Updated 11/8/2025

Part 3: Applications
L17: Channel Capacity, Distortion Theory, Information Bottleneck
[Channel Capacity]

Wolfgang Gatterbauer

cs7840 Foundations and Applications of Information Theory (fa25)

https://northeastern-datalab.github.io/cs7840/fa25/

11/6/2025

Pre-class conversations

- Last class recapitulation
- Projects: I can talk today after class

- Last time:
  - Why Max entropy? Involves just combinatorics, and limits, no "uncertainty"
  - Why Occam's razor? Again: a simple (convincing ?) argument
  - Kolmogorov complexity: the answer to all questions ?
- Today:
  - Channel capacity (communication)
  - Distortion theory

Channel Capacity

Largely based on chapter 7 of
[Cover, Thomas'06] Elements of Information Theory, 2006. https://www.doi.org/10.1002/047174882X

Shannon [1948]: Communicating over a noisy channel

The Bell System Technical Journal

Vol. XXVII *July, 1948* *No. 3*

A Mathematical Theory of Communication

By C. E. SHANNON

INTRODUCTION

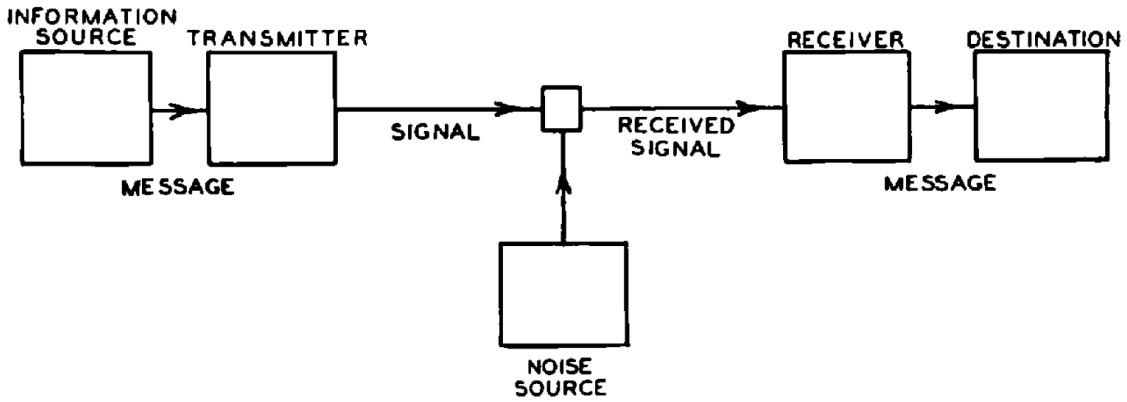


Fig. 1—Schematic diagram of a general communication system.

The entropy in the case of two possibilities with probabilities p and $q = 1 - p$, namely

$$H = -(p \log p + q \log q)$$

is plotted in Fig. 7 as a function of p.

The quantity H has a number of interesting properties which further substantiate it as a reasonable measure of choice or information.

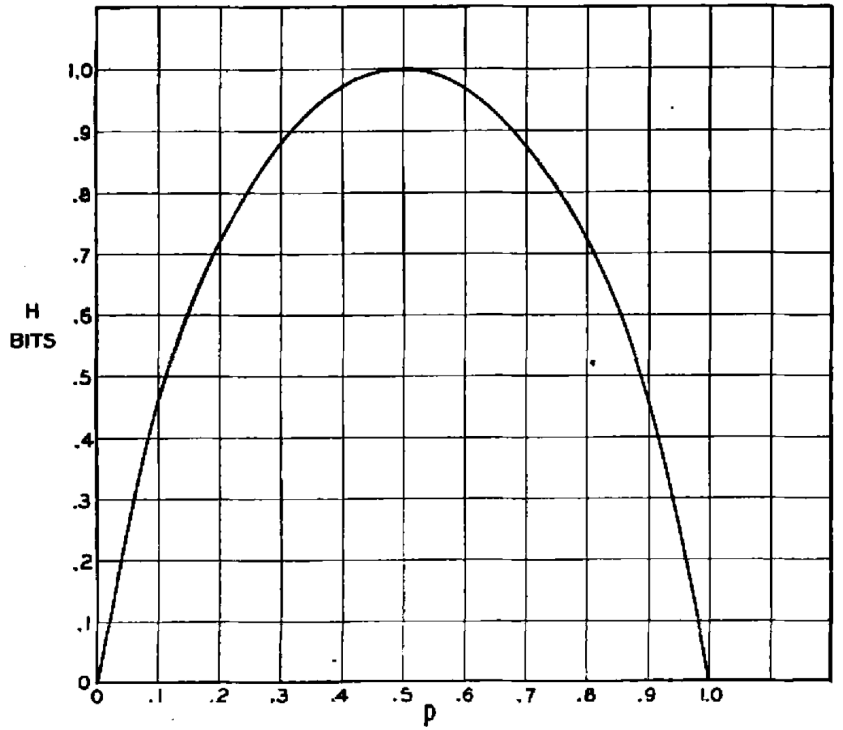
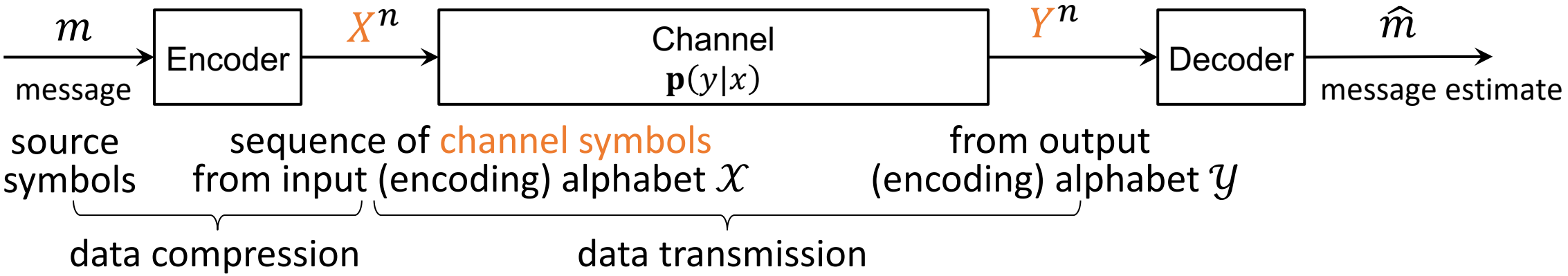


Fig. 7—Entropy in the case of two possibilities with probabilities p and $(1 - p)$.

From: Shannon. A Mathematical Theory of Communication, The Bell System Technical Journal, 1948. https://doi.org/10.1002/j.1538-7305.1948.tb01338.x
Gatterbauer. Foundations and Applications of Information Theory: https://northeastern-datalab.github.io/cs7840/fa25/

Channel Capacity C = highest rate R



intuition: capacity = lg[# of distinguishable inputs]

(1) "Information" channel capacity $\boxed{C = \max_{\mathbf{p}(x)} I(X;Y)}$

(2) "Operational" channel capacity: the highest communication rate R (in bits) per channel use

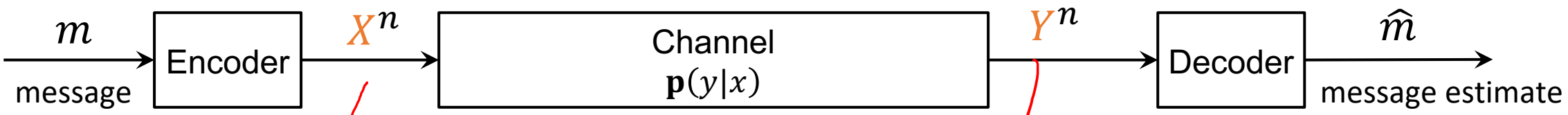
SHANNON'S CHANNEL CODING THEOREM: both are identical, i.e. the channel capacity can be achieved in the limit by using codes with a long block length.

Theorem 7.7.1 (*Channel coding theorem*) *For a discrete memoryless channel, all rates below capacity C are achievable. Specifically, for every rate $R < C$, there exists a sequence of $(2^{nR}, n)$ codes with maximum probability of error $\lambda^{(n)} \to 0$.*

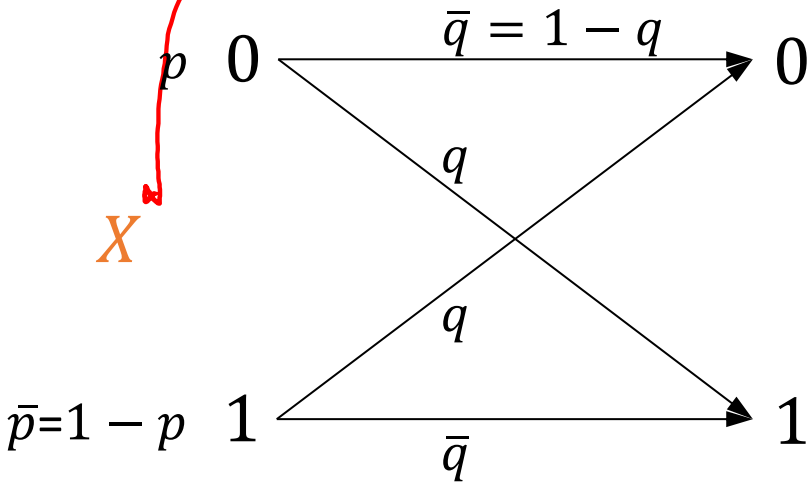
Conversely, any sequence of $(2^{nR}, n)$ codes with $\lambda^{(n)} \to 0$ must have $R \leq C$.

Theorem 7.7.1 taken from: [Cover, Thomas'06] Elements of Information Theory, 2006. Chapter 7 channel capacity, https://www.doi.org/10.1002/047174882X
Gatterbauer. Foundations and Applications of Information Theory: https://northeastern-datalab.github.io/cs7840/fa25/

Capacity of binary symmetric channel



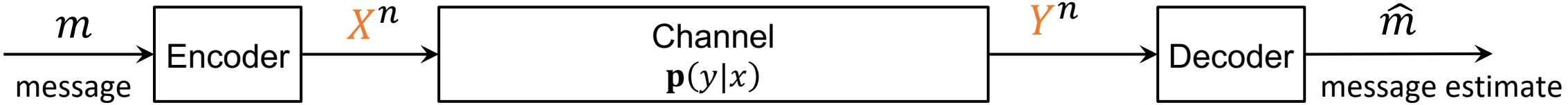
channel capacity $C = \max_{\mathbf{p}(x)} I(X;Y)$



X Y

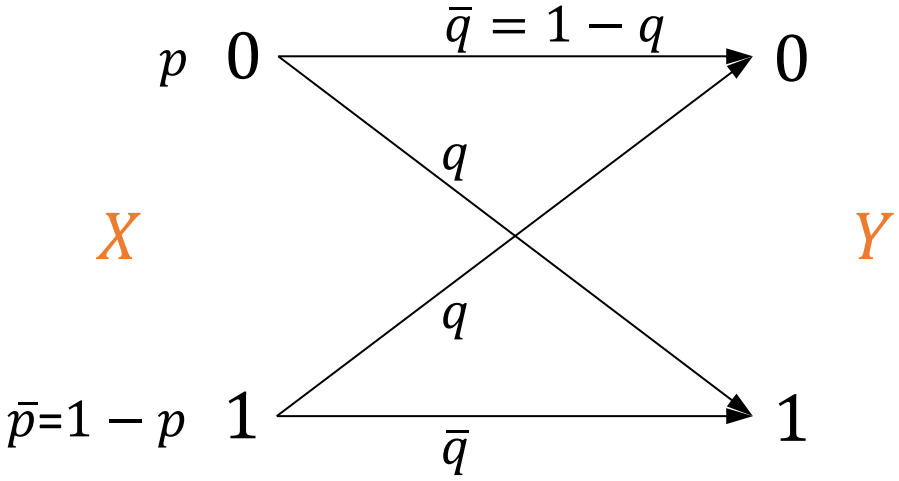
How do we calculate the channel capacity C ?

Capacity of binary symmetric channel

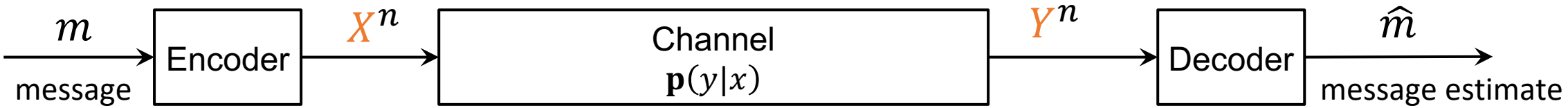


channel capacity $C = \max_{\mathbf{p}(x)} I(X;Y)$

$I(X;Y) =$?



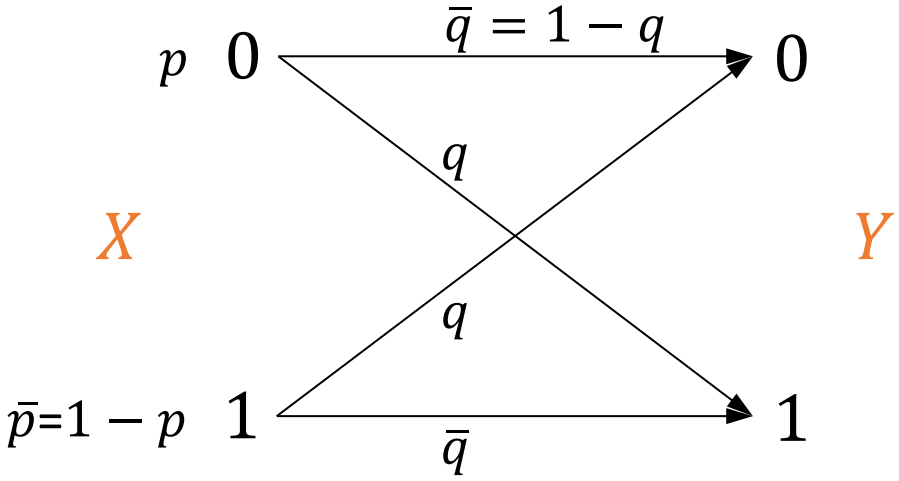
Capacity of binary symmetric channel



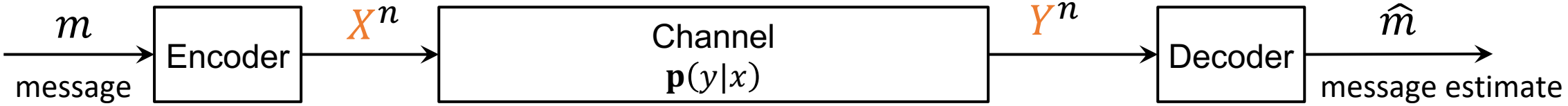
channel capacity $C = \max_{\mathbf{p}(x)} I(X;Y)$

$$I(X;Y) = H(Y) - H(Y|X)$$

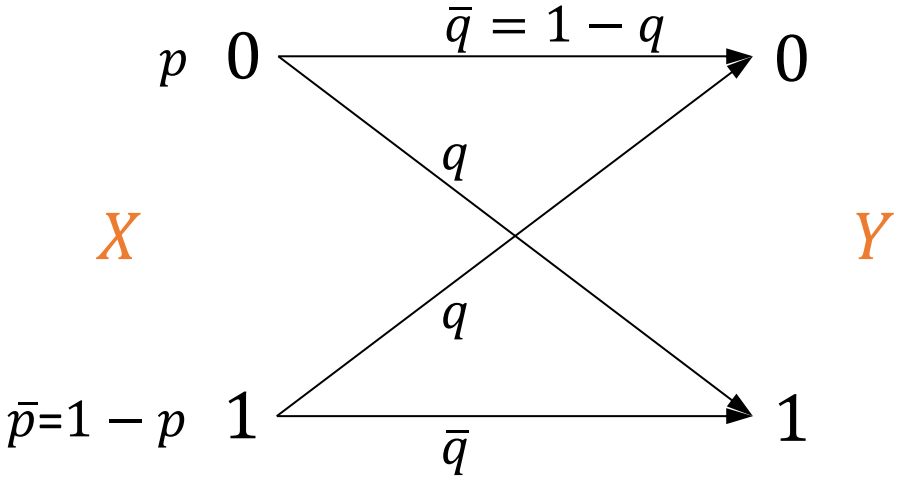
?



Capacity of binary symmetric channel



channel capacity $C = \max_{\mathbf{p}(x)} I(X;Y)$



$$I(X;Y) = H(Y) - H(Y|X)$$

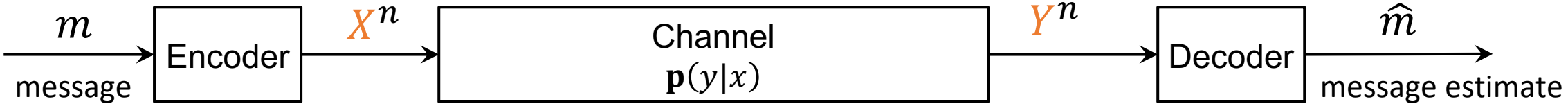
$\sum_x p(x)\, H(Y|X = x)$

$\sum_x p(x)\, H(q) = H(q)$

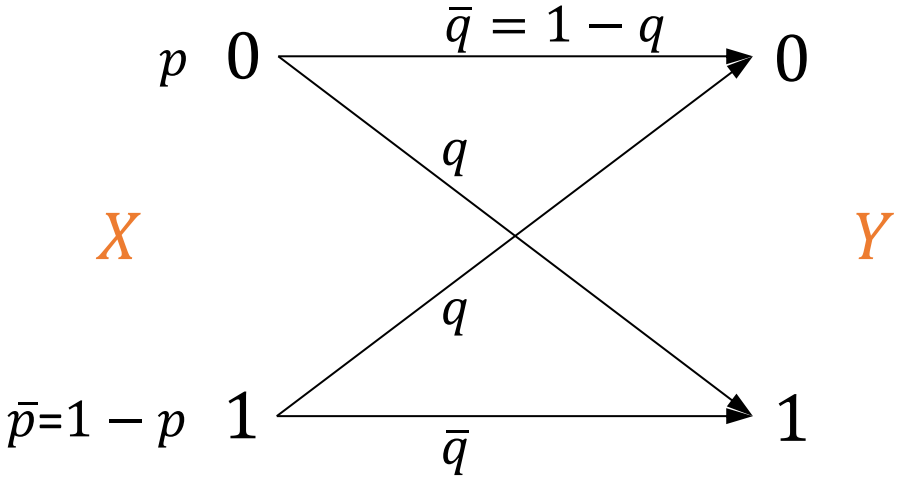
$$= H(Y) - H(q)$$

?

Capacity of binary symmetric channel



channel capacity $C = \max_{\mathbf{p}(x)} I(X;Y)$



$$I(X;Y) = H(Y) - H(Y|X)$$

$\sum_x p(x)\, H(Y|X=x)$

$\sum_x p(x)\, H(q) = H(q)$

$$= H(Y) - H(q)$$

Max of $H(Y) = 1$ (thus also max of $I(X;Y)$)

This is achieved for uniform $p(Y{=}0) = 0.5$.

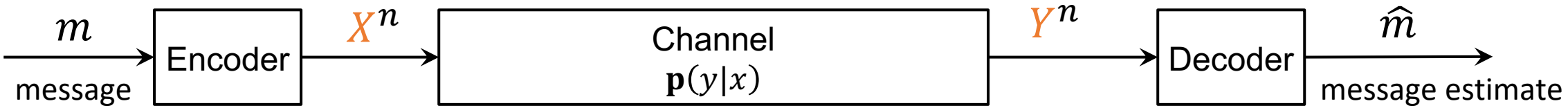
Thus also $p = 0.5$ has uniform input distribution

(solve for $p\bar{q} + \bar{p}q = p(Y{=}0) = 0.5$)

$$\leq 1 - H(q)$$

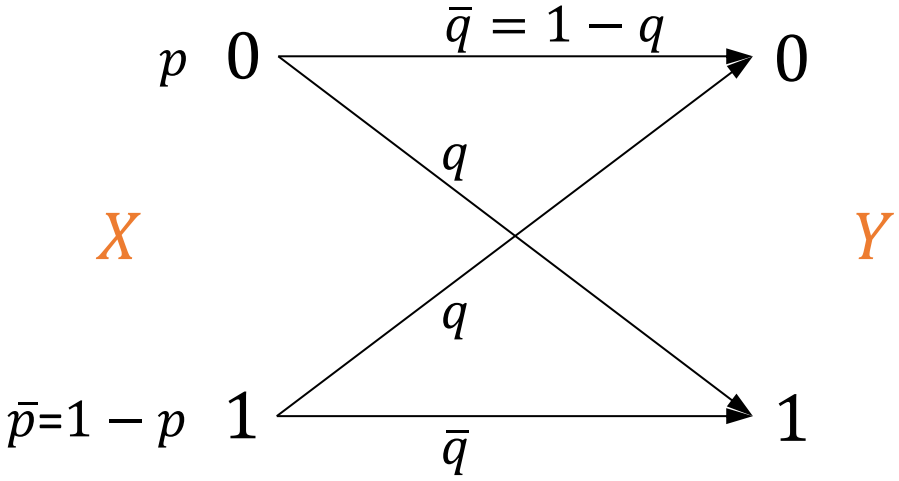
Hence, capacity for binary symmetric channel is $\boxed{C = 1 - H(q)}$

Capacity of binary symmetric channel



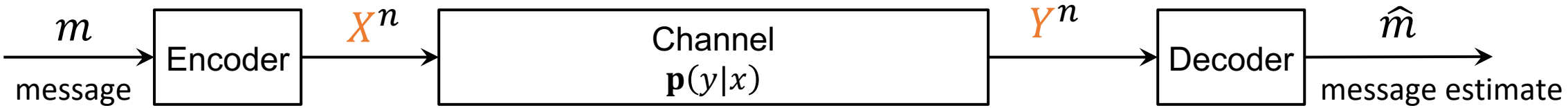
channel capacity $C = \max_{\mathbf{p}(x)} I(X;Y)$

$$I(X;Y) = H(X) - H(X|Y)$$



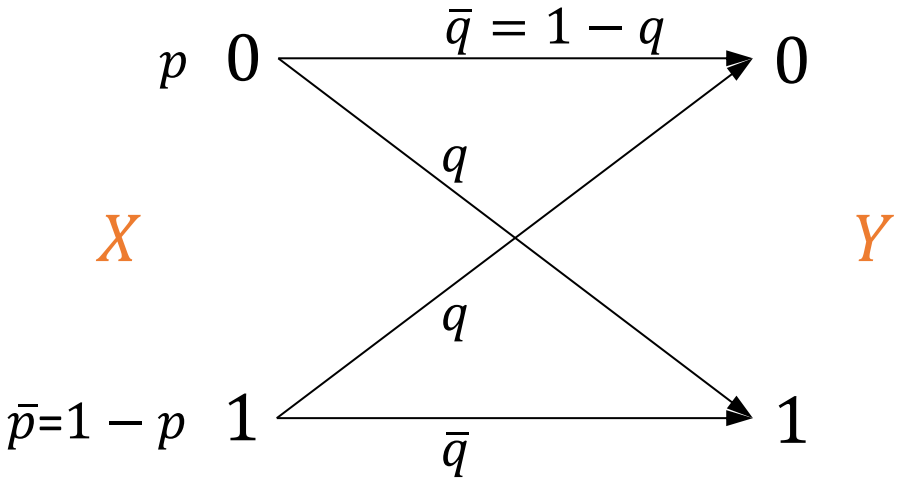
What about the other way around ?

Capacity of binary symmetric channel



channel capacity $C = \max_{\mathbf{p}(x)} I(X;Y)$

Trying to do it the other way around should work but becomes far more complicated ☹



$$I(X;Y) = H(X) - H(X|Y)$$

$H(p)$

$$H(X|Y{=}0) \cdot p(Y{=}0) + H(X|Y{=}1) \cdot p(Y{=}1)$$

$p\bar{q} + \bar{p}q$ $\qquad$ $pq + \bar{p}\bar{q}$

$$p(X{=}0|Y{=}0) = \frac{p(Y{=}0|X{=}0) \cdot p(X{=}0)}{p(Y{=}0)} = \frac{p\bar{q}}{p\bar{q}+\bar{p}q}$$

$$p(X{=}0|Y{=}1) = \frac{p(Y{=}1|X{=}0) \cdot p(X{=}0)}{p(Y{=}1)} = \frac{pq}{pq+\bar{p}\bar{q}}$$

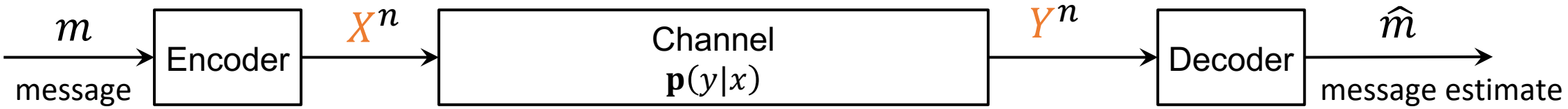
$$\max_{\mathbf{p}(x)} I(X;Y) = H(p) - H\left(\frac{p\bar{q}}{p\bar{q}+\bar{p}q}\right)(p\bar{q} + \bar{p}q) - H\left(\frac{pq}{pq+\bar{p}\bar{q}}\right)(pq + \bar{p}\bar{q})$$

while q is fixed

... gets messy

Let's check for $p=\frac{1}{2}$

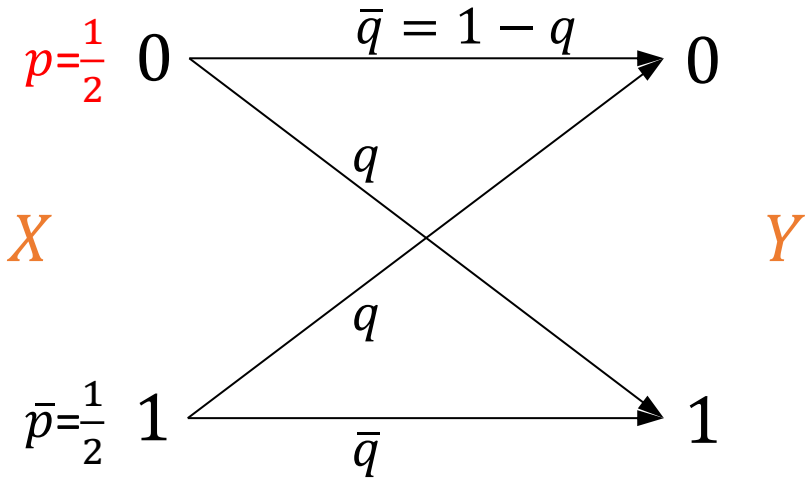
Capacity of binary symmetric channel



channel capacity $C = \max_{\mathbf{p}(x)} I(X;Y)$

Trying to do it the other way around should work but becomes far more complicated ☹

$$I(X;Y) = H(X) - H(X|Y)$$



$p=\frac{1}{2}$

$$= H\left(\frac{1}{2}\right) - H(\bar{q})\left(\frac{1}{2}\right) - H(\bar{q})\left(\frac{1}{2}\right)$$
$$= 1 - H(q) \checkmark$$

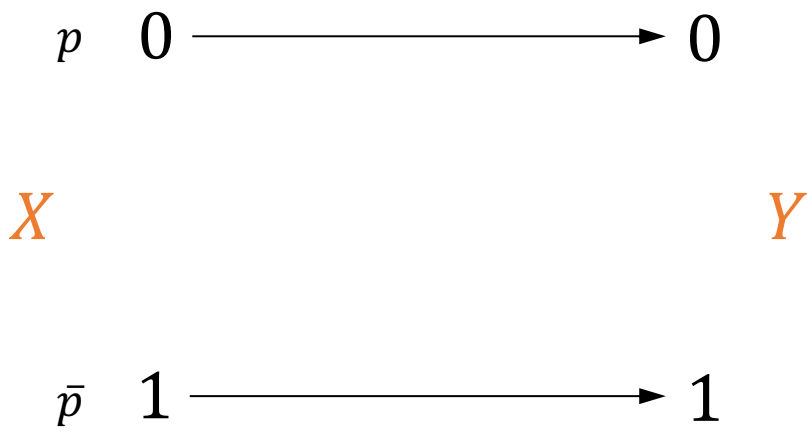
$$I(X;Y) = H(p) - H\left(\frac{p\bar{q}}{p\bar{q}+\bar{p}q}\right)(p\bar{q} + \bar{p}q)$$
$$-H\left(\frac{pq}{pq+\bar{p}\bar{q}}\right)(pq + \bar{p}\bar{q})$$

... gets messy

Channel Capacity: Binary Channel

EXAMPLE

Binary noiseless channel



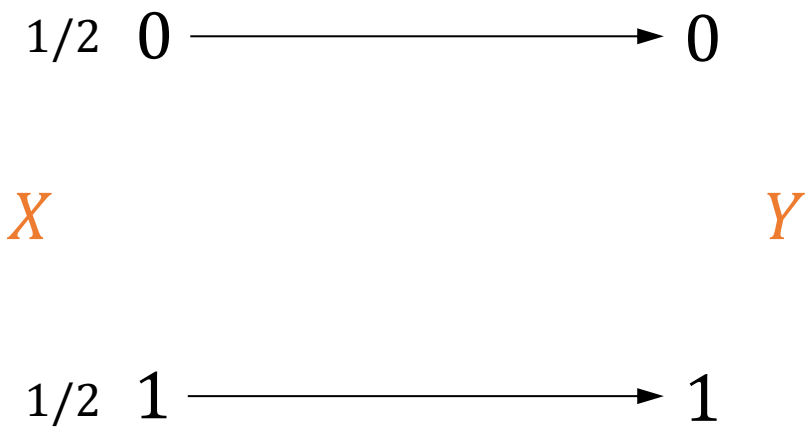
channel capacity:

$C =$?

Channel Capacity: Binary Channel

EXAMPLE

Binary noiseless channel



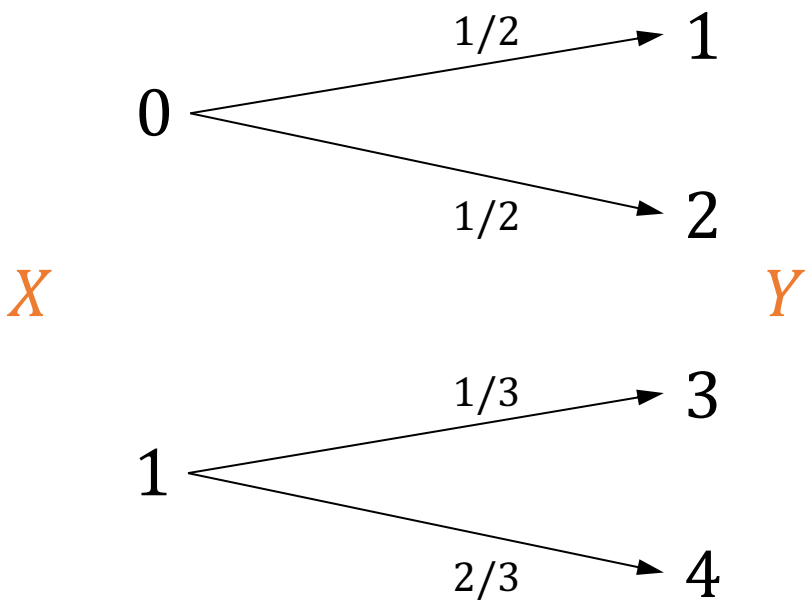
channel capacity:

$C = \lg(2) = 1$ bit

Channel Capacity: Non-overlapping Outputs

EXAMPLE

Noisy channel with non-overlapping outputs



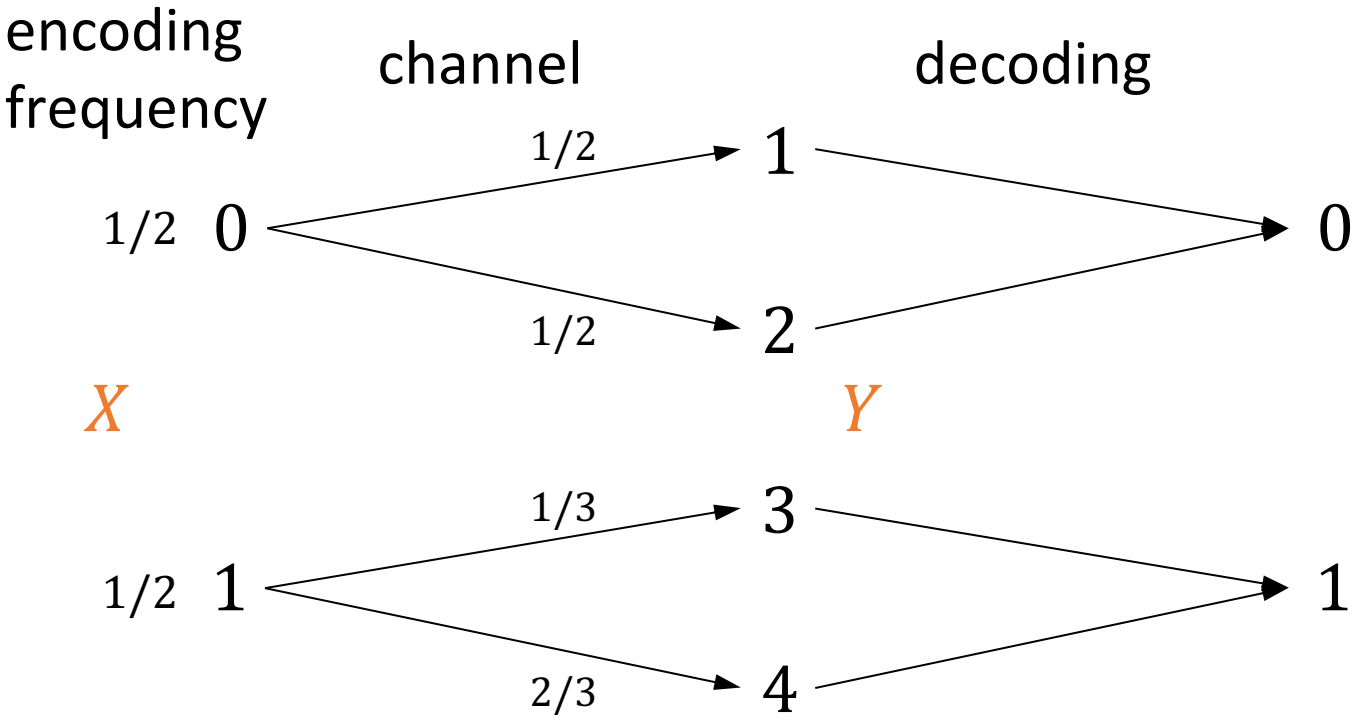
channel capacity:

$C =$?

Channel Capacity: Non-overlapping Outputs

EXAMPLE

Noisy channel with non-overlapping outputs



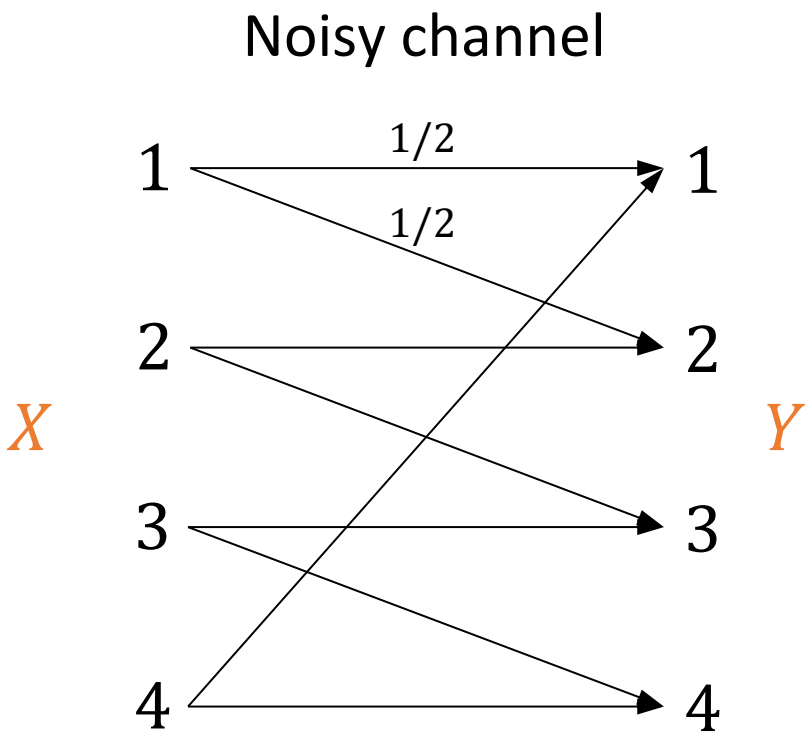
channel capacity:

$C = \lg(2) = 1$ bit

Channel Capacity: Noisy typewriter

EXAMPLE

Noisy typewriter



channel capacity:

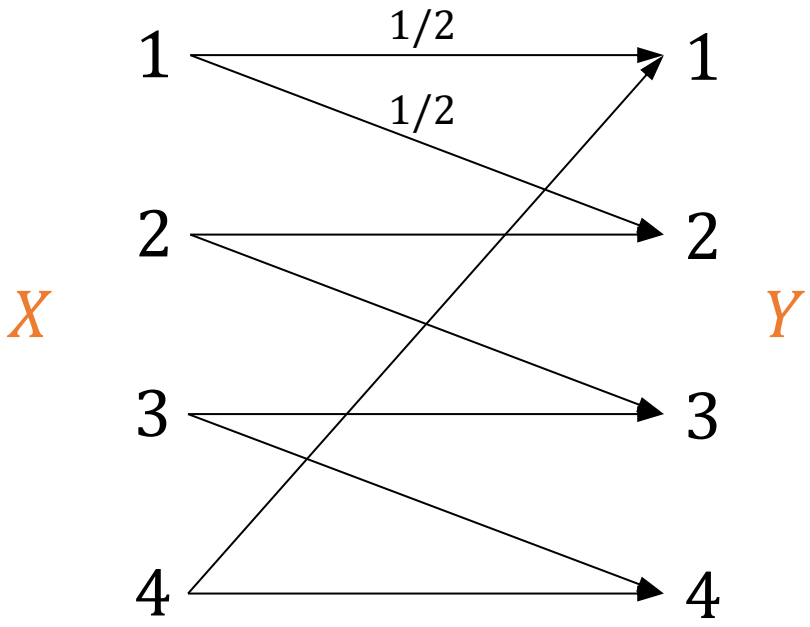
$C =$?

