Contents

5.2 Least squares

When we search for an *interpolating* polynomial of degree $n - 1$ to interpolate through n points, we are finding a solution to the system of linear equations $A\mathbf{x} = \mathbf{y}$. So long as all the x values are different, there is a unique solution; however, if the difference between two x values becomes too small, numerical instability begins to creep in.

A review of linear algebra

Suppose you have a single vector $\mathbf{u} \in \mathbf{R}^n$, then the *best approximation* of any other *target* vector $\mathbf{y} \in \mathbf{R}^n$ as a *scalar multiple* of the vector **u** is the projection of **u** onto **y**:

$$proj_{\mathbf{u}}\mathbf{y} = \frac{\mathbf{u}\bullet\mathbf{y}}{\mathbf{u}\bullet\mathbf{u}}\mathbf{u}$$

where $\mathbf{u}\bullet\mathbf{y}$ is the *dot product* or *inner product* of **u** and **y**. The reason that we call this the best approximation is because $\left\|\mathbf{y} - a\mathbf{u}\right\|_2 \geq \left\|\mathbf{y} - proj_{\mathbf{u}}\mathbf{y}\right\|_2$ for all possible values of the scalar a.

Please note that $\left\|\mathbf{v}\right\|_2$ refers to the 2-norm, also known as the Euclidean norm, of a vector, where $\left\|\mathbf{v}\right\|_2^2 = \mathbf{v}\bullet\mathbf{v}$ or $\left\|\mathbf{v}\right\|_2 = \sqrt{\mathbf{v}\bullet\mathbf{v}}$.

For example, the best approximation of the vector $\begin{pmatrix}-2\\3\\0\end{pmatrix}$ as a scalar multiple of $\begin{pmatrix}1\\2\\3\end{pmatrix}$ is $\frac{-2+6+0}{1+4+9}\begin{pmatrix}1\\2\\3\end{pmatrix} = \begin{pmatrix}\frac{2}{7}\\\frac{4}{7}\\\frac{6}{7}\end{pmatrix}$.

Next, suppose you have a collection of m orthogonal vectors $\mathbf{u}_1,\ldots,\mathbf{u}_m \in \mathbf{R}^n$ where $m \leq n$. A *linear combination* of the vectors $\mathbf{u}_1,\ldots,\mathbf{u}_m$ is any sum of scalar multiples of these vectors: $a_1\mathbf{u}_1 + \cdots + a_m\mathbf{u}_m$. Because these vectors are orthogonal, the best approximation of any target vector **y** is the sum of the projections onto the individual vectors:

$$proj_{\mathbf{u}_1}\mathbf{y} + \cdots + proj_{\mathbf{u}_m}\mathbf{y} = \frac{\mathbf{u}_1\bullet\mathbf{y}}{\mathbf{u}_1\bullet\mathbf{u}_1}\mathbf{u}_1 + \cdots + \frac{\mathbf{u}_m\bullet\mathbf{y}}{\mathbf{u}_m\bullet\mathbf{u}_m}.$$

Thus, for example, the best approximation of the vector target $\mathbf{y} = \begin{pmatrix} -2 \\ 3 \\ 0 \end{pmatrix}$ as a linear combination of $\begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$ and $\begin{pmatrix} -1 \\ 1 \\ 0 \end{pmatrix}$ is the linear combination

$$\frac{-2+3+0}{1+1+1}\begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} + \frac{2+3+0}{1+1+0}\begin{pmatrix} -1 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} \frac{1}{3} \\ \frac{1}{3} \\ \frac{1}{3} \end{pmatrix} + \begin{pmatrix} -\frac{5}{2} \\ \frac{5}{2} \\ 0 \end{pmatrix} = \begin{pmatrix} -\frac{13}{6} \\ \frac{17}{6} \\ \frac{1}{3} \end{pmatrix} \approx \begin{pmatrix} -2.167 \\ 2.833 \\ 0.333 \end{pmatrix}.$$

Looking at the approximation, you can see that it is reasonable close to the target vector $\begin{pmatrix} -2 \\ 3 \\ 0 \end{pmatrix}$, but also the difference between the target vector and the approximation is $\begin{pmatrix} \frac{1}{6} \\ \frac{1}{6} \\ -\frac{1}{3} \end{pmatrix}$, and this difference is orthogonal to both $\begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$ and $\begin{pmatrix} -1 \\ 1 \\ 0 \end{pmatrix}$.

Suppose, however, that the vectors $\mathbf{u}_1, \ldots, \mathbf{u}_m \in \mathbf{R}^n$ are not orthogonal. In this case, finding the linear combination that best approximates a given target vector is not as trivial to find—we **cannot** simply project the target vector onto each of the given vectors. For example, the best approximation of the target vector $\mathbf{y} = \begin{pmatrix} -2 \\ 3 \\ 0 \end{pmatrix}$ as a linear combination of $\begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$ and $\begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}$ is the vector $-\frac{5}{3}\begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} + \frac{1}{3}\begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}$ but the best approximation of the same target vector as a linear combination of $\begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$ and $\begin{pmatrix} 1 \\ 2 \\ 9 \end{pmatrix}$ is $\frac{1}{3}\begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} + 0\begin{pmatrix} 1 \\ 2 \\ 9 \end{pmatrix}$; that is, the scalar multiples depend on the dependencies between the two vectors. If the vectors were orthogonal, changing one of the vectors to another orthogonal vector would not affect the coefficients of the others—the projections would remain unchanged.

