机器学习与人工智能
Machine Learning and Artificial Intelligence

Lecture 2
k-Nearest Neighbors & Regression

Yingjie Zhang (张颖婕)

Peking University

yingjiezhang@gsm.pku.edu.cn

2024 Fall

ML Model Types

Criterion Whether or not they are trained with human supervision

Supervised Learning

Fraud detection
Prediction of stock markets

Unsupervised Learning

Customer segmentation
Recommendation

Semi-supervised Learning

Photo-hosting service
Speech analysis
Web-content classification

Reinforcement Learning

Robotics
Go games
Self-driving cars

Data Division

- **Training dataset**: the sample of data used to fit the model
- **Validation Dataset**: the sample of data used to provide an unbiased evaluation of a model fit on the training dataset while tuning model hyperparameters.
- **Test Dataset**: a set of examples used only to assess the performance (i.e. generalization) of a fully specified classifier

Supervised Learning

- **k-Nearest Neighbors**
- **Regressions**
- Decision Trees
- Support Vector Machine
- Naïve Bayes
- …

前四种为 diciminate，**判别式的模型**，x--y

后面为generated model 模拟数据的来源，同时做出判断

k-Nearest Neighbors

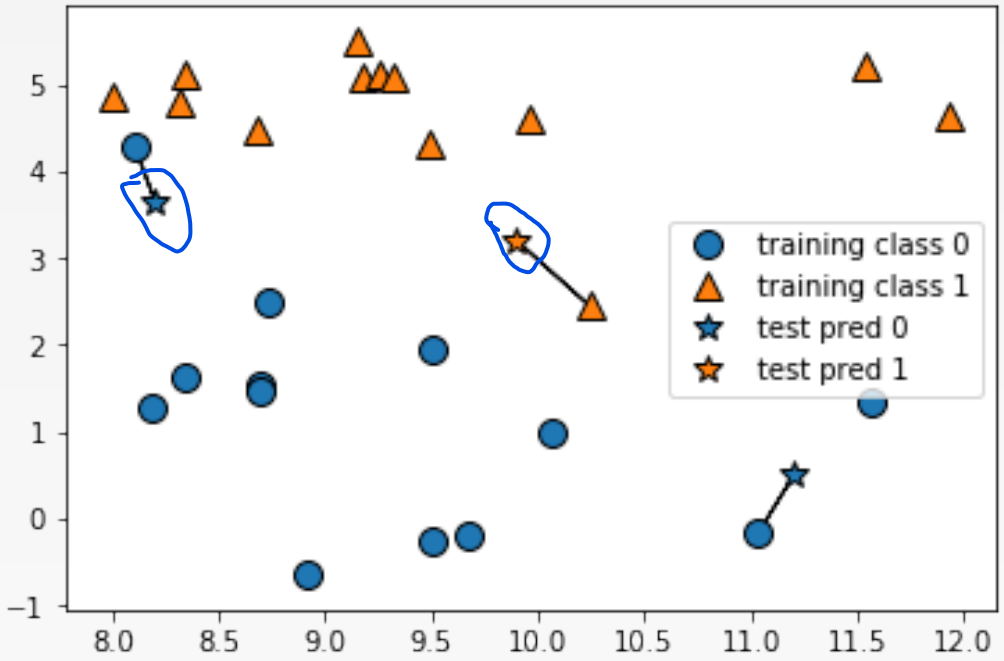
An Illustrative Example

K如何影响算大效果
1.K=N 很稳定，无异质性，所有的预测都相同
2.K=1 永远取最近的点 可能依赖的信息是outlier，原始数据错误导致预测错误，outlier**的影响很大** 受噪音、波动影响大,忽视整**个**parttern

K**太大** 丧失异质性；**太小，受噪音影响大**
underfitting overfitting

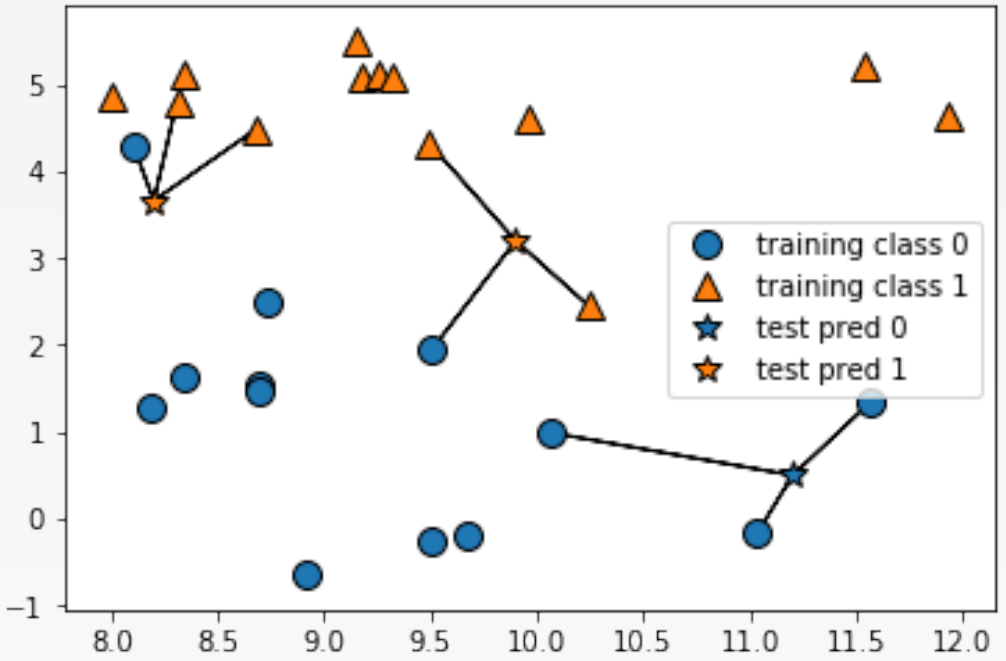
K：在预测时的近邻 neighbor
two features

（X_train,Y_train）-KNN->X_test=>{y^hat}_test



预测两个五角星的颜色，**根据最近的**data point 颜色预测

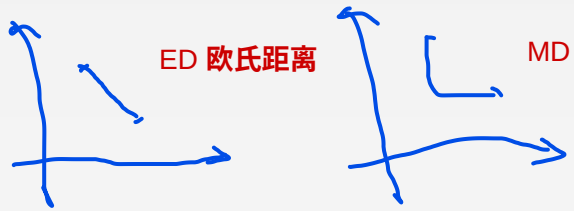
K = 1



综合k=3 **个** data point 的颜色，预测 五角星的颜色

K = 3

KNN: Classification Algorithm



如果数据点有很多features
MD计算量更小
看model中哪个更好

- Training: Store all the examples $(\boldsymbol{X}_{train}, Y_{train})$
- Prediction: $\boldsymbol{X}^{(new)}$
  - Let $\boldsymbol{X}^{(1)}, \ldots, \boldsymbol{X}^{(k)}$ be the k most similar examples to $\boldsymbol{X}^{(new)}$
  - Use certain method to determine $y^{(new)}$ based on $\left(y^{(1)}, \ldots, y^{(k)}\right)$

category binary
如何处理？
jaccard diatance =|A交B|/|A并B|

Keys

1. A distance metric

X、Y 数值范围不同，如何处理？ outlier 不要太影响结果 python boxplot给出数据分布
1.标准化 x-u/sigma
2.归一化 min-max x-min(x)/max-min
对outlier **更友好**
如作图时，要求数据点的范围限制

惯用ED

Euclidean distance $d\left(\boldsymbol{X}^{(j)}, \boldsymbol{X}^{(\mathrm{k})}\right) = \sqrt{\sum_i \left(X_i^{(j)} - X_i^{(k)}\right)^2}$

Manhattan distance $d\left(\boldsymbol{X}^{(j)}, \boldsymbol{X}^{(\mathrm{k})}\right) = \sum_i \left|X_i^{(j)} - X_i^{(k)}\right|$

cosine diatance

夹角sita

2. Value of "K"

如何选到最好的K：Cross validation: larger k? smaller k?

