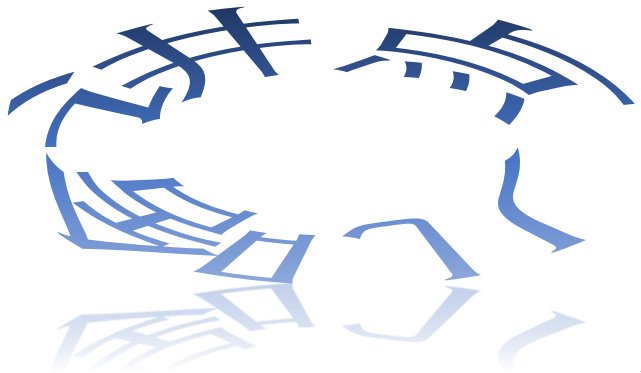
明理 精工

笃学 致远

第3章 线性回归

电子工程学院、人工智能学院

college of Electronic Engineering、college of Artificial Intelligence



- 线性回归模型
  - 回归任务的损失函数
  - 线性回归模型的正则项
  - 线性回归的的优化算法
  - 回归任务的模型性能评价

- 回忆：机器学习的三要素
  - 1. 函数集合$\{f_1, f_2, \dots\}$
  - 2. 目标函数$J(f)$：函数的好坏
  - 3. 优化算法：找到最佳函数

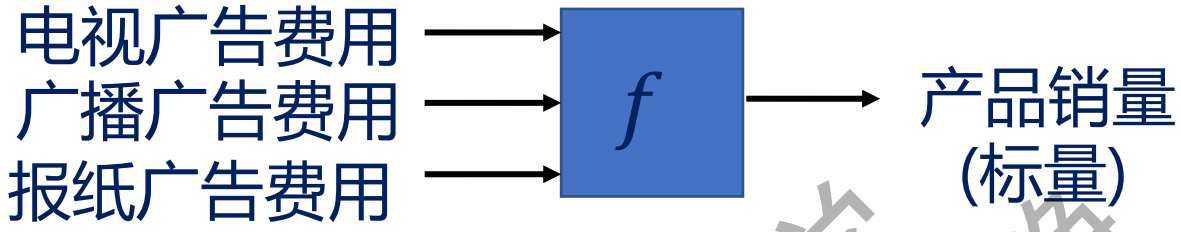
- 线性回归：函数集合为输入特征的线性组合，即假设输出y与输入$\boldsymbol{x}$之间的关系为线性关系

$$\hat{y} = f(\boldsymbol{x}) = w_0 + \sum_{j=1}^{D} w_j x_j = \begin{bmatrix} w_0 & w_1 & \dots & w_D \end{bmatrix} \begin{bmatrix} 1 \\ x_1 \\ \dots \\ x_D \end{bmatrix} = \boldsymbol{w}^{\mathrm{T}} \boldsymbol{x}$$

截距项

第j维特征的权重/回归系数

  - D: 特征维数，j为特征索引
  - $\boldsymbol{x} = (1, x_1, \dots, x_D)^{\mathrm{T}}$：在$D$维特征的基础上，增加一个常数项1（用于表示截距项）



输入

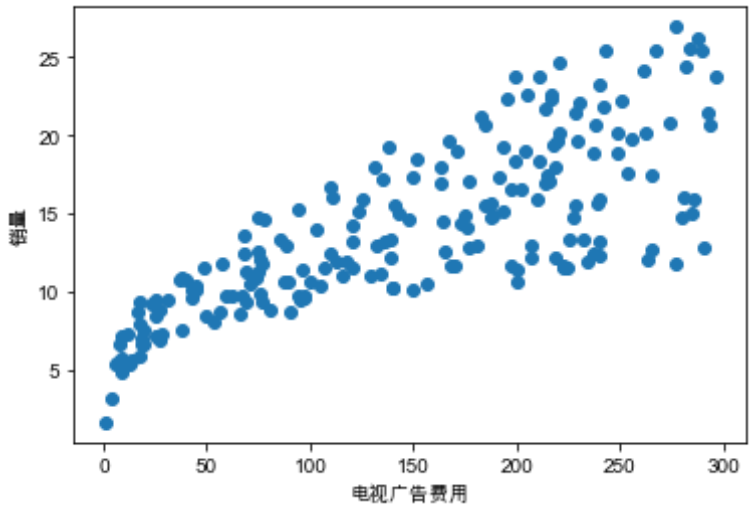
电视广告费用	广播广告费用	报纸广告费用	产品销量
230.1	37.8	69.2	22.1
44.5	39.3	45.1	10.4
17.2	45.9	69.3	9.3
151.5	41.3	58.5	18.5
180.8	10.8	58.4	12.9

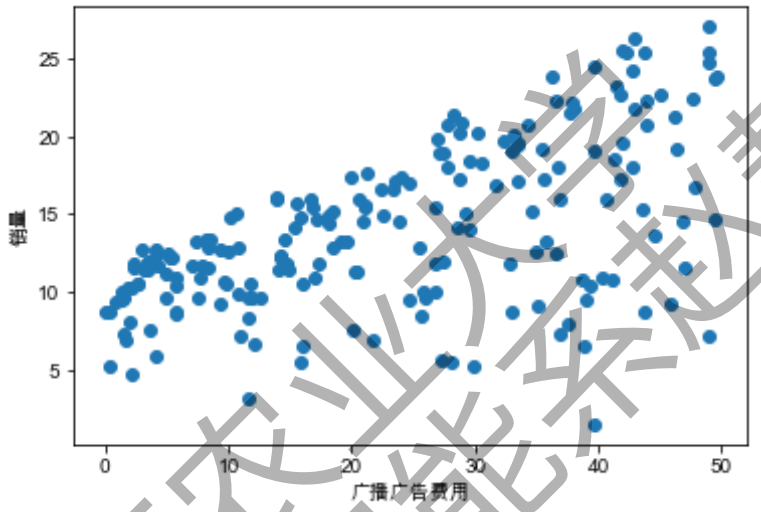
输出

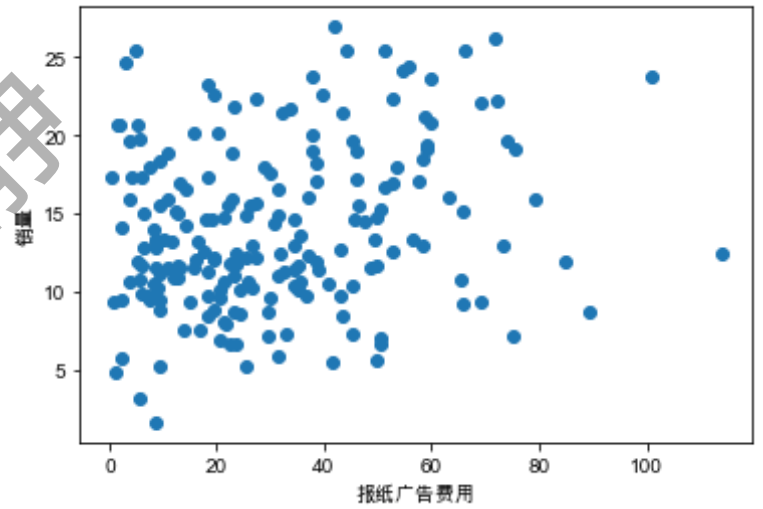
$\boldsymbol{x} = (x_1, x_2, x_3)$

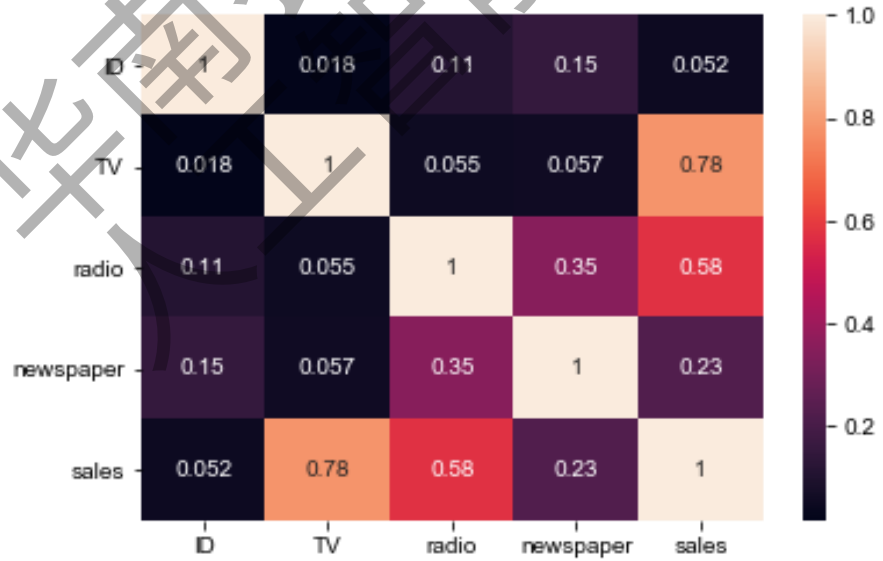
$y = f(\boldsymbol{x}) = f(x_1, x_2, x_3)$

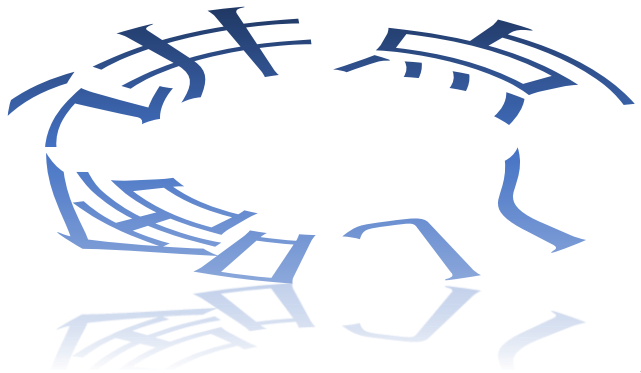
➢ 数据分析











- 线性回归模型
- 回归任务的损失函数
- 线性回归模型的正则项
- 线性回归的的优化算法
- 回归任务的模型性能评价

- 回忆：机器学习的三要素
  - 1. 函数集合$\{f_1, f_2, \dots\}$
  - 2. 目标函数$J(f)$：函数的好坏
  - 3. 优化算法：找到最佳函数

- 回归任务目标函数：

$$J(f, \lambda) = \sum_{i=1}^{N} L(y_i, \hat{y}_i) + \lambda R(f)$$

损失函数　　正则项

➢ 损失函数

1. L2损失，及预测残差的平方：

$$L(y,\hat{y}) = (y-\hat{y})^2 = r^2$$

其中 预测残差$r = y - \hat{y}$

- 训练集上所有训练数据的预测残差的平方和记为（Residual Sum of Squares，RSS）

$$\mathrm{RSS}(f) = \sum_{i=1}^{N} r_i^2 = \sum_{i=1}^{N} (y_i - \hat{y}_i)^2$$

✓ L2损失函数的概率解释*

- 最小L2损失函数等价于高斯白噪声下的极大似然。
- 在回归任务中，令模型预测值和真实值之间的差异为噪声ε。假设噪声ε_i相互独立，且ε_i的分布为0均值的正态分布，即$\varepsilon_i \sim N(0, \sigma^2)$，则
- $y_i = f(\boldsymbol{x}_i) + \varepsilon_i$
- $y_i|\boldsymbol{x}_i \sim N\left(f(\boldsymbol{x}_i), \sigma^2\right)$
- 所以 $$p(y_i|\boldsymbol{x}_i) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{\left(y_i - f(\boldsymbol{x}_i)\right)^2}{2\sigma^2}\right)$$

◆ 补充：极大似然估计

- 给定数据$\mathcal{D} = \{\boldsymbol{x}_i\}_{i=1}^{N}$，似然函数定义为数据出现的概率。
- 通常我们假定数据是独立同分布样本，因此所有数据出现的概率等于每个数据集中每个样本出现的概率相乘。
- log似然函数：

$$\ell(\boldsymbol{\theta}) = \ln p(\mathcal{D}|\boldsymbol{\theta}) = \sum_{i=1}^{N} \ln p(\boldsymbol{x}_i|\boldsymbol{\theta})$$

其中$\boldsymbol{\theta}$为分布的参数。

- 极大似然估计，即$\widehat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\operatorname{argmax}}\, \ell(\boldsymbol{\theta})$。

✓ L2损失函数的概率解释*

- 单个数据点的概率密度函数为：

$$p(y_i|\boldsymbol{x}_i) = \frac{1}{\sqrt{2\pi}\sigma}\exp\left(-\frac{\left(y_i - f(\boldsymbol{x}_i)\right)^2}{2\sigma^2}\right)$$

