SVD Resolve 实验报告

NO. 152 郭东昊 PB22020514

2024 年 3 月 31 日

摘要

本研究旨在探讨图像的矩阵表示及其存储机制，并利用奇异值分解（SVD）技术对黑白和彩色图像进行压缩。首先，我们从基础的黑白图像出发，应用 SVD 方法将图像矩阵拆分为两个 unitary（酉）矩阵与一个包含特征值的对角矩阵的乘积形式。通过选取特定数量的特征值，实现了图像数据的有效压缩。随后，我们扩展了这一方法，通过调整算法以处理彩色图像，使用 RGB 色彩空间对图片进行类似的 SVD 压缩处理。

实验结果显示，应用 SVD 技术于图像压缩是高效可行的，它不仅能够实现无损压缩，还能在可接受的误差范围内进行有损压缩，显著提升了图像的传输效率和减少了存储空间的需求。这一发现对于优化网络传输和数据存储具有重要意义。

一、前言

在数据分析中，常常出现少数数据点涵盖了数据集的大部分信息的现象。为了更高效地处理这些数据，线性代数提供了多种矩阵分解技术，将复杂的原始数据转换为更简洁的形式以适应不同的应用场景。

奇异值分解（Singular Value Decomposition, SVD）是这些技术中极为关键的一个，它广泛适用于任意形状的矩阵，可以视作一种普适的特征分解方法。具体而言，对于任一$m \times n$维的矩阵$\boldsymbol{A}$，它的 SVD 可以表示为三个矩阵的乘积形式，即：

$$\boldsymbol{A} = \boldsymbol{U} * \boldsymbol{S} * \boldsymbol{V}^{\boldsymbol{T}}$$

在这里，$\boldsymbol{U}$是一个$m \times m$维的酉矩阵，$\boldsymbol{S}$是一个同样大小的对角矩阵，其对角元素被称为奇异值，而$\boldsymbol{V}$是一个$n \times n$维的酉矩阵，且$\boldsymbol{V}^{\boldsymbol{T}}$表示$\boldsymbol{V}$的转置。

本实验通过 MATLAB 编程实现了基于 SVD 的图像压缩，探索了这一技术在图像数据处理领域的应用潜力，并对比分析了不同实现方法的效率和效果。

二、问题分析

我们要实现的任务有：

1. 对灰度图像的 SVD 分解
2. 对 RGB 图像的 SVD 分解

针对第一个任务，我们有以下实现思路：

(1) 读取图像并转换为灰度图像；

(2) 设置奇异值的数量 k，对图像进行 SVD，获取前 k 个左奇异向量、右奇异向量和奇异值；

(3) 通过这些值重构图像，将重构后的图像转换为 8 位无符号整数，以便于显示和保存；

(4) 生成与 k 值相关的文件名，并将压缩后的图像保存为该文件名。

针对第二个任务，我们有以下实现思路：

(1) 读取图像并获取其维度；

(2) 设置奇异值的数量 k，对图像的每个颜色通道进行 SVD，获取前 k 个左奇异向量、右奇异向量和奇异值；

(3) 通过这些值重构颜色通道，将重构后的图像转换为 8 位无符号整数，以便于显示和保存；

(4) 最后，生成与 k 值相关的文件名，并将压缩后的图像保存为该文件名。

三、数学模型建立

任务一 对灰度图像的 SVD 分解

```matlab
a=imread('lovely.jpg'); % 读取名为'Cat.jpg'的图像文件，并将其存储在变量a中

k=10; % 设置奇异值的数量，这将决定压缩后的图像质量

[~,~,dim] = size(a); % 获取图像的维度

if dim>1
a=rgb2gray(a); % 如果图像是彩色的（即维度大于1），则将其转换为灰度图像
end

imshow(a); % 显示原始图像
title('原图') ; % 设置图像标题为'原图'

a=double(a); % 将图像数据转换为双精度数据，以进行数学运算
r=rank(a); % 计算图像的秩
[u,s,v]=svd(a); % 对图像进行奇异值分解

u1=u(:,1:k); % 获取前k个左奇异向量
v1=v(:,1:k); % 获取前k个右奇异向量
ss=diag(s); % 获取奇异值向量
sss=ss(1:k); % 获取前k个奇异值
s1=diag(sss); % 将前k个奇异值转换为对角矩阵

re=u1*s1*v1'; % 通过前k个奇异值和对应的左右奇异向量重构图像

re=uint8(re); % 将重构后的图像数据转换为8位无符号整数，以便于显示和保存
figure; % 创建新的图像窗口
imshow(re); % 显示压缩后的图像
title(['图像压缩后, k=',num2str(k)]) ; % 设置图像标题，显示压缩后的奇异值数量

filename = ['lovely_k=', num2str(k), '.jpg']; % 生成与k值相关的文件名
imwrite(re, filename); % 将压缩后的图像保存为生成的文件名
```

图 1 灰度图的 SVD 代码

代码实现思路严格遵循前一节的问题分析，在此不多做解释。值得一提的是，matlab 中 svd 函数的存在使我们的工作一下变得简单许多。

任务二 对 RGB 图像的 SVD 分解

将可能高频使用的代码封装为函数，实现 SVD 压缩的核心功能，使得代码更为简洁，增强代码可读性。在主函数中，这些代码将对每个通道都作用一遍。

```matlab
% 定义一个函数，用于进行奇异值分解并返回前k个奇异值及其对应的左右奇异向量
function [u1, s1, v1] = svd_compression(u, s, v, k)
    u1 = u(:, 1:k); % 获取前k个左奇异向量
    v1 = v(:, 1:k); % 获取前k个右奇异向量
    ss = diag(s); % 获取奇异值向量
    sss = ss(1:k); % 获取前k个奇异值
    s1 = diag(sss); % 将前k个奇异值转换为对角矩阵
end
```

