TechX 深度强化学习课程（二）

何舜成

宽带网数字媒体实验室
清华大学自动化系

2018 年 7 月 27 日

目录

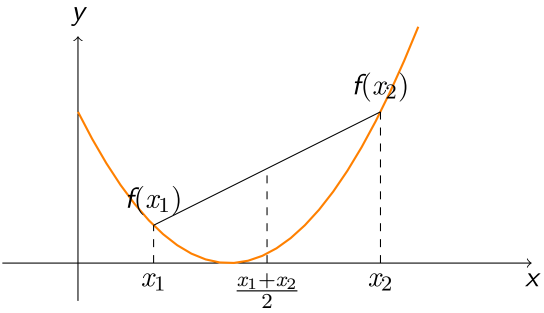
数值优化

凸函数

在最小二乘法里，为什么通过求偏导等于 0 就可以得到最小值呢？因为误差函数是一种凸函数，这种函数具有一些很好的性质。

一元函数中，凸函数指满足下面条件的函数

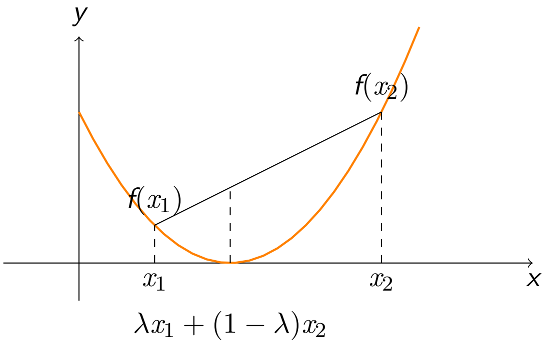
$$\forall x_1, x_2, \frac{1}{2}(f(x_1) + f(x_2)) \geq f\left(\frac{x_1 + x_2}{2}\right) \tag{1}$$



凸函数

可以证明凸函数还有一些等价定义，如

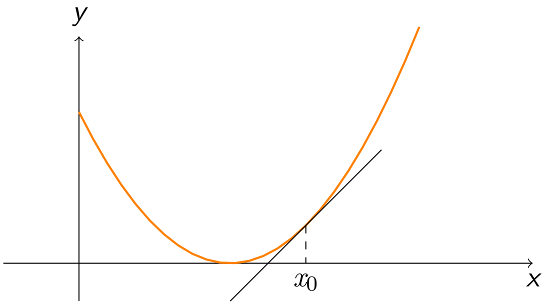
$$\forall x_1, x_2, 0 \le \lambda \le 1, \lambda f(x_1) + (1-\lambda) f(x_2) \ge f(\lambda x_1 + (1-\lambda) x_2) \tag{2}$$



凸函数

还可以用一阶导描述，即函数图像永远在切线的上方。

$$\forall x, x_0, f(x) \geq f'(x_0)(x - x_0) + f(x_0) \tag{3}$$



最重要的是凸函数还蕴含着二阶导不小于 0 的条件，即 $f''(x) \geq 0$。

凸函数的最值条件

如果某一个凸函数 $f(x)$ 只存在一个 x_0 使得 $f'(x_0) = 0$，那么该凸函数的最小值在 $x = x_0$ 时取得，最小值为 $f(x_0)$。

如果推广到多元函数，那么凸函数二阶导条件将变为 Hessian 矩阵条件，我们要求 Hessian 矩阵半正定，或者说 Hessian 矩阵的特征值全不小于 0。此时多元函数的最小值将在梯度等于 0（$\nabla f(\boldsymbol{x}) = \mathbf{0}$）时取得，如果该方程可解，那么可以据此求得最小值点与最小值。

凸函数的最值条件

回到之前的最小二乘法，误差函数被定义为

$$
\begin{aligned}
E &= \sum_{i=1}^{4} \epsilon_i^2 \\
&= [6-(\beta_1+1\beta_2)]^2+[5-(\beta_1+2\beta_2)]^2 \\
&\quad +[7-(\beta_1+3\beta_2)]^2+[10-(\beta_1+4\beta_2)]^2 \\
&= 4\beta_1^2+20\beta_1\beta_2+30\beta_2^2-56\beta_1-154\beta_2+210
\end{aligned}
$$

