Linear Modeling and Regression

Morteza H. Chehreghani
morteza.chehreghani@chalmers.se

Department of Computer Science and Engineering
Chalmers University of Technology

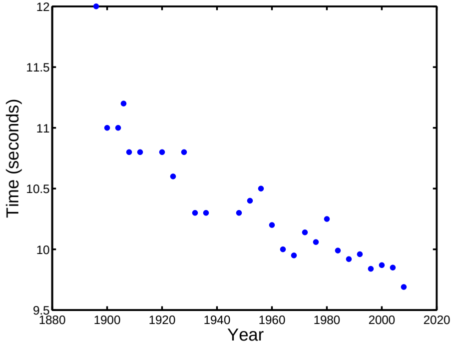
January 15, 2024

Reference

The content and the slides are adapted from

S. Rogers and M. Girolami, A First Course in Machine Learning (FCML), 2nd edition, Chapman & Hall/CRC 2016, ISBN: 9781498738484

Some data and a problem



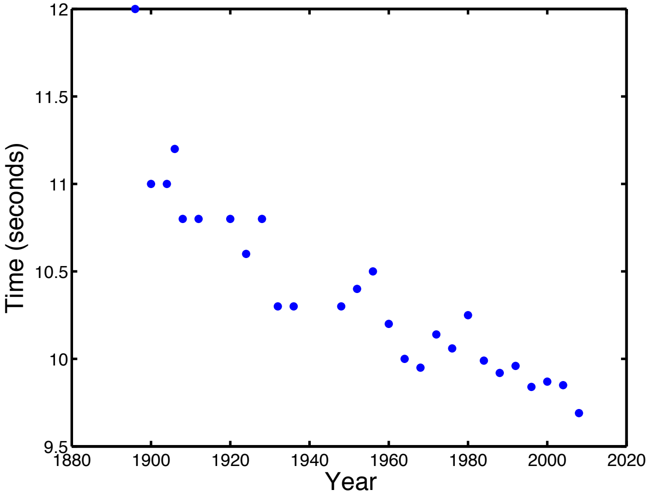
Winning times for the men's Olympic 100m sprint, 1896-2008.

In this lecture, we will use this data to predict the winning time in London 2012

Reading: Section 1.1 of FCML

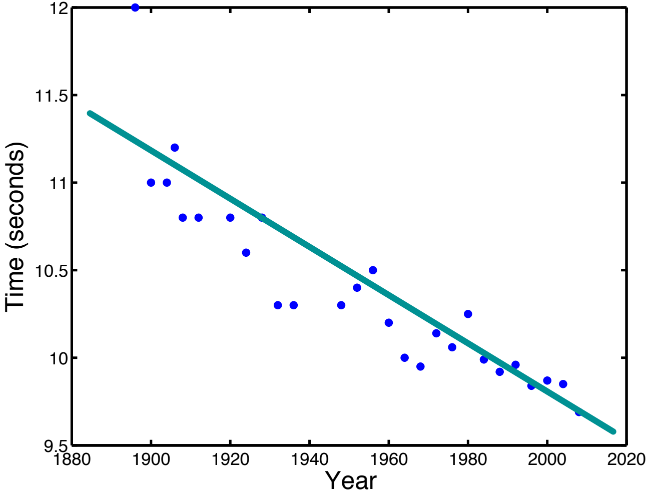
Back of envelope calculation

Draw a line through it!



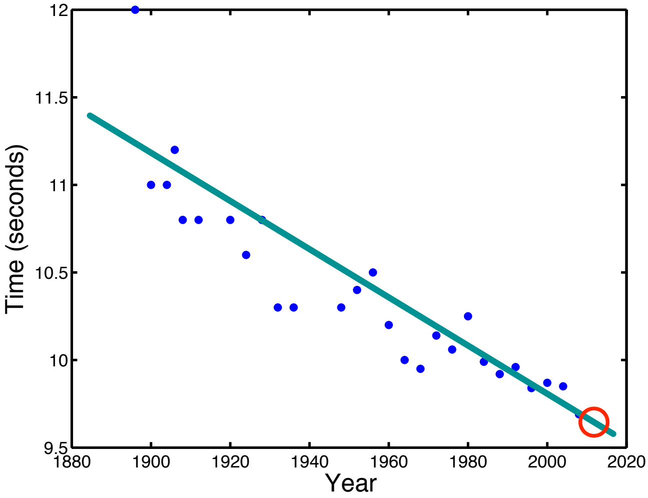
Back of envelope calculation

Draw a line through it!



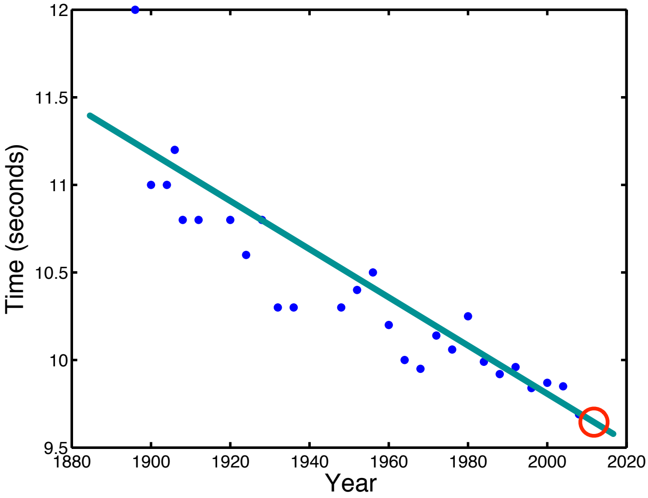
Back of envelope calculation

Read the winning time for 2012.



Back of envelope calculation

Read the winning time for 2012.



Our aim is to formalise this process.

What did we do?

Basically:

- **Decided to draw a line through our data.**
- Chose a straight line.
- Drew a good straight line.
- Extended the line to 2012.
- Read off the winning time for 2012.

Technically

- Decided we needed a model.
- Chose a linear model.
- Fitted a linear model.
- Evaluated the model at 2012.
- Used this as our prediction.

What did we do?

Basically:

- Decided to draw a line through our data.
- **Chose a straight line.**
- Drew a good straight line.
- Extended the line to 2012.
- Read off the winning time for 2012.

Technically

- Decided we needed a model.
- Chose a linear model.
- Fitted a linear model.
- Evaluated the model at 2012.
- Used this as our prediction.

What did we do?

Basically:

- Decided to draw a line through our data.
- Chose a straight line.
- **Drew a good straight line.**
- Extended the line to 2012.
- Read off the winning time for 2012.

Technically

- Decided we needed a model.
- Chose a linear model.
- Fitted a linear model.
- Evaluated the model at 2012.
- Used this as our prediction.

What did we do?

Basically:

- Decided to draw a line through our data.
- Chose a straight line.
- Drew a good straight line.
- **Extended the line to 2012.**
- Read off the winning time for 2012.

Technically

- Decided we needed a model.
- Chose a linear model.
- Fitted a linear model.
- Evaluated the model at 2012.
- Used this as our prediction.

What did we do?

Basically:

- Decided to draw a line through our data.
- Chose a straight line.
- Drew a good straight line.
- Extended the line to 2012.
- **Read off the winning time for 2012.**

Technically

- Decided we needed a model.
- Chose a linear model.
- Fitted a linear model.
- Evaluated the model at 2012.
- Used this as our prediction.

Assumptions

Our Assumptions

1. **That there exists a relationship between Olympic year and winning time.**

Assumptions

Our Assumptions

1. That there exists a relationship between Olympic year and winning time.
2. **That this relationship is linear (i.e. a straight line).**

Assumptions

Our Assumptions

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

Assumptions

Our Assumptions

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

Are they any good?

Definitions

Attributes (features) and targets (responses)

Typically in Supervised Machine Learning, we have a set of attributes and corresponding targets:

Definitions

Attributes (features) and targets (responses)

Typically in Supervised Machine Learning, we have a set of **attributes** and corresponding targets:

- **Attributes:** Olympic year.

Definitions

Attributes (features) and targets (responses)

Typically in Supervised Machine Learning, we have a set of attributes and corresponding **targets**:

- **Attributes:** Olympic year.
- **Targets:** Winning time.

Definitions

Attributes (features) and targets (responses)

Typically in Supervised Machine Learning, we have a set of attributes and corresponding targets:

- **Attributes:** Olympic year.
- **Targets:** Winning time.

Variables

Mathematically, each is described by a variable:

Definitions

Attributes (features) and targets (responses)

Typically in Supervised Machine Learning, we have a set of attributes and corresponding targets:

- **Attributes:** Olympic year.
- **Targets:** Winning time.

Variables

Mathematically, each is described by a variable:

- Olympic year: x.

Definitions

Attributes (features) and targets (responses)

Typically in Supervised Machine Learning, we have a set of attributes and corresponding targets:

- **Attributes:** Olympic year.
- **Targets:** Winning time.

