CS 6351 DATA COMPRESSION

THIS LECTURE:
LOSSY COMPRESSION FRAMEWORK & SCALAR QUANTIZATION

Instructor: Abdou Youssef

OBJECTIVES OF THIS LECTURE

By the end of this lecture, you will be able to:

- Describe the 3-stage general framework of lossy compression, and what every stage does
- Explain quantization and dequantization, and their roles and purposes within the lossy compression framework
- Design and develop different types of scalar quantizers: uniform, semi-uniform, and non-uniform
- Carry out quantization/dequantization, given a quantizer
- Apply the Max-Lloyd algorithm for deriving an optimal quantizer of a given number of levels for a given input data stream

OUTLINE

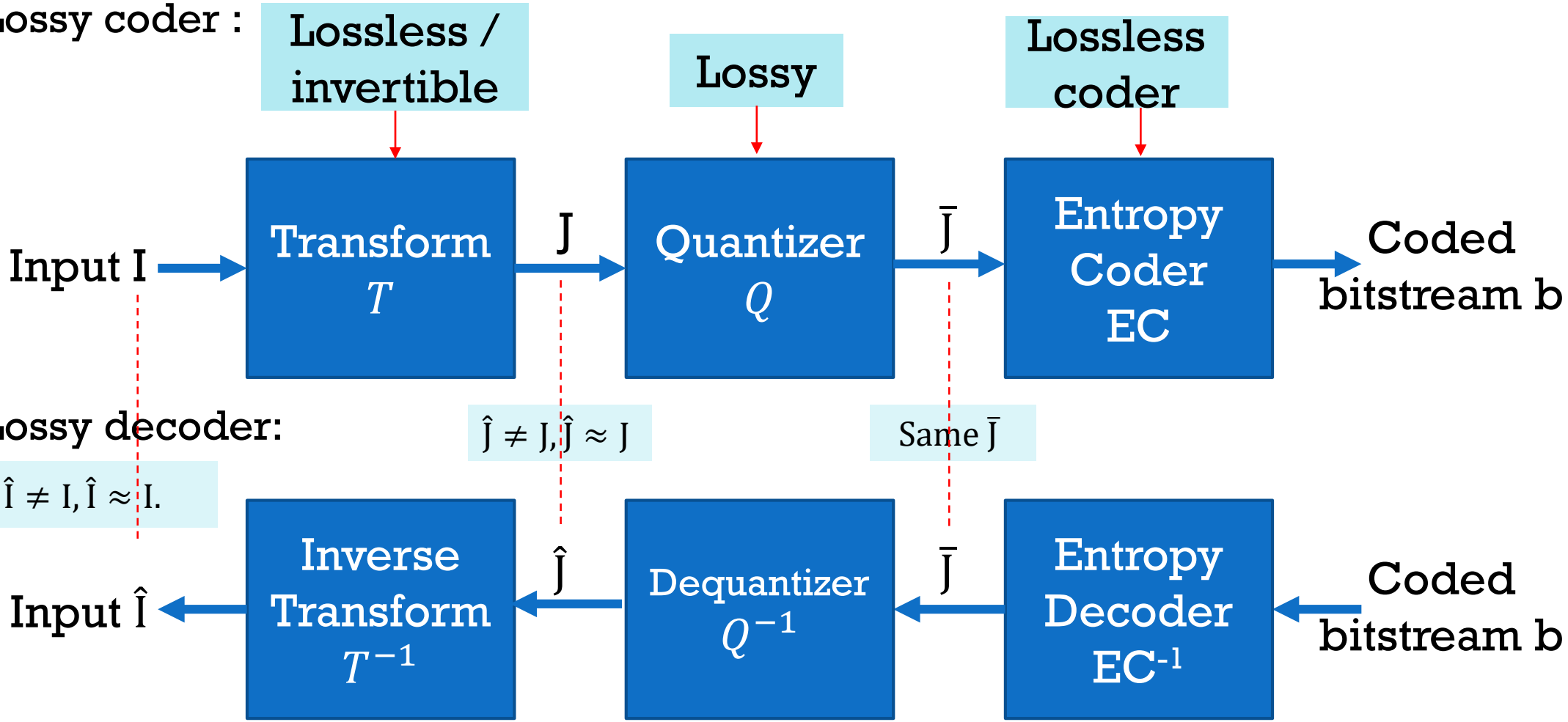
- The general framework of lossy compression
- The role and purpose of each stage in the lossy compression framework
- Scalar quantization
- Definition and operations of three different types of scalar quantizers: uniform, semi-uniform, and non-uniform
- Optimal Max-Lloyd non-uniform quantizers
  - Mathematical derivation and algorithm

LOSSY COMPRESSION
-- MOTIVATION --

- We saw that lossless compression does not have high enough compression ratio (or low enough bitrate) needed for demanding applications like in images and videos, especially when we get into increasingly higher definition
- Also, lossless compression cannot compress every input stream, while we want to have control in imposing any target bitrate at will or as needed
- Lossless compression has no way of exploiting the characteristics/limitations of the human audio-visual systems
- The top two reasons force us to look for alternatives to lossless compression, and the third reason opens the door for lossy compression, which, among other things, affords us the ability to control the bitrate at will (in tradeoff with quality)

GENERAL SCHEME OF LOSSY COMPRESSION

- Lossy coder :



- Lossy decoder:

The entire data loss is limited to the quantizer

THE ENTROPY CODER

- It is a fancy name for any lossless coder, but the idea is to use in that component an already proved, high-performing lossless coder, such as Huffman, Arithmetic Coding, RLE, etc.
- We have studied lossless coding to great depths
- So no more attention is needed to that part
- Rather, the rest of the study of lossy compression will focus on the other two components:
  - Transforms (and their inverses), and
  - Quantization/dequantization