- 则log似然函数为

$$\ell(f) = \ln p(\mathcal{D}) = \sum_{i=1}^{N} \ln p(y_i|\boldsymbol{x}_i)$$

$$= \sum_{i=1}^{N} \ln\left[\frac{1}{\sqrt{2\pi}\sigma}\exp\left(-\frac{\left(y_i - f(\boldsymbol{x}_i)\right)^2}{2\sigma^2}\right)\right]$$

$$= -\frac{N}{2}\ln(2\pi) - N\ln\sigma - \sum_{i=1}^{N}\frac{\left(y_i - f(\boldsymbol{x}_i)\right)^2}{2\sigma^2}$$

✓ L2损失函数的概率解释*

log似然

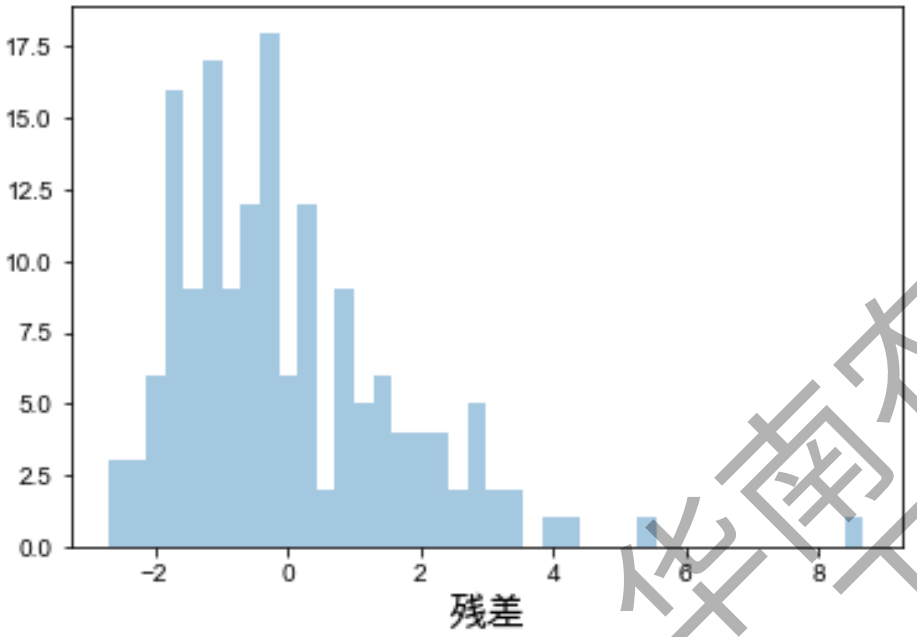
$$\ell(f) = -\frac{N}{2}\ln(2\pi) - N\ln\sigma - \sum_{i=1}^{N}\frac{\left(y_i - f(\boldsymbol{x}_i)\right)^2}{2\sigma^2}$$

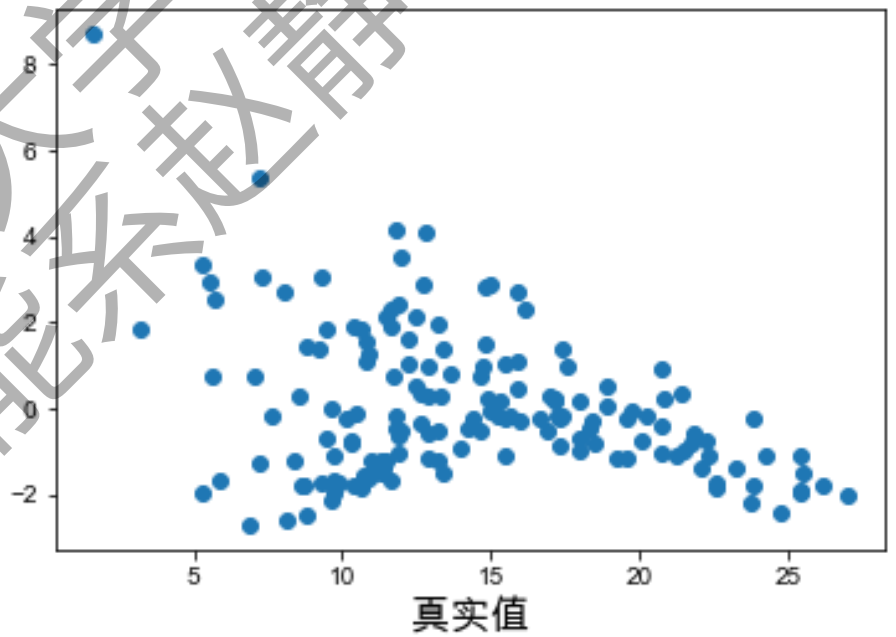
- 取最大值时，$-\sum_{i=1}^{N}\frac{\left(y_i - f(\boldsymbol{x}_i)\right)^2}{2\sigma^2}$取最大值，即$\sum_{i=1}^{N}\frac{\left(y_i - f(\boldsymbol{x}_i)\right)^2}{2\sigma^2}$取最小值。
- $\sum_{i=1}^{N}\left(y_i - f(\boldsymbol{x}_i)\right)^2 = \mathrm{RSS}(f)$

所以：极大似然估计等价于最小二乘（最小L2损失函数）。

L2损失是高斯白噪声假设下的极大似然（负log似然损失）

✓ 残差分布



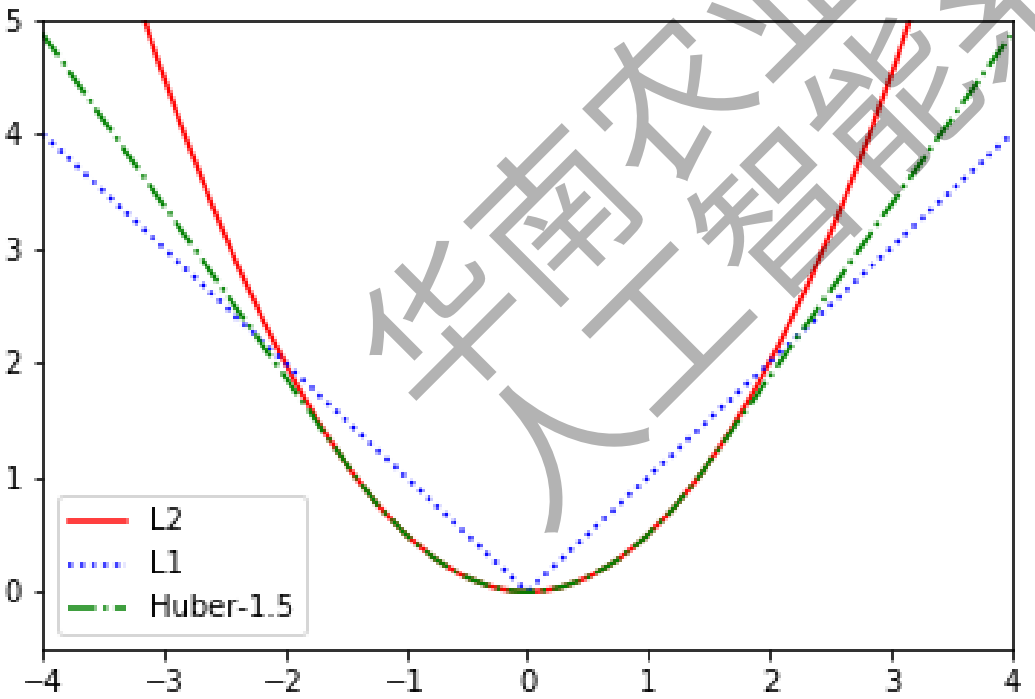


残差的分布并不符合0均值的正态分布
该模型（最小二乘回归模型）预测效果并不好

从真实值和残差的散点图来看，真实值较小和较大时，预测残差大多<0,其余情况残差大多>0。模型还没有完全建模y与x之间的关系，还有一部分关系残留在残差中

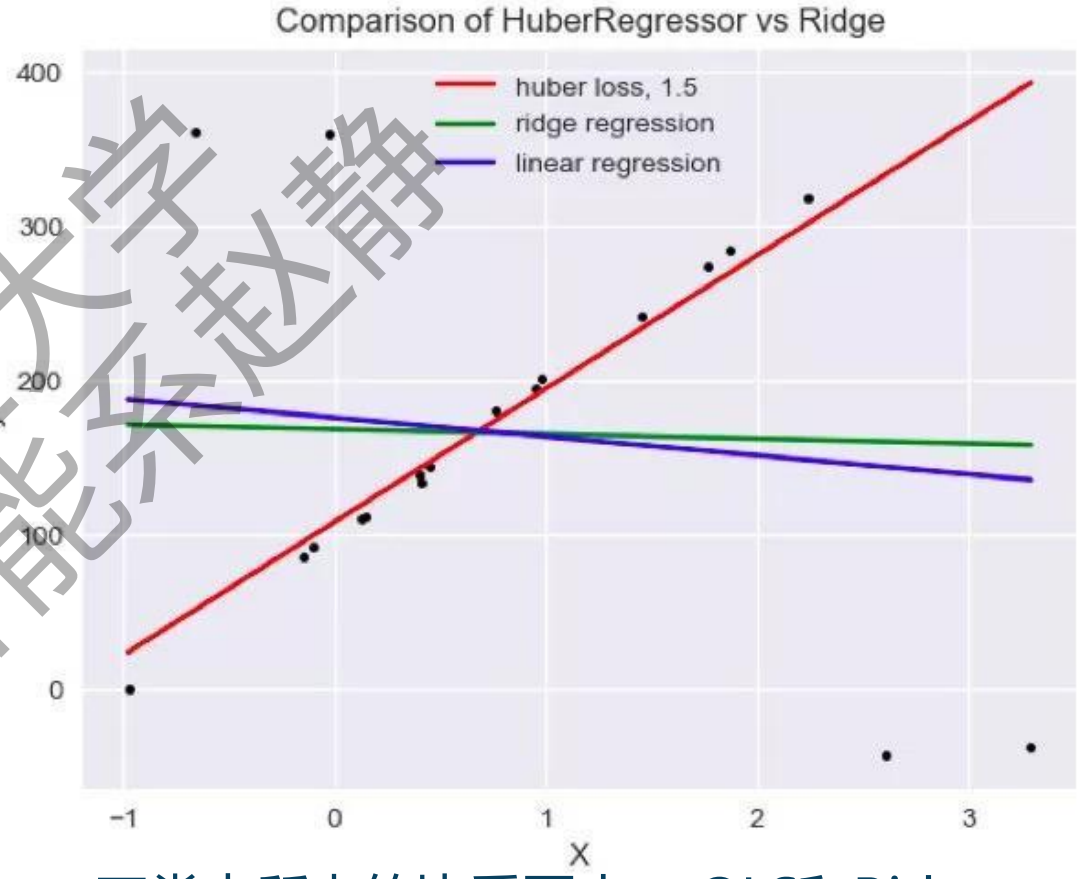
2. 胡伯损失函数（HuberRegressor）

$$L_\delta(y,\hat{y}) = \begin{cases} \frac{1}{2}(y-\hat{y})^2 & |r| \le \delta \\ \delta|y-\hat{y}| - \frac{1}{2}\delta^2 & otherwise \end{cases}$$



L2损失、L1损失和Huber损失比较
横轴：预测残差$r = y - \hat{y}$
纵轴：损失函数的值

左上方和右下出现了一些异常点
OLS和Ridge regression （L2损失）都不同程度上受到了异常点的影响，而Huber损失却没有受任何影响



正常点所占的比重更小， OLS和Ridge regression （L2损失）所决定出的回归模型几乎不工作，Huber损失性能依然完美

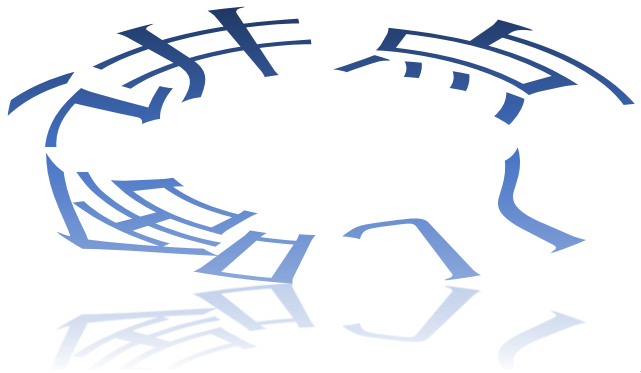
- sklearn中实现Huber损失的回归模型： HuberRegressor

Huber损失：

```
from sklearn.linear_model import HuberRegressor
huber = HuberRegressor()
huber.fit(X_train, y_train)
y_train_pred_huber = huber.predict(X_train)
```

