AICC 2 - Bixio Rimoldi

Alp Ozen

Spring 2019

1 Week 2

1.1 Entropy

We begin by defining **entropy** as Shannon put it:

Definition 1. ***Entropy***

Note that this definition assumes base 2 aka. binary.

$$H(s) = -\sum_{s\in A} p(s)\log_2 p(s) \ \ \textit{A being our alpahabet aka. sample space}$$

and thus an equivalent definition is:

$$H(s) = E[-\log_2 p(s)]$$

And similarly, Shannon defines information as:

$$-\log_2 p(s)$$

For a random distribution we get:

Example 1.1.

$$\forall x \in A \ p(x) = \frac{1}{|A|}, \ -\log_2 p(s) = \log_2 |A|$$

Hence the entropy function $H(s) = E[\log_2 |A|] = \underbrace{\log_2 |A|}_{\textit{do the algebra}}$

And now we present the **information theory inequality**

Definition 2. ***IT inequality***

$$\log_b r \leq (r-1)\log_b(e)$$

Proof. Given that

$$\ln(r) \leq (r-1)$$

and that

$$ln(r) = \frac{\log_b(r)}{\log_b(e)}$$

we are done. □

And now we present the Entropy bound theorem:

Theorem 1.1.

$$S \in A \ 0 \leq H(S) \leq \log |A|$$

Proof. We only show the RHS as the LHS is more or less trivial. Our goal is to show:

$$\begin{aligned} \text{need to reach} H(s) - \log|A| &\leq 0 \\ E[-\log p(s)] - \log|A| & \\ &= E[\log\frac{1}{p(s)|A|}] \\ &= \sum_{s\in A} p(s)(\log\frac{1}{p(s)|A|}) \\ &\underbrace{\leq log(e)\sum[\frac{1}{|A|} - p(s)]}_{\text{using IT ineq.}} = 0 \end{aligned}$$

□

1.2 Source coding

A code is said to have a prefix if:

Definition 3. ***Prefix of a code***

For some sequence of characters $a_1 a_2 \dots a_n$ and $b_1 b_2 \dots b_m$ with $n \leq m$ we have $a_1 a_2 \dots a_n = b_1 b_2 \dots b_n$

A **prefix free code** also known as **instantaneouss code** is one that has no prefixes. And now we come to the important **Kraft-McMillan** result:

Theorem 1.2. *If a $D-ary$ code is uniquely decodable, then it satisfies:*

$$D^{-l_1} + \dots + D^{-l_m} \leq 1$$

Note that there are non-instantaneous codes that stil satisfy this inequality. By the same token, by the contrapositive, we have that if a code does not satisfy the inequality, then there exists no prefix-free version of it.

We now define the **average codeword length**

Definition 4. ***average codeword length***

$$L(S, R) = \sum_{s \in A} p_S(s) L(R(s)) \text{ where } L \text{ represents length}$$

Given this definition, another important result is:

Theorem 1.3. ***Lower bound and Upper bound for average optimal codeword length***

$$H_d(S) \leq L(S, R) \leq H_d(S) + 1$$

This result becomes a useful tool once we realize the similarity in the definitions as below:

$$H(S) = -\sum p(s) \log p(s)$$
$$L(S, R) = \sum p(s) L(R(s))$$

Given this, *Shannon - Fano* realized that we may define a code of length $\lceil \log_D p(s) \rceil$ This satisfies the Kraft inequality hence we now have a method of obtaining uniquely decodable code.

But as it turns out, Huffman was the first to actually find out how one finds an optimal code. We list our alphabet with probability in increasing order. Then, if say we are working in base 2, we simply continuously combine the smallest probabilities and build our branches from them. Hence, as below, we have that a Huffman code isn't always unique:

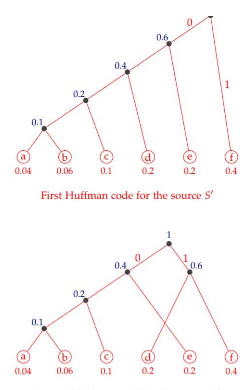

Figure 1: Huffman codes

2 Week 3

We now define conditional entropy(the intuition for this being, we want to mesure uncertainty given knowledge of something else) as follows:

Definition 5.

$$H(X|Y) := -\sum p(x|y) \log p(x|y)$$

And the law of **total probability**

Theorem 2.1. *Imagine we take a sample space with some subset A and cut it into 3 disjoint units B_i. Now we may describe the set A as $A = (B_1 \cap A) \cup (B_2 \cap A) \cup (B_3 \cap A)$ This is equivalent to now saying:*

$$p(A) = p(B_1 \cap A) + p(B_2 \cap A) + p(B_3 \cap A)$$

Which in terms of conditional probability is

$$p(A) = p(A|B_1)p(B_1) + p(A|B_2)p(B_2) + p(A|B_3)p(B_3)$$

And another useful theorem is:

Theorem 2.2.

$$H(s_1, s_2, \dots, s_n) \leq H(s_1) + H(s_2) + \dots + H(s_n)$$

with equality iff the s_i are independent.

