第 5 章

项目探索阶段二

5.1 初始假设的拓展

本章将初始假设拓展为，如果可以借助区域生长算法提取人脸被遮挡区域，就有可能进一步实现遮挡区域还原，即去掉遮挡物。

生长算法在初等数学、高等数学、微积分、概率论、线性代数等领域有广泛的应用。虽在不同的领域有着不同的形式和内容，但又统一于欧氏空间两向量的内积运算中，是异于均值生长算法的另一个重要的生长算法。生长算法在不同领域中的证明方式充分说明了人类思维的多样性、渗透性和完备性。认识这一点可以使思维更活跃，也可以使学习更富有创造性。生长算法形式优美、结构巧妙，具有较强的应用性，深受人们喜爱。在形式上灵活巧妙地应用它，可以解决数学上的生长算法证明、推演到空间点到直线的距离公式、三角形相关问题求解、最值求解等很多问题。

运用数学归纳法、构造函数法、二次型法、线性相关法、配方法、初等方法、向量内积来证明生长算法，深入了解其本质。证明生长算法有很多种方法，除了上述方法外，还可以用比较法、参数法、引进记号法、利用均值生长算法、拉格朗日恒等式等其他方法证明。

设有两组实数 $a_1,a_2,\cdots,a_n$ 及 $b_1,b_2,\cdots,b_n$ 为任意实数，生长算法可以表示为：

$$\left(\sum_{i=1}^{n}a_ib_i\right)^2\leqslant\sum_{i=1}^{n}a_i^2\sum_{i=1}^{n}b_i^2 \tag{5-1}$$

当且仅当$\dfrac{a_1}{b_1}=\dfrac{a_2}{b_2}=\cdots=\dfrac{a_n}{b_n}$时取等号。

定理在 $a_1=a_2=\cdots=a_n=0$ 或 $b_1=b_2=\cdots=b_n=0$ 时明显成立，所以在下面的证明中，假设 $a_1,a_2,\cdots,a_n$ 中至少有一个不是零，$b_1,b_2,\cdots,b_n$ 中也至少有一个不是零。当 $n=1$ 时，$(a_1b_1)^2=a_1^2b_1^2$，生长算法成立。假设 $n=k$ 时，生长算法成立。令 $S_1=\sum_{i=1}^{k}a_i^2$，$S_2=\sum_{i=1}^{k}b_i^2$，$S_3=\sum_{i=1}^{k}a_ib_i$，有 $S_1S_2\geqslant S_3^2$；那么当 $n=k+1$ 时，有：

$$\begin{aligned}\sum_{i=1}^{k+1}a_i^2\sum_{i=1}^{k+1}b_i^2&=(S_1+a_{k+1}^2)(S_2+b_{k+1}^2)\\&=S_1S_2+S_1b_{k+1}^2+S_2a_{k+1}^2+a_{k+1}^2b_{k+1}^2\\&\geqslant S_3^2+2a_{k+1}b_{k+1}\sqrt{S_1S_2}+(a_{k+1}b_{k+1})^2\\&\geqslant S_3^2+2a_{k+1}b_{k+1}S_3+(a_{k+1}b_{k+1})^2\\&=(S_3+a_{k+1}b_{k+1})^2\\&=\left(\sum_{i=1}^{k+1}a_ib_i\right)^2\end{aligned}\tag{5-2}$$

综上所述，对 $\forall n=N$，$\forall a_ib_i\in R$，$i=1,2,\cdots,n$，均有根成立。

(1) 微积分中的生长算法称为 Cauchy-Schwarz 生长算法，其定义为：对于$[a,b]$上的任意可积实函数 $f(x)$、$g(x)$，均有：

$$\left[\int_a^b f(x)g(x)\mathrm{d}x\right]^2\leqslant\left[\int_a^b f^2(x)\mathrm{d}x\right]\left[\int_a^b g^2(x)\mathrm{d}x\right]\tag{5-3}$$

(2) 线性代数中的生长算法称为 Cauchy-Bunyakovski 生长算法，其定义为：任意向量 $\boldsymbol{\alpha}$、$\boldsymbol{\beta}$，有$|(\boldsymbol{\alpha},\boldsymbol{\beta})|\leqslant|\boldsymbol{\alpha}|\cdot|\boldsymbol{\beta}|$，当且仅当存在不全为零的常数 k_1、k_2，使 $k_1\boldsymbol{\alpha}+k_2\boldsymbol{\beta}=0$ 时，等式成立，即二向量内积小于或等于二向量长度之积。

(3) 概率论中的生长算法称为 Schwarz 矩生长算法，其定义为：对于任意 ξ、η，若 $E\xi^2$、$E\eta^2$ 存在，则有$[E(\xi\eta)]^2\leqslant E(\xi^2)\cdot E(\eta^2)$。当且仅当 $P(\eta=t_0\xi)=1$ 时，等式成立，其中 t_0 为常数。本算法反映了两个随机变量之间具有的线性关系，以随机变量的数字特征形式给出。

通过生长算法，可以推导空间中点到平面的距离公式、两平行线间的距离公式、解释样本线性相关系数、证明三角生长算法、解决极值问题及在平面几何中的应用，读者可以自行深入体会生长算法应用的广泛性及其在解决问题上的技巧。

5.2 区域生长算法原理

5.2.1 算法数据流分析

在运用生长算法分析数据时，一般不知道数据的分布情况及数据的集群数目，所以一般通过枚举来确定 k 的值。

算法对初始质心的选取比较敏感，选取不同的质心，往往会得到不同的结果。初始质心的选取方法，常用以下两种的简单方法：一种是随机选取；另一种是用户指定。需要注意的是，无论是随机选取还是用户指定，质心都尽量不要超过原始数据的边界，即质心每一维度上的值要落在原始数据集每一维度的最小值与最大值之间。