调参：**先**set K 为某个值，再进行其他操作

K**太大** 丧失异质性；**太小，受噪音影响大**
underfitting overfitting

3. Aggregation of the classes of neighbor points

【WiYi ,Wi=1/di

Majority vote

k-NN: Pros and Cons

- **Advantages**:
  - Very simple and intuitive
  - The cost of the learning process is zero
  - No assumption about the characteristics/distributions
  - Works on both classification and regression tasks

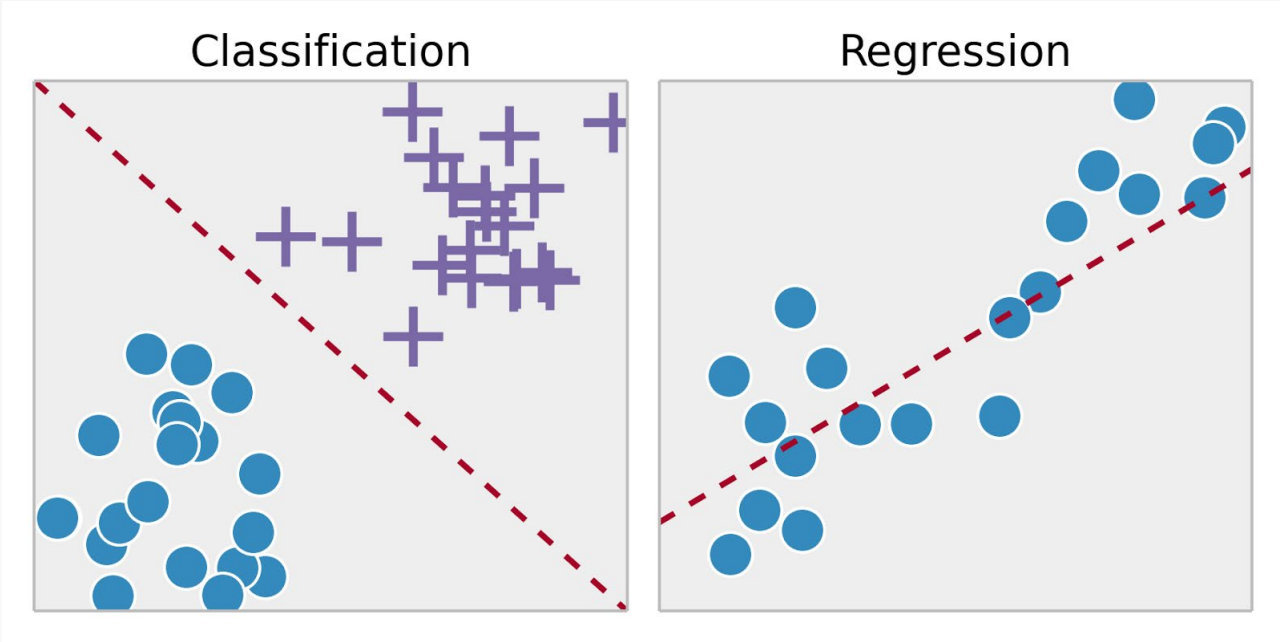
- **Drawbacks**:
  - Computationally expensive when the dataset is very large
    - Need to calculate the compare distance from new example to all other examples
  - Sensitive to outliers

计算成本高，无论是行、列增加，都要重新计算，**不可复用**;
行、列过多都要遍历，难计算

Python Review (k-NN)

Classification vs. Regression

- **Classification**: the goal is to predict a *class label*, which is a choice from a predefined list of possibilities
- **Regression**: the goal is to predict a continuous number, or a *floating-point number* (*real number*) in programming (math) terms



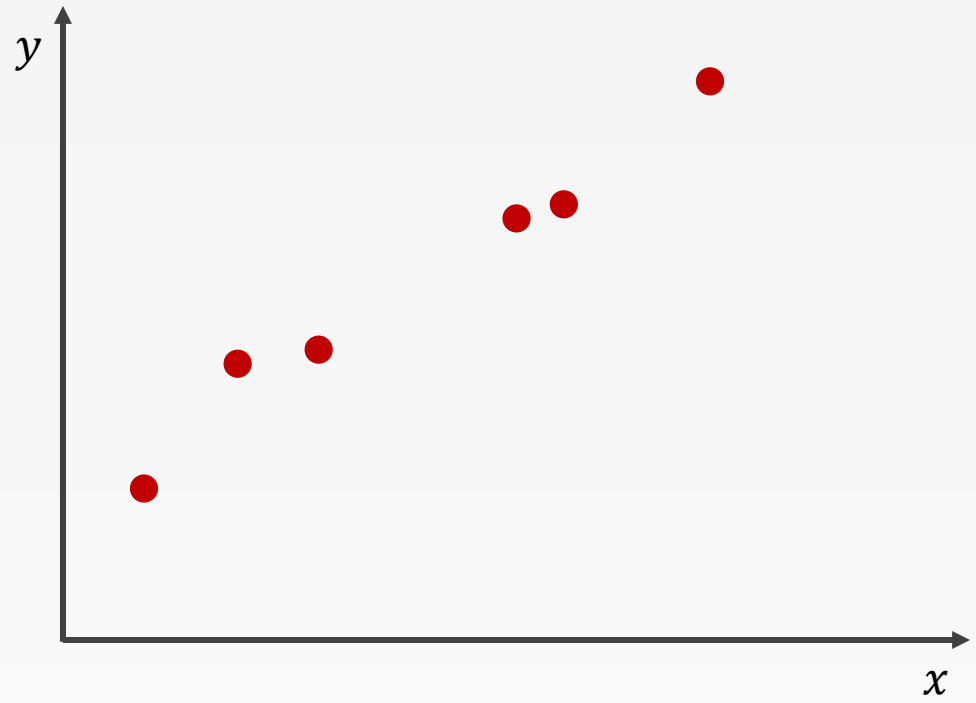
Regression

- Given the value of an input X, the output Y belongs to the set of real value R.
- Evaluation: predict output accurately
- Examples:
  - Predict housing price
  - Forecast precipitation

