回归算法——LR

OutLine

线性回归

逻辑回归

【实践】LR模型

回 归 技 术 概 述

- 因变量y和自变量x的关系
  - y与x相关：y=f(x, w)　w为参数
  - y还受到噪声的影响：y=f(x, w) + ε

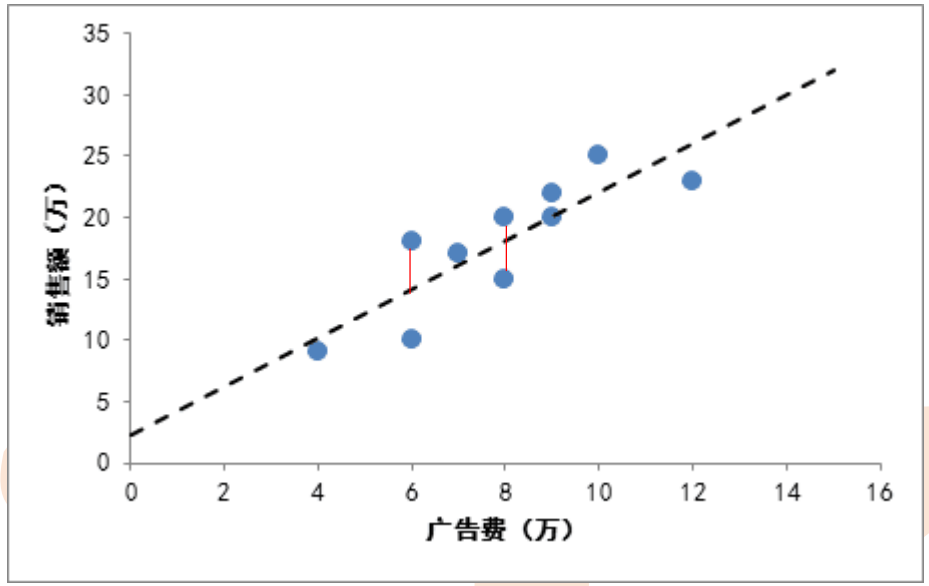
- 应用：
  - 发现y和x的关系
  - 发现规律，以预测新情况下的y值
    - 点击率预估，销量预测

回 归 技 术 概 述

- 模型
  - 线性 和 非线性
- 问题
  - 回归 vs. 分类中类别概率回归
- 典型方法：
  - 线性回归
  - 逻辑回归
  - 神经网络
  - 等等。。。

一元线性回归

- 输入：一元自变量x，一元因变量y
  - n个样本(x1, y1) , …… , (xn, yn)
- 回归模型
  - 模型输出：$\hat{y} = wx + b$
  - 模型参数：w, b
- 如何学习参数?
  - 对每个样本，误差：$\varepsilon_i = y_i - \hat{y_i}$
  - 最小化误差平方和：$\min_{w,b} \sum_i \varepsilon_i{}^2 = \sum_i (wx_i + b - y_i)^2$



多元线性回归

- 多元：
  - 自变量（向量）

$$x_i = \begin{bmatrix} x_{i1} \\ \vdots \\ x_{ik} \end{bmatrix}$$

  - 回归方程：

$$f(x_i, w, b) = \sum_j w_j x_{ij} + b = [x_{i1} \quad \cdots \quad x_{ik}] \begin{bmatrix} w_1 \\ \vdots \\ w_k \end{bmatrix} + b = w_i x + b$$

多元线性回归

- 统一写为：

$$f(x_i, w, b) = [x_{i1} \quad \cdots \quad x_{in}] \begin{bmatrix} w_1 \\ \vdots \\ w_n \end{bmatrix} + b = [x_{i1} \quad \cdots \quad x_{in} \quad 1] \begin{bmatrix} w_1 \\ \vdots \\ w_n \\ b \end{bmatrix} = \widetilde{w} \cdot \widetilde{x_i}$$

