线性回归

- 评估方法
  - 最常用的评估项 R^2：$1-\frac{\sum_{i=1}^{m}(\hat{y}_i-y_i)^2}{\sum_{i=1}^{m}(y_i-\bar{y})^2}$（残差平方和）（类似方差项）
  - R^2 的取值越接近于1我们认为模型拟合的越好

线性回归

- 梯度下降

  - 引入：当我们得到了一个目标函数后，如何进行求解？
    直接求解？（并不一定可解，线性回归可以当做是一个特例）

  - 常规套路：机器学习的套路就是我交给机器一堆数据，然后告诉它
    什么样的学习方式是对的（目标函数），然后让它朝着这个方向去做

  - 如何优化：一口吃不成个胖子，我们要静悄悄的一步步的完成迭代
    （每次优化一点点，累积起来就是个大成绩了）

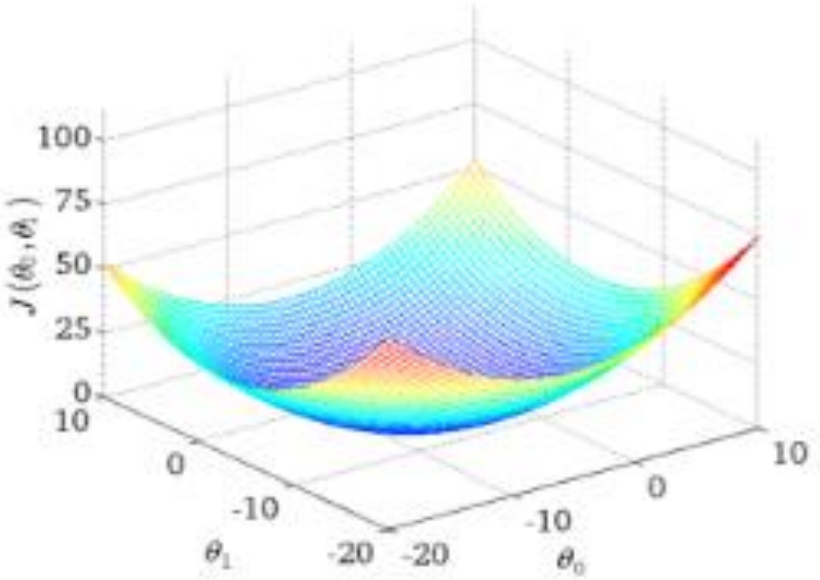
线性回归

- 梯度下降

  - 目标函数： $J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)$

  - 寻找山谷的最低点，也就是我们的目标函数终点
    （什么样的参数能使得目标函数达到极值点）

  - 下山分几步走呢？（更新参数）
    （1）：找到当前最合适的方向
    （2）：走那么一小步，走快了该”跌倒 “ 了
    （3）：按照方向与步伐去更新我们的参数



线性回归

- 梯度下降，目标函数： $J(\theta)=\frac{1}{2m}\sum_{i=1}^{m}(y^i-h_\theta(x^i))^2$

  - 批量梯度下降： $\frac{\partial J(\theta)}{\partial \theta_j}=-\frac{1}{m}\sum_{i=1}^{m}(y^i-h_\theta(x^i))x_j^i \quad \theta_j^{'}=\theta_j+\frac{1}{m}\sum_{i=1}^{m}(y^i-h_\theta(x^i))x_j^i$

    （容易得到最优解，但是由于每次考虑所有样本，速度很慢）

  - 随机梯度下降： $\theta_j^{'}=\theta_j+(y^i-h_\theta(x^i))x_j^i$

    （每次找一个样本，迭代速度快，但不一定每次都朝着收敛的方向）

  - 小批量梯度下降法： $\theta_j:=\theta_j-\alpha\frac{1}{10}\sum_{k=i}^{i+9}(h_\theta(x^{(k)})-y^{(k)})x_j^{(k)}$

    （每次更新选择一小部分数据来算，实用！）

线性回归

- 梯度下降
  - 学习率（步长）：对结果会产生巨大的影响，一般小一些
  - 如何选择：从小的时候，不行再小
  - 批处理数量：32，64，128都可以，很多时候还得考虑内存和效率