ELEC2146

Electrical Engineering Modelling and Simulation

Least Squares

Dr Julien Epps

S2, 2009

ELEC2146

Electrical Engineering Modelling and Simulation

Least Squares

Dr Ray Eaton

S2, 2016

Lecture notes written by Dr Julien Epps

Overview

- Motivation
- Error minimisation
- Polynomial form for LS
- General form for LS
- Linearization
- Multivariate LS
- Nonlinear LS
- Gradient descent

Motivation

- Consider:

  $y = c_1 x$

x	$y = f(x)$
-1	-3
2	6
3	9

- An exact solution exists: $c_1 = 3$

- What about:

x	$y = f(x)$
-1	-4
2	3
3	7

- No exact solution exists
  - Overdetermined
  - What kind of approximate solution should we look for ?

Motivation

- Common problem in modelling:
  - Have data from an experiment
    - Maybe inputs and outputs at different times, or for different conditions
    - Often have a reasonable amount of data $\rightarrow$ overdetermined equations
  - Have some hypothesis about the model structure
    - e.g. polynomial, sum of sinusoids, mixture of Gaussians etc
  - Want to use the data and assumed model structure to estimate the model parameters
  - More later on…
    - Model structure
    - Parameter estimation

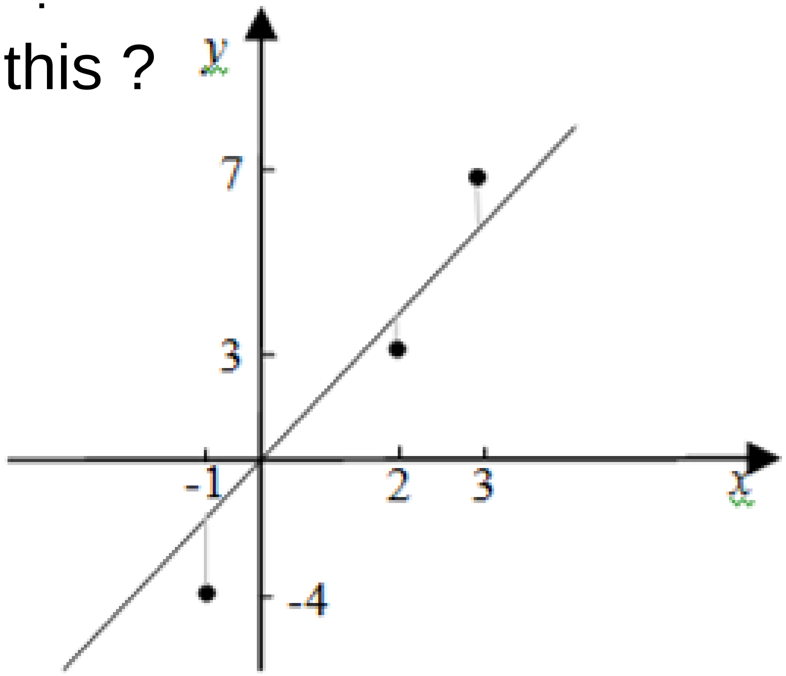
Error Minimisation

- Look for a solution to

$$y = c_1 x + \varepsilon$$

  - ε : error
  - "Smaller" $\varepsilon \Rightarrow$ better solution
    - But what do we mean by "small" ?
  - Also, how do we minimise this ?
  - From our example,

$$\boldsymbol{\varepsilon} = \begin{bmatrix} -4 \\ 3 \\ 7 \end{bmatrix} - c_1 \begin{bmatrix} -1 \\ 2 \\ 3 \end{bmatrix}$$



Error Minimisation

- Could try to minimise

$$|\boldsymbol{\varepsilon}| = \sum_{i=1}^{3} |\varepsilon_i| = \sum_{i=1}^{3} |y_i - c_1 x_i|$$

  - Sum of absolute errors

  - Introduces a nonlinearity in ε
    - Not very convenient
    - Piecewise linear

Error Minimisation

- Could try to minimise $\|\boldsymbol{\varepsilon}\|$ or $\|\boldsymbol{\varepsilon}\|^2$

$$\|\boldsymbol{\varepsilon}\|^2 = \sum_{i=1}^{3} \varepsilon_i^2 = \sum_{i=1}^{3} (y_i - c_1 x_i)^2 = (-4 + c_1)^2 + (3 - 2c_1)^2 + (7 - 3c_1)^2$$

  – Sum of squared errors
    - Larger errors are weighted more heavily than small ones

  – Conveniently, $\|\boldsymbol{\varepsilon}\|^2$ is a quadratic function of c_1, with a unique minimum

Error Minimisation

- Let's find the minimum:

$$\frac{d\|\boldsymbol{\varepsilon}\|^2}{dc_1} = 0$$

$$\frac{d\|\boldsymbol{\varepsilon}\|^2}{dc_1} = -8 + 2c_1 - 12 + 8c_1 - 42 + 18c_1$$

