Linear modelling - vector/matrix form

Rohit Budhiraja

Machine Learning for Wireless Communications (EE798L)

Jan 12, 2024

Recap of last lecture and today's agenda

- Recap of last class
  - Data modelling - idea and definition of a good model (loss function), and then linear data modeling
  - Derived parameters w_0 and w_1 to minimize the squared loss for a linear model $t = w_0 + w_1 x$
    - Process is also called least squares
- Today's agenda
  - Generalize the derivation by writing the model in vector/matrix notation
  - Obtain non-linear response from a linear model

Data modeling in vector/matrix notation[1]

- Problems where each data point (output) is described by a set of several input/attributes
  - For example, we might decide that using only Olympic year is unsuitable to model Olympic sprint data
  - Model that uses Olympic year and each athlete's personal best might be more accurate
- If we use $s_1, s_2, \ldots, s_8$ to denote personal best times for athletes running in lanes 1 to 8
  - A possible linear model might consist of

$$t = f(x, s_1, \cdots, s_8; w_0, \cdots, w_9) = w_0 + w_1 x + w_2 s_1 + w_3 s_2 + w_4 s_3 + w_5 s_4 + w_6 s_5 + w_7 s_6 + w_8 s_7 + w_9 s_8$$

- We could go through previous analysis to find $\widehat{w_0}, \ldots, \widehat{w_9}$
- After taking partial derivatives of the loss function, we would be left with 10 equations
  - Time consuming exercise and would rapidly become infeasible with increase in number of variables
- Motivation to derive parameters $\widehat{w_i}$ neater using vector/matrix notation

[1] Chapter-1 of FCML

Least square in vector/matrix notation (1)

- We denote

$$\mathbf{w} = \begin{bmatrix} w_0 \\ w_1 \end{bmatrix}, \mathbf{x}_n = \begin{bmatrix} 1 \\ x_n \end{bmatrix}$$

- Earlier model can be represented as

$$f(x_n; w_0, w_1) = w_0 + w_1 x_n = \mathbf{w}^T \mathbf{x}_n$$

- We want to re-express loss function $\mathcal{L} = \frac{1}{N} \sum_{n=1}^{N} (t_n - \mathbf{w}^T \mathbf{x}_n)^2$ in vector matrix notation
- We define

$$\mathbf{t} = \begin{bmatrix} t_1 \\ t_2 \\ \vdots \\ t_N \end{bmatrix}, \mathbf{X} = \begin{bmatrix} \mathbf{x}_1^T \\ \mathbf{x}_2^T \\ \vdots \\ \mathbf{x}_N^T \end{bmatrix} = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_N \end{bmatrix}$$

Least square in vector/matrix notation (2)

- We next have

$$\mathbf{X}\mathbf{w} = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_N \end{bmatrix} \times \begin{bmatrix} w_0 \\ w_1 \end{bmatrix} = \begin{bmatrix} w_0 + w_1 x_1 \\ w_0 + w_1 x_2 \\ \vdots \\ w_0 + w_1 x_N \end{bmatrix} \Rightarrow \mathbf{X}\mathbf{w} - \mathbf{t} = \begin{bmatrix} w_0 + w_1 x_1 - t_1 \\ w_0 + w_1 x_2 - t_2 \\ \vdots \\ w_0 + w_1 x_N - t_N \end{bmatrix}$$

- Therefore

$$\begin{aligned} (\mathbf{X}\mathbf{w} - \mathbf{t})^T (\mathbf{X}\mathbf{w} - \mathbf{t}) &= (w_0 + w_1 x_1 - t_1)^2 + (w_0 + w_1 x_2 - t_2)^2 + \cdots + (w_0 + w_1 x_N - t_N)^2 \\ &= \sum_{n=1}^{N} (w_0 + w_1 x_n - t_n)^2 = \sum_{n=1}^{N} (t_n - \mathbf{w}^T \mathbf{x}_n)^2 \end{aligned}$$

- Our loss function $\mathcal{L} = \frac{1}{N} \sum_{n=1}^{N} (t_n - \mathbf{w}^T \mathbf{x}_n)^2$ can now be written compactly as

$$\begin{aligned} \mathcal{L} &= \frac{1}{N} (\mathbf{X}\mathbf{w} - \mathbf{t})^T (\mathbf{X}\mathbf{w} - \mathbf{t}) = \frac{1}{N} \left((\mathbf{X}\mathbf{w})^T - \mathbf{t}^T \right) (\mathbf{X}\mathbf{w} - \mathbf{t}) \\ &= \frac{1}{N} \left((\mathbf{X}\mathbf{w})^T \mathbf{X}\mathbf{w} - \mathbf{t}^T \mathbf{X}\mathbf{w} - (\mathbf{X}\mathbf{w})^T \mathbf{t} + \mathbf{t}^T \mathbf{t} \right) = \frac{1}{N} \left(\mathbf{w}^T \mathbf{X}^T \mathbf{X}\mathbf{w} - 2\mathbf{w}^T \mathbf{X}^T \mathbf{t} + \mathbf{t}^T \mathbf{t} \right) \end{aligned}$$