Variables

Mathematically, each is described by a variable:

- Olympic year: x.
- Winning time: t.

Definitions

Model

Our goal is to create a model.

- This is a function that can relate x to t.

$$t = f(x)$$

- Hence, we can work out t when $x = 2012$.

Definitions

Model

Our goal is to create a model.

- This is a function that can relate x to t.

$$t = f(x)$$

- Hence, we can work out t when $x = 2012$.

Data

We're going to create the model from data:

- N attribute-target pairs, (x_n, t_n)
- e.g. $(1896, 12s), (1900, 11s), \ldots, (2008, 9.69s)$
- $x_1 = 1896,\ t_1 = 12$, etc.

Definitions

Model

Our goal is to create a model.

- This is a function that can relate x to t.

$$t = f(x)$$

- Hence, we can work out t when $x = 2012$.

Data

We're going to create the model from data:

- N attribute-target pairs, (x_n, t_n)
- e.g. $(1896, 12s), (1900, 11s), \ldots, (2008, 9.69s)$
- $x_1 = 1896,\ t_1 = 12$, etc.

Often called **training** data

A linear model

$$t = f(x)$$

A linear model

$$t = f(x) = w_0 + w_1 x$$

model parameters

A linear model

$$t = f(x) = w_0 + w_1 x = f(x; w_0, w_1)$$

- w_0 and w_1 are *parameters* of the model.

A linear model

$$t = f(x) = w_0 + w_1 x = f(x; w_0, w_1)$$

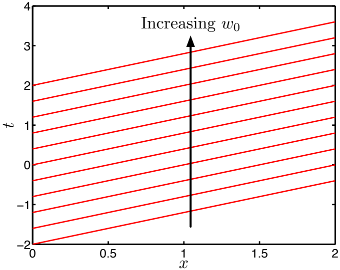
- w_0 and w_1 are *parameters* of the model.
- They determine the properties of the line.

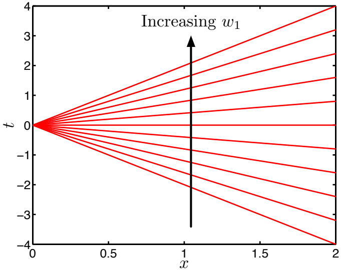
A linear model

linear model

$$t = f(x) = \underbrace{w_0 + w_1 x}_{} = f(x; w_0, w_1)$$

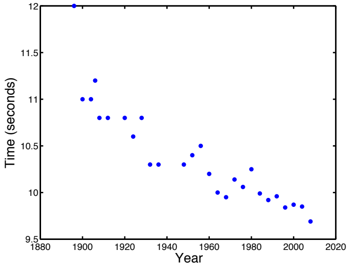
- w_0 and w_1 are *parameters* of the model.
- They determine the properties of the line.



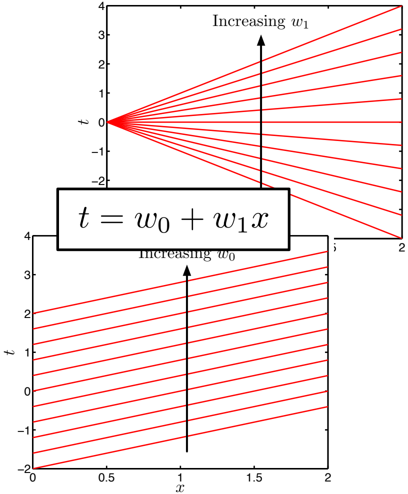


What next?

We have data and a family of models:

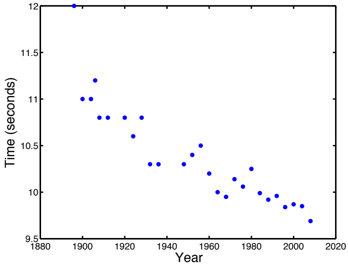


?

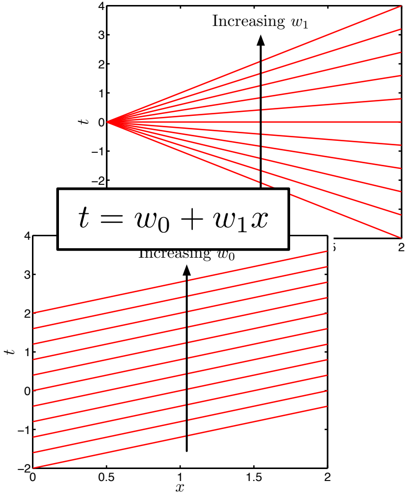


What next?

We have data and a family of models:



?



Need to find w_0, w_1 from $(x_1, t_1), (x_2, t_2), \ldots, (x_N, t_N)$

How good is a particular w_0, w_1?

- How good is a particular line (w_0, w_1)?

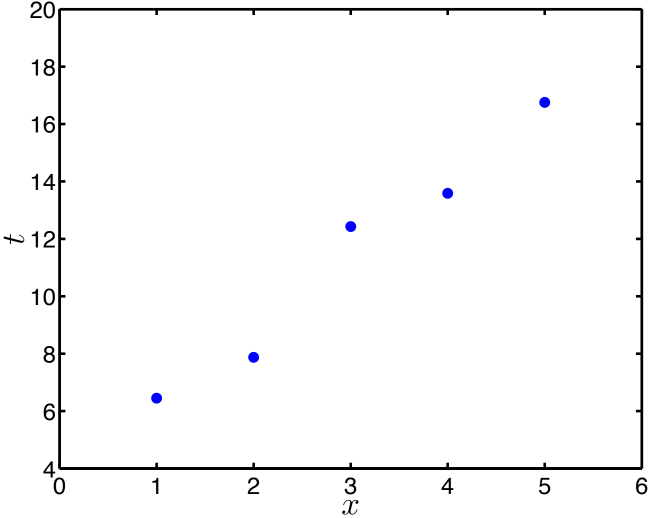
How good is a particular w_0, w_1?

- How good is a particular line (w_0, w_1)?
- We need to be able to provide a numerical value of goodness for any w_0, w_1.
  - How good is $w_0 = 5, w_1 = 0.1$?
  - Is $w_0 = 5, w_1 = -0.1$ better or worse?

How good is a particular w_0, w_1?

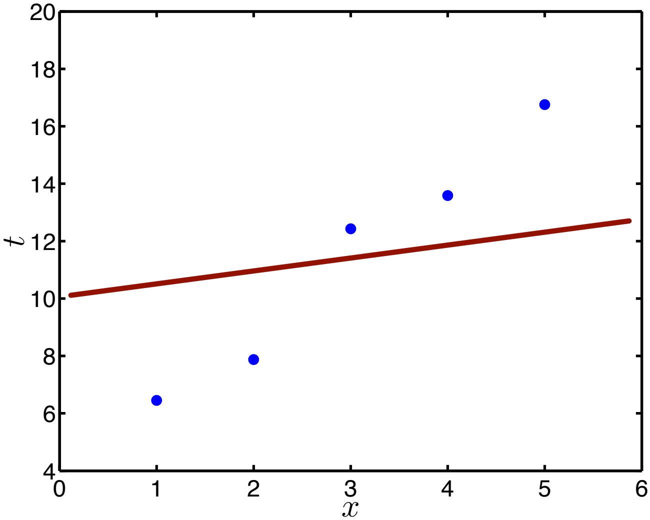
- How good is a particular line (w_0, w_1)?
- We need to be able to provide a numerical value of goodness for any w_0, w_1.
  - How good is $w_0 = 5, w_1 = 0.1$?
  - Is $w_0 = 5, w_1 = -0.1$ better or worse?
- Once we can answer these questions, we can search for the best w_0, w_1 pair.

Loss



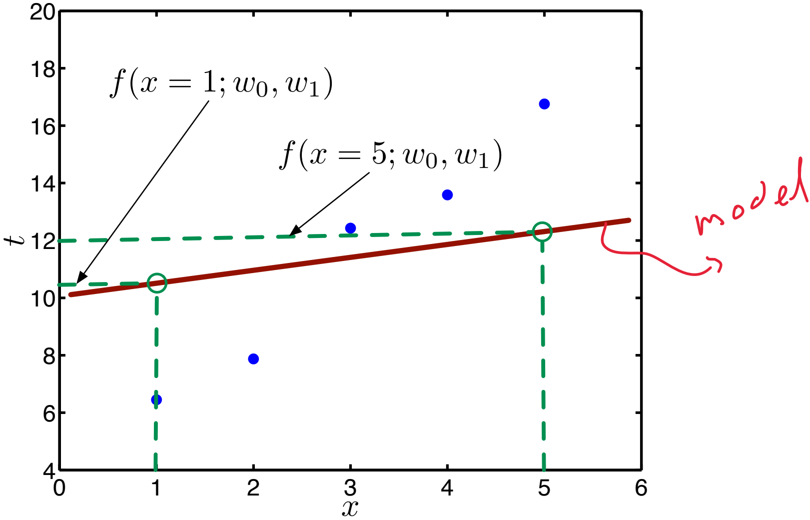
Some different data.