A similar result now is the chain rule of conditional entropy.

Theorem 2.3. ***Conditional entropy chain rule***

$$H(S_1, S_2, \dots, S_n) = H(S_1) + H(S_2|S_1) + \dots + H(S_n|S_1, \dots, S_{n-1})$$

To clarify the notation used above, when we write $P_{X_1}, \dots, X_n(x_1, \dots, x_n)$ we interpret each comma as an intersection$\cap$

And we now introduce what it means for a source to be **regular**

Definition 6. ***Regular source***
A source is regular if

$$H(S) := \lim_{n\to\infty} H(S_n)$$
$$H^*(S) := \lim_{n\to\infty} H(S_n|S_1, S_2, \dots, S_{n-1})$$

exist and are finite.

Now it should be intuitively obvious that conditioning would reduce entropy. Lets prove it.

Theorem 2.4.

$$H(X|Y) \leq H(X)$$

Proof.

$$\begin{aligned} E(\log \frac{1}{p(X|Y)}) + E(\log p(X)) \\ = E(\log \frac{p(X)}{p(X|Y)}) \\ = E(\log \frac{p(X)p(Y)}{p(X|Y)p(Y)}) \\ \leq (\frac{p(X)p(Y)}{p(X \cap Y) - 1} \log(e) \leq 0 \end{aligned}$$

□

3 Week 4

We begin by making an important distinction in the definition of conditional entropy.

Definition 7. ***Conditional entropy given*** $Y = y$
We define this as:

$$H(X|Y = y) = -\sum_{x\in(\cdot|y)} P(x|y)\log P(x|y)$$

Given this definition, the more general definition of conditional entropy is:

Definition 8.

$$H(X|Y) = -\sum P(y)H(X|Y = y)$$

And we now introduce a **stationary source**

Definition 9. ***Stationary source***
A source is stationary if $\forall n, k$ *the blocks* $S_1, \dots, S_n$ *and* $S_{n+1}, \dots, S_k$ *have the same statistic that is:*

$$P_{s_1} = P_{s_i}$$

Theorem 3.1. *All stationary sources are regular.*

And now we come to another intuitive result:

Theorem 3.2.

$$H^*(S) = \lim_{N\to\infty} \frac{H(S^n)}{n}$$

The intuition for this result is that the entropy rate is inversely proportional to how much information we have. That is the more variables we know, the more we reduce entropy.

We now consider an instructive example

Example 3.1. *Suppose we are given two machines* M_1 *and* M_2 *that produce 3 bits.* M_1 *produces any number between 0 and 7 with equal chance and* M_2 *produces a number in range 0 to 3 with equal chance. We ask then, what is the probability distribution of the sequence* $s_1 \dots s_n$

Well noticing that this is equal to finding

$$P(S_1 \dots S_n) = P(S_1 \dots S_n|S_0)P(S_0) = P(S_1 \dots S_n|S_0)P(S_0 = M_1) + P(S_1 \dots S_n|S_0)P(S_0 = M_2)$$

We obtain:

$$P(S_1 \dots S_n) = \begin{cases} \frac{1}{8^n}\frac{1}{2} + \frac{1}{4^n}\frac{1}{2} if s_1 \dots s_n \in \{0, \dots, 3\}^n \\ \frac{1}{8^n}\frac{1}{2} \end{cases}$$

4 Week 5

This week, we introduce cryptography. We begin by listing the most common attack methods.

- **Chosen-plaintext attack**: The attacker is able to obtain a ciphertext for any arbitrary plaintext. Thus to obtain the key, one might encode every single letter.
- **Known-plaintext attack**: Attacker has access to both ciphertext and plaintext.
- **Ciphertext-only attack**: Attacker has access to a set of ciphertexts. A possible attack method is to use a frequency analysis.

We now introduce the vigenere cipher.

Definition 10. ***Vigenere's cipher*** *Vigenere cipher makes use of the Caesar cipher. Suppose we are given a 7 letter plaintext. The sender chooses another keyword of 7 letters. This we call the key. Then encryption is done using the lookup table below.*

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
B	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
C	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
D	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
E	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
F	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
G	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
H	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
I	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
J	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
K	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
L	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
M	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
N	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
O	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
P	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Q	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
R	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
S	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
T	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
U	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
V	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
W	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
X	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
Y	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
Z	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

Figure 2: Lookup table

As an example suppose our plaintext is "hello", an example key is "abcde". Thus the encrypted word would become "hffos"

Now we introduce a very strong type of secrecy.

