中國農業大學
China Agricultural University

人工智能——回归

胡标

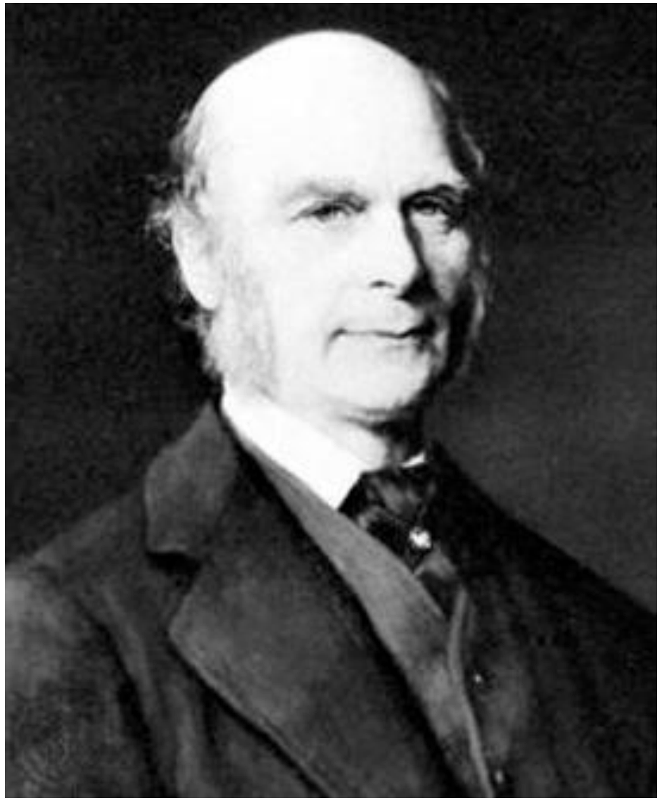
目录
Contents

回归概述

中国农业大学
China Agricultural University

- “**回归**”是由英国著名生物学家兼统计学家高尔顿(Francis Galton)在研究人类遗传问题时提出来的

- 高尔顿搜集了 1078 对父亲及其儿子的身高数据，对试验数据进行了深入的分析，发现了一个很有趣的现象



Francis Galton

线性回归

- 如果说到统计研究和机器学习中最常使用的模型，那么必然会提到线性回归
- 在现实生活中，往往需要分析若干变量之间的关系。这种分析不同变量之间存在关系的研究叫回归分析
- 刻画不同变量之间关系的模型被称为回归模型。如果这个模型是线性的，则称为线性回归模型
- 回归方法是一种对数值型连续随机变量进行预测和建模的监督学习算法
- 许多更高级的机器学习方法被视为线性回归的延伸。因此，理解好这一简单模型将为将来更复杂的学习打下良好基础

线性回归模型

输入：数据$Z = \{(x_1, y_1), \ldots, (x_n, y_n)\}$，其中$x_i \in \mathbb{R}^d$, $y_i \in \mathbb{R}$

输出：一个线性 $f_\beta(x) = \beta^T x + b$，使得$y_i \approx \beta^T x_i + b$

考虑线性函数 $f_\beta(x)$的空间，其定义为

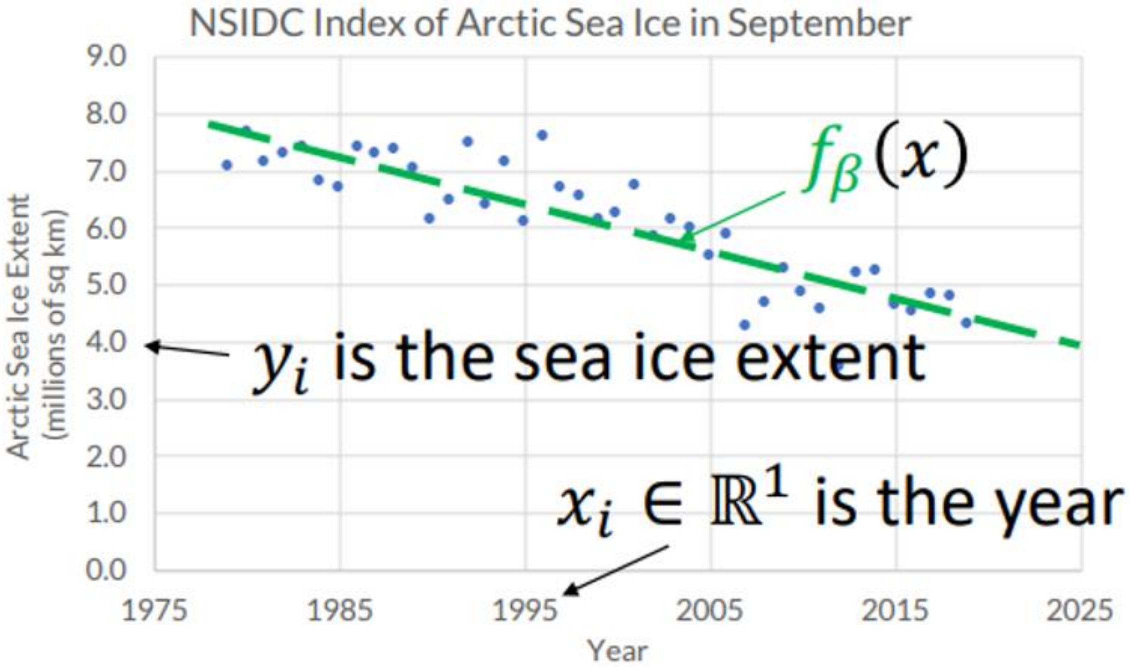
$$f_\beta(x) = \beta^T x + b = [\beta_1 \quad \ldots \quad \beta_d] \begin{bmatrix} x_1 \\ \vdots \\ x_d \end{bmatrix} + b$$

$x \in \mathbb{R}^d$ 称为输入（又称特征或协变量）

$\beta \in \mathbb{R}^d$ 称为参数（又称参数向量）

$y = f_\beta(x) + b$称为标签（又称输出或响应）

- 如果样本只有一维特征，即$d = 1$， $y = \beta_1 x + b$
- 如果样本有两维特征，即$d = 2$， $y = \beta_1 x_1 + \beta_2 x_2 + b$
- 如果样本有d维特征， $y = \beta_1 x_1 + \beta_2 x_2 + \cdots \beta_d x_d + b$
- y是预测值，β是权重，b 是偏置（截距）



波士顿房价预测

- 波士顿房价：该数据收集于1978年，每506个条目代表有关来自波士顿各个郊区的13个特征的汇总信息

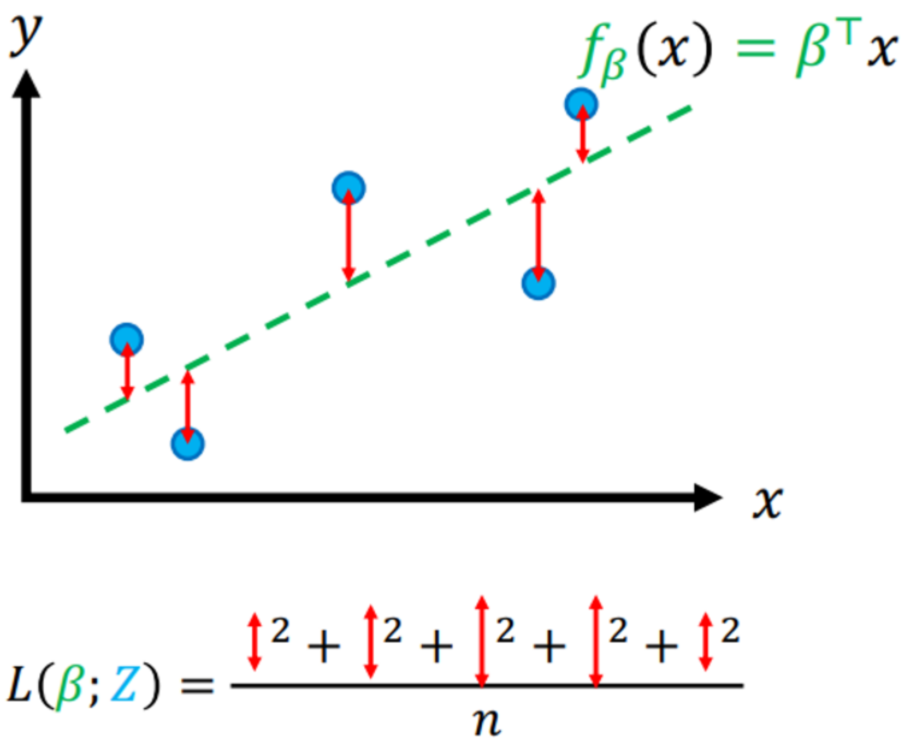
1、CRIM：城镇的人均犯罪率
2、ZN：大于25,000平方英尺的地块的住宅用地比例。
3、INDUS：每个镇的非零售业务英亩的比例。
4、CHAS：查尔斯河虚拟变量（如果环河，则等于1；否则等于0)
5、NOX：一氧化氮的浓度（百万分之几)
6、RM：每个住宅的平均房间数
7、AGE：1940年之前建造的自有住房的比例
8、DIS：到五个波士顿就业中心的加权距离
9、RAD：径向公路通达性的指标
10、TAX：每$ 10,000的全值财产税率
11、PTRATIO：各镇的师生比率
12、B：计算方法为1000（Bk-0.63）^{2}，其中Bk是按城镇划分的非裔美国人的比例
13、LSTAT：底层人口的百分比

损失函数的选择

当$(y_i - \beta^T x_i - b)^2$ 很小的时候有$y_i \approx \beta^T x_i + b$

因此引入均方误差（MSE）

$$L(\beta; Z) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \beta^T x_i - b)^2$$