Reg：
1.KNN 前面是分类问题，但也可以做回归
2.lineal regression ：definition、Optimazation OLS、侧重GD**梯度下降、**regularization (讨论overfitting Ridge、LASSQ)
引申： fearure engine 、polynomial、logistic reg

Regression

Example: Dataset with only one feature x and one scalar output y

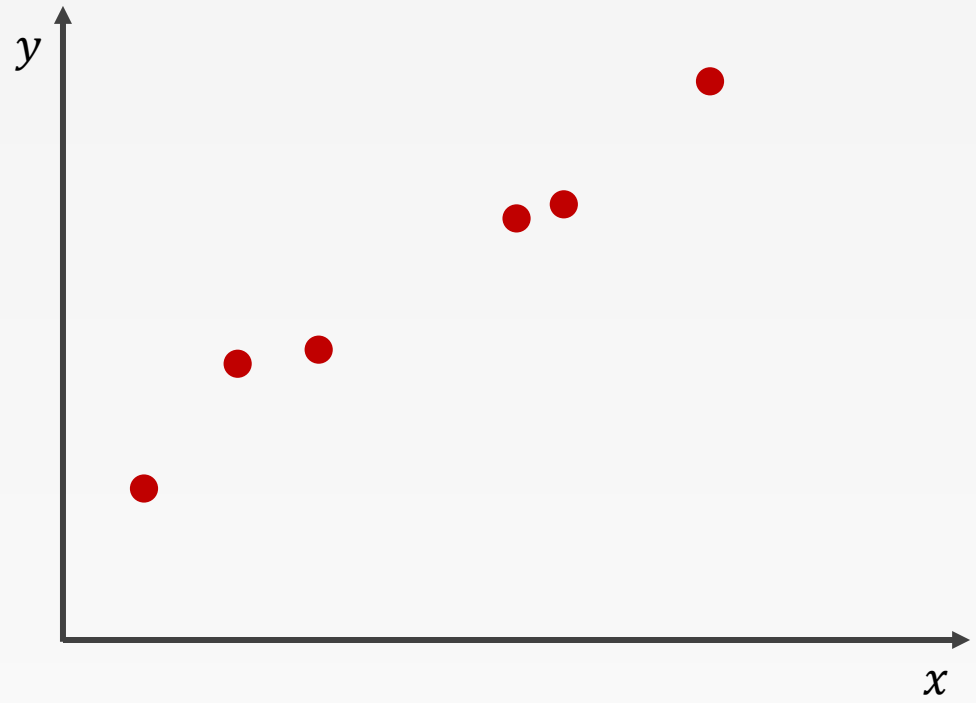


Q: What is the function that best fits these points?

k-NN Regression

KNN regression:
K 个近邻
(y1,y2,……)——》y^hat

Example: Dataset with only one feature x and one scalar output y



$k = 1$

- *Train*: store all (x, y) pairs
- *Predict*: pick the nearest x in the training data and return its y

$k = 2$ Nearest Neighbor Distance Weighted Regression

- *Train*: store all (x, y) pairs
- *Predict*: pick the nearest two instances $x^{(n1)}$ and $x^{(n2)}$ in training data and return the weighted average of their y values

Linear Regression

Linear Regression

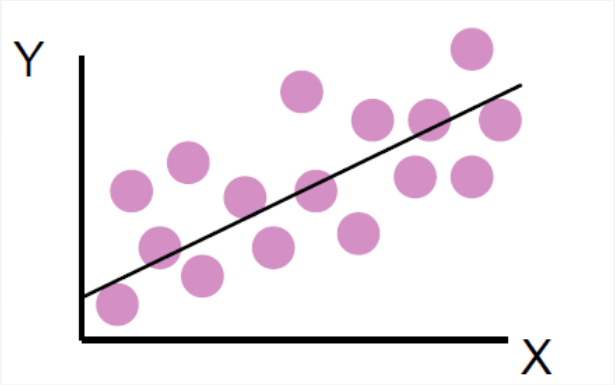
- Linear relationship: outcome (dependent) variable is a linear combination of predictor (independent) variables.

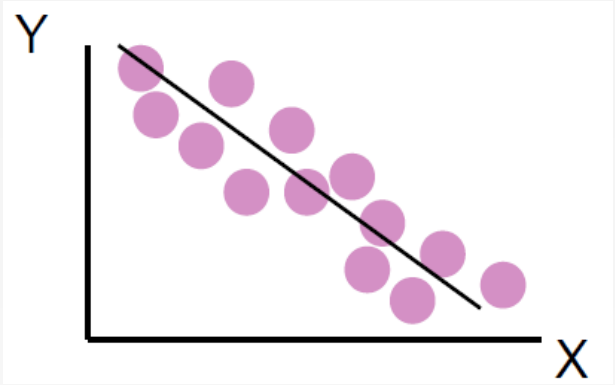
$$Y = b + w_1X_1 + w_2X_2 + \cdots + w_nX_n$$

Outcome | Intercept | Coefficients | Predictors

Linear Regression?

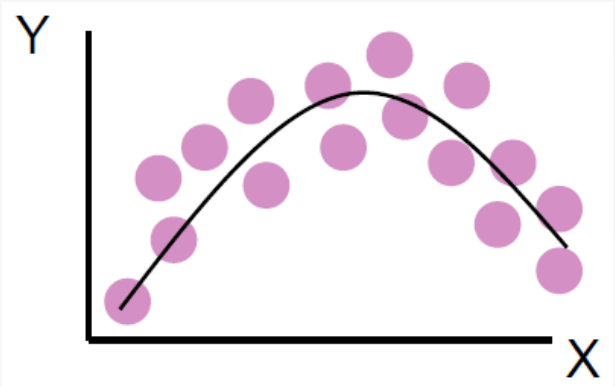
$$Y^{(i)} = b + wX^{(i)}$$

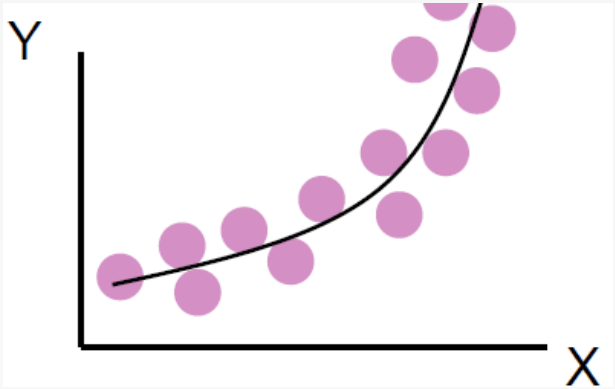




$$Y^{(i)} = b + w_1X^{(i)} + w_2\left(X^{(i)}\right)^2$$

ML中只要求参数线性.





A linear model means linear in parameters (not X)!

