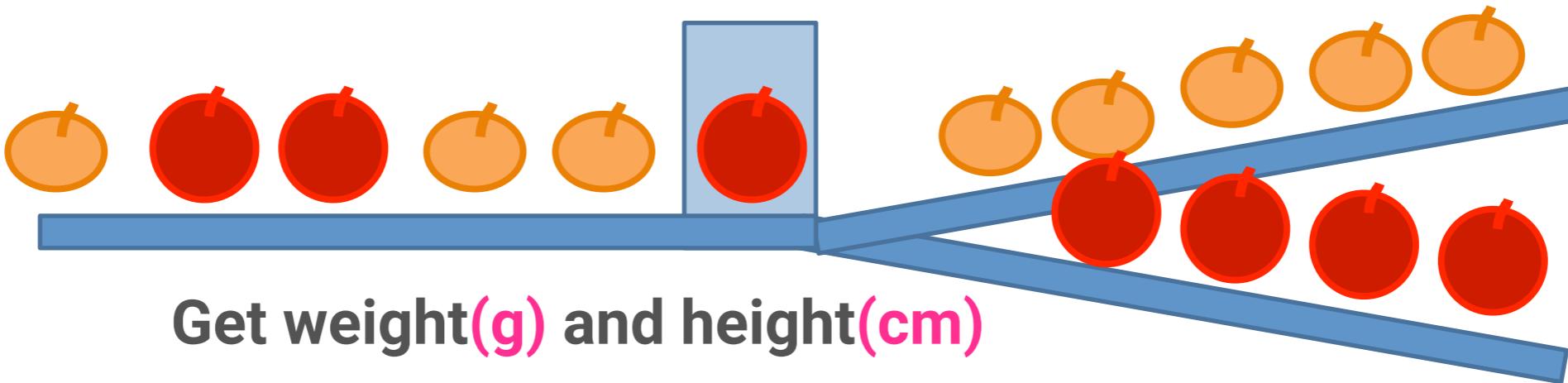
Patterns emerge from many examples?



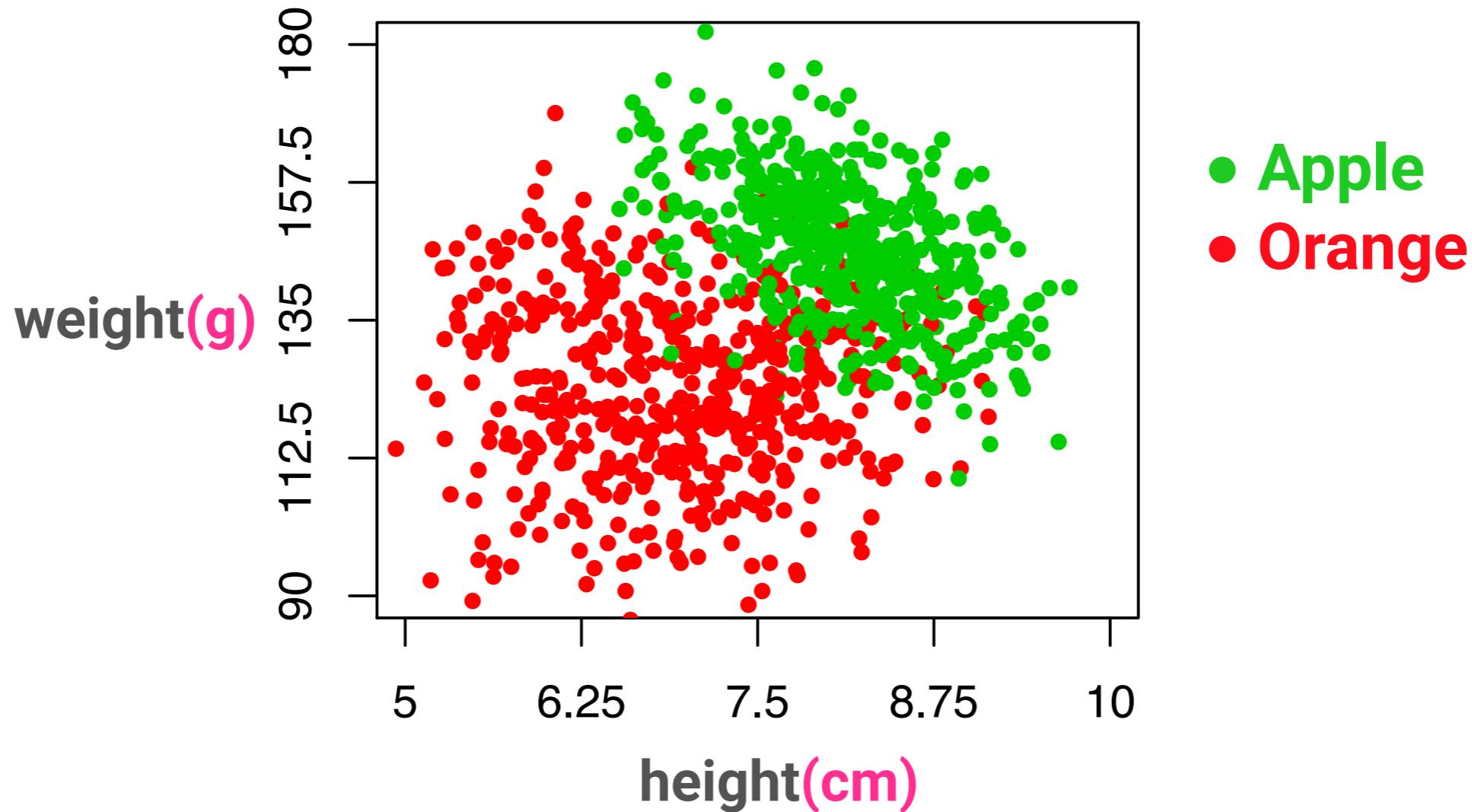
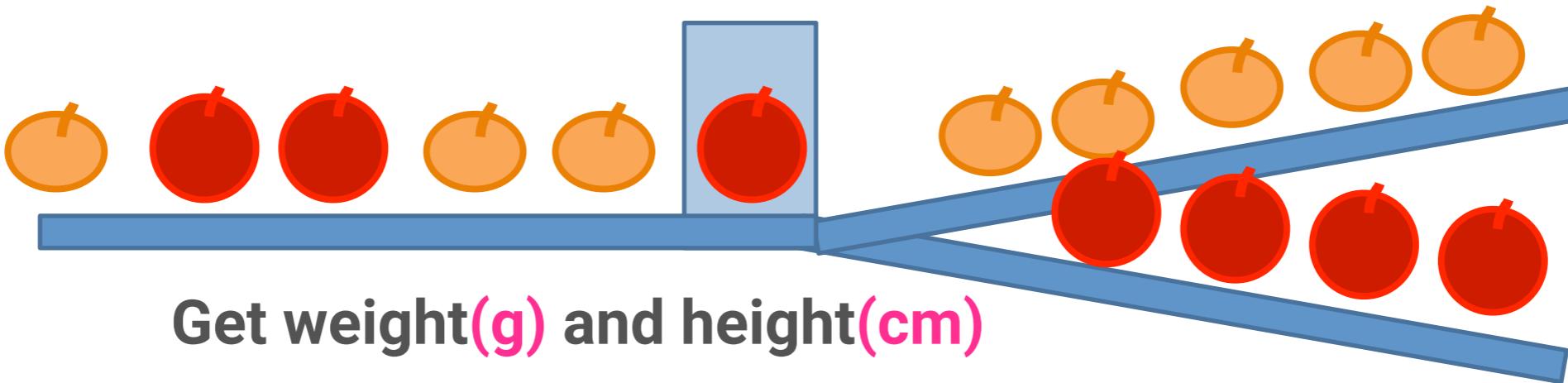
Patterns emerge from many examples?



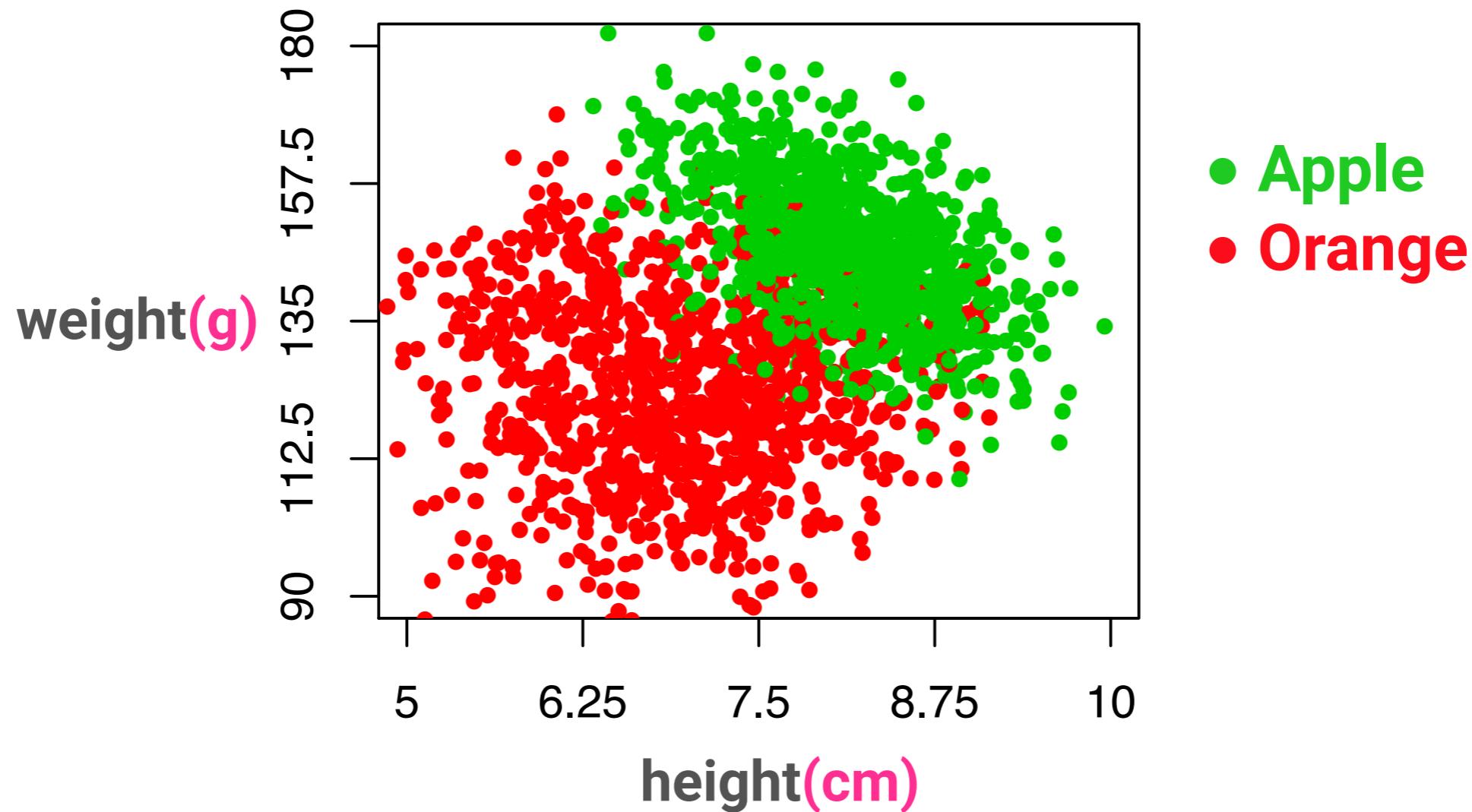
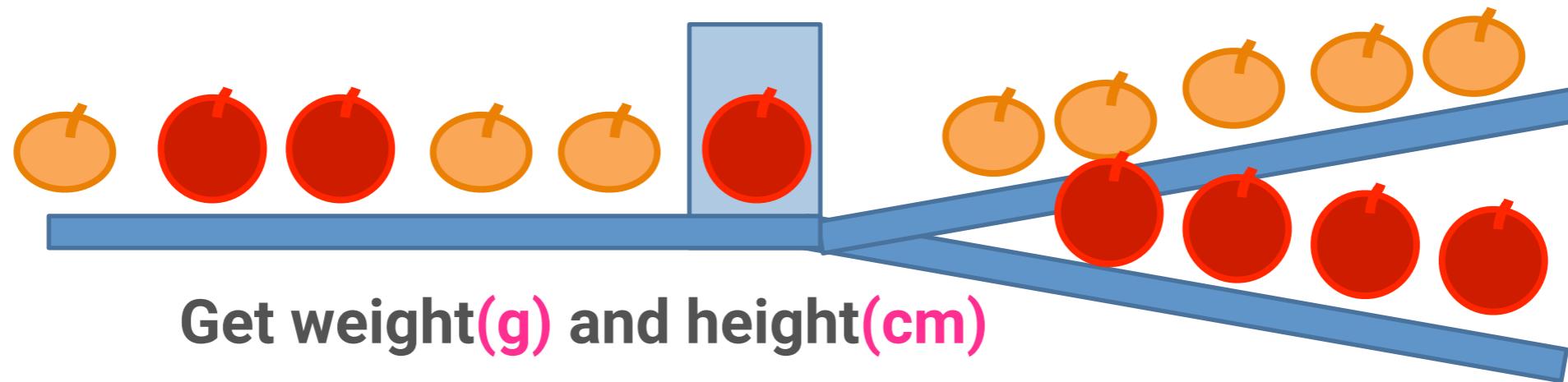
Patterns emerge from many examples?



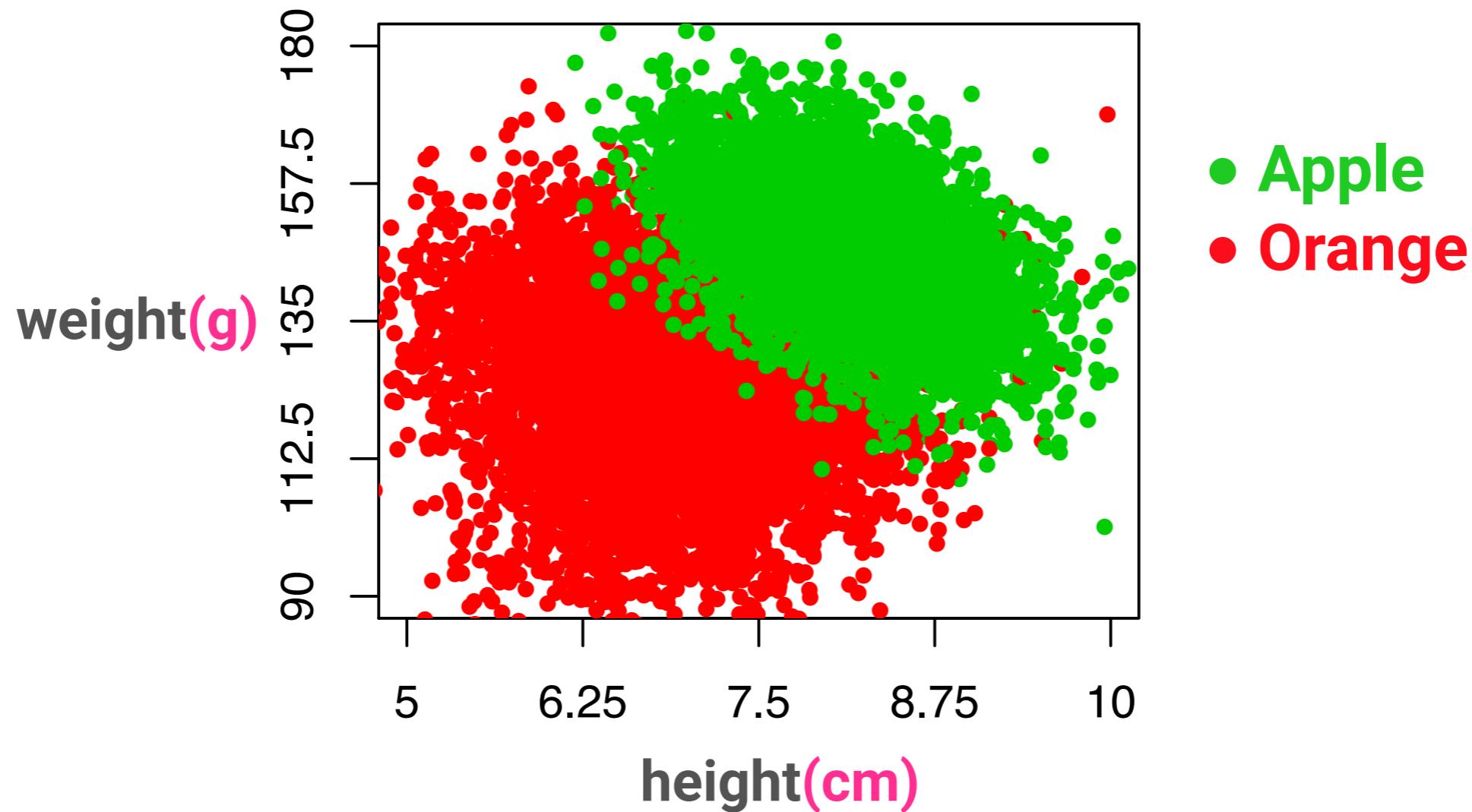
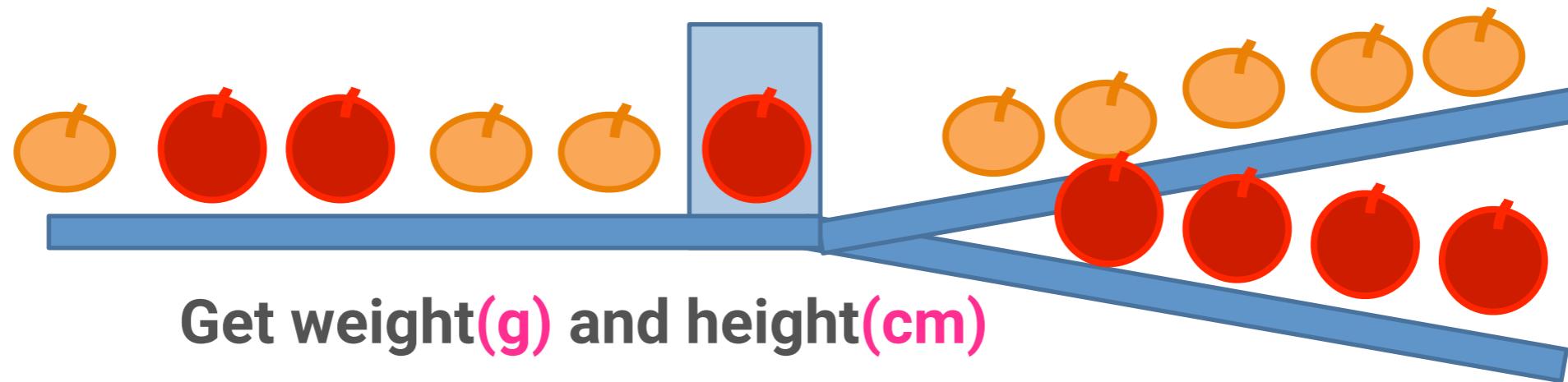
Patterns emerge from many examples?



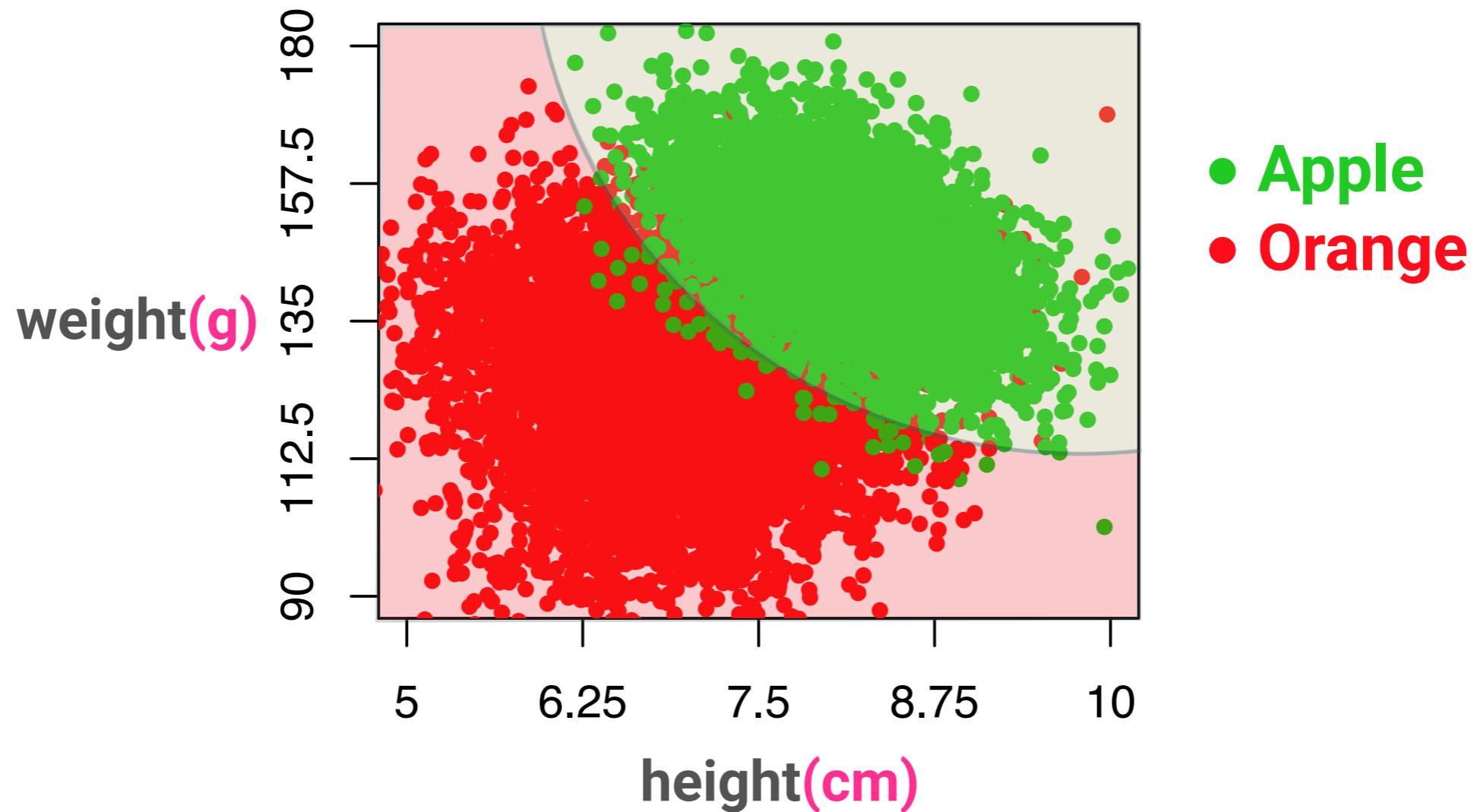
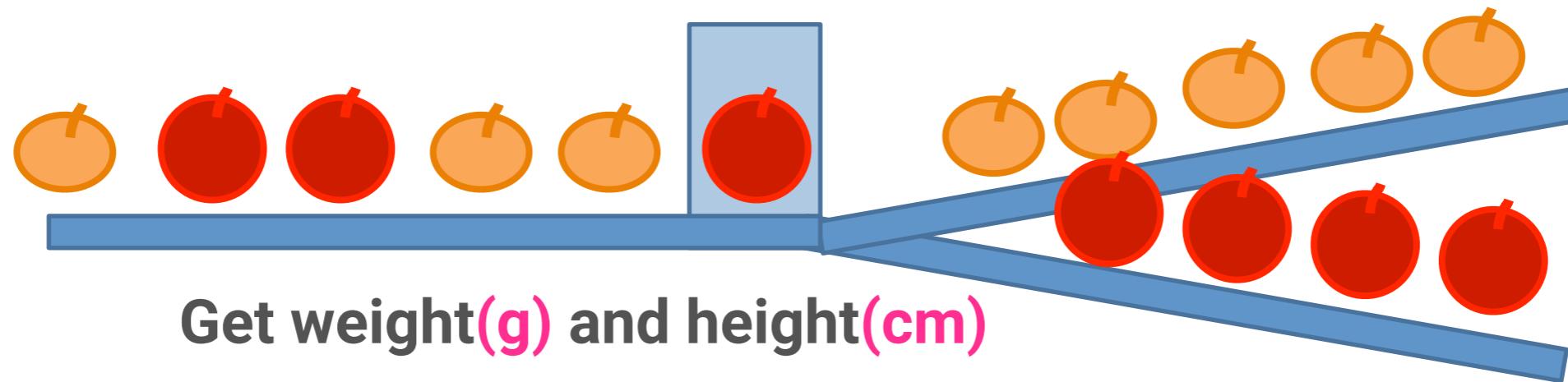
Patterns emerge from many examples?



