MLAPP 读书笔记 - 08 逻辑回归(Logistic regression)

> A Chinese Notes of MLAPP,MLAPP 中文笔记项目
> https://zhuanlan.zhihu.com/python-kivy

记笔记的人：cycleuser

2018年6月20日15:05:14

8.1 概论

构建概率分类器有一种方法是建立形式为$p(y,x)$的联合模型,然后以x为条件,推$p(y|x)$.这叫生成方法(generative approach).另外一种办法是直接以$p(y|x)$的形式去拟合一个模型.这就叫做启发式方法(discriminative approach),本章就要讲这个方法.具体来说就是要假设有一些参数为线性的启发模型.我们会发现这样的模型拟合起来特别简单.在8.6,我们会对生成方法和启发方法进行对比,在本书后面的章节中,还会讲到非线性的启发模型和非参数化的启发模型.

8.2 模型选择

正如在本书1.4.6讲过的,逻辑回归对应的是下面这种二值化分类模型:
$p(y|x,w)=Ber(y|sigm(w^Tx))$(8.1)

图1.19(b)所示的就是一个一维例子.逻辑回归可以很容易扩展用于高维度输入特征向量.比如图8.1所示的就是对二维输入和不同的权重向量w的逻辑回归$p(y=1|x,w)=sigm(w^Tx)$.如果设置概率0.5的位置为阈值,就能得到一个决策边界,其范数(垂线)为w.

8.3 模型拟合

本节讲对逻辑回归模型进行参数估计的算法.

此处参考原书图8.1

8.3.1 最大似然估计(MLE)

逻辑回归的负对数似然函数为:

$$NLL(w) = -\sum_{i=1}^{N} \log[\mu_i^{I(y_i=1)} \times (1-\mu_i)^{I(y_i=0)}] \tag{8.2}$$

$$= -\sum_{i=1}^{N} \log[y_i \log \mu_i + (1-y_i)\log(1-\mu_i)] \tag{8.3}$$

这也叫交叉熵误差函数(cross-entropy error function)参考本书2.8.2.

这个公式还有另外一个写法,如下所示.设$\tilde{y}_i \in \{-1,+1\}$,而不是$y_i \in \{0,1\}$.另外还有$p(y=-1) = \frac{1}{1+\exp(-w^T x)}$和$p(y=1) = \frac{1}{1+\exp(+w^T x)}$.这样有:

$NLL(w) = \sum_{i=1}^{N} \log(1+\exp(-\tilde{y}_i w^T x_i))$(8.4)

和线性回归不一样的是,在逻辑回归里面,我们不再能以闭合形式写出最大似然估计(MLE).所以需要使用一个优化算法来计算出来.为了这个目的,就要推到梯度(gradient)和海森矩阵(Hessian).

此处参考原书图8.2

如练习8.3所示,很明显梯度和海森矩阵分别如下所示:

$$g = \frac{d}{dw} f(w) = \sum_i *\mu_i - y_i)x_i = X^T(\mu - y) \tag{8.5}$$

$$H = \frac{d}{dw} g(w)^T = \sum_i (\nabla_w \mu_i) x_i^T = \sum_i \mu_i(1-\mu_i) x_x x_i^T \tag{8.6}$$

$$= X^T S X \tag{8.7}$$

其中的$S \triangleq diag(\mu_i(1-\mu_i))$.通过练习8.3也可以证明海森矩阵H是正定的(positive definite).因此负对数似然函数NLL就是凸函数(convex)有唯一的全局最小值.接下来就说一下找到这个最小值的方法.

8.3.2 梯度下降(gradient descent)

无约束优化问题的最简单算法,可能就是梯度下降(gradient descent)了,也叫做最陡下降(Steepest descent).写作下面的形式:

$\theta_{k+1} = \theta_k - \eta_k g_k$(8.8)

其中的η_k是步长规模(step size)或者也叫学习率(learning rate).梯度下降法的主要问题就是:如何设置步长.这个问题还挺麻烦的.如果使用一个固定的学习率,但又太小了,那收敛就会很慢,但如果要弄太大了呢,又可能最终不能收敛.这个过程如图8.2所示,其中对下面的(凸函数)进行了投图:

$f(\theta) = 0.5(\theta_1^2 - \theta_2) + 0.5(\theta_1)^2$(8.9)

任意设置从(0,0)开始.在图8.2(a)中,使用了固定步长为$\eta = 0.1$;可见沿着低谷部位移动很慢.在图8.2(b)中,使用的是固定步长$\eta = 0.6$;很明显这样一来算法很快就跑偏了,根本就不能收敛了.

此处参考原书图8.3

所以就得像个靠谱的办法来选择步长,这样才能保证无论起点在哪里最终都能收敛到局部最优值.(这个性质叫做全局收敛性(global convergence),可千万别跟收敛到全局最优值弄混淆哈.)通过泰勒定理(Taylor's theorem),就得到了:

$$f(\theta + \eta d) \approx f(\theta) + \eta g^T d \text{(8.10)}$$

其中的d是下降方向.所以如果η足够小了,则$f(\theta + \eta d) < f(\theta)$,因为梯度会是负值的.不过我们并不希望步长太小,否则就要运算很久才能到达最小值了.所以要选一个能够最小化下面这个项的步长η:

$$\phi(\eta) = f(\theta_k + \eta d_k) \text{(8.11)}$$

这就叫线性最小化(line minimization)或者线性搜索(line search).有很多种方法来借这个一维优化问题,具体细节可以参考(Nocedal and Wright 2006).

图8.3(a)展示了上面那个简单问题中的线性搜索.不过我们会发现线性搜索得到的梯度下降路径会有一种扭折行为(zig-zag behavior).可以看到其中一次特定线性搜索要满足$\eta_k = \arg\min_{\eta > 0} \phi(\eta)$.优化的一个必要条件就是导数为零,即$\phi'(\eta) = 0$.通过链式规则(chain rule),$\phi'(\eta) = d^T g$,其中$g = f'(\theta + \eta d)$是最后一步的梯度.所以要么就有$g = 0$,意思就是已经找到了一个固定点(stationary point);要么就是$g \perp d$,意味着这一步所在点的位置上局部梯度和搜索方向相互垂直.因此连续起来方向就是正交的,如图8.3(b)所示,这就解释了搜索路径的扭折行为.

降低这种扭折效应的一种简单的启发式方法就是增加一个动量项(momentum term)$\theta_k - \theta_{k-1}$:

$$\theta_{k+1} = \theta_k - \eta_k g_k + \mu_k (\theta_k - \theta_{k-1}) \text{(8.12)}$$

上式中的$0 \leq \mu_k \leq 1$控制了动量项的重要程度.在优化领域中,这个方法叫做重球法(heavy ball method ,参考 Bertsekas 1999).

另外一种最小化扭折行为的方法是使用共轭梯度(conjugate gradients)(参考Nocedal and Wright 2006第五章,或者Golub and van Loan 1996,10.2).这个方法是选择形式为$f(\theta) = \theta^T A \theta$的二次形式(quadratic objectives),这是在解线性方程组的时候出现的.不过非线性的共轭梯度就不太受欢迎了.

8.3.3 牛顿法

算法8.1 最小化一个严格凸函数的牛顿法

1. 初始化一个θ_0;
2. 对于k=1,2,...等,一直到收敛为止,重复下面步骤:
3. 估算$g_k = \nabla f(\theta_k)$
4. 估算$H_k = \nabla^2 f(\theta_k)$
5. 对d_k求解$H_k d_k = -g_k$
6. 使用线性搜索来找到沿着d_k方向的步长η_k
7. $\theta_{K+1} = \theta_k + \eta_k d_k$

如果把空间曲率(curvature)比如海森矩阵(Hessian)考虑进去,可以推导出更快速的优化方法.这样方法就成了二阶优化方法了(second order optimization metods).如果不考虑曲率的那个就叫做牛顿法(Newton's algorithm).这是一个迭代算法(iterative algorithm),其中包含了下面形式的更新步骤:

$\theta_{k+1} = \theta_k - \eta_k H_k^{-1} g_k$(8.13)

完整的伪代码如本书算法2所示.

这个算法可以按照下面步骤推导.设构建一个二阶泰勒展开序列来在θ_k附近估计$f(\theta)$:

$f_{quad}(\theta) = f_k + g_k^T(\theta - \theta_k) + \frac{1}{2}(\theta - \theta_k)^T H_k(\theta - \theta_k)$(8.14)

重写成下面的形式

$f_{quad}(\theta) = \theta^T A\theta + b^T\theta + c$(8.15)

其中:

$A = \frac{1}{2}H_k, b = g_k - H_k\theta_k, c = f_k - g_k^T\theta_k + \frac{1}{23}\theta_k^T H_k\theta_k$(8.16)

f_{quad}最小值为止在:

$\theta = -\frac{1}{2}A^{-1}b = \theta_k - H_k^{-1}g_k$(8.17)

因此牛顿步长$d_k = -H_k^{-1}g_k$就可以用来加到θ_k上来最小化在θ_k附近对f的二阶近似.如图8.4(a)所示.

