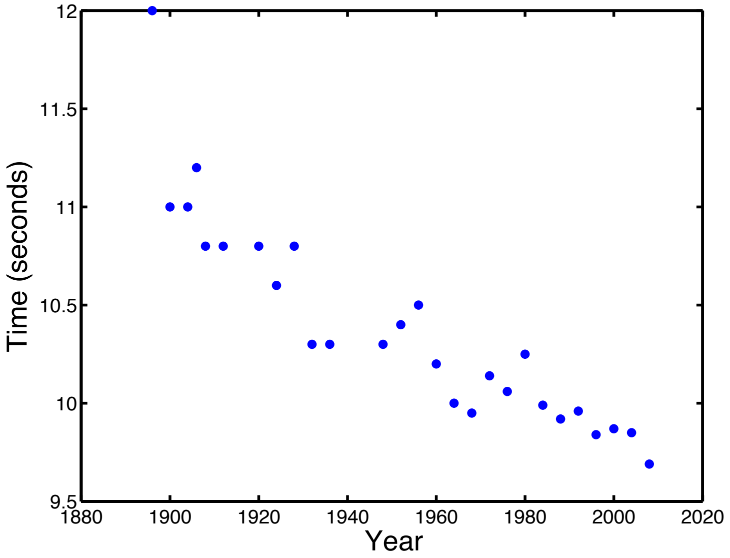
Beyond Linear Regression

CS4061 / CS5014 Machine Learning

Paul Henderson
University of Glasgow

paul.henderson@glasgow.ac.uk


Time (seconds)
12
11.5
11
10.5
10
9.5
1880
1900
1920
1940
1960
1980
2000
2020
Year

What we did...

- wrote x for year and t for winning time
- training data: pairs $(x_1, t_1), (x_2, t_2), \ldots, (x_N, t_N)$

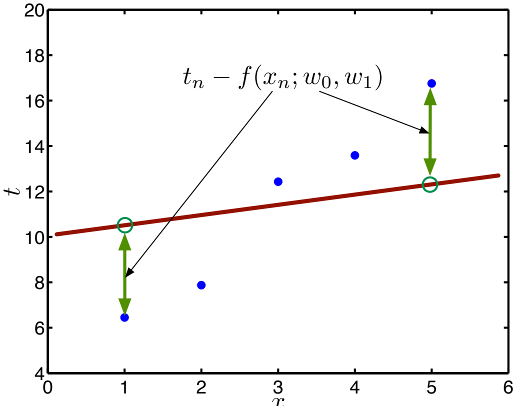
What we did...

- wrote x for year and t for winning time
- training data: pairs $(x_1,\ t_1),\ (x_2,\ t_2),\ \ldots,\ (x_N,\ t_N)$
- assumed some relationship: $t = f(x)$
- decided it's linear: $t = w_0 + w_1 x$

- decided it's linear: $t = w_0 + w_1 x$
- chose w_0 and w_1 so the line is as near the points $(x_n,\ t_n)$ as possible

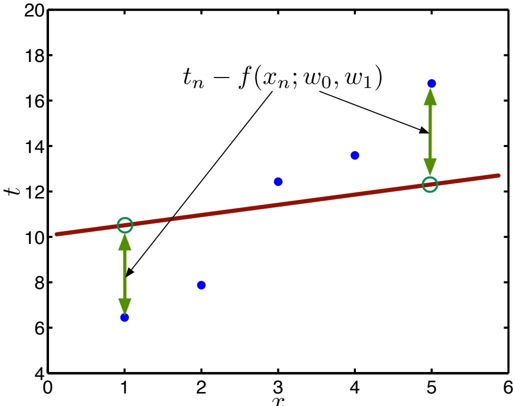
- decided it's linear: $t = w_0 + w_1 x$
- chose w_0 and w_1 so the line is as near the points $(x_n,\ t_n)$ as possible
- characterised that mathematically:

$$\mathcal{L} = \frac{1}{N}\sum_{n=1}^{N}(t_n - f(x_n; w_0, w_1))^2$$



- decided it's linear: $t = w_0 + w_1 x$
- chose w_0 and w_1 so the line is as near the points $(x_n,\ t_n)$ as possible
- characterised that mathematically:

$$\mathcal{L} = \frac{1}{N}\sum_{n=1}^{N}(t_n - f(x_n; w_0, w_1))^2 = \frac{1}{N}\sum_{n=1}^{N}(t_n - w_0 - w_1 x_n)^2$$



