Regression and Interpolation

CS 740

Ajit Rajwade

Problem statements

- Consider a set of N points $\{(x_i,y_i)\}$, $1 \leq i \leq N$.
- Suppose we know that these points actually lie on a function of the form $y = f(x;\mathbf{a})$ where f(.) represents a function family and **a** represents a set of parameters.
- For example: f(x) is a linear function of x, i.e. of the form $y = f(x) = mx+c$. In this case, **a** = (m,c).

Problem statements

- Example 2: f(x) is a quadratic function of x, i.e. of the form y = f(x) = px^2+qx+r. In this case, **a** = (p,q,r).

- Example 3: f(x) is a trigonometric function of x, i.e. of the form f(x) = p sin($qx+r$). In this case, **a** = (p,q,r).

- In each case, we assume knowledge of the function family. But we do not know the function parameters, and would like to estimate them from $\{(x_i,y_i)\}$.

- This is the problem of fitting a function (of known family) to a set of points.

Problem statements

- In function **regression** (or **approximation**), we want to find **a** such that for all *i*, $f(x_i;\mathbf{a})\approx y_i$.

- In function **interpolation**, we want to fit some function such that $f(x_i;\mathbf{a})=y_i$.

Polynomial Regression

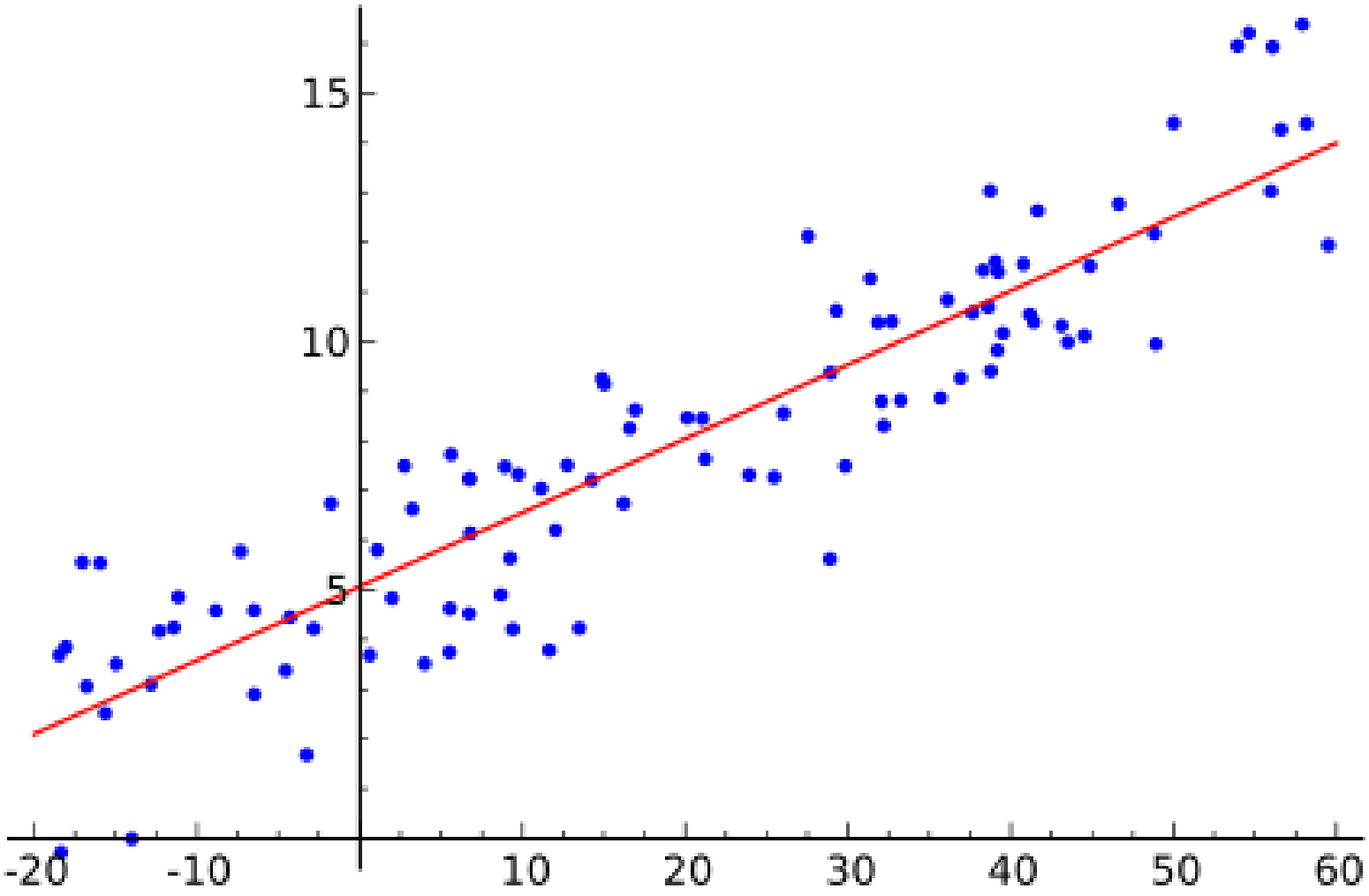
- A polynomial of degree *n* is a function of the form:

