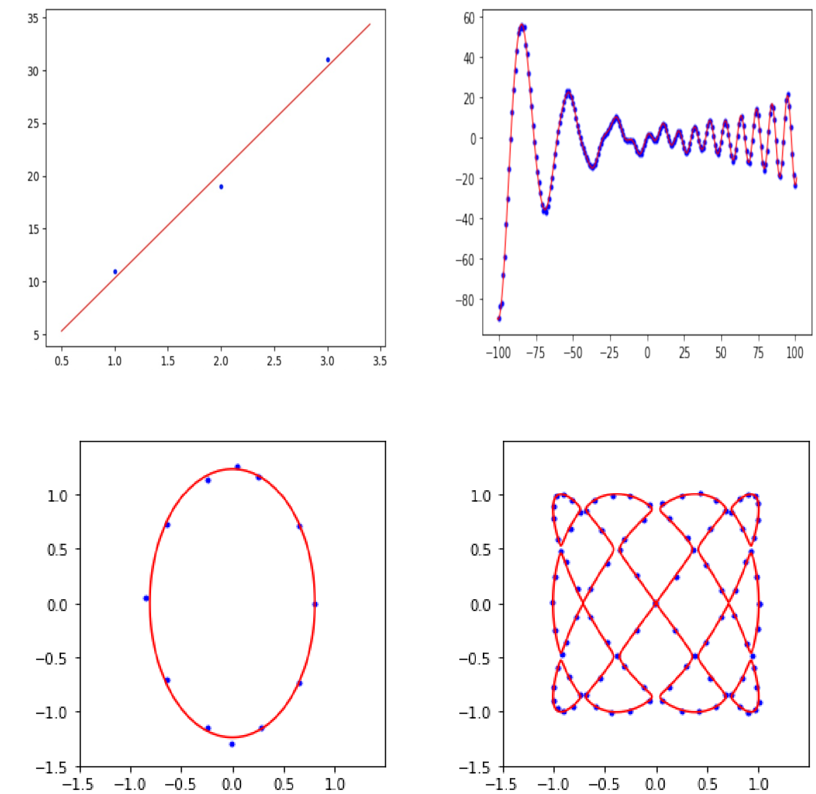
Data Fitting

Grant McNaughton

10 February 2024

We informally describe the data fitting problem by the following plots.



Roughly put, the problem is, given "data" (blue points), find a "model"(red curve) that fits the data "well". A few remarks.

1. The above plots are on the plane (2-dim space). However, many real-life problems involve higher dimension space. Thus we will need to develop theory and algorithms for arbitrary dimension.

2. Note that the top two cases could be fitted to a form $y = f(x)$ ("functional" model) , while the bottom two could not be fitted naturally to a functional model and hence we will need to fit to a form $f(x_1, x_2) = 0$ ("relational" model). Hence we will divide the paper into two parts: functional model and relational model.

3. In order to solve the problems, we will need to make the informal notions such as "data", "model" and "well". This is what we will do in the following.

Part I
Functional Model

1 Problem

We will search for a good f from linear combinations of a given model basis functions $b = (b_1, \ldots, b_n)$. Then we have the following statement of the problem.

Problem 1

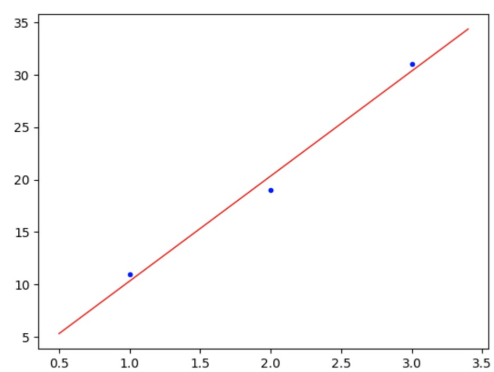
In:	$x \in \mathbb{R}^{m \times d}, \quad y \in \mathbb{R}^m$	*m points in d-dim space and corresponding values*
	$b \in \mathbb{R}^d \to \mathbb{R}^n$	*model basis functions*
Out:	$c \in \mathbb{R}^n$ *such that* $f = bc$ *fits* (x, y) *well.*	*linear combination coefficients*

Example 2 (Running) *We will use the following as a running example throughout this part.*