Loss



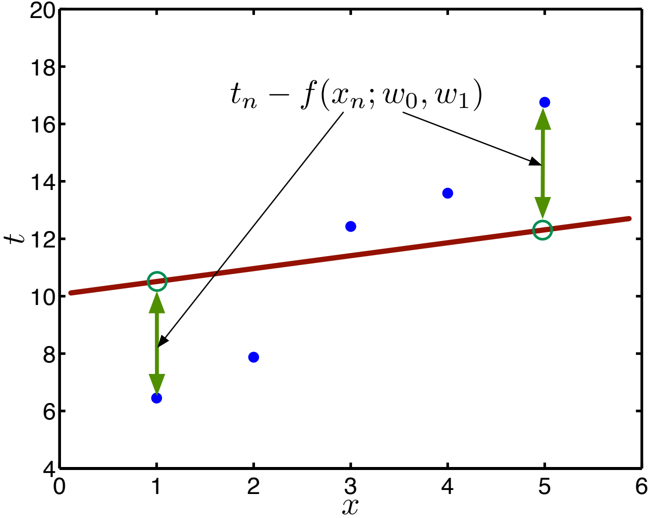
Given w_0 and w_1 you can draw a line.

Loss



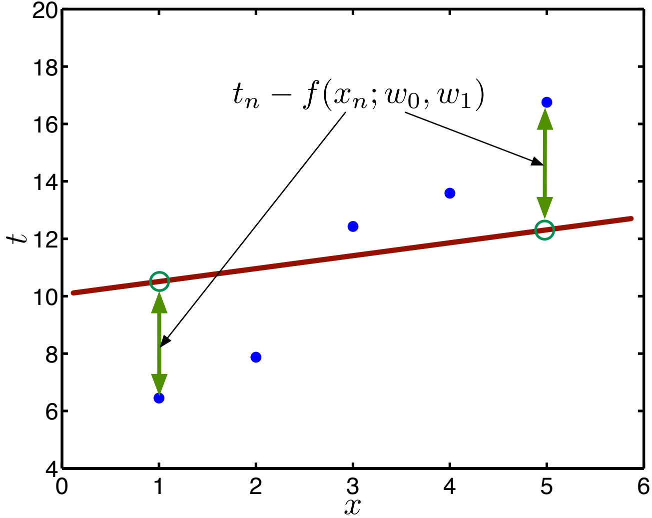
This means that we can compute $f(x_n; w_0, w_1)$ for each x_n.

Loss



$f(x_n; w_0, w_1)$ can be compared with the truth, t_n.

Loss



actual value

$f(x_n; w_0, w_1)$ can be compared with the truth, t_n.

$(t_n - \underbrace{f(x_n; w_0, w_1)}))^2$ tells us how *badly* we model (x_n, t_n).

→ model value

Squared loss

- The *Squared loss* of the n-th training point is defined as:

$$\mathcal{L}_n = (t_n - f(x_n; w_0; w_1))^2$$

Squared loss

- The *Squared loss* of the n-th training point is defined as:

$$\mathcal{L}_n = (t_n - f(x_n; w_0; w_1))^2$$

- It is the squared difference between the true response (winning time), t_n when the input is x_n and the response predicted by the model, $f(x_n; w_0, w_1) = w_0 + w_1 x_n$.

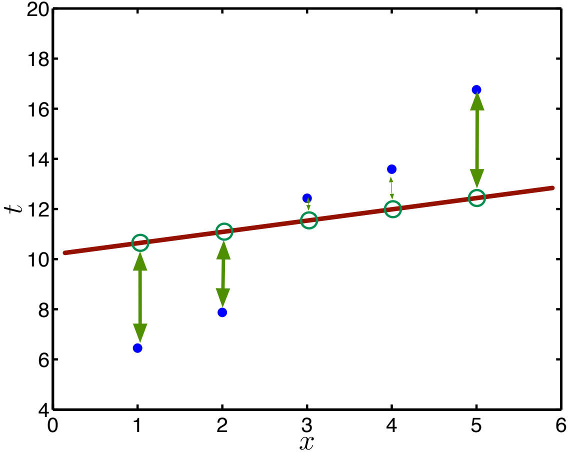
Squared loss

- The *Squared loss* of the n-th training point is defined as:

$$\mathcal{L}_n = (t_n - f(x_n; w_0; w_1))^2$$

- It is the squared difference between the true response (winning time), t_n when the input is x_n and the response predicted by the model, $f(x_n; w_0, w_1) = w_0 + w_1 x_n$.
- The lower $\mathcal{L}_n$, the closer the line at x_n passes to t_n.

Total squared loss



Average the loss at each training point to give single figure for all data:

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

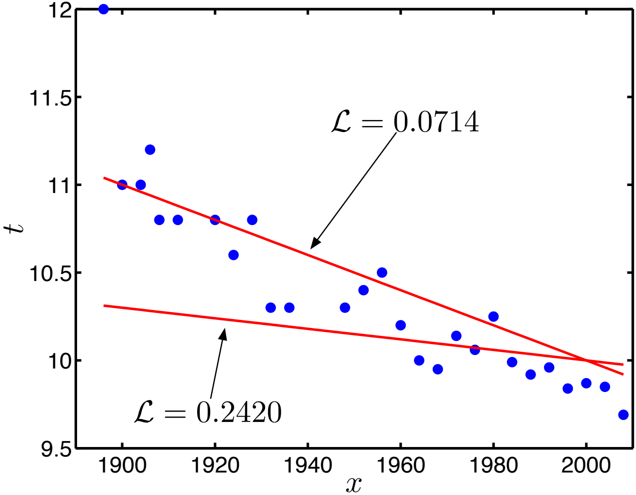
loss func.

- The average loss:

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

- $\mathcal{L}$ tells us how good the model is as a function of w_0 and w_1.
  - Remember that lower is better!
  - How good is $w_0 = 5, w_1 = 0.1$?
  - How good is $w_0 = 6, w_1 = -0.2$?
  - Which is better?

Example



$\mathcal{L} = 0.0714$
$\mathcal{L} = 0.2420$
12
11.5
11
10.5
10
9.5
1900
1920
1940
1960
1980
2000
t
x

An optimisation problem

- We've derived an expression for how good the model is for any w_0 and w_1.

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

- Could use trial and error to find a good w_0, w_1 combination.

An optimisation problem

- We've derived an expression for how good the model is for any w_0 and w_1.

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

- Could use trial and error to find a good w_0, w_1 combination.
- Can we get a mathematical expression?

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

Aside - finding maxima and minima

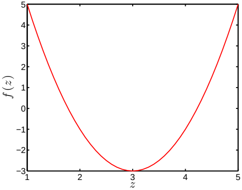
Say we want to find

$$\underset{z}{\operatorname{argmin}}\ f(z),\ \ f(z) = 2z^2 - 12z + 15.$$

Aside - finding maxima and minima

Say we want to find

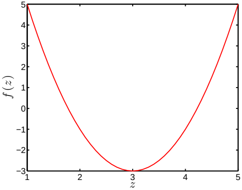
$$\underset{z}{\text{argmin}}\; f(z), \;\; f(z) = 2z^2 - 12z + 15.$$



Aside - finding maxima and minima

Say we want to find

$$\underset{z}{\operatorname{argmin}}\ f(z),\ \ f(z) = 2z^2 - 12z + 15.$$

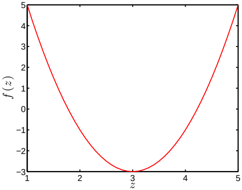


At a minimum (or a maximum), the gradient must be zero.

Aside - finding maxima and minima

Say we want to find

$$\underset{z}{\mathrm{argmin}}\ f(z),\ \ f(z) = 2z^2 - 12z + 15.$$



At a minimum (or a maximum), the gradient must be zero.

The gradient is given by the first derivative of the function:

$$\frac{df(z)}{dz} = 4z - 12$$

Setting to zero and solving for z

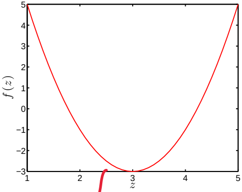
$$4z - 12 = 0,\ \ z = 12/4 = 3$$

Finding maxima and minima

- So, we know that the gradient is 0 at $z = 3$.
- How do we know if it is a minimum or a maximum?