其具有计算方便，实用性强的优点

优化问题

输入：数据$Z=\{(x_1,y_1),\dots,(x_n,y_n)\}$，其中$x_i\in\mathbb{R}^d$，$y_i\in\mathbb{R}$

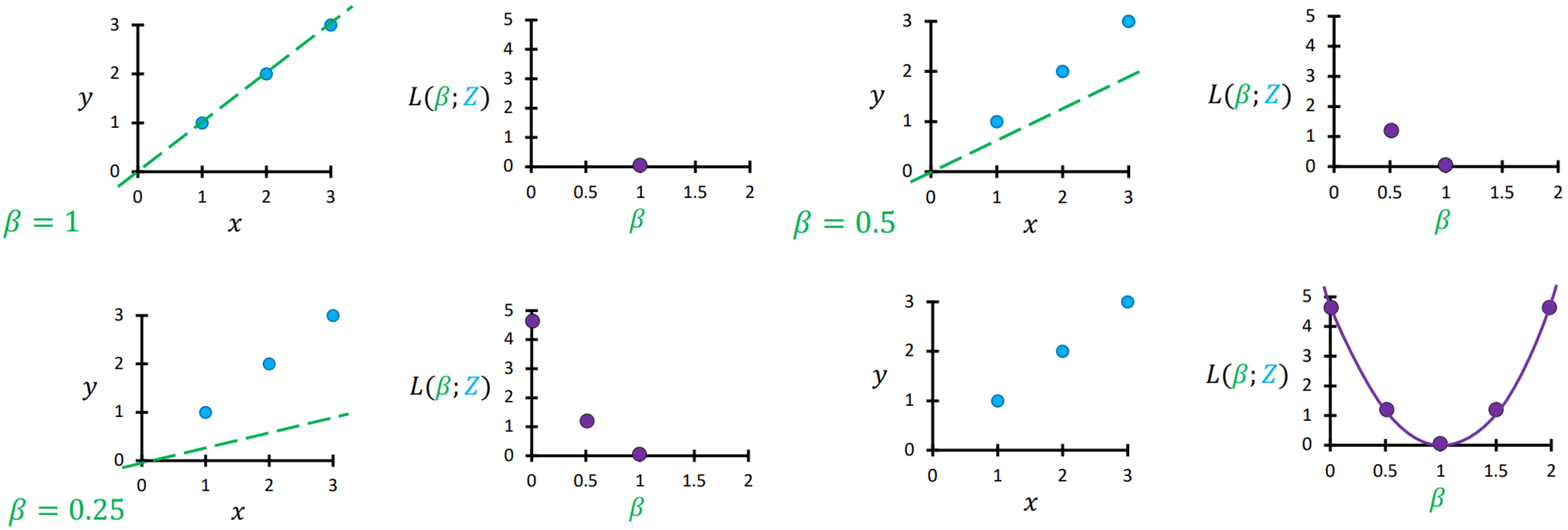
输出：使MSE最小化的线性函数$f_\beta=\beta^T x_i+b$

$$\boldsymbol{L}(\boldsymbol{\beta};\boldsymbol{Z})=\frac{\mathbf{1}}{\boldsymbol{n}}\sum_{\boldsymbol{i}=\mathbf{1}}^{\boldsymbol{n}}\left(\boldsymbol{y}_{\boldsymbol{i}}-\boldsymbol{\beta}^{\boldsymbol{T}}\boldsymbol{x}_{\boldsymbol{i}}-\boldsymbol{b}\right)^{\mathbf{2}}$$

$$\begin{aligned}\beta(Z)&=\underset{\beta\in\mathbb{R}^d}{\arg\min}\,\boldsymbol{L}(\boldsymbol{\beta};\boldsymbol{Z})\\&=\underset{\beta\in\mathbb{R}^d}{\arg\min}\,\frac{\mathbf{1}}{\boldsymbol{n}}\textstyle\sum_{\boldsymbol{i}=\mathbf{1}}^{\boldsymbol{n}}\left(\boldsymbol{y}_{\boldsymbol{i}}-\boldsymbol{\beta}^{\boldsymbol{T}}\boldsymbol{x}_{\boldsymbol{i}}-\boldsymbol{b}\right)^{\mathbf{2}}\end{aligned}$$

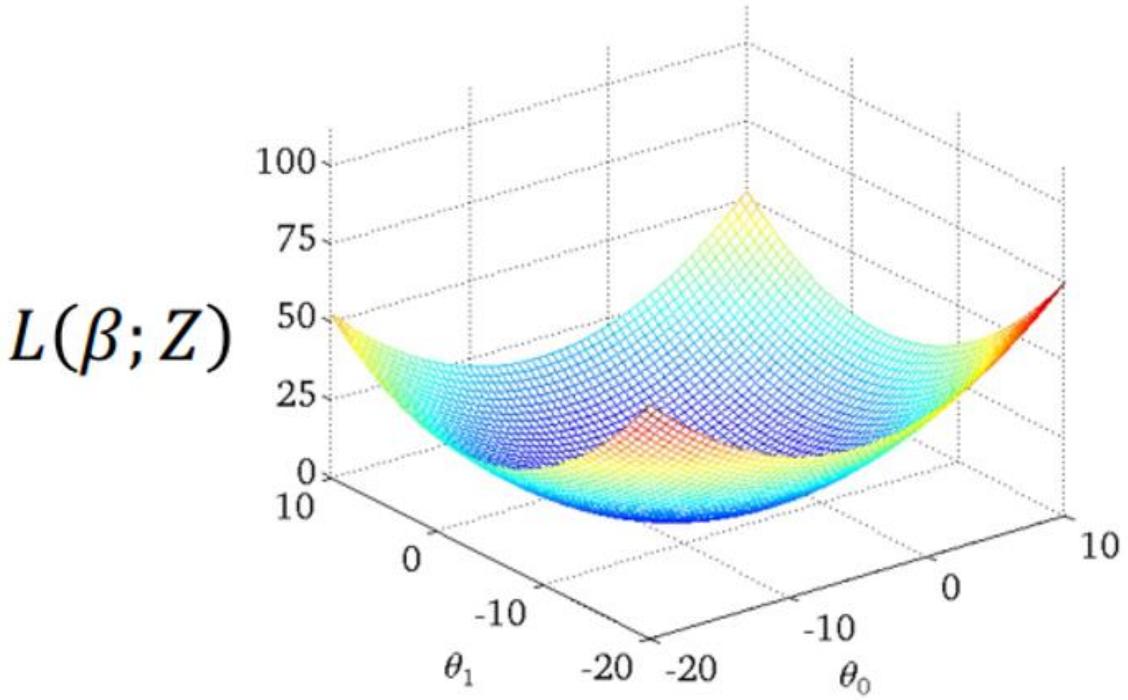
凸优化问题

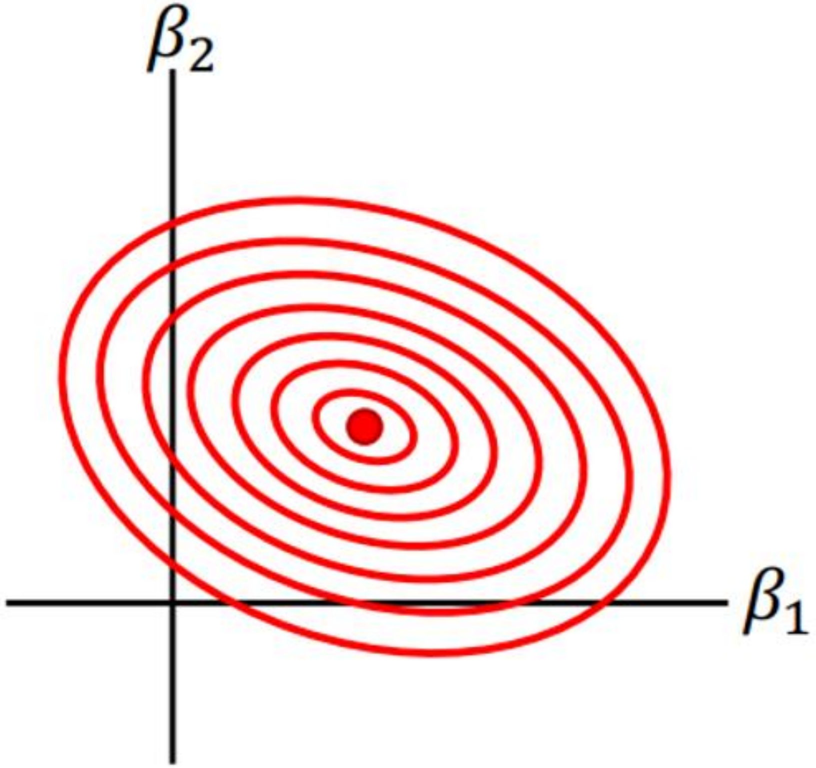
$x_i \in \mathbb{R}$, $y_i \in \mathbb{R}$



最小化均方误差的直观理解

均方误差函数通常是凸的（碗状的）





其它损失函数

- 平均绝对误差：$\frac{1}{n}\sum_{i=1}^{n}|\hat{y_i}-y_i|$
- 平均相对误差：$\frac{1}{n}\sum_{i=1}^{n}\frac{|\hat{y_i}-y_i|}{|y_i|}$
- 决定系数（R2 得分，越高越好,当为1时最佳）：$1-\frac{MSE}{Variance}$
- 皮尔逊相关性：$\frac{1}{n}\sum_{i=1}^{n}\frac{|(\hat{y_i}-\hat{\mu})(y_i-\hat{\mu})|}{\hat{\sigma}\sigma}$