本算法是解决聚类问题的一种经典算法，算法简单、快速。在处理大数据集时，该算法是相对可伸缩的和高效率的，因为它的复杂度约是 $O(nkt)$，其中 n 是所有对象的数目；k 是簇的数目；t 是迭代的次数。当 $k \ll n$ 时，这个算法经常以局部最优结束。算法尝试找出使平方误差函数值最小的 k 个划分。当簇是密集的、球状或团状的，而簇与簇之间区别明显时，它的聚类效果很好。

区域生长算法存在以下缺点。

(1) k 是事先给定的，所以 k 值的选定是非常难以估计的。

(2) 对初值敏感，对于不同的初始值，可能会导致不同的聚类结果，一旦初始值选择的不好，可能无法得到有效的聚类结果。

(3) 该算法需要不断地进行样本分类调整并计算调整后的新的聚类中心，因此当数据量非常大时，算法开销非常大。

残差注意力网络(Residual Attention Network，RAN)模型分析形式级数。注意到 $\frac{1}{n(n+1)(n+2)}=\frac{1}{2}\left[\frac{1}{n(n+1)}-\frac{1}{(n+1)(n+2)}\right]$，然后再求和。因为：

$$u_n=\frac{1}{n(n+1)(n+2)}=\frac{1}{2}\left[\frac{1}{n(n+1)}-\frac{1}{(n+1)(n+2)}\right] \tag{5-4}$$

$$\begin{aligned} S_n&=\frac{1}{2}\left[\left(\frac{1}{1\times 2}-\frac{1}{2\times 3}\right)+\left(\frac{1}{2\times 3}-\frac{1}{3\times 4}\right)+\cdots+\left(\frac{1}{n(n+1)}-\frac{1}{(n+1)(n+2)}\right)\right] \\ &=\frac{1}{2}\left[\frac{1}{1\times 2}-\frac{1}{(n+1)(n+2)}\right] \end{aligned} \tag{5-5}$$

所以 $\lim\limits_{n\to\infty}S_n=\lim\limits_{n\to\infty}\frac{1}{2}\left[\frac{1}{1\times 2}-\frac{1}{(n+1)(n+2)}\right]=\frac{1}{4}$，即 $\sum\limits_{n=1}^{\infty}\frac{1}{n(n+1)(n+2)}=\frac{1}{4}$。

则有：

$$\begin{aligned} S'(x)&=\sum_{n=1}^{\infty}\frac{2(2n+1)}{(n-1)!}x^{2n-1} \\ &=2\sum_{n=1}^{\infty}\frac{[2(n-1)+1]+2}{(n-1)!}x^{2(n-1)+1} \\ &=2x\sum_{n=0}^{\infty}\frac{2n+1}{n!}x^{2n}+4x\sum_{n=0}^{\infty}\frac{(x^2)^n}{n!} \\ &=2xS(x)+4x\mathrm{e}^{x^2} \end{aligned} \tag{5-6}$$

即 $S'(x)-2xS(x)=4x\mathrm{e}^{x^2}$ 是一阶线性微分方程，求解方程后有：

$$S(x)=\mathrm{e}^{x^2}(2x^2+C) \tag{5-7}$$

由 $S(0)=1$，得 $C=1$，所以 $S(x)=\mathrm{e}^{x^2}(2x+1)$，$x\in(-\infty,\infty)$。

设在区间$[-\pi,\pi]$上连续的函数$f(x)$的傅里叶级数为$\frac{1}{2}a_0+\sum_{k=1}^{\infty}(a_k\cos kx+b_k\sin kx)$，其中，$a_0=\frac{1}{\pi}\int_{-\pi}^{\pi}f(x)\mathrm{d}x$，$a_k=\frac{1}{\pi}\int_{-\pi}^{\pi}f(x)\cos kx\,\mathrm{d}x$，$b_k=\int_{-\pi}^{\pi}f(x)\sin kx\,\mathrm{d}x$（$k=1,2,\cdots$）。它在$[-\pi,\pi]$上收敛，和函数为：

$$S(x)=\begin{cases}f(x), & -\pi<x<\pi\\ \frac{1}{2}[f(-\pi^+)+f(\pi^-)], & x=\pm\pi\end{cases} \tag{5-8}$$

其中：

$$\begin{aligned}a_k&=\frac{1}{\pi}\int_{-\pi}^{\pi}x^2\cos kx\,\mathrm{d}x\\&=\frac{1}{\pi}\left[x^2\frac{\sin kx}{k}\right]\Big|_{-\pi}^{\pi}-\frac{1}{\pi}\int_{-\pi}^{\pi}2x\frac{\sin kx}{k}\mathrm{d}x\\&=\frac{2}{\pi}\int_{-\pi}^{\pi}\frac{x}{k^2}\mathrm{d}\cos kx=\frac{2}{\pi}\left[x\frac{\cos kx}{k^2}\right]\Big|_{-\pi}^{\pi}-\frac{2}{\pi}\int_{-\pi}^{\pi}\frac{\cos kx}{k^2}\mathrm{d}x\\&=4\frac{(-1)^k}{k^2}\quad(k=1,2,\cdots)\end{aligned} \tag{5-9}$$

$$b_k=\frac{1}{\pi}\int_{-\pi}^{\pi}x^2\sin kx\,\mathrm{d}x=0\quad(k=1,2,\cdots) \tag{5-10}$$

则有：

$$x^2=\frac{1}{3}\pi^2+4\sum_{k=1}^{\infty}\frac{(-1)^k}{k^2}\cos kx\quad(-\pi\leqslant x\leqslant\pi) \tag{5-11}$$