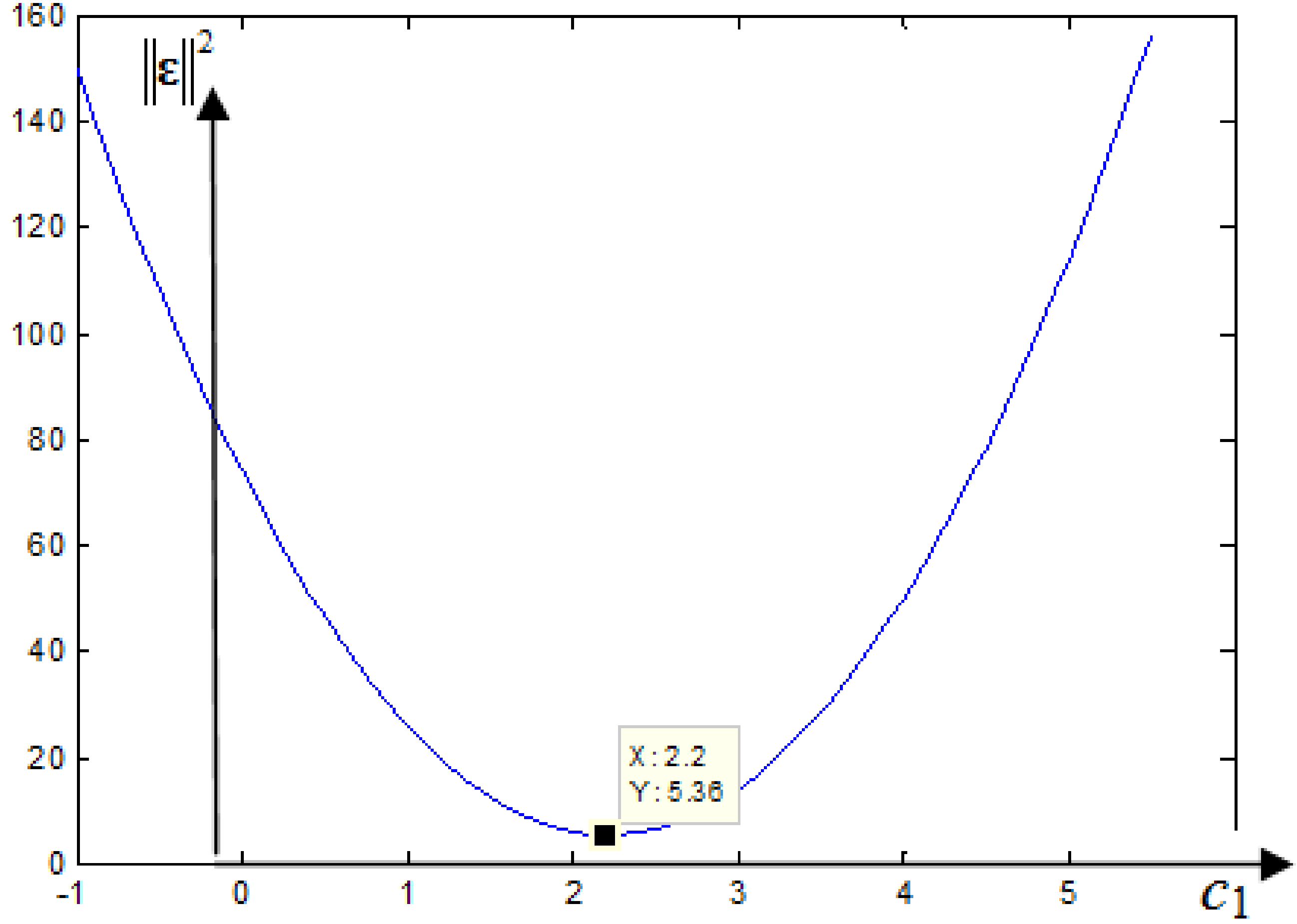
$$= -62 + 28c_1 = 0$$

$$c_1 = \frac{31}{14}$$

  - Error at this point is $\|\boldsymbol{\varepsilon}\|^2 = 5.357$

Error Minimisation

- Graphically:



LS: Polynomial Form

- Consider fitting:

$$y = c_n x^n + c_{n-1} x^{n-1} + \ldots + c_0$$

  - To some data $\{x_1, x_2, \ldots, x_m\}, \{y_1, y_2, \ldots, y_m\}$

$$y_1 = c_n x_1^n + c_{n-1} x_1^{n-1} + \ldots + c_0 + \varepsilon_1$$
$$y_2 = c_n x_2^n + c_{n-1} x_2^{n-1} + \ldots + c_0 + \varepsilon_2$$
$$\vdots$$
$$y_m = c_n x_m^n + c_{n-1} x_m^{n-1} + \ldots + c_0 + \varepsilon_m$$

