COGS 118A, Winter 2020

Supervised Machine Learning Algorithms

Lecture 5: Linear Regression

Zhuowen Tu

Midterm 1

Midterm I, 01/30/2020 (Thursday)

Time: 12:30-13:50PM

Location: Ledden Auditorium

You can bring one page “cheat sheet”. No use of computers/smart-phones during the exam.

Bring your pen.

Bring your calculator.

A study guide and practice questions will be provided.

No homework assignment for the next week.

An example

Birthweight based on the mother's Estriol

Estriol (mg/24h)	Birthweight (g/1000)
1	1
3	1.9
2	1.05
5	4.1
4	2.1
x	y

https://www.dailyclipart.net/

Linear Regression

The basic idea of linear regression is to learn a linear function:

$$
\begin{aligned}
f(\mathbf{x};\mathbf{w},b) &=< \mathbf{w},\mathbf{x} > +b \\
&= \mathbf{w}\cdot\mathbf{x} + b \\
&= \mathbf{w}^T\mathbf{x} + b
\end{aligned}
$$

$\mathbf{x} = \mathbb{R}^m$ $\qquad$ $\mathbf{w} = \mathbb{R}^m$ $\qquad$ $b \in \mathbb{R}$

Further: $W = (\mathbf{w}, b)$ since b can be also viewed as a parameter in W when a constant 1 is appended to every $\mathbf{x}$.

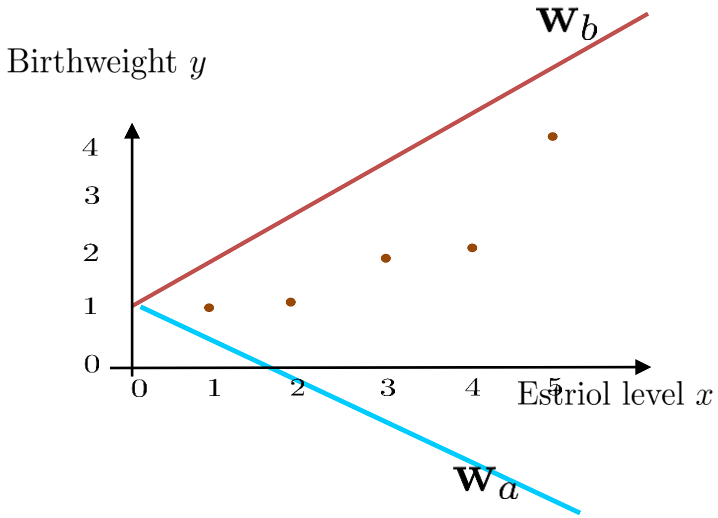
This is a linear function and our job is find the optimal **w** and **b** to best fit the prediction in learning.

Once learned, the linear regression function can be readily computed.

An example

Training data

Estriol (mg/24h)	Birthweight (g/1000)
1	1
3	1.9
2	1.05
5	4.1
4	2.1
x	y



Which is a more preferred model for fitting the data?

A. W_a

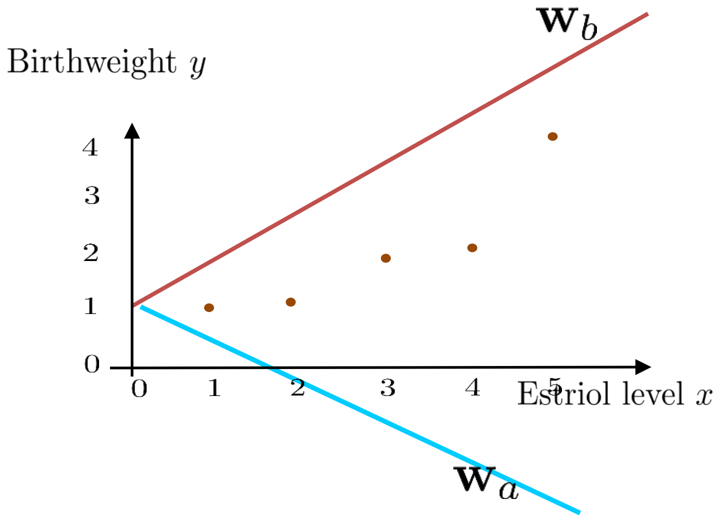
B. W_b

C. It depends.

An example

Training data

Estriol (mg/24h)	Birthweight (g/1000)
1	1
3	1.9
2	1.05
5	4.1
4	2.1
x	y



Which is a more preferred model for fitting the data?

A. W_a

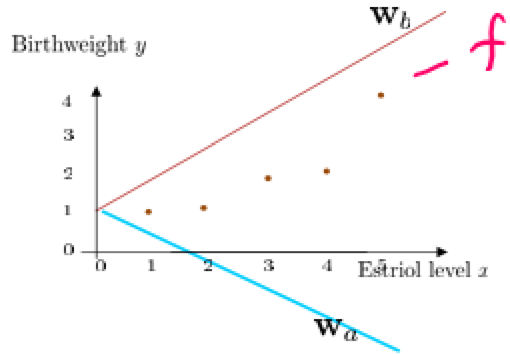
☆ B. W_b

C. It depends.

An example

Training data

Estriol (mg/24h)	Birthweight (g/1000)
1	1
3	1.9
2	1.05
5	4.1
4	2.1
x	y



$f(x; w, b) = w x + b$

$b = 0$

Which is a more preferred model for fitting the data?

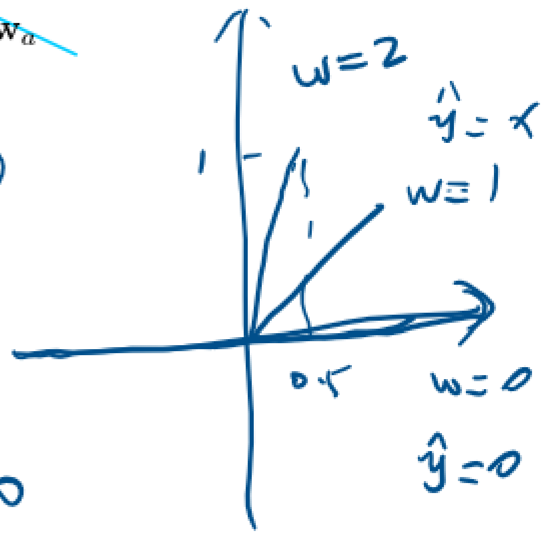
A. W_a

B. W_b

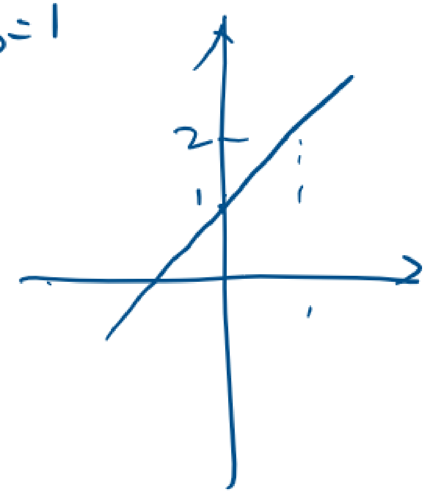
C. It depends.

$W = (w, b)$

$\hat{y} = w \times x + 0$

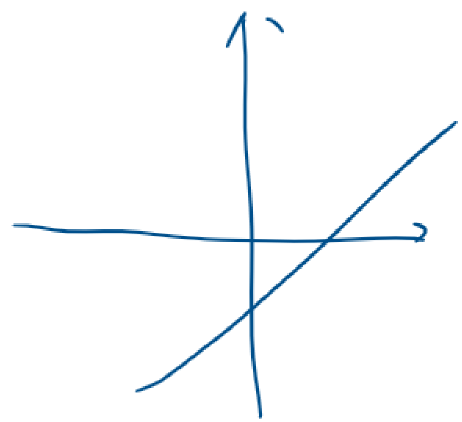


$w=1$, $b=1$



$1 \times 0 + 1 = 1$

$1 \times 1 + 1 = 2$

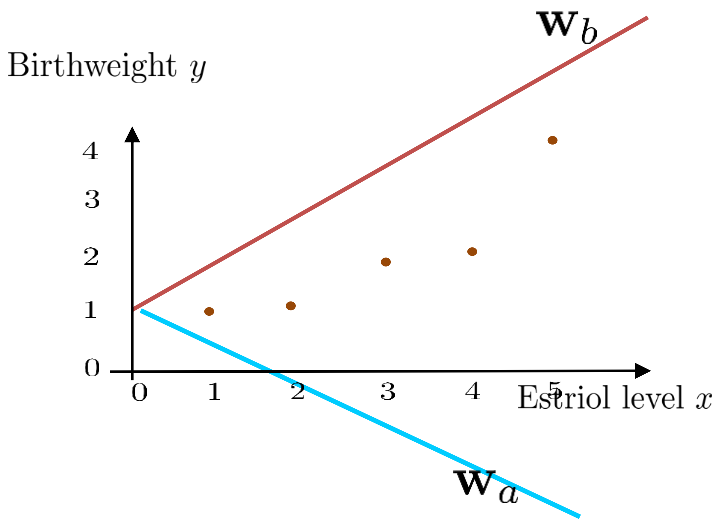


$w=1$, $b=-1$

An example

Training data

Estriol (mg/24h)	Birthweight (g/1000)
1	1
3	1.9
2	1.05
5	4.1
4	2.1



$$S_{training} = \{(x_i, y_i), i = 1..n\} = \{(1,1), (3,1.9), (2,1.05), (5,4.1), (4,2.1)\}$$