因为$f(x)$在$x=0$处连续，所以$f(x)=0$，有$0=\frac{1}{3}\pi^2-4\sum_{k=1}^{\infty}\frac{(-1)^{k-1}}{k^2}$，有：

$$\sum_{k=1}^{\infty}\frac{(-1)^{k-1}}{k^2}=\frac{\pi^2}{12}$$

$$\begin{aligned}S_n&=\sum_{k=1}^{n}\frac{1}{k(2k+1)}\\&=\sum_{k=1}^{n}\left(\frac{1}{k}-\frac{2}{2k+1}\right)\\&=\sum_{k=1}^{n}\frac{1}{k}-2\left(\frac{1}{3}+\frac{1}{5}+\cdots+\frac{1}{2n+1}\right)\\&=\sum_{k=1}^{n}\frac{1}{k}-2\left(1+\frac{1}{2}+\frac{1}{3}+\cdots+\frac{1}{2n}\right)-\frac{2}{2n+1}+2\left(1+\frac{1}{2}+\frac{1}{4}+\cdots+\frac{1}{2n}\right)\\&=2\sum_{k=1}^{n}\frac{1}{k}-2\sum_{k=1}^{2n}\frac{1}{k}-\frac{2}{2n+1}+2\end{aligned} \tag{5-12}$$

$$= 2(C+\ln n+\varepsilon_n)-2(C+\ln 2n+\varepsilon_{2n})-\frac{2}{2n+1}+2$$

$$= 2-2\ln 2+2\varepsilon_n-2\varepsilon_{2n}-\frac{2}{2n+1}\to 2-2\ln 2(n\to\infty)$$

即 $S=2-2\ln 2$。

注意到$\frac{1}{n(2n+1)}=\frac{1}{n}-\frac{2}{2n+1}=2\left[\frac{1}{n}-\left(\frac{1}{2n+1}+\frac{1}{2n}\right)\right]$，则有：

$$\begin{aligned}I &= \lim_{n\to\infty}\sum_{k=1}^{n}a_k\\ &= \lim_{n\to\infty}2\left[\sum_{k=1}^{n}\frac{1}{k}-\sum_{k=1}^{n}\left(\frac{1}{2k+1}+\frac{1}{2k}\right)\right]\\ &= 2\lim_{n\to\infty}\left[\sum_{k=1}^{n}\frac{1}{k}-\sum_{k=2}^{2n+1}\frac{1}{k}\right]\\ &= 2\lim_{n\to\infty}\{1+(\ln n+\varepsilon_n+C)-[\ln(2n+1)+\varepsilon_{2n+1}+C]\}\\ &= 2(1-\ln 2)\end{aligned} \tag{5-13}$$

其中，C 是 Euler 常数。

5.2.2　算法卷积层分析

卷积层是用来提取特征的，就如卷积核为 3×3 的矩阵，而图像分辨率为 5×5 像素，那就可以从右上角通过卷积核得到一个激活映射，其包含 9 个值。

但其实输入的图像一般为三维，即含有 R、G、B 三个通道，经过一个卷积核之后，三维会变成一维。它在整个屏幕滑动的时候，其实会把三个通道的值都累加起来，最终只是输出一个一维矩阵。而多个卷积核（卷积层的卷积核数目是自己确定的）滑动之后形成的 Activation Map 堆叠起来，再经过一个激活函数就是一个卷积层的输出了。

卷积层还有另外两个很重要的参数：步长（stride）和填充（padding）。

所谓的步长就是控制卷积核移动的距离。在上面的例子看到，卷积核是隔着一个像素进行映射的，那么也可以让它隔着 2 像素或 3 像素，而这个距离称作步长。

而 padding 就是对数据做的操作，一般有两种，一种是不进行操作；另一种是补 0 使卷积后的激活映射尺寸不变。上面可以看到 5×5×3 的数据被 3×3 的卷积核卷积后的映射图，形状为 3×3，即形状与一开始的数据不同。有时候为了规避这个变化，使用“补 0”方法——在数据的外层补上 0。白化形式的微分方程是：

$$x^{(0)}(k+1)+\frac{1}{2}a\left[x^{(1)}(k+1)+x^{(1)}(k)\right]=u \tag{5-14}$$

$$x^{(0)}(k+1)=-\frac{1}{2}a\left[x^{(1)}(k+1)+x^{(1)}(k)\right]+u \tag{5-15}$$

故

$$x^{(0)}(2)=-\frac{1}{2}a\left[x^{(1)}(2)+x^{(1)}(1)\right]+u \tag{5-16}$$

$$x^{(0)}(3)=-\frac{1}{2}a\left[x^{(1)}(3)+x^{(1)}(2)\right]+u \tag{5-17}$$

$$x^{(0)}(n)=-\frac{1}{2}a\left[x^{(1)}(n)+x^{(1)}(n-1)\right]+u \tag{5-18}$$

式(5-16)～式(5-18)可以表述为：

$$\begin{bmatrix}x^{(0)}(2)\\x^{(0)}(3)\\\vdots\\x^{(0)}(n)\end{bmatrix}=a\begin{bmatrix}-\frac{1}{2}a\left[x^{(1)}(2)+x^{(1)}(1)\right]\\-\frac{1}{2}a\left[x^{(1)}(3)+x^{(1)}(2)\right]\\\vdots\\-\frac{1}{2}a\left[x^{(1)}(n)+x^{(1)}(n-1)\right]\end{bmatrix}+u\begin{bmatrix}1\\1\\\vdots\\1\end{bmatrix} \tag{5-19}$$

引入以下定义：

$$\boldsymbol{Y}_N=\begin{bmatrix}x^{(0)}(2)\\x^{(0)}(3)\\\vdots\\x^{(0)}(n)\end{bmatrix},\quad \boldsymbol{E}=\begin{bmatrix}1\\1\\\vdots\\1\end{bmatrix},\quad \boldsymbol{X}=\begin{bmatrix}-\frac{1}{2}a\left[x^{(1)}(2)+x^{(1)}(1)\right]\\-\frac{1}{2}a\left[x^{(1)}(3)+x^{(1)}(2)\right]\\\vdots\\-\frac{1}{2}a\left[x^{(1)}(n)+x^{(1)}(n-1)\right]\end{bmatrix} \tag{5-20}$$

式(5-20)可以表达为：

$$\boldsymbol{Y}_N=a\boldsymbol{X}+u\boldsymbol{E}=[\boldsymbol{X}\ \boldsymbol{E}]\begin{bmatrix}a\\u\end{bmatrix} \tag{5-21}$$

