线性回归

线性回归模型

线性回归

线性回归问题的目标是用一个超平面来拟合离散的数据

假设数据集为：$\mathcal{D} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \cdots, (\mathbf{x}_N, y_N)\}$

这里： $\mathbf{x}_1 = (x_{i0}, x_{i1}, \ldots, x_{id})^T$， $x_{i0} = 1$

记

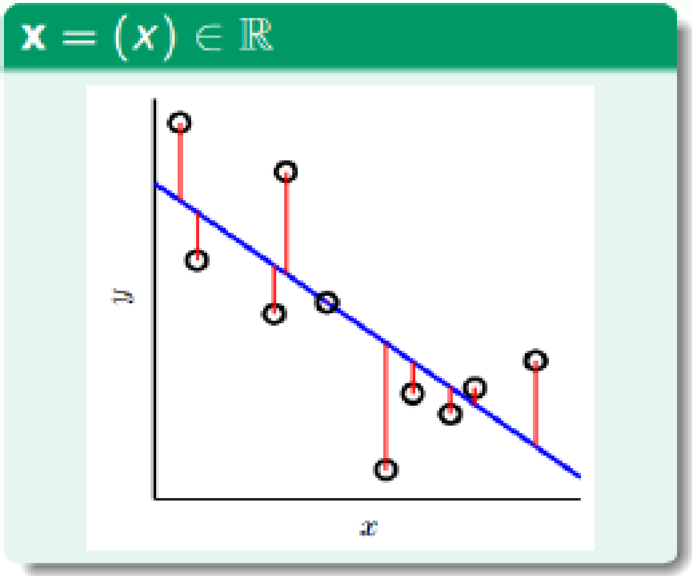
$$X = (\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N)^T, Y = (y_1, y_2, \cdots, y_N)^T$$

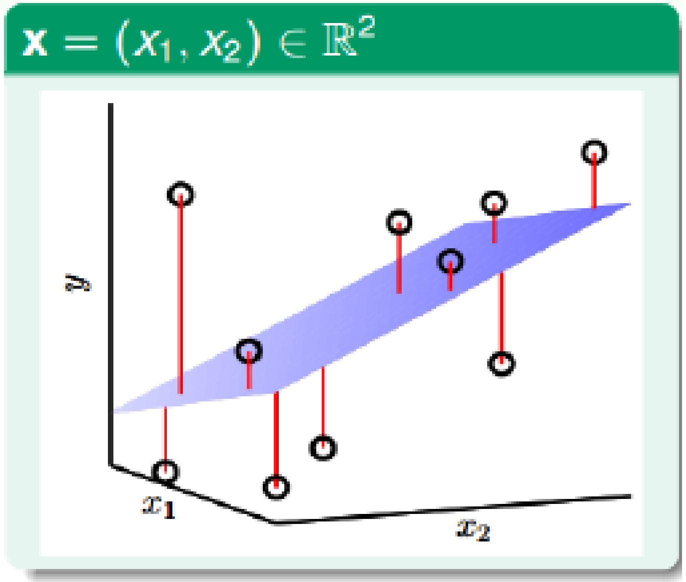
$$X = \begin{pmatrix} 1 & x_{11} & \cdots & x_{1d} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{N1} & \cdots & x_{Nd} \end{pmatrix}_{N\times(d+1)}$$

线性回归

设 $\omega = (\omega_0, \omega_1, \ldots, \omega_d)^T$

线性回归模型：$f(\mathbf{x}) = \omega^T \mathbf{x} = \omega_0 + \omega_1 x_1 + \ldots + \omega_d x_d = \sum_{i=0}^{d} \omega_i x_i$





损失函数：

$$L(\omega) = \sum_{i=1}^{N} (\omega^T \mathbf{x}_i - y_i)^2$$

线性回归：预测国内生产总值增长率

国内生成总值是一个国家在一年内生产的所有商品和服务的价值。经济学家们通常对决定一个国家GDP增长率的因素（例如，失业率、教育水平、人口总数、陆地面积、收入等级、投资率、平均寿命等）感兴趣、目的是制定更好的金融政策。为了理解一个国家的不同特征是如何与GDP增长率产生关联的，经济学家们通常使用线性回归方法。

线性回归：非线性拟合

通过为原始数据实行特征转换，设计出卓越的新特征，使得在新的特征上，可以提供好的线性拟合。

设： $\mathbf{x} = (x_1, x_2, \ldots, x_d)$

特征转换： $g : \mathbf{R}^d \to \mathbf{R}^s$

考虑： $\mathbf{z} = g(\mathbf{x})$，我们有

$$f(\mathbf{x}) = \omega_0 + \omega^T \mathbf{g}(\mathbf{x}) = \omega_0 + \omega_1 z_1 + \ldots + \omega_s z_s$$

这样在对原始特征来说，提供了一个好的非线性拟合

线性回归算法：最小二乘法

线性回归算法：最小二乘法

最小二乘法

$$
\begin{aligned}
L(\omega) &= (w^T\mathbf{x}_1 - y_1, \cdots, w^T\mathbf{x}_N - y_N) \cdot (w^T\mathbf{x}_1 - y_1, \cdots, w^T\mathbf{x}_N - y_N)^T \\
&= \|X\omega - Y\|_2^2 \\
&= (w^T X^T - Y^T)(Xw - Y) \\
&= (w^T X^T X w - 2w^T X^T Y + Y^T Y)
\end{aligned}
$$

计算最小化这个值的 $\hat{w}$：

$$
\begin{aligned}
\hat{w} = \underset{w}{argmin}\, L(w) &\longrightarrow \frac{\partial}{\partial w} L(w) = 0 \\
&\longrightarrow X^T X \hat{w} - X^T Y = 0 \\
&\longrightarrow \hat{w} = (X^T X)^{-1} X^T Y
\end{aligned}
$$

