9 Transcendental Functions 超越函数

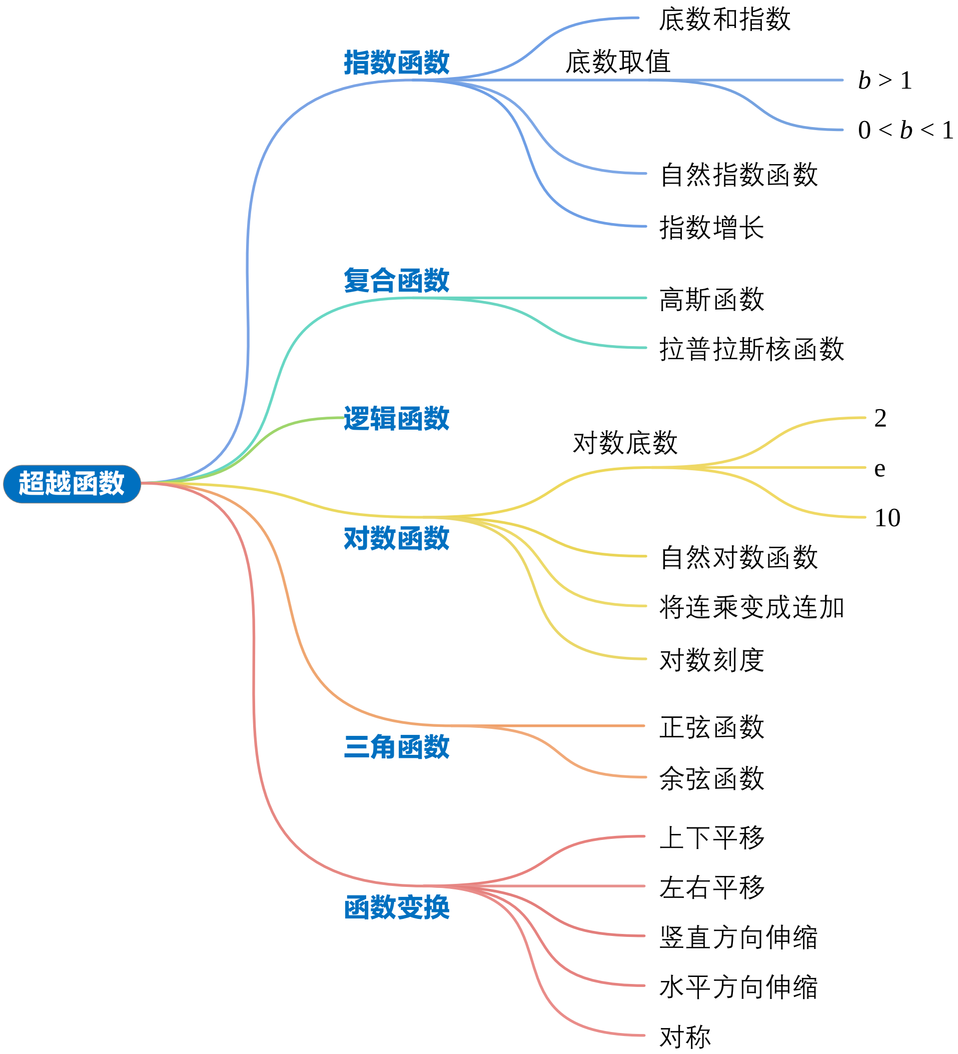
超出代数函数范围的函数

整个科学只不过是日常思维的提炼。

The whole of science is nothing more than a refinement of everyday thinking.

—— 阿尔伯特·爱因斯坦 (Albert Einstein) | 理论物理学家 | 1879 ~ 1955

- `matplotlib.pyplot.axhline()` 绘制水平线
- `matplotlib.pyplot.axvline()` 绘制竖直线
- `matplotlib.pyplot.contour()` 绘制平面等高线
- `matplotlib.pyplot.contourf()` 绘制平面填充等高线
- `matplotlib.pyplot.grid()` 绘制网格
- `matplotlib.pyplot.plot()` 绘制线图
- `matplotlib.pyplot.show()` 显示图片
- `matplotlib.pyplot.xlabel()` 设定 x 轴标题
- `matplotlib.pyplot.ylabel()` 设定 y 轴标题
- `numpy.absolute()` 计算绝对值
- `numpy.array()` 创建 array 数据类型
- `numpy.cbrt()` 计算立方根
- `numpy.ceil()` 计算向上取整
- `numpy.cos()` 计算余弦
- `numpy.floor()` 计算向下取整
- `numpy.linspace()`产生连续均匀向量数值
- `numpy.log()` 底数为 e 自然对数函数
- `numpy.log10()` 底数为 10 对数函数
- `numpy.log2()` 底数为 2 对数函数
- `numpy.meshgrid()` 创建网格化数据
- `numpy.power()` 乘幂运算
- `numpy.sin()` 计算正弦
- `numpy.sqrt()` 计算平方根


超越函数
指数函数
底数和指数
底数取值
$b > 1$
$0 < b < 1$
自然指数函数
指数增长
复合函数
高斯函数
拉普拉斯核函数
逻辑函数
对数函数
对数底数
2
e
10
自然对数函数
将连乘变成连加
对数刻度
三角函数
正弦函数
余弦函数
函数变换
上下平移
左右平移
竖直方向伸缩
水平方向伸缩
对称

9.1 指数函数：指数为自变量

指数函数 (exponential function) 的一般形式为。

$$f(x) = b^x \tag{1}$$

其中，b 为**底数** (base)，自变量 x 为**指数** (exponent)。上一章介绍的幂函数，自变量为底数；而指数函数的自变量为指数。

图 1 所示为当底数取不同值时指数函数图像，这几条曲线都经过 (0, 1)。注意区分底数 $b > 1$ 和 $0 < b < 1$ 两种情况对应的指数函数图像。

绘图时，可以用 `numpy.linspace()` 产生 `x` 数据，然后用 `b**x` 或 `numpy.power(b, x)` 计算指数函数值 b^x。

