机器学习与深度学习

——线性回归

Personal Website: https://www.miaopeng.info/

Email: miaopeng@stu.scu.edu.cn

Github: https://github.com/MMeowhite

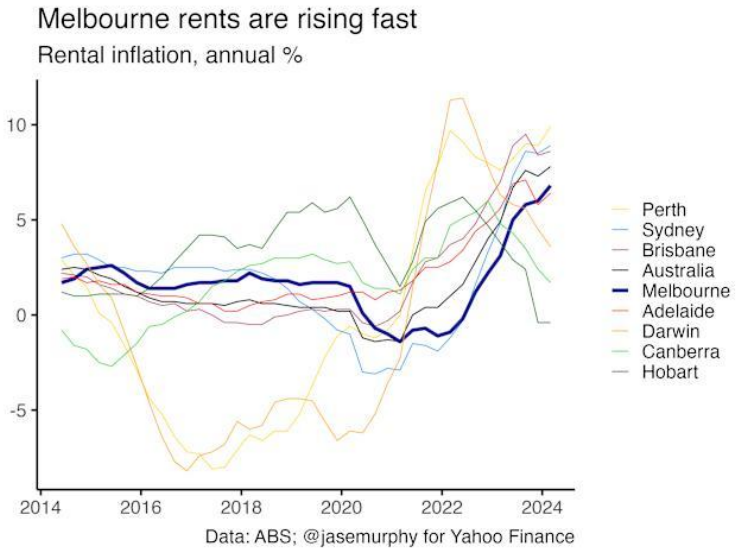
Youtube: https://www.youtube.com/@pengmiao-bmm

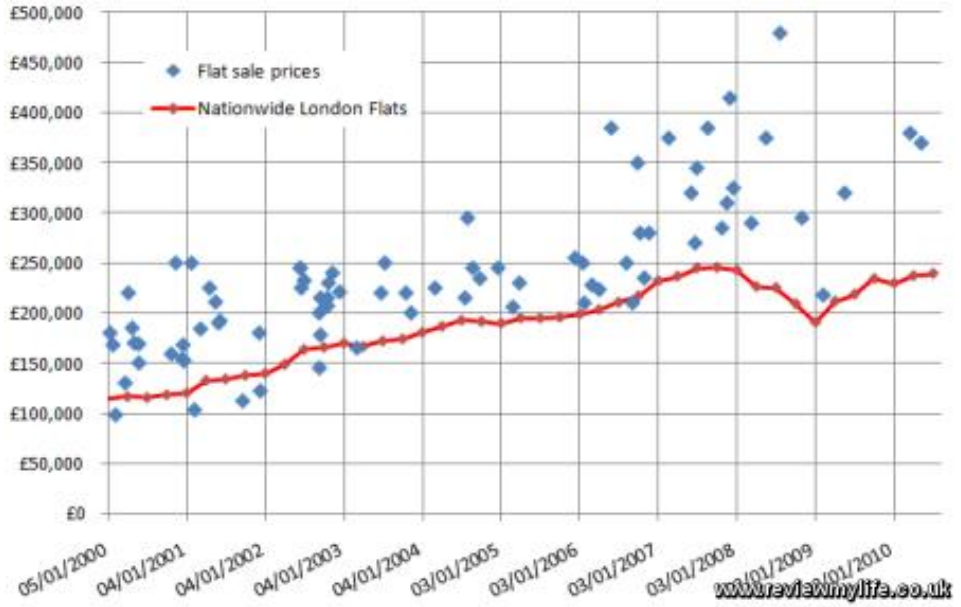
目录章节

CONTENTS

▶ 为什么学线性回归？

➢ 如何预测房价、销量、收入？线性关系是否真的存在？





- 所有复杂的模型，几乎都能追溯到一个简单的问题：**如何用一条直线解释数据**。它是统计学与机器学习的桥梁，帮助你理解建模的基本思想。
- 通过**回归系数可以解释每个特征对结果的影响大小和方向**，是后续深度模型不可替代的优势。
- 以线性回归入门打基础，为后续复杂模型做准备，逻辑回归、岭回归、Lasso、深度学习中的全连接层，都可以看作线性回归的延展或变形。

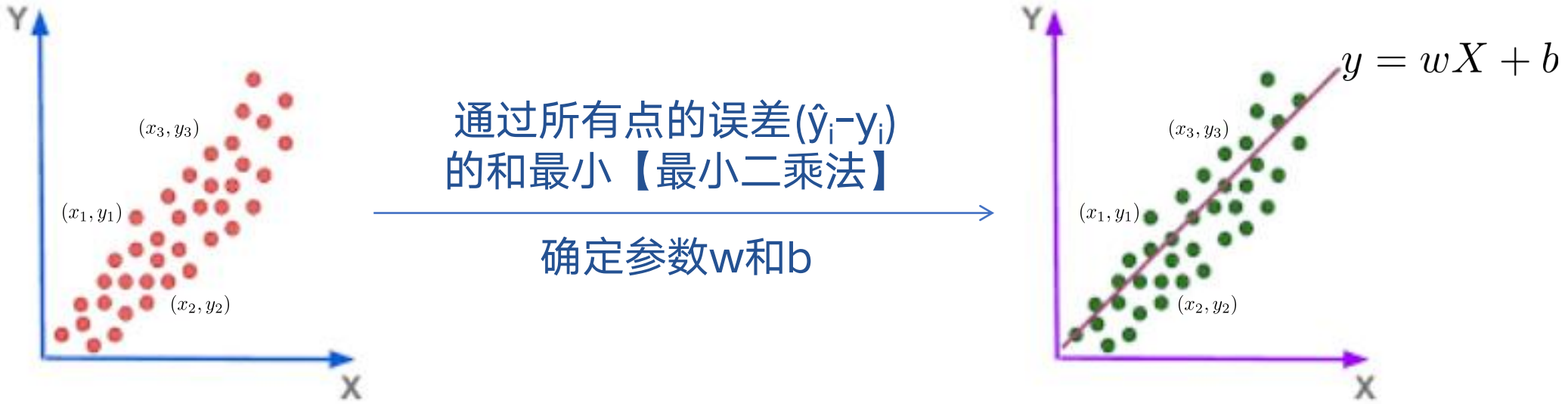
线性回归是机器学习的第一块基石。

什么是线性回归？

- 定义：通过一条**直线**拟合自变量 X 与因变量 y 之间的关系。
- 公式：假设我们有一系列的点（x_1, y_1），（x_2, y_2），（x_3, y_3），其中x_i表示第i个样本的特征（输入），y_i表示第i个样本的目标值。我们希望通过一个线性模型来拟合这些数据：

$$y = wX + b$$

- 核心目标：**让预测值$\hat{y}_i$与真实值y_i尽量接近**。
  - 直观理解：找到一条线，让他最贴近所有点（所有的点到这条直线的距离最短）。



▶ 线性回归的严谨完整形式（二维）

- 假设我们有一组观测数据点（**二维**）：

$$(x_1, y_1), (x_2, y_2), (x_3, y_3), ..., (x_n, y_n)$$

  - 其中x_i表示第i个样本的特征（输入），y_i表示第i个样本的目标值。
- 我们希望通过一个线性模型来拟合这些数据：

$$\hat{y}_i = \beta_0 + \beta_1 x_i$$

  - $\hat{y}_i$表示模型对第i个样本的预测值，β_1表示斜率，β_0表示截距。
- 建模的目标：找到最优的β_0和β_1，使得所有预测值$\hat{y}_i$和真实值y_i尽可能小，即最小化残差平方和（最小二乘法）：

$$\text{Loss} = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

线性回归的核心思想：就是用一条“最合适的直线”拟合点与点之间的关系，使预测值尽可能贴近真实值。

线性回归的严谨完整形式（n维）

- 假设我们有一组观测数据，共m个样本，每个样本有n个特征：
  - 第i个样本的输入特征为向量：$\mathbf{x}^{(i)} = [x_1^{(i)}, x_2^{(i)}, \ldots, x_n^{(i)}]^T$
  - 对应的目标值为：$y^{(i)}$
- 我们希望通过一个线性模型来拟合这些数据：

$$\hat{y}^{(i)} = \beta_0 + \beta_1 x_1^{(i)} + \beta_2 x_2^{(i)} + \cdots + \beta_n x_n^{(i)}$$

  - $\hat{y}^{(i)}$表示模型对第i个样本的预测值，β_i表示第i个特征的回归系数，β_0表示截距，$x^{(i)}$表示改样本的第i个特征输入
- 建模的目标：最小化残差平方和（最小二乘法），每个样本的残差定义为：

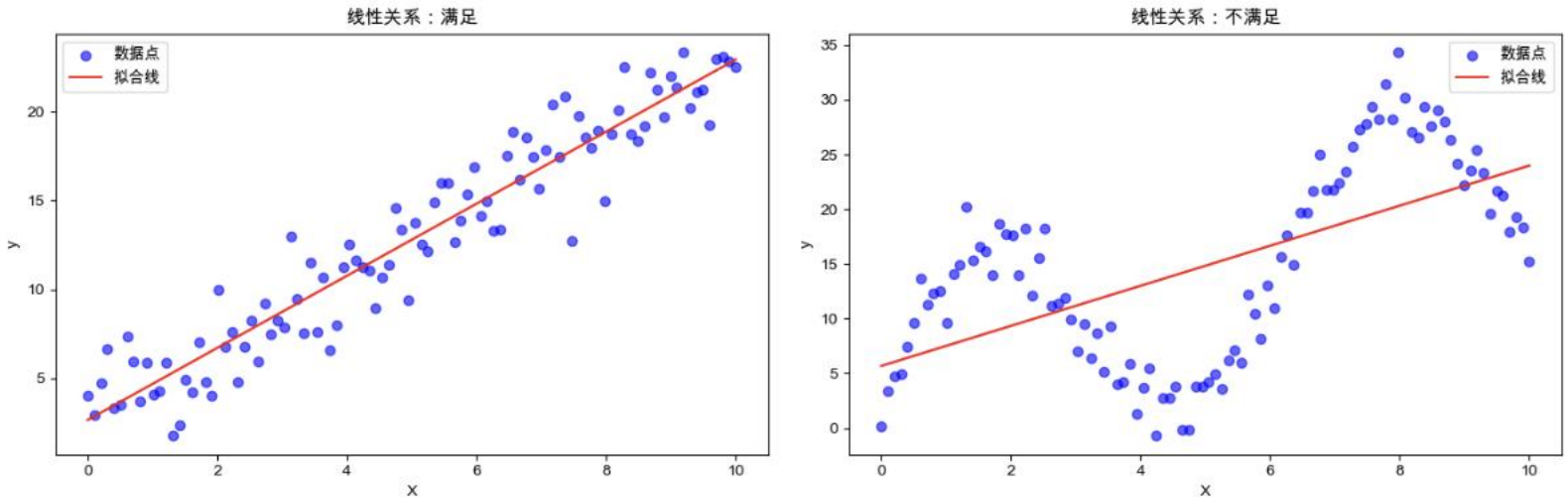
$$\epsilon^{(i)} = y^{(i)} - \hat{y}^{(i)} = y^{(i)} - \beta^T \mathbf{x}^{(i)}$$

