

机器学习

第二讲: 线性回归

顾小东

上海交通大学计算机学院



机器学习中的回归问题

机器学习

— 监督学习

— 回归 (✓)

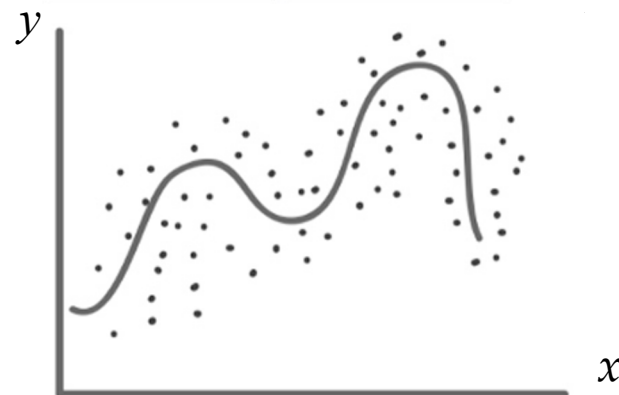
— 分类

— ...

— 无监督学习

— 强化学习

Advertisement	Sales
\$90	\$1000
\$120	\$1300
\$150	\$1800
\$100	\$1200
\$130	\$1380
\$200	??



Regression: predicts real-valued labels

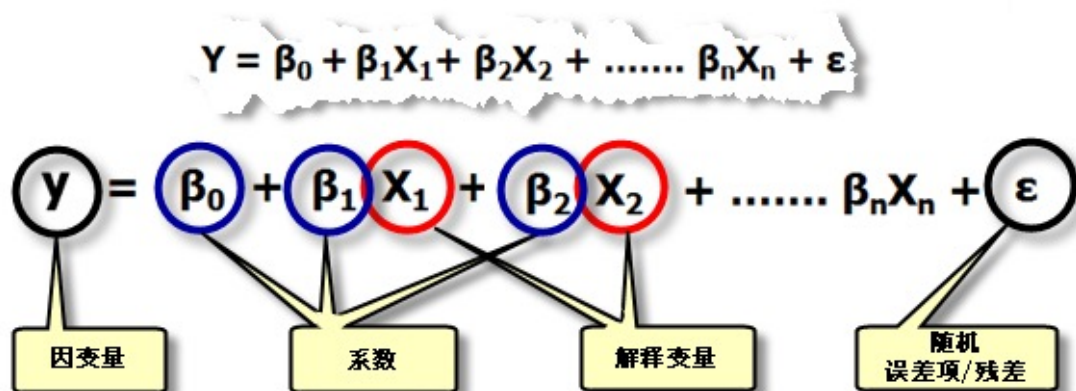
回归问题

为什么要作回归?

人们在测量事物的时候因为客观条件所限，求得的都是测量值，而不是事物真实的值，为了能够得到真实值，无限次的进行测量，最后通过这些测量数据计算**回归到真实值**

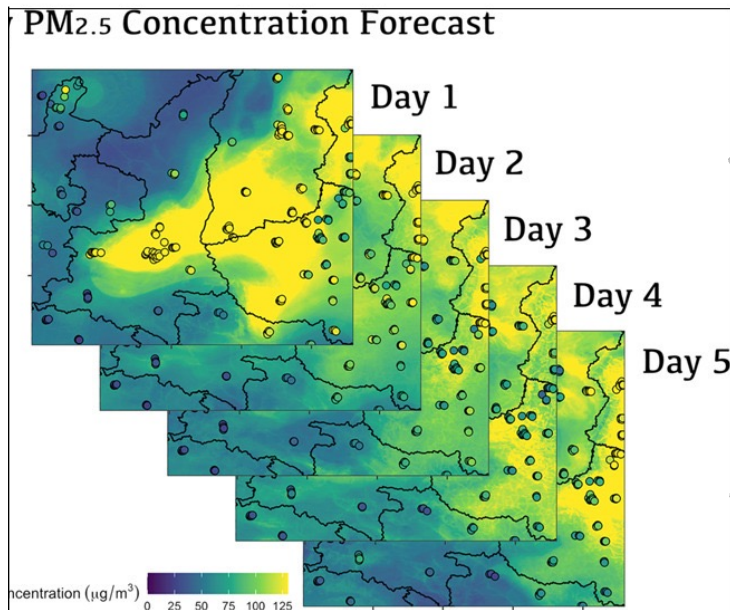
什么是回归?