QUANTIZATION

- This will be covered in this lecture in great detail
- For now, suffice it to say that this component reduces the precision of the data to save on bits
  - A crude example is to change every 8-bit number into a 2-bit number by keeping only the 2 most significant bits
  - With 8 bits, you have 256 (i.e., 2^8) levels of gray, while with 2 bits you can have 4 levels of gray, so it is a lot less resolution/precision
- The quantizer is a lossy component, as just illustrated by the above example
  - The original input of the quantizer can never be recovered fully
- In lossy compression, the data loss occurs only in the quantization stage
- Confining the loss to just one module/stage makes it a lot easier to study and control the process of data loss and its effect

TRANSFORMS (1/2)

- Transforms are a huge subject, and we'll spend several weeks studying them
- For now, let's outline the desirable properties of transforms
  - **Decorrelation of data**
  - **Separation of data** into
    - vision-sensitive data (low-frequency data)
    - vision-insensitive data (high-frequency data)
  - **Energy compaction**: concentrating the important data into a very small subset
- Advantage of decorrelation of the data
  - Quantizing decorrelated data does not cause blurring of contrasts (e.g., edges) as much as quantizing correlated data
  - In general, quantizing decorrelated data causes less loss of audio/visual patterns than quantizing correlated data

TRANSFORMS (2/2)

- Advantages of separation of data into vision-sensitive (i.e., important data) and vision-insensitive (i.e., unimportant data):
  - We can quantize more the less important data (thus achieving a lot of compression)
  - We can quantize lightly the more important data (thus retaining quality/fidelity)
    - With energy compaction, only a small subset of the data needs to be quantized lightly, which enables us to preserve quality while still compressing a lot
  - We can progressively adjust the harshness of the quantization based on the level of importance of the data
- In summary, a good transform paves the way for us to quantize more intelligently so that for the same target BR, we maximize the quality
- Without transforms, we can still quantize, but the quantization is "blind" and thus leads to damaging needlessly the reconstructed data quality

BACK TO QUANTIZATION

- There are two broad groups of quantizers
  - Scalar quantizers
  - Vector quantizers
- In this lecture we will cover scalar quantizers
  - It is because most quantizers used in lossy compression are scalar quantizers
- Vector quantizers will be covered near the end of the semester
  - They are useful, but have found limited use (like in audio compression)

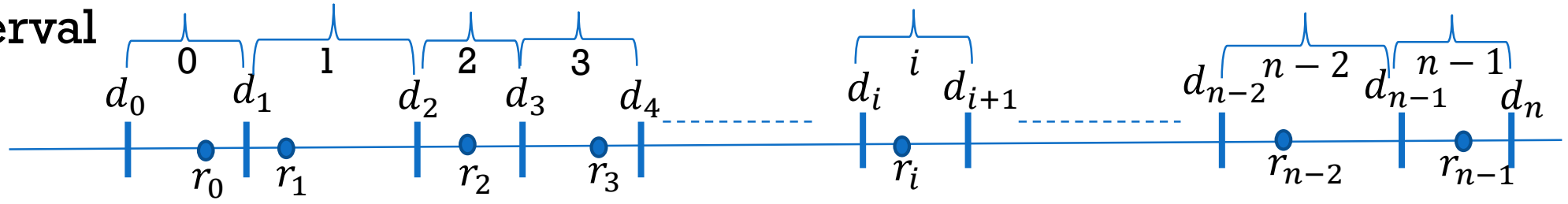
SCALAR QUANTIZATION
-- SPECIFICATION --

Definition: An n-level quantizer Q is typically characterized by n+1 ***decision levels*** $d_0, d_1, d_2, \ldots d_n$ where $d_0 < d_1 < d_2 < \cdots < d_n$, and by n ***reconstruction levels*** $r_0, r_1, r_2, \ldots, r_{n-1}$.

- The d_i's divide the range of data under quantization into k consecutive intervals $[d_0\ d_1), [d_1\ d_2), \ldots, [d_{n-1}\ \ d_n)$.

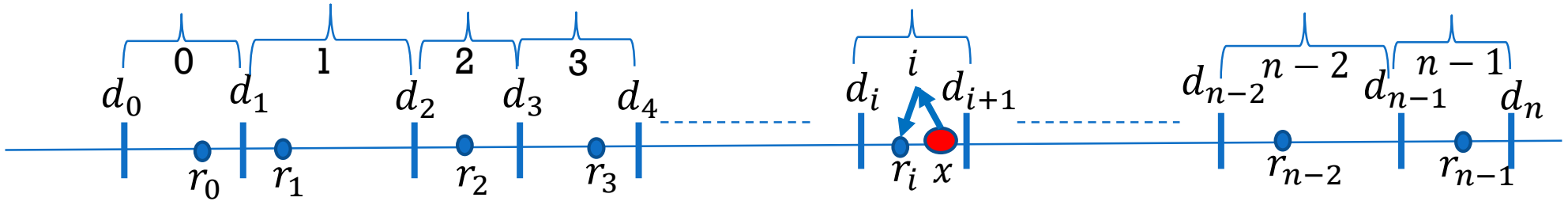
An interval [a b) is the set of all the numbers x where a≤x<b

- The intervals are labeled $0, 1, 2, \ldots, n-1$: interval $[d_i\ d_{i+1})$ is labeled i
- Each r_i is in interval $[d_i\ d_{i+1})$, and can be viewed as the "centroid" of its interval



SCALAR QUANTIZATION
-- QUANTIZATION AND DEQUANTIZATION PROCESS--

- The quantizer Q is a function (aka, mapping) that maps any real number $x \in [d_0\ d_n)$ into the unique integer $i \in \{0,1,2,\dots,n-1\}$ such that $x \in [d_i\ \ d_{i+1})$
- The dequantizer (denoted Q^{-1} or DQ) is a mapping from $\{0,1,2,\dots,n\}$ to $\{r_0, r_1, \dots, r_{n-1}\}$ where $Q^{-1}(i) = r_i\ \forall\ i$
- So the quantizer Q "remembers" for a given number x the interval (i) that contains x, and the dequantizer Q^{-1} reconstructs x into the reconstruction value r_i of that interval
- Often we say that the quantizer quantizes x to $\hat{x} = r_i$ (by applying Q then Q^{-1})
- Typically, the reconstructed value $\hat{x} \neq x$ but $\hat{x} \approx x$