  - 目标是**最小化所有样本的残差平方和**：

$$\min_{\beta} \sum_{i=1}^{m} (y^{(i)} - \beta^T \mathbf{x}^{(i)})^2$$

▶ 模型假设（非常重要！）

- 求解线性回归模型主要基于以下的假设，如果不满足以下的假设，那么模型拟合效果肯定不佳：
  - 1）**线性关系**：自变量和因变量之间存在线性关系。

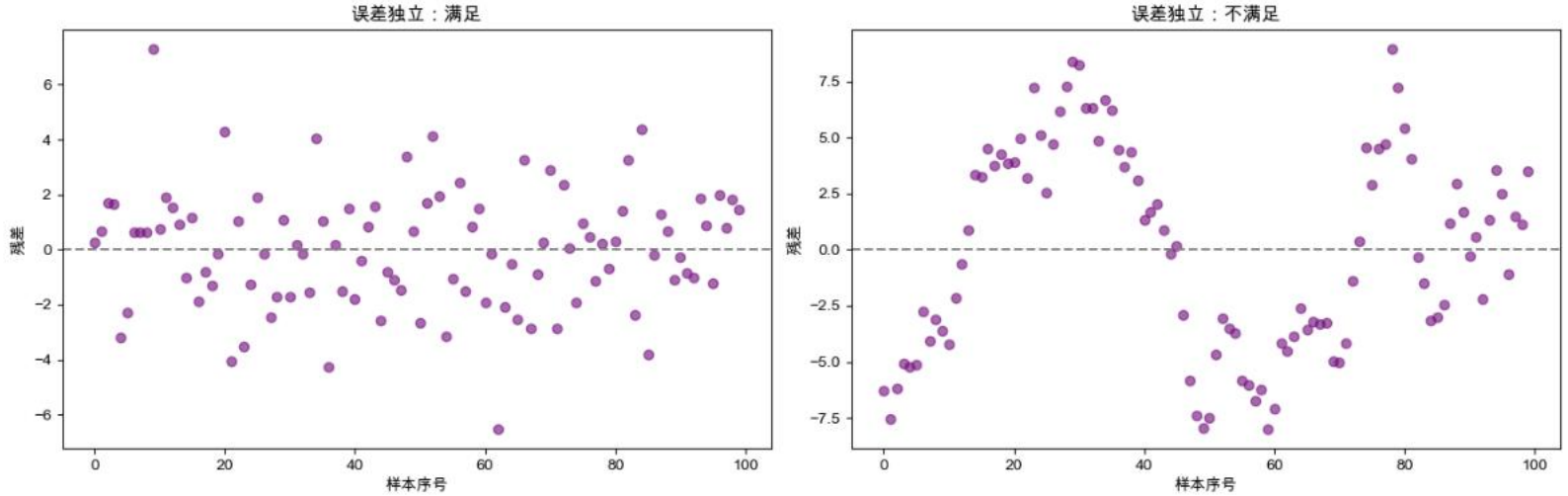


  - 如果你画出自变量和目标值的散点图，大致能画出一条“斜着的直线”，那就是线性关系。

线性回归建立在一组数学假设上，脱离这些假设，模型就失去了它应有的解释力。

▶ 模型假设（非常重要！）

- 求解线性回归模型主要基于以下的假设，如果不满足以下的假设，那么模型拟合效果肯定不佳：
  - 2）**误差独立**：残差之间不相关。各个样本的残差（预测误差）彼此之间不应存在系统性关联，也就是说，一个样本的预测误差不应该影响另一个样本的误差。

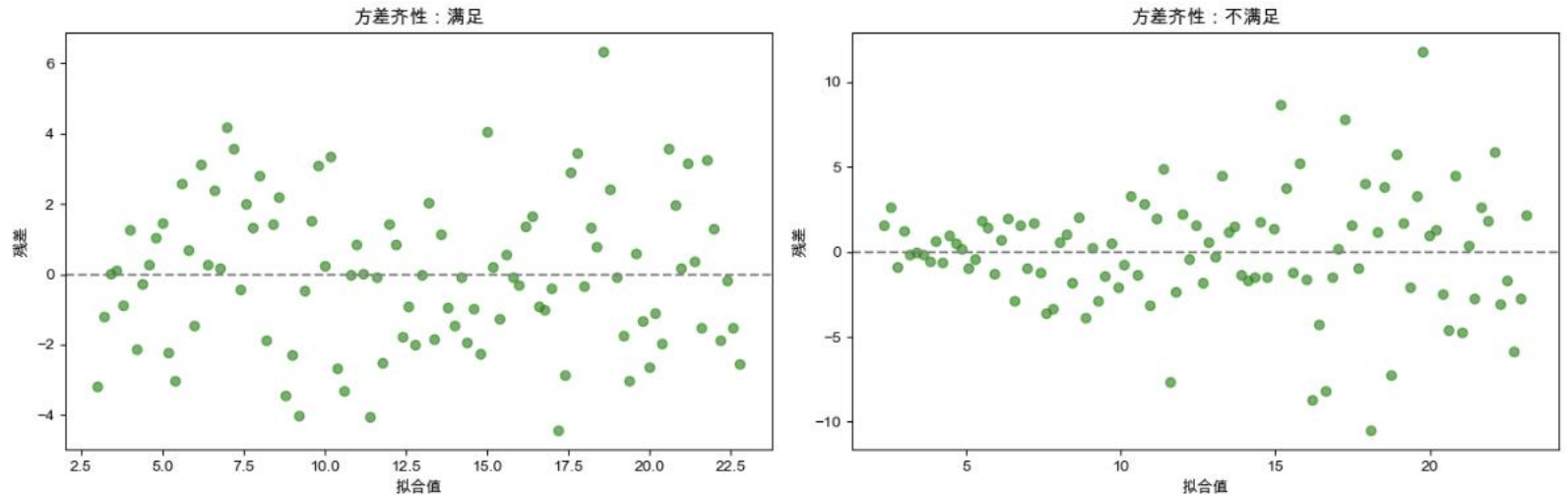


  - **时间序列数据**误差就会相互影响，例如：股票价格，气温预测，用户行为数据等。
    - 注：残差：实际值与模型预测值之间的差距 $e_i = y_i - \hat{y}_i$

线性回归建立在一组数学假设上，脱离这些假设，模型就失去了它应有的解释力。

▶ 模型假设（非常重要！）

- 求解线性回归模型主要基于以下的假设，如果不满足以下的假设，那么模型拟合效果肯定不佳：
  - 3）**方差齐性**：残差方差相等。无论预测值大还是小，模型的误差都应该“差不多大”。



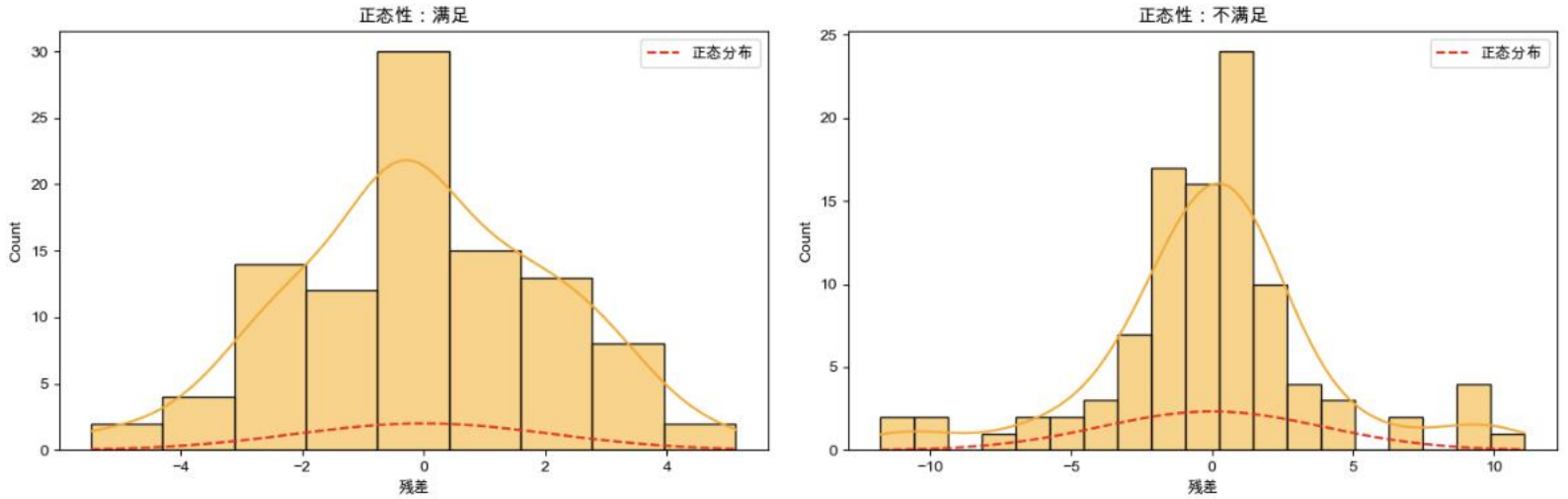
  - 模型在所有样本上的预测波动要保持一致，不能某些地方很准、某些地方很偏。
    - 注：残差：实际值与模型预测值之间的差距 $e_i = y_i - \hat{y}_i$