Channel Capacity: Noisy typewriter

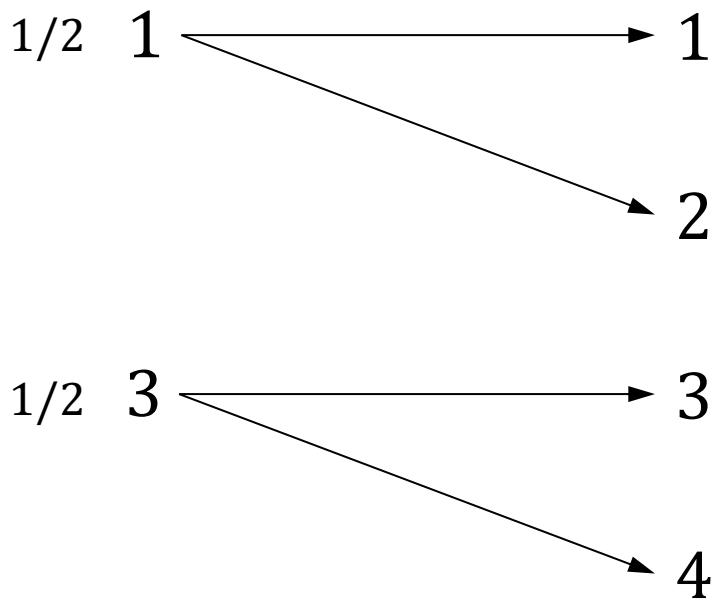
EXAMPLE

Noisy typewriter

Noisy channel



Noiseless subset of inputs



channel capacity:

$C = \lg(2) = 1$ bit

Channel Capacity: Binary erasure channel

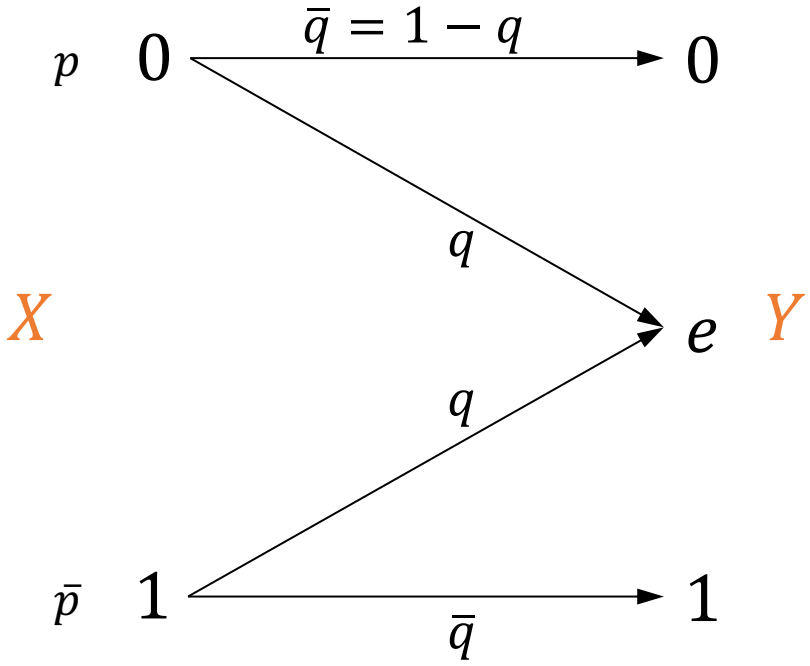
EXAMPLE

Binary erasure channel

A fraction of q bits are erased (rather than corrupted as in the binary symmetric channel). The receiver knows which bits are lost.

$$C = \max_{\mathbf{p}(x)}[I(X;Y)]$$

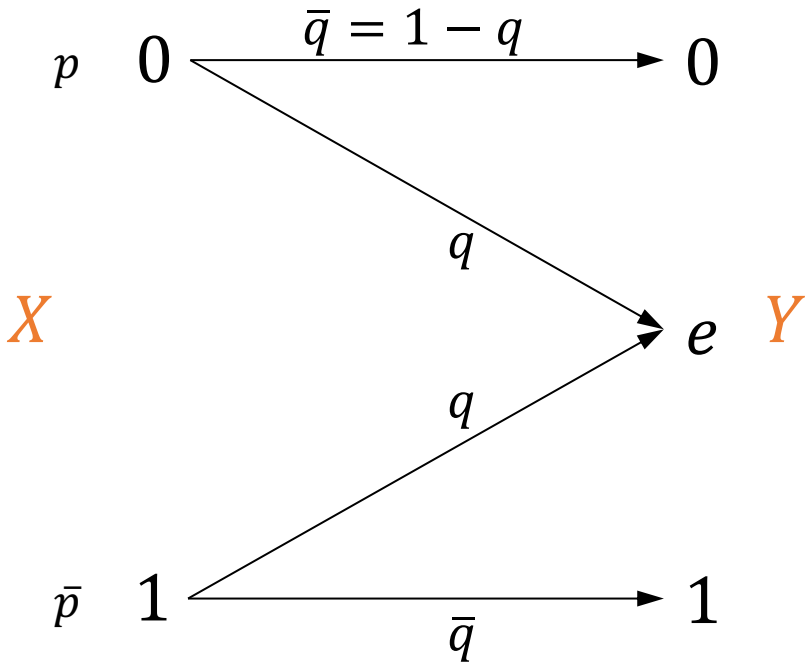
?



Channel Capacity: Binary erasure channel

EXAMPLE

Binary erasure channel



A fraction of q bits are erased (rather than corrupted as in the binary symmetric channel).

The receiver knows which bits are lost.

$$C = \max_{\mathbf{p}(x)}[I(X;Y)]$$

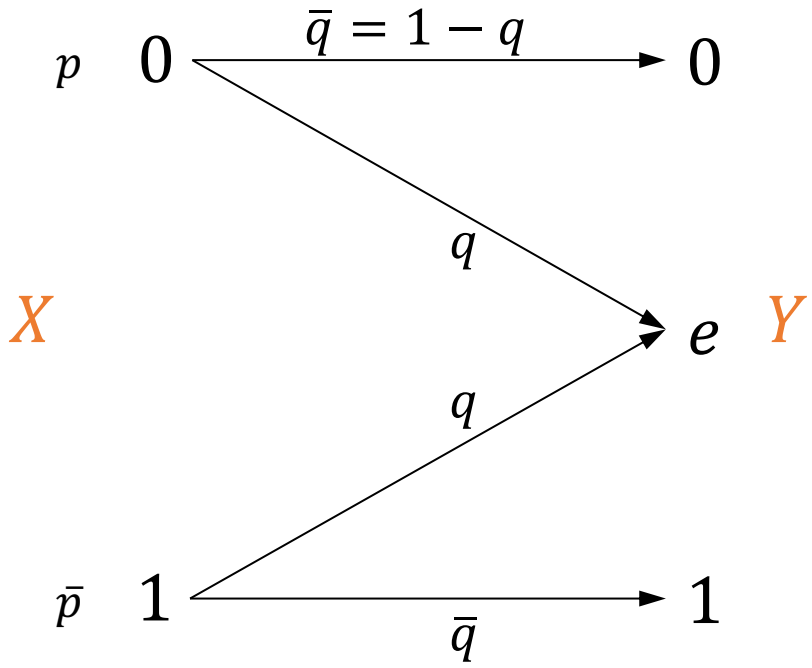
$$= \max_{\mathbf{p}(x)}[H(Y) - H(Y|X)]$$

?

Channel Capacity: Binary erasure channel

EXAMPLE

Binary erasure channel



A fraction of q bits are erased (rather than corrupted as in the binary symmetric channel).

The receiver knows which bits are lost.

$$C = \max_{\mathbf{p}(x)}[I(X;Y)]$$

$$= \max_{\mathbf{p}(x)}[H(Y) - H(Y|X)]$$

$$= H(q)$$

$$= \max_{\mathbf{p}(x)}[H(Y)] - H(q)$$

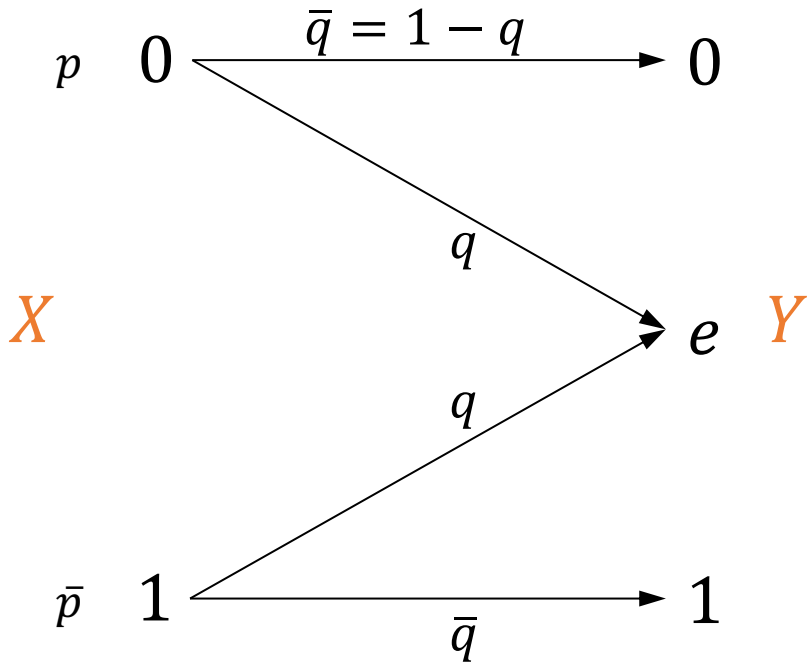
erasure happens yes/no

$$H(Y) = H(Y,E) = ?$$

Channel Capacity: Binary erasure channel

EXAMPLE

Binary erasure channel



A fraction of q bits are erased (rather than corrupted as in the binary symmetric channel).

The receiver knows which bits are lost.

$$C = \max_{\mathbf{p}(x)}[I(X;Y)]$$

$$= \max_{\mathbf{p}(x)}[H(Y) - H(Y|X)]$$

$H(Y|X) = H(q)$

$$= \max_{\mathbf{p}(x)}[H(Y)] - H(q)$$

erasure happens yes/no

$$H(Y) = H(Y,E) = H(E) + H(Y|E)$$

$H(q)$ $\quad$ $q \cdot 0 + (1-q)H(p)$

$$= \max_{\mathbf{p}(x)}[\cancel{H(q)} + (1-q)H(p) - \cancel{H(q)}]$$

$$= (1-q) \cdot \max_{\mathbf{p}(x)}[H(p)] = (1-q)$$

Channel Capacity: Binary erasure channel

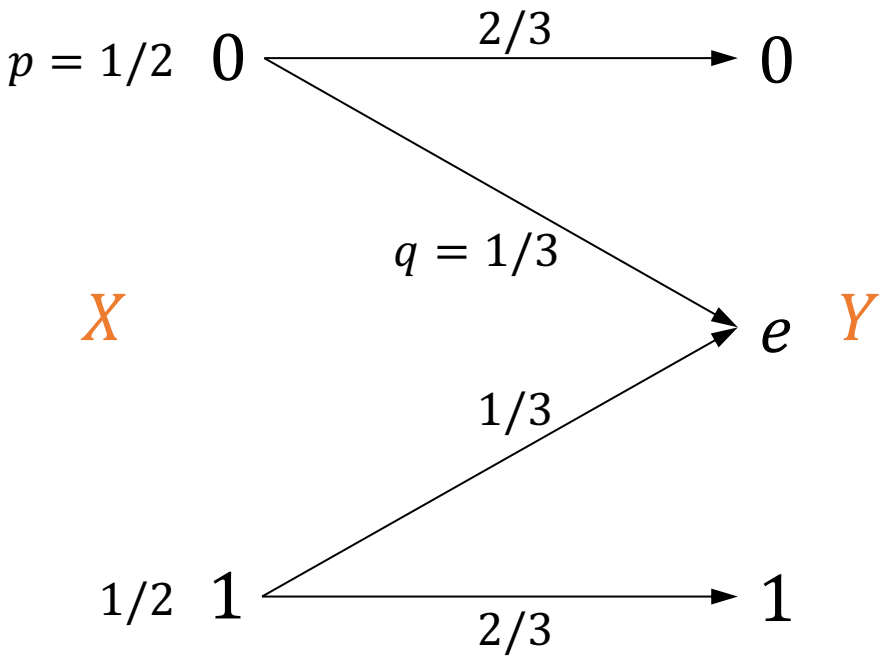
EXAMPLE

Binary erasure channel

A fraction of q bits are erased (rather than corrupted as in the binary symmetric channel). The receiver knows which bits are lost.

$$C = \max_{\mathbf{p}(x)}[I(X;Y)]$$

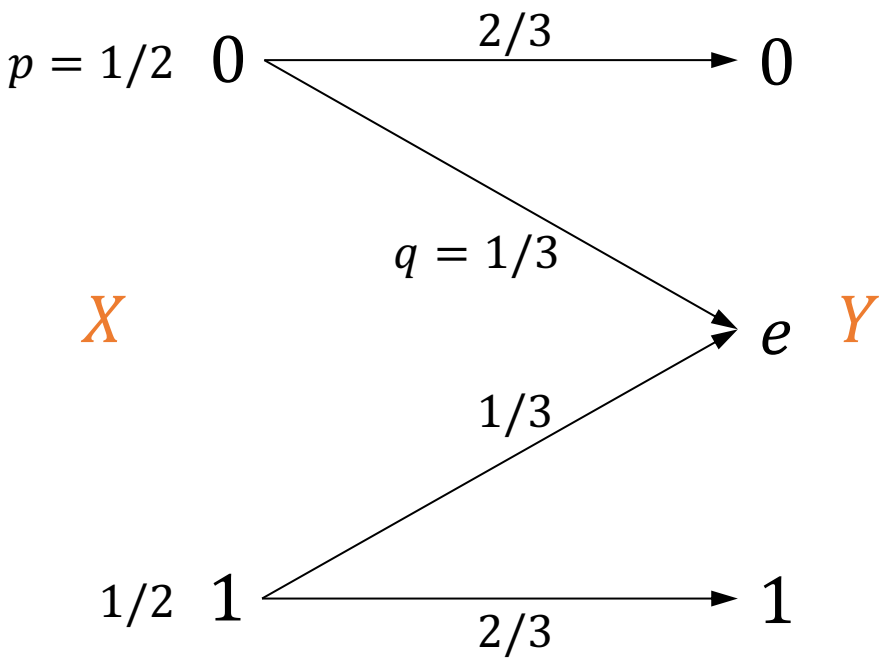
?



Channel Capacity: Binary erasure channel

EXAMPLE

Binary erasure channel



A fraction of q bits are erased (rather than corrupted as in the binary symmetric channel). The receiver knows which bits are lost.

$$C = \max_{\mathbf{p}(x)}[I(X;Y)] = I(X;Y) \text{ under the given } \mathbf{p}(x)$$

$$= H(Y) - H(Y|X)$$

$H(Y) = \lg(3)$

$$H(Y|X) = \frac{1}{2}H\left(\frac{1}{3}\right) + \frac{1}{2}H\left(\frac{1}{3}\right) = H\left(\frac{1}{3}\right)$$

$$= \frac{1}{3}\lg\left(\frac{1}{3}\right) + \frac{2}{3}\lg\left(\frac{2}{3}\right)$$

$$= -\frac{1}{3}\lg(3) + \frac{2}{3}\lg(2) - \frac{2}{3}\lg(3)$$

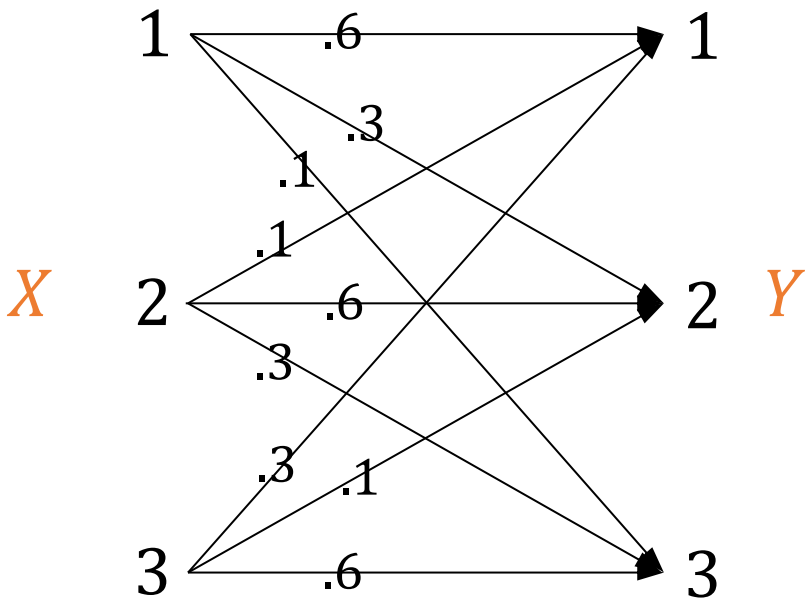
$$= \frac{2}{3} - \lg(3)$$

$$= \frac{2}{3} \quad = 1 - q \text{ (as predicated)}$$

Channel Capacity: Symmetric Channel

EXAMPLE

Symmetric channel



Symmetric channel: The state transition matrix $\mathbf{P}$ is symmetric if all the rows are permutations of each other (and so are the columns)

$$\mathbf{P} = \begin{pmatrix} .6 & .3 & .1 \\ .1 & .6 & .3 \\ .3 & .1 & .6 \end{pmatrix}$$

A symmetric $\mathbf{P}$ guarantees that a uniform input distribution leads to a uniform output distribution (the uniform distribution is an eigenvector)

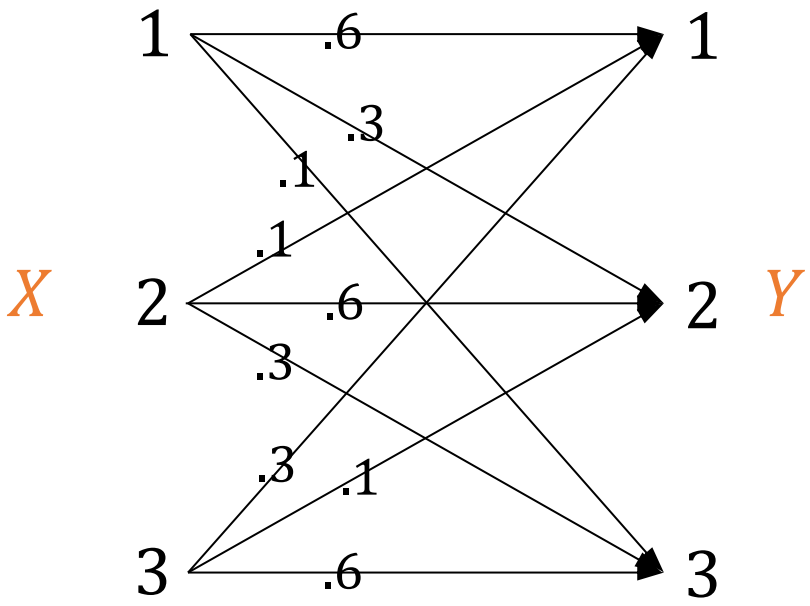
$$p(y) = \sum_x p(y|x)p(x) = \frac{1}{3}\sum_x p(y|x) = \frac{1}{3}$$

That guarantees that the capacity-achieving input distribution is uniform (which simplifies the math quite a bit and allows a closed form solution)

Channel Capacity: Symmetric Channel

EXAMPLE

Symmetric channel



Symmetric channel: The state transition matrix $\mathbf{P}$ (also channel matrix) is symmetric if all the rows are permutations of each other (and so are the columns)

$$\mathbf{P} = \begin{pmatrix} .6 & .3 & .1 \\ .1 & .6 & .3 \\ .3 & .1 & .6 \end{pmatrix} \quad \mathbf{p}(x) =: \mathbf{p} = \begin{pmatrix} 1/3 \\ 1/3 \\ 1/3 \end{pmatrix} \quad \mathbf{p}(y) =: \mathbf{q} = \begin{pmatrix} 1/3 \\ 1/3 \\ 1/3 \end{pmatrix}$$

$$C = \max_{\mathbf{p}}[I(X;Y)] = I(X;Y) \text{ under uniform}$$

$$= H(Y) - H(Y|X)$$

$$= \lg(3) - H(0.6,0.3,0.1)$$

$$\approx 1.585 - 1.295 \approx 0.290$$

Updated 11/13/2025

Part 3: Applications
L18: Channel Capacity, Distortion Theory, Information Bottleneck
[Channel Capacity: Blahut-Arimoto Algorithm, Distortion Theory]

Wolfgang Gatterbauer

cs7840 Foundations and Applications of Information Theory (fa25)

https://northeastern-datalab.github.io/cs7840/fa25/

11/13/2025

- **Lecture 15 (Thu, Oct 30)**
  Maximum Entropy
  (Deriving the Maximum Entropy Principle)
- **Lecture 16 (Mon, Nov 3)**
  MDL, Occam, Kolmogorov (2/2)
  (Occam, Kolmogorov, Minimum Description Length (MDL))
- **Lecture 17 (Thu, Nov 6)**
  Channel capacity [Cover Thomas'06: Ch 7], Distortion Theory (1/2) [Cover Thomas'06: Ch 10]
- **Lecture 18 (Mon, Nov 10)**
  Distortion Theory (2/2) [Cover Thomas'06: Ch 10]
  Python notebooks: 232
- **Lecture 19 (Thu, Nov 13) / P3 Intermediate report**
  30min remote guest lecture by Zsolt Zombori on [Zombori+'23] Towards Unbiased Exploration in Partial Label Learning, Information Bottleneck Theory

papers: information bottleneck

papers: Hinton on knowledge distillation

PART 4: The axiomatic approach (deriving formulations from first principles)

Covers the axiomatic approach from multiple angles: a few simple principles (axioms) leading to entropy or the laws of probability up to factors. Starting from a list of postulates leading to particular solution is a powerful approach that has been used across different areas of computer science (e.g. how to define the right scoring function for achieving a desired outcome)

- **Lecture 20 (Mon, Nov 17)**
  Derivation of Hartley measure and entropy function from first principles
- **Lecture 21 (Thu, Nov 20)**
  Cox's theorem: a derivation of the laws of probability theory from a certain set of postulates. Contrast with Kolmogorov's "probability axioms"
- **TBD**
  Shapley value
- **(Thu 11/27): no class (Fall break)**

more on compression

communication complexity

Shapley?

PART 5: Project presentations

- **Lecture 24 (Thu 12/4): P4 Project presentations / P5 Final report**
- **Lecture 25 (Mon 12/8): P4 Project presentations / P5 Final report**
- **Lecture 26 (Thu 12/11): P4 Project presentations / P5 Final report**

Blahut-Arimoto algorithm (for non-symmetric channels)

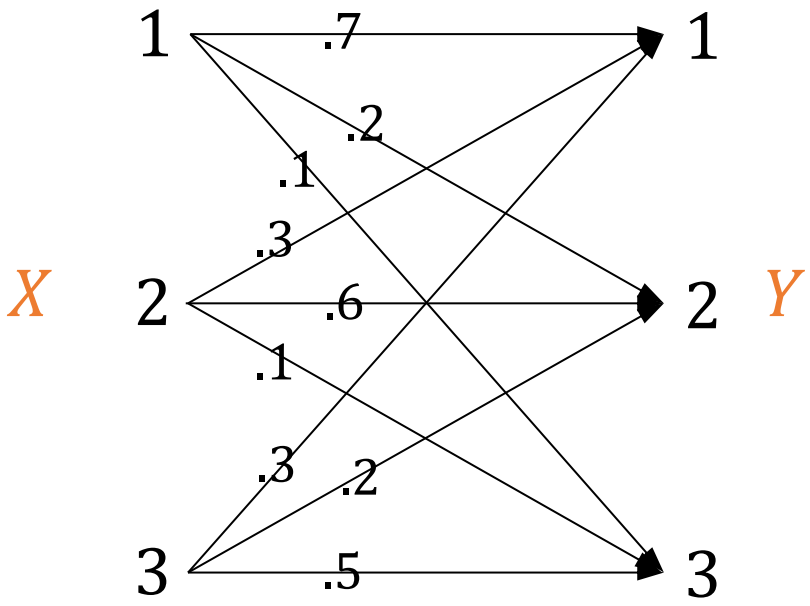
Published in two independent papers (Arimoto was first, Blahut is more general). Arimoto. *An algorithm for computing the capacity of arbitrary discrete memoryless channels*. TIT 1972. https://doi.org/10.1109/TIT.1972.1054753 , Blahut. *Computation of channel capacity and rate-distortion functions.* TIT 1972. https://doi.org/10.1109/TIT.1972.1054855

Channel Capacity: Non-Symmetric Channel

This is not a symmetric matrix anymore...

EXAMPLE

Non-symmetric discrete memoryless channel



row-stochastic

$$\mathbf{P} = \begin{pmatrix} .7 & .2 & .1 \\ .3 & .6 & .1 \\ .3 & .2 & .5 \end{pmatrix} \quad (x \downarrow,\ y \rightarrow)$$

$$\mathbf{p}(x) =: \mathbf{p} = \begin{pmatrix} ? \\ ? \\ ? \end{pmatrix} \qquad \mathbf{p}(y) =: \mathbf{q} = \begin{pmatrix} ? \\ ? \\ ? \end{pmatrix}$$

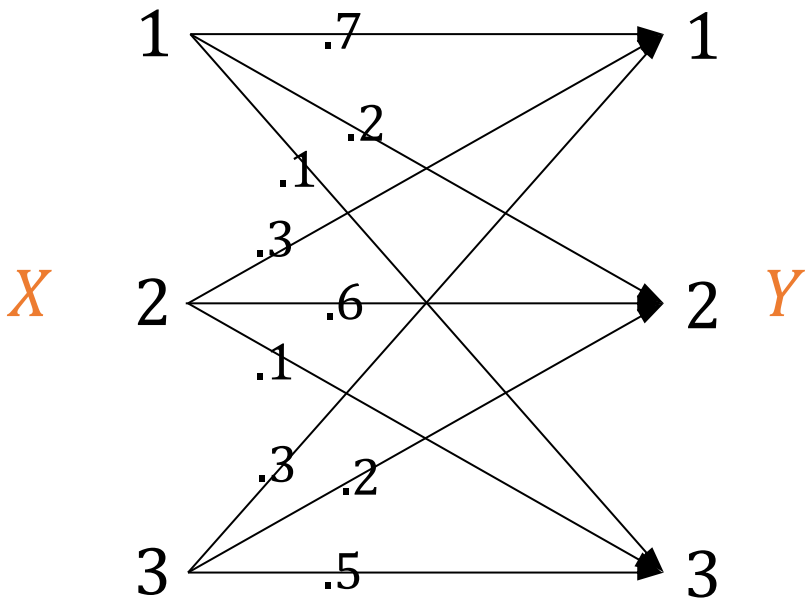
$$C = \max_{\mathbf{p}} [I(X;Y)]$$

what we do now ?

Channel Capacity: Non-Symmetric Channel

EXAMPLE

Non-symmetric discrete memoryless channel



This is a concave optimization problem in $\mathbf{p}$ over the simplex, can be solved with an iterative algorithm, such as the the BLAHUT–ARIMOTO algorithm

row-stochastic

$$\mathbf{P} = \begin{pmatrix} .7 & .2 & .1 \\ .3 & .6 & .1 \\ .3 & .2 & .5 \end{pmatrix} \quad \mathbf{p}(x) =: \mathbf{p} = \begin{pmatrix} 0.313 \\ 0.336 \\ 0.351 \end{pmatrix} \quad \mathbf{p}(y) =: \mathbf{q} = \begin{pmatrix} 0.425 \\ 0.334 \\ 0.241 \end{pmatrix}$$

x y

$$C = \max_{\mathbf{p}} [I(X;Y)]$$

$$= H(Y) - H(Y|X)$$

This is the solution. How do we get there?

?

$$\approx 1.554 - 1.313 \quad \approx 0.2289$$

Python activities file 231: https://github.com/northeastern-datalab/cs7840-activities/tree/main/notebooks/231_Blahut-Arimato.ipynb

Blahut-Arimoto algorithm (forward interpretation)

Start with an arbitrary input probability $\mathbf{p}$
and repeat until convergence:

- Calculate $\mathbf{q} \leftarrow \mathbf{P}^{\mathrm{T}}\mathbf{p}$
- For each input x, define the scalar $D_x(\mathbf{q})$ that measures how different (= informative) the output distribution $\mathbf{p}(:|x)$ is from the average output distribution $\mathbf{q}$. Collect these in vector $\mathbf{D}(\mathbf{q})$

  $C = \mathbb{E}_p[D_x]$
-

row $\mathbf{P}_{x:}$ $\qquad$ P_{xy}

$$D_x = D_{\mathrm{KL}}(\mathbf{p}(:|x)||\mathbf{q}) = \sum_y p(y|x) \cdot \lg\left(\frac{p(y|x)}{q(y)}\right)$$

D_x tells us how informative (or recognizable) x is:

- if sending x produces an output that looks similar to the usual channel output ($\mathbf{p}(:|x) \approx \mathbf{q}$), then seeing the output does not help you infer that x was sent.

  → if D_x is small, then x is not recognizable
- If sending x produces a distinctive output footprint, very unlike the average output $\mathbf{q}$, then the receiver can recognize that x was the one sent.

  → if D_x is large, then x is highly recognizable

Python activities file 231: https://github.com/northeastern-datalab/cs7840-activities/tree/main/notebooks/231_Blahut-Arimato.ipynb

Blahut-Arimoto algorithm (forward interpretation)

Start with an arbitrary input probability $\mathbf{p}$
and repeat until convergence:

- Calculate $\mathbf{q} \leftarrow \mathbf{P}^{\mathrm{T}}\mathbf{p}$
- For each input x, define the scalar $D_x(\mathbf{q})$ that measures how different (= informative) the output distribution $\mathbf{p}(:|x)$ is from the average output distribution $\mathbf{q}$. Collect these in vector $\mathbf{D}(\mathbf{q})$

row $\mathbf{P}_{x:}$ $\qquad$ P_{xy}

$$D_x = D_{\mathrm{KL}}(\mathbf{p}(:|x)||\mathbf{q}) = \sum_y p(y|x) \cdot \lg\left(\frac{p(y|x)}{q(y)}\right)$$

$$C = \mathbb{E}_p[D_x]$$

we multiply each weight p_x by an exponential factor, then renormalize.

- Push probability mass towards x with high D_x with multiplicative-weights update:

$$p_x \propto p_x 2^{D_x}$$

entrywise multiplication

$$\mathbf{p} \leftarrow \mathrm{softmax}_2(\mathbf{p} \odot \mathbf{D})$$

using 2 as base of the softmax instead of e

When converged: $D_1 = D_2 = \cdots = C$

Shifts the input distribution so that all used inputs end up having the same D_x value, i.e., they are equally distinguishable.

Optimality (convergence) is achieved when all $D_x = C$ for all used x (with $p(x)>0$) and is thus equally distinguishable at the receiver.

Python activities file 231: https://github.com/northeastern-datalab/cs7840-activities/tree/main/notebooks/231_Blahut-Arimato.ipynb

Blahut-Arimoto algorithm (reverse form)

Start with an arbitrary input probability $\mathbf{p}$ and alternate between updating $\mathbf{Q}$ and updating $\mathbf{p}$.

- maximize $I(X;Y)$ by choosing posterior $\mathbf{Q}$, while fixing $\mathbf{p}$:

- maximize $I(X;Y)$ by choosing $\mathbf{p}$, while fixing $\mathbf{Q}$:

Blahut-Arimoto algorithm (reverse form)

Start with an arbitrary input probability $\mathbf{p}$ and alternate between updating $\mathbf{Q}$ and updating $\mathbf{p}$.

- maximize $I(X;Y)$ by choosing posterior $\mathbf{Q}$, while fixing $\mathbf{p}$:

  Calculate $\mathbf{q} \leftarrow \mathbf{P}^{\mathrm{T}}\mathbf{p}$

  Calculate posterior matrix $\mathbf{Q}$ with $Q_{y:} = p(X|Y=y)$

  $$Q_{yx} \leftarrow \frac{p_x \cdot P_{xy}}{q_y}$$

- maximize $I(X;Y)$ by choosing $\mathbf{p}$, while fixing $\mathbf{Q}$:

  $S_x \leftarrow \sum_y P_{xy} \cdot \lg(Q_{yx})$ $S_x = \sum_y p(y|x) \cdot \lg\big(p(x|y)\big)$

  Collect these in $\mathbf{S}$

  $p_x \propto 2^{S_x}$

  $\mathbf{p} \leftarrow \mathrm{softmax}_2(\mathbf{S})$ $\mathbf{p} \leftarrow \mathrm{softmax}(\mathbf{S} \cdot \ln(2))$

S_x is an expected log-posterior score:

$$S_x = \mathbb{E}_{y\sim p(.|x)}[\lg(p(x|y))]$$

S_x is intuitively a "recognizability score".

If sending x leads the decoder (using current model $\mathbf{Q}$) to strongly conclude that x was indeed sent, then $p(x|Y) = Q(x|Y)$ is large → S_x is large → Softmax assigns more probability to x

If the channel behavior under x is ambiguous, then $Q(x|Y)$ is small → S_x is small → in the the next iteration, $p(x)$ is reduced.

Thus, the update pushes probability mass toward inputs that are easier to decode.

Blahut-Arimoto algorithm (reverse form)

Start with an arbitrary input probability $\mathbf{p}$ and alternate between updating $\mathbf{Q}$ and updating $\mathbf{p}$.

- maximize $I(X;Y)$ by choosing posterior $\mathbf{Q}$, while fixing $\mathbf{p}$:

  Calculate $\mathbf{q} \leftarrow \mathbf{P}^{\mathrm{T}}\mathbf{p}$

  Calculate posterior matrix $\mathbf{Q}$ with $Q_{y:} = p(X|Y=y)$

  $$Q_{yx} \leftarrow \frac{p_x \cdot P_{xy}}{q_y}$$

- maximize $I(X;Y)$ by choosing $\mathbf{p}$, while fixing $\mathbf{Q}$:

  $S_x \leftarrow \sum_y P_{xy} \cdot \lg(Q_{yx})$ $S_x = \sum_y p(y|x) \cdot \lg\big(p(x|y)\big)$

  Collect these in $\mathbf{S}$

  $p_x \propto 2^{S_x}$

  $\mathbf{p} \leftarrow \mathrm{softmax}_2(\mathbf{S})$ $\mathbf{p} \leftarrow \mathrm{softmax}(\mathbf{S} \cdot \ln(2))$

Connection forward / backward:

$$S_x = \sum_y P_{xy} \cdot \lg(Q_{yx})$$

$$S_x = \sum_y p(y|x) \cdot \lg\left(\frac{p(x)\cdot p(y|x)}{q(y)}\right)$$

$$= \sum_y p(y|x) \cdot \left(\lg\big(p(x)\big) + \lg\left(\frac{p(y|x)}{q(y)}\right)\right)$$

$$= \lg\big(p(x)\big) + \sum_y p(y|x) \cdot \lg\left(\frac{p(y|x)}{q(y)}\right)$$

$$= \lg\big(p(x)\big) + D_x$$

$$2^{S_x} = p(x) \cdot 2^{D_x}$$

Python activities file 231: https://github.com/northeastern-datalab/cs7840-activities/tree/main/notebooks/231_Blahut-Arimato.ipynb

Details on the reverse interpretation

Goal: maximize $I(X;Y)$. Given: $p(x|y)$ (written as $\mathbf{P} = [p(x|y)]$). We can choose $\mathbf{p} = [p(x)]$.

$$I(X;Y) = \sum_{x,y} p(x,y) \cdot \lg\left(\frac{p(x,y)}{p(x) \cdot p(y)}\right) = D_{\mathrm{KL}}(p(x,y)||p(x) \cdot p(y))$$

$$= \sum_{x,y} p(x,y) \cdot \lg\left(\frac{p(y|x)}{p(y)}\right)$$

$$= \sum_{x} p(x) \cdot \underbrace{\sum_{x,y} p(y|x) \cdot \lg\left(\frac{p(y|x)}{p(y)}\right)}_{D_x}$$

$\sum_{x'} p(x')p(y|x')$ (via $\mathbf{q} = \mathbf{P}^{\mathrm{T}}\mathbf{p}$)

This leads to the interpretation of the forward variant with multiplicative weights updates from the previous pages.

Details on the reverse interpretation

Goal: maximize $I(X;Y)$. Given: $p(x|y)$ (written as $\mathbf{P} = [p(x|y)]$). We can choose $\mathbf{p} = [p(x)]$.

$$I(X;Y) = \sum_{x,y} p(x,y) \cdot \lg\left(\frac{p(x,y)}{p(x) \cdot p(y)}\right)$$

$$= \sum_{x,y} p(x,y) \cdot \lg\left(\frac{p(y|x)}{p(y)}\right)$$

$$= \sum_{x} p(x) \cdot \underbrace{\sum_{x,y} p(y|x) \cdot \lg\left(\frac{p(y|x)}{p(y)}\right)}_{D_x}$$

$\sum_{x'} p(x')p(y|x')$ (via $\mathbf{q} = \mathbf{P}^{\mathrm{T}}\mathbf{p}$)

$$= \sum_{x,y} p(x) \cdot p(y|x) \cdot \lg\left(\frac{p(x|y)}{p(x)}\right)$$

LEMMA 1: If $p(x)$ and $p(y|x)$ are given (then $\mathbf{q} = [p(y)]$ follows), then $p(x|y)$ that maximizes $I(X;Y)$ is the posterior matrix $\mathbf{Q}$ with $Q_{:y} = p(X|Y=y)$

Thus $Q_{xy} = \frac{p_x \cdot P_{xy}}{q_y}$ (via $\mathbf{q} = \mathbf{P}^{\mathrm{T}}\mathbf{p}$)

$$p(x|y) = \frac{p(x) \cdot p(y|x)}{p(y)}$$

This leads to the interpretation of the forward variant with multiplicative weights updates from the previous pages.

Details on the reverse interpretation

Goal: maximize $I(X;Y)$. Given: $p(x|y)$ (written as $\mathbf{P} = [p(x|y)]$). We can choose $\mathbf{p} = [p(x)]$.

$$I(X;Y) = \sum_{x,y} p(x,y) \cdot \lg\left(\frac{p(x,y)}{p(x)\cdot p(y)}\right)$$

$$= \sum_{x,y} p(x,y) \cdot \lg\left(\frac{p(y|x)}{p(y)}\right)$$

$$= \sum_{x} p(x) \cdot \underbrace{\sum_{x,y} p(y|x) \cdot \lg\left(\frac{p(y|x)}{p(y)}\right)}_{D_x}$$

$\sum_{x'} p(x')p(y|x')$ (via $\mathbf{q} = \mathbf{P}^{\mathrm{T}}\mathbf{p}$)

$$= \sum_{x,y} p(x) \cdot p(y|x) \cdot \lg\left(\frac{p(x|y)}{p(x)}\right)$$

Step 1: max $p(x|y)$

$$= \sum_{x,y} p(x) \cdot p(y|x) \cdot \lg\left(\frac{Q_{yx}}{p(x)}\right)$$

sums up to 1 over y

$$= \sum_{x} p(x) \cdot \underbrace{\sum_{y} p(y|x) \cdot \lg(Q_{yx})}_{:= S_x} \underbrace{- \sum_{x,y} p(x) \cdot \lg(p(x)) \cdot p(y|x)}_{= +H(\mathbf{p})}$$

This leads to the interpretation of the forward variant with multiplicative weights updates from the previous pages.

LEMMA 2: The softmax is the solution to

$$\operatorname*{argmax}_{\mathbf{p}(x)\in\Delta} \left[\left(\sum_x p(x) S_x\right) + H(\mathbf{p})\right]$$

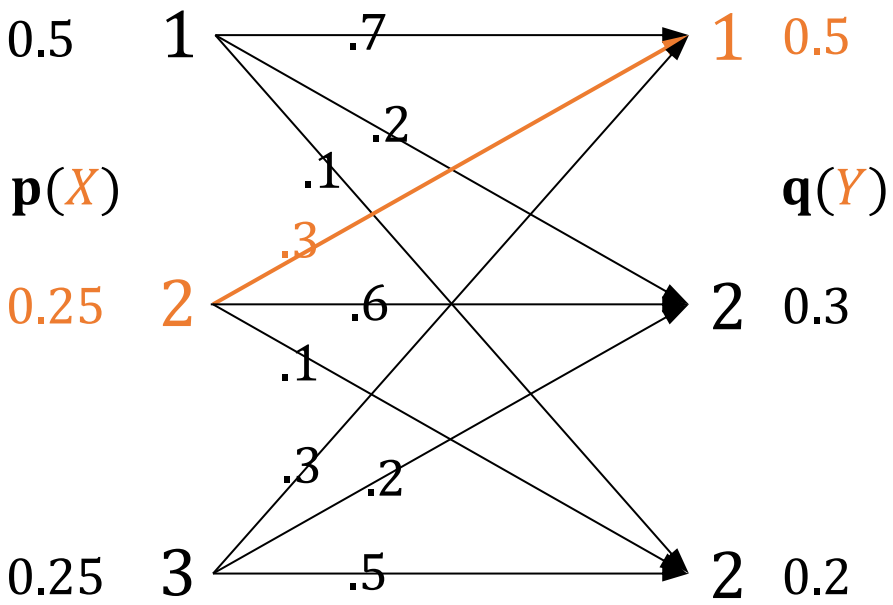
$$\mathbf{p} = \mathrm{softmax}_2(\mathbf{S}) \qquad p_i = \frac{2^{S_i}}{\sum_j 2^{S_j}}$$

Detail: Posterior matrix $Q(X|Y)$

Given fixed $\mathbf{p} = p(X)$ and $\mathbf{P} = [p(Y|X)]$.

