PRML（Pattern Recognition And Machine Learning）读书会

第四章 Linear Models for Classification

主讲人 planktonli

QQ 群 177217565

读书会微信公众平台请扫描下面的二维码

planktonli(1027753147) 19:52:28
现在我们就开始讲第四章，第四章的内容是关于 线性分类模型，主要内容有四点:
1) Fisher 准则的分类,以及它和最小二乘分类的关系 (Fisher 分类是最小二乘分类的特例)
2) 概率生成模型的分类模型
3) 概率判别模型的分类模型
4) 全贝叶斯概率的 Laplace 近似
需要注意的是，有三种形式的贝叶斯：
1) 全贝叶斯
2) 经验贝叶斯
3) MAP 贝叶斯
我们大家熟知的是 MAP 贝叶斯
MAP(poor man’ s Bayesian) :不涉及 marginalization ,仅是一种按后验概率最大化的 point estimate。这里的 MAP(poor man’ s Bayesian)是属于 点概率估计的。而全贝叶斯可以看作对 test 样本的所有参数集合的加权平均，PRML 说的 Bayesian 主要还是指 Empirical Bayesian：

Fully Bayesian：需要 marginalize with respect to hyper-parameters as well as parameters。而这往往是 analytical intractable 的。

对于 curve fitting 的例子($p(w|\alpha)=N(w|0,\alpha^{-1}I)$, $p(t|x,w,\beta)=N(t|y(x,w),\beta^{-1})$)来说，就是：

$$p(t|t)=\iiint p(t|w,\beta)p(w|t,\alpha,\beta)p(\alpha,\beta|t)dwd\alpha d\beta$$

Empirical Bayes/type 2 maximum likelihood/evidence approximation：对 hyper-parameter 采取这样的策略，即先求出使得 marginal likelihood 最大化的参数 α^* 和 β^*，然后使 hyper-parameter 取固定的值 α^* 和 β^*，再对 w 进行 marginalize：

$$p(t|t)\approx p(t|t,\alpha^*,\beta^*)=\int p(t|w,\beta^*)p(w|t,\alpha^*,\beta^*)dw$$

这里的 α^* 和 β^* 为超参 。

Curve fitting 为例子：
1) MLE，直接对 likelihood function 求最大值，得到参数 w。该方法属于 point estimate。
2) MAP (poor man’ s bayes)，引入 prior probability，对 posterior probability 求最大值，得到 w。MAP 此时相当于在 MLE 的目标函数（ likelihood function ）中加入一个 L2 penalty。该方法仍属于 point estimation。
3) fully Bayesian approach ,需使用 sum rule 和 product rule(因为“degree of belief”的 machinery 和概率相同，因此这两个 rule 对“degree of belief”成立），而要获得 predictive distribution 又需要 marginalize (sum or integrate) over the whole of parameter space w：

$$p(t|x,X,t)=\int p(t|x,w)p(w|X,t)dw$$

其中，x 是待预测的点，X 是观察到的数据集，t 是数据集中每个数据点相应的 label。其实是用参数 w 的后验概率为权 ,对 probability 进行一次加权平均 ;因此这个过程需要对 w 进行积分 ,即 marginalization。由于 marginalization 通常是非常难求取的，所以一般在针对 Graphical Model 的时候就需做 Laplace approximation、Variation inference、MCMC 采样这些。
所以我们要建立的概念是:Graphical Model 的东西是一个需要 marginalization 的。

下面我们看看本讲的内容：
首先将上节 LS(Least Square)方法直接用于求分类问题，就可以得到 Least squares for classification。

$$\widetilde{\mathbf{W}} = (\widetilde{\mathbf{X}}^{\mathrm{T}}\widetilde{\mathbf{X}})^{-1}\widetilde{\mathbf{X}}^{\mathrm{T}}\mathbf{T} = \widetilde{\mathbf{X}}^{\dagger}\mathbf{T}$$

$$\mathrm{y}(\mathrm{x}) = \widetilde{\mathbf{W}}^{\mathrm{T}}\tilde{\mathrm{x}} = \mathbf{T}^{\mathrm{T}}\left(\widetilde{\mathbf{X}}^{\dagger}\right)^{\mathrm{T}}\tilde{\mathrm{x}}.$$

Linear Discriminant Function

Linear discriminant function for a vector $\mathbf{x}$

$$y(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + w_0$$

where $\mathbf{w}$ is called weight vector, and w_0 is a bias.

The classification function is

$$C(\mathbf{x}) = \text{sign}(\mathbf{w}^T\mathbf{x} + w_0)$$

where step function $\text{sign}(\cdot)$ is defined as

$$\text{sign}(a) = \begin{cases} +1, & a \geqslant 0 \\ -1, & a < 0 \end{cases}$$

一般线性模型 Generalized Linear Model: an activation function acting on a linear function of the feature variables：

$$y(x) = f(w^T\phi(x) + w_0)$$

称为 Generalized Linear Model (GLM)，其中 x 为输入数据向量，$\phi(x)$ 为 x 的 feature vector，
而 f 是一个函数，称为 activation function，f 的反函数称为 link function。

(1) 当 f 是 nonlinear function 的时候，GLM 是一个 classification model；

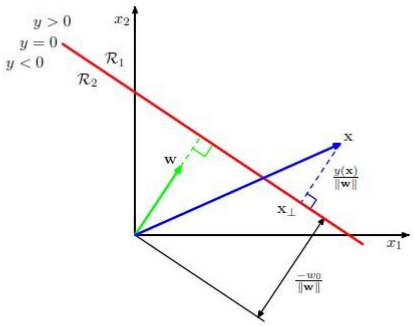
(2) 当 f 是 identity function 的时候，GLM 是一个 regression model。

Linear Model 对于回归和分类的区别在于：激活函数的不同

$$\text{sign}(\mathbf{w}^T\mathbf{x} + w_0)$$

这里 sign 就是一个非线性的函数，其实是一个间断函数,非连续的。
下图证明了点到平面的距离公式。超平面：在一个 D 维 Euclidean space 中的超平面是一它的一个 D-1 维流形，而且该空间是一个线性空间。Linearly separable：分布于 D 维空间中的全部数据点可以用超平面无错地分隔成类。Coding scheme：1-of-K binary coding scheme，即如果有 K 个类，某数据点属于第 i 个类，则表示为一个 K 维向量，该向量除了第 i 个分量是 1，其余都是 0。

Properties of Linear Discriminant Function



$y(\mathbf{x}) = 0$ for $\mathbf{x}$ on the decision surface. The normal distance from the origin to the decision surface is

$$\frac{\mathbf{w}^T\mathbf{x}}{\|\mathbf{w}\|} = -\frac{w_0}{\|\mathbf{w}\|}$$

So w_0 determines the location of the decision surface.