线性回归建立在一组数学假设上，脱离这些假设，模型就失去了它应有的解释力。

▶ 模型假设（非常重要！）

➢ 求解线性回归模型主要基于以下的假设，如果不满足以下的假设，那么模型拟合效果肯定不佳：

- 4）**正态性**：残差服从正态分布。线性回归模型中的残差（error term）应该服从均值为 0 的正态分布。



- 目的：为了推理和检验；提高模型置信度；小样本场景更敏感。
  - 注：残差：实际值与模型预测值之间的差距 $e_i = y_i - \hat{y}_i$

线性回归建立在一组数学假设上，脱离这些假设，模型就失去了它应有的解释力。

▶ 小结

- 为什么需要学习线性回归？
  - 机器学习与统计建模的入门基石；模型简单、可解释性强；广泛应用于预测、趋势分析、特征关系探索。
- 什么是线性回归？
  - 用线性方程描述自变量X与因变量Y的关系：

$$y = \beta_0 + \beta_1 X + \epsilon$$

$$y = \beta_0 + \beta_1 X_1 + \cdots \beta_n X_n + \epsilon$$

- 模型假设：
  - 线性可加性：X和Y之间是线性关系
  - 独立性：观测值之间相互独立。
  - 同方差性：误差方差相等。
  - 正态性：误差服从正态分布。

线性回归是一种在满足线性、独立、同方差、正态等假设下，用线性方程刻画自变量与因变量关系的基础预测与分析方法。

目录章节

CONTENTS

▶ 模型求解的目标：最小化损失

➢ 回顾核心思想：通过找到最优参数（如斜率和截距）让预测值最接近真实值。

- 第i个样本的输入特征为向量：$\mathbf{x}^{(i)} = [x_1^{(i)}, x_2^{(i)}, \ldots, x_n^{(i)}]^T$
- 对应的目标值为：$y^{(i)}$
- 线性回归的表达如下，任务就是求得损失的最小值【最小二乘损失函数（Least Squares Loss）】：

$$\hat{y}^{(i)} = \beta_0 + \beta_1 x_1^{(i)} + \beta_2 x_2^{(i)} + \cdots + \beta_n x_n^{(i)}$$

$$\epsilon^{(i)} = y^{(i)} - \hat{y}^{(i)} = y^{(i)} - \beta^T \mathbf{x}^{(i)}$$

$$\min_{\beta} \sum_{i=1}^{m} (y^{(i)} - \beta^T \mathbf{x}^{(i)})^2$$

- 使用梯度下降等数值优化方法时，可加上系数1/2m，取平均误差平方，即真正意义上的均方误差（MSE），这样损失函数的值不会随着样本数变化而剧烈改变：

$$\min_{\beta} \frac{1}{2m} \sum_{i=1}^{m} (y^{(i)} - \beta^T \mathbf{x}^{(i)})^2$$

- 意义：是为了在用梯度下降法推导时，求导后常数项2可以消掉，简化更新公式。**对于理论最优解（如正规方程），是否加系数不会影响结果**。

模型求解的方法：正规方程法

- 通过之前的推导，我们的目标是求：

$$\min_{\beta} \frac{1}{2m} \sum_{i=1}^{m} (y^{(i)} - \beta^T \mathbf{x}^{(i)})^2$$

  - 我们可以通过**向量化和求导法（正规方程推导）**来**精确**地求解最优参数β。这是线性回归最经典、最基本的数学推导之一。

- 首先，我们先进行矩阵化表示：$\hat{y}^{(i)} = \beta_0 + \beta_1 x_1^{(i)} + \beta_2 x_2^{(i)} + \cdots + \beta_n x_n^{(i)}$ 假设：

  - 输入矩阵：$\mathbf{X} \in \mathbb{R}^{m \times n}$ 每行为一个样本的特征向量（带常数项的偏置列）
  - 标签向量：$\mathbf{y} \in \mathbb{R}^{m \times 1}$
  - 参数向量：$\beta \in \mathbb{R}^{n \times 1}$

- 那么预测值写成矩阵的形式为：

$$\hat{\mathbf{y}} = \mathbf{X}\beta$$

- 损失函数可以写成向量形式：

$$J(\beta) = \frac{1}{2m} ||\mathbf{y} - \mathbf{X}\beta||^2$$

模型求解的方法：正规方程法

- 1）首先展开损失函数：

$$J(\beta) = \frac{1}{2m}(\mathbf{y} - \mathbf{X}\beta)^T(\mathbf{y} - \mathbf{X}\beta)$$

- 2）对β求导：

$$\begin{aligned} J(\beta) &= \frac{1}{2m}(\mathbf{y} - \mathbf{X}\beta)^T(\mathbf{y} - \mathbf{X}\beta) \\ \nabla_\beta J(\beta) &= \frac{1}{2m} \cdot \nabla_\beta \left[\mathbf{y}^T\mathbf{y} - 2\beta^T\mathbf{X}^T\mathbf{y} + \beta^T\mathbf{X}^T\mathbf{X}\beta\right] \\ &= \frac{1}{2m}\left[-2\mathbf{X}^T\mathbf{y} + 2\mathbf{X}^T\mathbf{X}\beta\right] \\ &= \frac{1}{m}\left(\mathbf{X}^T\mathbf{X}\beta - \mathbf{X}^T\mathbf{y}\right) \end{aligned}$$

- 3）令梯度为0，解得最优参数：

$$\nabla_\beta J(\beta) = 0 \Rightarrow \mathbf{X}^T\mathbf{X}\beta = \mathbf{X}^T\mathbf{y}$$

$$\boxed{\beta = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}}$$

- **注意事项：该解法要求X^TX可逆，若不可逆，可采用正则化方法，如岭回归（加入系数λI）**

模型求解的方法：梯度下降法

- 通过之前的推导，我们的目标是求：

$$\min_{\beta} \frac{1}{2m} \sum_{i=1}^{m} (y^{(i)} - \beta^T \mathbf{x}^{(i)})^2$$

  - 我们还可以通过**梯度下降法**来**近似**地求解最优参数β。这是整个深度学习最经典的算法之一。

- 首先，我们仍先进行矩阵化表示：$\hat{y}^{(i)} = \beta_0 + \beta_1 x_1^{(i)} + \beta_2 x_2^{(i)} + \cdots + \beta_n x_n^{(i)}$ 假设：

  - 输入矩阵：$\mathbf{X} \in \mathbb{R}^{m \times n}$ 每行为一个样本的特征向量（带常数项的偏置列）
  - 标签向量：$\mathbf{y} \in \mathbb{R}^{m \times 1}$
  - 参数向量：$\beta \in \mathbb{R}^{n \times 1}$

- 那么预测值写成矩阵的形式为：

$$\hat{\mathbf{y}} = \mathbf{X}\beta$$

- 损失函数可以写成向量形式：

$$J(\beta) = \frac{1}{2m} ||\mathbf{y} - \mathbf{X}\beta||^2$$

▶ 模型求解的方法：梯度下降法

➢ 1）首先展开损失函数：

$$J(\beta) = \frac{1}{2m}(\mathbf{y} - \mathbf{X}\beta)^T(\mathbf{y} - \mathbf{X}\beta)$$

➢ 2）对β求导：

$$\begin{aligned} J(\beta) &= \frac{1}{2m}(\mathbf{y} - \mathbf{X}\beta)^T(\mathbf{y} - \mathbf{X}\beta) \\ \nabla_\beta J(\beta) &= \frac{1}{2m} \cdot \nabla_\beta \left[\mathbf{y}^T\mathbf{y} - 2\beta^T\mathbf{X}^T\mathbf{y} + \beta^T\mathbf{X}^T\mathbf{X}\beta\right] \\ &= \frac{1}{2m}\left[-2\mathbf{X}^T\mathbf{y} + 2\mathbf{X}^T\mathbf{X}\beta\right] \\ &= \frac{1}{m}\left(\mathbf{X}^T\mathbf{X}\beta - \mathbf{X}^T\mathbf{y}\right) \end{aligned}$$

➢ 3）设学习率（步长）为λ>0，梯度下降的批量更新（Batch GD）为：

$$\beta \leftarrow \beta - \lambda\nabla_\beta J(\beta) = \beta - \lambda \cdot \frac{1}{m}X^T(X\beta - y)$$

- 按照坐标分量写，第j个分量梯度为：

$$\boxed{\beta_j \leftarrow \beta_j - \lambda \cdot \frac{1}{m}\sum_{i=1}^{m}(\beta^T x^{(i)} - y^{(i)})x_j^{(i)}}$$

- 当梯度为0时，梯度下降至最优点。

▶ 模型评估：目标与基本指标

- 为什么要评估线性回归模型？
  - 假设我们已经训练出了模型参数，那么问题来了：1）它预测得准吗？2）它对未来数据能泛化吗？**训练损失小 ≠ 模型好**。
- 评估的目的：
  - 衡量拟合效果：模型是否能准确地描述数据之间的关系。
  - 判断是否过拟合或欠拟合：判断模型是否仅仅记住了训练数据而不能推广。
  - 比较不同模型优劣：在不同模型之间选择最优。
- 评估的基本指标概述：
  - 均方误差（MSE）：衡量预测值与真实值差距的平方平均，大误差会被放大惩罚。
  - 均方根误差（RMSE）：MSE 开平方，表示平均每次预测大概错了多少，单位直观。
  - 决定系数R2：表示模型解释了多少原始数据的变化，越接近 1 说明越好。

评估不是为了看训练得有多“准”，而是看模型能不能“举一反三”。

▶ 模型评估：目标与基本指标

➢ 评估的基本指标概述：

指标名称	缩写	公式	含义说明
均方误差	MSE	$MSE = \frac{1}{m}\sum_{i=1}^{m}(\hat{y}_i - y_i)^2$	平均预测误差的平方，衡量整体误差
均方根误差	RMSE	$RMSE = \sqrt{MSE}$	MSE 的平方根，单位与原数据一致，更直观
决定系数	R^2	$R^2 = 1 - \frac{\sum(\hat{y}_i - y_i)^2}{\sum(\hat{y}_i - \bar{y})^2}$	表示模型对结果方差的解释能力，越接近1越好

