Lecture 11

Lossy Compression Basics and Quantization

Announcements

Quiz Q1

Unmatched Literals	Match Length	Match Offset
AABBB	4	1
-	5	9
CDCD	2	2

1. What is the output string after decoding the first row? Ans: AABBBBBBB
2. What is the output string after decoding the second row? Ans: AABBB
3. What is the output string after decoding the third row? Ans: CDCDCD

Quiz Q2

Unmatched Literals	Match Length	Match Offset
TA	1	`X1`
W	`X2`	3
-	2	2
`X3`	-	-

1. What is match offset `X1` ? Ans: 2
2. What is match length `X2` ? Ans: 1
3. What are the unmatched literals `X3` ? Ans: DI

Quiz Q3

Consider an English text. Also consider a reversibly transformed version where each byte is replaced with itself plus one. So A becomes B, B becomes C and so on.

- Zstd would perform similarly on both the original text and the transformed version.
  ( ) True
  ( ) False
- A LLM-based compressor trained on English would perform similarly on both the original text and the transformed version.
  ( ) True
  ( ) False

Quiz Q4

Your company produces a lot of data of a particular kind, and you are asked to find ways to efficiently compress it to save on bandwidth and storage costs. Which of these is a good approach to go about this?

( ) Use CMIX since your company deserves the best possible compression irrespective of the compute costs.

( ) Use gzip because LZ77 is a universal algorithm and so it has to be the best compressor for every occasion.

( ) Make a statistically accurate model of your data, design a predictor, and then use a context-based arithmetic coder.

( ) Understand the application requirements, try existing compressors like zstd and then evaluate whether there are benefits to create a domain specific compressor based on an approximate model for the data.

Recap

So far learnt about lossless compression and tradeoffs for various *entropy coders*

- Learnt about fundamental limits on lossless compression: H(p)
- Thumb rule: $L(x) \propto \lceil log_2(1/p(x)) \rceil$
- Learn about various lossless compressors aka *entropy coders* and their implementations
  - Block codes: Shannon coding and Huffman coding
  - Streaming codes: Arithmetic coding and Asymmetric Numeral Systems
  - Universal (pattern) matching codes: LZ77
- Learnt about how to deal with non-IID sources
  - Context-based coding
  - Adaptive coding

Lossy Compression Basics

Second-half of the course will look into lossy compression

- Lossless compression is a special case of lossy compression
- Everything we learnt about lossless compression is applicable to lossy compression, and in-fact, all lossy compressors deploy entropy coding as a final step
    - we will learn more about this in the coming lectures
- Lossless compression assumed discrete sources but in practice we deal with continuous sources - sensors, images, audio, video, etc.

Quiz-1: How much information does a continuous source X contain?

Lossy Compression Basics

Second-half of the course will look into lossy compression

- Lossless compression is a special case of lossy compression
- Everything we learnt about lossless compression is applicable to lossy compression, and in-fact, all lossy compressors deploy entropy coding as a final step
    - we will learn more about this in the coming lectures
- Lossless compression assumed discrete sources but in practice we deal with continuous sources

Quiz-1: How much information does a continuous source X contain?

Ans: Infinite! Fact about real numbers: they are uncountable, i.e., there are infinite real numbers between any two real numbers.

Lossy Compression Basics

So we **cannot** represent a continuous source exactly. We need to approximate it.

Approximation imply *loss* of information.

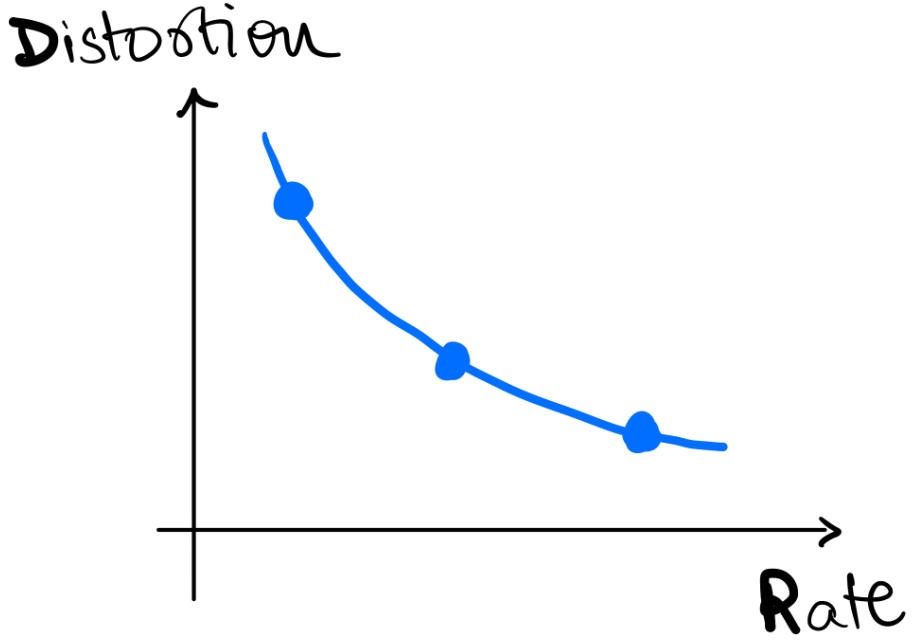
- Distortion (D) is a measure of loss of information, e.g.
  - MSE: $D = \mathbb{E}[(X - \hat{X})^2]$, MAE: $D = \mathbb{E}[|X - \hat{X}|]$, etc.