- mean square loss:

$$\mathcal{L} = \frac{1}{N} \sum_{n=1}^{N} (t_n - w_0 - w_1 x_n)^2$$

- minimise this wrt w_0 and w_1

- mean square loss:

$$\mathcal{L} = \frac{1}{N} \sum_{n=1}^{N} (t_n - w_0 - w_1 x_n)^2$$

- minimise this wrt w_0 and w_1

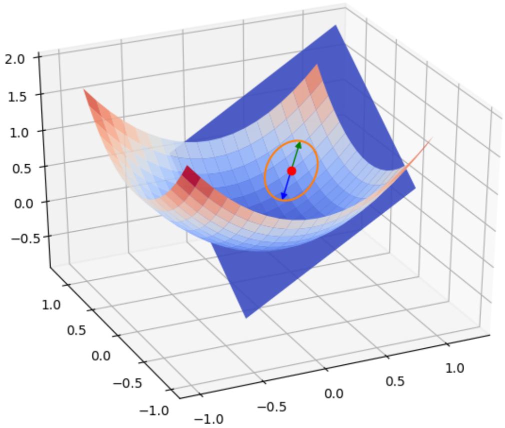


Figure: Pierre Vigier

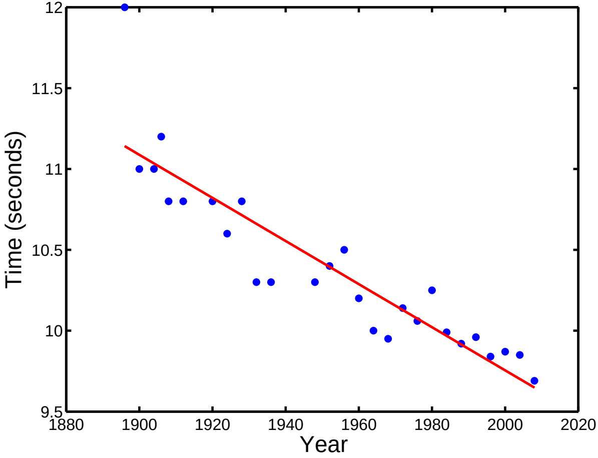
- mean square loss:

$$\mathcal{L} = \frac{1}{N}\sum_{n=1}^{N}(t_n - w_0 - w_1 x_n)^2$$

- minimise this wrt w_0 and w_1

animation from https://pvigier.github.io/media/img/part1/gradient_descent.gif

Figure: Pierre Vigier



$$t = f(x) = 36.416 - 0.0133x$$

$$t_{2012} = f(2012) = 36.416 - 0.0133 \times 2012$$

$$t_{2012} = 9.59\ s$$

Assumptions

1. There exists a relationship between Olympic year and winning time
2. This relationship is linear (i.e. a straight line)
3. This relationship will continue into the future

Assumptions

1. There exists a relationship between Olympic year and winning time
2. This relationship is linear (i.e. a straight line)
3. This relationship will continue into the future

The model is ‘wrong’ but it might still be useful! How useful depends on the questions we wish to answer

What's next?

- Linear model in vector form
- Not-so-linear regression
- Generalisation, overfitting, cross-validation

Encapsulate parameters in a vector

- Simple model: $t = w_0 + w_1 x$

Encapsulate parameters in a vector

- Simple model: $t = w_0 + w_1 x$
- We can combine the parameters into a vector:

$$\mathbf{w} = \left[\begin{array}{c} w_0 \\ w_1 \end{array} \right]$$

Encapsulate parameters in a vector

- Simple model: $t = w_0 + w_1 x$
- We can combine the parameters into a vector:

$$\mathbf{w} = \left[\begin{array}{c} w_0 \\ w_1 \end{array} \right]$$

- We'll use bold, lowercase letters for vectors
- A list of values – similar to arrays when programming.

Vector model

- Our model:

$$t = w_0 + w_1 x$$

Vector model

- Our model:
$$t = w_0 + w_1 x = \sum_{k=0}^{K} w_k x^k$$

Vector model

- Our model:

$$t = w_0 + w_1 x = \sum_{k=0}^{K} w_k x^k = \mathbf{w}^\mathsf{T}\mathbf{x}$$

- where...

$$\mathbf{w} = \left[\begin{array}{c} w_0 \\ w_1 \end{array}\right]$$

$$\mathbf{x} = \left[\begin{array}{c} x^0 \\ x^1 \end{array}\right] = \left[\begin{array}{c} 1 \\ x \end{array}\right]$$

- Vector model:

$$t = \mathbf{w}^\mathsf{T}\mathbf{x}$$

- Loss for n^{th} observation:

$$\mathcal{L}_n = (t_n - \mathbf{w}^\mathsf{T}\mathbf{x}_n)^2$$

- Vector model:

$$t = \mathbf{w}^\mathsf{T}\mathbf{x}$$

- Loss for n^{th} observation:

$$\mathcal{L}_n = (t_n - \mathbf{w}^\mathsf{T}\mathbf{x}_n)^2$$

- Mean loss:

$$\mathcal{L} = \frac{1}{N}\sum_{n=1}^{N}(t_n - \mathbf{w}^\mathsf{T}\mathbf{x}_n)^2$$

- Can we vectorize this further?

