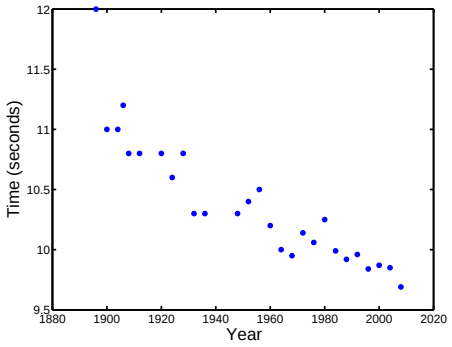
TDA231
Multivariate and non-linear models

Devdatt Dubhashi
dubhashi@chalmers.se

Dept of Computer Science and Engg.
Chalmers University

March 22, 2018

Data and problem



Winning times for the men's Olympic 100m sprint, 1896-2008.

Problem: Predict Olympic winning time in 2012.
Generally:

- Data: $(x_1, y_1), (x_2, y_2), \cdots, (x_N, y_N)$.
- Predict y_{new} corresponding to x_{new}.

Introduction
D. Dubhashi
Recap
Introduction
Vectors
Matrices
Objectives
Minimising the loss
Examples
Predictions
Methods for choosing models
Generalisation and over-fitting
Validation
Cross-validation
Summary

Model and Loss

- Linear model: $t = f(x) = w_0 + w_1 x$.
- The average loss:

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

- $\mathcal{L}$ tells us how good the model is as a function of w_0 and w_1.

Introduction
D. Dubhashi
Recap
Introduction
Vectors
Matrices
Objectives
Minimising the loss
Examples
Predictions
Methods for choosing models
Generalisation and over-fitting
Validation
Cross-validation
Summary

Optimization

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

Solution:

$$w_1 = \frac{\overline{xt} - \bar{x}\bar{t}}{\overline{x^2} - (\bar{x})^2}, \quad w_0 = \bar{t} - w_1\bar{x}$$

Where

$$\bar{t} = \frac{1}{N}\sum_{n=1}^{N} t_n,\ \bar{x} = \frac{1}{N}\sum_{n=1}^{N} x_n,\ \overline{x^2} = \frac{1}{N}\sum_{n=1}^{N} x_n^2,\ \overline{xt} = \frac{1}{N}\sum_{n=1}^{N} x_n t_n$$

Introduction
D. Dubhashi
Recap
Introduction
Vectors
Matrices
Objectives
Minimising the loss
Examples
Predictions
Methods for choosing models
Generalisation and over-fitting
Validation
Cross-validation
Summary

Overview

- Linear model in vector form.
- More general formulation.
  - Output can depend on several input variables
  - The function could be non–linear.
- Making predictions.
- Generalisation, overfitting, cross-validation.

Linear model in vector form

- Simple model: $t = w_0 + w_1 x$.
- To find $\widehat{w_0}, \widehat{w_1}$:
  - Take partial derivative of loss with respect to each parameter.
  - Set to zero and solve (2 simultaneous equations)

Linear model in vector form

- Simple model: $t = w_0 + w_1 x$.
- To find $\widehat{w_0}, \widehat{w_1}$:
  - Take partial derivative of loss with respect to each parameter.
  - Set to zero and solve (2 simultaneous equations)
- More complex model (polynomial):

$$t = w_0 + w_1 x + w_2 x^2 + w_3 x^2 + \ldots + w_K x^K = \sum_{k=0}^{K} w_k x^k$$

- To find $\widehat{w_0}, \ldots, \widehat{w_K}$:
  - Define loss $\mathcal{L} = \frac{1}{N} \sum_{n=1}^{N} \left(t_n - \sum_{k=0}^{K} w_k x^k \right)^2$
  - Differentiate loss with respect to every parameter
  - Set to zero and solve (K simultaneous equations)

Linear model in vector form

- Simple model: $t = w_0 + w_1 x$.
- To find $\widehat{w_0}, \widehat{w_1}$:
  - Take partial derivative of loss with respect to each parameter.
  - Set to zero and solve (2 simultaneous equations)
- More complex model (polynomial):

$$t = w_0 + w_1 x + w_2 x^2 + w_3 x^2 + \ldots + w_K x^K = \sum_{k=0}^{K} w_k x^k$$

- To find $\widehat{w_0}, \ldots, \widehat{w_K}$:
  - Define loss $\mathcal{L} = \frac{1}{N} \sum_{n=1}^{N} \left(t_n - \sum_{k=0}^{K} w_k x^k \right)^2$
  - Differentiate loss with respect to every parameter
  - Set to zero and solve (K simultaneous equations)
- Very tedious! Use vector/matrix notation instead.

Encapsulate parameters in a vector

- Sticking with our linear model: $t = w_0 + w_1 x$ (length 2 vectors are easier to work with).