Calculate posterior matrix $\mathbf{Q}$ with $Q_{y:} = p(X|Y = y)$

$$Q_{yx} = \frac{p_x \cdot P_{xy}}{q_y}$$



row-stochastic → y

$$\mathbf{P} = \begin{pmatrix} .7 & .2 & .1 \\ .3 & .6 & .1 \\ .3 & .2 & .5 \end{pmatrix} \quad \mathbf{p}(x) =: \mathbf{p} = \begin{pmatrix} 0.5 \\ 0.25 \\ 0.25 \end{pmatrix} \quad \mathbf{p}(y) =: \mathbf{q} = \begin{pmatrix} 0.5 \\ 0.3 \\ 0.2 \end{pmatrix}$$

x

Notice some textbooks write column-stochastic Q_{xy} instead of row-stochastic Q_{yx}

row-stochastic → x

$$\mathbf{Q} = \begin{pmatrix} .7 & .15 & .15 \\ .3 & .5 & .16 \\ .25 & .125 & .625 \end{pmatrix}$$

y

$$Q_{12} = \frac{p_2 P_{21}}{q_1} = \frac{0.25 \cdot 0.3}{0.5} = 0.15$$

$\mathbf{Q}_{1:}$ is $\mathbb{P}(X|Y = 1)$

each row $\mathbf{Q}_{y:}$ is a posterior $\mathbb{P}(X|Y = y)$

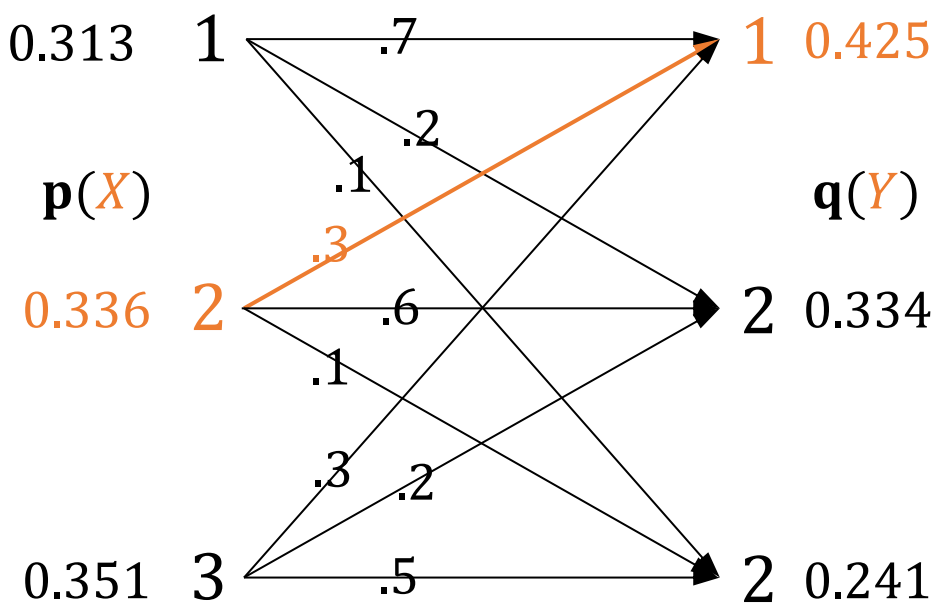
Python activities file 231: https://github.com/northeastern-datalab/cs7840-activities/tree/main/notebooks/231_Blahut-Arimato.ipynb

Detail: Posterior matrix $Q(X|Y)$

Given fixed $\mathbf{p} = p(X)$ and $\mathbf{P} = [p(Y|X)]$.

Calculate posterior matrix $\mathbf{Q}$ with $Q_{:y} = p(X|Y = y)$

$$Q_{yx} = \frac{p_x \cdot P_{xy}}{q_y}$$



row-stochastic

$$\mathbf{P} = \begin{pmatrix} .7 & .2 & .1 \\ .3 & .6 & .1 \\ .3 & .2 & .5 \end{pmatrix} \quad (x \downarrow,\ y \rightarrow)$$

$$\mathbf{p}(x) =: \mathbf{p} = \begin{pmatrix} 0.313 \\ 0.336 \\ 0.351 \end{pmatrix} \qquad \mathbf{p}(y) =: \mathbf{q} = \begin{pmatrix} 0.425 \\ 0.334 \\ 0.241 \end{pmatrix}$$

Notice some textbooks write column-stochastic Q_{xy} instead of row-stochastic Q_{yx}

row-stochastic

$$\mathbf{Q} = \begin{pmatrix} .515 & .237 & .248 \\ .187 & .603 & .210 \\ .130 & .140 & .730 \end{pmatrix} \quad (y \downarrow,\ x \rightarrow)$$

$$Q_{12} = \frac{p_2 P_{21}}{q_1} = \frac{0.336 \cdot 0.3}{0.425} = 0.237$$

$\mathbf{Q}_{1:}$ is $\mathbb{P}(X|Y = 1)$

each row $\mathbf{Q}_{y:}$ is a posterior $\mathbb{P}(X|Y = y)$

Python activities file 231: https://github.com/northeastern-datalab/cs7840-activities/tree/main/notebooks/231_Blahut-Arimato.ipynb

Detail: Softmax maximizes linear reward + entropy

CLAIM $\mathbf{p} = \text{softmax}_2(\mathbf{S})$ $\quad p_i = \frac{2^{S_i}}{\sum_j 2^{S_j}}$

Is the solution to the optimization problem $\underset{\mathbf{p}\in\Delta}{\text{argmax}} \left[\left(\sum_i p_i S_i\right) + H(\mathbf{p})\right]$ ($\mathbb{E}_{i\sim p}[S_i]$)

Entropy regularization adds a "keep it uncertain" bias to optimization. The unique optimizer of a linear objective plus entropy regularization is the softmax distribution.

PROOF Jensen's inequality for concave f:

$$f(\mathbb{E}[X]) \geq \mathbb{E}[f(X)]$$

$$\log(\mathbb{E}[X]) \geq \mathbb{E}[\log(X)] \qquad \log(X) \text{ is concave}$$

$$\log\left(\sum_i p_i \cdot X_i\right) \geq \sum_i p_i \cdot \log(X_i)$$

equality for: $X_i = X_j$

$$\log\left(\sum_i p_i \cdot \left(\frac{a_i}{p_i}\right)\right) \geq \sum_i p_i \cdot \log\left(\frac{a_i}{p_i}\right)$$

equality for: $\frac{a_i}{p_i} = \frac{a_j}{p_j}$

$$\log\left(\sum_i a_i\right) \geq \sum_i p_i \cdot \log\left(\frac{a_i}{p_i}\right)$$

set $a_i = 2^{S_i}$ and use $\lg = \log_2$

$$\lg\left(\sum_i 2^{S_i}\right) \geq \sum_i p_i \cdot \left(S_i - \lg(p_i)\right)$$

$$= \sum_i \left(p_i S_i - p_i \lg(p_i)\right)$$

$$= \left(\sum_i p_i S_i\right) + H(p)$$

Equality holds for $\frac{2^{S_i}}{p_i} = \frac{2^{S_j}}{p_j} = C$.

Hence, $p_i = \frac{2^{S_i}}{C}$.

From from $\sum_i p_i = 1$, we get $C = \sum_j 2^{S_j}$.

Hence, $p_i = \frac{2^{S_i}}{\sum_j 2^{S_j}}$.

Maximizing mutual information (capacity). We have a fixed $P_{Y|X}$ and the optimization problem

$$C = \max_{P_X} I(X;Y) = \max_{P_X} \max_{Q_{X|Y}} \mathbb{E}_{P_{X,Y}} \left[\log \frac{Q_{X|Y}}{P_X} \right] ,$$

where in the second equality we invoked (4.7). This results in the iterations:

$$Q_{X|Y}(x|y) \leftarrow \frac{1}{Z(y)} P_X(x) P_{Y|X}(y|x)$$

$$P_X(x) \leftarrow Q'(x) \triangleq \frac{1}{Z} \exp \left\{ \sum_y P_{Y|X}(y|x) \log Q_{X|Y}(x|y) \right\} ,$$

where $Z(y)$ and Z are normalization constants. To derive this, notice that for a fixed P_X the optimal $Q_{X|Y} = P_{X|Y}$. For a fixed $Q_{X|Y}$, we can see that

$$\mathbb{E}_{P_{X,Y}} \left[\log \frac{Q_{X|Y}}{P_X} \right] = \log Z - D(P_X \| Q') ,$$

and thus the optimal $P_X = Q'$.

Denoting P_n to be the value of P_X at the nth iteration, we observe that

$$I(P_n, P_{Y|X}) \le C \le \sup_x D(P_{Y|X=x} \| P_{Y|X} \circ P_n) . \tag{5.29}$$

This is useful since at every iteration not only we get an estimate of the optimizer P_n, but also the gap to optimality $C - I(P_n, P_{Y|X}) \le C - \text{RHS}$. It can be shown, furthermore, that both RHS and LHS in (5.29) monotonically converge to C as $n \to \infty$, see [113] for details.

If $I(X;Y) < \infty$ then

$$I(X;Y) = \sup_{Q_{X|Y}} \mathbb{E}_{P_{X,Y}} \left[\log \frac{dQ_{X|Y}}{dP_X} \right] , \tag{4.7}$$

where the supremum is over Markov kernels $Q_{X|Y}$ as in the first sentence.

There are two mathematically equivalent forms of the Blahut-Arimoto (BA) algorithm:
1. we started with the forward form that optimizes for $\mathbf{p}$, using transition matrix $\mathbf{P}$ or $\mathbf{p}(y|x)$.
2. Here and on Wikipedia, you find the reverse form that optimizes for $\mathbf{p}(x|y) = \mathbf{Q}_{yx}$ using Bayes' law.

Polyanskiy, Wu. *Information Theory: From Coding to Learning.* Cambridge University Press 2025. https://doi.org/10.1017/9781108966351
See also Wikipedia: https://en.wikipedia.org/wiki/Blahut%E2%80%93Arimoto_algorithm#Algorithm_for_Channel_Capacity
Gatterbauer. Foundations and Applications of Information Theory: https://northeastern-datalab.github.io/cs7840/fa25/

Error Correcting Codes

Efficient error correcting codes are quite complicated

Chapter 14

Polar Codes

In contrast to source coding where soon after the fundamental result (Theorem 5.2) was published by Shannon, an optimal scheme was found (Huffman coding), channel coding proved to be a much harder nut to crack. For years engineers tried to find a practical system (i.e., one with manageable complexity) that would approach the performance of an optimal coding scheme. The irony is that a randomly picked system will quite likely work very well (as can be seen from the random coding proof of Shannon!), but any such system is impossible in practice as there is no structure in the code that would allow efficient encoding and decoding. All investigated structured codes like, e.g., algebraic codes that use the structure of vector spaces and subspaces, turned out to be far from optimal.

The first real breakthrough was the discovery of *turbo codes* [BGT93] in 1993 (see the end of Chapter 17 for a more detailed discussion). Nevertheless, turbo codes (and also the even more efficient *low-density parity-check (LDPC) codes* [MN96], [DM98]) are not proven to be good (or even optimal), but are known to perform well simply from experience.

The first *provably* capacity-achieving coding scheme that at the same time also has decent complexity is *polar coding* introduced by Erdal Arıkan in 2007 [Arı09]. In this chapter we are going to study Arıkan's idea in its original form that was restricted to binary-input DMCs (with an arbitrary finite output alphabet).

The discovery of a first deterministic construction of a capacity-achieving coding scheme is one of the chief breakthroughs in information theory of the first decade of the 21st century. Polar coding still suffers from some drawbacks that at the moment still hampers its widespread use, but there is a lot of research effort put into these issues.

Moser. *Information Theory (lecture Notes, 6th ed)*, 2018. https://moser-isi.ethz.ch/scripts.html
Gatterbauer. Foundations and Applications of Information Theory: https://northeastern-datalab.github.io/cs7840/fa25/

Efficient error correcting codes are quite complicated

7.11 HAMMING CODES

The channel coding theorem promises the existence of block codes that will allow us to transmit information at rates below capacity with an arbitrarily small probability of error if the block length is large enough. Ever since the appearance of Shannon's original paper [471], people have searched for such codes. In addition to achieving low probabilities of error, useful codes should be "simple," so that they can be encoded and decoded efficiently.

The search for simple good codes has come a long way since the publication of Shannon's original paper in 1948. The entire field of coding theory has been developed during this search. We will not be able to describe the many elegant and intricate coding schemes that have been developed since 1948. We will only describe the simplest such scheme developed by Hamming [266]. It illustrates some of the basic ideas underlying most codes.

We may still try to gain intuition on those towards the end of the class (if we have time)

Moser. *Information Theory (lecture Notes, 6th ed)*, 2018. https://moser-isi.ethz.ch/scripts.html
Gatterbauer. Foundations and Applications of Information Theory: https://northeastern-datalab.github.io/cs7840/fa25/

Duality (max / min perspectives on mutual information)

- Data compression:
  - we remove all the redundancy in the data to form the most compressed version possible

- Data transmission:
  - we add redundancy in a controlled manner to combat errors in the channel

Arithmetic Circuits, Structured Matrices and (not so) Deep Learning

ATRI RUDRA

Department of Computer Science and Engineering
University at Buffalo
atri@buffalo.edu

5.1 Fast Fourier Transform (FFT)

As mentioned earlier, a vast majority of efficient matrix vector multiplication algorithms are equivalent to small (both in size and depth) linear arithmetic circuit. For example the FFT can be thought of as an efficient arithmetic circuit to compute the Discrete Fourier Transform (indeed when one converts the linear arithmetic circuit for FFT into a matrix decomposition, then each matrix in the decomposition is so called *Butterfly matrix*, with each block matrix in each factor being the same). For an illustration of this consider the DFT with $n = 4$ as illustrated in Figure 1.

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -i & -1 & i \\ 1 & -1 & 1 & -1 \\ 1 & i & -1 & -i \end{bmatrix}$$

Figure 1: DFT of order 4.

Figure 2 represent the arithmetic circuit corresponding to FFT with $n = 4$.

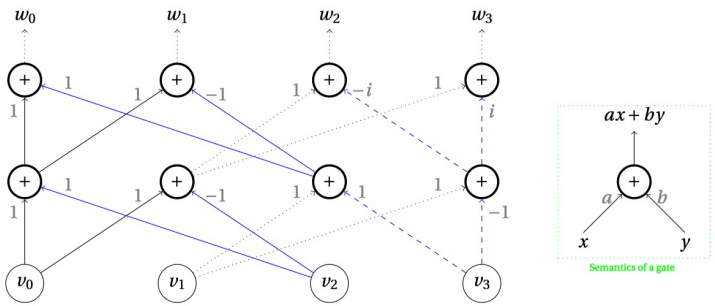


Figure 2: Arithmetic circuit for 4-DFT from Figure 1.

Finally, Figure 3 is representation of the arithmetic circuit of Figure 2 as a product of a butterfly matrix and (the bit-reversal) permutation.

FLASHATTENTION: Fast and Memory-Efficient Exact Attention with IO-Awareness

Tri Dao[†], **Daniel Y. Fu** [†], **Stefano Ermon** [†], **Atri Rudra** [‡], **Christopher Ré** [†]
[†] Department of Computer Science, Stanford University
[‡] Department of Computer Science and Engineering, University at Buffalo, SUNY
{trid,danfu}@stanford.edu, ermon@stanford.edu, atri@buffalo.edu, chrismre@cs.stanford.edu

Updated 11/17/2025

Part 3: Applications
L19: Channel Capacity, Distortion Theory, Information Bottleneck
[Distortion Theory]

Wolfgang Gatterbauer

cs7840 Foundations and Applications of Information Theory (fa25)

https://northeastern-datalab.github.io/cs7840/fa25/

11/17/2025

Pre-class conversations

- Last class recapitulation
- Projects: commented on 5/9 projects so far
  - Figures, Python notebook outputs (ideally in the appendix) are *highly* encourage and do not count against page count for projects

- Last time:
  - Zsolt on a softmax variant
  - Channel capacity with Blahut-Arimato
  - Intro to distortion (I move the slides to after this intro)
- Today:
  - Distortion theory
  - information bottleneck theory

- **Lecture 18 (Thu, Nov 13) / P3 Intermediate report**
  30min remote guest lecture by Zsolt Zombori on [Zombori+'23] Towards Unbiased Exploration in Partial Label Learning,
  Channel capacity, Distortion Theory, Information Bottleneck (2/4)
  [Cover Thomas'06: Ch 10 Distortion Theory]
- **Lecture 19 (Mon, Nov 17)**
  Channel capacity, Distortion Theory, Information Bottleneck (3/4)
  [Cover Thomas'06: Ch 10 Distortion Theory]
  Python notebooks on Rate Distortion and Quantization: 232
  Information bottleneck theory: [Zaslavsky+'18], [Webb+'24]
- **Lecture 20 (Thu, Nov 20)**
  Channel capacity, Distortion Theory, Information Bottleneck (4/4)
  Khowledge distillation [Hinton+'15]

PART 4: The axiomatic approach (deriving formulations from first principles)

Covers the axiomatic approach from multiple angles: a few simple principles (axioms) leading to entropy or the laws of probability up to factors. Starting from a list of postulates leading to particular solution is a powerful approach that has been used across different areas of computer science (e.g. how to define the right scoring function for achieving a desired outcome)

- **Lecture 21 (Mon, Nov 24)**
  Shapley values, Communication Complexity, Coding with errors
- **(Thu 11/27): no class (Fall break)**
- **skipped**
  Derivation of Hartley measure and entropy function from first principles
- **skipped**
  Cox's theorem: a derivation of the laws of probability theory from a certain set of postulates. Contrast with Kolmogorov's "probability axioms"

PART 5: Project presentations

- **Lecture 24 (Thu 12/4): P4 Project presentations / P5 Final report**
- **Lecture 25 (Mon 12/8): P4 Project presentations / P5 Final report**
- **Lecture 26 (Thu 12/11): P4 Project presentations / P5 Final report**

Examples for "The axiomatic approach" across CS

Title: Quantifying Inefficiency
When: Wednesday, Nov 12 @ 12:00–1:25 p.m.
Where: West Village H **#366** (440 Huntington Avenue, Boston, MA 02115)
Faculty Host: Mahsa Derakhshan

Abstract:
The mainstream view within economic theory is that an individual's cardinal utility values are mere representations of the individual's ordinal preferences (over lotteries over alternative outcomes). As such, each individual's cardinal utilities are only unique up to monotone affine transformations. This poses challenges for the social aggregation of utilities, and furthermore for forming a foundation for approximation theorems for social efficiency. We axiomatically define a cardinal social inefficiency function, which, given a set of alternative outcomes and individuals' preferences over these alternatives, assigns a unique number—the social inefficiency—to each alternative. These numbers—and not only their order—are uniquely defined by our axioms despite no exogenously given interpersonal comparison, outside option, or disagreement point. We interpret these numbers as per capita losses in endogenously normalized utility. We apply our social inefficiency function to a setting in which interpersonal comparison is notoriously hard to justify—object allocation without money—leveraging techniques from the Price-of-Anarchy literature to prove an approximate-efficiency result for the widely used Random Serial Dictatorship mechanism.

Joint work with Ella Segev.

Bio:
Yannai A. Gonczarowski is an Assistant Professor of Economics and of Computer Science at Harvard University—the first faculty member at Harvard to have been appointed to both of these departments. Interested in both economic theory and theoretical computer science, Yannai explores computer-science-inspired economics: he harnesses approaches, aesthetics, and techniques traditionally originating in computer science to derive economically meaningful insights. Yannai received his PhD from the Departments of Mathematics and Computer Science, and the Center for the Study of Rationality, at the Hebrew University of Jerusalem. Yannai is also a professionally-trained opera singer, having acquired a bachelor's degree and a master's degree in Classical Singing at the Jerusalem Academy of Music and Dance. Yannai's doctoral dissertation was recognized with several awards, including the Michael B. Maschler Prize of the Israeli Chapter of the Game Theory Society and the ACM SIGecom Doctoral Dissertation Award. For the design and implementation of the National Matching System for Gap-Year Programs in Israel, he was awarded the inaugural INFORMS AMD Michael H. Rothkopf Junior Researcher Paper Prize (first place). Yannai was also the recipient of the inaugural ACM SIGecom Award for Best Presentation by a Student or Postdoctoral Researcher. His first textbook, "Mathematical Logic through Python" (Gonczarowski and Nisan), which introduces a new approach to teaching the material of a basic Logic course to Computer Science students, tailored to the unique intuitions and strengths of this cohort of students, was published by Cambridge University Press.

KAN: Kolmogorov–Arnold Networks

Ziming Liu[1,4*] **Yixuan Wang**[2] **Sachin Vaidya**[1] **Fabian Ruehle**[3,4]
James Halverson[3,4] **Marin Soljačić**[1,4] **Thomas Y. Hou**[2] **Max Tegmark**[1,4]
[1] Massachusetts Institute of Technology
[2] California Institute of Technology
[3] Northeastern University
[4] The NSF Institute for Artificial Intelligence and Fundamental Interactions

Model	Multi-Layer Perceptron (MLP)	Kolmogorov-Arnold Network (KAN)
Theorem	Universal Approximation Theorem	Kolmogorov-Arnold Representation Theorem
Formula (Shallow)	$f(\mathbf{x}) \approx \sum_{i=1}^{N(\epsilon)} a_i \sigma(\mathbf{w}_i \cdot \mathbf{x} + b_i)$	$f(\mathbf{x}) = \sum_{q=1}^{2n+1} \Phi_q \left(\sum_{p=1}^{n} \phi_{q,p}(x_p) \right)$
Model (Shallow)	(a) *fixed* activation functions on *nodes*; *learnable* weights on *edges*	(b) *learnable* activation functions on *edges*; sum operation on *nodes*
Formula (Deep)	$\mathrm{MLP}(\mathbf{x}) = (\mathbf{W}_3 \circ \sigma_2 \circ \mathbf{W}_2 \circ \sigma_1 \circ \mathbf{W}_1)(\mathbf{x})$	$\mathrm{KAN}(\mathbf{x}) = (\mathbf{\Phi}_3 \circ \mathbf{\Phi}_2 \circ \mathbf{\Phi}_1)(\mathbf{x})$
Model (Deep)	(c) $\mathrm{MLP}(\mathbf{x})$, $\mathbf{W}_3$, σ_2, $\mathbf{W}_2$, σ_1, $\mathbf{W}_1$, $\mathbf{x}$; *nonlinear, fixed*; *linear, learnable*	(d) $\mathrm{KAN}(\mathbf{x})$, $\mathbf{\Phi}_3$, $\mathbf{\Phi}_2$, $\mathbf{\Phi}_1$, $\mathbf{x}$; *nonlinear, learnable*

Figure 0.1: Multi-Layer Perceptrons (MLPs) vs. Kolmogorov-Arnold Networks (KANs)

Kan: Kolmogorov-arnold networks
Z Liu, Y Wang, S Vaidya, F Ruehle, J Halverson… - arXiv preprint arXiv …, 2024 - arxiv.org
… We instead focus on the **Kolmogorov-Arnold** representation theorem, which can be realized … **network** called **Kolmogorov-Arnold networks** (**KAN**). We review the **Kolmogorov-Arnold** …
☆ Save 99 Cite Cited by 2548 Related articles All 13 versions »

Dissociation and Propagation for Approximate Lifted Inference with Standard Relational Database Management Systems

Wolfgang Gatterbauer · **Dan Suciu**

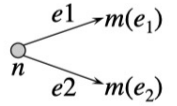
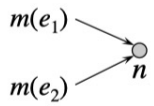
	Messages across edge e	Relevance of node n
		
Pseudo-probabilistic	$m(e_i) \leftarrow p_{e_1} \cdot \rho(n)$ **product**	$\rho(n) \leftarrow \bigotimes_{e_i} m(e_i)$ **independent-or**
PageRank	$m(e_i) \leftarrow \frac{1}{d_n} \cdot \rho(n)$ **product**	$\rho(n) \leftarrow \sum_{e_i} m(e_i)$ **addition**
Belief propagation	$\mathbf{m}(e_i) \leftarrow \psi_{e_i} \cdot \boldsymbol{\rho}_{\setminus e_i}(n)$ **matrix-vector product**	$\boldsymbol{\rho}(n) \leftarrow \frac{1}{Z} \bigodot_{e_i} \mathbf{m}(e_i)$ **component-wise prod.**
Linearized belief prop.	$\mathbf{m}(e_i) \leftarrow \psi_{e_i} \cdot \boldsymbol{\rho}_{\setminus e_i}(n)$ **matrix-vector product**	$\boldsymbol{\rho}(n) \leftarrow \sum_{e_i} \mathbf{m}(e_i)$ **addition**

Fig. 2 "*Relevance propagation*" in graphs works by iteratively calculating messages $m(e)$ across edges e and relevance scores $\rho(n)$ of nodes n. The propagation method we consider is *pseudoprobabilistic* in that the two operators are "*independent-and*" or product $(\cdot)$, and "*independent-or*" $(\otimes)$. PageRank and related methods from semi-supervised learning replace the probability p_e of an edge with a weight (here d_n stands for the out-degree of node n) and the independent-or with addition or sum $(\sum)$. Belief Propagation propagates not just one message across an edge but a vector $\mathbf{m}(e)$ of messages, scales this message vector with a matrix ψ_e (also called "edge potential"), and replaces the independent-or with a component-wise product $(\odot)$, followed by a normalization (here Z stands for a normalizer). Linearized Belief Propagation uses again addition as second operator and requires no normalization. Intuitively, the method developed in this paper generalizes pseudoprobabilistic relevance propagation to hypergraphs.

Liu, Wang, Vaidya, Ruehle, Halverson, Soljačić, Hou, Tegmark. *KAN: Kolmogorov-Arnold Networks*, ICLR 2025. https://arxiv.org/abs/2404.19756 ,
Gatterbauer, Suciu. *Dissociation and propagation for approximate lifted inference with standard relational database management systems*, VLDBJ 2017. https://arxiv.org/abs/1310.6257
Gatterbauer. Foundations and Applications of Information Theory: https://northeastern-datalab.github.io/cs7840/fa25/

Distortion Theory

Largely based on chapter 10 of
[Cover, Thomas'06] Elements of Information Theory, 2006. https://www.doi.org/10.1002/047174882X

Rate distortion theory

- A finite representation of a continuous RV can never be perfect
- How well can we represent it?
- Requires a notion of "goodness" of a representation
  - Distortion measure: distance between RV and its representation

- Rate distortion theory:
  - Given: source distribution p and a distortion measure d
  - Describes: trade-off between communication rate R and distortion d
  - Lossy compression framework with zero-error data compression (earlier topics in class, to be seen if we revisit later again) a special case

Quantization

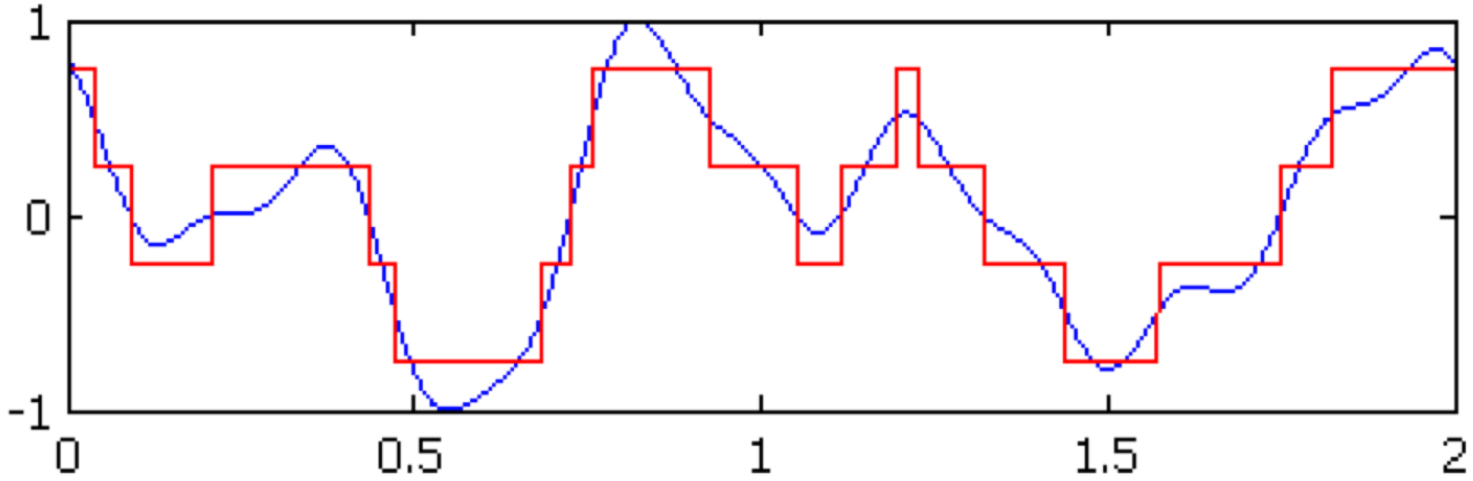
- Let X be a continuous RV (e.g. from a Gaussian distribution)
- We approximate X by $\hat{X}$
- Using R bits to represent X, then $\hat{X}(X)$ has 2^R possible values
  - Example $R = 8$ bits, then then $\hat{X}$ has how many possible values?

Quantization

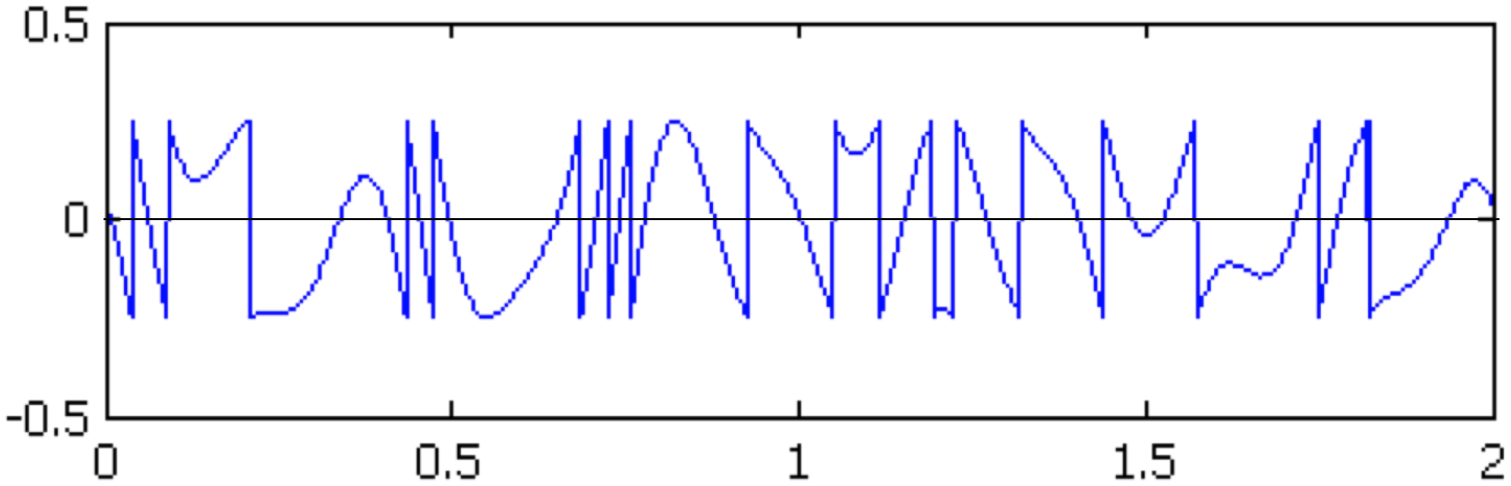
- Let X be a continuous RV (e.g. from a Gaussian distribution)
- We approximate X by $\hat{X}$
- Using R bits to represent X, then $\hat{X}(X)$ has 2^R possible values
  - Example $R = 8$ bits, then then $\hat{X}$ has $2^8 = 256$ possible values

- Goal: find the optimal set of values ("**representatives**") for $\hat{X}$ and associated regions ("**assignment regions**") for each value

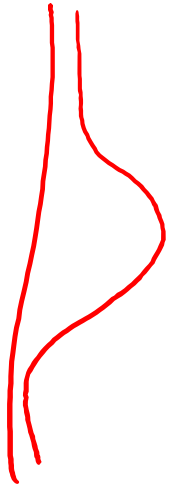
Original and Quantized Signal



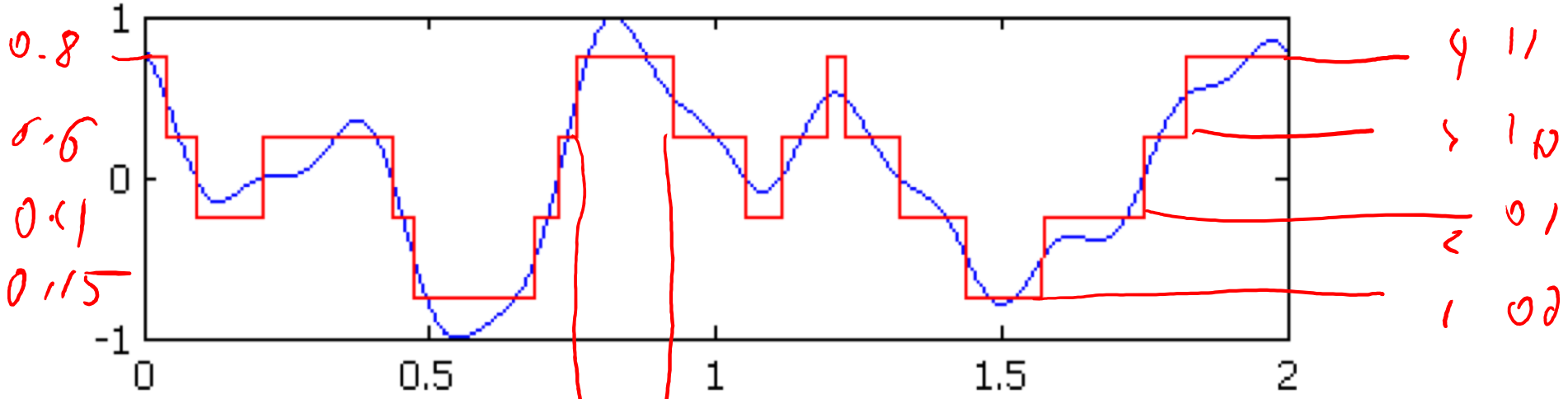
Quantization Error



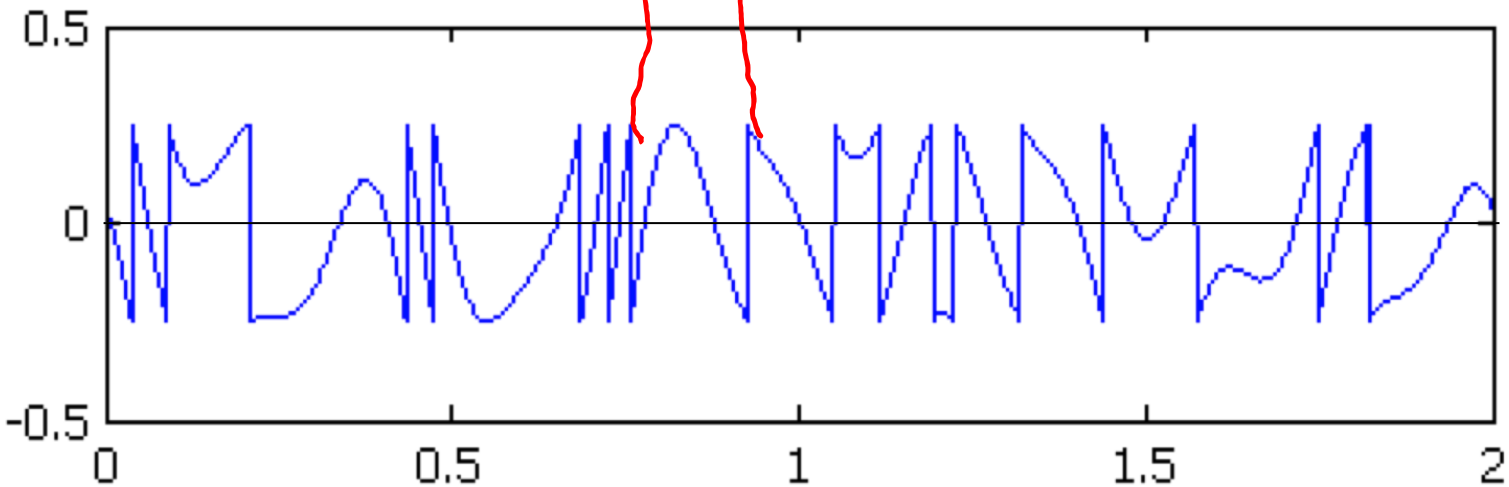
Source: Bernd Girod, 2012: https://web.stanford.edu/class/ee398a/handouts/lectures/05-Quantization.pdf
Gatterbauer. Foundations and Applications of Information Theory: https://northeastern-datalab.github.io/cs7840/fa25/



Original and Quantized Signal

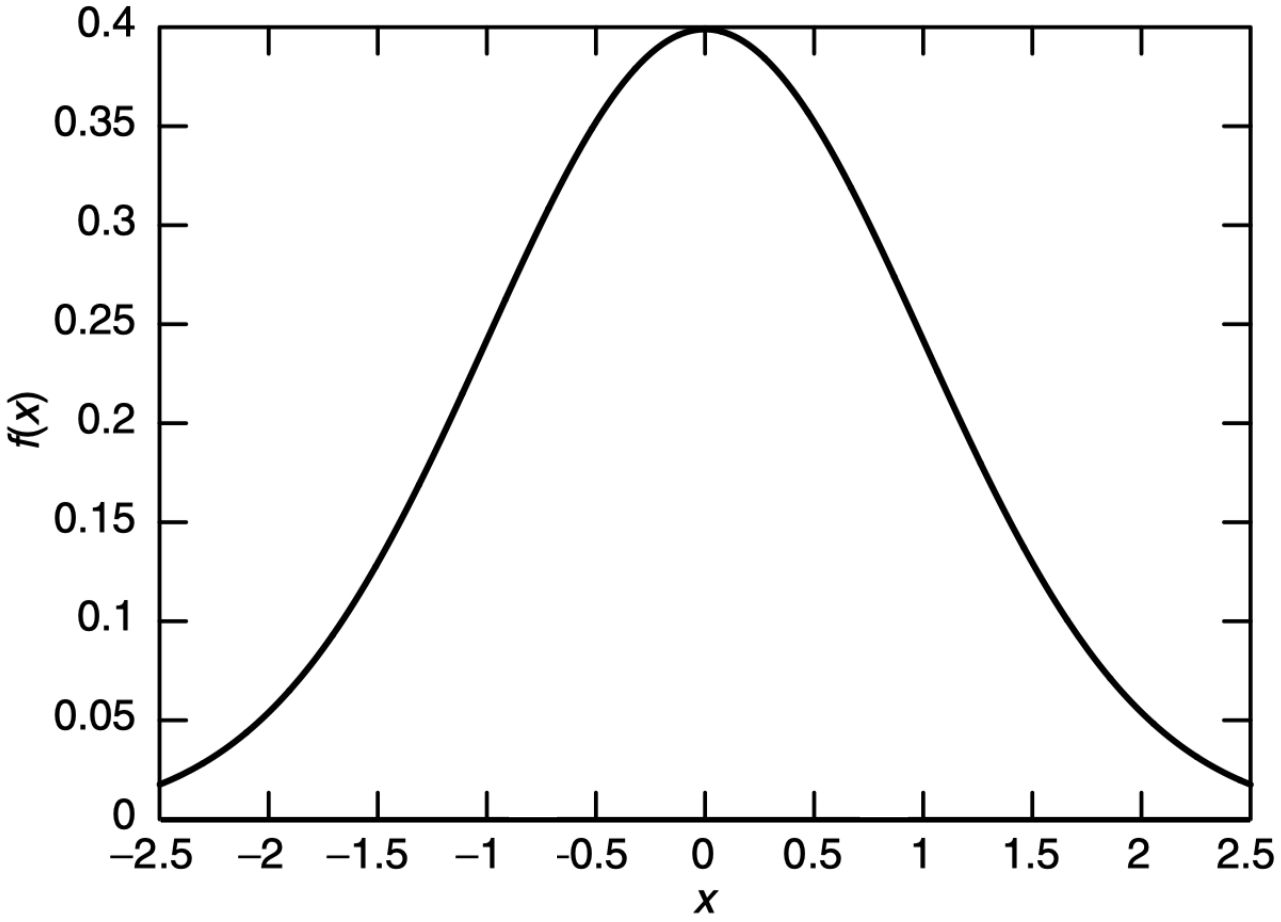


Quantization Error



Source: Bernd Girod, 2012: https://web.stanford.edu/class/ee398a/handouts/lectures/05-Quantization.pdf
Gatterbauer. Foundations and Applications of Information Theory: https://northeastern-datalab.github.io/cs7840/fa25/

Quantization of a Gaussian



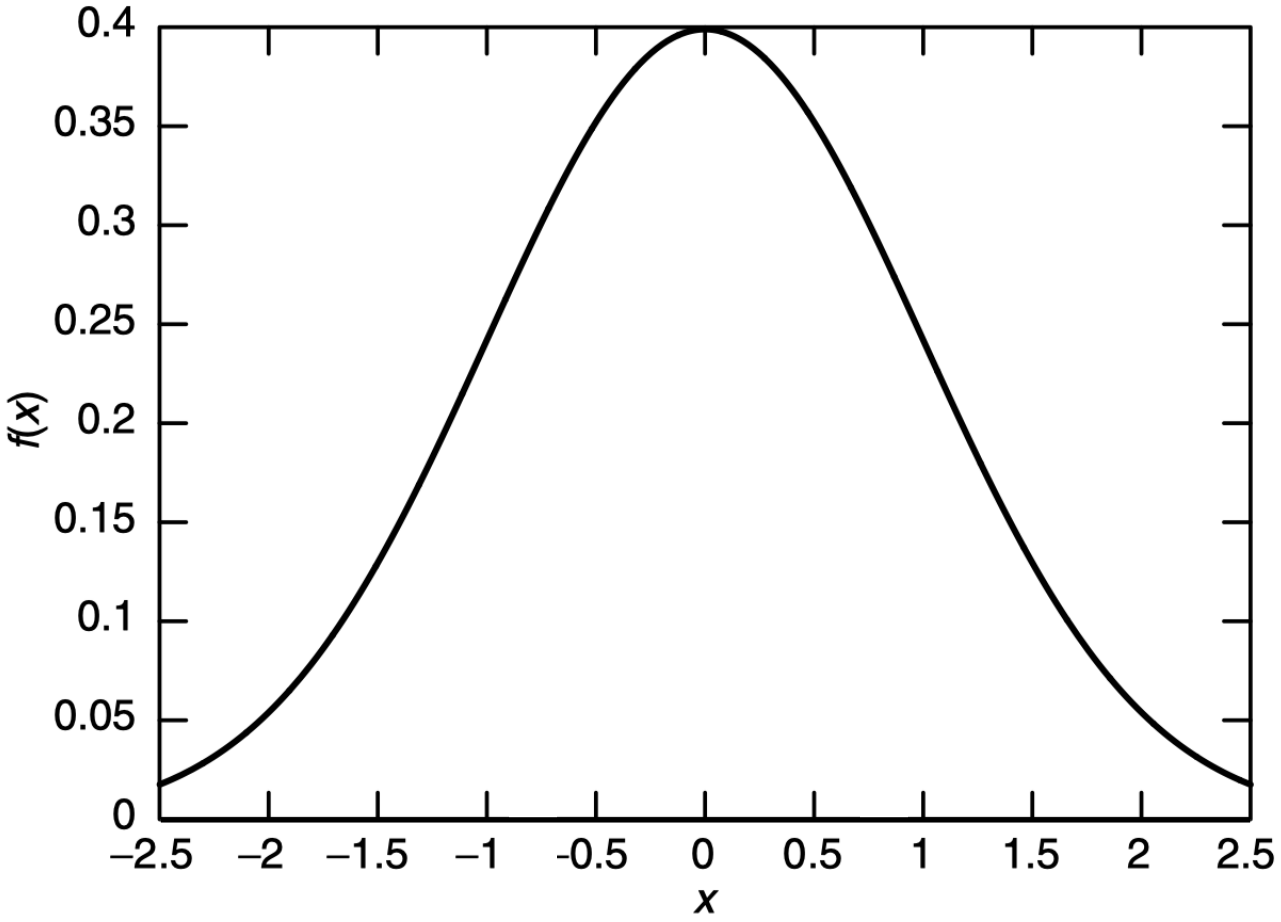
Assume you have $R = 1$ bit (2 values). What is the best way to quantize a Gaussian distribution

?