线性回归算法：最小二乘法

预测：$\hat{y} = \hat{x}^T(X^TX)^{-1}X^TY$

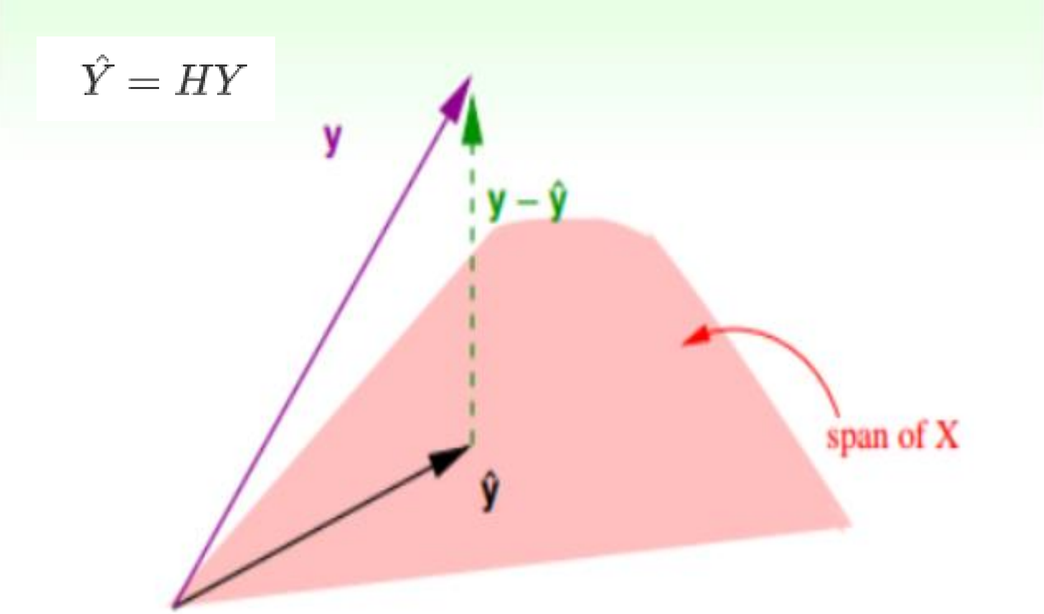
拟合：$\hat{Y} = X(X^TX)^{-1}X^TY$

定义投影算子：$H = X(X^TX)^{-1}X^T$

伪逆：$X^\dagger = (X^TX)^{-1}X^T$

奇异值分解：$X = U\Sigma V^T$

则：$X^\dagger = V\Sigma^{-1}U^T$



线性回归算法：梯度下降法

线性回归算法：梯度下降法

梯度下降法（GD）
1847年由著名的数学家柯西Cauchy给出。梯度下降是最小化风险函数、损失函数的一种常用方法。在应用机器学习算法时，通常采用梯度下降法对采用的算法进行训练

基本思想

- 假设我们爬山，如果想最快的上到山顶，那么我们应该从山势最陡的地方上山。也就是山势变化最快的地方上山
- 同样，如果从任意一点出发，需要最快搜索到函数最大值，那么我们也应该从函数变化最快的方向搜索。
- 函数变化最快的方向是什么呢？函数的梯度：$\nabla J(\theta)$

如果函数为一元函数，梯度就是该函数的导数

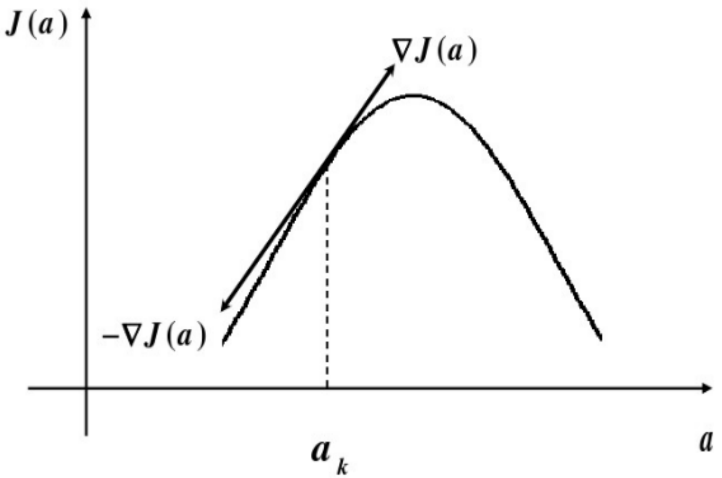
$$\nabla J(\theta) = J'(\theta)$$

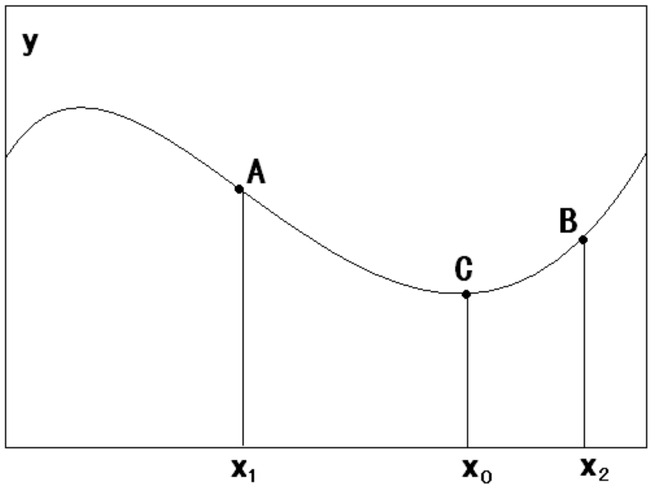
线性回归算法：梯度下降法

如果为n元函数，梯度定义为： $\nabla J(\vec{\theta}) = \left(\frac{\partial J(\vec{\theta})}{\partial \theta_1}, \frac{\partial J(\vec{\theta})}{\partial \theta_2}, \ldots, \frac{\partial J(\vec{\theta})}{\partial \theta_n}\right)$