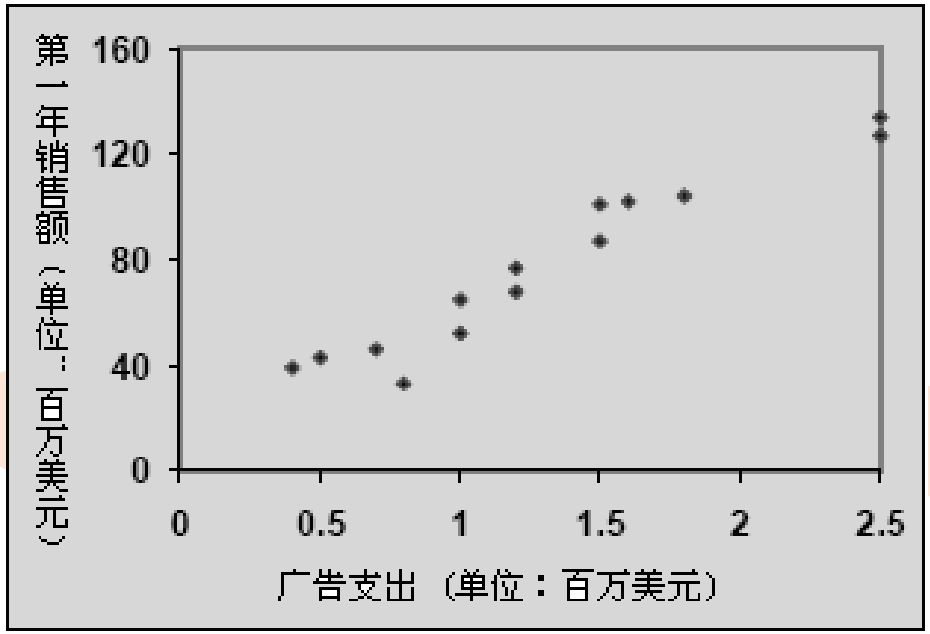
- 只需要研究：

$$f(x_i, w) = wx$$

  - 参数：w

线性回归应用

- 销售预测
- 广告支出预算制定



Appleglo	第一年广告支出（单位：百万美元）	第一年销售额（单位：百万美元）
地域	x	y
缅因州	1.8	104
新罕布什尔州	1.2	68
佛蒙特州	0.4	39
马萨诸塞州	0.5	43
康涅狄格	2.5	127
罗得岛州	2.5	134
纽约	1.5	87
新泽西	1.2	77
宾夕法尼亚州	1.6	102
特拉华	1.0	65
马里兰	1.5	101
西弗吉尼亚	0.7	46
维吉尼亚	1.0	52
俄亥俄州	0.8	33

OutLine

- 线性回归
- 逻辑回归
- 【实践】LR模型

逻辑回归

- 因变量y是类别标号{0, 1}
  - 1：发生，0：不发生
- 目标：根据特征x，预测发生概率p(y=1|x)
- 假定p(y=1|x)依赖于线性函数wx
  - 设定p(y=1|x)=wx?　　wx可能不在[0, 1]内！
  - 指数变换：exp(wx) ∈ (0, +∞)
  - 设定几率：$\frac{p(y=1|x)}{p(y=0|x)} = \frac{p(y=1|x)}{1-p(y=1|x)} \in [0, +\infty)$
  - 设定：$\frac{p(y=1|x)}{p(y=0|x)} = \exp(wx)$