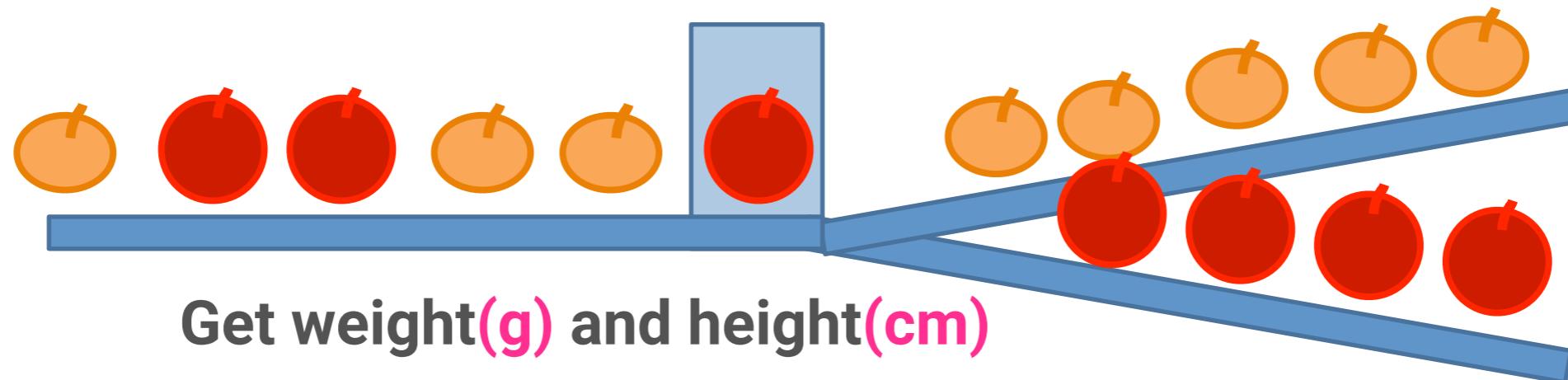
Patterns emerge from many examples?



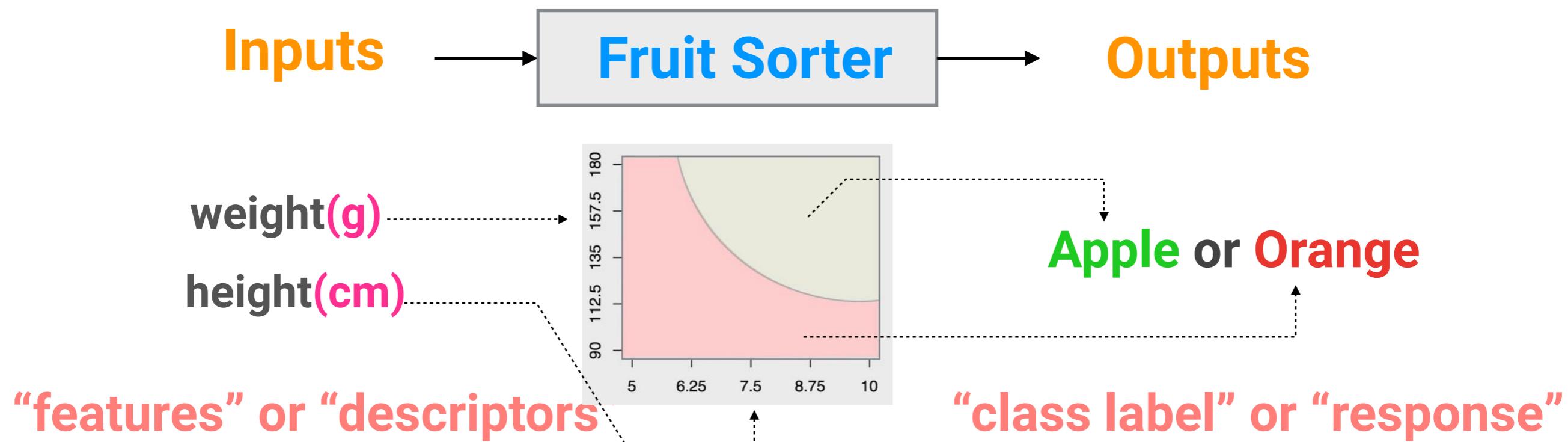
Patterns emerge from many examples?



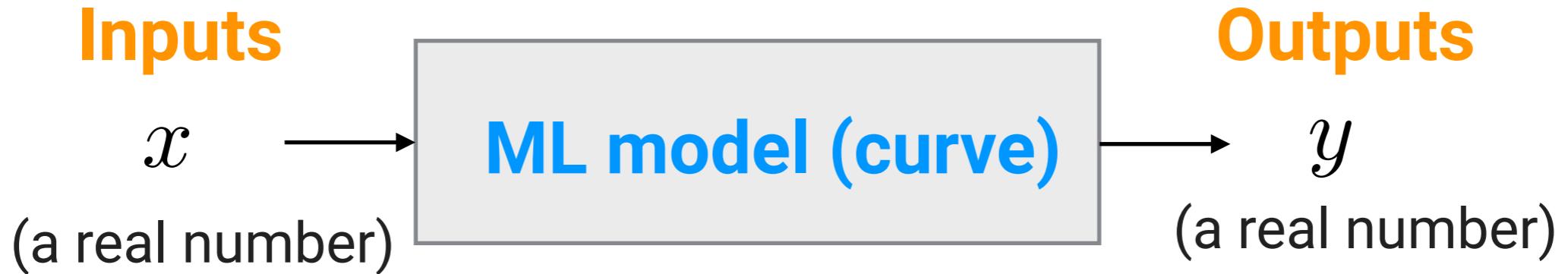
This is all about (supervised) machine learning



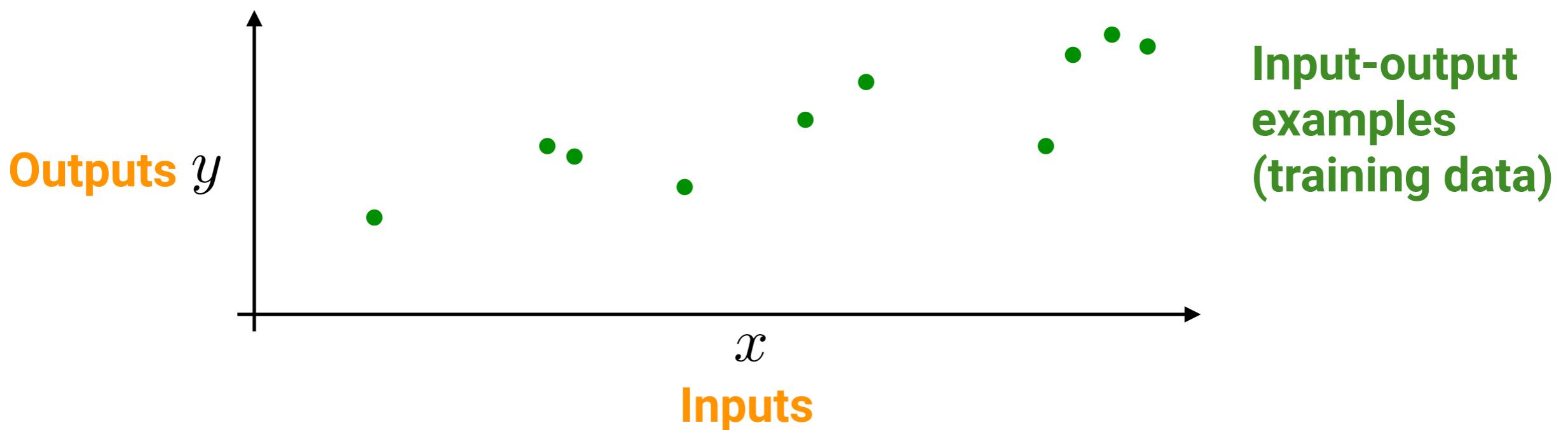
Now we got a computer program for this classification problem.



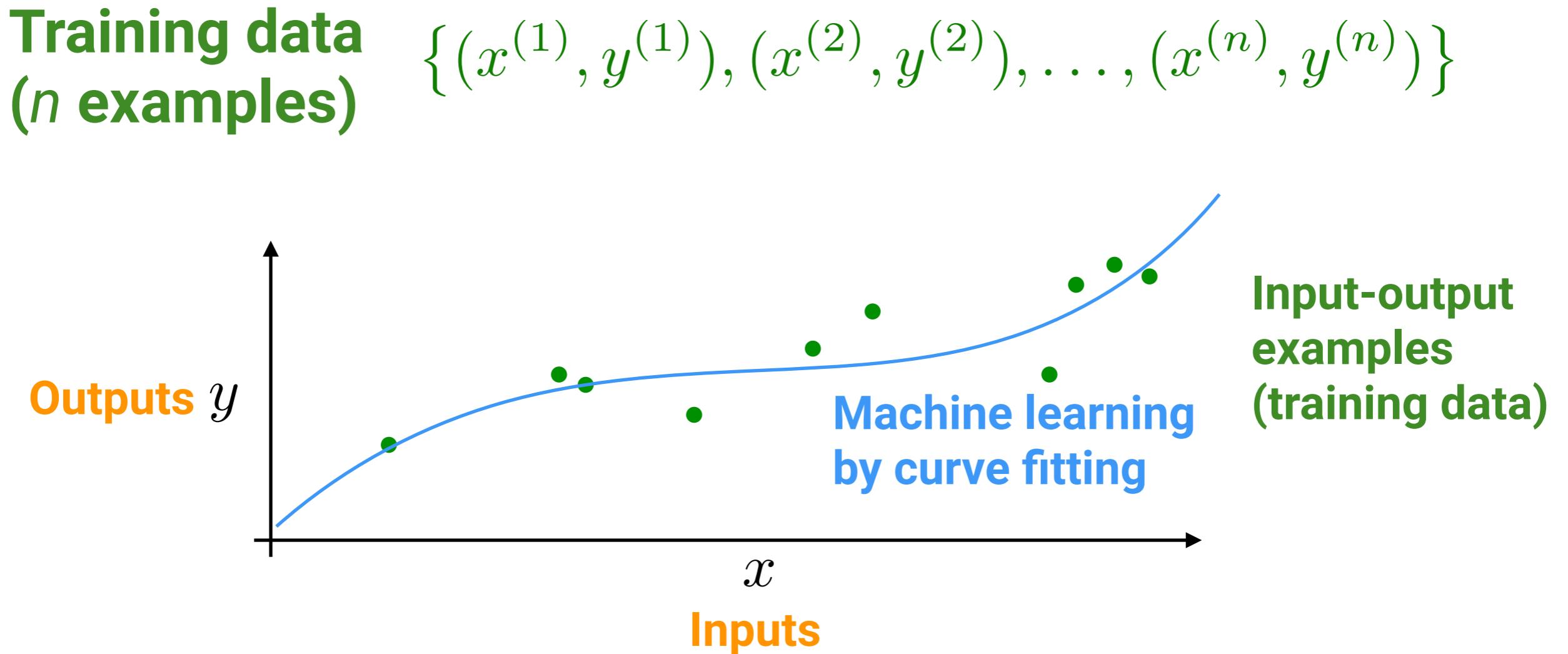
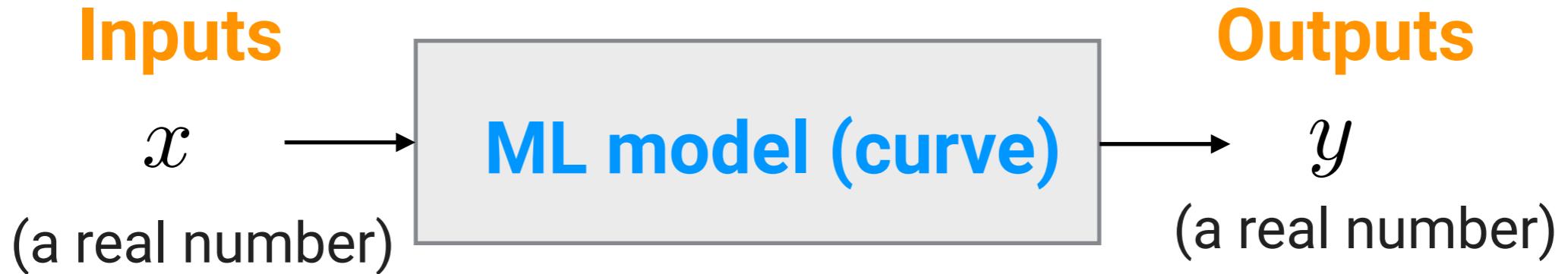
1-dimensional real-number cases = curve fitting



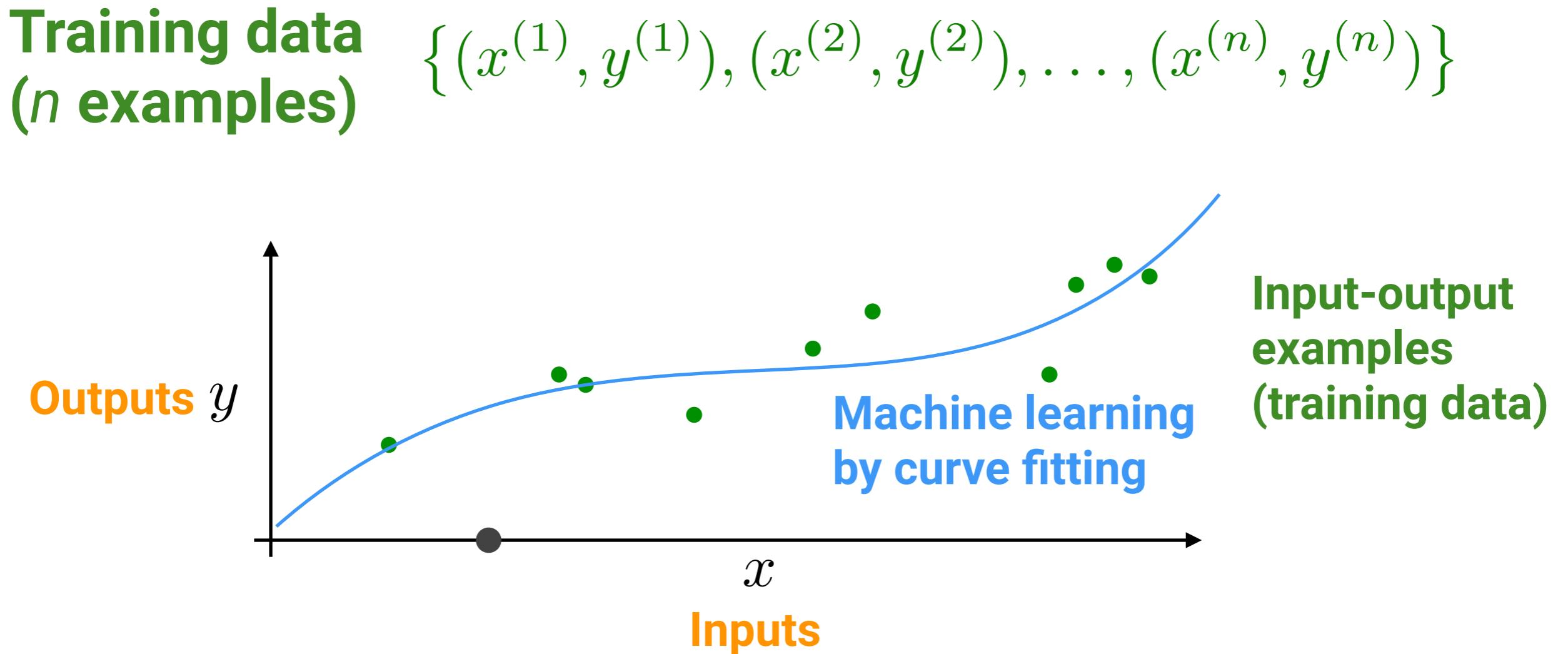
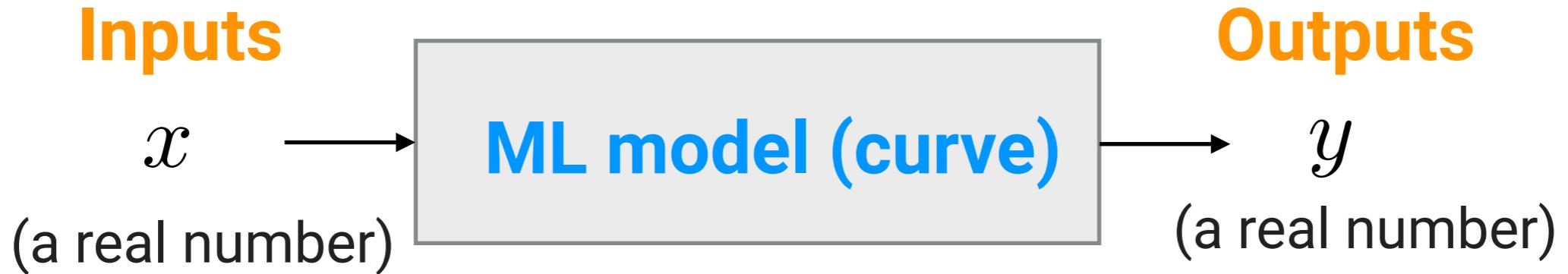
**Training data
(n examples)**

$$\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \dots, (x^{(n)}, y^{(n)})\}$$


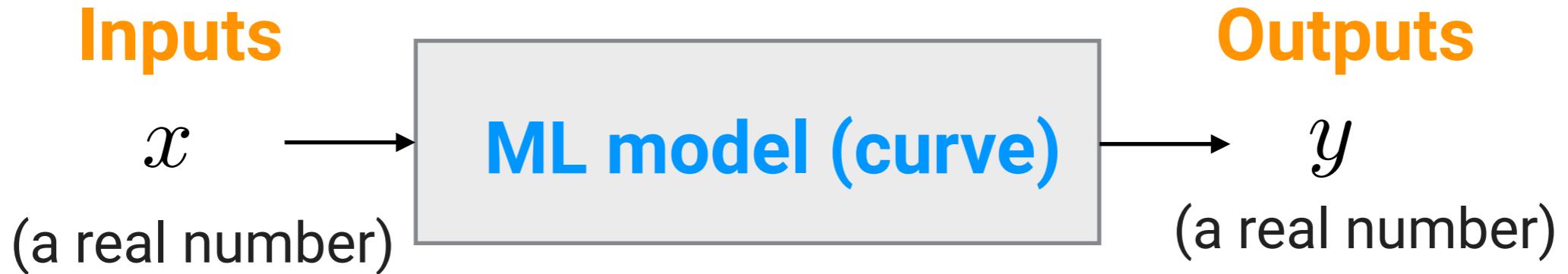
1-dimensional real-number cases = curve fitting



