第八周学习笔记—监督学习的总结（一）

2022-06-02

第一章　线性回归

线性回归是比较简单的机器学习算法，很多书籍介绍的第一种机器学习算法就是线性回归算法，我们查阅的中文书籍都是给出线性回归的表达式，然后告诉你怎么求参数最优化。

因此忽视了一些重要的问题。因此，本文重点介绍了平常容易忽视的三类问题:

（1）线性回归的理论依据是什么

（2）过拟合意味着什么

（3）模型优化的方向

1.1　线性回归的理论依据

1.1.1　泰勒公式

若函数 $f(x)$ 在包含 x_0 的某个闭区间 $[a,b]$ 上具有 n 阶导数，且在开区间 (a,b) 上具有 $(n+1)$ 阶导数，则对闭区间 $[a,b]$ 上任意一点 x，成立下式：

$$f(x)=\frac{f(x_0)}{0!}+\frac{f'(x_0)}{1!}(x-x_0)+\frac{f''(x_0)}{2!}(x-x_0)^2+\cdots+\frac{f^n(x_0)}{n!}(x-x_0)^n+o(x-x_0)^n$$

令 $\phi(x)=[\phi_0(x-x_0),\phi_1(x-x_0),\phi_2(x-x_0),\cdots,\phi_n(x-x_0)]^T$

其中 $\phi_0(x)=1,\phi_n(x)=(x-x_0)^n$

令 $\theta_n=\frac{f^n(x_0)}{n!},\theta=(\theta_0,\theta_1,\cdots,\theta_n)^T$

即：

$$f(x)=\sum_{k=0}^{n}\theta_k\phi_k(x)+o(x-x_0)^n$$

$$f(x)=\hat{\theta}*\hat{\phi(x)}+o(x-x_0)^n$$

结论：对于区间 $[a,b]$ 上任意一点，函数值都可以用两个向量内积的表达式近似，其中 $\phi_k(x)$ 是基函数（basis function），θ_k 是相应的系数。高阶表达式 $o(x-x_0)^n$ 表示两者值的误差。

1.1.2　傅里叶级数

对于周期为 T 的函数，频率 $w_0 = \frac{2\pi}{T}$，函数 $f(x)$ 的表达式如下：

$$f(x) = c_0 + \sum_{i=1}^{\infty} c_n cos(nw_0 x)$$

当 $n \geq N$ 时，$f(t)$ 收敛，则：

$$f(x) = c_0 + \sum_{n=1}^{N} c_n cos(nw_0 x) + \epsilon, \epsilon \to 0$$

令 $\phi(x) = [\phi_0(x - x_0), \phi_1(x - x_0), \phi_2(x - x_0), \cdots, \phi_n(x - x_0)]^T$

其中 $\phi_0(x) = 1, \phi_n(x) = cos(nw_0 t)$

$\hat{c} = (c_0, c_1, c_2, \cdots, c_n)^T$

则：

$$f(x) = \sum_{n=0}^{N} c_n \phi_n(x) + \epsilon$$

$$f(x) = (\hat{c})^T * \hat{\phi(x)} + \epsilon$$

周期函数 $f(x)$ 可以用向量内积近似，$\phi_n(x)$ 表示基函数，c_n 表示相应的系数，ϵ 表示误差。

1.1.3　线性回归

由泰勒公式和傅里叶级数可知，当基函数的数量足够多时，向量内积无限接近于函数值。线性回归的向量内积表达式如下：

$$f(x) = \theta_0 + \theta_1 \phi_1(x) + \theta_2 \phi_2(x) + \cdots + \theta_n \phi_n(x) + \epsilon$$

$$f(x) = \sum_{j=0}^{n} \theta_j \phi_j + \epsilon$$

$$f(x) = \hat{\theta_j}^T \hat{\phi_j(x)} + \epsilon$$

其中，$\hat{\theta_j} = [\theta_0, \theta_1, \theta_2, \cdots, \theta_n]^T, \hat{\phi_j(x)} = [\phi_0(x), \phi_1(x), \phi_2(x), \cdots, \phi_n(x)]^T$

若令 $\phi_n(x) = x^n$，除了多了方差 ϵ 这一项，$f(x)$ 的表达式就是最常见的线性回归表达式。

1.2　过拟合问题

1.2.1　过拟合定义

构建模型的训练误差很小或为 0，测试误差很大，这一现象称为过拟合。

1.2.2　高斯噪声数据模型

我们采集的样本数据其实包含了噪声，假设该噪声的高斯噪声模型，均值为 0，方差为 σ^2。若样本数据的标记为 y_1，理论标记为 y，噪声为 η，则有：

$$y_1 = y + \eta$$

η 是高斯分布的抽样。

上节的线性回归表达式的方差 ϵ 表示的意义是噪声高斯分布的随机抽样，书本的线性回归表达式把方差 ϵ 也包含进去了。

1.2.3　过拟合原因

数学术语

当基函数的个数足够大时，线性回归表达式的方程恒相等。

如下所示：

$$f(x) \equiv \hat{\theta_j}^T \hat{\phi_j}(x) + \epsilon$$

对于任意样本数据，等式恒成立。

机器学习术语

模型太过复杂以致于把无关紧要的噪声也学进去了。

当线性回归的系数向量间差异比较大时，则大概率设计的模型处于过拟合了。用数学角度去考虑，若某个系数很大，对于相差很近的 x 值，结果会有较大的差异，这是较明显的过拟合现象。

过拟合的解决办法是降低复杂度。

1.3　模型的优化方向

模型的不同主要是体现在参数个数，参数大小以及正则化参数 λ，优化模型的方法是调节上面三个参数（但不仅限于此，如核函数），目的是找到最优模型。

1.4　总结

通过泰勒公式和傅里叶级数的例子说明线性回归的合理性，线性回归表达式包含了方差项，该方差是高斯噪声模型的随机采样，若训练数据在线性回归的表达式恒相等，那么就要考虑过拟合问题了，回归系数间差异比较大也是判断过拟合的一种方式。模型优化的方法有很多种，比较常见的方法是调节参数个数，参数大小以及正则化参数 λ。

第二章　线性回归的理解

线性回归算法包括最小二乘法和最大似然法。

2.1　最小二乘法和最大似然函数

2.1.1　最小二乘法

训练数据 D 共有 N 个观测数据，数据的输入 $\hat{x} = (x_1, x_2, \cdots, x_N)^T$，输出 $\hat{y} = (y_1, y_2, \cdots, y_N)^T$。假设模型有 M 个参数个数。

最小二乘法求数据集的线性回归模型步骤如下：

（1）线性回归模型表达式

$$y(x, \theta) = \theta_0 + \sum_{i=1}^{M-1} \theta_i \phi_i(x)$$

$$y(x, \theta) = \sum_{i=0}^{M-1} \theta_i \phi_i(x) = \hat{\theta}^T \hat{\phi(x)}$$

其中，$\hat{\theta} = (\theta_0, \theta_1, \cdots, \theta_{M-1})^T, \hat{\phi(x)} = (\phi_0(x), \phi_1(x), \cdots, \phi_{M-1}(x))^T$

（2）最小二乘法求最优参数 $\hat{\theta}$

误差平方和 $E_D(\theta)$

$$E_D(\hat{\theta}) = \sum_{n=1}^{N} (t_n - \hat{\theta}^T \hat{\phi(x)})^2$$

由 $\frac{\partial E_D(\hat{\theta})}{\partial \hat{\theta}} = 0$，可得最优化参数 $\hat{\theta}$。

2.1.2　最大似然函数

假设训练数据的目标变量 t 是由确定性方程 $y(x, \theta)$ 和高斯噪声叠加产生的，即：

$$t = y(x, \theta) + \epsilon$$

其中 ϵ 是期望为 0，精度为 β（方差的倒数））的高斯噪声的随机抽样。

目标变量 t 的分布推导如下：

$t = y(x, \theta) + \epsilon$

等式两边取期望，得到：

$E(t) = E(y(x, \theta)) + E(\epsilon)$

因为 $y(x, \theta)$ 是确定性方程，可以理解为常数，变量 ϵ 期望为 0，精度为 β。

所以 $E(t) = y(x, \theta)$

同理可得，$D(t) = \beta^{-1}$

因此，目标变量 t 的分布：

$$P(t|x, \theta, \beta) = N(t|y(x, \theta), \beta^{-1}$$

即观测数据集的似然函数：

$$P(t|\hat{x}, \hat{\theta}, \beta) = \Pi_{n=1}^{N} N(t|\hat{\theta}^T \phi(\hat{x_n}), \beta^{-1})$$

为了书写方便，求最大似然函数对应的参数 $\hat{\theta}$。

似然函数取对数并不影响结果：

$$lnP(t|\hat{\theta}, \hat{\beta}) = \sum_{n=1}^{N} lnN(t_n|\hat{\theta}^T \phi(\hat{x_n}), \beta^{-1})$$

$$= \frac{N}{2} ln\beta - \frac{N}{2} ln(2\pi) - \beta E_D(\hat{\theta})$$

由上式可知:最大似然函数 $lnP(t|\hat{\theta}, \hat{\beta})$ 等价于最小化方差平方和 $E_D(\hat{\theta}) = \frac{1}{2} \sum_{n=1}^{N} (t_n - \hat{\theta}^T \phi(\hat{x}))^2$

似然函数对变量 $\hat{\theta}$ 求梯度，并令其等于 0，得出最优参数。（与正规方程的推导一致。）

因此，对于输入变量 x，即可求得输出变量 t 的期望。

$$E(t) = \hat{\theta}^T \phi(\hat{x})$$

期望值就是模型的预测输出变量，与最小二乘法的预测结果相同。

2.2　算法若干细节的分析

2.2.1　偏置系数

线性回归表达式的偏置参数w0有什么意义，我们最小化$E_D(w)$来求解w0，根据w0结果来说明其意义。

$$E_D(w)=\frac{1}{2}\sum_{n=1}^{N}\{t_n-\sum_{j=0}^{M-1}w_j\phi_j(x_n)\}^2$$

$$E_D(w)=\frac{1}{2}\sum_{n=1}^{N}\{t_n-w_0-\sum_{j=1}^{M-1}w_j\phi_j(x_n)\}^2$$

$$\frac{\partial E_D(w)}{\partial w_0}=\sum_{n=1}^{N}\{t_n-w_0-\sum_{j=1}^{M-1}w_j\phi_j(x_n)\}$$

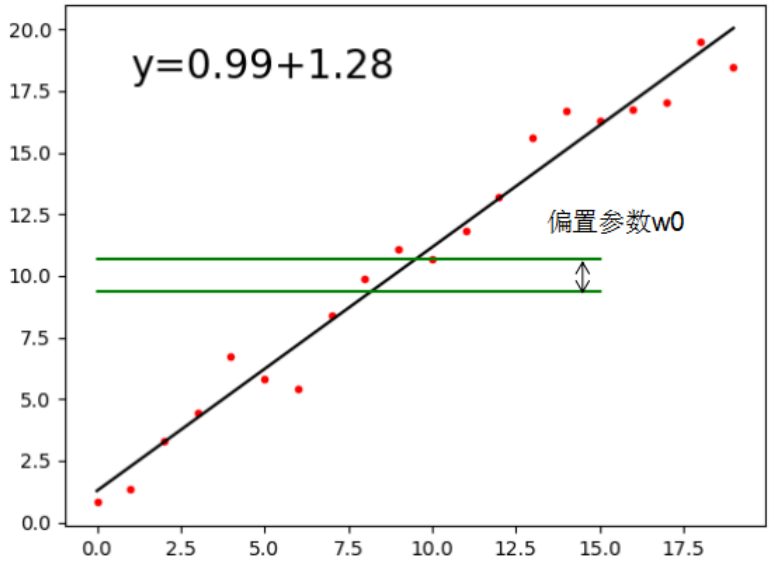
$$令\frac{\partial E_D(w)}{\partial w_0}=0$$

$$w_0=\bar{t}-\sum_{j=1}^{M-1}w_j\overline{\phi_j}$$

$$其中,\bar{t}=\frac{1}{N}\sum_{n=1}^{N}t_n,\overline{\phi_j}=\frac{1}{N}\sum_{n=1}^{N}\phi_j(x_n)$$

