BRUTE-FORCING THE HILL CIPHER

BOSS Technical Note on Breaking Hill Ciphers with Force

Suppose we want to break a Hill cipher that uses $n \times n$ matrices. To use brute force, we could try all possible $26^{n \cdot n}$ matrices (and maybe rule out those that are not invertible), generate a candidate plaintex for each, and choose one with the highest trigram or tetragram fitness. For 2×2 matrices, this is not too difficult, since $26^4 = 456{,}976$. But the work involved in the attack grows like an exponential of a square as the size of the matrices increases. For 3×3 matrices, the number is 5,429,503,678,976, which already makes the attack infeasible. For 4×4 matrices, the number is 43,608,742,899,428,874,059,776, which would require that we check over 100,000 matrices every second from the beginning of the universe until last Tuesday. We need a better way.

The trick to improving the brute-force attack is to notice that from each block of ciphertext, one letter has been enciphered with the same *vector*. So, for example, if we pick out the first letter of each block, we know that it was enciphered with the vector made up of the entries on the top row of the matrix. We need to work with the inverse matrix (the deciphering matrix), because we do not know the contents of each block, yet. So rewrite that previous sentence to say that one letter in each block of the plaintext is found by deciphering with the same row of the inverse matrix. This means that if we try all possible values for the n entries on just that row alone, we may be able to recover every n^{th} letter. But those letters are not contiguous in the text, so we need to use monogram fitness (how well do the letter frequencies match English?) to choose the best set of deciphered letters. This reduces the attack complexity from an exponential of a square to the exponential of a linear function. There is a catch, however: when we find a good vector, we do not a priori know on which row of the inverse matrix it belongs. Therefore, the second stage of the attack is to sort the rows; this is equivalent to breaking a columnar transposition cipher. Also, because we are relying on monogram statistics from a set of n letters for each row, this attack requires a ciphertext that is at least about $100n$ letters in length.

As is customary, we look at an example to see how it works. Consider this plaintext from *The Old Man's Guide to Health and Longer Life* by John Hill:

> Old men's diseases are hard to cure; but they are easy to prevent. It must be a good natural fabric which has preserved itself so long; and the same strength may keep it much longer well, under good regulation. Moderate diet, and due exercise, are the best guardians of health in all: but in the advanced period here considered there are two great preservatives besides; these are Ease, and Cheerfulness: both are the natural offspring of health; and they will continue the blessing to which they owe their birth.

In blocks we have

> OLD MEN SDI SEA SES ARE HAR DTO CUR EBU TTH EYA REE ASY TOP REV ENT ITM UST BEA GOO DNA TUR ALF ABR ICW HIC HHA SPR ESE RVE DIT SEL FSO LON GAN DTH ESA MES TRE NGT HMA YKE EPI TMU CHL ONG ERW ELL UND ERG OOD REG ULA TIO NMO DER ATE DIE TAN DDU EEX ERC ISE ARE THE BES TGU ARD IAN SOF HEA LTH INA LLB UTI NTH EAD VAN CED PER IOD HER ECO NSI DER EDT HER EAR ETW OGR EAT PRE SER VAT IVE SBE SID EST HES EAR EEA SEA NDC HEE RFU LNE SSB OTH ARE THE NAT URA LOF FSP RIN GOF HEA LTH AND THE YWI LLC ONT INU ETH EBL ESS ING TOW HIC HTH EYO WET HEI RBI RTH

First, let us look at the encryption. Take this 3×3 matrix:

$$M = \begin{pmatrix} 226 & 7 \\ 7 & 4 & 6 \\ 12 & 12 & 4 \end{pmatrix}$$

When we encipher it, the first letter of each block in the ciphertext is found by finding the product of the vector on the top row of **M** with the column vector of plaintext letters. For the first block,

$$c_1 = (2267)\begin{pmatrix} p_1 \\ p_2 \\ p_3 \end{pmatrix} = (2267)\begin{pmatrix} 14 \\ 11 \\ 3 \end{pmatrix} = 5$$

where the arithmetic is done modulo 26, and so the letter is F. The first letter of the second block of the ciphertext is calculated with the same row vector from **M** but the next column vector from the plaintext and found to be P.

$$c_4 = (2267)\begin{pmatrix} p_4 \\ p_5 \\ p_6 \end{pmatrix} = (2267)\begin{pmatrix} 12 \\ 4 \\ 13 \end{pmatrix} = 15$$

What we are doing is enciphering the first slice of the ciphertext.

