机器学习笔记

Notes on Machine Learning

J.R.Tsien
jade.ray.tsien@gmail.com

目　　录

第1章　绪论

§ 1.1 最优化问题

1.1.1 原始问题

1.1.2 拉格朗日对偶问题

§ 1.2 梯度下降法（gradient descent）

梯度下降法或者最速下降法（steepest descent）是求解无约束最优化问题的方法。特点是实现起来比较简单。其原理是如果函数$f(x)$在点a处可微且有定义，那么函数$f(x)$在a点沿着梯度的反方向，即$-\nabla f(a)$，下降最快。

所以，可以从一个初始值x_0出发，沿梯度反方向迭代的更新解。如下

$$x_{n+1} = x_n - \alpha \nabla f(x_n)$$

直到x_n的值不再发生变化，或者变化很小，此时，x_n等于或者接近$f(x)$的极小值。α称为学习率（learning rate）。α值过大，可能会在最小值附近振荡。α值过小，可能学习的时间比较长。同时，α值的选取可以是预先设定的固定值，也可以是根据解更新的情况变化的值。

梯度下降法的一个问题在于，能否得到最优解取决于初始值的选取。

§ 1.3 牛顿法和拟牛顿法

1.3.1 牛顿法（Newton Method）

牛顿法，或牛顿-拉夫逊法（Newton-Raphson Method）也是求解无约束优化问题的常用方法。牛顿法是二阶收敛的算法（不仅考虑梯度方向，同时考虑梯度的梯度），而梯度下降法是一阶收敛的，因此牛顿法的收敛速度比梯度下降快。换句话说，牛顿法用二次曲面来拟合当前所在位置的局部曲面，然后按照曲率最大的方向下降。而梯度法是用一个平面去拟合局部曲面，然后按照平面的法向量的方向下降。但是一次迭代的代价比较高，因为需要计算矩阵的逆。

考虑无约束的最优化问题

$$\min_{x\in\mathbb{R}^n} f(x)$$

假设$f(x)$有二阶连续偏导数，且设第k次迭代的解为x_k，将$f(x)$在点x_k处进行二阶泰勒展开

$$f(x)=f(x_k)+(x-x_k)f'(x_k)+\frac{1}{2}(x-x_k)^2f''(x_k)$$

函数$f(x)$在下次迭代点x_{k+1}处取得极值的必要条件是$f'(x_{k+1})=0$，即

$$f'(x)|_{x_{k+1}}=f'(x_k)+(x_{k+1}-x_k)f''(x_k)=0$$

解上式得到

$$x_{k+1}=x_k-\frac{f'(x_k)}{f''(x_k)}$$

迭代停止的条件可以设定为$f'(x_k)<\epsilon$。

当x是向量的时候，其一阶导数要修改成梯度的形式，二阶导数修改成其Hessian矩阵，即

$$f(x)=f(x_k)+(x-x_k)\nabla_x f(x_k)+\frac{1}{2}(x-x_k)^T H(x_k)(x-x_k)$$

当$H(x_k)$是正定矩阵时，$f(x)$的极值为极小值。其更新公式是

$$x_{k+1}=x_k-H^{-1}(x_k)\nabla_x f(x_k)$$

迭代终止的条件$\nabla_x f(x_k)<\epsilon$.

当初始点离极值点较远时，牛顿法可能不收敛，因为此时由Hessian矩阵的逆矩阵和梯度规定的牛顿方向不一定是下降方向。

1.3.2 拟牛顿法（Quasi Newton Method）

牛顿法中Hessian矩阵求逆比较复杂，所以实际中会考虑用一个近似的正定对称矩阵来代替Hessian矩阵，这就是拟牛顿法。不同的替代方法形成了不同的拟牛顿法。

1.3.2.1 拟牛顿条件

首先将函数$f(x)$在第k次迭代点处泰勒展开

$$f(x) = f(x_k) + (x - x_k)\nabla_x f(x_k) + \frac{1}{2}(x - x_k)^T H(x_k)(x - x_k)$$