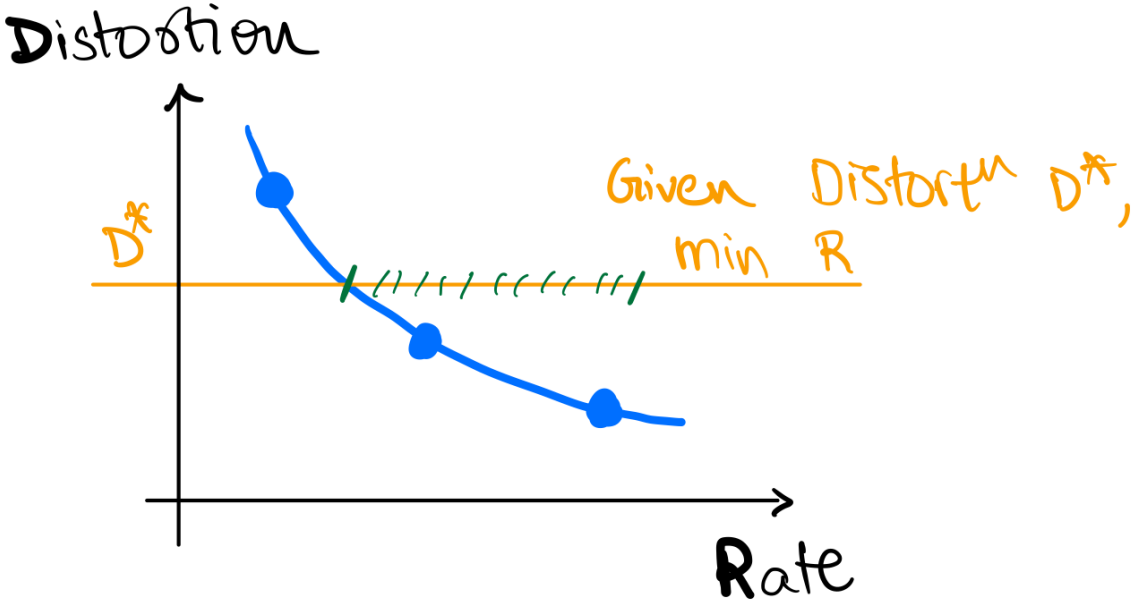
Also, this means we have a choice on how much information we *want* or are *OK* to lose.