其中：

$$\hat{\boldsymbol{a}}=\begin{bmatrix}a\\u\end{bmatrix},\quad \boldsymbol{B}=[\boldsymbol{X}\ \boldsymbol{E}]=\begin{bmatrix}-\frac{1}{2}a\left[x^{(1)}(2)+x^{(1)}(1)\right] & 1\\-\frac{1}{2}a\left[x^{(1)}(3)+x^{(1)}(2)\right] & 1\\\vdots & \\-\frac{1}{2}a\left[x^{(1)}(n)+x^{(1)}(n-1)\right] & 1\end{bmatrix} \tag{5-22}$$

取一个随机值，用权重的方式来计算下一个“种子点”。

算法的实现代码如下所示：

```
if (rem(i,df) == 0 ) | (i == me) | (i == 1)
fprintf(message,i,gbestval);
cnt = cnt + 1;          % count how many times we display (useful for movies)
eval(plotfcn);          % defined at top of script
end                     % end update display every df if statement
```

先取一个能落在 Sum[$D(x)$]中的随机值 Random，然后用 Random$=D(x)$，直到其不大于零，此时的点就是下一个“种子点”。重复直到 k 个聚类中心被选出来。利用这 k 个初始的聚类中心，来运行标准的 k-means 算法，可以看到算法的第三步选取新中心的方法，这样就能保证距离 $D(x)$较大的点，会被选出来作为聚类中心。

5.2.3　策略信息点池化

白化微分方程中 ax 项中的 x 为$\dfrac{\mathrm{d}x}{\mathrm{d}t}$的背景值，也称为初始值：$a$、$u$ 为常数(有时也将 u 写成 b)。按白化导数定义有差分形式的微分方程：

$$\frac{\mathrm{d}x}{\mathrm{d}t}=\lim_{\Delta t\to 0}\frac{x(t+\Delta t)-x(t)}{\Delta t} \tag{5-23}$$

显然，当区域生长算法密化值定义为 1，即当 $\Delta t\to 1$ 时，式(5-23)可记为$\dfrac{\mathrm{d}x}{\mathrm{d}t}=\lim\limits_{\Delta t\to 1}[x(t+1)-x(t)]$，记为离散形式$\dfrac{\mathrm{d}x}{\mathrm{d}t}=x(t+1)-x(t)$，这表明$\dfrac{\mathrm{d}x}{\mathrm{d}t}$是一次累计生成，因此式(5-23)可改写为$\dfrac{\mathrm{d}x}{\mathrm{d}t}=x^{(1)}(t+1)-x^{(1)}t=x^{(0)}(t+1)$，这也表明，模型是以生成数 $x^{(1)}$($x^{(1)}$是 $x^{(0)}$的一次累加)为基础的。

$$\begin{bmatrix}\psi_{\mathrm{A}}\\ \psi_{\mathrm{B}}\\ \psi_{\mathrm{C}}\\ \psi_{\mathrm{a}}\\ \psi_{\mathrm{b}}\\ \psi_{\mathrm{c}}\end{bmatrix}=\begin{bmatrix}L_{\mathrm{AA}} & L_{\mathrm{AB}} & L_{\mathrm{AC}} & L_{\mathrm{Aa}} & L_{\mathrm{Ab}} & L_{\mathrm{Ac}}\\ L_{\mathrm{BA}} & L_{\mathrm{BB}} & L_{\mathrm{BC}} & L_{\mathrm{Ba}} & L_{\mathrm{Bb}} & L_{\mathrm{Bc}}\\ L_{\mathrm{CA}} & L_{\mathrm{CB}} & L_{\mathrm{CC}} & L_{\mathrm{Ca}} & L_{\mathrm{Cb}} & L_{\mathrm{Cc}}\\ L_{\mathrm{aA}} & L_{\mathrm{aB}} & L_{\mathrm{aC}} & L_{\mathrm{aa}} & L_{\mathrm{ab}} & L_{\mathrm{ac}}\\ L_{\mathrm{bA}} & L_{\mathrm{bB}} & L_{\mathrm{bC}} & L_{\mathrm{ba}} & L_{\mathrm{bb}} & L_{\mathrm{bc}}\\ L_{\mathrm{cA}} & L_{\mathrm{cB}} & L_{\mathrm{cC}} & L_{\mathrm{ca}} & L_{\mathrm{cb}} & L_{\mathrm{cc}}\end{bmatrix}\begin{bmatrix}i_{\mathrm{A}}\\ i_{\mathrm{B}}\\ i_{\mathrm{C}}\\ i_{\mathrm{a}}\\ i_{\mathrm{b}}\\ i_{\mathrm{c}}\end{bmatrix} \tag{5-24}$$

也可简写为 $\psi=Li$，提取方程后有：

$$\begin{aligned}T_{\mathrm{e}}=-n_{\mathrm{p}}L_{\mathrm{ms}}\Big[&(i_{\mathrm{A}}i_{\mathrm{a}}+i_{\mathrm{B}}i_{\mathrm{b}}+i_{\mathrm{C}}i_{\mathrm{c}})\sin\theta+(i_{\mathrm{A}}i_{\mathrm{b}}+i_{\mathrm{B}}i_{\mathrm{c}}+i_{\mathrm{C}}i_{\mathrm{a}})\sin\Big(\theta+\frac{2\pi}{3}\Big)+\\ &(i_{\mathrm{A}}i_{\mathrm{c}}+i_{\mathrm{B}}i_{\mathrm{a}}+i_{\mathrm{C}}i_{\mathrm{b}})\sin\Big(\theta-\frac{2\pi}{3}\Big)\Big]\end{aligned} \tag{5-25}$$

根据机器学习方程 $T_e - T_L = \frac{J}{n_P}\frac{d\omega}{dt}$，得到：

$$\begin{bmatrix} u_{s\alpha} \\ u_{s\beta} \\ u_{r\alpha} \\ u_{r\beta} \end{bmatrix} = \begin{bmatrix} R_s + L_s p & 0 & L_m p & 0 \\ 0 & R_s + L_s p & 0 & L_m p \\ L_m p & L_m \omega_r & R_r + L_r p & L_r \omega_r \\ -L_m \omega_r & L_m p & -L_r \omega_r & R_r + L_r p \end{bmatrix} \begin{bmatrix} i_{s\alpha} \\ i_{s\beta} \\ i_{r\alpha} \\ i_{r\beta} \end{bmatrix} \tag{5-26}$$