求其此时的梯度

$$\nabla f(x) = \nabla f(x_k) + (x - x_k)H(x_k)$$

令$x = x_{k+1}$，

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

左边是梯度的变化，右边$(x_{k+1} - xk)$是自变量的变化，分别记为

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

$$\delta_k = x_{k+1} - x_k$$

得到拟牛顿条件

$$g_k = H_k\delta_k \tag{1.1}$$

或

$$H_k^{-1}g_k = \delta_k \tag{1.2}$$

拟牛顿法选择$G_k \approx H_k^{-1}$，或者$B_k \approx H_k$。而且，尽量使得近似矩阵的更新方式为迭代更新

$$G_{k+1} = G_k + \Delta G_k$$

且G_k满足拟牛顿条件

$$G_{k+1}g_k = \delta_k \tag{1.3}$$

一般令$G_0 = I$为单位阵，所以只需要找到ΔG_k即可。

1.3.2.2 DFP算法

DFP算法最早由William C. Davidon于1959年提出，后由Roger Fletcher和Michael J.D. Powell发展和完善。DFP算法令校正矩阵为

$$\Delta G_k = \frac{\delta_k\delta_k^T}{\delta_k^T g_k} - \frac{G_k g_k g_k^T G_k}{g_k^T G_k g_k}$$

构造过程主要的规则在于保证如果初始矩阵G_0正定，那么每个G_k都是正定。

DFP算法

输入：目标函数$f(x)$，梯度$\nabla f(x)$，精度要求ϵ

输出：$f(x)$的极小值点x^*

（1）选定初始点x_0，取G_0为正定对称矩阵（单位阵），置$k=0$
（2）计算$\nabla f(x_k)$，若$\|\nabla f(x_k)\|<\epsilon$，令$x^\star=x_k$，停止计算
（3）令$p_k=-G_k\nabla f(x_k)$
（4）一维搜索：求λ_k使得

$$f(x_k+\lambda_k p_k)=\min_{\lambda\geq 0} f(x_k+\lambda p_k)$$

（5）令$x_{k+1}=x_k+\lambda_k p_k$
（6）计算$\nabla f(x_{k+1})$，若$\|\nabla f(x_{k+1})\|<\epsilon$，令$x^\star=x_{k+1}$，停止计算；否则，按$G_{k+1}=G_k+\Delta G_k$计算$G_{k+1}$
（7）置$k=k+1$，转（3）

1.3.2.3 BFGS算法

BFGS算法（Broyden-Fletcher-Goldfarb-Shanno）是最流行的拟牛顿法。DFP用一个正定矩阵来近似H^{-1}，BFGS用一个正定矩阵来近似H，其表示如下

$$B_{k+1}=B_k+\Delta B_k \tag{1.4}$$

$$\Delta B_k=\frac{g_k g_k^T}{g_k^T\delta_k}-\frac{B_k\delta_k\delta_k^T B_k}{\delta_k^T B_k\delta_k} \tag{1.5}$$

构造的核心也是满足如果初始矩阵B_0是正定的，那么迭代过程中的每一个B_k都是正定的。

BFGS算法

输入：目标函数$f(x)$，梯度$\nabla f(x)$，精度ϵ

输出：$f(x)$的极小值点$x^\star$

（1）选定初始点x_0，取B_0为正定对称矩阵（单位阵），置$k=0$
（2）计算$\nabla f(x_k)$，若$\|\nabla f(x_k)\|<\epsilon$，令$x^\star=x_k$，停止计算
（3）由$B_k p_k=-\nabla f(x_k)$求出p_k
（4）一维搜索：求λ_k使得

$$f(x_k+\lambda_k p_k)=\min_{\lambda\geq 0} f(x_k+\lambda p_k)$$

（5）令$x_{k+1}=x_k+\lambda_k p_k$
（6）计算$\nabla f(x_{k+1})$，若$\|\nabla f(x_{k+1})\|<\epsilon$，令$x^\star=x_{k+1}$，停止计算；否则，按$B_{k+1}=B_k+\Delta B_k$计算$B_{k+1}$
（7）置$k=k+1$，转（3）