由 w_0 的结果可知，偏置参数补偿了目标值的平均值（在训练集）与基函数的值的加权求和之间的差。

图形表示为：

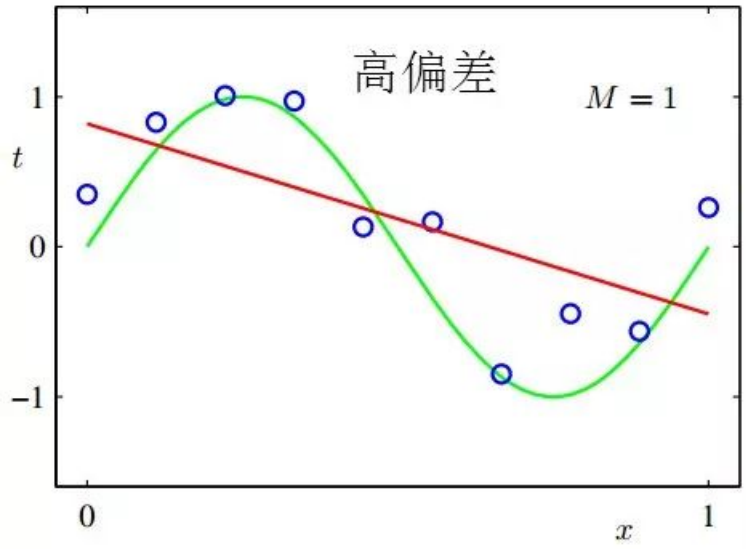


2.3　偏差和方差

2.3.1　高偏差

若模型参数个数比较少，即模型复杂度很低，模型处于高偏差状态。

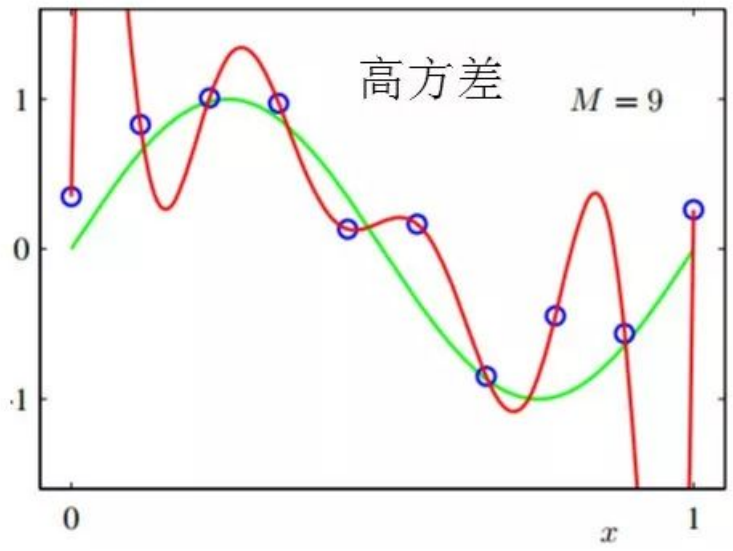
如下图用直线去拟合正弦曲线。



2.3.2　高方差

若模型参数个数较大，即复杂度较高，则模型处于高方差（过拟合）状态。

如下图 M=9 拟合正弦曲线，模型训练误差为 0。



第三章　正则项的详细分析

当模型的复杂度达到一定程度时，则模型处于过拟合状态，类似这种说法我们已经见到了很多次了，接下来我们首先讨论了怎么去理解复杂度这一概念，然后回顾贝叶斯思想，并从贝叶斯的角度去理解正则项的含义以及正则项降低模型复杂度的方法。

3.1　怎么去理解复杂度

怎么去理解复杂度，可能有人认为模型的参数越多，模型越复杂。笔者认为最好是通过结果去理解复杂度，比如当模型训练误差很小且测试误差很大时，则模型的复杂度较高，降低复杂度的方法包括减少模型参数的个数和降低模型参数值的大小等。

3.1.1　方差理解复杂度

当模型的复杂度较高时，模型对训练数据集非常敏感，符合同一分布的不同训练数据集构建的模型相差很大，即方差越高，模型的复杂度越大。

方差定义：

$$\text{variance} = \int \mathbb{E}_{\mathcal{D}}\left[\{y(\mathbf{x};\mathcal{D}) - \mathbb{E}_{\mathcal{D}}[y(\mathbf{x};\mathcal{D})]\}^2\right] p(\mathbf{x})\,\mathrm{d}\mathbf{x}$$

其中，x 表示抽样的测试数据，D 为抽样的训练数据集，y(x;D) 表示输入变量 x 在特定训练数据集 D 构建的模型的输出，不同的训练数据集 D 有不同的输出变量。

用样本的统计量来表示方差，如下图：

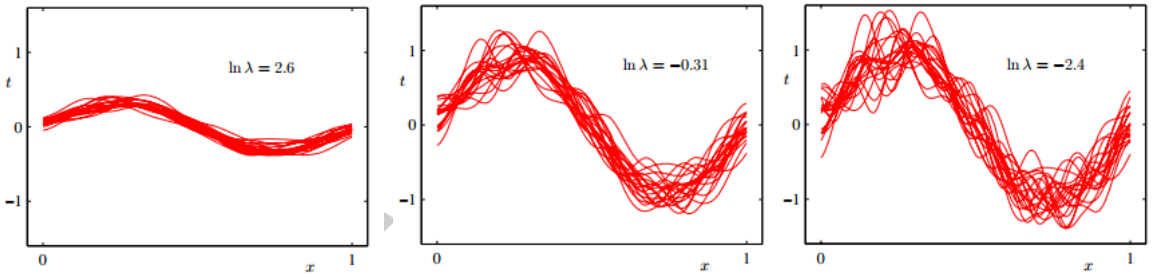
$$\text{variance} = \frac{1}{N}\sum_{n=1}^{N}\frac{1}{L}\sum_{l=1}^{L}\left\{y^{(l)}(x_n) - \overline{y}(x_n)\right\}^2$$

其中：

$$\overline{y}(x) = \frac{1}{L}\sum_{l=1}^{L} y^{(l)}(x)$$

上式含义：N 个测试样本在 L 个模型的输出方差

下面三张图表示了复杂度与方差之间的关系：

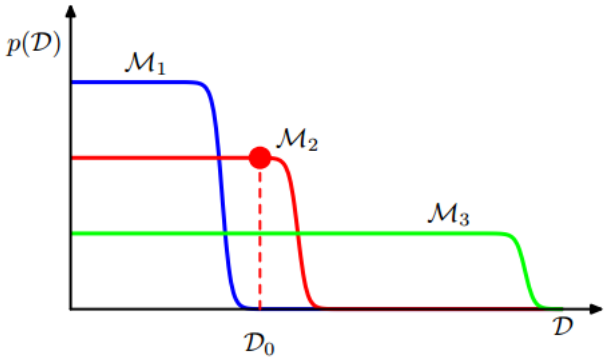


由上面三张图可知，第三张图的振动最剧烈，即方差最大，根据方差定义来理解复杂度，那么相应的复杂度也越高。

3.1.2　数据集分布理解复杂度

若模型越复杂，那么从该模型抽样的数据集变化越大，数据集覆盖的范围也越广。

如下图数据集 D 在模型 M1，模型 M2 和模型 M3 的分布情况：



由于数据集 D 在模型 M3 分布的范围最广，则模型 M3 的复杂度越高，M2 次之，M1 最低。

3.2　回顾贝叶斯思想

贝叶斯思想是根据当前的观测数据再加上自己的先验知识主观判断事件发生的概率。因此随着观测数据的增加，事件发生的概率会相应的发生改变，同时先验知识是影

响主观判断事件发生概率的另一个重要因素。

贝叶斯评估模型参数 w 分布的公式：

$$P(\mathrm{w}\mid D)=\frac{P(w)P(D\mid w)}{P(D)}$$

$$P(\mathrm{w}\mid D)=\frac{P(D\mid w)P(w)}{\int P(D\mid w)P(w)dw}$$

$\because P(D)$是标准化项

$\therefore P(\mathrm{w}\mid D)\infty P(D\mid w)P(w)$

$P(w)$是参数w的先验分布,$P(D\mid w)$是数据集D的似然函数

由该式可知，后验分布$P(\mathrm{w}\mid D)$由先验分布和似然函数决定

3.3　贝叶斯角度下的正则项

若模型的复杂度较高，那么通过在损失函数项增加正则项的方式来降低模型的复杂度。

如下图：

$$\tilde{E}(w)=\frac{1}{2}\sum_{n=1}^{N}(t_n-\vec{w}^{T}\overrightarrow{\phi(x_n)})^2+\frac{\lambda}{2}\sum_{j=1}^{M}|w_j|^q \qquad (3.1)$$

（1）若 q=1 时，则正则化项为 L1 范数，构建的线性回归称 LASSO 回归。

（2）若 q=2 时，则正则化项为 L2 范数，构建的线性回归称 Ridge 回归。

最小化损失函数 $E(\hat{w})$ 得到的参数 w 即是模型的最优解。

3.3.1　贝叶斯角度分析损失函数

先验分布是高斯分布

由上节可知，贝叶斯估计模型参数 w 的分布需要知道参数的先验分布和数据集的似然函数，若数据集 D 已知，参数 w 的先验分布是均值为 0 精度为　的高斯分布。

则参数 w 的后验分布的推导过程如下：

数据集D包含N个高斯分布的样本数据，精度为β

输入数据为$\mathbf{x}=(x_1,x_2,\cdots x_N)^T$，输出数据为$\mathbf{t}=(t_1,t_2,\cdots t_N)^T$

假设参数$\mathbf{w}$的先验分布是均值为0，精度为α的高斯分布。

后验概率最大时对应的参数是最优模型参数$\vec{w}^*$

即$\vec{w}^* = \arg\max_w P(w|D) = \arg\max_w \frac{P(D|w)*P(w)}{P(D)}$

$= \arg\max_w (P(D|w)*P(w))$

取对数

$\vec{w}^* = \arg\max_w (\ln(P(D|w)) + \ln P(w))$

由$P(D|w)=\prod_{n=1}^{N} P(t_n|\vec{w}^T\vec{\phi(x_n)},\beta^{-1})$，$P(w)=N(w|0,\alpha^{-1})$

得：$\vec{w}^* = \arg\min_w (-\frac{\beta}{2}\sum_{n=1}^{N}(t_n-\vec{w}^T\vec{\phi(x_n)})^2 - \frac{\alpha}{2}\vec{w}^T\vec{w}+const)$