$$y = \sum_{i=0}^{n} a_i x^i = a_0 + a_1 x + a_2 x^2 + \ldots + a_n x^n$$

- Polynomial regression is the task of fitting a polynomial function to a set of points.

Polynomial regression

- Let us assume that *x* is the independent variable, and *y* is the dependent variable.
- In the point set $\{(x_i,y_i)\}$ containing *N* points, we will assume that the *x*-coordinates of the points are available accurately, whereas the *y*-coordinates are affected by measurement error – called as noise.


15
10
5
-20
-10
10
20
30
40
50
60

Least Squares Polynomial regression

- If we assume the polynomial is linear, we have $y_i = mx_i + c + e_i$, where e_i is the noise in y_i. We want to estimate *m* and *c*.
- We will do so by minimizing the following w.r.t. *m* and *c*:

$$\sum_{i=0}^{N}(y_i - mx_i - c)^2$$

- This is called as **least squares regression**.

Least Squares Polynomial regression

- In matrix, form we can write the following

$$\begin{pmatrix} y_1 \\ y_2 \\ . \\ . \\ y_N \end{pmatrix} = \begin{pmatrix} x_1 & 1 \\ x_2 & 1 \\ . & . \\ . & . \\ x_N & 1 \end{pmatrix} \begin{pmatrix} m \\ c \end{pmatrix} + \begin{pmatrix} e_1 \\ e_2 \\ . \\ . \\ e_N \end{pmatrix}$$

$$\mathbf{y} = \mathbf{X}\mathbf{a}$$

$$\mathbf{a} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y} = \mathbf{X}^+\mathbf{y}$$

The solution will exist provided there are at least 2 non-coincident points. Basically, $\mathbf{X}^T\mathbf{X}$ must be non-singular.

Least Squares Polynomial regression

- For higher order polynomials, we have:

$$\begin{pmatrix} y_1 \\ y_2 \\ . \\ . \\ y_N \end{pmatrix} = \begin{pmatrix} x_1^n & . & . & x_1^2 & x_1 & 1 \\ x_2^n & . & . & x_2^2 & x_2 & 1 \\ . & & & . & . & . \\ . & & & . & . & . \\ x_N^n & . & . & x_N^2 & x_N & 1 \end{pmatrix} \begin{pmatrix} a_n \\ . \\ . \\ a_2 \\ a_1 \\ a_0 \end{pmatrix} + \begin{pmatrix} e_1 \\ e_2 \\ . \\ . \\ e_N \end{pmatrix}$$

$$\mathbf{y} = \mathbf{X}\mathbf{a}$$

$$\mathbf{a} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y} = \mathbf{X}^+\mathbf{y}$$

The solution will exist provided there are at least *n* non-coincident, non-collinear points. Basically, $\mathbf{X}^T\mathbf{X}$ must be non-singular.

Least Squares Polynomial regression

- Regardless of the order of the polynomial, we are dealing with a linear form of regression – because in each case, we are solving an equation of the form $\mathbf{y} \approx \mathbf{Xa}$.