此处参考原书图8.4

在最简单的形式下,牛顿法需要海森矩阵H_k为正定矩阵,这保证了函数是严格凸函数.否则,目标函数非凸函数,那么海森矩阵H_k就可能不正定了,所以$d_k = -H_k^{-1}g_k$就可能不是一个下降方向了(如图8.4(b)所示).这种情况下,简单的办法就是逆转最陡下降方向,$d_k = -g_k$.列文伯格-马夸特算法(Levenberg Marquardt algorithm)是一种在牛顿步长和最陡下降步长之间这种的自适应方法.这种方法广泛用于解非线性最小二乘问题.一种替代方法就是:不去直接计算$d_k = -H_k^{-1}g_k$,可以使用共轭梯度来解关于d_k的线性方程组$H_k d_k = -g_k$.如果H_k不是正定矩阵,只要探测到了负曲率,

就可以简单地截断共轭梯度迭代,这样就叫做截断牛顿法(truncated Newton).

8.3.4 迭代重加权最小二乘法(Iteratively reweighted least squares,缩写为IRLS)

接下来试试将牛顿法用到二值化逻辑回归中求最大似然估计(MLE)上面.在这个模型中第$k+1$次迭代中牛顿法更新如下所示(设$\eta_k=1$,因此海森矩阵(Hessian)是确定的):

$$
\begin{aligned}
w_{k+1} &= w_k - H^{-1}g_k && (8.18)\\
&= w_k + (X^T S_k X)^{-1} X^T (y-\mu_k) && (8.19)\\
&= (X^T S_k X)^{-1}[(X^T S_k X)w_k + X^T(y-\mu_k)] && (8.20)\\
&= (X^T S_k X)^{-1} X^T [S_k X w_k + y - \mu_k] && (8.21)\\
&= (X^T S_k X)^{-1} X^T S_k z_k && (8.22)
\end{aligned}
$$

然后就可以定义工作响应函数(working response)如下所示:

$z_k \triangleq Xw_k + S_k^{-1}(y-\mu_k)$(8.23)

等式8.22就是一个加权最小二乘问题(weighted least squares problem),是要对下面的项最小化:

$\sum_{i=1}^N S_{ki}(z_{ki}-w^T x_i)^2$(8.24)

由于S_k是一个对角矩阵,所以可以把目标函数写成成分的形式(对每个$i=1:N$):

$z_{ki} = w_k^T x_i + \frac{y_i-\mu_{ki}}{\mu_{ki}(1-\mu_{ki})}$(8.25)

这个算法就叫做迭代重加权最小二乘法(Iteratively reweighted least squares,缩写为IRLS),因为每次迭代都解一次加权最小二乘法,其中的权重矩阵S_k在每次迭代都变化.伪代码参考本书配套算法10.

算法8.2 迭代重加权最小二乘法(IRLS)

1. $w=0_D$;
2. $w_0 = \log(\bar{y}/(1-\bar{y}))$
3. 重复下面步骤:
4. $\eta_i = w_0 + w^T x_i$
5. $\mu_i = sigm(\eta_i)$
6. $s_i = \mu_i(1-\mu_i)$
7. $z_i = \eta_i + \frac{y_i-\mu_i}{s_i}$
8. $S = diag(s_{1:N})$
9. $w = (X^T S X)^{-1} X^T S z$

10. 直到收敛

8.3.5 拟牛顿法

二阶优化算法的源头都是牛顿法,在本书8.3.3中讲到过.不过很不幸的是计算出来海森矩阵H的运算开销成本太高了.拟牛顿法(Quasi-Newton methods)就应运而生了,以迭代方式使用从每一步的梯度向量中学到的信息来构建对海森矩阵的估计.最常用的方法就是BFGS方法(这四个字母是发明这个算法的四个人的名字的首字母Broyden, Fletcher, Goldfarb, Shanno),这个方法是使用下面所示定义的$B_k \approx H_k$来对海森矩阵进行估计:

$$B_{k+1} = B_k + \frac{y_k y_k^T}{y_k^T s_k} - \frac{(B_k s_k)(B_k s_k)^T}{s_k^T B_k s_k} \tag{8.26}$$

$$s_k = \theta_k - \theta_{k-1} \tag{8.27}$$

$$y_k = g_k - g_{k-1} \tag{8.28}$$

这是对矩阵的二阶更新(rank-two update),这确保了矩阵保持正定(在每个步长的特定限制下).通常使用一个对角线估计来启动算法,即设$B_0 = I$.所以BFGS方法可以看做是对海森矩阵使用对角线加上低阶估计的方法.

另外BFGS方法也可以对海森矩阵的逆矩阵进行近似,通过迭代更新 $C_l \approx= H_k^{-1}$,如下所示:

$$C_{k+1} = (I - \frac{s_k s_k^T}{y_k^T s_k}) C_k (I - \frac{y_k s_k^T}{y_k^T s_k}) + \frac{s_k s_k^T}{y_k^T s_k} \tag{8.29}$$

存储海森矩阵(Hessian)需要消耗$O(D^2)$的存储空间,所有对于很大规模的问题,可以使用限制内存BFGS算法(limited memory BFGS),缩写为L-BFGS,其中的H_k或者H_k^{-1}都是用对角矩阵加上低阶矩阵来近似的.具体来说就是积(product)$H_k^{-1} g_k$可以通过一系列的s_k和y_k的内积来得到,只使用m个最近的s_k, y_k对,忽略掉更早的信息.这样存储上就只需要$O(mD)$规模的空间了.通常设置m大约在20左右,就足够有很好的性能表现了.更多相关信息参考(Nocedal and Wright 2006, p177).L-BFGS在机器学习领域中的无约束光滑优化问题中通常都是首选方法.

8.3.6 l_2规范化(regularization)

相比之下我们更倾向于选岭回归而不是线性回归,类似得到了,对于逻辑回归我们更应该选最大后验估计(MAP)而不是计算最大似然估计(MLE).实际上即便数据规模很大了,在分类背景下规范化还是很重要的.假设数据是线性可分的.这时候最大似然估计(MLE)就可以通过$||m|| \to \infty$来得到,对应的就是一个无穷陡峭的S形函数(sigmoid function)$I(w^T x > w_0)$,这也叫一个线性阈值单元(linear threshold unit).这将训练数据集赋予了最大规模概率质量.不过这样一来求解就很脆弱而且不好泛化.

所以就得和岭回归里面一样使用l_2规范化(regularization).这样一来新的目标函数.梯度函数.海森矩

阵就如下所示:

$$f'(w) = NLL(w) + \lambda w^T w \tag{8.30}$$
$$g'(w) = g(w) + \lambda w \tag{8.31}$$
$$H'(w) = H(w) + \lambda I \tag{8.32}$$

调整过之后的等式就更适合用于各种基于梯度的优化器了.

8.3.7 多类逻辑回归

接着就要讲多类逻辑回归(multinomial logistic regression)了,也叫作最大熵分类器(maximum entropy classifier).这种方法用到的模型形式为:

$$p(y = c|x, W) = \frac{\exp(w_c^T x)}{\sum_{c'=1}^{C} \exp(w_{c'}^T x)} \tag{8.33}$$

还有一个轻微变体,叫做条件逻辑模型(conditional logit model),是在对每个数据情况下一系列不同的类集合上进行了规范化;这个可以用于对用户在不同组合的项目集合之间进行的选择进行建模.

然后引入记号.设$\mu_{ic} = p(y_i = c|x_i W) = S(\eta_i)c$,其中的$\eta_i W^T x_i$是一个$C \times 1$ 向量.然后设$y_{ic} = I(y_i = c)$是对y_i的一种编码方式(one-of-C encoding);这样使得y_i成为二进制位向量(bit vector),当且仅当$y_i = c$的时候第c个位值为1.接下来是参考(Krishnapuram et al. 2005),设$w_C = 0$,这样能保证可辨识性(identifiability),然后定义$w = vec(W(:, 1 : C-1))$是一个$D \times (C-1)$维度的列向量(column vector).

