EECS 16A Designing Information Devices and Systems I
Spring 2015 Note 17

Lecture notes by Rachel Zoll (03/19/2015)

1 Communications

In order to obtain more data from a given system, it is necessary to add more beacons. This allows location estimates to have smaller errors using the least-squares method, and permits smaller error bounds for a given set of data.

Note that for a given system, if there are more equations n than unknowns m, the equivalent setup in the real would be adding more beacons.

For example, consider a case where $n > m$:

$$\begin{gathered}
a_{11}x_1 + a_{12}x_2 + \ldots + a_{1m}x_m = b_1 \\
a_{21}x_1 + \ldots + a_{2m}x_m = b_2 \\
\vdots \\
a_{n1}x_1 + \ldots + a_{nm}x_m = b_n
\end{gathered}$$

In the past, we could say that our equations were EXACTLY equal to b_1. In this scenario, though, there is some error - we must include some small error to account for small amounts of noise in our measurement. Here, we represent error as a constant term, e_1, being added to our linear expression of the measurement. Our equation would then be represented as $a_{11}x_1 + a_{12}x_2 + \ldots + a_{1m}x_m = b_1 + e_1$, where e_1 represents the error in our main measurement.

We can now write this set of equations as a matrix:

$$\begin{bmatrix} a_{11} & a_{12} & \ldots & a_{1m} \\ a_{21} & a_{22} & \ldots & a_{2m} \\ \vdots & \vdots & \vdots & \vdots \\ a_{n1} & \ldots & \ldots & a_{nm} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \ldots \\ x_m \end{bmatrix} = \begin{bmatrix} \vec{b} \end{bmatrix} + \begin{bmatrix} \vec{e} \end{bmatrix}$$

Here, we are considering e as unknown. If we had enough equations, we could more accurately estimate e; however, in this case, we can only estimate because we don't know x or e. Adding more equations would reduce our error bounds because our errors would begin to average out. A diversity of measurements allows the aggregated error to be reduced.

To better understand this scenario, let's take the simplest case, in which there is one variable, x, and two equations, $a_{11}x = b_1 + e_1$ and $a_{21}x = b_2 + e_2$. A naive approach to solve this system would be to take $x = \frac{b_1}{a_{11}}$ and $x = \frac{b_2}{a_{21}}$. This approach is faulty though, because it cannot predict which measurements will have noise. In the physical world, all measurements have some amount of noise.

Another approach is to use linear algebra:

$$\begin{bmatrix} a_{11} \\ a_{21} \end{bmatrix} x = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} + \begin{bmatrix} e_1 \\ e_2 \end{bmatrix} = A\vec{x} - \vec{b}$$

where x is a scale factor. Here, we are trying to minimize e by finding an x of best fit.

From a more mathematical standpoint, we are attempting to find a subspace such that $\begin{bmatrix} a_{11} \\ a_{21} \end{bmatrix}$ lies within the subspace. Additionally, we want to minimize the distance between our point and our vector $\begin{bmatrix} a_{11} \\ a_{21} \end{bmatrix}$. To do so, we find the minimum error that could be associated with our function:

$$\min_x ||\vec{e}||^2 = \min_x ||\vec{A}x - \vec{b}||^2 = \min_x [(a_{11}x - b_1)^2 + (a_{21}x - b_2)^2]$$

Notation-wise, $\min_x$ specifies that we should choose an x such that $\min_x f(x)$ is a minimum. Implicitly, we can choose any real number x to put into our function.

This notation makes sense because we want the mean-squared value to be as small as possible. This would represent the case where our data has as little noise as possible. The process for minimizing our function of x is as follows:

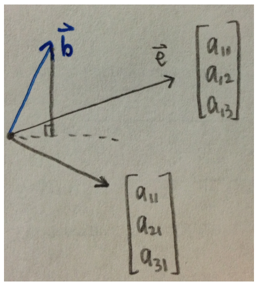
1) Take the first derivative of the function

$$\frac{\partial}{\partial x}[(a_{11}x - b_1)^2 + (a_{21}x - b_2)^2] = 2(a_{11}x - b_1)a_{11} + 2(a_{21}x - b_2)a_{21}$$

2) Set the first derivative of the function equal to zero to find its critical points. This step is similar to cross-correlation and the dot product.

$$x = \frac{b_1 a_{11} + b_2 a_{21}}{(a_{11})^2 + (a_{21})^2} = \frac{\vec{a} \cdot \vec{b}}{||\vec{a}||^2}$$

Now that we understand the basic steps necessary to find x, how can we generalize to n-dimensions?