Hessian 矩阵为

$$
\boldsymbol{H} = \begin{bmatrix} 8 & 20 \\ 20 & 60 \end{bmatrix}
$$

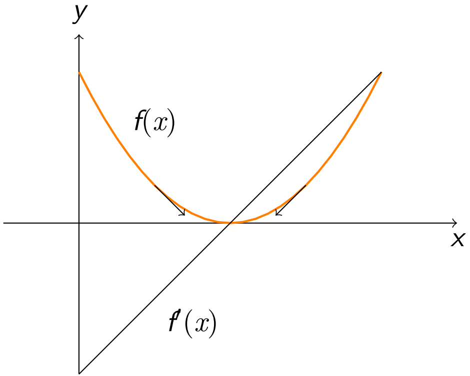
求得特征值 $\lambda_1 = 1.1976$，$\lambda_2 = 66.8024$，因此矩阵正定，误差函数是凸函数。

事实上，线性函数的平方和都是凸函数。

梯度优化方法

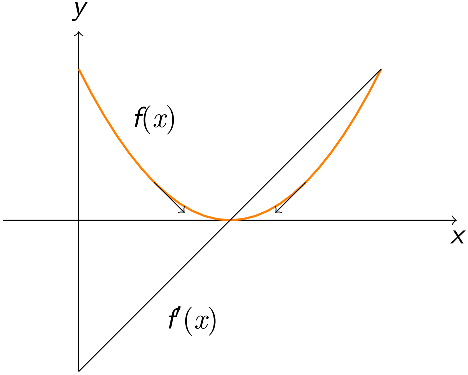
不管是分类还是回归，实质上是优化问题，都以最小化损失函数为目的。当目标函数为凸函数时，通过解 $\nabla f(x) = \mathbf{0}$ 可以直接得到最小值。但有些时候，这个方程无法求解，或者目标函数根本不是凸函数，我们就需要一些其他优化方法。

下图函数为 $f(x) = (x-2)^2/2$，其导数为 $f'(x) = x-2$



梯度优化方法

当我们随机选取一点 x_0 作为初始点，先计算其导数 $f'(x_0)$，若导数大于 0，说明 x_0 的左侧函数值更小，那么可以选取 $x_1 = x_0 - \epsilon$，分别计算该点的函数值与导数值；若导数小于 0，说明往右侧挪动函数值更小，此时选取 $x_1 = x_0 + \epsilon$。重复步骤上述步骤，直至导数值等于 0 或小于某个阈值。



梯度优化方法

多元函数情况下，负梯度方向是函数值下降最快的方向，此时我们的更新策略是

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \epsilon \nabla f(\boldsymbol{x}) \tag{4}$$

ϵ 通常是一个比较小的量，称为**步长**或**学习率**。这种梯度优化方法称为**最速下降法**。

应用

最速下降法可以用来解最小二乘。优化目标是

$$f(\boldsymbol{x}) = \frac{1}{2}\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2^2 \tag{5}$$

梯度是

$$\nabla \boldsymbol{x} = \boldsymbol{A}^\top(\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}) \tag{6}$$

确定一个步长 ϵ，从任意一点 $\boldsymbol{x}_0$ 开始，执行迭代方程 $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \epsilon\nabla\boldsymbol{x}_k$，直到梯度足够小（$\|\nabla\boldsymbol{x}\|_2^2 \leq \delta$）。

信息论基础

数据的经验分布

设 $f(x)$ 是某一概率分布的概率密度函数，$x_1, \ldots, x_n$ 是从该分布中随机抽取的 n 个样本。设另一随机变量 y 满足 $P(\mathrm{y} = x_i) = 1/n, i = 1, \ldots, n$，即 y 是从上述 n 个样本中等概率抽取的。称 y 遵循一种经验分布。