令$G_k=B_k^{-1}$，对$B_{k+1}=B_k+\Delta B_k$运用两次Sherman-Morrison公式有

$$G_{k+1}=(I-\frac{\delta_k g_k^T}{\delta_k^T g_k})G_k(I-\frac{\delta_k g_k^T}{\delta_k^T g_k})^T+\frac{\delta_k g_k^T}{\delta_k^T g_k}$$

称为BFGS算法关于G_k的迭代公式。

Sherman-Morrison公式：假设A是n阶可逆矩阵，u，v是n维向量，且$A+uv^T$也是可逆矩阵，则

$$(A+uv^T)^{-1}=A^{-1}-\frac{A^{-1}uv^TA^{-1}}{1+v^TA^{-1}u}$$

记DFP关于G_k的迭代公式得到的G_{k+1}为G^{DFP}，由BFGS得到的记为G^{BFGS}，它们都满足拟牛顿公式，且其线性组合

$$G_{k+1}=\alpha G^{DFP}+(1-\alpha)G^{BFGS}$$

也满足拟牛顿条件，而且是正定是。其中$0\leq\alpha\leq 1$。这样得到的一类拟牛顿法称为Broyden类算法。

实际在使用的时候使用的是LBFGS算法（Limited-memory-BFGS）。有很成熟的开源实现，直接调用即可。

§ 1.4 矩阵分析（matrix analysis）

1.4.1 迹(Trace)和导数(matrix derivatives)

令$f:\mathbb{R}^{m\times n}\mapsto\mathbb{R}$表示将$m\times n$（$m$-by-$n$）矩阵映射为实数的函数。定义$f$对矩阵$\boldsymbol{A}$的导数

$$\nabla_{\boldsymbol{A}}f(\boldsymbol{A})=\begin{pmatrix}\frac{\partial f(\boldsymbol{A})}{\partial a_{11}} & \cdots & \frac{\partial f(\boldsymbol{A})}{\partial a_{1n}}\\ \vdots & \ddots & \vdots\\ \frac{\partial f(\boldsymbol{A})}{\partial a_{n1}} & \cdots & \frac{\partial f(\boldsymbol{A})}{\partial a_{nn}}\end{pmatrix}$$

矩阵的迹（trace）表示的是矩阵的对角元素的和，

$$tr\boldsymbol{A}=\sum_{i=1}^{n}a_{ii}$$

假设A，B，C，D均是方阵

$$trABCD=trDABC=trCDAB=trBCDA \tag{1.6}$$

循环将最右边矩阵放到最左边。假设a是实数

$$trA=trA^T \tag{1.7}$$

$$tr(A+B)=trA+trB \tag{1.8}$$

$$traA = atrA \tag{1.9}$$

下面的一些公式出自Andrew Ng的机器学习讲义，这里证明一下。

$$\nabla_A trAB = B^T \tag{1.10}$$

$$\nabla_{A^T} f(A) = (\nabla_A f(A))^T \tag{1.11}$$

$$\nabla_A trABA^TC = CAB + C^TAB^T \tag{1.12}$$

$$\nabla_A |A| = |A|(A^{-1})^T \tag{1.13}$$

证明　(1)　$(\nabla_A trAB)_{ij} = \frac{\partial trAB}{\partial a_{ij}} = \frac{\partial \sum_m \sum_k a_{mk}b_{km}}{\partial a_{ij}}$，只有当$m=i,k=j$时才有$a_{ij}$的系数，所以$(\nabla_A trAB)_{ij} = b_{ji}$，即证。

(2)　$(\nabla_{A^T} f(A))_{ij} = \frac{\partial f(A)}{\partial a_{ji}}$，即证。

(3)　$trABA^TC = \sum_m \sum_k \sum_t \sum_s a_{mk}b_{kt}a_{st}c_{sm}$，所以

$$(\nabla_A trABA^TC)_{ij} = \frac{\partial \sum_m \sum_k \sum_t \sum_s a_{mk}b_{kt}a_{st}c_{sm}}{\partial a_{ij}}$$

$$= \sum_m \sum_k \sum_t \sum_s \frac{\partial a_{mk}}{\partial a_{ij}} b_{kt}a_{st}c_{sm} + \sum_m \sum_k \sum_t \sum_s a_{mk}b_{kt}\frac{\partial a_{st}}{\partial a_{ij}}c_{sm}$$

