Linear Regression and Gradient Descent

Prof. Mingkui Tan

SCUT Machine Intelligence Laboratory (SMIL)

Contents

1 Introduction to Machine Learning

2 Linear Regression

3 Closed-form Solution

4 Gradient Descent

Contents

1 Introduction to Machine Learning

2 Linear Regression

3 Closed-form Solution

4 Gradient Descent

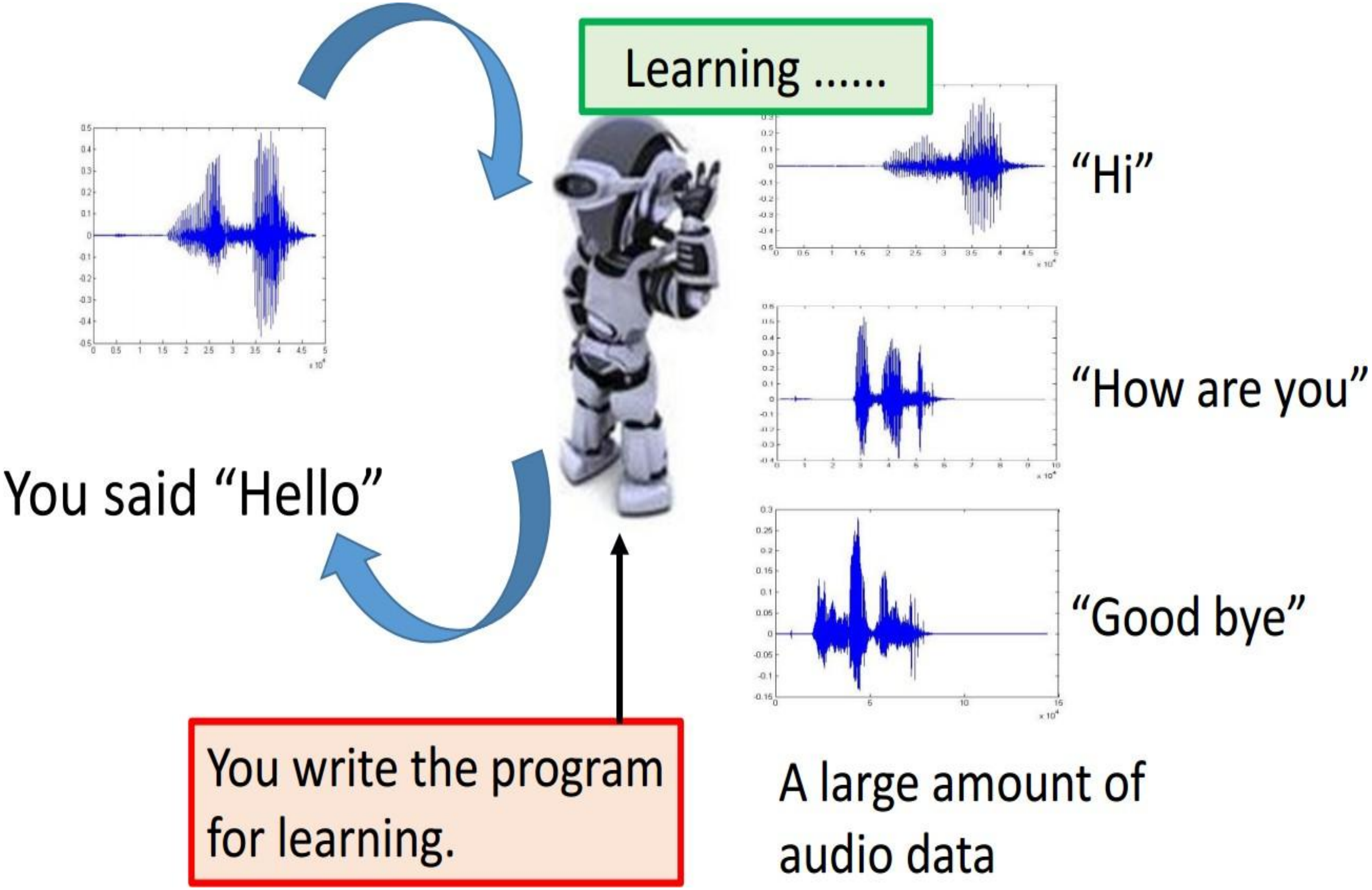
Introduction to Machine Learning

What is Machine Learning?

Machine Learning composes of three parts:

- Data
- Model
- Loss Function

Introduction to Machine Learning



Learning ......

“Hi”

“How are you”

“Good bye”

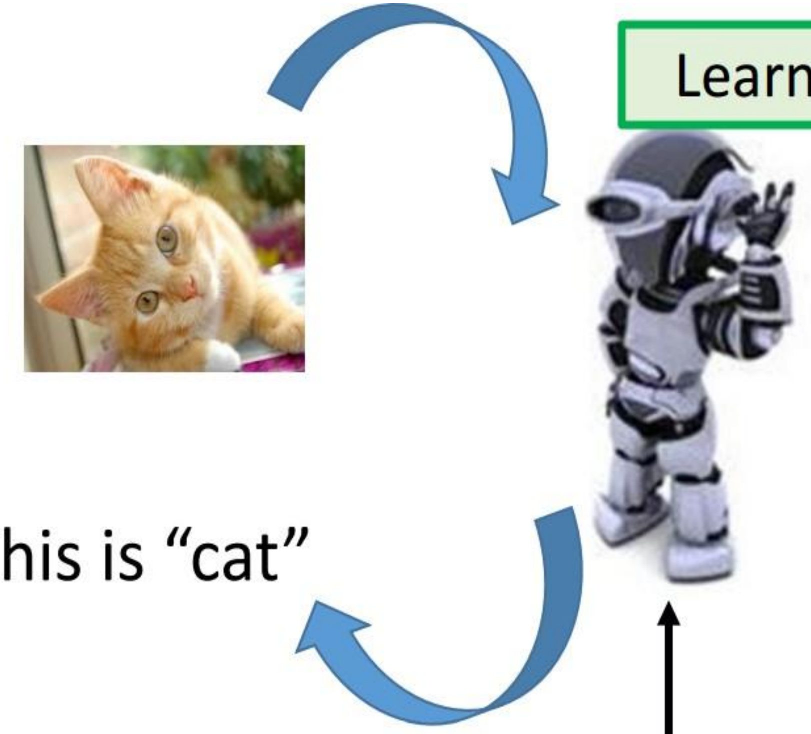
You said “Hello”

You write the program for learning.

A large amount of audio data

Speech Recognition

Introduction to Machine Learning



Learning ......

This is “cat”

You write the program for learning.

“monkey”

“cat”

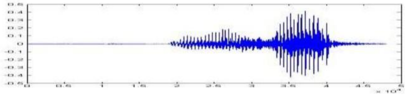
“dog”

A large amount of images

Introduction to Machine Learning

Machine Learning ≈ Looking for a Function

- Speech Recognition

$f($  $)=$ “How are you”

- Image Recognition

$f($ $)=$ “Cat”

- Playing Go

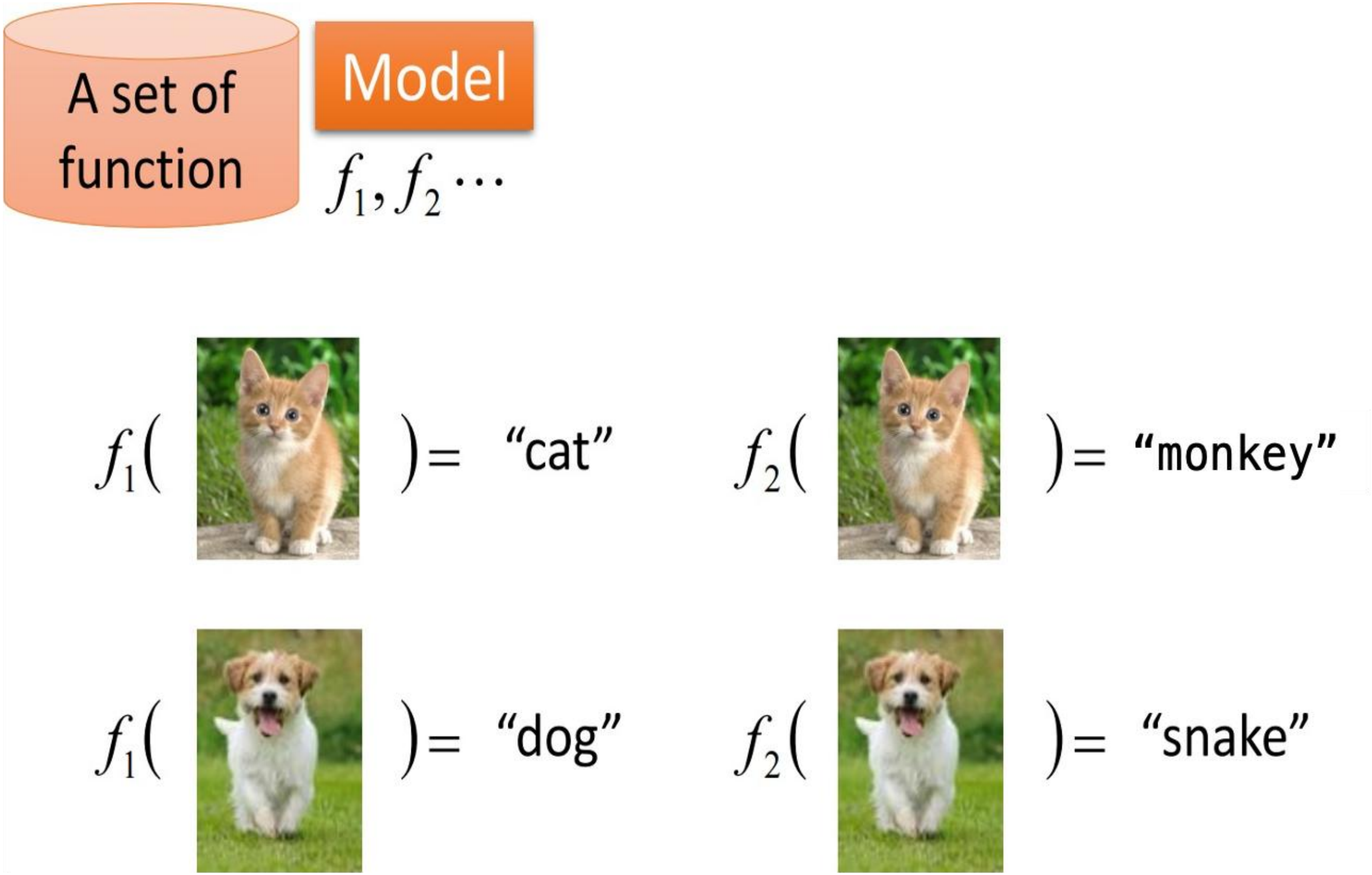
$f($ $)=$ “5-5” (next move)

