# 第5章 chapter 5

# 信息、信息表示及处理

我们经常说或听到“信息技术”、“21世纪是信息时代”、“计算机是人类社会进入信息时代的基础和重要标志”等耳熟能详的词语和语句。那么，到底什么是信息呢？一般来说，**信息**是客观存在的表现形式，是事物之间相互作用的媒介，是事物复杂性和差异性的反映。更有意义的是，信息是对人有用、能够影响人的行为的数据。信息可以是不精确的，可以是事实，也可以是谎言。香农(Shannon)也对信息下过定义：**信息是事物运动状态或存在方式的不确定性的表述**，即信息是确定性和非确定性、预期和非预期的组合。

## 5.1 信息论基础

信息是个很抽象的概念。我们常常说信息量很大，或者信息量较少，但却很难说清楚信息量到底有多少。通常人们通过各种消息获得信息，那么，每条消息带来的信息量是多少呢？即存在信息量如何度量的问题。比如一本五十万字的中文书到底有多少信息量？1948年，香农提出了信息熵的概念，解决了信息量的度量问题。一般来说，信息度量的尺度必须统一，有说服力，因此，需要遵循下面几条原则。

(1) 能度量任何消息，并与消息的种类无关。

(2) 度量方法应该与消息的重要程度无关。

(3) 消息中所含信息量和消息内容的不确定性有关。

在继续讨论之前，回顾一下人类对问题的认识过程。通常碰到一个问题时，开始时对问题毫无了解，对它的认识是不确定的。然后，人们会通过各种途径获得信息，逐渐消除不确定性。最后，经过不断地尝试，人们对这一问题会非常了解，此时，不确定性将会变得很小。如图5-1所示，人们对问题的认识是通过不断获得信息，消除对问题认识的不

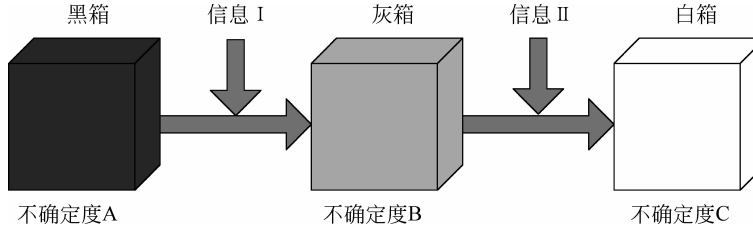


**图 5-1 问题的认识过程**

确定性的过程，是一个从黑盒到灰盒，最后变成白盒的过程。因此，启发人们尝试用消除不确定性的多少来度量信息。下面的例子展示了不确定性与信息度量之间的关系。

**例 5-1** 当你去大会堂找某个人时，甲告诉你两条消息：①此人不坐在前十排；②他也不坐在后十排。乙只告诉你一条消息：此人坐在第十五排。问谁提供的信息量大？

**分析**：乙虽然只提供了一条消息，但这一条消息对此人在什么位置上这一不确定性消除得更多，所以后者包含的信息量应比前者提供的两条消息所包含的总信息量更大。

**例 5-2** 假如在盛夏季节气象台突然预报“明天无雪”的消息。一般来说在明天是否下雪的问题上，根本不存在不确定性，所以这条消息包含的信息量为零。但是播报“明天有雪”的消息更令人惊讶，信息量更大。

通过对消息中不确定性消除的观察和分析，香农（美国贝尔实验室）应用概率论知识和逻辑方法推导出了信息量的计算公式。**事件的不确定程度可以用其出现的概率来描述**，消息出现的概率越小，则消息中包含的信息量就越大。

令 $P(x)$表示消息 $x$ 发生的概率，有 $0 \leqslant P(x) \leqslant 1$；令 $I$ 表示消息 $x$ 中所含的信息量，则 $P(x)$与 $I$ 的关系满足：

