机器学习（本科生公选课）GEC6531

第8节 梯度下降 Gradient descent

计算机科学与技术学院

张瑞 教授

邮箱：ruizhang6@hust.edu.cn

签到 & 思考

■ 微助教签到（学校要求）

1. 加入课堂：微信扫码或者通过微助教公众号

课堂名称：GEC6531 机器学习（公选课）
课堂编号：OA628

1、扫码关注公众号：微助教服务号。

2、点击系统通知：“点击此处加入【GEC6531 机器学习（公选课）】课堂”，填写学生资料加入课堂。

*如未成功收到系统通知，请点击公众号下方“学生” - “全部(A)” - “加入课堂” --- “输入课堂编号”手动加入课堂

二维码有效期至：2024-11-16

2. 微信扫码签到

回顾线性回归和逻辑回归的梯度下降

今天的目录

- **基本算法**
  - 思路
  - 基本算法
  - 算法变体
- **理论**
  - 最小值
  - 凸函数和凹函数
  - 泰勒展开
  - 梯度下降核心思想
  - 梯度下降正确性
  - 步长
  - AdaGrad
- **牛顿法**
  - 核心思想
  - 例子

今天的目录

- **基本算法**
  - 思路
  - 基本算法
  - 算法变体
- **理论**
  - 最小值
  - 凸函数和凹函数
  - 泰勒展开
  - 梯度下降核心思想
  - 梯度下降正确性
  - 步长
  - AdaGrad
- **牛顿法**
  - 核心思想
  - 例子

损失函数 (Loss function / Cost function)

假设 Hypothesis: $h_\theta(x) = \theta_0 + \theta_1 x$

参数 Parameters: θ_0, θ_1

损失函数 Loss/Cost Function: $J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{n} \left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$

目标 Objective: $\underset{\theta_0, \theta_1}{\text{minimize}}\ J(\theta_0, \theta_1)$

L_p norm (*p* 范数) $\quad ||x_p|| = \left(\sum_{i=1}^{n} |x_i|^p\right)^{1/p} \quad where\ p > 0$

梯度下降预览 (Gradient Descent)