即后验概率分布$P(\mathrm{w}|D) = -\frac{\beta}{2}\sum_{n=1}^{N}(t_n-\vec{w}^T\vec{\phi(x_n)})^2 - \frac{\alpha}{2}\vec{w}^T\vec{w}+const$

等价于：

$$P(\mathrm{w}|D) = -\frac{1}{2}\sum_{n=1}^{N}(t_n-\vec{w}^T\vec{\phi(x_n)})^2 - \frac{\alpha}{2\beta}\vec{w}^T\vec{w}+const$$

令$\frac{\alpha}{\beta}=\lambda$

$$P(\mathrm{w}|D) = -\frac{1}{2}\sum_{n=1}^{N}(t_n-\vec{w}^T\vec{\phi(x_n)})^2 - \frac{\lambda}{2}\vec{w}^T\vec{w}+const \quad (3.2)$$

由 3.2 可知，后验概率最大化等于包含 L2 正则化项损失函数的最小化。

式（3.1）第一项表示损失函数，第二项表示惩罚函数。

式（3.2）第一项表示数据 D 的似然函数，第二项表示参数的先验分布。

比较两式可知，参数的先验分布对应于正则化项。当参数的先验分布为高斯分布时，则正则化项为 L2 范数，构建的回归模型称为 Ridge 回归。

先验分布是拉普拉斯分布

当参数的先验分布为拉普拉斯分布，则正则化项为 L1 范数，构建的回归模型称为 LASSO 回归。

小结

贝叶斯定理的后验分布与似然函数和先验分布相关，不考虑先验分布时，则损失函数不包含正则化；考虑先验分布时，则损失函数包含正则化；最大化后验分布等同于最小化正则化的损失函数。

3.4　正则项降低模型复杂度的方法

降低模型复杂度的方法主要包括减少模型参数的个数和降低模型参数的值。本节介绍正则项降低模型复杂度的方法。

$$\widetilde{E}(w)=\frac{1}{2}\sum_{n=1}^{N}(t_n-\overrightarrow{w}^T\overrightarrow{\phi(x_n)})^2+\frac{\lambda}{2}\sum_{j=1}^{M}|w_j|^q$$

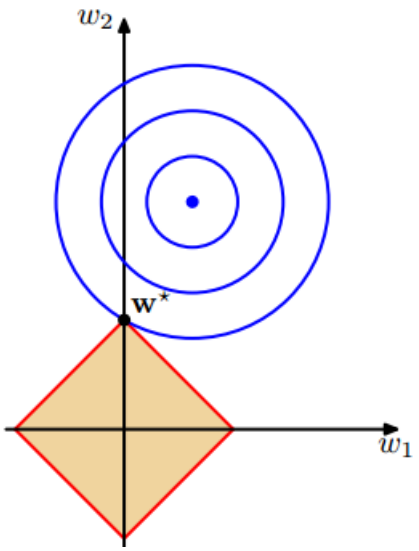
最小化 $E(\hat{w})$ 等价于:

$$(E(w))_{\min}=\left\{\frac{1}{2}\sum_{n=1}^{N}(t_n-\overrightarrow{w}^T\overrightarrow{\phi(x_n)})^2\right\}\min \quad (1)$$

$$\sum_{j=1}^{M}|w_j|^q\le\eta \quad (2)$$

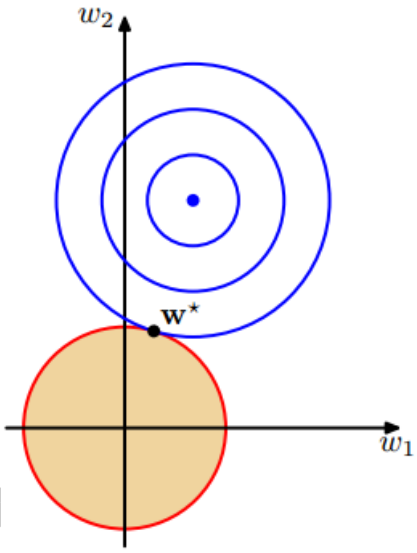
即在式（2）的条件下，求（1）的最小值，L1 和 L2 正则项都是利用这种思想来求最优参数。

3.4.1　正则项是 L1 范数



如上图含 L1 正则项的损失函数，蓝色线为损失函数，红色线为 L1 正则项包含的区域。当参数处于交点图片时，含正则项的损失函数最小。由图可知该交点的 w2 为 0，则模型参数个数较少了，相应的模型复杂度降低了。

3.4.2　正则项是 L2 范数



分析方法与 L1 类似，该交点所处的坐标为 w1 较小，即改变了模型参数值的大小，复杂度也相应的降低。

第四章　线性判别模型分析

4.1　相关数学知识回顾

4.1.1　直线方程和平面方程

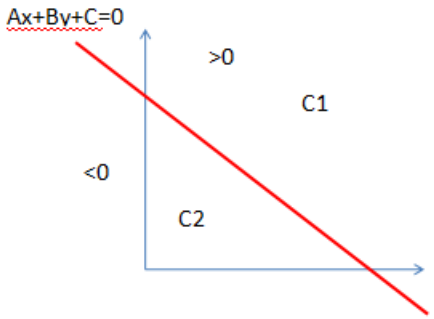
直线方程$l: Ax+By+C=0$，点P的坐标(x_0, y_0)

(1)A落在直线l上方时，$Ax_0+By_0+C>0$

(2)A落在直线l下方时，$Ax_0+By_0+C<0$

(3)A落在直线l时，　　$Ax_0+By_0+C=0$

拓展到分类思想：直线 l 为分类决策方程，坐标点落在直线 l 上方时，则分类为 C1；坐标点落在直线 l 下方时，则分类为 C2（如下图）。



4.1.2　点到直线和点到平面的距离

点到直线的距离：

直线l方程：$Ax+By+b=0$，若某一点A的坐标为(x_0,y_0)

求直线l法向量和点A到直线l的距离d;

解：直线l的方向向量$\vec{c}=(-B,A)$,法向量与方向向量垂直相交

$\therefore$法向量$\vec{m}=(A,-B)$

点A到直线l的距离$\mathrm{d}=\left|\frac{Ax_0+By_0+b}{\sqrt{A^2+B^2}}\right|$

点到平面的距离:

平面方程$Ax+By+Cz+D=0$，点P的坐标(x_0,y_0,z_0)

求平面的法向量和点P到平面的距离d

如下图：

法向量$\vec{\mathrm{n}}=(A,B,C)^T$

$\mathrm{d}=\left|\frac{Ax_0+By_0+Cz_0+D}{\sqrt{A^2+B^2+C^2}}\right|$

等价于下面两向量的内积：

$\vec{\mathrm{w}}=(D,x_0,y_0,z_0)^T$，法向量$\vec{\mathrm{n}}=(1,A,B,C)^T$

$d=\frac{\vec{\mathrm{w}}^T*\vec{\mathrm{n}}}{\sqrt{A^2+B^2+C^2}}$

$\because\sqrt{A^2+B^2+C^2}$为常数

$\therefore \mathrm{d}\propto\vec{\mathrm{w}}^T*\vec{\mathrm{n}}$

因此点到直线或点到面的距离可以用向量内积表示

拓展到分类思想：平面方程为决策方程，正确分类的情况下，当点 P 到决策方程的距离越大，则分类模型越好；错误分类的情况下，点 P 到决策方程的距离作为损失函数，损失函数最小化过程即是模型参数最优化过程。

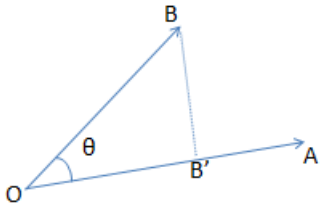
4.1.3　向量内积的数学意义

向量A与向量B的内积定义为：

$\mathrm{A}*\mathrm{B}=|\mathrm{A}\|\mathrm{B}|\cos\theta$，$\theta$是向量$A$和向量$B$的夹角

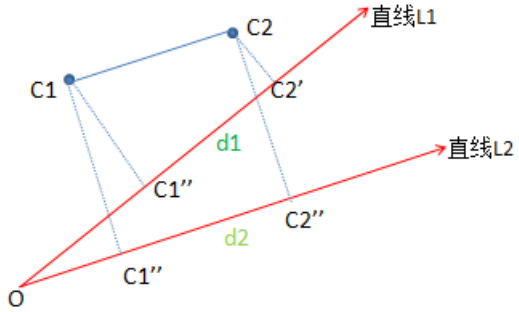
几何意义：向量 A 与向量 B 的内积等于向量 A 在向量 B 的投影与向量 B 的乘积，当向量 B 是单位向量时，则等于向量 A 在单位向量方向的投影，单位向量类似于基函

数或者可以理解成坐标轴, 即向量 A 在向量 B 的投影可理解成向量 A 在向量 B 方向的坐标，如下图，B’ 是 B 在 OA 坐标轴方向的投影。



拓展到分类思想：C1 与 C2 属于不同的类，给定一条决策性直线 l，当 C1 与 C2 在直线 L2 的投影间距越大，则分类效果越好。增加不同类间的距离可以作为模型参数优化的方向。

如下图，C1 和 C2 的在直线 L2 的投影距离 |C1”C2”| 大于 |C1’C2’|，因此决策方程直线 L2 优于直线 L1



4.1.4　梯度下降法

梯度的定义如下：

$$gradf(x_0, x_1, \ldots, x_n) = \left(\frac{\partial f}{\partial x_0}, \ldots, \frac{\partial f}{\partial x_j}, \ldots, \frac{\partial f}{\partial x_n}\right)$$

函数 f(x0,x1,...,xn) 在梯度方向是函数值变化（增加或减少）最快的方向（本文只给出结论，后续文章会有详细的说明）。

拓展到分类思想：损失函数最小化过程即是模型参数最优化过程，损失函数最小化可通过梯度下降法来实现，当迭代到一定程度，损失函数收敛，则迭代结束，参数 w 即是要求的最优参数。

流程图如下：

假设损失函数$E(w)$，设置初始化参数$w_1 = w_0$和步长α

求模型的最优参数w

(1) 计算损失函数的梯度，梯度表达式如下：

$$\frac{\partial E(w)}{\partial w}$$

(2)更新参数w，

$$w'_1 = w_1 - \frac{\partial E(w)}{\partial w} * \alpha$$

(3) 重新计算损失函数$E'(w)$,

(4)若$E(w) - E'(w)$小于阈值，则w参数是最优参数；反之进入步骤(1)

4.2　判别式模型和生成性模型

我们常把分类问题分成两个阶段：推断阶段和决策阶段，对于输入变量 x，分类标记为 Ck。推断阶段和决策阶段具体表示为：

推断阶段：估计 P(x,Ck) 的联合概率分布，对 P(x,Ck) 归一化，求得后验概率 P(Ck|x)。

决策阶段：对于新输入的 x，可根据后验概率 P(Ck|x) 得到分类结果。

判别式模型和生成性模型的区别:

判别式模型：简单的学习一个函数，将输入 x 直接映射为决策，称该函数为判别式函数。

生成式模型：推断阶段确定后验概率分布，决策阶段输出分类结果，生成式模型包含两个阶段。

4.3　最小平方法

最小平方法与最小二乘法的算法思想类似，K 类判别函数由 K 个方程决定，训练集 x_n, t_n，K 类判别函数为 $y_k(x)$。

令损失函数$E(\vec{W})$

$$E(\vec{W})=\frac{1}{2}Tr\{(\vec{X}\vec{W}-\vec{T})^T(\vec{X}\vec{W}-\vec{T})\}$$

