Math Basis

罗文水

2021 年 8 月 5 日

目录

1 Math Basis 01

本节的主要目的是从频率派的角度使用极大似然估计，通过观察到数据，是观察到的数据出现的概率最大化，来对高斯分布的参数进行估计。并且分析了高斯分布的参数，μ，σ^2 的无偏性和有偏性。其中，μ 是关于参数的无偏估计，而 σ 是有偏估计。

数据矩阵为：

$$X = (x_1, x_2, \cdots, x_N)^T = \begin{pmatrix} x_1^T \\ x_2^T \\ \vdots \\ x_N^T \end{pmatrix} = \begin{pmatrix} x_{11} & x_{12} & \dots & x_{1p} \\ x_{21} & x_{32} & \dots & x_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ x_{N1} & x_{N2} & \dots & x_{Np} \end{pmatrix}_{N \times p} \tag{1.1}$$

在数据矩阵的基础上，有 $x_i \in \mathbb{R}$，$x_i \sim \mathcal{N}(\mu, \Sigma)$，那么参数为 $\theta = \mu, \Sigma$。

1.1 求解目的

首先对于单变量的高斯分布 $\mathcal{N}(\mu, \sigma^2)$，概率密度函数为：

$$p(x) = \frac{1}{\sqrt{2\pi}\sigma} exp\left\{-\frac{(x-\mu)^2}{2\sigma^2}\right\} \tag{1.2}$$

对于多变量的高斯分布 $\mathcal{N}(\mu, \Sigma)$，概率密度函数为：

$$p(x) = \frac{1}{(2\pi)^{\frac{p}{2}}|\Sigma|^{\frac{1}{2}}} exp\left\{-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right\} \tag{1.3}$$

我们希望通过观察到的数据来计算参数 $\theta = (\mu, \Sigma)$ 的值，那么我们使用极大似然估计的优化目标为 $\theta_{MLE} = \arg\max_\theta p(X|\theta)$。于是我们可以转化为 $\theta_{MLE} = \arg\max_\theta \log p(X|\theta)$。那么，计算公式可以化简为：

$$\arg\max_\theta \log p(X|\theta) = \log \prod_{i=1}^{N} p(x_i|\theta) = \sum_{i=1}^{N} \log p(x_i|\theta) \tag{1.4}$$

$$= \sum_{i=1}^{N} \left(\log \frac{1}{\sqrt{2\pi}} + \log \frac{1}{\sigma} - \frac{(x-\mu)^2}{2\sigma^2}\right) \tag{1.5}$$

1.2 极大似然法求解参数 μ 和 σ^2

在求解 μ_{MLE} 时，计算目标为 $\frac{\partial \log p(x|\theta)}{\partial \mu}$，推导公式如下：

$$\frac{\partial \log p(x|\theta)}{\partial \mu} = \sum_{i=1}^{N} -\frac{(x_i - \mu)}{\sigma^2} = 0 \tag{1.6}$$

$$\sum_{i=1}^{N} x_i = \sum_{i=1}^{N} \mu \tag{1.7}$$

$$\mu_{MLE} = \frac{1}{N}\sum_{i=1}^{N} x_i \tag{1.8}$$

在求解 σ^2_{MLE} 时，计算目标为 $\frac{\partial \log p(x|\theta)}{\partial \sigma}$，推导公式如下：

$$\frac{\partial \log p(x|\theta)}{\partial \sigma} = \sum_{i=1}^{N} -\frac{1}{\sigma} - \frac{1}{2}(x_i - \mu)^2(-2)\sigma^{-3} = 0 \tag{1.9}$$

$$\sum_{i=1}^{N} \sigma^2 = \sum_{i=1}^{N}(x_i - \mu)^2 \tag{1.10}$$

$$\sigma^2_{MLE} = \frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2 \tag{1.11}$$

实际上这里的 μ 是 μ_{MLE}，所以，

$$\sigma^2_{MLE} = \frac{1}{N}\sum_{i=1}^{N}(x_i - \mu_{MLE})^2 \tag{1.12}$$

1.3 验证 μ_{MLE} 和 σ^2_{MLE} 的无偏性

首先需要明确什么是无偏估计，所谓无偏估计也就是，$\mathbb{E}(\hat{x}) = x$。那么利用这个性质我们就可以很方便的判断一个估计是否为无偏估计。

1.3.1 验证 μ_{MLE} 的无偏性

$$\begin{aligned}\mathbb{E}[\mu_{MLE}] =& \mathbb{E}[\frac{1}{N}\sum_{i=1}^{N} x_i] \\ =& \frac{1}{N}\sum_{i=1}^{N}\mathbb{E}[x_i] \\ =& \frac{1}{N}N\mu = \mu\end{aligned} \tag{1.13}$$

根据上述的推导，我们可以得出 μ_{MLE} 是无偏估计。

1.3.2 验证 σ^2_{MLE} 的无偏性

$$
\begin{aligned}
\mathbb{E}[\sigma^2_{MLE}] =&\mathbb{E}[\frac{1}{N}\sum_{i=1}^{N}(x_i-\mu_{MLE})^2] \\
=&\mathbb{E}[\frac{1}{N}\sum_{i=1}^{N}(x_i^2-2\mu_{MLE}x_i+\mu^2_{MLE})] \\
=&\mathbb{E}[\frac{1}{N}\sum_{i=1}^{N}(x_i^2-\mu^2_{MLE})] \\
=&\mathbb{E}[\frac{1}{N}\sum_{i=1}^{N}(x_i^2-\mu^2)-(\mu^2_{MLE}-\mu^2)] \\
=&\mathbb{E}[\frac{1}{N}\sum_{i=1}^{N}(x_i^2-\mu^2)]-\mathbb{E}[(\mu^2_{MLE}-\mu^2)] \\
=&\mathbb{E}[\frac{1}{N}\sum_{i=1}^{N}(x_i^2-(\frac{1}{N}\sum_{i=1}^{N}x_i)^2)]-\mathbb{E}[(\mu^2_{MLE}-\mu^2)] \\
=&\frac{1}{N}\sum_{i=1}^{N}(\mathbb{E}[x_i^2]-\mathbb{E}[x]^2)-\mathbb{E}[(\mu^2_{MLE}-\mu^2)] \\
=&\sigma^2-(\mathbb{E}[\mu^2_{MLE}]-\mathbb{E}[\mu^2]) \\
=&\sigma^2-(\mathbb{E}[\mu^2_{MLE}]-\mathbb{E}[\mathbb{E}[\mu_{MLE}]^2]) \\
=&\sigma^2-(\mathbb{E}[\mu^2_{MLE}]-\mathbb{E}[\mu_{MLE}]^2] \\
=&\sigma^2-Var[\mu_{MLE}] \\
=&\sigma^2-Var[\frac{1}{N}\sum_{i=1}^{N}x_i] \\
=&\sigma^2-\frac{1}{N^2}Var[\sum_{i=1}^{N}x_i] \\
=&\sigma^2-\frac{1}{N^2}\sum_{i=1}^{N}Var[x_i] \\
=&\sigma^2-\frac{1}{N^2}N\sigma^2 \\
=&\frac{N-1}{N}\sigma^2
\end{aligned}
\tag{1.14}
$$

有上述推导我们可以得出，σ^2_{MLE} 为有偏估计量，而且和真实值比较偏小。为什么会造成这个结果呢？主要原因是出在 μ_{MLE} 上，因为我们在求 σ^2_{MLE} 时使用的是 μ_{MLE} 而不是 μ。而 μ_{MLE} 是拟合数据得到的，所以波动的角度讲，肯定会比使用真实的 μ 算出来要小。所以在高斯分布中，利用极大似然估计得到的 σ^2_{MLE} 的值，是比真实值偏小的有偏估计。

2 Math Basis 02

本节的主要目的是从概率的角度来分析高斯分布，包括马氏距离和高斯分布的几何表示，以及高斯分布的局限性和解决的方法等等。对于多变量的高斯分布 $X \sim \mathcal{N}(\mu, \Sigma)$，概率密度函数为：

$$p(x) = \frac{1}{(2\pi)^{\frac{p}{2}}|\Sigma|^{\frac{1}{2}}} exp\left\{-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right\} \tag{2.1}$$

其中，$x \in \mathbb{R}^p$，

$$x = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_p \end{pmatrix} \qquad \Sigma = \begin{pmatrix} \sigma_{11} & \sigma_{12} & \cdots & \sigma_{1p} \\ \sigma_{21} & \sigma_{22} & \cdots & \sigma_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_{p1} & \sigma_{p2} & \cdots & \sigma_{pp} \end{pmatrix} \tag{2.2}$$

其中，Σ 为正定矩阵或者半正定矩阵。

2.1 什么是马氏距离

在高斯分布中，$\sqrt{(x-\mu)^T\Sigma^{-1}(x-\mu)}$ 的计算结果是一个数，这个数被称为马氏距离。设 $z_1 = (z_{11}, z_{12})^T$，$z_2 = (z_{21}, z_{22})^T$。那么 z_1 和 z_2 之间的马氏距离的平方为：

$$(z_1 - z_2)^T\Sigma^{-1}(z_1 - z_2) = \begin{pmatrix} z_{11} - z_{12} & z_{21} - z_{22} \end{pmatrix} \Sigma^{-1} \begin{pmatrix} z_{11} - z_{12} \\ z_{21} - z_{22} \end{pmatrix} \tag{2.3}$$

显然，当 $\Sigma^{-1} = I$ 时，马氏距离等于欧式距离 $(z_1 - z_2)^T\Sigma^{-1}(z_1 - z_2) = (z_{11} - z_{12})^2 + (z_{21} - z_{22})^2$。

2.2 对 $(x-\mu)^T\Sigma^{-1}(x-\mu)$ 的值进行推导

由于 Σ 为实对称矩阵，那么可以对 Σ 进行特征分解，那么有 $\Sigma = U\Lambda U^T$，并且 $UU^T = U^TU = I$，所以 $U^{-1} = U^T$，$\Lambda = diag(\lambda_i) \quad (i = 1, 2, \cdots, N)$，并且 $U = (u_1, u_2, \cdots, u_p)_{p\times p}$。

$$\Sigma = U\Lambda U^T \tag{2.4}$$

$$= (u_1, u_2, \cdots, u_p) \begin{pmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_p \end{pmatrix} \begin{pmatrix} u_1^T \\ u_2^T \\ \vdots \\ u_p^T \end{pmatrix} \tag{2.5}$$

$$= (u_1\lambda_1, u_2\lambda_2, \cdots, u_p\lambda_p) \begin{pmatrix} u_1^T \\ u_2^T \\ \vdots \\ u_p^T \end{pmatrix} \tag{2.6}$$

$$= \sum_{i=1}^{p} u_i\lambda_i u_i^T \tag{2.7}$$

而 Σ^{-1} 的求解过程如下所示：

$$\Sigma^{-1} = (U\Lambda U^T)^{-1} = (U^T)^{-1}\Lambda^{-1}U^{-1} = U\Lambda^{-1}U^T \tag{2.8}$$

代入可以解得：

$$\Sigma^{-1} = \sum_{i=1}^{p} u_i \frac{1}{\lambda_i} u_i^T \tag{2.9}$$

那么，

$$(x-\mu)^T\Sigma^{-1}(x-\mu) = (x-\mu)^T \sum_{i=1}^{p} u_i \frac{1}{\lambda_i} u_i^T (x-\mu) \tag{2.10}$$

$$= \sum_{i=1}^{p} (x-\mu)^T u_i \frac{1}{\lambda_i} u_i^T (x-\mu) \tag{2.11}$$

令 $y_i = (x-\mu)^T u_i$，这是一个典型的投影算法，其中 u_i 是 Σ 的特征值为 λ_i 的特征向量，那么

$$(x-\mu)^T\Sigma^{-1}(x-\mu) = \sum_{i=1}^{p} y_i \frac{1}{\lambda_i} y_i^T = \sum_{i=1}^{p} \frac{y_i^2}{\lambda_i} \tag{2.12}$$

2.3 高斯分布的几何意义

如何令 $p=2$，则有 $\Delta = \frac{y_1^2}{\lambda_1} + \frac{y_2^2}{\lambda_2}$，这实际上就是一个椭圆，如果 Δ 取不同的值，就会像等高线一样，一圈圈的环绕，因为每一个 Δ 都对应着一个二次型，即对应着概率密度函数的一个取值，同样的 Δ 取值对应着同样的概率密度。那么最终的概率密度等高线表示为：

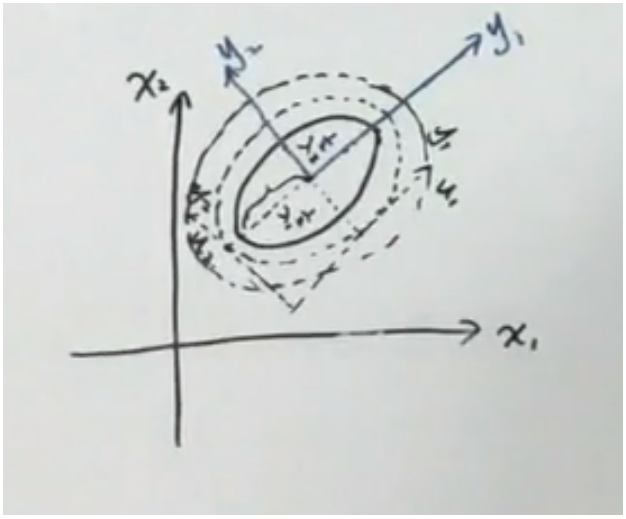

图 1: 二维高斯分布的可视化表示图

2.4 高斯分布中遇到的困难

2.4.1 维度灾难 (curse of dimension)

由于 Σ 是一个 $p \times p$ 的矩阵，矩阵中一共有 $\frac{p(p+1)}{2}$ 个参数，算法的复杂度为 $O(p^2)$。一旦输入维度过大，这个矩阵的计算会变得很复杂。所以，在某些时候，将 Σ 矩阵简化成对角矩阵将算法复杂度

降低到 $O(p)$。进一步简化，可以令 $\lambda_1 = \lambda_2 = \cdots = \lambda_p$。这时，高斯分布的可视化表示即为一个中心在原点的同心圆。这样的高斯分布，被我们称为“各向同性”的高斯分布。

2.4.2 高斯分布的表达的局限性

很多时候，高斯分布的表达有局限性，这时，学者提出了混合高斯模型 (GMM) 来拟合现实情况中复杂多样的分布。具体有关于混合高斯模型 (GMM) 的内容将在后续部分进行详细解释，

3 Math Basis 03

本节的主要内容是在高斯分布中，已知联合概率密度求条件概率密度和边缘概率密度。还有已知 x 的边缘概率和在条件为 x 下 y 的条件概率下，求 y 的边缘概率和在条件为 y 下 x 的条件概率。用数学的语言描述即为，如下所示。

对于多变量的高斯分布 $x \sim \mathcal{N}(\mu, \Sigma)$，概率密度函数为：

$$p(x) = \frac{1}{(2\pi)^{\frac{p}{2}}|\Sigma|^{\frac{1}{2}}} exp\left\{-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right\} \tag{3.1}$$

其中，$x \in \mathbb{R}^p$，

$$x = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_p \end{pmatrix} \qquad \Sigma = \begin{pmatrix} \sigma_{11} & \sigma_{12} & \cdots & \sigma_{1p} \\ \sigma_{21} & \sigma_{22} & \cdots & \sigma_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_{p1} & \sigma_{p2} & \cdots & \sigma_{pp} \end{pmatrix}_{p\times p} \tag{3.2}$$

已知联合概率密度求条件概率密度和边缘概率密度，可描述为已知：

$$x = \begin{pmatrix} x_a \\ x_b \end{pmatrix} \quad m+n=p \quad \mu = \begin{pmatrix} \mu_a \\ \mu_b \end{pmatrix} \quad \Sigma = \begin{pmatrix} \Sigma_{aa} & \Sigma_{ab} \\ \Sigma_{ba} & \Sigma_{bb} \end{pmatrix} \tag{3.3}$$

求：$p(x_a)$ 和 $p(x_b|x_a)$

另一个可描述为，已知：$p(x) = \mathcal{N}(x|\mu, \Lambda^{-1})$，$p(y|x) = \mathcal{N}(Ax+b, L^{-1})$，求 $p(y)$ 和 $p(x|y)$。

3.1 已知联合概率密度求条件概率密度和边缘概率密度

在进行此次推导之前，首先需要引入一个推论。已知：

$$x \sim \mathcal{N}(\mu, \Sigma) \tag{3.4}$$

$$y = Ax + b \tag{3.5}$$

那么可以解得：$y \sim \mathcal{N}(A\mu + b, A\Sigma A^T)$。

3.1.1 求解边缘概率密度 $p(x_a)$

$$x_a = (I_m, 0)\begin{pmatrix} x_a \\ x_b \end{pmatrix} \tag{3.6}$$

很显然，等式的第一部分就是 A，等式的第二部分就是 X，那么

$$\mathbb{E}[x_a] = (I_m, 0)\begin{pmatrix}\mu_a \\ \mu_b\end{pmatrix} = \mu_a \qquad Var[x_a] = (I_m, 0)\begin{pmatrix}\Sigma_{aa} & \Sigma_{ab} \\ \Sigma_{ba} & \Sigma_{bb}\end{pmatrix}\begin{pmatrix}I_m \\ 0\end{pmatrix} = \Sigma_{aa} \tag{3.7}$$

所以，$x_a \sim \mathcal{N}(\mu_a, \Sigma_{aa})$，同理 $x_b \sim \mathcal{N}(\mu_b, \Sigma_{bb})$。

3.1.2 求解条件概率密度 $p(x_b|x_a)$

这里使用的是构造证明法，有一点猜出了结论，再使用结论来证结论的感觉。在这里引入了一个构造函数 $x_{b\cdot a} = x_b - \Sigma_{ba}\Sigma_{aa}^{-1}x_a$。那么根据我们的推论，我们可以很简单的得出以下的构造方法

$$\mu_{b\cdot a} = \mu_b - \Sigma_{ba}\Sigma_{aa}^{-1}\mu_a \tag{3.8}$$

$$\Sigma_{bb\cdot a} = \Sigma_{bb} - \Sigma_{ba}\Sigma_{aa}^{-1}\Sigma_{ab} \tag{3.9}$$

其中 $\Sigma_{aa\cdot b}$ 被称为 Σ_{aa} 的 Schur Complementary。我们接下来可以很简单的将 $x_{b\cdot a}$ 化简成 $Ax+b$ 的性质。

$$x_{b\cdot a} = (-\Sigma_{ba}\Sigma_{aa}^{-1} \quad I_m)\begin{pmatrix}x_a \\ x_b\end{pmatrix} \tag{3.10}$$

那么我们进行以下的推导，

$$\mathbb{E}[x_{b\cdot a}] = (-\Sigma_{ba}\Sigma_{aa}^{-1} \quad I_m)\begin{pmatrix}\mu_a \\ \mu_b\end{pmatrix} = \mu_b - \Sigma_{ba}\Sigma_{aa}^{-1}\mu_a \tag{3.11}$$

$$Var[x_{b\cdot a}] = (-\Sigma_{ba}\Sigma_{aa}^{-1} \quad I_m)\begin{pmatrix}\Sigma_{aa} & \Sigma_{ab} \\ \Sigma_{ba} & \Sigma_{bb}\end{pmatrix}\begin{pmatrix}-\Sigma_{aa}^{-1^T}\Sigma_{ba}^T \\ I_m\end{pmatrix} \tag{3.12}$$

计算得出，$Var[x_{b\cdot a}] = \Sigma_{bb} - \Sigma_{ba}\Sigma_{aa}^{-1}\Sigma_{ab}$。所以，$p(x_{b\cdot a}) = \mathcal{N}(\mu_{b\cdot a}, \Sigma_{bb\cdot a})$。

又因为 $x_b = x_{b\cdot a} + \Sigma_{ba}\Sigma_{aa}^{-1}x_a$，$x_b$ 是一个关于 x_a 的线性表达。那么后续的求解过程将变得非常的简单了。

$$\mathbb{E}[x_b|x_a] = \mathbb{E}[x_{b\cdot a}] + \Sigma_{ba}\Sigma_{aa}^{-1}x_a = \mu_b + \Sigma_{ba}\Sigma_{aa}^{-1}(x_a - \mu_a) \tag{3.13}$$

而 $Var[x_b|x_a]$ 中

$$Var[x_b|x_a] = \Sigma_{bb\cdot a} = \Sigma_{bb} - \Sigma_{ba}\Sigma_{aa}^{-1}\Sigma_{ab} \tag{3.14}$$

于是综上所述，$x_b|x_a \sim \mathcal{N}(\mu_b + \Sigma_{ba}\Sigma_{aa}^{-1}(x_a - \mu_a), \Sigma_{bb} - \Sigma_{ba}\Sigma_{aa}^{-1}\Sigma_{ab})$。

同理可得 $x_a|x_b \sim \mathcal{N}(\mu_a + \Sigma_{ab}\Sigma_{bb}^{-1}(x_b - \mu_b), \Sigma_{aa} - \Sigma_{ab}\Sigma_{bb}^{-1}\Sigma_{ba})$。

3.2 边缘高斯和条件高斯

问题可以被描述为，$p(x) = \mathcal{N}(x|\mu, \Lambda^{-1})$，$p(y|x) = \mathcal{N}(Ax+b, L^{-1})$，求 $p(y)$ 和 $p(x|y)$。

这个高斯分布的公式的推导非常的重要，特别是在 Linear Gaussian Model 中被大量的运用。同时也在贝叶斯公式中有着大量的运用，所以某种意义上说，这个公式的运用比上一个公式更加的重要。

3.2.1 边缘高斯

根据已知条件，我们可以设 $y = Ax + b + \varepsilon$，其中 $\varepsilon \sim \mathcal{N}(0, L^{-1})$。

$$\mathbb{E}[y] = \mathbb{E}[Ax + b + \varepsilon] = \mathbb{E}[Ax + b] + \mathbb{E}[\varepsilon] = A\mu + b \tag{3.15}$$

$$Var[y] = Var[Ax + b + \varepsilon] = Var[Ax] + Var[\varepsilon] = A\Lambda^{-1}A^T + L^{-1} \tag{3.16}$$

所以综上所述，$p(y) \sim \mathcal{N}(A\mu + b, A\Lambda^{-1}A^T + L^{-1})$。

3.2.2 条件高斯

根据上述的推导我们已经知道的条件有，$p(y)$，$p(x)$，$p(y|x)$ 那么我们可以如何求得 $p(x|y)$ 呢? 很显然我们还差一个 x 和 y 的联合分布。如果知道 x 和 y 的联合分布，那么我们就和上一节已知联合概率密度求条件概率密度和边缘概率密度的内容接起来了。

设

$$z = \begin{pmatrix} x \\ y \end{pmatrix} \sim \mathcal{N}\left(\begin{pmatrix} \mu \\ A\mu + b \end{pmatrix} \quad \begin{pmatrix} \Lambda^{-1} & \Delta \\ \Delta^T & A\Lambda^{-1}A^T + L^{-1} \end{pmatrix} \right) \tag{3.17}$$

这里的 Δ 是 x 和 y 的协方差矩阵，我们可以直接利用协方差的定义来进行求解。

$$\begin{aligned} \Delta =& Cov(x, y) = \mathbb{E}[(x - \mathbb{E}[x])(y - \mathbb{E}[y])^T] \\ =& \mathbb{E}[(x - \mu)(y - (A\mu + b))^T] \\ =& \mathbb{E}[(x - \mu)(Ax + \varepsilon - A\mu)^T] \\ =& \mathbb{E}[(x - \mu)(Ax - A\mu + \varepsilon)^T] \\ =& \mathbb{E}[(x - \mu)(Ax - A\mu)^T + ((x - \mu)\varepsilon)^T] \\ =& \mathbb{E}[(x - \mu)(Ax - A\mu)^T] + \mathbb{E}[(x - \mu)\varepsilon^T] \end{aligned} \tag{3.18}$$

$\mathbb{E}[(x - \mu)\varepsilon^T] = \mathbb{E}[(x - \mu)]\mathbb{E}[\varepsilon^{\mathbb{T}}] = 0$。那么，推导继续：

$$\begin{aligned} \Delta =& \mathbb{E}[(x - \mu)(Ax - A\mu)^T] \\ =& \mathbb{E}[(x - \mu)(x - \mu)^T] \cdot A^T \\ =& Var[x] \cdot A^T \\ =& \Lambda^{-1}A^T \end{aligned} \tag{3.19}$$

所以，x 和 y 之间的联合概率分布可表达为，

$$z = \begin{pmatrix} x \\ y \end{pmatrix} \sim \mathcal{N}\left(\begin{pmatrix} \mu \\ A\mu + b \end{pmatrix}, \begin{pmatrix} \Lambda^{-1} & \Lambda^{-1}A^T \\ A\Lambda^{-1} & A\Lambda^{-1}A^T + L^{-1} \end{pmatrix} \right) \tag{3.20}$$

通过上述的推导，我们可以成功的得到 x 和 y 的联合概率密度分布。那么利用上节得出的公式，就可以成功的推导出 $p(x|y)$。代入上述公式中，描述的结果如下：

$$x|y \sim \mathcal{N}\left(\mu + \Lambda^{-1}A^T K^{-1} A\Lambda^{-1}\left(y - A\mu - b\right), \Lambda^{-1} - \Lambda^{-1}A^T K^{-1} A\Lambda^{-1}\right) \tag{3.21}$$

其中，

$$K = A\Lambda^{-1}A^T + L^{-1} \tag{3.22}$$

4 Math Basis 04

本节主要的内容是描述琴生不等式 (Jensen's Inequality)。有关琴生不等式的描述为，如果函数 $f(x)$ 为凸函数 (convex function)，那么有 $\mathbb{E}[f(x)] \geq f(\mathbb{E}[x])$。

4.1 Jensen's Inequality 中的证明

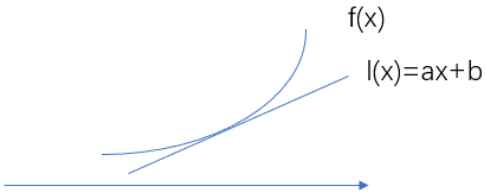

图 2: 函数和它在某点的切线的表达式

设切点的横坐标为 $\mathbb{E}[x]$,那么 $f(\mathbb{E}[x]) = L(x) = a\mathbb{E}[x]+b$。又因为 function 为 convex function。那么很显然, 对于 $\forall x$ 都有 $f(x) \geq L(x)$。然后, 我们同时对不等式两边求期望, 可以得到 $\mathbb{E}[f(x)] \geq \mathbb{E}[L(x)]$。那么我们进行如下的推导:

$$\begin{aligned} \mathbb{E}[f(x)] \geq & \mathbb{E}[L(x)] \\ = & \mathbb{E}[a\mathbb{E}[x] + b] \\ = & a\mathbb{E}[x] + b \\ = & f(\mathbb{E}[x]) \end{aligned} \tag{4.1}$$

可以很简单的得出结论。

Linear Regression 01

Chen Gong

12 October 2019

数据集 $D = \{(x_1, y_1), (x_2, y_2), \cdots, (x_N, y_N)\}$，其中 $x_i \in \mathbb{R}^p$，$y_i \in \mathbb{R}$，$i = 1,\ 2, \cdots,\ N$。
数据矩阵为：(这样可以保证每一行为一个数据点)

$$X = (x_1, x_2, \cdots, x_N)^T = \begin{pmatrix} x_1^T \\ x_2^T \\ \vdots \\ x_N^T \end{pmatrix} = \begin{pmatrix} x_{11} & x_{12} & \dots & x_{1p} \\ x_{21} & x_{32} & \dots & x_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ x_{N1} & x_{N2} & \dots & x_{Np} \end{pmatrix}_{N \times p} \tag{0.1}$$

$$Y = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{pmatrix}_{N \times 1} \tag{0.2}$$

设拟合的函数为：$f(w) = w^T x$。

1 最小二乘估计：矩阵表示

很简单可以得到损失函数 (Loss function) 为：

$$L(w) = \sum_{i=1}^{N} ||w^T x_i - y_i||^2 \tag{1.1}$$

$$= (w^T x_1 - y_1, w^T x_2 - y_2, \dots, w^T x_N - y_N) \begin{pmatrix} w^T x_1 - y_1 \\ w^T x_2 - y_2 \\ \vdots \\ w^T x_N - y_N \end{pmatrix} \tag{1.2}$$

其中:

$$\begin{aligned} (w^T x_1 - y_1, w^T x_2 - y_2, \dots, w^T x_N - y_N) &= [(w^T x_1, w^T x_2, \cdots, w^T x_N) - (y_1, y_2, \cdots, y_N)] \\ &= w^T X^T - Y^T \end{aligned} \tag{1.3}$$

所以:

$$\begin{aligned} L(w) &= (Xw - Y)^T (Xw - Y) \\ &= w^T X^T X w - 2 w^T X^T Y + Y^T Y \end{aligned} \tag{1.4}$$

由于 X^TX 是一个半正定矩阵,$L(w)$ 是一个凸函数,那么我需要求的 w,可记为 $\hat{w} = \arg\min_w\ L(w)$。这是一个无约束优化问题，可以通过求偏导解决。那么有：

$$\frac{\partial L(w)}{\partial w} = 2X^TXw - 2X^TY = 0 \tag{1.5}$$

解得：

$$\hat{w} = (X^TX)^{-1}X^TY \tag{1.6}$$

2 最小二乘估计：几何意义

将 X 矩阵从列向量的角度来看，可以看成一个 p 维的向量空间 S，为了简便计算，令 $w^TX = X\beta$。可以看成 Y 向量到 S 的距离最短，那么将有约束条件：

$$X^T(Y - X\beta) = 0 \tag{2.1}$$

$$X^TY - X^TX\beta = 0 \tag{2.2}$$

$$\beta = (X^TX)^{-1}X^TY \tag{2.3}$$

3 最小二乘估计：概率角度

假设一个分布 $\varepsilon \sim \mathcal{N}(0, \sigma^2)$，那么所有的观测值可看为 $y = w^Tx + \varepsilon$。因为 $\varepsilon \sim \mathcal{N}(0, \sigma^2)$，那么 $p(y|x; w) \sim \mathcal{N}(w^Tx, \sigma^2)$。我们的目的是求 w 使，y 出现的概率最大，在这里可以使用极大似然估计 (MLE) 求解。首先写出 $p(y|x; w)$ 的概率密度函数为：

$$p(y|x; w) = \frac{1}{\sqrt{2\pi}\sigma} exp\left(-\frac{(y - w^Tx)^2}{2\sigma^2}\right) \tag{3.1}$$

对数似然函数为 $\log\ p(Y|X; w)$，使似然函数最大化的过程求解如下：

$$L(w) = \log\ p(Y|X; w) = \log \prod_{i=1}^{N} p(y_i|x_i; w) \tag{3.2}$$

$$= \sum_{i=1}^{N} \log\ p(y_i|x_i; w) \tag{3.3}$$

$$= \sum_{i=1}^{N} \left(\log \frac{1}{\sqrt{2\pi}\sigma} + \log\ exp\left(-\frac{(y_i - w^Tx)^2}{2\sigma^2}\right)\right) \tag{3.4}$$

求解目标为 $\hat{w} = \arg\max_w\ L(w)$，因为第一项其中并没有包含 w，于是可以直接省略，那么有：

$$\hat{w} = \arg\max_w\ L(w) \tag{3.5}$$

$$= \arg\max_w\ \sum_{i=1}^{N} -\frac{(y_i - w^Tx_i)^2}{2\sigma^2}$$

$$= \arg\min_w\ \sum_{i=1}^{N} (y_i - w^Tx_i)^2$$

$$\tag{3.6}$$

那么我可以可以得到一个结论：最小二乘估计 $\Longleftrightarrow$ 极大似然估计 (噪声符合高斯分布)。最小二乘估计中隐藏了一个假设条件，那就是噪声符合高斯分布。

Linear Regression 02

Chen Gong

14 October 2019

1 正则化概述

过拟合问题 (over-fitting) 问题是深度学习中一个很重要的问题，往往是由少量的数据拟合高维的向量所造成的。解决 over-fitting 的方法有很多，通常是使用这几种思路：1. 增加数据量；2. 特征选择/特征提取 (PCA)；3. 增加正则项的方法。

正则项通常可以描述为 Loss Function + Penalty，也就是 $L(w)+\lambda P(w)$。正则化的方法通常有以下两种：

1. Lasso，其中 $P(w)=||w||_1=\sum_{i=1}^{N}|w_i|$，LASSO 回归等价于最小二乘回归加上 $||w||_1<\varepsilon$ 条件，也就是将其中的每个维度都尽量压缩到 0，使得系数稀疏化。
2. Redge，岭回归，也就是 $P(w)=||w||_2^2=\sum_{i=1}^{N}w_i^2$，等价于最小二乘回归加上了 $w^Tw<\varepsilon$ 条件，也就是让系数之间相差不会太大。

2 岭回归频率派角度

Loss function 可写为 $L(w)=\sum_{i=1}^{N}||w^Tx_i-y_i||^2+\lambda w^Tw$

$$\begin{aligned}J(w)=&\sum_{i=1}^{N}||w^Tx_i-y_i||^2+\lambda w^Tw \\ =&(w^TX^T-Y^T)(Xw-Y)+\lambda w^Tw \\ =&w^TX^TXw-2W^TX^TY-Y^TY+\lambda w^Tw \\ =&w^T(X^TX+\lambda I)w-2w^TX^TY-Y^TY\end{aligned} \tag{1}$$

我们的求解目标是 $\hat{w}=\arg\min_w J(w)$，求解过程为：

$$\frac{\partial J(w)}{\partial w}=2(X^TX+\lambda I)W-2X^TY=0 \tag{2}$$

解得：

$$W=(X^TX+\lambda I)^{-1}X^TY \tag{3}$$

根据以上的推导我们可以得出，首先 $(X^TX+\lambda I)$ 一定是可逆的。因为，半正定矩阵 + 单位矩阵 = 正定矩阵。这里不需要再求伪逆了。

3 岭回归贝叶斯派估计角度

类似于前文提到的贝叶斯回归的角度，假设一个分布 $\varepsilon \sim \mathcal{N}(0,\sigma^2)$，那么所有的观测值可看为 $y = w^T x + \varepsilon$。因为 $\varepsilon \sim \mathcal{N}(0,\sigma^2)$，那么 $p(y|x;w) \sim \mathcal{N}(w^T x, \sigma^2)$。假设 w 符合一个先验分布 $\mathcal{N}(0,\sigma_0^2)$。于是，我们可以得到 $p(w)$ 和 $p(y|w)$ 的解析表达式:

$$p(y|w) = \frac{1}{\sqrt{2\pi}\sigma} exp\left(-\frac{(y-w^T x)^2}{2\sigma^2}\right) \tag{4}$$

$$p(w) = \frac{1}{\sqrt{2\pi}\sigma_0} exp\left(-\frac{w^T w}{2\sigma_0^2}\right) \tag{5}$$

我们的目标是求 w 的最大后验估计 (MAP)，也就是定义为求 $\hat{w} = argmax_w p(w|y)$。由于

$$p(w|y) = \frac{p(y|w)p(w)}{p(y)} \tag{6}$$

但是 y 是我们的观察量，所以 $p(y)$ 是一个常量，在求解优化问题的时候可以不考虑进来。而且，可以加入 log 函数来简化运算，而且与计算结果无关，于是问题变成了求解如下的无约束优化问题:

$$\arg\max_w p(w|y) = \log p(y|w)p(w) \tag{7}$$

代入可得:

$$argmax_w p(w|y) = \sum_{i=1}^{N} \log \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y_i - w^T x_i)^2}{2\sigma^2}\right) \frac{1}{\sqrt{2\pi}\sigma_0} \exp\left(-\frac{||w||^2}{2\sigma_0^2}\right) \tag{8}$$

$$= \sum_{i=1}^{N} \log \frac{1}{2\pi\sigma\sigma_0} \exp\left(-\frac{(y_i - w^T x_i)^2}{2\sigma^2} - \frac{||w||^2}{2\sigma_0^2}\right) \tag{9}$$

$$= \sum_{i=1}^{N} \log \frac{1}{2\pi\sigma\sigma_0} + \log \exp\left(-\frac{(y_i - w^T x_i)^2}{2\sigma^2} - \frac{||w||^2}{2\sigma_0^2}\right) \tag{10}$$

由于 $\log \frac{1}{2\pi\sigma\sigma_0}$ 与求解无关，所以优化问题等价于:

$$\arg\max_w p(w|y) = \sum_{i=1}^{N} \log \exp\left(-\frac{(y_i - w^T x_i)^2}{2\sigma^2} - \frac{||w||^2}{2\sigma_0^2}\right) \tag{11}$$

$$= \sum_{i=1}^{N} -\frac{(y_i - w^T x_i)^2}{2\sigma^2} - \frac{||w||^2}{2\sigma_0^2} \tag{12}$$

$$\tag{13}$$

公式可以转化为:

$$\arg\min_w p(w|y) = \sum_{i=1}^{N} (y_i - w^T x_i)^2 + \frac{\sigma^2}{\sigma_0^2}||w||^2 \tag{14}$$

然后我们惊奇的发现将 $\frac{\sigma^2}{\sigma_0^2}$ 换成 λ 就又变成了和之前从频率角度看岭回归一样的结果。所以，对于上节我们得出的结论: **最小二乘估计 $\Longleftrightarrow$ 极大似然估计 (噪声符合高斯分布)。那么我们的最小二乘估计中隐藏了一个假设条件，那就是噪声符合高斯分布。我们进一步补充可得，Regularized LSE(L_2 范数正则化) 可以等价为最大后验估计 (MAP) 其中噪声为 Guassian Distribution，并且 w 的先验也为 Guassian Distribution**。

Linear Classification 01 Introduction

Chen Gong

29 October 2019

本节的主要目的是，有关于机器学习的导图。对频率派的有关统计学习方法做一个大致的梳理。而在贝叶斯派的学习中，是使用有关于概率图的模型。在频率派的有关统计学习方法中，我们可以大致的分为，线性回归和线性分类。

1 线性回归

在前文中已经提到了，我们的线性回归模型可以写为 $f(w,b) = w^T x + b$。线性回归主要有三条性质：线性，全局性和数据未加工。而我们从每一条入手，打破其中的一条规则就是一个新的算法。

1.1 线性

线性可以分为，属性非线性，全局非线性和系数非线性。

1.1.1 属性非线性

所谓的属性非线性也就是从未知数入手，比如特征变换的方法还有将变量从一维，变换到高维。有点类似于引入二次型的思想，使用 $x_1^2 + x_2^2 + x_1 x_2 + \cdots$，的方法打破属性的线性。

1.1.2 全局非线性

全局非线性的方法，是通过对函数的运算结果增加一个函数，来将线性函数改造成非线性函数。比如，神经网络中的激活函数，还有阈值函数来将软分类函数变成硬分类函数。

1.1.3 系数非线性

所谓系数非线性，感觉就是系数的生成结果并不是单一的，固定的。就像神经网络算法一样。算法的收敛结果是一个分布，也就是位于一个区间之中，这样的算法的结果一定不是线性的，这样通过了不确定的方法来引入非线性。

1.2 全局性

所谓全局性，也就是将所有的数据看成一个整体来进行拟合。而打破的方法很简单，也就是将数据之间分隔开，分段进行拟合。典型的方法有线性样条回归，决策树等方法。

1.3 数据未加工

从字面的意义上理解非常的简单，那就是输入数据不经过加工直接的输入模型中。有一系列类似的方法来打破，比如主成分分析法 (PCA)，流形等方法来对输入数据进行预处理。

2 线性分类

线性回归和线性分类之间有着很大的联系。从某种意义上说，线性分类就是线性回归函数使用激活函数的结果，同时也可以看成是线性回归降维的结果。对于一个线性回归函数，我们可以通过添加全局函数的形式来将其转换为线性分类函数。也就是

$$y = w^T x + b \longrightarrow y = f(w^T x + b) \tag{1}$$

这样就可以将值域从 $[0,1]$ 转换为 $\{0,1\}$。其中 f 被定义为 activation function，f^{-1} 定义为 link function。那么这个 f 实现了这样一个功能，也就是将 $w^T x + b \mapsto \{0,1\}$。而 f^{-1} 恰好是反过来的，也就是将 $\{0,1\} \mapsto w^T x + b$。

而线性分类，大致上可以划分成硬分类和软分类两个部分。

2.1 硬分类

所谓硬分类，也就是 $y \in [0,1]$，大致上可以分成线性判别分析，也就是 Fisher 判别分析和感知机这两类。

2.2 软分类

所谓硬分类，也就是 $y \in \{0,1\}$，大致上可以分成生成式模型，Gaussian Discriminate Analysis 和著名的判别式模型，Logistic Regression。

$$p(y|x) = \frac{p(x|y)p(y)}{p(x)} \propto p(x|y)p(y) \tag{2}$$

也就是在求解 $p(y=0|x)$ 或 $p(y=1|x)$ 的时候，我们不直接求谁大谁小，而是转向求 $p(x|y=0)p(y=0)$ 和 $p(x|y=1)p(y=1)$，即求联合概率。

3 总结

通过这节的学习，我们已经大体上建立了有关于统计学习方法的知识的框架，包括线性分类和线性回归的内容，并作出了一定的梳理。

Linear Classification 02 Perceptron

Chen Gong

30 October 2019

本节的主要内容是描述两类硬分类模型，也就是感知机模型和线性判别模型 (Fisher 判别模型) 的算法原理和推导过程。

1 感知机模型

感知机模型是一类错误驱动的模型，它的中心思想也就是" 错误驱动"。什么意思呢？也就是哪些数据点分类错误了，那么我们就进行调整权值系数 w，直到分类正确为止。

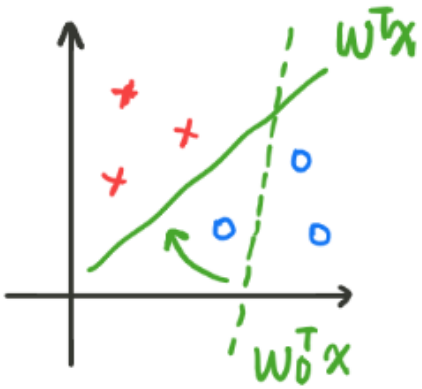

图 1: 感知机概念模型图

感知机可以做如下的描述：

$$f(x) = sign\{w^T x\} \quad x \in \mathbb{R}^p \; w \in \mathbb{R}^p \tag{1}$$

$$sign(a) = \begin{cases} +1 & a \geq 0 \\ -1 & a < 0 \end{cases} \tag{2}$$

其中 D : { 被错误分类的样本 }，样本集为：$\{(x_i, y_i)\}_{i=1}^N$。

1.1 感知机模型的迭代过程

我们将损失函数定义为：

$$\mathcal{L}(w) = \sum_{i=1}^{N} I\left\{y_i w^T x_i < 0\right\} \tag{3}$$

而其中 $y_i w^T x_i < 0$ 就代表分类错误的类，为什么这么理解呢？因为：

$$\begin{cases} w^T x_i \geq 0 & y_i = +1 \\ w^T x_i < 0 & y_i = -1 \end{cases} \tag{4}$$

那么当分类正确时，必然有 $w^T x_i y_i > 0$。只有当错误分类的时候，才会出现 $w^T x_i y_i < 0$ 的情况。而在上述的函数中，I 干了一个什么事，那就是将函数的值离散化，令 $\mathcal{L}$ 的值等于错误分类的点的个数，也就是这样一个映射 $I \mapsto 0, 1$。加这个函数的目的是得到损失函数的值，和普通的梯度下降法的过程一样。显然这不是一个连续的函数，无法求得其梯度来进行迭代更新。那么，我们需要想的办法是将离散的梯度连续。那么，我们将损失函数改写为：

$$\mathcal{L}(w) = \sum_{x_i \in D} -y_i w^T x_i \tag{5}$$

那么，梯度可以表示为:

$$\nabla_w \mathcal{L} = -\sum_{x_i \in D} y_i x_i \tag{6}$$

很显然，有关于 w 的迭代公式，可以表示为：

$$w^{(t+1)} \longleftrightarrow w^{(t)} - \lambda \nabla_w L \tag{7}$$

代入可得，权值参数 w 的更新过程为：

$$w^{(t+1)} \longleftrightarrow w^{(t)} + \lambda \sum_{x_i \in D} y_i x_i \tag{8}$$

那么，通过上述的推导，我们就得到了感知机中 w 的更新过程。那么，感知机算法的推导过程就已经完成了。

Linear classification 03 LDA

Chen Gong

31 October 2019

本小节为线性分类的第三小节，主要推导了线性判别分析算法，也就是 Fisher 算法。Fisher 算法的主要思想是：类内小，类间大。这有点类似于，软件过程里的松耦合，高内聚的思想。这个思想转换成数学语言也就是，同一类数据之间的方差要小，不同类数据之间的均值的差距要大。那么，我们对数据的描述如下所示：

$$X = (x_1, x_2, \cdots, x_N)^T = \begin{pmatrix} x_1^T \\ x_2^T \\ \vdots \\ x_N^T \end{pmatrix} = \begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1p} \\ x_{21} & x_{32} & \cdots & x_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ x_{N1} & x_{N2} & \cdots & x_{Np} \end{pmatrix}_{N \times P} \tag{1}$$

$$Y = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{pmatrix}_{N \times 1} \tag{2}$$

那么，我们的数据集可以记为 $\{(x_i, y_i)\}_{i=1}^N$，其中，$x_i \in \mathbb{R}^p$，$y_i \in \{+1, -1\}$，且 $\{y = +1\}$ 为 C_1 类，且 $\{y = -1\}$ 为 C_2 类。那么，X_{c_1} 被定义为 $\{x_i | y_i = +1\}$，X_{c_2} 被定义为 $\{x_i | y_i = -1\}$。所以，很显然可以得到 $|X_{c_1}| = N_1$，$|X_{c_2}| = N_2$，并且 $N_1 + N_2 = N$。

1 Fisher 线性判别分析

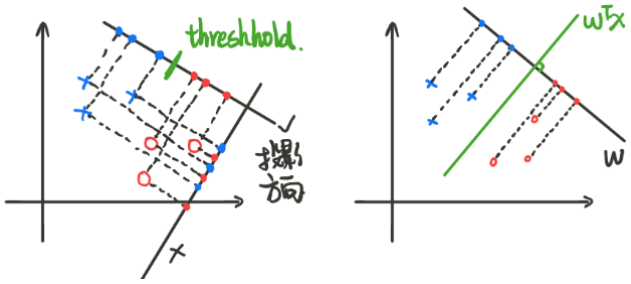

图 1: Fisher 线性判别分析模型图

在左图中，我们设置了两个投影方向，很显然上面那个投影方向可以更好的将两个点之间分开。我们可以在投影方向上找一个点作为两个类别的分界点，也就是阈值 (Threshhold)。首先，我们先引入

一个有关投影算法的小知识。

1.1 投影算法

首先，我们需要设定一个投影向量 w，为了保险起见，对这个投影向量 w 作出约束，令 $||w|| = 1$。那么，在空间中的一个数据点，也就是一个向量，在投影向量上的投影长度可以表述为：

$$x_i \cdot w = |x_i||w| \cos\theta = |x_i| \cos\theta = \triangle \tag{3}$$

1.2 Fisher 判别分析的损失函数表达式

在这个部分，主要是要得出 Fisher 判别分析的损失函数表达式求法。对于，投影的平均值和方差，我们可以分别表述为:

$$\bar{z} = \frac{1}{N}\sum_{i=1}^{N} z_i = \frac{1}{N}\sum_{i=1}^{N} w^T x_i \tag{4}$$

$$S_z = \frac{1}{N}\sum_{i=1}^{N}(z_i - \bar{z})(z_i - \bar{z})^T \tag{5}$$

那么对于第一类分类点 X_{c_1} 和第二类分类点 X_{c_2} 可以表述为：

$$C_1: \qquad \bar{z_1} = \frac{1}{N_1}\sum_{i=1}^{N_1} w^T x_i \qquad S_1 = \frac{1}{N_1}\sum_{i=1}^{N}(z_i - \bar{z_1})(z_i - \bar{z_1})^T \tag{6}$$

$$C_2: \qquad \bar{z_2} = \frac{1}{N_2}\sum_{i=1}^{N_2} w^T x_i \qquad S_2 = \frac{1}{N_2}\sum_{i=1}^{N}(z_i - \bar{z_2})(z_i - \bar{z_2})^T \tag{7}$$

那么类间的距离我们可以定义为：$(\bar{z_1} - \bar{z_2})^2$, 类内的距离被我们定义为 $S_1 + S_2$。那么我们的目标函数 Target Function $\mathcal{J}(w)$，可以被定义为:

$$\mathcal{J}(w) = \frac{(\bar{z_1} - \bar{z_2})^2}{S_1 + S_2} \tag{8}$$

因为，我们的目的是使方差越小越好，均值之间的差越大越好。

1.3 损失函数表达式的化简

1.3.1 $(\bar{z_1} - \bar{z_2})^2$

分子的化简过程如下所示：

$$\begin{aligned}(\bar{z_1} - \bar{z_2})^2 &= \left(\frac{1}{N_1}\sum_{i=1}^{N_1} w^T x_i - \frac{1}{N_2}\sum_{i=1}^{N_2} w^T x_i\right)^2 \\ &= \left(w^T(\frac{1}{N_1}\sum_{i=1}^{N_1} x_i - \frac{1}{N_2}\sum_{i=1}^{N_2} x_i)\right)^2 \\ &= \left(w^T(\bar{X}_{c_1} - \bar{X}_{c_2})\right)^2 \\ &= w^T(\bar{X}_{c_1} - \bar{X}_{c_2})(\bar{X}_{c_1} - \bar{X}_{c_2})^T w\end{aligned} \tag{9}$$

1.3.2 $S_1 + S_2$

分母的化简过程如下所示：

$$\begin{aligned} S_1 =& \frac{1}{N_1}\sum_{i=1}^{N}(z_i - \bar{z}_1)(z_i - \bar{z}_1)^T \\ =& \frac{1}{N_1}\sum_{i=1}^{N}(w^T x_i - \frac{1}{N_1}\sum_{i=1}^{N_1} w^T x_i)(w^T x_i - \frac{1}{N_1}\sum_{i=1}^{N_1} w^T x_i)^T \\ =& w^T \frac{1}{N_1}\sum_{i=1}^{N}(x_i - \frac{1}{N_1}\sum_{i=1}^{N_1} x_i)(x_i - \frac{1}{N_1}\sum_{i=1}^{N_1} x_i)^T w \\ =& w^T S_{c_1} w \end{aligned} \tag{10}$$

同理可得，

$$S_1 = w^T S_{c_2} w \tag{11}$$

所以，

$$S_1 + S_2 = w^T (S_{c_1} + S_{c_2}) w \tag{12}$$

1.3.3 $\mathcal{J}(w)$ 的最简表达形式

$$\mathcal{J}(w) = \frac{w^T(\bar{X}_{c_1} - \bar{X}_{c_2})(\bar{X}_{c_1} - \bar{X}_{c_2})^T w}{w^T(S_{c_1} + S_{c_2})w} \tag{13}$$

令 S_b 为 between-class 类间方差，S_w 为 within-class，也就是类内方差。那么有

$$S_b = (\bar{X}_{c_1} - \bar{X}_{c_2})(\bar{X}_{c_1} - \bar{X}_{c_2})^T \qquad S_w = (S_{c_1} + S_{c_2}) \tag{14}$$

于是，我们可以得到进一步化简的表达式；

$$\mathcal{J}(w) = \frac{w^T S_b w}{w^T S_w w} \tag{15}$$

1.4 损失函数 $\mathcal{J}(w)$ 的梯度

为了方便求导，我们令 $\mathcal{J}(w) = (w^T S_b w)(w^T S_w w)^{-1}$。

$$\begin{aligned} \frac{\partial \mathcal{J}(w)}{\partial w} &= 2S_b w(w^T S_w w)^{-1} + (-1)(w^T S_b w)(w^T S_w w)^{-2} \cdot 2S_w w = 0 \\ S_b w(w^T S_w w)^{-1} &= (w^T S_b w)(w^T S_w w)^{-2} S_w w \end{aligned} \tag{16}$$

显然，w 的维度是 $p \times 1$，w^T 的维度是 $1 \times p$，S_w 的维度是 $p \times p$，所以，$w^T S_w w$ 是一个实数，同理可得，$w^T S_w w$ 是一个实数所以，可以得到

$$S_b w = \frac{(w^T S_b w)}{(w^T S_w w)} S_w w \tag{17}$$

我们主要是需要求 w 的方向，大小不是很重要了。并且根据我们的定义，$w^T S_b w$ 和 $w^T S_w w$ 都是正的。所以，我们可得

$$w = \frac{(w^T S_b w)}{(w^T S_w w)} S_b^{-1} S_w w \propto S_b^{-1} S_w w \tag{18}$$

$$S_w w = (\bar{X}_{c_1} - \bar{X}_{c_2})(\bar{X}_{c_1} - \bar{X}_{c_2})^T w \tag{19}$$

而 $(\bar{X}_{c_1} - \bar{X}_{c_2})^T w$ 是一个实数，不会改变 w 的方向，所以汇总可得:

$$S_b^{-1} S_w w \propto S_w^{-1}(\bar{X}_{c_1} - \bar{X}_{c_2}) \tag{20}$$

那么，我们就可以求得 w 的方向为 $S_w^{-1}(\bar{X}_{c_1} - \bar{X}_{c_2})$。如果，$S_w^{-1}$ 是一个各向同性的对角矩阵，那么 $S^{-1} \propto I$。所以，$w \propto (\bar{X}_{c_1} - \bar{X}_{c_2})$。既然，求得了 w 的方向，其实 w 的大小就不重要的。

Linear Classification 04 Logistic Regression

Chen Gong

1 November 2019

在前面的两小节中我们，我们讨论了有关于线性分类问题中的硬分类问题，也就是感知机和 Fisher 线性判别分析。那么，我们接下来的部分需要讲讲软分类问题。软分类问题，可以大体上分为概率判别模型和概率生成模型，概率生成模型也就是高斯判别分析 (Gaussian Discriminate Analysis)，朴素贝叶斯 (Naive Bayes)。而线性判别模型也就是本章需要讲述的重点，Logistic Regression。

1 从线性回归到线性分类

线性回归的问题，我们可以看成这样一个形式，也就是 $w^T x$。而线性分类的问题可以看成是 $\{0,1\}$ 或者 $[0,1]$ 的问题。其实，从从线性回归到线性分类之间通过一个映射，也就是 Activate Function 来实现的，通过这个映射我们可以实现 $w^T x \longmapsto \{0,1\}$。

而在 Logistic Regression 中，我们将激活函数定义为：

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

那么很显然会有如下的性质：

1.$\lim_{z\longrightarrow+\infty}\sigma(z) = 1$

2.$\lim_{z\longrightarrow 0}\sigma(z) = \frac{1}{2}$

3.$\lim_{z\longrightarrow-\infty}\sigma(z) = 0$

那么，通过这样一个激活函数 σ，我们就可以将实现 $\mathbb{R} \longrightarrow (0,1)$。那么我们会得到以下的表达式：

$$p(y|x) = \begin{cases} p_1 = p(y=1|x) = \sigma(w^T x) = \frac{1}{1+exp\{-w^t x\}} & y=1 \\ p_2 = p(y=0|x) = 1 - p(y=1|x) = \frac{exp\{-w^t x\}}{1+exp\{-w^t x\}} & y=0 \end{cases} \tag{2}$$

而且，我们可以想一个办法来将两个表达式合二为一，那么有：

$$p(y|x) = p_1^y \cdot p_0^{1-y} \tag{3}$$

2 最大后验估计

$$
\begin{aligned}
MLE =& \hat{w} = \arg\max_{w} \log p(y|x) \\
&= \arg\max_{w} \log p(y_i|x_i) \\
&= \arg\max_{w} \sum_{i=1}^{N} \log p(y_i|x_i) \\
&= \arg\max_{w} \sum_{i=1}^{N} y \log p_1 + (1-y) \log p_2
\end{aligned} \tag{4}
$$

我们令，

$$
\frac{1}{1+exp\{-w^T x\}} = \varphi(x,w) \qquad \frac{exp\{-w^T x\}}{1+exp\{-w^T x\}} = 1-\varphi(x,w) \tag{5}
$$

那么，

$$
MLE = argmax_w \sum_{i=1}^{N} y \log \varphi(x,w) + (1-y)\log(1-\varphi(x,w)) \tag{6}
$$

实际上 $y \log \varphi(x,w) + (1-y)\log(1-\varphi(x,w))$ 就是一个交叉熵 (Cross Entropy)。那么，我们成功的找到了我们的优化目标函数，可以表述为 MLE (max) $\longrightarrow$ Loss function (Min Cross Entropy)。所以，这个优化问题就转换成了一个 Cross Entropy 的优化问题，这样的方法就很多了。

交叉熵是用来衡量两个分布的相似程度的，通过如下公式进行计算，其中 $p(x)$ 为真实分布，$q(x)$ 为预测分布：

$$
H(p,q) = \sum_{x} -p(x) log\, q(x) \tag{7}
$$

$$
H(p,q) = \int_{x} -p(x) log\, q(x)\, dx = \mathbb{E}_{x\sim p(x)}[-log\, q(x)] \tag{8}
$$

Linear Classification 05 Gaussian Discriminate Analysis

Chen Gong

03 November 2019

前面讲的方法都是概率判别模型，包括，Logistic Regression 和 Fisher 判别分析。接下来我们将要学习的是概率生成模型部分，也就是现在讲到的 Gaussian Discriminate Analysis。数据集的相关定义为：

$$X = (x_1, x_2, \cdots, x_N)^T = \begin{pmatrix} x_1^T \\ x_2^T \\ \vdots \\ x_N^T \end{pmatrix} = \begin{pmatrix} x_{11} & x_{12} & \dots & x_{1p} \\ x_{21} & x_{32} & \dots & x_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ x_{N1} & x_{N2} & \dots & x_{Np} \end{pmatrix}_{N \times P} \tag{1}$$

$$Y = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{pmatrix}_{N \times 1} \tag{2}$$

那么，我们的数据集可以记为 $\{(x_i, y_i)\}_{i=1}^N$，其中，$x_i \in \mathbb{R}^p$，$y_i \in \{+1, -1\}$。我们将样本点分成了两个部分：

$$\begin{cases} C_1 = \{x_i | y_i = 1, i = 1, 2, \cdots, N_1\} \\ C_2 = \{x_i | y_i = 0, i = 1, 2, \cdots, N_2\} \end{cases} \tag{3}$$

并且有 $|C_1| = N_1$，$|C_2| = N_2$，且 $N_1 + N_2 = N$。

1 概率判别模型与生成模型的区别

什么是判别模型？所谓判别模型，也就是求

$$\hat{y} = \arg\max_y \ p(y|x) \qquad y \in \{0, 1\} \tag{4}$$

重点在于求出这个概率来，知道这个概率的值等于多少。而概率生成模型则完全不一样。概率生成模型不需要知道概率值具体是多大，只需要知道谁大谁小即可，具体是对联合概率进行建模。举例即为 $p(y = 0|x)$ 和 $p(y = 1|x)$，谁大谁小的问题。而概率生成模型的求法可以用贝叶斯公式来进行求解，即为：

$$p(y|x) = \frac{p(x|y)p(y)}{p(x)} = \frac{p(x, y)}{p(x)} \propto p(x, y) \tag{5}$$

因为在这个公式中，比例大小 $p(x)$ 与 y 的取值无关，所以它是一个定值。所以，概率生成模型实际上关注的就是一个求联合概率分布的问题。那么，总结一下

$$p(y|x) \propto p(x|y)p(y) \propto p(x,y) \tag{6}$$

其中，$p(y|x)$ 为 Posterior function，$p(y)$ 为 Prior function，p(x|y) 为 Likelihood function。所以有

$$\hat{y} = \arg\max_{y\in\{0,1\}} p(y|x) \propto \arg\max_{y\in\{0,1\}} p(x|y)p(y) \tag{7}$$

2 Gaussian Discriminate Analysis 模型建立

在二分类问题中，很显然可以得到，我们的**先验概率**符合，$p(y)$ ~Bernoulli Distribution。也就是，

y	1	0
p	φ	$1-\varphi$

表 1: Bernoulli 分布的概率分布表

所以，可以写出：

$$p(y) = \begin{cases} \varphi^y & y=1 \\ (1-\varphi)^{1-y} & y=0 \end{cases} \Rightarrow \varphi^y(1-\varphi)^{1-y} \tag{8}$$

而随后是要确定**似然函数**，我们假设他们都符合高斯分布。对于不同的分类均值是不同的，但是不同变量之间的协方差矩阵是一样的。那么我们可以写出如下的形式：

$$p(x|y) = \begin{cases} p(x|y=1) \sim \mathcal{N}(\mu_1, \Sigma) \\ p(x|y=0) \sim \mathcal{N}(\mu_2, \Sigma) \end{cases} \Rightarrow \mathcal{N}(\mu_1,\Sigma)^y \mathcal{N}(\mu_2,\Sigma)^{1-y} \tag{9}$$

那么我们的 Likelihood function 可以被定义为：

$$\begin{aligned} \mathcal{L}(\theta) &= \log \prod_{i=1}^{N} p(x_i, y_i) \\ &= \sum_{i=1}^{N} \log p(x_i, y_i) \\ &= \sum_{i=1}^{N} \log p(x_i|y_i)p(y_i) \\ &= \sum_{i=1}^{N} [\log p(x_i|y_i) + \log p(y_i)] \\ &= \sum_{i=1}^{N} \left[\log \mathcal{N}(\mu_1,\Sigma)^{y_i} \mathcal{N}(\mu_2,\Sigma)^{1-y_i} + \log \varphi_i^{y}(1-\varphi)^{1-y_i}\right] \\ &= \sum_{i=1}^{N} \log \mathcal{N}(\mu_1,\Sigma)^{y_i} + \sum_{i=1}^{N} \log \mathcal{N}(\mu_2,\Sigma)^{1-y_i} + \sum_{i=1}^{N} \log \varphi^{y_i} + \sum_{i=1}^{N} \log(1-\varphi)^{1-y_i} \end{aligned} \tag{10}$$

为了方便后续的推演过程，所以，我们将 Likelihood function 写成，

$$\mathcal{L}(\theta) = ①+②+⑧$$

并且，我们令：$① = \sum_{i=1}^{N} \log \mathcal{N}(\mu_1, \Sigma)_i^y$，$② = \sum_{i=1}^{N} \log \mathcal{N}(\mu_2, \Sigma)^{1-y_i}$，
$③ = \sum_{i=1}^{N} \log \varphi^{y_i} + \sum_{i=1}^{N} \log(1-\varphi)^{1-y_i}$。那么上述函数我们可以表示为：

$$\theta = (\mu_1, \mu_2, \Sigma, \varphi) \qquad \hat{\theta} = \arg\max_{\theta} \mathcal{L}(\theta) \tag{11}$$

3 Likelihood function 参数的极大后验估计

Likelihood function 的参数为 $\theta = (\mu_1, \mu_2, \Sigma, \varphi)$，下面我们分别用极大似然估计对这四个参数进行求解。下面引入几个公式：

$$tr(AB) = tr(BA) \tag{12}$$

$$\frac{\partial tr(AB)}{\partial A} = B^T \tag{13}$$

$$\frac{\partial |A|}{\partial A} = |A| A^{-T} \tag{14}$$

$$\frac{\partial \log |A|}{\partial A} = A^{-T} \tag{15}$$

3.1 求解 φ

$$③ = \sum_{i=1}^{N} \log \varphi^{y_i} + \sum_{i=1}^{N} \log(1-\varphi)^{1-y_i} = \sum_{i=1}^{N} y_i \log \varphi + \sum_{i=1}^{N} (1-y_i) \log(1-\varphi)$$

$$\frac{\partial ③}{\partial \varphi} = \sum_{i=1}^{N} y_i \frac{1}{\varphi} - \sum_{i=1}^{N} (1-y_i) \frac{1}{1-\varphi} = 0 \tag{16}$$

$$\sum_{i=1}^{N} y_i(1-\varphi) - (1-y_i)\varphi = 0 \tag{17}$$

$$\sum_{i=1}^{N} (y_i - \varphi) = 0 \tag{18}$$

$$\hat{\varphi} = \frac{1}{N} \sum_{i=1}^{N} y_i \tag{19}$$

又因为 $y_i = 0$ 或 $y_i = 1$，所以 $\hat{\varphi} = \frac{1}{N} \sum_{i=1}^{N} y_i = \frac{N_1}{N}$。

3.2 求解 μ_1

$$① = \sum_{i=1}^{N} \log \mathcal{N}(\mu_1, \Sigma)^{y_i}$$
$$= \sum_{i=1}^{N} y_i \log \frac{1}{(2\pi)^{\frac{p}{2}} |\Sigma|^{\frac{1}{2}}} exp\left\{-\frac{1}{2}(x_i - \mu_1)^T \Sigma^{-1} (x_i - \mu_1)\right\}$$

那么求解过程如下所示：由于到对 μ_1 求偏导，我们只需要关注公式中和 μ_1 有关的部分。那么我们可以将问题简化为：

$$\max_{\mu_1} \sum_{i=1}^{N} y_i \log exp\left\{-\frac{1}{2}(x_i - \mu_1)^T \Sigma^{-1} (x_i - \mu_1)\right\} \tag{20}$$

然后将 exp 和 log 抵消掉，再将括号打开，我们可以得到最终的化简形式：

$$\max_{\mu_1} -\frac{1}{2}\sum_{i=1}^{N} y_i \left\{x_i^T\Sigma^{-1}x_i - 2\mu_1^T\Sigma^{-1}x_i + \mu_1^T\Sigma^{-1}\mu_1\right\} \tag{21}$$

为了方便表示，我们令① = △。所以，极大似然法求解过程如下：

$$\begin{aligned}
\frac{\partial \triangle}{\partial \mu_1} =& -\frac{1}{2}\sum_{i=1}^{N} y_i(-2\Sigma^{-1}x_i + 2\Sigma^{-1}\mu_1) = 0 \\
=& \sum_{i=1}^{N} y_i(\Sigma^{-1}x_i - \Sigma^{-1}\mu_1) = 0 \\
=& \sum_{i=1}^{N} y_i(x_i - \mu_1) = 0 \\
=& \sum_{i=1}^{N} y_i x_i = \sum_{i=1}^{N} y_i\mu_1 \\
\mu_1 =& \frac{\sum_{i=1}^{N} y_i x_i}{\sum_{i=1}^{N} y_i} = \frac{\sum_{i=1}^{N} y_i x_i}{N_1}
\end{aligned} \tag{22}$$

3.3 求解 μ_2

μ_2 的求解过程与 μ_1 的基本保持一致性。区别点从公式 (22) 开始，我们有：

$$\max_{\mu_2} -\frac{1}{2}\sum_{i=1}^{N}(1-y_i) \left\{x_i^T\Sigma^{-1}x_i - 2\mu_2^T\Sigma^{-1}x_i + \mu_2^T\Sigma^{-1}\mu_2\right\} \tag{23}$$

极大似然法的求解过程如下所示：

$$\begin{aligned}
\frac{\partial \triangle}{\partial \mu_2} =& -\frac{1}{2}\sum_{i=1}^{N}(1-y_i)(-2\Sigma^{-1}x_i + 2\Sigma^{-1}\mu_2) = 0 \\
=& \sum_{i=1}^{N}(1-y_i)(x_i - \mu_2) = 0 \\
=& \sum_{i=1}^{N} x_i - \sum_{i=1}^{N} y_i x_i = N\mu_2 - \sum_{i=1}^{N} y_i\mu_2 \\
\mu_2 =& \frac{\sum_{i=1}^{N} x_i - \sum_{i=1}^{N} y_i x_i}{N - \sum_{i=1}^{N} y_i} = \frac{\sum_{i=1}^{N} x_i - \sum_{i=1}^{N} y_i x_i}{N - N_1} \\
=& \frac{\sum_{i=1}^{N}(1-y_i)x_i}{N_2}
\end{aligned} \tag{24}$$

也可以对于求 μ_1 来说，求 μ_2 可以类比，将其中的 N_1 换成 N_2，其中的 y_i 换成 $1-y_i$，可以得到同样的结果。

3.4 求解 Σ

如果要使用极大似然估计来求解 Σ，这只与 $\mathcal{L}(\theta)$ 中的①和②有关。并且①+②的表达式为：

$$\sum_{i=1}^{N} \log\mathcal{N}(\mu_1, \Sigma)^{y_i} + \sum_{i=1}^{N} \log\mathcal{N}(\mu_2, \Sigma)^{1-y_i} \tag{25}$$

那么，按照分类点的方法，我们可以将其改写为：

$$\hat{\Sigma} = \arg\min_{\Sigma} \sum_{x \in C_1} \log \mathcal{N}(\mu_1, \Sigma) + \sum_{x \in C_2} \log \mathcal{N}(\mu_2, \Sigma) \tag{26}$$

公式加号前后都是一样的，所以，为了方便计算我们暂时只考虑一半的计算：

$$\begin{aligned}
\sum_{i=1}^{N} \log \mathcal{N}(\mu, \Sigma) =& \sum_{i=1}^{N} \log \frac{1}{(2\pi)^{\frac{p}{2}} |\Sigma|^{\frac{1}{2}}} exp\left\{ -\frac{1}{2}(x_i - \mu)^T \Sigma^{-1} (x_i - \mu) \right\} \\
=& -\sum_{i=1}^{N} \frac{p}{2} \log 2\pi - \sum_{i=1}^{N} \frac{1}{2} \log |\Sigma| - \frac{1}{2}(x_i - \mu)^T \Sigma^{-1} (x_i - \mu) \\
=& C - \frac{N}{2} \log |\Sigma| - \frac{1}{2} \sum_{i=1}^{N} (x_i - \mu)^T \Sigma^{-1} (x_i - \mu) \\
=& C - \frac{N}{2} \log |\Sigma| - \frac{1}{2} \sum_{i=1}^{N} tr\left((x_i - \mu)^T \Sigma^{-1} (x_i - \mu) \right) \\
=& C - \frac{N}{2} \log |\Sigma| - \frac{1}{2} \sum_{i=1}^{N} tr\left((x_i - \mu)(x_i - \mu)^T \Sigma^{-1} \right)
\end{aligned} \tag{27}$$

而且，

$$S = \frac{1}{N} \sum_{i=1}^{N} (x_i - \mu)(x_i - \mu)^T \tag{28}$$

所以，

$$\sum_{i=1}^{N} \log \mathcal{N}(\mu, \Sigma) = C - \frac{N}{2} \log |\Sigma| - \frac{N}{2} tr(S\Sigma^{-1}) \tag{29}$$

那么代入公式 (27) 中，我们可以得到：

$$\begin{aligned}
\hat{\Sigma} =& \arg\max_{\Sigma} \ C - \frac{N_1}{2} \log |\Sigma| - \frac{N_1}{2} tr(S_1 \Sigma^{-1}) + C - \frac{N_2}{2} \log |\Sigma| - \frac{N_2}{2} tr(S_2 \Sigma^{-1}) \\
=& \arg\max_{\Sigma} \ -\frac{N}{2} \log |\Sigma| - \frac{N_1}{2} tr(S_1 \Sigma^{-1}) - \frac{N_2}{2} tr(S_2 \Sigma^{-1}) \\
=& \arg\min_{\Sigma} \ N \log |\Sigma| + N_1 tr(S_1 \Sigma^{-1}) + N_2 tr(S_2 \Sigma^{-1})
\end{aligned} \tag{30}$$

我们令函数 $N \log |\Sigma| + N_1 tr(S_1 \Sigma^{-1}) + N_2 tr(S_2 \Sigma^{-1}) = \triangle$，那么对 Σ 求偏导并令其等于 0 可得：

$$\frac{\partial \triangle}{\partial \Sigma} = N\Sigma^{-1} - N_1 \Sigma^{-1} S_1 \Sigma^{-1} - N_2 \Sigma^{-1} S_2 \Sigma^{-1} = 0 \tag{31}$$

对上式左乘 Σ，又乘 Σ 得到 $N\Sigma - N_1 S_1 - N_2 S_2 = 0$。

解得：

$$\Sigma = \frac{N_1 S_1 + N_2 S_2}{N} \tag{32}$$

其中对 $tr(S_1 \Sigma^{-1})$ 求偏导的过程如下（由于 $\Sigma\Sigma^{-1} = \mathbb{I}$，所以 $d(\Sigma^{-1}\Sigma) = \mathbb{O} \Rightarrow (d\Sigma)\Sigma^{-1} + \Sigma d(\Sigma^{-1}) = 0 \Rightarrow d\Sigma^{-1} = -\Sigma^{-1}(d\Sigma)\Sigma^{-1}$：

$$\begin{aligned}
d\, tr(S_1 \Sigma^{-1}) =& tr(S_1 d\Sigma^{-1}) \\
=& tr(-S_1 \Sigma^{-1} (d\Sigma) \Sigma^{-1}) \\
=& tr(-\Sigma^{-1} S_1 \Sigma^{-1} d\Sigma)
\end{aligned} \tag{33}$$