$\log(p/(1-p)) = wx$

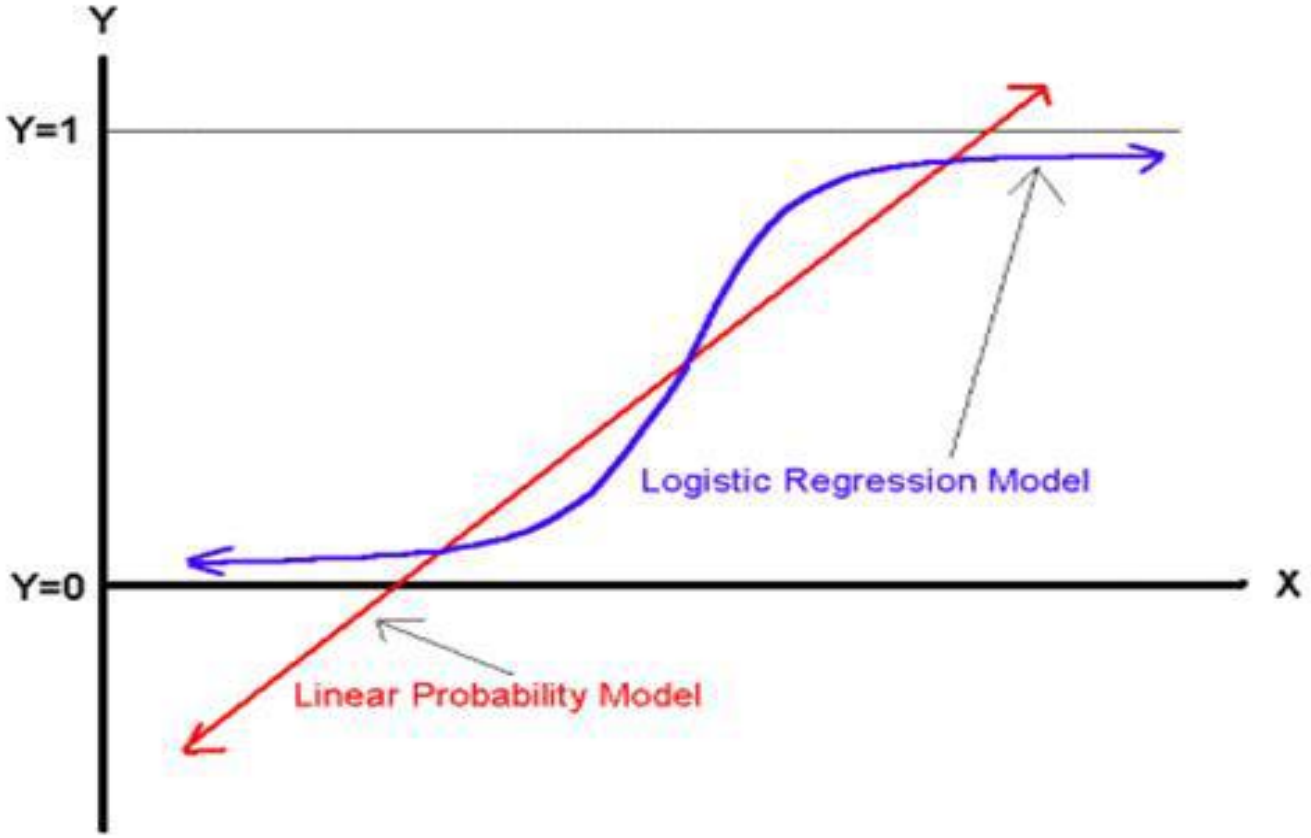
Logit函数

- 所以 $p(y=1|x) = \frac{\exp(wx)}{1+\exp(wx)} = \frac{1}{1+\exp(-wx)}$

逻辑回归

- Sigmoid函数：

$$\eta(t) = \frac{1}{1 + \exp(-t)}$$



- 类别概率：

$$p(y = 1|x) = \eta(wx)$$

$$p(y = 0|x) = 1 - \eta(wx)$$

最大似然

- 当从模型总体随机抽取n组样本观测后，最合理的参数估计量应该使得从模型中抽取该n组样本观测值的概率最大。
- 举例：
  - 一个袋子中有20个球，只有黑白两色，有放回的抽取十次，取出8个黑球和2个白球，计算袋子里有白球黑球各几个。那么我会认为我所抽出的这个样本是被抽取的事件中概率最大的。p（黑球）=p^8*（1-p）^2,让这个值最大。极大似然法就是基于这种思想

逻辑回归

- 给定样本（xi,yi），假设样本独立
  - 负对数似然：$L(w) = -\sum_i (I(y_i = 1)\log(p(y_i = 1|x_i, w)) + I(y_i = 0)\log(p(y_i = 0|x_i, w)))$

    $= -\sum (I(y_i = 1)\log(\eta(wx_i)) + I(y_i = 0)\log(1 - \eta(wx_i)))$
  - 每个样本属于其真实标记的概率越大越好
  - 由于L(w)是高阶连续可导的凸函数，根据凸优化理论，可利用梯度下降法、牛顿法等求解
  - 梯度：$\nabla L(w) = -\sum_i (y_i - \eta(wx_i))x_i$