W_a

If $\mathbf{w}_a = (w_0, w_1) = (1, -0.5)$

$$e_{training}(\mathbf{w}_a) = \frac{1}{5}\sum_{i=1}^{5}(y_i - (1 - 0.5x_i))^2 = 9.62$$

W_b

If $\mathbf{w}_b = (w_0, w_1) = (1, 0.5)$

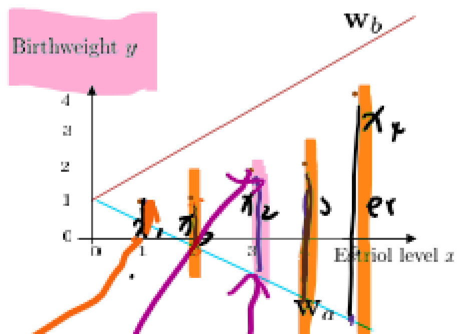
$$e_{training}(\mathbf{w}_b) = \frac{1}{5}\sum_{i=1}^{5}(y_i - (1 + 0.5x_i))^2 = 0.54$$

$\mathbf{w}_b$ is better than $\mathbf{w}_a$ since $e_{training}(\mathbf{w}_b) < e_{training}(\mathbf{w}_a)$

An example

Training data

Estriol (mg/24h)	Birthweight (g/1000)
1	1
3	1.9
2	1.05
5	4.1
4	2.1
x	y



$S_{training} = \{(x_i, y_i), i = 1..n\} = \{(1,1), (3,1.9), (2,1.05), (5,4.1), (4,2.1)\}$

W_a If $\mathbf{w}_a = (w_0, w_1) = (1, -0.5)$

$e_{training}(\mathbf{w}_a) = \frac{1}{5}\sum_{i=1}^{5}(y_i - (1 - 0.5x_i))^2 = 9.62$

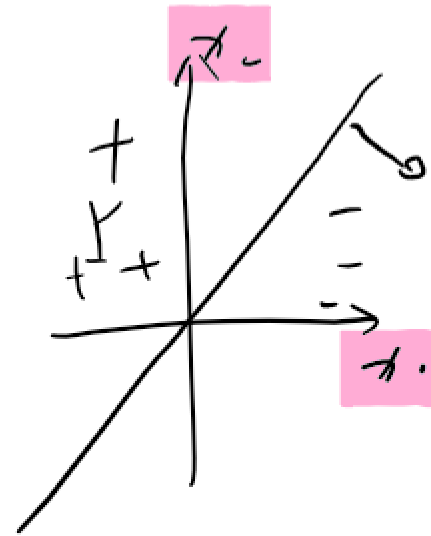
W_b If $\mathbf{w}_b = (w_0, w_1) = (1, 0.5)$

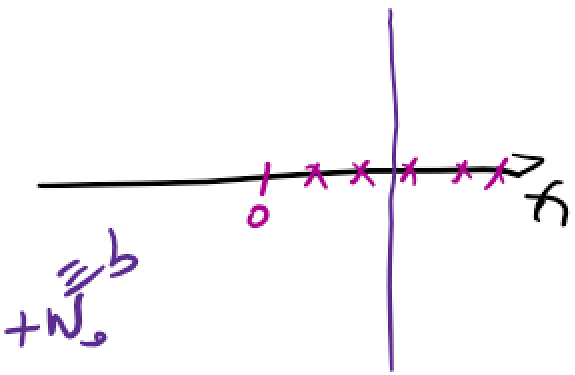
$e_{training}(\mathbf{w}_b) = \frac{1}{5}\sum_{i=1}^{5}(y_i - (1 + 0.5x_i))^2 = 0.54$

$\mathbf{w}_b$ is better than $\mathbf{w}_a$ since $e_{training}(\mathbf{w}_b) < e_{training}(\mathbf{w}_a)$

$w_1 \times x + w_0 \equiv b$

$(1 - (1 - 0.5 \times 1))^2 + (1.9 - (1 - 0.5 \times 2))^2 + \cdots +$

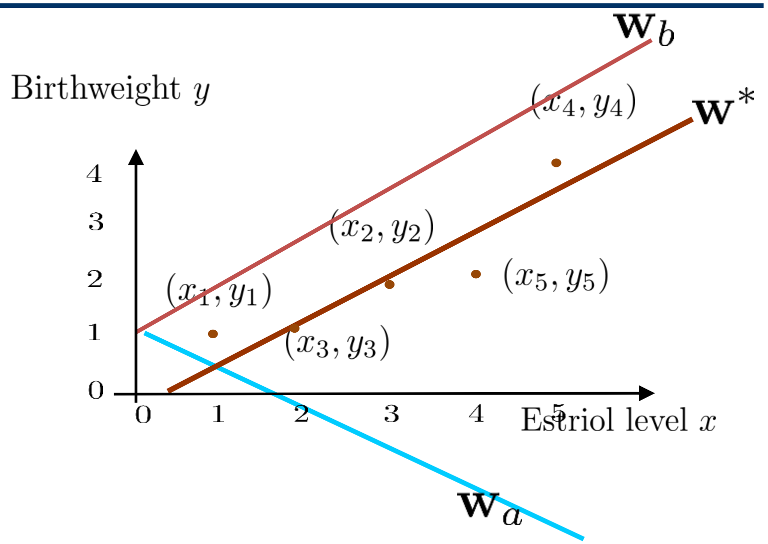




An example

Training data

Estriol (mg/24h)	Birthweight (g/1000)
1	1
3	1.9
2	1.05
5	4.1
4	2.1



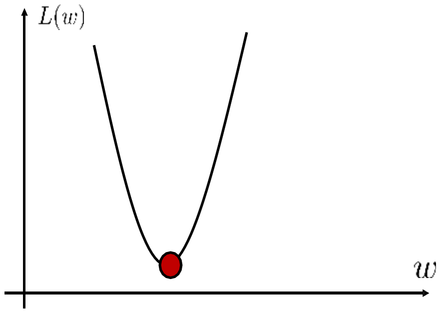
$$S_{training} = \{(x_i, y_i), i = 1..n\} = \{(1,1), (3,1.9), (2,1.05), (5,4.1), (4,2.1)\}$$

Our goal is to find the optimal $\mathbf{w}^* = (w_0, w_1)^*$:

$$\mathbf{w}^* = \arg_{\mathbf{w}} \min e_{training}(\mathbf{w}) = \frac{1}{5} \sum_{i=1}^{5} (y_i - (w_0 + w_1 \times x_i))^2$$

Why do we study/care about derivatives?

1. Find the optimal solution using an analytical (closed) form for a convex function that is everywhere differentiable.



$$\frac{\partial L(w)}{\partial w}\Big|_{w^*} = 0$$

Analytical (closed) form refers to a direct solution as:

$w^* = q(X, Y)$ where X and Y consists of your training data with the corresponding ground-truth labels.

That is, you obtain your model by one-shot (no iterations needed).