回归是用来描述一个因变量和若干自变量之间关系的一种统计方法。

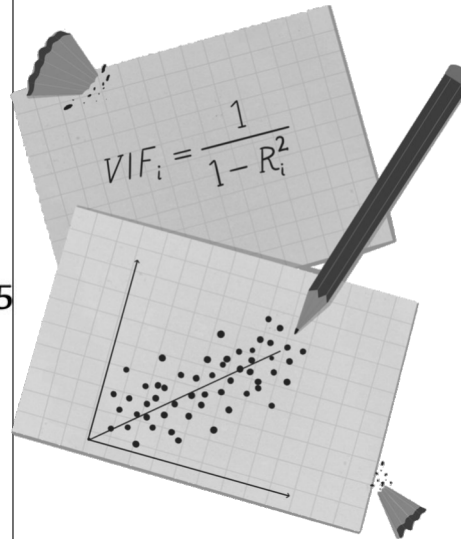


回归问题的应用

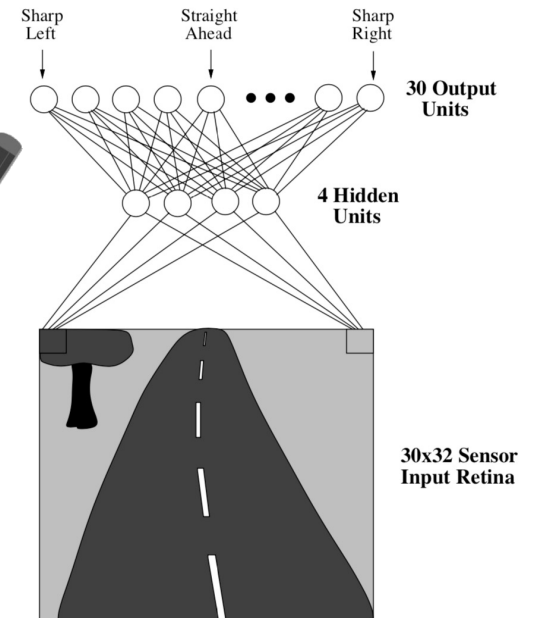
预测



归因分析



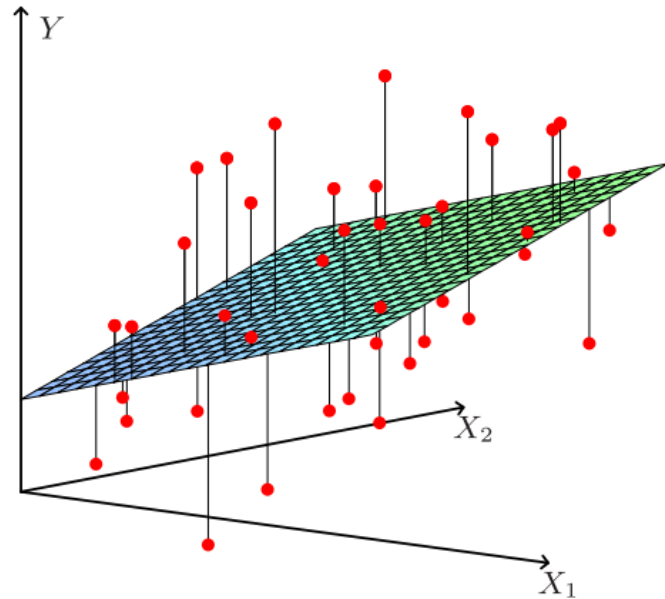
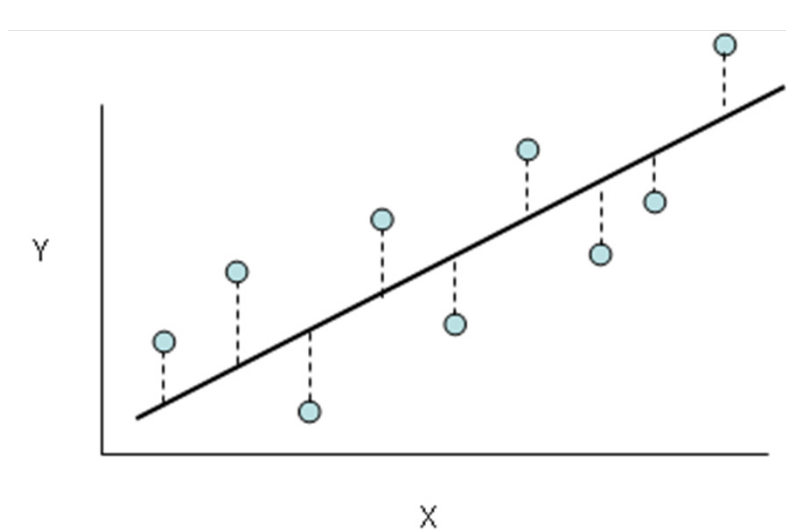
控制



