1. 基本介绍

Matplotlib 是一个 Python 的 2D 绘图库。通过Matplotlib，开发者可以仅需要几行代码,便可以生成绘图，直方图，功率谱，条形图，错误图，散点图等。学习 Matplotlib ，可让数据可视化，更直观的真实给用户。使数据更加客观、更具有说服力。 Matplotlib 是 Python 的库，又是开发中常用的库。

1. 1. 优点

Matplotlib 是一个非常强大的 Python 数据可视化库。以下是其主要优点：

- 全面性：Matplotlib 支持各种图表，包括直方图、散点图、线图、柱状图、饼图、箱线图等等。几乎所有的统计图表，Matplotlib 都能生成。
- 控制能力：使用 Matplotlib，可以控制图表的每一个细节，包括颜色、线宽、坐标轴、图例、标签等等。
- 易用性：Matplotlib 对于初学者而言非常友好，可以使用一行代码就创建出一个简单的图表。
- 可移植性：Matplotlib 可以用于各种操作系统和图形界面，你也可以将你的图表保存为各种常见的图形格式，例如 PNG、PDF、SVG、EPS 和 PGF。
- 集成性：Matplotlib 可以与 Python 的其他科学计算库，例如 NumPy 和 SciPy，完美地集成在一起。

1. 2. 应用场景

Matplotlib 的主要应用场景包括：

- 数据分析：Matplotlib 可以帮助数据分析师可视化他们的数据，发现数据的规律和趋势。
- 机器学习：在机器学习中，可以使用 Matplotlib 展示模型的学习曲线，评估模型的性能。
- 科学研究：在科学研究中，可以使用 Matplotlib 展示实验结果，帮助我们理解和解释实验现象。

1. 3. 环境搭建

在对应的Python环境执行下面的命令

```
pip install matplotlib
pip install matplotlib -i https://pypi.tuna.tsinghua.com.edu.cn/simple
```

```
# 检验是否安装成功
import matplotlib
print("Matplotlib version:", matplotlib.__version__)
```

2. 基本使用

2. 1. 加载和保存图片

读取本地图片

```
import matplotlib.pyplot as plt
import matplotlib.image as mpimg

# 传入读取图片路径 data是一个ndarray 数组
data = mpimg.imread('./a.jpg')
# 显示图片
plt.imshow(data)
# 关闭坐标轴
plt.axis('off')
# 显示图片
plt.show()
# 保存图片
# cmap='viridis' 彩色图片,  cmap='gray' 黑白图片
plt.imsave("b.jpg",data, cmap='viridis')
```

2. 2. 案例: 完成下面的图片处理功能

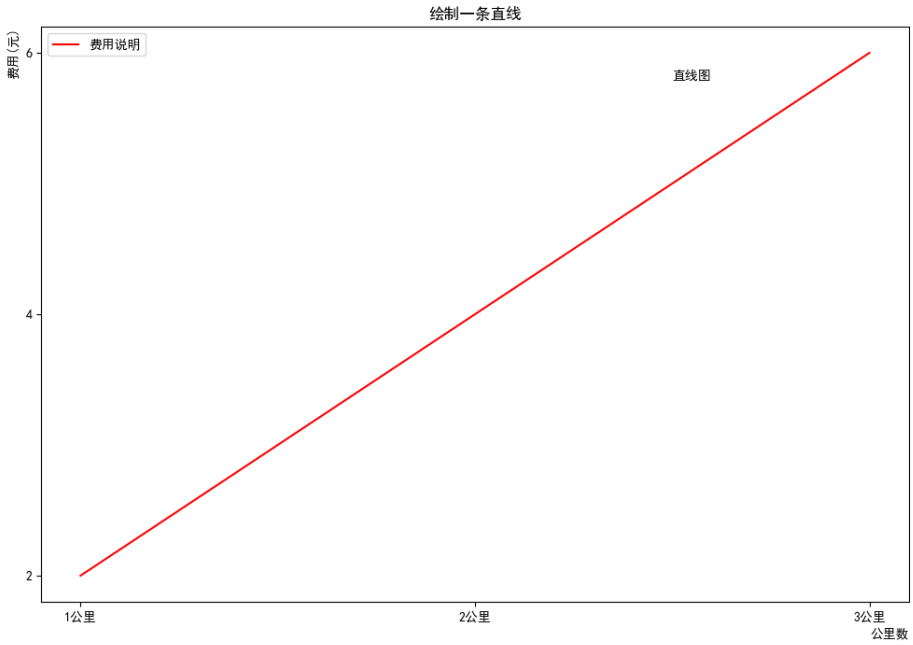
1. 读取图片 a.jpg (分辨率是:960*1280)
2. 降低图片分辨率为原来的一半(分辨率是:480*640)
3. 把图片进行水平翻转
4. 保存处理后的图片

```
# 1. 读取图片 a.jpg (分辨率是:960*1080)
```

```
# 2. 降低图片分辨率为原来的一半(分辨率是:480*540)
# 3. 把图片进行水平翻转
# 4. 保存处理后的图片
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
import numpy as np

# 1. 读取图片 a.jpg
data = mpimg.imread('./a.jpg')
# print(data.shape)
# 2. 降低图片分辨率为原来的一半
data_copy = data[::2,::2]
# print(data_copy.shape)
# 3. 把图片进行水平翻转
data_copy = data_copy[:,::-1]
# 4. 保存处理后的图片
plt.imsave('b.jpg',data_copy)
plt.imshow(data_copy)
plt.axis(False)
plt.show()
```

