

题目

- [第一题](#)
- [第二题](#)
- [第三题](#)
- [第四题](#)
- [第五题](#)
- [第六题](#)
- [第七题](#)
- [第八题](#)
- [第九题](#)
- [第十题](#)

题目

第一题

题目：建立系统用于识别垃圾邮件和非垃圾邮件

解答：考虑使用朴素贝叶斯 Naive-Bayes 模型。按照 8: 2 划分训练集和测试集，并尽可能保证两个集合中（正常邮件）和（垃圾邮件）样本个数大致相当，即样本是均衡的。首先，需要对训练集邮件内容数据进行预处理，我们去除文本中的空格并将文本进行分词（tokenize）和装袋（BOW）处理。一般来说我们会在预处理阶段去掉文本中的标点符号和特殊符号（停词，stop words），但是基于垃圾邮件中可能存在更多的标点符号和特殊符号的观点，我们也可以进行保留。

在对训练集完成分词和DF（Document Frequency）统计后，我们可以选取最常出现的 $m = 500$ 个关键词 ($t_i, i = 1, \dots, m$) 作为描述文本的变量。其次，对特定文本的规范化表示为： $\mathbf{x} = (x_1, \dots, x_m) \in \mathbb{R}^m$ 其中 $x_i \in \{0, 1\}$ 表示第 i 个关键词 t_i 在该文本中是否出现，或者更一般的，也可以让 x_i 为第 i 个关键词的TF（Term Frequency）。为便于理解，我们采取前者定义。

我们假设分类指标 $c \in \{c_s, c_h\}$ 其中 c_s 表示垃圾邮件， c_h 表示为正常邮件，根据贝叶斯定理如果文本 d 的分词表示为 \mathbf{x} 那么，

$$p(c|\mathbf{x}) = \frac{p(c) \cdot p(\mathbf{x}|c)}{p(\mathbf{x})} = \frac{p(c) \cdot p(\mathbf{x}|c)}{p(c_s) \cdot p(\mathbf{x}|c_s) + p(c_h) \cdot p(\mathbf{x}|c_h)} \quad (1)$$

其中 $p(c|\mathbf{x})$ 表示在给定分词表示为 \mathbf{x} 下文本 d 被分类为 c 的概率。一般的，我们可以取先验分布 $p(c_s) = p(c_h) = 0.5$ 并给定阈值 $T = 0.5$ 即 $p(c_s|\mathbf{x}) > 0.5$ 时将其划分为垃圾邮件。所以接下来还需要设计 $p(\mathbf{x}|c)$ 。我们采取 Bernoulli NB 的方式进行定义，假设文本的分词表示 \mathbf{x} 如前定义，那么定义

$$p(\mathbf{x}|c) = \prod_{i=1}^m p(t_i|c)^{x_i} (1 - p(t_i|c))^{1-x_i} \quad (2)$$

其中 $p(t_i|c) = \frac{1 + M_{t,c}}{2 + M_c}$ ， $M_{t,c}$ 表示在训练集分类为 c 的所有文档中出现关键词 t 的文档个数， M_c 表示训练集中分类为 c 的总文档个数。常数的设计主要是为了防止在测试集中出现未曾遇见的新分词时导致分母为零。当然这里的常数也是可以依据分类指标 c 改动的，比如我们可以认为垃圾邮件中更常出现奇怪的新词。因此，结合上式，我们可以得到

$$p(c|\mathbf{x}) = \frac{\prod_{i=1}^m p(t_i|c)^{x_i} (1 - p(t_i|c))^{1-x_i}}{\sum_{c \in \{c_s, c_h\}} \prod_{i=1}^m p(t_i|c)^{x_i} (1 - p(t_i|c))^{1-x_i}} \quad (3)$$

分类准则为 $p(c_s|\mathbf{x}) > T$ 或者等效的，可以用计算量更少的 $\frac{p(c_s|\mathbf{x})}{p(c_h|\mathbf{x})} > \tilde{T}$ 作为划分标准。通过调节 T 观察算法在测试集上的表现，我们可以进一步改进算法。

Reference:

1. https://www2.aueb.gr/users/ion/docs/ceas2006_paper.pdf

第二题

题目：深证B股指数有大约50多家样本股的价格加权平均而得，请通过一段时间的历史数据，挑出尽量少的样本股精确估计该指数。

解答：可以考虑使用 LASSO 模型，假设 T 时间段股指内所有样本股的数据集合为 $X \in \mathbb{R}^{n \times T}$ ，股指历史走势为 $Y \in \mathbb{R}^T$ ；那么考虑模型：

$$Y = X\beta + \epsilon, \text{s.t } \min_{\beta} \left\{ \frac{1}{2} \|Y - X\beta\|_2^2 + \lambda \|\beta\|_1 \right\} \quad (4)$$

其中 $\beta \in \mathbb{R}^T$ 即是我们待求的权重， $\|\cdot\|_2$ 和 $\|\cdot\|_1$ 分别表示 \mathbb{R}^T 空间中的2范数和1范数。由于 LASSO 采用了如上的 $L-1$ 正则化，因此模型会根据超参数 λ 的选择自动过滤和筛选变量，从而达到使用尽量少的样本股精确估计该指数的目的。 λ 的选择可以根据 AIC, BIC 等一般准则也可以查看 Solution Path 选择稳定的超参值。优化方法可以采取近端梯度下降 (Proximal Gradient Descent, PGD) 或者交替方向乘子法 (Alternating Direction Method of Multipliers, ADMM)。

第三题

题目：多神经元对刺激方向的响应模型如下。假设每个神经元的条件发放率 $f_a(s)$ (每秒发放次数) 表示如下：

$$f_a(s) = \exp\left(-\frac{(s - s_a)^2}{2\sigma_a^2}\right) \quad (5)$$

其中 $a = 1, \dots, N$ 是神经元编号， s 是受到外部刺激的方向角度 ($[0, 180^\circ]$)， s_a 是神经元 a 的偏好方向， $\sigma_a > 0$ 是神经元调谐函数的参数。假设

- 神经元动作电位发放之间是统计独立的；
- 神经元动作电位发放是一个泊松过程；

假设贝叶斯损失为平方差： $\text{loss} = (s - \hat{s})^2$ ， \hat{s} 是估计方向角度。

通过观测这 N 个神经元的在 T 长度时间段的发放率，建立对外部刺激方向 s 的最小方差的无偏 (MVUB) 估计，并计算其平方误差。

解答：由于我们假设神经元动作电位发放是一个泊松过程，所以有

$$P(S_a = k) = \frac{(\lambda T)^k}{k!} \exp(-\lambda T) = \frac{(f_a(s)T)^k}{k!} \exp(-f_a(s)T) \quad (6)$$