Tr表示求矩阵的迹（这里就不给出详细解释了，若有问题请微我）

最小化误差平方和，即$\frac{\partial E(\vec{W})}{\partial \vec{W}}=0$

即可求得最优参数矩阵$\vec{W}$,$\vec{W}$的每一列表示决策函数的最优参数$\vec{w}$

求得最优参数 w 后，输入变量 x 所属 K 类的判别方法如下：

K类判别函数由K个决策方程决定，假设输入$\vec{x_0}=(x_1,x_2,x_3)^T$

决策方程：$y_k(\vec{x})=\vec{w_k}^T\vec{x}+w_0$

分类思想：计算点到K个决策方程的距离，选取距离最大对应的决策方程即是输入向量所属类别。

分类算法：由点到平面的距离（可看第一节）可知：

$y_k(\vec{x_0})=\vec{w_k}^T\vec{x}+w_0$就是输入到输出的距离

选取最大距离对应的决策方程即是分类结果，$k=(y_k(\vec{x_0}))_{max}$

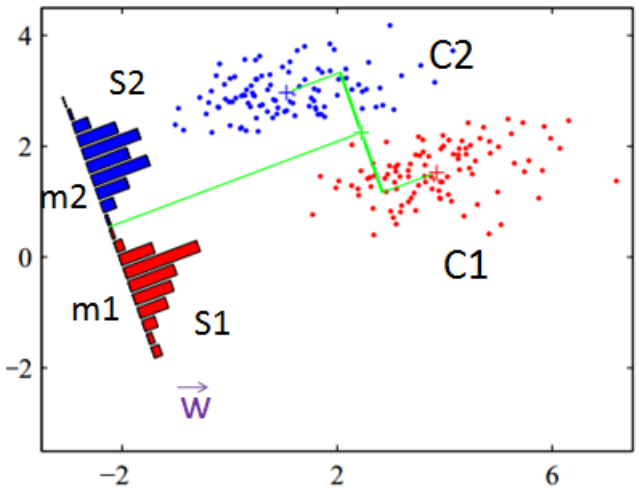
4.4　Fisher 线性判别函数

第一节讲到，若两个类在同一个决策方程的投影距离相隔越大，则该决策方程越好。再深入一点，相同类投影到决策方程的方差越小，则该决策方程越好，方差代表类投影到决策方程的聚集程度。这就是 Fisher 线性判别法参数优化思想。

参数优化思想：同类样本投影到决策方程的方差最小，不同类样本投影到决策方程的均值间隔最大。用表达式 J(w) 表示，J（w）越大越好。

$$J(\boldsymbol{w})=\frac{(m_2-m_1)^2}{s_1^2+s_2^2}$$

如下图：



其中，m1，m2 分别表示不同类在决策方程的投影均值；S1,S2 分别表示不同类投影到决策方程的方差。

$$令\frac{\partial J(w)}{\partial w}=0，即可求得最优参数w$$

求得最优参数 w 后，输入变量 x 所属类的判别方法如下：

$$设置一阈值y_0，若\vec{w}^T\vec{x}\geq y_0,则分类结果为C2，反之为C1。$$

4.5　感知器算法

感知器算法的目的是找到能够准确分离正负样本训练数据集的超平面。

超平面定义：

$$\vec{w}^T * \vec{x} + b = 0$$

其中，参数$\vec{w}$和b是超平面参数

感知器学习策略：

对训练数据集某一样本点 (x,y)，若 wx+b>0，则 y=1；若 wx+b<0，则 y=-1；

即感知机模型为：

$$y = sign(wx + b)$$

因此，对于误分类的数据 (xi,yi) 来说：

$$-y_i(w \cdot x_i + b) > 0$$

成立．因为当 $w \cdot x_i + b > 0$ 时，$y_i = -1$，而当 $w \cdot x_i + b < 0$ 时，$y_i = +1$．因此，误分类点 x_i 到超平面 S 的距离是

$$-\frac{1}{\|w\|} y_i(w \cdot x_i + b)$$

因此，感知器学习策略是最小化误分类点到平面 S 的距离，不考虑分母项。

假设训练数据集有 M 个误分类点，损失函数为：

$$L(w,b) = -\sum_{x_i \in M} y_i(w * x_i + b)$$

运用梯度下降算法最小化损失函数 L(w,b)。

$$\frac{\partial L(w,b)}{\partial w} = -y_i x_i$$

$$\frac{\partial L(w,b)}{\partial b} = -y_i$$

设学习率 η，感知器学习策略步骤：

（1）选取处置 w0，b0；

（2）选取训练集 (xi,yi)；

（3）如果 $yi(w * xi + b) <= 0$，则更新权值参数 w,b：

$$w \leftarrow w + \eta y_i x_i$$

$$b \leftarrow b + \eta y_i$$

4）转至（2），直至训练数据集没有误分类点，得到超平面最优参数 w，b。

感知机学习算法由于采用不同的初值或选取不同的误分类点，参数解可能不同。

因此，对某一输入点，若感知机模型大于 0，则分类为 1；反之分类为-1。

第五章　logistic 回归模型分析

5.1　logistic 回归模型的含义

我们把分类模型分成两个阶段，推断阶段和决策阶段，推断阶段对联合概率分布建模，然后归一化，得到后验概率。决策阶段确定每个新输入 x 的类别。

我们用推断阶段的方法来推导 logistic 回归模型，首先对类条件概率密度 $P(\hat{x}|C_k)$ 和类先验概率分布 $P(C_k)$ 建模，然后通过贝叶斯定理计算后验概率密度。

考虑二分类的情形，类别C1的后验概率密度：

$$P(C1|\vec{x}) = \frac{P(\vec{x}|C1)P(C1)}{P(\vec{x})}$$

$$P(C1|\vec{x}) = \frac{P(\vec{x}|C1)P(C1)}{P(\vec{x}|C1)P(C1)+P(\vec{x}|C2)P(C2)}$$

$$P(C1|\vec{x}) = \frac{1}{1+\frac{P(\vec{x}|C2)P(C2)}{P(\vec{x}|C1)P(C1)}}$$

令 $\ln\frac{P(\vec{x}|C1)P(C1)}{P(\vec{x}|C2)P(C2)} = a$

则：$P(C1|\vec{x}) = \frac{1}{1+e^{-a}} = \sigma(a)$

式中的$\sigma(a)$就是log *istic*函数

因此，log *istic*回归的值等于输入变量为x的条件下类别为C1的概率($P(C1|\vec{x})$

$$\sigma(a)=\frac{1}{1+e^{-a}}$$

$$a=\ln\frac{P(\vec{x}|C1)P(C1)}{P(\vec{x}|C2)P(C2)}$$

$$a=\ln\frac{P(\vec{x},C1)}{P(\vec{x},C2)}$$

(1)当 $a\geq 0$ 时，$P(\vec{x},C1)\geq P(\vec{x},C2)$，$P(C1|\vec{x})\geq\frac{1}{2}$，分类结果为 $C1$

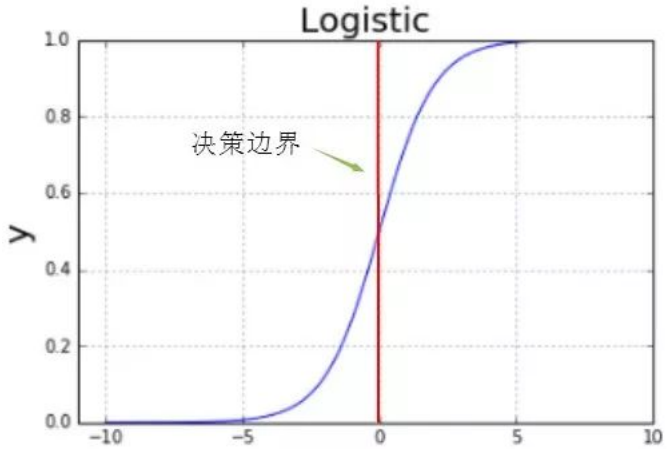
(2)当 $a<0$ 时，$P(\vec{x},C1)<P(\vec{x},C2)$，$P(C1|\vec{x})<\frac{1}{2}$，分类结果为 $C2$

结论：logistic 回归值表示所属类的后验概率，无论是二分类还是多分类，分类结果都是后验概率最大所对应的类。

5.2 logistic 的决策边界函数分析

决策边界函数，简而言之，就是函数的两侧是不同的分类结果。

logistic 曲线如下图，红色直线（a=0）表示决策边界函数：



假设类条件概率密度是高斯分布，即 P(x|Ck)，然后求解后验概率的表达式，即 P(Ck|x)。由第一节可知 logistic 回归值就是所求的后验概率。

假设类条件概率密度的协方差相同，类条件概率密度为：

$$p(\boldsymbol{x}\mid\mathcal{C}_k)=\frac{1}{(2\pi)^{\frac{D}{2}}}\frac{1}{|\boldsymbol{\Sigma}|^{\frac{1}{2}}}\exp\left\{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_k)^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_k)\right\}$$

由第一节的推导公式可得后验概率为：

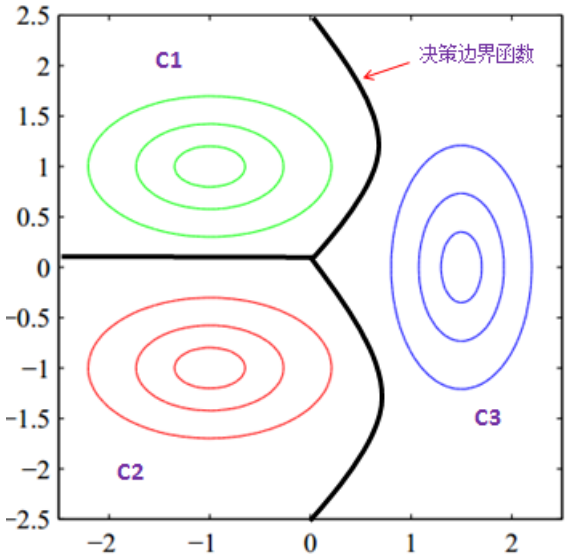
$$P(C_k \mid x) = \sigma(w_k^T x + w_{k0})$$

其中：

$$\boldsymbol{w}_k = \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_k$$
$$w_{k0} = -\frac{1}{2}\boldsymbol{\mu}_k^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_k + \ln p(\mathcal{C}_k)$$

由后验概率 $P(C_k|x)$ 的表达式可知可知，当类条件的协方差矩阵相等时，决策边界函数是随 x 线性变化的直线。

结论：如下图，若两类的条件概率密度的协方差相同时（如 C1 和 C2 的协方差相同），则决策边界函数是直线；若两类的条件概率密度的协方差不相同时（如 C1 和 C3，C2 和 C3），则决策边界函数是曲线。判断协方差矩阵是否相同可以根据分布图形形状是否相同来判断，如 C1 和 C2 的协方差相同，C3 和 C1、C2 的协方差不相同，协方差如何影响多元变量分布可参考上一小节。



假设类条件概率密度符合高斯分布且具有相同的协方差矩阵，则决策边界函数是一条直线；若类条件概率密度符合更一般的指数分布且缩放参数 s 相同，决策边界函数仍是一条直线。

