Editor: Dianne P. O'Leary, oleary@cs.umd.edu

THE DIRECTION-OF-ARRIVAL PROBLEM: COMING AT YOU

By Dianne P. O'Leary

IF YOU BREAK YOUR LEG ON A MOUNTAIN BUT HAVE A CELL PHONE OR OTHER TRANSMITTER WITH YOU, YOU WOULD HOPE A RESCUER COULD DETERMINE THE DIRECTION IN WHICH to travel to reach you. Similarly, if a navy detects a transmission from a submarine, it would want to determine the signal's *direction of arrival* (DOA) to locate that sub. The problem is complicated if more than one signal appears—especially if the number of signals is unknown—and even more complicated if you or the submarine is moving.

Surprisingly, we will see that your rescuer can solve an eigenvalue problem (involving the product of some unknown matrices) and use that information to find you. The DOA-finding algorithm we'll use is called *Esprit*. To understand the process, we also will use several matrix decompositions and illustrate the necessity of using *update techniques* for real-time computations.

The DOA Problem Definition

Suppose we have d signal sources, each a long distance from the receivers. Also, suppose that the signals are *narrow band*; we can approximate each one via a plane wave of fixed frequency ω.

Let's take m sensor pairs whose locations are arbitrary except that the spacing δ and orientation of the two sensors in each pair is constant. We measure the signal reaching each sensor as a function of time. Figure 1 illustrates a sample configuration with $m = 4$ sensor pairs and $d = 2$ signal sources. Let $s_k(t)$ be the signal emitted from the kth source at time t, $k = 1, \ldots, d$, and let $\mathbf{s}(t)$ be the vector composed of these components. The vector function $\mathbf{x}_1(t)$ denotes the m signal measurements $x_{1j}(t)$, $j = 1, \ldots, m$, for the first sensor in each pair at time t, and $\mathbf{x}_2(t)$ denotes the corresponding measurements for the second sensor in each pair.

After we take measurements for some time, we want to determine the DOAs: the angles between each plane wave and a line parallel to the lines connecting each sensor pair. We call these d angles θ_k, $k = 1, \ldots, d$; in Figure 1, these angles are 45° and –30°.

We model the sensor measurements as a function of the signal as

$$\begin{aligned}\mathbf{x}_1(t) &= A\mathbf{s}(t) + \epsilon_1(t),\\ \mathbf{x}_2(t) &= A\Phi\mathbf{s}(t) + \epsilon_2(t).\end{aligned}$$

The matrix A of size $m \times d$ is an unknown matrix of *array-steering vectors*, and the matrix Φ is a diagonal matrix that accounts for phase delays between the sensors in each pair. The kth diagonal entry is

$$\phi_k = e^{i\,\omega\,\delta\sin\theta_k/c},\ k = 1, \ldots, d,$$

where $i = \sqrt{-1}$ and c is the speed of the signals. Our problem, then, is to determine Φ, given $\mathbf{x}$, δ, and ω, without knowing A, $\mathbf{s}(t)$, the measurement noise $\epsilon_1(t)$ and $\epsilon_2(t)$, or even d! As an added complication, if the sources are moving, then Φ also is a function of t.

Finding Φ When We Know d

We can build algorithms on a clever observation about how to manipulate our problem to extract Φ. Let's assume we know the number of signals $d < m$. Let's next observe the system for n time steps. Let X_1 have m rows and n columns containing the data values $\mathbf{x}_1(t)$. Similarly, construct X_2 from

In this month's problem, we use linear algebra and matrix-updating techniques to track a set of moving signals. The solution to last month's problem on clustering data, which appears at the end of this article, illustrates the complications of determining useful clusters.