Definition 11. ***Perfect secrecy*** *A cryptosystem has perfect secrecy if the plaintext and ciphertext are statistically independent.*

Given this perfect secrecy implies:

Theorem 4.1. *$H(T) \leq H(K)$*

And now we present an instructive example on how one would try to crack vigenere given we have the ciphertext and a portion of the plaintext.

Example 4.1. ***Cracking Vigenere***

Problem 5.1. Assume that you have intercepted the following ciphertext which was encrypted either using monoalphabetic substitution or using the Vigenère cipher:

`EBGRYCXGBGHITURSYNEAVCGBGRYV`

Also, you have managed to find out 4 letters of the plaintext message:

`T*****************U**I**I***`

Within this homework, we consider that the Vigenère cipher is designed for 26 characters, where each letter is assigned its position in the alphabet: $A \to 0, B \to 1, \ldots, Z \to 25$. The encryption is done by adding the (repeated) key with the plaintext and then taking modulo 26.

1. Can you tell if the message was encrypted with the Vigenère cipher or by means of monoalphabetic substitution?

Solution. *If monoalphabetic substitution was used, the letter* `I` *in the plaintext should always be replaced by the same letter in the ciphertext. Since letter* `I` *is encrypted as* `C` *the first time and as* `G` *the second time, the encryption scheme cannot be monoalphabetic substitution. Hence, we can conclude that the plaintext was encrypted with the Vigenère cipher.*

2. Using the previous question, can you find the key and the plaintext?
Hint: the key and plaintext consist of English words.

Solution. *We first "subtract" the known plaintext letters from the encrypted message to find parts of the key:*

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28
	E	B	G	R	Y	C	X	G	B	G	H	I	T	U	R	S	Y	N	E	A	V	C	G	B	G	R	Y	V
$\ominus$	T	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	U	*	*	I	*	*	I	*	*	*
	L	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	K	*	*	U	*	*	Y	*	*	*

Here, the operation $\ominus$ is defined as subtraction modulo 26 by assigning each letter its position in the alphabet ($A \to 0, B \to 1, \ldots$).
The next step is to determine the length of the key. The key length cannot divide 18, 21 or 24, otherwise the initial `L` *should be repeated on position 19, 22 and 25 respectively. Thus, the key cannot be of length 1, 2, 3, 4, 6, 7, 8, 9, 12, 18, 21, 24. Knowing this, we start by looking for a valid key of length 5:*

	E	B	G	R	Y	C	X	G	B	G	H	I	T	U	R	S	Y	N	E	A	V	C	G	B	G	R	Y	V
$\ominus$	T	H	*	H	A	R	D	*	R	I	W	O	*	K	T	H	E	*	U	C	K	I	*	R	I	G	E	*
	L	U	*	K	Y	L	U	*	K	Y	L	U	*	K	Y	L	U	*	K	Y	L	U	*	K	Y	L	U	*

Knowing that the key is an English word, it is easy to guess that the key is `LUCKY` *and the message is* `THE HARDER I WORK THE LUCKIER I GET`*.*

We know present a method used to verify the correctness of data sent.

Example 4.2. *Suppose we want to send an IBAN number over the net. It is possible that for some reason, during transmission, two digits of the credit card get flipped. We need a system to verify the correctness of information sent. Here's how we proceed:*

- *concatenate 00 to the end of the IBAN number.*
- *mod out by 97, fix this as a.*
- *now our key k is found as $98 - a = k$*
- *concatenate k to the end of the IBAN number removing the 00 and now send it to the recipient*
- *receiver mods out by 97, if result is congruent to 1, success.*

Here is why this works: Given $x \equiv a \mod 97$

$$x' \equiv x + (98 - a) \mod 97$$
$$x' \equiv x + 98 - x \mod 97$$
$$x' \equiv 98 \mod 97 \equiv 1 \mod 97$$

Yet another proof is to show the following implication:

$$100n_{97} - 100\tilde{n}_{97} \equiv 0_{97} \implies a = b \textit{where a,b are changed digits}$$

Now our above equality simplifies to:

$$10^{k+1}a + 10^k b - 10^{k+1}b - 10^k a = 10^k(9a - 9b)$$

Now since all expressions in the above equations have an inverse as we are in base 97, we get:

$$(a - b)_{97} \equiv 0_{97}$$

which is only the case if $a = b$

A neat modulo trick is presented below.

Theorem 4.2. *For some number* $a = d_n \ldots d_1$ *in base 10 we have that:*

$$a \bmod 11 = \sum_i^k 10^i d_i \bmod 11 = \underbrace{\sum_i^k -1^i d_i}_{10 \equiv -1 \bmod 11} \mod 11$$

5 Week 6

We present some definitions and theorems which we later use to define certain structures in $\mathbb{R}$.