线性回归

使用一个线性函数进行回归

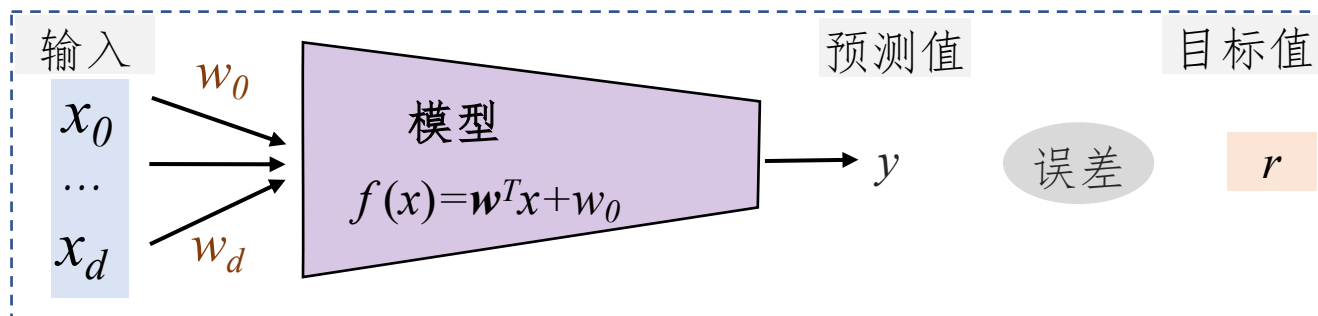
$$y = f(x) = \mathbf{w}^T \mathbf{x} + w_0$$



Linear model for regression is a $(d+1)$ -dimensional hyperplane

模型结构

线性回归模型的核心是一个简单的线性函数



- **训练过程:**
 - 根据数据误差，估算合适的参数 $\mathbf{w}$ 和 w_0
- **预测过程:**
 - 对于输入的新数据 x , 计算 $f(x) = \mathbf{w}^T \mathbf{x} + w_0$, 得到回归值 y

损失函数

- 对于任一输入样本 x , 模型输出预测值 y . 令 $r \in \mathbb{R}$ 为该样本对应的目标值, 则相应的平方误差定义为:

$$l(\mathbf{w}, w_0 \mid x, r) = (r - y)^2$$

- 对于完整数据集 $D = \{(x^{(1)}, r^{(1)}), \dots, (x^{(N)}, r^{(N)})\}$, 损失函数定义为均方误差 (MSE):

$$L(\mathbf{w}, w_0 \mid D) = \frac{1}{2N} \sum_{\ell=1}^N (r^{(\ell)} - y^{(\ell)})^2$$

训练优化

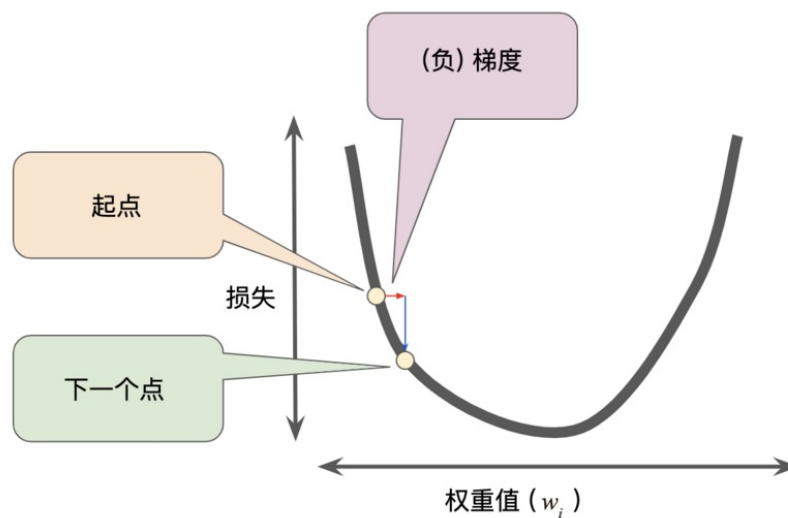
一般采用**梯度下降法**(Gradient Descend)优化损失函数:

- 优化目标:

$$\min_w L(w)$$

- 迭代步骤:

$$w_{t+1} = w_t - \eta_t \frac{\partial L}{\partial w}$$



如何计算 $\frac{\partial L}{\partial w}$?