（1）$I$ 是 $P(x)$的函数：$I=I[P(x)]$。

（2）$I[P(x)]$是一个连续函数，即如果消息只有细微差别，则其包含的信息量也只有细微差别。

（3）$I[P(x)]$是一个严格递增函数。

（4）$P(x)$与 $I$ 成反比，即 $P(x)$增大则 $I$ 减小，$P(x)$减小则 $I$ 增大。

（5）$P(x)=1$ 时，$I=0$，即如果消息 $x$ 发生的概率为 1，并且我们被告知消息 $x$ 发生了，则我们没有获得任何信息；$P(x)=0$ 时，$I=\infty$。

**定义 5-1** **自信息量**是一个事件（消息）本身所包含的信息量，它是由事件的不确定性决定的，定义为

$$I(x) = \log_a \frac{1}{P(x)} = -\log_a P(x)$$

（1）若 $a=2$，信息量的单位称为比特（bit），可简记为 $b$。

（2）若 $a=\mathrm{e}$，信息量的单位称为奈特（nat）。

（3）若 $a=10$，信息量的单位称为哈特莱（Hartley）。

自信息量说明：

（1）事件 $x$ 发生以前，事件发生的不确定性的大小。

（2）当事件 $x$ 发生以后，事件 $x$ 所含或所能提供的信息量（在无噪情况下）。

自信息量是信源（或消息源，见图 5-2）发出某一具体消息所含有的信息量，发出的消息不同所含有的信息量不同。因此，自信息量不能用来表征整个信源的不确定度。通常用**平均自信息量**表征整个信源的不确定度，**平均自信息量**指的是事件集（用随机变量表示）所包含的平均信息量，它表示信源的平均不确定性，又称为**信息熵**或**信源熵**，简称为**熵**。香农信息论的开创性想法，为一个消息源赋予了一定的信息熵。简单地说，假设 $S$ 为一个信源，它能发出的消息来自于集合 $x_1$、$x_2$、…、$x_n$，$S$ 发出消息 $x_1$、$x_2$、…、$x_n$的概率分别为 $p_1$、$p_2$、…、$p_n$，其中 $p_i \geqslant 0$，并且有 $\sum_{i=1}^{n} p_i = 1$。则根据自信息量公式，$S$ 发出消息 $x_i$

时，接收端可以获得 $I(p_i)=-\log_2 p_i$ 位的信息量，则每个消息 $x_i$ 包含的平均信息量为

$$H(x)=\sum_{i=1}^{n} p_i I(p_i)=-\sum_{i=1}^{n} p_i \log_2 p_i \quad \text{（比特）}$$

$H(S)$ 称为信源 $S$ 的熵①。信源的熵可以指信源输出后，消息所提供的平均信息量；也可以指信源输出前，信源的平均不确定性；或信息的随机性。

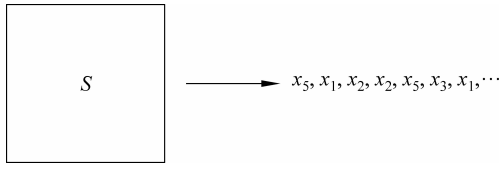


**图 5-2　一个信源**

根据上面的定义和信息熵的公式，可以对日常生活中各类现象包含的信息量进行度量，以下几个例子说明了如何进行度量。

**例 5-3**　投掷一枚骰子的结果有 6 种，即出现 1～6 点，且出现每种情况的概率均为 1/6，故熵 $H=-\sum_{i=1}^{6}\frac{1}{6}\log_2\frac{1}{6}=\log_2 6\approx 2.585$（比特）。

**例 5-4**　抛一枚硬币的结果为正、反面两种，出现的概率均为 1/2，故熵 $H=-\sum_{i=1}^{2}\frac{1}{2}\log_2\frac{1}{2}=\log_2 2=1$（比特）。

**例 5-5**　向石块上猛摔一只鸡蛋，其结果必然是将鸡蛋摔破，出现的概率为 1，故熵 $H=\log_2 1=0$（比特）。

**例 5-6**　某离散信源由 0、1、2 和 3 四个符号组成，它们出现的概率分别为 3/8、1/4、1/4 和 1/8，且每个符号的出现都是独立的。试求某消息“20102013021300120321010032101002310200201031203210012021 0”的信息量。