Encapsulate parameters in a vector

- Sticking with our linear model: $t = w_0 + w_1 x$ (length 2 vectors are easier to work with).
- We can combine the parameters into a vector:

Encapsulate parameters in a vector

- Sticking with our linear model: $t = w_0 + w_1 x$ (length 2 vectors are easier to work with).
- We can combine the parameters into a vector:

$$w_0$$
$$w_1$$

Encapsulate parameters in a vector

- Sticking with our linear model: $t = w_0 + w_1 x$ (length 2 vectors are easier to work with).
- We can combine the parameters into a vector:

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

- Vectors are bold, lowercase letters.
- A list of values – similar to arrays when programming.
  - If you've never seen them before – do some reading!

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:

$$\mathbf{w} = \begin{bmatrix} w_0 \\ w_1 \end{bmatrix}, \ \mathbf{x} = \begin{bmatrix} x^0 \\ x^1 \end{bmatrix}$$

- Make sure you're happy with this notation. There is a lengthy description and exercises in the book chapter.

- Vector model:

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

- Loss:

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

- Total loss:

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

- Vector model:
$$t = \mathbf{w}^\mathsf{T}\mathbf{x}$$
- Loss:
$$\mathcal{L}_n = (t_n - \mathbf{w}^\mathsf{T}\mathbf{x}_n)^2.$$
- Total 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:
$$\mathcal{L}_n = (t_n - \mathbf{w}^\mathsf{T}\mathbf{x}_n)^2.$$
- Total 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:
$$\sum_{d=1}^{D} a_d^2 = \mathbf{a}^\mathsf{T}\mathbf{a}$$

Introduction
D. Dubhashi
Recap
Introduction
Vectors
Matrices
Objectives
Minimising the loss
Examples
Predictions
Methods for choosing models
Generalisation and over-fitting
Validation
Cross-validation
Summary

- Vector model:
$$t = \mathbf{w}^\mathsf{T}\mathbf{x}$$
- Loss:
$$\mathcal{L}_n = (t_n - \mathbf{w}^\mathsf{T}\mathbf{x}_n)^2.$$
- Total 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:
$$\sum_{d=1}^{D} a_d^2 = \mathbf{a}^\mathsf{T}\mathbf{a}$$
- So:
$$\mathcal{L} = \frac{1}{N}\sum_{n=1}^{N} q_n^2 = \frac{1}{N}\mathbf{q}^\mathsf{T}\mathbf{q}.$$

Introduction
D. Dubhashi
Recap
Introduction
Vectors
Matrices
Objectives
Minimising the loss
Examples
Predictions
Methods for choosing models
Generalisation and over-fitting
Validation
Cross-validation
Summary

What is $\mathbf{q}$?

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

Therefore...

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

Introduction
D. Dubhashi
Recap
Introduction
Vectors
Matrices
Objectives
Minimising the loss
Examples
Predictions
Methods for choosing models
Generalisation and over-fitting
Validation
Cross-validation
Summary

What is $\mathbf{q}$?

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

Therefore...

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

Subtraction

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

What is $\mathbf{q}$?

Define

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

What is $\mathbf{q}$?

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 $\mathbf{x}_n^\mathsf{T}$ on top of one another:

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

Matrices

- Stack all $\mathbf{x}_n^\mathsf{T}$ 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.
- Matrices are bold, uppercase letters.

What is $\mathbf{q}$?

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

What is $\mathbf{q}$?

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

And the total loss is:

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

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.

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

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

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

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

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

- When minimising the scalar loss

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

- we took partial derivatives with respect to each parameter and set to zero.

Minimising the loss

- When minimising the scalar loss

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

- we took partial derivatives with respect to each parameter and set to zero.
- We now have a vector/matrix loss

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

- and will take partial derivatives with respect to the vector $\mathbf{w}$ and set to zero:

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

Partial diff. wrt vector

The result of taking the partial derivative with respect to a vector is a vector where each element is the partial derivative with respect to one parameter:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{w}} = \begin{bmatrix} \frac{\partial \mathcal{L}}{\partial w_0} \\ \frac{\partial \mathcal{L}}{\partial w_1} \\ \vdots \\ \frac{\partial \mathcal{L}}{\partial w_K} \end{bmatrix}$$

Partial diff. wrt vector

The result of taking the partial derivative with respect to a vector is a vector where each element is the partial derivative with respect to one parameter:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{w}} = \begin{bmatrix} \frac{\partial \mathcal{L}}{\partial w_0} \\ \frac{\partial \mathcal{L}}{\partial w_1} \\ \vdots \\ \frac{\partial \mathcal{L}}{\partial w_K} \end{bmatrix}$$

Useful identites:

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

Computing $\frac{\partial \mathcal{L}}{\partial \mathbf{w}}$

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

Matrix transpose

$$\mathbf{X} = \begin{bmatrix} x_{11} & x_{12} \\ x_{21} & x_{22} \\ x_{31} & x_{32} \end{bmatrix}, \ \mathbf{X}^\mathsf{T} = \begin{bmatrix} x_{11} & x_{21} & x_{31} \\ x_{12} & x_{22} & x_{32} \end{bmatrix}$$

Transpose of sum/product

$$(\mathbf{a}+\mathbf{b})^\mathsf{T} = \mathbf{a}^\mathsf{T} + \mathbf{b}^\mathsf{T}, \ (\mathbf{X}\mathbf{w})^\mathsf{T} = \mathbf{w}^\mathsf{T}\mathbf{X}^\mathsf{T}$$

Computing $\frac{\partial \mathcal{L}}{\partial \mathbf{w}}$

$$\begin{aligned} \frac{\partial}{\partial \mathbf{w}}\left(\frac{1}{N}(\mathbf{t}-\mathbf{X}\mathbf{w})^\mathsf{T}(\mathbf{t}-\mathbf{X}\mathbf{w})\right) &= \frac{1}{N}(2\mathbf{X}^\mathsf{T}\mathbf{X}\mathbf{w} - 2\mathbf{X}^\mathsf{T}\mathbf{t}) = \mathbf{0} \\ \mathbf{X}^\mathsf{T}\mathbf{X}\mathbf{w} &= \mathbf{X}^\mathsf{T}\mathbf{t} \end{aligned}$$

Matrix transpose

$$\mathbf{X} = \begin{bmatrix} x_{11} & x_{12} \\ x_{21} & x_{22} \\ x_{31} & x_{32} \end{bmatrix}, \ \mathbf{X}^\mathsf{T} = \begin{bmatrix} x_{11} & x_{21} & x_{31} \\ x_{12} & x_{22} & x_{32} \end{bmatrix}$$

Transpose of sum/product

$$(\mathbf{a}+\mathbf{b})^\mathsf{T} = \mathbf{a}^\mathsf{T} + \mathbf{b}^\mathsf{T}, \ (\mathbf{X}\mathbf{w})^\mathsf{T} = \mathbf{w}^\mathsf{T}\mathbf{X}^\mathsf{T}$$

Computing $\frac{\partial \mathcal{L}}{\partial \mathbf{w}}$

$$\mathbf{X}^\top \mathbf{X}\mathbf{w} = \mathbf{X}^\top \mathbf{t}$$

Computing $\frac{\partial \mathcal{L}}{\partial \mathbf{w}}$

$$\mathbf{X}^\top \mathbf{X}\mathbf{w} = \mathbf{X}^\top \mathbf{t}$$

Matrix inverse

Inverse is defined (for a square matrix $\mathbf{A}$) as the matrix $\mathbf{A}^{-1}$ that satisfies:

$$\mathbf{A}\mathbf{A}^{-1} = \mathbf{I}$$

Where $\mathbf{I}$ is the *identity* matrix,

$$\mathbf{I} = \begin{bmatrix} 1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 1 \end{bmatrix}, \text{ and } \mathbf{I}\mathbf{A} = \mathbf{A}, \text{ for any } \mathbf{A}$$

Computing $\frac{\partial \mathcal{L}}{\partial \mathbf{w}}$

$$\begin{aligned} \mathbf{X}^\top \mathbf{X}\mathbf{w} &= \mathbf{X}^\top \mathbf{t} \\ (\mathbf{X}^\top \mathbf{X})^{-1}\mathbf{X}^\top \mathbf{X}\mathbf{w} &= (\mathbf{X}^\top \mathbf{X})^{-1}\mathbf{X}^\top \mathbf{t} \end{aligned}$$

Matrix inverse

Inverse is defined (for a square matrix $\mathbf{A}$) as the matrix $\mathbf{A}^{-1}$ that satisfies:

$$\mathbf{A}\mathbf{A}^{-1} = \mathbf{I}$$

Where $\mathbf{I}$ is the *identity* matrix,

$$\mathbf{I} = \begin{bmatrix} 1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 1 \end{bmatrix}, \text{ and } \mathbf{I}\mathbf{A} = \mathbf{A}, \text{ for any } \mathbf{A}$$

Computing $\frac{\partial \mathcal{L}}{\partial \mathbf{w}}$

$$\begin{aligned} \mathbf{X}^\top \mathbf{X}\mathbf{w} &= \mathbf{X}^\top \mathbf{t} \\ (\mathbf{X}^\top \mathbf{X})^{-1}\mathbf{X}^\top \mathbf{X}\mathbf{w} &= (\mathbf{X}^\top \mathbf{X})^{-1}\mathbf{X}^\top \mathbf{t} \\ \mathbf{w} &= (\mathbf{X}^\top \mathbf{X})^{-1}\mathbf{X}^\top \mathbf{t} \end{aligned}$$