SCALAR QUANTIZATION
-- A FEW REMARKS --

- In some applications, $d_0 = -\infty$ and $d_n = \infty$, meaning that the data to be quantized is unbounded from below and from above
- But in most applications, we do have finite upper and lower bounds on the data, in which case, d_0 is the minimum possible data value, and d_n is slightly larger than the maximum data value in the application
- We can have d_n to be = the maximum data value in the application, and so the rightmost interval will be $[d_{n-1}\ \ d_n]$ instead of $[d_{n-1}\ \ d_n)$
- That is OK because the n intervals remain non-overlapping

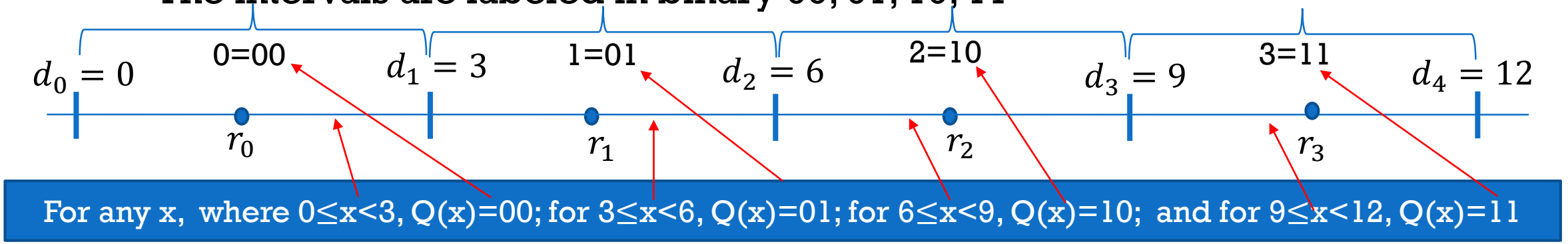
SCALAR QUANTIZATION
-- AN EXAMPLE (1/3) --

- Suppose we have a data set (e.g., the pixels of a given image), where
  - The data **ranges between 0 and 12**,
  - The data can be **real** data (i.e., not just integers), and
  - The data is **uniformly** (i.e., evenly) **distributed** in that range
- Need to quantize it so that each number is represented with just **2 bits**
- Design a quantizer for that data set

SCALAR QUANTIZATION
-- AN EXAMPLE (2/3) --

- Answer:
  - Because of the 2-bit requirement, the quantizer must have $2^2 = 4$ intervals
  - Because the data is evenly distributed, it is better that the intervals be equal and the reconstruction values be the mid-points of the intervals
  - Thus, Q is specified by $d_0 = 0,\ \ d_1 = 3,\ d_2 = 6,\ \ \ d_3 = 9,\ \ \ d_4 = 12,$

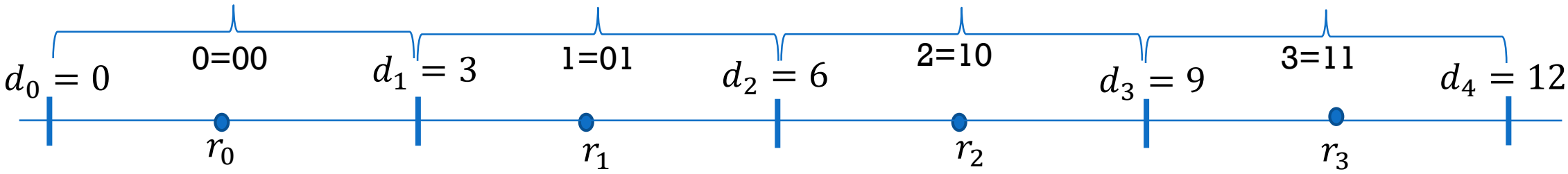
    $r_0 = 1.5,\ r_1 = 4.5,\ \ r_2 = 7.5,\ r_3 = 10.5.$
  - The intervals are labeled in binary 00, 01, 10, 11



SCALAR QUANTIZATION
-- AN EXAMPLE (3/3) --

- How big can the error get between original and reconstructed data?
- Well, let's look at a few values of x and $\hat{x}$ first:
- The maximum error seems to be 1.5
- Since every interval is of length 3, and the reconstruction values are mid points, the error is for sure at most half an interval, i.e., 1.5

x	Q(x)	$\hat{x}$	|error|
4	01	4.5	0.5
3	01	4.5	1.5
2.75	00	1.5	1.25
9.5	11	10.5	1

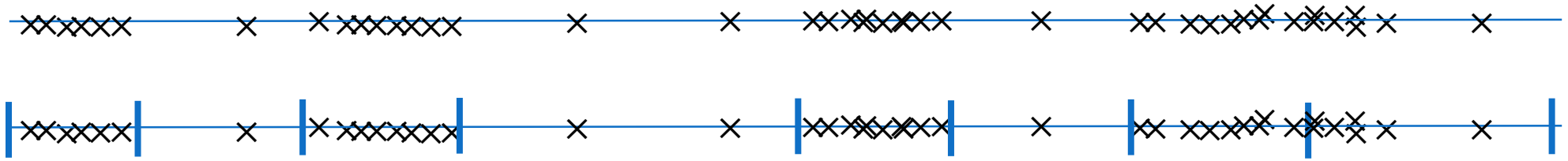


SCALAR QUANTIZATION
-- REMARKS ABOUT INTERVAL SIZES (1/2) --

- What happens when intervals are very small or very large?
- If all the intervals are very small, then there are two implications:
  1. The error will then be correspondingly small (as we saw in the example, where the error is up to half-intervals), leading to better quality of reconstructed data
  2. The number of intervals will be larger (to cover the whole range of our data), which means that we need more bits to represent each quantized value
     - The number of bits to represent each interval is log(# intervals)
     - So, less compression and higher bitrate
- And vice versa: larger intervals lead to higher error but more compression
- This tradeoff is not surprising: less compression, better quality of reconstructed data; more compression, worse quality.