- Vector model:

$$t = \mathbf{w}^\mathsf{T}\mathbf{x}$$

- Loss for n^{th} observation:

$$\mathcal{L}_n = (t_n - \mathbf{w}^\mathsf{T}\mathbf{x}_n)^2$$

- Mean loss:

$$\mathcal{L} = \frac{1}{N}\sum_{n=1}^{N}(t_n - \mathbf{w}^\mathsf{T}\mathbf{x}_n)^2$$

- Can we vectorize this further? Recall:

$$\sum_{d=1}^{D} a_d^2 = \mathbf{a}^\mathsf{T}\mathbf{a}$$

- Vector model:

$$t = \mathbf{w}^\mathsf{T}\mathbf{x}$$

- Loss for n^{th} observation:

$$\mathcal{L}_n = (t_n - \mathbf{w}^\mathsf{T}\mathbf{x}_n)^2$$

- Mean loss:

$$\mathcal{L} = \frac{1}{N}\sum_{n=1}^{N}(t_n - \mathbf{w}^\mathsf{T}\mathbf{x}_n)^2$$

- Can we vectorize this further? Recall:

$$\sum_{d=1}^{D} a_d^2 = \mathbf{a}^\mathsf{T}\mathbf{a}$$

- Decide to write:

$$\mathcal{L} = \frac{1}{N}\sum_{n=1}^{N} q_n^2 = \frac{1}{N}\mathbf{q}^\mathsf{T}\mathbf{q}$$

- Vector model:

$$t = \mathbf{w}^\mathsf{T}\mathbf{x}$$

- Loss for n^{th} observation:

$$\mathcal{L}_n = (t_n - \mathbf{w}^\mathsf{T}\mathbf{x}_n)^2$$

- Mean loss:

$$\mathcal{L} = \frac{1}{N}\sum_{n=1}^{N}(t_n - \mathbf{w}^\mathsf{T}\mathbf{x}_n)^2$$

- Can we vectorize this further? Recall:

$$\sum_{d=1}^{D} a_d^2 = \mathbf{a}^\mathsf{T}\mathbf{a}$$

- Decide to write:

$$\mathcal{L} = \frac{1}{N}\sum_{n=1}^{N} q_n^2 = \frac{1}{N}\mathbf{q}^\mathsf{T}\mathbf{q} \quad \text{with} \;\; q_n = (t_n - \mathbf{w}^\mathsf{T}\mathbf{x}_n)$$

What is q?

$$q_n = (t_n - \mathbf{w}^\mathsf{T}\mathbf{x}_n)$$

...so...

$$\mathbf{q} = \begin{bmatrix} t_1 - \mathbf{w}^\mathsf{T}\mathbf{x}_1 \\ t_2 - \mathbf{w}^\mathsf{T}\mathbf{x}_2 \\ t_3 - \mathbf{w}^\mathsf{T}\mathbf{x}_3 \\ \vdots \\ t_N - \mathbf{w}^\mathsf{T}\mathbf{x}_N \end{bmatrix}$$

What is q?

$$q_n = (t_n - \mathbf{w}^\mathsf{T}\mathbf{x}_n)$$

...so...

$$\mathbf{q} = \begin{bmatrix} t_1 - \mathbf{w}^\mathsf{T}\mathbf{x}_1 \\ t_2 - \mathbf{w}^\mathsf{T}\mathbf{x}_2 \\ t_3 - \mathbf{w}^\mathsf{T}\mathbf{x}_3 \\ \vdots \\ t_N - \mathbf{w}^\mathsf{T}\mathbf{x}_N \end{bmatrix}$$

Reminder: Subtracting vectors

$$\mathbf{a} - \mathbf{b} = \begin{bmatrix} a_1 - b_1 \\ a_2 - b_2 \\ \vdots \\ a_D - b_D \end{bmatrix}$$

What is q?

Define

$$\mathbf{t} = \begin{bmatrix} t_1 \\ t_2 \\ \vdots \\ t_N \end{bmatrix}$$

What is q?

Define

$$\mathbf{t} = \begin{bmatrix} t_1 \\ t_2 \\ \vdots \\ t_N \end{bmatrix}$$

Therefore:

$$\mathbf{q} = \begin{bmatrix} t_1 - \mathbf{w}^\mathsf{T}\mathbf{x}_1 \\ t_2 - \mathbf{w}^\mathsf{T}\mathbf{x}_2 \\ t_3 - \mathbf{w}^\mathsf{T}\mathbf{x}_3 \\ \vdots \\ t_N - \mathbf{w}^\mathsf{T}\mathbf{x}_N \end{bmatrix} = \mathbf{t} - ?$$

Matrices

- Stack all $\mathbf{x}_n^\top$ on top of one another:

$$\begin{matrix} [1, & x_1] \\ [1, & x_2] \\ \vdots & \\ [1, & x_N] \end{matrix}$$

Matrices

- Stack all $\mathbf{x}_n^\top$ on top of one another:

$$\mathbf{X} = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \\ 1 & x_N \end{bmatrix}$$

- This is a matrix
- We'll use bold, uppercase letters for matrices

What is q?

$$\mathbf{q} = \begin{bmatrix} t_1 - \mathbf{w}^\mathsf{T}\mathbf{x}_1 \\ t_2 - \mathbf{w}^\mathsf{T}\mathbf{x}_2 \\ t_3 - \mathbf{w}^\mathsf{T}\mathbf{x}_3 \\ \vdots \\ t_N - \mathbf{w}^\mathsf{T}\mathbf{x}_N \end{bmatrix} = \mathbf{t} - \mathbf{X}\mathbf{w}$$

What is q?