L2损失：

```
from sklearn.linear_model import LinearRegression
lr = LinearRegression()
lr.fit(X_train, y_train)
y_train_pred_lr = lr.predict(X_train)
```



- 线性回归模型
- 回归任务的损失函数
- 线性回归模型的正则项
- 线性回归的的优化算法
- 回归任务的模型性能评价

- 回忆：机器学习的三要素
  - 1. 函数集合$\{f_1, f_2, \dots\}$
  - 2. 目标函数$J(f)$：函数的好坏
  - 3. 优化算法：找到最佳函数
- 回归任务目标函数：

$$J(f,\lambda) = \sum_{i=1}^{N} L(y_i, \hat{y}_i) + \lambda R(f)$$

损失函数　　正则项

抑制过拟合：在目标函数中加入正则项

➢ 正则项

$f(\boldsymbol{x}) = \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}$，正则项$R(f) = R(\boldsymbol{w})$

- L2正则：$R(\boldsymbol{w}) = \|\boldsymbol{w}\|_2^2 = \sum_{j=1}^{D} w_j^2$
- L1正则：$R(\boldsymbol{w}) = \|\boldsymbol{w}\|_1 = \sum_{j=1}^{D} |w_j|$

其中$\boldsymbol{w}$为模型参数，D为参数的维数。

注意：正则项不对截距项惩罚，因为截距项不影响模型的复杂度（函数的平滑程度：$\Delta\boldsymbol{x} \to \Delta y$）

✓ 无正则：最小二乘线性回归

$$J(\boldsymbol{w}) = \sum_{i=1}^{N} L(y_i, f(\boldsymbol{x}_i, \boldsymbol{w})) = \sum_{i=1}^{N} (y_i - f(\boldsymbol{x}_i, \boldsymbol{w}))^2 = \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}\|_2^2$$

- Scikit-Learn中实现的最小二乘线性回归为：LinearRegression

- *class* sklearn.linear_model.LinearRegression(*fit_intercept=True*, *normalize=False*, *copy_X=True*, *n_jobs=1*)

$$J(\boldsymbol{w}) = \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}\|_2^2$$

参数	说明	备注
fit_intercept	模型中是否包含截距项。	如果数据已经中心化（训练样本集的y的均值为0），可设置为False。
normalize	是否对输入特征X做归一化（减去均值，除以L2模），使得每个样本的模长为1。	对数据进行归一化会使得超参数学习更鲁棒，且几乎和样本数目没有关系。回归中用的不多，更多的时候用标准化
copy_X	是否拷贝数据X。设置为False时， X会被重写覆盖	
n_jobs	并行计算时使用CPU的数目。	-1表示使用所有的CPU资源（建议与设置为CPU核的数目相同，现在CPU基本上都支持超线程，-1可能对应的是超线程后的值，最好设置为CPU核的数目）。

- *class* sklearn.linear_model.LinearRegression(*fit_intercept=True*, *normalize=False*, *copy_X=True*, *n_jobs=1*)

$$J(\boldsymbol{w}) = \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}\|_2^2$$

属性	说明	备注
coef_	回归系数/权重，与特征维数相同。	如果是多任务回归，标签 y 为二维数组，则回归系数也是二维数组。
intercept_	截距项。	

方法	说明	备注
fit(X, y[, sample_weight])	模型训练。参数X, y为训练数据，也可以通过sample_weight设置每个样本的权重。	如果样本来自不同设备（如精度度不同），可设置样本权重。
predict(X)	使用训练好的模型进行预测，返回预测值	
score(X, y[, sample_weight])	评估模型预测性能，返回模型预测的R^2分数（预测值和真实值y的差异）。	请见回归模型评价指标部分。

✓ 岭回归：L2正则的线性回归

$$J(\boldsymbol{w},\lambda)=\sum_{i=1}^{N}L\big(y_i,f(\boldsymbol{x}_i,\boldsymbol{w})\big)+\lambda R(\boldsymbol{w})=\sum_{i=1}^{N}(y_i-\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i)^2+\lambda\sum_{j=1}^{D}w_j^2$$

$$=\|\boldsymbol{y}-\boldsymbol{X}\boldsymbol{w}\|_2^2+\lambda\|\boldsymbol{w}\|_2^2=(\boldsymbol{y}-\boldsymbol{X}\boldsymbol{w})^{\mathrm{T}}(\boldsymbol{y}-\boldsymbol{X}\boldsymbol{w})+\lambda\boldsymbol{w}^{\mathrm{T}}\boldsymbol{w}$$

- 岭回归的目标函数$J(\boldsymbol{w},\lambda)=\|\boldsymbol{y}-\boldsymbol{X}\boldsymbol{w}\|_2^2+\lambda\|\boldsymbol{w}\|_2^2$

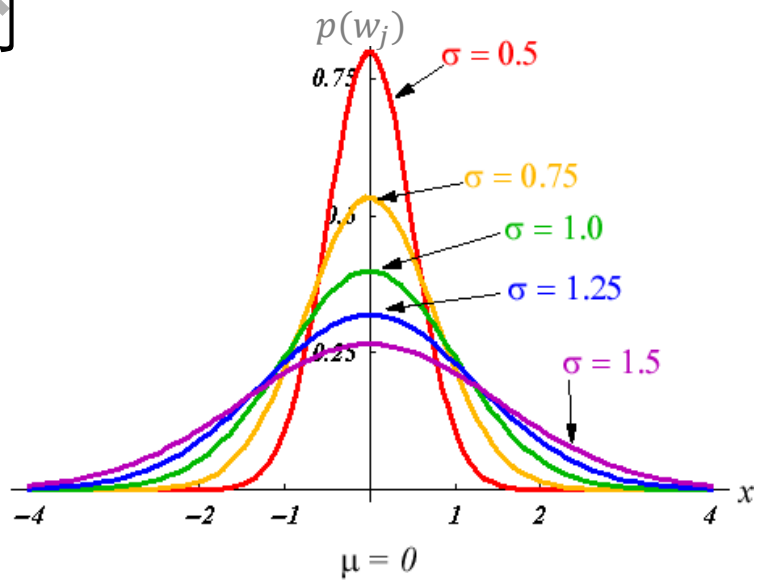
L2正则可视为参数先验分布为正态分布的贝叶斯估计。

■ 回忆：贝叶斯估计

- 数据的似然：$\ell(\boldsymbol{\theta})=\ln p(\mathcal{D}|\boldsymbol{\theta})=\sum\ln(p(y_i|\boldsymbol{x}_i,\boldsymbol{\theta}))$
- 参数的先验：$p(\boldsymbol{\theta})$
- 参数的后验：$p(\boldsymbol{\theta}|\mathcal{D})=p(\boldsymbol{\theta})p(\mathcal{D}|\boldsymbol{\theta})$
- 参数的最大后验估计：$\hat{\boldsymbol{\theta}}=\underset{\boldsymbol{\theta}}{\operatorname{argmax}}\,p(\boldsymbol{\theta}|\mathcal{D})$

- 岭回归的贝叶斯估计
  - 数据的似然：$$\ell(w)=\sum_{i=1}^{N}\ln p(y_i|\boldsymbol{x}_i)=-\frac{N}{2}ln(2\pi)-N\ln\sigma-\sum_{i=1}^{N}\frac{\left(y_i-f(\boldsymbol{x}_i,w)\right)^2}{2\sigma^2}$$ **(2-9)**
  - 参数的先验：$w_j\sim N(0,\tau^2)$，且w_j相互独立，则

$$p(\boldsymbol{w})=\prod_{j=1}^{D}p(w_j)=\prod_{j=1}^{D}\frac{1}{\sqrt{2\pi}\tau}\exp\left(-\frac{w_j^2}{2\tau^2}\right)$$



  - 参数的后验：

$$\ln p(\mathbf{w}|\mathcal{D})\propto\ln p(\mathbf{w})+\ln p(\mathcal{D}|\mathbf{w})$$

$$=-\frac{D}{2}\ln(2\pi)-D\ln\tau-\sum_{j=1}^{D}\frac{w_j^2}{2\tau^2}-\frac{N}{2}\ln(2\pi)-N\ln\sigma-\sum_{i=1}^{N}\frac{\left(y_i-f(\boldsymbol{x}_i)\right)^2}{2\sigma^2}$$

- 最大后验估计(Maximum a posteriori estimation, MAP)

$$\hat{\boldsymbol{w}} = \underset{\boldsymbol{w}}{\operatorname{argmax}} \log p(\boldsymbol{w}|\mathcal{D})$$

$$= \underset{\mathbf{w}}{\operatorname{argmin}} \left(\frac{D}{2}\ln(2\pi) + D\ln\tau + \sum_{j=1}^{D}\frac{w_j^2}{2\tau^2} + \frac{N}{2}\ln(2\pi) + N\ln\sigma + \sum_{i=1}^{N}\frac{\left(y_i - f(\boldsymbol{x}_i)\right)^2}{2\sigma^2} \right)$$

$$= \underset{\boldsymbol{w}}{\operatorname{argmin}} \left(\sum_{j=1}^{D}\frac{w_j^2}{2\tau^2} + \sum_{i=1}^{N}\frac{\left(y_i - f(\boldsymbol{x}_i)\right)^2}{2\sigma^2} \right)$$ (去掉与w无关的项)

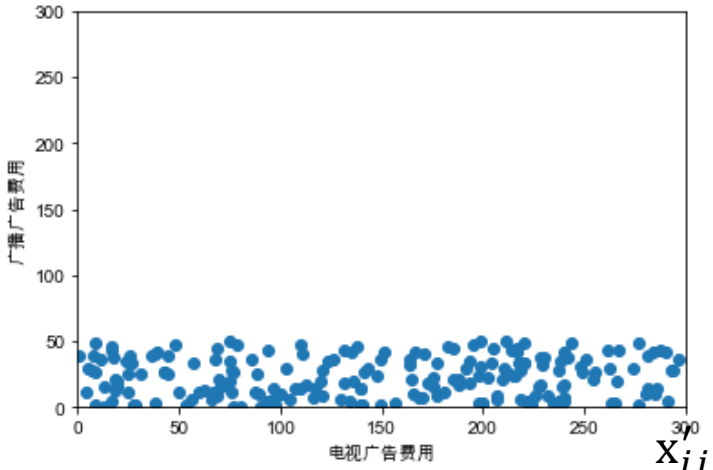
$$= \underset{\boldsymbol{w}}{\operatorname{argmin}} \left(\frac{\sigma^2}{\tau^2}\sum_{j=1}^{D} w_j^2 + \sum_{i=1}^{N}\left(y_i - \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i\right)^2 \right)$$ (乘以$2\sigma^2$) **(2-20)**

- 等价于岭回归的目标函数:

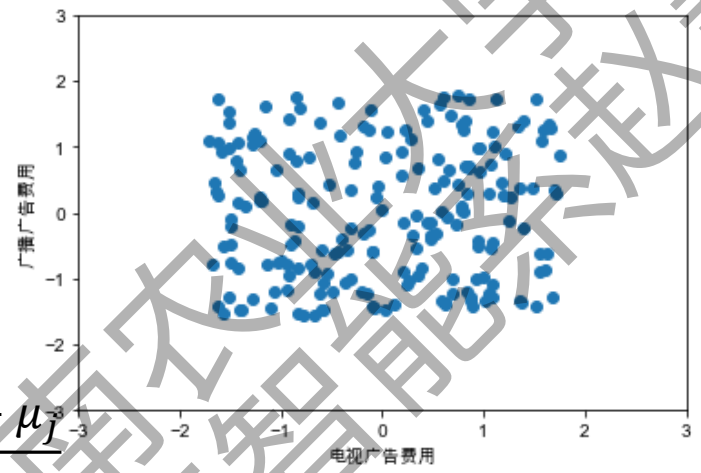
$$J(\boldsymbol{w}, \lambda) = (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w})^{\mathrm{T}}(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}) + \lambda \boldsymbol{w}^{\mathrm{T}}\boldsymbol{w}$$

- 数据标准化

- sklearn中的实现： StandardScaler



$$x_{ij} = \frac{x_{ij} - \mu_j}{\sigma_j}$$



$$\mu_j = \frac{1}{N}\sum_{i=0}^{N} x_{ij}$$

$$\sigma_j = \sqrt{\frac{1}{N-1}\left(\sum_{i=0}^{N} x_{ij} - \mu_j\right)^2}$$

- 亦可采用MinMaxScaler 将特征缩放到[0,1]。

```
# 数据标准化
from sklearn.preprocessing import StandardScaler
# 构造输入特征的标准化器
ss_X = StandardScaler()
# 分别对训练和测试数据的特征进行标准化处理
X_train = ss_X.fit_transform(X_train)
X_test = ss_X.transform(X_test)
```

sklearn中的岭回归：Ridge

class sklearn.linear_model.Ridge(*alpha=1.0*, *fit_intercept=True*, *normalize=False*, *copy_X=True*, *max_iter=None*, *tol=0.001*, *solver='auto'*, *random_state=None*)