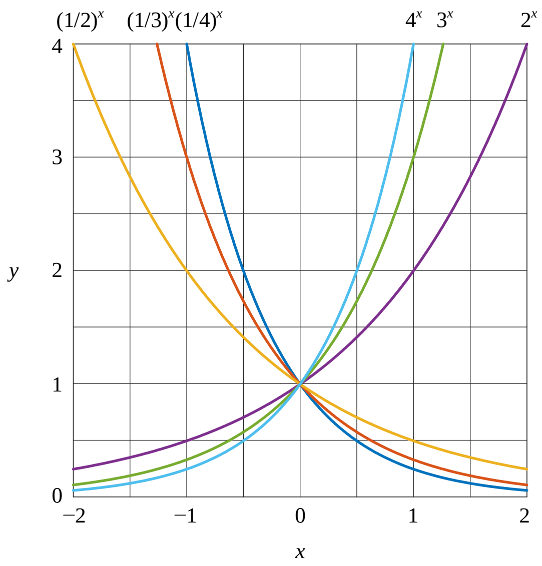


图 1. 不同底数的指数函数

更多情况，指数函数指的是**自然指数函数** (natural exponential function)。

$$f(x) = \mathrm{e}^x = \exp(x) \tag{2}$$

“自然”指的是自然常数 e 为底，$\mathrm{e} \approx 2.718$。

(1) 可以转化成以 e 为底数的函数。

$$y = f(x) = b^x = \mathrm{e}^{\ln b \cdot x} = \exp(\ln b \cdot x) \tag{3}$$

表 1. 用英文读指数

数学表达	英文表达
e^x	e raised to the *x*th power e to the *x* e to the power of *x* exponential of *x* exponential *x*
$y = e^x$	*y* equals (is equal to) exponential *x*
$y = b^x$	*y* equals (is equal to) *b* to the *x* *y* equals (is equal to) *b* raised to the power of *x*
e^{x+y}	e to the quantity *x* plus *y* power e raised to the power of *x* plus *y*
$e^x + y$	the sum of e to the *x* and *y* e to the *x* power plus *y*
$e^x e^y$	the product of e to the *x* power and e to the *y* power
$e^x y$	the product of e to the *x* power and *y* e raised to the *x* power times *y*

指数增长 (exponential growth) 就是用指数函数来表达。

$$G(t) = (1 + r)^t \tag{4}$$

其中，r 为年化增长率，t 是年限。

当增长率 r 取不同值时，指数增长和年限对应的关系如图 2 所示。图中**翻倍时间** (doubling time) 指的是当增长翻倍时所用的时间。图 2 中平行于横轴的虚线就是增长翻倍所对应的高度。

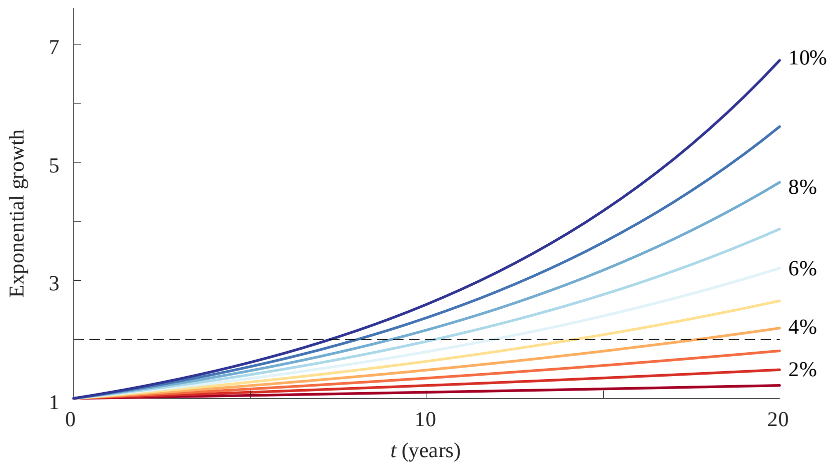


图 2. 指数增长

9.2 对数函数：把连乘变成连加

对数函数 (logarithmic function) 解析式如下。

$$y = f(x) = \log_b(x) \tag{5}$$

其中，b 为**对数底数** (logarithmic base)。对数函数的定义域为 $x > 0$。

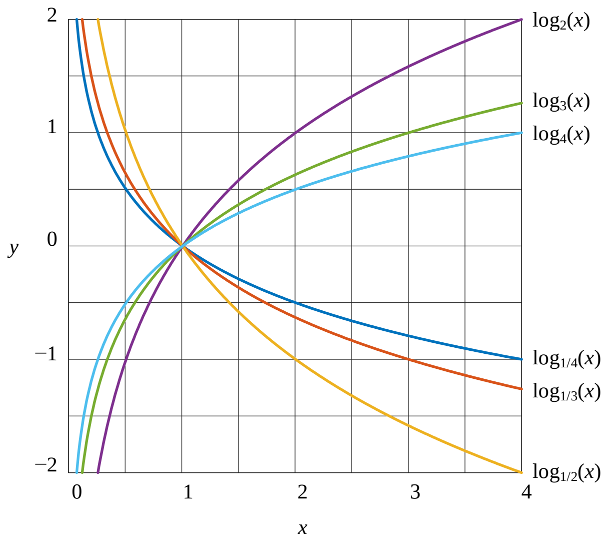


图 3. 不同底数的对数函数

图 4 中红色曲线为自然指数函数；图 4 中蓝色曲线为**自然对数函数** (natural logarithmic function)，函数如下。

$$y = f(x) = \ln(x) = \log_e(x) \tag{6}$$

对数函数是指数函数的反函数，也就是对数函数和指数函数互为逆函数；因此，图 4 中红色和蓝色曲线关于图中划线对称。

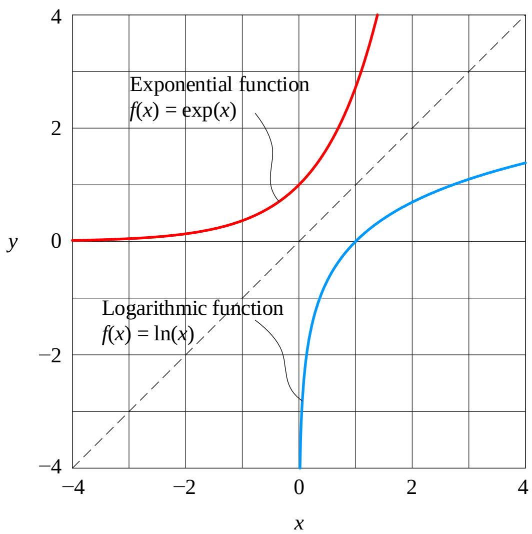


图 4. 自然对数函数和自然指数函数

Numpy 提供三个特殊底数对数函数运算，

◀ 底数为 2 对数函数 $\log_2(x)$，函数为 `numpy.log2()`

◀ 底数为 e 自然对数函数 $\ln(x)$，函数为 `numpy.log()`

◀ 底数为 10 对数函数 $\log_{10}(x)$，函数为 `numpy.log10()`

其他底数对数函数运算可以利用下述公式完成。

$$\log_b x = \frac{\ln(x)}{\ln(b)} \tag{7}$$

请读者注意以下几个对数运算规则。

$$\begin{aligned}
&\log_b a = \frac{\log_k x}{\log_k b} \\
&\log_b x = \frac{\log_{10} x}{\log_{10} b} = \frac{\log_e x}{\log_e b}. \\
&\log_{b^n} x^m = \frac{m}{n}\log_b x \\
&x = b^{\log_b(x)} \\
&x^{\log_b(y)} = y^{\log_b(x)}
\end{aligned} \tag{8}$$