$$\mathbf{q} = \begin{bmatrix} t_1 - \mathbf{w}^\mathsf{T}\mathbf{x}_1 \\ t_2 - \mathbf{w}^\mathsf{T}\mathbf{x}_2 \\ t_3 - \mathbf{w}^\mathsf{T}\mathbf{x}_3 \\ \vdots \\ t_N - \mathbf{w}^\mathsf{T}\mathbf{x}_N \end{bmatrix} = \mathbf{t} - \mathbf{X}\mathbf{w}$$

And the mean loss is:

$$\mathcal{L} = \frac{1}{N}(\mathbf{t} - \mathbf{X}\mathbf{w})^\mathsf{T}(\mathbf{t} - \mathbf{X}\mathbf{w})$$

What is q?

$$\mathbf{q} = \begin{bmatrix} t_1 - \mathbf{w}^\mathsf{T}\mathbf{x}_1 \\ t_2 - \mathbf{w}^\mathsf{T}\mathbf{x}_2 \\ t_3 - \mathbf{w}^\mathsf{T}\mathbf{x}_3 \\ \vdots \\ t_N - \mathbf{w}^\mathsf{T}\mathbf{x}_N \end{bmatrix} = \mathbf{t} - \mathbf{X}\mathbf{w}$$

And the mean loss is:

$$\mathcal{L} = \frac{1}{N}(\mathbf{t} - \mathbf{X}\mathbf{w})^\mathsf{T}(\mathbf{t} - \mathbf{X}\mathbf{w})$$

...still equiv. to

$$\mathcal{L} = \frac{1}{N}\sum_{n=1}^{N}(t_n - w_0 - w_1 x_n)^2$$

Summary

- Put data and parameters into vectors
- Written our model in vector form
- Put all data vectors together into a matrix
- Written loss in vector/matrix form

Summary

- Put data and parameters into vectors
- Written our model in vector form
- Put all data vectors together into a matrix
- Written loss in vector/matrix form

Why?

More attributes (inputs): $t = w_0 + w_1 x + w_2 y$

Summary

- Put data and parameters into vectors
- Written our model in vector form
- Put all data vectors together into a matrix
- Written loss in vector/matrix form

Why?

More attributes (inputs): $t = w_0 + w_1 x + w_2 y$

$$\mathbf{w} = \begin{bmatrix} w_0 \\ w_1 \\ w_2 \end{bmatrix}$$

Summary

- Put data and parameters into vectors
- Written our model in vector form
- Put all data vectors together into a matrix
- Written loss in vector/matrix form

Why?

More attributes (inputs): $t = w_0 + w_1 x + w_2 y$

$$\mathbf{w} = \begin{bmatrix} w_0 \\ w_1 \\ w_2 \end{bmatrix}, \ \mathbf{x}_n = \begin{bmatrix} 1 \\ x_n \\ y_n \end{bmatrix}$$

$$t = \mathbf{w}^\mathsf{T}\mathbf{x},$$

Summary

- Put data and parameters into vectors
- Written our model in vector form
- Put all data vectors together into a matrix
- Written loss in vector/matrix form

Why?

More attributes (inputs): $t = w_0 + w_1 x + w_2 y$

$$\mathbf{w} = \begin{bmatrix} w_0 \\ w_1 \\ w_2 \end{bmatrix}, \ \mathbf{x}_n = \begin{bmatrix} 1 \\ x_n \\ y_n \end{bmatrix}, \ \mathbf{X} = \begin{bmatrix} 1 & x_1 & y_1 \\ 1 & x_2 & y_2 \\ \vdots & \vdots & \vdots \\ 1 & x_N & y_N \end{bmatrix}$$

$$t = \mathbf{w}^\mathsf{T}\mathbf{x},$$

Summary

- Put data and parameters into vectors
- Written our model in vector form
- Put all data vectors together into a matrix
- Written loss in vector/matrix form

Why?

More attributes (inputs): $t = w_0 + w_1 x + w_2 y$

$$\mathbf{w} = \begin{bmatrix} w_0 \\ w_1 \\ w_2 \end{bmatrix}, \ \mathbf{x}_n = \begin{bmatrix} 1 \\ x_n \\ y_n \end{bmatrix}, \ \mathbf{X} = \begin{bmatrix} 1 & x_1 & y_1 \\ 1 & x_2 & y_2 \\ \vdots & \vdots & \vdots \\ 1 & x_N & y_N \end{bmatrix}$$

$$t = \mathbf{w}^\mathsf{T}\mathbf{x}, \quad \mathcal{L} = \frac{1}{N}(\mathbf{t} - \mathbf{X}\mathbf{w})^\mathsf{T}(\mathbf{t} - \mathbf{X}\mathbf{w})$$

Summary

- Put data and parameters into vectors
- Written our model in vector form
- Put all data vectors together into a matrix
- Written loss in vector/matrix form

Why?

More complex model: $t = w_0 + w_1 x + w_2 x^2 + \ldots + w_K x^K$

Summary

- Put data and parameters into vectors
- Written our model in vector form
- Put all data vectors together into a matrix
- Written loss in vector/matrix form

Why?