左边，令$m=i,k=j$，右边，令$s=i,t=j$，

$$\begin{aligned}(\nabla_A trABA^TC)_{ij} &= \sum_t \sum_s b_{jt}a_{st}c_{si} + \sum_m \sum_k a_{mk}b_{kj}c_{im} \\ &= \sum_t \sum_s b_{jt}a_{st}c_{si} + \sum_m \sum_k c_{im}a_{mk}b_{kj} \\ &= (BA^TC)_{ji} + (CAB)_{ij} \\ &= (C^TAB^T + CAB)_{ij}\end{aligned}$$

§ 1.5 常用不等式

1.5.1 柯西不等式（Cauchy Inequality）

柯西不等式，又称柯西-施瓦茨不等式（Cauchy-Schwarz inequality）。对于一个内积空间

所有向量$\boldsymbol{x}$和$\boldsymbol{y}$，

$$|\langle \boldsymbol{x}, \boldsymbol{y} \rangle|^2 \leq \langle \boldsymbol{x}, \boldsymbol{x} \rangle \cdot \langle \boldsymbol{y}, \boldsymbol{y} \rangle$$

其中$\langle \cdot, \cdot \rangle$表示内积（点积），当且仅当$\boldsymbol{x}$与$\boldsymbol{y}$线性相关时等式成立。

对于欧几里得空间$\mathbb{R}^2$，

$$(\sum_{i=1}^{n} x_i y_i)^2 \leq (\sum_{i=1}^{n} x_i^2)(\sum_{i=1}^{n} y_i^2)$$

当且仅当$\frac{x_1}{y_1} = \frac{x_2}{y_2} = \cdots = \frac{x_n}{y_n}$时等式成立。

1.5.2 赫尔德不等式（Hölder Inequality）

赫尔德不等式揭示了L^p空间的相互关系。设S为测度空间，$1 \leq p, q \leq \inf$，且$\frac{1}{p} + \frac{1}{q} = 1$，若$f \in L^p(S)$，$g \in L^q(S)$，则$fg \in L^1(S)$，且

$$\| fg \|_1 \leq \| f \|_p \| g \|_q$$

写成序列或向量的形式

$$\sum_{i=1}^{n} |a_i b_i| \leq (\sum_{i=1}^{n} |a_i|^p)^{\frac{1}{p}} (\sum_{i=1}^{n} |b_i|^q)^{\frac{1}{q}}$$

1.5.3 闵可夫斯基不等式（Minkowski Inequality）

闵可夫斯基不等式表明L^p空间是一个赋范向量空间。设S是一个度量空间，$f, g \in L^p(S), 1 \leq p \leq \inf$，那么$f + g \in L^p(S)$，有

$$\| f + g \|_p \leq \| f \|_p + \| g \|_p$$

写成序列或向量的形式

$$(\sum_{k=1}^{n} |x_k + y_k|^p)^{\frac{1}{p}} \leq (\sum_{k=1}^{n} |x_k|^p)^{\frac{1}{p}} (\sum_{k=1}^{n} |y_k|^p)^{\frac{1}{p}}$$

第2章　回归分析

给定数据集

$$T = \{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \cdots, (x^{(m)}, y^{(m)})\},$$

其中，$x^{(i)} = (x_0, x_1, \cdots, x_n) \in \mathcal{X} = \mathbb{R}^{n+1}$，$y^{(i)} \in \mathcal{Y}$, $i = 0, 2, \cdots, m$，且$x_0 = 1$（表示截距，intercept term）。回归分析的任务是找出输入x与输出y之间的关系。

§ 2.1 线性回归

假设输入与输出之间满足的关系是线性的，$theta$称为参数（parameters）或者权重（weights）

$$y = h_\theta(x) = \theta^T x = \sum_{i=0}^{n} \theta_i x_i, \qquad \theta \in \mathbb{R}^{n+1}$$