于是 $\frac{\partial tr(S_1 \Sigma^{-1})}{\partial \Sigma} = -\Sigma^{-1} S_1 \Sigma^{-1}$。同理可以知道 $\frac{\partial tr(S_2 \Sigma^{-1})}{\partial \Sigma} = -\Sigma^{-1} S_2 \Sigma^{-1}$。

4 总结

下面对 Gaussian Discriminate Analysis 做一个简单的小结。我们使用模型为:

$$\hat{y} = \arg\max_{y\in\{0,1\}} p(y|x) \propto \arg\max_{y\in\{0,1\}} p(x|y)p(y) \tag{34}$$

$$\begin{cases} p(y) = \varphi^y(1-\varphi)^{1-y} \\ p(x|y) = \mathcal{N}(\mu_1,\Sigma)^y\mathcal{N}(\mu_2,\Sigma)^{1-y} \end{cases} \tag{35}$$

利用极大似然估计得到的结果为:

$$\theta = (\mu_1,\mu_2,\Sigma,\varphi) = \begin{cases} \hat{\varphi} = \frac{N_1}{N} \\ \mu_1 = \frac{\sum_{i=1}^N y_i x_i}{N_1} \\ \mu_2 = \frac{\sum_{i=1}^N (1-y_i) x_i}{N_2} \\ \Sigma = \frac{N_1 S_1 + N_2 S_2}{N} \end{cases} \tag{36}$$

Linear Classification 06 Naive Bayes

Chen Gong

04 November 2019

本节主要是介绍一下 Naive Bayes Classification，也就是朴素贝叶斯分类。朴素贝叶斯分类器的核心思想也就是，条件独立性假设。这是一种最简单的概率图模型，也就是一种有向图模型。

1 条件独立性假设

条件独立性假设用简单的图来进行表述，可以表示为如下图所示的形式：

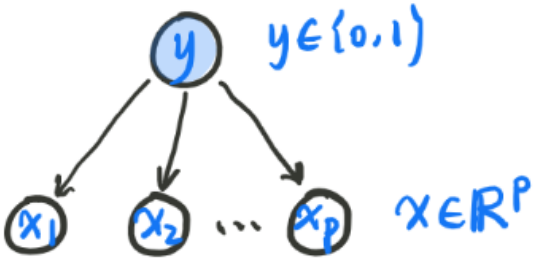

图 1: 条件独立性假设

我们可以将其定义为 $x_i \perp x_j|y \ (i \neq j)$。根据贝叶斯公式可以得：

$$p(y|x) = \frac{p(x|y)p(y)}{p(x)} = \frac{p(x,y)}{p(x)} \propto p(x,y) \tag{1}$$

而做条件独立性假设的最终目的，是为了简化运算。因为对于一个数据序列 $x = (x_1, x_2, \cdots, x_p)$。如果 x_i 和 x_j 之间有关系的话，这个计算难度可能会变得很难，所以就假设各个变量之间是相互独立的。而且，马尔可夫决策链也就是这样类似的思想。

2 Naive Bayes Classification

朴素贝叶斯算法的优化目的即为：

$$\begin{aligned} \hat{y} &= \arg \max_{y \in \{0,1\}} p(y|x) \\ &= \arg \max_{y \in \{0,1\}} p(x|y)p(y) \end{aligned} \tag{2}$$

其中,

$$p(x|y) = \prod_{i=1}^{N} p(x_i|y) \tag{3}$$

对于 $p(y)$ 这个先验概率密度函数的确定，对于二分类问题，也就是 y ~Bernoulli Distribution，而对于多分类问题，先验概率为 y ~Categorial Distribution。而对于，$p(x|y) = \prod_{i=1}^{N} p(x_i|y)$。如果 x 是离散的，那么 $x_i|y$ ~Categorial Distribution；如果 x 是连续的，那么 $x_i|y \sim \mathcal{N}(\mu_y, \Sigma_y^2)$。对于每一类都有一个高斯分布。

而有关于 $p(x|y)$ 用极大似然估计 MLE，估计出来就行。因为分布的形式我们已经知道了，那么只要利用数据来进行学习，使用极大似然估计就可以得到想要的结果了。其实对于多分类的情况，Naive Bayes Classification 和 Guassian Discriminate Analysis 很像的。

Support Vector Machine 01 Hard Margin Modeling and Solution

Chen Gong

13 November 2019

众所周知，Support Vector Machine (SVM) 有三宝，间隔，对偶，核技巧。所以，SVM 可以大致被分为三类：hard-margin SVM；soft-margin SVM；kernel SVM。

1 SVM 基本思想

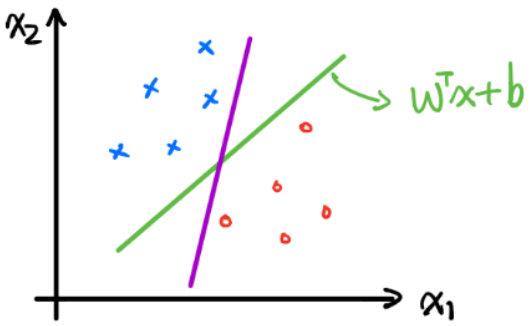

图 1: 二分类问题模型图

支持向量机模型可以被简要的描述为：$f(w) = w^T x + b$。很显然这是一个判别模型。实际上，我们想一想就知道，这样的直线其实有很多的。但是紫色的那条虽然可以做到分类的效果，但是效果也太差了，没有什么鲁棒性，泛化能力并不行。显然，绿色的那条直线要更好一些。那么，SVM 的基本思想可以被简要的概述为，找到一条最好的直线，离样本点距离足够的大。

2 SVM 模型建立

数据集可以描述为 $D = \{(x_i, y_i)\}_{i=1}^N$，其中 $x_i \in \mathbb{R}^p$，$y_i \in \{1, -1\}$。

首先我们希望，把这些点的间隔分得越大越好，并且根据符号函数给不同的值相应的类别标号。那么，我们可以写做：

$$\begin{aligned} &\max_{w,b} \ margin\ (w, b) \\ &s.t. \begin{cases} w^T x_i + b > 0 & y_i = +1 \\ w^T x_i + b < 0 & y_i = -1 \end{cases} \end{aligned} \tag{1}$$

由于 y_i 和 $w_i^T x + b$ 是同号的，那么很显然有 $y_i(w_i^T x + b) > 0$，所以，模型被我们改写为：

$$\begin{aligned} &\max_{w,b} \ margin\ (w,b) \\ &s.t. \quad y_i(w_i^T x + b) > 0 \quad (i = 1, 2, \cdots, N) \end{aligned} \tag{2}$$

平面上一点到某一直线的矩阵的计算方法比较简单。对于平面上一条直线 $y = w^T x + b$，点 (x_i, y_i) 到直线的距离，可以被记做：

$$distance = \frac{1}{||w||}|w^T x + b| \tag{3}$$

我们的希望是离超平面最近的点分得越开越好。离超平面最近的点就是 $\min distance(w, b, x_i)$，这个是针对点 $x_i (i = 1, 2, \cdots, n)$。然后就是分得越开越好，那么我们可以描述为 $\max \min distance(w, b, x_i)$，这个是针对 w, b 进行优化的。那么我们可以把模型进一步改写为：

$$\begin{aligned} &\max_{w,b} \min_{x_i} \frac{1}{||w||}|w^T x_i + b| \\ &s.t. \quad y_i(w_i^T x + b) > 0 \quad (i = 1, 2, \cdots, N) \end{aligned} \tag{4}$$

对于约束条件 $y_i(w_i^T x + b) > 0 \quad (i = 1, 2, \cdots, N)$，很显然可以得到 $\exists \gamma > 0$ 使得 $s.t.\ \min y_i(w_i^T x + b) = \gamma$。这里很显然我们可以使用一个小技巧来做一些的调整，来使我们方便计算，我们可以把约束条件转换为 $s.t.\ \min \frac{y_i(w_i^T x + b)}{z} = \frac{\gamma}{z}$。我们很显然可以看到，$w$ 和 b 之间是可以自由放缩的，那么就放缩到令 $\frac{\gamma}{z} = 1$，那么就有 $\min y_i(w_i^T x + b) = 1$。于是，模型可以化简为：

$$\begin{aligned} &\max_{w,b} \frac{1}{||w||} \\ &s.t. \quad \min_{x_i} y_i(w_i^T x + b) = 1 \Longrightarrow y_i(w_i^T x + b) \geq 1 \quad (i = 1, 2, \cdots, N) \end{aligned} \tag{5}$$

将该优化问题进行等价变换：

$$\begin{aligned} &\min_{w,b} \ \frac{1}{2} w^T w \\ &s.t. \quad y_i(w_i^T x + b) \geq 1 \quad (i = 1, 2, \cdots, N) \end{aligned} \tag{6}$$

很显然，这是一个凸优化 (Convex Optimization) 问题，目标函数是二次函数，一共有 N 个约束。那么这是一个二次规划问题 (Quadratic Programming)，通常也被描述为 QP 问题。

3 模型求解

在支持向量机的模型求解中，一个非常重要的概念就是将原问题 (Prime Problem) 转换为对偶问题 (Dual Problem)。我们将模型进一步改写为：

$$\begin{aligned} &\max_{w,b} \ \frac{1}{2} w^T w \\ &s.t. \quad 1 - y_i(w_i^T x + b) \leq 0 \quad (i = 1, 2, \cdots, N) \end{aligned} \tag{7}$$

求解带约束的极值，显然需要采用拉格朗日乘子法，我们定义拉格朗日函数为：

$$\mathcal{L}(w, b, \lambda) = \frac{1}{2} w^T w + \sum_{i=1}^{N} \lambda_i (1 - y_i(w^T x_i + b)) \tag{8}$$

在拉格朗日数乘法里，λ 一定是大于零的数。所以模型为：

$$\begin{aligned} &\min_{w,b} \ \max_{\lambda} \ \mathcal{L}(w,b,\lambda) \\ &s.t. \quad \lambda_i \geq 0 \quad (i=1,2,\cdots,N) \end{aligned} \tag{9}$$

很显然，在这里，我们就将一个带约束的问题转换成了一个无约束的问题。

然而我们需要考虑一个问题，那就是 $\mathcal{L}(w,b,\lambda)$ 是否一定和公式 (7) 等价呢？这需要探究验证一下。

$$\begin{aligned} &if \ 1-y_i(w^Tx_i+b) \geq 0, \ \max_{\lambda} \mathcal{L}(\lambda,w,b) = +\infty \\ &if \ 1-y_i(w^Tx_i+b) \leq 0, \ \max_{\lambda} \mathcal{L}(\lambda,w,b) = 0 \end{aligned} \tag{10}$$

很显然在 $\min_{w,b} \ \max_{\lambda} \ \mathcal{L}(w,b,\lambda)$ 的计算中可以表示为：

$$\min_{w,b} \ \max_{\lambda} \ \mathcal{L}(w,b,\lambda) = \min_{w,b}\{+\infty, \frac{1}{2}w^Tw\} = \frac{1}{2}w^Tw \tag{11}$$

所以在上述的描述中，我们可以得到，实际上 $\min_{w,b} \ \max_{\lambda} \ \mathcal{L}(w,b,\lambda)$ 中隐藏了一个 $1-y_i(w^Tx_i+b) \leq 0$ 的隐藏条件。所以两种写法实际上是等价的。为了方便计算，下面我们需要使用对偶的方法，也就是将模型作如下的转换：

$$\begin{cases} \min_{w,b} \ \max_{\lambda} \ \mathcal{L}(w,b,\lambda) \\ s.t. \quad \lambda_i \geq 0 \end{cases} \overset{dual}{\Longrightarrow} \begin{cases} \max_{\lambda} \ \min_{w,b} \ \mathcal{L}(w,b,\lambda) \\ s.t. \quad \lambda_i \geq 0 \end{cases} \tag{12}$$

这里我们需要介绍两种对偶关系，所谓：

弱对偶关系就是：$\min\max\mathcal{L} \geq \max\min\mathcal{L}$。

强对偶关系就是：$\min\max\mathcal{L} = \max\min\mathcal{L}$。

大家或许对这个关系会有点懵逼，其实仔细用直觉来想想还是很好接受的，具体的证明过程这里就不再做过多的阐述了。中国有句古话叫：“宁做鸡头不做凤尾”，但是凤就是凤始终要比鸡好。先取 max 就是凤的意思，然后取 min 就是凤尾。同理先取 min 就是鸡的意思，然后取 max 就是鸡头的意思。凤尾肯定比鸡头要好，当然这是直观的理解。而对于强对偶关系，需要我们满足 KKT 条件，这个后面会详细的说。

3.1 估计参数的值

我们的目标是 $\min_{w,b} \ \mathcal{L}(w,b,\lambda)$，那么

$$\frac{\partial\mathcal{L}}{\partial b} = \frac{\partial}{\partial b}\sum_{i=1}^{N}\lambda_i[1-y_i(w^Tx_i+b)] = 0 \tag{13}$$

$$-\sum_{i=1}^{N}\lambda_iy_i = 0 \tag{14}$$

代入到 $\mathcal{L}(w,b,\lambda)$ 中可得，

$$\mathcal{L}(w,b,\lambda) = \frac{1}{2}w^Tw + \sum_{i=1}^{N}\lambda_i(1-y_i(w^Tx_i+b)) \tag{15}$$

$$=\frac{1}{2}w^Tw+\sum_{i=1}^{N}\lambda_i-\sum_{i=1}^{N}\lambda_iy_iw^Tx_i-\sum_{i=1}^{N}\lambda_iy_ib \tag{16}$$

$$=\frac{1}{2}w^Tw+\sum_{i=1}^{N}\lambda_i-\sum_{i=1}^{N}\lambda_iy_iw^Tx_i \tag{17}$$

下一步，则是对 w 求偏导，

$$\frac{\partial\mathcal{L}}{\partial w}=\frac{\partial}{\partial w}[\frac{1}{2}w^Tw+\sum_{i=1}^{N}\lambda_i-\sum_{i=1}^{N}\lambda_iy_iw^Tx_i]=w-\sum_{i=1}^{N}\lambda_iy_ix_i=0 \tag{18}$$

解得：

$$w=\sum_{i=1}^{N}\lambda_iy_ix_i \tag{19}$$

将 w 的值代入到 $\mathcal{L}(w,b,\lambda)$ 中可以得到：

$$\begin{aligned}\mathcal{L}(w,b,\lambda)=&\frac{1}{2}(\sum_{i=1}^{N}\lambda_iy_ix_i)^T(\sum_{i=1}^{N}\lambda_iy_ix_i)+\sum_{i=1}^{N}\lambda_i-\sum_{i=1}^{N}\lambda_iy_i(\sum_{i=1}^{N}\lambda_iy_ix_i)^Tx_i\\=&\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\lambda_i\lambda_jy_iy_j(x_i^Tx_j)-\sum_{i=1}^{N}\sum_{j=1}^{N}\lambda_i\lambda_jy_iy_j(x_j^Tx_i)+\sum_{i=1}^{N}\lambda_i\\=&-\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\lambda_i\lambda_jy_iy_j(x_i^Tx_j)+\sum_{i=1}^{N}\lambda_i\end{aligned} \tag{20}$$

所以，模型被我们改写为：

$$\begin{cases}\max_\lambda -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\lambda_i\lambda_jy_iy_j(x_i^Tx_j)+\sum_{i=1}^{N}\lambda_i\\ s.t.\quad \lambda_i\geq 0,\ \sum_{i=1}^{N}\lambda_iy_i=0\end{cases} \tag{21}$$

4 KKT 条件

这个 KKT 条件或许会让很多人都感觉一脸懵逼，作者自己也理解了很久才勉强把它看懂的，如果有什么不到位的地方，欢迎发邮件到 gongchen2020@ia.ac.cn 与作者取得联系。深刻理解 KKT 条件需要掌握一些凸优化的知识，支持向量机是一个典型的凸二次优化问题。KKT 条件可以帮助我们理解支持向量机的精髓，什么是支持向量？支持向量机只需要用少量的数据，有很强的鲁棒性，并且可以取得很好的效果。

KKT 条件可以描述为：

$$\begin{cases}\frac{\partial\mathcal{L}}{\partial w}=0,\quad \frac{\partial\mathcal{L}}{\partial b}=0\\ \lambda_i(1-y_i(w^Tx_i+b))=0\\ \lambda_i\geq 0\\ 1-y_i(w^Tx_i+b)\leq 0\end{cases} \tag{22}$$

其中 $\lambda_i(1-y_i(w^Tx_i+b))=0$ 是互补松弛条件 (Complementary Relaxation Condition)。满足 KKT 条件是原问题的对偶 (dual) 问题有强对偶关系的充分必要条件。下面我们用一张图来进行理解 KKT 条件的作用：

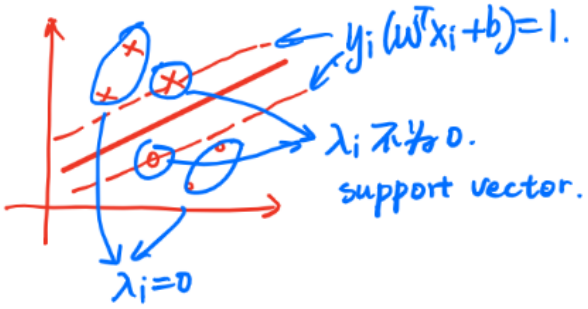

图 2: 支持向量的 KKT 条件

首先，需要明确，离分界面最近的数据点满足这个条件，$y_i(w^Tx_i+b)=1$ 至于为什么？前面的公式 (4) 有详细的分析。那么离分界面最近的数据点就被我们称为支持向量了。在支持向量上 $1-y_i(w^Tx_i+b)=0$，那么 λ_i 可以不为 0。而在其他向量上一定会有 $1-y_i(w^Tx_i+b)<0$ 为了满足 $\lambda_i(1-y_i(w^Tx_i+b))=0$，必然有 $\lambda_i=0$，那么我们就可以理解为这个数据点失去了作用。所以，KKT 条件使得，支持向量机中只有支持向量在模型的优化中有作用，这实在是太棒了。

为了确定这个超平面，我们已经得到了

$$w^*=\sum_{i=1}^{N}\lambda_i y_i x_i \tag{23}$$

但是，现在怎么求 b^* 是一个很尴尬的问题，因为我们在求 $\frac{\partial\mathcal{L}}{\partial b}$ 的时候，并没有看到和 b 相关的等式。但是我们知道只有支持向量会在模型求解中起作用，那么有支持向量 (x_k,y_k) 使得 $1-y_k(w^Tx_k+b)=0$。所以：

$$y_k(w^Tx_k+b)=1 \tag{24}$$

$$y_k^2(w^Tx_k+b)=y_k \tag{25}$$

$$b^*=y_k-w^Tx_k=y_k-\sum_{i=1}^{N}\lambda_i y_i x_i^Tx_k \tag{26}$$

那么做到这里，我们的 hard-margin SVM 就已经做完了。模型为 $f(x)=sign(w^{*T}x+b^*)$，超平面为 $w^{*T}+b^*$。其中 $w^*=\sum_{i=1}^{N}\lambda_i y_i x_i$，$b^*=y_k-\sum_{i=1}^{N}\lambda_i y_i x_i^Tx_k$。

5 小结

本节主要探究了 Hard-margin SVM 的建模和求解。最终解得对于一个 $\{(x_i,y_i)_{i=1}^N\}$ 的分类问题，使用支持向量机来求解，我们可以得到，分类模型为：

$$f(x)=sign(w^{*T}x+b^*) \qquad \begin{cases} w^*=\sum_{i=1}^{N}\lambda_i y_i x_i \\ b^*=y_k-\sum_{i=1}^{N}\lambda_i y_i x_i^Tx_k \end{cases} \tag{27}$$

KKT 条件是原问题的对偶 (dual) 问题有强对偶关系的充分必要条件。它成功的使支持向量机模

型的求解只和支持向量有关，这也是支持向量机的强大之处，运算比较简单，而且具有较强的鲁棒性。

$$\begin{cases} \frac{\partial \mathcal{L}}{\partial w} = 0, \quad \frac{\partial \mathcal{L}}{\partial b} = 0, \quad \frac{\partial \mathcal{L}}{\partial \lambda} = 0 \\ \lambda_i (1 - y_i (w^T x_i + b)) = 0 \\ \lambda_i \geq 0 \\ 1 - y_i (w^T x_i + b) \leq 0 \end{cases} \tag{28}$$

Support Vector Machine 02 Soft Margin

Chen Gong

15 November 2019

在上一小节中，我们介绍了 Hard-Margin SVM 的建模和求解过程。这个想法很好，但是实际使用过程中会遇到很多的问题。因为，并不一定数据集就可以被很好的分开，而且实际数据没有那么简单，其间有很多的噪声。而 Soft Margin 的基础思想就是允许那么一点点的错误。这样在实际运用中往往可以得到较好的效果。下面我们将进行 Soft Margin SVM 的详细演变过程。

1 Soft Margin SVM

最简单的思路就是在优化函数里面引入一个 loss function。也就是：

$$\min \ \frac{1}{2}w^T w + loss\ function \tag{1}$$

那么，我们如何来定义这个 loss function 呢？大致可以分这两种引入的模式：

1. loss = 错误点的个数 $= \sum_{i=1}^{N} I\{y_i(w^T x_i + b) < 1\}$，这个方法非常容易想到，但是我们马上就发现了一个问题，那就是这个函数不连续的，无法进行优化。这种方法非常容易想到。

2. loss：距离。现在我们做如下定义：

1) 如果 $y_i(w^T x_i + b) \geq 1$，$loss = 0$。

2) 如果 $y_i(w^T x_i + b) < 1$，$loss = 1 - y_i(w^T x_i + b)$。

那么，我们就可以将 loss function 定义为：

$$loss = \max\{0, 1 - y_i(w^T x_i + b)\} \tag{2}$$

进一步，我们令 $y_i(w^T x_i + b) = z$，那么：

$$loss_{max} = \max\{0, 1 - z\} \tag{3}$$

我们将 loss function 的图像画出来就如下图所示：

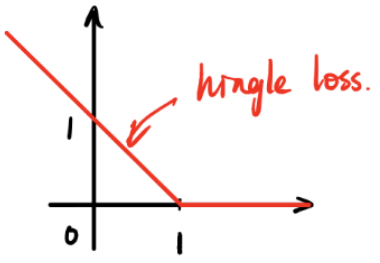

图 1: loss function 的展示图

这个 loss function 已经是连续的了，而且看起来是不是很像书的开着的样子。所以，它有一个非常形象的名字也就是 “合页函数” (Hinge loss)。那么到这里，我们的 Soft Margin SVM 可以被定义为：

$$\begin{cases} \min \ \frac{1}{2}w^T w + C\sum_{i=1}^{N} \max\{0, 1 - y_i(w^T x_i + b)\} \\ s.t. \quad y_i(w^T x_i + b) \geq 1 \end{cases} \tag{4}$$

但是，这样写显然不是我们想要的形式，我们需要得到更简便一些的写法。我们引入 $\xi_i = 1 - y_i(w^T x_i + b)$, $\xi_i \geq 0$。我们仔细的想一想 $\max\{0, 1 - y_i(w^T x_i + b)\}$ 和 ξ_i 之间的关系。有了 $\xi_i \geq 0$，我们可以得到其实 $\xi_i \geq 0$ 和 $\max\{0, 1 - y_i(w^T x_i + b)\}$ 实际上是等价的。那么这个优化模型我们可以写成:

$$\begin{cases} \min \ \frac{1}{2}w^T w + C\sum_{i=1}^{N} \xi_i \\ s.t. \quad y_i(w^T x_i + b) \geq 1 - \xi_i, \ \xi_i \geq 0 \end{cases} \tag{5}$$

在图像上表示即为：

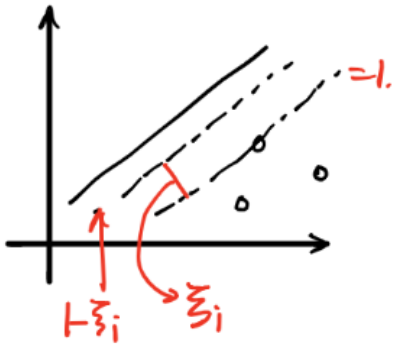

图 2: Soft Margin SVM 模型展示图

在以前的基础上我们增加了一个缓冲区，由于这个缓冲区的存在我们可以允许有点点的误差。而支持向量的区间被放宽到了 $1 - \xi_i$。

Support Vector Machine 03 Weak Duality Proof

Chen Gong

16 November 2019

在前面我们已经展示的 Hard Margin 和 Soft Margin SVM 的建模和求解。前面提到的 SVM 有三宝，间隔，对偶，核技巧。前面我们已经分析了间隔，大家对于其中用到的对偶，虽然我们用比较直觉性的方法进行了解释，但是估计大家还是有点懵逼。这节我们希望给到通用性的证明，这里实际上就是用到了约束优化问题。

1 弱对偶性证明

首先，我们需要证明约束优化问题的原问题和无约束问题之间的等价性。

1.1 约束优化问题与无约束问题的等价性

对于一个约束优化问题，我们可以写成：

$$\begin{cases} \min_{x\in\mathcal{R}^p} f(x) \\ s.t. \quad m_i(x) \le 0,\ i=1,2,\cdots,N \\ \qquad n_i(x) = 0,\ i=1,2,\cdots,N \end{cases} \tag{1}$$

我们用拉格朗日函数来进行表示：

$$\mathcal{L}(x,\lambda,\eta) = f(x) + \sum_{i=1}^{N}\lambda_i m_i + \sum_{i=1}^{N}\eta_i n_i \tag{2}$$

我们可以等价的表示为：

$$\begin{cases} \min_x \max_{\lambda,\eta}\ \mathcal{L}(x,\lambda,\eta) \\ s.t.\ \lambda_i \ge 0,\ i=1,2,\cdots,N \end{cases} \tag{3}$$

这就是将一个约束优化问题的原问题转换为无约束问题。那么这两种表达形式一定是等价的吗？我们可以来分析一下：

如果，x 违反了约束条件 $m_i(x) \le 0$，那么有，$m_i(x) > 0$。且 $\lambda_i > 0$，那么很显然 $max_\lambda\ \mathcal{L} = +\infty$。

如果，x 符合约束条件 $m_i(x) \le 0$，那么很显然 $max_\lambda\ \mathcal{L} \ne +\infty$。

那么：

$$\min_x \max_{\lambda,\eta}\ \mathcal{L}(x,\lambda,\eta) = \min_x \{\max\ \mathcal{L}, +\infty\} = \min_x \{\max\ \mathcal{L}\} \tag{4}$$

其实大家可以很明显的感觉到，这个等式自动的帮助我们过滤到了一半 $m_i(x) \ge 0$ 的情况，这实际上就是一个隐含的约束条件，帮我们去掉了一部分不够好的解。

1.2 证明弱对偶性

原问题我们可以写为：

$$\begin{cases} \min_x \max_{\lambda,\eta} \ \mathcal{L}(x,\lambda,\eta) \\ s.t. \ \lambda_i \geq 0, \ i = 1, 2, \cdots, N \end{cases} \tag{5}$$

而原问题的对偶问题则为：

$$\begin{cases} \min_{\lambda,\eta} \max_x \ \mathcal{L}(x,\lambda,\eta) \\ s.t. \ \lambda_i \geq 0, \ i = 1, 2, \cdots, N \end{cases} \tag{6}$$

原问题是一个关于 x 的函数，而对偶问题是一个关于 λ, η 的最小化问题，而弱对偶性则可以描述为：对偶问题的解 $\leq$ 原问题的解。为了简化表达，后面对偶问题的解我们用 d 来表示，而原问题的解我们用 p 来表示。那么我们用公式化的语言表达也就是：

$$\min_{\lambda,\eta} \max_x \ \mathcal{L}(x,\lambda,\eta) = d \leq \min_x \max_{\lambda,\eta} \ \mathcal{L}(x,\lambda,\eta) = p \tag{7}$$

在前面我们使用感性的方法证明了 $\max\min \ \mathcal{L} \leq \min\max \ \mathcal{L}$，下面我们给出严谨的证明：

很显然可以得到：

$$\min_x \ \mathcal{L}(x,\lambda,\eta) \leq \mathcal{L}(x,\lambda,\eta) \leq \max_{\lambda,\eta} \ \mathcal{L}(x,\lambda,\eta) \tag{8}$$

那么，$\min_x \ \mathcal{L}(x,\lambda,\eta)$ 可表示为一个与 x 无关的函数 $A(\lambda,\eta)$，同理 $\max_{\lambda,\eta} \ \mathcal{L}(x,\lambda,\eta)$ 可表示为一个与 λ, η 无关的函数 $B(x)$。显然，我们可以得到一个恒等式：

$$A(\lambda,\eta) \leq B(x) \tag{9}$$

那么接下来就有：

$$\begin{aligned} A(\lambda,\eta) &\leq \min \ B(x) \\ \max \ A(\lambda,\eta) &\leq \min \ B(x) \\ \min_{\lambda,\eta} \max_x \ \mathcal{L}(x,\lambda,\eta) &\leq \min_x \max_{\lambda,\eta} \ \mathcal{L}(x,\lambda,\eta) \end{aligned} \tag{10}$$

弱对偶性，证毕!!

Support Vector Machine 04 Weak Duality Geometric Interpretation

Chen Gong

17 November 2019

上一小节中我们讨论了有关弱对偶性的证明，这一节我们从几何的角度来解释一下有关对偶问题。为了方便描述，我们将对偶问题进行简化为如下形式：

$$\begin{cases} \min_{x\in\mathcal{R}^p} \ f(x) \\ s.t. \quad m_i \leq 0 \end{cases} \tag{1}$$

$\mathbb{D}$：定义域，$D = dom\ f \cap dom\ m_i$，这是一种常见的定义域的表示方法。其中，$x \in \mathbb{D}$。我们将模型表达为拉格朗日函数的形式，

$$\mathcal{L}(x,\lambda) = f(x) + \lambda m_1(x), \qquad \lambda \leq 0 \tag{2}$$

我们将原问题的最优解记为：$p^* = \min\ f(x)$。
我们将对偶问题的最优解记为：$d^* = \max_\lambda \min_x\ \mathcal{L}(x,\lambda)$。

1 模型表述

上述表述中，表达了模型的基本问题，下面我们进一步抽象模型。首先，我们需要描述一个集合：

$$G = \{(m_1(x), f(x)) | x \in \mathbb{D}\} \tag{3}$$

为了简化运算，我们需要简化符号，令 $m_1(x) = \mu$，$f(x) = t$。那么，

$$G = \{(\mu, t) | x \in \mathbb{D}\} \tag{4}$$

我们需要想想如何集合话来表示，首先 $p^* = \min\ f(x) = \min\ t$，其中，$\{t|(\mu,t) \in G\}$。那么，我们用 inf 来表示下确界的意思，就有：

$$p^* = inf\ \{t|(\mu,t) \in G, \mu \leq 0\} \tag{5}$$

那么对偶问题，我们可以写成，

$$d^* = \max_\lambda \min_x\ \mathcal{L}(x,\lambda) = \max_\lambda \min_x (t + \lambda\mu) \tag{6}$$

又因为 $(t + \lambda\mu)$ 只和 λ 有关，那么可以被记做 $g(\lambda)$。而且，$g(\lambda)$ 可以被写作，$g(\lambda) = inf(t + \lambda\mu)|(\mu,t) \in G$。在对偶条件中不需要那个 $\mu \leq 0$，因为已经包含在原等式的隐藏条件里了。但是，在原问题中，我们一定不能忘记这个条件。

2 模型表达

2.1 p^* 的几何表示

下一步的主要问题就是，我们需要如何来表达 p^* 和 $d^*(g(\lambda))$。首先我们来看 p^*，其实它的表达还算比较简单。我们来看这个图像的表达式：

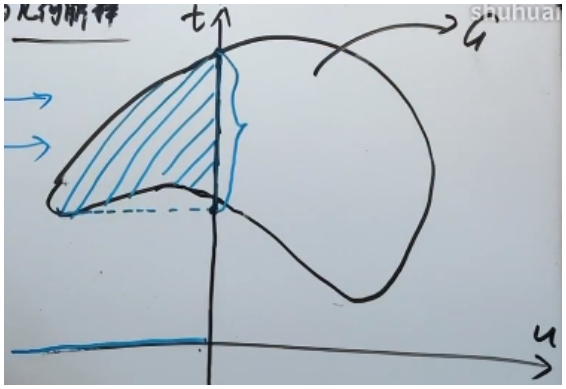

图 1: p^* 的几何表示

我们假设 G 就是 (μ, t) 的定义域的几何表示区间。$p^* = inf\{t|(\mu, t) \in G, \mu \leq 0\}$，由于 $\mu \leq 0$，所以我们只看左边一半。那么 t 的值就是坐标纵轴上的一截部分。最小值非常的好确定，就是平行于 μ 轴，最下方的切点。

2.2 $d^*(g(\lambda))$ 的几何表示

这个等式的几何表示就会有点困难了，我们需要分解成两步，第一步确定 $g(\lambda)$ 的几何表达；第二步，确定 d^* 的几何表达。

2.2.1 $g(\lambda)$ 的几何表达

由于 $t + \lambda\mu$ 是一个关于 x 的变量，在这其中 λ 起到的是一个斜率的作用，这个斜率是一直保持不变的。而得到的 $t + \lambda\mu$ 的结果我们记为 △。△ 也就是 $t + \lambda\mu$ 和 t 轴的交点。那么，也就是一根固定斜率的直线在 t 的方向上进行移动。

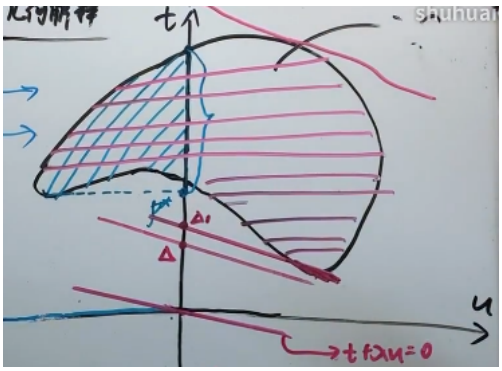

图 2: $g(\lambda)$ 的几何表达

我们可以假设 $t+\lambda\mu$ 与 t 轴的交点是一个集合，这个集合就是 $\{\triangle_1, \triangle_2, \cdots, \triangle_N\}$。

2.2.2 d^* 的几何表达

下一步，我们需要求的是 $d^* = \max_\lambda g(\lambda)$。现在相当于是固定了一个点，然后围着这个点在转。这个点是哪个店呢？就是 $(0, t)$。大家仔细想一想比对一下上图就知道是不是转到与集合 G 相切的时候得到的这个解是最优解，但是这个解一定会比 p^* 得到的解会更小。为什么？用屁股想都知道，一个是横着切，一个是斜着切，哪个会更小？不言而喻了吧。通过这个我们也可以得到，

$$d^* \leq p^* \tag{7}$$

3 小结

上面我们从几何的角度来重新解释了这个问题，其实仔细的想一想也不算很难。但是，强对偶性的证明这个东西有点难，实际上学习机器学习中的 SVM，学到这就差不多够了。如果是强对偶性，我们还需要满足两个条件，也就是 1. 是一个凸优化问题；2. slate 条件。就可以得到 $d^* = p^*$。下一节会进一步解释，但是这只是一个充分必要条件，满足其他的条件也可能是强对偶关系。而 SVM 是一个二次规划问题，那么它一定是一个强对偶问题。

Support Vector Machine 05 Slate & KKT Condition

Chen Gong

18 November 2019

首先，我们整理一下前面得到的有关约束优化的模型。我们可以描述为：

$$\begin{cases} \min\ f(x) \\ s.t.\quad m_i(x) \le 0,\ i = 1, 2, \cdots, M \\ \qquad n_j(x) = 0,\ j = 1, 2, \cdots, N \end{cases} \tag{1}$$

其中，

$$D = \left\{ dom\ f \bigcap_{i=1}^{M} dom\ m_i \bigcap_{j=1}^{N} dom\ n_j \right\} \tag{2}$$

我们将模型进行简化可得：

$$\begin{cases} \min\ f(x) \\ s.t.\quad m_i(x) \end{cases} \quad \Longrightarrow G = \{(m, f) | x \in D\} = \{(\mu, t) | x \in D\} \tag{3}$$

那么，我们的优化目标为：

$$p^* = inf\{t | (\mu, t) \in G, \mu \le 0\} \tag{4}$$

$$g(\lambda) = inf(t + \lambda\mu | (\mu, t) \in G) \tag{5}$$

通常来说，凸优化问题，不一定是强对偶问题。往往都是凸优化问题需要加上一些限定条件才可以构成强对偶问题。比如说 slate condition，但是这些条件往往都是充分非必要的。这样的条件有很多种，slate condition 只是其中一种，类似的还有 KKT condition。

1 Slate Condition

下面简述一下 Slate Condition，详细的证明过程就不做过多的描述。也就是 $\exists \hat{x} \in relint\ D,\ s.t. \forall i = 1, 2, \cdots, m,\ m_i(\hat{x}) \le 0$。而 relint 的意思就是，relative interior，相对内部的意思。

而对于绝大部分的凸优化问题，通常 Slate 条件是成立的。而放松的 Slate 条件为：假设 M 中有 k 个仿射函数，$M - k$ 个仿射。而 SVM 是一个典型的凸二次规划问题，也就是目标函数 f 是凸函数，m_i 是仿射函数，n_j 为仿射。那么在几何上是什么意思呢？也就是限制至少有一个点在坐标系的左边，限制直线不是垂直的，这里需要结合 Support Vector Machine 04 中的几何解释来看。

2 KKT Condition

在上文中我们知道了 Convex 和 Slate Condition 可以得到强对偶关系，也就是 $d^* = p^*$。但是这只是一个充分非必要条件。同样的在满足 KKT Condition 的情况下，我们也可以得出是一个强对偶问题，并且这是一个充分必要的条件。

我们在来回顾一下模型的原问题：

$$\begin{cases} \min \ f(x) \\ s.t. \quad m_i(x) \le 0, \ i = 1, 2, \cdots, M \\ \qquad n_j(x) = 0, \ j = 1, 2, \cdots, N \end{cases} \tag{6}$$

而拉格朗日形式的表达为：

$$\mathcal{L}(x, \lambda) = f(x) + \sum_i \lambda_i m_i(x) + \sum_j \eta_j n_j(x) \tag{7}$$

对于对偶问题，我们可以描述对应的 $g(\lambda, \eta) = \min_x \mathcal{L}(x, \eta, \lambda)$；$d* \longleftarrow \lambda^*, \eta^*$。所以对偶问题 (Dual Prob) 也就是：

$$\begin{cases} \max_{\lambda,\eta} \ g(\lambda, \eta) \\ s.t. \quad \lambda_i \ge 0, \ i = 1, 2, \cdots, M \end{cases} \tag{8}$$

下面进行 KKT 条件的推导：

首先一定需要满足的是，在可行域以内。所以，一定会有：$m_i(x^*) \le 0, n_i(x^*) = 0, \lambda^* \ge 0$。并且还需要满足：

$$\begin{aligned} d^* =& \max_{\lambda,\eta} g(\lambda, \eta) = g(\lambda^*, \eta^*) \\ =& \min_x \mathcal{L}(x, \lambda^*, \eta^*) \\ \le& \mathcal{L}(x, \lambda^*, \eta^*), \quad \forall x \in D \\ =& \mathcal{L}(x^*, \lambda^*, \eta^*) \\ =& f(x^*) + \sum_i \lambda_i m_i(x^*) + \sum_j \eta_j n_j(x^*) \\ =& f(x^*) + \sum_i \lambda_i m_i(x^*) \end{aligned} \tag{9}$$

上式中的 $f(x^*)$ 也就是 p^*，用因为 $\lambda_i m_i(x^*) \le 0$ 是必然存在的。所以，$d^* \le f(x^*)$。这就是弱对偶关系，如果是强对偶关系，就需要我们需要在两个小于或等于号那取等才行。

第一，对于 $\forall i = 0, 1, 2, \cdots, M$，都有 $\sum_i \lambda_i m_i = 0$。

第二，$\min \mathcal{L}(x, \lambda^*, \eta^*), \quad \forall x \in D = \mathcal{L}(x^*, \lambda^*, \eta^*)$。也就是：

$$\frac{\partial \mathcal{L}(x, \lambda^*, \eta^*)}{\partial x} \mid_{x=x^*} = 0 \tag{10}$$

所以，KKT 条件就已经完成了，我们总结一下，KKT 条件分成 3 个部分。

1. 可行条件：也就是需要满足定义域的条件，$m_i(x^*) \le 0, n_i(x^*) = 0, \lambda^* \ge 0$。
2. 互补松弛条件：$\lambda_i m_i = 0$。
3. 梯度为零：$\frac{\partial \mathcal{L}(x, \lambda^*, \eta^*)}{\partial x} \mid_{x=x^*} = 0$。

我们可以对比之前学习的 SVM 的 KKT 条件。

Kernel Method 01 Background

Chen Gong

20 November 2019

在 Support Vector Machine 的章节中，我们已经分析了支持向量机前面“两宝”，也就是间隔和对偶，而第三宝，核技巧在这里我们需要抽出来将分析。其实，我最开始学习核的时候，真的是一脸懵逼，这玩意到底是个什么鬼？来龙去脉是什么？这这节有关于 Kernel Method 的背景介绍中，我想分析一下，我们为什么要使用核？以及怎么用核？来给大家一个直观的感受。

本小节主要从 Kernel Method，Kernel Function 和 Kernel Trick，三个方面来进行分析和讨论，我们为什么要用核？我们怎么样用核？

1 Kernel Method

核方法是一种思想，在 Cover Theorem 中提出：高维空间比低维空间更容易线性可分。这句话非常的直观，我们想想就理解了，这里我不做出详细的证明。在这里我们举一个例子，对于经典的线性不可分问题异或问题，图像描述如下所示：

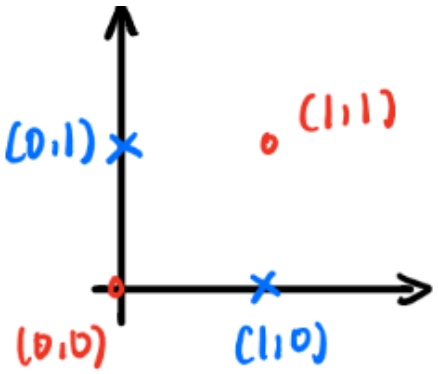

图 1: 异或问题的图像在二维空间中的描述

这二维空间中的点可以被我们记为 $X = (x_1, x_2)$，如果我们使用一个变换函数，将其变换到三维空间中就会发生有意思的事情。我们设定一个变换函数为 $\phi(X)$，将二维空间中的点，变换到一个三维空间 Z 中，并且令 $Z = (x_1, x_2, (x_1 - x_2)^2)$，那么我们再来看看异或问题在三维空间中点的分布，我们惊奇的发现变得线性可分了：

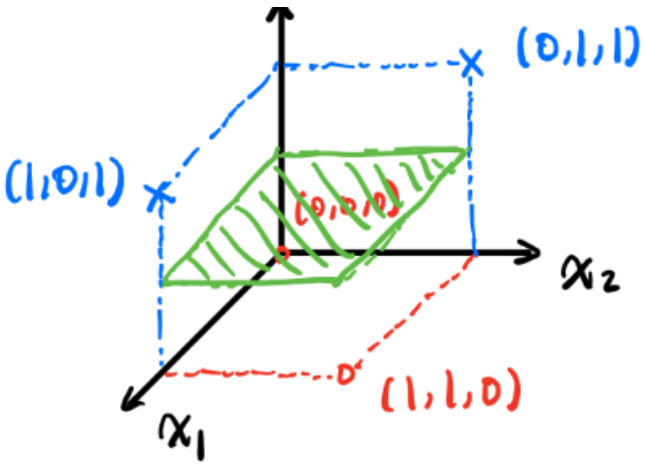

图 2: 异或问题的图像在三维空间中分布

通过这个例子，想想大家都直观的感受到了高维空间带来的好处。实际上对于解决非线性问题，我们有两种思路：

1. 也就是之前提到的，由 Perceptron Layer Analysis (PLA) 引出的多层感知机 (Multilayer Perceptron) 也就我们经常听到的神经网络，以及之后发展得到的 Deep Learning。

2. 而第二种思路就是通过非线性变换 $\phi(x)$，将非线性可分问题转换为线性可分问题。上述的异或问题，可以表述为：

$$\mathcal{X} = (x_1, x_2) \overset{\phi(x)}{\longmapsto} \mathcal{Z} = (x_1, x_2, (x_1 - x_2)^2) \tag{1}$$

第二类方法也就是我们讨论的重点，其实在我们机器学习理论的研究中，第二种方法有很大的威力，大部分同学在学习的时候都会忽掉，例子可以看看之前发的再生核希尔伯特空间。

2 Kernel Function

核函数，从模型的角度讲可以带来给非线性带来高维的转换，这个我们上面已经分析过了。从优化的角度讲可以为对偶带来内积，这两个角度可以合在一起看看。

以我们之前学习的 Hard Margin SVM 为例，原问题和对偶问题的表述为：

$$\begin{cases} \max_{w,b} \ \frac{1}{2} w^T w \\ s.t. \quad 1 - y_i(w_i^T x + b) \leq 0 \end{cases}$$
$$\begin{cases} \min_{\lambda} \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \lambda_i \lambda_j y_i y_j (x_i^T x_j) - \sum_{i=1}^{N} \lambda_i \\ s.t. \quad \lambda_i \geq 0, \ \sum_{i=1}^{N} \lambda_i y_i = 0 \end{cases} \tag{2}$$

在我们的对偶问题中，是不是有一个 $x_i^T x_j$。在线性可分问题中，我们直接计算就好了，在线性不可分问题中，就需要将 x 通过一个变换 ϕ 转换到高维空间中。那么 $x_i^T x_j$ 就变成了 $\phi(x_i^T)\phi(x_j)$。那么我们就将两个角度的分析联系起来了。那么核函数我们可以定义为：

对于一个 $K(x, x') = \phi(x)^T \cdot \phi(x) =< \phi(x), \phi(x') >$,

有 $\forall x, x' \in \mathcal{X}, \exists \phi : x \mapsto z \ s.t. \ K(x, x') = \phi(x)^T \cdot \phi(x')$。则称 $K(x, x')$ 是一个核函数。比如：

$$K(x, x') = exp(-\frac{(x - x')^2}{2\sigma^2}) \tag{3}$$

3 Kernel Track

下面我们需要引入核技巧了，也就是想想，核函数有什么用？前面我们讲到将 x 通过一个变换 ϕ 转换到高维空间。但是，有可能 $\phi(x)$ 的维度非常的高，甚至是无限维的，那么这将变得非常的难求。如果还要继续求 $\phi(x_i^T)\phi(x_j)$，这个计算量恐怕会要原地爆炸。

大家通过上面的表达会发现我们实际上关注的不是 $\phi(x_i)$ 本身，而是 $\phi(x_i^T)\phi(x_j)$。那么，我们完全可直接求跳过求 $\phi(x_i)$ 的过程，然后 $\phi(x_i^T)\phi(x_j)$。我们看看核函数的定义，是不是 $K(x_i, x_j)$ 就等于 $\phi(x_i^T)\phi(x_j)$。这就省去了很多麻烦的计算过程，核函数在这实在是太好用了，这就是核技巧的思想。总的来说，就是非线性转换上的一个内积。

我们为什么引入 kernel? 就是原来的方法有不足，不能解决非线性可分问题。所以，我们想到利用核函数将 $\mathcal{X} \mapsto \mathcal{Z}$，到更高维的空间来转换成线性可分问题。又因为 $\phi(x_i)$ 的计算很难，我们有想到用核函数来直接求 $\phi(x_i^T)\phi(x_j)$。这里面其实是一环扣一环的，逻辑性非常的强。

对于前面讨论的线性可分问题 Perceptron Layer Analysis 和 Hard Margin SVM。允许出现错误就出现了 Pocket Algorithm 和 Soft Margin SVM。进一步如果是严格的非线性问题，引入了 $\phi(x)$ 就得到了 $\phi(x) + PLA$ 和 $\phi(x) + Hargin$ (Kernel SVM)，就是将输入变量的内积转换为核函数。

那么，我们怎么找一个核函数，核函数具有怎样的性质？我们在下一小节中进行分析。

Kernel Method 02 The Definition of Positive Kernel Function

Chen Gong

21 November 2019

上一节中，我们已经讲了什么是核函数，也讲了什么是核技巧，以及核技巧存在的意义是什么。我们首先想想，上一小节我们提到的核函数的定义。

对于一个映射 K，我们有两个输入空间 $\mathcal{X} \times \mathcal{X}, \mathcal{X} \in \mathbb{R}^p$，可以形成一个映射 $\mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$。对于，$\forall\ x, z \in \mathcal{X}$，存在一个映射 $\phi : \mathcal{X} \mapsto \mathbb{R}$，使得 $K(x, z) =< \phi(x), \phi(z) >$。那么这个 $K(\cdot)$，就被我们称为核函数。(<> 代表内积运算)

这既是我们上一节中将的核函数的定义，实际上这个核函数的精准定义，应该是正定核函数。在本小节中，我们将会介绍核函数的精准定义，什么是正定核函数? 并介绍内积和希尔伯特空间 (Hilbert Space) 的定义。这一小节虽然看着会有些枯燥，实际上非常的重要。

1 核函数的定义

核函数的定义，也就是对于一个映射 K，存在一个映射 $\mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$，对于 $x, z \in \mathcal{X}$ 都成立，则称 $K(x, z)$ 为核函数。

对比一下，我们就会发现，这个定义实际上比我们之前学的定义要简单很多。好像是个阉割版，下面我们来介绍两个正定核的定义方法。

2 正定核的定义

正定核函数的定义有两个，我首先分别进行描述一下：

2.1 第一个定义

现在存在一个映射 $K : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$。对于 $\forall x, z \in \mathcal{X}$。如果存在一个 $\phi : \mathcal{X} \mapsto \mathbb{R}^p$，并且 $\phi(x) \in \mathcal{H}$，使得 $K(x, z) =< \phi(x), \phi(z) >$，那么称 $K(x, z)$ 为正定核函数。

2.2 第二个定义

对于一个映射 $K : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$，对于 $\forall x, z \in \mathcal{X}$，都有 $K(x, z)$。如果 $K(x, z)$ 满足，1. 对称性；2. 正定性；那么称 $K(x, z)$ 为一个正定核函数。

我们来分析一个，首先什么是对称性? 这个非常的好理解，也就是 $K(x, z) = K(z, x)$。那么什么又是正定性呢? 那就是任取 N 个元素，$x_1, x_2, \cdots, x_N \in \mathcal{X}$，对应的 Gram Matrix 是半正定的，其中 Gram Matrix 用 K 来表示为 $K = [K(x_i, x_j)]$。

对于第一个对称性，我们其实非常好理解，不就是内积嘛！有一定数学功底的同学一定知道，内积和距离是挂钩的，距离之间一定是对称的。那么正定性就要好好讨论一下了。我们知道这两个定义之间是等价的，为什么会有正定性呢？我们需要进行证明，这个证明可以被我们描述为：

$$K(x,z) =< \phi(x), \phi(z) > \Longleftrightarrow \text{Gram Matrix 是半正定矩阵}$$

这个等式的证明我们留到下一节再来进行，这里我们首先需要学习一个很重要的概念叫做，希尔伯特空间 ($\mathcal{H}$:Hilbert Space)。小编之前被这个概念搞得头晕，特别还有一个叫再生核希尔伯特空间的玩意，太恶心了。

3 Hilbert Space ($\mathcal{H}$)

Hilbert Space 是一个完备的，可能是无限维的，被赋予内积运算的线性空间。下面我们对这个概念进行逐字逐句的分析。

线性空间：也就是向量空间，这个空间的元素就是向量，向量之间满足加法和乘法的封闭性，实际上也就是线性表示。空间中的任意两个向量都可以由基向量线性表示。

完备的：完备性简单的认为就是对极限的操作是封闭的。我们怎么理解呢？若有一个序列为 $\{K_n\}$，这里强调一下 Hilbert Space 是一个函数空间，空间中的元素就是函数。所以，K_n 就是一个函数。那么就会有：

$$\lim_{n \longrightarrow +\infty} K_n = K \in \mathcal{H} \tag{1}$$

所以，我们理解一下就是会和无限维这个重要的概念挂钩。我理解的主要是 Hilbert Space 在无限维满足线性关系。

内积：内积应该满足三个定义，1. 正定性；2. 对称性；3. 线性。下面我们逐个来进行解释：

1. 对称性：也就是 $f, g \in \mathcal{H}$，那么就会有 $< f, g >=< g, f >$。其中，f, g 是函数，我们可以认为 Hilbert Space 是基于函数的，向量是一个特殊的表达。其实，也就是函数可以看成一个无限维的向量。大家在这里是不是看到了无限维和完备性的引用，他们的定义之间是在相互铺垫的。

2. 正定性：也就是 $< f, f >\leq 0$，等号当且仅当 $f = 0$ 是成立。

3. 线性也就是满足：$< r_1 f_1 + r_2 f_2, g >= r_1 < f_1, g > + r_2 < f_2, g >$。

描述上述三条性质的原因是什么呢？也就是我们要证明一个空间中加入一些运算。如果，判断这个运算是不是内积运算，我们需要知道这个运算满不满足上述三个条件。

现在我们介绍了大致的基本概念了，我们回到这样一个问题，对于正定核我们为什么要有两个定义？这个思想和我们之前学到的 Kernel Trick 非常的类似了，Kernel Trick 跳过了寻找 ϕ 这个过程。因为，直接用定义不好找，

Kernel Method 03 Necessary and Sufficient Conditions

Chen Gong

22 November 2019

在上一小节中，我们描述了正定核的两个定义，并且认为这两个定义之间是相互等价的。下面我们就要证明他们之间的等价性。

1 充分性证明

大家注意到在上一节的描述中，我似乎没有谈到对称性，实际上是因为对称性的证明比较的简单。就没有做过多的解释，那么我重新描述一下我们需要证明的问题。

已知：$K(x,z) =< \phi(x), \phi(z) >$，证：Gram Matrix 是半正定的，且 $K(x,z)$ 是对称矩阵。

对称性：已知：

$$K(x,z) =< \phi(x), \phi(z) > \qquad K(z,x) =< \phi(z), \phi(x) > \tag{1}$$

又因为，内积运算具有对称性，所以可以得到：

$$< \phi(x), \phi(z) >=< \phi(z), \phi(x) > \tag{2}$$

所以，我们很容易得到：$K(x,z) = K(z,x)$，所以对称性得证。

正定性：我们想要证的是 Gram Matrix= $K[K(x_i, x_j)]_{N\times N}$ 是半正定的。那么，对一个矩阵 $A_{N\times N}$，我们如何判断这是一个半正定矩阵？大概有两种方法，1. 这个矩阵的所有特征值大于等于 0；2. 对于 $\forall\alpha \in \mathbb{R}^N$，$\alpha^T A\alpha \geq 0$。这个是充分必要条件。那么，这个问题上我们要使用的方法就是，对于 $\forall\alpha \in \mathbb{R}^N$，$\alpha^T A\alpha \geq 0$。

$$\alpha^T K\alpha = \begin{bmatrix}\alpha_1 & \alpha_2 & \cdots & \alpha_N\end{bmatrix} \begin{bmatrix} K_{11} & K_{12} & \cdots & K_{1N} \\ K_{21} & K_{22} & \cdots & K_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ K_{N1} & K_{N2} & \cdots & K_{NN} \end{bmatrix} \begin{bmatrix}\alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_N\end{bmatrix} \tag{3}$$

$$= \sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_i\alpha_j K_{ij} \tag{4}$$

$$= \sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_i\alpha_j < \phi(x_i), \phi(x_j) > \tag{5}$$

$$= \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j \phi(x_i)^T \phi(x_j) \tag{6}$$

$$= \sum_{i=1}^{N} \phi(x_i)^T \sum_{j=1}^{N} \phi(x_j) \tag{7}$$

$$= \left[\sum_{i=1}^{N} \phi(x_i) \right]^T \left[\sum_{j=1}^{N} \phi(x_j) \right] \tag{8}$$

$$= \left|\left| \sum_{i=1}^{N} \alpha_i \phi(x_i) \right|\right|^2 \geq 0 \tag{9}$$

所以，我们可以得到 K 是半正定的，必要性得证。

2 必要性证明

充分性得到证明之后，必要性的证明就会变得很简单了。这个证明可以被我们描述为：

已知：Gram Matrix 是半正定的，且 $K(x,z)$ 是对称矩阵。证：存在一个映射 $\phi : \mathcal{X} \mapsto \mathbb{R}^p$，使得 $K(x,z) =< \phi(x), \phi(z) >$。

对于我们建立的一个映射 $\phi(x) = K(x,\cdot)$,我们可以得到 $K(x,\cdot)K(z,\cdot) = K(x,z)$。所以有 $K(x,z) = K(x,\cdot)K(z,\cdot) = \phi(x)\phi(z)$。我们就得证了，具体的理解可以参考我之前写的关于可再生核希尔伯特空间的理解。

另外一种证明方法：对 K 进行特征值分解，$K = V\Lambda V^T$，那么令 $\phi(x_i) = \sqrt{\lambda_i} V_i$，于是构造了 $K(x_i, x_j) = \sqrt{\lambda_i \lambda_j} V_i V_j$。

Exponential Family Distribution 01 Introduction

Chen Gong

23 October 2019

本节主要对指数族分布的概念和性质的一个小小的总结。指数族分布是一个广泛存在于机器学习研究中的分布。包括，Guassian 分布，Bernoulli 分布 (类别分布)，二项分布 (多项式分布)，泊松分布，Beta 分布，Dirichlet 分布，Gamma 分布和 Gibbs 分布等。

1 指数族分布的基本形式

指数族分布的基本形式可以表示为：

$$p(x|y) = h(x)exp\left\{\eta^T\varphi(x) - A(\eta)\right\} \tag{1}$$

η：参数向量，$\eta \in \mathbb{R}^p$。

$A(\eta)$：log partition function (对数配分函数)。

$h(x)$：这个函数只和 x 有关系，所以并不是很重要。

η 和 $h(x)$ 的理解比较简单，但是 log partition function 的理解难度比较大。所以，在这里对此函数做出一定的解释。

1.1 log partition function (对数配分函数)

什么是配分函数呢? 我的理解这是一个归一化的函数因子，用来使概率密度函数的积分值为 1。推导过程如下：

$$\begin{aligned} p(x|\theta) =& \frac{\hat{p}(x|\theta)}{z} \\ \int p(x|\theta)dx =& \int \frac{\hat{p}(x|\theta)}{z}dx = 1 \\ z =& \int \hat{p}(x|\theta)dx \end{aligned} \tag{2}$$

而在指数族函数中有关于 $A(\eta)$ 的配分函数的推导如下：

$$
\begin{aligned}
p(x|\eta) =& h(x)exp\{\eta^T\varphi(x)\}exp\{-A(\eta)\} \\
=& \frac{1}{exp\{A(\eta)\}}h(x)exp\{\eta^T\varphi(x)\} \\
\int p(x|\eta)dx =& \int \frac{1}{exp\{A(\eta)\}}h(x)exp\{\eta^T\varphi(x)\}dx = 1 \\
exp\{A(\eta)\} =& \int h(x)exp\{\eta^T\varphi(x)\}dx \\
A(\eta) =& \log \int h(x)exp\{\eta^T\varphi(x)\}dx
\end{aligned} \tag{3}
$$

所以，$A(\eta)$ 被称为带有的 log 的 Partition Function。

2 指数族分布的相关知识

和指数族分布的相关知识，可以用下面这张图表来进行概况。

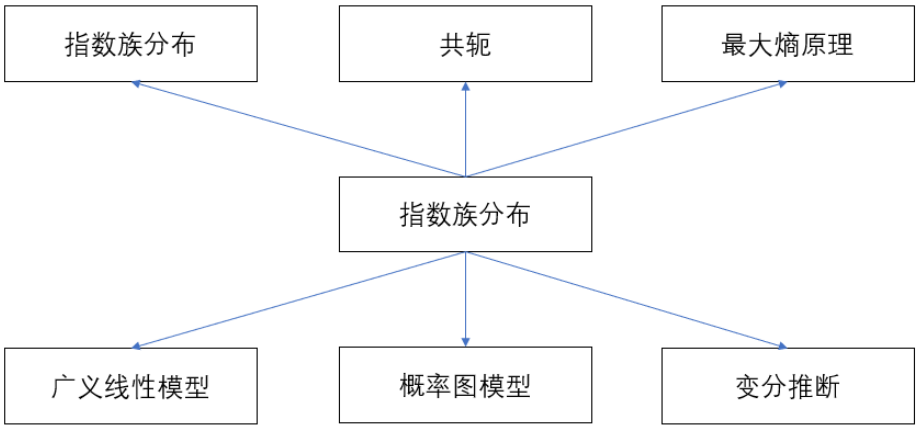

图 1: 指数族分布相关知识表示图

2.1 充分统计量

什么是充分统计量? 我自己的理解，充分统计量是一个有关于样本的函数，有了这个统计量就可以完整的表示出数据集整体的特征。从某种意义上说，我们就可以丢弃样本数据集了。下面对 Guassian Distribution 进行举例，数据集 Data set 为：$\{x_1, x_2, x_3, \cdots, x_N\}$

我们只需要一组充分统计量：

$$
\varphi(x) = \begin{pmatrix} \sum_{i=1}^N x_i \\ \sum_{i=1}^N x_i^2 \end{pmatrix} \tag{4}
$$

就可以反映出 Guassian 的所有特征 $\theta = (\mu, \Sigma)$。充分统计量在 online learning 中的使用有很大的作用。这样可以不记录那么多的数据集，只使用少量的数据就可以估计得到数据集整体的特征，可以用来简化计算。

2.2 共轭分布

为什么要使用共轭的概念呢？首先来看看贝叶斯公式：

$$p(z|x) = \frac{p(x|z)p(z)}{\int_z p(x|z)p(z)dz} \tag{5}$$

在这个公式中，$p(z|x)$ 为后验概率分布，$p(x|z)$ 为似然函数，$p(z)$ 为先验分布。在求解 $\int_z p(x|z)p(z)dz$ 时，计算难度是非常大的。或者说很多时候，根本算不出来。而且，换句话说，就算我们求得了 $p(z|x)$，也有可能因为 $p(z|x)$ 的形式过于复杂，导致 $\mathbb{E}_{p(z|x)}[f(x)]$ 根本算不出来。所以，为了解决这个问题，科研人员们想了很多的办法。近似推断的方法，比如，变分和采样。

变分的方法，是用简单的分布来拟合一个很难计算的分布，从而计算得出 $p(z|x)$ 的近似分布形式。而采样的方法，比如蒙特卡罗采样，隐马尔可夫蒙特卡罗采样 (MCMC) 等，是直接来求 $\mathbb{E}_{p(z|x)}[f(x)]$，这样直接跳过了中间那一堆的过程，在强化学习中经常使用。

而共轭是一种很取巧的方法，**它的效果是使先验和后验有着相同的分布形式，只是参数不同**。这样可以大大的简化计算，解决上述的问题。举例，

$$p(z|x) \propto p(x|z)p(z) \tag{6}$$

如果，$p(x|z)$ 为二项分布，$p(z)$ 为 Beta 分布，那么后验分布 $p(z|x)$ 也为 Beta 分布。

2.3 最大熵原理

下面列举几种确定先验 (prior distribution) 的方法，

1. 共轭，主要是为了计算的简单；
2. 最大熵方法，主要是为了解决无信息先验问题；
3. Jerrif。

最大熵原理会在后面的小节做详细的描述，主要思想就是 “等可能”。也就是尽量使所有的结论等可能的出现，来增加不确定性，保证每一项都是公平的。

2.4 广义线性模型

广义线性模型包括：

1. 线性组合，比如，$w^T x$；
2. link function，也就是激活函数的反函数；
3. 指数族分布，$y|x \sim$ 指数族分布，包括：
   (a) 线性回归，在我们的线性回归模型中，我们曾定义过假设噪声符合 Guassian Distribution，那么 $y|x \sim \mathcal{N}(\mu, \Sigma)$;
   (b) 二分类问题：
      i. $y|x \sim$Bernoulli 分布；
      ii. $y|x \sim$Possion 分布；

2.5 概率图模型和变分推断

包括无向图等，有玻尔兹曼滤波器等。后续的章节会进行详细的描述。变分推断也在后续的章节有详细的描述。

Exponential Family Distribution 02 Example

Chen Gong

23 October 2019

本节的主要内容是演示 Guassian Distribution 的指数族表达形式，将高斯函数的形式转换为指数族分布的通用表达形式。

指数族分布的基本形式可以表示为：

$$p(x|y) = h(x)exp\left\{\eta^T\varphi(x) - A(\eta)\right\} \tag{1}$$

η：参数向量 parameter，$\eta \in \mathbb{R}^p$。

$A(\eta)$：log partition function (配分函数)。

$\varphi(x)$：充分统计量 sufficient statistics magnitude。

1 思路分析

高斯分布的概率密度函数可表示为：

$$p(x|\mu,\sigma^2) = \frac{1}{\sqrt{2\pi}\sigma}exp\left\{-\frac{(x-\mu)^2}{\sigma^2}\right\} \tag{2}$$

观察指数族分布的表达形式，高斯分布的参数向量是有关于 $\theta = (\mu,\sigma)$ 的。首先观察指数部分的第一部分 $\eta^T\varphi(x)$，只有这个部分和 x 相关。那么把这个部分搞定，系数就是参数矩阵，剩下的就是配分函数了，而且配分函数是一个关于 η 的函数。

2 将 Guassian Distribution 改写为指数族分布的形式

具体推导过程如下所示：

$$\begin{aligned}
p(x|\theta) =& \frac{1}{\sqrt{2\pi}\sigma}exp\left\{-\frac{(x-\mu)^2}{2\sigma^2}\right\} && (3)\\
=& \frac{1}{\sqrt{2\pi}\sigma}exp\left\{-\frac{1}{2\sigma^2}(x^2-2\mu x+\mu^2)\right\} && (4)\\
=& \frac{1}{\sqrt{2\pi}\sigma}exp\left\{-\frac{x^2}{2\sigma^2}+\frac{\mu x}{\sigma^2}-\frac{\mu^2}{2\sigma^2}\right\} && (5)\\
=& \frac{1}{\sqrt{2\pi}\sigma}exp\left\{\begin{pmatrix}\frac{\mu}{\sigma^2}\\ -\frac{1}{2\sigma^2}\end{pmatrix}\begin{pmatrix}x & x^2\end{pmatrix}-\frac{\mu^2}{2\sigma^2}\right\} && (6)\\
=& exp\log\frac{1}{\sqrt{2\pi}\sigma}exp\left\{\begin{pmatrix}\frac{\mu}{\sigma^2}\\ -\frac{1}{2\sigma^2}\end{pmatrix}\begin{pmatrix}x & x^2\end{pmatrix}-\frac{\mu^2}{2\sigma^2}\right\} && (7)
\end{aligned}$$

$$=exp\left\{\begin{pmatrix}\frac{\mu}{\sigma^2}\\-\frac{1}{2\sigma^2}\end{pmatrix}\begin{pmatrix}x & x^2\end{pmatrix}-\left(\frac{\mu^2}{2\sigma^2}-\frac{1}{2}\log 2\pi\sigma\right)\right\} \tag{8}$$

令:

$$\eta=\begin{pmatrix}\eta_1\\\eta_2\end{pmatrix}=\begin{pmatrix}\frac{\mu}{\sigma^2}\\-\frac{1}{2\sigma^2}\end{pmatrix}\Longrightarrow\begin{cases}\eta_1=\frac{\mu}{\sigma^2}\\\eta_2=-\frac{1}{2\sigma^2}\end{cases}\Longrightarrow\begin{cases}\mu=-\frac{\eta_1}{2\eta_2}\\\sigma^2=-\frac{1}{2\eta_2}\end{cases} \tag{9}$$

到了现在，我们离最终的胜利只差一步了，

$$\eta=\begin{pmatrix}\eta_1\\\eta_2\end{pmatrix}\quad\varphi(x)=\begin{pmatrix}x\\x^2\end{pmatrix} \tag{10}$$

$$A(\eta)=-\frac{\eta_1^2}{4\eta_2}+\frac{1}{2}\log(2\pi\cdot-\frac{1}{2\eta_2})=-\frac{\eta_1^2}{4\eta_2}+\frac{1}{2}\log(-\frac{\pi}{\eta_2}) \tag{11}$$

于是，Guassian Distribution 成功的被我们化成了指数族分布的形式 $exp\left\{\eta^T\varphi(x)-A(\eta)\right\}$。

Exponential Family Distribution 03 Property

Chen Gong

24 October 2019

本小节主要介绍 Exponential Distribution 中对数配分函数和充分统计量，还有极大似然估计和充分统计量的关系。

指数族分布的基本形式可以表示为：

$$p(x|\eta) = h(x)exp\left\{\eta^T\varphi(x) - A(\eta)\right\} \tag{1}$$

$$p(x|\eta) = \frac{1}{exp\{A(\eta)\}}h(x)exp\left\{\eta^T\varphi(x)\right\} \tag{2}$$

1 对数配分函数和充分统计量

现在有一个问题，那就是我们如何求得对数配分函数 $exp\{A(\eta)\}$，或者说我们可不可以简单的求得对数配分函数。于是，就可以很自然的想到，前面所提到的充分统计量 $\varphi(x)$ 的概念。对数配分函数的目的是为了归一化，那么我们很自然的求出对数配分函数的解析表达式：

$$\begin{aligned}
\int p(x|\eta)dx =& \int \frac{1}{exp\{A(\eta)\}}h(x)exp\left\{\eta^T\varphi(x)\right\}dx \\
\int p(x|\eta)dx =& \frac{\int h(x)exp\left\{\eta^T\varphi(x)\right\}dx}{exp\{A(\eta)\}} = 1 \\
exp\{A(\eta)\} =& \int h(x)exp\left\{\eta^T\varphi(x)\right\}dx
\end{aligned} \tag{3}$$

下一步则是在 $exp\{A(\eta)\}$ 中对 η 进行求导。

$$\begin{aligned}
\frac{\partial exp\{A(\eta)\}}{\partial\eta} =& \nabla_\eta A(\eta)exp\{A(\eta)\} \\
=& \frac{\partial}{\partial\eta}\int h(x)exp\left\{\eta^T\varphi(x)\right\}dx \\
=& \int \frac{\partial}{\partial\eta}h(x)exp\left\{\eta^T\varphi(x)\right\}dx \\
=& \int h(x)exp\left\{\eta^T\varphi(x)\right\}\varphi(x)dx
\end{aligned} \tag{4}$$

将等式的左边的 $exp\{A(\eta)\}$ 移到等式的右边可得，

$$\nabla_\eta A(\eta) = \int h(x)exp\left\{\eta^T\varphi(x) - A(\eta)\right\}\varphi(x)dx \tag{5}$$

$$\nabla_\eta A(\eta) = \int p(x|\eta)\varphi(x)dx \tag{6}$$

$$\nabla_\eta A(\eta) = \mathbb{E}_{x\sim p(x|\eta)}[\varphi(x)] \tag{7}$$

其实通过同样的方法可以证明出，

$$\nabla\eta^2 A(\eta) = Var_{x\sim p(x|\eta)}[\varphi(x)] \tag{8}$$

又因为，协方差矩阵总是正定的矩阵，于是有 $\nabla_\eta^2 A(\eta) \succeq 0$。所以，由此得出 $A(\eta)$ 是一个凸函数。并且，由 $\mathbb{E}_{x\sim p(x|\eta)}[\varphi(x)]$ 和 $Var_{x\sim p(x|\eta)}[\varphi(x)]$ 就可以成功的求解得到 $A(\eta)$ 函数。那么我们做进一步思考，知道了 $\mathbb{E}[x]$ 和 $\mathbb{E}[x^2]$，我们就可以得到所有想要的信息。那么：

$$\mathbb{E}[\varphi(x)] = \begin{pmatrix} \mathbb{E}[x] \\ \mathbb{E}[x^2] \end{pmatrix} \tag{9}$$

2 极大似然估计和充分统计量

假设有一组观察到的数据集：$D = \{x_1, x_2, x_3, \cdots, x_N\}$，那么我们的求解目标为：

$$\begin{aligned} \eta_{MLE} =& argmax \log \prod_{i=1}^{N} p(x_i|\eta) \\ =& argmax \sum_{i=1}^{N} \log p(x_i|\eta) \\ =& argmax \sum_{i=1}^{N} \log h(x_i) exp\left\{\eta^T \varphi(x_i) - A(\eta)\right\} \\ =& argmax \sum_{i=1}^{N} \log h(x_i) + \sum_{i}^{N} \left(\eta^T \varphi(x_i) - A(\eta)\right) \end{aligned} \tag{10}$$

$$\frac{\partial}{\partial \eta} \left\{ \sum_{i=1}^{N} \log h(x_i) + \sum_{i=1}^{N} \left(\eta^T \varphi(x_i) - A(\eta)\right) \right\} = 0 \tag{11}$$

$$\sum_{i=1}^{N} \varphi(x_i) = N \cdot \nabla_\eta A(\eta) \tag{12}$$

$$\nabla_\eta A(\eta) = \frac{1}{N} \sum_{i=1}^{N} \varphi(x_i) \tag{13}$$

或者说，我们可以认为是：$\nabla_\eta A(\eta_{MLE}) = \frac{1}{N}\sum_{i=1}^{N} \varphi(x_i)$。并且，$\nabla_\eta A(\eta_{MLE})$ 是一个关于 η_{MLE} 的函数。那么反解，我们就可以得到 η_{MLE}。所以我们要求 η_{MLE}，我们只需要得到 $\frac{1}{N}\sum_{i=1}^{N} \varphi(x_i)$ 即可。所以，$\varphi(x)$ 为一个充分统计量。

3 总结

在本小节中，我们使用了极大似然估计和对数配分函数来推导了，充分统计量，这将帮助我们理解 Exponential Distribution 的性质。

Exponential Family Distribution 04 Maximum Entropy

Chen Gong

26 October 2019

从这节开始，我们将从最大熵的角度来解析指数族分布。首先，我们需要定义一下什么是熵？所谓熵，就是用来衡量信息反映的信息量的多少的单位。这里我们首先介绍一下，什么是熵?