> 注意 如果在使用imshow显示灰度图像时, 需要指定一个参数 cmap='gray'
>
> plt.imshow(data_copy, cmap='gray')

2. 3. 绘制直线



需求: 绘制上述图形

1. 画布的大小设置为 宽 12英寸, 高 8英寸
2. 绘制的数据 x=[1,2,3], y=[2,4,6]
3. 线条的颜色设置为红色

4. 设置图形的标题为 绘制一条直线
5. 设置x轴的label标题 公里数, 并且显示在右边
6. 设置y轴的label标题 费用(元), 并且显示在上边
7. 设置x轴的刻度信息 设置为 '1公里','2公里','3公里'
8. 设置y轴的刻度信息 [2,4,6]
9. 在位置(2,5,5,8)位置上显示 直线图
10. 设置图例信息
11. 保存图片信息 plot.png
12. 显示图片

```
import matplotlib.pyplot as plt
# 设置中文字体
plt.rcParams['font.sans-serif']=['SimHei']
# x 轴的数据
x = [1,2,3]
# y 轴的数据
y = [2,4,6]
# 设置画布的大小, 宽:12英寸, 高 8英寸
plt.figure(figsize=(12,8))
# 绘制线段 x: x轴的数据列表, y: y轴的数据列表
plt.plot(x,y,color='red',label='费用说明')
# 设置图形标题
plt.title("绘制一条直线")
# 设置x轴的label标题
plt.xlabel("公里数",loc='right')
# 设置y轴的label标题
plt.ylabel("费用(元)",loc='top')
# 设置x轴的刻度信息
plt.xticks(ticks=[1,2,3],labels=['1公里','2公里','3公里'])
# 设置y轴的刻度信息
plt.yticks(ticks=[2,4,6])
# 设置一个文本内容显示
plt.text(2.5,5.8,"直线图")
# 设置图例信息
plt.legend()
# 保存图片信息
plt.savefig('plot.png')
# 显示图片
plt.show()
```

2. 4. 中文字体的处理

默认情况下, 如果在图表中需要显示中文会乱码,Matplotlib 配置为使用一些如 `DejaVu Sans` 等主要支持拉丁字符的字体。

如果需要改为支持中文, 可以通过全局配置字典 `plt.rcParams`

```
plt.rcParams['font.sans-serif']=['SimHei']
```

上述配置是windows上面的配置, 如果是linux或者mac, 需要配置对应的中文字体库

1. Windows

   在window系统中, 大多数字体文件存放的目录是 `C:\Windows\Fonts`

2. macOS

在macOS重, 字体通常存放在以下几个位置

```
/System/Library/Fonts
/Library/Fonts
# ~ 表示当前用户的主目录
~/Library/Fonts
```

3. Ubuntu

   在Ubuntu以及其他Linux发行版中, 字体可能存放在多个目录 常见的有

```
/usr/share/fonts
/usr/local/share/fonts
~/.fonts
```

