Machine Learning I

3. Linear Regression

WU Xiaokun 吴晓堃

xkun.wu [at] gmail

Contents

线性回归

什么是回归?

回归 regression

- Galton 发现父亲的身高和儿子的身高之间存在着某种给定的关系
  - “子辈的平均身高是其父辈及其所处族群平均身高的加权平均和”。
  - 差异性？从整个人群上来看，父亲和孩子的身高分布是很相近的。

什么是回归?

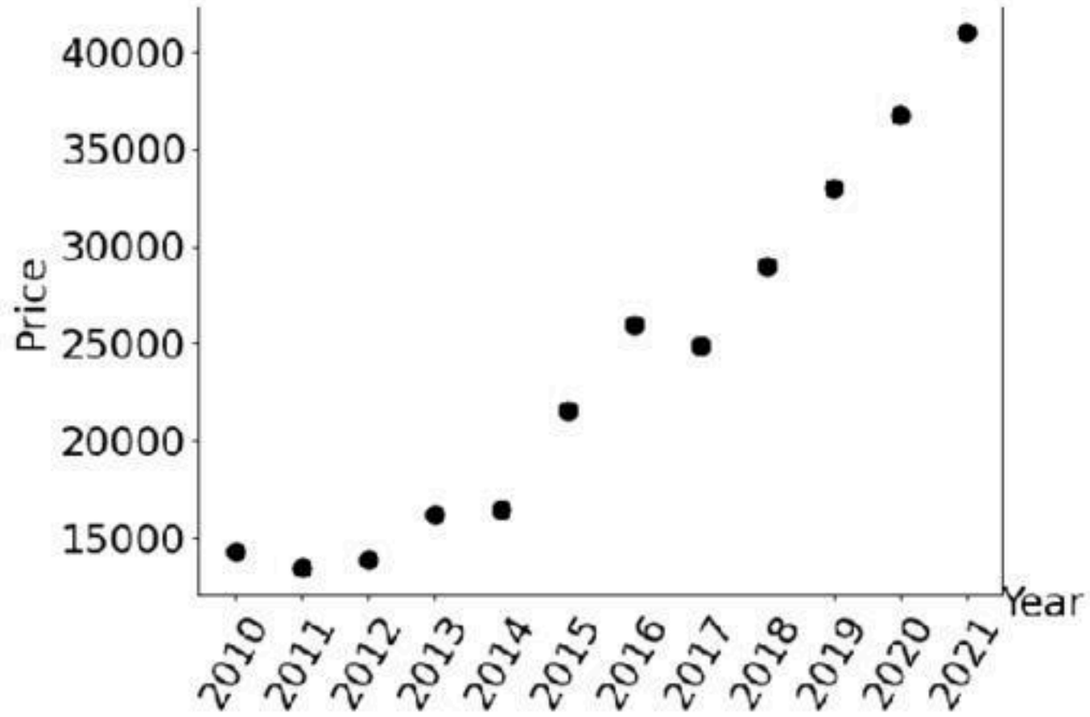
回归 regression

- Galton 发现父亲的身高和儿子的身高之间存在着某种给定的关系
  - “子辈的平均身高是其父辈及其所处族群平均身高的加权平均和”。
  - 差异性？从整个人群上来看，父亲和孩子的身高分布是很相近的。

- 统计数据**驱向均值**的现象
  - re-：重复；gress：行走；-sion：名词化
  - 多次模拟考试成绩的均值可以作为真实水准的依据

为什么讨论回归?

回归分析：对自变量和因变量之间的关系建模



- 用于预测：源于**对趋势（主观判断）的信念**
  - 统计学信条："相关性并不意味着因果关系"
  - 推断：人脑对客观规律的简化描述

线性假设

购房时需要对其价值进行评估

- 找出几个**关键因素**，如面积，地段，存款等
- 做最简单的线性假设：固定比率变化
  - 人脑不善于做非线性计算

线性假设

购房时需要对其价值进行评估

- 找出几个**关键因素**，如面积，地段，存款等
- 做最简单的线性假设：固定比率变化
  - 人脑不善于做非线性计算

线性模型：$y = w_1 x_1 + w_2 x_2 + .. + w_d x_d + b + \epsilon$

- 任务：预测房价
  - **特征**：自变量，相关的一些因素
  - **权重**：决定对应自变量的相对影响力
  - **偏置**：基准值，如当年均值
  - **噪音**ϵ：随机误差；**观测值**y可能偏离真实值

线性回归问题

假如能够确定参数$\mathbf{w}, b$，使**估计值**$\hat{y}$尽可能接近观测值y：

$$\hat{y} = \mathbf{w}^T \mathbf{x} + b \approx \mathbf{w}^T \mathbf{x} + b + \epsilon = y$$

线性回归问题

假如能够确定参数$\mathbf{w}, b$，使**估计值**$\hat{y}$尽可能接近观测值y：

$$\hat{y} = \mathbf{w}^T\mathbf{x} + b \approx \mathbf{w}^T\mathbf{x} + b + \epsilon = y$$

- 问题转化：如何确定参数$\mathbf{w}, b$?

线性回归问题

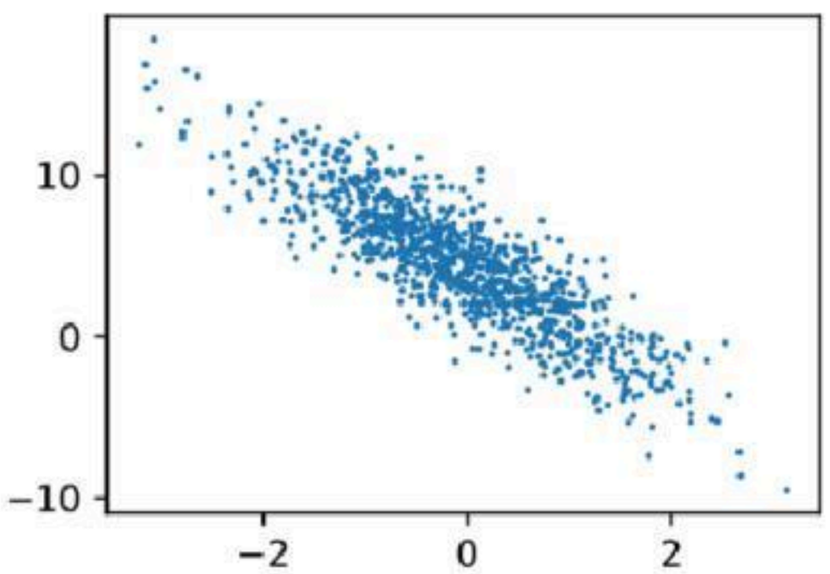
假如能够确定参数$\mathbf{w}, b$，使**估计值**$\hat{y}$尽可能接近观测值y：

$$\hat{y} = \mathbf{w}^T\mathbf{x} + b \approx \mathbf{w}^T\mathbf{x} + b + \epsilon = y$$

- 问题转化：如何确定参数$\mathbf{w}, b$?

基本假设：

- 自变量和因变量**线性相关**：适用线性模型
- 噪音符合**正态分布**：大数定律；适用平方损失



数据驱动

收集样本数据，并发掘数据内在规律

- **样本**：可观测的数据对$(\mathbf{x}, \mathbf{y})$，列向量
  - **样本-特征存储（矩阵）**：每行都是一个展开成特征向量的样本

$$\mathbf{y} = \mathbf{X}\mathbf{w} + b, \mathbf{X} \in \mathbb{R}^{N \times d}$$

- 提示：维数易错！写出每个变量的维数，然后根据形状画成矩形
  - 编程时多打印，或测试驱动

数据驱动

收集样本数据，并发掘数据内在规律

- **样本**：可观测的数据对$(\mathbf{x}, \mathbf{y})$，列向量
  - **样本-特征存储（矩阵）**：每行都是一个展开成特征向量的样本

$$\mathbf{y} = \mathbf{X}\mathbf{w} + b, \mathbf{X} \in \mathbb{R}^{N \times d}$$

- 提示：维数易错！写出每个变量的维数，然后根据形状画成矩形
  - 编程时多打印，或测试驱动

两种求解思路：

- **显式解（最小二乘、正规方程）**：$\bar{\mathbf{w}}^* = (\bar{\mathbf{X}}^T \bar{\mathbf{X}})^{-1} \bar{\mathbf{X}}^T \mathbf{y}$
  - 只适用于有解析解的简单问题
  - 满秩：数据之间不能有相关性
- **数值优化**：例如梯度下降法

最小二乘法

单变量线性回归问题

单变量（一元）是最简单的线性回归问题

$$\hat{\mathbf{y}} = w\mathbf{x} + b$$

- $\mathbf{x}$是列向量：自变量只有一个，但会收集N个样本
- w, b：模型参数

单变量线性回归问题

单变量（一元）是最简单的线性回归问题

$$\hat{\mathbf{y}} = w\mathbf{x} + b$$

- $\mathbf{x}$是列向量：自变量只有一个，但会收集N个样本
- w, b：模型参数

例如：收集3个样本时：

$$\hat{\mathbf{y}} = w \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + b = \begin{bmatrix} wx_1 + b \\ wx_2 + b \\ wx_3 + b \end{bmatrix}$$

最小二乘法：原理

回顾线性回归的目标：使**估计值**$\hat{\mathbf{y}}$尽可能接近观测值$\mathbf{y}$：

$$\hat{\mathbf{y}} = w\mathbf{x} + b \approx w\mathbf{x} + b + \epsilon = \mathbf{y}$$

最小二乘法：原理

回顾线性回归的目标：使**估计值**$\hat{\mathbf{y}}$尽可能接近观测值$\mathbf{y}$：

$$\hat{\mathbf{y}} = w\mathbf{x} + b \approx w\mathbf{x} + b + \epsilon = \mathbf{y}$$

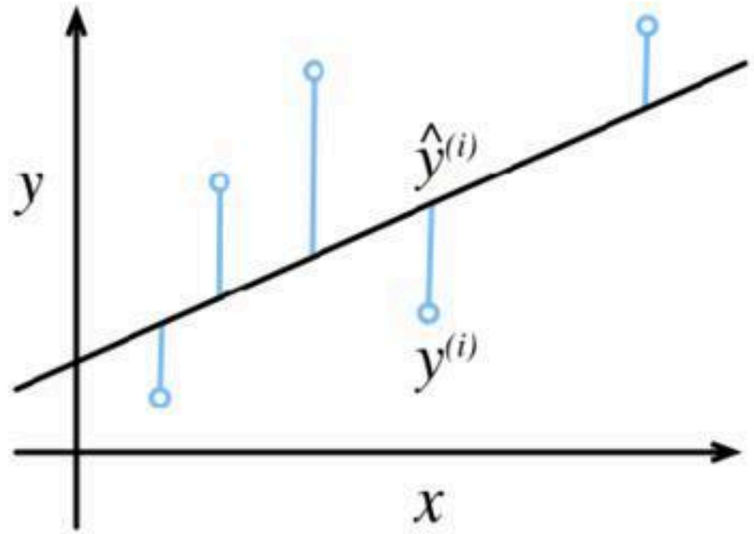
最小二乘法试图使**均方误差(mean squared error, MSE)**最小

$$Loss(y, \hat{y}) = \frac{1}{2N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2 = \frac{1}{2N}\sum_{i=1}^{N}(y_i - w_i x_i - b)^2 \qquad \text{(MSE)}$$

- $Loss(y, \hat{y})$称为**损失函数**

均方误差理解

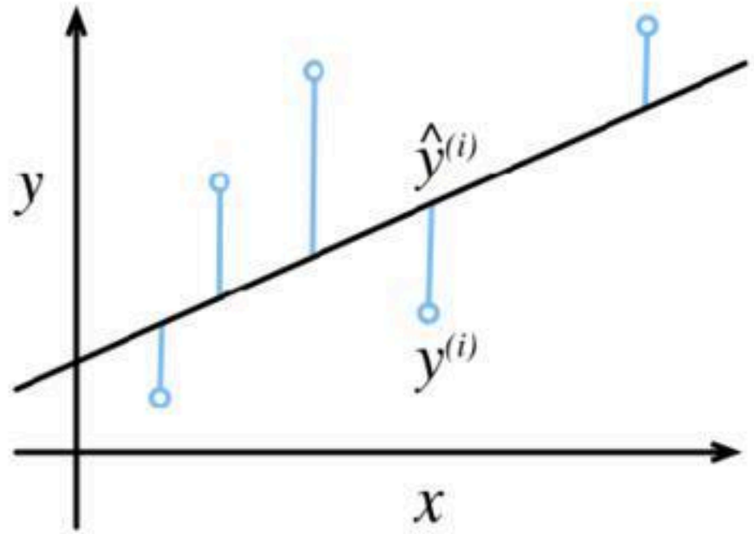
均方误差$\frac{1}{2N}\sum_{i=1}^{N}(y_i - w_i x_i - b)^2$用于度量样本到直线的距离



- 理论上需要计算垂直距离：计算复杂
- 实际工程使用竖直距离

均方误差理解

均方误差$\frac{1}{2N}\sum_{i=1}^{N}(y_i - w_i x_i - b)^2$用于度量样本到直线的距离



- 理论上需要计算垂直距离：计算复杂
- 实际工程使用竖直距离

注意：不可能每个点都最优；“按下葫芦浮起瓢”