1 最大熵原理

假设 p 是一个分布，所谓信息量就是分布的对数的相反数 (p 是小于 1 的，为了使信息量的值大于 0)，即为 $-\log p$。而熵则被我们定义为:

$$\begin{aligned}\mathbb{E}_{x\sim p(x)}[-\log p(x)] &= \int_x -p(x)\log p(x)dx \\ &= -\sum_x p(x)\log p(x)\end{aligned} \tag{1}$$

而最大熵原理实际上就可以定义为等可能。这是一种确定无信息先验分布的方法，它的原理就是是所有的可能都尽可能的出现，而不会出现类似于偏见的情况。接下来，我们令

$$H(x) = -\sum_x p(x)\log p(x) \tag{2}$$

假设 x 是离散的，

x	1	2	$\cdots$	k
p	p_1	p_2	$\cdots$	p_k

表 1: 随机变量 x 的概率密度分布情况

并且，需要满足约束条件，

$$s.t. \qquad \sum_{i=1}^{N} p_i = 1 \tag{3}$$

那么，总结一下上述的描述，优化问题可以写为:

$$\begin{cases} \text{argmax} -\sum_x p(x)\log p(x) \\ s.t. \qquad \sum_{i=1}^{N} p_i = 1 \end{cases} \tag{4}$$

可以将其改写为:

$$\begin{cases} \text{argmin} \sum_x p(x)\log p(x) \\ s.t. \qquad \sum_{i=1}^{N} p_i = 1 \end{cases} \tag{5}$$

实际上也就是求 $\hat{p_i} = \text{argmin} -H(p(x))$，其中 $p = \begin{pmatrix} p_1 & p_2 & \cdots & p_k \end{pmatrix}^T$。我们使用拉格朗日乘子法来求带约束的方程的极值。定义损失函数为：

$$\mathcal{L}(p, \lambda) = \sum_{i=1}^{N} p(x_i) \log p(x_i) + \lambda(1 - \sum_{i=1}^{k} p_i) \tag{6}$$

下面是对 $\hat{p_i}$ 的求解过程，

$$\frac{\partial \mathcal{L}}{\partial p_i} = \log p_i + p_i \frac{1}{p_i} - \lambda = 0 \tag{7}$$

解得：

$$p_i = exp(\lambda - 1) \tag{8}$$

又因为 λ 是一个常数，所以 $\hat{p_i}$ 是一个常数，那么我们可以轻易得到

$$\hat{p}_1 = \hat{p}_2 = \hat{p}_3 = \cdots = \hat{p}_k = \frac{1}{k} \tag{9}$$

很显然 $p(x)$ 是一个均匀分布，**那么关于离散变量的无信息先验的最大熵分布就是均匀分布**。

2 指数族分布的最大熵原理

我们首先写出指数族分布的形式：

$$p(x|\eta) = h(x) exp\left\{\eta^T \varphi(x) - A(\eta)\right\} \tag{10}$$

我们可以换一种形式来定义，为了方便之后的计算：

$$p(x|\eta) = \frac{1}{Z(\eta)} h(x) exp\left\{\eta^T \varphi(x)\right\} \tag{11}$$

但是，我们用最大熵原理来求指数族分布的时候，还差一个很重要的东西，也就是经验约束。也就是我们的分布要满足既定的事实上基础上进行运算。那么，我们需要怎么找到这个既定事实的分布呢？假设我们有一个数据集 $Data = \{x_1, x_2, x_3, \cdots, x_N\}$。那么，我们定义分布为，

$$\hat{p}(X = x) = \hat{p}(x) = \frac{Count(x)}{N} \tag{12}$$

那么我们可以得到一系列的统计量 $\mathbb{E}_{\hat{p}}(x)$，$Var_{\hat{p}}(x)$，$\cdots$。那么假设，$f(x)$ 是关于任意 x 的函数向量。那么我们定义 $f(x)$ 为：

$$f(x) = \begin{pmatrix} f_1(x) \\ f_2(x) \\ \vdots \\ f_Q(x) \end{pmatrix} \qquad \Delta = \begin{pmatrix} \Delta_1 \\ \Delta_2 \\ \vdots \\ \Delta_Q \end{pmatrix} \tag{13}$$

其中，假设 $\mathbb{E}_{\hat{p}}[f(x)] = \Delta$(已知)。同样，我们将熵表达出来，

$$H[p] = -\sum_{x} p(x) \log p(x) \tag{14}$$

那么，这个优化问题，可以被我们定义为:

$$\begin{cases} \text{argmin} \sum_x p(x) \log p(x) \\ s.t. \quad \sum_{i=1}^{N} p_i = 1 \\ \qquad \mathbb{E}_p[f(x)] = \mathbb{E}_{\hat{p}}[f(x)] = \Delta \end{cases} \tag{15}$$

其中，我们期望在总体数据上的特征和在给定数据上的特征一致。同样，我们使用拉格朗日乘子法来求带约束的方程的极值。定义损失函数为:

$$\mathcal{L}(p, \lambda_0, \lambda) = \sum_{i=1}^{N} p(x_i) \log p(x_i) + \lambda_0 (1 - \sum_x p) + \lambda^T (\Delta - \mathbb{E}_p[f(x)]) \tag{16}$$

将 $\mathbb{E}_p[f(x)])$ 进行改写为:

$$\mathcal{L}(p, \lambda_0, \lambda) = \sum_{i=1}^{N} p(x_i) \log p(x_i) + \lambda_0 (1 - \sum_x p) + \lambda^T (\Delta - \sum_x p(x) f(x)) \tag{17}$$

我们的目的是求一个 $\hat{p}(x)$，那么使用求偏导的方法（关于一个给定的 x，对于 $p(x)$ 求偏导）:

$$\frac{\mathcal{L}(p, \lambda_0, \lambda)}{p(x)} = \left(\log p(x) + p(x) \frac{1}{p(x)} \right) - \lambda_0 + \lambda^T f(x) = 0 \tag{18}$$

$$\log p(x) + 1 - \lambda_0 - \lambda^T f(x) = 0 \tag{19}$$

$$\log p(x) = \lambda_0 - 1 + \lambda^T f(x) \tag{20}$$

$$p(x) = exp\left\{ \lambda_0 - 1 + \lambda^T f(x) \right\} \tag{21}$$

整理一下即可得到 $p(x) = exp\left\{ \lambda^T f(x) - (1 - \lambda_0) \right\}$，那么我们可以将 $\eta = \begin{pmatrix} \lambda_0 \\ \lambda \end{pmatrix}$，$f(x) = \varphi(x)$，$(1 - \lambda_0) = A(\eta)$。很显然，$p(x)$ 是一个指数族分布。**那么我们可以得到一个结论，一个无先验信息先验的分布的最大熵分布是一个指数族分布**。

Probability Graph 01 Background

Chen Gong

23 November 2019

机器学习的重要思想就是，对已有的数据进行分析，然后对未知数据来进行预判或者预测等。这里的图和我们之前学习的数据结构中的图有点不太一样，俗话说有图有真相，这里的图是将概率的特征引入到图中，方便我们进行直观分析。

1 概率的基本性质

我们假设现在有一组高维随机变量，$p(x_1, x_2, \cdots, x_n)$，它有两个非常基本的概率，也就是条件概率和边缘概率。条件概率的描述为 $p(x_i)$，条件概率的描述为 $p(x_j|x_i)$。

同时，根据这两个基本的概率，我们可以得到两个基本的运算法则：Sum Rule 和 Product Rule。

Sum Rule：$p(x_1) = \int p(x_1, x_2)dx_2$。

Product Rule：$p(x_1, x_2) = p(x_1)p(x_2|x_1) = p(x_2)p(x_1|x_2)$。

根据这两个基本的法则，我们可以推出 Chain Rule 和 Bayesian Rule。

Chain Rule：

$$p(x_1, x_2, \cdots, x_N) = \prod_{i=1}^{N} p(x_i|x_1, x_2, \cdots, x_{i-1}) \tag{1}$$

Bayesian Rule：

$$p(x_2|x_1) = \frac{p(x_1, x_2)}{p(x_1)} = \frac{p(x_1, x_2)}{\int p(x_1, x_2)dx_2} = \frac{p(x_1|x_2)p(x_2)}{\int p(x_1|x_2)p(x_2)dx_2} \tag{2}$$

2 条件独立性

首先，我们想想高维随机变量所遇到的困境，也就是维度高，计算复杂度高。大家想想，当维度较高时，这个 $p(x_1, x_2, \cdots, x_N) = \prod_{i=1}^{N} p(x_i|x_1, x_2, \cdots, x_{i-1})$ 肯定会算炸去。所以，我们需要简化运算，之后我们来说说我们简化运算的思路。

1. 假设每个维度之间都是相互独立的，那么我们有 $p(x_1, x_2, \cdots, x_N) = \prod_{i=1}^{N} p(x_N)$。比如，朴素贝叶斯就是这样的设计思路，也就是 $p(x|y) = \prod_{i=1}^{N} p(x_i|y)$。但是，我们觉得这个假设太强了，实际情况中的依赖比这个要复杂很多。所以我们像放弱一点，增加之间的依赖关系，于是我们有提出了马尔科夫性质 (Markov Propert)。

2. 假设每个维度之间是符合马尔科夫性质 (Markov Property) 的。所谓马尔可夫性质就是，对于一个序列 $\{x_1, x_2, \cdots, x_N\}$，第 i 项仅仅只和第 $i-1$ 项之间存在依赖关系。用符号的方法我们可以表示为：

$$x_j \perp x_{i+1} | x_i, j < i \tag{3}$$

在 HMM 里面就是这样的齐次马尔可夫假设，但是还是太强了，我们还是要想办法削弱。自然界中经常会出现，序列之间不同的位置上存在依赖关系，因此我们提出了条件独立性。

3. 条件独立性：**条件独立性假设是概率图的核心概念。它可以大大的简化联合概率分布**。而用图我们可以大大的可视化表达条件独立性。我们可以描述为：

$$X_A \perp X_B | X_C \tag{4}$$

而 X_A, X_B, X_C 是变量的集合，彼此之间互不相交。

3 概率图算法分类

概率图的算法大致可以分为三类，也就是，表示 (Representation)，推断 (Inference) 和学习 (Learning)。

3.1 Representation

知识表示的方法，可以分为有向图，Bayesian Network；和无向图，Markov Network，这两种图通常用来处理变量离散的情况。对于连续性的变量，我们通常采用高斯图，同时可以衍生出，Gaussian Bayesian Network 和 Guassian Markov Network。

3.2 Inference

推断可以分为精准推断和近似推断。所谓推断就是给定已知求概率分布。近似推断中可以分为确定性推断 (变分推断) 和随机推断 (MCMC)，MCMC 是基于蒙特卡罗采样的。

3.3 Learning

学习可以分为参数学习和结构学习。在参数学习中，参数可以分为变量数据和非隐数据，我们可以采用有向图或者无向图来解决。而隐变量的求解我们需要使用到 EM 算法，这个 EM 算法在后面的章节会详细推导。而结构学习则是，需要我们知道使用那种图结构更好，比如神经网络中的节点个数，层数等等，也就是现在非常热的 Automate Machine Learning。

Probability Graph 02 Bayesian Network

Chen Gong

24 November 2019

概率图模型中，图是用来表达的，将概率嵌入到了图中之后，使得表达变得非常的清晰明了。在我们的联合概率计算中，出现了一些问题：

$$p(x_1, x_2, \cdots, x_N) = \prod_{i=1}^{N} p(x_i|x_{1:i-1}) \tag{1}$$

这样的计算维度太高了，所以我们引入了条件独立性，表达为 $X_A \perp X_B|X_C$。那么采用因子分解的方法我们可以将联合概率的计算进行分解为：

$$p(x_1, x_2, \cdots, x_N) = \prod_{i=1}^{N} p(x_i|x_{pa\{i\}}) \tag{2}$$

其中，$pa\{i\}$ 表示为 x_i 的父亲节点。而概率图可以有效的表达条件独立性，直观性非常的强，我们接下来看看概率图中经典的三种结构。

1 概率图的三种基本结构

对于一个概率图，我们可以使用拓扑排序来直接获得，条件独立性的关系。如果存在一个关系由一个节点 x_i 指向另一个节点 x_j，我们可以记为 $p(x_j|x_i)$。我们现在需要定义一些规则来便于说明，对于一个概率图如下所示：

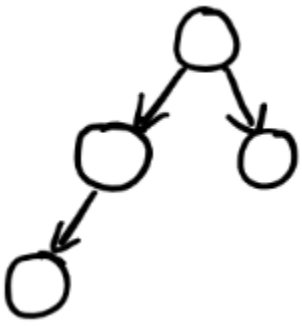

图 1: 基本概率图模型

对于一个箭头 $\longrightarrow$ 来说，箭头所在的方向称为 Head，另一端被称为 Tail。

1.1 Tail to Tail 结构

Tail to Tail 的模型结构图，如下图所示，由于 b 节点在 a 节点和 c 节点的 Tail 部分，所以被我们称为 Tail to Tail 结构。

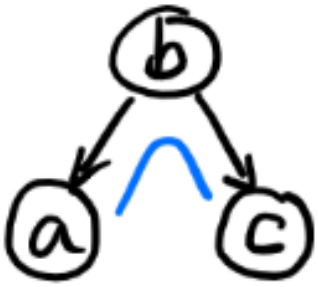

图 2: Tail to Tail 结构示意图

我们使用因子分析来计算联合概率可以得到：

$$p(a,b,c) = p(b)p(a|b)p(c|b) \tag{3}$$

使用链式法则，同样我们也可以得到：

$$p(a,b,c) = p(b)p(a|b)p(c|b,a) \tag{4}$$

对比一下公式 (3) 和公式 (4)，我们可以对比得到：

$$p(c|b) = p(c|b,a) \tag{5}$$

实际上，这里就已经就可以看出 $a \perp c$ 了，因为 a 的条件增不增加都不会改变 c 的概率，所以 a 和 c 之间是相互独立的。可能有的同学还是觉得不好理解，那么我们做进一步的分析：

$$p(c|b)p(a|b) = p(c|b,a)p(a|b) = p(a,c|b) \tag{6}$$

$$\Rightarrow p(c|b)p(a|b) = p(a,c|b) \tag{7}$$

这样，我们就可以看得很明白了。这就是条件独立性，在 a 的条件下，b 和 c 是独立的。实际在概率图中就已经蕴含这个分解了，只看图我们就可以看到这个性质了，这就是图的直观性，条件独立性和图是一样的。那么 $a \perp c$ 可以被我们看为：给定 b 的情况下，如果 b 被观测到，那么 a 和 c 之间是阻塞的，也就是相互独立。

1.2 Head to Tail 结构

图 3: Head to Tail 结构示意图

其实，和 Head to Head 结构的分析基本是上一模一样的，我们可以得到 $a \perp c|b$。也就是给定 b 的条件下，a 和 c 之间是条件独立的。也就是 b 被观测的条件下，路径被阻塞。

1.3 Head to Head 结构

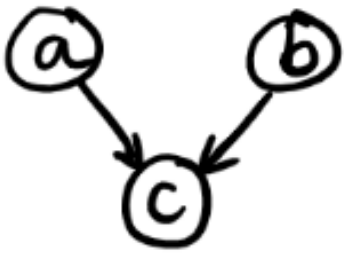

图 4: Head to Head 结构示意图

在默认情况下 $a \perp b$，也就是若 c 被观测，a 和 b 之间是有关系的。我们可以推导一下默认情况。

$$\begin{aligned} p(a,b,c) =& p(a)p(b)p(c|a,b) \\ =& p(a)p(b|a)p(c|a,b) \end{aligned} \tag{8}$$

我们可以得出 $p(b) = p(b|a)$，也就是 $a \perp b$。

Probability Graph 03 D-Separation

Chen Gong

25 November 2019

上一小节中，我们已经大致介绍了概率图之间的三种基本拓扑结构。下面我们来介绍一下，这三种拓扑结构的运用，以及如何扩展到我们的贝叶斯模型中。

1 D-separation

假设我们有三个集合，X_A, X_B, X_C，这三个集合都是可观测的，并且满足 $X_A \perp X_C | X_B$。那我们想想，如果有一些节点连成的拓扑关系图，如果一个节点 $a \in X_A, c \in X_C$，那么如果 a 和 c 之间相互独立的话，他们之间连接的节点需要有怎样的条件？我们通过一个图来进行描述。

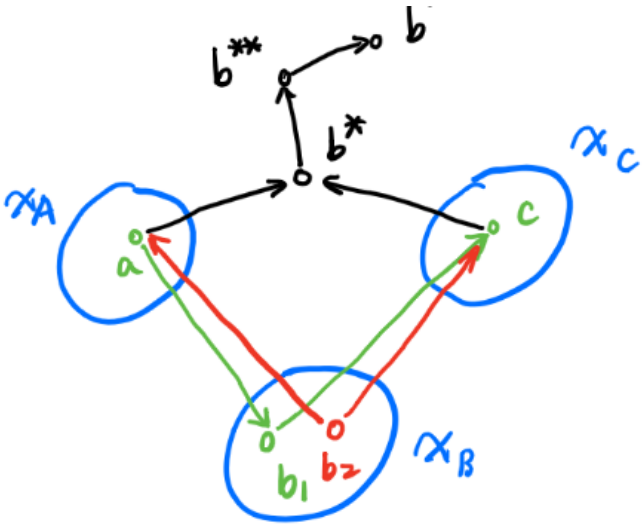

图 1: 节点之间的拓扑联系图

根据上一小节我们的讲解，我们可以想想，假如从 a 到 c 中间通过一个节点 b，那么路径什么时候是连通的？什么时候又是阻塞的呢？我们可以分两种情况来讨论，为什么是两种？上一节我们就已经讲解过，Tail to Tail 结构和 Head to Tail 结构其实是一样的，但是 Head to Head 结构是反常的。所以我们就分开进行讨论。

1. 如果是 Tail to Tail 结构和 Head to Tail 结构，那么中间节点 b_1, b_2 必然要位于可观测集合 X_B 中，那么 a 和 c 才是相互独立的。

2. 如果是 Head to Head 结构，那就不一样了，在一般情况下 $a \perp c$，那就是 $b \notin X_B$，包括他的子节点都不可以被观测到，不然就连通了。

如果，符合上述两条规则，那么我们就可以称 a 和 c 之间是 D-Separation 的，实际上还有一个名字，叫做全局马尔科夫性质 (Global Markov Property)。D-Separation 非常的关键，我们可以直接用这个性质来检测两个集合关于另外一个集合被观测的条件是不是条件独立。

2 Markov Blanket

我们首先定义 x_{-i} 为 $\{x_1, x_2, \cdots, x_{i-1}, x_{i+1}, \cdots, x_N\}$，这个序列中唯独不包括 x_i。那么，我们假设除了 x_i 节点，其他节点都是可观测的，那么我们需要计算概率：

$$p(x_i|x_{-i}) = \frac{p(x_i, x_{-i})}{p(x_{-i})} = \frac{p(x)}{\int_{x_i} p(x)dx_i} = \frac{\prod_{j=1}^{N} p(x_j|x_{pa(j)})}{\int_{x_i} \prod_{j=1}^{N} p(x_j|x_{pa(j)})dx_i} \tag{1}$$

我们分析一下上述等式，我们可以分成两部分，将和 x_i 相关的部分记为 $\bar{\Delta}$，和 x_i 不相关的部分记为 Δ。那么公式 (1) 可以被我们改写为：

$$p(x_i|x_{-i}) = \frac{\Delta \cdot \bar{\Delta}}{\int_{x_i} \Delta \cdot \bar{\Delta} dx_i} = \frac{\Delta \cdot \bar{\Delta}}{\Delta \cdot \int_{x_i} \bar{\Delta} dx_i} == \frac{\bar{\Delta}}{\int_{x_i} \bar{\Delta} dx_i} \tag{2}$$

我们可以将 $p(x_i|x_{-i})$ 表示为一个函数 $f(\bar{\Delta})$。那么 x_i 和其他所有点的关系可以被化简为只和 x_i 相关的点的关系。那么，我们将这个关系大致抽象出来，通过图来进行分析，找一找哪些节点是和 x_i 相关的，直观性也是概率图模型的一大优点。假设 x_i 是可以被观测到的

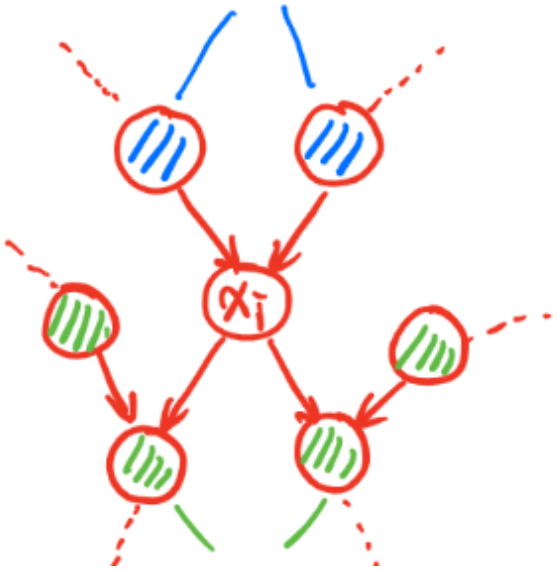

图 2: 节点之间的抽象拓扑联系图

$P(x_i|x_{-1})$ 首先包括一部分为 $p(x_i|x_{pa(i)})$，第二部分为 $p(x_{child(i)}|x_i, x_{pa(child(x_i))})$。为什么第二部分可以这样写呢？首先 x_i 作为两个父亲节点，得到两个孩子节点，毫无疑问对吧，那么我们可以写成 $p(x_{child(i)}|x_i)$。但是和 $p(x_{child(i)})$ 相关的变量除了 x_i 肯定还有其他的，也就是 $x_{pa(child(x_i))}$，所以我们就可以得到 $p(x_{child(i)}|x_i, x_{pa(child(x_i))})$。实际上，这些 x_i 周围的节点，也就是我用阴影部分画出的那些，可以被称为 Markov Blanket。从图上看就是留下和父亲，孩子，孩子的另一个双亲，其他的节点可以忽略，也就是和周围的关系的连接。

总结：D-Separation 是在概率图中，帮助我们快速的判断条件独立性。

Probability Graph 04 Example

Chen Gong

26 November 2019

上一节中，我们讲的是模型通用的一些概念，这一节开始，我们要讲一讲贝叶斯网络具体的例子。我们从单一，混合，时间和连续，四个角度来看看 Bayesian Network，这个四个方法是一步一步越来越难的。

1 单一

单一最典型的代表就是 Naive Bayesian，这是一种 classification 的模型。对于 $p(x|y)$ 的问题来说，假设各维度之间相互独立，于是就有：

$$p(x|y) = \prod_{i=1}^{N} p(x_i|y=1) \tag{1}$$

概率图模型表示如下所示：

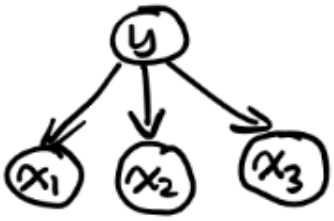

图 1: Naive Bayesian Network 拓扑表示

很显然是一个 Tail to Tail 的模型，我们很简单可以得出 $x_1 \perp x_2 \perp \cdots \perp x_N$。

2 混合

最常见的就是 Gaussian Mixture Model (GMM)，这是一种聚类模型，将每个类别拟合一个分布，计算数据点和分布之间的关系来确定，数据点所属的类别。我们假设 Z 是一个隐变量，并且 Z 是离散的变量，那么 $x|z \sim \mathcal{N}(\mu, \Sigma)$。我们用模型可以表示为：

图 2: Gaussian Mixture Model 的可视化表示

3 时间

时间上我们大致可以分成两种。第一种是 Markov chain，这是随机过程中的一种；第二种事 Gaussian Processing，实际上就是无限维的高斯分布。

实际上时间和混合可以一起看，我们称之为动态系统模型。并且，我们就可以衍生出三种常见的模型，这里讲的比较的模糊，在后面的章节我们会进行详细的分析。第一种是隐马尔可夫模型 (HMM)，这是一种离散的模型；第二种是线性动态系统 (LDS)，这是一种线性的连续的模型，包括典型的 Kalman Filter。第三种是 Particle Filter，一种非高斯的，非线性的模型。

4 连续

连续就是 Guassian Bayesian Network，前面有提到过。

大家可能听到这么多的名词会一脸懵逼呀，懵逼是很正常的，因为这些名词只是给了个印象，后面我们会进行详细的分析。实际上一个整体的趋势就是从单一到混合，从有限到无限。也就是从空间和时间两个角度来进行分析，都是从离散到连续的过程。至于具体为什么这么分，还得具体学习了算法我们才能够很好的理解，本小节只是起了一个高屋建瓴的作用。

Probability Graph 05 Markov Network

Chen Gong

27 November 2019

上一小节中，我们分析了有向图 Bayesian Network，得到了因子分解法，$p(x) = \prod_{i=1}^{N} p(x_i|x_{pa(i)})$。虽然，有向图中可以方便直观的表达条件独立性，但是它也有它的局限性。也就是我们提到的对于 Head to Head 的结构来说，当中间节点被观察到的时候，反而是两端的节点是相关的。这违反了条件独立性的特点，也就是当某些变量被观察到时，其他变量之间是独立的特点，这种情况有点反常，并不太好办。

但是，在无向图中就完全不会出现这样的情况，因为本来就没有方向，而且在无向图中也有类似的 D-Separation 性质。

1 Condition Independence in Markov Network

Markov 中的条件独立，大致可以被我们分成三种情况，Global Markov，Local Markov 和 Pair Markov。

1.1 Global Markov

假设现在有三个集合 $X_A \perp X_B | X_C$，我们想得到 $a \in X_A, b \in X_B$ 之间相互独立，这个应该怎么办？我们给出，只有 a 和 b 的中间节点至少有一个位于 c 中，那么我们就可以得到 $a \perp c$。

1.2 Local Markov

我们以下图的一个 Markov Network 为例，

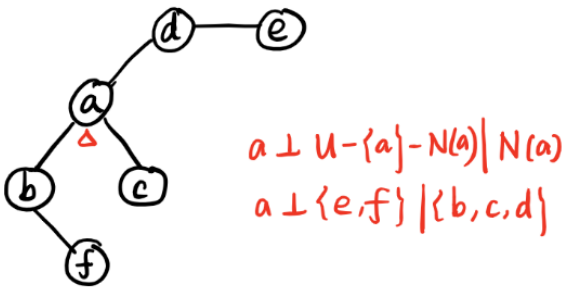

图 1: Markov Network 示意图

用文字的语言来描述就为 $a\perp\{$ 全集 $-a-$ 邻居 $\}|$ 邻居。那么在这个图中，我们就可以表示为 $a\perp\{e,f\}|\{b,c,d\}$。

1.3 Pair Markov

成对马尔可夫性质可以被我们描述为：$x_i\perp x_j|x_{-i-j}$ $(i\neq j)$。其中，x_{-i-j} 为从全集中去掉 x_i 和 x_j 而留下了的集合。

那么条件独立性就可以提现在，Global，Local 和 Pair 中。其中 Global⇔Local⇔Pair。也就是这三种条件独立的方法得到的结果是一样的。

2 因子分解法

我们想一想在一个无向图中，如何来体现我们想要的条件独立性。这里的引入和之前的不太一样，我们首先需要引入几个概念。**团**: 这是一个关于节点的集合，集合中的节点是相互连通的。而最大团，就很好理解了吧，也就是最大的连通子集。我们可以将无向图的分离定义到团上，我们假设 $c_1,c_2,\cdots,c_k$ 表示为团。那么，我们可以将联合概率定义为：

$$p(x)=\frac{1}{Z}\prod_{i=1}^{k}\phi_i(x_{c_i}) \tag{1}$$

其中，z 是归一化因子，因为没有归一化因子的话，这不能被称为一个概率分布，因为概率分布首先就要保证和等于 1。那么，z 被定义为

$$z=\sum_{x}\prod_{i=1}^{k}\phi_i(x_{c_i})=\sum_{x_1}\cdots\sum_{x_N}\prod_{i=1}^{k}\phi_i(x_{c_i}) \tag{2}$$

3 Gibbs Distribution 和 Exponential Distribution

这个部分是上面部分的一个加深理解，首先我们需要总结一下。

1. Global Markov：$X_A\perp X_C|X_B$。也就是 X_A 和 X_B 之间所有的连接都必须在 X_C 中，此时在无向图中满足全局马尔可夫性。

2. Local Markov Network：$x_i\perp x_{-i-nb(i)}|x_{nb(i)}$，其实 nb(i)：neighbor of node i。

3. Pair Markov：$x_i\perp x_j|x_{-i-j}$。

1⇔2⇔3⇔ 因子分解 (基于最大团)。这里面实际上是有个 Hammesley-Clifford 定理的，这个定理的证明非常的困难，我们这里就不做过多的阐述 (实际上我也看的有点懵逼，有需求的小伙伴可以查下 Hammesley-Clifford 定理的证明)。

因子分解：

在我们之前的定义中，

$$p(x)=\frac{1}{Z}\prod_{i=1}^{k}\phi_i(x_{c_i}) \tag{3}$$

c_i：最大团；x_{c_i}：最大团的随机变量集合；$\phi(x_{c_i})$：势函数，必须为正。这里的概念都是来自于统计物理学和热力学的过程。这里的势函数还有可以做文章的地方。

因为，势函数必定为正，我们可以将势函数表达为 $\phi(x_{c_i}) = \exp\{-E(x_{c_i})\}$。其中，$E(x_{c_i})$ 称为 Energy function。实际上用这种形式表达的 $p(x)$，为 Gibbs Distribution，或者又被称之为 Boltzman Distribution。有了 $\phi(x_{c_i})$ 的形式，我们可以进一步推导得：

$$p(x) = \frac{1}{Z}\prod_{i=1}^{K}\phi(x_{c_i}) = \frac{1}{Z}\prod_{i=1}^{K}\exp\{-E(x_{c_i})\} = \frac{1}{Z}\exp\{-\sum_{i=1}^{k}E(x_{c_i})\} \tag{4}$$

我们再来回顾一下指数族分布，指数族分布的通用表达形式为：

$$p(x) = h(x)\exp\{\eta^T\phi(x) - A(\eta)\} = h(x)\frac{1}{Z(\eta)}\exp\{\eta^T\phi(x)\} \tag{5}$$

在这里我们把 $\exp\{-A(\eta)\}$，直接记为 $Z(\eta)$。大家观察就会发现势函数也就是 Gibbs Distribution 就是一个指数族分布。Gibbs 是来自统计物理学，形式上和指数族分布时一样的。而指数族分布实际上是由最大熵分布得到的，那么我们可以理解 Gibbs 分布也是有最大熵原理得到的。而马尔可夫随机场 (Markov Random Field) 实际上等价于 Gibbs 分布。至于为什么？这实际上全部都在 Hammesley-Clifford 定理中，有兴趣的同学，请自行查阅。

Probability Graph 06 Inference Background

Chen Gong

28 November 2019

推断 (Inference) 这个词，对于有一定机器学习基础的同学来说，一定是听说过，这也是贝叶斯方法中一个非常重要的理论性研究。那么什么是推断呢？推断说白了，就是求概率。比如，对于一个联合概率密度函数 $p(x) = p(x_1, x_2, \cdots, x_p)$。我们需要求的有哪些呢?

1. 边缘概率：$p(x_i) = \sum_{x_1} \cdots \sum_{x_{i-1}} \cdots \sum_{x_{i+1}} \cdots \sum_{x_p} p(x)$。

2. 条件概率：$p(x_A|x_B)$，令 $x = x_A \cup x_B$。

3. MAP Inference：也就是 $\hat{z} = \arg\max_z \ p(z|x) \propto \arg\max \ p(x, z)$。因为 $p(z|x) = \frac{p(x,z)}{p(x)} \propto p(x, z)$，我们的目标是求一个最优的参数 z，并不需要知道具体的数值是多少，只要知道谁大谁小就行，所以 $p(x)$ 可以直接不看了。

现在我们知道了，Inference 在求什么？下一步，我们要总结 Inference 有哪些方法。

1 Inference 求解方法

1.1 精准推断 (Deterministic Inference)

Variable Elimination (VE)，变量消除法；Belief Propagation (BP) 信念传播，这个可不是我们之前学习的反向传播算法，这里需要注意。同时这个算法衍生出的 Sum Product Algorithm，这就是推断的核心，这是一种树结构的；而 Junction Tree Algorithm，这是一种普通图结构。

1.2 近似推断 (Approximate Inference)

典型的有有向环 (Loop Belief Propagation)；采样方法，包括 Mente Carlo Inference：Importance Sampling，MCMC；最后一个就是我现在主要研究的变分推断 (Variational Inference)。

2 隐马尔可夫模型 (Hidden Markov Model)

Hidden Markov Model (HMM) 算法将在后面的章节中做详细的描述，在这一小节中，我们主要

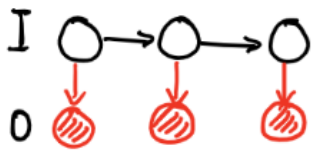

图 1: HMM 的模型结构

做一点概述性的描述。HMM 的模型可视化为上图所示，其中 O 是隐变量，也就是我们的观测变量。

我们主要考虑三个问题，在三个问题为 Inference 的问题。1. Evaluation，也就是求一个边缘密度 $p(O) = \sum_I P(I, O)$。2. Learning，也就是寻找 $\hat{\lambda}$。3. Decoding：$\hat{I} = \arg\max_I P(I|O)$，包括 Vitebi Algorithm，这是一个动态规划算法。而隐马尔可夫模型实际上一种动态规划模型 (Dynamic Bayesian Network)。

Probability Graph 07 Variable Elimination

Chen Gong

07 December 2019

在上一小节中，我们简单的介绍了推断的背景和分类，我们知道了大致有哪些推断的方法。推断的任务可以被我们介绍为：给定已知的 $p(x) = (x_1, x_2, \cdots, x_p)$，我们需要求的有三个：

1. 边缘概率：$p(x_i) = \sum_{x_1,\cdots,x_{i-1},x_{i+1},\cdots,x_p} p(x_1, x_2, \cdots, x_p)$。
2. 条件概率：$p(x_A|x_B)$，也就是在已知 x_B 集合的情况下，如何求得 x_A 集合的概率。
3. 最大后验概率 (MAP)：$\hat{x}_A = \arg\max_{x_A} p(x_A|x_B) = \arg\max_{x_A}\ p(x_A, x_B)$。

下面我们要介绍最简单的一个精确推断中中的东西，名为变量消除法 (Variable Elimination)。这是一种最简单的推断方法，也是我们学习推断法的核心概念之一。下面我们做详细的解释。

1 变量消除法 (Variable Elimination Algorithm)

假如我们有一个马氏链：

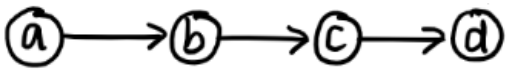

图 1: 一个马氏链的抽象模型

那么我们怎么来求 $p(d)$ 呢？根据公式我们可以得到：

$$p(d) = \sum_{a,b,c} p(a, b, c, d) \tag{1}$$

然后使用因子分解，我们可以得到：

$$p(d) = \sum_{a,b,c} p(a)p(b|a)p(c|b)p(d|c) \tag{2}$$

假定，a, b, c, d 都为均匀离散的二值 random variable，所以 $a, b, c, d \in \{0, 1\}$。
所以，

$$\begin{aligned} p(d) =& p(a=0)\cdot p(b=0|a=0)\cdot p(c=0|b=0)\cdot p(d|c=0) \\ &+p(a=1)\cdot p(b=0|a=1)\cdot p(c=0|b=0)\cdot p(d|c=0) \\ &+\cdots \\ &+p(a=1)\cdot p(b=1|a=1)\cdot p(c=1|b=1)\cdot p(d|c=1) \end{aligned} \tag{3}$$

实际上，这里有 8 个因子的积。那么我们来做进一步的分析，我们可以令

$$\begin{aligned} p(d) &= \sum_{a,b,c} p(a)p(b|a)p(c|b)p(d|c) \\ &= \sum_{b,c} p(c|b)p(d|c) \sum_{a} p(a)p(b|a) \end{aligned} \tag{4}$$

而 $p(a)p(b|a) = p(a,b)$，而 $\sum_a p(a)p(b|a) = p(b)$。我们可以将 a 看成 $\phi(a)$ 这是一个和 a 相关的函数，同理 $p(b|a)$ 看成 $\phi(a,b)$。所以，我们可以将 $\sum_a p(a)p(b|a)$ 看成 $\phi_a(b)$，这样就相当于一个关于 b 的函数，并且是从 a 中导出的。所以，我们做如下替换可得：

$$\sum_{b,c} p(c|b)p(d|c) \sum_{a} p(a)p(b|a) = \sum_{b,c} p(c|b)p(d|c)\phi_a(b) = \sum_{c} p(d|c) \sum_{b} p(c|b)\phi_a(b) \tag{5}$$

同理，我们将 $\sum_b p(c|b)\phi_a(b)$ 看成 $\phi_b(c)$。所以，原始将被改写为：

$$\sum_{c} p(d|c)\phi_b(c) = \phi_c(d) \tag{6}$$

这个算法的核心就是乘法对加法的分配律。那我们怎么类比到乘法的分配律呢？首先先来简单的回顾一下乘法的分配律，也就是 $ac + ab = a(b + c)$。那么我们仔细的来看看这个计算 $p(d)$ 的过程。这是不是就是一个不断的提取公因子，进行计算的过程？有没有觉得和分配律很像？先提取 a 的部分，计算 a 的部分，然后再依次的提取 b 的部分，c 的部分，最后剩下的就是 d 的部分。那么，我们就可以把这么一长串的公式进行逐步化简了，这就是变量消元的思想。同样，在无向图中，我们也可以使用到马尔可夫网络中。

$$p(a,b,c,d) = \frac{1}{z} \prod_{i=1}^{k} \phi_{c_i}(x_{c_i}) \tag{7}$$

写成因子分解的形式就是 $p(x) = \prod_{x_i} \phi_i(x_i)$。这实际上就是分配律，一个变量一个变量的提取，然后进行分解计算。同时这种算法的缺点也非常的明显。

首先，就是重复计算的问题，无论计算那个变量的概率都要重复的计算一遍所有的概率。这个原因就会导致算法的计算难度非常的大。第二个就是计算次序的问题，我们举的例子还比较的简单，所以我们可以一眼就看出来，按 $a - b - c - d$ 的次序开始算。但是，实际上，并没有这么容易就得到计算的次序，而且计算次序不一样会导致计算的难度有很大的区别。而有数学家已经证明了，确定最优的计算顺序，本身就是一个 NP hard 的问题，非常难求解。

Probability Graph 08 Belief Propagation

Chen Gong

08 December 2019

在上一小节中，我们已经介绍了变量消除 (Variable Elimination)，Variable Elimination 的思想是 Probability Graph 中的核心思想之一。上一节中我们就已经介绍了，这实际上就是乘法对加法的分配律。但是，Variable Elimination 中有很多的问题，比如重复计算和最优计算次序不好确定的问题。所以，我们这一节来介绍 Belief Propagation 来解决重复计算的问题。

1 Forward and Backward Algorithm

假设，我们现在有一个马氏链模型：

图 1: 链式马氏链模型结构图

联合概率可以被我们表示为：$p(a,b,c,d,e) = p(a) \cdot p(b|a) \cdot p(c|b) \cdot p(d|c) \cdot p(e|d)$。

如果，我们想要求的是 $p(e)$，那么：

$$\begin{aligned} p(e) &= \sum_{a,b,c,d} p(a,b,c,d,e) \\ &= \sum_{a} p(e|d) \cdot \sum_{c} p(d|c) \cdot \sum_{b} p(c|b) \cdot \sum_{a} p(b|a) \cdot p(a) \end{aligned} \tag{1}$$

为了简化表达，这里我们需要定义一个很重要的符号。因为，$\sum_a p(b|a) \cdot p(a)$，是一个关于 b 的表达式，也就是相当于把 a 给约掉了。所以我们可以把 $\sum_a p(b|a) \cdot p(a)$ 记为 $m_{a \longrightarrow b}(b)$。同理，我们也可以将 $\sum_c p(d|c) \cdot \sum_b p(c|b) \cdot m_{a \longrightarrow b}(b)$ 记为 $m_{b \longrightarrow c}(c)$。那么为了求得 $p(e)$，我们依次的求解顺序为 $m_{a \longrightarrow b}(b)$，$m_{b \longrightarrow c}(c)$，$m_{c \longrightarrow d}(d)$ 和 $m_{d \longrightarrow e}(e)$。也就相当于沿着这个链这个马氏链一直往前走，也就是前向算法 (Forward Algorithm)。我们用公式表达即为：

$$a \overset{m_{a \longrightarrow b}(b)}{\longrightarrow} b \overset{m_{b \longrightarrow c}(c)}{\longrightarrow} c \overset{m_{c \longrightarrow d}(d)}{\longrightarrow} d \overset{m_{d \longrightarrow e}(e)}{\longrightarrow} e \tag{2}$$

如果是要求 $p(c)$，那么我们的传递过程为 $a \longrightarrow b \longrightarrow c \longleftarrow d \longleftarrow e$。这里，我们就不能只用前向算法来解决了，需要用到 Forward-Backward 算法来解决了。也就是同时使用 Forward 和 Backward 的方法，那么我们来看看 $p(c)$ 怎么求?

$$\begin{aligned} p(c) &= \sum_{a,b,d,e} p(a,b,c,d,e) \\ &= (\sum_{b} p(c|b) \sum_{a} p(b|a)p(a)) \cdot (\sum_{d} p(d|c) \sum_{e} p(e|d)) \end{aligned} \tag{3}$$

对比上面的计算 $p(e)$ 的过程，我们就可以发现，$\sum_b p(c|b)\sum_a p(b|a)p(a)$ 部分的计算也就是 $m_{b\longrightarrow c}(c)$ 的计算是一模一样的。所以说，Variable Elimination 里面有大量的重复计算。Belief 的想法很简单，也就是将 $m_{i\longrightarrow j}(j)$，全部事先计算好，就像一个个积木一样，然后再用这个积木来搭建运算。那么也就是，我们事先将方向全部定义好，正向和反向的全部都求了再说。为了进一步探究 Belief Background，我们需要讨论一些更加 Generalize 的情况。也就是从 Chain$\longrightarrow$Tree，有向 $\longrightarrow$ 无项的情况。

2 Belief Propagation 的扩展

我们的 Generalize 的后，分析了一个树形的无向图结构。图的网络结构如下所示：

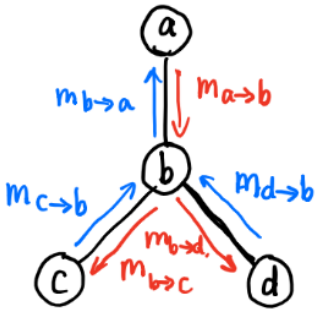

图 2: 树形无向图结构拓扑结构

下面第一步，我们把上面那个模型的设置写出来。所以，我们需要进行因式分解，我们用 $\varphi(i)$ 来表示和 i 有关的部分。所以，我们可以将联合概率密度写为：

$$p(a,b,c,d)=\frac{1}{Z}\varphi_a(a)\varphi_b(b)\varphi_c(c)\varphi_d(d)\cdot\varphi_{a,b}(a,b)\varphi_{b,c}(b,c)\varphi_{b,d}(b,d) \tag{4}$$

我们要求 $p(a)=\sum_{b,c,d}p(a,b,c,d)$ 和 $p(b)=\sum_{a,c,d}p(a,b,c,d)$，其间一定会出现大量的重复计算。这个模型中有四个点，三条边，每条边都有两个方向，所以我们要求的是 6 个 “积木”。我们来一步步的看看，如何可以得到想要的 $p(a)$。

1. 首先，我们需要求的是 $c\longrightarrow b$ 和 $d\longrightarrow b$ 两个过程。其中，$c\longrightarrow b$ 的过程也就是 $m_{c\longrightarrow b}(b)$，可以被我们表达为 $\sum_c\varphi_c\cdot\varphi_{b,c}$。同理，$m_{d\longrightarrow b}(b)$ 可以被我们表达为 $\sum_d\varphi_b\cdot\varphi_{b,d}$。

2. 第二步，我们需要求 $b\longrightarrow a$ 的过程，也就是 $m_{b\longrightarrow a}(a)$。它等于 $m_{c\longrightarrow b}(b),m_{d\longrightarrow b}(b)$ 乘上 b 自己的部分求和得到，我们可以写为：

$$m_{b\longrightarrow a}(a)=\sum_b m_{c\longrightarrow b}(b)\cdot\varphi_b\cdot\varphi_{a,b}\cdot m_{d\longrightarrow b}(b) \tag{5}$$

3. 最后，$m_{b\longrightarrow a}(a)$ 乘上 a 自己的部分就得到了 $p(a)$，也就是：$p(a)=\varphi_a\cdot m_{b\longrightarrow a}(a)$。

所以，我们总结一下：

$$m_{b\longrightarrow a}(x_a)=\sum_{x_b}\varphi_{a,b}\varphi_b m_{c\longrightarrow b}(x_b)m_{d\longrightarrow b}(x_b) \tag{6}$$

而，

$$p(a)=\varphi_a m_{b\longrightarrow a}(x_a) \tag{7}$$

我相信到这里，大家应该是可以理解这个意思的，会有点抽象，但并不是很难。下一步我想做个 Generalize 为了便于大家进行理解，我这里尽量不跳步：

$$m_{b\longrightarrow a}(x_a) = \sum_{x_b} \varphi_{a,b}\varphi_b \prod_{\{k\in NB(b)\}\longrightarrow a} m_{k\longrightarrow b}(x_b) \tag{8}$$

这里的 $NB(b)$ 代表的是所有节点 b 的邻接节点。我们可以进一步表示为：

$$m_{j\longrightarrow i}(x_i) = \sum_{x_j} \varphi_{i,j}\varphi_j \prod_{\{k\in NB(j)\}\longrightarrow i} m_{k\longrightarrow j}(x_j) \tag{9}$$

而，

$$p(x_i) = \varphi_i \prod_{k\in NB(i)} m_{k\Longrightarrow i}(x_i) \tag{10}$$

通过对上面表达式的观察，我们是不是发现了一个很有意思的现象。也就是这些概率都是由 $m_{i\longrightarrow j}$ 这样的小积木拼接起来的。所以我们可以 get a conclusion：我们不要一上来就直接去求边缘概率密度，比如 $p(a), p(b), p(c), p(d)$ 这些的。我们可以先建立一个 Cache，把 $m_{i\longrightarrow j}$ 全部算出来。然后，要求什么的话，直接进行搭建和拼接就可以了。从这里，我们就引出了 Belief Propagation。

3 Belief Propagation

我们之前就已经得到了信息传递的表达式：

$$m_{b\longrightarrow a} = \sum_{b} \varphi_{a,b}\varphi_b \cdot m_{c\longrightarrow b} \cdot m_{d\longrightarrow b} \tag{11}$$

在开始理解 Belief Propagation 之前，我们在分析一下 $m_{b\longrightarrow a}$ 的公式中，每个部分表达的具体含义。其中，$m_{c\longrightarrow b} \cdot m_{d\longrightarrow b}$ 中反映的是孩子 (Children) 的信息。而 φ_b 中表示是 b 节点自己的信息。那么，$\varphi_b \cdot m_{c\longrightarrow b} \cdot m_{d\longrightarrow b}$ 可以被我们看成是 b 节点的所有信息，包括节点自己本身的信息和其他节点传播来的信息，所以我们将这个部分记为：Belief。所以，Belief$(b) = \varphi_b \cdot$ children。b 节点收集孩子和自己的信息，整合和 b 相关的所有信息，通过 Belief$(b) = \varphi_b \cdot$ children 向 a 传去。

那么，Belief Propagation 可以看成是 BP = VE + Cashing。这个算法的核心思想就在于，直接求 $m_{i\longrightarrow j}$，然后再导出边缘概率 $p(x_i)$。第二小节中我们已经详细的给出了 $p(x_i)$ 的推导方法。下一步，我们需要知道如何来求 $m_{i\longrightarrow j}$。

3.1 Sequential Implementation

顺序计算的思路，我们需要借助一个队列来实现：

1. Get Root: 首先我们需要假设一个节点为根节点。

2. Collect Message: 对于每一个在根节点的邻接点中的节点 x_i，Collect Message (x_i)。对应的就是图 2 中蓝色的线条。

3. Distribute Message: 对于每一个在根节点的邻接点中的节点 x_i，Distribute Message (x_i)。对应的就是图 2 中红色的线条。

经过这三个步骤以后，我们可以得到所有的 $i, j \in v$，从而计算出 $p(x_k), k \in v$。

3.2 Parallel Implementation

这种思想在图结构的网络中经常使用，大致也就是随意选一个节点，然后向四周其他的节点辐射信息。其他节点收到信息之后，更新自己的状态，然后反馈自己的信息给源节点，源节点再更新自己的信息。不断地重复这个工作，直到最后收敛为止。

4 小结

实际上，Belief Propagation 就是一种 Variable Elimination。但是，我们发现了 Variable Elimination 中有很多的重复计算。所以，我们想到了提取出来先算好，要用的时候直接放进去就行了。所以，Belief Propagation 中分解了传递的过程，先计算消息传递的机制，再来组装出计算边缘概率。其实本质还是 Variable Elimination 算法，不过就是使表达更加的规范了，通过拆解的方法来消除重复计算。

Probability Graph 09 Max Product Algorithm

Chen Gong

10 December 2019

我们在这里再总结一下概率图模型有什么用。对于一个图，Graph $= \{X, E\}$，其中 X 代表的是普通变量，E 代表的是 Evidence，也就是观测变量。

1. 首先要解决的是边缘变量的问题，也就是已知：$E = \{e_1, e_2, \cdots, e_k\}$，如何求 $p(E)$ 的问题，其中 E 为一个变量或者为一个子集。实际上就是一个 likelihood 的问题。

2. 条件概率，也就是一个求后验概率的问题，目标概率为 $X = (Y, Z)$。而 $p(Y|E) = \sum_z p(X|E)$。

3. 最大后验估计 (MAP)，也被我们称为 Decoding 的问题。也就是我们希望找到一个隐序列，使得：$\hat{x} = argmax_x p(X|E)$，$\hat{y} = argmax_y p(Y|E)$。

这里的 Max-Product 算法和隐马尔可夫模型 (HMM) 中的 Viterbi 算法非常的类似。其实，从算法上讲它就是 Belief Propagation 算法的一种改进，从模型上讲是 Viterbi 算法的推广。在这里我们求的不是概率值了，而是一个最优的概率序列 $(\hat{a}, \hat{b}, \hat{c}, \hat{d}) = argmax_{a,b,c,d} p(x_a, x_b, x_c, x_d|E)$。

1 Max Product Algorithm

下面展示一个树的拓扑结构图。

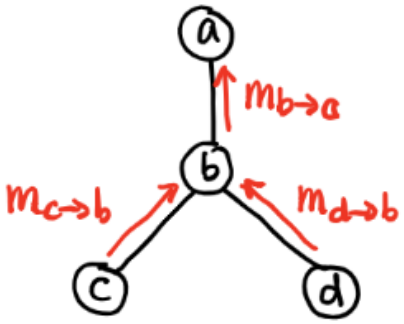

图 1: 概率树模型的拓扑结构图

在这个树中，我们将 $m_{b\longrightarrow a}$ 看成是能使 $p(x_b, x_c, x_d|E)$ 联合概率达到最大的值。每一个节点代表的是到这个节点为止的路径联合概率达到最大的值。我们表达为：

$$m_{j\longrightarrow i} = \max_{x_j} \varphi_i \varphi_{ij} \prod_{k\in\{NB(j)-i\}} m_{k\longrightarrow j} \tag{1}$$

那么，在图一所示的概率图模型中，$m_{c\longrightarrow b}$ 可以表示为：

$$m_{c\longrightarrow b} = \max_{x_c} \varphi_c \cdot \varphi_{bc} \tag{2}$$

其中，$\varphi_c \cdot \varphi_{bc}$ 可以表示为和 c 相关的函数。

$$m_{d \longrightarrow b} = \max_{x_d} \varphi_c \cdot \varphi_{cd} \tag{3}$$

其中，$\varphi_d \cdot \varphi_{cd}$ 可以表示为和 d 相关的函数。

$$m_{b \longrightarrow a} = \max_{x_b} \varphi_b \cdot \varphi_{ab} \cdot m_{c \longrightarrow b} \cdot m_{d \longrightarrow b} \tag{4}$$

最终，我们将得到的是：

$$\max p(a, b, c, d) = \max_{x_a} \varphi_a m_{b \longrightarrow a} \tag{5}$$

而 $\varphi_a m_{b \longrightarrow a}$ 就可以看成是一个关于 a 的函数。这里我们再提一下 Belief Propagation，这里的 Max-Product 实际上就是 Belief Propagation 的一个变形。

2 Belief Propagation

实际上这个算法的提出时因为，多次求边缘概率密度会发现中间有很多的步骤是重复的。我们用 $m_{i \longrightarrow j}$ 记录每一个边缘概率，最后进行组合就行。所以，

$$m_{j \longrightarrow i}(x_i) = \sum_{x_j} \varphi_{i,j}(x_i, x_j) \varphi_j(x_j) \prod_{\{k \in NB(j)\} \longrightarrow i} m_{k \longrightarrow j}(x_j) \tag{6}$$

而：

$$p(x_i) = \varphi(x_i) \prod_{k \in NB(x_i)} m_{k \longrightarrow i}(x_i) \tag{7}$$

3 Compare

其实，我们一比较就可以很简单的看出，Max-product 和 Belief Propagation 只有一个地方不一样。那就是前者是求最大，后者是求和。也就是 Max-product 到 Sum-product。在求得了 $\max p(a, b, c, d) = \max_{x_a} \varphi_a m_{b \longrightarrow a}$ 之后，我们利用回溯法我们比较就可以比较简单的得到 $x_a^*, x_b^*, x_c^*, x_d^*$ 了。在这个算法中，我们就不需要事先计算 $m_{i \longrightarrow j}$ 了，直接在迭代中进行计算就可以了，也不会存在什么重复计算的问题。

Probability Graph 10 Moral Graph & Factor Graph

Chen Gong

11 December 2019

在这一小节中，我们将要介绍两种特殊的概率结构，也就是 Moral Graph 和 Factor Graph。

1 Moral Graph

首先我们需要知道，为什么要有 Moral Graph 的存在？Moral Graph 存在的意义就是将有向图转化为无向图来研究。因为无向图比有向图更加的 Generalize 一些。在概率图中，我们可以分为贝叶斯网络 (有向图) 和马尔可夫网络 (无向图)。

无向图可以表示为：

$$p(x) = \frac{1}{z}\prod_{i=1}^{k}\phi_{c_i}(x_{c_i}) \tag{1}$$

有向图可以表示为：

$$p(x) = \prod_{i=1}^{p} p(x_i|x_{pa(i)}) \tag{2}$$

其中，ϕ_{c_i} 代表的是最大团的意思。通过道德图，我们可以有效的将有向图转换为无向图。

我们看一下如图所示的链式网络：

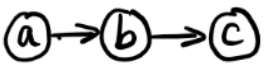

图 1: 链式有向图模型

其中，$p(a,b,c) = p(a)p(b|a)p(c|b)$。如果，把有向图转换成有向图是一件非常简单的事情，首先把所有的线条换成直线。由于在无向图中，我们考虑的是最大团，所以 $p(a)p(b|a) = \varphi(a,b)$，$p(c|b) = \varphi(b,c)$。这个的转换是非常的简单了。

第二种，我们需要讨论的图也就是 Tail to Tail 的图，所下图所示：

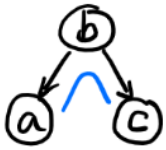

图 2: Tail to Tail 有向图模型

其中，$p(a,b,c) = p(a)p(b|a)p(c|a)$。还是按照一样的套路，首先把所有的有向箭头改成直线。那么我们就可以得到 $p(a)p(b|a) = \phi(a,b)$，$p(c|a) = \phi(a,c)$。其中 $\{a,c\}$ 和 $\{b,c\}$ 是分别属于两个团。这个也比较的简单，但是 Head to Head 的转换就有点不一样了。

第三种，我们需要讨论的图是 Head to Head 的模型，如下图所示：

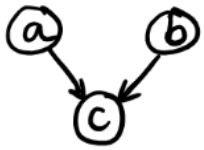

图 3: Head to Head 有向图模型

同样我们使用一样的分析思路来看这个问题，$p(a,b,c) = p(a)p(b)p(c|a,b)$。我们进行拆解的话，只能令 $p(a)p(b)p(c|a,b) = \varphi(a,b,c)$，不然再也找不到其他的拆解方法。但是，如果简单的将模型中所有的有向箭头改成直线得到的并不是一个团。因为 “团” 的概念的要求，团里面的元素都要求是两两相互连接的。所以，我们需要进行改进，将 Head to Head 的无向图形式改进为：

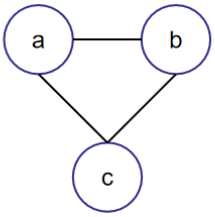

图 4: Head to Head 无向图模型

那么，将 Head to Head 的有向图转换为无向图的过程可以被我们描述为：

对于 $\forall x_i \in G$，将 $parent(x_i)$ 的两个父亲节点连接，然后将 G 中所有的有向边替换成无向边。下面我们举一个例子：

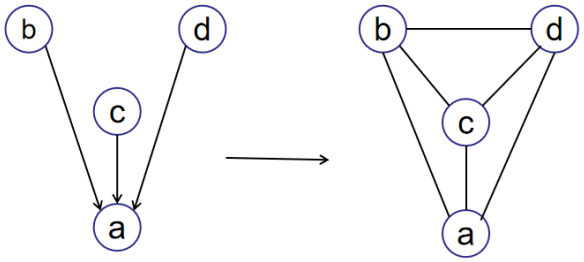

图 5: Head to Head 有向图模型转无向图模型举例

而我们将有向图转换成无向图之后，有什么好处吗？也就是在判断条件独立性的时候，有时图形非常复杂的时候。我们在有向图中很难看出来，而在无向图中却可以很简单的得到我们想要的结果。也就是 $Sep(A,B|C) \Longleftrightarrow D-Sep(A,B|C)$。

2 Factor Graph

在上一小节中，我们介绍了道德图 (Moral Graph)，它的主要作用是将有向图转换为无向图。我们考虑的都是树结构，但是在 Head to Head 结构中，会引入环的结构。但是，在我们的 Belief Propagation (BP) 算法中，只能对树进行分解。所以，这里我们就引入了因子图。因子图主要发挥两个作用：1. 去环，也就是消除无向图中的环结构；2. 使算法变得更加的简洁，简化计算。

如图二表达的那样，他的有向图和无向图的联合概率可以分别表达为：

$$p(a,b,c) = p(a)p(b|a)p(c|a) \qquad p(a,b,c) = \frac{1}{Z}\phi(a,b)\phi(a,c) \tag{3}$$

那什么是因子图分解呢？公式表达可以被我们表示为：

$$p(x) = \prod_S p(x_S) \tag{4}$$

其中，S 是图的节点子集，X_S 为对应的 X 的子集，也就是 X 的随机变量的子集。那么对于一个如图 4 所示的有环无向图，我们怎么进行因子图分解呢？

首先进行第一种分解，如下图所示：

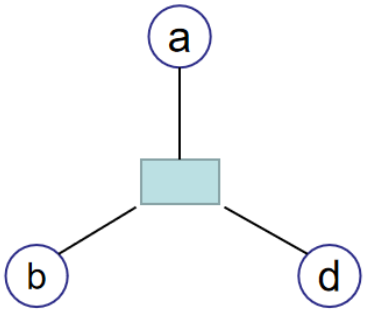

图 6: Head to Head 无向图中心节点因子图分解

这时可以被我们描述为，$f = f(a,b,c)$。或者我们也可以进行更细的分解。如下图所示：

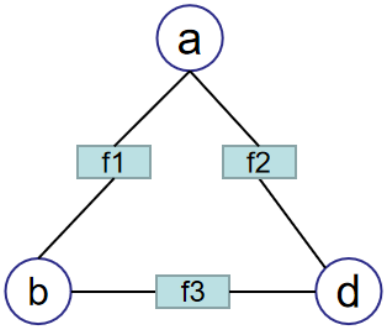

图 7: Head to Head 无向图三节点因子图分解

这个分解的结果可以被我们表示为：

$$p(x) = f_1(a,b)f_2(a,c)f_3(b,c) \tag{5}$$

不仅是可以在两个节点之间插入关系，同时也可以对于单个节点引入函数。

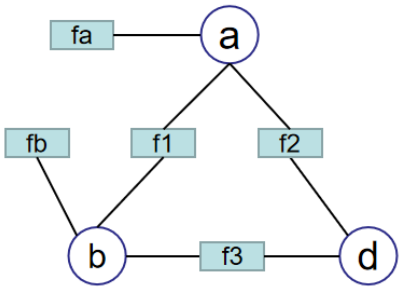

图 8: Head to Head 无向图三节点和独立节点因子图分解

那么这个分解结果可以被我们表示为：

$$p(x) = f_1(a,b)f_2(a,c)f_3(b,c)f_a(a)f_b(b) \tag{6}$$

实际上，就可以看成是对因式分解的进一步分解。这样我们就可以成功的消除环结构。如下图所示：

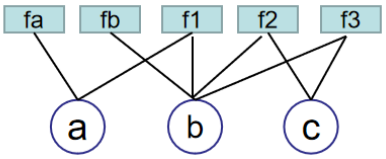

图 9: Head to Head 无向图三节点和独立节点因子图分解

所以，大家仔细一想就知道了因子图存在的意义了，它可以有效的消除环结构，通过一个重构的方式，重建出树的结构。这样可以有效的帮助我们使用 Belief Propagation 中的变量消除法等方法。

Expectation Maximization 01 Algorithm Convergence

Chen Gong

17 December 2019

Expectation Maximization (EM) 算法，是用来解决具有隐变量的模型的概率计算问题。在比较简单的情况中，我们可以直接得出我们想要求得的参数的解析解，比如：MLE: $p(X|\theta)$。我们想要求解的结果就是：

$$\theta_{MLE} = \arg\max_{\theta} \log p(X|\theta) \tag{1}$$

然而一旦问题变得复杂起来以后，就不是这么简单了，特别是引入了隐变量之后。

1 EM 算法简述

实际上，EM 算法的描述也并不是很难，我们知道，通常我们想求的似然函数为 $p(X|\theta)$。引入隐变量之后，原式就变成了：

$$p(X|\theta) = \int \log p(X, Z|\theta) p(Z|X, \theta^{(t)}) dZ \tag{2}$$

EM 算法是一种迭代的算法，我们的目标是求：

$$\begin{aligned} \theta^{(t+1)} &= \arg\max_{\theta} \int_Z \log p(X, Z|\theta) p(Z|X, \theta^{(t)}) dZ \\ &= \arg\max_{\theta} \mathbb{E}_{Z\sim p(Z|X,\theta^{(t)})}[\log p(X, Z|\theta)] \end{aligned} \tag{3}$$

也就是找到一个更新的参数 θ，使得 $\log p(X, Z|\theta)$ 出现的概率更大。

2 EM 算法的收敛性

我们想要证的是当 $\theta^{(t)} \longrightarrow \theta^{(t+1)}$ 时，有 $\log p(X|\theta^{(t)}) \leq \log p(X|\theta^{(t+1)})$。这样才能说明我们的每次迭代都是有效的。

$$\log p(X|\theta) = \log \frac{p(X, Z|\theta)}{p(Z|X, \theta)} = \log p(X, Z|\theta) - \log p(Z|X, \theta) \tag{4}$$

下一步，则是同时对两边求关于 $p(Z|X, \theta^{(t)})$ 的期望。

左边：

$$\begin{aligned} \mathbb{E}_{Z\sim p(Z|X,\theta^{(t)})}[\log p(X|\theta)] &= \int_Z p(Z|X, \theta^{(t)} \log p(X|\theta) dZ \\ &= \log p(X|\theta) \int_Z p(Z|X, \theta^{(t)}) dZ \\ &= \log p(X|\theta) \cdot 1 = \log p(X|\theta) \end{aligned} \tag{5}$$

右边：

$$\underbrace{\int_Z p(Z|X,\theta^{(t)})\log p(X,Z|\theta)dZ}_{Q(\theta,\theta^{(t)})}-\underbrace{\int_Z p(Z|X,\theta^{(t)})\log p(Z|X,\theta)dZ}_{H(\theta,\theta^{(t)})} \tag{6}$$

大家很容易就观察到，$Q(\theta,\theta^{(t)})$ 就是我们要求的 $\theta^{(t+1)}=\arg\max_\theta \int_Z p(X,Z|\theta)p(Z|X,\theta^{(t)})dZ$。那么，根据定义，我们可以很显然的得到：$Q(\theta^{(t+1)},\theta^{(t)})\geq Q(\theta,\theta^{(t)})$。当 $\theta=\theta^{(t)}$ 时，等式也是显然成立的，那么我们可以得到：

$$Q(\theta^{(t+1)},\theta^{(t)})\geq Q(\theta^{(t)},\theta^{(t)}) \tag{7}$$

这时，大家想一想，我们已经得到了 $Q(\theta^{(t+1)},\theta^{(t)})\geq Q(\theta^{(t)},\theta^{(t)})$ 了。如果，$H(\theta^{(t+1)},\theta^{(t)})\leq H(\theta^{(t)},\theta^{(t)})$。我们就可以很显然的得出，$\log p(X|\theta^{(t)})\leq \log p(X|\theta^{(t+1)})$ 了。

证明：

$$\begin{aligned}
H(\theta^{(t+1)},\theta^{(t)})-H(\theta^{(t)},\theta^{(t)}) &= \int_Z p(Z|X,\theta^{(t)})\log p(Z|X,\theta^{(t+1)})dZ-\int_Z p(Z|X,\theta^{(t)})\log p(Z|X,\theta^{(t)})dZ\\
&= \int_Z p(Z|X,\theta^{(t)})\log\frac{p(Z|X,\theta^{(t+1)})}{p(Z|X,\theta^{(t)})}dZ\\
&= -KL(p(Z|X,\theta^{(t)})||p(Z|X,\theta^{(t+1)}))\leq 0
\end{aligned} \tag{8}$$

或者，我们也可以使用 Jensen inequality。很显然，log 函数是一个 concave 函数，那么有 $\mathbb{E}[\log X]\leq \log[\mathbb{E}[X]]$，那么：

$$\begin{aligned}
\int_Z p(Z|X,\theta^{(t)})\log\frac{p(Z|X,\theta^{(t+1)})}{p(Z|X,\theta^{(t)})}dZ &= \mathbb{E}_{Z\sim p(Z|X,\theta^{(t)})}\left[\log\frac{p(Z|X,\theta^{(t+1)})}{p(Z|X,\theta^{(t)})}\right]\\
&\leq \log\left[\mathbb{E}_{Z\sim p(Z|X,\theta^{(t)})}\left[\frac{p(Z|X,\theta^{(t+1)})}{p(Z|X,\theta^{(t)})}\right]\right]\\
&= \log\left[\int_Z p(Z|X,\theta^{(t)})\left[\frac{p(Z|X,\theta^{(t+1)})}{p(Z|X,\theta^{(t)})}\right]dZ\right]\\
&= \log\int_Z p(Z|X,\theta^{(t+1)})dZ\\
&= 0
\end{aligned} \tag{9}$$

所以，从两个方面我们都证明了，$\log p(X|\theta^{(t)})\leq \log p(X|\theta^{(t+1)})$。那么，经过每次的迭代，似然函数在不断的增大。这就证明了我们的更新是有效的，也证明了算法是收敛的。

Expectation Maximization 02 Derived Formula

Chen Gong

18 December 2019

机器学习中，所谓的模型实际上就可以看成是一堆的参数。根据极大似然估计的思想，我们要求解的对象的是：

$$\theta_{MLE} = \log P(X|\theta) \tag{1}$$

其中，X 为 observed data；Z 为 latent data；(X, Z) 为 complete data；θ 为 parameter。

那么，EM 公式就被我们描述为：

$$\theta^{(t+1)} = \arg\max_{\theta} \int_{Z} \log P(X, Z|\theta) P(Z|X, \theta^{(t)}) dZ \tag{2}$$

EM 算法可以被我们分解成 E-step 和 M-step 两个部分。

E-step:

$$P(Z|X, \theta^{(t)}) \longrightarrow \mathbb{E}_{Z\sim P(Z|X,\theta^{(t)})} \left[\log P(X, Z|\theta)\right] \tag{3}$$

M-step:

$$\theta^{(t+1)} = \arg\max_{\theta} \mathbb{E}_{Z\sim P(Z|X,\theta^{(t)})} \left[\log P(X, Z|\theta)\right] \tag{4}$$

前面我们已经证明了 EM 算法的收敛性了，也就是：

$$\log P(X|\theta^{(t+1)}) \geq \log P(X|\theta^{(t)}) \tag{5}$$

收敛性告诉了我们算法确实是有效的，我们可以放心的去使用它。而大家会不会觉得这个公式的得来有点懵逼？懵逼就对了，那么下一步，我们的目标就是要推导出 EM 算法究竟是怎么来的，给出一个理论的证明。

1 从 KL Divergence 进行分析

这是个什么东西呢？中文名字叫做 “证据下界”。这个名字读起来似乎有一点点奇怪。我们首先看看它是怎么来的。首先，我们定义一个有关于表示层 Z 的表示层变量 $q(Z)$，$q(Z)$ 可以表示任何一个变量。

$$\begin{aligned} \log P(X|\theta) &= \log P(X, Z|\theta) - \log P(Z|X, \theta) \\ &= \log \frac{P(X, Z|\theta)}{Q(Z)} - \log \frac{P(Z|X, \theta)}{Q(Z)} \end{aligned} \tag{6}$$

两边同时对于 $Q(Z)$ 求期望，我们可以得到：

左边：

$$\begin{aligned}\int_Z Q(Z)\log P(X|\theta)dZ &= \log P(X|\theta)\int_Z Q(Z)dZ \\ &= \log P(X|\theta)\cdot 1 \\ &= \log P(X|\theta)\end{aligned} \tag{7}$$

右边：

$$\underbrace{\int_Z Q(Z)\log\frac{P(X,Z|\theta)}{Q(Z)}dZ}_{ELBO} - \underbrace{\int_Z Q(Z)\log\frac{P(Z|X,\theta)}{Q(Z)}dZ}_{KL} \tag{8}$$

所以，实际上，$\log P(X|\theta) = ELBO + KL(Q||P)$。其中，$P(Z|X,\theta)$ 为后验分布 (Posterior)。并且，KL 散度的值一定是大于零的。所以，$\log P(X|\theta) \geq ELBO$，当且仅当 $P(Z|X,\theta) = Q(Z)$ 时等号成立。

EM 算法的一个想法就是想让 ELBO 不断的增加，从而使 $\log P(X|\theta)$ 不断的变大的一种攀爬的迭代方法。

那么，我们对下界进行优化，使下界尽可能的变大，就可以使目标函数不断的上升，那么我们可以得到：

$$\hat{\theta} = \arg\max_\theta ELBO = \arg\min_\theta -\int Q(Z)\log\frac{P(X,Z|\theta)}{Q(Z)}dZ \tag{9}$$