**解**：信源的平均信息量为 $H=-\frac{3}{8}\log_2\frac{3}{8}-\frac{1}{4}\log_2\frac{1}{4}-\frac{1}{4}\log_2\frac{1}{4}-\frac{1}{8}\log_2\frac{1}{8}=1.906$（比特/符号）。因此，这条消息的信息量为 $I=57\times 1.906=108.64$（比特）。

从上述例子可以看出，香农利用信息的熵回答了消息的信息量的问题：**即任一消息的信息量由用于传输该消息的 1 和 0 的数量构成**。

## 5.2　信息的数字化

在 5.1 节中一直用 2 为底的对数求信息熵，得到的信息量的单位为 bit（比特，又称为**位**），而位这个单位，恰好是二进制信息表示的基本单位，又称为二进制位。在香农著名

① 假设信源 $S$ 是离散无记忆信源，即消息 $x_i$ 之间是独立同分布的。关于更多信息论的论述，有兴趣的读者可参考 Thomas M. Cover and Joy A. Thomas. 2006. *Elements of Information* Theory 2nd *Ed. Wiley-Interscience*，机械工业出版社出版了中文版。

的论文《通信的数学理论》(*A Mathematical Theory of Communication*)第1页中该术语第一次被正式使用。香农在其1938年的硕士论文《继电器与开关电路的符号分析》(*A Symbolic Analysis of Relay and Switching Circuits*)中,探讨了如何用硬件电路来实现二进制的运算,奠定了数字电路的理论基础,被评论为"这可能是20世纪最重要、最著名的一篇硕士论文"。香农也因此被称为信息论和数字电路设计理论之父。

二进制是计算机信息表示的基础,计算机中用0和1构成的字串来表示各种信息。将声、光、电、磁等信号及语言、图像、报文等信息转变成为0和1字串编码后进行处理、存储、传递,称为信息的**数字化**。

### 5.2.1 数值的数字化

一个数值能够用不同进制表示,这些表示之间存在转换关系。计算机使用二进制表示数值,而人类惯用十进制。当将数值输入到计算机中时,必须将十进制转换为二进制,而将计算机中的数值输出时,一般要将其转换为十进制,以便于人阅读和理解。对整数而言,虽然进制不同,但是一个数的不同进制表示在数值上是相等的,因此有

$$(N)_{10} = a_n \times 2^n + \cdots + a_1 \times 2^1 + a_0 \times 2^0 \tag{5-1}$$

上式等号左边下标10表示用十进制表示整数$N$。由上式可得,将二进制整数转换为十进制整数,可直接按照等号右边的式子,做十进制的乘法和加法就能完成。例如,二进制整数$(10111)_2$可按照上式转换为十进制整数:$1\times2^4+0\times2^3+1\times2^2+1\times2^1+1\times2^0=(23)_{10}$。

而十进制整数到二进制整数的转换可采用"除2取余"法,其方法也可由上式推导出来。$N$代表给定的十进制整数,$a_n$、…、$a_1$、$a_0$分别代表需要求出的各位二进制数字。式(5-1)等号两边同时除以2,等式保持不变。从等式右边可看出,$N$除以2得到的余数是$a_0$,得到的商为$a_n\times2^{n-1}+\cdots+a_2\times2^1+a_1$,对商再除以2,又得余数$a_1$和商$a_n\times2^{n-2}+\cdots+a_3\times2^1+a_2$,等等,依此进行下去,直到商为0。这个过程中得到的所有余数或为0或为1,将它们按照求得的顺序的反序拼接在一起,就得到所需要的二进制表示形式。图5-3给出了将$(37)_{10}$转换成二进制的过程,可知$(37)_{10}=(100101)_2$。