More Complex Linear Combinations

- Univariate linear regression

- Polynomial Linear Regression

- Multivariate Linear Regression

Linear regression algorithm

- Step 1: Select a model.
  - $y = w_0 + w_1 x$ (For us right now – not always)

Linear Regression Algorithm

- Step 1: Select a model.
  - $y = w_0 + w_1 x$ (For us right now – not always)
- Step 2: Put data into matrix form.
  - $\mathbf{y} = X\mathbf{w}$;

Step 3: Solve for the weights

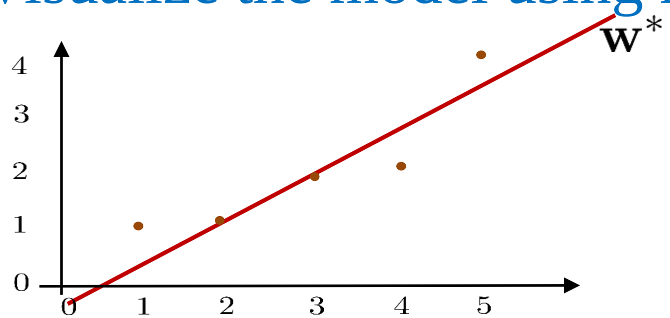
- Step 1: Select a model.
  - $y = w_0 + w_1 x$ (For us right now – not always)
- Step 2: Put data into matrix form.
  - $\mathbf{y} = X\mathbf{w}$;
- Step 3: Solve for the weights (**w**) in Python.
  - **from numpy.linalg import inv**
  - **w=np.dot(inv(np.dot(np.transpose(X), X)),np.dot(np.transpose(X),Y))**

Step 4: Report the model

- Step 1: Select a model.
  - $y = w_0 + w_1 x$ (For us right now – not always)
- Step 2: Put data into matrix form.
  - $\mathbf{y} = A\mathbf{w}$;
- Step 3: Solve for the weights (**w**) in Python.
  - **from numpy.linalg import inv**
  - **w=np.dot(inv(np.dot(np.transpose(X), X)),np.dot(np.transpose(X),Y))**
- Step 4: Report the model.
  - Fill in the **w** values in our model (step 1) and write out again.
  - $\mathbf{y = -0.145 + 0.725x}$ (For us right now – not always)

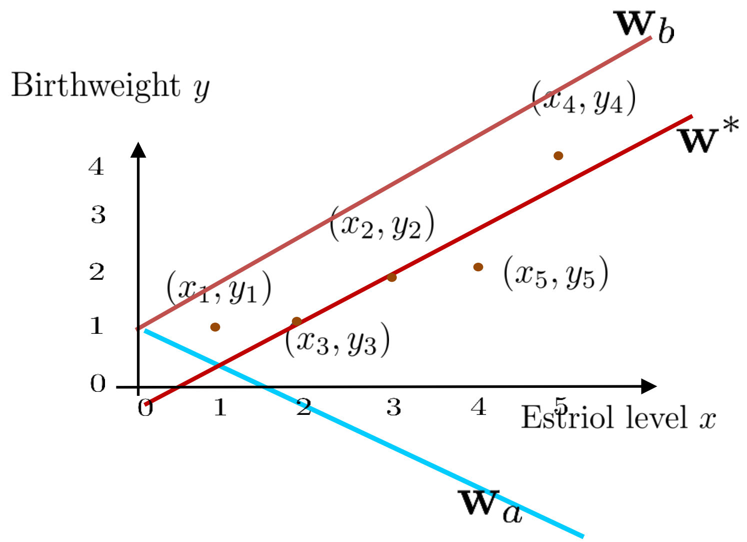
Step 5: Visualize the model

- Step 1: Select a model.
  - $y = w_0 + w_1 x$ (For us right now – not always)
- Step 2: Put data into matrix form.
  - $\mathbf{y} = A\mathbf{w}$;
- Step 3: Solve for the weights (**w**) in Python.
  - **from numpy.linalg import inv**
  - **w=np.dot(inv(np.dot(np.transpose(X), X)),np.dot(np.transpose(X),Y))**
- Step 4: Report the model.
  - Fill in the **w** values in our model (step 1) and write out again.
  - **$y = -0.145 + 0.725x$** (For us right now – not always)
- Step 5: Visualize the model using model predictions.



Least Square Solution

$$S_{training} = \{(x_i, y_i), i = 1..n\} = \{(1,1), (3,1.9), (2,1.05), (5,4.1), (4,2.1)\}$$



Basic Equations

$$y = w_0 + w_1 x$$

$$1 = w_0 + w_1 \times 1$$
$$1.9 = w_0 + w_1 \times 3$$
$$1.05 = w_0 + w_1 \times 2$$
$$4.1 = w_0 + w_1 \times 5$$
$$2.1 = w_0 + w_1 \times 4$$

Matrix Form

$$Y = XW$$

$$\begin{array}{ccccc} Y & = & X & W & \\ \begin{pmatrix} 1 \\ 1.9 \\ 1.05 \\ 4.1 \\ 2.1 \end{pmatrix} & & \begin{pmatrix} 1, 1 \\ 1, 3 \\ 1, 2 \\ 1, 5 \\ 1, 4 \end{pmatrix} & \begin{pmatrix} w_0 \\ w_1 \end{pmatrix} & \begin{pmatrix} -0.145 \\ 0.725 \end{pmatrix} \end{array}$$

$$W^* = (X^T X)^{-1} X^T Y$$

In python from numpy.linalg import inv

W* = np.dot(inv(np.dot(np.transpose(X), X)),np.dot(np.transpose(X),Y))

Least Square Solution

$S_{training} = \{(x_i, y_i), i = 1..n\} = \{(1,1), (3,1.9), (2,1.05), (5,4.1), (4,2.1)\}$

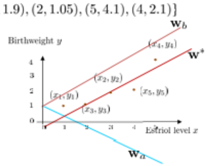
Basic Equations

$y = w_0 + w_1 x$

$1 = w_0 + w_1 \times 1$
$1.9 = w_0 + w_1 \times 3$
$1.05 = w_0 + w_1 \times 2$
$4.1 = w_0 + w_1 \times 5$
$2.1 = w_0 + w_1 \times 4$

Matrix Form

$Y = XW$



$$\begin{array}{cccc} Y & = & X & W \\ \begin{pmatrix} 1 \\ 1.9 \\ 1.05 \\ 4.1 \\ 2.1 \end{pmatrix} & & \begin{pmatrix} 1,1 \\ 1,3 \\ 1,2 \\ 1,5 \\ 1,4 \end{pmatrix} & \begin{pmatrix} w_0 \\ w_1 \end{pmatrix} \end{array} \qquad \begin{pmatrix} -0.145 \\ 0.725 \end{pmatrix}$$

$W^* = (X^T X)^{-1} X^T Y$

In python from numpy.linalg import inv
W* = np.dot(inv(np.dot(np.transpose(X), X)),np.dot(np.transpose(X),Y))

$W = (w_0, w_1)$

$(y_1 - (w_0 + w_1 x_1))^2 \quad (1 - (w_0 + w_1 \times 1))^2$

$\vdots$

$\sum_{i=1}^{5} (y_i - (w_0 + w_1 x_i))^2$

$= (Y - XW)^T (Y - XW)$

$(Y - XW)^T = (1 - (w_0 + w_1), \ldots, 2.1 - (w_0 + 4w_1))$ $\begin{pmatrix} 1 - (w_0 + w_1) \\ \vdots \\ 2.1 - (w_0 + 4w_1) \end{pmatrix}$

1×5

$$\underset{5\times 1}{\begin{pmatrix} 1 \\ 1.9 \\ 1.05 \\ 4.1 \\ 2.1 \end{pmatrix}} - \underset{5 \times 1}{\overset{5\times 2}{\begin{pmatrix} 1,1 \\ 1,3 \\ 1,2 \\ 1,5 \\ 1,4 \end{pmatrix}}} \overset{2\times 1}{\begin{pmatrix} w_0 \\ w_1 \end{pmatrix}} = \overset{Y}{\begin{pmatrix} 1 \\ 1.9 \\ 1.05 \\ 4.1 \\ 2.1 \end{pmatrix}} - \overset{XW}{\begin{pmatrix} w_0 + w_1 \\ w_0 + 3w_1 \\ w_0 + 2w_1 \\ w_0 + 5w_1 \\ w_0 + 4w_1 \end{pmatrix}} = \overset{Y - XW}{\begin{pmatrix} 1 - (w_0 + w_1) \\ 1.9 - (w_0 + 3w_1) \\ 1.05 - (w_0 + 2w_1) \\ 4.5 - (w_0 + 5w_1) \\ 2.1 - (w_0 + 4w_1) \end{pmatrix}}$$