- 不断改变直线的斜率和位置，使总体误差最小

(*)单变量线性回归最优解

损失函数$Loss(y,\hat{y})$是**凸函数**：最小值处偏导数为0，且解唯一

- $\frac{\partial Loss}{\partial w}=\frac{\partial(\frac{1}{2m}\sum_{i=1}^{m}(y_i-wx_i-b)^2)}{\partial w}=\frac{1}{m}\sum_{i=1}^{m}(y_i-wx_i-b)(-x_i)=0$
- $\frac{\partial Loss}{\partial b}=\frac{\partial(\frac{1}{2m}\sum_{i=1}^{m}(y_i-wx_i-b)^2)}{\partial b}=\frac{1}{m}\sum_{i=1}^{m}(y_i-wx_i-b)(-1)=0$

(*)单变量线性回归最优解

损失函数$Loss(y,\hat{y})$是**凸函数**：最小值处偏导数为0，且解唯一

- $\frac{\partial Loss}{\partial w}=\frac{\partial(\frac{1}{2m}\sum_{i=1}^{m}(y_i-wx_i-b)^2)}{\partial w}=\frac{1}{m}\sum_{i=1}^{m}(y_i-wx_i-b)(-x_i)=0$
- $\frac{\partial Loss}{\partial b}=\frac{\partial(\frac{1}{2m}\sum_{i=1}^{m}(y_i-wx_i-b)^2)}{\partial b}=\frac{1}{m}\sum_{i=1}^{m}(y_i-wx_i-b)(-1)=0$

省略中间步骤：

- $w=\frac{N(\sum x_jy_j)-(\sum x_j)(\sum y_j)}{N(\sum x_j^2)-(\sum x_j)^2}$
- $b=((\sum y_j)-w(\sum x_j))/N$

单变量线性回归实现

```
def ls_w(xv,yv,n):
    p = n*sum(xv*yv) - sum(xv)*sum(yv)
    q = n*sum(xv*xv) - sum(xv)*sum(xv)
    return p/q

def ls_b(xv,yv,w,n):
    return sum(yv-w*xv)/n
```

- $w = \frac{N(\sum x_j y_j) - (\sum x_j)(\sum y_j)}{N(\sum x_j^2) - (\sum x_j)^2}$
- $b = ((\sum y_j) - w(\sum x_j))/N$

单变量线性回归实现

```
def ls_w(xv,yv,n):
    p = n*sum(xv*yv) - sum(xv)*sum(yv)
    q = n*sum(xv*xv) - sum(xv)*sum(xv)
    return p/q

def ls_b(xv,yv,w,n):
    return sum(yv-w*xv)/n
```

- $w = \frac{N(\sum x_j y_j) - (\sum x_j)(\sum y_j)}{N(\sum x_j^2) - (\sum x_j)^2}$
- $b = ((\sum y_j) - w(\sum x_j))/N$

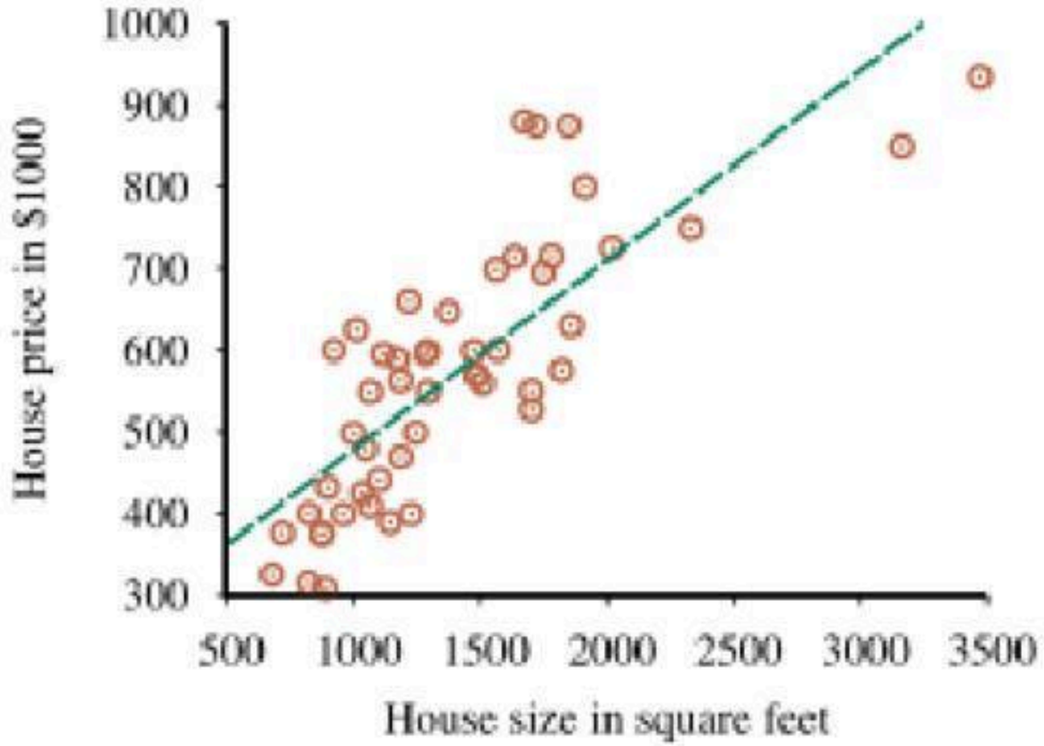
```
xv = np.array([1, 2, 3]).reshape(-1, 1)
yv = np.array([1, 2, 3]).reshape(-1, 1)
w = ls_w(xv,yv,3)
b = ls_b(xv,yv,w,3)
print(f'least square solution: w = {w}, b = {b}')
```

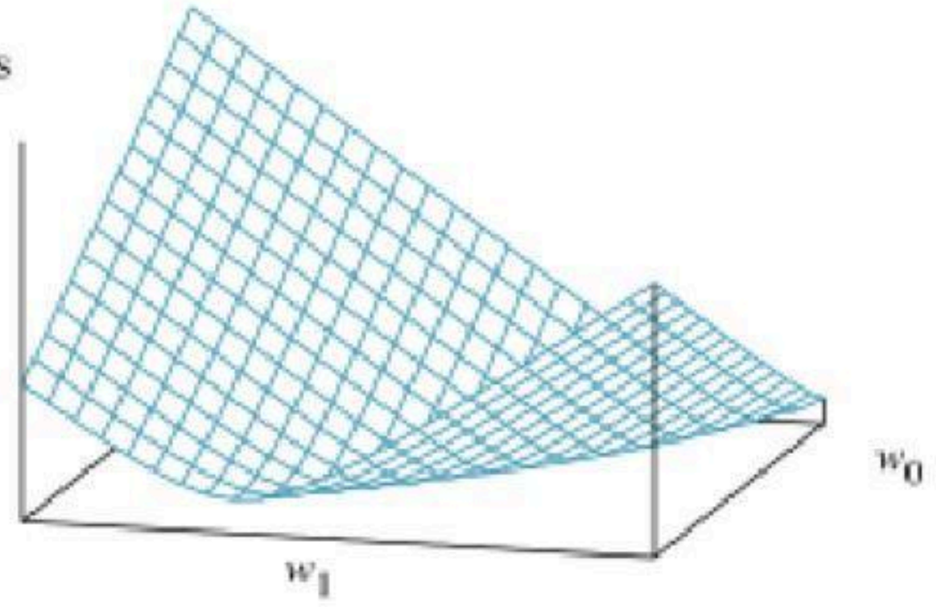
```
least square solution: w = [1.], b = [0.]
```

(*)单变量回归示例

左图为某地房价相对占地面积的数据采样

- 线性假设是合理的：最优解（绿线）从直观上验证

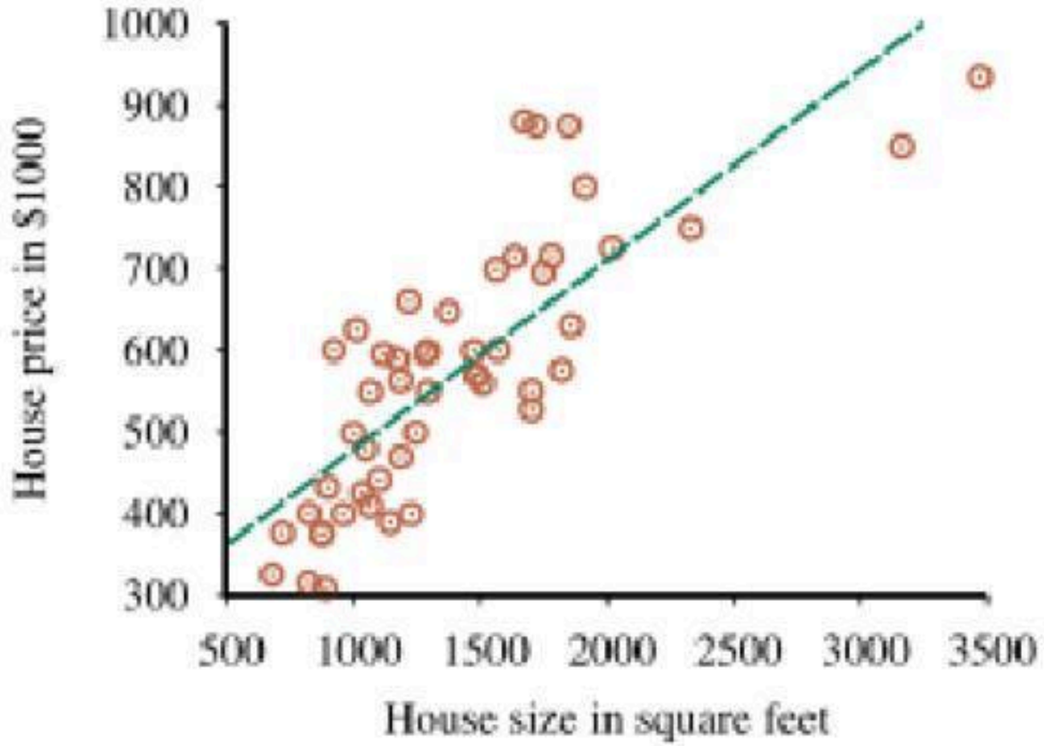


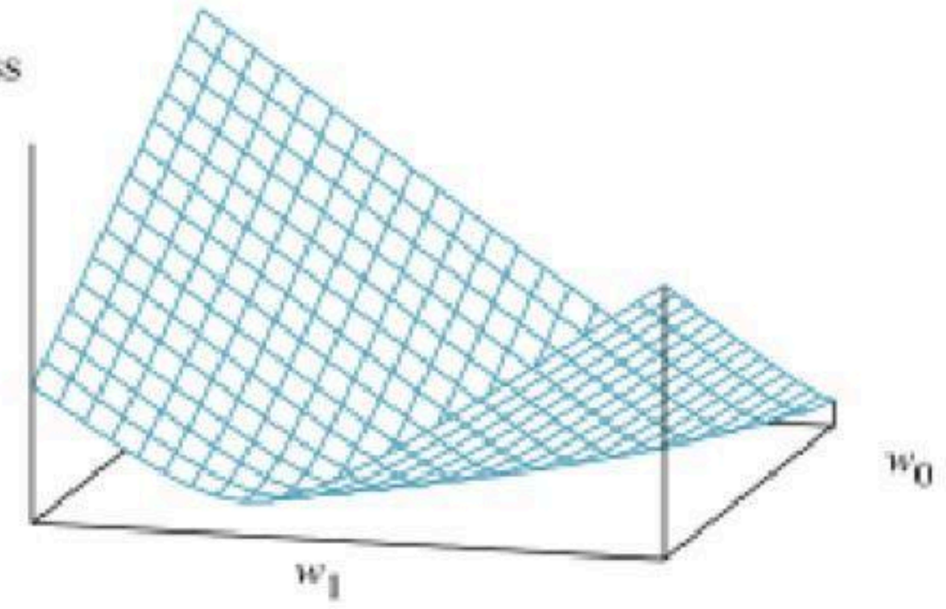


(*)单变量回归示例

左图为某地房价相对占地面积的数据采样

- 线性假设是合理的：最优解（绿线）从直观上验证





- 右图为权重空间：损失函数相对参数是平滑函数
  - 可微，能够计算梯度

多变量线性回归问题

实际问题可能需要多个（特征）指标去评估

- 例如房价依赖于面积、位置、朝向、学区、周边设施、建筑年份等等

多变量线性回归问题

实际问题可能需要多个（特征）指标去评估

- 例如房价依赖于面积、位置、朝向、学区、周边设施、建筑年份等等
- 选取特征的一般准则：
  - 自变量对因变量必须有**显著影响**，并呈密切的**线性相关**
  - 自变量之间应具有一定**互斥性**：自变量之间的相关程度不应高于自变量与因变量之相关性
  - 自变量应具有完整的统计数据，其预测值容易确定

多变量线性回归问题

实际问题可能需要多个（特征）指标去评估

- 例如房价依赖于面积、位置、朝向、学区、周边设施、建筑年份等等
- 选取特征的一般准则：
  - 自变量对因变量必须有**显著影响**，并呈密切的**线性相关**
  - 自变量之间应具有一定**互斥性**：自变量之间的相关程度不应高于自变量与因变量之相关性
  - 自变量应具有完整的统计数据，其预测值容易确定

多元线性回归模型：$y = w_1x_1 + w_2x_2 + .. + w_dx_d + b$

- 例如只考虑面积、位置两个相对重要因素：$y = w_1x_1 + w_2x_2 + b$

多元线性回归：矩阵写法

回顾：**样本**是可观测的数据对$(\mathbf{x}, \mathbf{y})$，列向量

- **样本-特征存储（矩阵）**：每行都是一个展开成特征向量的样本

$$\mathbf{y} = \mathbf{X}\mathbf{w} + b, \mathbf{X} \in \mathbb{R}^{N \times d}$$

