人工智能与自动化学院

模式识别

授课教师：杨卫东　邹腊梅

第三讲　线性回归（*Linear Regression*）

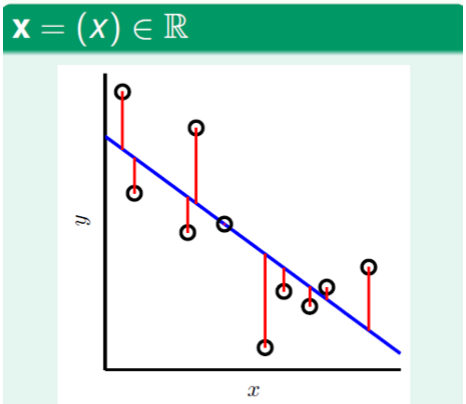
3.1 线性回归问题　（*Linear Regression Problem*）

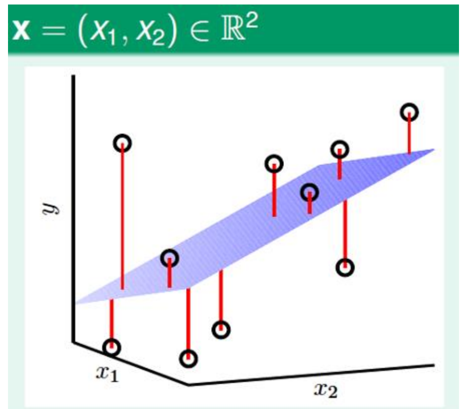
3.2 线性回归算法　（*Linear Regression Algorithm*）

3.3 梯度下降法　（*Gradient Descent*）

3.1 线性回归问题

线性回归示例





回归问题：$y \in \mathbb{R}$

3.1 线性回归问题

为什么需要回归?

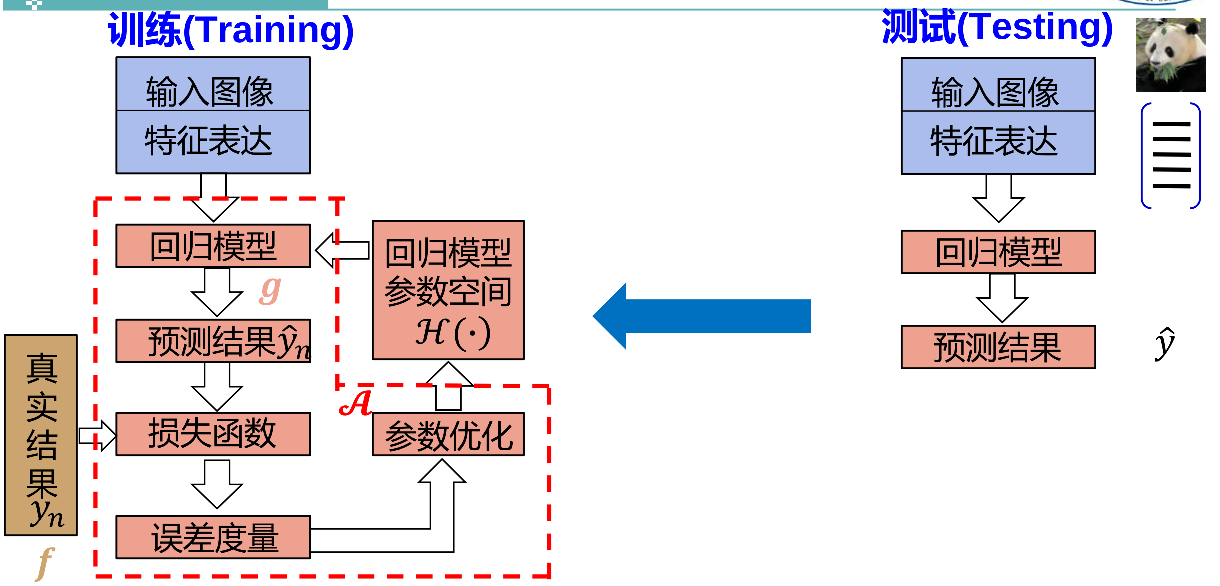
分类(*Classification*)

输出结果(*Output*)：类别标签(Class label)

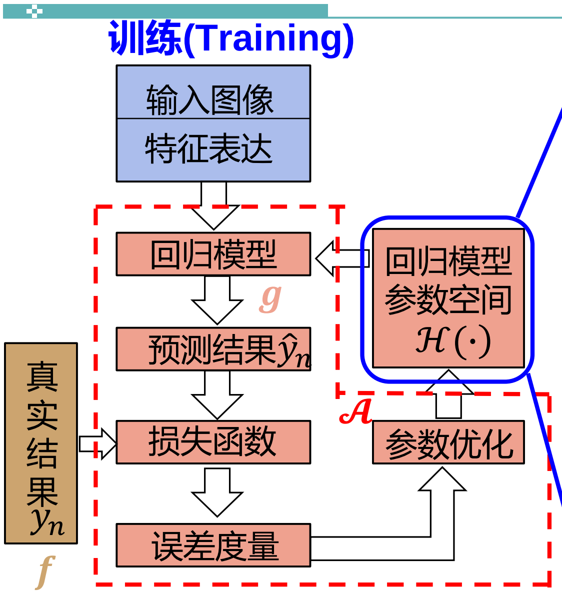
检测(分类+定位)
Detection(Classification+ Localization)

输出结果(*Output*)：类别标签+位置坐标
Class label + Box in the image (x, y, w, h)

3. 1 线性回归问题



3. 1 线性回归问题



$\mathcal{H}(\cdot)$ 像什么?

输入： $\mathbf{x} = (x_1, x_2, \ldots x_d)^T$

计算输入 $\mathbf{x}$ 的加权和： $\hat{y} = \sum_{i=1}^{d} w_i x_i$

回归模型的参数空间： $h(\quad) \in \mathcal{H}(\cdot)$

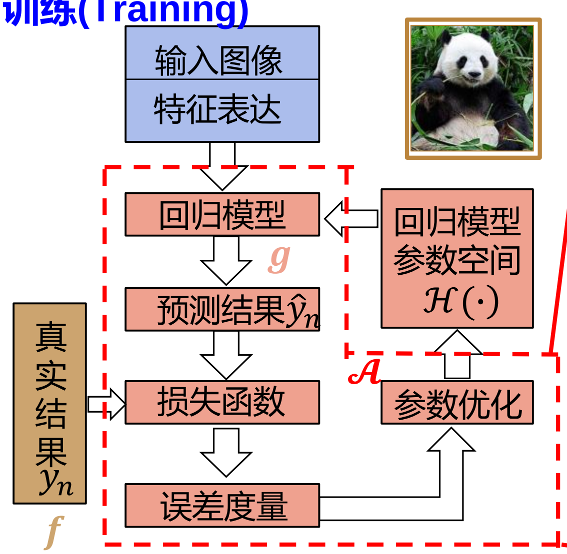
$h(\mathbf{x}) = \mathbf{w}^T\mathbf{x}$

线性回归的目的：找到误差最小的拟合模型

第三讲 线性回归（*Linear Regression*）

3.2 线性回归算法

训练(Training)



算法$\mathcal{A}$的目的是在$\mathcal{H}(h(\cdot))$中找到最优结果作为回归模型 g

- 最优结果： $g \approx f$
- 挑战： f 未知
- 学习的资源：在训练集$\mathcal{D}$上，如果每一个样本都有：$g(\mathbf{x}_n) = y_n = f(\mathbf{x}_n)$，则在训练集$\mathcal{D}$上做到了$g \approx f$
- 困难：$\mathcal{H}(h(\cdot))$ 的候选模型无穷多

3.2 线性回归算法

训练(Training)

输入图像
特征表达

回归模型 g

回归模型参数空间 $\mathcal{H}(\cdot)$

预测结果 $\hat{y}_n$

真实结果 y_n f

损失函数

$\mathcal{A}$ 参数优化

误差度量

$$\hat{y}_n = \boldsymbol{w}^T \mathbf{x}_n$$

$$L = (\hat{y}_n - y_n)^2$$

3.2 线性回归算法

训练(Training)

输入图像
特征表达

回归模型 g

回归模型参数空间 $\mathcal{H}(\cdot)$

预测结果 $\hat{y}_n$

真实结果 y_n f

损失函数

$\mathcal{A}$ 参数优化

误差度量

$$\hat{y}_n = \boldsymbol{w}^T \mathbf{x}_n$$

$$L = (\hat{y}_n - y_n)^2$$

平方误差函数(*Squared error*), L_2 损失（L_2 *Loss*）

3.2 线性回归算法

训练(Training)

输入图像
特征表达

回归模型

g

预测结果$\hat{y}_n$

损失函数

误差度量

回归模型
参数空间
$\mathcal{H}(\cdot)$

$\mathcal{A}$

参数优化

真实结果y_n

f

$$\hat{y}_n = \boldsymbol{w}^T \mathbf{x}_n$$

$$L_{in} = \frac{1}{N}\sum_{n=1}^{N}(\hat{y}_n - y_n)^2$$

计算训练样本集所有样本产生的平均损失

3.2 线性回归算法

训练(Training)

输入图像
特征表达

回归模型

g

预测结果$\hat{y}_n$

损失函数

误差度量

回归模型
参数空间
$\mathcal{H}(\cdot)$

$\mathcal{A}$

参数优化

真实结果y_n

f

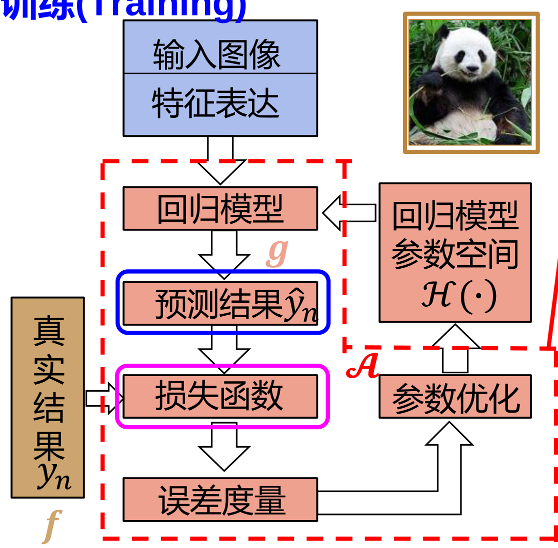
$$\hat{y}_n = \boldsymbol{w}^T \mathbf{x}_n$$

$$L_{in}(h) = \frac{1}{N}\sum_{n=1}^{N}(h(\mathbf{x}_n) - y_n)^2$$

计算训练样本集所有样本产生的平均损失

3.2 线性回归算法

训练(Training)



$$\hat{y}_n = \boldsymbol{w}^T \mathbf{x}_n$$

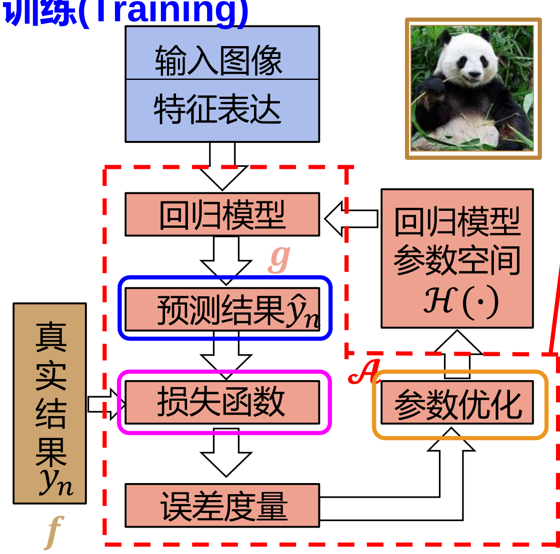
$$L_{in}(\boldsymbol{w}) = \frac{1}{N}\sum_{n=1}^{N}(\boldsymbol{w}^T\mathbf{x}_n - y_n)^2$$

