

浙江大学

本科实验报告

图像恢复重建

课程名称:	人工智能
姓名:	刘星烨
学院:	计算机科学与技术
专业:	计算机科学与技术
学号:	3180105954
指导教师:	吴飞

浙江大学实验报告

课程名称: _____ 人工智能 _____ 实验类型: _____
实验项目名称: _____ 图像恢复重建 _____
姓名: _____ 刘星烨 _____ 专业: _____ 计算机科学与技术 _____ 学号: _____ 3180105954 _____
同组学生姓名: _____ 指导教师: _____ 吴飞 _____
实验地点: _____ 实验日期: _____

1 问题重述

对图像添加噪声，并对添加噪声的图像进行基于模型的去噪

- 生成受损图像
 - 受损图像 (X) 是由原始图像添加了不同噪声遮罩 (noise masks) ($M \in \mathbb{R}^{H \times W \times C}$) 得到的 ($X = I \odot M$), 其中 \odot 是逐元素相乘。
 - 噪声遮罩仅包含 0,1 值。对原图的噪声遮罩的可以每行分别用 0.8/0.4/0.6 的噪声比率产生的, 即噪声遮罩每个通道每行 80%/40%/60% 的像素值为 0, 其他为 1。
- 使用算法模型, 进行图像恢复。
- 评估误差为所有恢复图像 (R) 与原始图像 (I) 的 2-范数之和, 此误差越小越好。其中 $(:)$ 是向量化操作, 其他评估方式包括 Cosine 相似度以及 SSIM 相似度。

2 设计思想

通过线性回归进行图相恢复。本次实验采用了二元线性回归的方法。本质上是把图片理解成为多个区域, 判断像素是否为噪声点, 随后对每个区域内的数据进行线性回归。

- 假设有 3 个高斯函数构成了每一行的图相也就是说对于每一个像素都是 $w_i * \Phi_{ii}(x_i)$ $i=1,2,3$ 即每一行每个高斯函数对应的权重 * 每个高斯函数 x_i 时的概率
- 然后通过最小二乘法, 利用没有被污染的像素, 也就是说正确的值来计算出权重 w_i ;
- 再用这个权重 w_i * 每个高斯函数 x_i (被污染的像素) 来恢复图像;

二元线性回归思想:

Maximum likelihood and least squares

- Assume: $t = y(\mathbf{x}, \mathbf{w}) + \epsilon$ $y(\mathbf{x}, \mathbf{w}) = \sum_{j=0}^{M-1} w_j \phi_j(\mathbf{x}) = \mathbf{w}^T \boldsymbol{\phi}(\mathbf{x})$
- Thus: $p(t|\mathbf{x}, \mathbf{w}, \beta) = \mathcal{N}(t|y(\mathbf{x}, \mathbf{w}), \beta^{-1}) \rightarrow \mathbb{E}[t|\mathbf{x}] = \int t p(t|\mathbf{x}) dt = y(\mathbf{x}, \mathbf{w})$
- For data set $\mathbf{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_N\}$ and target vector $\mathbf{t} = (t_1, \dots, t_N)^T$, the likelihood function:

$$p(\mathbf{t}|\mathbf{X}, \mathbf{w}, \beta) = \prod_{n=1}^N \mathcal{N}(t_n | \mathbf{w}^T \boldsymbol{\phi}(\mathbf{x}_n), \beta^{-1})$$

$$\ln p(\mathbf{t}|\mathbf{w}, \beta) = \sum_{n=1}^N \ln \mathcal{N}(t_n | \mathbf{w}^T \boldsymbol{\phi}(\mathbf{x}_n), \beta^{-1}) = \frac{N}{2} \ln \beta - \frac{N}{2} \ln(2\pi) - \beta E_D(\mathbf{w})$$

SSE: sum-of-squares error function

$$E_D(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^N \{t_n - \mathbf{w}^T \boldsymbol{\phi}(\mathbf{x}_n)\}^2 = \frac{1}{2} \|\mathbf{t} - \Phi \mathbf{w}\|^2$$