参数说明：

- alpha: 目标函数$J(\mathbf{w},\lambda)=(\boldsymbol{y}-\boldsymbol{X}\boldsymbol{w})^{\mathrm{T}}(\boldsymbol{y}-\boldsymbol{X}\boldsymbol{w})+\lambda\boldsymbol{w}^{\mathrm{T}}\boldsymbol{w}$中的$\lambda$
- fit_intercept、normalize、copy_X: 意义同LinearRegression
- 其余参数与优化计算有关（请见优化求解部分）。

属性同LinearRegression

常用方法同LinearRegression

✓ Lasso：L1正则的线性回归

- 当损失函数取L2损失：$L\big(y_i, f(\boldsymbol{x}_i; \boldsymbol{w})\big) = (y_i - f(\boldsymbol{x}_i, \boldsymbol{w}))^2$
- 正则项取L1正则：$R(\boldsymbol{w}) = \sum_{j=1}^{D} |w_j|$
- 线性回归模型：$f(\boldsymbol{x}, \boldsymbol{w}) = \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} = \sum_{j=0}^{D} w_j x_j \quad (x_0 = 1)$
- 得到Lasso (Least Absolute Shrinkage and Selection Operator)的目标函数为：

$$J(\boldsymbol{w}, \lambda) = \sum_{i=1}^{N} L\big(y_i, f(\boldsymbol{x}_i, \boldsymbol{w})\big) + \lambda R(\boldsymbol{w}) = \sum_{i=1}^{N} (y_i - \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i)^2 + \lambda \sum_{j=1}^{D} |w_j|$$

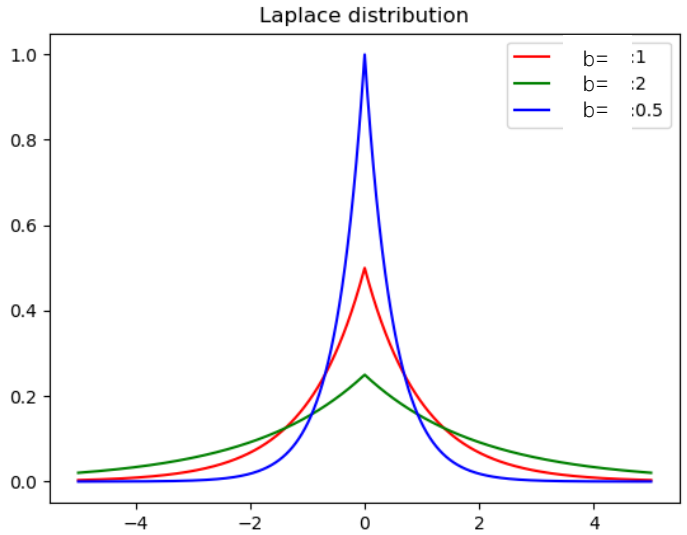
$$= \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}\|^2 + \lambda \|\boldsymbol{w}\|_1$$

- 贝叶斯估计
  - 数据的似然：

$$\ell(f)=\sum_{i=1}^{N}\ln p(y_i|\boldsymbol{x}_i)-\frac{N}{2}\ln(2\pi)-N\ln\sigma-\sum_{i=1}^{N}\frac{\left(y_i-f(\boldsymbol{x}_i)\right)^2}{2\sigma^2}$$

  - 参数的先验：$w_j\sim\text{Laplace}(0,b)$，且$w_j$相互独立，则

$$p(\boldsymbol{w})=\prod_{j=1}^{D}p(w_j)=\prod_{j=1}^{D}\frac{1}{2b}\exp\left(-\frac{|w_j|}{b}\right)$$



  - 参数的后验：

$$\ln p(\mathbf{w}|\mathcal{D})\propto\ln p(\mathbf{w})+\ln p(\mathcal{D}|\mathbf{w})$$

$$=-D\ln(2b)-\sum_{j=1}^{D}\frac{|w_j|}{b}-\frac{N}{2}\ln(2\pi)-N\ln\sigma-\sum_{i=1}^{N}\frac{\left(y_i-f(\boldsymbol{x}_i)\right)^2}{2\sigma^2}$$

- 最大后验估计(Maximum a posteriori estimation, MAP)

$$\hat{\boldsymbol{w}} = \underset{\boldsymbol{w}}{\operatorname{argmax}} \ln p(\boldsymbol{w}|\mathcal{D})$$

$$= \underset{\mathbf{w}}{\operatorname{argmin}} \left(+D\ln(2b) + \sum_{j=1}^{D} \frac{|w_j|}{b} + \frac{N}{2}\ln(2\pi) + N\ln\sigma + \sum_{i=1}^{N} \frac{\left(y_i - f(\boldsymbol{x}_i)\right)^2}{2\sigma^2} \right)$$

$$= \underset{\boldsymbol{w}}{\operatorname{argmin}} \left(+ \sum_{j=1}^{D} \frac{|w_j|}{b} + \sum_{i=1}^{N} \frac{\left(y_i - f(\boldsymbol{x}_i)\right)^2}{2\sigma^2} \right) \text{(去掉与}\boldsymbol{w}\text{无关的项)}$$

$$= \underset{\boldsymbol{w}}{\operatorname{argmin}} \left(\frac{2\sigma^2}{b} \sum_{j=1}^{D} |w_j| + \sum_{i=1}^{N} \left(y_i - \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i\right)^2 \right) \text{(乘以}2\sigma^2\text{)}$$

- 等价于Lasso回归的目标函数（去掉负号，取最大变成取最小）：

$$J(\boldsymbol{w}, \lambda) = (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w})^{\mathrm{T}}(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}) + \lambda \sum_{j=1}^{D} |w_j|$$

✓ 弹性网络：L1正则 + L2正则

- 正则项还可以为L1正则和L2正则的线性组合：

$$R(\boldsymbol{w},\rho)=\sum_{j=1}^{D}\left(\rho|w_j|+\frac{(1-\rho)}{2}w_j^2\right)$$

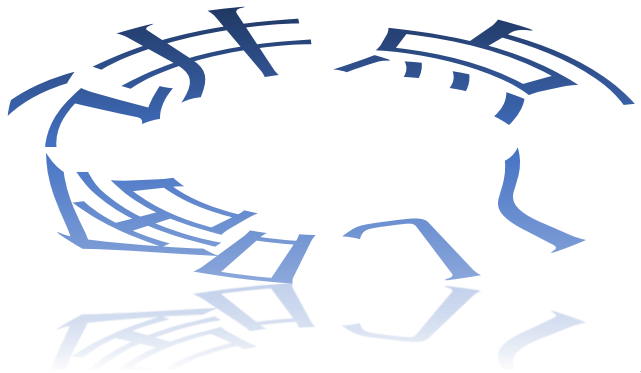
- 得到弹性网络的目标函数：

$$J(\boldsymbol{w},\lambda,\rho)=\frac{1}{2N}\|\boldsymbol{y}-\boldsymbol{X}\boldsymbol{w}\|^2+\left(\lambda_1\|\boldsymbol{w}\|_1+\lambda_2\|\boldsymbol{w}\|_2^2\right),\quad 0\le\rho\le 1.$$

- Sklearn中实现的弹性网络为：ElasticNet

class sklearn.linear_model.ElasticNet(*alpha=1.0*, *l1_ratio=0.5*, *fit_intercept=True*, *normalize=False*, *precompute=False*, *max_iter=1000*, *copy_X=True*, *tol=0.0001*, *warm_start=False*, *positive=False*, *random_state=None*, *selection='cyclic'*)

alpha参数为上述目标函数中的λ，l1_ratio为上述目标函数中的ρ



- 线性回归模型
- 回归任务的损失函数
- 线性回归模型的正则项
- **线性回归的的优化算法**
  - **解析求解（正规方程组）**
  - **梯度下降**
  - **坐标轴下降**
- 回归任务的模型性能评价

目标函数最优值

- 给定超参数λ的情况下，目标函数最优解：$\hat{\boldsymbol{w}} = \underset{\boldsymbol{w}}{\operatorname{argmin}} J(\boldsymbol{w}, \lambda)$
- 根据优化理论，函数极值点只能在边界点、不可导点、临界点（导数为0的点）
- 临界点：一阶偏导数组成的向量（亦被称为梯度）为0向量：

$$g(\boldsymbol{w}) = \frac{\partial J(\boldsymbol{w}, \lambda)}{\partial \boldsymbol{w}} = \begin{bmatrix} \frac{\partial J}{\partial w_1} \\ \frac{\partial J}{\partial w_2} \\ \vdots \\ \frac{\partial J}{\partial w_D} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

- 给定超参数λ的情况下，目标函数最优解：$\hat{\boldsymbol{w}} = \underset{\boldsymbol{w}}{\operatorname{argmin}} J(\boldsymbol{w})$
- 一阶偏导数组成的向量（梯度）为0向量：$\frac{\partial J(\boldsymbol{w},\lambda)}{\partial \boldsymbol{w}} = \boldsymbol{0}$
- 若二阶海森（Hessian）矩阵

$$\boldsymbol{H} = \frac{\partial^2 J(\boldsymbol{w},\lambda)}{\partial \boldsymbol{w} \partial \boldsymbol{w}^{\mathrm{T}}} = \begin{bmatrix} \frac{\partial^2 J}{\partial w_1 \partial w_1} & \frac{\partial^2 J}{\partial w_1 \partial w_2} & \cdots & \frac{\partial^2 J}{\partial w_1 \partial w_D} \\ \frac{\partial^2 J}{\partial w_2 \partial w_1} & \frac{\partial^2 f}{\partial w_2 \partial w_2} & \cdots & \frac{\partial^2 J}{\partial w_2 \partial w_D} \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\partial^2 J}{\partial w_D \partial w_1} & \frac{\partial^2 J}{\partial w_D \partial w_2} & \vdots & \frac{\partial^2 J}{\partial w_D \partial w_D} \end{bmatrix}$$

为正定矩阵，则在临界点目标函数取最小值

➢ 解析求解：正规方程组

- 最小二乘（Ordinary Linear Square, OLS）目标函数（矩阵形式）为：

$$J(\boldsymbol{w}) = \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}\|_2^2 = (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w})^{\mathrm{T}}(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w})$$

$$\left[\frac{\partial(\boldsymbol{y}^{\mathrm{T}}\boldsymbol{b})}{\partial \boldsymbol{y}} = \boldsymbol{b}\right]$$

$$\left[\frac{\partial(\boldsymbol{y}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{y})}{\partial \boldsymbol{y}} = (A^{\mathrm{T}} + \boldsymbol{A})\boldsymbol{y}\right]$$

梯度为：

$$\frac{\partial J}{\partial \boldsymbol{w}} = -2\boldsymbol{X}^{\mathrm{T}}(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}) = -2\boldsymbol{X}^{\mathrm{T}}\boldsymbol{y} + 2\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}\boldsymbol{w} = \boldsymbol{0}$$

从而：

$$\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}\boldsymbol{w} = \boldsymbol{X}^{\mathrm{T}}\boldsymbol{y} \rightarrow \hat{\boldsymbol{w}}_{OLS} = (\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X})^{-1}\boldsymbol{X}^{\mathrm{T}}\boldsymbol{y}$$

正规方程组(Normal Equations)求解，奇异值分解求解

- 岭回归解析求解

岭回归的目标函数为:

$$J(\boldsymbol{w},\lambda)=(\boldsymbol{y}-\boldsymbol{X}\boldsymbol{w})^{\mathrm{T}}(\boldsymbol{y}-\boldsymbol{X}\boldsymbol{w})+\lambda\boldsymbol{w}^{\mathrm{T}}\boldsymbol{w}$$

梯度:

$$\frac{\partial J(\boldsymbol{w},\lambda)}{\partial \boldsymbol{w}}=-2\boldsymbol{X}^{\mathrm{T}}\boldsymbol{y}+2(\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X})\boldsymbol{w}+2\lambda\boldsymbol{w}=\boldsymbol{0}$$

$$\hat{\mathbf{w}}_{Ridge}=(\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}+\lambda\boldsymbol{I})^{-1}\boldsymbol{X}^{\mathrm{T}}\boldsymbol{y}$$

➢ 梯度下降

- 梯度下降——最小二乘回归

最小二乘的目标函数：$J(\boldsymbol{w},\lambda)=(\boldsymbol{y}-\boldsymbol{X}\boldsymbol{w})^{\mathrm{T}}(\boldsymbol{y}-\boldsymbol{X}\boldsymbol{w})$