- Dialogue System

$f($ “Hi” $)=$ “Hello”

(what the user said) (system response)

Introduction to Machine Learning



A set of function

Model

$f_1, f_2 \cdots$

$f_1($ $)=$ "cat"

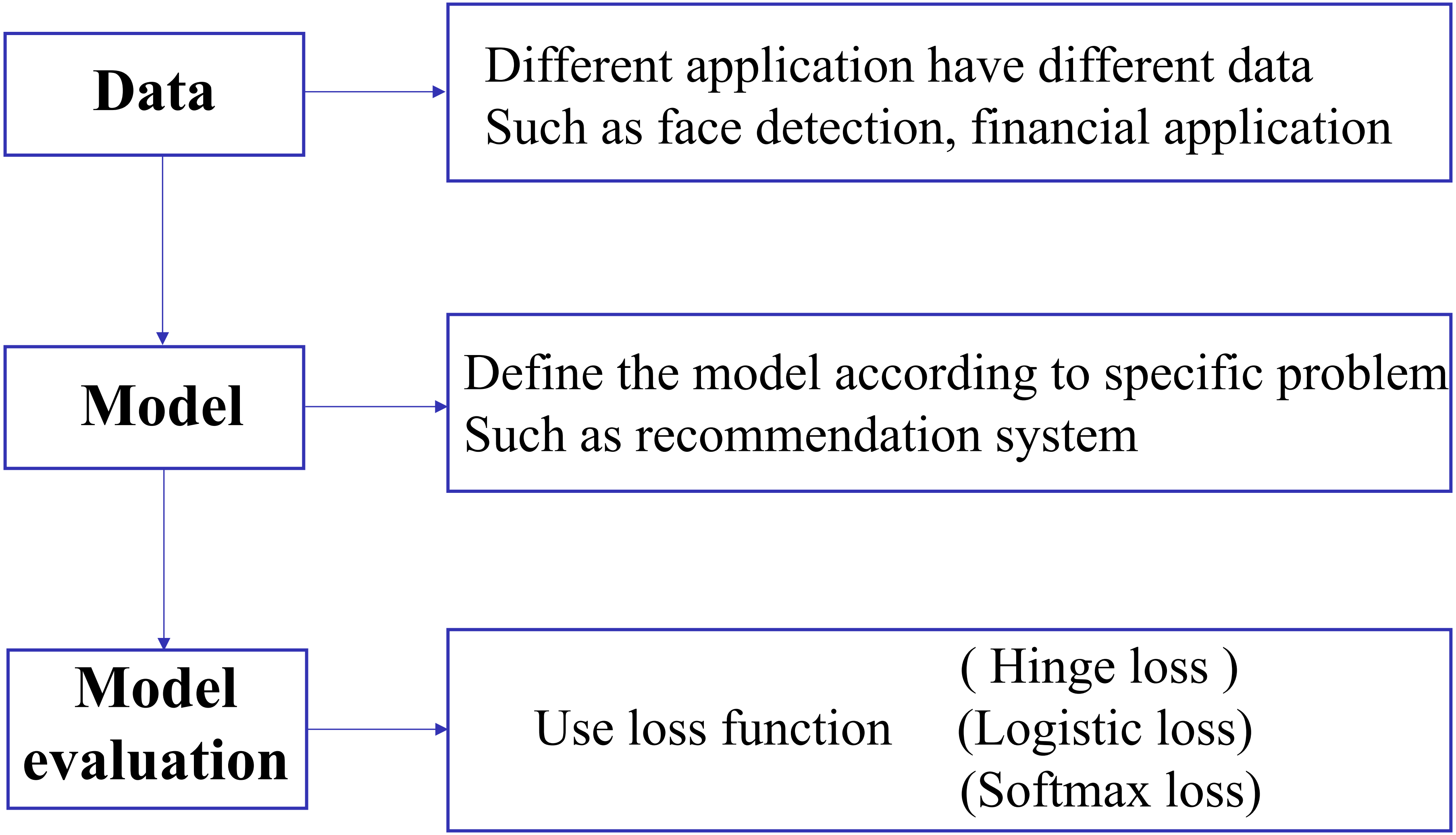
$f_2($ $)=$ "monkey"

$f_1($ $)=$ "dog"

$f_2($ $)=$ "snake"

Image Recognition

Three Main Elements of Machine Learning



Data

Different application have different data
Such as face detection, financial application

Model

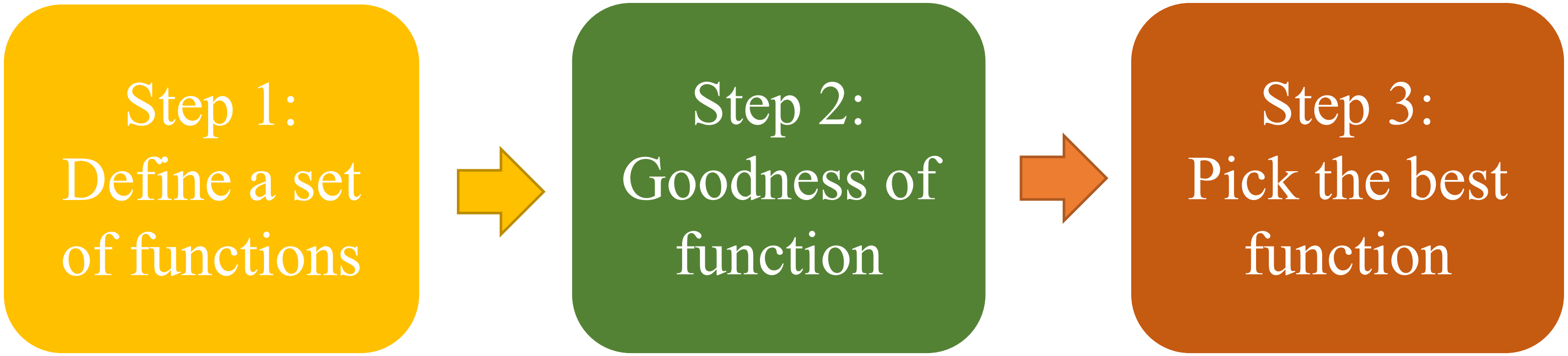
Define the model according to specific problem
Such as recommendation system

Model evaluation

Use loss function (Hinge loss)
(Logistic loss)
(Softmax loss)

Introduction to Machine Learning

Machine Learning is so simple…

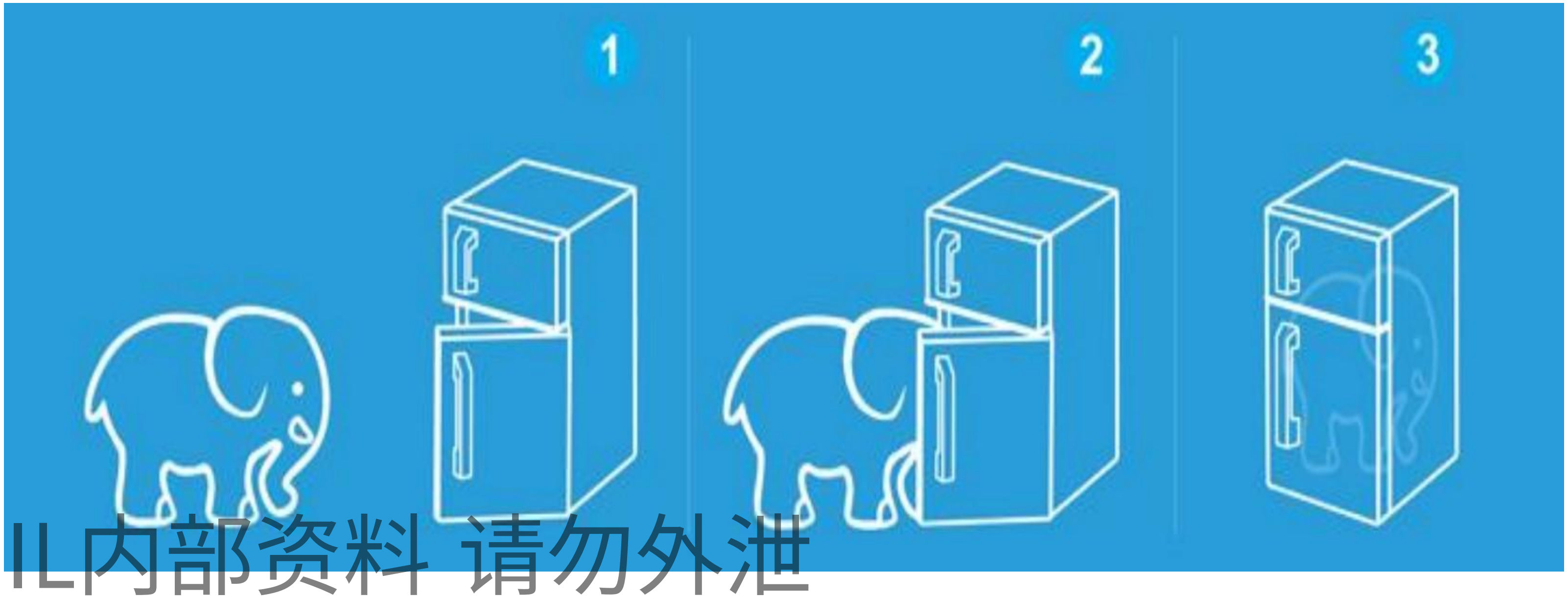


Step 1:
Define a set of functions

Step 2:
Goodness of function

Step 3:
Pick the best function

Just like putting an elephant into the fridge…



Introduction to Machine Learning

- Use a function to predict y:

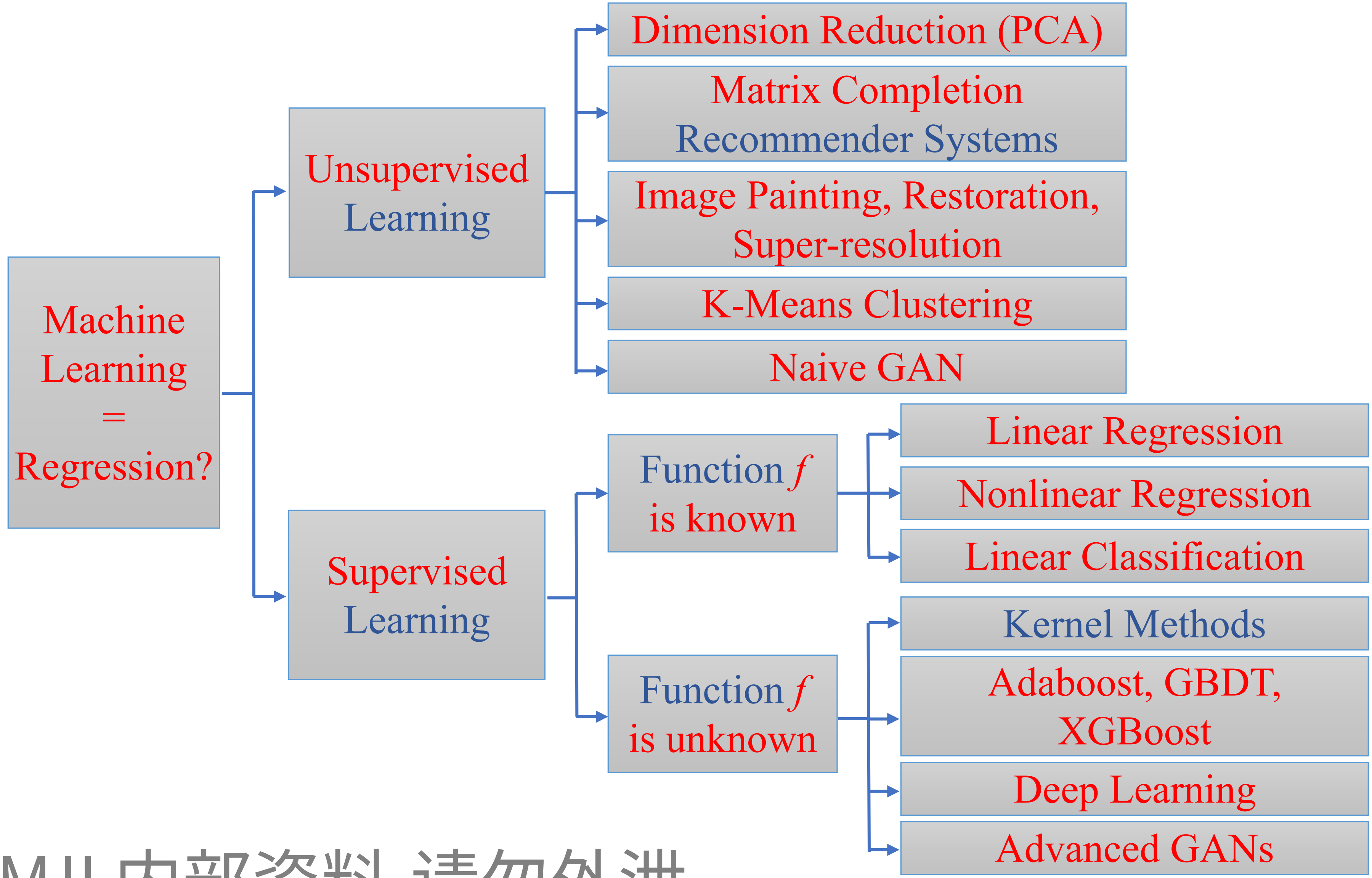
$$\hat{y} = f(x)$$

- However, the prediction may be inconsistent with the ground-truth
- Calculate the difference by loss function:

$$\mathcal{L}_{\mathcal{D}}(\mathbf{W}) = \sum_{i=1}^{n} l(\hat{y}_i, y_i)$$

where $\mathcal{D}$ refers to data and $\mathbf{W}$ refers to parameter

Introduction to Machine Learning



Machine Learning = Regression?

Unsupervised Learning

Dimension Reduction (PCA)

Matrix Completion
Recommender Systems

Image Painting, Restoration, Super-resolution

K-Means Clustering

Naive GAN

Supervised Learning

Function *f* is known

Linear Regression

Nonlinear Regression

Linear Classification

Function *f* is unknown

Kernel Methods

Adaboost, GBDT, XGBoost

Deep Learning

Advanced GANs

Supervised Machine Learning

- **Supervised learning:**

learning a model/function *f* from labeled training data

Labeled data

cat

dog

Unlabeled data

Column Vector

Data:

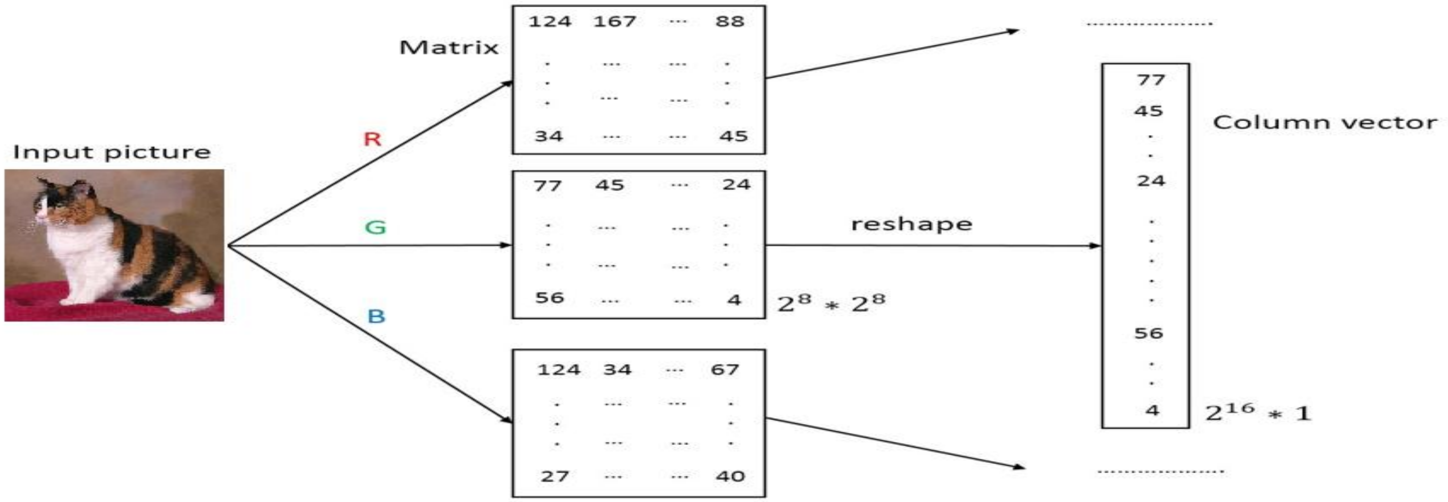
$$\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{n}$$

$\mathbf{x}$ is the input, which is usually presented as a column vector

y is the output, for example, a person's name

n is the number of samples

For example, $\mathbf{x}$ can be a picture stored as a matrix:



Typical Datasets for Supervised Learning

Libsvm dataset

- It contains many classification, regression, multi-label and string data sets
  https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/
- You can use LIBSVM, a package, with these sets
  http://www.csie.ntu.edu.tw/~cjlin/libsvm
- You can also use LIBLINEAR, a linear classifier, with the sets
  https://www.csie.ntu.edu.tw/~cjlin/liblinear/#document
- Other tutorials you can read are as follows:

Tools: https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/

Guide: https://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf

Introduction to the Format of LIBSVM

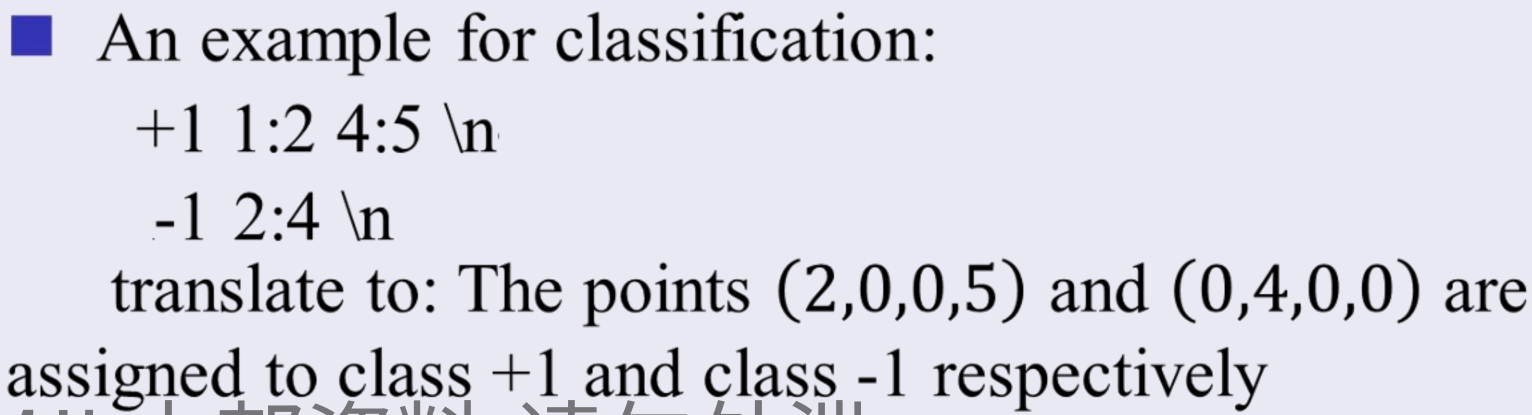
Two properties of data:

- The number of features is large
- Each instance is sparse for most feature values are zero

Sparse format:

<label1> <index1>:<value1> <index2>:<value2> …

<label2> <index1>:<value1> <index2>:<value2> …



- An example for classification:

    +1 1:2 4:5 \n

    -1 2:4 \n

    translate to: The points (2,0,0,5) and (0,4,0,0) are assigned to class +1 and class -1 respectively