➢ 特点比较：

- MSE所有误差都被平方导致大误差惩罚得更严重，单位是原始单位的平方，不太直观。
- RMSE单位和原始数据一致，更直观，但是对大误差敏感。
- R^2范围通常在 0 到 1：R^2=1表示完美预测，R^2=0跟平均值猜的一样差，R^2<0还不如瞎猜。

➢ 应用建议：

- MSE：常用作训练目标函数，对异常值比较敏感（因为平方会放大）。
- RMSE：用来报告结果时比 MSE 更容易理解，是Kaggle比赛时常见的评价指标。
- R^2：最常用的拟合优度指标（越接近1越好），可以快速判断模型解释了多少“变化”。

▶ 小结

➢ 线性模型的求解主要包含两种方法：梯度下降法、正规方程法。

- 正规方程法：

$$\beta = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$$

- 梯度下降法：

$$\beta_j \leftarrow \beta_j - \lambda \cdot \frac{1}{m}\sum_{i=1}^{m}(\beta^T x^{(i)} - y^{(i)})x_j^{(i)}$$

➢ 线性模型的评估主要包含三个指标：MSE，RMSE和R^2。

指标名称	缩写	公式	含义说明
均方误差	MSE	$MSE = \frac{1}{m}\sum_{i=1}^{m}(\hat{y}_i - y_i)^2$	平均预测误差的平方，衡量整体误差
均方根误差	RMSE	$RMSE = \sqrt{MSE}$	MSE 的平方根，单位与原数据一致，更直观
决定系数	R^2	$R^2 = 1 - \frac{\sum(\hat{y}_i - y_i)^2}{\sum(\hat{y}_i - \bar{y})^2}$	表示模型对结果方差的解释能力，越接近1越好

线性回归可用正规方程法或梯度下降法求解，模型好坏可用 MSE、RMSE 衡量误差大小，用R^2衡量解释能力。

目录章节

CONTENTS

模型扩展与改进——加入正则项：防止过拟合

- 什么是正则项：
  - 正则化的原始形式（惩罚形式）可以总结为如下：

$$\min_{\theta} \underbrace{J(\theta)}_{Loss} + \lambda \underbrace{R(\theta)}_{Reg}$$

  - J(θ)：为拟合误差，比如MSE，后续的逻辑回归误差也可以使用。
  - R(θ)：正则项，比如$||\theta||_1$或$||\theta||^2_2$。
  - λ：调节惩罚力度的超参数。
- 这时，如果想最小化一个损失函数J(θ)，但**希望参数不要太大**，于是加入一个约束：

$$\min_{\theta} J(\theta) \quad \text{subject to} \quad R(\theta) \leq t$$

  - t是允许的“模型复杂度”上限。
- 这就是带约束的优化问题，所以我们需要一种方法，把约束条件合并进目标函数里——这就是**拉格朗日函数**。

▶ 扩展：拉格朗日函数

➢ 目标：求解一个带约束的最优化问题，一般形式如下：

$$\min_{\theta} f(\theta) \quad \text{subject to} \quad g_i(\theta) = 0, h_j(\theta) \leq 0$$

➢ 1）构造广义拉格朗日函数（拉格朗日乘子法的扩展【带有不等式约束】）：

$$\mathcal{L}(\theta, \lambda, \mu) = f(\theta) + \sum_i \lambda_i g_i(\theta) + \sum_j \mu_j h_j(\theta)$$

- f(θ)：表示目标函数。
- $g_i(\theta)$：等式约束。
- $h_i(\theta)$：不等式约束。
- λ_i、μ_j：拉格朗日乘子，权衡目标和约束。

➢ 经数学家证明，**函数在满足正则性条件（如约束函数光滑、满足约束资格条件）时，原始问题的极值点，可以通过拉格朗日函数的驻点来求解**，前提：

- 可微性
- 可行解存在
- 约束函数满足资格条件（regularity/constraint qualification）

▶ 扩展：拉格朗日函数

➢ 2）求偏导数（梯度），即对所有变量求偏导，构造如下系统：

$$\nabla_\theta \mathcal{L} = 0$$
$$g_i(\theta) = 0$$
$$h_j(\theta) \le 0, \quad \mu_j \ge 0, \quad \mu_j h_j(\theta) = 0$$

➢ 3）解联立方程（最优解）：

- 这个系统通常是非线性联立方程组，可通过：1）手工代数解（简单情况下）；2）数值优化方法（复杂情况下）进行求解。

➢ 4）拉格朗日函数求解例子：

- 求此函数的局部最小值：

$$f(x,y) = x^2 y, \quad \text{restriction: } x^2 + y^2 = 1$$
$$\mathcal{L}(x,y,\lambda) = x^2 y + \lambda(x^2 + y^2 - 1)$$

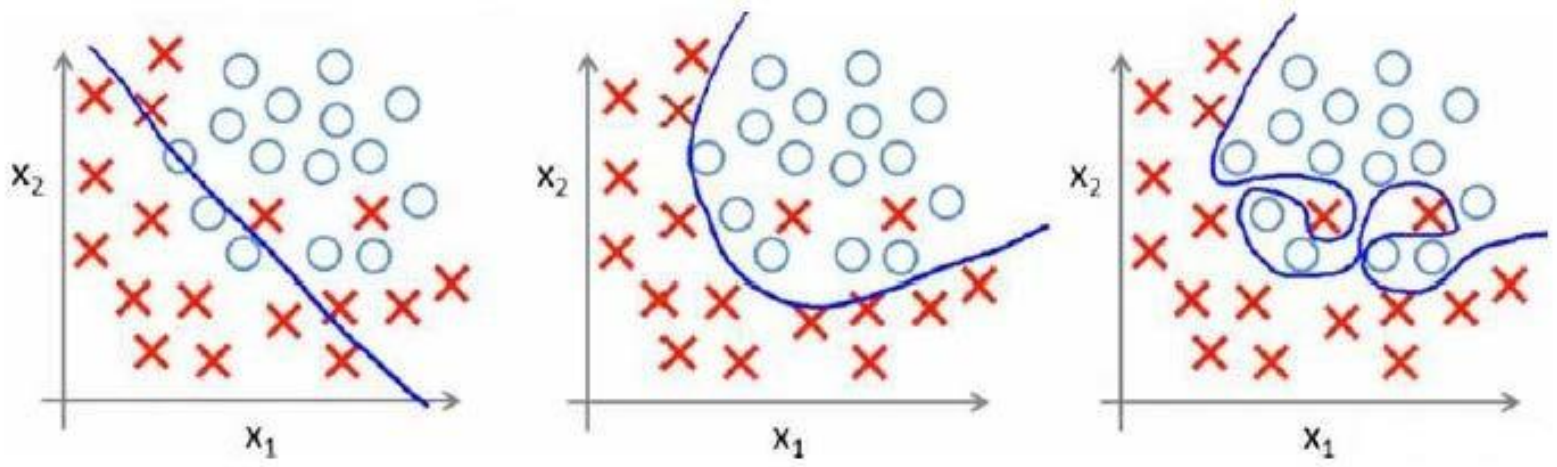
- 将该方程的偏微分设置为0，得到一个方程组，局部最小值可能就是以下方程组的解中的一个：

$$2xy + 2\lambda x = 0$$
$$x^2 + 2\lambda y = 0$$
$$x^2 + y^2 = 1$$

▶ 模型扩展与改进——加入正则项：防止过拟合

➢ 为什么正则化？

- **控制过拟合**：当模型参数过多时，很容易在训练集上拟合得很好，却在测试集上表现差。正则项通过限制参数的大小或复杂度，使模型更“平滑”，从而提高泛化能力。
- **控制模型复杂度**：模型复杂度越高，往往越容易出现噪声学习（尤其是小样本时）。正则化鼓励模型参数较小甚至稀疏（尤其是L1正则），从而使模型更简单、可解释。
- **提高数值稳定性**：特别是在线性回归中，如果特征高度相关（共线性问题），解可能会非常不稳定。加入 L2 正则（岭回归）可以起到抑制病态矩阵、改善可逆性的效果。
- **实现特征选择（L1正则的作用）**：L1 正则（Lasso）具有自动选择特征的能力，会使某些参数变为0。在高维数据中，这种稀疏性有助于提升模型解释力、减少噪声。