More complex model: $t = w_0 + w_1 x + w_2 x^2 + \ldots + w_K x^K$

$$\mathbf{w} = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_K \end{bmatrix},$$

Summary

- Put data and parameters into vectors
- Written our model in vector form
- Put all data vectors together into a matrix
- Written loss in vector/matrix form

Why?

More complex model: $t = w_0 + w_1 x + w_2 x^2 + \ldots + w_K x^K$

$$\mathbf{w} = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_K \end{bmatrix}, \mathbf{x}_n = \begin{bmatrix} 1 \\ x_n \\ x_n^2 \\ \vdots \\ x_n^K \end{bmatrix},$$

$$t = \mathbf{w}^\mathsf{T}\mathbf{x},$$

Summary

- Put data and parameters into vectors
- Written our model in vector form
- Put all data vectors together into a matrix
- Written loss in vector/matrix form

Why?

More complex model: $t = w_0 + w_1 x + w_2 x^2 + \ldots + w_K x^K$

$$\mathbf{w} = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_K \end{bmatrix}, \mathbf{x}_n = \begin{bmatrix} 1 \\ x_n \\ x_n^2 \\ \vdots \\ x_n^K \end{bmatrix}, \mathbf{X} = \begin{bmatrix} 1 & x_1^1 & x_1^2 & \ldots & x_1^K \\ 1 & x_2^1 & x_2^2 & \ldots & x_2^K \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_N^1 & x_N^2 & \ldots & x_N^K \end{bmatrix}$$

$$t = \mathbf{w}^{\mathsf{T}} \mathbf{x},$$

Summary

- Put data and parameters into vectors
- Written our model in vector form
- Put all data vectors together into a matrix
- Written loss in vector/matrix form

Why?

More complex model: $t = w_0 + w_1 x + w_2 x^2 + \ldots + w_K x^K$

$$\mathbf{w} = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_K \end{bmatrix}, \mathbf{x}_n = \begin{bmatrix} 1 \\ x_n \\ x_n^2 \\ \vdots \\ x_n^K \end{bmatrix}, \mathbf{X} = \begin{bmatrix} 1 & x_1^1 & x_1^2 & \ldots & x_1^K \\ 1 & x_2^1 & x_2^2 & \ldots & x_2^K \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_N^1 & x_N^2 & \ldots & x_N^K \end{bmatrix}$$

$$t = \mathbf{w}^\mathsf{T}\mathbf{x}, \quad \mathcal{L} = \frac{1}{N}(\mathbf{t} - \mathbf{X}\mathbf{w})^\mathsf{T}(\mathbf{t} - \mathbf{X}\mathbf{w})$$

Different models, same loss

- We have a single loss that corresponds to many different models, with different $\mathbf{w}$ and $\mathbf{X}$

$$\mathcal{L} = \frac{1}{N}(\mathbf{t} - \mathbf{X}\mathbf{w})^{\mathsf{T}}(\mathbf{t} - \mathbf{X}\mathbf{w}).$$

- We can get an expression for the $\mathbf{w}$ that minimises $\mathcal{L}$, that will work for any of these models

Minimising the loss

- Given our vector/matrix loss

$$\mathcal{L} = \frac{1}{N}(\mathbf{t} - \mathbf{X}\mathbf{w})^\mathsf{T}(\mathbf{t} - \mathbf{X}\mathbf{w}),$$

- can take partial derivatives wrt vector $\mathbf{w}$ and set to zero:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{w}} = \nabla_{\mathbf{w}}\mathcal{L} = \mathbf{0}$$

Minimising the loss

Summary

- Now we a have a general expression for best $\mathbf{w}$:

$$\widehat{\mathbf{w}} = (\mathbf{X}^\mathsf{T}\mathbf{X})^{-1}\mathbf{X}^\mathsf{T}\mathbf{t}$$

- Some examples...

Linear model – Olympic data

$$\mathbf{w} = \begin{bmatrix} w_0 \\ w_1 \end{bmatrix}, \; \mathbf{X} = \begin{bmatrix} 1 & 1896 \\ 1 & 1900 \\ \vdots & \\ 1 & 2008 \end{bmatrix}, \; \mathbf{t} = \begin{bmatrix} 12.00 \\ 11.00 \\ \vdots \\ 9.85 \end{bmatrix}$$

Linear model – Olympic data

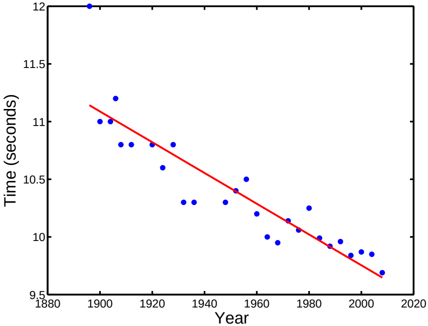
$$\mathbf{w} = \begin{bmatrix} w_0 \\ w_1 \end{bmatrix}, \ \mathbf{X} = \begin{bmatrix} 1 & 1896 \\ 1 & 1900 \\ \vdots & \\ 1 & 2008 \end{bmatrix}, \ \mathbf{t} = \begin{bmatrix} 12.00 \\ 11.00 \\ \vdots \\ 9.85 \end{bmatrix}$$

$$\widehat{\mathbf{w}} = (\mathbf{X}^\mathsf{T}\mathbf{X})^{-1}\mathbf{X}^\mathsf{T}\mathbf{t} = \begin{bmatrix} 36.416 \\ -0.0133 \end{bmatrix}$$