- There are some assumptions made on the errors in the measurement of **y**. We will not go into those details.

Least Squares Polynomial Regression: Geometric Interpretation

- The least squares solution seeks a set of polynomial coefficients that minimize the sum of the squared difference between the **measured y coordinate of each point** and the **y coordinate if the assumed polynomial model was strictly obeyed**, i.e.

$$\min_{\{a_j\}} \sum_{i=0}^{N} (\mathbf{y}_i - \sum_{j=0}^{n} a_j \mathbf{x}_i^j)^2$$

$$\equiv$$

$$\min_{\{\mathbf{a}\}} \|\mathbf{y} - \mathbf{X}\mathbf{a}\|_2^2$$

Least Squares Polynomial Regression: Geometric Interpretation

- The least squares solution seeks a set of polynomial coefficients that minimize the sum of the squared difference between the **measured y coordinate of each point** and the **y coordinate if the assumed polynomial model was strictly obeyed**, i.e. we are trying to find a model that will minimize **vertical distances** (distances along the Y axis).

Weighted least squares: Row weighting

$$\min_{\{a_j\}} \sum_{i=0}^{N} (\mathbf{y}_i - \sum_{j=0}^{n} a_j \mathbf{x}_i^j)^2$$

$$\equiv$$

$$\min_{\{\mathbf{a}\}} \|\mathbf{y} - \mathbf{X}\mathbf{a}\|_2^2$$

Different instances of y get different weights (higher the weight the more the importance to that instance!)

$$\min_{\{a_j\}} \sum_{i=1}^{N} w_i (\mathbf{y}_i - \sum_{j=0}^{n} a_j \mathbf{x}_i^j)^2$$

$$\equiv$$

$$\min_{\{\mathbf{a}\}} \|\mathbf{W}(\mathbf{y} - \mathbf{X}\mathbf{a})\|_2^2$$

$$\mathbf{W} = diag(w_1, w_2, \ldots, w_N)$$

$$\mathbf{W}\mathbf{y} = \mathbf{W}\mathbf{X}\mathbf{a}$$

$$\mathbf{a} = \left((\mathbf{W}\mathbf{X})^T \mathbf{W}\mathbf{X}\right)^{-1} (\mathbf{W}\mathbf{X})^T \mathbf{W}\mathbf{y}$$

Regularization

- In some cases, the system of equations may be ill-conditioned.
- In such cases, there may be more than one solution.
- It is common practice to choose a “simple” or “smooth” solution.
- This practice of choosing a “smooth” solution is called as regularization.
- Regularization is a common method used in several problems in machine learning, computer vision and statistics.

Regularization: Ridge regression

$$\min_{\{\mathbf{a}\}} \|\mathbf{y} - \mathbf{X}\mathbf{a}\|_2^2 + \lambda \|\mathbf{a}\|_2^2$$

Penalize (or discourage) solutions in which the magnitude of vector **a** is too high. The parameter λ is called the regularization parameter. Larger the value of λ, the greater the encouragement to find a smooth solution.

$$\left(\mathbf{X}^T\mathbf{X} + \lambda \mathbf{I}\right)\mathbf{a} = \mathbf{X}^T\mathbf{y}$$

Taking derivative w.r.t. **a**, we get this regularized solution for **a**.

$$\mathbf{X} = \mathbf{U}\mathbf{S}\mathbf{V}^T$$

$$\left(\mathbf{V}\mathbf{S}^2\mathbf{V}^T + \lambda \mathbf{I}\right)\mathbf{a} = \mathbf{V}\mathbf{S}\,\mathbf{U}^T\mathbf{y}$$

$$\mathbf{V}\left(\mathbf{S}^2 + \lambda \mathbf{I}\right)\mathbf{V}^T\mathbf{a} = \mathbf{V}\mathbf{S}\,\mathbf{U}^T\mathbf{y}$$

$$\left(\mathbf{S}^2 + \lambda \mathbf{I}\right)\mathbf{V}^T\mathbf{a} = \mathbf{S}\,\mathbf{U}^T\mathbf{y}$$

$$\mathbf{V}^T\mathbf{a} = \sum_{i=1}^{r} \frac{S_{ii}\mathbf{U}_i^T\mathbf{y}}{S_{ii}^2 + \lambda}$$

