Least Squares Approximation in Several Dimensions

(Martin Konrad)

Problem: Given a set of sample points $S := \{s^{(1)}, ..., s^{(M)}\} \subset \mathbb{R}^{n+1}$, find a n-dimensional surface (i.e. hypersurface) $H \subset \mathbb{R}^{n+1}$ that approximates their distribution.

Idea: Linear regression (aka Least Squares Approximation)
First: Select 1 (arbitrary) variable $y := x_k$ to be dependent of the n independent variables $x_1, ...x_{k-1}, x_{k+1}, ..., x_{n+1}$, without loss of generality we may assume $k = n + 1$. To safe space, we will further write $x := (x_1, ..., x_n) \in \mathbb{R}^n$.

For given coefficients $a_0, a_1, ..., a_n \in \mathbb{R}$ we can now define

$$H(a_0, a_1, ..., a_n) := \{(x_1, ...x_n, y) \in \mathbb{R}^{n+1} : y = a_0 + a_1 x_1 + ... + a_n x_n\}. \tag{1}$$

This is a hypersurface of $\mathbb{R}^{n+1}$ (i.e. n-dimensional), as long as at least one of the coefficients is non-zero. For compactness, we will write

$$\Psi(x) := \Psi(x_1, ..., x_n) := a_0 + \sum_{k=1}^{n} a_k x_k, \tag{2}$$

so equation (1) simplifies to

$$H(a_0, a_1, ..., a_n) = \{(x, y) \in \mathbb{R}^{n+1} : y = \Psi(x)\} \tag{3}$$

To solve our initial problem, we want a set of coefficients $a_0, a_1, ..., a_n \in \mathbb{R}$, for which the *mean square error*

$$\mathrm{E}[\Psi] := \frac{1}{2} \sum_{i=1}^{M} \left(\Psi\left(s_1^{(i)}, ..., s_n^{(i)}\right) - s_{n+1}^{(i)} \right)^2 \tag{4}$$

is minimized.

Plugging in (2) into (4) yields

$$\mathrm{E}[\Psi] = \frac{1}{2} \sum_{i=1}^{M} \left(a_0 + \left[\sum_{k=1}^{n} a_k s_k^{(i)} \right] - s_{n+1}^{(i)} \right)^2 = \frac{1}{2} \|\sigma a - \tau\|_2^2, \tag{5}$$

where

$$\sigma = \begin{bmatrix} 1 & s_1^{(1)} & \cdots & s_n^{(1)} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & s_1^{(M)} & \cdots & s_n^{(M)} \end{bmatrix} \in \mathbb{R}^{M \times (n+1)}, \; a = \begin{bmatrix} a_0 \\ \vdots \\ a_n \end{bmatrix} \in \mathbb{R}^{n+1} \text{ and } \tau = \begin{bmatrix} s_{n+1}^{(1)} \\ \vdots \\ s_{n+1}^{(M)} \end{bmatrix} \in \mathbb{R}^{n+1}. \tag{6}$$

The function $a \mapsto \mathrm{E}[\Psi](a)$ in (5) is continuously differentiable with derivative

$$\nabla \mathrm{E}[\Psi](a) = \sigma^t \sigma a - \sigma^t \tau. \tag{7}$$

Thus, we can find our candidates for extrema (i.e. points with $\nabla \mathrm{E}[\Psi](a) = 0$) by solving the system of linear equations $\sigma^t \sigma a = \sigma^t \tau$. Under the condition that σ has full rank, the matrix $\sigma^t \sigma$ is positive definite, which not only means that we can invert it and get $a = \left(\sigma^t \sigma\right)^{-1} \sigma^t \tau$ as the unique solution, but also allows one to prove that this point actually is the global minimum.

For our hyperplane H from (3), this means $\Psi(x) = (1, x) \cdot a = a_0 + \sum_{k=1}^{n} a_k x_k$.

Pseudo-Code

The following code will work, if σ satisfies the rank-condition mentioned above, i.e. if σ has full rank. This assumption is no actual limitation though, as will be explained in the next paragraph.

```
INPUT
        Matrix A, so that the i-th row of A is the i-th point of S

PROGRAM
        n = (number of columns of A) - 1;
        M = number of rows of A;
        sigma = matrix(M, n+1);
        tau = matrix(M, 1);

        FOR i = 1...M

                sigma(i,0) = 1;
                tau(i) = A(i,n+1);

                FOR j = 1...n
                        sigma(i,j) = A(i,j);
                END
        END

        sigmaTrans = transpose(sigma);
        sigmaInv = inverse( sigma * sigmaTrans);

OUTPUT
        a = sigmaInv * sigmaTrans * tau;
```

Remark: Depending on M, the matrices involved are possibly very large. For implementing this code efficiently, one should either use libraries dedicated to fast matrix computations, or alternatively use a matrix-based language like `Matlab`.

Discussion of the rank-condition on σ
Let us have another look at the matrix σ from (6):

$$\sigma = \left[v_0|\cdots|v_n\right] = \begin{bmatrix} 1 & s_1^{(1)} & \cdots & s_n^{(1)} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & s_1^{(M)} & \cdots & s_n^{(M)} \end{bmatrix} \in \mathbb{R}^{M\times(n+1)} \tag{8}$$

Here we denote the k-th column with the vector $v_k \in \mathbb{R}^M$.

Since in general there will be a lot more data points $s^{(i)}$ than dimensions (i.e. $n << M$), the rank-condition for σ means that all the vectors $v_0, v_1, ..., v_n$ need to be linearly independent.

If we ignore v_0 first and just assume linear dependence of the remaining vectors, we can write at least one v_k as a linear combination of the others:

$$v_k = \sum_{j\neq k} c_j v_j \text{ for some } c_1, ..., c_{k-1}, c_k, ..., c_n \in \mathbb{R}$$

But since v_k contains *the k-th component of every point* $s^{(i)} \in S$, this implies that S is contained inside

$$M = \left\{ (x,y) \in \mathbb{R}^{n+1} : \begin{bmatrix} x_k \\ y \end{bmatrix} = \begin{bmatrix} \sum_{j\neq k} c_j x_j \\ \Psi\left(x_1, ..., \sum_{j\neq k} c_j x_j, ..., x_n\right) \end{bmatrix} \right\}, \tag{9}$$

which is a (linear) submanifold of $\mathbb{R}^{n+1}$ that has dimension smaller or equal to n.

Should any vector v_k be linearly dependent on v_0, the story is very similiar: Because v_0 was just filled with ones, any linearly dependent vector has to be filled with some constant $c \in \mathbb{R}$. And because, again, v_k consists of the k-th components of the points in S, this just tells us that S is contained inside the manifold

$$M = \left\{ (x,y) \in \mathbb{R}^{n+1} : \begin{bmatrix} x_k \\ y \end{bmatrix} = \begin{bmatrix} \lambda \\ \Psi\left(x_1, ..., \lambda, ..., x_n\right) \end{bmatrix} \right\}. \tag{10}$$

In any case, we can plug in the found function for x_k from (9) or (10) into (2) to get a new Ψ which now only has $n-1$ variables. After iterating this procedure a finite number of times (at maximum n times, in which case S would just be a 1-dimensional line), σ will have full rank and we will be able to apply our algorithm.

Remark: When the given set of points S comes from experimental data, either case is extremely unlikely and the algorithm should work just fine.