12.1 Recap From Last Class

PPM
Assigns probabilities to characters in a string according to their context. Uses a fixed number of previous k characters as context. Encodes with a context table to derive conditional probabilities. Some techniques adopted include:

- Using escape character to notify the decoder to trim down the scope
- Avoid multiple escapes with exclusion, i.e. exclude impossible candidate characters for a certain context. This can be done because we know which characters could have caused the escape character to be sent, and which ones could not have.
- Use of arithmetic code to utilize the high probability of result.

Lempel-Ziv
Dictionary-based approach (or sliding window with LZ77). Some high level ideas of LZ77:

1. Encoding: keeps a sliding window (look ahead buffer) of size b, and a dictionary of size d. For each match, output a pair \((p, l, c)\) for the longest matched substring in the dictionary, where \(p\) is the offset (to the left) from the start of the sliding buffer, \(l\) is the length of the match, \(c\) being the next character in the buffer that’s beyond the match. Then advance the window \(b\) by \(l + 1\).

2. Decoding: also keeps the dictionary, and inserts the matched substring copy at the end of the dictionary.

3. Optimization: allows to have a substring match that is longer than the size of the window (Allowing matched substring starting at the sliding window to overlap with the matched in the dictionary).

For LZSS:

1. Encoding: outputs either :
   
   (a) \((0, \text{position}, \text{offset})\) for the substring matched, OR
   
   (b) \((1, \text{char})\) for a character. Choose the latter encoding if \(\text{length} < 2\).
Burrows-Wheeler

Reorders characters in a string in a reversible way, to place similar characters closer to each other, thus allowing for better compression. It does so by breaking a file into fix-sized blocks, and then considering full context for each character (by wrapping around), and sorting these character contexts.

12.2 Overview

In this lecture, we will finish the topic on data compression, and move on to the next topic hashing, with the introduction of Markov’s Inequality. We will now consider lossy compression.

12.3 Quantization

Quantization is a process in which we drop some bits of information.

Where is quantization useful? (Page 13)

Currently getting more traction in the field of machine learning. With deep learning, the weights of the trained neural network could enjoy the benefit of quantization to achieve a better performance, with dropping of bits per weight value.

12.3.1 Vector Quantization

Mapping a multi-dimensional space into a smaller set of messages $S$ is called vector quantization. It usually takes in the below two steps:

- Selecting a set of representatives: the representatives could be both dynamic or static for a set of messages sent.
- Determine for each message sequence, what representative to be used.

How are representatives selected? (Page 15)

 Typically using a clustering algorithm to come up with a codebook. It will largely depend on how the error metrics are chosen. For example, the representatives could be chosen so that the $L_2$ distance is used as an error metrics.

When is vector quantization the most effective? (Page 16)

It is most effective when the variables along the dimensions of the space are highly correlated, so that representatives in the concentrated areas will make much sense.
What would a loseless version of quantization be? (Page 17)

One can include in the message the selected representatives, and the distance (offset) the true value is from the representative.

### 12.4 Linear Transform Coding

Let $A$ be the transform matrix, and $x$ be the input vector. $\Theta$ being the transformed vector.

$$\Theta = Ax^T = \begin{bmatrix} a_{0,0} & a_{0,1} & a_{0,2} & \ldots & a_{0,n} \\ a_{1,0} & a_{1,1} & a_{1,2} & \ldots & a_{1,n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{m,0} & a_{m,1} & a_{m,2} & \ldots & a_{m,n} \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} a_{0,x} & a_{1,x} & \ldots & a_{m,x} \end{bmatrix}$$

$a_{0,x}$ (inner product) is then the coefficient for a particular basis. One can choose to drop the low coefficient to control the level of lossiness, or drop those base which we "know" will be less useful (i.e. the high frequency that human eyes are not sensitive to).

There are several transforms we could choose (see Page 20 for examples). Choosing the right one depends on the data.

#### 12.4.1 JPEG case study

Why is the cosine transform done on blocks of pixels rather than the entire image? (Page 22)

As pictures are not periodic, but the cosine function is. So if the entire image is transformed by the periodic cosine function, too much of the information will be lost.

#### 12.5 Video Compression - MPEG

Why is B frame useful? (Page 30)

Part of the intermediate P-Frames might be completely missing from the previous frames. Since P-Frames only look at previous frames, anything from the following frames will not be considered. B-Frames will look for information in both direction, so that it will hopefully pick up information that is only present in the future frames. For example, consider a car entering a shot from the side, and for a pair of consecutive I-frames, in which one does not have the car, and the other has the car present completely. We would ideally want to use P-frames for the intermediate scene, and the B-frame will be able to capture it while P-frame not.
12.6 Hashing

For the hashing unit, we will start with learning useful concentration bounds, and then later learn how to apply them in various hashing problems. Concentration bounds try to answer the following question:

What is the probability that a random variable deviates much from its expected value?

The first bound we will learn is Markov’s Inequality.

12.6.1 Markov’s Inequality

This is a basic, but very useful inequality.

Theorem 12.1 (Markov’s Inequality). Let $X$ be a non-negative random variable with mean $\mu$. Then

$$P[X \geq \alpha] \leq \frac{\mu}{\alpha}.$$ 

Proof intuition for Markov’s Inequality

$$\mu = E[X] = \sum x_i p(x_i)$$

$$= \sum_{x_i \leq \alpha} x_i p(x_i) + \sum_{x_j > \alpha} x_j p(x_j)$$

$$\geq \alpha \sum p(x_j)$$

$$\geq \alpha P(X \leq \alpha)$$

$$\rightarrow P(X \leq \alpha) \leq \frac{\mu}{\alpha}.$$ 

(12.1)