What is an appropriate measure of distortion

?

Quantization of a Gaussian



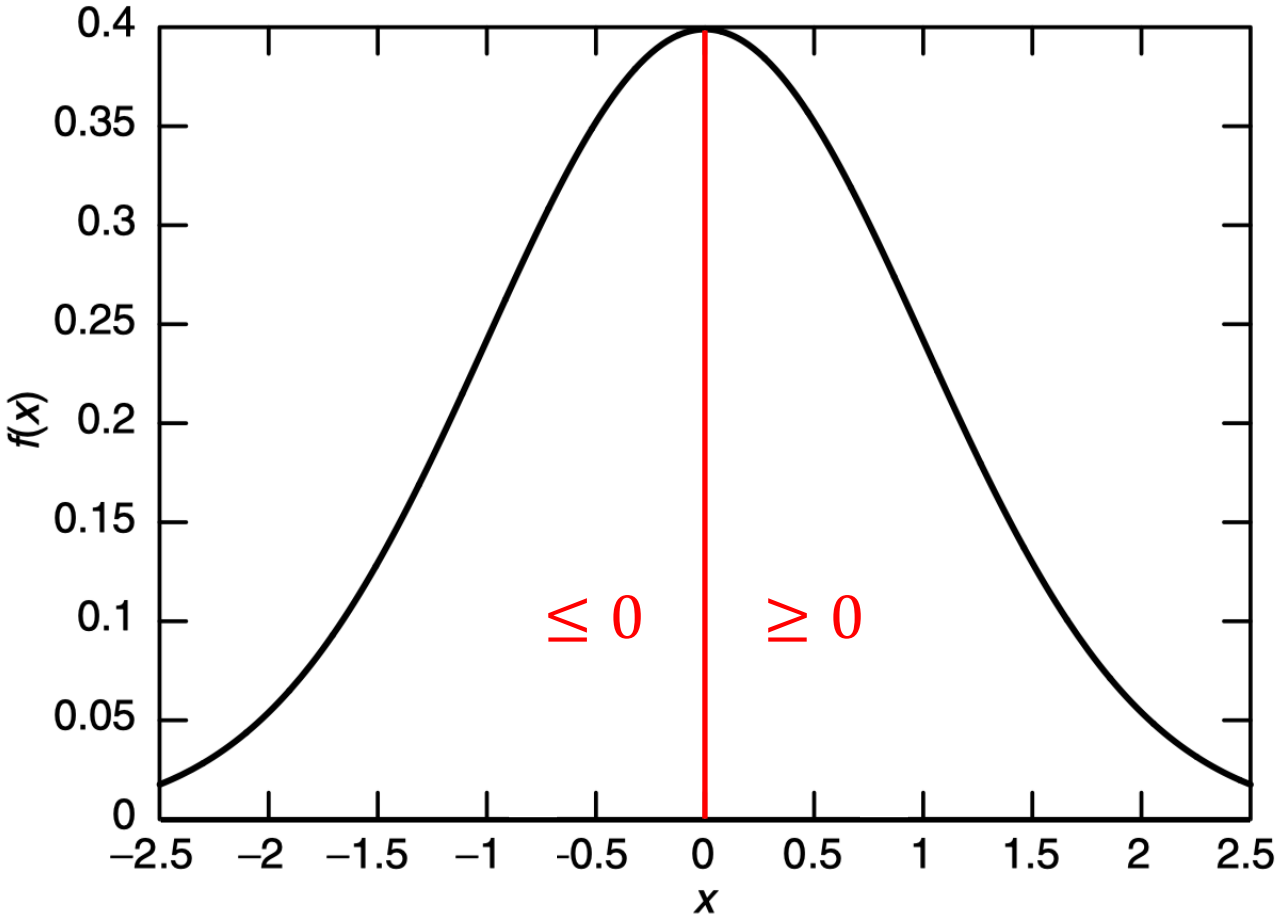
Assume you have $R = 1$ bit (2 values). What is the best way to quantize a Gaussian distribution

?

Assume we like to minimize the mean of squared errors (MSE)

Recall from our probability primer: The mean minimizes the sum of squared errors, and thus also the MSE (while the median minimizes the sum of absolute errors).

Quantization of a Gaussian



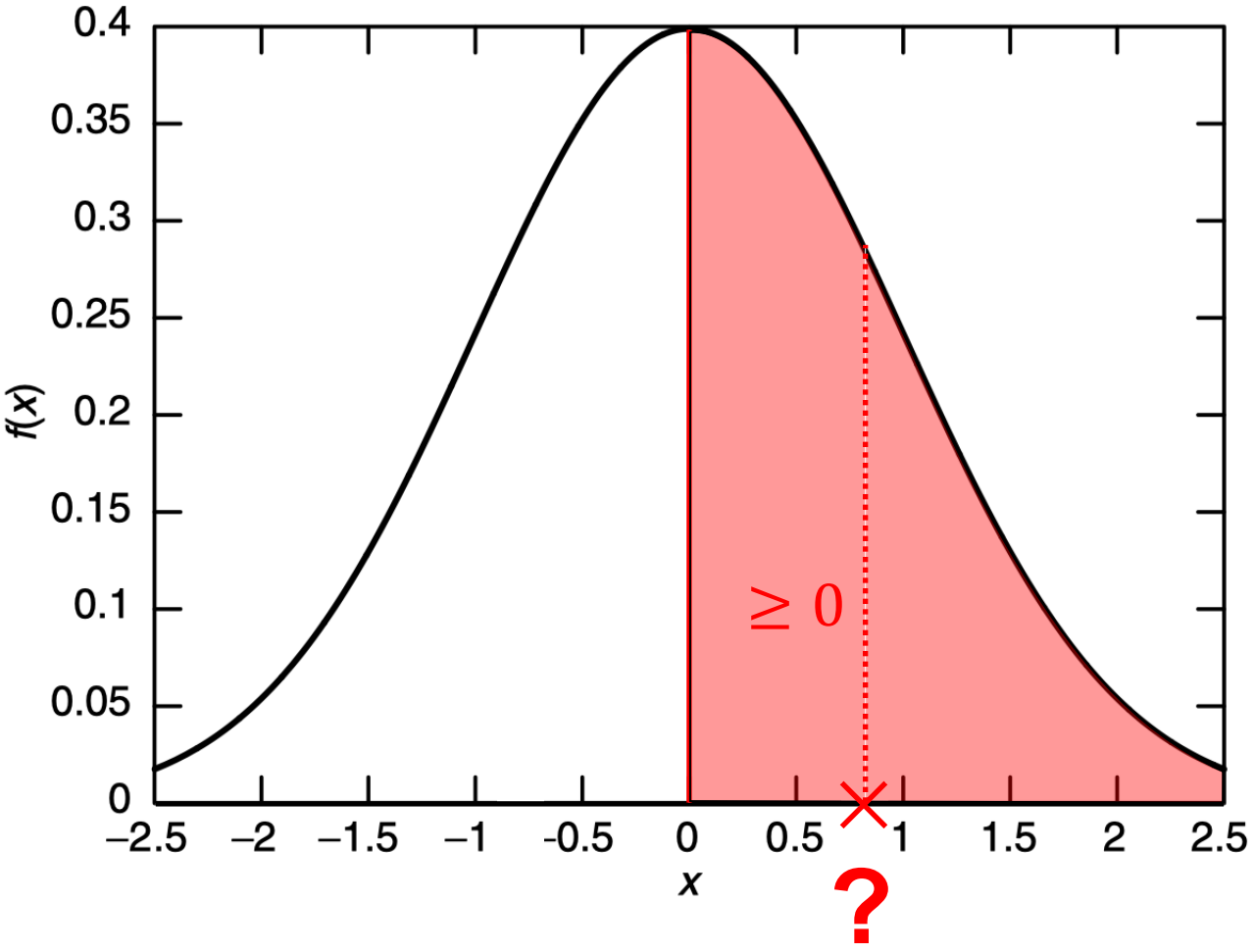
Assume you have $R = 1$ bit (2 values). What is the best way to quantize a Gaussian distribution

?

Assume we like to minimize the mean of squared errors (MSE)

If we have 2 values. It makes sense to choose ≥ 0 and ≤ 0. But what should be the representatives?

Quantization of a Gaussian



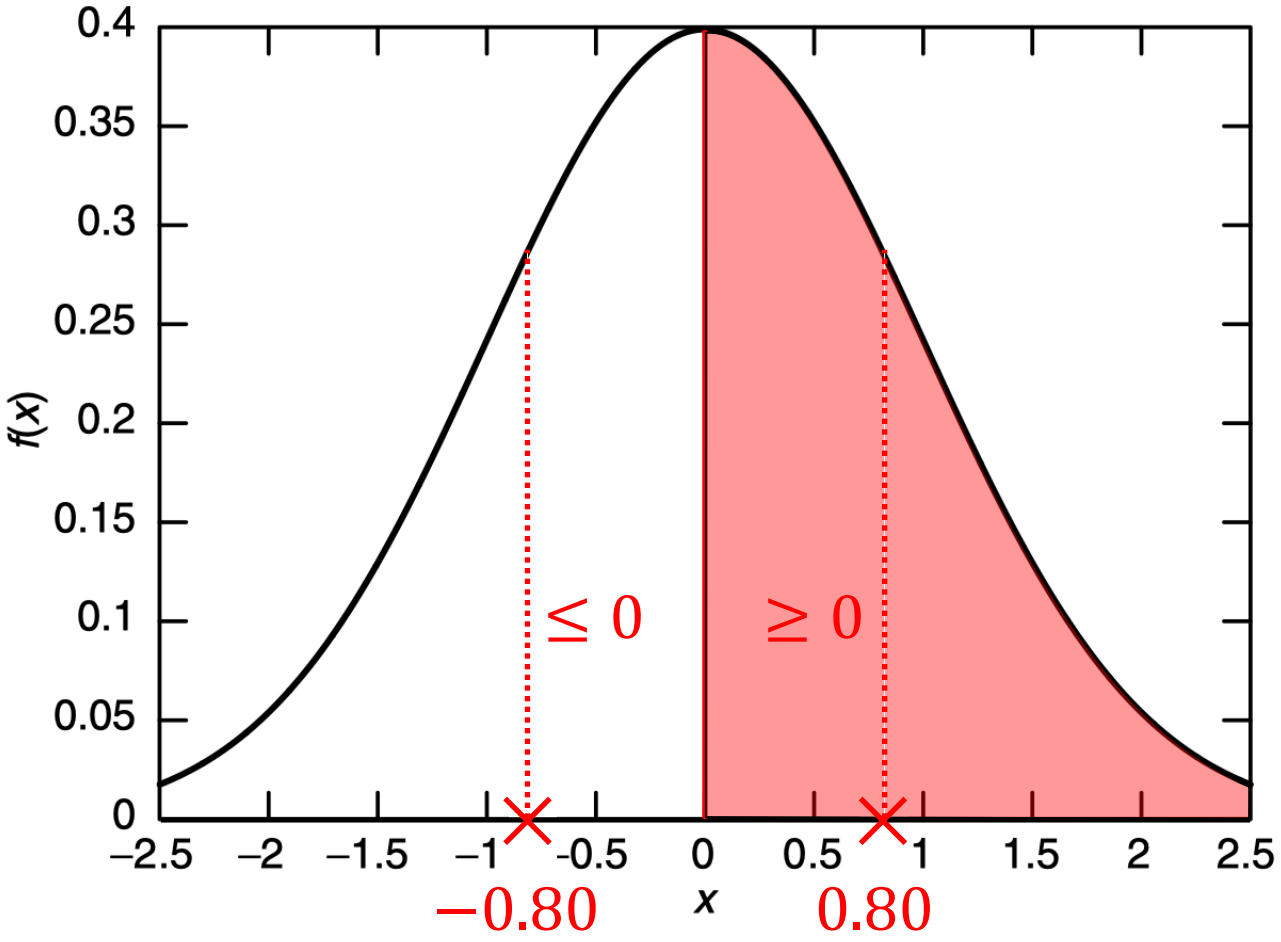
Assume you have $R = 1$ bit (2 values). What is the best way to quantize a Gaussian distribution

Assume we like to minimize the mean of squared errors (MSE)

What should be the representative of the region ≥ 0

If we have 2 values. It makes sense to choose ≥ 0 and ≤ 0. But what should be the representatives?

Quantization of a Gaussian



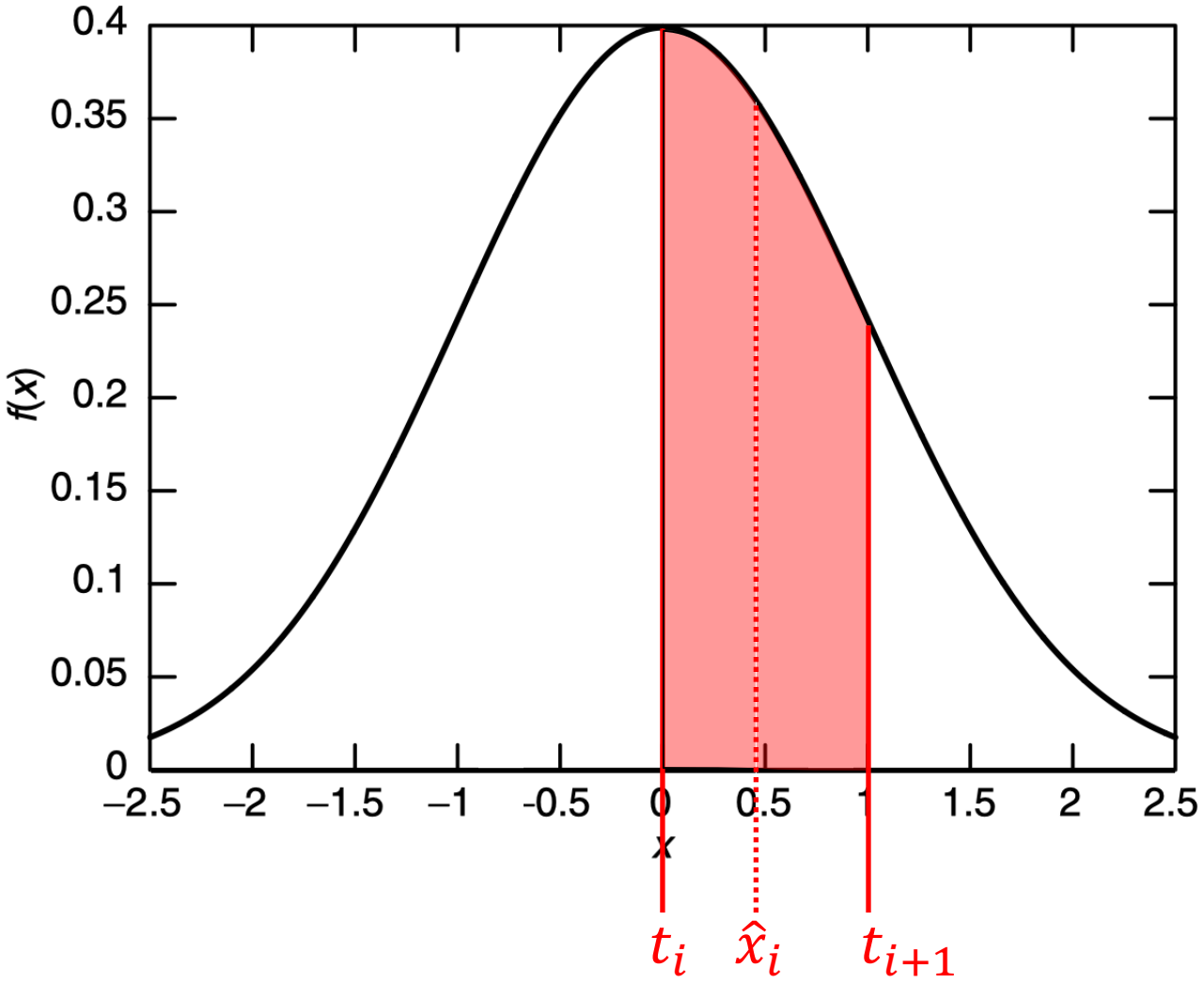
Assume you have $R = 1$ bit (2 values). What is the best way to quantize a Gaussian distribution

Assume we like to minimize the mean of squared errors (MSE)

The (conditional) mean (centroid) of a region minimizes the MSE!

If we have 2 values. It makes sense to choose ≥ 0 and ≤ 0. But what should be the representatives?

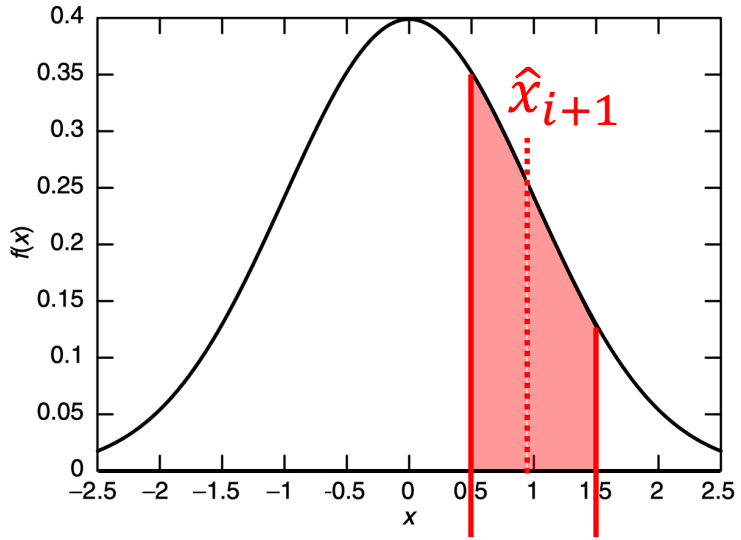
Quantization of a Gaussian



Assume you have $R = 2$ bits (4 values). What is the best way to quantize a Gaussian distribution under MSE?

Now we need to determine 3 boundaries $\{t_i\}$ and 4 reconstruction points $\{\hat{x}_i\}$. But how?

Two properties of optimal boundaries and construction points

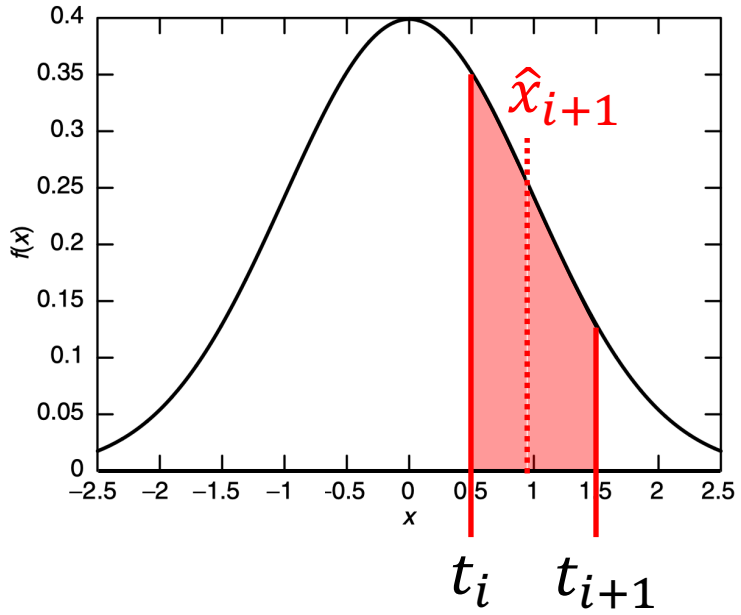


$\{t_i\} \Rightarrow \{\hat{x}_{i+1}\}$

Given two thresholds t_i, t_{i+1} marking the boundaries of a region. What is the best representative $\hat{x}_{i+1}$ of the region?

?

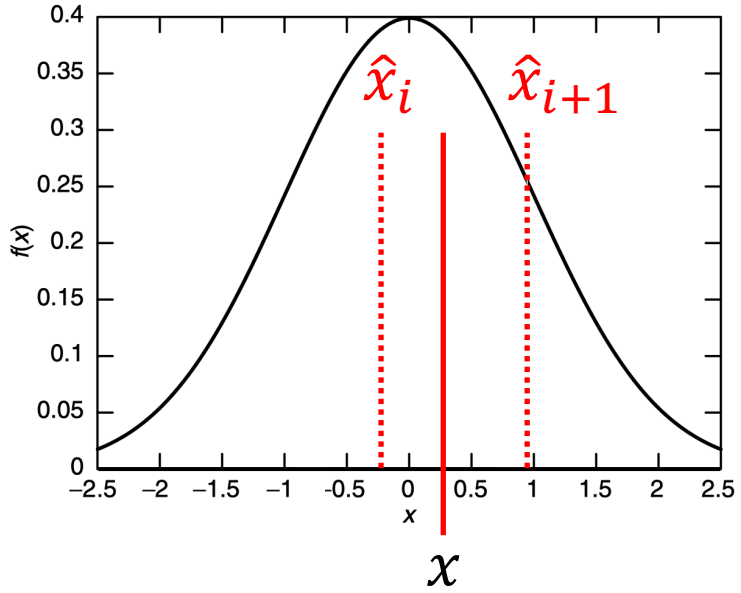
Two properties of optimal boundaries and construction points



$\{t_i\} \Rightarrow \{\hat{x}_{i+1}\}$

Given two thresholds t_i, t_{i+1} marking the boundaries of a region. What is the best representative $\hat{x}_{i+1}$ of the region?

The conditional means (conditioned on the region = centroids) minimize the MSE and should thus be the reconstruction points.

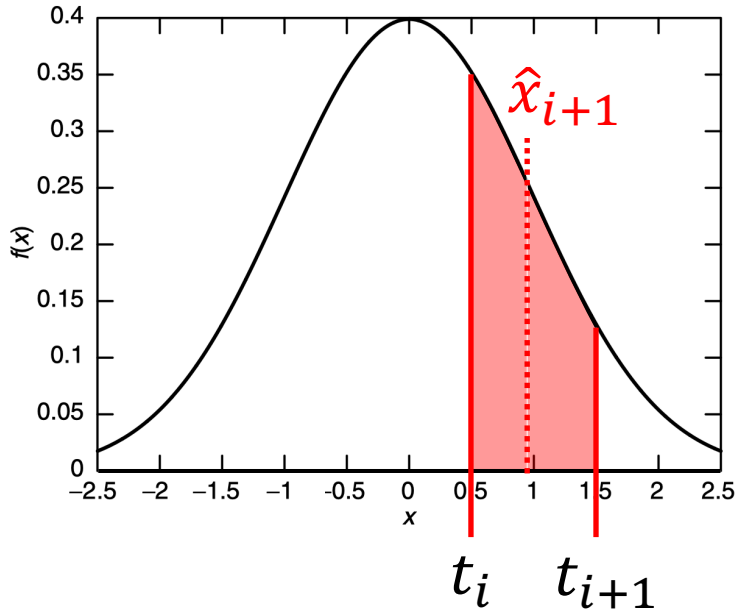


$\{\hat{x}_i\} \Rightarrow \{t_i\}$

Given a set of representative values $\{\hat{x}_{i+1}\}$, which representative should we choose for any given x?

?

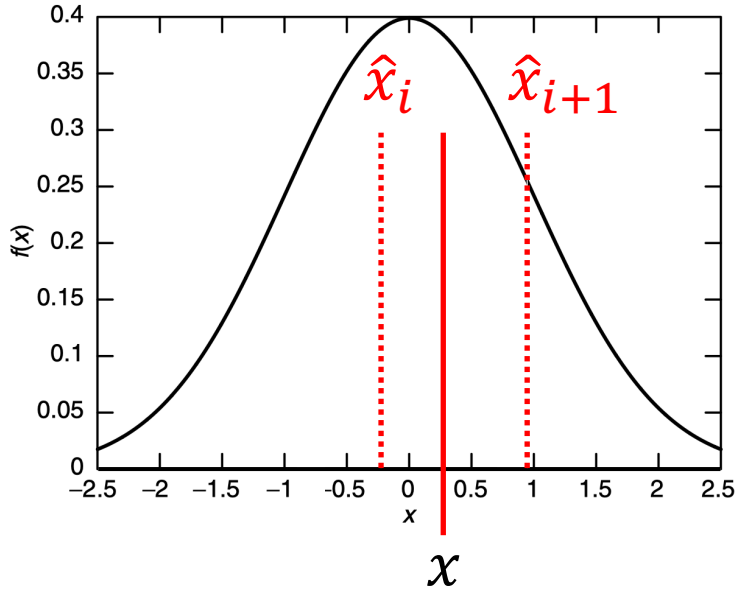
Two properties of optimal boundaries and construction points



$\{t_i\} \Rightarrow \{\hat{x}_{i+1}\}$

Given two thresholds t_i, t_{i+1} marking the boundaries of a region. What is the best representative $\hat{x}_{i+1}$ of the region?

The conditional means (conditioned on the region = centroids) minimize the MSE and should thus be the reconstruction points.

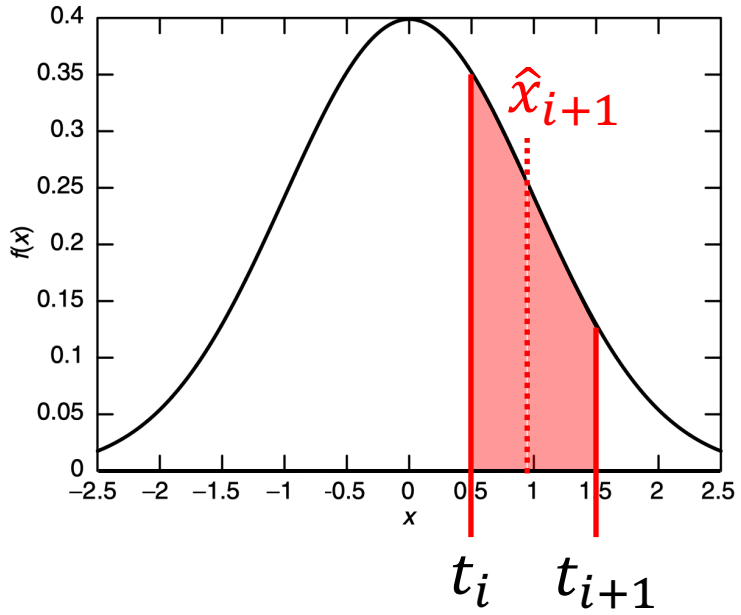


$\{\hat{x}_i\} \Rightarrow \{t_i\}$

Given a set of representative values $\{\hat{x}_{i+1}\}$, which representative should we choose for any given x?

Distortion (MSE) is minimized by assigning values to their closest points. Thus a Voronoi partition gives use the optimal thresholds.

Two properties of optimal boundaries and construction points

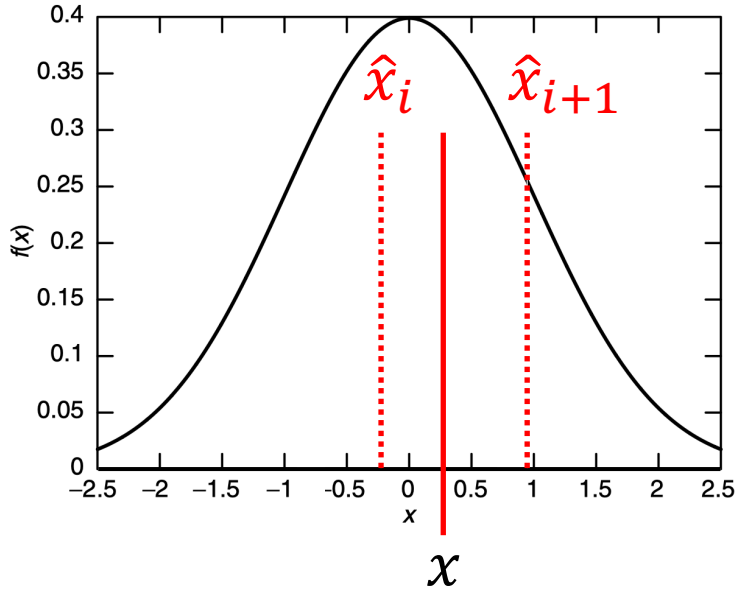


$\{t_i\} \Rightarrow \{\hat{x}_{i+1}\}$

Given two thresholds t_i, t_{i+1} marking the boundaries of a region. What is the best representative $\hat{x}_{i+1}$ of the region?

The conditional means (conditioned on the region = centroids) minimize the MSE and should thus be the reconstruction points.

corresponds to the M-step in EM algorithm: optimize the model parameters (the codewords) given the current assignments



$\{\hat{x}_i\} \Rightarrow \{t_i\}$

Given a set of representative values $\{\hat{x}_{i+1}\}$, which representative should we choose for any given x?

Distortion (MSE) is minimized by assigning values to their closest points. Thus a Voronoi partition gives use the optimal thresholds.

corresponds to E-step in EM: for each input, compute an assignment to a cluster (here it is a deterministic version of computing expected cluster responsibilities)

Lloyd-Max scalar quantizer

Problem: For a signal x with given PDF $f_X(x)$ find a quantizer with m representative levels (or "codes" that minimizes

$$d = MSE = \mathbb{E}[(X - \hat{X})^2]$$

Lloyd-Max quantizer

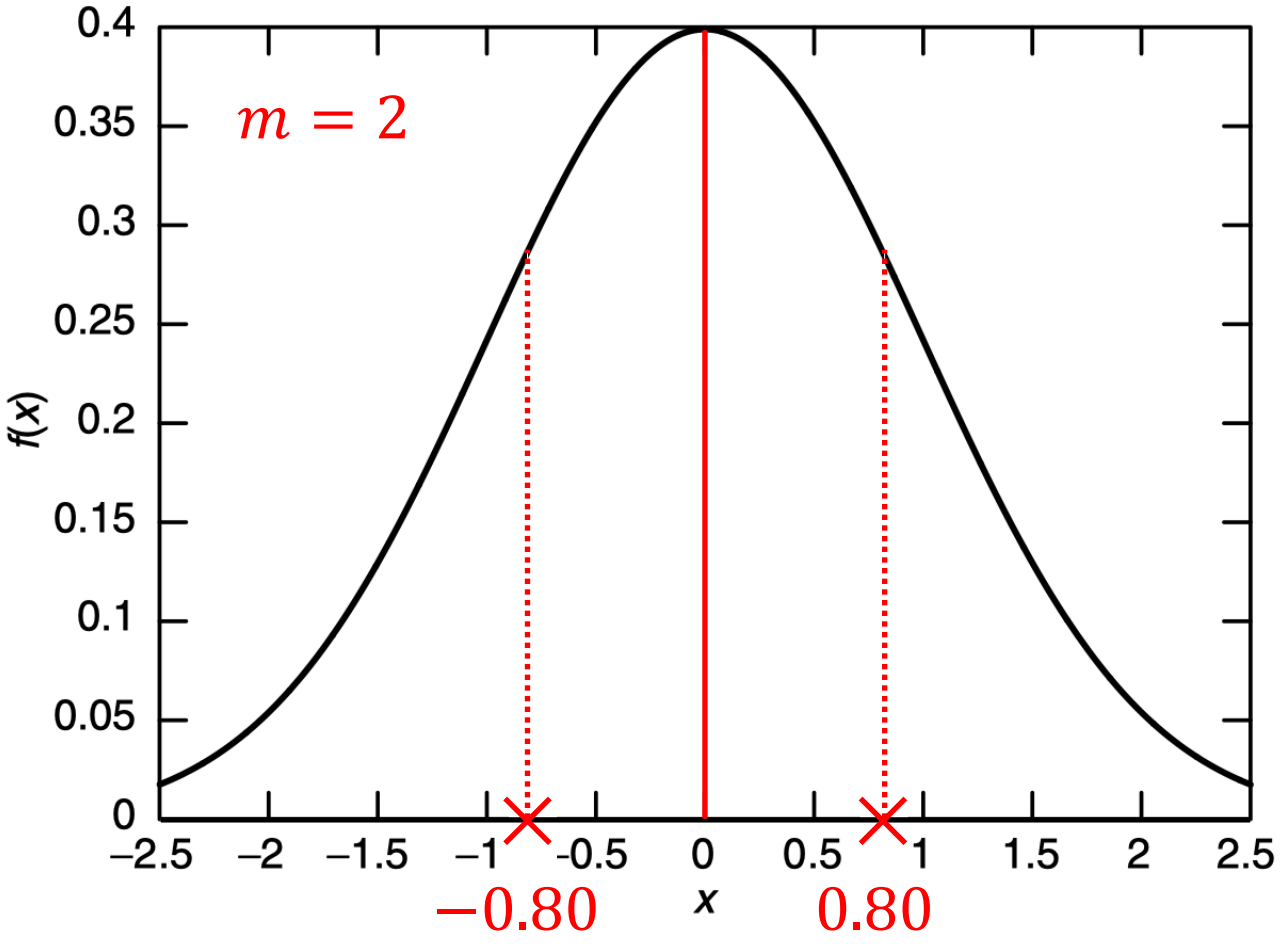
Input: initial vector $\hat{\mathbf{x}}$ of m representative levels
Repeat {

- Create $m-1$ decision thresholds $\mathbf{t}$ exactly half-way between representative levels
- Create m representative levels $\hat{\mathbf{x}}$ as the centroids of PDF between two successive decision thresholds

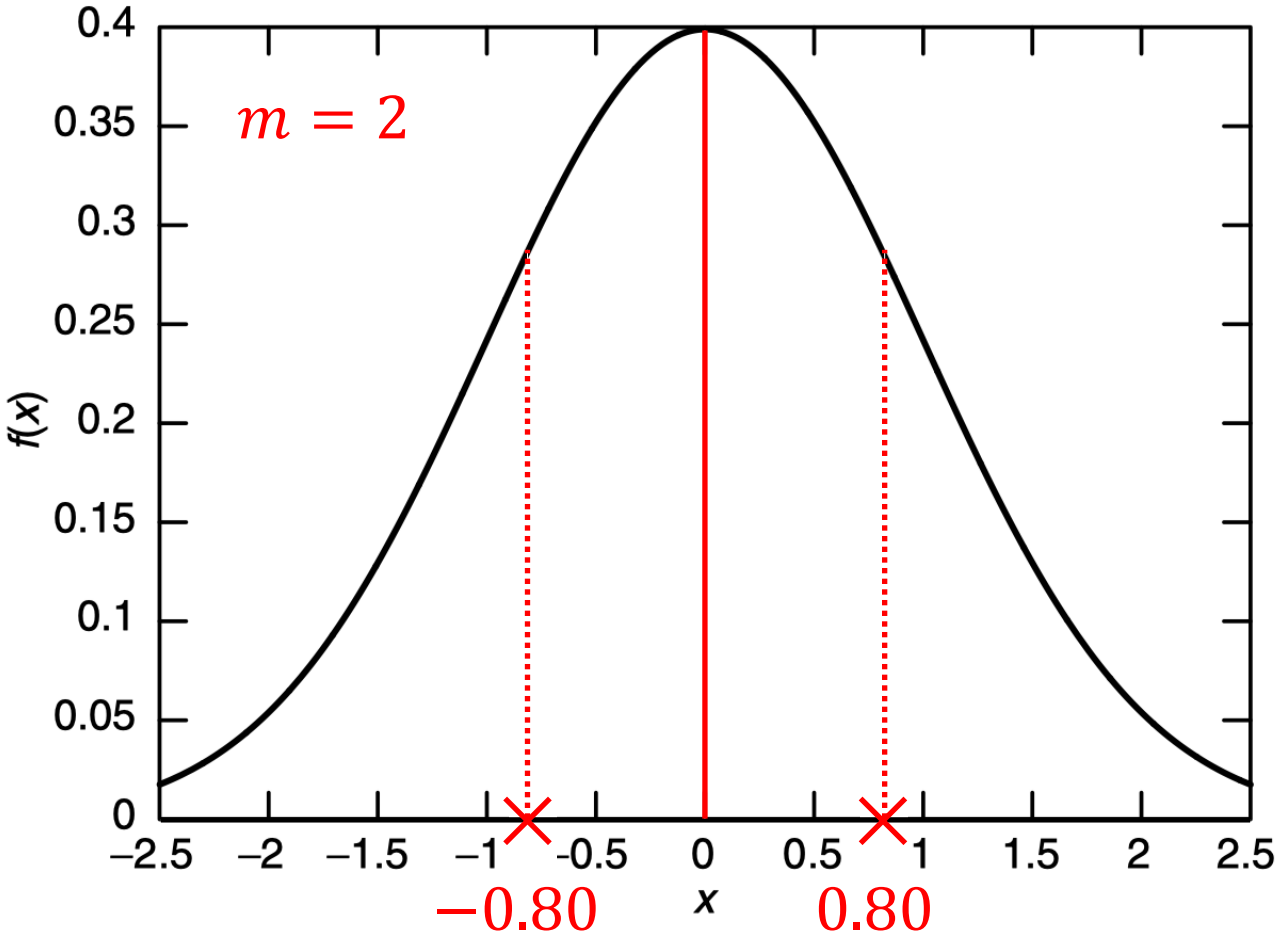
until (likely) convergence}

$$t_i = \frac{\hat{x}_{i-1} + \hat{x}_i}{2}, \; i = 1, \dots, m-1$$

$$\hat{x}_i = \frac{\int_{t_i}^{t_{i+1}} x \cdot f_X(x) dx}{\int_{t_i}^{t_{i+1}} f_X(x) dx}, \; i = 0, \dots, m-1$$