对数的一个重要的性质是，把连乘变成连加。

$$\log_b(xyz) = \log_b x + \log_b y + \log_b z \tag{9}$$

连乘不容易求偏导，而对连加求偏导则容易很多；特别地，高斯函数存在 exp() 项，ln() 可以把指数项也变成求和形式。而且 ln() 不改变单调性。

在概率计算中，概率累积会出现数值非常小的情况，比如 1e-30 (10^{-30})，由于计算机的精度是有限的，无法识别这一类数据，取对数之后，更易于计算机的识别；因为，1e-30 以 10 为底取对数后得到−30)。

对数刻度 (logarithmic scale 或 logarithmic axis) 是一种**非线性刻度** (nonlinear scale)，用来描述较大的数值。

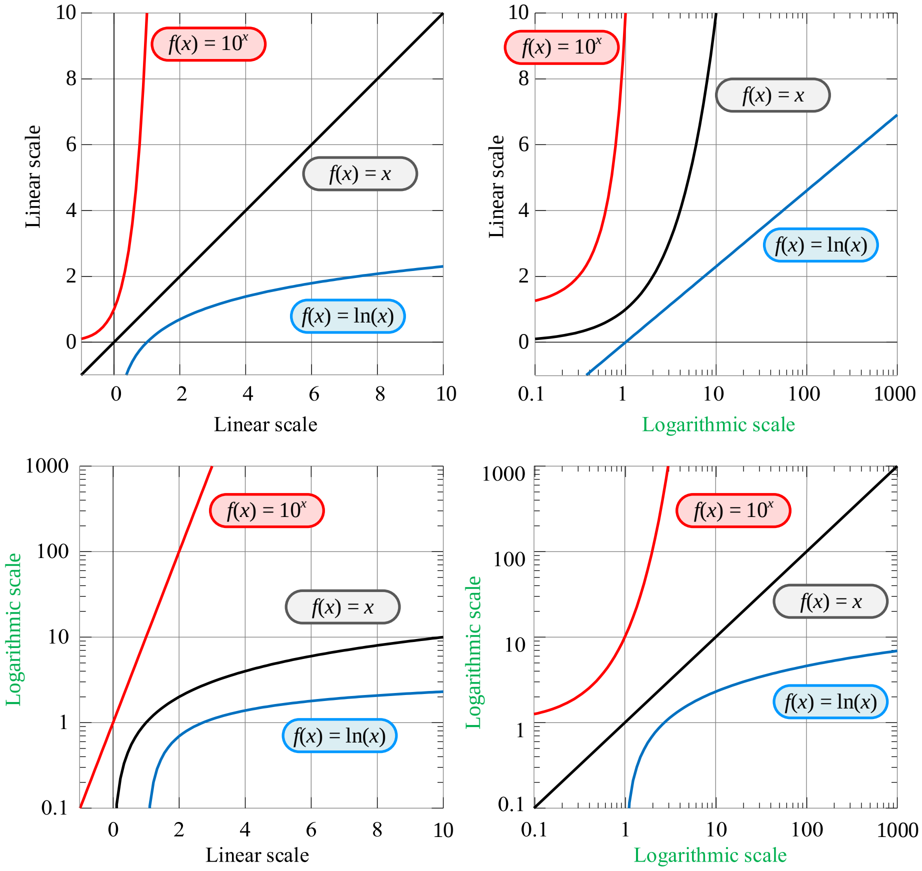


图 5. 几种对数刻度

表 2. 用英文读对数

数学表达	英文表达
$\log_4 x$	logarithm of x with base four
$y = \log_a x$	y is the logarithm of x to the base a y is equal to log base a of x
$\ln y$	log y to the base e log to the base e of y natural log (of) y
$\log_2 8 = 3$	the log base 2 of 8 is equal to 3 the logarithm of 8 with base 2 is 3 log base 2 of 8 is 3
$\log_4 16 = 2$	The log base 4 of 16 is equal to 2
$\log_2 \frac{1}{8} = -3$	the log base 2 of 1/8 is equal to −3.
$\log_6 108 = 3$	the logarithm of 108 to the base 6 is 3, 3 is the logarithm of 108 to the base 6

以下代码绘制图 5。

```python
# Bk_Ch9_01

import numpy as np
import matplotlib.pyplot as plt

x1 = np.linspace(0.1, 10, 100)
x = np.linspace(0.1, 1000, 100)

# x log scale
f1 = 10**x1
f2 = x1
f3 = np.log(x)

fig, ax = plt.subplots()
plt.plot(x1,f1,color = 'r')
plt.plot(x1,f2,color = 'k')
plt.plot(x,f3,color = 'b')

plt.xscale("log")
plt.ylim((-0.5,10))
plt.grid()
plt.tight_layout()
ax.set_box_aspect(1)

# y log scale
f1 = 10**x1
f2 = x1
f3 = np.log(x1)

fig, ax = plt.subplots()
plt.plot(x1,f1,color = 'r')
plt.plot(x1,f2,color = 'k')
plt.plot(x1,f3,color = 'b')

plt.yscale("log")
plt.ylim((0.1,1000))
plt.grid()
plt.tight_layout()
ax.set_box_aspect(1)


# x and y log scale
x_log = np.logspace(np.log(0.1), np.log(1000), num=100,
                    endpoint=True, base=10.0)
f1 = 10**x_log
f2 = x_log
f3 = np.log(x_log)

fig, ax = plt.subplots()
plt.plot(x_log,f1,color = 'r')
plt.plot(x_log,f2,color = 'k')
plt.plot(x_log,f3,color = 'b')

plt.yscale("log")
plt.xscale("log")
plt.ylim((0.1,1000))
plt.xlim((0.1,1000))
plt.grid()
plt.tight_layout()
ax.set_box_aspect(1)
```