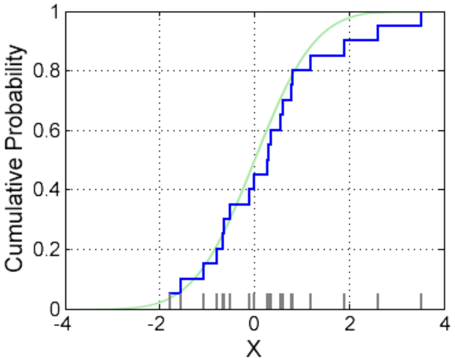


图 1: 经验分布的 cdf 可逼近原分布的 cdf

信息的量化

一张图片携带有多少信息？一段视频最多能被压缩到多小？将信息量化成数字之后，我们就可以解答这些问题。信息论是通信工程的基础，而在机器学习领域，信息论可以用来衡量概率分布分布之间的相似性。

考虑一件事情发生的可能性与携带的信息量之间的关系：

- 明天是晴天——晴天不是小概率事件，信息量不大
- 明天有雨夹雪——夏天几乎不可能下雪，信息量非常大

因此我们应当发现

- 小概率事件有更多的信息量，大概率事件有更少的信息量
- 多个独立事件的信息量应该是各事件信息量相加

信息的量化

可以设计这样一个函数来刻画一个随机变量的自信息

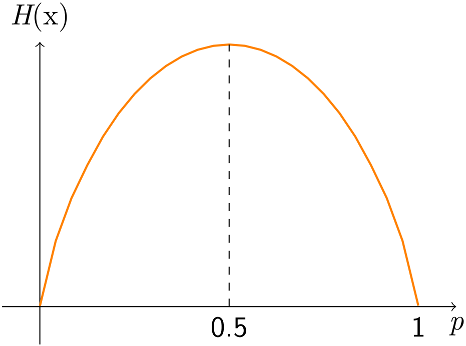
$$I(x) = -\log P(x) \tag{7}$$

在之后的描述中，log 将替代 ln 来表示自然对数。自信息只能衡量单个事件的信息量，如果要衡量一个概率分布的不确定性，需要用类似热学中的熵——信息熵

$$H(\mathrm{x}) = \mathbb{E}_{\mathrm{x}\sim P}[I(x)] = -\mathbb{E}_{\mathrm{x}\sim P}[\log P(x)] \tag{8}$$

信息的量化

二值随机变量的信息熵为 $(p-1)\log(1-p) - p\log p$，其中 p 为事件 $\mathrm{x}=1$ 发生的概率，信息熵随参数 p 变化的规律如下图



KL 散度

在分类问题里，算法输出的往往是样本 x 属于 C_i 类的概率，结合起来看，算法给出的是一个定义在 $C_1, \ldots, C_n$ 这 n 个类别上的概率分布。如果要衡量这个概率分布与真实值之间的差别，那么就需要用到 KL 散度：

$$D_{\mathrm{KL}}(P||Q) = \mathbb{E}_{\mathrm{x}\sim P}\left[\log \frac{P(x)}{Q(x)}\right] = \mathbb{E}_{\mathrm{x}\sim P}[\log P(x) - \log Q(x)] \tag{9}$$

KL 散度非负，当 KL 散度为零是，当且仅当（1）离散情况下，两概率分布完全相同，（2）连续情况下，量概率分布“几乎处处”相同。但 KL 散度不是真正的一种“距离”，因为不满足对称性 $D_{\mathrm{KL}}(P||Q) \neq D_{\mathrm{KL}}(Q||P)$。

交叉熵

交叉熵定义为

$$H(P, Q) = H(P) + D_{\mathrm{KL}}(P||Q) = -\mathbb{E}_{\mathrm{x}\sim P} \log Q(x) \tag{10}$$

最大似然估计准则

考虑一个广义的线性回归问题，有 m 个样本的数据集 $\mathbb{X} = \{(x_1, y_1), \ldots, (x_m, y_m)\}$ 由一个未知的联合概率分布 $P_{\text{data}}(x, y)$ 生成，我们要用一个带参数 $\boldsymbol{w}$ 的模型 $y = w_0 + w_1 x$ 去拟合这些样本，这个等式关系也可以视作一个概率分布 $P_{\text{model}}(x, y; \boldsymbol{w})$。