图 2 封装的 svd_compression 函数

```
% 定义一个函数，用于处理图像的一个颜色通道
function re = process_channel(a, k)
    [u, s, v] = svd(a); % 对颜色通道进行奇异值分解
    [u1, s1, v1] = svd_compression(u, s, v, k); % 获取前k个奇异值及其对应的左右奇异向量
    re = uint8(u1 * s1 * v1'); % 通过前k个奇异值和对应的左右奇异向量重构颜色通道
end
```

图 3 封装的 process channel 函数

```
clear % 清除工作空间的变量
clc % 清除命令窗口

a = imread('ikun.jpg'); % 读取名为'Swan.jpg'的图像文件，并将其存储在变量a中

k = 10; % 设置奇异值的数量，这将决定压缩后的图像质量
[~,~,dim] = size(a); % 获取图像的维度

imshow(a); % 显示原始图像
title('原图') ; % 设置图像标题为'原图'

a = double(a); % 将图像数据转换为双精度数据，以进行数学运算

% 对图像的每个颜色通道进行处理
for i = 1:dim
    rea(:,:,i) = process_channel(a(:,:,i), k); % 调用process_channel函数处理颜色通道，并将结果存储
end

figure; % 创建新的图像窗口
imshow(rea); % 显示压缩后的图像
title(['图像压缩后, k=',num2str(k)]) ; % 设置图像标题，显示压缩后的奇异值数量

filename = ['kunnitaimei_RGB_k=', num2str(k), '.jpg']; % 生成与k值相关的文件名
imwrite(rea, filename); % 将压缩后的图像保存为生成的文件名
```

图 4 主函数代码

RGB 与灰度图的不同之处仅仅在于有三个通道，对三个通道单独作用即可。

四、结果与对比

灰度图

对一个可爱小女孩头像 lovely.jpg（169KB）进行压缩

图 5 lovely.jpg

$k = 100$ 时，压缩结果如下：

图 6 lovely_k=100.jpg

文件大小压缩至 67KB，压缩了 60.4%。画质得到了最大程度的保留，几乎看不出压缩。效果很好。

$k = 50$ 时，压缩结果如下：

图 7 lovely_k=50.jpg

文件大小压缩至 59KB，又压缩了 11.9%。画质得到了一定保留，效果不错。

$k = 10$ 时，压缩结果如下：

图 8 lovely_k=10.jpg

文件大小进一步压缩至 44KB，进一步压缩了 25%。但是女孩儿变得不可爱了。

RGB 图像

原图 ikun.jpg，大小 22KB

图 9 ikun.jpg

压缩后效果图如下：

图 10 k=100

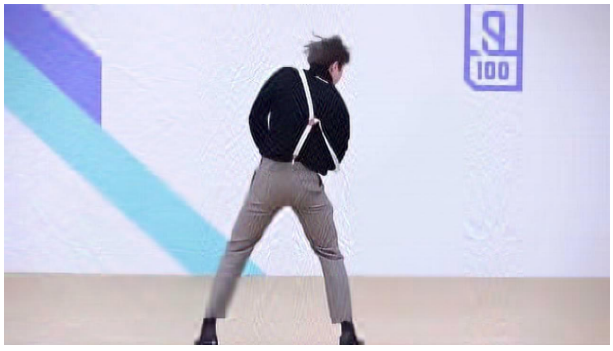

图 11 k=50

图 12 k=10

然而，图片大小并未变小，不减反增。

图 13 压缩文件大小异常

猜测原因可能有如下两条，但都是瞎猜：

1. 编码效率降低：原始的 RGB 图像数据可能有较高的编码效率，而经过 SVD 处理后，原有的编码方式可能被改变，导致编码效率降低，进而影响文件大小。

2. 压缩算法选择：SVD 是一种线性代数的分解方法，它本身并不直接涉及压缩算法。如果在应用 SVD 之后没有采用有效的压缩算法，或者所采用的压缩算法不适合处理 SVD 结果的数据结构，也可能导致最终的文件大小没有减少。

不过，换用别的图片压缩，文件大小显著见效。根据这个事实，猜测与 ikun.jpg 中大量重复出现的相同颜色像素有关，jpeg 压缩已经对其进行了非常好的处理，在经 SVD 分解后反而难以被 jpeg 很好地压缩。

五、结论

本实验通过应用奇异值分解（SVD）技术对黑白和彩色图像进行压缩，探讨了图像的矩阵表示及其存储机制。实验结果表明，SVD 在图像压缩方面的应用是高效且可行的。通过对黑白图像 lovely.jpg 进行 SVD 压缩，我们观察到随着奇异值数量的减少，图像文件大小相应减小，尽管画质有不同程度的损失，但整体上压缩效果显著，尤其是在保留大部分画质的情况下实现了高达 60.4%的压缩率。

对于彩色图像 ikun.jpg，虽然采用类似的 SVD 方法进行了处理，但是最终的文件大小并未如预期般减小。这可能归因于编码效率的降低以及后续压缩算法选择不当。原始 RGB 图像可能具有较高的编码效率，而 SVD 处理后改变了其编码方式，导致效率下降。此外，SVD 作为线性代数的分解手段，并不直接涉及压缩算法；如果不采用适合的压缩算法，或已有算法不适配 SVD 结果的数据结构，也可能导致文件大小未减反增。

综上所述，SVD 技术在图像压缩领域展现出较大的潜力，尤其对于灰度图像

能实现有效的数据简化与压缩。然而，对于彩色图像而言，仍需进一步研究合适的编码与压缩策略，以优化文件大小并保证图像质量。未来的工作可以集中在改善压缩算法以及探索更多适用于彩色图像 SVD 处理后的编码技术上。