Matrix inverse

Inverse is defined (for a square matrix $\mathbf{A}$) as the matrix $\mathbf{A}^{-1}$ that satisfies:

$$\mathbf{A}\mathbf{A}^{-1} = \mathbf{I}$$

Where $\mathbf{I}$ is the *identity* matrix,

$$\mathbf{I} = \begin{bmatrix} 1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 1 \end{bmatrix}, \text{ and } \mathbf{I}\mathbf{A} = \mathbf{A}, \text{ for any } \mathbf{A}$$

Summary

- Introduced partial differentiation with respect to a vector.
- Introduced matrix transpose and inverse and identity matrix.
- Have 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

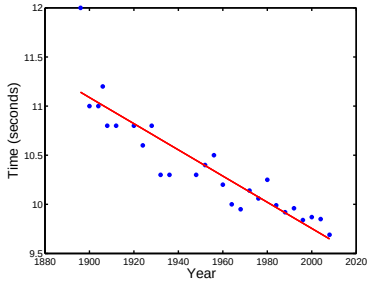
$$\mathbf{w} = \begin{bmatrix} w_0 \\ w_1 \end{bmatrix}, \mathbf{X} = \begin{bmatrix} 1 & 1896 \\ 1 & 1900 \\ \vdots & \\ 1 & 2008 \end{bmatrix}, \mathbf{t} = \begin{bmatrix} 12.00 \\ 11.00 \\ \vdots \\ 9.85 \end{bmatrix}$$

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

Linear model - Olympic data

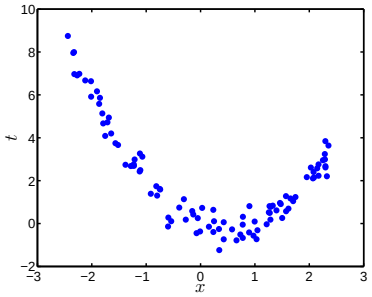
$$\mathbf{w} = \begin{bmatrix} w_0 \\ w_1 \end{bmatrix}, \mathbf{X} = \begin{bmatrix} 1 & 1896 \\ 1 & 1900 \\ \vdots & \\ 1 & 2008 \end{bmatrix}, \mathbf{t} = \begin{bmatrix} 12.00 \\ 11.00 \\ \vdots \\ 9.85 \end{bmatrix}$$

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



Quadratic model - synthetic data

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

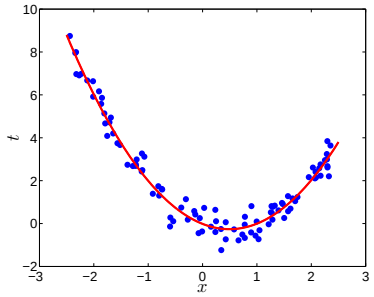


Quadratic model - synthetic data

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

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

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



8th order model - Olympic data

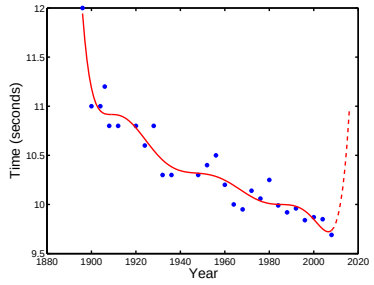
$$t = w_0 + w_1 x + w_2 x^2 + \ldots + w_8 x^8$$

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

8th order model - Olympic data

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

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



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

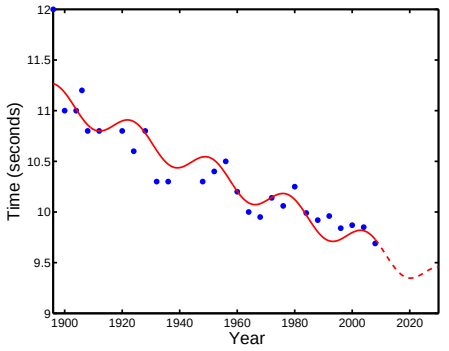
$$t = w_0 + w_1 x + w_2 \sin\left(\frac{x-a}{b}\right)$$

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

Example – Olympic data

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

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



Summary

- Formulated our loss in terms of vectors and matrices.
- Differentiated it with respect to the parameter vector.
- Used this to find a general expression for $\widehat{\mathbf{w}}$ - the parameters that minimise the loss.
- Shown examples of models with differing numbers of terms.
- Not restricted to x^K - can have any function of x.
- Shown example of model including a sin term.