$\underset{1\times 5}{(Y - XW)^T} \underset{5 \times 1}{(Y - XW)} = \boxed{\overset{1 \times 1}{\sum_{i=1}^{n} (y_i - w_0 + w_1 x_i)^2}}$

$\min (y - XW)(y - XW)$

$\frac{\partial (y - XW)^2}{\partial w} = 0$

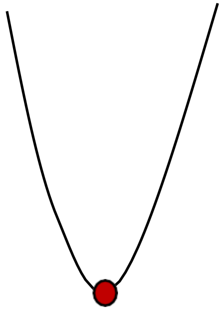
$(Y - XW)^T (Y - XW)$ scalar

$= \underset{1\times 5}{Y^T} \underset{5 \times 1}{Y} - \underset{1 \times 5}{Y^T} \underbrace{\underset{5\times 2}{X} \underset{2 \times 1}{W}} - \underset{1\times 2}{W^T} \underset{2 \times 5}{X^T} \underset{5\times 1}{Y} + \underset{1\times 2}{W^T} \underset{2\times 5}{X^T} \underset{5 \times 2}{X} \underset{2\times 1}{W}$

$\frac{\partial}{W}$ vector

$X^T X = A$

Least square estimation

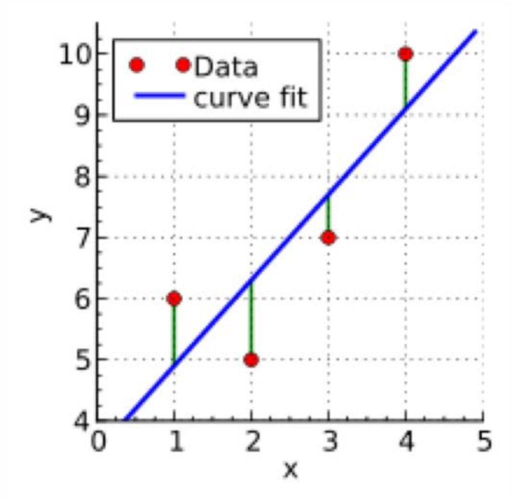


$$S_{training} = \{(x_i, y_i), i = 1..n\} \quad y_i \in \mathcal{R}$$

Obtain/train: $f(x, W) = w_0 + w_1 x$

$$W^* = \arg\, min_W \sum_i (\mathbf{x}_i^T W - y_i)^2$$

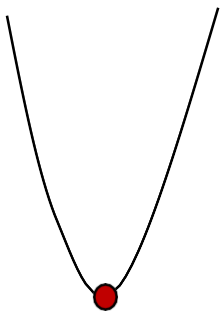
$$W = \begin{pmatrix} w_0 \\ w_1 \end{pmatrix} \quad \mathbf{x}_i = \begin{pmatrix} 1 \\ x_i \end{pmatrix}$$



Let: $X = (\mathbf{x}_1, ..., \mathbf{x}_n)^T \quad Y = (y_1, ..., y_n)^T$

$$W^* = \arg\, min_W = \arg\, min_W L(W) = (XW - Y)^T (XW - Y)$$

Least square estimation



Obtain/train: $f(x, W) = w_0 + w_1 x$

$$W^* = \arg min_W \sum_i (\mathbf{x}_i^T \cdot W - y_i)^2$$

$$W = \begin{pmatrix} w_0 \\ w_1 \end{pmatrix} \quad \mathbf{x}_i = \begin{pmatrix} 1 \\ x_i \end{pmatrix}$$

$$W^* = \arg min_W = \arg min_W L(W) = (XW - Y)^T (XW - Y)$$

$$L(W) = W^T X^T X W - W^T X^T Y - Y^T X W + Y^T Y$$

$$\frac{dL(W)}{dW} = 2X^T X W - 2X^T Y = 0$$

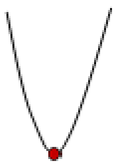
$$W^* = (X^T X)^{-1} X^T Y$$

In [ ]:

For simplification $A = (X^T X)$

```
import numpy as np
from numpy.linalg import inv
# Comput X^T X and denote it as A
A = np.dot(np.transpose(X), X)
# Obtain optimal W
W = np.dot(inv(A),np.dot(np.transpose(X),Y))
```

Least square estimation



Obtain/train: $f(x, W) = w_0 + w_1 x$

$W^* = \arg min_W \sum_i (\mathbf{x}_i^T \cdot W - y_i)^2$

$W = \begin{pmatrix} w_0 \\ w_1 \end{pmatrix} \quad \mathbf{x}_i = \begin{pmatrix} 1 \\ x_i \end{pmatrix}$

$$W^* = \arg min_W = \arg min_W L(W) = \underline{(XW - Y)^T(XW - Y)}$$

$$L(W) = W^T X^T X W - W^T X^T Y - Y^T X W + Y^T Y$$

$$\frac{dL(W)}{dW} = 2X^T X W - 2X^T Y = 0$$

$$W^* = (X^T X)^{-1} X^T Y$$

In [ ]:

For simplification $A = (X^T X)$

```
import numpy as np
from numpy.linalg import inv
# Comput X^T X and denote it as A
A = np.dot(np.transpose(X), X)
# Obtain optimal W
W = np.dot(inv(A),np.dot(np.transpose(X),Y))
```


convex (quadratic)
differentiable

$X^T X = A$

$2X^T X W - 2X^T Y = 0$

given ? given given "ground truth"

$X^T X W = X^T Y$

$W = (X^T X)^{-1} X^T Y$

Matrix calculus

Condition	Expression	Numerator layout, i.e. by $\mathbf{x}^\top$; result is row vector	Denominator layout, i.e. by $\mathbf{x}$; result is column vector
a is not a function of **x**	$\frac{\partial(\mathbf{a}\cdot\mathbf{x})}{\partial\mathbf{x}} = \frac{\partial(\mathbf{x}\cdot\mathbf{a})}{\partial\mathbf{x}} =$ $\frac{\partial\mathbf{a}^\top\mathbf{x}}{\partial\mathbf{x}} = \frac{\partial\mathbf{x}^\top\mathbf{a}}{\partial\mathbf{x}} =$	$\mathbf{a}^\top$	$\mathbf{a}$
A is not a function of **x** **b** is not a function of **x**	$\frac{\partial\mathbf{b}^\top\mathbf{A}\mathbf{x}}{\partial\mathbf{x}} =$	$\mathbf{b}^\top\mathbf{A}$	$\mathbf{A}^\top\mathbf{b}$
A is not a function of **x**	$\frac{\partial\mathbf{x}^\top\mathbf{A}\mathbf{x}}{\partial\mathbf{x}} =$	$\mathbf{x}^\top(\mathbf{A}+\mathbf{A}^\top)$	$(\mathbf{A}+\mathbf{A}^\top)\mathbf{x}$
A is not a function of **x** **A** is symmetric	$\frac{\partial\mathbf{x}^\top\mathbf{A}\mathbf{x}}{\partial\mathbf{x}} =$	$2\mathbf{x}^\top\mathbf{A}$	$2\mathbf{A}\mathbf{x}$
A is not a function of **x**	$\frac{\partial^2\mathbf{x}^\top\mathbf{A}\mathbf{x}}{\partial\mathbf{x}^2} =$	$\mathbf{A}+\mathbf{A}^\top$	
A is not a function of **x** **A** is symmetric	$\frac{\partial^2\mathbf{x}^\top\mathbf{A}\mathbf{x}}{\partial\mathbf{x}^2} =$	$2\mathbf{A}$	

https://en.wikipedia.org/wiki/Matrix_calculus

Linear Regression with the Least Square Estimation

The basic idea of linear regression is to learn a linear function:

$$f(\mathbf{x}; \mathbf{w}, b) = \mathbf{w} \cdot \mathbf{x} + b$$

$\mathbf{w} = \mathbb{R}^m$

$b \in \mathbb{R}$

$\mathbf{x} = \mathbb{R}^m$

Further: $W = (\mathbf{w}, b)$ since b can be also viewed as a parameter in W when a constant 1 is appended to every $\mathbf{x}$.

$$f(\mathbf{x}; W) = W \cdot \mathbf{x}$$

$$W^* = \arg min_W \sum_i (\mathbf{x}_i^T W - y_i)^2$$

least square estimation

Linear Regression with the Least Square Estimation

$$f(\mathbf{x}; W) = W \cdot \mathbf{x}$$

$$W^* = \arg\, min_W \sum_i (\mathbf{x}_i^T W - y_i)^2$$

least square

$$W^* = \arg\, min_W L(W) = (XW - Y)^T (XW - Y)$$

$L(W)$ is quratic function (convex) with a closed-form solution.

$$L(W) = W^T X^T X W - W^T X^T Y - Y^T X W + Y^T Y$$

$$\frac{dL(W)}{dW} = 2X^T X W - 2X^T Y = 0$$

$$W^* = (X^T X)^{-1} X^T Y$$

Closed-form with an analytic solution.