计算训练样本集所有样本产生的平均损失

人工智能与自动化学院 13

3.2 线性回归算法

训练(Training)



$$\hat{y}_n = \boldsymbol{w}^T \mathbf{x}_n$$

$$L_{in}(\boldsymbol{w}) = \frac{1}{N}\sum_{n=1}^{N}(\boldsymbol{w}^T\mathbf{x}_n - y_n)^2$$

$$g = \underset{\boldsymbol{w}}{argmin}\frac{1}{N}\sum_{n=1}^{N}(\boldsymbol{w}^T\mathbf{x}_n - y_n)^2$$

人工智能与自动化学院 14

3. 2 线性回归算法

用矩阵/向量形式表示$L_{in}(\mathbf{w})$

$$L_{in}(\mathbf{w}) = \frac{1}{N}\sum_{n=1}^{N}(\mathbf{w}^T\mathbf{x}_n - y_n)^2 = \frac{1}{N}\sum_{n=1}^{N}({\mathbf{x}_n}^T\mathbf{w} - y_n)^2$$

$$= \frac{1}{N}\left\|\begin{matrix}{\mathbf{x}_1}^T\mathbf{w} - y_1\\{\mathbf{x}_2}^T\mathbf{w} - y_2\\\cdots\\{\mathbf{x}_N}^T\mathbf{w} - y_N\end{matrix}\right\|^2 = \frac{1}{N}\left\|\begin{bmatrix}--{\mathbf{x}_1}^T--\\--{\mathbf{x}_2}^T--\\\cdots\\--{\mathbf{x}_N}^T--\end{bmatrix}\mathbf{w} - \begin{bmatrix}y_1\\y_2\\\cdots\\y_N\end{bmatrix}\right\|^2$$

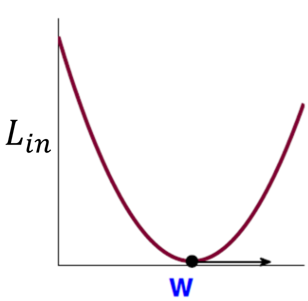
$$= \frac{1}{N}\|\mathbf{X}\mathbf{w} - Y\|^2$$

$\mathbf{X}$: $N \times (d+1)$
$\mathbf{w}$: $(d+1) \times 1$
Y: $N \times 1$

3. 2 线性回归算法

求最佳解： $\min\limits_{\mathbf{w}} L_{in}(\mathbf{w}) = \frac{1}{N}\|\mathbf{X}\mathbf{w} - Y\|^2$

L_{in} 曲线具有连续、可微、凸函数的特点



$$\nabla L_{in}(\mathbf{w}) = \begin{bmatrix}\frac{\partial L_{in}(\mathbf{w})}{\partial w_0}\\\frac{\partial L_{in}(\mathbf{w})}{\partial w_1}\\\vdots\\\frac{\partial L_{in}(\mathbf{w})}{\partial w_d}\end{bmatrix} = \begin{bmatrix}0\\0\\\vdots\\0\end{bmatrix}$$

$\nabla L_{in}(\mathbf{w}) = \mathbf{0}$时求得最佳解$\mathbf{w}^*$

3.2 线性回归算法

$\nabla L_{in}(\mathbf{w})$**求解：**

$$L_{in}(\mathbf{w}) = \frac{1}{N}\|\mathbf{X}\mathbf{w} - Y\|^2 = \frac{1}{N}(\mathbf{w}^T\mathbf{X}^T\mathbf{X}\mathbf{w} - 2\mathbf{w}^T\mathbf{X}^T\mathrm{Y} + \mathrm{Y}^T\mathrm{Y}) = \frac{1}{N}(\mathbf{w}^T A\mathbf{w} - 2\mathbf{w}^T b + c)$$

当w是单变量时

$$L_{in}(w) = \frac{1}{N}(aw^2 - 2bw + c)$$

$$\nabla L_{in}(w) = \frac{1}{N}(2aw - 2b)$$

当$\mathbf{w}$是向量时

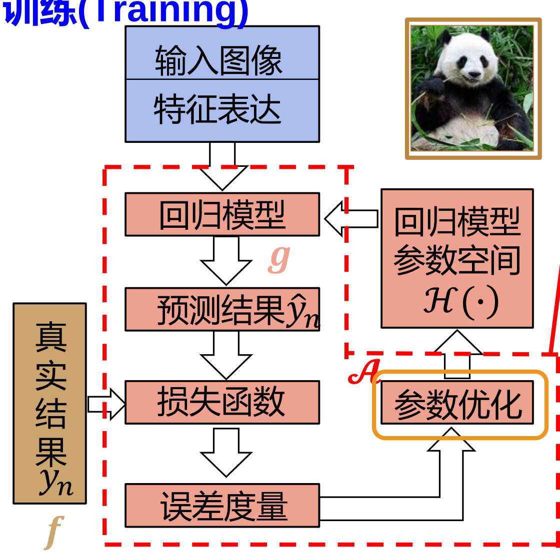
$$L_{in}(w) = \frac{1}{N}(\mathbf{w}^T A\mathbf{w} - 2\mathbf{w}^T b + c)$$

$$\nabla L_{in}(w) = \frac{1}{N}(2A\mathbf{w} - 2b)$$

$$\nabla L_{in}(\mathbf{w}) = \frac{2}{N}(\mathbf{X}^T\mathbf{X}\mathbf{w} - \mathbf{X}^T\mathrm{Y})$$

3.2 线性回归算法

训练(Training)



$$\nabla L_{in}(\mathbf{w}) = \frac{2}{N}(\mathbf{X}^T\mathbf{X}\mathbf{w} - \mathbf{X}^T\mathrm{Y}) = \mathbf{0}$$

$$\mathbf{w}^* = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathrm{Y}$$

$$g = \mathbf{w}^* = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathrm{Y} = \mathbf{X}^\dagger\ \mathrm{Y}$$

$\mathbf{X}^\dagger = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T$，称为广义逆

3.2 线性回归算法

线性回归算法

- 对训练样本集 $\mathcal{D}$ 构造输入特征向量矩阵 $\mathbf{X}$ 和输出向量 Y

$$\mathbf{X} = \begin{bmatrix} --\mathbf{x}_1{}^T-- \\ --\mathbf{x}_2{}^T-- \\ \vdots \\ --\mathbf{x}_N{}^T-- \end{bmatrix}, \qquad Y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix}$$

- 计算广义逆： $\mathbf{X}^{\dagger} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T$
- 计算回归值： $g = \boldsymbol{w}^{*} = \mathbf{X}^{\dagger}Y$

$\mathbf{X}$: $N \times (d+1)$

Y: $N \times 1$

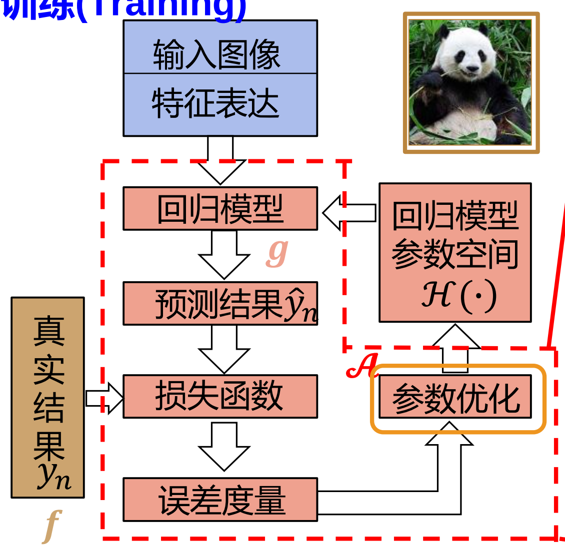
$\mathbf{X}^{\dagger}$: $(d+1) \times N$

$\boldsymbol{w}$: $(d+1) \times 1$

第三讲　线性回归（*Linear Regression*）

3. 3 梯度下降法

训练(Training)



$$\nabla L_{in}(\boldsymbol{w}) = \frac{2}{N}(\mathbf{X}^T\mathbf{X}\boldsymbol{w} - \mathbf{X}^T\mathrm{Y}) = \mathbf{0}$$

$$\boldsymbol{w}^* = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathrm{Y}$$

$$g = \boldsymbol{w}^* = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathrm{Y} = \mathbf{X}^\dagger\ \mathrm{Y}$$

$\mathbf{X}^\dagger = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T$，称为广义逆

当$N = 10000$，$d = 9999$，$\mathbf{X}$是一个巨大的矩阵，N，d可以更大，如何计算$\mathbf{X}^\dagger$?

3. 3 梯度下降法

回顾感知器算法：

- 对样本的特征向量$\mathbf{x}$和权向量$\boldsymbol{w}$ 增广化
- 初始化权向量$\boldsymbol{w}_0$ (例如： $\boldsymbol{w}_0 = \mathbf{0}$)
- $\boldsymbol{for}\ t = 0,1,2,\ldots$ (t 代表迭代次数)

  ① 进行到第t 次迭代时权向量为$\boldsymbol{w}_t$，它对样本$(\mathbf{x}_{n(t)}, y_{n(t)})$错分

  $$\mathrm{sign}(\boldsymbol{w}_t^T\mathbf{x}_{n(t)}) \neq y_n$$

  ② 通过下式对权向量$\boldsymbol{w}_t$进行更新： $\boldsymbol{w}_{t+1} = \boldsymbol{w}_t + y_n\mathbf{x}_{n(t)}$

…直到满足停止条件，此时的$\boldsymbol{w}_{t+1}$作为学到的g

3.3 梯度下降法

回顾感知器算法：

- 对样本的特征向量$\mathbf{x}$和权向量$\boldsymbol{w}$ 增广化
- 初始化权向量$\boldsymbol{w}_0$ (例如： $\boldsymbol{w}_0 = \mathbf{0}$)
- $\boldsymbol{for}\ t = 0,1,2,\dots$ (t 代表迭代次数)

① 进行到第t 次迭代时权向量为$\boldsymbol{w}_t$，它对样本$(\mathbf{x}_{n(t)}, y_{n(t)})$错分

$$\operatorname{sign}(\boldsymbol{w}_t^T \mathbf{x}_{n(t)}) \neq y_n$$

② 通过下式对权向量$\boldsymbol{w}_t$进行更新： $\boldsymbol{w}_{t+1} = \boldsymbol{w}_t + y_n \mathbf{x}_{n(t)}$

① 对某些样本n，通过下式对权向量$\boldsymbol{w}_t$进行更新：

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t + [\![\operatorname{sign}(\boldsymbol{w}_t^T \mathbf{x}_{n(t)}) \neq y_n]\!] \mathbf{x}_{n(t)}$$

…直到满足停止条件，此时的$\boldsymbol{w}_{t+1}$作为学到的g

3.3 梯度下降法

回顾感知器算法：

- 对样本的特征向量$\mathbf{x}$和权向量$\boldsymbol{w}$ 增广化
- 初始化权向量$\boldsymbol{w}_0$ (例如： $\boldsymbol{w}_0 = \mathbf{0}$)
- $\boldsymbol{for}\ t = 0,1,2,\dots$ (t 代表迭代次数)