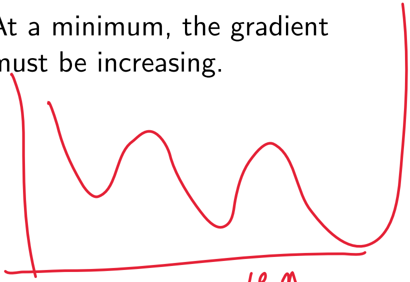
Finding maxima and minima

- So, we know that the gradient is 0 at $z = 3$.
- How do we know if it is a minimum or a maximum?



convex

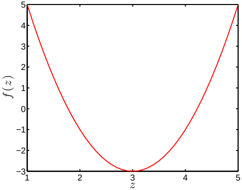
At a minimum, the gradient must be increasing.



non convex

Finding maxima and minima

- So, we know that the gradient is 0 at $z = 3$.
- How do we know if it is a minimum or a maximum?



At a minimum, the gradient must be increasing.

Taking the second derivative:

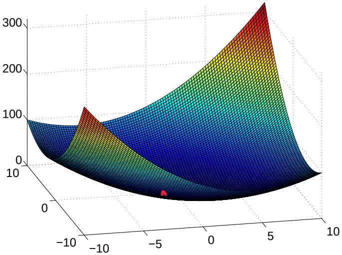
$$\frac{df(z)}{dz} = 4z - 12$$
$$\frac{d^2z}{dz^2} = 4$$

The gradient is always increasing. Therefore, we have found a minimum and it is the only minumum.

Finding maxima and minima

What about functions of more than one parameter?

$$\underset{y,z}{\operatorname{argmin}}\ f(y,z),\ \ f(y,z) = y^2 + z^2 + y + z + yz$$



We now use *partial derivatives*, $\frac{\partial f}{\partial z}$ and $\frac{\partial f}{\partial y}$

When calculating the partial derivative with respect to y we assume everything else (including z) is a constant.

$$\frac{\partial f}{\partial y} = 2y + 1 + z, \quad \frac{\partial f}{\partial z} = 2z + 1 + y$$

$$\frac{\partial f}{\partial y} = 2y + 1 + z = 0, \quad \frac{\partial f}{\partial z} = 2z + 1 + y = 0$$

To find a potential minimum, set both to zero and solve for y and z:

$$\begin{aligned} y &= -\frac{1}{3} \\ z &= -\frac{1}{3}. \end{aligned}$$

To make sure its a minimum, check second derivatives:

$$\frac{\partial^2 f}{\partial y^2} = 2, \quad \frac{\partial^2 f}{\partial z^2} = 2.$$

Both are positive so we have a minimum.

Back to our function

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

Now, recall that:

$$f(x_n; w_0, w_1) = w_0 + w_1 x$$

So:

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

Back to our function

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

Now, recall that:

$$f(x_n; w_0, w_1) = w_0 + w_1 x$$

So:

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

We need to find $\frac{\partial \mathcal{L}}{\partial w_0}$ and $\frac{\partial \mathcal{L}}{\partial w_1}$, and use thoese to find the *best* values!

Differentiating the loss

- Taking partial derivatives with respect to w_0 and w_1:

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

Finding w_0:

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

$$0 = -\frac{2}{N}\sum_{n=1}^{N}(t_n - w_0 - w_1 x_n)$$

$$w_0 \underbrace{\frac{1}{N}\sum_{n=1}^{N} 1}_{=1} = \qquad \frac{\cancel{2}}{N}\sum_{n=1}^{N} w_0 = \frac{\cancel{2}}{N}\sum_{n=1}^{N} t_n - \frac{\cancel{2}}{N}\sum_{n=1}^{N} w_1 x_n \qquad \frac{1}{N} w_1 \sum_{n=1}^{N} x_n$$

$$w_0 = \bar{t} - w_1 \bar{x}$$

Where

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

Finding w_1:

$$
\begin{aligned}
\frac{\partial \mathcal{L}}{\partial w_1} &= -\frac{2}{N}\sum_{n=1}^{N} x_n(t_n - w_0 - w_1 x_n) \\
0 &= -\frac{2}{N}\sum_{n=1}^{N} x_n(t_n - w_0 - w_1 x_n) \\
w_1 \frac{1}{N}\sum_{n=1}^{N} x_n^2 &= \frac{1}{N}\sum_{n=1}^{N} x_n t_n - w_0 \frac{1}{N}\sum_{n=1}^{N} x_n \\
w_1 \overline{x^2} &= \overline{xt} - w_0 \overline{x}
\end{aligned}
$$

Where

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

Substituting:

Substituting our expression for w_0 into that for w_1:

$$
\begin{aligned}
w_0 &= \bar{t} - w_1\bar{x} \\
w_1\overline{x^2} &= \overline{xt} - w_0\bar{x} \\
w_1\overline{x^2} &= \overline{xt} - \bar{x}(\bar{t} - w_1\bar{x}) \\
w_1 &= \frac{\overline{xt} - \bar{x}\bar{t}}{\overline{x^2} - (\bar{x})^2}
\end{aligned}
$$

$\bar{t} - w_0\bar{x}$

So, to summarise:

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

Note that $\overline{xt} \neq \bar{x}\bar{t}$ and $\overline{x^2} \neq (\bar{x})^2$.

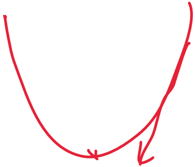
Gradient Descent: an alternative approach

numerical optimization

Repeatedly move in the direction of the gradient using *step size* η:

$$w_0 \leftarrow w_0 - \eta \frac{\partial \mathcal{L}}{\partial w_0}$$

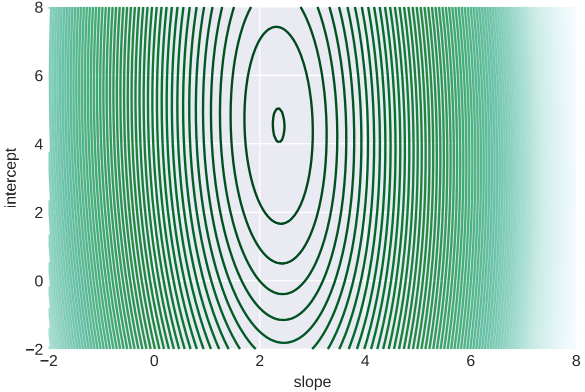
$$w_1 \leftarrow w_1 - \eta \frac{\partial \mathcal{L}}{\partial w_1}$$



For *convex* functions, this is guaranteed to *converge* to the *global optimum*.

There are many *accelerated* variations to speed up convergence.

Searching for the best parameters

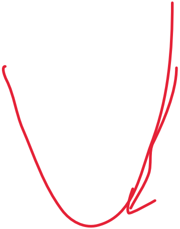


“Climbing down” formally: gradient descent

1. define a “learning rate” η
2. initialize the parameters w_0, w_1 (slope and intercept)
3. compute the gradients (steepest direction)
4. update the parameters as

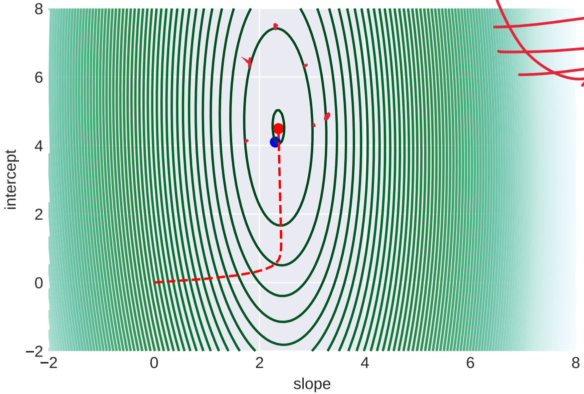
$$w_0 \leftarrow w_0 - \eta \frac{\partial \mathcal{L}}{\partial w_0}$$

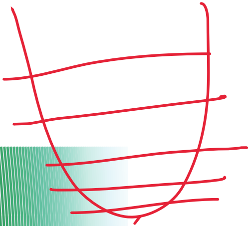
$$w_1 \leftarrow w_1 - \eta \frac{\partial \mathcal{L}}{\partial w_1}$$



5. is the gradient close to zero? if no, go back to 3

gradient descent example





w_0

w_1

Olympic data

n	x_n	t_n	$x_n t_n$	x_n^2
1	1896	12.00	22752.0	3.5948e+06
2	1900	11.00	20900.0	3.6100e+06
3	1904	11.00	20944.0	3.6252e+06
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
26	2004	9.85	19739.4	4.0160e+06
27	2008	9.69	19457.5	4.0321e+06
$(1/N)\sum_{n=1}^{N}$	1952.37	10.39	20268.1	3.8130e+06
	$\overline{x}$	$\overline{t}$	$\overline{xt}$	$\overline{x^2}$