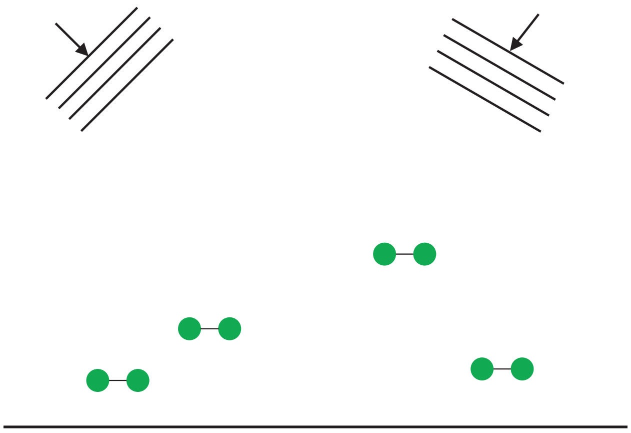

Figure 1. Two signals (plane waves) received by four sensor pairs. In this figure, $m = 4$ sensor pairs and $d = 2$ signal sources.

the data $\mathbf{x}_2(t)$. This way of collecting the data is called *rectangular windowing*, with a window size of n. If we neglect the errors $\in(t)$, then our system becomes

$$X = \begin{bmatrix} X_1 \\ X_2 \end{bmatrix} = \begin{bmatrix} A \\ A\Phi \end{bmatrix} S,$$

where the columns of S are $\mathbf{s}(t)$. If A is full rank, then its rank is d, which is also the maximal rank of X. Consider the following recipe:

- Find a matrix B of size $d \times m$ so that BA is $d \times d$ and full rank.
- Find a matrix C of size $n \times d$ so that SC is $d \times d$ and full rank.
- Find d vectors $\mathbf{z}_k$ and d values λ_k so that $BA\Phi SC\mathbf{z}_k = \lambda_k BASC\mathbf{z}_k$.

Note that λ_k is an *eigenvalue* of the generalized eigenproblem involving the matrices $BA\Phi SC$ and $BASC$, and that $\mathbf{z}_k$ is the corresponding eigenvector.

Problem 1. Show that the eigenvalues λ_k are equal to the diagonal entries of Φ.

By solving Problem 1, we've accomplished something rather surprising: without knowing A or Φ, we can choose matrices B and C of the proper dimensions and construct the matrices for the eigenproblem just by knowing X, because $BASC = BX_1C$ and $BA\Phi SC = BX_2C$. But we need to make sure that BA and SC have full rank.

Tools

In this problem, we use three matrix decompositions: the singular value decomposition (SVD),1 the eigendecomposition,1 and the rank-revealing URV decomposition with updating.2

Linear algebra and numerical linear algebra texts typically discuss the *generalized* eigenvalue problem used in Problem 1.[1] To solve this problem, try to reduce $BA\Phi SC\mathbf{z}_k = \lambda_k BASC\mathbf{z}_k$ to $\Phi\mathbf{w}_k = \lambda_k \mathbf{w}_k$ for some vector $\mathbf{w}_k$.

Radhika Roy and Thomas Kailath originally proposed the Esprit algorithm;[3] K.J.R. Liu and his colleagues proposed the URV variant in an article for *IEEE Transactions on Signal Processing*,[4] which also is the data source we use in Problems 3, 5, and 7.

Problem 6 relies on a formula for $a + a^2 + \ldots + a^k$ when $0 < a < 1$, and information in any basic statistics text discussing the normal distribution.

For Problem 8, you can find a detailed discussion of the URV-Esprit algorithm in the article mentioned above.[4]

A more common algorithm for determining DOAs is the *Music* algorithm. We also can formulate this algorithm in terms of matrix decompositions. Standard textbooks such as Simon Haykin's[5] provide further information.