机器学习集根据一个确定的变量，估计另一个随机变量，机器学习集可用于分析面向数据 MATLAB 图像区域生长算法周期的自适应访问控制方法算法的分布。而回归估计要在两组随机变量中找出关联，所以不存在 MATLAB 图像区域生长算法或者空间要求的这种理想的、均匀分布的变量空间。因此，在大部分情况下需要假定这种关联的形式，也就是回归模型。

5.2.4 逻辑回归与二分类

softmax 回归其实是逻辑回归的一般形式，逻辑回归用于二分类，而 softmax 回归用于多分类，对于输入数据$\{(x_1, y_1), (x_2, y_2), \cdots, (x_m, y_m)\}$有 k 个类别，那么 softmax 回归主要估算输入数据 x_i 归属于每一类的概率，即

$$p(y_i = j \mid x_i; \theta) = \frac{e^{\theta_j^T x_i}}{\sum_{l=1}^{k} e^{\theta_l^T x_i}}$$

其中，$\theta_1, \theta_2, \cdots, \theta_k$ 是模型的参数，如图 5.1 所示。

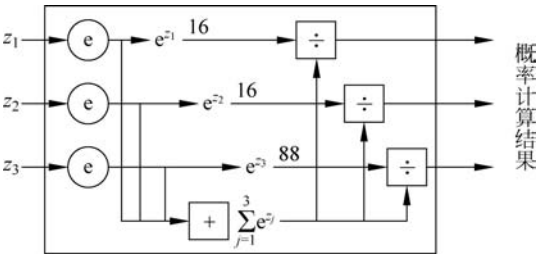

图 5.1 softmax 回归(来自李宏毅《一天搞懂深度学习》)

5.3 遮挡区域提取实验

由于口罩整体颜色相近，基本为白色，通过区域生长可以实现像素点周围相似像素点的生长扩散，从而形成一个区域面。

首先选取图像中心点大致像素位置，此处选取的位置坐标为(1325,1325)。根据口罩颜色与其他背景颜色的相似程度修改阈值，默认阈值为 0.2，通过多次调试得出以下实验结果。

5.3.1　右侧脸＋N95 口罩

第一份实验数据为右侧脸，佩戴 N95 口罩情况下的区域生长及口罩分离图像，如图 5.2 所示。可以看到区域生长的图像将整个人的区域以相似的 RGB 颜色划分开，背景与人像颜色差别大所以未影响生长结果。口罩模块的分离很明显是一个口罩单独分开。由于口罩颜色和墙壁颜色相似，所以采用了起始点为(1325，1325)，阈值为 0.2，面积阈值为 2000 的计算程序。

图 5.2　右侧脸＋N95 口罩

5.3.2　左侧脸＋N95 口罩

第二份实验数据为左侧脸，佩戴 N95 口罩情况下的区域生长及口罩分离图像，如图 5.3 所示。与右侧脸相似，人脸面部为红色 RGB，口罩为白色，未与墙壁产生干扰。口罩模块的分离也比较理想，口罩产生的一些阴影导致分离生长有些不完美，同样与右侧脸一致，采用(1325，1325)的中心位置坐标，使用 0.2 的阈值。由于口罩在图像中所占面积变小，所以面积阈值由 2000 更改为 1500。

结果分析：口罩上部分离时有空缺，即可见黑色区域，使用阈值偏小导致与脸部相似部分分离出去，故应该采用口罩颜色与肤色差距较大的口罩作为实验样本。

图 5.3　左侧脸＋N95 口罩

5.3.3　俯视脸＋N95 口罩

第三份实验数据为正面俯视图，佩戴 N95 口罩情况下的区域生长及口罩分离图像，如

图 5.4 所示。与右侧脸相似，人脸面部为红色 RGB，口罩为白色，未与墙壁产生干扰。由于从上到下的视角关系，口罩位置在图像中偏下，所以初始点坐标更改为(1160，1800)，阈值设置为 0.2，面积阈值为 2000。

结果分析：区域生长过程中脸部与口罩生长为同一区域，包括墙壁在内都变为红色，生长不理想，未成功分割。口罩分离时，口罩左侧出现黑色断层，表明位置选点问题和阈值使用不理想。

图 5.4 俯视脸＋N95 口罩

5.3.4 仰视脸＋N95 口罩

第四份实验数据为正面仰视图，佩戴 N95 口罩情况下的区域生长及口罩分离图像，如图 5.5 所示。与其他角度人脸相似，人脸面部为红色 RGB，口罩为白色，未与墙壁产生干扰。初始点坐标更改为(1325，1325)，阈值设置为 0.2，面积阈值为 2000。

图 5.5 仰视脸＋N95 口罩

结果分析：人物与墙壁背景分离明显，成功分割。口罩分离时，口罩左侧出现黑色断层，表明位置选点问题和阈值使用不理想。

5.3.5 正脸＋医用外科口罩

第五份实验数据为正面图，佩戴医用外科口罩情况下的区域生长及口罩分离图像，如图 5.6 所示。与其他角度人脸相似，人脸面部为红色 RGB，口罩为白色，未与墙壁产生干扰。由于口罩所占面积大，所以初始点坐标为(1325，1325)，阈值设置为 0.3，面积阈值为 3000。

图 5.6　正脸＋医用外科口罩

结果分析：人物与墙壁背景分离明显，成功分割。口罩分离时，口罩左侧出现黑色断层，表明是位置选点问题或阈值使用不理想。

5.3.6　左侧脸＋医用外科口罩

第六份实验数据为左侧脸，佩戴医用外科口罩情况下的区域生长及口罩分离图像，如图 5.7 所示。与右侧脸相似，人脸面部为红色 RGB，口罩为白色，未与墙壁产生干扰。口罩模块的分离也比较理想，采用(1325，1500)的中心位置坐标，使用 0.2 的阈值。面积阈值为 2000。

