

机器学习概论 Introduction to Machine Learning

Lecture1. 机器学习概论

- 机器学习的发展历程：
- 1952年亚瑟塞缪尔开发了第一个跳棋程序，1959年提出机器学习概念，1957年提出感知器模型，1964年建立起统计学习理论，建立起SVM支持向量机，1969年闵斯基指出感知机模型的局限性，1984年瓦利恩特提出PAC学习理论框架，机器学习领域有了坚实的数学基础，1986年罗斯昆兰提出ID3决策树算法，1988年玻尔提出贝叶斯网络，1990年夏皮亚提出Boosting。
- 无监督学习包括聚类分析、降维、主题建模等，监督学习包括分类和回归。从概率视角来看，**无监督学习**的核心是给定若干样本 x_n 之后学习这些样本的概率分布 $p(x)$ ，而**监督学习**的核心是给定若干样本 (x_n, y_n) 之后，学习这些样本背后的条件概率分布 $p(y|x)$ 。
- 矩阵求偏导法则：
- $\frac{\partial x^T A x}{\partial x} = (A^T + A)x$, $\frac{\partial a^T x}{\partial x} = a$,
- 以线性回归为例， $MSE_{train} = \frac{1}{N} \|X^{train} w - y^{train}\|_2^2$ ，即线性回归的损失函数，因此只需要对这个 MSE_{train} 求对 w 的梯度即可。
- $$\begin{aligned} \nabla_{\tilde{w}} MSE_{train} &= \frac{\partial}{\partial \tilde{w}} MSE_{train} = \frac{1}{N} \frac{\partial}{\partial \tilde{w}} (X^{train} \tilde{w} - y^{train})^T (X^{train} \tilde{w} - y^{train}) \\ &= \frac{1}{N} \frac{\partial}{\partial \tilde{w}} ((X^{train} \tilde{w})^T X^{train} \tilde{w} - 2(y^{train})^T X^{train} \tilde{w} + (y^{train})^T y^{train}) \\ &\quad \frac{\partial x^T A x}{\partial x} = (A^T + A)x \quad \downarrow \quad \frac{\partial a^T x}{\partial x} = a \quad (X^{train} \tilde{w})^T y^{train} = (y^{train})^T X^{train} \tilde{w} \\ &= \frac{1}{N} (2(X^{train})^T X^{train} \tilde{w} - 2(X^{train})^T y^{train}) \\ \nabla_{\tilde{w}} MSE_{train} \Big|_{\tilde{w}^*} &= 0 \Rightarrow (X^{train})^T X^{train} \tilde{w}^* - (X^{train})^T y^{train} = 0 \\ \tilde{w}^* &= \left((X^{train})^T X^{train} \right)^{-1} (X^{train})^T y^{train} \end{aligned}$$
- 因此有 $\hat{w} = ((X^{train})^T X^{train})^{-1} (X^{train})^T y^{train}$ ，同时也有**拉格朗日插值法**，即给定 $n + 1$ 个点，可以找到穿过这 $n + 1$ 个点的唯一的 n 阶多项式函数。模型有容量之分，**容量**指的是一个模型能够模拟函数的能力，即模型的表示能力。验证集的目的是解决超参数的选择问题。模型的最终性能评估还是需要基于测试集，测试集从头到尾都没有参与模型的学习，是最有效的评估手段。同时也有**交叉验证**的方式，用**k-fold cross validation**的方式验证数据，将数据均分为K个不重不相交的子集，第 i 次训练的时候把 $d_{train} - d_i$ 作为训练集训练模型，把 d_i 作为验证集。选择验证集多次水平最高的超参数。
- 本课程中采用的向量表示都是**列向量**。

• $f: \mathbb{R}^n \rightarrow \mathbb{R}^m$

$$x = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} \quad f(x) = \begin{bmatrix} f_1(x) \\ \vdots \\ f_m(x) \end{bmatrix} \quad H_f(x) = \nabla^2 f \triangleq \begin{bmatrix} \frac{\partial^2 f}{\partial x_1 \partial x_1} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \cdots & \frac{\partial^2 f}{\partial x_n \partial x_n} \end{bmatrix} \quad J_f(x) \triangleq \frac{\partial f}{\partial x^T} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix}$$

- $f(x, y) = x^2 + 3xy + 2y^2$ ，因此 $\nabla f(x, y) = (2x + 3y, 3x + 4y)^T$ ，对每个值再分别对 x 和 y 求导就有：
 $H_f(x, y) = (2, 3, 3, 4)^T$ 。

$$H_f(x, y) = \begin{bmatrix} \frac{\partial}{\partial x}(2x + 3y) & \frac{\partial}{\partial y}(2x + 3y) \\ \frac{\partial}{\partial x}(3x + 4y) & \frac{\partial}{\partial y}(3x + 4y) \end{bmatrix}$$

- 常用的矩阵微分：

- $$\frac{\partial a^T x}{\partial x} = a, \quad \frac{\partial b^T A x}{\partial x} = A^T b, \quad \frac{\partial x^T A x}{\partial x} = (A + A^T)x$$

$$\frac{\partial}{\partial X} (a^T X b) = a b^T \qquad \frac{\partial}{\partial X} (a^T X^T b) = b a^T$$

$$\frac{\partial}{\partial X} \text{tr}(A X B) = A^T B^T \qquad \frac{\partial}{\partial X} \text{tr}(X^T A) = A$$

$$\frac{\partial}{\partial X} \text{tr}(X^{-1} A) = -X^{-T} A^T X^{-T} \qquad \frac{\partial}{\partial X} \text{tr}(X^T A X) = (A + A^T)X$$

- $$\frac{\partial}{\partial X} \det(A X B) = \det(A X B) X^{-T} \qquad \frac{\partial}{\partial X} \log(\det(X)) = X^{-T}$$

Lecture2. 概率基础

• 概率基础

- ✓独立和条件独立
- ✓贝叶斯公式及其应用
- ✓联合分布、边缘分布和条件概率分布
- ✓方差、期望
- ✓熵、交叉熵、KL散度

- $P(A^c) = 1 - P(A)$, $P(A \cap B) = P(A, B) = P(AB)$,
- $P(A, B) = P(A)P(B|A)$
- 全概率公式: $A_1, A_2 \dots$ 两两互不相容, 且 A_n 的并集是全集, 则

•
$$P(B) = \sum_{n=1}^N P(A_n, B) = \sum_{n=1}^N P(A_n)P(B|A_n)$$

- 乘法公式: $P(A_1, A_2, \dots, A_N) = P(A_1)P(A_2|A_1) \dots P(A_N|A_1, A_2 \dots A_{N-1})$
- 条件独立: 给定事件C, 称事件A和B在事件C下是条件独立的, 如果: $P(A, B|C) = P(A|C) * P(B|C)$, 条件独立与独立是完全不同的, 条件独立更弱一些, 但是独立不代表条件独立, 条件独立也不代表独立, 二者没有包含与被包含的关系。
- 累积分布函数 CDF: $P(x) = P(X \leq x)$
- 概率密度函数 PDF: $P(x) = P(X = x)$
- 二元离散随机变量联合分布: 边缘分布与联合分布

- X, Y 是两个连续随机变量
- 累积分布函数 (cumulative distribution function, cdf)

- $P(a, b) \triangleq P(X \leq a, Y \leq b) = \int_{-\infty}^a \int_{-\infty}^b p(x, y) dx dy$

- $p(x, y)$ 是联合概率密度函数

- 边缘分布 (marginal distribution)

- $P(x) \triangleq P(X \leq x) \triangleq \int_{-\infty}^x \int_{-\infty}^{\infty} p(u, y) dy du$

- $P(y) \triangleq P(Y \leq y) \triangleq \int_{-\infty}^y \int_{-\infty}^{\infty} p(x, v) dx dv$

- 边缘分布的概率密度函数

- $p(x) \triangleq \int_{-\infty}^{\infty} p(x, y) dy$

- $p(y) \triangleq \int_{-\infty}^{\infty} p(x, y) dx$

• 概率分布统计量: 均值和方差:

- 给定一个离散随机变量X, $E(x) = \sum_{x \in X} xp(x)$, 期望和求和可以交换, 如果 X_1, X_2, \dots, X_N 独立的话, $E(\prod_{n=1}^N X_n) = \prod_{n=1}^N E(X_n)$ 。
- 给定一个离散随机变量X, $\sigma^2 = E((X - E(X))^2) = E(X^2) - (E(X))^2$ 。
- $Cov(X_i, X_j) = E((X_i - E(X_i))(X_j - E(X_j)))$, $Cov(X_i, X_j) = E(X_i X_j) - E(X_i)E(X_j)$
- $\rho_{i,j} = \frac{Cov(X_i, X_j)}{\sqrt{V(x_i)V(x_j)}}$, 为相关系数。

- 协方差矩阵

$$Cov[X] \triangleq \mathbb{E}[(X - \mathbb{E}[X])(X - \mathbb{E}[X])^T]$$

$$= \begin{bmatrix} Cov[X_1, X_1] & Cov[X_1, X_2] & \dots & Cov[X_1, X_D] \\ Cov[X_2, X_1] & Cov[X_2, X_2] & \dots & Cov[X_2, X_D] \\ \vdots & \vdots & \ddots & \vdots \\ Cov[X_D, X_1] & Cov[X_D, X_2] & \dots & Cov[X_D, X_D] \end{bmatrix}$$

$$Cov[X_i, X_j] \triangleq \mathbb{E}[(X_i - \mathbb{E}[X_i])(X_j - \mathbb{E}[X_j])] = \mathbb{E}[X_i X_j] - \mathbb{E}[X_i]\mathbb{E}[X_j]$$

$$\mathbb{E}[X_i X_j] = \sum_{x_i, x_j} x_i x_j p(x_i, x_j) \quad \mathbb{E}[X_i X_j] = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} x_i x_j p(x_i, x_j) dx_i dx_j$$

- $\rho = 0$ 代表不相关, 但是不相关推不出独立, 独立可以推出不相关。辛普森悖论告诉我们, 相关并不代表因果。
- 全期望/方差法则: $E(X) = E(E(X|Y))$, $V(X) = E(V(X|Y)) + V(E(X|Y))$

- Bayes' rule

$$P(H = h|X = x) = \frac{P(X = x|H = h) \times P(H = h)}{P(X = x)}$$

- 简写

$$p(h|x) = \frac{p(x|h) \times p(h)}{p(x)}$$

后验分布
posterior distribution

- 边缘似然

✓ H 是离散随机变量 $p(x) = \sum_{h'} p(x|h') \times p(h')$

✓ H 是连续随机变量 $p(x) = \int p(x|h') \times p(h') dh'$

- 贝叶斯公式: 结果是后验分布, 即已知x之后求出h的可能性, 分子是似然和先验分布, 分母是边缘似然。 $p(h|x) = \frac{p(x|h)p(h)}{p(x)}$,
- 伯努利分布与二项分布:
- 伯努利分布, $p(x) = \theta^x (1 - \theta)^{1-x}$, 即 $X \sim Ber(\theta)$, $E(X) = \theta$, $V(X) = \theta(1 - \theta)$

- **二项分布**，其是N次重复独立的伯努利实验， $E(X) = N\theta$ ， $V(X) = N\theta(1 - \theta)$
- **泊松分布**：当实验次数趋于无穷时，二项分布变成泊松分布，当参数 λ 趋向于无穷时，泊松分布变成高斯分布。 $p(x) = e^{-\lambda} \frac{\lambda^x}{x!}$ ， $E(X) = V(x) = \lambda$ ，
- **类型分布**：参数 θ 代表的是C类的选择概率，是一个C维度的向量， $X \sim Cat(\theta)$ ，因此含义就是按照参数 θ 为概率随机选择C类中的一类。

泊松分布 (Poisson distribution)

$-x \in \{0, 1, 2, \dots\}$

$p(x) = P(X=x) = \text{Poi}(x; \lambda) \triangleq e^{-\lambda} \frac{\lambda^x}{x!}$

- 当试验次数 N 趋于无穷时，二项分布变成泊松分布

- 当参数 λ 趋于无穷时，泊松分布变成高斯分布

$E[X] = \sum_{x=0}^{\infty} x e^{-\lambda} \frac{\lambda^x}{x!} = \sum_{x=1}^{\infty} e^{-\lambda} \frac{\lambda^x}{(x-1)!} = \lambda e^{-\lambda} \sum_{x=1}^{\infty} \frac{\lambda^{x-1}}{(x-1)!} = \lambda e^{-\lambda} \sum_{x=0}^{\infty} \frac{\lambda^x}{x!} = \lambda e^{-\lambda} e^{\lambda} = \lambda$

$V[X] = E[X^2] - (E[X])^2 = \lambda$

高斯分布 (Gaussian distribution)

- 也称正态分布 (normal distribution)，是使用最为广泛的概率分布

- 概率密度函数 (其中，称 μ 为均值， σ^2 为方差)

$p(x) = \mathcal{N}(x; \mu, \sigma^2) \triangleq \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2\sigma^2}(x-\mu)^2} = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2\sigma^2}(x-\mu)^2\right)$

- 累积分布函数

$P(a) = \int_{-\infty}^a \mathcal{N}(x; \mu, \sigma^2) dx$

$P(a < x \leq b) = \int_a^b \mathcal{N}(x; \mu, \sigma^2) dx$

- 线性回归、高斯判别分析等方法的建模基础

$E[X] = \mu \quad V[X] = \sigma^2$

- **高斯分布**：也被称为正态分布， $p(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2\sigma^2}(x-\mu)^2}$ 当方差趋于0的时候，其分布在均值处无限高，形成类似狄拉克delta函数的样态，只在t=0的位置取正无穷，其余均取0。
- **拉普拉斯分布**： $p(x) = \frac{1}{2b} \exp(-\frac{1}{b}|x - \mu|)$ ， $E(X) = \mu$ ， $V(X) = 2b^2$ 是L1正则化方式的建模基础。还有Gamma分布和经验分布。其中经验分布就是基于已有的观测直接用概率估计之后的概率，比如说扔色子结果是{1, 2, 3, 3, 3, 4, 5, 6}，那么估计扔到2的概率就是1/8。

拉普拉斯分布 (Laplace distribution)

- 概率密度函数 ($b > 0$)

$p(x) = \text{Lap}(x; \mu, b) \triangleq \frac{1}{2b} \exp\left(-\frac{1}{b}|x - \mu|\right)$

- 累积分布函数

$P(a) = \int_{-\infty}^a \text{Lap}(x; \mu, b) dx$

- 拉普拉斯分布适合对有异常点的数据建模，是Lasso线性回归的建模基础

$E[X] = \mu \quad V[X] = 2b^2$

Gamma分布 (Gamma distribution)

- 定义在正值随机变量上的覆盖面较广的概率分布， $x \in \mathbb{R}^+$

- 概率密度函数 ($a > 0, b > 0$)

$p(x) = \text{Ga}(x; a, b) \triangleq \frac{b^a}{\Gamma(a)} x^{a-1} e^{-xb}$

$E[X] = \frac{a}{b} \quad V[X] = \frac{a}{b^2}$

- 特殊形式

- 指数分布 (exponential distribution)

$p(x) = \text{Ga}(x; 1, \lambda) = \lambda e^{-\lambda x}$

- 卡方分布 (Chi-squared distribution)

$p(x) = \text{Ga}\left(x; \frac{v}{2}, \frac{1}{2}\right)$

Gamma函数 (Gamma function)

$\Gamma(a) = \int_0^{\infty} x^{a-1} e^{-x} dx$

- **多元高斯正态分布**：

- 概率密度函数

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

- $\mathbf{x} \in \mathbb{R}^D$, 均值 $\boldsymbol{\mu} \in \mathbb{R}^D$, 协方差矩阵 $\boldsymbol{\Sigma}$ 是 $D \times D$ 对称矩阵

- 多元高斯分布的边缘分布：其中 x_1 和 x_2 都是随机向量，联合分布满足高斯分布，边缘分布的均值是联合分布的高斯分布的均值的一块，边缘分布的方差是协方差矩阵分块的左上角右下角结果。条件分布会更难计算一些。

- $\mathbf{x}_1 \in \mathbb{R}^D$ ， $\mathbf{x}_2 \in \mathbb{R}^K$ 是两个随机向量， \mathbf{x}_1 和 \mathbf{x}_2 的联合分布服从以下高斯分布

$$p(\mathbf{x}_1, \mathbf{x}_2) = \mathcal{N}\left(\begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{bmatrix}; \boldsymbol{\mu}, \boldsymbol{\Sigma}\right)$$

$$\boldsymbol{\mu} = \begin{bmatrix} \boldsymbol{\mu}_1 \\ \boldsymbol{\mu}_2 \end{bmatrix} \quad \boldsymbol{\Sigma} = \begin{bmatrix} \boldsymbol{\Sigma}_{11} & \boldsymbol{\Sigma}_{12} \\ \boldsymbol{\Sigma}_{21} & \boldsymbol{\Sigma}_{22} \end{bmatrix} \quad \boldsymbol{\Lambda} = \boldsymbol{\Sigma}^{-1} = \begin{bmatrix} \boldsymbol{\Lambda}_{11} & \boldsymbol{\Lambda}_{12} \\ \boldsymbol{\Lambda}_{21} & \boldsymbol{\Lambda}_{22} \end{bmatrix}$$

精度矩阵 (precision matrix)

- 边缘分布
$$p(\mathbf{x}_1) = \mathcal{N}(\mathbf{x}_1; \boldsymbol{\mu}_1, \boldsymbol{\Sigma}_{11})$$

$$p(\mathbf{x}_2) = \mathcal{N}(\mathbf{x}_2; \boldsymbol{\mu}_2, \boldsymbol{\Sigma}_{22})$$

$\boldsymbol{\Sigma}_{11}$ 是 $D \times D$ 矩阵
 $\boldsymbol{\Sigma}_{12}$ 是 $D \times K$ 矩阵
 $\boldsymbol{\Sigma}_{21}$ 是 $K \times D$ 矩阵
 $\boldsymbol{\Sigma}_{22}$ 是 $K \times K$ 矩阵

- 条件分布

$$p(\mathbf{x}_2|\mathbf{x}_1) = \mathcal{N}(\mathbf{x}_2; \boldsymbol{\mu}_{2|1}, \boldsymbol{\Sigma}_{2|1})$$

$$\boldsymbol{\mu}_{2|1} = \boldsymbol{\mu}_2 + \boldsymbol{\Sigma}_{21}\boldsymbol{\Sigma}_{11}^{-1}(\mathbf{x}_1 - \boldsymbol{\mu}_1)$$

$$= \boldsymbol{\mu}_2 - \boldsymbol{\Lambda}_{22}^{-1}\boldsymbol{\Lambda}_{21}(\mathbf{x}_1 - \boldsymbol{\mu}_1)$$

$$= \boldsymbol{\Sigma}_{2|1}(\boldsymbol{\Lambda}_{22}\boldsymbol{\mu}_2 - \boldsymbol{\Lambda}_{21}(\mathbf{x}_1 - \boldsymbol{\mu}_1))$$

$$\boldsymbol{\Sigma}_{2|1} = \boldsymbol{\Sigma}_{22} - \boldsymbol{\Sigma}_{21}\boldsymbol{\Sigma}_{11}^{-1}\boldsymbol{\Sigma}_{12} = \boldsymbol{\Lambda}_{22}^{-1}$$

Why?

$$p(\mathbf{x}_1|\mathbf{x}_2) = \mathcal{N}(\mathbf{x}_1; \boldsymbol{\mu}_{1|2}, \boldsymbol{\Sigma}_{1|2})$$

$$\boldsymbol{\mu}_{1|2} = \boldsymbol{\mu}_1 + \boldsymbol{\Sigma}_{12}\boldsymbol{\Sigma}_{22}^{-1}(\mathbf{x}_2 - \boldsymbol{\mu}_2)$$

$$= \boldsymbol{\mu}_1 - \boldsymbol{\Lambda}_{11}^{-1}\boldsymbol{\Lambda}_{12}(\mathbf{x}_2 - \boldsymbol{\mu}_2)$$

$$= \boldsymbol{\Sigma}_{1|2}(\boldsymbol{\Lambda}_{11}\boldsymbol{\mu}_1 - \boldsymbol{\Lambda}_{12}(\mathbf{x}_2 - \boldsymbol{\mu}_2))$$

$$\boldsymbol{\Sigma}_{1|2} = \boldsymbol{\Sigma}_{11} - \boldsymbol{\Sigma}_{12}\boldsymbol{\Sigma}_{22}^{-1}\boldsymbol{\Sigma}_{21} = \boldsymbol{\Lambda}_{11}^{-1}$$

- **舒尔补**：是一种计算矩阵逆的方式，按照下图的方式求解即可得知如何计算矩阵的逆

$$\mathbf{M} = \begin{bmatrix} \mathbf{E} & \mathbf{F} \\ \mathbf{G} & \mathbf{H} \end{bmatrix}$$

$$\mathbf{M}^{-1} = \begin{bmatrix} \mathbf{E}^{-1} + \mathbf{E}^{-1}\mathbf{F}(\mathbf{M}/\mathbf{E})^{-1}\mathbf{G}\mathbf{E}^{-1} & -\mathbf{E}^{-1}\mathbf{F}(\mathbf{M}/\mathbf{E})^{-1} \\ -(\mathbf{M}/\mathbf{E})^{-1}\mathbf{G}\mathbf{E}^{-1} & (\mathbf{M}/\mathbf{E})^{-1} \end{bmatrix} \quad \mathbf{M}/\mathbf{E} \triangleq \mathbf{H} - \mathbf{G}\mathbf{E}^{-1}\mathbf{F}$$

- 等效替换舒尔补就可以得出多元高斯分布的条件概率和分布。之后根据定义，求出 $p(x_1, x_2)$ ，得知其正比于 $\exp(-1/2 * F)$ ，之后将F拆分为 F_1 和 $F_2|1$ ，求出联合概率分布，之后先通过边缘分布求出p(x1)，积分后面的那部分显然是一个常数，因此有正比关系，一个分布正比于一个正态分布，则这个分布一定是正态分布。之后再带回，可以求出条件概率分布，进而p(x1)和p(x2|x1)都可以求出。因此联合分布p(x1, x2)可以求出。
- 最终需要记住的结论有：联合多元高斯分布的边缘分布还是高斯，条件分布还是高斯，条件协方差不大于原始协方差。 $\boldsymbol{\mu}_{2|1} = \boldsymbol{\mu}_2 + \sum_{21} \sum_{11}^{-1} (x_1 - \boldsymbol{\mu}_1)$ ， $\boldsymbol{\Sigma}_{2|1} = \boldsymbol{\Sigma}_{22} - \sum_{21} \sum_{11}^{-1} \sum_{12}$ ，

- 指数分布族：
- 通过参数 θ 刻画的一类概率分布形式， $p(x; \theta) = h(x) \exp(\theta^T \phi(x) - A(\theta))$ ，伯努利分布就可以写成这样的指数分布族的形式，即 $\log p(x) = x \log \mu + (1-x) \log(1-\mu) = x \log \frac{\mu}{1-\mu} + \log(1-\mu)$ ，因此 $p(x) = \exp(x \log \frac{\mu}{1-\mu} - (-\log(1-\mu)))$ ，其中 $\log \frac{\mu}{1-\mu}$ 是 θ ， $\phi(x) = x$ 。高斯分布也是指数族分布。

$$\begin{aligned}
 p(x) &= \mathcal{N}(x; \mu, \sigma^2) \triangleq \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2\sigma^2}(x-\mu)^2\right) \\
 &= \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2\sigma^2}x^2 + \frac{\mu}{\sigma^2}x - \frac{1}{2\sigma^2}\mu^2\right) \\
 &= \exp\left(-\frac{1}{2\sigma^2}x^2 + \frac{\mu}{\sigma^2}x - \left(\frac{\mu^2}{2\sigma^2} + \frac{\log(2\pi\sigma^2)}{2}\right)\right) \\
 \phi(x) &= \begin{bmatrix} \phi_1(x) \\ \phi_2(x) \end{bmatrix} \triangleq \begin{bmatrix} x \\ x^2 \end{bmatrix} \quad \theta = \begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix} \triangleq \begin{bmatrix} \frac{\mu}{\sigma^2} \\ -\frac{1}{2\sigma^2} \end{bmatrix} \quad \begin{matrix} \downarrow \\ A(\theta) = \frac{\mu^2}{2\sigma^2} + \frac{\log(2\pi\sigma^2)}{2} = -\frac{\theta_1^2}{4\theta_2} - \frac{1}{2}\log(-2\theta_2) + \frac{1}{2}\log 2\pi \end{matrix} \\
 &= \exp\left(\theta^T \phi(x) - A(\theta)\right)
 \end{aligned}$$

- 其实本质上就是写成指数分布族的形式之后对号入座，确定 ϕ 和 θ 所代表的含义，最后根据公式就能写出分布。

自然参数为一维的情况，即 $\theta \in \mathbb{R}$ 自然参数为多维的情况，即 $\theta \in \mathbb{R}^K (K > 1)$

$$\begin{aligned}
 A'(\theta) &= \frac{dA(\theta)}{d\theta} = \mathbb{E}[\phi(X)] & \nabla A(\theta) &= \frac{\partial A(\theta)}{\partial \theta} = \mathbb{E}[\phi(X)] \\
 A''(\theta) &= \frac{d^2 A(\theta)}{d\theta^2} = \mathbb{V}[\phi(X)] & \nabla^2 A(\theta) &= \frac{\partial}{\partial \theta^T} \frac{\partial A(\theta)}{\partial \theta} = \text{Cov}[\phi(X)]
 \end{aligned}$$

- 按照这种方式，我们就可以进行指数族分布的计算。代入即可求得各个值，最后计算出期望。
- 概率分布的度量：
- 熵： $H(x) = -\sum_{s=1}^S P(X = Xs) \log P(X = Xs)$ ， $H(Y|X) = H(X, Y) - H(X)$ 。
- 熵有一些性质，即 $0 \leq H(Y|X) \leq H(Y)$ ，如果知道X，我们对Y不确定性的认知不会增大。 $\max\{H(x), H(y)\} \leq H(x, y)$ 告诉我们不能通过仅增加变量来减少熵而使得问题更好解决。均匀分布的时候，熵取到最大值，这个结论可以通过拉格朗日乘子法来证明。

ITML

熵的主要性质



- $0 \leq \mathbb{H}(X) \leq \log|\mathcal{X}| = \log S$ ，当X服从均匀分布时，其熵取得最大值
- $0 \leq \mathbb{H}(Y|X) \leq \mathbb{H}(Y)$
 - 如果知道X，我们对Y不确定性的认知不会增大
- $\max\{\mathbb{H}(X), \mathbb{H}(Y)\} \leq \mathbb{H}(X, Y) \leq \mathbb{H}(X) + \mathbb{H}(Y)$
 - 不能仅通过增加变量来减少熵而使得问题更好处理
- 链式法则 (Chain Rule)： $\mathbb{H}(X, Y) = \mathbb{H}(X) + \mathbb{H}(Y|X) = \mathbb{H}(Y) + \mathbb{H}(X|Y)$
- 如果X和Y相互独立，则： $\mathbb{H}(Y|X) = \mathbb{H}(Y)$
 - 这意味着X不影响我们对Y不确定性的认知
- $\mathbb{H}(Y|Y)=0$
 - 如果事先知道Y，那么我们对Y的认知是确定的
- 证明的时候常用 $H(X, Y) = H(X) + H(Y|X)$ ，可以类比概率分布来记忆。
- 交叉熵： $H(p, q) = -\sum_x p(x) \log q(x)$ ，当p=q的时候交叉熵最小。衡量两个分布之间的差距。
- KL散度：衡量两个分布间差异度量的方法，

$$KL(p||q) = \sum_x p(x) \log \frac{p(x)}{q(x)} = -\sum_x p(x) \log \frac{q(x)}{p(x)} = E_{x \sim p(x)}[\log \frac{p(x)}{q(x)}]$$

- KL散度是大于等于0的， $\mathbf{KL(p||q) = H(p, q) - H(p)}$ ，这个公式可以用来最小化KL散度，即H(p)是固定的，我们只需要最小化交叉熵即可满足最小化KL散度。
- 如果我们需要使用一个简单的分布q(x)去近似分布p(x)，我们一般采用前向KL散度，即 $\min_q KL(p||q)$ ，这种KL散度的求法主要关注的是p(x)的信息损失。（被近似的目标放在前面）

Lecture3: 统计推断方法

- 统计推断方法
 - ✓最大似然估计
 - ✓最大后验估计
 - ✓贝叶斯方法
- 最大似然估计、最大后验概率、贝叶斯-共轭先验方法(Beta-Bernoulli, Dirichlet-Categorical, Gaussian-Gaussian)
- 以概率为模型框架的机器学习体系的中心思想：不知道分布函数 $p(x; \theta)$ ，通过数据反推来估计 θ 的过程。对参数 θ 的估计可以分为两大类方法，第一种方法是视 θ 为未知变量，学习的目标是求解 $\theta^* = \operatorname{argmax}_{\theta} p(D; \theta)$ ，这种方法是频率统计方法；第二种方法是视 θ 为未知随机变量，学习的目标是求解后验分布 $p(\theta|D)$ ，这种方法是贝叶斯统计方法。

• **最大似然估计(MLE):**

- 找到参数 θ ，使得数据集在这个参数下的似然最大。我们认为数据集中的采样是独立同分布的，因此使用N个数据的连乘代表最终的似然。由于乘法使得概率太低，我们考虑求对数似然，改为计算加法，同样也有负对数似然。这是对于无监督学习的分析。 $p(D; \theta) = \prod_{n=1}^N p(x_n; \theta)$,
 $LL(\theta) = \log p(D; \theta) = \sum_{n=1}^N \log p(x_n; \theta)$ ，一般求 $-LL(\theta)$ 的最小值为结果。
- 对于数据集D的似然函数有多种形式，包括 $p(D; \theta) = \prod_{n=1}^N p(y_n|x_n; \theta)$ 和 $p(D; \theta) = \prod_{n=1}^N p(x_n, y_n; \theta)$ 。
- 最大似然估计估计出的概率分布是尽可能接近经验分布的。**这个结论可以通过KL散度的推导得出， $q(x) = p(x; \theta)$ 具体来说， $KL(p_D||q) = \sum_x p_D(x) \log p_D(x) - \sum_x p_D(x) \log q(x)$ ，而
 $P_D(x) = \frac{1}{N} \sum_{n=1}^N \delta(x - x_n)$ ，即每个数据点和 x_n 完全相同的值的均值。代入之后有
 $KL(P_D||q) = -H(P_D) - \frac{1}{N} \sum_{n=1}^N \log q(x_n) = const + \frac{1}{N} NLL(\theta)$ ，因此最小化KL散度就等价于最小化 $NLL(\theta)$ 。同理，对于有监督标签的情况也是这样。

• 如果 $D = \{(x_n, y_n)\}_{n=1}^N$ ，则定义经验分布如下

$$\hat{p}_D(x, y) = \hat{p}_D(y|x) \hat{p}_D(x) \triangleq \frac{1}{N} \sum_{n=1}^N \delta(x - x_n) \delta(y - y_n)$$

$$\begin{aligned} \mathbb{E}_{\hat{p}_D(x)}[KL(\hat{p}_D(y|x) || q(y|x))] &\triangleq \sum_x \hat{p}_D(x) \left(\sum_y \hat{p}_D(y|x) \log \frac{\hat{p}_D(y|x)}{q(y|x)} \right) \\ q(y|x) &\triangleq p(y|x; \theta) \\ &= \sum_{x, y} \hat{p}_D(x, y) \log \hat{p}_D(y|x) - \sum_{x, y} \hat{p}_D(x, y) \log q(y|x) \\ &= const - \frac{1}{N} \sum_{n=1}^N \log p(y_n|x_n; \theta) = const + \frac{1}{N} NLL(\theta) \end{aligned}$$

- 在抛硬币实验中，令抛硬币正面朝上的概率为 θ ，正面朝下的概率为 $1 - \theta$ 。显然 θ 对我们而言是未知的，我们希望通过实验中观测到的结果来估计 θ 的值，进而判断该硬币是否均匀。我们假设其为伯努利分布。

• 假设我们抛了十次硬币，结果如下

- {1, 1, 0, 0, 0, 1, 0, 1, 1, 1}

- 即 $D = \{x_1 = 1, x_2 = 1, x_3 = 0, x_4 = 0, x_5 = 0, x_6 = 1, x_7 = 0, x_8 = 1, x_9 = 1, x_{10} = 1\}$

$$\begin{aligned} LL(\theta) = \log p(D; \theta) &= \sum_{n=1}^{10} \log p(x_n; \theta) \\ &= \sum_{x_n=1} \log p(x_n; \theta) + \sum_{x_n=0} \log p(x_n; \theta) = 6 \log \theta + 4 \log(1 - \theta) \end{aligned}$$

极值在目标函数导数或梯度上的值为零

$$\frac{d}{d\theta} LL(\theta) = \frac{6}{\theta} - \frac{4}{1-\theta}$$

注意：也可利用
 $\theta = \operatorname{argmin}_{\theta} NLL(\theta)$
进行求解，方法基本一样

$$\theta = \operatorname{argmax}_{\theta} LL(\theta) \Rightarrow \frac{d}{d\theta} LL(\theta) = 0 \Rightarrow \frac{6}{\theta} - \frac{4}{1-\theta} = 0 \Rightarrow \theta = \frac{3}{5}$$