目标函数的梯度：$$g(\boldsymbol{w})=\frac{\partial J(\boldsymbol{w},\lambda)}{\partial \boldsymbol{w}}=-2\boldsymbol{X}^{\mathrm{T}}\boldsymbol{y}+2(\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X})\boldsymbol{w}$$

$$=-2\boldsymbol{X}^{\mathrm{T}}(\boldsymbol{y}-\boldsymbol{X}\boldsymbol{w})$$ 预测残差r

参数更新：$$\boldsymbol{w}^{(t+1)}=\boldsymbol{w}^{(t)}-\eta g(\boldsymbol{w}^{(t)})$$

$$=\boldsymbol{w}^{(t)}+2\eta\boldsymbol{X}^{\mathrm{T}}(\boldsymbol{y}-\boldsymbol{X}\boldsymbol{w}^{(t)})$$

参数的更新量与输入与预测残差的相关性有关。

- 梯度下降——岭回归
  - 岭回归的目标函数：$J(\boldsymbol{w},\lambda)=(\boldsymbol{y}-\boldsymbol{X}\boldsymbol{w})^{\mathrm{T}}(\boldsymbol{y}-\boldsymbol{X}\boldsymbol{w})+\lambda\boldsymbol{w}^{\mathrm{T}}\boldsymbol{w}$
  - 目标函数的梯度：

$$g(\boldsymbol{w})=\frac{\partial J(\boldsymbol{w},\lambda)}{\partial \boldsymbol{w}}=-2\boldsymbol{X}^{\mathrm{T}}\boldsymbol{y}+2(\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X})\boldsymbol{w}+2\lambda\boldsymbol{w}$$

  - 参数更新：

$$\begin{aligned}\boldsymbol{w}^{(t+1)}&=\boldsymbol{w}^{(t)}-\eta g\left(\boldsymbol{w}^{(t)}\right)\\&=\boldsymbol{w}^{(t)}+2\eta\boldsymbol{X}^{\mathrm{T}}\left(\boldsymbol{y}-\boldsymbol{X}\boldsymbol{w}^{(t)}\right)-2\eta\lambda\boldsymbol{w}^{(t)}\end{aligned}$$

➢ 梯度下降的高阶话题

- ✓ 梯度的计算：计算梯度时需用到每个训练样本，当样本数目很多时，计算费用高：随机梯度下降、小批量梯度下降
- ✓ 动量梯度下降：比梯度更好的移动方向
- ✓ 学习率：
  - 太小，收敛慢
  - 学习率太大，不收敛
  - 各参数公用一个学习率：特征缩放

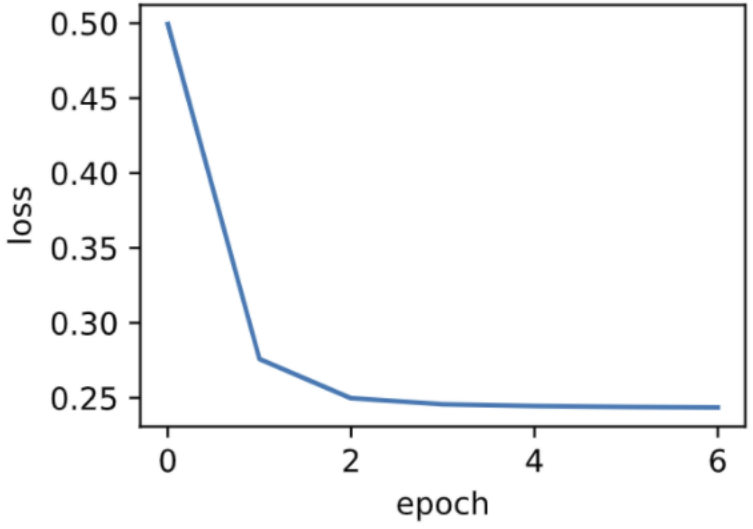
✓ 梯度计算：Mini-batch SGD

- 机器学习算法的目标函数大多为：$J(\boldsymbol{\theta}) = \sum_{i=1}^{N} L_i(y_i, \hat{y}_i) + \lambda R(\boldsymbol{\theta})$
- 梯度：$\frac{\partial J(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = \sum_{i=1}^{N} \frac{\partial L_i(\boldsymbol{\theta})}{\partial(\boldsymbol{\theta})} + \lambda \frac{\partial R(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}$

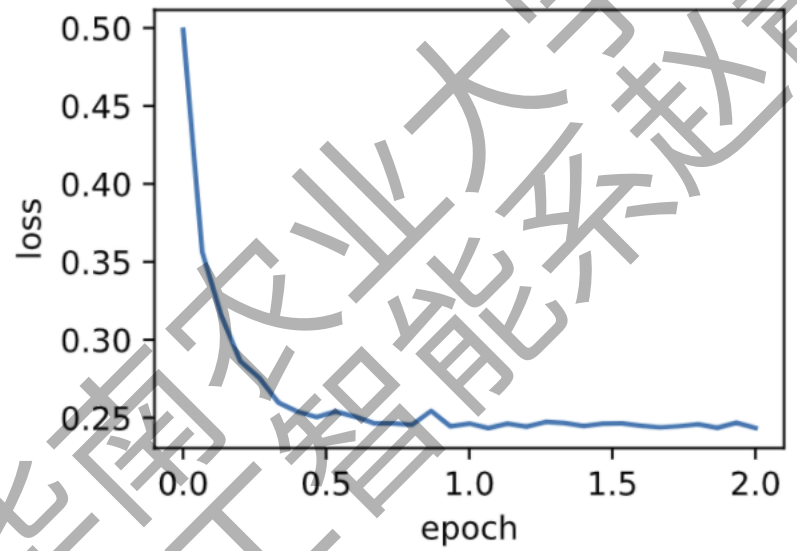
  缺点：对N个样本求和：当N很大时，计算慢
- 随机梯度下降（Stochastic Gradient Descent, SGD）：每次只计算一个样本上梯度

