


**‘From Statistics to Data Mining’
Computer Lab Session n° 8:
Linear Regression (2/2)**

**Master 1 COSI / CPS²
Saint-Étienne, France**

Fabrice Muhlenbach

Laboratoire Hubert Curien, UMR CNRS 5516
Université Jean Monnet de Saint-Étienne
18 rue du Professeur Benoît Luras
42000 SAINT-ÉTIENNE, FRANCE
<https://perso.univ-st-etienne.fr/muhlfabr/>

Outcome

The objective of this lab is to become familiar with  functions for working with linear regression.

1 Gradient Descent and Closed-Form Solution

How do we choose the parameters θ so that our hypothesis h will make accurate predictions about all the observations?

The objective function is

$$\min_{\theta} = \frac{1}{2} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})^2$$

Let us define $J(\theta) = \min_{\theta} = \frac{1}{2} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})^2$. Therefore, the objective becomes: $\min_{\theta} J(\theta)$.

A way to perform the minimization of $J(\theta)$ that allows you to solve for the parameters θ 's in closed-form, without needing to run an iterative algorithm.

A mathematical problem is said to have a **closed-form** solution if, and only if, at least one solution of that problem can be expressed analytically in terms of a finite number of certain “well-known” functions

To solve the linear regression problem in closed form, we are going to make use of the properties on the trace of a matrix, where if A is a $n \times n$ square matrix, then the trace $\text{tr}(A) = \sum_{i=1}^n a_{ii}$ (or the sum of the eigenvalues).

The closed-form solution is given by

$$\theta = (X^T X)^{-1} X^T y$$

Imagine that we want to optimize the function

$$f(x) = 1.2(x - 2)^2 + 3.2$$

The gradient function is

$$f'(x) = 1.2 \times 2(x - 2) = 2.4(x - 2)$$

If $x = 2$ then $2.4(2 - 2) = 0$, so the actual solution we will approximate with gradient descent is $x = 2$.

In a loop of 100 iterations, we will be able to converge on the solution.

In this example, the learning rate $\alpha = 1$. Try with other values (greater or smaller).

```
# Create some values
xs <- seq(0,4, len=20)

# Define the function we want to optimize
f <- function(x)
{
  1.2 * (x-2)^2 + 3.2
}

# Plot the function
plot(xs, f(xs), type="l", xlab="x",
      ylab=expression(1.2(x-2)^2 + 3.2))

# calculate the gradient df/dx
grad <- function(x)
{
  1.2*2*(x-2)
}

lines(c(2,2), c(3,8), col="red", lty=2)
text(2.1,7, "Closedform_solution", col="red", pos=4)

# Gradient descent implementation
# Initialize the first guess for x-value
x <- 0.1

# Store x-values for graphing purposes (initial)
xtrace <- x

# Store y-values (function evaluated at x)
# for graphing purposes (initial)
ftrace <- f(x)
alpha <- 0.6 # learning rate 'alpha'
for (step in 1:100)
{
  x <- x - alpha*grad(x) # Gradient descent update
  xtrace <- c(xtrace, x) # Update for graph
  ftrace <- c(ftrace, f(x)) # Update for graph
}

lines(xtrace, ftrace, type="b", col="blue")
text(0.5,6, "Gradient_Descent", col="blue", pos=4)

# Print final value of x
print(x)

# Result: x converges to 2.0
```

2 Logistic Regression and Newton's Method

Let's assume that we have a function $f(\theta)$.

We aim at finding a value of θ such that $f(\theta) = 0$.

In the Newton's method (seen in the lesson), one iteration updates θ^t as follows:

$$\theta^{t+1} = \theta^t - \frac{f(\theta^t)}{f'(\theta^t)}$$

When θ is no longer a raw number but a feature vector, we get:

$$\theta^{t+1} = \theta^t - H^{-1} \nabla_{\theta} \ell$$

where:

- H is the Hessian matrix,
- $\nabla_{\theta} \ell$ is the gradient.

The idea of Newton's method is as follows: one starts with an initial guess which is reasonably close to the true root, then the function is approximated by its tangent line (which can be computed using the tools of calculus), and one computes the x -intercept of this tangent line (which is easily done with elementary algebra). This x -intercept will typically be a better approximation to the function's root than the original guess, and the method can be iterated.

Suppose $f : [a, b] \rightarrow \mathbf{R}$ is a differentiable function defined on the interval $[a, b]$ with values in the real numbers \mathbf{R} . The formula for converging on the root can be easily derived. Suppose we have some current approximation x_n . Then we can derive the formula for a better approximation. The equation of the tangent line to the curve $y = f(x)$ at the point $x = x_n$ is $y = f'(x_n)(x - x_n) + f(x_n)$, where f' denotes the derivative of the function f .

The x -intercept of this line (the value of x such that $y = 0$) is then used as the next approximation to the root, x_{n+1} . In other words, setting y to zero and x to x_{n+1} gives $0 = f'(x_n)(x_{n+1} - x_n) + f(x_n)$.

Solving for x_{n+1} gives

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}.$$

With \mathbb{R} , it is easy to calculate the derivatives. For finding the first and the second derivatives of a polynomial function like $5x^3 - 7x^2 - 40x + 100$, we can create some values for x (for the plots), define an expression and obtain the derivatives with the **D** function.