进一步抽象，$\mathbb{X} = \{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{m}\}$ 是从 $P_{\text{data}}(\mathrm{x})$ 中独立抽取的 m 个样本，我们用带参数的模型 $P_{\text{model}}(\mathrm{x}; \boldsymbol{\theta})$ 去拟合这些样本。考虑一个自然而然的准则

$$\boldsymbol{\theta}_{\text{ML}} = \arg\max_{\boldsymbol{\theta}} p_{\text{model}}(\mathbb{X}; \boldsymbol{\theta}) \tag{11}$$

这个估计准则叫**最大似然估计**。

最大似然估计准则

由于 m 个样本相互独立，该式还可以写成

$$\boldsymbol{\theta}_{\mathrm{ML}} = \arg\max_{\boldsymbol{\theta}} \prod_{i=1}^{m} p_{\mathrm{model}}(\boldsymbol{x}^{(i)}; \boldsymbol{\theta}) \tag{12}$$

因为连乘式不容易优化，加之对数函数是单调增的，上式还可以写成

$$\boldsymbol{\theta}_{\mathrm{ML}} = \arg\max_{\boldsymbol{\theta}} \sum_{i=1}^{m} \log p_{\mathrm{model}}(\boldsymbol{x}^{(i)}; \boldsymbol{\theta}) \tag{13}$$

利用经验分布，将其写成期望的形式

$$\boldsymbol{\theta}_{\mathrm{ML}} = \arg\max_{\boldsymbol{\theta}} \mathbb{E}_{\mathrm{x}\sim\hat{p}_{\mathrm{data}}} \log p_{\mathrm{model}}(\boldsymbol{x}; \boldsymbol{\theta}) \tag{14}$$

最大似然估计准则

联想到 KL 散度，我们可以计算一下模型产生的分布与数据的经验分布之间的 KL 散度

$$D_{\mathrm{KL}}(\hat{p}_{\mathrm{data}}||p_{\mathrm{model}}) = \mathbb{E}_{\mathrm{x}\sim\hat{p}_{\mathrm{data}}}[\log \hat{p}_{\mathrm{data}}(\boldsymbol{x}) - \log p_{\mathrm{model}}] \quad (15)$$

考虑到第一项是常量，当最小化 KL 散度时，等价于最大化似然度。

结论：最大似然估计是使得样本的经验分布与模型分布的 KL 散度最小的分布。

再探最小二乘法

最大似然估计还可以用于估计条件概率 $P(y|\boldsymbol{x};\boldsymbol{\theta})$。当给定样本的输入 $\boldsymbol{X}$ 时，我们要找到一组参数 $\boldsymbol{\theta}$ 最能预测真实值 $\boldsymbol{Y}$，即

$$\boldsymbol{\theta}_{\mathrm{ML}} = \arg\max_{\boldsymbol{\theta}} P(\boldsymbol{Y}|\boldsymbol{X};\boldsymbol{\theta}) = \arg\max_{\boldsymbol{\theta}} \sum_{i=1}^{m} P(y^{(i)}|\boldsymbol{x}^{(i)};\boldsymbol{\theta}) \quad (16)$$

之前的线性回归算法学习的是输入 $\boldsymbol{x}$ 到确定值 $\hat{y}$ 的算法。实际上，我们的样本可能是从一个概率分布中随机抽取的，我们也需要学习一个概率分布，即 $p(y|\boldsymbol{x})$。通常我们会假设这是正态分布 $p(y|\boldsymbol{x}) = \mathcal{N}(y;\hat{y}(\boldsymbol{x};\boldsymbol{w}),\sigma^2)$，之前的函数 $\hat{y}(\boldsymbol{x};\boldsymbol{w})$ 预测的是正态分布的均值，σ^2 是一固定的方差。