References

1. G.H. Golub and C.F. Van Loan, *Matrix Computations*, 2nd ed., Johns Hopkins Press, 1989.
2. G.W. Stewart, "An Updating Algorithm for Subspace Tracking," *IEEE Trans. Signal Processing*, vol. 40, 1992, pp. 1535–1541.
3. R. Roy and T. Kailath, "ESPRIT – Estimation of Signal Parameters via Rotational Invariance Techniques," *Signal Processing Part II: Control Theory and Applications*, F.A. Grünbaum, J. W. Helton, and P. Khargonekar, eds., Springer-Verlag, 1990, pp. 369–411.
4. K.J.R. Liu et al., "URV ESPRIT for Tracking Time-Varying Signals," *IEEE Trans. Signal Processing*, vol. 42, 1994, pp. 3441–3448.
5. S. Haykin, *Adaptive Filter Theory*, 2nd ed., Prentice-Hall, 1991, Chapter 12.

SVD-Esprit and Rectangular Windowing

To ensure the full-rank conditions, we use a matrix's *singular value decomposition* (SVD): we can factor any matrix F of dimension $p \times q$ as

$$F = U\Sigma W^H,$$

where U is $p \times p$, W is $q \times q$, $U^H U = I$, and $W^H W = I$. (The i, j element of W^H is $\overline{\omega}_{ji}$. The condition $W^H W = I$ means that W is a *unitary* matrix.) The real diagonal matrix Σ of dimension $p \times q$ has nonzeros $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_{\hat{q}}$, where $\hat{q} = \min(p, q)$. Because $\boldsymbol{U}$ and $\boldsymbol{W}$ are unitary matrices, their columns (and rows) are well-conditioned bases for the subspaces they span; using unitary matrices leads to numerically stable choices for $\boldsymbol{B}$ and $\boldsymbol{C}$, which we define in Problem 2.

Problem 2. Suppose that the SVD of $\boldsymbol{X}$ is $\boldsymbol{U}\Sigma\boldsymbol{W}^H$, where $\sigma_i = 0$, when $i > d$. Let Σ_1 be the square diagonal matrix with entries $\sigma_1, \ldots, \sigma_d$, and partition $\boldsymbol{U}$ into

$$\boldsymbol{U} = \begin{bmatrix} \boldsymbol{U}_1 & \boldsymbol{U}_3 \\ \boldsymbol{U}_2 & \boldsymbol{U}_4 \end{bmatrix},$$

where $\boldsymbol{U}_1$ and $\boldsymbol{U}_2$ have m rows and d columns, so that

$$\begin{aligned} \boldsymbol{X}_1 &= \boldsymbol{AS} = \boldsymbol{U}_1[\Sigma_1, \boldsymbol{O}_{d\times(n-d)}]\boldsymbol{W}^H, \\ \boldsymbol{X}_2 &= \boldsymbol{A}\Phi\boldsymbol{S} = \boldsymbol{U}_2[\Sigma_1, \boldsymbol{O}_{d\times(n-d)}]\boldsymbol{W}^H, \end{aligned}$$

where $\boldsymbol{O}_{d\times(n-d)}$ is the zero matrix of size $d \times (n-d)$. Let $\hat{U} = [\boldsymbol{U}_1\boldsymbol{U}_2]$ have SVD $\boldsymbol{T}\Delta\boldsymbol{V}^H$, and denote the leading $d \times d$ submatrix of Δ by Δ_1. Partition

$$\boldsymbol{V} = \begin{bmatrix} \boldsymbol{V}_1 & \boldsymbol{V}_3 \\ \boldsymbol{V}_2 & \boldsymbol{V}_4 \end{bmatrix},$$

so that $\boldsymbol{V}_1$ and $\boldsymbol{V}_2$ have dimension $d \times d$. Let

$$\boldsymbol{B} = [\Delta_1^{-1}, \boldsymbol{O}_{d\times(m-d)}]\boldsymbol{T}^H$$

and

$$\boldsymbol{C} = \boldsymbol{W}\begin{bmatrix} \Sigma_1^{-1} \\ \boldsymbol{O}_{(n-d)\times d} \end{bmatrix}.$$

Show that the eigenvalues λ_k that satisfy the equation $V_2^H \mathbf{z}_k = \lambda_k V_1^H \mathbf{z}_k$ are ϕ_k.