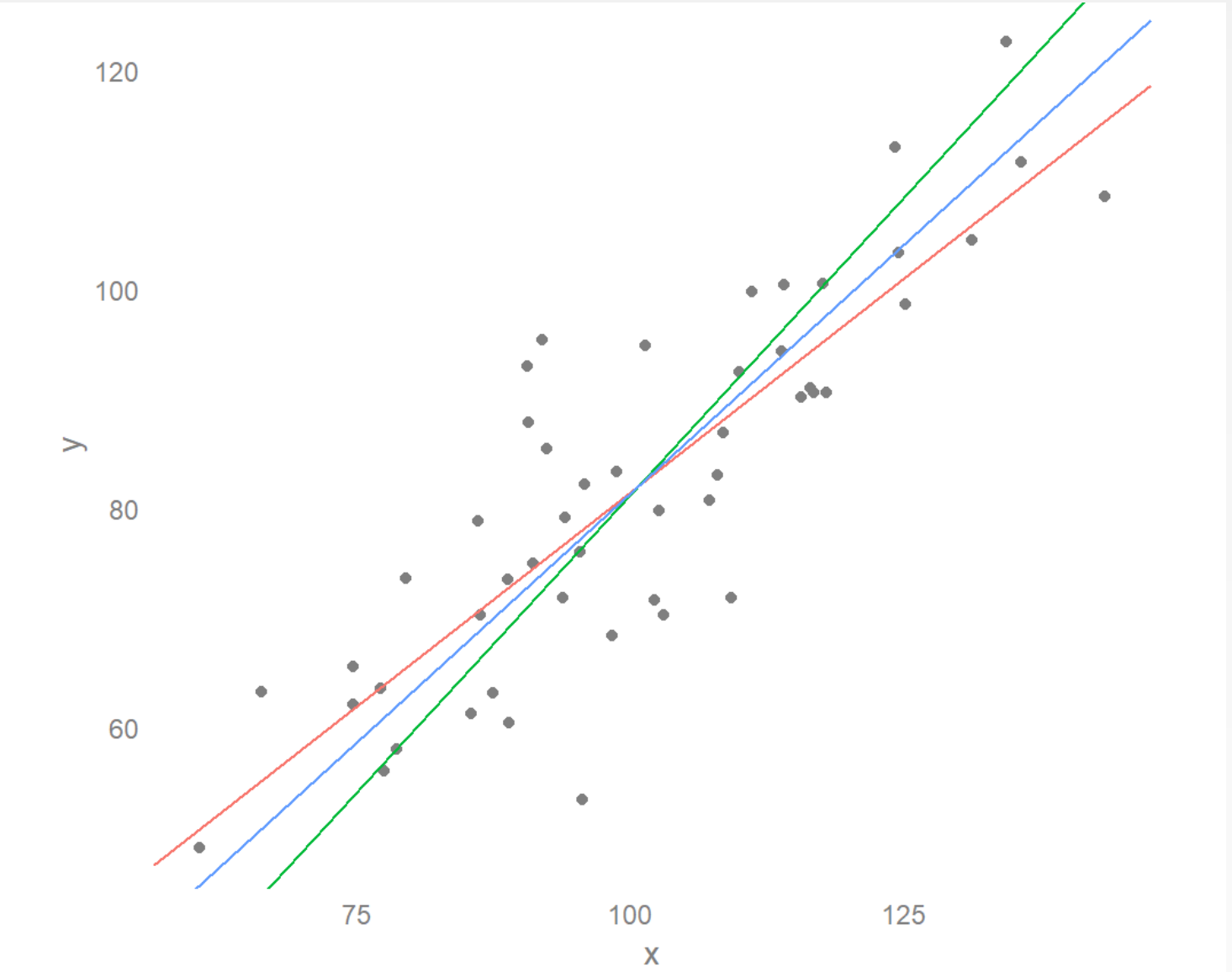
Linear Regression?

- $y = \sum_j \omega_j f_j(x)$ ✓
- $y = \sum_j \omega_j x_j$ ✓
- $y = \sum_j e^{w_j x_j}$ ✗
- $y = \sum_j w_j \sin(j^2 x_j)$ ✓

a linear combination of predictor variable.

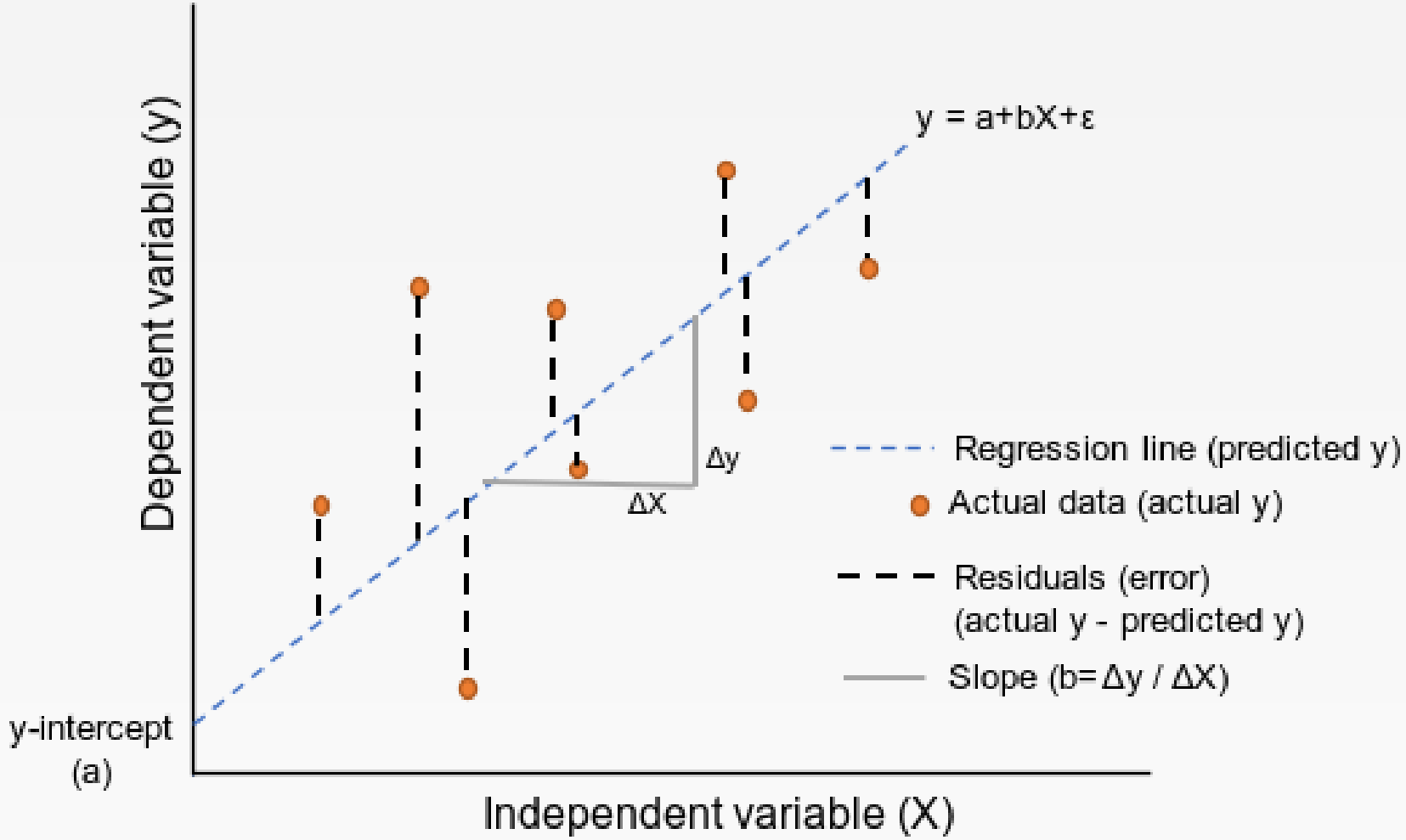
Training?

to find the parameters



Residuals

Residuals e = observed (y) – predicted (y)



Train a Linear Model

- **Goal**: minimize the Error

residual $e = y - \hat{y}$.
observed predicted.