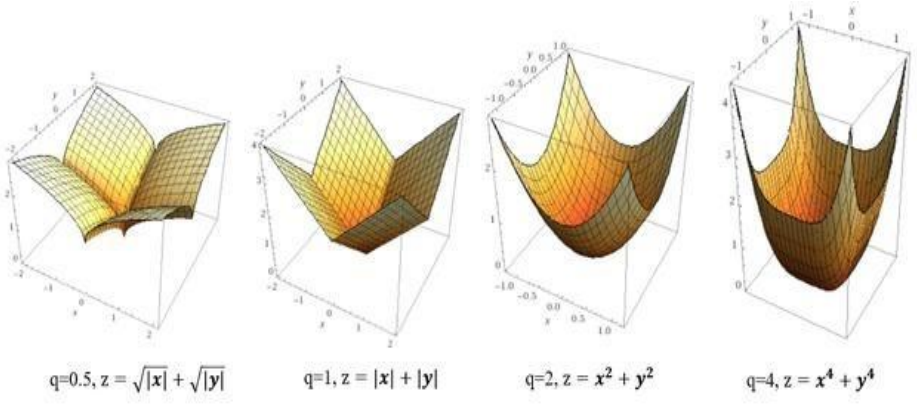


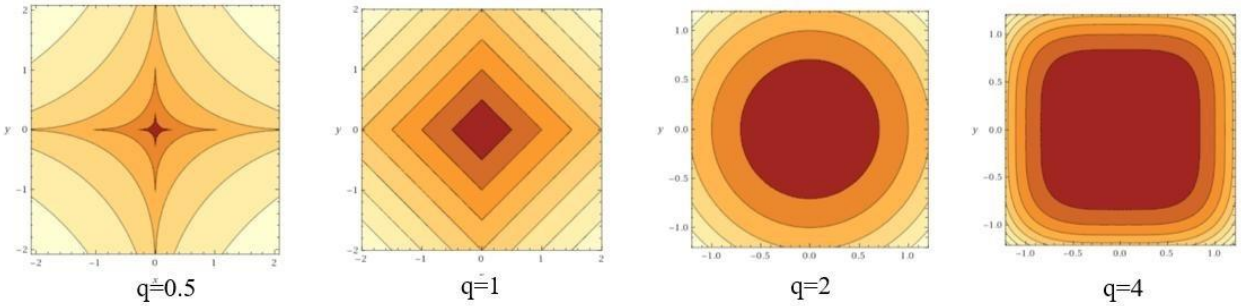
▶ 模型扩展与改进——加入正则项：防止过拟合

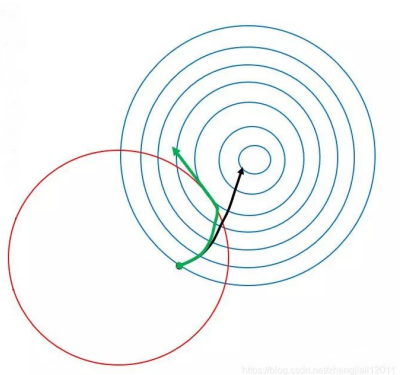
➢ 正则项的几何意义？

- 考虑到在高维数据下很难给出正则项的几何意义，我们假设数据源只有两个特征：

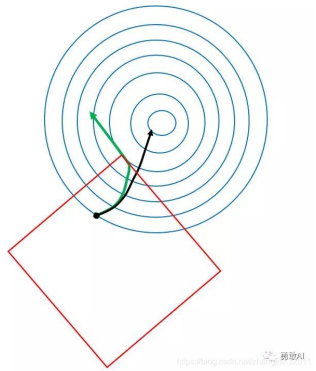
$$x = \{x_1, x_2\}, w = \{w_1, w_2\}$$







L2正则



L1正则

- 在参数空间中，原始损失函数的等值线可能是椭圆形（如果特征经过标准化，也可视为圆形）。无正则化时，优化过程会沿着梯度下降方向移动到椭圆中心。
- 加入正则化后，相当于在参数空间中增加了一个约束区域（L2 是圆，L1 是菱形），我们只能在该区域内寻找最优解。**当原损失的等值线与正则化区域的边界首次相切时，就得到了正则化下的最优解**。
- 这种“相切”点体现了原始损失最小化与模型复杂度限制之间的平衡。

▶ 模型扩展与改进——加入正则项：防止过拟合

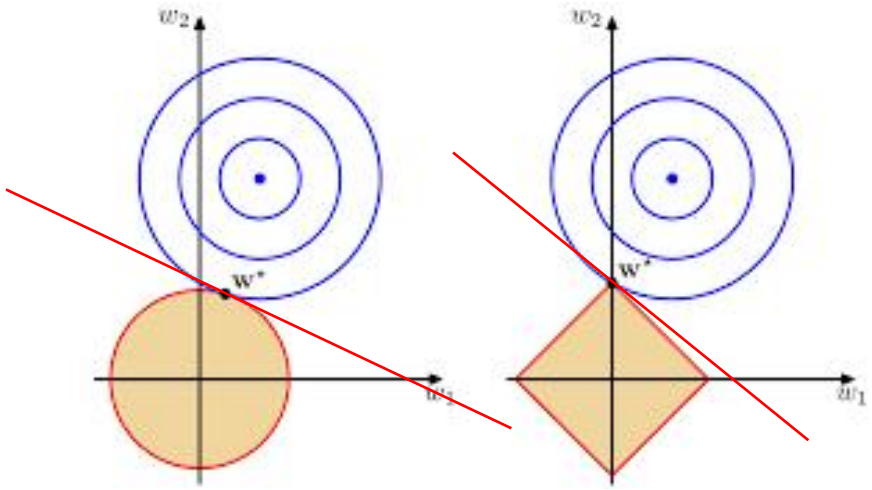
➢ 为什么原损失函数的等值线与约束条件的等值线相切之后就到达了最优点？

- 假设我们要最小化一个损失函数（目标函数f(w)，约束条件g(w)）：

$$\min f(\mathbf{w}), \quad g(\mathbf{w}) \leq c$$

  - f(w) → 原损失函数（等值线可以画成椭圆/圆）
  - g(w) → 正则化项目
  - c → 允许的模型复杂度（正则化“圈”的半径）
- 拉格朗日乘子法的核心条件：
  - 如果约束是等式约束g(w)=c，那么最优解需要满足：

$$\nabla \mathcal{L} = 0 \rightarrow \nabla(f(\mathbf{w}^*) - \lambda g(\mathbf{w}^*)) = 0 \rightarrow \nabla f(\mathbf{w}^*) = \lambda \nabla g(\mathbf{w}^*)$$



注：L1 正则化的可行域是尖角菱形，顶点处没有唯一切线，最优解是等值线撞到角而不是相切，通过子梯度条件而非切线条件确定。

  - 在最优点w，目标函数的梯度和约束函数的梯度方向相同（只差一个比例系数λ）。
  - 由于梯度方向是等值线的法线方向，所以说：**目标函数的等值线与约束边界在最优点处法线方向一致 → 等值线与边界相切。**

- 通俗理解：原损失函数等值线就像一圈一圈的山谷线，约束边界就是约定我们往下走的范围，当没有约束的时候，我们可以直接走到山谷最低点，但是有约束的时候，最优的位置就是边界和等值线刚好接触的地方，要么：1）不相切：说明还能继续在边界内走到更低的等值线；2）交叉相交：说明交点不是最优，能找到更低的线在边界上。相切的本质：梯度平衡了“让损失变小”和“保持模型简单”两个力。

▶ 扩展：正则化的特点

➢ 特点：1）L1正则化可以获得稀疏特征：

- 假设设计矩阵X的各列是标准正交（$X^TX=I$），考虑Lasso的目标：

$$\min_{\beta} \frac{1}{2}||y - X\beta||_2^2 + \lambda||\beta||_1$$

- 写成残差平方和项：

$$\frac{1}{2}||y - X\beta||_2^2 = \frac{1}{2}(y^Ty - 2\beta^TX^Ty + \beta^TX^TX\beta) = \frac{1}{2}(y^Ty - 2\beta^TX^Ty + \beta^T\beta)$$

- 令向量$z=X^Ty$（分量$z_j = X_jy$，可以理解为第j个特征与响应的相关量或投影系数）。于是目标函数变为：

$$\text{obj}(\beta) = \frac{1}{2}y^Ty + \sum_{j=1}^{p}[\frac{1}{2}\beta_j^2 - z_j\beta_j + \lambda|\beta_j|]$$

- 注意第一项为常数项与β无关。重要的是**目标函数变成了各坐标之和**：

$$\min_{\beta} \sum_{j=1}^{p}[\frac{1}{2}\beta_j^2 - z_j\beta_j + \lambda|\beta_j|] = \min_{\beta} \sum_{j=1}^{p}[\frac{1}{2}(\beta_j - z_j)^2 + \lambda|\beta_j|] + \text{const}$$

- 即每个j的子问题是独立的：

$$\min_{\beta_j} \frac{1}{2}(\beta_j - z_j)^2 + \lambda|\beta_j|$$

▶ 扩展：正则化的特点

➢ 特点：1）L1正则化可以获得稀疏特征：

- 固定某个j，令目标：

$$\phi(\beta_j) = \min_{\beta_j} \frac{1}{2}(\beta_j - z_j)^2 + \lambda|\beta_j|$$

- 我们分三种情形处理（注意 L1 在 0 处不可导，所以要分情况）：
- a.假设$\beta_j>0$，此时$|\beta_j| = \beta_j$，对β_j求导并令导数为0：

$$\phi'(\beta_j) = \beta_j - z_j + \lambda = 0 \rightarrow \beta_j = z_j - \lambda$$

- 这个解自洽的条件是 RHS 要严格大于 0（因为我们在$\beta_j>0$情形），即：

$$z_j - \lambda = 0 \rightarrow z_j > \lambda$$

- b.假设$\beta_j<0$，此时$|\beta_j| = -\beta_j$，对β_j求导并令导数为0：

$$\phi'(\beta_j) = \beta_j - z_j - \lambda = 0 \rightarrow \beta_j = z_j + \lambda$$

- 这个解自洽的条件是 RHS 要严格大于 0（因为我们在$\beta_j>0$情形），即：

$$z_j + \lambda = 0 \rightarrow z_j < -\lambda$$

- c.假设$\beta_j=0$，因为在0点不可导，用子梯度条件判断最优性：0是最优解的充要条件为0属于$\partial\phi(0)$。计算子梯度：

$$\partial\phi(0) = (0 - z_j) + \lambda \cdot \partial|0| = -z_j + \lambda[-1, 1]$$

- 0 属于这个集合等价于存在$s\in[-1,1]$使得$-z_j + \lambda s = 0$，即$|z_j|\leqslant\lambda$。所以若$|z_j|\leqslant\lambda$，则$\beta_j=0$。

▶ 扩展：正则化的特点

➢ **特点：1）L1正则化可以获得稀疏特征：**

- 汇总结果：

$$\hat{\beta}_j = \begin{cases} z_j - \lambda, & z_j > \lambda, \\ 0, & |z_j| \le \lambda, \\ z_j + \lambda, & z_j < -\lambda. \end{cases}$$

- 写成软阈值（soft-thresholding）算子的形式：

$$\hat{\beta}_j = S(z_j, \lambda) = \text{sign}(z_j)\ \max\{\, |z_j| - \lambda,\, 0 \,\}$$

➢ 直观理解：

- $z_j = X_j y$表示特征j与响应的“相关强度”（在标准化、正交条件下）。
- 如果$|z_j|$小于阈值λ，说明该特征与响应的相关性不足以抵消正则化乘法，系数直接变为0（产生系数）。
- 如果$|z_j|$大于阈值λ，系数被缩小了λ（偏倚），但保留非零符号。