Thus, we will now investigate how to find the best approximation of a target vector **y** as a linear combination of a set of vectors $\mathbf{u}_1, \ldots, \mathbf{u}_m \in \mathbf{R}^n$ that is not necessarily orthogonal. Whatever algorithm we come up with should continue to work if the vectors happen to be orthogonal.

First, recall that the collection of all linear combinations of a set of vectors forms a subspace. Thus, given the two vectors $\begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$ and $\begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}$ (which are linearly independent), if we consider all possible linear combinations, e.g.,

$a_1\begin{pmatrix}1\\1\\1\end{pmatrix}+a_2\begin{pmatrix}1\\2\\3\end{pmatrix}$ where a_1 and a_2 can be any real values, this forms a plane in $\mathbf{R}^3$. Similarly, given m vectors $\mathbf{u}_1,\ldots,\mathbf{u}_m \in \mathbf{R}^n$, all linear combinations of the form $a_1\mathbf{u}_1+\cdots+a_m\mathbf{u}_m$ forms a subspace of $\mathbf{R}^n$, and if the vectors are linearly independent, they form an m-dimensional subspace of $\mathbf{R}^n$.

Now, if you have a point in $\mathbf{R}^2$, the closest point to a line passing through the origin, the closest point on that line is that point such that the difference is perpendicular to the line, as shown in Figure 1. Similarly, in $\mathbf{R}^3$, the closest point on a line passing through the origin is that point such that the difference is perpendicular to the line, and the closest point on a plane is that point perpendicular to every point on the plane, as shown in Figures 2 and 3, respecitvely.

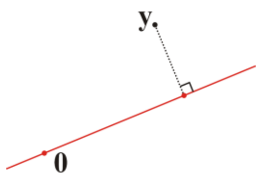


Figure 1. The best approximation of a vector **y** on a 1-dimensional subspace of $\mathbf{R}^2$.

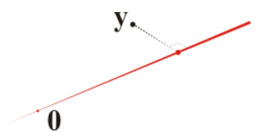


Figure 2. The best approximation of a vector **y** on a 1-dimensional subspace of $\mathbf{R}^3$.

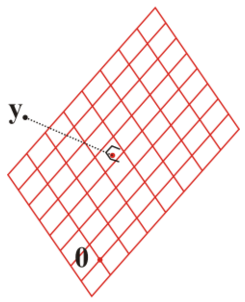


Figure 3. The best approximation of a target vector **y** on a 2-dimensional subspace of $\mathbf{R}^3$.

Now, the subspace defined by the two vectors $\begin{pmatrix}1\\1\\1\end{pmatrix}$ and $\begin{pmatrix}1\\2\\3\end{pmatrix}$ is all linear combinations of these vectors:

$$a_1\begin{pmatrix}1\\1\\1\end{pmatrix}+a_2\begin{pmatrix}1\\2\\3\end{pmatrix}.$$

You can see that this is equivalent to being the range of the matrix $U:\mathbf{R}^2\rightarrow\mathbf{R}^3$ where

$$U=\begin{pmatrix}1&1\\1&2\\1&3\end{pmatrix}$$

and where $U\mathbf{a}$ is the image of the vector $\mathbf{a} = \begin{pmatrix} a_1 \\ a_2 \end{pmatrix}$. Thus, every vector in the range of U is in the subspace defined by the two vectors $\begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$ and $\begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}$.

Note, it is important to remember that matrix-vector multiplication is nothing more or less than taking a linear combination of the column vectors of the matrix. For example, if $A = \begin{pmatrix} 3.2 & -2.6 \\ 4.7 & 0.9 \\ 8.0 & -5.4 \end{pmatrix}$ and $\mathbf{v} = \begin{pmatrix} v_1 \\ v_2 \end{pmatrix}$, you will note that the matrix-vector product $A\mathbf{v} = \begin{pmatrix} 3.2 & -2.6 \\ 4.7 & 0.9 \\ 8.0 & -5.4 \end{pmatrix}\begin{pmatrix} v_1 \\ v_2 \end{pmatrix} = \begin{pmatrix} 3.2v_1 - 2.6v_2 \\ 4.7v_1 + 0.9v_2 \\ 8.0v_1 - 5.4v_2 \end{pmatrix}$, and yet

$$v_1\begin{pmatrix} 3.2 \\ 4.7 \\ 8.0 \end{pmatrix} + v_2\begin{pmatrix} -2.6 \\ 0.9 \\ -5.4 \end{pmatrix} = \begin{pmatrix} 3.2v_1 \\ 4.7v_1 \\ 8.0v_1 \end{pmatrix} + \begin{pmatrix} -2.6v_2 \\ 0.9v_2 \\ -5.4v_2 \end{pmatrix} = \begin{pmatrix} 3.2v_1 - 2.6v_2 \\ 4.7v_1 + 0.9v_2 \\ 8.0v_1 - 5.4v_2 \end{pmatrix}$$

which is equal to the matrix-vector product $A\mathbf{v}$. This is true in general.