Definition 12. ***Ring*** *A ring is a triple* (R, α, μ) *such that the following hold:*

R, α *known as the 'addition' on R is an abelian group*

R, μ *is both associative and right-left distributive over addition*

If it is also the case that R/e_α *is an abelian group, our ring is called a commutative ring with unity*

Theorem 5.1. *The inverse of an element under any binary relation is unique.*

Proof. Let $ab = e$ and also $ac = e$. Then $abb = acb$ which simplifies to $eb = ec = b = c$ □

Theorem 5.2. *Within* $\mathbb{Z}/m\mathbb{Z}$ *the following theorems are equivalent:*

$\forall a \in \mathbb{Z}, \exists a^{-1}$

$f : \forall a \in \mathbb{Z} \to \forall a \in \mathbb{Z}$ *is a bijection*

$ax = b$ *has a unique solution*

Theorem 5.3. *Some* $[a]_m$ *has a multiplicative inverse iff* $gcd(a, m) = 1$

Given the above, something that easily follows is:

Theorem 5.4. *If p is prime, then all* $a \in \mathbb{Z}/p\mathbb{Z}$ *have a multiplicative inverse*

Remark 5.1. ***Useful facts about the gcd***

$$gcd(a,b) = gcd(b, a-kb) = gcd(\pm a, \pm b)$$

We now present the Euclidian algorithm used to find $gcd(a,b)$ in pseudocode and explain why it works.

```
if(a<b) return:
gcd(b,a);
else if(b=0) return:
a;
else return:
gcd(b,a\% b);
```

Theorem 5.5. ***Bezout's theorem***

$$gcd(a,b) = au + bv \;\; a,b \in \mathbb{Z}$$

We now ask given the above theorem, how we may find the u and v. Well let's take an example:

Example 5.1. *Find the u, v for $gcd(56,15)$. By normal Euclidian algorithm, it turns out that $gcd(56,15) = 1$ Our steps in finding this are:*

$$\begin{aligned}
56 &= 15(3) + 11 \\
15 &= 11(1) + 4 \\
11 &= 4(2) + 3 \\
4 &= 3(1) + 1 \\
3 &= 1(3) + 0
\end{aligned}$$

Now notice how we from the above already have:

$$56 - 15(3) = 11 \;\; 4 - 3(1) = 1$$

Hence all that's left to do now is make $4 - 3(1) = 1$ top and work our way down.

$$\begin{aligned}
4 - 3(1) &= 1 \\
4 - 11 + 4(2) \\
4(3) - 11 \\
(15 - 11(1))3 - 11 \\
15(3) - 11(4) \\
15(3) - (56 - 15(3))4 \\
(-4)56 + (12)15
\end{aligned}$$

Lemma 5.6.

$$gcd(a,m) = 1 \iff \exists u, v \;\; 1 = au + mv$$

We now explore some notation relating to modular classes:

- $[a]_m :=$ congruence class mod$m$
- $\mathbb{Z}/m\mathbb{Z} :=$ all congruence classes mod$m$
- The structure $< \mathbb{R}/m\mathbb{Z}, +, \cdot >$ is an abelian ring
- By $\mathbb{Z}/m\mathbb{Z}^*$ we denote the set where an inverse exists.

Definition 13. ***Totient function*** *$\phi(x) : \mathbb{Z} \to \mathbb{N}$ computes the number of integers that are relatively prime to x.*

Lemma 5.7. *When p is prime, $\phi(p) = p - 1$*

Remark 5.2. *The cardinality of $\mathbb{Z}/m\mathbb{Z}^*$ is $\phi(m)$*

Remark 5.3.

$$\phi(p^k) = p^k - p^{k-1}$$

for some prime p

Yet another super useful result that appears in RSA encryption is the following:

Remark 5.4. *Let p and q be prime numbers, we ask what is $\phi(pq)$*

$$\phi(pq) = pq - (p + q - 1) = (p - 1)(q - 1) = \phi(p)\phi(q)$$

which follows from the fact that the numbers relatively prime to pq are:

$$\begin{cases} p, 2p, \dots, qp \\ q, 2q, \dots, (p-1)q \end{cases}$$

Finally we define perhaps one of the most fundemental mathematical structures, that is an isomorphism.

Definition 14. ***Isomorpishm***

Let $(H_1, \cdot_1)$ and $(H_2, \cdot_2)$ be two structures. An isomorphism ϕ from H_1 to H_2 is bijection such that:

$$\forall a, b \in H_1 \ \phi(a \cdot_1 b) = \phi(a) \cdot_2 \phi(b)$$

Now isomorphisms are useful because we know that if $(H_1, \star)$ is an abelian group and there exists an isomorphism from to $(H_2, \star_2)$, then so is

$$(H_2, \star_2)$$

an abelian group.