Input: $$x = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} \in \mathbb{R}^{3\times 1}, \; y = \begin{bmatrix} 11 \\ 19 \\ 31 \end{bmatrix} \in \mathbb{R}^3, \; b = \begin{bmatrix} x & 1 \end{bmatrix} \in \mathbb{R}^1 \to \mathbb{R}^2$$

Output: $$c = \begin{bmatrix} 10 \\ \frac{1}{3} \end{bmatrix} \in \mathbb{R}^2$$

giving $f = bc = 10x + \frac{1}{3}$



2 Theory

Notation 3 *We will use the following notations.*

1. $x = \begin{bmatrix} x_1 \\ \vdots \\ x_m \end{bmatrix} = \begin{bmatrix} x_{11} & \cdots & x_{1d} \\ \vdots & & \vdots \\ x_{m1} & \cdots & x_{md} \end{bmatrix}$

2. $y = \begin{bmatrix} y_1 \\ \vdots \\ y_m \end{bmatrix}$

3. $b = \begin{bmatrix} b_1 & \cdots & b_n \end{bmatrix}$

4. $c = \begin{bmatrix} c_1 \\ \vdots \\ c_m \end{bmatrix}$

2.1 Reducing data fitting to over-determined system solving

We need to find c such that

$$\begin{aligned} y_1 &= c_1 b_1 (x_1) + \cdots + c_n b_n (x_1) \\ &\vdots \\ y_m &= c_1 b_1 (x_m) + \cdots + c_n b_n (x_m) \end{aligned}$$

Using matrix/vector, we have

$$\begin{bmatrix} y_1 \\ \vdots \\ y_m \end{bmatrix} = \begin{bmatrix} b_1 (x_1) & \cdots & b_n (x_1) \\ \vdots & & \vdots \\ b_1 (x_m) & \cdots & b_n (x_m) \end{bmatrix} \begin{bmatrix} c_1 \\ \vdots \\ c_n \end{bmatrix}$$

Compactly put, we have

$$y = Bc$$

where

$$B = \begin{bmatrix} b_1 (x_1) & \cdots & b_n (x_1) \\ \vdots & & \vdots \\ b_1 (x_m) & \cdots & b_n (x_m) \end{bmatrix} \in \mathbb{R}^{m \times n}$$

is the "generalized Vandermonde matrix". Thus we have reduced the data fitting problem to the linear system of equations solving.

Input: $B \in \mathbb{R}^{m \times n}$

$y \in \mathbb{R}^m$

Output: $c \in \mathbb{R}^n$ such that $y = Bc$

Note that usually $m \gg n$. Hence we have over-constrained problem. Hence in general there is no solution. Hence, we will try to find c which is "close" to being a solution. The following subsection elaborate on this.

2.2 Solving over-constrained system

From our equation $Bc = y$ we want to find $\bar{c} = My$ for some matrix computation M. To do this, we can use the following logic:

$$
\begin{aligned}
B\bar{c} &= y \\
B^\intercal B\bar{c} &= B^\intercal y \\
(B^\intercal B)^{-1}(B^\intercal B)\bar{c} &= (B^\intercal B)^{-1}B^\intercal y \\
\bar{c} &= (B^\intercal B)^{-1}B^\intercal y
\end{aligned}
$$

Note that we must use $B^\intercal B$ instead of just B to ensure the matrix is invertible. This allows us to find an approximation of c even though our system is overconstrained and has no perfect solution.

2.3 Solving over-constrained system via SVD

As B grows, it quickly becomes computationally costly to find $(B^\intercal B)^{-1}$. One way to avoid this problem is to use singular value decomposition, which is much easier for large matrices B.

Let us first define B with the singular value decomposition $B = U\Sigma V^\intercal$ for an orthogonal matrix U composed of the left singular vectors of $BB^\intercal$, an orthogonal matrix V composed of the right singular vectors of $B^\intercal B$, and a diagonal matrix Σ composed of the singular values of B. The singular vectors and values are defined as the set of eigenvectors and eigenvalues satisfying $BB^\intercal u_i = \sigma_i^2 u_i$ or $B^\intercal B v_i = \sigma_i^2 v_i$.