① 对某些样本n，通过下式对权向量$\boldsymbol{w}_t$进行更新：

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t + 1 \cdot ([\![\operatorname{sign}\left(\boldsymbol{w}_t^T \mathbf{x}_{n(t)}\right) \neq y_n]\!] \mathbf{x}_{n(t)})$$

…直到满足停止条件，此时的$\boldsymbol{w}_{t+1}$作为学到的g

3.3 梯度下降法

回顾感知器算法：

- 对样本的特征向量$\mathbf{x}$和权向量$\boldsymbol{w}$ 增广化
- 初始化权向量$\boldsymbol{w}_0$ (例如： $\boldsymbol{w}_0 = \mathbf{0}$)
- $\boldsymbol{for}\ t = 0,1,2,\ldots$ (t 代表迭代次数)

① 对某些样本n，通过下式对权向量$\boldsymbol{w}_t$进行更新：

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t + \underbrace{1}_{\eta} \cdot \underbrace{(\llbracket \mathrm{sign}(\boldsymbol{w}_t^T \mathbf{x}_{n(t)}) \neq y_n \rrbracket \mathbf{x}_{n(t)})}_{\mathbf{v}}$$

…直到满足停止条件，此时的$\boldsymbol{w}_{t+1}$作为学到的g

算法可理解成通过选择(η，$\mathbf{v}$)，以及确定“停止条件”的找到最佳解的迭代优化过程

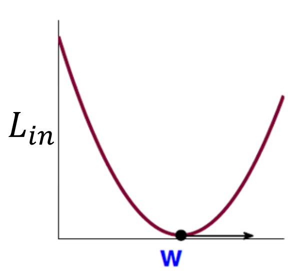
3.3 梯度下降法

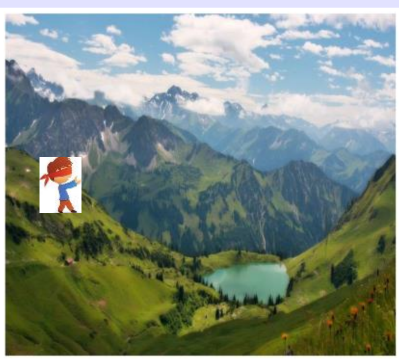
迭代优化：

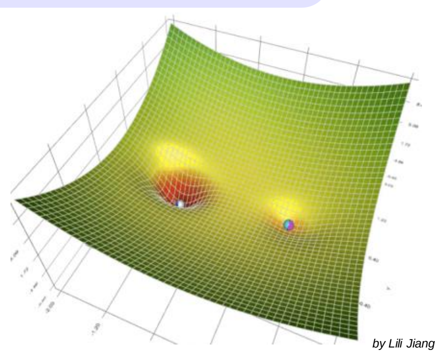
- $\boldsymbol{for}\ t = 0,1,2,\ldots$ (t 代表迭代次数)

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t + \eta \cdot \mathbf{v}$$

…直到满足停止条件，此时的$\boldsymbol{w}_{t+1}$作为学到的g







3. 3 梯度下降法

迭代优化：

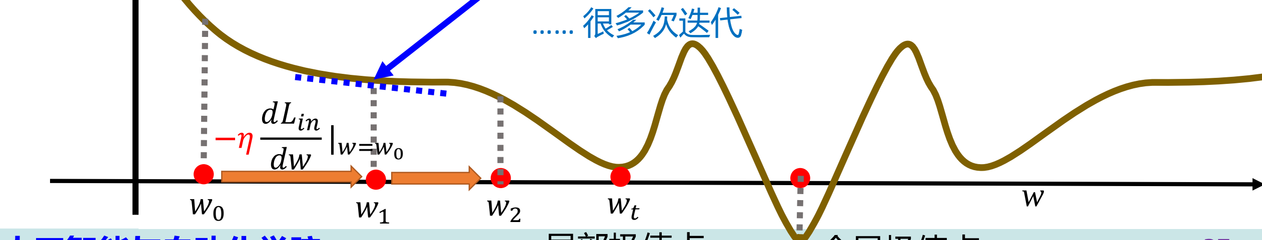
假设$L_{in}(w)$是单变量w的函数　　　$w^* = arg\min_{w} L_{in}(w)$

Loss $L_{in}(w)$

η 称为学习率 "*learning rate*"

- 随机地选择一初始值w_0
- 计算 $\frac{dL_{in}}{dw}|_{w=w_0}$　　　$w_1 \leftarrow w_0 - \eta \frac{dL_{in}}{dw}|_{w=w_0}$
- 计算 $\frac{dL_{in}}{dw}|_{w=w_1}$　　　$w_2 \leftarrow w_1 - \eta \frac{dL_{in}}{dw}|_{w=w_1}$

...... 很多次迭代



3. 3 梯度下降法

迭代优化：

假设$L_{in}(w)$是单变量w的函数　　　$w^* = arg\min_{w} L_{in}(w)$

Loss $L_{in}(w)$

η 称为学习率 "*learning rate*"

- 随机地选择一初始值w_0
- 计算 $\frac{dL_{in}}{dw}|_{w=w_0}$　　　$w_1 \leftarrow w_0 - \eta \frac{dL_{in}}{dw}|_{w=w_0}$
- 计算 $\frac{dL_{in}}{dw}|_{w=w_1}$　　　$w_2 \leftarrow w_1 - \eta \frac{dL_{in}}{dw}|_{w=w_1}$

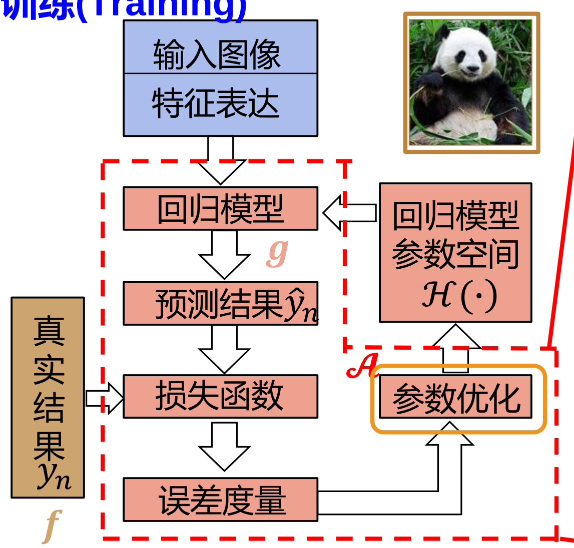
...... 很多次迭代

固定学习率的梯度下降法：

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \eta \nabla L_{in}(\mathbf{w}_t)$$

3. 3 梯度下降法

训练(Training)



梯度下降法:

$$\boldsymbol{w}_{t+1} \leftarrow \boldsymbol{w}_t - \eta \nabla L_{in}(\boldsymbol{w}_t)$$

$$L_{in}(\boldsymbol{w}) = \frac{1}{N}\|\mathbf{X}\boldsymbol{w} - Y\|^2$$

$$\nabla L_{in}(\boldsymbol{w}) = \frac{2}{N}(\mathbf{X}^T\mathbf{X}\boldsymbol{w} - \mathbf{X}^T Y)$$

$$L_{in}(\boldsymbol{w}) = \frac{1}{N}\sum_{n=1}^{N}(\boldsymbol{w}^T\mathbf{x}_n - y_n)^2$$

$$\nabla L_{in}(\boldsymbol{w}) = \frac{2}{N}\sum_{n=1}^{N}(\boldsymbol{w}^T\mathbf{x}_n - y_n)\mathbf{x}_n$$

3. 2 线性回归算法

梯度下降法实现线性回归

- 初始化权向量$\boldsymbol{w}_0$
- $\boldsymbol{for}\ t = 0,1,2,\ldots$ (t 代表迭代次数)

  ① 计算梯度： $\nabla L_{in}(\boldsymbol{w}) = \sum_{n=1}^{N}(\boldsymbol{w}^T\mathbf{x}_n - y_n)\mathbf{x}_n$

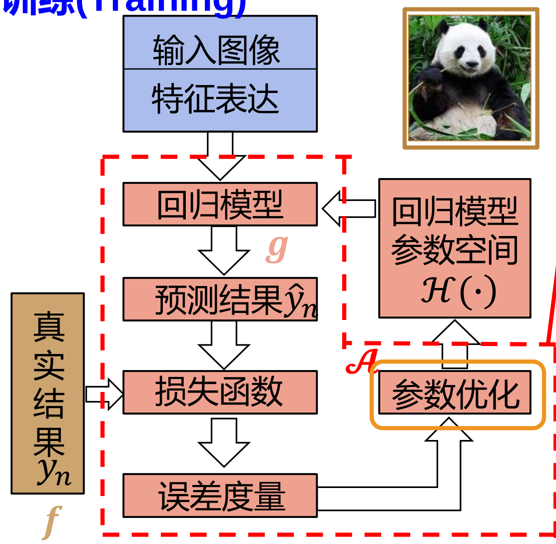
  ② 对权向量$\boldsymbol{w}_t$进行更新：$\boldsymbol{w}_{t+1} \leftarrow \boldsymbol{w}_t - \eta\nabla L_{in}(\boldsymbol{w}_t)$

…直到$\nabla L_{in}(\boldsymbol{w}) = \mathbf{0}$，或者迭代足够多次数

返回最终的$\boldsymbol{w}_{t+1}$作为学到的g

3. 3 梯度下降法

训练(Training)



梯度下降法:

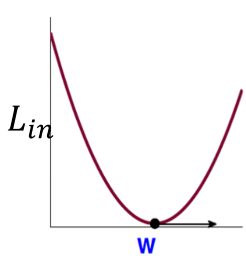
$$\nabla L_{in}(\mathbf{w}) = \sum_{n=1}^{N} (\mathbf{w}^T \mathbf{x}_n - y_n)\mathbf{x}_n$$

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \eta \nabla L_{in}(\mathbf{w}_t)$$

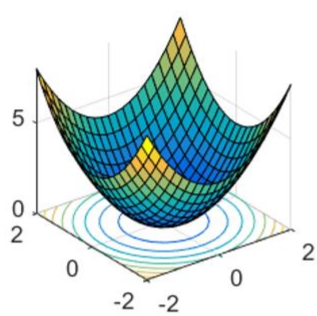
- *问题1：学习率η如何取值?*

3. 3 梯度下降法

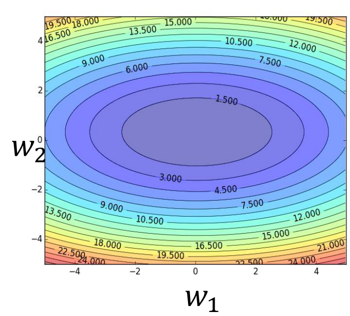
损失函数曲线的可视化:

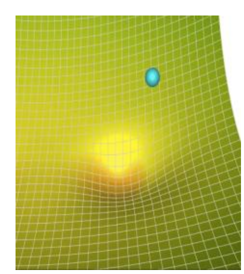


单变量w的
损失函数曲线



两个变量w的
损失函数曲线





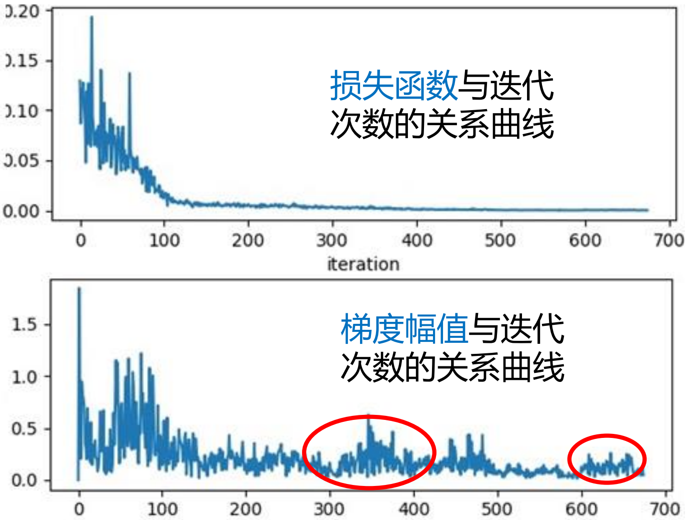
损失函数曲线的等高线表示

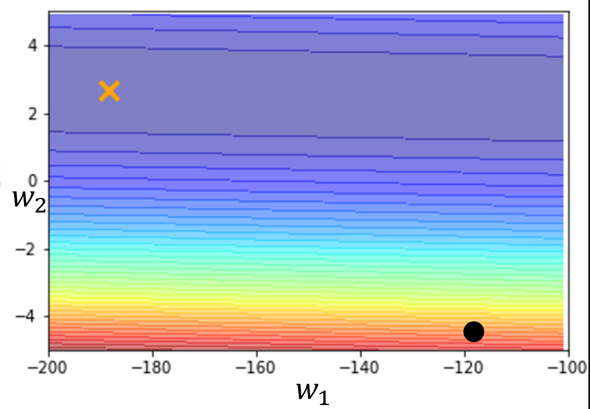
(*红色陡峭，蓝色平缓*)

3.3 梯度下降法

学习率 η的取值讨论：

此时已经到达谷底了?





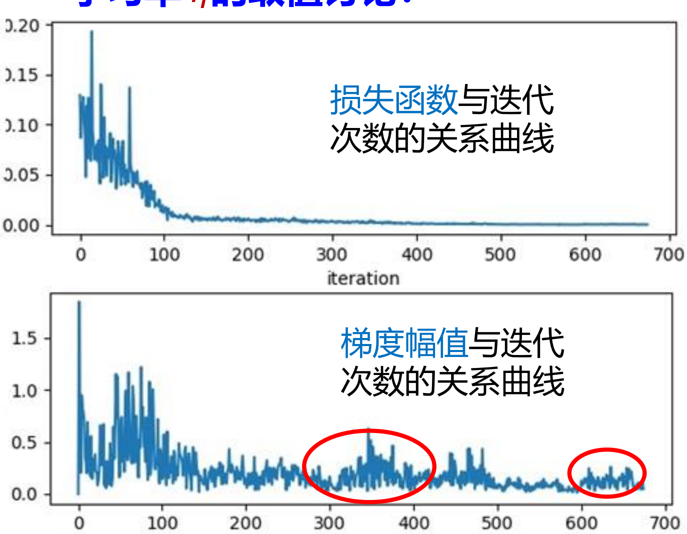
损失函数曲线的等高线表示

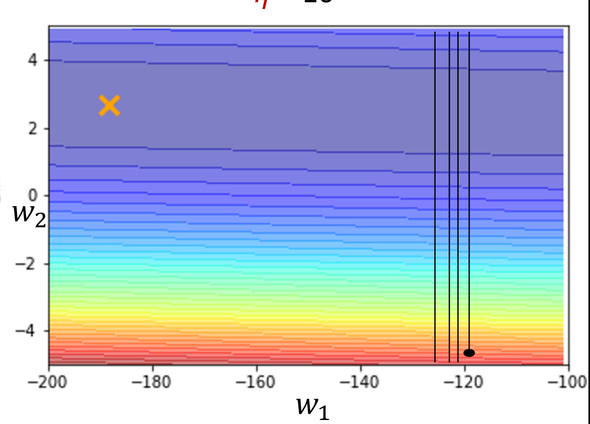
3.3 梯度下降法

学习率 η的取值讨论：

此时已经到达谷底了?

η = 10^{-2}

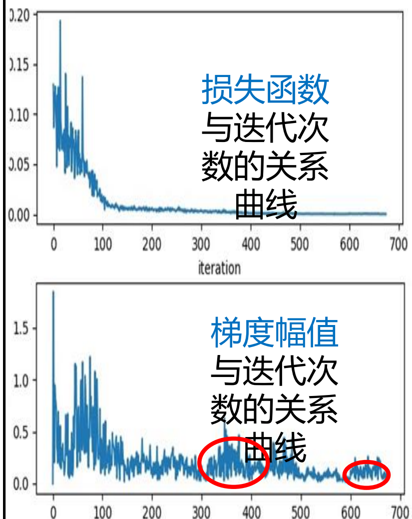




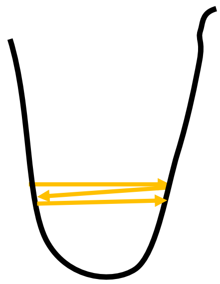
损失函数曲线的等高线表示

3.3 梯度下降法

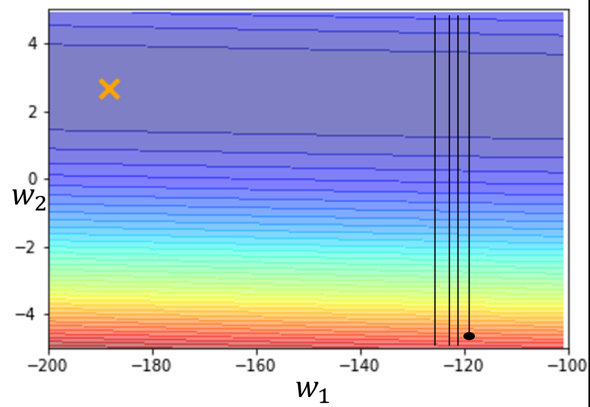
学习率 η 的取值讨论：



此时已经到达谷底了？



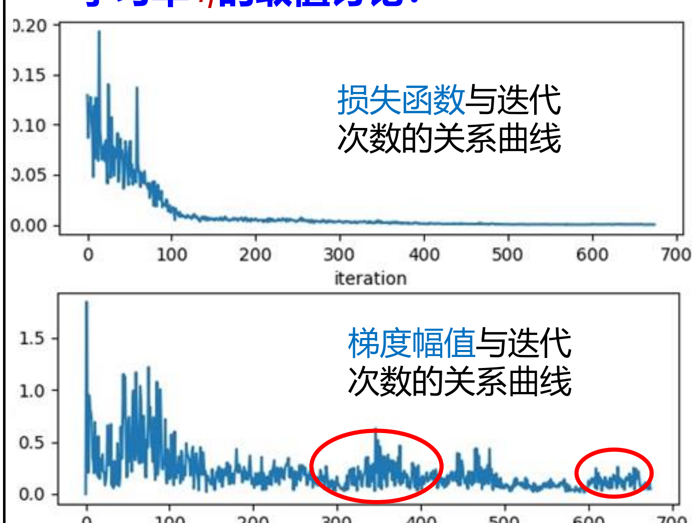
$\eta = 10^{-2}$



损失函数曲线的等高线表示

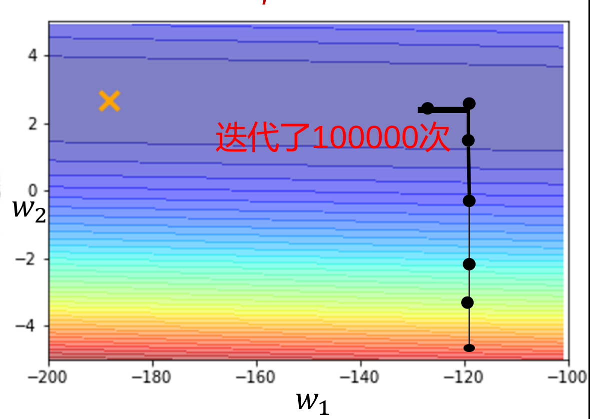
3.3 梯度下降法

学习率 η 的取值讨论：



此时已经到达谷底了？

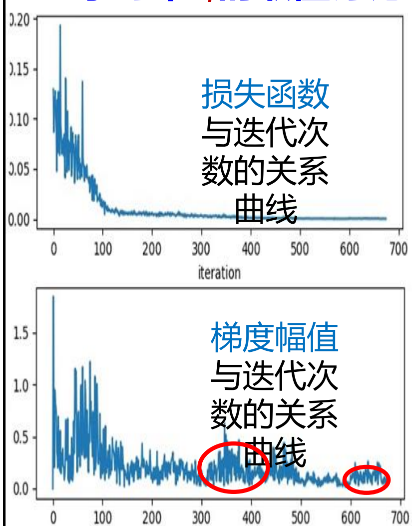
$\eta = 10^{-7}$

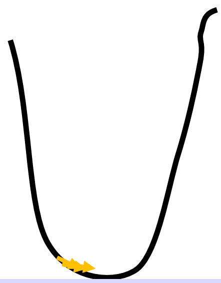


损失函数曲线的等高线表示

3.3 梯度下降法

学习率 η 的取值讨论：

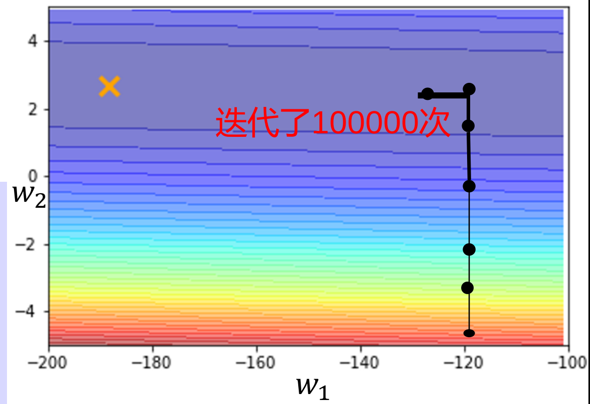




- 学习率的取值应该与梯度辐值相关
- 每个 w_i 的学习率取值应该有所不同

此时已经到达谷底了?