Contents

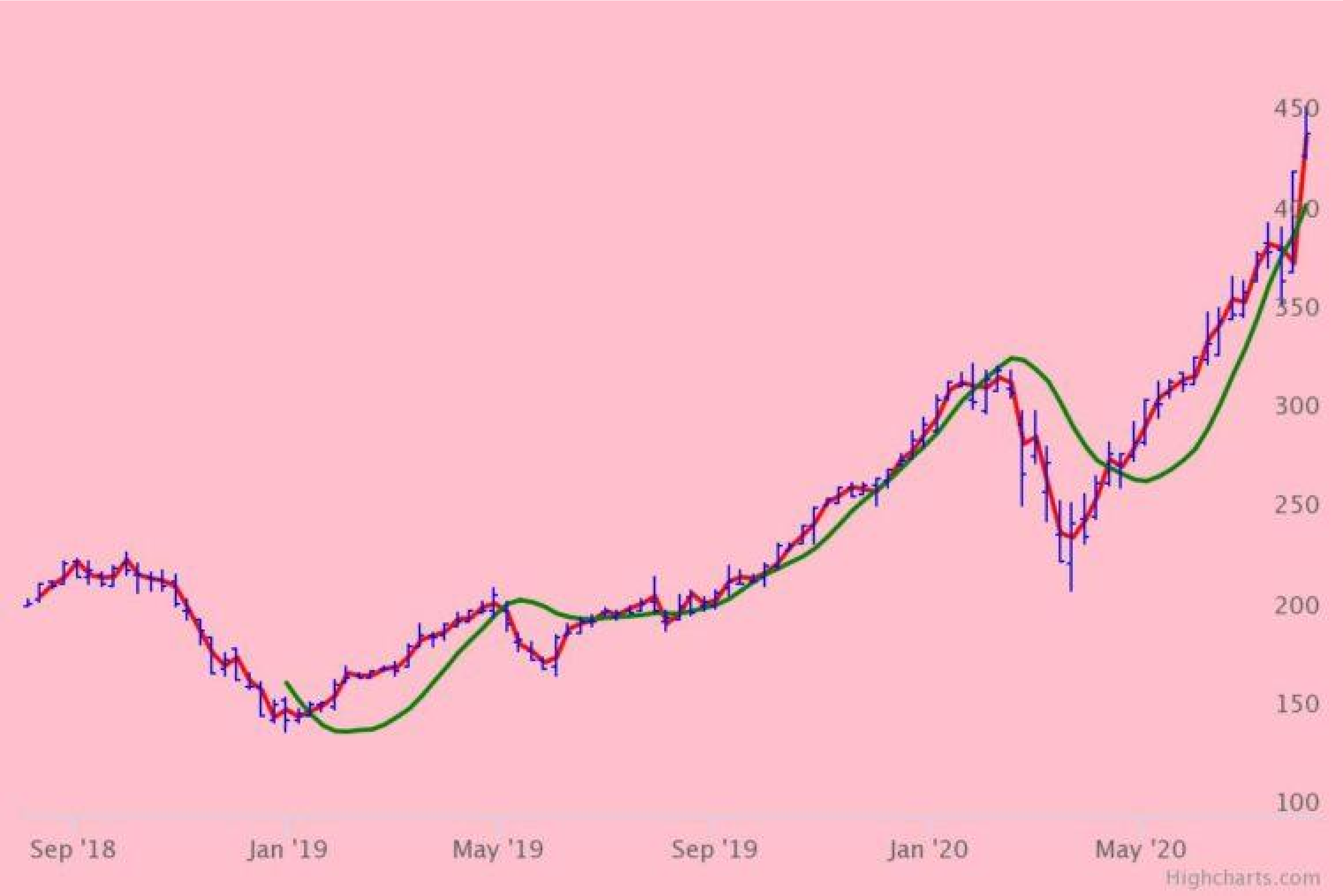
1 Introduction to Machine Learning

2 Linear Regression

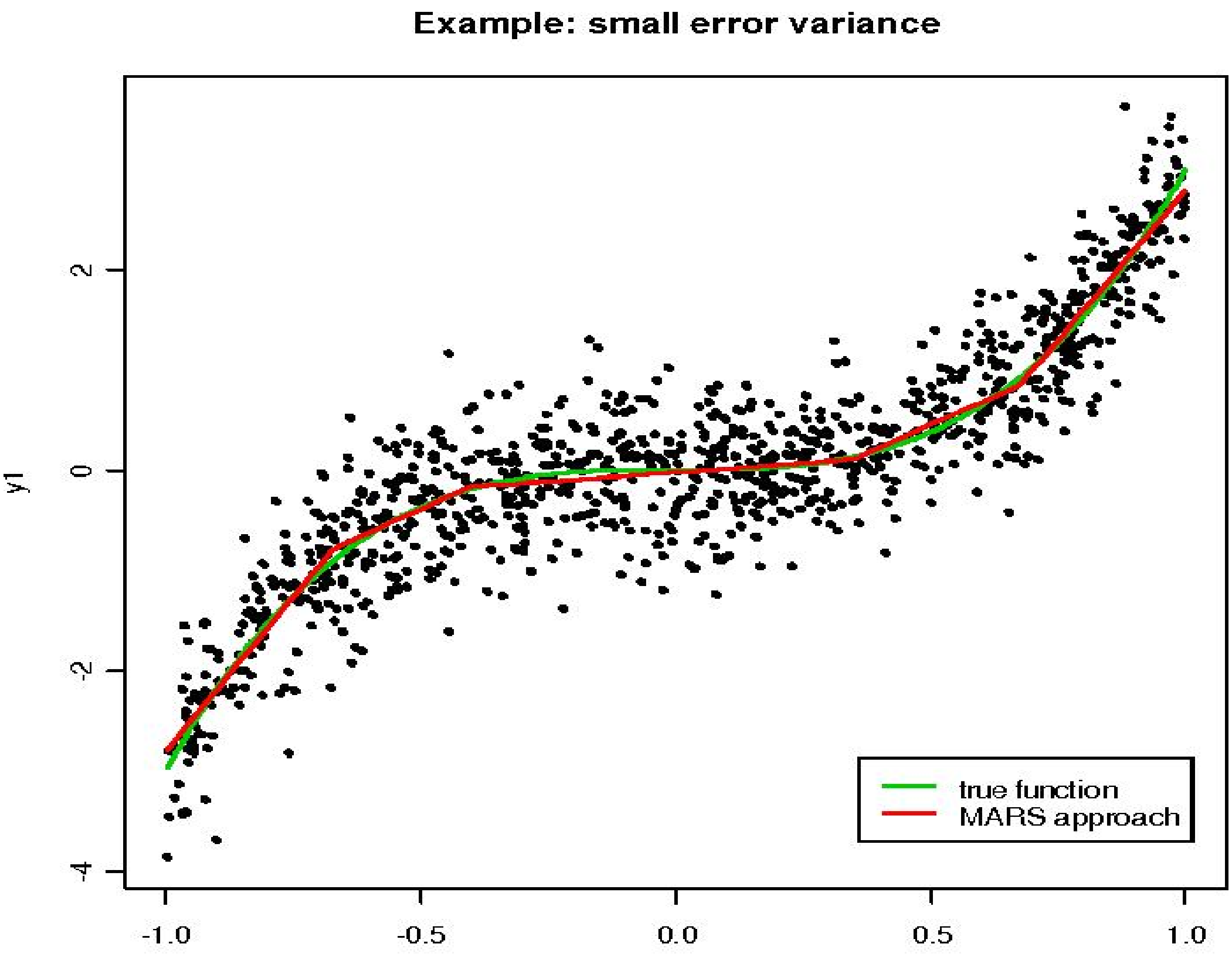
3 Closed-form Solution

4 Gradient Descent

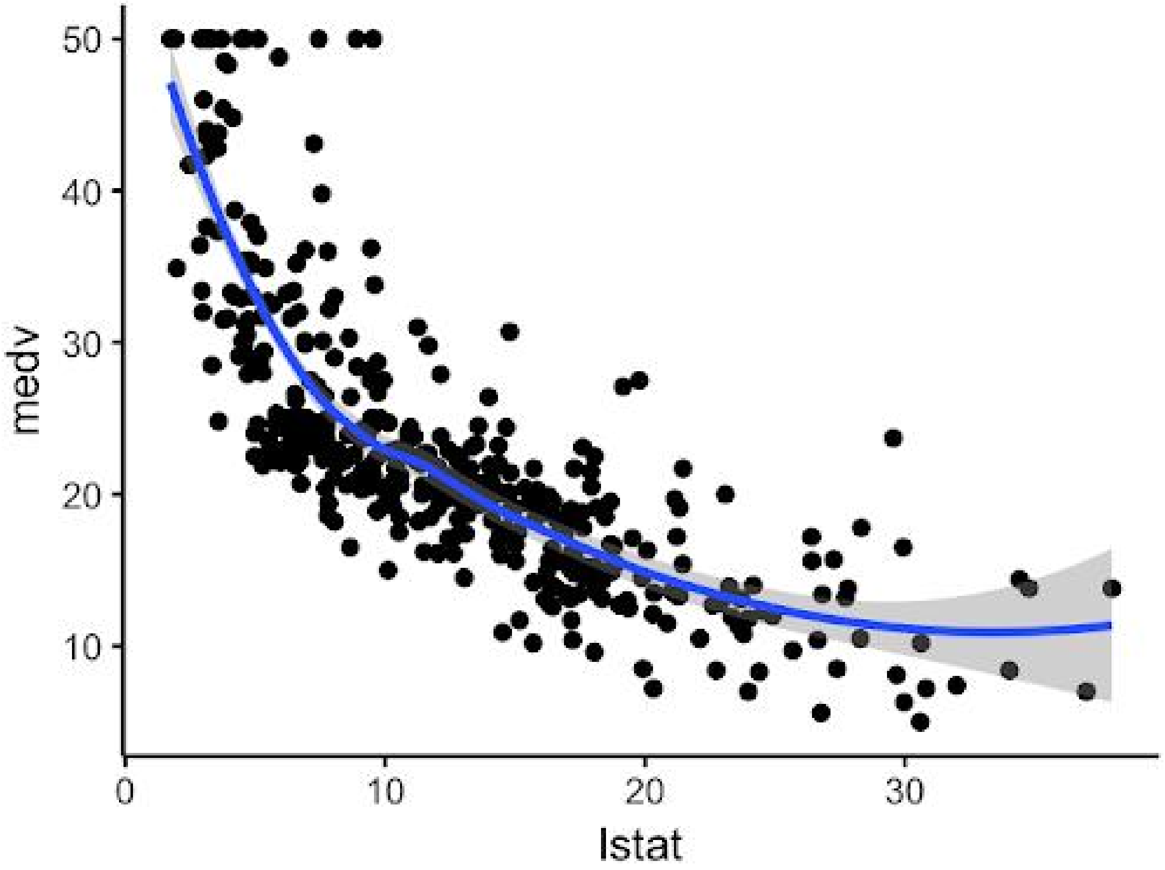
Regression


450
400
350
300
250
200
150
100
Sep '18
Jan '19
May '19
Sep '19
Jan '20
May '20
Highcharts.com

Regression


Example: small error variance
y1
2
0
-2
-4
-1.0
-0.5
0.0
0.5
1.0
true function
MARS approach

Regression



Problem Setup for Regression

- **Inputs**

  Input space: $\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^{N}, \mathbf{x}_i \in \mathbb{R}^m$

  N is the number of data samples

  $\mathbf{x}_i$ includes m features

- **Outputs**

  Output space: $\mathcal{Y} = \{y_i\}_{i=1}^{N}, y_i \in \mathbb{R}$

- **Goal**

  Learn a hypothesis / model $f: \mathcal{X} \to \mathcal{Y}$

Regression

Loss:

- Absolute value loss:

$$l(\hat{y}_i, y_i\ ) = |\hat{y}_i - y_i|$$

- Least squares loss:

$$l(\hat{y}_i, y_i) = \frac{1}{2}(\hat{y}_i - y_i)^2$$

Total loss function:

$$\mathcal{L}_{\mathcal{D}}(\mathbf{W}) = \sum_{i=1}^{n} l(\hat{y}_i, y_i)$$

Regression

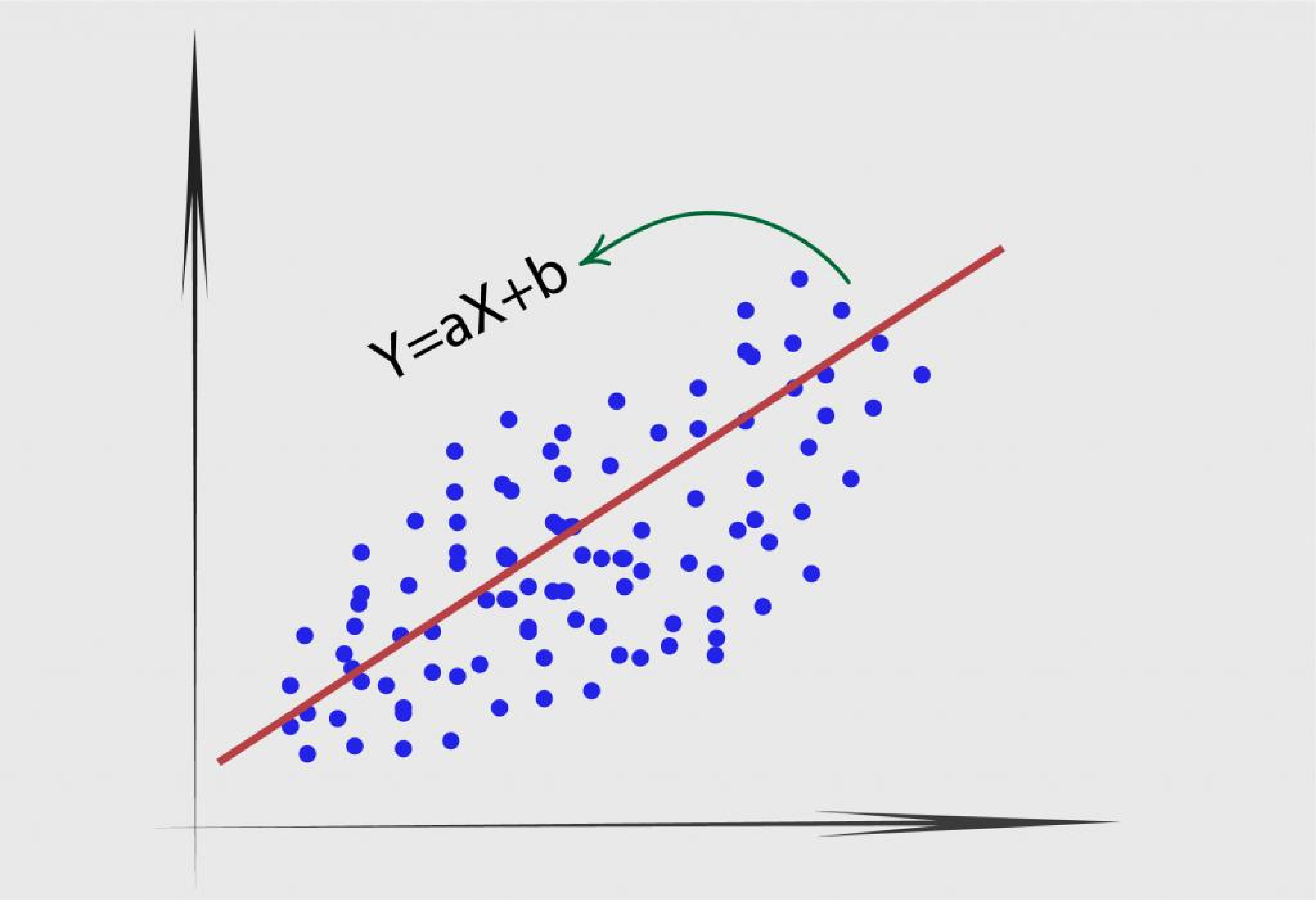
- The smaller value of $\mathcal{L}_{\mathcal{D}}$ is better, and loss function $\mathcal{L}_{\mathcal{D}}$ plays a major role in machine learning

Target:

- Find the best � by solving the following optimization problem:

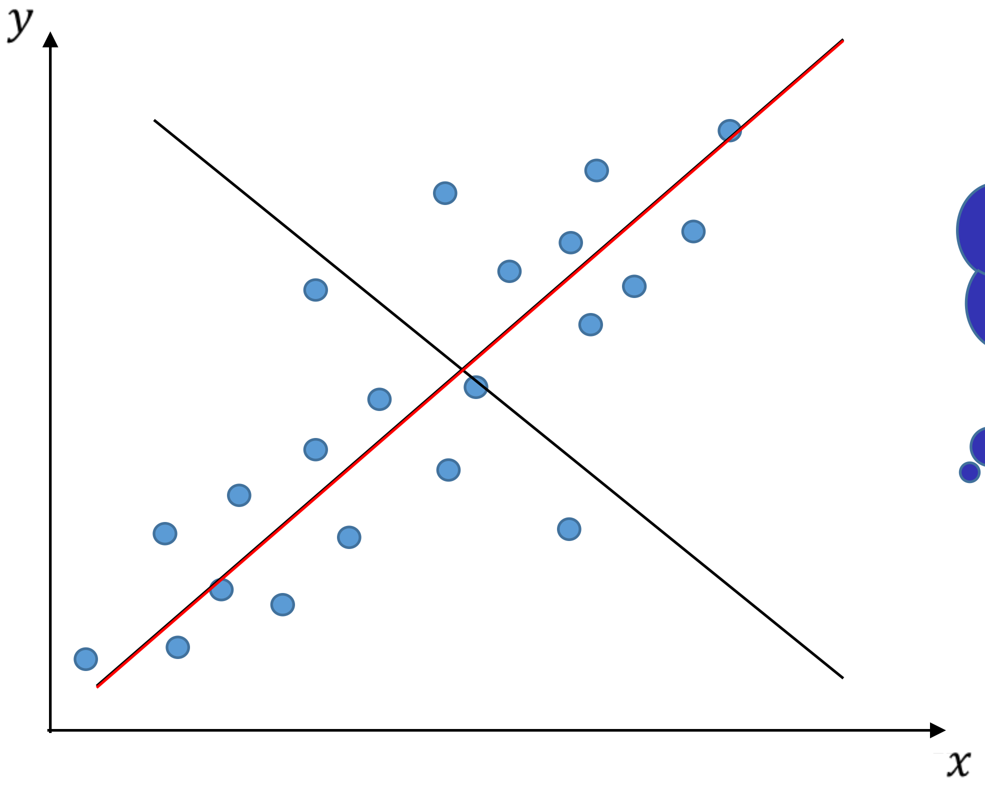
$$f^* = \underset{f}{\operatorname{argmin}} \sum_{i=1}^{n} l(f(x), y_i)$$

Linear Regression



Linear Regression

Simple linear regression describes the linear relationship between a variable x and a response variable y

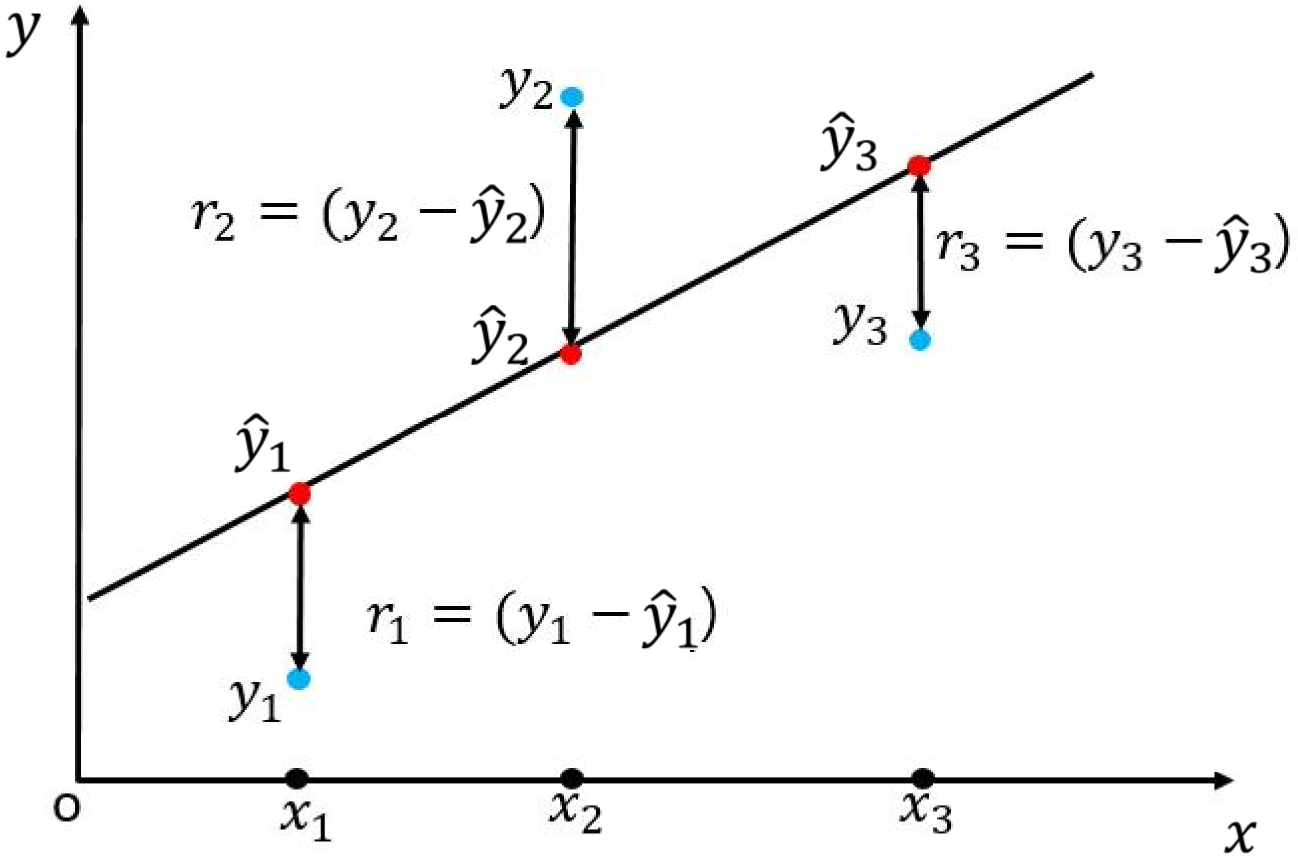


Which one is better?

Simple linear 1-D regression

Linear Regression

- **What makes a good model?**



y
y_2
$\hat{y}_3$
$r_2 = (y_2 - \hat{y}_2)$
$r_3 = (y_3 - \hat{y}_3)$
$\hat{y}_2$
y_3
$\hat{y}_1$
$r_1 = (y_1 - \hat{y}_1)$
y_1
O
x_1
x_2
x_3
x

Linear Regression

Learn $f(\mathbf{x}; \mathbf{w}, b)$ with

- Parameters: $\mathbf{w} \in \mathbb{R}^m, b \in \mathbb{R}$
- Input: $\mathbf{x}$ where $x_i \in \mathbb{R}$, features for $i \in \{1, \cdots, m\}$
- Model Function:

$$
\begin{aligned}
f(\mathbf{x}; \mathbf{w}, b) &= w_1 x_1 + \cdots + w_m x_m + b \\
&= \sum_{i=1}^{m} w_i x_i + b \\
&= \mathbf{w}^{\mathrm{T}} \mathbf{x} + b
\end{aligned}
$$

Performance Measure for Regression

- Least squared loss

$$\mathcal{L}_{\mathcal{D}}(\mathbf{w}, b) = \frac{1}{2}\sum_{i=1}^{n}(y_i - f(\mathbf{x}_i; \mathbf{w}, b))^2$$

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

Training: find minimizer of least squared loss

$$\mathbf{w}^*, b^* = \operatorname*{argmin}_{\mathbf{w}, b} \mathcal{L}_{\mathcal{D}}(\mathbf{w}, b)$$

Matrix Presentation for Loss Function

In order to simplify our proof, we introduce augmented matrix and augmented vector and still represent them by **w** and **X**.

i.e.
$$\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_i, \dots, \mathbf{x}_n)^{\mathrm{T}}$$
$$\mathbf{x}_i = (x_{i1}, x_{i2}, \dots, x_{im}, 1)$$
$$\mathbf{w} = (w_1, w_2, \dots, w_m, b)^{\mathrm{T}}$$

Loss function:

$$\mathcal{L}_D(w) = \frac{1}{2} \|Y - Xw\|_2^2$$

$$\textit{where } X = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} x_{11} & \cdots & x_{1m} & 1 \\ \vdots & \ddots & \vdots & \vdots \\ x_{n1} & \cdots & x_{nm} & 1 \end{pmatrix}, \quad Y = \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix}$$

Matrix Presentation for Loss Function

- Simple Proof:

$$\mathcal{L}_D(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^{n}(y_i - \mathbf{x}_i\mathbf{w})^2$$

$$= \frac{1}{2}\begin{bmatrix} y_1 - \mathbf{x}_1^{\mathrm{T}}\mathbf{w} \\ \vdots \\ y_n - \mathbf{x}_n^{\mathrm{T}}\mathbf{w} \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} y_1 - \mathbf{x}_1^{\mathrm{T}}\mathbf{w} \\ \vdots \\ y_n - \mathbf{x}_n^{\mathrm{T}}\mathbf{w} \end{bmatrix}$$

$$= \frac{1}{2}\left(\begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} - \begin{bmatrix} \mathbf{x}_1^{\mathrm{T}} \\ \vdots \\ \mathbf{x}_n^{\mathrm{T}} \end{bmatrix}\mathbf{w}\right)^{\mathrm{T}} \left(\begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} - \begin{bmatrix} \mathbf{x}_1^{\mathrm{T}} \\ \vdots \\ \mathbf{x}_n^{\mathrm{T}} \end{bmatrix}\mathbf{w}\right)$$

$$= \frac{1}{2}(\mathbf{y} - \mathbf{X}\mathbf{w})^{\mathrm{T}}(\mathbf{y} - \mathbf{X}\mathbf{w})$$

$$= \frac{1}{2}\|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2$$

Contents

1 Introduction to Machine Learning

2 Linear Regression

3 Closed-form Solution

4 Gradient Descent

Naïve Linear Regression:

Objective function for linear regression:

$$\mathcal{L}_D(w) = \frac{1}{2}\|Y - Xw\|_2^2$$

$$where\ X = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} x_{11} & \cdots & x_{1m} & 1 \\ \vdots & \ddots & \vdots & \vdots \\ x_{n1} & \cdots & x_{nm} & 1 \end{pmatrix},\quad Y = \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix}$$

Training: find the minimizer of $L_D\ (w, b)$

$$\mathbf{w}^*, b^* = \underset{\mathbf{w}, b}{\operatorname{argmin}}\, \mathcal{L}_D(\mathbf{w}, b)$$

Analytical Solution

How to address the linear regression question?

