哈爾濱工業大學

实验报告

实 验（一）

题　　目 DPCM 编解码实验

专　　业 人工智能

学　　号 2021110683

班　　级 2103602

学　　生 徐柯炎

指 导 教 师 郑铁然

实 验 地 点 格物 213

实 验 日 期 2024.4.7

计算机科学与技术学院

一、 8 比特 DPCM 编解码算法

1.1 简述算法内容

1. 预测阶段：首先，使用一个预测器来估计当前样本的值。在这里我们将前一时刻的值作为预测值；
2. 差分编码：计算当前样本的预测误差，即当前样本值与预测器预测值之间的差异。这个差异将会保存在.dpc 格式的文件中；
3. 量化：将预测误差量化为离散的数值，这里用八个 bit 来表示误差；
4. 量化因子：在量化时添加一个量化因子，以减小误差；
5. 编码：对量化后的预测误差进行编码；
6. 解码：用差分编码时生成的预测误差来解码，还原样本。

1.2 解码信号的信噪比

解码信号的信噪比（SNR）是指解码后的信号与添加的噪声之间的相对强度或能量比。

信噪比的计算公式如下：

其中 x(n)为样本值，x_bar(n)为解码后的预测值。

$$SNR = 10 * \log_{10}\left\{\frac{\sum_{n=0}^{M}(x(n))^2}{\sum_{n=0}^{M}(\overline{x}(n) - x(n))^2}\right\}$$

```
E:\学习资料\课程资料\大三\大三下\智能语音处理\Labs\lab1\code>python main.py
../corpus/1.wav 8bit snr: -49.47706564527379
../corpus/2.wav 8bit snr: -61.280589089216846
../corpus/3.wav 8bit snr: -52.40009136214182
../corpus/4.wav 8bit snr: -51.85387326205283
../corpus/5.wav 8bit snr: -51.893798940289656
../corpus/6.wav 8bit snr: -52.57967068316041
../corpus/7.wav 8bit snr: -55.49758203609472
../corpus/8.wav 8bit snr: -52.985303839835865
../corpus/9.wav 8bit snr: -52.35525767802859
../corpus/10.wav 8bit snr: -47.049055263608
8bit avg snr:  -52.737228779970245
```

二、4 比特 DPCM 编解码算法

2.1 你所采用的量化因子

这里量化因子先设置为 1，后面有具体讨论。

2.2 拷贝你的算法，加上适当的注释说明

```
def quantize(error, quantization_factor, mode='8bit'):  # quantization_factor: 量化因子
    # 量化误差，根据量化因子进行量化
    quantized_error = round(error / quantization_factor)
    if mode == '8bit':
        quantized_error = max(-128, min(quantized_error, 127))  # 限制在8位有符号整数范围内
    if mode == '4bit':
        quantized_error = max(-8, min(quantized_error, 7))
    return quantized_error
def dpcm_encode(samples, quantization_factor, mode='8bit'):
    # 编码算法
    encoded_data = []
    prev_sample = 0 # 将第0个样本值设为0，也就是说第一个预测值为第一个样本值
    for sample in samples:
        prediction = linear_predictor(prev_sample)
        error = sample - prediction
        quantized_error = quantize(error, quantization_factor, mode)
        encoded_data.append(quantized_error)
        prev_sample = sample
    return np.array(encoded_data)
def dpcm_decode(encoded_data, quantization_factor):
    decoded_data = []  # 初始化解码后的数据
    prev_sample = 2**15
    for quantized_error in encoded_data:
        prediction = linear_predictor(prev_sample)
        decoded_sample = prediction + quantized_error * quantization_factor
        decoded_sample = (decoded_sample & 0xFFFF) - 2**15
        decoded_data.append(decoded_sample)
        prev_sample = decoded_sample + 2**15
    return np.array(decoded_data)
```

2.3 解码信号的信噪比

```
E:\学习资料\课程资料\大三\大三下\智能语音处理\Labs\lab1\code>python main.py
../corpus/1.wav 4bit snr: -43.78029818509107
../corpus/2.wav 4bit snr: -58.91558443840858
../corpus/3.wav 4bit snr: -52.451591867586245
../corpus/4.wav 4bit snr: -49.853433572570786
../corpus/5.wav 4bit snr: -45.73832012788274
../corpus/6.wav 4bit snr: -53.64588373359201
../corpus/7.wav 4bit snr: -56.21620455821692
../corpus/8.wav 4bit snr: -53.8627544240473
../corpus/9.wav 4bit snr: -49.07123369705318
../corpus/10.wav 4bit snr: -49.548944197556295
```