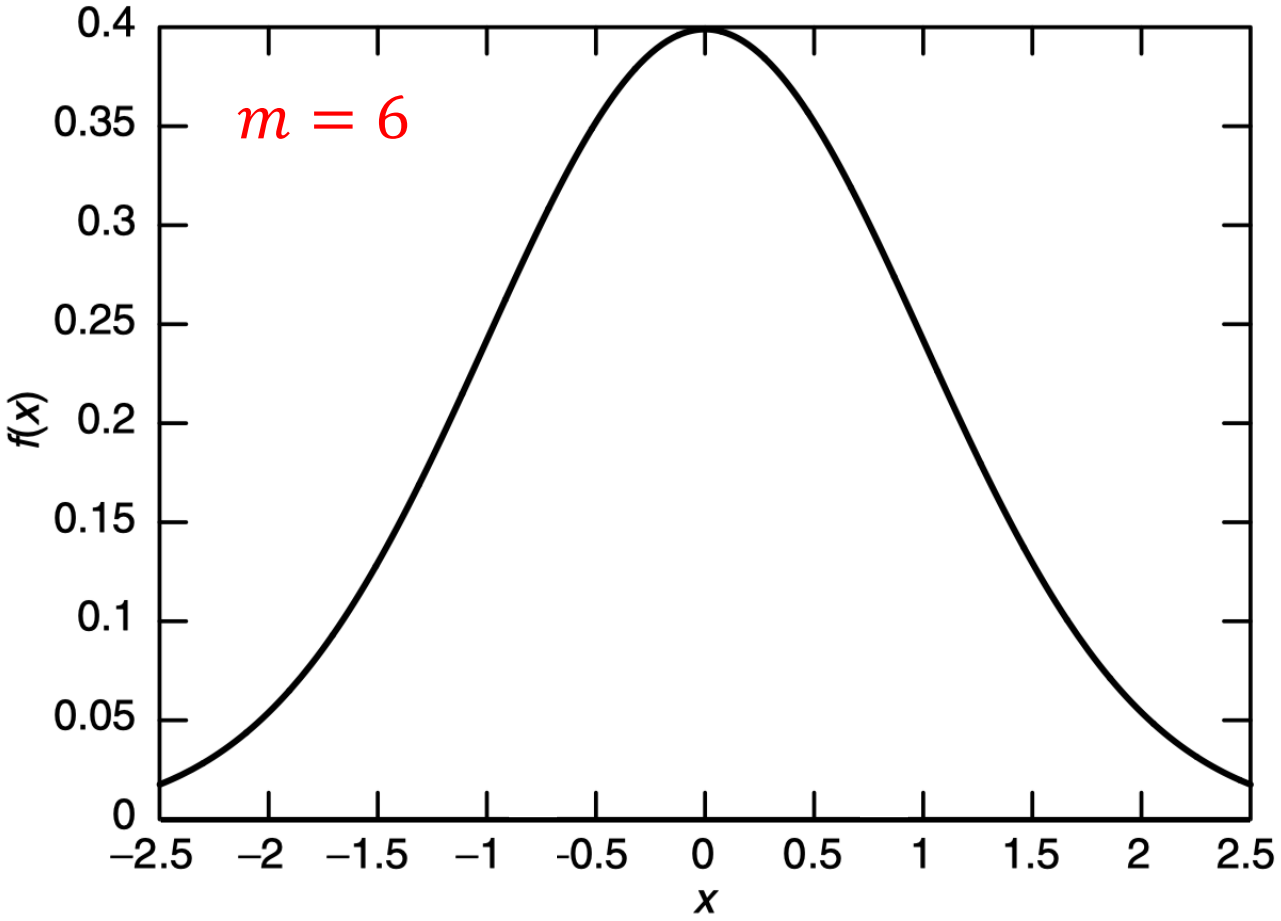



```
 0: representatives: [1 2], d: 1.9414
 1: representatives: [-0.1388  1.9387], d: 0.6019
 2: representatives: [-0.3261  1.4456], d: 0.4686
 3: representatives: [-0.479   1.1851], d: 0.4073
 4: representatives: [-0.5875  1.0354], d: 0.3814
 5: representatives: [-0.661   0.9457], d: 0.3707
 6: representatives: [-0.7095  0.8907], d: 0.3664
 7: representatives: [-0.7411  0.8564], d: 0.3646
 8: representatives: [-0.7616  0.8349], d: 0.3639
 9: representatives: [-0.7747  0.8214], d: 0.3636
10: representatives: [-0.7831  0.8128], d: 0.3635
11: representatives: [-0.7884  0.8074], d: 0.3634
12: representatives: [-0.7919  0.8039], d: 0.3634
13: representatives: [-0.7941  0.8017], d: 0.3634
14: representatives: [-0.7954  0.8003], d: 0.3634
15: representatives: [-0.7963  0.7994], d: 0.3634
16: representatives: [-0.7969  0.7989], d: 0.3634
17: representatives: [-0.7973  0.7985], d: 0.3634
18: representatives: [-0.7975  0.7983], d: 0.3634
19: representatives: [-0.7976  0.7981], d: 0.3634
20: representatives: [-0.7977  0.798 ], d: 0.3634
21: representatives: [-0.7978  0.798 ], d: 0.3634
22: representatives: [-0.7978  0.798 ], d: 0.3634
23: representatives: [-0.7978  0.7979], d: 0.3634
24: representatives: [-0.7979  0.7979], d: 0.3634
25: representatives: [-0.7979  0.7979], d: 0.3634
26: representatives: [-0.7979  0.7979], d: 0.3634
27: representatives: [-0.7979  0.7979], d: 0.3634
28: representatives: [-0.7979  0.7979], d: 0.3634
29: representatives: [-0.7979  0.7979], d: 0.3634
```




```
 0: representatives: [2 3], d: 4.9960
 1: representatives: [-0.0176  2.8227], d: 0.7932
 2: representatives: [-0.1622  1.8562], d: 0.5800
 3: representatives: [-0.3477  1.404 ], d: 0.4579
 4: representatives: [-0.4948  1.1617], d: 0.4027
 5: representatives: [-0.5984  1.0216], d: 0.3795
 6: representatives: [-0.6682  0.9373], d: 0.3699
 7: representatives: [-0.7143  0.8854], d: 0.3660
 8: representatives: [-0.7442  0.8532], d: 0.3645
 9: representatives: [-0.7635  0.8329], d: 0.3638
10: representatives: [-0.7759  0.8201], d: 0.3636
11: representatives: [-0.7839  0.812 ], d: 0.3635
12: representatives: [-0.789   0.8069], d: 0.3634
13: representatives: [-0.7922  0.8036], d: 0.3634
14: representatives: [-0.7943  0.8015], d: 0.3634
15: representatives: [-0.7956  0.8002], d: 0.3634
16: representatives: [-0.7964  0.7994], d: 0.3634
17: representatives: [-0.7969  0.7988], d: 0.3634
18: representatives: [-0.7973  0.7985], d: 0.3634
19: representatives: [-0.7975  0.7983], d: 0.3634
20: representatives: [-0.7976  0.7981], d: 0.3634
21: representatives: [-0.7977  0.798 ], d: 0.3634
22: representatives: [-0.7978  0.798 ], d: 0.3634
23: representatives: [-0.7978  0.7979], d: 0.3634
24: representatives: [-0.7978  0.7979], d: 0.3634
25: representatives: [-0.7979  0.7979], d: 0.3634
26: representatives: [-0.7979  0.7979], d: 0.3634
27: representatives: [-0.7979  0.7979], d: 0.3634
28: representatives: [-0.7979  0.7979], d: 0.3634
29: representatives: [-0.7979  0.7979], d: 0.3634
```

Python file 232: https://github.com/northeastern-datalab/cs7840-activities/tree/main/notebooks
Gatterbauer. Foundations and Applications of Information Theory: https://northeastern-datalab.github.io/cs7840/fa25/




```
 0: representatives: [-2.5 -1.5 -0.5  0.5  1.5  2.5], d: 0.0849
 1: representatives: [-2.3732 -1.3832 -0.4599  0.4599  1.3832  2.3732], d: 0.0748
 2: representatives: [-2.2658 -1.2991 -0.4292  0.4292  1.2991  2.2658], d: 0.0686
 3: representatives: [-2.182  -1.2349 -0.4059  0.4059  1.2349  2.182 ], d: 0.0647
 4: representatives: [-2.1177 -1.1851 -0.3878  0.3878  1.1851  2.1177], d: 0.0622
 5: representatives: [-2.0683 -1.1461 -0.3734  0.3734  1.1461  2.0683], d: 0.0607
 6: representatives: [-2.0303 -1.1154 -0.362   0.362   1.1154  2.0303], d: 0.0597
 7: representatives: [-2.0009 -1.0912 -0.3529  0.3529  1.0912  2.0009], d: 0.0590
 8: representatives: [-1.9779 -1.0722 -0.3456  0.3456  1.0722  1.9779], d: 0.0586
 9: representatives: [-1.96   -1.0571 -0.3399  0.3399  1.0571  1.96  ], d: 0.0584
10: representatives: [-1.946  -1.0452 -0.3353  0.3353  1.0452  1.946 ], d: 0.0582
11: representatives: [-1.9349 -1.0358 -0.3317  0.3317  1.0358  1.9349], d: 0.0581
12: representatives: [-1.9263 -1.0284 -0.3288  0.3288  1.0284  1.9263], d: 0.0581
13: representatives: [-1.9194 -1.0225 -0.3265  0.3265  1.0225  1.9194], d: 0.0580
14: representatives: [-1.914  -1.0179 -0.3247  0.3247  1.0179  1.914 ], d: 0.0580
15: representatives: [-1.9098 -1.0142 -0.3232  0.3232  1.0142  1.9098], d: 0.0580
16: representatives: [-1.9064 -1.0112 -0.3221  0.3221  1.0112  1.9064], d: 0.0580
17: representatives: [-1.9037 -1.0089 -0.3212  0.3212  1.0089  1.9037], d: 0.0580
18: representatives: [-1.9016 -1.0071 -0.3205  0.3205  1.0071  1.9016], d: 0.0580
19: representatives: [-1.8999 -1.0056 -0.3199  0.3199  1.0056  1.8999], d: 0.0580
20: representatives: [-1.8986 -1.0045 -0.3194  0.3194  1.0045  1.8986], d: 0.0580
21: representatives: [-1.8976 -1.0036 -0.3191  0.3191  1.0036  1.8976], d: 0.0580
22: representatives: [-1.8968 -1.0029 -0.3188  0.3188  1.0029  1.8968], d: 0.0580
23: representatives: [-1.8961 -1.0023 -0.3186  0.3186  1.0023  1.8961], d: 0.0580
24: representatives: [-1.8956 -1.0018 -0.3184  0.3184  1.0018  1.8956], d: 0.0580
25: representatives: [-1.8952 -1.0015 -0.3183  0.3183  1.0015  1.8952], d: 0.0580
26: representatives: [-1.8948 -1.0012 -0.3181  0.3181  1.0012  1.8948], d: 0.0580
27: representatives: [-1.8946 -1.001  -0.3181  0.3181  1.001   1.8946], d: 0.0580
28: representatives: [-1.8944 -1.0008 -0.318   0.318   1.0008  1.8944], d: 0.0580
29: representatives: [-1.8942 -1.0006 -0.3179  0.3179  1.0006  1.8942], d: 0.0580
30: representatives: [-1.8941 -1.0005 -0.3179  0.3179  1.0005  1.8941], d: 0.0580
31: representatives: [-1.894  -1.0004 -0.3178  0.3178  1.0004  1.894 ], d: 0.0580
32: representatives: [-1.8939 -1.0004 -0.3178  0.3178  1.0004  1.8939], d: 0.0580
33: representatives: [-1.8938 -1.0003 -0.3178  0.3178  1.0003  1.8938], d: 0.0580
34: representatives: [-1.8938 -1.0003 -0.3178  0.3178  1.0003  1.8938], d: 0.0580
35: representatives: [-1.8937 -1.0002 -0.3178  0.3178  1.0002  1.8937], d: 0.0580
36: representatives: [-1.8937 -1.0002 -0.3178  0.3178  1.0002  1.8937], d: 0.0580
37: representatives: [-1.8937 -1.0002 -0.3177  0.3177  1.0002  1.8937], d: 0.0580
38: representatives: [-1.8937 -1.0002 -0.3177  0.3177  1.0002  1.8937], d: 0.0580
39: representatives: [-1.8937 -1.0002 -0.3177  0.3177  1.0002  1.8937], d: 0.0580
40: representatives: [-1.8936 -1.0001 -0.3177  0.3177  1.0001  1.8936], d: 0.0580
41: representatives: [-1.8936 -1.0001 -0.3177  0.3177  1.0001  1.8936], d: 0.0580
42: representatives: [-1.8936 -1.0001 -0.3177  0.3177  1.0001  1.8936], d: 0.0580
43: representatives: [-1.8936 -1.0001 -0.3177  0.3177  1.0001  1.8936], d: 0.0580
44: representatives: [-1.8936 -1.0001 -0.3177  0.3177  1.0001  1.8936], d: 0.0580
45: representatives: [-1.8936 -1.0001 -0.3177  0.3177  1.0001  1.8936], d: 0.0580
46: representatives: [-1.8936 -1.0001 -0.3177  0.3177  1.0001  1.8936], d: 0.0580
47: representatives: [-1.8936 -1.0001 -0.3177  0.3177  1.0001  1.8936], d: 0.0580
48: representatives: [-1.8936 -1.0001 -0.3177  0.3177  1.0001  1.8936], d: 0.0580
49: representatives: [-1.8936 -1.0001 -0.3177  0.3177  1.0001  1.8936], d: 0.0580
```

Python file 232: https://github.com/northeastern-datalab/cs7840-activities/tree/main/notebooks

Example: Lloyd-Max quantizers for Gaussian PDF

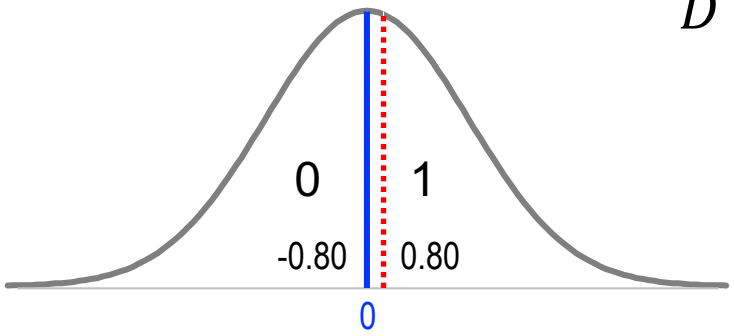
index

quantized value, representation

Data point x	Code name $f(x)$	Recon-struction $g(f(x))$	Reconstruction error d $\left(g\left(f(x)\right)-x\right)^2$
0.1	1	0.80	0.490=(0.8-0.1)^2

Expected Distortion $D = \mathbb{E}[d]$

0.3634



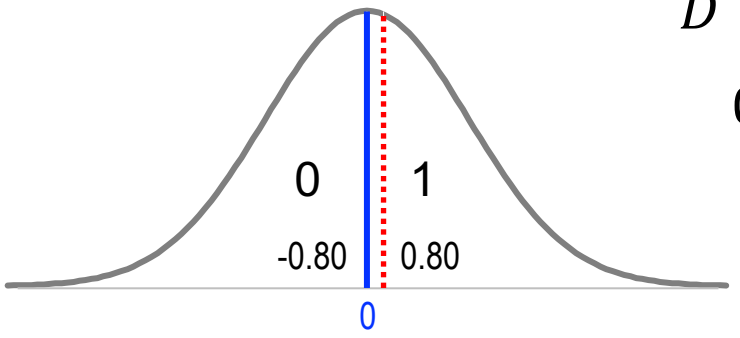
Example: Lloyd-Max quantizers for Gaussian PDF

index

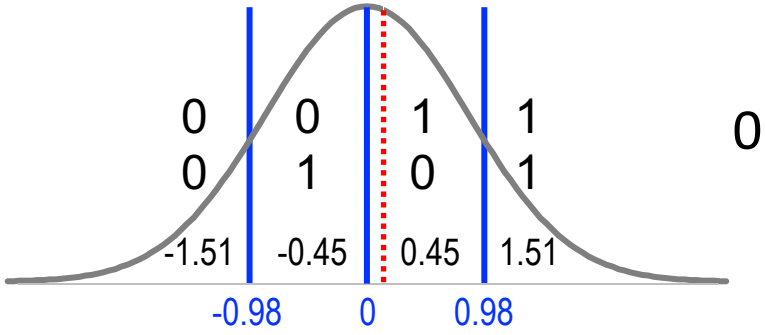
quantized value, representation

Data point x	Code length *bits*	Code name $f(x)$	Recon-struction $g(f(x))$	Reconstruction error d $\left(g(f(x))-x\right)^2$
0.1	1	1	0.80	0.490
	2	10	0.45	0.123
	3	100	0.25	0.023

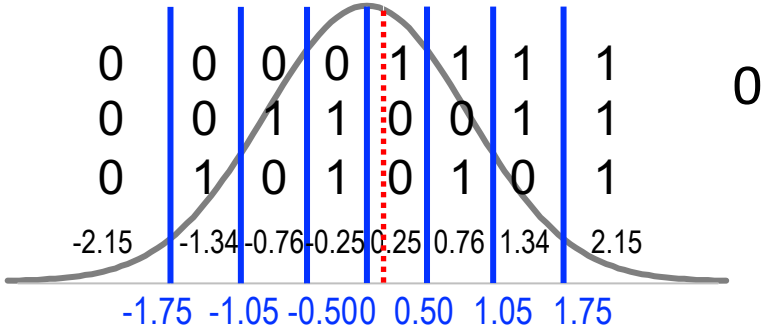
Expected Distortion $D = \mathbb{E}[d]$



0.3634



0.1175



0.0345

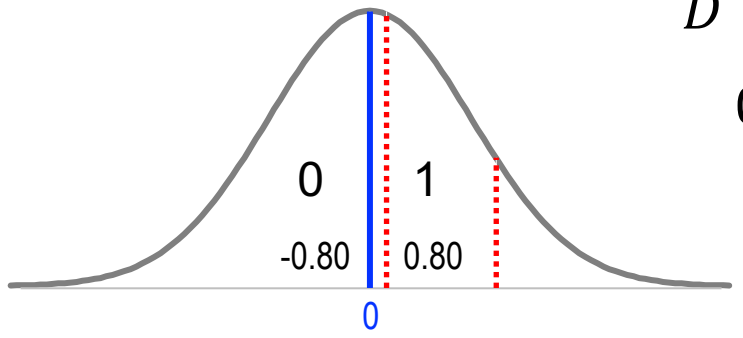
Example: Lloyd-Max quantizers for Gaussian PDF

index

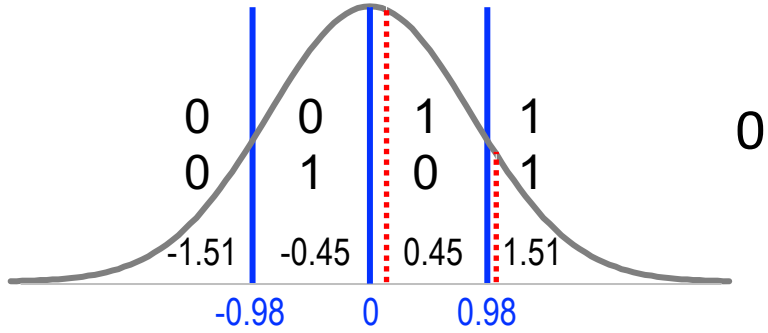
quantized value, representation

Data point x	Code length *bits*	Code name $f(x)$	Recon-struction $g(f(x))$	Reconstruction error d $\left(g(f(x)) - x\right)^2$
0.1	1	1	0.80	0.490
	2	10	0.45	0.123
	3	100	0.25	0.023
1.0	1	?	?	?
	2	?	?	?
	3	?	?	?

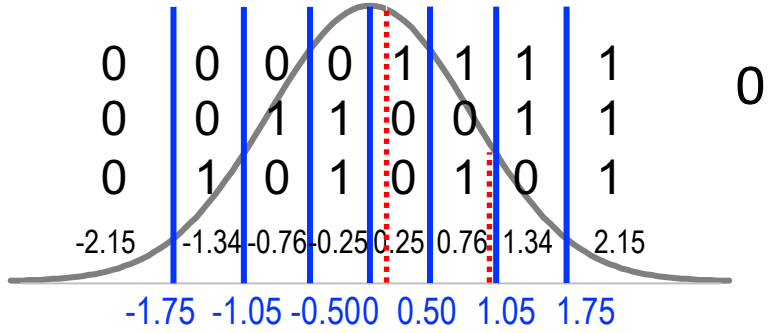
Expected Distortion $D = \mathbb{E}[d]$



0.3634



0.1175



0.0345

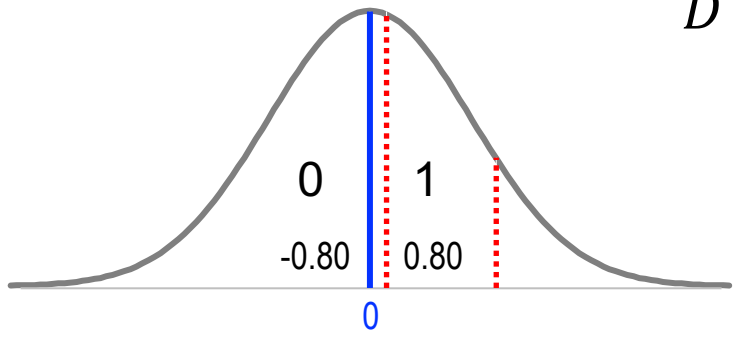
Example: Lloyd-Max quantizers for Gaussian PDF

index

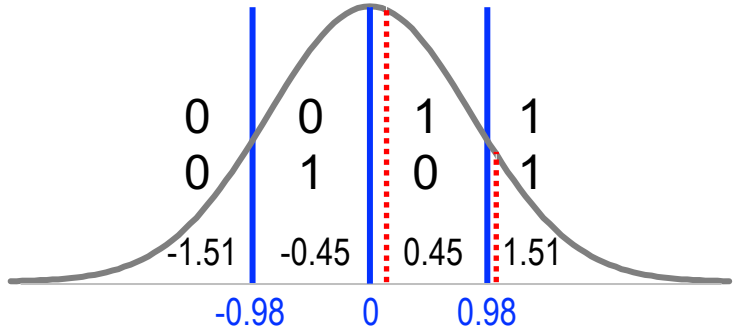
quantized value, representation

Data point x	Code length *bits*	Code name $f(x)$	Recon-struction $g(f(x))$	Reconstruction error d $\left(g\left(f(x)\right)-x\right)^2$
0.1	1	1	0.80	0.490
	2	10	0.45	0.123
	3	100	0.25	0.023
1.0	1	1	0.80	0.040
	2	11	1.51	0.260
	3	101	0.76	0.058

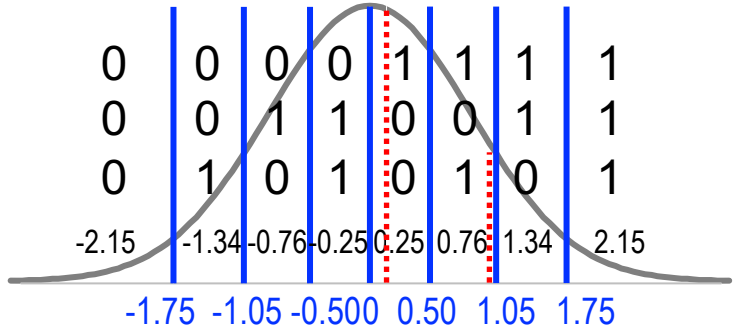
Expected Distortion $D = \mathbb{E}[d]$



0.3634



0.1175



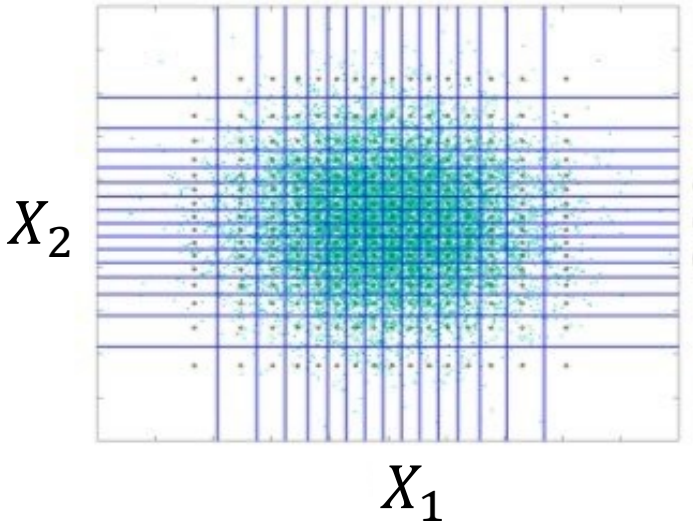
0.0345

Vector quantization:

the geometry of longer block length (higher dimensions): Voronoi tessellations and connection to k-means

The geometry of vector quantization

Independent 4-bit quantization (16 representatives) for $n = 2$ independent Gaussians:



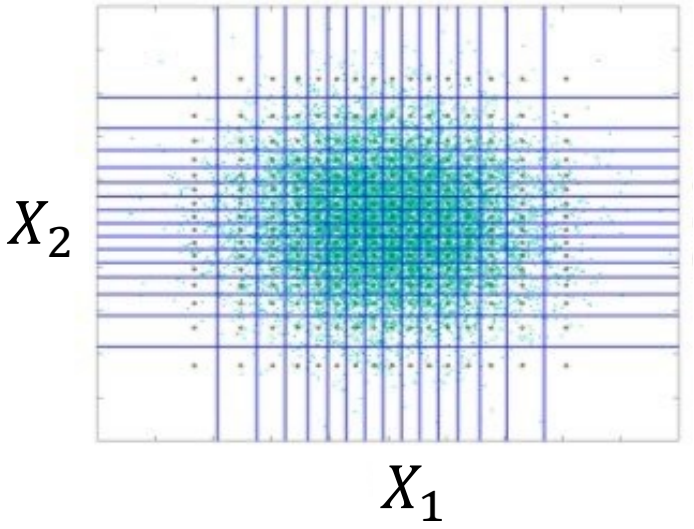
Joint encoding of $n = 2$ independent Gaussians:

?

Figure source: https://ieeexplore.ieee.org/document/7767821/

The geometry of vector quantization

Independent 4-bit quantization (16 representatives) for $n = 2$ independent Gaussians:



Joint encoding of $n = 2$ independent Gaussians: 2D vector quantization, i.e. block length $n = 2$ and 4-bit per sample, or 8-bit (and 256 representatives) for two samples together

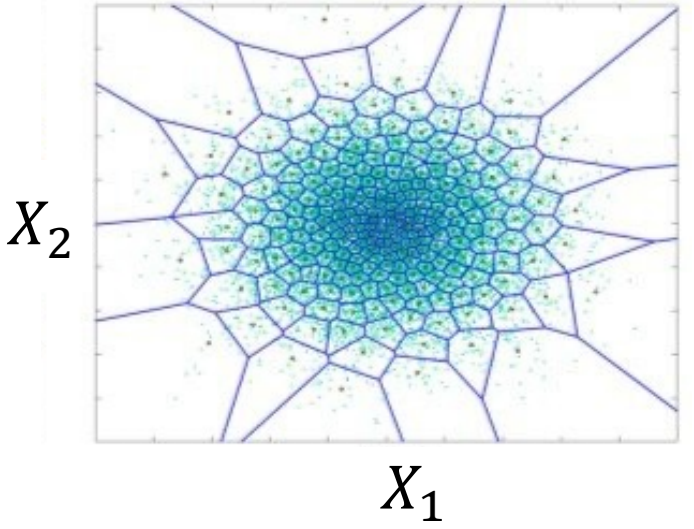


Figure source: https://ieeexplore.ieee.org/document/7767821/
Gatterbauer. Foundations and Applications of Information Theory: https://northeastern-datalab.github.io/cs7840/fa25/

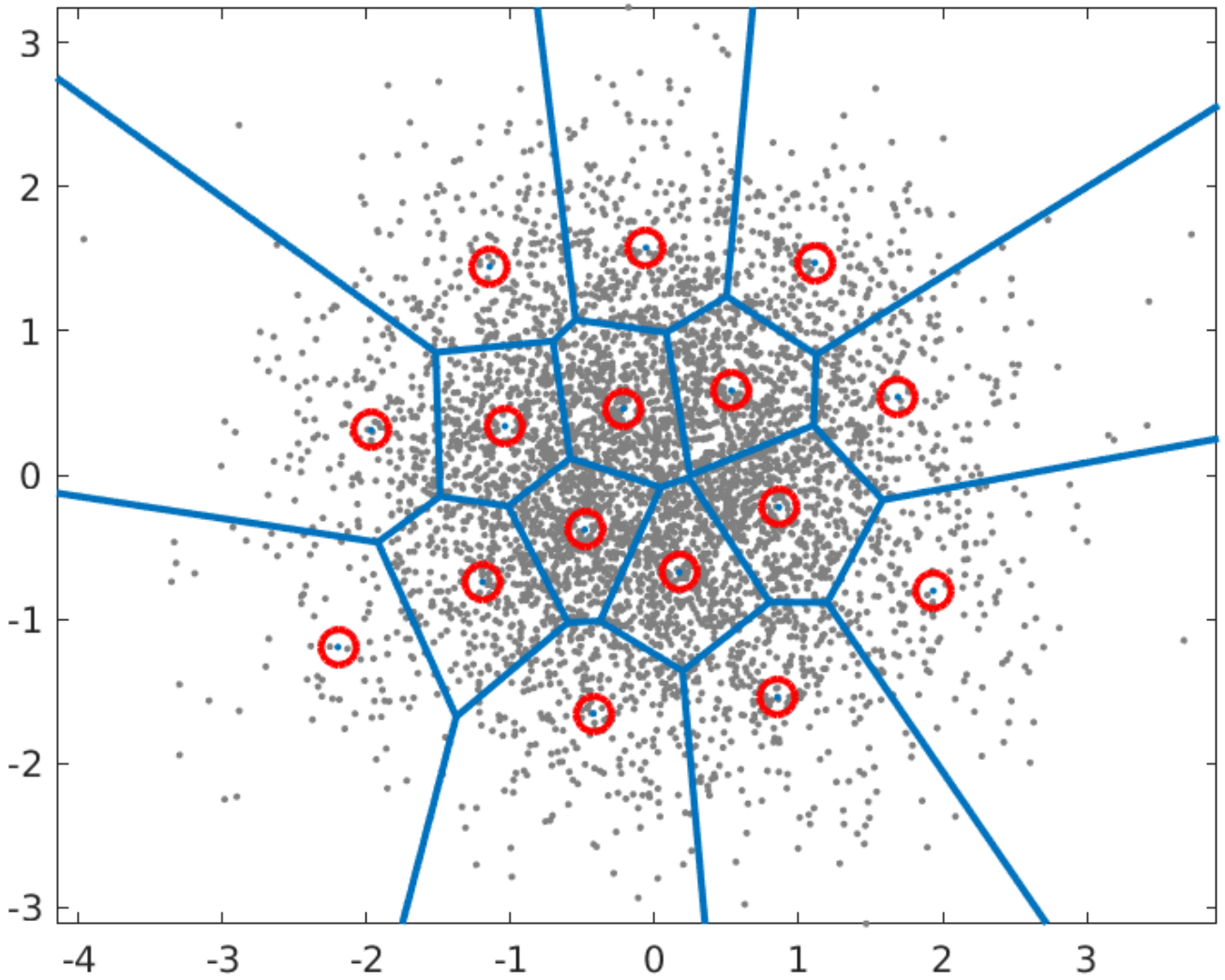


Figure source: https://speechprocessingbook.aalto.fi/Modelling/Vector_quantization_VQ.html
Gatterbauer. Foundations and Applications of Information Theory: https://northeastern-datalab.github.io/cs7840/fa25/

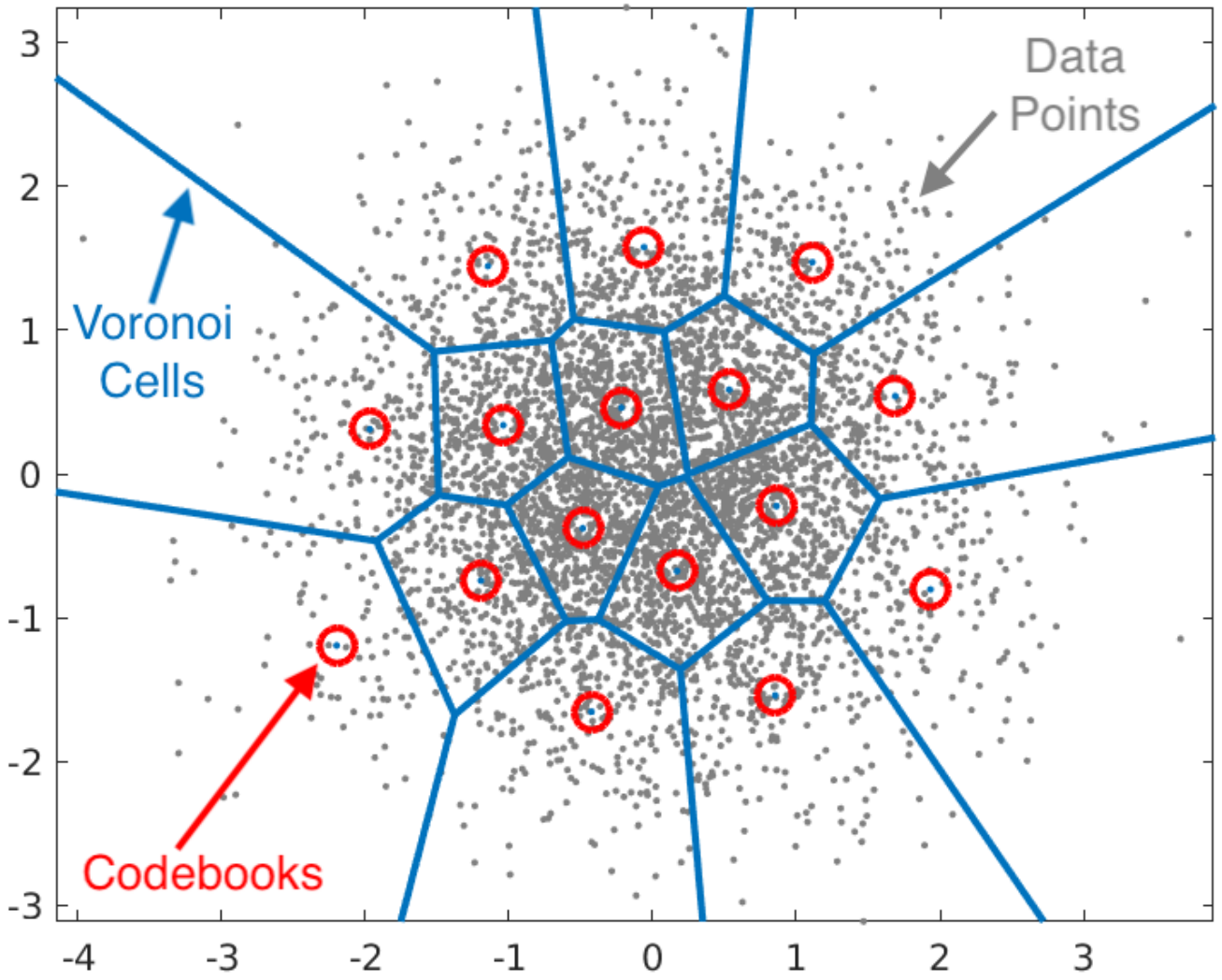


Figure source: https://speechprocessingbook.aalto.fi/Modelling/Vector_quantization_VQ.html

Lloyd's algorithm = k-means

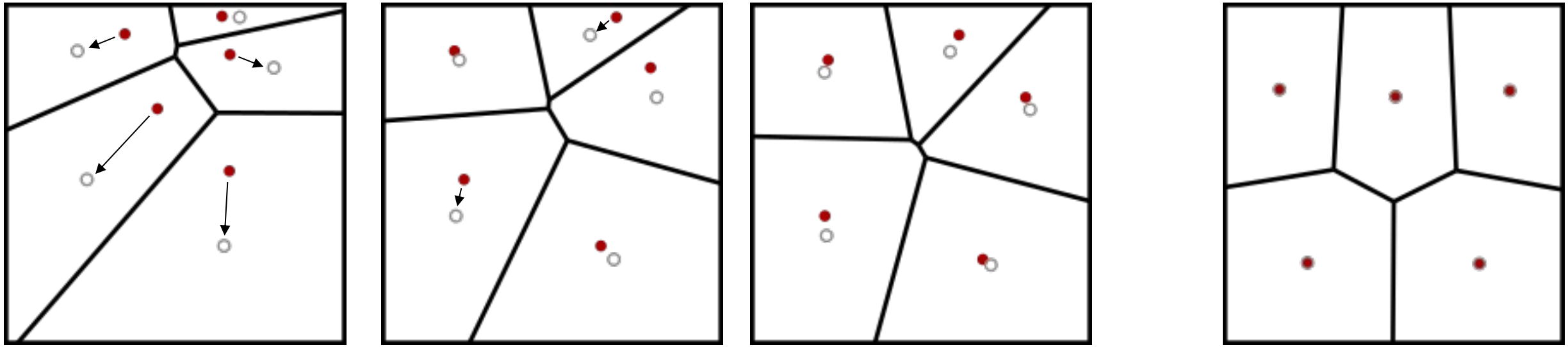
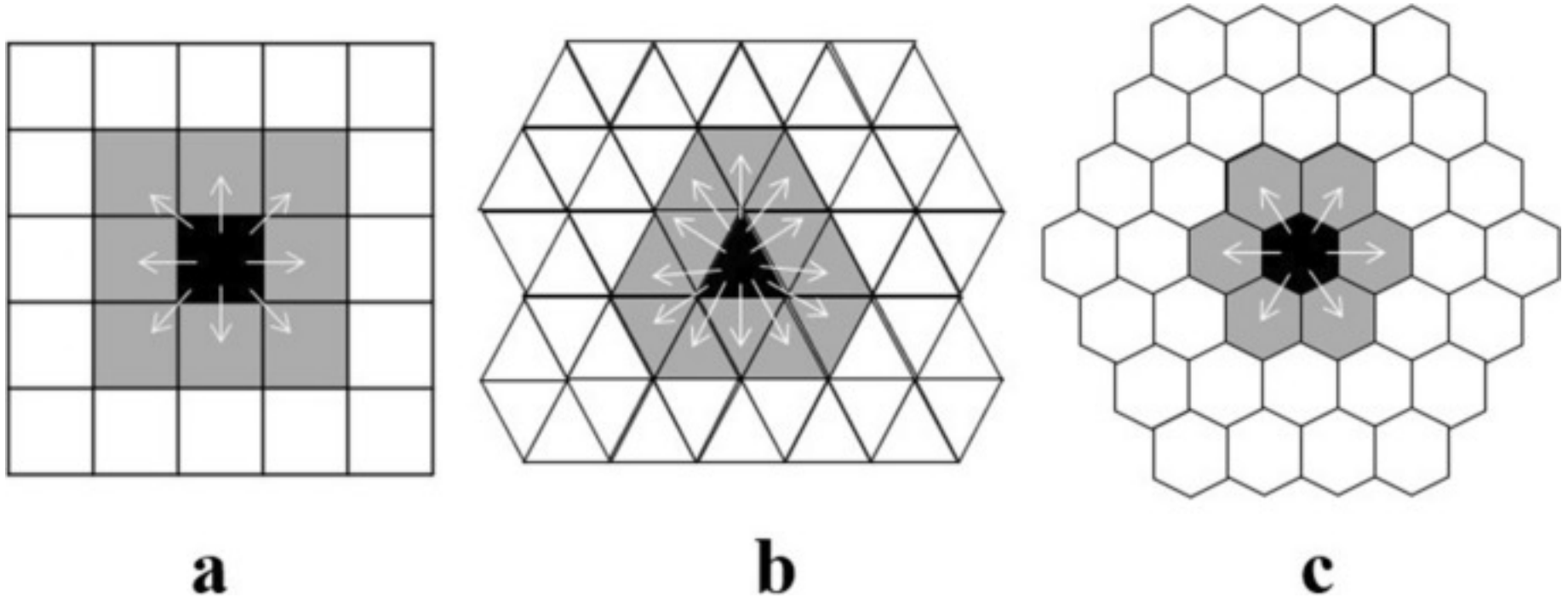


Figure source: https://en.wikipedia.org/wiki/Lloyd's_algorithm
Gatterbauer. Foundations and Applications of Information Theory: https://northeastern-datalab.github.io/cs7840/fa25/

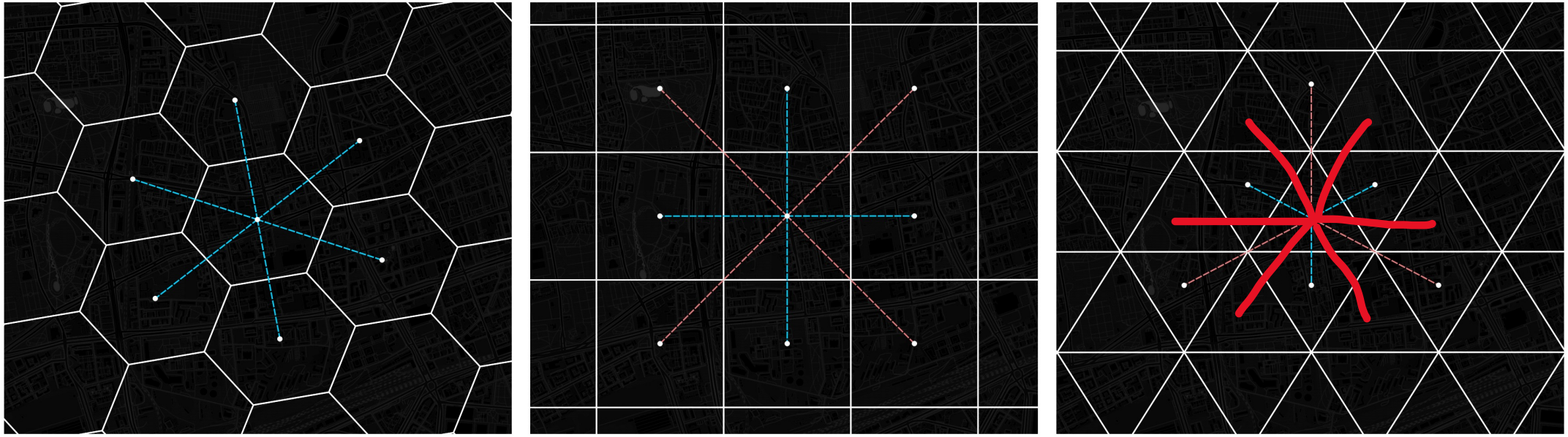
Optimal tessellations



?