如果需要找的是函数极小点，那么应该从负梯度的方向寻找，该方法称之为梯度下降法。

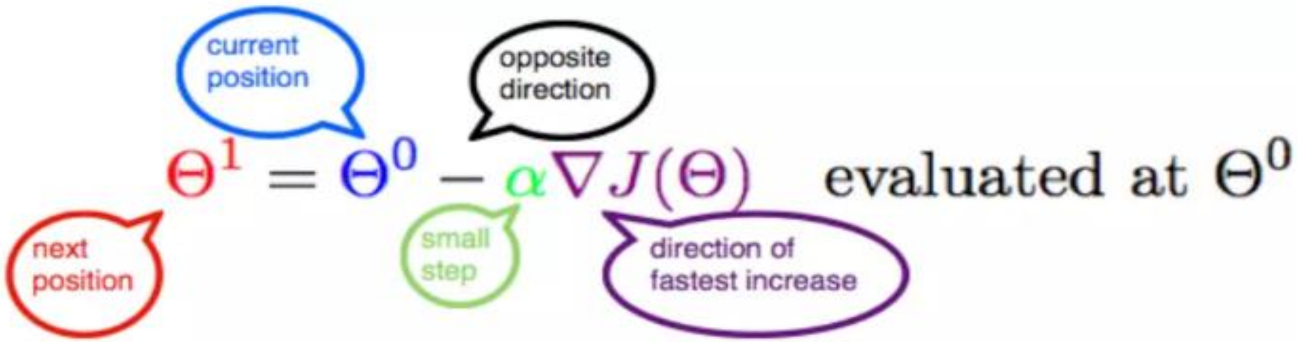
要搜索极小值C点，在A点必须向x增加方向搜索，此时与A点梯度方向相反；在B点必须向x减小方向搜索，此时与B点梯度方向相反。总之，搜索极小值，必须向负梯度方向搜索。



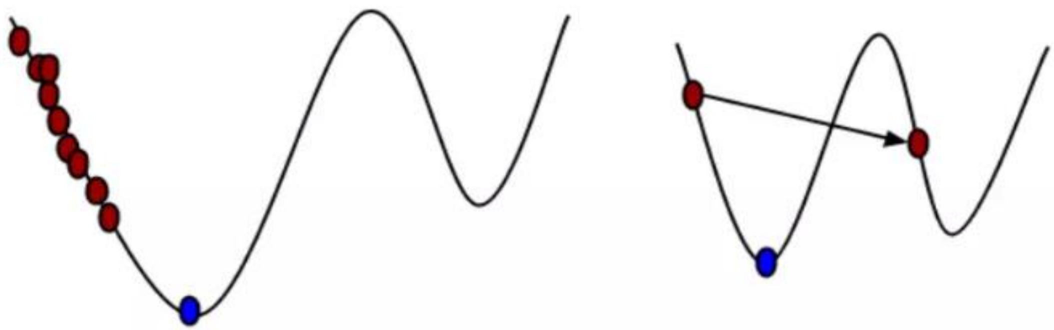


线性回归算法：梯度下降法

梯度下降法迭代思路：



步长大小问题：



very small learning rate needs lots of steps

too big learning rate: missed the minimum

梯度下降法：步骤

假设函数 $J(\theta_1, \theta_2, \ldots, \theta_n)$ 只有一个极小点。
初始给定自变量为 $\vec{\theta}_0 = (\theta_{10}, \theta_{20}, \ldots, \theta_{n0})$ 。从这个点如何搜索才能找到原函数的极小值点?

方法：

1. 首先设定一个较小的正数 η，ϵ 以及 $t = 0$
2. 求当前位置处的各个偏导数：

$$\nabla_m J(\vec{\theta}_t) = \frac{\partial J(\vec{\theta})}{\partial \theta_m}(\theta_{1t}, \theta_{2t}, \ldots, \theta_{nt}), \quad m = 1, 2, \ldots, n$$

3. 修改当前函数的变量值，公式如下：

$$\theta_m = \theta_m - \eta \nabla_m J(\vec{\theta}_t), \qquad m = 1, 2, \ldots, n$$