小故事：关于“最小二乘法”

- 1801年，意大利天文学家朱赛普·皮亚齐发现了第一颗小行星谷神星。经过40天的跟踪观测后，由于谷神星运行至太阳背后，使得皮亚齐失去了谷神星的位置。随后全世界的科学家利用皮亚齐的观测数据开始寻找谷神星，但是根据大多数人计算的结果来寻找谷神星都没有结果。时年24岁的高斯也计算了谷神星的轨道。奥地利天文学家海因里希·奥伯斯根据高斯计算出来的轨道重新发现了谷神星别人问高斯，你用什么方法计算的，高斯说保密藏着掖着长达9年之久，最后高斯将其使用的最小二乘法的方法发表于1809年他的著作《天体运动论》中。

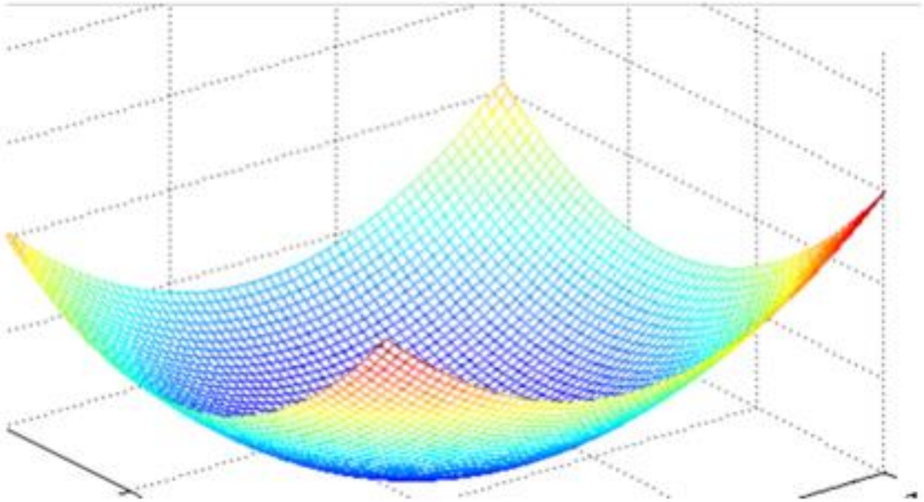
- 而法国科学家阿德里安-马里·勒让德于1806年独立发现最小二乘法"，但因不为世人所知而默默无闻。两人曾为谁最早创立最小二乘法原理发生争执。1829年，高斯提供了最小二乘法的优化效果强于其他方法的证明，见高斯-马尔可夫定理。

目录
Contents

解析解

最小解的梯度等于零：

$$\nabla_\beta L(\hat{\beta}; Z) = 0$$



$$\nabla_\beta L(\beta; Z) = \nabla_\beta \frac{1}{n}\|Y - X\beta\|_2^2 = \nabla_\beta \frac{1}{n}(Y - X\beta)^T(Y - X\beta)$$
$$= \frac{2}{n}[\nabla_\beta (Y - X\beta)^T](Y - X\beta) = -\frac{2}{n}X^T(Y - X\beta)$$
$$= -\frac{2}{n}X^TY + \frac{2}{n}X^TX\beta$$

$$\nabla_\beta L(\beta; Z) = \nabla_\beta \frac{1}{n}\|Y - X\beta\|_2^2 = -\frac{2}{n}X^TY + \frac{2}{n}X^TX\beta$$

设定$\nabla_\beta L(\hat{\beta}; Z) = 0$，$X^TX\hat{\beta} = X^TX$ 则有

若假设X^TX为可逆的，则有

$$\hat{\beta}(Z) = (X^TX)^{-1}X^TX$$

公式求解的问题

- 当特征量的数量很多的时候计算$\hat{\beta}(Z) = (X^T X)^{-1} X^T X$， 可能是困难的
- 计算$(X^T X)^{-1}$的时间复杂度是$\boldsymbol{O}(d^3)$
  - 即使只是储存$X^T X$也需要很多的内存
- 由于 "条件不佳 "造成的数值精度问题
  - 如果$X^T X$ "勉强 "可逆怎么办?
  - $X^T X$在某个维度上具有较大的方差

迭代优化

- 回顾一下，线性回归最大限度地减少了损失$L(\beta; Z) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \beta^T x_i - b)^2$
- 迭代优化 β
  - 初始化 $\beta_1 \leftarrow \mathrm{Init}(\cdots)$
  - 对于一定的迭代次数t，更新$\beta_t \leftarrow \mathrm{Step}(\cdots)$
  - 返回β_t

迭代优化

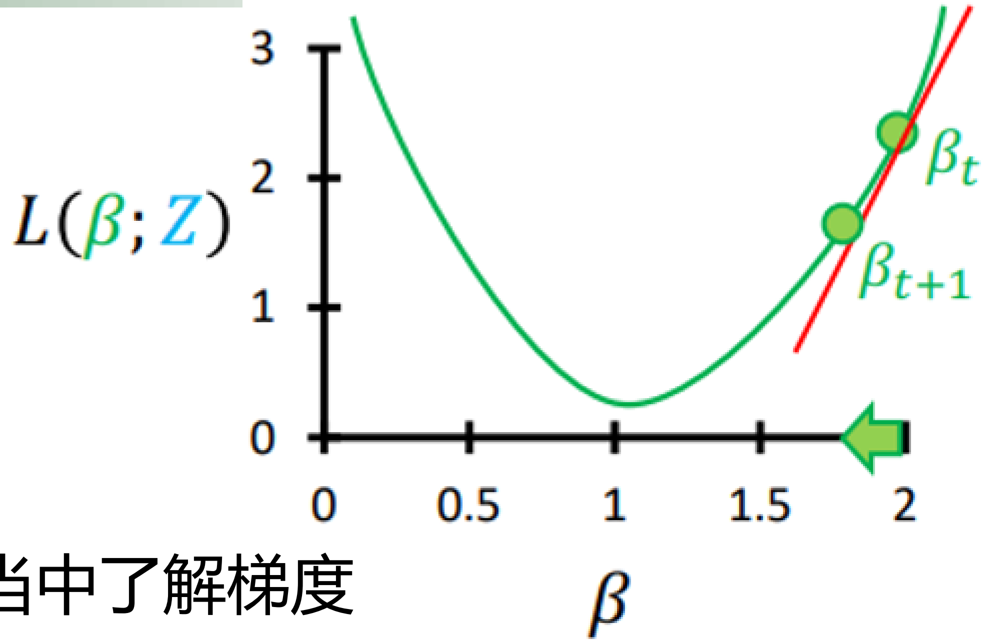
- **全局搜索：** 随机的尝试β并选择最佳值
  - β_t与β_{t-1}之间是相互独立的
  - 非结构化，需要花费较长的时间（特别是在高维的情况下）
- **局部搜索：** 从一些初始值β开始并且进行局部的调整
  - β_t的计算是基于β_{t-1}
  - 什么是局部调整，我们又应当如何寻找好的调整
- **梯度下降：** 根据损失函数$L(\beta;Z)$的梯度$\nabla_\beta L(\beta;Z)$更新参数β
  - $\beta_{t+1} \leftarrow \beta_t - \alpha \cdot \nabla_\beta L(\beta_t;Z), \alpha \in \mathbb{R}$是一个称之为学习率的超参数
- **直观感受：** 梯度是$L(\beta;Z)$ 沿着β变化最快的方向

梯度下降

初始化参数$\beta_1 = 0$

反复如下操作直到收敛

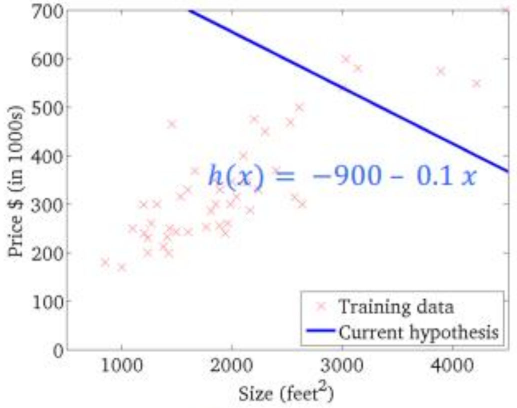
$$\beta_{t+1} \leftarrow \beta_t - \alpha \cdot \nabla_\beta L(\beta_t; Z)$$



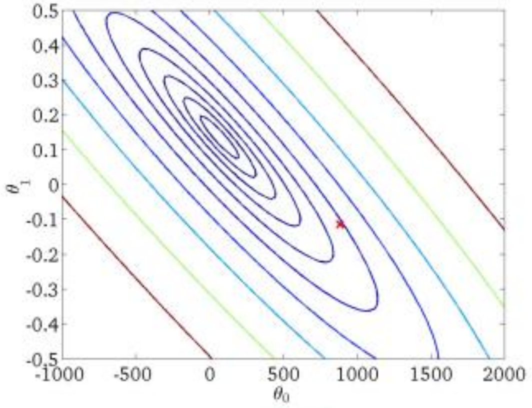
对于线性回归，可以从前文的推导当中了解梯度

对于更新$\beta_{t+1} \leftarrow \beta_t - \alpha \cdot \nabla_\beta L(\beta_t; Z)$，在修改$\beta$之前先计算$\nabla_\beta L(\beta; Z)$的所有分量