而这里的 $Q(Z)$ 的分布我们怎么得到呢？这里我们就要来讲一讲 EM 算法的一个核心的理解了。首先我们给出这个理解的图示结果，再对这个图来进行讲解：

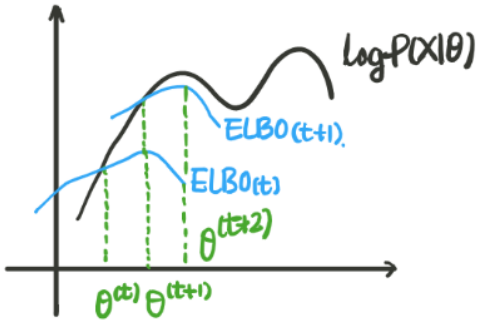

图 1: EM 算法迭代流程图

由于我们的目标是最大化 ELBO。这个下界我们怎么优化？因为我们需要优化的是 ELBO 的参数 θ。那么，对于某一个时刻的 $\theta^{(t)}$，我们可以的得到一个关于 θ 的函数：

$$\log P(X|\theta^{(t)}) = \int_Z Q(Z)\log\frac{P(X,Z|\theta)}{Q(Z)}dZ - \int_Z Q(Z)\log\frac{P(Z|X,\theta^{(t)})}{Q(Z)}dZ \tag{10}$$

由于想让 $\int_Z Q(Z)\log\frac{P(X,Z|\theta)}{Q(Z)}dZ$ 更大。由于 $\log P(X|\theta^{(t)})$ 是一个定值，那么也就是想让 KL 散度的值越小。所以，我们想让 KL 散度的值为零，也就是让 $Q(Z) = P(X,Z|\theta^{(t)})$。这样我们在固定了 $\theta^{(t)}$ 之后就得到了一个 ELBO 关于 θ 的函数。然后我们找到这个函数令值最大的 $\theta^{(t+1)}$ 后开始进行下

一步迭代。实际上我们的目的就是在不断的优化 ELBO，使 ELBO 不断的变大，那么我们想要的结果自然也就变大了，这是一个间接优化的方法。所以，我们紧接着公式 (9) 进行推导：

$$\begin{aligned}\hat{\theta} &= \arg\max_{\theta} \int Q(Z) \log \frac{P(X,Z|\theta)}{Q(Z)} dZ \\ &= \arg\max_{\theta} \int P(X,Z|\theta^{(t)}) \log \frac{P(X,Z|\theta)}{P(X,Z|\theta^{(t)})} dZ \\ &= \arg\max_{\theta} \int P(X,Z|\theta^{(t)}) \log P(X,Z|\theta) - P(X,Z|\theta^{(t)}) P(X,Z|\theta^{(t)}) dZ\end{aligned} \tag{11}$$

由于，$P(X,Z|\theta^{(t)})P(X,Z|\theta^{(t)})$ 与 θ 的求解无关。所以我们可以直接省略掉。那么下一步的 $\theta^{(t+1)}$ 的表达自然也就是：

$$\begin{aligned}\theta^{(t+1)} &= \arg\max_{\theta} \int_{Z} P(X,Z|\theta^{(t)}) \log P(X,Z|\theta) dZ \\ &= \arg\max_{\theta} \mathbb{E}_{Z\sim P(Z|X,\theta^{(t)})} [\log P(X,Z|\theta)]\end{aligned} \tag{12}$$

而这个公式 (12)，实际上就是我们之前直接给出的公式 (3) 和公式 (4)。

2 从 Jensen Inequality 的角度进行分析

首先，我们介绍一下什么是 Jensen Inequality。实际上，进行过一些机器学习理论研究的同学，都应该听说过这个概念。在这里我们做一个简述。首先我们需要保证函数是一个凸函数，下面我们来画一个凸函数：

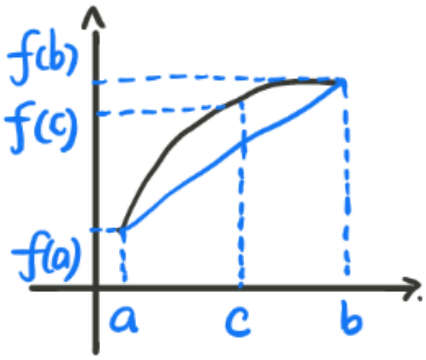

图 2: 凸函数示意图

那么对于一个 $t \in [0,1]$，$c = ta + (1-t)b$，我们都可以得到：

$$f(c) = f[ta + (1-t)b] \geq tf(a) + (1-t)f(b) \tag{13}$$

当 $t = \frac{1}{2}$ 时，我们可以得到：

$$f(\frac{(a+b)}{2}) \geq \frac{1}{2}[f(a) + f(b)] \qquad f[E] \geq E[f] \tag{14}$$

所以，我们可以利用 Jensen Inequality 进行推导：

$$\begin{aligned}\log P(X|\theta) &= \log \int_z P(X,Z|\theta)dZ \\ &= \log \int_z Q(Z)\frac{P(X,Z|\theta)}{Q(Z)}dZ \\ &= \log \mathbb{E}_{Z\sim Q(Z)}\left[\frac{P(X,Z|\theta)}{Q(Z)}\right] \\ &\geq \mathbb{E}_{Z\sim Q(Z)}\left[\log \frac{P(X,Z|\theta)}{Q(Z)}\right]\end{aligned} \tag{15}$$

根据 Jensen Inequality 的定义，当 $\frac{P(X,Z|\theta)}{Q(Z)} = C$ 时可以取得等号。不知道，大家有没有发现这里的 $\mathbb{E}_{Z\sim Q(Z)}\left[\log \frac{P(X,Z|\theta)}{Q(Z)}\right]$ 实际上就是 $\int_Z Q(Z)\log \frac{P(X,Z|\theta)}{Q(Z)}dZ$，也就是之前在 KL Divergence 角度进行分析时得到的 ELBO。

毫无疑问，当我们取等时，可以达到最大。所以有，

$$\frac{P(X,Z|\theta)}{Q(Z)} = C \tag{16}$$

$$Q(Z) = \frac{1}{C}P(X,Z|\theta) \tag{17}$$

$$\int_Z Q(Z)dZ = \frac{1}{C}\int_Z P(X,Z|\theta)dZ \tag{18}$$

$$1 = \frac{1}{C}P(X|\theta) \tag{19}$$

所以，我们就可以得到：

$$Q(Z) = \frac{P(X,Z|\theta)}{P(X|\theta)} = P(Z|X,\theta) \tag{20}$$

所以，大家有没有惊奇的发现，这个 $Q(Z)$ 实际上就是 Posterior。当时我们随便引入的一个分布 $Q(Z)$，没想到当它取等的时候就是后验分布。那么像不断去优化这个 ELBO，从而使得 $\log P(X|\theta)$ 的值不断的增加。由于，我们是迭代式的上升，这里的 $Q(Z) = P(Z|X,\theta^{(t)})$，而 $\theta^{(t)}$ 是上一次迭代得到的，我们可以认为是一个常数。所以，

$$\mathbb{E}_{Z\sim Q(Z)}\left[\log \frac{P(X,Z|\theta)}{Q(Z)}\right] = \mathbb{E}_{Z\sim Q(Z)}\left[\log \frac{P(X,Z|\theta)}{P(Z|X,\theta^{(t)})}\right] \tag{21}$$

所以，

$$\theta^{(t+1)} = \arg\max_{\theta} \mathbb{E}_{Z\sim Q(Z)}\left[\log \frac{P(X,Z|\theta)}{P(Z|X,\theta^{(t)})}\right] \tag{22}$$

所以，从 Jensen Inequality 的角度，我们仍然可以得到 EM 算法的核心表达式。

3 小结

在最后，我们再来来梳理一下 EM 算法的实现思想。我们的目标是想使 $P(X|\theta)$ 似然函数值最大。但是，很不幸，我们直接优化非常的难。所以，我们想到了一个优化下降的方法。对于，每一个 $\theta^{(t)}$ 时，我们可以计算得到下界为：$\mathbb{E}_{Z\sim Q(Z)}\left[\log \frac{P(X,Z|\theta)}{P(Z|X,\theta^{(t)})}\right]$，令这个值最大我们就得到了，想要求得的 $\theta^{(t+1)}$。然后，按这个思路，不断的进行迭代。

Expectation Maximization 03 Generalized Expectation Maximization

Chen Gong

19 December 2019

本小节中，我们想要介绍三个方便的知识点。1. 从狭义的 EM 算法推广到广义的 EM 算法；2. 狭义的 EM 实际上只是广义的 EM 的一个特例；3. 真正的开始介绍 EM 算法。

X：Observed Variable $\longrightarrow X = \{x_i\}_{i=1}^N$；

Z：Latent Variable $\longrightarrow Z = \{Z_i\}_{i=1}^N$；

(X, Z)：Complete Model；

θ：Model Parameter。

我们希望得到一个参数 θ，可以来推导出 X，也就是 $P(X|\theta)$。而这个参数怎么求得呢？所以，这就是一个 learning 的问题了。

1 极大似然估计

所以，根据极大似然估计法的思路，我们要求的最优化参数 $\hat{\theta}$ 为：

$$\begin{aligned}\hat{\theta} &= \arg\max_{\theta} P(X|\theta) \\ &= \arg\max_{\theta} \prod_{i=1}^{N} P(x_i|\theta) \\ &= \arg\max_{\theta} \sum_{i=1}^{N} \log P(x_i|\theta)\end{aligned} \tag{1}$$

好像，我们这样做就可以解决问题了呀。为什么要多此一举的来引入隐变量 Z 呢？这是因为，我们实际观察的输入空间 $\mathcal{X}$ 的分布 $P(X)$，是非常复杂的。可能什么规律都找不出来，这时我们就想到了一个很好的解决办法。我们引入了一个隐变量 Z，这个变量中包含了我们自己的一些归纳总结，引入了内部结构。而 $P(X) = \int_Z P(X, Z)dZ$，实际上就是对 X 进行了分解处理。

2 广义的 EM 算法

EM 算法是为了解决参数估计问题，也就是 learning 问题：

$$\hat{\theta} = \arg\max_{\theta} P(X|\theta) \tag{2}$$

但是，$P(X|\theta)$ 可能会非常的复杂。那么，在生成模型的思路中，可以假设一个隐变量 Z。有了这个生成模型的假设以后，我们就可以引入一些潜在归纳出的结构进去。通过 $P(X)=\frac{P(X,Z)}{P(Z|X)}$，就可以把问题具体化了。

这里说明一下，我们习惯用的表达是 $\log P(X|\theta)$，但是也有的文献中使用 $P(X;\theta)$ 或者 $P_\theta(X)$。这三种表达方式代表的意义是等价的。

前面我们已经说过了，我们的目标是：

$$\log P(X|\theta)=\underbrace{ELBO}_{L(Q,\theta)}+KL(Q||P)\geq L(Q,\theta) \tag{3}$$

$$\begin{cases} ELBO=\int_Z Q(Z)\log\frac{P(X,Z|\theta)}{Q(Z)}dZ \\ KL(Q||P)=\int_Z Q(Z)\log\frac{Q(Z)}{P(Z|X,\theta)}dZ \end{cases} \tag{4}$$

但是，问题马上就上来了，那就是 $P(Z|X,\theta)$ 非常有可能求不出来。那么我们怎么来求解这个方程呢？也就是使下界变得更大。

首先第一步，我们把 θ 给固定住。那么，$P(Z|X,\theta)$ 的结果就是一个定值。那么 KL 越小，ELBO 就会越大。由于，$Q(Z)$ 是我们引入的一个中间变量，那么我们的第一步就是得到：

$$\hat{Q}(Z)=\arg\min_Q KL(Q||P)=\arg\max_Q L(Q,\theta) \tag{5}$$

当 Q 被我们求出来以后，我们就可以将 Q 固定了，再来求解 θ：

$$\hat{\theta}=\arg\max_\theta L(\hat{q},\theta) \tag{6}$$

那么，广义的 EM 算法，就可以被我们定义为：

$$\begin{aligned} &E-step: \quad Q^{(t+1)}=\arg\max_Q L(Q(Z),\theta^{(t)}) \\ &M-step: \quad \theta^{(t+1)}=\arg\max_\theta L(Q(Z)^{(t+1)},\theta) \\ &L(Q,\theta)=\mathbb{E}_Q\left[\log P(X,Z)-\log Q\right]=\mathbb{E}_Q\left[\log P(X,Z)\right]-\mathbb{E}_Q\left[\log Q\right] \end{aligned} \tag{7}$$

看到这里，我估计大家已经可以理解上一小节中，为什么有的 θ 带 (t) 有的不带。因为，首先第一步中是固定 θ 求 Q，这里的 θ 就是来自于上一次迭代的 $\theta^{(t+1)}$。第二次，是将上一步求得的 Q 固定，将 θ 看成参数，来求最优的表达结果的 $\theta^{(t+1)}$。另一个方面，从等式 (7) 的第三行，我们可以可以看出实际上：

$$ELBO=\mathbb{E}_{Q(Z)}[\log P(X,Z|\theta)]+H(Q(Z)) \tag{8}$$

我们对比一下上一节讲到的 EM 算法，就会惊奇的发现，ELBO 中最后那个 $H(Q(Z))$ 竟然不见了。这是为什么呢？其实也很好理解的。因为在 M-step 中，我们假定 $Q(Z)$ 已经是固定的了，那么显然 $H[Q(Z)]$ 就是一个定值了，并且与我们的优化目标 θ 之间并没有任何的关系，所以就被我们给省略掉了。

所以，本小节中引出了广义 EM 算法，也说明了原来的 EM 算法是广义 EM 算法的一种特殊情况。

3 坐标上升法

EM 算法的整体描述如下所示：

$$\begin{cases} E-step: & Q^{(t+1)} = \arg\max_Q L(Q(Z), \theta^{(t)}) \\ M-step: & \theta^{(t+1)} = \arg\max_\theta L(Q(Z)^{(t+1)}, \theta) \end{cases} \tag{9}$$

这个坐标上升法 (SMO) 是个什么东西呢？具体的描述，大家可以去网上找找资料看一看。两者都是迭代的思路，在这里我们将它和梯度下降法的优化思路放在一起，做一个小小的对比。大家就会发现有什么不一样的地方，

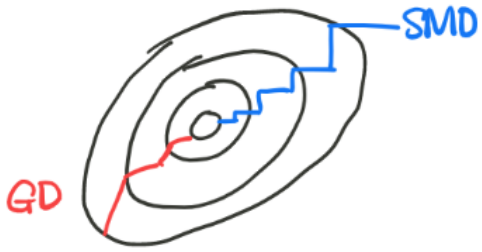

图 1: 坐标上升法和梯度上升法的优化思路对比

我们发现坐标上升法的优化方向基本是恒定不变的，而梯度下降法的优化方向是随着梯度方向而不断发生改变的。

讲到这里，好像一切都很完美，可以圆满的结束了。但是，很不幸的是，问题马上又来了。因为，现实生活中，并没有那么的容易，一切都没有我们想的那么的简单。实际上，有关 $P(Z|X,\theta)$ 的计算，有可能会非常的复杂。所以，我们将采用变分推断 (Variable Inference) 或者马尔可夫蒙特卡罗采样 (Markov Chain Monte Carlo) 的方法来求解。结合起来以后就是，VBEM/VEM 和 MCEM。这里注意一下，Variable Inference 和 Variable Bayes 实际上都是一种东西。

当然，虽然 EM 算法看上去好像很厉害的样子。但是，没有一种算法可以一劳永逸的解决所有的问题。它一定存在优点，也一定有无法解决的问题。具体描述，大家可以去网上寻找相关的资料，我这里就不做过多的描述了。

Gaussian Mixture Model 01 Model Introduction

Chen Gong

23 December 2019

这一章开始，我们将进入到 Guassian Mixture Model (GMM) 的学习。而为什么要学习 GMM 呢? 这是因为单峰分布已经不能准备的反映数据的分布了。正如下面的一个分布：

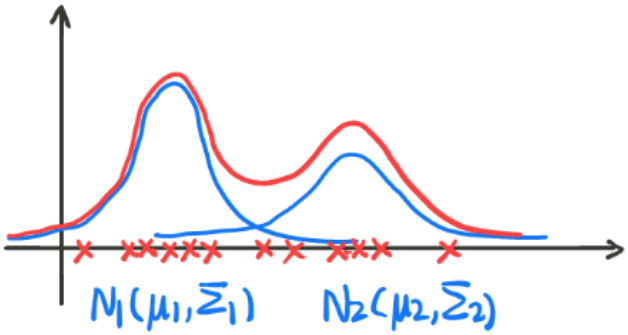

图 1: 数据分布举例

对于如上的数据分布来说，如果强行用单峰的 Guassian Distribution 来表示这个分布，显然是可以的。但是，很明显是不合适的。会造成较大的误差，不能较好的表示整个数据的分布特征。

1 从几何角度来看

从几何角度来看比较的简单，也就是多个高斯分布来取加权平均值。也就是一个混合高斯分布就是多个高斯分布叠加而成的。那么，概率密度函数，可以被我们写成：

$$p(x) = \sum_{k=1}^{K} \alpha_k \mathcal{N}(\mu_k, \Sigma_k), \qquad \sum_{k=1}^{K} \alpha_k = 1 \tag{1}$$

2 从混合模型角度来看 (生成模型)

如果当输入变量的维度高于一维的时候，我们就不能使用简单的加权来看了。因为，这时，我们已经无法简单的用加权平均来计算了，正如下图所示。其中，X 是 Observable Variable，Z 是 Latent Variable。这个 Z 是个什么意思呢? 我们先举一个小例子。看到图 2 中那个打了红圈圈的数据点。它既属于 C_1 的分布，并且也属于 C_2 的分布，我们可以写作：

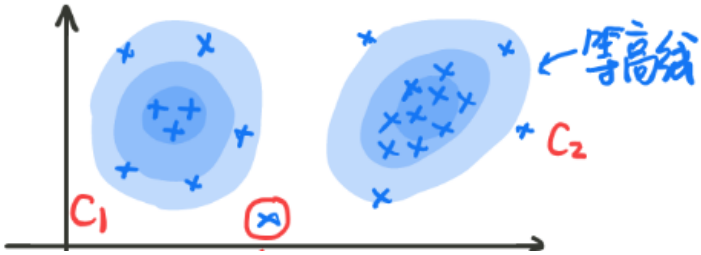

图 2: 二维数据分布举例

$$\begin{cases} X \sim C_1 \\ X \sim C_2 \end{cases} \tag{2}$$

这样写太麻烦了，我们可以直接写成 $X \sim Z$，这里的 Z 就是一个离散的随机变量，它包含了 $C_1, C_2, \cdots, C_N$ 的概率分布，Z 服从的是类别分布，其实就是看对应的样本 X 是属于哪一个高斯分布的概率。可以被我们写成：

Z	C_1	C_2	$\cdots$	C_k
$P(Z)$	P_1	P_2	$\cdots$	P_k

表 1: 隐变量 Z 的离散概率分布

并且，$\sum_k P_k = 1$。接下来，我们来说一说，如何来生成 N 个样本点，$x_1, x_2, \cdots, x_N$。

我们假设有一个骰子，有 K 个面，每个面都是不均匀的，假设我们可以控制每一个面的质量，那么这个骰子的面出现的概率会符合某个分布。有 K 个面，就有 K 个高斯分布。那么每次我们就投一下这个骰子，根据出现的面 K，选择在第 K 个高斯分布中进行采样，生成一个样本点 x_i。

概率图可以被我们描述为如下形式：

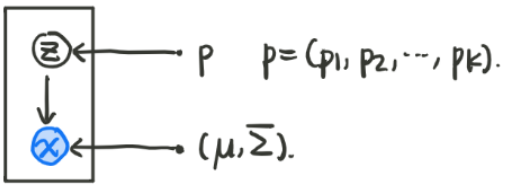

图 3: GMM 的概率图表达形式

我们根据一个离散的随机变量 Z 来选择是选取那个高斯分布，利用这个高斯分布 $\mathcal{N}(\mu, \Sigma)$ 来采样得到我们想要的样本点。而且，离散随机变量 Z 符合一个离散分布 $p = (p_1, p_2, \cdots, p_k)$。

Gaussian Mixture Model 02 Maximum Likelihood Estimation

Chen Gong

24 December 2019

本节我们想使用极大似然估计来求解 Gaussian Mixture Model (GMM) 的最优参数结果。首先，我们明确一下参数的意义：

X：Observed data，$X = (x_1, x_2, \cdots, x_N)$。

(X, Z)：Complete data，$(X, Z) = \{(x_1, z_1), (x_2, z_2), \cdots, (x_N, z_N)\}$。

θ：parameter，$\theta = \{P_1, \cdots, P_k, \mu_1, \cdots, \mu_k, \Sigma_1, \cdots, \Sigma_k\}$。

1 Maximum Likelihood Estimation 求解参数

$$
\begin{aligned}
P(x) &= \sum_{Z} P(X, Z) \\
&= \sum_{k=1}^{K} P(X, z = C_k) \\
&= \sum_{k=1}^{K} P(z = C_k) \cdot P(X | z = C_k) \\
&= \sum_{k=1}^{K} P_k \cdot \mathcal{N}(X | \mu_k, \Sigma_k)
\end{aligned}
\tag{1}
$$

其中，P_k 也就是数据点去第 k 个高斯分布的概率。下面我们开始使用 MLE 来求解 θ：

$$
\begin{aligned}
\hat{\theta}_{MLE} &= \arg\max_{\theta} \log P(X) \\
&= \arg\max_{\theta} \log \prod_{i=1}^{N} P(x_i) \\
&= \arg\max_{\theta} \sum_{i=1}^{N} \log P(x_i) \\
&= \arg\max_{\theta} \sum_{i=1}^{N} \log \sum_{k=1}^{K} P_k \cdot \mathcal{N}(x_i | \mu_k, \Sigma_k)
\end{aligned}
\tag{2}
$$

我们想要求的 θ 包括，$\theta = \{P_1, \cdots, P_k, \mu_1, \cdots, \mu_k, \Sigma_1, \cdots, \Sigma_k\}$。

2 MLE 的问题

按照之前的思路，我们就要分布对每个参数进行求偏导来计算最终的结果。但是问题马上就来了，大家有没有看到 log 函数里面是一个求和的形式，而不是一个求积的形式。这意味着计算非常的困难。甚至可以说，我们根本就求不出解析解。如果是单一的 Gaussian Distribution：

$$\log P(x_i) = \log \frac{1}{\sqrt{2\pi}\sigma} exp\left\{-\frac{(x_i-\mu)^2}{2\sigma}\right\} \tag{3}$$

根据 log 函数优秀的性质，这个问题是可以解的。但是，很不幸后面是一个求和的形式。所以，直接使用 MLE 求解 GMM，无法得到解析解。

Gaussian Mixture Model 03 Expectation Maximization

Chen Gong

25 December 2019

上一小节中，我们看到了使用极大似然估计的方法，我们根本就求不出最优参数 θ 的解析解。所以，我们使用迭代的方法来求近似解。

EM 算法的表达式，可以被我们写为：

$$\theta^{(t+1)} = \arg\max_{\theta} \underbrace{\mathbb{E}_{P(Z|X,\theta^{(t)})}\left[\log P(X,Z|\theta)\right]}_{Q(\theta,\theta^{(t)})} \tag{1}$$

经过一系列的迭代，我们可以得到 $\theta^0, \theta^1, \cdots, \theta^{(t)}$，迭代到一定次数以后我们得到的 $\theta^{(N)}$ 就是我们想要得到的结果。EM 算法大体上可以分成两个部分，E-step 和 M-step，

1 E-Step

$$\begin{aligned}
Q(\theta,\theta^{(t)}) &= \int_Z \log P(X,Z|\theta)\cdot P(Z|X,\theta^{(t)})dZ \\
&= \sum_Z \log \prod_{i=1}^N P(x_i,z_i|\theta)\cdot \prod_{i=1}^N P(z_i|x_i,\theta^{(t)}) \\
&= \sum_{z_1,\cdots,z_N}\sum_{i=1}^N \log P(x_i,z_i|\theta)\cdot \prod_{i=1}^N P(z_i|x_i,\theta^{(t)}) \\
&= \sum_{z_1,\cdots,z_N} \left[\log P(x_1,z_1|\theta)+\log P(x_2,z_2|\theta)+\cdots \log P(x_N,z_N|\theta)\right]\cdot \prod_{i=1}^N P(z_i|x_i,\theta^{(t)})
\end{aligned} \tag{2}$$

为了简化推导，我们首先只取第一项来化简一下，

$$\begin{aligned}
&\sum_{z_1,\cdots,z_N} \log P(x_1,z_1|\theta)\cdot \prod_{i=1}^N P(z_i|x_i,\theta^{(t)})dZ \\
=&\sum_{z_1,\cdots,z_N} \log P(x_1,z_1|\theta)\cdot P(z_1|x_1,\theta^{(t)})\cdot \prod_{i=2}^N P(z_i|x_i,\theta^{(t)}) \\
=&\sum_{z_1} \log P(x_1,z_1|\theta)\cdot P(z_1|x_1,\theta^{(t)})\cdot \sum_{z_2,\cdots,z_N}\prod_{i=2}^N P(z_i|x_i,\theta^{(t)})
\end{aligned} \tag{3}$$

而：

$$\begin{aligned}\sum_{z_2,\cdots,z_N}\prod_{i=2}^{N}P(z_i|x_i,\theta^{(t)}) &= \sum_{z_2,\cdots,z_N} P(z_2|x_2,\theta^{(t)})\cdot P(z_3|x_3,\theta^{(t)})\cdots P(z_N|x_N,\theta^{(t)}) \\ &= \sum_{z_2} P(z_2|x_2,\theta^{(t)})\cdot\sum_{z_3} P(z_3|x_3,\theta^{(t)})\cdots\sum_{z_N} P(z_N|x_N,\theta^{(t)}) \\ &= 1\cdot 1\cdots 1 \\ &= 1\end{aligned} \tag{4}$$

所以，式 (3) 也就等于：

$$\sum_{z_1,\cdots,z_N}\log P(x_1,z_1|\theta)\cdot\prod_{i=1}^{N}P(z_i|x_i,\theta^{(t)})dZ = \sum_{z_1}\log P(x_1,z_1|\theta)\cdot P(z_1|x_1,\theta^{(t)}) \tag{5}$$

将式 (5) 中得到的结果，代入到式 (2) 中，我们就可以得到：

$$\begin{aligned}Q(\theta,\theta^{(t)}) &= \sum_{z_1}\log P(x_1,z_1|\theta)\cdot P(z_1|x_1,\theta^{(t)}) + \cdots + \sum_{z_N}\log P(x_N,z_N|\theta)\cdot P(z_N|x_N,\theta^{(t)}) \\ &= \sum_{i=1}^{N}\sum_{Z_i}\log P(x_i,z_i|\theta)\cdot P(z_i|x_i,\theta^{(t)})\end{aligned} \tag{6}$$

那么，下一步我们就是要找到，$P(x_i,z_i|\theta)$ 和 $P(z_i|x_i,\theta^{(t)})$ 的表达方式了。其中：

$$P(X,Z) = P(Z)P(X|Z) = P_Z\cdot\mathcal{N}(X|\mu_Z,\Sigma_Z) \tag{7}$$

$$P(Z|X) = \frac{P(X,Z)}{P(X)} = \frac{P_Z\cdot\mathcal{N}(X|\mu_Z,\Sigma_Z)}{\sum_{i=1}^{K}P_{Zi}\cdot\mathcal{N}(X|\mu_{Zi},\Sigma_{Zi})} \tag{8}$$

所以，我们将式 (8) 代入到式 (6) 中，就可以得到：

$$Q(\theta,\theta^{(t)}) = \sum_{i=1}^{N}\sum_{Z_i}\log P_{Z_i}\cdot\mathcal{N}(X|\mu_{Z_i},\Sigma_{Z_i})\cdot\frac{P_{Z_i}^{\theta^{(t)}}\cdot\mathcal{N}(x_i|\mu_{Z_i}^{\theta^{(t)}},\Sigma_{Z_i}^{\theta^{(t)}})}{\sum_{k=1}^{K}P_k^{\theta^{(t)}}\cdot\mathcal{N}(x_i|\mu_k^{\theta^{(t)}},\Sigma_k^{\theta^{(t)}})} \tag{9}$$

2 M-Step

根据我们在 E-Step 中的推导，我们可以得到：

$$\begin{aligned}Q(\theta,\theta^{(t)}) &= \sum_{i=1}^{N}\sum_{Z_i}\log P_{Z_i}\cdot\mathcal{N}(X|\mu_{Z_i},\Sigma_{Z_i})\cdot\underbrace{\frac{P_{Z_i}^{\theta^{(t)}}\cdot\mathcal{N}(x_i|\mu_{Z_i}^{\theta^{(t)}},\Sigma_{Z_i}^{\theta^{(t)}})}{\sum_{k=1}^{K}P_k^{\theta^{(t)}}\cdot\mathcal{N}(x_i|\mu_k^{\theta^{(t)}},\Sigma_k^{\theta^{(t)}})}}_{P(Z_i|X_i,,\theta^{(t)})} \\ &= \sum_{i=1}^{N}\sum_{k=1}^{K}\log\left(P_k\cdot\mathcal{N}(X|\mu_k,\Sigma_k)\right)\cdot P(Z_i=C_k|X_i,,\theta^{(t)}) \\ &= \sum_{k=1}^{K}\sum_{i=1}^{N}\log\left(P_k\cdot\mathcal{N}(X|\mu_k,\Sigma_k)\right)\cdot P(Z_i=C_k|X_i,,\theta^{(t)}) \\ &= \sum_{k=1}^{K}\sum_{i=1}^{N}\left(\log P_k + \log\mathcal{N}(X_i|\mu_k,\Sigma_k)\right)\cdot P(Z_i=C_k|X_i,\theta^{(t)})\end{aligned} \tag{10}$$

我们的目的也就是进行不断迭代，从而得出最终的解，用公式表达也就是：

$$\theta^{(t+1)} = \arg\max_{\theta} Q(\theta, \theta^{(t)}) \tag{11}$$

我们需要求解的参数也就是，$\theta^{(t+1)} = \{P_1^{(t+1)}, \cdots, P_k^{(t+1)}, \mu_1^{(t+1)}, \cdots, \mu_k^{(t+1)}, \Sigma_1^{(t+1)}, \cdots, \Sigma_k^{(t+1)}\}$。

首先，我们来展示一下怎么求解 $P_K^{(t+1)}$：

由于在等式 (10), $\sum_{k=1}^{K}\sum_{i=1}^{N}(\log P_k + \log \mathcal{N}(X|\mu_k, \Sigma_k)) \cdot P(Z_i = C_k|X_i, , \theta^{(t)})$ 中的 $\log \mathcal{N}(X|\mu_k, \Sigma_k)$ 部分和 P_k 并没有什么关系。所以，可以被我们直接忽略掉。所以，求解问题，可以被我们描述为：

$$\begin{cases} \arg\max_{P_k} \sum_{k=1}^{K}\sum_{i=1}^{N} \log P_k \cdot P(Z_i = C_k|X_i, \theta^{(t)}) \\ s.t. \quad \sum_{k=1}^{K} P_k = 1 \end{cases} \tag{12}$$

使用拉格朗日算子法，我们可以写成：

$$\mathcal{L}(P, \lambda) = \sum_{k=1}^{K}\sum_{i=1}^{N} \log P_k \cdot P(Z_i = C_k|X_i, \theta^{(t)}) + \lambda(\sum_{k=1}^{K} P_k - 1) \tag{13}$$

$$\begin{aligned} \frac{\partial \mathcal{L}(P, \lambda)}{\partial P_k} &= \sum_{i=1}^{N} \frac{1}{P_k} \cdot P(Z_i = C_k|X_i, \theta^{(t)}) + \lambda = 0 \\ &\Rightarrow \sum_{i=1}^{N} P(Z_i = C_k|X_i, \theta^{(t)}) + P_k\lambda = 0 \\ &\overset{k=1,\cdots,K}{\Longrightarrow} \sum_{i=1}^{N} \underbrace{\sum_{k=1}^{K} P(Z_i = C_k|X_i, \theta^{(t)})}_{1} + \underbrace{\sum_{k=1}^{K} P_k}_{1} \lambda = 0 \\ &\Rightarrow N + \lambda = 0 \end{aligned} \tag{14}$$

所以，我们可以轻易的得到 $\lambda = -N$，所以有

$$P_K^{(t+1)} = \frac{1}{N}\sum_{i=1}^{N} P(Z_i = C_k|X_i, \theta^{(t)}) \tag{15}$$

那么，我们所有想要求的参数也就是 $P^{(t+1)} = (P_1^{(t+1)}, P_2^{(t+1)}, \cdots, P_k^{(t+1)})$。

求解 $P_k^{(t+1)}$ 是一个有约束的求最大值问题，由于带约束所以我们要使用拉格朗日乘子法。而且这里使用到了一个 track，也就是将从 1 到 k，所有的数据集做一个整合，非常的精彩，这样就直接消掉了 P_k 无法计算的问题。而至于 θ 的其他部分，也就是关于 $\{\mu_1^{(t+1)}, \cdots, \mu_k^{(t+1)}, \Sigma_1^{(t+1)}, \cdots, \Sigma_k^{(t+1)}\}$ 的计算，使用的方法也是一样的，这个问题就留给各位了。

为什么极大似然估计搞不定的问题，放在 EM 算法里面我们就可以搞定了呢？我们来对比一下两个方法中，需要计算极值的公式。

$$\sum_{k=1}^{K}\sum_{i=1}^{N} (\log P_k + \log \mathcal{N}(X_i|\mu_k, \Sigma_k)) \cdot P(Z_i = C_k|X_i, \theta^{(t)}) \tag{16}$$

$$\arg\max_{\theta} \sum_{i=1}^{N} \log \sum_{k=1}^{K} P_k \cdot \mathcal{N}(x_i|\mu_k, \Sigma_k) \tag{17}$$

极大似然估计一开始计算的就是 $P(X)$，而 EM 算法中并没有出现有关 $P(X)$ 的计算，而是全程计算都是 $P(X, Z)$。而 $P(X)$ 实际上就是 $P(X, Z)$ 的求和形式。所以，每次单独的考虑 $P(X, Z)$ 就避免了在 log 函数中出现求和操作。

Variational Inference 01 Background

Chen Gong

30 November 2019

这一小节的主要目的是清楚我们为什么要使用 Variational Inference，表达一下 Inference 到底有什么用。机器学习，我们可以从频率角度和贝叶斯角度两个角度来看，其中频率角度可以被解释为优化问题，贝叶斯角度可以被解释为积分问题。

1 优化问题

为什么说频率派角度的分析是一个优化问题呢？我们从回归和 SVM 两个例子上进行分析。我们将数据集描述为：$D = \{(x_i, y_i)\}_{i=1}^N, x_i \in \mathbb{R}^p, y_i \in \mathbb{R}$。

1.1 回归

回归模型可以被我们定义为：$f(w) = w^T x$，其中 loss function 被定义为：$L(w) = \sum_{i=1}^N ||w^T x_i - y_i||^2$，优化可以表达为 $\hat{w} = argmin\ L(w)$。这是个无约束优化问题。

求解的方法可以分成两种，数值解和解析解。解析解的解法为：

$$\frac{\partial L(w)}{\partial w} = 0 \Rightarrow w^* = (X^T X)^{-1} X^T Y \tag{1}$$

其中，X 是一个 $n \times p$ 的矩阵。而数值解中，我们常用的是 GD 算法，也就是 Gradient Descent，或者 Stochastic Gradient descent (SGD)。

1.2 SVM (Classification)

SVM 的模型可以被我们表述为：$f(w) = sign(w^T + b)$。loss function 被我们定义为：

$$\begin{cases} \min\ \frac{1}{2} w^T w \\ s.t.\quad y_i(w^T x_i + b) \geq 1 \end{cases} \tag{2}$$

很显然这是一个有约束的 Convex 优化问题。常用的解决条件为，QP 方法和 Lagrange 对偶。

1.3 EM 算法

我们的优化目标为：

$$\hat{\theta} = \arg\max_{\theta}\ \log p(X|\theta) \tag{3}$$

优化的迭代算法为：

$$\theta^{(t+1)} = \arg\max_{\theta} \int_z \log p(X, Z|\theta) \cdot p(Z|X, \theta^{(t)}) dz \tag{4}$$

2 积分问题

从贝叶斯的角度来说，这就是一个积分问题，为什么呢？我们看看 Bayes 公式的表达：

$$p(\theta|x) = \frac{p(x|\theta)p(\theta)}{p(x)} \tag{5}$$

其中，$p(\theta|x)$ 称为后验公式，$p(x|\theta)$ 称为似然函数，$p(\theta)$ 称为先验分布，并且 $p(x) = \int_\theta p(x|\theta)p(\theta)d\theta$。什么是推断呢？通俗的说就是求解后验分布 $p(\theta|x)$。而 $p(\theta|x)$ 的计算在高维空间的时候非常的复杂，我们通常不能直接精确的求得，这是就需要采用方法来求一个近似的解。而贝叶斯的方法往往需要我们解决一个贝叶斯决策的问题，也就是根据数据集 X(N 个样本)。我们用数学的语言来表述也就是，$\widetilde{X}$ 为新的样本，求 $p(\widetilde{X}|X)$：

$$\begin{aligned} p(\widetilde{X}|X) &= \int_\theta p(\widetilde{X}, \theta|X)d\theta \\ &= \int_\theta p(\widetilde{X}|\theta) \cdot p(\theta|X)d\theta \\ &= \mathbb{E}_{\theta|X}[p(\widetilde{X}|\theta)] \end{aligned} \tag{6}$$

其中 $p(\theta|X)$ 为一个后验分布，那么我们关注的重点问题就是求这个积分。

3 Inference

Inference 的方法可以被我们分为精确推断和近似推断，近似推断可以被我们分为确定性推断和随机近似。确定性推断包括 Variational Inference (VI)；随机近似包括 MCMC，MH，Gibbs Sampling 等

Variational Inference 02 Algorithm

Chen Gong

30 November 2019

我们将 X：Observed data；Z：Latent Variable + Parameters。那么 (X, Z) 为 complete data。根据我们的贝叶斯分布公式：

$$p(X) = \frac{p(X, Z)}{p(Z|X)} \tag{1}$$

在两边同时取对数并且引入函数 $q(Z)$ 我们可以得到：

$$\begin{aligned} \log p(X) &= \log \frac{p(X, Z)}{p(Z|X)} \\ &= \log p(X, Z) - \log p(Z|X) \\ &= \log \frac{p(X, Z)}{q(Z)} - \log \frac{p(Z|X)}{q(Z)} \end{aligned} \tag{2}$$

1 公式化简

左边 $= p(X) = \int_Z log\ p(X)q(Z)dZ$。

右边 $=$

$$\int_Z q(Z) \log \frac{p(X, Z)}{q(Z)} dZ - \int_Z q(Z) \log \frac{p(Z|X)}{q(Z)} dZ \tag{3}$$

其中，$\int_Z q(Z) \log \frac{p(X,Z)}{q(Z)} dZ$ 被称为 Evidence Lower Bound (ELBO)，被我们记为 $\mathcal{L}(q)$，也就是变分。

$-\int_Z q(Z) \log \frac{p(Z|X)}{q(Z)} dZ$ 被称为 $KL(q||p)$。这里的 $KL(q||p) \geq 0$。

由于我们求不出 $p(Z|X)$,我们的目的是寻找一个 $q(Z)$,使得 $p(Z|X)$ 近似于 $q(Z)$,也就是 $KL(q||p)$ 越小越好。并且，$p(X)$ 是个定值，那么我们的目标变成了 $\arg\max_{q(z)} \mathcal{L}(q)$。那么，我们理一下思路，我们想要求得一个 $\widetilde{q}(Z) \approx p(Z|X)$。也就是

$$\widetilde{q}(Z) = \arg\max_{q(z)} \mathcal{L}(q) \Rightarrow \widetilde{q}(Z) \approx p(Z|X) \tag{4}$$

2 模型求解

那么我们如何来求解这个问题呢?我们使用到统计物理中的一种方法,就是平均场理论 (mean field theory)。也就是假设变分后验分式是一种完全可分解的分布：

$$q(z) = \prod_{i=1}^{M} q_i(z_i) \tag{5}$$

在这种分解的思想中，我们每次只考虑第 j 个分布，那么令 $q_i(1,2,\cdots,j-1,j+1,\cdots,M)$ 个分布 fixed。

那么很显然：

$$\mathcal{L}(q) = \int_Z q(Z)\log p(X,Z)dz - \int_Z q(Z)\log q(Z)dZ \tag{6}$$

我们先来分析第一项 $\int_Z q(Z)\log p(X,Z)dZ$。

$$\begin{aligned}\int_Z q(Z)\log p(X,Z)dZ &= \int_Z \prod_{i=1}^M q_i(z_i)\log p(X,Z)dZ \\ &= \int_{z_j} q_j(z_j)\left[\int_{z_1}\int_{z_2}\cdots\int_{z_M}\prod_{i=1}^M q_i(z_i)\log p(X,Z)dz_1dz_2\cdots dz_M\right]dz_j \\ &= \int_{z_j} q_j(z_j)\mathbb{E}_{\prod_{i\neq j}^M q_i(x_i)}\left[\log p(X,Z)\right]dz_j\end{aligned} \tag{7}$$

然后我们来分析第二项 $\int_Z q(Z)\log q(Z)dZ$，

$$\begin{aligned}\int_Z q(Z)\log q(Z)dZ &= \int_Z \prod_{i=1}^M q_i(z_i)\sum_{i=1}^M \log q_i(z_i)dZ \\ &= \int_Z \prod_{i=1}^M q_i(z_i)\left[\log q_1(z_1) + q_2(z_2) + \cdots + q_M(z_M)\right]dZ\end{aligned} \tag{8}$$

这个公式的计算如何进行呢？我们抽出一项来看，就会变得非常的清晰：

$$\begin{aligned}\int_Z \prod_{i=1}^M q_i(z_i)\log q_1(z_1)dZ &= \int_{z_1z_2\cdots z_M} q_1q_2\cdots q_M\log q_1dz_1dz_2\cdots z_M \\ &= \int_{z_1} q_1\log q_1dz_1\cdot\int_{z_2}q_2dz_2\cdot\int_{z_3}q_3dz_3\cdots\int_{z_M}q_Mdz_M \\ &= \int_{z_1} q_1\log q_1dz_1\end{aligned} \tag{9}$$

因为，$\int_{z_2} q_2dz_2$ 每一项的值都是 1。所以第二项可以写为：

$$\sum_{i=1}^M\int_{z_i} q_i(z_i)\log q_i(z_i)dz_i = \int_{z_j} q_j(z_j)\log q_i(z_i)dz_j + C \tag{10}$$

因为我们仅仅只关注第 j 项，其他的项都不关注。为了进一步表达计算，我们将：

$$\mathbb{E}_{\prod_{i\neq j}^M q_i(z_i)}\left[\log p(X,Z)\right] = \log\hat{p}(X,z_j) \tag{11}$$

那么 (8) 式可以写作：

$$\int_{z_j} q_j(z_j)\log\hat{p}(X,z_j)dz_j \tag{12}$$

这里的 $\hat{p}(X,z_j)$ 表示为一个相关的函数形式，假设具体参数未知。那么 (7) 式将等于 (13) 式减 (11) 式：

$$\int_{z_j} q_j(z_j)\log q_i(z_i)dz_j - \int_{z_j} q_j(z_j)\log\hat{p}(X,z_j)dz_j + C = -KL(q_j||\hat{p}(x,z_j)) + C \tag{13}$$

$\arg\max_{q_j(z_j)} -KL(q_j||\hat{p}(x,z_j))$ 等价于 $\arg\min_{q_j(z_j)} KL(q_j||\hat{p}(x,z_j))$。那么这个 $KL(q_j||\hat{p}(x,z_j))$ 要如何进行优化呢？我们下一节将回归 EM 算法，并给出求解的过程。

Variational Inference 03 Algorithm Solution

Chen Gong

01 December 2019

在上一小节中，我们介绍了 Mean Field Theory Variational Inference 的方法。在这里我需要进一步做一些说明，z_i 表示的不是一个数，而是一个数据维度的集合，它表示的不是一个维度，而是一个类似的最大团，也就是多个维度凑在一起。在上一节中，我们得出：

$$\log q_j(z_j) = \mathbb{E}_{\prod_{i\neq j} q_i(z_i)} \left[\log p(X, Z|\theta)\right] + C \tag{1}$$

并且，我们令数据集为 $X = \{x^{(i)}\}_{i=1}^N$，$Y = \{y^{(i)}\}_{i=1}^N$。variation 的核心思想是在于用一个分布 q 来近似得到 $p(z|x)$。其中优化目标为，$\hat{q} = \arg\min\ KL(q||p)$。其中：

$$\log p(X|\theta) = ELBO(\mathcal{L}(q)) + KL(q||p) \geq \mathcal{L}(q) \tag{2}$$

在这个求解中，我们主要想求的是 $q(x)$，那么我们需要弱化 θ 的作用。所以，我们计算的目标函数为：

$$\hat{q} = \arg\min_q KL(q||p) = \arg\max_q \mathcal{L}(q) \tag{3}$$

在上一小节中，这是我们的便于观察的表达方法，但是我们需要严格的使用我们的数学符号。

1 数学符号规范化

在这里我们弱化了相关参数 θ，也就是求解过程中，不太考虑 θ 起到的作用。我们展示一下似然函数，

$$\log p_\theta(X) = \log \prod_{i=1}^N p_\theta(x^{(i)}) = \sum_{i=1}^N \log p_\theta(x^{(i)}) \tag{4}$$

我们的目标是使每一个 $x^{(i)}$ 最大，所以将对 ELBO 和 $KL(p||q)$ 进行规范化表达：

ELBO：

$$\mathbb{E}_{q(z)}\left[\log \frac{p_\theta(x^{(i)}, z)}{q(z)}\right] = \mathbb{E}_{q(z)}\left[\log p_\theta(x^{(i)}, z)\right] + H(q(z)) \tag{5}$$

KL:

$$KL(q||p) = \int q(z) \cdot \log \frac{q(z)}{p_\theta(z|x^{(i)})} dz \tag{6}$$

而，

$$\begin{aligned}\log q_j(z_j) =& \mathbb{E}_{\prod_{i\neq j} q_i(z_i)}\left[\log p_\theta(x^{(i)}, z)\right] + C \\ =& \int_{q_1}\int_{q_2}\cdots\int_{q_{j-1}}\int_{q_{j+1}}\cdots\int_{q_M} q_1 q_2 \cdots q_{j-1} q_{j+1} \cdots q_M dq_1 dq_2 \cdots dq_{j-1} dq_{j+1} \cdots dq_M\end{aligned} \tag{7}$$

2 迭代算法求解

在上一步中，我们已经将所有的符号从数据点和划分维度上进行了规范化的表达。在这一步中，我们将使用迭代算法来进行求解：

$$\hat{q}_1(z_1) = \int_{q_2} \cdots \int_{q_M} q_2 \cdots q_M \left[\log p_\theta(x^{(i)}, z)\right] dq_2 \cdots dq_M \tag{8}$$

$$\hat{q}_2(z_2) = \int_{\hat{q}_1(z_1)} \int_{q_3} \cdots \int_{q_M} \hat{q}_1 q_3 \cdots q_M \left[\log p_\theta(x^{(i)}, z)\right] \hat{q}_1 dq_2 \cdots dq_M \tag{9}$$

$$\vdots$$

$$\hat{q}_M(z_M) = \int_{\hat{q}_1} \cdots \int_{\hat{q}_{M-1}} \hat{q}_1 \cdots \hat{q}_{M-1} \left[\log p_\theta(x^{(i)}, z)\right] d\hat{q}_1 \cdots d\hat{q}_{M-1} \tag{10}$$

如果，我们将 $q_1, q_2, \cdots, q_M$ 看成一个个的坐标点，那么我们知道的坐标点越来越多，这实际上就是一种坐标上升的方法 (Coordinate Ascend)。

这是一种迭代算法，那我们怎么考虑迭代的停止条件呢？我们设置当 $\mathcal{L}^{(t+1)} \leq \mathcal{L}^{(t)}$ 时停止迭代。

3 Mean Field Theory 的存在问题

1. 首先假设上就有问题，这个假设太强了。在假设中，我们提到，假设变分后验分式是一种完全可分解的分布。实际上，这样的适用条件挺少的。大部分时候都并不会适用。

2. Intractable。本来就是因为后验分布 $p(Z|X)$ 的计算非常的复杂，所以我们才使用变分推断来进行计算，但是有个很不幸的消息。这个迭代的方法也非常的难以计算，并且

$$\log q_j(z_j) = \mathbb{E}_{\prod_{i \neq j} q_i(z_i)} \left[\log p(X, Z|\theta)\right] + C \tag{11}$$

的计算也非常的复杂。所以，我们需要寻找一种更加优秀的方法，比如 Stein Disparency 等等。Stein 变分是个非常 Fashion 的东西，机器学习理论中非常强大的算法，我们以后会详细的分析。

Variational Inference 04 Stochastic Gradient Variational Inference

Chen Gong

01 December 2019

在上一小节中，我们分析了 Mean Field Theory Variational Inference，通过平均假设来得到变分推断的理论，是一种 classical VI，我们可以将其看成 Coordinate Ascend。而另一种方法是 Stochastic Gradient Variational Inference (SGVI)。

对于隐变量参数 z 和数据集 x。$z \longrightarrow x$ 是 Generative Model，也就是 $p(x|z)$ 和 $p(x,z)$，这个过程也被我们称为 Decoder。$x \longrightarrow z$ 是 Inference Model，这个过程被我们称为 Encoder，表达关系也就是 $p(z|x)$。

1 SGVI 参数规范

我们这节的主题就是 Stochastic Gradient Variational Inference (SGVI)，参数的更新方法为：

$$\theta^{(t+1)} = \theta^{(t)} + \lambda^{(t)} \nabla \mathcal{L}(q) \tag{1}$$

其中，$q(z|x)$ 被我们简化表示为 $q(z)$，我们令 $q(z)$ 是一个固定形式的概率分布，ϕ 为这个分布的参数，那么我们将把这个概率写成 $q_\phi(z)$。

那么，我们需要对原等式中的表达形式进行更新，

$$ELBO = \mathbb{E}_{q_\phi(z)} \left[\log p_\theta(x^{(i)}, z) - \log q_\phi(z) \right] = \mathcal{L}(\phi) \tag{2}$$

而，

$$\log p_\theta(x^{(i)}) = ELBO + KL(q||p) \geq \mathcal{L}(\phi) \tag{3}$$

而求解目标也转换成了：

$$\hat{p} = \arg\max_\phi \mathcal{L}(\phi) \tag{4}$$

2 SGVI 的梯度推导

$$\begin{aligned} \nabla_\phi \mathcal{L}(\phi) =& \nabla_\phi \mathbb{E}_{q_\phi} \left[\log p_\theta(x^{(i)}, z) - \log q_\phi \right] \\ =& \nabla_\phi \int q_\phi \left[\log p_\theta(x^{(i)}, z) - \log q_\phi \right] dz \\ =& \int \nabla_\phi q_\phi \left[\log p_\theta(x^{(i)}, z) - \log q_\phi \right] dz + \int q_\phi \nabla_\phi \left[\log p_\theta(x^{(i)}, z) - \log q_\phi \right] dz \end{aligned} \tag{5}$$

我们把这个等式拆成两个部分，其中：
$\int \nabla_\phi q_\phi \left[\log p_\theta(x^{(i)}, z) - \log q_\phi\right] dz$ 为第一个部分；
$\int q_\phi \nabla_\phi \left[\log p_\theta(x^{(i)}, z) - \log q_\phi\right] dz$ 为第二个部分。

2.1 关于第二部分的求解

第二部分比较好求，所以我们才首先求第二部分的，哈哈哈！因为 $\log p_\theta(x^{(i)}, z)$ 与 ϕ 无关。

$$\begin{aligned} 2 =& \int q_\phi \nabla_\phi \left[\log p_\theta(x^{(i)}, z) - \log q_\phi\right] dz \\ =& -\int q_\phi \nabla_\phi \log q_\phi dz \\ =& -\int q_\phi \frac{1}{q_\phi} \nabla_\phi q_\phi dz \\ =& -\int \nabla_\phi q_\phi dz \\ =& -\nabla_\phi \int q_\phi dz \\ =& -\nabla_\phi 1 \\ =& 0 \end{aligned} \tag{6}$$

2.2 关于第一部分的求解

在这里我们用到了一个小 trick，这个 trick 在公式 (6) 的推导中，我们使用过的。那就是 $\nabla_\phi q_\phi = q_\phi \nabla_\phi \log q_\phi$。所以，我们代入到第一项中可以得到：

$$\begin{aligned} 1 =& \int \nabla_\phi q_\phi \left[\log p_\theta(x^{(i)}, z) - \log q_\phi\right] dz \\ =& \int q_\phi \nabla_\phi \log q_\phi \left[\log p_\theta(x^{(i)}, z) - \log q_\phi\right] dz \\ =& \mathbb{E}_{q_\phi}\left[\nabla_\phi \log q_\phi \log p_\theta(x^{(i)}, z) - \log q_\phi\right] \end{aligned} \tag{7}$$

那么，我们可以得到：

$$\nabla_\phi \mathcal{L}(\phi) = \mathbb{E}_{q_\phi}\left[\nabla_\phi \log q_\phi \log p_\theta(x^{(i)}, z) - \log q_\phi\right] \tag{8}$$

那么如何求这个期望呢？我们采用的是蒙特卡罗采样法，假设 $z^l \sim q_\phi(z)\ l = 1, 2, \cdots, L$，那么有：

$$\nabla_\phi \mathcal{L}(\phi) \approx \frac{1}{L} \sum_{l=1}^{L} \nabla_\phi \log q_\phi(z^{(l)}) \left[\log p_\theta(x^{(i)}, z) - \log q_\phi(z^{(l)})\right] \tag{9}$$

由于第二部分的结果为 0，所以第一部分的解就是最终的解。但是，这样的求法有什么样的问题呢？因为我们在采样的过程中，很有可能采到 $q_\phi(z) \longrightarrow 0$ 的点，对于 log 函数来说，$\lim_{x \longrightarrow 0} \log x = \infty$，那么梯度的变化会非常的剧烈，非常的不稳定。对于这样的 High Variance 的问题，根本没有办法求解。实际上，我们可以通过计算得到这个方差的解析解，它确实是一个很大的值。事实上，这里的梯度的方差这么的大，而 $\hat{\phi} \longrightarrow q(z)$ 也有误差，误差叠加，直接爆炸，根本没有办法用。也就是不会 work，那么我们如何解决这个问题？

3 Variance Reduction

这里采用了一种比较常见的方差缩减方法，称为 Reparameterization Trick，也就是对 q_ϕ 做一些简化。

我们怎么可以较好的解决这个问题? 如果我们可以得到一个确定的解 $p(\epsilon)$, 就会变得比较简单。因为 z 来自于 $q_\phi(z|x)$, 我们就想办法将 z 中的随机变量给解放出来。也就是使用一个转换 $z = g_\phi(\epsilon, x^{(i)})$, 其中 $\epsilon \sim p(\epsilon)$。那么这样做，有什么好处呢? 原来的 $\nabla_\phi \mathbb{E}_{q_\phi}[\cdot]$ 将转换为 $\mathbb{E}_{p(\epsilon)}[\nabla_\phi(\cdot)]$，那么不在是连续的关于 ϕ 的采样，这样可以有效的降低方差。并且，z 是一个关于 ϵ 的函数，我们将随机性转移到了 ϵ，那么问题就可以简化为:

$$z \sim q_\phi(z|x^{(i)}) \longrightarrow \epsilon \sim p(\epsilon) \tag{10}$$

而且，这里还需要引入一个等式，那就是:

$$|q_\phi(z|x^{(i)})dz| = |p(\epsilon)d\epsilon| \tag{11}$$

为什么呢? 我们直观性的理解一下，$\int q_\phi(z|x^{(i)})dz = \int p(\epsilon)d\epsilon = 1$，并且 $q_\phi(z|x^{(i)})$ 和 $p(\epsilon)$ 之间存在一个变换关系。

那么，我们将改写 $\nabla_\phi \mathcal{L}(\phi)$:

$$\begin{aligned}
\nabla_\phi \mathcal{L}(\phi) =& \nabla_\phi \mathbb{E}_{q_\phi}\left[\log p_\theta(x^{(i)}, z) - \log q_\phi\right] \\
=& \nabla_\phi \int \left[\log p_\theta(x^{(i)}, z) - \log q_\phi\right] q_\phi dz \\
=& \nabla_\phi \int \left[\log p_\theta(x^{(i)}, z) - \log q_\phi\right] p(\epsilon) d\epsilon \\
=& \nabla_\phi \mathbb{E}_{p(\epsilon)}\left[\log p_\theta(x^{(i)}, z) - \log q_\phi\right] \\
=& \mathbb{E}_{p(\epsilon)} \nabla_\phi \left[(\log p_\theta(x^{(i)}, z) - \log q_\phi)\right] \\
=& \mathbb{E}_{p(\epsilon)} \nabla_z \left[(\log p_\theta(x^{(i)}, z) - \log q_\phi(z|x^{(i)}))\nabla_\phi z\right] \\
=& \mathbb{E}_{p(\epsilon)} \nabla_z \left[(\log p_\theta(x^{(i)}, z) - \log q_\phi(z|x^{(i)}))\nabla_\phi z\right] \\
=& \mathbb{E}_{p(\epsilon)} \nabla_z \left[(\log p_\theta(x^{(i)}, z) - \log q_\phi(z|x^{(i)}))\nabla_\phi g_\phi(\epsilon, x^{(i)})\right]
\end{aligned} \tag{12}$$

那么我们的问题就这样愉快的解决了，$p(\epsilon)$ 的采样与 ϕ 无关，然后对先求关于 z 的梯度，然后再求关于 ϕ 的梯度，那么这三者之间就互相隔离开了。最后，我们再对结果进行采样，$\epsilon^{(l)} \sim p(\epsilon)$，$l = 1, 2, \cdots, L$:

$$\nabla_\phi \mathcal{L}(\phi) \approx \frac{1}{L} \sum_{i=1}^{L} \nabla_z \left[(\log p_\theta(x^{(i)}, z) - \log q_\phi(z|x^{(i)}))\nabla_\phi g_\phi(\epsilon, x^{(i)})\right] \tag{13}$$

其中 $z \longleftarrow g_\phi(\epsilon^{(i)}, x^{(i)})$。而 SGVI 为:

$$\phi^{(t+1)} \longrightarrow \phi^{(t)} + \lambda^{(t)} \nabla_\phi \mathcal{L}(\phi) \tag{14}$$

4 小结

那么 SGVI，可以简要的表述为: 我们定义分布为 $q_\phi(Z|X)$，ϕ 为参数，参数的更新方法为:

$$\phi^{(t+1)} \longrightarrow \phi^{(t)} + \lambda^{(t)} \nabla_\phi \mathcal{L}(\phi) \tag{15}$$

$\nabla_\phi \mathcal{L}(\phi)$ 为：

$$\nabla_\phi \mathcal{L}(\phi) \approx \frac{1}{L} \sum_{i=1}^{L} \nabla_z \left[\log p_\theta(x^{(i)}, z) - \log q_\phi(z|x^{(i)})) \nabla_\phi g_\phi(\epsilon, x^{(i)}) \right] \tag{16}$$

Markov Chain Monte Carlo 01 Sampling Method

Chen Gong

30 December 2019

其实在之前的 Inference Variational 那一节中，我们讲到过一些有关于 Markov Chain Monte Carlo (MCMC) 的知识。也就是我们有一些数据 X，看到这些数据 X，并且有一些隐变量 Z，我们给隐变量一些先验，根据观测数据来推后验知识，也就是 $P(Z|X)$。

但是，很不幸的是 $P(Z|X)$ 的计算非常的复杂，我们大致采用两种思路来解决这个问题，也就是精确推断和近似推断。精确推断无法达到我们想要的结果时，就会采用近似推断的方法。而近似推断中我们又可以分成两大类，即为确定性近似 (VI) 和随机近似 (MCMC)。

Monte Carlo Method 是一种基于采样的随机近似算法。我们的目标是求解后验概率 $P(Z|X)$，其中 Z 为 Latent data，X 为 Observed data。知道分布以后，我们通常的目标是求解：

$$\mathbb{E}_{Z|X}[f(Z)] = \int_Z P(Z|X)f(Z)dZ \approx \frac{1}{N}\sum_{i=1}^{N} f(z_i) \tag{1}$$

然后，问题马上就来了，我们知道了后验分布 $P(Z|X)$，怎么去采样呢？也就是如何通过采样得到 $z^{(1)}, z^{(2)}, \cdots, z^{(N)} \sim P(Z|X)$。那么，我们这一节将要主要介绍三种采样方法，概率分布采样，拒绝采样和重要性采样。

1 概率分布采样

我第一次看到这个概念是在 Distributional Reinforcement Learning 中的 Wesserstein Metric 中。当时，真的把我看得我一脸懵逼，而且作者并没有提到概率分布采样。还有有的文章中，经常省写 c.d.f (概率分布函数)，p.d.f (概率密度函数)，i.i.d (独立同分布)。我觉得我这里有必要提一下。

为什么要有概率分布采样呢？因为我们直接根据概率分布来进行采样非常的复杂。如果我们知道概率分布的具体形式吗？我们可以直接求得概率累积的概率分布函数。由于概率分布函数的值一定是 $[0, 1]$ 之间的。所以，我们可以在均匀概率密度分布 $U(0, 1)$ 上采样，得到 $u^{(i)} \sim U(0, 1)$。然后求 $x^{(i)} \sim cdf^{-1}(u^{(i)})$ 就可以计算得到我们想要的结果。这样就可以采样得到 $\{x^{(1)}, x^{(2)}, \cdots, x^{(N)}\}$N 个样本点。

虽然，理论上这个方法好像很有效，但是实际上很多情况我们都根本不知道 p.d.f 的具体表现形式。就算知道，很多时候 c.d.f 也并不是那么的好求。所以很多情况下，概率分布采样并没有那么的好求。

2 拒绝采样 (Rejection Sampling)

由于对目标分布 $p(Z)$ 的采样非常的困难，所以我们可以对一个比较简单的分布 $q(Z)$ 进行采样来辅助采样。那么我们具体做法怎么办呢？我们可以设定一个 proposal distribution：$q(Z)$。对于 $\forall z_i$，保

证 $M \cdot q(z^i) \geq p(z^i)$，那么我们为什么要引入 M 呢？这是因为 $\int_Z P(Z)dZ = \int_Z q(Z)dZ = 1$。要使 $q(z^i) \geq p(z^i)$ 是几乎不可能成立的。为了方便描述，我们画图来说明一下：

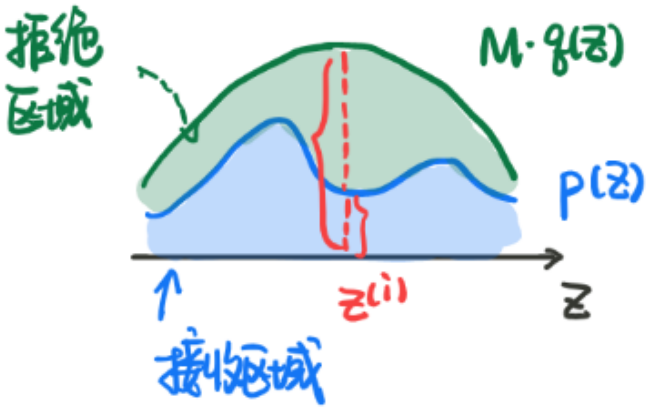

图 1: Rejection Sampling 示意图

在这里我们需要定义一个接受率：$\alpha = \frac{P(z^{(i)})}{M \cdot q(z^{(i)})}$，很显然 $0 \leq \alpha \leq 1$。这个实际就是上图中绿色的部分。

我们来看看具体的步骤：

(1) 首先进行采样 $z^{(i)} \sim q(z)$。

(2)$u \sim U(0,1)$；如果 $u \leq \alpha$，我们就接收 $z^{(i)}$，不然我们就拒绝。

所以，绿色的部分就被我们称为拒绝区域，就是这样来的，所以这个采样方法就是拒绝采样。同样这样的采样方法也有缺点。如果 $M \cdot q(z)$ 比 $p(z)$ 大很多的话，那么我们的采样老是是失败的，这就涉及到一个采样效率低下的问题。而当 $M \cdot q(z) = p(z)$ 的时候，$\alpha = 1$，我们每次采样的结果都是接受的。但是，实际上 $p(z)$ 的分布形式非常的复杂，我们根本就没有办法来得到那么准确的结果，特别是采样 cost 非常高的话，经常性的采样失败带来的损失是很大的。

3 重要性采样 (Importance Sampling)

重要性采样在我们的强化学习 (PPO) 中的应用非常的多。重要性采样并不是直接对概率进行采样，而是对概率分布的期望进行采样。也就是：

$$\begin{aligned} \mathbb{E}_{p(z)}[f(z)] = \int p(z)f(z)dz &= \int \frac{p(z)}{q(z)}q(z)f(z)dz \\ &= \int f(z)\frac{p(z)}{q(z)}q(z)dz \\ &\approx \frac{1}{N}\sum_{i=1}^{N} f(z_i)\frac{p(z_i)}{q(z_i)} \\ & z_i \sim q(z),\ i = 1, 2, \cdots, N \end{aligned} \tag{2}$$

而这里的 $\frac{p(z_i)}{q(z_i)}$ 也就是 Weight，用来平衡不同的概率密度值之间的差距。同样重要性采样也可能会出现一些问题，就是两个分布之间的差距太大了话，总是采样采不到重要的样本，采的可能都是实

际分布概率值小的部分。也就是采样效率不均匀的问题。在这个基础上，我们进一步提出了 Sampling Importance Resampling。

3.1 重要性重采样 (Sampling Importance Resampling)

经过重要性采样后，我们得到了 N 个样本点，以及对应的权重。那么我用权重来作为采样的概率，重新测采样出 N 个样本。也就是如下图所示：

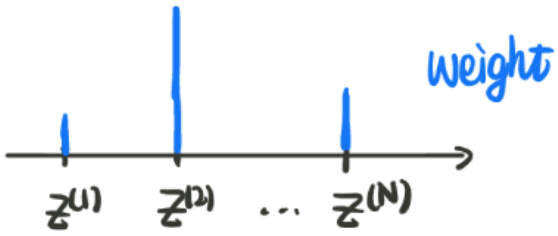

图 2: Sampling Importance Resampling 示意图

通过二次采样可以降低采样不平衡的问题。至于为什么呢？大家想一想，我在这里表达一下自己的看法。$\frac{p(z_i)}{q(z_i)}$ 是 Weight，如果 Weight 比较大的话，说明 $p(z_i)$ 比较大而 $q(z_i)$ 比较的小，也就是我们通过 $q(z_i)$ 采出来的数量比较少。那么我们按权重再来采一次，就可以增加采到重要性样本的概率，成功的弥补了重要性采样带来的缺陷，有效的弥补采样不均衡的问题。

Markov Chain Monte Carlo 02 Markov Chain

Chen Gong

31 December 2019

在上一小节中，我们描述了三种采样方法，也就是概率分布采样法，拒绝采样法和重要性采样法。这三种采样方法在高维情况下的采样效率很低，所以我们需要另找方法。

1 基础概念介绍

首先我们要明确什么是 Random Process，也就是它研究的变量是一个随机变量的序列 $\{x_t\}$。通俗的说就是，随机过程就是一个序列，而这个序列中的每一个元素都是一个随机变量。

而 Markov Chain 就是一个特殊的随机过程，它的时间和状态都是离散的。并且，Markov Chain 需要满足 Markov 性质，也就是未来和过去是无关的。我们用数学的语言表达就是：

$$P(x_{t+1} = x|x_1, x_2, \cdots, x_t) = P(x_{t+1}|x_1, x_2, \cdots, x_{t-m}) \tag{1}$$

上述公式就是一个 m 阶马尔可夫性质。当 $m = 0$ 时，我们就得到了齐次 (一阶) 马尔可夫链，也就是满足：

$$P(x_{t+1} = x|x_1, x_2, \cdots, x_t) = P(x_{t+1}|x_t) \tag{2}$$

而 $P(x_{t+1}|x_t)$ 这个概率我们用什么来表达呢?我们定义 $\mathcal{P}$ 为一个转移矩阵 $[P_{ij}]$,而 $P_{ij} = P(x_{t+1} = j|x_t = i)$。

2 平稳分布 (Stationary Distribution)

一个 Markov Chain 可以用下图来表示：

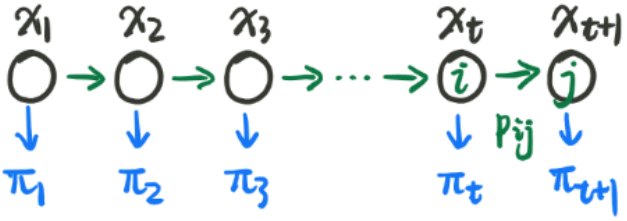

图 1: Markov Chain Model 示意图

此图就是一个时间序列，x_i 就表示在第 i 时刻的状态，而每一个状态都是一个随机变量。而 π_i 描述的就是第 i 个随机变量的分布。对于一个马氏链来讲，它在第 $t+1$ 时刻的概率分布，可以被我们表

达为：

$$\pi_{t+1}(x^*) = \int \pi_t(x) \cdot P(x \mapsto x^*) dx \tag{3}$$

熟悉强化学习的同学就会觉得这个公式非常的熟悉。通俗的讲，他实际上就是在 $t+1$ 时刻所有可能转移到状态 x^* 的概率的和。那么什么是随机分布呢?