LS: Polynomial Form

- Matrix-Vector form:

$$\mathbf{y} = \mathbf{X}\mathbf{c} + \boldsymbol{\varepsilon}$$

$$\mathbf{y} = \begin{bmatrix} y_1 & y_2 & \cdots & y_m \end{bmatrix}^T, \quad \mathbf{X} = \begin{bmatrix} x_1^n & x_1^{n-1} & \dots & 1 \\ x_2^n & x_2^{n-1} & \dots & 1 \\ \vdots & \vdots & \ddots & \vdots \\ x_m^n & x_m^{n-1} & \dots & 1 \end{bmatrix},$$

$$\mathbf{c} = \begin{bmatrix} c_n & c_{n-1} & \cdots & c_0 \end{bmatrix}^T, \quad \boldsymbol{\varepsilon} = \begin{bmatrix} \varepsilon_1 & \varepsilon_2 & \cdots & \varepsilon_m \end{bmatrix}^T$$

  – $\mathbf{y} = \mathbf{X}\mathbf{c}$ is a model for the data, **c** are the parameters and $\boldsymbol{\varepsilon}$ are the model errors

LS: Polynomial Form

- Rearranging:

$$\boldsymbol{\varepsilon} = \mathbf{y} - \mathbf{X}\mathbf{c},$$

- In matrix-vector form,

$$\|\boldsymbol{\varepsilon}\|^2 = \boldsymbol{\varepsilon}^T \boldsymbol{\varepsilon}$$

  – so

$$\begin{aligned}\|\boldsymbol{\varepsilon}\|^2 &= \boldsymbol{\varepsilon}^T \boldsymbol{\varepsilon} = (\mathbf{y} - \mathbf{X}\mathbf{c})^T (\mathbf{y} - \mathbf{X}\mathbf{c}) \\ &= \mathbf{y}^T \mathbf{y} - (\mathbf{X}\mathbf{c})^T \mathbf{y} - \mathbf{y}^T \mathbf{X}\mathbf{c} + (\mathbf{X}\mathbf{c})^T (\mathbf{X}\mathbf{c})\end{aligned}$$

LS: Polynomial Form

- Derivative wrt c_n:

$$\frac{\partial \|\boldsymbol{\varepsilon}\|^2}{\partial c_n} = -\begin{bmatrix} x_1^n & x_2^n & \dots & x_m^n \end{bmatrix}\mathbf{y} - \mathbf{y}^T \begin{bmatrix} x_1^n \\ x_2^n \\ \vdots \\ x_m^n \end{bmatrix} + 2\sum_{i=1}^{m} (x_i^n)^2 c_n$$

$$= -2\begin{bmatrix} x_1^n & x_2^n & \dots & x_m^n \end{bmatrix}\mathbf{y} + 2\begin{bmatrix} x_1^n & x_2^n & \dots & x_m^n \end{bmatrix} \begin{bmatrix} x_1^n \\ x_2^n \\ \vdots \\ x_m^n \end{bmatrix} c_n$$

LS: Polynomial Form

- Set derivative of error vector to zero:

$$\begin{bmatrix} \dfrac{\partial\|\boldsymbol{\varepsilon}\|^2}{\partial c_n} \\ \dfrac{\partial\|\boldsymbol{\varepsilon}\|^2}{\partial c_{n-1}} \\ \vdots \\ \dfrac{\partial\|\boldsymbol{\varepsilon}\|^2}{\partial c_0} \end{bmatrix} = -2\mathbf{X}^T\mathbf{y} + 2\mathbf{X}^T\mathbf{X}\mathbf{c} = \mathbf{0}$$

$$\hat{\mathbf{c}} = [\mathbf{X}^T\mathbf{X}]^{-1}\mathbf{X}^T\mathbf{y}$$

  – General least squares estimate of **c**
    - Vector of polynomial coefficients

LS: Polynomial Form

- Some notes:
  - Using $\hat{\mathbf{c}}$, we can estimate *y* values for given *x* values

$$\hat{\mathbf{y}} = \mathbf{X}\hat{\mathbf{c}}$$

    - Prediction
  - LS assumes a Gaussian distribution of the data

$$\{y_1, y_2, \ldots, y_m\}$$

    about the true *y*-values at the respective points
    - Not often true in practise
  - LS is a very popular method
    - Leads to a simple and optimal (sum of squared error) solution