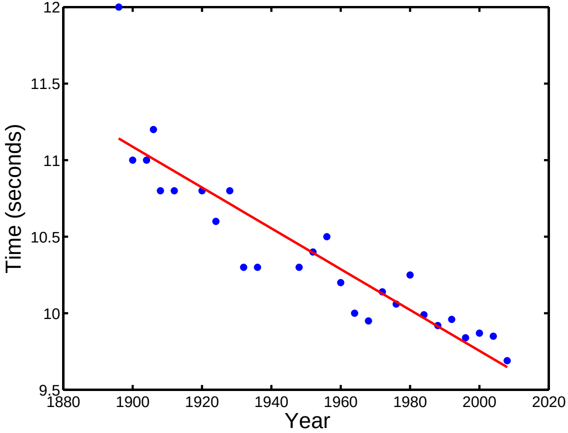
Olympic data

n	x_n	t_n	$x_n t_n$	x_n^2
1	1896	12.00	22752.0	3.5948e+06
2	1900	11.00	20900.0	3.6100e+06
3	1904	11.00	20944.0	3.6252e+06
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
26	2004	9.85	19739.4	4.0160e+06
27	2008	9.69	19457.5	4.0321e+06
$(1/N)\sum_{n=1}^{N}$	1952.37	10.39	20268.1	3.8130e+06
	$\overline{x}$	$\overline{t}$	$\overline{xt}$	$\overline{x^2}$

Substituting these values into our expressions gives:

$$w_1 = -0.0133, \ \ w_0 = 36.416$$

The model



$$t = 36.416 - 0.0133x \rightarrow 201$$

w_0 w_1

Our prediction

- We want to predict the winning time at London 2012.
- Substitute $x = 2012$ into our model.

$$\begin{aligned} t &= 36.416 - 0.0133x \\ t_{2012} &= 36.416 - 0.0133 \times 2012 \\ t_{2012} &= 9.5947\ s \end{aligned}$$

- Based on our modelling assumptions and the previous data, we predict a winning time of 9.5947 seconds.

Assumptions

Our Assumptions

1. **That there exists a relationship between Olympic year and winning time.**

Are they any good?

1. Is the relationship really between Olympic year and time?

Assumptions

Our Assumptions

1. That there exists a relationship between Olympic year and winning time.
2. **That this relationship is linear (i.e. a straight line).**

Are they any good?

1. Is the relationship really between Olympic year and time?
2. Seems a bit simple? Does the line go through all of the points?

Assumptions

Our Assumptions

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

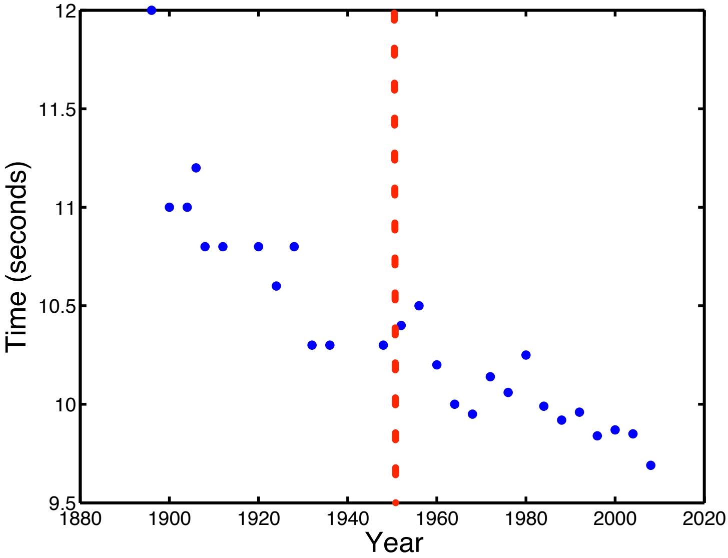
Are they any good?

1. Is the relationship really between Olympic year and time?
2. Seems a bit simple? Does the line go through all of the points?
3. Forever? Negative winning times?

Some things to think about

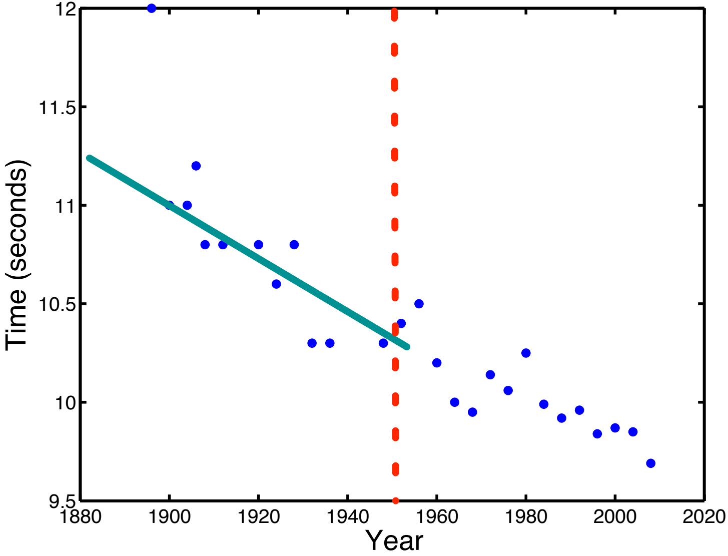
- Is this a good prediction?
- Would you go to the bookmakers and place a bet on the winning time being exactly 9.547 s?
- Are we asking the correct question? Being too precise?

A question we could have answered in 1950



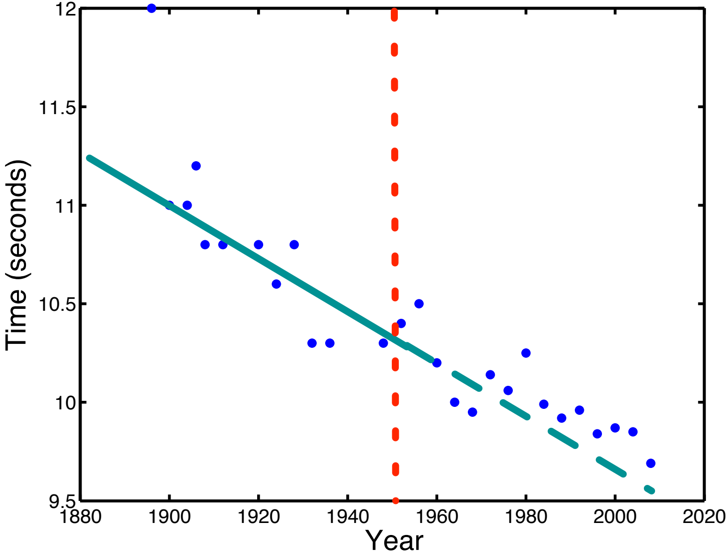
Will the winning time be sub 10s in 2000?

A question we could have answered in 1950



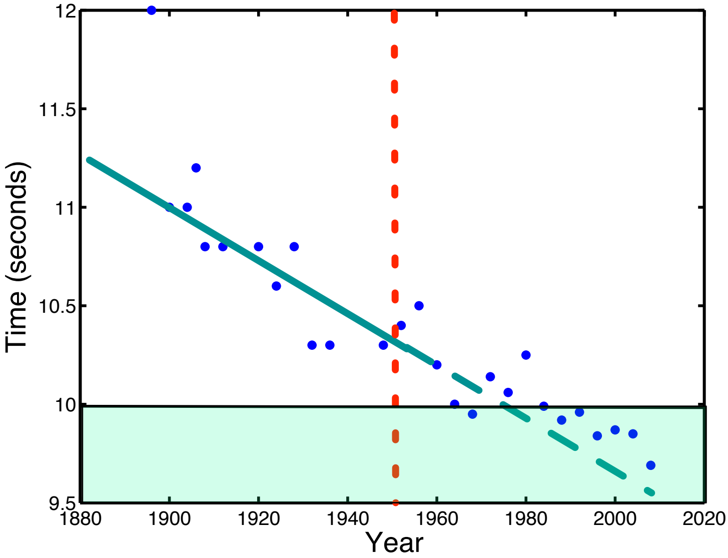
Will the winning time be sub 10s in 2000?

A question we could have answered in 1950



Will the winning time be sub 10s in 2000?

A question we could have answered in 1950



Will the winning time be sub 10s in 2000?

Regression in statistics and machine learning

- regression models are among the most widely used tools in statistics
- but regression is also an important problem in machine learning
- difference in emphasis:
  - in statistics, the purpose is often *explanation*: "how does x affect t?" "is x important for t?"
  - in machine learning, the purpose is typically *prediction*: "what's the most likely t, given x?"