这样就可以写出如下面形式的对数似然函数(log-likelihood):

$$l(W) = \log \prod_{i=1}^{N} \prod_{c=1}^{C} \mu_{ic}^{y_{ic}} = \sum_{i=1}^{N} \sum_{c=1}^{C} y_{ic} \log \mu_{ic} \tag{8.34}$$
$$= \sum_{i=1}^{N} [(\sum_{c=1}^{C} y_{ic} w_c^T x_i) - \log(\sum_{c'=1}^{C} \exp(w_{c'}^T x_i))] \tag{8.35}$$

加上负号就是负对数似然函数NLL了:
$f(w) = -l(w)$

然后就可以针对NLL计算器梯度和海森矩阵了.由于w是分块结构的(block-structured),记号会比较麻烦,但思路还是很简单的.可以定义一个$A \otimes B$表示在矩阵A和B之间的克罗内克积(kronecker product).如果A是一个$m \times n$矩阵,B是一个$p \times q$矩阵,那么$A \otimes B$就是一个$mp \times nq$矩阵:

$$A \otimes B = \begin{bmatrix} a_{11}B & \dots & a_{1n}B \\ \dots & \dots & \dots \\ a_{m1}B & \dots & a_{mn}B \end{bmatrix} \tag{8.37}$$

回到咱们刚刚的例子,很明显(练习8.4)梯度为:
$g(W) = \nabla f(w) = \sum_{i=1}^{N} (\mu_i - y_i) \otimes x_i$(8.38)

其中的$y_i = (I(y_i = 1), \ldots, I(y_i = C-1))$,
$\mu_i(W) = [p(y_i = 1|x_i, W), \ldots, p(y_i = C-1|x_i, W)]$这两个都是长度为$C-1$的列向量(column vectors).例如如果有特征维度D=3,类别数目C=3,这就成了:

$$g(W) = \sum_i \begin{pmatrix} (\mu_{i1} - y_{i1})x_{i1} \\ (\mu_{i1} - y_{i1})x_{i2} \\ (\mu_{i1} - y_{i1})x_{i3} \\ (\mu_{i2} - y_{i2})x_{i1} \\ (\mu_{i2} - y_{i2})x_{i2} \\ (\mu_{i2} - y_{i2})x_{i3} \end{pmatrix} \quad (8.39)$$

也就是说对于每个类c,第c列中权重的导数(the derivative for the weights)为:
$\nabla_{w_c} f(W) = \sum_i (\mu_i i - y_{ic}) x_i$(8.40)

这就和二值化逻辑回归的情况下形式一样了,名义上就是误差项乘以x_i.(其实这是指数族分布的一个通用特征,在本书9.3.2会讲到.)

(参考练习8.4)很容易发现海森矩阵是一个下面所示形式的$D(C-1) \times D(C-1)$分块矩阵:

$H(WW) = \nabla^2 f(w) = \sum_{i=1}^{N} (diag(\mu_i) - \mu_i \mu_i^T) \otimes (x_i x_i^T)$(8.41)

$$H(W) = \sum_i \begin{pmatrix} \mu_{i1} - \mu_{i1}^2 & -\mu_{i1}\mu_{i2} \\ -\mu_{i1}\mu_{i2} & \mu_{i2} - \mu_{i2}^2 \end{pmatrix} \otimes \begin{pmatrix} x_{i1}x_{i1} & x_{i1}x_{i2} & x_{i1}x_{i3} \\ x_{i2}x_{i1} & x_{i2}x_{i2} & x_{i2}x_{i3} \\ x_{i3}x_{i1} & x_{i3}x_{i2} & x_{i3}x_{i3} \end{pmatrix} \quad (8.42)$$

$$= \sum_i \begin{pmatrix} (\mu_{i1} - \mu_{i1}^2)X_i & -\mu_{i1}\mu_{i2}X_i \\ -\mu_{i1}\mu_{i2}X_i & (\mu_{i2} - \mu_{i2}^2)X_i \end{pmatrix} \quad (8.43)$$

其中的$X_i = x_i x_i^T$.也就是分块矩阵中块c, c'部分为:

$H_{c,c'}(W) = \sum_i \mu_{ic}(\delta_{c,c'} - \mu_{i,c'}) x_i x_i^T$(8.44)

这也是一个正定矩阵,所以有唯一最大似然估计(MLE).

接下来考虑最小化下面这个式子:
$f'(wW \triangleq -\log[(D|w) - \log p(W)$(8.45)

其中的$p(W) = \prod_c N(w_C|0, V_0)$.这样一来就得到了下面所示的新的目标函数/梯度函数/海森矩阵:

$$f'(w) = f(w) + \frac{1}{2}\sum_c w_c V_0^{-1} w_c \qquad (8.46)$$

$$g'(w) = g(W) + V_0^{-1}(\sum_c w_c) \qquad (8.47)$$

$$H'(w) = H(W) + I_C \otimes V_0^{-1} \qquad (8.48)$$

这样就可以传递给任意的基于梯度的优化器来找到最大后验估计(MAP)了.不过要注意这时候的海森矩阵规模是$O((CD) \times (CD))$,比二值化情况下要多C倍的行和列,所以这时候更适合使用限制内存的L-BFGS方法,而不适合用牛顿法.具体MATLAB代码参考本书配套PMTK3的logregFit.

8.4 贝叶斯逻辑回归

对逻辑回归模型,很自然的需求就是计算在参数上的完整后验分布$p(w|D)$.只要我们相对预测指定一个置信区间,这就很有用了.(另外在解决情景匪徒问题(contextual bandit problems)时候也很有用,参考本书5.7.3.1.)
然而很不幸,不像线性回归的时候了,这时候没办法实现这个目的,因为对于逻辑回归来说没有合适的共轭先验.本章这里讲的是一个简单近似;还有一些其他方法,比如马尔可夫链蒙特卡罗(Markov Chain Monte Carlo,缩写为MCMC,本书24.3.3.1),变分推导(variational inference,本书21.8.1.1),期望传播(expectation propagation,Kuss and Rasmussen 2005)等等.为了记号简单,这里还用二值逻辑回归为例.

8.4.1 拉普拉斯近似(Laplace approximation)

本节将对一个后验分布进行高斯近似.假如$\theta \in R^D$.设:
$p(\theta|D) = \frac{1}{Z}e^{-E(\theta)}$ (8.49)

其中的$E(\theta)$叫能量函数(energy function),等于未归一化对数后验(unnormalized log posterior)的负对数,即$E(\theta) = -\log p(\theta, D)$,$Z = p(D)$是归一化常数.然后进行关于众数$\theta^*$(对应最低能量状态)的泰勒级数展开,就得到了:

$E(\theta) \approx E(\theta^*) + (\theta - \theta^*)^T g + \frac{1}{2}(\theta - \theta^*)^T H(\theta - \theta^*)$ (8.50)

其中的g是梯度,H是在众数位置能量函数的海森矩阵:
$g \triangleq \nabla E(\theta)|_{\theta^*}, H \triangleq \frac{\partial^2 E(\theta)}{\partial\theta\partial\theta^T}|_{\theta^*}$ (8.51)

由于θ^*是众数,梯度项为零.因此:

$$\hat{p}(\theta|D) \approx \frac{1}{Z} e^{-E(\theta^*)} \exp[-\frac{1}{2}(\theta-\theta^*)^T H(\theta-\theta^*)] \tag{8.52}$$
$$= N(\theta|\theta^*, H^{-1}) \tag{8.53}$$
$$Z = p(D) \approx \int \hat{p}(\theta|D)d\theta = e^{-E(\theta^*)}(2\pi)^{D/2}|H|^{-\frac{1}{2}} \tag{8.54}$$

最后这一行是参考了多元高斯分布的归一化常数.

等式8.54就是对边缘似然函数的拉普拉斯近似.所以等式8.52有时候也叫做对后验的拉普拉斯近似.不过在统计学领域,拉普拉斯近似更多指的是一种复杂方法(具体细节参考Rue等,2009).高斯近似通常就足够近似了,因为后验分布随着样本规模增长就越来越高斯化,这个类似中心极限定理.(在物理学领域有一个类似的技术叫做鞍点近似(saddle point approximation).)

8.4.2 贝叶斯信息量(Bayesian information criterio,缩写为BIC)的推导

可以使用高斯近似来写出对数似然函数,去掉不相关的常数之后如下所示:
$\log p(D) \approx \log(p(D|\theta^*) + \log p(\theta^*) - \frac{1}{2}\log|H|$(8.55)

加在$\log(p(D|\theta^*)$后面的惩罚项(penalization terms)也叫作奥卡姆因子(Occam factor),是对模型复杂程度的量度.如果使用均匀先验,即$p(\theta) \propto 1$,就可以去掉第二项,然后把θ^*替换成最大似然估计(MLE)$\hat{\theta}$.

接下来看看对上式中第三项的估计.已知$H = \sum_{i=1}^{N} H_i$,其中的$H_i = \nabla\nabla \log p(D_i|\theta)$.然后用一个固定的矩阵$\hat{H}$来近似每个$H_i$.这样就得到了:

$\log|H| = \log|N\hat{H}| = \log(N^d|\hat{H}|) = D\log N + \log|\hat{H}|$(8.56)

其中$D = dim(\theta)$,并且假设H是满秩矩阵(full rank matrix).就可以去掉$\log|\hat{H}|$这一项了,因为这个独立于N,这样就可以被似然函数盖过去了.将所有条件结合起来,就得到了贝叶斯信息量分数(BIC score,参考本书5.3.2.4):
$\log p(D) \approx p(D|\hat{\theta}) - \frac{D}{2}\log N$(8.57)

8.4.3 逻辑回归的高斯近似

接下来对逻辑回归应用高斯近似.使用一个高斯先验,形式为$p(w) = N(w|0, V_0)$,正如在最大后验估计(MAP)里面一样.近似后验为:
$p(w|D) \approx N(w|\hat{w}, H^{-1})$(8.58)

上式中$\hat{w} = \arg\min_w \mathrm{E}(w), \mathrm{E}(w) = -(\log p(D|w) + \log p(w)), H = \nabla^2\mathrm{E}(w)|_{\hat{w}}$

以图8.5(a)中所示的线性可分二位数据为例.有很多个不同参数设置的线都能很好地区分开训练数据;在图中画了四条.图8.5(b)所示的是似然函数曲面,从中可以看到似然函数是无界的,随着向右上角的参数空间移动,似然函数沿着一个嵴线移动,$w_2/w_1 = 2.34$(通过对角线推测到的).这是因为将$||w||$推向无穷大可以是似然函数最大化.因为大的回归权重会让S形函数非常陡峭,就成了一个阶梯函数了.因此当数据线性可分的时候最大似然估计(MLE)不能很好定义.