对于这个模型，需要有一个损失函数（cost function）来表示其对训练数据的拟合程度

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

这个被称为最小二乘方效用函数（least-square cost function）。

则这个问题的求解可以表述为如下的无约束最优化问题

$$\min_\theta J(\theta)$$

2.1.1 直接求解

直接求$J(\theta)$对θ的极值。首先定义设计矩阵（design matrix）X

$$X = \begin{bmatrix} (x^{(1)})^T & (x^{(2)})^T & \cdots & (x^{(m)})^T \end{bmatrix}^T$$

$$Y = \begin{bmatrix} y^{(1)} & y^{(2)} & \cdots & y^{(m)} \end{bmatrix}^T$$

则有

$$X\theta - Y = \begin{bmatrix} h_\theta(x^{(1)}) - y^{(1)} & \cdots & h_\theta(x^{(m)}) - y^{(m)} \end{bmatrix}^T$$

考虑到$z^T z = \sum_i z_i^2$，有

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

使用公式$\nabla_A trABA^TC = CAB + C^TAB^T$，且由于$J(\theta)$只是个实数，所以

$$\begin{aligned} \nabla_\theta J(\theta) &= \nabla_\theta trJ(\theta) \\ &= X^TX\theta - X^TY \end{aligned}$$

令$\nabla_\theta J(\theta) = 0$得到

$$\theta = (X^TX)^{-1}X^TY$$

不过，这个公式用来直接计算θ不现实，因为矩阵求逆比较麻烦，同时可能会是数值不稳定的矩阵。

2.1.2 牛顿法（Newton's Method）

已知最优化问题

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

函数$J(\theta)$的Hessian矩阵是（注意$x^{(i)}$和θ都是列向量）

$$H(\theta) = \begin{bmatrix} \frac{\partial^2 J(\theta)}{\partial\theta_1\partial\theta_1} & \frac{\partial^2 J(\theta)}{\partial\theta_1\partial\theta_2} & \cdots & \frac{\partial^2 J(\theta)}{\partial\theta_1\partial\theta_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 J(\theta)}{\partial\theta_n\partial\theta_1} & \frac{\partial^2 J(\theta)}{\partial\theta_n\partial\theta_2} & \cdots & \frac{\partial^2 J(\theta)}{\partial\theta_n\partial\theta_n} \end{bmatrix} = \sum_{i=1}^{m} x^{(i)}(x^{(i)})^T = X^TX$$

又，$J(\theta) = \frac{1}{2}(X\theta - Y)^T(X\theta - Y)$，所以其梯度

$$\nabla_\theta J(\theta) = X^TX\theta - X^TY$$

代入牛顿法的迭代公式

$$\theta_{n+1} = \theta_n - (X^TX)^{-1}(X^TX\theta_n - X^TY) = (X^TX)^{-1}X^TY$$

2.1.3 批处理梯度下降法（batch gradient descent）

$J(\theta)$对θ的梯度

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

所以，更新公式为

$$\theta_{n+1} = \theta_n - \alpha\frac{\partial J(\theta)}{\partial \theta}$$

$$= \theta_n - \alpha\sum_{i=1}^{m}(h_{\theta_n}(x^{(i)}) - y^{(i)})x^{(i)}$$

这个公式更新时每次都需要全部的训练数据集，所以称之为批处理梯度下降法。当数据集比较大时，进行一次更新就比较耗费时间。

2.1.4 随机梯度下降法（stochastic gradient descent）

随机梯度下降法，也称增量梯度下降（incremental gradient descent），一次使用一条训练数据来更新参数。算法如下

$$\theta := \theta - \alpha(h_\theta(x^{(i)}) - y^{(i)})x^{(i)}, i = 1, 2, \cdots, m$$

随机梯度下降法更新的速度比梯度下降法快，但可能收敛不到最优值，不过通过调节学习率可以使得算法得到较好的解。而且随机梯度下降法可以用作在线学习的算法。

§ 2.2 局部加权线性回归（LWR）

线性回归的方法是参数学习算法（parametric learning algorithm），其参数的个数，即特征的个数是固定的，一旦算法学习完成，训练数据集就不再对参数产生影响。但是，当选取的参数过多时，可能存在过拟合问题，而当选取的参数过少时，存在欠拟合问题。局部加权线性回归（local weighted linear regression，LWR）是一种非参学习算法（non-parametric learning algorithm），其参数是随着预测点的不同而发生变化的，每有一个新的预测点，就需要整个训练数据集重新参与学习。所谓局部，是因为目标函数的逼近仅仅根据查询点附近的数据。所谓加权，是因为每个训练样例的贡献都是由它与查询点间的距离加权的。而回归是指数值逼近的方法。