这里 S_a 是 $[0, T]$ 时间段内观测到神经元 a 的点位发放总次数。我们假设 r_a 是观测到神经元 a 的平均发放率，那么

$$p(r_a | s) = \frac{(f_a(s)T)^{r_a T}}{r_a T!} \exp(-f_a(s)T) \quad (7)$$

记 $\mathbf{r} = [r_1, \dots, r_N]$ 则根据独立性，有联合分布

$$p(\mathbf{r}|s) = \prod_{a=1}^N \frac{(f_a(s)T)^{r_a T}}{r_a T!} \exp(-f_a(s)T) \quad (8)$$

根据贝叶斯法则 $p(s|\mathbf{r}) = \frac{p(\mathbf{r}|s)p(s)}{p(\mathbf{r})} \propto p(\mathbf{r}|s)p(s)$ 不妨假设先验分布 $p(s) \sim U(0, 180)$ 均匀分布。

则根据贝叶斯推断性质，当损失定义为平方差损失时，外部刺激方向 s 的最小方差的无偏 (MVUB) 估计为：

$$\begin{aligned} L(s, \hat{s}) &= (s - \hat{s})^T (s - \hat{s}) \Rightarrow \int (s - \hat{s})^T (s - \hat{s}) p(s|\mathbf{r}) ds \\ \hat{s} &= \int s \cdot p(s|\mathbf{r}) ds \\ &= C \int s \cdot p(\mathbf{r}|s) \cdot p(s) ds \\ &= C \int s \cdot p(\mathbf{r}|s) ds \\ &= E[s|\mathbf{r}] \end{aligned} \quad (9)$$

无偏性可以由重期望公式保证；其平方误差为 $\text{loss} = (s - \hat{s})^2 = (s - E[s|\mathbf{r}])^2$

可以根据 $C - R$ 不等式得到其估计的下界为 $1/E_s[I_F(s)]$ ，这里 $I_F(s)$ 为 Fisher 信息量

$$I_F(s) = T \sum_{a=1}^N \frac{(s - s_a)^2}{\sigma_a^4} \exp\left(-\frac{(s - s_a)^2}{2\sigma_a^2}\right) = T \sum_{a=1}^T \frac{f'_a(s)^2}{f_a(s)} \quad (10)$$

Reference:

1. <https://www.amcs.upenn.edu/sites/default/files/Optimal%20Neural%20Tuning%20Curves%20for%20Arbitrary%20Stimulus%20Distributions.pdf>
2. https://www.cmor-faculty.rice.edu/~caam415/M1_Notes_Population_coding.pdf
3. <https://courses.cs.washington.edu/courses/cse528/13sp/lecture-slides/Lecture5.pdf>

第四题

题目 利用 Hopfield 神经网络实现一个 TSP 问题的求解，并进行数值验证。

解答：Hopfield 网络的原理是通过建立神经元之间的连接来实现模式识别。它的网络结构通常由一个正方形矩阵来表示，每个神经元代表一个单元格，单元格之间的连接权重则表示神经元之间的相互影响关系。如下图是一个 3 个城市 TSP 问题的 Hopfield 网络：

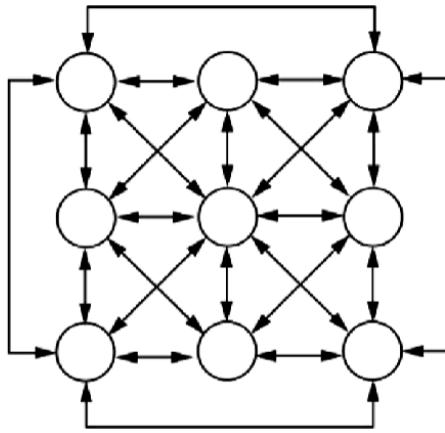


Figure 1: Hopfield NN for 3-cities TSP

在Hopfield网络中，每个神经元都有一个输入和一个输出。输入通常由外部信息提供，每个神经元都会根据输入来计算出自己的输出值。输出值通常以激活或抑制为两种状态，即每个神经元都会根据自己的输入来决定是否激活。

Hopfield网络是一种动态网络，每个神经元都会根据自己的输入来更新自己的输出值，然后再根据输出值来更新自己的输入值，以此循环进行直到所有的神经元都达到稳定状态。通过这种方式，Hopfield网络可以找到一组最优解，即使输入信息中存在噪声和冲突。

对于包含 N 个城市的 TSP 问题，我们定义我们的结果表示和输出矩阵（nodes Matrix）为一个 $N \times N$ 的矩阵 V ，输入矩阵为一个 $N \times N$ 的矩阵 U 。其中如果 $V_{ij} = 1$ 则表示第 j 步到达城市 i 并从此出发前往下一步（第 $j + 1$ 步）。当最后达到收敛状态时， V_{ij} 一定非0即1，但在初始化时，我们采用：

$$V_{X,i} = g(U_{X,i}) = \frac{1}{1 + \exp(-2U_{X,i}/u_0)} = \frac{1}{2}(1 + \tanh(U_{X,i}/u_0)) \quad (11)$$