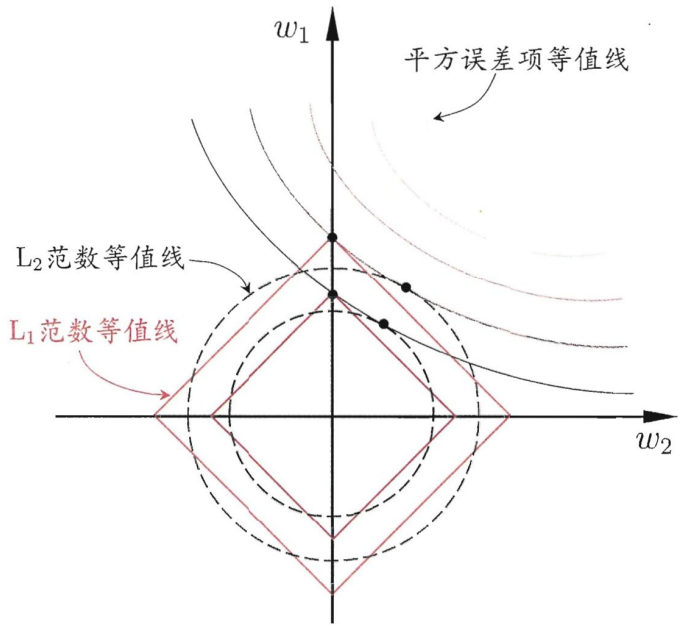
▶ 扩展：正则化的特点

➢ **特点：2）λ变化，菱形和圆形怎么变化？**

- 根据公式以及几何图像：

$$\min_{\theta} J(\theta) \quad \text{subject to} \quad R(\theta) \le t$$

$$R(\theta) = \lambda \sum_{j=1}^{d} |\theta_j|^{\{1,2\}}$$



- λ越大，菱形和圆形越小，求得的模型参数越小。

➢ **特点：3）为什么 L2 正则化比 L1 正则化应用更加广泛？**

- 1）**数学性质更“平滑”，优化容易**：L2 正则化的目标函数是可导的（甚至二次可导），梯度下降等方法可以直接用，而且收敛更稳定；L1 正则化在零点不可导（因为绝对值的尖角），优化时需要特殊处理（子梯度、坐标下降法等），实现起来更复杂。
- 2）**参数收缩的方式不同**：L2 正则化是 “均匀收缩”：所有参数都被拉向零，但一般不会真的变成零；L1 正则化是 “稀疏化”：部分参数直接被压成零（特征选择效果），但其余的参数收缩幅度可能更小。

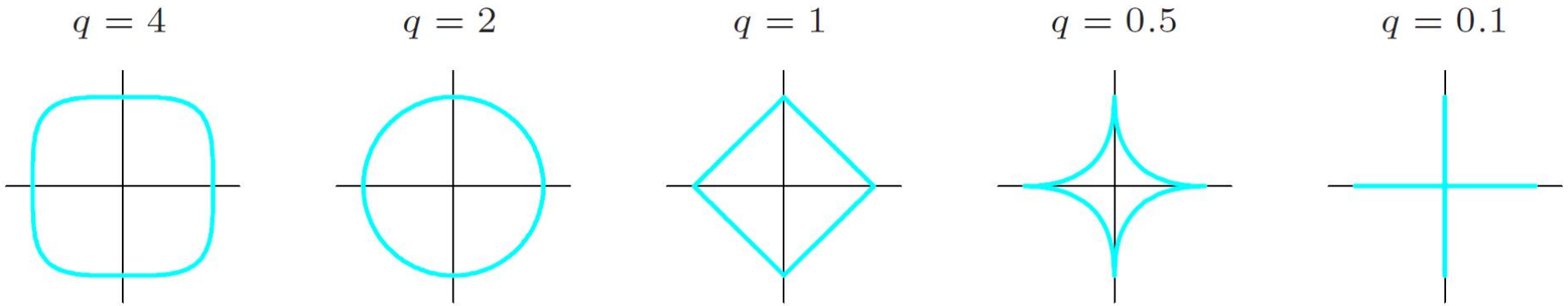
▶ 扩展：正则化的特点

➢ **特点：4）正则化可以推广至任意指数：**

- 不同的指数呈现不同的效果：

$$R(\theta) = \lambda \sum_{j=1}^{d} |\theta_j|^q, \quad q \geq 0$$

- 不同的 q 值对应着不同的约束边界，几何图像如下：



➢ **特点：5）L1正则化可以结合L2正则化使用：Elastic Net正则化【Zou & Hastie (2005)】：**

$$R(\theta) = \lambda \sum_{j=1}^{d} (\alpha\theta^2 + (1-\alpha)|\theta_j|)$$

- 可以通过参数α调节L1正则化和L2正则化的权重，兼备**既能筛特征又能提升模型泛化性**，适合**需要既做特征选择又防止过拟合的场景**。

▶ 模型扩展与改进——加入正则项：防止过拟合

- 1）岭回归（Ridge Regression）：
  - 在损失函数中加入**L2范数**：

$$J(\theta) = \frac{1}{2m}\sum(\hat{y}_i - y_i)^2 + \lambda||\theta||_2^2$$

  - 意义：控制参数大小，使模型更加平滑。
- 2）Lasso 回归（L1正则）：
  - 在损失函数中加入**L1范数**：

$$J(\theta) = \frac{1}{2m}\sum(\hat{y}_i - y_i)^2 + \lambda||\theta||_1$$

  - 意义：具有特征选择能力（可使某些参数变为0）。
- 3）弹性网回归（Elastic Net）
  - **综合 L1 + L2**：

$$J(\theta) = \frac{1}{2m}\sum(\hat{y}_i - y_i)^2 + \lambda_1||\theta||_1 + \lambda_2||\theta||_2^2$$

  - 意义：在稀疏性和稳定性之间找到平衡。

▶ 模型扩展与改进——扩展特征表达能力

- 1）多项式回归（Polynomial Regression）：
  - 对输入变量做高次扩展，例如x^2，x^3等：

$$y = \beta_0 + \beta_1 x + \epsilon \rightarrow y = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \epsilon$$

  - 本质：仍然是线性模型，但是在更高维空间拟合非线性关系。
  - 注意：高次项会导致多重共线性，可配合正则化（Ridge/Lasso）；同时，次数太高也容易过拟合。
  - 应用场景：一维非线性拟合；小规模特征集上的曲线建模。
- 2）基函数回归（Basis Expansion）：
  - 使用任意函数（如高斯核、正弦函数）扩展输入特征：

$$y = \beta_0 + \sum_{j=1}^{M} \beta_j \phi_j(x) + \epsilon, \quad \phi_j(x) = \exp(-\frac{(x-\mu_j)^2}{2\sigma^2}), \sin(kx), \cos(kx), \cdots$$

  - 特点：灵活，可根据先验知识选择基函数，但若基函数数量过多，也会面临高维问题。
  - 应用场景：非线性回归；信号处理（傅立叶基）；局部加权回归。

▶ 模型扩展与改进——解决特殊问题的变种

- 1）加权线性回归（Weighted Linear Regression, WLS）：
  - 给每个样本一个权重w_i，使重要样本影响更大：

$$\min_{\beta} \sum_{i=1}^{n} w_i (y_i - \mathbf{x}_i^T \beta)^2$$

  - 特点：本质是最小二乘法的加权版本，权重可事先给定或通过模型估计。
  - 应用场景：异方差性，即不同样本的噪声方差不同，取$w = 1/\sigma^2$；置信度不同的数据源融合。
- 2）稳健回归（Robust Regression）：
  - 用鲁棒损失代替平方损失，减少异常值的影响：

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

$$r = y - \mathbf{x}^T \beta$$

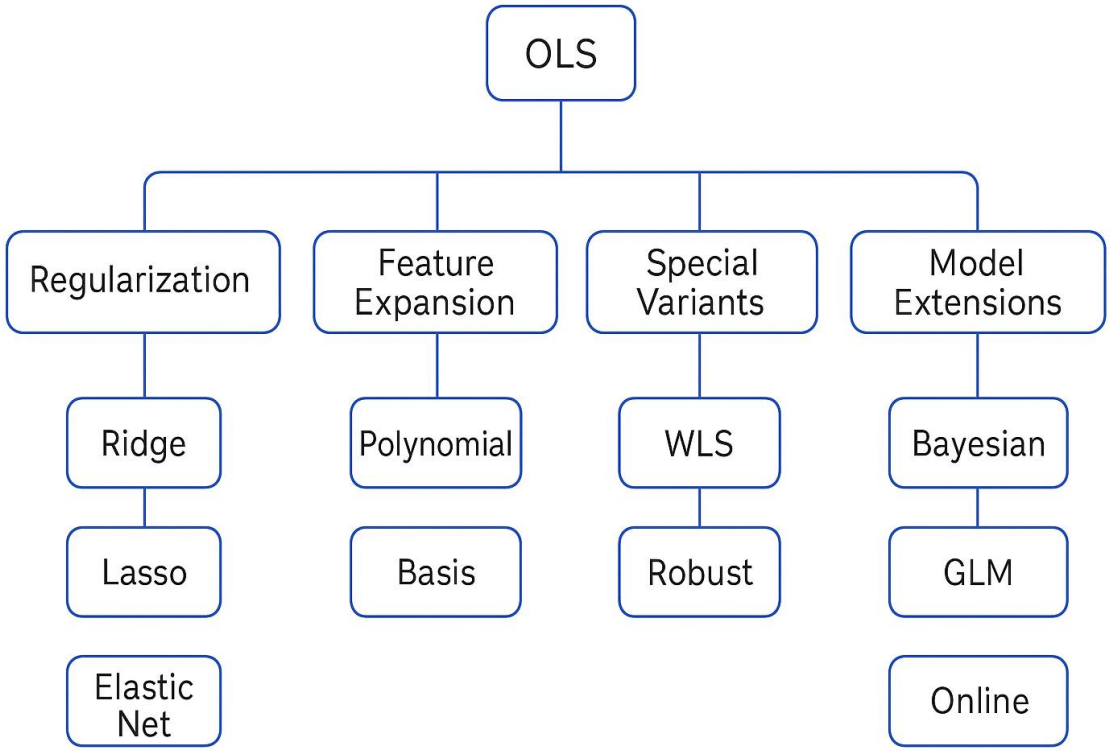
  - 特点：小误差时与普通回归相同，大误差切换为线性增长，降低异常点的影响。
  - 应用场景：数据中存在离群值，需要模型稳定性。