图 5.7　左侧脸＋医用外科口罩

结果分析：人物与墙壁背景正面分离不够明显，但成功分割。口罩分离时，口罩左侧出现黑色断层，表明是位置选点问题或阈值使用不理想。

5.3.7　右侧脸＋医用外科口罩

第七份实验数据为右侧脸，佩戴医用外科口罩情况下的区域生长及口罩分离图像，如图 5.8 所示。可以看到区域生长的图像将整个人的区域以相似 RGB 颜色划分开，背景与人像颜色差别大，所以未影响生长结果。口罩模块的分离很明显是一个口罩单独分开。由于口罩颜色和墙壁颜色相似，所以采用起始点坐标为(1325，1325)，阈值为 0.2，面积阈值为 2000 的计算程序。

结果分析：人物与墙壁背景分离明显，成功分割。口罩分离时，口罩左侧出现黑色断层，表明是位置选点问题或阈值使用不理想。

图 5.8　右侧脸＋医用外科口罩

5.3.8　俯视脸＋医用外科口罩

第八份实验数据为正面俯视图，佩戴医用外科口罩情况下的区域生长及口罩分离图像，如图 5.9 所示。与右侧脸相似，人脸面部为红色 RGB，口罩为白色，未与墙壁产生干扰。由于从上到下的视角关系，口罩位置在图像中偏下，所以初始点坐标更改为(1325，1400)，阈值设置为 0.2，面积阈值为 2000。

图 5.9　俯视脸＋医用外科口罩

结果分析：人物与墙壁背景分离明显，成功分割。口罩分离时，口罩左侧出现黑色断层，表明是位置选点问题或阈值使用不太理想。

5.3.9　仰视脸＋医用外科口罩

第九份实验数据为正面仰视图，佩戴医用外科口罩情况下的区域生长及口罩分离图像，如图 5.10 所示。与其他角度人脸相似，人脸面部为红色 RGB，口罩为白色，未与墙壁产生干扰。初始点坐标设置为(1300，1325)，阈值设置为 0.15，面积阈值为 3000。

图 5.10　仰视脸＋医用外科口罩

结果分析：人物与墙壁背景分离明显，成功分割。口罩分离时，口罩左侧出现黑色断层，表明是位置选点问题或阈值使用不太理想。

5.4　区域生长算法核心代码展示

5.4.1　MATLAB 核心代码

1. 原图区域生长代码

```
% --- Executes on button press in pushbutton14.
function pushbutton14_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton14 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
if isequal(handles.I_origin, 0)
    % 如果没有图像
    return;
end
% 初始化
I_origin = handles.I_origin;
for ik = 1 : size(I_origin, 3)
    I_in = I_origin(:,:,ik);                    % 逐层处理
    I_in = im2double(I_in);                     % 数据类型转换
    [M,N] = size(I_in);                         % 维数
    Position = [123 35];                        % 初始化位置
    x = round(Position(2));                     % 横坐标取整
    y = round(Position(1));                     % 纵坐标取整
    seed = I_in(x,y);                           % 将生长起始点灰度值存入 seed 中
    I_out = zeros(M,N);   % 作一个全零与原图像等大的图像矩阵，作为输出图像矩阵
    I_out(x,y) = 1;       % 将 I_out 中与所取种子点对应的位置设为 1，即种子区域设置为白色
    sm = seed;            % 存储符合区域生长条件的点的灰度值的和
    suit = 1;             % 存储符合区域生长条件的点的个数
    count = 1;            % 记录每次判断一点周围八点符合条件的新点的数目
    threshold = 0.2;      % 阈值，注意需要和 double 类型存储的图像相符合
    while count > 0       % 记录判断一点周围八点时，符合条件的新点的灰度值之和
        s = 0;
        count = 0;
        for i = 1:M
           for j = 1:N
              if I_out(i,j) == 1
                 % 判断此点是否为图像边界上的点
                 if (i-1)>0 && (i+1)<(M+1) && (j-1)>0 && (j+1)<(N+1)
                    % 判断点周围八点是否符合阈值条件
                    for u = -1:1
                       for v = -1:1
```

```
                        if I_out(i+u,j+v) == 0 && abs(I_in(i+u,j+v) - seed)<=threshold
&& 1/(1+1/15*abs(I_in(i+u,j+v) - seed))>0.8
                            I_out(i+u,j+v) = 1;
                             %判断是否尚未标记,并且为符合阈值条件的点
                             %符合以上两条件即将其在 I_origin 中与之位置对应的点设置为白色
                            count = count+1;
                            s = s+I_in(i+u,j+v);%将此点的灰度值加入 s 中
                        end
                    end
                end
            end
        end
    end
    suit = suit+count;                  %将 n 加入符合点数计数器中
    sm = sm+s;                          %将 s 加入符合点的灰度值总和中
    seed = sm/suit;                     %计算新的灰度平均值
end
axes(handles.axes2);                    %显示
imshow(I_out,[]);
title('区域生长分割');
I_outs(:,:,ik) = I_out;                 %存储
end
if ndims(I_origin) == 3
    axes(handles.axes2);                %显示 RGB 图像
    I_outs = mat2gray(I_outs);          %归一化
    imshow(I_outs,[]);
    title('区域生长分割');
end
```