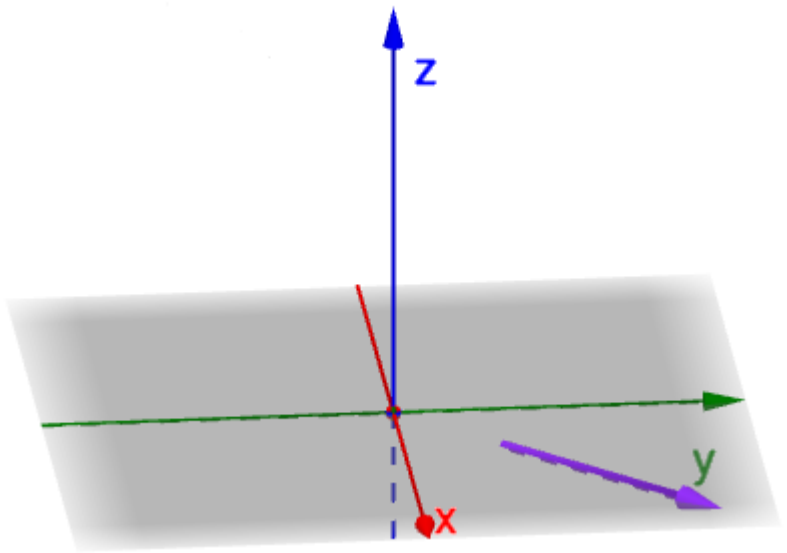
玩一个游戏

- 想要达到山谷底的湖泊，你该怎么办？

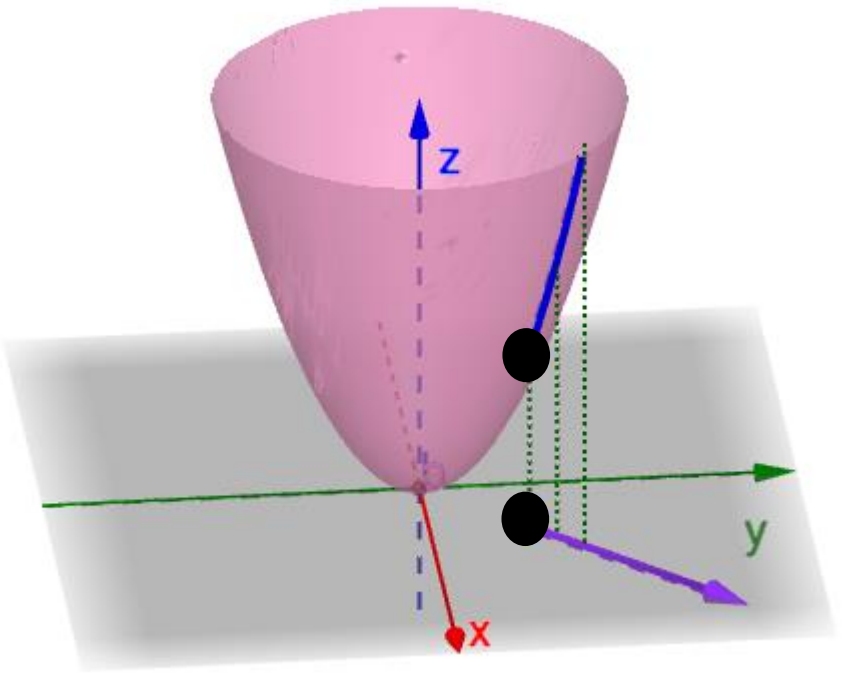
方向导数

- 某一个方向的导数

xy平面上一个矢量表示方向



函数f(x,y)在这个方向的图像