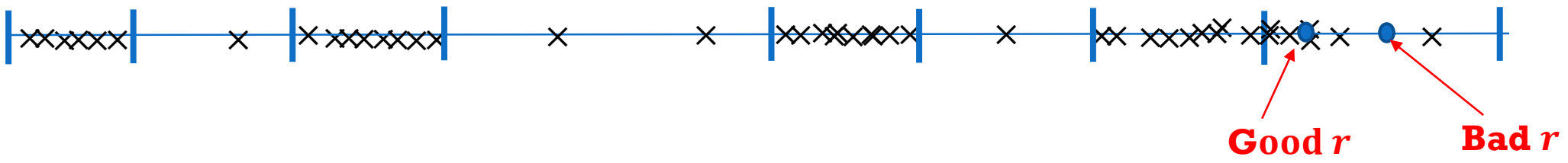
SCALAR QUANTIZATION
-- REMARKS ABOUT INTERVAL SIZES (2/2) --

- Which is better, equal intervals, or intervals of varying sizes?
- If the data is distributed evenly in the range, then equal intervals is ideal
- But if the data clusters in a few regions, then we would have less error on average if the intervals are small in dense regions of the data, and large intervals in less dense regions of the data
- Example. Consider this data, and assume an 8-interval quantizer:



SCALAR QUANTIZATION
-- REMARKS ABOUT INTERVAL CENTROIDS --

- Which is better, mid-interval centroids r_i's, or "more representative" centroids?
- The arguments are similar to the interval lengths
- If the data is distributed evenly in the range, then mid-interval centroids are ideal
- But if the data is skewed within intervals, then the reconstruction values need to shift close to the dense parts of an interval to minimize errors



SCALAR QUANTIZATION
-- LESSONS FROM THE LAST FEW OBSERVATIONS --

- There are several types of scalar quantizers, depending on whether the intervals are uniform in length and whether the centroids of the intervals are midpoints
- In the case where data is not uniformly distributed, we need a way to find optimal interval placement and centroid placement
- Those two questions will be addressed next

TYPES OF SCALAR QUANTIZERS

- Uniform quantizers
- Semi-uniform quantizers
- Non-uniform quantizers
  - Optimal non-uniform quantizers

UNIFORM QUANTIZERS (1/2)

- **Definition**: a quantizer is ***uniform*** if
  - all the intervals are of equal length Δ (i.e., $d_1 - d_0 = d_2 - d_1 = d_3 - d_2 = \cdots = \Delta$), and
  - the reconstruction values are the mid-points (centers) of their intervals
- A uniform quantizer is characterized by either (d_0, d_n, n) or (d_0, d_n, Δ):
  - From (d_0, d_n, n) we get (d_0, d_n, Δ): $\Delta = \frac{d_n - d_0}{n}$. And vice versa: $n = \frac{d_n - d_0}{\Delta}$
  - From either (d_0, d_n, n) or (d_0, d_n, Δ) we can get:
    - $d_i = d_0 + i\Delta$, for all $i = 1,2, \dots, n$.
    - $r_i = \frac{d_i + d_{i+1}}{2} = \frac{d_i + (d_i + \Delta)}{2} = d_i + \frac{\Delta}{2} = \mathrm{d}_0 + \left(\mathrm{i} + \frac{1}{2}\right)\Delta$, for all $i = 0,1,2, \dots, n-1$
- Simple algorithm for quantizing and dequantizing:
  - $Q(x) = \lfloor \frac{x - d_0}{\Delta} \rfloor$ // this gives i where $x \in [d_i \ \ d_{i+1})$ //Exercise: Prove that
  - $Q^{-1}(i) = \mathrm{d}_0 + \left(i + \frac{1}{2}\right)\Delta$ // that is because $Q^{-1}(i) \stackrel{\text{def}}{=} r_i$ and $r_i = \mathrm{d}_0 + \left(\mathrm{i} + \frac{1}{2}\right)\Delta$

UNIFORM QUANTIZERS (2/2)

- Uniform quantizers are
  - efficient to store/represent (only store (d_0, d_n, n))
  - Fast to quantize/dequantize with: each takes O(1) time

- Therefore, if the data is uniformly distributed, or nearly so, a uniform quantizer is quite desirable

NON-UNIFORM QUANTIZERS

- **Definition**: a quantizer ***is non-uniform*** if one or both of the following conditions are met
  - the decision intervals are not of equal size
  - the reconstruction levels are not all the centers of their intervals
- Algorithm for quantizing/dequantizing
  - Quantization: to find the value (i) of $Q(x)$ for a given input data value x, do binary search for x in the sorted array $d_0, d_1, d_2, \ldots, d_n$. This search finds d_i where $x = d_i$ or $d_i < x < d_{i+1}$, and returns i. Time: $O(\log n)$
  - Dequantization: $Q^{-1}(i) = r_i$, which takes a constant time (*aka*, $O(1)$)

SEMI-UNIFORM QUANTIZERS

- **Definition**: a quantizer ***is semi-uniform*** if
  - the decision intervals are all of equal size
  - the reconstruction levels are not all the midpoints of their intervals
- Characterization: $(d_0, d_n, n, r_0, r_1, \ldots r_{n-1})$
  - Much as in the case of uniform quantizers, $\Delta = \frac{d_n - d_0}{n}$ and $d_i = d_0 + i\Delta \ \forall i$
- Algorithms for quantizing/dequantizing
  - Quantization: as in uniform quantizers, $Q(x) = \lfloor \frac{x - d_0}{\Delta} \rfloor$, which takes O(1) time
  - Dequantization: $Q^{-1}(i) = r_i$, which takes a constant time (O(1))