多元线性回归：矩阵写法

回顾：**样本**是可观测的数据对$(\mathbf{x}, \mathbf{y})$，列向量

- **样本-特征存储（矩阵）**：每行都是一个展开成特征向量的样本

$$\mathbf{y} = \mathbf{X}\mathbf{w} + b, \mathbf{X} \in \mathbb{R}^{N \times d}$$

例如3个样本，2个特征：

$$\hat{\mathbf{y}} = \begin{bmatrix} x_1^1 & x_1^2 \\ x_2^1 & x_2^2 \\ x_3^1 & x_3^2 \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} + b = \begin{bmatrix} w_1 x_1^1 + w_2 x_1^2 + b \\ w_1 x_2^1 + w_2 x_2^2 + b \\ w_1 x_3^1 + w_2 x_3^2 + b \end{bmatrix}$$

齐次坐标

矩阵计算过程通常将常数项b整合到参数$\mathbf{w}$里

- 同时给输入$\mathbf{X}$添加一个常数维度，称为**齐次坐标**

例如3个样本，2个特征：

$$\hat{\mathbf{y}} = \begin{bmatrix} x_1^1 & x_1^2 & 1 \\ x_2^1 & x_2^2 & 1 \\ x_3^1 & x_3^2 & 1 \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ b \end{bmatrix} = \begin{bmatrix} w_1 x_1^1 + w_2 x_1^2 + b \\ w_1 x_2^1 + w_2 x_2^2 + b \\ w_1 x_3^1 + w_2 x_3^2 + b \end{bmatrix} = \begin{bmatrix} x_1^1 & x_1^2 \\ x_2^1 & x_2^2 \\ x_3^1 & x_3^2 \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} + b$$

齐次坐标

矩阵计算过程通常将常数项b整合到参数$\mathbf{w}$里

- 同时给输入$\mathbf{X}$添加一个常数维度，称为**齐次坐标**

例如3个样本，2个特征：

$$\hat{\mathbf{y}} = \begin{bmatrix} x_1^1 & x_1^2 & 1 \\ x_2^1 & x_2^2 & 1 \\ x_3^1 & x_3^2 & 1 \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ b \end{bmatrix} = \begin{bmatrix} w_1 x_1^1 + w_2 x_1^2 + b \\ w_1 x_2^1 + w_2 x_2^2 + b \\ w_1 x_3^1 + w_2 x_3^2 + b \end{bmatrix} = \begin{bmatrix} x_1^1 & x_1^2 \\ x_2^1 & x_2^2 \\ x_3^1 & x_3^2 \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} + b$$

此时线性回归方程简化成

$$\mathbf{y} = \bar{\mathbf{X}}\bar{\mathbf{w}}, \bar{\mathbf{X}} \in \mathbb{R}^{N \times (d+1)}$$

- 思考：等式两边直接乘以$\bar{\mathbf{X}}$的逆矩阵就可以求出参数值，对吗？
  - 即$\bar{\mathbf{w}} = \bar{\mathbf{X}}^{-1}\mathbf{y}$?

正规方程解法

矩阵可逆的前提条件：非奇异、方阵

- $\bar{\mathbf{X}}$不一定是方阵：首先乘以转置矩阵$\bar{\mathbf{X}}^T$、构造方阵$\bar{\mathbf{X}}^T\bar{\mathbf{X}}$

$$\bar{\mathbf{X}}^T\mathbf{y} = \bar{\mathbf{X}}^T\bar{\mathbf{X}}\bar{\mathbf{w}}$$

正规方程解法

矩阵可逆的前提条件：非奇异、方阵

- $\bar{\mathbf{X}}$不一定是方阵：首先乘以转置矩阵$\bar{\mathbf{X}}^T$、构造方阵$\bar{\mathbf{X}}^T\bar{\mathbf{X}}$

$$\bar{\mathbf{X}}^T\mathbf{y} = \bar{\mathbf{X}}^T\bar{\mathbf{X}}\bar{\mathbf{w}}$$

等式两边直接乘以$(\bar{\mathbf{X}}^T\bar{\mathbf{X}})^{-1}$：$\bar{\mathbf{w}} = (\bar{\mathbf{X}}^T\bar{\mathbf{X}})^{-1}\bar{\mathbf{X}}^T\mathbf{y}$

- 称为$\bar{\mathbf{w}}$的**正规方程 Normal Equations 解**

正规方程解法

矩阵可逆的前提条件：非奇异、方阵

- $\bar{\mathbf{X}}$不一定是方阵：首先乘以转置矩阵$\bar{\mathbf{X}}^T$、构造方阵$\bar{\mathbf{X}}^T\bar{\mathbf{X}}$

$$\bar{\mathbf{X}}^T\mathbf{y} = \bar{\mathbf{X}}^T\bar{\mathbf{X}}\bar{\mathbf{w}}$$

等式两边直接乘以$(\bar{\mathbf{X}}^T\bar{\mathbf{X}})^{-1}$：$\bar{\mathbf{w}} = (\bar{\mathbf{X}}^T\bar{\mathbf{X}})^{-1}\bar{\mathbf{X}}^T\mathbf{y}$

- 称为$\bar{\mathbf{w}}$的**正规方程 Normal Equations** 解

例如4个样本，2个特征：

$$\bar{\mathbf{X}} = \begin{bmatrix} 1 & 8 & 1 \\ 2 & 7 & 1 \\ 3 & 6 & 1 \\ 4 & 5 & 1 \end{bmatrix}, \mathbf{y} = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \end{bmatrix}$$

- 思考：$\bar{\mathbf{X}}^T\bar{\mathbf{X}}$的形状？$(\bar{\mathbf{X}}^T\bar{\mathbf{X}})^{-1}\bar{\mathbf{X}}^T$的形状？

单变量线性回归实现

```
def ne(X,yv):
    num_example = X.shape[0]
    one = np.ones((num_example,1))
    x = np.hstack((X, one))
    xtx = np.asmatrix(np.dot(x.T, x))
    pinv = np.dot(np.linalg.inv(xtx), x.T)
    return np.dot(pinv, yv)
```

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

单变量线性回归实现

```
def ne(X,yv):
    num_example = X.shape[0]
    one = np.ones((num_example,1))
    x = np.hstack((X, one))
    xtx = np.asmatrix(np.dot(x.T, x))
    pinv = np.dot(np.linalg.inv(xtx), x.T)
    return np.dot(pinv, yv)
```

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

```
X = np.array([[1, 8],[2, 7],[3, 6],[4, 5]])
yv = np.array([1,2,3,4]).reshape(-1, 1)
print(f'normal equation solution: w (transposed) = {ne(X,yv).T}')
```

```
normal equation solution: w (transposed) = [[ 1.75   -0.1875  0.
]]
```

预备知识：(*)多元微分

标量导数

一元微分（标量对标量）

- 基本函数：

y	a	x^n	$\exp(x)$	$\log(x)$	$\sin(x)$
$\frac{dy}{dx}$	0	nx^{n-1}	$\exp(x)$	$\frac{1}{x}$	$\cos(x)$

- 复合函数：

y	$u+v$	uv	$y=f(u),u=g(x)$
$\frac{dy}{dx}$	$\frac{du}{dx}+\frac{dv}{dx}$	$\frac{du}{dx}v+\frac{dv}{dx}u$	$\frac{dy}{du}\frac{du}{dx}$

梯度

多元微分（**标量对向量**）

$$df = \sum_i \frac{\partial f}{\partial x_i} dx_i = \frac{\partial f}{\partial x} d\mathbf{x}$$

- 全微分公式；梯度向量与微分向量的内积

梯度

多元微分（**标量对向量**）

$$df = \sum_i \frac{\partial f}{\partial x_i} dx_i = \frac{\partial f}{\partial x} d\mathbf{x}$$

- 全微分公式；梯度向量与微分向量的内积

- 微分：$d\mathbf{x} = [dx_1, dx_2, .., dx_n]^T$
- （偏）导数：$\frac{\partial}{\partial \mathbf{x}} = [\frac{\partial}{\partial x_1}, \frac{\partial}{\partial x_2}, .., \frac{\partial}{\partial x_n}]$
  - “梯度横着走”

雅可比矩阵 Jacobian matrix

多元微分（向量对向量）

$$\mathbf{x}=\begin{bmatrix}x_1\\x_2\\\vdots\\x_n\end{bmatrix},\mathbf{y}=\begin{bmatrix}y_1\\y_2\\\vdots\\y_m\end{bmatrix}$$

$$\frac{\partial \mathbf{y}}{\partial \mathbf{x}}=\begin{bmatrix}\frac{\partial y_1}{\partial \mathbf{x}}\\\frac{\partial y_2}{\partial \mathbf{x}}\\\vdots\\\frac{\partial y_m}{\partial \mathbf{x}}\end{bmatrix}=\begin{bmatrix}\frac{\partial y_1}{\partial x_1},\frac{\partial y_1}{\partial x_2},\cdots,\frac{\partial y_1}{\partial x_n}\\\frac{\partial y_2}{\partial x_1},\frac{\partial y_2}{\partial x_2},\cdots,\frac{\partial y_2}{\partial x_n}\\\vdots,\vdots,\ddots,\vdots\\\frac{\partial y_m}{\partial x_1},\frac{\partial y_m}{\partial x_2},\cdots,\frac{\partial y_m}{\partial x_n}\end{bmatrix}$$

特例

多元微分（**向量对标量**）

$$\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix}, \frac{\partial \mathbf{y}}{\partial x} = \begin{bmatrix} \frac{\partial y_1}{\partial x} \\ \frac{\partial y_2}{\partial x} \\ \vdots \\ \frac{\partial y_m}{\partial x} \end{bmatrix}$$

- $\frac{\partial y}{\partial \mathbf{x}}$是行向量，$\frac{\partial \mathbf{y}}{\partial x}$是列向量
  - 只是其中一种约定方式

规律

	$x:(1,)$	$\mathbf{x}:(n,1)$
$y:(1,)$	$\frac{\partial y}{\partial x}:(1,)$	$\frac{\partial y}{\partial \mathbf{x}}:(1,n)$
$\mathbf{y}:(m,1)$	$\frac{\partial \mathbf{y}}{\partial x}:(m,1)$	$\frac{\partial \mathbf{y}}{\partial \mathbf{x}}:(m,n)$

- x逆序排
- 先排y，再排x

规律

	$x:(1,)$	$\mathbf{x}:(n,1)$
$y:(1,)$	$\frac{\partial y}{\partial x}:(1,)$	$\frac{\partial y}{\partial \mathbf{x}}:(1,n)$
$\mathbf{y}:(m,1)$	$\frac{\partial \mathbf{y}}{\partial x}:(m,1)$	$\frac{\partial \mathbf{y}}{\partial \mathbf{x}}:(m,n)$

- x逆序排
- 先排y，再排x

聪明的数学家非常善于提取、总结客观规律

- 微分学被系统地搭建起来
- 高度抽象的概念需要大量的**标记符**来概括描述
  - 于是本来没有的**问题被创造出来**
- 更聪明的物理学家想出了简化标记

拓展到矩阵

	$x:(1,)$	$\mathbf{x}:(n,1)$	$\mathbf{X}:(n,k)$
$y:(1,)$	$\frac{\partial y}{\partial x}:(1,)$	$\frac{\partial y}{\partial \mathbf{x}}:(1,n)$	$\frac{\partial y}{\partial \mathbf{X}}:(k,n)$
$\mathbf{y}:(m,1)$	$\frac{\partial \mathbf{y}}{\partial x}:(m,1)$	$\frac{\partial \mathbf{y}}{\partial \mathbf{x}}:(m,n)$	$\frac{\partial \mathbf{y}}{\partial \mathbf{X}}:(m,k,n)$
$\mathbf{Y}:(m,l)$	$\frac{\partial \mathbf{Y}}{\partial x}:(m,l)$	$\frac{\partial \mathbf{Y}}{\partial \mathbf{x}}:(m,l,n)$	$\frac{\partial \mathbf{Y}}{\partial \mathbf{X}}:(m,l,k,n)$

拓展到矩阵

	$x:(1,)$	$\mathbf{x}:(n,1)$	$\mathbf{X}:(n,k)$
$y:(1,)$	$\frac{\partial y}{\partial x}:(1,)$	$\frac{\partial y}{\partial \mathbf{x}}:(1,n)$	$\frac{\partial y}{\partial \mathbf{X}}:(k,n)$
$\mathbf{y}:(m,1)$	$\frac{\partial \mathbf{y}}{\partial x}:(m,1)$	$\frac{\partial \mathbf{y}}{\partial \mathbf{x}}:(m,n)$	$\frac{\partial \mathbf{y}}{\partial \mathbf{X}}:(m,k,n)$
$\mathbf{Y}:(m,l)$	$\frac{\partial \mathbf{Y}}{\partial x}:(m,l)$	$\frac{\partial \mathbf{Y}}{\partial \mathbf{x}}:(m,l,n)$	$\frac{\partial \mathbf{Y}}{\partial \mathbf{X}}:(m,l,k,n)$

