Learning

Deductive Reasoning

- **Deductive Reasoning** – A type of logic in which one goes from a general statement to a specific instance.
- The classic example

  All men are mortal. (premise-I)

  Socrates is a man. (premise-II)

  Therefore, Socrates is mortal. (conclusion)

Inductive Reasoning

Inductive Reasoning, involves going from a series of specific cases to a general statement. The conclusion in an inductive argument is never guaranteed.

Example: What is the next number in the sequence 6, 13, 20, 27,…

There is more than one correct answer.

Inductive Reasoning

- Here's the sequence again 6, 13, 20, 27,…
- Look at the difference of each term.
- 13 – 6 = 7, 20 – 13 = 7, 27 – 20 = 7
- Thus the next term is 34, because 34 – 27 = 7.
- However what if the sequence represents the dates. Then the next number could be 3 (31 days in a month).
- The next number could be 4 (30 day month)
- Or it could be 5 (29 day month – Feb. Leap year)
- Or even 6 (28 day month – Feb.)

Two types of learning in AI

Deductive: Deduce rules/facts from already known rules/facts. (We have already dealt with this)

$$(A \Rightarrow B \Rightarrow C) \Rightarrow (A \Rightarrow C)$$

Inductive: Learn new rules/facts from a data set $\mathcal{D}$.

$$\mathcal{D} = \{\mathbf{x}(n), y(n)\}_{n=1\ldots N} \Rightarrow (A \Rightarrow C)$$

Motivation

- Learning is important for agents to deal with
  - Unknown environments (lacks omniscience)
  - Changes
- In many cases, it is more efficient to train an agent via examples, than to "manually" extract knowledge from the examples
- Agents capable of learning can improve their performance

Traditional Programming

Machine Learning

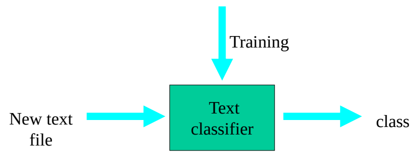
Example 1: Text Classification

Classified text files

Text file 1 trade

Text file 2 ship

… …



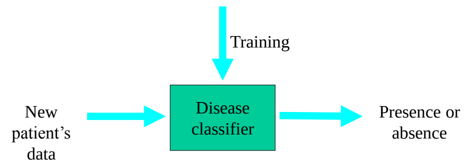
Example 2: Disease Diagnosis

Database of medical records

Patient 1's data Absence

Patient 2's data Presence

… …



Inductive Learning

- **Inductive learning** or "**Prediction**":
  - **Given** examples of a function *(X, F(X))*
  - **Predict** function *F(X)* for new examples *X*
- Classification: Learning categories
  F(X) = Discrete
- Regression: Learning function values
  F(X) = Continuous

Inductive learning

Simplest form: learn a function from examples

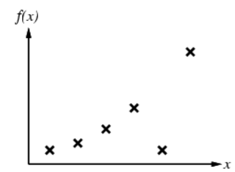
f is the target function

An example is a pair (*x*, *f(x)*)

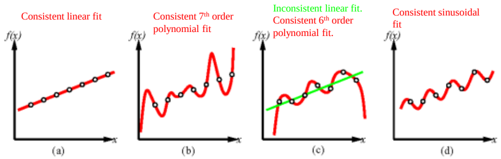
Problem: find a hypothesis ***h***
such that $h \approx f$
given a training set of examples

Inductive learning method

- Construct/adjust *h* to agree with *f* on training set
- (*h* is consistent if it agrees with *f* on all examples)
- E.g., curve fitting:



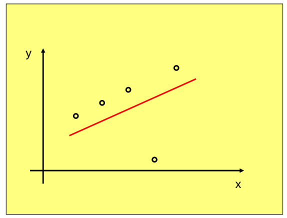
Inductive learning



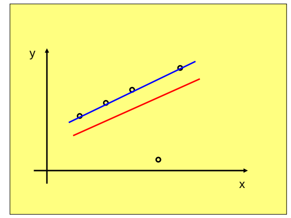
- Construct h so that it agrees with f.
- The hypothesis h is consistent if it agrees with f on all observations.

- How to achieve good generalization?