9.3 高斯函数：高斯分布之基础

复合函数 (function composition) 通俗地说就是函数套函数，是把几个简单的函数复合得到一个较为复杂的函数。

指数函数经常和其他函数构造复合函数。比如，指数函数复合二次函数，得到**高斯函数** (Gaussian function)。

$$f(x) = \exp\left(-\gamma x^2\right) \tag{10}$$

其中，γ 为参数，$\gamma > 0$。高斯函数的定义域是 $(-\infty, +\infty)$，而值域是 $(0, 1]$。高斯函数无限接近 0，却不到达 0。

图 6 (a) 所示为 γ 决定高斯函数的形状。

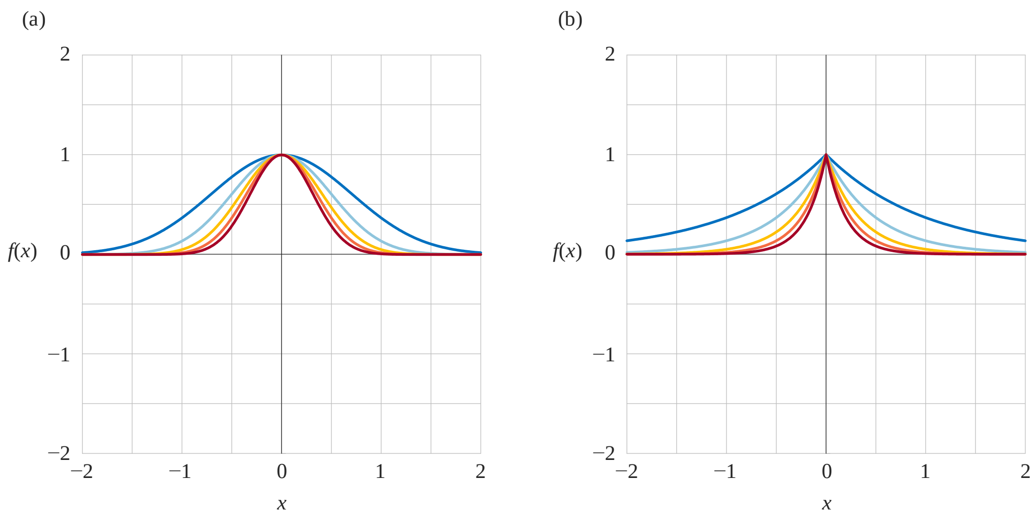


图 6. 高斯函数和拉普拉斯核函数

最基本的高斯函数为。

$$f(x) = \exp\left(-x^2\right) \tag{11}$$

如下也是常用的高斯函数的一般形式。

$$f(x) = a \cdot \exp\left(\frac{-(x-b)^2}{2c^2}\right) \tag{12}$$

(12) 可以通过 (11) 缩放、平移等变换来得到，本章最后讲解函数变换。

高斯函数和高斯分布 (Gaussian distribution) 概率密度函数 (Probability Density Function, PDF) 直接相关；高斯函数可以进一步推广得到**径向基核函数** (radial basis function, RBF)。

下一章将介绍二元高斯函数的性质；此外，鉴于高斯函数的重要性，本书后续导数、积分相关内容都会以高斯函数作为实例。

高斯函数以著名数学家高斯 (Carl Friedrich Gauss) 命名。在数据科学和机器学习领域，高斯的名字无处不在。从大家耳熟能详的高斯函数、高斯消去、高斯分布、最小二乘法，到高斯平滑、高斯朴素贝叶斯、高斯判别分析、高斯过程、高斯混合模型等等。并不是高斯发明了这些算法；而是，后来人在创造这些算法时，都用到了高斯分布。

被称作数学王子的高斯，出身贫寒。母亲做过女佣，近乎文盲；父亲多半生靠体力讨生活。据说，高斯自幼喜欢读书，特别是和数学相关的书籍；渴望学习、热爱知识是他的驱动力，而不是颜如玉、黄金屋。

卡尔·弗里德里希·高斯 (Carl Friedrich Gauss)
德国数学家、物理学家、天文学家 | 1777 ~ 1855
常被称作"数学王子"，在数学的每个领域开疆拓土。丛书关键词：• 等差数列 • 高斯分布 • 最小二乘法 • 高斯朴素贝叶斯 • 高斯判别分析 • 高斯过程 • 高斯混合模型 • 高斯核函数

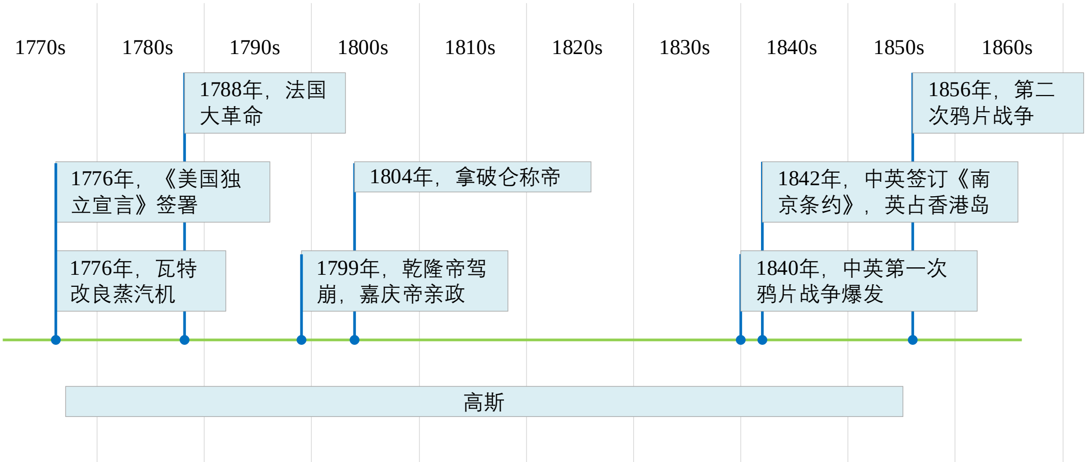


图 7. 高斯所处时代大事记

此外，绝对值函数 $|x|$ 和指数函数的复合，得到的是一元**拉普拉斯核函数** (Laplacian kernel function)。

$$f(x) = \exp(-\gamma|x|) \tag{13}$$

图 6 (b) 所示为 γ 决定拉普拉斯核函数的形状。注意，图 6 (b) 中拉普拉斯核函数在 $x = 0$ 处有"尖点"，它破坏了函数的平滑。拉普拉斯核函数也经常出现在机器学习当中。