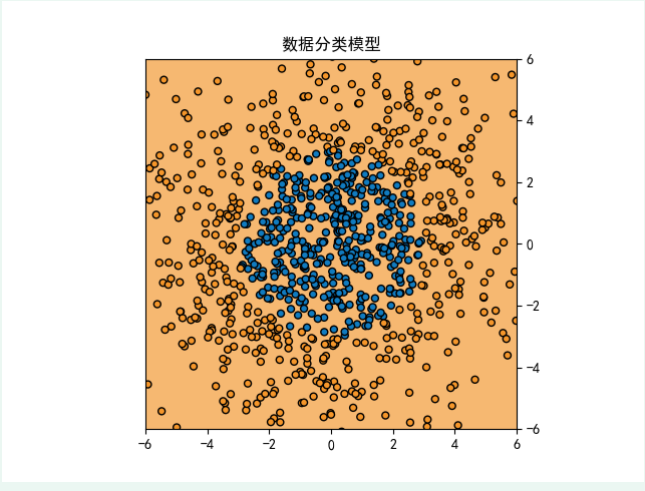
使用绝对路径指定字体

```
import matplotlib.pyplot as plt
from matplotlib.font_manager import FontProperties
# 字体文件的绝对路径
font_path = 'C:/Windows/Fonts/simhei.ttf'
font_prop = FontProperties(fname=font_path)
print(font_prop.get_name())
plt.rcParams['font.sans-serif'] = [font_prop.get_name()]
```

2. 5. 绘制散点图

散点图（Scatter Plot）是用来探索两个（或更多）变量之间关系的基本工具之一。Matplotlib 提供了强大的散点图绘制功能，它可以帮助分析和解释变量间的相互作用、趋势、异常点、关联性等。

图形连接地址: **https://playground.tensorflow.org/**



需求: 绘制上述图形

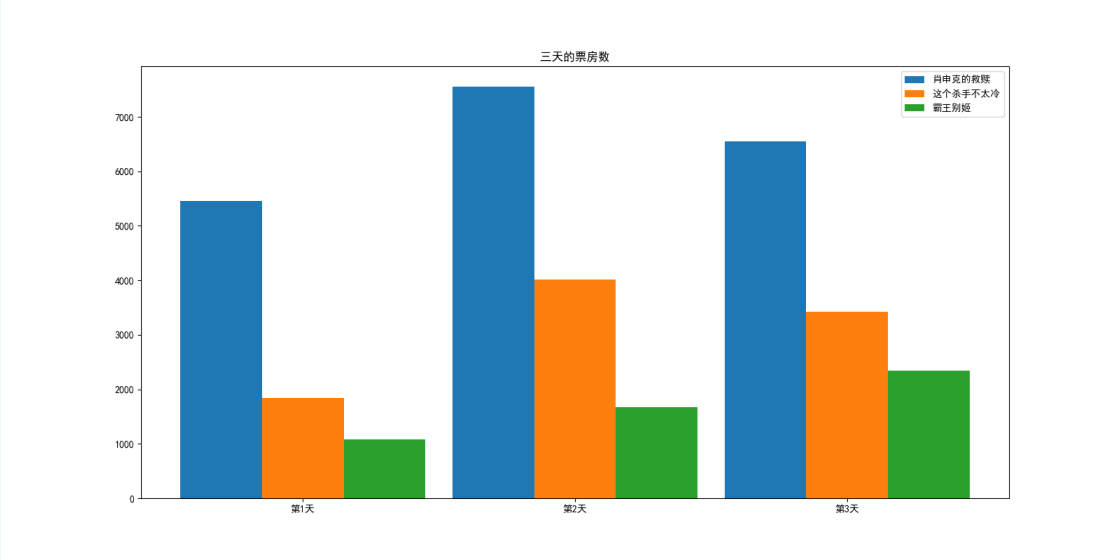
1. 模拟生成对应的1000个的数据点, 其中数据点符合正态分布, 均值是0, 标准差是3
2. 设置数据点的颜色,距离圆点(0,0)在半径为3的圆内, 使用颜色'#117bbf', 否则使用颜色'#f5962a'
3. 设置图表标题为"数据分类模型"
4. 设置X轴的取值范围是:[-6,6]
5. 设置Y轴的取值范围是:[-6,6],并且Y轴线图形的右边显示
6. 设置图表的背景颜色为:'#f6b871'
7. 保存图像为 scatter.png