- Potential ways:
  - Sum (mean) of absolute errors $|e_1| + |e_2| + |e_3| + \cdots$
  - Sum (mean) of squared errors $e_1^2 + e_2^2 + e_3^2 + \cdots$

Function Approximation

- **Objective function: mean squared error (MSE)**

$$J(\boldsymbol{\theta}) = \frac{1}{N}\sum_{i=1}^{N} e_i^2$$

$$= \frac{1}{N}\sum_{i=1}^{N}\left(y^{(i)} - \boldsymbol{\theta}^T \boldsymbol{x}^{(i)}\right)^2$$

$$\boldsymbol{x}' = [1, x_1, x_2, \dots, x_M]^T$$

$$\boldsymbol{\theta} = [b, w_1, \dots, w_M]^T$$

- **Solve the unconstrained optimization problem**
  - Closed form OLS
  - Gradient descent
    - Stochastic gradient descent 随机

$$\min_{\theta} J(\theta) \Rightarrow \hat{\theta}$$

$$\widehat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} J(\boldsymbol{\theta})$$

- **Test time**
  - Given a new x, make a prediction $\hat{y} = \widehat{\boldsymbol{\theta}}^T \boldsymbol{x}$

Closed-form Solution (OLS)

Criteria Minimize the sum of squared errors $\min \sum_n \left(y^{(i)} - \hat{y}^{(i)}\right)^2$

Solution $y_i = b + w_1 x_i + \varepsilon_i$

w_1: slope for the estimated regression equation $w_1 = \frac{\sum_n (x^{(i)} - \bar{x})(y^{(i)} - \bar{y})}{\sum_n (x^{(i)} - \bar{x})^2}$

b: intercept for the estimated regression equation $b = \bar{y} - w_1 \bar{x}$

$$\boldsymbol{\beta} = (X^T X)^{-1} X^T y$$

TPE

$D = (\vec{X}^{train}, Y^{train}) = \{(\vec{x}^{(1)}, y), \dots (\vec{x}^{(N)}, y)\}$ → training size

$eg\ (\vec{x}^{(1)}, y^{(1)}) = (x_1^{(n)}, \dots, x_D, y)$

↳ feature space

$y \in R$ or $y \in \{-1, 1\}$

Multi-class: $C^*: \vec{x} \to y$ $y = C^*(x)$

Hypothesis set H. find $h^* \in H$ best approximate C^* ; lose function $\hat{y} = h^*(\vec{x})$ $y = C^*(\vec{x})$

$f(\hat{y} - y)$

Gradient Descent 梯度下降

Linear Regression Solutions

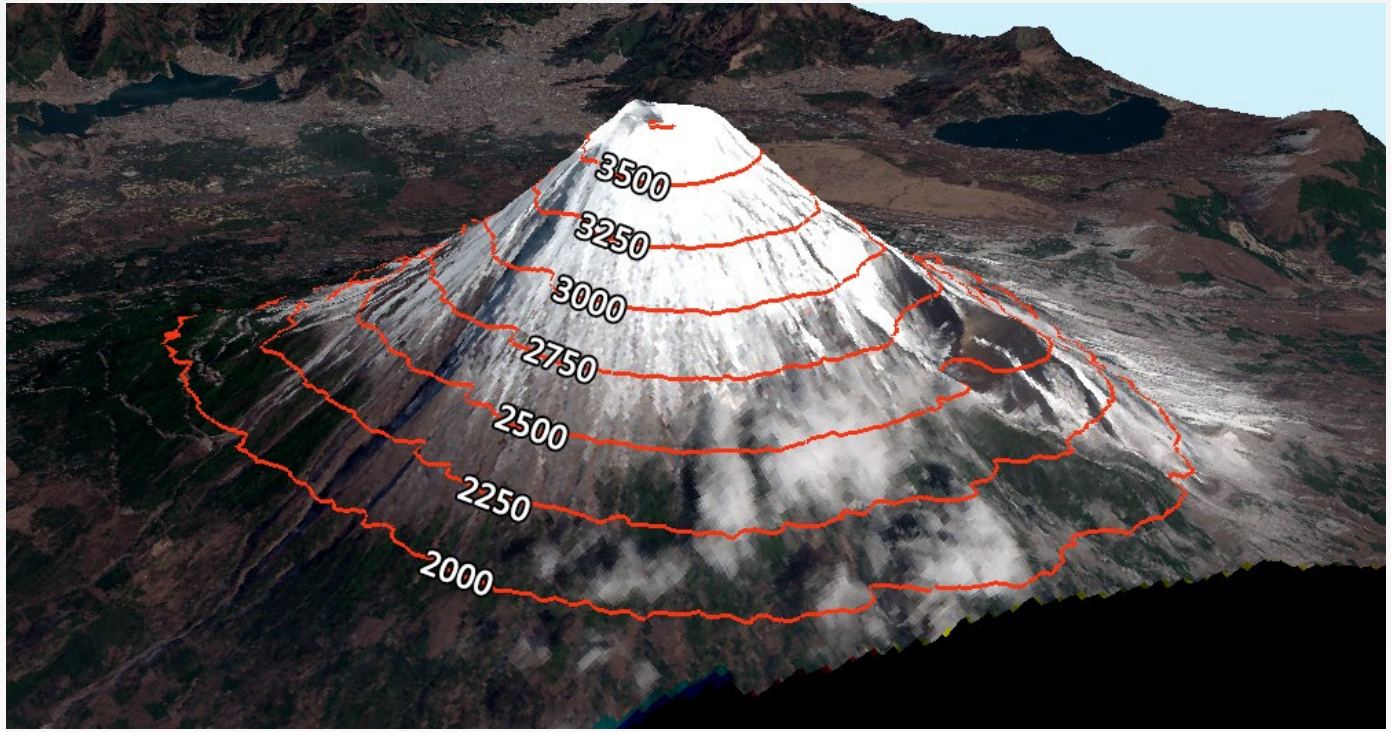
OLS 算得慢，不可逆，不一定有解

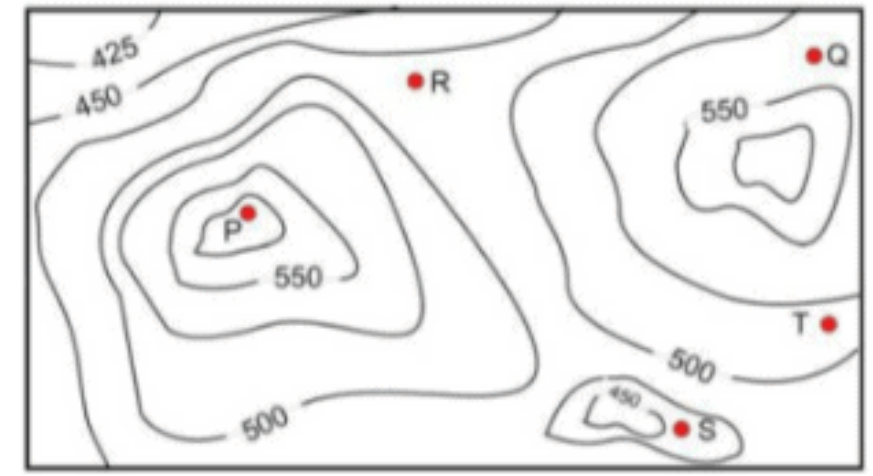
- Closed-form solution:
  - Computational complexity
  - Stability

