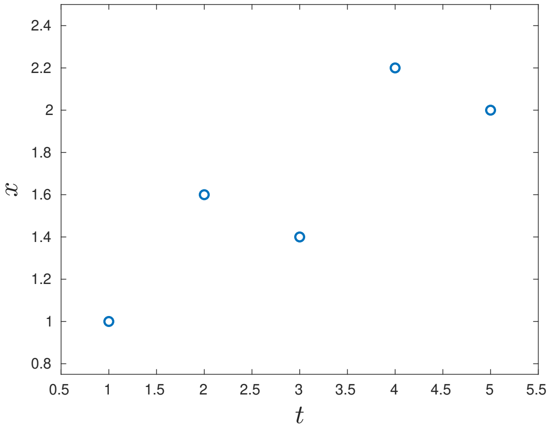
Trajectory Smoothing by Polynomial Regression using Singular Value Decomposition

Yi-Chen Zhang

September 10, 2018

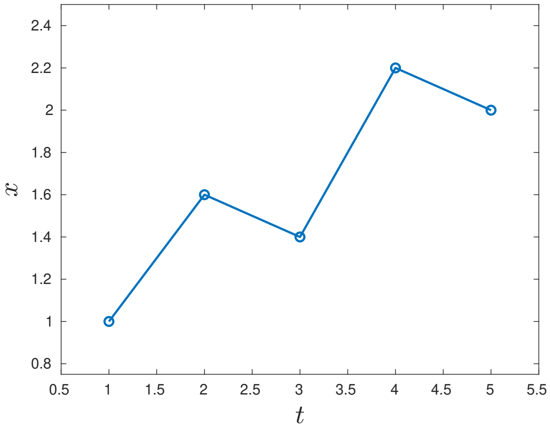
Data Points

t	1	2	3	4	5
x	1	1.6	1.4	2.2	2.0



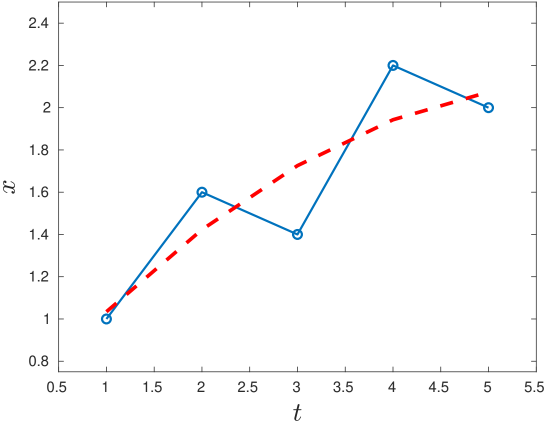
Data Points

t	1	2	3	4	5
x	1	1.6	1.4	2.2	2.0



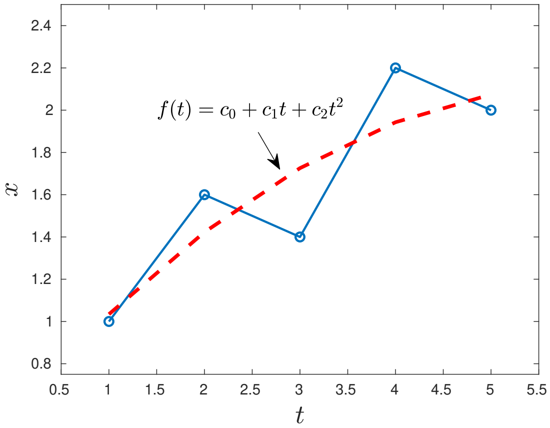
Smoothing Trajectory

t	1	2	3	4	5
x	1	1.6	1.4	2.2	2.0



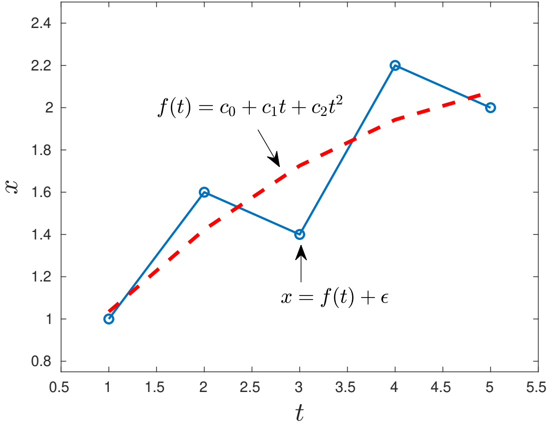
Smoothing Trajectory

t	1	2	3	4	5
x	1	1.6	1.4	2.2	2.0



Smoothing Trajectory

t	1	2	3	4	5
x	1	1.6	1.4	2.2	2.0



Polynomial Regression

We can write each data point as a function of t plus a measurement erros ϵ (perturbation), i.e.,