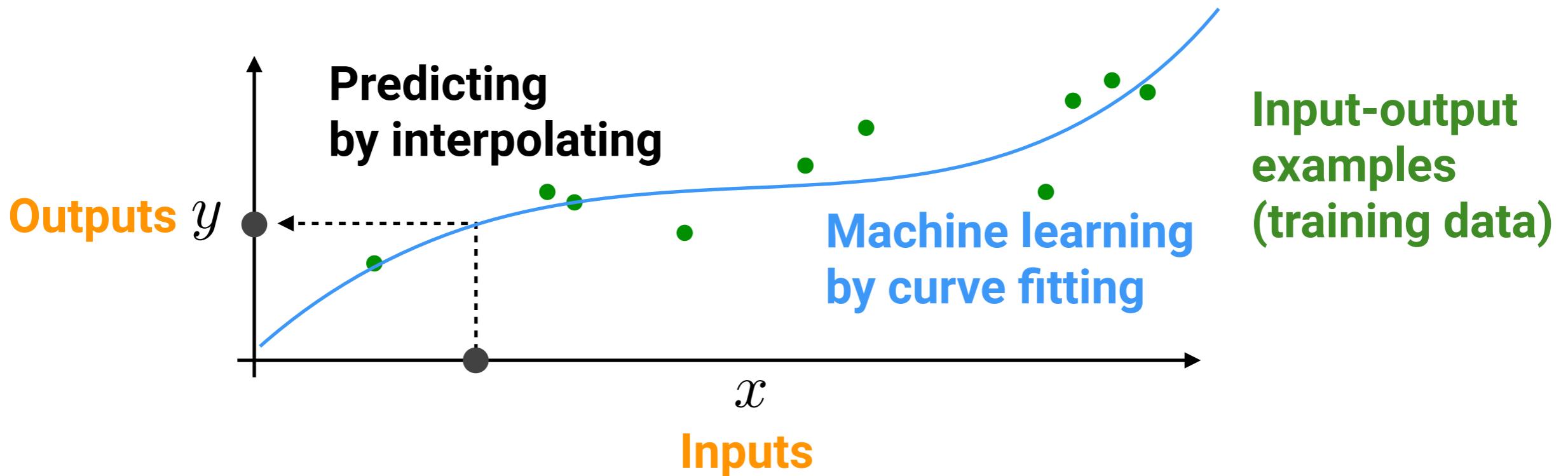
1-dimensional real-number cases = curve fitting



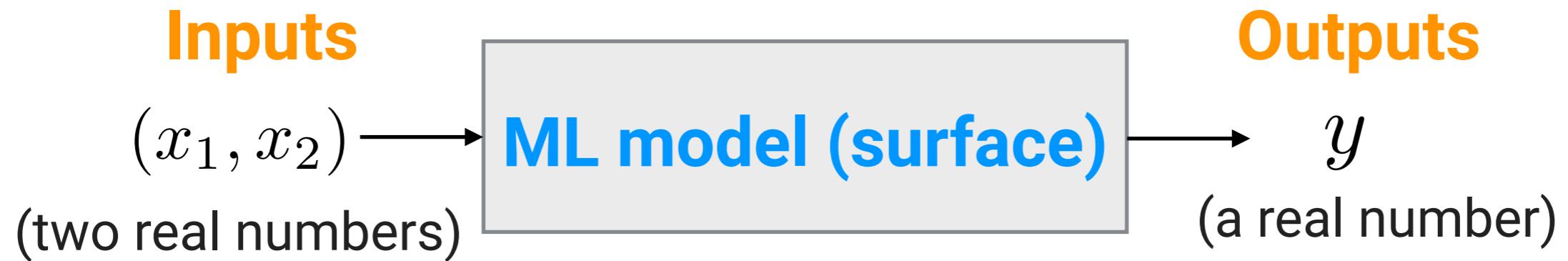
1-dimensional real-number cases = curve fitting



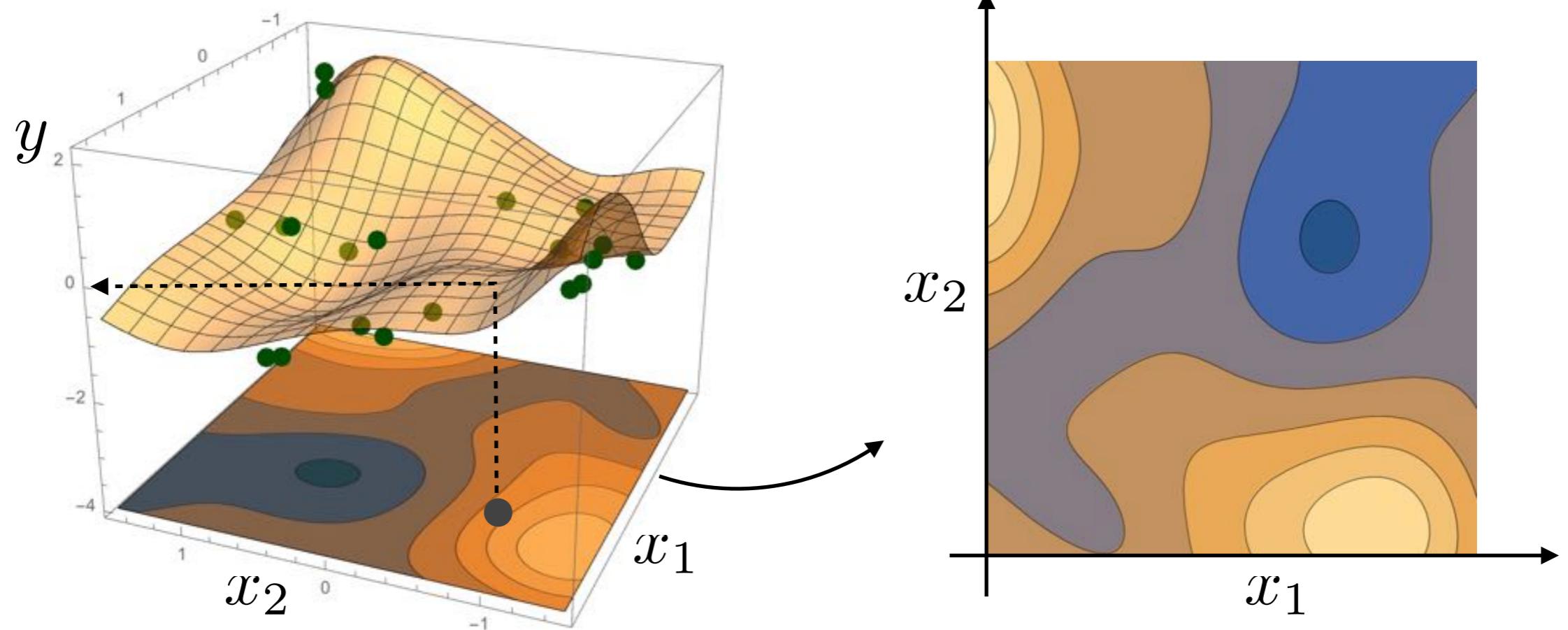
**Training data
(n examples)** $\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \dots, (x^{(n)}, y^{(n)})\}$



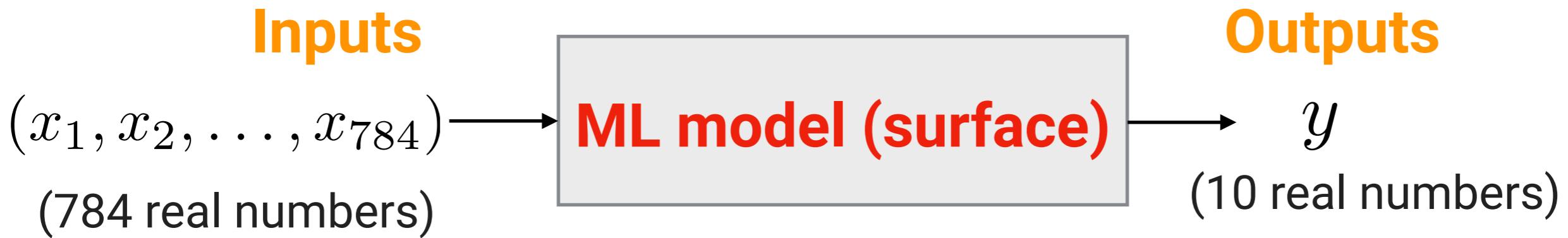
2-dimensional real-number cases = surface fitting



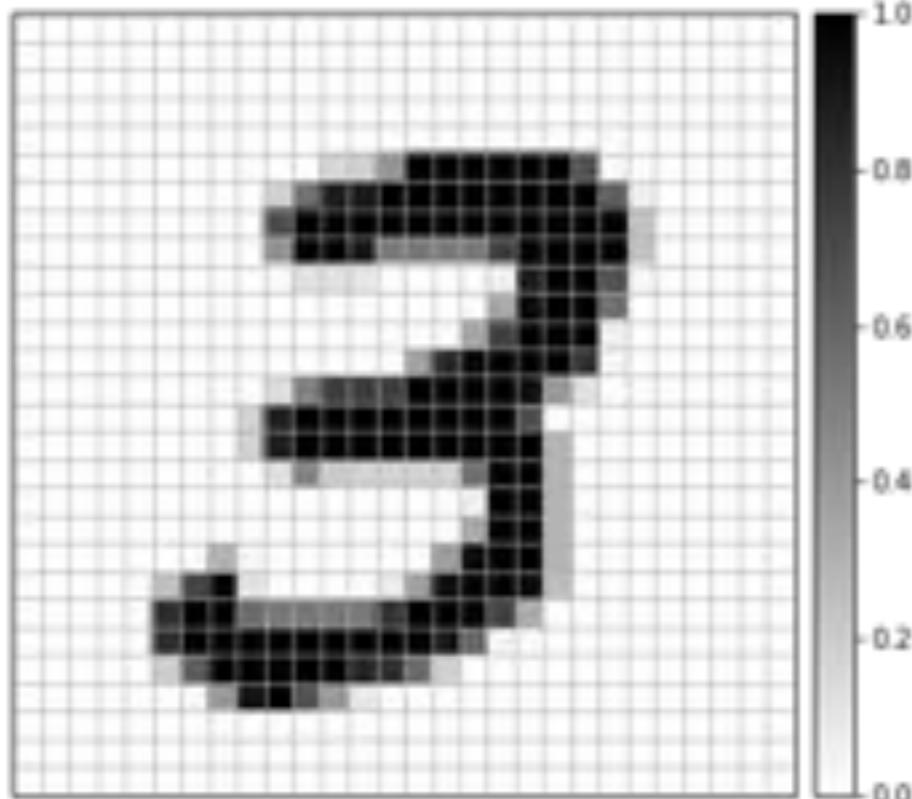
Training data (n examples) $\left\{ \left((x_1^{(1)}, x_2^{(1)}), y^{(1)} \right), \dots, \left((x_1^{(n)}, x_2^{(n)}), y^{(n)} \right) \right\}$



High-dimensional cases = function fitting



An 28 x 28 (=784) pixel image



Probabilities for 0,1,2,...,9

0	1	2	3	4
[0.0, 0.0, 0.0, 0.9 , 0.0,				
5	6	7	8	9
0.0, 0.0, 0.0, 0.1 , 0.0]				

Just fit a 10-dimensional-valued function in 784-dimensional space!

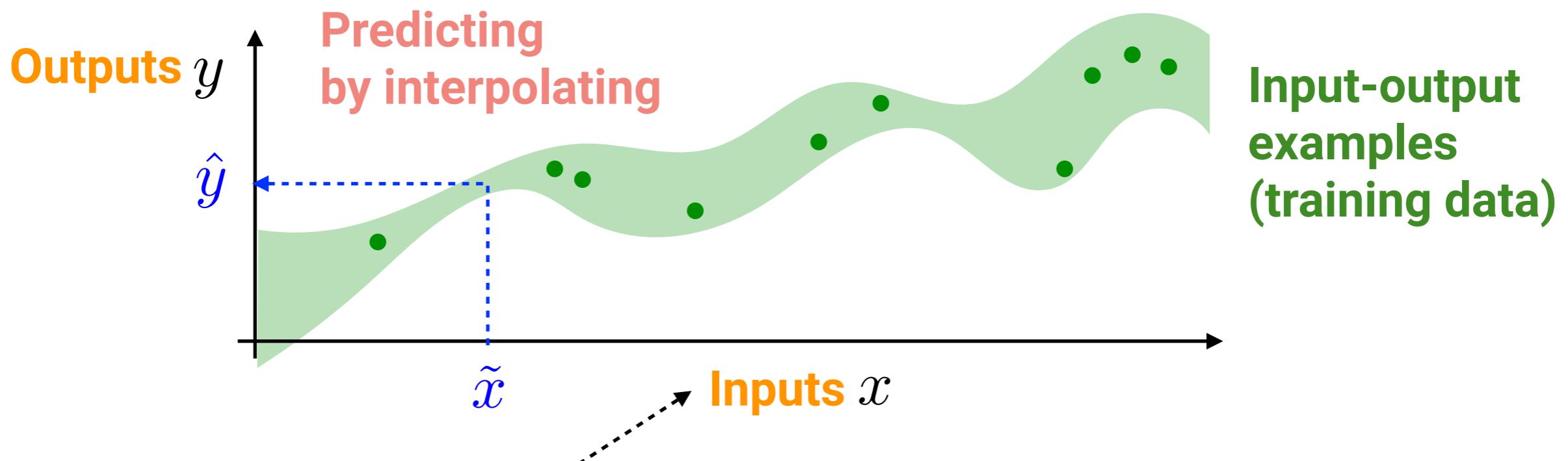
Find a nice mapping $f : \mathbb{R}^{784} \rightarrow \mathbb{R}^{10}$

Machine learning in a nutshell

What ML is doing to tell something valuable about “**data in the future**” from “**data we already have at hand**” is:

High-dimensional interpolation by function fitting

→ any surface/curve model



This can be **very high-dimensional** rather than only 1 dimensional or so in practical situations.

Machine learning in a nutshell

The current “machine learning” usually means

Just an interpolation by curve fitting! 😊

and does **NOT** means

- any human-like flexible and deep thinking/reasoning
- any magical ways to bring something unknowable

Aha! Just an interpolation? I know. It'll be easy! 😌

Unfortunately NO. First of all, curve fitting in a high-dimensional space is not trivial at all, technically speaking. 😢

And many other hard things come out 😞

High-dimensional interpolation is counter-intuitive

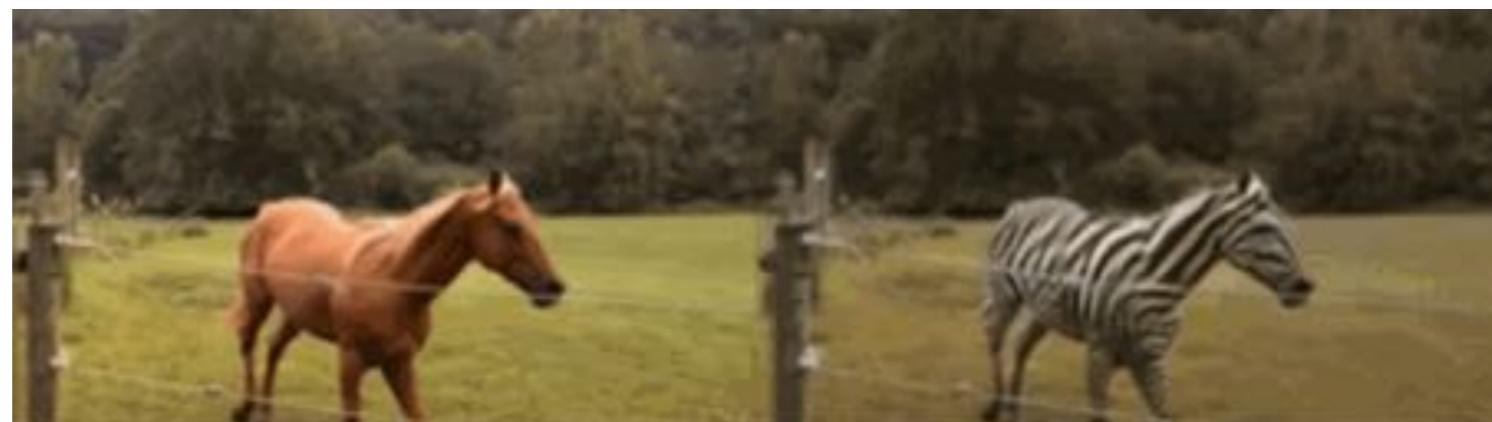
pix2pix



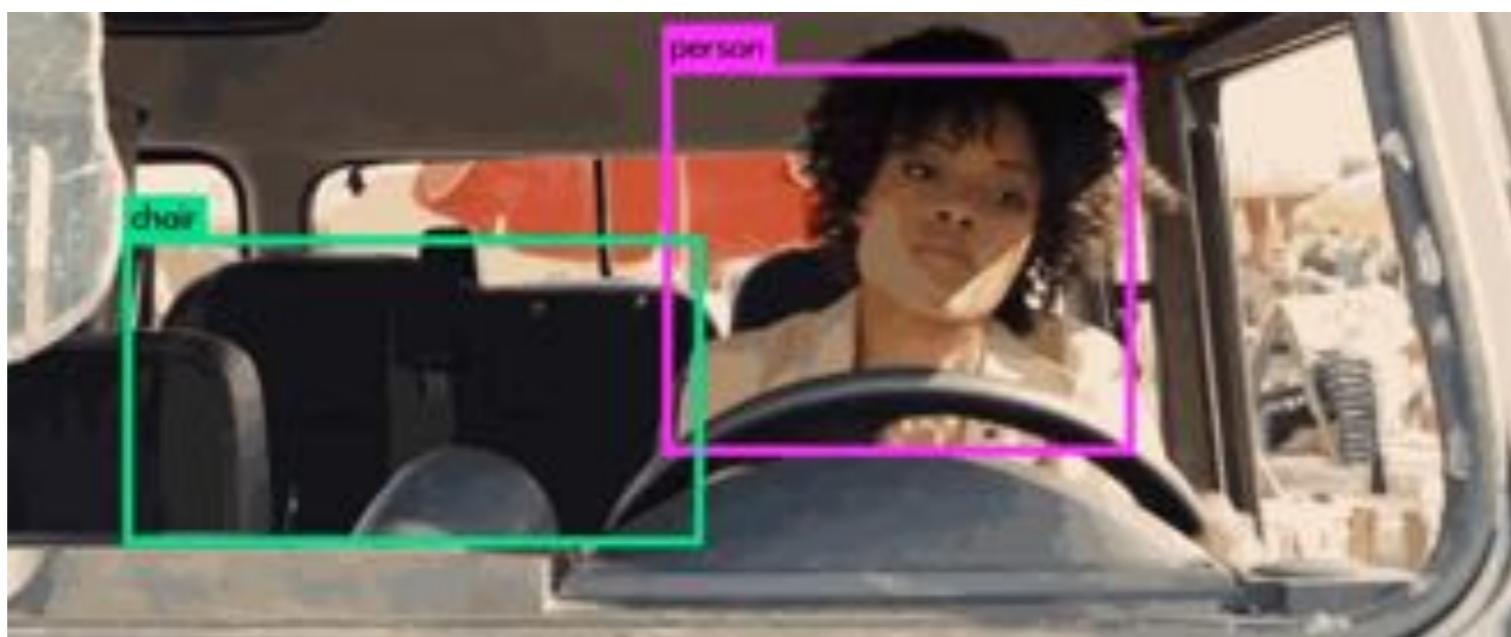
Face swapping (e.g. DeepFake)



CycleGAN



YOLO



Even for machine-learning or AI experts!

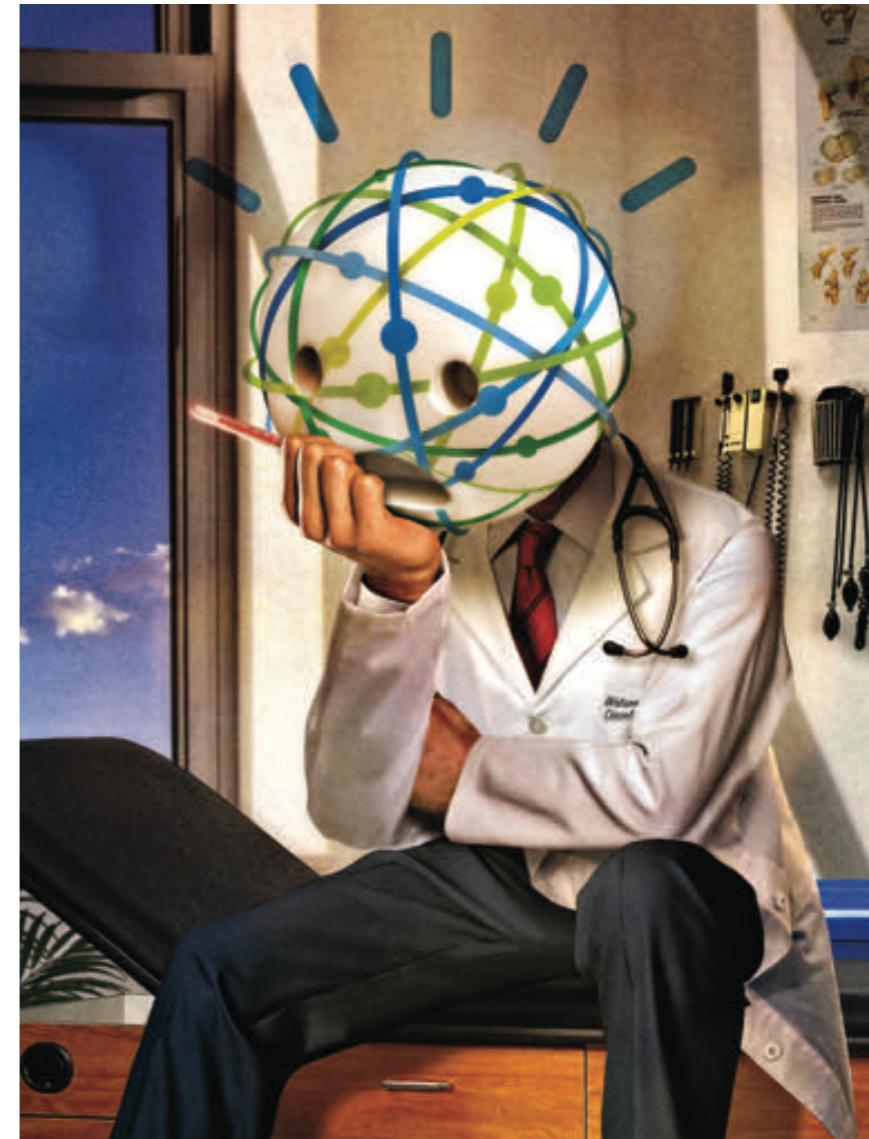
Sales teams of high-tech companies sometimes set an unnecessarily high hurdle without knowing what is actually going on...