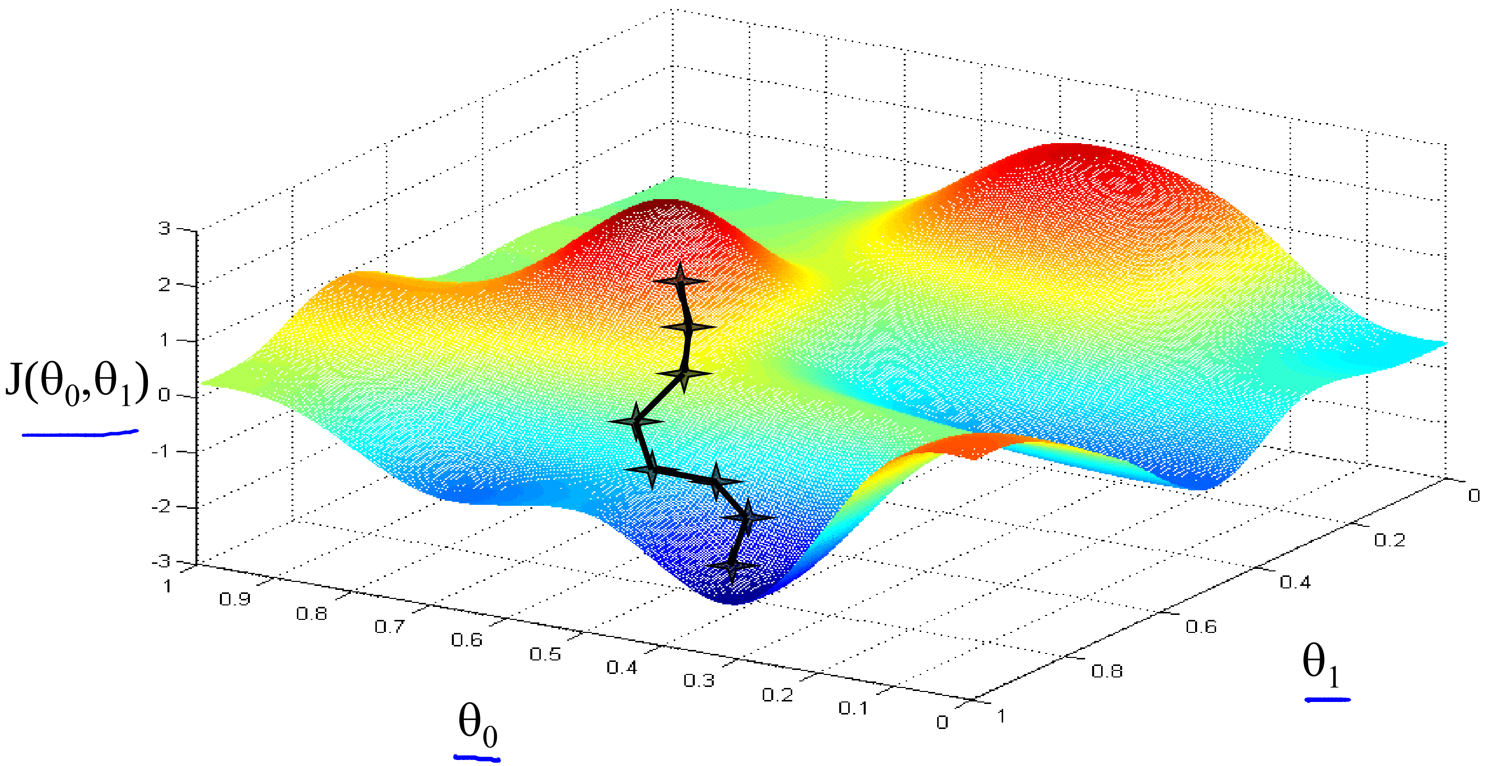
损失函数 $J(\theta_0, \theta_1)$

目标 $\min_{\theta_0, \theta_1} J(\theta_0, \theta_1)$

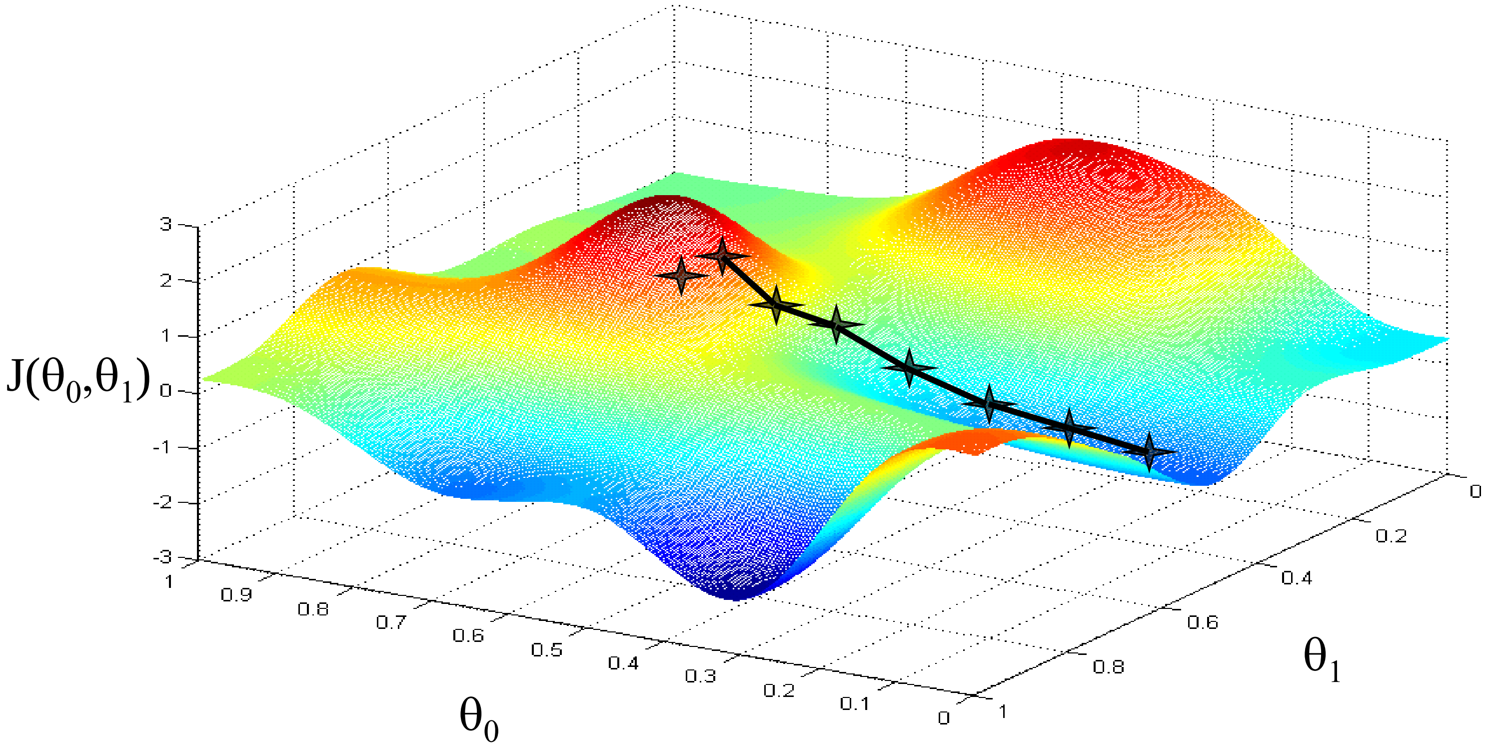
概览:

- 以某个值开始 θ_0, θ_1
- 不停更新 θ_0, θ_1 来减小 $J(\theta_0, \theta_1)$
- 直到我们认为达到了最小值

梯度下降预览 (Gradient Descent)



梯度下降预览 (Gradient Descent)



梯度定义

- 函数 J 在 θ 这点的梯度定义为 $\left[\frac{\partial J}{\partial \theta_1}, \dots, \frac{\partial J}{\partial \theta_m}\right]'$，简写为 ∇J, ∇ 称为 Nabla 算符
- 梯度是一个向量，指向 $J(\theta)$ 离开 θ 时改变最快的方向
- 对两个变量的梯度，海森矩阵 (Hessian matrix): $\nabla J_{ij} = \frac{\partial^2 J}{\partial \theta_i \partial \theta_j}$

梯度下降基本算法（Gradient descent algorithm）

$$
\begin{aligned}
&\text{repeat until convergence } \{ \\
&\qquad \theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1) \qquad (\text{for } j = 0 \text{ and } j = 1) \\
&\}
\end{aligned}
$$

Correct: Simultaneous update

$$
\begin{aligned}
&\text{temp0} := \theta_0 - \alpha \tfrac{\partial}{\partial \theta_0} J(\theta_0, \theta_1) \\
&\text{temp1} := \theta_1 - \alpha \tfrac{\partial}{\partial \theta_1} J(\theta_0, \theta_1) \\
&\theta_0 := \text{temp0} \\
&\theta_1 := \text{temp1}
\end{aligned}
$$

Incorrect:

$$
\begin{aligned}
&\text{temp0} := \theta_0 - \alpha \tfrac{\partial}{\partial \theta_0} J(\theta_0, \theta_1) \\
&\theta_0 := \text{temp0} \\
&\text{temp1} := \theta_1 - \alpha \tfrac{\partial}{\partial \theta_1} J(\theta_0, \theta_1) \\
&\theta_1 := \text{temp1}
\end{aligned}
$$

线性回归的梯度下降

$$\frac{\partial}{\partial\theta_j}J(\theta_0,\theta_1) = \frac{\partial}{\partial\theta_j}\frac{1}{2m}\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$$

$$= \frac{\partial}{\partial\theta_j}\frac{1}{2m}\sum_{i=1}^{m}\left(\theta_0 + \theta_1 x^{(i)} - y^{(i)}\right)^2$$

$$j = 0 : \frac{\partial}{\partial\theta_0}J(\theta_0,\theta_1) = \frac{1}{m}\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right)$$

$$j = 1 : \frac{\partial}{\partial\theta_1}J(\theta_0,\theta_1) = \frac{1}{m}\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right) \cdot x^{(i)}$$