8. 绘制一个散点图,并且显示图片

```
import matplotlib.pyplot as plt
import numpy as np
# 设置负号可以正常显示
plt.rcParams['axes.unicode_minus'] = False
# 生成数据点
num_points = 1000
x = np.random.normal(0, 3, num_points)
y = np.random.normal(0, 3, num_points)
radii = np.sqrt(x**2 + y**2)
# 设置颜色：圆内为蓝色，圆外为黄色
colors = np.where(radii <= 3, '#117bbf', '#f5962a')
# 设置标题
plt.title("数据分类模型")
# 设置图表属性
plt.xlim([-6, 6])
plt.ylim([-6, 6])
plt.gca().set_aspect('equal')    # 确保图形为正方形
plt.gca().set_facecolor('#f6b871')
plt.gca().yaxis.set_ticks_position('right')
# 绘制散点图
plt.scatter(x, y, color=colors, edgecolor='k')
# 保存图片信息
plt.savefig('scatter.png')
# 显示图形
plt.show()
```

> 注意：对于坐标轴上的负数不能正常显示, 需要配置
>
> plt.rcParams['axes.unicode_minus'] = False

2. 6. 绘制柱状图



根据要求绘制上述图形

1. 设置画布大小为(16,9)

2. 设置字体为SimHei
3. 准备3部电影连续三天的票房数据

   # 电影

   movies = ['肖申克的救赎', '这个杀手不太冷', '霸王别姬']

   # 电影票房

   movies_ticket_offices1 = [5453, 7548, 6543]

   movies_ticket_offices2 = [1840, 4013, 3421]

   movies_ticket_offices3 = [1080, 1673, 2342]
4. 设置图片的标题为 "三天的票房数"
5. 设置X轴的刻度数据为 ["第1天","第2天","第3天"]
6. 显示图例信息
7. 保存图片bar.png
8. 显示图片

```
import matplotlib.pyplot as plt
# 设置中文字体
plt.rcParams['font.sans-serif']=['SimHei']
# 设置画布大小
plt.figure(figsize=(16,9))
# 电影
movies = ['肖申克的救赎', '这个杀手不太冷', '霸王别姬']
# 电影票房
movies_ticket_offices1 = [5453, 7548, 6543] # 第一部电影3天的票房
movies_ticket_offices2 = [1840, 4013, 3421]
movies_ticket_offices3 = [1080, 1673, 2342]
# 设置柱的宽度
plt.bar([1,5,9], movies_ticket_offices1, width=1, label=movies[0])
plt.bar([2,6,10], movies_ticket_offices2, width=1, label=movies[1])
plt.bar([3,7,11], movies_ticket_offices3, width=1, label=movies[2])
# 修改 x 坐标
plt.xticks([2,6,10], ['第1天','第2天','第3天'])
# 添加标题
plt.title('三天的票房数')
# 添加图例
plt.legend()
plt.savefig('bar.png')
plt.show()
```

2. 7. 绘制直方图

在深度学习和大模型研究中，直方图是一个非常有用的工具，常用于数据分析、模型诊断、以及监控模型训练过程。直方图通过提供数据的分布视图，帮助研究人员和工程师理解和优化他们的模型。

2. 7.1. 需求背景

在深度学习领域，特别是在处理图像识别、分类和分析任务时，输入数据的质量直接影响到模型的训练效果和最终性能。图像数据，由于其复杂性和高维度特性，尤其需要经过精细的预处理步骤，以确保模型能够有效学习并做出准确的预测。

为什么关注图像的亮度和对比度?

图像的亮度和对比度是影响模型感知和学习能力的两个基本视觉属性。亮度影响图像的明暗程度，而对比度影响图像中对象与背景的区分度。在自然环境中拍摄的图像常常因为光照条件的不一而有很大差异，例如，过暗或过亮的图像可能隐藏或扭曲了重要的视觉信息，使得深度学习模型难以从中学习到有效的特征。

数据分布的影响

一个理想的训练集应该有助于模型学习到能够泛化到各种不同条件下的数据。这就要求训练数据在亮度和对比度上能够反映出现实世界中可能遇到的各种情况。如果数据集中大部分图像都非常暗或者非常亮，模型在训练时可能就会偏向于这种特定类型的图像，从而影响其在更广泛或不同亮度条件下的表现。