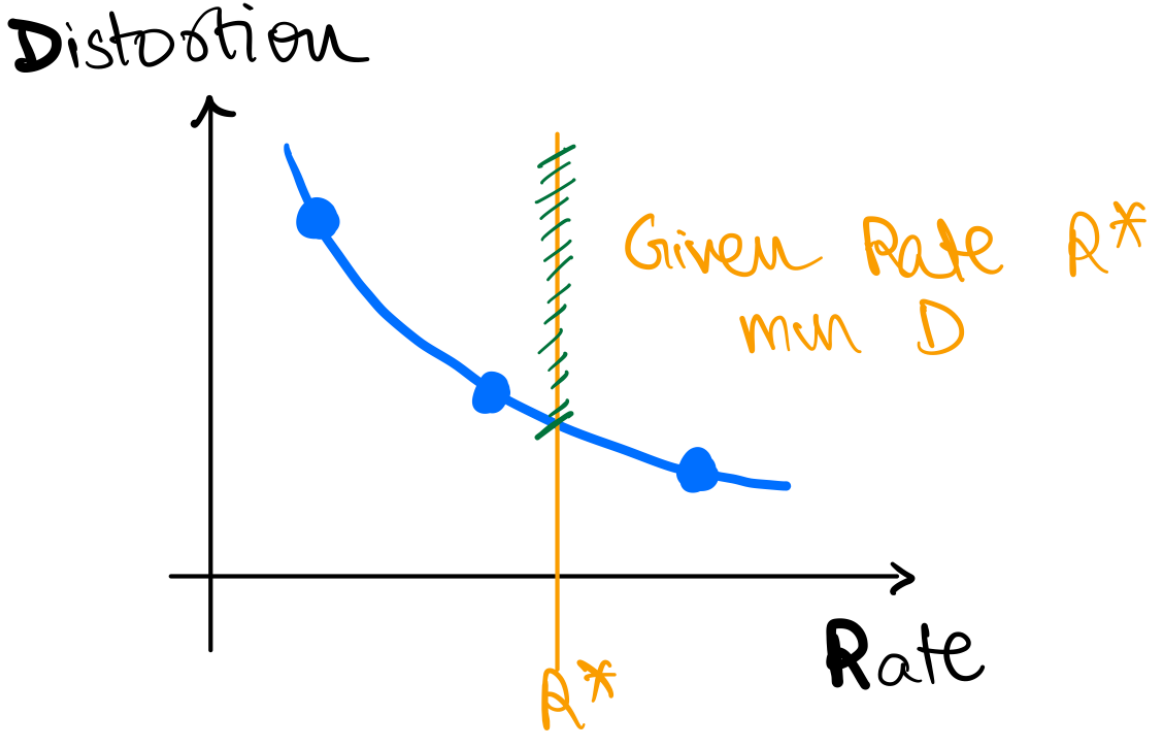
- Rate (R) is the number of bits used to represent the source
- Higher rate, imply we should be to represent the source more accurately, i.e., lower distortion

This is the fundamental *rate-distortion tradeoff* in lossy compression.

Rate-Distortion Tradeoff or RD curve




Distortion
D*
Given Distort^n D*,
min R
Rate


Distortion
Given Rate R*
min D
R*
Rate

RD Example

`Example 1`

- Let's say you are measuring temperature (`T`) in a room, say in Celsius, at an hourly interval.
    - Remember, physical `T` is a continuous source.
- Say your sensor is very sensitive and it records `T = [38.110001, 36.150901, 37.122020, 37.110862, 35.827111]`

Quiz-2: How many bits do we want to represent `T` ?

RD Example

`Example 1`

- Let's say you are measuring temperature (`T`) in a room, say in Celsius, at an hourly interval.
  - Remember, physical `T` is a continuous source.
- Say your sensor is very sensitive and it records `T = [38.110001, 36.150901, 37.122020, 37.110862, 35.827111]`

Quiz-2: How many bits do we want to represent using `T` ?

Ans: Depends on the application! If we are using it to control the AC, we might need more bits than if we are using it to decide whether to wear hoodie or T-shirt. In either case,

- we need to decide on the *distortion* we are OK with
- we can agree these many decimals are waste of bits

RD Example

`Example 1`

- Let's say you are measuring temperature (`T`) in a room, say in Celsius, at an hourly interval.
    - Remember, physical `T` is a continuous source.
- Say your sensor is very sensitive and it records `T = [38.110001, 36.150901, 37.122020, 37.110862, 35.827111]`

Quiz-3: What are some reasonable values to encode?

RD Example

`Example 1`

- Let's say you are measuring temperature (`T`) in a room, say in Celsius, at an hourly interval.
    - Remember, physical `T` is a continuous source.
- Say your sensor is very sensitive and it records `T = [38.110001, 36.150901, 37.122020, 36.827111, 35.201022]`

Quiz-3: What are the reasonable values to encode?

Ans: We can decide to round `T` to the nearest integer, i.e., `T_lossy = [38, 36, 37, 37, 35]` . This is similar to converting `T` to `int` from `float` .

Quantization

- What we did in the previous example is called *quantization*.
  - Quantization is the process of mapping a continuous source to a discrete source.
- Quantization is a lossy process, i.e., it introduces distortion.
- Quantization is a fundamental operation in lossy compression!
- Quantized values are also sometimes called *symbols* or *codewords*, and the set of available quantized values is called *codebook* or *dictionary*.
  - In previous example, *codebook* is `{35, 36, 37, 38}` and *codewords* for each symbol are `{35, 36, 37, 37, 35}`.

Quiz-4: For a codebook of size `N`, what is the rate `R`?