$$U_{X,i}^0 = -\frac{u_0}{2} \ln(N - 1)$$

我们假设权重是一致不变的。其中 u_0 是初始给定的值， $U_{X,i}$ 和 $V_{X,i}$ 分别表示对应矩阵的第 X (X 代表城市) 行第 i 列。

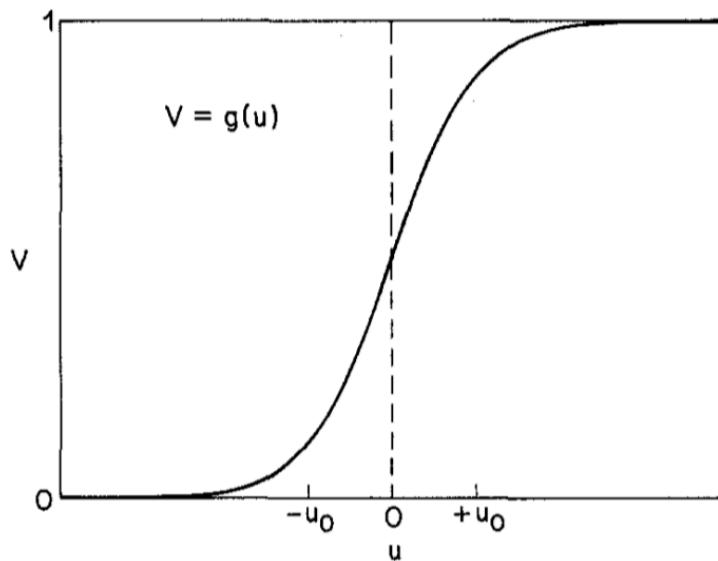


Figure 2: The input-output relations for neurons

当算法最终收敛时，矩阵 V 应该满足：

1. 每行有且只有一个 1
2. 每列有且只有一个 1
3. V 中有且只有 N 个 1
4. 得到的结果是TSP问题的解，即最小化路程

Hopfield网络每一个时刻的状态都可以用一个能量来表示：我们定义为

$$2E = A \sum_{X \in N} \sum_{i \in N} \sum_{j \neq i} V_{X,i} V_{X,j} + B \sum_{i \in N} \sum_{X \in N} \sum_{Y \neq X} V_{X,i} V_{X,j} \\ + C \left(\sum_{X \in N} \sum_{i \in N} V_{X,i} - N \right)^2 + D \sum_{X \in N} \sum_{Y \neq X} \sum_{i \in N} d_{X,Y} V_{X,i} (V_{Y,i+1} + V_{Y,i-1}) \quad (12)$$

还是同样的， X, Y 表示城市， i, j 是步的编号。 A, B, C, D 是四个超参，它们所对应的四个约束与我们之前矩阵 V 应该满足的限制条件是一一对应的。 $d_{X,Y}$ 是城市 X, Y 之间的距离。由于上述表达式过于复杂为方便起见，在实际计算时我们可以简化参数为：

$$2E = A \sum_{X \in N} \left(\sum_{i \in N} V_{X,i} - 1 \right)^2 + A \sum_{i \in N} \left(\sum_{X \in N} V_{X,i} - 1 \right)^2 + D \sum_{X \in N} \sum_{Y \in N} V_{X,i} d_{X,Y} V_{Y,i+1} \quad (13)$$

根据上述能量的表达式，我们可以优化Hopfield网络的动态输入为：

$$\frac{dU_{X,i}}{dt} = -\frac{\partial E}{\partial V_{X,i}} = - \left(A \left(\sum_{X \in N} V_{X,i} - 1 \right) + A \left(\sum_{i \in N} V_{X,i} - 1 \right) + D \sum_{Y \in N} d_{X,Y} V_{Y,i+1} \right) \quad (14)$$

$$U_{X,i}(t+1) = U_{X,i}(t) + \frac{dU_{X,i}}{dt} \Delta_t$$

这里 t 是迭代指标， Δ_t 表示步长（超参）。事实上，我们不必计算能量 E 而只需要在每一步完成迭代格式即可。所以，我们可以将整个迭代过程总结如下：

1. 初始化超参
2. 计算距离矩阵 $d_{X,Y}$
3. 初始化网络输出和输入状态 V 和 U
4. 利用动态方程计算输入状态增量：

$$\frac{dU_{X,i}}{dt} = - \left(A \left(\sum_{X \in N} V_{X,i} - 1 \right) + A \left(\sum_{i \in N} V_{X,i} - 1 \right) + D \sum_{Y \in N} d_{X,Y} V_{Y,i+1} \right)$$

5. 更新下一个迭代中的输入状态：

$$U_{X,i}(t+1) = U_{X,i}(t) + \frac{dU_{X,i}}{dt} \Delta_t$$

6. 更新下一个迭代中的输出状态：

$$V_{X,i}(t+1) = \frac{1}{2} (1 + \tanh(U_{X,i}(t+1)/u_0))$$

7. 判断当前 V 是否已经达到最终稳定状态（符合TSP矩阵规则）

Reference:

1. <https://towardsdatascience.com/hopfield-networks-are-useless-heres-why-you-should-learn-them-f0930ebedc>
2. <https://www.jianshu.com/p/30fdc835d5b3>

第五题

题目 利用Reinforcement Learning算法求解迷宫最短路径搜索问题。要求：任意给出终点位置，规划出最短路径或者告知不能到达。最好给出交互式的界面。

解答：我们采用 **Q-learning** 算法来解决问题，核心在于维护 Q-table

我们给出一个完整的算法推导。定义 t 时刻的状态和动作空间 ($1 \leq t \leq T$) :

	意义
\mathcal{A}_t	t 时刻的所有可选 Action 集合 $[A_t \in \mathcal{A}_t]$
\mathcal{S}_t	t 时刻的状态 (State) 空间 $[X_t \in \mathcal{S}_t]$
u_t	$\mathbb{R}^{\mathcal{S}_t}$ 空间中的一个函数，表示 utility (reward)
H_t	到达 t 时的已知信息 $\{X_1, A_1, \dots, A_{t-1}, X_t\}$
Y_t	$Y_t = u_t(X_t)$

\mathcal{H}_t 是由 H_t 张成的乘积 Sigma 代数空间。更进一步的，定义 $P_t : \mathcal{S}_t \times \mathcal{H}_t \times \mathcal{A}_t \rightarrow [0, 1]$ 表示在当前时刻的状态转移函数； $\gamma \in (0, 1)$ 是一个衰退系数，以此来保证最终的累积收益收敛。

于是，我们可以定义我们的策略为 $\pi := \{\pi_t\}_{t=1}^T$ 其中 π_t 是从 \mathcal{H}_t 到 \mathcal{A}_t 的所有测度构成的空间的一个映射。我们可以得到策略为 π 下的 V 值的定义为：

$$V^\pi(x) := E^\pi \left[\sum_{t=1}^T \gamma^{t-1} Y_t | X_1 = x \right] \quad (15)$$

也即是采取当前策略，从状态 x 出发所能得到的累积收益的期望（ γ 满足定义的情况下， $T = \infty$ 也无妨）。如果我们将初始状态也看作一个分布 $v(x)$ 那么我们可以定义 $V(\pi) = \int V^\pi(x) dv(x)$ 。所以自然的，我们的目标就是要：

$$\pi^* := \arg \max_{\pi \in \Pi} V(\pi) \quad (16)$$

但是当问题体现出时间非齐的时候，推导非常繁琐。注意到在我们的迷宫问题中，所有时刻的转移概率都是一致的，并且我们也可以约定 $\mathcal{S}_t, \mathcal{A}_t, u_t$ 也不随时间发生变化，因此下标 t 可以被省略。为了解决上述问题，我们定义：

$$Q(x, a) = E^\pi \left(\sum_{t=1}^T \gamma^{t-1} Y_t | X_1 = x, A_1 = a \right) \quad (17)$$

定义算子 B 为：

$$(Bf)(x, a) = E^\pi \left(Y_t + \gamma \max_{a' \in \mathcal{A}} f(X_{t+1}, a') | X_t = x, A_t = a \right) \quad (18)$$