$$\begin{aligned} x_1 &= c_0 + c_1 t_1 + c_2 t_1^2 + \epsilon_1 \\ x_2 &= c_0 + c_1 t_2 + c_2 t_2^2 + \epsilon_2 \\ &\vdots \\ x_n &= c_0 + c_1 t_n + c_2 t_2^2 + \epsilon_n \end{aligned}$$

We further write the above equation into a matrix representation

$$\Rightarrow \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} 1 & t_1 & t_1^2 \\ 1 & t_2 & t_2^2 \\ \vdots & \vdots & \vdots \\ 1 & t_n & t_n^2 \end{pmatrix} \begin{pmatrix} c_0 \\ c_1 \\ c_2 \end{pmatrix} + \begin{pmatrix} \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \epsilon_n \end{pmatrix}$$

$$\Rightarrow \quad \boldsymbol{x} = \boldsymbol{S} \quad \boldsymbol{c} + \boldsymbol{\epsilon}$$

To find the polynomial function $f(t)$ is equivalent to find $\boldsymbol{c} = (c_0, c_1, c_2)$. The best solution of $\boldsymbol{c}$ shall minimize the overall measurement error $\|\epsilon\|$.

$$\begin{aligned} \|\epsilon\|^2 &= \|\boldsymbol{Sc} - \boldsymbol{x}\|^2 \\ &= (\boldsymbol{Sc} - \boldsymbol{x})^T(\boldsymbol{Sc} - \boldsymbol{x}) \\ &= \boldsymbol{c}^T\boldsymbol{S}^T\boldsymbol{Sc} - \boldsymbol{c}^T\boldsymbol{S}^T\boldsymbol{x} - \boldsymbol{x}^T\boldsymbol{Sc} + \boldsymbol{x}^T\boldsymbol{x} \end{aligned}$$

Now, taking derivative with respect to $\boldsymbol{c}$ and let the gradient to be zero,

$$\frac{d}{d\boldsymbol{c}}\|\epsilon\|^2 = 2\boldsymbol{S}^T\boldsymbol{Sc} - 2\boldsymbol{S}^T\boldsymbol{x} \overset{Let}{=} 0$$

The normal equation is then derived

$$\Rightarrow \boldsymbol{S}^T\boldsymbol{Sc} = \boldsymbol{S}^T\boldsymbol{x}$$

The solution of $\boldsymbol{c}$ is

$$\boldsymbol{c} = (\boldsymbol{S}^T\boldsymbol{S})^{-1}\boldsymbol{S}^T\boldsymbol{x}$$

The polynomial function $f(t)$ can be estimated by $\hat{f}(t) = \hat{c}_0 + \hat{c}_1 t + \hat{c}_2 t^2$, where $\hat{\boldsymbol{c}} = (\hat{c}_0, \hat{c}_1, \hat{c}_2)$ is the solution of the normal equation.

Sngular Value Decomposition

For any matrix $\boldsymbol{S} \in \mathbb{R}^{m\times n}$ there exist orthogonal matrices $\boldsymbol{U} \in \mathbb{R}^{m\times m}$, $\boldsymbol{V} \in \mathbb{R}^{n\times n}$, and a 'diagonal' matrix $\boldsymbol{D} \in \mathbb{R}^{m\times n}$, i.e.,

$$\boldsymbol{D} = \begin{pmatrix} d_1 & & & & & 0 & \cdots & 0 \\ & \ddots & & & & & & \\ & & d_r & & & & & \\ & & & 0 & & & & \\ & & & & \ddots & & & \\ & & & & & 0 & \cdots & 0 \end{pmatrix} \quad \text{for} \quad m \leq n$$

with diagonal entries

$$d_1 \geq \cdots \geq d_r > d_{r+1} = \cdots = d_{\min\{m,n\}} = 0$$

such that $\boldsymbol{S} = \boldsymbol{U}\boldsymbol{D}\boldsymbol{V}^T$.