More Complex Linear Combinations

- Univariate linear regression

Now we extend the basic linear regression into more generalized forms.

- Polynomial Linear Regression

$$\text{Output: } y = w_0 + w_1 x_1 + w_2 x_1^2 + \ldots + w_q x_m^q$$

- Multivariate Linear Regression

$$\text{Output: } y = w_0 + w_1 x_1 + w_2 x_2 + \ldots + w_m x_m$$

Linear in terms of W

The good news is that the estimation function is still linear in terms of W.

$$\text{Output: } y = w_0 + w_1 x_1 + w_2 x_1^2 + ... + w_q x_m^q$$

$$\text{Output: } y = w_0 + w_1 x_1 + w_2 x_2 + ... + w_m x_m$$

Output function: $f(\mathbf{x}; W) = W \cdot \mathbf{x}$

$$\mathbf{x} = \begin{pmatrix} 1 \\ x \\ x^2 \\ .. \\ x^q \end{pmatrix} \qquad \mathbf{x} = \begin{pmatrix} 1 \\ x_1 \\ x_2 \\ .. \\ x_m \end{pmatrix}$$

$$X = (\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n)$$

Estimation in learning:

$$L(W) = W^T X^T X W - W^T X^T Y - Y^T X W + Y^T Y$$

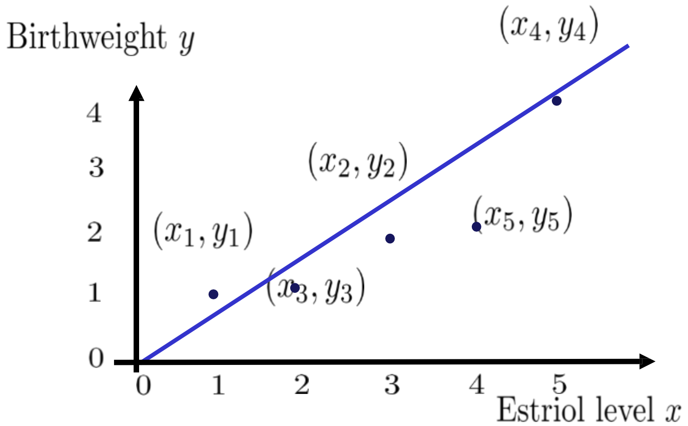
$$\frac{dL(W)}{dW} = 2X^T X W - 2X^T Y = 0$$

Univariate linear regression

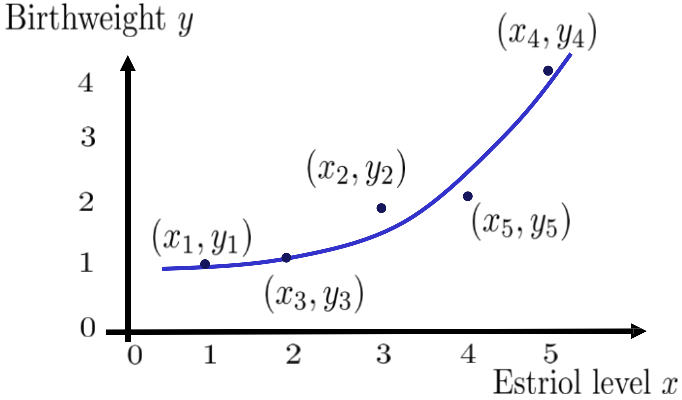
Why is this supervised? How is it labeled?

We are treating the variable we are plotting on the y-axis as a label – as a truth that we want to estimate and predict for new data.

There are different types of regressors, leading to different levels of complexity.



Linear



Polynomial

Polynomial Linear Regression

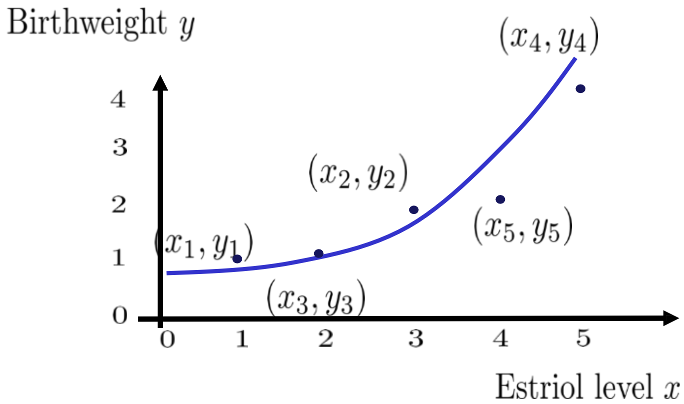
$$S_{training} = \{(x_i, y_i), i = 1..n\}$$

Input: $x,\ x \in R$

Model parameter: $\mathbf{w} = (w_0, w_1, ..., w_d),\ w_i \in R$

Output: $y = w_0 + w_1 x_1 + w_2 x_1^2 + ... + w_q x_m^q$

The combination of terms has your input variable raised to an addition power for each subsequent term.



Polynomial

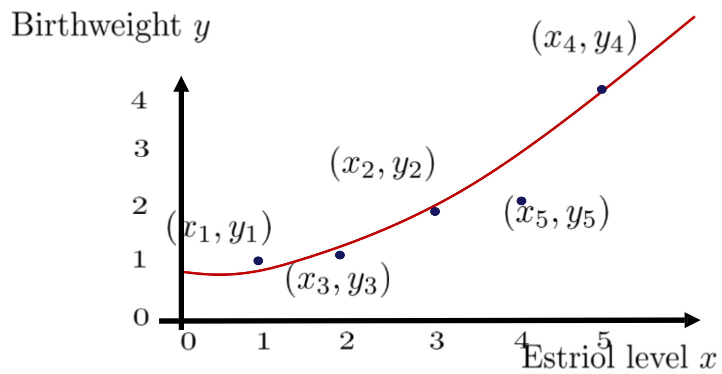
Put Data Into Matrix Form

$$S_{training} = \{(x_i, y_i), i = 1..n\} = \{(1,1), (3,1.9), (2,1.05), (5,4.1), (4,2.1)\}$$

Basic Equations **Matrix Form**

$$y = w_0 + w_1 x + w_2 x^2 \qquad Y = XW$$

$$1 = w_0 + w_1 \times 1 + w_2 \times 1$$
$$1.9 = w_0 + w_1 \times 3 + w_2 \times 9$$
$$1.05 = w_0 + w_1 \times 2 + w_2 \times 4$$
$$4.1 = w_0 + w_1 \times 5 + w_2 \times 25$$
$$2.1 = w_0 + w_1 \times 4 + +w_2 \times 16$$



In python from numpy.linalg import inv

```
W* = np.dot(inv(np.dot(np.transpose(X), X) ),np.dot(np.transpose(X),Y))
```

$$\underset{Y}{\begin{pmatrix} 1 \\ 1.9 \\ 1.05 \\ 4.1 \\ 2.1 \end{pmatrix}} = \underset{X}{\begin{pmatrix} 1,1,1 \\ 1,3,9 \\ 1,2,4 \\ 1,5,25 \\ 1,4,16 \end{pmatrix}} \underset{W}{\begin{pmatrix} w_0 \\ w_1 \\ w_2 \end{pmatrix}}$$