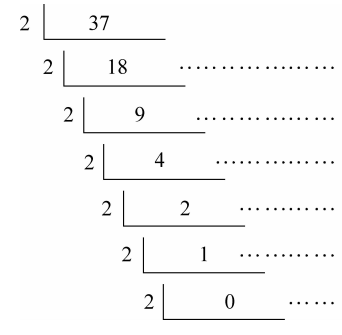


**图5-3 十进制整数转换为二进制整数示例**

二进制小数与十进制小数之间的转换方法也能通过公式推导出来。如式(5-2)所示,其中$0.a_{-1}a_{-2}\cdots a_{-m}$是二进制小数,$m$可能为无穷大,$N$是等价的十进制小数。

$$(N)_{10} = a_{-1} \times 2^{-1} + a_{-2} \times 2^{-2} + \cdots + a_{-m} \times 2^{-m} \tag{5-2}$$

同样地,将二进制小数转换为十进制小数,可直接按照等号右边的式子,做十进制的除法和加法即可。例如,已知$(0.1011)_2$,求其等价的十进制小数,转换过程为$1\times2^{-1}+0\times2^{-2}+1\times2^{-3}+1\times2^{-4}=(0.6875)_{10}$。

十进制小数到二进制小数的转换可采用"乘2取整"法,公式(5-2)两边同时乘以2,等式仍成立,此时,右边整数部分变成$a_{-1}$,小数部分变为$a_{-2}\times2^{-1}+a_{-3}\times2^{-2}+\cdots+$

$a_{-m}\times 2^{-m+1}$。再对结果的小数部分两边乘以 2，右边整数部分变成 $a_{-2}$，小数部分为 $a_{-3}\times 2^{-1}+\cdots+a_{-m}\times 2^{-m+2}$，等等，依此进行下去，直到乘 2 的结果中小数部分为 0，或者达到所需要的二进制位数。对于很多十进制小数，上述乘 2 的过程，达不到结果小数部分为 0 的情形。因此，十进制小数到二进制小数的转换是**不精确**的转换。图 5-4 给出了十进制小数到二进制小数转换的两个例子，左边是将十进制数 0.625 转换成二进制数，它是精确转换；右边是将十进制数 0.34 转换成二进制数，它是不精确转换。从图中可得，$(0.625)_{10}=(0.101)_2$，$(0.34)_{10}\approx(0.010101)_2$。

| 0. | 625 | (×2 |
|---|---|---|
| 1. | 25 | |
| 0. | 5 | |
| 1. | 0 | |

| 0. | 34 | (×2 |
|---|---|---|
| 0. | 68 | |
| 1. | 36 | |
| 0. | 72 | |
| 1. | 44 | |
| 0. | 88 | |
| 1. | 76 | |

**图 5-4　十进制小数转换为二进制小数示例**

从图 5-4 的例子还可知道，用二进制来表示十进制数时，有些数不能精确表示，只能在表示能力范围内给出近似表示。这与可用的二进制位数和十进制数自身相关。

(1) 可用的二进制位数是由计算机的能力决定的，对于 32 位计算机，可以用来表示数的位数通常有 8、16 和 32 位，位数越多，能表示的数值越多，精度也越高。

(2) 对不能精确转化为二进制的十进制小数，即便能用的二进制位数很多，也无法精确地表示出来。例如，0.1 这个十进制小数转换成二进制小数是一个无限循环小数，理论上就无法精确表示。

除了表示数值，还需表示数值的符号：＋或－。通常用二进制表示的最高位来表示数值的符号位，0 表示正数，1 表示负数。因此，如果能用的二进制位为 8 位，那么，$(23)_{10}=(00010111)_2$，而$(-23)_{10}=(10010111)_2$，这种表示方式称为**原码**表示。

计算机中通常用**浮点数**来表示实数，浮点数是指有理数中某特定子集的数的数字表示，在计算机中用以近似表示任意的某个实数。浮点计算是指浮点数参与的运算，这种运算通常伴随着因为无法精确表示而进行的近似或舍入。观察下面的 Python 语句：

```
>>>print(0.1)
0.1
>>>print("%.17lf"%0.1)
0.10000000000000001
```