Multivariate Data

- Olympic winning time may depend also on weather, track conditions etc.
- Each data point is thus represented by a *vector* of dimension D of *features* or *attributes*, $\mathbf{x}$.
- Our problem thus is to find a function $t = f(\mathbf{x})$.
- *Multi-linear* function:

$$t = f(x, w_0, w_1, \cdots, w_D) := w_0 + w_1 x_1 + \cdots + w_D x_D.$$

Squared loss

- The squared loss of the *n*-th training point is:

$$\mathcal{L}_n = (t_n - f(\mathbf{x}_n; w_0; w_1 \cdots, w_D))^2$$

- The *averaged squared loss* is:

$$\mathcal{L} = \frac{1}{N}\sum_{n=1}^{N}(t_n - f(\mathbf{x}_n; w_0, w_1, \cdots, w_D))^2$$

Squared loss

$x_n = \begin{bmatrix} 1 \\ x_{1,n} \\ x_{2,n} \\ \vdots \\ x_{D,n} \end{bmatrix}, \quad w = \begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ \vdots \\ w_D \end{bmatrix}$

- The *averaged squared loss* is:

$$\mathcal{L} = \frac{1}{N}\sum_{n=1}^{N}(t_n - f(\mathbf{x}_n; w_0, w_1, \cdots, w_D))^2$$

$w^T x_n = 1 \times w_0 + w_1 x_{1,n} + \ldots + w_D x_{D,n}$

- Then

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

Note that: (we append 1 to the begining of $\mathbf{x}_n$)

$$\mathbf{x}_n \leftarrow [1 \quad \mathbf{x}_n]$$

- Therefore

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

Recipe

- Put data and parameters into vectors/matrix.
- Write the model in vector form.
- Write the loss in vector/matrix form.

Recipe

- Put data and parameters into vectors/matrix.
- Write the model in vector form.
- Write the loss in vector/matrix form.

Why?

More features: $t = w_0 + w_1 x_1 + \cdots + w_D x_D$
More complex models: $t = w_0 + w_1 x + w_2 x^2 + \ldots + w_D x^D$

Recipe

- Put data and parameters into vectors/matrix.
- Write the model in vector form.
- Write the loss in vector/matrix form.

Why?

More features: $t = w_0 + w_1 x_1 + \cdots + w_D x_D$

More complex models: $t = w_0 + w_1 x + w_2 x^2 + \ldots + w_D x^D$

$$\mathbf{x}_n = \begin{bmatrix} 1 \\ x_{n,1} \\ x_{n,2} \\ \vdots \\ x_{n,D} \end{bmatrix},$$

Recipe

- Put data and parameters into vectors/matrix.
- Write the model in vector form.
- Write the loss in vector/matrix form.

$\tilde{w}^T x_n = x_n^T w$

Xw

a vector of $N \times 1$

Why?

More features: $t = w_0 + w_1 x_1 + \cdots + w_D x_D$

More complex models: $t = w_0 + w_1 x + w_2 x^2 + \ldots + w_D x^D$

$$\mathbf{x}_n = \begin{bmatrix} 1 \\ x_{n,1} \\ x_{n,2} \\ \vdots \\ x_{n,D} \end{bmatrix}, \mathbf{X} = \begin{bmatrix} 1 & x_{1,1} & x_{1,2} & \ldots & x_{1,D} \\ 1 & x_{2,1} & x_{2,2} & \ldots & x_{2,D} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{N,1} & x_{N,2} & \ldots & x_{N,D} \end{bmatrix}, \mathbf{t} = \begin{bmatrix} t_1 \\ t_2 \\ \vdots \\ t_N \end{bmatrix}$$

Recipe

- Put data and parameters into vectors/matrix.
- Write the model in vector form.
- Write the loss in vector/matrix form.

Why?

More features: $t = w_0 + w_1 x_1 + \cdots + w_D x_D$
More complex models: $t = w_0 + w_1 x + w_2 x^2 + \ldots + w_D x^D$

$$\mathbf{x}_n = \begin{bmatrix} 1 \\ x_{n,1} \\ x_{n,2} \\ \vdots \\ x_{n,D} \end{bmatrix}, \mathbf{X} = \begin{bmatrix} 1 & x_{1,1} & x_{1,2} & \ldots & x_{1,D} \\ 1 & x_{2,1} & x_{2,2} & \ldots & x_{2,D} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{N,1} & x_{N,2} & \ldots & x_{N,D} \end{bmatrix}, \mathbf{t} = \begin{bmatrix} t_1 \\ t_2 \\ \vdots \\ t_N \end{bmatrix}$$

$$\mathcal{L} = \frac{1}{N}(\mathbf{t} - \mathbf{X}\mathbf{w})^\mathsf{T}(\mathbf{t} - \mathbf{X}\mathbf{w}), \texttt{ where } \mathbf{w} = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_D \end{bmatrix}.$$

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:

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

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:

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

Useful identites:

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

$w_0 + w_1 x_1 + \cdots + w_D x_D$

$x = \begin{bmatrix} 1 \\ x_1 \\ \vdots \\ x_D \end{bmatrix}$

$\begin{bmatrix} 1 \\ x_1 \\ \vdots \\ x_D \end{bmatrix}$

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

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

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

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

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

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

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

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

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

$$
\begin{aligned}
\mathbf{X}^{\mathsf{T}}\mathbf{X}\mathbf{w} &= \mathbf{X}^{\mathsf{T}}\mathbf{t} \\
\underbrace{\underbrace{(\mathbf{X}^{\mathsf{T}}\mathbf{X})^{-1}}_{A^{-1}}\underbrace{\mathbf{X}^{\mathsf{T}}\mathbf{X}}_{A}}_{I w = w}\mathbf{w} &= (\mathbf{X}^{\mathsf{T}}\mathbf{X})^{-1}\mathbf{X}^{\mathsf{T}}\mathbf{t}
\end{aligned}
$$

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

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

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,

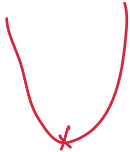
$$\mathbf{I} = \begin{bmatrix} 1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 1 \end{bmatrix}, \text{ and } \mathbf{I}\mathbf{A} = \mathbf{A}, \text{ for any } \mathbf{A}$$

An alternative optimization: Gradient Descent

Repeatedly move in the direction of the gradient for w using *step size* η:

$$\mathbf{w} \leftarrow \mathbf{w} - \eta \frac{\partial \mathcal{L}}{\partial \mathbf{w}}$$

For *convex* functions, this is guaranteed to *converge* to the *global optimum*.
There are many *accelerated* variations to speed up convergence.



Linear model - Olympic data

$t = w_0 + w_1 x$

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

Linear model - Olympic data

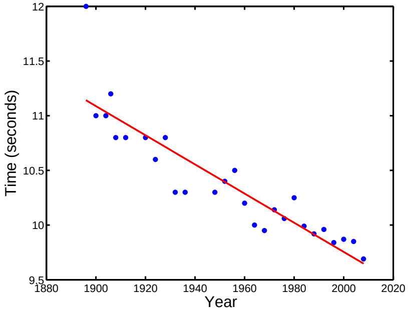
$$\mathbf{w} = \begin{bmatrix} w_0 \\ w_1 \end{bmatrix}, \ \mathbf{X} = \begin{bmatrix} 1 & 1896 \\ 1 & 1900 \\ \vdots & \\ 1 & 2008 \end{bmatrix}, \ \mathbf{t} = \begin{bmatrix} 12.00 \\ 11.00 \\ \vdots \\ 9.85 \end{bmatrix}$$

$$\widehat{\mathbf{w}} = (\mathbf{X}^\mathsf{T}\mathbf{X})^{-1}\mathbf{X}^\mathsf{T}\mathbf{t} = \begin{bmatrix} 36.416 \\ -0.0133 \end{bmatrix} \begin{array}{l} \to \hat{w}_0 \\ \to \hat{w}_1 \end{array}$$

Linear model - Olympic data

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

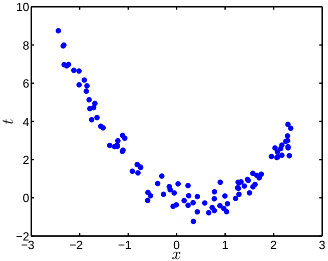
$$\widehat{\mathbf{w}} = (\mathbf{X}^\mathsf{T}\mathbf{X})^{-1}\mathbf{X}^\mathsf{T}\mathbf{t} = \begin{bmatrix} 36.416 \\ -0.0133 \end{bmatrix}$$



Quadratic model - synthetic data

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

$\cancel{t = w_0 + w_1 x}$ $\rightarrow$ $t = w_0 + w_1 x + w_2 x^2$

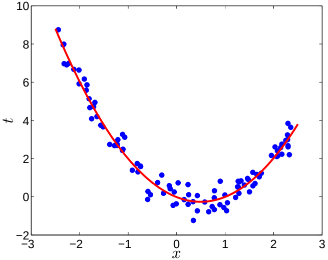


Quadratic model - synthetic data

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

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

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



8th order model - Olympic data

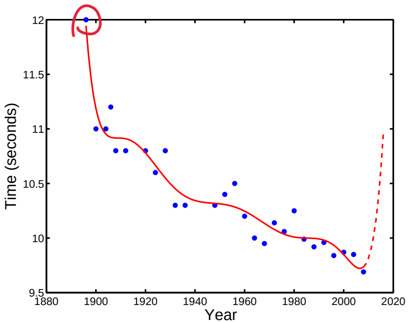
$$t = w_0 + w_1 x + w_2 x^2 + \ldots + w_8 x^8$$