为了规范表达这个问题,假设使用一个以原点为中心的模糊球形先验(vague spherical prior),$N(w|0, 100I)$.将这个球形先验乘以似然函数曲面就得到了一个严重偏斜的后验,如图8.5(c)所示.(这是因为似然函数截断了参数空间中与实验数据不符合的区域.)最大后验估计(MAP)如图中蓝色点所示.这就不像是最大似然估计(MLE)那样跑到无穷远了.

对这个后验的高斯近似如图8.5(d)所示.其中可以看到是一个对称分布,因此也不算是一个很好的估计.不过好歹众数还是正确的(通过构造得到的),而且至少表现了沿着西南东北方向比垂直这个方向上有更大不确定性的这个事实(这对应着稀疏线方向的不确定性).虽然这个高斯近似很粗糙,也肯定比用最大后验估计(MAP)得到的δ函数近似要好很多了.

8.4.4 近似后验预测

给定了后验,就可以计算置信区间,进行假设检验等等.就跟之前在7.6.3.3里面对线性回归案例所做的一样.不过在机器学习里面,更多兴趣还是在预测上面.后验预测分布的形式为:
$p(y|x, D) = \int p(y|x, w)p(w|D)dw$(8.59)

此处参考原书图8.5

很不幸这个积分可难算了.

最简单的近似就是插值估计(plug-in approximation),在二值化分类的情况下,形式为:
$p(y = 1|x, D) \approx p(y = 1|x, \mathrm{E}[w])$(8.60)

其中的$\mathrm{E}[w]$是后验均值.在这个语境下,$\mathrm{E}[w]$也叫作贝叶斯点(Bayes point).当然了,这种插值估计肯定是低估了不确定性了.所以接下来会说更好的近似方法.

此处参考原书图8.6

8.4.4.1 蒙特卡罗方法近似

更好的方法就是蒙特卡罗方法(Monte Carlo approximation),定义如下:
$p(y = 1|x, D) \approx \frac{1}{S}\sum_{s=1}^{S} sigm((w^s)^T x)$(8.61)

其中的$w^s \sim p(w|D)$是在后验中的取样.(这个方法很容易扩展到多类情况.)如果使用蒙特卡罗方法估计后验,就可以服用这些样本来进行预测.如果对后验使用高斯估计,就要用标准方法从高斯分布中取得独立样本.

此处查看原书图8.7

图8.6(b)展示了在我们的二维样例中从后验预测分布取的样本.图8.6(c)展示的是这些样本的均值.通过对多个预测取平均值,就可以发现决策边界的不确定性随着远离训练数据而散发开.所以虽然决策边界是线性的,后验预测密度还是非线性的.要注意后验均值的决策边界大概和每个类的距离都是一样远;这是对大便捷原则的贝叶斯模拟,在本书14.5.2.2会进一步讲到.

图8.7(a)所示的是一维下第一个例子.红色的点表示的是训练数据中后验预测分布评估出来的均值.竖着的蓝色线段表示的是对该后验预测的95%置信区间;蓝色的小星星是中位数位置.可见贝叶斯方法可以基于SAT分数来对一个学生能通过考试的概率的不确定性进行建模,而不是仅仅给出一个点估计.

8.4.4.2 Probit近似(适度输出)*

如果有一个对后验的高斯近似,$p(w|D) \approx N(w|m_N, V_N)$,就可以计算后验预测分布的确定性近似(deterministic approximation),至少在二值化情况下是可以的.步骤如下所示:

$$p(y=1|x,D) \approx \int sigm(w^T x)p(w|D)dw = \int sigma(a)N(a|\mu_a, \sigma_a^2)da \tag{8.62}$$

$$a \triangleq w^T x \tag{8.63}$$

$$\mu_a \triangleq \mathrm{E}[a] = m_N^T x \tag{8.64}$$

$$\sigma_a^2 \triangleq var[a] = \int p(a|D)[a^2 - \mathrm{E}[a^2]]da \tag{8.65}$$

$$= \int p(w|D)[(w^T x)^2 - (m_N^T x)^2]dw = x^T V_N x \tag{8.66}$$

然后就能发现需要去评估一个对应高斯分布的S形函数(sigmoid function)的期望.可以利用S形函数类似概率函数(probit function)的性质来进行近似,通过标准正态分布的累计密度函数(cdf):

$\Phi(a) \triangleq \int_{-\infty}^{a} N(x|0,1)dx$(8.67)

图8.7(b)所示的就是s形函数和概率函数.图中的坐标轴做了缩放,使得$sigm(a)$在原点位置附近有和$\Phi(\lambda a)$有类似的范围,其中$\lambda^2 = \pi/8$.

利用概率函数(probit function)的优势就是可以用一个高斯分布以解析形式卷积(convolve)出来:

$\int \Phi(\lambda a)N(a|\mu, \sigma^2)da = \Phi(\frac{a}{(\lambda^{-2}+\sigma^2)^{\frac{1}{2}}})$(8.68)

然后再等号两边都使用 $sigm(a) \approx \Phi(\lambda a)$ 来插值近似,就得到了:

$$\int sigm(a)N(a|\mu,\sigma^2)da \approx sigm(k(\sigma^2)\mu) \tag{8.69}$$

$$k(\sigma^2) \triangleq (1+\pi\sigma^2/8)^{-\frac{1}{2}} \tag{8.70}$$

将上面的近似用到逻辑回归模型中,就得到了下面的表达式(最初引用自(Spiegelhalter and Lauritzen 1990)):

$$p(y=1|x,D) \approx sigm(k(\sigma_a^2)\mu_a) \tag{8.71}$$

图8.6(d)所示表面这个近似能得到和蒙特卡洛近似相似的结果.

使用等式8.71这种近似,也叫作调节输出(moderated output),因为这比起插值估计不那么极端.要理解这一点,要注意到$0 \leq k(\sigma^2) \leq 1$,因此就有:

$$sigm(k(\sigma^2)\mu) \leq sigma(\mu) = p(y=1|x,\hat{w}) \tag{8.72}$$

上式中的不等号当$\mu \neq 0$的时候严格成立.如果$\mu > 0$,就有$p(y=1|x,\hat{w}) > 0.5$,但调节的预测(moderated prediction)总是在0.5附近,所以不太可信(less confident).不过,决策边界存在于$p(y=1|x,D) = sigm(k(\sigma^2)\mu) = 0.5$的位置,也就意味着$\mu = \hat{w}^T x = 0$.因此调节预测的决策边界和差值近似是一样的.所以两种方法的误分类率是一样的,但对数似然函数是不一样的.(要注意在多类情况下,后验协方差给出的结果和插值方法是不一样的,参考本书练习3.10.3以及(Rasmussen
and Williams 2006)).

8.4.5 残差分析(异常值检测)*

检验数据中的异常值(outlier)有时候很有用.这个过程就叫残差分析(residual analysis)或者案例分析(case analysis).在回归情况下,这个可以通过计算残差来得到:$r_i = y_i - \hat{y}_i$,其中的$\hat{y}_i = \hat{w}^T x_i$.如果模型假设正确无误,这些值应该遵循一个正态分布$N(0,\sigma^2)$.可以投图(qq-plot)评定,其中两个坐标轴的值分别是高斯分布的N个理论值(theoretical quantiles)和r_i的经验值(empirical quantiles).偏离这条直线的点就是潜在的异常值.

基于残差的简单方法不适用于二进制数据,因为依靠测试统计的渐进正态(asymptotic normality).不过用贝叶斯方法,就能定义异常值了,可以定义为$p(y_i|\hat{y}_i)$小的点,一般使用$\hat{y}_i = sigm(\hat{w}^T x)$.要注意这里的$\hat{w}$是从全部数据估计出来的.在预测$y_i$时更好的方法是从对w的估计排除掉$(x_i,y_i)$.也就是定义异常值为在交叉验证后验预测分布下有低概率的点,定义形式为:

$$p(y_i|x_i,x_{-i},y_{-i}) = \int p(y_i|x_i,w)\prod_{i' \neq i} p(y_{i'}|x_{i'},w)p(w)dw \tag{8.73}$$

这就可以通过抽样方法来有效近似(Gelfand 1996).更多关于逻辑回归模型中残差分析的相关内容可以参考(Johnson and Albert 1999, Sec 3.4).