$\vec{b}$
$\vec{e}$
$\begin{bmatrix} a_{11} \\ a_{12} \\ a_{13} \end{bmatrix}$
$\begin{bmatrix} a_{11} \\ a_{21} \\ a_{31} \end{bmatrix}$

In this case, where would we want to move such that our error vector is orthogonal to $\vec{b}$ and $\begin{bmatrix} a_{11} \\ a_{21} \\ a_{31} \end{bmatrix}$, as in the image above? Our vector equations can be represented as follows:

$$\begin{bmatrix} a_{11} \\ a_{21} \\ a_{31} \end{bmatrix} x + \begin{bmatrix} a_{12} \\ a_{22} \\ a_{32} \end{bmatrix} y = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} + \vec{e} \text{ OR } \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \\ a_{31} & a_{32} \end{bmatrix}$$

For even larger setups, our equations can be generalized to:

$$\begin{bmatrix} a_{11} & a_{12} & \dots & a_{1m} \\ \vdots & \vdots & \vdots & \vdots \\ a_{n1} & \dots & \dots & a_{nm} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix} + \vec{e}$$

Here, we want our error vector to be orthogonal to all of the columns. Another way to think of this matrix-vector multiplication is as a scaling by the elements of x.

How can we make our error be orthogonal to x? By definition of orthogonality, if $A^T\vec{e} = 0$, then $\vec{e}$ is orthogonal to every single row of our matrix:

$$\begin{bmatrix} a_!11 & a_{21} & \dots & a_{n1} \end{bmatrix} \begin{bmatrix} e_1 \\ e_2 \\ \vdots e_n \end{bmatrix} = 0$$

(Note that A^T has dimensions MxN, while A has dimensions NxM). Additionally, we know that $A^T(A\vec{x} - \vec{b}) = 0$ and $A^TA\vec{x} = A^T\vec{b}$.

Now that we've defined a procedure for determining x, how can we handle cases where A is non-invertible, meaning that our data is inconsistent, or we don't have enough data. The easiest case of an invertible A is where A is equivalent to the zero matrix. Another case to consider is where all of our beacons lies on the same line. To make A invertible, all we need to do is use more equations in our analysis. If we have enough equations, we'll be able to invert A eventually.

To find $\vec{x}$: $\vec{x} = (A^TA)^-1A^T \cdot \vec{b}$. Similarly, $A^TA = ||\vec{a}^2$ and $(A^TA)^{-1} = \frac{1}{(a_{11})^2+(a_{21})^2}$.

As a general overview, use least squares as a way of analyzing noisy data. To gain a more accurate representation of the data, factor in more measurements.

2 Application of Least Squares: Planetary Movement

Ceres $\in [Mars, Jupiter]$. Gauss used the method of least squares to determine where Ceres would emerge after its transit behind the sun, given a set of data points.

Goal: Find an equation for the ellipse-like orbit of Ceres to predict where the planet ends up after emerging.

We have an equation, $\alpha x^2 + \beta xy + \gamma y^2 + \delta x + \varepsilon y = 0$. We also have data. Our goal here is to determine the coefficients $\alpha, \beta, \gamma, \delta$, and ε, and to fit data to our points. This fitting is known as polynomial interpolation. As a first step, we must acknowledge the fact that there are an infinite number of points that would satisfy our equation. The first step is to normalize our equation so that we can decouple our time data from our actual position:

$$\alpha x^2 + \beta xy + \gamma y^2 + \delta x + \varepsilon y = 1$$

Next, write as a matrix-vector equation:

$$\begin{bmatrix} x_1^2 & x_1y_1 & y_1^2 & x_1 & y_1 \\ x_2^2 & \dots & \dots & \dots & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ x_n^2 & \dots & \dots & \dots & y_n \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \\ \gamma \\ \delta \\ \varepsilon \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}$$

If the data is too close together (in this case, if the data for the location of the planets were taken at too close of time intervals), this method won't produce a very accurate fit. Least squares is useful for polynomials, but it may be necessary to look at the data to determine which type of fit is best. We can tell that our data and fit are accurate by looking at our data to determine which one has the least error.

General Polynomial Fits: $p(x) = a_d x^d + a_{d-1} x^{d-1} + \ldots + a_1 x + a_) = 0$

Polynomial fits are possible if we have have many points (x, y) that we know pass through this line. For a degree d polynomial, we would want $d+1$ polynomials to have as many data points as possible. An example of a generalized polynomial fit:

$$\begin{bmatrix} x_1^d & x_1^{d-1} & \dots & x & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ x_n^d & x_n^{d-1} & \dots & x_n & 1 \end{bmatrix} \begin{bmatrix} a_d \\ a_{d-1} \\ \vdots \\ a_0 \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$$