Linear model – Olympic data

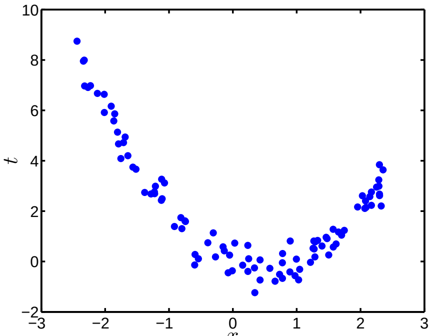
$$\mathbf{w} = \begin{bmatrix} w_0 \\ w_1 \end{bmatrix}, \ \mathbf{X} = \begin{bmatrix} 1 & 1896 \\ 1 & 1900 \\ \vdots & \\ 1 & 2008 \end{bmatrix}, \ \mathbf{t} = \begin{bmatrix} 12.00 \\ 11.00 \\ \vdots \\ 9.85 \end{bmatrix}$$

$$\widehat{\mathbf{w}} = (\mathbf{X}^\mathsf{T}\mathbf{X})^{-1}\mathbf{X}^\mathsf{T}\mathbf{t} = \begin{bmatrix} 36.416 \\ -0.0133 \end{bmatrix}$$



Quadratic model – synthetic data

$$\mathbf{w} = \begin{bmatrix} w_0 \\ w_1 \\ w_2 \end{bmatrix}, \ \mathbf{X} = \begin{bmatrix} 1 & x_1 & x_1^2 \\ \vdots & \vdots & \vdots \\ 1 & x_N & x_N^2 \end{bmatrix}$$

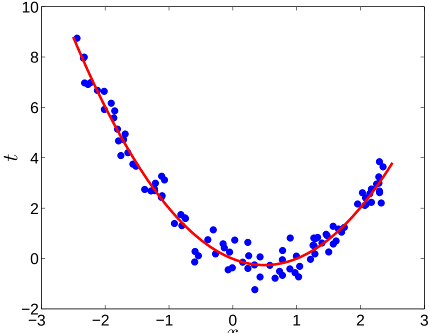


Quadratic model – synthetic data

$$\mathbf{w} = \begin{bmatrix} w_0 \\ w_1 \\ w_2 \end{bmatrix}, \quad \mathbf{X} = \begin{bmatrix} 1 & x_1 & x_1^2 \\ \vdots & \vdots & \vdots \\ 1 & x_N & x_N^2 \end{bmatrix}$$

$$\widehat{\mathbf{w}} = (\mathbf{X}^\mathsf{T}\mathbf{X})^{-1}\mathbf{X}^\mathsf{T}\mathbf{t} = \begin{bmatrix} -0.0149 \\ -0.9987 \\ 1.0098 \end{bmatrix}$$

$$t_n = -0.0149 - 0.9987x_n + 1.0098x_n^2$$



8th order model – Olympic data

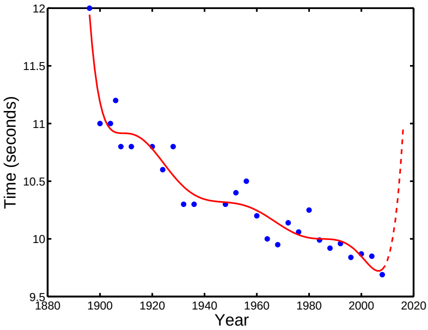
$$t = w_0 + w_1 x + w_2 x^2 + \ldots + w_8 x^8$$

$$\mathbf{w} = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_8 \end{bmatrix}, \; \mathbf{X} = \begin{bmatrix} 1 & x_1 & x_1^2 & \ldots & x_1^8 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_N & x_N^2 & \ldots & x_N^8 \end{bmatrix}$$

8th order model – Olympic data

$$t = w_0 + w_1 x + w_2 x^2 + \ldots + w_8 x^8$$

$$\mathbf{w} = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_8 \end{bmatrix}, \ \mathbf{X} = \begin{bmatrix} 1 & x_1 & x_1^2 & \ldots & x_1^8 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_N & x_N^2 & \ldots & x_N^8 \end{bmatrix}$$



More general models

- So far, we've only considered functions of the form

$$t = w_0 + w_1 x + w_2 x^2 + \ldots + w_K x^K$$

- In fact, each term can be any function of x

$$t = w_0 h_0(x) + w_1 h_1(x) + \ldots + w_K h_K(x)$$

- For example,

$$t = w_0 + w_1 x + w_2 \sin(x) + w_3 x^{-1} + \ldots$$

More general models

- So far, we've only considered functions of the form

$$t = w_0 + w_1 x + w_2 x^2 + \ldots + w_K x^K$$

- In fact, each term can be any function of x

$$t = w_0 h_0(x) + w_1 h_1(x) + \ldots + w_K h_K(x)$$

- For example,

$$t = w_0 + w_1 x + w_2 \sin(x) + w_3 x^{-1} + \ldots$$

- In general:

$$\mathbf{X} = \begin{bmatrix} h_0(x_1) & h_1(x_1) & \ldots & h_K(x_1) \\ h_0(x_2) & h_1(x_2) & \ldots & h_K(x_2) \\ \vdots & \vdots & \ddots & \vdots \\ h_0(x_N) & h_1(x_N) & \ldots & h_K(x_N) \end{bmatrix}$$