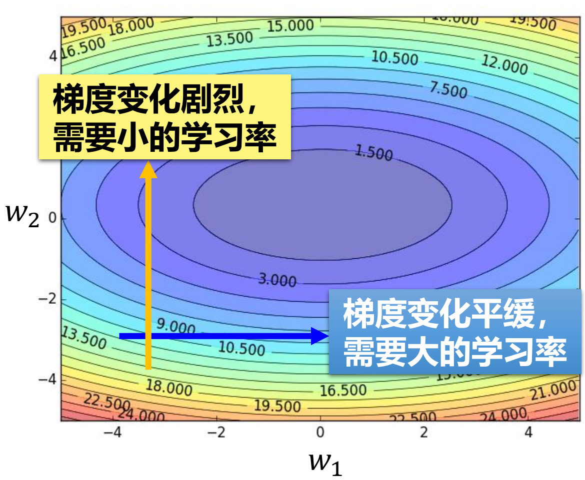
$\eta = 10^{-7}$



损失函数曲线的等高线表示

3.3 梯度下降法

学习率 η 的取值讨论：



$$\boldsymbol{w}_{t+1} \leftarrow \boldsymbol{w}_t - \eta \nabla L_{in}(\boldsymbol{w}_t)$$

$$w_{i,t+1} \leftarrow w_{i,t} - \eta \frac{\partial L_{in}}{\partial w_{i,t}}$$

$$w_{i,t+1} \leftarrow w_{i,t} - \boxed{\frac{\eta}{\sigma_{i,t}}} \frac{\partial L_{in}}{\partial w_{i,t}}$$

与 i 有关，
与迭代时刻的梯度有关

3.3 梯度下降法

Root Mean Square:

$$w_{i,t+1} \leftarrow w_{i,t} - \frac{\eta}{\sigma_{i,t}} \frac{\partial L_{in}}{\partial w_{i,t}}$$

$$w_{i,1} \leftarrow w_{i,0} - \frac{\eta}{\sigma_{i,0}} \frac{\partial L_{in}}{\partial w_{i,0}} \qquad \sigma_{i,0} = \sqrt{\left(\frac{\partial L_{in}}{\partial w_{i,0}}\right)^2}$$

$$w_{i,2} \leftarrow w_{i,1} - \frac{\eta}{\sigma_{i,1}} \frac{\partial L_{in}}{\partial w_{i,1}} \qquad \sigma_{i,1} = \sqrt{\frac{1}{2}\left[\left(\frac{\partial L_{in}}{\partial w_{i,0}}\right)^2 + \left(\frac{\partial L_{in}}{\partial w_{i,1}}\right)^2\right]}$$

$$w_{i,3} \leftarrow w_{i,2} - \frac{\eta}{\sigma_{i,2}} \frac{\partial L_{in}}{\partial w_{i,2}} \qquad \sigma_{i,2} = \sqrt{\frac{1}{3}\left[\left(\frac{\partial L_{in}}{\partial w_{i,0}}\right)^2 + \left(\frac{\partial L_{in}}{\partial w_{i,1}}\right)^2 + \left(\frac{\partial L_{in}}{\partial w_{i,2}}\right)^2\right]}$$

$$w_{i,t+1} \leftarrow w_{i,t} - \frac{\eta}{\sigma_{i,t}} \frac{\partial L_{in}}{\partial w_{i,t}} \qquad \sigma_{i,t} = \sqrt{\frac{1}{t+1}\sum_{t=0}^{t}\left(\frac{\partial L_{in}}{\partial w_{i,t}}\right)^2}$$

3.3 梯度下降法（AdaGrad）

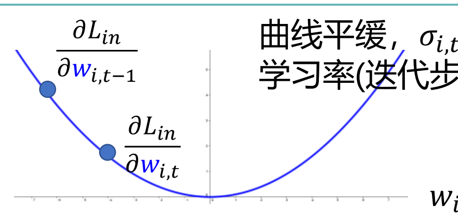
自适应动态学习率

(Learning rate adapts dynamically):

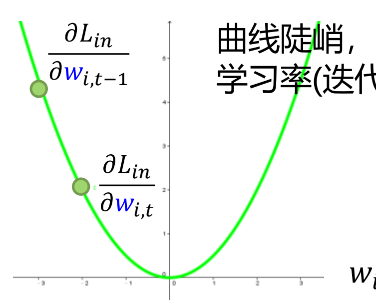
$$w_{i,t+1} \leftarrow w_{i,t} - \frac{\eta}{\sigma_{i,t}} \frac{\partial L_{in}}{\partial w_{i,t}}$$

$$\sigma_{i,t} = \sqrt{\frac{1}{t+1}\sum_{t=0}^{t}\left(\frac{\partial L_{in}}{\partial w_{i,t}}\right)^2}$$

AdaGrad



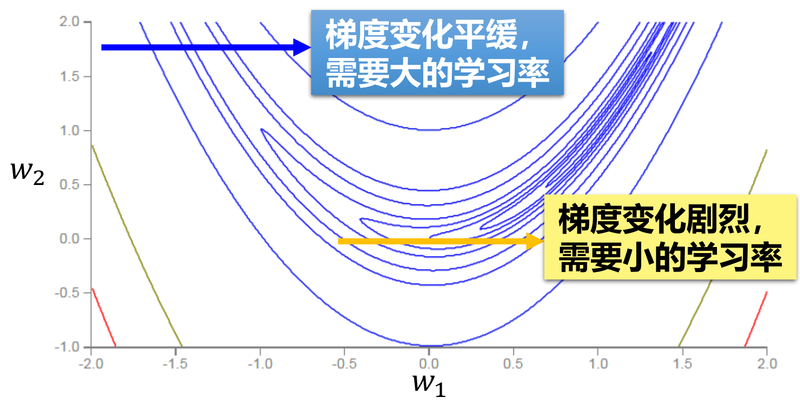
曲线平缓，$\sigma_{i,t}$小
学习率(迭代步长)大



曲线陡峭，$\sigma_{i,t}$大
学习率(迭代步长)小

3.3 梯度下降法

自适应动态学习率(Learning rate adapts dynamically):



3.3 梯度下降法（ RMSProp）

RMSProp:

$$w_{i,t+1} \leftarrow w_{i,t} - \frac{\eta}{\sigma_{i,t}} \frac{\partial L_{in}}{\partial w_{i,t}}$$

$$w_{i,1} \leftarrow w_{i,0} - \frac{\eta}{\sigma_{i,0}} \frac{\partial L_{in}}{\partial w_{i,0}} \qquad \sigma_{i,0} = \sqrt{(\frac{\partial L_{in}}{\partial w_{i,0}})^2}$$

$$0 < \alpha < 1$$

$$w_{i,2} \leftarrow w_{i,1} - \frac{\eta}{\sigma_{i,1}} \frac{\partial L_{in}}{\partial w_{i,1}} \qquad \sigma_{i,1} = \sqrt{\alpha(\sigma_{i,0})^2 + (1-\alpha)(\frac{\partial L_{in}}{\partial w_{i,1}})^2}$$

$$w_{i,3} \leftarrow w_{i,2} - \frac{\eta}{\sigma_{i,2}} \frac{\partial L_{in}}{\partial w_{i,2}} \qquad \sigma_{i,2} = \sqrt{\alpha(\sigma_{i,1})^2 + (1-\alpha)(\frac{\partial L_{in}}{\partial w_{i,2}})^2}$$

$$w_{i,t+1} \leftarrow w_{i,t} - \frac{\eta}{\sigma_{i,t}} \frac{\partial L_{in}}{\partial w_{i,t}} \qquad \sigma_{i,t} = \sqrt{\alpha(\sigma_{i,t-1})^2 + (1-\alpha)(\frac{\partial L_{in}}{\partial w_{i,t}})^2}$$

3.3 梯度下降法（ RMSProp）

RMSProp:

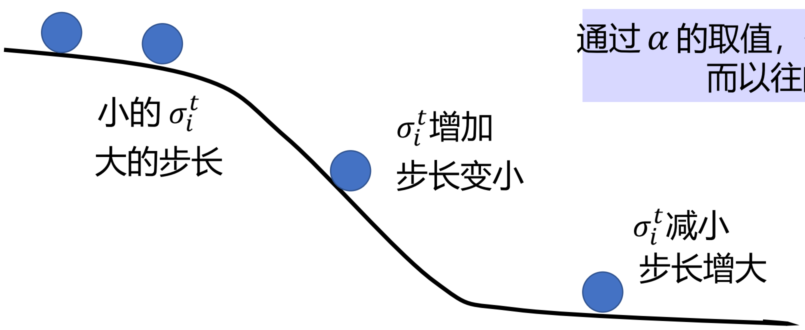
$$\frac{\partial L_{in}}{\partial w_{i,0}}, \frac{\partial L_{in}}{\partial w_{i,1}}, \dots, \frac{\partial L_{in}}{\partial w_{i,t-1}},$$

$$w_{i,t+1} \leftarrow w_{i,t} - \frac{\eta}{\sigma_{i,t}} \frac{\partial L_{in}}{\partial w_{i,t}}$$

$$\sigma_{i,t} = \sqrt{\alpha(\sigma_{i,t-1})^2 + (1-\alpha)(\frac{\partial L_{in}}{\partial w_{i,t}})^2}$$

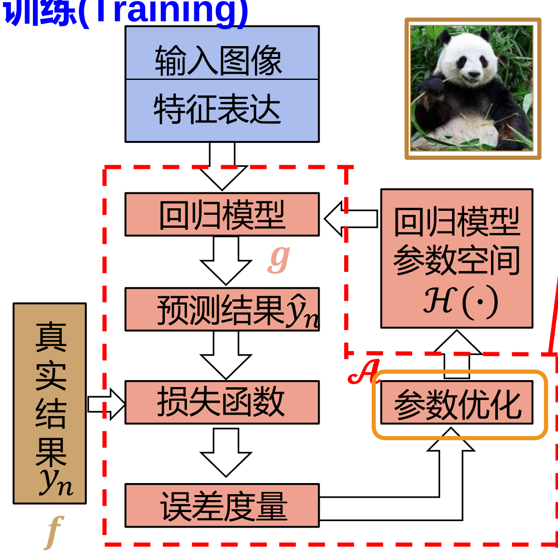
$$0 < \alpha < 1$$

通过 α 的取值，使得当前的梯度影响更大，而以往的梯度影响较小



3.3 梯度下降法

训练(Training)



梯度下降法：

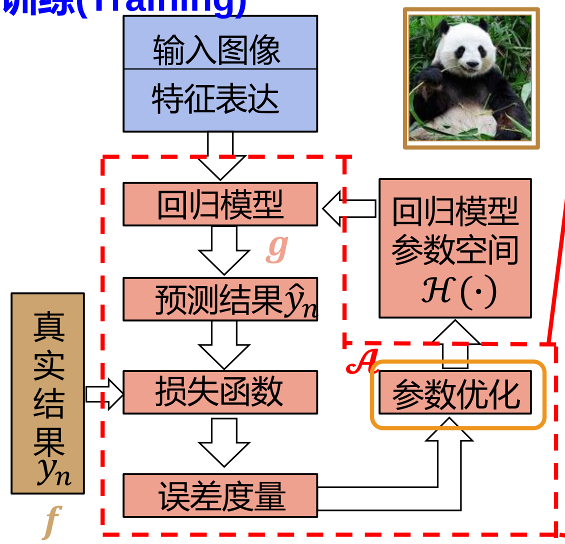
$$\nabla L_{in}(\mathbf{w}) = \sum_{n=1}^{N} (\mathbf{w}^T \mathbf{x}_n - y_n)\mathbf{x}_n$$

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \eta \nabla L_{in}(\mathbf{w}_t)$$

- *问题1：学习率η如何取值?*

3. 3 梯度下降法

训练(Training)



梯度下降法:

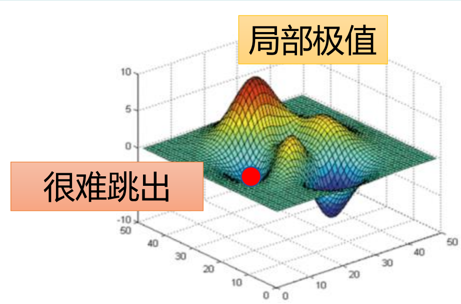
$$\nabla L_{in}(\mathbf{w}) = \sum_{n=1}^{N}(\mathbf{w}^T\mathbf{x}_n - y_n)\mathbf{x}_n$$