9.4 逻辑函数：在 0 和 1 之间取值

逻辑函数 (logistic function) 也可以视作是自然指数函数扩展得到。下式为最简单的一元逻辑函数。

$$f(x)=\frac{1}{1+e^{-x}}=\frac{e^{x}}{1+e^{x}} \tag{14}$$

更一般的一元逻辑函数形式为。

$$f(x)=\frac{1}{1+\exp\left(-\left(b_0+b_1x\right)\right)} \tag{15}$$

可以明显发现逻辑函数的取值范围在 0 和 1 之间，函数无限接近 0 和 1，却不能达到；而 b_1 影响图像的陡峭程度，具体如图 8 所示；注意图中 $b_0 = 0$。

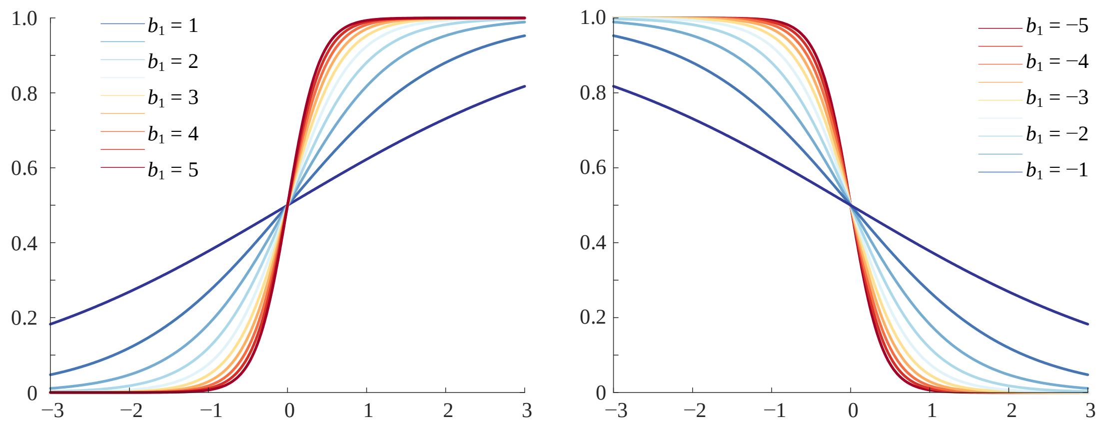


图 8. b_1 影响逻辑函数的陡峭程度

下面确定 $f(x) = 1/2$ 位置。

$$f(x)=\frac{1}{1+\exp\left(-\left(b_0+b_1x\right)\right)}=\frac{1}{2} \tag{16}$$

整理得到

$$x=-\frac{b_0}{b_1} \tag{17}$$

这个点被称作为逻辑函数中心所在位置。图 9 中 $b_1 = 1$，b_0 决定逻辑函数中心所在位置。

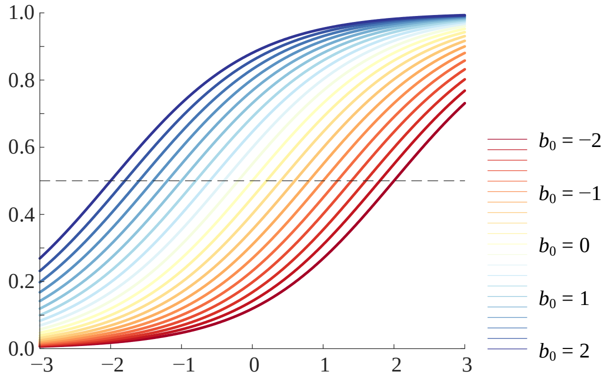


图 9. $b_1 = 1$ 时，b_0决定逻辑函数中心位置

逻辑回归 (logistic regression) 基于逻辑函数，逻辑回测虽然被称作回归模型，但是它经常用来做分类，特别是二分类。

逻辑回归可以看做是在线性回归基础上增加了一个非线性映射。线性回归中输出值 y 是连续值，而逻辑回归中 y 可以为离散值，比如 0、1。

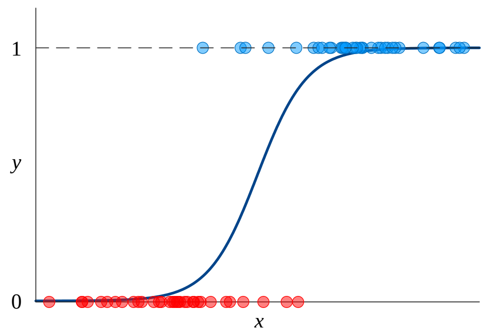


图 10. 逻辑回归可以用来做二分类

9.5 三角函数：周期函数的代表

本节介绍几个常用的三角函数。**三角函数** (trigonometric function 或 circular function) 是一类**周期函数** (periodic function)。

正弦波 (sine wave 或 sinusoid) 是一种常见的波形，比如**正弦交流电** (sinusoidal alternating current)。图 11 (a) 所示为如下最基本的**正弦函数** (sine function)。

$$y = f(x) = \sin(x) \tag{18}$$

图 11 (a) 所示正弦函数定义域为整个实数域。$f(x) = \sin(x)$ 函数是**奇函数** (odd function)，关于原点对称。这个函数的**周期** (period) 是 $T = 2\pi$，函数值域是 $[-1, 1]$。

图 11 (a) 所示正弦函数正弦函数的最大值为 1 对应的 x 为。

$$x = \frac{\pi}{2} + 2\pi n \tag{19}$$

其中，n 为整数。

正弦函数的最小值为−1 对应 x 为。

$$x = -\frac{\pi}{2} + 2\pi n \tag{20}$$

`numpy.sin()` 函数可以用来完成正弦计算。

图 11 (b) 所示为如下**余弦函数** (cosine function)。

$$y = f(x) = \cos(x) \tag{21}$$

图 11 (b) 所示余弦函数是偶函数，关于纵轴对称；$y = \cos(x)$ 相当于图 11 (a) 正弦函数水平向左移动 $\pi/2$。余弦函数也是周期为 2π 的周期函数。`numpy.cos()` 函数可以用来完成余弦计算。