We can also describe B as

$$
B = U\Sigma V^\intercal = \begin{bmatrix} | & | & & | \\ u_1 & u_2 & \dots & u_m \\ | & | & & | \end{bmatrix} \begin{bmatrix} \sigma_1 & 0 & \dots & 0 & 0 & \dots & 0 \\ 0 & \sigma_2 & \dots & 0 & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & & \vdots \\ 0 & 0 & \dots & \sigma_r & 0 & \dots & 0 \\ 0 & 0 & \dots & 0 & 0 & \dots & 0 \\ \vdots & \vdots & & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 0 & 0 & \dots & 0 \end{bmatrix} \begin{bmatrix} \text{---} & v_1^\intercal & \text{---} \\ \text{---} & v_2^\intercal & \text{---} \\ & \vdots & \\ \text{---} & v_n^\intercal & \text{---} \end{bmatrix}
$$

We can then perform the following calculations to get our SVD-based model fitting:

$$
\begin{aligned}
\bar{c} &= (B^\intercal B)^{-1}B^\intercal y \\
&= ((U\Sigma V^\intercal)^\intercal (U\Sigma V^\intercal))^{-1} V\Sigma U^\intercal y \\
&= (V\Sigma U^\intercal U\Sigma V^\intercal)^{-1} V\Sigma U^\intercal y \\
&= (V^\intercal)^{-1}\Sigma^{-1}U^{-1}(U^\intercal)^{-1}\Sigma^{-1}V^{-1}V\Sigma U^\intercal y \\
&= (V^\intercal)^{-1}\Sigma^{-1}U^{-1}y \\
&= V\Sigma^{-1}U^\intercal y
\end{aligned}
$$

Note that because U and V are orthogonal, $U^\intercal = U^{-1}$ and $V^\intercal = V^{-1}$. This final $\bar{c}$ can be used to quickly calculate our model.

3 Algorithm

Algorithm 4

1. $x = \begin{bmatrix} \text{---} & x_1 & \text{---} \\ & \vdots & \\ \text{---} & x_m & \text{---} \end{bmatrix}, y = \begin{bmatrix} y_1 \\ \vdots \\ y_m \end{bmatrix}, b = \begin{bmatrix} b_1(x) & \cdots & b_n(x) \end{bmatrix}$

2. $B = \begin{bmatrix} b_1(x_1) & \cdots & b_n(x_1) \\ \vdots & & \vdots \\ b_1(x_m) & \cdots & b_n(x_m) \end{bmatrix}$

3. $U, \Sigma, V = \mathrm{SVD}(B)$

4. $\bar{c} = V\Sigma^{-1}U^{\intercal}y$

Example 5

1. $x \leftarrow \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}, y \leftarrow \begin{bmatrix} 11 \\ 19 \\ 31 \end{bmatrix}, b \leftarrow \begin{bmatrix} x & 1 \end{bmatrix}$

2. $B \leftarrow \begin{bmatrix} 1 & 1 \\ 2 & 1 \\ 3 & 1 \end{bmatrix}$

3. $U, \Sigma, V \leftarrow SVD(B) \approx \begin{bmatrix} -.85 & .32 & .41 \\ -.18 & .55 & -.82 \\ .49 & .77 & .41 \end{bmatrix}, \begin{bmatrix} .6 & 0 \\ 0 & 4.08 \\ 0 & 0 \end{bmatrix}, \begin{bmatrix} -.91 & .4 \\ .4 & .91 \end{bmatrix}$

4. $\bar{c} \leftarrow \begin{bmatrix} \frac{1}{3} \\ 10 \end{bmatrix}$

Part II
Relational Model

4 Problem

We will search for a good b from linear combinations of a given model basis functions $b = (b_1, \ldots, b_n)$. Then we have the following statement of the problem.

Problem 6

In:	$x \in \mathbb{R}^{m \times d}$		*m unlabeled points in d-dim space*
	$b \in \mathbb{R}^d \to \mathbb{R}^n$		*model basis functions*
Out:	$c \in \mathbb{R}^n$ *such that* $f = bc$ *fits* (x, y) *well.*		*linear combination coefficients*

5 Theory

5.1 Idea of unlabeled relation fitting

We need to find c such that $c_1 b_1(x) + c_2 b_2(x) + \cdots + c_n b_n(x) = 0$. Using matrix/vector notation, we have

$$\begin{bmatrix} b_1(x_1) & \cdots & b_n(x_1) \\ \vdots & \ddots & \vdots \\ b_1(x_m) & \cdots & b_n(x_n) \end{bmatrix} \begin{bmatrix} c_1 \\ \vdots \\ c_n \end{bmatrix} = 0$$

Compactly put, we have

$$Bc = 0$$

where, once again,

$$B = \begin{bmatrix} b_1(x_1) & \cdots & b_n(x_1) \\ \vdots & & \vdots \\ b_1(x_m) & \cdots & b_n(x_m) \end{bmatrix} \in \mathbb{R}^{m \times n}$$

is the "generalized Vandermonde matrix".

