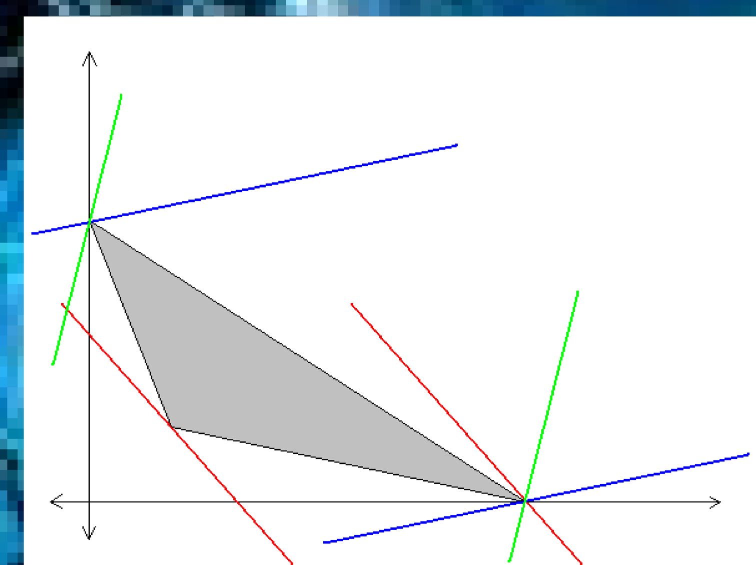
Compressive Sampling

The future of efficiency and signal processing

Researched by Willard Miller and Peter Mueller

1. Imagine this: you are given a feasible set of points satisfying three linear inequalities, so contained within the triangle shown. You want to find the solution which minimizes a linear expression (represented by one of the colored lines and all lines parallel to it). Instead of testing every possible feasible solution, you can simply test the vertices of the feasible triangle. For example, look at the linear expression represented by the blue lines. All points which lie on one of the lines have the same value. Also, as we decrease the y-intercept of one colored line, the value of the line either linearly decreases or increases. This means that any new blue line drawn through the triangle will not contain the minimum value we want because either the blue line above him or the blue line below it has a smaller value.

2. This property that the optimal solutions are found at the vertices of polytopes persists in all dimensions and is what makes linear programming and, more generally, convex optimization possible and is one main component of compressive sampling's efficiency.



3. Compressive sampling, also known as compressed sensing, is a process of taking a small sample of a signal and then re-computing the signal from the sample, in order to increase efficiency. The signal is a vector which contains some type of information; it can be data or sound or anything else that can be represented as numbers. The signal has dimensions nx1, it is compressed into a vector of dimensions mx1 (where m < n) by a sampling matrix. This new vector is called the sample and is smaller; hence, the sample takes up less space in storage and is quicker to transfer to someone else. This would of course be useless unless the original signal could be recreated from the sample. The theories behind compressive sampling, which were developed by Emmanuel Candès, Ronald DeVore, and Terrence Tao, do exactly this.

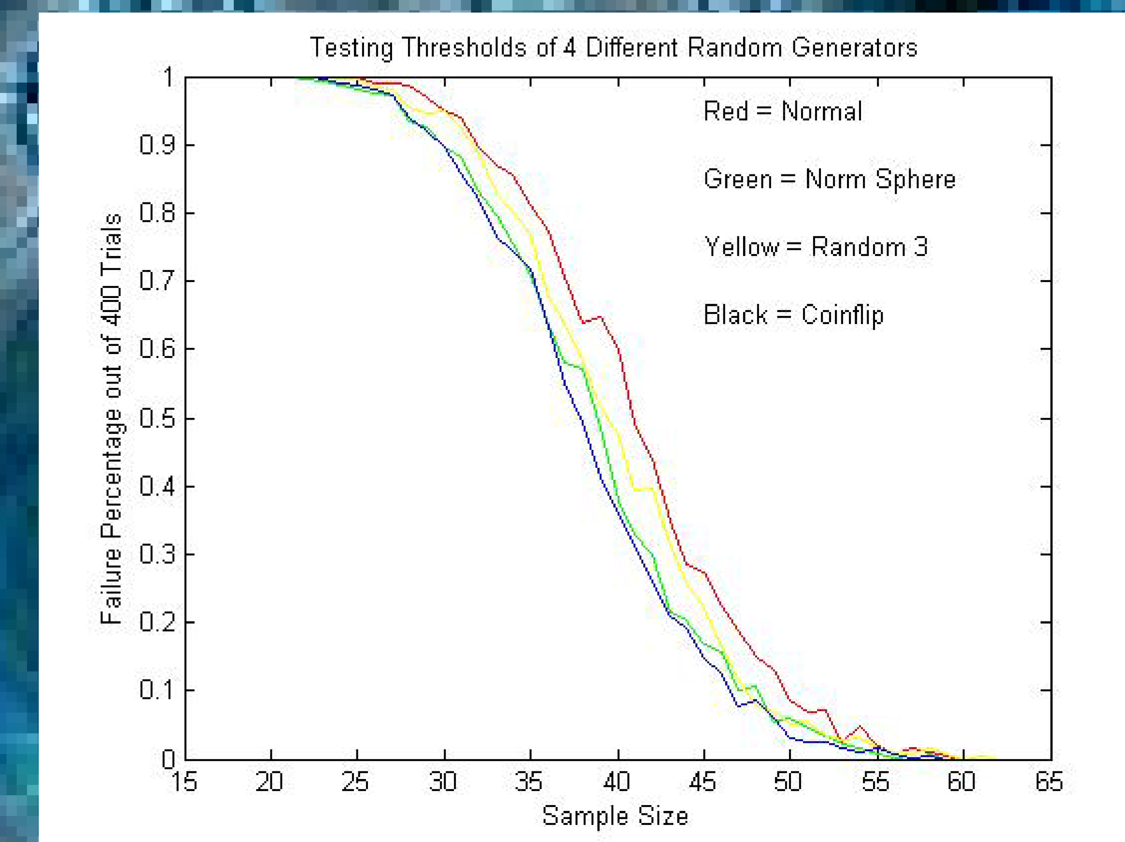
4. The key requirement of compressive sampling which allows it to be successful is that the signals must be sparse. Sparsity means that most of the elements in the signal are zeros. The deterministic theory talks about how for given choices of n, m, and k (k being the number of nonzero elements in the signal), a sampling matrix can be designed which is guaranteed not to fail. When solving for the signal from the sample we have an underdetermined system of equations, as shown in the example (x are the elements of the signal which we are solving for, A is the compression matrix, and b is the sample). Because of an underdetermined system of equations, we will get several possible signal solutions. The deterministic theory shows that the solution with the smallest L1 norm (the minimum value of the sum of the absolute values of each element) is the original signal.