正态分布的重要性

正态分布（或称高斯分布）是许多自然和人造过程中最常见的数据分布类型，它在统计学上的普遍性使得许多机器学习算法都假设输入数据接近正态分布。在图像处理中，通过调整使像素值的分布接近正态分布，可以增加模型训练的稳定性和效率。例如，通过调整图像的亮度和对比度使像素值的分布更加均匀，可以帮助模型更好地理解图像内容，避免训练过程中的数据偏差。

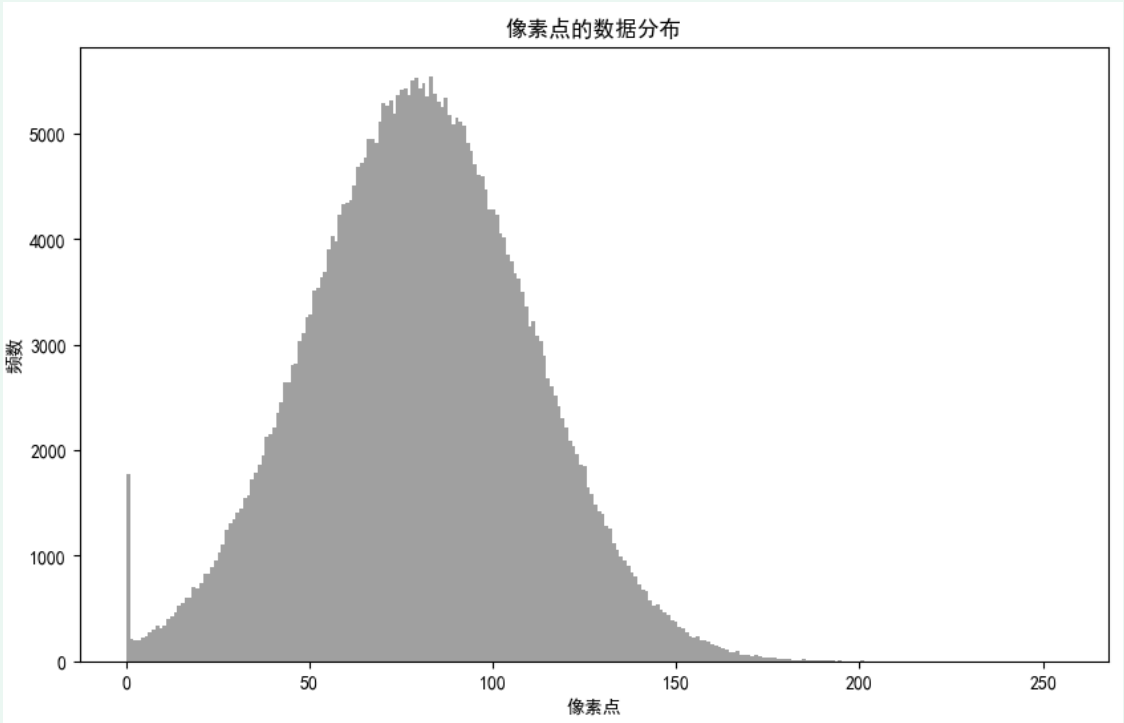
2. 7.2. 目标设置

本案例的主要目标是通过使用正态分布来生成模拟的图像数据，然后利用直方图分析这些数据的像素值分布。通过这种分析，我们将评估图像数据中的亮度和对比度分布，并根据分布的结果确定是否需要对图像进行亮度调整或对比度优化，以确保数据集更加均衡，从而为深度学习模型的训练提供更理想的输入条件。这个过程不仅帮助我们理解数据预处理的重要性，还展示了如何通过简单的统计工具优化机器学习项目中的数据处理步骤。