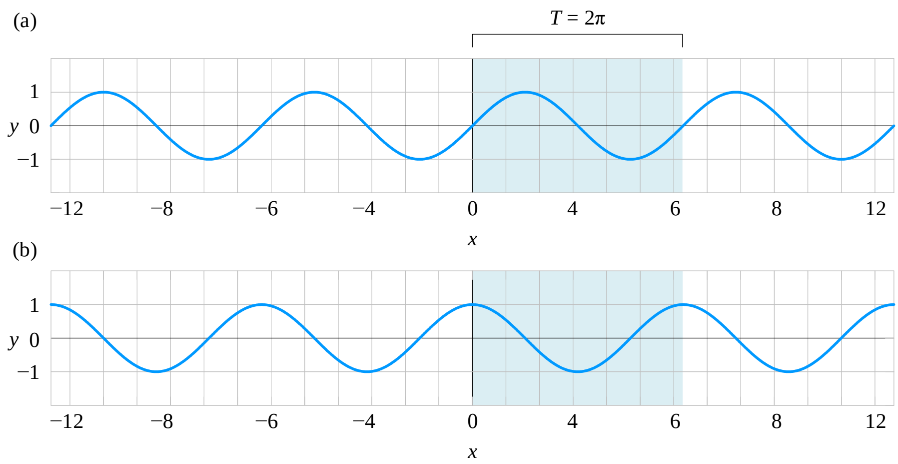


图 11. 正弦函数和余弦函数

表 3 总结了六个常用三角函数图像及性质。

表 3. 六个三角函数的图像和性质

函数	性质	图像

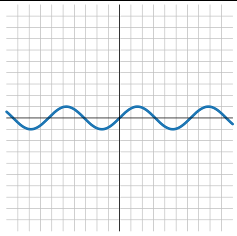
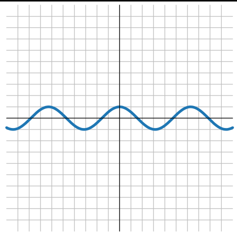
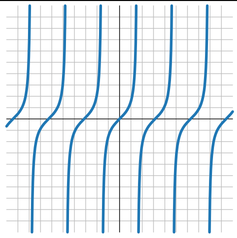
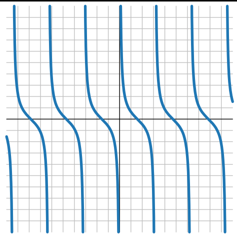
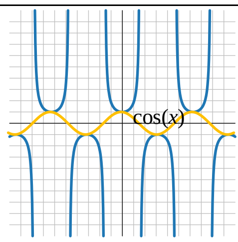
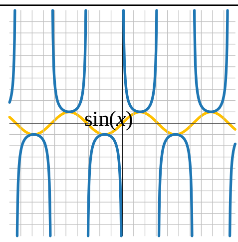
正弦 (sine) $y = \sin(x)$ `numpy.sin()`	定义域：整个实数集 值域：$[-1, 1]$ 最小正周期：2π 奇函数，图像关于原点对称 极大值为 1，极小值为-1	
余弦 (cosine) $y = \cos(x)$ `numpy.cos()`	定义域：整个实数集 值域：$[-1, 1]$ 最小正周期：2π 偶函数，图像关于 y 轴对称 极大值为 1，极小值为-1	
正切 (tangent) $y = \tan(x)$ 也记做 $y = \mathrm{tg}(x)$ `numpy.tan()`	定义域：$\left\{x \mid x \neq k\pi + \frac{\pi}{2}, k \in \mathbb{Z}\right\}$ 值域：整个实数集 最小正周期：π 奇函数，图像关于原点对称 不存在极值	
余切 (cotangent) $y = \cot(x)$ 也记做 $y = \mathrm{ctg}(x)$ `1/numpy.tan()`	定义域：$\{x \mid x \neq k\pi, k \in \mathbb{Z}\}$ 值域：整个实数集 最小正周期：π 偶函数，图像关于 y 轴对称 不存在极值	
正割 (secant) $y = \sec(x)$ `1/numpy.cos()`	定义域：$\left\{x \mid x \neq k\pi + \frac{\pi}{2}, k \in \mathbb{Z}\right\}$ 值域：$\lvert\sec(x)\rvert \geqslant 1$ 最小正周期：2π 偶函数，图像关于 y 轴对称 不存在极值	
余割 (cosecant) $y = \csc(x)$ `1/numpy.sin()`	定义域：$\{x \mid x \neq k\pi, k \in \mathbb{Z}\}$ 值域：$\lvert\csc(x)\rvert \geqslant 1$ 最小正周期：2π 奇函数，图像关于原点对称 不存在极值	

表 4. 用英文读三角函数

数学表达	英文表达
$\sin\theta + x$	Sine of theta, that quantity plus x.
$\sin(\theta + \omega)$	Sine of sum theta plus omega. Sine of the quantity theta plus omega.
$\sin(\theta) \cdot x$	Sine theta times x.

$\sin(\theta\omega)$	Sine of the product theta time omega.
$(\sin\theta^2)\cdot x$	Sine of theta squared, that quantity times *x*.
$(\sin^2\theta)\cdot x$	Sine squared of theta, that quantity times *x*.

9.6 函数变换：平移、缩放、对称

本章最后利用高斯函数和大家探讨函数变换。

给定某个函数 $y = f(x)$ 解析式为。

$$f(x) = 2\exp\left(-(x-1)^2\right) \tag{22}$$

如图 12 所示，相对 $y = f(x)$，$f(x) + c$ 相对原函数竖直向上平移 c 单位 (vertical shift up by c units)；$f(x) - c$ 为原函数竖直向下平移 c 单位 (vertical shift down by c units)。

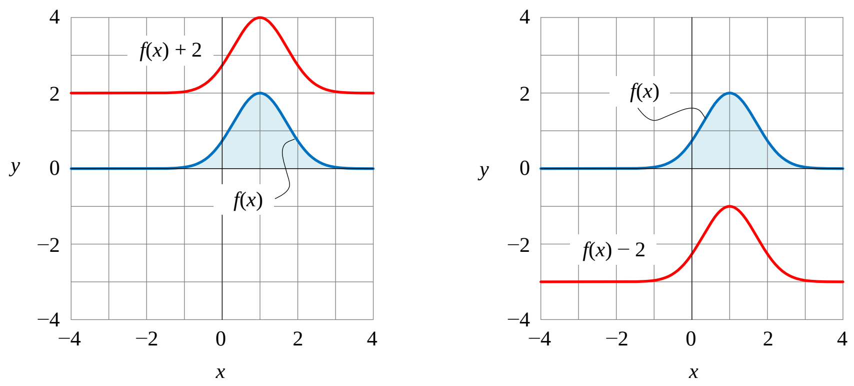


图 12. 原函数 $y = f(x)$ 上下平移

如图 13 所示，相对 $y = f(x)$，$f(x + c)$ 相当于函数向左平移 c 单位 (horizontal shift left by c units)，$c > 0$；$f(x - c)$ 相对原函数向右平移 c 单位 (horizontal shift right by c units)。

