Classical Cipher - Part 2

Giulia Mauri

Politecnico di Milano

email: *giulia.mauri@polimi.it*
website: *http://home.deib.polimi.it/gmauri*

April 22, 2015

Overview

References:

[1] SAGE: http://sagemath.org
[2] PYTHON: http://docs.python.org

The Vigenère Cipher - Definition

Definition (Vigenère Cipher)

Let m a positive integer. Let $\mathcal{P} = \mathcal{C} = \mathcal{K} = (\mathbb{Z}_{26})^m$. For a key $K = (k_1, k_2, \ldots, k_m)$, we define

$$e_K(x_1, x_2, \ldots, x_m) = (x_1 + k_1, x_2 + k_2, \ldots, x_m + k_m)$$

and

$$d_K(y_1, y_2, \ldots, y_m) = (y_1 - k_1, y_2 - k_2, \ldots, y_m - k_m),$$

where all operations are performed $\mathbb{Z}_{26}$).

The Vigenère cipher associates each key K with an alphabetic string of length m, the keyword. The number of possible keywords of length m is 26^m, so an exhaustive key search would require a long time. The Vigenère cipher encrypts m alphabetic characters at a time: each plaintext element is equivalent to m alphabetic characters.

The Vigenère Cipher - Sage Implementation

Here is an example over the upper-case letters of the English alphabet.

```
sage:  AS = AlphabeticStrings()
sage:  V = VigenereCryptosystem(AS, 14); V
Vigenere cryptosystem on Free alphabetic string monoid on
A-Z of period 14
sage:  K = AS('ABCDEFGHIJKLMN')
sage:  x = V.encoding("THECATINTHEHAT")
sage:  y = V.enciphering(K, x); y
TIGFEYOUBQOSMG
sage:  V.deciphering(K, y)
THECATINTHEHAT
sage:  V.deciphering(K, y) == x
True
```

You can use the functions `random_key()`, `inverse_key()` also for the Vigenère cipher.

The Vigenère Cipher - Finding the Key Length

How we decrypt a message encrypted with the Vigenère Cipher?
The first step is to determine the keyword length, which we denote by m.
Then, the second step is to find the key itself.
Consider a ciphertext $\mathbf{y}$, put $\mathbf{y}$ above the same ciphertext but displaced by a certain number of places (the potential key length). Then, count the total number of coincidences between the two strings. Repeat the same process for different displacements. We would find a displacement to which corresponds the bigger number of coincidences: this is the length of the key.

Example

$\mathbf{y} = \quad VVHQWVVRHMUSGJG\ldots$
$\mathbf{y}_2 = VVHQWVVRHMUSGJGTH\ldots$
Mark orange each time a letter and the one below it are the same, and count the total number of coincidence. Repeating the process with different displacements gives us:

displacements	1	2	3	4	5	6
coincidences	14	15	14	16	24	11

The Vigenère Cipher - Finding the Key

Assume we already have determined that the key length is m.
For $i = 1$ to m, do the following:

1. Count the occurrences of the letters in positions i mod m, and put them in a vector: $\mathbf{V} = (0, 0, 7, 1, 1, 2, 9, \ldots)$
   Then, divide by the total number of letters counted, and obtain the vector $\mathbf{W} = (0, 0, .1045, .0149, .0299, \ldots)$.
   The letters in positions i mod m are all shifted by the same amount.
   Therefore, they represent a random sample of English letters.
   Their frequencies should approximate the vector $\mathbf{A}_j$, where $\mathbf{A}_j$ is the result of shifting $\mathbf{A}_0$ (the frequencies of English letters into a vector) by j spaces to the right.

The Vigenère Cipher - Finding the Key

The problem is to determine j. Recall that $\mathbf{A}_j \cdot \mathbf{A}_k$ is largest when $j = k$, and that $\mathbf{W}$ approximates A_j. If we compute $\mathbf{W} \cdot \mathbf{A}_k$ for $0 \leq k \leq 25$, the maximum value should occur when $k = j$.

2. For $j = 1$ to 25, compute $\mathbf{W} \cdot \mathbf{A}_j$.
3. Let $k_i = j_0$ give the maximum value of $\mathbf{W} \cdot \mathbf{A}_j$.

