CHECK DIGITS ERROR-CORRECTING CODES

APRIL 26, 2020

If you see a word "blackbourd", you can easily see that it is not a real word (at least, in English), so most likely it was a result of a typo. You can also probably guess how it must be corrected: the only actual English word which is close to it is "blackboard". This is possible because not every possible combination of letters is a word in English. However, you pay for it: English has about 100,000 words, but the number of combinations of 26 letters of up to 10 letters long is $(26)^{100}$, which is much larger. If we wanted to create the most economical language, we could use one where every combination of letters is a real word; this would allow us to cover 100,000 words using only four-letter words, but then we wouldn't be able to correct typos.

Same ideas apply in other situations: you want to be able to detect or correct errors in some information (words, id numbers,....) by adding extra symbols. This is called error-detecitng and error-correction codes.

Here is the general setup:

1. We consider "words" formed by sequences of symbols from some alphabet S. It can be usual 26 letters of English alphabet, or digits 0-9, or binary digits 0 and 1. We will denote by S^n the set of *all* such "words" — no restrictions. We assume that n is fixed: all words have same length.
2. We choose some subset $C \subset S^n$ of "valid" words (in computer science literature, this is usually called "codewords", and the subset C is called "code").

We say that C can **detect** k errors if, whenever we take a codeword and change at most k symbols in it, the result will not be a codeword (and thus, we know for certain that errors have been introduced).

We say that C can **correct** k errors if, whenever we take a codeword and change at most k symbols, we can *uniquely* recover the original word.

For example, in English, sometimes we can correct 2 letter errors (e.g. "blakkbourd"); sometimes we can't correct even one. For example, the word "plone" clearly contains an error (so we have detected one error), but we can't say if the original word was "clone" or "prone" - both of them differ from "plone" in one position.

1. Check digits: parity bit.

   Consider the "code" where words are 8-bit sequences $b_1 \dots b_8$, and "valid words" are those satisfying

   $$b_1 + b_2 + \cdots + b_8 \equiv 0 \mod 2$$

   Explain why this code can always detect a single-bit error. How many codewords does it have? (This is the oldest type of error-detecting code used in computer communication.)

2. Check digits: UPC.

   A Universal Product Code (UPC) is a numeric code assigned to virtually all products sold in stores; you usually see it as a bar code, which you scan at the cash register. The UPC is a 12-digit numeric code $a_1 \dots a_{12}$; the 12 digits have to satisfy the condition below:

   $$3a_1 + a_2 + 3a_3 + a_4 + \cdots + a_{12} \equiv 0 \mod 10$$

   (a) Which of the UPCs below are valid?
   036000291452; 892685001003;
   (b) Explain why UPC code can always detect a single digit error.
   (c) Is it true that UPC code can always detect a transposition of 2 adjacent digits (e.g. $47 \to 74$ or $38 \to 83$)?

3. Define *distance* $d(X, Y)$ between two words X, Y (each consisting of n symbols in the same alphabet S) as the number of positions in which they differ. For example, distance between words "card" and "cold" is 2. Show that it satisfies the usual rule: $d(X, Z) \le d(X, Y) + d(Y, Z)$.

4. Show that a code C (i.e. a collection of "valid" words in S^n):
   (a) Can detect any k errors if, for any two codewords $X, Y \in C$, we have $d(X, Y) > k$.
   (b) Can correct any k errors if, for any two codewords $X, Y \in C$, we have $d(X, Y) > 2k$

5. Suppose you need to constuct a binary code which consists of 4 codewords and can correct single bit error. Can you do it using codewords of length 5? of length 4?

Hint: think of your codewords consisting of 2 initial data bits and 2 or 3 additional "check bits":

$00 * **$

$01 * **$

$10 * **$

$11 * **$

***6.** What is the shortest code you can do that would have 8 codewords (equivalent to having 3 data bits) and correct a single bit error? what if you need 16 codewords (4 data bits)?

Hint: if you are stuck, google for information about Hamming $(7, 4)$ code.