Properties of Linear Discriminant Function

Let

$$\mathbf{x} = \mathbf{x}_\perp + r\frac{\mathbf{w}}{\|\mathbf{w}\|}$$

where $\mathbf{x}_\perp$ is the projection $\mathbf{x}$ on the decision surface. Then

$$\mathbf{w}^T\mathbf{x} = \mathbf{w}^T\mathbf{x}_\perp + r\frac{\mathbf{w}^T\mathbf{w}}{\|\mathbf{w}\|}$$
$$\mathbf{w}^T\mathbf{x} + w_0 = \mathbf{w}^T\mathbf{x}_\perp + w_0 + r\|\mathbf{w}\|$$
$$y(\mathbf{x}) = r\|\mathbf{w}\|$$
$$r = \frac{y(\mathbf{x})}{\|\mathbf{w}\|}$$

Simpler notion: define $\widetilde{\mathbf{w}} = (w_0, \mathbf{w})$ and $\widetilde{\mathbf{x}} = (1, \mathbf{x})$ so that

$$y(\mathbf{x}) = \widetilde{\mathbf{w}}^T\widetilde{\mathbf{x}}$$

关于超平面,线性可分的一些概念，在多类情况,可以使用 1 对 1,1 对多分类器的方式，例如:

你要分类 3 类物体: 苹果,西瓜,香蕉

那么 1 对 1 就是建立 6 个分类器

那么 1 对 1 就是建立 3 个分类器

苹果,西瓜

苹果,香蕉

西瓜,香蕉

1 对多分类器就是:

苹果和非苹果

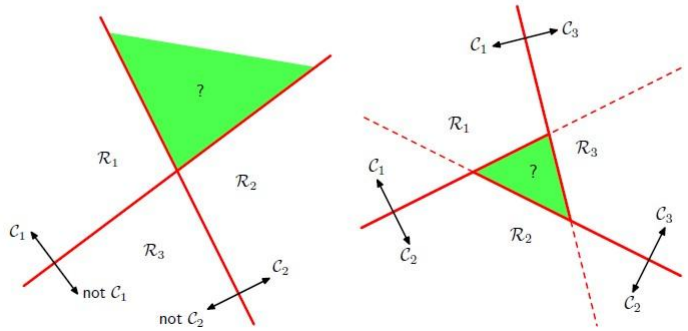
西瓜和非西瓜

香蕉和非香蕉

Multiple Classes: Simple Extension

One-versus-the-rest classifier: classify $\mathcal{C}_k$ and samples not in $\mathcal{C}_k$.

One-versus-one classifier: classify every pair of classes.



左边是 1 对多,右边是 1 对 1，都存在一些无法分类的情况,也就是绿色区域部分。

多分类的决策域是单连通,而且是凸的，下面给出了证明：

Multiple Classes: K-Class Discriminant

A single K-class discriminant comprising K linear functions

$$y_k(\mathbf{x}) = \mathbf{w}_k^T\mathbf{x} + w_{k0}$$

Decision function

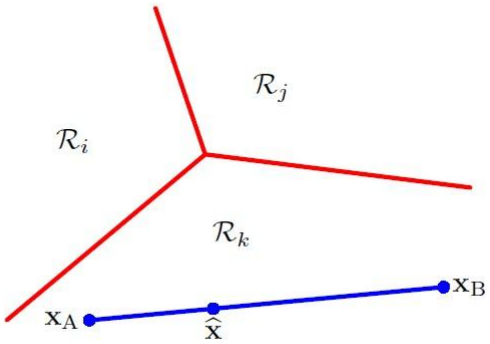
$$\mathcal{C}(\mathbf{x}) = k, \text{ if } y_k(\mathbf{x}) > y_j(\mathbf{x}) \; \forall j \neq k$$

The decision boundary between class $\mathcal{C}_k$ and $\mathcal{C}_j$ is given by $y_k(\mathbf{x}) = y_j(\mathbf{x})$

$$(\mathbf{w}_k - \mathbf{w}_j)^T\mathbf{x} + (w_{k0} - w_{j0}) = 0$$

证明的图形示意：

Property of the Decision Regions



上面有没有问题?没有就继续讲 Fisher's Linear Discriminant 了。

echo<echotooo@gmail.com> 20:26:46

无法分类的情况一般怎么办？就是绿色区域了
planktonli(1027753147) 20:27:35
恩,那就是可能出现判断错误了，这个没有办法。
echo<echotooo@gmail.com> 20:28:12
哦，好的，pass。
planktonli(1027753147) 20:28:54
好了,现在看看 Fisher's Linear Discriminant，Linear Discriminant Analysis, LDA),也叫做 Fisher 线性判别，这个要和 Graphical Model 的 Latent Dirichlet allocation 区分开。我个人认为 LDA 可以看成一个有监督降维的东西，这些 PCA(主成分分析),ICA(独立成分分析)也是降低维的，不过是无监督的东西，包括 mainfold dimension reduction 的，都是无监督的。LDA 是降低到一个投影方向上,使得它的可分性最好而 PCA 是要找它的主要成分也就是使得 Loss 最小的方向，LDA 要求 类间散度最大,类内聚合度最强。

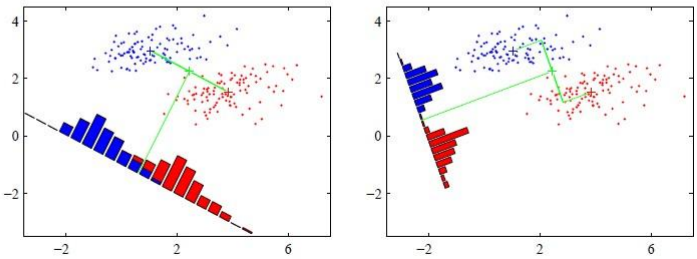
Fisher's Linear Discriminant

Pursue the optimal linear projection on which the two classes can be maximally separated

$$y = \mathbf{w}^T\mathbf{x}$$

The mean vectors of the two classes

$$\mathbf{m}_1 = \frac{1}{N_1}\sum_{n\in\mathcal{C}_1}\mathbf{x}_n, \quad \mathbf{m}_2 = \frac{1}{N_2}\sum_{n\in\mathcal{C}_2}\mathbf{x}_n$$



Difference of means　　Fisher linear discriminant

类间散度最大是通过它们的均值距离体现的，而 类内聚合度最强 是通过 类内的点到均值的散的程度表达的，也就是说 Fisher 分类是 LS 的特例，好了,看大家对 Fisher 还有什么疑问?
echo<echotooo@gmail.com> 20:41:53
....LDA 是基于高斯分布假设上的吗？
planktonli(1027753147) 20:44:12
LDA 的样本是需要在 Gaussian 假设下,会有 power performance 的，如果 data 的 distribution 是非常不规则的，那么 LDA 肯定是失效的。那就需要用些 Kernel 等的 trick 了。
echo<echotooo@gmail.com> 20:45:29
这是为什么，是因为它的公式推导的时候有用到高斯分布的假设吗？
planktonli(1027753147) 20:45:32
推导不需要 Gaussian 假设
网络上的尼采(813394698)　20:46:01
用到了均值
echo<echotooo@gmail.com> 20:46:40
协方差部分呢？
planktonli(1027753147) 20:47:00
需要两类的 Between class Variance，这个是通过 均值差表达的。
echo<echotooo@gmail.com> 20:47:28

嗯嗯，好像懂了，谢谢。
planktonli(1027753147) 20:47:45
如果两个类的 mean 完全相等,那么 LDA 肯定是失效的。
if the distribution of data is not so good, then we may use Kernel Fisher discriminant analysis
I mean that the distribution doesn't meet the gaussian。
echo<echotooo@gmail.com> 20:50:35
难道 kfda 不对高斯分布有偏好吗？
planktonli(1027753147) 20:50:42
the detail info you can c the web site http://en.wikipedia.org/wiki/Kernel_Fisher_discriminant_analysis KDA 算法步聚，大部分跟 LDA 相同，不同的地方是用到了 Kernel 方法构造了矩阵 Sb, Sw，这里的 KDA 就是 kernel fisher 了。
电闪雷鸣(37633749) 20:52:11
OK，明白
planktonli(1027753147) 20:52:24
好了,下面看看 NN 神经网络的 perceptron ，这个是一个单层的东西，注意它的 training error 函数 ，优化过程用的是梯度下降法：

The Perceptron Algorithm

More general form of a linear classifier

$$y(\mathbf{x}) = \text{sign}\left(\mathbf{w}^T\phi(\mathbf{x})\right)$$

where $\phi(\mathbf{x})$ is the feature vector of $\mathbf{x}$ (recall the basis functions in linear regression), and typically includes a bias component $\phi_0(\mathbf{x}) = 1$.