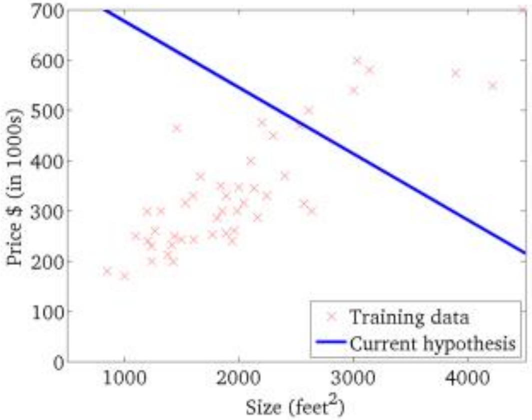
梯度下降



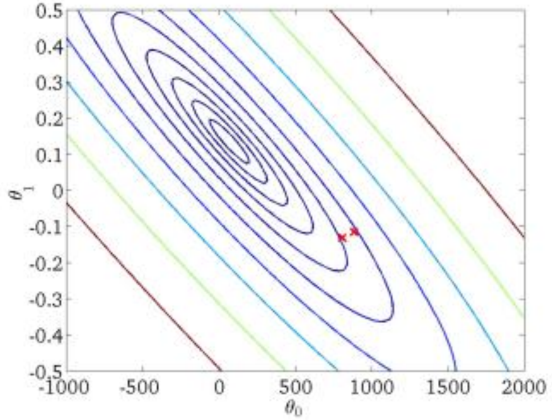
$f_\beta(x)$



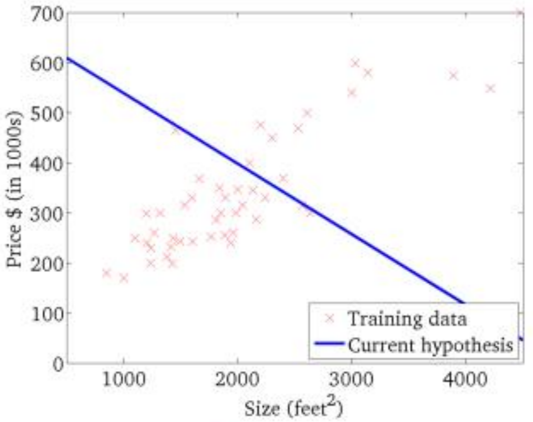
$L(\beta; Z)$



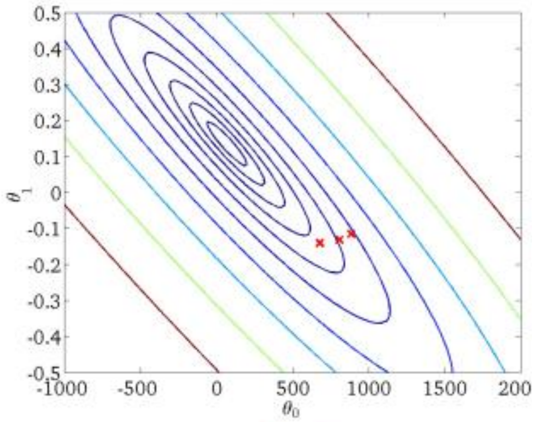
$f_\beta(x)$



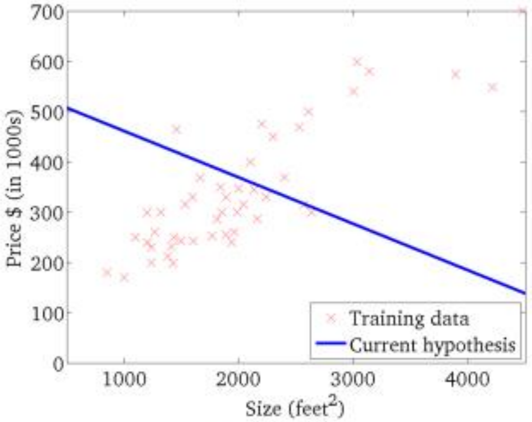
$L(\beta; Z)$



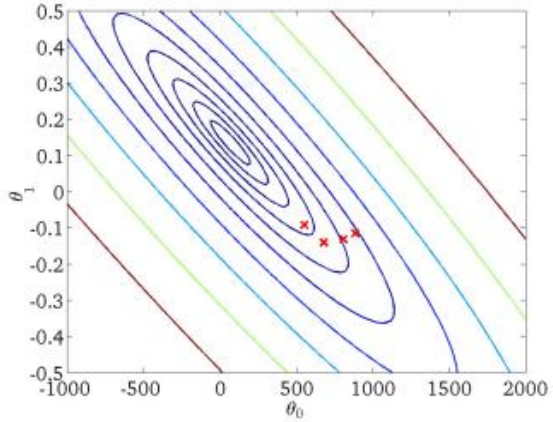
$f_\beta(x)$



$L(\beta; Z)$

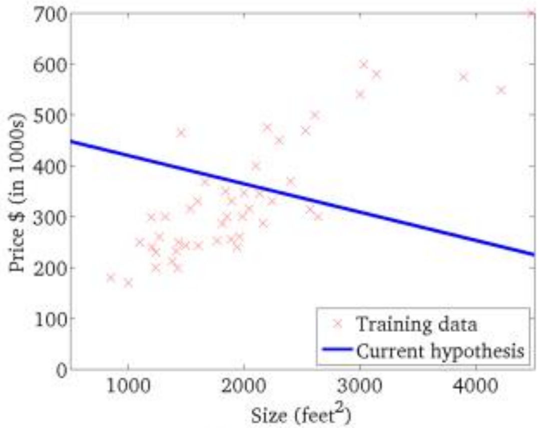


$f_\beta(x)$

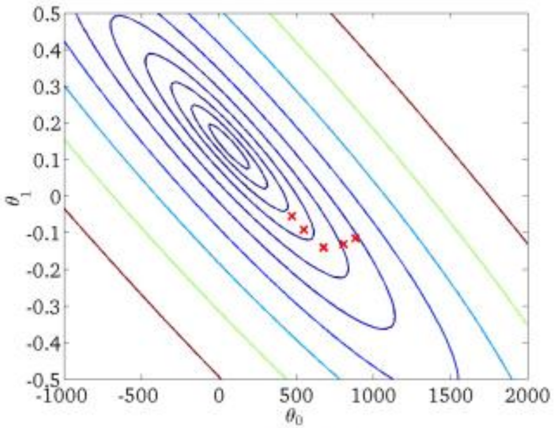


$L(\beta; Z)$

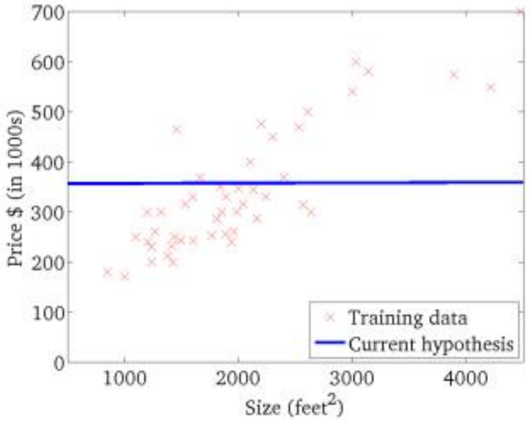
梯度下降



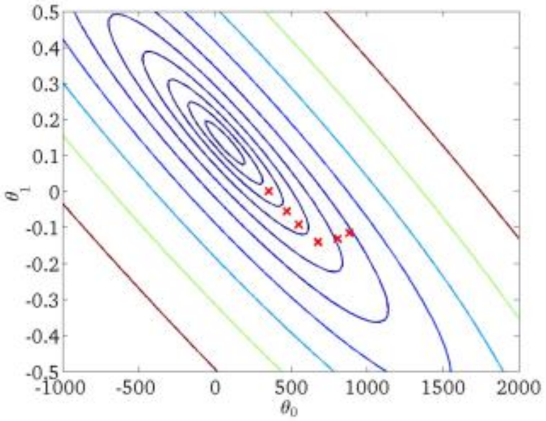
$f_\beta(x)$



$L(\beta; Z)$

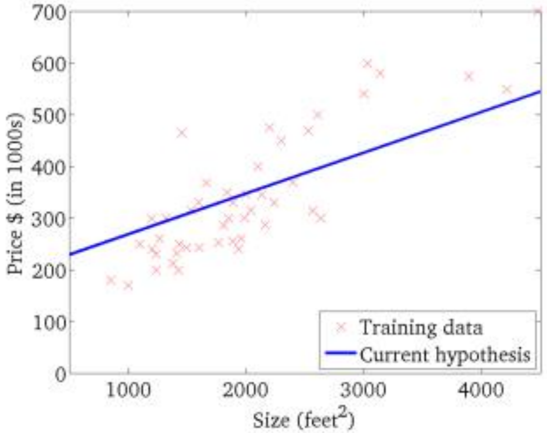


$f_\beta(x)$

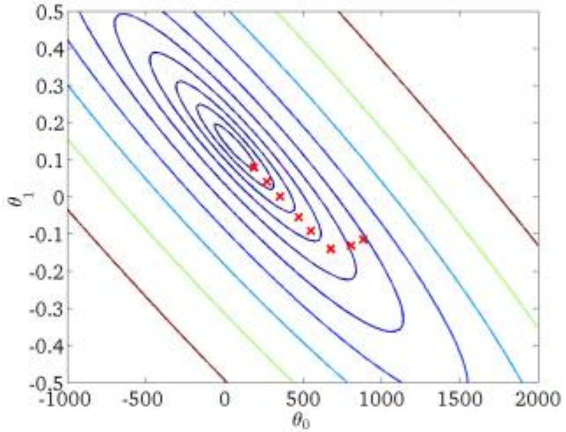


$L(\beta; Z)$

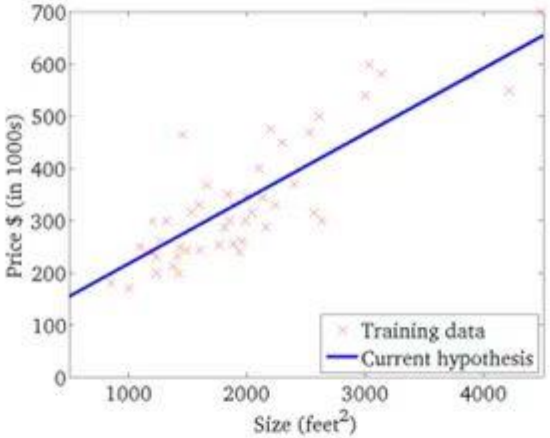
最小化损失函数



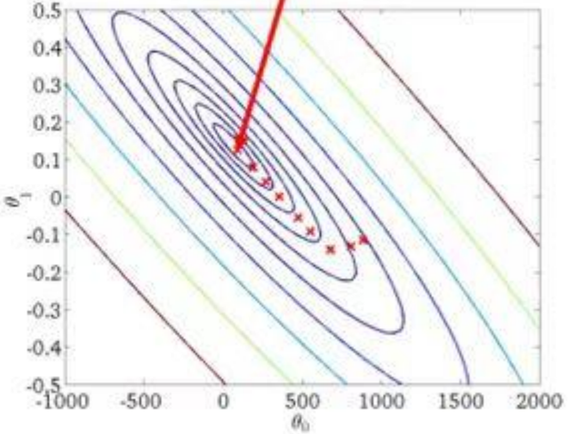
$f_\beta(x)$



$L(\beta; Z)$

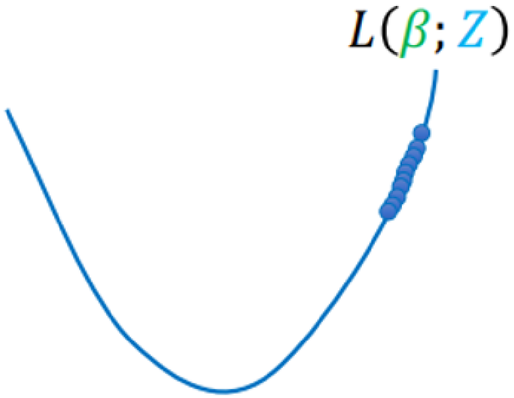


$f_\beta(x)$

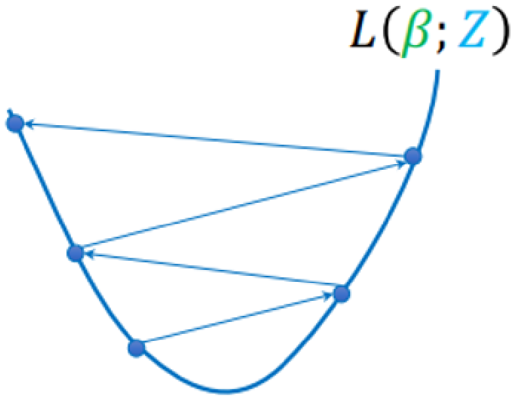


$L(\beta; Z)$

学习率的选择



学习率太小，损失函数减小过慢



学习率太大，损失函数爆炸

绘制损失函数$L(\beta;Z)$与t的关系图以诊断这些问题

- 批量梯度下降(Batch Gradient Descent, BGD)
  - 梯度下降的每一步中，都用到了所有的训练样本
- 随机梯度下降(Stochastic Gradient Descent, SGD)
  - 梯度下降的每一步中，用到一个样本，在每一次计算之后便更新参数，而不需要首先将所有的训练集求和