$$\beta = (X^T X)^{-1} X^T y$$

- Gradient Descent for Linear Regression

Contour Plots





1. Each level curve labeled with value
2. Value label indicates the value of the function for all points lying on that level curve

Optimization by Random Guessing

迭代

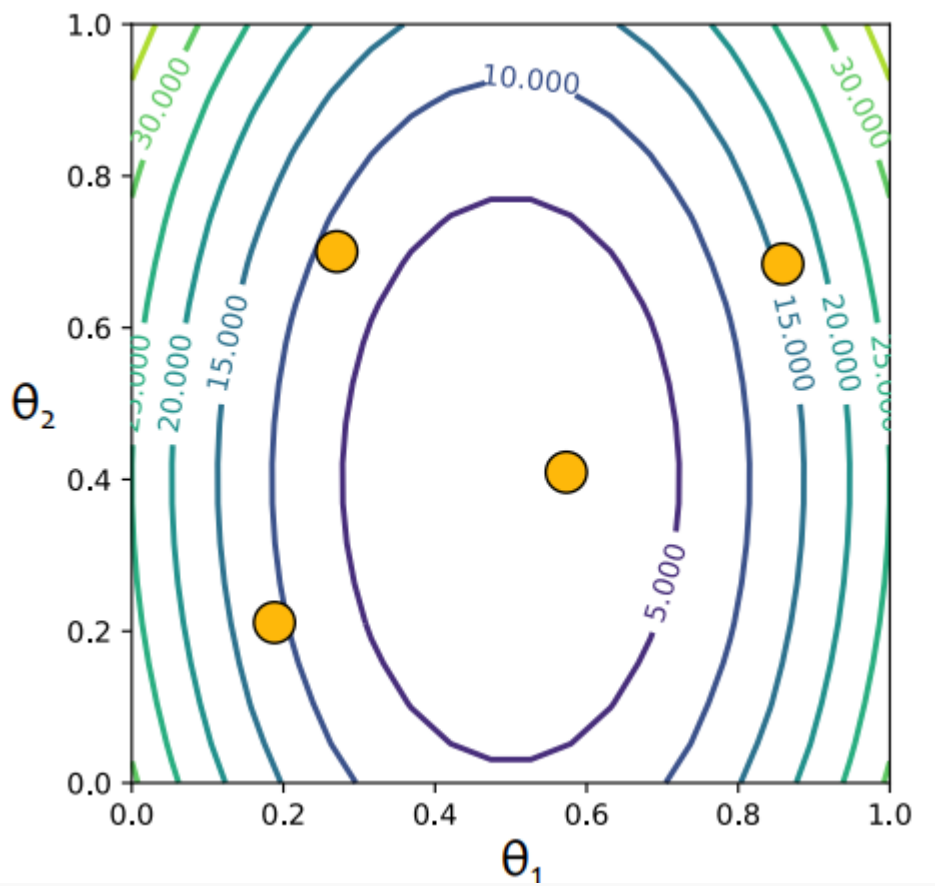
$$J(\boldsymbol{\theta}) = J(\theta_1, \theta_2) = (10(\theta_1 - 0.5))^2 + \left(6(\theta_2 - 0.4)\right)^2$$

Random guessing:

1. Pick a random θ
2. Evaluate $J(\theta)$
3. Repeat steps 1 and 2 many times
4. Return θ that gives smallest $J(\theta)$

Linear Regression:

1. Objective function: MSE
2. contour plot: each line labeled with MSE – lower means a better fit
3. **minimum** corresponds to parameters $(w, b) = (\theta_1, \theta_2)$ that **best fit** some training dataset



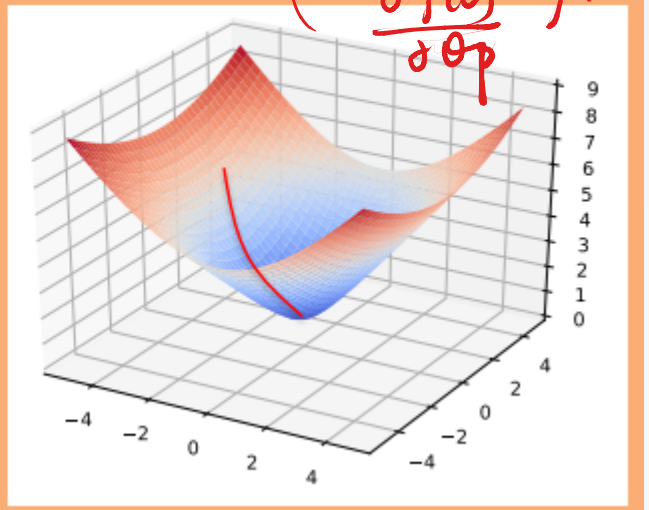

最陡的路，速度最快 => 梯度
$\vec{\theta}^{(j)} \to \vec{\theta}^{(j+1)} \to \cdots \theta^*$

Algorithm

Algorithm 1 Gradient Descent

1: **procedure** GD($\mathcal{D}$, $\boldsymbol{\theta}^{(0)}$)
2: $\quad\boldsymbol{\theta} \leftarrow \boldsymbol{\theta}^{(0)}$
3: $\quad$**while** not converged **do**
4: $\quad\quad\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \gamma \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$
5: $\quad$**return** $\boldsymbol{\theta}$

$\nabla_\theta J(\theta) = \begin{pmatrix} \frac{\partial J(\theta)}{\partial \theta_1} \\ \vdots \\ \frac{\partial J(\theta)}{\partial \theta_p} \end{pmatrix}$



Convergence Criteria (one example): $\|\nabla_\theta J(\boldsymbol{\theta})\|_2 \leq \epsilon$