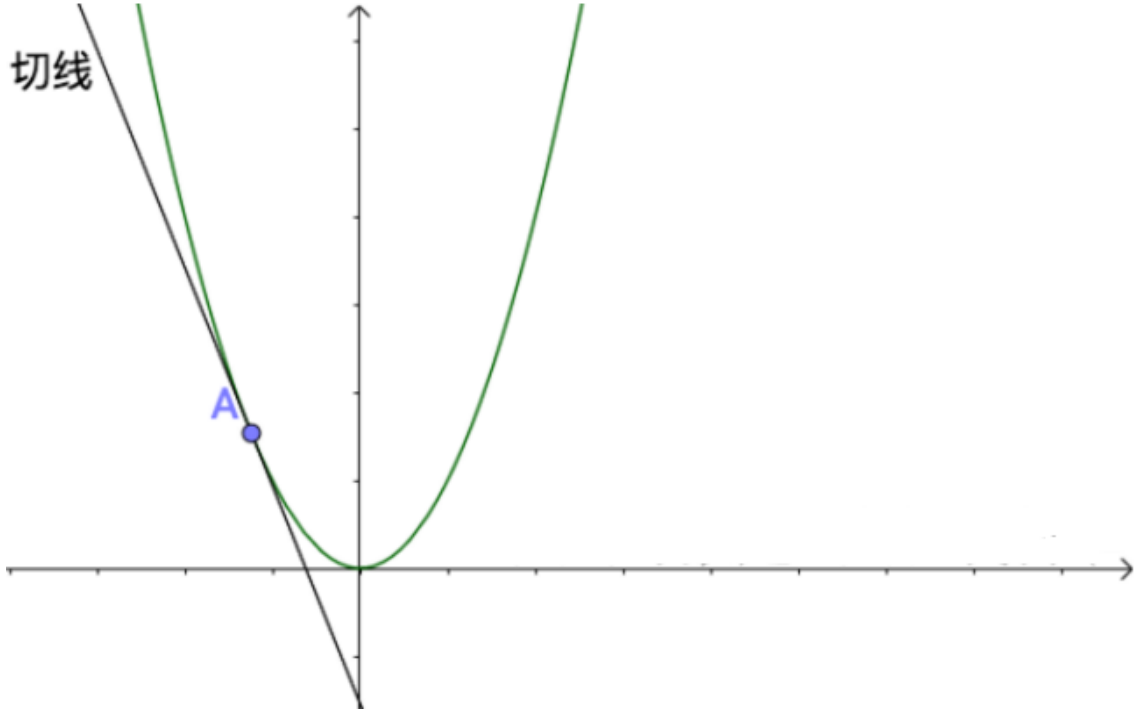


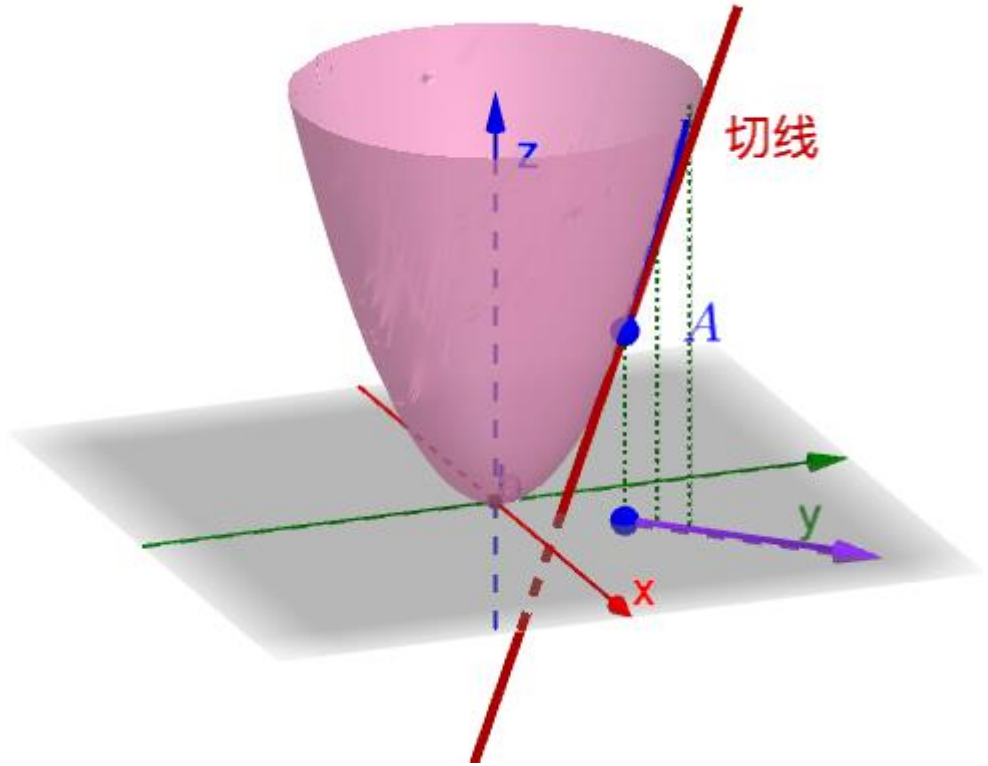
方向导数

- 某一个方向的导数

函数f(x,y)在A点在这个方向也是有切线的，其切线斜率就是方向导数！

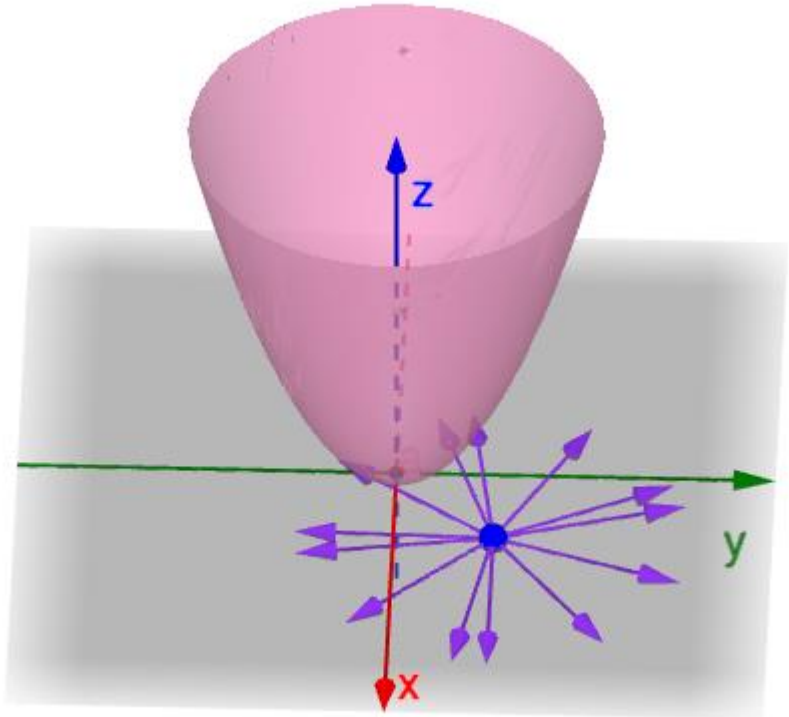