> F__ P__ C__ E__ A__ A__ N__ S__ X__ A__ J__ Y__ K__ . . .

The second row of the matrix is used to encipher the second slice, and the third row for the third slice. The resulting ciphertext is

> FER PWS CEV EMM AQC AOJ NVY SZB XOK AWJ JRZ YUU KDS QIW JTE ZBK NMX KWN FOM UXQ CAG OVX HDG XWB VCT EOC IPK OZN HCP QUG ITJ NLG DAQ EJY BDQ PQU VJP OWO YAI CRD NVU STS SYE AGV GPS HEJ MEN GUV XIL TCN YCB XQU YPO MCR SPA OPK BJG MWL MZK PDU QXZ NAG WEJ AWC AOJ UDT QBG WRM TIL HES VEM WNK PNH UEF DXL MEJ HBF FUQ HRS LWW FPU VAA LLC QQW IDC BJG FYN LLC ZAO SCX VQK NMK SPH TKE XBG SIF OYB XUK RGC SRA ZAO ISA EMM GLZ YLC YZN KXH RWY DIR AOJ UDT DXO WAX XHG LPW TVE RYY WNK PNH VSH UDT OSI KDJ ZEN EUR RQB PUB KAE KOT GJQ IPK FLL SAS RYW AJU UPH RDB

But we are here to break ciphers, so let us look at the decipherment using the same row-by-row method, and then a bit later we will attack it as if we did not know the matrix. The inverse matrix is

$$M^{-1} = \begin{pmatrix} 14 & 7 & 18 \\ 18 & 24 & 5 \\ 15 & 2 & 6 \end{pmatrix}$$

The first letter of the plaintext is calculated from the product of the first row of the inverse matrix and the first column (the first block) of ciphertext letters:

$$p_1 = (14718)\begin{pmatrix} c_1 \\ c_2 \\ c_3 \end{pmatrix} = (14718)\begin{pmatrix} 5 \\ 4 \\ 17 \end{pmatrix} = 14$$

As usual, all arithmetic is done modulo 26. The first letter is O, as we already knew. In the following equation, as *i* runs over the block number, we calculate the first letter of each plaintext block:

$$p_{3i+1} = (14718)\begin{pmatrix} c_{3i+1} \\ c_{3i+2} \\ c_{3i+3} \end{pmatrix}$$

This gives us one slice of the plaintext:

O__ M__ S__ S__ S__ A__ H__ D__ C__ E__ T__ E__ R__ . . .

If we collect all of the letters from this slice, we have

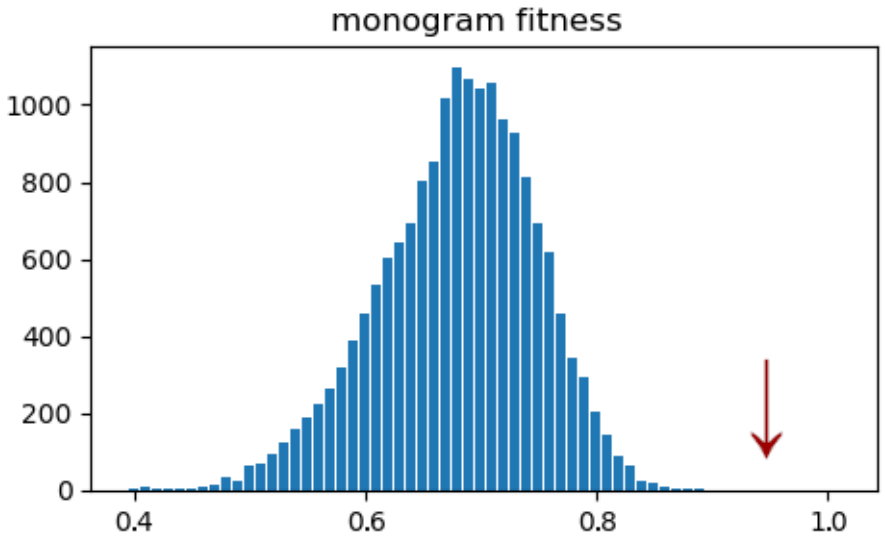
OMSSSAHDCETERATREIUBGDTAAIHHSERDSFLGDEMTNHYETCOEEUEORUTND
ADTDEEIATBTAISHLILUNEVCPIHENDEHEEOEPSVISSEHEESNHRLSOATNULFRG
HLATYLOIEEEITHHEWHRR