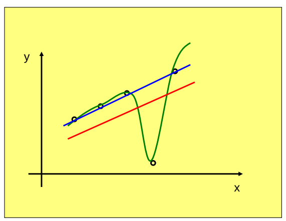
Inductive learning – example



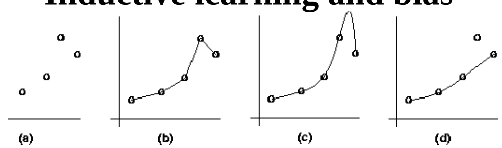
Inductive learning – example



Inductive learning – example



Inductive learning and bias

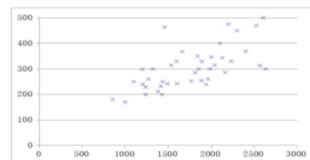


- Suppose that we want to learn a function h and we are given some sample (x, f(x)) pairs, as in figure (a)
- There are several hypotheses we could make about this function, e.g.: (b), (c) and (d)
- A preference for one over the others reveals the **bias** of our learning technique, e.g.:
  - prefer piece-wise functions
  - prefer a smooth function
  - prefer a simple function and treat outliers as noise

Linear Regression with one Variable

Housing Prices (Portland, OR)

Price (in 1000s of dollars)



Size (feet²)

Supervised Learning

Given the "right answer" for each example in the data.

Regression Problem

Predict real-valued output

Training set of housing prices

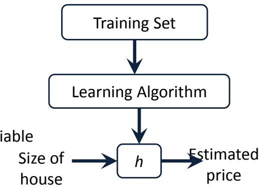
Size in feet² (x)	Price ($) in 1000's (y)
2104	460
1416	232
1534	315
852	178
…	…

Notation:

m = Number of training examples

x's = "input" variable / features

y's = "output" variable / "target" variable



Question : How to describe ***h***?

Training Set

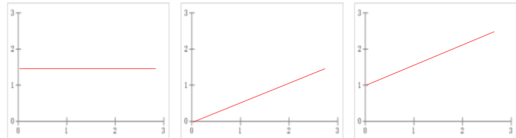
Size in feet² (x)	Price ($) in 1000's (y)
2104	460
1416	232
1534	315
852	178
…	…

Hypothesis: $h_\theta(x) = \theta_0 + \theta_1 x$

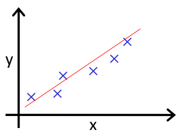
θ_i's: Parameters

How to choose θ_i's ?

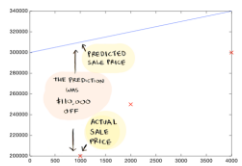
$h_\theta(x) = \theta_0 + \theta_1 x$



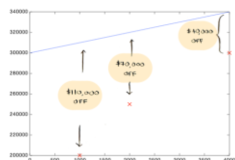
$\theta_0 = 1.5$	$\theta_0 = 0$	$\theta_0 = 1$
$\theta_1 = 0$	$\theta_1 = 0.5$	$\theta_1 = 0.5$



Idea: Choose θ_0, θ_1 so that
$h_\theta(x)$ is close to y for our
training examples (x, y)



Maybe a drunk person drew this line...it is $110,000 dollars off the mark for that house. It is also far off all the other values:

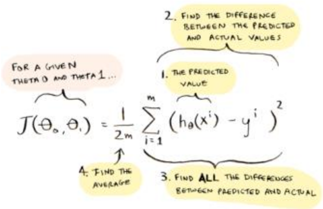


On average, this line is $73,333 off ($110,000 + $70,000 + $40,000 / 3).