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:

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

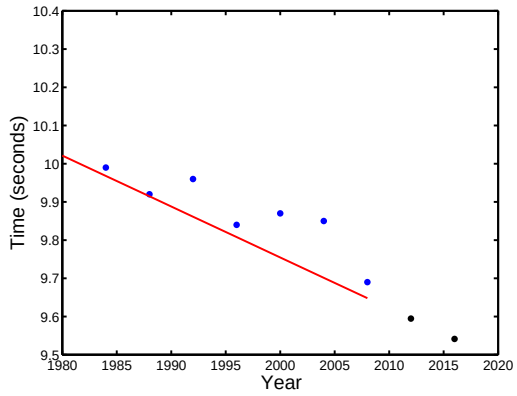
To predict t at a new value of x, we first create $\mathbf{x}_{\text{new}}$:

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

and then compute

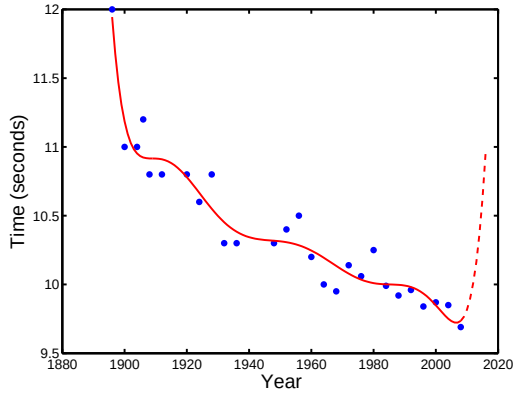
$$t_{\text{new}} = \widehat{\mathbf{w}}^\mathsf{T}\mathbf{x}_{\text{new}}$$

Example - Olympic data



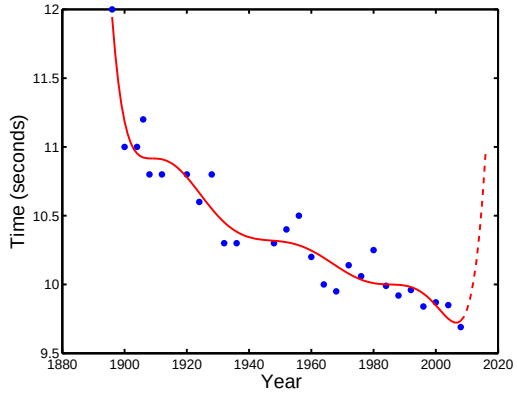
Linear model – predictions OK?

Example - Olympic data



8th order model – predictions terrible!

Example - Olympic data



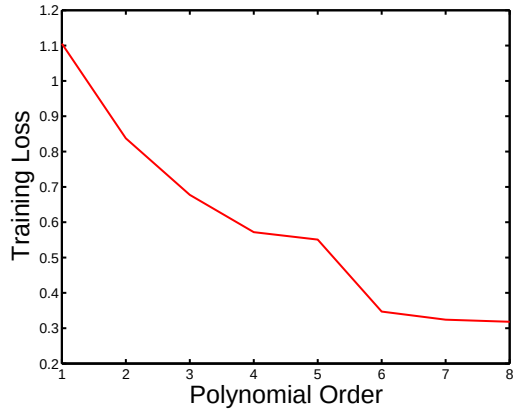
8th order model – predictions terrible!

Choice of model is **very** important.

Possible ways of choosing

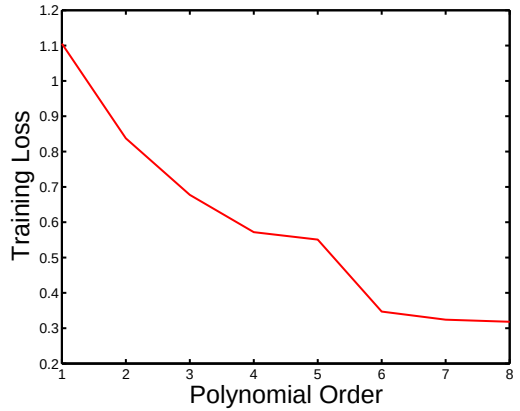
- Lowest loss, $\mathcal{L}$?

How does loss change?



Loss, L, on the Olympic 100m data as additional terms (x^k) are added to the model.

How does loss change?

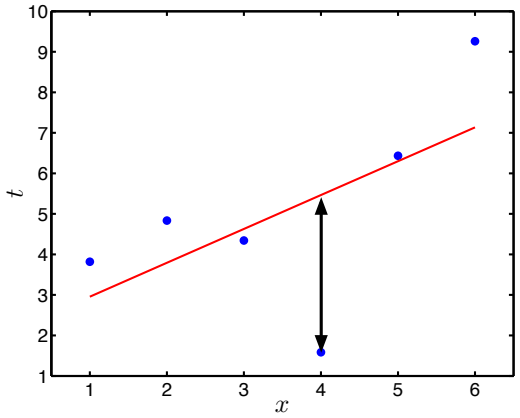


Loss, L, on the Olympic 100m data as additional terms (x^k) are added to the model.