2. 7.3. 完整代码实现

```
import numpy as np
import matplotlib.pyplot as plt

# 模拟生成图像数据
# 假设我们有100张图像，每张图像大小为64x64
# 使用正态分布生成暗图像数据，均值较低（暗）
num_images = 100
image_size = 64  # 64x64 pixels
low_brightness_mean = 80  # 较低的亮度均值
brightness_std = 30  # 标准差

# 生成模拟数据
images = np.random.normal(low_brightness_mean, brightness_std, (num_images, image_size, image_size))

# 将像素值限制在0-255的范围内
images = np.clip(images, 0, 255)

# 绘制单个图像的直方图
plt.figure(figsize=(10, 6))
plt.hist(images.flatten(), bins=256, range=[0, 255], color='gray', alpha=0.75)
```

```
plt.title('像素点的数据分布')
plt.xlabel('像素点')
plt.ylabel('频数')

# 显示直方图
plt.show()

# 分析结果
if images.mean() < 128:
    print("大部分图像看起来较暗。建议增加亮度或调整对比度以改善模型训练效果。")
else:
    print("图像亮度适中，不需要调整。")
```



3. 案例应用

3. 1. 函数 Sigmoid 和 ReLU

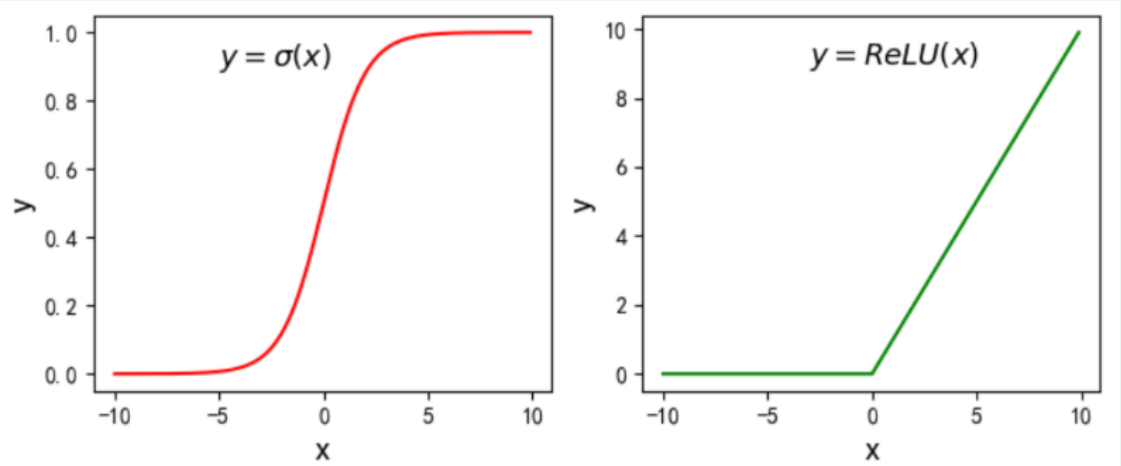
使用 ndarray 数组可以很方便的构建数学函数，并利用其底层的矢量计算能力快速实现计算。下面以是后面神经网络中比较常用激活函数 Sigmoid 和 ReLU 为例