Now, suppose we have a matrix $A : \mathbf{R}^m \to \mathbf{R}^n$ and $\mathbf{v} \in \mathbf{R}^n$. If **v** is perpendicular to every vector in the range of *A*, it follows that $A\mathbf{u} \bullet \mathbf{v} = 0$ for every vector $\mathbf{u} \in \mathbf{R}^m$.

Now, the transpose of a matrix is that matrix A^T such that $A\mathbf{u} \bullet \mathbf{v} = \mathbf{u} \bullet A^\mathrm{T}\mathbf{v}$ for all vectors $\mathbf{u} \in \mathbf{R}^m$ and all vectors $\mathbf{v} \in \mathbf{R}^n$. For example, if $A = \begin{pmatrix} 1 & 1 \\ 1 & 2 \\ 1 & 3 \end{pmatrix}$, $\mathbf{u} = \begin{pmatrix} -1 \\ 3 \end{pmatrix}$ and $\mathbf{v} = \begin{pmatrix} 1 \\ 0 \\ -2 \end{pmatrix}$. Observe that $A\mathbf{u} = \begin{pmatrix} 1 & 1 \\ 1 & 2 \\ 1 & 3 \end{pmatrix}\begin{pmatrix} -1 \\ 3 \end{pmatrix} = \begin{pmatrix} 2 \\ 5 \\ 8 \end{pmatrix}$ and $A\mathbf{u} \bullet \mathbf{v} = \begin{pmatrix} 2 \\ 5 \\ 8 \end{pmatrix} \bullet \begin{pmatrix} 1 \\ 0 \\ -2 \end{pmatrix} = -14$, while $A^\mathrm{T}\mathbf{v} = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 2 & 3 \end{pmatrix}\begin{pmatrix} 1 \\ 0 \\ -2 \end{pmatrix} = \begin{pmatrix} -1 \\ -5 \end{pmatrix}$ and $\mathbf{u} \bullet A^\mathrm{T}\mathbf{v} = \begin{pmatrix} -1 \\ 3 \end{pmatrix} \bullet \begin{pmatrix} -1 \\ -5 \end{pmatrix} = -14$, and thus we see that $A\mathbf{u} \bullet \mathbf{v} = \mathbf{u} \bullet A^\mathrm{T}\mathbf{v}$.

Okay, so if $A\mathbf{u} \bullet \mathbf{v} = 0$ for every vector $\mathbf{u} \in \mathbf{R}^m$, then it follows that $\mathbf{u} \bullet A^\mathrm{T}\mathbf{v} = 0$ for every vector $\mathbf{u} \in \mathbf{R}^m$. Now, the only vector that is perpendicular to all vectors in $\mathbf{R}^m$ is the zero vector. Therefore, if **v** is perpendicular to $A\mathbf{u}$ for every vector in $\mathbf{u} \in \mathbf{R}^m$, it follows that $A^\mathrm{T}\mathbf{v} = \mathbf{0}$. Therefore, **v** must be in the null space of A^T.

Thus, returning to our original problem: we want to find the linear combination of the vectors $\mathbf{u}_1, \ldots, \mathbf{u}_m \in \mathbf{R}^n$ that best approximates a target vector $\mathbf{y} \in \mathbf{R}^n$. Define the matrix $U = (\mathbf{u}_1 \cdots \mathbf{u}_m)$ that maps $U : \mathbf{R}^m \to \mathbf{R}^n$ and we must find that vector $\mathbf{a} \in \mathbf{R}^m$ such that $U\mathbf{a} - \mathbf{y}$ is perpendicular to the range of *U*. Thus, from the previous paragraph, it follows that $U\mathbf{a} - \mathbf{y}$ must be in the null space of U^{T}, so it must be true that

$$U^{\mathrm{T}}(U\mathbf{a} - \mathbf{y}) = \mathbf{0}.$$

Expanding this, we have

$$U^{\mathrm{T}}U\mathbf{a} - U^{\mathrm{T}}\mathbf{y} = \mathbf{0}.$$

Bringing the vector to the right-hand side, we now have the system of linear equations defined by

$$U^{\mathrm{T}}U\mathbf{a} = U^{\mathrm{T}}\mathbf{y}.$$

Note that the dimensions make sense:

1. $\mathbf{a}$ is an *m*-dimensional vector,
2. *U* is an $n \times m$ matrix, so U^{T} is an $m \times n$ matrix, and thus $U^{\mathrm{T}}U$ is an $m \times m$ matrix.
3. $\mathbf{y}$ is an *n*-dimensional vector, and U^{T} is an $m \times n$ matrix, so $U^{\mathrm{T}}\mathbf{y}$ is an *m*-dimensional vector.