Quantization

- What we did in the previous example is called *quantization* (or *binning*).
  - Quantization is the process of mapping a continuous source to a discrete source.
- Quantization is a lossy process, i.e., it introduces distortion.
- Quantization is a fundamental operation in lossy compression!
- Quantized values are also sometimes called *symbols* or *codewords*, and the set of quantized values is called *codebook* or *dictionary*.
  - In previous example, *codebook* is `{35, 36, 37, 38}` and *codewords* for each symbol are `{35, 36, 37, 37, 35}`.

Quiz-4: For a codebook of size `N`, what is the rate `R`?

Ans: $R = \log_2(N)$. Alternatively, we can say the quantized value for each symbol can take 2^R unique values.

Quantization Example - Gaussian

`Example 2`

- Now say, X is a Gaussian source with mean 0 and variance 1, i.e., $X \sim \mathcal{N}(0, 1)$
- Say we want to represent X using just 1 bit per symbol.

Quiz-5: What are some reasonable values to encode?

Quantization Example - Gaussian

`Example 2`

- Now say, X is a Gaussian source with mean 0 and variance 1, i.e., $X \sim \mathcal{N}(0, 1)$
- Say we want to represent X using just 1 bit per symbol.

Quiz-5: What are some reasonable values to encode?

Ans: We can decide to convey just the sign of X, i.e., $\hat{X} = \text{sign}(X)$ as the distribution is symmetric around 0.

Quantization Example - Gaussian

`Example 2`

- Now say, X is a Gaussian source with mean 0 and variance 1, i.e., $X \sim \mathcal{N}(0, 1)$
- Say we want to represent X using just 1 bit per symbol.
- Encode $\hat{X}$ using the sign of X, i.e., $\hat{X} = \text{sign}(X)$
- Say we get a positive value for $\hat{X}$, what should be the *quantized value* of the recovered symbol?
    - Need to decide on the *distortion* we are OK with, say MSE distortion

Quiz-6: What should be the codebook?

Quantization Example - Gaussian

`Example 2`

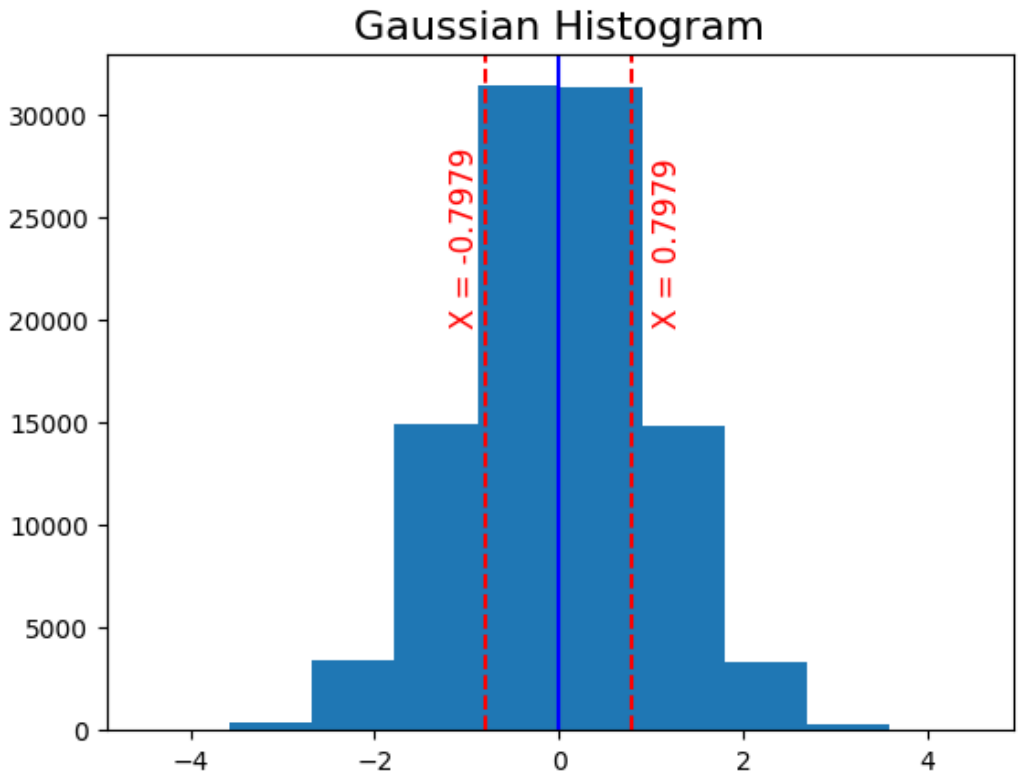
- Now say, X is a Gaussian source with mean 0 and variance 1, i.e., $X \sim \mathcal{N}(0, 1)$
- Say we want to represent X using just 1 bit per symbol.
- Encode $\hat{X}$ using the sign of X, i.e., $\hat{X} = \text{sign}(X)$
- Say we get a positive value for $\hat{X}$, what should be the *quantized value* of the recovered symbol?
    - Need to decide on the *distortion* we are OK with, say MSE distortion

Quiz-6: What should be the codebook?