- 小批量梯度下降(Mini-Batch Gradient Descent, MBGD)
  - 梯度下降的每一步中，用到了一定批量的训练样本

批量梯度下降(Batch Gradient Descent, BGD)

- 梯度下降的每一步中，都用到了所有的训练样本

$$\beta_j := \beta_j - \underbrace{\alpha}_{\text{学习率}} \overbrace{\frac{1}{m}\sum_{i=1}^{m}\left(\left(h\left(x^{(i)}\right) - y^{(i)}\right) \cdot x_j^{(i)}\right)}^{\text{梯度}}$$

(同步更新β_j, $j = 0,1,\dots,n$)

随机梯度下降(Stochastic Gradient Descent, SGD)

- 梯度下降的每一步中，用到一个样本，在每一次计算之后便更新参数，而不需要首先将所有的训练集求和

学习率

$$\beta_j := \beta_j - \alpha \overbrace{\left(\left(h\left(x^{(i)}\right) - y^{(i)}\right) \cdot x_j^{(i)}\right)}^{\text{梯度}}$$

(同步更新β_j, $j = 0,1,\ldots,n$)

□ 小批量梯度下降(Mini-Batch Gradient Descent, MBGD)

- 梯度下降的每一步中，用到了**一定批量**的训练样本
- 每计算常数b次训练实例，便更新一次参数β

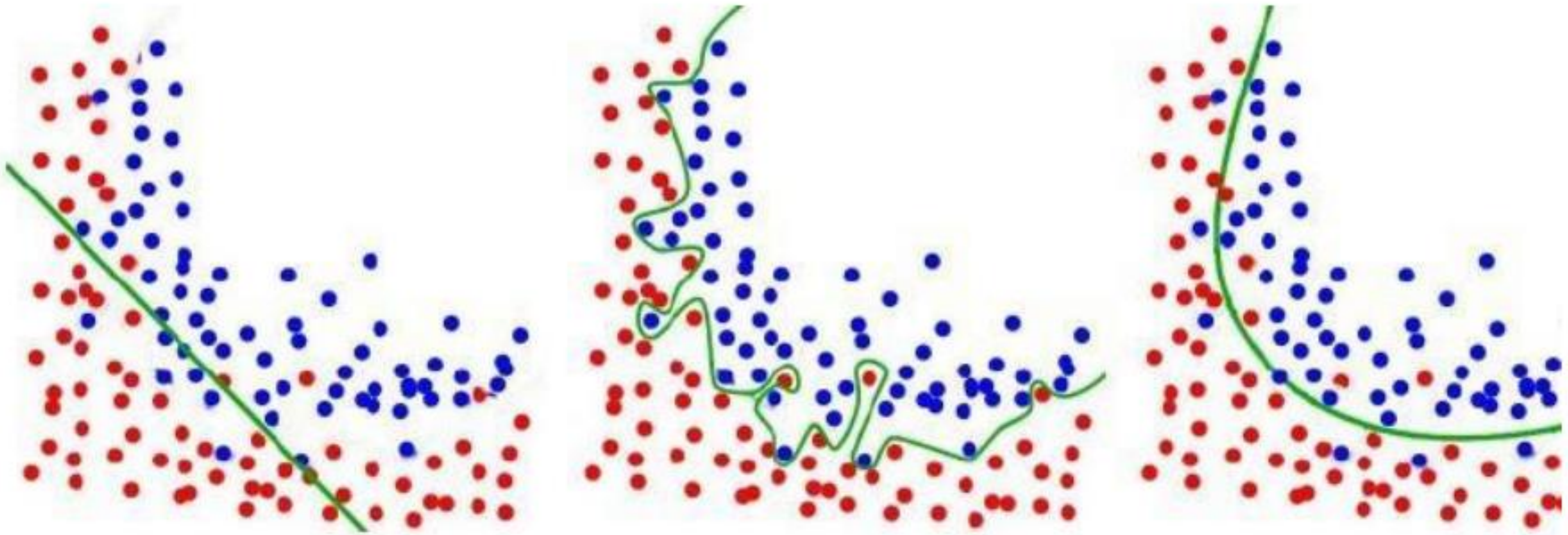
$$\beta_j := \beta_j - \alpha \frac{1}{b} \sum_{k=i}^{i+b-1} \left(\left(h\left(x^{(k)}\right) - y^{(k)} \right) \cdot x_j^{(k)} \right)$$

(同步更新β_j, $j = 0,1,\ldots,n$)

$b = 1$(随机梯度下降,SGD)
$b = m$(批量梯度下降,BGD)
$b = batch\ size$，通常是2的指数倍，常见有32,64,128等(小批量梯度下降,MBGD)

目录
Contents

拟合化的问题



欠拟合　　过拟合　　正合适

正则化

原始的损失+正则化 $$L(\beta;Z)=\frac{1}{n}\sum_{i=1}^{n}(y_i-\beta^T x_i)^2+\lambda\cdot\|\beta\|_2^2$$