在一元函数中，A点的导数就是A点切线斜率





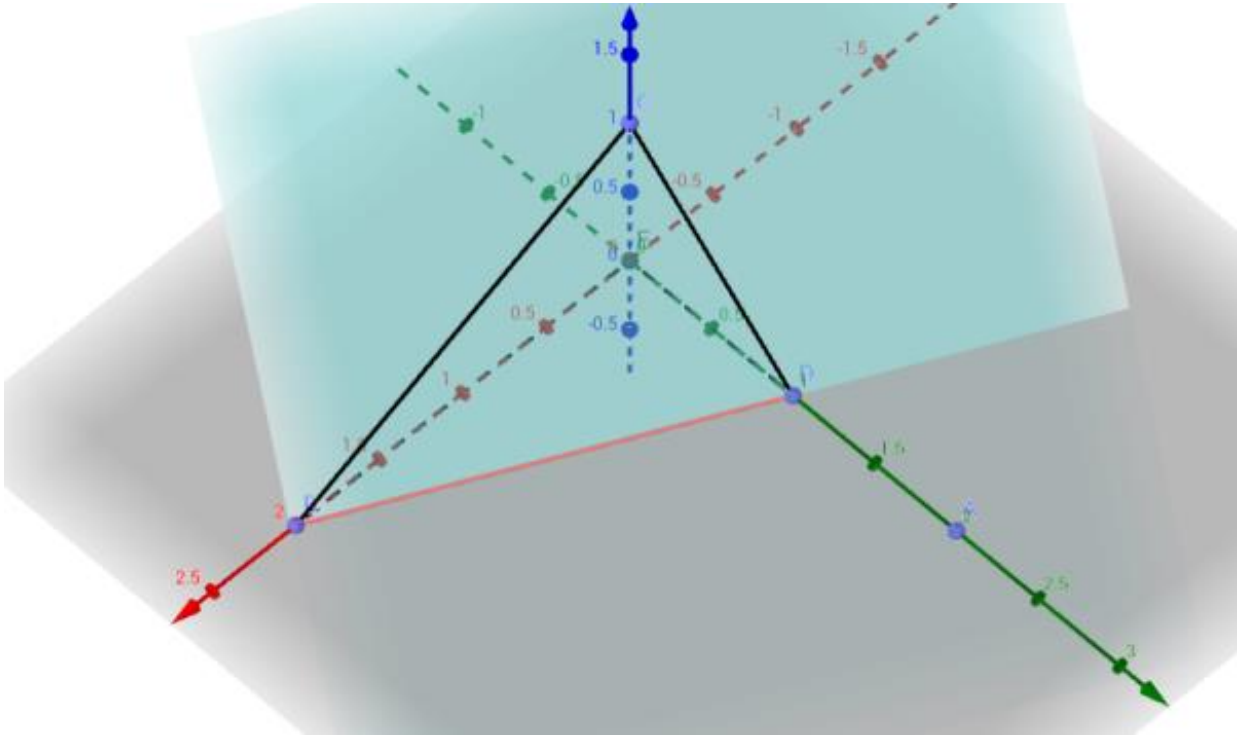
梯 度

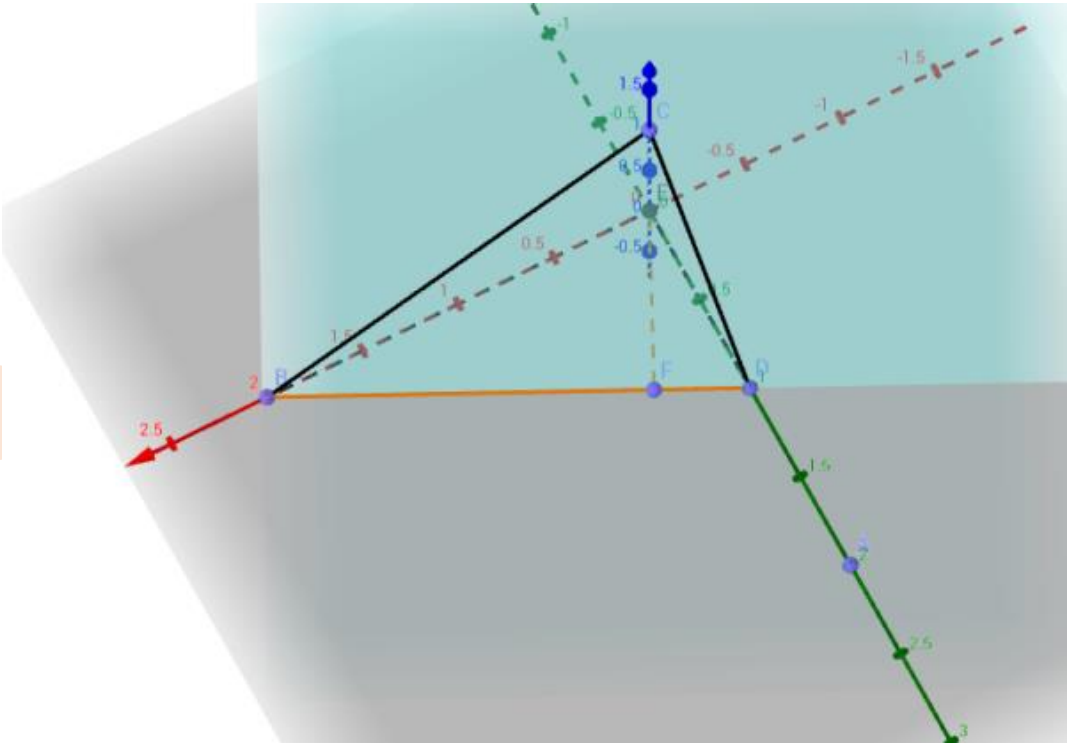
- 显然A点不止一个方向！每个方向都有方向导数！
- 定义：梯度是一个向量，其方向上的方向导数最大，其大小为此最大方向的导数