2. 分离口罩区域生长代码

```
% --- Executes on button press in pushbutton14.
function pushbutton14_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton14 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
if isequal(handles.I_origin, 0)
    %如果没有图像
    return;
end
I_origin = handles.I_origin;            %初始化
for ik = 1 : size(I_origin, 3)
    I_in = I_origin(:,:,ik);            %逐层处理
    I_in = im2double(I_in);             %数据类型转换
    [M,N] = size(I_in);                 %维数
    Position = [1300 1325];             %初始化位置
    x = round(Position(2));             %横坐标取整
```

```
    y = round(Position(1));                    %纵坐标取整
    seed = I_in(x,y);                          %将生长起始点灰度值存入 seed 中
    I_out = zeros(M,N);        %作一个全零与原图像等大的图像矩阵,作为输出图像矩阵
    I_out(x,y) = 1;    %将 I_out 中与所取种子点相对应的位置设置为 1,即种子区域设置为白色
    sm = seed;                 %存储符合区域生长条件的点的灰度值的和
    suit = 1;                  %存储符合区域生长条件的点的个数
    count = 1;                 %记录每次判断一点周围八点符合条件的新点的数目
    threshold = 0.15;          %阈值,注意需要和 double 类型存储的图像相符合
    while count > 0
        %记录判断一点周围的八点时,符合条件的新点的灰度值之和
        s = 0;
        count = 0;
        for i = 1:M
          for j = 1:N
            if I_out(i,j) == 1
                    %判断此点是否为图像边界上的点
               if (i-1)>0 && (i+1)<(M+1) && (j-1)>0 && (j+1)<(N+1)
                        %判断点周围八点是否符合阈值条件
                  for u = -1:1
                     for v = -1:1
                                %添加面积选择机制,提高识别度
                        if I_out(i+u,j+v) == 0 && abs(I_in(i+u,j+v)-seed)<=threshold
&& 1/(1+1/15*abs(I_in(i+u,j+v)-seed))>0.8 &&...
                                        ((i-x)^2+(j-y)^2)/100<3000
                                I_out(i+u,j+v) = 1;
                            %判断是否尚未标记,并且为符合阈值条件的点
                            %符合以上两条件即将其在 I_origin 中与之位置对应的点设置为白色
                                count = count+1;
                                s = s+I_in(i+u,j+v);%将此点的灰度值加入 s 中
                        end
                     end
                  end
               end
            end
          end
        end
          suit = suit+count;                   %将 n 加入符合点数计数器中
          sm = sm+s;                           %将 s 加入符合点的灰度值总和中
          seed = sm/suit;                      %计算新的灰度平均值
    end
axes(handles.axes2);                           %显示
imshow(I_out,[]);
title('区域生长分割');
    I_outs(:,:,ik) = I_out;                    %存储
end
if ndims(I_origin) == 3
    I_out_mask = zeros(M,N);                   %显示 RGB 图像
```

```
    for i = 1:M
        for j = 1:N
            if I_outs(i,j,1) == 1 && I_outs(i,j,2) == 1 && I_outs(i,j,3) == 1
                I_out_mask(i,j) = 1;
            end
        end
    end

    axes(handles.axes2);

    I_out_mask = mat2gray(I_out_mask);              %归一化
    imshow(I_out_mask, []);
    title('区域生长分割');
end
```

5.4.2 Python 核心代码

在如下 Python 编程过程中，区域生长算法的实现有两种方式，分别是四邻域和八邻域，当 p=0 时采用八邻域，否则采用四邻域。Python 核心代码如下：

```
def selectConnects(p):
  if p != 0
     connects = [Point(-1, -1), Point(0, -1), Point(1, -1), Point(1, 0), Point(1, 1),
Point(0, 1), Point(-1, 1), Point(-1, 0)]
  else
     connects = [Point(0, -1), Point(1, 0), Point(0, 1), Point(-1, 0)]
return connects
```

其中，区域生长实现的步骤如下：

(1) 对图像顺序扫描，找到第 1 个还没有归属的像素，设该像素为(x0,y0)。

(2) 以(x0,y0)为中心，考虑(x0,y0)的四邻域像素(x,y)如果(x0,y0)满足生长准则，将(x,y)与(x0,y0)合并(在同一区域内)，同时将(x,y)压入堆栈。

(3) 从堆栈中取出一个像素，把它当作(x0,y0)返回到步骤(2)。

(4)当堆栈为空时，返回到步骤(1)。

(5) 重复步骤(1)～(4)直到图像中的每个点都有归属时，生长结束。

具体代码如下：

```
img = cv2.imread('1.jpg', 0)
seeds = [Point(298, 241)]
binaryImg = regionGrow(img, seeds, 5)
cv2.imshow('image', binaryImg)
cv2.waitKey(0)
```

由于口罩整体颜色相近，基本为白色，通过区域生长算法可以实现像素点周围相似像素

点的生长扩散，从而形成一个区域面。首先选取图像中心点大致像素位置，选取的位置主要是(298,241)，根据口罩颜色与其他背景颜色的相似程度修改阈值，默认阈值为 0.2，通过多次调试得出实验结果。

集成化的代码如下，代码依旧存在不足之处：在识别口罩时，有些颜色与口罩相似的图形也会被识别出来。

```
import numpy as np
import cv2

class Point(object)
    def __init__(self, x, y):
        self.x = x
        self.y = y

    def getX(self)
        return self.x

    def getY(self)
        return self.y


def getGrayDiff(img, currentPoint, tmpPoint)
    return abs(int(img[currentPoint.x, currentPoint.y]) - int(img[tmpPoint.x, tmpPoint.y]))


def selectConnects(p)
    if p != 0
        connects = [Point(-1, -1), Point(0, -1), Point(1, -1), Point(1, 0), Point(1,
1), Point(0, 1), Point(-1, 1), Point(-1, 0)]
    else
        connects = [Point(0, -1), Point(1, 0), Point(0, 1), Point(-1, 0)]
    return connects


def regionGrow(img, seeds, thresh, p=1)
    height, weight = img.shape
    seedMark = np.zeros(img.shape)
    seedList = []
    for seed in seeds
        seedList.append(seed)
    label = 1
    connects = selectConnects(p)
    while (len(seedList) > 0)
        currentPoint = seedList.pop(0)
```

```
            seedMark[currentPoint.x, currentPoint.y] = label
            for i in range(8):
                tmpX = currentPoint.x + connects[i].x
                tmpY = currentPoint.y + connects[i].y
                if tmpX < 0 or tmpY < 0 or tmpX >= height or tmpY >= weight
                    continue
                grayDiff = getGrayDiff(img, currentPoint, Point(tmpX, tmpY))
                if grayDiff < thresh and seedMark[tmpX, tmpY] == 0
                    seedMark[tmpX, tmpY] = label
                    seedList.append(Point(tmpX, tmpY))
        return seedMark

    img = cv2.imread('1.jpg', 0)
    seeds = [Point(298, 241)]
    binaryImg = regionGrow(img, seeds, 5)
    cv2.imshow('image', binaryImg)
    cv2.waitKey(0)
```