The monogram fitness using the cosine of the angle between 26-dimensional vectors for this collection of letters is 0.957, which is very close to 1. This fact is worth mentioning, because this is the kind of statistics that we can use to make sure that we have used a correct decryption vector. The other two slices of the plaintext are found in the same way, using the second and third rows of the inverse matrix.

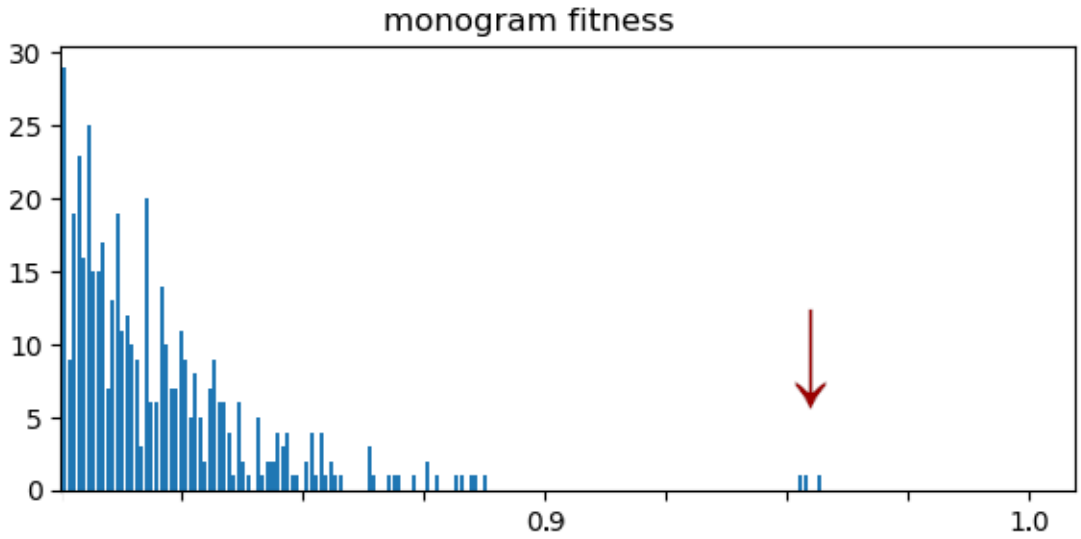
Now let's back up to the ciphertext and forget that we know the matrices. All we know is that we have a 3×3 Hill cipher. We will try all possible 3-vectors and use them to decipher a slice of the plaintext. Once we find three slices that have good fitness, we can be confident that we have found the three rows of the inverse matrix. For the 3×3 case, we set up three nested loops in our algorithm for this first stage of the attack:

```
set up storage for three 3-vectors and three values of fitness
for v1 = 0, ..., 25 do:
        for v2 = 0, ..., 25 do:
                for v3 = 0, ..., 25 do:
                        use row vector (v1 v2 v3) to decrypt a slice of plaintext
                        calculate the fitness of that slice
                        if that value is higher than the lowest of the stored values:
                                replace the corresponding stored 3-vector with (v1 v2 v3)
                                replace the lowest stored value of fitness with the new value
                        if the three stored values of fitness are good enough:
                                exit loops
output the three stored 3-vectors
```

To generate the following histogram, we let the algorithm run through *all* possible 3-vectors, and did not exit when we had found the three we wanted. The fitness function is the monogram fitness based on the cosine of the angle between the frequency vector of the text and the vector of English frequencies. Notice that at the high end of the graph, three data points stand out. These are the values of the statistic that correspond to the three best 3-vectors, which are (14, 7, 18), (15, 2, 6), and (18, 24, 5).