梯 度

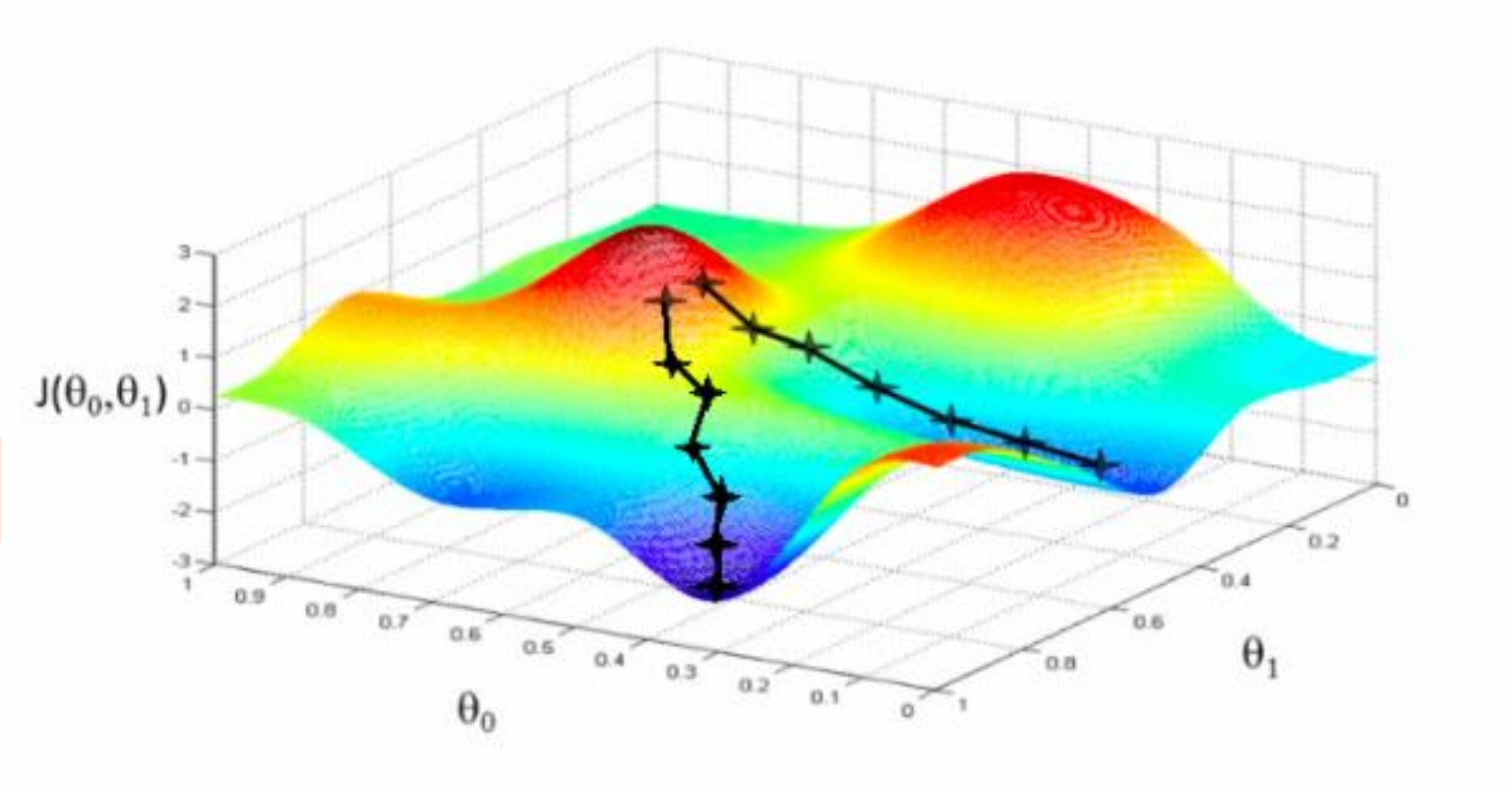
- 梯度的反方向是下降最快的方向！





梯 度

- 梯度的反方向是下降最快的方向！



梯度下降法

- 梯度下降法：最小化F(w)
  - 1、设置初始w，计算F(w)
  - 2、计算梯度 $\nabla F(w)$
    - 下降方向：dir = $(-\nabla F(w))$
  - 3、 尝试梯度更新
    - $w^{new} = w +$ 步长 $* dir$ 得到下降后的 w^{new} 和 $F(w^{new})$
  - 4、如果 $F(w^{new}) - F(w)$ 较小，停止；

  否则 $w = w^{new}; F(w) = F(w^{new})$ 跳到第2步

$$\nabla L(w) = -\sum_i (y_i - \eta(wx_i))x_i$$

梯度下降总结：

- 批量梯度下降BGD

$$h_\theta(x)=\theta^T X \Rightarrow \min_\theta J_\theta,\ J(\theta)=\frac{1}{2}\sum_{i=1}^{m}(h_\theta(x^{(i)})-y^{(i)})^2 \Rightarrow \frac{\partial J(\theta)}{\partial \theta_j}=-\frac{1}{m}\sum_{i=1}^{m}(y^i-h_\theta(x^i))x_j^i$$

$$\theta_j^{'}=\theta_j+\lambda\frac{1}{m}\sum_{i=1}^{m}\left(y^i-h_\theta\left(x^i\right)\right)x_j^i$$

```
repeat until convergency{
    for j=1;j<n ; j++:
        w_j = w_j − a((1/m) Σ_{i=1}^{m} (y^(i) − w · x^(i)) x_j^(i) + λ w_j)
}
```

- 优点：每次迭代遍历所有样本，共同决定最优方向
- 缺点：样本数量大时，不适用

梯度下降总结：

- 随机梯度下降SGD
  - 每次从训练样本中抽取一个样本进行更新，每次都不用遍历所有数据集，迭代速度快，需要迭代更多次数
  - 每次选取方向不一定是最优

repeat until convergency{