- 推广到一般情形， $\theta = \frac{N_1}{N}$ ，看似是概率分布估计，但是本质上是最大似然估计。同理示例2中，掷骰子判断是否均匀的时候，采用的是类别分布， $p(x) = \prod_{k=1}^K \theta_k^{I(x=k)}$ ，
 $NLL(\theta) = -\sum_{n=1}^N \log \prod_{k=1}^K \theta_k^{I(x_n=k)} = \sum_{k=1}^K N_k \log \theta_k$ ，约束 $\sum_{k=1}^K \theta_k = 1$ 下使用拉格朗日乘子， $\theta_i = \frac{N_i}{N}$ 。
对于示例3，即高斯分布中， $NLL(\theta) = \frac{N}{2} \log 2\pi + \frac{N}{2} \log \sigma^2 + \frac{1}{2\sigma^2} \sum_{n=1}^N (x_n - \mu)^2$ ，分别对 μ 和 σ 求偏导，有：

$$\theta_{MLE} = \operatorname{argmin}_{\theta} NLL(\theta) \rightarrow \text{求解} \begin{cases} \frac{\partial}{\partial \mu} NLL(\theta) = 0 \\ \frac{\partial}{\partial \sigma^2} NLL(\theta) = 0 \end{cases}$$
$$NLL(\theta) = \frac{N}{2} \log 2\pi + \frac{N}{2} \log \sigma^2 + \frac{1}{2\sigma^2} \sum_{n=1}^N (x_n - \mu)^2$$
$$\frac{\partial}{\partial \mu} NLL(\theta) = 0 \rightarrow \mu = \frac{1}{N} \sum_{n=1}^N x_n$$

充分统计量

$$\frac{\partial}{\partial \sigma^2} NLL(\theta) = 0 \rightarrow \sigma^2 = \frac{1}{N} \sum_{n=1}^N (x_n - \mu)^2 = \frac{1}{N} \sum_{n=1}^N x_n^2 - \left(\frac{1}{N} \sum_{n=1}^N x_n \right)^2$$

- 最大似然估计的问题：**其优化目标是使得看到的数据出现的概率最大，这会导致对未看到的数据难以给出合适的概率估计，造成**过拟合**的问题。比如说扔硬币的时候我们抛了五次，结果都是正面朝上，根据最大似然估计的理论，此时向上的概率应该是1，但是这个与实际不同。为了解决这个问题，我们引入了最大后验估计，用先验知识来解决。

• **最大后验估计(MAP):**

- 把待估计的参数 θ 视为随机变量，且**服从某个先验分布**，即数据集D是先验分布，在这个先验分布的情况下估计参数 θ ，这样我们可以使用贝叶斯公式计算后验分布。

$$p(\theta|D) = \frac{p(D|\theta)p(\theta)}{p(D)}$$

根据 $p(\theta|D)$ 对参数 θ 进行采样，最有可能选择的样本点是 $p(\theta|D)$ 的众数，用这个众数来作为采样出数据集D

的概率分布的参数。

$$\theta_{MAP} = \arg \max_{\theta} p(\theta \mid D) = \arg \max_{\theta} \frac{p(D \mid \theta)p(\theta)}{p(D)} = \arg \max_{\theta} p(D \mid \theta)p(\theta)$$

ITML

最大后验估计



$$\theta_{MAP} = \operatorname{argmax}_{\theta} p(\theta|D) = \operatorname{argmax}_{\theta} \frac{p(D|\theta)p(\theta)}{p(D)} = \operatorname{argmax}_{\theta} p(D|\theta)p(\theta)$$

$$\max_{\theta} p(D|\theta)p(\theta) \Leftrightarrow \max_{\theta} (\log p(D|\theta) + \log p(\theta))$$

正则项，后续课程有详述

$$\theta_{MAP} = \operatorname{argmax}_{\theta} (\log p(D|\theta) + \log p(\theta))$$

注意：给定 θ ， $p(D|\theta)$ 就是 $p(D; \theta)$

$$\theta_{MAP} = \operatorname{argmax}_{\theta} (LL(\theta) + \log p(\theta)) = \operatorname{argmax}_{\theta} \log(p(D; \theta) + \log p(\theta))$$

当 θ 的先验分布 $p(\theta)$ 是均匀分布时， $\theta_{MAP} = \theta_{MLE}$

由此，

$$\theta_{MAP} = \operatorname{argmax}_{\theta} \log(p(D; \theta) + \log p(\theta))$$

- 请注意，给定 θ 的情况下， $p(D|\theta) = p(D; \theta)$ 。
- 对MAP进行拆分之后得出的结论是上左图所示，具体来说， θ_{MAP} 的结果里包含了 $LL(\theta)$ 和 $\log p(\theta)$ 这两项，即**如果 θ 的先验分布是均匀分布时，MAP和MLE是一样的。**
- 示例1：在扔硬币的问题之中，我们假设扔硬币有先验分布 $Beta(a, b) = \frac{1}{B(a, b)} \theta^{a-1} (1 - \theta)^{b-1}$ ，其中计算MAP的前半部分跟MLE是一样的，后半部分是不一样的，后半部分多了对 $p(\theta)$ 的计算，再求导得出结果。我们可以认为**MLE是MAP的一种特例，即考虑均匀分布当作先验的MAP**。这其中 $p(D|\theta) = p(D; \theta)$ ，再拆分为每个D中的例子的相乘，相乘之后拆分为相加。

ITML

最大后验估计：示例1



$$D = \{1, 1, 0, 0, 0, 1, 0, 1, 1, 1\}$$

$$L_{MAP}(\theta) \triangleq \log p(D|\theta) + \log p(\theta)$$

$$= \sum_{x_n=1} \log p(x_n; \theta) + \sum_{x_n=0} \log p(x_n; \theta) - \log B(a, b) + (a - 1) \log \theta + (b - 1) \log(1 - \theta)$$

$$= (a + 5) \log \theta + (b + 3) \log(1 - \theta) - \log B(a, b)$$

$$\frac{d}{d\theta} L_{MAP}(\theta) = \frac{a + 5}{\theta} - \frac{b + 3}{1 - \theta}$$

$$\theta_{MAP} = \operatorname{argmax}_{\theta} L_{MAP}(\theta) \Rightarrow \frac{d}{d\theta} L_{MAP}(\theta) = 0 \Rightarrow \frac{a + 5}{\theta} - \frac{b + 3}{1 - \theta} = 0 \Rightarrow \theta = \frac{a + 5}{a + b + 8}$$

- 对于一般情况，如果认为先验是Beta(2, 2)分布，后验是伯努利分布的时候，最终的估计结果是**加一平滑**，即 $\theta = \frac{N_1+1}{N+2}$ ，如果未知beta分布参数的时候是 $\theta = \frac{N_1+a-1}{N+a+b-2}$ 。
- 同理，示例2展示的是假设先验分布是已知参数的高斯分布，则同样可以估计出 μ 的取值是

$\mu = \frac{\frac{1}{\sigma^2} \sum_{n=1}^N x_n + \frac{1}{\sigma_1^2} \mu_1}{\frac{N}{\sigma^2} + \frac{1}{\sigma_1^2}}$ ，总结基本思路就是根据公式写出MAP的求解式，之后对待求参数求偏导令其值为0。

贝叶斯方法（Bayesian）：

- 最大似然估计和最大后验分布都认为**待估计的参数是一个常数，即固定值**，而在贝叶斯方法中， θ 被视作随机变量，看到数据之前有一个先验信念 $p(\theta)$ ，看到数据之后更新得到后验分布 $p(\theta|D)$ 。因此贝叶斯方法的核心就是**估计 $p(\theta|D)$** 。MAP的思路是求这个后验分布的众数，而贝叶斯方法则是希望能够将这个分布求出。
- 在估计获得参数的分布之后，我们需要估计**当一个新样本x输入之后，其函数的分布情况**，为了估算出这个值，我们不能直接取出一个参数，而是希望对不同的参数进行加权，因此对于无标签的无监督问题， $p(x|D) = E_{\theta \sim p(\theta|D)}(p(x|\theta))$ ， $p(y|x, D) = E_{\theta \sim p(\theta|D)}(p(y|x, \theta))$ 。

ITML

贝叶斯方法



- 求出后验分布 $p(\theta|D)$ 后，对给定的输入数据 x ，我们可以如下计算其后验预测分布
- 无监督场景： $D = \{x_n\}_{n=1}^N$

$$p(x|D) = \int p(x, \theta|D) d\theta$$
$$= \int p(x|\theta, D)p(\theta|D) d\theta$$
$$= \int p(x|\theta)p(\theta|D) d\theta$$
$$= \mathbb{E}_{\theta \sim p(\theta|D)}[p(x|\theta)]$$

$$p(x|D) = \sum_{\theta} p(x, \theta|D)$$
$$= \sum_{\theta} p(x|\theta, D)p(\theta|D)$$
$$= \sum_{\theta} p(x|\theta)p(\theta|D)$$
$$= \mathbb{E}_{\theta \sim p(\theta|D)}[p(x|\theta)]$$

- 通过以上形式来进行预测的机器学习方法也称为贝叶斯机器学习 (Bayesian machine learning)
- 上述方法实际上是用模型预测结果的期望（加权平均）来进行最终预测，而不是仅用一个“最好”的参数模型（如MLE或MAP）来进行预测。因此，贝叶斯机器学习方法能有效规避过学习问题

- 对于监督场景下，即 $D = \{(x_n, y_n)\}_{n=1}^N$ ，在求出后验分布 $p(\theta|D)$ 后，对给定的输入数据 x ，我们可以如下计算其标签后验预测分布

$$p(y|x, D) = \int p(y, \theta|x, D) d\theta$$
$$= \int p(y|x, \theta, D)p(\theta|x, D) d\theta$$
$$= \int p(y|x, \theta)p(\theta|D) d\theta$$
$$= \mathbb{E}_{\theta \sim p(\theta|D)}[p(y|x, \theta)]$$

$$p(y|x, D) = \sum_{\theta} p(y, \theta|x, D)$$
$$= \sum_{\theta} p(y|x, \theta, D)p(\theta|x, D)$$
$$= \sum_{\theta} p(y|x, \theta)p(\theta|D)$$
$$= \mathbb{E}_{\theta \sim p(\theta|D)}[p(y|x, \theta)]$$

- 其基本方式是基于MAP进行的，在通过MAP求出后验分布之后，我们对后验分布化为积分形式，不断化简，第三步的化简是因为给出 θ 的分布之后 x 的值就与D无关。最后使用一个期望来描述。

- 贝叶斯方法在我们已知似然函数 $p(D|\theta)$ 和先验分布 $p(\theta)$ 之后，我们如果想要计算出后验概率 $p(\theta|D)$ 还需要 $p(D)$ ，但是这个数很难求出，因此需要使用**共轭先验方法**、**变分近似**和**MCMC近似**的方法来计算。

贝叶斯-共轭先验方法：

- 如果先验分布 $p(\theta)$ 和后验分布 $p(\theta|D)$ 都属于同一个分布族，那么我们认为先验分布和似然函数是共轭的，这种情况存在闭式解，因此我们称之为贝叶斯-共轭先验方法。如果都是指数分布族，则后验分布具有闭式解。Beta-Bernoulli模型、Dirichlet-categorical模型和Gaussian-Gaussian模型是最常用的

1. Beta-Bernoulli模型：

- 先验分布和后验分布都是**Beta函数的分布族**。如果我们认为抛硬币问题中单次抛硬币的结果是 $x_n \sim Ber(\theta)$ ，则 $p(D|\theta) = \theta^{x_n=1}(1-\theta)^{x_n=0}$ ，满足这样状态的分布只有Beta分布。因此 θ 的先验分布应该具有 $p(\theta) \sim \theta^\alpha(1-\theta)^\beta$ ，即是Beta分布。
- 正比于一个分布，则代表这两个分布是一个分布**。根据这个理论，在计算后验分布的时候可以省略考虑分母， $p(\theta|D) \propto p(D|\theta)p(\theta)$ ，其中我们认为 θ 的分布是 $Beta(\alpha, \beta)$ ，化简之后得出的结果是：

$$p(\theta|D) = Beta(\theta; N1 + \alpha, N0 + \beta)$$

- 这其中 $N_1 = (x_n = 1), N_0 = (x_n = 0)$ ， $p(\theta) \sim Beta(\alpha, \beta)$ 。为先验分布。
- 之后我们希望求出后验分布的**众数**，展开 $p(\theta|D)$ 的分布之后，求这个分布中 $argmax_{\theta}$ ，通过求偏导得到的结论是 $\theta^{N_1+\alpha-1}(1-\theta)^{N_0+\beta-1}(\frac{N_1+\alpha-1}{\theta} - \frac{N_0+\beta-1}{1-\theta})$ ，因此最重令括号里的式子结果为0，结果是 $\theta_{mode} = \frac{N_1+\alpha-1}{N+\alpha+\beta-2}$ 。
- 后验分布统计量的均值**也可以计算，具体计算方法是先令 $p(\theta) = Beta(\theta; \alpha, \beta)$ ，直接通过积分的方式求解 $E(\theta) = \int \frac{1}{B(\alpha, \beta)} \theta^{\alpha-1}(1-\theta)^{\beta-1} \theta d\theta$ ，之后展开 $Beta(\theta; \alpha+1, \beta)$ ，通过其全部积分为1的公式可以推出 $B(\alpha+1, \beta) = \int \theta^\alpha(1-\theta)^{\beta-1} d\theta$ ，之后计算出 $E(\theta)$ ，之后得出 $E(\theta) = \frac{\alpha}{\alpha+\beta}$ ，同理代入真正估计的分布 $p(\theta|D) = Beta(\theta; N1 + \alpha, N0 + \beta)$ 有

$$E(\theta|D) = \frac{N1 + \alpha}{N + \alpha + \beta}$$

，这个值可以理解成 θ_{MLE} 和 θ_{prior} 的线性组合。 θ_{MLE} 的值是 $\frac{N_1}{N}$ ， θ_{prior} 的值是 $\frac{\alpha}{\alpha+\beta}$

- 后验分布的方差**，其衡量的是给定数据之后参数的波动值。 $V(\theta|D) = E(\theta^2|D) - E(\theta|D)^2$ ，后面的值是后验分布的均值，其已经得出结论，需要前面的值，用积分展开期望的平方项之后有：
 $E(\theta^2|D) = \frac{B(N_1+\alpha+2, N_0+\beta)}{B(N_1+\alpha, N_0+\beta)}$ ，继续化简的结果是 $\frac{(N_1+\alpha+1)(N_1+\alpha)}{(N+\alpha+\beta+1)(N+\alpha+\beta)}$ ，因此 $V(\theta|D) = \frac{\beta}{\alpha(\alpha+\beta+1)}(E(\theta|D)^2)$ ，如果 $N \gg \alpha + \beta$ ，则 $E(\theta|D) = \frac{N_1}{N}$ ， $\frac{\beta}{\alpha(\alpha+\beta+1)} = \frac{N_0}{N_1 N}$ ，这是因为 $\hat{\alpha} = N_1 + \alpha, \hat{\beta} = N_0 + \beta$ ，最终的方差结果是：

$$V(\theta|D) = \frac{N_0 N_1}{N^3} = \frac{\theta_{MLE} * (1 - \theta_{MLE})}{N}$$

- 贝叶斯公式里的分母同样有意义， $p(D) = \int p(D|\theta)p(\theta)d\theta$ 。具体来说，我们称 $p(D)$ 为**边缘似然**或者证据，这个值很难算，但是也是可以计算的。对于Beta-Bernoulli分布，使用贝叶斯公式展开之后：

$$P(D) = \frac{B(N1 + \alpha, N0 + \beta)}{B(\alpha, \beta)}$$

$$\begin{aligned} p(x = 1|D) &= \int p(x = 1, \theta|D) d\theta \\ &= \int p(x = 1|\theta, D)p(\theta|D) d\theta \\ &= \int p(x = 1|\theta)p(\theta|D) d\theta \\ &= \int \theta Beta(\theta; N_1 + \alpha, N_0 + \beta) d\theta \\ &= E[\theta|D] = \frac{N_1 + \alpha}{N + \alpha + \beta} = \frac{\hat{\alpha}}{\hat{\alpha} + \hat{\beta}} \end{aligned}$$

如果 $\theta \sim Beta(1,1)$ ，即参数先验服从均匀分布，则 $p(x = 1|D) = \frac{N_1+1}{N+2}$ 。这就是之前最大后验估计时提到的加一平滑

注意：在最大后验估计时，我们是通过假定 $\theta \sim Beta(2,2)$ 来获得加一平滑。这有点不自然。而贝叶斯方法，通过利用参数先验服从均匀分布达到相同的效果

- 这里显示的是如果参数先验服从均匀分布，则 $p(x = 1|D) = \frac{N_1+1}{N+2}$ ，这就是加一平滑。

2. Dirichlet-categorical模型：

- 对于类型分布， $p(D|\theta) = \prod_{k=1}^K \theta_k^{N_k}$ ，迪利克雷分布与这个形式相同，即 $p(\theta) = Dir(\theta; \alpha) = \frac{1}{B(\alpha)} \prod_{k=1}^K \theta_k^{\alpha_k-1}$ ，迪利克雷分布是Beta分布的多元变量扩展



狄利克雷 (Dirichlet) 分布

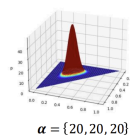
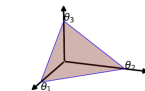
- 是Beta分布的多元变量扩展，也称为关于概率分布的分布
- 定义域为 $\{\theta | 0 \leq \theta_k \leq 1, \sum_{k=1}^K \theta_k = 1\}$ ，即其输入变量是概率分布

$$p(\theta) = \text{Dir}(\theta; \alpha) = \frac{1}{B(\alpha)} \prod_{k=1}^K \theta_k^{\alpha_k - 1}$$

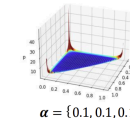
- 基本性质

$$\text{均值: } \mathbb{E}[\theta] \triangleq [\mathbb{E}[\theta_1], \dots, \mathbb{E}[\theta_K]]^T, \mathbb{E}[\theta_k] = \frac{\alpha_k}{\sum_{i=1}^K \alpha_i}$$

$$\text{众数: } M[\theta] \triangleq [M[\theta_1], \dots, M[\theta_K]]^T, M[\theta_k] = \frac{\alpha_k - 1}{\sum_{i=1}^K \alpha_i - K}$$



$\alpha = \{20, 20, 20\}$



$\alpha = \{0.1, 0.1, 0.1\}$

- 利用同样的方法可以求出: $p(\theta|D) \propto \prod_{k=1}^K \theta_k^{N_k} \prod_{k=1}^K \theta_k^{\alpha_k - 1} = \frac{1}{B(\alpha)} \prod_{k=1}^K \theta_k^{N_k + \alpha_k - 1}$,

$$E(\theta_k|D) = \frac{N_k + \alpha_k}{\sum_{i=1}^K (N_i + \alpha_i)}$$

$$M(\theta_k|D) = \frac{N_k + \alpha_k - 1}{\sum_{i=1}^K (N_i + \alpha_i) - K}$$

- 当 $\alpha_1 = \alpha_2 = \dots = \alpha_k = 1$ 的时候, $M(\theta_k|D) = \frac{N_k}{\sum_{i=1}^K N_i}$, 也就是最大似然估计。

- $p(x = k|D) = \frac{N_k + \alpha_k}{\sum_{i=1}^K N_i + \sum_{i=1}^K \alpha_i}$, 这是后验预测。

$$p(D) = \frac{B(\alpha_1 + N_1, \dots, \alpha_K + N_K)}{B(\alpha_1, \dots, \alpha_K)}$$

- 以上三个值分别是后验均值、最大后验估计、边缘似然。

3. Gaussian-Gaussian模型:

- 讨论一元高斯分布, σ 是已知常量, μ 是参数变量。 $p(D|\mu, \sigma^2) \propto \exp(-\frac{1}{2\sigma^2} \sum_{n=1}^N (x_n - \mu)^2)$, 为了保证与以上的似然函数共轭, μ 的先验分布需要具有以上的形式, 高斯分布满足要求。 $x \sim N(\mu, \sigma^2)$, $\mu \sim N(\mu_1, \sigma_1^2)$, 经过冗长推导有: $p(\mu|D) = N(\mu; \beta^2 \alpha, \beta^2)$, 其中 $\beta^2 = (\frac{N}{\sigma^2} + \frac{1}{\sigma_1^2})^{-1}$ 。

- 当仅有一个观测数据的时候, 同理可以求出 $E(\mu|D) = \frac{N\sigma_1^2}{N\sigma_1^2 + \sigma^2} x + \frac{\sigma^2}{N\sigma_1^2 + \sigma^2} \mu_1$, 由于后验分布也是高斯分布, 所以众数也是均值, $M = E(\mu|D)$, 方差 $V(\mu|D) = \frac{\sigma_1^2 \sigma^2}{N\sigma_1^2 + \sigma^2}$,

$$p(D) = \sqrt{2\pi\beta^2} \times \text{Const} = \left(\frac{1}{\sqrt{2\pi\sigma^2}} \right)^N \frac{\sqrt{2\pi\beta^2}}{\sqrt{2\pi\sigma_1^2}} \exp\left(\frac{\beta^2 \alpha^2}{2} - \frac{1}{2\sigma^2} \left(N^2 \bar{x}^2 - \sum_{i \neq j} x_i x_j \right) - \frac{\mu_1^2}{2\sigma_1^2} \right)$$

Lecture4. 朴素贝叶斯

- 生成模型不是直接学 $p(y|x)$, 而是通过学 $p(y)$ 和 $p(x|y)$ 的方式利用贝叶斯计算后验。
- 首先需要明确生成模型的定义, 即最终的结果是给定样本 x 之后能够计算出所属不同类别的概率, 为了计算整个值需要使用贝叶斯公式, $p(y = c; \theta)$ 代表的是先验, 即不看任何特征之前类别 c 的常见度; $p(x|y = c; \theta)$ 代表的是**生成模型**的特点, 即给出一个类别 c 之后得出特征 x 的大致样态。这个公式就是生成的核心。
- 朴素贝叶斯假定, 给定一个标签之后, **数据的特征互相独立**。 $p(x|y = c; \theta) = \prod_{d=1}^D p(x_d|y = c; \theta)$

生成模型

$$p(y = c|x; \theta) = \frac{p(y = c; \theta) p(x|y = c; \theta)}{p(x; \theta)} = \frac{p(y = c; \theta) p(x|y = c; \theta)}{\sum_{i=1}^C p(y = i; \theta) p(x|y = i; \theta)}$$

- $p(y = c; \theta)$: 类标签先验分布

- $p(x|y = c; \theta)$: 数据 x 的类条件分布; 通过采样 $p(x|y = c; \theta)$, 能够生成类别 c 的数据 x

- θ : 模型参数

- 朴素贝叶斯**是一种模型的假设方式, 其朴素的核心假设是**类条件独立**, 认为**数据的特征之间是彼此相互独立的**。通过这种假设, 极大减少了需要学的参数量。与高斯判别分析来对比可知, 其是高斯判别分析更强的一种假设。

$$p(y = c | x; \theta) = \frac{p(y = c; \theta) \prod_{d=1}^D p(x_d | y = c; \theta)}{\sum_{i=1}^C p(y = i; \theta) \prod_{d=1}^D p(x_d | y = i; \theta)}$$

- 因此需要学的参数是 $p(y = c; \theta)$ 和 $p(x_d|y = c; \theta)$ ，对于 $p(x|y = c; \theta)$ 来说，也即生成的过程，主要的建模方式就是伯努利分布建模、类型分布建模、高斯分布建模。类条件独立的意义很大，在这种独立条件下，给定类标签 c 之后，类条件的概率数量是 $\sum_{i=1}^D K_i$ ，而无类条件独立的情况下，概率数量是 $\prod_{i=1}^D K_i$ ，因为每一维度的取值都对应着其余所有维度的取值。

- $x_d \in \{0,1\}$ ，特征类条件分布可用伯努利分布建模
 - $-p(x|y = c; \theta) = \prod_{d=1}^D \beta_{dc}^{x_d} (1 - \beta_{dc})^{1-x_d} = \prod_{d=1}^D \text{Ber}(x_d; \beta_{dc})$
 - $x_d \in \{1, 2, \dots, K_d\}$ ，特征类条件分布可用类型分布建模
 - $-p(x|y = c; \theta) = \prod_{d=1}^D \prod_{k=1}^{K_d} \beta_{kdc}^{\mathbb{I}(x_d=k)} = \prod_{d=1}^D \text{Cat}(x_d; \{\beta_{kdc}\})$
 - $x_d \in \mathbb{R}$ ，特征类条件分布可用高斯分布建模
 - $-p(x|y = c; \theta) = \prod_{d=1}^D \mathcal{N}(x_d; \mu_{dc}, \sigma_{dc}^2)$

模型参数

朴素贝叶斯-伯努利建模：

- 伯努利建模的含义是生成过程，即 $p(x|y = c; \theta)$ 是伯努利分布，再利用朴素贝叶斯假设化简。
- $x_d \in \{0, 1\}$ ，而且模型的参数 $\theta = \alpha_c, \beta_{dc}$ ， $\alpha_c = p(y = c), \beta_{dc} = p(x_d = 1|y = c)$ ，一个参数是先验，一个参数是似然，也就是生成过程。目的是求解最优的参数，使得 $p(D; \theta)$ 最大。

- $$p(x|y = c; \theta) = \prod_{d=1}^D \beta_{dc}^{x_d} (1 - \beta_{dc})^{1-x_d} = \prod_{d=1}^D \text{Ber}(x_d; \beta_{dc})$$
- $$p(D; \theta) = \prod_{n=1}^N p(y_n; \theta) \prod_{n=1}^N p(x_n|y_n; \theta)$$
，化简出 $y = c$ 的成分，之后再次化简，得出 $LL(\theta)$ 的值

$$p(D; \theta) = \left(\prod_{n=1}^N \prod_{c=1}^C \alpha_c^{\mathbb{I}(y_n=c)} \right) \left(\prod_{n=1}^N \prod_{c=1}^C \left(\prod_{d=1}^D \beta_{dc}^{x_{nd}} (1 - \beta_{dc})^{1-x_{nd}} \right)^{\mathbb{I}(y_n=c)} \right)$$



$$LL(\theta) = \log p(D; \theta)$$

$$= \sum_{n=1}^N \sum_{c=1}^C \mathbb{I}(y_n = c) \log \alpha_c + \sum_{n=1}^N \sum_{c=1}^C \mathbb{I}(y_n = c) \left(\sum_{d=1}^D (x_{nd} \log \beta_{dc} + (1 - x_{nd}) \log(1 - \beta_{dc})) \right)$$

$$= \sum_{c=1}^C \sum_{n=1}^N \mathbb{I}(y_n = c) \log \alpha_c + \sum_{c=1}^C \sum_{d=1}^D \sum_{n=1}^N \mathbb{I}(y_n = c) (x_{nd} \log \beta_{dc} + (1 - x_{nd}) \log(1 - \beta_{dc}))$$

- 上述的 $LL(\theta)$ 约束条件是 $\sum_{c=1}^C \alpha_c = 1$ ，在这种约束条件下产生两个优化问题，一个是优化 α_c ，另一个是优化 β_{dc} 。 $LL(\theta_c) = \sum_{c=1}^C \sum_{n=1}^N \mathbb{I}(y_n = c) \log \alpha_c$ ，在约束条件下求偏导，分别对 λ 和 θ_c 求。
- 最终结果是

$$\alpha_i = \frac{1}{N} \sum_{n=1}^N \mathbb{I}(y_n = i), \beta_{dc} = \frac{\sum_{n=1}^N \mathbb{I}(y_n = c) x_{nd}}{N_c}, \text{ 其中 } \beta_{dc} = p(x_d = 1|y = c)$$

。

朴素贝叶斯-分类建模：

- $x_d \in \{0, 1\}$ ，而且模型的参数 $\theta = \alpha_c, \beta_{kdc}$ ， $\alpha_c = p(y = c), \beta_{kdc} = p(x_d = k|y = c)$ ， x_d 的含义是特征维度。注意这是类型分布，代表的含义是参数学习模型的参数配置为 θ^* 时，训练样本集出现的概率最大。

$$p(D; \theta) = \prod_{n=1}^N p(y_n; \theta) \prod_{n=1}^N p(x_n|y_n; \theta),$$

$$LL(\theta) = \sum_{c=1}^C \sum_{n=1}^N \mathbb{I}(y_n = c) \log \alpha_c + \sum_{c=1}^C \sum_{d=1}^D \sum_{k=1}^{K_d} \sum_{n=1}^N \mathbb{I}(y_n = c) \mathbb{I}(x_{n,d} = k) \log \beta_{kdc}$$

- 同理拆分为两个独立的子问题进行计算， $\alpha_c = \frac{N_c}{N}$ ， $\beta_{idc} = \frac{\sum_{n=1}^N \mathbb{I}(y_n=c) \mathbb{I}(x_{n,d}=i)}{N_c}$ ，分子的含义是类标签为 c 且所有样本第 d 维度为 i 的样本数量。分母的含义是类标签为 c 的样本数量。
- 朴素贝叶斯的条件下，其实结果直接相当于用频率估计概率。
- 下面用一个例子展示一下朴素贝叶斯的分类建模：

- 类别: $c \in \{0, 1\}$
 - $y = 0$: 正常 (ham)
 - $y = 1$: 垃圾 (spam)
- 特征维度: $D = 2$, 而且 $x_d \in \{0, 1\}$
 - $d = 1$: 是否包含“free” (0=否, 1=是)
 - $d = 2$: 是否包含“win” (0=否, 1=是)

所以每封邮件是一个向量: $x = (x_1, x_2)$ 。

训练数据 ($N = 6$)

我造一个小训练集:

n	$x_{n,1}$ (free)	$x_{n,2}$ (win)	y_n
1	1	1	1
2	1	0	1
3	0	1	1
4	0	0	0
5	0	0	0
6	1	0	0

- 首先计算先验, $N_1 = \sum_n I(y_n = 1) = 3$, $N_0 = 3$, 因此 $\alpha_1 = \alpha_0 = \frac{3}{6}$, 这是先验。之后计算条件概率, $\beta_{1,1,1} = \frac{2}{3}$, 其中分子的含义是标签为1且第一个维度是1的($x_{n,1} = 1$)的个数。同理有: $\beta_{0,1,1} = \frac{1}{3}$, $\beta_{1,2,1} = \frac{2}{3}, \beta_{0,2,1} = \frac{1}{3}$, $\beta_{1,1,0} = \frac{1}{3}, \beta_{0,1,0} = \frac{2}{3}$, $\beta_{1,2,0} = \frac{0}{3}$, $\beta_{0,2,0} = \frac{2}{3}$ 。估计求出值之后, 若到来一个信件 $x = (1, 1)$, 需要进行分类, 因此 $p(y = c|x) \propto p(y = c) \prod_{d=1}^D p(x_d|y = c)$, 当c=1的时候, $\alpha_1 * \beta_{1,1,1} * \beta_{1,2,1} = 0.222$, (相同目标标签找其余维度与待分类的一致) 当c=0的时候, $\alpha_0 * \beta_{xd,1,0} * \beta_{xd,2,0} = 0$, 因此选择最大的, 即垃圾邮件。注意此时的三个维度里面, 第一个维度是待分类信件的维度值, 第二个维度是考虑的特征维度, 第三个维度是假定标签。

朴素贝叶斯-高斯建模:

- 估计的参数是 μ_{dc}, σ_{dc}^2 , $\alpha_c = p(y = c)$, $p(x|y = c; \theta) = \prod_{d=1}^D N(x_d; \mu_{dc}, \sigma_{dc}^2)$, 最终的目标还是 $argmax_{\theta} p(D; \theta)$,

$$\mu_{dc} = \frac{\sum_{n=1}^N I(y_n = c) x_{n,d}}{N_c}$$

$$\sigma_{dc}^2 = \frac{\sum_{n=1}^N I(y_n = c) (x_{n,d} - \mu_{dc})^2}{N_c}$$

- 上述告诉我们如何估计参数, 下面我们在**已知参数之后也要知道应该如何预测标签**。最大类标签后验是类标签的预测值, 我们可以用**风险函数**来计算, $R(c|x) = \sum_{a=1}^C p(a|x) l(c, a)$, 这其中 $l(c, a) = 1, \text{ if } c \neq a$, $l(c, a) = 0, \text{ if } c = a$, 是0-1损失函数。因此 $R(c|x) = 1 - p(c|x)$ 。因此我们希望有最小风险就等价于**选择最大后验的类标签**。

- 通过例子我们可以推断出参数估计的方式, 但是估计的结果与数据量及数据质量关系密切, 如果数据量不足的情况下, 很容易出现无法估计的情况, 因此我们可以通过引入**模型参数先验**的方式解决数据缺失的问题。

朴素贝叶斯的示例:

Table3.1 爬山气候记录					
日期ID	天气	温度	湿度	风力	爬山
1	晴	热	高	弱	否
2	晴	热	高	强	否
3	阴	热	高	弱	是
4	雨	适中	高	弱	是
5	雨	冷	高	弱	是
6	雨	冷	中	强	否
7	阴	冷	中	强	否
8	晴	适中	中	弱	否
9	晴	冷	中	弱	是
10	雨	适中	中	弱	是
11	晴	适中	中	强	是
12	阴	适中	高	强	是
13	阴	热	中	弱	是
14	阴	冷	高	强	是
15	雨	适中	高	强	否

问: 在 “天气为雨, 温度适中, 湿度中和风力强的气候情况下” , 是否适合爬山?