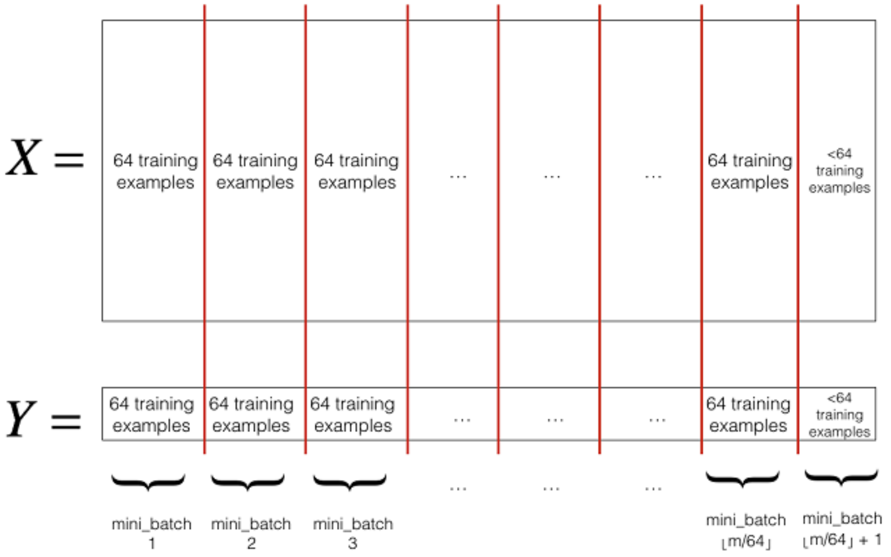
线性回归的梯度下降

$$
\begin{aligned}
&\text{repeat until convergence } \{ \\
&\quad \theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right) \\
&\quad \theta_1 := \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right) \cdot x^{(i)} \\
&\}
\end{aligned}
$$

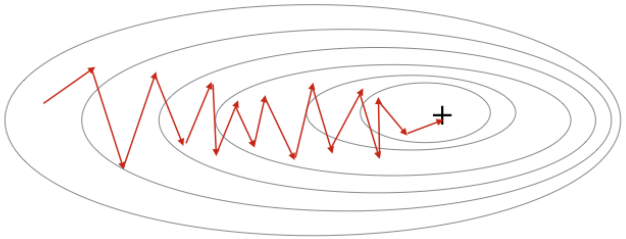
update θ_0 and θ_1 simultaneously

- 批梯度下降（Batch gradient descent)
- 随机梯度下降（stochastic gradient descent） （SGD）
- 小批随机梯度下降（mini-batch gradient descent）

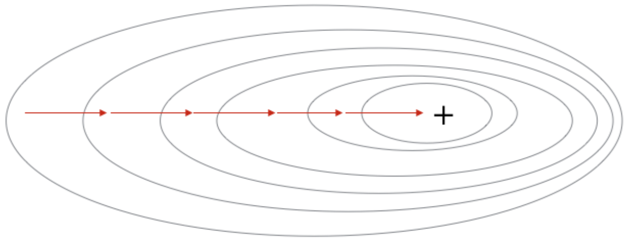
mini-batch gradient descent



Stochastic Gradient Descent



Gradient Descent



梯度下降算法变体

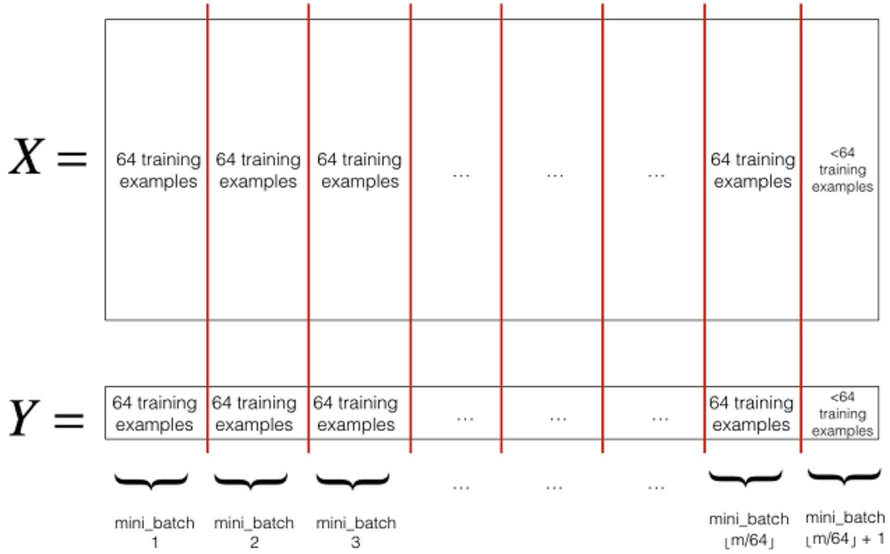
1. Choose $\boldsymbol{\theta}^{(1)}$ and some T
2. For i from 1 to T*
   1. $\boldsymbol{\theta}^{(i+1)} = \boldsymbol{\theta}^{(i)} - \alpha \nabla J(\boldsymbol{\theta}^{(i)})$
3. Return $\hat{\boldsymbol{\theta}} \approx \boldsymbol{\theta}^{(i)}$

可能使用其他的停止条件，例如 θ 变化小于某个阈值

Stochastic gradient descent: two loops

* Outer for loop: each loop (called epoch) sweeps through all training data
* Within each epoch, randomly shuffle training data; then for loop: do gradient steps only on batches of data. Batch size might be 1 or few

- α 可能在不同的循环里变化
- 使用训练数据的策略
  - 批梯度下降（Batch gradient descent）
  - 随机梯度下降（stochastic gradient descent, SGD）
  - 小批随机梯度下降（mini-batch SGD）
- 算法变体：Momentum, AdaGrad, **Adam**



逻辑回归梯度下降 (Gradient Descent)

$$J(\theta) = -\sum_{i=1}^{m}(y^{(i)} \log h_\theta(x^{(i)}) + (1 - y^{(i)}) \log(1 - h_\theta(y^{(i)})))$$

$$\downarrow$$

偏导数:

$$\frac{\partial J(\theta)}{\partial \theta_j} = \sum_i x_j^{(i)}(h_\theta(x^{(i)}) - y^{(i)})$$

$$\downarrow$$

权重更新:

$$\theta_j = \theta_j - \alpha \frac{\partial J(\theta)}{\partial \theta_j}$$

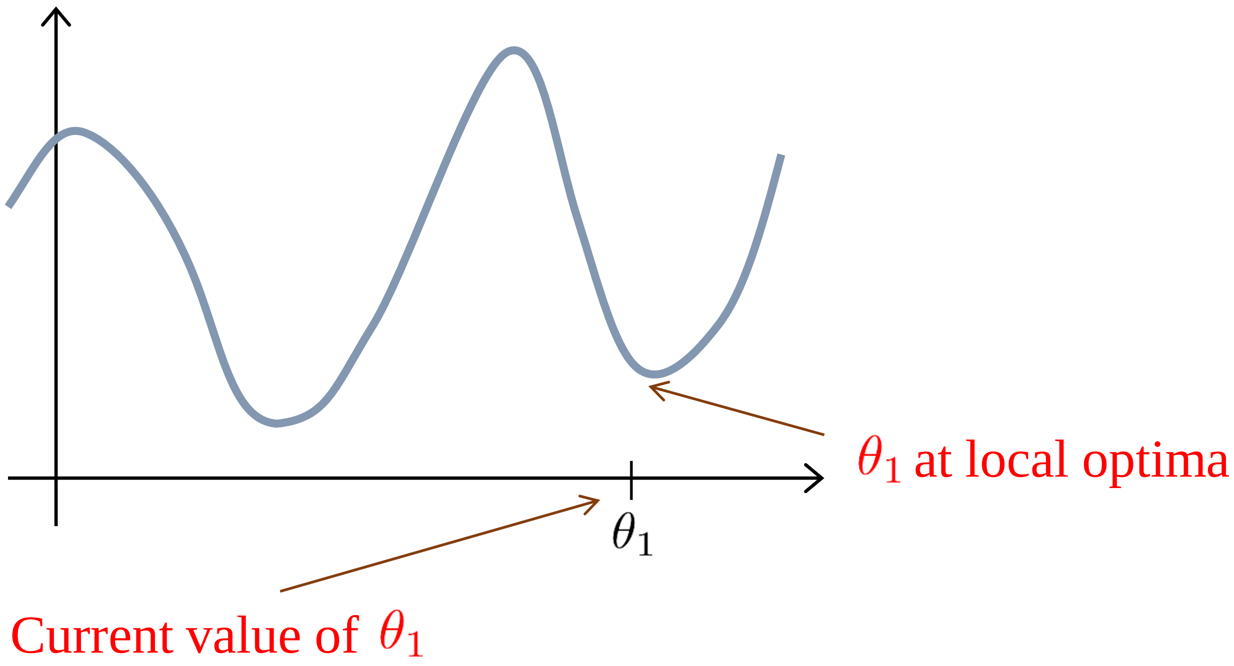
$$\frac{\partial J(\theta)}{\partial \theta_j}$$

$$\theta_j := \theta_j - \alpha \boxed{\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)} \cdot \frac{1}{m}}$$

今天的目录

- **基本算法**
  - 思路
  - 基本算法
  - 算法变体
- **理论**
  - 最小值
  - 凸函数和凹函数
  - 泰勒展开
  - 梯度下降核心思想
  - 梯度下降正确性
  - 步长
  - AdaGrad
- **牛顿法**
  - 核心思想
  - 例子

局部最小值（Local minima)



凸的连续可微函数的最小化

我们想最小化一个**凸的连续可微**的损失函数 $\ell(w)$。

- ℓ 是凸的。这使得所找到的任何局部最小值也是全局最小值，且有助于简化对牛顿法的讨论。
- ℓ 至少是三次连续可微的。我们将使用泰勒展开近似。这个假设极大地简化了讨论。
- 对 w 没有约束。增加对 w 的约束会增加讨论的复杂性，我们对此不展开讨论。

本节将讨论两种被广泛使用的“爬坡”算法，梯度下降法和牛顿法 $\min_w \ell(w)$。

什么是（局部）最小值

- 问题：求解 $\min_w \ell(w)$ 实际上意味着什么。
- 我们称 w^* 为 ℓ 的局部最小值，如果满足：

局部最小值：

存在 $\epsilon > 0$ 对于 $\{w \mid \|w - w^*\|_2 < \epsilon\}$ 满足 $\ell(w^*) \le \ell(w)$.

- 我们之前假设 ℓ 是凸的，这意味着若找到这样一个 w^*，则对于所有 $w \in \mathbb{R}^d$ 均有 $\ell(w^*) \le \ell(w)$。
- 也可以通过 $\ell(w^*) < \ell(w)$ 来定义一个严格的局部最小值。
- 注意，一些凸函数没有严格的局部最小值，例如常数函数 $\ell(w) = 1$ 是凸函数。
- 一些凸函数没有局部最小值，例如，对于任意非零向量 $c \in \mathbb{R}^d$，$c^T w$ 是凸的，但可以使其任意小。
- 一个点是局部最小点的关键必要条件是 ℓ 在 w^* 处的梯度为 0，即 $\nabla \ell(w^*) = 0$。
- 假设函数的梯度在 w^* 处为 0，该点为严格局部最小的充分条件为其海森矩阵 $\nabla^2 \ell(w^*)$ 是正定的。

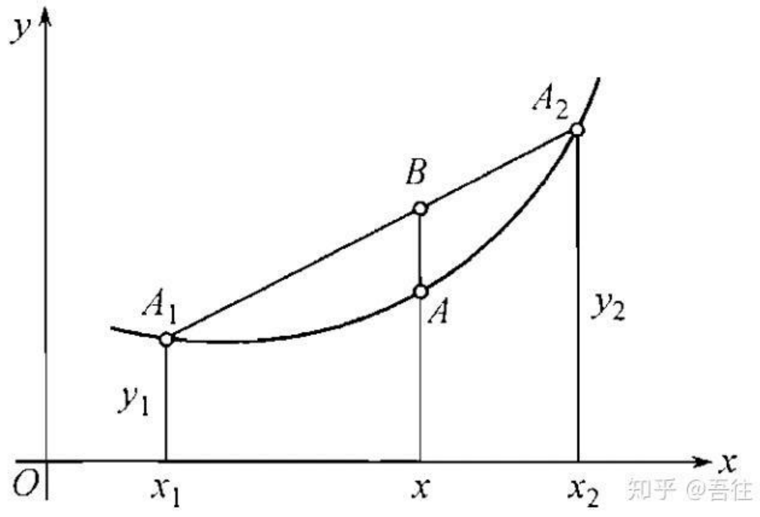
凸函数（Convex) 和凹函数（Concave)