ILLUSTRATION OF THE 3 TYPES OF QUANTIZERS (1/4)

- Take the data stream X=[0 0.01 2.8 3.4 1.99 3.6 5 3.2 4.5 7.1 7.9]

X:	0	0.01	2.8	3.4	1.99	3.6	5	3.2	4.5	7.1	7.9

- Denote by
  - IX the quantized indices (i.e., the i's) of the elements of X
  - $\hat{X}$ the reconstructed data of X
- We will consider 4-level quantizers of three types, where the data range is between 0 and 8 (i.e., $d_0 = 0, d_4 = 8$)
- We measure the performance using MSE:

$$\mathrm{MSE}(X, \hat{X}) = \frac{1}{N}\sum_{k=1}^{N}(x_k - \hat{x}_k)^2$$

ILLUSTRATION OF THE 3 TYPES OF QUANTIZERS (2/4)

- Uniform:

d_0	d_1	d_2	d_3	d_4
0	2	4	6	8

r_0	r_1	r_2	r_3
1	3	5	7

X:	0	0.01	2.8	3.4	1.99	3.6	5	3.2	4.5	7.1	7.9
IX:	0	0	1	1	0	1	2	1	2	3	3
$\hat{X}$:	1	1	3	3	1	3	5	3	5	7	7

MSE = 0.42

- Semi-uniform:

d_0	d_1	d_2	d_3	d_4
0	2	4	6	8

r_0	r_1	r_2	r_3
2/3	3.25	4.75	7.5

X:	0	0.01	2.8	3.4	1.99	3.6	5	3.2	4.5	7.1	7.9
IX:	0	0	1	1	0	1	2	1	2	3	3
$\hat{X}$:	2/3	2/3	3.25	3.25	2/3	3.25	4.75	3.25	4.75	7.5	7.5

MSE = 0.31

ILLUSTRATION OF THE 3 TYPES OF QUANTIZERS (3/4)

- Non-Uniform (optimal quantizer will be derived later):

d_0	d_1	d_2	d_3	d_4
0	1.5	3.87	6.125	8

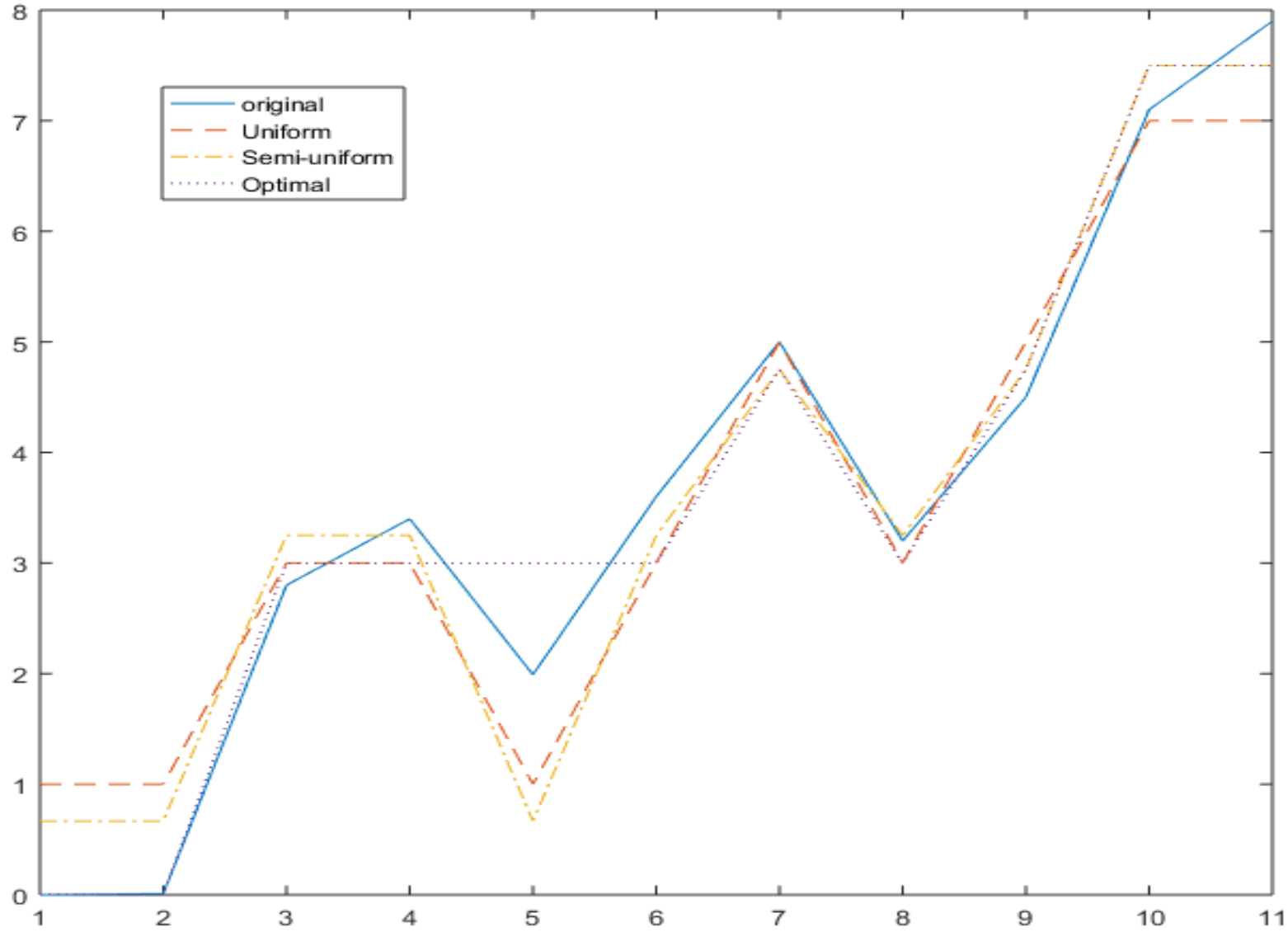
r_0	r_1	r_2	r_3
0.005	2.998	4.75	7.5

X:	**0**	**0.01**	**2.8**	**3.4**	**1.99**	**3.6**	**5**	**3.2**	**4.5**	**7.1**	**7.9**
IX:	0	0	1	1	1	1	2	1	2	3	3
$\hat{X}$:	0.005	0.005	2.998	2.998	2.998	2.998	4.75	2.998	4.75	7.5	7.5