注意这里的 X_{t+1} 是一个随机变量，其分布满足 $P(x, H_t, a')$ 。于是就有：

$$\begin{aligned} Q^*(x, a) &:= Q^{\pi^*}(x, a) \\ &= E^{\pi^*} \left(Y_1 + \gamma \max_{a' \in \mathcal{A}} Q^*(X_1, a') | X_1 = x, A_1 = a_1 \right) \\ &= (BQ^*)(x, a) \end{aligned} \quad (19)$$

上面的等式被称为 Bellman optimality equation 并且 Q^* 是这个等式的不动点。对于状态和行动空间都十分有限的问题，我们可以用如下方式迭代求解以获得 Q^* ：

$$Q_{k+1}(X_t, A_t) = Q_k(X_t, A_t) + \alpha \left\{ Y_t + \gamma \max_{a' \in \mathcal{A}} Q_k(X_{t+1}, a') - Q_k(X_t, A_t) \right\} \quad (20)$$

这是一个贪婪算法，其中 $\alpha \leq 1$ 是学习率。在迷宫问题中 \mathcal{S} 是迷宫中所有可到达点的集合， \mathcal{A} 是上下左右四个行动。为了加速进程，在实际实现中，则采用了 ϵ -Greedy 算法。

Reference:

1. <https://samyzaf.com/ML/rl/qmaze.html>
2. https://web.archive.org/web/20200424193039id_/https://www.annualreviews.org/doi/pdf/10.1146/annurev-statistics-031219-041220

第六题

题目 叙述并推导Resnet型神经网络的BP监督学习算法。分别推导以下神经网络层的随机梯度公式包含：

1. 卷积层（以Relu函数作为激发函数）
2. Max Pooling层和Normalization层、残差层
3. 全连接层（以Softmax作为激发函数）以交叉熵加上参数L2的regularization作为惩罚函数

解答：Resnet 的提出是因为作者发现56层的卷积神经网络的表现反而不如 20层的。但作者认为问题并不是梯度爆炸/梯度消失（这应该由 batch normalization完成）或者是过拟合问题（与深层CNN的测试误差表现不符）。他认为是因为深层CNN引入了太多的非线性因素而让“什么都不做的”恒同映射几乎不可能被学习到。换句话说，让神经网络的输出拟合恒同映射 $H(x) = x$ 是困难的。但如果我们将一个角度，直接引入恒同映射而让网络去学习 $F(x) = H(x) - x$ （残差）则会比直接拟合恒同映射容易的多。一个基本的残差层结构如下图所示：

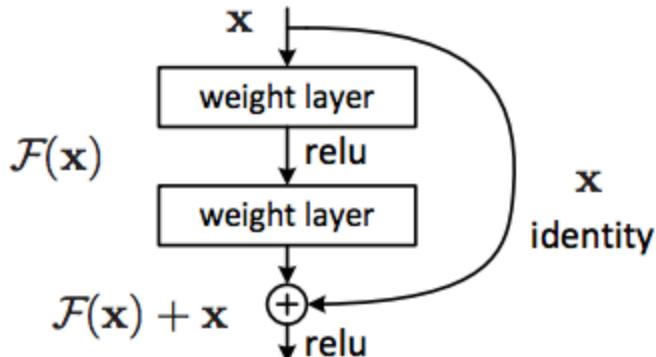


Figure 2. Residual learning: a building block.

这里，右侧的曲线是跳接（shortcut connection），通过跳接在最后的激活函数之前将前两层的输出直接与本层的输出 ($F(x)$) 相加 $y = F(x, \{W_i\}) + x$ ，然后将求和结果输入到激活函数中作为最后的输出 $e = R(y)$ 。我们约定，

$$ReLU(x) := R(x) = \max(x, 0) \Rightarrow \frac{\partial R}{\partial x} = \mathbb{1}_{\{x \geq 0\}} \quad (21)$$

其中， W_i 是残差层结构中第 i 层的权重参数， $\mathbb{1}_{\{x \geq 0\}}$ 与 x 维度一致（一种mask表示）。

1. 方便起见，我们假设最简单的情形， $x \in \mathbb{R}^{n \times m}$ 维矩阵且 $F = W_2 R(W_1 x)$ 。那么， $W_1 \in \mathbb{R}^{n \times n}$ ， $W_2 \in \mathbb{R}^{n \times n}$ 我们容易推导：约定矩阵元素的脚标从 0 开始

$$\frac{\partial e}{\partial x_{uv}} = \sum_{i=0}^{n-1} \sum_{j=0}^{m-1} \mathbb{1}_{\{y_{ij} \geq 0\}} \frac{\partial y_{ij}}{\partial x_{uv}} \frac{\partial e}{\partial y_{ij}} \cdot \frac{\partial y_{ij}}{\partial x_{uv}} = \sum_{i=0}^{n-1} \sum_{j=0}^{m-1} \mathbb{1}_{\{y_{ij} \geq 0\}} \frac{\partial y_{ij}}{\partial x_{uv}} \quad (22)$$

注意到根据定义有 ($w_{ij}^{(i)}$ 表示 W_i 中的元素)

$$y_{ij} = \sum_{k=0}^{n-1} w_{ik}^{(2)} R(W_1 X)_{kj} = \sum_{k=0}^{n-1} w_{ik}^{(2)} \max(0, \sum_{l=0}^n w_{kl}^{(1)} x_{lj})$$

$$\frac{\partial y_{ij}}{\partial x_{uv}} = \begin{cases} \sum_{k=0}^{n-1} w_{ik}^{(2)} w_{ku}^{(1)} \mathbb{1}_{\{\sum_{l=0}^n w_{kl}^{(1)} x_{lj} \geq 0\}} & , \text{if } j = v \\ 0 & , \text{otherwise} \end{cases} \quad (23)$$

结合上述两式，即可求得残差层的随机梯度公式；

对于梯度更新，我们让 L 为残差层的损失函数，则

$$\frac{\partial L}{\partial w_{uv}^{(l)}} = \sum_{i,j} \frac{\partial L}{\partial e_{ij}} \frac{\partial e_{ij}}{\partial w_{uv}^{(l)}}, l = 1, 2 \quad (24)$$

求得梯度后，我们可以采用牛顿法或者拟牛顿法更新参数 W_l ($l = 1, 2$) 直至收敛。

2. 卷积层（以Relu函数作为激发函数）的SGD公式为

假设 $X \in \mathbb{R}^{m \times n}$ 卷积核为 $W \in \mathbb{R}^{p \times q}$ ，步长为 $t \geq 1$ 时，卷积后的结果为：下面约定矩阵元素的脚标从 0 开始

$$y_{ij} = \sum_{r=0}^{p-1} \sum_{s=0}^{q-1} x_{it+r, jt+s} w_{r,s} + b, i \leq (m-p)/t, j \leq (n-q)/t \quad (25)$$