8.5 在线学习(Online learning)和随机优化(stochastic optimization)

传统机器学习都是线下的,也就意味着是有一批量的数据,然后优化一个下面形式的等式:

$$f(\theta) = \frac{1}{N}\sum_{i=1}^{N} f(\theta, z_i) \quad (8.74)$$

其中如果有$z_i = (x_i, y_i)$是监督学习情况(supervised case),或者只有x_i就对应着无监督学习的情况,而$f(\theta, z_i)$这个函数是某种损失函数.比如可以使用下面这样的损失函数:

$$f(\theta, z_i) = L(y_i, h(x_i, \theta)) \quad (8.75)$$

其中的$h(x_i, \theta)$是预测函数,而$L(y, \hat{y})$是某种其他的损失函数,比如可以使平方误差或者胡博损失函数(Huber loss).在频率论统计学方法中,平均损失函数也叫作风险(参考本书6.3),所以对应地就将这个方法整体叫做经验风险最小化(empirical risk minimization,缩写为ERM),参考本书6.5.

可是如果有一系列的流数据(streaming data)不停出现,就需要进行在线学习(online learning),也就是要随着每次有新数据来到而更新估计,而不是等到尽头,因为可能永无止境.另外有时候虽然数据是成批的一整个数据,也可能会因为太大没办法全部放进内存等原因也需要使用在线学习.接下来就要讲这类校本化的学习方法.

8.5.1 在线学习和遗憾最小化(regret minimization)

假如在每一步中,客观世界都提供了一个样本z_k,而学习者必须使用一个参数估计θ_k对此进行响应.在理论机器学习社区中,在线学习关注的目标是遗憾值(regret),定义为相对于使用单个固定参数值时候能得到的最好结果所得到的平均损失:

$$regret_k \triangleq \frac{1}{k}\sum_{t=1}^{k} f(\theta_t, z_t) - \min_{\theta^* \in \Theta} \frac{1}{k}\sum_{t=1}^{k} f(\theta_*, z_t) \quad (8.77)$$

比如我们要调查股票市场.设θ_j是我们在股票j上面投资的规模,而z_j表示这个股票带来的回报.这样损失函数就是$f(\theta, z) = -\theta^T z$.遗憾值(regret)就是我们通过每次交易而得到的效果,而不只是依据什么神秘预言来选择买那个股票然后购买和持有的策略.

在线学习的简单算法是在线梯度下降法(online gradient descent (Zinkevich 2003)),步骤如下:在每次第k步,使用下列表达式更新参数:

$$\theta_{k+1} = \roj_{\Theta}(\theta_k - \eta_k g_k) \quad (8.78)$$

其中的$proj_v(v) = \arg\min_{w \in V} ||w - v||_2$是向量v在空间V上的投影,$g_k = \nabla f(\theta_k, z_k)$是梯度项,而$\eta_k$是补偿规模.(只有当参数必须要约束在某个$R^D$的子集内的时候才需要使用投影这个步骤.更多细节参考本书13.4.3.)接下来要看看这个让遗憾最小化的方法和更传统的关注对象之间的关系,比如最大似然估计(MLE).

当然遗憾最小化也由很多其他方法,这就超出这本书的覆盖范围了,更多细节可以参考Cesa-Bianchi and Lugosi (2006).

8.5.2 随机优化和风险最小化

接下来我们要尝试的不是让过去步骤的遗憾最小化,而是希望未来损失最小化,这在很多(频率论)统计学习理论中更长久.也就是要最小化:

$$f(\theta = \mathrm{E}[f(\theta, z)] \tag{8.79}$$

其中这个期望是对未来数据上取的.优化这种某些变量是随机变量的函数的过程就叫做随机优化(Stochastic optimization).
假如要从一个分不中得到一系列有限的抽样样本.一个方法就是优化8.79里面的期望值,在每一步应用等式8.78进行更新.这就叫做随机梯度下降法(stochastic gradient descent,缩写为SGD,出自Nemirovski and Yudin 1978).通常我们都想要一个简单的参数估计,可以用下面的进行平均:

$$\bar{\theta}_k = \frac{1}{k} \sum_{t=1}^{k} \theta_t \tag{8.80}$$

这就叫Polyak-Ruppert 平均,可以递归使用,如下所示:

$$\bar{\theta}_k = \bar{\theta}_{k-1} - \frac{1}{k}(\nabla \theta_{k-1} - \theta_k) \tag{8.81}$$
更多细节参考(Spall 2003; Kushner and Yin 2003).

8.5.2.1 设定步长规模

接下来要讨论的是要保证随机梯度下降(SGD)收敛所需要的学习速率(learning rate)的充分条件(sufficient conditions).这也叫做Robbins-Monro条件:

$$\sum_{k=1}^{\infty} \eta_k = \infty, \sum_{k=1}^{\infty} \eta_k^2 = \infty \tag{8.82}$$

η_k在时间上的取值集合也叫作学习速率列表(learning rate schedule).可以用很多公式,比如$\eta_k = 1/k$,或者可以用下面这个(Bottou 1998; Bach and Moulines 2011):
$\eta_k = (\tau_0 + k)^{-k}$(8.83)

上面的$\tau_0 \geq 0$减慢了算法的早期迭代,而$k \in (0.5, 1]$控制了旧值被遗忘的速率.

随机优化的一个主要缺陷就是需要去调整这些个参数.一个简单的启发式办法(Bottou 2007)如下所示:存储数据的一个初始子集,然后对这个自己适用一系列不同的η值;然后选择能使得目标对象降低最快的,再将其用于其他的全部数据上.要注意这可能会导致不收敛,不过当算法的性能提升达到某个设定好的位置(hold-out set plateaus)的时候可以终止(这也叫早期停止,early stopping)

8.5.2.2 分参数步长规模(Per-parameter step sizes)

随机梯度下降法(SGD)的一个问题就是对于不同的参数都用同样的补偿规模.接下来要简单介绍一种新方法,自适应梯度下降法(adaptive gradient),缩写为adagrad,出自(Duchi et al. 2010),这个方法的思路类似使用对角海森矩阵近似(diagonal Hessian approximation).(类似方法也可以参考Schaul et al. 2012).具体来说就是如果$\theta_i(k)$是第k次的参数i,而$g_i(k)$是对应的梯度,就可以用下面的方式进行更新:

$$\theta_i(k+1) = \theta_i(k) - \eta \frac{g_i(k)}{\tau_o + \sqrt{s_i(k)}} \text{(8.84)}$$

其中对角步长规模向量(diagonal step size vector)是梯度向量的平方(gradient vector squared),加上整个时间上的所有补偿.这也可以使用下面的形式来递归更新:

$$s_i(k) = s_i(k-1) + g_i(k)^2 \text{(8.85)}$$

这个结果就是一个分参数步长规模,可以适应损失函数的曲率(curvature).这个方法最开始推导出来是为了遗憾最小化的情形,不过还可以有很多其他用途.

8.5.2.3 随机梯度下降(SGD)和批量学习(batch learning)的对比

如果没有一个无限的数据流,可以去模拟一个,只要随机从训练集中取样数据点就可以了.本质上是将等式8.74作为对经验分布的期望来优化。

算法8.3 随机梯度下降法

1. 初始化θ, η;
2. 重复下面步骤直至收敛:
3. 随机交流数据(permute data)
4. for $i = 1:N$ 重复下面操作:
5. $g = \nabla f(\theta, z_i)$
6. $\theta \leftarrow proj_\Theta(\theta - \eta g)$
7. 更新η

理论上应该有替代取样,不过实际上通常都是无替代随机交流数据和样本效果更好,然后后面都重复这样操作.每次对全部数据集进行一次抽样就叫做一代(epoch).伪代码参考本书算法8.

在线下学习的情况下,更好的方法是以B份数据来进行小批量(mini-batch)梯度计算.如果B=1,这就是标准的随机梯度下降发,如果B=N,这就成了标准的最陡下降(Steepest descent).一般都设置$B \sim 100$.

虽然随机梯度下降法(SGD)是很简单的一阶方法(first-order method),但用于一些问题的时候效果出奇地好,尤其是数据规模很大的情况(Bottou 2007).直观理解起来,原因可能是只要看过几个样本之后就能对梯度进行很好的估计了.而使用很大规模的数据来仔细计算精准的梯度就很可能是浪费

时间了,因为算法反正也要在下一步重新计算梯度.为了充分利用计算时间,最好是使用一个有噪音的估计,然后沿着参数空间快速移动.一个极端的例子,比如要对每个样本都重复一份来复制整个训练样本集.这样批量方法就要花费两倍时间,而在线学习方法就几乎不会被影响,因为梯度方向其实没有变化(双倍数据规模知识改变了梯度烈度(magnitude),但并不相关,因为梯度反正也要被步长规模所缩放).

除了能加速之外,随机梯度下降法(SGD)还不太容易在浅局部最小值部位卡住,因为通常增加了一定规模的噪音.因此这种方法在机器学习社区中很流行于拟合有非凸函数对象的模型,比如神经网络(neural networks,本书16.5)和深层信念网络(deep belief networks本书28.1)等等.