梯度下降法优化模型

$$L(\mathbf{w}, w_0 | D) = -\frac{1}{2N} \sum_{\ell=1}^N (r^{(\ell)} - y^{(\ell)})^2$$

怎么求 $\frac{\partial L}{\partial w}$?

对于任意 w_j ($j=1, \dots, d$):

$$\frac{\partial L}{\partial w_j} = -\frac{1}{N} \sum_{\ell} \underbrace{(r^{(\ell)} - y^{(\ell)})}_{\text{Chain rule}} \frac{\partial y^{(\ell)}}{\partial w_j} = -\frac{1}{N} \sum_{\ell} (r^{(\ell)} - y^{(\ell)}) x^{(\ell)}$$

迭代规则：

$$\mathbf{w}_{\text{new}} = \mathbf{w}_{\text{old}} + \frac{1}{N} \sum_{\ell=1}^N (r^{(\ell)} - y^{(\ell)}) \mathbf{x}^{(\ell)}$$

算法总结

采用梯度下降法训练线性回归模型

```
Input:  $D = \{(x^{(l)}, r^{(l)})\} (l=1:N)$ 
for  $j = 0, \dots, d$ 
     $w_j \leftarrow \text{rand}(-0.01, 0.01)$ 
repeat
    for  $j = 0, \dots, d$ 
         $\Delta w_j \leftarrow 0$ 
    for  $l = 1, \dots, N$ 
         $y \leftarrow 0$ 
        for  $j = 0, \dots, d$ 
             $y \leftarrow y + w_j x_j^{(l)}$ 
             $\Delta w_j \leftarrow \Delta w_j + (r^{(l)} - y) x_j^{(l)}$ 
         $\Delta w_j = \Delta w_j / N$ 
    for  $j = 0, \dots, d$ 
         $w_j \leftarrow w_j + \eta \Delta w_j$ 
until convergence
```

线性回归的矩阵形式

$$\text{令 } X = \begin{bmatrix} \mathbf{x}^{(1)} \\ \mathbf{x}^{(2)} \\ \vdots \\ \mathbf{x}^{(N)} \end{bmatrix} = \begin{bmatrix} x_0^{(1)} & x_1^{(1)} & x_2^{(1)} & \cdots & x_d^{(1)} \\ x_0^{(2)} & x_1^{(2)} & x_2^{(2)} & \cdots & x_d^{(2)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_0^{(N)} & x_1^{(N)} & x_2^{(N)} & \cdots & x_d^{(N)} \end{bmatrix} \quad \mathbf{w} = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_d \end{bmatrix} \quad \mathbf{r} = \begin{bmatrix} r^{(1)} \\ r^{(2)} \\ \vdots \\ r^{(N)} \end{bmatrix}$$

- 预测值: $y = X\mathbf{w} = \begin{bmatrix} x^{(1)}\mathbf{w} \\ x^{(2)}\mathbf{w} \\ \vdots \\ x^{(N)}\mathbf{w} \end{bmatrix}$
- 损失函数: $L(\mathbf{w}) = \frac{1}{2} (\mathbf{r} - \mathbf{y})^T (\mathbf{r} - \mathbf{y}) = \frac{1}{2} (\mathbf{r} - X\mathbf{w})^T (\mathbf{r} - X\mathbf{w})$

线性回归的矩阵形式

- 梯度

$$\frac{\partial L(w)}{\partial w} = -X^T(r - Xw)$$

- 最优参数

$$\begin{aligned}\frac{\partial L(w)}{\partial w} = 0 &\Rightarrow X^T(r - Xw) = 0 \\ &\Rightarrow X^T r = X^T X w \\ &\Rightarrow \mathbf{w}^* = (X^T X)^{-1} X^T r\end{aligned}$$