Ans: Codebook $\mathscr{C} = \left\{\mathbb{E}[(X|\hat{X} > 0)], \mathbb{E}[(X|\hat{X} < 0)]\right\}$.

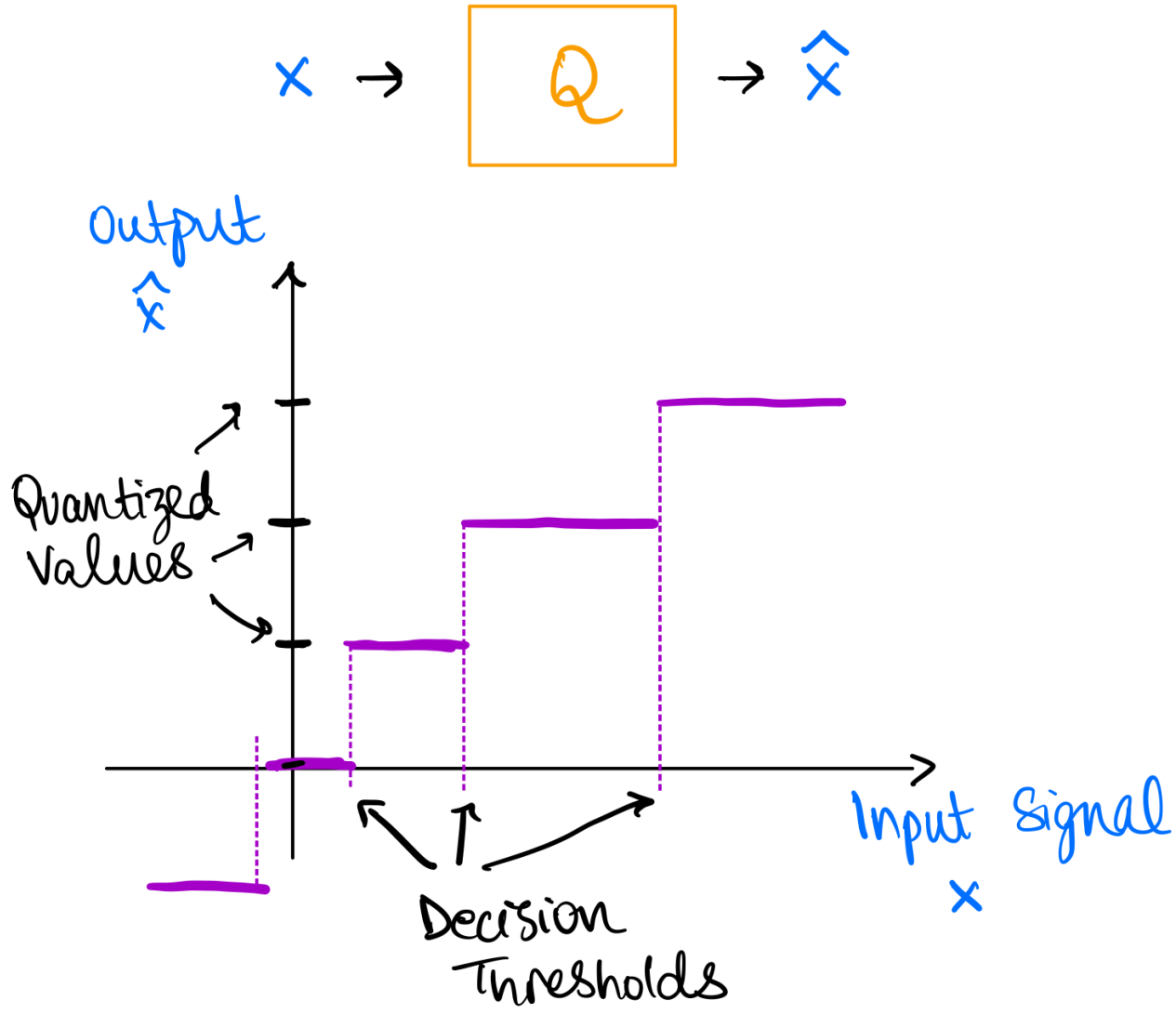
For gaussian, this is $\mathscr{C} = \left\{\sqrt{\frac{2}{\pi}}, -\sqrt{\frac{2}{\pi}}\right\}$.