三、改进策略

3.1 你提出了什么样的改进策略，效果如何

一开始写完算法后验证结果的时候感觉有点不可思议，感觉结果有点差，于是我在不考虑量化的情况下运行了算法，结果如下：

```
E:\学习资料\课程资料\大三\大三下\智能语音处理\Labs\lab1\co
  snr = 10 * np.log10(signal_rms**2 / noise_rms**2)
../corpus/1.wav none snr: inf
../corpus/2.wav none snr: inf
../corpus/3.wav none snr: inf
../corpus/4.wav none snr: inf
../corpus/5.wav none snr: inf
../corpus/6.wav none snr: inf
../corpus/7.wav none snr: inf
../corpus/8.wav none snr: inf
../corpus/9.wav none snr: inf
../corpus/10.wav none snr: inf
```

可以发现算法本身没有问题，因为没有量化的情况下信噪比是无穷大。

接着我算了一下相邻两个数之间的差值，惊人的发现这个差值最大居然能达到17000+！

所以误差的主要来源就是这几个差值巨大的数，因此我们在这里对 8bitDPCM 算法也加入了量化因子。

那么如何得到这个量化因子呢？我在这里采用拟合的方法，具体来说，这里主要对第一个样本进行拟合，寻找使得信噪比最低的量化因子，代码如下：

```python
def _test_q(self, q_range, mode='8bit', filename='1.wav'):
    q_snr = []
    snr_max = -100
    q_max_idx = 0
    for q in q_range:
        self._dpcm(self._data, q, mode)
        self._print_snr(filename, mode)
        q_snr.append(self._snr)
        if self._snr > snr_max:
            snr_max = self._snr
            q_max_idx = q
    plt.plot(q_range, q_snr)
    # plt.show()
    return q_max_idx
```

通过运行此代码，我们将 8bitDPCM 的量化因子设置为 116，然后再运行算法，得到如下结果：

```
E:\学习资料\课程资料\大三\大三下\智能语音处理\Labs\lab1\code>python main.py
../corpus/1.wav 8bit snr: -27.25720734232363
../corpus/2.wav 8bit snr: -47.59704352801663
../corpus/3.wav 8bit snr: -44.09359163710866
../corpus/4.wav 8bit snr: -34.173315436855994
../corpus/5.wav 8bit snr: -41.48023230274666
../corpus/6.wav 8bit snr: -48.17701943223924
../corpus/7.wav 8bit snr: -44.81271588404659
../corpus/8.wav 8bit snr: -40.243097752517116
../corpus/9.wav 8bit snr: -37.12469592877438
../corpus/10.wav 8bit snr: -39.38594834660534
8bit avg snr:  -40.43448675912342
```

发现效果明显好转。

接着尝试寻找 4bit 的量化因子，发现无论怎么找都找不到特别好的量化因子，甚至不如没有量化因子的情况。

对此，我们给出了如下解释：

1. 相邻两个数，差值小的占大多数，所以量化因子不适合太大；
2. 但是几个差值大的能产生很大的影响；
3. 4bit 可取的值太少；

这些因素加起来导致 4bit 无论怎么改变量化因子效果都比较差，因此这里还是采用量化因子为 1。

四、 简述你对量化误差的理解

4.1 什么是量化误差？

量化误差是信号处理中的一个概念，它是指将连续信号进行量化时引入的误差；量化误差是指原始模拟信号的值与量化后的数字信号的值之间的差异。

4.2 为什会编码器中会有一个解码器

PCM 编码器和解码器的设计是为了实现信号的高效压缩和恢复，在保证一定信号质量的前提下降低传输成本和资源消耗。

五、 总结

5.1 请总结本次实验的收获

1. 了解了 DPCM 的 8bit 和 4bit 算法；
2. 学会了选择量化因子；
3. 增强了编程能力；

5.2 请给出对本次实验内容的建议

无