$$t = t + 1, \qquad \vec{\theta}_t = (\theta_1, \theta_2, \ldots, \theta_n)$$

4. 如果变化量小于 ϵ，退出；否则返回2。

梯度下降法

例 任给一个初始出发点，设为x_0=-4，利用梯度下降法求函数$y=x^2/2-2x$的极小值。

(1) 首先给定两个参数： $\eta=0.9, \varepsilon=0.01$

(2) 计算导数： $\dfrac{dy}{dx}=x-2$

(3) 计算当前导数值： $y'=-6$

(4) 修改当前参数：

$$x'=x-\eta\frac{dy}{dx}=-4-0.9*(-6)=1.4$$

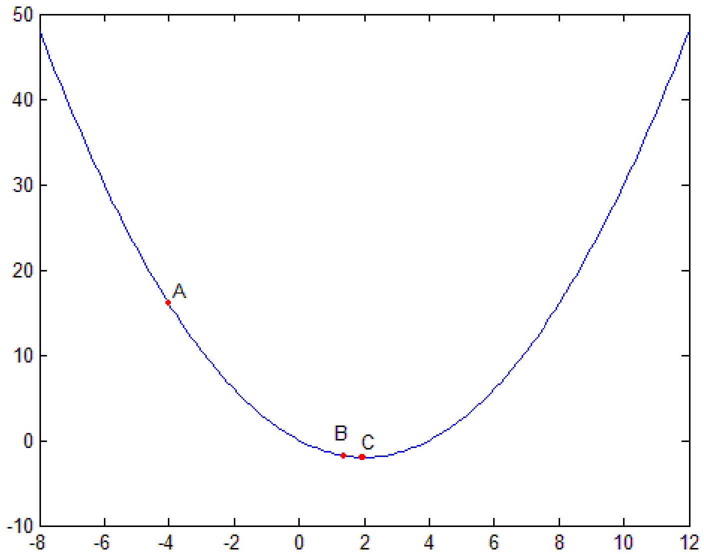
$$\Delta x=-0.9*(-6)=5.4$$

(5) 计算当前导数值： $y'=-0.6$

(6) 修改当前参数：

$$x'=x-\eta\frac{dy}{dx}=1.4-0.9*(-0.6)=1.94$$

$$\Delta x=-0.9*(-0.6)=0.54$$



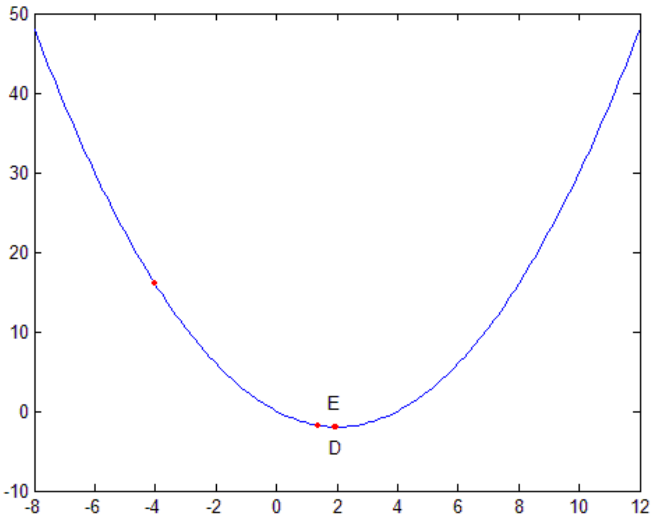
梯度下降法

(7) 计算当前导数值： $y' = -0.06$

(8) 修改当前参数：

$$x' = x - \eta \frac{dy}{dx} = 1.94 - 0.9*(-0.06) = 1.994$$

$$\Delta x = -0.9*(-0.06) = 0.054$$



(9) 计算当前导数值： $y' = -0.006$

(10) 修改当前参数：

$$x' = x - \eta \frac{dy}{dx} = 1.994 - 0.9*(-0.006) = 1.9994$$

$$\Delta x = -0.9*(-0.006) = 0.0054 < \varepsilon$$

(11)此时变化量满足终止条件，终止。

梯度下降法

梯度下降法包含三种不同形式：

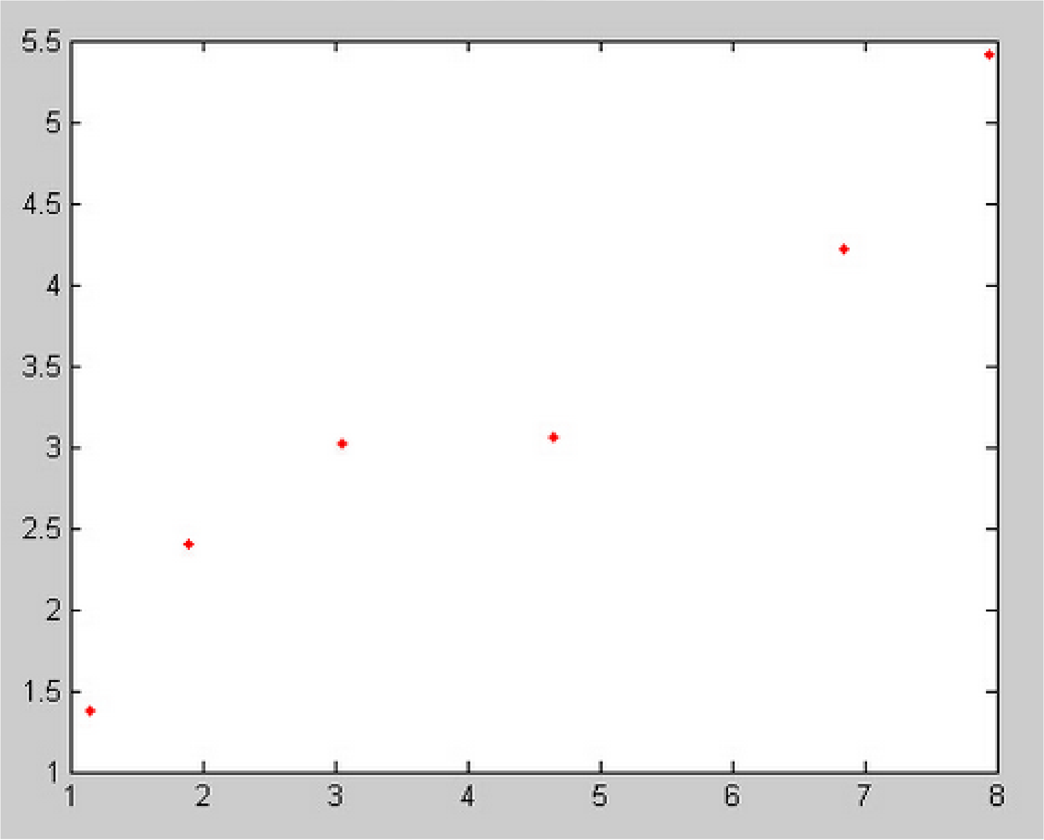
- 批量梯度下降**BGD**（Batch Gradient Descent ）
- 随机梯度下降**SGD**（Stochastic Gradient Descent ）
- 小批量梯度下降法**MBGD** (Mini-Batch Gradient Descent)

下面将以线性回归算法为例来对三种梯度下降法进行比较

梯度下降法

- 一元线性回归(拟合曲线)
- 假设这里存在m=6组数据(x, y)

y	x
1.37	1.15
2.4	1.9
3.02	3.06
3.06	4.66
4.22	6.84
5.42	7.95



梯度下降法

从图上看出，大致数据的大致走势是可以用线性模型 $h_\theta(x) = \theta_1 x + \theta_0$ 表示的，为此我们建立一维线性回归模型。