Given a set of samples $\{\mathbf{x}_n, t_n\}_{n=1}^N$, $\mathbf{x}_n \in \mathbb{R}^D$, $t_n \in \{-1, 1\}$, our goal is to find the optimal parameter $\mathbf{w}$ to minimize the training error

$$E_p(\mathbf{w}) = -\sum_{n\in\mathcal{M}} \mathbf{w}^T\phi_n t_n$$

where $\phi_n = \phi(\mathbf{x}_n)$ and $\mathcal{M}$ denotes the set of all misclassified patterns.

The Perceptron Algorithm

Let $\mathcal{M} = \{n\}$ where $\mathbf{x}_n$ is misclassified. Then

$$E_p(\mathbf{w}) = -\mathbf{w}^T\phi_n t_n$$

Stochastic gradient descent:

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} - \eta\nabla E_p(\mathbf{w}) = \mathbf{w}^{(\tau)} + \eta\phi_n t_n$$

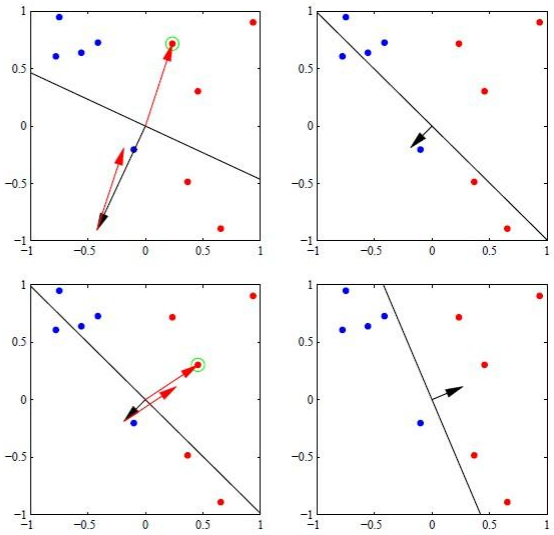
where η is the learning rate parameter and τ is an integer that indexes the steps of the algorithm. Without loss of generality, we can set $\eta = 1$.

For sample n

$$-\mathbf{w}^{(\tau+1)T}\phi_n t_n = -\mathbf{w}^{(\tau)T}\phi_n t_n - (\phi_n t_n)^T\phi_n t_n < -\mathbf{w}^{(\tau)T}\phi_n t_n$$

where we have set $\eta = 1$. This does not imply that the contribution to the error function from the other misclassified patterns will have been reduced.

Illustration of the Perceptron Algorithm



Properties of the Perceptron Algorithm

Perceptron convergence theorem: if there exists an exact solution (which means the training data is linearly separable), then the perceptron learning algorithm is guaranteed to find an exact solution in a finite number of steps.

In practice, the number of steps required to achieve convergence could be substantial. Until convergence is achieved, we are not able to distinguish between a nonseparable problem and one that is simply slow to converge.

The perceptron algorithm does not converge when the data are not linearly separable.

好了,perceptron 是比较简单的。

下面看，Probabilistic Generative Models，通过 MAP 方式建立概率模型，需要 先验概率,类条件概率和边缘概率。2 类的 Probabilistic Generative Models 就是 logistic sigmoid function：

Generative Models

Bayesian decision theory

$$\mathcal{C}(\mathbf{x}) = \mathcal{C}_k, \text{ if } p(\mathcal{C}_k|\mathbf{x}) \geqslant p(\mathcal{C}_j|\mathbf{x}) \; \forall j \neq k$$

The *generative* approach: model the class-conditional density $p(\mathbf{x}|C_k)$ and the priors $p(C_k)$, and use the posterior $p(C_k|\mathbf{x})$ through Bayes' theorem.

Two-Class Generative Model

The posterior for class $\mathcal{C}_1$:

$$\begin{aligned} p(\mathcal{C}_1|\mathbf{x}) &= \frac{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1)}{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1) + p(\mathbf{x}|\mathcal{C}_2)p(\mathcal{C}_2)} \\ &= \frac{1}{1 + \frac{p(\mathbf{x}|\mathcal{C}_2)p(\mathcal{C}_2)}{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1)}} \\ &= \frac{1}{1 + \exp(-a)} \\ &= \sigma(a) \end{aligned}$$

where we have defined

$$a = \log \frac{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1)}{p(\mathbf{x}|\mathcal{C}_2)p(\mathcal{C}_2)}$$

and $\sigma(a)$ is the *logistic sigmoid* function defined by

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

which is also called 'squashing function' because it maps the real axis to a finite

Logistic Sigmoid and Logit Functions

Logistic sigmoid function has the property

$$\sigma(-a) = 1 - \sigma(a)$$

The inverse of the logistic sigmoid is given by

$$a = \log \frac{\sigma}{1 - \sigma}$$

which is known as the *logic function*, or *log odds*.

这种方法需要假设 input 的分布，即得到 class-conditional distribution，用贝叶斯定理转化成后验概率后，就是和 Discriminant model 一样进行 make decision 了。

Multi-class Generative Model

For the case of $K > 2$ classes, we have

$$\begin{aligned} p(\mathcal{C}_k|\mathbf{x}) &= \frac{p(\mathbf{x}|\mathcal{C}_k)p(\mathcal{C}_k)}{\sum_j p(\mathbf{x}|\mathcal{C}_j)p(\mathcal{C}_j)} \\ &= \frac{\exp(a_k)}{\sum_j \exp(a_j)} \end{aligned}$$

which is known as the *normalized exponential*, and a_k's are defined by

$$a_k = \log p(\mathbf{x}|\mathcal{C}_k)p(\mathcal{C}_k)$$

The normalized exponential is also known as the *softmax* function. If $a_k \gg a_j \; \forall j \neq k$, then $p(\mathcal{C}_k|\mathbf{x}) \approx 1$, and $p(\mathcal{C}_j|\mathbf{x}) \approx 0$.

在 gaussian 分布的情况下,我们分析：

Gaussian Likelihood Function

Assume that the class-conditional densities are Gaussians that share the same covariance matrix. The pdf for class C_k is

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

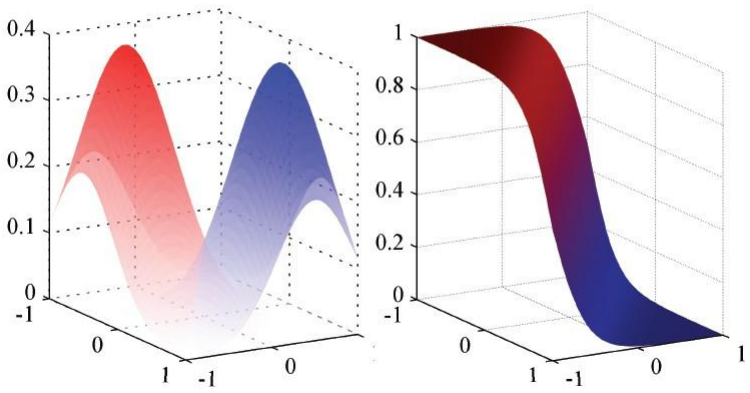
For the two-class case, we have

$$\begin{aligned} a(\mathbf{x}) &= \log\frac{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1)}{p(\mathbf{x}|\mathcal{C}_2)p(\mathcal{C}_2)} \\ &= -\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_1)^T\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu}_1) + \frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_2)^T\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu}_2) + \log\frac{p(\mathcal{C}_1)}{p(\mathcal{C}_2)} \\ &= (\boldsymbol{\mu}_1-\boldsymbol{\mu}_2)^T\boldsymbol{\Sigma}^{-1}\mathbf{x} - \frac{1}{2}\boldsymbol{\mu}_1^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_1 + \frac{1}{2}\boldsymbol{\mu}_2^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_2 + \log\frac{p(\mathcal{C}_1)}{p(\mathcal{C}_2)} \\ &= \mathbf{w}^T\mathbf{x} + w_0 \end{aligned}$$

where

$$\begin{aligned} \mathbf{w} &= \boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_1-\boldsymbol{\mu}_2) \\ w_0 &= -\frac{1}{2}\boldsymbol{\mu}_1^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_1 + \frac{1}{2}\boldsymbol{\mu}_2^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_2 + \log\frac{p(\mathcal{C}_1)}{p(\mathcal{C}_2)} \end{aligned}$$