于是， $e = R(y) = \mathbb{1}_{\{y \geq 0\}} \in \mathbb{R}^{g \times h}$ 其中 g 是 $(m-p)/t$ 取 floor， h 是 $(n-q)/t$ 取 floor。

$$\frac{\partial e}{\partial x_{uv}} = \sum_{i=0}^{g-1} \sum_{j=0}^{h-1} \frac{\partial e}{\partial y_{ij}} \cdot \frac{\partial y_{ij}}{\partial x_{uv}} \quad (26)$$

注意到，当 $it + r = u, jt + s = r$ 根据 y_{ij} 表达式有，

$$\frac{\partial y_{ij}}{\partial x_{uv}} = \begin{cases} w_{u-it, v-jt} = w_{r,s} & \text{if } it \leq u \leq it + p - 1, jt \leq v \leq jt + q - 1 \\ 0 & \text{otherwise} \end{cases} \quad (27)$$

当卷积区域出现重叠时，i.e. $1 \leq t \leq p-1$ or $1 \leq t \leq q-1$

且当 $it \leq u \leq it + p - 1, jt \leq v \leq jt + q - 1$ 时

$$\frac{\partial e}{\partial x_{uv}} = \begin{cases} (\sum_{x=0}^{p-t} \mathbb{1}_{\{y_{i-xt,j} > 0\}}) w_{u-it, v-jt}, & 1 \leq t \leq p-1, t \geq q \\ (\sum_{y=0}^{q-t} \mathbb{1}_{\{y_{i,j-yt} > 0\}}) w_{u-it, v-jt}, & t \geq p, 1 \leq t \leq q-1 \\ (\sum_{x=0}^{p-t} \sum_{y=0}^{q-t} \mathbb{1}_{\{y_{i-xt,j-yt} > 0\}}) w_{u-it, v-jt}, & 1 \leq t \leq p-1, 1 \leq t \leq q-1 \end{cases} \quad (28)$$

其他情况我们默认SGD导数为0

3. Max Pooling层和Normalization层、残差层、全连接层（以Softmax作为激发函数）。我们约定，

$$\text{Softmax}(Y) := S(Y) = \left(\sum_{k=0}^{n-1} \frac{e^{y_{ik}}}{1 + e^{y_{ik}}} \right)_{m \times n}, Y \in \mathbb{R}^{m \times n} \quad (29)$$

则容易得到，

$$\frac{\partial S}{\partial y_{ij}} = \sum_{k \neq j} \frac{e^{y_{ik}}}{1 + e^{y_{ik}}} + \frac{e^{y_{ij}}}{(1 + e^{-y_{ij}})^2} \quad (30)$$

- Max Pooling 层：我们约定 $e = S(\text{Max Pooling}(X)) := S(Y) \in \mathbb{R}^{g \times h}$ ；

假设 $X \in \mathbb{R}^{m \times n}$ 核为 $W \in \mathbb{R}^{p \times q}$, 步长为 $t \geq 1$ 时

我们记 $K = (k_{uv})_{m \times n}$ 表示在每一次Max Pooling核最大化的操作中 x_{ij} 为最大值的次数。

$$y_{ij} = \max_{0 \leq r \leq p-1, 0 \leq s \leq q-1} (x_{it+r, jt+s}) \quad (31)$$

注意到，当 $it + r = u, jt + s = r$ 根据 y_{ij} 表达式有

$$\frac{\partial y_{ij}}{\partial x_{uv}} = \frac{1}{\sum_{u,v} k_{uv}} k_{uv} \mathbb{1}_{\{y_{ij}=x_{uv}\}} \quad (32)$$

此时应满足条件 $it \leq u \leq it + p - 1$ 以及 $jt \leq u \leq jt + q - 1$, 否则上述导数为0。

注意当池化区域不出现重叠时，我们可以有化简结果： $\frac{\partial y_{ij}}{\partial x_{uv}} = \mathbb{1}_{\{y_{ij}=x_{uv}\}}$

$$\Rightarrow \frac{\partial S}{\partial x_{ij}} = \sum_{i=0}^{g-1} \sum_{j=0}^{h-1} \frac{\partial S}{\partial y_{ij}} \cdot \frac{\partial y_{ij}}{\partial x_{ij}} \quad (33)$$

- Normalization层 【Ioffe, Sergey and Christian Szegedy. "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift." *ArXiv* abs/1502.03167 (2015): n. pag.】

Normalize 步骤：假设 mini-batch 是 $\mathcal{B} = \{x_1, \dots, x_m\}$, ϵ 是一个小的常数，防止分母为0

$$\begin{aligned} \mu_{\mathcal{B}} &= \frac{1}{m} \sum_{i=1}^m x_i, \sigma_{\mathcal{B}}^2 = \frac{1}{m} \sum_{i=1}^m (x_i - \mu_{\mathcal{B}})^2 \\ \hat{x}_i &= \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}}, y_i = \gamma \hat{x}_i + \beta \end{aligned} \quad (36)$$

这里 γ 和 β 都是常数，对原始数据进行一定的放缩和偏移变换；

假设损失为 l 则BP求导可有以下结果：

$$\begin{aligned} \frac{\partial l}{\partial \hat{x}_i} &= \frac{\partial l}{\partial y_i} \cdot \gamma \\ \frac{\partial l}{\partial \sigma_{\mathcal{B}}^2} &= \sum_{i=1}^m \frac{\partial l}{\partial \hat{x}_i} \cdot (x_i - \mu_{\mathcal{B}}) \cdot \frac{-1}{2(\sigma_{\mathcal{B}}^2 + \epsilon)^{3/2}} \\ \frac{\partial l}{\partial \mu_{\mathcal{B}}} &= \sum_{i=1}^m \frac{\partial l}{\partial \hat{x}_i} \cdot \frac{-1}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \\ \frac{\partial l}{\partial x_i} &= \frac{\partial l}{\partial \hat{x}_i} \cdot \frac{-1}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} + \frac{\partial l}{\partial \sigma_{\mathcal{B}}^2} \cdot \frac{2(x_i - \mu_{\mathcal{B}})}{m} + \frac{\partial l}{\partial \mu_{\mathcal{B}}} \cdot \frac{1}{m} \\ \frac{\partial l}{\partial \gamma} &= \sum_{i=1}^m \frac{\partial l}{\partial y_i} \cdot \hat{x}_i \\ \frac{\partial l}{\partial \beta} &= \sum_{i=1}^m \frac{\partial l}{\partial y_i} \end{aligned} \quad (37)$$