假如这里存在一个 π，这里的 π 和前面的 π_t 和 π_{t+1} 都没有一毛钱关系。假如 π 是一个概率分布，那么它可以被我们写成一个无限维向量的形式：

$$\pi = [\pi(1), \pi(2), \cdots, \pi(t), \cdots], \qquad \sum_{i=1}^{\infty} \pi(i) = 1 \tag{4}$$

如果存在式 (4) 使得公式成立：

$$\pi(x^*) = \int \pi(x) \cdot P(x \mapsto x^*) dx \tag{5}$$

我们就称 $\{\pi(k)\}_{k=1}^{\infty}$ 是马氏链 $\{x_k\}$ 的平稳分布。看了数学的描述我相信大部分同学还是不懂这个平稳分布时是个什么东西？用通俗的话讲就是，对于一个马氏链来说，每个时刻都符合一个概率分布，如果每一个时刻的概率的分布都是一样的都是 $\pi(k)$，那么我们就可以称这个马氏链满足一个平稳分布。

那么下一个问题就是我们为什么要引入平稳分布呢？其实，我们想要去求的这个 $P(Z)$，可以被我们看成是一个平稳分布 $\pi(k)$，那么我们就可以通过构建一系列的 $\{x_1, x_2, \cdots, x_t, \cdots\}$ 的马氏链，让它来逼近这个平稳分布。那么我们构建这样的一个马氏链，包括随机变量和转移矩阵，如果它满足平稳分布的条件，确实是可以收敛到平稳分布的。那么，我就可以让构建出来的这个马氏链收敛到平稳分布来求得 $P(Z)$。

既然，已经知道了什么是平稳分布了，那么下一个问题就是，我们需要知道什么样的分布可以称为平稳分布，也就是我们怎样才能构建出一个马氏链让它收敛到一个平稳分布。这里我们需要引入一个条件，也就是 Detailed Balance：

$$\pi(x) \cdot P(x \mapsto x^*) = \pi(x^*) \cdot P(x^* \mapsto x) \tag{6}$$

大家直观的来想想这个公式，为什么满足它就满足了是一个平稳分布呢？其实并不难想到，对于任意两个状态之间，使用概率分布称为转移概率得到的结果都是可逆的，那么这两个状态之间的分布一定是一样的。而说如果一个分布，满足 Detailed Balance 那么它一定可以是一个平稳分布，但是反过来并不能成立。而证明过程也不难，如下所示：

$$\begin{aligned} \int \pi(x) \cdot P(x \mapsto x^*) dx &= \int \pi(x^*) \cdot P(x^* \mapsto x) dx \\ &= \pi(x^*) \underbrace{\int P(x^* \mapsto x)\, dx}_{\sum_{j=1}^{\infty} P_{ij} = 1} \\ &= \pi(x^*) \end{aligned} \tag{7}$$

这样的话，我们就可以吧不用从定义上来证明一个随机过程是马尔可夫链，直接看它满不满足 Detailed Balance 就可以了。而且这个公式中，π 是平稳分布，$P(x \mapsto x^*)$ 是马尔可夫链的状态转移概率，这样就成功的将平稳分布和马尔可夫链结合在了一起。

Markov Chain Monte Carlo 03 Metropolis Hastings Sampling

Chen Gong

01 January 2020

上一节中我们讲解了 Detailed Balance，这是平稳分布的充分必要条件。Detailed Balance 为：

$$\pi(x)P(x \mapsto x^*) = \pi(x^*)P(x^* \mapsto x) \tag{1}$$

这里的 $P(x \mapsto x^*)$ 实际上就是条件概率 $P(z^*|x^*)$，这样写只是便于理解。

首先，我们需要明确一点，我们想要求的是后验概率分布 $P(Z)$，也就是我们推断问题的核心目标。我们求 $P(Z)$ 主要是为了求在 $P(Z)$ 概率分布下的一个相关函数的期望，也就是：

$$\mathbb{E}_{P(Z)}[f(Z)] \approx \frac{1}{N}\sum_{i=1}^{N} f(z^{(i)}) \tag{2}$$

而我们是通过采样来得到 $P(Z) \sim \{z^{(1)}, z^{(2)}, \cdots, z^{(N)}\}$ 样本点。$\pi(x)$ 是最终的平稳分布，可以看成我们这里的 $P(Z)$，下面的问题就是求出概率转移矩阵 P_{ij}，才能满足 Detailed Balance 条件。知道了上面的条件以后，我们每次这样进行采样，$x_1 \sim P(x|x_1)$，$x_2 \sim P(x|x_1)$，$x_3 \sim P(x|x_2)$，$\cdots$，x_N。最终就可以得到我们想要的 N 个样本。

1 Proposal Matrix

那我们怎么来找这个状态转移矩阵 P_{ij} 呢？首先我们可以随机一个状态转移矩阵 Q_{ij}，也就是 Proposal Matrix。

那么肯定是：

$$P(Z)Q(Z^*|Z) \neq P(Z^*)Q(Z|Z^*) \tag{3}$$

那么我们就要想办法找到 Q_{ij} 使得：

$$P(Z)Q(Z^*|Z) = P(Z^*)Q(Z|Z^*) \tag{4}$$

那么，我们怎么来解决这个问题呢？我们可以在左右两边乘上一个因子来解决这个问题。也就是，

$$P(Z)\underbrace{Q(Z^*|Z)\alpha(Z^*, Z)}_{P(Z \mapsto Z^*)} = P(Z^*)\underbrace{Q(Z|Z^*)\alpha(Z, Z^*)}_{P(Z^* \mapsto Z)} \tag{5}$$

而 $\alpha(Z, Z^*)$ 定义为接收率，大小为：

$$\alpha(Z, Z^*) = \min\left(1, \frac{P(Z^*)Q(Z|Z^*)}{P(Z)Q(Z^*|Z)}\right) \tag{6}$$

这样定义就行了？就可以满足 Detailed Balance 吗？我们可以证明一下，

$$\begin{aligned} P(Z)Q(Z^*|Z)\alpha(Z,Z^*) =& P(Z)Q(Z^*|Z)\min\left(1, \frac{P(Z^*)Q(Z|Z^*)}{P(Z)Q(Z^*|Z)}\right) \\ =& \min(P(Z)Q(Z^*|Z), P(Z^*)Q(Z|Z^*)) \\ =& P(Z^*)Q(Z|Z^*)\min\left(\frac{P(Z)Q(Z^*|Z)}{P(Z^*)Q(Z|Z^*)}, 1\right) \\ =& P(Z^*)Q(Z|Z^*)\alpha(Z^*,Z) \end{aligned} \tag{7}$$

那么我们就成功的证明了：

$$P(Z)\underbrace{Q(Z^*|Z)\alpha(Z^*,Z)}_{P(Z\mapsto Z^*)} = P(Z^*)\underbrace{Q(Z|Z^*)\alpha(Z,Z^*)}_{P(Z^*\mapsto Z)} \tag{8}$$

所以，$P(Z)$ 在转移矩阵 $Q(Z|Z^*)\alpha(Z^*,Z)$ 下是一个平稳分布，也就是一个马尔可夫链，通过在这个马尔可夫链中采样就可以得到我们的相应的数据样本点了。实际上这就是大名鼎鼎的 Metropolis-Hastings 采样法。

2 Metropolis-Hastings Sampling

第一步，我们从一个均匀分布中进行采样，$u \sim U(0,1)$；

第二步，从 $Q(Z|Z^{(i-1)})$ 中进行采样得到样本点 Z^*；

第三步，计算接受率，$\alpha = \min\left(1, \frac{P(Z^*)Q(Z|Z^*)}{P(Z)Q(Z^*|Z)}\right)$。注意，这里的 $P(Z) = \frac{\hat{P}(Z)}{Z_p}$。其中 Z_p 指的是归一化因子，几乎非常难以计算，所以一般是未知的。而 $\hat{P}(Z) = \text{likelihood} \times \text{prior}$。所以这里的 $P(Z)$ 和 $P(Z^*)$ 就是 $\hat{P}(Z)$ 和 $\hat{P}(Z^*)$。由于归一化因子被抵消了，干脆就直接写成了 $P(Z)$ 和 $P(Z^*)$。

第四步，如果 $u \leq \alpha$ 时 $Z^i = Z^*$，不然 $Z^i = Z^{(i-1)}$。

这样执行了 N 次，就可以得到 $\{Z^{(1)}, Z^{(2)}, \cdots, Z^{(N)}\}$ 个样本点。

Markov Chain Monte Carlo 04 Gibbs Sampling

Chen Gong

02 January 2020

如果我们要向一个高维的分布 $P(Z) = P(Z_1, Z_2, \cdots, Z_N)$ 中进行采样。那么我们怎么来进行采样呢？我们的思想就是一维一维的来，在对每一维进行采样的时候固定住其他的维度，这就是 Gibbs Sampling。

我们首先规定一个 z_{-i} 是去除 z_i 后的序列，$\{z_1, z_2, \cdots, z_{i-1}, z_{i+1}, \cdots, z_N\}$。

1 A Example

假设 t 时刻，我们获得的样本为 $z_1^{(t)}, z_2^{(t)}, z_3^{(t)}$。

那么 $t+1$ 时刻，我们的采样顺序为：

$$\begin{aligned} z_1^{(t+1)} &\sim P(z_1|z_2^{(t)}, z_3^{(t)}) \\ z_2^{(t+1)} &\sim P(z_2|z_1^{(t+1)}, z_3^{(t)}) \\ z_3^{(t+1)} &\sim P(z_3|z_1^{(t+1)}, z_2^{(t+1)}) \end{aligned} \tag{1}$$

从这个例子中，我们应该可以大致理解固定其他的维度然后进行一维一维采样的意思了。而实际上 Gibbs 是一种特殊的 MH 采样，为什么呢？我们来证明一下。

2 接受率 α 的计算

我们首先回顾一下，MH 采样的方法。我们的目的是从 $Q(Z|Z^{(t)})$ 中采样获得 Z^*，然后计算接受率

$$\alpha = \min\left(1, \frac{P(Z^*)Q(Z|Z^*)}{P(Z)Q(Z^*|Z)}\right) \tag{2}$$

首先我们来看 $Q(Z|Z^{(t)})$：

$$Q(Z|Z^{(t)}) = Q(Z_i, Z_{-i}|Z_i^{(t)}, Z_{-i}^{(t)}) \tag{3}$$

假设我们现在是在对第 i 维进行采样，我们只要关注 $P(Z_i^*|Z_{-i})$。所以，我们可以得到：$Q(Z|Z^{(t)}) = P(Z_i^*|Z_{-i}^{(t)})$。

已经成功的将 $Q(Z|Z^{(t)})$ 做了等价转换以后。那么我们想要求的 α 可以被我们成功的转换成如下的形式：

$$\alpha = \min\left(1, \frac{P(Z_i^*|Z_{-i}^*)P(Z_{-i}^*)P(Z_i|Z_{-i}^*)}{P(Z_i|Z_{-i})P(Z_{-i})P(Z_i^*|Z_{-i})}\right) \tag{4}$$

计算到了这里，我们还是不好进行计算，上面和下面好像还是不好消除。如果我们可以得到 Z_{-i}^{*} 和 Z_{-i} 之间的关系就好了。下面我们会得出一个重要的结论来帮助我们计算 α 的具体值。首先我们来举一个例子：

那么假设当 $t=1$ 的时刻，有一个样本为：$Z_1^{(1)}, Z_2^{(1)}, Z_3^{(1)}$。

当 $t=2$ 的时刻，我们假设先对第一维进行采样就可以得到：$Z_1^{(2)}, Z_2^{(1)}, Z_3^{(1)}$。

很显然 $Z_2^{(1)}, Z_3^{(1)}$ 根本没有发生变化。我们可以得到 $Z_{-1} = Z_{-1}^{*}$。也就是在 Gibbs 采样时，采样前后只关注于一个维度，其他的维度我们都没有关注到。所以就可以得到结论：

$$Z_{-i} = Z_{-i}^{*} \tag{5}$$

那么，我们把这个结论代入到公式 (4) 中，就可以得到：

$$\alpha = \min\left(1, \frac{P(Z_i^{*}|Z_{-i}^{*})P(Z_{-i}^{*})P(Z_i|Z_{-i}^{*})}{P(Z_i|Z_{-i}^{*})P(Z_{-i}^{*})P(Z_i^{*}|Z_{-i}^{*})}\right) = 1 \tag{6}$$

那么计算出接受率为 1，也就是每次都必定被接受。所以，每次从 $Q(Z|Z^{(t)}) = P(Z_i^{*}|Z_{-i})$ 中进行采样得到 Z_i^{*} 即可，一维一维的进行采样就可以采到整个高维的分布，各个维度上的样本。

所以，解释到了这里，大家基本就可以知道 Gibbs Samplings 是 $\alpha = 1$ 的 MH Sampling 的意义了。在 Gibbs Sampling 中 $\alpha = 1$，而且状态转移矩阵 $Q(Z|Z^{(t)}) = P(Z_i^{*}|Z_{-i}^{(t)})$，所以 Gibbs Sampling 就是把目标分布 P 对应的条件概率当作状态转移分布 Q。

这里我们需要额外提醒一下，使用 Gibbs Sampling 是有使用前提的，也就是固定其他维度后的一维分布时方便进行采样的，如果固定其他维度的时候得到的一维分布仍然是非常难进行采样的，那么使用 Gibbs Sampling 也是没有用的。

Markov Chain Monte Carlo 05 Sampling

Cheh Gong

03 January 2020

在前面的章节中，我们已经基本介绍了 Markov Chain Monte Carlo Sampling 的基本概念，基本思路和主要方法。那么这一小节中，我们将主要来介绍一下，什么是采样？我们为什么而采样？什么样的样本是好的样本？以及我们采样中主要会遇到哪些困难？

1 采样的动机

这一小节的目的就是我们要知道什么是采样的动机，我们为什么而采样？

1. 首先第一点很简单，采样本身就是发出常见的任务，我们机器学习中经常需要进行采样来完成各种各样的任务。如果从一个 $P(X)$ 中采出一堆样本。

2. 求和求积分。包括大名鼎鼎的 Monte Carlo 算法。我们求 $P(X)$ 主要是为了求在 $P(X)$ 概率分布下的一个相关函数的期望，也就是：

$$\int P(x)f(x)dx = \mathbb{E}_{P(X)}[f(X)] \approx \frac{1}{N}\sum_{i=1}^{N} f(x^{(i)}) \tag{1}$$

而我们是通过采样来得到 $P(X) \sim \{x^{(1)}, x^{(2)}, \cdots, x^{(N)}\}$ 样本点。

2 什么样的是好样本

既然，我们知道了采样的目的和动机，下一个问题就自然是，同样是采样，什么样的样本就是好样本呢？或者说是采样的效率更高一些。

1. 首先样本的分布肯定是要趋向于原始的目标分布吧，也就是说样本要趋向于高概率选择区域。或者是说，采出来的样本出现的概率和实际的目标分布的概率保持一致。

2. 样本和样本之间是相互独立的。这个就没有那么直观了。大家想一想就知道了，如果我采出来的一堆样本之间都差不多，那么就算采出来了趋向于高概率选择区域的样本，那采样效率太低了，样本中反映的信息量太有限了。

3 实际采样中的困难

实际采样中，采样时困难的，为什么呢？我们这里主要介绍两点：

1. **Partation function is intractable.** 我们的后验分布往往被写成 $P(X) = \frac{1}{Z}\hat{P}(X)$，上面这个 $\hat{P}(X)$ 都比较好求，就是等于 Likelihood $\times$ Prior。而 Z 就是我们要求的归一化常数，它非常的难

以计算，$Z = \int \hat{P}(X)dX$，这几乎就是不可计算的。所以，有很多采样方法就是想要跳过求 $P(X)$ 的过程，来从一个近似的分布中进行采样，当然这个近似的分布采样要比原分布简单。比如：Rejection Sampling 和 Importance Sampling。

2. **The curse of high dimension**. 如果样本空间 $\mathcal{X} \in \mathbb{R}^p$，每个维度都有 K 个状态的话。那么总的样本空间就有 K^p 的状态。要知道那个状态的概率高，就必须要遍历整个样本空间，不然就不知道哪个样本的概率高，如果状态的数量是这样指数型增长的话，全看一遍之后进行采样时不可能的。所以，直接采样的方法是不可行的。

4 采样方法

Rejection Sampling 和 Importance Sampling，都是借助一个 $Q(x)$ 去逼近目标分布 $P(x)$，通过从 $Q(x)$ 中进行采样来达到在 $P(x)$ 中采样的目的，而且在 $Q(x)$ 中采样比较简单。当时如果 $Q(x)$ 和 $P(x)$ 直接的差距太大的话，采样效率会变得很低。

而 MCMC 方法，我们主要介绍了 MH Sampling 和 Gibbs Sampling，我们主要是通过构建一个马氏链去逼近目标分布，具体的描述将在下一节中展开描述。

Markov Chain Monte Carlo 06 Method of MCMC

Chen Gong

04 January 2020

这一小节主要是对前面的补充，希望可以详细的介绍一下 MCMC 原理，将前面的知识点可以顺利的串起来。MCMC 采样中，我们借助了一条马氏链，马氏链的性质，经过若干步以后会收敛到一个平稳分布。马尔可夫链的组成可以大致分成两个部分：

1. 状态空间：$\{1, 2, 3, \cdots, k\}$；
2. 状态转移空间 $Q = [Q_{ij}]_{k\times k}$。

马尔可夫链的模型可以被我们表达为：

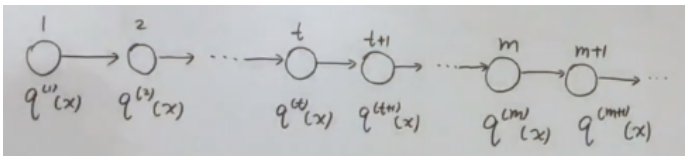

图 1: 马尔可夫链模型抽象图

每一个时间点有一个状态分布，表示当前时间点位于某个状态的概率分布情况，我们表示为 $q^{(t)}(x)$。如果，是在 $t = 1$ 的时间节点，状态的概率分布为 $q^{(1)}(x)$，我们可以用下列表来描述：

x	1	2	3	$\cdots$	k
$q^{(1)}(x)$	q_1^1	q_1^2	q_1^3	$\cdots$	q_1^k

我们假设在 $t = m$ 时刻之后到达了平稳分布状态，那么我们就可以得到：$q^{(m)} = q^{(m+1)} = q^{(m+2)}$。这时的平稳分布就是我们想要的目标分布。相邻时间节点之间的状态转移矩阵为：

$$Q = \begin{bmatrix} Q_{11} & Q_{12} & \cdots & Q_{1k} \\ Q_{21} & Q_{22} & \cdots & Q_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ Q_{k1} & Q_{k2} & \cdots & Q_{kk} \end{bmatrix}_{k\times k} \tag{1}$$

状态转移矩阵描述的是，$Q_{ij} = Q(x^2 = j|x^1 = i)$。描述的是从一个状态转移到另外一个状态的概率。所以，状态转移矩阵的每一行 i 表示为目前状态是 i 时，到其他状态的概率，那么必然有 $\sum_{k=1}^{k} Q_{ik} = 1$。实际上了解强化学习的同学，对于这些概率应该是非常的熟练了，这些都是强化学习的基础。

1 Markov Chain 收敛性介绍

在这一小节中，我们将详细的介绍一下，Markov Chain 中状态转移的过程。并将证明在 Markov Chain 随着迭代一定会收敛到一个平稳分布。

1.1 Markov Chain 状态转移计算

假设在 $t+1$ 时刻，状态是 $x=j$，那么它的分布为所有可能转移到这个状态的概率 i 乘以这个状态的分布 $q^{(t)}(x=i)$，我们用公式表达就是：

$$q^{(t+1)}(x=j)=\sum_{i=1}^{k}q^{(t)}(x=i)Q_{ij} \tag{2}$$

那么，这仅仅是当 $x=j$ 时概率，实际上在 $t+1$ 时刻，可能出现的状态有 k 个，那么 q^{t+1} 的分布，就是将转移到各个状态的概率分别计算出来，也就是如下所示：

$$q^{(t+1)}=\begin{bmatrix}q^{(t+1)}(x=1) & q^{(t+1)}(x=2) & q^{(t+1)}(x=3) & \cdots & q^{(t+1)}(x=k)\end{bmatrix}_{1\times k} \tag{3}$$

而，

$$q^{(t+1)}(x=j)=\sum_{i=1}^{k}q^{(t)}(x=i)Q_{ij} \tag{4}$$

那么，$q^{(t+1)}$ 可以被我们表示为：

$$\begin{aligned}q^{(t+1)}=&\begin{bmatrix}\sum_{i=1}^{k}q^{(t)}(x=i)Q_{i1} & \sum_{i=1}^{k}q^{(t)}(x=i)Q_{i2} & \cdots & \sum_{i=1}^{k}q^{(t)}(x=i)Q_{ik}\end{bmatrix}_{1\times k}\\ =&q^{(t)}\cdot Q\end{aligned} \tag{5}$$

其中，$q^{(t)}=\begin{bmatrix}q^{(t)}(x=1) & q^{(t)}(x=2) & q^{(t)}(x=3) & \cdots & q^{(t)}(x=k)\end{bmatrix}_{1\times k}$。那么，通过这个递推公式，我们可以得到，$q^{(t+1)}=q^{(t)}Q=q^{(t-1)}Q^2=\cdots=q^{(1)}Q^t$。通过上述的描述，详细大家都已经详细的了解了 Markov Chain 中，每个时刻点的状态的分布 $q^{(t)}$ 的计算方法。既然我们知道了每个时间点的概率分布的计算方法，下一个问题就是我们怎么可以知道一定是收敛的呢?

1.2 Markov Chain 收敛性

由于 Q 是一个随机概率矩阵，那么我们可以得到，每个值都是小于 1 的，所以也必然有特征值的绝对值 ≤ 1。为什么呢? 我们可以从特征值的几何意义上好好的想一想，特征值代表变换中方向不变的向量的变化尺度。随机矩阵的变化尺度必然是小于 1 的。所以，我们可以对概率转移矩阵做特征值分解，分解成对角矩阵：

$$Q=A\Lambda A^{-1}\qquad \Lambda=\begin{bmatrix}\lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_k\end{bmatrix},\qquad |\lambda_i|\leq 1\qquad (i=1,2,\cdots,k) \tag{6}$$

我们假设只有一个 $\lambda_i=1$，则：

$$q^{(t+1)}=q^{(1)}(A\Lambda A^{-1})^t=q^{(1)}A\Lambda^t A^{-1} \tag{7}$$

当 $t \to \infty$ 时，必然有：

$$\Lambda^t = \begin{bmatrix} 0 & & & \\ & 1 & & \\ & & \ddots & \\ & & & 0 \end{bmatrix} \tag{8}$$

我们可以假设存在足够大的 M：

$$s.t. \quad \Lambda^M = \begin{bmatrix} 0 & & & \\ & 1 & & \\ & & \ddots & \\ & & & 0 \end{bmatrix} \tag{9}$$

所以，

$$\begin{aligned} q^{(m+1)} =& q^{(1)} A \Lambda^m A^{-1} \\ q^{(m+2)} =& q^{(m+1)} A \Lambda A^{-1} \\ =& q^{(1)} A \Lambda^m A^{-1} A \Lambda A^{-1} \\ =& q^{(1)} A \Lambda^{(m+1)} A^{-1} \\ =& q^{(m+1)} \end{aligned} \tag{10}$$

通过上述的证明，我们成功的证明了 $q^{(m+2)} = q^{(m+1)}$。我们用数学的语言来表述一下，也就是当 $t > m$ 时，$q^{(m+1)} = q^{(m+2)} = \cdots = q^{(\infty)}$。这就是平稳分布，我们成功的证明了 Markov Chain 经过足够大的步数 m 之后，一定会收敛到一个平稳分布。于是，这就启发了我们设计一个 Markov Chain，收敛到我们想要采样的分布 $p(x)$。那么。怎么我们才能让它收敛呢？实际上就是由状态转移矩阵 Q 所决定的。我们的核心问题就是设计一个合适的状态转移矩阵 Q。

那么，我们要做的就是设计一个 MCMC，利用 Markov Chain 收敛到一个平稳分布 $q(x)$，使得平稳分布 $\approx$ 目标分布 $p(x)$。也就是当 m 足够大的时候，$q^{(m)}(x) = q^{(m+1)}(x) = q^{(m+2)}(x) = q(x)$。

那么，我们的 Markov Chain 解决了当维度很高的时候，$q(x) \approx p(x)$ 找不到的情况，在 MCMC 中不要显示的去找，而是构建一个 Markov Chain 去近似，跳过了直接去寻找的过程。

这里我们介绍一个概念，也就是从开始到收敛到 m 的这段时期被我们称为 bum-in，中文翻译为燃烧期 (个人觉得非常的难听，所以我从来不用中文的表述形式)。也有说法称这个时间 t 为 Mix-time。当然也不是任何的分布都可以用 MCMC 来进行采样。但是它可以有效的避免我们去寻找 $q(z)$。下面我们将描述一些用 MCMC 采样时遇到的困难的地方。

2 Existing Problem

1. 虽然，我们可以证明出 MCMC 最终可以收敛到一个平稳分布。但是并没有理论来判断 Markov Chain 是否进入了平稳分布，也就是不知道 Markov Chain 什么时候会收敛。

2. Mixing Time 过长，这就是有高维所造成的，维度和维度之间的相关性太强了，$p(x)$ 太过于复杂了。理论上 MCMC 是可以收敛的，但是如果 m 如果实在是太大的话，我们基本就是认为它是不收敛的。实际上，现在有各种各样的 MCMC 算法的变种都是在想办法解决这个 Mixing Time 过长的问题。

3. 我们希望采到的样本之间的样本相互独立，也就是采到的样本之间的相关性越小越好。

这个有关于样本之间独立性的问题,大家可能不太好理解,这是实际在高维分布中我们采用 MCMC 来进行采样很有可能造成样本单一，相关性太强的问题。我们我们来举一个 Mixture Gaussian Distribution 的例子。下图所示是一个 Mixture Gaussian Distribution 的例子：

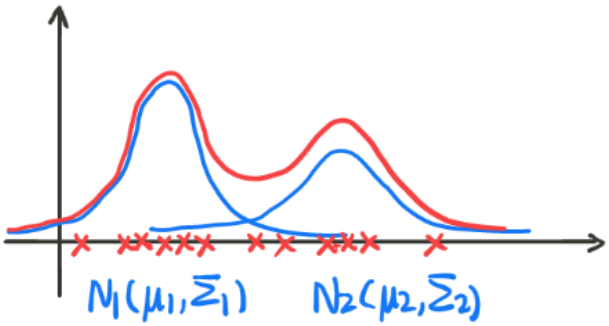

图 2: Mixture Gaussian Distribution 举例

会有一个什么问题呢？就是样本都趋向于一个峰值附近，很有可能会过不了低谷，导致样本都聚集在一个峰值附近。这个问题出现的原因我们可以从能量的角度来解释这个问题。在无向图中，我们常用下列公式来进行表示：

$$P(X) = \frac{1}{Z}\hat{P}(X) = \frac{1}{Z}exp^{-\mathbb{E}(X)} \tag{11}$$

实际上这里的 $\mathbb{E}(X)$ 指的就是能量函数，能量和概率是成反比的，概率越大意味着能量越低，能量越低，越难发生跳跃的现象。所以，采样很容易陷入到一个峰值附近。并且，多峰还可以分为均匀和陡峭，陡峭的情况中，能量差实在是太大了，就很难发生跳跃。就像孙悟空翻出如来佛祖的五指山一样，佛祖的维度很好，孙悟空在翻跟头的时候，一直在一个低维里面不同的打转，根本就跳不出来，就是来自佛祖的降维打击。

所以，在高维情况下，很容易发生在一个峰值附近不停的采样，根本就跳不出来，导致采到的样本的多样性低，样本之间的关联性大，独立性低。

Hidden Markov Model 01 Background

Chen Gong

07 January 2020

机器学习大致可以分为两个派别，也就是频率派和贝叶斯派的方法，这个之前，我们都有过详细的说明。这里再大致的回顾一下。

频率派的思想就衍生出了统计学习方法，说白了统计学习方法的重点在于优化，找 loss function。频率派的方法可以分成三步，1. 定义 Model，比如 $f(w) = w^T x + b$；2. 寻找策略 strategy，也就是定义 Loss function；3. 求解，也就是优化的方法，比如梯度下降 (GD)，随机梯度下降 (SGD)，牛顿法，拟牛顿法等等。

贝叶斯派的思想也就衍生出了概率图模型。概率图模型重点研究的是一个 Inference 的问题，我们要求的是一个后验概率分布 $P(Z|X)$，其中 X 为观测变量，Z 为隐变量。实际上就是一个积分问题，为什么呢？因为贝叶斯框架中的归一化因子需要对整个状态空间进行积分，非常的复杂。代表性的有前面讲到的 MCMC，MCMC 的提出才是彻底的把贝叶斯理论代入到实际的运用中。

1 概率图模型回顾

概率图模型，如果不考虑时序的关系，我们可以大致的分为：有向图的 Bayesian Network 和无向图的 Markov Random Field (Markov Network)。这样，我们根据分布获得的样本之间都是 iid (独立同分布) 的。比如 Gaussian Mixture Model (GMM)，我们从 $P(X|\theta)$ 的分布中采出 N 个样本 $\{x_1, x_2, \cdots, x_n\}$。N 个样本之间都是独立同分布的。也就是对于隐变量 Z，观测变量 X 之间，我们可以假设 $P(X|Z) = \mathcal{N}(\mu, \Sigma)$，这样就可以引入我们的先验信息，从而简化 X 的复杂分布。

如果引入了时间的信息，也就是 x_i 之间不再是 iid 的了，我们称之为 Dynamic Model。模型如下所示：

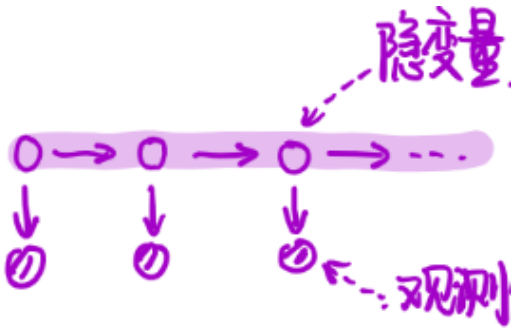

图 1: Dynamic Model 拓扑结构图

Dynamic Model 可以从两个层面来看，横着看就是 time 的角度，如果是竖着看就可以表达为 $P(X|Z)$ 的形式，也就是 Mixture 的形式。概率系统根据状态与状态之间的关系，可以分为两类。

如果是离散的则有 HMM 算法。

如果是连续的，按照线性和非线性可以分为 Kalman Filter 和 Paricle Filter。

2 HMM 算法简介

Hidden Markov Model 的拓扑结构图如下所示：

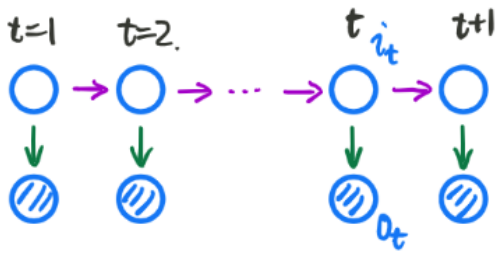

图 2: Hidden Markov Model 拓扑结构图

大家看到这个模型就会觉得和上一讲提到的，MCMC 模型方法有点类似。HMM 可以看做一个三元组 $\lambda = (\pi, \mathcal{A}, \mathcal{B})$。其中：

π：是初始概率分布。

$\mathcal{A}$：状态转移矩阵。

$\mathcal{B}$：发射矩阵。

拓扑结构图的第二行为观测变量，观测变量 o：$o_1, o_2, \cdots, o_t, \cdots \leftarrow \mathcal{V} = v_1, v_2, \cdots, v_M$。其中 $\mathcal{V}$ 是观察变量 o 的值域，代表每一个观测变量 o_i 可能有 M 个状态。

拓扑结构图的第一行为状态变量，状态变量 i：$i_1, i_2, \cdots, i_t, \cdots \leftarrow \mathcal{Q} = q_1, q_2, \cdots, q_N$。其中 $\mathcal{Q}$ 是状态变量 i 的值域，代表每一个状态变量 i 可能有 N 个状态。

$\mathcal{A} = [a_{ij}]$ 表示状态转移矩阵，$a_{ij} = P(i_{i+1} = q_j | i_t = q_i)$。

$\mathcal{B} = [b_j(k)]$ 表示发射矩阵，$b_j(k) = P(o_t = V_k | i_t = q_j)$。

而 π 是什么意思呢？假设当 t 时刻的隐变量 i_t，可能有 $\{q_1, q_2, \cdots, q_N\}$ 个状态，而这些状态出现的概率分别为 $\{p_1, p_2, \cdots, p_N\}$。这就是一个关于 i_t 隐变量的离散随机分布。

$\mathcal{A}$ 表示为各个状态转移之间的概率。

$\mathcal{B}$ 表示为观测变量和隐变量之间的关系。

2.1 两个假设

这是有关 Hidden Markov Model 的两个假设：

1. 齐次 Markov 假设 (无后向性)；2. 观察独立假设。

齐次马尔可夫假设：未来与过去无关，只依赖与当前的状态。也就是：

$$P(i_{t+1} | i_t, i_{t-1}, \cdots, i_1, o_t, \cdots, o_1) = P(i_{t+1} | i_t) \tag{1}$$

2. **观测独立假设：**

$$P(o_t|i_t, i_{t-1}, \cdots, i_1, o_t, \cdots, o_1) = P(o_t|i_t) \tag{2}$$

2.2 三个问题

1. Evaluation 的问题，我们要求的问题就是 $P(O|\lambda)$。也就是前向后向算法，给定一个模型 λ，求出观测变量的概率分布。

2. Learning 的问题，λ 如何求的问题。也就是 $\lambda_{MLE} = \arg\max_\lambda P(O|\lambda)$。求解的方法是 EM 算法和 Baum Welch 算法。

3. Decoding 的问题，状态序列为 I，也就是隐变量序列，$\hat{I} = \arg\max_I P(I|O, \lambda)$。也就是在在观测变量 O 和 λ 的情况下使隐变量序列 I 出现的概率最大。而这个问题大致被分为预测和滤波。

预测问题为：$P(i_{t+1}|o_1, \cdots, o_t)$；也就是在已知当前观测变量的情况下预测下一个状态，也就是 Viterbi 算法。

滤波问题为：$P(i_t|o_1, \cdots, o_t)$；也就是求 t 时刻的隐变量。

Hidden Markov Model，可以被我们总结成一个模型 $\lambda = (\pi, \mathcal{A}, \mathcal{B})$，两个假设，三个问题。而其中我们关注得最多的就是 Decoding 的问题。

Hidden Markov Model 02 Evaluation

Chen Gong

08 January 2020

Evaluation 的问题可以被我们描述为：给定一个 λ，如何求得 $P(O|\lambda)$。也就是在给定模型 λ 的情况下，求某个观测序列出现的概率。

1 模型求解

对于 $P(O|\lambda)$ 我们利用概率的基础知识进行化简可以得到：

$$P(O|\lambda) = \sum_I P(O, I|\lambda) = \sum_I P(O|I, \lambda)P(I|\lambda) \tag{1}$$

其中 $\sum_I$ 表示所有可能出现的隐状态序列；$\sum_I P(O|I, \lambda)$ 表示在某个隐状态下，产生某个观测序列的概率；$P(I|\lambda)$ 表示某个隐状态出现的概率。

那么：

$$\begin{aligned} P(I|\lambda) =& P(i_1, \cdots, i_T|\lambda) \\ =& P(i_T|i_1, \cdots, i_{T-1}, \lambda) \cdot P(i_1, \cdots, i_{T-1}|\lambda) \end{aligned} \tag{2}$$

根据 Hidden Markov Model 两个假设中的，齐次马尔可夫假设，我们可以得到：$P(i_T|i_1, \cdots, i_{T-1}, \lambda) = P(i_T|i_{T-1}) = a_{i_{T-1}, i_T}$。后面按照一样的思路进行迭代就可以了。那么我们继续对公式 (2) 进行化简可以得到：

$$\begin{aligned} P(i_T|i_1, \cdots, i_{T-1}, \lambda) \cdot P(i_1, \cdots, i_{T-1}|\lambda) =& P(i_T|i_{T-1}) \cdot P(i_1, \cdots, i_{T-1}|\lambda) \\ =& a_{i_{T-1}, i_T} \cdot a_{i_{T-2}, i_{T-1}} \cdots a_{i_1, i_2} \cdot \pi(a_{i_1}) \\ =& \pi(a_{i_1}) \prod_{t=2}^{T} a_{i_{t-1}, i_t} \end{aligned} \tag{3}$$

然后，运用观察独立假设，我们可以知道：

$$\begin{aligned} P(O|I, \lambda) =& P(o_1, o_2, \cdots, o_T|I, \lambda) \\ =& \prod_{t=1}^{T} P(o_t|I, \lambda) \\ =& \prod_{t=1}^{T} b_{i_t}(o_t) \end{aligned} \tag{4}$$

那么，结合公式 (2-5)，我们可以得到：

$$\begin{aligned} P(O|\lambda) &= \sum_{I} \pi(a_{i_1}) \prod_{t=2}^{T} a_{i_{t-1},i_t} \prod_{t=1}^{T} b_{i_t}(o_t) \\ &= \sum_{i_1} \cdot \sum_{i_2} \cdots \sum_{i_T} \pi(a_{i_1}) \prod_{t=2}^{T} a_{i_{t-1},i_t} \prod_{t=1}^{T} b_{i_t}(o_t) \end{aligned} \tag{5}$$

因为一共有 T 个状态，每个状态有 N 种可能，所以算法复杂度为 $\mathcal{O}(N^T)$。既然这样直接求太困难了，我们就需要另外想办法。

2 Forward Algorithm

下面，我们首先展示一下 Hidden Markov Model 的拓扑结构图。

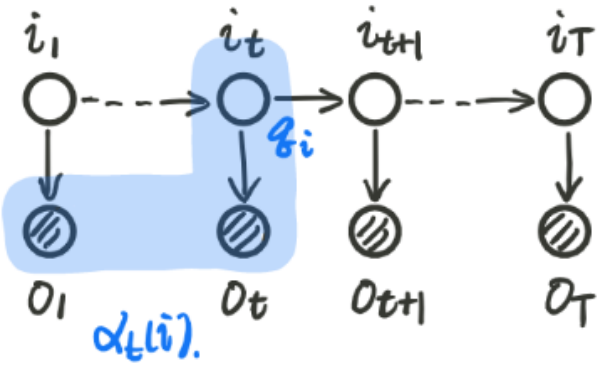

图 1: 矩阵与列向量的乘法

我们记，$\alpha_t(i) = P(o_1, \cdots, o_t, i_t = q_i|\lambda)$，这个公式表示的是在之前所有的观测变量的前提下求出当前时刻的隐变量的概率。那么：

$$P(O|\lambda) = \sum_{i=1}^{N} P(O, i_t = q_i|\lambda) = \sum_{i=1}^{N} \alpha_t(i) \tag{6}$$

其中，$\sum_{i=1}^{N}$ 表示对所有可能出现的隐状态情形求和，而 $\alpha_t(i)$ 表示对所有可能出现的隐状态情形求和。我们的想法自然就是寻找 $\alpha_t(i)$ 和 $\alpha_t(i+1)$ 之间的关系，这样通过递推，我们就可以得到整个观测序列出现的概率。那么，下面我们来进行推导：

$$\alpha_t(i+1) = P(o_1, \cdots, o_t, o_{t+1}, i_{t+1} = q_j|\lambda) \tag{7}$$

因为 $\alpha_t(i)$ 里面有 $i_t = q_j$，我们就要想办法把 i_t 给塞进去，所以：

$$\begin{aligned} \alpha_t(i+1) &= P(o_1, \cdots, o_t, o_{t+1}, i_{t+1} = q_j|\lambda) \\ &= \sum_{i=1}^{N} P(o_1, \cdots, o_t, o_{t+1}, i_t = q_i, i_{t+1} = q_j|\lambda) \\ &= \sum_{i=1}^{N} P(o_{t+1}|o_1, \cdots, o_t, i_t = q_i, i_{t+1} = q_j, \lambda) \cdot P(o_1, \cdots, o_t, i_t = q_i, i_{t+1} = q_j|\lambda) \end{aligned} \tag{8}$$

又根据观测独立性假设，我们可以很显然的得到 $P(o_{t+1}|o_1,\cdots,o_t,i_t=q_i,i_{t+1}=q_j,\lambda)=P(o_{t+1}|i_{t+1}=q_j)$。所以：

$$\begin{aligned}\alpha_t(i+1)=&\sum_{i=1}^{N}P(o_{t+1}|o_1,\cdots,o_t,i_t=q_i,i_{t+1}=q_j,\lambda)\cdot P(o_1,\cdots,o_t,i_t=q_i,i_{t+1}=q_j|\lambda)\\=&\sum_{i=1}^{N}P(o_{t+1}|i_{t+1}=q_j)\cdot P(o_1,\cdots,o_t,i_t=q_i,i_{t+1}=q_j|\lambda)\end{aligned}\tag{9}$$

看到这个化简后的公式，我们关注一下和 $\alpha_t(i)$ 相比，好像还多了一项 $i_{t+1}=q_j$，我们下一步的工作就是消去它。所以：

$$P(o_1,\cdots,o_t,i_t=q_i,i_{t+1}=q_j|\lambda)=P(i_{t+1}=q_j|o_1,\cdots,o_t,i_t=q_i,\lambda)\cdot P(o_1,\cdots,o_t,i_t=q_i|\lambda)\tag{10}$$

根据齐次马尔可夫性质，我们可以得到 $P(i_{t+1}=q_j|o_1,\cdots,o_t,i_t=q_i,\lambda)=P(i_{t+1}=q_j=i_t=q_i)$。所以根据以上的推导，我们可以得到：

$$\begin{aligned}\alpha_{t+1}(j)=&\sum_{i=1}^{N}P(o_{t+1}|i_{t+1}=q_j)\cdot P(i_{t+1}=q_j|i_t=q_i)\cdot P(o_1,\cdots,o_t,i_t=q_i|\lambda)\\=&b_j(o_{t+1})\cdot a_{ij}\cdot\alpha_t(i)\end{aligned}\tag{11}$$

经过上述的推导，我们就成功的得到了 $\alpha_{t+1}(j)$ 和 $\alpha_t(i)$ 之间的关系。通过这个递推关系，就可以遍历整个 Markov Model 了。这个公式是什么意思呢？它可以被我们表达为，所有可能出现的隐变量状态乘以转移到状态 j 的概率，乘以根据隐变量 i_{t+1} 观察到 o_{t+1} 的概率，乘上根据上一个隐状态观察到的观察变量的序列的概率。

我们可以用一个图来进行表示：

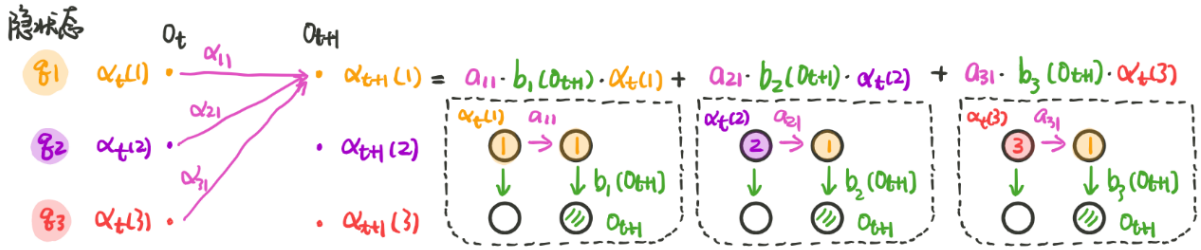

图 2: Hidden Markov Model 前向传播示意图

其实读神经网络了解的同学就会发现，这实际上和前向传播神经网络非常的像，实际上就是状态的值乘以权重。也就是对于上一个隐状态的不同取值分别计算概率之后再求和。这样每次计算，有隐状态的状态空间数为 N，序列的长度为 T，那么总的时间复杂度为 $\mathcal{O}(TN^2)$。

3 Backward Algorithm

后向概率的推导实际上比前向概率的理解要难一些，前向算法实际上是一个联合概率，而后向算法则是一个条件概率，所以后向的概率实际上比前向难求很多。

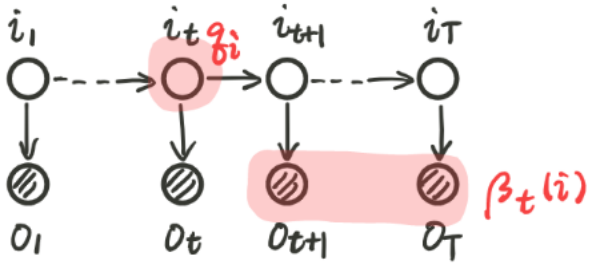

图 3: Hidden Markov Model 后向算法示意图

我们设 $\beta_t(i) = P(o_{t+1}, \cdots, o_T | i_t = q_i, \lambda)$，以此类推，$\beta_t(1) = P(o_2, \cdots, o_T | i_1 = q_i, \lambda)$。我们的目标是计算 $P(O|\lambda)$ 的概率，我们首先来推导一下这个公式：

$$
\begin{aligned}
P(O|\lambda) =& P(o_1, o_2, \cdots, o_N | \lambda) \\
=& \sum_{i=1}^{N} P(o_1, o_2, \cdots, o_N, i_1 = q_i | \lambda) \\
=& \sum_{i=1}^{N} P(o_1, o_2, \cdots, o_N | i_1 = q_i, \lambda) P(i_1 = q_i | \lambda) \\
=& \sum_{i=1}^{N} P(o_1 | o_2, \cdots, o_N, i_1 = q_i, \lambda) \cdot P(o_2, \cdots, o_N, i_1 = q_i | \lambda) \cdot \pi_i \\
=& \sum_{i=1}^{N} P(o_1 | i_1 = q_i, \lambda) \cdot \beta_1(i) \cdot \pi_i \\
=& \sum_{i=1}^{N} b_i(o_1) \cdot \pi_i \cdot \beta_1(i)
\end{aligned}
\tag{12}
$$

现在我们已经成功的找到了 $P(O|\lambda)$ 和第一个状态之间的关系。其中，π_i 为某个状态的初始状态的概率，$b_i(o_1)$ 表示为第 i 个隐变量产生第 1 个观测变量的概率，$\beta_1(i)$ 表示为第一个观测状态确定以后生成后面观测状态序列的概率。结构图如下所示：

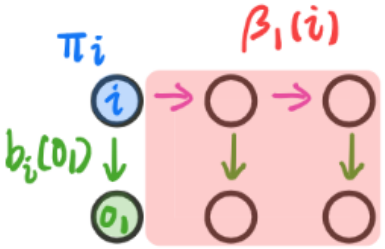

图 4: $P(O|\lambda)$ 与第一个状态之间的关系结构图

那么，我们下一步要通过递推，找到最后一个状态与第一个状态之间的关系。下面做如下的推导：

$$
\begin{aligned}
\beta_t(i) =& P(o_{t+1}, \cdots, o_T | i_t = q_i) \\
=& \sum_{j=1}^{N} P(o_{t+1}, \cdots, o_T, i_{t+1} = q_j | i_t = q_i) \\
=& \sum_{j=1}^{N} P(o_{t+1}, \cdots, o_T | i_{t+1} = q_j, i_t = q_i) \cdot \underbrace{P(i_{t+1} = q_j | i_t = q_i)}_{a_{ij}} \\
=& \sum_{j=1}^{N} P(o_{t+1}, \cdots, o_T | i_{t+1} = q_j) \cdot a_{ij} \\
=& \sum_{j=1}^{N} P(o_{t+1} | o_{t+2} \cdots, o_T, i_{t+1} = q_j) \cdot \underbrace{P(o_{t+2} \cdots, o_T | i_{t+1})}_{\beta_{t+1}(j)} \cdot a_{ij} \\
=& \sum_{j=1}^{N} P(o_{t+1} | i_{t+1} = q_j) \cdot \beta_{t+1}(j) \cdot a_{ij} \\
=& \sum_{j=1}^{N} b_j(o_{t+1}) \cdot \beta_{t+1}(j) \cdot a_{ij}
\end{aligned}
\tag{13}
$$

其中第三行到第四行的推导 $P(o_{t+1}, \cdots, o_T | i_{t+1} = q_j, i_t = q_i) = P(o_{t+1}, \cdots, o_T | i_{t+1} = q_j)$ 使用的马尔可夫链的性质，每一个状态都是后面状态的充分统计量，与之前的状态无关。通过这样的迭代从后往前推，我们就可以得到 $\beta_i(1)$ 的概率，从而推断出 $P(O|\lambda)$。整体的推断流程图如下图所示：

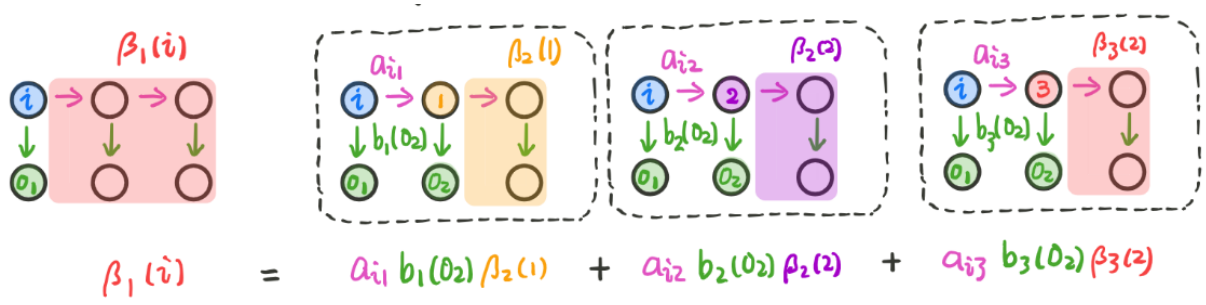

图 5: Hidden Markov Model 后向算法拓扑结构图

Hidden Markov Model 03 Learning

Chen Gong

09 January 2020

首先我们回顾一下，上一节讲的有关 Evaluation 的问题。Evaluation 可以被我们描述为在已知模型 λ 的情况下，求观察序列的概率。也就是：

$$P(O|\lambda) = \sum_{I} P(O, I|\lambda) = \sum_{i_1} \cdots \sum_{i_T} \pi_{i_1} \prod_{t=2}^{T} a_{i_{t-1}, i_t} \prod_{t=1}^{T} b_{i_1}(o_t) \tag{1}$$

此时的算法复杂度为 $\mathcal{O}(N^T)$。算法的复杂度太高了，所以，就有了后来的 forward 和 backward 算法。那么就有如下定义：

$$\begin{aligned}
&\alpha_t(i) = P(o_1, \cdots, o_t, i_t = q_i|\lambda) \\
&\beta_t(i) = P(o_{t+1}, \cdots, o_T|i_t = q_i, \lambda) \\
&\alpha_T(i) = P(O, i_T = q_i) \rightarrow P(O|\lambda) = \sum_{i=1}^{N} \alpha_T(i) \\
&\beta_1(i) = P(o_2, \cdots, o_T|i_1 = q_i, \lambda) \rightarrow P(O|\lambda) = \sum_{i=1}^{N} \pi_i b_i(o_1) \beta_1(i)
\end{aligned} \tag{2}$$

而使用 forward 和 backward 算法的复杂度为 $\mathcal{O}(TN^2)$。这一节，我们就要分析 Learning 的部分，Learning 就是要在已知观测数据的情况下求参数 λ，也就是：

$$\lambda_{MLE} = \arg\max_{\lambda} P(O|\lambda) \tag{3}$$

1 Learning

我们需要计算的目标是：

$$\lambda_{MLE} = \arg\max_{\lambda} P(O|\lambda) \tag{4}$$

又因为：

$$P(O|\lambda) = \sum_{i_1} \cdots \sum_{i_T} \pi_{i_1} \prod_{t=2}^{T} a_{i_{t-1}, i_t} \prod_{t=1}^{T} b_{i_1}(o_t) \tag{5}$$

对这个方程的 λ 求偏导，实在是太难算了。所以，我们考虑使用 EM 算法。我们先来回顾一下 EM 算法：

$$\theta^{(t+1)} = \arg\max_{\theta} \int_{z} \log P(X, Z|\theta) \cdot P(Z|X, \theta^{(t)}) dZ \tag{6}$$

而 $X \rightarrow O$ 为观测变量；$Z \rightarrow I$ 为隐变量，其中 I 为离散变量；$\theta \rightarrow \lambda$ 为参数。那么，我们可以将公式改写为：

$$\lambda^{(t+1)} = \arg\max_{\lambda} \sum_{I} \log P(O, I|\lambda) \cdot P(I|O, \lambda^{(t)}) \tag{7}$$

这里的 $\lambda^{(t)}$ 是一个常数，而：

$$P(I|O, \lambda^{(t)}) = \frac{P(I, O|\lambda^{(t)})}{P(O|\lambda^{(t)})} \tag{8}$$

并且 $P(O|\lambda^{(t)})$ 中 $\lambda^{(t)}$ 是常数，所以这项是个定量，与 λ 无关，所以 $\frac{P(I,O|\lambda^{(t)})}{P(O|\lambda^{(t)})} \propto P(I, O|\lambda^{(t)})$。所以，我们可以将等式 (7) 改写为：

$$\lambda^{(t+1)} = \arg\max_{\lambda} \sum_{I} \log P(O, I|\lambda) \cdot P(I, O|\lambda^{(t)}) \tag{9}$$

这样做有什么目的呢？很显然这样可以把 $\log P(O, I|\lambda)$ 和 $P(I, O|\lambda^{(t)})$ 变成一种形式。其中，$\lambda^{(t)} = (\pi^{(t)}, \mathcal{A}^{(t)}, \mathcal{B}^{(t)})$，而 $\lambda^{(t+1)} = (\pi^{(t+1)}, \mathcal{A}^{(t+1)}, \mathcal{B}^{(t+1)})$。

我们定义：

$$Q(\lambda, \lambda^{(t)}) = \sum_{I} \log P(O, I|\lambda) \cdot P(O, I|\lambda^{(t)}) \tag{10}$$

而其中，

$$P(O|\lambda) = \sum_{i_1} \cdots \sum_{i_T} \pi_{i_1} \prod_{t=2}^{T} a_{i_{t-1}, i_t} \prod_{t=1}^{T} b_{i_1}(o_t) \tag{11}$$

所以，

$$Q(\lambda, \lambda^{(t)}) = \sum_{I} \left[\left(\log \pi_{i_1} + \sum_{t=2}^{T} \log a_{i_{t-1}, i_t} + \sum_{t=1}^{T} \log b_{i_t}(o_t) \right) \cdot P(O, I|\lambda^{(t)}) \right] \tag{12}$$

2 以 $\pi^{(t+1)}$ 为例

这小节中我们以 $\pi^{(t+1)}$ 为例，在公式 $Q(\lambda, \lambda^{(t)})$ 中，$\sum_{t=2}^{T} \log a_{i_{t-1}, i_t}$ 与 $\sum_{t=1}^{T} \log b_{i_t}(o_t)$ 与 π 无关，所以，

$$\begin{aligned} \pi^{(t+1)} &= \arg\max_{\pi} Q(\lambda, \lambda^{(t)}) \\ &= \arg\max_{\pi} \sum_{I} [\log \pi_{i_1} \cdot P(O, I|\lambda^{(t)})] \\ &= \arg\max_{\pi} \sum_{i_1} \cdots \sum_{i_T} [\log \pi_{i_1} \cdot P(O, i_1, \cdots, i_T|\lambda^{(t)})] \end{aligned} \tag{13}$$

我们观察 $\{i_2, \cdots, i_T\}$ 就可以知道，联合概率分布求和可以得到边缘概率。所以：

$$\begin{aligned} \pi^{(t+1)} &= \arg\max_{\pi} \sum_{i_1} [\log \pi_{i_1} \cdot P(O, i_1|\lambda^{(t)})] \\ &= \arg\max_{\pi} \sum_{i=1}^{N} [\log \pi_i \cdot P(O, i_1 = q_i|\lambda^{(t)})] \qquad (s.t. \sum_{i=1}^{N} \pi_i = 1) \end{aligned} \tag{14}$$

2.1 拉格朗日乘子法求解

根据拉格朗日乘子法，我们可以将损失函数写完：

$$\mathcal{L}(\pi,\eta)=\sum_{i=1}^{N}\log\pi_i\cdot P(O,i_1=q_i|\lambda^{(t)})+\eta(\sum_{i=1}^{N}\pi_i-1) \tag{15}$$

使似然函数最大化，则是对损失函数 $\mathcal{L}(\pi,\eta)$ 求偏导，则为：

$$\frac{\mathcal{L}}{\pi_i}=\frac{1}{\pi_i}P(O,i_1=q_i|\lambda^{(t)})+\eta=0 \tag{16}$$

$$P(O,i_1=q_i|\lambda^{(t)})+\pi_i\eta=0 \tag{17}$$

又因为 $\sum_{i=1}^{N}\pi_i=1$，所以，我们将公式 (17) 进行求和，可以得到：

$$\sum_{i=1}^{N}P(O,i_1=q_i|\lambda^{(t)})+\pi_i\eta=0\Rightarrow P(O|\lambda^{(t)})+\eta=0 \tag{18}$$

所以，我们解得 $\eta=-P(O|\lambda^{(t)})$，从而推出：

$$\pi_i^{(t+1)}=\frac{P(O,i_1=q_i|\lambda^{(t)})}{P(O|\lambda^{(t)})} \tag{19}$$

进而，我们就可以推导出 $\pi^{(t+1)}=(\pi_1^{(t+1)},\pi_2^{(t+1)},\cdots,\pi_N^{(t+1)}$。而 $\mathcal{A}^{(t+1)}$ 和 $\mathcal{B}^{(t+1)}$ 也都是同样的求法。这就是大名鼎鼎的 Baum Welch 算法，实际上思路和 EM 算法一致。不过在 Baum Welch 算法诞生之前，还没有系统的出现 EM 算法的归纳。所以，这个作者还是很厉害的。

Hidden Markov Model 04 Decoding

Chen Gong

10 January 2020

Decoding 问题可被我们描述为：

$$\hat{I} = \arg\max_{I} P(I|O, \lambda) \tag{1}$$

也就是在给定观察序列的情况下，寻找最大概率可能出现的隐概率状态序列。也有人说 Decoding 问题是预测问题，但是实际上这样说是并不合适的。预测问题应该是，$P(o_{t+1}|o_1, \cdots, o_t)$ 和 $P(i_{t+1}|o_1, \cdots, o_t)$，这里的 $P(i_1, \cdots, i_t|o_1, \cdots, o_t)$ 看成是预测问题显然是不合适的。

1 Decoding Problem

下面我们展示一下 Hidden Markov Model 的拓扑模型：

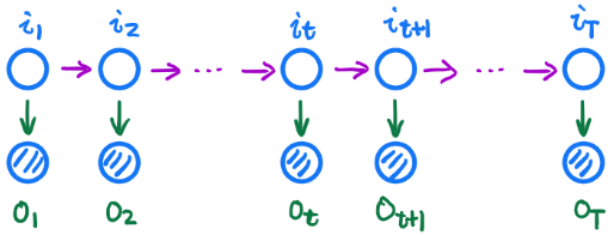

图 1: Hidden Markov Model 的拓扑模型

这里实际上就是一个动态规划问题，这里的动态规划问题实际上就是最大概率问题，只不过将平时提到的最大距离问题等价于最大概率问题，理论上都是一样的。每个时刻都有 N 个状态，所有也就是从 N^T 个可能的序列中找出概率最大的一个序列，实际上就是一个动态规划问题，如下图所示：

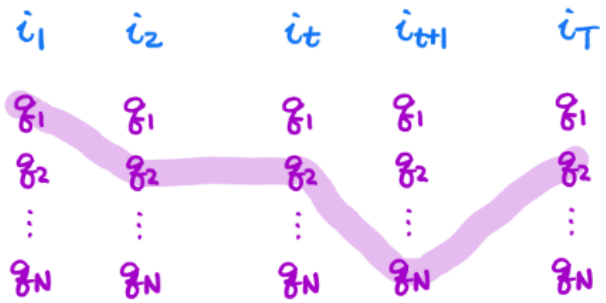

图 2: Decoding 的动态规划问题

我们假设：

$$\delta_t(i) = \max_{i_1,\cdots,i_{t-1}} P(o_1,\cdots,o_t,i_1,\cdots,i_{t-1},i_t = q_i) \tag{2}$$

这个等式是什么意思呢？也就是当 t 个时刻是 q_i，前面 $t-1$ 个随便走，只要可以到达 q_i 这个状态就行，而从中选取概率最大的序列。我们下一步的目标就是在知道 $\delta_t(i)$ 的情况下如何求 $\delta_t(i+1)$，那么这样就能通过递推来求得知道最后一个状态下概率最大的序列。$\delta_t(i+1)$ 的求解方法如下所示：

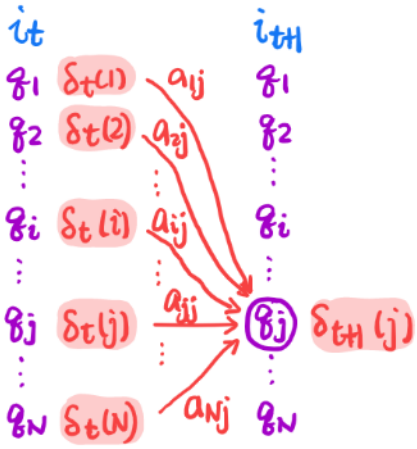

图 3: 根据 $\delta_t(i)$ 求解 $\delta_t(i+1)$ 的方法示意图

所以，

$$\begin{aligned}\delta_{t+1}(j) &= \max_{i_1,\cdots,i_t} P(o_1,\cdots,o_{t+1},i_1,\cdots,i_t,i_{t+1} = q_j) \\ &= \max_{i_1,\cdots,i_t} \delta_t(i)\cdot a_{ij}\cdot b_j(o_{t+1})\end{aligned} \tag{3}$$

这就是 Viterbi 算法，但是这个算法最后求得的是一个值，没有办法求得路径，如果要想求得路径，我们需要引入一个变量：

$$\varphi_{t+1}(j) = \arg\max_{1\le i\le N} \delta_t(i)\cdot a_{ij}\cdot b_j(o_{t+1}) \tag{4}$$

这个函数用来干嘛的呢？他是来记录每一次迭代过程中经过的状态的 index。这样我们最终得到的 $\{\varphi_1,\varphi_2,\cdots,\varphi_T\}$，就可以得到整个路径了。

Hidden Markov Model 05 Conclusion

Chen Gong

11 January 2020

Hidden Markov Model 实际上是一个 Dynamic Model。我们以 Guassian Mixture Model (GMM) 为例。对于一个观测状态，在隐变量状态给定的情况下，是符合一个 Gaussian Distribution，也就是 $D(O|i_1) \sim \mathcal{N}(\mu, \Sigma)$。如果，加入了 time 的因素就是 Hidden Markov Model，而其中 $\{i_1, i_2, \cdots, i_T\}$ 是离散的就行，这些我们在第一章的背景部分有过讨论。而观测变量 o_1 是离散的还是连续的都不重要。

1 Hidden Markov Model 简述

Hidden Markov Model，可以用一个模型，两个假设和是三个问题来描述。一个模型就是指 $\lambda = (\pi, \mathcal{A}, \mathcal{B})$。其中，$\pi$：指的是初始概率分布；$\mathcal{A}$：指的是状态转移矩阵；$\mathcal{B}$：指的是发射矩阵，也就是在已知隐变量的情况下，得到观测变量的概率分布。

两个假设：1. 齐次马尔可夫模型，马尔科夫性质中非常重要的一条。2. 观测独立假设，也就是观测变量只和当前的隐变量状态有关。

三个问题：1. Evaluation：$P(O|\lambda)$，也就是在在已知模型的情况下，求观测变量出现的概率。2. Learning：$\hat{\lambda} = \arg\max_\lambda P(O|\lambda)$，在已知观测变量的情况下求解隐马尔可夫模型的参数。3. Decoding：$P(I|O) = P(i_1, \cdots, i_t|o_1, \cdots, o_t)$，用公式的语言描述就是 $\hat{I} = \arg\max_I P(I, O|\lambda)$。

2 Dynamic Model

Dynamic Model 实际上就是一个 State Space Model，通常我们可以将 Dynamic Model 的问题分成两类。第一类为 Learning 问题，即为，参数 λ 是未知的，通过数据来知道参数是什么；第二类就是 Inference 问题，也就是在 λ 未知的情况下，推断后验概率。实际上，我们需要求的就是 $P(Z|X)$，其中 $X = \{x_1, x_2, \cdots, x_N\}$ 而数据之间是非 i.i.d 的。

Inference 问题大概可以被我们分成四类，Filtering Problem；Smoothing；Prediction Problem；Decoding。

2.1 Learning

Learing 问题中 λ 是已知的，$\lambda_{MLE} = \arg\max_\lambda P(X|\lambda)$。我们采用的是 Baum Welch Algorithm，算法思想上和 EM 算法类似，实际上也是 Forward-Backward 算法。

2.2 Inference

这一小节中，我们将分别来介绍 Filtering；Smoothing；Prediction；Decoding，四个问题。

2.2.1 Decoding

这里前面已经做出过详细的描述了，这里就不再展开进行描述了，主要可以概括为：在已知观测数据序列的情况下，求得出现概率最大的隐变量序列，被我们描述为：$Z = \arg\max_z P(z_1, \cdots, z_t | x_1, \cdots, x_t)$。我们使用的一种动态规划的算法，被称为 Viterbi Algorithm。

2.3 Evaluation

在还有大家应该见得比较多的 Prob of Evidence 问题，也就是：$P(X|\theta) = P(x_1, \cdots, x_t\theta)$。我们通俗的称之为证据分布，实际上就是我们前面讲到的 Evaluation 方法。也就是在已知参数的情况下，求观测数据序列出现的概率，用公式描述即为：$P(X|\theta) = P(x_1, x_2, \cdots, x_t|\theta)$。

2.3.1 Filtering

实际上是一个 Online-Learning 的过程，也就是如果不停的往模型里面喂数据，我们可以得到概率分布为：$P(z_t|x_1, \cdots, x_t)$。所以 Filtering 非常的适合与 on-line update。我们要求的这个就是隐变量的边缘后验分布。为什么叫滤波呢？这是由于我们求的后验是 $P(z_t|x_1, \cdots, x_t)$，运用到了大量的历史信息，比 $P(z_t|x_t)$ 的推断更加的精确，可以过滤掉更多的噪声，所以被我们称为 “过滤”。求解过程如下所示：

$$P(z_t|x_{1:t}) = \frac{P(z_t, x_1, \cdots, x_t)}{P(x_1, \cdots, x_t)} = \frac{P(z_t, x_1 : x_t)}{\sum_{z_t} P(z_t, x_1 : x_t)} \propto P(z_t, x_1 : x_t) \tag{1}$$

2.3.2 Smoothing

Smoothing 问题和 Filtering 问题的性质非常的像，不同的是，Smoothing 问题需要观测的是一个不变的完整序列。对于 Smoothing 问题的计算，前面的过程和 Filtering 一样，都是：

$$P(z_t|x_{1:T}) = \frac{P(z_t, x_1, \cdots, x_T)}{P(x_1, \cdots, x_T)} = \frac{P(z_t, x_1 : x_T)}{\sum_{z_t} P(z_t, x_1 : x_T)} \propto P(z_t, x_1 : x_T) \tag{2}$$

同样因为 $\sum_{z_t} P(z_t, x_1 : x_T)$ 是一个归一化常数，我们这里不予考虑。下面的主要问题是关于 $P(z_t, x_1 : x_T)$ 如何计算，我们来进行推导：

$$\begin{aligned} P(x_{1:T}, z_t) =& P(x_{1:t}, x_{t+1:T}, z_t) \\ =& P(x_{t+1:T}|x_{1:t}, z_t) \cdot \underbrace{P(x_{1:t}, z_t)}_{\alpha_t} \end{aligned} \tag{3}$$

推导到了这里就是要对 $P(\underbrace{x_{t+1:T}}_{C} | \underbrace{x_{1:t}}_{A}, \underbrace{z_t}_{B})$ 进行分析，在这个概率图模型中，符合如下结构：

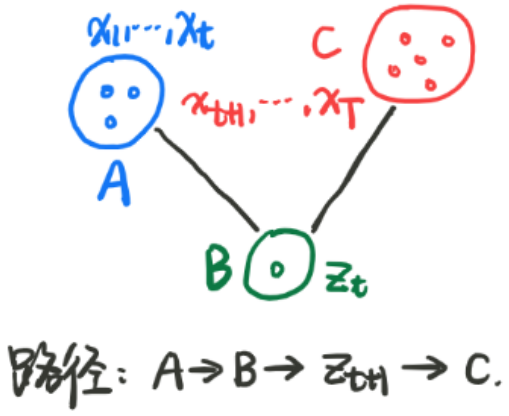

图 1: A，B，z_{t+1}，C，概率图结构图

根据概率图模型中提到 D-Separation 中，我们可以很简单的得出，$A \perp C|B$。所以，$P(x_{t+1:T}|x_{1:t}, z_t) = P(x_{t+1:T}|x_{1:t}, z_t = \beta_t)$。所以，我们可以得到:

$$P(x_{1:T}, z_t) = \alpha_t \cdot \beta_t \tag{4}$$

那么，最终得到的就是:

$$P(z_t|x_{1:T}) \propto P(x_{1:T}, z_t) = \alpha_t\beta_t \tag{5}$$

所以，我们需要同时用到 Forward Algorithm 和 Backward Algorithm，所以，被我们称为 Forward-Backward Algorithm。

2.3.3 Prediction

预测问题，大体上被我们分成两个方面:

$$\begin{aligned} P(z_{t+1}|x_1, \cdots, x_t) &= \sum_{z_t} P(z_{t+1}, z_t|x_1, \cdots, x_t) \\ &= \sum_{z_t} \underbrace{P(z_{t+1}|z_t, x_1, \cdots, x_t)}_{P(z_{t+1}|z_t)} \underbrace{P(z_t|z_t, x_1, \cdots, x_t)}_{Filtering} \end{aligned} \tag{6}$$

$$\begin{aligned} P(x_{t+1}|x_1, \cdots, x_t) &= \sum_{z_{t+1}} P(x_{t+1}, z_{t+1}|x_1, \cdots, x_t) \\ &= \underbrace{P(x_{t+1}|z_{t+1}, x_1, \cdots, x_t)}_{P(x_{t+1}|z_{t+1})} \cdot \underbrace{P(z_{t+1}|x_1, \cdots, x_t)}_{Formula(6)} \end{aligned} \tag{7}$$

公式 (7) 选择从 z_{t+1} 进行积分的原因是因为想利用齐次马尔科夫性质。实际上求解的过程大同小异都是缺什么就补什么。

其实，我们已经大致的介绍了 Dynamic Model 的几种主要模型，后面我们会详细的来解释线性动态系统。

Kalman Filter 01 Introduction

Chen Gong

16 January 2020

我们知道在概率图模型中，加入了 time 的因素，就得到了 Dynamic Model，实际上也就说我们通常所说的 State Space Model。

如果状态是离散的，就是我们上一节提到了 Hidden Markov Model (HMM)；**如果状态是连续的**，如果状态之间的关系是线性的，就是 Linear Dynamic System (Kalman Filter)，或者说是 Linear Gaussian Model；如果状态之间的关系是 Non-Linear 的或者 Non-Gaussian 的，那么也就是 Particle Filter。我们这一章主要描述的就是 Kalman Filter。

1 Dynamic Model Introduction

第一类问题，Learning 问题，即为在已知观测序列 O 的情况下求解 $P(\pi|O)$。其中，模型可以描述为 $\pi\{\lambda, \mathcal{A}, \mathcal{B}\}$。代表性的就是 Hidden Markov Model。

第二类问题就是 Inference 问题，大致可以分为 Decoding，Probability of Evidence，Filtering，Smoothing 和 Prediction 五类问题。这里中 Hidden Markov Model 05 Conclusion 我们有非常详细的描述。详情可以关注 Hidden Markov Model。

2 Kalman Filtering: Linear Gaussian Model / linear Dynamic System

Filtering 问题就是求 $P(z_t|x_1, x_2, \cdots, x_t)$，实际上就是一个 Marginal Posterior 问题。对于 Linear 关系，Linear 主要反映在相邻时刻的两个状态之间的转移关系，当前时刻的隐变量状态和观测状态之间的关系。描述如下所示：

$$\begin{aligned} z_t &= A \cdot z_{t-1} + B + \epsilon \\ x_t &= C \cdot z_t + D + \delta \end{aligned} \tag{1}$$

z_t, z_{t-1} 和 x_t, z_t 之间体现了线性的关系。而 ϵ, δ 是符合 Gaussian Distribution 的，$\epsilon \sim \mathcal{N}(0, Q), \delta \sim \mathcal{N}(0, R)$。所以，大家都明白了 Linear 和 Gaussian 都是从何而来的，所以 Kalman Filtering 被称为 Linear Gaussian Model 更合适。

Filtering 是一类问题的总称，我们之前在 Hidden Markov Model 中有详细的讨论过。那么，我们回顾一下 Hidden Markov Model 的基本信息做一个对比。

HMM：$\lambda = \{\pi, \mathcal{A}, \mathcal{B}\}$。

状态转移矩阵：

$$\begin{aligned} A &= [a_{ij}] \quad a_{ij} = P(i_{t+1} = q_j | i_t = q_i) \\ B &= [b_j(k)] \quad b_j k = P(o_t = v_t | i_t = q_j) \end{aligned} \tag{2}$$

那么，对于 Kalman Filtering 来说，状态转移矩阵，发射概率，初始矩阵，模型参数我们可以做出类似的表达：

$$P(z_t | z_{t-1}) \sim \mathcal{N}(A \cdot z_{t-1} + B, Q) \tag{3}$$

$$P(x_t | z_t) \sim \mathcal{N}(C \cdot z_t + D, R) \tag{4}$$

$$z_1 \sim \mathcal{N}(\mu_1, \Sigma_1) \tag{5}$$

$$\theta = \{A, B, C, D, Q, R, \mu_1, \Sigma_1\} \tag{6}$$

在这一小节中，我们已经了解了基础的相关概念，那下一小节中，我们将描述了 Filtering 问题的建模和求解。

Kalman Filter 02 Model Construction & Solution

Chen Gong

17 January 2020

Filtering 问题公式话的表达即为 $P(z_t|x_1, x_2, \cdots, x_t)$，是一种 On-Line Learning 的思路，随着越来越多的数据不断的被观测到，隐藏状态得到不断的更新。也就是在观察变量序列 $\{x_1, x_2, \cdots, x_t\}$ 下，求得隐变量状态 z_t 的分布。模型表达为如下所示：

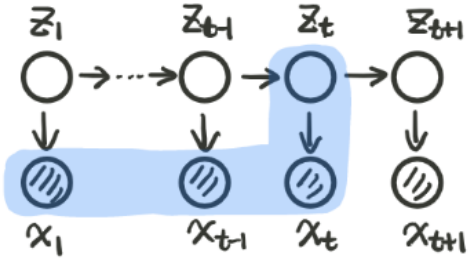

图 1: 模型基本拓扑结构

首先我们回顾一下前向算法的求解思路。在这个算法中首先定义了中间变量为：

$$\alpha_t(i) = P(x_1, x_2, \cdots, x_t, z_t = q_i) \tag{1}$$

而我们下一步则是要寻找 $\alpha_{t+1}(i)$ 和 $\alpha_t(i)$ 之间的关系。所以，可以按 $\alpha_1(i), \alpha_2(i), \cdots, \alpha_t(i)$ 的顺序依次推断得到 $\alpha_t(i)$，从而得到根据当前的模型推断出观测序列的分布 $P(O|\lambda)$。