注意，平移不影响函数和横轴包围的面积。

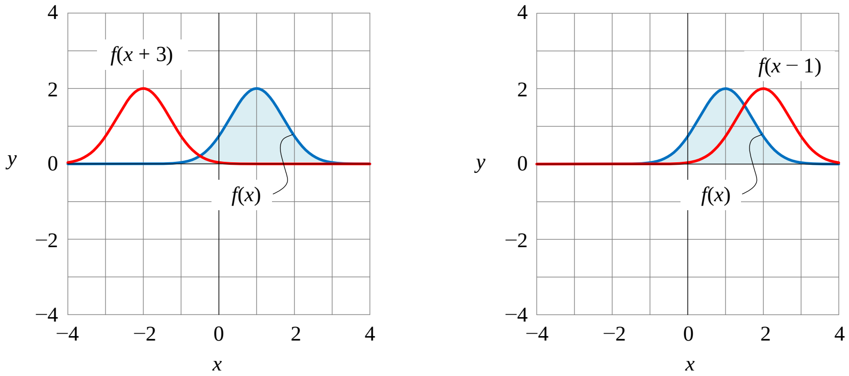


图 13. 原函数 $y = f(x)$ 左右平移

如图 14 所示，相对 $y = f(x)$，$cf(x)$ 相当于函数竖直方向伸缩 (vertical scaling)。$c > 1$ 时，竖直方向拉伸 (vertical stretch)；$0 < c < 1$ 时，竖直方向压缩 (vertical compression)。注意，这种几何变换等比例缩放面积。

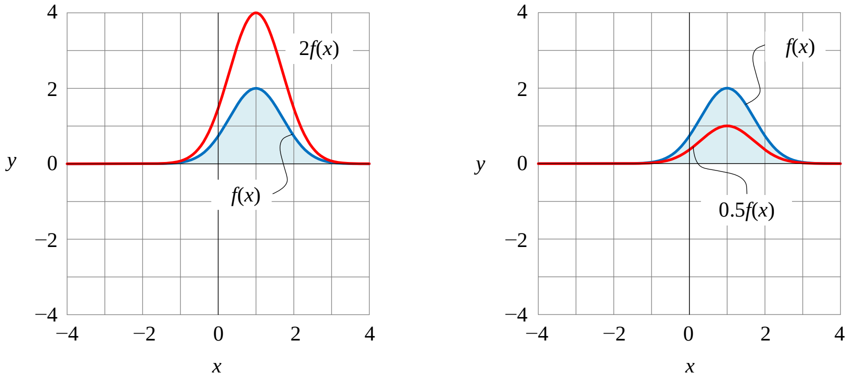


图 14. 原函数 $y = f(x)$ 竖直方向伸缩

如图 15 所示，相对 $y = f(x)$，$f(cx)$ 相当于函数水平方向伸缩 (horizontal scaling)。$c > 1$ 时，水平方向压缩 (horizontal compression)；$0 < c < 1$ 时，水平方向拉伸 (horizontal stretch)。面积等比例缩放，缩放比例为 $1/c$。

如图 16 所示，相对于 $f(x)$，$cf(cx)$ 面积不变。

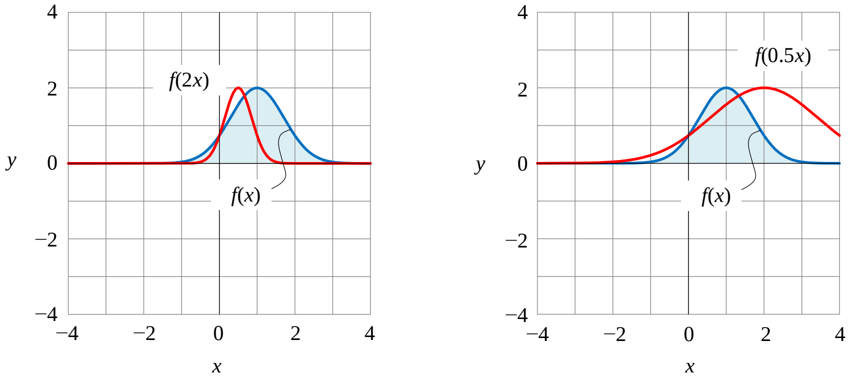


图 15. 原函数 $y = f(x)$ 水平方向伸缩

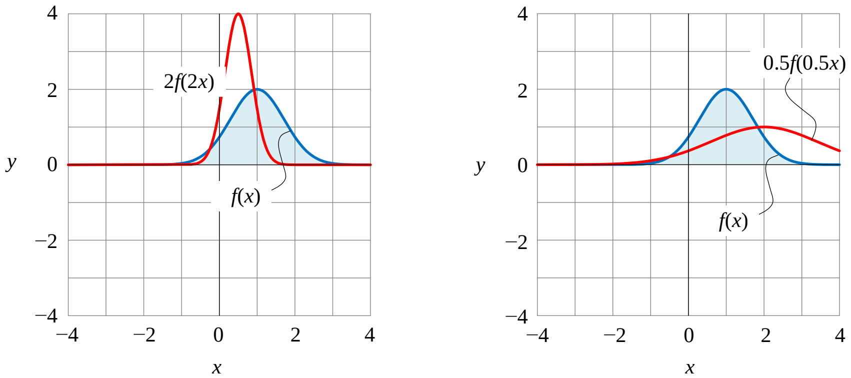


图 16. 原函数 $y = f(x)$ 水平方向、竖直方向同时伸缩

如图 17 所示，相对 $y = f(x)$，$f(-x)$ 相当于函数关于 y 轴对称 (reflection about y axis)。$-f(x)$ 相当于函数关于 x 轴对称 (reflection about x axis)。

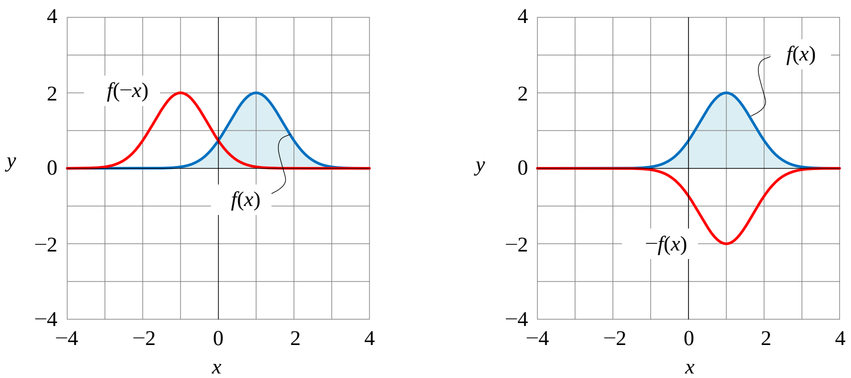


图 17. 原函数 $y = f(x)$ 关于横轴、纵轴对称

高斯分布的概率密度函数基于高斯函数，具体解析式如下。

$$p(x)=\frac{1}{\sigma\sqrt{2\pi}}\exp\left(\frac{-1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right) \tag{23}$$

