Least squares and least norm in Matlab

Least squares approximate solution

Suppose $A \in \mathbf{R}^{m\times n}$ is skinny (or square), *i.e.*, $m \geq n$, and full rank, which means that $\mathbf{Rank}(A) = n$. The least-squares approximate solution of $Ax = y$ is given by

$$x_{\rm ls} = (A^T A)^{-1} A^T y.$$

This is the unique $x \in \mathbf{R}^n$ that minimizes $\|Ax - y\|$.

There are several ways to compute $x_{\rm ls}$ in Matlab. The simplest method is to use the *backslash operator*:

```
xls=A\y;
```

If A is square (and invertible), the backslash operator just solves the linear equations, *i.e.*, it computes $A^{-1}y$. If A is not full rank, then `A\b` will generate an error message, and then *a* least-squares solution will be returned.

You can also use the formula to compute the least squares approximate solution:

```
xls=inv(A'*A)*A'*y;
```

We encourage you to verify that you get the same answer (except possibly for some small difference due to roundoff error) as you do using the backslash operator. Here you'll get an error if A is not full rank, or fat. Compared with the backslash operator, this method is a bit less efficient, and a bit more sensitive to roundoff errors, but you won't notice it unless m and n are a thousand or so.

You can also use the pseudo-inverse function `pinv()`, which computes the pseudo-inverse, which is

$$A^\dagger = (A^T A)^{-1} A^T$$

when A is full rank and skinny (or square). (The pseudo-inverse is also defined when A is not full rank, but it's not given by the formula above.) To find the least-squares approximate solution using the pseudo-inverse, you can use

```
xls=pinv(A)*y;
```

You better be sure here that A is skinny (or square) and full rank; otherwise you'll compute something (with no warning messages) that isn't the least-squares approximate solution.

Yet another method is via a QR decomposition. In Matlab, `[Q,R]=qr(A)` returns the 'full' QR decomposition, with square, orthogonal $Q \in \mathbf{R}^{m\times m}$, and $R \in \mathbf{R}^{m\times n}$ upper triangular (with zeros in its bottom part). The 'economy' QR decomposition, in which $Q \in \mathbf{R}^{m\times n}$ (with orthonormal columns) and invertible R, is obtained using `[Q,R]=qr(A,0)`. You can compute the least-squares approximate solution using the economy QR decomposition using, for example,

```
[Q,R]=qr(A,0); % compute economy QR decomposition
xls=R\(Q'*y);
```

To be sure you've really computed the least-squares approximate solution, we encourage you to check that the residual is orthogonal to the columns of A, for example with the commands

```
r=A*x-y; % compute residual
A'*r % compute inner product of columns of A and r
```

and checking that the result is very small.

Least norm solution

Now suppose $A \in \mathbf{R}^{m\times n}$ and is fat (or square), *i.e.*, $m \leq n$, and full rank, which means that $\mathbf{Rank}(A) = m$. The least-norm solution of $Ax = y$ is given by

$$x_{\mathrm{ln}} = A^T(AA^T)^{-1}y.$$

Among all solutions of $Ax = y$, x_{ln} has the smallest norm.

We can compute x_{ln} in Matlab in several ways. You can use the formula to compute x_{ln}:

```
xln=A'*inv(A*A')*y;
```

You'll get an error here if A is not full rank, or if A is skinny.

You can also use the pseudo-inverse function `pinv()`, which computes the pseudo-inverse, which is

$$A^\dagger = A^T(AA^T)^{-1}$$

when A is full rank and fat or square. (The pseudo-inverse is also defined when A is not full rank, but it's not given by the formula above.)

```
xln=pinv(A)*y;
```

You better be sure here that A is fat (or square) and full rank; otherwise you'll compute something (with no warning messages) that isn't the least-norm solution.

You can find the least-norm solution via QR as well:

```
[Q,R]=qr(A',0);
xln=Q*(R'\y);
```

Warning! We should also mention a method that *doesn't work*: the *backslash operator*. If A is fat and full rank, then `A\y` gives a solution of the $Ax = y$, not necessarily the least-norm solution.