- 我们在第一部分中推导过残差层的随机梯度更新公式，注意到只需在这里将 ReLU 函数替换为 SoftMax 作为激活函数即可；
4. 全连接层（以Softmax作为激发函数）以交叉熵加上参数L2的regularization作为惩罚函数；

我们约定交叉熵损失表示为

$$c(Y, \hat{Y}) = - \sum_{i=0}^{m-1} (y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)) \quad (38)$$

这里假设 $Y \in R^m$ ，即有 m 个样本，优化目标为：

$$\begin{aligned} L(W) &= \frac{1}{m} c(Y, \hat{Y}) + \frac{\lambda}{2m} \sum_l \sum_k \sum_j W_{k,j}^{(l),2} \\ &= -\frac{1}{m} \sum_{k=0}^{m-1} (y_i \log(a_i^{(L)}) + (1 - y_i) \log(1 - a_i^{(L)})) + \frac{\lambda}{2m} \sum_l \sum_k \sum_j W_{k,j}^{(l),2} \end{aligned} \quad (39)$$

这里的上标 $(l), l = 1, \dots, L$ 表示全连接层的指标。实际上，可以假设每一层 l 我们有 n_l 个神经元。注意到，对于特定的某一层的 W ，我们有，

$$\frac{d}{dW} \left(\frac{\lambda}{2m} W^2 \right) = \frac{\lambda}{m} W \quad (40)$$

我们定义记号 $z^{(l+1)} = W^{(l+1)} a^{(l)}, l = 0, \dots, L-1, a^{(l)} = S(z^{(l)}), l = 1, \dots, L$

于是对 $i = 1, \dots, L$ ，我们可以得到如下递推关系：

$$\begin{aligned} \frac{\partial C}{\partial a^{(l)}} &= \frac{\partial C}{\partial a^{(l+1)}} \frac{\partial a^{(l+1)}}{\partial z^{(l+1)}} \frac{\partial z^{(l+1)}}{\partial a^{(l)}} = \frac{\partial C}{\partial a^{(l+1)}} S'(z^{(l+1)}) W^{(l+1)} \\ \frac{\partial C}{\partial W^{(l)}} &= \frac{\partial C}{\partial a^{(l)}} \frac{\partial a^{(l)}}{\partial z^{(l)}} \frac{\partial z^{(l)}}{\partial W^{(l)}} = \frac{\partial C}{\partial a^{(l)}} S'(z^{(l)}) a^{(l-1)} \end{aligned} \quad (41)$$

从后向前进推则可以得到对应梯度，又注意到，

$$\frac{\partial L}{\partial W^{(l)}} = \frac{1}{m} \frac{\partial C}{\partial W^{(l)}} + \frac{\lambda}{m} W^{(l)} \quad (42)$$

给定学习率 $\alpha > 0$ ，我们即可更新参数 $\widetilde{W}^{(l)} = W^{(l)} - \alpha \cdot \frac{\partial L}{\partial W^{(l)}}$ 直至收敛。

Reference:

- <https://cs231n.github.io/optimization-2/>
- <https://stackoverflow.com/questions/44512126/how-to-calculate-gradients-in-resnet-architecture>
- <https://blog.csdn.net/oBrightLamp/article/details/84561088>
- <https://www.cnblogs.com/pinard/p/6494810.html>
- <http://static.googleusercontent.com/media/research.google.com/en//pubs/archive/43442.pdf>
- https://blog.csdn.net/oBrightLamp/article/details/85067981#MaxPool_116

第七题

题目 关于潜在顾客的购买可能的估计问题。每个顾客的信息包括：年龄（Age）、收入情况（Incoming）、是否是学生（Is Student）、信用等级（Credit Rating）。是通过下表（表一）中的数据建立 **Decision Tree**，并判断如下用户的购买可能。

新用户的信息如下：年龄（50），收入（Medium），非学生（no），信用记录（excellent）

User id	Age	Incoming	Student	Credit Rating	Buying
1	<=30	high	no	fair	no
2	<=30	high	no	excellent	no
3	[31,40]	high	no	fair	yes
4	>40	medium	no	fair	yes
5	>40	low	yes	fair	yes
6	>40	low	yes	excellent	no
7	[31,40]	low	yes	excellent	yes
8	<=30	medium	no	fair	no
9	<=30	low	yes	fair	yes
10	>40	medium	yes	fair	yes
11	<=30	medium	yes	excellent	yes
12	[31,40]	medium	no	excellent	yes
13	[31,40]	high	yes	fair	yes

解答：我们采用 ID3 算法来解决问题

假设训练集为 D ，共有 K 个类别（这里是二分类问题，因此 $K = 2$ ）根据某个特征 A 的取值将训练集划分为 n 个子集 D_1, \dots, D_n 。记子集 D_i 中属于类别 C_k 的样本集合为 D_{ik}

则我们可以计算数据集 D 的经验熵

$$H(D) = - \sum_{i=1}^K \frac{|C_k|}{|D|} \log_2 \frac{|C_k|}{|D|} \quad (43)$$

计算某个特征 A 对数据集 D 的经验条件熵为

$$H(D|A) = \sum_{i=1}^n \frac{|D_i|}{|D|} H(D_i) = - \sum_{i=1}^n \frac{|D_i|}{|D|} \sum_{k=1}^K \frac{|D_{ik}|}{|D_i|} \log_2 \frac{|D_{ik}|}{|D_i|} \quad (44)$$

计算信息增益为

$$g(D, A) = H(D) - H(D|A) \quad (45)$$

信息增益比为

$$g_R(D, A) = \frac{g(D, A)}{H_A(D)}, H_A(D) = \sum_{i=1}^n \frac{|D_i|}{|D|} \log_2 \frac{|D_i|}{|D|} \quad (46)$$

ID3 算法的核心是在决策树各个节点上应用信息增益准则来选择特征，递归地构建决策树。从根节点开始，每次选择使得信息增益最大的特征作为节点的特征，由该特征的不同取值建立子节点，再对子节点递归地调用以上方法，构建决策树。当所有特征的信息增益均很小或者没有特征可以选择时停止递归（返回单节点子树）。

对于本二分类问题，可以计算

$$H(D) := I(9, 4) = 0.890 \quad (47)$$

我们约定：（对于0的非法运算处理做了一些约定）

```
def I(x,y):
    n = x+y
    if n == 0:
        return 0
    if x == 0:
        x = x+1
    if y == 0:
        y = y+1
    return -x/n*math.log2(x/n)-y/n*math.log2(y/n)
```

则对于不同特征，我们可以依次计算经验条件熵 ($I(x, y) = I(y, x)$)

$$\begin{aligned} H_{age} &= \frac{5}{13}I(2, 3) + \frac{4}{13}I(3, 1) + \frac{4}{13}I(4, 0) = 0.154 \\ H_{Incoming} &= 0.278, H_{Student} = 0.319, H_{Credit} = 0.499 \end{aligned} \quad (48)$$