Figure 1: 二元线性回归算法

3 代码内容

3.1 添加噪声

根据噪声比率，先生成不同噪声比的图像矩阵。再将数据进行分割重组，得到满足各通道条件的噪声矩阵。最后将矩阵与原图像矩阵相乘进行遮罩。

```
def noise_mask_image(img, noise_ratio):
    """
    根据题目要求生成受损图片
    :param img: 图像矩阵，一般为 np.ndarray
    :param noise_ratio: 噪声比率，可能值是0.4/0.6/0.8
    :return: noise_img 受损图片，图像矩阵值 0-1 之间，数据类型为 np.
            array, 数据类型对象 (dtype): np.double, 图像形状:(height,width,
            channel),通道(channel) 顺序为RGB
    """
    # 受损图片初始化
    noise_img = None

    # -----实现受损图像答题区域-----
    row, col = img.shape[0], img.shape[1]
    rgb = [None, None, None] # rgb
    for i in range(3):
        # 构造其中一个通道的噪声图
        rgb[i] = np.random.choice(2, (row, col), p=[noise_ratio, 1-
            noise_ratio])

    # 扩展 shape
    for i in range(3):
        rgb[i] = rgb[i][:, :, np.newaxis]
    # 合并
    rst = np.concatenate((rgb[0], rgb[1], rgb[2]), axis=2)
    noise_img = rst * img
    # -----
```

```
return noise_img
```

3.2 图像恢复

划分数据区域，对每个 $n*n$ 范围内数据分析，将非噪声点加入回归的训练集和，使用回归结果对噪声点进行恢复。

在实践中，我尝试了多种半径进行处理，得到的效果区别不大。

```
def restore_by_region(res_img, noise_mask, radius = 4):
    res_img = np.copy(noise_img)
    rows, cols, channel = res_img.shape
    radius = 10 # 10 * 10
    row_cnt = rows
    col_cnt = cols

    for chan in range(channel):
        for rn in range(int((row_cnt + radius - 1) / radius)):
            ibase = rn * radius
            if rn == int((row_cnt + radius - 1) / radius):
                ibase = rows - radius
            for cn in range(int((col_cnt + radius - 1) / radius)):
                jbase = cn * radius
                if cn == int((col_cnt + radius - 1) / radius):
                    jbase = cols - radius
                x_train = []
                y_train = []
                x_test = []
                for i in range(ibase, ibase + radius):
                    for j in range(jbase, jbase + radius):
                        if noise_mask[i, j, chan] == 0: # 噪音点
                            x_test.append([i, j])
                            continue
                        x_train.append([i, j])
                        y_train.append([res_img[i, j, chan]])
                if x_train == []:
                    print("x_train is None")
                    continue
                reg = LinearRegression()
                reg.fit(x_train, y_train)
                pred = reg.predict(x_test)
                for i in range(len(x_test)):
                    res_img[x_test[i][0], x_test[i][1], chan] = pred[i][0]
            res_img[res_img > 1.0] = 1.0
            res_img[res_img < 0.0] = 0.0
    return res_img

def restore_image(noise_img, size=4):
    """
    使用 区域二元线性回归模型 进行图像恢复。
    :param noise_img: 一个受损的图像
    :param size: 输入区域半径，长宽是以 size*size 方形区域获取区域，
        默认是 4
    :return: res_img 恢复后的图片，图像矩阵值 0-1 之间，数据类型为 np
        .array,
        数据类型对象 (dtype): np.double, 图像形状:(height,width,
        channel), 通道(channel) 顺序为RGB
    """
```

```
# 恢复图片初始化，首先 copy 受损图片，然后预测噪声点的坐标后作为  
# 返回值。  
res_img = np.copy(noise_img)  
  
# 获取噪声图像  
noise_mask = get_noise_mask(noise_img)  
  
# -----实现图像恢复代码答题区域  
# -----  
res_img = restore_by_region(res_img, noise_mask, 10)  
# -----  
return res_img
```

4 实验结果

通过网站测试点测试。