MSE = 0.18

- Comparison of MSE's:
  - Uniform: 0.42
  - Semi-uniform: 0.31
  - Non-uniform: 0.18

ILLUSTRATION OF THE 3 TYPES OF QUANTIZERS (4/4)



OPTIMAL NON-UNIFORM QUANTIZERS
-- MAX-LLOYD QUANTIZERS (1/8) --

- We are interested in finding the optimal n-level quantizer, for a fixed n, for a given data set whose minimum value is denoted d_0 and its maximum value is denoted d_n
- Need to determine the optimal values for $d_1, d_2, \ldots, d_{n-1}$ and $r_0, r_1, \ldots, r_{n-1}$
- Assume the data being quantized follows a probability distribution $p(x)$
- Denote by $\hat{x}$ the reconstructed (i.e., quantized and dequantized) value of x
- The optimization is with respect to the mean-square error (MSE) E

$$E = \int_{d_0}^{d_n} (x - \hat{x})^2 p(x) dx$$

OPTIMAL NON-UNIFORM QUANTIZERS
-- MAX-LLOYD QUANTIZERS (2/8) --

- The (MSE) $E = \int_{d_0}^{d_n} (x - \hat{x})^2 p(x) dx$
- We need to minimize E
- Strategy:
  - E is a function of the quantizer parameters $d_1, d_2, \ldots, d_{n-1}$ and $r_0, r_1, \ldots, r_{n-1}$
  - Compute the partial derives of E with respect to each of the parameters $d_1, d_2, \ldots, d_{n-1}$ and $r_0, r_1, \ldots, r_{n-1}$
  - Set those partial derivatives to 0
  - We get $(2n - 1)$ equations
  - Solve those equations to obtain $d_1, d_2, \ldots, d_{n-1}$ and $r_0, r_1, \ldots, r_{n-1}$
- We will need a good dose of multi-variable calculus
- If you can follow, great; if not, don't worry: we care more about the algorithm that will come after the math

- $2n - 1$ unknowns
- $2n - 1$ equations

OPTIMAL NON-UNIFORM QUANTIZERS
-- MAX-LLOYD QUANTIZERS (3/8) --

- The (MSE) $E = \int_{d_0}^{d_n} (x - \hat{x})^2 p(x) dx$
- We will expand E using

In calculus: $\int_a^b f(x)dx = \int_a^c f(x)dx + \int_c^b f(x)dx$

- $E = \int_{d_0}^{d_1} (x - \hat{x})^2 p(x) dx + \int_{d_1}^{d_2} (x - \hat{x})^2 p(x) dx + \cdots + \int_{d_i}^{d_{i+1}} (x - \hat{x})^2 p(x) dx + \cdots + \int_{d_{n-1}}^{d_n} (x - \hat{x})^2 p(x) dx$
- Noting that for any $x \in [d_i \ \ d_{i+1})$, the reconstructed value $\hat{x} = r_i$, we get
- $E = \int_{d_0}^{d_1} (x - r_0)^2 p(x) dx + \int_{d_1}^{d_2} (x - r_1)^2 p(x) dx + \cdots + \int_{d_i}^{d_{i+1}} (x - r_i)^2 p(x) dx + \cdots + \int_{d_{n-1}}^{d_n} (x - r_{n-1})^2 p(x) dx$
- Let's now compute partial derivatives of E
- Denote by
  - E_{r_i} the partial derivative of E wrt r_i,
  - E_{d_i} the partial derivative of E wrt d_i,

OPTIMAL NON-UNIFORM QUANTIZERS
-- MAX-LLOYD QUANTIZERS (4/8) --

- $E = \int_{d_0}^{d_1}(x-r_0)^2 p(x)dx + \int_{d_1}^{d_2}(x-r_1)^2 p(x)dx + \cdots + \boldsymbol{\int_{d_i}^{d_{i+1}}(x-r_i)^2 p(x)dx} + \cdots + \int_{d_{n-1}}^{d_n}(x-r_{n-1})^2 p(x)dx$
- When computing E_{r_i}, observe that none of the integrals in the sum in E involves r_i, except for the red integral $\boldsymbol{\int_{d_i}^{d_{i+1}}(x-r_i)^2 p(x)dx}$,
- Therefore, E_{r_i}= the derivative of $\int_{d_i}^{d_{i+1}}(x-r_i)^2 p(x)dx$ wrt r_i
- Hence, $E_{r_i} = -2\int_{d_i}^{d_{i+1}}(x-r_i)p(x)dx = -2\left(\int_{d_i}^{d_{i+1}} xp(x)dx - \int_{d_i}^{d_{i+1}} r_i p(x)dx\right)$
- Setting $E_{r_i} = 0$, we get $\int_{d_i}^{d_{i+1}} xp(x)dx = \int_{d_i}^{d_{i+1}} r_i p(x)dx = r_i \int_{d_i}^{d_{i+1}} p(x)dx$
- Therefore $r_i = \dfrac{\int_{d_i}^{d_{i+1}} xp(x)dx}{\int_{d_i}^{d_{i+1}} p(x)dx}$. Looking at this last formula closely, we see:
- **r_i = the average value of the data that fall in interval $[d_i \;\; d_{i+1})$**

If $f(t) = \int_a^b g(t,x)dx$,
$f'(t) = \int_a^b g_t(t,x)dx$
where $g_t(t,x)$ is the partial derivative of $g(t,x)$ wrt t.
$(u^2)' = 2uu'$