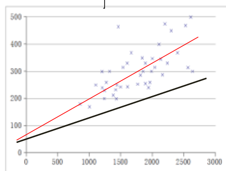
Cost Function

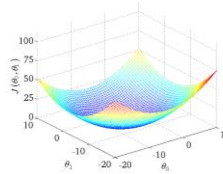
Hypothesis:	Simplified:
$h_\theta(x) = \theta_0 + \theta_1 x$	$h_\theta(x) = \theta_1 x$
Parameters:	
θ_0, θ_1	θ_1
Cost Function:	
$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$	$J(\theta_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$
Goal: $\underset{\theta_0, \theta_1}{\text{minimize}}\ J(\theta_0, \theta_1)$	$\underset{\theta_1}{\text{minimize}}\ J(\theta_1)$

Cost Function



Price ($) in 1000's





Size in feet^2 (x)

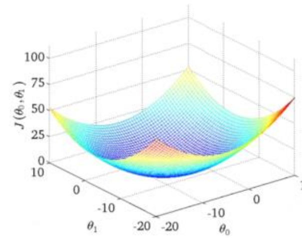
$h_\theta(x) = 50 + 0.06x$

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$$

$$\underset{\theta_0, \theta_1}{\text{minimize}}\ J(\theta_0, \theta_1)$$

Question:How to minimize ***J***?

The mean is halved as a convenience for the computation of the gradient descent, as the derivative term of the square function will cancel out the ($\frac{1}{2}$) term.



We need an algorithm to automatically find the set of parameters θ_0 and θ_1 that minimizes the cost function $J(\theta_0, \theta_1)$, which is always a **convex** function.

The cost function used for lecture is a convex function. In general don't necessarily have a convex cost function though we prefer to have one.

Gradient Descent

Have some function $J(\theta_0, \theta_1)$

Want $\min_{\theta_0, \theta_1} J(\theta_0, \theta_1)$

Outline:

- Start with some θ_0, θ_1
- Keep changing θ_0, θ_1 to reduce $J(\theta_0, \theta_1)$ until we hopefully end up at a minimum

Gradient descent algorithm

repeat until convergence {
$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1) \quad (\text{for } j = 0 \text{ and } j = 1)$$
}

Correct: Simultaneous update

$\text{temp0} := \theta_0 - \alpha \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1)$
$\text{temp1} := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1)$
$\theta_0 := \text{temp0}$
$\theta_1 := \text{temp1}$

Incorrect:

$\text{temp0} := \theta_0 - \alpha \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1)$
$\theta_0 := \text{temp0}$
$\text{temp1} := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1)$
$\theta_1 := \text{temp1}$

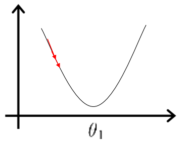
Gradient descent algorithm

repeat until convergence {
$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1) \quad (\text{simultaneously update } j = 0 \text{ and } j = 1)$$
}

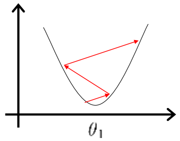
α is the learning rate.

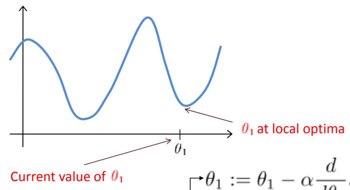
$$\theta_1 := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_1)$$

If α is too small, gradient descent can be slow.



If α is too large, gradient descent can overshoot the minimum. It may fail to converge, or even diverge.





$$\theta_1 := \theta_1 - \alpha \frac{d}{d\theta_1} J(\theta_1)$$

Unchange

Gradient descent can converge to a local minimum, even with the learning rate α fixed.

Gradient Descent for Linear Regression

Gradient descent algorithm

$$\begin{aligned} &\text{repeat until convergence } \{ \\ &\quad \theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1) \\ &\quad\quad (\text{for } j = 1 \text{ and } j = 0) \\ &\} \end{aligned}$$

Linear Regression Model

$$h_\theta(x) = \theta_0 + \theta_1 x$$

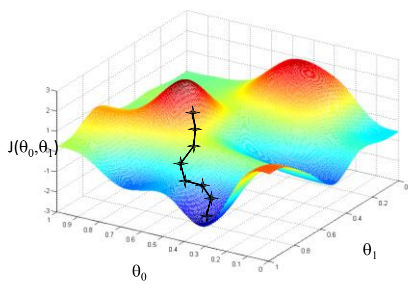
$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$$

Gradient descent algorithm

$$\begin{aligned} &\text{repeat until convergence } \{ \quad \frac{\partial}{\partial \theta_1} J(\theta) \\ &\quad \theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right) \\ &\quad \theta_1 := \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right) \cdot x^{(i)} \\ &\} \end{aligned}$$

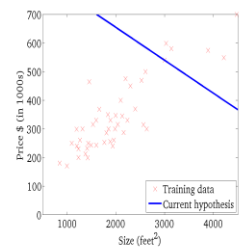
update θ_0 and θ_1 simultaneously

This method looks at each example in the entire training set on every step, and is called <u>batch gradient descent</u>.