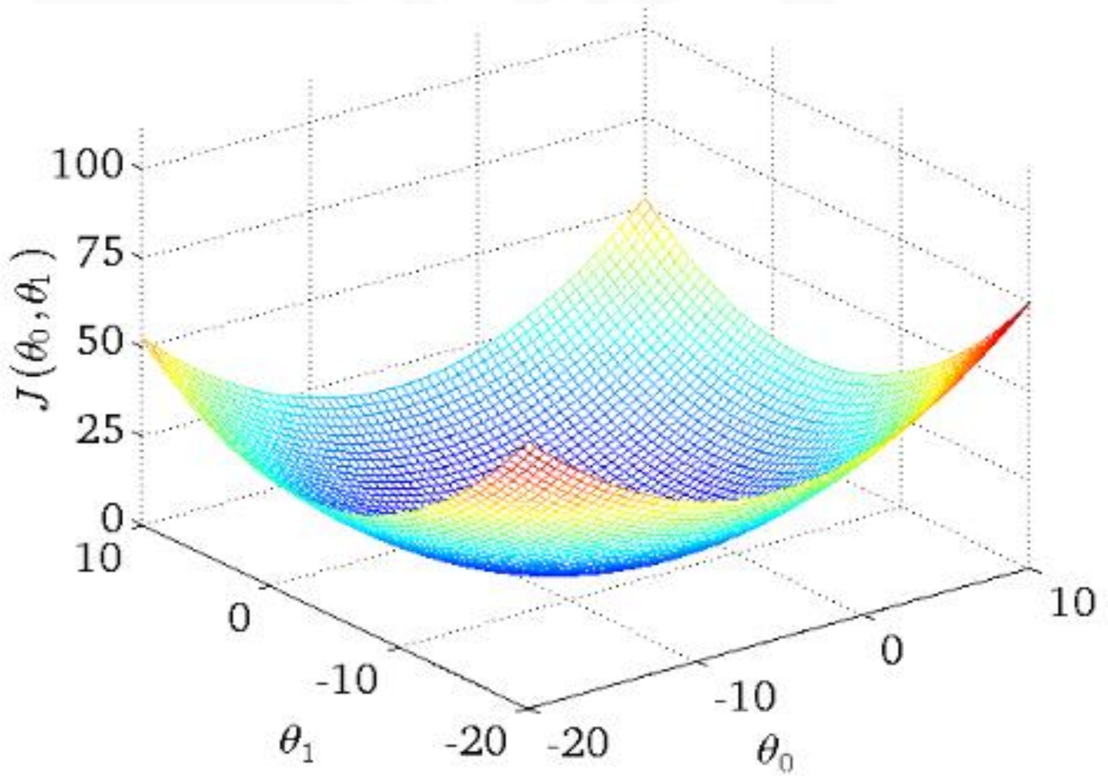
对应的**损失/误差函数**，即估计值与真实值之间的差距：

$$J(\theta) = \frac{1}{2N} \sum_{i=1}^{N} (h_\theta(x_i) - y_i)^2$$

其中：**N**是训练集的**样本个数**，1/2是为了后面求导计算方便

梯度下降法

- 一个二维参数（θ_0，θ_1）组对应能量函数（描述整个系统的优化程度，随着网络的变化而减小，最终**网络稳定时能量达到最小**）的可视化图



梯度下降法：批量梯度下降法BGD

- 更新算法的目的：**误差函数尽可能小**，即求解参数使误差函数尽可能小。
- 主要思想：
  - 首先，**随机初始化参数**；
  - 然后，**不断反复的更新参数使得误差函数减小**，直到满足要求时停止。

梯度下降法：批量梯度下降法BGD

- 梯度下降算法，利用初始化的参数 θ 并且反复更新参数 θ：

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

- **α代表学习率**，表示**每次向着函数J最陡峭的方向迈步的大小（步长）**

梯度下降法：批量梯度下降法BGD

（1）将J(θ)对θ求偏导，得到每个θ对应的的梯度

- J对第j个参数θ_j的偏导数是：

$$\frac{\partial J(\theta)}{\partial \theta_j} = \frac{1}{2N}\sum_{i=1}^{N}(h_\theta(\mathbf{x}_i) - y_i)^2$$

$$= 2 \cdot \frac{1}{2N}\sum_{i=1}^{N}(h_\theta(\mathbf{x_i}) - y_i)\frac{\partial}{\partial \theta_j}(\sum_{k=0}^{d}\theta_i x_{ik} - y_i)$$

$$= \frac{1}{N}\sum_{i=1}^{N}(h_\theta(\mathbf{x_i}) - y_i)x_{ij}$$

梯度下降法：批量梯度下降法BGD

（2）由于是要**最小化风险函数**，所以按每个参数 θ 的**梯度负方向**，来更新每个 θ_j(j=0, 1, 2, …, n)