$|\theta^{(j+1)} - \theta^{(j)}| \leq \varepsilon$; $|J(\theta)$

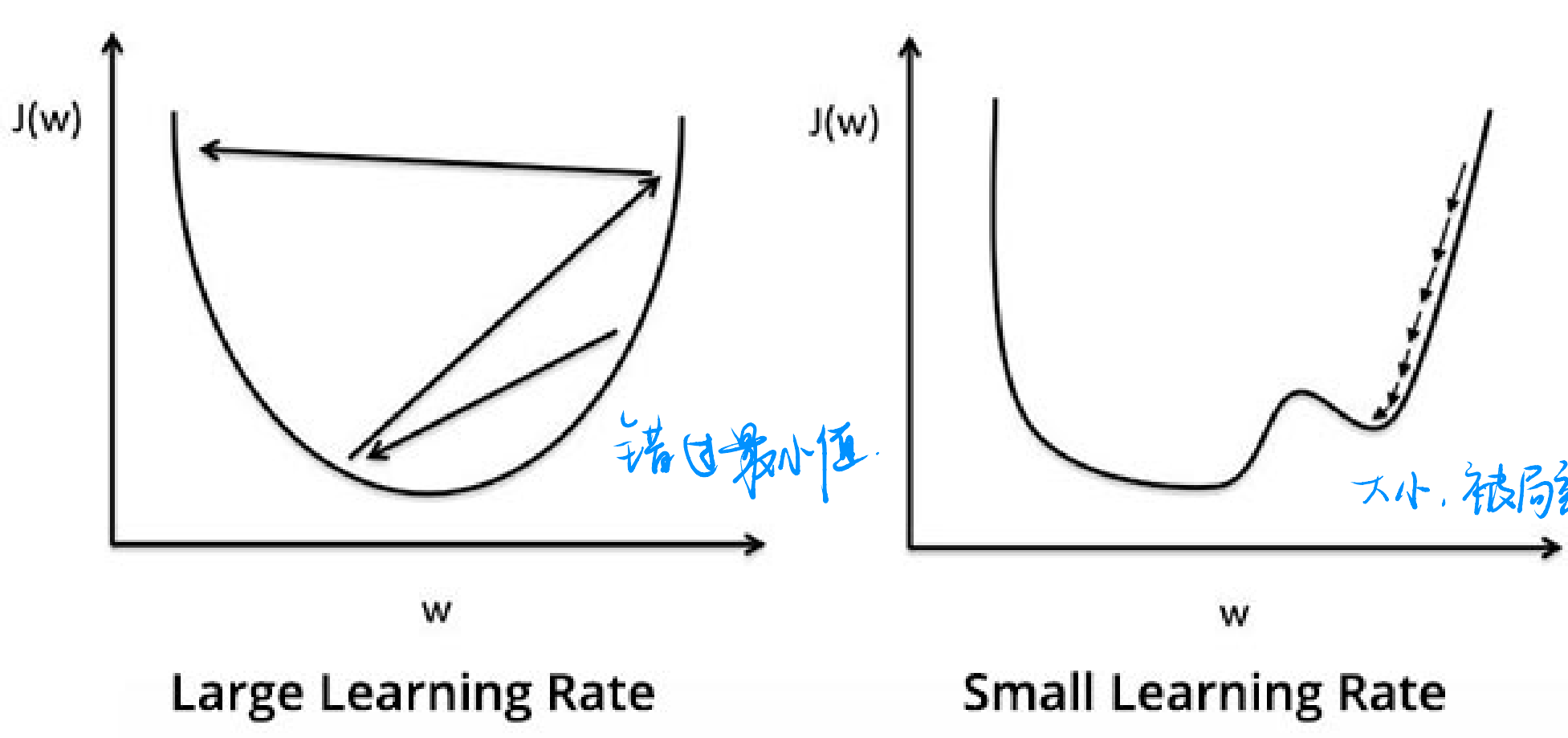
$\min J(\theta)$

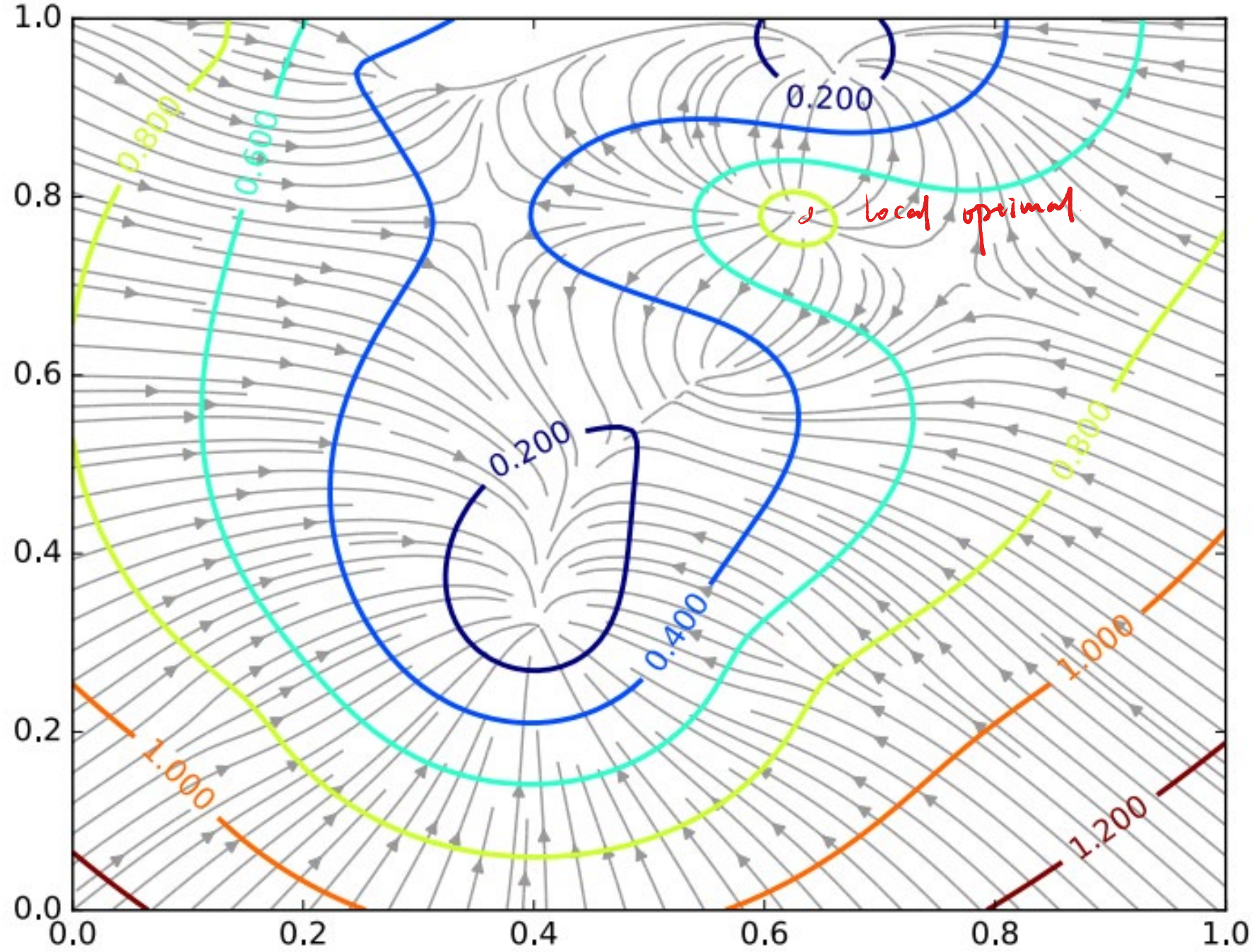
① $\vec{\theta}^{(0)}$

② $\vec{\theta}^{(1)} = \vec{\theta}^{(0)} - \gamma \nabla J(\vec{\theta}^{(0)})$