So, now we have our first algorithm for solving the DOA problem:

- Compute the SVD of $\boldsymbol{X} = \boldsymbol{U}\Sigma\boldsymbol{W}^H$.
- Compute the SVD of $\hat{U} = [\boldsymbol{U}_1, \boldsymbol{U}_2] = \boldsymbol{T}\Delta\boldsymbol{V}^H$.
- Solve the generalized eigenvalue problem $V_2^H\mathbf{z} = \lambda V_1^H\mathbf{z}$ for the values $\lambda_k = \phi_k$, $k = 1, \ldots, d$.

In Problem 3, we see how this algorithm performs.

Problem 3. Program the SVD algorithm and experiment with rectangularly windowed data and a window size of $n = 10$. Note that we need to compute $\boldsymbol{U}$ and $\boldsymbol{V}$, but we do not need $\boldsymbol{B}$ or $\boldsymbol{C}$. You can find sample data for $\boldsymbol{X}$ and Φ on the Web site (www.computer.org/cise/homework/v5n6.htm). Plot the true and computed DOAs as a function of time and compute the average absolute error in your DOA estimates (absolute value of true value minus computed value) and the average relative error (absolute error divided by absolute value of true value).

Eigen-Esprit and Exponential Windowing

Experimenting further with the data in Problem 3, we discover that rectangular windowing has a drawback: if the window size n is too small, then the DOAs will be sensitive to errors in the measurements and our estimates could change abruptly. But if the window size is too large, then very old data could contribute to our measurements, which will make our estimates bad if the sources move too quickly. The cure for this is to use old data but give more weight to newer data. We do this in *exponential windowing* by multiplying all our old data by a *forgetting factor* f between 0 and 1 every time we add new data. Thus, after n observations, column ℓ of $\boldsymbol{X}_1$ contains data from observation ℓ multiplied by $f^{n-\ell}$, and similarly for $\boldsymbol{X}_2$.

Using exponential windowing, the number of columns in the matrix $\boldsymbol{X}$ can grow quite large, which makes the SVD too expensive since the cost is proportional to nm^2. We could avoid an SVD (for either exponential windowing or rectangular windowing) but still use an orthogonal basis by noting that $\boldsymbol{XX}^H = \boldsymbol{U}\Sigma\Sigma^T\boldsymbol{U}^H$, so we can compute $\boldsymbol{U}$ from the eigenvectors of the $2m \times 2m$ matrix $\boldsymbol{XX}^H$. Problem 4 shows how quickly we can form this matrix.

Problem 4. Suppose the matrix $\boldsymbol{X}$ contains the exponential windowing data and that a new data vector $\mathbf{x}$ arrives. Give a formula for the new exponential windowing data matrix and show that the cost of computing it from $\boldsymbol{X}$ and $\mathbf{x}$ is $O(m^2)$ multiplications.

Now, in Problem 5, we try this eigenvalue variant of Esprit, computing an eigendecomposition of $\boldsymbol{XX}^H$ in place of an SVD of $\boldsymbol{X}$.

Problem 5. Program the Eigen-Esprit algorithm and experiment with exponential windowing for Problem 3's data. Use the forgetting factor $f = 0.9$, and compare the results with those of Problem 3.

Determining the Number of Sources

Suppose we don't know how many source signals we're receiving. Recall that if $m \geq d$ and $\boldsymbol{A}$ is full rank, then its rank is d, which is also the maximal rank of $\boldsymbol{X}$. Therefore, we can estimate d experimentally by taking it to be the rank of the matrix $\boldsymbol{X}$. This works fine in the absence of error if the signals are all stationary, but if they're moving or if there is error in our measurements, the matrix $\boldsymbol{X}$ will have some small nonzero singular values in addition to d nonzeros. We must be able to distinguish between real signals and noise. If you know some statistics, you can solve Problem 6 to predict how large the noise will be.

Problem 6. Suppose we have a matrix $\boldsymbol{X}$ of size $m \times n$, $m \leq n$ and that each element of $\boldsymbol{X}$ is normally distributed with mean 0 and standard deviation ψ.