Figure source: https://link.springer.com/article/10.1007/s41651-024-00200-5
Gatterbauer. Foundations and Applications of Information Theory: https://northeastern-datalab.github.io/cs7840/fa25/

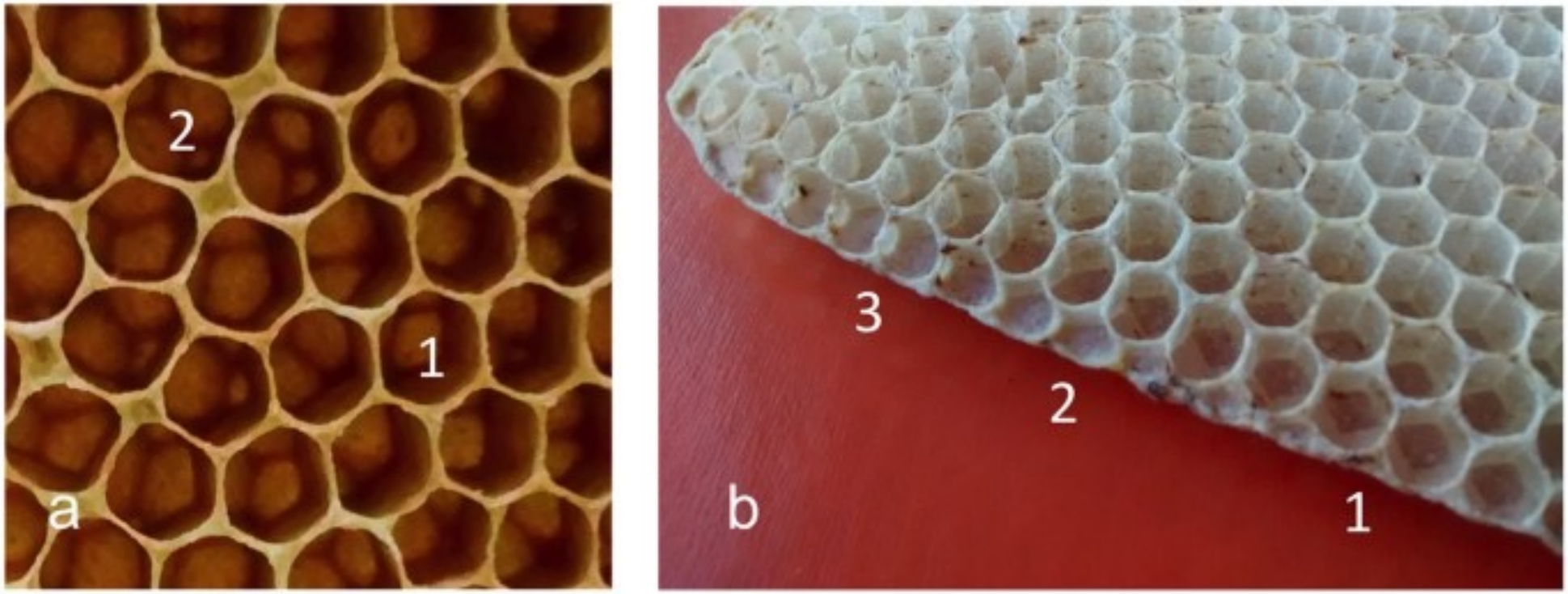
Optimal tessellations



Three types of spatial grids: hexagonal, square, and triangular.
Only the hexagonal grid provides an equal distance between the centers of neighboring cells.
There are at least two different distance categories for other kinds of grids.

Source: https://www.kontur.io/blog/why-we-use-h3/
Gatterbauer. Foundations and Applications of Information Theory: https://northeastern-datalab.github.io/cs7840/fa25/

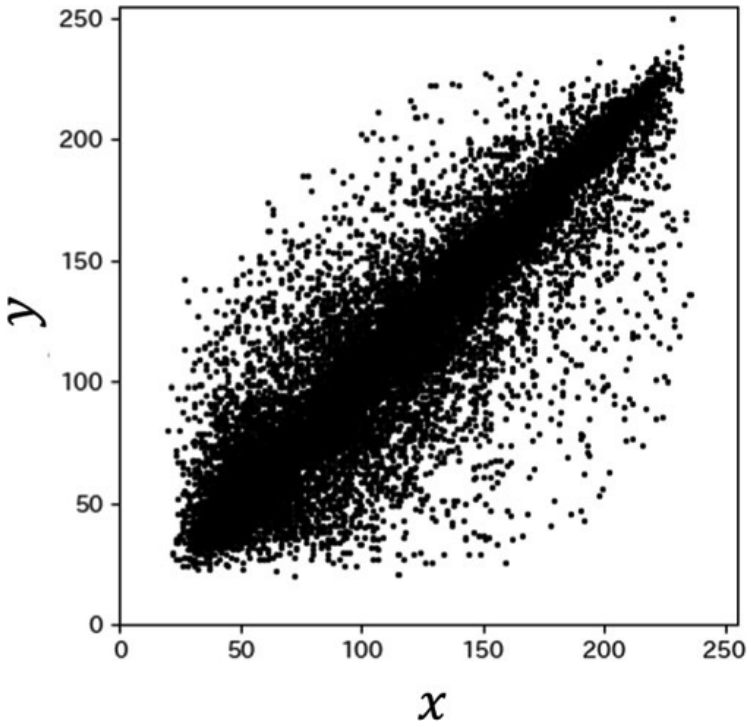
Optimal tessellations



"Early natural philosophers, like Marcus Terentius Varro [37 BC], based on the observation that hexagons possess the highest surface/perimeter ratio, compared to other polygons that can be used for tiling the plane, suggested that honey bees build their hexagonal cells in order to achieve the best economy of material."

Source: Nazzi, "The hexagonal shape of the honeycomb cells depends on the construction behavior of bees", Nature, 2016. https://www.nature.com/articles/srep28341
Gatterbauer. Foundations and Applications of Information Theory: https://northeastern-datalab.github.io/cs7840/fa25/

It even gets better *with* correlations



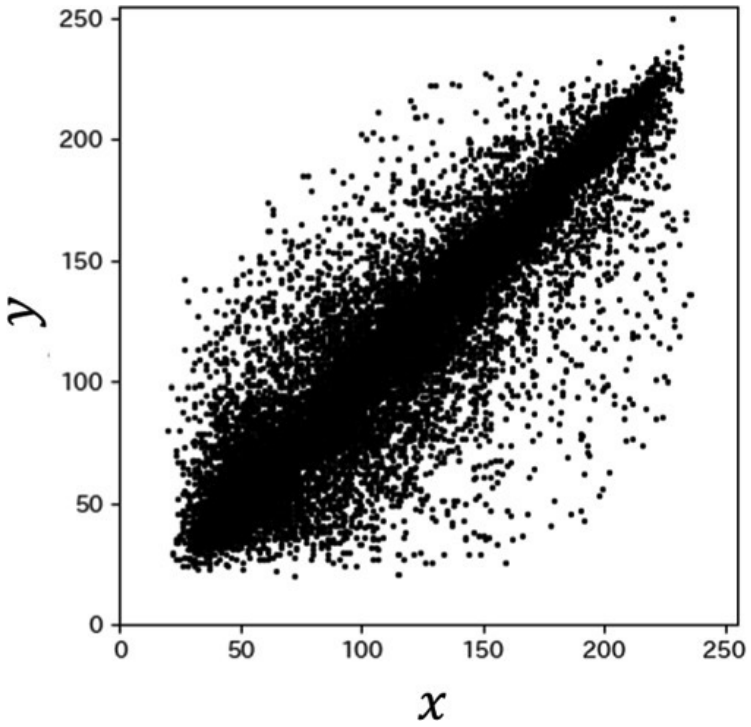
Correlation of neighboring pixels

?

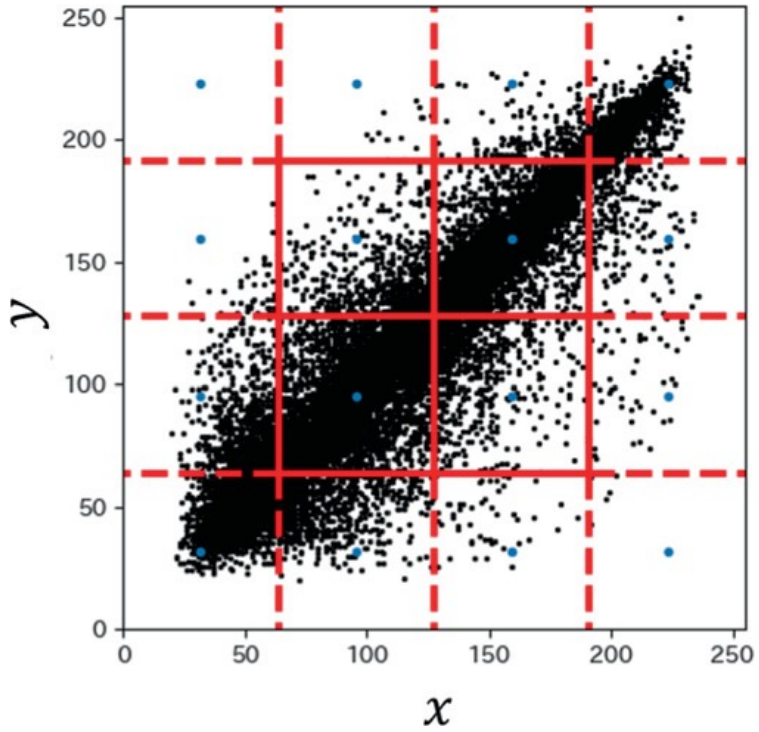
Vector space partitioning in scalar quantization (approximate)

Figure source: https://link.springer.com/chapter/10.1007/978-3-031-57840-3_40
Gatterbauer. Foundations and Applications of Information Theory: https://northeastern-datalab.github.io/cs7840/fa25/

It even gets better *with* correlations



Correlation of neighboring pixels



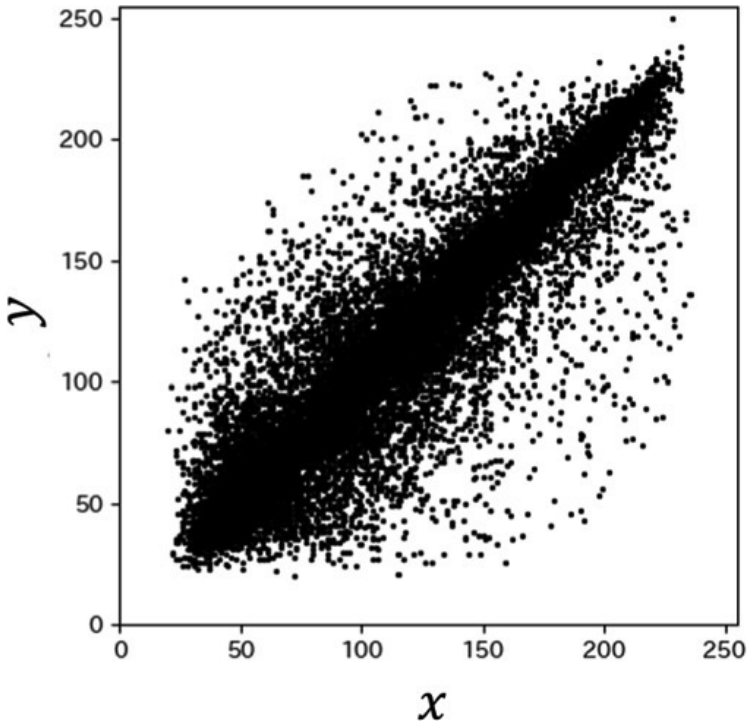
Vector space partitioning in scalar quantization (approximate)

?

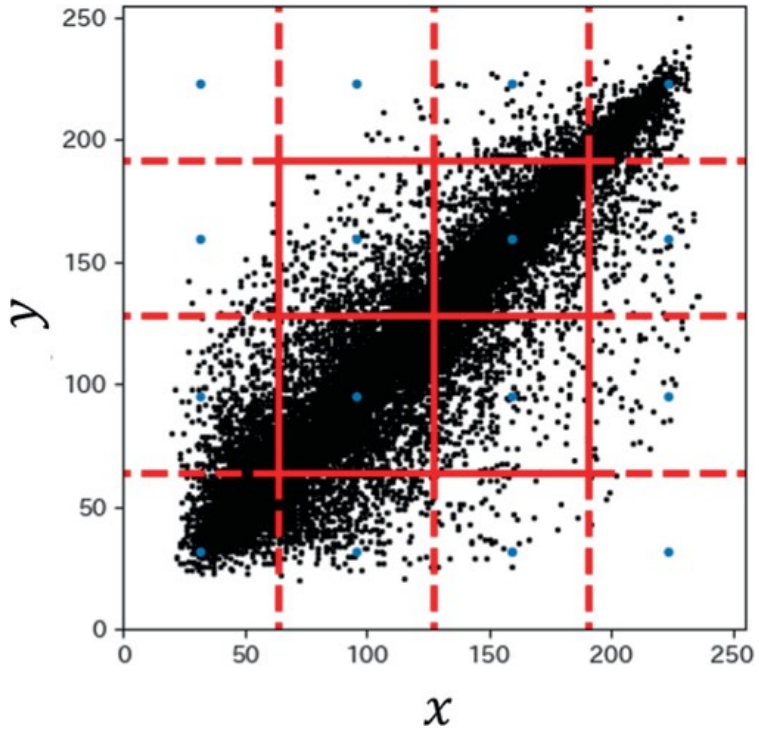
Arrangement of cells with the smallest average quantization error in vector quantization

Figure source: https://link.springer.com/chapter/10.1007/978-3-031-57840-3_40
Gatterbauer. Foundations and Applications of Information Theory: https://northeastern-datalab.github.io/cs7840/fa25/

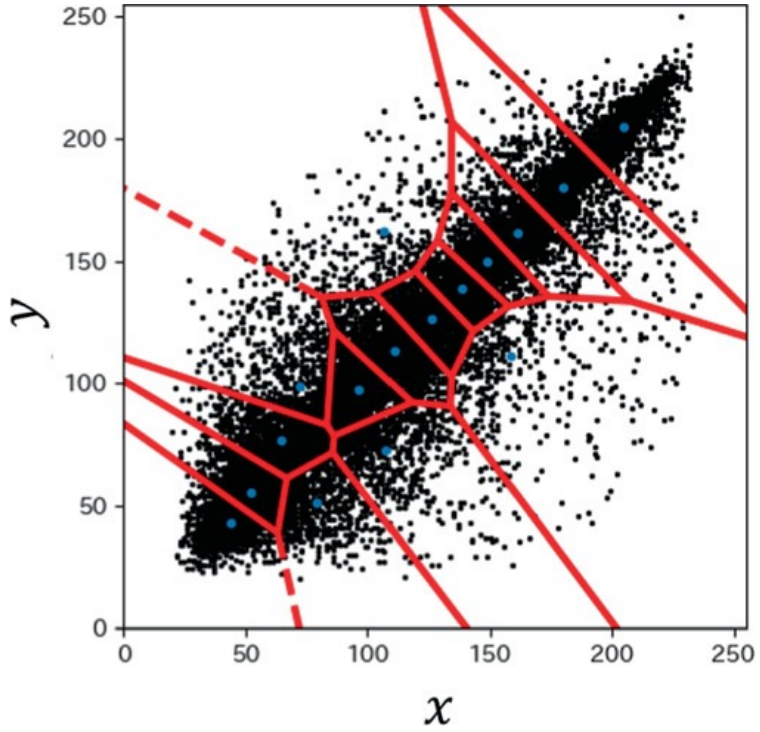
It even gets better *with* correlations



Correlation of neighboring pixels



Vector space partitioning in scalar quantization (approximate)



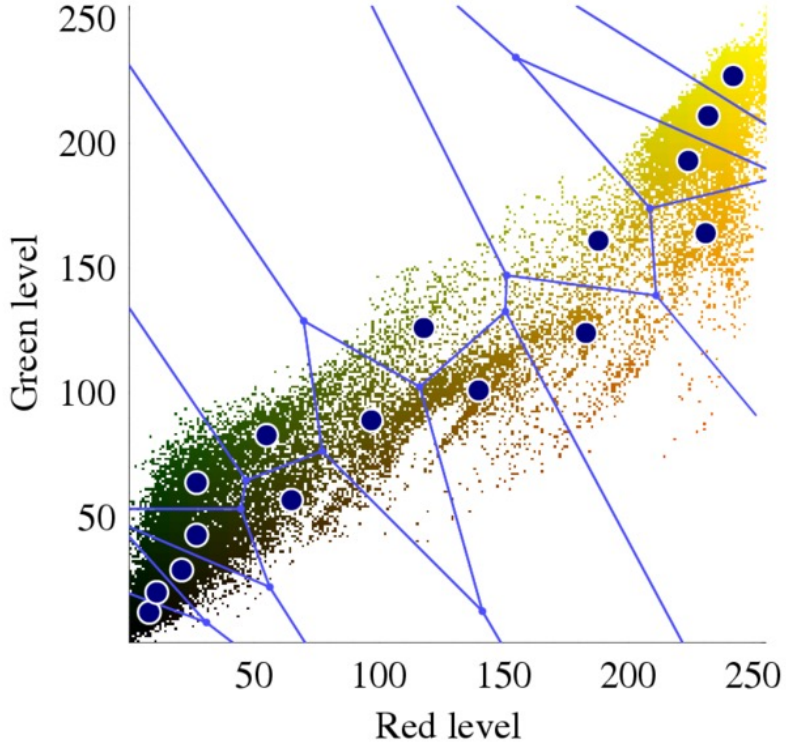
Arrangement of cells with the smallest average quantization error in vector quantization

Figure source: https://link.springer.com/chapter/10.1007/978-3-031-57840-3_40
Gatterbauer. Foundations and Applications of Information Theory: https://northeastern-datalab.github.io/cs7840/fa25/

Rate-distortion code vs. k-means

$\mathcal{X} = \{0,1,\dots,255\}$ thus 8 bit resolution (=256 levels) per color channel
$n = 2$ channels per pixel (will be encoded together); thus 16 bits per source sequence $\mathcal{X}^2$
$nR = 4$ bits (per channel sequence), thus only 16 representatives per $\mathcal{X}^2$ instead of $65536 = 256^2$
$R = 2$ bits per channel (instead of 8)

Example image with only red and green channel (for illustration)



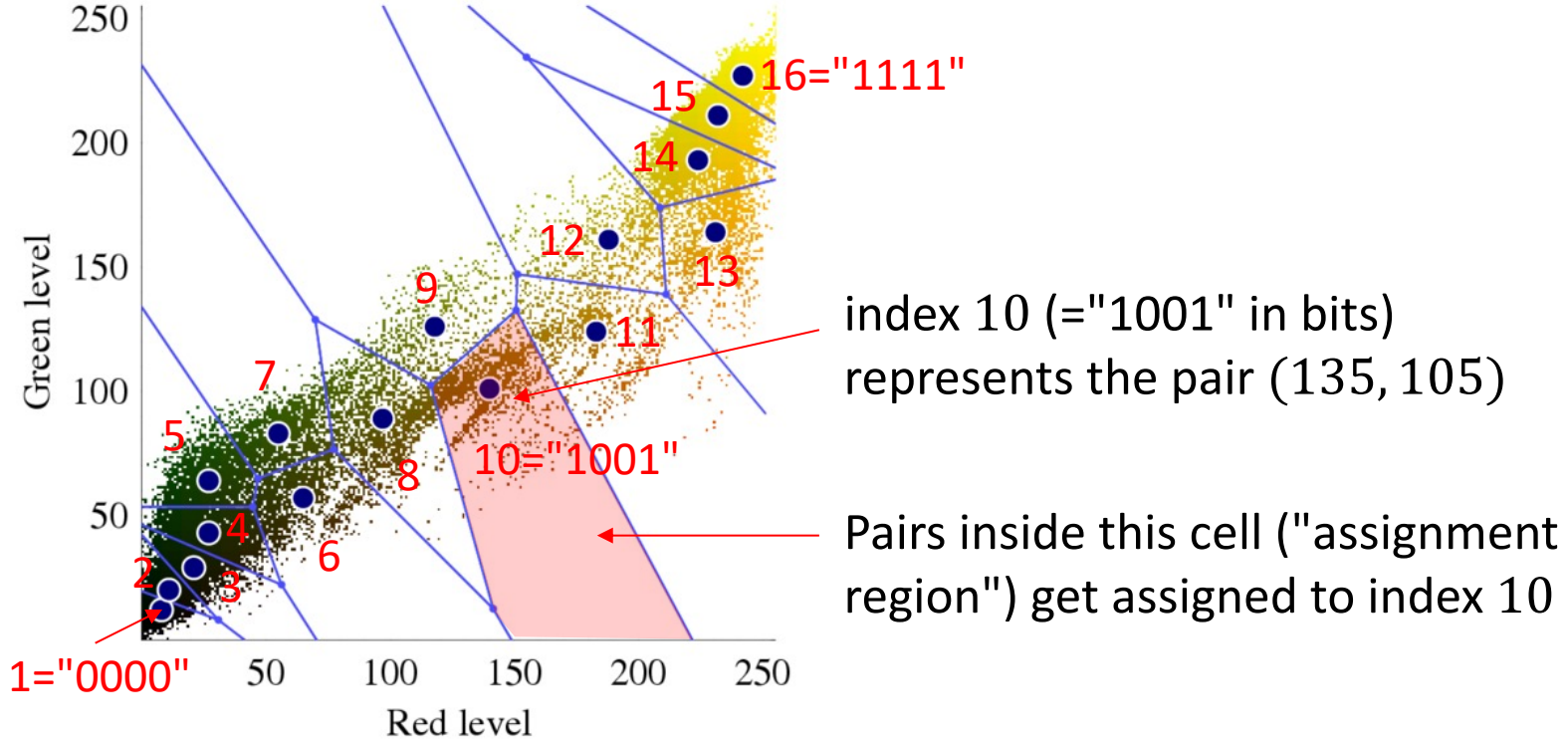
Vector quantization of colors present in the image into Voronoi cells using *k*-means

Source: https://en.wikipedia.org/wiki/K-means_clustering
Gatterbauer. Foundations and Applications of Information Theory: https://northeastern-datalab.github.io/cs7840/fa25/

Rate-distortion code vs. k-means

$\mathcal{X} = \{0,1,\ldots,255\}$ thus 8 bit resolution (=256 levels) per color channel
$n = 2$ channels per pixel (will be encoded together); thus 16 bits per source sequence $\mathcal{X}^2$
$nR = 4$ bits (per channel sequence), thus only 16 representatives per $\mathcal{X}^2$ instead of $65536 = 256^2$
$R = 2$ bits per channel (instead of 8)

Example image with only red and green channel (for illustration)



Vector quantization of colors present in the image into Voronoi cells using *k*-means

Source of images: https://en.wikipedia.org/wiki/K-means_clustering
Gatterbauer. Foundations and Applications of Information Theory: https://northeastern-datalab.github.io/cs7840/fa25/

The magic of vector quantization

- Given a set of n samples (e.g. iid from Gaussian distribution)
- We want to jointly quantize the vector $(X_1, \dots, X_n)$
- Represent these vectors using nR bits
- Represent the entire sequence by a single index taking 2^{nR} values ("representatives")

- Vector quantization achieves a lower distortion than linear (independent, scalar) quantization

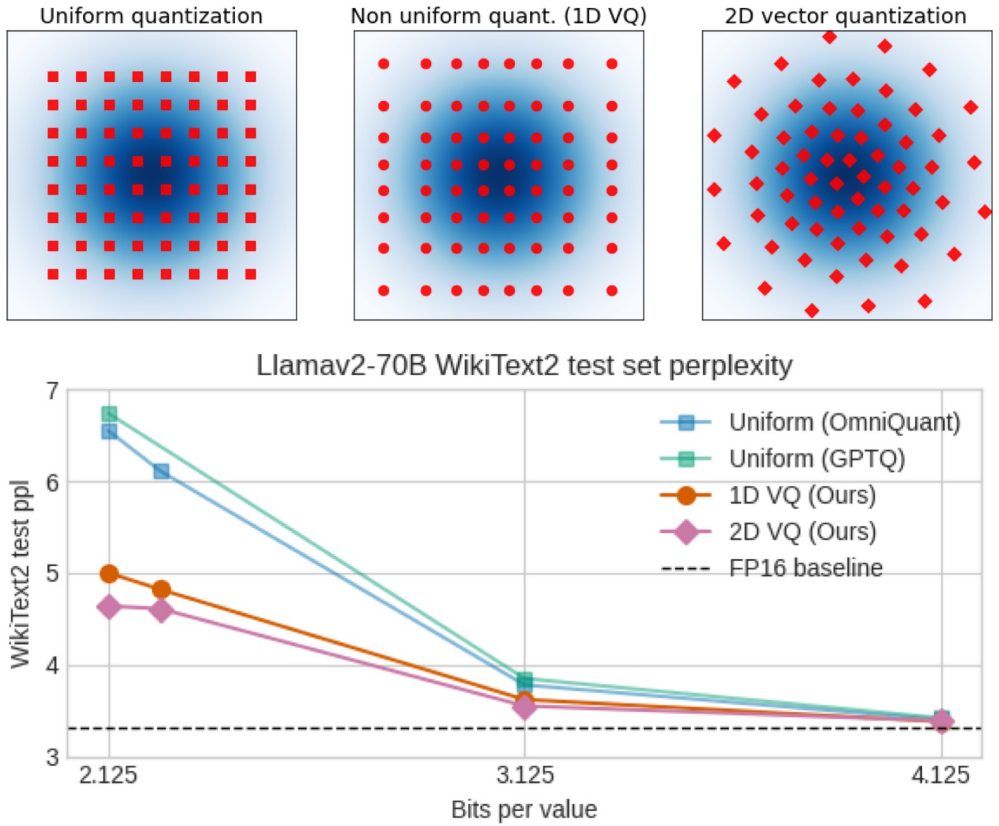


Figure 1. **Top:** An example of how vector quantization can better represent 2D normally distributed data compared to uniform quantization, non-uniform quantization. **Bottom:** Comparing GPTVQ to state-of-the-art uniform quantization on Llama 70B.

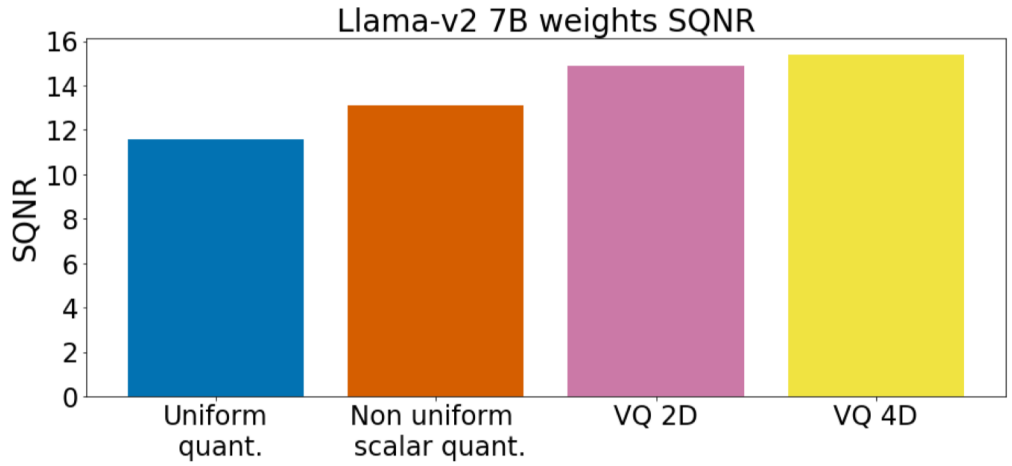
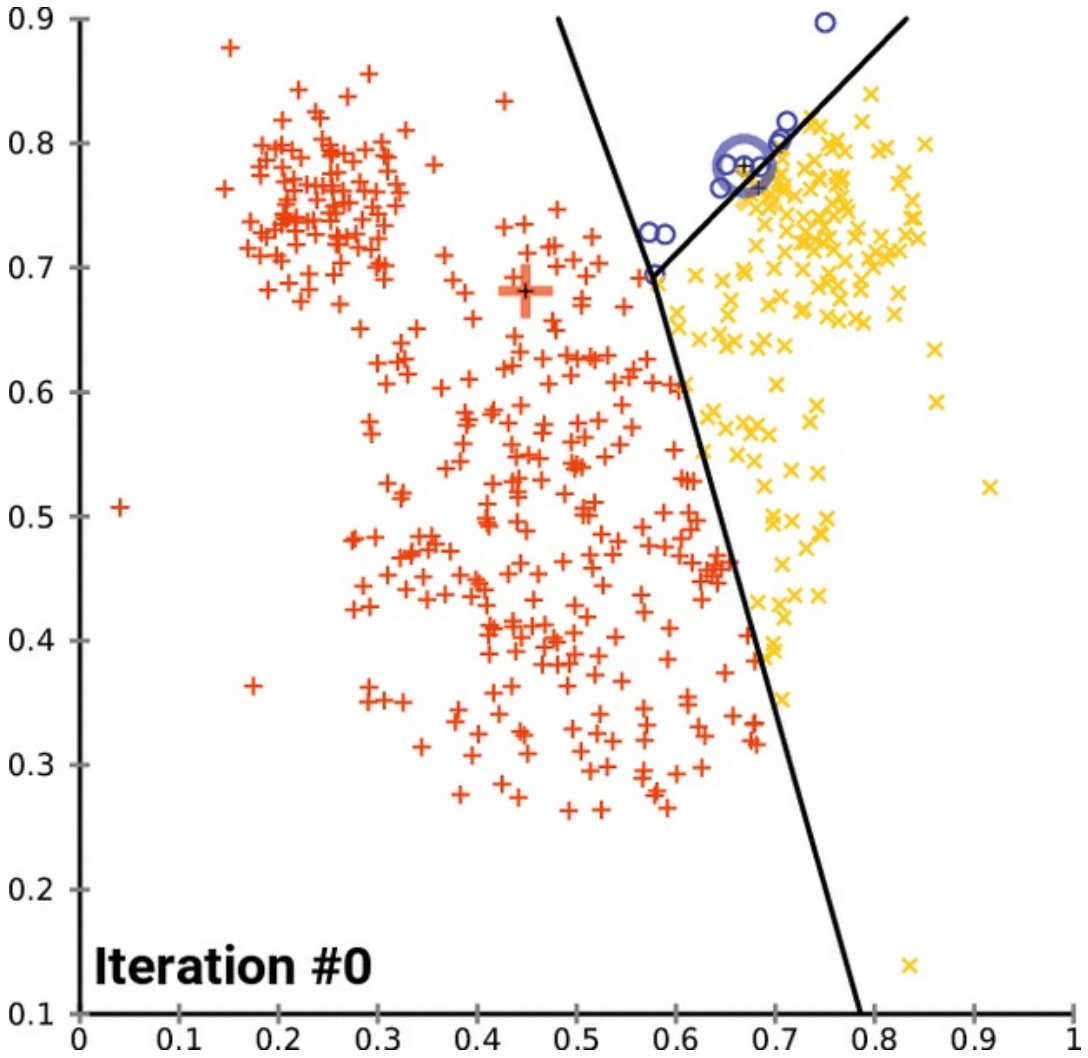


Figure 2. Quantization SQNR depending on the dimensionality for Llama-v2 7B weights. Signal-to-noise ratio increases with quantization dimensionality due to additional flexibility in the quantization grid.