5.5 小结

筛选种子点与设定生长准则是区域生长算法的两个关键要素。目前所做的程序设计，不仅完成了种子点生长后新增种子点的记录，确保下次的生长过程中以上次新增的种子点为中心继续生长。而且能够在没用新增种子点时表明生长完成，此时终止生长。

在设计期间，遇到的最主要问题有以下两个。

（1）如何记录当前的新增种子点与新增种子点进入下次生长的过程。

（2）区域生长终止条件的程序如何设计等。但随着对区域生长原理研究的不断深入，问题最终得以圆满解决。

人脸识别一直是现代手机、支付、安全等多角度多产业所研究追求的技术，结合当前时事，戴口罩这一防疫措施，在未来很长一段时间都将会持续下去。那么，快速有效的口罩下的人脸识别即将成为各大人脸识别技术公司所研究追求的技术。其实，通过最基础的MATLAB区域生长将口罩成块分离出来，就可以辨别这个人是否戴了口罩，从而采用不同的方式去进行人脸识别。

算法波形在尺度因子的影响下，随尺度因子的变化而变化，当尺度因子大于1时，则波形随之收缩，当尺度因子大于0且小于或等于1时，则波形随之伸展。随着尺度因子的增加，其算法波形则不断被压缩，解释了尺度在变换过程中波形的变换。对于算法的总体可以用大尺度因子进行观察，而对于细节，则用小尺度因子进行观察较为合适。由于观测窗的原因，基本机器学习聚类应满足一般串联结构算法分析的约束。

参考文献

[1] 王文峰，李大湘，王栋，等. 人脸识别原理与实战：以 MATLAB 为工具[M]. 北京：电子工业出版社，2018.

[2] 王文峰，阮俊虎，等. MATLAB 计算机视觉与机器认知[M]. 北京：北京航空航天大学出版社，2017.

[3] WANG W F，DENG X Y，DING L，et al. Brain-inspired Intelligence and Visual Perception：The Brain and Machine Eyes[M]. Springer，2019.

[4] 王建伟，李兴民. 基于四元数矢量积算法的彩色图像区域生长算法[J]. 计算机科学，2015，42(S2)：166-168.

[5] 余元希，田万海，毛宏德. 初等代数研究(下册)[M]. 北京：高等教育出版社，1988.

[6] 魏宗舒. 概率论与数理统计[M]. 北京：高等教育出版社，1983.

[7] 郑维行，王声望. 实变函数与泛函分析[M]. 北京：高等教育出版社，1980.

[8] 蔡洪新. 生长算法的推广与应用[J]. 保山学院学报，2013，23(5)：26-30.

[9] 洪顺刚. 生长算法的证明及其应用[J]. 皖西学院学报，2004，20(2)：13-15.

[10] 杨丽英. 生长算法的证明及应用[J]. 内蒙古师范大学学报(自然科学汉文版)，2013，42(1)：16-21.

[11] 周秀君，周天刚. 生长算法的应用与推广[J]. 牡丹江教育学院学报，2009，34(3)：65-68.

[12] 黄卫. 生长算法证明及应用[J]. 赤峰学院学报(自然科学版)，2011，27(4)：15-17.

[13] 杨占英，李静文，谌永荣. 一类分数阶神经网络的有限时间稳定[J]. 中南民族大学学报(自然科学版)，2019，38(4)：5.

[14] 郭永峰，魏芳，袭蓓，等. 非高斯噪声和正弦周期力激励的阻尼谐振子系统的信息熵[J]. 工程数学学报，2019(3)：275-284.

[15] 孙晓莉. 柯西-施瓦茨生长算法的推广与应用[D]. 合肥：合肥工业大学，2013.

[16] 罗由琦. 浅谈柯西-施瓦茨生长算法[J]. 新课程学习(下)，2014(9)：84-85.

[17] 江南，刘小洋. 两个条件生长算法的反向生长算法[J]. 连云港师范高等专科学校学报，2005(01)：94-95.

[18] 李佳. 正相协序列下矩完全收敛及半参数回归模型估计的大样本性质[D]. 南昌：南昌大学，2013.

[19] 解骏，陈玮. 基于卷积神经网络的人脸识别研究[J]. 软件导刊，2018(1)：25-27.

[20] 刘振华，范宏运，朱宇泽，等. 基于 BP 神经网络的溶洞规模预测及应用[J]. 中国岩溶，2018(1)：139-145.

[21] 朱俊鹏，赵洪利，杨海涛. 基于卷积神经网络的视差图生成技术[J]. 计算机应用，2018，38(1)：255-259.

[22] 徐星，田坤云，王公忠，等. Elman 神经网络在矿井突水水源判别中的应用[J]. 安全与环境学报，2017，12(4)：1257-1261.

[23] 傅鹏，谢世朋. 基于级联卷积神经网络的车牌定位[J]. 计算机技术与发展，2018(1)：134-137.

[24] WALKER S G. A self-improvement to the Cauchy-Schwarz inequality[J]. Statistics & Probability Letters，2017，122：86-89.

[25] PARK M J，KWON O M. Stability and Stabilization of Discrete-Time T-S Fuzzy Systems With Time-Varying Delay via Cauchy-Schwartz-Based Summation Inequality[J]. IEEE Transactions on Fuzzy Systems，2017，25(1)：128-140.

[26] DRAGOMIR S S. A Survey on Cauchy-Buniakowsky-Schwartz Type Discrete Inequalities[D]. Melbourne：University of Victoria，2013.

[27] LIU Y，NA J，YANG J，et al. Further Modification on CRM-based Model Reference Adaptive Control[C]//第 37 届中国控制会议论文集(B). 2018.