$$p(\text{是}|\text{雨, 适中, 强}) = \frac{p(\text{雨, 适中, 中, 强}|\text{是})p(\text{是})}{p(\text{雨, 适中, 中, 强})} = \frac{p(\text{雨, 适中, 中, 强}|\text{是})p(\text{是})}{p(\text{雨, 适中, 中, 强}|\text{是})p(\text{是}) + p(\text{雨, 适中, 中, 强}|\text{否})p(\text{否})}$$

$$p(\text{是}|\text{雨, 适中, 中, 强}) = \frac{p(\text{雨}|\text{是})p(\text{适中}|\text{是})p(\text{中}|\text{是})p(\text{强}|\text{是})p(\text{是})}{p(\text{雨}|\text{是})p(\text{适中}|\text{是})p(\text{中}|\text{是})p(\text{强}|\text{是})p(\text{是}) + p(\text{雨}|\text{否})p(\text{适中}|\text{否})p(\text{中}|\text{否})p(\text{强}|\text{否})p(\text{否})}$$

$$p(\text{是}|\text{雨, 适中, 中, 强}) = \frac{\frac{3}{10} \times \frac{2}{5} \times \frac{3}{5} \times \frac{2}{5} \times \frac{2}{3}}{\frac{3}{10} \times \frac{2}{5} \times \frac{3}{5} \times \frac{2}{5} \times \frac{2}{3} + \frac{2}{5} \times \frac{2}{5} \times \frac{2}{5} \times \frac{1}{5} \times \frac{3}{5} \times \frac{1}{3}} = \frac{3}{4}$$

同理, $p(\text{否}|\text{雨, 适中, 中, 强}) = \frac{1}{4}$

$\frac{3}{4} > \frac{1}{4}$ 因此, 建议**爬山**

- 但是单纯的朴素贝叶斯无法解决类别未出现的问题, 此时概率会直接降低为0。

- **Dirichlet先验-朴素贝叶斯：**
- 认为参数 α 和参数 β 都服从狄利克雷分布，人工设定狄利克雷分布的超参数。
- 经过推导，最终得到的结论是**当狄利克雷分布的参数都是1的时候**，采用加一平滑，即每一类里加上1，确保不会有类中个数为0导致估计完全错误。

$$p(y = c|D) = \frac{N_c + 1}{\sum_{i=1}^C N_i + C}$$

这个公式可以理解为在估计的时候每一类额外加上1，在这样的情况下确保估计不会失败。如果计算的概率的可能性不止是2个，此时加的个数取决于有多少种的取值。

$$p(x_d = k|y = c, D) = \frac{N_{kdc} + 1}{N_c + K_d}$$

- 而如果我们**不指定狄利克雷分布的参数**，则是

$$p(y = c|D) = \frac{N_c + \alpha_c}{N + \sum_{i=1}^c \alpha_i}, \quad p(x_d = k|y = c, D) = \frac{N_{kdc} + \beta_{kdc}}{N_c + \sum_{i=1}^K \beta_{idc}}$$

- 下面就利用**加一平滑**计算结果：

未记录到湿度值为“低”的天气情况

日期ID	天气	温度	湿度	风力	爬山
1	晴	热	高	弱	否
2	晴	热	高	强	是
3	阴	热	高	弱	是
4	雨	适中	高	弱	是
5	雨	冷	中	弱	是
6	雨	冷	中	强	否
7	阴	冷	中	强	否
8	晴	适中	中	弱	是
9	晴	冷	中	弱	是
10	雨	适中	中	弱	是
11	晴	适中	中	强	是
12	阴	适中	中	强	是
13	阴	热	高	弱	是
14	阴	冷	高	强	是
15	雨	适中	高	强	否

假定先验分布都是均匀分布
 注意：这里 \mathcal{D} 代表左边的数据集，
 类标签1代表是，类标签2代表否

$$p(y = c|\mathcal{D}) = \frac{N_c + 1}{\sum_{i=1}^C N_i + C}$$

$$p(\text{爬山} = \text{是}|\mathcal{D}) = \frac{N_{\text{是}} + 1}{N_{\text{是}} + N_{\text{否}} + 2} = \frac{10 + 1}{10 + 5 + 2}$$

$$p(\text{爬山} = \text{否}|\mathcal{D}) = \frac{N_{\text{否}} + 1}{N_{\text{是}} + N_{\text{否}} + 2} = \frac{5 + 1}{10 + 5 + 2}$$

•

日期ID	天气	温度	湿度	风力	爬山
1	晴	热	高	弱	否
2	晴	热	高	强	是
3	阴	热	高	弱	是
4	雨	适中	高	弱	是
5	雨	冷	中	弱	是
6	雨	冷	中	强	否
7	阴	冷	中	强	否
8	晴	适中	高	弱	是
9	晴	冷	中	弱	是
10	雨	适中	中	弱	是
11	晴	适中	中	强	是
12	阴	适中	高	强	是
13	阴	热	中	弱	是
14	阴	冷	高	强	是
15	雨	适中	高	强	否

$$p(x_d = k|y = c, \mathcal{D}) = \frac{N_{kdc} + 1}{N_c + K_d}$$

$N_{\text{是}} = 10, K_{\text{天气}} = 3, N_{\text{雨, 天气是}} = 3$
 $p(\text{天气} = \text{雨}|\text{爬山} = \text{是}, \mathcal{D}) = \frac{3 + 1}{10 + 3} = \frac{4}{13}$

$N_{\text{是}} = 5, K_{\text{天气}} = 3, N_{\text{阴, 天气否}} = 0$
 $p(\text{天气} = \text{阴}|\text{爬山} = \text{否}) = \frac{0 + 1}{5 + 3} = \frac{1}{8}$

$N_{\text{是}} = 10, K_{\text{湿度}} = 3, N_{\text{低, 湿度是}} = 0$
 $p(\text{湿度} = \text{低}|\text{爬山} = \text{是}) = \frac{0 + 1}{10 + 3} = \frac{1}{13}$

•

- 请注意，第二个写的有问题，应该是 $N_{\text{否}} = 5$ ， $K_{\text{天气}} = 3$ 的含义是天气那一栏有三种类别，
- **模型参数最大后验估计(MAP)：** $\alpha_c = \frac{N_c + \alpha_c - 1}{N + \sum_{i=1}^c \alpha_i - C}$ ， $\beta_{kdc} = \frac{N_{kdc} + \beta_{kdc} - 1}{N_c + \sum_{i=1}^{K_d} \beta_{idc} - K_d}$ ，如果 α 的先验分布是均匀分布，则 $\alpha_c = \frac{N_c}{N}$ ， $\beta_{kdc} = \frac{N_{kdc}}{N_c}$

Lecture5. 高斯判别分析

- 首先，我们应该明白，高斯判别分析GDA是一种**生成模型**架构，而不是参数估计的方法。MLE和MAP都是参数估计的方法，而不是模型的架构。GDA假设

$$y \sim \text{Bernoulli}(\phi), x \mid y = k \sim N(\mu_k, \Sigma)$$

- 在GDA这种模型架构之下，我们可以采用该MLE或者MAP的方式来估计参数，其中MLE最大似然估计的方法，其有闭式解，如果如果参数本身有先验，即协方差矩阵服从逆威沙特分布，则是MAP的使用条件。
- **高斯判别分析是生成模型**，对于生成模型而言，其不直接断然分类结果，而是对类别先进行建模，比如说在通过身高来区分男生和女生的时候，我们先需要得知出男生身高和女生身高满足的概率分布，之后找到最高的高斯概率密度才能得出结果。简单一些来说，生成模型关心的是**数据怎么产生**，而判别模型关注的是分类边界。

ITML 参数学习

- 给定训练数据集 $\mathcal{D} = \{(x_n, y_n)\}_{n=1}^N$,
 - $x_n \in \mathbb{R}^D, y_n \in \{1, 2, \dots, C\}$
- 模型参数 $\theta = \{\alpha_c\}_{c=1}^C \cup \{\mu_c\}_{c=1}^C \cup \{\Sigma_c\}_{c=1}^C$
- 目标: 求解 θ^* , 使得

$$\theta^* = \underset{\theta}{\operatorname{argmax}} p(\mathcal{D}; \theta)$$
 - 即, 模型参数配置为 θ^* 时, 训练样本集 \mathcal{D} 出现 (生成) 的概率最大
- 方法
 - 最大似然估计
 - 最大后验估计

- 生成模型

$$p(y = c|x; \theta) = \frac{p(y = c; \theta)p(x|y = c; \theta)}{p(x; \theta)} = \frac{p(y = c; \theta)p(x|y = c; \theta)}{\sum_{c'=1}^C p(y = c'; \theta)p(x|y = c'; \theta)}$$
 - $p(y = c; \theta)$: 类标签先验分布
 - $p(x|y = c; \theta)$: 类 c 的类条件分布; 通过采样 $p(x|y = c; \theta)$, 能够生成类别 c 的数据 x
 - θ : 模型参数
- 高斯判别模型
 - $x \in \mathbb{R}^D, p(x|y = c; \theta)$ 为多元高斯分布

- 生成模型的原因是 $p(x|y = c; \theta)$, 其可以刻画出生成数据 x 的一些特征, 因此被称之为生成模型。先刻画出分布特征之后再通过贝叶斯公式等方式进行判别。 $\alpha_c = p(y = c; \theta)$ 被称之为**类后验分布**, $p(x|y = c; \theta)$ 被常称之为类条件分布。对于GDA, 我们假设**类条件分布: $p(x|y = c; \theta)$ 是高斯分布**, 最终的判别结果, 即已知 x 和 θ 推知 y 的过程可以用**类后验分布**和**高斯分布**相乘得出。给定一个数据集, 有 C 类, 我们需要求解的模型参数包括 α_c, μ_c 和 Σ_c , 一共有 $3C$ 个参数。

$$\log p(y = c|x; \theta) = \log \alpha_c - \frac{1}{2} \log |\Sigma_c| - \frac{1}{2} (x - \mu_c)^T \Sigma_c^{-1} (x - \mu_c) - \frac{D}{2} \log 2\pi$$

- 高斯判别分析下的最大似然估计(MLE):**
- 之后我们按照同样的方式进行负对数似然的计算, 我们希望最大化似然, 因此需要最小化 NLL , 结果如下:

$$NLL(\theta) = -\log p(\mathcal{D}; \theta)$$

$$= -\sum_{c=1}^C \sum_{n=1}^N \mathbb{I}(y_n = c) \log \alpha_c$$

$$-\sum_{c=1}^C \sum_{n=1}^N \mathbb{I}(y_n = c) \log \mathcal{N}(x_n; \mu_c, \Sigma_c)$$

- 又已知所有类别的联合概率 α_c 加和是1, 因此使用拉格朗日乘子计算: $L(\theta, \lambda) = NLL(\theta) + \lambda(\sum_{c=1}^C \alpha_c - 1)$ 。根据拉格朗日乘子, 肯定需要对 θ 和 λ 的梯度确保为0, 进一步拆解, 需要确保以下四个梯度的值为0。

$$\left. \begin{aligned} \frac{\partial L(\theta, \lambda)}{\partial \lambda} &= \sum_{c=1}^C \alpha_c - 1 \\ \frac{\partial L(\theta, \lambda)}{\partial \theta} &\implies \begin{cases} \frac{\partial L(\theta, \lambda)}{\partial \alpha_i}, 1 \leq i \leq C \\ \frac{\partial L(\theta, \lambda)}{\partial \mu_i}, 1 \leq i \leq C \\ \frac{\partial L(\theta, \lambda)}{\partial \Sigma_i}, 1 \leq i \leq C \end{cases} \end{aligned} \right\} \implies \begin{aligned} 0 &= \frac{\partial L(\theta, \lambda)}{\partial \lambda} = \sum_{c=1}^C \alpha_c - 1 \\ 0 &= \frac{\partial L(\theta, \lambda)}{\partial \alpha_i} = \lambda - \frac{N_i}{\alpha_i} \\ 0 &= \frac{\partial L(\theta, \lambda)}{\partial \mu_i} = \sum_{n=1}^N \mathbb{I}(y_n = i) \Sigma_i^{-1} (\mu_i - x_n) \\ 0 &= \frac{\partial L(\theta, \lambda)}{\partial \Sigma_i} = \frac{1}{2} \sum_{n=1}^N \mathbb{I}(y_n = i) (\Sigma_i^{-1} - \Sigma_i^{-1} (x_n - \mu_i)(x_n - \mu_i)^T \Sigma_i^{-1}) \end{aligned}$$

- 最终得到的三个参数的估计结果是:

$$\alpha_i = \frac{N_i}{N}; \quad \mu_i = \frac{1}{N_i} \sum_{n: y_n = i} x_n; \quad \Sigma_i = \frac{1}{N_i} \sum_{n: y_n = i} (x_n - \mu_i)(x_n - \mu_i)^T$$

- 继续从协方差矩阵 Σ_i 考虑, 但是如果 $N_i < D$, 即类别 i 的样本数小于数据维度时, **协方差矩阵会退化为半正定的矩阵**, 此时的估计不是良态。(数据维度指的是一个向量 x_n 的维度数, 对于 Σ_i 而言, 求和的矩阵的维度是 $D * D$, 因此如果 $N_i < D$ 就代表着用 N_i 个点去描述一个 D 维的空间, 这一定会导致空间的坍塌)
- 因此可以通过**共享协方差矩阵**的方式, 即 $\Sigma_i = \Sigma$ 实现不同类别之间共享同一个协方差矩阵, 这样参数量会减少。 $\Sigma = \frac{1}{N} \sum_{i=1}^C \sum_{n: y_n = i} (x_n - \mu_i)(x_n - \mu_i)^T$, 将其改变形式为: $\Sigma = \sum_{i=1}^C \frac{N_i}{N} (\frac{1}{N_i} \sum_{n: y_n = i} (x_n - \mu_i)(x_n - \mu_i)^T)$, 其后面部分的项就是 Σ_i , 因此 $\Sigma = \sum_{i=1}^C \frac{N_i}{N} \Sigma_i$, 即**LDA的协方差等于各类协方差的样本数加权平均**。这种估计方式被称为**线性判别分析(LDA)**
- 除了共享协方差矩阵这种方式之外, 还可以通过将协方差矩阵 Σ_i 建模为**对角矩阵**等方式来减少参数量, 将参数量从 $O(CD^2)$ 降低到 $O(CD)$, 或者直接设置LDA的协方差矩阵为对角矩阵, 这样参数量是 $O(D)$ 。但是这些方式都会损失一部分的性能。

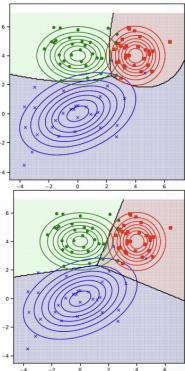
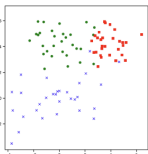
- 高斯判别分析
 - 二次决策边界

$\log p(y = c|\mathbf{x}; \boldsymbol{\theta})$
 $= \log \alpha_c - \frac{D}{2} \log 2\pi - \frac{1}{2} \log |\Sigma_c| - \frac{1}{2} (\mathbf{x} - \boldsymbol{\mu}_c)^T \Sigma_c^{-1} (\mathbf{x} - \boldsymbol{\mu}_c)$

$\log p(y = c_1|\mathbf{x}; \boldsymbol{\theta}) - \log p(y = c_2|\mathbf{x}; \boldsymbol{\theta}) = 0$

- 线性判别分析
 - 线性决策边界

$\log p(y = c|\mathbf{x}; \boldsymbol{\theta})$
 $= \boldsymbol{\mu}_c^T \Sigma^{-1} \mathbf{x} + \log \alpha_c - \frac{1}{2} \boldsymbol{\mu}_c^T \Sigma^{-1} \boldsymbol{\mu}_c - \frac{D}{2} \log 2\pi - \frac{1}{2} \log |\Sigma| - \frac{1}{2} \mathbf{x}^T \Sigma^{-1} \mathbf{x} - \log p(\mathbf{x})$



上图中的上面一个子图展示的是高斯判别分析的**二次决策边界**，即不同类别的数据采用不同的协方差矩阵。决策边界的定义就是红色部分，即两个概率分布一致的情况。在GDA进行相减的时候， $\mathbf{x}^T \Sigma_{c_1}^{-1} \mathbf{x}$ 不会抵消，因此最后边界还会剩下一个二次曲面。下面一个子图展示的是线性判别分析的决策边界，即不同类别的数据采用同样的一个协方差矩阵。

高斯判别分析下的最大后验估计(MAP):

- 由于最大似然估计方法在处理高维数据的时候估计出的协方差矩阵是奇异矩阵(半正定)居多，效果不佳，因此提出将**协方差矩阵视作随机变量**，服从**某个概率分布**的方式来计算。
- 将协方差矩阵视作随机变量，服从某个概率分布，这个概率分布被称作**逆威沙特分布**：IW。

$$\boldsymbol{\theta} = \underbrace{\{\alpha_c\}_{c=1}^C \cup \{\boldsymbol{\mu}_c\}_{c=1}^C}_{\text{未知变量 (非随机变量)}} \cup \underbrace{\{\Sigma_c\}_{c=1}^C}_{\text{随机变量}}$$

令 $\boldsymbol{\alpha} = \{\alpha_c\}_{c=1}^C, \boldsymbol{\mu} = \{\boldsymbol{\mu}_c\}_{c=1}^C$ 。假定 $\Sigma_c = \Sigma, 1 \leq c \leq C$ ；且服从如下先验分布（逆威沙特分布，inverse Wishart distribution），其中 \mathbf{S} 为 $D \times D$ 正定矩阵， ν 是自由度

$$\text{IW}(\Sigma; \mathbf{S}, \nu) = \frac{1}{Z_{\text{IW}}} |\Sigma|^{-\frac{\nu+D+1}{2}} \exp\left(-\frac{1}{2} \text{tr}(\mathbf{S}^{-1} \Sigma^{-1})\right)$$

超参数，预先设定

归一化因子

我们的目标是 $\max_{\Sigma} p(\Sigma | D; \boldsymbol{\alpha}, \boldsymbol{\mu})$ ，先贝叶斯展开，之后找到合适的优化部分，取对数求梯度。

$$\nabla_{\Sigma} \mathbf{a}(\Sigma) = \frac{\partial \mathbf{a}(\Sigma)}{\partial \Sigma} = \frac{\partial}{\partial \Sigma} \left(\sum_{n=1}^N \sum_{c=1}^C \mathbb{I}(y_n = c) \log \mathcal{N}(\mathbf{x}_n; \boldsymbol{\mu}_c, \Sigma) \right) + \frac{\partial}{\partial \Sigma} (\log W(\Sigma; \mathbf{S}, \nu)) = 0$$

$$\Sigma_{\text{ma}} = \Sigma = \frac{1}{N + D + 1} \left(\mathbf{1} + \sum_{n=1}^N \sum_{c=1}^C \mathbb{I}(y_n = c) (\mathbf{x}_n - \boldsymbol{\mu}_c)(\mathbf{x}_n - \boldsymbol{\mu}_c)^T \right)$$

- 这其中N是样本总数， ν 是自由度，D是数据维度，S是一个正定矩阵
- 通过这个结果和MLE LDA估计的结果进行对比我们可以得到MAP其实就是**IW分布众数与MLE估计的线性加和**。 $IW(\Sigma; \mathbf{S}, \nu) = \frac{S^{-1}}{\nu + D + 1}$ ，记作 Σ_0 。 $\Sigma_{\text{map}} = \lambda \Sigma_0 + (1 - \lambda) \Sigma_{\text{mle}}$
- 对于S矩阵，一般采用 $\mathbf{S} = (\nu + D + 1) \text{diag} \Sigma_{\text{mle}}$ ，对角线上MAP和MLE元素一致，但是非对角线上是 $(1 - \lambda)$ 倍的矩阵元素。
- 下面这个图展示的是在估计参数之后，应该如何利用已知的参数进行分类推断，具体的方式就是利用贝叶斯理论进行。直接求出生成函数的最后结果即可。

- Nearest centroid classifier (或nearest class mean classifier, NCM)
- 在最大似然估计中，如果假定类的先验分布服从均匀分布，即 $p(c) = \frac{1}{C}$ ，且共享类协方差矩阵 Σ ，则所求解的最优化问题变成

$$\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}}{\text{argmin}} \text{NLL}(\boldsymbol{\theta})$$

$$\text{NLL}(\boldsymbol{\theta}) = - \sum_{c=1}^C \sum_{n=1}^N \mathbb{I}(y_n = c) \log \frac{1}{C} - \sum_{c=1}^C \sum_{n=1}^N \mathbb{I}(y_n = c) \log \mathcal{N}(\mathbf{x}_n; \boldsymbol{\mu}_c, \Sigma) \quad \boldsymbol{\theta} = \{\boldsymbol{\mu}_c\}_{c=1}^C \cup \{\Sigma\}$$

- 可求解出 $\boldsymbol{\mu}_c = \frac{1}{N_c} \sum_{n: y_n = c} \mathbf{x}_n$ $\Sigma = \frac{1}{N} \sum_{i=1}^C \sum_{n: y_n = i} (\mathbf{x}_n - \boldsymbol{\mu}_i)(\mathbf{x}_n - \boldsymbol{\mu}_i)^T$
- 对类别未知数据 \mathbf{x} ，如下判定其类别 \hat{y} Mahalanobis Distance
 \mathbf{x} 与类 c 中心点 $\boldsymbol{\mu}_c$ 的马氏距离平方

$$\hat{y} = \underset{c}{\text{argmax}} \left(-\frac{1}{2} \log |\Sigma| - \frac{1}{2} (\mathbf{x} - \boldsymbol{\mu}_c)^T \Sigma^{-1} (\mathbf{x} - \boldsymbol{\mu}_c) \right) = \underset{c}{\text{argmin}} (\mathbf{x} - \boldsymbol{\mu}_c)^T \Sigma^{-1} (\mathbf{x} - \boldsymbol{\mu}_c)$$

之前提到的是GDA，即高斯判别分析，其假设生成符合一个高斯分布，我们为了减少参数将协方差矩阵共享(LDA假设)。如果我们能够在此基础上进一步假设先验是均匀的（ π_c 是均匀分布），这种情况下结果将退化为**NCM(最近类中心点分类器)**。 $\boldsymbol{\mu}_c = \frac{1}{N_c} \sum_{n: y_n = c} \mathbf{x}_n$ ， $\Sigma = \frac{1}{N} \sum_{i=1}^C \sum_{n: y_n = i} (\mathbf{x}_n - \boldsymbol{\mu}_i)(\mathbf{x}_n - \boldsymbol{\mu}_i)^T$ ，对于未知的数据，利用 $\underset{c}{\text{argmin}} (\mathbf{x} - \boldsymbol{\mu}_c)^T \Sigma^{-1} (\mathbf{x} - \boldsymbol{\mu}_c)$ 来判定。通过选择不同的协方差矩阵我们有不同的距离判定方式，比如说上述的就是马氏距离。

- 如果是考虑不同的距离估计方式，我们能得到不同的距离表达。如果 $W=I$ ，则是标准的马氏距离，但是如果 $W = \Sigma^{(-1/2)}$ ，则是LDA/GDA，这个 W 矩阵也可以设置为可学习的参数。这样就不用指定生成数据的过程服从高斯分布。这种直接从数据里学的方式是**最近类平均度量学习**。这种方式是判别模型。

• 距离度量模型

$$d^2(x, \mu_c) = \|x_n - \mu_c\|_W^2 = (W(x_n - \mu_c))^T (W(x_n - \mu_c)) = \|W(x_n - \mu_c)\|_2^2$$

• 后验分布

-注意：这是判别模型

$$p(y = c | x; \mu, W) = \frac{\exp\left(-\frac{1}{2} \|W(x_n - \mu_c)\|_2^2\right)}{\sum_{i=1}^C \exp\left(-\frac{1}{2} \|W(x_n - \mu_i)\|_2^2\right)}$$

-可通过梯度下降算法求解最优值，具体可参见后续“logistic回归”课程

Fisher线性判别 (FLDA):

- 也是缓解维度D过高的方式，其核心是通过降维技术，将数据从D维空间映射到K维空间。我们考虑的是二分类问题，标签是{1, 2}
- 具体应用过程是：首先计算 $\mu_1 = \frac{1}{N_1} \sum_{n:y_n=1} x_n$ ， $\mu_2 = \frac{1}{N_2} \sum_{n:y_n=2} x_n$ ，这是样本的均值信息，之后利用 $S_w = \sum_{n:y_n=1} (x_n - \mu_1)(x_n - \mu_1)^T + \sum_{n:y_n=2} (x_n - \mu_2)(x_n - \mu_2)^T$ ，计算出 S_w 矩阵。再然后使用 $w = S_w^{-1}(\mu_2 - \mu_1)$ 直接计算出 w ，此时需要注意，如果 S_w 不可逆的话需要加上极小正则化项 ϵI 。求出的 w 就是映射的候选方向，如果需要评估好坏，则直接使用 $J(w)$ 计算一个数， $J(w) = \frac{(m_2 - m_1)^2}{s_1^2 + s_2^2}$ ，其中 $m_2 - m_1 = w^T(\mu_2 - \mu_1)$ ， $s_k^2 = \sum_{n:y_n=k} (w^T x_n - w^T \mu_k)^2$ 。
- 如果需要进行分类的话通过 $z = w^T x$ 进行分类，对样本直接投影即可。
- FLDA的核心就是**找到一个投影方向，使得类间差在这个方向上最大，类内差在这个方向上最小**，解释为 w 的求解式。FLDA和逻辑和回归在二分类下是等价的，在二分类 + 共享协方差（LDA 假设）下：FLDA给出的投影方向 = Logistic Regression 的最优方向。

ITML Fisher线性判别分析

目标：求解 w^* ，使得

$$w^* = \operatorname{argmax}_w J(w) = \operatorname{argmax}_w \frac{(m_2 - m_1)^2}{s_1^2 + s_2^2}$$

- 希望投影到 w^* 上，两类的均值（中心点）之间的距离尽可能地远。

- 希望投影到 w^* 上，每类的类内方差尽可能地小。即，同类数据投影到 w^* 上尽可能聚集在一起

$J(w)$ 可改写成

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

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

$$S_W = \sum_{n:y_n=1} (x_n - \mu_1)(x_n - \mu_1)^T + \sum_{n:y_n=2} (x_n - \mu_2)(x_n - \mu_2)^T$$

$$S_B w = \frac{w^T S_B w}{w^T S_W w} S_W w$$

$$\lambda = \frac{w^T S_B w}{w^T S_W w}$$

$$S_B w = \lambda S_W w$$

假定 S_W 可逆

$$S_W^{-1} S_B w = \lambda w$$

$$S_B w = (\mu_2 - \mu_1)(\mu_2 - \mu_1)^T w = (\mu_2 - \mu_1)(m_2 - m_1)$$

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

对 $C(> 2)$ 类分类可类似处理

不足：只能降到 $K (\leq C - 1)$ 维

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

只关注投影方向

- FLDA也有不足：对于**C类的分类问题只能降低维度到C-1维**。本质上是类间散度矩阵

$S_B = \sum_{k=1}^C N_k(\mu_k - \mu)(\mu_k - \mu)^T$ 中的C个向量是线性相关的， $\sum_{k=1}^C N_k(\mu_k - \mu) = 0$ ，所以最多只能有 $C - 1$ 个线性独立的方向。

Lecture6. Logistic回归

- 除了之前讲解的生成式模型之外，我们还有一类模型为**判别式模型**。判别式模型直接对类标签的后验分布 $p(y = c | x; \theta)$ 进行建模，我们将讨论二项logistic回归、多项logistic回归和层次softmax。

两项logistic回归:

- 即二分类问题，可以很自然的使用**伯努利分布建模**标签后验概率，因此建模为 $p(y | x; \theta) = p^y (1 - p)^{1-y}$ ，对于离散的0/1分布，其代表了选择两个不同的概率。而我们定义 $p(y = 1 | x; \theta) = \sigma(w^T x + b)$ ，即线性组合计算出的结果过一个sigmoid函数转化为概率。
- 同样为方便计算，我们将 w, b 都新增一维度， $w = (w_0 = b, w_1, \dots, w_D)$ ， $x = (x_0 = 1, x_1, \dots, x_D)$ 。
- 我们的目标是求解出一个 w^* ，使得模型参数配置为 w^* 的时候，产生数据集D的概率最大，有最大似然估计和最大后验估计这两个方式来求解。

因此问题转化为求上述右式的最大值。

$$p(\{(y_n | x_n)\}_{n=1}^N; w) = \prod_{n=1}^N p(y_n | x_n; w)$$

$$= \prod_{n=1}^N \text{Ber}(y_n; \sigma(w^T x_n))$$

$$= \prod_{n=1}^N (\sigma(w^T x_n))^{y_n} (1 - \sigma(w^T x_n))^{1-y_n}$$

- 之后对这个公式求log，求导，得到最后的导数为0的结果是一个无法求解出w的式子，这说明无法求出二项logistic回归的闭式解，需要借助最优化技术：梯度下降进行呈现。

$$\frac{\partial}{\partial \mathbf{w}} \left(\sum_{n=1}^N y_n \log \sigma(\mathbf{w}^T \mathbf{x}_n) + \sum_{n=1}^N (1 - y_n) \log (1 - \sigma(\mathbf{w}^T \mathbf{x}_n)) \right) = 0$$

就是对上述公式求log之后再求偏导的结果，也是需要优化的对象。最终化简和链式法则之后 $\sum_{n=1}^N (y_n - \frac{1}{1 + e^{-\mathbf{w}^T \mathbf{x}_n}}) \mathbf{x}_n = 0$ ，这样求解不出w，因此需要借助其余的最优化技术。

- 梯度下降：**
- 对 $\mathcal{L}(\theta_0 + \epsilon)$ 做泰勒展开，求梯度之后得到 $\epsilon^* = -\gamma_0 \nabla_{\theta} \mathcal{L}(\theta_0)$ ，多次重复 $\theta_{t+1} = \theta_t - \gamma_t \nabla_{\theta} \mathcal{L}(\theta)$ 即可得到最小值。此时的最小值可能是全局最小值，可能是局部最小值，也可能是鞍点。如果海森矩阵是半正定矩阵的话可以得证求出来的值是全局最小值。**凸函数** ($f(\lambda x + (1 - \lambda)y) \leq \lambda f(x) + (1 - \lambda)f(y)$) 是一个性质极好的函数，其**所有局部最小值都是全局最小值**，严格凸函数有唯一的全局最小值。如果一个函数是两次可微分的，其海森矩阵 $H(x) = \nabla_x^2 f(x)$ 是半正定的，则 $f(x)$ 是凸函数，如果是正定的，则 $f(x)$ 是严格凸函数。
- 对 $NLL(\mathbf{w})$ 求梯度，之后得到的结果是

$$\nabla_{\mathbf{w}} NLL_{mle}(\mathbf{w}) = \frac{1}{N} \sum_{n=1}^N \left(\frac{1}{1 + e^{-\mathbf{w}^T \mathbf{x}_n}} - y_n \right) \mathbf{x}_n$$

ITML

梯度下降—算法描述



Input: $\mathcal{D} = \{(\mathbf{x}_n, y_n) | 0 \leq n \leq N\}$; ρ , the constant learning rate
 Output: weights \mathbf{w}

Logistic_GD (\mathcal{D}, ρ)

Initialize \mathbf{w} ; // to a vector with some small random value in each dimension

Until **the terminating condition** is met, do

$\Delta \mathbf{w} \leftarrow 0$

For each $(\mathbf{x}_n, y_n) \in \mathcal{D}$, do

$\Delta \mathbf{w} \leftarrow \Delta \mathbf{w} + \left(\frac{1}{1 + e^{-\mathbf{w}^T \mathbf{x}_n}} - y_n \right) \mathbf{x}_n$

$\mathbf{w} \leftarrow \mathbf{w} - \rho \frac{1}{N} \Delta \mathbf{w}$

这里学习率采用固定值。学习率也可以是随迭代次数动态变化的值。后续课程有介绍

$$|\mathbf{w}_{new} - \mathbf{w}_{old}| \leq \epsilon$$

- 如果训练样本数量N很大，梯度下降算法收敛可能比较慢。这个时候可以采用 minibatch 梯度下降方法，以便提高收敛速度。即，每次迭代从训练样本集中随机选K个样本，用K个样本来近似计算梯度
- ✓ 当 $K = 1$ ，也称为随机梯度下降方法 (SGD, stochastic gradient descent)

$$\begin{aligned} \nabla_{\mathbf{w}} NLL_{mle}(\mathbf{w}) &\triangleq -\frac{1}{K} \sum_{n=1}^K \left(\frac{1}{1 + e^{-\mathbf{w}^T \mathbf{x}_n}} - y_n \right) \mathbf{x}_n \\ &\approx -\frac{1}{N} \sum_{n=1}^N \left(\frac{1}{1 + e^{-\mathbf{w}^T \mathbf{x}_n}} - y_n \right) \mathbf{x}_n \end{aligned}$$

- 由于数据量过大更新起来难度比较高，一般使用minibatch的方式提高收敛速度，具体来说，SGD的方式进行梯度下降，选择**K个样本近似计算梯度**，而不是N个样本全部使用来计算梯度。
- 梯度下降是一阶优化方法，牛顿法是二阶优化方法，其能够提供显著更快的优化收敛速度。
- 牛顿法：**
- $\theta_{t+1} = \theta_t - \gamma_t (H_L(\theta_t))^{-1} \nabla_{\theta} \mathcal{L}(\theta)$ ，其中 $H_L(\theta_t) = \nabla_{\theta}^2 \mathcal{L}(\theta)$ 是海森矩阵。
- 这个结果同样可以通过泰勒展开公式得到，对其二阶泰勒近似即可得到同样的解答。

ITML

牛顿法—海森矩阵

$$H_L(\theta) = \begin{bmatrix} \frac{\partial^2 \mathcal{L}}{\partial \theta_1 \partial \theta_1} & \cdots & \frac{\partial^2 \mathcal{L}}{\partial \theta_1 \partial \theta_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 \mathcal{L}}{\partial \theta_n \partial \theta_1} & \cdots & \frac{\partial^2 \mathcal{L}}{\partial \theta_n \partial \theta_n} \end{bmatrix} \quad g_L(\theta) = \nabla_{\theta} \mathcal{L}(\theta) = \begin{bmatrix} \frac{\partial \mathcal{L}}{\partial \theta_1} \\ \vdots \\ \frac{\partial \mathcal{L}}{\partial \theta_n} \end{bmatrix}$$

$$\theta = \begin{bmatrix} \theta_1 \\ \vdots \\ \theta_n \end{bmatrix}$$

$$H_L(\theta) = \begin{bmatrix} \frac{\partial(\nabla_{\theta} \mathcal{L}(\theta))^T}{\partial \theta_1} \\ \vdots \\ \frac{\partial(\nabla_{\theta} \mathcal{L}(\theta))^T}{\partial \theta_n} \end{bmatrix} = \frac{\partial(\nabla_{\theta} \mathcal{L}(\theta))^T}{\partial \theta} = \nabla_{\theta} (\nabla_{\theta} \mathcal{L}(\theta))^T$$