5.3　logistic 模型的参数最优化

5.3.1　logistic 模型损失函数

logistic 回归模型的含义是后验概率分布，因此可以从概率的角度去设计损失函数。

考虑两分类情况，假设有N个训练样本，logistic模型是$h_\theta(x)$

$h_\theta(x)$表示后验概率$y=1$的概率，则$1-h_\theta(x)$表示$y=0$的概率

变量y_i取值1或0，且分别代表模型$h_\theta(x)$和$1-h_\theta(x)$

因此，似然函数$L(\theta)$:

$$L(\theta)=\prod_{i=1}^{N}(h_\theta(x)^{y_i})(1-h_\theta(x)^{y_i})$$

损失函数$J(\theta)$:

$$J(\theta)=-L(\theta)$$

$$J(\theta)=-\prod_{i=1}^{N}(h_\theta(x)^{y_i})(1-h_\theta(x)^{y_i})$$

5.3.2 logistic 模型的参数最优化

损失函数最小化等价于模型参数的最优化，如下图：

$$J(\theta)=-\prod_{i=1}^{N}(h_\theta(x)^{y_i})(1-h_\theta(x)^{1-y_i})$$

$$(J(\theta))_{\min}=\ln(J(\theta))\min$$

$$\ln(J(\theta))=-\prod_{i=1}^{N}(y_i\ln(h_\theta(x)+(1-y_i)\ln(1-h_\theta(x)))$$

利用梯度下降法求最优解，学习速率α：

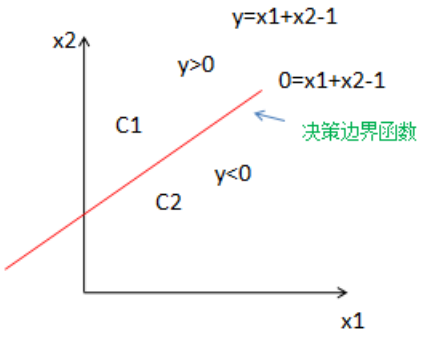
$$\theta=\theta-\alpha\frac{\partial J(\theta)}{\partial\theta}$$

为了避免过拟合问题，则在原来的损失函数增加正则项，然后利用梯度下降法求最优解

5.4 logistic 模型与感知机模型的比较

5.4.1 相同点

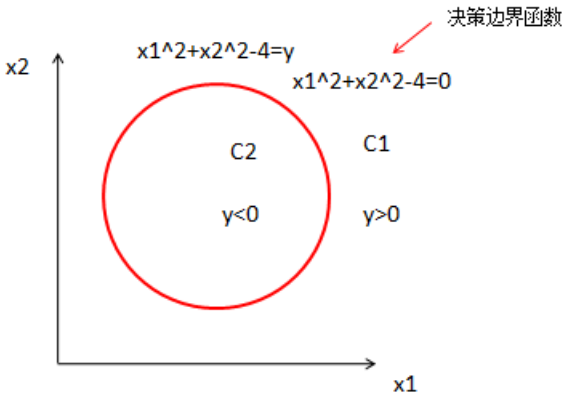
由第二节分析可知，假设类条件概率分布的协方差相同，则 logistic 模型的决策边界函数是随 x 线性变化的直线，因此，感知机模型与 logistic 模型的分类策略一样，即决策边界函数是一样的。如下图。



感知机模型：当点落在直线上方，y>0，则分类结果 C1；反之为 C2。

logistic 模型：当点落在直线上方，y>0，则后验概率 P(C1|X)>0.5，分类结果 C1；反之为 C2。

考虑到对输入变量 x 进行非线性变换 $\theta(x)$，感知机和 logistic 模型的分类策略仍一样，决策边界函数相同，如下图：

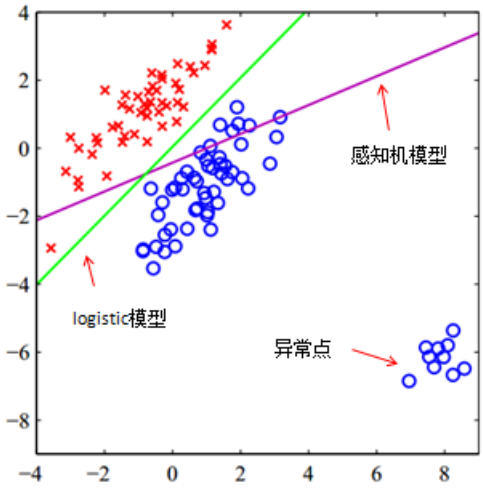


感知机模型：当点落在圆外，y>0，则分类结果 C1；反之为 C2。

logistic 模型：当点落在圆外，y>0，则后验概率 P(C1|X)>0.5，分类结果 C1；反之为 C2。

5.4.2　异常处

（1）logistic 回归模型限制值的范围在 0 1，感知机模型对值范围没有限制，因此 logistic 模型相比感知机模型，对异常点有更强的鲁棒性。如下图，当有异常数据时，logistic 模型要好于感知机模型。



（2）感知机模型用误分类点到超平面的距离衡量损失函数，而 logistic 模型则从概率角度去衡量损失函数。

综上所述，logistic 模型相比于感知机模型对异常数据具有更好的鲁棒性。

第六章　L1 和 L2 正则化的解释

L1 是通过稀疏参数（减少参数的数量）来降低复杂度，L2 是通过减小参数值的大小来降低复杂度。

6.1　优化角度分析

损失函数：

$$L(\mathrm{w}) = E_D(w) + \lambda E_w(w)$$

模型最优化等价于损失函数的最小化：

$$\min_w L(w) = \min_w (E_D(w) + \lambda E_w(w))$$

6.1.1　L2 正则化的优化角度分析

$L2$正则化模型的最优化问题等价于：

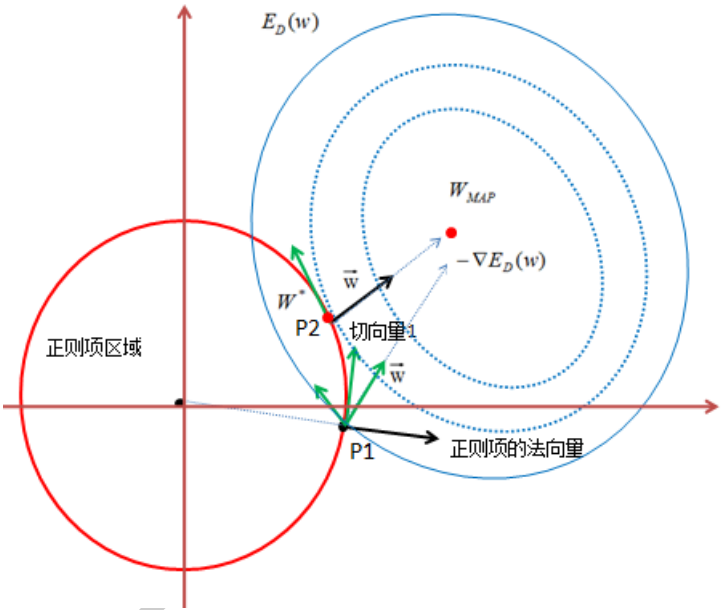
$$\min_w L(w) = \min_w (E_D(w) + \lambda \sum_{i=1}^{n} w_i^2)$$

等价为凸优化问题：

$$\begin{cases} \min\limits_w E_D(w) \\ \sum\limits_{i=1}^{n} w_i^2 \le C, \text{其中} C \text{与正则化参数} \lambda \text{成反比关系} \end{cases}$$

在限定的区域，找到使 $E_D(w)$ 最小的值。

图形表示为：



上图所示，红色实线是正则项区域的边界，蓝色实线是 $E_D(w)$ 的等高线，越靠里的等高圆，$E_D(w)$ 越小，梯度的反方向是 $E_D(w)$ 减小最大的方向，用 $\vec{w}$ 表示，正则项边界的法向量用实黑色箭头表示。

正则项边界在点P1的切向量有 $E_D(w)$ 负梯度方向的分量，所以该点会有往相邻的等高虚线圆运动的趋势；当P1点移动到P2点，正则项边界在点P2的切向量与 $E_D(w)$ 梯度方向的向量垂直，即该点没有往负梯度方向运动的趋势；所以P2点是 $E_D(w)$ 最小的点。

结论：L2 正则化项使 $E_D(w)$ 值最小时对应的参数变小。

6.1.2　L1 正则化的优化角度分析

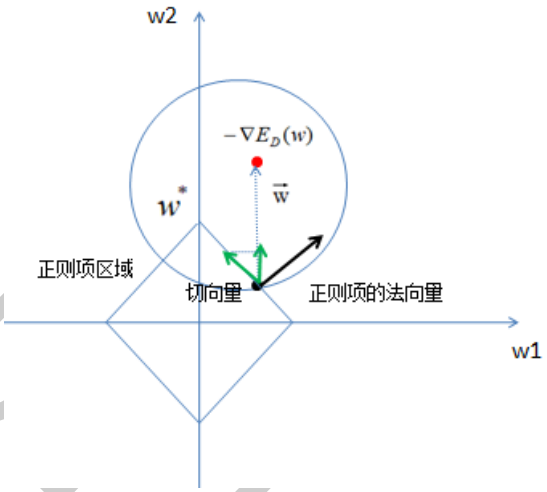
L1正则化模型的最优化问题等价于：

$$\min_w L(w) = \min_w (E_D(w) + \lambda \sum_{i=1}^{n} | w_i |)$$

等价为凸优化问题：

$$\begin{cases} \min\limits_{w} E_D(w) \\ \sum\limits_{i=1}^{n} |w_i| \le C, \text{其中} C \text{与正则化参数} \lambda \text{成反比关系} \end{cases}$$

在限定的区域，找到使 $E_D(w)$ 最小的值。



结论：如上图，因为切向量始终指向 w2 轴，所以 L1 正则化容易使参数为 0，即特征稀疏化。

6.2　梯度角度分析

6.2.1　L1 正则化

L1 正则化的损失函数为：

$$L(\mathrm{w}) = E_D(w) + \frac{\lambda}{n}\sum_{i=1}^{n}|w_i|$$

求$L(\mathrm{w})$的梯度

$$\frac{\partial L(\mathrm{w})}{\partial w} = \frac{\partial E_D(w)}{\partial w} + \frac{\lambda \operatorname{sgn}(w)}{n}$$

参数w更新：

$$w' = w - \eta\frac{\partial L(\mathrm{w})}{\partial w}$$

$$w' = w - \frac{\eta\lambda \operatorname{sgn}(w)}{n} - \frac{\partial E_D(w)}{\partial w}, \text{其中} \eta \text{为学习率}$$

上式可知，当 w 大于 0 时，更新的参数 w 变小；当 w 小于 0 时，更新的参数 w 变大；所以，L1 正则化容易使参数变为 0，即特征稀疏化。

6.2.2　L2 正则化

L2 正则化的损失函数为：

$$L(w) = E_D(w) + \frac{\lambda}{2n}\sum_{i=1}^{n} w_i^2$$

求$L(w)$的梯度：

$$\frac{\partial L(w)}{\partial w} = \frac{\partial E_D(w)}{\partial w} + \lambda w$$