Two-Class Gaussian Likelihood Function



Multi-Class Gaussian Likelihood Function

For general K classes, we have

$$\begin{aligned} a_k &= \log p(\mathbf{x}|\mathcal{C}_k)p(\mathcal{C}_k) \\ &= \boldsymbol{\mu}_k^T\boldsymbol{\Sigma}^{-1}\mathbf{x} - \frac{1}{2}\boldsymbol{\mu}_k^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_k + \log p(\mathcal{C}_k) + \mathbf{x}^T\boldsymbol{\Sigma}^{-1}\mathbf{x} + \text{const} \end{aligned}$$

Notice that $\mathbf{x}^T\boldsymbol{\Sigma}^{-1}\mathbf{x}$ is cancelled in the softmax function. Therefore

$$a_k(\mathbf{x}) = \mathbf{w}_k^T\mathbf{x} + w_{k0}$$

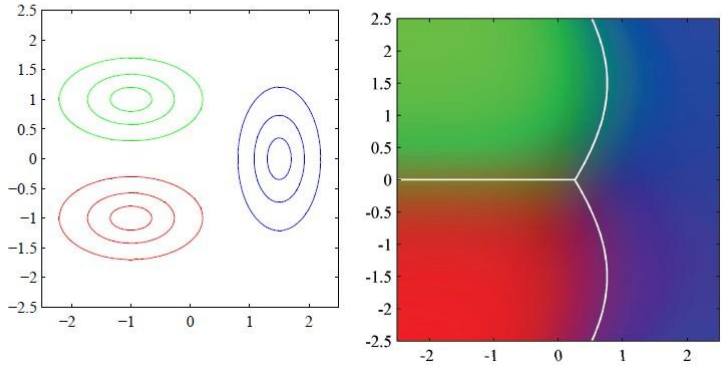
where

$$\begin{aligned} \mathbf{w}_k &= \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_k \\ w_{k0} &= -\frac{1}{2}\boldsymbol{\mu}_k^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_k + \log p(\mathcal{C}_k) \end{aligned}$$

Hence $a_k(\mathbf{x})$ is again linear function of $\mathbf{x}$.

Multi-Class Gaussian Likelihood Function

The covariance matrix of each class may not be identical, and the cancellation no long
occur. This leads to *quadratic discriminant.*



协方差矩阵不同,则变成了 2 次分类了，在 2 类情况下,我们用 MLE 方法，估计参数：

Maximum Likelihood Estimator for Two Classes

Given a data set $\{\mathbf{x}_n, t_n\}_{n=1}^N$, $t_n \in \{0, 1\}$. Denote the prior class probability $p(\mathcal{C}_1) = \pi$ and $p(\mathcal{C}_2) = 1 - \pi$.

For a data point $\mathbf{x}_n$ generated from class $\mathcal{C}_1$ we have $t_n = 1$, and hence

$$p(\mathbf{x}_n, \mathcal{C}_1) = p(\mathcal{C}_1)p(\mathbf{x}_n|\mathcal{C}_1) = \pi\mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_1, \boldsymbol{\Sigma})$$

For a data point $\mathbf{x}_n$ generated from class $\mathcal{C}_2$ we have $t_n = 0$, and hence

$$p(\mathbf{x}_n, \mathcal{C}_2) = p(\mathcal{C}_2)p(\mathbf{x}_n|\mathcal{C}_2) = (1 - \pi)\mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_2, \boldsymbol{\Sigma})$$

Therefore, the likelihood function is

$$p(\mathbf{t}|\pi, \boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \boldsymbol{\Sigma}) = \prod_{n=1}^{N} [\pi\mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_1, \boldsymbol{\Sigma})]^{t_n} [(1 - \pi)\mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_2, \boldsymbol{\Sigma})]^{1-t_n}$$

where $\mathbf{t} = (t_1, \cdots, t_N)^T$.

MLE for π

Maximize the log likelihood

$$\log p(\mathbf{t}|\pi, \boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \boldsymbol{\Sigma}) = \sum_{n=1}^{N} \Big\{ t_n \log \pi + t_n \log \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_1, \boldsymbol{\Sigma}) + (1 - t_n) \log(1 - \pi) + (1 - t_n) \log \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_2, \boldsymbol{\Sigma}) \Big\}$$

The terms that depend on π are

$$\sum_{n=1}^{N} \{t_n \log \pi + (1 - t_n) \log(1 - \pi)\}$$

Setting the derivative w.r.t. π equal to zero, we obtain

$$\pi = \frac{1}{N} \sum_{n=1}^{N} t_n = \frac{N_1}{N} = \frac{N_1}{N_1 + N_2}$$

where N_1 denotes the total number of data points in class $\mathcal{C}_1$ and N_2 denotes the total number of data points in class $\mathcal{C}_2$.

MLE for $\boldsymbol{\mu}_1$ and $\boldsymbol{\mu}_2$

The terms in the log likelihood that depend on $\boldsymbol{\mu}_1$

$$\frac{1}{N}\sum_{n=1}^{N} t_n \log \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_1, \boldsymbol{\Sigma}) = -\frac{1}{2}\sum_{n=1}^{N} t_n (\mathbf{x}_n - \boldsymbol{\mu}_1)^T \boldsymbol{\Sigma}^{-1}(\mathbf{x}_n - \boldsymbol{\mu}_1) + \text{const}$$

Setting the derivative w.r.t. $\boldsymbol{\mu}_1$ to zero and obtain

$$\boldsymbol{\mu}_1 = \frac{1}{N_1}\sum_{n=1}^{N} t_n \mathbf{x}_n$$

or simply the mean of all the input vectors $\mathbf{x}_n$ labeled as class $\mathcal{C}_1$. Similarly

$$\boldsymbol{\mu}_2 = \frac{1}{N_2}\sum_{n=1}^{N} (1 - t_n)\mathbf{x}_n$$

MLE for $\boldsymbol{\Sigma}$

The terms in the log likelihood that depend on $\boldsymbol{\Sigma}$

$$\begin{aligned}
&-\frac{1}{2}\sum_{n=1}^{N} t_n \log|\boldsymbol{\Sigma}| - \frac{1}{2}\sum_{n=1}^{N} t_n (\mathbf{x}_n - \boldsymbol{\mu}_1)^T \boldsymbol{\Sigma}^{-1}(\mathbf{x}_n - \boldsymbol{\mu}_1) \\
&-\frac{1}{2}\sum_{n=1}^{N} (1 - t_n) \log|\boldsymbol{\Sigma}| - \frac{1}{2}\sum_{n=1}^{N} (1 - t_n)(\mathbf{x}_n - \boldsymbol{\mu}_2)^T \boldsymbol{\Sigma}^{-1}(\mathbf{x}_n - \boldsymbol{\mu}_2) \\
&= -\frac{N}{2}\log|\boldsymbol{\Sigma}| - \frac{N}{2}\mathrm{Tr}\boldsymbol{\Sigma}^{-1}\mathbf{S}
\end{aligned}$$

where

$$\begin{aligned}
\mathbf{S} &= \frac{N_1}{N}\mathbf{S}_1 + \frac{N_2}{N}\mathbf{S}_2 \\
\mathbf{S}_1 &= \frac{1}{N_1}\sum_{n\in\mathcal{C}_1} (\mathbf{x}_n - \boldsymbol{\mu}_1)(\mathbf{x}_n - \boldsymbol{\mu}_1)^T \\
\mathbf{S}_2 &= \frac{1}{N_2}\sum_{n\in\mathcal{C}_2} (\mathbf{x}_n - \boldsymbol{\mu}_2)(\mathbf{x}_n - \boldsymbol{\mu}_2)^T
\end{aligned}$$

