# [FIN414 Machine Learning Algorithms]

## Supervised Learning

# Regression

**Jae Yun JUN KIM** and **Yves RAKOTONDRATSIMBA**

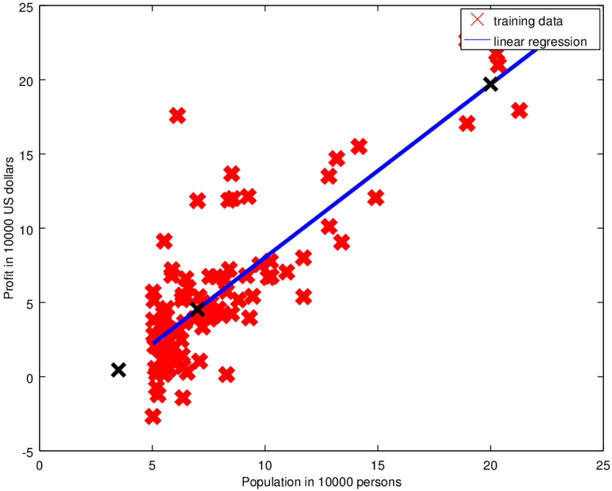
**ECE Paris, Graduate School of Engineering**
37, quai de Grenelle 75015 Paris, France.

September 21, 2016

## Supervised Learning

# Linear regression

# Motivation


training data
linear regression
Profit in 10000 US dollars
Population in 10000 persons
25
20
15
10
5
0
-5
0
5
10
15
20
25

# Notation

$N$ = number of training examples

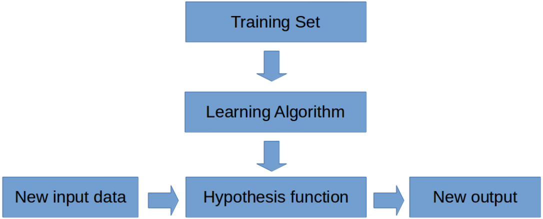
$x$ = input variables / features

$y$ = output variable(s) / "target" variable(s)

$(x, y)$- training example

For example, $i^{th}$ example: $(x^{(i)}, y^{(i)})$.

# Learning Scheme




Hypothesis function:

$$h(x) = h_\beta(x) = \beta_0 + \beta_1 x_1 + \cdots + \beta_p x_p$$

For conciseness, define $x_0 = 1$. Then

$$h_\beta(x) = \sum_{j=0}^{p} \beta_j x_j = \beta^T x$$

where $p$ is the number of features, and $\beta$ are the **model parameters**.

# Goal of regression problems

The goal of regression problems is to find the parameter values (i.e., $\beta$) that minimize the error between $y$ and $h_\beta(x)$ for all the training examples. One possible definition of error can be the following

$$J(\beta) = \frac{1}{2}\sum_{i=1}^{N}\left(h_\beta\left(x^{(i)}\right) - y^{(i)}\right)^2.$$

Hence, a *regression problem* can be mathematically defined as

$$\min_\beta J(\beta) = \min_\beta \frac{1}{2}\sum_{i=1}^{N}\left(h_\beta\left(x^{(i)}\right) - y^{(i)}\right)^2.$$

But, how can this problem be solved?

Method 1: Batch gradient descent

Method 2: Stochastic gradient descent

Method 3: Closed-form solution

# Method 1: Batch Gradient Descent

The idea is

1. Initialize $\beta$ with some values (say $\beta = \overrightarrow{0}$).
2. Keep changing $\beta$ to reduce $J(\beta)$ with the following update rule:
$$\beta_j := \beta_j - \alpha \tfrac{\partial}{\partial \beta_j} J(\beta)$$

But, what is $\frac{\partial}{\partial \beta_j} J(\beta)$?

For $i$ fixed (i.e., for a given training example),

$$\begin{aligned}
\frac{\partial}{\partial \beta_j} J(\beta) =& \frac{\partial}{\partial \beta_j} \frac{1}{2} \left( h_\beta(x) - y \right)^2 \\
=& \left( h_\beta(x) - y \right) \frac{\partial}{\partial \beta_j} \left( h_\beta(x) - y \right) \\
=& \left( h_\beta(x) - y \right) \frac{\partial}{\partial \beta_j} \left( \beta_0 x_0 + \cdots + \beta_j x_j + \beta_p x_p - y \right) \\
=& \left( h_\beta(x) - y \right) x_j
\end{aligned}$$

# Method 1: Batch Gradient Descent

Hence, the update rule with a single training example is

$$\beta_j := \beta_j - \alpha \left( h_\beta(x^{(i)}) - y^{(i)} \right) x_j^{(i)}$$

But, we need to do this taking into account for all the training examples,

$$\beta_j := \beta_j - \alpha \sum_{i=1}^{N} \left( h_\beta(x^{(i)}) - y^{(i)} \right) x_j^{(i)}$$

This update rule is known as the **Batch Gradient Descent**.

# Method 2: Stochastic Gradient Descent

But, when $N >> 1$, the *Batch Gradient Descent* method may be very inefficient.

Alternatively, one might be able to use the **Stochastic Gradient Descent**, which can be formulated as follows:

$$
\begin{array}{l}
\text{For } i = 1 \text{ to } N \ \{ \\
\qquad \text{For } i = 1 \text{ to } p \ \{ \\
\qquad\qquad \beta_j := \beta_j - \alpha \left( h_\beta(x^{(i)}) - y^{(i)} \right) x_j^{(i)} \\
\qquad \} \\
\}
\end{array}
$$

This method can allows us to *approximately but more quickly* find the parameters that minimize $J(\beta)$.