1 Filtering 问题思路

我们还是采用的前向算法的思路：

$$\begin{aligned}
P(z_t|x_1, x_2, \cdots, x_t) =& \frac{P(z_t, x_1, x_2, \cdots, x_t)}{P(x_1, x_2, \cdots, x_t)} \\
\propto& P(z_t, x_1, x_2, \cdots, x_t) \\
=& \underbrace{P(x_t|x_1, x_2, \cdots, x_{t-1}, z_t)}_{P(x_t|z_t)} P(x_1, x_2, \cdots, x_{t-1}, z_t) \\
=& P(x_t|z_t) \underbrace{P(z_t|x_1, x_2, \cdots, x_{t-1})}_{prediction} \underbrace{P(x_1, x_2, \cdots, x_{t-1})}_{const} \\
\propto& P(x_t|z_t) P(z_t|x_1, x_2, \cdots, x_{t-1})
\end{aligned} \tag{2}$$

很显然通过如上的推导，我们将 Filtering 问题回归到了一个 Prediction 的问题。那么这个 Prediction 的问题，如何进一步求解呢？下一步，我们对 Prediction 的部分进行推导。

$$\begin{aligned} P(z_t|x_1,x_2,\cdots,x_{t-1}) &= \int_{z_{t-1}} P(z_t,z_{t-1}|x_1,x_2,\cdots,x_{t-1})dz_{t-1} \\ &= \int_{z_{t-1}} \underbrace{P(z_t|z_{t-1},x_1,x_2,\cdots,x_{t-1})}_{P(z_t|z_{t-1})} \underbrace{P(z_{t-1}|x_1,x_2,\cdots,x_{t-1})}_{Filtering} dz_{t-1} \end{aligned} \tag{3}$$

通知上述的推导，我们又回到了一个 Filtering 的问题，那么这样我们形成了一个递归的表达。那么，我们可以总结为在一个 Filtering 问题中，我们通过一个 Prediction 问题，来构建形成了一个回归。那么，下面我将详细的说明一下求解的过程：

$$\begin{aligned} t=1 \quad & \begin{cases} P(z_1|x_1) & update \\ p(z_2|x_1) & prediction \end{cases} \\ t=2 \quad & \begin{cases} P(z_2|x_1,x_2) & update \\ p(z_3|x_1,x_2) & prediction \end{cases} \\ \cdots \quad & \cdots \\ t=T \quad & \begin{cases} P(z_T|x_1,x_2,\cdots,x_T) & update \\ p(z_{T+1}|x_1,x_2,\cdots,x_{T-1}) & prediction \end{cases} \end{aligned} \tag{4}$$

很显然，我们可以不断的往里面添加数据来更新隐变量状态 z_t。

2 Filtering 问题求解具体分析

首先，我们需要明确一个问题，Gaussian Distribution 是一个具有非常好的性质的自共轭分布。通俗的讲就是，Gaussian 分布的边缘分布，条件分布，联合概率分布等都是符合高斯分布的。首位，我先回忆一下在 Math Basis 那小节中，总结的线性高斯模型中，已知条件高斯分布，求变量高斯分布的公式：

$$P(X) = \mathcal{N}(X|\mu,\Lambda^{-1}) \tag{5}$$

$$P(Y|X) = \mathcal{N}(X|AX+b,L^{-1}) \tag{6}$$

$$P(Y) = \mathcal{N}(Y|AX+b,L^{-1}+A^{-1}\Lambda A) \tag{7}$$

$$P(X|Y) = \mathcal{N}(\Sigma\{A^TL(y-b)+\Lambda\mu\},\Sigma) \quad \Sigma=(\Lambda+A^TLA)^{-1} \tag{8}$$

从上小节中我们分析了 Filtering 问题的推导过程，我们可以看到 Filtering 问题可以被大致分成两个部分，也就是 Prediction 和 Update 两个部分。上一小节中我描述了大致的求解思路，那么这一小节我们将详细的描述怎么计算。

2.1 Prediction

预测问题被我们描述为，$P(z_t|x_1,x_2,\cdots,x_{t-1})$，那下面我们来进行分析怎么求。

$$P(z_t|x_1,x_2,\cdots,x_{t-1}) = \int_{z_{t-1}} P(z_t|z_{t-1})P(z_{t-1}|x_1,x_2,\cdots,x_{t-1})dz_{t-1} \tag{9}$$

根据 Gaussian Distribution 的自共轭性，所以 $P(z_t|x_1, x_2, \cdots, x_{t-1})$ 一定是一个 Gaussian Distribution。事实上 $x_1, x_2, \cdots, x_{t-1}$ 中所有的信息都是已知的。为了方便表达 $P(z_t|x_1, x_2, \cdots, x_{t-1}) \sim P(z_t)$。

那么，第 $t-1$ 时刻的假设 $P(z_{t-1}|x_1, x_2, \cdots, x_{t-1}) \sim \mathcal{N}(\mu_{t-1}, \Sigma_{t-1})$。那么，第 t 时刻的分布 $P(z_t|x_1, x_2, \cdots, x_t) \sim \mathcal{N}(\mu_t^*, \Sigma_t^*)$。并且，根据 Gaussian Distribution 的自共轭性，我们可以令 $P(z_t|z_{t-1}) \sim \mathcal{N}(z_t|Az_{t-1} + B, Q)$。令 $x = z_{t-1}, y = z_t$，代公式 (5)-(7)，可以计算出：

$$\begin{cases} \mu_t^* = A\mu_{t-1} + B \\ \Sigma_t^* = Q + A\Sigma_{t-1}A^T \end{cases} \tag{10}$$

2.2 Update

然而，对于 update 问题，我们的目标是求解：

$$P(z_t|x_1, x_2, \cdots, x_t) \propto P(x_t|z_t) \cdot P(z_t|x_1, x_2, \cdots, x_{t-1}) \tag{11}$$

在这个问题中 $x_1, x_2, \cdots, x_{t-1}$ 都是已知的，而 x_t 是未知的。所以，公式 (11) 可以被改写为：

$$\underbrace{P(z_t|x_1, x_2, \cdots, x_t)}_{P(X|Y)} \propto \underbrace{P(x_t|z_t)}_{P(Y|X)} \cdot \underbrace{P(z_t|x_1, x_2, \cdots, x_{t-1})}_{P(X)} \tag{12}$$

同样利用 Guassian Distribution 的自共轭性，我们可以将公式改写为：

$$\mathcal{N}(\mu_t, \Sigma_t) \propto \mathcal{N}(x_t|Cz_t + D, R) \cdot \mathcal{N}(\mu_t^*, \Sigma_t^*) \tag{13}$$

所以，利用公式 (5,6,8) 我们可以求解出 μ_t 和 Σ_t。根据公式 (8)，我们其实可以看到这个求解过程实际上非常的复杂，实际上也是一个代公式的过程。我们就不再做过多的描述了 (实际上把公式一代入，把符号换一下就可以了)。

所以将第一小节和第二小节结合起来一下，第一小节给出了求解的主体思路，第二小节中给出了每一步具体如何实现。并且利用了 Gaussian Linear Model 的计算公式来进行求解。

Particle Filter

Chen Gong

09 February 2020

目录

1 背景介绍

Dynamic Model 是在概率图模型中加入了时序的因素，所以样本之间不再是独立同分布 (i.i.d) 的，而是有依赖关系的。而 Dynamic Model 的一个主要特点是，混合模型。因为，我们看到的都是观测变量序列，而每一个观测变量都对应着一个隐变量，隐变量也被称之为系统变量 (System Variable)，所以有时我们也将 Dynamic Model 称之为 State Space Model。

而 Dynamic Model 我们可以从两个假设，两个方程，三个问题的角度去分析。

1.1 两个假设

这是有关 Dynamic Model 的两个假设，也是我们研究这个模型的前提条件。这两个假设可以大大的帮助我们简化模型：

1. **齐次 Markov 假设 (无后向性)：**未来与过去无关，只依赖与当前的状态。数学公式描述也就是。

$$P(i_{t+1}|i_t, i_{t-1}, \cdots, i_1, o_t, \cdots, o_1) = P(i_{t+1}|i_t) \tag{1}$$

2. **观测独立假设：**当前时刻的观测变量只依赖于当前时刻的隐变量。

$$P(o_t|i_t, i_{t-1}, \cdots, i_1, o_t, \cdots, o_1) = P(o_t|i_t) \tag{2}$$

1.2 两个方程

这两个方程分别描述的是，隐变量状态与状态之间的转移概率，由当前时刻隐变量推出当前时刻观测变量的概率，符合化描述如下所示：

$$\begin{cases} z_t = g(z_{t-1}, \mu, \xi) \\ x_t = h(z_t, \mu, \delta) \end{cases} \tag{3}$$

经过类比，我们就可以很简单的发现，$g(\cdot)$ 函数就是 HMM 中的状态转移矩阵 A，$h(\cdot)$ 函数就是 HMM 中的发射转移矩阵 B。

1.3 三大类问题

在 Dynamic Model 中，我们主要可以分为三大类问题。

1.3.1 隐马尔可夫模型 (Hidden Markov Model)

这个在之前的章节中，我们已经有了详细的讲解。主要特点就是，隐状态 z 之间是离散的，而观测变量 s 之间并没有限制。在 Hidden Markov Model 中，主要关注的是 Decoding 问题，也就是在已知观测序列的情况下，最大化可能的隐状态。

1.3.2 线性动态系统 (Linear Dynamic System)

线性动态系通常也被我们称为线性高斯系统，而线性高斯系统这个名字更加形象，而线性和高斯的来源主要如下所示：

$$\begin{cases} z_t = A \cdot z_{t-1} + B + \epsilon & \epsilon \sim \mathcal{N}(0, Q) \\ x_t = C \cdot z_t + D + \delta & \delta \sim \mathcal{N}(0, R) \end{cases} \tag{4}$$

也就是式 (3) 中的两个函数将符合线性和噪声符合 Gaussian Distribution 的原则。而 Dynamic Model 中重点关注的就是 Filter 问题，Filter 问题就是求解 $P(z_t|x_1, x_2, \cdots, x_t)$，在已知观测序列的情况下，求解当前时刻的隐变量的状态。

1.3.3 Filter 问题的求解

我们的求解目标是 $P(z_t|x_1, x_2, \cdots, x_t)$。

Step 1. Prediction：这个过程我们可以理解成给 z_t 一个先验，

$$P(z_t|x_1, x_2, \cdots, x_{t-1}) = \int_{z_{t-1}} P(z_t|z_{t-1})P(z_{t-1}|x_1, x_2, \cdots, x_{t-1})dz_{t-1} \tag{5}$$

Step 2. Update：这个过程我们可以理解成给 z_t 在已知 x_t 之后的后验，

$$P(z_t|x_1, x_2, \cdots, x_t) \propto P(x_t|z_t) \cdot P(z_t|x_1, x_2, \cdots, x_{t-1}) \tag{6}$$

1.3.4 Preiction 的证明

$$\begin{aligned} P(z_t|x_1, x_2, \cdots, x_{t-1}) &= \int_{z_{t-1}} P(z_t, z_{t-1}|x_1, x_2, \cdots, x_{t-1})dz_{t-1} \\ &= \int_{z_{t-1}} P(z_t|z_{t-1}, x_1, x_2, \cdots, x_{t-1})P(z_{t-1}|x_1, x_2, \cdots, x_{t-1})dz_{t-1} \end{aligned} \tag{7}$$

根据 Markov 齐次假设，$P(z_t|z_{t-1}, x_1, x_2, \cdots, x_{t-1}) = P(z_t|z_{t-1})$，$P(z_{t-1}|x_1, x_2, \cdots, x_{t-1})$ 就是 $t-1$ 时刻的 data。替换一下就可以得到公式 (5)。

1.3.5 Update 的证明

首先，我们提一下，$P(x_1, x_2, \cdots, x_t)$ 这种，只和观察数据有关的概率分布都是可以计算出来的，我们都用不同的常数来表示。:

$$\begin{aligned} P(z_t|x_1, x_2, \cdots, x_t) =& \frac{P(z_t, x_1, x_2, \cdots, x_t)}{P(x_1, x_2, \cdots, x_t)} \\ =& \frac{1}{C}P(z_t, x_1, x_2, \cdots, x_t) \\ =& \frac{1}{C}\underbrace{P(x_t|z_t, x_1, x_2, \cdots, x_{t-1})}_{P(x_t|z_t)}P(z_t, x_1, x_2, \cdots, x_{t-1}) \\ =& \frac{1}{C}P(x_t|z_t)P(z_t|x_1, x_2, \cdots, x_{t-1})\underbrace{P(x_1, x_2, \cdots, x_{t-1})}_{const\ D} \\ =& \frac{D}{C}P(x_t|z_t)P(z_t|x_1, x_2, \cdots, x_{t-1}) \end{aligned} \tag{8}$$

而 $P(x_t|z_t)$ 就是发射矩阵，$P(z_t|x_1, x_2, \cdots, x_{t-1})$ 就是 $t-1$ 时刻的 Prediction。

由于多维 Gaussian Distribution 非常强大的自共轭性，所以条件概率，边缘概率，联合概率这些都是 Gaussian Distribution 的，所以可以将高维的高斯分布进行拆解成多个低维的来进行求解就会简单一点。而 Linear Dynamic System 也被称为 Kalman Filter。

1.4 非线性动态系统 (Non-Linear Dynamic System)

Non-Linear Dynamic System 和 Linear Dynamic System 区别是，相邻两个隐变量状态 z_t 和 z_{t-1} 之间的关系是任意的，而且，Noise 也是 Gaussian 的。所以，无法使用 Kalman Filter 一样的方法得到解析解。所以，只能采用 Monte-Carlo Sampling 一样的采样方法来得出近似解。

Non-Linear 没有那么好的特征，求不出，只能采样。在贝叶斯框架的 Inference 主要要求解的是一个后验分布 $P(Z|X)$。而这个分布主要是用来求解期望，$\mathbb{E}_{z|x}[f(x)] = \int f(z)p(z|x)dx$。如果，我们可以采取 N 个样本，$z^{(i)} \sim P(Z|X), \{z^{(1)}, z^{(2)}, \cdots, z^{(N)}\}$。那么，我们就可以通过 $\frac{1}{N}\sum_{i=1}^{N} f(z^{(i)})$ 来求解。这就是最简单的 Monte-Carlo Sampling 的方法。

2 重要性采样 (Importance Sampling)

重要性采样并不是直接对概率分布进行采样，而是对提议 (Proposal) 分布进行采样。也就是：

$$\begin{aligned}\mathbb{E}_{p(z)}[f(z)] = \int p(z)f(z)dz &= \int \frac{p(z)}{q(z)}q(z)f(z)dz \\ &= \int f(z)\frac{p(z)}{q(z)}q(z)dz \\ &\approx \frac{1}{N}\sum_{i=1}^{N} f(z_i)\frac{p(z_i)}{q(z_i)} \ (z_i \sim q(z),\ i = 1, 2, \cdots, N)\end{aligned} \tag{9}$$

而这里的 $\frac{p(z_i)}{q(z_i)}$ 也就是 Weight，用来平衡不同的概率密度值之间的差距。同样重要性采样也可能会出现一些问题，就是两个分布之间的差距太大了话，总是采样采不到重要的样本，采的可能都是实际分布概率值小的部分。也就是采样效率不均匀的问题。

之后为了方便描述，我们用 $x_{1:t} = x_1, x_2, \cdots, x_t$。

如果是在 Filtering 的问题中，目标是求解 $P(z_t|x_{1:t})$，权值为：

$$w_t^{(i)} = \frac{P(z_t^{(i)}|x_{1:t})}{Q(z_t^{(i)}|x_{1:t})} \tag{10}$$

而在每一个时刻，都有 N 个样本点：

$$\begin{aligned}&t = 1: \ w_1^{(i)},\ i = 1, 2, \cdots, N \sim w_1^{(1)}, w_1^{(2)}, \cdots, w_1^{(N)} \\ &t = 2: \ w_2^{(i)},\ i = 1, 2, \cdots, N \sim w_2^{(1)}, w_2^{(2)}, \cdots, w_2^{(N)} \\ &\cdots\cdots \\ &t = T: \ w_T^{(i)},\ i = 1, 2, \cdots, N \sim w_T^{(1)}, w_T^{(2)}, \cdots, w_T^{(N)}\end{aligned} \tag{11}$$

而实际上 $P(z_t^{(i)}|x_{1:t})$ 的求解非常的复杂，而这样的迭代式求解，实际上复杂度非常的高。我们计算起来非常的复杂。我们想在 $w_T^{(i)}$ 和 $w_{T-1}^{(i)}$ 之间寻找一个递推关系式来大大的简化计算。然后就引出了下面的 Sequential Importance Sampling (SIS) 算法。

3 顺序重要性采样 (Sequential Importance Sampling)

这个算法中的主要思想就是，找到 $w_T^{(i)}$ 和 $w_{T-1}^{(i)}$ 之间的一个递推关系式来简化计算。

在这个算法的开始，做出了一个我觉得有点神奇的铺垫。那就是将求解的重点做了一个转变：

$$P(z_t|x_{1:t}) \longrightarrow P(z_{1:t}|x_{1:t}) \tag{12}$$

实际上 $P(z_t|x_{1:t})$ 是 $P(z_{1:t}|x_{1:t})$ 的一个边缘概率分布，而老师讲了一个很自然我们可以直接丢弃 $z_{1:t-1}$，从而使得 $P(z_t|x_{1:t})$ 和 $P(z_{1:t}|x_{1:t})$ 等价处理。本人水平有限，所以实在没有想清楚这到底是怎样一个自然法。所以，我理解的是，在 $x_{1:t}$ 都知道的情况下，$P(z_{1:t}|x_{1:t})$ 的边缘概率求起来比较方便。大家有什么好的想法，欢迎留言讨论。

而 $P(z_{1:t}|x_{1:t})$ 对应的权重 $w_t^{(i)}$ 为：

$$w_t^{(i)} \propto \frac{P(z_{1:t}^{(i)}|x_{1:t})}{Q(z_{1:t}^{(i)}|x_{1:t})} \tag{13}$$

我们要将这个公式中的分子和分母拆开进行化简.

3.1 分子 $P(z_{1:t}^{(i)}|x_{1:t})$ 解析

$$\begin{aligned}
P(z_{1:t}^{(i)}|x_{1:t}) =& \frac{P(z_{1:t}^{(i)}, x_{1:t})}{\underbrace{P(x_{1:t})}_{C}} \\
=& \frac{1}{C} P(z_{1:t}^{(i)}, x_{1:t}) \\
=& \frac{1}{C} \underbrace{P(x_t|z_{1:t}^{(i)}, x_{1:t-1})}_{P(x_t|z_t^{(i)})} P(z_{1:t}^{(i)}, x_{1:t-1}) \\
=& \frac{1}{C} P(x_t|z_t^{(i)}) \underbrace{P(z_t^{(i)}|z_{1:t-1}^{(i)}, x_{1:t-1})}_{P(z_t^{(i)}|z_{t-1})} P(z_{1:t-1}^{(i)}, x_{1:t-1}) \\
=& \frac{1}{C} P(x_t|z_t^{(i)}) P(z_t^{(i)}|z_{t-1}) P(z_{1:t-1}^{(i)}, x_{1:t-1}) \\
=& \frac{1}{C} P(x_t|z_t^{(i)}) P(z_t^{(i)}|z_{t-1}) P(z_{1:t-1}^{(i)}|x_{1:t-1}) \underbrace{P(x_{1:t-1})}_{D} \\
=& \frac{D}{C} P(x_t|z_t^{(i)}) P(z_t^{(i)}|z_{t-1}) P(z_{1:t-1}^{(i)}|x_{1:t-1})
\end{aligned} \tag{14}$$

3.2 分母 $Q(z_{1:t}^{(i)}|x_{1:t})$ 解析

分母中，我们假设：

$$Q(z_{1:t}^{(i)}|x_{1:t}) = Q(z_t^{(i)}|z_{1:t-1}^{(i)}, x_{1:t}) Q(z_{1:t-1}^{(i)}|x_{1:t-1}) \tag{15}$$

有的同学会很好奇 $Q(z_{1:t-1}^{(i)}|x_{1:t-1})$ 为什么不是 $Q(z_{1:t-1}^{(i)}|x_{1:t})$。这个很好解释，这是一个假设。$Q(\cdot)$ 本来就是一个 Proposal Distribution，我们想设成什么样都没有关系。

所以，我们将分子和分母的解析结果汇总可以得到：

$$\begin{aligned} w_t^{(i)} &\propto \frac{P(z_{1:t}^{(i)}|x_{1:t})}{Q(z_{1:t}^{(i)}|x_{1:t})} \propto \frac{P(x_t|z_t^{(i)})P(z_t^{(i)}|z_{t-1})P(z_{1:t-1}^{(i)}|x_{1:t-1})}{Q(z_t^{(i)}|z_{1:t-1}^{(i)},x_{1:t})Q(z_{1:t-1}^{(i)}|x_{1:t-1})} \\ &= \frac{P(x_t|z_t^{(i)})P(z_t^{(i)}|z_{t-1})}{Q(z_t^{(i)}|z_{1:t-1}^{(i)},x_{1:t})} \cdot w_{t-1}^{(i)} \end{aligned} \tag{16}$$

$P(x_t|z_t^{(i)})$ 是已知的发射矩阵，$P(z_t^{(i)}|z_{t-1})$ 也是已知的状态转移矩阵，$Q(z_t^{(i)}|z_{1:t-1}^{(i)},x_{1:t})$ 是 Proposal Distribution 很简单，很好进行求解。那么对于在 t 时刻的 N 个值 $w_t^{(i)}$，我们不在需要一个个的进行计算了，使用迭代公式就可以很快的求得解。

3.3 算法小结

我们反复强调过，求解后验概率分布 $P(Z|X)$ 大多数时候都是为了求解期望 $\mathbb{E}_{Z|X}[f(Z)]$，而实际上期望的问题就是一个积分问题。为什么求期望这么的重要呢，我们仔细想想。求方差的过程本质上就是一个求期望的过程：$Var[X] = \mathbb{E}[X^2] - [\mathbb{E}[X]]^2$。而求边缘概率也可以转换成一个求期望的问题，本质上还是一个积分的问题。

前面在公式 (16) 中得到了，

$$w_t^{(i)} \propto \frac{P(z_{1:t}^{(i)}|x_{1:t})}{Q(z_{1:t}^{(i)}|x_{1:t})} \propto \frac{P(x_t|z_t^{(i)})P(z_t^{(i)}|z_{t-1})}{Q(z_t^{(i)}|z_{1:t-1}^{(i)},x_{1:t})} \cdot w_{t-1}^{(i)} \tag{17}$$

我们之前提过了关于 Proposal Distribution $Q(\cdot)$ 是可以随意设置了，为了求解的方便。我们将 $Q(z_t^{(i)}|z_{1:t-1}^{(i)},x_{1:t})$ 改写成 $Q(z_t^{(i)}|z_{t-1}^{(i)},x_{1:t})$。这个转换非常的自然。下面，我们将梳理一下这个算法：

Sequential Importance Sampling Algorithm：

前提：$t-1$ 时刻的采样都已经完成；

t 时刻对于 N 个采样值，for i=1, 2,$\cdots$, N:

$$z_t^{(i)} \sim Q(z_t|z_{t-1},x_{1:t}) \tag{18}$$

$$w_t^{(i)} \propto \frac{P(x_t|z_t^{(i)})P(z_t^{(i)}|z_{t-1})}{Q(z_t^{(i)}|z_{1:t-1}^{(i)},x_{1:t})} \cdot w_{t-1}^{(i)} \tag{19}$$

end

$w_t^{(i)}$ 归一化，令 $\sum_{i=1}^N w_t^{(i)} = 1$。

实际上就这么简单，归一化是为了方便计算和避免程序计算时出现精度问题，而且可以对值的范围进行约束。所以，公式 (9) 中，我们可以将最后的结果进行改写，可得：

$$\frac{1}{N}\sum_{i=1}^N f(z^{(i)})w^{(i)} = \sum_{i=1}^N f(z^{(i)})\hat{w}^{(i)} \tag{20}$$

这里讲的也很模糊，我用直观性的方法来理解。由于有了归一化的过程，$\hat{w}^{(i)}$ 就相当于一个概率密度函数，那么 $\sum_{i=1}^N f(z^{(i)})\hat{w}^{(i)}$ 求得的不就是期望。这样一样的可以得到异曲同工的作用。

3.4 Sequential Importance Sampling 中的问题

Sequential Importance Sampling 最大的问题就是**权值退化**。也就是 $w_t^{(i)}$ 会变得越来越小。举个例子，假如有 100 个权值，有 99 个都趋向于 0，中有 1 个趋向于 1。这样当然会出问题。

而为什么会出现这个问题呢？由于高维空间的原因，我们需要更多的样本来反映空间的特征。样本不够就会忽略一些高维特征，导致某些维度特征接近 0 的情况出现。

那么，解决方法有三种：

1. 采更多的样本；
2. Resampling 的方法；
3. 选择一个更好的 Proposal Distribution Q(z)。

4 Sampling Importance Resampling (SIR)

那么 Resampling 的方法如何来解决权值退化的方法？那就是舍弃权重。之前我们用 Weight 来代表一个粒子的重要程度，而 Resampling 就是通过用粒子数量来代替 Weight 的方法来表示重要程度。

4.1 SIR 采样方法

经过重要性采样后，我们得到了 N 个样本点，以及对应的权重。那么我用权重来作为采样的概率，重新测采样出 N 个样本。也就是如图 1所示：

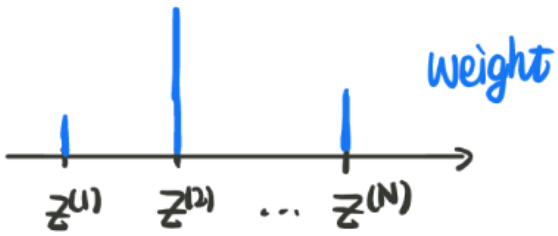

图 1: Sampling Importance Resampling 示意图

通过二次采样可以降低采样不平衡的问题。至于为什么呢？大家想一想，我在这里表达一下自己的看法。$\frac{p(z_i)}{q(z_i)}$ 是 Weight，如果 Weight 比较大的话，说明 $p(z_i)$ 比较大而 $q(z_i)$ 比较的小，也就是我们通过 $q(z_i)$ 采出来的数量比较少。那么我们按权重再来采一次，就可以增加采到重要性样本的概率，成功的弥补了重要性采样带来的缺陷，有效的弥补采样不均衡的问题。

那么，权值大的样本我们就采样的比较多，权值小的样本就采样的比较少。这样就去掉了权值，换了一种方法等价的来表示重要程度。就可以解决权值退化的问题。

4.2 How to do it?

具体实现的思路其实就很简单。那就是根据概率密度函数 (pdf) 来计算得到概率分布函数 (cdf)。然后通过从 [0,1] 之间进行均匀采样，来得到相应的的值。这个，我们在 Markov Chain Monte Carlo 01 Sampling Method 的**概率分布采样**中已经做了详细的描述。

讲到这里，Sampling Importance Resampling (SIR) 实际上就是一种 Basis Particle Filter，也就是 SIS+Resampling，这是一种基本可以 work 的算法。

4.3 找一个更好的 $Q(Z)$

选择 $Q(z_t|z_{1:t-1},x_{1:t})=P(z_t|z_{t-1}^{(i)})$，用状态转移矩阵来进行定义，那么我们就可以将权值改写成：

$$w_t^{(i)} \propto \frac{P(x_t|z_t^{(i)})P(z_t^{(i)}|z_{t-1})}{P(z_t|z_{t-1}^{(i)})}\cdot w_{t-1}^{(i)} = P(x_t|z_t^{(i)})\cdot w_{t-1}^{(i)} \tag{21}$$

所以，$w_t^{(i)}=P(x_t|z_t^{(i)})\cdot w_{t-1}^{(i)}$，且 $z^{(i)}\sim P(z_t|z_{t-1}^{(i)})$。改写后的算法为 SIR Filter，也就是下列三个部分组成的：

$$\text{Sequential Importance Sampling} + \text{Resampling} + Q(z_t|z_{1:t-1},x_{1:t}) = P(z_t|z_{t-1}^{(i)})$$

那么我们为什么要令 $Q(z_t|z_{1:t-1},x_{1:t})=P(z_t|z_{t-1}^{(i)})$ 呢？因为在 t 时刻采样的时候，不是要去找一个新的分布，而是就从之前就用的一个中进行寻找。我们可以用一个很简单的例子来说明：就像很多人寻找自己的另一半，找了很久都没有合适的，直到最后才发现另一半就在自己身边的朋友中，也就是“兔子就吃窝边草”吧。我们用之前就算好的会方便很多。

我们用，generate and test，来描述这个过程非常的形象。Generate 是 $z_t \longrightarrow P(z_t|Z_{t-1}^{(i)})$，也就是状态转移矩阵。而 Test 是 w_{t-1}，也就是 $P(x_t|z_t^{(i)})$ 这个实际上就是发射矩阵，这个实际上是一举两得的作用。$P(x_t|z_t^{(i)})$ 越大一方面表示了发射矩阵的概率大；另一方面是表示了采样的权重很大，采出来的样本越重要，采样的效率越高。所以，$Q(z_t|z_{1:t-1},x_{1:t})=P(z_t|z_{t-1}^{(i)})$ 呢？因为在 t 起到了一语双关的重用，实现了共同目标的优化。

5 Sampling Importance Resampling (SIR) 总结

Sampling Importance Resampling Algorithm:

前提：$t-1$ 时刻的采样都已经完成；

1.Sampling:

t 时刻对于 N 个采样值，for i=1, 2,$\cdots$, N:

$$z_t^{(i)} \sim P(z_t|z_{t-1}^{(i)}) \tag{22}$$

$$w_t^{(i)} \propto \frac{P(x_t|z_t^{(i)})P(z_t^{(i)}|z_{t-1})}{P(z_t|z_{t-1}^{(i)})}\cdot w_{t-1}^{(i)} = P(x_t|z_t^{(i)})\cdot w_{t-1}^{(i)} \tag{23}$$

end

2. Normalized:

$$w_t^{(i)} \longrightarrow \hat{w}_t^{(i)}, \quad \sum_{i=1}^{N}\hat{w}_t^{(i)} = 1 \tag{24}$$

3. Resampling:

$$\hat{\hat{w}}_t^{(i)} = \frac{1}{N} \tag{25}$$

Conditional Random Field

Chen Gong

20 February 2020

目录

1 Introduction

本讲主要介绍的是条件随机场 (Conditional Random Field)，这个东西在机器学习中，曾经有过较大的用处，在图像处理和标注问题中大放光彩。本小节的讲解，主要是 CRF 机器学习体系中的背景，我们为什么要研究 CRF，CRF 和其他的模型相比它在什么地方进行了演变，然后对 CRF 模型的建立和求解进行了分析，最后得出 CRF 适用于怎样的问题，它有怎样的优缺点等。这个过程是很流畅的，和前面讲到的概率图模型中的隐马尔可夫模型 (Hidden Markov Model) 和最大熵原理等都有一定的联系。细节请往下详细阅读。

2 Background

实际上机器学习中一类最重要的问题就是分类 (classify) 的问题。分类的方法可以被我们分为两类，硬分类和软分类。所谓硬分类就是输出值是离散的，比如 $y \in \{0, 1\}$；而软分类就是输出的是属于某一个类的概率，比如 $P(y = 1), P(y = 0)$ 的概率。

2.1 硬分类 (Hard classification)

硬分类就是输入特征，输出的离散变量，通俗的讲就是，分类器看到了一堆特征，直接告诉你这个个体属于哪一类。

1. **支持向量机 (Support Vector Machine)**：支持向量机的思路来源是几何间隔。模型可以被我们写为：

$$\begin{cases} \min \frac{1}{2} w^T w \\ s.t.\ y_i(w^T x_i + b) \leq 1,\ i = 1, 2, \cdots, N \end{cases} \tag{1}$$

2. **多层感知机 (PLA)：**多层感知机的思路来源是误差驱动。模型可以被我们写成：

$$f(w) = \text{sign}(w^T x)$$

3. **线性判别分析 (Linear Discriminate analysis)：**主要采用的思想是类间大，类内小。

2.2 软分类 (Soft classification)

软分类就是输入特征，输出的是每一种可能性的概率。通俗的讲就是，给定一组特征，分类器输出的是属于各个类别的概率，最后选择是属于哪一类？你自己就看着办吧，看你怎么选了。所以，软分类模型求得是一个分布就是这个原因。算法可以分为两个方面，概率判别模型和概率生成模型。这两者有什么不一样呢？

概率判别模型求的是 $P(y|x)$；也就是将 label 根据提供的特征，学习，最后画出了一个明显或许比较明显的边界，比如 SVM，Perceptron，神经网络，LR 等都是这么工作的。

而概率生成模型求的是联合概率分布 $P(x, y)$，然后你来了一个新的数据以后，我们通过贝叶斯公式化简，我们可以得到 $P(y|x) \propto P(x, y)$，从而得到 $P(y|x)$。生成模型关注结果是如何产生的，可以求出所有 label 的概率。它包含的信息非常的全，不仅可以用来输出 label，还可以用来做很多别的事情。所以，生成模型需要非常多的数据量来保证采样到了数据本来的面目，所以速度慢。

2.2.1 概率判别模型

概率判别模型中，最基础的算法就是 Logistics Regression，这个算法非常的简单。我们曾经在指数族分布那一章讲过，在给定数据和已知事实的情况下，指数族分布是可以使所有的结果经历等可能出现的分布，也就是满足最大熵原理。看到这，大家估计对最大熵原理有点懵逼，之前老师也只是提到了这个原理并说明了这个原理是什么，对于为什么并没有更层次的讲解。我们接下来讲解一下：

最大熵原理：其定义是对于概率模型，在所有可能分布的概率模型中，熵最大的模型是最好的模型，也就是令所有的结果都有可能出现。有的同学就会发出疑问，一方面我们总希望将事物的不确定性降到最低；另一方面对于不确定的事物我们又认为保留最大的不确定性是最好的选择。这样做不会精神分裂吗？我觉得可以这样来思考，对于我们知道的部分，我们要尽可能去靠近它；对于我们不知道的部分，我们要保持它的结果等可能的出现的可能性，就是最大限度的保留不确定性。而等可能性就是由熵最大来刻画的。俗话所说："知之为知之，不知为不知，是知也。"

这里我们只给出最大熵原理一些直观的概念，并没有对其进行深入的理论研究，这并不是我们这节的重点。在这里我们描述的目的，是想说明 **Logistics Regression 是最大熵原理 (Maximum Entropy Model) 的一个特例，使用的是最大 Entropy 的思想**。

小伙伴们可能在这里有点懵逼，我来简要描述一下：

首先我们介绍一下最大熵原理的求解结果，这里的推导，有兴趣的同学自己去看，我这里只写结论。实际上指数族分布那一章就推导过了，结果就是指数族分布。

$$P_w(y|x) = \frac{\exp(\sum_{i=1}^{n} w_i f_i(x,y))}{\sum_y \exp\left(\sum_{i=1}^{n} w_i f_i(x,y)\right)} \tag{2}$$

写到了这里不知道大家有没有发现，这个最大熵原理最后的求解结果和 Softmax 函数是一毛一样的，是不是终于知道 Softmax 的作用了？Softmax 本来就是用来解决多分类问题的，而 Logistics Regressio 只是一个二分问题，所以说，LR 模型本质上是最大熵模型的一个特例，下面给出详细的证明过程：

- 对于给定数据集，$X = \{x_1, x_2, \cdots, x_n\}^T$，我们构建如下图所示的特征函数：

$$f_i(x,y) = \begin{cases} x_i & y = 1 \\ 0 & y = 0 \end{cases} \tag{3}$$

- 根据最大熵的求解模型为：

$$\begin{aligned} \sum_y \exp\left(\sum_{i=1}^{n} w_i f_i(x,y)\right) &= \exp\left(\sum_{i=1}^{n} w_i f_i(x,y=0)\right) + \exp\left(\sum_{i=1}^{n} w_i f_i(x,y=1)\right) \\ &= 1 + \exp(WX) \end{aligned} \tag{4}$$

而根据公式 (2)；我们可以简单的计算出后面的结果。

- 当 $y = 1$ 时，可得：

$$P_w(y=1|x) = \frac{\exp(WX)}{1+\exp(WX)} \tag{5}$$

- 当 $y = 0$ 时，可得：

$$P_w(y=1|x) = \frac{1}{1+\exp(WX)} \tag{6}$$

所以，我们可以看到 LR 模型，实际上就是最大熵模型的一种特殊情况。而 LR 模型实际上是一种 Log Linear Model，为什么这么说？实际上，我们可以知道，最大熵模型的结果就是指数族分布，详情请见指数族分布。那么：

$$\log P(x) = \log \exp(\eta^T f(x)) = \eta^T f(x)$$

就是一个线性模型，所以叫 Log Linear Model。实际上看到这里，大家应该已经对深度学习中的一些设定有感觉了，不是凭空来的是有理论基础的。

最后，我们对最大熵原理做一个小结，我们的 CRF 中将会用到这个知识。**在给定已知事实的分析下，能够令熵达到最大的分布是指数族分布。而在给定均值，方差的情况下，Gaussian Distribution 的熵最大，也就是最随机，最具等可能性。**

2.2.2 概率生成模型

Naive Bayes 算法：概率生成模型的主要基础算法是 Naive Bayes 算法，贝叶斯算法主要是要求满足贝叶斯假设：

$$P(X|Y = 1/0) = \prod_{i=1}^{P} P(x_i|Y = 1/0)$$

Bayes 假设的概率图模型如下所示，根据概率图模型的 D-Separation 原则，这是一个 Tail To Tail 的模型在中间值给定的情况下，其他的节点之间都是相互独立的。

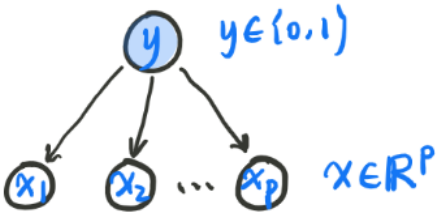

图 1: 条件独立性假设

我们可以将其定义为 $x_i \perp x_j|y$ $(i \neq j)$。根据贝叶斯公式可以得：

$$p(y|x) = \frac{p(x|y)p(y)}{p(x)} = \frac{p(x,y)}{p(x)} \propto p(x,y) \tag{7}$$

而做条件独立性假设的最终目的，是为了简化运算。因为对于一个数据序列 $x = (x_1, x_2, \cdots, x_p)$。如果 x_i 和 x_j 之间有关系的话，这个计算难度可能会变得很难，所以就假设各个变量之间是相互独立的。但是为了简化计算，这个假设太强了，显然有点不合理，没有充分利用数据之间的分布。

而我们将 $y(0/1 \to \text{Seq})$ 就得到了 Hidden Markov Model (HMM)，这是我们得到 HMM 的第一种思路。

高斯混合模型 (Gaussian Mixture Model)：高斯混合模型的概率图可以被我们描述为如下形式：

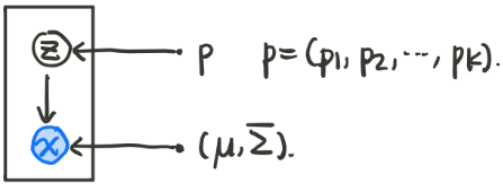

图 2: GMM 的概率图表达形式

我们根据一个离散的随机变量 Z 来选择是选取那个高斯分布，利用这个高斯分布 $\mathcal{N}(\mu, \Sigma)$ 来采样得到我们想要的样本点。而且，离散随机变量 Z 符合一个离散分布 $p = (p_1, p_2, \cdots, p_k)$。也就是 $P(X|Z) \sim \mathcal{N}(\mu, \Sigma)$，而在此基础上加上时间的影响就构成了 Hidden Markov Model，这是 HMM 的第二种引出方式。

隐马尔可夫模型 (Hidden Markov Model)：这个在我们之间的章节中有非常详细的描述，这里不再做过多的阐述，有兴趣的同学可以去仔细的观看。HMM 中有两条非常重要的假设，齐次马尔可夫假设和观测独立假设，我们的计算基本都是基于这两个假设来的。

Hidden Markov Model 的概率图模型如下所示：

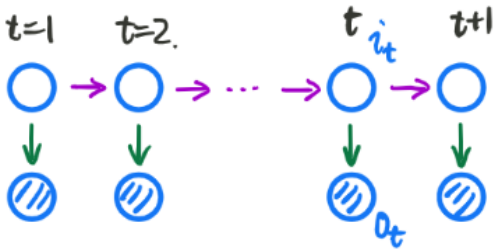

图 3: Hidden Markov Model 拓扑结构图

其中，x_1 为 observe data，y_1 为 lactent data，模型整体描述为 $\lambda = (\pi, \mathcal{A}, \mathcal{B})$。而实际上，我们还是感觉两个假设不太合理，限制了对数据集的利用，只是因为需要简化计算，所有没办法。但是，后来研究人员还是想消除假设，从而释放数据中更多的信息。

最大熵马尔可夫模型 (Maximum Entropy Markov Model)：这里我们只对 MEMM 模型做一些简单的描述，尽量上直观的去解释。

MEMM 是最大熵模型和 HMM 的结合体，它是一个判别模型。这里强调一下，它和 HMM 的主要区别是：**打破了 HMM 的观测独立假设**。这里我们后面会详细的描述。MEMM 在建模过程中用到了公式 (2) 中，最大熵模型的求解结果，主体还是用的 HMM 的结构，所以，被称为最大熵马尔可夫模型。

MEMM 的做法就是把输入分成两部分，1. 是所有的的 $x_{1:T}$ 对 y 的输入；2. 单独一个上一个状态 x_{t-1}，local 输入。有的同学可能会觉得有点奇怪，$x_{1:T}$ 中不就包括了 x_{t-1}，对的实际上很多时候都只看 $x_{1:T}$，这样分开说是为了便于理解在 HMM 上的改进。MEMM 的概率图模型如下所示：

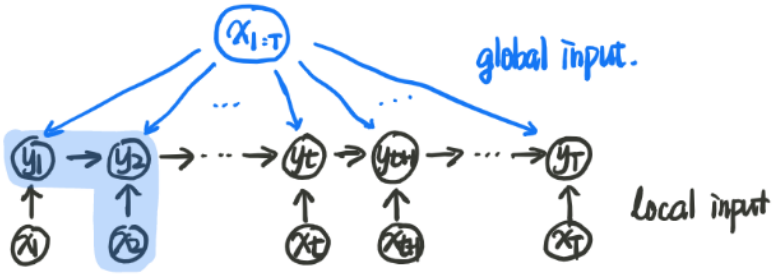

图 4: MEMM 的概率图模型

这小节的主要目的是引出我们为什么得到了 Conditional Random Field。MEMM 的细节不做过多解释。MEMM 的诞生 1. 首先将 HMM 中的观测独立假设给丢掉了，这样可以更多的使用原始数据中的信息；2. 而且，我们关注的问题从求解联合概率分布 $P(X,Y)$ 转换成了求解 $P(Y|X)$。在标注问题中并不需要计算那么多，求解 $P(Y|X)$ 就够了。

条件随机场 (Conditional Random Field) 虽然，MEMM 有了很大的改进，但是在历史的舞台了，MEMM 并没有大放光彩，这是为什么呢? 由于 MEMM 有一个致命的问题叫做 “Label Bias Problem”，这个问题主要产生的原因是局部归一化，这个问题还挺难的，这里简要的介绍一下，有兴趣的同学去看 John Latterfy 的相关文献。

我们简单介绍一下，看到图四中的阴影部分：为 $P(y_2|y_1,x_2)$ 局部是一个条件概率。为了让最后的结果都是一个概率分布，我们必须要做归一化处理：

$$\frac{\phi(y_1,x_1,y_2)}{\sum\phi(y_1,\cdots)}$$

这就是局部归一化的意思，至于会出现怎样的原因，这里没说，后面再做出详细的分析。这个原因是因为有向图的原因造成的，那么我们把有向图变成无向图不就可以解决这个问题了，于是条件随机场算法就诞生了。

2.3 小结

本小节我们从机器学习的整个框架中讨论了条件随机场是怎么来的，希望对同学们有一定的启发。判别模型和生成模型的概念还是挺重要的，搞懂之后，理解很多算法会有不一样的感受。

3 HMM VS MEMM

这节主要介绍演变过程。话不多说，首先介绍的是 HMM。

3.1 Hidden Markov Model

HMM 可以由 Naive Bayes 和 Gaussian Mixture Model 演变而来，是一种典型的生成模型。它的概率图模型如下所示：

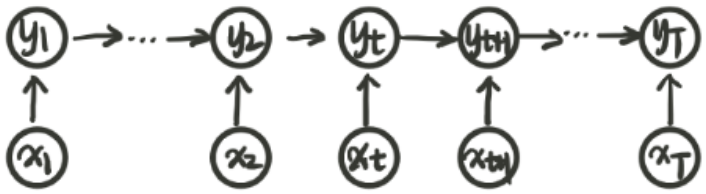

图 5: HMM 的概率图模型

这里 x 与 y 之间的箭头画反了方向，读者阅读的时候注意一下。模型描述为 $\lambda = (\pi, \mathcal{A}, \mathcal{B})$。两个假设，齐次马尔可夫假设和观测独立假设。

- **齐次一阶 Markov 假设：** 一阶：$P(y_t|y_{1:t-1}) = P(y_t|y_{t-1})$；齐次：$P(y_t|y_{t-1})$ 与时间无关。
- **观测独立假设：** $P(x_t|y_{1:t}, x_{1:t-1}) = P(x_t|y_t)$，大家想想观测独立假设为什么会成立？这实际和 Naive Bayes 中提到的概率图的 D-Separation 有异曲同工之处。因为 $y_{t-1} \to y_t \to x_t$ 是一个 head to head 结构，在 y_t 被观测的情况下 y_{t-1} 和 x_t 之间是独立的。
- **建模对象**：是 $P(X, Y|\lambda)$，它要求的是一个联合概率分布。这是一个生成模型，在 HMM 的 Evaluating 问题中，需要求的是 $P(X|\lambda)$，采用的大致思路都是 $P(X|\lambda) = \sum_Y P(X, Y|\lambda)$。经过大致的推导可以得出：

$$P(X, Y|\lambda) = \prod_{t=1}^{T} P(x_t, y_t|\lambda) = \prod_{t=1}^{T} P(y_t|y_{t-1}, \lambda) \cdot P(x_t|y_t, \lambda) \tag{8}$$

  但是，这个实际上并不会求，算法的复杂度是 $\mathcal{O}(T^N)$ 的，所有才有了后来的 Forward 和 Backward 算法这些。

其实观测独立假设，并没有什么道理可讲的。从 NB→HMM 是一样的。NB 中假设所有的样本之间相互独立，采用的是 Tail to Tail 的概率图模型，这样都是为了使计算变得简单而已。实际上这两种假设都是非必须的，都是为了计算，有它们反而破坏了原始数据分布的完整性。

3.2 Maximum Entropy Markov Model

MEMM 模型的概率图如下图所示：

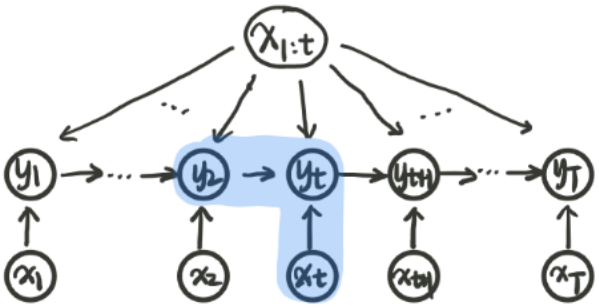

图 6: MEMM 的概率图模型

在 MEMM 中打破了观测独立假设，它是如何做到了呢?

大家注意观察，在 MEMM 中的 y 和 x 之间的箭头是相反的。那么，阴影部分就变成了 $y_{t-1} \rightarrow y_t \leftarrow x_t$。这种结构是典型的 Head to Tail 结构，根据 D-Separation 原则，在 y_t 被观察到的情况下，y_{t-1} 和 x_t 之间反而是有关系的。所以，它成功的打破了观测独立假设。

MEMM 不再是一个生成式模型而是一个判别式模型。

根据上面的描述我们发现，y_t 和 $y_{t-1}, x_t, x_{1:T}$ 之间都是有关系的。所以我们可以得到：

$$P(Y|X,\lambda) = \prod_{t=1}^{T} P(y_t|x_t, y_{t-1}, \lambda) = \prod_{t=1}^{T} P(y_t|x_{1:T}, y_{t-1}, \lambda) \tag{9}$$

所以，我们可以很清楚的看到可以是一个全量的输出，实际上只写上面就是 OK 的。在概率图模型中，只画上一半就可以了。而为了和 HMM 作较，我们这样写可以看出分成了两个部分，看到了 Global 和 Local 的影响。

3.3 小结

MEMM 比 HMM 有两个根有优势的地方：

1. 将生成模型变成了判别模型，不是说生成模型不够好，实际上是各具优点的。这里好的原因是，对于链式结构，我们主要处理的是标注问题，只要对条件建模就行了，没必要算那么多。至于为什么将生成模型变成了判别模型呢？这里老师没有讲，我来说说自己的观点。

在 HMM 中，根据概率图我们得到的解是 $P(x_t|y_t)$ 这种形式的，我们想要求的是 $p(y_t|x_t)$ 这种形式的。对于 $P(Y|X)$ 型的，我们只能利用 $P(Y|X)P(X) = P(X,Y)$，然后来求解 $P(X|Y)$，所以是生成模型。而，MEMM 中直接就是 $P(y_t|\cdots)$ 所以是判别模型。有同学有更好的理解，欢迎留言。

2. 打破了观测独立假设，物理上更有优势了，可以尽可能多的利用到原始数据分布之间的信息。

4 MEMM VS CRF

MEMM 的概率图模型在之前我们就展示过了，为了方便大家对下面内容的理解。这里再放一下：

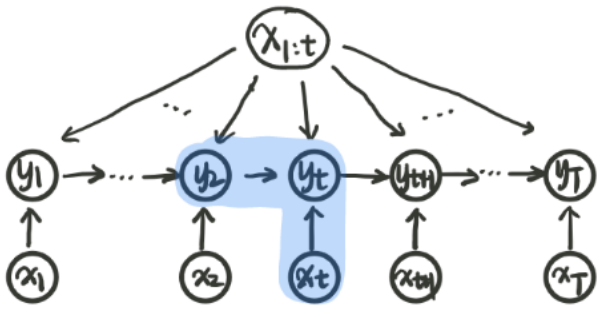

图 7: MEMM 的概率图模型

优点 (和 HMM 想比)：1. 判别式模型，有利于标注问题的 Decoding；2. 打破了观察独立假设。

模型为：$P(Y|X,\lambda) = \prod_t P(y_t|y_{t-1}, x_{1:T}, x_t, \lambda) = \prod_t P(y_t|y_{t-1}, x_{1:T}, \lambda)$。MEMM 是有向图，条件概率的条件都是在图中根据因果关系可以得出来的。

实际上我们之前说到，HMM 中的两个假设实际上，并不是很合理。而在 HMM 中，我们通过了将 x 和 y 之间的箭头改变的方法，成功的去除了观测独立假设。而接下来 MEMM 到 CRF 的演变，实际上是解决了齐次马尔可夫假设。

在 MEMM 中，它的概率图模型中带来了一个重大的缺陷就是 Label Bias Problem。通过将这个问题解决，我们正好就解除了齐次马尔可夫假设的限制。下面我们来描述这个问题：

4.1 MEMM 中的 Label Bias Problem

4.1.1 Label Bias Problem 定性分析

我们的定性的来看一下 Label Bias Problem。实际上图 7 中的阴影部分可以被我们看成是一个 Logistics Regression，为什么可以这样看？实际上是因为在 MEMM 的建模中，我们使用的是最大熵模型，而 Logistics Regression 就是最大熵模型的一个特例，这里可以将一个局部看成是一个 Logistics Regression。

假设我们将这个局部看成是一个系统的话，我可以将 y 看成是系统内部的状态，$y_{t-1} \to y_t$ 的转移符合一个概率分布，而 x_t 被我们看成是外界的输入。那么这个局部会不会产生怎样的问题呢？我们现在把这个局部产生的效果记为 “mass score”，那么这个名词它究竟是来自哪呢？如果针对一组连续变量，它的概率密度函数就是 Probability Density Function (PDF)；如果是连续的变量呢？则被称之为 Probability Mass Function (PMF)，这个 mass 就是来自这，特征离散变量。这个 mass score，可以看成是一个函数，就是系统从 $t-1$ 时刻到 t 时刻转移的一个得分，这个得分就是一个概率分布。这是一个函数，它和 y_{t-1}，y_t 和 x_t 都是有关系的。并且状态之间的转移都是具有定的能量的，而我们的问题出在哪呢？就是这个得分函数被归一化了。为什么要归一化？因为这三个小单体形成的一个局部，实际上是一个**概率分布**，概率分布就一定要归一化。

这个归一化是会带来一系列问题的，我们直观上来理解一下，他会带来的问题。如果把这个链条看成是一根长的绳子，我们在 t 这个点抖动一下这根绳子，肯定会对后续的绳子 $t+1, t+2, \cdots$ 都造成影响，但是如果在中间进行归一化，就相当于在某个点把绳子的结构粗细发生了变化。那么这个波就无法正常的影响到后面的绳子了，破坏了这个波能量的传递连贯性。这是一个比较形象的例子，我们下面引入 John Lafferty 论文中例子来给出一个形象的解释。

4.1.2 John Lafferty 论文中例子

归一化的不恰当会带来一些问题，但是论文中写的很简单，大多数同学看着会一脸懵逼，在这里我们做出了比较详细的讲解。例子的概率图模型如下所示：

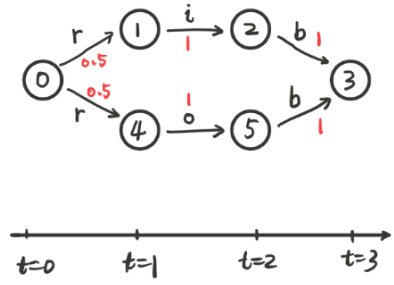

图 8: John Lafferty 论文中例子

在此例中，我们将 $0-1-2-3-4-5$ 这些看成是 tag 标准，将 r,i,b 看成是观测。

1. 假设 $t=1$ 时刻，我们观察到的是 0，0 时刻在观测到 r 的情况下转移到 1 和 4 状态的转移概率都一样。

2. 因为局部归一化的过程，我们发现从 1 到 2 的状态转移概率变成了 1，和输入的变量根本就没有关系了。这里的路只有一条，我们假设 $1 \to y_{t-1}$，$2 \to y_t$，$i \to r_t$，我们看到这里输入根本就不起作用了，不 care 输入是什么，概率都是 1，这就是问题的所在：

$$P(2|1,i) = 1 = P(2|1)$$
$$P(5|4,o) = 1 = P(5|4)$$

可以看出从 1 转移到 2 根本没有关注 observation。

3. 我们这里举一个例子，“小明爱中国!”，这里“小明”是名词，“爱”是动词，正常的思路是看到“中国”然后标名词。而局部归一化就是不考虑 observation，不看中国了，直接根据历史的结果得出中国的词性，这显然会出问题。

4. 我们知道图模型的参数结构都是通过训练得来的。我们假设有四个样本，3 个 rob，1 个 rib。

在 training 的过程中，我们可以得到 $P(1|0,r)=0.25$，$P(4|0,r)=0.75$。

我们使用 Viterbi 算法进行 Decoding 操作，也就是在观察序列的情况下求最大隐状态序列，$Y=(y_1,y_2,y_3)$。

$$\hat{y} = \arg\max_{y_1,y_2,y_3}(y_1,y_2,y_3|rib) = \max\{0.75\times 1\times 1, 0.25\times 1\times 1,\}$$

这样我们求出的，在观测为 rib 的情况下，隐状态链反而为 $0-4-5-3$，这显然是不合适的，没有关注输入变量的影响，之间就做出了判断。

我们举的例子是一个极端，这里的局部分布的熵直接等于 0 了。在这个例子中局部归一化使得系统不关注 observation 直接做出了判断。这是因为概率分布的熵太低了，直接等于 0 了，不再关注输入是什么。熵太低了就会影响模型的泛函能力，对于无法确定的信息过多的使用历史信息。**较好的模型应该，对于知道的部分，要尽可能去靠近它；对于不知道的部分，要保持它的结果等可能的出现的可能性，就是最大限度的保留不确定性，也就等价于模型的熵越大**。

在我们研究的这个问题中，可以描述为**Conditional Distribution with a low entropy take less notice of observation**。

4.2 小结

MEMM 的致命缺陷就是局部归一化，由于这个特性，会导致模型不 care observation 的内容来直接得到结果。其实，就是局部分布的熵太低了，对于不确定的部分，没有保留更多的可能性。

那么，打破的方法就是将各个 y 节点之间的连接从有向图变成无向图。无向图局部就不是一个分布了，就不需要进行局部归一化，而是进行全局归一化了，整条链是一个分布，这里后面后详细讲解。从而提高模型之间的熵，而有向图变成了无向图就自然打破了齐次马尔可夫假设。这是我们得到了一个 chain-structure CRF，所有的 Y 节点就构成了一个随机场。

我们看到从 HMM→MEMM→CRF，这两步过程就是先破坏了观测独立假设，再破坏了齐次马尔可夫假设。去掉了这两个不太好的假设，使原始数据得到了很好的利用。

5 CRF 概率密度函数的参数形式

前面我们花了大量的功夫来将 CRF 是如何演变来的，讲述了从 HMM-MEMM-CRF 的演化过程，理性的讲述了 CRF 图结构的合理性。所谓条件随机场，我们分成两个部分来进行解释：条件指的是，条件概率；随机场指的是，y 节点连接而成的无向图模型，称之为 Markov Field。CRF 的概率图模型如下所示：

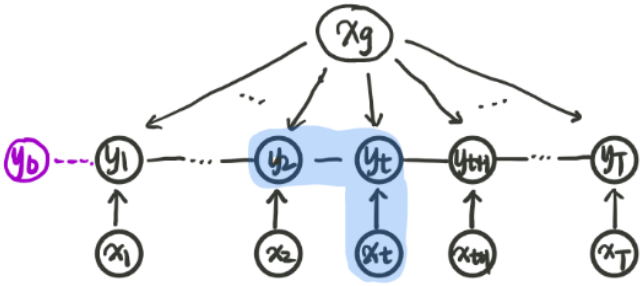

图 9: CRF 的概率图模型

5.1 势函数化简

我们想要得出 $P(Y|X)$ 的形式，很可惜在无向图中，我们并不能根据因果关系直接写出来。我们之间讲到过无向图的分解方法，这里再重申一下团的概念，**一个图中所有节点之间都相互相连称之为团**。对于 Markov Random Field 做因子分解，对于 $x \in \mathbb{R}^p$，我们可以令：

$$P(X) = \frac{1}{Z}\prod_{i=1}^{k}\phi_i(x_{c_i}) = \frac{1}{Z}\prod_{i=1}^{k}\exp\{-E_i(x_{c_i})\}$$

这里的 K 表示有 k 个最大团，c_i 指的是第 i 个最大团，而 $\phi_i(x_{c_i})$ 表示第 i 个最大团的势函数。$\phi_i(x_{c_i})$：势函数，必须为正。这里的概念都是来自于统计物理学和热力学的过程。因为，势函数必定为正，我们可以将势函数表达为 $\phi_i(x_{c_i}) = \exp\{-E_i(x_{c_i})\}$。其中，$E_i(x_{c_i})$ 称为 Energy function。实际上用这种形式表达的 $p(x)$，为 Gibbs Distribution，或者又被称之为 Boltzman Distribution。为了进一步简化，我们令 $-E_i(x_{c_i}) = F_i(x_{c_i})$，所以就得到了最终的形式：

$$P(X) = \frac{1}{Z}\prod_{i=1}^{k}\exp\{F_i(x_{c_i})\} \tag{10}$$

5.2 CRF 概率参数结构

如果不加说明，我们认为 CRF 为线性链。

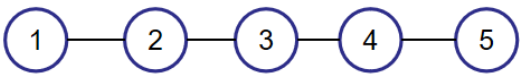

图 10: 线性链模型

我们可以看到在线性链中，每两个相邻的节点之间形成了一个团，而且总有团的机构都是一样的，图 10 所例的线性链可以被我们分成 4 个团。每个团都和 y_t，y_{t-1} 和 $x_{1:T}$ 有关系，所以，

$$P(X) = \frac{1}{Z}\prod_{i=1}^{k}\exp\{F_i(x_{c_i})\} = \frac{1}{Z}\exp\sum_{i=1}^{T-1}F_t(y_{t-1}, y_t, x_{1:T}) \tag{11}$$

这个公式的转换，首先将连乘写到指数函数里变成了连加，又因为线性链所有团的结构都是一样的。所以相当于从第 1 个加到第 $T-1$ 个。由于线性链中所有团的结构都是一样的，我们只考虑其中的一个团结构。那么，下一步的目标则是考虑，$F_t(y_{t-1}, y_t, x_{1:T})$ 怎么表示?

这个结构的得来实际上有一定的 intuitive，我们将它分成两部分，状态函数和转移函数。状态函数包括 y_t 时刻和 y_{t-1} 时刻和 $x_{1:T}$ 之间的影响，转移函数包括 y_t, y_{t-1} 时刻共同和 $x_{1:T}$ 之间的影响。公式表达如下所示：

$$F_t(y_{t-1}, y_t, x_{1:T}) = \underbrace{\triangle_{y_{t-1},x_{1:T}} + \triangle_{y_t,x_{1:T}}}_{\text{State Function}} + \underbrace{\triangle_{y_{t-1},y_t,x_{1:T}}}_{\text{Transfer Function}} \tag{12}$$

这里再次提醒一下，我们代入到标注问题中，$x_{1:T}$ 为 “小明是中国人。” 这是我们观测到的值，y_{t-1} 和 y_t 代表隐变量为词语的词性，名词，动词和形容词等。状态函数中 y_t 对 t 时刻的影响已经包含了 y_{t-1} 时刻的影响，我们没有必要再计算两遍，所以在状态函数中我们可以忽略掉 y_{t-1}。所以，最终结果为;

$$F_t(y_{t-1}, y_t, x_{1:T}) = \triangle_{y_t,x_{1:T}} + \triangle_{y_{t-1},y_t,x_{1:T}} \tag{13}$$

我们用一个函数来表达 $\triangle_{y_t,x_{1:T}}$：

$$\triangle_{y_t,x_{1:T}} = \sum_{l=1}^{L}\eta_l g_l(y_t, x_{1:T}) \tag{14}$$

其中 f_k 为特征函数，指示函数，满足某个状态为 1，比如：

$$f_k(y_{t-1} = \text{noun}, y_t = \text{verb}, x_{1:T}) = 1$$

$$f_k(y_{t-1} = \text{noun}, y_t = \text{auxiliary}, x_{1:T}) = 0$$

而 λ_k 则是我们需要学习的参数，利用数据通过 Learning 来得到。同样，我们可以用类似的思路来定义 $\triangle_{y_{t-1},y_t,x_{1:T}}$：

$$\triangle_{y_{t-1},y_t,x_{1:T}} = \sum_{k=1}^{K}\lambda_k f_k(y_{t-1}, y_t, x_{1:T})$$

这里的 f_k 和 g_l 都是给定的特征函数，λ_k 和 η_l 都是需要学习的参数。所以，我们最后就得到了 CRF 条件概率的表达形式为：

$$P(Y|X) = \frac{1}{Z}\exp\left\{\sum_{t=1}^{T}\left[\sum_{k=1}^{K}\lambda_k f_k(y_{t-1}, y_t, x_{1:T}) + \sum_{l=1}^{L}\eta_l g_l(y_t, x_{1:T})\right]\right\} \tag{15}$$

这就是 CRF 的概率密度函数，实际上这个得出的过程是比较 Intuitive 的，可能大家有点不太好理解。那么，我们来做一个类比，对于学习机器的同学来说，肯定对神经网络非常的熟悉。对于图像识别问题，我们输入一张图片，通过一个网络，得到一个结果。对于这个网络，我们选择怎样的网络结构，是前馈神经网络，CNN 还是 RNN，什么样的激活函数，和怎样的激活函数都是一个结构，然后根据我们的结果使用梯度下降来训练得到相应的参数。实际上这里和神经网络的结构是一个的，就像神经网络为什么设置成那个样子，实际上也不能说的很明白，就是这样的结构会 make sence 而已。在这里也一样，我的设计出的概率密度函数形式，就是感觉符合 CRF 的网络结构，并且会 make sence，当然小伙伴有更好的想法，也可以改进这个参数结果。得到结构后，我们就可以通过训练来求得参数了。

5.3 小结

在这一小节中，我们比较 intuitive 的得到了 CRF 概率密度函数的参数形式，和神经网络一样，这都是我们为了实现目的根据概率图的结构而设计出的一种会 make sence 的结构。结构确定之后，通过对数据的训练来求得参数，那么 CRF 模型就建立完成了。

6 概率密度函数的向量形式

上一节中，我们讲到了可以将每一个最大团分成两个部分，即为状态函数和转移函数。

$$\begin{aligned} P(Y|X) =& \frac{1}{Z}\exp\sum_{t=1}^{T}[\triangle_{\text{transfer function}} + \triangle_{\text{state function}}] \\ =& \frac{1}{Z}\exp\left\{\sum_{t=1}^{T}\left[\sum_{k=1}^{K}\lambda_k f_k(y_{t-1}, y_t, x_{1:T}) + \sum_{l=1}^{L}\eta_l g_l(y_t, x_{1:T})\right]\right\} \end{aligned} \tag{16}$$

这里的 K 和 L 分别是多少呢？加入对于 $y_t \in S = \{noun, verb, aux, \cdots\}$ 一共有 S 个状态。我们看看转移图像：

图 11: y_{t-1} 和 $y-t$ 状态之间的转移

所以，我们看到最多有 $|S|^2$ 种可能，所以 $\mathrm{K} \leq |S|^2$。对于归一化因子 Z 而言，肯定是和 y 没有关系的，归一化因子是对 y 进行了积分的。

6.1 概率密度函数的向量形式化简

所以，概率模型被我们写为：

$$P(Y = y|X = x) = \frac{1}{Z(x, \lambda, \eta)}\exp\left\{\sum_{t=1}^{T}\left[\sum_{k=1}^{K}\lambda_k f_k(y_{t-1}, y_t, x_{1:T}) + \sum_{l=1}^{L}\eta_l g_l(y_t, x_{1:T})\right]\right\} \tag{17}$$

令：

$$\lambda = \begin{bmatrix}\lambda_1 \\ \lambda_2 \\ \vdots \\ \lambda_k\end{bmatrix}, \quad \eta = \begin{bmatrix}\eta_1 \\ \eta_2 \\ \vdots \\ \eta_l\end{bmatrix}, \quad f = \begin{bmatrix}f_1 \\ f_2 \\ \vdots \\ f_k\end{bmatrix} = f(y_{t-1}, y_t, x_{1:T}), \quad g = \begin{bmatrix}g_1 \\ g_2 \\ \vdots \\ g_l\end{bmatrix} = g(y_t, x_{1:T}) \tag{18}$$

根据向量的内积方法,我们可以将 $\sum_{k=1}^{K}\lambda_k f_k(y_{t-1},y_t,x_{1:T})$ 改写为 $\lambda^T f(y_{t-1},y_t,x_{1:T})$,$\sum_{l=1}^{L}\eta_l g_l(y_t,x_{1:T})$ 改写为 $\eta^T g(y_t,x_{1:T})$。所以，原式等于:

$$\begin{aligned}P(Y=y|X=x)=&\frac{1}{Z(x,\lambda,\eta)}\exp\left\{\sum_{t=1}^{T}\left[\lambda^T f(y_{t-1},y_t,x_{1:T})+\eta^T g(y_t,x_{1:T}))\right]\right\}\\=&\frac{1}{Z(x,\lambda,\eta)}\exp\left\{\left[\lambda^T\sum_{t=1}^{T}f(y_{t-1},y_t,x_{1:T})+\eta^T\sum_{t=1}^{T}g(y_t,x_{1:T}))\right]\right\}\end{aligned}\tag{19}$$

我们现在看着还是觉得太复杂还想进行下一步化简，这里的 λ 和 $\sum_{t=1}^{T}f(y_{t-1},y_t,x_{1:T})$ 都是 $k\times 1$ 的向量，η 和 $\sum_{t=1}^{T}g(y_t,x_{1:T})$ 都是 $l\times 1$ 的向量，相加是在同一个维度相加，并不会破坏矩阵的尺寸。所以我们可以进行合并。于是令:

$$\theta=\begin{bmatrix}\lambda\\ \eta\end{bmatrix}_{(k+l)\times 1},\qquad H=\begin{bmatrix}\sum_{t=1}^{T}f(y_{t-1},y_t,x_{1:T})\\ \sum_{t=1}^{T}g(y_t,x_{1:T}))\end{bmatrix}_{(k+l)\times 1},\tag{20}$$

所以，我们得到的形式为:

$$P(Y=y|X=x)=\frac{1}{Z(x,\theta)}\exp\left\{\theta^T\cdot H(y_t,y_{t-1},x_{1:T})\right\}\tag{21}$$

$H(y_t,y_{t-1},x_{1:T})$ 表示 H 矩阵和 $y_t,y_{t-1},x_{1:T}$ 有关。我们可以将 $\theta^T\cdot H(y_t,y_{t-1},x_{1:T}$ 写成向量内积的形式，最终得到最后的形式为:

$$P(Y=y|X=x)=\frac{1}{Z(x,\theta)}\exp\langle\theta,H\rangle\tag{22}$$

6.2 小结

我们最终得到了概率密度函数的向量表达形式为:

$$P(Y=y|X=x)=\frac{1}{Z(x,\theta)}\exp\langle\theta,H\rangle$$

实际上这章非常的简单，都是一些很简单的数学变形而已。我们这样做的目的是什么呢? 很简单就是去掉 $\sum$ 符号，在 learning 和 Inference 问题中，我们可能遇到大量的求导，把这个 $\sum$ 去掉是为了简化运算，带上了 $\sum$ 不方便进行求导运算。

7 CRF 模型要解决的问题

上一节我们用向量形式来进行表达，为 Learning 和 Inference 问题做好了准备。所谓机器学习中需要解决的问题，大致被我们分为两类，通过 learning 来得到模型，并利用求得的模型对未知的数据进行 Inference。

7.1 Learning

也就是 Parameter estimation，利用数据来对模型中的参数进行求解。这么任务可以被描述为: 给定一个数据集 $\{(x^{(i)},y^{(i)})\}_{i=1}^{N}$，其中 $x,y\in\mathbb{R}^T$ 都是 T 维的。

我们可以来举一个简单的例子，比如 “小明爱中国”，这个数据的 x 由三个部分组成，分别是 “小明”，“爱” 和 “中国” 组成的，标注 y 则是 “名词”，“动词”，“名词” 组成的。在我们的 CRF 概率图中，不是一个节点是一个样本，而是所有的 y 节点合在一起才是一个样本，这里需要强调一下以防同学弄混。所有，我们最终的求解目标为：

$$\hat{\theta} = \arg\max \prod_{i=1}^{N} P(y^{(i)}|x^{(i)}) \tag{23}$$

7.2 Inference

Inference 通常可以被我们分为 Marginal Probability；Conditional Probability 和 MAP Inference: Decoding 三类问题。而 CRF 是判别模型，所以我们一般不谈 Conditional Probability，这个一般是针对生成模型中才会用到的，因为求联合概率，我们通常是使用贝叶斯公式来进行转换的。所以，我们这里不讨论 CRF 的 Conditional Probability 的问题。

7.3 Marginal Probability

在概率图模型中，我们的建模对象往往都是 join distribution。比如有一个建模对象为 $P(X)$，$X = (x_1, x_2, \cdots, x_p)^T$。我们要求 $P(x_1)$，往往都是先求 $P(X)$，然后再对其他变量进行积分来求得 $P(x_1)$。

举例，词性标注问题，对于 $P(y_1 = verb|X)$，也就是给定观察序列，求第 i 个词是动词的概率。

7.4 MAP Inference: Decoding

Decoding 问题，也就是在给定观察序列的情况下，寻找最大概率可能出现的隐概率状态序列。求解的是：

$$\hat{Y} = \arg \max_{Y=y_1,\cdots,y_t} P(Y|X) \tag{24}$$

7.5 小结

下面我们主要针对 CRF 的三个问题进行求解，分别是 1. Marginal Probability；2. Parameter Estimation；3. Decoding Problem。

8 Marginal Probability Calculate

在描述这个问题之前，先假设参数是已知的，如果参数是未知的，那么就假设参数 θ 是一个常量。我们的求解目标是，**给定** $P(Y = y|X = x)$ **的情况下，求** $P(y_t = i|x)$ **的概率**，这里再强调一下，每一个 $\{x, y\}$ 都是一个时序的样本，比如一个句子，每个样本可以根据时间分成很多段。

8.1 直接进行求解

概率密度函数被定义为：

$$P(Y = y|X = x) = \frac{1}{Z} \prod_{t=1}^{T} \phi_t(y_{t-1}, y_t, x) \tag{25}$$

实际上就等于最大团的势函数之积。我们要求某个时刻在已知所有的观察变量的情况下隐变量 $y_t = i$ 的概率，实际上就是将 $P(Y = y|X = x)$ 这个联合概率分布在 y 的各个维度上积分，就可以得到 y_t 时刻的条件概率了，而这里都是离散变量，那么就是求和。这个思路和想法很简单，我们来看看它怎么样?

$$P(y_t = i|x) = \sum_{y_1,\cdots,y_{t-1},y_{t+1},\cdots,y_T} P(y|x) = \sum_{y_{1:t-1}} \sum_{y_{t+1:T}} \frac{1}{Z} \prod_{t'=1}^{T} \phi_{t'}(y_{t'-1}, y_{t'}, x) \tag{26}$$

由于我们求的是一个概率分布，每一个 $y_t \in \mathcal{S}$，我们一共要对 $T-1$ 个 y 进行积分，每一个 y 有 $|S|$ 种可能，所以在这个求积分的过程中，算法的复杂度是 $\mathcal{O}(|S|^T)$ 的。**大家都是学过高等数学的，如果你理解不了，这就是一个 T-1 重积分，T-1 重积分是 T-1 维空间中的体积，因为所有势函数的乘积，所以每次都需要考虑概率图中所有的依赖关系，计算需要进行 $|S|^T$ 次划分**。而在求势函数的积的过程中，需要进行 T 次运算，所以这个算法最后的复杂度是 $\mathcal{O}(T \cdot |S|^T)$。这个算法的复杂度实在是太高了，我们根本就没有办法求解，这个想法确实很简单，但是很可惜是 intractable 的。

那么我们有没有什么办法来简化运算呢?