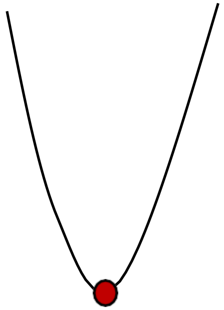
$$\begin{pmatrix} 1.48 \\ -0.67 \\ 0.2321 \end{pmatrix}$$

$$e_{training}(\mathbf{w}^*) = 0.063$$

$$e_{training}(\mathbf{w}_a) = 9.62$$
$$e_{training}(\mathbf{w}_b) = 0.54$$
$$e_{training}(\mathbf{w}^*) = 0.21$$

previous linear model

Quadratic function: least square estimation



$$S_{training} = \{(x_i, y_i), i = 1..n\} \quad y_i \in \mathcal{R}$$

Obtain/train: $f(x, W) = w_0 + w_1 x + w_2 x^2$

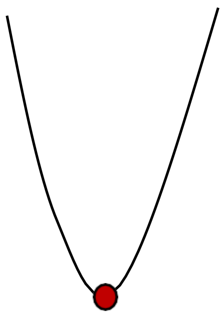
$$W^* = \arg min_W \sum_i (\mathbf{x}_i^T W - y_i)^2$$

$$W = \begin{pmatrix} w_0 \\ w_1 \\ w_2 \end{pmatrix} \quad \mathbf{x}_i = \begin{pmatrix} 1 \\ x_i \\ x_i^2 \end{pmatrix}$$

Let: $X = (\mathbf{x}_1, ..., \mathbf{x}_n)^T \quad Y = (y_1, ..., y_n)^T$

$$W^* = \arg min_W = \arg min_W L(W) = (XW - Y)^T (XW - Y)$$

Quadratic function: least square estimation



Obtain/train: $f(x, \mathbf{w}) = w_0 + w_1 x + w_2 x^2$

$W^* = \arg min_W \sum_i (\mathbf{x}_i^T \cdot W - y_i)^2$

$$W = \begin{pmatrix} w_0 \\ w_1 \\ w_2 \end{pmatrix} \quad \mathbf{x}_i = \begin{pmatrix} 1 \\ x_i \\ x_i^2 \end{pmatrix}$$

$$W^* = \arg min_W = \arg min_W L(W) = (XW - Y)^T (XW - Y)$$

$$L(W) = W^T X^T X W - W^T X^T Y - Y^T X W + Y^T Y$$

$$\frac{dL(W)}{dW} = 2X^T X W - 2X^T Y = 0$$

$$W^* = (X^T X)^{-1} X^T Y$$

In [ ]:

For simplification $A = (X^T X)$

```
import numpy as np
from numpy.linalg import inv
# Comput X^T X and denote it as A
A = np.dot(np.transpose(X), X)
# Obtain optimal W
W = np.dot(inv(A),np.dot(np.transpose(X),Y))
```

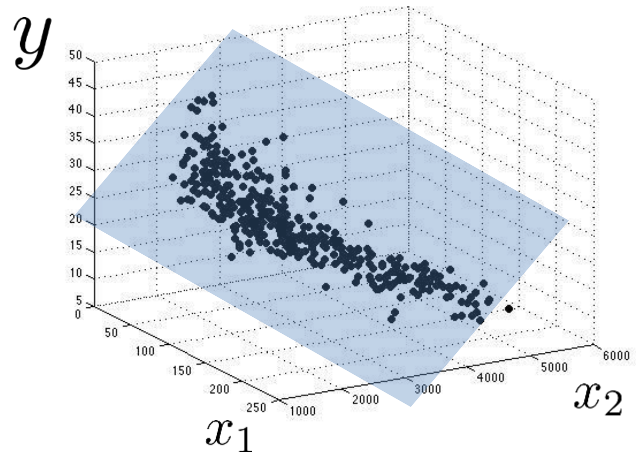
More Complex Linear Combinations

- Univariate linear regression

- Polynomial Linear Regression

- Multivariate Linear Regression

Multi-variate Linear Regression



Input: $\mathbf{x} = (x_1, ..., x_m),\ x_i \in R$

Model parameter: $\mathbf{w} = (w_0, w_1, ..., w_m),\ w_i \in R$

Output: $y = w_0 + w_1 x_1 + w_2 x_2 + ... + w_m x_m$

We apply do the least squre fitting method again.

Put Data Into Matrix Form

$$S_{training} = \{((x_{i1}, x_{i2}), y_i), i = 1..n\}$$
$$= \{((1, 0.5), 1), ((3, 0.9), 1.9), ((2, 1.0), 1.05), ((5, 6.7), 4.1), ((4, 2.5), 2.1)\}$$

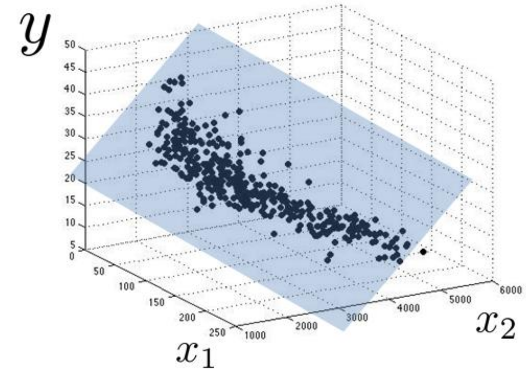
Basic Equations

$$y = w_0 + w_1 x_1 + w_2 x_2$$

$$1 = w_0 + w_1 \times 1 + w_2 \times 0.5$$
$$1.9 = w_0 + w_1 \times 3 + w_2 \times 0.9$$
$$1.05 = w_0 + w_1 \times 2 + w_2 \times 1.0$$
$$4.1 = w_0 + w_1 \times 5 + w_2 \times 6.7$$
$$2.1 = w_0 + w_1 \times 4 + w_2 \times 2.5$$

Matrix Form

$$Y = XW$$



$$\underset{Y}{\begin{pmatrix} 1 \\ 1.9 \\ 1.05 \\ 4.1 \\ 2.1 \end{pmatrix}} = \underset{X}{\begin{pmatrix} 1, 1, 0.5 \\ 1, 3, 0.9 \\ 1, 2, 1.0 \\ 1, 5, 6.7 \\ 1, 4, 2.5 \end{pmatrix}} \underset{W}{\begin{pmatrix} w_0 \\ w_1 \\ w_2 \end{pmatrix}}$$

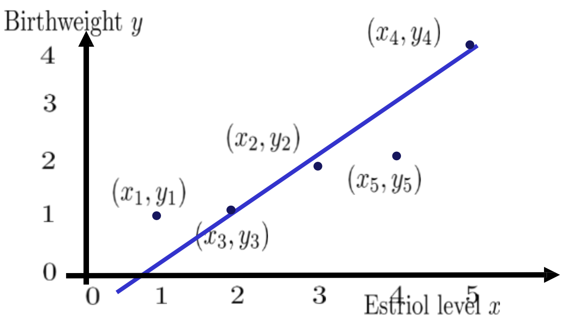
$$W^* = (X^T X)^{-1} X^T Y$$

$$\begin{pmatrix} 0.482 \\ 0.2552 \\ 0.338 \end{pmatrix}$$

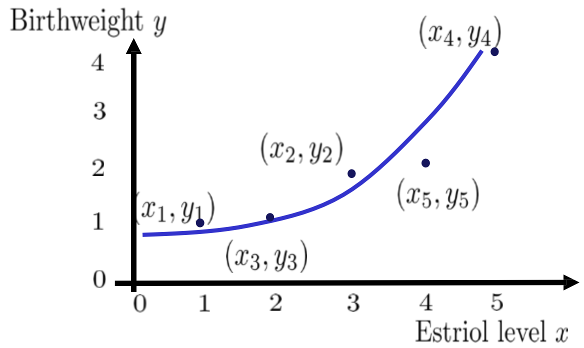
In python from numpy.linalg import inv

```
W* = np.dot(inv(np.dot(np.transpose(X), X) ),np.dot(np.transpose(X),Y))
```

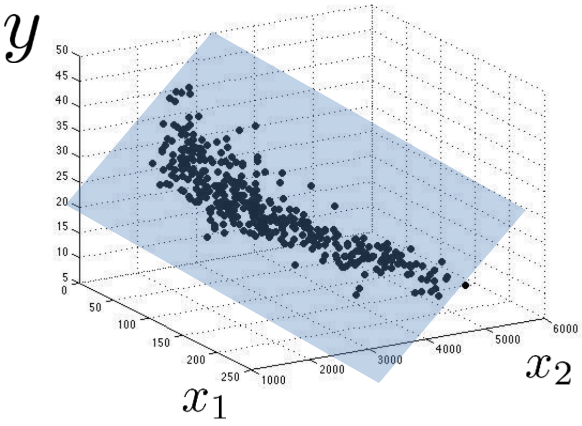
Select your model



Linear



Polynomial



$$e_{training}(\mathbf{w}^*) = 0.21$$

$$e_{training}(\mathbf{w}^*) = 0.063$$

$$e_{training}(\mathbf{w}^*) = 0.052$$

Conclusions for linear regression with the least square estimation

Linear regression:

- Univariate linear regression

Output: $y = w_0 + w_1 x_1$

- Polynomial linear regression

Output: $y = w_0 + w_1 x_1 + w_2 x_1^2 + \ldots + w_q x_m^q$

- Multivariate linear regression

Output: $y = w_0 + w_1 x_1 + w_2 x_2 + \ldots + w_m x_m$

They all share a general form (when generalizing X):

$$Y = X W$$

linear w.r.t. the W!