Thus, $U^{\mathrm{T}}U\mathbf{a} = U^{\mathrm{T}}\mathbf{y}$ defines a system of *m* linear equations in *m* unknowns.

Going back to our original problem: find the best approximation of the target vector $\mathbf{y} = \begin{pmatrix} -2 \\ 3 \\ 0 \end{pmatrix}$ as a linear combination of $\begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$ and $\begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}$. $U = \begin{pmatrix} 1 & 1 \\ 1 & 2 \\ 1 & 3 \end{pmatrix}$, so $U^{\mathrm{T}}U = \begin{pmatrix} 3 & 6 \\ 6 & 14 \end{pmatrix}$ and $U^{\mathrm{T}}\mathbf{y} = \begin{pmatrix} 1 \\ 4 \end{pmatrix}$, and solving this, we the solution $\mathbf{a} = \begin{pmatrix} -\frac{5}{3} \\ 1 \end{pmatrix}$, and thus the best approximation is $\tilde{\mathbf{y}} = \begin{pmatrix} -\frac{2}{3} \\ \frac{1}{3} \\ \frac{4}{3} \end{pmatrix}$. This may seem like a horrible approximation, but observe that $\mathbf{y} - \tilde{\mathbf{y}} = \begin{pmatrix} -\frac{4}{3} \\ \frac{8}{3} \\ -\frac{4}{3} \end{pmatrix}$, and this difference is perpendicular to both $\begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$ and $\begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}$—you simply cannot get a better approximation as a linear combination of these two vectors.

Let’s look at a different example: suppose we are trying to find the best approximation $\mathbf{y} = \begin{pmatrix} 0.5 \\ 1.9 \\ 2.7 \end{pmatrix}$ of the same vectors. In this case, $U^{\mathrm{T}}\mathbf{y} = \begin{pmatrix} 5.1 \\ 12.4 \end{pmatrix}$ and solving $\begin{pmatrix} 3 & 6 \\ 6 & 14 \end{pmatrix}\mathbf{a} = \begin{pmatrix} 5.1 \\ 12.4 \end{pmatrix}$ yields $\mathbf{a} = \begin{pmatrix} -0.5 \\ 1.1 \end{pmatrix}$. Thus, the best approximation of **y** is $\tilde{\mathbf{y}} = \begin{pmatrix} 0.6 \\ 1.7 \\ 2.8 \end{pmatrix}$. The difference is $\mathbf{y} - \tilde{\mathbf{y}} = \begin{pmatrix} -0.1 \\ 0.2 \\ -0.1 \end{pmatrix}$, and once again, you can see that this difference vector is orthogonal to both $\begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$ and $\begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}$.

For interest’s sake, let us look at what happens if the vectors are orthogonal. Recall that $\begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$ and $\begin{pmatrix} -1 \\ 1 \\ 0 \end{pmatrix}$ are orthogonal, and thus $U = \begin{pmatrix} 1 & -1 \\ 1 & 1 \\ 1 & 0 \end{pmatrix}$ and thus $U^{\mathrm{T}}U = \begin{pmatrix} 3 & 0 \\ 0 & 2 \end{pmatrix}$ is a diagonal matrix. If $\mathbf{y} = \begin{pmatrix} -2 \\ 3 \\ 0 \end{pmatrix}$, $U^{\mathrm{T}}\mathbf{y} = \begin{pmatrix} 1 \\ 5 \end{pmatrix}$, and thus $\mathbf{a} = \begin{pmatrix} \frac{1}{3} \\ \frac{5}{2} \end{pmatrix}$, which is the exact same linear combination we saw previously when calculating the projections.

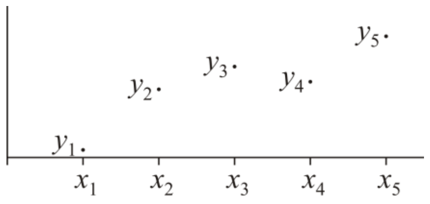
Best-fitting linear polynomial

Given n different points with different x values, we saw previously that we always find a polynomial of degree $n - 1$ that interpolates the n points exactly.

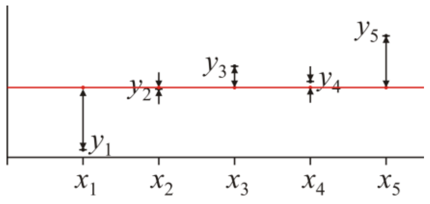
First, suppose we have n points $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$. For example, suppose we have the points

$$(1, 0.1), (2, 0.9), (3, 1.2), (4, 1.0), (5, 1.4)$$

shown here:

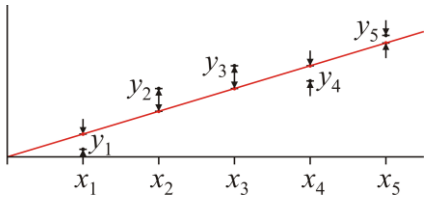


Suppose I want to find the best constant-valued polynomial that passes through these points:



This is essentially saying, what is the best approximation of the target vector $\mathbf{y} = \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \end{pmatrix}$ by a scalar multiple of the vector $\mathbf{u} = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}$. From our previous discussion, let $U = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}$ be the 5 x 1 matrix containing the given vector as the only column, and then solve $U^{\mathrm{T}} U \mathbf{a} = U^{\mathrm{T}} \mathbf{y}$ for the one-dimensional vector $\mathbf{a} = (a_0)$. In this example, $\mathbf{y} = \begin{pmatrix} 0.1 \\ 0.9 \\ 1.2 \\ 1.0 \\ 1.4 \end{pmatrix}$ and $U^{\mathrm{T}} U = (5)$ and $U^{\mathrm{T}} \mathbf{y} = (4.6)$, so you are solving a system of one linear equation in one unknown. This becomes a trivial case of calculating 4.6/5 = 0.92. Thus, the constant polynomial that best fits the given data is $y(x) = 0.92$.

Suppose I want to find the best linear polynomial that passes through the origin that also comes as close as possible to the points:



We could interpret this as a question in calculus: find that polynomial *y*(*x*) = *ax* that minimizes the sum of the squares of the errors; that is, we want to minimize $\sum_{k=1}^{5}\left(ax_k - y_k\right)^2$. However, we will interpret this in another way: we want to find that scalar multiple of $\mathbf{u} = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{pmatrix}$ that comes as close as possible to the target vector $\mathbf{y} = \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \end{pmatrix}$. As before, we set $U = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{pmatrix}$ to be the 5 x 1 matrix, and we determine that $U^\mathrm{T}U = (55)$ and $U^\mathrm{T}\mathbf{y} = (1\cdot 0.1 + 2\cdot 0.9 + 3\cdot 1.2 + 4\cdot 1.0 + 5\cdot 1.4) = (16.5)$ while $U^\mathrm{T}U = (55)$. Thus, solving this system of one linear equation in one unknown, we get that **a** = (16.5/55) = (0.3). Thus, the linear polynomial passing through the origin that best matches the data is *y*(*x*) = 0.3*x*.

Next, suppose I want to find that polynomial of the form $y(x) = ax + b$ that allows $ax_k + b$ to be as close as possible to y_k for all k. Thus, we want to find a vector $\begin{pmatrix} ax_1+b \\ ax_2+b \\ ax_3+b \\ ax_4+b \\ ax_5+b \end{pmatrix}$ which is as close as possible to the target vector $\mathbf{y} = \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \end{pmatrix}$.

Note that the vector $\begin{pmatrix} ax_1+b \\ ax_2+b \\ ax_3+b \\ ax_4+b \\ ax_5+b \end{pmatrix} = a\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{pmatrix} + b\begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}$, so we are asking, what is the linear combination of the two vectors $\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}$ that best approximates our target vector **y**? As before, we set up the matrix $U = \begin{pmatrix} x_1 & 1 \\ x_2 & 1 \\ x_3 & 1 \\ x_4 & 1 \\ x_5 & 1 \end{pmatrix}$, and create the system of two linear equations in two unknowns $U^{\mathrm{T}}U\mathbf{a} = U^{\mathrm{T}}\mathbf{y}$ where the vector $\mathbf{a} = \begin{pmatrix} a \\ b \end{pmatrix}$.

In our example, $U = \begin{pmatrix} 1 & 1 \\ 2 & 1 \\ 3 & 1 \\ 4 & 1 \\ 5 & 1 \end{pmatrix}$ and $\mathbf{y} = \begin{pmatrix} 0.1 \\ 0.9 \\ 1.2 \\ 1.0 \\ 1.4 \end{pmatrix}$, so $U^{\mathrm{T}}U = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix}\begin{pmatrix} 1 & 1 \\ 2 & 1 \\ 3 & 1 \\ 4 & 1 \\ 5 & 1 \end{pmatrix} = \begin{pmatrix} 55 & 15 \\ 15 & 5 \end{pmatrix}$ and $U^{\mathrm{T}}\mathbf{y} = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix}\begin{pmatrix} 0.1 \\ 0.9 \\ 1.2 \\ 1.0 \\ 1.4 \end{pmatrix} = \begin{pmatrix} 16.5 \\ 4.6 \end{pmatrix}$. Thus, to find $\mathbf{a} = \begin{pmatrix} a \\ b \end{pmatrix}$, we must solve $\begin{pmatrix} 55 & 15 \\ 15 & 5 \end{pmatrix}\mathbf{a} = \begin{pmatrix} 16.5 \\ 4.6 \end{pmatrix}$.

Solving this as a system of linear equations, we have

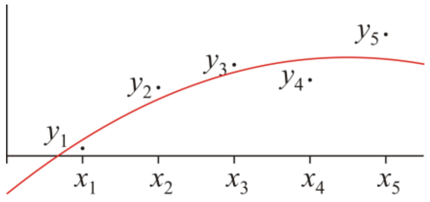
$$\left(\begin{array}{cc:c} 55 & 15 & 16.5 \\ 15 & 5 & 4.6 \end{array}\right).$$

Adding $-\frac{15}{55} = -\frac{3}{11}$ times Row 1 onto Row 2, we get

$$\left(\begin{array}{cc:c} 55 & 15 & 16.5 \\ 0 & \frac{10}{11} & 0.1 \end{array}\right).$$

Solving this using backward substitution, we get that $b = 0.11$ and therefore $a = 0.27$. Therefore, the best-fitting linear polynomial that passes as close as possible to these points is $y(x) = 0.27x + 0.11$.