Regularization: Tikhonov regularization

$$\min_{\{\mathbf{a}\}} \|\mathbf{y} - \mathbf{X}\mathbf{a}\|_2^2 + \lambda \|\mathbf{B}\mathbf{a}\|_2^2 \quad \Rightarrow \quad \left(\mathbf{X}^T\mathbf{X} + \lambda \mathbf{B}^T\mathbf{B}\right)\mathbf{a} = \mathbf{X}^T\mathbf{y}$$

Penalize (or discourage) solutions in which the magnitude of vector **Ba** is too high. The parameter λ is called the regularization parameter. Larger the value of λ, the greater the encouragement to find a smooth solution. The matrix **B** can be chosen in different ways – very commonly, one penalizes large values of the gradient of vector **a**, given as follows:

$$\nabla\mathbf{a} = \begin{pmatrix} a_1 & a_2 - a_1 & a_3 - a_2 & . & . & a_{n-1} - a_{n-2} & a_n - a_{n-1} \end{pmatrix}$$

In such a case, **B** is given as follows:

$$\mathbf{B} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 1 & 0 & . & . & . & 0 \\ . & -1 & 1 & . & . & . & 0 \\ . & . & . & . & . & . & . \\ . & . & . & . & . & . & . \\ 0 & . & . & 0 & -1 & 1 & 0 \\ 0 & . & . & . & 0 & -1 & 1 \end{pmatrix}$$

Total Least Squares

- There are situations when there are errors in not just the y_i values, but in the x_i values as well.
- In such a situations, least squares is not applicable – instead a method called **total least squares** is used.
- Total least squares solutions heavily use the SVD – so this is one more application of the SVD for you!

Total Least Squares

- Consider the equation $\mathbf{y} \approx \mathbf{X}\mathbf{a}$ for polynomial regression.
- We want to find **a**, in the case where both **y** and **X** have errors (in the earlier case, only **y** had errors).
- So we seek to find a new version of **y** (denoted $\mathbf{y}^*$) close to **y**, and a new version of **X** (denoted $\mathbf{X}^*$) close to **X**, such that $\mathbf{y}^* = \mathbf{X}^*\mathbf{a}$.
- Thus we want (see next slide):

Find $(\mathbf{y}^*, \mathbf{X}^*)$ to minimize

$\|\mathbf{y} - \mathbf{y}^*\|_2^2 + \|\mathbf{X} - \mathbf{X}^*\|_F^2$ such that $\mathbf{y}^* = \mathbf{X}^* \mathbf{a}$.

Define $\mathbf{Z}^* = (\mathbf{X}^* | \mathbf{y}^*)$, $\mathbf{Z} = (\mathbf{X} | \mathbf{y})$

Minimize $\|\mathbf{Z} - \mathbf{Z}^*\|_F^2$ such that $\mathbf{Z}^* \begin{pmatrix} \mathbf{a} \\ -1 \end{pmatrix} = \mathbf{0}$.

Size of $\mathbf{X}^*$ is $N \times (n+1)$. Size of $\mathbf{Z}^*$ is $N \times (n+2)$.

As the vector $\begin{pmatrix} \mathbf{a} \\ -1 \end{pmatrix}$ is not all zeros,

the column rank of $\mathbf{Z}^*$ is less than $n+2$.

To minimize $\|\mathbf{Z} - \mathbf{Z}^*\|_F^2$, $\mathbf{Z}^*$ is the best rank $n+1$ approximation to $\mathbf{Z}$, given by

$\mathbf{Z}^* = \sum_{i=1}^{n+1} \sigma_i \mathbf{u}_i \mathbf{v}_i^t$ from SVD of $\mathbf{Z}$ (Eckhart - Young Theorem).