IEEE Spectrum, 2019 Apr

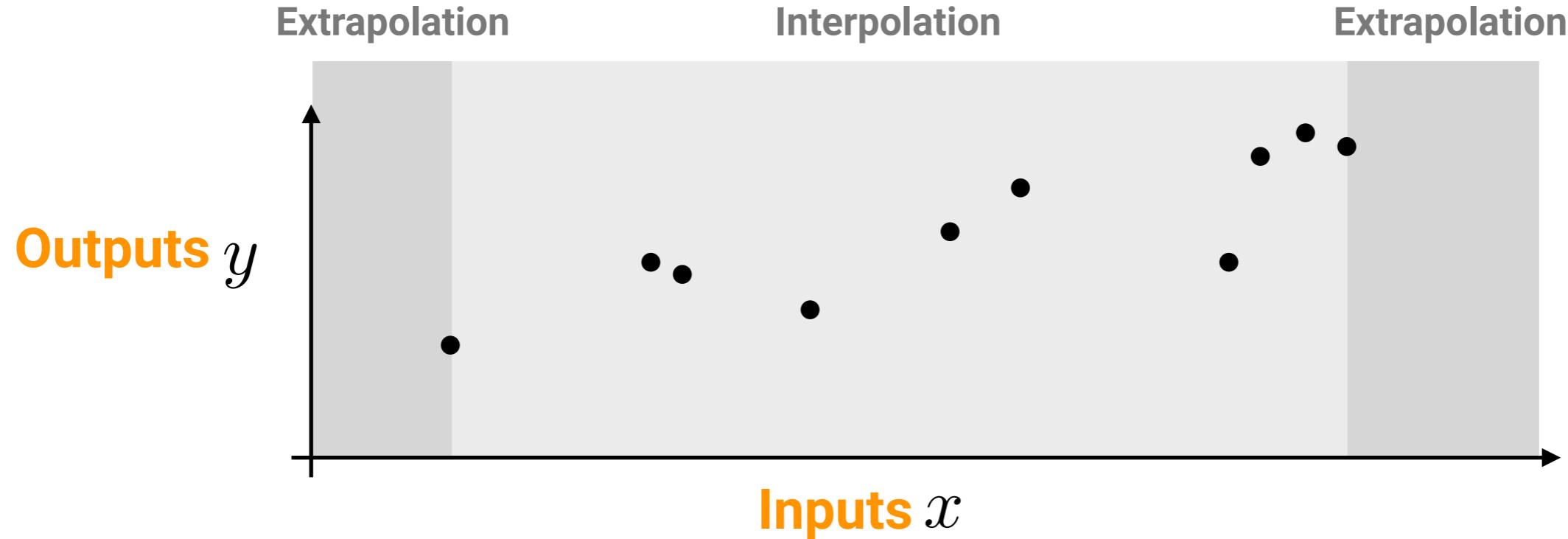
IBM Watson, *Heal Thyself*

How IBM overpromised and
underdelivered on AI health care

By ELIZA STRICKLAND ILLUSTRATIONS BY EDDIE GUY

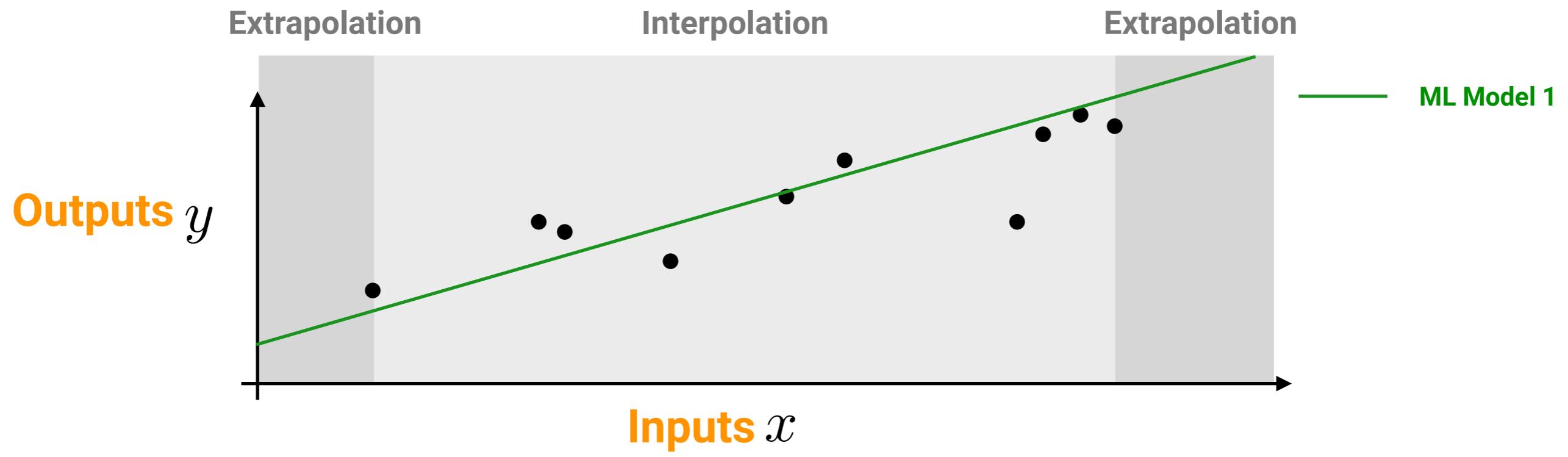


Why ML looks so complicated?

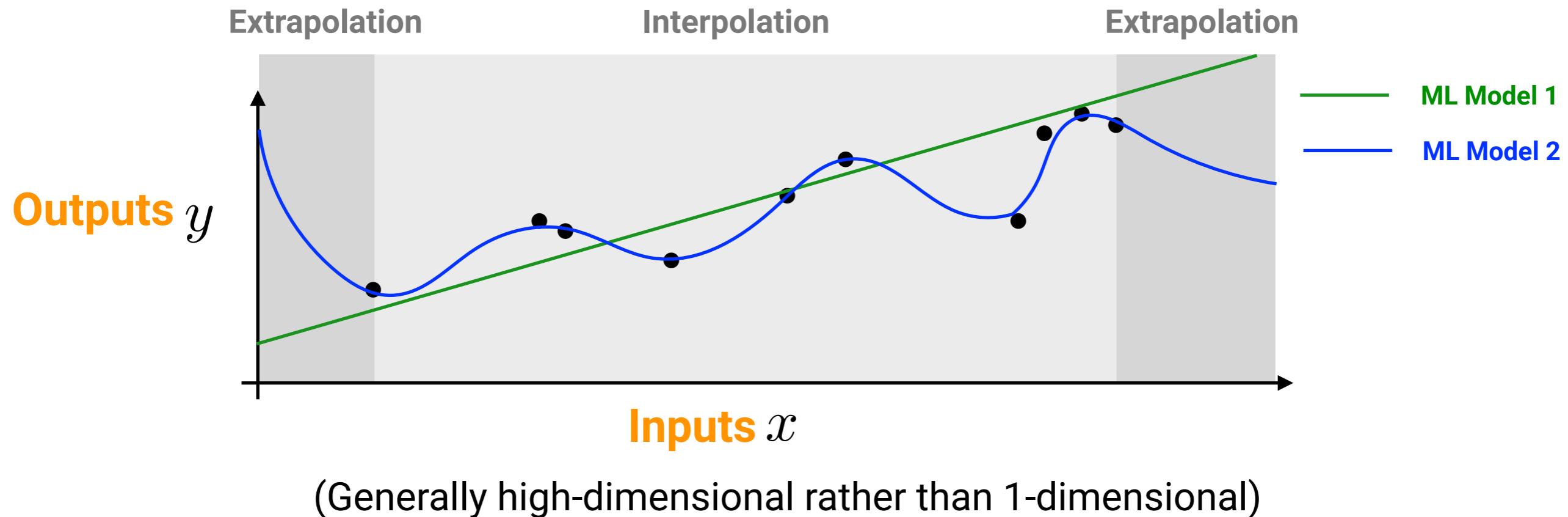


(Generally high-dimensional rather than 1-dimensional)

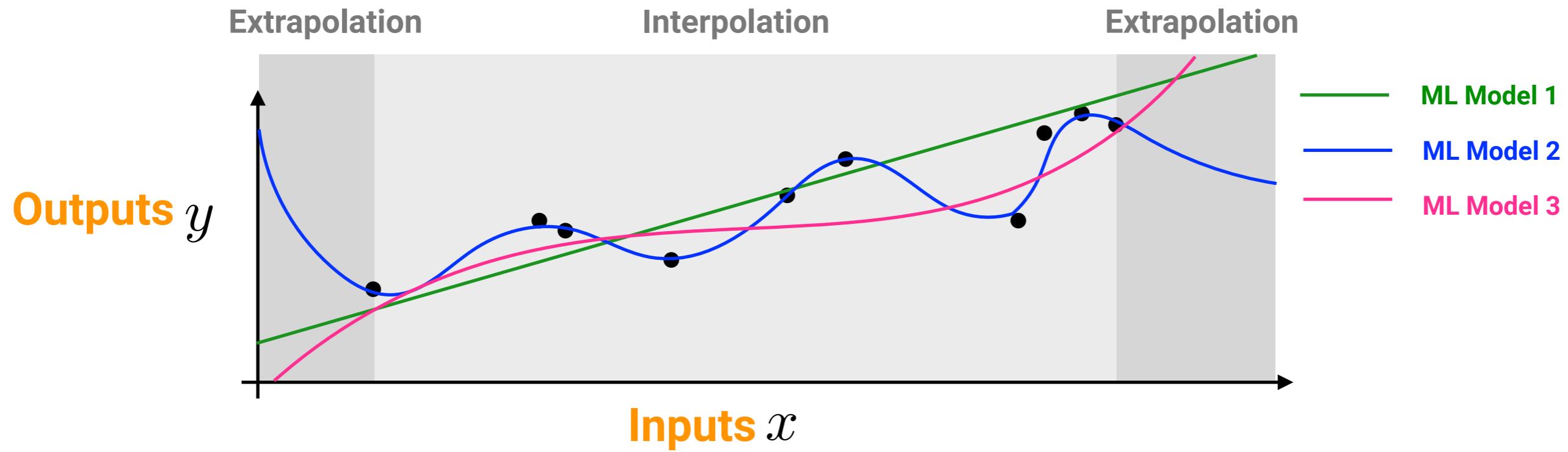
Why ML looks so complicated?



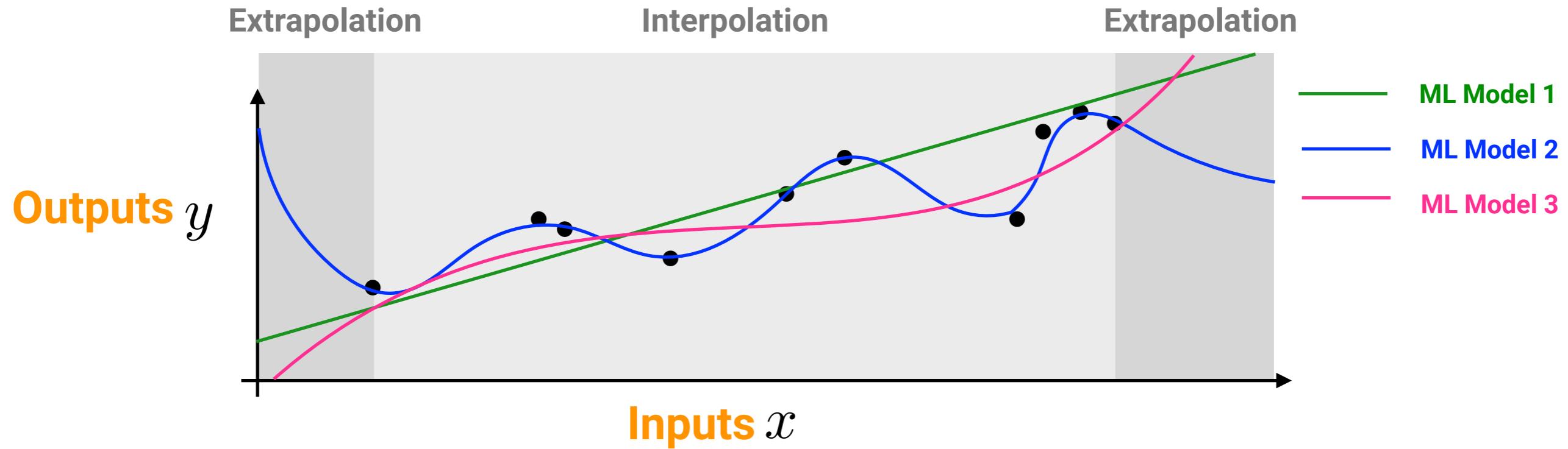
Why ML looks so complicated?



Why ML looks so complicated?

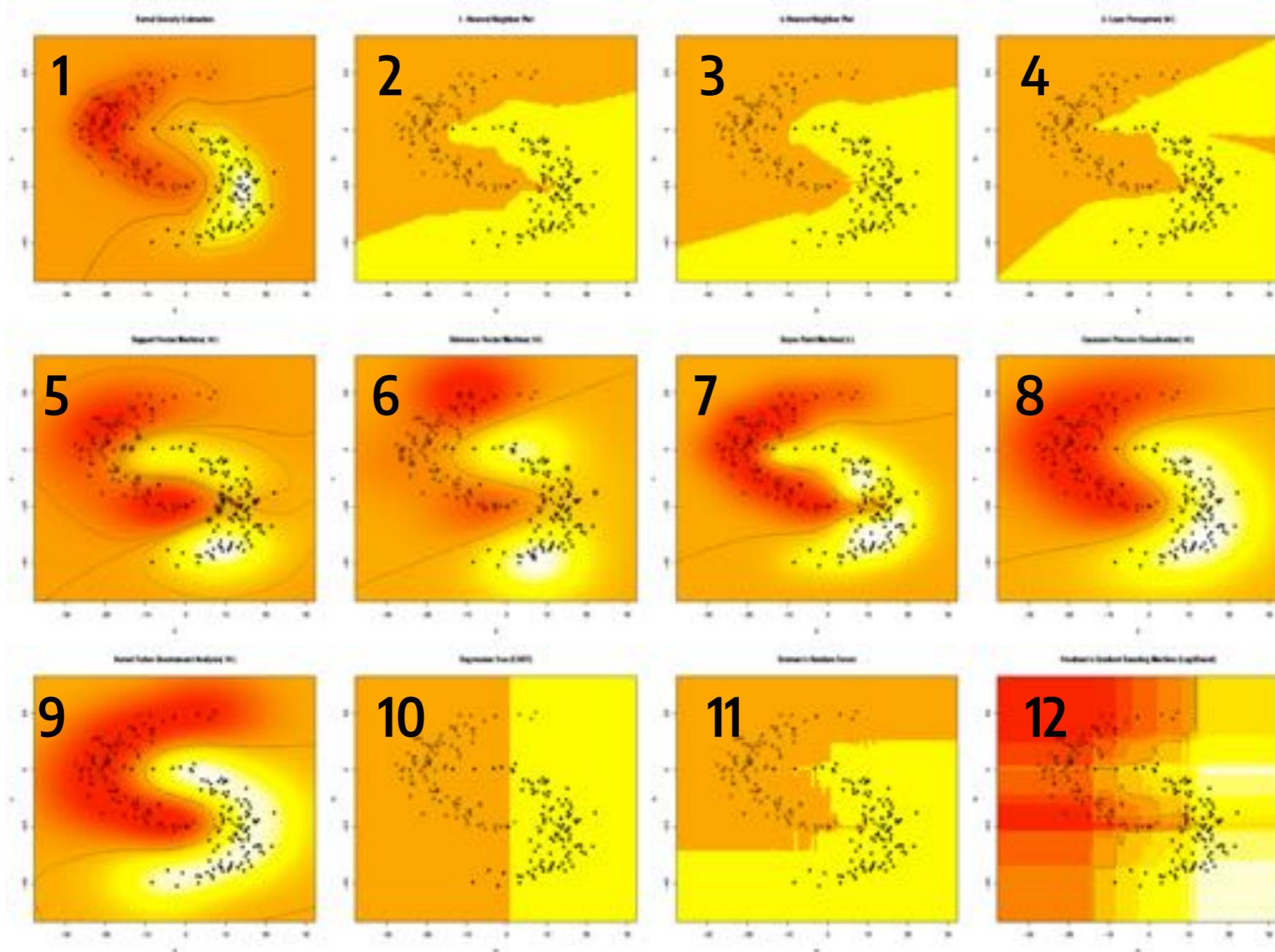


Why ML looks so complicated?



- Many ways (ML models) exist for “curves (surfaces)”
- What models and input representations work best depends on the target problem
- We need to choose appropriate ML models for each given dataset and target problem

Many ML models (Deep learning is one of them)



1. Plugin Bayes Classifier
2. 1-Nearest Neighbor Method
3. 5-Nearest Neighbor Method
4. 3-Layer Neural Networks
5. Support Vector Machine
6. Relevance Vector Machine
7. Bayes Point Machine
8. Gaussian Process Classifier
9. Kernel Discriminant Analysis
10. Regression Tree (CART)
11. Random Forest
12. Gradient Boosting Machine

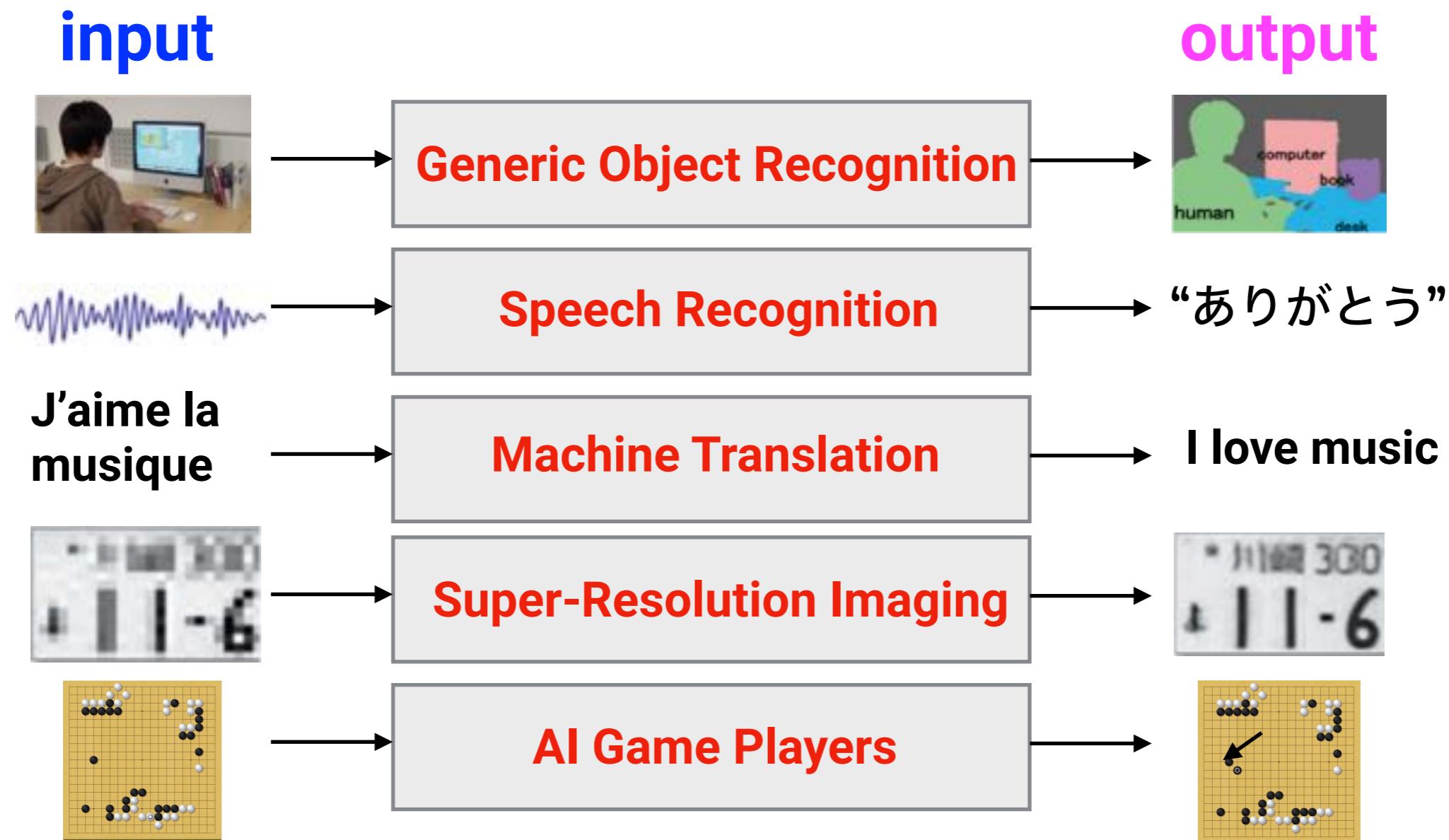
Many ML models (and problems)

Machine Learning Landscape

Supervised Learning	Unsupervised Learning	Others	
Classification [Linear classification] - Logistic / Softmax regression - Linear discriminant analysis - Naïve Bayes classifiers - Perceptron - Linear Support Vector Machines (SVM) [Nonlinear classification] - k-nearest neighbor classifiers - Decision trees (Classification trees) - Polynomial classifiers / Factorization machines - Tree ensemble classifiers - Random Forest classifiers - Extra Trees classifiers - Gradient Boosted Decision Trees (GBDT) - Kernel method classifiers - Support Vector Machines (SVM) - Gaussian process classifiers - Neural network (Deep learning) classifiers - Multi-layer perceptrons (MLP) - Convolutional networks (CNN) - VGG (OxfordNet) - Inception (GoogLeNet) - ResNet / ResNeXt - DenseNet - Recurrent networks (RNN)	Regression [Linear regression] - Least squares regression - Principal component regression - Partial Least Squares (PLS) regression - Penalized linear regression - LASSO regression (L1-penalized) - Ridge regression (L2-penalized) - ElasticNet regression (L1 & L2-penalized) [Nonlinear regression] - k-nearest neighbor regressors - Decision trees (Regression trees, Model trees) - Polynomial regressors / Factorization machines - Tree ensemble regressors - Random Forest regressors - Extra Trees regressors - Gradient Boosted Regression Trees (GBRT) - Kernel method regressors - Support Vector Regression (SVR) - Kernel Ridge Regression - Gaussian process regressors - Neural network (Deep learning) regressors - Multi-layer perceptrons (MLP) - Convolutional networks (CNN) - VGG (OxfordNet) - Inception (GoogLeNet) - ResNet / ResNeXt - DenseNet - Recurrent networks (RNN)	Clustering - k-means - Hierarchical clustering - Gaussian mixtures - Spectral methods - DBSCAN Decomposition - Principal component analysis (PCA) - Independent component analysis (ICA) - Canonical correlation analysis (CCA) - Nonnegative matrix factorization (NMF) - Latent Dirichlet allocation (LDA) Manifold learning - Multidimensional scaling (MDS) - Self-organizing maps (SOM) - Isomap - Locally linear embedding (LLE) - Spectral embedding (Laplacian eigenmaps) - t-distributed Stochastic Neighbor Embedding (t-SNE) - Autoencoders Density estimation	Semi supervised learning Ranking Transfer learning K-shot learning Domain adaptation Multitask learning Reinforcement learning Active learning Model-based optimization Time series/Sequence models Probabilistic inference (Bayesian, Generative, Graphical) Causal inference Online/Incremental learning Anomaly/Outlier detection Ensemble learning Relational/Network learning Representation learning Structured prediction Meta Learning :

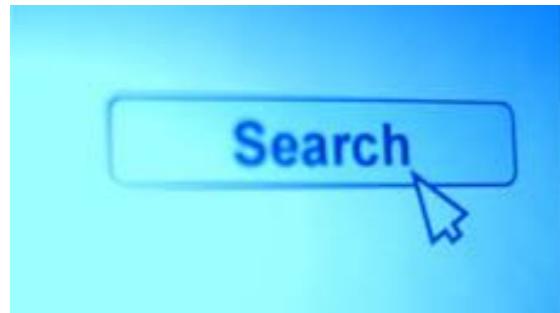
(Supervised) machine learning in other words

Machine learning is a systematic way to find a high-dimensional mapping from **input** to **output** just by giving a lot of **input-output** "examples".