λ是一个必须要进行调整的超参数（要求$\lambda\geq 0$）

等价于X的$L2$范数：

$$\|\beta\|_2^2=\sum_{j=1}^{d}\beta_j^2$$

即，将β“拉向”0，λ越大，“拉力”越大

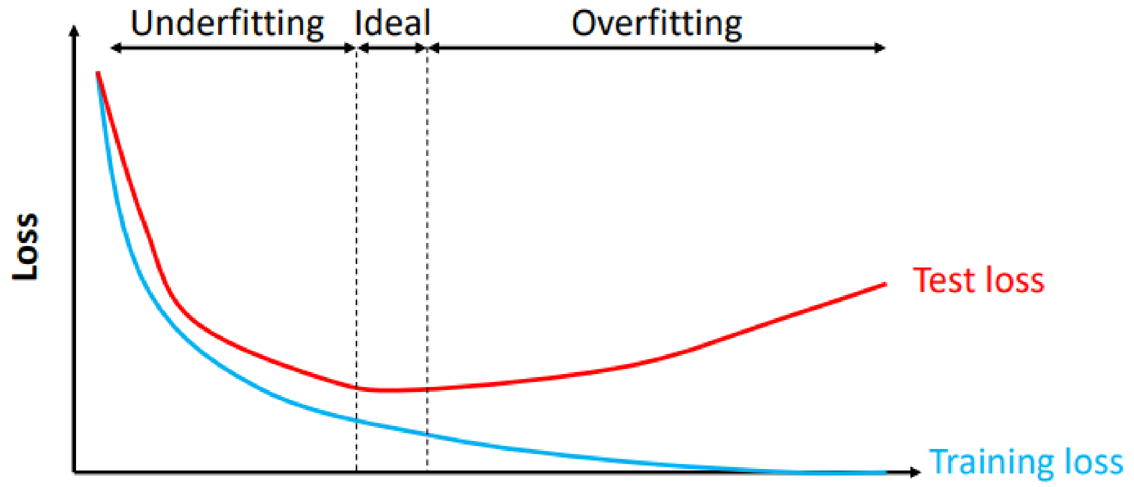
正则化解释

- $L2$正则化的直观理解
  - 为什么它能够起到作用

    鼓励“简单”的函数

    当参数λ趋向无穷，参数β趋向 0

    使用λ调整方差-偏差权衡

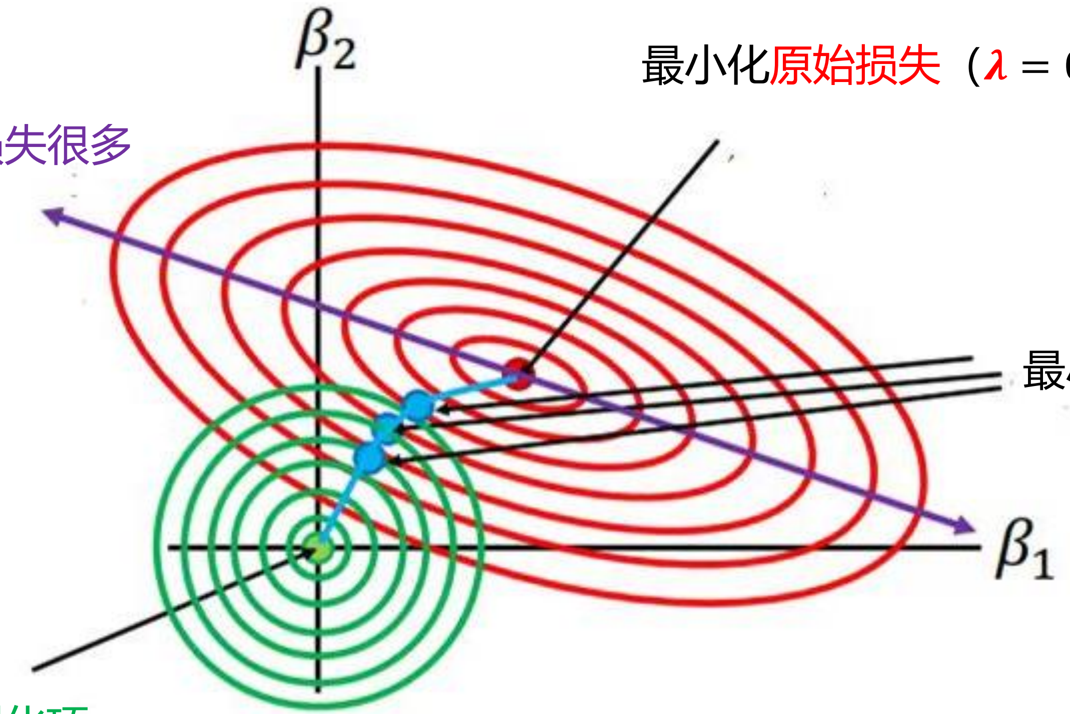


正则化解释

在这一个方向损失很多

即惩罚的更多

最小化原始损失（$\lambda = 0$）



最小化全损失

此时，梯度相等（符号相反）

取舍取决于λ

最小化正则化项

（$\lambda \to \infty$）

$$L(\beta; Z) = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - \beta^T x_i\right)^2 + \lambda \cdot \sum_{j=1}^{d} \beta_j^2$$

正则化解释

中国农业大学 China Agricultural University

L1正则化：$J(w)=\frac{1}{2}\sum_{i=1}^{m}(h(x^{(i)})-y^{(i)})^2+\lambda\sum_{j=1}^{n}|w_j|$，Lasso Regression（Lasso回归）

L2正则化：$J(w)=\frac{1}{2}\sum_{i=1}^{m}(h(x^{(i)})-y^{(i)})^2+\lambda\sum_{j=1}^{n}w_j^2$，Ridge Regression （岭回归）

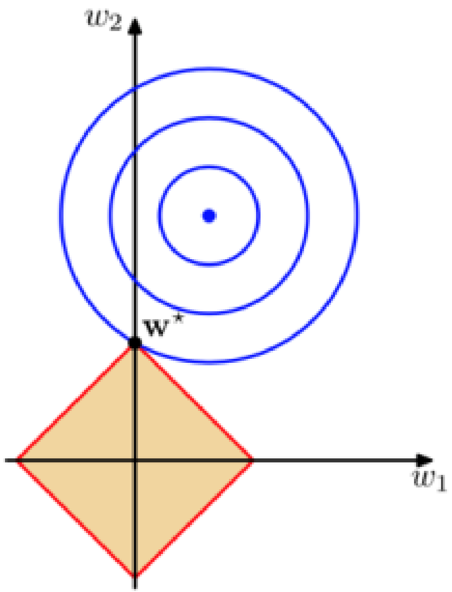
Elastic Net：$J(w)=\frac{1}{2}\sum_{i=1}^{m}(h(x^{(i)})-y^{(i)})^2+\lambda(\rho\cdot\sum_{j=1}^{n}|w_j|+(1-\rho)\cdot\sum_{j=1}^{n}w_j^2)$

(弹性网络)

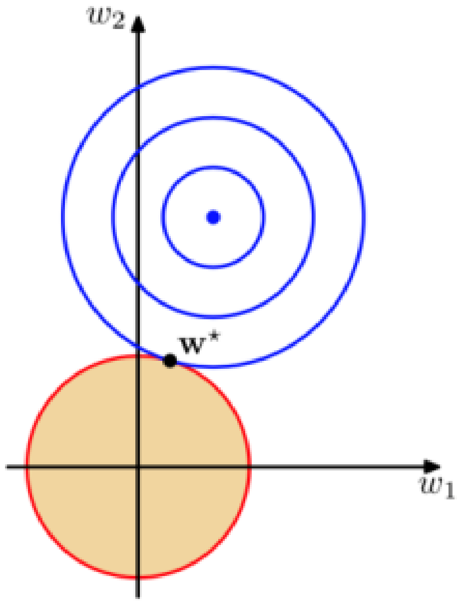
其中：

- λ为正则化系数，调整正则化项与训练误差的比例，λ>0。
- 1≥ρ≥0为比例系数，调整L1正则化与L2正则化的比例。

正则化解释



L1正则化是指在损失函数中加入权值向量w的绝对值之和，L1的功能是使权重稀疏



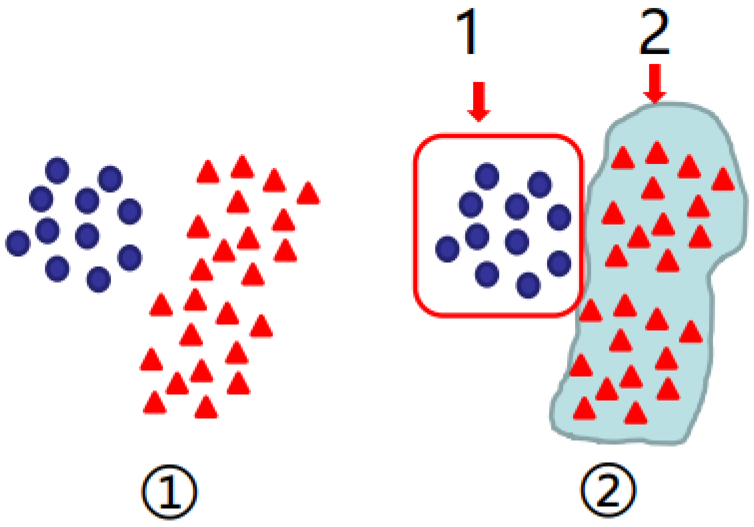
在损失函数中加入权值向量w的平方和，L2的功能是使权重平滑。

L1正则化可以产生稀疏模型 **L2正则化可以防止过拟合**

- 图上面中的蓝色轮廓线是没有正则化损失函数的等高线，中心的蓝色点为最优解，左图、右图分别为L1、L2正则化给出的限制。
- 可以看到在正则化的限制之下，L1正则化给出的最优解w^*是使解更加靠近原点,也就是说L2正则化能降低参数范数的总和。
- L1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L1正则化能使得到的参数稀疏化