Least square in vector/matrix notation (3)

- Loss function in compact vector/matrix form $\mathcal{L} = \frac{1}{N}\left(\mathbf{w}^T\mathbf{X}^T\mathbf{X}\mathbf{w} - 2\mathbf{w}^T\mathbf{X}^T\mathbf{t} + \mathbf{t}^T\mathbf{t}\right)$
- Calculate value of $\mathbf{w}$ for minimum of $\mathcal{L}$ – partial derivate of $\mathcal{L}$ with respect to $\mathbf{w}$
- Take partial derivatives of $\mathcal{L}$ wrt each element of $\mathbf{w}$ and then stack as $\frac{\partial \mathcal{L}}{\partial \mathbf{w}} = \begin{bmatrix} \frac{\partial \mathcal{L}}{\partial w_0} \\ \frac{\partial \mathcal{L}}{\partial w_1} \end{bmatrix}$
- Some useful identities when differentiating with respect to a vector

$f(\mathbf{w})$	$\frac{\partial f}{\partial \mathbf{w}}$
$\mathbf{w}^\mathsf{T}\mathbf{x}$	$\mathbf{x}$
$\mathbf{x}^\mathsf{T}\mathbf{w}$	$\mathbf{x}$
$\mathbf{w}^\mathsf{T}\mathbf{w}$	$2\mathbf{w}$
$\mathbf{w}^\mathsf{T}\mathbf{C}\mathbf{w}$	$2\mathbf{C}\mathbf{w}$

- Differentiating $\mathcal{L}$ wrt $\mathbf{w}$

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial \mathbf{w}} &= \frac{2}{N}\mathbf{X}^T\mathbf{X}\mathbf{w} - \frac{2}{N}\mathbf{X}^T\mathbf{t} = 0 \\ \mathbf{X}^T\mathbf{X}\mathbf{w} &= \mathbf{X}^T\mathbf{t} \\ \Rightarrow \widehat{\mathbf{w}} &= (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{t} \end{aligned}$$

- Given a new vector of attributes $\mathbf{x}_{new}$, prediction from the model $t_{new} = \widehat{\mathbf{w}}^T\mathbf{x}_{new}$

Non linear response from a linear model (1)

- Linear model we have seen thus far is linear in terms of both parameters ($\mathbf{w}$) and data (x)

$$f(x; \mathbf{w}) = w_0 + w_1 x$$

- We want to design a model which is non-linear function of data (x)

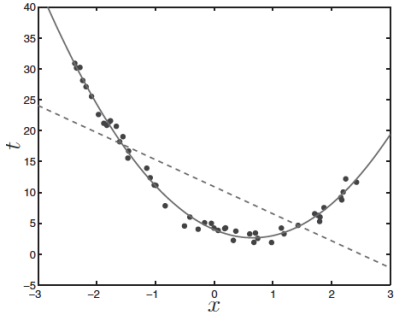
$$f(x; \mathbf{w}) = w_0 + w_1 x + w_2 x^2 = \mathbf{w}^T \mathbf{x}$$

- As the model is still linear in the parameters, we can estimate $\widehat{\mathbf{w}} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{t}$ by defining

$$\mathbf{x}_n = \begin{bmatrix} 1 \\ x_n \\ x_n^2 \end{bmatrix}, \mathbf{X} = \begin{bmatrix} \mathbf{x}_1^T \\ \mathbf{x}_2^T \\ \vdots \\ \mathbf{x}_N^T \end{bmatrix} = \begin{bmatrix} 1 & x_1 & x_1^2 \\ 1 & x_2 & x_2^2 \\ \vdots & \vdots & \vdots \\ 1 & x_N & x_N^2 \end{bmatrix}, \mathbf{w} = \begin{bmatrix} w_0 \\ w_1 \\ w_2 \end{bmatrix}$$

- Augmented our data matrix $\mathbf{X}$ with additional column x_n^2 and parameter vector $\mathbf{w}$ with w_2
- Function/model we are fitting is quadratic in data but linear in parameters

Non linear response from a linear model (1)



- To study non-linear modelling, we consider a synthetic dataset in the above figure
- We use above method to fit a function quadratic in the data to a suitable dataset
- Also shown is the linear (in the data) model (dashed line, $t = w_0 + w_1 x$)
- Clear from the quality of data fit that the quadratic model is a more appropriate