This lazy idea really works in diverse applications

Search Engine



Translation



Self-driving



Medicine



Advertising



Transportation



Weather



Security



Smart devices, IoT, e-Commerce, Manufacturing, Agriculture, Disaster Prevention, Finance, Education, Employment, Matchmaking, and, of course, Science.

Aug 26: 10:30~12:00 (90min)

1. What is "machine learning"?
2. Why does it matter to chemists?
3. Let's try it in your browser (with no setup!)

Aug 26: 13:00~14:30 (90min)

4. Five things all beginners should know
 - "The quality of your inputs decide the quality of your output"
 - Training / validation / test data
 - Tuning hyperparameters
 - Identification and design of input variables (or "descriptors")
 - "Correlation does not imply causation"
5. Standard pipeline and deep learning
6. Current efforts and future directions

Empirical optimization or "Edisonian empiricism"



Problem: time and cost when we use this for all possible candidates



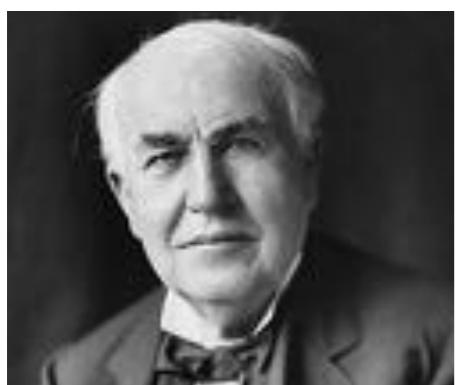
feedback to next plans

available data and findings

Generate hypotheses

- Experiments
- Simulations

Check and validate results



- Genius is 1% inspiration and 99% perspiration.
- There is no substitute for hard work.
- I have not failed. I've just found 10,000 ways that won't work.
- :



Thomas Edison

Experience + Intuition = Incredible hardwork + Luck



Empirical optimization or "Edisonian empiricism"



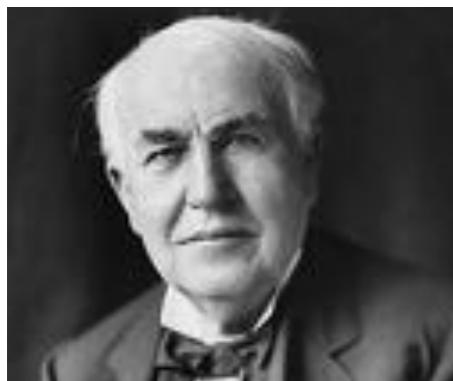
Problem: time and cost when we use this for all possible candidates



feedback to next plans

available data and findings

Gen
hy



Thomas Edison

But, wait. Is this still valid now in 2020?

Everything looks complicated than Edison thought, and how "incredibly hard" work do I need...?

that won't work.
:



Experience + Intuition = Incredible hardwork + Luck

<http://www.fourthparadigm.org/>

by Tony Hey, Stewart Tansley, Kristin Tolle



Tony Hey



In *The Fourth Paradigm: Data-Intensive Scientific Discovery*, the collection of essays expands on the vision of pioneering computer scientist Jim Gray for a new, fourth paradigm of discovery based on data-intensive science and offers insights into how it can be fully realized.

Jim Gray on "eScience"

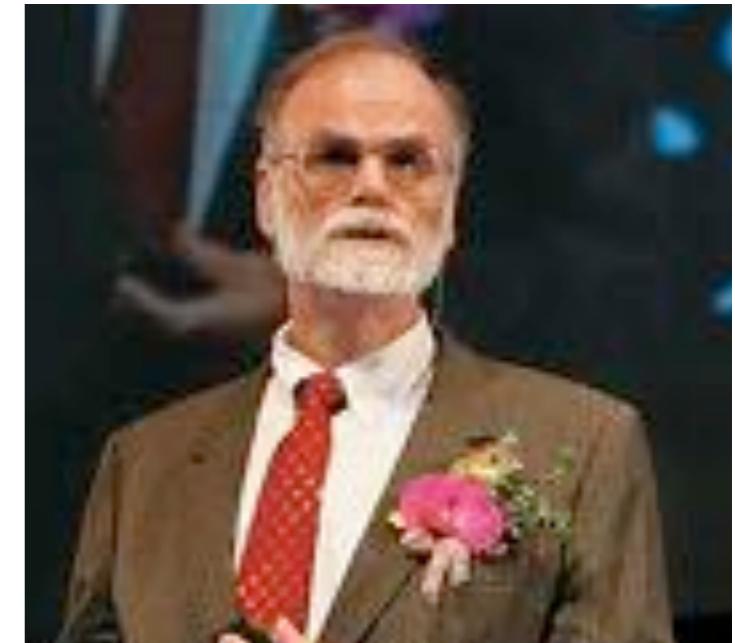
<http://research.microsoft.com/en-us/um/people/gray/JimGrayTalks.htm>

"eScience" Talk at NRC-CSTB meeting

Mountain View CA, 11 January 2007.

His 'last' talk before the disappearance

On January 28, 2007 he failed to return from a short solo trip to the Farallon Islands near San Francisco to scatter his mother's ashes.



Jim Gray

NRC = National Research Council

<http://sites.nationalacademies.org/NRC/index.htm>;

CSTB = Computer Science and Telecom- munications Board

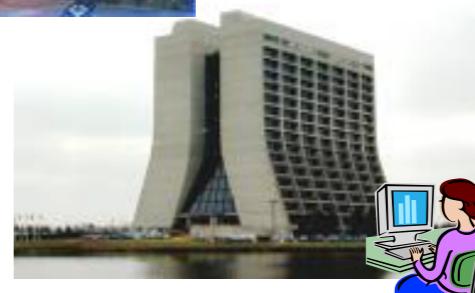
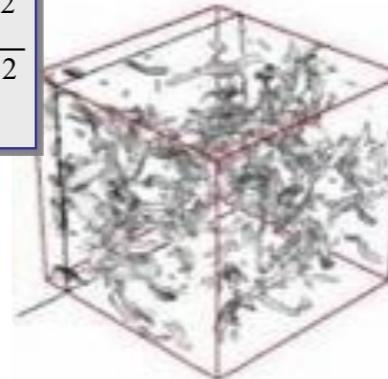
<http://sites.nationalacademies.org/cstb/index.htm>.

Science Paradigms

- Thousand years ago:
science was empirical
describing natural phenomena
- Last few hundred years:
theoretical branch
using models, generalizations
- Last few decades:
a computational branch
simulating complex phenomena
- Today:
data exploration (eScience)
unify theory, experiment, and simulation
 - Data captured by instruments
Or generated by simulator
 - Processed by software
 - Information/Knowledge stored in computer
 - Scientist analyzes database / files
using data management and statistics

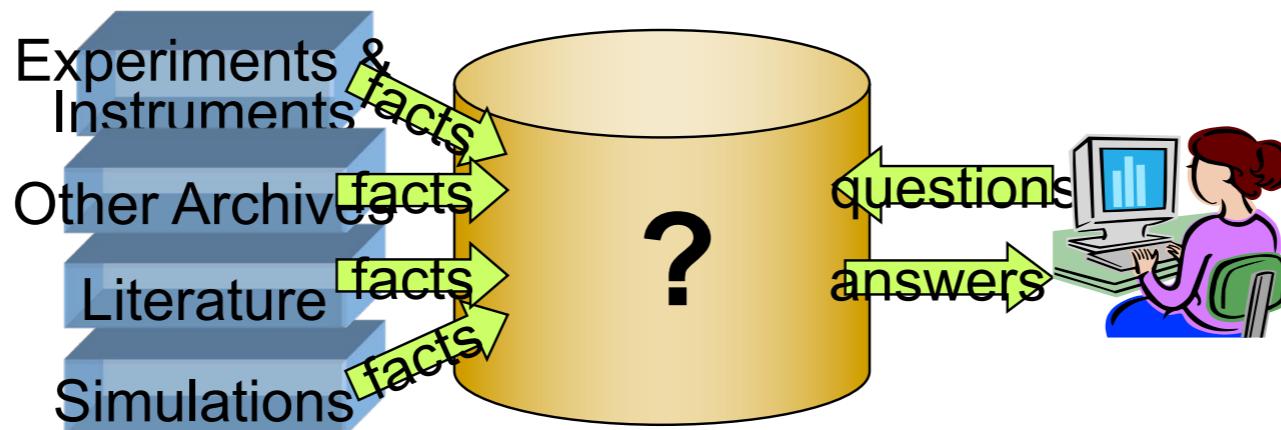


$$\left(\frac{\dot{a}}{a}\right)^2 = \frac{4\pi G\rho}{3} - K \frac{c^2}{a^2}$$



X-Info

- The evolution of X-Info and Comp-X
for each discipline X
- How to codify and represent our knowledge



The Generic Problems

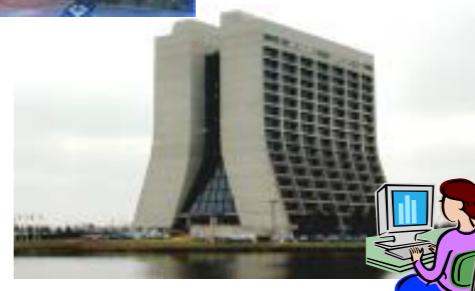
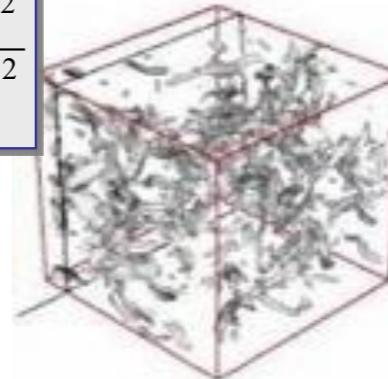
- Data ingest
- Managing a petabyte
- Common schema
- How to organize it
- How to reorganize it
- How to share with others
- Query and Vis tools
- Building and executing models
- Integrating data and Literature
- Documenting experiments
- Curation and long-term preservation

Science Paradigms

- Thousand years ago:
science was empirical
describing natural phenomena
- Last few hundred years:
theoretical branch
using models, generalizations
- Last few decades:
a computational branch
simulating complex phenomena
- Today:
data exploration (eScience)
unify theory, experiment, and simulation
 - Data captured by instruments
Or generated by simulator
 - Processed by software
 - Information/Knowledge stored in computer
 - Scientist analyzes database / files
using data management and statistics



$$\left(\frac{\dot{a}}{a}\right)^2 = \frac{4\pi G\rho}{3} - K \frac{c^2}{a^2}$$

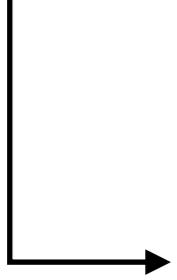


- Today:

data exploration (eScience)

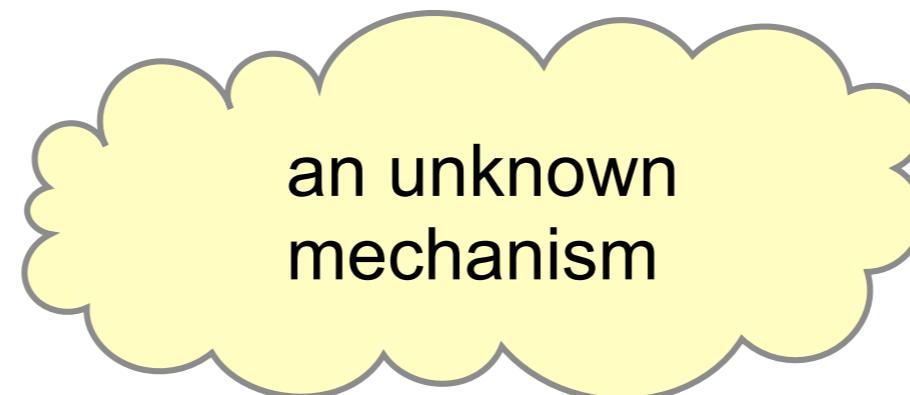
unify theory, experiment, and simulation

- Data captured by instruments
Or generated by simulator
- Processed by software
- Information/Knowledge stored in computer
- Scientist analyzes database / files
using data management and statistics

 **This would be more strongly complemented by
Machine learning or AI, or data sciences.**

The interest of science

Related factors
and their states



observations
(data)

• Theory-driven (rational, deductive) approach

Related factors
and their states

An explicit model
from first principles

observations
(data)

Ideally, the smallest number
of dominant factors

Whether or not theory is correct can be validated

• Data-driven (empirical, inductive) approach

Related factors
and their states

A versatile unfixed function
that can take a variety of forms
by tuning parameters

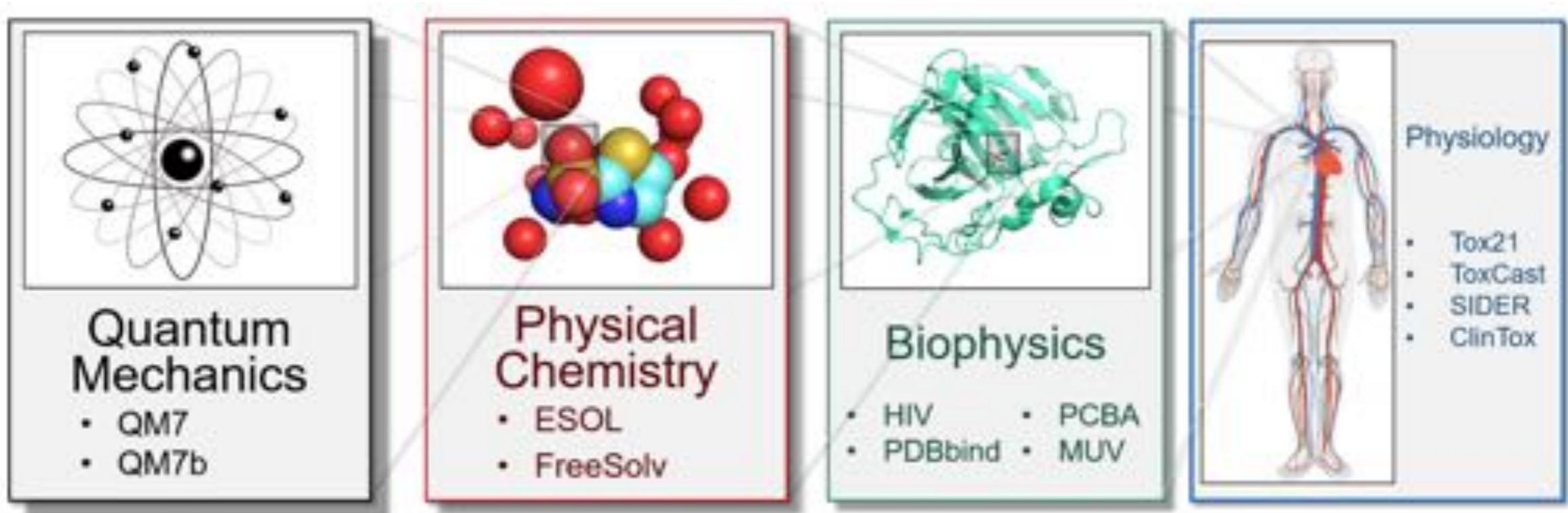
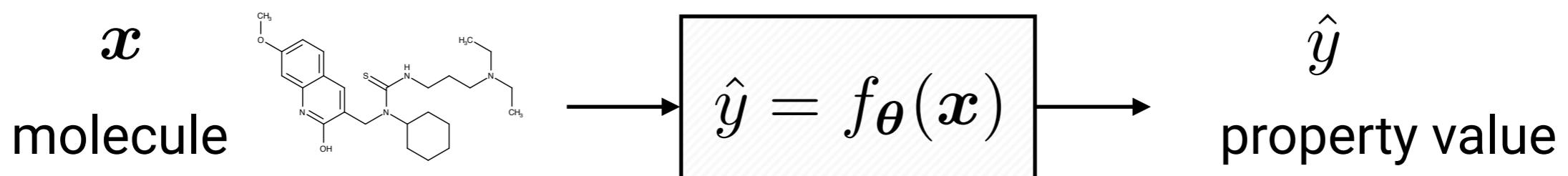
observations
(data)

All potentially related factors
(the number can be very large
as long as they are sensible)

Function is best fitted to data by tuning parameters

Predictive modeling by machine learning

Quantitative structure–activity/property relationship (QSAR/QSPR)



MoleculeNet: A Benchmark for Molecular Machine Learning

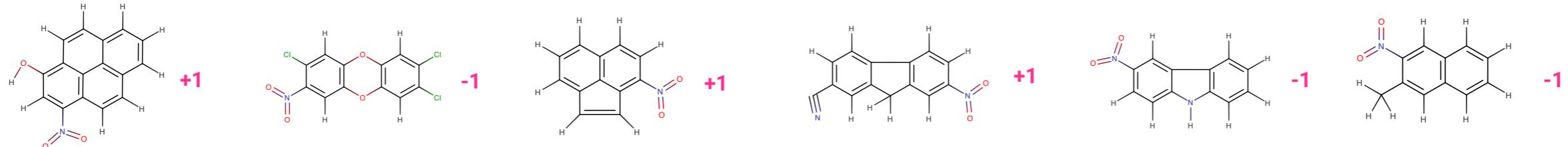
<https://arxiv.org/abs/1703.00564>

<https://github.com/deepchem/deepchem> (<https://deepchem.io/>)

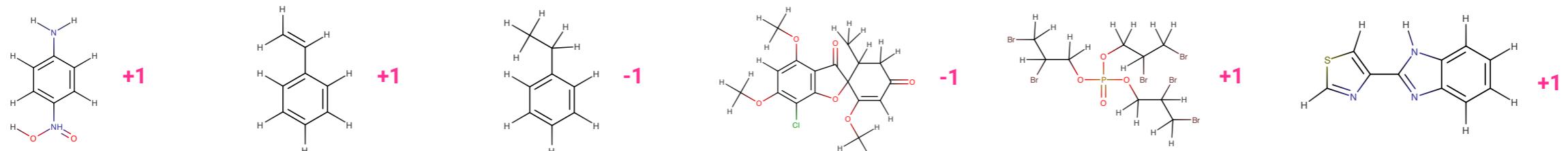


Many levels of interest

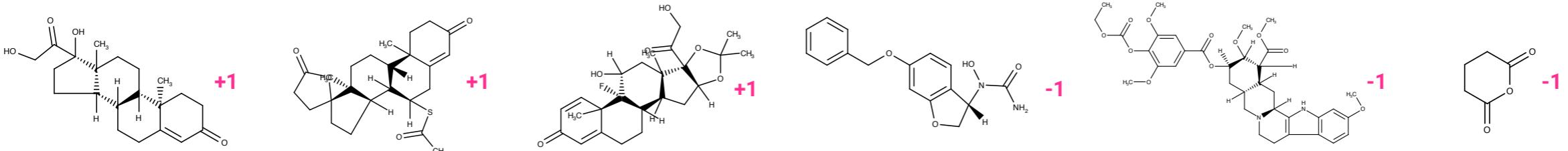
- Mutagenic potency



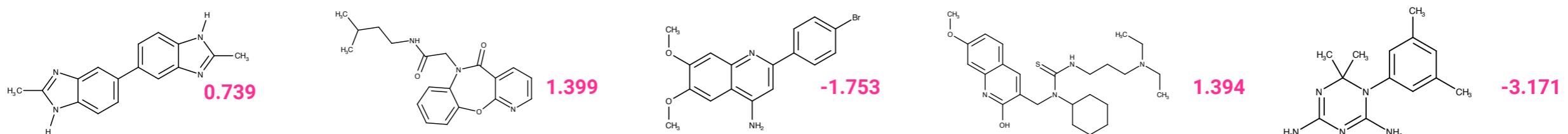
- Carcinogenic potency



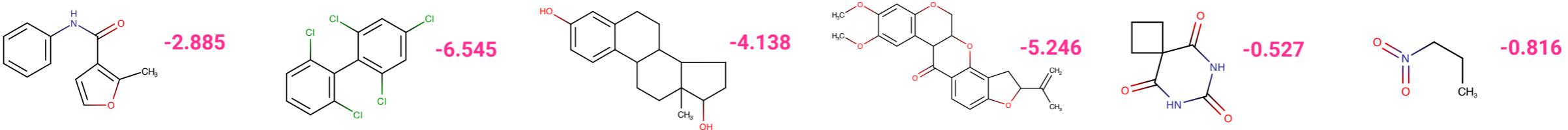
- Endocrine disruption



- Growth inhibition



- Aqueous solubility

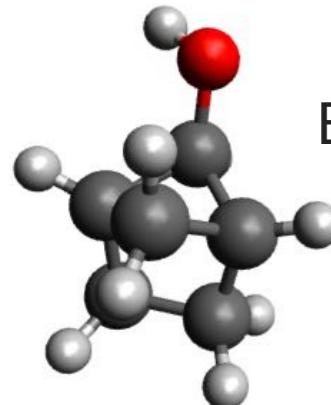


Data-driven approximation to quantum chem

Quantum chemistry structures and properties of 134 kilo molecules, *Scientific Data* 1, 140022 (2014)