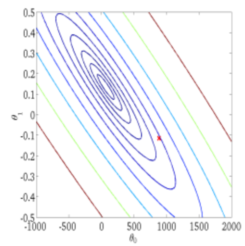
$h_\theta(x)$

(for fixed θ_0, θ_1 ,this is a function of x)



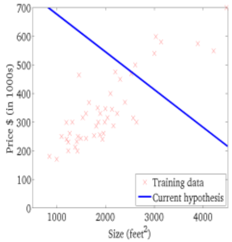
$J(\theta_0, \theta_1)$

(function of the parametersθ_0, θ_1)



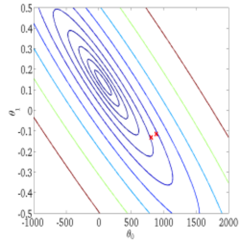
$h_\theta(x)$

(for fixed θ_0, θ_1 , this is a function of x)



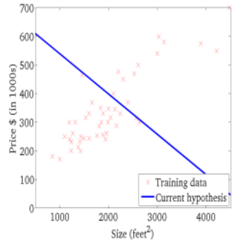
$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)



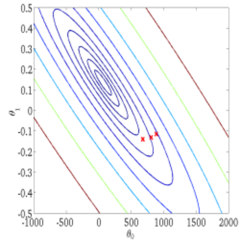
$h_\theta(x)$

(for fixed θ_0, θ_1 , this is a function of x)



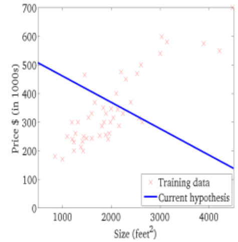
$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)



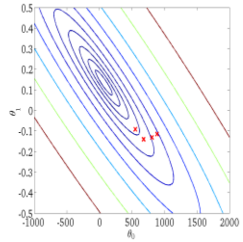
$h_\theta(x)$

(for fixed θ_0, θ_1 , this is a function of x)



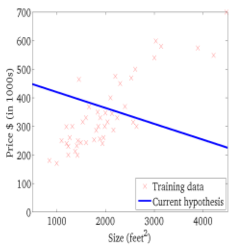
$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)



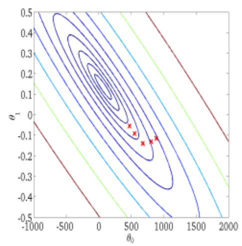
$h_\theta(x)$

(for fixed θ_0, θ_1 , this is a function of x)



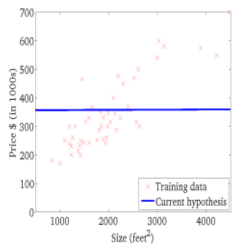
$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)



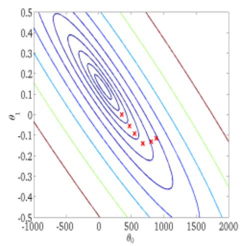
$h_\theta(x)$

(for fixed θ_0, θ_1 , this is a function of x)



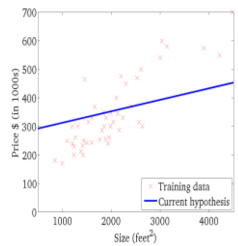
$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)



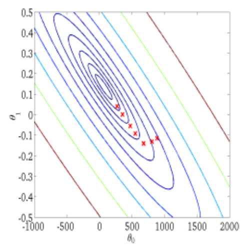
$h_\theta(x)$

(for fixed θ_0, θ_1 , this is a function of x)



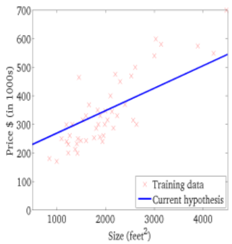
$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)