再探最小二乘法

应用最大似然估计

$$\sum_{i=1}^{m} \log p(y^{(i)}|\boldsymbol{x}^{(i)};\boldsymbol{\theta}) = -m\log\sigma - \frac{m}{2}\log(2\pi) - \sum_{i=1}^{m}\frac{\left\|\hat{y}^{(i)} - y^{(i)}\right\|^2}{2\sigma^2} \quad (17)$$

与最小化均方误差 $\sum_{i=1}^{m}\left\|\hat{y}^{(i)} - y^{(i)}\right\|^2/m$ 等价。

监督学习

监督学习与非监督学习

监督学习是十分常见的学习问题，在给定样本中，有输入数据 $\boldsymbol{X}$ 和真值 $\boldsymbol{Y}$（groundtruth、label），需要用这些带标签的样本去训练算法，使之能很好地进行预测。分类和回归都是监督学习问题，区别就在于真值是离散的还是连续的。

而非监督学习没有这些真值，要学习的是数据内部的结构。

监督学习与非监督学习

聚类是很典型的非监督学习问题。

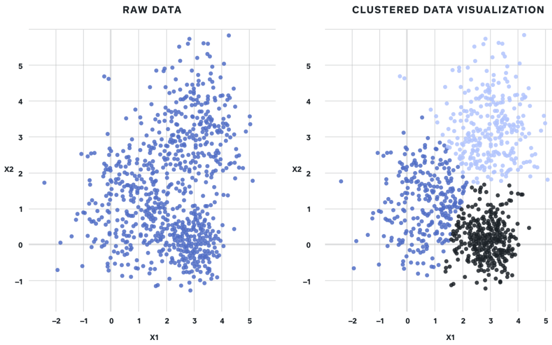


图 2: 将杂乱的数据分类聚合

非监督学习不是本课程需要重点关注的内容。

Logistic 回归

考虑一组数据 $\boldsymbol{X}=\{\boldsymbol{x}_1,\ldots,\boldsymbol{x}_m\}$，以及一组标签 $\boldsymbol{Y}=\{y_1,\ldots,y_m\},\forall i=1,\ldots,m,y_i\in\{0,1\}$。这是一个二分类问题，Logistic 回归输出数据点 $\boldsymbol{x}$ 为分类为 1 的概率

$$p(y=1|\boldsymbol{x};\boldsymbol{\theta})=\sigma(\boldsymbol{w}^\top\boldsymbol{x}) \tag{18}$$

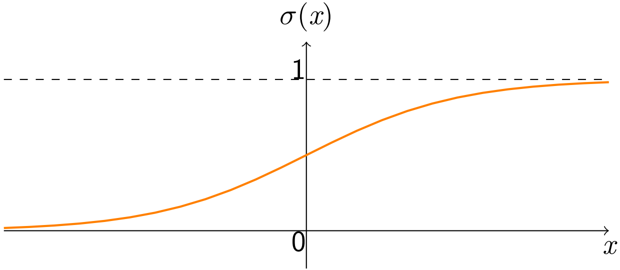
其中 $\sigma(x)$ 称为 Sigmoid 函数。表达式为

$$\sigma(x)=\frac{1}{1+e^{-x}} \tag{19}$$

Logistic 回归

Sigmoid 函数有很好的性质：

- 连续、可微分
- 将实数值区间 $[-\infty, +\infty]$ 映射到 $[0, 1]$
- 非线性
- 导数易于计算 $\sigma'(x) = \sigma(x)(1 - \sigma(x))$



支持向量机

假设数据点的标签均为 +1 或-1，我们需要用一个平面将两类数据点分隔开，为了提高区分度，两类数据点之间的间隔要足够大，这就是最大间隔分类器

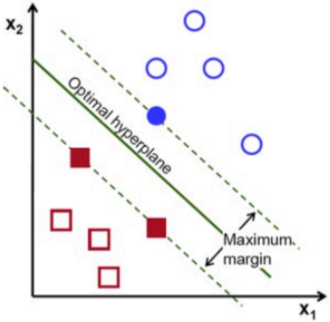


图 3: 最大间隔分类器示意

支持向量机

写成优化问题的形式：

$$
\begin{aligned}
\max_{\boldsymbol{w}, b, t} \quad & t \\
s.t. \quad & y_i(\boldsymbol{w}^\top \boldsymbol{x}_i + b) \geq t, \forall i \in [n] \\
& \|\boldsymbol{w}\|_2^2 = 1
\end{aligned}
$$