8.2 Variables Elimination 算法复杂度降低分析

这个算法我们之前在 HMM 中是讲过的，在这个问题中我们怎么求解呢? 上述直接进行求解的办法，**因为每一次积分都是对所有的势函数乘积进行积分，所以需要考虑所有的状态，算法复杂度是 $\mathcal{O}(|S|^T)$，这是暴力求解的思路**。

我们发现，在对某一个时刻的变量 y_i 进行求和的时候，和很多的势函数都没有关系。比如在 $t = 5$ 时刻，求和只和势函数 $\phi(y_4, y_5, x)$ 有关。所以，我们可以利用动态规划的思路，将连加号分解开，移到后面后具体相关的势函数联系起来，这样每次积分时只考虑和势函数相关的变量，不需要考虑所有的变量。**设 m 为单个条件概率的最大变量数量，我们化简后，只有依赖最多的这个节点会被考虑 m 次，其他的都会少于 m 次，所以算法复杂度被就降低到了 $\mathcal{O}(|S|^m)$，而 $m << T$**。这就是我们思路的大致来源。

8.3 Variables Elimination 算法运算过程

由于在第 t 时刻，y_t 的状态是已知的。所以，我们将序列从 y_t 时刻分开，分成两个部分，如下图所示:

图 12: 序列分割图

$$P(y_t = i|x) = \sum_{y_{1:t-1}} \sum_{y_{t+1:T}} \frac{1}{Z} \prod_{t'=1}^{T} \phi_{t'}(y_{t'-1}, y_{t'}, x) = \frac{1}{Z} \triangle_{\text{left}} \cdot \triangle_{\text{right}} \tag{27}$$

其中：

$$\begin{aligned}\triangle_{\text{left}} &= \sum_{y_{1:t-1}} \phi_1(y_0, y_1, x)\phi_2(y_1, y_2, x)\cdots\phi_{t-1}(y_{t-2}, y_{t-1}, x)\phi_t(y_{t-1}, y_t = i, x) \\ \triangle_{\text{right}} &= \sum_{y_{t+1:T}} \phi_{t+1}(y_t = i, y_{t+1}, x)\phi_{t+2}(y_{t+1}, y_{t+2}, x)\cdots\phi_T(y_{T-1}, y_T, x)\end{aligned} \tag{28}$$

下一步目标就是想怎么去拆分了，首先来看看每一个势函数究竟和哪些变量有关，示意图如下所示：

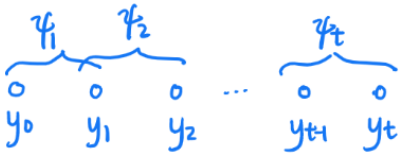

图 13: 势函数变量关系图

变量消除的过程就是积分的过程，我们需要将累加的变量进行分解，让求和过程只和相关的势函数进行求和，这样分解避免多个势函数混着一起积分，来减少运算过程，因为这样每次积分我们都只要考虑单个变量的依赖变量的关系。根据上图所示的，关系依赖图，我们进行如下划分：

$$\begin{aligned}\triangle_{\text{left}} &= \sum_{y_{1:t-1}} \phi_1(y_0, y_1, x)\phi_2(y_1, y_2, x)\cdots\phi_{t-1}(y_{t-2}, y_{t-1}, x)\phi_t(y_{t-1}, y_t = i, x) \\ &= \sum_{y_{t-1}} \phi_t(y_{t-1}, y_t = i, x)\left(\sum_{y_{t-2}} \phi_{t-1}(y_{t-2}, y_{t-1}, x)\left(\cdots\left(\sum_{y_1} \phi_2(y_1, y_2, x)\left(\sum_{y_0} \phi_1(y_0, y_1, x)\right)\right)\right)\right)\end{aligned} \tag{29}$$

那么这样我们手动的把所有的 $\sum$ 连加符号，一个个的嵌入到该去的地方，计算复杂度过高的问题已经解决了。实际上写成公式 (29) 这样的样子就可以了。我们为了进一步简化表达作如下操作。

我们令：

$$\alpha_t(i) = \sum_{y_{t-1}} \phi_t(y_{t-1}, y_t = i, x)\left(\sum_{y_{t-2}} \phi_{t-1}(y_{t-2}, y_{t-1}, x)\left(\cdots\left(\sum_{y_1} \phi_2(y_1, y_2, x)\left(\sum_{y_0} \phi_1(y_0, y_1, x)\right)\right)\right)\right)$$

$$\alpha_{t-1}(j) = \sum_{y_{t-2}} \phi_{t-1}(y_{t-2}, y_{t-1} = j, x)\left(\cdots\left(\sum_{y_1} \phi_2(y_1, y_2, x)\left(\sum_{y_0} \phi_1(y_0, y_1, x)\right)\right)\right)$$

那么，很显然我们就可以得出 α 函数之间的表达式：

$$\alpha_t(i) = \sum_{y_{t-1}} \phi_t(y_{t-1} = j, y_t = i, x)\alpha_{t-1}(j) \tag{30}$$

因为，$y_{t-1} = j$，所以我们可以得到等价表达方式：

$$\alpha_t(i) = \sum_{j \in S} \phi_t(y_{t-1} = j, y_t = i, x)\alpha_{t-1}(j) \tag{31}$$

实际上推导到了这里就很简单了，那么 $\triangle_{\text{left}} = \alpha_t(i)$。按照同样的方法进行变量提取化简，我们可以

得到一样的结论：

$$\begin{aligned}\triangle_{\text{right}} &= \sum_{y_{t+1:T}} \phi_{t+1}(y_t = i, y_{t+1}, x)\phi_{t+2}(y_{t+1}, y_{t+2}, x)\cdots\phi_T(y_{T-1}, y_T, x) \\ &= \sum_{y_T}\phi_T(y_{T-1}, y_T, x)\left(\sum_{y_{T-1}}\phi_{T-1}(y_{T-2}, y_{T-1}, x)\left(\cdots\left(\sum_{y_{t+2}}\phi_2(y_{t+1}, y_{t+2}, x)\left(\sum_{y_{t+1}}\phi_1(y_t = i, y_{t+1}, x)\right)\right)\right)\right) \\ &= \beta_t(i)\end{aligned} \tag{32}$$

所以，最终得到的最简形式为：

$$P(y_t = i|x) = \frac{1}{Z}\alpha_t(i)\cdot\beta_t(i) \tag{33}$$

8.4 小结

本小节我们探讨了给定 $P(Y = y|X = x)$ 的情况下，求 $P(y_t = i|x)$ 的概率，x 和 y 都是时序序列。直接采用积分求和的方法算法复杂度为 $\mathcal{O}(|S|^T)$ 的太高了，利用变量消元的方法，准确的分离变量和待求和的维度。**成功的将算法复杂度从 $\mathcal{O}(|S|^T)$ 降到 $\mathcal{O}(|S|^m)$，且 m 为单个条件概率的最大变量数量，$m << T$。**

9 CRF Learning Problem

在这个问题中，问题具体就是用极大似然估计来求 CRF 模型的参数，

$$\hat{\theta} = \arg\max_{\theta}\prod_{i=1}^{N} P(y^{(i)}|x^{(i)}) \tag{34}$$

N 为训练样本的数量。而 θ 实际上包含两个部分，即为 λ, η，所以，目标函数为：

$$\hat{\lambda}, \hat{\eta} = \arg\max_{\lambda,\eta}\prod_{i=1}^{N} P(y^{(i)}|x^{(i)}) \tag{35}$$

而我们，在之前求过了，

$$P(Y = y|X = x) = \frac{1}{Z(x, \lambda, \eta)}\exp\left\{\sum_{t=1}^{T}\left[\lambda^T f(y_{t-1}, y_t, x) + \eta^T g(y_t, x))\right]\right\} \tag{36}$$

所以：

$$\begin{aligned}\hat{\lambda}, \hat{\eta} &= \arg\max_{\lambda,\eta}\prod_{i=1}^{N} P(y^{(i)}|x^{(i)}) \\ &= \arg\max_{\lambda,\eta}\sum_{i=1}^{N}\log P(y^{(i)}|x^{(i)}) \\ &= \arg\max_{\lambda,\eta}\sum_{i=1}^{N}\left\{-\log Z(x^{(i)}, \lambda, \eta) + \sum_{t=1}^{T}\left[\lambda^T f(y_{t-1}^{(i)}, y_t^{(i)}, x) + \eta^T g(y_t^{(i)}, x^{(i)}))\right]\right\} \\ &= \arg\max_{\lambda,\eta} L(\lambda, \eta, x^{(i)})\end{aligned} \tag{37}$$

9.1 梯度上升法求解极大似然解

$$\nabla_\lambda L = \sum_{i=1}^{N} \left\{ \sum_{t=1}^{T} f(y_{t-1}^{(i)}, y_t^{(i)}, x^{(i)}) - \nabla_\lambda \log Z(x^{(i)}, \lambda, \eta) \right\} \tag{38}$$

现在问题就是 $\nabla_\lambda \log Z(x^{(i)}, \lambda, \eta)$ 怎么来求？注意观察公式 (36)，$P(Y = y|X = x)$ 实际上是一个指数族分布。而 $Z(x^{(i)}, \lambda, \eta)$ 就是 log partition function。我们在指数族分布的第三部分曾经推导出，log partition function 的导数为充分统计量的均值。所以，$\nabla_\lambda \log Z(x^{(i)}, \lambda, \eta)$ 我们可以直接写出来的。

$$\mathbb{E}\left[\sum_{i=1}^{T} f(y_{t-1}, y_t, x^{(i)}) \right]$$

注意这里的 y 是自变量而不是样本，所以：

$$\begin{aligned} \mathbb{E}\left[\sum_{i=1}^{T} f(y_{t-1}, y_t, x^{(i)}) \right] &= \sum_{y} P(y|x^{(i)}) \cdot \sum_{i=1}^{T} f(y_{t-1}, y_t, x^{(i)}) \\ &= \sum_{i=1}^{T} \left(\sum_{y} P(y|x^{(i)}) \cdot f(y_{t-1}, y_t, x^{(i)}) \right) \end{aligned} \tag{39}$$

因为在 $f(y_{t-1}, y_t, x^{(i)})$ 中，我们是知道具体的值的。所以，我们把其他维度的 y 都通过求和消掉。那么：

$$\begin{aligned} \sum_{i=1}^{T} \left(\sum_{y} P(y|x^{(i)}) \cdot f(y_{t-1}, y_t, x^{(i)}) \right) &= \sum_{i=1}^{T} \left(\sum_{y_{1:t-2}} \sum_{y_{t-1}} \sum_{y_t} \sum_{y_{t+1:T}} P(y|x^{(i)}) \cdot f(y_{t-1}, y_t, x^{(i)}) \right) \\ &= \sum_{i=1}^{T} \sum_{y_{t-1}} \sum_{y_t} \left(\sum_{y_{1:t-2}} \sum_{y_{t+1:T}} P(y|x^{(i)}) \cdot f(y_{t-1}, y_t, x^{(i)}) \right) \\ &= \sum_{i=1}^{T} \sum_{y_{t-1}} \sum_{y_t} \left(P(y_{t-1}, y_t|x^{(i)}) \cdot f(y_{t-1}, y_t, x^{(i)}) \right) \end{aligned} \tag{40}$$

那么，我们的下一步思路就是如何求解 $P(y_{t-1}, y_t|x^{(i)})$。这实际上就是一个求解边缘概率的问题。这其中 y_{t-1}, y_t 都是已知的。

$$\begin{aligned} P(y_{t-1} = j, y_t = i|x^{(i)}) &= \sum_{y_{1:t-2}} \sum_{t+1:T} \prod_{t'=1}^{T} \phi_{t'}(y_{t'-1}, y_{t'}, x) \\ &= \frac{1}{Z} \triangle_{\text{left}} \cdot \phi(y_t, y_{t-1}, x) \cdot \triangle_{\text{right}} \\ &= \frac{1}{Z} \alpha_{t-1}(j) \cdot \phi(y_t = i, y_{t-1} = j, x) \cdot \beta_t(i) \\ &= A(y_{t-1}, y_t) \end{aligned} \tag{41}$$

所以，最后我们得到的结论为：

$$\nabla_\lambda L = \sum_{i=1}^{N} \sum_{t=1}^{T} \left[f(y_{t-1}^{(i)}, y_t^{(i)}, x^{(i)}) - \sum_{y_{t-1}} \sum_{y_t} A(y_{t-1}, y_t) f(y_{t-1}, y_t, x) \right] \tag{42}$$

其中，$y_t^{(i)}$ 为样本，y_t 为自变量。那我们用类似的方法同样可以求解出：

$$\nabla_\eta L = \sum_{i=1}^{N}\sum_{t=1}^{T}\left[f(y_{t-1}^{(i)}, y_t^{(i)}, x^{(i)}) - \sum_{y_{t-1}}\sum_{y_t} A(y_{t-1}, y_t)g(y_t, x)\right] \tag{43}$$

最终，梯度上升算法将被我们总结为：

$$\begin{cases} \lambda^{t+1} = \lambda^t + \text{step} \cdot \nabla_\lambda L(\lambda^{(t)}, \eta^{(t)}) \\ \eta^{t+1} = \eta^t + \text{step} \cdot \nabla_\eta L(\lambda^{(t)}, \eta^{(t)}) \end{cases} \tag{44}$$

实际上，这只是一种理论上可行的求解方法。实际求解的过程中，这样做收敛速度太慢了，这里只是给出一种理论的解，还有很多类似的方法，有兴趣的同学可以深入研究。

9.2 MAP Inference

这个问题，仔细观察就知道和 HMM 中的 Decoding 问题是一样的。HMM 和 CRF 没有任何区别。我们想要在已知观测序列的情况下，求隐变量序列的最大的可能状态。实际上就是使用动态规划算法，一步步求得最大的序列，也就是 Viterbi 算法，在 Hidden Markov Model 的第四节，有详细的描述，这里就不多说。

9.3 小结

这一节中，我们介绍了用极大似然估计算法来求 CRF 模型的参数。实际上计算过程都还挺普通的。有一个难点是利用**指数族函数 log partition function 的导数为充分统计量的均值**这个结论来进行求解。实际上这一步明白之后，还是比较简单的。而其中也遇到了求边缘概率的方法，我们用到了之前描述的本章第 8 节求 Marginal Probability 的方法。MAP Inference 和 HMM 中的 Decoding 问题是一样的，使用 Viterbi 算法求解。

10 Conclusion

本章的思路实际是很流畅的，首先介绍了整个机器学习体系结构。然后，引出了概率图模型在其中的地位。从最简单的概率图模型，Naive Bayes 算法，一步步的改进缺点到 HMM，到 MEMM，最后演化成了 CRF。我个人感觉这个过程是非常重要的，很多教科书一上来就摆了一推的概率和定义，我等凡人真的头疼。非常感谢 B 站白板推导系列的 up 主。而之后就，定义了 CRF 的模型结构，定义完了之后就是如何使用数据来求解这个 CRF 模型的参数，求解出来之后就是如何用这个模型来进行 inference，整个过程的思维是非常流畅和附有逻辑性的。

Gaussian Network

Chen Gong

26 February 2020

目录

这小节，我们主要讲解的是高斯网络的结构背景，模型的基本概念。根据有向图和无向图，我们可以将高斯网络分成高斯贝叶斯网络和高斯马尔可夫网络。

1 Background

概率图模型 (Probability Graphic Model)，我们之前学习的是贝叶斯网络和马尔可夫随机场，之前学习的概率图中每个节点都是离散随机变量。所以根据图是有向图还是无向图，我们可以将概率图模型分成贝叶斯网络 (Bayesian Network) 和马尔可夫随机场 (Markov Random Field)。

而如果概率图中每个节点都是一维连续随机变量，根据图是有向图还是无向图，可以被分为高斯贝叶斯网络 (Gaussian Bayesian Network) 和高斯马尔可夫随机场 (Gaussian Markov Random Field)。

1.1 高斯网络概念

假设高斯马尔可夫随机场的概率图模型如下所示：

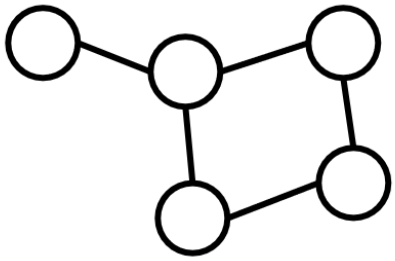

图 1: 高斯马尔可夫随机场的概率图模型

每一个节点都是一个一维随机变量，每一个节点 $x_i \sim \mathcal{N}(\mu_i, \Sigma_i)$，而所有的节点构成的集合：$\mathcal{X} = (x_1, x_2, \cdots, x_p)^T$。多维变量的高斯分布为：

$$P(X) = \frac{1}{(2\pi)^{\frac{p}{2}}|\Sigma|^{\frac{1}{2}}} \exp\left\{-\frac{1}{2}(X-\mu)^T\Sigma^{-1}(X-\mu)\right\} \tag{1}$$

所以，**一个高斯网络就可以映射为一个高维高斯分布**。而协方差矩阵 Σ 实际就反映了变量之间的联系。由于高斯分布的自共轭性，单个变量和整个高斯网络都是符合高斯分布的。

协方差矩阵为：

$$\Sigma = (\sigma_{ij}) = \begin{bmatrix} \sigma_{11} & \sigma_{12} & \cdots & \sigma_{1p} \\ \sigma_{21} & \sigma_{22} & \cdots & \sigma_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_{p1} & \sigma_{p2} & \cdots & \sigma_{pp} \end{bmatrix}_{p\times p} \tag{2}$$

而协方差矩阵描述的就是两个变量之间关系，当 $\sigma_{ij} = 0$，就等价于 $x_i \perp x_j$。整个是绝对独立或者称之为边缘独立，而在我们概率图模型中往往研究的是条件独立，$x_A \perp x_B | x_C$(x_A, x_B, x_C 都是节点的集合)。我们研究条件独立性实际上是为了简化计算，因为完全图的复杂度实在是太高了。

我们定义信息矩阵 (Information Matrix) 或者也被称为精度矩阵 (Precision Matrix) 为：

$$\Lambda = \Sigma^{-1} = \begin{bmatrix} \lambda_{11} & \lambda_{12} & \cdots & \lambda_{1p} \\ \lambda_{21} & \lambda_{22} & \cdots & \lambda_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ \lambda_{p1} & \lambda_{p2} & \cdots & \lambda_{pp} \end{bmatrix}_{p \times p} \tag{3}$$

当 $\lambda_{ij} = 0$ 就意味着 $x_i \perp x_j | - \{x_i, x_j\}$ 这就可以用来表示条件概率了。这也是信息矩阵的微妙之处，至于为什么我们相信大部分同学都是一脸懵逼，这里会在后面进行描述。

1.2 小结

Gaussian Network 是连续型的概率图模型，一个高斯网络实际上可以看成是一个高维高斯分布。协方差矩阵中可以反映完全独立性，而精度矩阵中可以反映条件独立性，也就是在其他所有节点已知的情况下，两个节点之间的独立性。下面我们要分别介绍 Gaussian Bayesian Network 和 Gaussian Markov Random Field。

2 Gaussian Bayesian Network

在连续型的 Probability Graphic Model 中，有向图，被我们称之为 Gaussian Bayesian Network。假设概率图中一共有 p 个节点，根据我们之前学习的贝叶斯网络的因子分析法，可以得到：

$$P(X) = \prod_{i=1}^{p} P(x_i | x_{\text{pa}(i)}) \tag{4}$$

$\text{pa}(i)$ 是一个集合，代表 x_i 节点的父亲节点集合。

GBN is based on Linear Gaussian Model。GBN 是一个 global 的概念，代表的是整个高斯网络，也就是 X 之间的高维高斯分布。而 Linear Gaussian Model 指的是 local 的模型，也就是局部父亲节点与孩子节点之间的关系是符合高斯线性模型的。

我们看看标准的线性高斯模型：

$$\begin{cases} P(x) = \mathcal{N}(x | \mu_x, \Sigma_x) \\ P(y|x) = \mathcal{N}(y | Ax + b, \Sigma_y) \end{cases} \tag{5}$$

线性就体现在了 y 与 x 之间的关系。

2.1 Kalman Filter 回顾

我们以 HMM 为例，概率图模型如下所示：

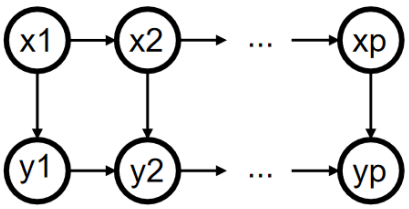

图 2: Kalman Filter 的概率图模型

Kalman Filter 是用来解决 HMM 问题中的 Filter 问题，也被称之为线性高斯系统，线性主要体现在下面两个地方：

$$\begin{cases} P(x_t|x_{t-1}) = \mathcal{N}(x_t|Ax_{t-1}+B, Q) \\ P(y_t|x_t) = \mathcal{N}(y_t|Cx_t+D, R) \end{cases} \tag{6}$$

变量之间的线性关系即为：

$$\begin{cases} x_t = Ax_{t-1}+B+\epsilon \quad \epsilon \sim \mathcal{N}(0,Q) \\ y_t = Cx_t+D+\delta \qquad \delta \sim \mathcal{N}(0,R) \end{cases} \tag{7}$$

这就是 Kalman Filter 的 Representation Model，也就是一种特殊的 Gaussian Bayesian Model。

2.2 Gaussian Bayesian Model 详解

下面我以如下所示的 Gaussian Bayesian Model 为例：

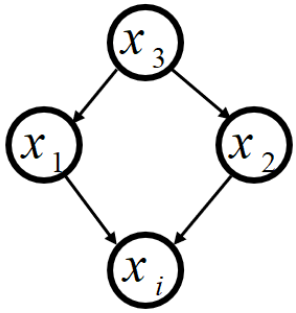

图 3: Gaussian Bayesian Model 的概率图模型

很显然 $x_{\mathrm{pa}(i)} = \{x_1, x_2\}$。根据因子分解法，我们可以得到：

$$P(X) = \prod_{i=1}^{p} P(x_i|x_{\mathrm{pa}(i)}) \tag{8}$$

我们将 $x_{\mathrm{pa}(i)}$ 写成向量形式，所以 $x_{\mathrm{pa}(i)} = (x_1, x_2, \cdots, x_k)$。根据父子节点之间的线性关系，我们可以得到：

$$P(x_i|x_{\mathrm{pa}(i)}) = \mathcal{N}(x_i|\mu_i + w_i^T x_{\mathrm{pa}(i)}, \sigma_i^2) \tag{9}$$

其中，x_i 是一维变量，实际上我们可以看**成** x_i **就是它的父亲节点的线性组合**，所以说 Guassian Network 是基于局部模型，局部模式是一个 Linear Gaussian Model，为了清楚表示可以写为：

$$x_i = \mu_i + \sum_{j \in x_{\mathrm{pa}(i)}} w_{ij} \cdot (x_j - \mu_j) + \sigma_i \cdot \xi_i \tag{10}$$

这里有的小伙伴可能会有点懵逼了，实际上我们将这个 x_j 写成 $(x_j - \mu_j)$，是为了使其进行归一化，使均值等于 0，以 0 为中心，便于对 learning 的运算。而 $\xi_i \sim \mathcal{N}(0,1)$，我们知道 $\sigma_i \cdot \xi_i$ 的方差还是为 σ^2，而乘上一个随机变量是使 x_i 变成一个随机变量。而为了统一结构，我们将其写为：

$$x_i - \mu_i = \sum_{j \in x_{\mathrm{pa}(i)}} w_{ij} \cdot (x_j - \mu_j) + \sigma_i \cdot \xi_i \tag{11}$$

很显然，这就是一种线性组合。从全局角度看就是 Gaussian Network，从局部角度来看就是一个 Linear Gaussian Model。

2.3 Gaussian Bayesian Model 的矩阵表达形式

假如，高斯网络中有 p 个节点，各个节点变量组成的集合为：

$$X = (x_1, x_2, \cdots, x_p)^T$$

相应每个节点的均值为：

$$\mu = (\mu_1, \mu_2, \cdots, \mu_p)^T$$

每个节点对应的一个 $\mathcal{N}(0,1)$ 的随机变量为：

$$\xi = (\xi_1, \xi_2, \cdots, \xi_p)^T$$

节点与节点之间的权重组成了一个矩阵为：

$$W = [w_{ij}]_{p\times p}$$

而 S 为 σ_i 组成一个对角矩阵：

$$S = \mathrm{diag}(\sigma_i)_{p\times p}$$

所以，我们就可以得到 Gaussian Linear Representation 的向量表达形式为：

$$X - \mu = W(X - \mu) + S \cdot \xi \tag{12}$$

化简就可以得到：

$$X - \mu = (I - W)^{-1} S \cdot \xi \tag{13}$$

当然我们都假设这里都是可逆的。因为 $X \sim \mathcal{N}(\mu, \Sigma)$，我们把变量看成 $X - \mu$，所以重点要计算的就是 Σ 了。

$$\Sigma = \mathrm{cov}(X) = \mathrm{cov}(X - \mu) = \mathrm{cov}((I - W)^{-1} S \cdot \xi)$$

$$\mathrm{cov}((I - W)^{-1} S \cdot \xi) = ((I - W)^{-1} S)^T ((I - W)^{-1} S)\mathrm{cov}(\xi) = ((I - W)^{-1} S)^T ((I - W)^{-1} S)$$

因为这里的 $(I - W)^{-1} S$ 是一个确定的常数。

2.4 小结

在这一小节中，我们主要研究了连续随机变量的有向图，为 Gaussian Bayesian Model，我们主要研究的是模型的表达形式，包括向量表达形式。最后，老师在求协方差矩阵的时候，没有求完，我觉得比较简单就顺手写了。**高斯贝叶斯模型，全局是多维高斯分布，局部是线性高斯模型**。

3 Gaussian Markov Random Field

这小节我们要介绍的连续变量的无向图结构，高斯马尔可夫随机场 (Gaussian Markov Random Field)。多维高斯分布的概率密度函数如下所示：

$$P(X) = \frac{1}{(2\pi)^{\frac{p}{2}} |\Sigma|^{\frac{1}{2}}} \exp\left\{ -\frac{1}{2} (X - \mu)^T \Sigma^{-1} (X - \mu) \right\} \tag{14}$$

假设现有一个 Gaussian Markov Random Field 如下图所示：

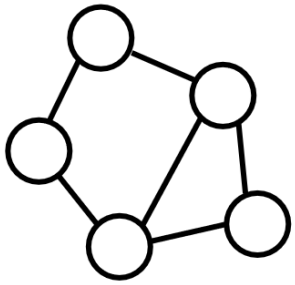

图 4: Gaussian Markov Random Field 的概率图模型

3.1 Gaussian Markov Random Field 的概率计算

之前，我们使用了最大团的势函数的乘积来计算随机变量的概率。这里成对 Markov 性质，我们使用另一种表达方式，更适合对此问题的分析。我们假设模型中有 p 个节点，那么概率表达如下所示：

$$P(X) = \frac{1}{Z}\prod_{i=1}^{p}\phi(x_i)\cdot\prod_{i,j\in X}\phi_{ij}(x_i,x_j) \tag{15}$$

其中 $\phi(x_i)$ 表示的是一个点的势函数，被称为 node potential；而 $\phi_{ij}(x_i,x_j)$ 表示的是一条边的势函数，被称为 edge potential。令 $\Sigma^{-1}=\Lambda$，下面我来计算 $P(X)$ 的表达形式，为了清晰的分析，我们只考虑和 X 相关的部分。

$$\begin{aligned} P(X) &\propto \exp\left\{-\frac{1}{2}(X-\mu)^T\Sigma^{-1}(X-\mu)\right\} \\ &= \exp\left\{-\frac{1}{2}(X^T\Lambda-\mu^T\Lambda)(X-\mu)\right\} \\ &= \exp\left\{-\frac{1}{2}(X^T\Lambda X-X^T\Lambda\mu-\mu^T\Lambda X+\mu^T\Lambda\mu\right\} \end{aligned} \tag{16}$$

其中，X 和 μ 是 $p\times 1$ 的矩阵，Λ 是 $p\times p$ 的矩阵。所以，$X^T\Lambda\mu$ 和 $\mu^T\Lambda X$ 都是一维实数，且 $X^T\Lambda\mu=\mu^T\Lambda X$(这还无法理解就自己拆开算一下吧)。

$$P(X)\propto\exp\left\{-\frac{1}{2}(X^T\Lambda X-2\mu^T\Lambda X+\mu^T\Lambda\mu)\right\}$$

最后一项 $\mu^T\Lambda\mu$ 与 X 无关，所以

$$P(X)\propto\exp\left\{-\frac{1}{2}(X^T\Lambda X-2\mu^T\Lambda X)\right\}=\exp\left\{-\frac{1}{2}X^T\Lambda X+\mu^T\Lambda X\right\}$$

又因为，Λ 是对称矩阵，所以 $(\mu^T\Lambda)^T=(\Lambda\mu)$，所以：

$$P(X)\propto\exp\left\{-\frac{1}{2}X^T\Lambda X+(\Lambda\mu)^TX\right\} \tag{17}$$

其中 $X^T\Lambda X$ 为二次项，$(\Lambda\mu)^TX$ 为一次性，其中 Λ 为 Precision Matrix，$\Lambda\mu$ 为 Potential Matrix。

3.2 Gaussian Markov Random Field 次项分析

我们来从 (17) 中提取一下和 x_i 相关的项，提取的方法直接寻找相关项就行：

$$x_i:\ -\frac{1}{2}x_i^2\lambda_{ii}+h_ix_i \tag{18}$$

其中，h_i 为一个 p 维的向量。

接着从 (17) 提取和 x_i 和 x_j 的相关项：

$$x_i, x_j: \ -\frac{1}{2}(\lambda_{ij}x_ix_j + \lambda_{ji}x_jx_i) = -\lambda_{ij}x_ix_j \tag{19}$$

因为，Λ 是对称矩阵。

当 $\lambda_{ij}x_ix_j = 0$，就意味着边上的势函数就等于 0，就意味着两个点之间是没有边是直接相连的。**因为没有边直接向量，那么在其他点都观察到的情况下，x_i 和 x_j 之间是相互独立的**。所以，$\lambda_{ij}x_ix_j = 0$ 就蕴涵着 $x_i \perp x_j|-\{x_i, x_j\}$，这实际上非常的巧妙。所以，通过上述的分析，我们就成功的把多维高斯分布和一个无向图结合在了一起。所以，我们对 $P(X)$ 进行研究的目的就是想知道，多维高斯模型和无向图之间的关系。**通过 Λ 矩阵做到了一个结合，这也就是我们在公式 (3) 下方给出的结论的原因**。

我们可以看到，**在对 Gaussian Distribution 的参数进行学习的过程中，我们不仅可以学习到了参数，也同时根据 Λ 的也就可以学习到结构** ($\lambda_{ij}x_ix_j = 0 \Longleftrightarrow x_i \perp x_j|-\{x_i, x_j\}$)。因为高斯分布和高斯马尔可夫场的巧妙联系，在学习的过程中参数和结构都有考虑，这样的做法是非常棒的。

3.3 小结

本小节，最重要的就是三点，这里总结一下：

- $x_i \perp x_j$ 中，$\Sigma = [\sigma_{ij}]$，$x_i \perp x_j \Longleftrightarrow \sigma_{ij} = 0$，也就是边缘独立，或者说是完全独立。
- $x_i \perp x_j|-x_i, x_j$ 中。$\Lambda = \Sigma^{-1} = [\lambda_{ij}]$，$x_i \perp x_j|-x_i, x_j \Longleftrightarrow \lambda_{ij} = 0$，这就是条件独立的特点。
- $\forall x$，$P(x_i|-x_i) \sim \mathcal{N}(\sum_{i\neq j}\frac{\lambda_{ij}}{\lambda_{ii}}x_j, \lambda_{ii}^{-1}(\sigma_{ii}))$，根据这个公式，我们可以深刻的体会到 x_i 可以看做其他与其相连的 x_j 的线性组合。这个公式是怎么算来的呢？实际上思路很简单，

  $$X = \begin{bmatrix} x_i \\ -\{x_i\} \end{bmatrix} = \begin{bmatrix} x_a \\ x_b \end{bmatrix}$$

  而如果在知道高斯分布联合概率分布的情况下，求解条件概率分布是有系统的推导方法的。这里就不推了，有兴趣的同学自己请查阅之前讲到的 “[机器学习基础 02] 白板推导数学基础”，**并且参考 up 主的思路，或者 PRML 的第二章 P85，条件高斯分布。有详细的推导**。

4 总结 (Conclusion)

本小节主要介绍了连续变量的概率图模型，高斯网络，根据有向图和无向图，可以分为高斯贝叶斯网络和高斯马尔可夫网络。我觉得高斯贝叶斯的主要思路是考虑整体高斯网络和局部线性高斯模型之间的联系，高斯马尔可夫的主要思路是考虑网络和联合概率密度函数之间的关系。他们都是将复杂的模型进行简化，迁移到简单的模型中去解决。但是，我们只学习了模型的建立和结构特点，并没有学习根据这个结构使用数据来求得参数的过程，相关研究领域的同学可以自行研究，如果不是要用到这个，基本就可以到此为止了。小编的主要研究方向是强化学习，此处就不再做更多的解释。

Bayes Linear Classification 01 Background

Chen Gong

05 November 2019

数据集 $D = \{(x_i, y_i)\}_{i=1}^{N}$，其中 $x_i \in \mathbb{R}^p$，$y_i \in \mathbb{R}$。
数据矩阵为：(这样可以保证每一行为一个数据点)

$$X = (x_1, x_2, \cdots, x_N)^T = \begin{pmatrix} x_1^T \\ x_2^T \\ \vdots \\ x_N^T \end{pmatrix} = \begin{pmatrix} x_{11} & x_{12} & \dots & x_{1p} \\ x_{21} & x_{32} & \dots & x_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ x_{N1} & x_{N2} & \dots & x_{Np} \end{pmatrix}_{N \times P} \tag{1}$$

$$Y = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{pmatrix}_{N \times 1} \tag{2}$$

拟合函数我们假设为：$f(x) = w^T x = x^T w$。
预测值 $y = f(x) + \varepsilon$，其中 ε 是一个 Guassian Noise，并且 $\varepsilon \sim \mathcal{N}(0, \sigma^2)$。
并且，x, y, ε 都是 Random variable。

1 最小二乘估计 (Least Square Estimation)

这实际上就是一个利用数据点的极大似然估计 (MLE)，并且有一个默认的隐含条件，也就是噪声 ε 符合 Gaussian Distribution。我们的目标是通过估计找到 w，使得：

$$w_{MLE} = argmax_w p(Data|w) \tag{3}$$

而如果仅仅只是这样来使用，很容易会出现过拟合的问题。所以，我们引入了 Regularized LSE，也就是正则化最小二乘法。同时也有一个默认的隐含条件，也是噪声 ε 符合 Gaussian Distribution。在 Liner Regression 中我们提到了有两种方法来进行思考，也就是 Lasso 和 Ridge Regression。在这里我们可以使用一个 Bayes 公式，那么：

$$p(w|Data) \propto p(Data|w)p(w) \tag{4}$$

$$w_{MAP} = argmax_w p(w|Data) = argmax_w p(Data|w)p(w) \tag{5}$$

那么假设 $p(w)$ 符合一个高斯分布 $\mathcal{N}(\mu_0, \Sigma_0)$ 时，这时是属于 Ridge（具体在线性回章节有介绍，也就是正则化的最小二乘估计 $\Leftrightarrow$ 先验服从高斯分布的极大后验估计）；而如果 $p(w)$ 符合一个 Laplace

分布，这是就是 Lasso。从概率的角度来思考和统计的角度来思想，我们其实获得的结果是一样的，这在 Linear Regression 中有证明。但是，我们只证明了 Ridge 的部分。

2 贝叶斯估计与频率派估计

其实在第一部分，我们讲的都是点估计，频率派估计的部分。因为在这些思路中，我们把参数 w 当成 a unknown random variable。这实际上就是一个优化问题。而在 Beyesian method 中，认为 w 是一个随机变量，也就是一个分布，那么我们求的 w 不再是一个数了，而是一个分布。下面我们将要进行 Bayes Linear Regression 的部分。

Bayes Linear Classification 02 Inference

Chen Gong

05 November 2019

数据集 $D=\{(x_i,y_i)\}_{i=1}^N$，其中 $x_i\in\mathbb{R}^p$，$y_i\in\mathbb{R}$。数据矩阵为：(这样可以保证每一行为一个数据点)

$$X=(x_1,x_2,\cdots,x_N)^T=\begin{pmatrix}x_1^T\\x_2^T\\\vdots\\x_N^T\end{pmatrix}=\begin{pmatrix}x_{11}&x_{12}&\dots&x_{1p}\\x_{21}&x_{32}&\dots&x_{2p}\\\vdots&\vdots&\ddots&\vdots\\x_{N1}&x_{N2}&\dots&x_{Np}\end{pmatrix}_{N\times P} \tag{1}$$

$$Y=\begin{pmatrix}y_1\\y_2\\\vdots\\y_N\end{pmatrix}_{N\times 1} \tag{2}$$

拟合函数我们假设为：$f(x)=w^Tx=x^Tw$。

预测值 $y=f(x)+\varepsilon$，其中 ε 是一个 Guassian Noise，并且 $\varepsilon\sim\mathcal{N}(0,\sigma^2)$。

并且，x,y,ε 都是 Random variable。

贝叶斯估计方法 (Bayesian Method)，可以分为两个步骤，1.Inference，2.Prediction。Inference 的关键在于估计 posterior(w)；而 Prediction 的关键在于对于给定的 x^* 求出预测值 y^*。

1 Bayesian Method 模型建立

首先我们需要对公式使用贝叶斯公式进行分解，便于计算：

$$p(w|Data)=p(w|X,Y)=\frac{p(w,Y|X)}{p(Y|X)}=\frac{p(Y|X,w)p(w)}{\int_w p(Y|X,w)p(w)dw} \tag{3}$$

其中 $p(Y|X,w)$ 是似然函数 (likelihood function)，$p(w)$ 是一个先验函数 (prior function)。实际这里省略了一个过程，$p(w,Y|X)=p(Y|X,w)p(w|X)$。但是很显然，$p(w|X)$ 中 X 与 w 之间并没有直接的联系（也就是说每个数据样本中的 x 都是从数据总体分布 $p(x)$ 中抽样得到的，与先验分布无关)。所以 $p(w|X)=p(w)$。

似然函数的求解过程为：

$$p(Y|X,w)=\prod_{i=1}^N p(y_i|x_i,w) \tag{4}$$

又因为 $y = w^T x + \varepsilon$，并且 $\varepsilon \sim \mathcal{N}(0, \sigma^2)$。所以

$$p(y_i|x_i, w) = \mathcal{N}(w^T x_i, \sigma^2) \tag{5}$$

所以，

$$p(Y|X, w) = \prod_{i=1}^{N} p(y_i|x_i, w) = \prod_{i=1}^{N} \mathcal{N}(w^T x_i, \sigma^2) \tag{6}$$

而下一步，我们假设 $p(w) = \mathcal{N}(0, \Sigma_p)$。又因为 $p(Y|X)$ 与参数 w 无关，所以这是一个定值。所以，我们可以将公式改写为：

$$p(w|X, Y) \propto p(Y|w, X)p(w) \tag{7}$$

在这里我们将使用到一个共轭的技巧，因为 likelihood function 和 prior function 都是 Gaussian Distribution，所有 posterior 也一定是 Gaussian Distribution。所以，我们可以将公式改写为：

$$p(w|Data) \sim \mathcal{N}(\mu_w, \Sigma_w) \propto \prod_{i=1}^{N} \mathcal{N}(w^T x_i, \sigma^2)\mathcal{N}(0, \Sigma_p) \tag{8}$$

我们的目的就是求解 $\mu_w = ?, \Sigma_w = ?$。

2 模型的求解

对于 likelihood function 的化简如下所示：

$$p(Y|X, w) = \prod_{i=1}^{N} \frac{1}{(2\pi)^{\frac{1}{2}}\sigma} exp\left\{-\frac{1}{2\sigma^2}(y_i - w^T x_i)^2\right\} \tag{9}$$

$$= \frac{1}{(2\pi)^{\frac{N}{2}}\sigma^N} exp\left\{-\frac{1}{2\sigma^2}\sum_{i=1}^{N}(y_i - w^T x_i)^2\right\} \tag{10}$$

下一步，我们希望将 $\sum_{i=1}^{N}(y_i - w^T x_i)^2$ 改写成矩阵相乘的形式，

$$\begin{aligned}\sum_{i=1}^{N}(y_i - w^T x_i)^2 &= \begin{bmatrix} y_1 - w^T x_1 & y_2 - w^T x_2 & \cdots & y_i - w^T x_i \end{bmatrix} \begin{bmatrix} y_1 - w^T x_1 \\ y_2 - w^T x_2 \\ \vdots \\ y_i - w^T x_i \end{bmatrix} \\ &= (Y^T - w^T X^T)(Y^T - w^T X^T)^T \\ &= (Y^T - w^T X^T)(Y - Xw)\end{aligned} \tag{11}$$

所以，

$$\begin{aligned}p(Y|X, w) &= \frac{1}{(2\pi)^{\frac{N}{2}}\sigma^N} exp\left\{-\frac{1}{2\sigma^2}\sum_{i=1}^{N}(Y^T - w^T X^T)(Y - Xw)\right\} \\ &= \frac{1}{(2\pi)^{\frac{N}{2}}\sigma^N} exp\left\{-\frac{1}{2}\sum_{i=1}^{N}(Y^T - w^T X^T)\sigma^{-2} I(Y - Xw)\right\} \\ p(Y|X, w) &\sim \mathcal{N}(Xw, \sigma^2 I)\end{aligned} \tag{12}$$

那么，将化简后的结果带入有：

$$p(w|Data) \sim \mathcal{N}(\mu_w, \Sigma_w) \propto \mathcal{N}(Xw, \sigma^2 I)\mathcal{N}(0, \Sigma_p) \tag{13}$$

$$\begin{aligned}\mathcal{N}(Xw, \sigma^2 I)\mathcal{N}(0, \Sigma_p) \propto & exp\left\{-\frac{1}{2}(Y - Xw)^T\sigma^{-2}I(Y - Xw) - \frac{1}{2}w^T\Sigma_p^{-1}w\right\} \\ = & exp\left\{-\frac{1}{2\sigma^2}(Y^TY - 2Y^TXw + w^TX^TXw) - \frac{1}{2}w^T\Sigma_p^{-1}w\right\}\end{aligned} \tag{14}$$

那么这个公式长得怎么的难看我们怎么确定我们想要的 μ_w, Σ_w。由于知道 posterior 必然是一个高斯分布，那么我们采用待定系数法来类比确定参数的值即可。对于一个分布 $p(x) \sim \mathcal{N}(\mu, \Sigma)$，他的指数部分为：

$$exp\left\{-\frac{1}{2}(x - \mu)^T\Sigma^{-1}(x - \mu)\right\} = exp\left\{-\frac{1}{2}(x^T\Sigma^{-1}x - 2\mu^T\Sigma^{-1}x + \triangle)\right\} \tag{15}$$

常数部分已经不重要了，对于我们的求解来说没有任何的用处，所以，我们直接令它为 $\triangle$。那么，我们类比一下就可以得到，

$$w^T\Sigma_w^{-1}w = w^T\left(\sigma^{-2}X^TX + \Sigma_p^{-1}\right)W \tag{16}$$

所以，我们可以得到 $\Sigma_w^{-1} = \sigma^{-2}X^TX + \Sigma_p^{-1}$。并且，我们令 $\Sigma_w^{-1} = A$。

从二次项中我们得到了 Σ_w^{-1}，那么，下一步，我们期望可以从一次项中得到 μ_A 的值。我们将一次项提取出来进行观察，可以得到。

$$\mu^TA = \sigma^{-2}Y^TX \tag{17}$$

$$(\mu^TA)^T = (\sigma^{-2}Y^TX)^T \tag{18}$$

$$A^T\mu = \sigma^{-2}X^TY \tag{19}$$

$$\mu = \sigma^{-2}(A^T)^{-1}X^TY \tag{20}$$

又因为，Σ_w 是一个协方差矩阵，那么他一定是对称的，所以 $A^T = A$。于是

$$\mu_w = \sigma^{-2}A^{-1}X^TY \tag{21}$$

3 小结

我们利用贝叶斯推断的方法来确定参数之间的分布，也就是确定 $p(W|X, Y)$。我们使用 Bayes 的方法，确定为 $p(W|X, Y) \propto p(Y|W, X)p(W)$。并且确定一个噪声分布 $\varepsilon \sim \mathcal{N}(0, \sigma^2)$。那么，

$$p(Y|w, X) \sim \mathcal{N}(Xw, \sigma^2) \tag{22}$$

$$P(w) \sim \mathcal{N}(0, \Sigma_p) \tag{23}$$

通过推导，我们可以得出，

$$p(w|X, Y) \sim \mathcal{N}(\mu_w, \Sigma_w) \tag{24}$$

其中，

$$\Sigma_w^{-1} = \sigma^{-2}X^TX + \Sigma_p^{-1} \qquad \mu_w = \sigma^{-2}A^{-1}X^TY \qquad \Sigma_w^{-1} = A \tag{25}$$

Bayes Linear Classification 03 Prediction & Conclusion

Chen Gong

06 November 2019

根据上一节中提到的 Inference，我们已经成功的推断出了 $p(w|Data)$ 的分布。表述如下所示：

$$p(w|X,Y) \sim \mathcal{N}(\mu_w, \Sigma_w) \tag{1}$$

其中，

$$\Sigma_w^{-1} = \sigma^{-2}X^TX + \Sigma_p^{-1} \qquad \mu_w = \sigma^{-2}A^{-1}X^TY \qquad \Sigma_w^{-1} = A \tag{2}$$

而我们的 Prediction 过程，可以被描述为，给定一个 x^* 如果计算得到 y^*。而我们的模型建立如下所示：

$$\begin{cases} f(x) = w^Tx = x^Tw \\ y = f(x) + \varepsilon \qquad\qquad \varepsilon \sim \mathcal{N}(0, \sigma^2) \end{cases} \tag{3}$$

1 Prediction

模型预测的第一步为，

$$f(x^*) = {x^*}^Tw \tag{4}$$

而在 Inference 部分，我们得到了 $p(w|Data) = \mathcal{N}(\mu_w, \Sigma_w)$。所以，我们可以推断出，

$$f(x^*) = {x^*}^Tw \sim \mathcal{N}({x^*}^T\mu_w, {x^*}^T\Sigma_w x^*) \tag{5}$$

那么公式 (5) 我们可以写作：

$$p(f(x^*)|Data, x^*) \sim \mathcal{N}({x^*}^T\mu_w, {x^*}^T\Sigma_w x^*) \tag{6}$$

又因为 $y = f(x) + \varepsilon$，所以

$$p(y^*|Data, x^*) \sim \mathcal{N}({x^*}^T\mu_w, {x^*}^T\Sigma_w x^* + \sigma^2) \tag{7}$$

那么计算到这里，我们的模型预测也算是完成了。

2 Conclusion

Data：$D = \{(x_i, y_i)\}_{i=1}^N$，其中 $x_i \in \mathbb{R}^p$，$y_i \in \mathbb{R}$。

Model：

$$\begin{cases} f(x) = w^TX = x^Tw \\ y = f(x) + \varepsilon \qquad\qquad \varepsilon \sim \mathcal{N}(0, \sigma^2) \end{cases} \tag{8}$$

Bayesian Method：w 不在是一个未知的常数，w 而是一个概率分布。贝叶斯线性分类可以被分成两个部分，Inference 和 Prediction。

1. Inference：$p(w|Data)$ 是一个 posterior 分布，假定 $p(w|Data) = \mathcal{N}(\mu_w, \Sigma_w) \propto likelihood \times prior$。这里使用了共轭的小技巧，得到 posterior 一定是一个 Gaussian Distribution。在这一步中，我们的关键是求出 μ_w 和 Σ_w。

2. Prediction：这个问题实际上也就是，给定一个 x^* 如果计算得到 y^*。我们可以描述为：

$$p(y^*|Data, x^*) = \int_w p(y^*|w, Data, x^*)p(w|Data, x^*)dw \tag{9}$$

又因为，y^* 只依赖于 w 和 x^*，不依赖于历史数据，所以 $p(y^*|w, Data, x^*) = p(y^*|w, x^*)$。并且，$w$ 的获得与 x^* 没有关系，所以 $p(w|Data)$。所以：

$$p(y^*|Data, x^*) = \int_w p(y^*|w, x^*)p(w|Data)dw = \mathbb{E}_{w\sim p(w|Data)}[p(y^*|w, x^*)] \tag{10}$$

之后通过自共轭特性不用计算积分即可得到服从的正态分布。

Gaussian Process 01 Introduction

Chen Gong

13 December 2019

本小节我们将进入 Gaussian Process 的学习。Gaussian 自然指的就是 Gaussian Distribution，而 Process 指的就是随机过程。在一维的 Gaussian Distribution 中我们可以令 $p(x) = \mathcal{N}(\mu, \sigma^2)$。如果对应到高维高斯分布的话，也就是 (Multivariate Gaussian Distribution) 也就是我们通常意义上说的 Gaussian Network，对于任意的 $x \in \mathbb{R}^p$，有 $p(x) = \mathcal{N}(\mu, \Sigma)$，且 Σ 是一个 $p \times p$ 维的向量，$p < +\infty$。如果是一个无限维的 Gaussian Distribution，那么就是我们今天要讨论的 Gaussian Process 了。首先我们给出 Gaussian Process 的详细定义，Gaussian Process：定义在连续域上的无限多个高维随机变量所组成的随机过程。所谓连续域指的就是时间或者空间。下面我们来进行详细的解释。

1 Gaussian Process 解释

我们假设有一组随机变量 $\{\xi_t\}_{t\in T}$，T 是一个连续域，如果 $\forall n \in N^+$，都有 $t_1, t_2, \cdots, t_n \in T$，并且存在一个约束条件 $s.t.$ $\{\xi_{t_1}, \xi_{t_2}, \cdots, \xi_{t_n}\} \triangleq \xi_{t_1 \sim t_n} \sim \mathcal{N}(\mu_{t_1\sim t_n}, \Sigma_{t_1\sim t_n})$。那么我们就称 $\{\xi_t\}_{t\in T}$ 是一个 Gaussian Process。那么，我们怎么来通俗的理解这个概念呢? 也就是说有一系列在时间上或空间上连续的点，他们之间分开看都是符合一个高斯分布的，而合起来看则是符合一个多维高斯分布的。也就是如下图所示的，在五个不同的时刻有 5 个不同的点，之上的随机变量为 $\{\xi_{t_1}, \xi_{t_2}, \xi_{t_3}, \xi_{t_4}, \xi_{t_5}\}$，他们分别都符合一个高斯分布。

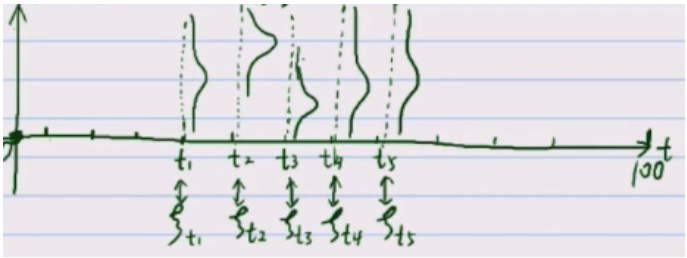

图 1: 多个点符合高斯分布的情况

2 Gaussian Process 举例说明

为了帮助大家更好的来理解高斯分布，我们在这里讲一个故事。假如一个人的一生可以活到 100 岁，横坐标就为时间 t，而纵坐标表示的为在这个时间点的表现值。(大概就这样将就的理解一下)。其中，$t \in [0, 100]$，每一个 $\xi_t \sim \mathcal{N}(\mu_t, \sigma_t)$。这是什么意思，也就是在每一个时刻这个人的表现值都符合一个独立的高斯分布，也就是在 t 时刻他的表现为 $[0, 100]$。

现在，我们做一个假设，假设一个人，当 $t=0$ 的时刻，他的一生就已经确定了，也就是他的每一个时刻的表现值都会符合一个确定的 Gaussian Distribution，μ_t 和 σ_t^2 都是确定的。假设人可以活很多次，每个点的表现值都是一个高斯分布，那么他每一生都将是不一样的，每过一生都是从高斯过程中的一次采样，如下图所示，

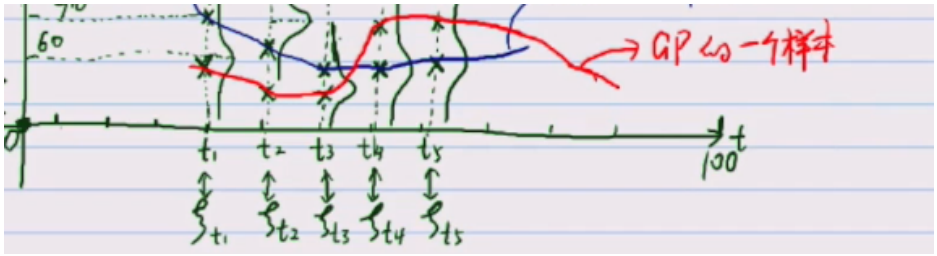

图 2: 高斯过程的样本

所以，$\{\xi_{t_1}, \xi_{t_2}, \cdots, \xi_{t_n}\}$，本身就是一个高斯分布，他们联合起来也是高斯分布，任何一个采样都属于高斯分布，我们可以看成是高斯过程的一个样本。用符号的语言描述就是 $GP(m(t), k(s,t))$。其中

$$m_t = \mathbb{E}[\xi_t] = mean\ function \tag{1}$$

$$k(s,t) = covariance function = \mathbb{E}\left[[\xi_s - m(s)][\xi_t - m(t)]^T\right] \tag{2}$$

也就是说，一个高斯过程属于一个多维高斯分布，服从分布的形式为 $\mathcal{N}(m_t, k(s,t))$。

Gaussian Process 02 Weight Space

Chen Gong

14 December 2019

Gaussian Process 在这里我们主要讲解的是 Gaussian Process Regression。我们从 Bayesian Linear Regression 的角度来引出的 Gaussian Progress Regression。

1 Recall Bayesian Linear Regression

首先，我们需要回顾一下 Bayesian Linear Regression。

1. 首先对于一个贝叶斯线性回归来说，参数符合的分布 $p(w|Data) = \mathcal{N}(w|\mu_w, \Sigma_w)$。其中，$\mu_w = \sigma^{-2}A^{-1}X^TY$，$\Sigma_w = A^{-1}$，其中，$A = \sigma^{-2}X^TX + \Sigma_p^{-1}$。从这一步我们就成功的得到了在已知 Data 的情况下，未知参数的分布形式。

2. 在给定一个新的未知数向量 X^* 的情况下，我们可以首先利用 noise-free 形式：$f(x) = w^Tx = x^Tw$，然后再求得 noise 形式：$y = f(x) + \epsilon$，而 $\epsilon \sim \mathcal{N}(0, \sigma^2)$。来获得我们想要的 prediction 值。这样，我们就可以得到：

$$p(f(x^*)|Data, x^*) \sim \mathcal{N}(x^{*T}\mu_w, x^{*T}\Sigma_w x^*) \tag{1}$$

$$p(y^*|Data, x^*) \sim \mathcal{N}(x^{*T}\mu_w, x^{*T}\Sigma_w x^* + \sigma^2) \tag{2}$$

但是，问题马上就来了，因为很多时候，我们不能仅仅使用线性分类的方法来解决问题。现实生活中有许多非线性的问题来待我们求解。而一种经常使用的方法，也就是将数据投影到高维空间中来将非线性问题，转换成一个高维空间中的线性可分问题。或者是使用 Bayesian Logistics Regression 来进行分类。如果，是将数据投影到高维空间中的话，我们很自然的就想到了 Kernel Bayesian Linear Regression。

那么这个非线性转换可以被我们写成：If $\phi : x \mapsto z$，$x \in \mathbb{R}^p$，$x \in \mathbb{R}^p$，$z \in \mathbb{R}^q$，$z = \phi(x)$。

2 非线性转换后的表达

数据集被我们描述为：$X = (x_1, x_2, \cdots, x_N)^T, Y = (y_1, y_2, \cdots, y_N)^T$。根据之前我们得到的 Bayesian Linear Regression 结果，我们代入可以得到：

$$p(f(x^*)|X, Y, x^*) \sim \mathcal{N}(x^{*T}(\sigma^2 A^{-1}X^TY), x^{*T}A^{-1}x^*) \tag{3}$$

而其中，$A = \sigma^{-2}X^TX + \Sigma_p^{-1}$，If $\phi : x \mapsto z$，$x \in \mathbb{R}^p$，$z \in \mathbb{R}^q$，$z = \phi(x)(q > p)$。这里的 ϕ 是一个非线性转换。我们定义：$\Phi = (\phi(x_1), \phi(x_2), \cdots, \phi(x_N))^T_{N \times q}$。

转换之后为：$f(x) = \phi(x)^T w$。那么，

$$p(f(x^*)|X, Y, x^*) \sim \mathcal{N}(\sigma^{-2}\phi(x^*)^T(A^{-1}\Phi(X)^T Y), \phi(x^*)^T A^{-1}\phi(x^*)) \tag{4}$$

而其中，$A = \sigma^{-2}\Phi(X)^T\Phi(X) + \Sigma_p^{-1}$。但是，很快我们又将面临一个新的问题，也就是 A^{-1} 应该如何计算呢？这里我们需要使用到一个公式为，Woodbury Formula 公式（在矩阵含有一定结构时加速矩阵求逆）：

$$(A + UCV)^{-1} = A^{-1} - A^{-1}U(C^{-1} + VA^{-1}U)^{-1}VA^{-1} \tag{5}$$

也可以通过如下变换的形式计算 A^{-1}：

$$\begin{aligned} A &= \sigma^{-2}\Phi(X)^T\Phi(X) + \Sigma_p^{-1} \\ A\Sigma_p &= \sigma^{-2}\Phi(X)^T\Phi(X)\Sigma_p + I \\ A\Sigma_p\Phi(X)^T &= \sigma^{-2}\Phi(X)^T\Phi(X)\Sigma_p\Phi(X)^T + \Phi(X)^T = \sigma^{-2}\Phi(X)^T(K + \sigma^2 I) \\ \sigma^{-2}A^{-1}\Phi(X)^T &= \Sigma_p\Phi(X)^T(K + \sigma^2 I)^{-1} \end{aligned} \tag{6}$$

然后，两边同乘一个 $\phi(x^*)^T$ 和 Y 就可以得到：

$$\sigma^{-2}\phi(x^*)^T A^{-1}\Phi(X)^T Y = \phi(x^*)^T\Sigma_p\Phi(X)^T(K + \sigma^2 I)^{-1}Y \tag{7}$$

而这个 $\sigma^{-2}\phi(x^*)^T A^{-1}\Phi(X)^T Y$ 正好就是 $p(f(x^*)|X, Y, x^*)$ 的期望（通过贝叶斯回归类比出来的正态分布中的期望）。而这里的 $\Sigma_p = p(w)$ 是一个先验 $\sim \mathcal{N}(0, \Sigma_p)$，而 σ^2 为先验分布的噪声，x^* 是一个 new input，而$K = \Phi\Sigma_p\Phi^T$。所以，使用类似的方法我们可以得到，$p(f(x^*)|X, Y, x^*)$'s Covarience 为：$\phi(x^*)^T\Sigma_p\phi(x^*) - \phi(x^*)^T\Sigma_p\Phi(X)^T(K + \sigma^2 I)^{-1}\Phi(X)\Sigma_p\phi(x^*)$。所以：

$$\begin{aligned} p(f(x^*)|X, Y, x^*) \sim &\mathcal{N}(\phi(x^*)\Sigma_p\Phi(X)^T(K + \sigma^2 I)^{-1}Y, \\ &\phi(x^*)^T\Sigma_p\phi(x^*) - \phi(x^*)^T\Sigma_p\Phi(X)^T(K + \sigma^2 I)^{-1}\Phi(X)\Sigma_p\phi(x^*)) \end{aligned} \tag{8}$$

而大家注意观察一下，下面几个等式：

$$\phi(x^*)^T\Sigma_p\Phi^T \qquad \phi(x^*)^T\Sigma_p\phi(x^*) \qquad \Phi\Sigma_p\Phi^T \qquad \Phi\Sigma_p\phi(x^*) \tag{9}$$

这里的 $\Phi_{N\times q}$ 表示的是经过变换之后的数据矩阵：

$$\Phi_{N\times q} = (\phi(x_1), \phi(x_2), \cdots, \phi(x_N))^T_{N\times q} \tag{10}$$

所以大家想一想就知道了，公式 (9) 中的四个公式实际上是一个东西，而 $\Phi(X)$ 只不过是将多个向量拼接在了一起而已。而 $K(x, x') = \phi(x)^T\Sigma_p\phi(x')$，$x, x'$ 是两个不一样的样本，矩阵展开以后，形式都是一样的。那么下一个问题就是 $K(x, x')$ 是否可以表达为一个 Kernel Function 的形式？那么，相关的探究就变得有趣了。

3 Kernel Trick

因为 Σ_p 是一个 positive define matrix，并且它也是 symmetry 的。所以，令 $\Sigma_p = (\Sigma_p^{\frac{1}{2}})^2$。那么，我们可以做如下的推导：

$$\begin{aligned} K(x, x') =& \phi(x)^T \Sigma_p^{\frac{1}{2}} \Sigma_p^{\frac{1}{2}} \phi(x') \\ =& (\Sigma_p^{\frac{1}{2}} \phi(x))^T \cdot \Sigma_p^{\frac{1}{2}} \phi(x) \\ =& \varphi(x)^T \varphi(x') \end{aligned} \tag{11}$$

其中，$\varphi(x) = \Sigma_p^{\frac{1}{2}} \phi(x)$。那么，我们利用 Kernel Trick 可以有效的避免求 $\phi(X)$，而是直接通过 $K(x, x')$ 中包含的高维空间的转化。而Bayesian Linear Regression + Kernel Trick 中就蕴含了一个 Non-Linear Transformation inner product。我们就可以将这个转换定义到一个核空间中，避免了直接来求这个复杂的转化。这也就是 Kernel Trick。

Gaussian Process Regression ，可以从两种视角去解释，而这两种视角可以得到 equal result：

1. Weight-Space view，也就是我们这一小节所讲的东西。指的就是那两个等式，$f(x) = \phi(x)^T w$ 和 $y = f(x) + \epsilon$。在这里我们的研究对象就是 w，假设 w 的先验，需要求得 w 的后验，所以是从 Weight-Space 的角度分析的。

2. Function-Space view，我们将 $f(x)$ 看成是一个随机变量，$f(x) \sim GP(m(x), K(x, x'))$。这个我们会在后面的小节中进行详细的描述，大家就可以看到 GP 的思想在其中的运用了。

而有一句话对 GPR 的总结，非常的有意思，Gaussian Process Regress is the extension of Bayesian Linear Regression with kernel trick. 仔细想一想就知道了，我们把逻辑思路理一下，我们想用贝叶斯练习回归来解决非线性的问题，所以我们需要把输入空间投射到一个高维空间中，低维空间中的线性不可分问题将可以转化为高维空间中的线性可分问题。那么，我们就需要一个转换函数来完成这个工作，但是这个转换函数怎么求？有可能会很难求，而且维度很高。那么，我们就不求了，直接使用核技巧，也就是两个向量的内积等于一个核函数的值就可以了。这大概就是本节中 Weight-Space View 的一个主线的思路。

Gaussian Process 03 Function View

Chen Gong

15 December 2019

在上一小节中，我们从 Weight-Space View 来看 Gaussian Process Regression，好像和 Gaussian Process 并没有什么关系。但是这一小节，我们从函数的角度来看就可以看到了。

1 Recall Gaussian Process

对于一组随机变量 $\{\xi_t\}_{t\in T}, T:$ continuous space or time。If:$\forall n \in N^+ \ (n \geq 1)$, Index:$\{t_1, t_2, \cdots, t_n\} \longrightarrow$ random variable: $\{\xi_{t_1}, \xi_{t_2}, \cdots, \xi_{t_n}\}$。令 $\xi_{1:n} = \{\xi_{t_1}, \xi_{t_2}, \cdots, \xi_{t_n}\}^T$。If $\xi_{1:n} \sim \mathcal{N}(\mu_{1:n}, \Sigma_{1:n})$，那么我们称 $\{\xi_t\}_{t\in T}$ is a Gaussian Distribution。并且，$\xi_t \sim GP(m(t), k(t,s))$，$m(t)$ 为 mean function，$k(t,s)$ 为 covariance function。下面我们回到 Weight-Space View 中。

2 Weight-Space view to Function-Space view

在这里 w 是一个先验分布，$f(x)$ 是一个随机变量。$f(x) = \phi(x)^T w$，$y = f(x) + \epsilon$，$\epsilon \sim \mathcal{N}(0, \sigma^2)$。在 Bayesian 的方法中，对于给定的先验信息 (prior)：$w \sim \mathcal{N}(0, \Sigma_p)$。因为，$f(x) = \phi(x)^T w$，所以可以得到：

$$\mathbb{E}_w[f(x)] = \mathbb{E}_w[\phi(x)^T w] = \phi(x)^T \mathbb{E}_w[w] = 0 \tag{1}$$

那么对于 $\forall x, x' \in \mathbb{R}^p$，

$$\begin{aligned} cov(f(x), f(x')) =& \mathbb{E}[(f(x) - \mathbb{E}[f(x)])(f(x') - \mathbb{E}[f(x')])] \\ =& \mathbb{E}[f(x)f(x')] \\ =& \mathbb{E}[\phi(x)^T w \phi(x')^T w] \\ =& \mathbb{E}[\phi(x)^T w w^T \phi(x')] \end{aligned} \tag{2}$$

因为 $\phi(x')^T w$ 的结果是一个实数，所以它的转置就等于它自己。又因为 $w \sim \mathcal{N}(0, \Sigma_p)$，均值为 0，协方差为 Σ_p。并且有 $\mathbb{E}[ww^T] = \mathbb{E}[(w-0)(w^T-0)]$，这个东西不就是协方差矩阵 $cov(w) = \Sigma_p$。

而 $\phi(x)^T \Sigma_p \phi(x')$ 是一个 kernel function，前面我们已经证明过了，$\varphi(x) = \Sigma_p^{\frac{1}{2}}$。而 $\phi(x)\Sigma_p\phi(x') =< \varphi(x), \varphi(x') >= K(x, x')$。

推导进行到了这里，我们就知道了 $f(x)$ 的期望为 0，协方差矩阵由一个核函数 $K(x, x')$ 产生。那么我们是不是惊奇的发现，这个和我们高斯过程的定义：$\xi_t \sim GP(m(t), K(t,s))$，是多么惊人的相似呀。所以，这里可以启发我们：$f(x)$ 的组成是否可以看成一个 GP，而 $\{f(x)\}_{x\in\mathbb{R}^p}$。那么，首先 $f(x)$

是一个 function，而且 $f(x)$ 还是一个服从高斯分布的随机变量，$m(t)$ 是一个 mean function，$K(t,s)$ 是一个 covariance function。为了加深大家的理解，我们做进一步清晰的对比：

$$\begin{cases} t \longrightarrow \xi_t,\ \{\xi_t\}_{t\in T} \sim GP \\ x \longrightarrow f(x),\ \{f(x)\}_{x\in\mathbb{R}^p} \sim GP \end{cases} \tag{3}$$

其实，我这样一对比，就非常的清晰了。在 GPR 的算法中，

1. Weight-Space view 中关注的是 w，即为：

$$x^* \longrightarrow y^* \quad p(y^*|Data, x^*) = \int p(y^*|Data, x^*, w)p(w)dw \tag{4}$$

又因为 w 本身就是从 Data 中，推导得到的，所以 $p(y^*|Data, x^*, w) = p(y^*|x^*, w)$。

2. Function-Space view 中关注的是 $f(x)$，即为：

$$p(y^*|Data, x^*) = \int p(y^*|f(x), x^*)p(f(x))df(x) \tag{5}$$

写到了这里，不知道大家有没有一定感觉了，这里就是把 $f(x)$ 当成了一个随机变量来看的。这里也就是通过 $f(x)$ 来直接推导出 y^*。在 Weight-Space View 中，我们没有明确的提到 GP，但是在 Weight-Space view 中，$f(x)$ 是符合 GP 的，只不过是没有显性的表示出来而已。我们可以用一个不是很恰当的例子来表述一个，Weight-Space view 就是两个情侣之间，什么都有了，孩子都有了，但是就是没有领结婚证，那么他们两个之间的关系就会比较复杂。而 Function-Space view 就是两个情侣之间先领结婚证，在有了孩子，按部就班的来进行，所以他们之间的关系就会比较简单。

3 Function-Space View

上一小节中，我们从 Weight-Space View 过渡到了 Function-Space View，而 Weight 指的就是参数。

$$\begin{aligned} &\{f(x)\}_{x\in\mathbb{R}^p} \sim GP(m(x), K(x,x')) \\ &m(x) = \mathbb{E}[f(x)] \quad K(x,x') = \mathbb{E}[(f(x)-m(x))(f(x')-m(x'))] \end{aligned} \tag{6}$$

Regression 问题被我们描述为：

Data：$\{(x_i, y_i)\}_{i=1}^N$，$x = (x_1, x_2, \cdots, x_N)^T_{N\times p}$，$Y = (y_1, y_2, \cdots, y_N)^T_{N\times 1}$。又因为 $f(x)$ 符合一个 GP，所以，$f(x) \sim \mathcal{N}(\mu(x), K(x,x'))$。且 $Y = f(X) + \epsilon$，所以 $Y \sim \mathcal{N}(\mu(x), K(x,x') + \sigma^2 I)$。那么，给定 new input：$X^* = (x_1^*, x_2^*, \cdots, x_N^*)$，我们想要的 Prediction output 为 $Y^* = f(X^*) + \epsilon$。那么，我们可以得到 Y 和 $f(X^*)$ 的联合概率密度分布为：

$$\begin{bmatrix} Y \\ f(X^*) \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \mu(X) \\ \mu(X^*) \end{bmatrix}, \begin{bmatrix} K(X,X) + \sigma^2 I & K(X,X^*) \\ K(X^*,X) & K(X^*,X^*) \end{bmatrix} \right) \tag{7}$$

在这里，我必须要首先列举一下，前面我们曾经提到的更加联合概率密度求边缘概率密度的方法。已知，$x \sim \mathcal{N}(\mu, \Sigma)$，

$$x = \begin{bmatrix} x_a \\ x_b \end{bmatrix} \qquad \mu = \begin{bmatrix} \mu_a \\ \mu_b \end{bmatrix} \qquad \Sigma = \begin{bmatrix} \Sigma_{aa} & \Sigma_{ab} \\ \Sigma_{ba} & \Sigma_{bb} \end{bmatrix} \tag{8}$$

而我们可以得到：

$$
\begin{aligned}
&p(x_b|x_a) \sim \mathcal{N}(\mu_{b|a}, \Sigma_{b|a}) \\
&\mu_{b|a} = \Sigma_{ba}\Sigma_{aa}^{-1}(x_a - \mu_a) + \mu_b \\
&\Sigma_{b|a} = \Sigma_{bb} - \Sigma_{ba}\Sigma_{aa}^{-1}\Sigma_{ab}
\end{aligned} \tag{9}
$$