- 将牛顿法代入二项logistic回归中可以得出： $H_{NLL_{mle}}(\mathbf{w}) = \frac{1}{N} \mathbf{X}^T \mathbf{D} \mathbf{X}$ ， $\mathbf{D} = \text{diag}(\hat{y}_1(1 - \hat{y}_1), \dots, \hat{y}_N(1 - \hat{y}_N))$ ，进一步更新有： $\mathbf{w}_{t+1} = \mathbf{w}_t + (\mathbf{X}^T \mathbf{D} \mathbf{X})^{-1} \mathbf{X}^T (\mathbf{y} - \hat{\mathbf{y}})$ ，这其中X是样本设计矩阵，y是真实标签向量， $\hat{\mathbf{y}}$ 是模型给出的预测概率，即 $\hat{y} = \sigma(\mathbf{w}^T \mathbf{x}_n)$ 。
- 在实际例子中，我们首先将样本展成 (x_1, x_2) 横着铺的矩阵，初始化 w_0 参数，计算预测的 \hat{y} ，再计算D矩阵，其是一个对角矩阵。之后通过公式就可以更新出新的 w_1 ，然后再预测即可。
- 在二项logistic回归中我们可以得知， $H(\mathbf{w})$ 是半正定矩阵，因此NLL函数是凸函数，梯度下降等算法求出

的解是全局最小值。

• 二项logistic回归的最大后验估计：

• 我们假定w满足先验 $\mathcal{N}(0, \frac{1}{\gamma} I)$ ，这样推导出的目标函数正是带有**L2正则化**的MLE估计结果，如果先验是Laplace分布，推导出的目标函数则是**L1正则化**的MLE估计结果。

• 在 $w \sim \mathcal{N}(0, \frac{1}{\gamma} I)$ 中， $NLL_{map}(w) = NLL_{mle}(w) + \frac{\lambda}{2} w^T w$ ，这就是L2正则化，对这个值求梯度就是

$$\nabla_w NLL_{map}(w) = \nabla_w NLL_{mle}(w) + \lambda w = \frac{1}{N} \sum_{n=1}^N (\hat{y}_n - y_n) x_n + \lambda w \quad \hat{y}_n = \sigma(w^T x_n)$$

Input: $\mathcal{D} = \{(x_n, y_n) \mid 0 \leq n \leq N\}$; ρ , the constant learning rate; $w \sim \mathcal{N}(0, \frac{1}{\gamma} I)$

Output: weights w

Logistic_MAP_GD(\mathcal{D}, ρ)

Initialize w ;

Until **the terminating condition** is met, do

$\Delta w \leftarrow 0$

For each $(x_n, y_n) \in \mathcal{D}$, do

$$\Delta w \leftarrow \Delta w + \left(\frac{1}{1 + e^{-w^T x_n}} - y_n \right) x_n$$

$$w \leftarrow \left(1 - \frac{\gamma}{N} \rho \right) w - \rho \frac{1}{N} \Delta w \quad w_{t+1} = w_t - \rho \left(\nabla_w NLL_{mle}(w_t) + \frac{\gamma}{N} w_t \right)$$

• 这种先验方式认为数据在**每一个维度上的量级是一样的**，但是这种假设是有缺陷的，一些数据量级天然比另一些数据量级更高。为此我们需要进行**数据归一化**。归一化的方式有两种，一种是**标准化**，即归一化到(-1, 1)的方式，先根据样本计算均值和方差，之后 $x_{nd} = \frac{x_{nd} - \mu_d}{\sigma_d}$ ，其中 μ 代表均值， σ 代表方差。当然也有**Min-Max归一化**，让结果归一到(0, 1)，归一的方式是 $x_{nd} = \frac{x_{nd} - x_{min}}{x_{max} - x_{min}}$

• **感知器模型**：是神经网络的初步形态，其接收一系列数据，经过一系列计算之后生成0或者1的结果，代表输出。其和logistic回归的根本不同在于：logistic回归的内部采用的是sigmoid函数，因此输出是一个连续的可求导的值，而感知器模型采用的是阶跃函数(Heaviside)， $\hat{y} = H(w^T x + b) = 1/0$ ，即内部结果大于0的时候输出1，内部结果西小于等于0的时候输出0，这是**不可导的**。更新过程是 $w = w - \rho(H(w^T x) - y_n)x_n$ ，因此这个更新是**只有预测错误的时候才会更新**，不是明确的优化目标，而是基于错误驱动的修正算法。无法解决线性不可分的问题。

• **多项logistic回归**：

• 此时是**多分类**问题，使用的是**类型分布进行建模**，因此结果函数不使用sigmoid，而使用**softmax**。结果的标签用one-hot vector进行表述。任务描述是模型参数的配置为 W^* 的时 候，以n个 x_n 为输入，n个 x_n 的出

• 目标：求解 W^* ，使得

$$W^* = \underset{W}{\operatorname{argmax}} p(\{(y_n | x_n)\}_{n=1}^N; W)$$

现的概率最大，

求解方式同样有最大似然估计和最大后验估计。

在推导的过程中需要使用**交叉熵**的概念，即 $(p, q) = - \sum_{i=1}^c p_i \log q_i$ ，推导中有

$$NLL_{mle}(w) = -\frac{1}{N} \sum_{n=1}^N \sum_{c=1}^C y_{nc} \log \hat{y}_{nc} = \frac{1}{N} \sum_{n=1}^N H(y_n, \hat{y}_n)$$

，这里面 y_n 代表的是 x_n 类标签的真实分布， \hat{y}_n 代表的是 x_n 类标签的预测分布。我们不难从交叉熵的定义中可以推断出 $(p, q) = (p) + KL(p, q)$ 。其中两项分别是熵和KL散度，真实标签的熵是一个定值，因此我们将这个问题转化为求出 $\min KL(y_n, \hat{y}_n)$ ，其含义就是尽量**使得预测分布和真实分布一致**。

• 对于求解，一般使用梯度下降和牛顿法。在**梯度下降法**之中对 $NLL_{mle}(w)$ 求梯度之后的结果是 $\frac{\partial NLL_{mle}(W)}{\partial(W)} = \frac{1}{N} \sum_{n=1}^N (\hat{y}_n - y_n) \otimes x_n^T$ ，这里的 \otimes 是**kroncker积**。

• Kronecker product

• 给定 $i \times j$ 矩阵A和 $m \times n$ 矩阵B，这两个矩阵的克罗内克积 $A \otimes B$

定义如下：

$$A \otimes B = \begin{bmatrix} a_{11}B & \cdots & a_{1j}B \\ \vdots & \ddots & \vdots \\ a_{i1}B & \cdots & a_{ij}B \end{bmatrix} \quad \text{注意：} A \otimes B \text{是} im \times jn \text{的矩阵}$$

$$\begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \\ a_{31} & a_{32} \end{pmatrix} \otimes \begin{pmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \end{pmatrix} = \begin{pmatrix} a_{11}b_{11} & a_{11}b_{12} & a_{11}b_{13} & a_{12}b_{11} & a_{12}b_{12} & a_{12}b_{13} \\ a_{11}b_{21} & a_{11}b_{22} & a_{11}b_{23} & a_{12}b_{21} & a_{12}b_{22} & a_{12}b_{23} \\ a_{21}b_{11} & a_{21}b_{12} & a_{21}b_{13} & a_{22}b_{11} & a_{22}b_{12} & a_{22}b_{13} \\ a_{21}b_{21} & a_{21}b_{22} & a_{21}b_{23} & a_{22}b_{21} & a_{22}b_{22} & a_{22}b_{23} \\ a_{31}b_{11} & a_{31}b_{12} & a_{31}b_{13} & a_{32}b_{11} & a_{32}b_{12} & a_{32}b_{13} \\ a_{31}b_{21} & a_{31}b_{22} & a_{31}b_{23} & a_{32}b_{21} & a_{32}b_{22} & a_{32}b_{23} \end{pmatrix}$$

• 对于**牛顿法**，我们需要计算海森矩阵，如上右图所示，利用Kronecker积进行不断简化运算之后得到的结果是

$$H_{NLL_{mle}}(W) = \frac{1}{N} \sum_{n=1}^N (diag(\hat{y}_n) - \hat{y}_n \hat{y}_n^T) \otimes x_n x_n^T$$

• 因此代入之前得到的牛顿法迭代求解公式中可以得知： $W_{t+1} = W_t - \rho_t (H_{NLL_{mle}}(W_t))^{-1} g_L(W_t)$ ，但是H

矩阵的大小是 $CD * CD$ ，其逆的计算复杂度是 $O(CD)^3$ ，因此这种方式求解极其困难。

• 多项logistic回归的最大后验估计：

ITML 最大后验估计：一般形式

- 如同二项Logistic回归，可以通过引入权重的先验分布，提高模型的泛化新能
- 假定 $w_c \sim \mathcal{N}(0, \Sigma)$ 且相互独立，则 $p(W) = \prod_{c=1}^C \mathcal{N}(w_c; 0, \Sigma)$

$$NLL_{map}(W) = NLL_{mle}(W) + \frac{1}{2} \sum_{c=1}^C w_c^T \Sigma^{-1} w_c = NLL_{mle}(W) + \frac{1}{2} \text{tr}(W^T \Sigma^{-1} W)$$

$$\nabla_W NLL_{map}(W) = \nabla_W NLL_{mle}(W) + \Sigma^{-1} W$$

- 对于MAP最大后验概率，假设 $w_c \sim N(0, \frac{1}{\lambda} I)$ 我们可以得出：
- $NLL_{map}(W) = NLL_{mle}(W) + \frac{\lambda}{2} \sum_{c=1}^C w_c' w_c = NLL_{mle}(W) + \frac{\lambda}{2} \text{tr}(W' W)$ ，在求梯度之后是 $\nabla_w NLL_{map}(W) = \nabla_w NLL_{mle}(W) + \lambda W$ 。
- 如果我们放宽假设， $w_c \sim N(0, \Sigma)$ 且互相独立的话， $\nabla_w NLL_{map}(W) = \nabla_w NLL_{mle}(W) + \Sigma^{-1} W$ 。

• 二类贝叶斯logistic回归：

- 目的是度量Logistic回归预测结果的不确定性，此时的w不再是一个确定的值，而是一个分布，即 $p(w|D)$ ，我们希望通过这个分布去计算 $p(y|x, D)$ ，但是极其难以计算，因为全部投票需要对w积分，为此我们采用Laplace近似，Plug-in和Monte Carlo的方式。
- 理解为二项logistic回归的一种扩展，传统的二项logistic回归目标是找出一组最优参数，而贝叶斯logistic回归的目的是找出参数分布的概率。根据之前学过的贝叶斯方法，我们的做法是找到与似然函数 $p(D|w)$ 形成共轭关系的 $p(w)$ ，这两个值是容易计算的，但是贝叶斯公式的分母 $p(D)$ 是极其困难和繁琐的。一般考虑使用共轭先验的方式计算，即之前提到的同分布族，但是对于logistic回归而言，不存在这样的 $p(w)$ 和 $p(w|D)$ 的同分布关系。因此只能考虑近似方法。
- 总结一下，使用贝叶斯的难点是：分母 $p(D)$ 难以积分求出；求出的结果 $p(w|D)$ 难以写成闭式解。

• Laplace approximation：

- 用一个高斯分布去近似参数后验。先计算MAP，之后计算Hessian，因此 $p(w|D) = N(w_{MAP}, H^{-1})$ 。
- 定义 $p(w|D) = \frac{1}{Z} e^{-E(w)}$ ，其中 $Z = p(D)$ ， $E(w) = -\log p(w, D)$ ，其目的是把求不出来的复杂后验分布 $p(w | D)$ ，在最大后验点 w^* 附近用一个高斯分布来代替，从而让积分与预测变得可计算。结论是

$$p(w|D) = \mathcal{N}(w; w^*, H(w^*)^{-1})$$

，因此 $p(D) = \int p(w, D) dw$ 就可以计算得到。

- $p(D) = e^{-E(w^*)} (2\pi)^{D/2} |H(w^*)^{-1}|^{1/2}$
- 但是为了求出真正的预测，我们还需要求出 $p(y|x; D) = \int p(y|x, w) p(w|D) dw$ ，这个积分仍然难以计算，因此还需要plug-in近似和蒙特卡洛近似这两种方式来解释。plug-in近似的目的是只考虑 w^* 位置的解，蒙特卡洛则是通过大量采样进行。
- 对于plug-in，其直接用一个代表点对应概率分布，即 $p(y = 1|x, D) = \sigma(\hat{w}^T x)$ ，对于蒙特卡洛，则是 $p(y = 1|x, D) = \frac{1}{K} \sum_{k=1}^K \sigma(w_k^T x)$ ， $w_k \sim N(w^*, H^{-1})$
- 最大熵分类模型：从多项logistic模型出发，使用 $\phi(x, c)$ 代替 x ，因此有： $p(y + c|x; w) = \frac{\exp(w_c^T \phi(x, c))}{\sum_{i=1}^C \exp(w_i^T \phi(x, i))}$ ，我们将 $\phi(x, c)$ 称之为类c的特征函数，分母 $Z(w, x)$ 称之为配分函数。

• 层次softmax：

- 计算多分类问题softmax的时候，如果顺序性计算会极其耗费时间，因此考虑使用Huffman树的结果来层次化计算。即将已有的节点顺次构建起一棵哈夫曼树，向左或者向右的概率用模型学习得到，这样只需要从root节点遍历到叶子节点一次就可以计算出一个softmax的值了， $O(n)$ 的时间复杂度降低到 $O(\log n)$ 。

ITML 霍夫曼树 

- 以类标签在训练样本集中出现的次数为权值，构造霍夫曼树
 - 权值大的节点为左子节点，权值小的节点为右子节点
 - 左子节点编码为1，右子节点编码为0

• 算法

1. 将每个标签（其出现频率为权值）看成一个树，形成有n棵树的森林
2. 在森林中选出两个根节点的权值最小的树合并，作为一棵新树的左右子树，且新树的根节点权值为其左、右子树根节点权值之和
3. 从森林中删除选出的两颗树，并将新树加入森林
4. 重复第2步和第3步，直到森林中只剩下一个树为止，该树即为所求的Huffman树

- 首先构建Huffman树，构建方式如上，之后每个结点的位置防止一个二分类器，初始化概率之后，每当输入一个值 x_n ，我们已知正确的标签，因此已知正确的路径，将其沿着正确路径走到终点之后计算路径上的概率值的乘积，这个值就是需要被优化的。使用负对数似然的方法优化即可。

- **总结：**
- 朴素贝叶斯和Logistic回归是不一样的，朴素贝叶斯是生成模型，其只需要简单计算就可以拟合数据，而logistic回归是判别模型，需要使用梯度下降等方式求解非凸优化问题，代价更高。对于生成模型而言，类条件概率是独立估计的，**新增加类别之后不需要重新训练**，而且能处理无标签数据，但是容易过拟合。判别式分类模型由于参数之间是交互的，因此**增加类别需要重新训练**，但是分类预测准确性高，而且可以兼容各种预处理技术，概率预测校准良好。

ITML	生成式分类模型 VS. 判别式分类模型	
• 生成式分类模型	• 判别式分类模型	
- 容易拟合数据	- 模型分类预测准确性更好	
✓朴素贝叶斯通过简单计算就可以拟合数据，Logistic回归需要求解凸优化问题，神经网络需要求解非凸优化问题，计算代价高	✓相比 $p(x,y)$ ， $p(y x)$ 建模更简单，更好学习，不需要建模输入数据的分布 $p(x y)$	
- 可分类别单独处理	- 可兼容各类数据预处理技术	
✓类条件概率分布式独立估计的，所以新增加类不需要重新训练。判别式由于所有参数之间是交互的，所有，增加类别需要重新训练	✓可以对输入数据 x 进行预处理，得到 x^* ，直接建模 $p(y x^*)$	
- 能方便处理无标签数据	- 概率预测校准良好	
✓ $p(x) = \sum_y p(x,y)$	✓生成式分类模型，比如朴素贝叶斯由于条件独立假定过于强，会导致极端的概率分布	

•

Lecture7. Linear Regression

- Logistic回归的最后结果是一个概率值(0, 1)，但是线性回归的最后结果是一个实数值，即通过 $w^T x + b$ 计算出 y 之后用这个值作为高斯分布的均值进行采样，最终得到 $N(\mu, \sigma^2)$ ，
- 线性回归分为**简单线性回归**、**多重线性回归**和**多元线性回归**。简单线性回归是一个输入对应一个输出，多重线性回归是多个输入对应一个输出，多元线性回归是多个输入对应多个输出。我们可以对线性回归使用最大似然模型、最大后验模型、贝叶斯线性回归模型和广义线性模型进行设计。

最大似然模型

- 有两类建模方式，一种是认为 w 是未知量，而 σ^2 是常量，另一类认为二者都是未知量。我们一般认为 $p(y|x; \theta) = \mathcal{N}(y; w^T x, \sigma^2)$
- 对于未知数是 $\theta = \mathbf{w}$ 的情况，
- $MLE(w) = \prod_{n=1}^N \mathcal{N}(y_n; w_k^T x_n, \sigma^2)$ ，因此 NLL 可以写出： $NLL = \sum_{n=1}^N (y_n - w^T x_n)^2$ ，故可以得到结论： $max_w MLE(w) = min_w NLL(w) = min_w MSE(w)$ ，**最大化似然的过程就是在最小化均方误差**。
- 我们进一步对MSE进行求解， $NLL(w) = \sum_{n=1}^N (y_n - w^T x_n)^2 = (Xw - y)^T (Xw - y)$ ，故 $\frac{\partial}{\partial w} NLL(w) = 2(X^T Xw - X^T y) = 0$ ，因此在X列满秩， $X^T X$ 是可逆矩阵的情况下：

•
$$\hat{w} = (X^T X)^{-1} X^T y$$

- 直接对其求解极其困难，如果 $X^T X$ 矩阵不可逆则无法计算，求逆矩阵也很麻烦。我们可以使用矩阵的**QR分解**（即将一个A矩阵分解为A=QR，其中Q是一个正交矩阵，R是一个上三角矩阵），我们要求Aw=z，即相当于求 $w = R^{-1} Q^T z$ ，因此用这种方式直接求解 $(X^T X)\hat{w} = X^T y$ 即可。将 $X^T X$ 直接QR分解。
- 对于其几何解释，可以理解为在X的**列向量张成的线性子空间**里找到一个向量 \hat{y} ，使之与y向量最为接近。即寻找 $\hat{y} = argmin_{y \in span(x_0, \dots, x_d)} ||y - \hat{y}||_2$
- 对于未知数是 $\theta = \mathbf{w}, \sigma$ 的情况，
- 这时 $NLL(w, \sigma^2) = \frac{1}{2\sigma^2} \sum_{n=1}^N (y_n - w^T x_n)^2 + \frac{N}{2} \log(2\pi\sigma^2)$ ，此时对w求偏导，结果是求解 $\hat{w} = (X^T X)^{-1} X^T y$ ，之后求解 $\frac{\partial}{\partial \sigma^2} NLL(\hat{w}, \sigma^2) = 0$ 。这个式子求出的结果是

$$\hat{\sigma}^2 = \frac{1}{N} \sum_{n=1}^N (y_n - \hat{w}^T x_n)^2$$

•
$$\hat{w} = (X^T X)^{-1} X^T y$$

- 如果 $X^T X$ 不可逆的话，采用梯度下降的方式求解
- **拟合度度量**用来评估回归模型对数据的**拟合程度**（好或不好），通过残差图和残差平方和和均方根误差的方式来定义。有RSS和RMSE等不同的求解方法。其中残差图包括 x_n 和 $y_n - w^T x_n$ 的对照图，也包括 y_n 和 $w^T x_n$ 的对照图。残差平方和 $RSS(W) = \sum_{n=1}^N (y_n - w^T x_n)^2$ ， $RMSE(w) = \sqrt{\frac{1}{N} RSS(W)}$ 。
- 上述我们认为线性回归是判别视角，即 $p(y|x; \theta) = \mathcal{N}(y; w^T x, \sigma^2)$ 其实线性回归也可以是**生成视角**。生成视角就是认为**已知输入信息和对输出的先验假设**，之后通过输入信息来估计输出的真正分布。
- 我们假设 $p(x, y)$ 服从多元高斯分布，我们通过最大似然估计的方式来估计 μ 和 Σ 。对此我们有对数似然函数 $LL(\mu, \Sigma) = \sum_{n=1}^N \log p(z_n; \mu, \Sigma) = -\frac{N(D+2)}{2} \log(2\pi) - \frac{N}{2} \log |\Sigma| - \frac{1}{2} \sum_{n=1}^N (z_n - \mu)^T \Sigma^{-1} (z_n - \mu)$

- 为此求解过程中我们希望先固定 Σ ，求解对 μ 的偏导，之后我们希望再固定 μ ，求解对 Σ 的偏导。
 $\frac{\partial}{\partial \mu} LL(\mu, \Sigma) = \sum_{n=1}^N \Sigma^{-1}(z_n - \mu)$ ，这里面 z 定义为 (x, y) 向量，因此

$$\hat{\mu} = \frac{1}{N} \sum_{n=1}^N z_n$$

，同理，我们通过一些换元的技巧可以计算出

$$\hat{\Sigma} = \frac{1}{N} \sum_{n=1}^N (z_n - \hat{\mu})(z_n - \hat{\mu})^T$$

- 这个最终的化简结果，即 $\hat{\Sigma} = [\hat{\Sigma}]_{xx}$ 这样排列而成的四个块的分块矩阵。

$$\hat{b} \triangleq \hat{\mu}_y - \hat{\Sigma}_{yx}(\hat{\Sigma}_{xx})^{-1}\hat{\mu}_x$$

$$\hat{w} \triangleq (\hat{\Sigma}_{yx}(\hat{\Sigma}_{xx})^{-1})^T = (X_C^T X_C)^{-1} X_C^T y_C$$

$$= (X \mathrm{C}^T X \mathrm{C})^{-1} X \mathrm{C}^T y \mathrm{C}$$

- 因此： $y = \hat{b} + \hat{w}^T x$ 是真正的解，即说明判别式和生成式的情况下二者的形式相同。

ITML 判别式 VS. 生成式

判别式

生成式

$p(y|x; \theta) = \mathcal{N}(y; \hat{w}^T x, \sigma^2)$
 $p(x, y) = \mathcal{N}(\hat{\mu}, \hat{\Sigma})$

$\mathbb{E}[y|x] = \hat{w}^T x$
 $p(y|x) = \mathcal{N}(\mu_{y|x}, \Sigma_{y|x})$

$\mathbb{E}[y|x] = \mu_{y|x} = \hat{\mu}_y + \hat{\Sigma}_{yx}(\hat{\Sigma}_{xx})^{-1}(x - \hat{\mu}_x)$
 $\mathbb{E}[y|x] = \mu_{y|x} = \hat{\mu}_y + \hat{\Sigma}_{yx}(\hat{\Sigma}_{xx})^{-1}(x - \hat{\mu}_x)$

$\hat{w} = (X^T X)^{-1} X^T y$
 $\hat{w} \triangleq (\hat{\Sigma}_{yx}(\hat{\Sigma}_{xx})^{-1})^T = (X_C^T X_C)^{-1} X_C^T y_C$

注意：仅在用高斯分布建模条件概率和联合概率分布才有相同的形式

 $b \triangleq \hat{\mu}_y - \hat{\Sigma}_{yx}(\hat{\Sigma}_{xx})^{-1}\hat{\mu}_x$

- 判别式模型**是给其一个 x ，其建模生成一个 y ，显式建模的就是 $p(y|x)$
- 生成式模型**是先建模一个 $p(x, y)$ ，一般建立为高斯，之后给一个 x ，求建模一个 y ，使用概率论的性质求出 $p(y|x) = N(\mu_y + \sum_{yx} \sum_{xx}^{-1}(x - \mu_x), \sum_{y|x})$ ，此时 $E(y|x) = \sum_{yx} \sum_{xx}^{-1} x + (\mu_y - \sum_{yx} \sum_{xx}^{-1} \mu_x)$ ，刚好就是线性形式，因此 $w = \sum_{yx} \sum_{xx}^{-1}$ ， $b = \mu_y - w^T \mu_x$ 。
- 最大后验模型：**
- ridge regression和lasso regression都是，这两个分别要求 w 的先验分布服从高斯分布或者拉普拉斯分布。
- 岭回归(L2正则化)**，其认为 w 服从以0为均值的高斯先验分布 $p(w) = \mathcal{N}(w; 0, \alpha^2 I)$ 。再根据后验分布的公式， $p(w|D) = \frac{p(D|w)p(w)}{p(D)}$ 正比于 $\mathcal{N}(w; 0, \alpha^2 \mathcal{I}) \prod_{n=1}^N \mathcal{N}(y_n; w^T x_n, \sigma^2)$ ，因此我们有 $NLL(w) \propto \sum_{n=1}^N (y_n - w^T x_n)^2 + \lambda w^T w$ ，即理解为 $\max_w p(w|D) = \min_w MSE(w) + \frac{\lambda}{N} \|w\|_2^2$ ，最终的结果是

$$\hat{w} = (X^T X + \lambda I)^{-1} X^T y$$

，同理，求解可以通过QR分解和梯度下降的方式进行。

- Lasso回归 (L1正则化)**，每个维度服从均值为0，方差为 $\frac{2}{\gamma^2}$ 的拉普拉斯分布，同理计算后验分布有：
 $p(w|D) \propto \prod_{d=1}^{D+1} Lap(w_d; 0, \frac{1}{\gamma}) \prod_{n=1}^N \mathcal{N}(y_n; w^T x_n, \sigma^2)$ ，再使用一系列的方式先化简，再求负对数似然，
 $NLL(w) = \|Xw - y\|_2^2 + \lambda \|w\|_1$ ，即理解为 $\max_w p(w|D) = \min_w MSE(w) + \frac{\lambda}{N} \|w\|_1$
- 对于lasso回归，在 $wd = 0$ 处是不可导的，因此**没有像岭回归这样的解析解**，需要使用优化算法，比如坐标下降法、投影梯度下降、近端梯度下降。下面给出一些求解Lasso回归的方式。
- 坐标下降法**是很简单高效的，其不沿着梯度方向下降寻找函数的最小值，而是依次沿着单个坐标轴的方向最小化目标的函数值，把一个**高维度的优化问题分解为了多个简单的一维的优化问题**。关注优化的NLL函数，其中的 $\|Xw - y\|_2^2$ 是下凸函数，而 $\|w\|_1$ 也是下凸函数，因此通过坐标下降法**一定能够找到全局最小值**。

$$\underset{w}{\operatorname{argmin}} f(w) = f(w_0, w_1, w_2, \dots, w_D) \rightarrow \begin{cases} w_0^{(t+1)} = \underset{w_0}{\operatorname{argmin}} f(w_0, w_1^{(t)}, w_2^{(t)}, \dots, w_D^{(t)}) \\ w_1^{(t+1)} = \underset{w_1}{\operatorname{argmin}} f(w_0^{(t+1)}, w_1, w_2^{(t)}, \dots, w_D^{(t)}) \\ w_2^{(t+1)} = \underset{w_2}{\operatorname{argmin}} f(w_0^{(t+1)}, w_1^{(t+1)}, w_2, \dots, w_D^{(t)}) \\ \vdots \\ w_D^{(t+1)} = \underset{w_D}{\operatorname{argmin}} f(w_0^{(t+1)}, w_1^{(t+1)}, w_2^{(t+1)}, \dots, w_D) \end{cases}$$

- 如果 $f(w)$ 是下凸函数，则坐标下降法能找到全局最小值
- 单权重最优解**的方法类似坐标下降法，即同样考虑每次优化一个维度上的值，最终使得整体达到优化，但是优化思路跟坐标下降法是不一样的。具体来说，坐标下降法的目的是**一次一次迭代直到最终达到目**

标值，而单权重最优解的方法强调的是**最优解**，其每个参数迭代的时候都要确保其能够直到成为最优解。

• $\alpha_d \triangleq 2 \sum_{n=1}^N x_{nd}^2$ $\beta_d \triangleq 2 \sum_{n=1}^N x_{nd} \left(y_n - \sum_{i \neq d} w_i x_{ni} \right)$

$$\hat{w}_d(\beta_d) = \begin{cases} \frac{\beta_d + \lambda}{\alpha_d}, & \beta_d < -\lambda \\ 0, & \beta_d \in [-\lambda, \lambda] \\ \frac{\beta_d - \lambda}{\alpha_d}, & \beta_d > \lambda \end{cases}$$

- $\frac{\partial}{\partial w_d} \|Xw - y\|_2^2 = \alpha_d w_d - \beta_d$, $\frac{\partial}{\partial w_d} NLL(w) = \alpha_d w_d - \beta_d + \lambda \frac{\partial}{\partial w_d} \|w\|_1$
- 上图对 $\hat{w}_d(\beta_d)$ 的描述就代表了正则化的作用，不添加正则化的时候 $\hat{w}_d(\beta_d) = \frac{\beta_d}{\alpha_d}$ ，正则化相当于施加了一种**静摩擦**，当权重不大的时候直接置为0，当权重较大的时候施加一个减少的力。
- 上述的这种计算w的方式就是**软阈值**，即防止产生明显的跳变。在利用正则化优化权重的时候，如果硬阈值，则会产生明显的跳变，进而使得一系列的权重被迫置为0，而软阈值会防止这样事情的发生，其会使得权重的减少更为平缓。我们将上述的这个公式用 $\hat{w}_d(\beta_d) = SoftThreshold(\frac{\beta_d}{\alpha_d}, \frac{\lambda}{\alpha_d})$ 进行表示。

ITML Lasso回归—坐标下降算法

- ```
1 initialize w;
2 repeat
3 for d=1 to D do
4 $\alpha_d = \frac{2}{N} \sum_{n=1}^N x_{nd}^2$;
5 $\beta_d \triangleq \frac{2}{N} \sum_{n=1}^N x_{nd} (y_n - \sum_{i \neq d} w_i x_{ni})$;
6 $w_d(\beta_d) = SoftThreshold(\frac{\beta_d}{\alpha_d}, \frac{\lambda}{\alpha_d})$;
7 until converged;
```
- **投影梯度下降(PGD)**：提出这个方法的根本原因是Lasso回归在 $w = 0$ 的位置是不可导的，为了使得其可导，我们将其拆分为两部分：一正一负，这样在每个部分之中都是可导的。通过变量拆分技巧，将 $w = w^+ - w^-$ ，其中 $w^+ = max(w, 0)$ ， $w^- = -min(w, 0)$ 。因此 $|w| = \sum_{d=0}^D |w_d| = \sum_{d=0}^D (w_d^+ + w_d^-)$ ，目标函数也可以改写为 $w^+$ 和 $w^-$ 的部分。每一次利用正常的方式进行梯度下降之后可能 $w^+$ 和 $w^-$ 不再满足约束要求，因此需要P+映射函数(构造一个 $P^+(w_d) = max\{w_d, 0\}$ )，确保约束，即权重都是正数。这样不断梯度下降+映射即可达成最终的目的。
- **近端梯度下降**：理解为投影梯度下降的升级版，其认为约束映射函数不只是求max的函数。其主要思想是 $min L(\theta) = L_s(\theta) + L_r(\theta)$ ，即将一个不好直接优化的函数拆分为两部分，一部分是**光滑可微分的**，另一部分是**不可微分的**，但是是**凸的函数**。之后先用梯度下降优化可微分的函数，再利用投影的方式优化不可微分的函数。

**ITML** 近端梯度下降

- 优化任务： $min L(\theta) = L_s(\theta) + L_r(\theta)$ 
  - 其中 $L_s(\theta)$ 是光滑可微的函数， $L_r(\theta)$ 是下凸但不可微分函数
- 主要思想
  - 先优化一步 $L_s(\theta)$ ，按照梯度下降更新当前权重向量
$$\tilde{\theta} = \theta_t - \rho_t \nabla L_s(\theta_t)$$
  - 再把 $\tilde{\theta}$ 投影到 $L_r(\theta)$ 对应的空间
$$\theta_{t+1} = \operatorname{argmin} \left( L_r(\theta) + \frac{1}{2\rho} \|\theta - \tilde{\theta}\|_2^2 \right)$$
    - ✓ $\rho$ 控制 $\theta$ 与 $\tilde{\theta}$ 的临近程度
  - 重复以上两步，直至收敛

**ITML** 讨论：线性回归、岭回归、Lasso回归

- 为讨论方便，假定 $X^T X = I$ 
$$NLL(w) = (Xw - y)^T (Xw - y) = (w^T w - 2w^T X^T y + y^T y)$$
- 线性回归
$$\hat{w}_d^{mle} = \frac{\beta_d}{\alpha_d} = \mathbf{x}_{:d}^T \mathbf{y}$$