Sigmoid 函数公式如下：$y = \sigma(x)$

ReLU 函数公式如下：$y = ReLU(x)$

但仅仅只是从公式看，很不直观，不易理解，若是使用 Matplotlib 画出函数图形，就利于大家分析理解。下面就使用代码画出各自函数图形：

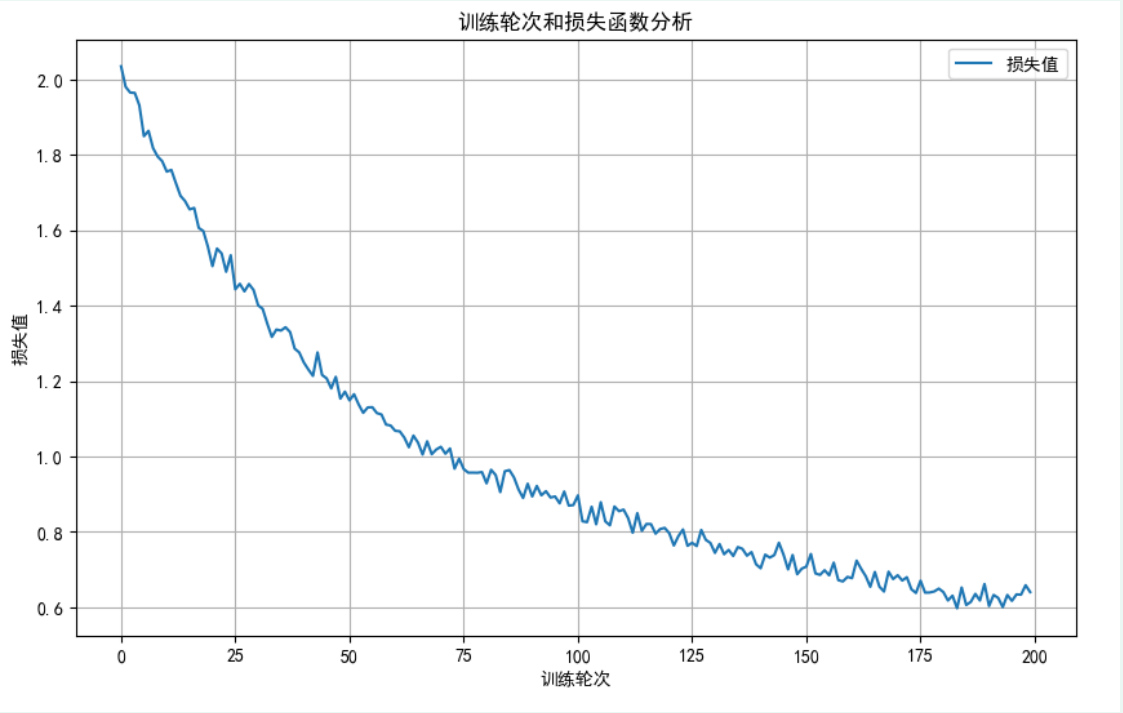
```
import numpy as np
import matplotlib.pyplot as plt

# 设置图片大小 宽度为8英寸, 高度为3英寸
plt.figure(figsize=(8, 3))

# x是1维数组，数组大小是从 -10 到 10 的实数，每隔 0.1 取一个点
x = np.arange(-10, 10, 0.1)
# 根据 Sigmoid 函数计算值
s = 1.0 / (1 + np.exp(-x))
# 根据 ReLU 函数计算值
y = np.clip(x, a_min=0., a_max=None)

# 设置两个子图窗口，将 Sigmoid 的函数图像画在左边
plt.subplot(121)
# 画出函数曲线
plt.plot(x, s, color='r')
# 添加文字说明
plt.text(-5., 0.9, r'$y=\sigma(x)$', fontsize=13) # 前两个参数指定文本的位置
# 设置坐标轴格式
currentAxis=plt.gca() # 获取到当前的坐标轴
currentAxis.xaxis.set_label_text('x', fontsize=15)
currentAxis.yaxis.set_label_text('y', fontsize=15)

# 将 ReLU 的函数图像画在右边
plt.subplot(122)
# 画出函数曲线
plt.plot(x, y, color='g')
# 添加文字说明
plt.text(-3.0, 9, r'$y=ReLU(x)$', fontsize=13)
# 设置坐标轴格式
currentAxis=plt.gca()
currentAxis.xaxis.set_label_text('x', fontsize=15)
currentAxis.yaxis.set_label_text('y', fontsize=15)

plt.show()
```



3. 2. 损失函数

在机器学习模型的训练过程中，监控损失函数的变化是至关重要的。损失函数（或成本函数）衡量的是模型预测值与真实值之间的差异。理想的目标是随着训练的进行，通过优化算法逐渐减小这个损失值，从而提高模型的预测精度。本案例将模拟一个典型的损失下降过程，使用 Matplotlib 绘制损失随训练次数增加而变化的图表，直观地展示模型学习的效果。

```
import matplotlib.pyplot as plt
import numpy as np

# 设置中文字体
plt.rcParams['font.sans-serif']=['Simhei']

# 设置随机种子
np.random.seed(0)

# 模拟训练次数，例如 200 次训练迭代
num_epochs = 200

# 生成模拟的损失数据，这里我们使用一个更快的指数衰减速度，并在训练后期加上更小的衰减率
initial_loss = 1
decay_rate = 5    # 增大衰减率以使初始下降更快
later_decay = 0.5    # 后期的衰减率，使下降平滑

# 使用指数函数模拟损失下降，加入不同阶段的衰减率
t = np.linspace(0, 1, num_epochs)
loss_values = initial_loss * np.exp(-decay_rate * t) + np.random.normal(0, 0.02, num_epochs)
loss_values += np.exp(-later_decay * t)    # 添加更平缓的衰减趋势

# 绘制损失函数随训练次数的变化图
plt.figure(figsize=(10, 6))
plt.plot(loss_values, label='损失值')
plt.title('训练轮次和损失函数分析')
plt.xlabel('训练轮次')
plt.ylabel('损失值')
plt.legend()
plt.grid(True)
plt.show()
```


