|  |  |  |  |
| --- | --- | --- | --- |
| **实验内容** | 网络爬取两类（猫狗）图片，使用支持向量机（SVM）分类器对图片进行二分类 | | |
| **小组成员** | | | |
| **姓名** | | **班级** | **学号** |
|  | |  |  |
|  | |  |  |
| **实验目的** | | | |
| 1. 学习使用支持向量机（SVM）进行图片分类任务。 2. 掌握数据预处理的技巧，特别是如何对图片进行尺寸调整和特征提取。 3. 熟悉使用Python的Pillow库进行图像处理，和scikit-learn库进行机器学习任务。 4. 理解训练集和测试集的划分，学会评估模型的分类效果。 | | | |
| **实验原理、步骤、算法** | | | |
| 1. 实验原理   支持向量机（SVM）是一种常用的监督学习算法，特别适用于分类任务。其基本原理是通过寻找一个超平面，将不同类别的样本点分开，从而实现分类。在本实验中，SVM用于对两类图片进行分类。   1. 实验步骤 2. 数据准备   搜集并整理两类图片，每类50张，总共100张图片。  将所有图片调整为32x32像素，确保数据的统一性。  将数据分为训练集和测试集，80张作为训练集，20张作为测试集。   1. 图片预处理   使用Pillow库加载图片，将其转换为RGB模式并调整为32x32像素。  将每张图片转换为一维向量，以便SVM能够处理。   1. 模型训练   使用scikit-learn中的SVC（支持向量分类器）来训练模型。  选择线性核函数，并进行训练。   1. 模型测试与评估   使用测试集对训练好的SVM模型进行评估，计算模型的准确率和其它评估指标（如精确率、召回率等）。   1. 实验算法   支持向量机（SVM）分类器  SVM通过构建一个超平面，在高维空间中最大化类间的间隔，来进行分类。其基本算法步骤包括：   * 选择支持向量，确定分类边界。 * 使用核函数将数据映射到高维空间。   最终利用训练集得到的支持向量和边界进行预测。   1. 图像处理   使用Pillow库将图片加载为RGB格式，并调整为32x32像素。每张图片被展平为一维数组，作为SVM的输入特征。 | | | |
| **实验结果** | | | |
|  | | | |
| **附录：程序代码** | | | |
| 1. 爬虫代码   from selenium import webdriver  from selenium.webdriver.common.by import By  from selenium.webdriver.common.action\_chains import ActionChains  import time  import os  import requests  # 创建文件夹  os.makedirs("images/cats", exist\_ok=True)  os.makedirs("images/dogs", exist\_ok=True)  def download\_image(url, folder, count):  try:  response = requests.get(url, stream=True)  if response.status\_code == 200:  file\_path = os.path.join(folder, f"image\_{count}.jpg")  with open(file\_path, 'wb') as file:  for chunk in response.iter\_content(1024):  file.write(chunk)  print(f"下载成功: {file\_path}")  else:  print(f"无法下载图片，HTTP状态码: {response.status\_code}")  except Exception as e:  print(f"下载失败: {url}, 错误: {e}")  def scrape\_images(query, folder, num\_images):  # 设置 Selenium 驱动（以 Chrome 为例）  options = webdriver.ChromeOptions()  options.add\_argument("--headless") # 无界面模式  options.add\_argument("--disable-gpu")  options.add\_argument("--no-sandbox")  driver = webdriver.Chrome(options=options)  search\_url = f"https://www.google.com/search?q={query}&source=lnms&tbm=isch"  driver.get(search\_url)  time.sleep(2) # 等待页面加载  img\_urls = set()  count = 0  while count < num\_images:  # 向下滚动加载更多图片  driver.execute\_script("window.scrollTo(0, document.body.scrollHeight);")  time.sleep(2)  images = driver.find\_elements(By.CSS\_SELECTOR, "img")  for img in images:  if count >= num\_images:  break  try:  src = img.get\_attribute("src")  if src and src.startswith("http") and src not in img\_urls:  img\_urls.add(src)  download\_image(src, folder, count)  count += 1  except Exception as e:  print(f"图片处理失败: {e}")  driver.quit()  # 爬取猫的图片  scrape\_images("cat", "images/cats", 100)  # 爬取狗的图片  scrape\_images("dog", "images/dogs", 100)  print("图片下载完成！")   1. 图片预处理代码   import os  from PIL import Image  # 定义路径  class1\_path = "images/cats/" # 类别1图片路径  class2\_path = "images/dogs/" # 类别2图片路径  output\_path = "dataset/" # 输出路径  os.makedirs(output\_path, exist\_ok=True)  # 定义图片处理函数  def process\_and\_save\_images(input\_path, output\_folder, prefix, size=(32, 32)):  """  处理图片并保存到指定文件夹  :param input\_path: 输入图片路径  :param output\_folder: 输出文件夹  :param prefix: 输出文件名前缀  :param size: 调整后的图片尺寸 (宽, 高)  """  os.makedirs(output\_folder, exist\_ok=True) # 确保输出文件夹存在  count = 0  for file in os.listdir(input\_path):  img\_path = os.path.join(input\_path, file)  try:  # 加载图片并调整大小  img = Image.open(img\_path).convert("RGB")  img\_resized = img.resize(size)  # 保存图片，命名为 "prefix\_0.jpg", "prefix\_1.jpg" ...  output\_file = os.path.join(output\_folder, f"{prefix}\_{count}.jpg")  img\_resized.save(output\_file)  count += 1  except Exception as e:  print(f"跳过无法处理的图片: {file}, 错误: {e}")  # 处理并保存类别1和类别2的图片  process\_and\_save\_images(class1\_path, output\_path, "cats/cat", size=(32, 32))  process\_and\_save\_images(class2\_path, output\_path, "dogs/dog", size=(32, 32))  print("图片处理完成，已保存到 dataset 文件夹！")   1. SVM分类代码   import os  import numpy as np  from PIL import Image  from sklearn.svm import SVC  from sklearn.metrics import classification\_report, accuracy\_score  from sklearn.model\_selection import train\_test\_split  # 设置图片路径  class1\_path = "dataset/cats/" # 类别1图片路径  class2\_path = "dataset/dogs/" # 类别2图片路径  # 加载图片数据  def load\_images(path, label):  images = []  labels = []  for file in os.listdir(path):  img\_path = os.path.join(path, file)  try:  img = Image.open(img\_path)  images.append(np.array(img).flatten()) # 展平为1D数组  labels.append(label)  except Exception as e:  print(f"跳过无法加载的图片: {file}, 错误: {e}")  return images, labels  # 加载两类数据  class1\_images, class1\_labels = load\_images(class1\_path, 0) # 类别1标记为0  class2\_images, class2\_labels = load\_images(class2\_path, 1) # 类别2标记为1  # 合并数据  X = np.array(class1\_images + class2\_images)  y = np.array(class1\_labels + class2\_labels)  # 划分训练集和测试集  X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)  # 训练SVM分类器  svm = SVC(kernel='linear', random\_state=42)  svm.fit(X\_train, y\_train)  # 测试并输出结果  y\_pred = svm.predict(X\_test)  print("分类报告:")  print(classification\_report(y\_test, y\_pred))  print("准确率:", accuracy\_score(y\_test, y\_pred)) | | | |