我们要求的概率为 $p(f(X^*)|Y, X, X^*)$，就是一个我们要求的条件概率，等价于 $p(f(x^*)|Y)$，为什么这里可以把 X, X^* 给忽略掉了？因为 X 和 Y 相关，因为 $Y = \phi(X)^T w + \epsilon$。而 X^* 涵盖在了 $f(X^*)$ 中，可以把 X^* 当做已知的条件，因为 $f(X^*) = \phi(X^*)^T w$。

所以，我们的目标也就是求 $p(f(X^*)|Y)$，也就是已知联合概率分布的情况下求条件概率分布。

我们对比公式 (8) 和公式 (9) 就可以发现，$Y \to x_a, f(x^*) \to x_b, K(X,X)+\sigma^2 I \to \Sigma_{aa}, K(X, X^*) \to \Sigma_{ba}, K(X^*, X^*) \to \Sigma_{bb}$。那么，我们可以令 $p(f(X^*)|Y, X, X^*) \sim \mathcal{N}(\mu^*, \Sigma^*)$，代入之前获得的公式的结果我们就可以得到：

$$
\begin{aligned}
\mu^* =& K(X^*, X)(K(X,X) + \sigma^2 I)^{-1}(Y - \mu(X)) + \mu(X^*) \\
\Sigma^* =& K(X^*, X^*) - K(X^*, X)(K(X,X) + \sigma^2 I)^{-1}
\end{aligned} \tag{10}
$$

并且，$Y^* = f(X^*) + \varepsilon$。那么 noise-free 的形式可以被我们写完：$p(f(x^*)|Y, X, X^*) = \mathcal{N}(\mu^*, \Sigma^*)$。而 $p(Y^*|Y, X, x^*) = \mathcal{N}(\mu_y^*, \Sigma_y^*)$，$\mu_y^* = \mu^*$，$\Sigma_y^* = \Sigma^* + \sigma^2 I$。

在 Function-Space View 中，$f(x)$ 本身是符合 GP 的，那么我们可以直接写出 *Prediction* 矩阵，并将其转化为已知联合概率密度分布求条件概率密度的问题。Function-Space View 和 Weight-Space View 得到的结果是一样的，但是更加的简单。

Restricted Boltzmann Machine

Chen Gong

28 February 2020

目录

1 Background

本小节主要介绍的是受限玻尔兹曼机 (Restricted Boltzmann Machine, RBM)。本小节，我们主要讨论的是什么是 Boltzmann Machine，然后讲讲它的历史，为我们引出 Restricted Boltzmann Machine 做铺垫。

1.1 什么是 Boltzmann Machine？

其实 Boltzmann Machine 就是一种 Markov Random Field，也就是无向图而已。那么，Boltzmann Machine 和普通的无向图有什么不同呢? 区别就在于 **Markov Random Field with hidden nodes**，即为无向图中的节点，有一部分是可观测的，一部分是不可观测的。

马尔可夫随机场中的每一个节点代表一个随机变量 (Random variable)，而所有的 Random variable 可以被分为两类，即为 observed variable v 和 hidden variable h；如下图所示，灰色代表不可观测变量，白色代表可观测变量。

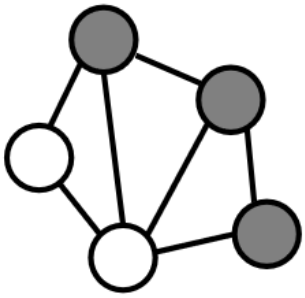

图 1: 玻尔兹曼机实例图

1.2 无向图中的因子分解

在无向图中，最重要的就是因子分解，**因子分解是对联合概率进行建模**。它基于最大团的概念来进行分解的，理论基础是 Hammersley Clifford Theorem。因子分解的公式表达为:

$$P(X) = \frac{1}{Z}\prod_{i=1}^{k}\phi_i(x_{c_i}) \tag{1}$$

其中，x_{c_i} 表示第 i 个最大团；x_{c_i} 表示第 i 个最大团中的随机变量组成的集合；$\phi_i(x_{c_i})$ 表示他们的势函数 (Potential Function)。而 Z 是归一化因子，有时也被称为配分函数 (Partition Function)。注意两个约束条件，1. $\phi_i(x_{c_i})$ 是严格正定的；2. Z 是归一化因子:

$$Z = \sum_{X}\prod_{i=1}^{k}\phi_i(x_{c_i}) = \sum_{x_1}\sum_{x_2}\cdots\sum_{x_p}\prod_{i=1}^{k}\phi_i(x_{c_i})$$

在概率图模型中，没有特殊情况都是指离散变量。

因为**指数族分布是满足最大熵原理的分布** (个人觉得这个最大熵原理简直无处不在，也是为什么很多函数，动不动就变指数函数的原因，实际上是有理论依据的，不是随便加的)。为了简化表达，我们令:

$$\phi_i(x_{c_i}) = \exp\{-\mathrm{E}(x_{c_i})\}$$

而且这样就正好满足了 $\phi_i(x_{c_i})$ 是严格正定的需求，而 E 则被称为能量函数 (Energy Function)。所以，联合概率分布，被改写为：

$$P(X) = \frac{1}{Z}\prod_{i=1}^{k}\phi_i(x_{c_i}) = \frac{1}{Z}\exp\left\{-\sum_{i=1}^{k}\mathrm{E}(x_{c_i})\right\} \tag{2}$$

而最大团中的所有变量，可以用 X 来表达，最后可以化简为一个和 X 相关的能量函数，表达为：

$$P(X) = \frac{1}{Z}\exp(-\mathrm{E}(X)) \tag{3}$$

很显然，这个分布符合指数族分布的形式，被我们称为 Boltzmann Distribution，或者 Gibbs Distribution。所以，**如果取势函数是一个指数函数，那么整体为 Boltzmann Distribution**。

前面讲了那么多，我们看了很多概念，我相信大家基本没搞懂，为什么叫“势函数”和“能量函数”，这种奇奇怪怪的叫法。下面我们来看看 Boltzmann Distribution 的历史，来帮助我们进行理解。

1.3 Boltzmann Distribution 的历史

Boltzmann Distribution 最早来自于统计物理学，这是一个物理学的概率，这里我们采用感性的理解方式。

一个物理系统由各种各样的粒子组成。而一个系统的状态 (State)，由其中各种各样的粒子的状态联合而成。系统状态的概率满足：

$$P(\mathrm{State}) \propto \exp\left\{-\frac{\mathrm{E}}{\mathrm{kT}}\right\} \tag{4}$$

其中，E 为能量函数，T 表示温度，k 为一个系数。而能量函数是一个离散的分布，一个 System 可能有 M 个状态，每个状态对应一个值，如下所示：

State	1	2	$\cdots$	i	$\cdots$	M
P	p_1	p_2	$\cdots$	p_i	$\cdots$	p_M

因为粒子有速度，受到其他粒子的干扰，所以 E 和所有的粒子有关。对于 X 的概率函数，和 E 成反比，能量越大，越不稳定，越容易发生状态的跃迁，当前状态出现的可能性就越小，概率就越低。

所以，如果没有外界的干扰，系统最终就到达一个能量比较低的稳态，因为能量高的状态都待不住。举一个例子，一个人年轻的时候，能量很强，很不稳定，工作对象什么的都很容易换。到了中年以后，这个时候可能追求的是自己的事业上的成功，而相对稳定了一些。到了老年，见的实在是太多了，这是追求的是一种心灵上的平静，内心基本没有什么冲动，一切回归平稳。

这就是无向图中一些概念的来源，用来辅助理解。这里我谈谈自己的理解：一个无向图就是一个系统，系统包括所有的节点，所以系统的状态就是系统中所有节点的联合概率。这个系统的状态的概率和内部的每一个节点都有关系，我们可以用能量函数来进行衡量一个状态出现的可能性，而能量越高的状态越容易发生转移，出现的概率越低，反之亦然。

1.4 小结

本节主要描述了，什么是 Boltzmann Machine，核心就是节点分为可观测和不可观测的马尔可夫随机场。并且，概率图的联合分布，当势函数为指数函数时，联合分布是玻尔兹曼分布（吉布斯分布）。

随后我们介绍了 Boltzmann 分布在统计物理学中的来源，来辅助我们对无向图中的一些概念的理解。介绍完了 Boltzmann Machine，下面将引出 Restricted Boltzmann Machine。

2 Restricted Boltzmann Machine 模型表示

Boltzmann Machine 就是内部的所有节点分为可观测和不可观测的马尔可夫随机场。假设一共有 p 个节点，m 个不可观测的节点组成集合 h，n 个可观测的节点组成集合 v。即为：

$$X = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_p \end{bmatrix} = \begin{bmatrix} h \\ v \end{bmatrix}, \quad h = \begin{bmatrix} h_1 \\ h_2 \\ \vdots \\ h_m \end{bmatrix}, \quad v = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{bmatrix} \tag{5}$$

其中，$m + n = p$。Boltzmann Machine 看着好像很好，但是实际上有一些问题。首先，Inference 问题很难做，精确推断根本不可能，而近似推断基本也是 intractable。正是因为有了这些问题，我们才要想办法对模型进行简化，从而得到了 Restricted Boltzmann Machine。

2.1 Restricted Boltzmann Machine

我们只考虑 Boltzmann Machine 中，h 和 v 之间的连接，不考虑它们内部的连接。如下图所示：

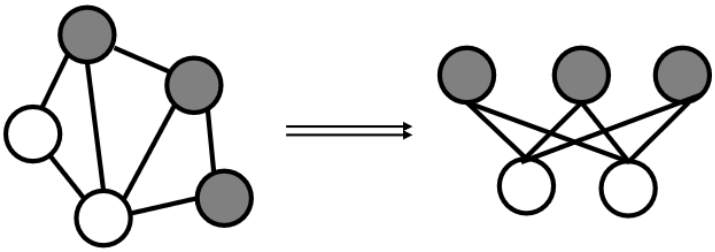

图 2: 玻尔兹曼机到受限玻尔兹曼机

我们接下来来定义能量函数的结构：

$$\begin{aligned} P(X) =& \frac{1}{Z} \exp(-\mathrm{E}(X)) \\ P(v,h) =& \frac{1}{Z} \exp(-\mathrm{E}(v,h)) \end{aligned} \tag{6}$$

接下来的问题就是如何定义 $\mathrm{E}(v,h)$。考虑到，能量函数和系统内部的每一个节点之间有关系。由于不考虑节点内部之间的关系，所以能量函数可以被分解为：h 节点中每个节点自身的影响，v 节点中每个节点自身的影响，和 h 和 v 节点之间的影响。前两者考虑的是点自身的影响，后者是考虑两个集合中的点连接的边的影响。

下一步则假设，两个集合中的点连接的边的关系用矩阵 $w = [w_{ij}]_{m\times n}$ 表示；v 集合中的点的关系参数矩阵 $\alpha = [\alpha_i]_{1\times m}$；$h$ 集合中的点的关系参数矩阵 $\alpha = [\alpha_i]_{n\times 1}$；$v$ 集合中的点的关系参数矩阵 $\beta = [\beta_i]_{m\times 1}$。然后采用线性的方法来表达 E：

$$\mathrm{E}(v,h) = -(h^T w v + \alpha^T v + \beta^T h) \tag{7}$$

求得的能量函数 $\mathrm{E}(v,h)$ 是一个一维实数。所以，联合概率为：

$$\begin{aligned} P(X) =& \frac{1}{Z}\exp(-\mathrm{E}(v,h)) = \frac{1}{Z}\exp(h^T wv + \alpha^T v + \beta^T h) \\ =& \frac{1}{Z}\exp(h^T wv)\exp(\alpha^T v)\exp(\beta^T h) \end{aligned} \tag{8}$$

其中 w,α,β 都是参数矩阵，可以利用数据来学习出来。而为什么要这样写？我们其实可以从因子图的角度来解释。因子图就是在一个图的所有点和所有边上都加一个因子，如下图所示：

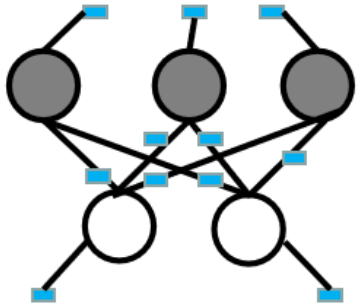

图 3: 受限玻尔兹曼机的因子图

我们可以看到因子的种类可以分成三种，可以分为一组边和两组点的因子。因子和点或者边进行组合就得到了公式 (8) 一样的形式。

2.2 Restricted Boltzmann Machine 概率密度函数

所以 Restricted Boltzmann Machine 概率密度函数的表现形式如下所示：

$$P(v,h) = \frac{1}{Z}\exp(h^T wv)\exp(\alpha^T v)\exp(\beta^T h) \tag{9}$$

而其中：

$$\exp(h^T wv) = \exp(\sum_{i=1}^{m}\sum_{j=1}^{n} h_i w_{ij} v_i) = \prod_{i=1}^{m}\prod_{j=1}^{n}\exp(h_i w_{ij} v_i) \tag{10}$$

用类似的方法进行转换，我们可以得到：

$$P(v,h) = \frac{1}{Z}\underbrace{\prod_{i=1}^{m}\prod_{j=1}^{n}\exp(h_i w_{ij} v_i)}_{\text{edge}}\underbrace{\prod_{j=1}^{n}\exp(\alpha_j v_j)}_{\text{node } v}\underbrace{\prod_{i=1}^{m}\exp(\beta_i h_i)}_{\text{node } h} \tag{11}$$

2.3 小结

本小节首先讲解了，为什么要有 Restricted Boltzmann Machine? 原因很简单，Boltzmann Machine 的复杂度太高。大家有没有觉得 Restricted Boltzmann Machine 的结构很像神经网络，它和神经网络之间有什么不可告人的秘密呢？然后从点和边的角度对其进行了分解，然后得到了它的概率密度函数。下一节将 RBM 和之前的东西结合起来，因为它本质上还是一种无向图。

3 RBM 和其他概率图模型的联系

RBM 本质上还是一种无向图，所以我们把之前的东西都总结一下联系起来，来一起看看 RBM 的发展历史。

3.1 Naive Bayes

朴素贝叶斯算法是最简单的 PGM，也是最基础的模型。此算法的核心就是朴素贝叶斯假设，或者说是条件独立假设。这个假设描述的是，当 label y 已知的情况下，各个属性之间是相互独立的。公式表达为：$x_i \perp x_j|y$。朴素贝叶斯概率图如下所示：

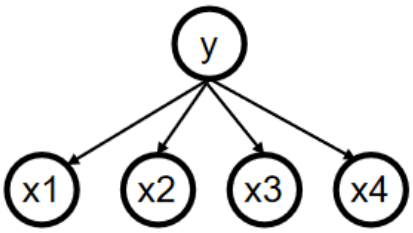

图 4: 朴素贝叶斯概率图模型

3.2 Gaussian Mixture Model

高斯混合模型中引入了隐变量，y 是一个隐变量，x 是观测变量。Gaussian Mixture Model 概率图模型如下所示：

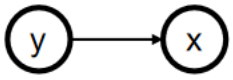

图 5: 高斯混合模型概率图模型

y 是隐变量，并且是一个离散变量，一共有 k 种选择。并且在 y 给定的情况下，x 符合一个高斯分布，即为：$P(x|y)$ ~Gaussian Distribution。在此模型中，y 是一个离散的变量，如果将其扩充为一个变量序列 (Sequence)，就演变成了 State Space Model。

3.3 State Space Model

State Space Model 的主要特点就是两个：

1. 引入了隐变量，也就是 State；

2. 符合两个假设，即为齐次马尔可夫假设和观测独立假设，这两个假设在之前都有过非常详细的介绍。

而 State Space Model，大致可以分为三类：1. Hidden Markov Model；2. Kalman Filter；3. Particle Filter。

其中，Hidden Markov Model 要求隐变量之间都是离散的；Kalman Filter 是线性高斯系统，隐变量之间的转移概率和隐变量到观测变量之间，或者说是转移矩阵和发射矩阵之间都符合高斯分布；

Particle Filter 是在 Kalman Filter 的基础上解除了线性高斯分布，认为转移矩阵和发射矩阵之间可以是很复杂的未知分布，通常采用采用的方法来近似求解。这三种模型的概率图模型都一样，如下图所示：

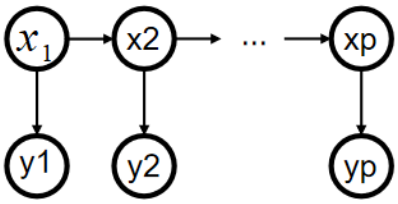

图 6: State Space Model 概率图模型

3.4 Maximum Entropy Markov Model

Logistics Regression 是一种特殊的最大熵模型，简单的说就是最大熵模型求解出的分布是指数族分布。利用最大熵与 HMM 结合，就得到了 MEMM。并且与 HMM 还有一点主要的不同就是改变了 y 与 x 之间的有向图方向。MEMM 概率图模型如下所示：

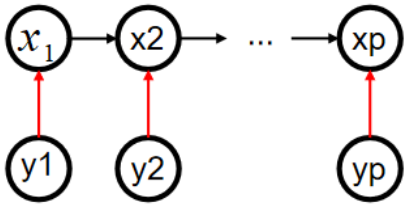

图 7: Maximum Entropy Markov Model 概率图模型

MEMM 有两条主要的性质：1. 这是一个判别模型，MEMM 主要解决的是标注问题，其中没有隐变量。2. 打破了观测独立假设。

3.5 Conditional Random Field

因为 MEMM 存在局部归一化的问题，为了解决这个问题，将 x 之间的有向图变成了无向图就得到了条件随机场。而同时也打破了齐次马尔可夫假设。同样 CRF 主要解决的是标注问题，其中没有隐变量，也是判别模型。概率图模型如下所示：

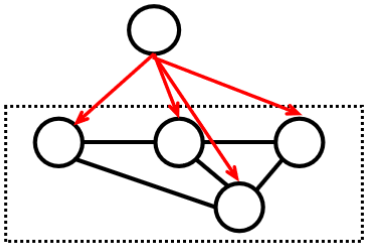

图 8: Conditional Random Field 概率图模型

但是，我们通常说的是 Linear Chain Condition Random Field (LC-CRF)，也就是马尔可夫随机场是线型的，概率图模型如下所示：

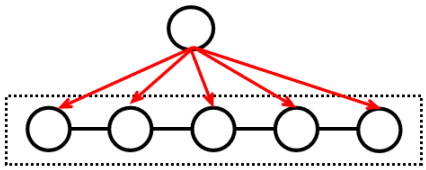

图 9: Linear Chain Condition Random Field 概率图模型

3.6 Boltzmann Machine

Boltzmann Machine 本章节已经详细的描述过了，这里不再🈳嗦了。主要三个特点：1. 无向图；2. 引入了隐变量；3. 所有节点的联合概率 PDF 必须是指数族分布，被称为 Boltzmann Distribution 或者是 Gibbs Distribution。概率图模型如下所示：

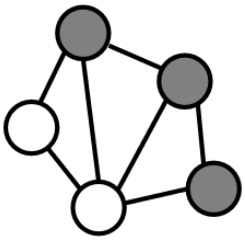

图 10: Boltzmann Machine 概率图模型

3.7 Restricted Boltzmann Machine

Boltzmann Machine 的算法复杂度太高了，假设观测节点集合内部所有的节点之间相互独立，不可观测节点集合内部所有的节点之间相互独立，就得到了 Restricted Boltzmann Machine。概率图如下所示：

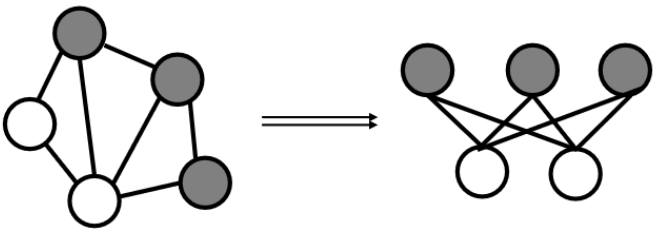

图 11: Restricted Boltzmann Machine 概率图模型

3.8 小结

概率图模型和条件独立之间有着不可划分的关系。条件独立性是尽可能的保留数据之间的结构信息，同时又简化计算。比如，朴素贝叶斯中的条件独立假设是在给定 y 的情况下，属性与属性之间是相互独立的。而 State Space Model 中的两个假设也是条件独立的。

概率图模型表示可以从以下五个方面分析：

- 方向：(有向图/无向图) 对应着 Bayesian Network 和 Markov Random Field，很显然有向图有着更强的限制。**这是从边的角度进行分析**。
- 节点的变量是离散/连续：通常情况下，无特殊说明，都认为变量是离散变量。如果，变量是连续的，则为 Gaussian Network。当然，也可以是混合的，部分为离散变量，部分为连续变量。**这是从点的角度进行分析**。
- 条件独立性：NB 中的条件独立性是在随机变量各属性之间；HMM 中就是齐次马尔可夫假设和观测独立假设上表示了条件独立性；MEMM 打破了观测独立假设，仍然是在条件独立性上做文章；RBM 的条件独立性表现在，给定观测变量的情况下，隐变量之间是条件独立的。当然，不仅属性之间可以是条件独立的，结构上也可以使条件独立的。**这是从边的角度进行分析**。
- 隐变量：是否引入隐变量也是一条重要的性质。Boltzmann Machine 和马尔可夫随机场最重要的区别，就是 Boltzmann Machine 中将节点分成两类，可观测和不可观测。**这是从点的角度进行分析**。
- PDF 是否是指数族分布：PDF 是指节点的联合概率分布函数。根据最大熵原理，在给定数据的情况下，指数族分布是使得预测分布熵最大的分布。而 BM 的 PDF 一定是一个指数族分布。如果，图结构一个小局部，或者是指数族分布，在计算上也更加有优势。

实际上，仔细回想，各种概率图模型说白了就是在这 5 个性质上进行组合，有或者没有，有的话强弱也可以不一样。概率图模型的表示，主要就是围绕这 5 点来做文章的。

4 The Inference of Restricted Boltzmann Machine

前面我们已经详细的介绍过了 Restricted Boltzmann Machine。假设一共有 p 个节点，其中 m 个不可观测的节点组成集合 h，n 个可观测的节点组成集合 v。概率图模型如下所示：

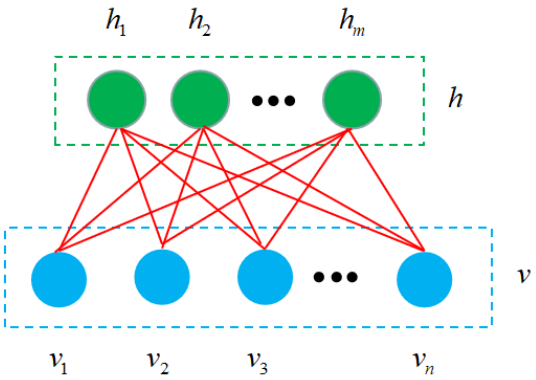

图 12: Restricted Boltzmann Machine 概率图模型

公式化表达如下所示，即为：

$$X = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_p \end{bmatrix} = \begin{bmatrix} h \\ v \end{bmatrix}, \quad h = \begin{bmatrix} h_1 \\ h_2 \\ \vdots \\ h_m \end{bmatrix}, \quad v = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{bmatrix} \tag{12}$$

其中，$m+n=p$。节点的联合概率密度函数为：

$$P(X) = \frac{1}{Z}\exp(-\mathrm{E}(X)) \Longleftrightarrow P(v,h) = \frac{1}{Z}\exp(-\mathrm{E}(v,h)) \tag{13}$$

而其中，

$$\begin{aligned} \mathrm{E}(v,h) &= -h^T wv + \alpha^T v + \beta^T h \\ &= -\left(\sum_{i=1}^{m}\sum_{j=1}^{n} h_i w_{ij} v_i + \sum_{j=1}^{n}\alpha_j v_j + \sum_{i=1}^{m}\beta_i h_i\right) \end{aligned} \tag{14}$$

个人觉得机器学习中，研究问题的主要流程基本可以拆解成，首先 1. 需要知道模型怎么表示 (Representation)；2. 然后，通过数据的学习来得到模型的参数 (Learning)；3. 最后，利用模型来对未知的数据进行推断 (Inference)。

4.1 明确 Inference 的问题

首先假设 Learning 的过程已经完成，那么所有的参数我们都已经知道了，所以我们已知的有：

1. 所有的势函数，这样归一化因子就知道了；
2. 能量函数；
3. 知道能量函数就知道 h 和 v 的联合概率分布 $P(v,h)$。

需要 Inference 的是三个问题：

1. $P(h|v)$；
2. $P(v|h)$；
3. $P(v)$。

然而，为什么不求 $P(v)$ 呢？实际上没什么必要，我们更多的是关注在已知 v 的情况下，$P(h|v)$ 的条件概率分布。$P(h)$ 中 h 反正也是不可观测的，求了边缘概率分布也没什么用。如果要求解的话和求解 $P(v)$ 的方法一样。

4.1.1 求解 $P(h|v)$ and $P(v|h)$

$P(h|v)$ 和 $P(v|h)$ 求解方法都是一样的，这里就放在一起推导。

$P(h|v)$ 求解的是 v 中所有节点都知道的情况下，集合 h 的联合概率分布，即为：

$$P(h_1, h_2, \cdots, h_m|v)$$

那么，首先我们就要根据条件独立性来对联合概率分布进行化简。那么，我们想想 $h_i \perp h_j|v,\ i \neq j$ 是成立的吗？其实一看就知道是成立的。为什么呢？无向图满足局部马尔可夫性质，这个性质的意思

是，当无向图中一个节点，除这个节点以外的所有节点都知道的话，这个节点只和他的邻居有关，和其他节点都是独立的。也就是$P(h_i|-h_i,v)=P(h_i|$ **邻居节点** $)=P(h_i|v)$**（可以通过概率无向图进行观察，给定 v 的情况下** h_{-i} **和** h_i **的路径被阻塞）**。根据：

$$P(h_i|-h_i,v)=P(h_i|v)$$

我们就可以得到 $h_i \perp h_j|v,\ i\neq j$。所以，根据条件独立性，联合概率（联合后验概率）可以被简化为：

$$P(h|v)=\prod_{l=1}^{m}P(h_l|v) \tag{15}$$

为了简化，我们假设无向图所有节点都是二值的，也就是 $h,v\in\{0,1\}$。实际上 RBM 的节点是离散变量，而 0/1 分布是最简单的离散分布。但是，为了详细的解析，这里用了简单的分布来进行解析，其他的离散分布形式可以看成是 0/1 分布的变种。

那么，假设我们要求的是 $P(h_l=1|v)$，我们已经知道的是 $P(v,h)$。那么自然就想到将 h 补齐，我们用 h_{-l} 来表示除 h_l 外的所有节点，所以有：

$$\begin{aligned}P(h_l=1|v)=&P(h_l=1|h_{-l},v)=\frac{P(h_l=1,h_{-l},v)}{P(h_{-l},v)}=\frac{P(h_l=1,h_{-l},v)}{\sum_{h_l}P(h_l,h_{-l},v)}\\=&\frac{P(h_l=1,h_{-l},v)}{P(h_l=1,h_{-l},v)+P(h_l=0,h_{-l},v)}\end{aligned}$$

那么，怎么求解呢？首先看分子怎么求。$P(h_l=1,h_{-l},v)$ 非常的特殊，和联合概率分布不一样的地方在于其中某一个变量的状态是已知的。那么我们把这个变量从联合概率中分解出来，赋予具体的值就可以了。

于是，我们的下一步操作就是对能量函数进行改写，将 h_l 相关的项解析出来。实际上就是和 h_l 自己和相关的边有关。

$$\begin{aligned}\mathrm{E}(v,h)=&-\left(\sum_{i=1}^{m}\sum_{j=1}^{n}h_iw_{ij}v_i+\sum_{j=1}^{n}\alpha_jv_j+\sum_{i=1}^{m}\beta_ih_i\right)\\=&\left(\underbrace{\sum_{i=1,i\neq l}^{m}\sum_{j=1}^{n}h_iw_{ij}v_i}_{\triangle_1}+\underbrace{\sum_{j=1}^{n}h_lw_{lj}v_i}_{\triangle_2}+\underbrace{\sum_{j=1}^{n}\alpha_jv_j}_{\triangle_3}+\underbrace{\sum_{i=1,i\neq l}^{m}\beta_ih_i}_{\triangle_4}+\underbrace{\beta_lh_l}_{\triangle_5}\right)\end{aligned} \tag{16}$$

令 $H_l(v)=\triangle_2+\triangle_5$，表示和 h_l 相关的部分，很显然因为 h_l 已知，不含和 h 相关的部分了；

令 $\bar{H}_l(h_{-l},v)=\triangle_1+\triangle_3+\triangle_4$，表示和 h_l 不相关的部分；所以：

$$H_l(v)=\triangle_2+\triangle_5=h_l\left(\sum_{j=1}^{n}w_{lj}v_i+\beta_l\right)$$

我们将 $\sum_{j=1}^{n}w_{lj}v_i+\beta_l$ 定义为 $H_l(v)$，所以：

$$\mathrm{E}(v,h)=h_lH_l(v)+\bar{H}_l(h_{-l},v) \tag{17}$$

那么，分子为：

$$P(h_l = 1, h_{-l}, v) = \frac{1}{Z} \exp\left\{h_l(v) + \bar{H}_l(h_{-l}, v)\right\} \tag{18}$$

那么，分母为：

$$P(h_l = 1, h_{-l}, v) + P(h_l = 0, h_{-l}, v) = \frac{1}{Z} \exp\left\{h_l(v) + \bar{H}_l(h_{-l}, v)\right\} + \frac{1}{Z} \exp\left\{\bar{H}_l(h_{-l}, v)\right\} \tag{19}$$

所以，

$$\begin{aligned} P(h_l = 1|v) =& \frac{P(h_l = 1, h_{-l}, v)}{P(h_l = 1, h_{-l}, v) + P(h_l = 0, h_{-l}, v)} \\ =& \frac{\frac{1}{Z} \exp\left\{h_l(v) + \bar{H}_l(h_{-l}, v)\right\}}{\frac{1}{Z} \exp\left\{h_l(v) + \bar{H}_l(h_{-l}, v)\right\} + \frac{1}{Z} \exp\left\{\bar{H}_l(h_{-l}, v)\right\}} \\ =& \frac{1}{1 + \exp\left\{\bar{H}_l(h_{-l}, v) - h_l(v) - \bar{H}_l(h_{-l}, v)\right\}} \\ =& \frac{1}{1 + \exp\left\{-h_l(v)\right\}} \end{aligned} \tag{20}$$

而 $\frac{1}{1+\exp\{-h_l(v)\}}$ 实际就是 Sigmoid 函数，Sigmoid 函数的表达形式为：$\sigma(x) = \frac{1}{1+e^{-x}}$。

$$P(h_l = 1|v) = \sigma(h_l(v)) = \sigma(\sum_{j=1}^{n} w_{lj} v_i + \beta_l) \tag{21}$$

既然已经求得了 $P(h_l|v)$，根据公式 (15) 就可以得到 $P(h|v)$ 的结果了：

$$P(h|v) = \prod_{l=1}^{m} P(h_l|v) = \left(\sigma\left(\sum_{j=1}^{n} w_{lj} v_i + \beta_l\right)\right)^k \left(1 - \sigma\left(\sum_{j=1}^{n} w_{lj} v_i + \beta_l\right)\right)^{m-k} \tag{22}$$

其中 k 为 h 集合中，$h_l = 1$ 的节点数。

已经成功求得了 $P(h|v)$，那么求解 $P(v|h)$ 的过程是一模一样的，基本上可以做一个转换，直接得到结果：

$$P(v|h) = \prod_{l=1}^{m} P(h_l|v) = \left(\sigma\left(\sum_{j=1}^{m} w_{jl} h_j + \alpha_l\right)\right)^k \left(1 - \sigma\left(\sum_{j=1}^{m} w_{jl} h_j + \alpha_l\right)\right)^{n-k} \tag{23}$$

其中 k 为 v 集合中，$v_l = 1$ 的节点数。

那么，到这里对于后验的计算已经完成了，后验实际上就是 Sigmoid 函数。大家有没有觉得 RBM 和神经网络很像，我其实早就有这种感觉了，不可观测节点不就是隐藏层。而 Sigmoid 函数，经常被用来当做神经网络的激活函数。这之间是巧合还是有必然的联系呢？后面的章节我们会有分析的，神经网络实际上是从 RBM 中发展得到的。

4.2 求解 $P(v)$(inference→ marginal → P_v)

这一小节，我们的目标是通过 Inference 来求解 Marginal Distribution $P(v)$。思路很简单，既然我们知道联合概率分布 $P(v,h)$，那么把 h 节点的变量积分掉不就可以了，所以：

$$\begin{aligned} P(v) =& \sum_h P(v,h) = \sum_h \frac{1}{Z}\exp\left\{-\mathrm{E}(v,h)\right\} = \frac{1}{Z}\sum_h \exp(h^T wv + \alpha^T v + \beta^T h) \\ =& \frac{1}{Z}\sum_{h_1}\cdots\sum_{h_m}\exp(h^T wv + \alpha^T v + \beta^T h) \\ =& \frac{1}{Z}\exp(\alpha^T v)\sum_{h_1}\cdots\sum_{h_m}\exp(h^T wv + \beta^T h) \end{aligned} \tag{24}$$

我们下一步的目标就是将等式 (24) 中的所有和 h 相关的项提取出来，分别进行计算。为了方便计算，我们令：

$$h = \begin{bmatrix} h_1 \\ h_2 \\ \vdots \\ h_m \end{bmatrix},\ h_l \in \{0,1\} \qquad w = [w_{ij}]_{m\times n} = \begin{bmatrix} ---w_1^T--- \\ ---w_2^T--- \\ \vdots \\ ---w_m^T--- \end{bmatrix} \tag{25}$$

这里的 w，我们用 m 个行向量来表示。那么，

$$h^T wv = \begin{bmatrix} h_1 & h_2 & \cdots & h_m \end{bmatrix} \begin{bmatrix} ---w_1^T--- \\ ---w_2^T--- \\ \vdots \\ ---w_m^T--- \end{bmatrix} v = \sum_{i=1}^m h_i w_i^T v$$

$$\beta^T h = \sum_{i=1}^m \beta_i h_i$$

所以，

$$\begin{aligned} P(v) =& \frac{1}{Z}\exp(\alpha^T v)\sum_{h_1}\cdots\sum_{h_m}\exp\left(\sum_{i=1}^m h_i w_i^T v + \sum_{i=1}^m \beta_i h_i\right) \\ =& \frac{1}{Z}\exp(\alpha^T v)\sum_{h_1}\cdots\sum_{h_m}\exp\left(\sum_{i=1}^m (h_i w_i^T v + \beta_i h_i)\right) \\ =& \frac{1}{Z}\exp(\alpha^T v)\sum_{h_1}\cdots\sum_{h_m}\exp\left(\sum_{i=1}^m (h_i w_i^T v + \beta_i h_i)\right) \\ =& \frac{1}{Z}\exp(\alpha^T v)\sum_{h_1}\cdots\sum_{h_m}\exp\left((h_1 w_1^T v + \beta_1 h_1) + (h_2 w_2^T v + \beta_2 h_2)\cdots(h_m w_m^T v + \beta_m h_m)\right) \\ =& \frac{1}{Z}\exp(\alpha^T v)\sum_{h_1}\exp(h_1 w_1^T v + \beta_1 h_1)\sum_{h_2}(h_2 w_2^T v + \beta_2 h_2)\cdots\sum_{h_m}(h_m w_m^T v + \beta_m h_m) \end{aligned} \tag{26}$$

由于 $h_l \in \{0,1\}$，所以，

$$\begin{aligned} P(v) =& \frac{1}{Z}\exp(\alpha^T v)\sum_{h_1}\exp(h_1 w_1^T v + \beta_1 h_1)\sum_{h_2}(h_2 w_2^T v + \beta_2 h_2)\cdots\sum_{h_m}(h_m w_m^T v + \beta_m h_m) \\ =& \frac{1}{Z}\exp(\alpha^T v)\left[1+\exp(w_1 v+\beta_1)\right]\cdots\left[1+\exp(w_m^T v+\beta_m)\right] \\ =& \frac{1}{Z}\exp(\alpha^T v)\exp\left[\log(1+\exp(w_1^T v+\beta_1))\right]\cdots\exp\left[\log(1+\exp(w_m^T v+\beta_m))\right] \\ =& \frac{1}{Z}\exp\left(\alpha^T v+\sum_{i=1}^{m}\log(1+\exp(w_i^T v+\beta_i))\right) \end{aligned} \tag{27}$$

而 $\log(1+\exp(w_i^T v+\beta_i))$ 是一种 softplus 函数的形式，softplus 函数可以描述为：$\text{softplus}(x) = \log(1+e^x)$，函数图像如下所示：

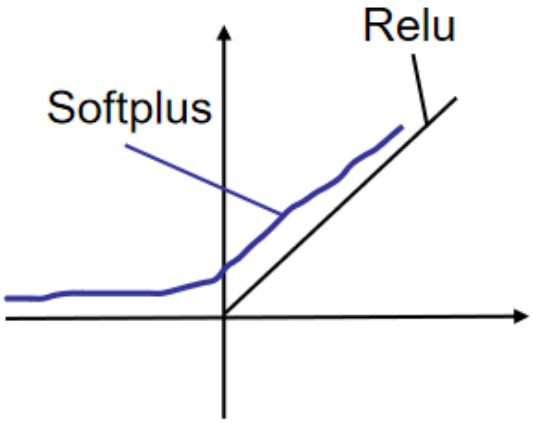

图 13: Softplus 函数图像

我们可以看到此函数在正半轴越来越接近 Relu 函数，所以，接着公式 (27) 继续向下推导得：

$$\begin{aligned} P(v) =& \frac{1}{Z}\exp\left(\alpha^T v+\sum_{i=1}^{m}\log(1+\exp(w_i^T v+\beta_i))\right) \\ =& \frac{1}{Z}\exp\left(\alpha^T v+\sum_{i=1}^{m}\text{softplus}(w_i^T v+\beta_i)\right) \end{aligned} \tag{28}$$

那么，就可以求得关于 v 的边缘概率分布了。v 可能有 k 种状态，将每种状态的具体值代入即可。最后在总结一下：

$$P(v) = \frac{1}{Z}\exp\left(\alpha^T v+\sum_{i=1}^{m}\text{softplus}(w_i^T v+\beta_i)\right) \tag{29}$$

其中，w_i^T 为 w 矩阵的行向量。

4.3 小结

在本小节中，我们主要计算了三个推断问题，$P(v|h)$，$P(h|v)$，$P(v)$。计算结果如下所示：

$$
\begin{aligned}
P(v|h) &= \prod_{l=1}^{m} P(v_l|h) = \left(\sigma\left(\sum_{j=1}^{m} w_{jl}h_j + \alpha_l\right)\right)^k \left(1 - \sigma\left(\sum_{j=1}^{m} w_{jl}h_j + \alpha_l\right)\right)^{n-k} \\
P(h|v) &= \prod_{l=1}^{m} P(h_l|v) = \left(\sigma\left(\sum_{j=1}^{n} w_{lj}v_i + \beta_l\right)\right)^k \left(1 - \sigma\left(\sum_{j=1}^{n} w_{lj}v_i + \beta_l\right)\right)^{m-k} \\
P(v) &= \frac{1}{Z}\exp\left(\alpha^T v + \sum_{i=1}^{m} \text{softplus}(w_i^T v + \beta_i)\right)
\end{aligned}
\tag{30}
$$

我看计算的思路都差不多，都是把已知条件分类出来，然后赋予具体的值。我们采用的离散分布是 0/1 分布是为了简化计算，当值具有多个时，计算的思路也是一样的。**但是，无论可能的取值变成多少个，整体还是符合指数族分布的**。

5 Conclusion

本章节，主要描述了 Restricted Boltzmann Machine。主要的讲述思路是先从马尔可夫随机场中引出了 Boltzmann Machine，介绍了什么是 Boltzmann Machine；然后描述了 Boltzmann Machine 的计算 intractable，然后引出了 Restricted Boltzmann Machine；紧接着介绍了 Restricted Boltzmann Machine 模型表示方法，并讲述了它在概率图模型整体结构中的地位；最后讲述了如何用 Restricted Boltzmann Machine 来进行推断。

大家可能发现，在使用模型 Inference 的时候，需要通过 Learning 来从数据中得到参数的值，这部分会在之后的直面配分函数中描述。

Spectral Clustering

Chen Gong

02 March 2020

目录

1 Background

本章节主要是描述的一种聚类算法，谱聚类 (Spectral Clustering)。对机器学习有点了解的同学对聚类算法肯定是很熟悉的，那么谱聚类和之前普通的聚类算法有什么不一样呢？或者说它有什么优势呢？

1.1 聚合型聚类 (Compactness)

常见的聚类方法有两种思路，一种就是聚合型聚类 (Compactness)。典型的算法有 K-means 和 Gaussian Mixture Model 这种。GMM 我们在前面的章节中有详细的描述，GMM Clustering 的思想可以这样来表述。我们首先看到 GMM 的概率图模型，如下所示：

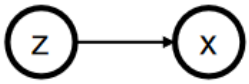

图 1: GMM 的概率图模型

GMM 从几何角度来看，就是也就是多个高斯分布来取加权平均值。我们想将样本分成多少类，那么 Z 就有多少种可能，假设我们需要将其分成 N 类。那么 Z 是一个离散变量，$Z \in \{1, 2, 3, \cdots, N\}$，当 Z 取每次取不同的值时都对应着一个不同的高斯分布，公式化表达为：$P(x|z) \sim \mathcal{N}(\mu, \sigma^2)$。那么对于一个样本，我们可以计算出它属于每一个类别的不同的概率，从中选取概率最大的即可。GMM 举例如下图所示：

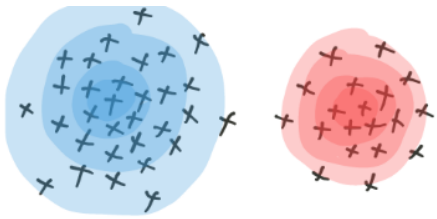

图 2: GMM 的概率图模型

GMM 相关的具体知识，包括 GMM 的定义和 EM 算法进行求解等，请阅读小编之前写的“白板推导高斯混合模型”。很明显，我们看到 GMM 的边界轮廓都是圆的，学术的讲就是“凸”(Convex) 的。**这样的考虑忽略了数据之间的结构关系，主要考虑的是特征之间的相似度，而且是一种中心的距离方式**。而这时候我们需要引出另一种聚类算法的思路了，连通性 (Connectivity)。

1.2 连通性聚类 (Connectivity)

连通性聚类 (Connectivity) 算法的典型代表就是谱聚类 (Spectral Clustering)。比如，下面的数据分布，很显然用 Spectral Clustering 的方法来聚类成如下的形式更加的合理。如下图所示。**很显然这是一个 non-convex 的聚类方式，更加注重的是数据分布之间的连通性，并且利用图结构考虑到了数据内部的结构性特点，目标是使不同的类别数据点分的越开越好，至于为什么？请接着往下看 Spectral Clustering 的模型描述**。

图 3: Spectral Clustering 示例

1.3 小结

聚合型聚类 (Compactness) 和连通性聚类 (Connectivity) 方法各有千秋。对于“凸”型形状，聚合型聚类 (Compactness) 更好一些，主要考虑的是特征之间的相似度。在复杂情况下，结合 Kernel 的做法可以简化计算，这个在“核技巧”那章有详细的说明。而连通性聚类 (Connectivity) 方法，主要考虑的是数据的结构上的特点，适合随意的形状。

2 Spectral Clustering 的模型表示

2.1 Spectral Clustering 参数说明

Spectral Clustering 的模型表示实现手段上是使用基于无向带权图的思想。我们将所有的数据样本都分别当成无向图中的一个节点，我们假设概率图模型为：

$G = \{V, E\}$；

$V = \{1, 2, 3, \cdots, N\}$：无向图中每一个节点代表一个数据样本，图中有 N 个节点，代表有 N 个样本。

$X = \left[x_1, x_2, \cdots, x_N\right]^T = \begin{bmatrix} x_1^T \\ x_2^T \\ \vdots \\ x_N^T \end{bmatrix}_{N \times p}$：代表有 N 个样本，每个样本都是 p 维的。而 V 中的 i 就对应着 x_i，表示第 i 个样本。

$E : [w_{ij}]_{n \times n}$：这个被称为相似度矩阵，也就是用来表示样本与样本之间的权重。只有点与点之间存在边，才有权重，否则为 0。比如如下图所示的一个概率图结构：

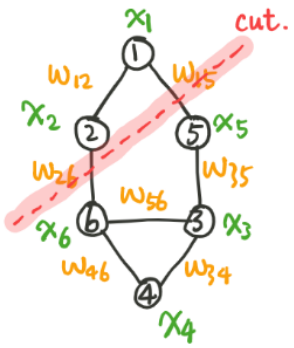

图 4: Spectral Clustering 示例

其中：

$$w_{ij} = \begin{cases} k(x_i, x_j) = \exp\left\{-\frac{||x_i - x_j||_2^2}{2\sigma^2}\right\} & (i,j) \in E \\ 0 & (i,j) \neq E \end{cases}$$

其实在概率图中体现就是，有连接的地方就有权重，没有连接的地方就没有权重。所以，这样就可以反映数据的条件独立性和结构特点，只有某些点之间才有联系，而不是像 GMM 那样，默认所有点之间都有联系。

2.2 Spectral Clustering 优化目标

定义：$A \subset V$，$B \subset V$，$A \cap B = \varnothing$，$W(A,B) = \sum_{i \in A} \sum_{j \in B} w_{ij}$。这个公式描述的是将节点分成两份，每个节点都只能属于其中一类，他们之间的距离定义为，一个集合中的每一个节点，分别到另一个集合中的所有节点的相似度的和。

如果，想将节点分解成 K 类，那么公式化描述如下所示：

$$\begin{aligned} &\text{cut}(V) = \text{cut}(A_1, A_2, \cdots, A_K) \\ &\begin{cases} V = \bigcup_{k=1}^{K} A_k \\ A_i \cap A_j = \varnothing, \ \forall i,j \in \{1,2,\cdots,K\} \end{cases} \end{aligned} \tag{1}$$

并且令：

$$\text{cut}(V) = \sum_{k=1}^{K} W(A_k, \bar{A}_k) = \sum_{k=1}^{K} \sum_{p \in A_k} \sum_{q \notin A_k} w_{pq} \tag{2}$$

其中，$\bar{A}_k$ 表示出了 A_k 以外的所有子集构成的集合。那么这个公式的意思就是，每个集合中的所有点，分别到其他集合中的所有点的相似度的和。那么我们的目标函数，首先可以定义为最小化这个 cut 函数。因为**该值与聚类的目标一致，即每个子图内部的连接很强，而子图之间的连接很弱，换一种语言来表述就是同一个子图内的样本相似，不同子图之间的样本不相似**。即为：

$$\min_{\{A_k\}_{i=k}^{K}} \text{cut}(V) \tag{3}$$

其实，也就是要将 V 划分 K 类，使得 cut 函数值最小，实际上就是一个组合优化问题。

2.3 Spectral Clustering 优化目标改进

但是，我们观察上述的式子，实际上是有不妥的地方的。比如，图四中，有一类为 2 个节点，另一类为 4 个节点。**但直接通过最小化这个值实现聚类还有问题，它没有考虑子图规模对代价函数的影响，使得这个指标最小的切分方案不一定就是最优切割**。因此需要对代价函数进行归一化。

第一种最简单的思路，既然我们需要考虑子图的规模，就是除以集合中点的个数（优化类别的平均代价之和），即为：

$$\text{cut}(V) = \sum_{k=1}^{K} \frac{W(A_k, \bar{A}_k)}{|A_k|} \tag{4}$$

但是，这样做仍然不妥，因为这样即使考虑了集合中点的个数。子图的复杂度还需要考虑子图中节点的连接情况，而不是仅仅考虑子图的个数即可。考虑子图中的连接也侧面包含了子图的节点个数所带来的复杂度。为了量化连接情况，这里引入一个概念为“度”：这是离散数学中的定义，有向图中

分“出度”和“入度”，其实很简单“出度”就是这个节点指向了几个节点，“入度”就是有几个节点指向这个节点。无向图中我们只用度的概念。

而在带权无向图中，一个节点的度即为与这个节点连接的所有的边的权重和。利用度作为归一化因子就比较全面的考虑了子图的复杂度了。

$$\mathrm{degree(A_k)} = \sum_{i \in A_k} d_i, \quad d_i = \sum_{j=1}^{N} w_{ij} \tag{5}$$

那么就得到了，最终的优化目标为：

$$\min_{\{A_k\}_{k=1}^K} \mathrm{Ncut}(V) = \sum_{k=1}^{K} \frac{W(A_k, \bar{A}_k)}{\sum_{i \in A_k} d_i}, \quad d_i = \sum_{j=1}^{N} w_{ij} \tag{6}$$

2.4 小结

本小节我们定义了谱聚类的目标函数，该函数值反映的聚类的目标为，同一个子图内的样本相似，不同子图之间的样本不相似。因为需要考虑子图的复杂度，我们对目标函数进行了优化。考虑子图中的连接情况，用“度”的概念来定义子图的复杂度。从而通过引入“度”来表示对子图复杂度的影响，来优化目标函数。

3 目标函数的矩阵表达形式

通过上一小节，我们得到了算法的目标函数为：

$$\{\hat{A}_k\}_{k=1}^K = \arg \min_{\{A_k\}_{k=1}^K} \sum_{k=1}^{K} \frac{W(A_k, \bar{A}_k)}{\sum_{i \in A_k} \sum_{j=1}^{N} w_{ij}} \tag{7}$$

因为求解的过程中，连加符号并不方便进行各种运算操作。所以，我们要把目标函数转换为矩阵形式，从而方便计算。首先，这个 $\{\hat{A}_k\}_{k=1}^K$(表示的是将所有的点进行划分后的结果) 看着就不好进行运算，所以第一步想办法把这个变成矩阵化表示。

3.1 指示向量 (Indicator vector)

令：

$$\begin{cases} y_i \in \{0,1\}^K \\ \sum_{j=1}^{K} y_{ij} = 1 \end{cases}$$

这个公式想要表达的意思为，y_i 是一个 K 维向量，每一个维度的值要么是 0，要么是 1。$y_{ij} = 1$ 表示第 i 个样本属于第 j 个类别，而每个样本只能属于一个类别，所以 $\sum_{j=1}^{K} y_{ij} = 1$，$1 \le i \le N$，$1 \le j \le K$。那么使用 Indicator vector 后，目标函数为：

$$Y = (y_1, y_2, \cdots, y_N)_{N \times K}^T$$

$$\hat{Y} = \arg\min_{Y} \sum_{k=1}^{K} \frac{W(A_k, \bar{A}_k)}{\sum_{i \in A_k} \sum_{j=1}^{N} w_{ij}}$$

利用指示函数将 $\{\hat{A}_k\}_{k=1}^K$ 变成矩阵以后，下一步目标就是将后面那一坨改写成矩阵形式。

3.2 对角矩阵

在使用只是向量后，我们成功的将目标函数化简成了：

$$
\begin{aligned}
&Y = (y_1, y_2, \cdots, y_N)^T_{N\times K} \\
&\hat{Y} = \arg\min_{Y} \sum_{k=1}^{K} \frac{W(A_k, \bar{A}_k)}{\sum_{i\in A_k} d_i}, \quad d_i = \sum_{j=1}^{N} w_{ij}
\end{aligned} \tag{8}
$$

我们的下一步目标就是想想如何将 $\sum_{k=1}^{K} \frac{W(A_k,\bar{A}_k)}{\sum_{i\in A_k} d_i}$ 表达成矩阵形式。**求和符号可以被我们写做对角矩阵的 trace**；所以有：

$$
\begin{aligned}
\mathrm{Ncut}(V) =& \sum_{k=1}^{K} \frac{W(A_k, \bar{A}_k)}{\sum_{i\in A_k} d_i} = \mathrm{tr}\left(\begin{bmatrix} \frac{W(A_1,\bar{A}_1)}{\sum_{i\in A_1} d_i} & 0 & \cdots & 0 \\ 0 & \frac{W(A_2,\bar{A}_2)}{\sum_{i\in A_2} d_i} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \frac{W(A_K,\bar{A}_K)}{\sum_{i\in A_K} d_i} \end{bmatrix}_{K\times K} \right) \\
=& \mathrm{tr}\left(\underbrace{\begin{bmatrix} W(A_1,\bar{A}_1) & 0 & \cdots & 0 \\ 0 & W(A_2,\bar{A}_2) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & W(A_K,\bar{A}_K) \end{bmatrix}}_{O_{K\times K}} \underbrace{\begin{bmatrix} \sum_{i\in A_1} d_i & 0 & \cdots & 0 \\ 0 & \sum_{i\in A_2} d_i & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sum_{i\in A_K} d_i \end{bmatrix}^{-1}}_{P_{K\times K}} \right) \\
=& \mathrm{tr}(OP^{-1})
\end{aligned} \tag{9}
$$

其中，O 和 P 都是对角矩阵。**首先我们需要明确一下，哪些变量是知道的，1. 相似度矩阵：$W = [w_{ij}]_{n\times n}$；2. 指示变量：Y(Y 每次都是已知的，我们的目标就是众多种可能性中，找使得目标函数最大的 Y)**。所以，我们的目标是用 W 和 Y 来表示 O 和 P。

首先来看看如何表示 P 矩阵。

$\sum_{i\in A_k} d_i$ 代表的是，第 k 类集合中，所有的节点的度的和。而无向图中的度就是一个节点与其他所有节点所连接的边的权重之和。而 $W = [w_{ij}]_{n\times n}$ 矩阵中 w_{ij} 代表第 i 个节点与第 j 个节点的权值，所以 W 矩阵，**第 i 行的所有值的和正好就是第 i 个节点与其他所有节点连接的边的权重之和**。

所以，计算方法就是将 W 矩阵，每一行的所有值都相加。度的对角矩阵可以描述为：

$$
D = \mathrm{diag}\left[W \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}_{N\times 1} \right] = \begin{bmatrix} d_1 & & & \\ & d_2 & & \\ & & \ddots & \\ & & & d_N \end{bmatrix} \tag{10}
$$

那么，下一步目标就是将这些度分别按划分的类别进行求和，也就是比如 $A_1 = \{1, 2, 5\}$，那么就要把第 1,2,5 三个节点的度加在一起。那么，应该如何去实现呢?

Y 矩阵是 $N \times K$ 维的，行代表节点标号，列代表属于的类别。那么，$y_{ij} = 1$ 表示第 i 个样本属于第 j 个类别。那么，我们从 Y 中取出第 i 行向量 $(1 \times K)$ 来，那么通过这一行向量，根据第几列的值等于 1，我们可以看出这个样本属于第几类。假设这个行向量第 j 列等于 0；那么 $y_i^T y_i$ 结果为：

$$y_i^T y_i = \begin{bmatrix} 0 \\ 0 \\ \cdots \\ y_{ij} = 1 \\ \cdots \\ 0 \end{bmatrix} \begin{bmatrix} 0 & 0 & \cdots & y_{ij} = 1 & \cdots & 0 \end{bmatrix} = \begin{bmatrix} 0 & \cdots & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ 0 & \cdots & y_{jj} = 1 & \cdots & 0 \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & \cdots & 0 \end{bmatrix} \tag{11}$$

那么，很显然 $y_i^T D y_i$ 求得的是一个 $K \times K$ 的矩阵，而这个矩阵中只有一个元素不为零，其他的元素都为零。如果 y_i 样本属于第 k 类，那么这个元素的位置为第 k 行，第 k 列，元素的值为 d_i。通过这样的运算，**我们成功的实现了，如果第 i 个样本属于第 k 类，就将这个样本的度 d_i，放在了 $K \times K$ 的矩阵第 k 行，第 k 列的位置。而且这个矩阵一定是对角矩阵**。

有 N 个样本就算有 N 个这样的矩阵，把这 N 个矩阵加起来就实现了将所有的 d_i 按所属的类别求和的工作。也就是：

$$\sum_{i=1}^{N} y_i d_i y_i^T = Y^T D Y \tag{12}$$

这个式子的计算可以这样考虑：先算 $y_i d_i$，也就是 $(y_1 d_1, y_2 d_2, \cdots, y_N d_N)$，可以表示成：

$$(y_1, y_2, \cdots, y_N) diag(d_1, d_2, \cdots, d_N) \tag{13}$$

所以，我们就得到了用 Y 和 W 对 P 矩阵的进行表达的形式为：

$$P = Y^T \text{diag}(W \cdot \mathbf{1_N}) Y \tag{14}$$

其中，$\mathbf{1_N}$ 元素全为 1 的 $N \times 1$ 维向量。

3.3 拉普拉斯矩阵 (Lapalacian Matrix)

成功将 P 表示以后，下一步目标则是想办法表示 O 矩阵。

$$\begin{bmatrix} W(A_1, \bar{A}_1) & 0 & \cdots & 0 \\ 0 & W(A_2, \bar{A}_2) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & W(A_K, \bar{A}_K) \end{bmatrix}_{K \times K} \tag{15}$$

而 $\bar{A}_i$ 代表的是，整个样本集中去除 A_i 中的样本后的所有样本。那么，可以做下列变换：

$$W(A_k, \bar{A}_k) = W(A_k, V) - W(A_k, A_k) \tag{16}$$

根据 $W(\cdot)$ 函数的定义，$W(A, B)$ 函数表示的是 A 中所有点分别到 B 中所有点的相似度的和，即为：

$$W(A_i, A_j) = \sum_{p \in A_i} \sum_{q \notin A_j} w_{pq} \tag{17}$$

所以，$W(A_k,V)=\sum_{i\in A_k}d_i$，$W(A_k,A_k)=\sum_{i\in A_k}\sum_{j\in A_k}w_{ij}$。而 $W(A_k,V)=\sum_{i\in A_k}d_i$ 是 $Y^T\mathrm{diag}(W\cdot 1_N)Y$ 这一对角矩阵对角上的一个元素，下一步只要想办法表达 $W(A_k,A_k)=\sum_{i\in A_k}\sum_{j\in A_k}w_{ij}$ 即可。我们的目标是求解同一个类别中，任意两个不同的样本之间的相似度。这其实和上一个求 P 的问题非常的类似，那么首先来看看 Y^TWY 会等于什么，因为 $Y=(y_1,y_2,\cdots,y_N)^T_{N\times K}$，所以：

$$\begin{aligned}Y^TWY&=\begin{bmatrix}y_1 & y_2 & \cdots & y_N\end{bmatrix}\begin{bmatrix}w_{11} & w_{12} & \cdots & w_{1N}\\ w_{21} & w_{22} & \cdots & w_{2N}\\ \vdots & \vdots & \ddots & \vdots\\ w_{N1} & w_{N2} & \cdots & w_{NN}\end{bmatrix}\begin{bmatrix}y_1^T\\ y_2^T\\ \vdots\\ y_N^T\end{bmatrix}\\&=\begin{bmatrix}\sum_{i=1}^N y_iw_{i1} & \cdots & \sum_{i=1}^N y_iw_{iN}\end{bmatrix}\begin{bmatrix}y_1^T\\ y_2^T\\ \vdots\\ y_N^T\end{bmatrix}\\&=\sum_{j=1}^N\sum_{i=1}^N y_iw_{ij}y_j^T\end{aligned}\tag{18}$$

有因为 w_{ij} 是一个一维实数，所以，$Y^TWY=\sum_{j=1}^N\sum_{i=1}^N y_iw_{ij}y_j^T=\sum_{j=1}^N\sum_{i=1}^N y_iy_j^Tw_{ij}$。那么，我们考虑一下 $y_iy_j^T$ 等于什么。y_i 表示的是一个样本属于第几类，那么 $y_i\in A_p$，$y_i\in A_q$，则 $y_iy_j^T$ 得到的矩阵中第 i 行，第 j 列的元素值为 1，其余的全部为 0。所以：

$$Y^TWY=\sum_{j=1}^N\sum_{i=1}^N y_iw_{ij}y_j^T=\begin{bmatrix}\sum_{i\in A_1}\sum_{j\in A_1}w_{ij} & \sum_{i\in A_1}\sum_{j\in A_2}w_{ij} & \cdots & \sum_{i\in A_1}\sum_{j\in A_k}w_{ij}\\ \sum_{i\in A_2}\sum_{j\in A_1}w_{ij} & \sum_{i\in A_2}\sum_{j\in A_2}w_{ij} & \cdots & \sum_{i\in A_2}\sum_{j\in A_k}w_{ij}\\ \vdots & \vdots & \ddots & \vdots\\ \sum_{i\in A_k}\sum_{j\in A_1}w_{ij} & \sum_{i\in A_k}\sum_{j\in A_2}w_{ij} & \cdots & \sum_{i\in A_k}\sum_{j\in A_k}w_{ij}\end{bmatrix}\tag{19}$$

而：

$$\begin{aligned}\begin{bmatrix}W(A_1,\bar{A}_1) & 0 & \cdots & 0\\ 0 & W(A_2,\bar{A}_2) & \cdots & 0\\ \vdots & \vdots & \ddots & \vdots\\ 0 & 0 & \cdots & W(A_K,\bar{A}_K)\end{bmatrix}&=\begin{bmatrix}W(A_1,V) & 0 & \cdots & 0\\ 0 & W(A_2,V) & \cdots & 0\\ \vdots & \vdots & \ddots & \vdots\\ 0 & 0 & \cdots & W(A_K,V)\end{bmatrix}\\&\quad-\begin{bmatrix}W(A_1,A_1) & 0 & \cdots & 0\\ 0 & W(A_2,A_2) & \cdots & 0\\ \vdots & \vdots & \ddots & \vdots\\ 0 & 0 & \cdots & W(A_K,A_K)\end{bmatrix}\\&=Y^TDY-\begin{bmatrix}W(A_1,A_1) & 0 & \cdots & 0\\ 0 & W(A_2,A_2) & \cdots & 0\\ \vdots & \vdots & \ddots & \vdots\\ 0 & 0 & \cdots & W(A_K,A_K)\end{bmatrix}\end{aligned}\tag{20}$$

我们关注的是 trace，所以只要对角线上的元素是一样的就可以了。令 $O' = Y^T DY - Y^T WY$，而 $\mathrm{tr}(O'P^{-1}) = \mathrm{tr}(OP^{-1})$，因为 O' 和 O 对角线上的元素都是一样的。所以，最终我们把 Ncut 目标函数化简成了矩阵的表达形式为：

$$\hat{Y} = \arg\min_{Y} \mathrm{tr}\left\{Y^T(D-W)Y \cdot (Y^T DY)^{-1}\right\}, \quad D = \mathrm{diag}(W \cdot 1_N) \tag{21}$$

其中 $(D-W)$ 被称为拉普拉斯矩阵 (Lapalacian Matrix)。最终，我们成功的用已知的 Y 和 W 来完成了对目标函数 Ncut 的矩阵化。有关拉普拉斯矩阵的性质，后面会讲到，这是图中一个很重要的矩阵，感兴趣的同学也可以看看 https://zhuanlan.zhihu.com/p/67336297，对拉普拉斯矩阵和拉普拉斯算子的详细介绍。

3.4 小结

这一小节中，我们主要的工作就是将目标函数 “Ncut 函数” 化简为矩阵形式。首先我们用指示向量来表示样本所属的类别。然后利用指示函数 (Y) 和相似度矩阵 (W) 来对 Ncut 函数进行表示。其实我觉得这里就可以看成是特征值分解后的结果。最终的变换结果和拉普拉斯矩阵有关系，有关拉普拉斯矩阵的详细内容有兴趣的同学可以自行查阅。

4 总结 (Conclusion)

这节主要讲述的是谱聚类算法，首先讲述了两种不同的聚类思路，一种就是聚合型聚类 (Compactness)，另一种是连通性聚类 (Connectivity) 算法。聚合性聚类更多的考虑是所有样本之间都是一视同仁的，根据特征的相似度来聚类。连通性聚类更多的考虑的是数据之间的分布结构，不同的数据之间可以有关系也可以没有关系，这样便于人们引入对数据的侧重点的分析，有点条件独立的意思在里面。而连通性聚类 (Connectivity) 算法需要借助图结构来实现。我们介绍了谱聚类算法的目标函数，然后对目标函数进行了优化，为了计算的方便，又描述了目标函数的矩阵表达形式。整个过程比较流程。

Feedforward Neural Network 01 Background

Chen Gong

10 November 2019

本节的主要目的是从一个较高的角度来介绍一下，什么是深度学习，并且给深度学习一个较好的总结，给大家一个较好的印象。机器学习是目前最火热的一个研究方向，而机器学习大致可以分为，频率派和贝叶斯派。频率派逐渐演变出了统计机器学习，而贝叶斯派逐渐演变出了 PGM，也就是概率图模型。下面我们分开进行描述。

1 频率派

统计机器学习方法基本就是由频率派的估计思想得到的。统计机器学习方法大概可以分成四种。

1. 正则化：L_1, L_2 也就是之前提到的 Lasso 和岭回归，这实际上并没有产生新的模型，而是在之前模型的基础上进行了改进。我们可以把它描述为 Loss function + regularized。用来抑制训练的过拟合。

2. 核化：最著名的就是我们之前提到的，Kernel Support Vector Machine (SVM) 了。

3. 集成化：也就是 Adaboost 和 Random Forest。

4. 层次化：层次化主要就是我们指的 Neural Network，也就是神经网络，神经网络进一步发展就得到了我们现在研究的深度学习。而神经网络中比较著名的几类就是：1. 多层感知机 (Multiple Layer Perception)；2. Autoencoder；3. CNN；4. RNN。这几个组合起来就是我们经常听到的 Deep Network。

2 贝叶斯派

贝叶斯派的估计方法就演化得到了概率图模型 (Probability Graphic Model。他们大致可以分成以下三类：

1. 有向图：Bayesian Network，深度增加之后就是 Deep Directed Network，包括大家听得很多的：Variational Automation Encode (VAE)，Generative Adversarial Network (GAN) 和 Sigmoid Belief Network 等等。

2. 无向图：Markov Network，深度增加之后就是 Deep Boltzmann Modeling，这就是我们的第二类图模型。

3. 有向图和无向图混合在一起，就是我们常说的 Mixed Network，主要包括，Deep Belief Network 等等。

而上述几个图模型，加上深度之后就是我们常说的 Deep Generative Network，深度生成模型。

在我们狭义的深度学习的理解中，什么是深度学习，实际上就是统计学习方法中的层次化中的 Deep Network。而广义的深度学习中，还应该包括，Deep Generative Network。而实际上绝大多数

的深度学习者都不太了解 Deep Generative Network，确实涉及到贝叶斯的理论，深度学习就会变得很难。而且它的训练也会变得非常的复杂。

Feedforward Neural Network 02 Development

Chen Gong

11 November 2019

本节主要是来讨论一下，机器学习的发展历史，看看如何从感知机到深度学习。

1 从时间的发展角度来看

1958 年：up，首次提出了 Perceptron Linear Algorithm (PLA)，这里就是我们机器学习的开端了。

1969 年：down，Marvin Lee Minsky 提出了，PLA has a limitation。因为 PLA 算法解决不了 non-linear 问题，比如说 XOR 问题。非常戏剧的是，这一年，Marvin Lee Minsky 获得了图灵奖，他也是“人工智能之父”，第一位因为 AI 而获得图灵奖的科学家。

1981 年：up，学者提出了 Multiple Layer Perceptron (MLP)，可以用来解决非线性的问题，就是是最初的 Feedforward Neural Network。

1986 年：up，Hinton 提出了将 Back Propagation (BP) 算法和 MLP 完美的融合在了一起，并且发展出了 Recurrent Neural Network (RNN) 算法。

1989 年：up，提出了 CNN。但是也迎来了人工智能的寒冬。down，在这一年中提出了一个 Universal Approximation theorem，也就是一个大于 1 层的 Hidden Layer 就可以用来拟合任何的连续函数。那么这是就提出了一个疑问：1 layer is OK，why deep? 并且，在 BP 算法中，随着深度的增加还会出现梯度消失的问题。

1993 年和 1995 年，down，这一年中 Support Vector Machine (SVM) + Kernel + Theory，获得了很好的效果。并且，Adaboost 和 Rondom Forest 等 Ensemble algorithm 流派的提出，获得了很好的效果。

1997 年，up，提出了 LSTM，但是远不足以止住深度学习发展的颓势。

2006 年，up，Hinton，提出了 Deep Belief Network (RBM) 和 Deep Auto-encoder。

2009 年，up，GPU 的飞速发展。

2011 年，up，Deep Learning 运用到了语音 (Speech) 中。

2012 年，up，斯坦福大学李飞飞教授，开办了一个非常重要的比赛和数据库 ImageNet。

2013 年，up，提出了 Variational Automation Encode (VAE) 算法。

2015 年，up，提出了非常重要的 Generative Adversarial Network (GAN)。

2016 年，up，围棋上 AlphaGo 彻底引爆了 Deep Learning。

2018 年，up，提出了重要的 Graphic Neural Network (GNN)，传统的神经网络是连接主义的，而 GNN 中将符合主义和连接主义进行了联合，使之具有推理的功能。

2 总结

其实 Deep Learning 的崛起是很多因素融合的结果。这些年来，主要都是在实践上的发展，而在机器学习理论上基本没有什么进步。它的发展得益于以下几点：1. data 的则增加，big data 时代的到来；2. 分布式计算的发展；3. 硬件水平的发展。其实最主要的说白了就是效果，效果比 SVM 要更好，就占据了主要的地位。

计算机学科就是一门实践为主的科学，现在在实际上取得了很好的效果。之后随着理论研究的不断深入，我们一定可以不断的完善理论知识。之后 AI 方向的研究，也将是以深度学习为主流，而其他机器学习学派的知识和优点将不断地丰富深度学习，扩充深度学习，来给它更强大的效果。

Feedforward Neural Network 03 Non-Linear Problem

Chen Gong

12 November 2019

实际上在 1958 年就已经成功的提出了 Perceptron Linear Analysis (PLA)，标志着人工智能的正式诞生。但是，Minsky 在 1969 年提出 PLA 无法解决非线性分类问题，让人工智能陷入了 10 年的低谷。后来的发展，人们开始寻找到越来越多的，解决非线性分类问题的方法。于是，我们提出了三种解决非线性问题的方法。

1 Non-Transformation

这实际上就是一种明转换，将向量从 input space 转换到 feature space，可以写做 $\phi = \mathcal{X} \longmapsto \mathcal{Z}$。在 Conver' s theory 中提出，高维空间比低维空间更加容易线性可分。很显然对于一个异或问题 (XOR) 来说，我们将 $x = (x_1, x_2) \xrightarrow{\phi} z = (x_1, x_2, (x_1 - x_2)^2)$

$$\begin{array}{cccc} 0 & 1 & \longrightarrow & 1 \\ 1 & 0 & \longrightarrow & 1 \\ 1 & 1 & \longrightarrow & 0 \\ 0 & 0 & \longrightarrow & 0 \end{array} \quad \xrightarrow{\phi} \quad \begin{array}{cccccc} 0 & 1 & 1 & \longrightarrow & 1 \\ 1 & 0 & 1 & \longrightarrow & 1 \\ 1 & 1 & 0 & \longrightarrow & 0 \\ 0 & 0 & 0 & \longrightarrow & 0 \end{array} \tag{1}$$

很显然在三维空间中，进行空间映射后，就会变得比较容易进行线性划分了。可以自己画图来进行验证，这里不再作图。

2 Kernel Method

这实际上是一种暗转的思路，也就是令 $K(x, x') =< \phi(x), \phi(x') >$，在这个核函数中实际上隐藏了一个 ϕ，而 $x, x' \in \mathcal{X}$。

3 Neural Network

神经网络算法实际上就是一个 Multit-Layer Perceptron (MLP)，有时也会被称为，Feedforward Neural Network (FNN)，所以大家在其他书上见到这几种描述都不要感到意外。我们以 XOR (位运算) 为例吧。在我们的逻辑运算中，大致有四种运算方法。

$$\begin{array}{cccc} XOR & OR & AND & NOT \\ \oplus & \vee & \wedge & \neg \end{array} \tag{2}$$

而后三种运算为基础运算，因为异或运算实际上是可以由后三种运算组成，也就是 $x_1 \oplus x_2 = (\neg x_1 \wedge x_2) \vee (x_1 \wedge \neg x_2)$。实际上就是先做两个与运算，然后做一个或运算。把一个线性不可分的东西来分层实现，将特征空间进行了分解而已。然后，在分层运算中插入了激活函数，来达到非线性映射的效果。这部分内容，比较的简单，而且网上也有大量的资料，此处就不再做过多的阐述。

实际上神经网络就是一个有向无环图。所以，某种意义上说可以引入概率图的模型，当然这就是后话了。