Hence, the choice between the two optimization methods depends on the preference between the *accuracy* and the *efficiency*, respectively.

# Method 3: Closed-form solution

Let us define the *gradient* of the function $J(\beta)$ as follows

$$\nabla_\beta J = \begin{bmatrix} \frac{\partial J}{\partial \beta_0} \\ \vdots \\ \frac{\partial J}{\partial \beta_p} \end{bmatrix} \in \mathbb{R}^{(p+1)\times 1} \tag{1}$$

Hence, the gradient descent method can be vectorially expressed as

$$\beta := \beta - \alpha \nabla_\beta J \tag{2}$$

where $\beta \in \mathbb{R}^{(p+1)\times 1}$.

# Method 3: Closed-form solution

In the sequel, some mathematical backgrounds are given to be able to find the closed-form solution for the linear regression problem.

Let $f : \mathbb{R}^{N\times(p+1)} \longrightarrow \mathbb{R}$ (i.e., $f(A) \in \mathbb{R}$, $A \in \mathbb{R}^{N\times(p+1)}$).

The *gradient* of $f$ is defined as

$$\nabla_A f = \begin{bmatrix} \frac{\partial f}{\partial A_{11}} & \cdots & \frac{\partial f}{\partial A_{1(p+1)}} \\ \vdots & \ddots & \vdots \\ \frac{\partial f}{\partial A_{m1}} & \cdots & \frac{\partial f}{\partial A_{N(p+1)}} \end{bmatrix}$$

The *trace* of $A \in \mathbb{R}^{N\times(p+1)}$ is defined as

$$tr\, A = \sum_{i=1}^{p} A_{ii}$$

# Method 3: Closed-form solution

Some facts with respect to the *gradient of f* and the *trace of A* are given as follows (which will be used to obtain the closed-form solution)

$$tr\, AB = tr\, BA$$

$$tr\, ABC = tr\, CAB = tr\, BCA$$

If $f(A) = tr\, AB$, then $\nabla_A tr\, AB = B^T$

If $a \in \mathbb{R}$, then $tr\, a = a$

$$\nabla_A\, tr\, ABA^T C = CAB + C^T AB^T$$

# Method 3: Closed-form solution

Let

$$X = \begin{bmatrix} \left(x^{(1)}\right)^T \\ \vdots \\ \left(x^{(N)}\right)^T \end{bmatrix}, \qquad y = \begin{bmatrix} y^{(1)} \\ \vdots \\ y^{(N)} \end{bmatrix}$$

Then,

$$X\beta = \begin{bmatrix} \left(x^{(1)}\right)^T \beta \\ \vdots \\ \left(x^{(N)}\right)^T \beta \end{bmatrix} = \begin{bmatrix} h_\beta\left(x^{(i)}\right) \\ \vdots \\ h_\beta\left(x^{(N)}\right) \end{bmatrix}$$

and,

$$X\beta - y = \begin{bmatrix} h_\beta\left(x^{(1)}\right) - y^{(1)} \\ \vdots \\ h_\beta\left(x^{(N)}\right) - y^{(N)} \end{bmatrix}$$

# Method 3: Closed-form solution

By recalling that $z^T z = \sum_i z_i^2$,

$$(X\beta - y)^T (X\beta - y) = \sum_{i=1}^{N} \left( h\left(x^{(i)}\right) - y \right)^2 \triangleq 2J(\beta)$$

Because the *regression problem* consists of finding the parameters ($\beta$) that minimize $J(\beta)$, let us impose that

$$\nabla_\beta J(\beta) = \overrightarrow{0},$$

and find an expression for $\beta$ that satisfies this constraint.

# Method 3: Closed-form solution

Let us now develop the expression of $\nabla_\beta J(\beta)$.

$$\begin{aligned}
\nabla_\beta J(\beta) =& \nabla_\beta \frac{1}{2} (X\beta - y)^T (X\beta - y) \\
=& \frac{1}{2} \nabla_\beta \left( \beta^T X^T X \beta - \beta^T X^T y - y^T X \beta + y^T y \right) \\
=& \frac{1}{2} \nabla_\beta \, tr \, \left( \beta^T X^T X \beta - \beta^T X^T y - y^T X \beta + y^T y \right) \\
=& \frac{1}{2} \left[ \nabla_\beta \, tr \left( \beta \beta^T X^T X \right) - \nabla_\beta \, tr \left( y \beta^T X^T \right) - \nabla_\beta \, tr \left( y^T X \beta \right) \right]
\end{aligned}$$

Using the mathematical facts given on a previous slide, one can show that

$$\nabla_\beta \, tr \left( \beta \beta^T X^T X \right) = X^T X \beta + X^T X \beta$$

$$\nabla_\beta \, tr \left( y \beta^T X^T \right) = X^T y$$

$$\nabla_\beta \, tr \left( y^T X \beta \right) = X^T y$$

# Method 3: Closed-form solution

Therefore,

$$\nabla_\beta J(\beta) = \frac{1}{2}\left(X^T X\beta + X^T X\beta - X^T y - X^T y\right) = X^T X\beta - X^T y$$

Now imposing the condition

$$\nabla_\beta J(\beta) = \overrightarrow{0}$$

we obtain

$$X^T X\beta = X^T y$$

which are known as the **normal equations**.

Finally, the expression of the *parameters* $\beta$ that optimize $J(\beta)$ is

$$\boxed{\beta = \left(X^T X\right)^{-1} X^T y}$$

# Further readings

Lecture Notes (available on campus.ece.fr)

Exercises (available on campus.ece.fr)