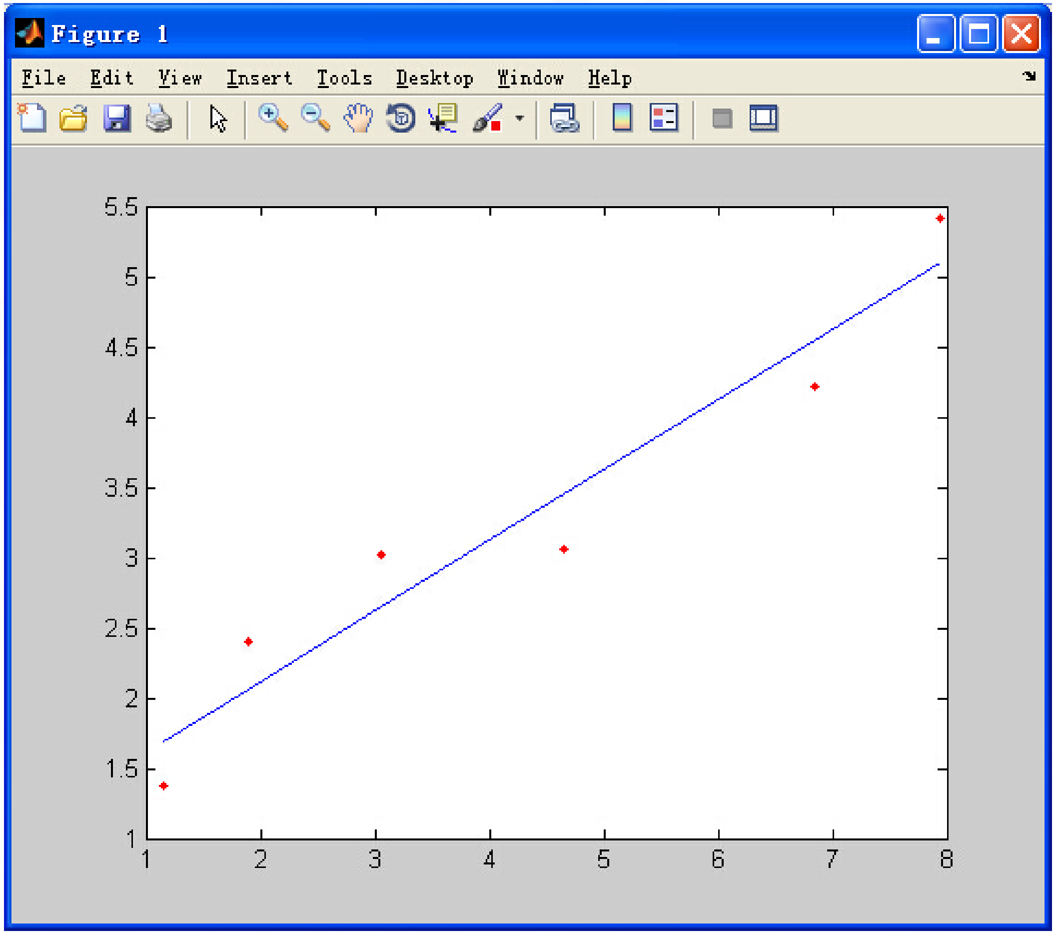
$$\theta'_j = \theta_j - \frac{1}{N}\sum_{i=1}^{N}(h_\theta(\mathbf{x_i}) - y_i)x_{ij}$$

梯度下降法：批量梯度下降法BGD

- 上例中，利用BGD求得

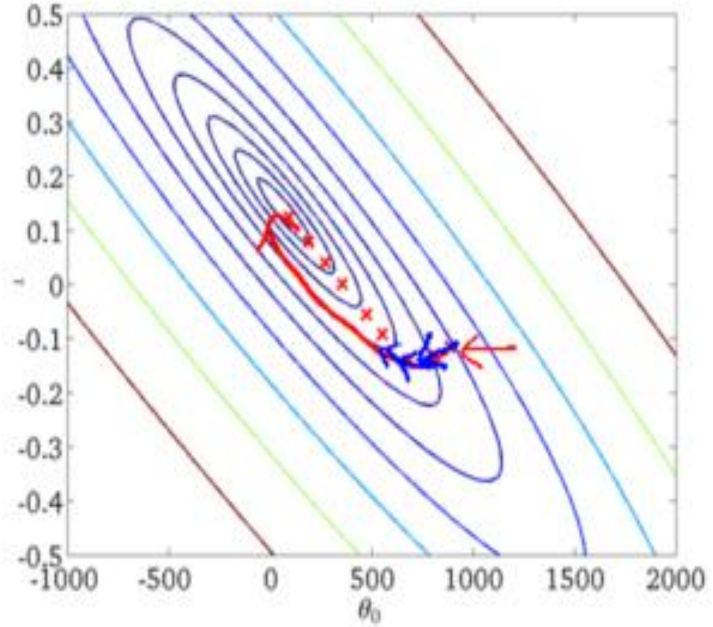
theta0=1.111094;

theta1=0.502401;



梯度下降法：批量梯度下降法BGD

- 由更新公式可知，批量梯度下降得到的是一个**全局最优解**，**每一次的参数更新都用到了所有的训练数据**，如果训练数据非常多的话，执行效率较低。



- 批量梯度下降法的收敛图（**迭代的次数相对较少**）

梯度下降法：随机梯度下降法SGD

- 由于批梯度下降每更新一个参数的时候，要用到**所有样本**，所以训练速度会随着样本数量的增加而变得非常缓慢。
- 随机梯度下降正是为了解决这个办法而提出的。它是利用**单个样本**的损失函数对θ求偏导得到对应的梯度，来更新θ。