$\mathbf{x}_{:d}$  的箭头指向 X的第d列

$$\frac{\partial}{\partial w} NLL(w) = (w - X^T y) = 0$$
$$\hat{w}^{mle} = X^T y$$
- 岭回归
$$\hat{w}_d^{ridge} = \frac{1}{1 + \lambda} \hat{w}_d^{mle}$$
$$\hat{w}^{ridge} = (X^T X + \lambda I)^{-1} X^T y = \frac{1}{1 + \lambda} X^T y$$
- Lasso回归
$$\hat{w}_d^{lasso} = \operatorname{sign}(\hat{w}_d^{mle}) \left( |\hat{w}_d^{mle}| - \frac{\lambda}{\alpha_d} \right)^+$$
$$\hat{w}_d^{lasso} = SoftThreshold\left(\frac{\beta_d}{\alpha_d}, \frac{\lambda}{\alpha_d}\right)$$
- **正则化**：即向原始模型之中引入**额外信息**，以便防止过拟合和提高模型的泛化性能的一类方法的统称。**L1正则化**的解具有稀疏性，很多的权重都是0，能够起到特征选择的作用。**L2正则化**的解具有较小的值，即权值小，表示模型的复杂度低，泛化性能更好。有分组稀疏正则的做法，参数以分组的形式实现稀疏性。分组稀疏正则就是把一组参数的二范数都放到正则项里，一起进行Lasso优化，这样的话一组里的值要么都是非0的，要么都是0。
- 当然有时候人们会将两个正则法都用上，用类似l1+l2正则的方法进行， $L_R(\theta) = L(\theta) + \lambda_1 \|\theta\|_1 + \lambda_2 \|\theta\|_2^2$
- 贝叶斯线性回归和广义线性模型暂时不提及。

Lecture8. 多层感知器

- 多层感知器可以表示**异或XOR问题**。为了增加线性模型的灵活性，有些人采用**基函数扩展**的方式进行，即把 $x$ 替换为 $\phi(x)$ ， $f(x;\theta) = Wx + b$ 改成 $f(x;\theta) = W\phi(x) + b$ ，但是对于参数 $\theta$ 来说还是线性的，而且选择合适的基函数也是一个挑战。一个很自然的思路就是再引入一个参数 $\theta_2$ ，在第一次输出之后套接上这个参数。这就是神经网络和深度学习的核心思想。
- 万能逼近定理：一个具有线性输出层和至少一层隐藏层的多层感知器，只要给与足够数量的神经元，就可以以任意精度来逼近连续函数。输入层被称为第0层。（输入层、隐层、输出层）由于所表示的函数非常复杂，无法求解析解，而且代价函数呈现非凸的性质，因此采用基于梯度下降的迭代方法学习网络参数。
- 一阶梯度的学习算法包括**梯度下降算法**、**随机梯度下降算法**和**Mini-batch随机梯度下降算法**。其中梯度下降算法BGD需要处理完所有的训练样本之后才更新一次网络参数，计算代价极高。随机梯度下降算法SGD则每输入一个训练样本就更新一次网络参数，但是这样的随机性太大，收敛慢。mini-batch SGD平衡了二者的关联，每次采样K个样本，这K个样本同时输入算法之后进行更新一次网络参数。

• 代价函数 (cost function)

- 分类 (也称损失函数, loss function)

假定标签向量 $\mathbf{y}^{(n)}$ 的第 $n_i$ 个维度值为1, 则表示第 $n$ 个输入 $\mathbf{x}^{(n)}$ 类别标签是 $class_{n_i}$

One-hot vector

✓负对数似然 (NLL)

$$J(\mathbf{b}, \mathbf{W}) = - \sum_{n=1}^N \mathbf{y}^{(n)} \log p(\hat{\mathbf{y}}^{(n)}) = - \sum_{n=1}^N \sum_{c=1}^C y_c^{(n)} \log \hat{y}_c^{(n)} = - \sum_{n=1}^N \log \hat{y}_{n_i}^{(n)}$$

- 回归

$$\hat{y}_{n_i}^{(n)} = \text{softmax}(h_{n_i}^L) = \frac{\exp(h_{n_i}^L)}{\sum_j \exp(h_j^L)}$$

✓均方误差 (MSE, 其实也是负对数似然)

$$J(\mathbf{b}, \mathbf{W}) = \frac{1}{2N} \sum_{n=1}^N (\hat{\mathbf{y}}^{(n)} - \mathbf{y}^{(n)})^T (\hat{\mathbf{y}}^{(n)} - \mathbf{y}^{(n)})$$

以 $\mathbf{x}^{(n)}$ 为输入时最后一层隐层值

- 对于分类问题，一般使用负对数似然NLL，比如说我的模型预测的结果是(0.2, 0.3, 0.5)而真实的标签是(0, 1, 0)，因此 $NLL = -\log(0.3)$ 。交叉熵的定义是： $H(p, q) = - \sum_{k=1}^K p_k \log q_k$ ，其中 $p$ 代表的是真实分布， $q$ 代表的是模型的预测分布， $K$ 代表的是类别。当采用one-hot vector对真实标签进行编码的时候，化简之后的结果就是负对数似然。因此多分类+one-hot-vector=交叉熵损失。
- 反向传播是基于**有向无环图(DAG)**进行的，即如果一个节点有诸多的子节点，则逐个子节点进行考虑并且相加得到最终的结果。 $\frac{\partial J}{\partial h_i} = \sum_{h \in children(h_i)} \frac{\partial J}{\partial h} \frac{\partial h}{\partial h_i}$ ，即对多个子节点进行收纳统计之后更新父亲节点。
- $\sigma(u) = \frac{1}{1+\exp(-u)}$ ， $\frac{\partial \sigma(u)}{\partial u} = \sigma(u)(1 - \sigma(u))$ 。
- **回归任务**：如果考虑隐藏层的最终映射是恒等映射，则反传的梯度是 $\frac{\partial J}{\partial h_c^L} = \hat{y}_c - y_c$ ，即预测的输出值减去真实的输出值。此时如果预测值大于输出值，则梯度为正值。反向传播的过程会沿着梯度的逆方向移动，因此会降低预测的输出值。
- **分类任务**：softmax+交叉熵得到的结果对最后一层logits ( $h^L$ ) 的梯度 $\frac{\partial J}{\partial h_k^L} = \hat{y}_k - y_k$ 。其中 $y_k$ 是one-hot vector，唯一设置为1的值是真实的标签。比如说，softmax后的结果是(0.2, 0.6, 0.2)，而one-hot vector的结果是(0, 1, 0)，假设最后的损失函数是交叉熵损失函数，则梯度是(0.2, -0.4, 0.2)。
- **Jacobian Matrix**：
  - 是多元函数映射的一种表征，即多元函数其实相当于多个一元函数，为了刻画这些一元函数需要使用Jacobian Matrix。为了利用Jacobian Matrix，我们有**JVP&JVP**，这其中JVP是正向模式，其 $J \cdot v$ 的结果代表输入向v方向微小移动的时候，输出的变化率是这个向量。VJP则是计算反向微分的关键核心。

$$J = \begin{bmatrix} 2x_1 & 1 \\ \cos(x_1) & 0 \end{bmatrix}$$

✎ JVP 示例 (正向模式)

取一个向量  $v = [1, 2]^T$ ,  
则:  
$$Jv = \begin{bmatrix} 2x_1 & 1 \\ \cos(x_1) & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \end{bmatrix} = \begin{bmatrix} 2x_1 * 1 + 1 * 2 \\ \cos(x_1) * 1 + 0 * 2 \end{bmatrix} = \begin{bmatrix} 2x_1 + 2 \\ \cos(x_1) \end{bmatrix}$$

解释:  
输入往方向  $[1, 2]$  微小移动时, 输出的变化率是这个向量。

✎ VJP 示例 (反向模式)

取一个向量  $v = [3, 4]$ ,  
则:  
$$v^T J = [3, 4] \begin{bmatrix} 2x_1 & 1 \\ \cos(x_1) & 0 \end{bmatrix} = [3 * (2x_1) + 4 * \cos(x_1), 3 * 1 + 4 * 0] = [6x_1 + 4 \cos(x_1), 3]$$

解释:  
如果损失函数对输出  $f_1, f_2$  的敏感度分别是 3 和 4,  
那么对输入  $x_1, x_2$  的梯度就是  $[6x_1 + 4 \cos(x_1), 3]$ 。

- 反向传播 = 对标量输出函数，沿计算图反向连续做 VJP。

- 对于**参数初始化**，如果权重符合一些特殊的正态分布的话，我们可以设置一些初始化的值，使得输入和输出的值的方差变化不大，有利于梯度更新。如果 $w_{ji}^{l+1} \sim N(0, \sigma^2)$ ， $E(o_j^l) = 0, V(o_j^l) = \gamma^2$ ，因此初始化的时候考虑 $\sigma^2 = \frac{2}{N_l + N_{l+1}}$ 即可。正如下图所示，计算出 $l + 1$ 层的方差之后如果要保证输入和输出的方差变化不大，需要 $N_l \sigma^2 = 1$ ，此时输出的方差跟输入的方差基本一致。因此可以得到结论。

考虑如下简单线性单元

其中 $N_l$ 是输入单元数量。假定 $w_{ji}^{l+1} \sim N(0, \sigma^2)$ ， $E[o_j^l] = 0$ ， $V[o_j^l] = \gamma^2$ ， $w_{ji}^{l+1}$ 与 $h_j^l$ 独立

$$h_i^{l+1} = \sum_{j=1}^{N_l} w_{ji}^{l+1} o_j^l$$

$$E[h_i^{l+1}] = E\left[\sum_{j=1}^{N_l} w_{ji}^{l+1} o_j^l\right] = \sum_{j=1}^{N_l} E[w_{ji}^{l+1} o_j^l] = \sum_{j=1}^{N_l} E[w_{ji}^{l+1}] E[o_j^l] = 0$$

$$V[h_i^{l+1}] = E[(h_i^{l+1})^2] - (E[h_i^{l+1}])^2 = \sum_{j=1}^{N_l} E[(w_{ji}^{l+1})^2 (o_j^l)^2] = \sum_{j=1}^{N_l} E[(w_{ji}^{l+1})^2] E[(o_j^l)^2] = N_l \sigma^2 \gamma^2$$

要保证输出与输入的方差变化不大，需要 $N_l \sigma^2 = 1$

类似，要反向传播的梯度方差变化不大，需要 $N_{l+1} \sigma^2 = 1$

$$\left. \begin{aligned} & \frac{1}{2} (N_l \sigma^2 + N_{l+1} \sigma^2) = 1 \\ & \sigma^2 = \frac{2}{N_l + N_{l+1}} \end{aligned} \right\}$$

- 对于**Batch Normalization**，其提出是因为使用mini-batch进行训练的时候，随着输入数据不断变化，**各层输入数据的分布也在不断变化**，因此训练时间长，且难以收敛。bn把每一层的输入数据进行归一化，确保每一层的数据分布都是稳定一致的。在训练的时候可以使用大量的输入数据对均值和方差进行估计，但是在预测的时候，一般默认使用训练时候的均值和方差进行。

mini-batch输入数据集 $B = \{x_k\}_{k=1}^K$ ， $o_{ki}^l$ 代表以 $x_k$ 为网络输入数据时，第 $l$ 层第 $i$ 个单元的输出。  
 $o_{ki}^l$ 是第 $l$ 层输出向量。 $\gamma$ 和 $\beta$ 是可学习参数。 $\varepsilon > 0$ 是超参数，置为小的常数

$$\mu_{Bi}^l = \frac{1}{K} \sum_{k=1}^K o_{ki}^l \quad \sigma_{Bi}^2 = \frac{1}{K} \sum_{k=1}^K (o_{ki}^l - \mu_{Bi}^l)^2 \quad \delta_{ki}^l = \frac{o_{ki}^l - \mu_{Bi}^l}{\sqrt{\sigma_{Bi}^2 - \varepsilon}}$$

- 对于**Momentum**，其**避免一切的更新都由梯度完全决定**，原本计算完梯度之后通过 $\theta_t = \theta_{t-1} - \rho g$ 进行更新，使用动量之后会变成 $m_t = \alpha m_{t-1} - \rho g_t$ ， $\theta_t = \theta_{t-1} + m_t$ ，其中 $m_t$ 是动量项， $\alpha$ 是动量衰减系数，一般是0.9左右， $\rho$ 是学习率。这种算法的核心是**不用梯度直接更新，而是使用梯度更新动量，动量进行更新**。
- 当然为了避免动量冲过头，也有Nestov Momentum，即 $\hat{\theta} = \theta_{t-1} + \alpha m_{t-1}$ ， $g_t = \nabla_{\theta} J(\hat{\theta})$ ， $m_t = \alpha m_{t-1} - \rho g_t$ ， $\theta_t = \theta_{t-1} + m_t$ ，即先用动量走一步，预测之后计算梯度，最后再用预测的梯度来更新动量和 $\theta$ 。
- 对于**学习率**的设置，除了直接设置之外还有包括AdaGrad, RMSProp, Adam等方式自适应设置学习率。随机梯度下降算法是具有误差的，因此保证随机梯度下降方法收敛的充分条件是**Robbins-Monro条件**，（学习率本身趋于0，学习率加和趋于无穷，学习率的平方加和小于无穷）即一开始线性递减学习率，等到迭代步数足够多之后保持学习率不变。
- $g_t$ 代表小批量的梯度， $*$ 代表是Hadamard逐元素相乘， $\rho$ 是基准学习率，下面来考虑一下几种学习率的更新方式：**AdaGrad**即累计每个参数历史梯度的平方和，对于经常被更新梯度较大的坐标，有效步长会变小进而加速收敛，但是也有明显的缺点，即后期学习率极其容易衰减到0，学习率本身是一个单调递减的函数。**RMSProp**改进了AdaGrad，即对其有一个遗忘因子，防止平方和越积越大。**Adam**则加入了动量的机制，一阶矩计算动量，二阶矩计算平均平方梯度，最终进行偏置的校正和更新。校正可以确保步伐一开始较大。

| 3) Adam                                                                                                                                                                                                                                                                                                                                                                                                                                                             |                                                                 |  |
|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------|--|
| 口号: RMSProp + Momentum (再加偏置校正)                                                                                                                                                                                                                                                                                                                                                                                                                                     |                                                                 |  |
| 公式                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |                                                                 |  |
| <ul style="list-style-type: none"> <li>一阶矩 (动量/平均梯度): <math>s_t = \lambda_1 s_{t-1} + (1 - \lambda_1) g_t</math></li> <li>二阶矩 (平均平方梯度): <math>r_t = \lambda_2 r_{t-1} + (1 - \lambda_2) (g_t \odot g_t)</math></li> <li>偏置校正 (因为从 0 开始估计, 会偏小): <math>\hat{s}_t = \frac{s_t}{1 - \lambda_1^t}</math>, <math>\hat{r}_t = \frac{r_t}{1 - \lambda_2^t}</math></li> <li>更新: <math>\Delta \theta_t = -\rho \frac{\hat{s}_t}{\sqrt{\hat{r}_t + \delta}}</math></li> </ul> |                                                                 |  |
| 1) AdaGrad                                                                                                                                                                                                                                                                                                                                                                                                                                                          | 2) RMSProp                                                      |  |
| 公式                                                                                                                                                                                                                                                                                                                                                                                                                                                                  | 思路: 给 AdaGrad 加一个“遗忘因子” (滑动平均), 别让平方和越积越大。                      |  |
| 累积每个参数历史梯度的“平方和”:                                                                                                                                                                                                                                                                                                                                                                                                                                                   | 公式                                                              |  |
| $a_t = a_{t-1} + g_t \odot g_t, a_0 = 0$                                                                                                                                                                                                                                                                                                                                                                                                                            | 对梯度平方做指数滑动平均 (EMA):                                             |  |
| 更新:                                                                                                                                                                                                                                                                                                                                                                                                                                                                 | $a_t = \lambda a_{t-1} + (1 - \lambda) (g_t \odot g_t)$         |  |
| $\Delta \theta_t = -\frac{\rho}{\sqrt{a_t + \delta}} \odot g_t$                                                                                                                                                                                                                                                                                                                                                                                                     | 更新:                                                             |  |
|                                                                                                                                                                                                                                                                                                                                                                                                                                                                     | $\Delta \theta_t = -\frac{\rho}{\sqrt{a_t + \delta}} \odot g_t$ |  |

- 并行训练**: 分为**模型并行**和**数据并行**，其中模型并行是把模型分割在不同的机器上进行训练，数据并行是把数据分割到不同的机器上进行训练，每个机器运行一个完整的模型算法。对于数据并行，把数据集拆分为 $k$ 部分，分别发送给 $k$ 个机器。每个机器都会计算自身部分的梯度，之后将值汇总给主机，主机把全体的值得到之后相加sum求和，然后把sum value传给 $K$ 个机器。 $K$ 个机器用sum值分别更新自己的梯度。
- 梯度爆炸**: 是极其容易发生的，为了避免梯度爆炸，可以使用**梯度裁剪**或者**修改激活函数**。而梯度消失的话则需要**修改激活函数**或者使用**残差连接**。
- $\sigma$ 即sigmoid函数不可以用于神经网络的训练，其导数值一定处于(0, 0.25)，因此多层下几乎肯定会发生梯度消失。softmax是sigmoid函数在多分类问题上的泛化版。tanh函数，即 $\tanh(u) = \frac{e^u - e^{-u}}{e^u + e^{-u}}$ 的梯度也是位于0到1之间，可以视为sigmoid函数的拉伸版本，也有梯度消失的风险。Hard Tanh函数计算代价低，可以避免梯度消失的问题，但是只有中间区域会被更新。ReLU则避免梯度消失的范围更为广阔。但是面对

负数值神经网络仍然难以更新，因此有Leaky ReLU和Param ReLU，还有SELU、GELU和Swish等函数。

ITML

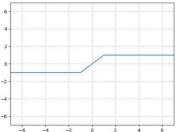
激活函数：Hard Tanh

• Hard Tanh函数

$$\text{HTanh}(u) = \begin{cases} -1, & \text{if } u < -1 \\ u, & \text{if } -1 \leq u \leq 1 \\ 1, & \text{if } u > 1 \end{cases} \quad \frac{\partial \text{HTanh}(u)}{\partial u} = \begin{cases} 1, & \text{if } -1 \leq u \leq 1 \\ 0, & \text{otherwise} \end{cases}$$

-与Tanh相比，Hard Tanh计算代价低，而且可以避免梯度消失问题

-但是，如果输入值在[-1,1]之外，梯度均为0，这意味着不会进行权重更新。因此，不利于基于梯度下降的权重学习



证明早停法与L2正则化是一致对应的：我们从假设损失函数在最优点  $w^*$  附近可被二次近似开始： $J(w) \approx J(w^*) + \frac{1}{2}(w - w^*)^\top H(w - w^*)$  因此其梯度为  $\nabla_w J(w) = H(w - w^*)$ 。梯度下降的更新公式为  $w^{(\tau)} = w^{(\tau-1)} - \epsilon \nabla_w J(w^{(\tau-1)}) = w^{(\tau-1)} - \epsilon H(w^{(\tau-1)} - w^*)$  移项可得： $w^{(\tau)} - w^* = (I - \epsilon H)(w^{(\tau-1)} - w^*)$  由于  $H$  为对称半正定矩阵，可作特征分解  $H = Q\Lambda Q^\top$ 。令  $u^{(\tau)} = Q^\top(w^{(\tau)} - w^*)$ ，则有  $u^{(\tau)} = (I - \epsilon\Lambda)u^{(\tau-1)}$  递推展开并假设  $w^{(0)} = 0$ ，得到  $u^{(\tau)} = [I - (I - \epsilon\Lambda)^\tau]w^*$  这表明在每个特征方向  $i$  上： $u_i^{(\tau)} = [1 - (1 - \epsilon\lambda_i)^\tau]w_i^*$  现在考虑  $L_2$  正则化问题： $\min_w J(w) + \frac{\alpha}{2}\|w\|^2$  在二次近似下有  $(H + \alpha I)\tilde{w} = Hw^*$ ，于是  $\tilde{w} = (H + \alpha I)^{-1}Hw^*$  代入  $H = Q\Lambda Q^\top$  得  $Q^\top \tilde{w} = [(\Lambda + \alpha I)^{-1}\Lambda]Q^\top w^*$  即在每个特征方向上  $\tilde{w}_i = \frac{\lambda_i}{\lambda_i + \alpha}w_i^*$  因此，早停法与  $L_2$  正则化的两种结果可写为  $Q^\top w^{(\tau)} = [I - (I - \epsilon\Lambda)^\tau]Q^\top w^*$ ， $Q^\top \tilde{w} = [(\Lambda + \alpha I)^{-1}\Lambda]Q^\top w^*$  若二者在特征方向上等价，则需满足  $1 - (1 - \epsilon\lambda_i)^\tau \approx \frac{\lambda_i}{\lambda_i + \alpha}$  当  $\epsilon\lambda_i \ll 1$  时，有指数近似  $(1 - \epsilon\lambda_i)^\tau \approx e^{-\epsilon\lambda_i\tau}$ ，于是

$$1 - e^{-\epsilon\lambda_i\tau} \approx \frac{\lambda_i}{\lambda_i + \alpha}$$

Dropout则随机切断一部分的神经元的连接，其实就相当于把一个神经网络改组为多个神经网络连接起来的结果。

Lecture9. 卷积神经网络

稀疏链接、参数共享是主要特征，稀疏连接的意思是局部连接，稀疏交互，一个图上的像素不会跟所有其余的像素都产生关联。虽然连接稀疏，但是高层节点还是可以感受到大部分的底层输入数据。

注意

• 输入： $L_i \times L_i \times D$

• 卷积层四个超参数

-卷积核宽（高）： $F$

-卷积核个数： $K$

-步长： $S$

-零值填充： $P$

• 输出： $L_o \times L_o \times K$

$L_o = (L_i + 2P - F)/S + 1$

这里假定输入的高和宽相等（默认形式），但相关结论可自然推广到高和宽不等场景

由于卷积核深度等于输入深度，故在表示卷积核大小的时候就忽略深度，卷积核大小用宽×高（ $F \times F$ ）表示。实际应该是宽×高×深（ $F \times F \times D$ ）

每个卷积核参数数量为 $F \times F \times D$ 。另外，每个卷积核附带1个偏置。故总的参数数量为  $(F \times F \times D + 1) \times K$

$K$ 通常为2的幂次方例如：32, 64, 128, 512

$F$ 通常取1, 3, 5等  $S$ 通常取1或2  $P$ 视需要定

重点的公式就是计算输出尺寸的，即 $L_o = (L_i + 2P - F)/S + 1$ 。一般都会选择尺寸较小的卷积核，这是因为卷积一个同样大小的图，小卷积核的参数数量更少，对内存的消耗更少。zero padding可以保存边缘信息。

batchnorm是对整个batch，每个通道单独计算均值和方差，比如说一个班级的考试成绩，整个班级是一个batch，每科是一个channel，因此batchnorm计算的是全班级每个科的均值和方差。Layernorm计算的是每个样本的所有维度的均值和方差，即每个batch样本下一个例子的情况，对应到班级得分中就是一个学生的三门成绩的平均分和方差。Instancenorm是对每个样本的每个通道计算均值和方差。Groupnorm是Instancenorm的进阶版，即选择多个通道计算均值和方差。

ITML

标准化

• 为了方便卷积神经网络的训练，通常在卷积后（激活处理前）或激活处理后进行标准化，有四类标准化方式，具体如下图所示

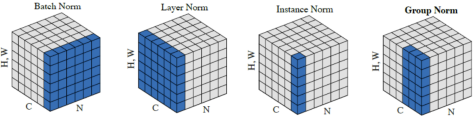
-Batch Norm（广为使用）、Layer Norm、Instance Norm和Group Norm

Batch Norm

Layer Norm

Instance Norm

Group Norm



请注意，卷积核是有深度的，即多个卷积核是存在的，其可以把特征图的深度信息增大（RGB是3，可以增大到10及以上）。池化层是没有深度的，其也没有参数。

- 输入:  $32 \times 32 \times 3$
- 卷积核:  $5 \times 5 \times 3$ 
  - 共6个
  - 一个卷积核参数
  - $5 \times 5 \times 3$
- 步长: 1
- 零值填充: 0
- 每个卷积核带1个偏置参数
- 卷积层参数总数: 456
- 输出:  $28 \times 28 \times 6$ 
  - 宽:  $(32 + 2 \times 0 - 5) / 1 + 1$
  - 高:  $(32 + 2 \times 0 - 5) / 1 + 1$
  - 深度: 6
  - $6 \times (5 \times 5 \times 3 + 1)$

- 任何一个全连接层均可转换成一个卷积层，一个输入为 $7 \times 7 \times 512$ ，输出为4096的全连接层等价于具有4096个 $7 \times 7 \times 512$ 卷积核，步长为1，零值填充为0的卷积层。
- 至于反向传播，其核心还是上一讲提到的MLP的机制，当一个结点有多个子结点的时候，将这些子节点的梯度求和即可。即每个子节点上都考虑梯度，如下图的蓝色部分显示的一样。（下图展示的是2维情况下的反向传播，对于3维度的情况与这个是一样的）

$$h_{i,j}^l = \sum_{w=0}^{F_l-1} \sum_{h=0}^{F_l-1} o_{w+i,h+j}^{l-1} \times w_{w,h}^l + b^l$$

第 $l$ 层的尺寸 $N_l \times N_l$

$w_{a,b}^l \longrightarrow$ 

|                 |          |                     |
|-----------------|----------|---------------------|
| $h_{0,0}^l$     | $\cdots$ | $h_{0,N_l-1}^l$     |
| $\vdots$        | $\ddots$ | $\vdots$            |
| $h_{N_l-1,0}^l$ | $\cdots$ | $h_{N_l-1,N_l-1}^l$ |

 $\xrightarrow{\hspace{1cm}} \mathcal{L}$

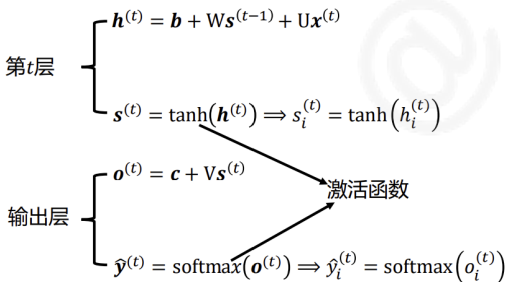
$$\frac{\partial \mathcal{L}}{\partial w_{a,b}^l} = \sum_{i=0}^{N_l-1} \sum_{j=0}^{N_l-1} \frac{\partial \mathcal{L}}{\partial h_{i,j}^l} \frac{\partial h_{i,j}^l}{\partial w_{a,b}^l} = \sum_{i=0}^{N_l-1} \sum_{j=0}^{N_l-1} \delta_{i,j}^l o_{a+i,b+j}^{l-1}$$

$$\frac{\partial \mathcal{L}}{\partial b^l} = \sum_{i=0}^{N_l-1} \sum_{j=0}^{N_l-1} \frac{\partial \mathcal{L}}{\partial h_{i,j}^l} \frac{\partial h_{i,j}^l}{\partial b^l} = \sum_{i=0}^{N_l-1} \sum_{j=0}^{N_l-1} \frac{\partial \mathcal{L}}{\partial h_{i,j}^l} = \sum_{i=0}^{N_l-1} \sum_{j=0}^{N_l-1} \delta_{i,j}^l$$

$\delta_{i,j}^l \triangleq \frac{\partial \mathcal{L}}{\partial h_{i,j}^l}$

- $\frac{\partial \mathcal{L}}{\partial W} = \sum_{i,j}$  对应patch，比如说上一层的输出是 $3 \times 3$ 的1-9的矩阵，卷积核是 $2 \times 2$ 的 $w_1$ 到 $w_4$ ，反向传播的时候  $\frac{\partial \mathcal{L}}{\partial w_1} = 12$ ，即卷积核 $w_1$ 扫到的那四个位置：1，2，4，5。这四个数的和就是12。
- 对于池化层而言，反向传播会记录原本的位置信息，之后更新到指定的位置上，即生成一个与输入矩阵相同大小的矩阵，其中选择的位置上写上原本的数，没有选择的位置上置0。Avgpooling则是平滑过渡。
- 膨胀卷积：在卷积核的元素之间插空，其目的是在不增加参数的情况下扩大感受野。比如说  $x = (x_0, x_1, x_2, x_3, x_4, x_5, x_6)$  里面  $w = (w_0, w_1, w_2)$ ，如果采用膨胀卷积的话就跳着进行卷积。
- 转置卷积：放大空间尺寸，可学习的

## Lecture10. LSTM长短期记忆网络



- 为处理序列数据产生的，序列中的数据不是独立同分布的，长度不定，而且前后之间存在相关性。为了实现与序列长度无关的参数量，一般采用参数共享， $s_t = f(x_t, s_{t-1}; \theta)$ ，这就是循环结构。

| 符号                                          | 意义                             | 符号                              | 意义         |
|---------------------------------------------|--------------------------------|---------------------------------|------------|
| $K$                                         | 标签数量                           | $W \in \mathbb{R}^{H \times H}$ | 隐层到隐层的权重矩阵 |
| $T$                                         | 序列长度 (时间步总数)                   | $U \in \mathbb{R}^{H \times D}$ | 输入到隐层的权重矩阵 |
| $H$                                         | 隐层的维数                          | $V \in \mathbb{R}^{K \times H}$ | 隐层到输出的权重矩阵 |
| $x^{(t)} \in \mathbb{R}^{D \times 1}$       | $t$ 时刻输入数据                     | $b \in \mathbb{R}^{H \times 1}$ | 输入到隐层的偏置   |
| $y^{(t)} \in \mathbb{R}^{K \times 1}$       | $t$ 时刻输出的标签真实概率分布 (one-hot 向量) | $c \in \mathbb{R}^{K \times 1}$ | 隐层到输出的的偏置  |
| $\hat{y}^{(t)} \in \mathbb{R}^{K \times 1}$ | $t$ 时刻输出的标签预测概率分布              |                                 |            |
| $h^{(t)} \in \mathbb{R}^{H \times 1}$       | $t$ 时刻隐层 (输入求和, 激活前)           |                                 |            |
| $s^{(t)} \in \mathbb{R}^{H \times 1}$       | $t$ 时刻隐层输出 (激活后)               |                                 |            |
| $o^{(t)} \in \mathbb{R}^{K \times 1}$       | $t$ 时刻输出                       |                                 |            |

|               |                   |          |
|---------------|-------------------|----------|
| 值             | 标签                |          |
| 0             | $\leftrightarrow$ | 1        |
| $\vdots$      | $\leftrightarrow$ | $\vdots$ |
| $y^{(t)} = 1$ | $\leftrightarrow$ | $l_t$    |
| $\vdots$      | $\leftrightarrow$ | $\vdots$ |
| 0             | $\leftrightarrow$ | $K$      |

$\nearrow$   
 $t$ 时刻的真实标签

- $h^{(t)} = b + Ws^{(t-1)} + Ux^{(t)}$   
 $s^{(t)} = \tanh(h^{(t)})$   
 $o^{(t)} = c + Vs^{(t)}$   
 $\hat{y}^{(t)} = \text{softmax}(o^{(t)})$
- $\mathcal{L} = \sum_{t=1}^T \mathcal{L}^{(t)}$   
 $\mathcal{L}^{(t)} = - (y^{(t)})^T \log(\hat{y}^{(t)}) = - \sum_{k=1}^K y_k^{(t)} \log(\hat{y}_k^{(t)})$
- $L = \sum_{t=1}^T L^{(t)}$ ，而我们需要求出的梯度包括： $\frac{\partial L}{\partial c}$ ， $\frac{\partial L}{\partial b}$ ， $\frac{\partial L}{\partial V}$ ， $\frac{\partial L}{\partial W}$ ， $\frac{\partial L}{\partial U}$ ，首先对于 $y^{(t)}$ ， $\frac{\partial L}{\partial \hat{y}^t} = \frac{\partial L^{(t)}}{\partial \hat{y}^t}$ ；因此  $\frac{\partial L^{(t)}}{\partial \hat{y}^{(t)}} = -y^{(t)} * \frac{1}{\hat{y}^{(t)}}$ ，比如说预测标签是(0.2, 0.5, 0.3)，真实标签是(0, 1, 0)，则  $\frac{\partial L}{\partial \hat{y}} = (0, -2, 0)$ ，原因是真实标签为0的类别展示为0，真实标签为1的类别值为预测标签的负倒数。

- 其次对于 $o^{(t)}$ ， $\frac{\partial \mathcal{L}}{\partial o^{(t)}} = \frac{\partial \hat{y}^{(t)}}{\partial o^{(t)}} * \frac{\partial \mathcal{L}}{\partial \hat{y}^{(t)}} = \hat{y}^{(t)} - y^{(t)}$ ，
- 对于 $c$ ， $\frac{\partial \mathcal{L}}{\partial c} = \sum_{t=1}^T (\hat{y}^{(t)} - y^{(t)})$
- 对于 $b$ ， $\frac{\partial \mathcal{L}}{\partial b} = \sum_{t=1}^T \frac{\partial \mathcal{L}}{\partial h^{(t)}}$

$$\frac{\partial \mathcal{L}}{\partial v_{ij}} = \sum_{t=1}^T \frac{\partial o^{(t)}}{\partial v_{ij}} \frac{\partial \mathcal{L}}{\partial o^{(t)}} = \sum_{t=1}^T s_j^{(t)} (\hat{y}_i^{(t)} - \mathbb{I}(i = l_t)) \quad \frac{\partial \mathcal{L}}{\partial w_{ij}} = \sum_{t=1}^T \frac{\partial h^{(t)}}{\partial w_{ij}} \frac{\partial \mathcal{L}}{\partial h^{(t)}} \quad \frac{\partial \mathcal{L}}{\partial u_{ij}} = \sum_{t=1}^T \frac{\partial h^{(t)}}{\partial u_{ij}} \frac{\partial \mathcal{L}}{\partial h^{(t)}}$$