参数w更新：

$$w' = w - \eta\frac{\partial L(w)}{\partial w}$$

$$w' = w - \frac{\eta\lambda w}{n} - \frac{\partial E_D(w)}{\partial w}$$

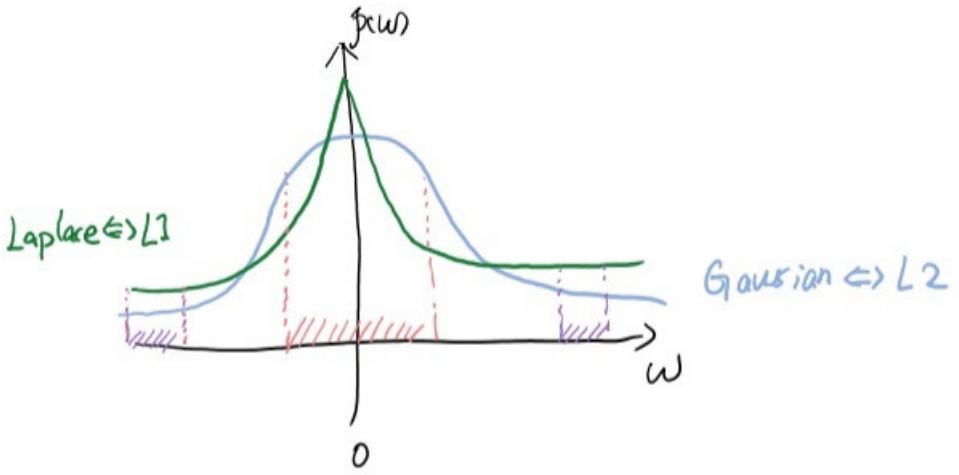
$$w' = (1 - \frac{\eta\lambda}{n})w - \frac{\partial E_D(w)}{\partial w}$$

由上式可知，正则化的更新参数相比于未含正则项的更新参数多了 $\frac{\eta\lambda}{n}w$ 项，当 w 趋向于 0 时，参数减小的非常缓慢，因此 L2 正则化使参数减小到很小的范围，但不为 0。

6.3　先验概率角度分析

当先验分布是拉普拉斯分布时，正则化项为 L1 范数；当先验分布是高斯分布时，正则化项为 L2 范数。本节通过先验分布来推断 L1 正则化和 L2 正则化的性质。

画高斯分布和拉普拉斯分布图（来自知乎某网友）：



由上图可知，拉普拉斯分布在参数 w=0 点的概率最高，因此 L1 正则化相比于 L2 正则化更容易使参数为 0；高斯分布在零附近的概率较大，因此 L2 正则化相比于 L1 正则化更容易使参数分布在一个很小的范围内。

6.4　限制条件

思想用到了凸函数的性质。

考虑标量 w，希望 $\min_w f(w)+\lambda|w|_1$

假设 $f(w)$ 在 $w=0$ 附近为凸，则对充分小的 Δ，我们有

$$f(\Delta)\geq f(0)+f'(0)\Delta$$

这时候 $w=0$ 为局部最小值的充分必要条件就是：对 $\forall\Delta$ 接近于0，

$$f(\Delta)+\lambda|\Delta|\geq f(0)$$

由于有凸性质，$w=0$ 的充分条件为：

$$f'(0)\Delta+\lambda|\Delta|\geq 0$$

分情况考虑 Δ 正负，得到，

$$|f'(0)|\leq\lambda$$

而注意如果是l2:

$$\min_w f(w)+\lambda||w||_2^2$$

$w=0$ 为局部最小值的充分必要条件为 $f'(0)=0$。

结论：含 L1 正则化的损失函数在 0 点取得极值的条件比相应的 L2 正则化要宽松的多，所以，L1 正则化更容易得到稀疏解（w=0）。

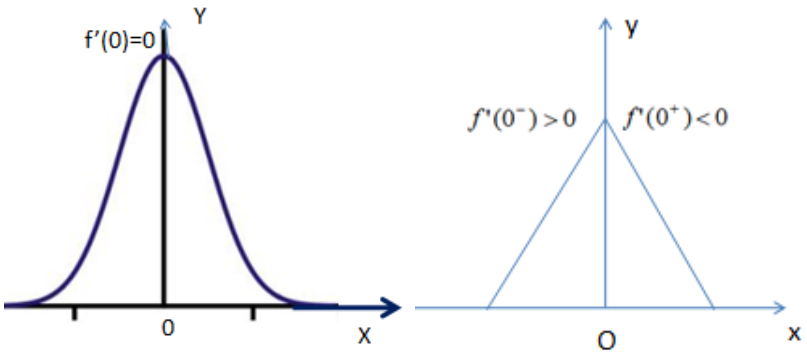
6.5　知乎点赞最多的图形角度分析

6.5.1　函数极值的判断定理

（1）当该点导数存在，且该导数等于零时，则该点为极值点；

（2）当该点导数不存在，左导数和右导数的符号相异时，则该点为极值点。

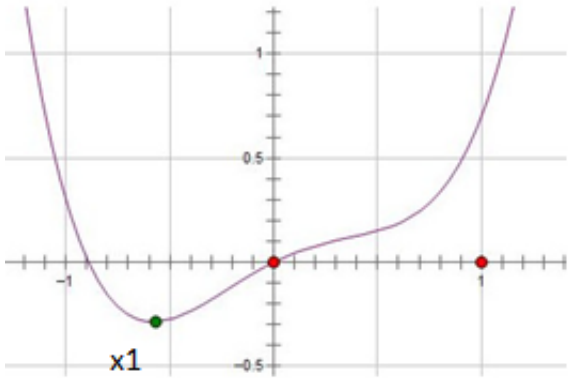
如下面两图：



左图对应第一种情况的极值，右图对应第二种情况的极值。本节的思想就是用了第二种极值的思想，只要证明参数 w 在 0 附近的左导数和右导数符合相异，等价于参数 w 在 0 取得了极值。

6.5.2　图形角度分析

损失函数 L 如下：



黑色点为极值点 x1，由极值定义：L'(x1)=0；

含 L2 正则化的损失函数: $f_2(x) = L + Cx^2(C > 0)$

对$f_2(x)$求导：

$f_2'(x) = L'(x) + 2Cx$

令$x = x1$

$f_2'(x1) = L'(x1) + 2Cx1$

$\because L'(x1) = 0$

$\therefore f_2'(x1) = 2Cx1$

$\because x1 < 0$

$\therefore f_2'(x1) < 0$

又$\because f_2'(0) = L'(0)$

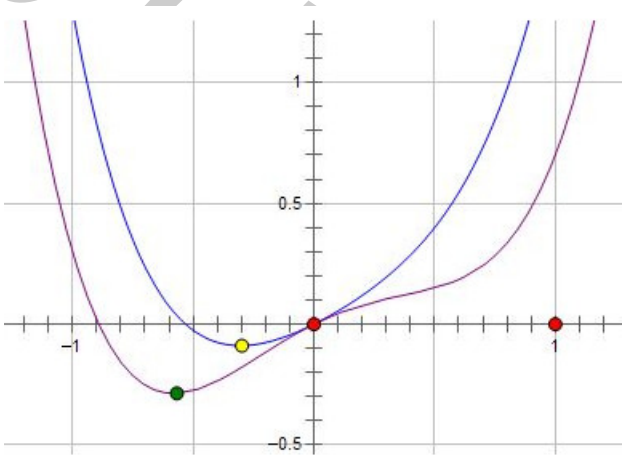
由上张图可知，$L'(0) > 0$

$\therefore f_2'(0) > 0$

即$f_2'(x)$在x1和0是异号，

$\therefore f_2(x)$在(x1,0)取极值

由结论可定性的画含 L2 正则化的图：



极值点为黄色点，即正则化 L2 模型的参数变小了。

含 L1 正则化的损失函数:$f_1(x) = L + C|x|, C > 0$

对$f_1(x)$求导

$$f_1'(x) = L'(x) + C * \mathrm{sgn}(x)$$

当$x \to 0^+$时

$$f_1'(0^+) = L'(x) + C$$

当$x \to 0^-$时

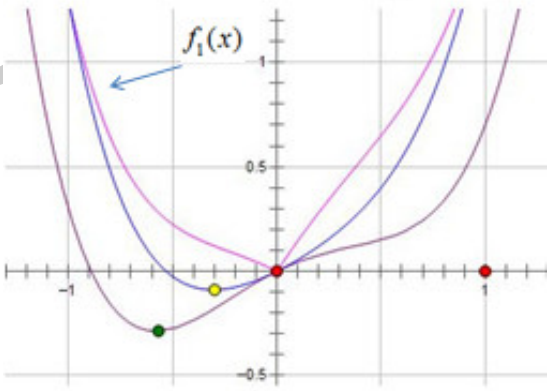
$$f_1'(0^-) = L'(x) - C$$

由第二条定理可知，

$$f_1'(0^+) * f_1'(0^-) < 0$$

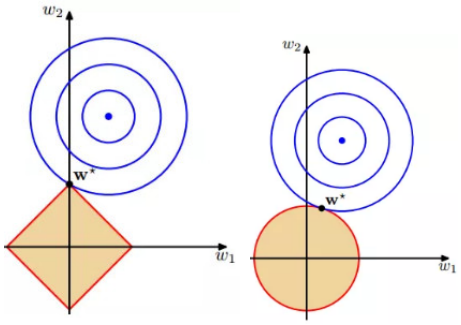
得：$C > |L'(x)|$

因此，只要 C 满足推论的条件，则损失函数在 0 点取极值 (粉红色曲线)，即 L1 正则化模型参数个数减少了。



6.6　PRML 的图形角度分析

因为 L1 正则化在零点附近具有很明显的棱角，L2 正则化则在零附近比较平缓。所以 L1 正则化更容易使参数为零，L2 正则化则减小参数值，如下图。



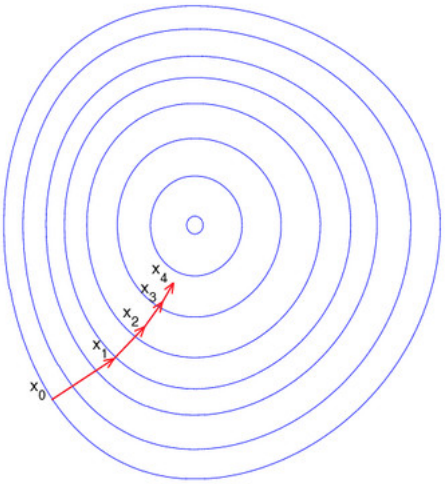
（1）L1 正则化使参数为零（2）L2 正则化使参数减小

第七章　常见的几种最优化方法

学习和工作中遇到的大多问题都可以建模成一种最优化模型进行求解，比如我们现在学习的机器学习算法，大部分的机器学习算法的本质都是建立优化模型，通过最优化方法对目标函数（或损失函数）进行优化，从而训练出最好的模型。常见的最优化方法有梯度下降法、牛顿法和拟牛顿法、共轭梯度法等等。

7.1　梯度下降法（Gradient Descent）

梯度下降法是最早最简单，也是最为常用的最优化方法。梯度下降法实现简单，当目标函数是凸函数时，梯度下降法的解是全局解。一般情况下，其解不保证是全局最优解，梯度下降法的速度也未必是最快的。梯度下降法的优化思想是用当前位置负梯度方向作为搜索方向，因为该方向为当前位置的最快下降方向，所以也被称为是”最速下降法“。最速下降法越接近目标值，步长越小，前进越慢。梯度下降法的搜索迭代示意图如下图所示：