■ 凸函数：

在区间 $\mathcal{X}$ 上[1] 有定义而且连续的函数 $f(x)$ 叫做是凸函数(向下凸)，如果对 $\mathcal{X}$ 中的任何两点 x_1 与 $x_2(x_1 \lessgtr x_2)$ 有不等式

$$f(q_1x_1+q_2x_2) \leqslant q_1f(x_1)+q_2f(x_2), \tag{1}$$

凸函数如果有严格的局部最小值，这个严格局部最小值就是全局最小值

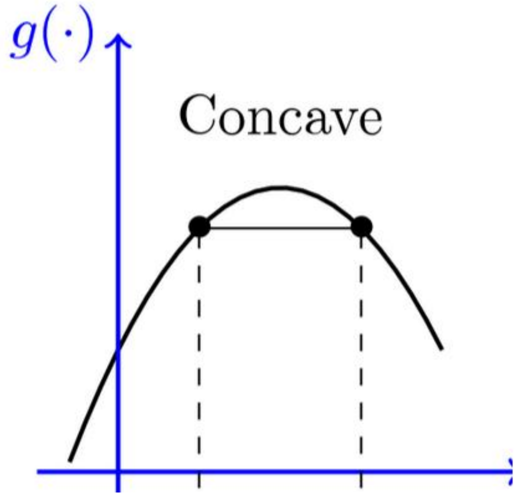


■ 凹函数

而 q_1 与 q_2 为相加等于 1 的任何正数. 函数叫做凹函数，如果成立的不是 (1) 而是

$$f(q_1x_1+q_2x_2) \geqslant q_1f(x_1)+q_2f(x_2).^{②} \tag{1a}$$

凹函数如果有严格的局部最大值，这个严格局部最大值就是全局最大值



泰勒展开

一阶泰勒展开

以 w 为中心的一阶泰勒展开可以写为:

$$\ell(w+s) \approx \ell(w) + s^T g(w),$$

其中 $g(w)$ 是 ℓ 在 w 处的梯度，即 $(g(w))_j = \frac{\partial \ell}{\partial w_j}(w)$，对于 $j = 1, \dots, d$.

二阶泰勒展开

以 w 为中心的二阶泰勒展开可以写为:

$$\ell(w+s) \approx \ell(w) + s^T g(w) + \frac{1}{2} s^T H(w) s,$$

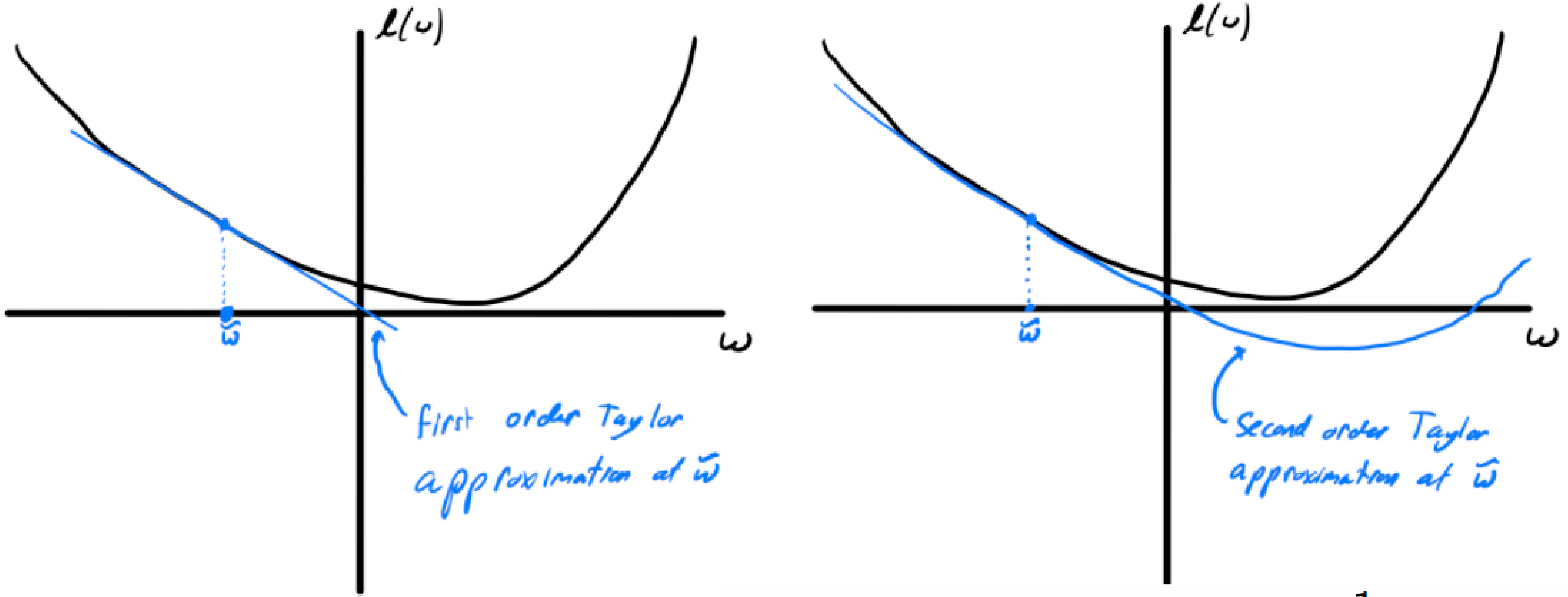
其中 $H(w)$ 是 ℓ 在 w 处的 Hessian 矩阵，即:

$$[H(w)]_{i,j} = \frac{\partial^2 \ell}{\partial w_i \partial w_j}(w),$$

对于 $j = 1, \dots, d$.