- 对于 $h^{(t)}$ ，其展示的是损失函数变化对隐藏状态的影响，不止会影响在这之前的隐藏状态，还会影响这之后的所有的隐藏状态，具体的计算式如下：

$$\frac{\partial \mathcal{L}}{\partial h^{(t)}} = \frac{\partial s^{(t)}}{\partial h^{(t)}} \frac{\partial o^{(t)}}{\partial s^{(t)}} \frac{\partial \mathcal{L}}{\partial o^{(t)}} + \frac{\partial s^{(t)}}{\partial h^{(t)}} \frac{\partial h^{(t+1)}}{\partial s^{(t)}} \frac{\partial \mathcal{L}}{\partial h^{(t+1)}}$$

$$= \text{diag}(1 - (s^{(t)})^2) \left( V^T (\hat{y}^{(t)} - y^{(t)}) + W^T \frac{\partial \mathcal{L}}{\partial h^{(t+1)}} \right)$$

按时间反向传播

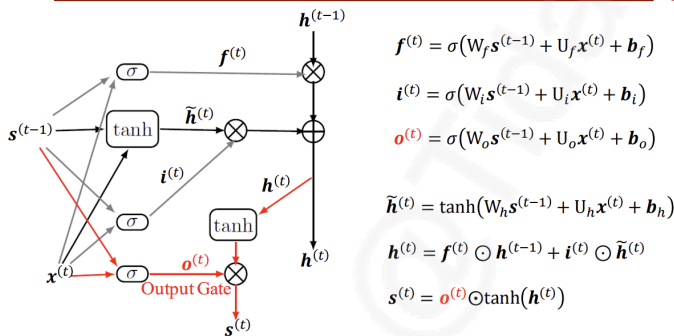
- 我们可以通过一个具体的示例展示一下，假设 $W=0.5$ ， $V=1$ ， $\hat{y}^{(t)} - y^{(t)} = 1$ ， $1 - (s^{(t)})^2 = 1$ ，比如说一共有 $t=3$ 步，最后一步的时候 $\frac{\partial \mathcal{L}}{\partial h^{(3)}} = 1 * (1) = 1$ ， $\frac{\partial \mathcal{L}}{\partial h^{(2)}} = 1 * (1 + 0.5 * 1) = 1.5$ ， $\frac{\partial \mathcal{L}}{\partial h^{(1)}} = 1 * (1 + 0.5 * 1.5) = 1.75$ ，这说明越靠前的 $h$ ，梯度越大，前面的 $W$ 会通过通路一直影响之后所有的时间步。
- 循环神经网络存在一些内在问题，具体来说，在处理长序列数据的时候，会有梯度消失和梯度爆炸的问题，门控单位元是接解决梯度消失的有效技术。
- 对于一个简化的循环神经网络而言， $h^{(t)} = Wh^{(t-1)}$ ，显然 $\frac{h^{(t)}}{\partial h^{(t-1)}} = W^T$ 。

$$\nabla_{h^{(j)}} \mathcal{L}^{(i)} = \frac{\partial h^{(j+1)}}{\partial h^{(j)}} \times \frac{\partial h^{(j+2)}}{\partial h^{(j+1)}} \times \dots \times \frac{\partial h^{(i-1)}}{\partial h^{(i-2)}} \times \frac{\partial h^{(i)}}{\partial h^{(i-1)}} \times \nabla_{h^{(i)}} \mathcal{L}^{(i)}$$

$$= (W^T)^k \times \nabla_{h^{(i)}} \mathcal{L}^{(i)}$$

- 假设 $W$ 是对称矩阵，我们将其正交对角化， $W = Q\Lambda Q^T$ ，其中 $Q$ 是正交矩阵， $\Lambda$ 是对角矩阵，对角线上的元素是 $W$ 的特征值。我们不妨假设 $W$ 的特征值的绝对值都小于1，因此梯度会趋于0。

#### LSTM:



$$f^{(t)} = \sigma(W_f s^{(t-1)} + U_f x^{(t)} + b_f)$$

$$i^{(t)} = \sigma(W_i s^{(t-1)} + U_i x^{(t)} + b_i)$$

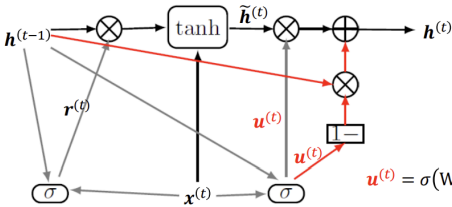
$$o^{(t)} = \sigma(W_o s^{(t-1)} + U_o x^{(t)} + b_o)$$

$$\tilde{h}^{(t)} = \tanh(W_h s^{(t-1)} + U_h x^{(t)} + b_h)$$

$$h^{(t)} = f^{(t)} \odot h^{(t-1)} + i^{(t)} \odot \tilde{h}^{(t)}$$

$$s^{(t)} = o^{(t)} \odot \tanh(h^{(t)})$$

#### GRU:



$$u^{(t)} = \sigma(W_u h^{(t-1)} + U_u x^{(t)} + b_u)$$

$$r^{(t)} = \sigma(W_r h^{(t-1)} + U_r x^{(t)} + b_r)$$

$$\tilde{h}^{(t)} = \tanh(W_h (r^{(t)} \odot h^{(t-1)}) + U_h x^{(t)} + b_h)$$

$$h^{(t)} = (1 - u^{(t)}) \odot h^{(t-1)} + u^{(t)} \odot \tilde{h}^{(t)}$$

- $u^{(t)}$ 被称作是修正门，其目的是是否更新当前状态。 $r^{(t)}$ 是重置门，其目的是控制忘掉过去还是充分利用过去。候选状态的目的是如果我要更新，我更新成啥。最终状态是慢慢换，还是保持。
- 双向循环神经网络：在每一个时间步上可以同时获得前后时间步传递的信息，上下文信息更为全面。还有深度循环神经网络、强制监督循环神经网络

## Lecture11. GNN 图神经网络

- 图表示网络：
- 早期使用图数据学习，给定图，首先抽取节点、边和图层面的特征，之后基于SVM和神经网络的方式建立特征到标签的映射。
- 图表示学习：设计任务无关的算法学习图特征嵌入表示，因此任务是把相似嵌入的节点投影到嵌入空间之中。我们定义 $similarity(u, v) = z_v^T z_u$ ，单独的内积有问题，因为值与向量的长度有关系。我们希望学

习 $ENC(v) = z_v$ ，最容易想到的编码方式是使用编码器 $Z \in R^{d \times V}$ ， $Enc(v) = Zv$ 。

- 如何定义结点相似：基于**随机游走**的方式，如果一个节点通过在一定步数内的**随机游走**的方式能够到达另一个节点，则称这两个节点是相似的。以一个节点为起始点，随机选择邻居节点然后移动到该邻居节点，之后继续随机游走，直到到达一定的长度。因此我们可以定义最大化似然函数

$$\max_{\theta} \sum_{u \in V} \log P(N_R(u)|u) = \max_{\theta} \sum_{u \in V} \sum_{v \in N_R(u)} \log P(v|u)$$

，这个等号前面部分的含义是对图中的每个节点而言考虑与自己的邻居的最大似然概率，而我们认为**邻居节点是相互独立的**，因此 $P(x_1, x_2, x_3|x_0) = P(x_1|x_0)P(x_2|x_0)P(x_3|x_0)$ ，取log之后改成加法运算。

$$P(v|u) = \frac{\exp(z_u^T z_v)}{\sum_{n \in V} \exp(z_u^T z_n)}$$

，这个值 $\propto z_u^T z_v$ ，因此是合理的。如果图的节点数量太大，**对数似然计算代价太高** ( $O(n^2)$ )。因此我们采用**负采样(Negative Sampling)**的方式。

-  $n_i \sim P(v)$ ，节点的随机分布

-  $k=5-20$

- $\log \frac{\exp(z_u^T z_v)}{\sum_{n \in V} \exp(z_u^T z_n)} \rightarrow \log \frac{1}{1 + \exp(-z_u^T z_v)} - \sum_{i=1}^k \log \frac{1}{1 + \exp(-z_u^T z_{n_i})}$
- 描述一张图的方式有两种方案，第一种是把每个节点的**嵌入表示直接相加**得到，另一种是**构造一个虚节点**，虚节点与所有节点相连，这个**虚节点的嵌入表示**就是图的嵌入表示。这种图表示学习有局限性，本质上是**直推式学习**，不能获得训练集之外的节点嵌入表示，不具有泛化性，而且不能刻画结构的相似性（有时候结构一致但是无法随机游走到）
- 图神经网络：**
- 不是CNN或者RNN，可以使用CNN或者RNN等方式来构造其
- 核心思想是为每个节点定义了一个**计算图**，希望利用**邻居节点的信息来构造本节点**。我们应该得知如何处理不定长的节点数。消息、聚合、层连接、图扩展、学习目标是五个主要目标。
- 消息计算：**目的是**每个节点都会生成一个消息**，在后续操作中将该消息将送到其他节点：  
 $m_u^l = MSG^l(h_u^{l-1})$ ，这其中 $h_u^{l-1}$ 代表的是u结点认为自己的状态。MSG函数可以定义为线性函数，即是一个W矩阵。
- 消息聚合：**目的是**节点收集汇总来自其邻居的消息**， $h_v^l = AGG^l([m_u^l, u \in N(v)])$ ，可以使用sum函数，mean函数或者max函数来充当汇聚函数。但是这样进行**节点自身的信息被丢失**，因此一般把自己的信息也要经过一个新的矩阵计算一起当作输出。 $m_v^l = B^l h_v^{l-1} h_v^l = CONCAT(AGG^l([m_u^l, u \in N(v)]), m_v^l)$ 。

- 把所有上述细节综合起来，GNN层包括
  - 消息计算 (Message computation)
    - ✓ 每个节点计算一条消息

$$m_u^{(l)} = MS^{(l)}(h_u^{(l-1)}), u \in \{N(v) \cup v\}$$

- 消息聚合
  - ✓ 聚合来自邻居和自身的消息

$$h_v^{(l)} = A^{(l)}(\{m_u^{(l)}, u \in N(v)\}, m_v^{(l)})$$

- 非线性激活函数 $\sigma(\cdot)$ 
  - ✓ 可以加在消息计算或消息聚合中
  - ✓ 典型的函数：ReLU, ...

- GCN(卷积图神经网络):**

- GCN (Graph Convolutional Networks)

$$h_v^{(l)} = \sigma \left( W^{(l)} \sum_{u \in N(v) \cup v} \frac{h_u^{(l-1)}}{\sqrt{|N(u)||N(v)|}} \right)$$



- 以Message + Aggregation形式改写上述公式如下

$$h_v^{(l)} = \sigma \left( \underbrace{\sum_{u \in N(v)} \frac{W^{(l)} h_u^{(l-1)}}{\sqrt{|N(u)||N(v)|}}}_{\text{Message}} + \underbrace{W^{(l)} h_v^{(l-1)}}_{\text{Aggregation}} \right)$$

- GraphSAGE：**

- GraphSAGE

$$\mathbf{h}_v^{(l)} = \sigma \left( \mathbf{W}^{(l)} \text{CONCAT} \left( \mathbf{h}_v^{(l-1)}, \text{A} \left( \left\{ \mathbf{h}_u^{(l-1)}, u \in N(v) \right\} \right) \right) \right)$$

- 以Message + Aggregation形式改写上述公式如下

– 聚合分两步

- ✓ 第一步：聚合来自邻居的消息

$$\mathbf{h}_{N(v)}^{(l)} = \text{A} \left( \left\{ \mathbf{h}_u^{(l-1)}, u \in N(v) \right\} \right)$$

- ✓ 第二步：进一步聚合来自自身的消息

$$\mathbf{h}_v^{(l)} = \sigma \left( \mathbf{W}^{(l)} \text{CONCAT} \left( \mathbf{h}_v^{(l-1)}, \mathbf{h}_{N(v)}^{(l)} \right) \right)$$

•

- 每一层都应用L2正则化确保尺度一致性，即 $h_v^l = \frac{h_v^l}{||h_v^l||_2}$ ，确保嵌入向量在尺度上差异化不大。

- **GAT：**

- **不同邻居的消息对节点v的重要程度是不一样的**，因此使用attention代表注意力的强弱。

- Graph Attention Networks

– 关键思想：不同邻居的消息对节点v重要程度是不一样的

– 为此，引入代表重要性程度的 $\alpha_{vu}$ ，具体如下

$$\mathbf{h}_v^{(l)} = \sigma \left( \sum_{u \in N(v)} \alpha_{vu} \mathbf{W}^{(l)} \mathbf{h}_u^{(l-1)} \right)$$

- 实际上，GCN和GraphSAGE也做了类似的事情，直接用节点的度来定义 $\alpha_{vu}$

$$\alpha_{vu} = \frac{1}{|N(v)|}$$

- ✓ 不足：所有邻居的消息度节点v重要程度是一样的，这个显然不合理

- 我们将 $\alpha_{vu}$ 称作v对u的注意力， $e_{vu}$ 表示节点u的消息对节点v的重要性，a是计算注意力系数的函数。

- $e_{vu} = a(W^l h_u^{l-1}, W^l h_v^{l-1})$ ，描述二者的相似性，之后使用softmax函数，把 $\alpha_{vu}$ 归一化： $\alpha_{vu} = \frac{\exp(e_{vu})}{\sum_{k \in N(v)} \exp(e_{vk})}$

，同时也可以使用多头注意力机制

- **过平滑问题：**

- 堆叠太多的GNN层会带来过平滑问题，即**所有的节点嵌入表示收敛到相同的值**。这是由于过学习导致的，**感受域**过大导致几乎全部图都被节点表示，感受域重叠程度越高，其嵌入表示也会更加高度相似。我们需要先对图的直径进行分析，如果层数较少的话如何提高其表达能力？

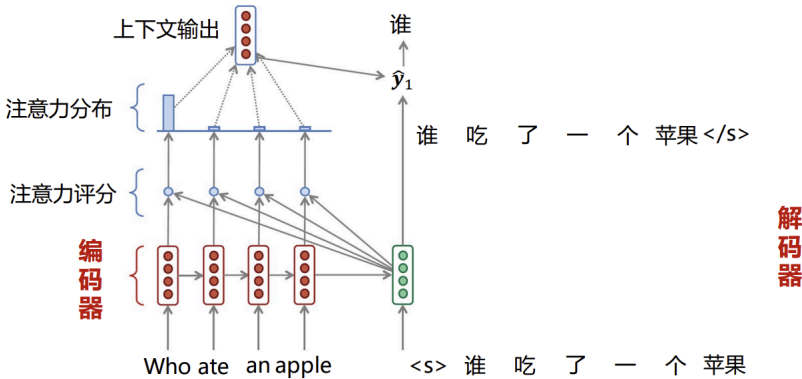
- 对于浅层的GNN，可以**增加聚合信息层**的复杂度，也可以**增加传递信息**的层。同样也可以**增加跳跃连接**，每一层GNN层都加入图的结构信息。（跳跃模型的本质是形成一个混合模型，N跳形成了 $2^N$ 个可能的路径，每一跳都有经过和不经过两种方案）

- **图神经网络的下游任务：**

- 节点、边、子图和图的预测任务，对于**节点级任务**，包括蛋白质折叠；对于**边级预测任务**，基于已有的边，预测新的、缺失的和未知的边，其应用包括基于图的推荐系统等，也可以预测药物蛋白质之间的相互作用；对于**图级预测任务**，目标是对整个图或者子图进行预测，可以用来预测药物，天气预报。

### Lecture12. Transformer

- 注意力是有选择地关注某些信息进而忽略其余信息的感知行为，Seq2seq应用广泛，其在自然语言处理的众多任务中取得了良好效果，但是本来编码器的**所有隐层都应该含有表达句子语义的信息**，由于RNN的时序特点，这种信息被丢失，每次只传入一个隐层的信息。因此需要编码器将全部的输入信息进行编码，这就是transformer制造的依据。



•

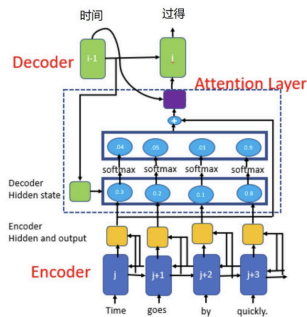
- 正如图所示，在对输入信息全部编码之后，每次给其一个token作为query，其会与所有已存在的输入信息计算点积，求出注意力评分采样得到词。解码器的第一个token一定是[s]，即start of the sentence。之后解码器的输入是上一个的输出。

- 注意力包括三个部分：首先是**注意力评分**， $s_{ti} = f_s(h_t^D, h_i^E)$ ，其次是**注意力权重分布**，即 $a_t = f_a(s_t)$ ，将刚才计算的注意力评分统计化归为分布。最后是**上下文**，即 $c_t = f_c(a_t, h_{1:n}^E)$ ，生成解码器在输入端的上下文，并且参与该步的预测。一般注意力评分采用向量内积，注意力权重分布采用softmax，上下文采用加权和。

- **B模型：**
  - Bahdanau Model
  - 编码器：双向RNN；解码器：单向RNN
  - 注意力
 
$$s_{ti} = f_s(h_{t-1}^D, h_i^E) = \text{MLP}(h_{t-1}^D, h_i^E)$$

$$a_{ti} = \frac{\exp(s_{ti})}{\sum_j \exp(s_{tj})}$$

$$c_t = \sum_{i=1}^n a_{ti} h_i^E$$
  - 解码器第t步隐层
 
$$-h_t^D = g(h_{t-1}^D, c_t, y_{t-1})$$



- 还有L模型，其编码器和解码器是栈LSTM。
- 但是seq2seq的模型在长距离依赖中表现不佳，而且只能串行执行，无法并行。
- 注意力机制建模的话需要位置编码，一般使用cosine, sine的方式编码。 $l_t[2j] = \sin(\frac{i}{10000^{2j/D}})$ ,  $l_t[2j+1] = \cos(\frac{i}{10000^{2j/D}})$ ,
- BERT：是一个基于 Transformer Encoder 的通用文本表示学习模型，擅长双向上下文建模，主要用于各类自然语言理解任务，而非文本生成。

## Lecture 13. 泛化理论

- 泛化理论
  - ✓ 泛化误差
  - ✓ 训练误差
  - ✓ 贝叶斯误差
  - ✓ 近似误差和估计误差
  - ✓ 正则化、结构风险最小化和经验风险最小化
  - ✓ 偏差和方差
- **经验风险最小化(Empirical Risk Minimization, ERM)：**
- 选定参数化模型，通过**最小化训练误差去拟合训练数据**。一切的假设都是未见到的测试数据和训练数据是独立同分布的。 $\theta^* = \operatorname{argmin}_{\theta} L_D(h_{\theta}) = \frac{1}{N} \sum_{n=1}^N f(h_{\theta}(x_n), y_n)$ 。
- **泛化误差**，假定训练数据服从分布 $p(x, y)$ ，则**泛化误差**定义为损失值的期望 $L(h_{\theta}) = E_{(x,y) \sim p} f(h_{\theta}(x), y)$ ，而**训练误差**被定义为 $L(h_{\theta}) = E_{(x,y) \sim D} f(h_{\theta}(x), y)$ ，其中 $D(x, y) = \frac{1}{N} \sum_{n=1}^N \delta(x - x_n) \delta(y - y_n)$ 。因此我们发现**训练误差和泛化误差的表达形式是不一样的**。
- **误差分析：**本课程为了简单起见，以二分类任务为例，以ERM框架进行分析。
- 训练误差是： $\hat{E}_D(h_{\theta}) = \frac{1}{N} \sum_{n=1}^N I(h_{\theta}(x_n) \neq y_n)$ ，即数据集中与预测结果不同的就是误差。泛化误差是 $E(h_{\theta}) = E_{(x,y) \sim p} I(h_{\theta}(x) \neq y)$ ，即在分布中取值与预测结果不同的就是误差。因为训练集和测试集是独立同分布，因此我们可以推出**训练误差的期望就是泛化误差**。

- 训练误差 $\hat{E}_D(h_k)$ 的期望 $\mathbb{E}[\hat{E}_D(h_k)]$

$$\begin{aligned} \mathbb{E}[\hat{E}_D(h_k)] &= \mathbb{E}\left[\frac{1}{N} \sum_{n=1}^N \mathbb{I}(h_k(x_n) \neq y_n)\right] \\ &= \frac{1}{N} \sum_{n=1}^N \mathbb{E}[\mathbb{I}(h_k(x_n) \neq y_n)] \\ &= \frac{1}{N} \sum_{n=1}^N \mathbb{E}[z_n] \\ &= \frac{1}{N} \sum_{n=1}^N \mathbb{E}[z] = \mathbb{E}[z] = \mathbb{E}_{(x,y) \sim p}[h_k(x) \neq y] = E(h_k) \end{aligned}$$

- 联合界定理 (The union bound)：令 $A_1, A_2, \dots, A_M$ 是M个事件，有

$$P(A_1 \cup A_2 \cup \dots \cup A_M) \leq \sum_{m=1}^M P(A_m)$$

- 霍夫丁不等式 (Hoeffding Inequality)：令 $z_1, z_2, \dots, z_N$ 是N个独立且服从伯努利分布 $\text{Ber}(\theta)$ 的随机变量，即 $p(z_n = 1) = \theta$ ，且 $E(z_n) = \theta$ 。记 $\hat{\theta} = \frac{1}{N} \sum_{n=1}^N z_n$ ，

则对任意的 $\varepsilon > 0$ ，有

$$P(|\theta - \hat{\theta}| > \varepsilon) \leq 2\exp(-2N\varepsilon^2)$$

- 根据霍夫丁不等式，当N足够大的时候，即**训练样本足够多**的时候，**训练误差将以很高的概率接近泛化误差**。 $P(|E(h_k) - \hat{E}_D(h_k)| \leq \varepsilon) \geq 1 - 2e^{-2N\varepsilon^2}$ 。但是这个局限在特定的假设 $h_k$ 上的，我们不希望其局限在特定的假设上，因此令 $A_k$ 表示 $|E(h_k) - \hat{E}_D(h_k)| > \varepsilon$ ，因此可以得出对于任何一个 $H = [h_1, h_2, \dots, h_K]$ ，泛化误差和训练误差的差值小于等于 $\varepsilon$ 的概率大于等于 $1 - 2Ke^{-2N\varepsilon^2}$ ，当N趋于正无穷的时候这个概率趋

近于1，即代表对**两分类的有限假设空间**而言，所有假设的训练误差一致地以很高的概率接近其泛化误差。

- $P(\forall h \in H, |E(h) - \hat{E}_D(h)| \leq \epsilon) \geq 1 - 2K \exp(-2N\epsilon^2)$ ，因此 **$\epsilon$ ，N和差值概率知二求一**。
- 给定 $\epsilon$ 和 $\delta$ ，可以求出至少需要N个训练样本才能以1- $\delta$ 的概率保证假设的训练误差与泛化误差的差值不超过 $\epsilon$ ， $N \geq \frac{1}{2\epsilon^2} \log \frac{2K}{\delta}$ 。这代表**训练样本越多，训练误差和泛化误差的差别越少**。
- 给定N和 $\delta$ ，可以得知训练误差和泛化误差的差值上界为多少的时候，能保证其概率至少为1 -  $\delta$ 。结果是 $\epsilon \geq \sqrt{\frac{1}{2N} \log \frac{2K}{\delta}}$ ，这说明N越大，概率差值越小。
- 通过**经验风险最小化**获得的假设的泛化性能：
- 令 $\hat{h} = \operatorname{argmin}_{h \in H} \hat{E}_D(h)$ ，其含义是训练误差最小的假设。 $h^* = \operatorname{argmin}_{h \in H} E(h)$ ，其含义是泛化误差最

• 定理1：给定假设空间 $\mathcal{H} = \{h_1, h_2, \dots, h_K\}$ ，训练样本数量N和 $\delta$ ，有

$$P\left(E(\hat{h}) \leq E(h^*) + 2\sqrt{\frac{1}{2N} \log \frac{2K}{\delta}}\right) \geq 1 - \delta$$

小的假设。我们有下图的结论：  
和泛化误差的期望的差值的最大值也会随着N的增大而降低。

- 推论1：给定假设空间 $\mathcal{H} = \{h_1, h_2, \dots, h_K\}$ ， $\epsilon$ 和 $\delta$ ，要以至少1 -  $\delta$ 概率成立 $E(\hat{h}) \leq E(h^*) + 2\epsilon$ ，则训练样本数量N需要满足

$$N \geq \frac{1}{2\epsilon^2} \log \frac{2K}{\delta} = O\left(\frac{1}{\epsilon^2} \log \frac{K}{\delta}\right)$$

- 对于**无限假设空间**，我们引入度量指标**VC维**，即令D是一个数据集，如果对其中的数据的数据的任何标注（打标签），假设空间H都能区分，则称H能打散D，**H的VC维就代表H能打散的最大数据集的数据数量**。
- H维空间中的超平面VC维为H+1，类比二维平面的VC维是3，4个物体则会产生二维不可分的问题。下面的VC维泛化界可以得知**VC维不利于泛化，其与数据量是反作用**，这对应了模型过于复杂的话就容易过拟合。

- 定理（VC维泛化界）：令假设空间 $\mathcal{H}$ 的VC维为H，其输出为布尔值。那么对任意 $\delta > 0$ ，对所有 $h \in \mathcal{H}$ 以至少1 -  $\delta$ 概率成立如下不等式

$$E(h) \leq \hat{E}_D(h) + \sqrt{\frac{2H}{N} \log \frac{eN}{H}} + \sqrt{\frac{\log \frac{1}{\delta}}{2N}}$$

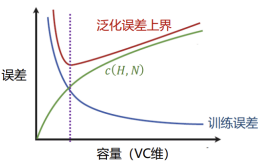
- **贝叶斯误差**：贝叶斯误差定义为**所有的可能假设中泛化误差最小值，是最优的泛化误差**，对假设没有任何限制，任意的假设都可以， $E_B = \operatorname{argmin}_h E(h)$ ，如果假设h'的泛化误差 $E(h') = E_B$ ，则称h为**贝叶斯假设**，贝叶斯假设是理论上最好的假设，但是是不可求解的。
- 给定一个假设h，h和贝叶斯误差之间的差可以分解如下 $E(h) - E(H^*) = (E(h) - E(h^*)) + (E(h^*) - E_B)$ ，其中 $h^*$ 是H中泛化误差最小的假设，前一项是**估计误差**，度量假设h与 $\mathcal{H}$ 中最好的假设在性能上的差异程度。用之前的不等式可以处理，后一项是**近似误差**，度量假设空间 $\mathcal{H}$ 能够逼近贝叶斯误差的程度，由于数据的真实分布是不可知的，因此**近似误差无法获取**。
- 我们的考虑目标还是改为希望**结构风险最小化**，上述的VC维泛化定理告诉我们，泛化误差在概率上有上界，上界包含**经验误差**（训练误差， $E_D(h)$ ）和**VC维相关项 c(H, N)**。

$$P\left(E(h) \leq \hat{E}_D(h) + c(H, N)\right) \leq 1 - \delta \quad c(H, N) = \sqrt{\frac{2H}{N} \log \frac{eN}{H}} + \sqrt{\frac{\log \frac{1}{\delta}}{2N}}$$

- 直接最小化经验误差是错误的，其没有考虑空间的容量。容量大，则经验误差肯定小，但是容量大也代表VC维大，因此 $c(H, N)$ 较大，会有较多的过拟合，因此需要平衡以达到小的泛化误差。（模型不能过于复杂，也不能过于简单）
- **结构风险最小化(SRM)**：

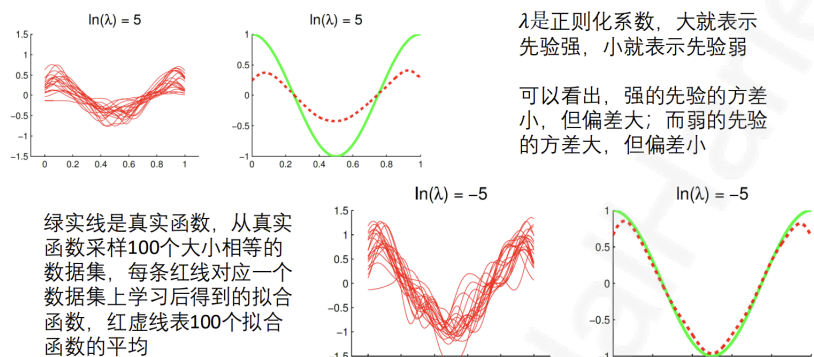
ITML 结构风险最小化

- Structural risk minimization (SRM)
- 一种同时兼顾模型容量和经验误差的学习方法
- 考虑一系列假设空间，按照容量（VC维）从小到大排列如下
  - $\mathcal{H}_1 \subset \mathcal{H}_2 \subset \dots \subset \mathcal{H}_i \subset \dots$ ，其中 $\mathcal{H}_i$ 的VC维是 $H_i$
- 步骤
  - 对每个 $\mathcal{H}_i$ ，利用经验风险最小化方法找到 $h_i^{ERM}$
  - 结构风险最小化选择的假设定义如下
$$h^{SRM} = \operatorname{argmin}_i (\hat{E}_D(h_i^{ERM}) + c(\mathcal{H}_i, N))$$
- 问题



- 理论可行，但计算量太大，而且VC维也没有通用的计算方法，实际可操作性差
- 兼顾模型容量和经验误差的学习方法，其核心思路是首先考虑**划分出一系列的假设空间**，将其按照容量从小到大排列，对于每个假设空间，先利用**经验风险最小化ERM**找到 $h_i^{ERM}$ ，之后每一层都**计算结构风险最小化**的值，取所有层中结构风险最小的值作为选择。 $h^{SRM} = \operatorname{argmin}_i (E_D(h_i^{ERM}) + c(H_i, N))$ ，但是这个方法实际难以操作。
- **正则化与模型选择**：

- 正则化方法是**结构风险最小化的替代方案**， $L_\lambda(h) = \hat{E}_D(h) + \lambda C(h)$ ，利用C来度量假设空间h的复杂性。  
 $h^{REG} = \operatorname{argmin}_{h \in H} (E_D(h) + \lambda C(h))$ ，这与上图SRM的表达式极其类似，在线性回归任务中的L2-正则化回归也是类似的思路，通过结构风险这个思想我们能够对正则化有更多的理解。
- 模型选择：**
- 我们可以假定有K个模型，对K个模型进行训练和预测，选择其中泛化性能最好的作为最终的输出值。但是我们需要得知如何估计所学到的预测函数的泛化误差。采用**交叉验证**的方式即可，把有标注的训练样本集D分成互不相交的两个子集 $D_{val}$ 和 $D_{train}$ ，训练集训练模型，验证集用来估计泛化误差。分割之后如果验证集的大小较小的话会导致估计不准确，如果验证集的大小较大的话会导致训练不充分。因此为了更好的估计泛化误差，我们可以采用**K折交叉验证**，将数据集分成K份，其中每次轮转K-1份用于训练，1份用于验证，轮转K次之后确认最优的当作选择的模型。
- 偏差和方差分析：**
- 除了偏差我们需要理解，方差也很重要，否则我们对分布的估计也容易不准确。我们假定训练数据服从概率分布 $p(x, y|\theta^*)$ ，D是从概率分布中采样出的包含N个数据的集合，假定我们从该概率分布中采样出了K个这样的数据集，进而估计出K个模型参数 $\theta$ ，因此这K个量可以被称作 $\hat{\theta}$ 的采样分布。
- 示例：我们认为 $x_n \sim N(\theta^*, \sigma^2)$ ，我们希望估计未知均值 $\theta^*$ ，对于MLE的估计而言，其零偏差，但是 $var(\hat{\theta}) = \frac{\sigma^2}{N}$ ，估计的 $MSE = (Bias)^2 + Var = \frac{\sigma^2}{N}$ ；对于MAP估计而言，其在MLE的估计之后使用 $\theta = w\hat{\theta} + (1-w)\theta_0$ ，因此 $bias = (1-w)(\theta_0 - \theta^*)$ ， $var = w^2 \frac{\sigma^2}{N}$ ，因此也可以计算出MSE。通告计算我们得知，先验均值距离真值较近，样本少噪音大，先验强度适中的时候MAP比MLE更强。
- 如果我们的优化目标是最小化估计量的均方误差，则可知MSE等于偏差加上方差。
- 我们希望估计是无偏的，但是仅有无偏是不够的，还希望偏差较小，通过推理我们可以得知，虽然最大后验估计是有偏的，但是其方差比最大似然估计低，就最小均方误差而言，通过选择恰当的先验，其均值方差也能小。



- 上图展示了方差和偏差的权衡，**正则化系数较大的时候先验强，其方差小但是偏差大，正则化系数较小的时候先验弱，其方差大但是偏差小。**

## Lecture14. Decision Tree

- 决策树**
  - ✓ID3算法
  - ✓分类回归树
  - ✓特征选择方法
- Lecture Include：ID3算法，分类回归树，剪枝与模型选择，随机森林
- 采用分治得到策略构造树，典型的逼近离散目标函数的方法，其优势是**可解释性极好，没有黑箱操作**，能够方便**融入专家知识和常识**，且决策速度极快。
- CART的效果更好，能够回归也能够分类，ID3效果稍微差一些，但是不容易过拟合，只能进行分类。目标是**生成最小的树**（奥卡姆剃刀原则，如无必要，勿增实体），学习一个最小的决策树是一个NP完全问题，因此一般使用贪心搜索的思想分治建造树。
- 首先构造一个空树，之后选择一个最好的属性（最好的属性代表有助于生成最小的树），对训练样本进行分割。最后递归进行。
- ID3算法：**

- 给定训练数据集 $\mathcal{D} = \{(x_n, y_n)\}_{n=1}^N$ ，其中
  - $x_n$  是由离散值构成的向量，其每个属性（维度）取值为离散值
  - $F = \{f_1, f_2, \dots, f_D\}$  表示属性集合，其中 $f_j$  表示第 $j$ 个属性
  - $y_n \in \{c_1, c_2, \dots, c_L\}$  为是 $x_n$ 的标签，其中 $c_k$ 代表第 $k$ 个类别