  缺点：没有有效利用CPU和GPU的计算效率
- Mini-batch SGD：每次只随机送入少量样本

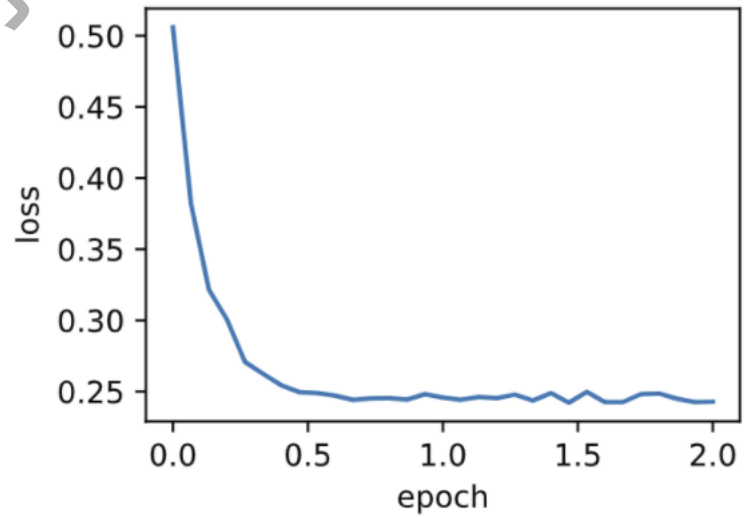
例：小批处理梯度下降



梯度下降
$(bathsize = N = 1500)$



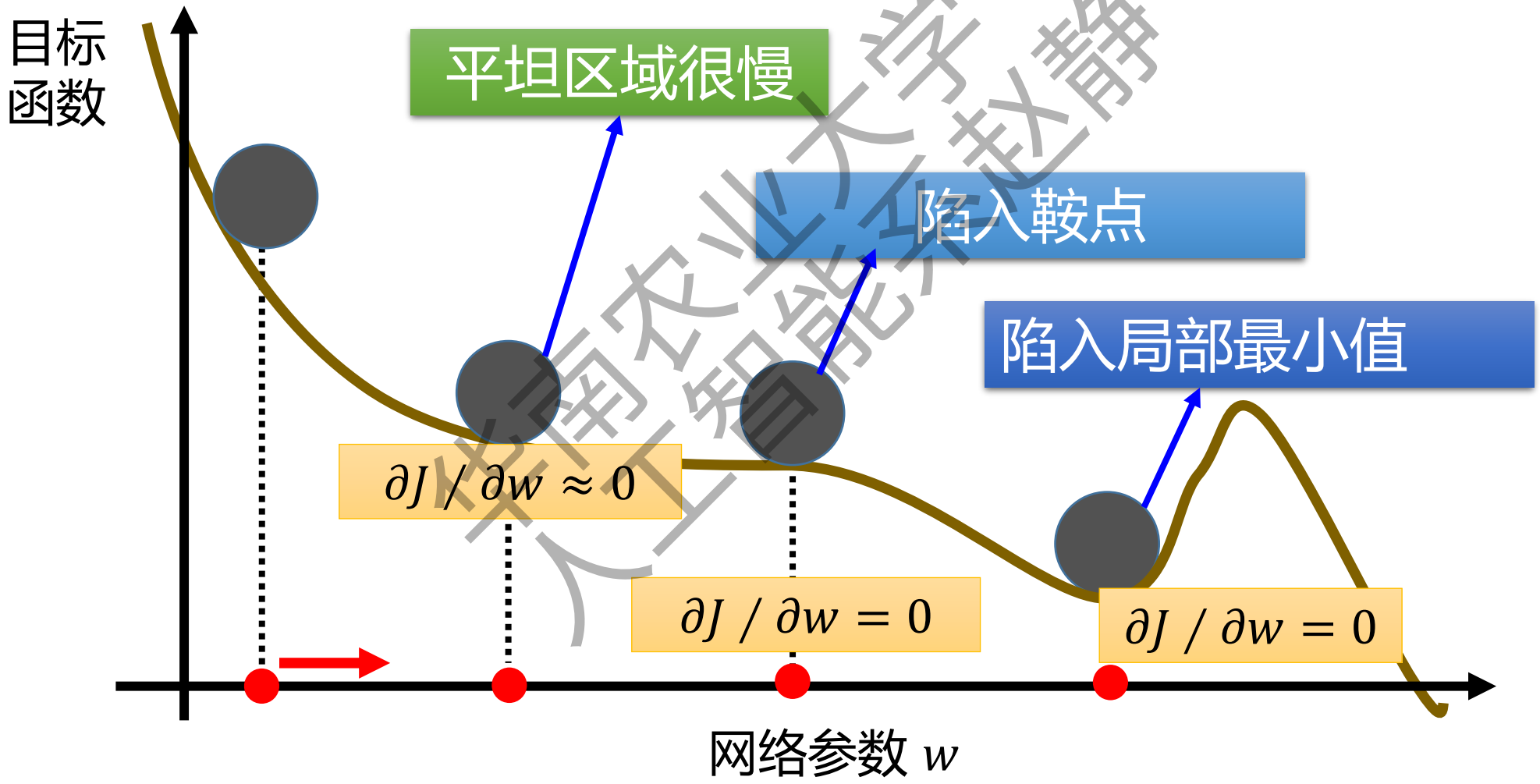
随机梯度下降
$(bathsize = 1)$



小批量梯度下降
$(bathsize = 10)$

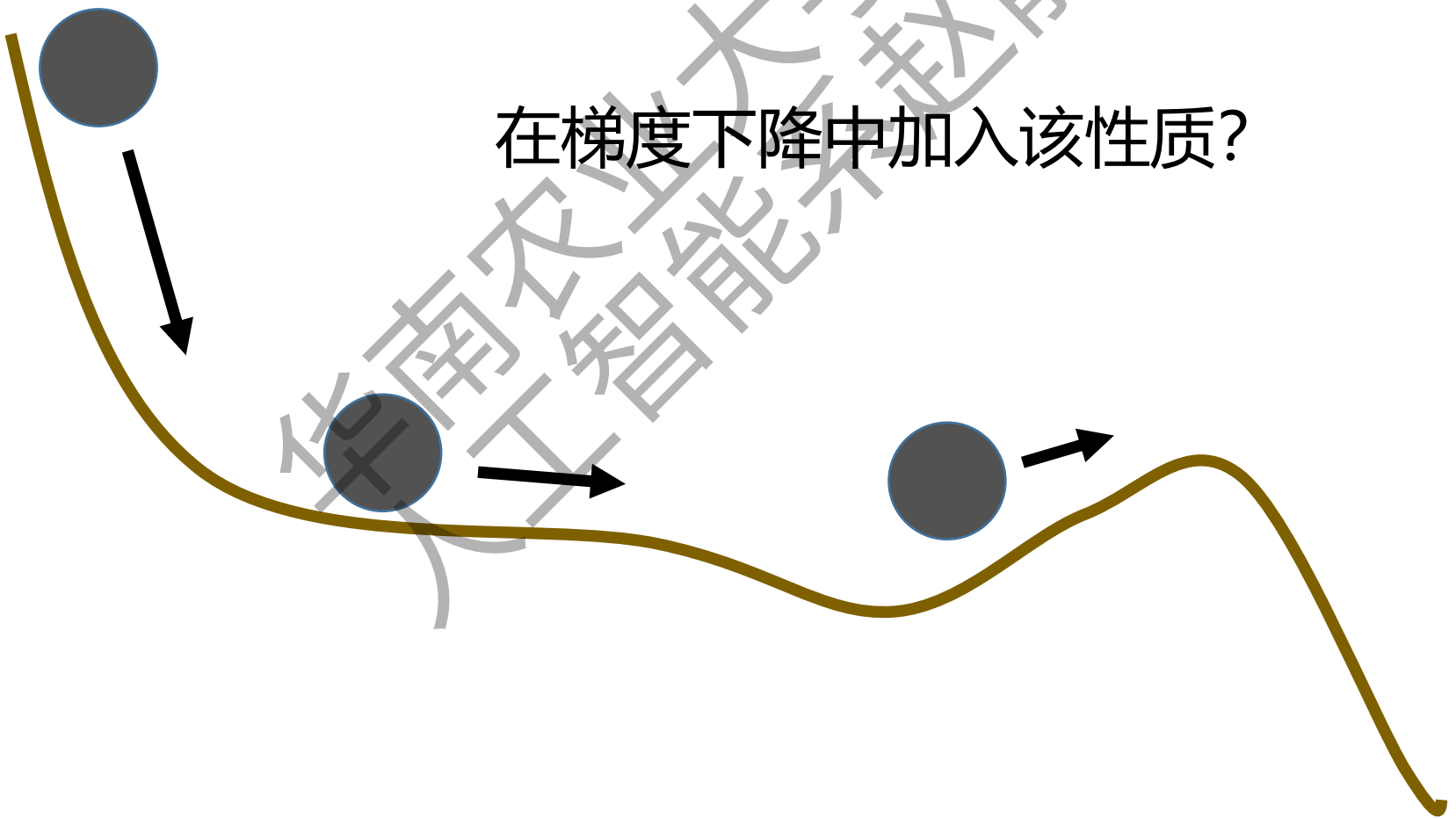
https://tangshusen.me/Dive-into-DL-PyTorch/#/chapter07_optimization/7.3_minibatch-sgd

✓ 动量梯度下降

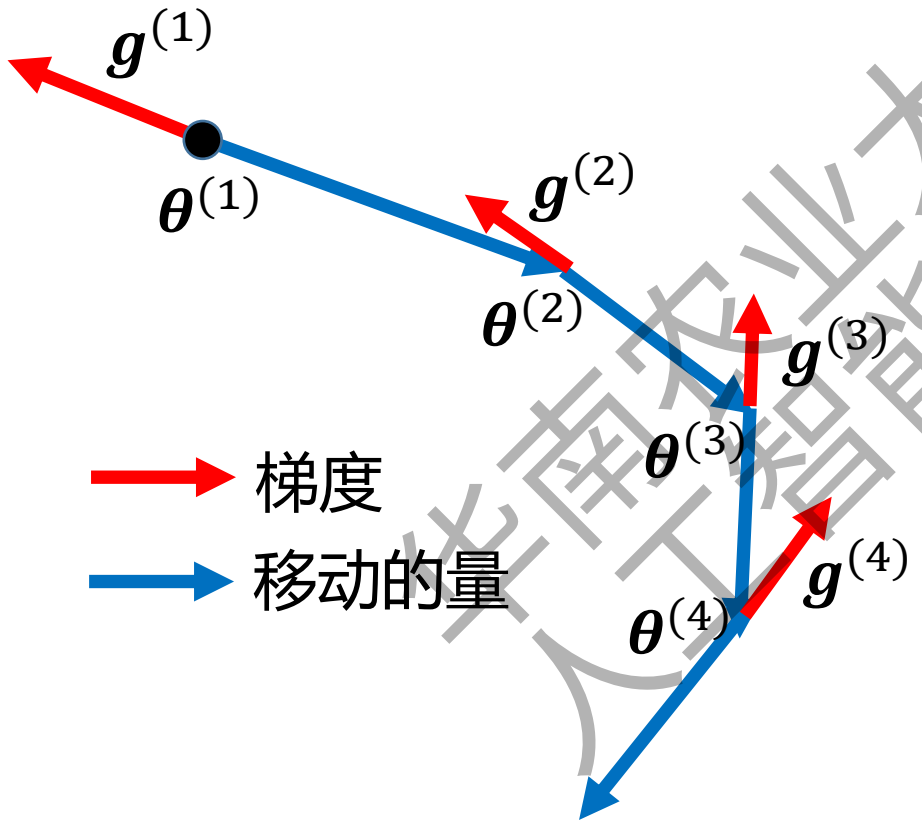


物理世界中 ……

动量/惯性



- 梯度下降



初始位置$\boldsymbol{\theta}^{(1)}$

计算$\boldsymbol{\theta}^{(1)}$处的梯度$\boldsymbol{g}^{(1)}$

移到$\boldsymbol{\theta}^{(2)} = \boldsymbol{\theta}^{(1)} - \eta \boldsymbol{g}^{(1)}$

计算$\boldsymbol{\theta}^{(2)}$处的梯度$\boldsymbol{g}^{(2)}$

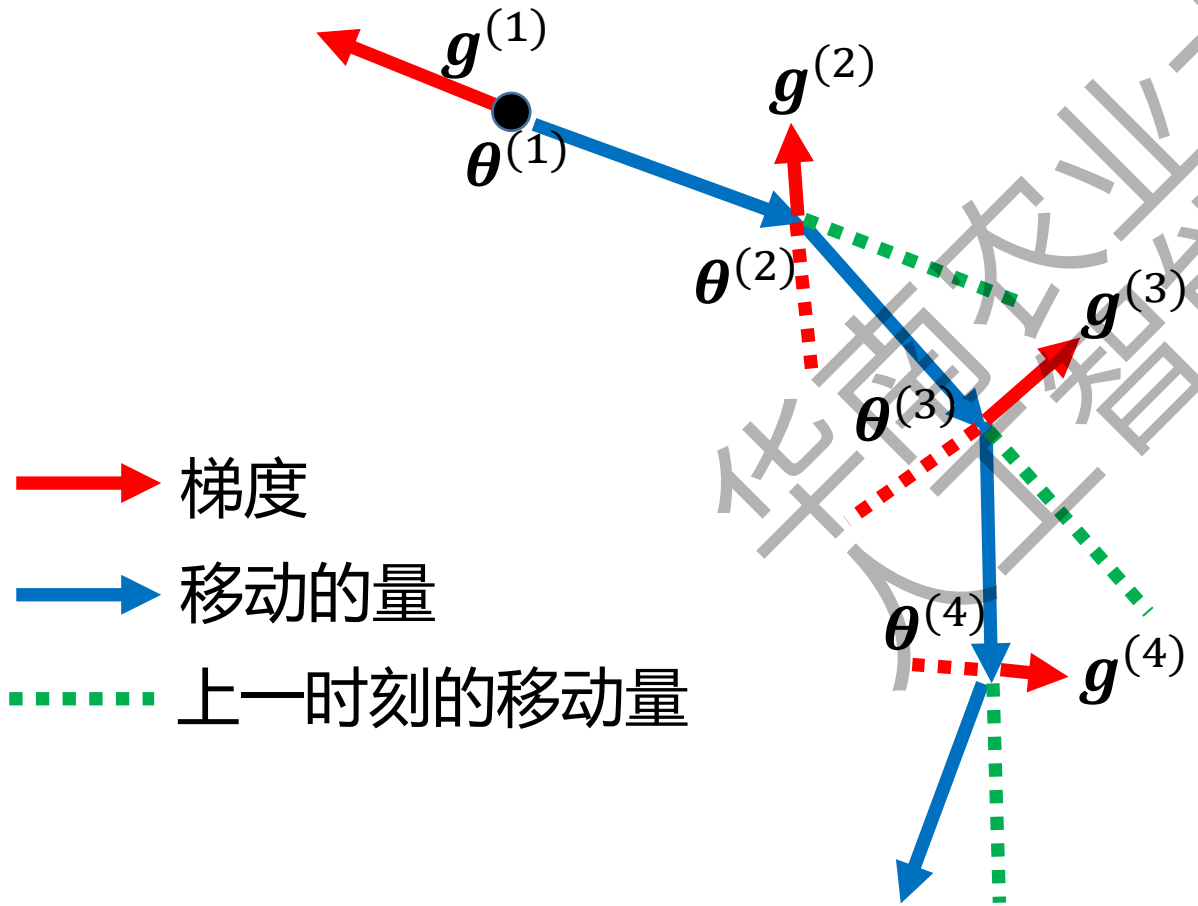
移到$\boldsymbol{\theta}^{(3)} = \boldsymbol{\theta}^{(2)} - \eta \boldsymbol{g}^{(2)}$

$\vdots$

迭代，直到$\boldsymbol{g}^{(t)} \approx 0$

- 动量梯度下降

动量：上一时刻的动量减去当前的梯度



初始动量$\boldsymbol{v}^{(0)} = \mathbf{0}$

初始位置$\boldsymbol{\theta}^{(1)}$

计算$\boldsymbol{\theta}^{(1)}$处的梯度$\boldsymbol{g}^{(1)}$

动量$\boldsymbol{v}^{(1)} = \rho\boldsymbol{v}^{(0)} - \eta\boldsymbol{g}^{(1)}$

移到$\boldsymbol{\theta}^{(2)} = \boldsymbol{\theta}^{(1)} + \boldsymbol{v}^{(1)}$

计算$\boldsymbol{\theta}^{(2)}$处的梯度$\boldsymbol{g}^{(2)}$

动量$\boldsymbol{v}^{(2)} = \rho\boldsymbol{v}^{(1)} - \eta\boldsymbol{g}^{(2)}$

移到$\boldsymbol{\theta}^{(3)} = \boldsymbol{\theta}^{(2)} + \boldsymbol{v}^{(2)}$

移动量不仅与梯度有关，还与前一时刻的移动量有关。

动量梯度下降

- 事实上，$\boldsymbol{v}^{(t)}$ 为之前所有梯度 $\boldsymbol{g}^{(0)},\boldsymbol{g}^{(1)},\dots\boldsymbol{g}^{(t)}$的加权和

$$\boldsymbol{v}^{(0)} = 0$$

$$\boldsymbol{v}^{(1)} = \rho\boldsymbol{v}^{(0)} - \eta\boldsymbol{g}^{(1)} = -\eta\boldsymbol{g}^{(1)}$$

$$\boldsymbol{v}^{(2)} = \rho\boldsymbol{v}^{(1)} - \eta\boldsymbol{g}^{(2)} = -\rho\eta\boldsymbol{g}^{(1)} - \eta\boldsymbol{g}^{(2)}$$

$$\vdots$$

$$\boldsymbol{v}^{(t)} = \rho\boldsymbol{v}^{(t-1)} - \eta\boldsymbol{g}^{(t)}$$

$$= \rho\left(-\rho\boldsymbol{v}^{(t-2)} - \eta\boldsymbol{g}^{(t-1)}\right) - \eta\boldsymbol{g}^{(t)}$$