<http://www.nature.com/articles/sdata201422>

<http://quantum-machine.org/datasets/>



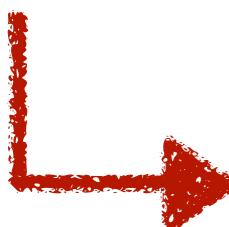
DFT
B3LYP/6-31G(2df,p)

U_0, U, H, G
 ω_1, ZPVE
 $\varepsilon_{\text{HOMO}}, \varepsilon_{\text{LUMO}}, \Delta\varepsilon$
 $\langle R^2 \rangle, \mu, \alpha$

Structure in
the ground state

15 properties in
the ground state

HOMO: -0.3877 LUMO: 0.1171 Dipole_moment: 0.0	HOMO: -0.257 LUMO: 0.0829 Dipole_moment: 1.6256	HOMO: -0.2928 LUMO: 0.0687 Dipole_moment: 1.8511	HOMO: -0.2845 LUMO: 0.0506 Dipole_moment: 0.0	HOMO: -0.3684 LUMO: 0.0191 Dipole_moment: 2.8937
HOMO: -0.267 LUMO: -0.8406 Dipole_moment: 2.1889	HOMO: -0.3385 LUMO: 0.1041 Dipole_moment: 0.0	HOMO: -0.2653 LUMO: 0.0784 Dipole_moment: 1.5258	HOMO: -0.2609 LUMO: 0.0613 Dipole_moment: 0.7156	HOMO: -0.3264 LUMO: 0.0376 Dipole_moment: 3.8266
HOMO: -0.254 LUMO: -0.8198 Dipole_moment: 2.5482	HOMO: -0.2543 LUMO: 0.0302 Dipole_moment: 3.7286	HOMO: -0.323 LUMO: 0.0949 Dipole_moment: 0.0597	HOMO: -0.2619 LUMO: 0.0798 Dipole_moment: 1.4131	HOMO: -0.2525 LUMO: 0.091 Dipole_moment: 1.1502
HOMO: -0.2888 LUMO: 0.1842 Dipole_moment: 5.8E-4	HOMO: -0.2682 LUMO: 0.1842 Dipole_moment: 1.7675	HOMO: -0.2431 LUMO: -0.0087 Dipole_moment: 2.7362	HOMO: -0.2436 LUMO: 0.0347 Dipole_moment: 3.6367	HOMO: -0.2495 LUMO: 0.0556 Dipole_moment: 3.4869
HOMO: -0.3167 LUMO: 0.0843 Dipole_moment: 0.8897	HOMO: -0.2612 LUMO: 0.074 Dipole_moment: 1.4259	HOMO: -0.2599 LUMO: -0.0214 Dipole_moment: 0.0	HOMO: -0.3182 LUMO: -0.0543 Dipole_moment: 3.792	HOMO: -0.3696 LUMO: -0.0926 Dipole_moment: 0.0023



JCTC
Journal of Chemical Theory and Computation

Cite This: *J. Chem. Theory Comput.* 2017, 13, 5255–5264

Article

pubs.acs.org/JCTC

Prediction Errors of Molecular Machine Learning Models Lower than Hybrid DFT Error

Felix A. Faber,[†] Luke Hutchison,[‡] Bing Huang,[†] Justin Gilmer,[‡] Samuel S. Schoenholz,[‡] George E. Dahl,[‡] Oriol Vinyals,[†] Steven Kearnes,[†] Patrick F. Riley,[‡] and O. Anatole von Lilienfeld^{*†}

PERSPECTIVES

nature reviews chemistry

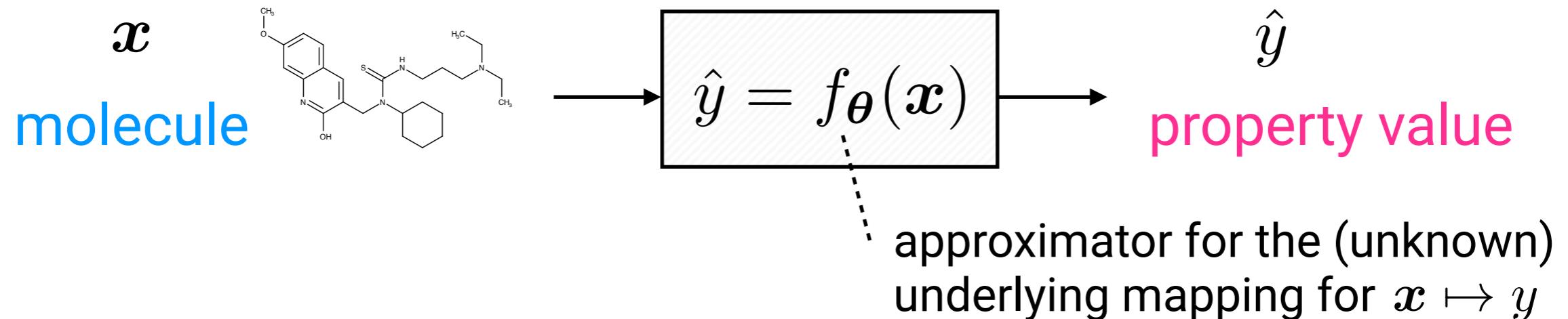
Exploring chemical compound space with quantum-based machine learning

O. Anatole von Lilienfeld, Klaus-Robert Müller and Alexandre Tkatchenko

Abstract | Rational design of compounds with specific properties requires understanding and fast evaluation of molecular properties throughout chemical compound space — the huge set of all potentially stable molecules. Recent advances in combining quantum-mechanical calculations with machine learning provide powerful tools for exploring wide swathes of chemical compound space. We present our perspective on this exciting and quickly developing field by discussing key advances in the development and applications of quantum-mechanics-based machine-learning methods to diverse compounds and properties, and outlining the challenges ahead. We argue that significant progress in the exploration and understanding of chemical compound space can be made through a systematic combination of rigorous physical theories, comprehensive synthetic data sets of microscopic and macroscopic properties, and modern machine-learning methods that account for physical and chemical knowledge.

Nat Rev Chem 4, 347–358 (2020)

“Molecular Machine Learning”



Given n **input-output** instances (as the training data)

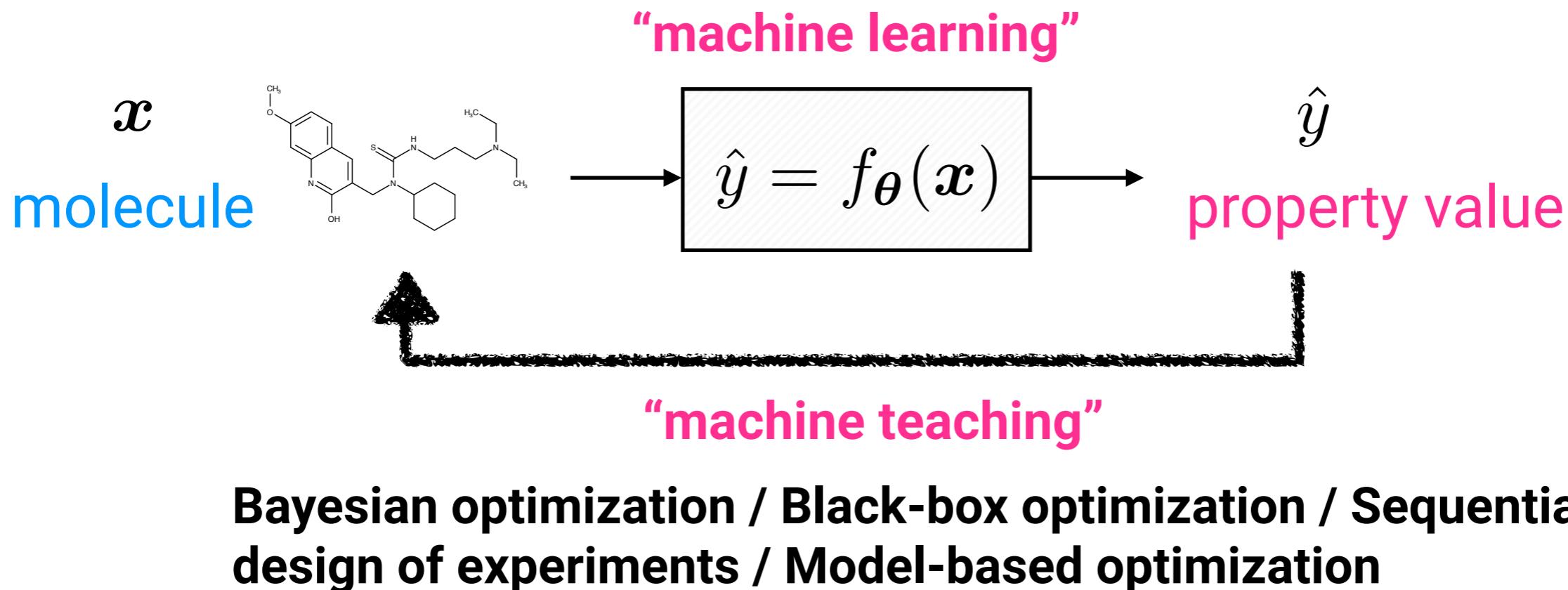
$$\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_n, y_n)\}$$

Fit the model f_{θ} by tuning θ as

$$\min_{\theta} \sum_{i=1}^n \text{error}(y_i, \hat{y}_i) \quad \text{where} \quad \hat{y}_i = f_{\theta}(\mathbf{x}_i)$$

Note: the error measure (called "loss function" in ML) depends on problems

Inverse design?



Aug 26: 10:30~12:00 (90min)

1. What is "machine learning"?
2. Why does it matter to chemists?
3. Let's try it in your browser (with no setup!)

Aug 26: 13:00~14:30 (90min)

4. Five things all beginners should know
 - "The quality of your inputs decide the quality of your output"
 - Training / validation / test data
 - Tuning hyperparameters
 - Identification and design of input variables (or "descriptors")
 - "Correlation does not imply causation"
5. Standard pipeline and deep learning
6. Current efforts and future directions

Now is the good timing to start machine learning!

Let's give it a try anyway!

All you need for now is a **web browser only.
(on your PC or smartphones)**

<https://colab.research.google.com/>



Google Colab (Google Collaboratory)

<https://colab.research.google.com/>

Welcome To Colaboratory

File Examples Recent Google Drive GitHub Upload

Table of c

Getting s

Data m

Machin

More Res

Machin

Specia

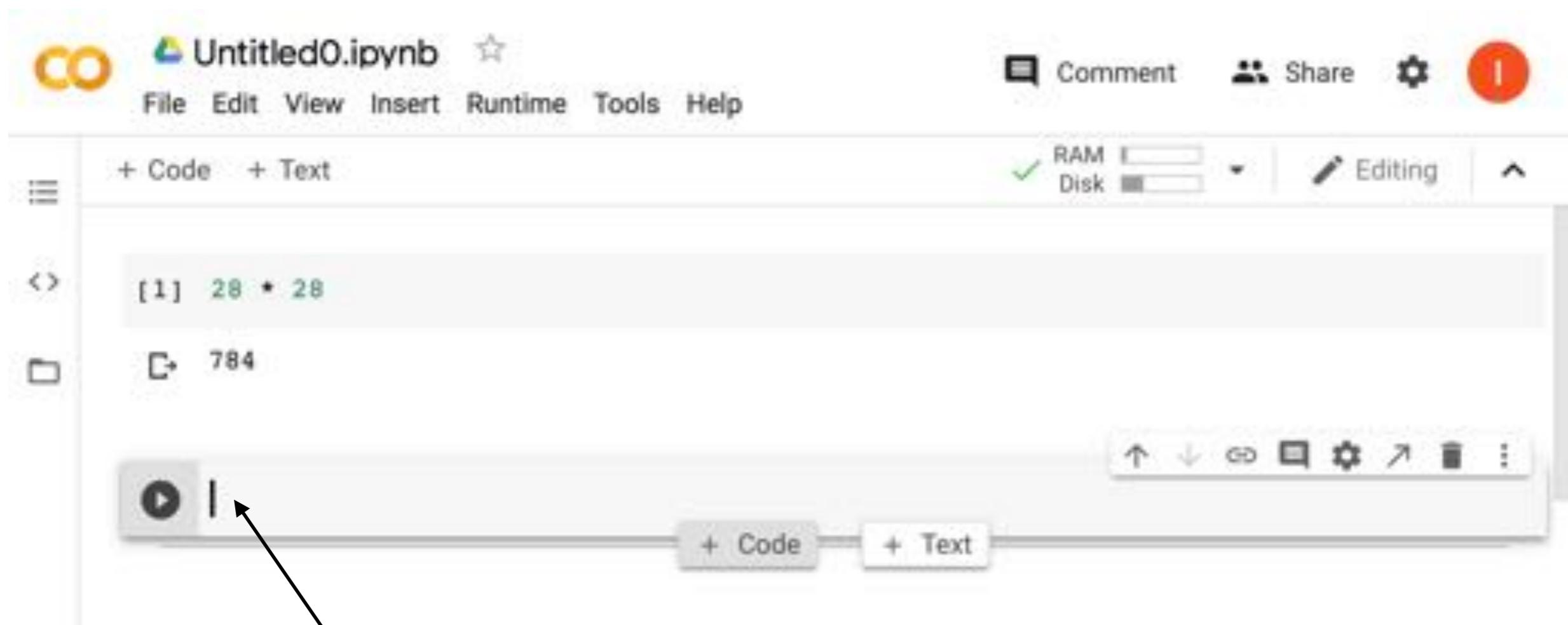
Filter notebooks

Title	First opened	Last opened	⋮
Welcome To Colaboratory	7 minutes ago	0 minutes ago	⋮

NEW NOTEBOOK CANCEL

Google Colab (Google Collaboratory)

<https://colab.research.google.com/>



You can input a **Python** code here and
run it by clicking the button
(or by Shift+Enter)

Let's reproduce a paper-level result!

RSC Advances



PAPER

[View Article Online](#)
[View Journal](#) | [View Issue](#)



Cite this: RSC Adv., 2016, 6, 52587

Machine-learning prediction of the d-band center for metals and bimetals

Ichigaku Takigawa,^{*ab} Ken-ichi Shimizu,^{cd} Koji Tsuda^{efg} and Satoru Takakusagi^c

The d-band center for metals has been widely used in order to understand activity trends in metal-surface-catalyzed reactions in terms of the linear Brønsted-Evans-Polanyi relation and Hammer-Nørskov d-band model. In this paper, the d-band centers for eleven metals (Fe, Co, Ni, Cu, Ru, Rh, Pd, Ag, Ir, Pt, Au) and their pairwise bimetals for two different structures (1% metal doped- or overlayer-covered metal surfaces) are statistically predicted using machine learning methods from readily available values as descriptors for the target metals (such as the density and the enthalpy of fusion of each metal). The predictive accuracy of four regression methods with different numbers of descriptors and different test-set/training-set ratios are quantitatively evaluated using statistical cross validations. It is shown that the d-band centers are reasonably well predicted by the gradient boosting regression (GBR) method with only six descriptors, even when we predict 75% of the data from only 25% given for training (average root mean square error (RMSE) < 0.5 eV). This demonstrates a potential use of machine learning methods for predicting the activity trends of metal surfaces with a negligible CPU time compared to first-principles methods.

Received 18th February 2016
Accepted 23rd May 2016

DOI: 10.1039/c6ra04345c

www.rsc.org/advances

Let's reproduce a paper-level result!

Table 1 DFT calculated d-band centers (eV) of metals (*italic*) and 1% guest metals (M_g) doped in the surface of host metals (M_h) as reported by Nørskov's group^{1,2}

M_h	M_g	Fe	Co	Ni	Cu	Ru	Rh	Pd	Ag	Ir	Pt	Au
Fe		-0.92	-0.87	-1.12	-1.05	-1.21	-1.46	-2.16	-1.75	-1.28	-2.01	-2.34
Co		-1.16	-1.17	-1.45	-1.33	-1.41	-1.75	-2.54	-2.08	-1.53	-2.36	-2.73
Ni		-1.20	-1.10	-1.29	-1.10	-1.43	-1.60	-2.26	-1.82	-1.43	-2.09	-2.42
Cu		-2.11	-2.07	-2.40	-2.67	-2.09	-2.35	-3.31	-3.37	-2.09	-3.00	-3.76
Ru		-1.20	-1.15	-1.40	-1.29	-1.41	-1.58	-2.23	-1.68	-1.39	-2.03	-2.25
Rh		-1.49	-1.39	-1.57	-1.29	-1.69	-1.73	-2.27	-1.66	-1.56	-2.08	-2.22
Pd		-1.46	-1.29	-1.33	-0.89	-1.59	-1.47	-1.83	-1.24	-1.30	-1.64	-1.66
Ag		-3.58	-3.46	-3.63	-3.83	-3.46	-3.44	-4.16	-4.30	-3.16	-3.80	-4.45
Ir		-1.90	-1.84	-2.06	-1.90	-2.02	-2.26	-2.84	-2.24	-2.11	-2.67	-2.85
Pt		-1.92	-1.77	-1.85	-1.53	-2.11	-2.02	-2.42	-1.81	-1.87	-2.25	-2.30
Au		-2.93	-2.79	-2.93	-3.01	-2.86	-2.81	-3.39	-3.35	-2.58	-3.10	-3.56

Table 3 Input features (descriptors) used for prediction of d-band centers from ref. 34^a

Metal	G	$R/\text{\AA}$	AN	AM/g mol ⁻¹	P	EN	IE/eV	$\Delta_{\text{fus}}H/J \text{ g}^{-1}$	$\rho/\text{g cm}^{-3}$
Fe	8	2.66	26	55.85	4	1.83	7.90	247.3	7.87
Co	9	2.62	27	58.93	4	1.88	7.88	272.5	8.86
Ni	10	2.60	28	58.69	4	1.91	7.64	290.3	8.90
Cu	11	2.67	29	63.55	4	1.90	7.73	203.5	8.96
Ru	8	2.79	44	101.07	5	2.20	7.36	381.8	12.10
Rh	9	2.81	45	102.91	5	2.28	7.46	258.4	12.40
Pd	10	2.87	46	106.42	5	2.20	8.34	157.3	12.00
Ag	11	3.01	47	107.87	5	1.93	7.58	104.6	10.50
Ir	9	2.84	77	192.22	6	2.20	8.97	213.9	22.50
Pt	10	2.90	78	195.08	6	2.20	8.96	113.6	21.50
Au	11	3.00	79	196.97	6	2.40	9.23	64.6	19.30

Table 1 and Table 3



Fig 1

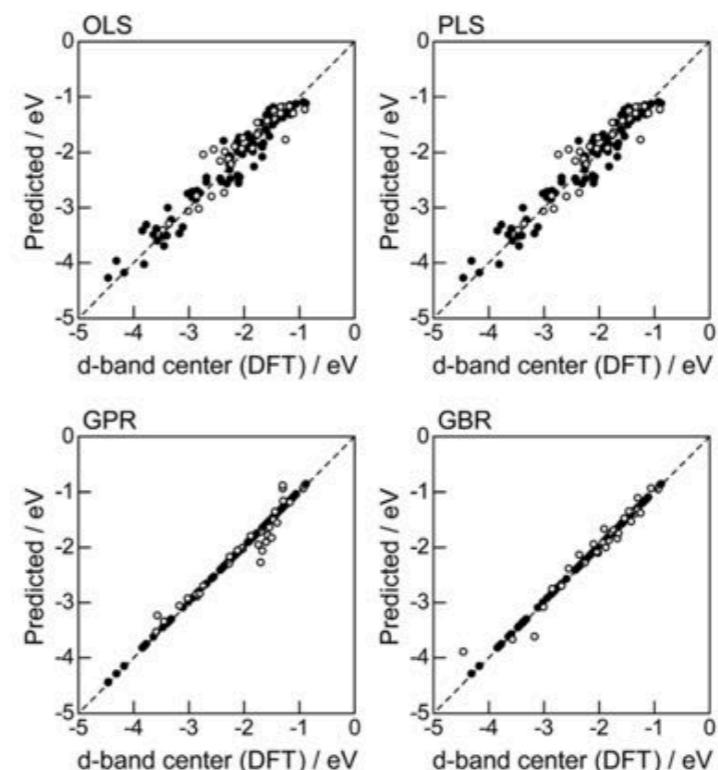


Fig. 1 DFT calculated local d-band center for metals and 1% guest metal-doped metals (Table 1) and the values predicted by linear (OLS, PLS) and nonlinear regression (GPR, GBR): (●) training set = 75%, (○) test set = 25%.