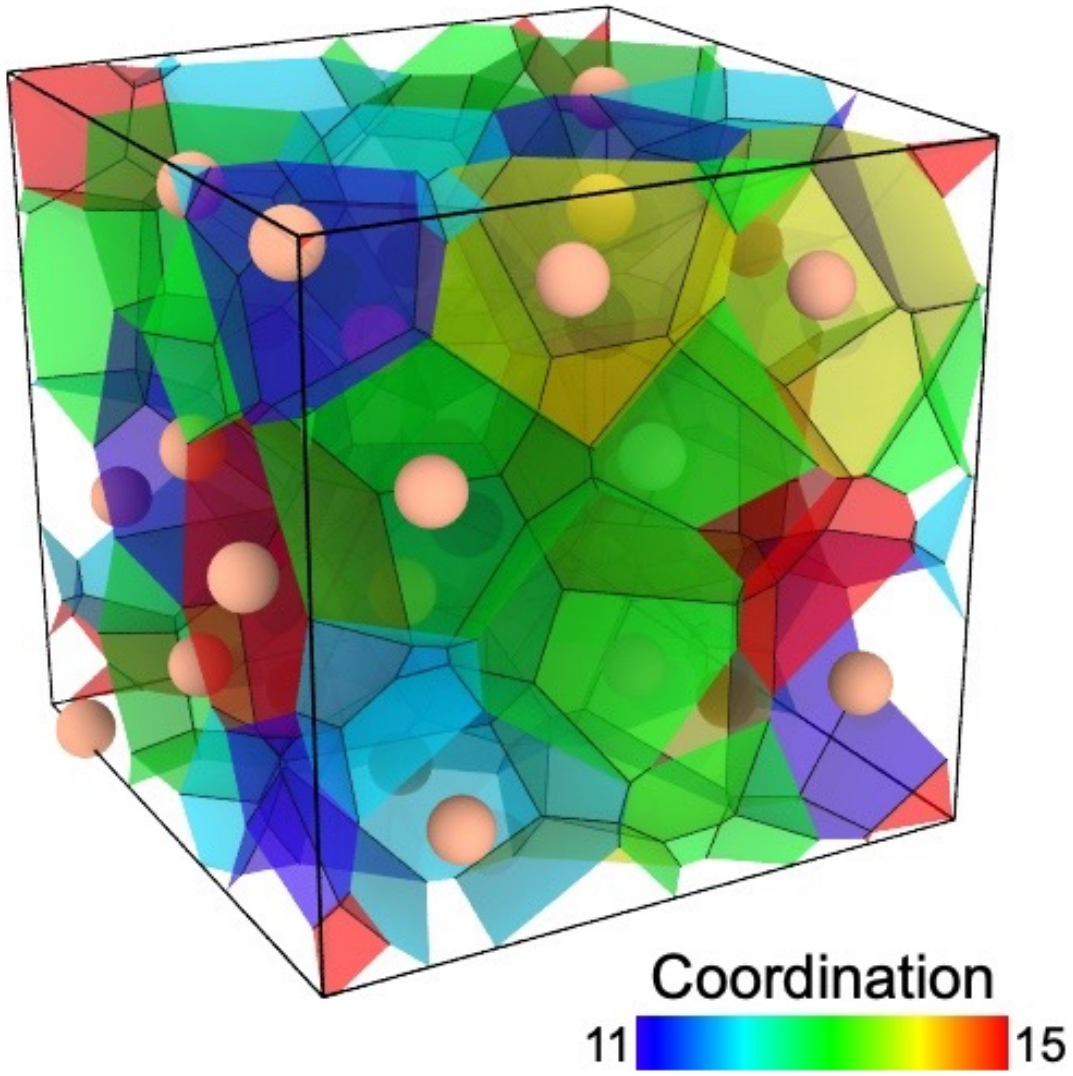
Source: Van Baalen et al. *GPTVQ: The Blessing of Dimensionality for LLM Quantization*. ICML Workshop ES-FoMo-II, 2024. https://arxiv.org/abs/2402.15319
Gatterbauer. Foundations and Applications of Information Theory: https://northeastern-datalab.github.io/cs7840/fa25/

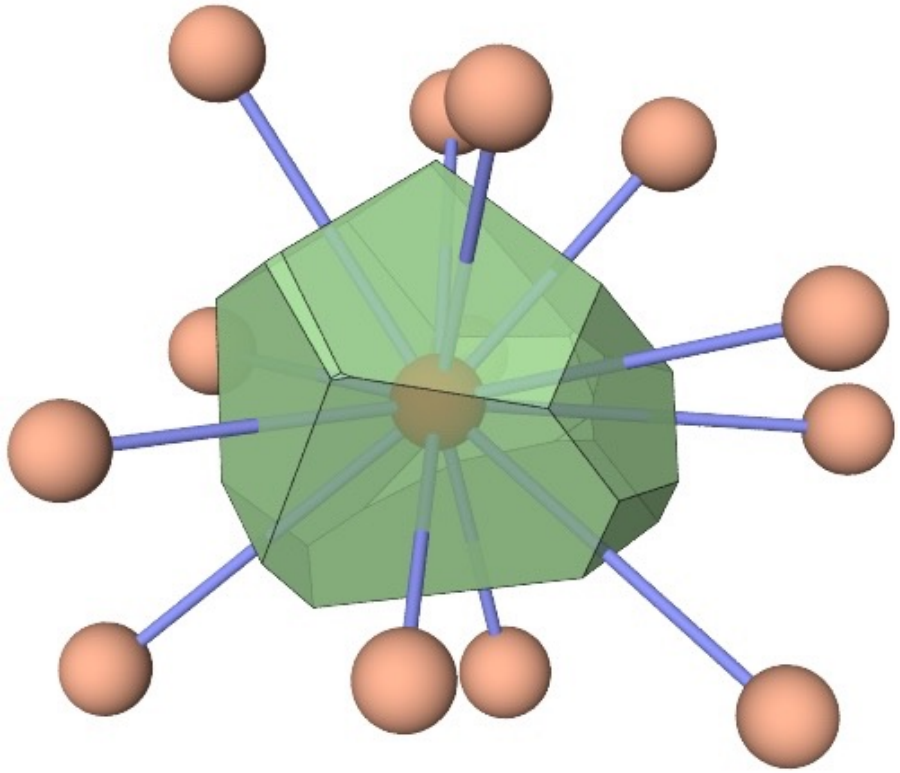
An animation of k-means



Source of animation figure: https://en.wikipedia.org/wiki/K-means_clustering
Gatterbauer. Foundations and Applications of Information Theory: https://northeastern-datalab.github.io/cs7840/fa25/

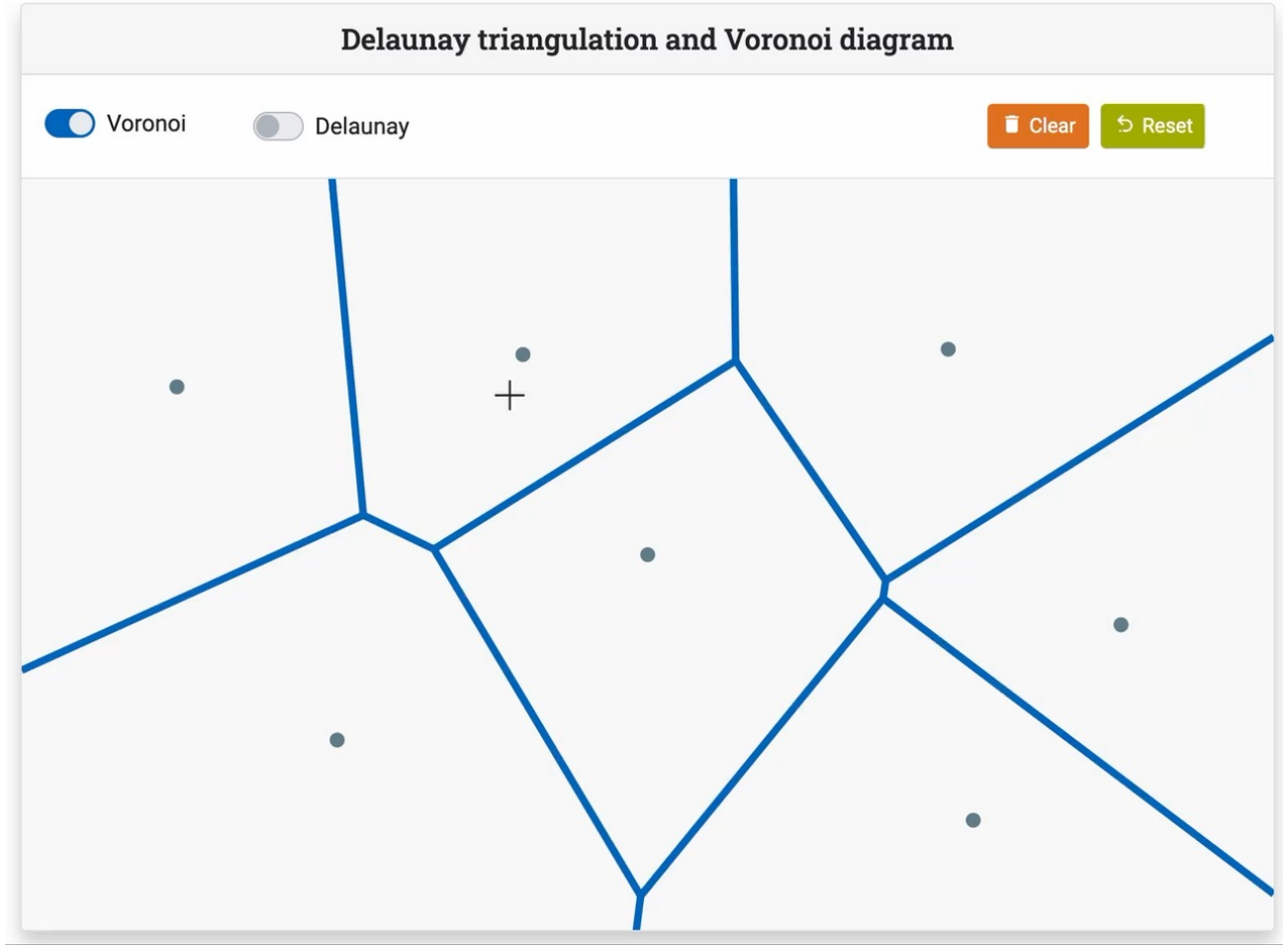
k-means in higher dimensions





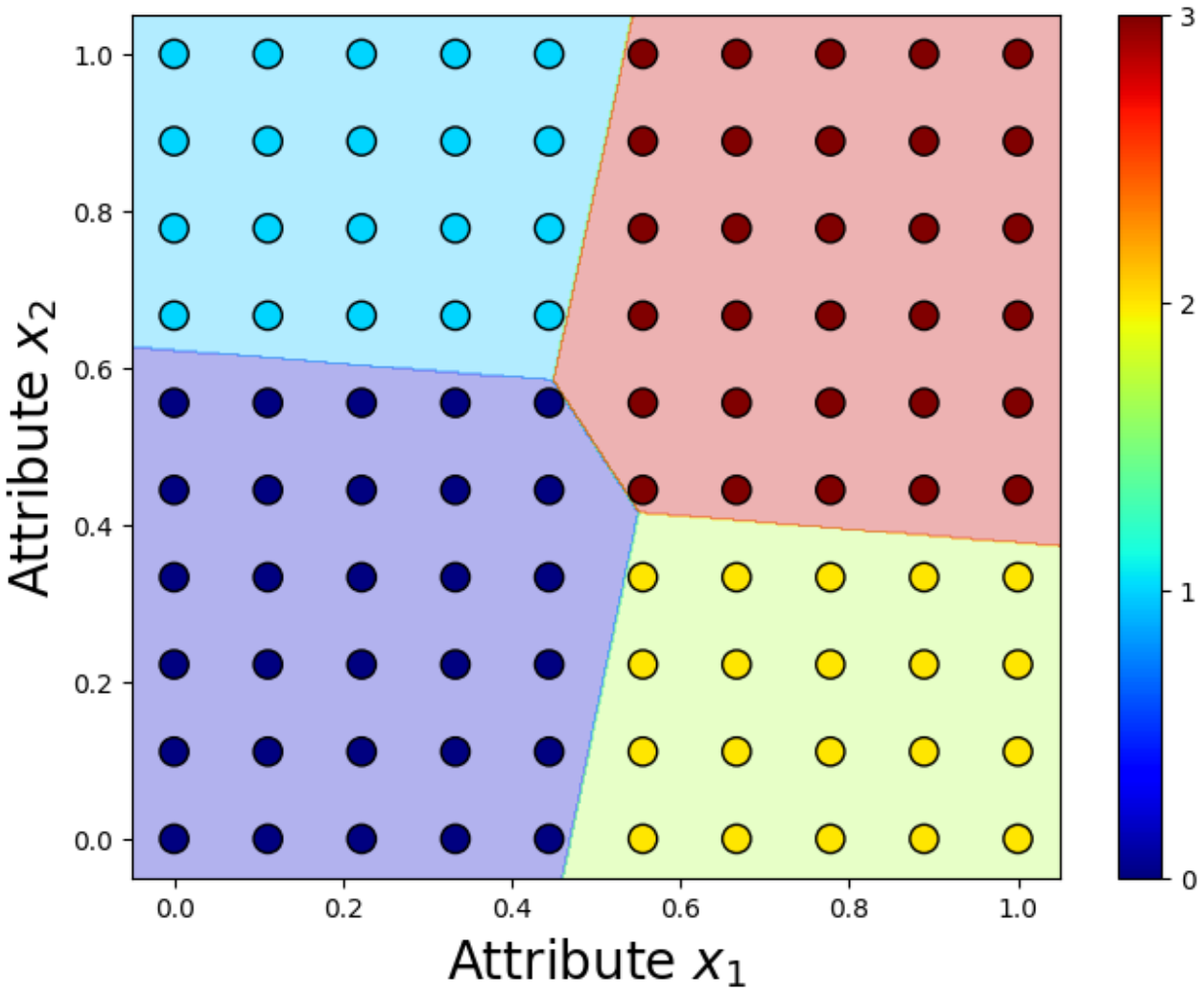
Source: https://www.ovito.org/docs/current/reference/pipelines/modifiers/voronoi_analysis.html
Gatterbauer. Foundations and Applications of Information Theory: https://northeastern-datalab.github.io/cs7840/fa25/

An animation of Voronoi tessellation

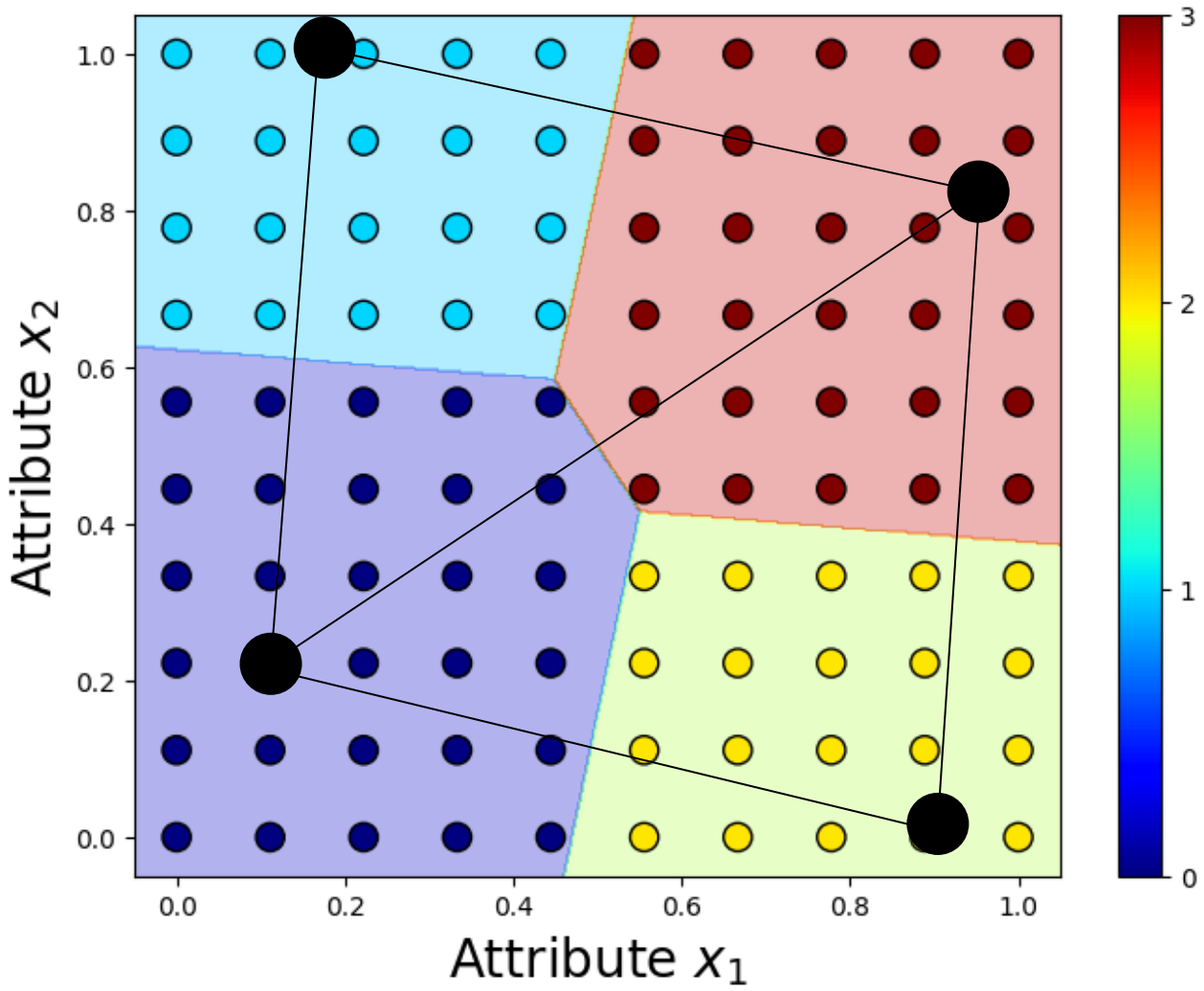


Source: https://cartography-playground.gitlab.io/playgrounds/triangulation-delaunay-voronoi-diagram/
Gatterbauer. Foundations and Applications of Information Theory: https://northeastern-datalab.github.io/cs7840/fa25/

Logistic regression vs. (soft) k-means



Logistic regression vs. (soft) k-means



Is this always possible ?

Let's make this more formal (Definitions)

Largely based on chapter 10 of
[Cover, Thomas'06] Elements of Information Theory, 2006. https://www.doi.org/10.1002/047174882X

Distortion theory

- Given: source distribution p, distortion measure d. What is the minimum expected distortion D achievable at a particular transmission rate R (in bits)?
  - In particular: What is the fundamental lower-bound on distortion D for a given rate R?
  - Intuition: more bits available (higher rate R), then fewer errors (smaller distortion D)
- Equivalently: what is the min rate R required to achieve a given distortion D?

- An intriguing aspect of this theory is that joint descriptions (think block codes) are more efficient than individual descriptions, even for independent RVs
  - The reason is found in the geometry: rectangular grid points (arising from independent descriptions) do not fill up the space efficiently (recall the earlier Voronoi diagrams)
  - Instead of representing each RV using R bits, we represent a sequence of n RVs by a single index taking 2^{nR} values. Encoding entire sequences at once achieves a lower distortion D for the same rate than independent quantization of the individual samples

Distortion function d

Distortion function (measure) d:
cost of representing a symbol by its quantized version

$$d: \mathcal{X} \times \hat{\mathcal{X}} \to \mathbb{R}^+$$

source alphabet

reproduction alphabet

Usually, $\mathcal{X} = \hat{\mathcal{X}}$

We assume the distortion to be bounded:

$$d_{\max} = \max_{x \in \mathcal{X}, \hat{x} \in \hat{\mathcal{X}}} d(x, \hat{x}) < \infty$$

What is then the distortion between sequences ?

Distortion function d

Distortion function (measure) d:
cost of representing a symbol by its quantized version

$$d: \mathcal{X} \times \hat{\mathcal{X}} \to \mathbb{R}^+$$

source alphabet

reproduction alphabet

Usually, $\mathcal{X} = \hat{\mathcal{X}}$

We assume the distortion to be bounded:

$$d_{\max} = \max_{x \in \mathcal{X}, \hat{x} \in \hat{\mathcal{X}}} d(x, \hat{x}) < \infty$$

Distortion between sequences is the <u>average per symbol</u> distortion:

$$d(x^n, \hat{x}^n) = \frac{1}{n} \sum_i d(x_i, \hat{x}_i)$$

Hamming distortion:

$$d(x, \hat{x}) = \begin{cases} 0 & \text{if } x = \hat{x} \\ 1 & \text{if } x \neq \hat{x} \end{cases}$$

same as "probability of error" distortion ?

Distortion function d

Distortion function (measure) d:
cost of representing a symbol by its quantized version

$$d: \mathcal{X} \times \hat{\mathcal{X}} \to \mathbb{R}^+$$

source alphabet

reproduction alphabet

Usually, $\mathcal{X} = \hat{\mathcal{X}}$

We assume the distortion to be bounded:

$$d_{\max} = \max_{x \in \mathcal{X}, \hat{x} \in \hat{\mathcal{X}}} d(x, \hat{x}) < \infty$$

Distortion between sequences is the <u>average per symbol</u> distortion:

$$d(x^n, \hat{x}^n) = \frac{1}{n} \sum_i d(x_i, \hat{x}_i)$$

Hamming distortion:

$$d(x, \hat{x}) = \begin{cases} 0 & \text{if } x = \hat{x} \\ 1 & \text{if } x \neq \hat{x} \end{cases}$$

same as "probability of error" distortion

$$\mathbb{E}\left[d\left(X, \hat{X}\right)\right] = \mathbb{P}[X \neq \hat{X}]$$

Squared-error distortion:

$$d(x, \hat{x}) = (x - \hat{x})^2$$

Why are we always so excited about squared errors? Think "least squares", "sum of squared errors (SSE)", or "mean of squared errors (MSE)", in linear regression, etc...

?

Distortion function d

Distortion function (measure) d:
cost of representing a symbol by its quantized version

$$d: \mathcal{X} \times \hat{\mathcal{X}} \to \mathbb{R}^+$$

source alphabet

reproduction alphabet

Usually, $\mathcal{X} = \hat{\mathcal{X}}$

We assume the distortion to be bounded:

$$d_{\max} = \max_{x \in \mathcal{X}, \hat{x} \in \hat{\mathcal{X}}} d(x, \hat{x}) < \infty$$

Distortion between sequences is the average per symbol distortion:

$$d(x^n, \hat{x}^n) = \frac{1}{n} \sum_i d(x_i, \hat{x}_i)$$

Hamming distortion:

$$d(x, \hat{x}) = \begin{cases} 0 & \text{if } x = \hat{x} \\ 1 & \text{if } x \neq \hat{x} \end{cases}$$

same as "probability of error" distortion

$$\mathbb{E}\left[d\left(X, \hat{X}\right)\right] = \mathbb{P}[X \neq \hat{X}]$$

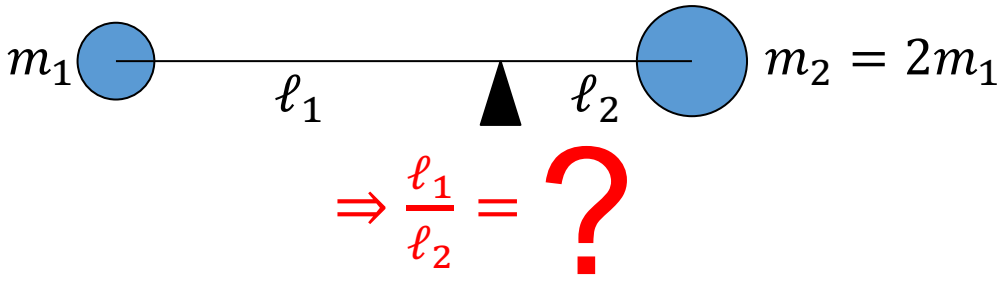
Squared-error distortion:

$$d(x, \hat{x}) = (x - \hat{x})^2$$

Connection to simple **expectations (means)**:

The squared error distortion penalizes large deviations quadratically. The **conditional mean** of X (given some available information or constraint) minimizes this penalty.

Distortion function d: Squared-error distortion



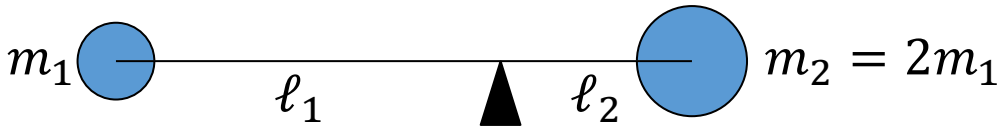
Squared-error distortion:

$$d(x, \hat{x}) = (x - \hat{x})^2$$

Connection to simple **expectations (means)**:

The squared error distortion penalizes large deviations quadratically. The **conditional mean** of X (given some available information or constraint) minimizes this penalty.

Distortion function d: Squared-error distortion



$$\Rightarrow \frac{\ell_1}{\ell_2} = 2$$

What does this have to do with squared-error distortion?

?

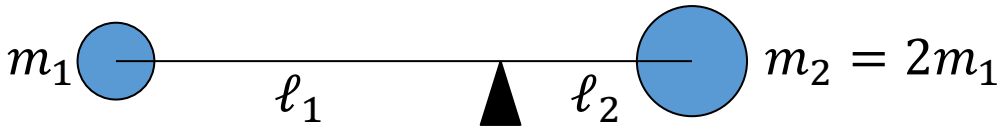
Squared-error distortion:

$$d(x, \hat{x}) = (x - \hat{x})^2$$

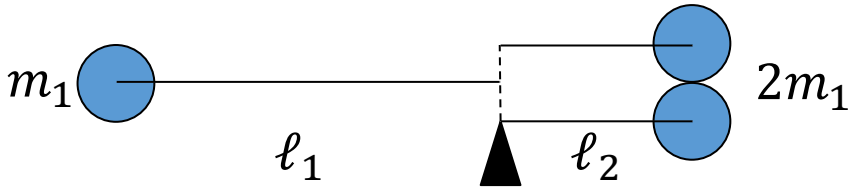
Connection to simple **expectations (means)**:

The squared error distortion penalizes large deviations quadratically. The **conditional mean** of X (given some available information or constraint) minimizes this penalty.

Distortion function d: Squared-error distortion



$$\Rightarrow \frac{\ell_1}{\ell_2} = 2$$



$$\min[\ell_1{}^2 + 2\ell_2{}^2], \text{ s.t. to } \ell_1 + \ell_2 = c$$

$$\text{SSE}(\ell_1) = \ell_1{}^2 + 2(c - \ell_1)^2$$

$$\frac{\partial \text{SSE}}{\partial \ell_1} = 2\ell_1 + 2(-2c + 2\ell_1) = 0$$

$$\Rightarrow \ell_1 = \frac{2c}{3} \quad \Rightarrow \frac{\ell_1}{\ell_2} = 2 \quad ☺$$

The arithmetic mean is the "center" ("centroid" or center of mass) of the distribution that balances the squared error!

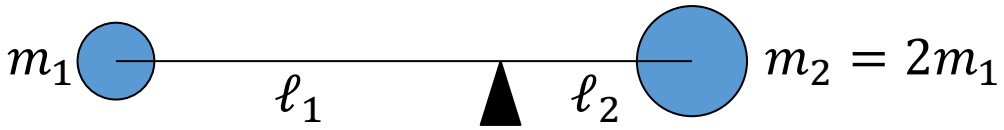
Squared-error distortion:

$$d(x, \hat{x}) = (x - \hat{x})^2$$

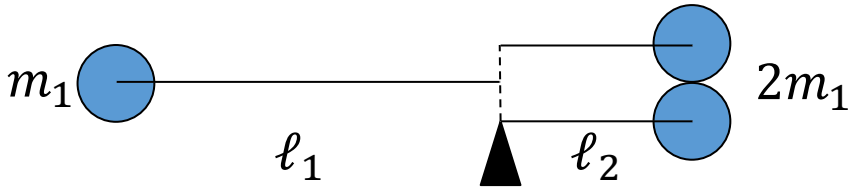
Connection to simple **expectations (means)**:

The squared error distortion penalizes large deviations quadratically. The **conditional mean** of X (given some available information or constraint) minimizes this penalty.

Distortion function d: Squared-error distortion



$$\Rightarrow \frac{\ell_1}{\ell_2} = 2$$



$$\min[\ell_1{}^2 + 2\ell_2{}^2], \text{ s.t. to } \ell_1 + \ell_2 = c$$

$$\Rightarrow \frac{\ell_1}{\ell_2} = 2$$ ☺

$$\min[\ell_1 + 2\ell_2], \text{s.t. to } \ell_1 + \ell_2 = c$$?

The arithmetic mean is the "center" ("centroid" or center of mass) of the distribution that balances the squared error!

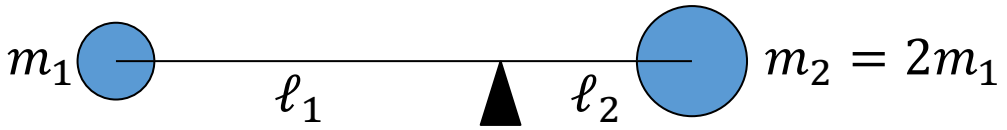
Squared-error distortion:

$$d(x, \hat{x}) = (x - \hat{x})^2$$

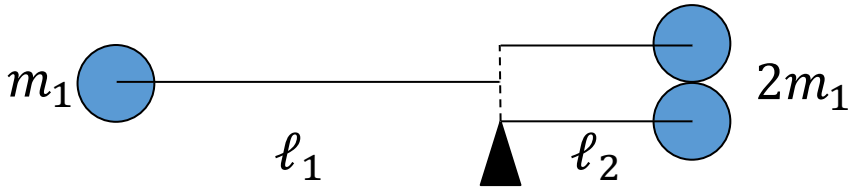
Connection to simple **expectations (means)**:

The squared error distortion penalizes large deviations quadratically. The **conditional mean** of X (given some available information or constraint) minimizes this penalty.

Distortion function d: Squared-error distortion



$$\Rightarrow \frac{\ell_1}{\ell_2} = 2$$

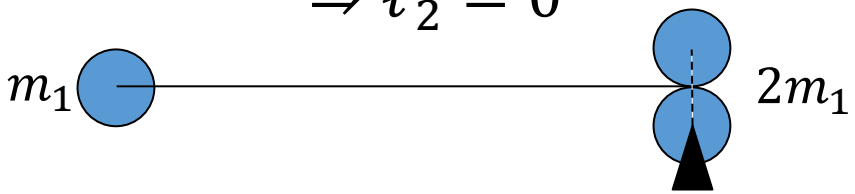


$$\min[\ell_1{}^2 + 2\ell_2{}^2], \text{ s.t. to } \ell_1 + \ell_2 = c$$

$$\Rightarrow \frac{\ell_1}{\ell_2} = 2$$ ☺

$$\min[\ell_1 + 2\ell_2], \text{s.t. to } \ell_1 + \ell_2 = c$$

$$\Rightarrow \ell_2 = 0$$



☹

The arithmetic mean is the "center" ("centroid" or center of mass) of the distribution that balances the squared error!

Squared-error distortion:

$$d(x, \hat{x}) = (x - \hat{x})^2$$

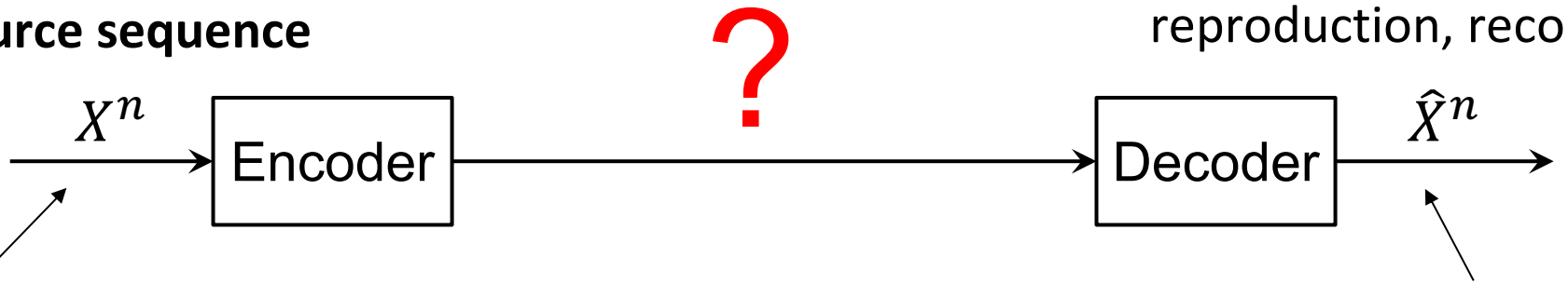
Connection to simple **expectations (means)**:

The squared error distortion penalizes large deviations quadratically. The **conditional mean** of X (given some available information or constraint) minimizes this penalty.

Rate-distortion code

representation, vector quantization, reproduction, reconstruction, ... of X^n

source sequence



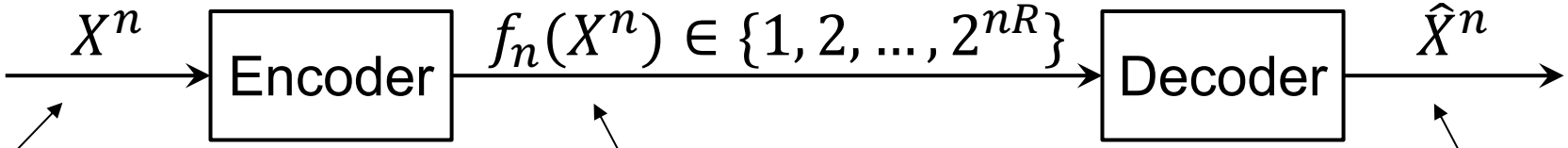
A source produces an iid **sequence** $X_1, X_2, \ldots, X_n$ with $X_i \sim p(X)$ and X taken from a source alphabet $\mathcal{X}$

The **representation** of X is $\hat{X}(X)$. The decoder represents X^n by an estimate $\hat{X}^n \in \hat{\mathcal{X}}^n$ with $\hat{\mathcal{X}}$ being the reproduction alphabet

Rate-distortion code

representation, vector quantization, reproduction, reconstruction, ... of X^n

source sequence **index**



A source produces an iid **sequence** $X_1, X_2, \dots, X_n$ with $X_i \sim p(X)$ and X taken from a source alphabet $\mathcal{X}$

We are given R bits to represent X. Thus the function $\hat{X}$ can take on 2^R different values

The **representation** of X is $\hat{X}(X)$. The decoder represents X^n by an estimate $\hat{X}^n \in \hat{\mathcal{X}}^n$ with $\hat{\mathcal{X}}$ being the reproduction alphabet

The encoder describes the source sequence X^n via an **encoding function** that maps X^n to an **index**

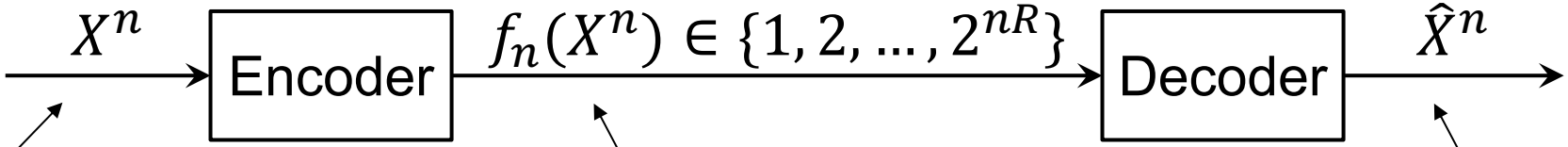
$$f_n: \mathcal{X}^n \to \{1, 2, \dots, 2^{nR}\}$$

The **decoding function** maps an index to a reconstructed sequence

$$g_n: \{1, 2, \dots, 2^{nR}\} \to \hat{\mathcal{X}}^n$$

Rate-distortion code

source sequence **index** **representation**, vector quantization, reproduction, reconstruction, ... of X^n



A source produces an iid **sequence** $X_1, X_2, \dots, X_n$ with $X_i \sim p(X)$ and X taken from a source alphabet $\mathcal{X}$

We are given R bits to represent X. Thus the function $\hat{X}$ can take on 2^R different values

The **representation** of X is $\hat{X}(X)$. The decoder represents X^n by an estimate $\hat{X}^n \in \hat{\mathcal{X}}^n$ with $\hat{\mathcal{X}}$ being the reproduction alphabet

The encoder describes the source sequence X^n via an **encoding function** that maps X^n to an **index**

$$f_n: \mathcal{X}^n \to \{1, 2, \dots, 2^{nR}\}$$

The **decoding function** maps an index to a reconstructed sequence

$$g_n: \{1, 2, \dots, 2^{nR}\} \to \hat{\mathcal{X}}^n$$

A $(2^{nR}, n)$-**rate distortion code** consists of f_n and g_n.

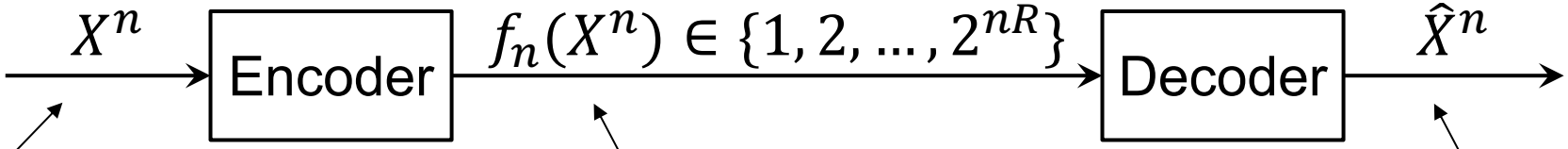
$g_n(1), \dots, g_n(2^{nR})$: **codebook** (contains **code points**)

$f_n^{-1}(1), \dots, f_n^{-1}(2^{nR})$: **assignment regions**

What is its associated distortion ?

Rate-distortion code

source sequence **index** **representation**, vector quantization, reproduction, reconstruction, ... of X^n



A source produces an iid **sequence** $X_1, X_2, \dots, X_n$ with $X_i \sim p(X)$ and X taken from a source alphabet $\mathcal{X}$

We are given R bits to represent X. Thus the function $\hat{X}$ can take on 2^R different values

The **representation** of X is $\hat{X}(X)$. The decoder represents X^n by an estimate $\hat{X}^n \in \hat{\mathcal{X}}^n$ with $\hat{\mathcal{X}}$ being the reproduction alphabet

The encoder describes the source sequence X^n via an **encoding function** that maps X^n to an **index**

$$f_n: \mathcal{X}^n \to \{1, 2, \dots, 2^{nR}\}$$

The **decoding function** maps an index to a reconstructed sequence

$$g_n: \{1, 2, \dots, 2^{nR}\} \to \hat{\mathcal{X}}^n$$

A $(2^{nR}, n)$-**rate distortion code** consists of f_n and g_n.

$g_n(1), \dots, g_n(2^{nR})$: **codebook** (contains **code points**)

$f_n^{-1}(1), \dots, f_n^{-1}(2^{nR})$: **assignment regions**

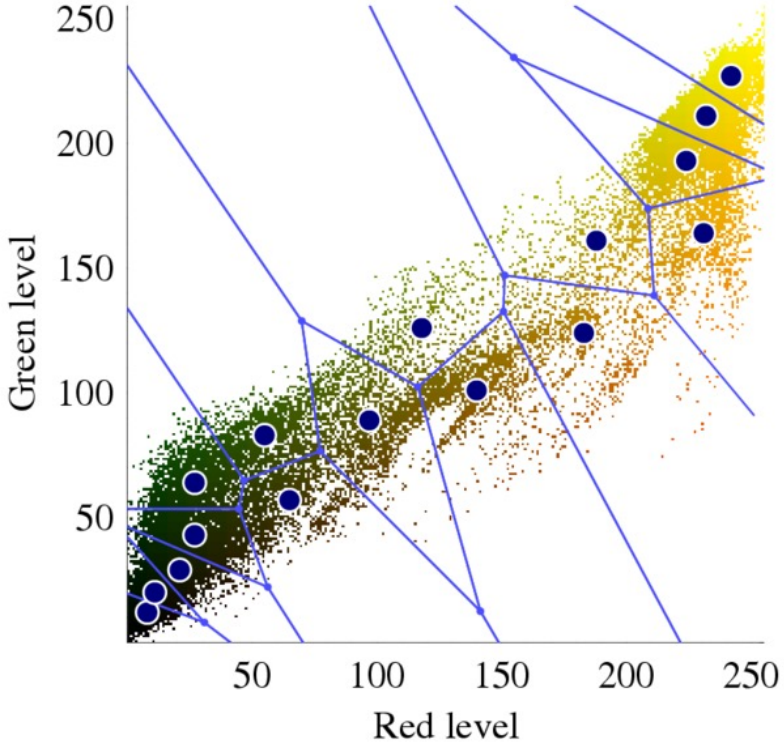
Its associated distortion is:

$$D = \mathbb{E}_{X \sim p}[d(X^n, \overbrace{g_n(f_n(X^n))}^{\hat{X}^n})] = \sum_{x^n} p(x^n) \cdot d(x^n, g_n(f_n(x^n)))$$

Rate-distortion code vs. k-means

$\mathcal{X} = \{0,1,\dots,255\}$ thus 8 bit resolution (=256 levels) per color channel
$n = 2$ channels per pixel (will be encoded together); thus 16 bits per source sequence $\mathcal{X}^2$
$nR = 4$ bits (per channel sequence), thus only **???** representatives per $\mathcal{X}^2$ instead of $65536 = 256^2$

Example image with only red and green channel (for illustration)



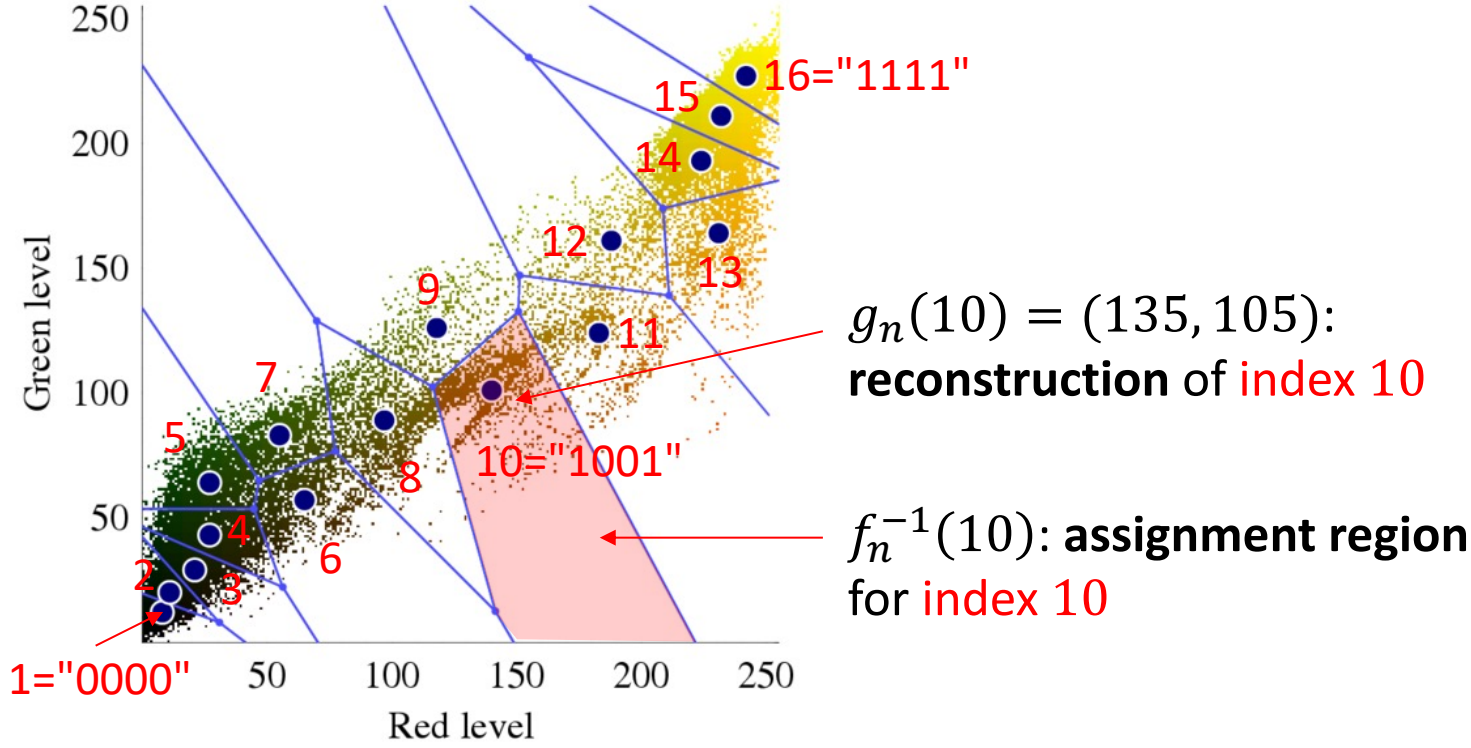
Vector quantization of colors present in the image into Voronoi cells using *k*-means

Source of images: https://en.wikipedia.org/wiki/K-means_clustering
Gatterbauer. Foundations and Applications of Information Theory: https://northeastern-datalab.github.io/cs7840/fa25/

Rate-distortion code vs. k-means

$\mathcal{X} = \{0,1,\ldots,255\}$ thus 8 bit resolution (=256 levels) per color channel
$n = 2$ channels per pixel (will be encoded together); thus 16 bits per source sequence $\mathcal{X}^2$
$nR = 4$ bits (per channel sequence), thus only 16 representatives per $\mathcal{X}^2$ instead of $65536 = 256^2$
$R = 2$ bits per channel (instead of 8)

Example image with only red and green channel (for illustration)



Vector quantization of colors present in the image into Voronoi cells using *k*-means

Source of images: https://en.wikipedia.org/wiki/K-means_clustering
Gatterbauer. Foundations and Applications of Information Theory: https://northeastern-datalab.github.io/cs7840/fa25/

Main theorem of Rate-distortion theory

A **rate distortion pair** (R, D) is **achievable** if there exists a sequence of $(2^{nR}, n)$-rate distortion codes (f_n, g_n) with

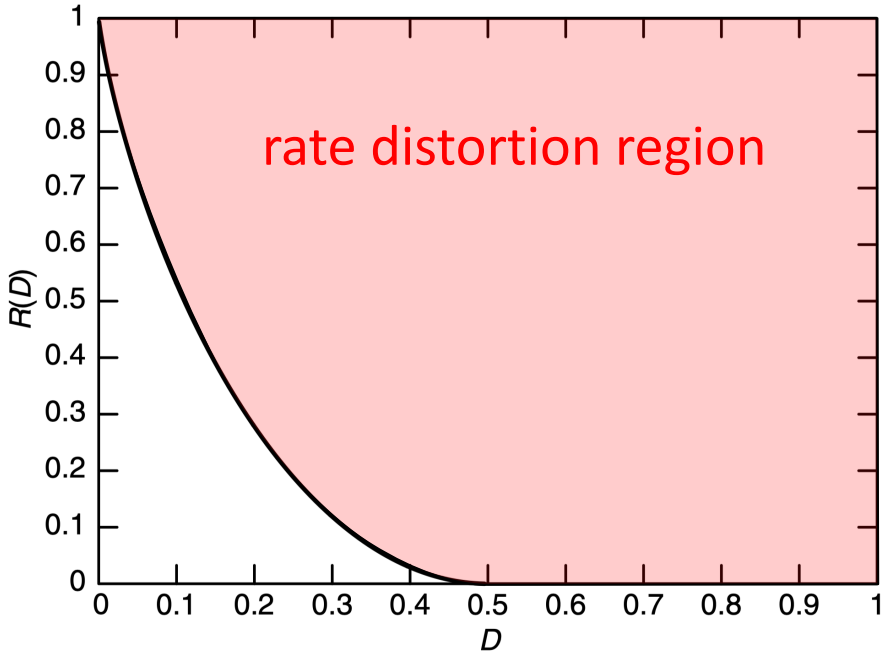
$$\lim_{n\to\infty} \mathbb{E}_{X\sim p}[d(X^n, g_n(f_n(X^n)))] \leq D$$

A **rate distortion region** for a source is the closure of the set of achievable distortion pairs (R, D).