线性回归的优化目标是

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

而LWR的优化目标是在上述公式上增加一个距离乘法项

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

一个相对标准的权重选择是

$$w^{(i)} = exp(-\frac{(x^{(i)} - x)^2}{2\tau^2})$$

其中，x是要预测的输入数据，τ称为带宽（bandwidth）。权重项使得离输入数据x越近的点影响越大。

非参数学习算法在局部预测能力上有时要比参数学习算法好，但是缺点是每次做预测都要重新学习，耗费时间空间。

第3章　逻辑斯谛回归

§ 3.1 二项逻辑斯谛回归

逻辑斯谛回归（logistic regression）是分类模型，跟线性回归模型有相同的输入，但是其输出是离散值。但是，之所以叫“回归”是因为它依然采用线性回归的算法来预测x与y之间的关系。也因此，需要限制模型中y值的输出，令

$$h_\theta(x) = g(\theta^T x) = \frac{1}{1 + e^{-\theta^T x}}$$

其中

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

称为逻辑斯谛函数（logistic function），或者S形曲线（sigmoid curve）。

为了推导方便，首先给出$g(z)$的导数

$$g'(z) = g(z)(1 - g(z))$$

设

$$P(y = 1|x; \theta) = h_\theta(x)$$

$$P(y = 0|x; \theta) * = 1 - h_\theta(x)$$

或者简写成

$$p(y|x; \theta) = (h_\theta(x))^y (1 - h_\theta(x))^{1-y}$$

其对数似然函数

$$l(\theta) = \log \prod_{i=1}^{m} p(y^{(i)}|x^{(i)}; \theta) = \sum_{i=1}^{m} (y^{(i)} \log h_\theta(x^{(i)}) + (1 - y^{(i)}) \log(1 - h_\theta(x^{(i)})))$$

$$= \sum_{i=1}^{m} (y^{(i)} h_\theta(x^{(i)}) + \log(1 - h_\theta(x^{(i)})))$$

其中，m是训练样本的个数。对θ求导，得

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

则对于单条训练样本的更新公式是

$$\theta := \theta + \alpha(y^{(i)} - h_\theta(x^{(i)}))x^{(i)}$$

注意，之所以是+号，是因为所求的最优化问题是似然函数的最大化，也即按照梯度的方向增加。

这个更新公式跟LMS（least mean squares）更新准则一样。也即，更新的幅度由学习率以及特征向量和误差的乘积决定。

§ 3.2 多项逻辑斯谛回归

逻辑斯谛回归其实是建立了逻辑斯谛概率模型之后依据似然函数最大化准备建立的回归模型，其学习算法一般是梯度下降或者牛顿法。多项逻辑斯谛回归的概率模型可以表述为：如果离散随机变量Y的取值集合是$\{1, 2, \cdots, K\}$，那么多项逻辑斯谛回归模型是

$$P(Y = k|x; \theta) = \frac{e^{\theta^T x}}{1 + \sum_{i=1}^{K-1} e^{\theta^T x}}, k = 1, 2, \cdots, K-1 \tag{3.1}$$

$$P(Y = K|x; \theta) = \frac{1}{1 + \sum_{i=1}^{K-1} e^{\theta^T x}}, k = K \tag{3.2}$$

其中，$x, \theta \in \mathbb{R}^{n+1}$.