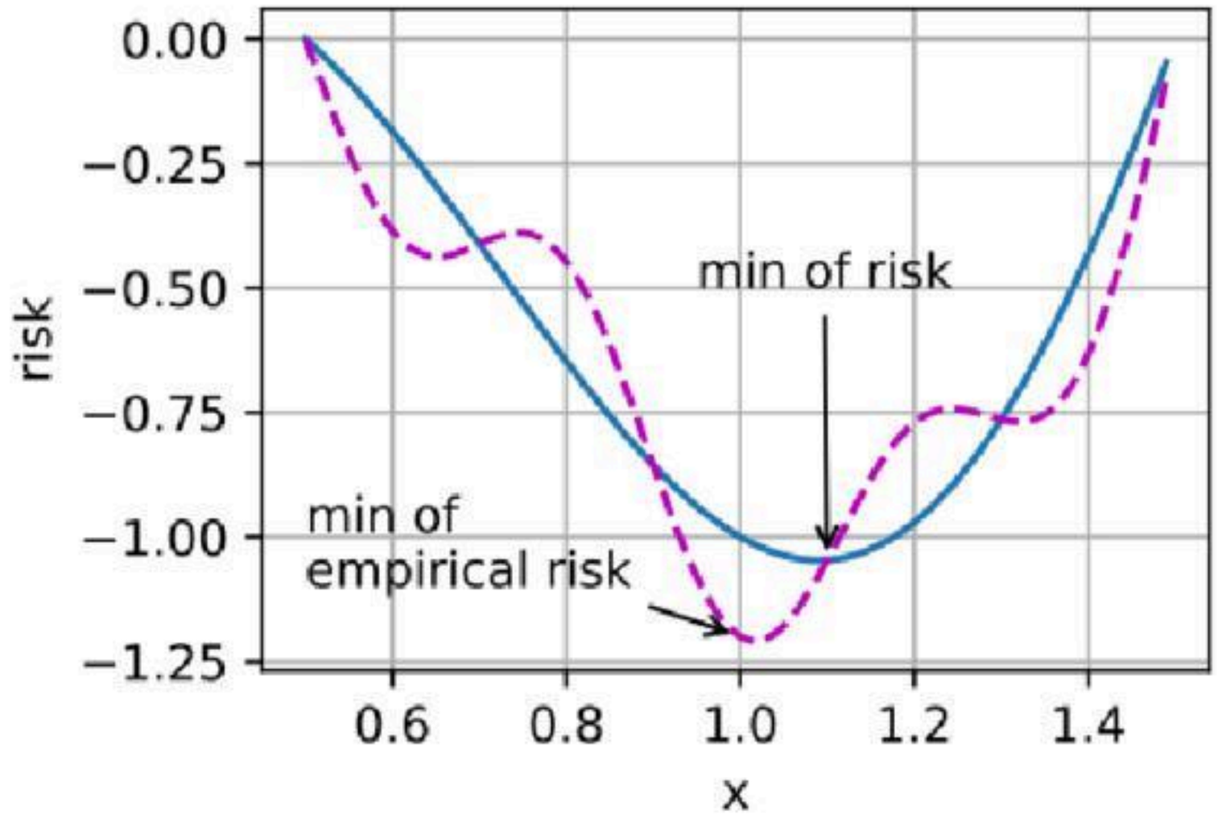
思考：矩阵是二维张量，拓展到N维张量?