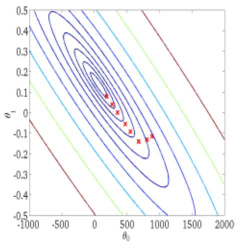
$h_\theta(x)$

(for fixed θ_0, θ_1 , this is a function of x)



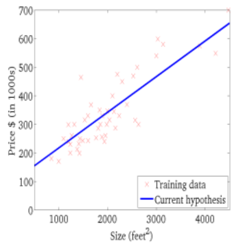
$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)



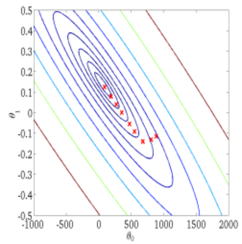
$h_\theta(x)$

(for fixed θ_0, θ_1, this is a function of x)



$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)



Linear Regression with multiple variables

Hypothesis: $h_\theta(x) = \theta^T x = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$

Parameters: $\theta_0, \theta_1, \ldots, \theta_n$

Cost function:

$$J(\theta_0, \theta_1, \ldots, \theta_n) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$$

Gradient descent:

Repeat $\{$

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \ldots, \theta_n)$$

$\}$ (simultaneously update for every $j = 0, \ldots, n$)

Gradient Descent

Previously (n=1):

Repeat $\{$

$$\theta_0 := \theta_0 - \alpha \underbrace{\frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})}_{\frac{\partial}{\partial \theta_0} J(\theta)}$$

$$\theta_1 := \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x^{(i)}$$

(simultaneously update θ_0, θ_1)

$\}$

New algorithm $(n \geq 1)$:

Repeat $\{$

$$\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

(simultaneously update θ_j for $j = 0, \ldots, n$)

$\}$

$$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_0^{(i)}$$

$$\theta_1 := \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_1^{(i)}$$

$$\theta_2 := \theta_2 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_2^{(i)}$$

Speeding up Gradient Descent

- We can speed it up by having each of our input values roughly in the same range. This is because θ will descend quickly on small ranges than on large ranges, and hence will oscillate inefficiently down to the optimum when the variables are very uneven.
- Two ways:
  - Feature scaling
  - Mean Normaization

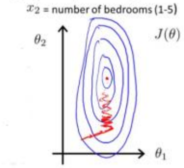
Feature Scaling

- It involves dividing the input values by the range of the input variables, resulting in the new range of just 1.

Idea: Make sure features are on a similar scale.

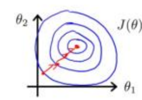
E.g. x_1 = size (0-2000 feet^2)

x_2 = number of bedrooms (1-5)



$x_1 = \frac{\text{size (feet}^2\text{)}}{2000}$

$x_2 = \frac{\text{number of bedrooms}}{5}$



Mean Normalization

Replace x_i with $x_i - \mu_i$ to make features have approximately zero mean (Do not apply to $x_0 = 1$).

E.g. $x_1 = \frac{size - 1000}{2000}$ mean

$x_2 = \frac{\#bedrooms - 2}{5}$ Range (max-min)

House Price Prediction Example

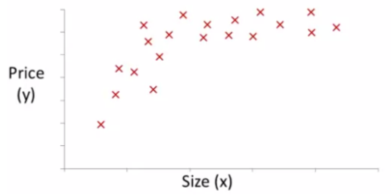
- Let the hypothesis be:

$$h_\theta(x) = \theta_0 + \theta_1 \times frontage + \theta_2 \times depth$$

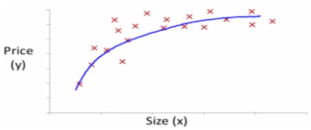
- But, its not necessary to use the features provided as it is. We can form new features like
  area= frontage * depth
  and refine the hypothesis
- Model might work better

Polynomial Regression

- The idea close to defining new features is called polynomial regression
- Eg:



The idea of fitting a line does not seem promising, so we might try
To fit a quadratic function like the one given below:



$$\theta_0 + \theta_1 x + \theta_2 x^2$$

We might try a cubic function and others.

A cubic function seems to be a better fit and we might use techniques of multivariate regression to solve it.