第4章　朴素贝叶斯法

§ 4.1 贝叶斯公式

划分　设S为实验E的样本空间，$B_1, B_2, \cdots, B_n$为E的一组事件，若

(1)$B_i B_j = \emptyset, i \neq j, i, j = 1, 2, \cdots, n$

(2)$B_1 \cup B_2 \cup \cdots \cup B_n = S$

则称$B_1, B_2, \cdots, B_n$为样本空间S的一个**划分**。

全概率公式　设实验E的样本空间为S，A为E的事件，$B_1, B_2, \cdots, B_n$为S的一组划分，且$P(B_i) > 0, i = 1, 2, \cdots, n$，则

$$P(A) = \sum_{i=1}^{n} P(A|B_i)$$

称为**全概率公式**。

贝叶斯公式　设实验E的样本空间为S，A为E的事件，$B_1, B_2, \cdots, B_n$为S的一组划分，且$P(A) > 0, P(B_i) > 0, i = 1, 2, \cdots, n$，则

$$P(B_i|A) = \frac{P(A|B_i)P(B_i)}{P(A)} = \frac{P(A|B_i)P(B_i)}{\sum_{i=1}^{n} P(A|B_i)}$$

称为**贝叶斯公式**。

其中，$P(A), P(B_i)$分别称为先验概率（prior probability）。$P(A|B_i), P(B_i|A)$称为后验概率（posterior probability）。

§ 4.2 模型

朴素贝叶斯法（naive Bayes）之所以被称为朴素的，是因为其包含有一些朴素的假设：类条件独立假设。

假设D是训练元组及其分类标号的集合。特征向量表示为$X = \{x_1, x_2, \cdots, x_n\}$，其中$x_1, x_2, \cdots, x_n$为$n$个属性的取值。假设有$m$个类，$C_1, C_2, \cdots, C_m$。给定元组$X$，朴素贝叶斯分类法将$X$分给后验概率最大的类。即，$X$属于$C_i$类，当且仅当

$$P(C_i|X) > P(C_j|X), j = 1, 2, \cdots, m, i \neq j$$

则这个分类过程等价于最大化后验概率$P(C_i|X)$。由贝叶斯公式得

$$\max_i P(C_i|X) = \frac{P(X|C_i)P(C_i)}{P(X)}$$

其中，$P(X)$对所有分类都是一样的，$P(C_i)$如果做等概率假设，即$P(C_1) = P(C_2) = \cdots = P(C_m)$，则上式等价于

$$\max_i P(X|C_i)$$

否则，用训练样本里面每个分类出现的频次作为其先验概率估计值，即$P(C_i) = \frac{freq(C_i)}{|D|}$，则最优化问题等价于

$$\max_i P(X|C_i)P(C_i)$$

下面考虑$P(X|C_i)$的计算方法。为了简化计算量，可以做**类条件独立**的朴素假设，即属性之间不存在依赖关系，则

$$P(X|C_i) = \prod_{k=1}^{n} P(x_k|C_i) = P(x_1|C_i)P(x_2|C_i)\cdots P(x_n|C_i)$$

这样，求$P(X|C_i)$的问题就转化为求属性x_k在分类C_i中出现频次的问题了。此时，需要区分属性取值是离散的，还是连续的。

(1)如果属性k的取值是离散的，则有$P(x_k|C_i) = \frac{freq(<x_k, C_i>)}{freq(C_i)}$。

(2)如果属性k的取值是连续的，假定连续值属性的取值$x_k \backsim \mathcal{N}(\mu, \sigma)$，则定义

$$g(x, \mu, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

同时，有

$$P(x_k|C_i) = g(x_k, \mu_{C_i}, \sigma_{C_i})$$

μ_{C_i}和σ_{C_i}分别表示C_i类中属性k的均值和标准差。

第5章　决策树

第6章　最大熵

第7章　支持向量机

第8章　提升方法

第9章　EM方法

§ 9.1 Jensen不等式

第10章　隐马尔可夫模型

参考文献

[1] 李航著. 《统计学习方法》. 北京:清华大学出版社，2012，3

[2] Jiawei Han, Micheline Kamber, Jian Pei 著.范明, 孟小峰译. 《数据挖掘：概念与技术》. 机械工业出版社, 2012, 8

[3] Tom M. Mitchell 著.曾华军等译. 《机器学习》.机械工业出版社, 2003, 1

[4] Andrew Ng, 《Machine Learning》公开课讲义