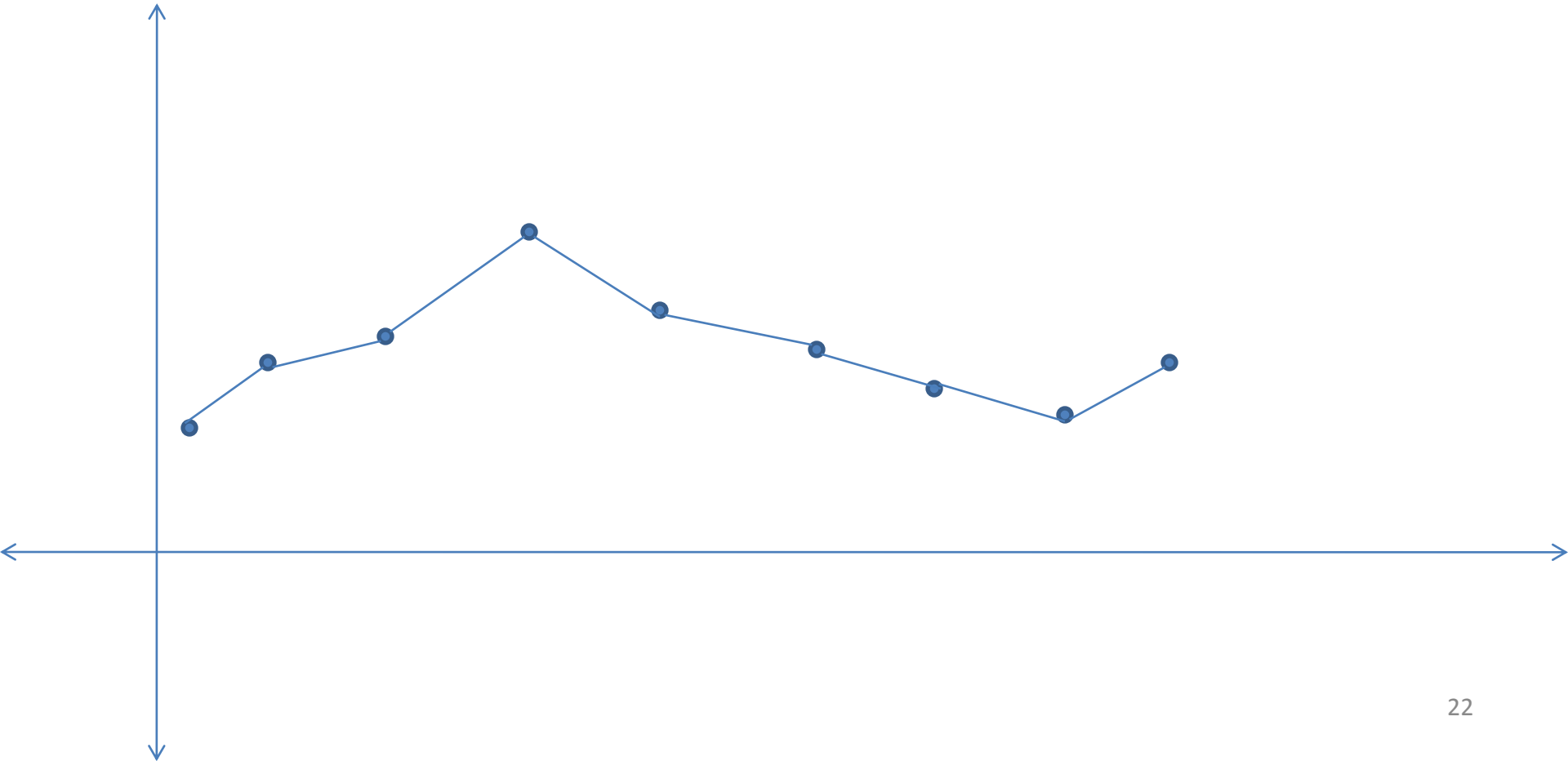
Also $\begin{pmatrix} \mathbf{a} \\ -1 \end{pmatrix} \propto \mathbf{v}_{n+2}$, i.e. $\mathbf{a} = -\dfrac{\mathbf{v}_{1:n+1,n+2}}{\mathbf{v}_{n+2,n+2}}$.