With an analytical solution:

$$W^* = (X^T X)^{-1} X^T Y$$

Conclusions for linear regression with the least square estimation

Linear regression:

They all share a general form (when generalizing X):

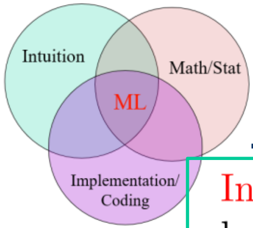
$$Y \quad = \quad X \quad W$$

Linear regression is the definition for the estimation function.

With an analytical solution:

$$W^* = (X^T X)^{-1} X^T Y$$

Least square estimation is about the estimation method. We can use methods other than the least square estimation to solve the linear regression problem.



Recap: Linear Regression

Intuition: Linear (polynomial) regression is one of the most widely used machine learning models. Typically, a squared loss is adopted in training to minimize the averaged difference between the ground-truth values and model predictions for all the input data samples. In this case, the loss/objective function is in a quadratic (convex) form w.r.t. the model parameters, hence a convex function with a unique closed-form (analytical) solution by setting the gradient of the loss function to be zero. The learned model predicts a real number (e.g. length, price, weight) for a given input.

Math:

$$\overset{Y}{\begin{pmatrix} 1 \\ 1.9 \\ 1.05 \\ 4.1 \\ 2.1 \end{pmatrix}} = \overset{X}{\begin{pmatrix} 1,1,0.5 \\ 1,3,0.9 \\ 1,2,1.0 \\ 1,5,6.7 \\ 1,4,2.5 \end{pmatrix}} \overset{W}{\begin{pmatrix} w_0 \\ w_1 \\ w_2 \end{pmatrix}}$$

$$W^* = \arg min_W = \arg min_W L(W) = (XW - Y)^T (XW - Y)$$

$$L(W) = W^T X^T XW - W^T X^T Y - Y^T XW + Y^T Y$$

$$\frac{dL(W)}{dW} = 2X^T XW - 2X^T Y = 0$$

$$W^* = (X^T X)^{-1} X^T Y$$

Implementation:

For simplification

$A = (X^T X)$

In [ ]:

```
import numpy as np
from numpy.linalg import inv
# Comput X^T X and denote it as A
A = np.dot(np.transpose(X), X)
# Obtain optimal W
W = np.dot(inv(A),np.dot(np.transpose(X),Y))
```

Robust Estimation

Estimation and optimization

$$S_{training} = \{(\mathbf{x}_i, y_i), i = 1..n\}$$

Different choices of the penalty will lead to different robustness measure:

L2 norm:

$$e = \sum_{i=1}^{n} (y_i - f(\mathbf{x}_i; \theta))^2$$

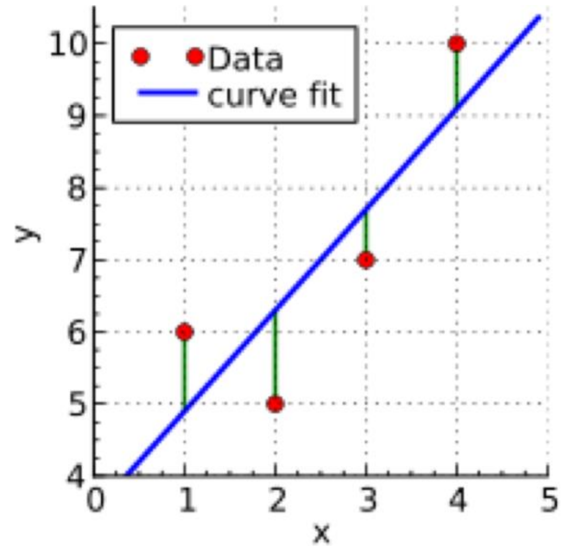
L1 norm:

$$e = \sum_{i=1}^{n} |y_i - f(\mathbf{x}_i; \theta)|$$

Estimation and optimization

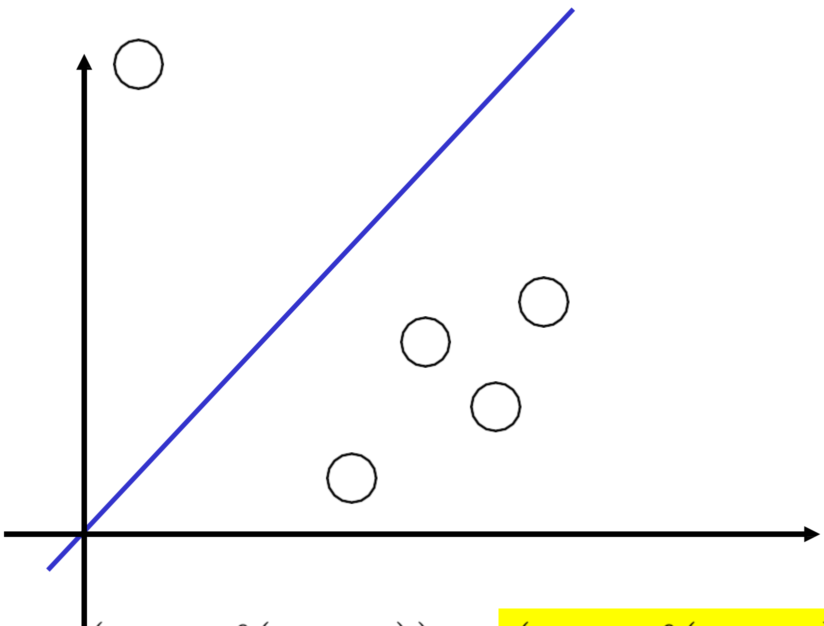
$$S_{training} = \{(\mathbf{x}_i, y_i), i = 1..n\}$$

$$cost(y_i - f(\mathbf{x}_i;\theta)) = (y_i - f(\mathbf{x}_i;\theta))^2$$

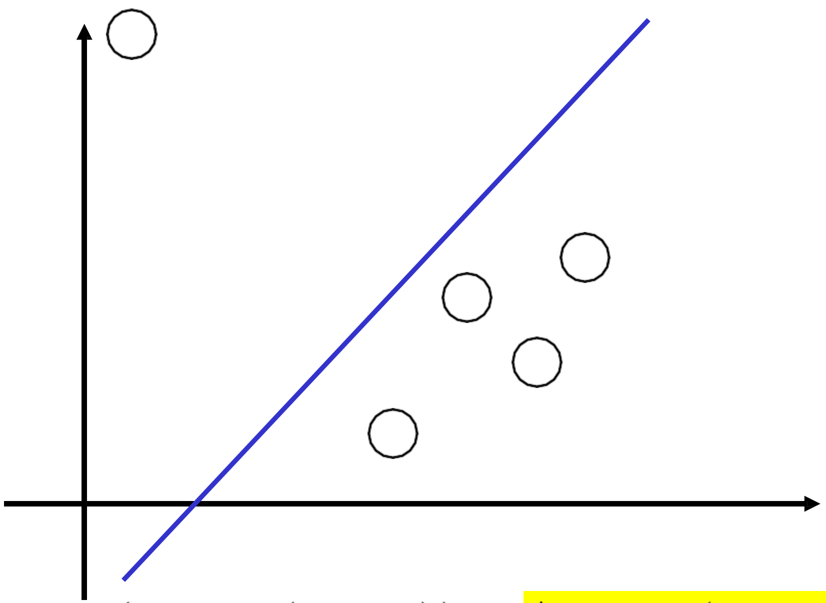


A general form:

$$\mathbf{w} = \arg\min_\theta \sum_{i=1}^{m} cost(y_i - f(\mathbf{x}_i;\mathbf{w}))$$



$$cost(y_i - f(\mathbf{x}_i;\mathbf{w})) = (y_i - f(\mathbf{x}_i;\mathbf{w}))^2$$



$$cost(y_i - f(\mathbf{x}_i;\mathbf{w})) = |y_i - f(\mathbf{x}_i;\mathbf{w})|$$

L1

$$S_{training} = \{(x_i, y_i), i = 1..n\}$$

E.g. $S_{training} = \{(x_i, y_i), i = 1..n\}$
$= \{(1,1), (3,1.9), (2,1.05), (5,4.1), (4,2.1)\}$

$$X = \begin{pmatrix} 1,1 \\ 1,3 \\ 1,2 \\ 1,5 \\ 1,4 \end{pmatrix} \qquad Y = \begin{pmatrix} 1 \\ 1.9 \\ 1.05 \\ 4.1 \\ 2.1 \end{pmatrix}$$