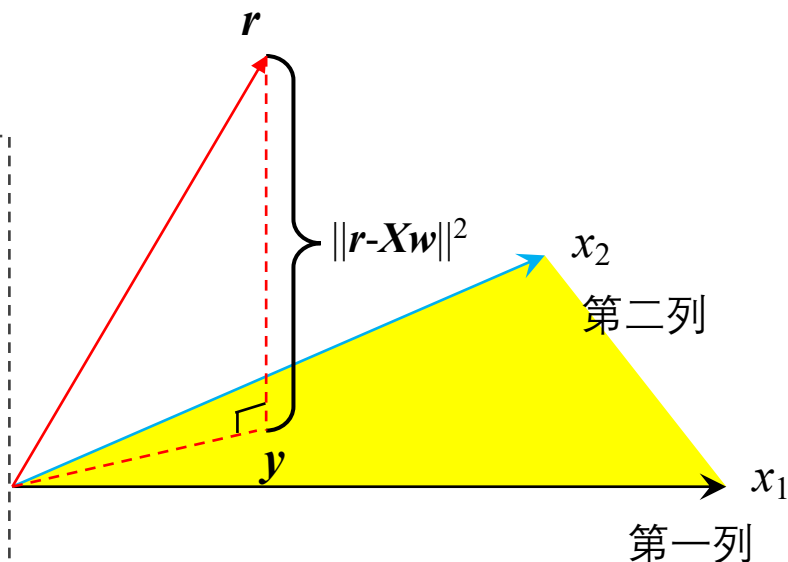
线性回归的矩阵形式

- 将最优参数 w^* 代入原始模型，得到预测值为

$$\begin{aligned} y &= X(X^T X)^{-1} X^T r \\ &= H r \end{aligned}$$

几何解释

- y 是 x 的线性组合，因此是 X 列向量 $[x_1, x_2, \dots, x_d]$ 构成的子空间上的一点
- $y = Hr$ ，说明 H 将 r 投射到 y
- 需要 y 与 r 距离最短，因此 H 是向量 r 在该子空间上距离最短的一个投影



正则化

问题: $X^T X$ 可能不可逆

When some column vectors are not independent (e.g., $\mathbf{x}_2=3\mathbf{x}_1$), then $X^T X$ is singular, thus $\mathbf{w}^* = (X^T X)^{-1} X^T \mathbf{r}$ cannot be directly calculated.

解决方法: 正则化 (对损失函数引入一个惩罚项)

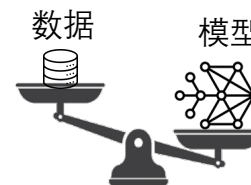
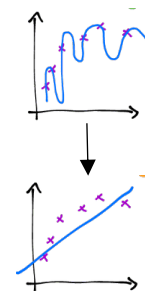
$$L(\mathbf{w}) = \frac{1}{2} (\mathbf{r} - \mathbf{y})^T (\mathbf{r} - \mathbf{y}) = \frac{1}{2} (\mathbf{r} - X\mathbf{w})^T (\mathbf{r} - X\mathbf{w}) + \frac{\lambda}{2} \|\mathbf{w}\|_2^2$$

新的梯度: $\frac{\partial L(\mathbf{w})}{\partial \mathbf{w}} = -X^T (\mathbf{r} - X\mathbf{w}) + \lambda \mathbf{w}$

新的最优解: $\frac{\partial L(\mathbf{w})}{\partial \mathbf{w}} = 0 \Rightarrow -X^T (\mathbf{r} - X\mathbf{w}) + \lambda \mathbf{w} = 0$

$$\Rightarrow X^T \mathbf{r} = (X^T X + \lambda \mathbf{I}) \mathbf{w}$$

$$\Rightarrow \mathbf{w}^* = (X^T X + \lambda \mathbf{I})^{-1} X^T \mathbf{r}$$



Penalty to model amounts to data augmentation (adding data prior)

敲一敲代码



Python

<https://colab.research.google.com/github/cs231n/cs231n.github.io/blob/master/python-colab.ipynb>

Linear regression with Python

<https://www.kaggle.com/code/sudhirn17/linear-regression-tutorial/data?select=insurance.csv>

What's Next?



Classifications

Find a decision boundary that **maximizes the margin** between two classes.

Machine Learning

- **Supervised Learning**
 - Regression
 - Classification(✓)
 - ...
- Unsupervised Learning
- Reinforcement Learning