$$A = \begin{pmatrix} A_{11} & A_{12} & A_{13} & A_{14} \\ A_{21} & A_{22} & A_{23} & A_{24} \end{pmatrix} \quad x = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} \quad b = \begin{pmatrix} b_1 \\ b_2 \end{pmatrix}$$

$$A*x = b$$

$$\begin{array}{ccccc} A_{11}x_1 & A_{12}x_2 & A_{13}x_3 & A_{14}x_4 & = & b_1 \\ A_{21}x_1 & A_{22}x_2 & A_{23}x_3 & A_{24}x_4 & = & b_2 \end{array}$$

5. The probabilistic theory deals with the problem of constructing sampling matrices which successfully compress and reconstruct arbitrary k-sparse signals. The binary bit counter example shown above and to the right is an example of a compression matrix specifically designed for signals of k=1 and n=8. Although this compression matrix never fails with m=4, it is too special because we want compression matrices which work for a variety of k,m,n signals and are easy to construct. It turns out that creating compression matrices by random generators is the best solution to our problem. Some of the random generators we researched are described in the next section.



6. We used four different random generation methods for the compression matrices. The first generator we used was the normal distribution; each element in the 'normal' compression matrix was chosen from the normal distribution with mean 0 and standard deviation $1/\sqrt{m}$. Secondly we used the normal distribution again to create random points on the unit sphere in dimension m. To do this we did the same as above except we normalized each column so the L2 norm of each column was 1 (therefore it would land on the m-sphere). This random generator is listed as 'norm sphere' on the graphic. Next we used a uniform distribution of numbers in the range from $-\sqrt{(3/m)}$ to $\sqrt{(3/m)}$ 'random 3'. Lastly we used a random generator of 2 numbers for each element and then altered those two numbers so they became -1 or 1, 'coinflip'.

7. Sample matrices constructed by random methods are not guaranteed to work but they can be designed so that the probability of failure is negligible. The data we collected above was done with n=1000 and k=6. The threshold region is the area in which the failure rate changes from nearly certain failure to practically perfect success. We found the coin flip method to work the best because the threshold region occurs sooner than the remaining methods.

8. The visual to the right is known as the binary bit counter. A bit counter is a compressive sampling example with k=1. The binary example has a compression matrix specifically designed for this case and size. This is not a practical compression matrix because of this; to create a practical compression matrix we used the randomly generated compression matrices with a 1-sparse signal.

$$\begin{pmatrix} 6.1 \\ 0 \\ 6.1 \\ 6.1 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 6.1 \\ 0 \end{pmatrix}$$

9. The coin flip method was the worst matrix for small signals because the chance of a row in the compression matrix having the exact same values as another row is high because we are only picking between two numbers. When two rows have the same values, we lose an equation when solving back for the signal.

10. The theory also produced an inequality for the sample size: $m \geq C\, k \ln(n/k)$. The theory however only guarantees a C which is very large and only practical for large signals. On the bright side this estimate of C is very conservative and in practice it is magnitudes smaller. Additionally, C decreases as the ratio n/k increases. One example we have from the theory is for n=1,000,000 and k=4, m has to be 22,000 to guarantee success. From the above practical example with n=1000 and k=6, an m of 65 will actually work, but to be safe we can go to 100. If we square n and k, this only makes m at most 12 times bigger (this is a generous number because as the ratio n/k goes from 1000/6 to 1,000,000/36, C decreases). Thus in practice compressive sampling is that much better because the theory says for n=1,000,000 and k=4, m has to be 22,000; in practice m only has to be 1200 and this is with k=36 which should require more samples than with a k=4.

Some Matlab commands were derived with help from Emmanuel Candès's notes. Convex optimization in Matlab was done thanks to an add-on made by Stephen Boyd, Michael Grant, and Yinyu Ye. The theories and equations were made possible by Emmanuel Candès, Ronald DeVore, Willard Miller, and Terrence Tao.