Quantization Example - Gaussian



Scalar Quantization

- This is an example of *scalar quantization*.
- We are quantizing each symbol independently. But can we do better?



Vector Quantization

- Maybe we can work with two (or more) symbols at a time?
- Say we have $X = [X_1, X_2]$, where $X_1, X_2 \sim \mathcal{N}(0, 1)$
  - you can also think of it as you generated `2*N` samples from $\mathcal{N}(0, 1)$ and then split them into two groups of size `N` (similar to `block codes` in lossless compression)
  - or you can think of it as you have two sensors measuring the same source
  - or you can think of it as having two sensors measuring two different sources

Quiz-7: I want to compare it with 1 bit/symbol scalar quantization. What's the size of codebook allowed?

Vector Quantization

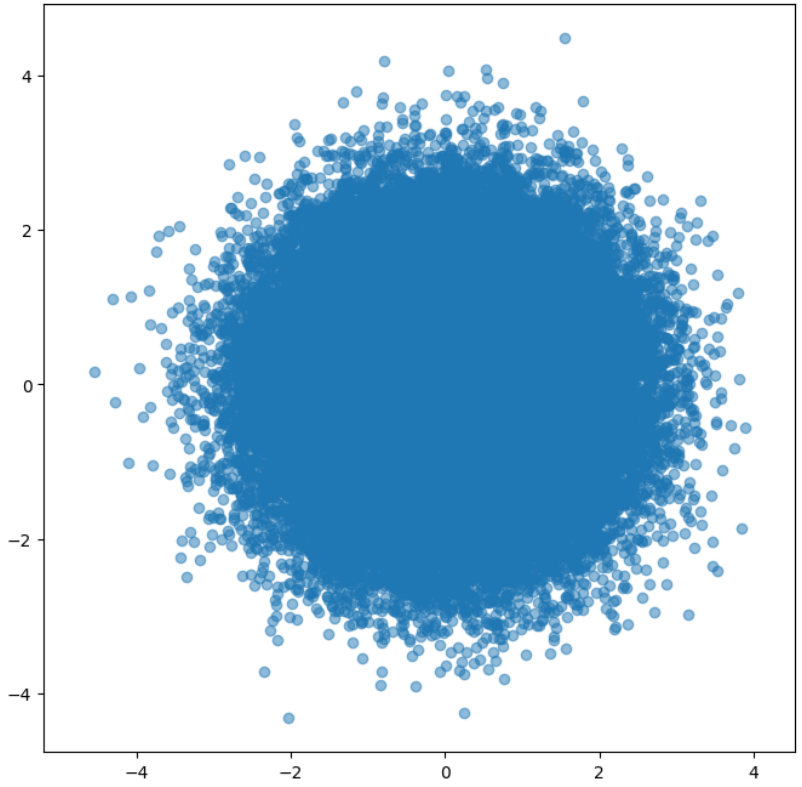
- Maybe we can work with two (or more) symbols at a time?
- Say we have $X = [X_1, X_2]$, where $X_1, X_2 \sim \mathcal{N}(0, 1)$
  - you can also think of it as you generated `2*N` samples from $\mathcal{N}(0, 1)$ and then split them into two groups of size `N` (similar to `block codes` in lossless compression)
  - or you can think of it as you have two sensors measuring the same source
  - or you can think of it as having two sensors measuring two different sources

Quiz-7: I want to compare it with 1 bit/symbol scalar quantization. What's the size of codebook allowed?

Ans: $2^{1 \text{ bit/symbol} \times 2 \text{ symbol/code-vector}} = 4$. Generalizing, we can have codebook of size $N = 2^{R*k}$ for vectors (blocks) of size k and R bits/symbol.

Vector Quantization

We can have codebook of size $N = 2^{R*K}$ for vectors (blocks) of size k and R bits/symbol. In other words, $R = (\log_2 N)/k$ bits/symbol.