实验：微分计算

预备知识：优化

(*)优化、泛化

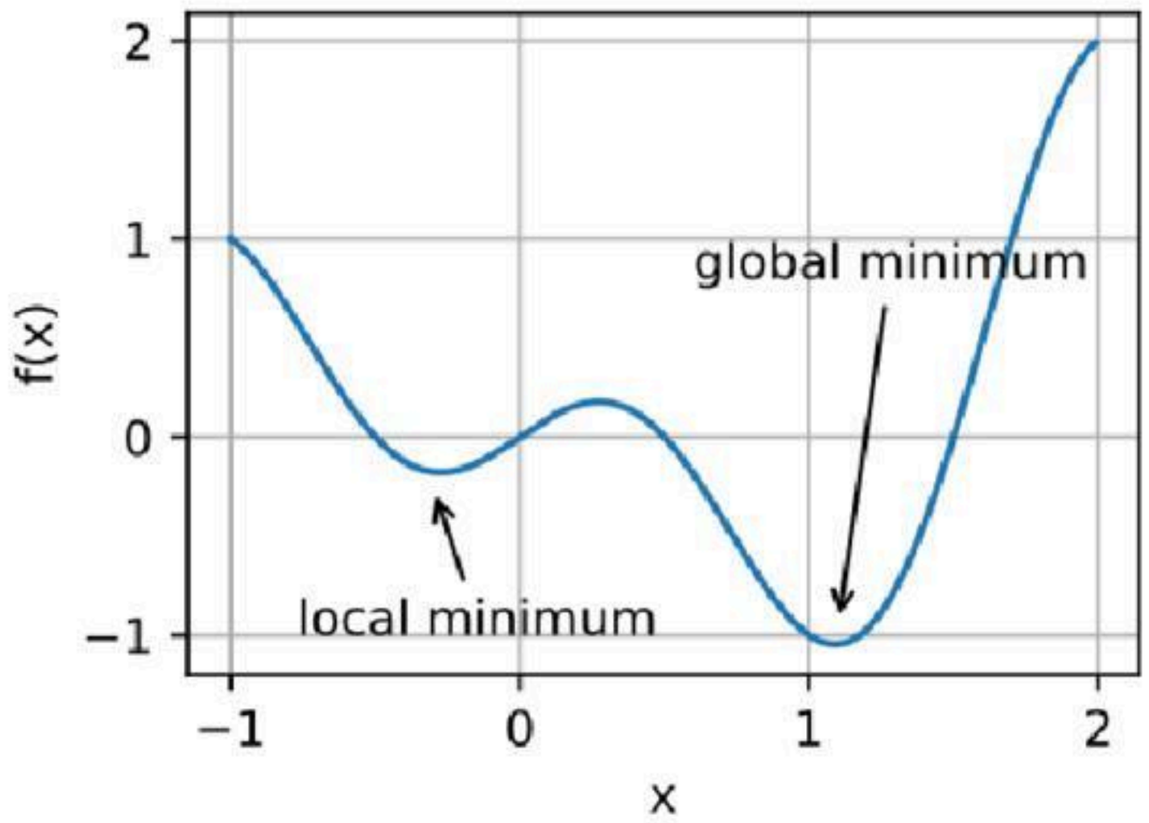
经验风险是训练数据集的平均损失；而**风险**则是整个数据群的预期损失。



局部最小、全局最小

深度学习模型的目标函数通常有许多**局部最优解**。

$$f(x) = x \cdot \cos(\pi x), -1.0 \leq x \leq 2.0$$



梯度下降

Taylor展开：函数的**一阶近似**

$$f(x+\epsilon) = f(x) + \epsilon f'(x) + O(\epsilon^2)$$

梯度下降

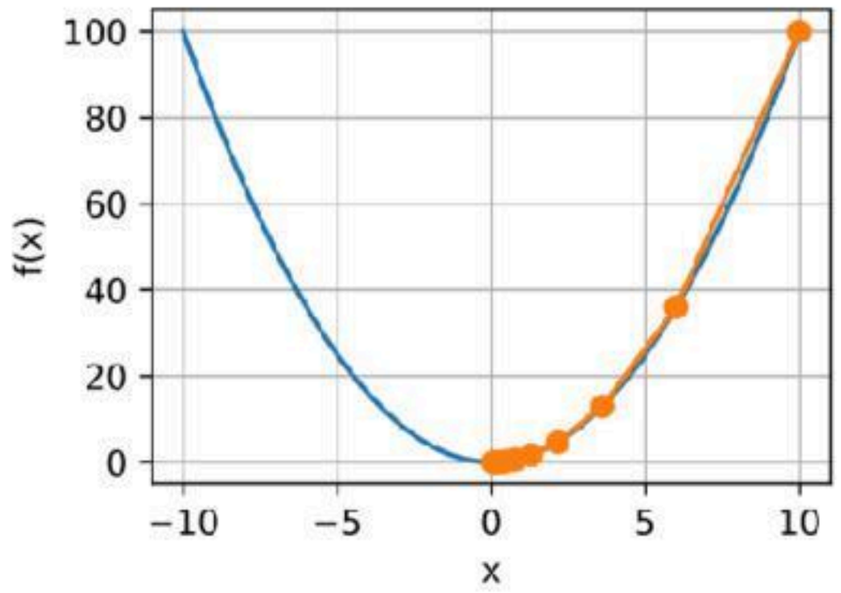
Taylor展开：函数的**一阶近似**

$$f(x+\epsilon) = f(x) + \epsilon f'(x) + O(\epsilon^2)$$

梯度：函数值**增长最快**的方向

- 向梯度的**反方向**走一小步

- $f(x - f'(x)\eta) \lessapprox f(x)$
- $x \leftarrow x - f'(x)\eta$



梯度下降

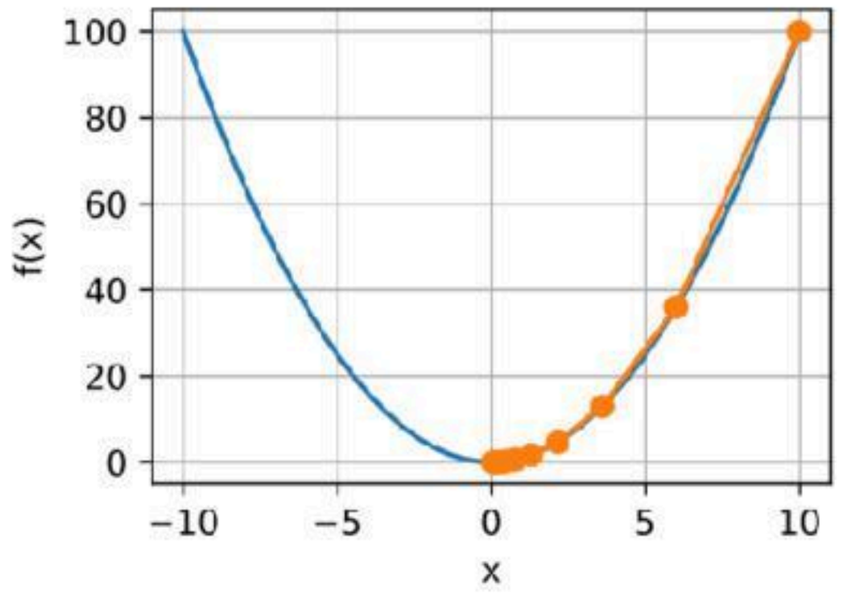
Taylor展开：函数的**一阶近似**

$$f(x+\epsilon) = f(x) + \epsilon f'(x) + O(\epsilon^2)$$

梯度：函数值**增长最快**的方向

- 向梯度的**反方向**走一小步

- $f(x - f'(x)\eta) \lessapprox f(x)$
- $x \leftarrow x - f'(x)\eta$

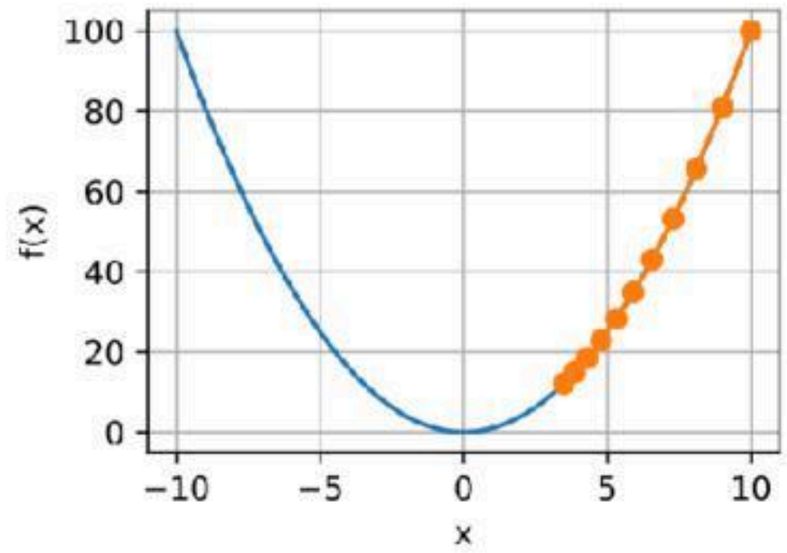


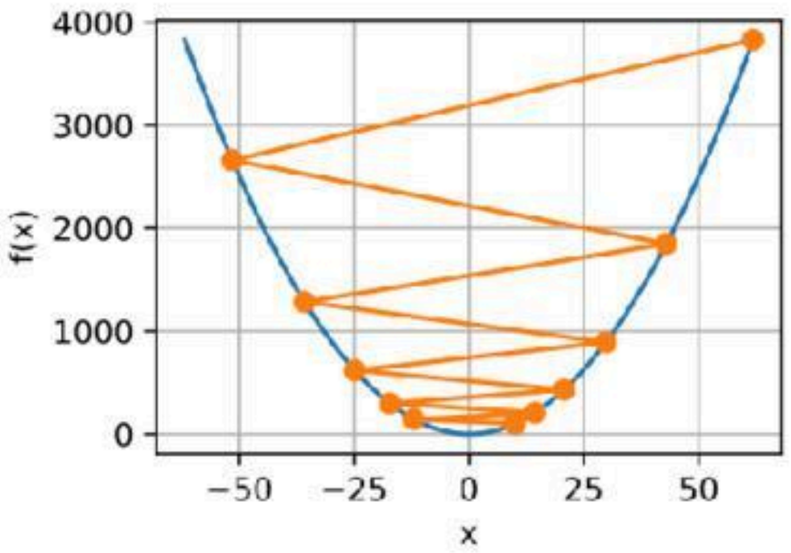
牛顿法：**二阶展开**，需要能够计算二阶导数（Hessian）

学习率

学习率 learning rate： η

- 太小：收敛慢
- 太大：发散

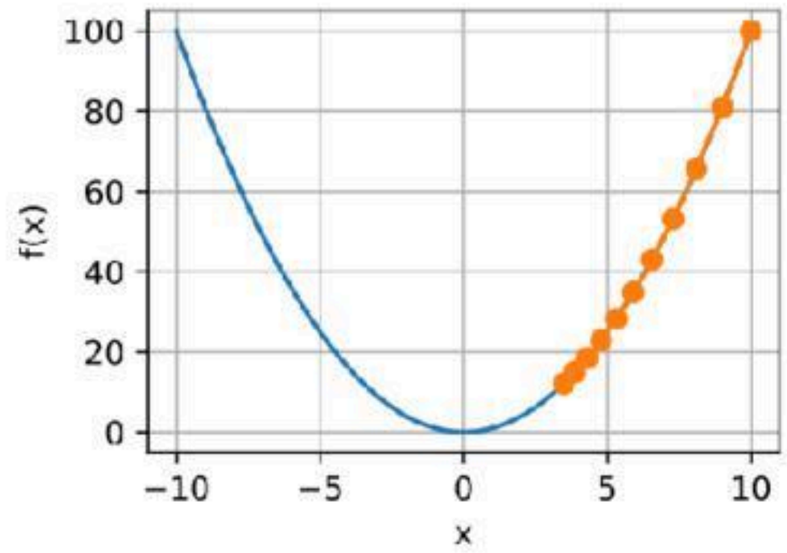


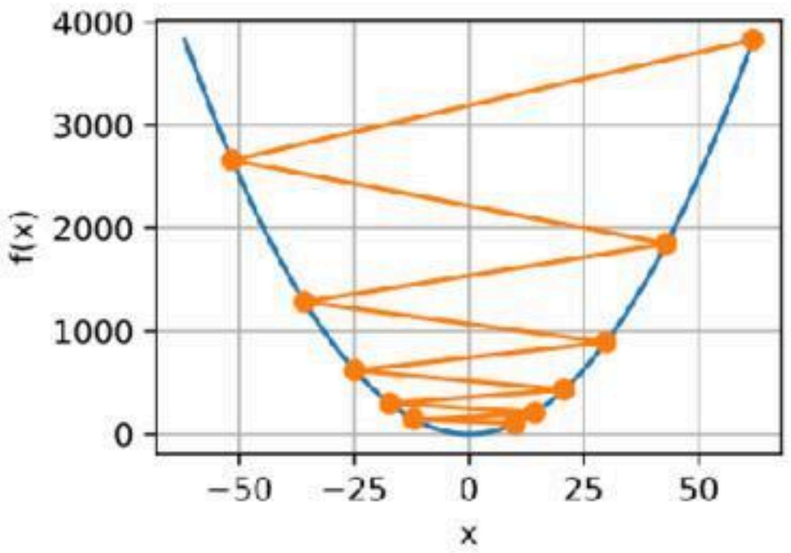


学习率

学习率 learning rate：η

- 太小：收敛慢
- 太大：发散



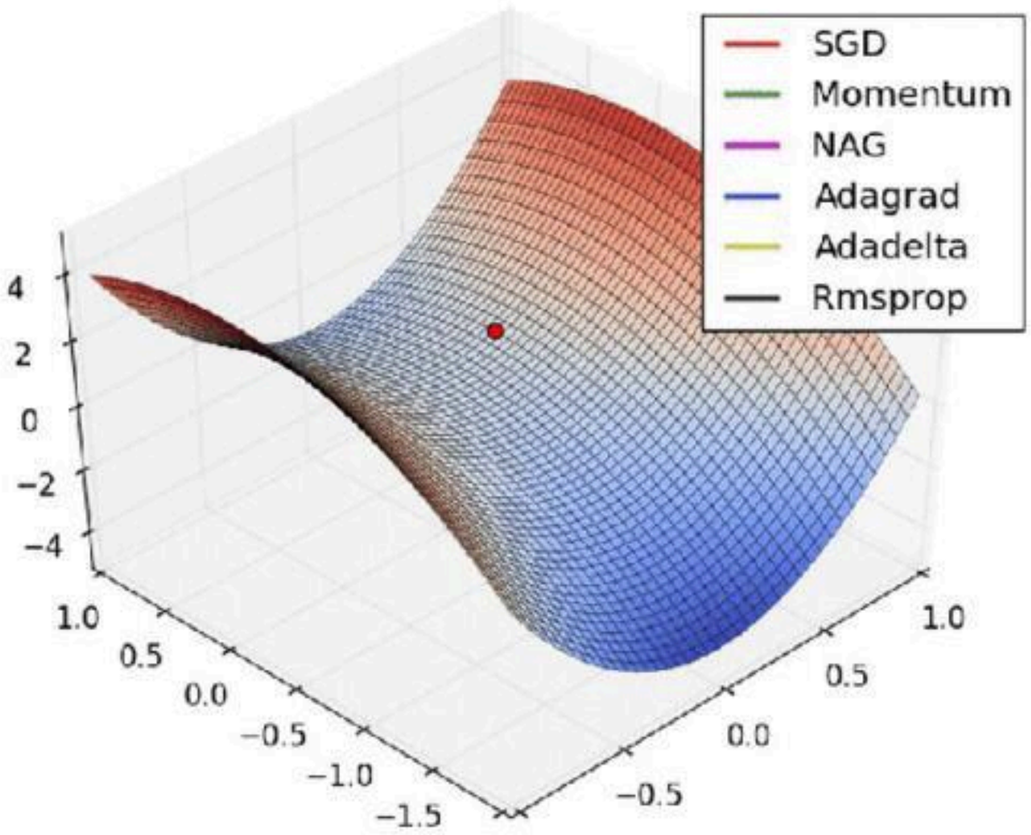


注意：地形平坦的区域步长自然变小；$x \leftarrow x - f'(x)\eta$

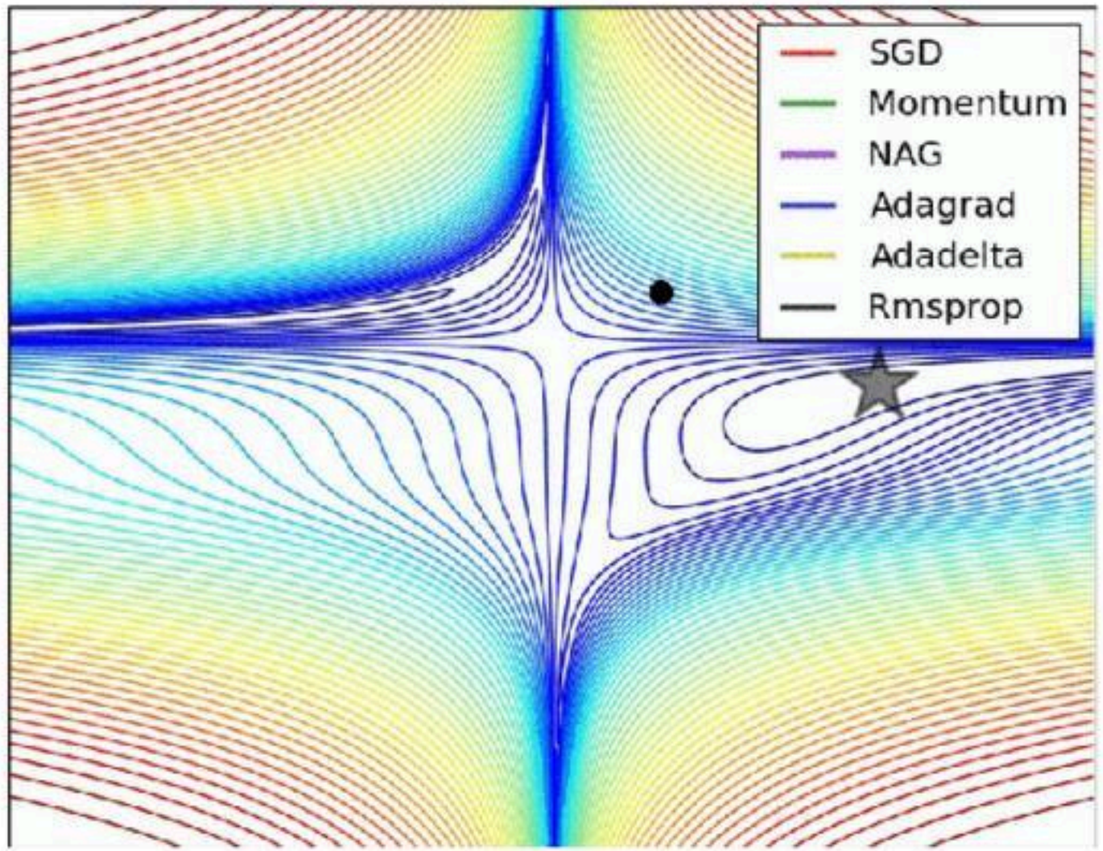
- 通常出现在极值附近的区域；更难推进优化

带动量的梯度下降

摆脱鞍点：



摆脱多个极小值：

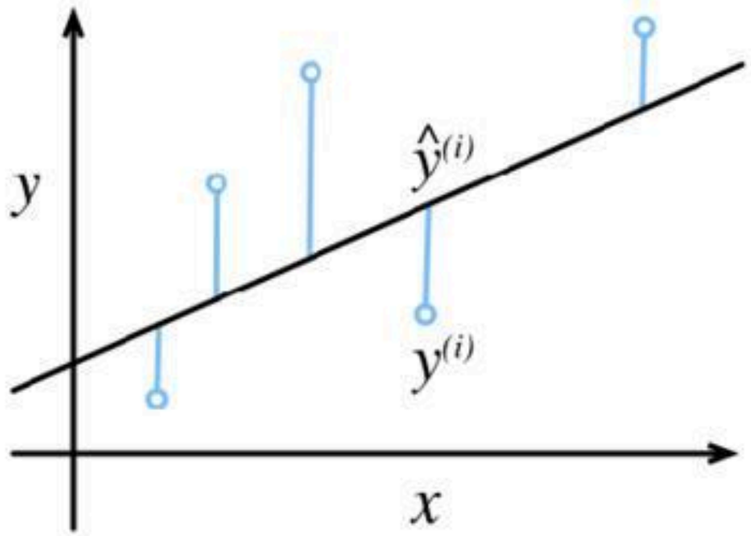


梯度下降法

(*)优化问题定义

定义损失函数，线性回归常用**平方损失**：$Loss(y, \hat{y}) = \frac{1}{2N} \sum_{i=1}^{N} (\hat{y}_i - y_i)^2$

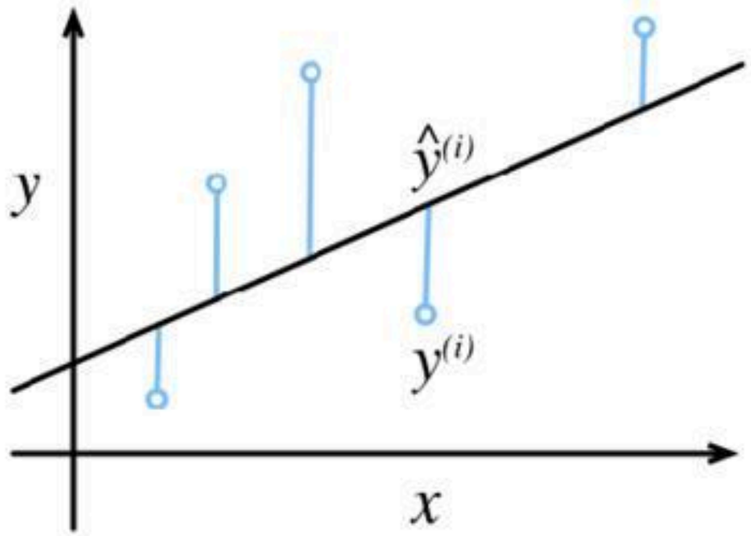
- 不可能每个点都最优："按下葫芦浮起瓢"



(*)优化问题定义

定义损失函数，线性回归常用**平方损失**：$Loss(y, \hat{y}) = \frac{1}{2N} \sum_{i=1}^{N} (\hat{y}_i - y_i)^2$

- 不可能每个点都最优："按下葫芦浮起瓢"

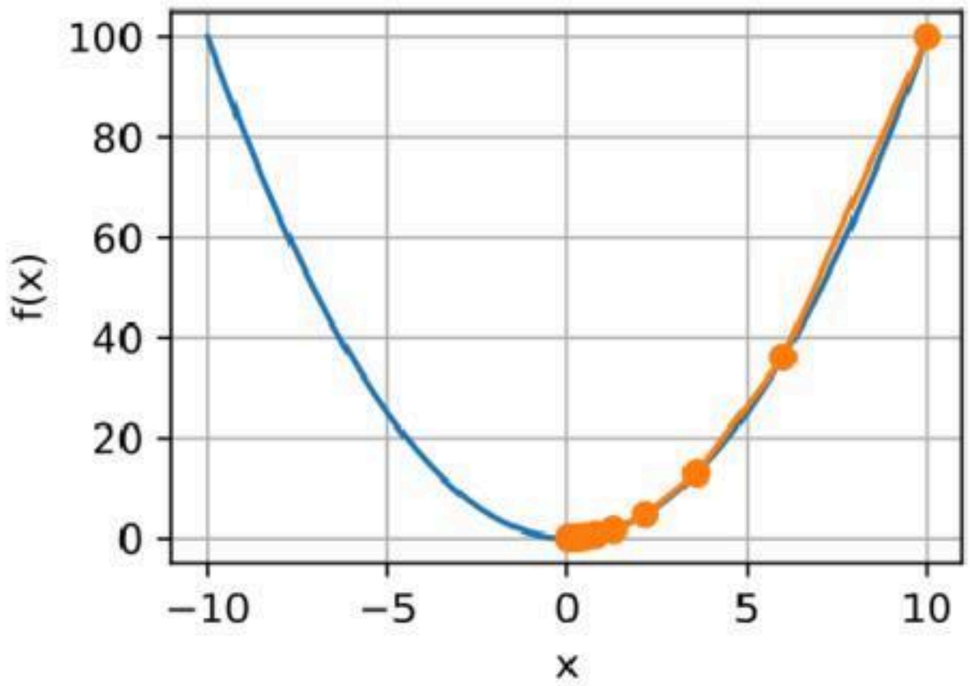


优化目标：$\mathcal{L}(\mathbf{X}, \mathbf{y}; \mathbf{w}, b) = \frac{1}{2N} \|\mathbf{y} - \mathbf{X}\mathbf{w} - b\|^2$