After that, we can evaluate the values for these functions and plot them on the same graph (see Figure 1). We expect that when the initial function decreases, the derivative has negative values, and when the positive function increases, the derivative has positive values.

```
# Create some values for x
x <- seq(-5,5,length.out=200)

# My function is an expression
myFunction <- expression(5 * x^3 - 7 * x^2 - 40 * x + 100)

# First derivative
derivativeF1 <- D(myFunction, 'x') ; print(derivativeF1)

# Second derivative
derivativeF2 <- D(derivativeF1, 'x') ; print(derivativeF2)
```

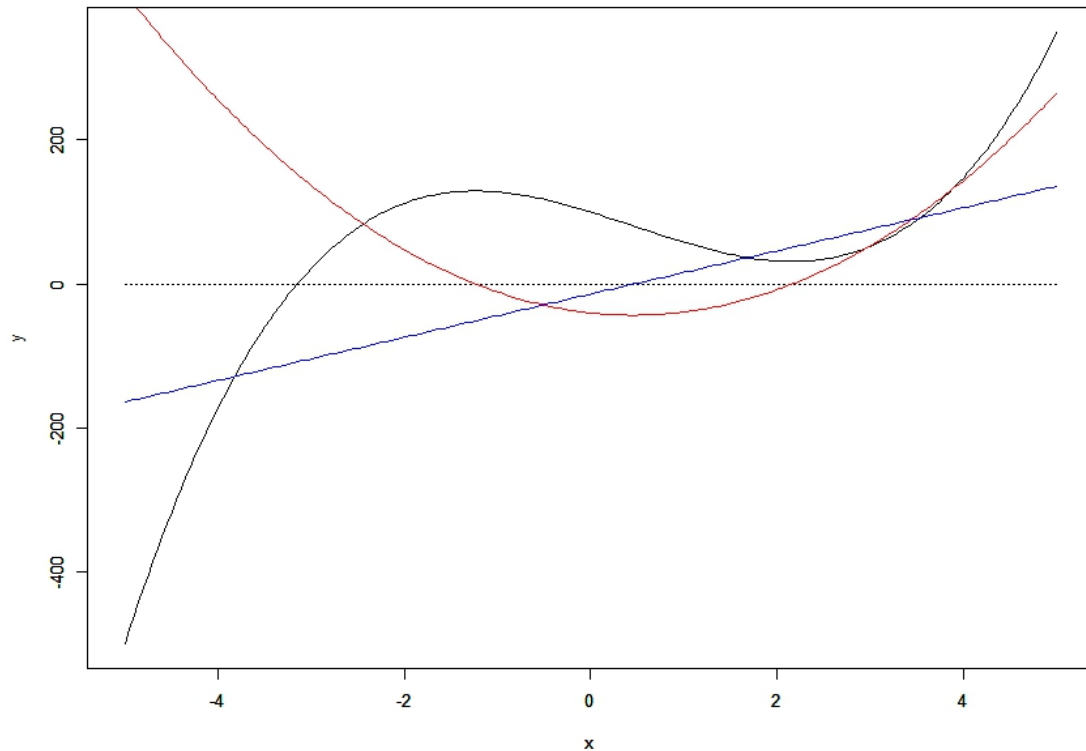


FIG. 1 – Function $f(x) = 5x^3 - 7x^2 - 40x + 100$ in a black curve, its first derivative $f'(x)$ in red, and its second derivative $f''(x)$ in blue.

```
# Plot of my function , the first and the second derivatives
y <- eval(myFunction)
yF1 <- eval(derivativeF1)
yF2 <- eval(derivativeF2)

plot(x, y, type="l")
segments(-5, 0, 5, 0, lty=3)
par(new=TRUE)
plot(x, y = yF1, type="l", col="red", ylim=range(y), ylab="")
par(new=TRUE)
plot(x, y = yF2, type="l", col="blue", ylim=range(y), ylab="")
```

We will install and load the animation package and test Newton's method on some examples: $x^2 - 4 = 0$, $5 * x^3 - 7 * x^2 - 40 * x + 100 = 0$, $\exp(-x)x = 0$ and $\text{atan}(x) = 0$ (the last one will not converge).

After loading the package, we are defining the parameters that control the behaviour of the animation (such as time interval, maximum number of animation frames, height and width, etc.) with `ani.options`.

Then we define the symbol of the chart plot (here, with 20, a small dot).

We can finally plot the Newton's method for a given function, an initial value (starting point), and a range for plotting the curve.

```

install.packages("animation")
library(animation)

oopt = ani.options(interval = 1, nmax = ifelse(interactive(), 50, 2))
par(pch = 20)

## default example
xx = newton.method()
xx$root # solution

## take a long long journey
xx = newton.method(function(x)
  5 * x^3 - 7 * x^2 - 40 * x + 100, 7.15,
  c(-6.2, 7.1))
xx$root

## another function
ani.options(interval = 0.5)
xx = newton.method(function(x) exp(-x) * x, rg = c(0, 10), init = 2)
xx$root

```

For more animations on Newton's method, you can follow the instructions of the help of animation package, page 54:

<https://cran.r-project.org/web/packages/animation/animation.pdf>

3 Recommended Readings

The main literature for this section is:

- Adler (2010), "R in a Nutshell – a Desktop Quick Reference," Chapter 20 "Regression Models".
- Cohen and Cohen (2008), "Statistics and Data with R," Chapter 14 "Simple linear regression".
- Cornillon and Matzner-Løber (2010), « Régression avec R » (in French).
- Dalgaard (2008), "Introductory Statistics with R," Chapter 6 "Regression and correlation."

References

- Adler, J. (2010). *R in a Nutshell – a Desktop Quick Reference*. O'Reilly.
- Cohen, Y. and J. Y. Cohen (2008). *Statistics and Data with R: An Applied Approach Through Examples*. John Wiley & Sons, Ltd.
- Cornillon, P.-A. and E. Matzner-Løber (2010). *Régression avec R*. Pratique R. Springer. [Book available at the library of the Faculty of Science and Technology].
- Dalgaard, P. (2008). *Introductory Statistics with R* (2nd ed.). Springer.