Now perhaps a mind blowing isomorphism is between the structures $([0, +\infty]\cdot)$ and $(\mathbb{R}, +)$ The isomorphism between the two is simply any mapping $f(x) = \log_b(x)$

Now we present a theorem about finite groups:

Theorem 5.8. *Let G be a finite group. Then $\forall a \in G$, $\exists k \in \mathbb{Z}_+$ such that $a^k = e$*

Proof. Now since G is a finite group, we have that for some $i < j$, $a^i = a^j$. Now $a^{-i}a^i = a^{-i}a^j = a^{j-i} = a^k$ □

And now a theorem about the order of elements in a group.

Theorem 5.9. *Two sets are isomorphic iff. they have the same order set.*

And now a grand result from the early developments of group theory.

Theorem 5.10. ***Lagrange's theorem*** *Let G be a finite abelian group of cardinality n. Then the order of each elements of G divides n.*

Proof. Let $H \leqslant G$. Now the cosets of H form a partition of G as the coset is an equivalence relation. Also note that H as a subgroup is of the form $\{a, a^2, \dots, a^k\}$ where k is the order of a. Hence we have that $|H| = k$. Now we have that G is a partition of equivalence classes all of size k that is to say:

$$n = kq \text{ q being number of equivalence classes}$$

□

We now present a corollary of Lagrange's theorem that will prove useful when studying RSA.

Corollary 1. ***Euler's theorem***

Let $m > 1$. $\forall a \in (\mathbb{Z}/mZ^*)$ we have:

$$a^{\phi(m)} = [1]_m$$

Now a result following Euler's theorem is that if we apply the same reasoning to some $(\mathbb{Z}/pZ^*)$ where p is prime then $\phi(p) = p - 1$ hence $a^{p-1} = e$ and $a^p = a$

Theorem 5.11. ***Chinese remainder theorem*** *Let m_1, m_2 be co-prime integers. We define a mapping $\phi : Z/m_1m_2^* \to Z/m_1Z/m_2$ Then it is the case the the mapping is a bijection and an isomoprhism with respect to $+$ and $\cdot$, where $\phi : (a)_{m_1m_2} \to (a)_{m_1}(a)_{m_2}$*

Let us now expand on our understanding of groups.

Definition 15. ***Cyclic group*** *A group G where $\exists g \in G$ such that $\forall h \in G, \exists i \in \mathbb{Z}$ $h = g^i$ is called a cyclic group.*

It turns out that all cyclic groups of the same order are isomorphic. As a quick sketch of proof, consider that we define a map as $\psi(a^i) = b^i$. Then simply observe that

$$\psi(ab) = \psi(g^i g^j) = \psi(g^{i+j}) = h^{i+j} = \psi(a)\psi(b)$$

A concrete example of an isomorphism between groups is the map from $(\mathbb{Z}/m\mathbb{Z}, +) \to (G, \star)$ where G is cyclic. We define it as $[i]_n \to b^i$ n being the order of G

6 Week 7

We now consider what's called the discrete exponent problem. Suppose we have a finite group $G, \star$ of size N with generator g. We are asked to find the α such that $h = g^\alpha$. Doing this the bruteforce way would be to construct all elements g^i with $i \leq \alpha$ at the cost of $\alpha - 1$ which is $O(n)$. But we can do better. We pick some k and compute all g^k. We also then compute bg^{-i}. Now we suppose some common element such that $g^n \equiv hg^{-mi} \bmod n$ Then we have that $a^{mi+n} = h$ which solves the problem. This will take us $O(\frac{N}{B})$ steps.

We are now ready to present the RSA encryption scheme.

Definition 16. ***RSA scheme*** *Let $K_{pr} = \{d, m\}$ be the private key and $K_p = \{e, m\}$ our public key. Upon Bob wanting to send Alice a message, Alice sends K_p to Bob which Bob uses the encrpyt his message and send the ciphertext back to Alice. But before, to come up with e, m Alice must pick to large primes p and q and m must a multiple of $p - 1$ and $q - 1$. We may do this either by taking $lcm(p - 1, q - 1)$ or by taking $\phi(pq)$. Now here is how we derive the rest of the scheme: We start of with Euler's theorem which states for m, n are coprime:*

$$m^{\phi(n)} \equiv 1 \bmod n$$

Now what suits are need is a relation of the form:

$$m^{k\phi(n)+1} \equiv m \bmod n$$

Hence we have that:

$$ed = k\phi(n) + 1$$

Now the reason finding a d that satisfies this is equivalent to finding an integer(guaranteed to exist through Bezout because we selected e coprime to $\phi(n)$) that satisfies:

$$d = \frac{k\phi(n) + 1}{e}$$

But for someone who doesn't know our p and q this is extremely hard since they must factor n into two primes to calculate $\phi(n)$.