优化问题：$\mathbf{w}^*, b^* = \arg\min_{\mathbf{w},b} \mathcal{L}(\mathbf{X}, \mathbf{y}; \mathbf{w}, b)$

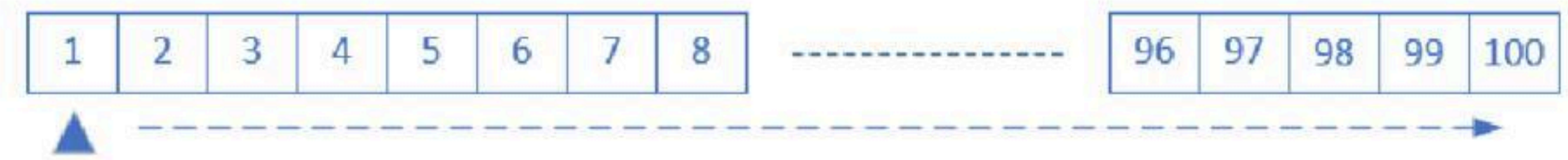
线性回归：梯度下降法

1. 初始化参数：$\mathbf{w} = \mathbf{w_0}$
2. 循环：
   1. 计算当前梯度：$\frac{\partial \mathcal{L}}{\partial \mathbf{w}}|_{\mathbf{w}}$
   2. 更新参数：$\mathbf{w} = \mathbf{w} - \eta \frac{\partial \mathcal{L}}{\partial \mathbf{w}}|_{\mathbf{w}}$



情况1：单变量、单样本

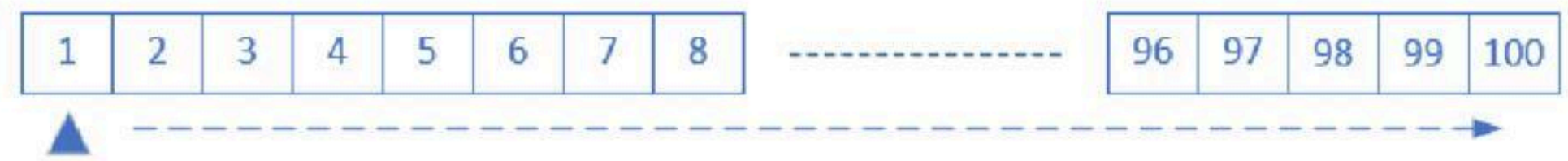
随机梯度下降 Stochastic Gradient Descent, SGD：逐个访问样本



- 训练样本：每次使用一个样本数据进行一次训练，更新一次梯度，重复以上过程

情况1：单变量、单样本

随机梯度下降 Stochastic Gradient Descent, SGD：逐个访问样本



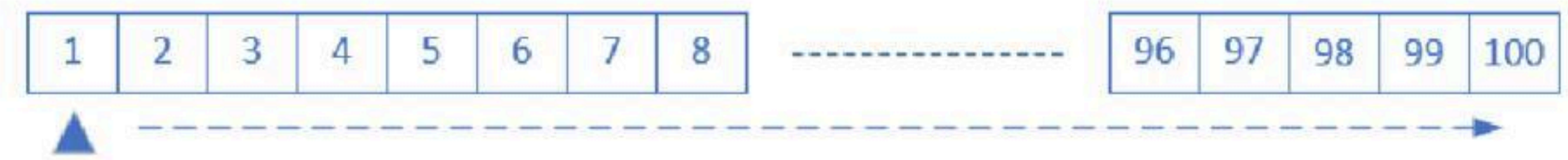
- 训练样本：每次使用一个样本数据进行一次训练，更新一次梯度，重复以上过程

损失函数（优化目标）：$\mathcal{L} = (\hat{y} - y)^2/2, \hat{y} = wx + b$

- $\frac{\partial \mathcal{L}}{\partial w} = (\hat{y} - y)x$
- $\frac{\partial \mathcal{L}}{\partial b} = (\hat{y} - y)$

情况1：单变量、单样本

随机梯度下降 Stochastic Gradient Descent, SGD：逐个访问样本



- 训练样本：每次使用一个样本数据进行一次训练，更新一次梯度，重复以上过程

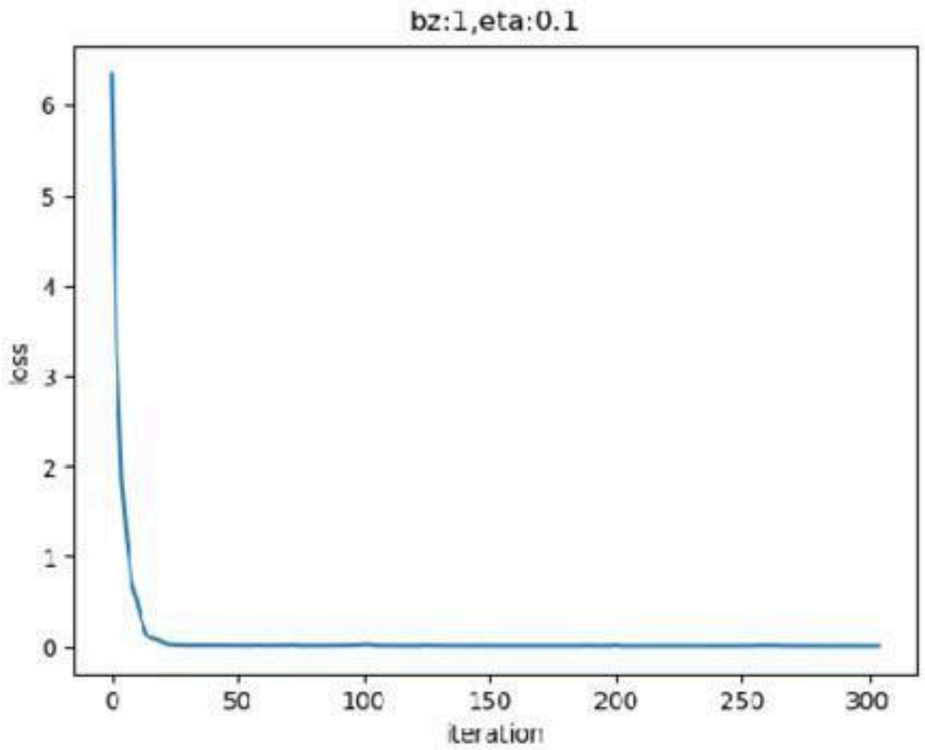
损失函数（优化目标）：$\mathcal{L} = (\hat{y} - y)^2/2, \hat{y} = wx + b$

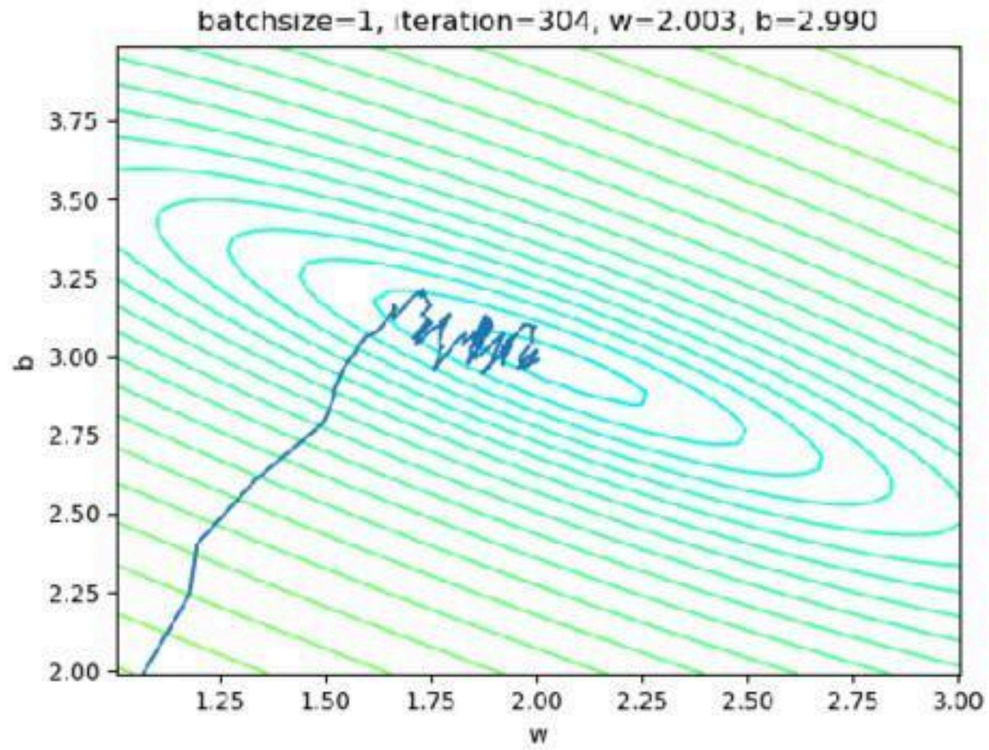
- $\frac{\partial \mathcal{L}}{\partial w} = (\hat{y} - y)x$
- $\frac{\partial \mathcal{L}}{\partial b} = (\hat{y} - y)$

更新法则：

- $w \leftarrow w - \eta(\hat{y} - y)x$：还需结合自变量考虑
- $b \leftarrow b - \eta(\hat{y} - y)$：参数更新与误差方向相反

单变量、单样本：训练曲线





- 优点：训练开始时损失值下降很快，随机性大，找到最优解的可能性大
- 缺点：受单个样本的影响最大，损失函数值波动大，到后期徘徊不前，在最优解附近震荡。不能并行计算
  - 梯度下降时，开始收敛较快，稍微有些弯曲地向中央地带靠近。到后期波动较大，找不到准确的前进方向，曲折地达到中心附近

情况2：单变量、全样本

全样本梯度下降 Full Batch Gradient Descent：访问全部样本

- 训练样本：每次使用全部数据集进行一次训练，更新一次梯度，重复以上过程

情况2：单变量、全样本

全样本梯度下降 Full Batch Gradient Descent：访问全部样本

- 训练样本：每次使用全部数据集进行一次训练，更新一次梯度，重复以上过程

需要在全部样本上计算梯度：$\nabla f = \frac{1}{N} \sum_{i=1}^{N} \nabla f_i$

- $w \leftarrow w - \eta \sum_j (y_j - \hat{y}_j) x_j$
- $b \leftarrow b - \eta \sum_j (y_j - \hat{y}_j)$

情况2：单变量、全样本

全样本梯度下降 Full Batch Gradient Descent：访问全部样本

- 训练样本：每次使用全部数据集进行一次训练，更新一次梯度，重复以上过程

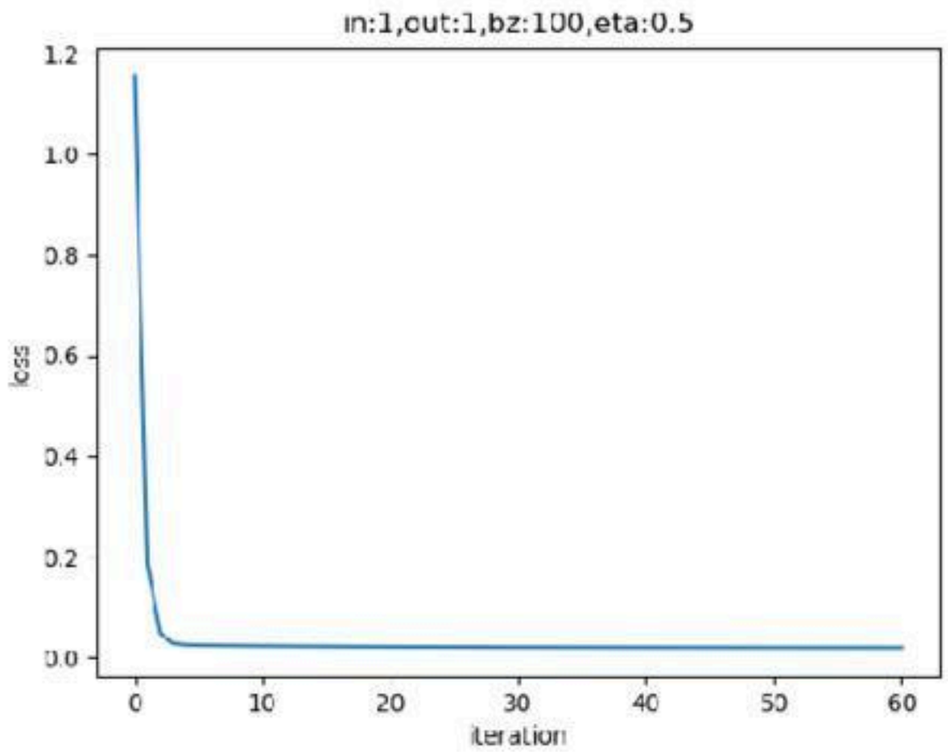
需要在全部样本上计算梯度：$\nabla f = \frac{1}{N}\sum_{i=1}^{N}\nabla f_i$