Loss **always** decreases as the model is made more complex (i.e. higher order terms are added)

Loss always decreases with model complexity

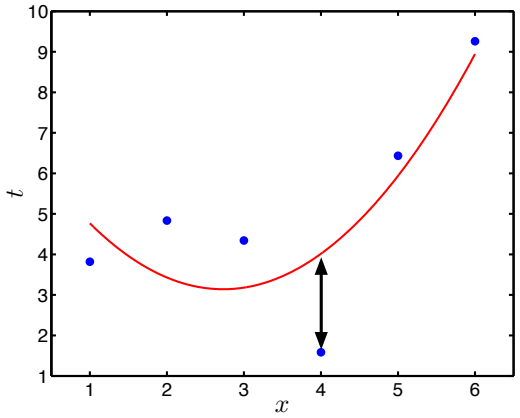
Data comes from $t = x$ with some *noise* added:



Linear model $t = w_0 + w_1 x$.

Loss always decreases with model complexity

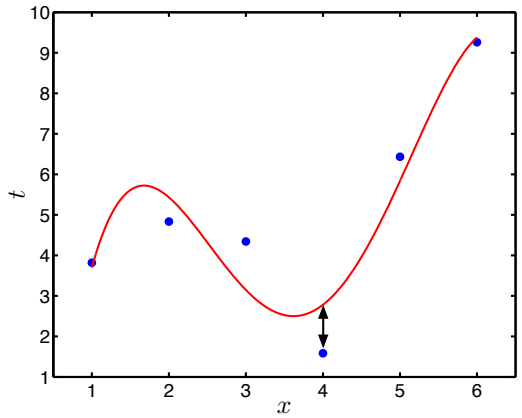
Data comes from $t = x$ with some *noise* added:



Quadratic model $t = w_0 + w_1 x + w_2 x^2$.

Loss always decreases with model complexity

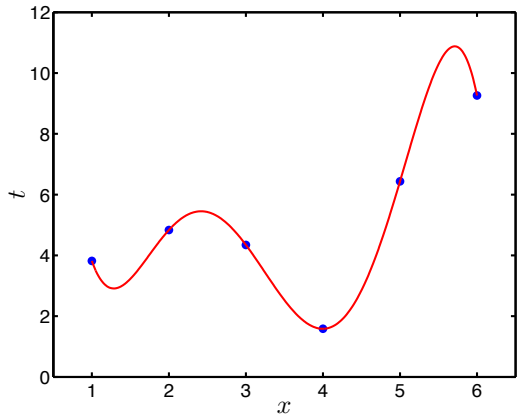
Data comes from $t = x$ with some *noise* added:



Fourth order $t = w_0 + w_1 x + w_2 x^2 + w_3 x^3 + w_4 x^4$.

Loss always decreases with model complexity

Data comes from $t = x$ with some *noise* added:



Fifth order $t = w_0 + w_1 x + w_2 x^2 + w_3 x^3 + w_4 x^4 + w_5 x^5$.

Generalisation and over-fitting

There is a trade-off between generalisation (predictive ability) and over-fitting (decreasing the loss).

Generalisation and over-fitting

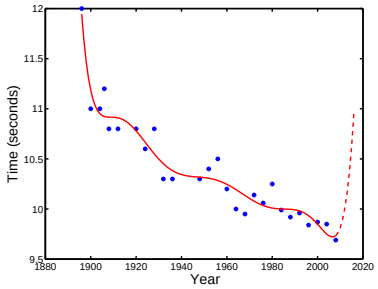
There is a trade-off between generalisation (predictive ability) and over-fitting (decreasing the loss).

- Fitting a model perfectly to the training data is likely to lead to poor predictions because there will almost always be *noise* present.

Generalisation and over-fitting

There is a trade-off between generalisation (predictive ability) and over-fitting (decreasing the loss).

- Fitting a model perfectly to the training data is likely to lead to poor predictions because there will almost always be *noise* present.



Noise

Not necessarily 'noise', just things we can't, or don't need to model.

Possible ways of choosing

- Lowest loss, $\mathcal{L}$?
  - Loss always decreases as model gets more complex.

Possible ways of choosing

- Lowest loss, $\mathcal{L}$?
  - Loss always decreases as model gets more complex.
  - Predictions don't necessarily get better.

Possible ways of choosing

- Lowest loss, $\mathcal{L}$?
  - Loss always decreases as model gets more complex.
  - Predictions don't necessarily get better.
- Best predictions?

Possible ways of choosing

- Lowest loss, $\mathcal{L}$?
  - Loss always decreases as model gets more complex.
  - Predictions don't necessarily get better.
- Best predictions?
  - Can't use future predictions because we don't know the answer!

Possible ways of choosing

- Lowest loss, $\mathcal{L}$?
  - Loss always decreases as model gets more complex.
  - Predictions don't necessarily get better.
- Best predictions?
  - Can't use future predictions because we don't know the answer!
  - Other data?