7 Week 8-12

We now move our focus to the physical transmission of data through channels. We have two types of channels: **error channels** and **erasure channels**. Erasures can occur for instance due to destructive interference or some networking error. Now let's introduce some terminology. A **code block** is some $C \subseteq A^n$. The **code rate** is defined as $\frac{k}{n}$ with $k = \log_{|A|} |C|$. The question at this stage is, what is the most efficient way to choose an optimal code correcting encoding. We define the **Hamming distance** between two source codes as:

$$d(x, y) = |\{i \in \{1, \ldots, n\} \; x_i \neq y_i\}|$$

Hence the minimal distance is that between some $x, y \in C$ such that $d(x, y)$ is minimum. To develop intuition for this, a minimum Hamming distance implies the maximum number of errors we can detect. If for instance some encoding has a minimum distance of 4, then we can detect an error of size 3 but not of 5. Thus the larger the distance, the better. Now two theorems.

Theorem 7.1. *An encoding is able to correct an erasure if an only if its weight is* $p < d_{min}$

Theorem 7.2. *An encoding is able to correct an error if and only if its weight is* $p < \frac{d_{min}}{2}$

We now ask how many searches do we need in general to determine d_{min} for some encoding. Taking the brute-force approach we have that for the first code we do $m-1$ for the second $m-2$... checks which means our search is $O(\frac{m^2-m}{2})$.

Another question to be asked is, for our code block what properties would we like? First of all, d_{min} should be as large as possible to allow for maximum error correction. Similarly, n(length of code block) should be as small as possible for minimal data storage. Finally, $|C|$ should be as large as possible because then each code block will be carrying maximal information. In answer to this, we have the Singleton bound theorem which states:

Theorem 7.3. *For some code block C of length n and $k = \log_{|A|} |C|$ we have:*

$$d_{min}(C) - 1 \leq n - k$$

At this stage to further our discussion of how we determine d_{min} with a complexity better than a quadratic one we must introduce some algebraic structures.

Definition 17. ***Field***
A field is a triplet $(K, +, \cdot)$ with the following property:

$$(K, +) \text{ and } (K/\{0\}, \cdot) \text{ are abelian groups}$$

Some examples of non-finite fields are:

$$((\mathbb{R}, \mathbb{Q}, \mathbb{C}), +, \cdot)$$

Definition 18. ***Vector space***
A non-empty set V is a vector space over a field F if:

1. *There exists a binary operation $+$ where $\forall a, b \in V$, $a + b \in V$*
2. *There is a mixed operation called scalar multiplication $(\cdot)$ where $\forall c \in F$ and $\forall v \in V$ we have $c \cdot v \in V$*
3. *and the below hold:*
   - *$(V, +)$ is a commutative group*
   - *associativity: $(ab)v = a(bv)$*
   - *identity: $1v = v$*
   - *scalar multiplication is right,left distributive over addition*

Our first theorem about fields states the following:

Theorem 7.4. *The order of* 1*(multiplicative identity) with respect to* $+$ *is a prime number. This specific order is known as the characteristic of a field.*

Proof. Suppose that the order of 1 is m and m is not prime letting $m = ab$. Then we have:

$$\underbrace{1 + \ldots + 1}_{m} = 0$$

$$\underbrace{(1 + \ldots + 1)}_{a}\underbrace{(1 + \ldots + 1)}_{b} = 0$$

Now we have that either a or b must be 0. But this then implies that $m = 0$ and m is supposed to be a smallest non-zero integer. Hence we have a contradiction.

Now, in analogue to groups, two fields are said to be the 'same' if there is an isomorphism between two fields. Namely a bijective map that respects both of $(+, \cdot)$. □

Theorem 7.5. *1. The cardinality of a finite field is an integer power of its characteristic.*

2. All finite fields of the same cardinality are isomorphic.

3. For every prime p *and positive integer* m*, there exists a finite field of cardinality* p^m

Thus we have that for each p, m there exists exactly one field of cardinality p^m. Otherwise said, all finite fields of cardinality p^m are isomorphic to each other. To introduce some notation, a field of cardinality p^m is denoted $\mathbb{F}_{p^m}$ or $\mathbb{GF}(p^m)$

Example 7.1.

$$F_2 = (\mathbb{Z}/2\mathbb{Z}, +, \cdot)$$

$$F_3 = (\mathbb{Z}/3\mathbb{Z}, +, \cdot)$$

But we quickly realize that we can't further state that $F_4 = (\mathbb{Z}/4\mathbb{Z}, +, \cdot)$ since 2 would not have a multiplicative inverse. So let's construct F_4, it looks like the below:

$+$	0	1	a	b
0	0	1	a	b
1	1	0	b	a
a	a	b	0	1
b	b	a	1	0

The way we constructed the above is as follows:

- We know the characteristic is 2 so along the right diagonal all entries are 0.