Here are the dot products: [.0250, .0391, .0713, .0388, ...]
The largest value is the third, which equals $\mathbf{W} \cdot A_2$. Thus, we guess that the first shift is 2, which corresponds to the key letter c.
Repeating the process for $i = 1$ to m, give us the key $\{k_1, \ldots, k_m\}$.

The Vigenère Cipher - Cryptanalysis

Exercise

Given the ciphertext obtained from a Vigenère Cipher stored in the file SnowWhiteEnc.txt

Decipher it using the method previously described.

Start reading the file:

Solution

```
sage: with open ("SnowWhiteEnc.txt", "r") as myfile:
sage:     data = myfile.read().replace('\n', '')
```

The Vigenère Cipher - Finding the Key Length

Solution

Find the length of the keyword used.

```
sage: AS = AlphabeticStrings()
sage: y = AS.encoding(data)
sage: coincidence = [0 for n in range(9)]# array of 9 elements
sage: for i in range (1, 10): # the key length ranges between 1 and 9
....:     for x in range(0, len(y)-i):
....:         if y[x] == y[x+i]:
....:             coincidence[i] += 1
sage: print coincidence
[159, 193, 178, 206, 334, 175, 191, 190, 179]
```

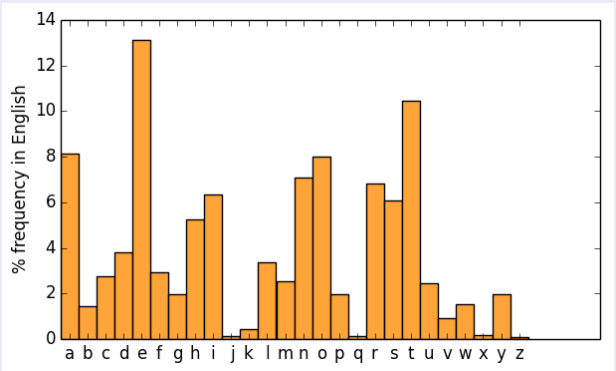
We have the most coincidences for a shift of 5. This is the best guess for the length of the key.

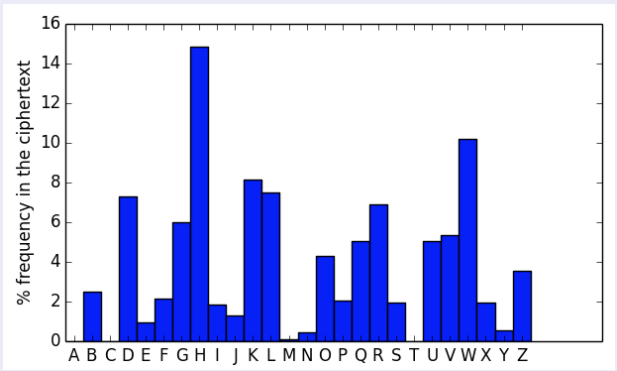
The Vigenère Cipher - Finding the Key

Once the length of the key is determined, we find the exact shifts.
In order to make things easier, we exploit some functions designed to work with the Vigenère cipher [1].

Solution

sage: load ("Fall2012Code.sage")
sage: y_str = str(y)
sage: Y = separate(y_str,0,5) # it creates a string Y that consists exactly of the characters of y in positions congruent to 0 mod 5.
sage: comparetoenglish(Y) # it finds the corresponding shift amount.





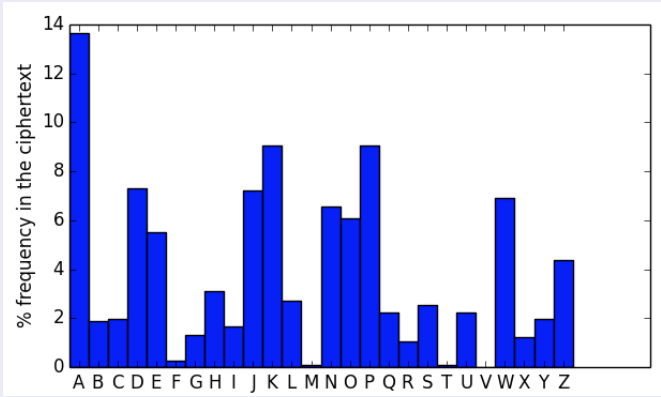
[1] http://www.math.uci.edu/~davis/Fall2012Code.sage