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3$$
$$= \theta_0 + \theta_1 (size) + \theta_2 (size)^2 + \theta_3 (size)^3$$
$$x_1 = (size)$$
$$x_2 = (size)^2$$
$$x_3 = (size)^3$$

Linear Regression Model

Relationship Between Variables Is a Linear Function

Population Y-Intercept

Population Slope

Random Error

$$Y_i = \beta_0 + \beta_1 X_i + \varepsilon_i$$

Dependent (Response) Variable

Independent (Explanatory) Variable

Fit the regression line $y = \beta_0 + \beta_1 x$ to the data

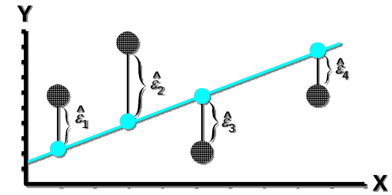
$$(x_1, y_1), \ldots, (x_n, y_n)$$

by finding the "best" match between the line and the data. The "best" choice of β_0, β_1 will be chosen to minimize

$$Q = \sum_{i=1}^{n} (y_i - (\beta_0 + \beta_1 x_i))^2 = \sum_{i=1}^{n} \varepsilon_i^2.$$

Least Squares Graphically

LS minimizes $\sum_{i=1}^{n} \hat{\varepsilon}_i^2 = \hat{\varepsilon}_1^2 + \hat{\varepsilon}_2^2 + \hat{\varepsilon}_3^2 + \hat{\varepsilon}_4^2$



This is called the least square fit. Let's solve...

$$\frac{\partial Q}{\partial \beta_0} = -2\sum(y_i - (\beta_0 + \beta_1 x_i)) = 0$$
$$\frac{\partial Q}{\partial \beta_1} = -2\sum x_i(y_i - (\beta_0 + \beta_1 x_i)) = 0$$

$$\Leftrightarrow \sum y_i = n\beta_0 + \beta_1 \sum x_i$$
$$-2\sum x_i(y_i - (\beta_0 + \beta_1 x_i)) = 0$$

After a little algebra, get

$$\hat{\beta}_1 = \frac{n\sum x_i y_i - (\sum x_i)(\sum y_i)}{n\sum x_i^2 - (\sum x_i)^2}$$
$$\hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \bar{x}, \text{ where } \bar{y} \equiv \frac{1}{n}\sum y_i \text{ and } \bar{x} \equiv \frac{1}{n}\sum x_i.$$

Let's introduce some more notation:

$$S_{xx} = \Sigma(x_i - \bar{x})^2 = \sum x_i^2 - n\bar{x}^2$$
$$= \sum x_i^2 - \frac{(\sum x_i)^2}{n}$$
$$S_{xy} = \Sigma(x_i - \bar{x})(y_i - \bar{y}) = \sum x_i y_i - n\bar{x}\bar{y}$$
$$= \sum x_i y_i - \frac{(\sum x_i)(\sum y_i)}{n}$$

These are called "sums of squares."

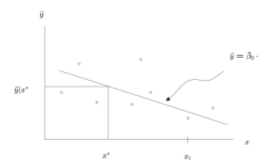
Then, after a little more algebra, we can write

$$\hat{\beta}_1 = \frac{S_{xy}}{S_{xx}}$$

the fitted regression line is:

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x.$$

Fix a specific value of the explanatory variable x^*, the equation gives a fitted value $\hat{y}|x^* = \hat{\beta}_0 + \hat{\beta}_1 x^*$ for the dependent variable y.



For actual data points x_i, the fitted values are $\hat{y}_i = \hat{\beta}_0 + \hat{\beta}_1 x_i$.

observed values : $y_i = \beta_0 + \beta_1 x_i + \varepsilon_i$

fitted values : $\hat{y}_i = \hat{\beta}_0 + \hat{\beta}_1 x_i$

Let's estimate the error variation σ^2 by considering the deviations between y_i and $\hat{y}_i$.

$$SSE = \Sigma(y_i - \hat{y}_i)^2 = \Sigma(y_i - (\hat{\beta}_0 + \hat{\beta}_1 x_i))^2$$
$$= \sum y_i^2 - \hat{\beta}_0 \sum y_i - \hat{\beta}_1 \sum x_i y_i.$$