5.2 Issues with this method

This method of relation fitting presents two main issues. The first of these issues is that the trivial solution $c = \vec{0}$ will always fit our model as the easiest to compute result. To deal this this challenge, we will set $||c|| = 1$, preventing $\vec{0}$ from being a possible solution.

The second main issue with our proposed method of relation fitting is that, once again, usually $m \gg n$. As with function fitting, we can best solve our overconstrained system by minimizing $||Bc||$ for our system instead of finding B, c such that $||Bc|| = \vec{0}$.

5.3 Solving the relation fitting problem via eigendecomposition

Consider $||Bc|| = (Bc)^\intercal Bc$. Let $S = B^\intercal B$ and $||Bc|| = c^\intercal B^\intercal Bc = c^\intercal Sc$. Set $V, \Lambda = \text{eigendecomposition}(S)$.

Because S must be symmetric based on its definition, all of its eigenvalues $\lambda_1, \ldots, \lambda_n$ are real and all eigenvector columns of V are orthogonal. Because V is an orthonormal matrix, we can express c as a linear combination of its vectors as follows:

$$c = a_1 v_1 + a_2 v_2 + \cdots + a_n v_n$$

for some $a_1, a_2, \ldots, a_n$. Let us now return to our main problem,

$$\begin{aligned}
\min(c^\intercal Sc) &= c^\intercal S(a_1 v_1 + a_2 v_2 + \cdots + a_n v_n) \\
&= c^\intercal (Sa_1 v_1 + Sa_2 v_2 + \cdots + Sa_n v_n) \\
&= c^\intercal (a_1 Sv_1 + a_2 Sv_2 + \cdots + a_n Sv_n) \\
&= c^\intercal (a_1 \lambda_1 v_1 + a_2 \lambda_2 v_2 + \cdots + a_n \lambda_n v_n) \\
&= (a_1 v_1^\intercal + a_2 v_2^\intercal + \cdots + a_n v_n^\intercal)(a_1 \lambda_1 v_1 + a_2 \lambda_2 v_2 + \cdots + a_n \lambda_n v_n)
\end{aligned}$$

Because all vectors v_i are orthogonal, $v_i^\intercal v_i = 1$ and $v_i^\intercal v_j = 0$ for all i, j in $[1, n]$. Thus we can continue our computation

$$\begin{aligned}
\min(c^\intercal Sc) &= (a_1 v_1^\intercal + a_2 v_2^\intercal + \cdots + a_n v_n^\intercal)(a_1 \lambda_1 v_1 + a_2 \lambda_2 v_2 + \cdots + a_n \lambda_n v_n) \\
&= \lambda_1 a_1^2 + \lambda_2 a_2^2 + \cdots + \lambda_n a_n^2
\end{aligned}$$

Without loss of generality, let the eigenvalues λ_i be sorted such that $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$. We also know that $||c|| = 1$, thus $\sum_{i=1}^{n} a_i^2 = 1$. We can use this to reach the final conclusion that

$$\begin{aligned}
\min(||Bc||) &= \min(c^\intercal Sc) \\
&= \min(\lambda_1 a_1^2 + \lambda_2 a_2^2 + \cdots + \lambda_n a_n^2) \\
&= 0 \cdot \lambda_1 + 0 \cdot \lambda_2 + \cdots + 0 \cdot lambda_{n-1} + 1 \cdot \lambda_n \\
&= \lambda_n
\end{aligned}$$

Thus $c^\intercal Sc$ is minimized exactly when c is the eigenvector corresponding to the smallest eigenvalue of S.

6 Algorithm

1. $x = \begin{bmatrix} — & x_1 & — \\ & \vdots & \\ — & x_m & — \end{bmatrix}, b = \begin{bmatrix} b_1(x) & \cdots & b_n(x) \end{bmatrix}$
2. $B = \begin{bmatrix} b_1(x_1) & \cdots & b_n(x_1) \\ \vdots & & \vdots \\ b_1(x_m) & \cdots & b_n(x_m) \end{bmatrix}$
3. $S = B^\intercal B$
4. $\Lambda, V = \text{eigendecomposition}(S)$
5. k = index of the smallest eigenvalue of S
6. $c = k^{\text{th}}$ eigenvector