0.1 这个十进制小数转换成二进制数时，是一个无限循环小数，在计算机中通常用浮点数表示，是一种近似的表示。而现代程序设计语言的输出带有一定的智能，在保证误

差较小的前提下会自动舍入。所以第一个 print 语句打印 0.1。但是，当用第 2 个 print 语句指定输出精度时，就能看到 0.1 在计算机中不是真正的 0.1，而是有一定误差的。同理，浮点数之间用>、<、==来比较大小是不可取的，需要看两个浮点数是否在合理的误差范围，如果误差合理，即认为相等；否则，两个在十进制中相等的数可能在计算机中是不相等的。

此外，浮点数的误差会在其计算过程中累积，观察下面的 Python 程序：

```
x=0.0
for i in range(100):
    x+=0.1
print("%.17lf"%x)
print(x)
```

运行该程序得到的输出如下所示：第 1 行是 x 的较为精确的表示，而第 2 行是 print 自动舍入，显示出来的看似正确的结果。

```
9.99999999999998046
10.0
```

## 5.2.2 字符的数字化

字符信息是最基本的信息类型之一。一个字符是指独立存在的一个符号，诸如汉字、大小写形式的英文字母、日文的假名、数字和标点符号等。还有一类所谓**控制字符**，用于通信、人机交互等方面，起控制作用，如“回车符”、“换行符”等。

在人类文明发展的过程中，发明了各种各样的符号体系，用来表征事物，交流思想。其中典型的符号体系是人类所使用的语言。在一个符号体系中，存在一组基本符号，它们可构成更大的语言单位。这组基本符号的数目一般比较小。如英语的字母，用之可构成英文的单词。英文单词有成千上万个，但英文字母加起来仅有 52 个(区分大小写)。例外的情况是汉语，汉语中可由汉字构成有意义的词或词组，但汉字数目比较大。

计算机内部用二进制对字符对象进行编码。对于任意一个字符对象集合，不同的人都可设计自己的编码体系。但是为了减少编码体系之间转换的复杂性，提高处理效率，相关组织发布了标准编码方案，以便信息交换和共享。如英文字符的 ASCII 编码、中国国家标准汉字编码和 Unicode 编码等。以 ASCII 码[①]为例，ASCII 码中所含字符个数不超过 128(见图 5-5)，其中包含控制符、通信专用字符、十进制数字符号、大小写英文字母、运算符和标点符号等。打印出来时，控制字符和通信专用字符是**不可见的**，不占介质空间，它们指明某种处理动作。其他字符是可见的，因而称为**可视字符**。

① ASCII 码(American Standard Code for Information Interchange)是美国国家标准化学会(American National Standards Institute，ANSI)维护和发布的用于信息交换的字符编码。

| | 0000 | 0001 | 0010 | 0011 | 0100 | 0101 | 0110 | 0011 |
|---|---|---|---|---|---|---|---|---|
| 0000 | NUL | DLE | SP | 0 | @ | P | ' | p |
| 0001 | SOH | DC1 | ! | 1 | A | Q | a | q |
| 0010 | STX | DC2 | " | 2 | B | R | b | r |
| 0011 | ETX | DC3 | # | 3 | C | S | c | s |
| 0100 | EOT | DC4 | $ | 4 | D | T | d | t |
| 0101 | ENQ | NAK | % | 5 | E | U | e | u |
| 0110 | ACK | SYN | & | 6 | F | V | f | v |
| 0111 | BEL | ETB | ' | 7 | G | W | g | w |
| 1000 | BS | CAN | ) | 8 | H | X | h | x |
| 1001 | HT | EM | ( | 9 | I | Y | i | y |
| 1010 | LF | SUB | * | : | J | Z | j | z |
| 1011 | VT | EAC | + | ; | K | [ | k | { |
| 1100 | FF | ES | ' | < | L | \ | l | \| |
| 1101 | CR | GS | – | = | M | ] | m | } |
| 1110 | SO | RS | . | > | N | ^ | n | ~ |
| 1111 | SI | US | / | ? | O | _ | o | DEL |