Interpolation

- In the case of interpolation, we want the function being fit to pass through the given data-points exactly.
- The type of interpolation is determined by whether it is a function of one or more variables, as well as the type of function.
- Let us consider lower-order polynomial interpolation in 1D.

Interpolation

- Given a set of *N* points lying on a curve, we can perform linear interpolation by simply joining consecutive points with a line.



Interpolation

- The problem with this approach is that the slopes of adjacent lines change abruptly at each data-point – called as C_1 *discontinuity*.
- This can be resolved by stitching piecewise quadratic polynomials between consecutive point pairs and requiring that

✓ Adjacent polynomials actually pass through the point *(interpolatory condition)*

✓ Adjacent polynomials have the same slope at that point *(C_1 continuity condition)*

Interpolation

- We could go one step further and require that
- ✓ Adjacent polynomials have the same second derivative at that point *(C_2 continuity condition)*
- This requires that the polynomials be piecewise *cubic* (degree 3) at least.
- This type of interpolation is called **cubic-spline interpolation** and is very popular in graphics and image processing.
- The data-points (where the continuity conditions) are imposed are called as *knots* or *control points*.
- A cubic spline is a piecewise polynomial that is twice continuously differentiable (including at the knots).

Cubic spline interpolation

- A curve is parameterized as $y = f(t)$ where 't' is the independent variable.
- Consider N points (t_i, y_i) where i ranges from 1 to N.
- We will have N-1 cubic polynomials, each of the form $y_i(t) = a_i + b_i\, t + c_i\, t^2 + d_i\, t^3$ where i ranges from 1 to N-1.
- Thus the overall function $y(t)$ is equal to $y_i(t)$ where t lies in between t_i and t_{i+1}.

Cubic spline interpolation

- The total number of unknowns here is 4(*N*-1) (why?).
- To determine them, we need to specify 4(*N*-1) equations.
- As the curve must pass through the N points, we get *N* equations as follows:

$$y_1 = a_1 + b_1 t_1 + c_1 t_1^2 + d_1 t_1^3$$

$$y_2 = a_1 + b_1 t_2 + c_1 t_2^2 + d_1 t_2^3$$

.

$$y_i = a_{i-1} + b_{i-1} t_i + c_{i-1} t_i^2 + d_{i-1} t_i^3$$

.

$$y_N = a_{N-1} + b_{N-1} t_N + c_{N-1} t_N^2 + d_{N-1} t_N^3$$

Cubic spline interpolation

- The interpolatory conditions at the *N*-2 interior points yield *N*-2 equations of the form:

$$y_2 = a_1 + b_1 t_2 + c_1 t_2^2 + d_1 t_2^3 = a_2 + b_2 t_2 + c_2 t_2^2 + d_2 t_2^3$$

.

$$y_i = a_{i-1} + b_{i-1} t_i + c_{i-1} t_i^2 + d_{i-1} t_i^3 = a_i + b_i t_i + c_i t_i^2 + d_i t_i^3$$

.

$$y_{N-1} = a_{N-2} + b_{N-2} t_{N-1} + c_{N-2} t_{N-1}^2 + d_{N-2} t_{N-1}^3 = a_{N-1} + b_{N-1} t_{N-1} + c_{N-1} t_{N-1}^2 + d_{N-1} t_{N-1}^3$$

Cubic spline interpolation

- The C_1 continuity conditions at the *N*-2 interior points yield *N*-2 equations of the form:

$$b_1 + 2c_1 t_2 + 3d_1 t_2^2 = b_2 + 2c_2 t_2 + 3d_2 t_2^2$$

.

$$b_{i-1} + 2c_{i-1} t_i + 3d_{i-1} t_i^2 = b_i + 2c_i t_i + 3d_i t_i^2$$

.

$$b_{N-2} + 2c_{N-2} t_{N-1} + 3d_{N-2} t_{N-1}^2 = b_{N-1} + 2c_{N-1} t_{N-1} + 3d_{N-1} t_{N-1}^2$$

Cubic spline interpolation

- The C_2 continuity conditions at the N-2 interior points yield N-2 equations of the form:

$$2c_1 + 6d_1 t_2 = 2c_2 + 6d_2 t_2$$

.

$$2c_{i-1} + 6d_{i-1} t_i = 2c_i + 6d_i t_i$$

.

$$2c_{N-2} + 6d_{N-2} t_{N-1} = 2c_{N-1} + 6d_{N-1} t_{N-1}$$

- The total number of equations is N+3(N-2) = 4N-6 whereas there are 4N-4 unknowns.