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

8th order model - Olympic data

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

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



More general models

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

$$t = w_0 + w_1 x + w_2 x^2 + \ldots + w_D x^D$$

- In fact, each term can be any function of x (or even $\mathbf{x}$)

$$t = w_0 \underline{h_0(x)} + w_1 \underline{h_1(x)} + \ldots + w_D \underline{h_D(x)}$$

- 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

$$t = w_0 + w_1 x + w_2 x^2 + \ldots + w_D x^D$$

- In fact, each term can be any function of x (or even $\mathbf{x}$)

$$t = w_0 h_0(x) + w_1 h_1(x) + \ldots + w_D h_D(x)$$

- For example,

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

- In General:

$\cos(x)$
$|x_{n,1} - x_{n,2}|$

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

Example – Olympic data

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

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

$h_0 = 1$

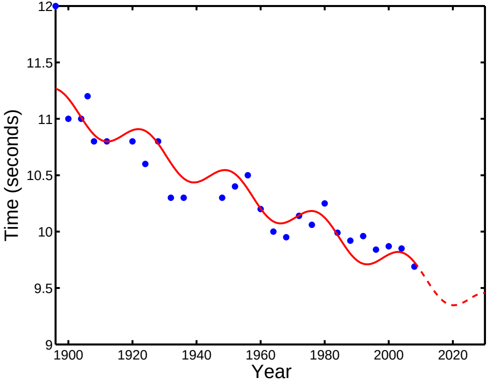
$h_1 = x$

$h_2 = \sin\left(\frac{x-a}{b}\right)$

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^D - can have any function of x (or even $\mathbf{x}$).
- 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:

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

$$t = h_0(x)w_0 + h_1(x)w_1 + \dots + h_D(x)w_D$$

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:

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

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

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

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:

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

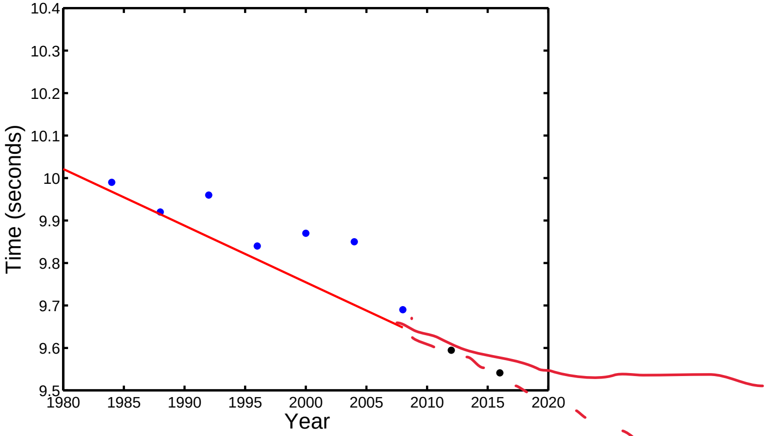
To predict t at a new value of x, we first create $\mathbf{x}_{\text{new}}$:

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

and then compute

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

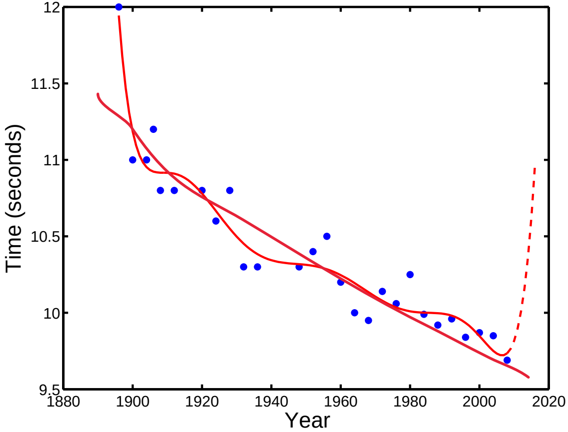
Example - Olympic data



Linear model – predictions OK?

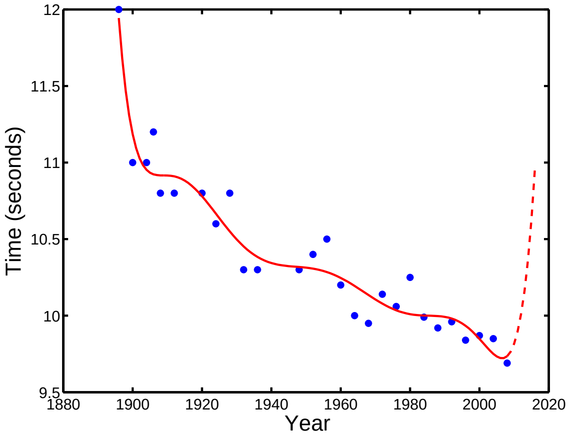
$w_0 + w_1 x$

Example - Olympic data



8th order model – predictions terrible!

Example - Olympic data



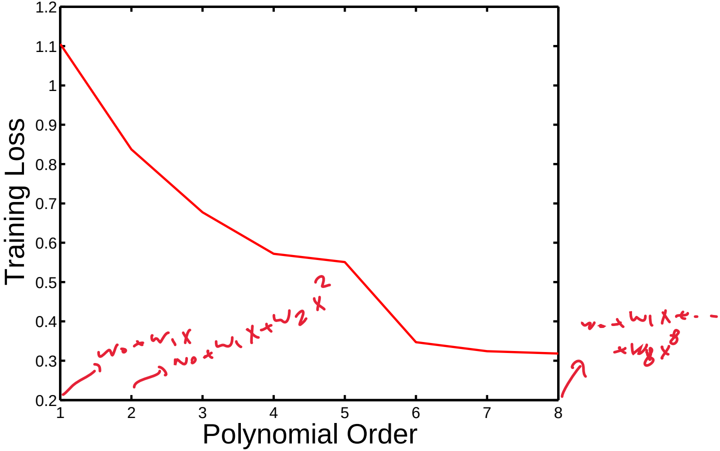
8th order model – predictions terrible!

Choice of model is **very** important.

Possible ways of choosing

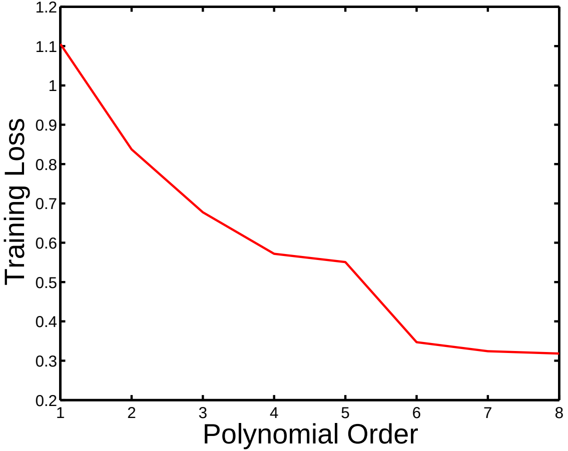
- Lowest loss, $\mathcal{L}$?

How does loss change?



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

How does loss change?

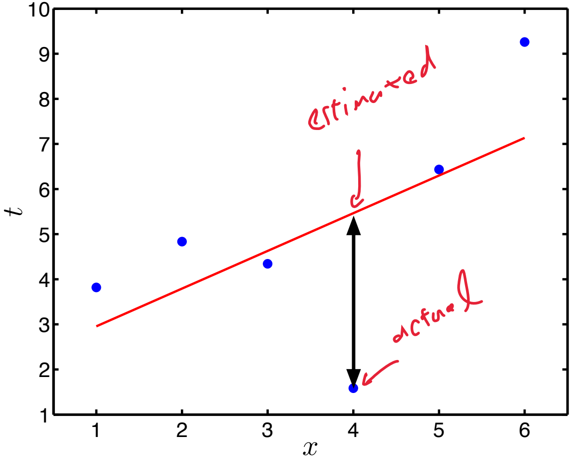


Loss, L, on the Olympic 100m data as additional terms (x^D) are added to the model.

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

Loss always decreases with model complexity

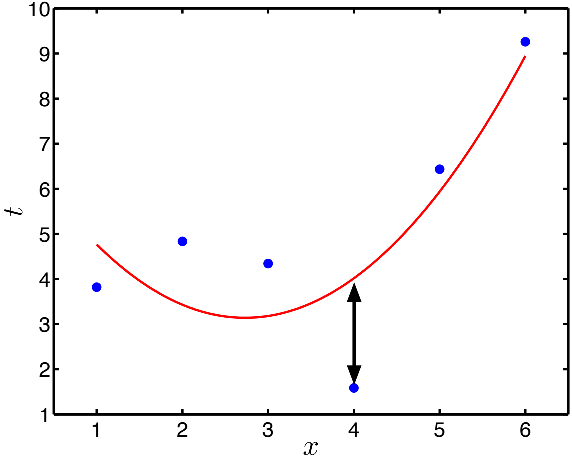
Data comes from $t = x$ with some *noise* added:



Linear model $t = w_0 + w_1 x$.