Sngular Value Decomposition

For any matrix $\boldsymbol{S} \in \mathbb{R}^{m \times n}$ there exist orthogonal matrices $\boldsymbol{U} \in \mathbb{R}^{m \times m}$, $\boldsymbol{V} \in \mathbb{R}^{n \times n}$, and a 'diagonal' matrix $\boldsymbol{D} \in \mathbb{R}^{m \times n}$, i.e.,

$$\boldsymbol{D} = \begin{pmatrix} d_1 & & & & & \\ & \ddots & & & & \\ & & d_r & & & \\ & & & 0 & & \\ & & & & \ddots & \\ 0 & & & & & 0 \\ \vdots & & & & & \vdots \\ 0 & & & & & 0 \end{pmatrix} \quad \text{for} \quad m \geq n$$

with diagonal entries

$$d_1 \geq \cdots \geq d_r > d_{r+1} = \cdots = d_{\min\{m,n\}} = 0$$

such that $\boldsymbol{S} = \boldsymbol{U}\boldsymbol{D}\boldsymbol{V}^T$.

- The decomposition

$$\boldsymbol{S} = \boldsymbol{U}\boldsymbol{D}\boldsymbol{V}^T$$

  is called **Singular Value Decomposition** (SVD).
- The diagonal entries d_i of $\boldsymbol{D}$ are called the singular values of $\boldsymbol{S}$.
- The columns of $\boldsymbol{U}$ are called left singular vectors and the column of $\boldsymbol{V}$ are called right singular vectors.
- Both of $\boldsymbol{U}$ and $\boldsymbol{V}$ are orthogonal matrices, i.e.,

$$\boldsymbol{U}\boldsymbol{U}^T = \boldsymbol{U}^T\boldsymbol{U} = \boldsymbol{I}_m \text{ and } \boldsymbol{V}\boldsymbol{V}^T = \boldsymbol{V}^T\boldsymbol{V} = \boldsymbol{I}_n$$

- The matrix $\boldsymbol{S}$ can be approximated by first few non-zero singular values (say r) and vectors, i.e.,

$$\boldsymbol{S} \approx \boldsymbol{U}_r\boldsymbol{D}_r\boldsymbol{V}_r^T.$$

  This step will smooth out the noise and keep only the important information.
- We denoted the approximated matrix $\boldsymbol{S}$ by $\boldsymbol{S}_r$

Using the SVD of S for Polynomial Regression

Now we can use the SVD of $\boldsymbol{S}$ to solve the polynomial regression.

$$
\begin{aligned}
\hat{\boldsymbol{c}} &= (\boldsymbol{S}^T\boldsymbol{S})^{-1}\boldsymbol{S}^T\boldsymbol{x} \\
&= \left((\boldsymbol{U}\boldsymbol{D}\boldsymbol{V}^T)^T(\boldsymbol{U}\boldsymbol{D}\boldsymbol{V}^T)\right)^{-1}(\boldsymbol{U}\boldsymbol{D}\boldsymbol{V}^T)^T\boldsymbol{x} \\
&= \cdots \\
&= \boldsymbol{V}\boldsymbol{D}^{-1}\boldsymbol{U}^T\boldsymbol{x} \\
&\approx \boldsymbol{V}_r\boldsymbol{D}_r^{-1}\boldsymbol{U}_r^T\boldsymbol{x}
\end{aligned}
$$

We do the following steps:

- Find the number of first few non-zero singular values, say r.
- Compute $\hat{\boldsymbol{c}} = \boldsymbol{V}_r\boldsymbol{D}_r^{-1}\boldsymbol{U}_r^T\boldsymbol{x}$

The smoothed function is $\hat{f}(t) = \hat{c}_0 + \hat{c}_1 t + \hat{c}_2 t^2$

MATLAB Code

```
n = 5;
t = [1:n]';
x = [1,1.6,1.4,2.2,2.0]';

% plot data
plot(t,x,'-o', 'LineWidth',1.5);
hold on;
xlabel('$$t$$','FontSize',18,'Interpreter','latex');
ylabel('$$x$$','FontSize',18,'Interpreter','latex');
xlim([0.5,5.5]);
ylim([0.75,2.5]);
```

```
% write data into matrix
S = [ones(n,1) t t.^2];

% SVD
[U,D,V] = svd(S)

% compute the coefficient
r = 3;
c = V(1:r,1:r)*D(1:r,1:r)^(-1)*U(1:n,1:r)'*x;

% find the smoothing trajectory
x_ = c(1)+c(2)*t+c(3)*t.^2;

% plot the smoothing trajectory
plot(t,x_, '--r', 'LineWidth', 2.5);
```