Vector Quantization (formally)

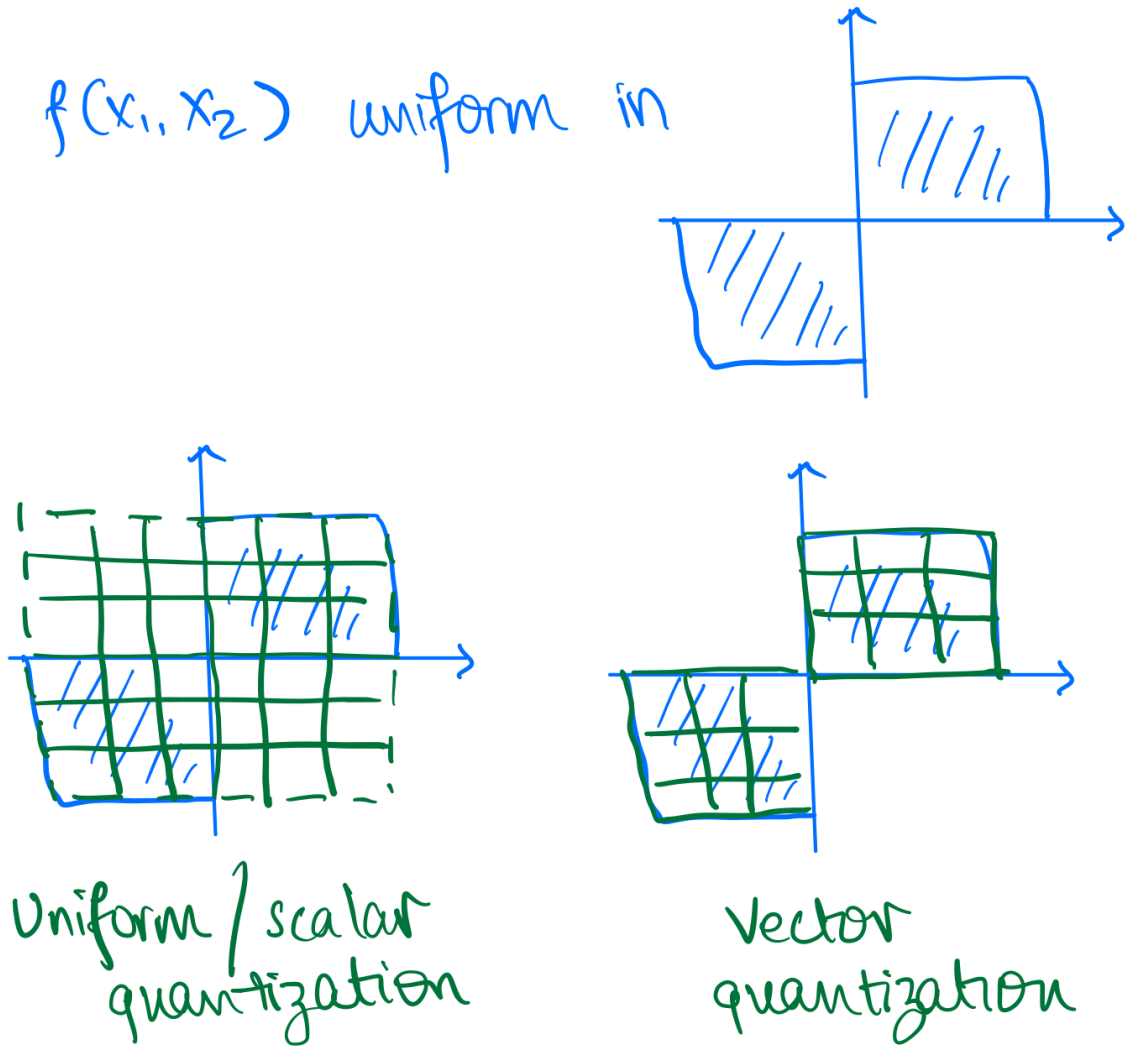
- A quantizer is a mapping $Q : \mathbb{R}^k \to \mathscr{C}$ where $\mathscr{C} = \left\{ \underline{y}_i \right\}_{i=1}^{N}$ is the "codebook" or "dictionary" comprising of N k-dimensional vectors.
- The mapping is defined by: $Q(\underline{x}) = \underline{y}_i$ if $\underline{x} \in S_i$ where $\{S_i\}_{i=1}^{N}$ is a partition of $\mathbb{R}^k$ $\bigcup_{i=1}^{N} S_i = \mathbb{R}^k; \quad S_l \cap S_m = \phi, l \neq m.$
- The rate is $R = \frac{\log N}{k} \frac{\text{bits}}{\text{sample}}; N = 2^{kR}$
- Let's map this to the examples we looked at.

Vector Quantization

Benefits:

- We can exploit dependence between vector components
- We can have more general decision regions (than could be obtained via Scalar Quantization)

Vector Quantization Example



Vector Quantization

Some comments (without proofs):

- Optimal regions are generally not uniform (as in scalar quantization) even in simple uniform IID case!
- In the 2D case of uniform IID, a hexagonal lattice provides most optimal regions with respect to MSE distortion.
    - This is called *lattice quantization* or *Voronoi diagram* and can accommodate more than 2 dimensions.
    - $\frac{MSE_{lattice}}{MSE_{SQ}} \approx 0.962$ for 2D case.