Let's reproduce a paper-level result!

Table 1: the energy of the d-band center relative to the Fermi level (ϵ_F), $\epsilon - \epsilon_F$ for 1% guest metals doped in the surface of host metals.

calculated from
the original data

A. Ruban, B. Hammer,
P. Stoltze, H. L. Skriver
and J. K. Nørskov, J.
Mol. Catal. A: Chem.,
1997, 115, 421–429.

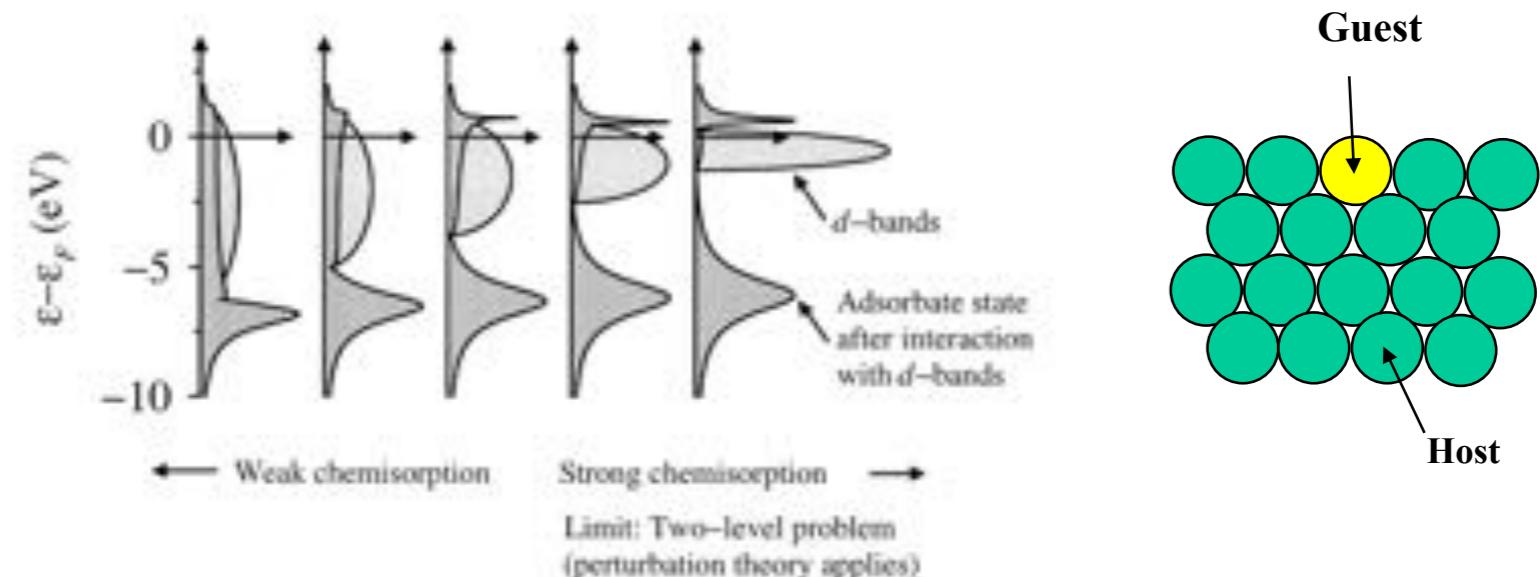


Table I
Shifts in d-band centers of surface impurities and overlayer relative to the clean metal values (italic)

	Fe	Co	Ni	Cu	Ru	Rh	Pd	Ag	Ir	Pt	Au
Fe	-0.92	0.05	-0.20	-0.13	-0.29	-0.54	-1.24	-0.83	-0.36	-1.09	-1.42
		0.14	-0.04	-0.05	-0.73	-0.72	-1.32	-1.25	-0.95	-1.48	-2.19
Co	0.01	-1.17	-0.28	-0.16	-0.24	-0.58	-1.37	-0.91	-0.36	-1.19	-1.56
	-0.01		-0.20	-0.06	-0.70	-0.95	-1.65	-1.36	-1.09	-1.89	-2.39
Ni	0.09	0.19	-1.29	0.19	-0.14	-0.31	-0.97	-0.53	-0.14	-0.80	-1.13
	0.96	0.11		0.12	-0.63	-0.74	-1.32	-1.14	-0.86	-1.53	-2.10
Cu	0.56	0.60	0.27	-2.67	0.58	0.32	-0.64	-0.70	0.58	-0.33	-1.09
	0.25	0.38	0.18		-0.22	-0.27	-1.04	-1.21	-0.32	-1.15	-1.96
Ru	0.21	0.26	0.01	0.12	-1.41	-0.17	-0.82	-0.27	0.02	-0.62	-0.84
	0.30	0.37	0.29	0.30		-0.12	-0.47	-0.40	-0.13	-0.61	-0.86
Rh	0.24	0.34	0.16	0.44	0.04	-1.73	-0.54	0.07	0.17	-0.35	-0.49
	0.31	0.41	0.34	0.22	0.03		-0.39	-0.08	0.03	-0.45	-0.57
Pd	0.37	0.54	0.50	0.94	0.24	0.36	-1.83	0.59	0.53	0.19	0.17
	0.36	0.54	0.54	0.80	-0.11	0.25		0.15	0.31	0.04	-0.14
Ag	0.72	0.84	0.67	0.47	0.84	0.86	0.14	-4.30	1.14	0.50	-0.15
	0.55	0.74	0.68	0.62	0.50	0.67	0.27		0.80	0.37	-0.21
Ir	0.21	0.27	0.05	0.21	0.09	-0.15	-0.73	-0.13	-2.11	-0.56	-0.74
	0.33	0.40	0.33	0.56	-0.01	-0.03	-0.42	-0.09		-0.49	-0.59
Pt	0.33	0.48	0.40	0.72	0.14	0.23	-0.17	0.44	0.38	-2.25	-0.05
	0.35	0.53	0.54	0.78	0.12	0.24	0.02	0.19	0.29		-0.08
Au	0.63	0.77	0.63	0.55	0.70	0.75	0.17	0.21	0.98	0.46	-3.56
	0.53	0.74	0.71	0.70	0.47	0.67	0.35	0.12	0.79	0.43	

The impurity/overlayer atoms are listed horizontally and the host entries are listed vertically. For each combination of the two numbers listed is first the isolated surface impurity given and than the overlayer. The surfaces considered are the most close packed and the overlayer structures are pseudomorphic. No relaxations from the host lattice positions are included. All values are in eV and the elemental d band centers are relative to the Fermi level.

Let's reproduce a paper-level result!

You can get the █ part of the original data

```
[1] import numpy as np  
import pandas as pd  
  
[2] url = "https://itakigawa.github.io/data/hsi2020/data\_impurities.csv"  
  
my_table = pd.read_csv(url, index_col=0)
```

```
[3] my_table
```

C	Fe	Co	Ni	Cu	Ru	Rh	Pd	Ag	Ir	Pt	Au
Fe	-0.92	0.05	-0.20	-0.13	-0.29	-0.54	-1.24	-0.83	-0.36	-1.09	-1.42
Co	0.01	-1.17	-0.28	-0.16	-0.24	-0.58	-1.37	-0.91	-0.36	-1.19	-1.56
Ni	0.09	0.19	-1.29	0.19	-0.14	-0.31	-0.97	-0.53	-0.14	-0.80	-1.13
Cu	0.56	0.60	0.27	-2.67	0.58	0.32	-0.64	-0.70	0.58	-0.33	-1.09
Ru	0.21	0.26	0.01	0.12	-1.41	-0.17	-0.82	-0.27	0.02	-0.62	-0.84

handle table data by "pandas"

```
▶ my_table.loc['Co', 'Ni']
```

```
⇨ -0.28
```

```
▶ my_table.loc['Co']
```

```
⇨ Fe      0.01  
Co     -1.17  
Ni     -0.28  
Cu     -0.16  
Ru     -0.24  
Rh     -0.58  
Pd     -1.37  
Ag     -0.91  
Ir     -0.36  
Pt     -1.19  
Au     -1.56
```

```
Name: Co, dtype: float64
```

	Fe	Co	Ni	Cu	Ru	Rh	Pd	Ag	Ir	Pt	Au
Fe	-0.92	0.05	-0.20	-0.13	-0.29	-0.54	-1.24	-0.83	-0.36	-1.09	-1.42
Co	0.01	-1.17	-0.28	-0.16	-0.24	-0.58	-1.37	-0.91	-0.36	-1.19	-1.56
Ni	0.09	0.19	-1.29	0.19	-0.14	-0.31	-0.97	-0.53	-0.14	-0.80	-1.13
Cu	0.56	0.60	0.27	-2.67	0.58	0.32	-0.64	-0.70	0.58	-0.33	-1.09
Ru	0.21	0.26	0.01	0.12	-1.41	-0.17	-0.82	-0.27	0.02	-0.62	-0.84

	Fe	Co	Ni	Cu	Ru	Rh	Pd	Ag	Ir	Pt	Au
Fe	-0.92	0.05	-0.20	-0.13	-0.29	-0.54	-1.24	-0.83	-0.36	-1.09	-1.42
Co	0.01	-1.17	-0.28	-0.16	-0.24	-0.58	-1.37	-0.91	-0.36	-1.19	-1.56
Ni	0.09	0.19	-1.29	0.19	-0.14	-0.31	-0.97	-0.53	-0.14	-0.80	-1.13
Cu	0.56	0.60	0.27	-2.67	0.58	0.32	-0.64	-0.70	0.58	-0.33	-1.09
Ru	0.21	0.26	0.01	0.12	-1.41	-0.17	-0.82	-0.27	0.02	-0.62	-0.84

handle table data by "pandas"

```
▶ for h in my_table.index:  
    for g in my_table.columns:  
        print(f'host {h}, guest {g}, val {my_table.loc[h, g]}')
```

```
↳ host Fe, guest Fe, val -0.92  
host Fe, guest Co, val 0.05  
host Fe, guest Ni, val -0.2  
host Fe, guest Cu, val -0.13  
host Fe, guest Ru, val -0.29  
host Fe, guest Rh, val -0.54
```

columns

	Fe	Co	Ni	Cu	Ru	Rh	Pd	Ag	Ir	Pt	Au
Fe	-0.92	0.05	-0.20	-0.13	-0.29	-0.54	-1.24	-0.83	-0.36	-1.09	-1.42
Co	0.01	-1.17	-0.28	-0.16	-0.24	-0.58	-1.37	-0.91	-0.36	-1.19	-1.56
Ni	0.09	0.19	-1.29	0.19	-0.14	-0.31	-0.97	-0.53	-0.14	-0.80	-1.13
Cu	0.56	0.60	0.27	-2.67	0.58	-0.32	-0.64	-0.70	0.58	-0.33	-1.09
Ru	0.21	0.26	0.01	0.12	-1.41	-0.17	-0.82	-0.27	0.02	-0.62	-0.84

index

```
graph LR; subgraph Index [index]; Fe1[Fe]; Co1[Co]; Ni1[Ni]; Cu1[Cu]; Ru1[Ru]; end; subgraph Columns [columns]; Fe2[Fe]; Co2[Co]; Ni2[Ni]; Cu2[Cu]; Ru2[Ru]; end; Fe1 --> Fe2; Co1 --> Co2; Ni1 --> Ni2; Cu1 --> Cu2; Ru1 --> Ru2;
```

Get Table 1

A. Ruban, B. Hammer, P. Stoltze, H. L. Skriver and J. K. Nørskov, J. Mol. Catal. A: Chem., 1997, 115, 421–429.

Table I

Shifts in d-band centers of surface impurities and overlayers relative to the clean metal values (italic)

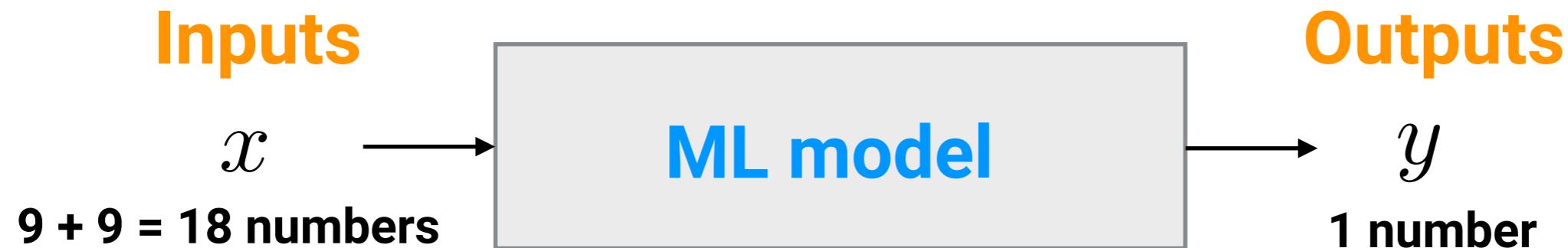
	Fe	Co	Ni	Cu	Ru	Rh	Pd	Ag	Ir	Pt
Fe	-0.92	0.05	-0.20	-0.13	-0.29	-0.54	-1.24	-0.83	-0.36	-1.09

```
▶ for h in my_table.index:  
    for g in my_table.columns:  
        if h != g:  
            my_table.loc[h, g] += my_table.loc[h, h]
```

my_table

	Fe	Co	Ni	Cu	Ru	Rh	Pd	Ag	Ir	Pt	Au
Fe	-0.92	-0.87	-1.12	-1.05	-1.21	-1.46	-2.16	-1.75	-1.28	-2.01	-2.34
Co	-1.16	-1.17	-1.45	-1.33	-1.41	-1.75	-2.54	-2.08	-1.53	-2.36	-2.73
Ni	-1.20	-1.10	-1.29	-1.10	-1.43	-1.60	-2.26	-1.82	-1.43	-2.09	-2.42
Cu	-2.11	-2.07	-2.40	-2.67	-2.09	-2.35	-3.31	-3.37	-2.09	-3.00	-3.76
Ru	-1.20	-1.15	-1.40	-1.29	-1.41	-1.58	-2.23	-1.68	-1.39	-2.03	-2.25

Our goal here is the following machine learning.



[8, 2.66, 26, 55.85, 4, 1.83, 7.90, 247.3, 7.87,
11, 2.67, 29, 63.55, 4, 1.90, 7.73, 203.5, 8.96] -1.05

9 features (host metal) from Table 3

+

9 features (guest metal) from Table 3

the value at Table 1
for the host and guest metal

Table 3 Input features (descriptors) used for prediction of d-band centers from ref. 34^a

Metal	G	R/Å	AN	AM/g mol ⁻¹	P	EN	IE/ eV	$\Delta_{\text{fus}}H/\text{J g}^{-1}$	$\rho/\text{g cm}^{-3}$
Fe	8	2.66	26	55.85	4	1.83	7.90	247.3	7.87
Co	9	2.62	27	58.93	4	1.88	7.88	272.5	8.86
Ni	10	2.60	28	58.69	4	1.91	7.64	290.3	8.90
Cu	11	2.67	29	63.55	4	1.90	7.73	203.5	8.96

	Fe	Co	Ni	Cu	Ru	Rh
Fe	-0.92	-0.87	-1.12	-1.05	-1.21	-1.46
Co	-1.16	-1.17	-1.45	-1.33	-1.41	-1.75
Ni	-1.20	-1.10	-1.29	-1.10	-1.43	-1.60
Cu	-2.11	-2.07	-2.40	-2.67	-2.09	-2.35

Let's make the inputs

```
[17] url2 = "https://itakigawa.github.io/data/ha12020/features9.csv"
feat = pd.read_csv(url2, index_col=0)
```

```
[18] feat.head(3)
```

			Bulk						
	name	Num of d-electrons	wigner-seitz radius	atomic number	atomic mass	period	electronegativity		Ionization energy(eV)
	symbol								
	Fe	Iron	8	2.66	26	55.8450	4	1.83	7.9024
	Co	Cobalt	9	2.62	27	58.9332	4	1.88	7.8810
	Ni	Nickel	10	2.60	28	58.6934	4	1.91	7.6398

```
[19] feat.loc['Co']
```

```
[C] name          Cobalt
Num of d-electrons      9
Bulk wigner-seitz radius 2.62
atomic number        27
```

Let's make the inputs

```
feat.drop('name', axis='columns', inplace=True)
```

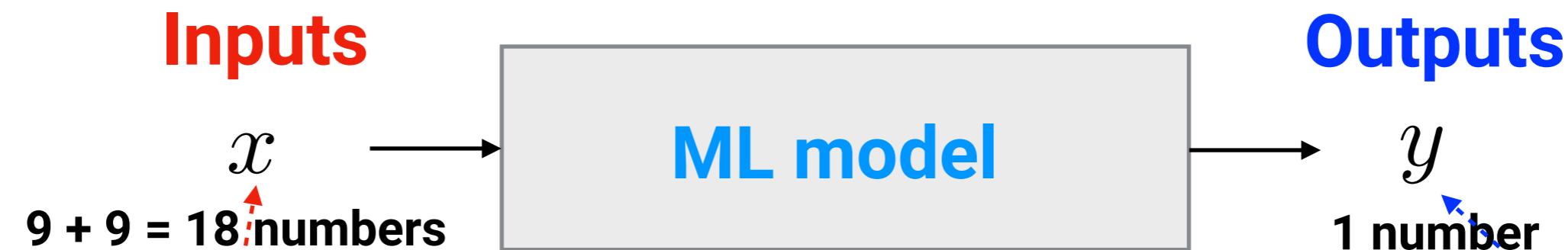
```
x = list()
y = list()
for h in my_table.index:
    for g in my_table.columns:
        vec_h = feat.loc[h].to_numpy()
        vec_g = feat.loc[g].to_numpy()
        x_val = np.concatenate((vec_h, vec_g))
        y_val = my_table.loc[h][g]
        x.append(x_val)
        y.append(y_val)
if h == 'Fe' and g == 'Cu':
    print(f'host({h}), guest({g}), input={x_val}, output={y_val}')
```

```
host(Fe), guest(Cu), input=[ 8.          2.66         26.          55.845       4.          1.83         7.9024      247.3
  7.87        11.          2.67         29.          63.546       4.          1.9          7.7264
 203.5        8.96      ], output=-1.05
```



[8, 2.66, 26, 55.85, 4, 1.83, 7.90, 247.3, 7.87, 11, 2.67, 29, 63.55, 4, 1.90, 7.73, 203.5, 8.96]	-1.05
---	-------

Our (X, y) data



```
df = pd.DataFrame(X, columns=[f"x{i+1}" for i in range(18)])
df['y'] = y
df
```

	x1	x2	x3	x4	x5	x6	x7	x8	x9	x10	x11	x12	x13	x14	x15	x16	x17	x18	y
0	11.0	2.67	29.0	63.54600	4.0	1.90	7.7264	203.5	8.96	9.0	2.84	77.0	192.21700	6.0	2.20	8.9670	213.9	22.50	-2.09
1	9.0	2.62	27.0	58.93320	4.0	1.88	7.8810	272.5	8.86	11.0	3.00	79.0	196.96655	6.0	2.40	9.2255	64.6	19.30	-2.73
2	10.0	2.90	78.0	195.07800	6.0	2.20	8.9588	113.6	21.50	8.0	2.79	44.0	101.07000	5.0	2.20	7.3605	381.8	12.10	-2.11
3	11.0	3.00	79.0	196.96655	6.0	2.40	9.2255	64.6	19.30	9.0	2.84	77.0	192.21700	6.0	2.20	8.9670	213.9	22.50	-2.58
4	10.0	2.90	78.0	195.07800	6.0	2.20	8.9588	113.6	21.50	9.0	2.62	27.0	58.93320	4.0	1.88	7.8810	272.5	8.86	-1.77
...	
116	9.0	2.62	27.0	58.93320	4.0	1.88	7.8810	272.5	8.86	9.0	2.81	45.0	102.90550	5.0	2.28	7.4589	258.4	12.40	-1.75
117	9.0	2.81	45.0	102.90550	5.0	2.28	7.4589	258.4	12.40	11.0	3.00	79.0	196.96655	6.0	2.40	9.2255	64.6	19.30	-2.22
118	8.0	2.79	44.0	101.07000	5.0	2.20	7.3605	381.8	12.10	9.0	2.62	27.0	58.93320	4.0	1.88	7.8810	272.5	8.86	-1.15
119	9.0	2.81	45.0	102.90550	5.0	2.28	7.4589	258.4	12.40	8.0	2.79	44.0	101.07000	5.0	2.20	7.3605	381.8	12.10	-1.69
120	9.0	2.62	27.0	58.93320	4.0	1.88	7.8810	272.5	8.86	9.0	2.84	77.0	192.21700	6.0	2.20	8.9670	213.9	22.50	-1.53

121 rows × 19 columns

The full code for preparation (only 21 lines!)

```
import numpy as np
import pandas as pd

url1 = "https://itakigawa.github.io/data/hsi2020/data_impurities.csv"
url2 = "https://itakigawa.github.io/data/hsi2020/features9.csv"

my_table = pd.read_csv(url1, index_col=0)

feat = pd.read_csv(url2, index_col=0)
feat.drop('name', axis='columns', inplace=True)

for h in my_table.index:
    for g in my_table.columns:
        if h != g:
            my_table.loc[h, g] += my_table.loc[h, h]

X = list()
y = list()
for h in my_table.index:
    for g in my_table.columns:
        vec_h = feat.loc[h].to_numpy()
        vec_g = feat.loc[g].to_numpy()
        X.append(np.concatenate((vec_h, vec_g)))
        y.append(my_table.loc[h][g])

X = np.stack(X)
y = np.array(y)
```

Now we can move on to the "machine learning" part!

```
[8] X.shape, y.shape
```

```
↳ ((121, 18), (121,))
```

The number of input-output examples is 121. So we'll use random 30 examples for evaluation, and the remaining 91 examples for the model fitting.

```
[21] from sklearn.utils import shuffle  
x, y = shuffle(X, y) ←  
x_train, y_train = X[:-30, :], y[:-30]  
x_test, y_test = X[-30:, :], y[-30:]
```

This shuffling is quite important.