**图 5-5　ASCII 码编码表**

一个 ASCII 码由八位二进制(1B)组成，实际使用低七位，最高位恒为 0。因此，ASCII 码中的字符个数不能超过 128 个。八位二进制能够编码 256 个符号，有一半编码空置，这主要是为以后的应用留下扩展空间，或最高位留作它用。图 5-5 中，第一行列出编码中高四位，第一列给出低四位。一个字符所在行列的高四位编码和低四位编码组合起来，即为该字符的**编码**。例如，大写字母 A 的编码为 01000001，十进制为 65；数字符号 0 的编码为 00110000，十进制为 48。从这里可以看出，数字符的 ASCII 码与它所代表的数值是完全不同的两个概念。分析 ASCII 码表，可看出其中常见编码的大小规则，即 0～9＜A～Z＜a～z。数字符 0 的编码比数字符 9 的编码小，并按 0 到 9 顺序递增，如“5”＜“8”；数字符编码小于英文字母编码，如“9”＜“A”；字母 A 的编码比字母 Z 的编码小，并按 A 到 Z 顺序递增。如“A”＜“Z”；同一个英文字母，其大写形式的编码比小写形式的编码小 32，如“a”－“A”＝32。

Python 语言内置函数 chr 和 ord 提供了字符及其 ASCII 码之间的转换功能，如下：

```
>>>chr(97)
'a'
>>>ord('a')
97
>>>ord('a')-ord('A')
32
```

### 5.2.3 声音的数字化

声音是由物体振动引发的一种物理现象，声源是一个振荡源，它使周围的介质（如空气、水等）产生振动，并以波的形式进行传播。声音是随时间连续变化的物理量，可以近似地看成是一种周期性的函数。如图 5-6 所示，它可用 3 个物理量来描述。

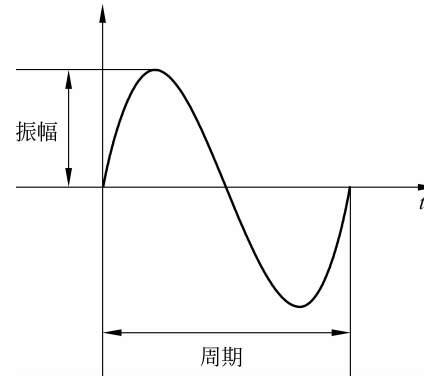


图 5-6 声音波形示例

（1）振幅：即波形最高点（或最低点）与基线的距离，它表示声音的强弱。

（2）周期：即两个相邻波峰之间的时间长度。

（3）频率：即每秒钟振动的次数，以 Hz 为单位。

计算机只能处理离散量，只有数字化形式的离散信息才能被接收和处理。因此，对连续的模拟声音信号必须先进行数字化离散处理，转换为计算机能识别的二进制表示的数字信号，才能对其进行进一步的加工处理。用一系列数字来表示声音信号，称为**数字音频**。

把模拟的声音信号转换为数字音频的过程称为**声音的数字化**。这个过程（见图 5-7）包括采样、量化和编码 3 个步骤。

（1）**采样**：每隔一个时间间隔测量一次声音信号的幅值，这个过程称为采样，测量到的每个数值称为**样本**，这个时间间隔称为**采样周期**。这样就得到了一个时间段内的有限个幅值。

（2）**量化**：采样后得到的每个幅度的数值在理论上可能是无穷多个，而计算机只能表示有限精度。因此，还要将声音信号的幅度取值的数量加以限制，这个过程称为量化。例如，假设所有采样值可能出现的取值范围在 0～1.5 之间，而实际只记录了有限个幅值：0、0.1、0.2、0.3、…、1.4、1.5 共 16 个值，那么如果采样得到的幅值是 0.4632，则近似地用 0.5 表示；如果采样得到的幅值是 1.4167，就取其近似值 1.4。