一阶和二阶泰勒展开

- 这些对应 ℓ 的线性近似和二次近似。
- 如果 s 很小，那么这些近似是合理有效的 (一阶误差为 $\mathcal{O}(\|s\|_2^2)$，二阶误差为 $\mathcal{O}(\|s\|_2^3)$)。



$$\ell(w+p) \approx \ell(w) + g(w)^T p \qquad \ell(w+p) \approx \ell(w) + g(w)^T p + \frac{1}{2} p^T H(w) p$$

梯度下降法

核心思想

考虑当前在这一点上，函数值下降最快的方向并朝该方向迈出一步。

考虑到展开点的线性逼近可以利用泰勒级数得到

$$\ell(w^k + s) = \ell(w^k) + s^T g(w^k)$$

那么下降最快的方向可以表示为 $s \propto -g(w^k)$。
我们在梯度下降中将 s 设为

$$s = -\alpha g(w^k)$$

其中设置步长 $\alpha > 0$。

梯度下降法：正确性

正确性

总有一些足够小的 α

$$\ell(w^k - \alpha g(w^k)) < \ell(w^k).$$

为什么？

$$\ell(w^k + s) = \ell(w^k) + g(w^k)^T s$$

$$s = -\alpha g(w^k)$$

$$\alpha > 0$$

梯度下降法：正确性

正确性

总能找到足够小的 α，使得

$$\ell(w^k - \alpha g(w^k)) < \ell(w^k).$$

$$\ell(w^k - \alpha g(w^k)) = \ell(w^k) - \alpha g(w^k)^T g(w^k) + \mathcal{O}(\alpha^2).$$

因为

$$g(w^k)^T g(w^k) > 0$$

并且当 $\alpha \to 0$，$\alpha^2 \to 0$ 收敛的比 α 更快.
因此我们可以得出结论，对于一个足够小的 $\alpha > 0$ 我们有 $\ell(w^k - \alpha g(w^k)) < \ell(w^k)$.

决定步长

- 在经典优化中，α 通常被称为步长 (在这种情况下 $g(w^k)$ 是搜索方向)。
- 然而，设置 α 大小固定的策略可能会产生更大的开销。问题在于将 α 设置得太小会导致收敛缓慢，而将 α 设置得太大会导致发散。

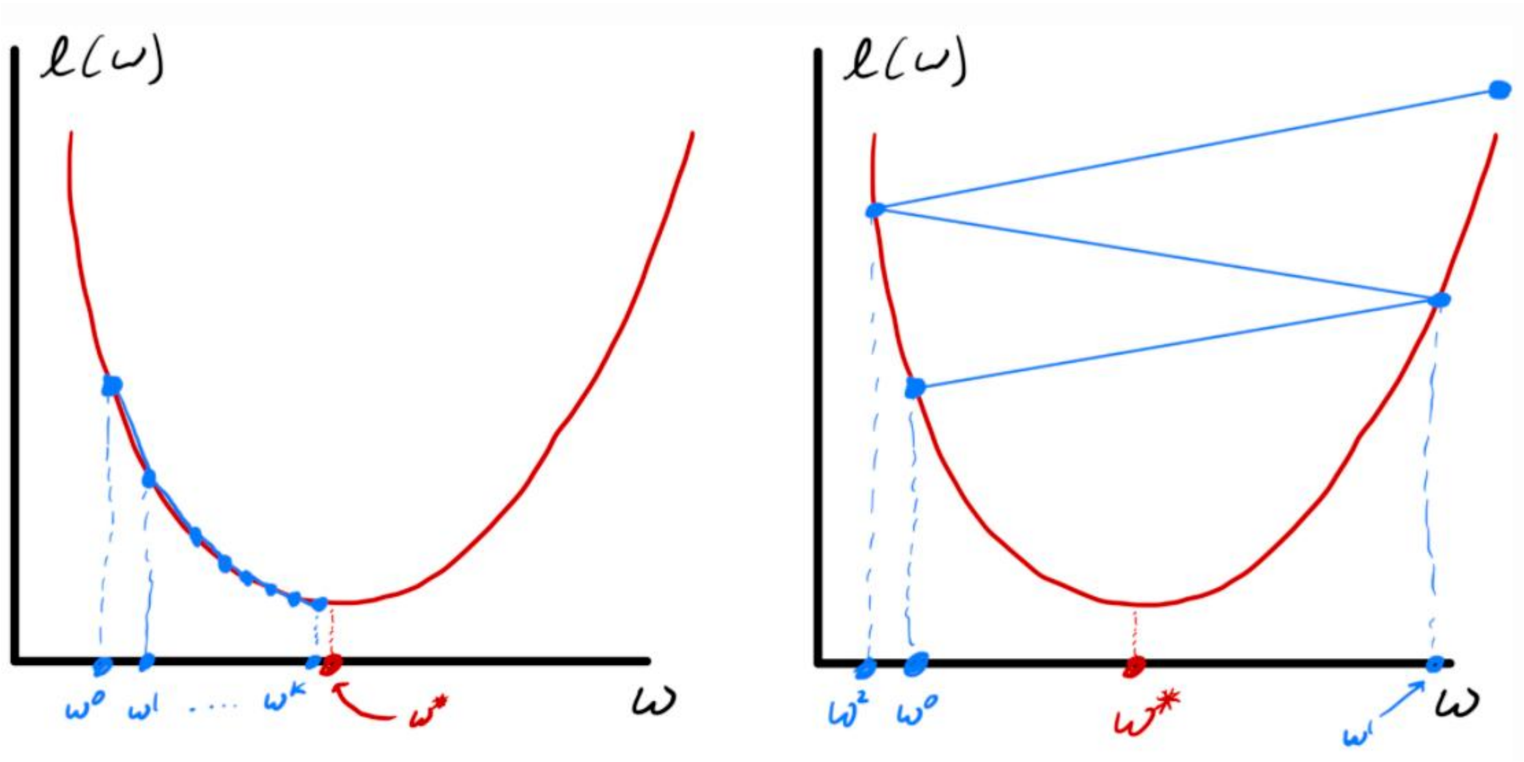


图: 步长选择导致收敛(左) 或发散(右)

AdaGrad

- 一种选择是为每个特征自适应地设置步长。
- Adagrad 通过运行每个优化变量平方梯度的平均值来实现这一点。
- 然后，它为梯度大的变量设置一个小的学习率，为梯度小的变量设置一个大的学习率。
- 如果 w 的项添加到特征 (例如在逻辑回归中，我们可以将 w 的每项与一个特征关联起来)，而这些特征在范围或频率上是不同的，那么这一点就很重要。

AdaGrad

Input: ℓ, $\nabla\ell$, parameter $\epsilon > 0$, and initial learning rate α.