Non linear response from a linear model (2)

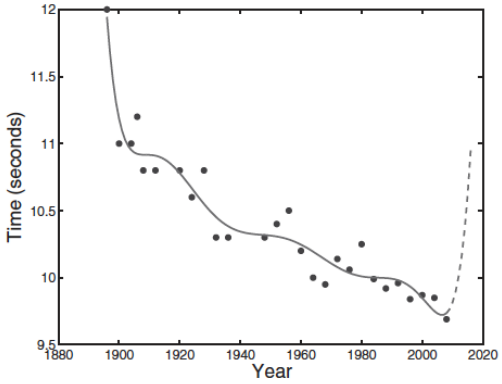
- More generally, we can add as many powers of x as we like to get a polynomial function of any order
- For Kth order polynomial, our augmented data matrix will be

$$\mathbf{X} = \begin{bmatrix} x_1^0 & x_1^1 & x_1^2 & \cdots & x_1^K \\ x_2^0 & x_2^1 & x_2^2 & \cdots & x_2^K \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ x_N^0 & x_N^1 & x_N^2 & \cdots & x_N^K \end{bmatrix}$$

- Function can be written in the more general form

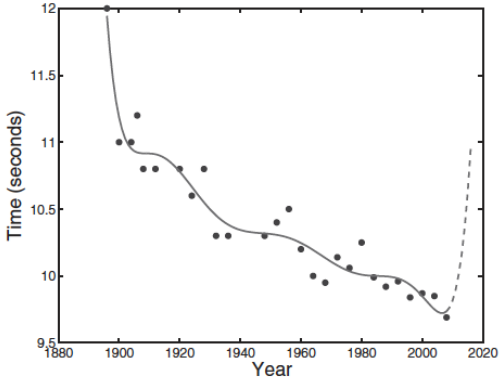
$$f(x; \mathbf{w}) = \sum_{k=0}^{K} w_k x^k$$

Non linear model for Olympic data (1)



- Figure shows the effect of fitting an eighth-order polynomial function to earlier 100m Olympic data
- Does the eighth-order model look better than the first-order model?
  - To answer this question, we need to be more precise about what we mean by better
- For models built to make predictions, best model is arguably the one that produces best predictions
- We shall return to the issue of model selection in more detail

Non linear model for Olympic data (2)



- For now, two things are apparent
- Firstly, the eighth-order polynomial gets closer to observed data than first-order polynomial
- Reflected in a lower value of the loss function: $\mathcal{L}^8 = 0.459, \mathcal{L}^1 = 1.358$
  - Increasing the polynomial order will always result in a model that gets closer to the training data
- Secondly, predictions (shown by the dashed line) do not look sensible

Non linear response from a linear model (4)

- We are not restricted to polynomial functions – free to define any set of K functions of x i.e., $h_k(x)$

$$\mathbf{X} = \begin{bmatrix} h_1(x_1) & h_2(x_1) & \cdots & h_K(x_1) \\ h_1(x_2) & h_2(x_2) & \cdots & h_K(x_2) \\ \vdots & \vdots & \cdots & \vdots \\ h_1(x_N) & h_2(x_N) & \cdots & h_K(x_N) \end{bmatrix}$$

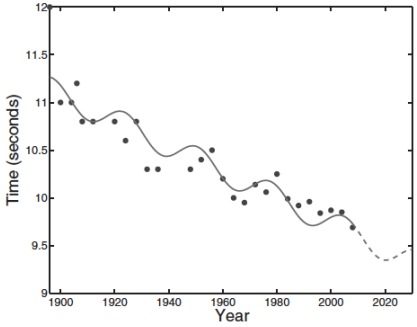
- A suitable set of functions might be

$$\begin{aligned} h_1(x) &= 1,\ h_2(x) = x,\ h_3(x) = \sin\left(\frac{x-a}{b}\right) \\ f(x;\mathbf{w}) &= w_0 + w_1 x + w_2 \sin\left(\frac{x-a}{b}\right) \end{aligned}$$

- Model has five parameters $w_0, w_1; w_2, a, b$. Only first three can be inferred using linear modeling
- Last two parameter a and b, appear inside a non-linear (sin) function
  - Taking partial derivatives wrt to them and equating to zero will not result in a set of equations that can be solved analytically

Non linear response from a linear model (5)

- Many ways to overcome this problem
  - simplest being a search over all values of a and b in some sensible range
- We ignore this problem for now and assume that we know of suitable values.
- Assuming $a = 2660; b = 4.3$, calculate the remaining parameters w_0, w_1, w_2, using earlier procedure



- In this case $\mathcal{L} = 1.1037$
- Model is fitting the observed data better than the first order polynomial
  - But not as well as the eighth order polynomial.