Cubic spline interpolation

- We need two more equations.
- One possible set of conditions is to impose that the second derivative of the curve at the two end-points be zero. This gives two more equations of the form:

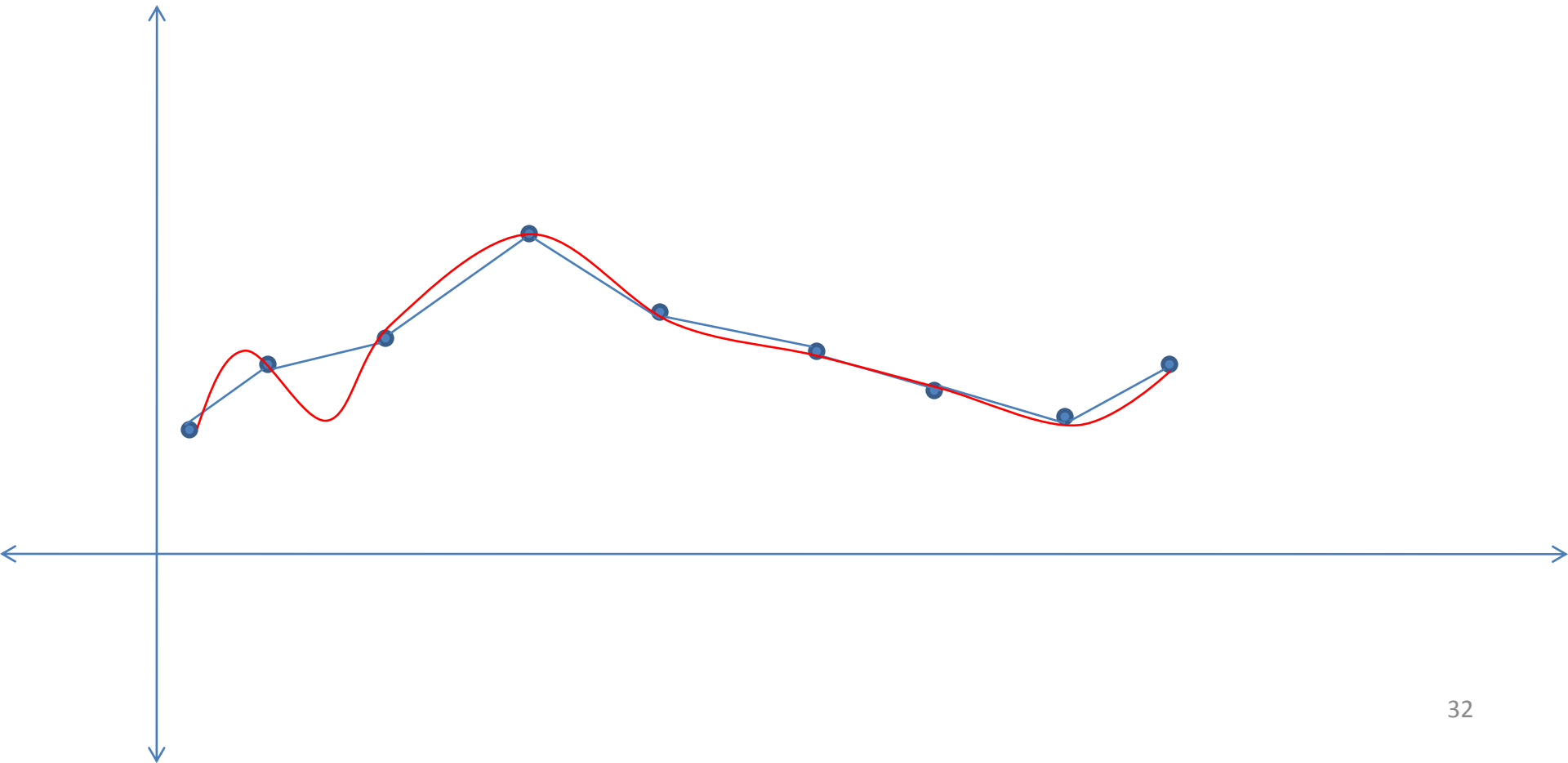
$$2c_1 + 6d_1 t_1 = 0$$
$$2c_{N-1} + 6d_{N-1} t_N = 0$$