（3）**编码**：将量化后的幅度值用二进制形式表示，这个过程称为编码。对于有限个幅值，可以用有限位的二进制数来表示。例如，可以将上述量化中所限定的 16 个幅值分别用 4 位二进制数 0000～1111 来表示，这样声音的模拟信号就转化为了数字音频。

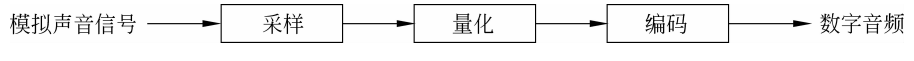


图 5-7 声音的数字化过程

图 5-8 给出了一个模拟声音信号数字化过程的示例。在横坐标上，$t_1$～$t_{20}$ 为采样的时间点，纵坐标上假定幅值的范围在 0～1.5，并且将幅值量化为 16 个等级，然后对每个等级用 4 位二进制数进行编码。例如，在 $t_1$ 采样点，它的采样值为 0.335，量化后取值为 0.3，编码就用 0011 表示。

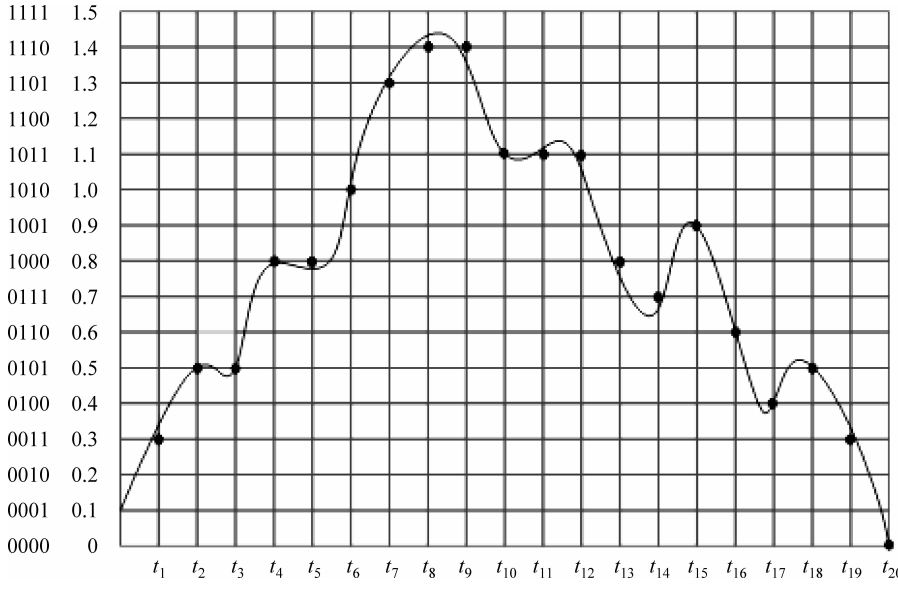


**图 5-8　声音信号的采样、量化、编码示例**

## 5.2.4　图像的数字化

图像可以看作是由二维平面上无穷多个点构成的，每个点通过各种方式呈现出颜色，被人眼感知后在大脑中留下印象，就成为了图像。可以通过各种方式记录图像，如胶片就是使用光学透镜系统在胶片上记录下现实世界的自然景物。这样记录下来的图像中，胶片上任何两点之间都会有无穷多个点，图像颜色的变化也会有无穷多个值。这种在二维空间中位置和颜色都是连续变化的图像称为**连续图像**。用计算机进行图像处理首先要把这种连续图像转换成计算机能够记录和处理的**数字图像**，这个过程就是**图像的数字化**过程。图像的数字化，就是按一定的空间间隔自左到右、自上而下提取画面信息，并按一定的精度进行量化的过程。和声音数字化类似，图像的数字化也要经过采样、量化和编码这 3 个步骤。

（1）**采样**：对二维空间上连续的图像在水平和垂直方向上等间距地分割成矩形网状结构，所形成的微小方格称为**像素点**，一副图像就被采样成有限个像素点构成的集合，如图 5-9 右边所示。左边是要采样的连续图像，右边是采样后的图像，每个小格即为一个像素点。