DTree( $\mathcal{D}, F$ ) returns a tree

If each  $(x, y) \in \mathcal{D}$ ,  $y = c_k$ , then return a leaf node with category label  $c_k$

Else if  $F = \emptyset$ , return a leaf node with the category label  $c_l$

$$c_l = \operatorname{argmax}_c |\{y = c | (x, y) \in \mathcal{D}\}|$$

Else **pick a feature**  $f \in F$  and create a node  $R$  for it

For each possible value  $v_j$  of  $f$

$\text{Example}_j = \{(x, y) | x.f = v_j, (x, y) \in \mathcal{D}\}$

Add an out-going edge  $E$  to node  $R$  labeled with the value  $v_j$

If  $\text{Examples}_j = \emptyset$ , then attach a leaf node to edge  $E$  labeled with the category label

$$c_l = \operatorname{argmax}_c |\{y = c | (x, y) \in \mathcal{D}\}|$$

Else

call DTree( $\text{Examples}_j, F - \{f\}$ ) and attach the resulting tree as the subtree under edge  $E$

Return the subtree rooted at  $R$

- 一开始选择一个信息增益最大的特征当作树根，之后取出这个特征的所有的值，每个值分配一个子节点，如果取值里标签一致，则用一个叶子节点连接收尾，如果没有对应的取值，则使用父节点的标签的值投票决定具体的标签。如果有对应的取值，且取值的标签不一致，则继续使用信息增益的方式递归构建树。
- 算法核心是找到最能**使得不确定性下降最多**的属性进行分割，语言形式类似：如果...那么...
- 递归调用函数，如果剩下的数据中所有元素的**标签一致**，则返回单个标签作为叶节点；如果**特征集是空集**，则用一个叶节点占据，叶节点的标签选择数据集D里有的最多的标签（投票决定）。
- 除此之外选择一个信息增益最大的特征，对特征的**每一个可能的取值都建立出一条边**，如果取值是空的，用叶子节点占位，叶子节点的标签选择父节点里最多的类别。如果取值不是空的，则递归调用构建树。
- 理想的属性是**希望树的规模最小**，尽可能选择“最纯节点”，使得节点下面的训练样本属于同一类。我们一般用熵来描述节点好坏。**信息增益IG**： $IG(Y|X) = H(Y) - H(Y|X)$ ，用熵和条件熵的差来描述，这个值就是信息论中的互信息，如果X和Y是独立的，则信息增益是0，即 $IG(Y|X) = 0$ 。
- 我们定义 $D_l^{f=v}$ 的含义是标签 $y = c_l$ ， $(x, y) \in D^{f=v}$ ，熵的定义是： $H(D) = -\sum_{l=1}^L p_l \log p_l$ ，**条件熵**则是对 $f = v$ 求期望进行平均 $H(D|f) = \sum_{v \in V(f)} p(f = v) H(D|f = v)$ 。

- 给定数据集 $\mathcal{D} = \{(x_n, y_n)\}_{n=1}^N$ ， $y_n \in \{c_1, c_2, \dots, c_L\}$ ，属性 $f$ ，其值域为 $V(f)$ 。记

$$- \mathcal{D}_l \triangleq \{(x, y) | (x, y) \in \mathcal{D} \wedge y = c_l\}$$

$$- \mathcal{D}^{f=v} \triangleq \{(x, y) | (x, y) \in \mathcal{D} \wedge x.f = v\}$$

$$- \mathcal{D}_l^{f=v} \triangleq \{(x, y) | (x, y) \in \mathcal{D}^{f=v} \wedge y = c_l\}$$

- $\mathcal{D}$ 的熵定义为

$$\mathbb{H}(\mathcal{D}) = -\sum_{l=1}^L p_l \log p_l \quad p_l = \frac{1}{N} |\mathcal{D}_l|$$

- 因此我们有： $IG(D|f) = H(D) - H(D|f)$ ，而IG就是我们进行特征划分的主要依据。
- **ID3算法示例：**

| 日期 | 天气 | 温度 | 湿度 | 风力 | 爬山 |
|----|----|----|----|----|----|
| 1  | 晴  | 热  | 高  | 弱  | 否  |
| 2  | 晴  | 热  | 高  | 强  | 否  |
| 3  | 阴  | 热  | 高  | 弱  | 是  |
| 4  | 雨  | 适中 | 高  | 弱  | 是  |
| 5  | 雨  | 冷  | 中  | 弱  | 是  |
| 6  | 雨  | 冷  | 中  | 强  | 否  |
| 7  | 阴  | 冷  | 中  | 强  | 是  |
| 8  | 晴  | 适中 | 中  | 弱  | 否  |
| 9  | 晴  | 冷  | 中  | 弱  | 是  |
| 10 | 雨  | 适中 | 中  | 弱  | 是  |
| 11 | 晴  | 适中 | 中  | 强  | 是  |
| 12 | 阴  | 适中 | 中  | 高  | 是  |
| 13 | 阴  | 热  | 中  | 弱  | 是  |
| 14 | 阴  | 冷  | 高  | 强  | 是  |
| 15 | 雨  | 适中 | 高  | 强  | 否  |

$$\mathbb{H}(\mathcal{D}) = 0.918$$

$$p(\text{天气} = \text{晴}) = \frac{1}{3} \quad p(\text{天气} = \text{阴}) = \frac{1}{3} \quad p(\text{天气} = \text{雨}) = \frac{1}{3}$$

$$\mathbb{H}(\mathcal{D}|\text{天气} = \text{晴}) = 0.971 \quad \mathbb{H}(\mathcal{D}|\text{天气} = \text{阴}) = 0 \quad \mathbb{H}(\mathcal{D}|\text{天气} = \text{雨}) = 0.971$$

$$\begin{aligned} \mathbb{H}(\mathcal{D}|\text{天气}) &= p(\text{天气} = \text{晴}) \mathbb{H}(\mathcal{D}|\text{天气} = \text{晴}) + p(\text{天气} = \text{阴}) \mathbb{H}(\mathcal{D}|\text{天气} = \text{阴}) + p(\text{天气} = \text{雨}) \mathbb{H}(\mathcal{D}|\text{天气} = \text{雨}) \\ &= \frac{1}{3} \times 0.971 + \frac{1}{3} \times 0 + \frac{1}{3} \times 0.971 = 0.647 \end{aligned}$$

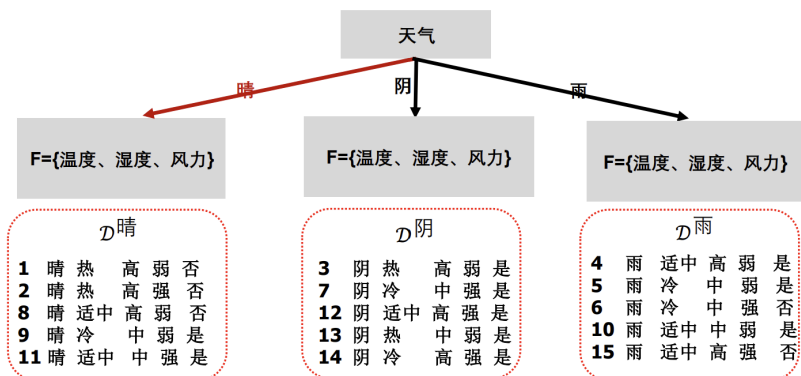
$$IG(\mathcal{D}|\text{天气}) = \mathbb{H}(\mathcal{D}) - \mathbb{H}(\mathcal{D}|\text{天气}) = 0.918 - 0.647 = 0.271$$

- 这个设定中 $x$ 代表日期、天气、温度、湿度和风力， $y$ 代表是否爬山，先计算**整个数据集上的熵** $H(D)$ ，这个熵是根据最后标签，即是否爬山来决定的。随后我们有天气、温度、湿度、风力这四个特征，我们需要确定先选择哪个特征，因此需要计算条件熵。对于“天气”这个特征，求解 $H(D|\text{天气} = \text{晴})$ 可以计算数据集中天气=晴的情况，在这个条件下有两个爬山和三个不爬山，因此通过这个即可计算出条件熵。据此可以计算出 $H(D|\text{天气} = \text{雨})$ 等三类。通过这三类计算出

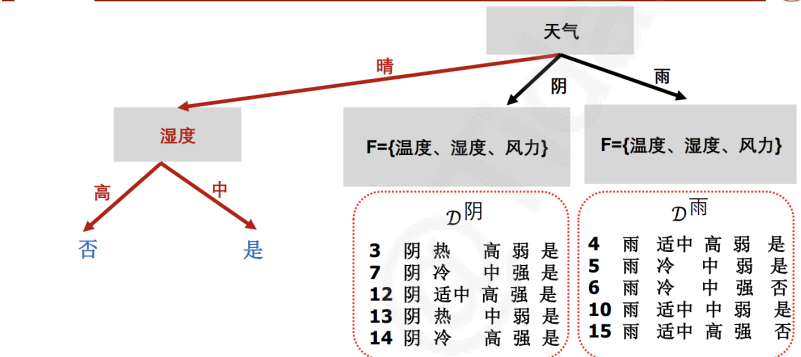
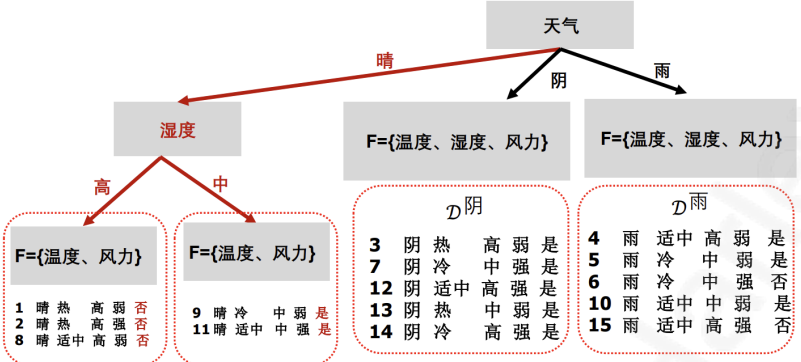
$$\begin{aligned} \mathbb{H}(\mathcal{D}) &= -\sum_{l=1}^L p_l \log p_l \\ &= -\frac{10}{15} \log \frac{10}{15} - \frac{5}{15} \log \frac{5}{15} \\ &\approx 0.918 \end{aligned}$$

- $H(D|\text{天气}) = p(\text{天气} = \text{晴})H(D|\text{天气} = \text{晴}) + p(\text{天气} = \text{阴})H(D|\text{天气} = \text{阴}) + p(\text{天气} = \text{雨})H(D|\text{天气} = \text{雨})$
- 进而计算出
- $IG(D|\text{天气}) = H(D) - H(D|\text{天气}) = 0.271$

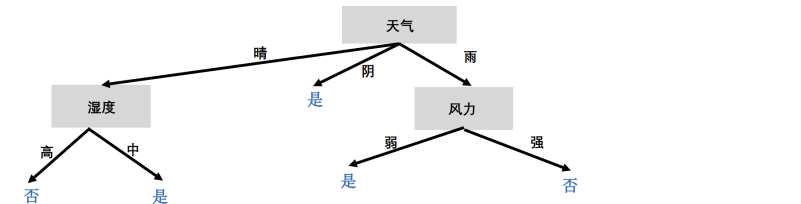
，对所有的“天气”，“温度”，“湿度”，“风力”计算IG之后发现**天气**标签下的IG值最大，因此第一次分类用天气。



- 随后需要在每个子树之中再次进行分类，分类方法不变，考虑的特征是温度、湿度和风力，计算熵和条件熵，最终找到下一个可分类的特征。经过计算我们得知是“湿度”。



- 此时按照湿度进行划分之后每一个特征值下面的特征空间里的标签一致，因此满足递归的结束条件，可以直接划归为**是**和**否**。



- 经过无限次重复计算就可以构建起一棵树，表示一种拟合和分割。ID3算法简单容易实现，**基于基本的分治递归**，但是也具有诸多问题。
- 其在特征选择熵信息收益偏好于取值多的属性，从而可能**导致过学习**的问题，而且**不能处理连续属性和回归任务**。而且信息增益会倾向于选择那些**选择取值多**的分支，（类似日期），因此会导致树的深度过浅，过拟合。
- 为此可以采用**增益率**的方式进行，计算的具体形式是 $SI(D|f) = -\sum_{v \in V(f)} p(f=v) \log p(f=v)$ ，增益率  $GR(D|f) = \frac{IG(D|f)}{SI(D|f)}$ ，比如说日期有15个不同的取值，每个取值只有一个，我们可以计算  $SI(D|\text{日期}) = -15 * \frac{1}{15} * \log \frac{1}{15} = 3.91$ ，用这个值做归一化就可以确保划分更合理。
- 对于**连续属性离散化**，可以采用**两分法**，对一个连续型属性A，构造一个布尔属性，对属性A的每一种可能的分割，评估其信息增益，选择信息增益最大的分割点为最好的分割点。只需要根据训练样本，在**训练样本不同的取值之间进行分割**即可，之后计算这些分割点中信息增益最大的点作为真正的分割点。

- **CART：**
- 分类回归树，既可以用于分类，也可以用于回归，既可以离散也可以连续。基本思想是假设决策树是**二叉树**，(ID3是多叉树) 内部节点是**特征取值**，递归二分每个特征，划分输入数据空间，在这些单元上分类回归预测。 $Gini(D) = \sum_{l=1}^L p_l(1 - p_l) = 1 - \sum_{l=1}^L p_l^2$ ，其是一种更均匀的熵。
  - 给定数据集 $\mathcal{D} = \{(x, y)\}$ ，如果属性 $f$ 是离散型，在属性 $f = a$ 的条件下， $\mathcal{D}$ 的基尼指数定义如下
$$Gini(\mathcal{D}|f = a) = \frac{|\mathcal{D}_{f=a}|}{|\mathcal{D}|} Gini(\mathcal{D}_{f=a}) + \frac{|\mathcal{D}_{f \neq a}|}{|\mathcal{D}|} Gini(\mathcal{D}_{f \neq a})$$
    - 其中
$$\mathcal{D}_{f=a} = \{(x, y) | (x, y) \in \mathcal{D}, x.f = a\}$$
$$\mathcal{D}_{f \neq a} = \{(x, y) | (x, y) \in \mathcal{D}, x.f \neq a\} = \mathcal{D} - \mathcal{D}_{f=a}$$
- 对于离散型数据，基尼系数的方式是类似全概率，用满足特征的基尼系数和不满足特征的基尼系数进行加权和计算出最后的结果。如果是对于连续型数据，则将其作为分割点，大于和小于分别计算。

• **CART示例：**

| 表1：爬山气候记录 |    |    |    |    |    |
|-----------|----|----|----|----|----|
| 日期        | 天气 | 温度 | 湿度 | 风力 | 爬山 |
| 1         | 晴  | 热  | 高  | 弱  | 否  |
| 2         | 晴  | 热  | 高  | 强  | 否  |
| 3         | 阴  | 热  | 高  | 弱  | 是  |
| 4         | 雨  | 适中 | 高  | 弱  | 是  |
| 5         | 雨  | 冷  | 中  | 弱  | 是  |
| 6         | 雨  | 冷  | 中  | 强  | 否  |
| 7         | 阴  | 冷  | 中  | 强  | 是  |
| 8         | 晴  | 适中 | 高  | 弱  | 否  |
| 9         | 晴  | 冷  | 中  | 弱  | 是  |
| 10        | 雨  | 适中 | 中  | 弱  | 是  |
| 11        | 晴  | 适中 | 中  | 强  | 是  |
| 12        | 阴  | 适中 | 高  | 强  | 是  |
| 13        | 阴  | 热  | 中  | 弱  | 是  |
| 14        | 阴  | 冷  | 高  | 强  | 是  |
| 15        | 雨  | 适中 | 高  | 强  | 否  |

$$Gini(\mathcal{D}_{\text{天气=晴}}) = 1 - \left(\frac{3}{5}\right)^2 - \left(\frac{2}{5}\right)^2 = \frac{12}{25}$$
$$Gini(\mathcal{D}_{\text{天气≠晴}}) = 1 - \left(\frac{2}{10}\right)^2 - \left(\frac{8}{10}\right)^2 = \frac{32}{100}$$
$$Gini(\mathcal{D}|\text{天气=晴}) = \frac{5}{15} Gini(\mathcal{D}_{\text{天气=晴}}) + \frac{10}{15} Gini(\mathcal{D}_{\text{天气≠晴}}) = \frac{12}{25}$$

- 最优切分点是**最小的Gini数**的位置，即 $a^* = \operatorname{argmin}_a Gini(\mathcal{D}|f = a)$ ，因此构建树的过程是如果到终止条件则用 $\operatorname{argmax}$ 最大数量的标签当作叶子节点，如果不到的话则找到最优切分点，如果满足最优切分点单独分成一类，如果不满足的话放到另一类中。
- 比如说对于“天气”这个特征来说， $Gini(\mathcal{D}_{\text{天气=晴}}) = 1 - (\frac{3}{5})^2 - (\frac{2}{5})^2$ ，同样的策略，找到符合要求的数数据，之后根据标签进行分类计算。同理可以计算出 $Gini(\mathcal{D}_{\text{天气≠晴}})$ 。随后计算**条件基尼系数**，即 $Gini(\mathcal{D}|\text{天气=晴}) = p(\text{天气=晴})Gini(\mathcal{D}_{\text{天气=晴}}) + p(\text{天气≠晴})Gini(\mathcal{D}_{\text{天气≠晴}})$ ，因此最后的结果可以计算。最优切分点就是所有的属性里面条件基尼系数最小的取值。
- 在确定了**最优切分点**之后直接将数据一分为二，之后分别递归计算。

ITML 分类树：算法描述

Input:  $\mathcal{D} = \{(x, y)\}$ ,  $y \in \{c_1, c_2, \dots, c_L\}$  is a label,  $F = \{f_1, f_2, \dots, f_K\}$  is the set of attributions

CART\_C( $\mathcal{D}$ ) returns a tree

If the **terminating condition** is satisfied,

Then return a leaf node with the category label  $c_l$

$c_l = \max_c |\{y = c | (x, y) \in \mathcal{D}\}|$

Else

For all  $f \in F$ , find  $(f^*, a^*) = \operatorname{argmin}_{a \in V(f), f \in F} Gini(\mathcal{D}|f = a)$

Create a node  $R$  for  $f^*$

$Example_{left} = \{(x, y) | x.f^* = a^*, (x, y) \in \mathcal{D}\}$  // 连续型  $f^* \leq a^*$

$Example_{right} = \{(x, y) | x.f^* \neq a^*, (x, y) \in \mathcal{D}\}$  // 连续型  $f^* > a^*$

For  $j \in \{left, right\}$

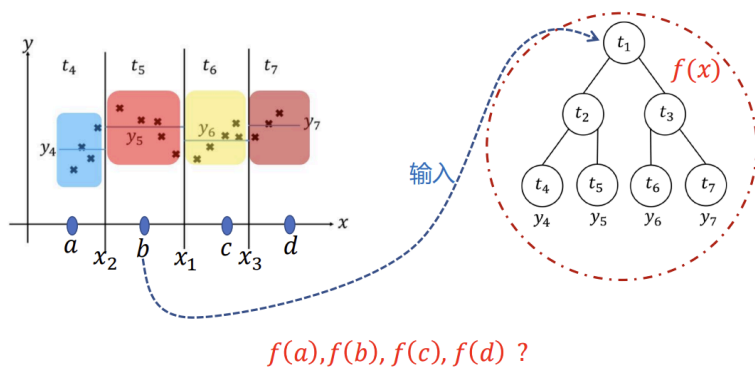
Add an out-going edge  $E$  to node  $R$  for  $Example_j$

Call CART\_C( $Example_j$ ) and attach the resulting tree as the subtree under edge  $E$

Return the subtree rooted at  $R$

思考：与ID3相比，分类树算法有什么优点

- **CART回归树：**
- 衡量方式不再是条件基尼系数，而是SE，核心目的是 $\operatorname{argmin}$  SE。
- $SE(f, a) = \sum_{x \in R_{left}} (y - o_{left})^2 + \sum_{x \in R_{right}} (y - o_{right})^2$ ，其中 $o_{left} = \operatorname{mean}(y \text{ in left})$ 。
- 对每个属性，取遍其在样本数据集中的所有取值的中间值作为切分点，(x=1, 2, 3, 4的时候可以考虑1.5, 2.5, 3.5作为潜在的切分点)，据此可以找到一个在平方误差指标下的最优切分点，对数据集进行二分，对分割后的两个子集重复上述步骤，不断二分。
- 比如说输入只有一个特征，其取值是{1, 2, 3, 4, 5}，输出的标签是{1, 2, 2, 4, 5}，则不分裂的时候y的均值是2.8，因此计算 $RSS = (1 - 2.8)^2 + (2 - 2.8)^2 + (2 - 2.8)^2 + (4 - 2.8)^2 + (5 - 2.8)^2 = 10.8$ ，枚举切分点为：{1.5, 2.5, 3.5, 4.5}，对于其中比如说a=2.5的切分点，左子集的y的均值是1.5， $RSS_L = (1 - 1.5)^2 + (2 - 1.5)^2 = 0.5$ ，同理，右子集的均值是3.67， $RSS_R = 4.67$ ，最终两者相加为RSS。注意，划分根据特征取值进行，但是计算RSS都是根据y的取值进行的。



- 在测试推理的时候， $f(a)=y_4$ ，看能落入哪个区间，用这个区间中的均值来描述点的取值。
- 剪枝**：决策树容易过学习，由于CART树是二叉树，CART树一般比ID3更深，因此需要剪枝，一个是**前剪枝**，另一个是**后剪枝**。
- 前剪枝**：如果样本量过少，则代表对树的划分没有统计意义，因此停止增长树，直接将其作为一个叶子节点。
- 后剪枝**：误差估计，利用统计学规律计算，求出估计误差。而一个子树根节点的预期误差指的是其所有子节点估计误差的加权和，加权系数是样本数量。如果对于一个节点而言，**估计误差小于预期误差**，则把子树替换成一个叶节点。

- 子树根节点R的**预期误差**是其所有子节点估计误差的加权和

$$E^* = \sum_{R_s \in \text{Son}(R)} \frac{|R_s|}{|R|} E(R_s)$$

子节点  $R_s$  下的样本数量      子节点  $R_s$  的估计误差

- 若子树根节点R**估计误差**小于其**预期误差**，则该子树替换成一个叶节点

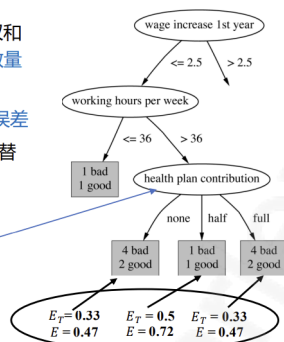
- 剪枝自底向上进行

- 例子： $z = 25\% \Rightarrow c = 0.69$

$$E_T = 5/14$$

$$\text{估计误差 } E = 0.46$$

$$E < 0.51 = E^*$$



- 因此，剪枝三个叶节点，该中间节点也变成叶节点 **预期误差  $E^*$  按照比率 6:2:6 合并为 0.51**

- 同样也可以使用**最小描述长度准则**对CART树进行剪枝，具体来说， $L_\lambda = L(T) + \lambda|T|$ ，其中  $L(T) = \sum_{i=1}^{|T|} N_i H(t_i)$ ，即对于每个叶子节点来看， $N_i$ 代表叶子节点的样本数，如果对于一个叶子节点来说，将自己和父节点合并之后的树的  $L_\lambda$  更小，则执行本次合并，如果合并之后的更大，则不执行本次合并。一直重复执行。

## ITML 最小描述长度准则

- 给定树  $T$ ，记  $|T|$  为树中叶节点的个数； $t_i$  表示  $T$  的叶子节点，该叶节点有  $N_i$  个样本，其中  $c_k$  类的样本有个  $N_i^k$  个，一共有  $K$  个类别。定义损失函数（loss function）如下

$$L_\lambda(T) = L(T) + \lambda|T| \quad L(T) = \sum_{i=1}^{|T|} N_i H(t_i) \quad H(t_i) = - \sum_{k=1}^K \frac{N_i^k}{N_i} \log \frac{N_i^k}{N_i}$$

- 其中

- ✓  $H(t_i)$  表示叶节点  $t_i$  的经验熵
- ✓  $L(T)$  表示该树对训练数据的预测误差，代表训练误差
- ✓  $|T|$  代表模型的复杂度
- ✓ 参数  $\lambda (\geq 0)$  控制模型训练误差和模型复杂度对学习结果的影响
  - 较大  $\lambda$  的倾向于选择简单的模型（树）
  - 较小  $\lambda$  的倾向于选择复杂的模型（树）
  - $\lambda = 0$  意味着只考虑与训练样本的拟合程度，不考虑模型复杂度

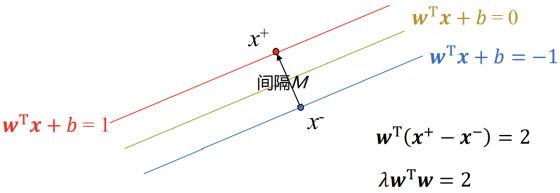
- 我们经过剪枝的方式能得到一系列不同超参数的树，之后需要进行K-fold cross validation，通过验证集来选择超参数  $\lambda$ 。
- 随机森林**：
- 重要的技术是**Bagging**，即从训练数据集中有放回地随机采样抽取  $N$  个训练数据，构建数据集  $D^*$ ，利用  $D^*$  构建一棵决策树，重复  $M$  次，获得的  $M$  棵决策树构成随机森林。其次重要的技术是**特征选择**，每当做特征选择的时候不选择所有的特征，只是从其中随机采样一个特征子集，在这个子集里面选当前最好的特征。

## Lecture15. SVM

- 支持向量机
  - ✓ 原问题和对偶问题
  - ✓ 支持向量
  - ✓ 核函数
- 线性可分问题**：存在决策超平面，使得  $w^T x + b = 1$ ， $w^T x + b = -1$ ，如果设PP和NP是两个超平面，那么  $x^+ = x^- + \lambda w$ ，我们记  $|x^+ - x^-| = M$ ，由已知信息， $w^T(x^+ - x^-) = 2$ ，代入之后  $\lambda w^T w = 2$ ，

$M = \frac{2}{\sqrt{w^T w}}$ ，最优决策平面的目的就是找到最大的间隔M。

- **支持向量**：指的是间隔边界两侧，距离分界线最近的点。也就是说，支持向量的这些点构成了分界线，在支持向量上的点满足： $y_i(w^T x_i + b) = 1$ 。
- **核函数**： $K(x, z) = \exp(-\gamma ||x - z||^2)$ ，即距离越近的两个点映射的影响值越大。目的是不显式升维，把数据映射到更高维上，在高维空间里用直线分。
- 基于约束平面（其含义是确保正确的x分在正确的标签y中）： $y_n(w^T x_n + b) \geq 1$ ， $\max M = \min \frac{w^T w}{2}$ ，



$$x^+ = x^- + \lambda w \Rightarrow x^+ - x^- = \lambda w$$
$$M = \|x^+ - x^-\| = \lambda \|w\| = \frac{2}{\sqrt{w^T w}}$$

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

约束条件:

$$g_i(x) \leq 0, \quad 1 \leq i \leq U$$
$$h_j(x) = 0, \quad 1 \leq j \leq V$$

- 带不等式约束的最优化问题：上述问题是原问题，解决带不等式约束的最优化问题时，常常利用拉格朗日对偶性将原问题转化为**对偶问题**。引入**广义拉格朗日函数**，考虑 $\theta(x) = \max L(x, \alpha, \beta) = f(x) + \sum_{i=1}^U \alpha_i g_i(x) + \sum_{j=1}^V \beta_j h_j(x)$ ，其中 $\alpha_i \geq 0$ ，因此如果其违反原本问题的约束条件，有 $\theta(x) = +\infty$ ，满足约束条件的时候 $\theta(x) = f(x)$ ，因此有 $\min_x \theta(x) = \min_x f(x)$ ，

- 首先，引进广义拉格朗日函数

$$L(x, \alpha, \beta) = f(x) + \sum_{i=1}^U \alpha_i g_i(x) + \sum_{j=1}^V \beta_j h_j(x)$$

- 考虑如下x的函数

$$\theta(x) = \max_{\alpha, \beta; \alpha_i \geq 0} L(x, \alpha, \beta)$$

- 对任何x，如果其违反原问题的约束条件，即：

$$\exists i, g_i(x) > 0 \text{ 或者 } \exists j, h_j(x) \neq 0$$

- 有

$$\theta(x) = \max_{\alpha, \beta; \alpha_i \geq 0} L(x, \alpha, \beta) = +\infty$$

- 因此在下列的约束条件下，上述的广义拉格朗日函数能够进行计算。因此最后的结果是如果x满足约束条件，则 $\theta(x) = f(x)$ 。
- 也就是说，利用广义拉格朗日函数将原问题转化为对偶问题，原本的 $\min f(x)$ 转化为 $\min \max L(x, \alpha, \beta)$ ，对于a和beta来说max的情况下找x的minimize值。

$$\min \frac{w^T w}{2}$$

约束条件:

$$1 - y_n(w^T x_n + b) \leq 0, 1 \leq n \leq N$$

拉格朗日函数:

$$L(w, b, \alpha) = \frac{1}{2} w^T w + \sum_{n=1}^N \alpha_n (1 - y_n(w^T x_n + b))$$

- 原问题的对偶问题是 $\max \min L(w, b, a)$ ，因此需要对w和b求极小值，之后再对a求极大值。因此先对w和b求偏导为0， $w = \sum_{n=1}^N \alpha_n y_n x_n$ ， $\sum_{n=1}^N \alpha_n y_n = 0$ 。代入到其中令导数为0。

$$L(w, b, \alpha) = \frac{1}{2} w^T w + \sum_{n=1}^N \alpha_n (1 - y_n(w^T x_n + b))$$

$$w = \sum_{n=1}^N \alpha_n y_n x_n \quad \sum_{n=1}^N \alpha_n y_n = 0$$

$$\min_{w, b} L(w, b, \alpha) = \frac{1}{2} \left( \sum_{i=1}^N \alpha_i y_i(x_i)^T \right) \left( \sum_{j=1}^N \alpha_j y_j x_j \right) + \sum_{i=1}^N \alpha_i \left( 1 - y_i \left( \sum_{j=1}^N \alpha_j y_j(x_j)^T x_i + b \right) \right)$$
$$= -\frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j y_i y_j (x_i)^T x_j + \sum_{i=1}^N \alpha_i$$
$$\min_{\alpha} \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j y_i y_j (x_i)^T x_j - \sum_{i=1}^N \alpha_i$$
$$\sum_{i=1}^N \alpha_i y_i = 0$$
$$\alpha_i \geq 0, \quad 1 \leq i \leq N$$

- 再对上面的最终表达式求max即可，这是一个典型的凸二次规划问题。以上都是理想的线性支持向量机，对于**有噪音的线性支持向量机**，使用**松弛变量**来体现违反可分条件的程度， $y_n(w^T x_n + b) \geq 1 - \eta_n$ ， $\min \frac{w^T w}{2} + C \sum_{n=1}^N (\eta_n)^k$ ，其中C和k是超参数。
- $\eta=0$ 代表数据点距离决策超平面的距离较大，可以被准确分类； $\eta$ 位于0-1之间代表可以被正确分类，但是分类的准确度不算高。 $\eta$ 大于等于1代表会被错误分类。

原问题

$$\min \frac{\mathbf{w}^T \mathbf{w}}{2} + C \sum_{n=1}^N (\xi_n)^k$$

$C(\geq 0)$ 和 $k(> 0)$ 是两超参数，用于控制错分的代价  
通常置 $k$ 等于1或2。 $k = 1$ 时称为hinge loss； $k = 2$ 时称为quadratic loss

估计偏离决策平面的损失，也就是错分的代价

约束条件:

$$y_n(\mathbf{w}^T \mathbf{x}_n + b) \geq 1 - \xi_n \Rightarrow 1 - \xi_n - y_n(\mathbf{w}^T \mathbf{x}_n + b) \leq 0, \quad 1 \leq n \leq N$$

$$\xi_n \geq 0 \Rightarrow -\xi_n \leq 0, \quad 1 \leq n \leq N$$

$\xi_n$  是最优化问题中的 “松弛变量” (slack variables)

如此也可以构建一个带有噪音的线性支持向量机。对于 $k=1$ 的情形，同样构建广义拉格朗日函数求偏导。通过对偶问题计算出取值，之后代入原问题。

$$\begin{aligned} & \frac{1}{2} \mathbf{w}^T \mathbf{w} + C \sum_{n=1}^N \xi_n + \sum_{n=1}^N \alpha_n (1 - \xi_n - y_n(\mathbf{w}^T \mathbf{x}_n + b)) + \sum_{n=1}^N \beta_n (-\xi_n) \\ &= \frac{1}{2} \mathbf{w}^T \mathbf{w} - \mathbf{w}^T \left( \sum_{n=1}^N \alpha_n y_n \mathbf{x}_n \right) - b \sum_{n=1}^N \alpha_n y_n + \sum_{n=1}^N \alpha_n + \sum_{n=1}^N \underbrace{(C - \alpha_n - \beta_n)}_{C - \alpha_n - \beta_n = 0} \xi_n \\ & \quad \mathbf{w} = \sum_{n=1}^N \alpha_n y_n \mathbf{x}_n \quad \sum_{n=1}^N \alpha_n y_n = 0 \\ &= -\frac{1}{2} \mathbf{w}^T \mathbf{w} + \sum_{n=1}^N \alpha_n = -\frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j y_i y_j (\mathbf{x}_i)^T \mathbf{x}_j + \sum_{i=1}^N \alpha_i \end{aligned}$$