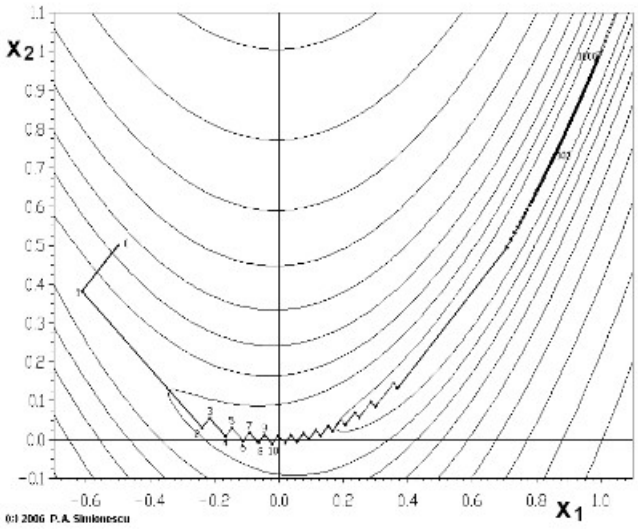


梯度下降法的缺点：

（1）靠近极小值时收敛速度减慢，如下图所示；

（2）直线搜索时可能会产生一些问题；

（3）可能会“之字形”地下降。



从上图可以看出，梯度下降法在接近最优解的区域收敛速度明显变慢，利用梯度下降法求解需要很多次的迭代。

在机器学习中，基于基本的梯度下降法发展了两种梯度下降方法，分别为随机梯度下降法和批量梯度下降法。

比如对一个线性回归（Linear Logistics）模型，假设下面的 h(x) 是要拟合的函数，$J(\theta)$ 为损失函数，θ 是参数，要迭代求解的值，θ 求解出来了，那最终要拟合的函数 $h(\theta)$ 就出来了。其中 m 是训练集的样本个数，n 是特征的个数。

7.1.1　批量梯度下降法（Batch Gradient Descent，BGD）

（1）将 J(theta) 对 theta 求偏导，得到每个 theta 对应的的梯度：

$$\sum_{i=1}^{m}\frac{\partial J(\theta, x_i)}{\partial \theta_j}$$

（2）由于是要最小化风险函数，所以按每个参数 theta 的梯度负方向，来更新每个 theta：

$$\theta_j := \theta_j - \eta \cdot \sum_{i=1}^{m}\frac{\partial J(\theta, x_i)}{\partial \theta_j}$$

（3）从上面公式可以注意到，它得到的是一个全局最优解，但是每迭代一步，都要用到训练集所有的数据，如果 m 很大，那么可想而知这种方法的迭代速度会相当的慢。所以，这就引入了另外一种方法——随机梯度下降。

对于批量梯度下降法，样本个数 m，x 为 n 维向量，一次迭代需要把 m 个样本全部带入计算，迭代一次计算量为 $m*n^2$。

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

（1）上面的风险函数可以写成如下这种形式，损失函数对应的是训练集中每个样本的梯度，而上面批量梯度下降对应的是所有的训练样本：

$$\frac{\partial J(\theta, x_i)}{\partial \theta_j}$$

（2）每个样本的损失函数，对 theta 求偏导得到对应梯度，来更新 theta：

$$\theta_j := \theta_j - \eta \cdot \frac{\partial J(\theta, x_i)}{\partial \theta_j}$$

（3）随机梯度下降是通过每个样本来迭代更新一次，如果样本量很大的情况（例如几十万），那么可能只用其中几万条或者几千条的样本，就已经将 theta 迭代到最优解了，对比上面的批量梯度下降，迭代一次需要用到十几万训练样本，一次迭代不可能最优，如果迭代 10 次的话就需要遍历训练样本 10 次。但是，SGD 伴随的一个问题是噪音较 BGD 要多，使得 SGD 并不是每次迭代都向着整体最优化方向。

随机梯度下降每次迭代只使用一个样本，迭代一次计算量为 n^2，当样本个数 m 很大的时候，随机梯度下降迭代一次的速度要远高于批量梯度下降方法。两者的关系可以这样理解：随机梯度下降方法以损失很小的一部分精确度和增加一定数量的迭代次数为代价，换取了总体的优化效率的提升。增加的迭代次数远远小于样本的数量。

对批量梯度下降法和随机梯度下降法的总结：

批量梯度下降—最小化所有训练样本的损失函数，使得最终求解的是全局的最优解，即求解的参数是使得风险函数最小，但是对于大规模样本问题效率低下。

随机梯度下降—最小化每条样本的损失函数，虽然不是每次迭代得到的损失函数都向着全局最优方向，但是大的整体的方向是向全局最优解的，最终的结果往往是在全局最优解附近，适用于大规模训练样本情况。

7.2　牛顿法和拟牛顿法（Newton's method Quasi-Newton Methods）

7.2.1　牛顿法（Newton's method）

牛顿法是一种在实数域和复数域上近似求解方程的方法。方法使用函数 $f(x)$ 的泰勒级数的前面几项来寻找方程 $f(x)=0$ 的根。牛顿法最大的特点就在于它的收敛速度很快。

具体步骤：

首先，选择一个接近函数 $f(x)$ 零点的 x_0，计算相应的 $f(x_0)$ 和切线斜率 $f'(x_0)$（这里 f' 表示函数 f 的导数）。然后我们计算穿过点 $(x_0, f(x_0))$ 并且斜率为 $f'(x_0)$ 的直线和 x 轴的交点的 x，也就是求如下方程的解：

$$x \cdot f'(x_0) + f(x_0) - x_0 \cdot f'(x_0) = 0$$

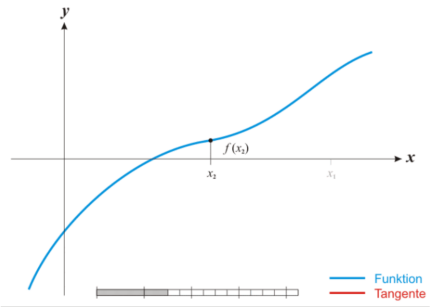
我们将新求得的点的 x 坐标命名为 x_1，通常 x_1 会比 x_0 更接近方程 $f(x)=0$ 的解。因此我们现在可以利用 x_1 开始下一轮迭代。迭代公式可化简为如下所示：

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

已经证明，如果 f' 是连续的，并且待求的零点 x 是孤立的，那么在零点 x 周围存在一个区域，只要初始值 x_0 位于这个邻近区域内，那么牛顿法必定收敛。并且，如果 $f'(x)$ 不为 0, 那么牛顿法将具有平方收敛的性能. 粗略的说，这意味着每迭代一次，牛顿法结果的有效数字将增加一倍。

由于牛顿法是基于当前位置的切线来确定下一次的位置，所以牛顿法又被很形象地称为是" 切线法"。牛顿法的搜索路径（二维情况）如下图所示：

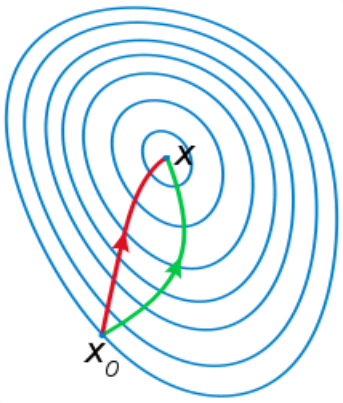
牛顿法搜索动态示例图：



关于牛顿法和梯度下降法的效率对比：

从本质上去看，牛顿法是二阶收敛，梯度下降是一阶收敛，所以牛顿法就更快。如果更通俗地说的话，比如你想找一条最短的路径走到一个盆地的最底部，梯度下降法每次只从你当前所处位置选一个坡度最大的方向走一步，牛顿法在选择方向时，不仅会考虑坡度是否够大，还会考虑你走了一步之后，坡度是否会变得更大。所以，可以说牛顿法比梯度下降法看得更远一点，能更快地走到最底部。（牛顿法目光更加长远，所以少走弯路；相对而言，梯度下降法只考虑了局部的最优，没有全局思想。）

根据 wiki 上的解释，从几何上说，牛顿法就是用一个二次曲面去拟合你当前所处位置的局部曲面，而梯度下降法是用一个平面去拟合当前的局部曲面，通常情况下，二次曲面的拟合会比平面更好，所以牛顿法选择的下降路径会更符合真实的最优下降路径。



注：红色的牛顿法的迭代路径，绿色的是梯度下降法的迭代路径。

牛顿法的优缺点总结：

优点：二阶收敛，收敛速度快；

缺点：牛顿法是一种迭代算法，每一步都需要求解目标函数的 Hessian 矩阵的逆矩阵，计算比较复杂。

7.2.2　拟牛顿法（Quasi-Newton Methods）

拟牛顿法是求解非线性优化问题最有效的方法之一。

拟牛顿法的本质思想是改善牛顿法每次需要求解复杂的 Hessian 矩阵的逆矩阵的缺陷，它使用正定矩阵来近似 Hessian 矩阵的逆，从而简化了运算的复杂度。拟牛顿法和最速下降法一样只要求每一步迭代时知道目标函数的梯度。通过测量梯度的变化，构造一个目标函数的模型使之足以产生超线性收敛性。这类方法大大优于最速下降法，尤其对于困难的问题。另外，因为拟牛顿法不需要二阶导数的信息，所以有时比牛顿法更为

有效。如今，优化软件中包含了大量的拟牛顿算法用来解决无约束，约束，和大规模的优化问题。

具体步骤：

拟牛顿法的基本思想如下。首先构造目标函数在当前迭代 x_k 的二次模型：

$$m_k(p) = f(x_k) + \nabla f(x_k)^T p + \frac{p^T B_k p}{2}$$
$$p_k = -B_k^{-1} \nabla f(x_k)$$

这里 B_k 是一个对称正定矩阵，于是我们取这个二次模型的最优解作为搜索方向，并且得到新的迭代点：

$$x_{k+1} = x_k + a_k p_k$$

其中我们要求步长 a_k 满足 Wolfe 条件。这样的迭代与牛顿法类似，区别就在于用近似的 Hesse 矩阵 B_k 代替真实的 Hesse 矩阵。所以拟牛顿法最关键的地方就是每一步迭代中矩阵 B_k 的更新。现在假设得到一个新的迭代 x_{k+1}，并得到一个新的二次模型：

$$m_{k+1}(p) = f(x_{k+1}) + \nabla f(x_{k+1})^T p + \frac{p^T B_{k+1} p}{2}$$

我们尽可能地利用上一步的信息来选取 B_k。具体地，我们要求:

从而得到

这个公式被称为割线方程。常用的拟牛顿法有 DFP 算法和 BFGS 算法。

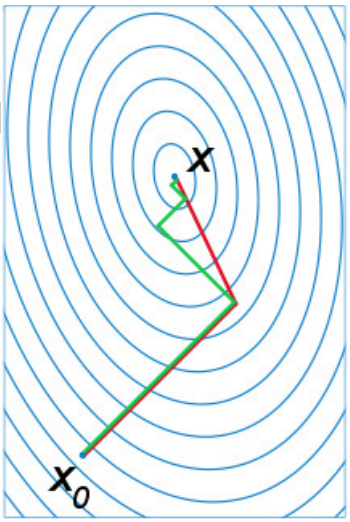
$$\nabla f(x_{k+1}) - \nabla f(x_k) = a_k B_{k+1} p_k$$

$$B_{k+1}(x_{k+1} - x_k) = \nabla f(x_{k+1}) - \nabla f(x_k)$$

7.3　共轭梯度法（Conjugate Gradient）

共轭梯度法是介于最速下降法与牛顿法之间的一个方法，它仅需利用一阶导数信息，但克服了最速下降法收敛慢的缺点，又避免了牛顿法需要存储和计算 Hesse 矩阵并求逆的缺点，共轭梯度法不仅是解决大型线性方程组最有用的方法之一，也是解大型非线性最优化最有效的算法之一。在各种优化算法中，共轭梯度法是非常重要的一种。其优点是所需存储量小，具有步收敛性，稳定性高，而且不需要任何外来参数。