Example – Olympic data

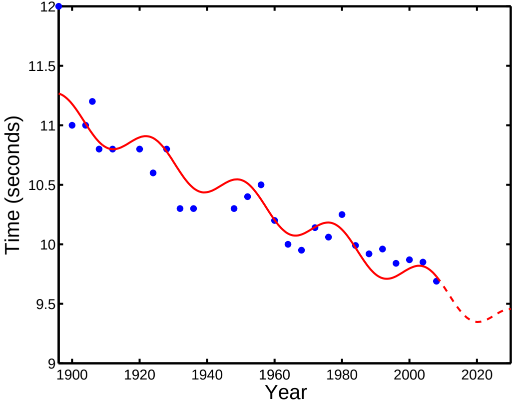
$$t = w_0 + w_1 x + w_2 \sin\left(\frac{x-a}{b}\right)$$

$$\mathbf{w} = \begin{bmatrix} w_0 \\ w_1 \\ w_2 \end{bmatrix}, \; \mathbf{X} = \begin{bmatrix} 1 & x_1 & \sin((x_1 - a)/b) \\ \vdots & \vdots & \vdots \\ 1 & x_N & sin((x_N - a)/b) \end{bmatrix}$$

Example – Olympic data

$$t = w_0 + w_1 x + w_2 \sin\left(\frac{x-a}{b}\right)$$

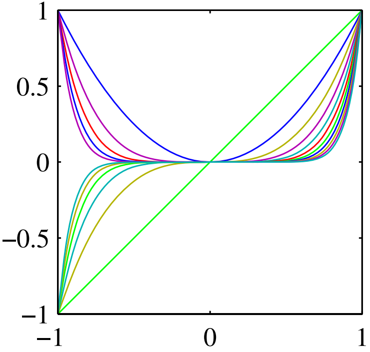
$$\mathbf{w} = \begin{bmatrix} w_0 \\ w_1 \\ w_2 \end{bmatrix}, \; \mathbf{X} = \begin{bmatrix} 1 & x_1 & \sin((x_1 - a)/b) \\ \vdots & \vdots & \vdots \\ 1 & x_N & sin((x_N - a)/b) \end{bmatrix}$$



Common basis functions $h(x)$

Polynomial

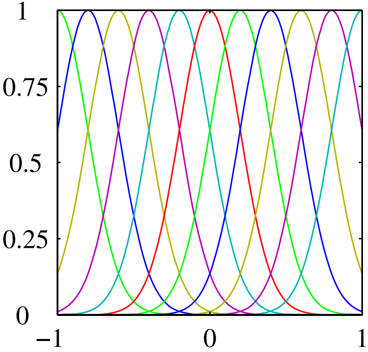
$$h_k(x) = x^k$$



Common basis functions $h(x)$

Radial basis function (RBF)

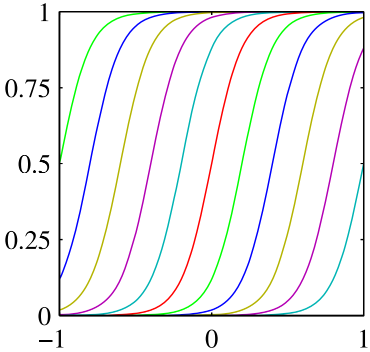
$$h_k(x) = \exp\left(-\frac{(x - \mu_k)^2}{2s^2}\right)$$



Common basis functions $h(x)$

Sigmoid

$$h_k(x) = \sigma\left(\frac{(x-\mu_k)^2}{s}\right)$$
$$\sigma(a) = \frac{1}{1+\exp(-a)}$$



Making predictions

$$\widehat{\mathbf{w}} = (\mathbf{X}^{\mathsf{T}}\mathbf{X})^{-1}\mathbf{X}^{\mathsf{T}}\mathbf{t}$$

Where $\mathbf{X}$ depends on the choice of model:

$$\mathbf{X} = \begin{bmatrix} h_0(x_1) & h_1(x_1) & \dots & h_K(x_1) \\ \vdots & \vdots & \ddots & \vdots \\ h_0(x_N) & h_1(x_N) & \dots & h_K(x_N) \end{bmatrix}$$

Making predictions

$$\widehat{\mathbf{w}} = (\mathbf{X}^\mathsf{T}\mathbf{X})^{-1}\mathbf{X}^\mathsf{T}\mathbf{t}$$

Where $\mathbf{X}$ depends on the choice of model:

$$\mathbf{X} = \begin{bmatrix} h_0(x_1) & h_1(x_1) & \dots & h_K(x_1) \\ \vdots & \vdots & \ddots & \vdots \\ h_0(x_N) & h_1(x_N) & \dots & h_K(x_N) \end{bmatrix}$$