The Vigenère Cipher - Finding the Key

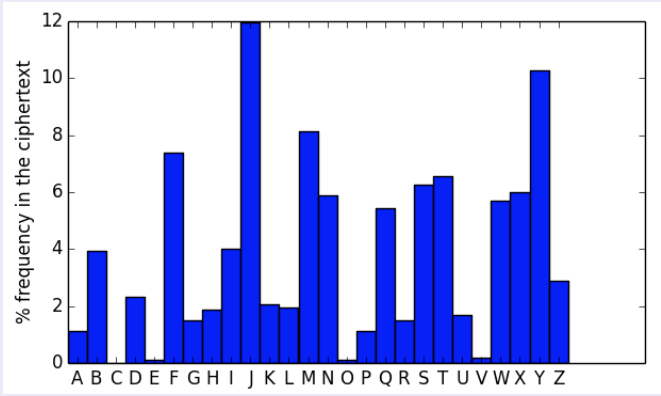
Once we have the 0-th shift amount, we need to find 4 more by repeating the previous steps:

Solution

sage: $Y = separate(y\_str,1,5)$
sage: $comparetoenglish(Y)$



sage: $Y = separate(y\_str,4,5)$
sage: $comparetoenglish(Y)$



The Vigenère Cipher - Finding the Key

At the end, we have the shifts amounts that are: [3,-4,0,17,5].
These shift corresponds to the key: K = "DWARF".
We check it by using two ways:

Solution

```
sage: Y = deciphervigenere(y_str, [3,-4,0,17,5])

sage: K = AS('DWARF')
sage: V.deciphering(K,y)

ONCEUPONATIMEINAGREATCASTLEAPRINCESDAUGHTER...
```

The Hill Cipher - Definition

Definition (Hill Cipher)

Let $m \geq 2$ be an integer. Let $\mathcal{P} = \mathcal{C} = (\mathbb{Z}_n)^m$ and let

$$\mathcal{K} = \{m \times m \quad \text{invertible matrices over} \quad \mathbb{Z}_n\}.$$

For a key K, we define

$$e_K(x) = xK$$

and

$$d_K(y) = yK^{-1}$$

where all operations are performed in $\mathbb{Z}_n$.

Note that the decryption is possible only if the matrix K has an inverse. A real matrix K has an inverse if and only if its determinant is non-zero. However, since we are working over $\mathbb{Z}_n$, the matrix K has an inverse modulo n if and only if gcd(det K, n) $= 1$.

The Hill Cipher - Sage Implementation

Here is an example over the upper-case letters of the English alphabet.

```
sage:  AS = AlphabeticStrings()
sage:  H = HillCryptosystem(AS, 3); H
Hill cryptosystem on Free alphabetic string monoid on A-Z of
block length 3
sage:  K = H.random_key()
sage:  x = H.encoding("Good day, mate!  How ya going?")
sage:  y = H.enciphering(K, x); y
OSMGPSSYELCGGAWKYWSOJ
sage:  H.deciphering(K, y)
GOODDAYMATEHOWYAGOING
sage:  H.deciphering(K, y) == x
True
```

You can use the functions `random_key()`, `inverse_key()` also for the Hill cipher.

The Hill Cipher - Implementation in Sage

We consider a Hill cipher operating in $\mathbb{Z}_{26}$.
We first define the appropriate finite field:

```
sage: R = Zmod(26)
```

Encrypt the message $x = \begin{pmatrix} 7 & 2 & 1 \\ 0 & 3 & 25 \\ 23 & 4 & 24 \end{pmatrix} \mod 26$,

by using the key $K = \begin{pmatrix} 1 & 2 & 3 \\ 0 & 1 & 4 \\ 5 & 6 & 0 \end{pmatrix} \mod 26$

```
sage: K = matrix(R, [[1,2,3], [0,1,4], [5,6,0]])
sage: x = matrix(R, [[7,2,1], [0,3,25], [23,4,24]])
```

The Hill Cipher - Implementation in Sage

We encrypt the plaintext x:

```
sage: y = x*K
sage: print y
[12 22 3]
[21 23 12]
[13 12 7]
```

To decrypt y, we need the inverse of K.

```
sage: y*K^(-1)
[7 2 1]
[0 3 25]
[23 4 24]
```

Supposing to know x and y, K can be computed as follows.

```
sage: x^(-1)*y
[1 2 3]
[0 1 4]
[5 6 0]
```