下图为共轭梯度法和梯度下降法搜索最优解的路径对比示意图：



注：绿色为梯度下降法，红色代表共轭梯度法

第八章　梯度下降法的三种形式

在应用机器学习算法时，我们通常采用梯度下降法来对采用的算法进行训练。其实，常用的梯度下降法还具体包含有三种不同的形式，它们也各自有着不同的优缺点。

下面我们以线性回归算法来对三种梯度下降法进行比较。

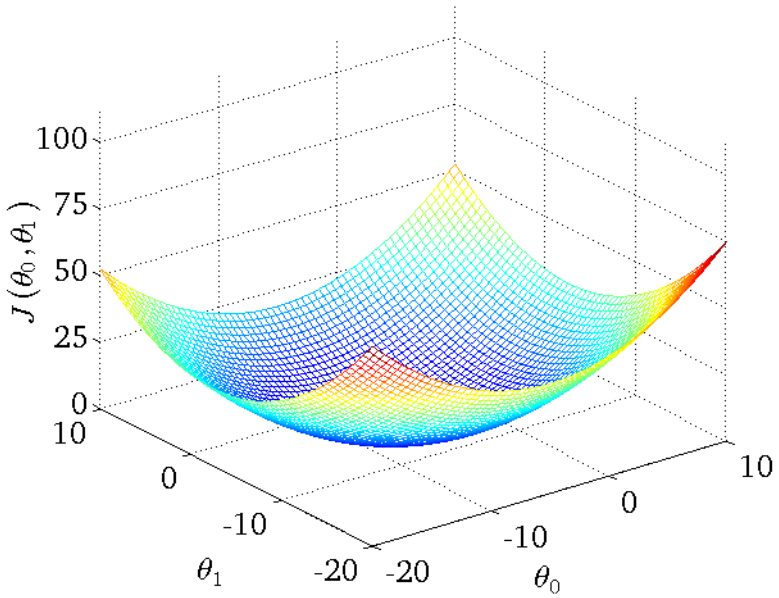
一般线性回归函数的假设函数为：

$$h_\theta = \sum_{j=0}^{n} \theta_j x_j$$

对应的能量函数（损失函数）形式为：

$$J_{train}(\theta) = 1/(2m) \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$$

下图为一个二维参数 (θ_0, θ_1) 组对应能量函数的可视化图：



8.0.1　批量梯度下降法 BGD

批量梯度下降法（Batch Gradient Descent，简称 BGD）是梯度下降法最原始的形式，它的具体思路是在更新每一参数时都使用所有的样本来进行更新，

其数学形式如下：

(1) 对上述的能量函数求偏导：

$$\frac{\partial J(\theta)}{\partial \theta_j} = -\frac{1}{m}\sum_{i=1}^{m}(y^i - h_\theta(x^i))x_j^i$$

(2) 由于是最小化风险函数，所以按照每个参数 θ 的梯度负方向来更新每个 θ:

$$\theta_j' = \theta_j + \frac{1}{m}\sum_{i=1}^{m}(y^i - h_\theta(x^i))x_j^i$$

具体的伪代码形式为：

```
repeat {
```

$$\theta_j' = \theta_j + \frac{1}{m}\sum_{i=1}^{m}(y^i - h_\theta(x^i))x_j^i$$

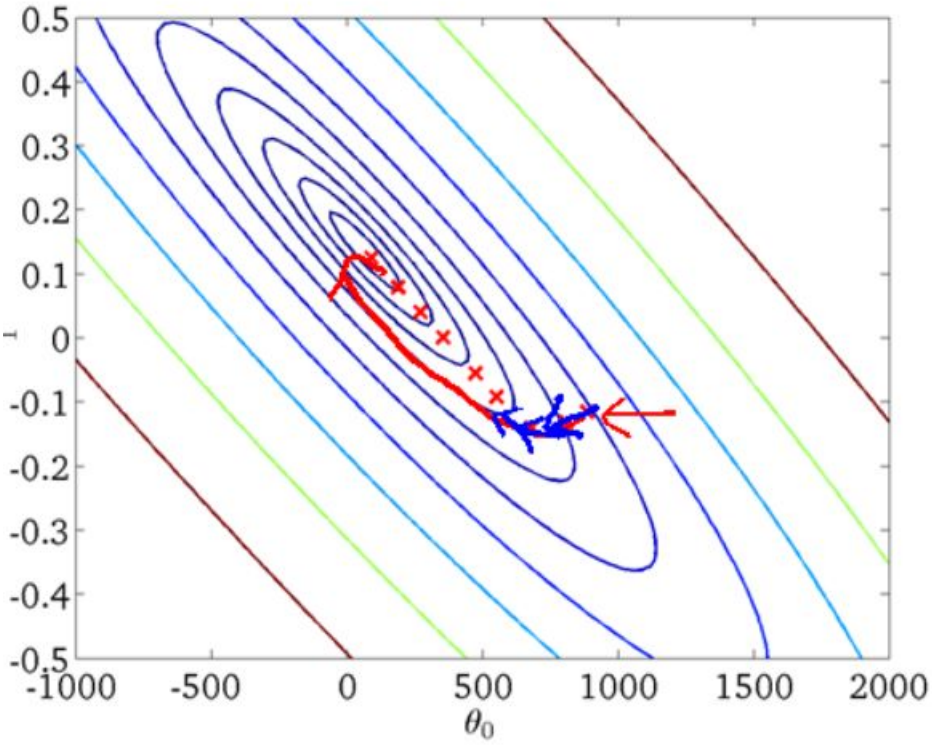
```
        (for every j=0, ... , n)
}
```

从上面公式可以注意到，它得到的是一个全局最优解，但是每迭代一步，都要用到训练集所有的数据，如果样本数目 m 很大，那么可想而知这种方法的迭代速度！所以，这就引入了另外一种方法，随机梯度下降。

优点：全局最优解；易于并行实现；

缺点：当样本数目很多时，训练过程会很慢。

从迭代的次数上来看，BGD 迭代的次数相对较少。其迭代的收敛曲线示意图可以表示如下：



8.0.2　随机梯度下降法 SGD

由于批量梯度下降法在更新每一个参数时，都需要所有的训练样本，所以训练过程会随着样本数量的加大而变得异常的缓慢。随机梯度下降法（Stochastic Gradient Descent，简称 SGD）正是为了解决批量梯度下降法这一弊端而提出的。

将上面的能量函数写为如下形式：

$$J(\theta)=\frac{1}{m}\sum_{i=1}^{m}\frac{1}{2}(y^i-h_\theta(x^i))^2=\frac{1}{m}\sum_{i=1}^{m}cost(\theta,(x^i,y^i))$$

$$cost(\theta,(x^i,y^i))=\frac{1}{2}(y^i-h_\theta(x^i))^2$$

利用每个样本的损失函数对 θ 求偏导得到对应的梯度，来更新 θ：

$$\theta_j^{'}=\theta_j+(y^i-h_\theta(x^i))x_j^i$$

具体的伪代码形式为：

随机梯度下降是通过每个样本来迭代更新一次，如果样本量很大的情况（例如几十万），那么可能只用其中几万条或者几千条的样本，就已经将　迭代到最优解了，对比上面的批量梯度下降，迭代一次需要用到十几万训练样本，一次迭代不可能最优，如果

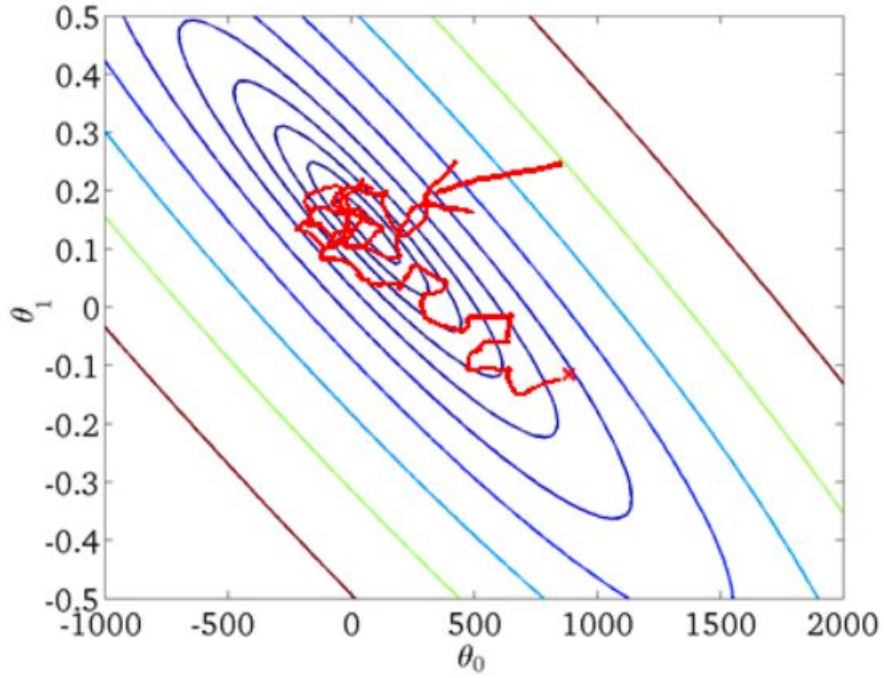
1. Randomly shuffle dataset;
2. **repeat** {
   for i=1, ... , m {
   $$\theta_j^{'} = \theta_j + (y^i - h_\theta(x^i))x_j^i$$
   (for j=0, ... , n)
   }
}

迭代 10 次的话就需要遍历训练样本 10 次。但是，SGD 伴随的一个问题是噪音较 BGD 要多，使得 SGD 并不是每次迭代都向着整体最优化方向。

优点：训练速度快；

缺点：准确度下降，并不是全局最优；不易于并行实现。

从迭代的次数上来看，SGD 迭代的次数较多，在解空间的搜索过程看起来很盲目。其迭代的收敛曲线示意图可以表示如下：



8.0.3　小批量梯度下降法 MBGD

有上述的两种梯度下降法可以看出，其各自均有优缺点，那么能不能在两种方法的性能之间取得一个折衷呢？即，算法的训练过程比较快，而且也要保证最终参数训练的

准确率，而这正是小批量梯度下降法（Mini-batch Gradient Descent，简称 MBGD）的初衷。

MBGD 在每次更新参数时使用 b 个样本（b 一般为 10），其具体的伪代码形式为：

```
Say b=10, m=1000.
Repeat {
    for i=1, 11, 21, 31, ... , 991 {
    θ_j := θ_j − α (1/10) Σ_{k=i}^{i+9} (h_θ(x^(k)) − y^(k)) x_j^(k)
    (for every j=0, ... , n)
    }
}
```