To predict t at a new value of x, we first create $\mathbf{x}_{\text{new}}$:

$$\mathbf{x}_{\text{new}} = \begin{bmatrix} h_0(x_{\text{new}}) \\ \vdots \\ h_K(x_{\text{new}}) \end{bmatrix},$$

Making predictions

$$\widehat{\mathbf{w}} = (\mathbf{X}^\mathsf{T}\mathbf{X})^{-1}\mathbf{X}^\mathsf{T}\mathbf{t}$$

Where $\mathbf{X}$ depends on the choice of model:

$$\mathbf{X} = \begin{bmatrix} h_0(x_1) & h_1(x_1) & \dots & h_K(x_1) \\ \vdots & \vdots & \ddots & \vdots \\ h_0(x_N) & h_1(x_N) & \dots & h_K(x_N) \end{bmatrix}$$

To predict t at a new value of x, we first create $\mathbf{x}_{\mathrm{new}}$:

$$\mathbf{x}_{\mathrm{new}} = \begin{bmatrix} h_0(x_{\mathrm{new}}) \\ \vdots \\ h_K(x_{\mathrm{new}}) \end{bmatrix},$$

and then compute

$$t_{\mathrm{new}} = \widehat{\mathbf{w}}^\mathsf{T}\mathbf{x}_{\mathrm{new}}$$

Summary

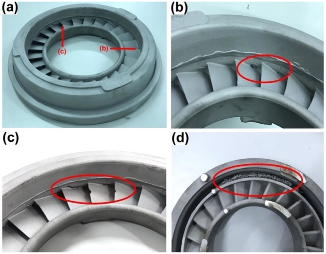
- Formulated our loss in terms of vectors and matrices
- Solved for best $\mathbf{w}$ (minimising the loss)
- Saw examples of models with differing numbers of terms
- Introduced basis functions

High-dimensional data

- input: image / audio / time-series / etc.

-   $\rightarrow$ **age**

High-dimensional data

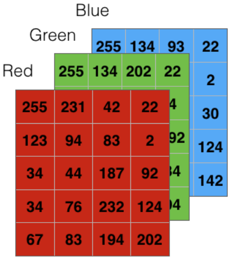
- input: image / audio / time-series / etc.
-  → **age**
- 

→ **defect severity**

Images: Karras, NeurIPS 2021; Huang, IJAMT 2019

Images

- Flatten images to vectors:

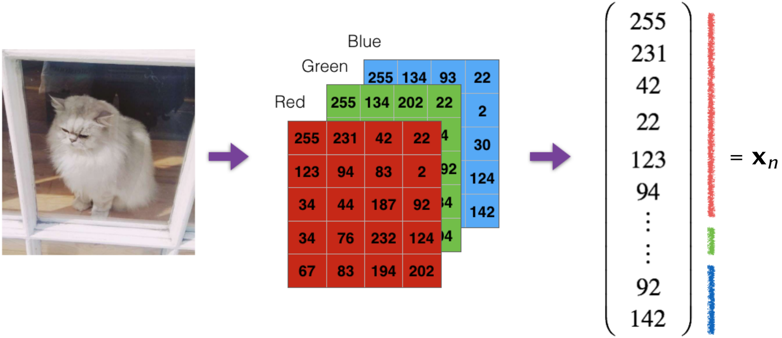


$$\begin{pmatrix} 255 \\ 231 \\ 42 \\ 22 \\ 123 \\ 94 \\ \vdots \\ \vdots \\ 92 \\ 142 \end{pmatrix} = \mathbf{x}_n$$

Photo: L Benoist; Diagram: S Dixon

Images

- Flatten images to vectors:



- $\mathbf{x}_n$ has millions of elements!

Photo: L Benoist; Diagram: S Dixon

High-dimensional data

1. **normalise / standardise**
   - centre faces in image
   - subtract average intensity; divide by standard deviation

High-dimensional data

1. normalise / standardise
    - centre faces in image
    - subtract average intensity; divide by standard deviation

2. **remove redundancy**
    - subsample
    - convert to greyscale / simpler representation

High-dimensional data

1. normalise / standardise
   - centre faces in image
   - subtract average intensity; divide by standard deviation
2. remove redundancy
   - subsample
   - convert to greyscale / simpler representation
3. **extract features**
   - landmark locations
   - blob attributes
   - *much lower-dimensional than pixels!*

Feature scaling

- attributes/features may differ in scale:

$$\mathbf{X} = \begin{pmatrix} 1 & 1896 & 21 \\ 1 & 1900 & 25 \\ \vdots & \vdots & \vdots \\ 1 & 2008 & 23 \end{pmatrix}$$

Feature scaling

- attributes/features may differ in scale:

$$\mathbf{X} = \begin{pmatrix} 1 & 1896 & 21 \\ 1 & 1900 & 25 \\ \vdots & \vdots & \vdots \\ 1 & 2008 & 23 \end{pmatrix}$$

- subtract mean
- divide by standard deviation

$$\mathbf{X} = \begin{pmatrix} 1 & -1.6 & -0.1 \\ 1 & -1.5 & 0.7 \\ \vdots & \vdots & \vdots \\ 1 & 1.6 & 0.3 \end{pmatrix}$$