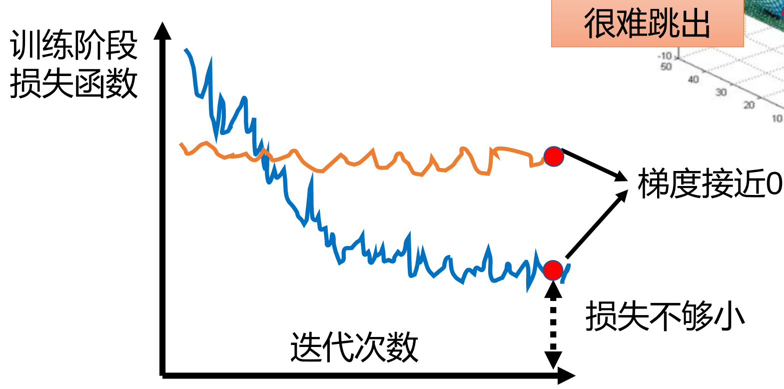
$$w_{i,t+1} \leftarrow w_{i,t} - \frac{\eta}{\sigma_{i,t}}\frac{\partial L_{in}}{\partial w_{i,t}}$$

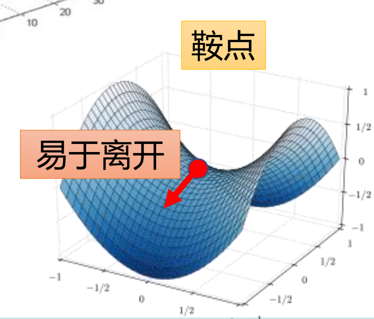
- *问题1：学习率η如何取值?*
- *问题2：梯度为0就能得到最佳解?*

3. 3 梯度下降法

梯度为0时没能得到最佳解

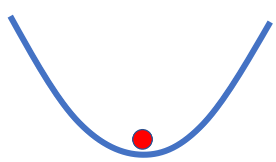


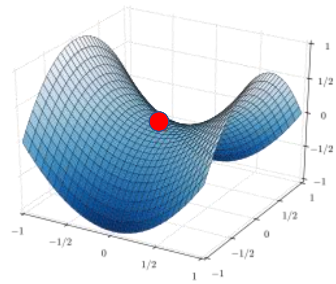


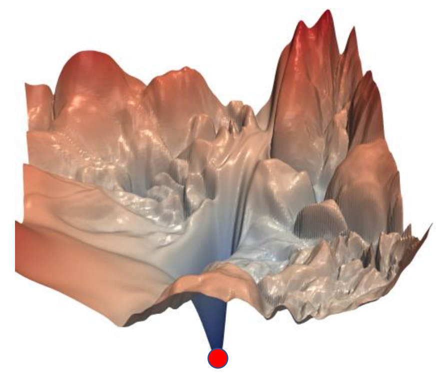


3. 3 梯度下降法

梯度为0时没能得到最佳解



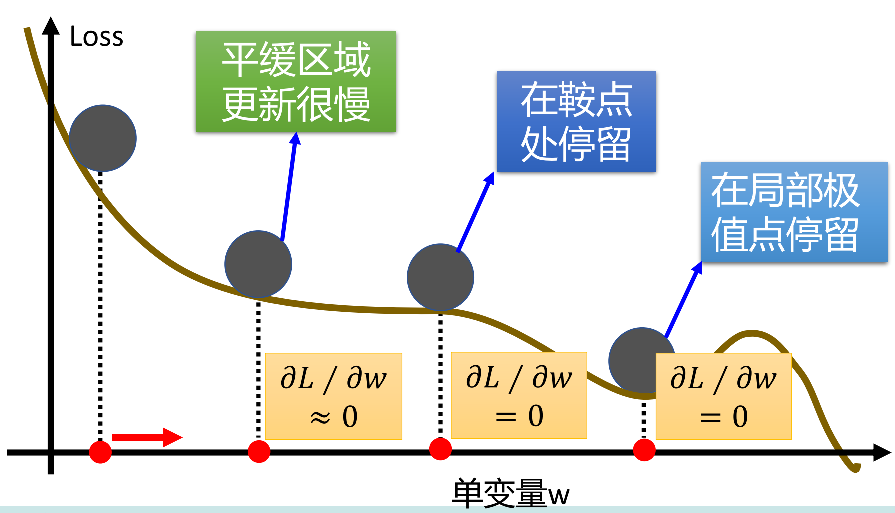




在更高维度空间上
它是鞍点吗?

3. 3 梯度下降法

梯度为0时没能得到最佳解

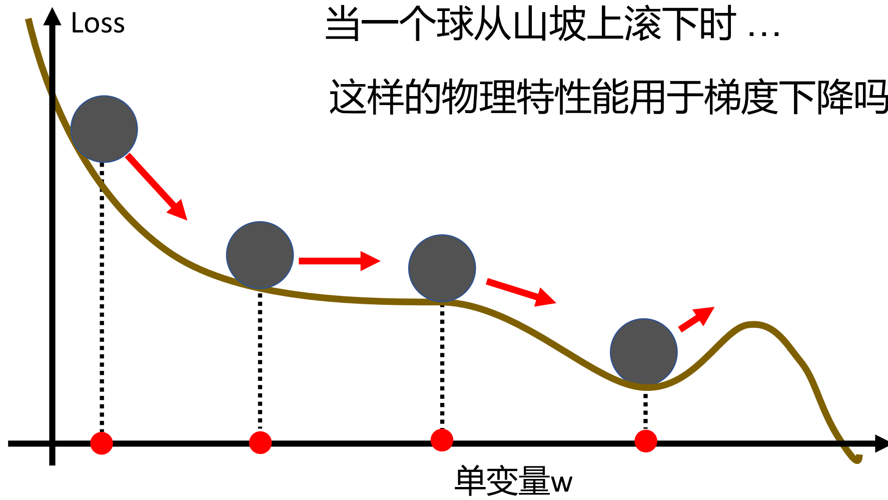


3. 3 梯度下降法

梯度较小时几乎停止更新合理吗?

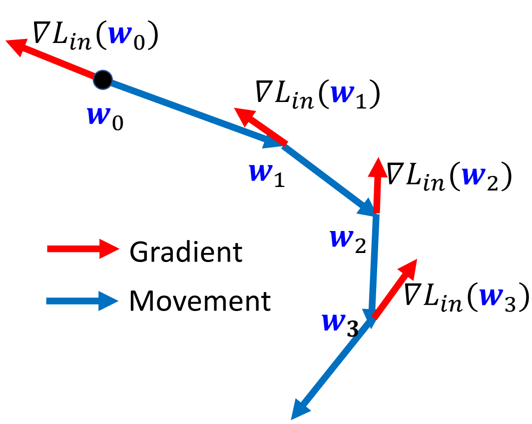
当一个球从山坡上滚下时 …

这样的物理特性能用于梯度下降吗?



3. 3 梯度下降法

用向量几何示意一般的梯度下降过程



初始值为$\boldsymbol{w}_0$

计算梯度：$\nabla L_{in}(\boldsymbol{w}_0)$

权向量更新：$\boldsymbol{w}_1 \leftarrow \boldsymbol{w}_0 - \eta \nabla L_{in}(\boldsymbol{w}_0)$

计算梯度：$\nabla L_{in}(\boldsymbol{w}_1)$

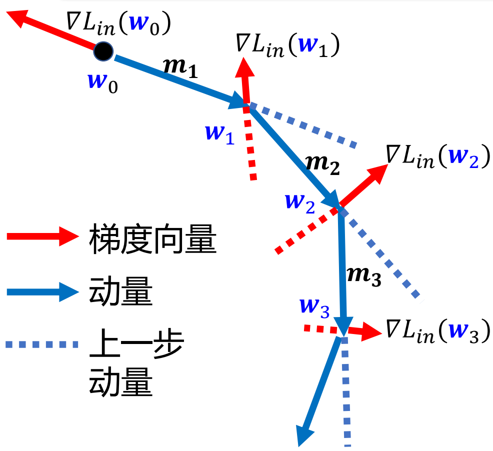
权向量更新：$\boldsymbol{w}_2 \leftarrow \boldsymbol{w}_1 - \eta \nabla L_{in}(\boldsymbol{w}_1)$

⋮

3.3 梯度下降法（Momentum）

结合动量特性的梯度下降法(*Gradient Descent + Momentum*)

动量:上一步的动量减去当前梯度



初始值为$\boldsymbol{w}_0$

动量 $\boldsymbol{m}_0 = \boldsymbol{0}$

计算梯度：$\nabla L_{in}(\boldsymbol{w}_0)$

动量 $\boldsymbol{m}_1 = \lambda \boldsymbol{m}_0 - \eta \nabla L_{in}(\boldsymbol{w}_0)$

权向量更新$\boldsymbol{w}_1 \leftarrow \boldsymbol{w}_0 + \boldsymbol{m}_1$

计算梯度：$\nabla L_{in}(\boldsymbol{w}_1)$

动量 $\boldsymbol{m}_2 = \lambda \boldsymbol{m}_1 - \eta \nabla L_{in}(\boldsymbol{w}_1)$

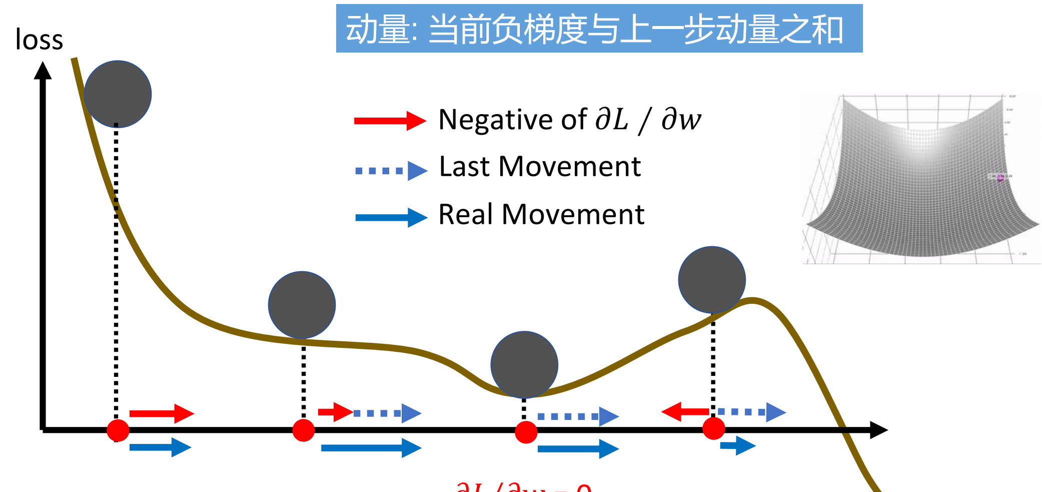
权向量更新$\boldsymbol{w}_2 \leftarrow \boldsymbol{w}_1 + \boldsymbol{m}_2$

权向量更新时不仅考虑负梯度方向
还同时要考虑上一步的动量

3.3 梯度下降法

结合动量特性的梯度下降法(*Gradient Descent + Momentum*)

动量: 当前负梯度与上一步动量之和



3.3 梯度下降法（Adam）

Adam: RMSProp + Momentum

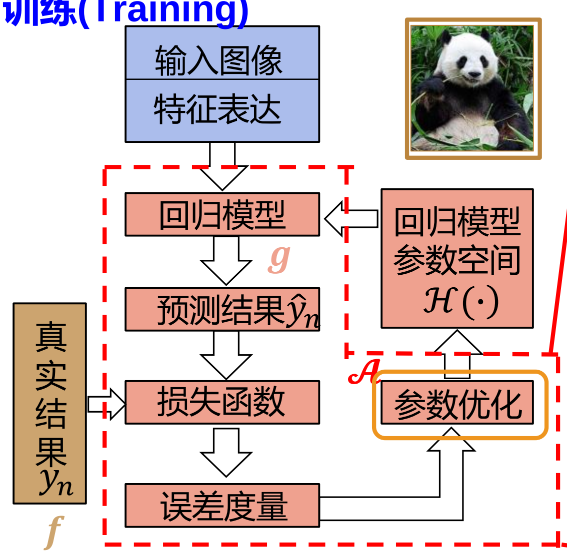
Algorithm 1: *Adam*, our proposed algorithm for stochastic optimization. See section 2 for details, and for a slightly more efficient (but less clear) order of computation. g_t^2 indicates the elementwise square $g_t \odot g_t$. Good default settings for the tested machine learning problems are $\alpha = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$. All operations on vectors are element-wise. With β_1^t and β_2^t we denote β_1 and β_2 to the power t.