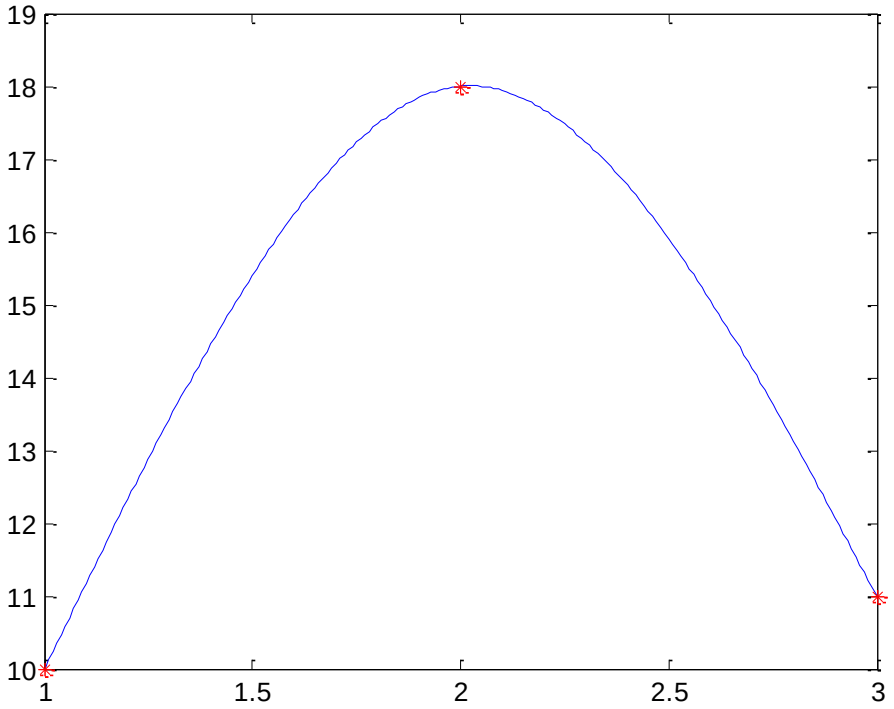
- A cubic spline with these two conditions is called a **natural cubic spline** and these conditions are called **natural boundary conditions**.
- We now have 4(*N*-1) linear equations and 4(*N*-1) unknowns.
- Solving the linear system gives us the cubic spline!

Cubic spline interpolation

- For the case of $N = 3$ points, the system of equations looks as follows:

$$\begin{pmatrix} 1 & t_1 & t_1^2 & t_1^3 & 0 & 0 & 0 & 0 \\ 1 & t_2 & t_2^2 & t_2^3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & t_2 & t_2^2 & t_2^3 \\ 0 & 0 & 0 & 0 & 1 & t_3 & t_3^2 & t_3^3 \\ 0 & 1 & 2t_2 & 3t_2^2 & 0 & -1 & -2t_2 & -3t_2^2 \\ 0 & 0 & 2 & 6t_2 & 0 & 0 & -2 & -6t_2 \\ 0 & 0 & 2 & 6t_1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 2 & 6t_3 \end{pmatrix} \begin{pmatrix} a_1 \\ b_1 \\ c_1 \\ d_1 \\ a_2 \\ b_2 \\ c_2 \\ d_2 \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \\ y_2 \\ y_3 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$





A cubic-spline fit to three points (1,10), (2,18) and (3,11) – marked with red asterisks.

References

- Scientific Computing, Michael Heath
- http://en.wikipedia.org/wiki/Total_least_squares