random choice j from all m training example:

$$w_j = w_j - a\left(\left(y^{(i)} - \mathbf{w}\cdot\boldsymbol{x}^{(i)}\right)x_j^{(i)} + \lambda w_j\right)$$

}

梯度下降总结：

- 小批量梯度下降MBGD
  - 介于以上两种方法的折中，每次随机选取大小batch_size的子集进行批量训练
  - 节省时间，更加准确

repeat until convergency{

for j=1;j<n ; j+=b:

$$w_j = w_j - a\left(\frac{1}{b}\sum_{i}^{i+b}\left(y^{(i)} - \mathbf{w}\cdot\boldsymbol{x}^{(i)}\right)x_j^{(i)} + \lambda w_j\right)$$

}

梯度下降并行化方案：

Algorithm 3 SimuParallelSGD(Examples $\{c^1, \ldots c^m\}$, Learning Rate η, Machines k)

Define $T = \lfloor m/k \rfloor$
Randomly partition the examples, giving T examples to each machine.
for all $i \in \{1, \ldots k\}$ **parallel do**
  Randomly shuffle the data on machine i.
  Initialize $w_{i,0} = 0$.
  for all $t \in \{1, \ldots T\}$: **do**
    Get the tth example on the ith machine (this machine), $c^{i,t}$
    $w_{i,t} \leftarrow w_{i,t-1} - \eta \partial_w c^i(w_{i,t-1})$
  end for
end for
Aggregate from all computers $v = \frac{1}{k} \sum_{i=1}^{k} w_{i,t}$ and **return** v.

OutLine

- 线性回归
- 逻辑回归
- 【实践】LR模型

Q & A

@八斗数据