Similarly, we can also find a quadratic polynomial that passes as closely as possible to a set of points:

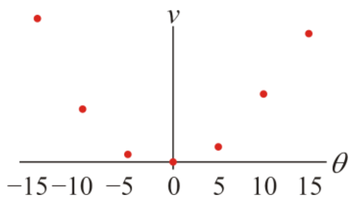


These points we have found are linear combinations of the columns of $U = \begin{pmatrix} x_1^2 & x_1 & 1 \\ x_2^2 & x_2 & 1 \\ x_3^2 & x_3 & 1 \\ x_4^2 & x_4 & 1 \\ x_5^2 & x_5 & 1 \end{pmatrix}$. Now we need to solve $U^{\mathrm{T}}U\mathbf{a} = U^{\mathrm{T}}\mathbf{y}$ for the unknown coefficient vector $\mathbf{a} = \begin{pmatrix} a \\ b \\ c \end{pmatrix}$. Multiplying out the matrices, we have the system of linear equations

$$\left(\begin{array}{ccc:c} 979 & 225 & 55 & 65.5 \\ 225 & 55 & 15 & 16.5 \\ 55 & 15 & 5 & 4.6 \end{array}\right).$$

Solving this, we get something a little more ugly: $\mathbf{a} = \begin{pmatrix} -0.0928571 \\ 0.8271429 \\ -0.54 \end{pmatrix}$. Thus, the best-fitting quadratic polynomial is $y(x) = -0.0928571x^2 + 0.8271429x - 0.54$.

As an aside, suppose you know that your system, as designed, should be responding as a pure quadratic. For example, suppose you have a joystick where the voltage is designed to be a quadratic function of the angle, so $v = a\theta^2$. If you took a number of readings of the voltage at various angles, your readings will have an error associated with them.



If you are looking for the best estimation of the coefficient a, you would **not** find the best-fitting quadratic polynomial, but rather the best-fitting polynomial of the form $v(\theta) = a\theta^2$. Thus, in this case, your U matrix would simply be the square of the angles (of course, in degrees):

$$U = \begin{pmatrix} 225 \\ 100 \\ 25 \\ 0 \\ 25 \\ 100 \\ 225 \end{pmatrix}$$

If the voltage readings were $\mathbf{v} = \begin{pmatrix} 1.9 \\ 0.7 \\ 0.1 \\ 0 \\ 0.2 \\ 0.9 \\ 1.7 \end{pmatrix}$, then you would calculate $U^TU = (122500)$ and $U^T\mathbf{v} = (977.5)$, and solving this we have that $a = 0.0080$ and we see that $0.008 \times 15^2 = 1.8$, $0.008 \times 10^2 = 0.8$ and $0.008 \times 5^2 = 0.2$, which approximate our values in the vector $\mathbf{v}$ very closely.

Now, the process of finding these polynomials can be very laborious: finding a best-fitting linear polynomial requires $2n$ multiplications, $5n - 5$ additions, all followed by the solving a system of two linear equations in two unknowns, requiring at least 4 multiplications and 3 additions, and more if pivoting is required for numerical stability. It also requires a commensurate amount of memory and memory accesses and assignments. Essentially, to find the best-fitting linear and quadratic polynomials, you must solve:

$$\begin{pmatrix} \sum_{k=1}^{n} x_k^2 & \sum_{k=1}^{n} x_k \\ \sum_{k=1}^{n} x_k & n \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} \sum_{k=1}^{n} x_k y_k \\ \sum_{k=1}^{n} y_k \end{pmatrix} \text{ and } \begin{pmatrix} \sum_{k=1}^{n} x_k^4 & \sum_{k=1}^{n} x_k^3 & \sum_{k=1}^{n} x_k^2 \\ \sum_{k=1}^{n} x_k^3 & \sum_{k=1}^{n} x_k^2 & \sum_{k=1}^{n} x_k \\ \sum_{k=1}^{n} x_k^2 & \sum_{k=1}^{n} x_k & n \end{pmatrix} \begin{pmatrix} a \\ b \\ c \end{pmatrix} = \begin{pmatrix} \sum_{k=1}^{n} x_k^2 y_k \\ \sum_{k=1}^{n} x_k y_k \\ \sum_{k=1}^{n} y_k \end{pmatrix}.$$

Fortunately, however, if the x values are equally spaced by a gap h, so $x_k = x_1 + (k - 1)h$, then many of the formulas simplify.

$$n\begin{pmatrix} x_1^2 + (n-1)hx_1 + \frac{(2n-1)(n-1)}{6}h^2 & x_1 + \frac{n-1}{2}h \\ x_1 + \frac{n-1}{2}h & 1 \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} (x_1 - h)\sum_{k=1}^{n} y_k + h\sum_{k=1}^{n} ky_k \\ \sum_{k=1}^{n} y_k \end{pmatrix}$$