OPTIMAL NON-UNIFORM QUANTIZERS
-- MAX-LLOYD QUANTIZERS (5/8) --

- Now, compute E_{d_i}, the partial derivative of E wrt d_i
- $E = \int_{d_0}^{d_1}(x-r_0)^2 p(x)dx + \cdots + \int_{\boldsymbol{d_{i-1}}}^{\boldsymbol{d_i}}\boldsymbol{(x-r_{i-1})^2 p(x)dx} + \int_{\boldsymbol{d_i}}^{\boldsymbol{d_{i+1}}}\boldsymbol{(x-r_i)^2 p(x)dx} + \cdots + \int_{d_{n-1}}^{d_n}(x-r_{n-1})^2 p(x)dx$
- In the above sum, only two successive integrals involve d_i (in red)
- Therefore, when we differentiate E wrt d_i, all the integrals will disappear except the derivatives of those two integrals
- $E_{d_i} = \boldsymbol{(d_i - r_{i-1})^2 p(d_i) - (d_i - r_i)^2 p(d_i)} = [(d_i - r_{i-1})^2 - (d_i - r_i)^2]p(d_i)$
- Setting E_{d_i}=0, we conclude that $(d_i - r_{i-1})^2 - (d_i - r_i)^2 = 0$, that is,
- $(d_i - r_{i-1})^2 = (d_i - r_i)^2$, which implies $d_i - r_{i-1} = r_i - d_i$, and thus
- $\boldsymbol{d_i = \frac{r_{i-1}+r_i}{2}}$

If $f(t) = \int_a^t g(x)dx$, then $f'(t) = g(t)$

If $f(t) = \int_t^a g(x)dx$, then $f'(t) = -g(t)$

OPTIMAL NON-UNIFORM QUANTIZERS
-- MAX-LLOYD QUANTIZERS (6/8) --

- To summarize so far, we have
- **r_i = the average value of the data that fall in interval $[d_i \ d_{i+1}), \forall i = 0, 1, \dots n-1$, and**
- $d_i = \frac{r_{i-1}+r_i}{2} \ \forall i = 1, 2, \dots, n-1$
- The above equations constitute $2n-1$ equations in $2n-1$ unknowns, where the unknowns are $d_1, d_2, \dots, d_{n-1}$ and $r_0, r_1, \dots, r_{n-1}$
- Unfortunately, these equations are non-linear, and thus hard to solve directly
- But, there is an algorithm for solving those equations:
- The Max-Lloyd technique is an iterative algorithm for deriving very accurate approximations of $d_1, d_2, \dots, d_{n-1}$ and $r_0, r_1, \dots, r_{n-1}$

OPTIMAL NON-UNIFORM QUANTIZERS
-- MAX-LLOYD QUANTIZERS (7/8) --

The Max-Lloyd Algorithm:

1. Initialize $d_1, d_2, \ldots, d_{n-1}$ and $r_0, r_1, \ldots, r_{n-1}$ to some random values; for example, initialize them as if the quantizer were a uniform quantizer (just for a start)
2. **Repeat**
   - For all i, compute r_i = the average value of the data that happen to fall in interval $[\boldsymbol{d_i}\ \boldsymbol{d_{i+1}})$
   - For all i, compute $\boldsymbol{d_i} = \frac{\boldsymbol{r_{i-1}+r_i}}{\boldsymbol{2}}$

   Until (the error between successive estimates of the $\boldsymbol{d_i}$'s is < a set tolerance)
3. **Return** the final values of $d_1, d_2, \ldots, d_{n-1}$ and $r_0, r_1, \ldots, r_{n-1}$

OPTIMAL NON-UNIFORM QUANTIZERS
-- MAX-LLOYD QUANTIZERS (8/8) --

- Insight into the Max-Lloyd algorithm
  - It is a "*ping-pong game*" between the d_i's and the r_i's
  - After updating the d_i's, the "ball" is tossed to the r_i's to update themselves using the latest values of the d_i's, and following the formulas that for all i:

    r_i = the average value of the data that happen to fall in interval $[\boldsymbol{d_i}\ \ \boldsymbol{d_{i+1}})$
  - After updating the r_i's, the "ball" is tossed to the d_i's to update themselves using the latest values of the r_i's, and following the formulas that for all i:

$$\boldsymbol{d_i = \frac{r_{i-1}+r_i}{2}}$$

  - The back-and-forth game continues, until the d_i's stop changing much
  - At that point, the algorithm stops and reports the final values of d_i's and r_i's
- **Theorem**: The algorithm "converges" (i.e., reaches final values), and when it stops, the final values are globally optimal. // the proof is beyond the scope

EXAMPLE OF OBTAINING A MAX-LLOYD QUANTIZER (1/5)

- Take the same data of a previous example,

X=[0 0.01 2.8 3.4 1.99 3.6 5 3.2 4.5 7.1 7.9]

- We want to get the optimal 4-level quantizer, using the iterative Max-Lloyd algorithm
- So, we need to find d_1, d_2, d_3 and r_0, r_1, r_2, r_3. Recall that $d_0 = 0$, and $d_4 = 8$
- Initialize the d's to uniform values: $d_1 = 2, d_2 = 4, d_3 = 6$
- In the first iteration, we need to update the r's (and then the d's)
  - r_0 = the average value of the data that happen to fall in interval $[\boldsymbol{d_0}\ \boldsymbol{d_1}) = [\mathbf{0}\ \mathbf{2})$
  - To do that, find the data in X that fall in $[\mathbf{0}\ \mathbf{2})$, and take their average
  - Those data are: 0, 0.01, 1.99. Their average is (0+0.01+1.99)/3=2/3
  - Thus r_0 =2/3
  - Compute r_1, r_2, r_3 the same way, we get $r_1 = 3.25, r_2 = 4.75, r_3 = 7.5$

EXAMPLE OF OBTAINING A MAX-LLOYD QUANTIZER (2/5)