Where can we get more data?

- We have N input-response pairs for training:

$$(x_1, t_1), (x_2, t_2), \ldots, (x_N, t_N).$$

Where can we get more data?

- We have N input-response pairs for training:

$$(x_1, t_1), (x_2, t_2), \ldots, (x_N, t_N).$$

- We could use $N - C$ pairs to find $\widehat{\mathbf{w}}$ for several models.

Where can we get more data?

- We have N input-response pairs for training:

$$(x_1, t_1), (x_2, t_2), \ldots, (x_N, t_N).$$

- We could use $N - C$ pairs to find $\widehat{\mathbf{w}}$ for several models.
- Choose the model that makes best predictions on remaining C pairs.

Where can we get more data?

- We have N input-response pairs for training:

$$(x_1, t_1), (x_2, t_2), \ldots, (x_N, t_N).$$

- We could use $N - C$ pairs to find $\widehat{\mathbf{w}}$ for several models.
- Choose the model that makes best predictions on remaining C pairs.
  - The $N - C$ pairs constitute *training data*.
  - The C pairs are known as *validation data*.

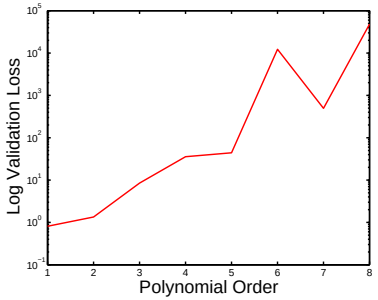
Where can we get more data?

- We have N input-response pairs for training:

$$(x_1, t_1), (x_2, t_2), \ldots, (x_N, t_N).$$

- We could use $N - C$ pairs to find $\widehat{\mathbf{w}}$ for several models.
- Choose the model that makes best predictions on remaining C pairs.
  - The $N - C$ pairs constitute *training data*.
  - The C pairs are known as *validation data*.
- Example – use Olympics pre 1980 to train and post 1980 to validate.

Validation example



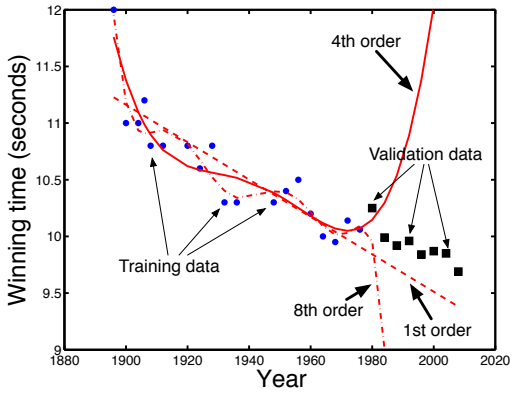
Predictions evaluated using validation loss:

$$\mathcal{L}_v = \frac{1}{C}\sum_{c=1}^{C}(t_c - \mathbf{w}^{\mathsf{T}}\mathbf{x}_c)^2$$

Best model?

Results suggest that a first order (linear) model $(t = w_0 + w_1 x)$ is best.

Validation example



Best model

First order (linear) model generalises best.

How should we choose which data to hold back?

- In some applications it will be clear.
  - Olympic data – validating on the most recent data seems sensible.
- In many cases – pick it randomly.

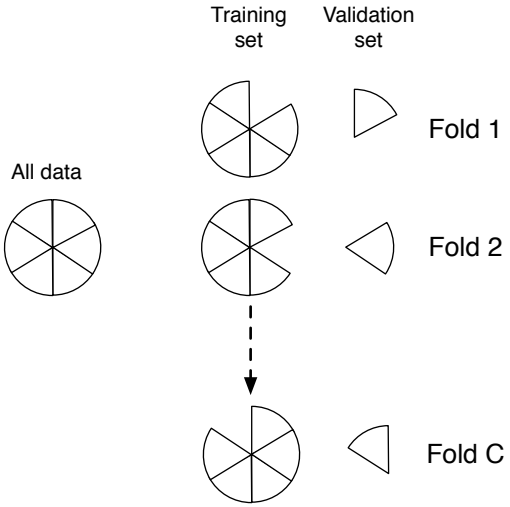
How should we choose which data to hold back?

- In some applications it will be clear.
  - Olympic data – validating on the most recent data seems sensible.
- In many cases – pick it randomly.
- Do it more than once – average the results.

How should we choose which data to hold back?

- In some applications it will be clear.
  - Olympic data – validating on the most recent data seems sensible.
- In many cases – pick it randomly.
- Do it more than once – average the results.
- Do cross-validation.
  - Split the data into C equal sets. Train on $C-1$, test on remaining.

Cross-validation



Average performance over the C 'folds'.