- Closed-form solution to linear regression:

$$\mathcal{L}_D(\mathbf{w}) = \frac{1}{2}(\mathbf{y}-\mathbf{X}\mathbf{w})^{\mathrm{T}}(\mathbf{y}-\mathbf{X}\mathbf{w}) \text{ , Let } \mathbf{a} = \mathbf{y}-\mathbf{X}\mathbf{w},$$

$$\begin{aligned}\frac{\partial \mathcal{L}_D(\mathbf{w})}{\partial \mathbf{w}} &= \frac{\partial \mathbf{a}}{\partial \mathbf{w}} \frac{\partial(\frac{1}{2}\mathbf{a}^T\mathbf{a})}{\partial \mathbf{a}} \\ &= \frac{1}{2}\frac{\partial \mathbf{a}}{\partial \mathbf{w}}(2\mathbf{a}) \\ &= \frac{\partial(\mathbf{y}-\mathbf{X}\mathbf{w})}{\partial \mathbf{w}}(\mathbf{y}-\mathbf{X}\mathbf{w}) \\ &= -\mathbf{X}^{\mathrm{T}}(\mathbf{y}-\mathbf{X}\mathbf{w})\end{aligned}$$

Since $\mathcal{L}_D(\mathbf{w})$ is a convex function, $\frac{\partial \mathcal{L}_D(\mathbf{w})}{\partial \mathbf{w}} = 0$ derive $\mathbf{w}^*$

Analytical Solution

- Assuming $|\mathbf{X}^{\mathrm{T}}\mathbf{X}| \neq 0$
- Let $\dfrac{\partial \mathcal{L}_{\mathcal{D}}(\mathbf{w})}{\partial \mathbf{w}} = -\mathbf{X}^{\mathrm{T}}\mathbf{y} + \mathbf{X}^{\mathrm{T}}\mathbf{X}\mathbf{w} = 0$

$$\Longrightarrow \mathbf{X}^{\mathrm{T}}\mathbf{X}\mathbf{w} = \mathbf{X}^{\mathrm{T}}\mathbf{y}$$

$$\Longrightarrow \mathbf{w} = (\mathbf{X}^{\mathrm{T}}\mathbf{X})^{-1}\mathbf{X}^{\mathrm{T}}\mathbf{y}$$

- Solve the optimal parameter $\mathbf{w}^*$

$$\mathbf{w}^* = \underset{\boldsymbol{w}}{\operatorname{argmin}}\, \mathcal{L}_{\mathcal{D}}(\mathbf{w}) = (\mathbf{X}^{\mathrm{T}}\mathbf{X})^{-1}\mathbf{X}^{\mathrm{T}}\mathbf{y}$$

Challenges about Analytical Solution

There are two challenges left to address about the analytical solution $\mathbf{w}^* = (\mathbf{X}^{\mathrm{T}}\mathbf{X})^{-1}\mathbf{X}^{\mathrm{T}}\mathbf{y}$:

- Many matrices are not invertible

  Necessary and Sufficient Condition:

  If $\mathbf{X}$ is a matrix of m rows and n columns $(n \leq m)$,

$$rank(\mathbf{X}) \leq n$$

- The inverse of a large matrix needs huge memory, which takes $O(m^3)$ to compute.

Issue of the Closed-form Solution

- Closed-form solution: $\mathbf{w}^* = (\mathbf{X}^{\mathrm{T}}\mathbf{X})^{-1}\mathbf{X}^{\mathrm{T}}\mathbf{y}$
  - The matrix $(X^T X)^{-1}$ may not be invertible, which means the matrix may have infinite number of solutions!

$$\mathbf{w}^* = (\mathbf{X}^{\mathrm{T}}\mathbf{X})^{-1}\mathbf{X}^{\mathrm{T}}\mathbf{y} \xrightarrow{(\mathbf{X}^{\mathrm{T}}\mathbf{X} + \lambda\mathbf{I}\,)^{-1}} \mathbf{w}^* = (\mathbf{X}^{\mathrm{T}}\mathbf{X} + \lambda\mathbf{I}\,)^{-1}\mathbf{X}^{\mathrm{T}}\mathbf{y}$$

Regularized Least Square （RLS） Regression

- Impose regularization on **w**:

$$\mathcal{L}_D(\mathbf{w}) = \frac{\lambda}{2}||\mathbf{w}||_2^2 + \frac{1}{2}\sum_{i=1}^{n}\left(y_i - f(\mathbf{x}_i; w)\right)^2$$

$$= \frac{\lambda}{2}||\mathbf{w}||_2^2 + \frac{1}{2}||\mathbf{y} - \mathbf{X}\boldsymbol{w}||_2^2$$

- Here, $\frac{\lambda}{2}||\mathbf{w}||_2^2$ is called Regularizer, λ is called **trade-off parameter** or **regularization parameter**

Training: find minimizer of least squared loss

$$\mathbf{w}^* = \underset{w}{argmin}\,\mathcal{L}_D(\mathbf{w})$$

Closed-form Solution for Regularized Least Square(RLS)

- First-order condition of the optimal solution:

$$\frac{\partial \mathcal{L}(\mathbf{w})}{\partial \boldsymbol{w}}=0$$

- For the Least Regression problem, we have

$$\begin{aligned}\frac{\partial \mathcal{L}(\mathbf{w})}{\partial \boldsymbol{w}}&=\lambda\mathbf{w}-\mathbf{X}^{\mathrm{T}}\mathbf{y}+\mathbf{X}^{\mathrm{T}}\mathbf{X}\mathbf{w}=0\\ &\Rightarrow (\lambda\mathbf{I}+\mathbf{X}^{\mathrm{T}}\mathbf{X})\mathbf{w}=\mathbf{X}^{\mathrm{T}}\mathbf{y}\\ &\Rightarrow \mathbf{w}=(\mathbf{X}^{\mathrm{T}}\mathbf{X}+\lambda\mathbf{I})^{-1}\mathbf{X}^{\mathrm{T}}\mathbf{y}\end{aligned}$$

- We obtain the optimal $\mathbf{w}^*$ by

$$\mathbf{w}^*=(\mathbf{X}^{\mathrm{T}}\mathbf{X}+\lambda\mathbf{I})^{-1}\mathbf{X}^{\mathrm{T}}\mathbf{y}$$

Issue of the Closed-form Solution

- Closed-form solution: $\mathbf{w}^* = (\mathbf{X}^{\mathrm{T}}\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^{\mathrm{T}}\mathbf{y}$
  - The inverse of a large matrix needs huge memory
  - The inverse takes $O(m^3)$ complexity to compute

Contents

1 Introduction to Machine Learning

2 Linear Regression

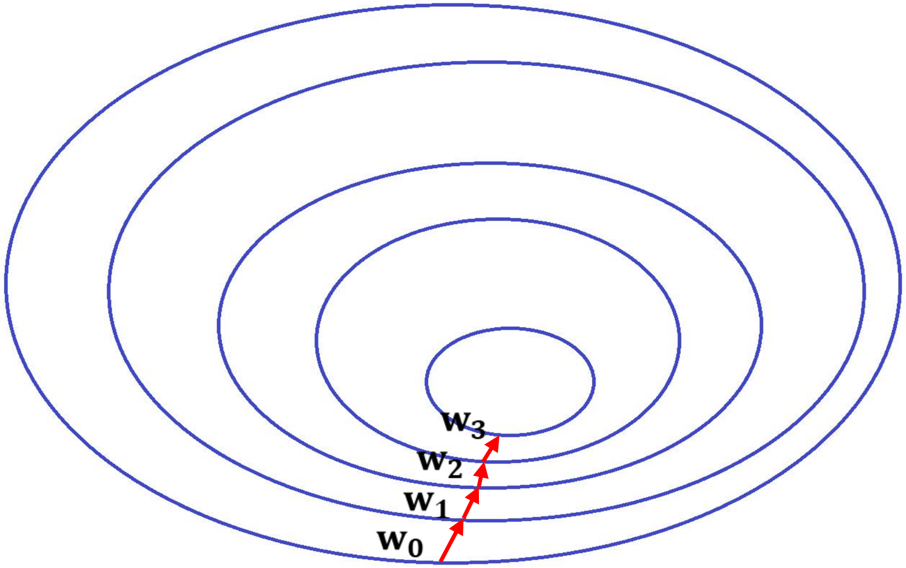
3 Closed-form Solution

4 Gradient Descent

Gradient Descent

- Get the best $\mathbf{w}$ by minimizing a loss function $\mathcal{L}_{\mathcal{D}}(\mathbf{w})$

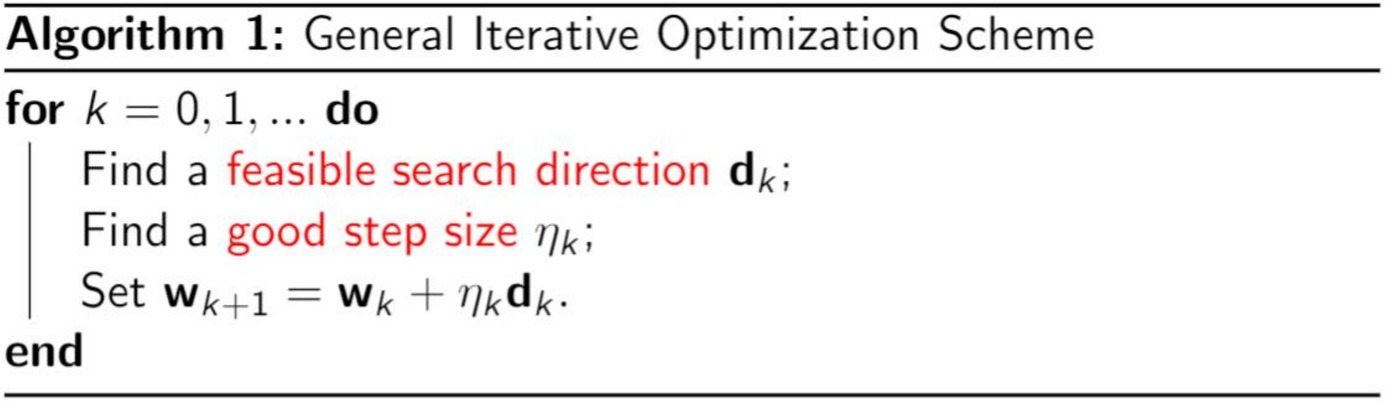
$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \mathcal{L}_{\mathcal{D}}(\mathbf{w})$$



- Do we have other optimization methods in addition to closed-form solution ?