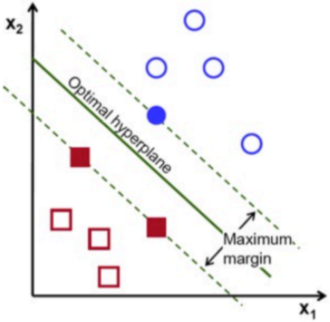
这个问题还等价于

$$
\begin{aligned}
\min_{\boldsymbol{w}, b} \quad & \frac{1}{2} \|\boldsymbol{w}\|_2^2 \\
s.t. \quad & y_i(\boldsymbol{w}^\top \boldsymbol{x}_i + b) \geq 1, \forall i \in [n]
\end{aligned}
$$

等价性留作习题。

支持向量机

从直觉上看，间隔最大的时候，一定会有两个类别的数据点分别落在两条绿色虚线上（为什么?）。这些数据点被称为**支持向量**，而且如果去掉支持向量以外的数据点，最优的解并不会改变（为什么?）。因此最优解完全取决于支持向量，这种分类器被称为**支持向量机**。



正则化方法

多项式拟合

同样考虑拟合空间上的一些点，除去线性模型，我们还可以考虑多项式模型。当我们用一次、二次、九次多项式去分别拟合，会发现这样的现象

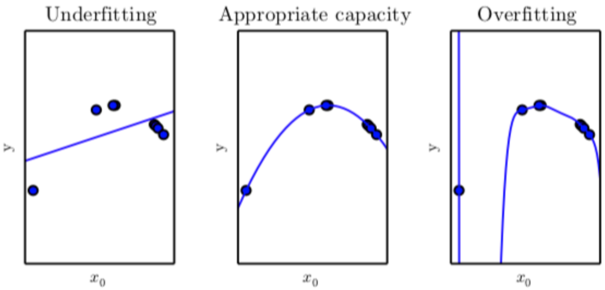


图 4: 模型容量与拟合的好坏

多项式拟合

当模型是一次的时候，无论如何也拟合不好，此时是**欠拟合**状态；模型是二次时，拟合得刚刚好；模型是 9 次多项式时，虽然每个点都被精确拟合，但能明显看到真实数据分布并没有这样的夸张的规律，此时是**过拟合**状态。

拟合得好不好，根本原因在于模型容量的选择。多项式一次、二次、九次，模型能表达的曲线越来越多、越来越复杂，模型容量依次增加。容量不够会导致欠拟合问题，容量太大会导致过拟合问题，因此选择合适的容量很关键。

除了选择合适的模型，还有一类降低模型容量以提高泛化能力的方法，即正则化方法。

正则化的作用

假如我们在九次多项式拟合的均方误差上加上 $\lambda \boldsymbol{w}^\top \boldsymbol{w}$ 作为惩罚

$$J(\boldsymbol{w}) = \frac{1}{2}\|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}\|_2^2 + \frac{1}{2}\lambda \boldsymbol{w}^\top \boldsymbol{w} \tag{20}$$

为了最小化二者之和，我们会偏好那些权值的范数小的那些解，实际上缩小了模型容量，可以防止过拟合。

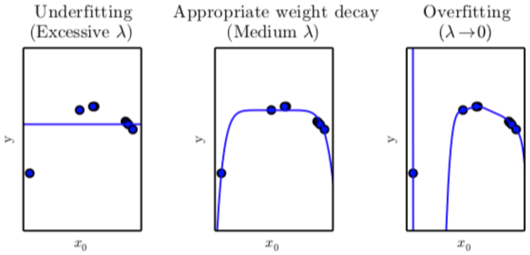


图 5: λ 参数的作用

岭回归

若在线性回归的损失函数上加上权值的 L^2 范数惩罚项，这种回归方法称为**岭回归**。如果从梯度下降算法的优化角度看，此时梯度变为

$$\nabla J(\boldsymbol{w}) = \boldsymbol{X}^\top(\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y}) + \lambda\boldsymbol{w} \tag{21}$$