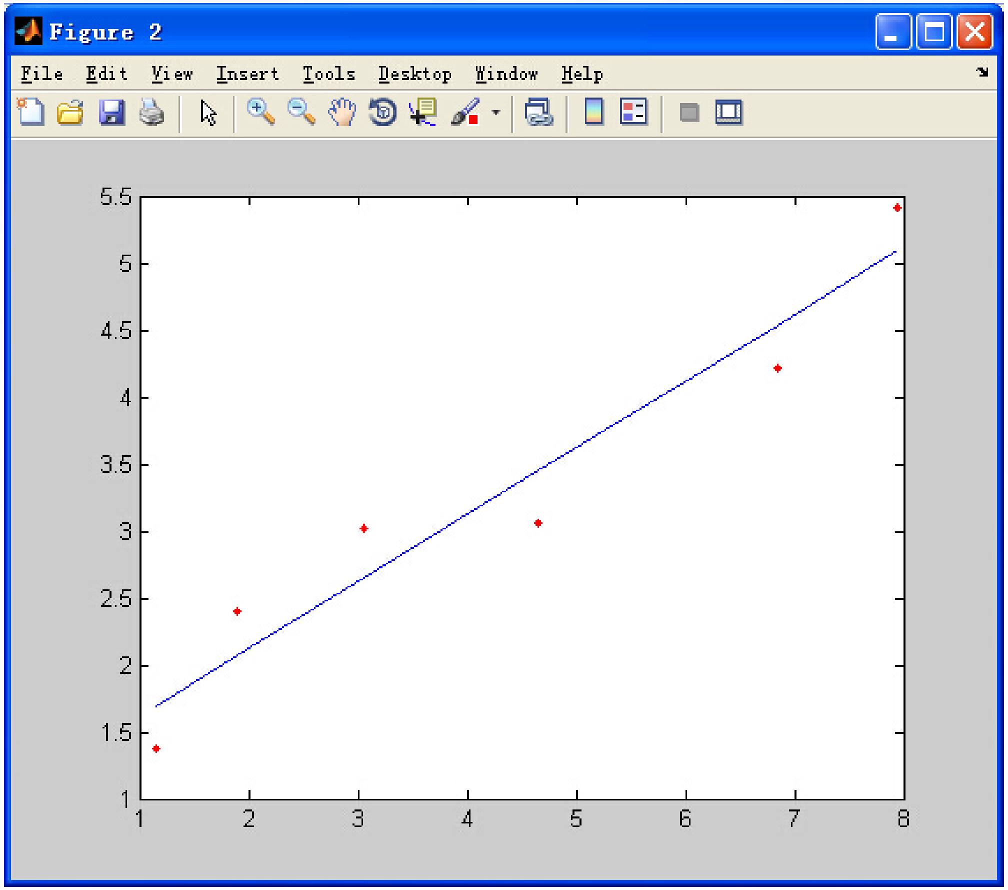
$$\theta'_j = \theta_j - (h_\theta(\mathbf{x_i}) - y_i)x_{ij}$$

梯度下降法：随机梯度下降法SGD

- 上例中，利用SGD求得

theta0=1.117690;

theta1=0.500151;

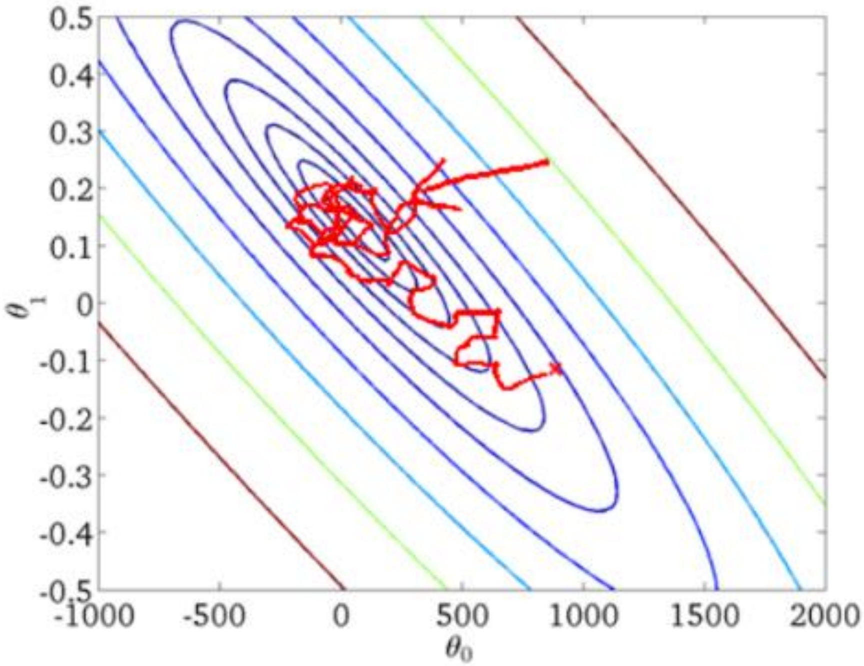


梯度下降法：随机梯度下降法SGD

- 随机梯度下降是通过**每个样本**来迭代更新一次，如果样本量很大的情况（例如几十万），那么可能只用其中几万条或者几千条的样本，就已经将参数迭代到最优解。
- 对比上面的批量梯度下降，迭代一次需要用到十几万训练样本，一次迭代不可能最优，如果迭代10次的话就需要遍历训练样本10次。
- SGD的问题是**噪音**较BGD要多，使得SGD并不是每次迭代都向着整体最优化方向。

梯度下降法：随机梯度下降法SGD

- 随机梯度下降收敛图（SGD迭代的次数较多，在解空间的搜索过程看起来很盲目。**但是大体上是往着最优值方向移动。**）



梯度下降法：小批量梯度下降法MBGD

- 为综合解决BGD的训练速度慢，以及SGD的准确性低的问题，提出MBGD
- 它是利用部分样本的损失函数对θ求偏导得到对应的梯度，来更新θ。

Repeat{

for i=1, 11, 21, 31, ... , 991{

$$\theta_j := \theta_j - \alpha \frac{1}{10} \sum_{k=i}^{i+9} (h_\theta(x^{(k)}) - y^{(k)}) x_j^{(k)}$$

(for every j=0, ... , n)

}

}

梯度下降法

方法	优点	缺点
BGD	最小化所有训练样本的损失函数，使得最终求解的是全局的最优解	如果样本值很大的话，更新速度会很慢。
SGD	最小化每个样本的损失函数，大大加快更新速度，最终的结果在全局最优解附近。	训练数据的噪声较多，导致不是每次迭代得到的损失函数都向着全局最优方向。
MBGD	训练速度快，参数准确性高	不同的问题需要设置不同的小批量值。