General Optimization Scheme

- General optimization scheme contains 3 iterative steps：



Algorithm 1: General Iterative Optimization Scheme

```
for $k = 0, 1, \ldots$ do
    Find a feasible search direction $\mathbf{d}_k$;
    Find a good step size $\eta_k$;
    Set $\mathbf{w}_{k+1} = \mathbf{w}_k + \eta_k \mathbf{d}_k$.
end
```

- The core questions are:
  - How to find a feasible search direction d ?
  - How to find a good step size η ?
- No matter what kind of problems are, we do just care the above two questions
- The construction of feasible search direction d is problem dependent and can be very complex

Descent Direction

- We use $\mathbf{d} = -\frac{\partial \mathcal{L}_{\mathcal{D}}(\mathbf{w})}{\partial \mathbf{w}}$ as the direction of optimization
- Gradient (vector of partial derivatives)

$$\frac{\partial \mathcal{L}_{\mathcal{D}}(\mathbf{w})}{\partial \mathbf{w}} = \begin{bmatrix} \frac{\partial \mathcal{L}_{\mathcal{D}}(\mathbf{w})}{\partial w_1} \\ \frac{\partial \mathcal{L}_{\mathcal{D}}(\mathbf{w})}{\partial w_2} \\ \vdots \\ \frac{\partial \mathcal{L}_{\mathcal{D}}(\mathbf{w})}{\partial w_m} \end{bmatrix}$$

(We always write a vector into column form)

- Why $\mathcal{L}_{\mathcal{D}}(\mathbf{w}') = \mathcal{L}_{\mathcal{D}}(\mathbf{w} + \eta \mathbf{d}) \leq \mathcal{L}_{\mathcal{D}}(\mathbf{w})$, $\eta \to 0^+$?

Descent Direction

Proof:

By Taylor expansion, when $\eta \to 0^+$:

$$\mathcal{L}_{\mathcal{D}}(\mathbf{w}+\eta\mathbf{d}) = \mathcal{L}_{\mathcal{D}}(\mathbf{w}) + \left(\frac{\partial \mathcal{L}_{\mathcal{D}}(\mathbf{w})}{\partial \mathbf{w}}\right)^{\mathrm{T}} \eta\mathbf{d} + o(\eta\mathbf{d})$$

$$= \mathcal{L}_{\mathcal{D}}(\mathbf{w}) + \eta' \left(\frac{\partial \mathcal{L}_{\mathcal{D}}(\mathbf{w})}{\partial \mathbf{w}}\right)^{\mathrm{T}} \mathbf{d}$$

Note that $\eta^{'} > 0$ and

$$\eta' \left(\frac{\partial \mathcal{L}_{\mathcal{D}}(\mathbf{w})}{\partial \mathbf{w}}\right)^{\mathrm{T}} \mathbf{d} = -\eta' \mathbf{d}^{\mathrm{T}} \mathbf{d} \leq 0$$

We have:

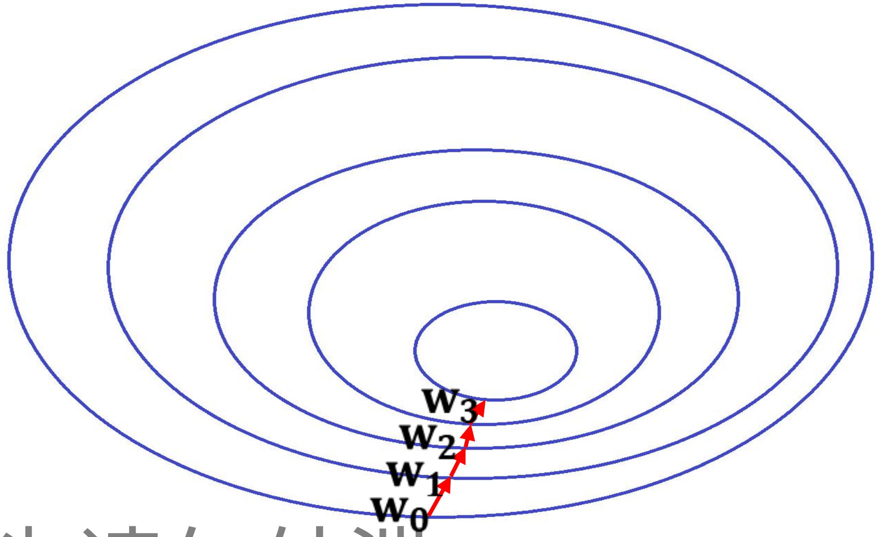
$$\mathcal{L}_{\mathcal{D}}(\mathbf{w}') = \mathcal{L}_{\mathcal{D}}(\mathbf{w}+\eta\mathbf{d}) \leq \mathcal{L}_{\mathcal{D}}(\mathbf{w})$$

Gradient Descent: Update Parameters

Minimize loss by repeated gradient steps (when no closed form):

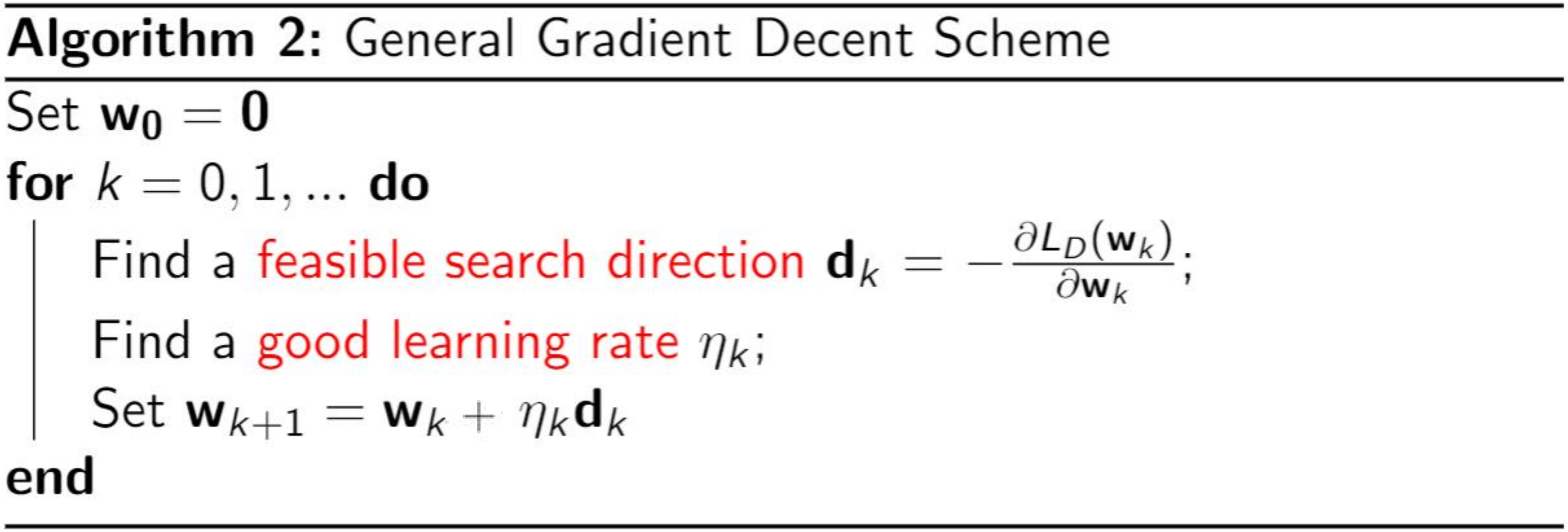
- Compute gradient of loss with respect to parameters $\frac{\partial \mathcal{L}_{\mathcal{D}}(\mathbf{w})}{\partial \mathbf{w}}$
- Update parameters with learning rate η

$$\mathbf{w}' = \mathbf{w} - \eta \frac{\partial \mathcal{L}_{\mathcal{D}}(\mathbf{w})}{\partial \mathbf{w}}$$



General Gradient Decent Scheme

- General gradient decent scheme contains 3 iterative steps：



Algorithm 2: General Gradient Decent Scheme

```
Set $\mathbf{w}_0 = \mathbf{0}$
for $k = 0, 1, \dots$ do
    Find a feasible search direction $\mathbf{d}_k = -\frac{\partial L_D(\mathbf{w}_k)}{\partial \mathbf{w}_k}$;
    Find a good learning rate $\eta_k$;
    Set $\mathbf{w}_{k+1} = \mathbf{w}_k + \eta_k \mathbf{d}_k$
end
```

- Why a good learning rate is necessary?

Appropriate Value of Learning Rate

Learning rate η has a large impact on convergence

- Too large η ⇒ oscillate and may even diverge
- Too small η ⇒ too slow to converge

Adaptive learning rate (For example) :

- Set larger learning rate at the beginning
- Use relatively smaller learning rate in the later epochs
- Decrease the learning rate:

$$\eta_{k+1} = \frac{\eta_k}{k+1}$$

Thank You