其中，μ 为均值，σ 为标准差。

观察 (23) 中指数部分存在两个几何变换——横轴缩放 (σ)、横轴平移 (μ)。令

$$z=\frac{x-\mu}{\sigma} \tag{24}$$

将 (24) 代入 (23)，整理得到。

$$f(x)=\frac{1}{\sigma\sqrt{2\pi}}\exp\left(\frac{-1}{2}z^2\right) \tag{25}$$

(25) 中分母 $\sigma\sqrt{2\pi}$，起到的是纵轴缩放作用，保证曲线下方面积为 1。理解这步变换需要积分知识。图 18 所示为以上三步几何变换。

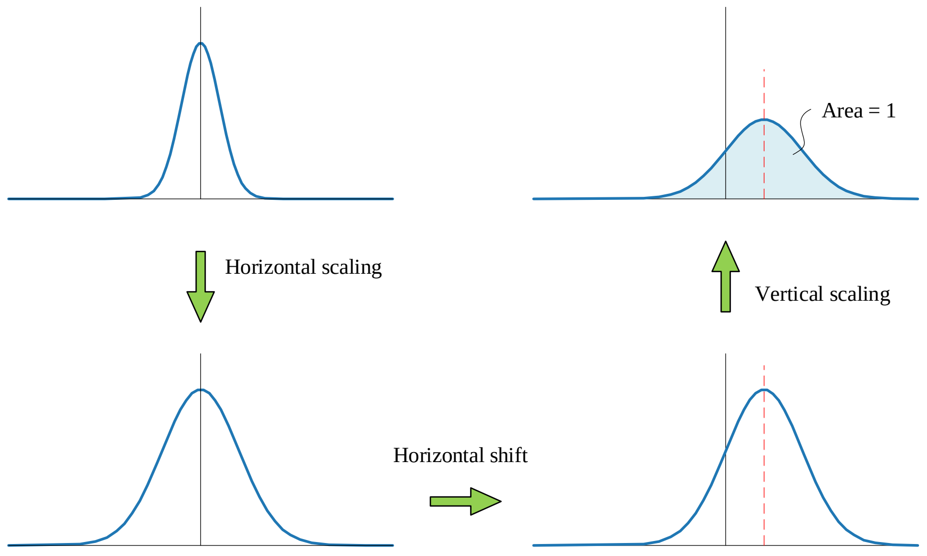


图 18. 高斯函数三步几何变换

以下代码绘制图 12~图 17。

```
# Bk_Ch9_02
```

```python
import matplotlib.pyplot as plt

def plot_curve(x_array, y_array,
               x_array_new, y_array_new):

    fig, ax = plt.subplots()

    plt.plot(x_array, y_array, color = '#0070C0',
             label = 'Original')

    ax.fill_between(x_array,
                    y_array,
                    edgecolor = 'none',
                    facecolor = '#0070C0',
                    alpha = 0.2)

    plt.plot(x_array_new, y_array_new, color = 'r',
             label = 'Transformed')

    plt.xlabel('x')
    plt.ylabel('y')
    plt.axhline(y=0, color='k', linestyle='-')
    plt.axvline(x=0, color='k', linestyle='-')
    plt.xticks(np.arange(-4, 4+1, step=1))
    plt.yticks(np.arange(-4, 4+1, step=1))
    plt.axis('scaled')

    ax.set_xlim(-4,4)
    ax.set_ylim(-4,4)
    ax.spines['top'].set_visible(False)
    ax.spines['right'].set_visible(False)
    ax.spines['bottom'].set_visible(False)
    ax.spines['left'].set_visible(False)

    plt.legend()
    ax.grid(linestyle='--', linewidth=0.25, color=[0.5,0.5,0.5])

    plt.rcParams["figure.figsize"] = [7.50, 3.50]
    plt.rcParams["figure.autolayout"] = True

import numpy as np
from sympy.abc import x
from sympy import exp, lambdify

x_array = np.arange(-4,4+0.01, step = 0.01)
f_x = 2*exp(- (x-1)**2);

f_x_fcn = lambdify([x],f_x)

f_x_array = f_x_fcn(x_array) # original function

#%% vertical shift

for c in [2,-3]:

    f_x_array_new = f_x_array + c

    plot_curve(x_array, f_x_array,
               x_array, f_x_array_new)


#%% horizontal shift

for c in [3,-1]:

    f_x_new = 2*exp(- ((x+c)-1)**2);
    f_x_new_fcn = lambdify([x],f_x_new)

    f_x_array_new = f_x_new_fcn(x_array)
```

```
    plot_curve(x_array, f_x_array,
               x_array, f_x_array_new)

#%% vertical scaling

for c in [1/2,2]:

    f_x_array_new = c*f_x_array

    plot_curve(x_array, f_x_array,
               x_array, f_x_array_new)

#%% horizontal scaling

for c in [1/2,2]:

    f_x_new = 2*exp(- (c*x-1)**2);
    f_x_new_fcn = lambdify([x],f_x_new)

    f_x_array_new = f_x_new_fcn(x_array)

    plot_curve(x_array, f_x_array,
               x_array, f_x_array_new)

#%% reflection about x-axis

f_x_array_new = -f_x_array

plot_curve(x_array, f_x_array,
           x_array, f_x_array_new)

#%% reflection about y-axis

f_x_new = 2*exp(- (-x-1)**2);
f_x_new_fcn = lambdify([x],f_x_new)

f_x_array_new = f_x_new_fcn(x_array)

plot_curve(x_array, f_x_array,
           x_array, f_x_array_new)
```

本章有两个要点——函数在数值转化的作用、函数变换。

机器学习各种算法中，函数起到数据转化的作用，比如把取值在正负无穷之间的数值转化在 0 和 1 之间。本系列丛书《数据科学》一册会专门探讨这个话题。

平移、缩放、对称等函数变换是几何变换在函数上的应用。请大家格外注意，函数变换过程前后，形状、单调性、极值点、对称轴、面积等性质的变化。