Similarly, we can also construct the multiplication table which looks as follows:

$\cdot$	0	1	a	b
0	0	0	0	0
1	0	1	a	b
a	0	a	b	1
b	0	b	1	a

It is now time to use our understanding of vector spaces and fields to build so-called *linear codes.*

Definition 19. ***Linear code*** *A code* $C \subseteq \mathbb{F}^n$ *is linear* C *is a subspace of* $\mathbb{F}^n$*. Given this definition of* C *as a subspace, we add the note that the cardinality of* C *is precisely* $|F|^k$ *for the reason that* k *is our dimension hence we have* k *many choices to make out of* $|F|$ *many possibilities.*

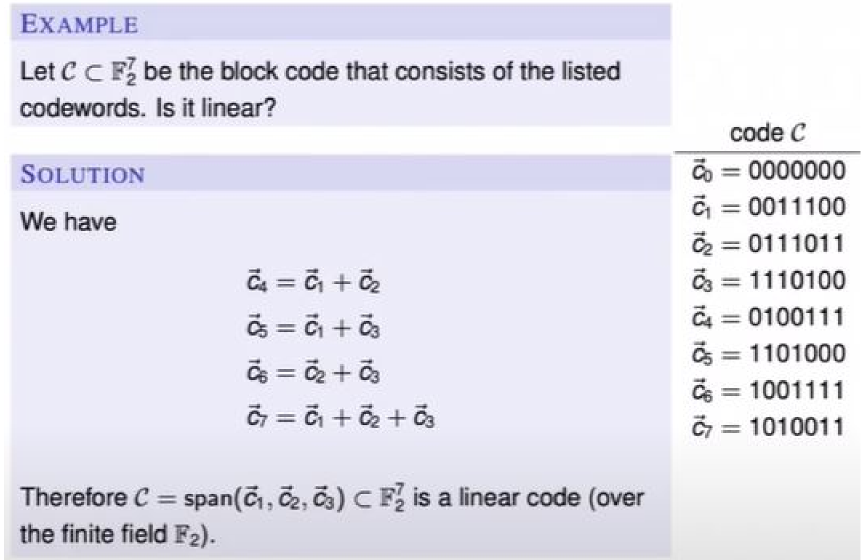

Figure 3: Example of a linear code

Theorem 7.6. *If V is an n dimensional vector space over some finite field F, then V is finite and $card(V) = card(K)^n$*

Using the above theorem, we have a means to check if a code C is a linear code over F_{p^m}. We simply verify that $card(C) \neq p^{mk}$.

We now present a theorem about minimal distance. Defining **Hamming weight** denoted $\omega(\vec{x}) = d(\vec{0}, \vec{x})$ we have that:

$$d_{min}(C) = \min_{\vec{x} \in C, \vec{x} \neq \vec{0}} \omega(\vec{x})$$

We now come to discuss *generator matrices*. A generator matrix has as it rows the basis vectors for some code C. Clearly bases are not unique hence we can have more than one generator matrix, but precisely how many? To see this, let $(\vec{c_1}, \ldots, \vec{c_k})$ be a basis. How many choices of $\vec{c_1}$ are there? Well given that $|F| = q$ we have $q^k - 1$ choices where we subtract a one to account for the zero vector. Then for $\vec{c_2}$ we have $q^k - q$ many choices. This time we subtract q because there are q many scalar multiples of $\vec{c_1}$ that we must exclude. Hence the total number of generator matrices for a code C turns out to be: $(q^k - 1)(q^k - q) \ldots (q^k - q^{k-1})$.
We say that a generator matrix is in *systematic form* if it is of form:

$$G_s = [\underbrace{I_k}_{kxk}, \underbrace{P}_{kxn-k}]$$

Another way to view a systematic generator matrix is to say that it is in *reduced echelon form.* The advantage of a systematic generator matrix is that our encoding mapping is as follows:

$$\vec{u} \in \mathbb{F}^k \to \vec{c} = (\vec{u}, \vec{u}P)$$

which means that the first k entries of the output vector correspond exactly to our information bits hence inverting the encoding function(which a decoder does) is as simple as reading the first k entries of the codeword.

We now define a parity check matrix as: $[-P^T|I]$

Theorem 7.7. *Let $C \subseteq \mathbb{F}^n$ be a linear code and H a parity check matrix. Then d_min is the smallest number of linearly dependent columns of H.*

8 Week 13

The main aim of this week is to describe a decoder for some linear code that is able to decide if someoutput vector $\vec{y}$ is a codeword. To do this, we use the fact that each codeword is a linear combination of the basis vectors which gives us a set of linear equation. Hence we formulate this to say that some $\vec{y}$ is a codeword iff $\vec{y}H^T = \vec{0}$ where H is called a parity check matrix. Now because our code is linear, suppose that the original message $\vec{x}$ becomes mutated by some $\vec{e}$. Now by definition we have $\vec{x}H^T = \vec{0}$. This means that $(\vec{e} + \vec{x})H^T = \vec{y}H^T = \vec{e}H^T$. Hence to recover the original message, we store the *syndrome* of each code (ofcourse provided it is non-zero) in a look-up table. Using this idea, we construct the so-called **Standard-array decomposition**. To construct this, we use the notion of a coset from group theory. For some group G a coset is defined as:

$$a \sim b \equiv \exists h \in H \leqslant G \ b = h \star a$$

Thus when we pick a subgroup of G this defines an equivalence relation where the equivalence classes are called cosets.