MSE and SSE

- Important quantities:
  - The mean-square error (MSE):

$$MSE = \frac{1}{m}\sum_{i=1}^{m}(\hat{y}_i - y_i)^2$$

    - Average of squared errors between original y_i and estimated $\hat{y}_i$

  - The sum of squared errors (SSE):

$$SSE = \sum_{i=1}^{m}(\hat{y}_i - y_i)^2$$

Linear Regression

- Special case: n = 1

$$\mathbf{c} = [\mathbf{X}^T \mathbf{X}]^{-1} \mathbf{X}^T \mathbf{y}$$

$$= \left[\begin{bmatrix} x_1 & x_2 & \cdots & x_m \\ 1 & 1 & \cdots & 1 \end{bmatrix} \begin{bmatrix} x_1 & 1 \\ x_2 & 1 \\ \vdots & \vdots \\ x_m & 1 \end{bmatrix} \right]^{-1} \begin{bmatrix} x_1 & x_2 & \cdots & x_m \\ 1 & 1 & \cdots & 1 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix}^T$$

$$= \begin{bmatrix} \sum_{i=1}^{m} x_i^2 & \sum_{i=1}^{m} x_i \\ \sum_{i=1}^{m} x_i & m \end{bmatrix}^{-1} \begin{bmatrix} \sum_{i=1}^{m} x_i y_i \\ \sum_{i=1}^{m} y_i \end{bmatrix}$$

Linear Regression

- Special case: $n = 1$

$$= \frac{1}{m\sum_{i=1}^{m} x_i^2 - \left(\sum_{i=1}^{m} x_i\right)^2} \begin{bmatrix} m & -\sum_{i=1}^{m} x_i \\ -\sum_{i=1}^{m} x_i & \sum_{i=1}^{m} x_i^2 \end{bmatrix} \begin{bmatrix} \sum_{i=1}^{m} x_i y_i \\ \sum_{i=1}^{m} y_i \end{bmatrix}$$

$$\begin{bmatrix} c_1 \\ c_0 \end{bmatrix} = \frac{1}{m\sum_{i=1}^{m} x_i^2 - \left(\sum_{i=1}^{m} x_i\right)^2} \begin{bmatrix} m\sum_{i=1}^{m} x_i y_i - \sum_{i=1}^{m} x_i \sum_{i=1}^{m} y_i \\ \sum_{i=1}^{m} x_i^2 \sum_{i=1}^{m} y_i - \sum_{i=1}^{m} x_i \sum_{i=1}^{m} x_i y_i \end{bmatrix}$$

LS General Form

- So far:
  - We looked at polynomial fitting
  - LS is applicable to fitting data of the form

$$y(x) = c_0 f_0(x) + c_1 f_1(x) + \ldots + c_n f_n(x)$$

  - Much more general
    - Many models have this form
  - LS useful in a huge range of applications

LS General Form

- Matrix-Vector form:

$$\mathbf{y} = \mathbf{X}\mathbf{c} + \boldsymbol{\varepsilon}$$

$$\mathbf{y} = \begin{bmatrix} y_1 & y_2 & \cdots & y_m \end{bmatrix}^T, \quad \mathbf{X} = \begin{bmatrix} f_0(x_1) & f_1(x_1) & \dots & f_n(x_1) \\ f_0(x_2) & f_1(x_2) & \dots & f_n(x_2) \\ \vdots & \vdots & \ddots & \vdots \\ f_0(x_m) & f_1(x_m) & \dots & f_n(x_m) \end{bmatrix},$$

$$\mathbf{c} = \begin{bmatrix} c_0 & c_1 & \cdots & c_n \end{bmatrix}^T, \quad \boldsymbol{\varepsilon} = \begin{bmatrix} \varepsilon_1 & \varepsilon_2 & \cdots & \varepsilon_m \end{bmatrix}^T$$

$$\hat{\mathbf{c}} = [\mathbf{X}^T\mathbf{X}]^{-1}\mathbf{X}^T\mathbf{y}$$

LS General Form

- Sum form:
  - Sum of squared errors:

$$\|\boldsymbol{\varepsilon}\|^2 = \sum_{i=1}^{m} \left(\sum_{j=0}^{n} c_j f_j(x_i) - y_i \right)^2$$

$$\frac{\partial \|\boldsymbol{\varepsilon}\|^2}{\partial c_k} = 2 \sum_{i=1}^{m} f_k(x_i) \left(\sum_{j=0}^{n} c_j f_j(x_i) - y_i \right) = 0, \quad k = 0,1,\ldots,n$$

$$\sum_{j=0}^{n} c_j \left(\sum_{i=1}^{m} f_j(x_i) f_k(x_i) \right) = \sum_{i=1}^{m} f_k(x_i) y_i, \quad k = 0,1,\ldots,n$$

  - The normal equations

Weighted LS

- Possible to weight contribution of data:
  – Sum of squared errors becomes:

$$\|\boldsymbol{\varepsilon}\|^2 = \sum_{i=1}^{m} w_i \left(\sum_{j=0}^{n} c_j f_j(x_i) - y_i \right)^2$$