See what happens if you skip it, and think why.

Let's go with "machine learning" with 1 line of "model.fit"

```
[23] from sklearn.ensemble import GradientBoostingRegressor  
model = GradientBoostingRegressor()  
model.fit(x_train, y_train)  
y_pred_train = model.predict(x_train)  
y_pred_test = model.predict(x_test)
```

Make a plot.

```
import matplotlib.pyplot as plt

fig, ax = plt.subplots()
ax.scatter(y_train, y_pred_train, \
           alpha=0.5, color="blue", label="training")
ax.scatter(y_test, y_pred_test, \
           alpha=0.5, color="red", label="test")
ax.legend()
ax.set_xlabel('groudtruth')
ax.set_ylabel('prediction')
ax.set_aspect("equal")
```

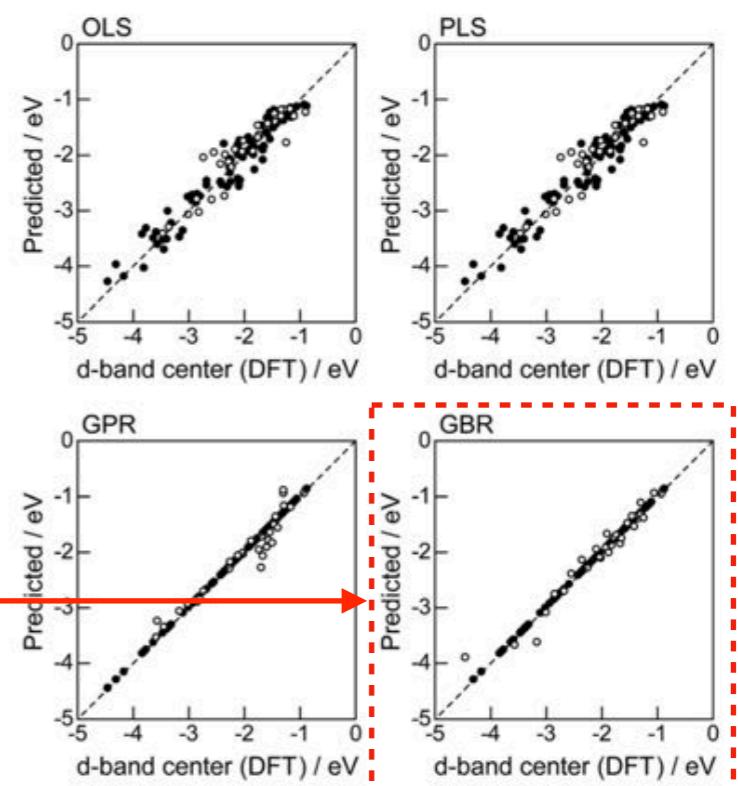
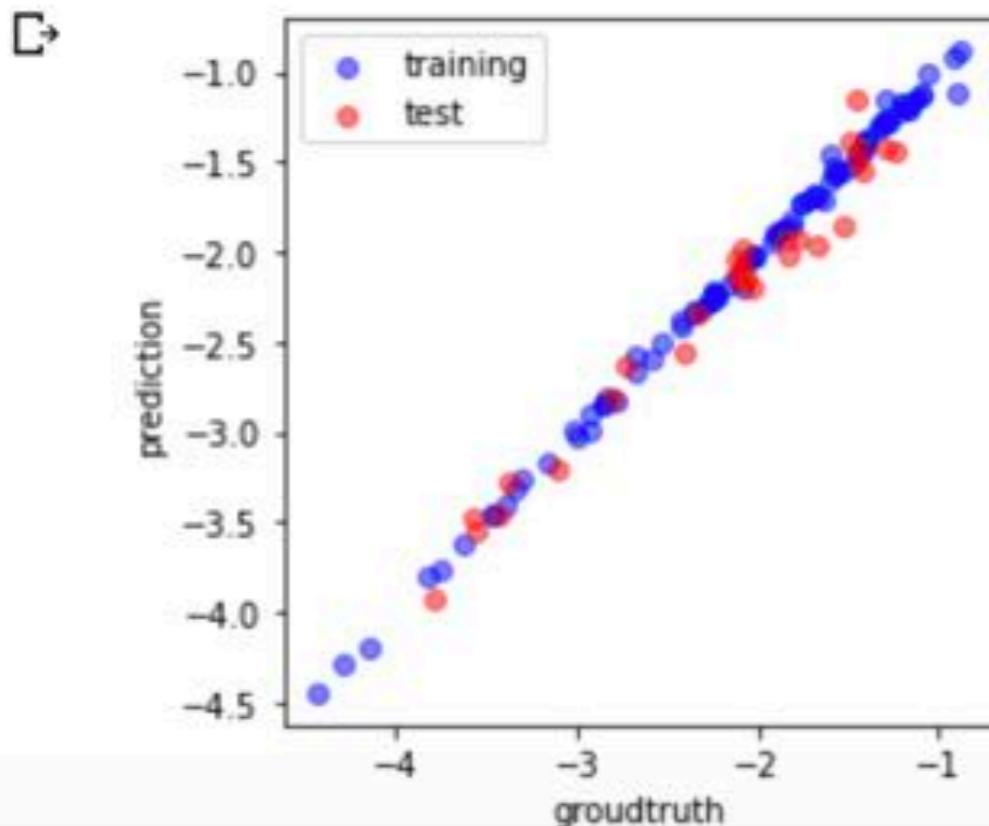


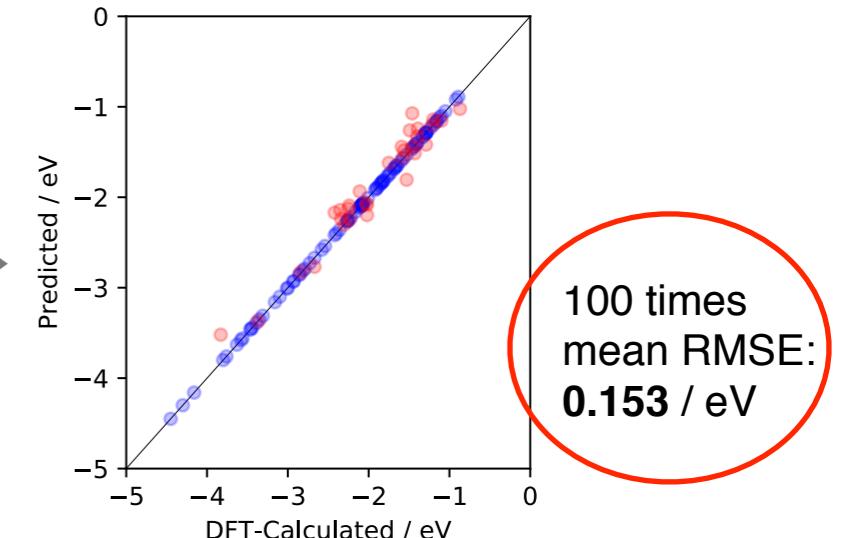
Fig. 1 DFT calculated local d-band center for metals and 1% guest metal-doped metals (Table 1) and the values predicted by linear (OLS, PLS) and nonlinear regression (GPR, GBR): (●) training set = 75%, (○) test set = 25%.

The paper used only 6 features out of 18...

	Fe	Co	Ni	Cu	Ru	Rh	Pd	Ag	Ir	Pt	Au
Fe	-0.92		-0.96	-0.97	-1.65	-1.64	-2.24		-1.87	-2.4	-3.11
Co			-1.37	-1.23		-2.12	-2.82	-2.53	-2.26		-3.56
Ni	-0.33	-1.18			-1.92	-2.03		-2.43	-2.15	-2.82	-3.39
Cu	-2.42		-2.49	-2.67	-2.89	-2.94				-3.82	-4.63
Ru	-1.11	-1.04	-1.12		-1.41		-1.88	-1.81	-1.54		-2.27
Rh	-1.42	-1.32		-1.51	-1.7	-1.73	-2.12	-1.81	-1.7	-2.18	-2.3
Pd	-1.47	-1.29	-1.29	-1.03		-1.58	-1.83	-1.68	-1.52	-1.79	
Ag	-3.75	-3.56	-3.62		-3.8		-4.03		-3.5	-3.93	-4.51
Ir	-1.78	-1.71	-1.78	-1.55		-2.14	-2.53	-2.2	-2.11	-2.6	-2.7
Pt			-1.71	-1.47	-2.13	-2.01	-2.23	-2.06	-1.96		-2.33
Au	-3.03	-2.82	-2.85		-2.89		-3.44				-3.56

gradient boosting
w/ 6 descriptors

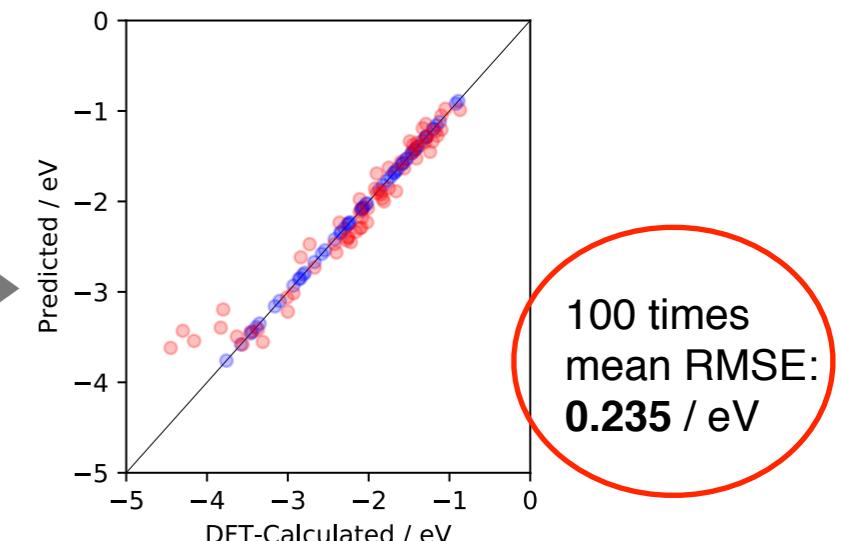
→
training sets (75%)
test sets (25%)



	Fe	Co	Ni	Cu	Ru	Rh	Pd	Ag	Ir	Pt	Au
Fe		-0.78			-1.65	-1.64			-1.87		
Co	-1.18	-1.17	-1.37		-1.87	-2.12	-2.82		-2.26		
Ni	-0.33	-1.18		-1.17			-2.61	-2.43	-2.15	-2.82	
Cu	-2.42				-2.89	-2.94		-3.88			-4.63
Ru	-1.11	-1.04	-1.12	-1.11	-1.41			-1.81			-2.27
Rh	-1.42			-1.51			-2.12	-1.81	-1.7		
Pd		-1.29	-1.29	-1.03		-1.58	-1.83		-1.52	-1.79	
Ag				-3.68	-3.8	-3.63					-4.51
Ir					-2.14				-2.11		-2.7
Pt				-1.71	-1.47	-2.13	-2.01	-2.23	-2.06		
Au				-2.86	-3.09	-2.89		-3.44			-3.56

gradient boosting
w/ 6 descriptors

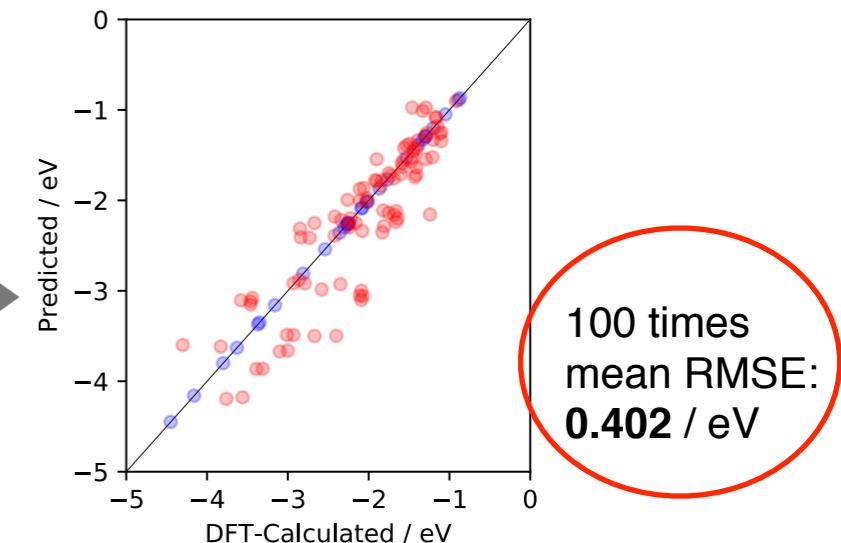
→
training sets (50%)
test sets (50%)



	Fe	Co	Ni	Cu	Ru	Rh	Pd	Ag	Ir	Pt	Au
Fe								-2.17			-3.11
Co		-1.17	-1.37				-2.12				
Ni	-0.33	-1.18					-2.61	-2.43			
Cu	-2.42	-2.29	-2.49				-3.71				-4.63
Ru									-2.02		
Rh		-1.32				-1.73	-2.12				
Pd					-1.94		-1.83				-1.97
Ag	-3.75			-3.68							-4.51
Ir	-1.78	-1.71									-2.7
Pt				-2.13							
Au				-3.09	-2.89						

gradient boosting
w/ 6 descriptors

→
training sets (25%)
test sets (75%)



We can also easily compute the quantitative performance!

https://en.wikipedia.org/wiki/Root-mean-square_deviation

```
from sklearn.metrics import mean_squared_error  
rmse_tr = mean_squared_error(y_train, y_pred_train, squared=False)  
rmse_te = mean_squared_error(y_test, y_pred_test, squared=False)  
print(f'RMSE(training) {rmse_tr:.3f}')  
print(f'RMSE(test) {rmse_te:.3f}')
```

```
RMSE(training) 0.041  
RMSE(test) 0.141
```

root-mean-square error
(RMSE) = $\text{sqrt}(\text{RMSE}) =$

$$\sqrt{\frac{\sum_{i=1}^n (\hat{y}_i - y_i)^2}{n}}$$

```
[34] from sklearn.model_selection import cross_val_score, ShuffleSplit  
cvf = ShuffleSplit(n_splits=100, test_size=0.25)  
scores = cross_val_score(model, X, y, cv=cvf,  
                        scoring='neg_root_mean_squared_error')  
print(f'100 times mean RMSE: {-scores.mean():.3f}')
```

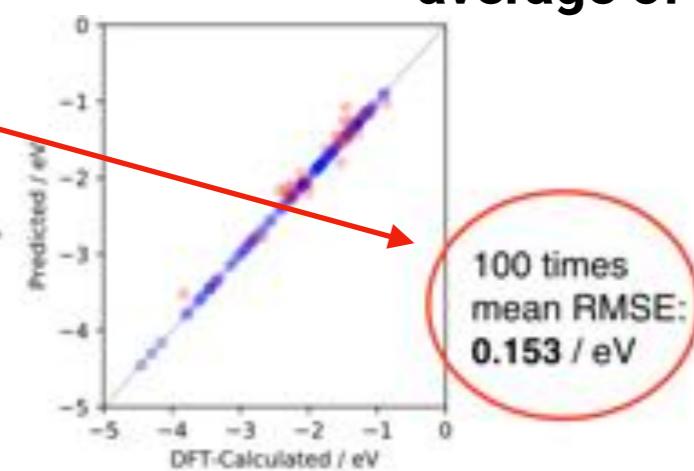
```
100 times mean RMSE: 0.153
```

Do this calculation over
100 random splits into 75%
/ 25% subsets, and take the
average of 100 RMSEs.

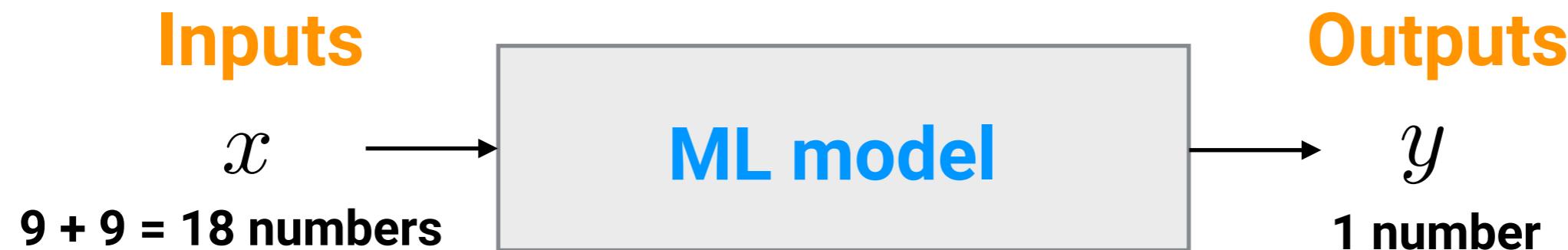
	Fe	Co	Ni	Cu	Ru	Rh	Pd	Ag	Ir	Pt	Au
Fe	-0.92		-0.96	-0.97	-1.65	-1.64	-2.24		-1.87	-2.4	-3.11
Co			-1.37	-1.23		-2.12	-2.82	-2.53	-2.26		-3.56
Ni	-0.33	-1.18			-1.92	-2.03		-2.43	-2.15	-2.82	-3.39
Cu	-2.42		-2.49	-2.67	-2.89	-2.94				-3.82	-4.63
Ru	-1.11	-1.04	-1.12		-1.41		-1.88	-1.81	-1.54		-2.27
Rh	-1.42	-1.32		-1.51	-1.7	-1.73	-2.12	-1.81	-1.7	-2.18	-2.3
Pd	-1.47	-1.29	-1.29	-1.03		-1.58	-1.83	-1.68	-1.52	-1.79	
Ag	-3.75	-3.56	-3.62		-3.8		-4.03		-3.5	-3.93	-4.51
Ir	-1.78	-1.71	-1.78	-1.55		-2.14	-2.53	-2.2	-2.11	-2.6	-2.7
Pt			-1.71	-1.47	-2.13	-2.01	-2.23	-2.06	-1.96		-2.33
Au	-3.03	-2.82	-2.85		-2.89		-3.44				-3.56

gradient boosting
w/ 6 descriptors

training sets (75%)
test sets (25%)



Ignoring all problem-specific data preparation...



[8, 2.66, 26, 55.85, 4, 1.83, 7.90, 247.3, 7.87,
11, 2.67, 29, 63.55, 4, 1.90, 7.73, 203.5, 8.96] -1.05

```
[23] from sklearn.ensemble import GradientBoostingRegressor  
model = GradientBoostingRegressor()  
model.fit(X_train, y_train)
```



"machine learning" part is only here!

<https://scikit-learn.org/stable/>

learn Install User Guide API Examples More ▾ Go

scikit-learn

Machine Learning in Python

Getting Started Release Highlights for 0.23 GitHub

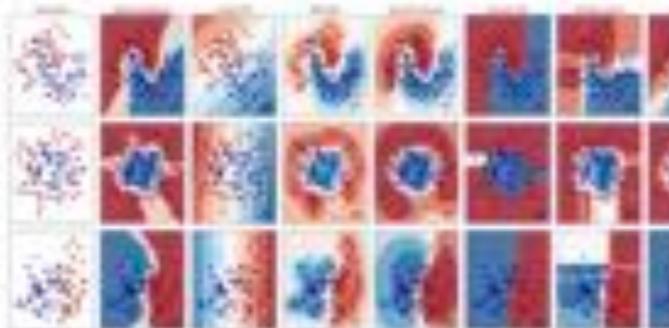
- Simple and efficient tools for predictive data analysis
- Accessible to everybody, and reusable in various contexts
- Built on NumPy, SciPy, and matplotlib
- Open source, commercially usable - BSD license

Classification

Identifying which category an object belongs to.

Applications: Spam detection, image recognition.

Algorithms: SVM, nearest neighbors, random forest, and more...



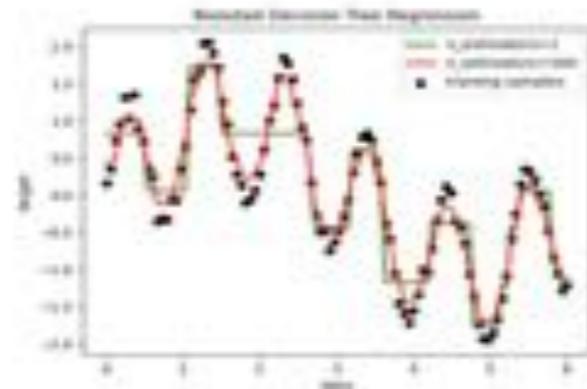
[Examples](#)

Regression

Predicting a continuous-valued attribute associated with an object.

Applications: Drug response, Stock prices.

Algorithms: SVR, nearest neighbors, random forest, and more...



[Examples](#)

Clustering

Automatic grouping of similar objects into sets.

Applications: Customer segmentation, Grouping experiment outcomes

Algorithms: k-Means, spectral clustering, mean-shift, and more...



[Examples](#)

Dimensionality reduction

Reducing the number of random variables to consider.

Applications: Visualization, increased efficiency

Algorithms: k-Means, feature selection, non-negative matrix factorization, and more...

Model selection

Comparing, validating and choosing parameters and models.

Applications: Improved accuracy via parameter tuning

Algorithms: grid search, cross validation,

Preprocessing

Feature extraction and normalization.

Applications: Transforming input data such as text for use with machine learning algorithms.

Algorithms: preprocessing, feature extraction, and more...

Aug 26: 10:30~12:00 (90min)

1. What is "machine learning"?
2. Why does it matter to chemists?
3. Let's try it in your browser (with no setup!)

Aug 26: 13:00~14:30 (90min)

4. Five things all beginners should know
 - "The quality of your inputs decide the quality of your output"
 - Training / validation / test data
 - Tuning hyperparameters
 - Identification and design of input variables (or "descriptors")
 - "Correlation does not imply causation"
5. Standard pipeline and deep learning
6. Current efforts and future directions