They are hard to see, so there is a close-up of the upper tail on the next page.



At this point, we have the three best 3-vectors, listed above, and their corresponding slices:

OMSSSAHDCETERATREIUBGDTAAIHHSERDSFLGDEMTNHYETCOEEUEORUTND
...
DNIASERORUHAEYPVTMTAOARFRWCAREETLONNHASETAEIULGWLDGDGAOO
R...
LEDEERATUBTYESOENTSEONULBCIHPSVIESOATSERGMKPMHNRLNROELIME...

We have to interleave them to make a sensible plaintext. We can do this by anagramming, and for small matrices, this is easy. Or, we can concatenate them and treat the whole like a columnar transposition ciphertext and attack it with the methods appropriate to that kind of cipher. Once we know the correct order of the slices, we can put the 3-vectors in the same order; the result is the decryption matrix.

Extension to affine Hill ciphers

We can extend the attack to the case of the affine Hill cipher. Recall that for each block of text, the ciphertext vector **C** is found by adding a shift vector **B** to the result of a Hill cipher with matrix **A** as follows:

$$\mathbf{C} = \mathbf{AP} + \mathbf{B}$$

Here **P** is the plaintext vector. Decipherment is the inverse:

$$\mathbf{P} = \mathbf{A}^{-1} (\mathbf{C} - \mathbf{B})$$

Now it looks as though we need to already know all the shifts in the vector **B** before we can apply the attack. But let us rewrite the equation by distributing $\mathbf{A}^{-1}$ and moving the constant term to the other side.

$$\mathbf{P} + \mathbf{A}^{-1}\mathbf{B} = \mathbf{A}^{-1}\mathbf{C}$$

The product $\mathbf{A}^{-1}\mathbf{B}$ is simply a new constant vector, which we can call $\mathbf{B}'$. Its entries do not depend on either the plaintext or the ciphertext. Our equation now looks like this:

$$\mathbf{P}' = \mathbf{P} + \mathbf{B}' = \mathbf{A}^{-1}\mathbf{C}$$

where $\mathbf{P}'$ is a "plaintext" vector that actually resembles a text enciphered with the Vigenère cipher, which you recall is a collection of Caesar shift ciphers. For each row of the matrix **A** there is one shift that can be thought of as having been applied before the Hill cipher or after it.

So our strategy is this: Apply the row-by-row attack from earlier using a method to recognize text that has letter frequencies that resemble English, but which has undergone a Caesar shift. What we do is to try each possible shift and measure the monogram fitness for each. This adds to the running time of the algorithm because because it adds another loop nested inside the others. We need to keep track of the best row vectors *and* the best values for their shifts that we find. Note that the index of coincidence will not work as a way to recognize shifted text without having to try all possible shifts; this is because some row vectors will give nonsense that has a high IoC, and we are led away from the true solution. To be clear, here is the algorithm for the first stage of the attack:

```
set up storage for three 3-vectors and three values of fitness and three shifts
for v1 = 0, ..., 25 do:
        for v2 = 0, ..., 25 do:
                for v3 = 0, ..., 25 do:
                        use row vector (v1 v2 v3) to decrypt a slice of plaintext
                        for shift = 0, ..., 25:
                                apply a Caesar shift to the slice
                                calculate the fitness of that slice
                                if that fitness is higher than the lowest of the stored values:
                                        replace the corresponding stored 3-vector with (v1 v2 v3)
                                        replace the lowest stored value of fitness with the new one
                                        replace the corresponding stored shift with the new one
                                if the three stored values of fitness are good enough:
                                        exit loops
output the three stored 3-vectors and their corresponding shifts
```

We take the best 3-vectors and use them to generate the three slices of "plaintext," then apply the three shifts. The three resulting slices can now be used, as we did before, as the ciphertext of a columnar transposition cipher, which we can break by brute-forcing its permutation.