The **rate distortion** $R(D)$ is the infimum of rates R s.t. (R, D) is in the rate distortion region of the source for given distortion D.

infimum: greatest lower bound (does not have to be in the set, in contrast to min)

rate distortion function for Bernoulli $p = 0.5$ with Hamming distortion



Main theorem of Rate-distortion theory

A **rate distortion pair** (R, D) is **achievable** if there exists a sequence of $(2^{nR}, n)$-rate distortion codes (f_n, g_n) with

$$\lim_{n\to\infty} \mathbb{E}_{X\sim p}[d(X^n, g_n(f_n(X^n)))] \leq D$$

A **rate distortion region** for a source is the closure of the set of achievable distortion pairs (R, D).

The **rate distortion** $R(D)$ is the infimum of rates R s.t. (R, D) is in the rate distortion region of the source for given distortion D.

infimum: greatest lower bound (does not have to be in the set, in contrast to min)

THEOREM: The rate distortion $R(D)$ for an iid source $X\sim p$ and bounded distortion $d(X, \hat{X})$ is

reconstruction of X

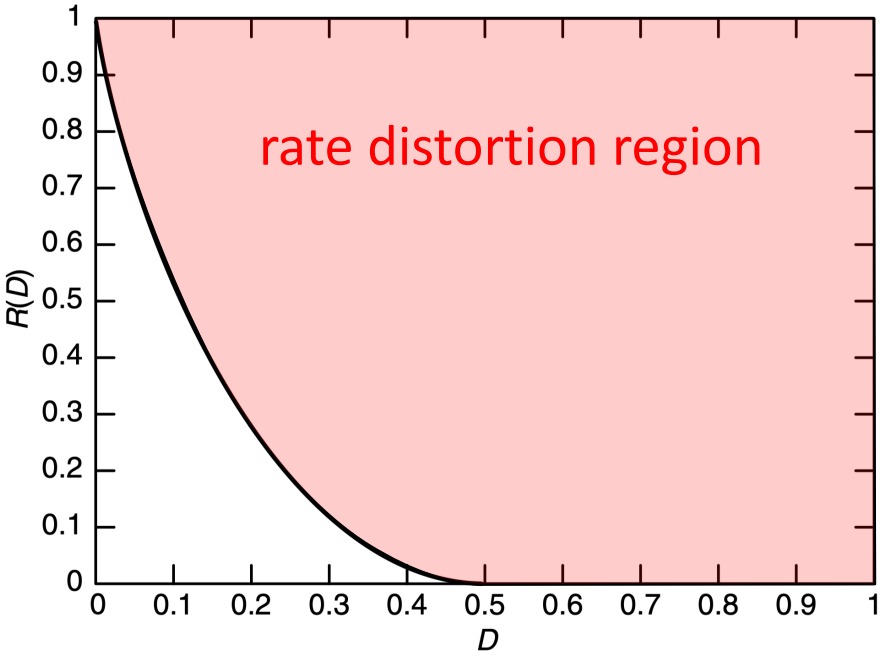
$$R(D) = \min_{p(\hat{X}|X):\ \mathbb{E}[d(X,\hat{X})] \leq D} I(X;\hat{X})$$

maximum allowable distortion

$\sum_{(x,\hat{x})} p(x,\hat{x}) \cdot d(x,\hat{x})$

$p(x) \cdot p(\hat{x}|x)$

rate distortion function for Bernoulli $p = 0.5$ with Hamming distortion



Rate Distortion function $R(D)$

reconstruction of X

$$R(D) = \min_{p(\hat{X}|X):\ \mathbb{E}[d(X,\hat{X})] \le D} I(X;\hat{X})$$

maximum allowable distortion

Channel capacity C

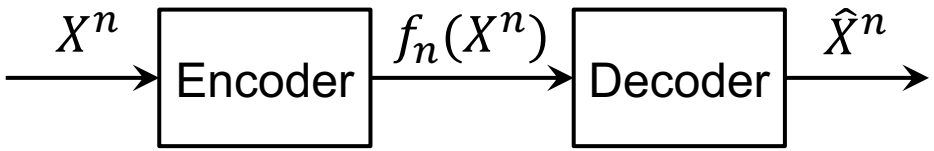
$$C = \max_{p(X)} I(X;Y)$$

RATE-DISTORTION THEORY

CHANNEL CODING THEORY

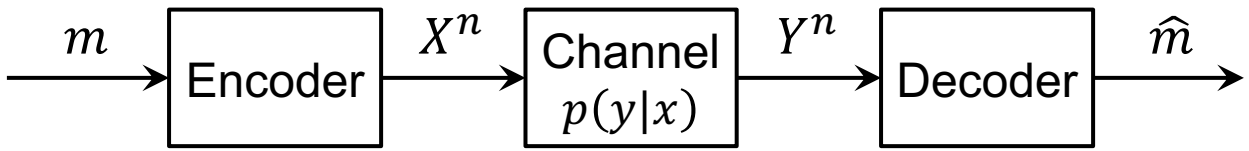
Why is one minimizing, the other maximizing mutual information ?

source sequence — index — representation



$\in \{1, 2, \dots, 2^{nR}\}$

message — channel symbols — message estimate



Rate Distortion function $R(D)$

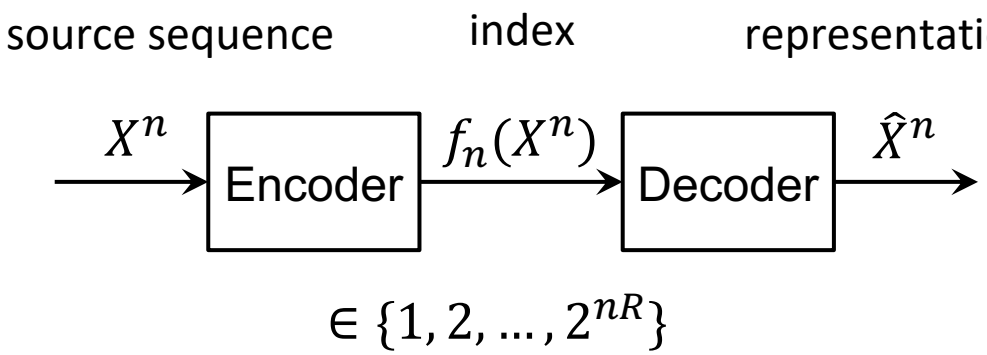
reconstruction of X

$$R(D) = \min_{p(\hat{X}|X):\ \mathbb{E}[d(X,\hat{X})] \le D} I(X;\hat{X})$$

maximum allowable distortion

RATE-DISTORTION THEORY

- compress data X into a small representation $\hat{X}$ while satisfying a given distortion constraint $\le D$ (and thus achieve a certain level of fidelity)

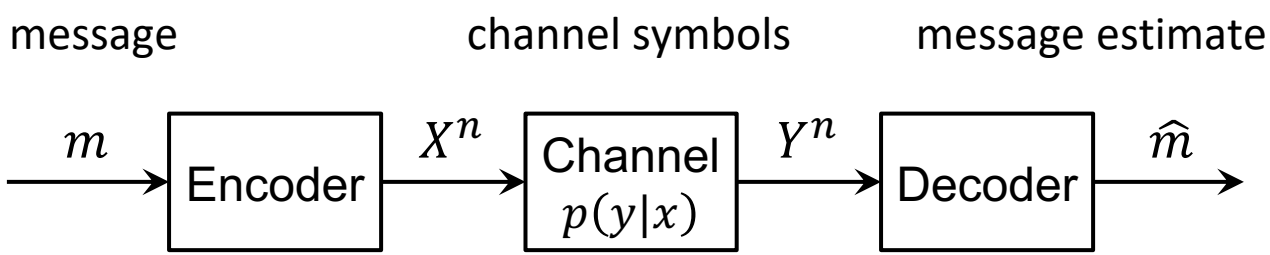


Channel capacity C

$$C = \max_{p(X)} I(X;Y)$$

CHANNEL CODING THEORY

- encode the information (via its input distribution $p(X)$) as to maximize the amount of information successfully transmitted through the channel



Rate Distortion function $R(D)$

reconstruction of X

$$R(D) = \min_{p(\hat{X}|X):\ \mathbb{E}[d(X,\hat{X})] \leq D} I(X;\hat{X})$$

maximum allowable distortion

RATE-DISTORTION THEORY

- compress data X into a small representation $\hat{X}$ while satisfying a given distortion constraint $\leq D$ (and thus achieve a certain level of fidelity)
- find the minimum communication rate $R = I(X;\hat{X})$ necessary to satisfy distortion $\leq D$
- Optimization (Minimization) over $p(\hat{X}|X)$ reflects the search for the most efficient encoding that meets the distortion D.

Channel capacity C

$$C = \max_{p(X)} I(X;Y)$$

CHANNEL CODING THEORY

- encode the information (via its input distribution $p(X)$) as to maximize the amount of information successfully transmitted through the channel
- find the maximum reliable communication rate $R = I(X;Y)$ that a channel can support (its capacity C)
- Optimization (Maximization) over $p(X)$ reflects the search for the input distribution that makes best use of the channel's capacity to transmit information.

2 Examples

Largely based on Ch10 of [Cover, Thomas'06] Elements of Information Theory, 2006. https://doi.org/10.1002/047174882X , and Ch 8 of [Yeung'08] Information Theory and Network Coding. https://doi.org/10.1007/978-0-387-79234-7

Rate Distortion for Bernoulli p with Hamming distortion

Consider a binary source $X \sim$ Bernoulli(p):

$$p(X = 1) = p$$
$$p(X = 0) = 1 - p$$

WLOG, assume $p \leq 0.5$.

Assume a Hamming distortion measure:

$$d(x, \hat{x}) = \begin{cases} 0 & \text{if } x = \hat{x} \\ 1 & \text{if } x \neq \hat{x} \end{cases}$$

If we had to guess x, should we rather guess x=0 or x=1?

$$\mathbb{P}[X = 0] = 1 - p \geq 0.5$$

Our minimum expected distortion between X and a constant estimate of x=0 is:

$$\begin{aligned} D_{\max} &= \mathbb{E}[d(X, 0)] \\ &= \mathbb{P}[X = 1] \\ &= p \end{aligned}$$

Rate Distortion for Bernoulli p with Hamming distortion

Consider a binary source $X \sim$ Bernoulli(p):

$$p(X = 1) = p$$
$$p(X = 0) = 1 - p$$

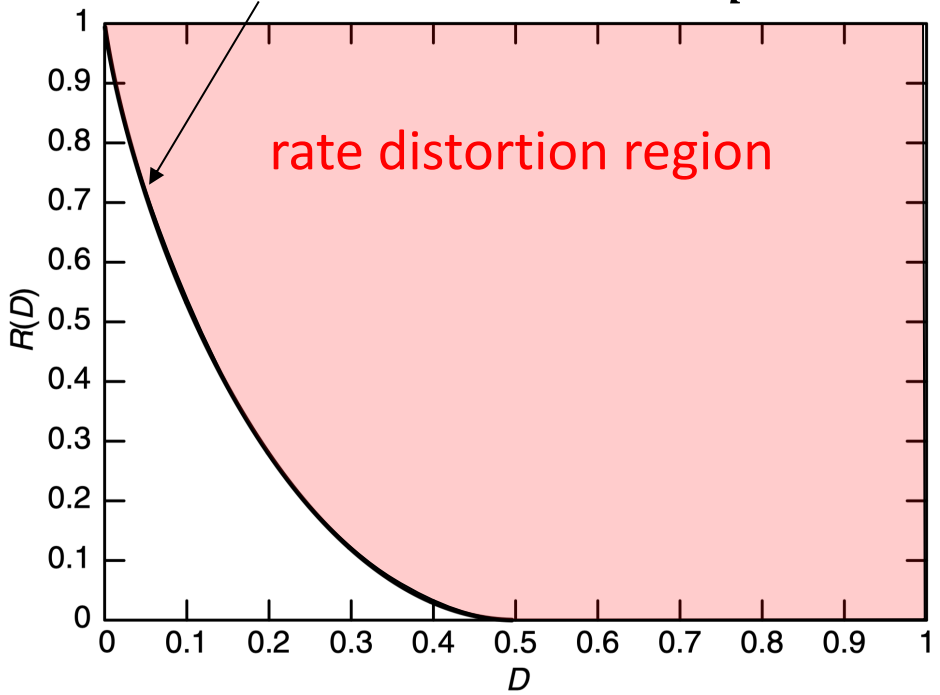
WLOG, assume $p \leq 0.5$.

Assume a Hamming distortion measure:

$$d(x, \hat{x}) = \begin{cases} 0 & \text{if } x = \hat{x} \\ 1 & \text{if } x \neq \hat{x} \end{cases}$$

What is the description rate $R(D)$ required to describe X with an expected proportion of errors less than or equal to D?

rate distortion function for $p = 0.5$



$$R(D) = \begin{cases} H(p) - H(D), & 0 < D < p \\ 0, & \text{else} \end{cases}$$

Two steps (instead of minimizing $I(X; \hat{X})$ directly): We first find a lower bound. We then show that this lower bound is achievable.

Python activities file 234: https://github.com/northeastern-datalab/cs7840-activities/tree/main/notebooks/234_rate_distortion_Bernoulli.ipynb

Rate Distortion for Bernoulli p with Hamming distortion

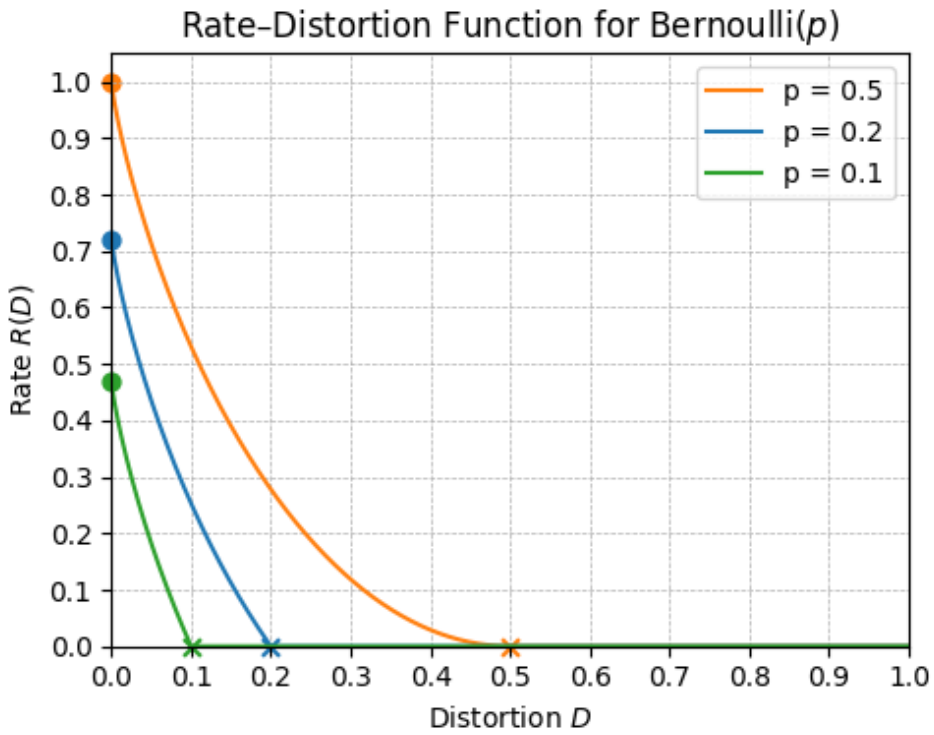
Consider a binary source $X \sim$ Bernoulli(p):

$$p(X = 1) = p$$
$$p(X = 0) = 1 - p$$

WLOG, assume $p \leq 0.5$.

Assume a Hamming distortion measure:

$$d(x, \hat{x}) = \begin{cases} 0 & \text{if } x = \hat{x} \\ 1 & \text{if } x \neq \hat{x} \end{cases}$$



What is the description rate $R(D)$ required to describe X with an expected proportion of errors less than or equal to D?

$$R(D) = \begin{cases} H(p) - H(D), & 0 < D < p \\ 0, & \text{else} \end{cases}$$

Two steps (instead of minimizing $I(X; \hat{X})$ directly): We first find a lower bound. We then show that this lower bound is achievable.

Python activities file 234: https://github.com/northeastern-datalab/cs7840-activities/tree/main/notebooks/234_rate_distortion_Bernoulli.ipynb

Rate Distortion for Bernoulli p with Hamming distortion

Lower bound:

For any joint distribution satisfying the distortion constraint, we know:

$$I(X;\hat{X}) = H(X) - H(X|\hat{X})$$

$$= H(p) - H(Y|\hat{X})$$

$$\geq H(p) - H(Y)$$

$$\geq H(p) - H(D)$$

We thus have:

$$R(D) \geq H(p) - H(D)$$

Let Y denote $d(X,\hat{X})$, or $(Y=1) \Leftrightarrow (X \neq \hat{X})$. Then conditioning on $\hat{X}$, X and Y determine each other, and thus the uncertainty (entropy H) is the same if we consider X or Y: $H(X|\hat{X}) = H(Y|\hat{X})$

$H(Y|\hat{X}) \leq H(Y)$: our uncertainty can only reduce by conditioning (i.e. learning additional information)

since $\mathbb{P}[Y] = \mathbb{P}[X \neq \hat{X}] = \mathbb{E}[d(X \neq \hat{X})] \leq D$ for $D \leq p$, and $H(x)$ increases with $x \leq 0.5$

Rate Distortion for Bernoulli p with Hamming distortion

We now show that the lower bound is actually the rate distortion function by finding a joint distribution $(X, \hat{X})$ that meets the distortion constraint and has $R(D) = I(X; \hat{X})$.

Concretely, for $0 \leq D \leq p$, we can achieve value $H(p) - H(D)$ for the rate distortion function $R(D)$ by choosing $(X; \hat{X})$ to have the joint distribution given by the following binary symmetric channel:

Recall that for a Binary Symmetric Channel $I(X; Y) = H(Y) - H(p)$.

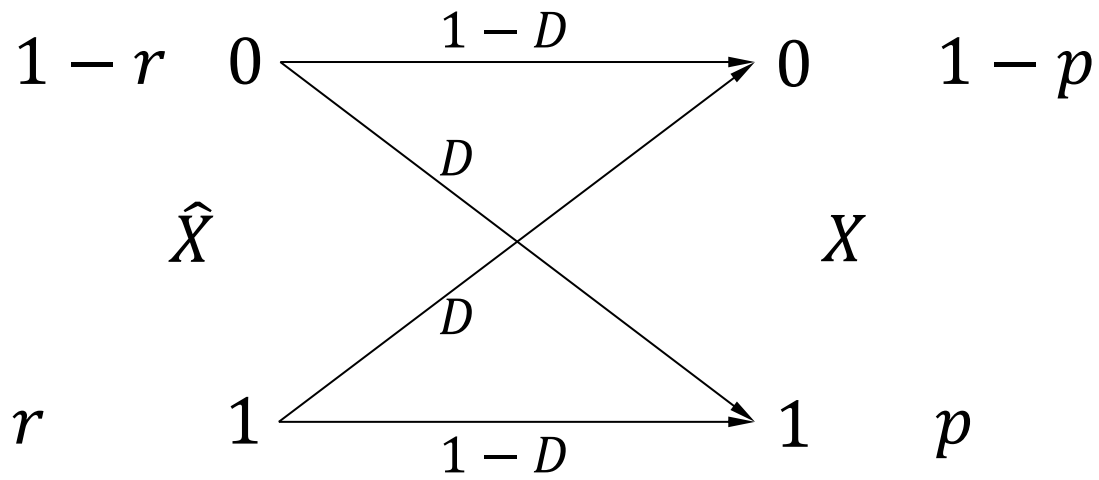
Here just p corresponds to D and Y to X: $I(X; \hat{X}) = H(p) - H(D)$.

We need to find an appropriate $r_{\hat{X}}$ of $\hat{X}$ at the input of the channel s.t. the output distribution of X is the specified p_X.

Let $r = \mathbb{P}[\hat{X} = 1]$. Then choose r s.t.

$$r(1 - D) + (1 - r)D = p$$

$$\Rightarrow \quad r = \frac{p - D}{1 - 2D}$$



Rate Distortion for Bernoulli p with Hamming distortion

If $D \leq p \leq 0.5$, then:

- $\mathbb{P}[\hat{X} = 1] \geq 0$ and $\mathbb{P}[\hat{X} = 0] \geq 0$
- $I(X; \hat{X}) = H(X) - H(X|\hat{X}) = H(p) - H(D)$

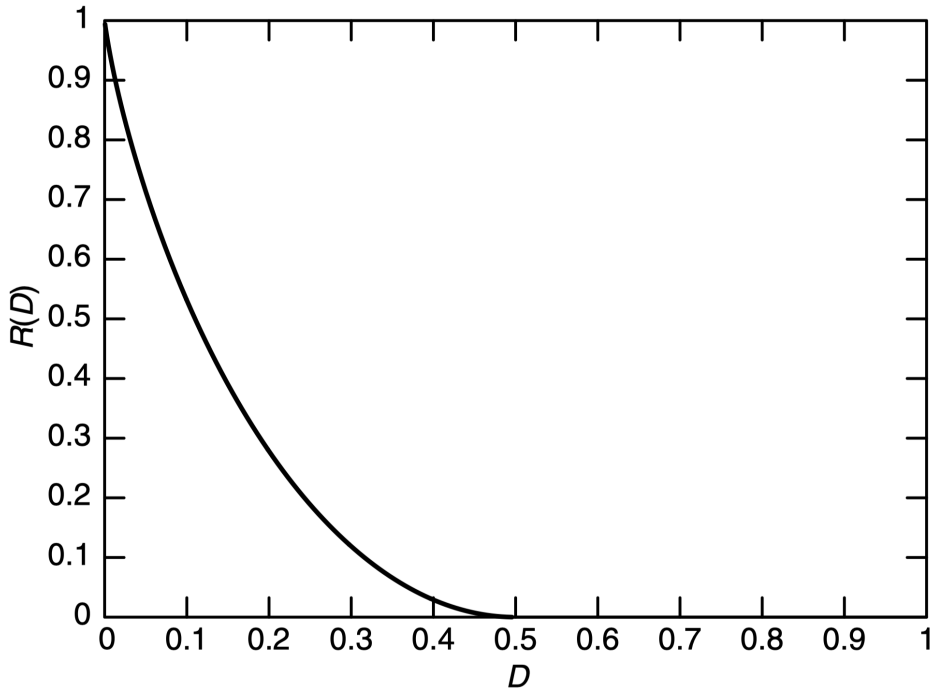
and the expected distortion is $\mathbb{P}[X \neq \hat{X}] = D$.

Indeed, the uncertainty of X when $\hat{X}$ is known is D, hence $H(X|\hat{X}) = H(D)$.

If $D \geq p$, then:

- We can achieve $R(D) = 0$ by letting $\hat{X} = 0$ with probability 1

rate distortion function



$$R(D) = \begin{cases} H(p) - H(D), & 0 < D < p \\ 0, & \text{else} \end{cases}$$

Rate Distortion for Gaussian source with squared error distortion

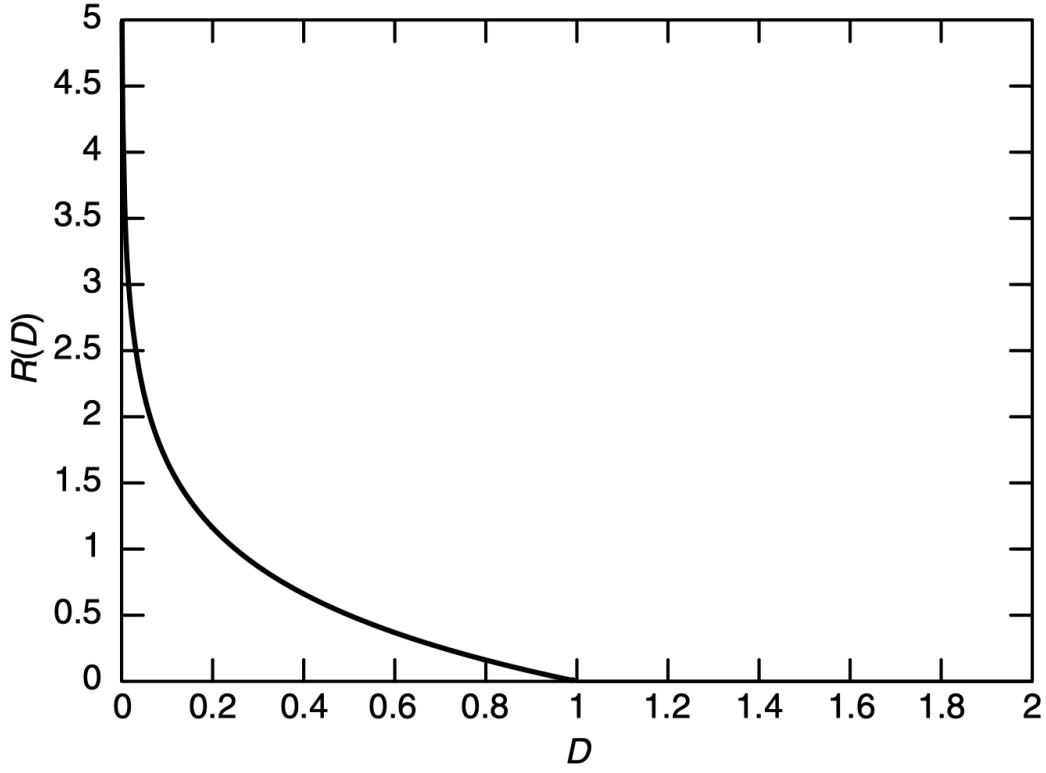
Consider a Gaussian source $X \sim \mathcal{N}(0, \sigma^2)$.

Assume a squared error distortion

$$d(x, \hat{x}) = (x - \hat{x})^2$$

WLOG, assume $p \leq 0.5$

Then the description rate $R(D)$ required to describe X with an expected proportion of errors less than or equal to D can be shown to be as follows:



$$R(D) = \begin{cases} \frac{1}{2}\ln\left(\frac{\sigma^2}{D}\right), & 0 \leq D \leq \sigma^2 \\ 0, & \text{else} \end{cases}$$

Proof: see book

Rate Distortion for Gaussian source with squared error distortion

We can rewrite $R(D)$ to express the distortion D in terms of the rate R:

$$D(R) = \sigma^2 2^{-2R}$$

Each bit of description reduces the expected distortion by a factor of 4.

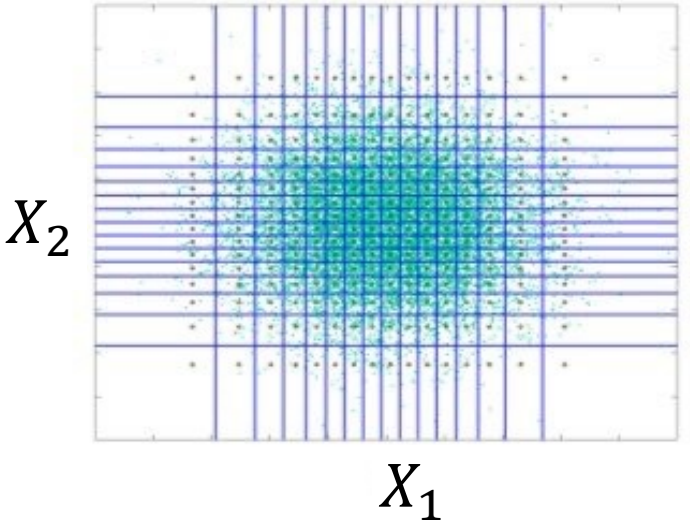
With a 1-bit description, the best expected square error is $0.25\sigma^2$.

Our simple 1-bit quantization from earlier can be calculated to be $0.36\sigma^2$.

The rate distortion limit $R(D)$ is achieved by considering several distortion problems in succession (longer block lengths) instead of considering each problem separately.

Geometry of longer block lengths:

Independent 4-bit quantization:



Blocklength $n = 2$ and 4-bit per sample

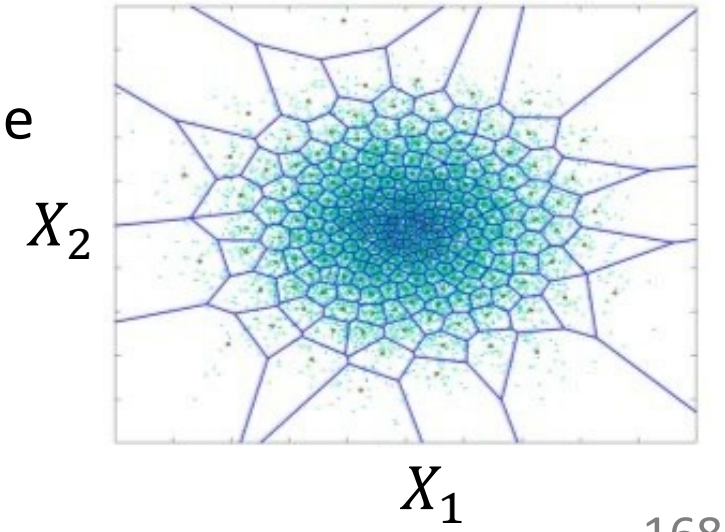


Figure source: https://ieeexplore.ieee.org/document/7767821/

Information Bottleneck

Three-step abstractions

$X \longrightarrow Y \longrightarrow Z$ Markov chain

What do we know? ?

Three-step abstractions

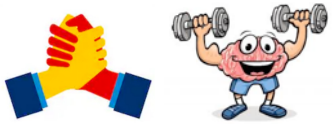
$X \longrightarrow Y \longrightarrow Z$ Markov chain $X \perp Z|Y$ $I(X;Y) \geq I(X;Z)$

$p(x,y,z) = p(x) \cdot p(y|x) \cdot p(z|\cancel{x,}y)$ also: $p(y) \cdot p(x|y) \cdot p(z|\cancel{x,}y)$

$X \longrightarrow Y \longrightarrow f(Y)$ Data processing inequality

?

Three-step abstractions



$X \longrightarrow Y \longrightarrow Z$ Markov chain $X \perp Z|Y$ $I(X;Y) \geq I(X;Z)$

$p(x,y,z) = p(x) \cdot p(y|x) \cdot p(z|\cancel{x},y)$ also: $p(y) \cdot p(x|y) \cdot p(z|\cancel{x},y)$

$X \longrightarrow Y \longrightarrow f(Y)$ Data processing inequality $I(X;Y) \geq I\big(X;f(Y)\big)$

$\theta \longrightarrow \mathbf{X} \longrightarrow T(\mathbf{X})$ Sufficient statistics

?

Three-step abstractions

$X \longrightarrow Y \longrightarrow Z$ Markov chain $X \perp Z|Y \quad I(X;Y) \geq I(X;Z)$

$p(x,y,z) = p(x) \cdot p(y|x) \cdot p(z|\cancel{x},y)$ also: $p(y) \cdot p(x|y) \cdot p(z|\cancel{x},y)$

$X \longrightarrow Y \longrightarrow f(Y)$ Data processing inequality $I(X;Y) \geq I\big(X;f(Y)\big)$

$\theta \longrightarrow \mathbf{X} \longrightarrow T(\mathbf{X})$ Sufficient statistics

A statistic T is sufficient for θ if it preserves all the information in $\mathbf{X}$ about θ:

$\theta \perp \mathbf{X}|T(\mathbf{X}) \Leftrightarrow I\big(\theta;T(\mathbf{X})\big) = I(\theta;\mathbf{X}) \Leftrightarrow \theta \to T(\mathbf{X}) \to \mathbf{X}$ also forms a Markov chain

minimal sufficient: simplest mapping of $\mathbf{X}$ that captures all the information in $\mathbf{X}$ about θ.

Information bottleneck

Assume: We want to determine Y from X. Goal: find a representation $\hat{X}$ of X that captures the relevant features, yet compresses X by removing irrelevant parts that do not contribute to the prediction of Y

?

Three-step abstractions


$X \longrightarrow Y \longrightarrow Z$ Markov chain $X \perp Z|Y \quad I(X;Y) \geq I(X;Z)$

$p(x,y,z) = p(x) \cdot p(y|x) \cdot p(z|\cancel{x},y)$ also: $p(y) \cdot p(x|y) \cdot p(z|\cancel{x},y)$

$X \longrightarrow Y \longrightarrow f(Y)$ Data processing inequality $I(X;Y) \geq I\big(X;f(Y)\big)$

$\theta \longrightarrow \mathbf{X} \longrightarrow T(\mathbf{X})$ Sufficient statistics

A statistic T is sufficient for θ if it preserves all the information in $\mathbf{X}$ about θ:

$\theta \perp \mathbf{X}|T(\mathbf{X}) \Leftrightarrow I\big(\theta;T(\mathbf{X})\big) = I(\theta;\mathbf{X}) \Leftrightarrow \theta \to T(\mathbf{X}) \to \mathbf{X}$ also forms a Markov chain

minimal sufficient: simplest mapping of $\mathbf{X}$ that captures all the information in $\mathbf{X}$ about θ.

Information bottleneck

Assume: We want to determine Y from X. Goal: find a representation $\hat{X}$ of X that captures the relevant features "max $I\big(Y;\hat{X}\big)$", yet compresses X by removing irrelevant parts that do not contribute to the prediction of Y: "min $I\big(X;\hat{X}\big)$".

Three-step abstractions

$X \longrightarrow Y \longrightarrow Z$ Markov chain $X \perp Z|Y \quad I(X;Y) \geq I(X;Z)$

$p(x,y,z) = p(x) \cdot p(y|x) \cdot p(z|\cancel{x,}y)$ also: $p(y) \cdot p(x|y) \cdot p(z|\cancel{x,}y)$

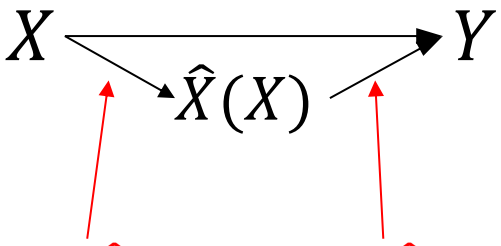
$X \longrightarrow Y \longrightarrow f(Y)$ Data processing inequality $I(X;Y) \geq I\big(X;f(Y)\big)$

$\theta \longrightarrow \mathbf{X} \longrightarrow T(\mathbf{X})$ Sufficient statistics

A statistic T is sufficient for θ if it preserves all the information in $\mathbf{X}$ about θ:

$\theta \perp \mathbf{X}|T(\mathbf{X}) \Leftrightarrow I\big(\theta;T(\mathbf{X})\big) = I(\theta;\mathbf{X}) \Leftrightarrow \theta \to T(\mathbf{X}) \to \mathbf{X}$ also forms a Markov chain

minimal sufficient: simplest mapping of $\mathbf{X}$ that captures all the information in $\mathbf{X}$ about θ.



Information bottleneck

Assume: We want to determine Y from X. Goal: find a representation $\hat{X}$ of X that captures the relevant features "max $I\big(Y;\hat{X}\big)$", yet compresses X by removing irrelevant parts that do not contribute to the prediction of Y: "min $I\big(X;\hat{X}\big)$".

$$\mathcal{L}^* = \min_{p(\hat{X}|X)} [\mathcal{L}(\hat{X})] \qquad \mathcal{L}(\hat{X}) = I(X;\hat{X}) - \beta I(Y;\hat{X})$$

$$\mathcal{L}'^* = \max_{p(\hat{X}|X)} [\mathcal{L}'(\hat{X})] \qquad \mathcal{L}'(\hat{X}) = I(Y;\hat{X}) - \beta' I(X;\hat{X})$$

bigger β (smaller β') allows more complex representations

Information Bottleneck (IB)

Consider an information processing system that receives as input the signal X and tries to predict a target signal Y. We want to process X to get a compressed representation of the input $\hat{X} = f(X)$ (the "bottleneck"), which is then used to predict Y.

$\hat{X}$ is sufficient for predicting Y if it contains all the information that X encodes about Z, i.e. $I(Y;\hat{X}) = I(X;\hat{X})$.

$\hat{X}$ is minimal-sufficient if it is sufficient for Y and does not contain any extraneous information about X which does not help in predicting Y, i.e. $I(X;\hat{X}) \leq I(X;\hat{X}')$ for any other sufficient representation $\hat{X}'$.

The information bottleneck objective tries to strike a balance in achieving max compression (small complexity) while retaining as much relevant information (high accuracy) as possible

bigger β allows more complex representations

$$\text{minimize } \mathcal{L}(\hat{X}) = I(X;\hat{X}) - \beta I(Y;\hat{X})$$

$$\text{maximize } \mathcal{L}(\hat{X}) = I(Y;\hat{X}) - \beta' I(X;\hat{X})$$

bigger $\beta' = 1/\beta$ penalizes more complex representations