训练轮次和损失函数分析
损失值
2.0
1.8
1.6
1.4
1.2
1.0
0.8
0.6
损失值
0
25
50
75
100
125
150
175
200
训练轮次

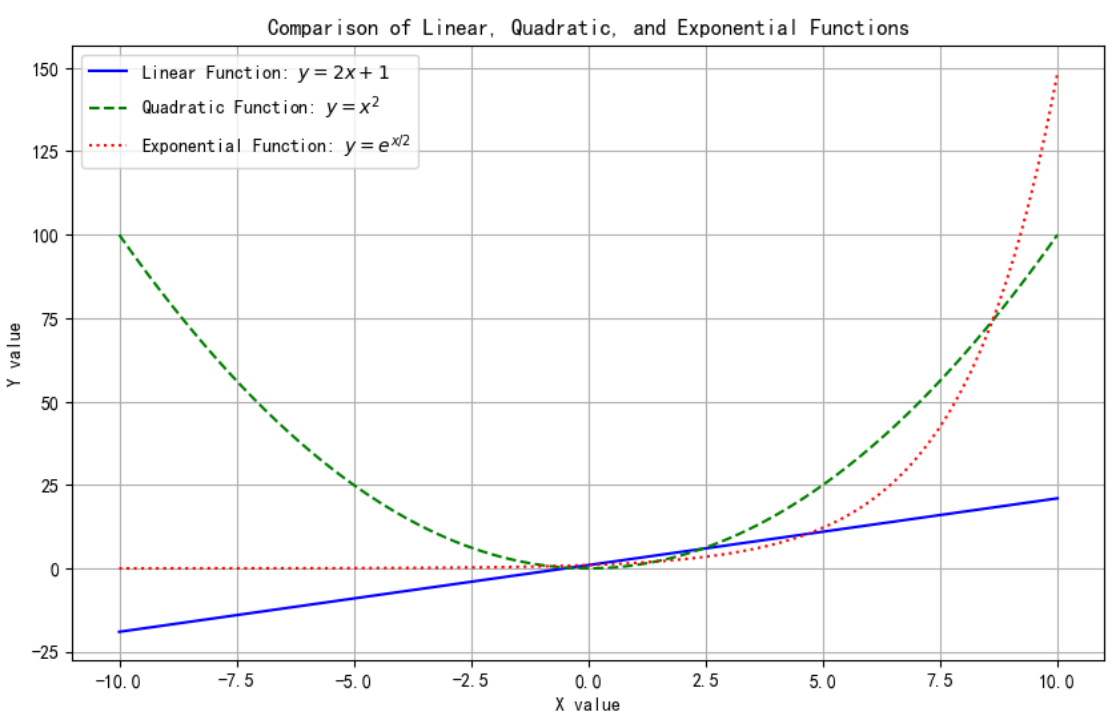
4. 作业

4. 1. 作业题目1: 绘制多功能直线图

使用 Matplotlib 绘制一个包含多条直线的图，每条直线代表一种不同的数学函数（线性、二次方和指数）。学生需要设置图例、坐标轴标签、标题，并改变线条的风格和颜色。

具体要求:

1. 在同一张图中绘制以下三个函数的图像:
   - y=2x+1
   - $y = x^2$
   - $y = e^{(x/2)}$
2. 为每条线选择不同的颜色和线型。
3. 添加图例，指明每条线代表哪个函数。
4. 设置坐标轴的标题和图形的总标题。
5. 图形的X轴范围从 -10 到 10，Y轴自动调整。



4. 2. 作业题目2:数据可视化与分析

给定一组数据，表示某城市过去10年每月的平均温度。要求学生使用 Matplotlib 绘制这些数据的线图，并标注出最高温和最低温的点。

具体要求:

1. 绘制过去10年每月的平均温度线图。
2. 使用不同颜色标记出历史最高温和最低温的月份。
3. 添加网格以便更好地阅读图表。

4. 图表需要包括合适的标签、标题和图例。

数据说明：

- 可以使用随机生成的数据或任何可公开获取的数据集。