正则化

正则化

- 在实际应用时，如果样本容量不远远大于样本的特征维度，很可能造成过拟合
  - 加数据
  - 特征选择（降低特征维度）如 PCA 算法。
  - 正则化一般是在损失函数上加入正则化项，表示模型的复杂度对模型的惩罚

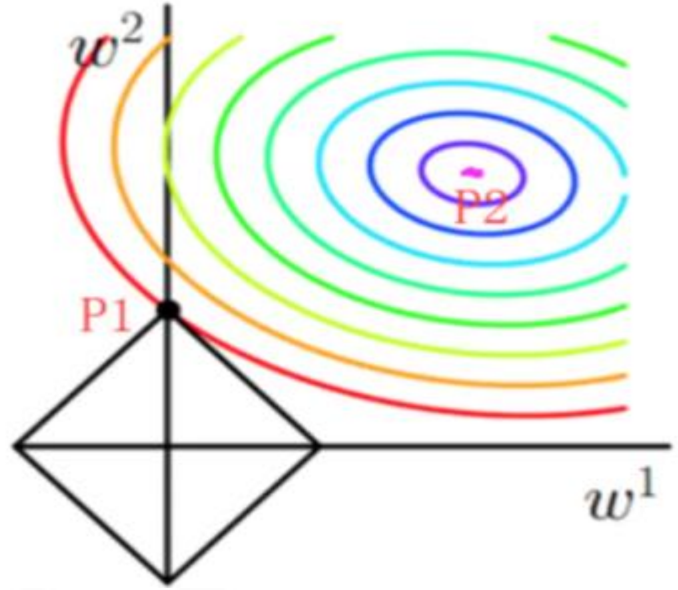
L1正则化

- Lasso回归

$$J(\omega) = (X\omega - Y)^T(X\omega - Y) + \lambda\|\omega_1\|$$

$$L^1 \text{正则化}\ \omega_1 := \omega_1 - \alpha \frac{d\bar{J}}{d\omega_1}$$

$$= \omega_1 - \frac{dJ}{d\omega_1} - \frac{\alpha\lambda}{2N} \operatorname{sign} \omega_1$$



- L1正则化相当于:

$$\underset{w}{argmin}\, L(w)$$

$$s.t.\ \|w\|_1 < C$$

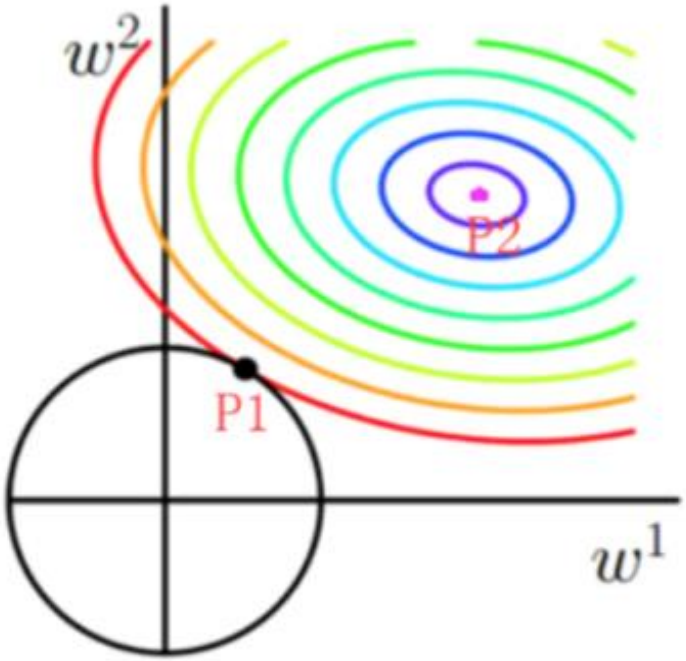
L2正则化

- 岭回归

$$\hat{w} = \underset{w}{argmin}\, L(w) + \lambda w^T w \longrightarrow \frac{\partial}{\partial w} L(w) + 2\lambda w = 0$$

$$\longrightarrow 2X^T X\hat{w} - 2X^T Y + 2\lambda \hat{w} = 0$$

$$\longrightarrow \hat{w} = (X^T X + \lambda \mathbb{I})^{-1} X^T Y$$



概率视角

噪声为高斯分布的MLE

- 对于一维的情况，记 $y=\omega^T x+\epsilon, \epsilon \sim \mathcal{N}(0,\sigma^2)$

则 $y \sim \mathcal{N}(\omega^T x, \sigma^2)$，根据极大似然法有：

$$
\begin{aligned}
L(w) = \log p(Y|X,\omega) &= \log \prod_{i=1}^{N} p(y_i|x_i,\omega) \\
&= \sum_{i=1}^{N} \log\left(\frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{(y_i-\omega^T x_i)^2}{2\sigma^2}}\right) \\
\underset{w}{argmax}\, L(\omega) &= \underset{\omega}{argmin} \sum_{i=1}^{N} (y_i-\omega^T x_i)^2
\end{aligned}
$$

这个表达式和最小二乘估计得到的结果一样

权重先验为高斯分布的MAP

- 取先验分布 $w \sim \mathcal{N}(0, \sigma_0^2)$

$$
\begin{aligned}
\hat{\omega} = \underset{w}{argmax}\, p(\omega|Y) &= \underset{w}{argmax}\, p(Y|\omega)p(\omega) \\
&= \underset{\omega}{argmax} \log p(Y|\omega)p(\omega) \\
&= \underset{\omega}{argmax}(\log p(Y|\omega) + \log p(\omega)) \\
&= \underset{w}{argmin}[\sum_{i=1}^{N}(y - \omega^T x)^2 + \frac{\sigma^2}{\sigma_0^2}\omega^T \omega]
\end{aligned}
$$

- 该问题对应Ridge回归
- 如果将先验分布取为Laplace分布，对应着L1正则化。

$$
f(x) = \frac{1}{2\lambda} e^{-\frac{|x-\mu|}{\lambda}}
$$

THE END