Require: α: Stepsize
Require: $\beta_1, \beta_2 \in [0, 1)$: Exponential decay rates for the moment estimates
Require: $f(\theta)$: Stochastic objective function with parameters θ
Require: θ_0: Initial parameter vector
  $m_0 \leftarrow 0$ (Initialize 1st moment vector) → for momentum
  $v_0 \leftarrow 0$ (Initialize 2nd moment vector) → for RMSprop
  $t \leftarrow 0$ (Initialize timestep)
  while θ_t not converged **do**
    $t \leftarrow t + 1$
    $g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$ (Get gradients w.r.t. stochastic objective at timestep t)
    $m_t \leftarrow \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$ (Update biased first moment estimate)
    $v_t \leftarrow \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$ (Update biased second raw moment estimate)
    $\widehat{m}_t \leftarrow m_t / (1 - \beta_1^t)$ (Compute bias-corrected first moment estimate)
    $\widehat{v}_t \leftarrow v_t / (1 - \beta_2^t)$ (Compute bias-corrected second raw moment estimate)
    $\theta_t \leftarrow \theta_{t-1} - \alpha \cdot \widehat{m}_t / (\sqrt{\widehat{v}_t} + \epsilon)$ (Update parameters)
  end while
  return θ_t (Resulting parameters)

人工智能与自动化学院 53

3.3 梯度下降法

训练(Training)



梯度下降法:

$$\nabla L_{in}(\boldsymbol{w}) = \sum_{n=1}^{N} (\boldsymbol{w}^T \mathbf{x}_n - y_n)\mathbf{x}_n$$

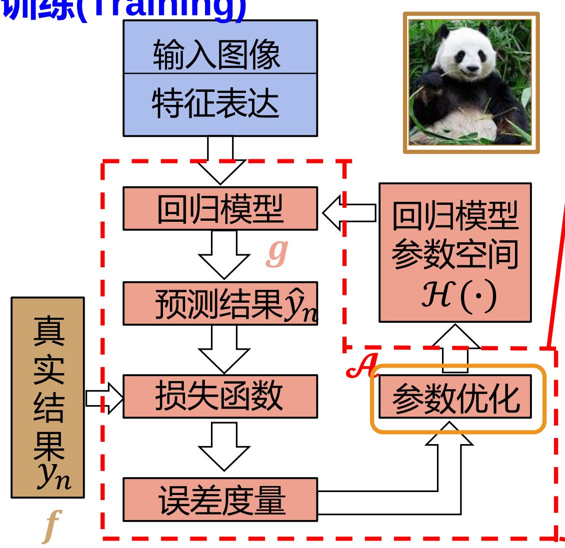
$$w_{i,t+1} \leftarrow w_{i,t} - \frac{\eta}{\sigma_{i,t}} \frac{\partial L_{in}}{\partial w_{i,t}}$$

- *问题1：学习率η如何取值?*
- *问题2：梯度为0就能得到最佳解?*

人工智能与自动化学院 54

3.3 梯度下降法

训练(Training)



梯度下降法:

$$\nabla L_{in}(\mathbf{w}) = \sum_{n=1}^{N} (\mathbf{w}^T \mathbf{x}_n - y_n)\mathbf{x}_n$$

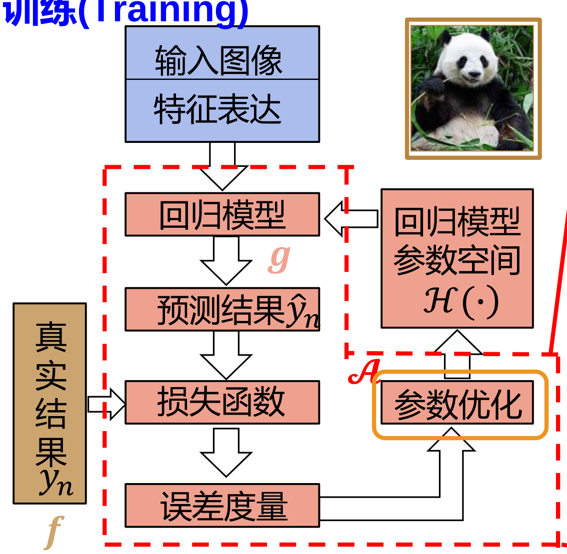
$$\boldsymbol{m}_{i,t+1} = \lambda \boldsymbol{m}_{i,t} - \frac{\eta}{\sigma_{i,t}} \frac{\partial L_{in}}{\partial w_{i,t}}$$

$$\boldsymbol{w}_{i,t+1} \leftarrow \boldsymbol{w}_{i,t} + \boldsymbol{m}_{i,t+1}$$

- *问题1：学习率η如何取值？*
- *问题2：梯度为0就能得到最佳解？*

3.3 梯度下降法

训练(Training)



梯度下降法:

$$\nabla L_{in}(\mathbf{w}) = \sum_{n=1}^{N} (\mathbf{w}^T \mathbf{x}_n - y_n)\mathbf{x}_n$$

$$\boldsymbol{m}_{i,t+1} = \lambda \boldsymbol{m}_{i,t} - \frac{\eta}{\sigma_{i,t}} \frac{\partial L_{in}}{\partial w_{i,t}}$$

$$\boldsymbol{w}_{i,t+1} \leftarrow \boldsymbol{w}_{i,t} + \boldsymbol{m}_{i,t+1}$$

- *问题1：学习率η如何取值？*
- *问题2：梯度为0就能得到最佳解？*
- *问题3：训练样本批量大小的影响？*

3.3 梯度下降法

梯度下降法实现线性回归

- 初始化权向量$\mathbf{w}_0$
- $for\ t=0,1,2,\ldots$ （t 代表迭代次数）

① 计算梯度：$\nabla L_{in}(\mathbf{w})=\boxed{\frac{2}{N}\sum_{n=1}^{N}(\mathbf{w}^T\mathbf{x}_n-y_n)\mathbf{x}_n}$

每次迭代N个样本均要计算，时间复杂度与Pocket算法相似

② 对权向量$\mathbf{w}_t$进行更新：$\mathbf{w}_{t+1}\leftarrow\mathbf{w}_t-\eta\nabla L_{in}(\mathbf{w}_t)$

…直到$\nabla L_{in}(\mathbf{w})=\mathbf{0}$，或者迭代足够多次数

返回最终的$\mathbf{w}_{t+1}$作为学到的g

3.3 梯度下降法

梯度下降法实现线性回归

- 初始化权向量$\mathbf{w}_0$
- $for\ t=0,1,2,\ldots$ （t 代表迭代次数）

随机梯度下降法
(Stochastic Gradient Descent)
(SGD)

① 计算梯度：$\nabla L_{in}(\mathbf{w})=(\mathbf{w}^T\mathbf{x}_n-y_n)\mathbf{x}_n$

② 对权向量$\mathbf{w}_t$进行更新：$\mathbf{w}_{t+1}\leftarrow\mathbf{w}_t-\eta\nabla L_{in}(\mathbf{w}_t)$

…直到$\nabla L_{in}(\mathbf{w})=\mathbf{0}$，或者迭代足够多次数

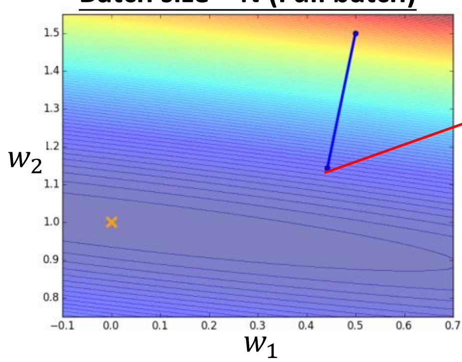
返回最终的$\mathbf{w}_{t+1}$作为学到的g

3.3 梯度下降法

迭代过程中一次计算样本多少(batch)对梯度下降的影响

假设一共有20个样本

Batch size = N (Full batch)

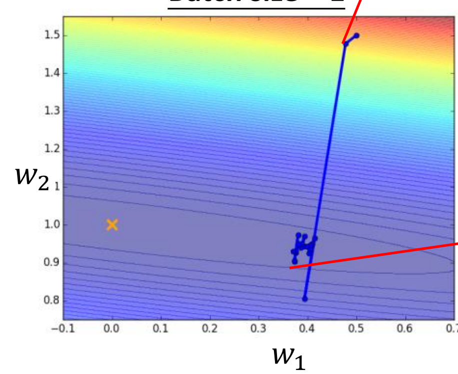


计算20个样本才更新

20个样本均参与计算后才更新

一个样本更新一次

Batch size = 1



20次更新以后

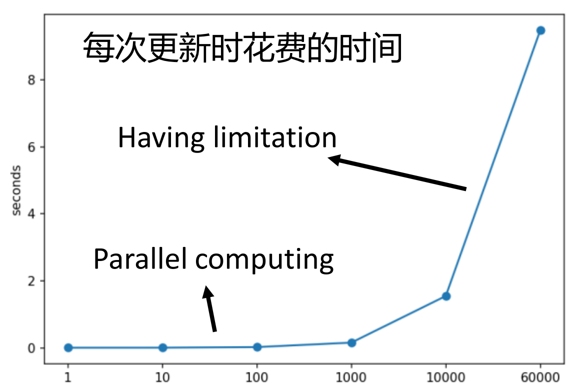
计算每个样本后就更新

3.3 梯度下降法

迭代过程中一次计算样本多少(batch)对梯度下降的影响

MNIST: digit classification

Tesla V100 GPU



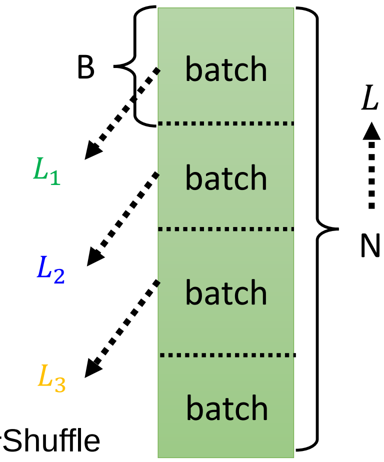
批量(batch size)大在计算梯度时并不意味着会花费更多的时间，除非batch size 太大

3. 3 梯度下降法

批量(batch)用于梯度下降

$$w^* = arg\min_{w} L$$

- 随机设置一个初始向量 w_0
- 计算梯度 $\nabla L_1(w_0)$，更新：$w_1 \leftarrow w_0 - \eta \nabla L_1(w_0)$
- 计算梯度 $\nabla L_2(w_1)$，更新：$w_2 \leftarrow w_1 - \eta \nabla L_2(w_1)$
- 计算梯度 $\nabla L_3(w_2)$，更新：$w_3 \leftarrow w_2 - \eta \nabla L_3(w_2)$