$$= -\eta\sum_{j=1}^{t}\rho^{t-j}\,\boldsymbol{g}^{(j)}$$

初始动量$\boldsymbol{v}^{(0)} = \boldsymbol{0}$

初始位置$\boldsymbol{\theta}^{(1)}$

计算$\boldsymbol{\theta}^{(1)}$处的梯度$\boldsymbol{g}^{(1)}$

动量$\boldsymbol{v}^{(1)} = \rho\boldsymbol{v}^{(0)} - \eta\boldsymbol{g}^{(1)}$

移到$\boldsymbol{\theta}^{(2)} = \boldsymbol{\theta}^{(1)} + \boldsymbol{v}^{(1)}$

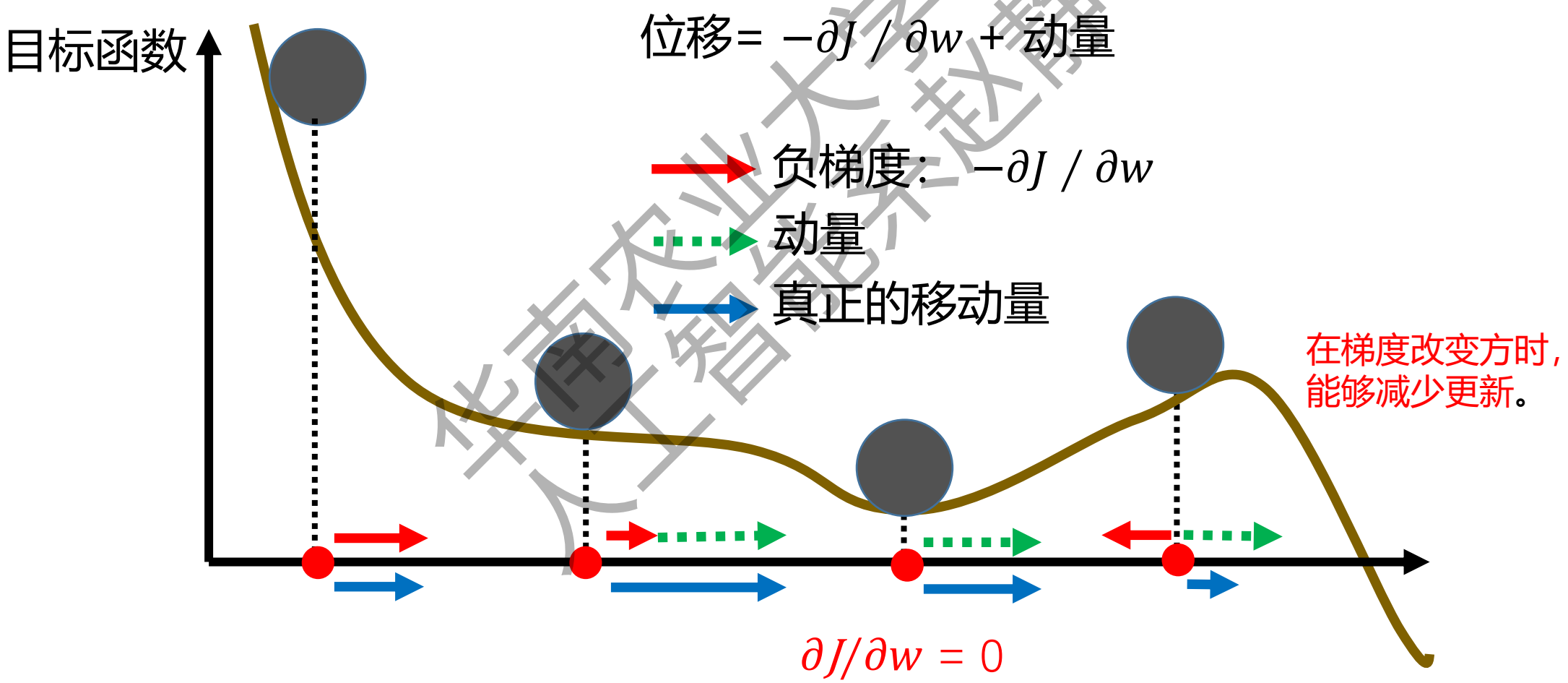
计算$\boldsymbol{\theta}^{(2)}$处的梯度$\boldsymbol{g}^{(2)}$

动量$\boldsymbol{v}^{(2)} = \rho\boldsymbol{v}^{(1)} - \eta\boldsymbol{g}^{(2)}$

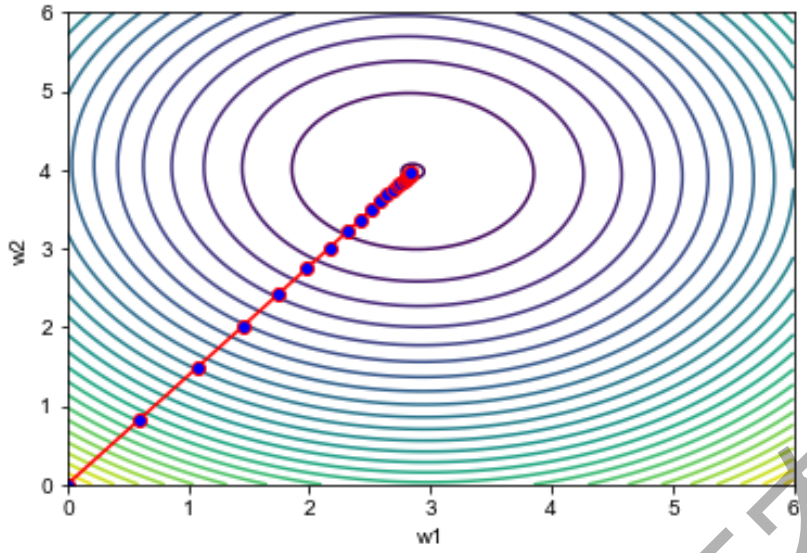
移到$\boldsymbol{\theta}^{(3)} = \boldsymbol{\theta}^{(2)} + \boldsymbol{v}^{(2)}$