Loss always decreases with model complexity

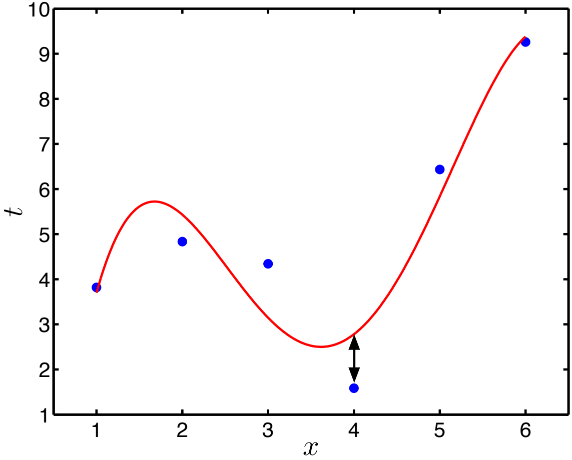
Data comes from $t = x$ with some *noise* added:



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

Loss always decreases with model complexity

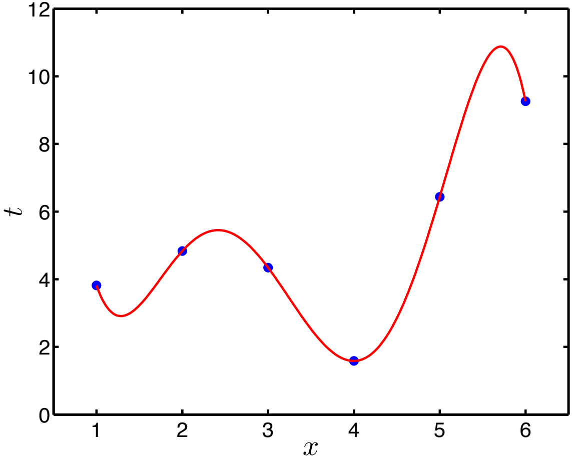
Data comes from $t = x$ with some *noise* added:



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

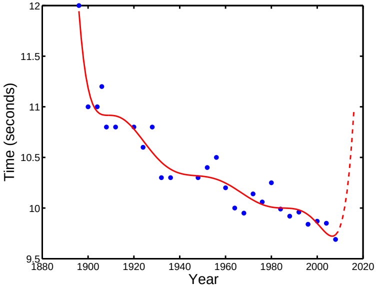
There is a trade-off between generalisation (predictive ability) and over-fitting (decreasing the loss).

- 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 $N - M$ pairs to find $\widehat{\mathbf{w}}$ for several models.

new training data

training data

validation data

unseen by the model

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

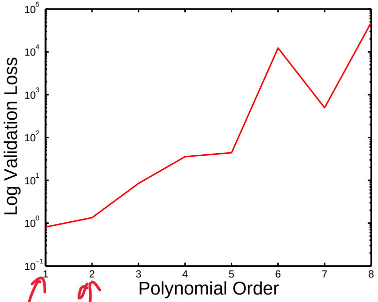
Where can we get more data?

- We have N input-response pairs for training:

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

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

Validation example



$w_0 + w_1 x$

$w_0 + w_1 x + w_2 x^2$

...

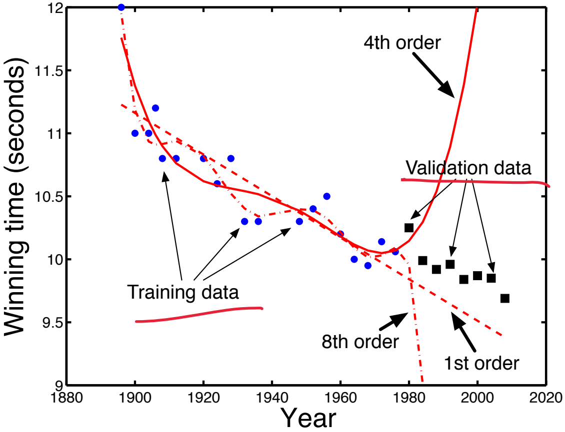
Predictions evaluated using validation loss:

$$\boxed{\mathcal{L}_v = \frac{1}{M}\sum_{m=1}^{M}(t_m - \mathbf{w}^\mathsf{T}\mathbf{x}_m)^2}$$

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.

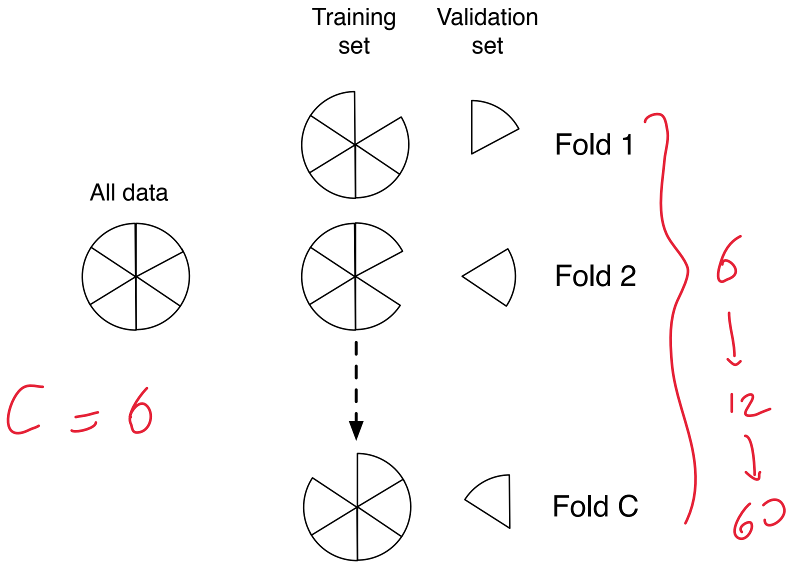
How should we choose which data to hold back?

- 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’.

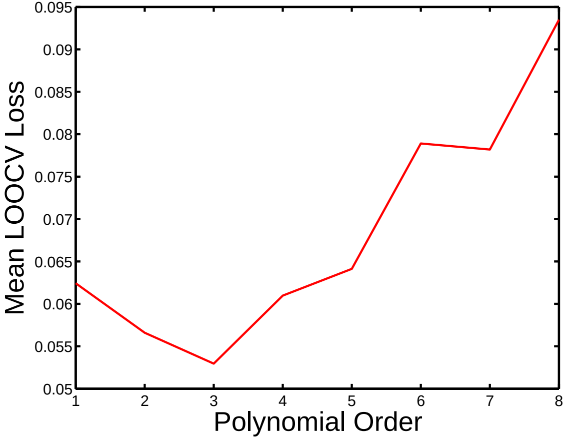
Leave-one-out Cross-validation

- 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

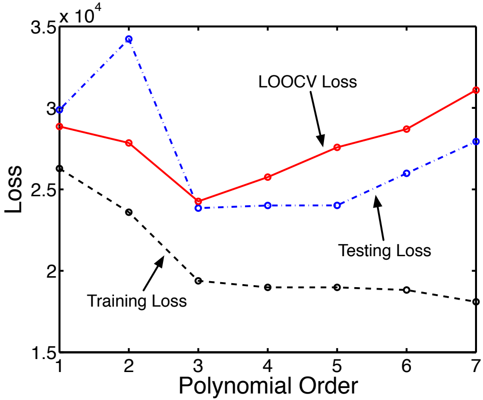
$$t = w_0 + w_1 x + w_2 x^2 + w_3 x^3.$$

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 our 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 our model N times.
- For $t = \mathbf{w}^\mathsf{T}\mathbf{x}$, this is feasible if D (number of terms in function) isn't too big:

$$
\begin{aligned}
t &= \sum_{d=0}^{D} w_d h_d(x) \\
\widehat{\mathbf{w}} &= \underbrace{(\mathbf{X}^\mathsf{T}\mathbf{X})^{-1}}_{(D+1) \times (D+1)}\mathbf{X}^\mathsf{T}\mathbf{t}
\end{aligned}
$$

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 our model N times.
- For $t = \mathbf{w}^\mathsf{T}\mathbf{x}$, this is feasible if D (number of terms in function) isn't too big:

$$
\begin{aligned}
t &= \sum_{d=0}^{D} w_d h_d(x) \\
\widehat{\mathbf{w}} &= (\mathbf{X}^\mathsf{T}\mathbf{X})^{-1}\mathbf{X}^\mathsf{T}\mathbf{t}
\end{aligned}
$$

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

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.