8.5.3 最小均方算法(LMS algorithm)

举个随机梯度下降法(SGD)的例子,假设要考虑去计算在线学习中线性回归的最大似然估计(MLE).在等式7.14中已经推导出了批量梯度(batch gradient).在第k次迭代的在线梯度为:

$g_k = x_i(\theta_k^T x_i - y_i)$(8.86)

此处查看原书图8.8

其中的$i = i(k)$是第k次迭代时候使用的训练样本.如果数据集是流式的,就用$i(k) = k$;为了表达简单,以后就这么用了.等式8.86很好理解:特征向量x_k乘以预测$\hat{y}_k = \theta_k^T x_k$和真实的响应变量$y_k$的差距作为权重.;因此梯度函数就像可以当做是一个误差信号了.

在计算了梯度之后,可以沿着梯度进行下一步长,如下所示:
$\theta_{k+1} = \theta_k - \eta_k(\hat{y}_k - y_k)x_k$(8.87)

(这里就没必要进行投影的步骤了,因为这是一个非约束的优化问题,unconstrained optimization problem.)这个算法也叫作最小均方算法(least mean squares,缩写为LMS),也被称作δ规则(delta rule),或者也叫做Widrow-Hoff 规则(Widrow-Hoff rule).

图8.8所示就是对图7.2当中的数据应用这个算法的结果.启动点为$\theta = (-0.5, 2)$,经过大约26次迭代而收敛(这里的收敛指的是$||\theta_k - \theta_{k-1}||_2^2$小于阈值$10^{-2}$).

要注意最小均方算法(LMS)可能需要很多次遍历整个数据才能找到最优解.对比之下,使用递归最小均方算法,基于卡尔曼过滤器(Kalman filter)使用二阶信息(second-order information),只要单次遍历数据就能找到最优解了(参考本书18.2.3).也可以参考练习7.7.

8.5.4 感知器算法(perceptron algorithm)

接下来考虑如何对在线情况下的二值化逻辑回归模型(binary logistic regression model)进行拟合.批量梯度(batch gradient)如等式8.5所示.在线情况下呢,加权重的更新的形式简单如下:

$$\theta_k = \theta_{k-1} - \eta_k g_i = \theta_{k-1} - \eta_k(\mu_i - y_i)x_i \text{(8.88)}$$

其中$\mu_i = p(y_i = 1|x_i, \theta_k) = \mathrm{E}[y_i|x_i, \theta_k]$.可见这和最小均方算法(LMS)形式一模一样.实际上这个性质对于多有的通用线性模型都成立(参考本书9.3).

然后对这个算法进行近似.设:
$$\hat{y}_i = \arg\max_{y \in \{0,1\}} p(y|x_i, \theta) \text{(8.89)}$$

代表了最大概率类标签.然后替代$\mu_i = p(y = 1|x_i, \theta) = sigm(\theta^T x_i)$中的剃度表达式为$\hat{y}_i$.这样就得到了近似的梯度:

$$g_i \approx (\hat{y}_i - y_i)x_i \text{(8.90)}$$

如果我们假设$y \in \{-1, +1\}$,而不是$y \in \{0, 1\}$,那么在代数计算上就能更简单了.这时候我们的预测就成了:
$$\hat{y}_i = sign(\theta^T x_i) \text{(8.91)}$$

然后如果$\hat{y}_i y_i = -1$,就分类错误了,而如果$\hat{y}_i y_i = +1$则表示猜对了分类标签.

在每一步,都要通过加上梯度来更新权重向量.关键的观察项目在于,如果预测正确,那么$\hat{y}_i = y_i$,所以(近似)梯度就是零了,也就不用去更改权重向量了.可是如果x_i是误分类的(misclassified),就要按照下面的步骤更新权重向量了:如果$\hat{y}_i = 1$而$y_i = -1$,那么负梯度就是$-(\hat{y}_i - y_i)x_i = -2x_i$;如果反过来$\hat{y}_i = -1$而$y_i = 1$,那么负梯度就是$-(\hat{y}_i - y_i)x_i = 2x_i$.上面这个因数2可以吸收进学习速率(learning rate)η里面,之写成更新的形式,在误分类的情况下为:

$$\theta_k = \theat_{k-1} + \eta_k y_i x_i \text{(8.92)}$$

由于只有权重的符号是重要的,而大小并不重要,所以就可以设$\eta_k = 1$.伪代码参考本书的算法11.上面这个算法也就叫做感知器算法(perceptron algorithm),出自(Rosenblatt 1958),在给的数据是线性可分(linearly separable)的情况下就会收敛,即存在参数θ使得在训练集上的预测$sign(\theta^T x)$会达到零误差(0 error).不过如果暑假不是线性可分的,这个算法就不收敛了,甚至可能虽然收敛也要花费很长时间才行.训练逻辑回归模型有很多更好的方法,比如使用适当的随机梯度下降(SGD)而不使用梯度近似,或者迭代重加权最小二乘法(IRLS,参考本书8.3.4).不过感知器算法还是有很重要历史地位的:这可以算是有史以来被推导出的第一个机器学习算法(Frank Rosenblatt 1957),甚至还被用模拟硬件进行了实现.另外,这个算法也可以用于计算边缘分布$p(y_i|x, \theta)$比计算最大后验分布输出(MAP output)$\arg\max_y p(y|x, \theta)$在运算上更昂贵的模型拟合;这还引出了一些结构输出(structured-output)分类问题.具体细节参考本书19.7.

算法8.4 感知器算法

1. 输入:线性可分数据集$x_i \in R^D, y_i \in \{-1. + 1\}, for i = 1 : N$
2. 初始化θ_0;

3. $k \leftarrow 0$
4. 重复下面步骤直至收敛:
5. $k \leftarrow k+1$
6. $i \leftarrow k mod N$
7. 如果$\hat{y}_i \neq y_i$:
8. $\theta_{k+1} \leftarrow \theta_k + y_i x_i$
9. 否则:
10. 无操作

8.5.5 贝叶斯视角

在线学习的另外一个方法就是从贝叶斯视角实现的.这个概念特别简单:就是递归应用贝叶斯规则:
$p(\theta|D_{1:k}) \propto p(D_k|\theta)p(\theta|d_{1:k-1})$(8.93)

这有个很明显的优势,因为返回的是一个后验,而不是一个点估计.这也允许超参数(hyper-parameters)的在线自适应(online adaptation),这是非常重要的,因为在线学习没办法使用交叉验证.最后的一个不那么明显的优点就是速度可以比随机梯度下降法(SGD)更快.具体原因在于,通过对每个参数加上其均值的后验方差建模,可以有效对每个参数赋予不同的学习速率(de Freitas et al. 2000),这是对空间曲率(curvature)建模的一种简单方法.这些房差通过概率论的常见规则就可以实现自适应(adapted).对比之下,使用二阶(second-order)优化方法来解决在线学习的问题可能就要麻烦多了(更多细节参考Schraudolph et al.2007; Sunehag et al. 2009; Bordes et al. 2009, 2010).

本书18.2.3是一个简单的样例,展示了如何使用卡尔曼过滤器(Kalman filter)来在线拟合一个线性回归模型.和最小均方算法(LMS)不同,这个方法遍历数据一次就可以收敛到最优解(离线).Ting et al.2010 讲了一种扩展方法,以在线的形式来学习一个健壮的非线性回归模型(robust non-linear regression model).对于通用线性模型(GLM)的情况,可以使用一个假设密度过滤器(assumed density filter,参考本书18.5.3),其中使用一个对角协方差的高斯分布来对后验近似;方差项目就可以用作分参数步长规模(per-parameter step-size).更多细节参考本书18.5.3.2.另外一种方法是使用粒子过滤器(particle filtering,本书23.5),这个方法被(Andrieu et al. 2000)用于核化的(kernelized)线性/逻辑回归模型的序列学习(sequentially learning).

8.6 生成分类器和判别分类器(Generative vs discriminative classifiers)

在本书4.2.2,已经降到了类标签的后验分布使用高斯判别分析(Gaussian discriminative analysis,缩写为GDA)之后形式和逻辑回归一模一样,都是$p(y=1|x) = sigm(w^T x)$.因此这个决策边界在两种情况下都是x的线性函数.不过要注意很多生成模型可能会给出一个逻辑回归后验,比如,每

个类条件密度都是泊松分布(Poisson)$p(x|y=c)=Poi(x|\lambda_c)$.所以使用GDA所做的假设要比使用逻辑回归的假设更强.

这些模型更进一步的区别就是训练方式.当你和一个判别模型的时候,一般是最大化条件对数似然函数$\sum_{i=1}^{N}\log p(y_i|x_i,\theta)$,而当拟合一个生成模型的时候,通常要最大化联合对数似然函数$\sum_{i=1}^{N}\log p(y_i,x_i|\theta)$.很明显,这就能导致不同结果(参考练习4.20).