移动量不仅与梯度有关，还与前一时刻的移动量有关。

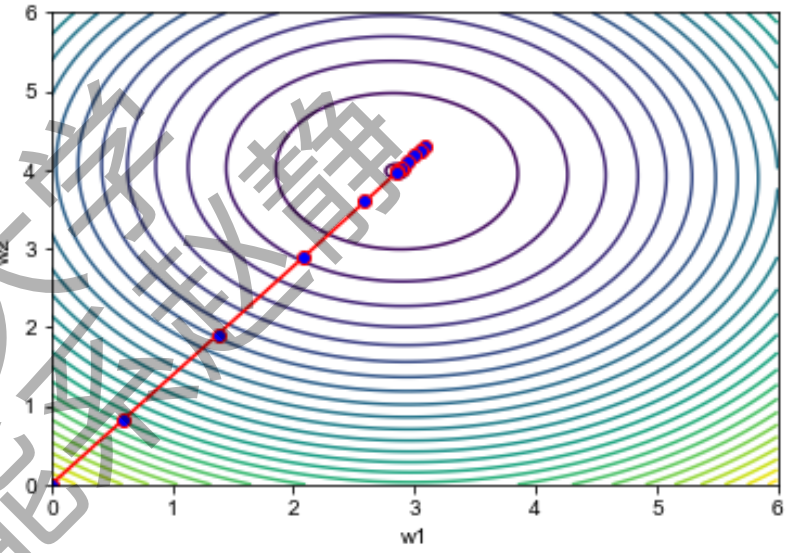
动量梯度下降



动量法比较



梯度下降

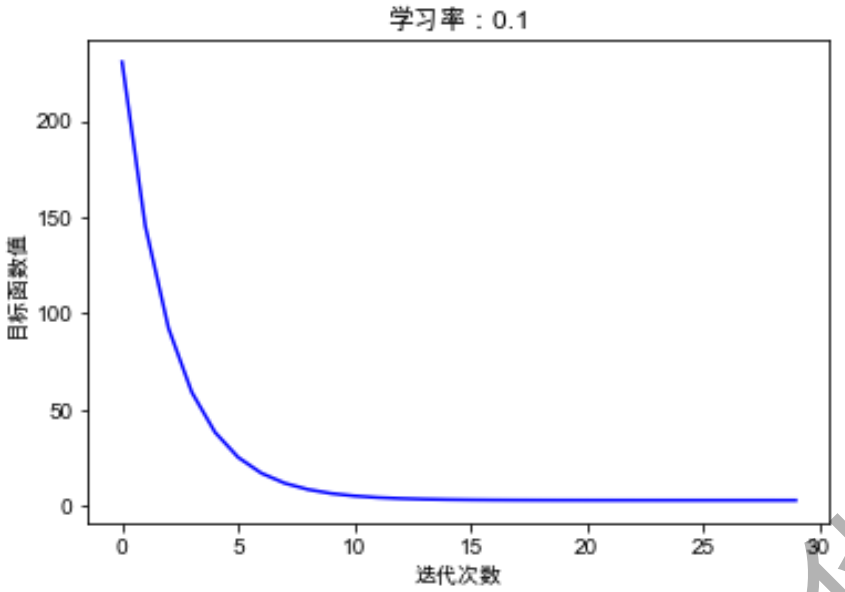


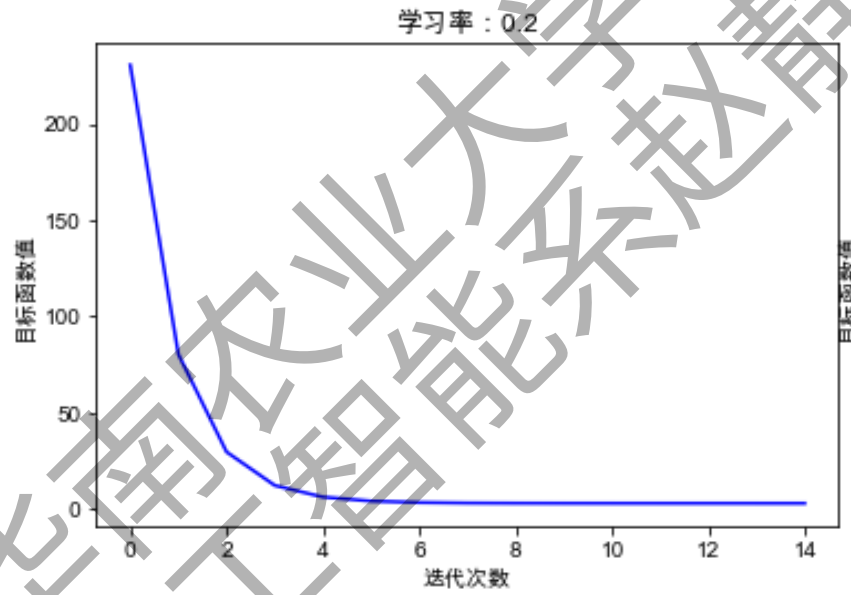
动量梯度下降

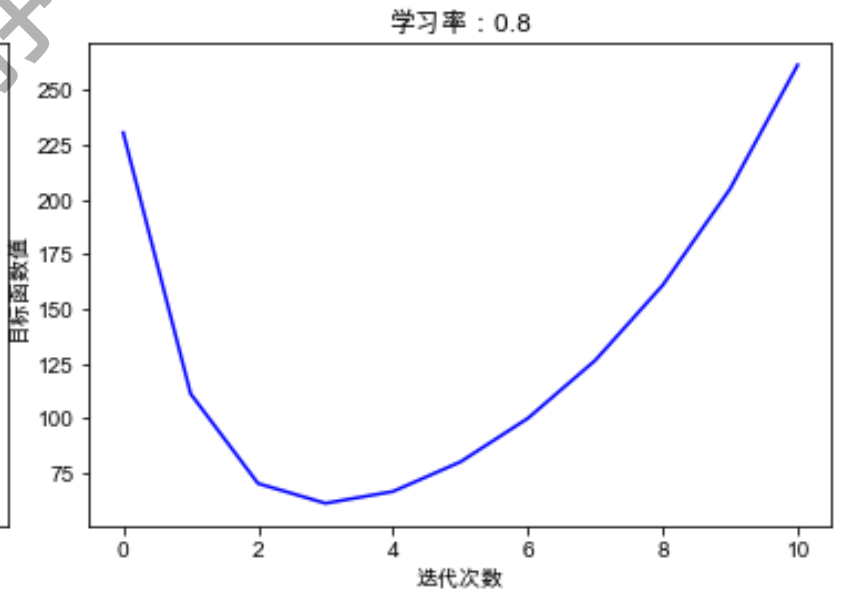
动量梯度下降法收敛快

动量梯度下降法越过了最佳位置时，会慢慢回到最佳位置

✓ 学习率



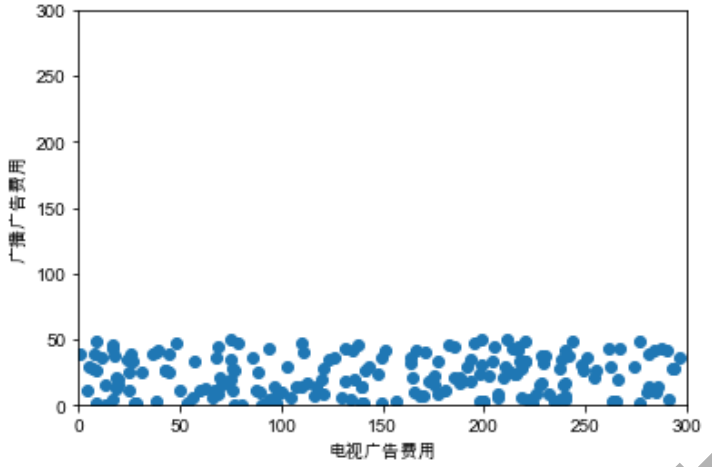




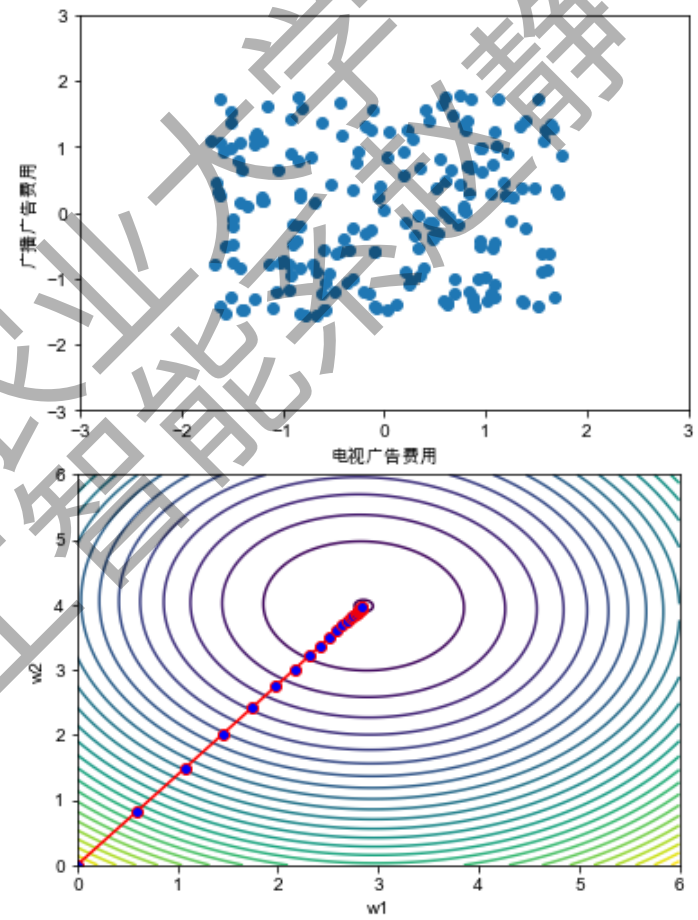
学习率对训练的影响

- 特征去量纲

OLS的梯度下降更新换代：$\boldsymbol{w}^{(t+1)} = \boldsymbol{w}^{(t)} - \eta g\left(\boldsymbol{w}^{(t)}\right) = \boldsymbol{w}^{(t)} + 2\eta \boldsymbol{X}^{\mathrm{T}}\left(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}^{(t)}\right)$



特征没有标准化



特征标准化

特征标准化：

$$x'_{i,j} = \frac{x_{i,j} - \mu_j}{\sigma_j}$$

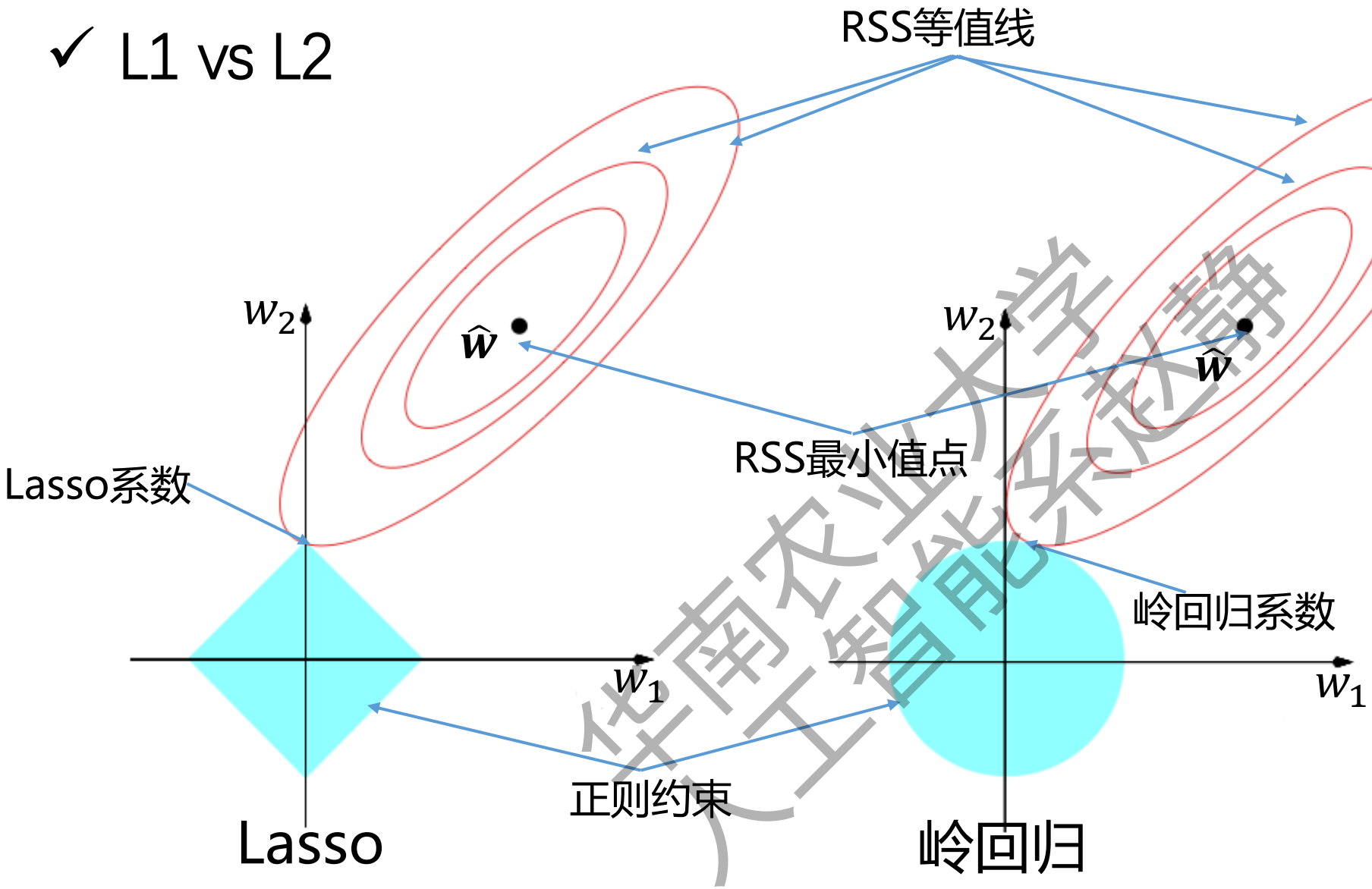
➢ 坐标轴下降法——Lasso求解

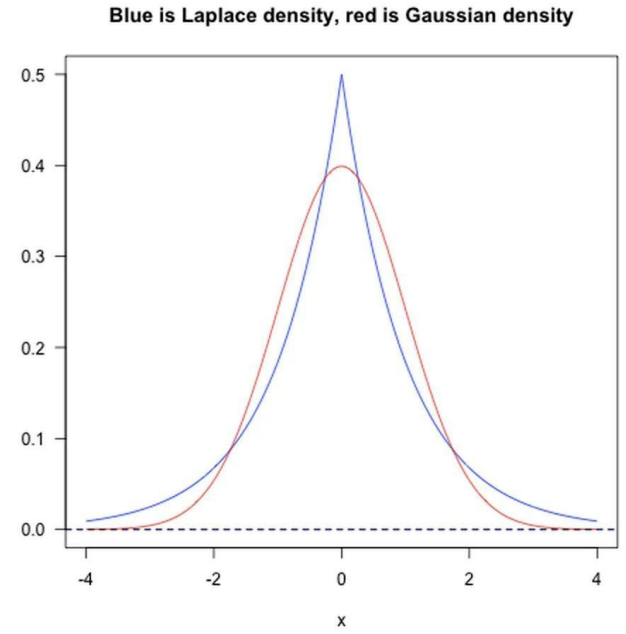
Lasso的目标函数为：$J(\boldsymbol{w}) = \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}\|_2^2 + \lambda\|\boldsymbol{w}\|_1$

坐标轴下降法：沿坐标轴方向搜索

- 引入次梯度，在不可微处直接赋值
- 在每次迭代中，在当前点处沿一个坐标轴方向进行一维搜索。
- 循环使用不同的坐标轴。一个周期的一维搜索迭代过程相当于一个梯度迭代。

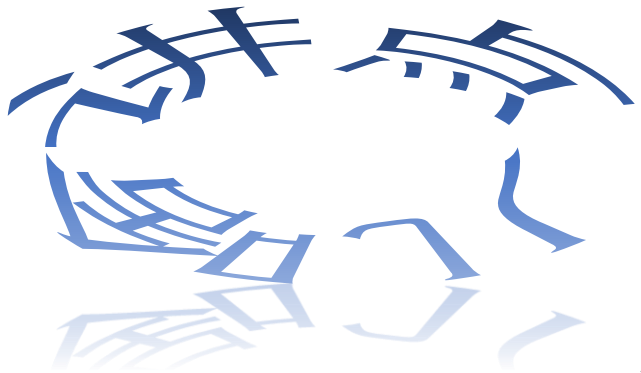
✓ L1 vs L2





L_1 正则化给出的最优解w*是使解更加靠近某些轴,而其它的轴则为0,所以 L_1 正则化能使得到的参数稀疏化。

L_2正则化给出的最优解w*是使解更加靠近原点,也就是说L_2正则化能降低参数范数的总和。



- 线性回归模型
- 回归任务的损失函数
- 线性回归模型的正则项
- 线性回归的的优化算法
- **回归任务的模型性能评价**

➢ 回归模型的评价指标

- 均方根误差（Rooted Mean Squared Error，RMSE）

$$\mathrm{RMSE}\,(\boldsymbol{y},\hat{\boldsymbol{y}}) = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i-\hat{y}_i)^2}$$

- 平均绝对误差（Mean Absolute Error，MAE）

$$\mathrm{MAE}(\boldsymbol{y},\hat{\boldsymbol{y}}) = \frac{1}{N}\sum_{i=1}^{N}|y_i-\hat{y}_i|$$

- 绝对误差中值（Median Absolute Error，MedAE）

$$\mathrm{MedAE}(\boldsymbol{y},\hat{\boldsymbol{y}}) = \mathrm{median}(|y_1-\hat{y}_1|,\ldots,|y_N-\hat{y}_N|)$$

- 均方对数误差（Mean Squared Logarithmic Error, MSLE）

$$\mathrm{MSLE}\,(\boldsymbol{y},\hat{\boldsymbol{y}}) = \frac{1}{N}\sum_{i=1}^{N}(\log(1+y_i)-\log(1+\hat{y}_i))^2$$

当y呈指数增长时可以使用（如计数、一年的平均销量...）

- R^2分数（R^2score）：既考虑了预测值与真值之间的差异，也考虑了问题本身真值之间的差异

$$SS_{res}(\boldsymbol{y},\hat{\boldsymbol{y}}) = \frac{1}{N}\sum_{i=1}^{N}(y_i-\hat{y}_i)^2,$$

$$SS_{tot}(\boldsymbol{y}) = \frac{1}{N}\sum_{i=1}^{N}(y_i-\bar{y})^2,$$

$$R^2(\boldsymbol{y},\hat{\boldsymbol{y}}) = 1-\frac{SS_{res}(\boldsymbol{y},\hat{\boldsymbol{y}})}{SS_{tot}(\boldsymbol{y})}$$

- 可解释方差分数（Explained variance score）：最佳分数为1，越小越不好

$$\text{explained\_variance}(\boldsymbol{y},\hat{\boldsymbol{y}}) = 1-\frac{\text{Var}(\boldsymbol{y}-\hat{\boldsymbol{y}})}{Var\{\boldsymbol{y}\}}$$

➢ 线性回归模型的超参数调优

- 信息准则： BIC/AIC
- 交叉验证: 广义交叉验证GCV

$$GCV(\lambda) = \frac{1}{N}\sum_{i=1}^{N}\left(\frac{y_i - f(\boldsymbol{x}_i)}{1-\frac{df(\lambda)}{N}}\right)^2, df(\lambda) = \sum_{j=1}^{D}\frac{\sigma_j^2}{\sigma_j^2+\lambda}$$

其中σ_j^2为矩阵$\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}$的特征值

- 函数集合：输入特征的线性组合
- 目标函数
  - 损失函数：L2损失、L1损失、Huber损失
  - 正则项：L2正则、L1正则
- 优化方法：解析法、梯度下降、坐标下降
- 模型评估：RMSE, MAE, R^2

Fun Time

以下哪个模型没有超参数?

1. 最小二乘线性回归
2. 岭回归
3. Lasso回归
4. 弹性网络回归