repeat

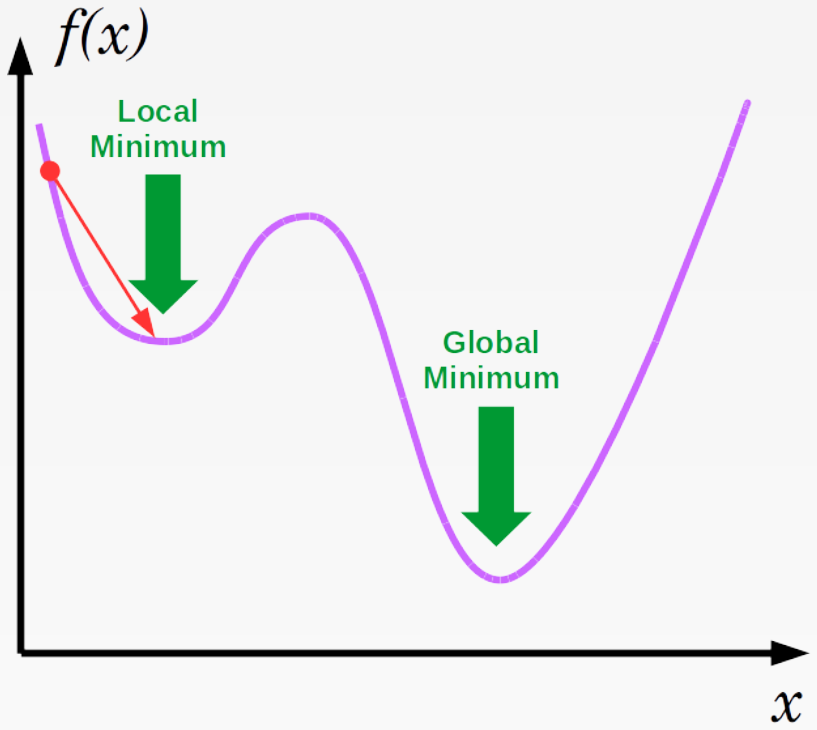
Learning Rate




1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
0.200
0.600
0.800
local optimal
0.200
0.400
0.800
1.000
1.000
1.200

Pros and Cons

- Advantages:
  - Simple and often quite effective on ML tasks
  - Often very scalable
- Drawbacks
  - Might find a local minimum
  - Only applies to smooth function (differentiable)



GD for Linear Regression

Algorithm 1 GD for Linear Regression

1: **procedure** GDLR($\mathcal{D}, \boldsymbol{\theta}^{(0)}$)
2: $\quad\boldsymbol{\theta} \leftarrow \boldsymbol{\theta}^{(0)}$ ▷ Initialize parameters
3: $\quad$**while** not converged **do**
4: $\quad\quad\mathbf{g} \leftarrow \sum_{i=1}^{N}(\boldsymbol{\theta}^T\mathbf{x}^{(i)} - y^{(i)})\mathbf{x}^{(i)}$ ▷ Compute gradient
5: $\quad\quad\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \gamma\mathbf{g}$ ▷ Update parameters
6: $\quad$**return** $\boldsymbol{\theta}$

Stochastic Gradient Descent (SGD)

Algorithm 2 Stochastic Gradient Descent (SGD)

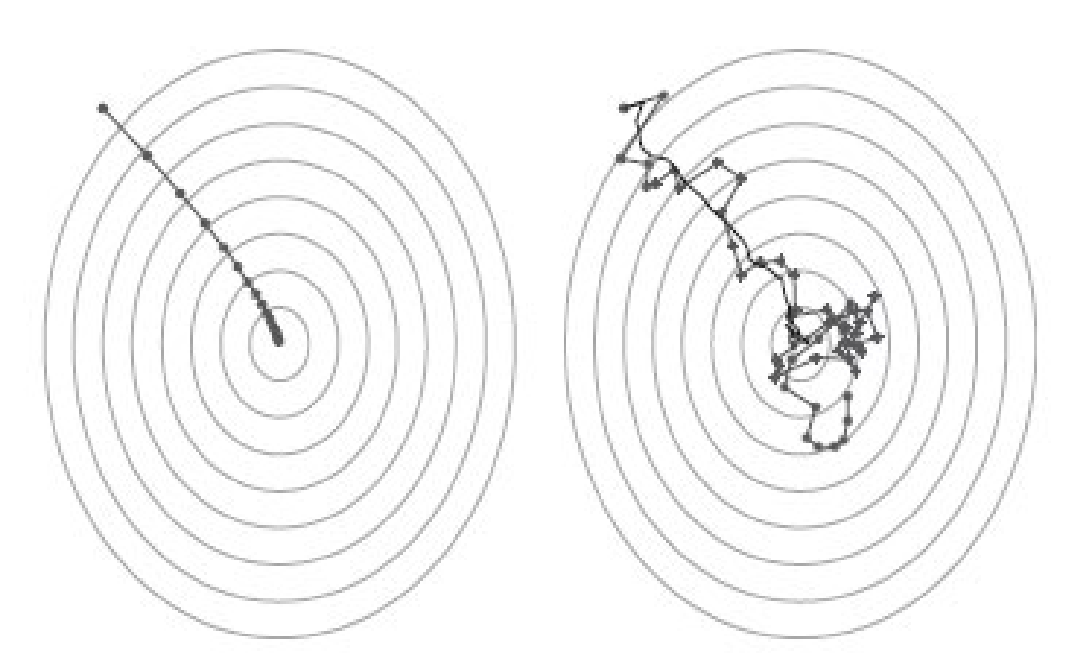
1: **procedure** SGD $(D, \theta^{(0)})$
2: $\quad \theta \leftarrow \theta^{(0)}$
3: $\quad$ **while** not converged **do**
4: $\quad\quad$ **for** $i \sim \text{Uniform}(\{1,2,3,\ldots,N\})$
5: $\quad\quad\quad \theta \leftarrow \theta - \gamma \nabla_\theta J^{(i)}(\theta)$
6: $\quad$ **return** θ

GD $\tilde{\sum}_{i=1}$
mini-batch GD ↕ 随机不放回取k个i
SGD 和一个i计算GD
时间量显著下降，但精度下降

为什么SGD可行？
（期望）

Stochastic Gradient Descent

- Just picks a random instance (or sampling) in the training set to compute the gradient



SGD v

Mini-Batch SGD

- Gradient Descent:
  - Compute true gradient exactly from all N examples
- Stochastic Gradient Descent (SGD):
  - Approximate true gradient by the gradient of one randomly chosen example
- Mini-Batch SGD:
  - Approximate true gradient by the average gradient of K randomly chosen examples