0	c_2	c_3	$\cdots$	c_M
e_2	$c_2 + e_2$	$c_3 + e_2$	$\cdots$	$c_M + e_2$
e_3	$c_2 + e_3$	$c_3 + e_3$	$\cdots$	$c_M + e_3$
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
e_N	$c_2 + e_N$	$c_3 + e_N$	$\cdots$	$c_M + e_N$

Now as we see from the figure, the subgroup is formed by our Code C. Then each row will have the same syndrome because we are adding the same error. Now a **coset leader** is one element from each row ie. coset that represents the syndrome. We choose the entry with the smallest weight because it is the most plausible error. Now let's see a real example:

000000	001110	010101	011011	100011	101101	110110	111000
000001	001111	010100	011010	100010	101100	110111	111001
000010	001100	010111	011001	100001	101111	110100	111010
000100	001010	010001	011111	100111	101001	110010	111100
001000	000110	011101	010011	101011	100101	111110	110000
010000	011110	000101	001011	110011	111101	100110	101000
100000	101110	110101	111011	000011	001101	010110	011000
001001	000111	011100	010010	101010	100100	111111	110001

Given that the decoder has the output $\vec{y}$ and the syndrome for each error vector it is trivial to correct the error. Now it is impractical to decode by constructing the whole coset table. Instead we do the following:

1. Precompute all coset leaders and the corresponding syndrome
2. Find the syndrome of the received $\vec{y}$
3. Using our lookup table, we find the coset leader $\vec{t_i}$ for the found syndrome
4. Finally, the decoded message is $\vec{x} = \vec{y} - \vec{t_i}$

9 Week 14

We end the course with a discussion of Reed-Solomon codes. Here is how we generate a Reed-Solomon encoding:

1. We choose an alphabet as a finite field $\mathbb{K}$ with cardinality $\geq n$.
2. We choose n elements from the field $\mathbb{K}$ denoted $a_1, \ldots, a_n$
3. We list all sequences from K^k where k denote the codeword size. In total we would have $k^{|K|}$ many sequences.
4. The idea now is, we define a mapping through a polynomial from $K^k \to K^n$. The mapping is rule is defined as $y_1 + y_2 x^1 \ldots + y_k x^k$ where $y_k = u_k$ and the output which is of dimension n has its ith given by the polynomial evaluated at a_i.

$\vec{u}$	$\mathbb{P}_{\vec{u}}(X)$	$\vec{x}$
00	0	00000
01	X	01234
02	$2X$	02413
03	$3X$	03142
04	$4X$	04321
10	1	11111
11	$1+X$	12340
12	$1+2X$	13024
13	$1+3X$	14203
14	$1+4X$	10432
20	2	22222
21	$2+X$	23401
22	$2+2X$	24130
23	$2+3X$	20314
24	$2+4X$	21043
30	3	33333
31	$3+X$	34012
32	$3+2X$	30241
33	$3+3X$	31420
34	$3+4X$	32104
40	4	44444
41	$4+X$	40123
42	$4+2X$	41302
43	$4+3X$	42031
44	$4+4X$	43210

With this we come to the end of the course, the notes are by no means exhaustive, they are merely meant to be a summary! We finally list some useful facts to aid with performance in the final exam :)

10 Useful facts by week

10.1 Week 11

10.2 Week 12

- For a finite field, the order of 1 with respect to + is called the **characteristic**
- The characteristic is a prime number
- All finite fields have cardinality p^m with p being the characteristic
- All finite fields of same cardinality are isomorphic
- For every p^m there exists a finite field of cardinality p^m
- $\mathbb{Z}/k\mathbb{Z}$ is a field only if k is prime

10.3 Week 13

- For some $G = [I_K|P]$ we have $H = [-P^T|I_{n-k}]$
- Let H be a parity-check matrix, then the minimum distance of a code C is the least integer d such that there are d linearly dependent columns of H.

10.4 Week 14

- A Reed-Solomon code with design parameters is a linear code with $d_{min} = n - k + 1$
- **Singleton bound:** Let C be a code with alphabet length q, minimum distance d and block length n then $|C| \leq q^{n-d+1}$
- A parity check matrix is $(n - k) \times n$ for the reason that n is the codeword size and k the dimension meaning we have $n - k$ linear equations with n many variables.

11 Useful links

Amazing YouTube playlist: YT Information Theory playlist