具体输出如下：输出的数值都是当前节点下该特征的经验条件熵

```
Age 0.15384615384615385
Incoming 0.2776646518797547
Student 0.31859303462125327
Credit Rating 0.49924807659023557
We choose Age in next.
```

因为要最大化信息增益，所以也就是要最小化上述经验条件熵，因此选择 Age 作为第一次的分类标准。经过一次分类之后，对于 $Age: [31, 40]$ 这一类别已经是纯叶子节点，因此不再分类。而对于 $Age: \leq 30$ 以及 $Age: > 40$ 则经过计算可以分别选择 $Student$ 和 $Incoming$ 作为下一个节点的分类标准。

```
#####
Age: <=30
Incoming 0.4
Student 0.2
Credit Rating 0.5509775004326938
We choose Student in next.
#####
Age: [31,40] is unique. Stops here!
#####
Age: >40
Incoming 0.25
Student 0.6887218755408672
Credit Rating 0.396240625180289
We choose Incoming in next.
```

因为新用户的年龄为50，所以我们只看最后一个上面的最后一个叶节点，对于训练集而言它还有4条(4, 5, 6, 10)信息。

```
+++++
Incoming: medium is unique. Stops here!
+++++
Incoming: low
Student 1.0
Credit Rating -0.0
We choose Credit Rating in next.
```

再进行分类后发现，收入(Medium)的子节点是纯净的，其Buying标签只有yes。因此根据ID3构造的二分类树判断该新用户会购买！

第八题

题目 简述分类(Classification)与聚类(Clustering)的区别与联系。在做分类问题时，如果训练数据量较少，如何处理来提高分类算法的可靠性。

解答：

- **区别：**分类是有监督学习问题，它的目标是通过已知的训练数据和标签来预测未知数据的类别。聚类是无监督学习问题，它的目标是将数据点自动分组，使得分组内的数据点具有相似性，分组间的数据点具有不同性，但它并不需要已经标记好的数据来学习。
- **联系：**它们都是用来对数据进行分组的机器学习方法。分类通过找到一个函数，将数据点分到已知的类别中；聚类通过找到一组模型参数（相似性定义等），将数据点分到自动分组的类别中。

Parameter	CLASSIFICATION	CLUSTERING
类型	有监督学习	无监督学习
基础	通过已知的训练数据和标签来预测未知数据的类别	依据数据点之间的相似性将数据点自动分组，不需要数据类别标签
需求	需要训练数据集和测试数据集	不需要训练数据集和测试数据集
复杂性	相比聚类算法更困难	相比分类算法更简单
算法举例	Logistic regression, Naive Bayes classifier, Support vector machines, etc.	k-means clustering algorithm, Fuzzy c-means clustering algorithm, Gaussian (EM) clustering algorithm, etc.

当训练数据量较少时，通常可以采用以下几种方法来提高分类算法的可靠性：

1. 采用交叉验证的方法，对训练数据进行多次划分，并在每次划分的数据集上进行训练和评估，最终取平均值作为算法的评估结果。这样可以减少数据量较小带来的波动性，提高算法的可靠性。例如采用K fold方法。
2. 选择较为简单的分类模型，例如线性分类器或决策树。这些模型在训练时的复杂度较低，能够有效避免过拟合，并在数据量较少的情况下依然能够较好的拟合数据。
3. 增加正则化项，通过限制模型的复杂度，来防止过拟合。正则化项可以帮助模型选择更加简单的解决方案，从而提高模型的泛化能力。
4. 增加数据量：通过数据增强或采样方法增加训练数据量，从而获得更多的信息来训练模型。这样可以避免数据量较小

带来的模型训练不足的问题。

Reference:

1. <https://www.geeksforgeeks.org/ml-classification-vs-clustering/>

第九题

题目 什么是Fisher Information Matrix? 并列举三个Fisher Information Matrix在理论分析和算法设计的应用。

解答: Fisher Information Matrix (FIM) 是一种用于评估概率模型参数 θ 估计的准确性 (估计好坏) 的指标, 我们定义 score function $s(\theta) = \frac{\partial \log p(x|\theta)}{\partial \theta}$ 是参数 θ 的对数似然函数的梯度。容易证明 $E_{p(x|\theta)}[s(\theta)] = 0$ 。则我们定义 FIM 的计算公式如下:

$$F = E_{p(x|\theta)} \left[\left(\frac{\partial \log p(x|\theta)}{\partial \theta} \right) \left(\frac{\partial \log p(x|\theta)}{\partial \theta} \right)^T \right] = \text{Var}_{p(x|\theta)}(s(\theta)) \quad (49)$$

为了可以计算, 对于训练数据集 $X = \{x_1, \dots, x_N\}$ 我们定义经验 FIM 为:

$$F = \frac{1}{N} \sum_{i=1}^N \left[\left(\frac{\partial \log p(x_i|\theta)}{\partial \theta} \right) \left(\frac{\partial \log p(x_i|\theta)}{\partial \theta} \right)^T \right] \quad (50)$$

此外, 可以证明 FIM 与参数的对数似然函数的 Hessian 矩阵 $H_{\log p(x|\theta)}$ 之间有如下关系:

$$E_{p(x|\theta)} [H_{\log p(x|\theta)}] = -F \quad (51)$$

在实际应用中, FIM 常用于评估估计器的精度和敏感度; 并作为协方差矩阵的近似值; 由上述和 Hessian 矩阵之间的联系, 也可以将 FIM 应用于 Fisher- scoring 迭代格式。

$$\theta^{(t+1)} = \theta^{(t)} - \left[E(l''(\theta^{(t)})) \right]^{-1} l'(\theta^{(t)}) = \theta^{(t)} + F^{-1} l'(\theta^{(t)}) \quad (52)$$

其中 $l(\theta) := \log p(x|\theta)$ 是参数的对数似然函数。这表明此时不再需要计算复杂的二阶导数, 只需计算一阶导数就可以获得和 Newton 法相近的收敛速度。

此外, Fisher Information Matrix 在理论分析和算法设计中还有以下几个应用:

1. FIM 是分布 $p(x|\theta)$ 相对于参数 θ 的 Hessian 矩阵, 它是概率分布有意义的度量方式;
2. FIM 是对称且半正定的, 使得对矩阵的优化更加简单有效;
3. 只要输出的概率可能性没有变化, FIM 对于重新参数化也是不变的;
4. FIM 的最大特征值能够估计一个适当大小的学习速率 (learning rate) 来实现梯度法的收敛。

Reference:

1. <https://zhuanlan.zhihu.com/p/563212799>
2. https://en.wikipedia.org/wiki/Fisher_information
3. https://blog.csdn.net/weixin_48294000/article/details/106485587

第十题

题目 在统计检测(Detection)问题中，什么是第一类错误和第二类错误？试叙述并证明Newman-Pearson引理。

解答：我们在做假设检验的时候会犯两种错误：第一，原假设是正确的，而模型判断它为错误的；第二，原假设是错误的，而模型判断它为正确的。我们分别称这两种错误为第一类错误 (Type I error) 和第二类错误 (Type II error)：

- 第一类错误：原假设 H_0 是正确的，由于样本的随机性，观察值落入了否定域 D 从而错误地否定了原假设（拒真）；
- 第二类错误：原假设 H_0 是错误的，由于样本的随机性，观察值落入了否定域 \bar{D} 从而错误地没有拒绝原假设（取伪）；

如果原假设为 H_0 ，我们定义：

$$\phi(x) = \begin{cases} 1 & , \text{拒绝 } H_0 \\ \gamma & , \text{以此概率拒绝 } H_0 \\ 0 & , \text{接受 } H_0 \end{cases} \quad (53)$$

为检验函数，定义 $\beta_\phi = E_\theta[\phi(X)]$ 为功效函数。

则犯第一类错误的概率可以表示为：

$$\alpha_\phi(\theta) = \begin{cases} \beta_\phi(\theta) & , \theta \in \Theta_0 \\ 0 & , \theta \in \Theta_1 \end{cases} \quad (54)$$

犯第二类错误的概率可以表示为

$$\gamma_\phi(\theta) = \begin{cases} 0 & , \theta \in \Theta_0 \\ 1 - \beta_\phi(\theta) & , \theta \in \Theta_1 \end{cases} \quad (55)$$

Newman-Pearson引理：假设检验问题为 $H_0 : \theta = \theta_0$ v.s $H_1 : \theta = \theta_1$ ，样本 X 的概率函数 $f(x, \theta)$ ，给定 $\alpha \in (0, 1)$ 则：

1. 存在性：存在检验函数 $\phi(x)$ 以及非负常数 k 和 $0 \leq r \leq 1$ ，满足条件：

◦

$$\phi(x) = \begin{cases} 1 & , \frac{f(X, \theta_1)}{f(X, \theta_0)} > c \\ \gamma & , \frac{f(X, \theta_1)}{f(X, \theta_0)} = c \\ 0 & , \frac{f(X, \theta_1)}{f(X, \theta_0)} < c \end{cases} \quad (56)$$

◦ $E_{\theta_0}[\phi(X)] = \alpha$

2. 满足上述两个条件的检验函数 $\phi(x)$ 是一个水平为 α 的UMP检验（最有效力的检验）

证明：先证明存在性，记随机变量 $f(X, \theta_1)/f(X, \theta_0)$ 的分布函数为

$$G(y) = P\left(\frac{f(X, \theta_1)}{f(X, \theta_0)} < y\right), y \in \mathbb{R} \quad (57)$$

则 $G(y)$ 具有分布函数的性质，从而由 $G(y)$ 的单调性可知：必存在 c 使得

$$G(c) \leq 1 - \alpha \leq G(c+) \quad (58)$$

- 如果 $G(c) = 1 - \alpha$ 则取 $r = 1$ ，则 $\phi(x)$ 满足：

$$E_{\theta_0}[\phi(X)] = 1 - P_{\theta_0} \left(\frac{f(X, \theta_1)}{f(X, \theta_0)} < c \right) = 1 - G(c) = \alpha \quad (59)$$

- 如果 $G(c+) = 1 - \alpha$ 则取 $r = 0$, 则 $\phi(x)$ 满足:

$$E_{\theta_0}[\phi(X)] = 1 - P_{\theta_0} \left(\frac{f(X, \theta_1)}{f(X, \theta_0)} \leq c \right) = 1 - (G(c) + G(c+) - G(c)) = 1 - G(c+) = \alpha \quad (60)$$

- 如果 $G(c) < 1 - \alpha < G(c+)$ 则取 $r = [G(c+) - (1 - \alpha)]/[G(c+) - G(c)]$, 则 $\phi(x)$ 满足:

$$\begin{aligned} E_{\theta_0}[\phi(X)] &= P_{\theta_0} \left(\frac{f(X, \theta_1)}{f(X, \theta_0)} > c \right) + r \cdot P_{\theta_0} \left(\frac{f(X, \theta_1)}{f(X, \theta_0)} = c \right) \\ &= 1 - G(c) - (G(c+) - G(c)) + \frac{G(c+) - (1 - \alpha)}{G(c+) - G(c)} (G(c+) - G(c)) = \alpha \end{aligned} \quad (61)$$

•

再证明最优性: 假设 $\phi'(X)$ 为任意的另一个水平为 α 的检验, 定义样本空间中子集

$$S^+ = \{x : \phi(x) > \phi'(x)\}, S^- = \{x : \phi(x) < \phi'(x)\} \quad (62)$$

则在 S^+ 上有 $\phi(x) > \phi'(x) \geq 0$, 因此有

$$\frac{f(X, \theta_1)}{f(X, \theta_0)} \geq c \iff f(X, \theta_1) - cf(X, \theta_0) \geq 0 \quad (63)$$

在 S^- 上有 $\phi(x) < \phi'(x) \leq 1$, 因此有

$$\frac{f(X, \theta_1)}{f(X, \theta_0)} \leq c \iff f(X, \theta_1) - cf(X, \theta_0) \leq 0 \quad (64)$$

所以在 $S = S^+ \cup S^-$ 上有: $(\phi(x) - \phi'(x))(f(X, \theta_1) - cf(X, \theta_0)) \geq 0$ 因此,

$$\int_H (\phi(x) - \phi'(x))(f(X, \theta_1) - cf(X, \theta_0)) dx = \int_S (\phi(x) - \phi'(x))(f(X, \theta_1) - cf(X, \theta_0)) dx \geq 0 \quad (65)$$

就是

$$\int_H \phi(x)f(X, \theta_1) dx - \int_H \phi(x)'f(X, \theta_1) dx \geq c \left(\int_H \phi(x)f(X, \theta_0) dx - \int_H \phi(x)'f(X, \theta_0) dx \right) \quad (66)$$

由于根据存在性定义, $E_{\theta_0}[\phi(X)] = \alpha$ 以及 $E_{\theta_0}[\phi'(X)] \leq \alpha$ 我们可以得到

$\int_H \phi(x)f(X, \theta_1) dx \geq \int_H \phi(x)'f(X, \theta_1) dx$ 证毕!