MLE for $\boldsymbol{\Sigma}$

Using what we derived for the MLE for the covariance matrix for a Gaussian distribution, we obtain

$$\boldsymbol{\Sigma} = \mathbf{S}$$

Therefore, the MLE for the parameters

$$\begin{aligned}
\pi &= \frac{N_1}{N_1 + N_2} \\
\boldsymbol{\mu}_1 &= \frac{1}{N_1}\sum_{n=1}^{N} t_n \mathbf{x}_n \\
\boldsymbol{\mu}_2 &= \frac{1}{N_2}\sum_{n=1}^{N} (1 - t_n)\mathbf{x}_n \\
\boldsymbol{\Sigma} &= \mathbf{S}
\end{aligned}$$

步骤小结：

1) 假定 class-conditional distribution 的分布形式，MLE 估计该分布中的参数（从而得到了 class-conditional distribution）

2) 计算每个类别的后验概率。在上面的例子中，得到的后验概率刚好是一个 GLM 模型（Logistic）

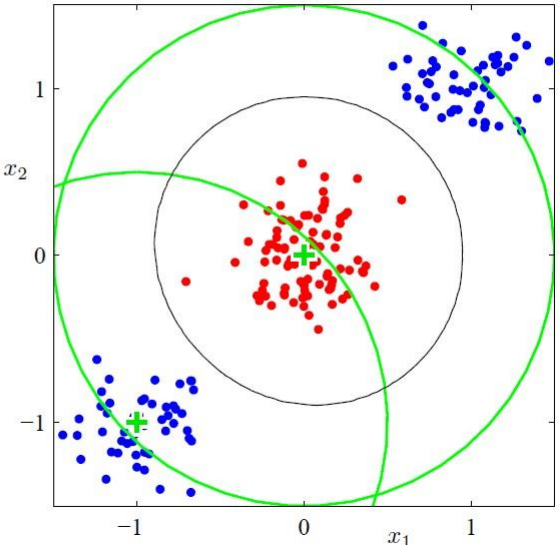
好了,这部分结束了。大家讨论下，没问题就继续了。

Probabilistic Discriminative Models：

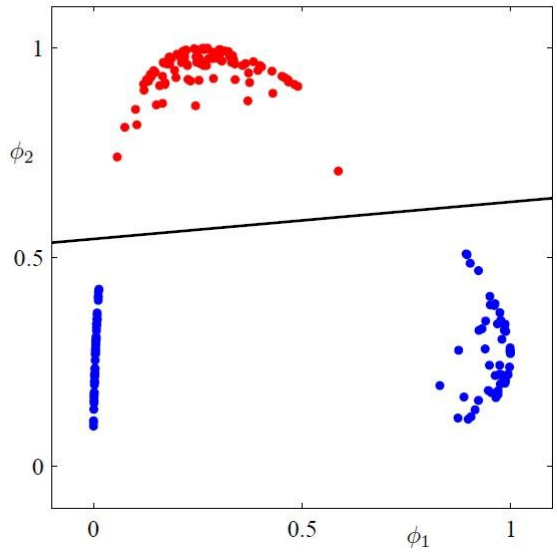
直接建立分类函数模型,而不是建立生成过程模型，生成模型和判别模型的区别：

maping the data from the orginal sapce to a new space may make it linearly separable

Linearly Not Separable in the Original Space



Linearly Separable in the Space of Basis Functions



Work in the New Space

We will work in the feature space $\phi(\mathbf{x})$ which includes $\phi_0(\mathbf{x}) = 1$ so that the corresponding parameter w_0 plays the role of a bias.

Nonlinear basis functions are double-edged sword. It can ease the problem, but can also introduce more overlap of the posterior. Domain expertise is often required to design good nonlinear basis functions.

Logistic Regression

We can directly express the posterior in the space of basis functions

$$p(\mathcal{C}_1|\phi) = y(\phi) = \sigma(\mathbf{w}^T\phi)$$

where $p(\mathcal{C}_2\phi) = 1 - p(\mathcal{C}_1|\phi)$ and $\sigma(\cdot)$ is the *logistic sigmoid* function.

For an $M-$dimensional feature space ϕ, this model has M parameters. In contrast, for a Gaussian generative model using MLE, we have $2M$ parameters for the means, $M(M+1)/2$ for the shared covariance matrix. Together with the class prior $p(\mathcal{C}_1)$, there are in total $M(M+5)/2+1$ parameters.

逻辑回归的最大似然参数估计方法：

MLE for Logistic Regression

Given a data set $\{\phi_n, t_n\}_{n=1}^N$, $\mathbf{t}_n \in \{0,1\}$ and $\phi_n = \phi(\mathbf{x}_n)$. The likelihood function is

$$p(\mathbf{t}|\mathbf{w}) = \prod_{n=1}^{N} y_n^{t_n}(1-y_n)^{1-t_n}$$

where $\mathbf{t} = (t_1, \cdots, t_N)^T$ and $y_n = \sigma(\mathbf{w}^T\phi_n)$.

The error function (negative log likelihood) is

$$E(\mathbf{w}) = -\log p(\mathbf{t}|\mathbf{w}) = -\sum_{n=1}^{N}\{t_n \log y_n + (1-t_n)\log(1-y_n)\}$$

Recall the logistic sigmoid function

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

Therefore

$$\frac{d\sigma}{da} = \frac{\exp(-a)}{(1+\exp(-a))^2} = \sigma(1-\sigma)$$

MLE for Logistic Regression

Because $y_n = \sigma(\mathbf{w}^T\phi_n)$, we have

$$\begin{aligned}\nabla E(\mathbf{w}) &= -\sum_{n=1}^{N}\left\{t_n\frac{y_n(1-y_n)}{y_n} - (1-t_n)\frac{y_n(1-y_n)}{(1-y_n)}\right\}\phi_n \\ &= -\sum_{n=1}^{N}\{t_n(1-y_n) - (1-t_n)y_n\}\,\phi_n \\ &= \sum_{n=1}^{N}(y_n - t_n)\phi_n\end{aligned}$$

The gradient is contributed by the error $y_n - t_n$ times the basis function vector ϕ_n. This update is similar to that of perceptron algorithm.

However, gradient descent algorithms tend to converge to bad minimums.

Newton-Raphson Update

Gradient and Hessian

$$\nabla E(\mathbf{w}) = \sum_{n=1}^{N}(y_n - t_n)\phi_n = \mathbf{\Phi}^T(\mathbf{y} - \mathbf{t})$$

$$\mathbf{H} = \nabla\nabla E(\mathbf{w}) = \sum_{n=1}^{N} y_n(1-y_n)\phi_n\phi_n^T = \mathbf{\Phi}^T\mathbf{R}\mathbf{\Phi}$$

where $\mathbf{\Phi} = (\phi_1, \cdots, \phi_N)^T \in \mathbb{R}^{N\times M}$ is the design matrix, $\mathbf{y} = (y_1, \cdots, y_N)^T$ is the vector of the posterior based on the current $\mathbf{w}$, and $\mathbf{R}$ is a diagonal matrix

$$R_{nn} = y_n(1-y_n)$$

This Hessian matrix H is positive definite because $0 < y_n < 1$.