▶ 模型扩展与改进——其他

- 1）贝叶斯线性回归（Bayesian Linear Regression）
  - 把参数β当作随机变量，引入先验p(β)，根据数据得到后验分布p(β|X, y)。
  - 优点：可以给出预测的不确定性（置信曲线），自然引入正则化（高斯先验->L2正则化，Laplace先验->L1正则化）。
  - 应用场景：小样本、高不确定性任务。
- 2）在线/增量线性回归（Online / Incremental Regression）
  - 不一次性读取所有数据，而是随着数据流到达，持续更新模型参数。
  - 特点：适合大规模或流式数据，内存占用低。
  - 典型算法：随机梯度下降（SGD）。
- 3）广义线性模型（GLM）
  - 在输入端仍是线性部分η=Xβ，在输出端用链接函数g(μ)=η连接不同分布。
  - 常用实例：逻辑回归（Logistic Regression）：二分类；Poisson 回归：计数数据；Gamma 回归：正值连续变量。
  - 特点：统一了不同类型响应变量的回归框架。

小结

- 线性回归的核心是 OLS，它通过最小化残差平方和来拟合数据。
- 在此基础上可以引入正则化（Ridge、Lasso、Elastic Net）防止过拟合、扩展特征空间（多项式、基函数）提升表达能力。



- 针对特殊需求发展变种（加权、鲁棒、贝叶斯、GLM、在线）以适应不同数据特性与应用场景。

线性回归以 OLS 为核心，通过正则化提升泛化能力、特征扩展增强表达能力，并结合多种变种以适应不同数据特性与应用需求。

目录章节

CONTENTS

算法实现：numpy实现

- 实现：MyLinearRegression类，第一步：先定义类及初始化方法及内部（私有_）方法。

```python
class MyLinearRegression:
    def __init__(self, learning_rate=0.01, n_iters=1000, fit_intercept=True, method="gd", alpha=0.0, l1_ration=0.0):
        # 学习率
        self.learning_rate = learning_rate
        # 迭代次数
        self.n_iters = n_iters
        # 是否拟合截距
        self.fit_intercept = fit_intercept
        # 选择方法（gd为梯度下降法，normal为正规方程）
        self.method = method  # "gd" for gradient descent, "normal" for normal equation
        # 正则化强度【如果为0】
        self.alpha = alpha
        # L1正则化比例【0为纯L2，1为纯L1】
        self.l1_ration = l1_ration
        # 参数theta初始化
        self.theta = None

    # 添加截距项
    def _add_intercept(self, X):
        # X: shape (n_samples, n_features)
        # 如果存在截距项，则在X前添加一列1
        if self.fit_intercept:
            intercept = np.ones((X.shape[0], 1))
            return np.hstack((intercept, X)) # shape (n_samples, n_features + 1)
        return X
```

- 类定义及初始化：可选参数包括学习率、迭代次数、是否加截距、正则化强度alpha和L1比例l1_ration（Elastic Net）。

- _add_intercept方法（_表示私有）：用于在特征矩阵前加一列1，实现截距。

算法实现：numpy实现

➢ 实现：MyLinearRegression类，第二步：实现核心的模型训练方法：fit(X, y)。

```python
# 拟合模型
def fit(self, X, y):
    X = np.array(X) # 转换为numpy数组
    y = np.array(y) # 转换为numpy数组

    # 添加截距项
    X = self._add_intercept(X)

    # 梯度下降法
    if self.method == "gd":
        # 初始化参数
        self.theta = np.zeros(X.shape[1])
        for _ in range(self.n_iters):
            y_pred = X @ self.theta
            error = y_pred - y
            grad = (1 / len(y)) * (X.T @ error)
            # L2正则项 + L1正则项
            grad += self.alpha * ((1 - self.l1_ratio) * self.theta + self.l1_ratio * np.sign(self.theta))
            self.theta -= self.learning_rate * grad

    # 正规方程法
    elif self.method == "normal":
        self.theta = np.linalg.pinv(X.T @ X) @ (X.T @ y) # @表示矩阵乘法
        I = np.eye(X.shape[1])
        I[0, 0] = 0  # 不对截距项进行正则化
        l2 = self.alpha * (1 - self.l1_ration)
        # 普通线性回归 (alpha=0), Ridge (l1_ratio=0), Elastic Net (l1_ratio>0, 只近似L2部分)
        self.theta = np.linalg.pinv(X.T @ X + l2 * I) @ (X.T @ y)
    else:
        raise ValueError("method must be 'gd' or 'normal'")
```

- 数据转换为numpy数组，并加截距项。
- 根据method参数，选择实现梯度下降法还是正规方程法进行实现。
- 梯度下降法：首先初始化参数theta为零，每次根据反向传播迭代计算预测值、误差和梯度。梯度中同时加入L2（Ridge）和L1（Lasso近似）正则项，支持Elastic Net。
- 正规方程法：直接解析计算参数，支持L2正则（Ridge）和Elastic Net的L2部分。注：L1正则正规方程没有解析解，这里只近似支持L2。

算法实现：numpy实现

- 实现：MyLinearRegression类，第三步：实现核心的模型预测、评估方法。

```python
# 预测
def predict(self, X):
    X = self._add_intercept(np.array(X))
    return X @ self.theta

# 评估: R^2 score
def score(self, X, y):
    """R^2 score"""
    y_pred = self.predict(X)
    ss_res = np.sum((y - y_pred) ** 2)
    ss_tot = np.sum((y - np.mean(y)) ** 2)
    return 1 - ss_res / ss_tot
```

- predict方法用于输出预测结果。
- score方法计算R^2分数，衡量模型拟合优度。

▶ 算法实现：pytorch实现（练习）

➢ 利用Pytorch实现关键步骤：

```python
def fit(self, X, y):
    X = torch.tensor(X, dtype=torch.float32, device=self.device)
    y = torch.tensor(y, dtype=torch.float32, device=self.device).view(-1, 1)
    X = self._add_intercept(X)

    if self.method == "gd":
        # 初始化参数
        self.theta = torch.zeros((X.shape[1], 1), dtype=torch.float32, device=self.device, requires_grad=True)
        optimizer = torch.optim.SGD([self.theta], lr=self.learning_rate)

        for _ in range(self.n_iters):
            optimizer.zero_grad()
            # 预测
            y_pred = X @ self.theta
            # MSE损失
            mse_loss = torch.mean((y_pred - y) ** 2)
            # 任务：补全l2_term, l1_term, reg_loss
            # L2 正则化 (不对截距项正则化)
            l2_term = None # TODO: 补全L2正则化项

            # L1 正则化 (不对截距项正则化)
            l1_term = None # TODO: 补全L1正则化项

            # Elastic Net 组合(需要用到l2_term和l1_term)
            reg_loss = None # TODO: 补全Elastic Net正则化项

            # 总损失(MSE + 正则化)
            loss = mse_loss + reg_loss

            # 反向传播
            loss.backward()
            optimizer.step()

        self.theta = self.theta.detach()

    elif self.method == "normal":
        # 正规方程解 (仅支持L2正则，L1无法解析求解)
        X_t = X.t()
        I = torch.eye(X.shape[1], device=self.device)
        if self.fit_intercept:
            I[0, 0] = 0  # 截距项不正则化
        l2_lambda = self.alpha * (1 - self.l1_ratio)
        # 任务：补全正规方程解
        self.theta = None # TODO: 补全正规方程解
    else:
        raise ValueError("method must be 'gd' or 'normal'")
```

- 任务：补充方框中的核心代码部分即可。类似Numpy实现，难度不高。但是需要注意的是Numpy中给出的是正则项的梯度计算，这里直接是正则项即可。最后的损失会通过loss.backward()进行自动计算。

- 注：1）Pytorch中用optimizer.step()替代theta -= lr * grad。但一定要保证optimizer只更新theta，并且lr一致；2）数据类型与精度：Numpy默认是float64，Pytorch默认的是float32，如果不统一，可能会出现细微的数值差异，尤其是学习率比较大的时候。

算法实战：房价预测

- 如之前一样，首先第一步就是导入数据，并查看数据的情况，决定下一步该干什么。

```
# 读取数据
import pandas as pd
data = pd.read_csv('housing.csv')

# 查看信息
data.info()
data.head()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20640 entries, 0 to 20639
Data columns (total 10 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   longitude           20640 non-null  float64
 1   latitude            20640 non-null  float64
 2   housing_median_age  20640 non-null  float64
 3   total_rooms         20640 non-null  float64
 4   total_bedrooms      20433 non-null  float64
 5   population          20640 non-null  float64
 6   households          20640 non-null  float64
 7   median_income       20640 non-null  float64
 8   median_house_value  20640 non-null  float64
 9   ocean_proximity     20640 non-null  object
dtypes: float64(9), object(1)
memory usage: 1.6+ MB
```

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	median_hous
0	-122.23	37.88	41.0	880.0	129.0	322.0	126.0	8.3252	45
1	-122.22	37.86	21.0	7099.0	1106.0	2401.0	1138.0	8.3014	35
2	-122.24	37.85	52.0	1467.0	190.0	496.0	177.0	7.2574	3
3	-122.25	37.85	52.0	1274.0	235.0	558.0	219.0	5.6431	3
4	-122.25	37.85	52.0	1627.0	280.0	565.0	259.0	3.8462	3

