1 Finding the sparsest solution

We’re given an \( m \times n \) matrix \( A \) and an \( m \)-vector \( b \), where \( m \ll n \). We want to find a solution \( x \) in \( \mathbb{R}^n \) to \( Ax = b \). Have many possible solutions to \( Ax = b \).

Try to find a sparse \( x \), that is, a solution \( x \) where many of the components are equal to zero.

We define the \( L_0 \)-norm of a vector \( x \) as

\[
||x||_0 := \text{number of nonzero components of } x.
\]

(Note that this is not actually a norm, since it violates homogeneity: the two vectors \( x \) and \( 5x \) have the same number of nonzero components. But “\( L_0 \)-norm” is standard terminology.)

So we can write down an optimization problem:

\[
\begin{align*}
\text{min} & \quad ||x||_0 \\
\text{subject to} & \quad Ax = b
\end{align*}
\]

This is an NP-Complete problem, which means that it is very likely that the time required to solve at least some problem instances is going to get very large as \( m \) and \( n \) increase. Finding the sparsest solution might require \( O(\frac{n}{m}) \) time.

We’d rather work with problems that can be solved in polynomial time. For example, Gaussian elimination requires \( O(n^3) \) time to find a solution to \( Ax = b \).

<table>
<thead>
<tr>
<th>( m )</th>
<th>( n )</th>
<th>( n^3 )</th>
<th>( \binom{n}{m} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>10</td>
<td>1000</td>
<td>252</td>
</tr>
<tr>
<td>20</td>
<td>40</td>
<td>64000</td>
<td>137,846,642,602</td>
</tr>
</tbody>
</table>

Problems that can be solved in polynomial time include linear optimization (although not by the simplex method). Such problems are said to be in the complexity class \( P \). It is an open problem to prove whether or not NP-complete problems can actually be solved in polynomial time. This is one of the Clay Millennium Challenge problems, see [http://www.claymath.org/millennium-problems/p-vs-np-problem](http://www.claymath.org/millennium-problems/p-vs-np-problem)

2 Formulate as an integer optimization problem

If we have upper bounds \( M_i \) on the absolute value of \( x_i \) for each \( i = 1, \ldots, n \) then we can formulate the problem of finding the sparsest solution as an integer optimization problem:

\[
\begin{align*}
\text{min} & \quad \sum_{i=1}^{n} z_i \\
\text{subject to} & \quad Ax = b \\
& \quad -M z_i \leq x_i \leq M_i z_i \quad i = 1, \ldots, n \\
& \quad z_i \text{ binary} \quad i = 1, \ldots, n
\end{align*}
\]
If $x_i$ is nonzero we must take $z_i = 1$, and if $x_i = 0$ then we can take $z_i = 0$.

3 Approximate with a different norm

How about minimizing the standard Euclidean norm instead of $||x||_0$?

$$\min \sum_{i=1}^{n} x_i^2$$
subject to $Ax = b$

But it seems this norm doesn’t encourage sparsity:

4 Approximate using the L1-norm

The $L_1$-norm of a vector $x \in \mathbb{R}^n$ is the sum of the absolute values of the components of $x$:

$$||x||_1 = \sum_{i=1}^{n} |x_i|$$

Approximate the problem of finding the sparsest solution by minimizing the $L1$-norm of $x$:

$$\min ||x||_1$$
subject to $Ax = b$

The $L1$-norm of a vector $x \in \mathbb{R}^n$ is the sum of the absolute values of the components of $x$:
5 Formulate as a linear optimization problem

Split the free variable $x$ into the difference of two nonnegative variables, so

$$x = u - v, \quad u, v \geq 0$$

Then $|x_i| = u_i + v_i$, assuming at least one of $u_i, v_i$ is zero.

Can represent our problem as a linear program:

$$\min \sum_{i=1}^{n} u_i + \sum_{i=1}^{n} v_i$$

subject to

$$Au - Av = b$$

$$u, v \geq 0$$

Basic feasible solutions will have no more than $m$ nonzeros. For many applications, will have sparse solutions with even fewer nonzeros.

6 Reweighting

Can often improve the solution by reweighting [2]: choose weights $w \in \mathbb{R}^n$ and solve

$$\min \sum_{i=1}^{n} w_i u_i + \sum_{i=1}^{n} w_i v_i$$

subject to

$$Au - Av = b$$

$$u, v \geq 0$$

This has the effect of rescaling our “diamond”.

The weights can be chosen based on the previous solution:

if $x_i$ is close to zero, increase $w_i$.

For example, choose a small positive $\delta$ and set

$$w_i = \frac{1}{|x_i| + \delta}.$$ 

7 What about noise?

In practice, we don’t expect to be able to measure $A$ and $b$ perfectly accurately. If we force $Ax = b$ then we are unlikely to get a sparse solution. Given an original sparse $\bar{x}$, we observe

$$b = A\bar{x} + \epsilon$$

for some small error vector $\epsilon \in \mathbb{R}^n$. We might get back the solution $\bar{x} + \beta$ for some $\beta$ satisfying $A\beta = \epsilon$, which may be far more dense than $\bar{x}$.

There are at least a couple of ways to handle errors:
- **Regularization**: Replace the linear constraints by a quadratic objective, penalizing violation of the constraints:

\[
\min_x \|x\|_1 + \mu\|Ax - b\|_2^2
\]

for some positive constant \(\mu\). The parameter \(\mu\) trades off between the sparsity objective \(\|x\|_1\) and the errors in \(Ax = b\). By splitting \(x = u - v\), can cast this as a quadratic program:

\[
\min \sum_{i=1}^n u_i + \sum_{i=1}^n v_i + \sum_{i=1}^n s_i^2 \\
\text{subject to } Au - Av + s = b \quad (QP_r)
\]

The objective function here is convex, so this can still be solved efficiently. In fact, the solver CPLEX can solve this type of problem.

The unconstrained problem can also be solved directly with appropriately designed nonlinear optimization algorithms.

- **Allow some errors**: Choose a parameter \(\sigma\), and don’t allow the norm of the errors to be larger than \(\sigma\):

\[
\min \|x\|_1 \\
\text{subject to } \|Ax - b\|_2^2 \leq \sigma^2 \quad (QP_b)
\]

Again, this is a convex program and can be solved efficiently (by CPLEX and other solvers).

Can get **performance guarantees** for this approach, at least if the constraint matrix satisfies something called the *Restricted Isometry Property (RIP)*. Let \(\hat{x}\) be the original \(x\) and assume it has \(s\) nonzeros. Assume also that the error is bounded by \(\epsilon\). More precisely, assume

\[
\|b - A\hat{x}\|_2 \leq \epsilon.
\]

Let \(x^{QP}\) be the solution returned by solving \((QP_b)\), with \(\sigma = O(\epsilon)\). Then Candes, Romberg, and Tao [1] proved that with high probability

\[
\|x^{QP} - \hat{x}\|_2 \leq \epsilon + \frac{\|x^{QP} - \hat{x}\|_1}{\sqrt{s}}.
\]

**References**