Newton-Raphson Update

Newton-Raphson update

$$\begin{aligned}\mathbf{w}^{(\text{new})} &= \mathbf{w}^{(\text{old})} - (\mathbf{\Phi}^T\mathbf{R}\mathbf{\Phi})^{-1}\mathbf{\Phi}^T(\mathbf{y} - \mathbf{t}) \\ &= (\mathbf{\Phi}^T\mathbf{R}\mathbf{\Phi})^{-1}\left\{\mathbf{\Phi}^T\mathbf{R}\mathbf{\Phi}\mathbf{w}^{(\text{old})} - \mathbf{\Phi}^T(\mathbf{y} - \mathbf{t})\right\} \\ &= (\mathbf{\Phi}^T\mathbf{R}\mathbf{\Phi})^{-1}\mathbf{\Phi}^T\mathbf{R}\mathbf{z}\end{aligned}$$

where

$$\mathbf{z} = \mathbf{\Phi}\mathbf{w}^{(\text{old})} - \mathbf{R}^{-1}(\mathbf{y} - \mathbf{t})$$

In other words

$$\mathbf{w}^{(\text{new})} - \mathbf{w}^{(\text{old})} = -(\mathbf{\Phi}^T\mathbf{R}\mathbf{\Phi})^{-1}\mathbf{\Phi}^T(\mathbf{y} - \mathbf{t})$$

Linearization

We can derive from another perspective.

Let $\mathbf{w}^{(\text{new})} = \mathbf{w}^{(\text{old})} + \Delta\mathbf{w}$. Taylor expansion leads to

$$\begin{aligned} y_n^{(\text{new})} &= \sigma(\mathbf{w}^{(\text{new})T}\phi_n) \\ &= \sigma(\mathbf{w}^{(\text{old})T}\phi_n + \Delta\mathbf{w}^T\phi_n) \\ &\approx y_n^{(\text{old})} + y_n^{(\text{old})}(1 - y_n^{(\text{old})})\phi_n^T\Delta\mathbf{w} \end{aligned}$$

Now we can drop the superscript (old) and put it into the derivative

$$\begin{aligned} \nabla E(\mathbf{w}) &= \sum_{n=1}^{N}(y_n - t_n)\phi_n \\ &= \sum_{N=1}^{N}\phi_n\left[y_n - t_n + y_n(1 - y_n)\phi_n^T\Delta\mathbf{w}\right] \\ &= \mathbf{\Phi}^T(\mathbf{y} - \mathbf{t}) + \mathbf{\Phi}^T\mathbf{R}\mathbf{\Phi}\Delta\mathbf{w} \end{aligned}$$

Setting the gradient to zeo we obtain

$$\Delta\mathbf{w} = -(\mathbf{\Phi}^T\mathbf{R}\mathbf{\Phi})^{-1}\mathbf{\Phi}^T(\mathbf{y} - \mathbf{t})$$

Iterative Reweighted Least Square

1. Initialize $\mathbf{w}$
2. Compute current prediction $y_n = \sigma(\mathbf{w}^T\phi_n)$, and weight $R_nn = y_n(1 - y_n)$
3. Solve linear system: $\Delta\mathbf{w} = -(\mathbf{\Phi}^T\mathbf{R}\mathbf{\Phi})^{-1}\mathbf{\Phi}^T(\mathbf{y} - \mathbf{t})$
4. $\mathbf{w} \leftarrow \mathbf{w} + \Delta\mathbf{w}$
5. If $\Delta\mathbf{w} < \varepsilon$, go to 2.

Frequentist 版本	Bayesian 版本	解模型所用的方法
Linear basis function regression	Bayesian linear basis function regression	前者和后者皆有 closed-form solution
Logistic regression	Bayesian logitstic regression	前者牛顿迭代(IRLS)，后者 Laplace approximation

注意这里 ，Logistic regression 是用于分类的,而不是回归。

好了,这就是 Probabilistic Discriminative Models 的内容，其实质还是 point estimization。

最后看看 Bayesian Logistic Regression：

这里是 we want to approximate the posterior using Gaussian，就是用高斯分布近似后验概率

来看 Laplace Approximation ：

The Laplace Approximation

Suppose the distribution $p(z)$ is defined by

$$p(z) = \frac{1}{Z} f(z)$$

where $Z = \int f(z)\mathrm{d}z$ is the normalization constant and is unknown.

Our goal is to find a Gaussian approximation to a continuous pdf at a mode. In general, it is a very useful technique to use one simple distribution (such as Gaussian) to approximate more complicated distributions.

The Laplace Approximation

The first step is to find a mode of $p(z)$, z_0, such that $p'(z_0) = 0$, or

$$\left.\frac{df(z)}{dz}\right|_{z=z_0} = 0$$

The logarithm of a Gaussian distribution is a quadratic function. Therefore, consider a Taylor expansion of $\log f(z)$ center at mode z_0

$$\log f(z) \approx \log f(z_0) - \frac{1}{2} A (z - z_0)^2$$

where

$$A = -\left.\frac{d^2}{dz^2} \log f(z)\right|_{z=z_0}.$$

The first-order term in the Taylor expansion does not appear because

$$\left.\frac{d}{dz} \log f(z)\right|_{z=z_0} = \frac{f'(z_0)}{f(z_0)} = 0$$

The Laplace Approximation

Taking the exponential we obtain

$$f(z) \approx f(z_0) \exp\left\{-\frac{A}{2}(z - z_0)^2\right\}$$

We then obtain a normalized distribution $q(z)$

$$q(z) = \left(\frac{A}{2\pi}\right)^{\frac{1}{2}} \exp\left\{-\frac{A}{2}(z - z_0)^2\right\}$$

The distribution is valid iff $A > 0$, or the second order derivative is negative at z_0, indicating that z_0 is a local maximum.

Laplace 近似将任意一个分布近似成了高斯分布

High-Dimensional Distribution

Extend the Laplace method to approximate a distribution $p(\mathbf{z}) = f(\mathbf{z})/Z$ where $\mathbf{z} \in \mathbb{R}^M$. At a local maximum z_0 the gradient $\nabla f(\mathbf{z})$ vanishes. So the Taylor expansion leads to

$$\log f(\mathbf{z}) \approx \log f(\mathbf{z}_0) - \frac{1}{2}(\mathbf{z} - \mathbf{z}_0)^T \mathbf{A} (\mathbf{z} - \mathbf{z}_0)$$

where the $M \times M$ Hessian matrix $\mathbf{A}$ is defined by

$$\mathbf{A} = -\nabla\nabla \log f(\mathbf{z})|_{\mathbf{z}=\mathbf{z}_0}$$

Taking the exponential of both sides we obtain

$$f(\mathbf{z}) \approx f(\mathbf{z}_0) \exp\left\{-\frac{1}{2}(\mathbf{z} - \mathbf{z}_0)^T \mathbf{A} (\mathbf{z} - \mathbf{z}_0)\right\}$$

Therefore, the approximate Gaussian distribution is

$$q(\mathbf{z}) = \frac{|\mathbf{A}|^{\frac{1}{2}}}{(2\pi)^{\frac{M}{2}}} \exp\left\{-\frac{1}{2}(\mathbf{z} - \mathbf{z}_0)^T \mathbf{A} (\mathbf{z} - \mathbf{z}_0)\right\} = \mathcal{N}(\mathbf{z}|\mathbf{z}_0, \mathbf{A}^{-1})$$

Bayesian Inference

Since we want to approximate the posterior using Gaussian, it is natural to set a Gaussian prior for the parameter

$$p(\mathbf{w}) = \mathcal{N}(\mathbf{w}|\mathbf{m}_0, \mathbf{S}_0)$$

Recall the likelihood

$$p(\mathbf{t}|\mathbf{w}) = \prod_{n=1}^{N} y_n^{t_n} (1 - y_n)^{1 - t_n}$$

Therefore the logarithm of the posterior is

$$\begin{aligned} \log p(\mathbf{w}|\mathbf{t}) = &-\frac{1}{2}(\mathbf{w} - \mathbf{m}_0)^T \mathbf{\Sigma}_0^{-1} (\mathbf{w} - \mathbf{m}_0) \\ &+ \sum_{n=1}^{N} \{t_n \log y_n + (1 - t_n) \log(1 - y_n)\} \end{aligned}$$

where $y_n = \sigma(\mathbf{w}^T \mathbf{\Phi}_n)$.