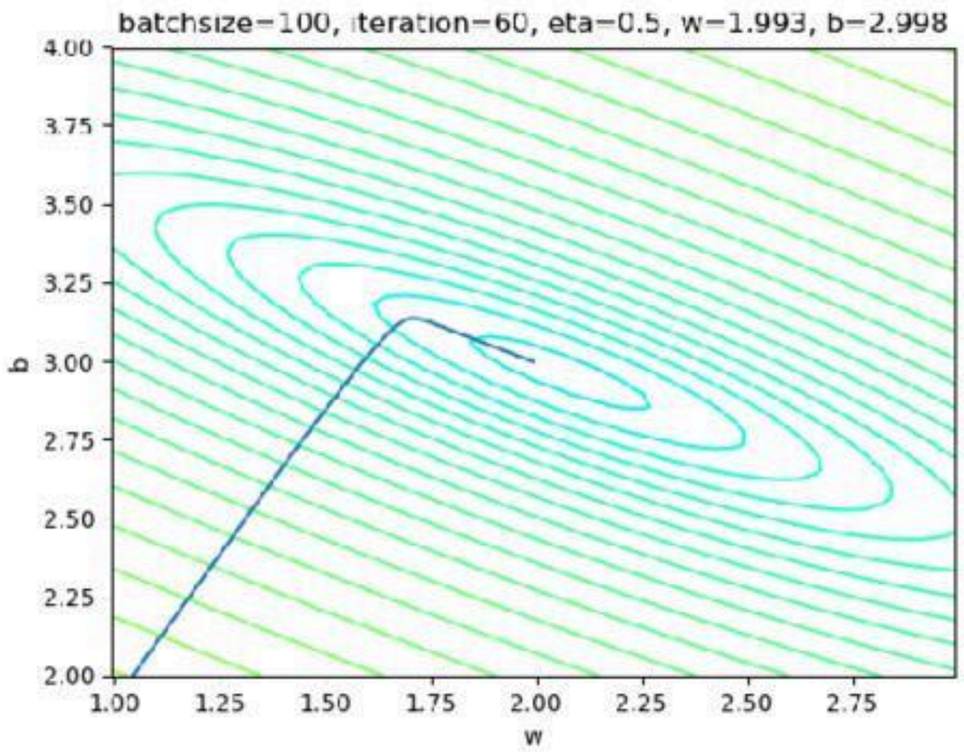
- $w \leftarrow w - \eta\sum_j(y_j - \hat{y}_j)x_j$
- $b \leftarrow b - \eta\sum_j(y_j - \hat{y}_j)$

思考：数据量太大怎么办?

- 计算复杂度：损失值、梯度值都是$O(n)$
- 空间复杂度：内存、显存放不下

单变量、全样本：训练曲线





- 优点：受单个样本的影响最小，一次计算全体样本速度快，损失函数值没有波动，到达最优点平稳。方便并行计算
- 缺点：数据量较大时不能实现（内存限制），训练过程变慢。初始值不同，可能导致获得局部最优解，并非全局最优解
  - 梯度下降时，在整个过程中只拐了一个弯儿，就直接到达了中心点

情况3：单变量、小批量

小批量梯度下降 Mini-Batch Gradient Descent：按批次访问样本

- 训练样本：选择一小部分样本进行训练，更新一次梯度，然后再选取另外一小部分样本进行训练，再更新一次梯度

情况3：单变量、小批量

小批量梯度下降 Mini-Batch Gradient Descent：按批次访问样本

- 训练样本：选择一小部分样本进行训练，更新一次梯度，然后再选取另外一小部分样本进行训练，再更新一次梯度

多样本相当于优化每个样本上的损失总和。更新法则：

- $w \leftarrow w - \eta \sum_j (y_j - \hat{y}_j) x_j$
- $b \leftarrow b - \eta \sum_j (y_j - \hat{y}_j)$

称为**批量梯度下降**，或**确定性梯度下降**

小批量随机梯度下降

随机：每次迭代对训练数据随机均匀采样，使算法更健壮

- 近似计算梯度：$\widehat{\nabla f} = \frac{1}{B}\sum_{i=1}^{B}\nabla f_i$
- B称为批量大小

小批量随机梯度下降

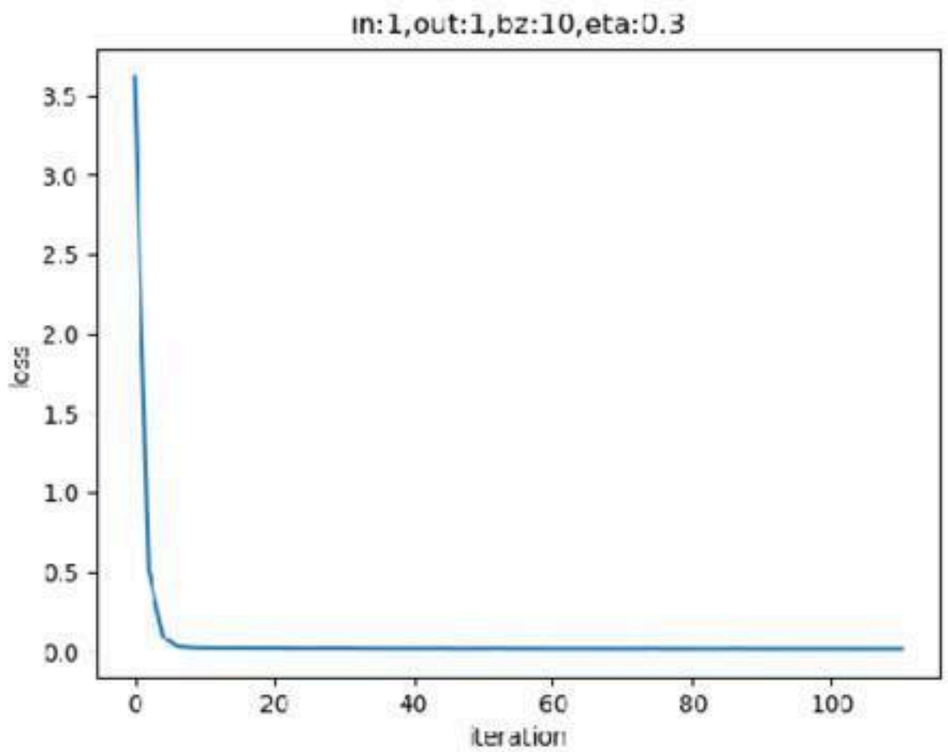
随机：每次迭代对训练数据随机均匀采样，使算法更健壮

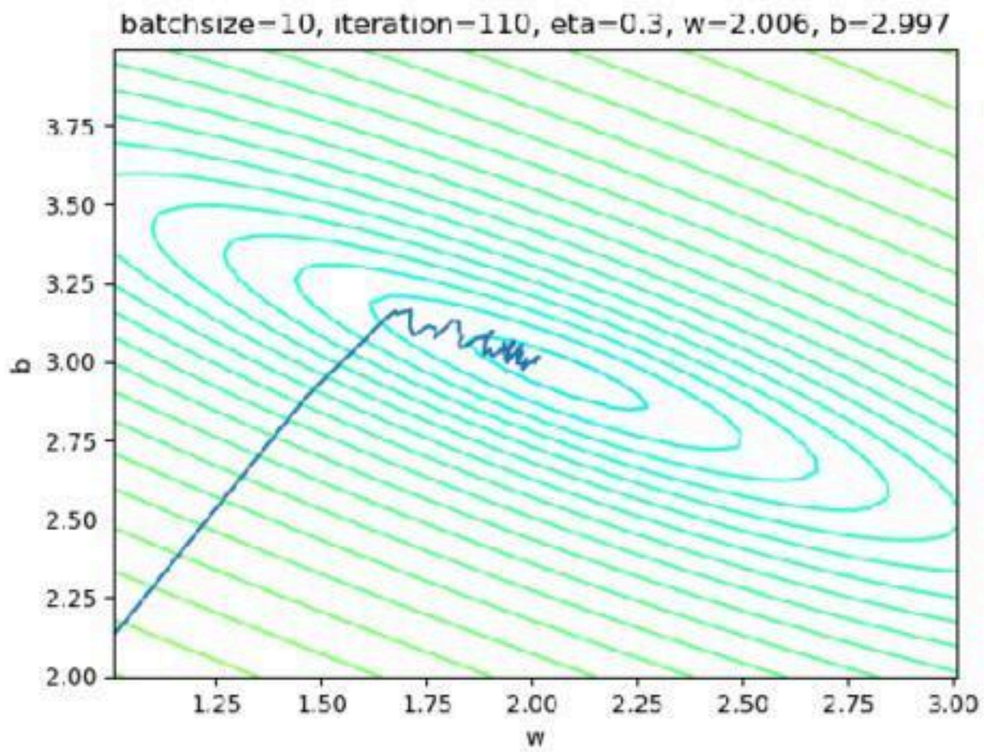
- 近似计算梯度：$\widehat{\nabla f} = \frac{1}{B}\sum_{i=1}^{B}\nabla f_i$
- B称为批量大小

(*)思考：随机梯度是对完整梯度$\nabla f = \frac{1}{N}\sum_{i=1}^{N}\nabla f_i$的**无偏估计**

- 提示：计算数学期望

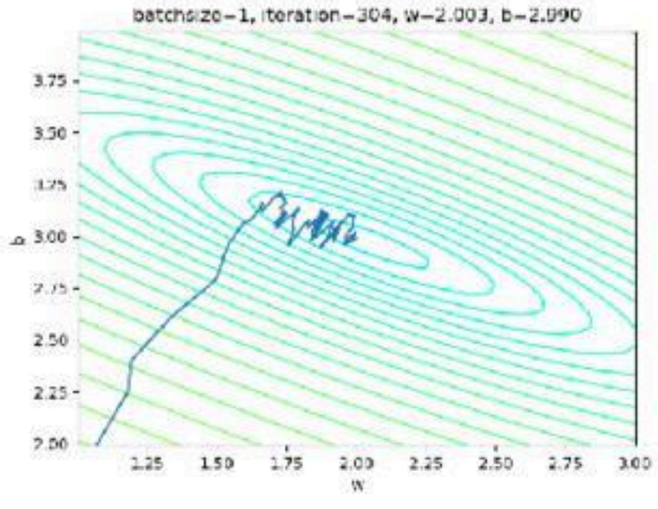
单变量、小批量：训练曲线

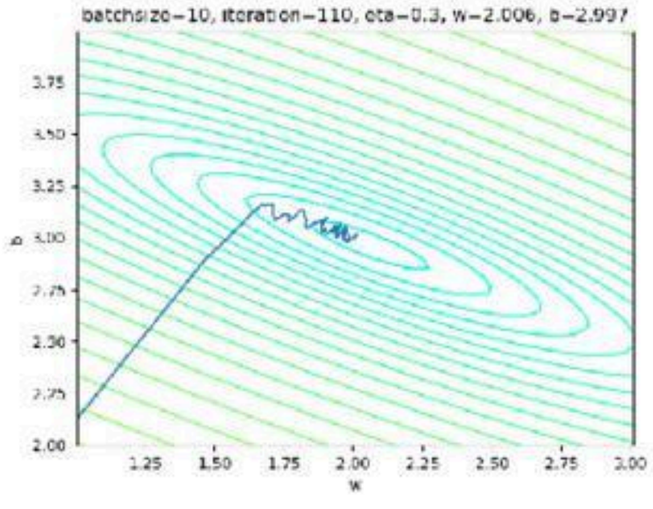


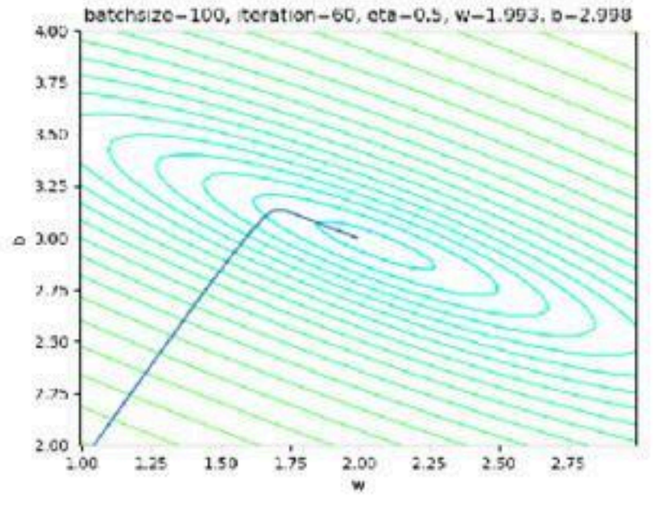


- 优点：不受单样本噪声影响，训练速度较快
- 缺点：批量大小的选择很关键，会影响训练结果
  - 梯度下降时，在接近中心时有小波动。和单样本方式比较，在中心区的波动已经缓解了很多

三种方式的比较







	单样本	小批量	全样本
批量	1	10	100
学习率	0.1	0.3	0.5
迭代数	304	110	60
epoch	3	10	60

选择批量大小

更大的批量会计算更精确的梯度；但不能太大

- **内存消耗**增加：内存限制批量上限
  - 最大可以和数据集大小相同：全样本梯度下降
- 容易陷入**极小值**：批量之间的差异不足
  - 回顾：极小值附近地形比较平坦，梯度自然减小

选择批量大小

更大的批量会计算更精确的梯度；但不能太大

- **内存消耗**增加：内存限制批量上限
  - 最大可以和数据集大小相同：全样本梯度下降
- 容易陷入**极小值**：批量之间的差异不足
  - 回顾：极小值附近地形比较平坦，梯度自然减小