Vector Quantization

***Abstract*—The quantization of n-dimensional vectors in R^n with an arbitrary probability measure, under a mean-square error constraint, is discussed. It is demonstrated that a uniform, one-dimensional quantizer followed by a noiseless digital variable-rate encoder ("entropy encoding") can yield a rate that is, for any n, no more than 0.754 bit-per-sample higher than the rate associated with the optimal n-dimensional quantizer, regardless of the probabilistic characterization of the input n-vector for the allowable mean-square error.**

- Vector quantization can be a lot of hard work with little gain. Be careful of the tradeoff between complexity and gain.
- Abstract from: On universal quantization, Ziv

Vector Quantization -- Practical Considerations

- In general, optimal regions are not easy to compute, and we need to resort to iterative algorithms.

  Quiz-8: Have you seen this problem before in some other context?

K-means Algorithm

- It's same as K-means clustering algorithm in ML! Also called as *Lloyd-Max* algorithm or *Generalized Lloyd* algorithm.
  - You want to cluster data points into `N` clusters corresponding to codebook (`k` in k-means) such that the average distortion is minimized.
- Historical Note:
  - First proposed by Stuart Lloyd in 1957 (motivated by audio compression) at Bell Labs
  - Was widely circulated but formally published only in 1982
  - Independently developed and published by Joel Max in 1960, therefore sometimes referred to as the Lloyd-Max algorithm
  - Generalized Lloyd specialized to squared error is the Kmeans clustering algorithm widely used in Machine Learning

K-means Algorithm

- Given some data points, we can compute the optimal codebook and the corresponding partition of the data points.
- Main idea is to do each-step iteratively:
  - Given a codebook, compute the best partition of the data points
  - Given a partition of the data points, compute the optimal codebook
  - Repeat until convergence

K-means Algorithm

```
def k_means(data, k, max_iterations=100):
    centroids = initialize_centroids(data, k) # some random initialization for centroids (codebook)
    for iteration in range(max_iterations): # some convergence criteria
        # Assign data points to the nearest centroid -- this is the partition step
        clusters = assign_data_to_centroids(data, centroids)
        # Calculate new centroids -- this is the codebook update step
        new_centroids = calculate_new_centroids(data, clusters)
        if np.allclose(centroids, new_centroids): # some convergence criteria
            break
        centroids = new_centroids # update centroids
    return clusters, centroids
```

Note: `k` is the size of the codebook (referred to as `N` in the rest of lecture) and `data` is the set of data points.

K-means Algorithm

```
def k_means(data, k, max_iterations=100):
    centroids = initialize_centroids(data, k) # some random initialization for centroids (codebook)
    for iteration in range(max_iterations): # some convergence criteria
        # Assign data points to the nearest centroid -- this is the partition step
        clusters = assign_data_to_centroids(data, centroids)
        # Calculate new centroids -- this is the codebook update step
        new_centroids = calculate_new_centroids(data, clusters)
        if np.allclose(centroids, new_centroids): # some convergence criteria
            break
        centroids = new_centroids # update centroids
    return clusters, centroids

def initialize_centroids(data, k):
    # Randomly select k data points as initial centroids
    return data[np.random.choice(len(data), k, replace=False)]

def assign_data_to_centroids(data, centroids):
    # Assign each data point to the nearest centroid
    distances = np.linalg.norm(data[:, np.newaxis] - centroids, axis=2)
    clusters = np.argmin(distances, axis=1)
    return clusters

def calculate_new_centroids(data, clusters):
    # Calculate new centroids as the mean of data points in each cluster
    new_centroids = np.array([data[clusters == i].mean(axis=0) for i in range(len(np.unique(clusters)))])
    return new_centroids
```

Example Notebook

https://colab.research.google.com/drive/16dYjBEc499HgHoZRxcyeg0YmNAb5AwAW?usp=sharing

More resources

We only scratched the surface of quantization. Many more advanced topics:

- Constrained vector quantization
- Predictive vector quantization
- Trellis coded quantization
- Generalized Lloyd algorithm
- ...

For more details, see:

- Fundamentals of Quantization: Gray
- Vector Quantization and Signal Compression: Gersho, Gray

Next time

We will look into the question of what is the fundamental limit on lossy compression, i.e. what is the best possible rate-distortion tradeoff.