迭代方程变为

$$\boldsymbol{w}_{k+1} = \boldsymbol{w}_k - \epsilon(\boldsymbol{X}^\top(\boldsymbol{X}\boldsymbol{w}_k - \boldsymbol{y}) + \lambda\boldsymbol{w}_k) \tag{22}$$

可以看到，每次迭代，权值都会衰减一定比例，因此 L^2 范数惩罚又称为**权值衰减**算法。

岭回归

从下图可以看到，为了平衡原目标函数与权值范数的惩罚项，优化算法最终会选择比最优解 $\boldsymbol{w}^*$ 范数更小的 $\tilde{\boldsymbol{w}}$，一定程度上限制了模型中参数的选择范围。

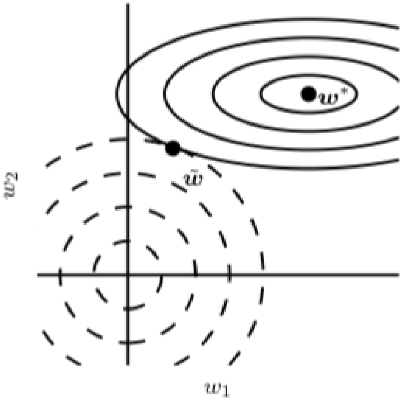


图 6: L^2 范数惩罚效果

岭回归

通过解 $\nabla J(\boldsymbol{w}) = \mathbf{0}$ 可以发现，精确解从

$$\boldsymbol{w}^* = (\boldsymbol{X}^\top \boldsymbol{X})^{-1} \boldsymbol{X}^\top \boldsymbol{y} \tag{23}$$

变成了

$$\tilde{\boldsymbol{w}} = (\boldsymbol{X}^\top \boldsymbol{X} + \lambda \boldsymbol{I})^{-1} \boldsymbol{X}^\top \boldsymbol{y} \tag{24}$$

由线性代数知识，当 $\boldsymbol{X}$ 中出现许多（接近）线性相关的数据点，或 $m \leq n$ 时，$\boldsymbol{X}^\top \boldsymbol{X}$ 将出现接近或等于 0 的特征值，求逆操作可能产生数值不稳定。而加上 $\lambda \boldsymbol{I}$ 之后，可以保证特征值全不小于 λ，将提高求逆的稳定性。

Lasso 回归

Lasso 回归是在原基础上增加了权值的 L^1 范数惩罚。此时的损失函数为

$$J(\boldsymbol{w}) = \frac{1}{2}\|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}\|_2^2 + \lambda\|\boldsymbol{w}\|_1 \tag{25}$$

此时的梯度（次梯度）为

$$\nabla J(\boldsymbol{w}) = \boldsymbol{X}^\top(\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y}) + \lambda sgn(\boldsymbol{w}) \tag{26}$$

$sgn(\boldsymbol{w})$ 表示根据 $\boldsymbol{w}$ 每一维的符号取 $+1, -1$。

Lasso 回归

与 L^2 范数惩罚相比，L^1 范数惩罚会导致一种稀疏性，使得某些特征维度对应的权值为 0。

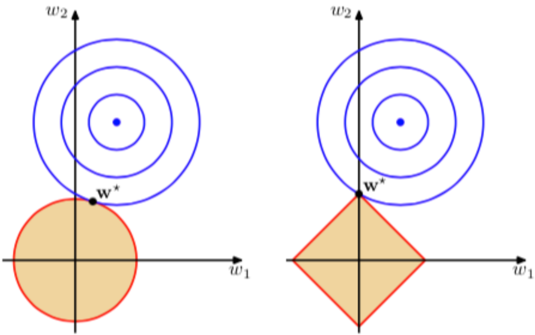


图 7: L^1 范数惩罚导致稀疏性

这种稀疏性可以用作特征选择。