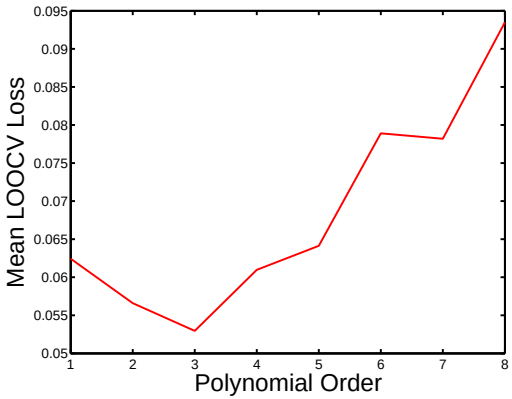
Leave-one-out Cross-validation

- Cross-validation can be repeated to make results more accurate.
- e.g. Doing 10-fold CV 10 times gives us 100 performance values to average over.

Leave-one-out Cross-validation

- Cross-validation can be repeated to make results more accurate.
- e.g. Doing 10-fold CV 10 times gives us 100 performance values to average over.
- Extreme example is when $C = N$ so each fold includes one input-response pair.
  - Leave-one-out (LOO) CV.
- Example....

LOOCV – Olympic data



Best model?

LOO CV suggests a 3rd order model. Previous method suggests 1st order. Who knows which is right!

LOOCV – synthetic data (we know the answer!)

- Generate some data from a 3rd order model

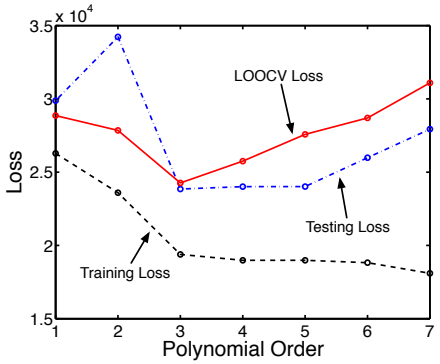
$$t = w_0 + w_1 x + w_2 x^2 + w_3 x^3.$$

LOOCV – synthetic data (we know the answer!)

- Generate some data from a 3rd order model

$$t = w_0 + w_1 x + w_2 x^2 + w_3 x^3.$$

- Use LOOCV to compare models from first to 7th order:



(Testing loss comes from another dataset)

Computational issues

- CV and LOOCV let us choose from a set of models based on predictive performance.
- This comes at a computational cost:

Computational issues

- CV and LOOCV let us choose from a set of models based on predictive performance.
- This comes at a computational cost:
  - For C-fold CV, need to train our model C times.
  - For LOO-CV, need to train out model N times.

Computational issues

- CV and LOOCV let us choose from a set of models based on predictive performance.
- This comes at a computational cost:
  - For C-fold CV, need to train our model C times.
  - For LOO-CV, need to train out model N times.
- For $t = \mathbf{w}^{\mathsf{T}}\mathbf{x}$, this is feasible if K (number of terms in function) isn't too big:

$$\begin{aligned} t &= \sum_{k=0}^{K} w_k h_k(x) \\ \widehat{\mathbf{w}} &= (\mathbf{X}^{\mathsf{T}}\mathbf{X})^{-1}\mathbf{X}^{\mathsf{T}}\mathbf{t} \end{aligned}$$

Computational issues

- CV and LOOCV let us choose from a set of models based on predictive performance.
- This comes at a computational cost:
  - For C-fold CV, need to train our model C times.
  - For LOO-CV, need to train out model N times.
- For $t = \mathbf{w}^{\mathsf{T}}\mathbf{x}$, this is feasible if K (number of terms in function) isn't too big:

$$\begin{aligned} t &= \sum_{k=0}^{K} w_k h_k(x) \\ \widehat{\mathbf{w}} &= (\mathbf{X}^{\mathsf{T}}\mathbf{X})^{-1}\mathbf{X}^{\mathsf{T}}\mathbf{t} \end{aligned}$$

- For some modelswill we will need to use $C \ll N$.

Introduction
D. Dubhashi
Recap
Introduction
Vectors
Matrices
Objectives
Minimising the loss
Examples
Predictions
Methods for choosing models
Generalisation and over-fitting
Validation
Cross-validation
Summary

Summary

- Showed how we can make predictions with our 'linear' model.
- Saw how choice of model has big influence in quality of predictions.
- Saw how the loss on the training data, $\mathcal{L}$, cannot be used to choose models.
  - Making model more complex always decreases the loss.
- Introduced the idea of using some data for validation.
- Introduced cross validation and leave-one-out cross validation.

Introduction
D. Dubhashi
Recap
Introduction
Vectors
Matrices
Objectives
Minimising the loss
Examples
Predictions
Methods for choosing models
Generalisation and over-fitting
Validation
Cross-validation
Summary