  – Derivation is similar
  – Normal equations similar, but include weight terms

Linearisation

- LS general form is great, but . . .
  - What if the equation (model) is not linear in c_j ?
  - Make it linear in terms of some other constants

$$y = ax^b \quad \rightarrow \quad \ln y = \ln a + b \ln x$$

$$y = ae^{bx} \quad \rightarrow \quad \ln y = \ln a + bx$$

$$y = \frac{ax}{b+x} \quad \rightarrow \quad \frac{1}{y} = \frac{b}{ax} + \frac{1}{a}$$

$$y = \frac{a}{b+x} \quad \rightarrow \quad \frac{1}{y} = \frac{b}{a} + \frac{x}{a}$$

LS General Form

- So far:
  - Only functions of one variable (*x*)
  - LS is applicable to linear combination of several variables

$$y(x_1,\ldots,x_n) = c_0 + c_1 x_1 + \ldots + c_n x_n$$

    - e.g. matrix multiplication, where matrix is unknown
  - Or linear combinations of functions of multiple variables

$$y(x_1,\ldots,x_p) = c_0 f_0(x_1,\ldots,x_p) + c_1 f_1(x_1,\ldots,x_p) + \ldots + c_n f_n(x_1,\ldots,x_p)$$

Nonlinear Least Squares

- So far:
  - Forms that are linear in the c_k, or could be linearised
  - What if the model is nonlinear in the parameters (c_k) ?
  - Nonlinear relations of the form

$$y = f(x, c_0, c_1, ..., c_n)$$

  - Start with an initial estimate of c_k: $\tilde{c}_k$
  - Evaluate

$$\hat{y}_i = f(x_i, \tilde{c}_0, \tilde{c}_1, ..., \tilde{c}_n)$$

Nonlinear Least Squares

– Define the error between true value and estimates as

$$\Delta\beta_i = y_i - \hat{y}_i$$

– Model the variation of $y = f(x, c_0, c_1, \ldots, c_n)$ about each data point due to each c_k using a first-order Taylor Series:

$$f(x_i, c_0, c_1, \ldots, c_n) \approx f(x_i, \tilde{c}_0, \tilde{c}_1, \ldots, \tilde{c}_n) + \sum_{k=0}^{n} \frac{\partial f(x_i, \tilde{c}_0, \tilde{c}_1, \ldots, \tilde{c}_n)}{\partial c_k} \Delta c_k$$

$$y_i \approx \hat{y}_i + \sum_{k=0}^{n} \frac{\partial f(x_i, \tilde{c}_0, \tilde{c}_1, \ldots, \tilde{c}_n)}{\partial c_k} \Delta c_k$$

– Δc_k is a small change in c_k $\quad \Delta c_k = c_k - \tilde{c}_k$

Nonlinear Least Squares

- So

$$\Delta\beta_i = y_i - \hat{y}_i = \sum_{k=0}^{n} \frac{\partial f}{\partial c_k} \Delta c_k \bigg|_{x_i,\tilde{\mathbf{c}}}, \quad i = 1,2,\ldots,m$$

$$\mathbf{\Delta\beta} = \begin{bmatrix} \left.\dfrac{\partial f}{\partial c_0}\right|_{x_1,\tilde{\mathbf{c}}} & \left.\dfrac{\partial f}{\partial c_1}\right|_{x_1,\tilde{\mathbf{c}}} & \cdots & \left.\dfrac{\partial f}{\partial c_n}\right|_{x_1,\tilde{\mathbf{c}}} \\ \left.\dfrac{\partial f}{\partial c_0}\right|_{x_2,\tilde{\mathbf{c}}} & \left.\dfrac{\partial f}{\partial c_1}\right|_{x_2,\tilde{\mathbf{c}}} & \cdots & \left.\dfrac{\partial f}{\partial c_n}\right|_{x_2,\tilde{\mathbf{c}}} \\ \vdots & \vdots & & \vdots \\ \left.\dfrac{\partial f}{\partial c_0}\right|_{x_m,\tilde{\mathbf{c}}} & \left.\dfrac{\partial f}{\partial c_1}\right|_{x_m,\tilde{\mathbf{c}}} & \cdots & \left.\dfrac{\partial f}{\partial c_n}\right|_{x_m,\tilde{\mathbf{c}}} \end{bmatrix} \mathbf{\Delta c} = \mathbf{X\Delta c}$$

Nonlinear Least Squares

- Now have a linear set of equations in Δc_k
  - Overdetermined ($m > n$)
- Solve as before:

$$\mathbf{\Delta c} = \left[\mathbf{X}^T \mathbf{X}\right]^{-1} \mathbf{X}^T \mathbf{\Delta \beta}$$