Set $w_j^0 = 0$ and $z_j = 0$ for $j = 1, \ldots, d$. k = 0;

While not converged:

> 1. Compute entries of the gradient $g_j = \frac{\partial \ell}{\partial w_j}(w^k)$
> 2. $z_j = z_j + g_j^2$ for $j = 1, \ldots, d$.
> 3. $w_j^{k+1} = w_j^k - \alpha \frac{g_j}{\sqrt{z_j + \epsilon}}$ for $j = 1, \ldots, d$.
> 4. $k = k + 1$
> 5. Check for convergence; if converged set $\hat{w} = w^k$

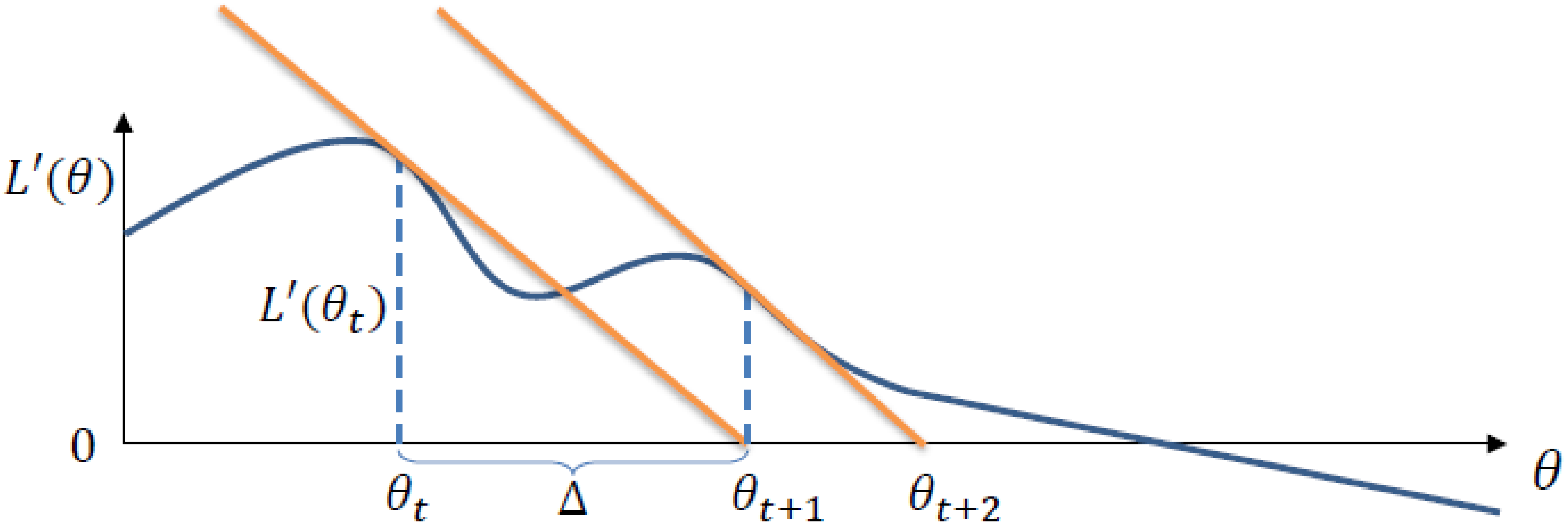
Return: $\hat{w}$

关键点：每一个维度都使用自己的学习率
问题：为什么加入 ϵ?

今天的目录

- **基本算法**
  - 思路
  - 基本算法
  - 算法变体
- **理论**
  - 最小值
  - 凸函数和凹函数
  - 泰勒展开
  - 梯度下降核心思想
  - 梯度下降正确性
  - 步长
  - AdaGrad
- **牛顿法**
  - 核心思想
  - 例子

牛顿方法



牛顿方法

核心思想

使用二阶信息 (二次近似)。

$$\ell(w^k + s) \approx \ell(w^k) + s^T g(w^k) + \frac{1}{2} s^T H(w^k) s.$$

- 我们选择一步 s，在 w^k 处显式地最小化 ℓ 的二次近似。
- 回想一下，因为 ℓ 是凸函数，对于所有 w，$H(w)$ 都是正半定的，所以这是一个明智的尝试。
- 事实上，牛顿的方法在严格的局部最小值附近具有非常好的性质，一旦足够接近一个解，它就会迅速收敛。

$$H(\mathbf{w}) = \begin{pmatrix} \frac{\partial^2 \ell}{\partial w_1^2} & \frac{\partial^2 \ell}{\partial w_1 \partial w_2} & \cdots & \frac{\partial^2 \ell}{\partial w_1 \partial w_n} \\ \vdots & \cdots & \cdots & \vdots \\ \frac{\partial^2 \ell}{\partial w_n \partial w_1} & \cdots & \cdots & \frac{\partial^2 \ell}{\partial w_n^2} \end{pmatrix},$$

牛顿方法

核心思想

使用二阶信息 (二次近似):

$$\ell(w^k + s) \approx \ell(w^k) + s^T g(w^k) + \frac{1}{2} s^T H(w^k) s.$$

- 为了简单起见，我们假设 $H(w^k)$ 是正定的。
- 我们的二次近似梯度是 $g(w^k) + H(w^k)s$。
- 这意味着线性系统 s 可以按照如下值设置:

$$g(w) + H(w)s = 0 \tag{1}$$

$$\Rightarrow s = -(H(w))^{-1} g(w). \tag{2}$$

参数更新:

$$w_{t+1} = w_t - (H(w_t))^{-1} g(w_t).$$

一个简单的例子

- 有一个简单的例子清楚地说明了二阶信息是如何起作用的。
- 假设函数是一个严格的凸二次函数，即

$$\ell(w) = \frac{1}{2}w^T A w + b^T w + c$$

其中 A 是一个正定矩阵，b 是一个向量，c 是一个数值。

问题：牛顿收敛多少步？

一个简单的例子

- 假设函数实际上是一个严格的凸二次函数，即，

$$\ell(w) = \frac{1}{2}w^T A w + b^T w + c$$

其中 A 是一个正定矩阵，b 是一个任意向量，c 是某个数字。

在这种情况下，牛顿法一步收敛 (因为 w^* 是 $Aw = b$ 的严格全局最小的唯一解)。

一个简单的例子

- 假设函数实际上是一个严格的凸二次函数，即，

$$\ell(w) = \frac{1}{2}w^T Aw + b^T w + c$$

其中 A 是一个正定矩阵，b 是一个任意向量，c 是某个数字。

同时，梯度下降得到迭代序列

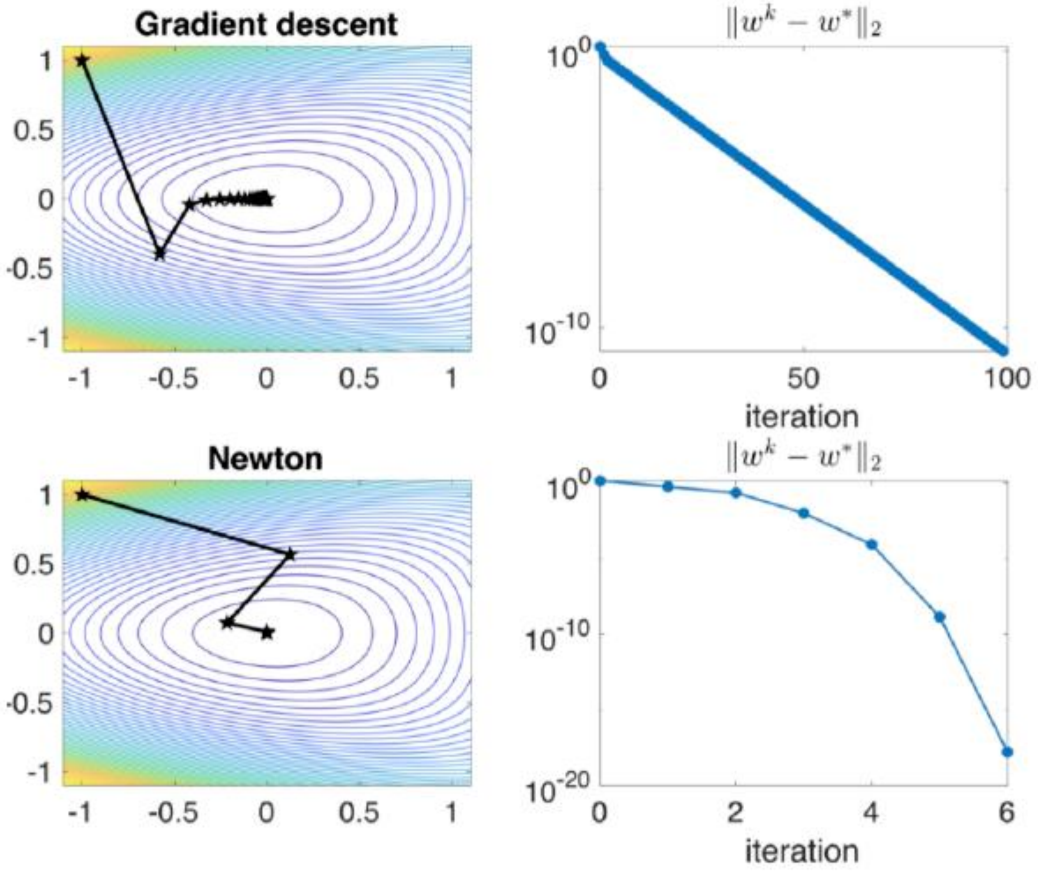
$$w^k = (I - \alpha A)w^{k-1} - \alpha b.$$

利用 $w^* = (I - \alpha A)w^* - \alpha b$ 我们可以得到

$$\begin{aligned} \|w^k - w^*\| &\le \|I - \alpha A\|_2 \|w^{k-1} - w^*\|_2 &(3) \\ &\le \|I - \alpha A\|_2^k \|w^0 - w^*\|_2. &(4) \end{aligned}$$

一个简单的例子

- 因此，只要 α 足够小，那么 $I-\alpha A$ 的所有特征值都在 $(-1,1)$，迭代就会收敛——但如果我们有接近 ± 1 的特征值，那么收敛速度就会很慢，。
- 更一般地，如下图所示，当接近局部最小时，牛顿法会加速收敛。



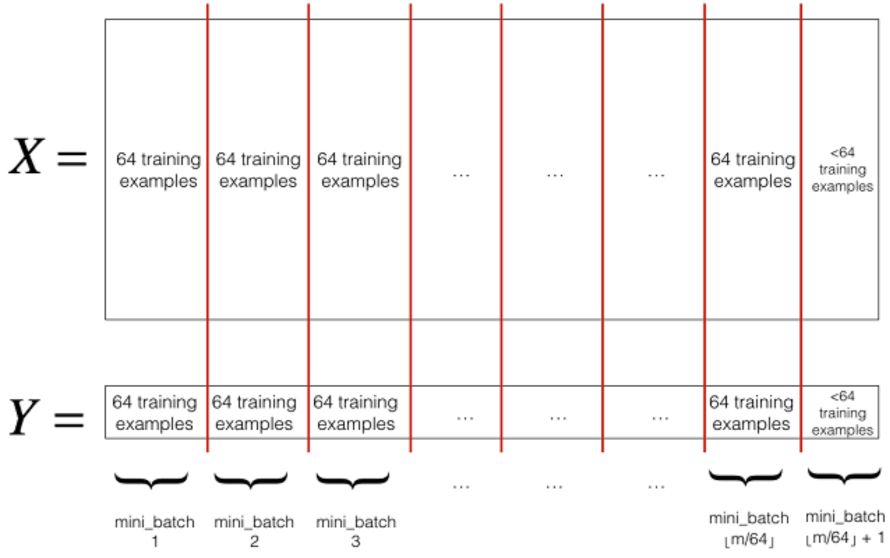
总结

1. Choose $\boldsymbol{\theta}^{(1)}$ and some T
2. For i from 1 to T*
    1. $\boldsymbol{\theta}^{(i+1)} = \boldsymbol{\theta}^{(i)} - \alpha \nabla J(\boldsymbol{\theta}^{(i)})$
3. Return $\widehat{\boldsymbol{\theta}} \approx \boldsymbol{\theta}^{(i)}$

可能使用其他的停止条件，例如 θ 变化小于某个阈值

Stochastic gradient descent: two loops

- Outer for loop: each loop (called epoch) sweeps through all training data
- Within each epoch, randomly shuffle training data; then for loop: do gradient steps only on batches of data. Batch size might be 1 or few



- α 可能在不同的循环里变化
- 使用训练数据的策略
    - 批梯度下降（Batch gradient descent)
    - 随机梯度下降（stochastic gradient descent, SGD）
    - 小批随机梯度下降（mini-batch SGD）
- 算法变体：Momentum, AdaGrad, **Adam**

总结

线性回归

$$
\begin{aligned}
&\text{repeat until convergence } \{ \\
&\quad \theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right) \\
&\quad \theta_1 := \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right) \cdot x^{(i)} \\
&\}
\end{aligned}
$$

逻辑回归

$$\theta_j := \theta_j - \alpha \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)} \cdot \frac{1}{m}$$

步长选择： 自适应的方法 AdaGrad