Laplace Approximation

Compute gradient and Hessian

$$\begin{aligned} -\nabla \log p(\mathbf{w}|\mathbf{t}) &= \mathbf{\Phi}^T(\mathbf{y} - \mathbf{t}) + \mathbf{S}_0^{-1}\mathbf{w} - \mathbf{S}_0^{-1}\mathbf{m}_0 \\ \mathbf{H} &= \mathbf{S}_0^{-1} + \mathbf{\Phi}^T \mathbf{R} \mathbf{\Phi} = \mathbf{S}_N^{-1} \end{aligned}$$

So

$$\Delta \mathbf{w} = -(\mathbf{S}_0^{-1} + \mathbf{\Phi}^T \mathbf{R} \mathbf{\Phi})^{-1} (\mathbf{\Phi}^T(\mathbf{y} - \mathbf{t}) + \mathbf{S}_0^{-1}\mathbf{w} - \mathbf{S}_0^{-1}\mathbf{m}_0)$$

We can use IRLS to obtain the *maximum a posteriori* (MAP) $\mathbf{w}_{\text{MAP}}$.

The Gaussian approximation to the posterior therefore takes the form

$$q(\mathbf{w}) = \mathcal{N}(\mathbf{w}|\mathbf{w}_{\text{MAP}}, \mathbf{S}_N(\mathbf{w}_{\text{MAP}}))$$

Predictive Distribution

The predictive distribution for class $\mathcal{C}_1$, given a new feature vector $\phi(\mathbf{x})$, is obtained by marginalizing w.r.t. the posterior distribution $p(\mathbf{w}|\mathbf{t})$, which is approximated by a Gaussian distribution $q(\mathbf{w})$:

$$p(\mathcal{C}_1|\phi,\mathbf{t}) = \int p(\mathcal{C}_1|\phi,\mathbf{w})p(\mathbf{w}|\mathbf{t})\mathrm{d}\mathbf{w} \approx \int \sigma(\mathbf{w}^T\phi)q(\mathbf{w})\mathrm{d}\mathbf{w}$$

The corresponding probability for class $\mathcal{C}_2$ is given by

$$p(\mathcal{C}_2|\phi,\mathbf{t}) = 1 - p(\mathcal{C}_1|\phi,\mathbf{t})$$

Predictive Distribution

Note that

$$\sigma(\mathbf{w}^T\phi) = \int \delta(a - \mathbf{w}^T\phi)\sigma(a)\mathrm{d}a$$

where $\delta(\cdot)$ is the Dirac delta function. Therefore

$$\begin{aligned}\int \sigma(\mathbf{w}^T\phi)q(\mathbf{w})\mathrm{d}\mathbf{w} &= \int\int \delta(a - \mathbf{w}^T\phi)\sigma(a)q(\mathbf{w})\mathrm{d}\mathbf{w}\mathrm{d}a \\ &= \int \sigma(a)\int \delta(a - \mathbf{w}^T\phi)q(\mathbf{w})\mathrm{d}\mathbf{w}\mathrm{d}a \\ &= \int \sigma(a)p(a)\mathrm{d}a\end{aligned}$$

where

$$p(a) = \int \delta(a - \mathbf{w}^T\phi)q(\mathbf{w})\mathrm{d}\mathbf{w}$$

The function $\delta(a - \mathbf{w}^T\phi)$ can be approximated by a Gaussian

$$\delta(a - \mathbf{w}^T\phi) \approx \mathcal{N}(\phi^T\mathbf{w}, \alpha\mathbf{I}), \alpha \to \infty$$

Predictive Distribution

Therefore

$$\begin{aligned}\mu_a &= \phi^T\mathbf{w}_{\mathrm{MAP}} \\ \sigma_a^2 &= \phi^T\mathbf{S}_N\phi + \frac{1}{\alpha}\mathbf{I} = \phi^T\mathbf{S}_N\phi\end{aligned}$$

So the predictive distribution becomes

$$p(\mathcal{C}_1|\mathbf{t}) = \int \sigma(a)p(a)\mathrm{d}a = \int \sigma(a)\mathcal{N}(a|\mu_a,\sigma_a^2)\mathrm{d}a$$

The integration is difficult due to the logistic function $\sigma(\cdot)$. Instead, we can use the inverse *probit* function $\Phi(a)$ to approximate the *logistic* function

$$\Phi(a) = \int_{-\infty}^{a} \mathcal{N}(\theta|0,1)\mathrm{d}\theta$$

好了,最后的 Bayesian Logistic Regression 也完了。

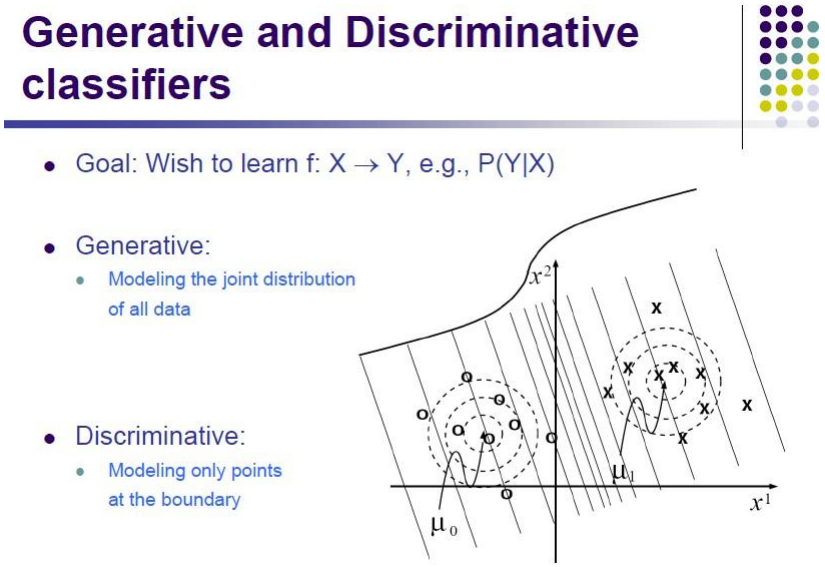
==========讨论==========

zeno(117127143) 21:25:06

没太明白 ，生成模型和判别模型的区别 ，优缺点呢 ？

planktonli(1027753147) 21:26:02

一个 model p(x,y) 生成式的，一个 model p(y|X)判别式的，判别式的只在乎 boundry 的这些点，生成式的需要知道这些点是怎么生成的：



zeno(117127143) 21:26:38

是判别式要好一点吗 ？

planktonli(1027753147) 21:27:34

生成模型: Naive Bayes, Graphical Model ，判别模型: NN,SVM,LDA,Decision Tree 等。不能说谁好谁不好 ,还有 生成模型 + 判别模型的 ,例如: SVM 的 kernel construcion 你可以用 generative 的方式去做，那就是 Generative + Discriminant 的了。

zeno(117127143) 21:30:38

生成式要求 X 互相独立吧 ？

planktonli(1027753147) 21:31:39

恩,生成式是需要 IID 的，这个在 statistics 上通常都有 IID 假设的，否则不好整，还有什么问题?

HX(458728037) 21:33:12

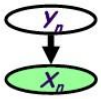
我想问一个简单的问题，其实生成模型最后做分类的时候不还是根据一个 boundary 来判断的吗？

planktonli(1027753147) 21:34:32

生成模型 在 two class 而且是 gaussian distribution 的时候，就可以转化为 判别式的，这个验证起来很简单，看看下面的 PPT 就明白了：

Gaussian Discriminative Analysis

- learning f: X → Y, where
  - X is a vector of real-valued features, $\mathbf{X}_n = < X_{n,1} \ldots X_{n,m} >$
  - Y is an indicator vector
- What does that imply about the form of P(Y|X)?