- Algorithm:
  - Start with initial estimates $\tilde{c}_k$
  - Calculate **X**, $\mathbf{\Delta\beta}$ and hence $\mathbf{\Delta c} = \left[\mathbf{X}^T \mathbf{X}\right]^{-1} \mathbf{X}^T \mathbf{\Delta \beta}$
  - Update the c_k using $\tilde{\tilde{c}}_k = \tilde{c}_k + \Delta c_k$

Nonlinear Least Squares

- Example
  - Original data: {(-1,-1), (2,3), (3,7)}
  - Initial estimate: $\tilde{c}_1 = 3$
- 1st iteration:

$$\mathbf{\Delta\beta} = \begin{bmatrix} -1 - -3 \\ 3 - 6 \\ 7 - 9 \end{bmatrix} = \begin{bmatrix} 2 \\ -3 \\ -2 \end{bmatrix}$$

$$\mathbf{X} = \begin{bmatrix} \left.\dfrac{dy}{dc_1}\right|_{x_1,\tilde{c}_1} \\ \left.\dfrac{dy}{dc_1}\right|_{x_2,\tilde{c}_1} \\ \left.\dfrac{dy}{dc_1}\right|_{x_3,\tilde{c}_1} \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} -1 \\ 2 \\ 3 \end{bmatrix}$$

$$\mathbf{\Delta\beta} = \mathbf{X}c_1, \quad \begin{bmatrix} 2 \\ -3 \\ -2 \end{bmatrix} = \begin{bmatrix} -1 \\ 2 \\ 3 \end{bmatrix} c_1$$

$$\mathbf{\Delta c} = \left[\mathbf{X}^T\mathbf{X}\right]^{-1}\mathbf{X}^T\mathbf{\Delta\beta} = [14]^{-1}\begin{bmatrix} -1 & 2 & 3 \end{bmatrix}\begin{bmatrix} 2 \\ -3 \\ -2 \end{bmatrix} = -1$$

Nonlinear Least Squares

- Example
  - Original data: {(-1,-1), (2,3), (3,7)}
  - New estimate: $\tilde{\tilde{c}}_1 = \tilde{c}_1 + \mathbf{\Delta c} = 3 - 1 = 2$

Close to optimum value 31/14

- 2nd iteration:

$$\mathbf{\Delta\beta} = \begin{bmatrix} -1 - -2 \\ 3 - 4 \\ 7 - 6 \end{bmatrix} = \begin{bmatrix} 1 \\ -1 \\ 1 \end{bmatrix}$$

$$\mathbf{X} = \begin{bmatrix} \left.\dfrac{dy}{dc_1}\right|_{x_1, \tilde{c}_1} \\ \left.\dfrac{dy}{dc_1}\right|_{x_2, \tilde{c}_1} \\ \left.\dfrac{dy}{dc_1}\right|_{x_3, \tilde{c}_1} \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} -1 \\ 2 \\ 3 \end{bmatrix}$$

$$\mathbf{\Delta\beta} = \mathbf{X}c_1, \quad \begin{bmatrix} 1 \\ -1 \\ 1 \end{bmatrix} = \begin{bmatrix} -1 \\ 2 \\ 3 \end{bmatrix} c_1$$

$$\mathbf{\Delta c} = \left[\mathbf{X}^T\mathbf{X}\right]^{-1}\mathbf{X}^T\mathbf{\Delta\beta} = [14]^{-1}\begin{bmatrix} -1 & 2 & 3 \end{bmatrix}\begin{bmatrix} 1 \\ -1 \\ 1 \end{bmatrix} = 0$$

Gradient Descent

- LS solutions keep looking like:

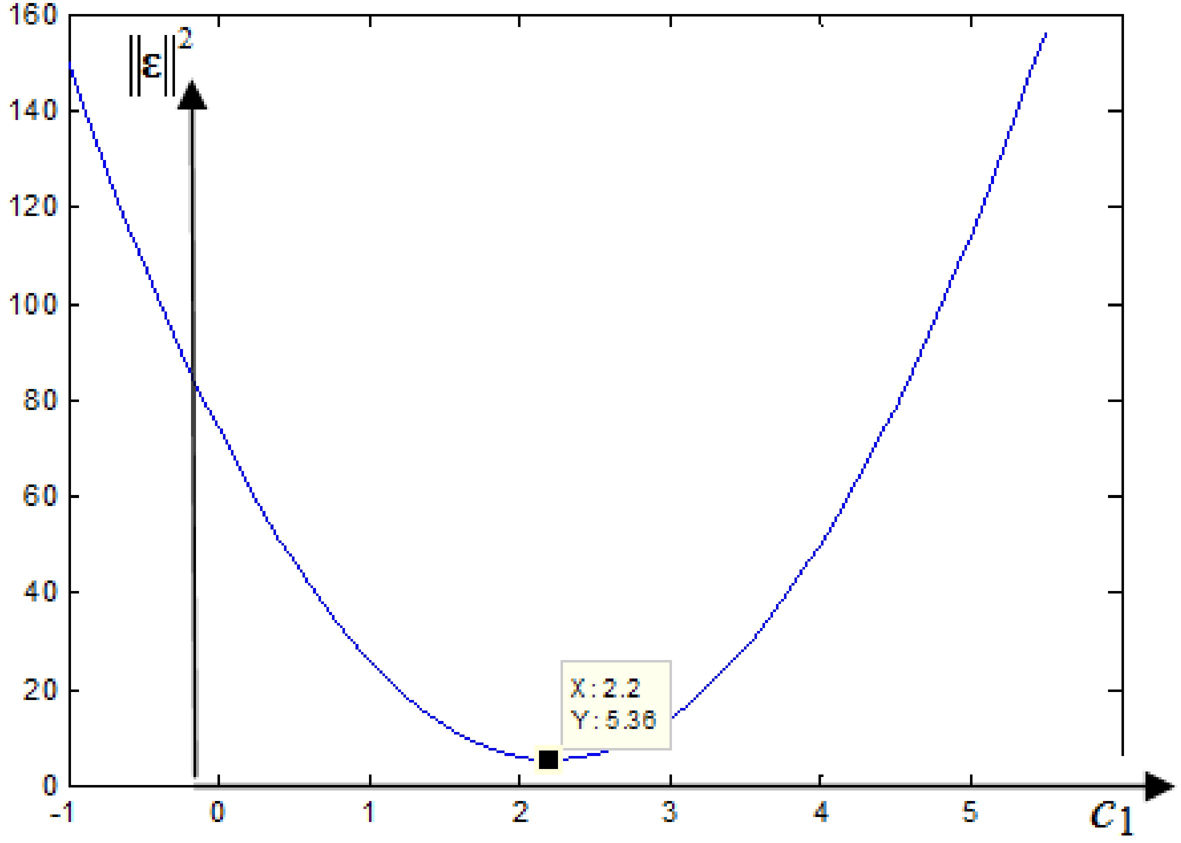
$$\mathbf{c} = [\mathbf{X}^T \mathbf{X}]^{-1} \mathbf{X}^T \mathbf{y}$$

  – Matrix inversion is not what we want to do
    - What if we have to do LS lots of times, or very fast, or we have lots of parameters (large $\mathbf{X}^T\mathbf{X}$) ?

  – Can we solve this some other way ?

Gradient Descent

- Previous example:
  - Gradient at every position on this error surface ‘points away’ from minimum



Gradient Descent

- Still on the original example:

$$\|\boldsymbol{\varepsilon}\|^2 = \sum_{i=1}^{3} \varepsilon_i^2 = (-4 + c_1)^2 + (3 - 2c_1)^2 + (7 - 3c_1)^2$$

$$= 14c_1^2 - 62c_1 + 74$$

  – Suppose we guess that c_1 is $c_1^0 = 5$
    - Can actually safely make any guess for c_1, because error surface is quadratic
  – What is the gradient of the error surface at this point ?

$$\left.\frac{d\|\boldsymbol{\varepsilon}\|^2}{dc_1}\right|_{c_1=5} = 28c_1 - 62\Big|_{c_1=5} = 28(5) - 62 = 78$$

Gradient Descent

- Still on the previous example:
  - Suppose we subtract some small proportion of the gradient, say 2%, from our initial guess of c_1
    - Pushes c_1 closer to optimum
    - Produces a new estimate of c_1

$$c_1^1 = c_1^0 - 0.02 \cdot 78 = 3.44$$

  - Now what is the gradient of the error surface at this new estimate c_1^1 ?

$$\left. \frac{d\|\boldsymbol{\varepsilon}\|^2}{dc_1} \right|_{c_1=3.44} = 28(3.44) - 62 = 34.32$$

Gradient Descent

- Still on the previous example:
  - Suppose we do it again: subtract 2% of this new gradient from our latest estimate of c_1

$$c_1^2 = c_1^1 - 0.02 \cdot 34.32 = 2.75$$

  - Now what is the gradient of the error surface at this new estimate c_1^2 ?

$$\left.\frac{d\|\boldsymbol{\varepsilon}\|^2}{dc_1}\right|_{c_1=2.75} = 28(2.75) - 62 = 15$$