目录
Contents

分类问题

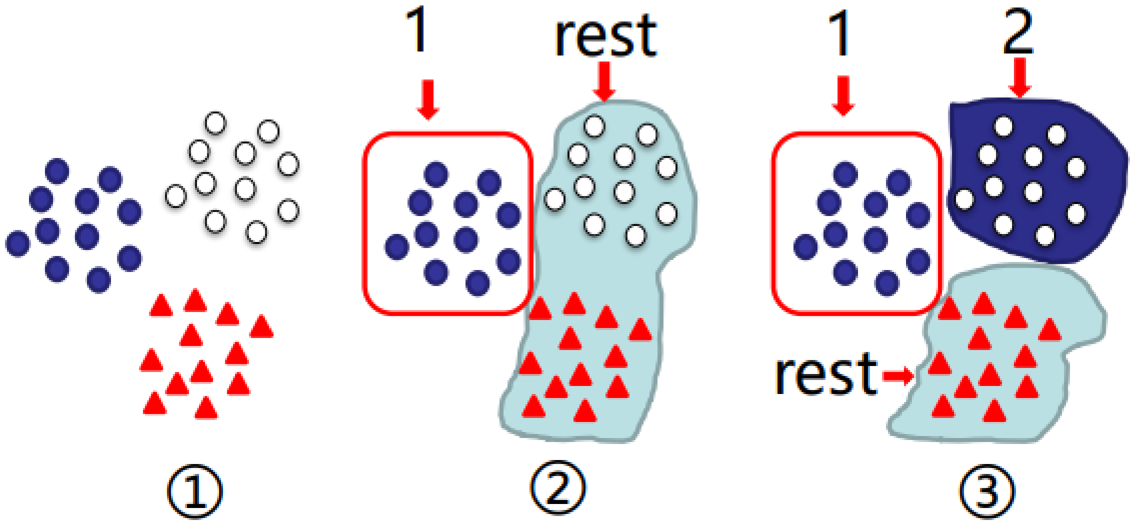


二分类

我们先从用蓝色圆形数据定义为类型1，其余数据为类型2；

只需要分类1次

步骤：①->②



One-vs-All (One-vs-Rest)

一对多 (一对余)

我们先定义其中一类为类型1（正类），其余数据为负类（rest）；接下来去掉类型1数据，剩余部分再次进行二分类，分成类型2和负类；如果有n类，那就需要分类n-1次

步骤：①->②->③->……

目录
Contents

二分类问题

- 线性回归的函数 $h(x) = z = w^T x + b$，范围是$(-\infty, +\infty)$
- 而分类预测结果需要得到[0,1]的概率值。
- 在二分类模型中，事件的几率odds:事件发生与事件不发生的概率之比为$\frac{p}{1-p}$
- 称为事件的发生比(the odds of experiencing an event)
- 其中p为随机事件发生的概率，p的范围为[0,1]。
- 取对数得到:$\log\frac{p}{1-p}$，而$\log\frac{p}{1-p} = z = w^T x + b$
- 求解得到:$p = \frac{1}{1+e^{-w^T x+b}} = \frac{1}{1+e^{-z}}$

二分类问题

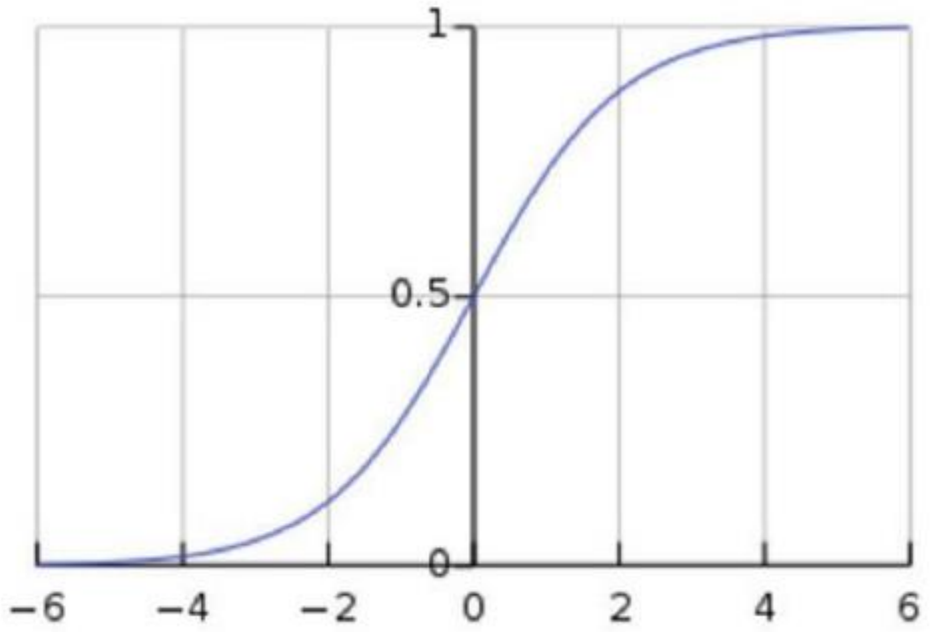
中国农业大学 China Agricultural University

Sigmoid 函数

$\sigma(z)$代表一个常用的逻辑函数（logistic function）为S形函数（Sigmoid function）

则：$\sigma(z) = g(z) = \frac{1}{1+e^{-z}}$　　z=$w^{\mathrm{T}}x + b$

合起来，我们得到逻辑回归模型的假设函数：



$$L(\hat{y}, y) = -y\log(\hat{y}) - (1-y)\log(1-\hat{y})$$

当$\sigma(z)$大于等于0.5时，预测 y=1

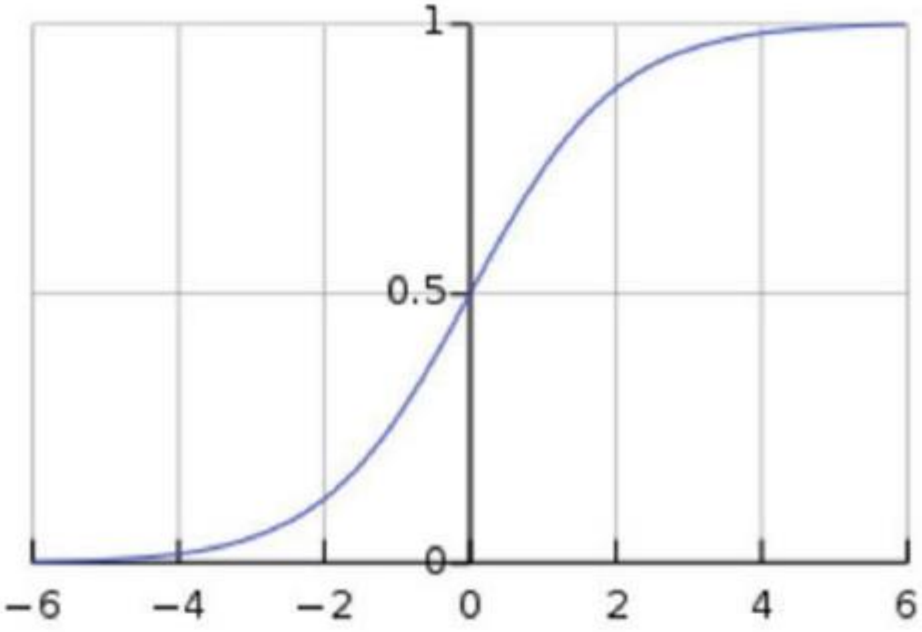
当$\sigma(z)$小于0.5时，预测 y=0

注意：若表达式 $h(x) = z = w_0 + w_1x_1 + w_2x_2 + \ldots + w_nx_n + b = w^{\mathrm{T}}x + b$， 则$b$可以融入到$w_0$，即：$z$=$w^{\mathrm{T}}x$

Sigmoid函数

将z进行逻辑变换：$g(z) = \frac{1}{1+e^{-z}}$

$$
\begin{aligned}
g'(z) &= \left(\frac{1}{1+e^{-z}}\right)' \\
&= \frac{e^{-z}}{(1+e^{-z})^2} \\
&= \frac{1+e^{-z}-1}{(1+e^{-z})^2} \\
&= \frac{1}{(1+e^{-z})}\left(1-\frac{1}{(1+e^{-z})}\right) \\
&= g(z)(1-g(z))
\end{aligned}
$$



$g(z)$

目录
Contents

Logistic函数

假设一个二分类模型：

$$p(y=1|x;w)=h(x)$$

$$p(y=0|x;w)=1-h(x)$$

则：

$$p(y|x;w)=(h(x))^{y}(1-h(x))^{1-y}$$

逻辑回归模型的假设是： $h(x)=g\left(w^{\mathrm{T}}x\right)=g(z)$

其中$z=w^{\mathrm{T}}x$，逻辑函数（**logistic function**)公式为：

$$g(z)=\frac{1}{1+e^{-z}},\ \ g'(z)=g(z)(1-g(z))$$

中国农业大学
China Agricultural University

损失函数

$$L(\hat{y}, y) = -y\log(\hat{y}) - (1-y)\log(1-\hat{y})$$

$\hat{y}$ 表示预测值$h(x)$

y 表示真实值

为了衡量算法在全部训练样本上的表现如何，我们需要定义一个算法的代价函数，算法的代价函数是对m个样本的损失函数求和然后除以m:

代价函数

$$J(w) = \frac{1}{m}\sum_{i=1}^{m} L\left(\hat{y}^{(i)}, y^{(i)}\right) = \frac{1}{m}\sum_{i=1}^{m}\left(-y^{(i)}\log \hat{y}^{(i)} - (1-y^{(i)})\log(1-\hat{y}^{(i)})\right)$$

梯度下降求解过程：

$$w_j := w_j - \alpha \frac{\partial J(w)}{\partial w}$$

$$J(w) = -\frac{1}{m}\sum_{i=1}^{m}(y^{(i)}\log(h(x^{(i)})) + (1 - y^{(i)})\log(1 - h(x^{(i)})))$$

$$\frac{\partial}{\partial w_j}J(w) = \frac{1}{m}\sum_{i=1}^{m}(h(x^{(i)}) - y^{(i)})x_j^{(i)}$$