（2）**量化**：采样后的每个像素的取值仍然是连续的，因为颜色的取值可能是无穷多个颜色中的任何一个，因此要对颜色进行离散化处理。为了把颜色取值离散化，要将颜色取值限定在有限个取值范围内，这称为量化。量化的结果是图像能够容纳的颜色总数，它反映了采样的质量。例如，如果以 4 位存储一个点，就表示图像只能有 16 种颜色；若采用 16 位存储一个点，则有 $2^{16}=65\ 536$ 种颜色。

（3）**编码**：将量化后每个像素的颜色用不同的二进制编码表示，于是就得到 $M\times N$

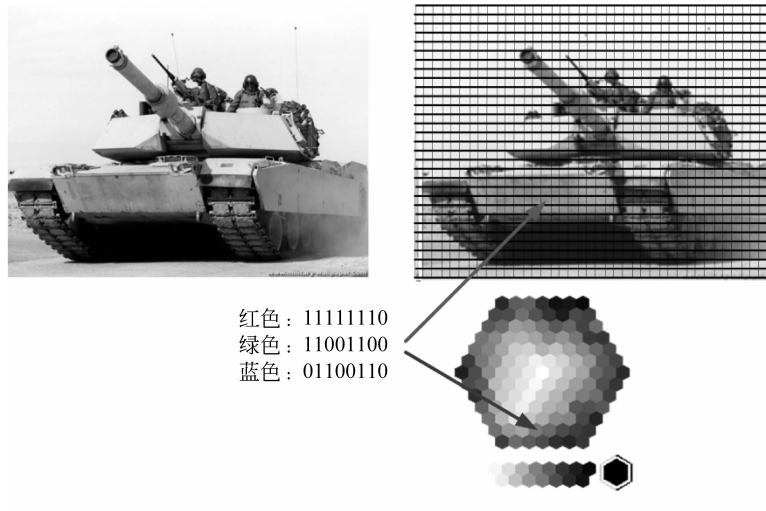


**图 5-9 图像数字化示例**

的数值矩阵，把这些编码数据一行一行地存放到文件中，就构成了数字图像文件的数据部分。

图 5-9 给出了一个图像数字化的例子。原始图像如图左边所示，采样过程可想象将一张同尺寸的网格覆盖于原图像上，每个格子即为一个像素。假设采用 24 位表示像素颜色，则颜色被限定为 $2^{24}=16\ 777\ 216$ 种。图中给出了某个点的颜色编码（土黄色）。此处采用 RGB 颜色模型。RGB 是 Red（红）、Green（绿）和 Blue（蓝）的缩写。采用红、绿、蓝 3 种颜色的不同比例的混合来产生颜色的模型称为 **RGB 模型**。某一种颜色和这 3 种基本颜色的关系可以用下面的式子来描述：颜色＝R（红色的百分比）＋G（绿色的百分比）＋B（蓝色的百分比）。

计算机中的数字图像可分为两大类：一类是**位图图像**，另一类是**矢量图形**。前面介绍的在空间和颜色上都离散化的图像称为位图图像，简称位图或图像。像素是组成位图最基本的元素，每个像素用若干个二进制位来描述。矢量图形用一组计算机绘图指令来描述和记录一幅图，显示时，按照绘制的过程逐一地显示。由于矢量图形文件并不保存每个像素的颜色，而是包含了计算机创建这些对象的形状、尺寸、位置和色彩等的指令，因此，文件的存储容量很小。

## 5.3 数据压缩

信息数字化后，需要占用存储空间，如果要将信息通过网络从一处发往另一处，还将花费传输时间。例如，声音和图像信息，质量的提高带来的是数据量的急剧增加，给存储和传输造成极大的困难。为了节约时间和空间，**数据压缩**是一个行之有效的方法。数据压缩是指对原始数据进行重新编码，去除原始数据中冗余数据的过程。将压缩数据还原为原始数据的过程称为**解压缩**。压缩方法可能是**无损的**和**有损的**。用无损的压缩方法，