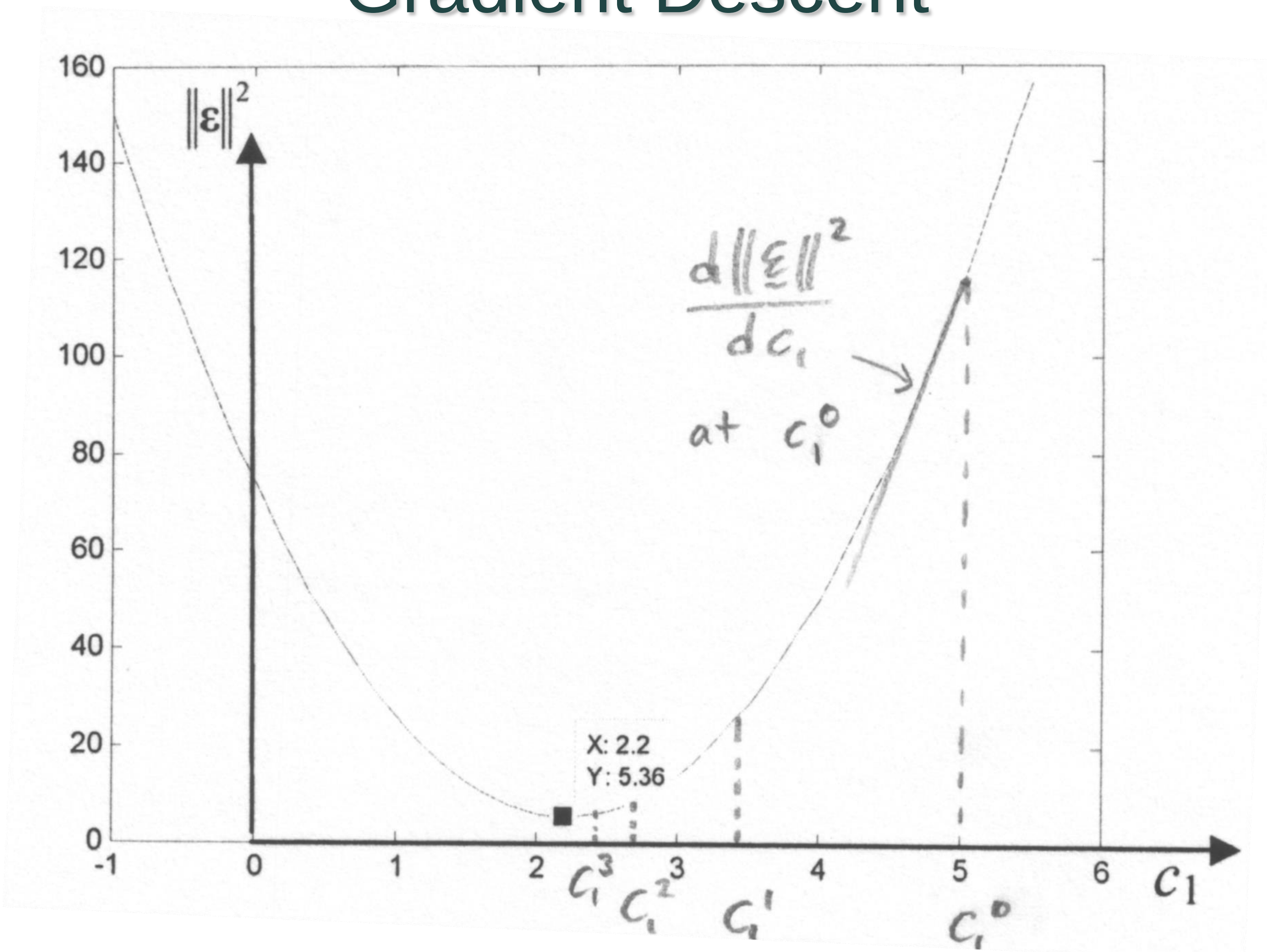
Gradient Descent

- Still on the previous example:
  - And again: subtract 2% of this new gradient from our latest estimate of c_1

$$c_1^3 = c_1^2 - 0.02 \cdot 15 = 2.45$$

  - Where are we now ?

Gradient Descent



Gradient Descent

- Algorithm:
  - Given an initial estimate of the parameter c^0,
  - An improved estimate of the minimum is obtained by

$$c^{i+1} = c^i - \lambda \left. \frac{d\|\boldsymbol{\varepsilon}\|^2}{dc} \right|_{c=c^i}$$

  - Where λ controls the rate of convergence
    - We had $\lambda = 0.02$
    - Large λ : estimates may oscillate about minimum
    - Small λ : convergence may be slow
  - i is the iteration number

Gradient Descent

- Algorithm (multiple parameter case):
  - Given an initial estimate of the parameters

$$c_k^0,\ k = 0,1,\ldots,n$$

  - An improved estimate of the minimum is obtained by

$$c_k^{i+1} = c_k^i - \lambda \left. \frac{\partial \|\boldsymbol{\varepsilon}\|^2}{\partial c_k} \right|_{c_k = c_k^i},\ k = 0,1,\ldots,n$$