小批量在学习过程中加入了噪声，会带来一些**正则化**的效果；但不能太小

- 极小批量通常难以充分利用多核架构，不利于**并行计算**
  - 决定了最小批量的数值，低于这个值的小批量处理不会减少计算时间
- 泛化误差通常在批量大小为1时最好；在线应用时也可能为1
  - 梯度估计的方差高，小批量使用**较小的学习率**，以保持稳定性
    - 但是降低学习率会使迭代次数增加

选择批量大小

更大的批量会计算更精确的梯度；但不能太大

- **内存消耗**增加：内存限制批量上限
  - 最大可以和数据集大小相同：全样本梯度下降
- 容易陷入**极小值**：批量之间的差异不足
  - 回顾：极小值附近地形比较平坦，梯度自然减小

小批量在学习过程中加入了噪声，会带来一些**正则化**的效果；但不能太小

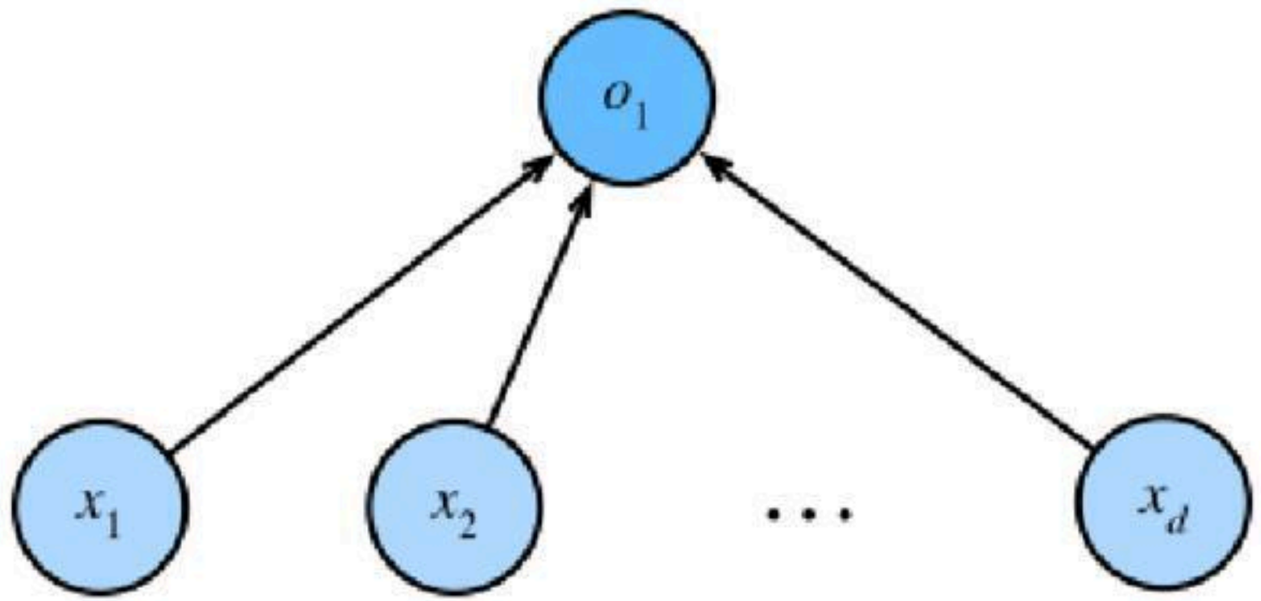
- 极小批量通常难以充分利用多核架构，不利于**并行计算**
  - 决定了最小批量的数值，低于这个值的小批量处理不会减少计算时间
- 泛化误差通常在批量大小为1时最好；在线应用时也可能为1
  - 梯度估计的方差高，小批量使用**较小的学习率**，以保持稳定性
    - 但是降低学习率会使迭代次数增加

确定批量大小：主要靠**经验估计**，如显存能够容纳的最大值；也可以二分搜索

- 某些硬件上使用特定大小的数组时，运行时间会更少，尤其是GPU
  - 通常使用**2的幂数**作为批量大小可以更快，如32,64,128,256，大模型时尝试用16

线性回归可以看成单层神经网络

$$y = w_1 x_1 + w_2 x_2 + .. + w_d x_d + b$$



注意：不存在线性的多层神经网络！

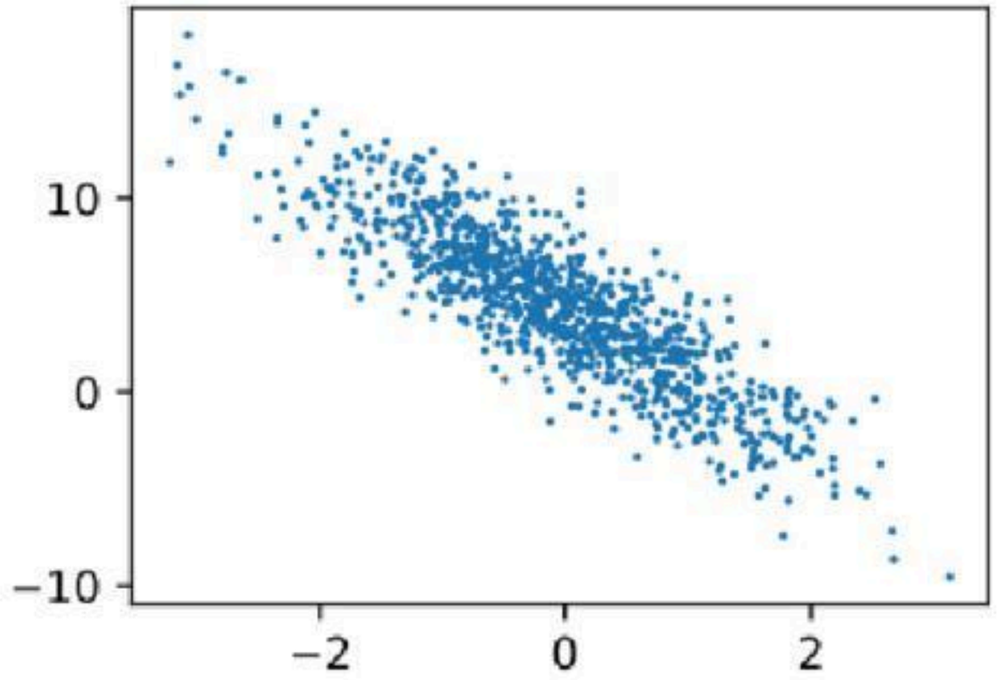
- 思考：为什么？

实验：线性回归的从零开始实现

人造数据集

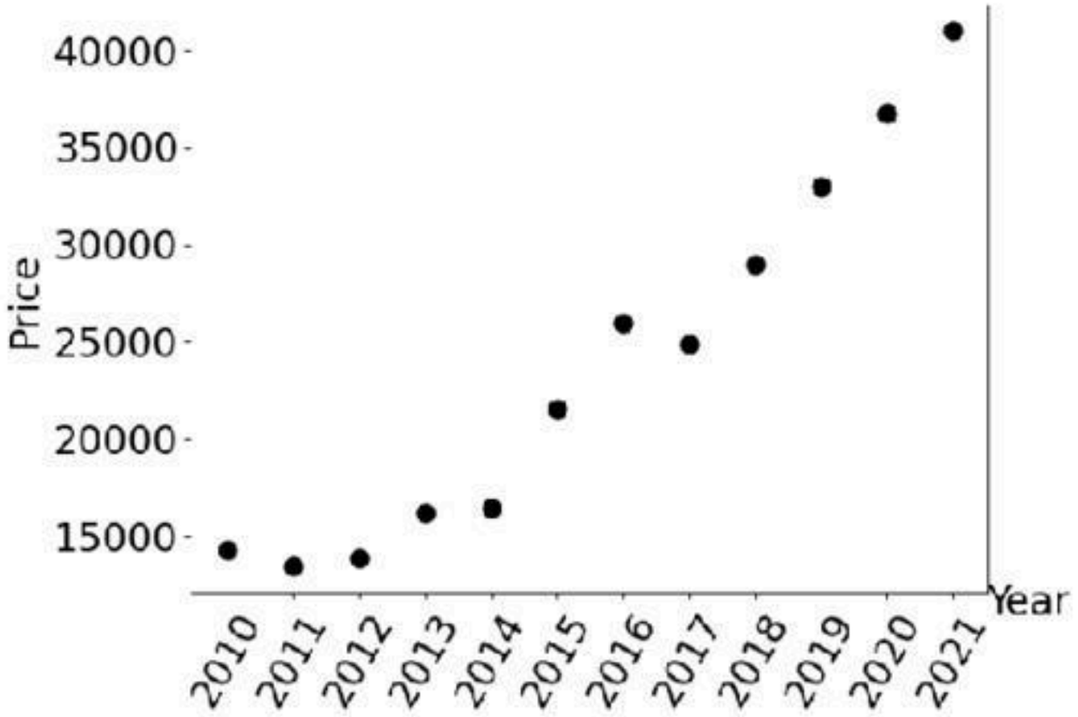
带噪音的人造数据：噪音满足高斯分布

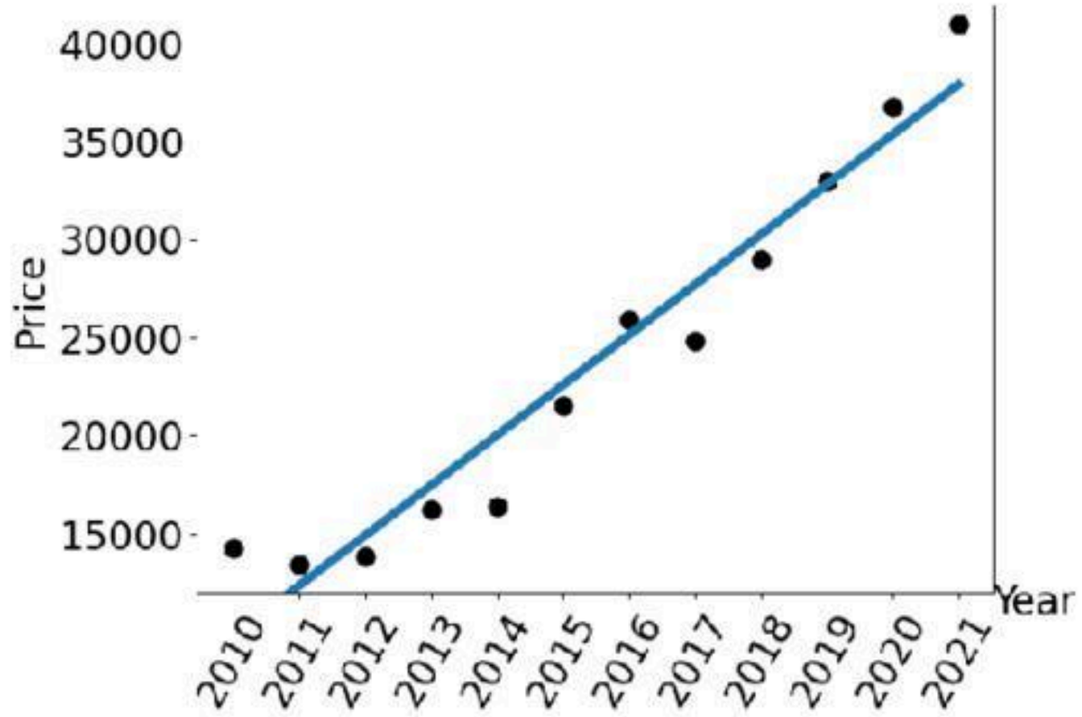
$$\mathbf{y} = \mathbf{X}\mathbf{w} + b + \epsilon$$



实验：线性回归的简洁实现

作业：线性回归模型


40000
35000
30000
25000
20000
15000
Price
Year
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021


40000
35000
30000
25000
20000
15000
Price
Year
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021

Review

本章内容

1. 线性回归模型
2. 最小二乘法
3. (*)多元微分、优化
4. 梯度下降法
5. 实验：线性回归模型

重点：线性回归问题建模；最小二乘法；梯度下降法；求解线性回归问题并实现算法。

难点：多元微分计算；优化理论；调节学习率；选择批量大小。

学习目标

1. 理解线性回归模型的一般形式、损失函数、优化算法
2. 掌握最小二乘法；掌握梯度下降法
3. 理解基于梯度的优化及随机梯度下降算法
4. 掌握使用线性回归解决预测问题的实现方法
5. 理解选取批量大小的注意事项

问题

1. 假设$\mathbf{y} = \mathbf{X}\mathbf{w}, \mathbf{X} \in \mathbb{R}^{N \times d}, \mathbf{w} \in \mathbb{R}^{d \times 1}$，求参数$\mathbf{w}$的解析解。
2. 假设$\mathbf{y} = \mathbf{X}\mathbf{w}, \mathbf{X} \in \mathbb{R}^{N \times d}, \mathbf{w} \in \mathbb{R}^{d \times 1}$，简述随机梯度下降法对参数$\mathbf{w}$的求解算法。
3. 简述梯度下降法的算法流程，及学习率对其影响。
4. 简述选取批量大小的注意事项。
5. (*) 假设$\mathbf{Y}$是(a, b, c, d)维张量，$\mathbf{X}$是(h, i, j, k, l)维张量，那么$\frac{\partial \mathbf{Y}}{\partial \mathbf{X}}$的维度是多少?

梯度下降法

优化目标

Objective: $\arg\min_{W,b} J$,

with: $J = \|\overline{y} - y\|, \overline{y} = \sum relu(\mathbf{W} * x + \mathbf{b})$

梯度下降更新

$W_1 = W_0 - \nabla J * s$

- 步长（学习率）要选取合适；
- 动量解决收敛速度、局部极值。