推导过程是一致的，结果也是一致的，因此得出的结论是带噪音的SVM和理想的SVM求解方式是一致的。只是多增加了一个上界 $C = x_0$ 。 $\alpha_i \leq C$ 。

以上考虑的都是线性可分的问题，但是对于线性不可分的问题，需要通过映射的方式解决。尝试使用核函数的方式映射， $k(x_i, x_j) = \phi(x_i)^T \phi(x_j)$ ，因此优化问题改为：  
 $\min_{\alpha} \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j y_i y_j \phi(x_i)^T \phi(x_j) - \sum_{i=1}^N \alpha_i$ ，核函数是一种不需要显式知道变换维度就能够提高向量映射维度的方式。包括多项式核函数，径向核函数等。两个核函数的线性组合仍然是核函数。其实本质上都是线性可分的支持向量机。

| 线性 (带噪音)                                                                                                                              | 非线性                                                                                                                                             |
|---------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------|
| $\min_{\alpha} \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j y_i y_j (\mathbf{x}_i)^T \mathbf{x}_j - \sum_{i=1}^N \alpha_i$ | $\min_{\alpha} \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j) - \sum_{i=1}^N \alpha_i$           |
| $\sum_{i=1}^N \alpha_i y_i = 0$                                                                                                       | $\min_{\alpha} \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j y_i y_j \Phi(\mathbf{x}_i)^T \Phi(\mathbf{x}_j) - \sum_{i=1}^N \alpha_i$ |
| $\sum_{i=1}^N \alpha_i y_i = 0$                                                                                                       | $\sum_{i=1}^N \alpha_i y_i = 0$                                                                                                                 |
| $0 \leq \alpha_i \leq C, \quad 1 \leq i \leq N$                                                                                       | $0 \leq \alpha_i \leq C, \quad 1 \leq i \leq N$                                                                                                 |

对于一个示例来说。SVM 的决策方向 $\mathbf{w}$ ，不是任意的，而是由“支持向量”的线性组合决定的。  
 $f(x) = \sum_n \alpha_n y_n K(x_n, x) + b$ ，这是分类函数。进而可以求出 $b$ 。因此显示写出分类函数。

SVM的优点是训练相对简单，不存在局部最优问题，这一点不同于神经网络，而且可扩展性较好，能够较好处理高维度数据。但是缺点是选择一个好的核函数是一个大问题，其选择核函数是极其具有技巧的。

Lecture16. EM Algorithm

• EM算法

- ✓期望计算
- ✓最大化期望
- ✓应用

EM 算法用于在含有隐变量 $z$ 的模型中，通过交替估计 $p(z | x; \theta)$ 和更新参数 $\theta$ ，来最大化不完全数据似然 $p(x; \theta)$ 的MLE。

全局理解： $z$ 是一个标签，类似对于大量的数据而言一个更高度凝练的指标，比如说大量的数据是 $x$ ，则 $z$ 可能是形成数据的 $K$ 个高斯分布的分布信息，属于哪个高斯分布。

EM中 $E$ 的步骤是通过大量的数据求这个凝练指标 $z$ 的概率分布，即将大量数据分配给凝练指标。之后 $M$ 的步骤中先通过 $Q$ 函数，即假设上述概率分配方案正确，求期望，这个期望是假设 $z$ 的分配正确， $x$ 和 $z$ 的联合概率分布，由于 $x$ 已知，其可以简化为求 $z$ 的概率分布。 $Q$ 函数估计出假设正确的情况下 $\theta$ 的取值情况，因此最大化这个函数，就是下一个估计的参数。

- Initialize  $\theta^{(0)}$  //initial estimate of  $\theta$
- $m \leftarrow 0$
- While the stopping criterion is not met, do
  - Given  $x$  and assume that current guess  $\theta^{(m)}$  is correct, compute the conditional probability distribution  $p(z|x; \theta^{(m)})$  for the complete data  $z$
  - Using  $p(z|x; \theta^{(m)})$  obtained above, form the conditional expected log-likelihood, named  $Q$ -function

$$Q(\theta|\theta^{(m)}) = \int_Z \log p(z; \theta) \times p(z|x; \theta^{(m)}) dz = E_{z \sim p(z|x; \theta^{(m)})} \log p(z; \theta)$$

- Obtaining  $\theta^{(m+1)}$  by maximizing  $Q(\theta|\theta^{(m)})$ 

$$\theta^{(m+1)} = \operatorname{argmax}_{\theta} Q(\theta|\theta^{(m)})$$

- $m \leftarrow m + 1$

- **完全数据假设：**数据分为可见数据x和不可见数据y。可见数据就是采样到的数据点，不可见数据包括隐变量，比如说对于高斯混合模型GMM，每个点的坐标就是可见数据，而每个点属于具体哪个簇的是不可见的。 $z = (x, y)$ ，因此我们有马尔可夫关系，即 $p(x|z; \theta) = p(x|z)$ 。
- EM算法主要解决在**观测数据不完整**，或者模型中存在未观测的随机变量时，如何找到模型参数 $\theta$ 的最大似然估计的问题。完全数据变量是由N个独立同分布的变量组成的集合，每个观测数据都只依赖于一个特定的变量 $z$ ，因此可以把Q函数分解为和的形式。

- 定理1
  - 假定
 
$$p(Z; \theta) = \prod_{i=1}^N p(z^{(i)}; \theta)$$
  - 则
 
$$Q(\theta|\theta^{(m)}) = E_{Z \sim p(Z|X; \theta^{(m)})} \log p(Z; \theta) = \sum_{i=1}^N Q_i(\theta|\theta^{(m)})$$

$$Q_i(\theta|\theta^{(m)}) = E_{z_i \sim p(z_i|x_i, \theta^{(m)})} \log p(z_i; \theta)$$

- 按照上面的理论，定理1代表的就是z作为标签，其独立性的体现。如果**完整数据在样本维度上是独立的**，那么**Q函数可以按样本分解成求和形式**。
- EM算法要求“定义完全数据”，首先需要确保 $p(Z; \theta)$ 容易计算，因此可以为后续M步提供一个解析且易于优化的目标函数。其次 $x, Z, \theta$ 三者之间满足马尔可夫关系，给定 $Z$ ，参数 $\theta$ 与观测数据 $x$ 是无关的。这样易于证明之后的EM算法收敛性，也可以简化计算Q函数，即 $Q(\theta | \theta^{(m)}) = \sum_{i=1}^N Q_i(\theta | \theta^{(m)})$ 。
- **EM算法的收敛性分析：**
- 定理2的意义在于将不容易求出的p概率分布的目标函数转化为了易于最大化的Q目标函数。这个定理预示着：只要持续优化Q函数，最终的概率一定会得到优化。

- EM算法具有如下的单调递增性

$$\log p(x; \theta^{(m+1)}) \geq \log p(x; \theta^{(m)})$$

- 定理2
  - 令 $z$ 和 $x$ 是随机变量，其分布函数包含参数 $\theta$ ，且满足马尔科夫关系。如果 $Q(\theta|\theta^{(m)}) \geq Q(\theta^{(m)}|\theta^{(m)})$ ，则有

$$\log p(x; \theta) \geq \log p(x; \theta^{(m)})$$

- 因此说明了我们argmax Q函数的根本意义。根据定义： $\theta^{m+1} = \operatorname{argmax}_{\theta} Q(\theta|\theta^m)$ ，因此对于任意的 $\theta$ ，有 $Q(\theta^{m+1}|\theta^m) \geq Q(\theta|\theta^m)$ ，令 $\theta$ 取值为 $\theta^m$ ，代入之后根据定理2，有 $\log p(x; \theta^{m+1}) \geq \log p(x; \theta^m)$ 。因此证明了**优化收敛性**。
- EM算法直接增大似然函数，而梯度上升方法则是通过梯度上升的方式间接增大目标函数。

- **EM算法的应用：**

- **混合模型分解框架MD：**

- **把复杂的数据分布，看成是若干个简单分布按权重加在一起的“混合”**， 给定N个数据和K个聚类（一堆数据和一些高斯分布），认为数据从这些聚类中采样得到的，先根据 $P(C_k)$ 采样出一个聚类，再按照 $P(x_i|C_k)$ 采样出 $x_i$ ，即生成数据。
- 因此 $P(x_i) = \sum_{k=1}^K P(x_i, C_k) = \sum_{k=1}^K P(C_k)P(x_i|C_k)$ ，因此全部数据集X的生成概率是 $P(X) = \prod_{i=1}^N \sum_{k=1}^K p(C_k)P(x_i|C_k)$ ，我们希望观察数据集X的生成方式来推断出 $x_i$ 的生成分布，因此考虑MLE最大似然估计，在 $\sum_{k=1}^K P_k = 1$ 的情况下使用拉格朗日乘子法优化，解不出来。
- 因此只能定义Q函数，L的含义是所有聚类标签总共的集合。
 
$$Q(\theta|\theta^m) = E_{L|X; \theta^m} \log P(X, L; \theta) = \sum_{i=1}^N \sum_{k=1}^K P(C_k|x_i, \theta^m) (\log P(x_i; \theta_k) + \log P_k)$$
，这个式子就更容易拉格朗日乘子法了。在 $\sum_{k=1}^K P_k = 1$ 的约束下最终得到的结果是：

$$L(\theta, \lambda) = \sum_{i=1}^N \sum_{k=1}^K P(C_k | x_i; \theta^{(m)}) (\log P(x_i; \theta_k) + \log P_k) + \lambda \left( 1 - \sum_{k=1}^K P_k \right)$$

$$\frac{\partial}{\partial \theta_j} \mathcal{L}(\theta, \lambda) = 0$$



$$\sum_{i=1}^N P(C_j | x_i, \theta^{(m)}) \frac{\partial \log P(x_i; \theta_j)}{\partial \theta_j} = 0$$

与  $P(x; \theta_k)$  的具体形式有关，有可能有解析解，也有可能无解析解（通过其他近似方法求解）

- 总结一下全过程，初始化概率信息和参数信息，之后计算  $P(C_k | x_i, \theta^0)$ ，这就是EM算法中的E步，之后更新概率信息，并且通过梯度为0的方式相当于M步的argmax操作。循环迭代更新。

初始化  $\theta_s^{(0)}$  和  $P_s^{(0)}$  计算  $P(C_k | x_i, \theta^{(0)}) = \frac{P(x_i; \theta_k^{(0)}) P_k^{(0)}}{\sum_{j=1}^K P(x_i; \theta_j^{(0)}) P_j^{(0)}}$

$\theta_s^{(1)}$  和  $P_s^{(1)}$  计算  $P(C_k | x_i, \theta^{(1)}) = \frac{P(x_i; \theta_k^{(1)}) P_k^{(1)}}{\sum_{j=1}^K P(x_i; \theta_j^{(1)}) P_j^{(1)}}$

$P_s^{(1)} = \frac{1}{N} \sum_{i=1}^N P(C_s | x_i, \theta^{(0)})$  计算  $\sum_{i=1}^N P(C_j | x_i; \theta^{(0)}) \frac{\partial}{\partial \theta_j} \log P(x_i; \theta_j) = 0$

$\theta_s^{(2)}$  和  $P_s^{(2)}$  计算  $P(C_k | x_i, \theta^{(2)}) = \frac{P(x_i; \theta_k^{(2)}) P_k^{(2)}}{\sum_{j=1}^K P(x_i; \theta_j^{(2)}) P_j^{(2)}}$

$P_s^{(2)} = \frac{1}{N} \sum_{i=1}^N P(C_s | x_i, \theta^{(1)})$  计算  $\sum_{i=1}^N P(C_j | x_i; \theta^{(1)}) \frac{\partial}{\partial \theta_j} \log P(x_i; \theta_j) = 0$

- Choose initial estimates,  $\{\theta_s^{(0)}\}$  and  $\{P_s^{(0)}\}$

$t \leftarrow 0$

Repeat

- Compute  $P(C_k | x_i, \theta^{(t)}) = \frac{P(x_i; \theta_k^{(t)}) P_k^{(t)}}{\sum_{j=1}^K P(x_i; \theta_j^{(t)}) P_j^{(t)}}$

- Set  $\theta_s^{(t+1)}$  equal to the solution of the equation  $\sum_{i=1}^N P(C_s | x_i; \theta^{(t)}) \frac{\partial}{\partial \theta_s} \log P(x_i; \theta_s) = 0$

- Set  $P_s^{(t+1)} = \frac{1}{N} \sum_{i=1}^N P(C_s | x_i, \theta^{(t)})$

$t \leftarrow t + 1$

$\| \cdot \| < \varepsilon$

- Until termination condition, with respect to  $\theta$  and  $P$ , is achieved

## 高斯混合模型(GMM):

- 利用上述提及的EM算法，只不过将概率分布改为高斯分布信息。具体来说， $P(C_k) = \pi_k$ ， $P(x | C_k) = N(x; \mu_k, \Sigma_k)$ ，即将聚类信息分给数据的方式是高斯分布。一般我们令  $\Sigma_k = I$ 。同样地， $\log P(x) = \sum_{i=1}^N \log P(x_i) = \sum_{i=1}^N \log(\sum_{k=1}^K \pi_k N(x_i; \mu_k, I))$ ，对数下面有求和，无法用拉格朗日方法直接求解。

ITML

高斯混合模型：求解



混合模型分解算法

高斯混合模型

$$P(C_k | x_i, \theta^{(t)}) = \frac{P(x_i; \theta_k^{(t)}) P_k^{(t)}}{\sum_{j=1}^K P(x_i; \theta_j^{(t)}) P_j^{(t)}}$$

$$P(C_k | x_i, \mu^{(t)}) = \frac{N(x_i; \mu_k^{(t)}, 1) \pi_k^{(t)}}{\sum_{j=1}^K N(x_i; \mu_j^{(t)}, 1) \pi_j^{(t)}}$$

$$P_s^{(t+1)} = \frac{1}{N} \sum_{i=1}^N P(C_s | x_i, \theta^{(t)})$$

$$P(x_i; \theta_k) = \frac{N(x_i; \mu_k, 1)}{\theta_k} \xrightarrow{\theta_k \leftarrow \mu_k} \mu_k$$

$$\pi_s^{(t+1)} = \frac{1}{N} \sum_{i=1}^N P(C_s | x_i, \mu^{(t)})$$

$$\sum_{i=1}^N P(C_s | x_i; \theta^{(t)}) \frac{\partial \log P(x_i; \theta_s)}{\partial \theta_s} = 0$$

$$\sum_{i=1}^N P(C_s | x_i; \mu^{(t)}) \frac{\partial \log N(x_i; \mu_s, 1)}{\partial \mu_s} = 0$$

- 在高斯分布的化简下，参数的优化有闭式解，即  $\mu_s^{t+1} = \frac{\sum_{i=1}^N P(C_s | x_i; \theta^t) x_i}{\sum_{i=1}^N P(C_s | x_i; \theta^t)}$ ，

$x \in \mathbb{R}$  是可观测随机变量，采自由  $N(\mu_1, 1)$  和  $N(\mu_2, 1)$  构成的混合高斯模型，给定观测序列  $X = \{1, 6, 2, 2, 7\}$ ，用EM算法求  $\pi_i$  和  $\mu_i$

$$x_1 = 1, x_2 = 6, x_3 = 2, x_4 = 2, x_5 = 7$$

$$P(C_k | x_i, \mu^{(t)}) = \frac{N(x_i; \mu_k^{(t)}, 1) \pi_k^{(t)}}{\sum_{j=1}^2 N(x_i; \mu_j^{(t)}, 1) \pi_j^{(t)}} = \frac{\exp(-(x_i - \mu_k^{(t)})^2) \pi_k^{(t)}}{\sum_{j=1}^2 \exp(-(x_i - \mu_j^{(t)})^2) \pi_j^{(t)}}$$

$$\pi_s^{(t+1)} = \frac{1}{5} \sum_{i=1}^5 \frac{\exp(-(x_i - \mu_k^{(t)})^2) \pi_k^{(t)}}{\sum_{j=1}^2 \exp(-(x_i - \mu_j^{(t)})^2) \pi_j^{(t)}}$$

$$\mu_s^{(t+1)} = \frac{\sum_{i=1}^5 P(C_s | x_i; \mu^{(t)}) x_i}{\sum_{i=1}^5 P(C_s | x_i; \mu^{(t)})}$$

- 高斯混合模型GMM与K-means算法有很多相似性，将**GMM稍加修改即可变成K-means**。首先，对于一个GMM，将软分配改成one-hot的硬分配，即 $P(C_k|x) \in \{0, 1\}$ ，其次，所有的高斯协方差相同且为各向同性，即 $\sum_k = \sigma^2 I$ ，其中 $\sigma$ 是一个标量。最后，需要忽略混合系数 $\pi$ ，或者认为它们相等，即认为 $\pi_k = \frac{1}{K}$ 。
- GMM示例：
- 对于一组观测数据，已知 $x = \{1.0, 1.2, 0.8, 4.8, 5.2, 5.0\}$ ，我们已知隐变量 $C_i \in \{1, 2\}$ ，已知随机化的参数是 $P_1(0) = P_2(0) = 0.5$ ， $\mu_1(0) = 0.0$ ， $\mu_2(0) = 6.0$ ， $\sigma_1^2 = \sigma_2^2 = 1.0$ 。我们希望将六个数据分布到两个高斯分布之中，并且优化参数使得高斯分布最能够分隔这六个数据。
- 在E步之中，我们需要计算对于每个数据点 $x_i$ ， $P(C_k|x_i)$ 的取值，对于这个题目，最终的计算是 $6 * 2$ 的，正如下图所示。（计算过程是 $\mu=1$ 的概率除以 $\mu=1$ 的概率加上 $\mu=6$ 的概率）

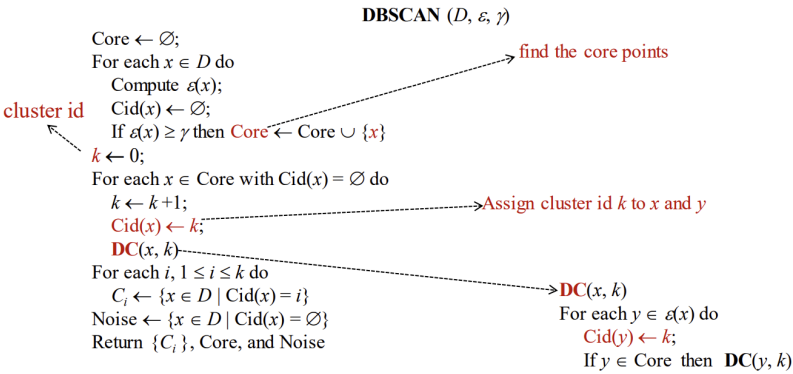
| $x_i$ | $\gamma_1=P(C_1 x_i)$ | $\gamma_2=P(C_2 x_i)$ |
|-------|-----------------------|-----------------------|
| 1.0   | 0.999                 | 0.001                 |
| 1.2   | 0.998                 | 0.002                 |
| 0.8   | 0.999                 | 0.001                 |
| 4.8   | 0.002                 | 0.998                 |
| 5.2   | 0.001                 | 0.999                 |
| 5.0   | 0.001                 | 0.999                 |

- 之后我们需要更新混合权重，即 $P_k^{(1)} = \frac{1}{N} \sum_{i=1}^N P(C_k|x_i)$ ，一组隐变量生成一个概率，最终的结果是 $P_1^{(1)} = P_2^{(1)} = 0.5$ 。随后我们需要更新均值，对于GMM而言，原本的求导简化为 $\mu_k^{(1)} = \frac{\sum_i \gamma_{ik} x_i}{\sum_i \gamma_{ik}}$ ，因此 $\mu_1 = \frac{1*0.999+1.2*0.998+0.8*0.999}{0.999+0.998+0.999} = 1.0$ ， $\mu_2 = 5.0$ 。更新参数之后再计算 $P(C_1|x = 1)$ 的结果就更接近1，同理 $P(C_2|x = 5)$ 也更接近1，这意味着分割更为准确。

Lecture17. 密度聚类与谱聚类

- 聚类
  - ✓密度聚类：DBSCAN算法的基本概念、算法内容
  - ✓谱聚类：算法的基本思想和框架
- 密度聚类：**
- 以数据密度为基础进行聚类，聚类内部数据稠密，聚类之间数据稀疏。
- DBSCAN算法：**
- $\epsilon$ 和 $\gamma$ 是两大超参数， $\epsilon$ 是邻居关系的最大半径， $\gamma$ 是以 $\epsilon$ 为半径的超球体中其他数据对象数量的最低阈值。给定这两个超参数，如果 $x$ 的 $\epsilon$ 邻居数量不少于 $\gamma$ ，即 $|\epsilon(x)| \geq \gamma$ 则称 $x$ 为**核心对象**。**直接密度可达**：称 $y$ 是从 $x$ 直接密度可达，如果 $x$ 是核心对象， $y$ 属于 $\epsilon(x)$ ，一连串直接密度可达可以推出**密度可达**，如果存在一个 $z$ ， $x$ 和 $y$ 都与 $z$ 密度可达，则 $x$ 和 $y$ **密度相连**。
- 因此密度相连的最大数据集构成一个聚类，不在聚类中的对象被称之为噪音，**边界点**不是核心对象，是**某个核心对象的 $\epsilon$ 邻居**。直接密度可达和密度可达都是非对称的，只有密度相连是对称的。
- $\gamma$ 的含义就是：邻居至少要有几个才算密； $\epsilon$ 的含义就是：多近才算邻居；**核心点**是自己+ $\epsilon$ 范围内的邻居数量大于等于 $\gamma$ 阈值，即这个点处于高密度的区域中心；**边界点**是自己邻居不够多，但是紧挨着某个核心点；噪声点是附近没有核心点，也不够密。
- 性质4：如果 $x$ 是核心对象， $D$ 是从 $x$ 密度可达的数据的集合，则 $D$ 是一个聚类
- 性质5：如果 $C$ 是一个聚类， $x$ 是 $C$ 中的核心对象，则 $C$ 等于从 $x$ 密度可达的对象的集合
- 结论1：聚类完全由其所含的核心对象决定

ITML DBSCAN：算法描述



- DBSCAN算法第一步是对每个点判断 $\epsilon$ 范围内有多少邻居，将其标记为核心点。之后从一个核心点开始，**找一个还没有分配簇编号的核心点**，给它一个**簇编号cluster id**，之后把它所有的邻居都拉进来，如果邻

居中还有核心点，则继续外扩。全部进行完毕之后剩下的就是噪音。

- 这个算法参数敏感，而且密度分布不均匀。

CDP算法：

- 基本思想是**找到聚类中心**，聚类中心的密度大，而且它的邻居的密度比他小，同时远离其他密度更大的数据点。
- 局部密度**：你周围有多挤；**局部距离**：离比你更挤的人有多远。局部密度的含义就是在你附近距离你距离小于截断距离 $d_c$ 的threshold的点有多少个；局部距离的含义是你到比你更密的那个最近的点有多远。
- 对每个点绘制二维图，横轴是局部密度，纵轴是局部距离，这两个维度都较大的就是**聚类中心**，如果看不出来的话将两个维度的值**相乘排序**即可得知。因此CDP算法就是先确定聚类中心，之后将其余的点降序密度排列，每个点都挂靠在离他最近的，密度比他高的点上。

ITML CDP：算法描述

CDP ( $X, d_c$ )  
For each  $x_i \in X$  do  
  Compute  $\alpha_i, \beta_i$  and  $\gamma_i$ ;  
 $\Gamma \leftarrow \text{sort the } \{\gamma_i\} \text{ in descending order};$   
 $Cindex \leftarrow \text{coreindex}\{\Gamma\};$   
 $k \leftarrow 1;$   
For  $i = 1$  to  $K$  do  
   $Cid(x_{Cindex[i]}) \leftarrow k;$   
   $k \leftarrow k + 1$   
 $L \leftarrow \text{sort the } \{\alpha_i\} - \{\alpha_i | j \in Cindex\} \text{ in descending order};$   
For  $i = 1$  to  $L.length$  do  
   $k \leftarrow \text{argmin}_{j: \alpha_j > \alpha_{L[i]}} d_{L[i], j};$   
   $Cid(x_{L[i]}) \leftarrow Cid(x_k)$   
For  $i = 1$  to  $Cindex$  do  
   $C_i \leftarrow \{x \in X | Cid(x) = i\}$   
Return  $\{C_i\}$

设置 $d_c$ ，使得每个数据对象的平均邻居个数约为数据对象总数的1-2%

获得前K个 $\gamma$ 值最大数据的序号（下标）。这些数据是聚类中心，用于形成初始聚类。具体方法可基于决策图部分

给聚类中心分配聚类号

对非聚类中心的其他数据按局部密度降序排序。 $L$ 中保留的是数据的序号（下标）

给序号为 $L[i]$ 的数据对象分配聚类号。令 $x_k$ 是 $\alpha$ 值比其大的所有数据中与其距离最小的数据对象。则该数据对象与 $x_k$ 属于同一类。

谱聚类：

- 基于图结构的聚类算法，通过最优化图割，获得对图顶点的聚类信息。首先基于带权无向图进行，定义带权无向图的**邻接矩阵(A)**，再定义**度矩阵(Δ)**，其含义是每一行的 $A_{ij}$ 化归到一个位置，形成一个对角矩阵。通过这两个定义得出**规范邻接矩阵M**，其含义是 $\Delta^{-1}A$ 。这个矩阵的每一行加和为1，因此1是规范邻接矩阵M的特征值。同理我们可以定义**图拉普拉斯矩阵L**，其含义是 $\Delta - A$ ，我们可以证明图拉普拉斯矩阵是**半正定矩阵**，因此其特征值非负且特征向量相互正交， $Lx$ 即可表示当前值与邻居的相似度，如果一个点和邻居相差很小，则 $Lx$ 很小，否则 $Lx$ 很大。
- 同理我们进一步可以定义**规范对称拉普拉斯矩阵**， $L^s = \Delta^{-\frac{1}{2}}L\Delta^{-\frac{1}{2}}$ ，这个矩阵也是半正定矩阵，特征值非负且特征向量相互正交。也有**规范非对称拉普拉斯矩阵**， $L^a = I - \Delta^{-1}A$ 。
- 对于一个带权无向图，A是邻接矩阵，S和T是其顶点集合，**割**的定义是这两个顶点集合上每个值的邻接矩阵值加和。**容量**的定义是 $vol(S) = c(S, V)$ ，即对于全部顶点的割集合的值。对于一个图，我们定义聚类为顶点集合，不重叠，每一类的顶点集合用N维的布尔向量表示，1代表顶点在，0代表顶点不在。

- 给定带权无向图 $G = (V, E)$ ，A是其邻接矩阵。令 $S, T (\subseteq V)$ 是顶点集合，它们的割（cut）如下

$$c(S, T) = \sum_{x_i \in S} \sum_{x_j \in T} A_{ij}$$

- S的容量(volume)如下

$$vol(S) = \sum_{x_i \in S} d_i = \sum_{x_i \in S} \sum_{x_j \in V} A_{ij} = c(S, V)$$

- 比如说对于 $S = \{1, 2\}, T = \{3, 4\}, V = \{1, 2, 3, 4\}$ 而言，割的含义就是一头在S，另一头在T的边，即 $A_{13}, A_{14}, A_{23}, A_{24}$ ，而容量的含义则是一头在S，另一头在V的边。
- 有一些结论： $c(V_k, V_k) = V_k^T A V_k$ ， $c(V_k, \hat{V}_k) = V_k^T L V_k$ ， $vol(V_k) = V_k^T \Delta V_k$ ，也很好理解， $V_k$ 是一个N维列向量，如果该点在此类之中则该维度是1，否则该维度是0。

Ratio Cut：

- $J_{RC}(V_k) = \sum_{k=1}^K \frac{c(V_k, \hat{V}_k)}{|V_k|}$ ，因此优化目标是希望**聚类之内和外部之间的相似度越小越好**， $\min_{V_k} J_{RC}(V_k)$ ，使得J值最小，这是NP-hard的，为方便对问题求解，我们需要放松对 $V_k$ 的约束， $V_k$ 是实值向量即可。 $u_k = \frac{V_k}{\|V_k\|}$ 。优化问题的直觉上变为：在所有互相正交、长度规范的K个方向里，找一组方向U让 $Tr(U^T L U)$ 最小。

- 因此，原问题相当于求解以下带约束的极值问题

$$\min_{\{u_k\}} \sum_{k=1}^K u_k^T L u_k$$

—满足以下约束

$$u_k^T u_k = 1, \quad 1 \leq k \leq K$$

- 带约束的极值问题采用拉格朗日乘子法，通过求解有 $u_i^T L u_i = \lambda_i$ ，假定L的特征值从小到大排列，只需要找到**K个最小的特征值**即可，最小的K个特征值的加和就是最终需要minimize的值。但是我们同样需要K个特征向量的信息来进行聚类，因此将**L=D-A的L的K个最小特征值对应的特征向量排成矩阵，每一行为**

一个新的值，行与行之间做聚类K-means。（排列的时候把特征向量一列一列排成矩阵，之后一行一行进行聚类）

Spectral Clustering based on Ratio Cut: SCRC (X, K)

- Compute the **similarity matrix**  $A \in \mathbb{R}^{N \times N}$  of  $X$
- Compute the graph laplacian matrice  $L$
- Solve the minimal  $K$  eigenvalues of  $L$  and their eigenvectors  $\{u_i\}$
- $U \leftarrow [u_1, \dots, u_K]$
- $y_i \leftarrow U_{i,:}$  // The  $j$ th row of  $U$
- Run k-means (or other clustering algorithms) on  $\{y_i\}$  to obtain  $\{C_k\}$
- If  $y_i \in C_k$ , then  $x_i \in C_k$

Normalized Cut:

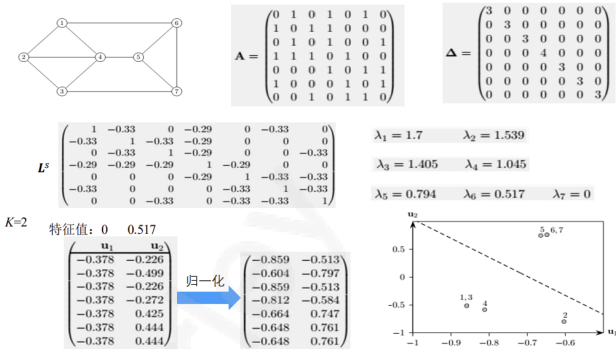
$J_{NC}(V_k) = \sum_{k=1}^K \frac{c(V_k, \hat{V}_k)}{vol(V_k)} = \sum_{k=1}^K \frac{V_k^T L V_k}{V_k^T \Delta V_k}$ ，显然分子越小，聚类与外部相似度越低，分母越大，容量越大，对顶点的分割越好。但是minimize这个值仍然极为困难，优化方法同Radio Cut。

因此，原问题相当于求解以下带约束的极值问题

$$\min_{\{u_k\}} \sum_{k=1}^K u_k^T L^s u_k \quad u_k = \frac{\Delta^{\frac{1}{2}} V_k}{\|\Delta^{\frac{1}{2}} V_k\|}$$
  
- 满足以下约束  
$$u_k^T u_k = 1, \quad 1 \leq k \leq K$$

后续聚类处理方法如Ratio Cut

- 矩阵 $[u_1, \dots, u_K]$ 每行需要做归一化处理



Average Cut:

$J_{AC}(V_k) = \sum_{k=1}^K \frac{c(V_k, V_k)}{|V_k|} = \sum_{k=1}^K \frac{V_k^T A V_k}{V_k^T V_k}$ ，其分子的含义是聚类内部的相似度，因此maximize是优化策略。同理换元 $u_k = \frac{V_k}{|V_k|}$ ，maximize特征值，但是只能找前K个最大的非负特征值，如果非负的特征值没有K个，则取所有的非负特征值即可。

Min-Max Cut:

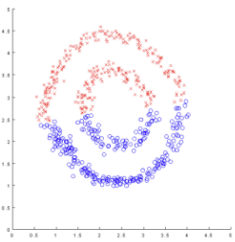
$J_{MMC}(V_k) = \sum_{k=1}^K \frac{c(V_k, \hat{V}_k)}{c(V_k, V_k)} = \sum_{k=1}^K \frac{V_k^T L V_k}{V_k^T A V_k}$ ，分子衡量的是聚类与外部的相似度，分母衡量的是聚类与内部的相似度，因此最小化J值即合理（最小化聚类与外界相似度，最大化聚类内部的相似度）。下图所示K是一个常量，可以舍弃。

$$J_{MMC}(\{V_k\}) = \sum_{k=1}^K \frac{V_k^T L V_k}{V_k^T A V_k} = \sum_{k=1}^K \frac{V_k^T (L - A) V_k}{V_k^T A V_k}$$
  
$$= \sum_{k=1}^K \frac{V_k^T \Delta V_k}{V_k^T A V_k} - K = \sum_{k=1}^K \left( \frac{V_k^T \Delta V_k}{V_k^T A V_k} \right)^{-1} - K$$
  
$$= \sum_{k=1}^K (u_k^T A^* u_k)^{-1} - K \quad u_k = \frac{\Delta^{\frac{1}{2}} V_k}{\|\Delta^{\frac{1}{2}} V_k\|} \quad A^* = \Delta^{-\frac{1}{2}} A \Delta^{-\frac{1}{2}}$$
  
Subject to  
$$u_k^T u_k = 1, \quad 1 \leq k \leq K$$
  
$$u_k^T A^* u_k > 0, \quad 1 \leq k \leq K$$

因此改为求解A 中前K个最大特征值，且其和大于0，随后处理同Normalized Cut中带约束优化情形。  
 $A^* = \Delta^{-\frac{1}{2}} A \Delta^{-\frac{1}{2}}$ 。

最后，谱聚类比K-means等聚类方法强在其可以对非凸区域进行聚类。

k-means (2个聚类)



谱聚类 (2个聚类)