- 可以看见households存在null数据，ocean_proximity的数据类型为object，需要转换为OneHot编码。

算法实战：房价预测

- 根据上面的分析，进行数据预处理，并进行数据集划分（训练集、测试集）。

```python
from sklearn.preprocessing import OneHotEncoder
from sklearn.model_selection import train_test_split


# 1. 处理缺失值：用中位数填补 total_bedrooms
data['total_bedrooms'] = data['total_bedrooms'].fillna(data['total_bedrooms'].median())

# 2. 独热编码处理类别变量 ocean_proximity
encoder = OneHotEncoder(sparse_output=False)
ocean_encoded = encoder.fit_transform(data[['ocean_proximity']])
# 只用类别名作为列名
ocean_categories = [str(cat) for cat in encoder.categories_[0]]
ocean_df = pd.DataFrame(ocean_encoded, columns=ocean_categories)
data = pd.concat([data.drop('ocean_proximity', axis=1), ocean_df], axis=1)

# 检查是否有非数值型数据
data.info()

# 3. 标签与特征分离
X = data.drop('median_house_value', axis=1)
y = data['median_house_value']

# 4. 划分数据集
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# 查看划分后的形状
print(f"X_train shape: {X_train.shape}, X_test shape: {X_test.shape}")
print(f"y_train shape: {y_train.shape}, y_test shape: {y_test.shape}")
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20640 entries, 0 to 20639
Data columns (total 14 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   longitude           20640 non-null  float64
 1   latitude            20640 non-null  float64
 2   housing_median_age  20640 non-null  float64
 3   total_rooms         20640 non-null  float64
 4   total_bedrooms      20640 non-null  float64
 5   population          20640 non-null  float64
 6   households          20640 non-null  float64
 7   median_income       20640 non-null  float64
 8   median_house_value  20640 non-null  float64
 9   <1H OCEAN           20640 non-null  float64
 10  INLAND              20640 non-null  float64
 11  ISLAND              20640 non-null  float64
 12  NEAR BAY            20640 non-null  float64
 13  NEAR OCEAN          20640 non-null  float64
dtypes: float64(14)
memory usage: 2.2 MB
X_train shape: (16512, 13), X_test shape: (4128, 13)
y_train shape: (16512,), y_test shape: (4128,)
```

- 经过Onehot独热编码之后，新增几列，改几列的值全部转换为0/1，并且Null全部被消除，训练机划分数据形状也正常，可以进行下一步。

▶ 算法实战：房价预测

➢ 定义模型进行训练、预测，最后通过R^2，RMSE指标评估效果。【这里没有进行特征工程，会导致模型的性能效果不太好】

```python
# 5. 使用自定义线性回归模型
model = MyLinearRegression(method="normal")  # 使用正规方程法，更稳定
model.fit(X_train, y_train)

# 使用L1正则化的线性回归
from sklearn.linear_model import Lasso
lasso_model = Lasso(alpha=0.1)  # alpha是正则化强度
lasso_model.fit(X_train, y_train)

# 使用L2正则化的线性回归
from sklearn.linear_model import Ridge
ridge_model = Ridge(alpha=1.0)  # alpha是正则化强度
ridge_model.fit(X_train, y_train)

# 使用Elastic Net正则化的线性回归
from sklearn.linear_model import ElasticNet
elastic_model = ElasticNet(alpha=0.1, l1_ratio=0.5)
elastic_model.fit(X_train, y_train)

# random forest回归
from sklearn.ensemble import RandomForestRegressor
rf_model = RandomForestRegressor(n_estimators=100, random_state=42)
rf_model.fit(X_train, y_train)
```

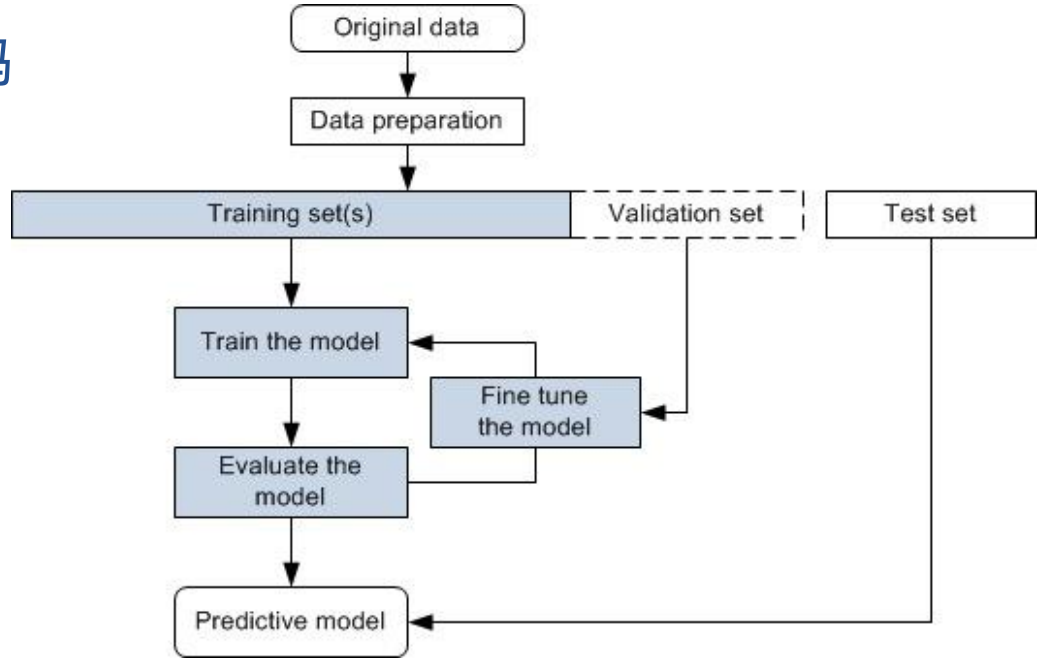
```python
from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score

# 6. 预测和评估
y_pred = model.predict(X_test)
r2 = r2_score(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
print(f"R² Score: {r2:.4f}")
print(f"RMSE: {rmse:.2f}")

print("============================")

y_lasso_pred = lasso_model.predict(X_test)
r2_lasso = r2_score(y_test, y_lasso_pred)
mse_lasso = mean_squared_error(y_test, y_lasso_pred)
rmse_lasso = np.sqrt(mse_lasso)
print(f"Lasso R² Score: {r2_lasso:.4f}")
print(f"Lasso RMSE: {rmse_lasso:.2f}")

print("============================")

y_ridge_pred = ridge_model.predict(X_test)
r2_ridge = r2_score(y_test, y_ridge_pred)
mse_ridge = mean_squared_error(y_test, y_ridge_pred)
rmse_ridge = np.sqrt(mse_ridge)
print(f"Ridge R² Score: {r2_ridge:.4f}")
print(f"Ridge RMSE: {rmse_ridge:.2f}")

print("============================")
y_elastic_pred = elastic_model.predict(X_test)
r2_elastic = r2_score(y_test, y_elastic_pred)
mse_elastic = mean_squared_error(y_test, y_elastic_pred)
rmse_elastic = np.sqrt(mse_elastic)
print(f"Elastic Net R² Score: {r2_elastic:.4f}")
print(f"Elastic Net RMSE: {rmse_elastic:.2f}")

print("============================")
y_rf_pred = rf_model.predict(X_test)
r2_rf = r2_score(y_test, y_rf_pred)
mse_rf = mean_squared_error(y_test, y_rf_pred)
rmse_rf = np.sqrt(mse_rf)
print(f"Random Forest R² Score: {r2_rf:.4f}")
print(f"Random Forest RMSE: {rmse_rf:.2f}")
```

```
R² Score: 0.6254
RMSE: 70060.52
============================
Lasso R² Score: 0.6254
Lasso RMSE: 70060.68
============================
Ridge R² Score: 0.6253
Ridge RMSE: 70070.20
============================
Elastic Net R² Score: 0.6248
Elastic Net RMSE: 70120.95
============================
Random Forest R² Score: 0.8169
Random Forest RMSE: 48977.75
```

● 可见L1，L2正则化对于该任务的提升效率不大，而比较复杂的模型的拟合效果较好。

▶ 小结

- 模型实现：
  - 主要的方法：初始化参数（__init__），拟合方法（fit），预测方法（predict），评估指标（score）
- 模型实战：
  - 数据准备：导入数据、清洗缺失值、特征选择/编码
  - 数据划分：训练集 / 测试集（常用70%/30%或80%/20%）。
  - 特征预处理：标准化/归一化/去NULL值/独热编码等。
  - 模型训练：1）正规方程法：一次计算直接得到参数；2）梯度下降法：迭代更新最佳参数。
  - 模型评估：MSE、RMSE（误差衡量）；R^2（解释能力）



线性回归类的实现核心是完成参数初始化、用正规方程或梯度下降在 fit() 中求解系数、在 predict() 中计算预测值，并提供 MSE、RMSE、R^2等评估方法。

目录章节

CONTENTS

总结

线性回归的概念与原理

- ✓ 线性回归通过线性方程刻画自变量与因变量之间的关系，用最小化残差平方和的方法估计参数。一元回归描述单一特征与目标的关系，多元回归则同时考虑多个特征。
- ✓ 其核心思想是找到一条（或一个超平面）最佳拟合数据，使预测值尽量接近真实值
- ✓ 模型假设：1）线性关系；2）独立性；3）方差齐性；4）正态分布。

线性回归的求解与评估

- ✓ 线性回归的求解方法主要包含两大类：1）梯度下降法；2）正规方程法。当参数训练完成之后，通过MSE，RMSE，R^2等指标进行评估，不同的指标具有不同的代表意义。

线性回归的扩展与改进

- ✓ 线性回归的扩展与改进主要有：1）添加正则项；2）添加扩展特征空间提升特征表达能力；3）针对特殊需求发展变种（加权、鲁棒、贝叶斯、GLM、在线）等。

线性回归通过最小化残差平方和拟合自变量与因变量的线性关系，可用正规方程或梯度下降求解，并用不同指标评估性能，且可扩展至正则化与非线性模型以提升泛化能力。

感谢聆听

Personal Website: https://www.miaopeng.info/

Email: miaopeng@stu.scu.edu.cn

Github: https://github.com/MMeowhite

Youtube: https://www.youtube.com/@pengmiao-bmm