则：$w_j := w_j - \alpha \frac{1}{m}\sum_{i=1}^{m}(h(x^{(i)}) - y^{(i)})x_j^{(i)}$

梯度下降

求解过程： $\frac{\partial}{\partial w_j}J(w)=\frac{1}{m}\sum_{i=1}^{m}(h(x^{(i)})-y^{(i)})x_j^{(i)}$的推导过程：

$$J(w)=-\frac{1}{m}\sum_{i=1}^{m}(\underline{y^{(i)}\log(h(x^{(i)}))+(1-y^{(i)})\log(1-h(x^{(i)}))})$$

$$\Downarrow$$

$$\begin{aligned}
&y^{(i)}\log(h(x^{(i)}))+(1-y^{(i)})\log(1-h(x^{(i)}))\\
&=y^{(i)}\log(\frac{1}{1+e^{-w^{\mathrm{T}}x^{(i)}}})+(1-y^{(i)})\log(1-\frac{1}{1+e^{-w^{\mathrm{T}}x^{(i)}}})\\
&=-y^{(i)}\log(1+e^{-w^{\mathrm{T}}x^{(i)}})-(1-y^{(i)})\log(1+e^{w^{\mathrm{T}}x^{(i)}})
\end{aligned}$$

求解过程： $\frac{\partial}{\partial w_j}J(w)=\frac{1}{m}\sum_{i=1}^{m}(h(x^{(i)})-y^{(i)})x_j^{(i)}$**的推导过程：**

$$\frac{\partial}{\partial w_j}J(w)=\frac{\partial}{\partial w_j}(-\frac{1}{m}\sum_{i=1}^{m}(-y^{(i)}\log\left(1+e^{-w^{\mathrm{T}}x^{(i)}}\right)-(1-y^{(i)})\log\left(1+e^{w^{\mathrm{T}}x^{(i)}}\right)))$$

$$=-\frac{1}{m}\sum_{i=1}^{m}(-y^{(i)}\frac{-x_j^{(i)}e^{-w^{\mathrm{T}}x^{(i)}}}{1+e^{-w^{\mathrm{T}}x^{(i)}}}-(1-y^{(i)})\frac{x_j^{(i)}e^{w^{\mathrm{T}}x^{(i)}}}{1+e^{w^{\mathrm{T}}x^{(i)}}})$$

$$=-\frac{1}{m}\sum_{i=1}^{m}(y^{(i)}-h(x^{(i)}))x_j^{(i)}$$

$$=\frac{1}{m}\sum_{i=1}^{m}(h(x^{(i)})-y^{(i)})x_j^{(i)}$$

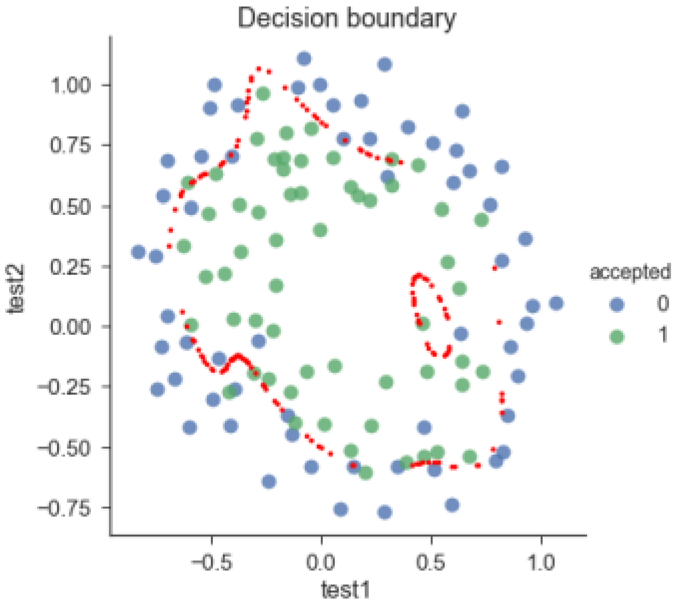
正则化

正则化：目的是**为了防止过拟合**

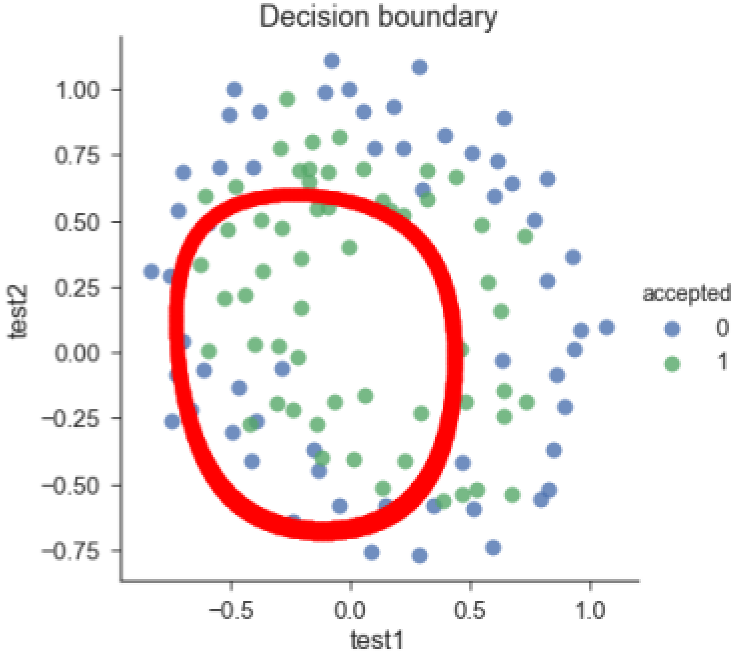
正则化项

$$J(w)=\frac{1}{m}\sum_{i=1}^{m}\left[-y^{(i)}\log\left(h\left(x^{(i)}\right)\right)-\left(1-y^{(i)}\right)\log\left(1-h\left(x^{(i)}\right)\right)\right]+\boxed{\frac{\lambda}{2m}\sum_{j=1}^{n}w_j^2}$$

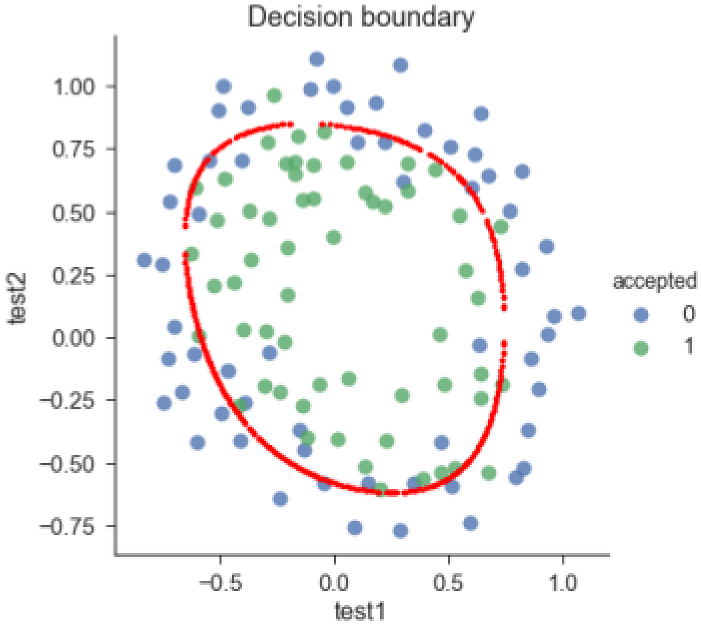
当λ的值开始上升时，降低了方差。



没有正则化，过拟合



正则化过度，欠拟合



适当的正则化

小故事：伯克森与Logit模型

- 约瑟夫·伯克森（Joseph Berkson，1899 – 1982），美国生物统计学家。伯克森早先在哥伦比亚大学攻读物理学，后来又去约翰霍普金斯大学学习医学和统计学；他的博士导师正是里德——里德和佩尔在上世纪20年代研究美国人口增长时发现了Logistic函数。
- 1944年，伯克森提出可以用Logistic函数替代正态分布的累积函数，并模仿Bliss创造的“Probit”一字，将the log of an odd缩写为“Logit”；于是相应的模型便被称为Logit模型。
- 伯克森创造Logit一词可谓一语双关：一方面取其与Logistic形似，凸显出Logit模型与Logistic函数之间的关联；另一方面也有与当时全胜的Probit模型分庭抗礼之意。
- 有意思的是，当Logit模型刚被提出的时候，很多学者并不接受Logit模型；甚至有人认为Logit模型要比Probit模型“低人一等”。这主要是因为，在当时，人们还无法将Logit模型中的随机项与某种特定的分布联系起来——Probit模型中的随机项服从正态分布；相比之下，Probit模型看起来有着更深厚的统计理论基础。
- 1974年，McFadden从随机效用理论出发，将Logit模型与Gumbel分布联系起来；这也奠定了Logit模型的理论基础——McFadden本人因此项发现获得了2000年的诺贝尔经济学奖。

Photo from the Nobel Foundation archive.

James J. Heckman

Prize share: 1/2

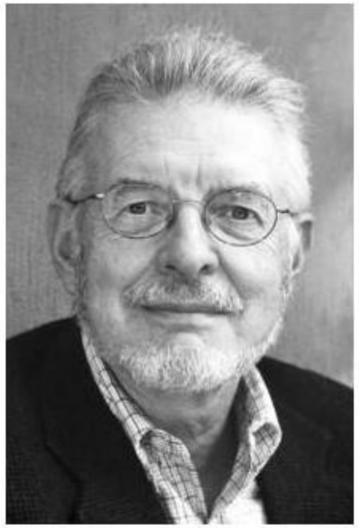


Photo from the Nobel Foundation archive.

Daniel L. McFadden

Prize share: 1/2

梯度下降寻低谷，正则化作舟护渡。
山高水远迷雾中，算法为帆解模糊。