当使用GDA做出的高斯假设正确无误的时候,生成模型需要的训练样本数据集要比逻辑回归更少,就能达到确定水平的性能,可是如果高斯假设不正确,逻辑回归效果就更好了(Ng and Jordan 2002).这是因为判别模型不需要对特征分布进行建模.这也如图8.10所示.可见其中类条件密度还是相当复杂的;具体来说就是$p(x|y=1)$是一个多项分布,就可能很难估计.不过类后验$p(y=c|x)$是一个简单的s形函数,中间部位就是阈值0.55.这就说明一般来说判别模型可能更精准,因为他们的工作某种程度上来说更简单.不过精确度并不是我们挑选方法时候的唯一重要因素.接下来就要说一下两种方法的优点和不足.

8.6.1 不同方法的各自优劣

- 容易拟合?我们已经看到了,拟合生成分类器通常都很简单.比如本书3.5.1.1和4.2.4,分别对朴素贝叶斯模型和线性判别分析模型(LDA)进行了拟合,只需要简单的计数和取平均.相比之下,逻辑回归就需要解一个凸优化问题了(具体细节比如本书8.3.4),这就要慢多了.
- 分开拟合各类?在生成分类器中,我们队每个类条件密度分开估计其参数,所以不需要再增加更多类的时候重新训练模型.对比之下在判别模型中,所有参数都参与互动,所以添加新类的时候必须要重新训练.(如果我们训练一个生成模型来最大化一个判别对象也是这样的情况,参考Salojarvi et al. 2005.)
- 处理缺失数据更容易?有时候一些输入(x的成分)可能并没有观察到.也就是缺失了.在生成分类器里面,有一种简单方法来应对这种情况,稍后本书8.6.2就会讲到.不过在判别分类器里面,原则上这种问题就无解了,因为模型假设x是永远可用的要能用来做条件的(当然其实也有一些启发式方法来应对,参考Marlin 2008).
- 能处理无标签训练数据?半监督学习(semi-supervised learning)是很有意思的,其中使用了一些未标签数据来解决一个监督任务.这对于生成模型来说很容易(参考(Lasserre et al. 2006; Liang et al. 2007)),但对于判别模型就难多了.
- 输入输出是否对称?对于生成模型,可以反着用,然后通过给定的输出来推测可能的输入,只要计算$p(x|y)$就可以了.这对于判别模型来说就是不可能了.因为生成模型定义了一个在x和y上的联合分布,因此输入输出就是对称的.
- 能否特征预处理?判别方法的一个大优势就是允许我们事先以任何方式来对数据进行预处理,比如可以把x替换成$\phi(x)$,可以是某种基函数扩展,比如图8.9所示.不过要是相对这种预处理数据定义生成模型就可能很难了,因为新的特征就会以复杂方式相互关联.
- 能否有校正良好的概率?有点生成模型,比如朴素贝叶斯,有很强的独立假设,而这些假设很可能是无效的.这就可能导致非常极端的后验类概率,比如很靠近0或者1.判别模型比如逻辑回归

等等,就通常在概率估计上有更好的校正.

上面就是对于两种模型的比较对比.这两种模型最好都掌握一些,放进你的工具箱.表格8.1所示是本书讲的分类和回归技术的总结.

此处参考原书图8.9
此处参考原书图8.10

模型名称	分类/回归	生成/判别	参数化/非参数化	章节
判别分析 Discriminant analysis	分类	生成	参数化	4.2.2,4.2.4
朴素贝叶斯分类器Naive Bayes classifier	分类	生成	参数化	3.5,3.5.1.2
树增广朴素贝叶斯分类器Tree-augmented Naive Bayes classifier	分类	生成	参数	10.2.1
线性回归 Linear regression	回归	判别	参数化	1.4.5,7.3,7.6
逻辑回归 Logistic regression	分类	判别	参数化	1.4.6,8.3.4,8.4.3,21.8.1.1
稀疏线性/逻辑回归 Sparse linear/ logistic regression	结合	判别	参数化	13
专家混合 Mixture of experts	结合	判别	参数化	11.2.4
多层感知器/神经网络 Multilayer perceptron (MLP)/ Neural network	结合	判别	参数化	16
条件随机域 Conditional random fiel				

d (CRF)	分类	判别	参数化	19.6
K最近邻分类器 K nearest neighbor classifier	分类	生成	非参数化	1.4.2,14.7.3
无穷混合判别分析 (Infinite) Mixture Discriminant analysis	分类	生成	非参数化	14.7.3
分类和回归树 Classification and regression trees (CART)	结合	判别	非参数化	16.2
增强模型 Boosted model	结合	判别	非参数化	16.4
稀疏核化线性/逻辑回归 Sparse kernelized lin/logreg (SKLR)	结合	判别	非参数化	14.3.2
相关向量机 Relevance vector machine (RVM)	结合	判别	非参数化	14.3.2
支持向量机 Support vector machine (SVM)	结合	判别	非参数化	14.5
高斯过程 Gaussian processes (GP)	结合	判别	非参数化	15
平滑样条 Smoothing splines	回归	判别	非参数化	15.4.6

表 8.1 本书所讲到的分类/回归模型的列表.此表格也可以从这个链接查看对应的PMTK代码.任何的生成概率模型(比如隐形马尔科夫链HMM,玻尔兹曼机器Boltzmann machines,贝叶斯网络Bayesian networks等等)都可以转化成用作类条件密度(class conditional density)的分类器.

8.6.2 处理缺失数据

有时候输入值(x的成分)可能没观测到;这可能是创安器故障导致的,也可能是记录时候没弄完整等等.这就叫做数据缺失问题 (missing data problem,Little. and Rubin 1987).对于生成模型来说,能在原则上有办法处理这种缺失数据是最大的优势了.

规范化表达一下,使用一个二值化响应变量$r_i \in \{0,1\}$,用来判断x_i是否被观测到.联合模型形式就是$p(x_i,r_i|\theta,\phi)=p(r_i|x_i,\phi)p(x_i|\theta)$,其中的$\phi$是控制该项目是否被检测到的参数.如果假设$p(r_i|x_i,\phi)=p(r_i|\phi)$,就说数据是随机缺失的(missing completely at random,缩写为MCAR).如

果这这两个假设都不成立,就说数据是非随机缺失(not missing completely at random,缩写为NMCAR).这种情况下,就必须对数据缺失机制进行建模了,因为确实的模式中包含了关于缺失数据和对应参数的信息.这在协作过滤(most collaborative filtering)问题中最常见.更多内容参考(Marlin 2008).接下来先假设数据是随机丢失的.

处理缺失数据的时候,最好要区分出来数据缺失发生在测试的时候(而训练数据是完整的)还是发生在训练的时候,训练时候有缺失就更麻烦一些.接下来会分别讨论两种情况.要注意在测试的时候分类标签总是缺失的,这就是测试的定义决定的;如果在训练的时候分类标签有时候也缺失,这样问题就是半监督学习(semi-supervised learning)了.

8.6.2.1 测试时候的数据丢失

在生成分类其中,随机缺失数据的特征可以通过边缘化而去掉(marginalizing them out).例如,如果缺失x_1的值,可以计算:

$$p(y=c|x_{2:D},\theta) \propto p(y=c|\theta)p(x_{2:D}|y=c,\theta) \tag{8.94}$$
$$= p(y=c|\theta)\sum_{x_1} p(x_1,x_{2:D}|y=c,\theta) \tag{8.95}$$

如果使用朴素贝叶斯加设,这个边缘化(maginalization)过程可以如下进行:
$\sum_{x_1} p(x_1,x_{2:D}|y=c,\theta) = [\sum_{x_1} p(x_1|\theta_{1c})]\prod_{j=2}^{D} p(x_j|\theta_{jc}) = \prod_{j=2}^{D} p(x_j|\theta_{jc})$(8.96)

其中利用了$\sum_{x_1} p(x_1|y=c,\theta)=1$.因此在一个朴素贝叶斯分类器里面,只要在测试的时候忽略掉确实特征就可以了.类似的在判别分析里面,不论估计参数的时候使用了什么样的规范化方法(regularization method),都可以解析地将确实变量边缘化来去掉(参考本书4.3):

$$p(x_{2:D}|y=c,\theta) = N(x_{2:D}|\mu_{c,2:D},\Sigma_{c,2:d,2:D}) \tag{8.97}$$

8.6.2.2 训练时候的数据丢失

训练时候数据丢失就不那么好处理了.尤其是这时候计算最大似然估计(MLE)或者最大后验分布(MAP)就都不再是简单的优化问题了,具体原因在本书11.3.2中会讲到.不过也不要紧,因为很快就会学到很多更复杂的算法(比如本书11.4要讲的期望最大化算法(EM))来对这种情况下的最大似然估计(MLE)和最大后验分布(MAP)进行近似.