- X=[0 0.01 2.8 3.4 1.99 3.6 5 3.2 4.5 7.1 7.9]
- 1st iteration
  - r_0 =2/3 , $r_1 = 3.25, r_2 = 4.75, r_3 = 7.5$
  - Now compute the d's using $\boldsymbol{d_i = \frac{r_{i-1}+r_i}{2}}$:

$$d_1 = \frac{r_0+r_1}{2} = \frac{\frac{2}{3}+3.25}{2} = 1.958, \quad d_2 = \frac{r_1+r_2}{2} = \frac{3.25+4.75}{2} = 4, \quad d_3 = \frac{r_2+r_3}{2} = \frac{4.75+7.5}{2} = 6.125$$

- 2nd iteration
  - Use the d's to update the r's
  - $r_0 = average\ of\ the\ data\ in\ [0\ 1.958) = \frac{0+0.01}{2} = 0.005$
  - $r_1 = average\ of\ the\ data\ in\ [1.958\ 4) = \frac{2.8+3.4+1.99+3.6+3.2}{5} = 2.998$
  - $r_2 = average\ of\ the\ data\ in\ [4\ 6.125) = \frac{5+4.5}{2} = 4.75$
  - $r_3 = average\ of\ the\ data\ in\ [6.125\ 8) = \frac{7.1\ 7.9}{2} = 7.5$

EXAMPLE OF OBTAINING A MAX-LLOYD QUANTIZER (3/5)

- X=[0 0.01 2.8 3.4 1.99 3.6 5 3.2 4.5 7.1 7.9]
- 2nd iteration
  - r_0 =0.005 , $r_1 = 2.998, r_2 = 4.75, r_3 = 7.5$
  - Now compute the d's using $\boldsymbol{d_i = \frac{r_{i-1}+r_i}{2}}$:

$$d_1 = \frac{r_0+r_1}{2} = \frac{0.005+2.998}{2} = 1.5015,\ d_2 = \frac{r_1+r_2}{2} = \frac{2.998+4.75}{2} = 3.874,\ d_3 = \frac{r_2+r_3}{2} = \frac{4.75+7.5}{2} = 6.125$$

- 3rd iteration
  - Use the d's to update the r's
  - $r_0 = average\ of\ the\ data\ in\ [0\ \ 1.5015) = \frac{0+0.01}{2} = 0.005$
  - $r_1 = average\ of\ the\ data\ in\ [1.5015\ \ 3.874) = \frac{2.8+3.4+1.99+3.6+3.2}{5} = 2.998$
  - $r_2 = average\ of\ the\ data\ in\ [3.874\ \ 6.125) = \frac{5+4.5}{2} = 4.75$
  - $r_3 = average\ of\ the\ data\ in\ [6.125\ \ 8) = \frac{7.1\ 7.9}{2} = 7.5$

EXAMPLE OF OBTAINING A MAX-LLOYD QUANTIZER (4/5)

- X=[0 0.01 2.8 3.4 1.99 3.6 5 3.2 4.5 7.1 7.9]
- 3rd iteration
  - r_0 =0.005 , $r_1 = 2.998, r_2 = 4.75, r_3 = 7.5$
  - Now compute the d's using $\boldsymbol{d_i = \frac{r_{i-1}+r_i}{2}}$:

$$d_1 = \frac{r_0+r_1}{2} = \frac{0.005+2.998}{2} = 1.5015,\ d_2 = \frac{r_1+r_2}{2} = \frac{2.998+4.75}{2} = 3.874,\ d_3 = \frac{r_2+r_3}{2} = \frac{4.75+7.5}{2} = 6.125$$

- Now observe that the r's and d's in the 3rd iteration did not change from their values in the 2nd iteration
- Therefore, the Max-Lloyd algorithm has **converged** to the following values:

r_0 =0.005 , $r_1 = 2.998, r_2 = 4.75, r_3 = 7.5$

$d_0 = 0,\ d_1 = 1.5015,\ d_2 = 3.874,\ d_3 = 6.125,\ d_4 = 8$

EXAMPLE OF OBTAINING A MAX-LLOYD QUANTIZER (5/5)

- X=[0 0.01 2.8 3.4 1.99 3.6 5 3.2 4.5 7.1 7.9]
- We summarize all the iterations and calculations in the following table

Initial Values	**d**	**0**	**2**	**4**	**6**	**8**
Iteration 1	**r**	2/3	3.25	4.75	7.5	
	d	0	1.958	4	6.125	8
Iteration 2	**r**	0.005	2.998	4.75	7.5	
	d	0	1.5015	3.874	6.125	8
Iteration 3	**r**	0.005	2.998	4.75	7.5	
	d	0	1.5015	3.874	6.125	8

ALGORITHM FOR CONSTRUCTING OPTIMAL SEMI-UNIFORM QUANTIZERS

- Recall that in a semi-uniform quantizers, all the intervals are of equal length
- So the d's are deterministic: $\Delta = \frac{d_n - d_0}{n}$ and $d_i = d_0 + i\Delta \; \forall i$
- The r's are the optimal centroids of their intervals
- We can show, in a way similar to the way we derived the Max-Lloyd algorithm, that each r_i is

r_i=the average of the data that falls in interval $[d_i \;\; d_{i+1})$

SUMMARY

- We saw how scalar quantizers fit in the 3-stage framework of lossy compression
- We studied three types of scalar quantizers
  - Uniform
  - Semi-uniform
  - Optimal (Max-Lloyd) non-uniform
- We learned how to compute the decision levels and reconstruction values in each type of scalar quantizer
- In the next several lectures, we'll turn our attention to the subject of transforms
  - It is very important to lossy compression
  - It has broader applications than compression, such as filtering, noise cancellation, feature extraction in machine learning, etc.

NEXT LECTURE

- Transforms
  - General Definition
  - Reason for transforms
  - Requirements for transforms
- End-user (computational) perspective of transforms
  - Simple mathematical/computational definitions of several popular transforms