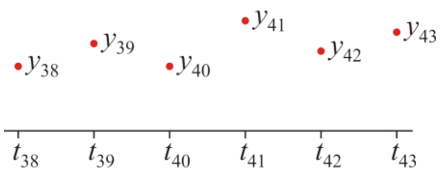
Adding $-\left(x_1 + \frac{n-1}{2}h\right)$ times Row 2 onto Row 1, we get

$$n\begin{pmatrix} (n^2-1)\frac{h^2}{12} & 0 \\ x_1 + \frac{n-1}{2}h & 1 \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} h\left(\sum_{k=1}^{n} ky_k - \frac{n+1}{2}\sum_{k=1}^{n} y_k\right) \\ \sum_{k=1}^{n} y_k \end{pmatrix}.$$

At this point, you could apply forward substitution to find a, and then substitute this value in to find b. However, what is elegant is that all of this simplifies to a simple linear combination of the y-values.

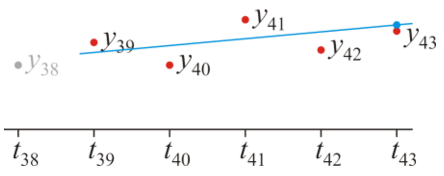
Approximations with equally spaced samples

As the previous analysis suggested, the least-squares best-fitting problem simplifies greatly if the sensor readings are taken periodically. In this section, we will assume that the sensor readings are taken with time h apart from each other. Suppose that we are currently sampling a sensor at time t_n, and the value the sensor gives is y_n. We are assuming that each of the sensor readings is associated with some form of error, and we would like to instead find the best estimate as to what a correct reading would be. For example, suppose we are taking our 40th reading:



If we are assuming we are moving at least approximately with rectilinear motion, we can see that there is a lot of noise in our sensor. Thus, we would like to find the best reasonable estimate as to where we actually are based on the current and previous data.

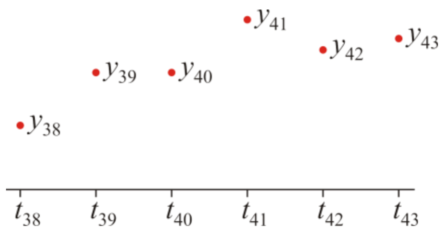
In this case, one solution would be to choose n = 5 points, and find the best-fitting least-squares line that passes through these five points:



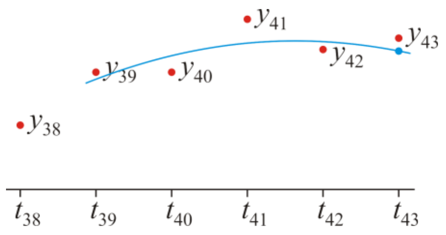
We could then evaluate this line at the point t_{43}. Fortunately, these formulas are quite straight-forward, as they all simplify to linear combinations of the current and previous y values.

Goal	Estimation
Estimate $y(t_n)$	$0.6\,y_n + 0.4\,y_{n-1} + 0.2\,y_{n-2} - 0.2\,y_{n-4}$
Estimate $y(t_n + h)$	$0.8\,y_n + 0.5\,y_{n-1} + 0.2\,y_{n-2} - 0.1\,y_{n-3} - 0.4\,y_{n-4}$
Estimate the rate of change of y over time	$\dfrac{0.2y_n + 0.1y_{n-1} - 0.1y_{n-3} - 0.2y_{n-4}}{h}$
Estimate the integral $\int_{t_n-4h}^{t_n} y(t)\,dt$	$(4h)(0.2y_n + 0.2y_{n-1} + 0.2y_{n-2} + 0.2y_{n-3} + 0.2y_{n-4})$
Estimate the integral $\int_{t_n-h}^{t_n} y(t)\,dt$	$h(0.5y_n + 0.35y_{n-1} + 0.2y_{n-2} + 0.05y_{n-3} - 0.1y_{n-4})$

Suppose, however, we know that there is acceleration occurring in our system: we are assuming our object has momentum, so rapid changes in velocity are not possible, but slow changes are expected:



At this point, finding a best-fitting line might seriously over-estimate the current position, so instead we will use an best-fitting least-squares quadratic polynomial:



If we were to evaluate this interpolating polynomial at time t_{43}, we would have a reasonable estimate as to our current location. We can also evaluate different aspects of this interpolating polynomial, but as before, the result is always a linear combination of the sensor readings:

Goal	Estimation
Estimate $y(t_n)$	$\frac{1}{35}\left(31y_n + 9y_{n-1} - 3y_{n-2} - 5y_{n-3} + 3y_{n-4}\right)$
Estimate $y(t_n + h)$	$1.8y_n - 0.8y_{n-2} - 0.6y_{n-3} + 0.6y_{n-4}$
Estimate the rate of change of y over time at time t_n	$\frac{54y_n - 13y_{n-1} - 40y_{n-2} - 27y_{n-3} + 26y_{n-4}}{70h}$
Estimate the acceleration of y over time at time t_n	$\frac{2y_n - y_{n-1} - 2y_{n-2} - y_{n-3} + 2y_{n-4}}{7h^2}$
Estimate the integral $\int_{t_n-4h}^{t_n} y(t)\,\mathrm{d}t$	$(4h)\frac{11y_n + 26y_{n-1} + 31y_{n-2} + 26y_{n-3} + 11y_{n-4}}{105}$
Estimate the integral $\int_{t_n-h}^{t_n} y(t)\,\mathrm{d}t$	$h\frac{230y_n + 137y_{n-1} + 64y_{n-2} + 11y_{n-3} - 22y_{n-4}}{420}$

Similar formulas exist for however many points you wish to include in your approximation. As with interpolation, it is dangerous to attempt to find too high a degree of best-fitting polynomial.

Other formulas may be used to estimate missing or inaccurate data. For example, suppose that the signal at time $t - h$ was lost. Can we estimate a value for that signal based on surrounding signals?

For a least-squares best-fitting linear polynomial, the formula is

$$y_{n-1} \approx \frac{1}{7}\left(4y_n + 2y_{n-2} + y_{n-3}\right),$$

which remarkably does not depend on y_{n-4}; while if we use a best fitting quadratic, we have

$$y_{n-1} \approx \frac{1}{22}\left(9y_n + 12y_{n-2} + 6y_{n-3} - 5y_{n-4}\right).$$

Alternatively, suppose that there is *jitter* in when the samples are taken. This is a common phenomenon in real-time systems, where the actual sample is taken at time $t + \varepsilon$; however, many formula provide better estimates when the time samples are equally spaced. Thus, if sample $\tilde{y}_n$ is taken at time $t_n + \varepsilon$, the best estimator of y_n is

$$y_n \approx \tilde{y}_n + \frac{\varepsilon}{h}\left(-0.2\tilde{y}_n - 0.1y_{n-1} + 0.1y_{n-3} + 0.2y_{n-4}\right) + \left(\frac{\varepsilon}{h}\right)^2\left(0.06y_{n-1} + 0.02y_{n-2} - 0.02y_{n-3} - 0.06y_{n-4}\right).$$

Note that this is an estimator of y_n, not an estimator of $y(t)$.

While division by h is potentially catastrophic in differentiation, here we have a ratio of $\frac{\varepsilon}{h}$, and hopefully the jitter will not be so large as to exceed the time interval over which samples are taken. If you were to try to estimate y_n using a best-fitting quadratic, you would get the formula

$$y_n \approx \tilde{y}_n + \frac{\varepsilon}{70h}\left(-54\tilde{y}_n + 13y_{n-1} + 40y_{n-2} + 27y_{n-3} - 26y_{n-4}\right) + \frac{1}{2450}\left(\frac{\varepsilon}{h}\right)^2\left(410\tilde{y}_n + 782y_{n-1} - 641y_{n-2} - 814y_{n-3} + 263y_{n-4}\right).$$

Again, this is an estimator of y_n, not an estimator of $y(t)$. Thus, rather than recording the actual reading, you would record the best estimator for the reading at time t_n.

Implementation

Going back to our data-collection data structure, we can estimate these values:

```
class Data_collation {
    private:
        double delta_t;
        double y_buffer[5];
        size_t last;
        double running_sum;
        double linear_combination( double coeffs[5] );

    public:
        Data_collection( double dt );
        void store_datum( double y );
        double current_y_value() const;
        double next_y_value() const;
        double next_rate_of_change() const;
        double integral() const;
};

Data_collection::Data_collection( double dt ):
delta_t{dt},
y_buffer{0.0, 0.0, 0.0, 0.0, 0.0},
last{0},
running_sum{0.0} {
    // Empty constructor
}

void Data_collection::store_datum( double y ) {
    last = (last + 1) % 5;
    y_buffer[last] = y;
    double coeffs[5] = {0.5, 0.35, 0.2, 0.05, -0.1};
    running_sum += delta_t*linear_combination( double coeffs[5] );
}

double Data_collection::linear_combination( double coeffs[5] ) {
    double result{0.0};

    for ( std::size_t k{0}; k < 5; ++k ) {
        result += coeffs[k]* y_buffer[(last - k + 5)%5];
    }
}

double Data_collection::current_y_value() const {
    double coeffs[5] = {0.6, 0.4, 0.2, 0.0, -0.2};
    return linear_combination( coeffs );
}

double Data_collection::next_y_value() const {
    double coeffs[5] = {0.8, 0.5, 0.2, -0.1, -0.4};
    return linear_combination( coeffs );
}

double Data_collection::next_rate_of_change() const {
    double coeffs[5] = {0.2, 0.1, 0.0, -0.1, -0.2};
    return linear_combination( coeffs )/delta_t;
}

double Data_collection::integral() const {
    return running_sum;
}
```

Acknowledgments

Jason Antao
Michel Georges Najarian
Mohamed Hamza Ali Baig