8.6.3 费舍尔线性判别分析(FLDA)*

判别分析(Discriminant analysis)是用来分类的一种生成方法(generative approach),需要对特征拟合一个多元正态分布(MVN).这在高维度的情况下可能就很困难.所以有一种办法就是降低特征$x \in R^D$的维度,然后对得到的低维度特征$z \in R^L$来拟合多元正态分布(MVN).最简单的方法就是使用线性投影矩阵$z=Wx$,这里的W就是一个$L \times D$的矩阵.可以使用主成分分析(PCA)来找到

矩阵W(参考本书12.2);得到的结果和正则化线性判别分析(RDA,参考本书4.2.6)类似,因为奇异值分解(SVD)和主成分分析(PCA)是本质上等价的.不过主成分分析(PCA)是无监督技术,不需要考虑类标签.所以得到的低维度特征就并不见得一定是对于分类来说的最优选择,如图8.11所示.度还有一种方法,就是找使用高斯类条件密度模型来尽可能降低维度数据分类最优的矩阵W.这里使用高斯分布是合理的,因为我们计算的是一系列特征的线性组合,虽然这些特征可能不见的是高斯分布的.这种方法就叫做费舍尔线性判别分析(Fisher's linear discriminant analysis,缩写为FLDA).

此处参考原书图8.11

费舍尔线性判别分析(FLDA)是对判别方法和生成方法的一种混合.这个方法的缺点就是限制在使用的$L \leq C-1$小于类标签数目减一的维度,而不论D的维度是多少,具体原因后面会解释.这样在二分类情况下,就意味着要找一个单独向量w来对数据进行投影.下面就推到一些二分类情况优化w的过程.然后再泛化到多分类的情况下,最终对这个方法给一个概率论的解释.

8.6.3.1 一维最优投影的推导

接下来要对二分类情况下的最优方向w进行推导,这部分参考了(Bishop 2006b, Sec 4.1.4).定义类条件均值(class-conditional means)为:

$\mu_1 = \frac{1}{N_1} \sum_{i:y_i=1} x_i, \mu_2 = \frac{1}{N_2} \sum_{i:y_i=2} x_i$(8.98)

设$m_k = w^T \mu_k$为每个均值在线w上的投影.另外设$z_i = w^T x_i$为数据在线上的投影.这样投影点的方差就正比于(proportional to):
$s_k^2 = \sum_{i:y_i=k} (z_i - m_k)^2$(8.99)

咱们的目标就是找出能让均值距离$m_2 - m_1$最大的w,同时也要保证投影的簇(cluster)是紧密的(tight):

$J(w) = \frac{(m_2-M_1)^2}{}$(8.100)

将等好友吧改写成关于w的形式如下所示:

$J(w) = \frac{w^T S_B w}{w^T S_W w}$(8.101)

其中的S_B为类间散布矩阵(between-class scatter matrix):

$S_B = (\mu_2 - \mu_1)(\mu_2 - \mu_1)^T$(8.102)

S_W为类内散布矩阵(within-class scatter matrix):
$S_W = \sum_{i:y_i=1} w^T (x_i - \mu_1)(x_i - \mu_1)^T + \sum_{i:y_i=2} w^T (x_i - \mu_2)(x_i - \mu_2)^T$(8.103)

然后两边分别乘以w^T和w:

$$w^T S_B w = w^T (\mu_2 - \mu_1)(\mu_2 - \mu_1)^T w = (m_2 - m_1)(m_2 - m_1)^T \text{(8.104)}$$

$$w^T S_W w = \sum_{i:y_i=1} w^T (x_i - \mu_1)(x_i - \mu_1)^T w + \sum_{i:y_i=2} w^T (x_i - \mu_2)(x_i - \mu_2)^T w \quad (8.105$$

$$= \sum_{i:y_i=1} (z_i - m_1)^2 + \sum_{i:y_i=2} (z_i - m_2)^2 \quad (8.106$$

等式8.101就是两个标量的比;可以关于我求导数然后就等于零了.很明显(参考练习12.6)$J(w)$最大化的时候为:

$$S_B w = \lambda S_W w \text{(8.107)}$$

其中的:
$\lambda = \frac{w^T S_B w}{w^T S_W w}$(8.108)

等式107也叫广义特征值问题(generalized eigenvalue problem).如果S_W可逆,就可以转换成一个规范特征值问题:
$S_W^{-1} S_B w = \lambda w$(8.109)

不过在二分类情况下,就有一个更简单的解了.由于

$$S_B w = (\mu_2 - \mu_1)(\mu_2 - \mu_1)^T w = (\mu_2 - \mu_1)(m_2 - m_1) \text{(8.110)}$$

所以通过等式8.109就有

$$\lambda w = S_W^{-1} (\mu_2 - \mu_1)(m_2 - m_1) \quad (8.111)$$

$$w \propto S_W^{-1} (\mu_2 - \mu_1) \quad (8.112)$$

由于我们只关心方向,缩放因数就无所谓了,所以可以直接写成:

$$w = S_W^{-1} (\mu_2 - \mu_1) \text{(8.113)}$$

这就是二分类情况下的最优解了.如果$$S_W \propto I$,就意味着汇总协方差矩阵(pooled covariance matrix)是各向同性的(isotropic),这样w就正比于联合类均值(joins class means)的向量.如图8.11所示,这是一个直觉上很合理的投影方向.

8.6.3.2 扩展到高维度和多分类情况

把上面的思路扩展到多分类情况,以及高维度子控件,只需要找到一个投影矩阵W,从D到L进行映射,最大化
$J(W) = \frac{W \Sigma_B W^T}{W \Sigma_W W^T}$(8.114)

其中

$$\Sigma_B \triangleq \sum_c \frac{N_c}{N}(\mu_c - \mu)(\mu_c - \mu)^T \quad (8.115)$$

$$\Sigma_W \triangleq \sum_c \frac{N_c}{N}\Sigma_c \quad (8.116)$$

$$\Sigma_c \triangleq \frac{1}{N_c}\sum_{i:y_i=c}(x_i - \mu_c)(x_i - \mu_c)^T \quad (8.117)$$

解就是:

$$W = \Sigma_W^{-\frac{1}{2}} U \quad (8.118)$$

其中的U是$\Sigma_W^{-\frac{1}{2}} \Sigma_B \Sigma_W^{-\frac{1}{2}}$的L主特征向量(L leading eigenvectors),假设Σ_W是非奇异的.(如果是奇异的,可以先对全部数据进行主成分分析(PCA).)

此处参考原书图8.12

图8.12所示为将这个方法应用到$D = 10$维度的运输局,其中有$C = 11$个不同的原因声音(vowel sounds).图中可见很明显FLDA得到的分类比PCA效果好.

要注意FLDA是限制在最多$L \leq C - 1$维度的线性子空间的,不管D有多大,这是因为类间协方差矩阵Σ_B的秩(rank)就是$C - 1$.(这里的-1是因为μ项是μ_c的线性函数.)这限制了FLDA的使用.

8.6.3.3 FLDA的概率论解释*

对FLDA方法的概率论解释参考了(Kumar and Andreo 1998; Zhou et al. 2009).他们提出了一个模型,叫做异方差线性判别分析(heteroscedastic LDA,缩写为HLDA),过程如下所示.设W是一个$D \times D$的可逆矩阵,设$z_i = Wx$是对数据的转换.然后对转换后的数据的每一类都拟合完整协方差高斯分布(full covariance Gaussians),但仅限于前面L个成分为分类拟合(class-specific);剩下的$H = D - L$个成分在类间共享,因此也就不做区分(not be discriminative).也就是使用:

$$p(z_i|\theta, y_i = c) = N(z_i|\mu_c, \Sigma_c) \quad (8.119)$$

$$\mu_c \triangleq (m_c; m_0) \quad (8.120)$$

$$\Sigma_c \triangleq \begin{pmatrix} S_c & 0 \\ 0 & S_0 \end{pmatrix} \quad (8.121)$$

其中的m_0是共享的H维度均值,而S_0是共享的$H \times H$协方差.原始数据(未变换过)的概率密度函数(pdf)为:

$$p(x_i|y_i = c, W, \theta) = |W|N(Wx_i|\mu_c, \Sigma_c) \quad (8.122)$$

$$= |W|N(W_L x_i|m_c, S_c)N(W_H x_i|m_0, S_0) \quad (8.123)$$

其中$w = \begin{pmatrix} W_L \\ W_H \end{pmatrix}$.对于固定的W,很不容易推导出$\theta$的最大似然估计(MLE).然后可以使用梯度方法优化W.

在Σ_c为对角阵的特殊情况下,有一个W的闭合形式的解(Gales 1999).当所有Σ_c都一样的情况下,就会到了经典的线性判别分析(classical LDA, Zhou et al. 2009).

总的来看,如果雷协方差在可判别子空间内不相等(比如Σ_c独立于c是个错误假设),那么HLDA就会优于LDA.用合成数据(synthetic dat)就很容易演示出这种情况,另外更有挑战性的一些任务比如语音识别等等当中也是如此(Kumar and Andreo 1998).另外我们还可以通过让每个类都有自己的投影矩阵来进一步扩展这个模型,这样就成了多重线性判别分析(multiple LDA,Gales 2002).

练习略