Obtain/train: $f(x, \mathbf{w}) = w_0 + w_1 x$

$$W^* = \arg min_W \sum_{i=1}^{n} |\mathbf{x}_i^T \cdot W - y_i|$$

$$W = \begin{pmatrix} w_0 \\ w_1 \end{pmatrix} \qquad \mathbf{x}_i = \begin{pmatrix} 1 \\ x_i \end{pmatrix}$$

1. Loss (Cost) Function

$$L(W) = \sum_{i=1}^{n} |\mathbf{x}_i^T W - y_i|$$

2. Obtain the gradient

$$\frac{\partial L(W)}{\partial W} = \sum_{i=1}^{n} sign(\mathbf{x}_i^T W - y_i) \times \mathbf{x}_i$$

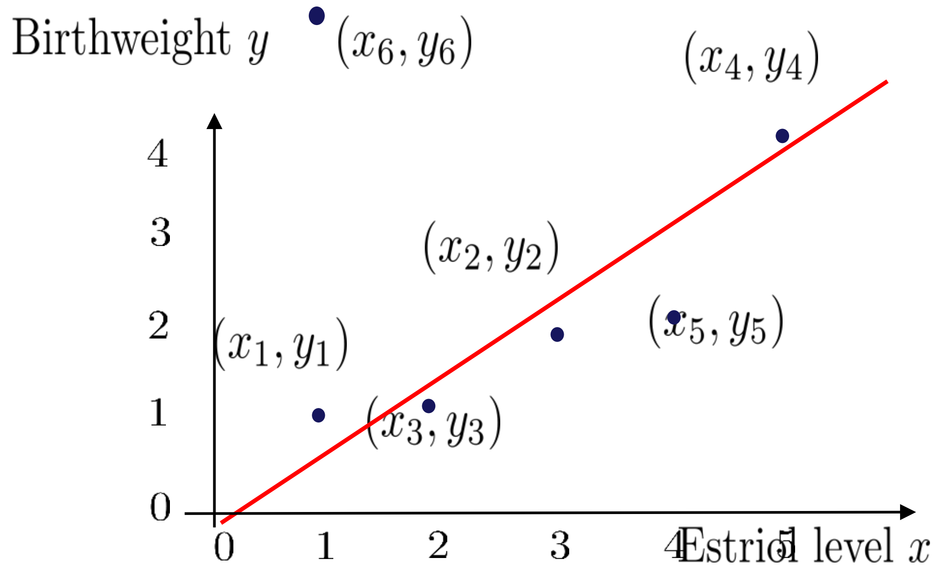
3. Update parameter W

$$W_{t+1} = W_t - \lambda_t \frac{\partial L(W)}{\partial W}$$

Robust estimation

$$Y = X \quad W$$

$$\begin{pmatrix} 1 \\ 1.9 \\ 1.05 \\ 4.1 \\ 2.1 \\ 6.0 \end{pmatrix} \begin{pmatrix} 1, 1 \\ 1, 3 \\ 1, 2 \\ 1, 5 \\ 1, 4 \\ 1, 1.1 \end{pmatrix} \begin{pmatrix} w_0 \\ w_1 \end{pmatrix}$$



$$W^* = (X^T X)^{-1} X^T Y$$

$$W^* = \arg min_W \sum_i (\mathbf{x}_i^T W - y_i)^2$$

$$W^* = \arg min_W \sum_i |\mathbf{x}_i^T W - y_i|$$

L1 Loss

$$S_{training} = \{(x_i, y_i), i = 1..n\} \qquad y_i \in \mathcal{R}$$

Obtain/train: $f(x, \mathbf{w}) = w_0 + w_1 x$

$W^* = \arg min_W \sum_{i=1}^n |\mathbf{x}_i^T \cdot W - y_i|$ $\qquad W = \begin{pmatrix} w_0 \\ w_1 \end{pmatrix} \quad \mathbf{x}_i = \begin{pmatrix} 1 \\ x_i \end{pmatrix}$

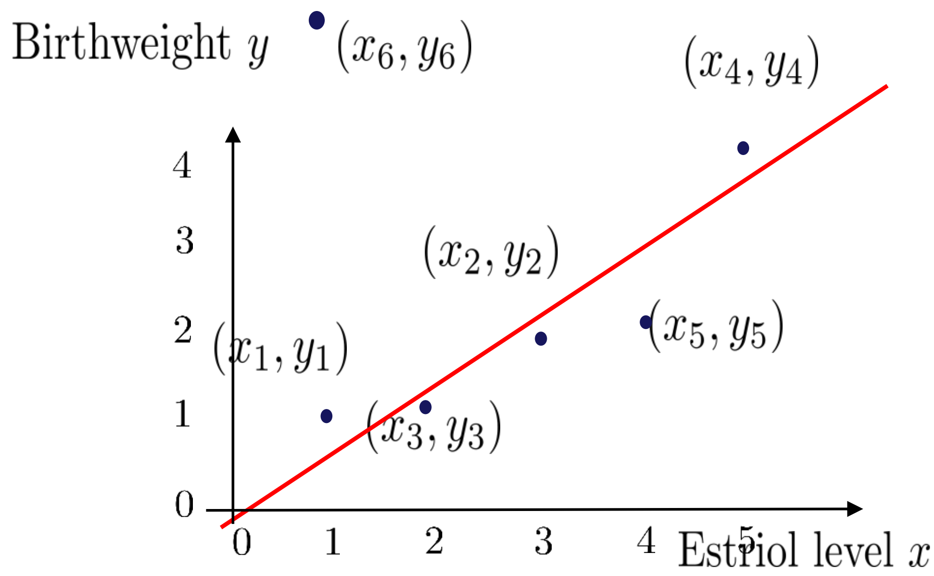
$$\frac{\partial |f(w)|}{\partial w} = \begin{cases} \frac{\partial f(w)}{\partial w} & f(w) > 0 \\ 0 & f(w) = 0 \\ -\frac{\partial f(w)}{\partial w} & otherwise \end{cases}$$

$$= sign(f(w)) \cdot \frac{\partial f(w)}{\partial w} \qquad sign(z) = \begin{cases} +1 & z > 0 \\ 0 & z = 0 \\ -1 & otherwise \end{cases}$$

1. Loss (Cost) Function $\quad L(W) = \sum_{i=1}^n |\mathbf{x}_i^T W - y_i|$
2. Obtain the gradient $\quad \frac{\partial L(W)}{\partial W} = \sum_{i=1}^n sign(\mathbf{x}_i^T W - y_i)\mathbf{x}_i$
3. Update parameter W $\quad W_{t+1} = W_t - \lambda_t \frac{\partial L(W)}{\partial W}$

Robust estimation

$$\begin{array}{cccc} Y & = & X & W \\ \begin{pmatrix} 1 \\ 1.9 \\ 1.05 \\ 4.1 \\ 2.1 \\ \color{red}{6.0} \end{pmatrix} & & \begin{pmatrix} 1,1 \\ 1,3 \\ 1,2 \\ 1,5 \\ 1,4 \\ \color{red}{1,1.1} \end{pmatrix} & \begin{pmatrix} w_0 \\ w_1 \end{pmatrix} \end{array}$$



1. Loss (Cost) Function $\quad W^* = \arg\min_W \sum_i |\mathbf{x}_i^T W - y_i|$
2. Obtain the gradient $\quad \frac{\partial L(W)}{\partial W} = \sum_{i=1}^n sign(\mathbf{x}_i^T W - y_i)\mathbf{x}_i$
3. Update parameter W $\quad W_{t+1} = W_t - \lambda_t \frac{\partial L(W)}{\partial W}$