(a) Show that the random variable equal to the sum of the squares of the entries of $\boldsymbol{X}$ is equal to the sum of the squares of the singular values of $\boldsymbol{X}$.

(b) Show, therefore, that for rectangular windowing of this data, the expected value of $\sigma_1^2 + \ldots + \sigma_m^2$ is $\psi^2 mn$, where σ_j is a singular value of $\boldsymbol{X}$.

(c) Using a similar argument, show that for exponential windowing, the expected value of $\sigma_1^2 + \ldots + \sigma_m^2$ is approximately $\psi^2 f^2 m/(1 - f^2)$, where σ_i is a singular value of $\boldsymbol{FX}$. Here, $\boldsymbol{F}$ is a diagonal matrix, with jth entry equal to f^j.

This gives us a way to experimentally determine d: choose it to be the smallest value for which the remaining singular values satisfy

$$\sigma_{d+1}^2 + \ldots + \sigma_m^2 < \kappa\psi^2 n(m - d)$$

for rectangular windowing, and

$$\sigma_{d+1}^2 + \ldots + \sigma_m^2 < \kappa f^2 \psi^2 (m - d)/(1 - f^2)$$

for exponential windowing, where $\kappa > 1$ is a user-chosen tolerance.

Now we can experiment with this algorithm for determining the number of signals and their DOAs.

Problem 7. Modify your programs to determine d and explore the methods' sensitivity to the choice of n, f, and κ.

Using URV for Efficiency

Computing SVDs and eigendecompositions from scratch can be too computationally intensive to keep up with incoming data; the operation counts are proportional to $m^2 n$ for the SVD and m^3 for the eigendecomposition. To keep up with incoming data, we must find ways to update our DOA estimates at a lower cost. Unfortunately, there is no easy way to update SVDs or eigendecompositions, but there is a closely related decomposition, the *rank-revealing URV decomposition*, which can be updated. If we substitute this for the SVD of $\boldsymbol{X}$ or the eigendecomposition of $\boldsymbol{XX}^H$, then our algorithm will have a cost proportional to $d^3 + m^2 + n^2$ and will be suitable for real-time applications—as long as the number of incoming signals is not too great.

The rank-revealing URV decomposition of $\boldsymbol{X}$ is

$$\boldsymbol{X}^H = \boldsymbol{URV}^H = \boldsymbol{U}\begin{bmatrix} \overline{\boldsymbol{R}} & \boldsymbol{F} \\ \boldsymbol{O} & \boldsymbol{G} \end{bmatrix}\boldsymbol{V}^H,$$

where $\boldsymbol{U}$ and $\boldsymbol{V}$ are square unitary matrices ($\boldsymbol{U}^H\boldsymbol{U} = \boldsymbol{I}$, $\boldsymbol{V}^H\boldsymbol{V} = \boldsymbol{I}$), $\overline{\boldsymbol{R}}$ is an upper triangular matrix of size $d \times d$, and $\boldsymbol{G}$ is an upper triangular matrix of size $(n - d) \times (2m - d)$. In addition, the norms of the matrices $\boldsymbol{F}$ and $\boldsymbol{G}$ should be small. Therefore, $\boldsymbol{X}$ is within $\sqrt{\|\boldsymbol{F}\|^2 + \|\boldsymbol{G}\|^2}$ of the matrix of rank d obtained by setting these two blocks to zero. The SVD is a special case of this decomposition in which $\boldsymbol{F}$ is zero and $\overline{\boldsymbol{R}}$ and $\boldsymbol{G}$ are diagonal, but by allowing the more general case, we gain the ability to update the factorization inexpensively as new data arrives.

(Extra) Problem 8. Implement the URV-updating algorithm (or use available software) and use it on the matrix $\boldsymbol{X}$ to solve the DOA problem for rectangular and exponential windowing.

Acknowledgments

This project benefited from helpful discussions with K.J.R. Liu and Simon P. Schurr.