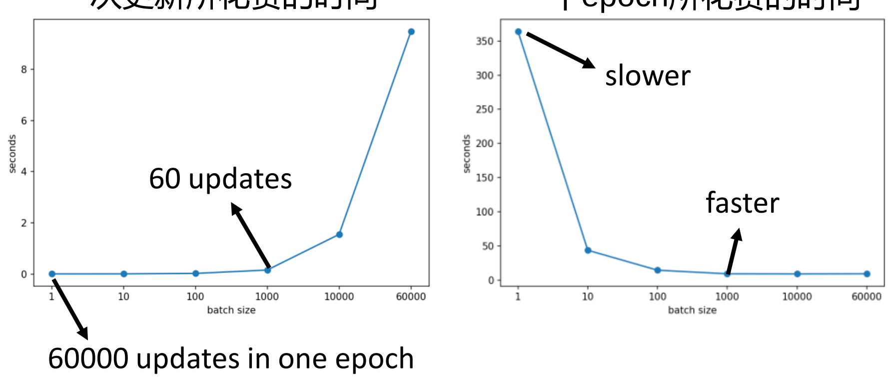


1 **epoch** = 所有的batch遍历一遍 ➜ 每一次epoch后进行Shuffle

3. 3 梯度下降法

批量(batch)大小对训练过程速度的影响

一次更新所花费的时间　　一个epoch所花费的时间

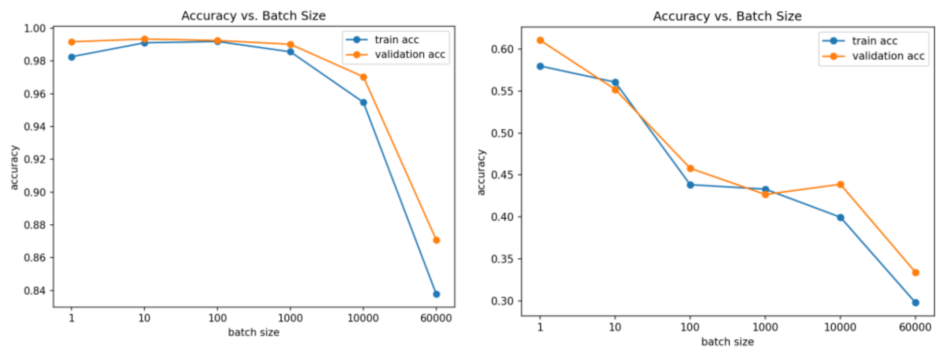


较小的批量(batch)做完一次epoch时需要花费更多的时间

3. 3 梯度下降法

批量(batch)大小对分类性能的影响

MNIST ***CIFAR-10***



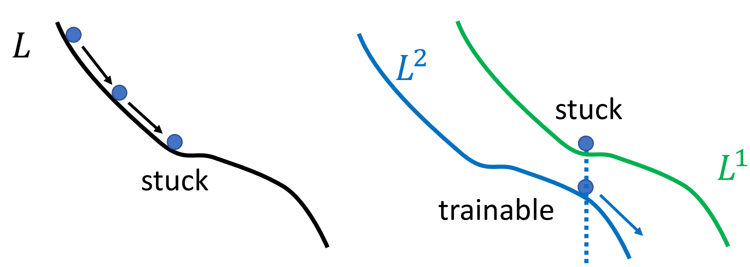
较小的批量获得更好的性能

为什么批量大了性能会下降呢?

3. 3 梯度下降法

批量(batch)大小对分类性能的影响

Full Batch ***Small Batch***



较小的批量获得更好的性能

为什么批量大了性能会下降呢?

3.3 梯度下降法

批量(batch)大小对分类性能的影响

SB = 256

LB = 0.1 x data set

Name	Network Type	Data set
F_1	Fully Connected	MNIST (LeCun et al., 1998a)
F_2	Fully Connected	TIMIT (Garofolo et al., 1993)
C_1	(Shallow) Convolutional	CIFAR-10 (Krizhevsky & Hinton, 2009)
C_2	(Deep) Convolutional	CIFAR-10
C_3	(Shallow) Convolutional	CIFAR-100 (Krizhevsky & Hinton, 2009)
C_4	(Deep) Convolutional	CIFAR-100

Name	Training Accuracy		Testing Accuracy	
	SB	LB	SB	LB
F_1	$99.66\% \pm 0.05\%$	$99.92\% \pm 0.01\%$	$98.03\% \pm 0.07\%$	$97.81\% \pm 0.07\%$
F_2	$99.99\% \pm 0.03\%$	$98.35\% \pm 2.08\%$	$64.02\% \pm 0.2\%$	$59.45\% \pm 1.05\%$
C_1	$99.89\% \pm 0.02\%$	$99.66\% \pm 0.2\%$	$80.04\% \pm 0.12\%$	$77.26\% \pm 0.42\%$
C_2	$99.99\% \pm 0.04\%$	$99.99\% \pm 0.01\%$	$89.24\% \pm 0.12\%$	$87.26\% \pm 0.07\%$
C_3	$99.56\% \pm 0.44\%$	$99.88\% \pm 0.30\%$	$49.58\% \pm 0.39\%$	$46.45\% \pm 0.43\%$
C_4	$99.10\% \pm 1.23\%$	$99.57\% \pm 1.84\%$	$63.08\% \pm 0.5\%$	$57.81\% \pm 0.17\%$

On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima

https://arxiv.org/abs/1609.04836

较小的批量在测试数据集上也能获得更好的性能

3.3 梯度下降法

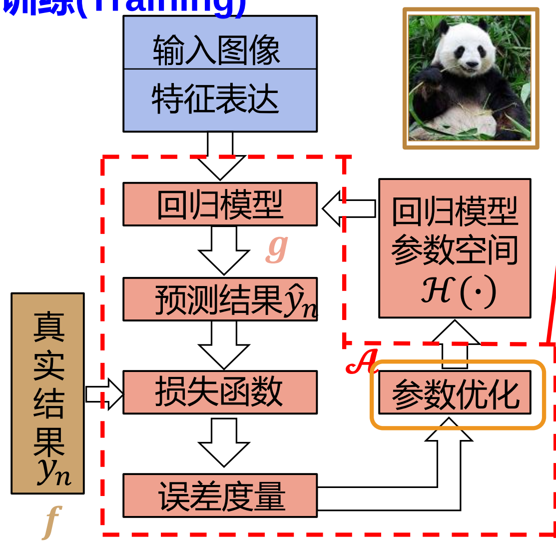
批量(batch)大小对梯度下降优化时的影响总结

	批量小	批量大
一次更新需要的速度(无并行处理)	Faster	Slower
一次更新需要的速度(有并行处理)	Same	Same (not too large)
一个epoch花费的时间	Slower	Faster WIN
梯度的特点	Noisy	Stable
优化性能	Better WIN	Worse
泛化性能	Better WIN	Worse

批量大小(batch size)作为超参数(hyperparameter)由算法设计者确定

3.3 梯度下降法

训练(Training)



梯度下降法:

$$\nabla L_{in}(\mathbf{w}) = \sum_{n=1}^{N} (\mathbf{w}^T \mathbf{x}_n - y_n)\mathbf{x}_n$$

$$\boldsymbol{m}_{i,t+1} = \lambda \boldsymbol{m}_{i,t} - \frac{\eta}{\sigma_{i,t}} \frac{\partial L_{in}}{\partial w_{i,t}}$$

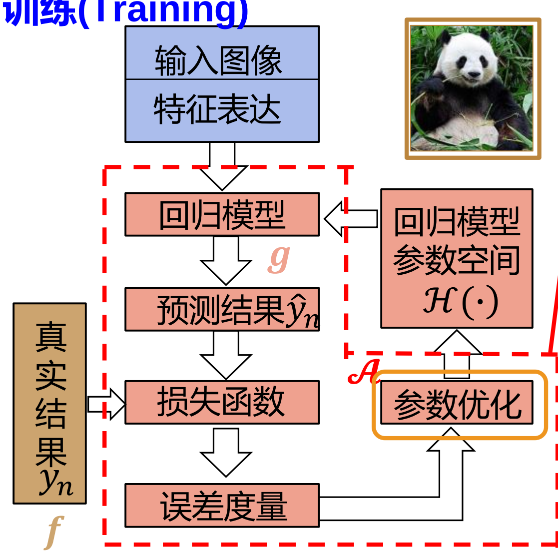
$$\boldsymbol{w}_{i,t+1} \leftarrow \boldsymbol{w}_{i,t} + \boldsymbol{m}_{i,t+1}$$

- *问题1：学习率η如何取值?*
- *问题2：梯度为0就能得到最佳解?*
- *问题3：训练样本批量大小的影响?*

人工智能与自动化学院 67

3.3 梯度下降法

训练(Training)



随机梯度下降法:

$$\nabla L_{in}(\mathbf{w}) = \sum_{n=1}^{B} (\mathbf{w}^T \mathbf{x}_n - y_n)\mathbf{x}_n$$

$$\boldsymbol{m}_{i,t+1} = \lambda \boldsymbol{m}_{i,t} - \frac{\eta}{\sigma_{i,t}} \frac{\partial L_{in}}{\partial w_{i,t}}$$

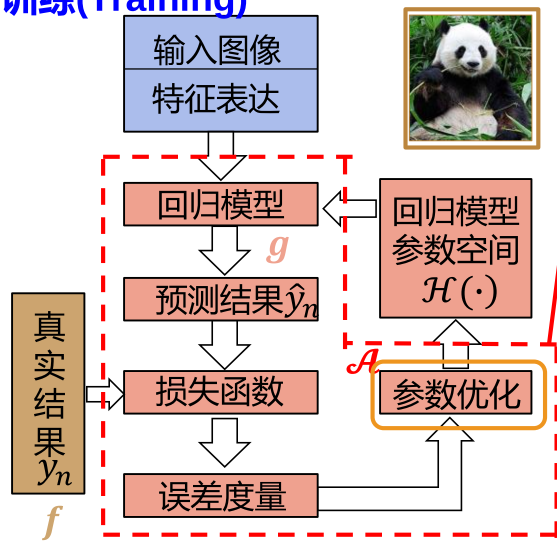
$$\boldsymbol{w}_{i,t+1} \leftarrow \boldsymbol{w}_{i,t} + \boldsymbol{m}_{i,t+1}$$

- *问题1：学习率η如何取值?*
- *问题2：梯度为0就能得到最佳解?*
- *问题3：训练样本批量大小的影响?*

人工智能与自动化学院 68

3.3 梯度下降法

训练(Training)



随机梯度下降法(SGD):

$$\nabla L_{in}(\mathbf{w}) = \sum_{n=1}^{B}(\mathbf{w}^T\mathbf{x}_n - y_n)\mathbf{x}_n$$

$\mathbf{m}_{i,t+1} = \lambda \mathbf{m}_{i,t}$ …

$\mathbf{w}_{i,t+1} \leftarrow \mathbf{w}$ …

- 问题1：学习率…
- 问题2：梯度为…佳解?
- 问题3：训练样本批量大小的影响?
- 问题4：损失函数的影响?

第五讲
再讨论

第三讲 线性回归 (*Linear Regression*)

3.1 线性回归问题

模型的输出为实数值，有众多应用场景

3.2 线性回归算法

损失函数为均方误差时，可通过求解广义逆得到解析解

3.3 梯度下降法

迭代优化，更一般的损失函数；固定学习率、AdaGrad、RMSProp 、动量(Momentum)、Adam、SGD、批量大小(batch size)