  - The joint probability of a datum and its label is:

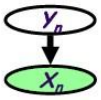
$$p(\mathbf{x}_n, y_n^k = 1 \mid \mu, \sigma) = p(y_n^k = 1) \times p(\mathbf{x}_n \mid y_n^k = 1, \mu, \sigma)$$
$$= \pi_k \frac{1}{(2\pi\sigma^2)^{1/2}} \exp\left\{-\frac{1}{2\sigma^2}(\mathbf{x}_n - \mu_k)^2\right\}$$

  - Given a datum $\mathbf{x}_n$, we predict its label using the conditional probability of the label given the datum:

$$p(y_n^k = 1 \mid \mathbf{x}_n, \mu, \sigma) = \frac{\pi_k \frac{1}{(2\pi\sigma^2)^{1/2}} \exp\left\{-\frac{1}{2\sigma^2}(\mathbf{x}_n - \mu_k)^2\right\}}{\sum_{k'} \pi_{k'} \frac{1}{(2\pi\sigma^2)^{1/2}} \exp\left\{-\frac{1}{2\sigma^2}(\mathbf{x}_n - \mu_{k'})^2\right\}}$$

Naïve Bayes Classifier

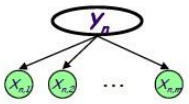
- When X is multivariate-Gaussian vector:



  - The joint probability of a datum and it label is:

$$p(\mathbf{x}_n, y_n^k = 1 \mid \vec{\mu}, \Sigma) = p(y_n^k = 1) \times p(\mathbf{x}_n \mid y_n^k = 1, \vec{\mu}, \Sigma)$$
$$= \pi_k \frac{1}{(2\pi|\Sigma|)^{1/2}} \exp\left\{-\tfrac{1}{2}(\mathbf{x}_n - \vec{\mu}_k)^T \Sigma^{-1} (\mathbf{x}_n - \vec{\mu}_k)\right\}$$

  - The naïve Bayes simplification



$$p(\mathbf{x}_n, y_n^k = 1 \mid \mu, \sigma) = p(y_n^k = 1) \times \prod_j p(x_{n,j} \mid y_n^k = 1, \mu_{k,j}, \sigma_{k,j})$$
$$= \pi_k \prod_j \frac{1}{(2\pi\sigma_{k,j}^2)^{1/2}} \exp\left\{-\frac{1}{2\sigma_{k,j}^2}(x_{n,j} - \mu_{k,j})^2\right\}$$

  - More generally: $p(\mathbf{x}_n, y_n \mid \eta, \pi) = p(y_n \mid \pi) \times \prod_{j=1}^{m} p(x_{n,j} \mid y_n, \eta)$
    - Where $p(. \mid .)$ is an arbitrary conditional (discrete or continuous) 1-D density

The predictive distribution

- Understanding the predictive distribution

$$p(y_n^k = 1 \mid x_n, \bar{\mu}, \Sigma, \pi) = \frac{p(y_n^k = 1, x_n \mid \bar{\mu}, \Sigma, \pi)}{p(x_n \mid \bar{\mu}, \Sigma)} = \frac{\pi_k N(x_n, \mid \mu_k, \Sigma_k)}{\sum_{k'} \pi_{k'} N(x_n, \mid \mu_{k'}, \Sigma_{k'})} \quad *$$

- Under naïve Bayes assumption:

$$p(y_n^k = 1 \mid x_n, \bar{\mu}, \Sigma, \pi) = \frac{\pi_k \exp\left\{-\sum_j \left(\frac{1}{2\sigma_{k,j}^2}(x_n^j - \mu_k^j)^2 - \log \sigma_{k,j} - C\right)\right\}}{\sum_{k'} \pi_{k'} \exp\left\{-\sum_j \left(\frac{1}{2\sigma_{k',j}^2}(x_n^j - \mu_{k'}^j)^2 - \log \sigma_{k',j} - C\right)\right\}} \quad **$$

- For two class (i.e., *K*=2), and when the two classes haves the same variance, ** turns out to be a logistic function

$$p(y_n^1 = 1 \mid x_n) = \frac{1}{1 + \frac{\pi_2 \exp\left\{-\sum_j \left(\frac{1}{2\sigma_j^2}(x_n^j - \mu_2^j)^2 - \log \sigma_j - C\right)\right\}}{\pi_1 \exp\left\{-\sum_j \left(\frac{1}{2\sigma_j^2}(x_n^j - \mu_1^j)^2 - \log \sigma_j - C\right)\right\}}} = \frac{1}{1 + \exp\left\{-\sum_j \left(x_n^j \frac{1}{\sigma_j^2}(\mu_1^j - \mu_2^j) + \frac{1}{\sigma_j^2}([\mu_1^j]^2 - [\mu_2^j]^2)\right) + \log \frac{(1-\pi_1)}{\pi_1}\right\}}$$

$$= \frac{1}{1 + e^{-\theta^T x_n}}$$

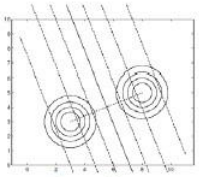
The decision boundary

- The predictive distribution

$$p(y_n^1 = 1 \mid x_n) = \frac{1}{1 + \exp\left\{-\sum_{j=1}^{M} \theta_j x_n^j - \theta_0\right\}} = \frac{1}{1 + e^{-\theta^T x_n}}$$

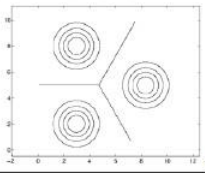
- The Bayes decision rule:

$$\ln \frac{p(y_n^1 = 1 \mid x_n)}{p(y_n^2 = 1 \mid x_n)} = \ln \left(\frac{\frac{1}{1 + e^{-\theta^T x_n}}}{\frac{e^{-\theta^T x_n}}{1 + e^{-\theta^T x_n}}} \right) = \theta^T x_n$$



- For multiple class (i.e., *K*>2), * correspond to a softmax function

$$p(y_n^k = 1 \mid x_n) = \frac{e^{-\theta_k^T x_n}}{\sum_j e^{-\theta_j^T x_n}}$$



19.048 厘米

zeno(117127143) 21:36:52

好的，谢谢，总是看见生成判别，弄不太清楚，关键是强调分清楚生成判别，对解决问题选那种工具有何影响，一直搞不清，我感觉 feature 不独立时用判别，feature 多时用判别，那么生成式有什么优势呢？

阳阳(236995728) 21:34:22

判别模型用的是频率学派的观点 ，也就是最大似然估计法， 生成模型用的是贝叶斯学派观点 ，也就是最大后验概率。

planktonli(1027753147) 21:37:10

阳阳，这个是不对的。MLE 也是 statistics 的东西，判别模型则根本不考虑 statistics，你可以看看 NN 的东西，包括今天的 LDA，它们直接求 boundry 的。

晴(498290717) 21:39:41

查了下别人的博客，我觉得这个是两者最最本质的区别

判别模型（Discriminative Model），又可以称为条件模型，或条件概率模型。估计的是条件概率分布

生成模型（Generative Model），又叫产生式模型。估计的是联合概率分布

ant/wxony(824976094) 21:42:18
产生式模型和判别式模型我觉得就是把人的知识用在模型的构建还是 feature 的设计上，特征多的时候一般还是判别式模型猛些。
HX(458728037) 21:44:51
生成式的模型在无监督的一些学习上应该才好用吧 ？
ant/wxony(824976094) 21:45:43
嗯，无监督学习上确实
planktonli(1027753147) 21:48:38
我把 Generative verses discriminative classifier 的一个 PPT 发到群里，大家有兴趣的看看:
"Lecture2.pdf" 下载