Non-linear regression on dependency trees

Ramon Ferrer-i-Cancho & Marta Arias

Version 0.3
Complex and Social Networks (2014-2015)
Master in Innovation and Research in Informatics (MIRI)

1 Introduction

In this session, we are going to practice on the fit of a non-linear function to data using collections of syntactic dependency trees from different languages. In a syntactic dependency trees, the vertices are the words (*tokens*) of a sentence and links indicate syntactic dependencies between words [Ferrer-i-Cancho, 2013].

n is defined as the number of vertices of a tree. $\langle k^2 \rangle$ is defined as its degree 2nd moment and $\langle d \rangle$ as the mean length of its edges. Through some procedure, two groups of students will be formed:

- The degree 2nd moment group. Its member will have to investigate the scaling of $\langle k^2 \rangle$ as a function of n.
- The mean length group. Its member will have to investigate the scaling of $\langle d \rangle$ as a function of n.

Each member of the group works independently from other members but is allowed to compare results with other members of the group.

2 Data preparation

Download the file `dependency_tree_metrics.tar.gz`, which contains files on information of real syntactic dependency trees from different languages. In each file, each row corresponds to the syntactic dependency tree of a real sentence. Within each row, the 1st, the 2nd and the 3rd cell contain, respectively, n, $\langle k^2 \rangle$ and $\langle d \rangle$ for a given sentence.

Produce an `Rscript` to check the validity of $\langle k^2 \rangle$ and $\langle d \rangle$. In order to be valid, these metrics must satisfy (at least)[Ferrer-i-Cancho, 2013]

$$4 - 6/n \leq \langle k^2 \rangle \leq n - 1 \quad (1)$$

Table 1: Summary of the properties of the degree sequences. N is the sample size (the number of sentences or dependency trees), μ_n and σ_n are, respectively, the mean and the standard deviation of n, the sentence length (n is the number of vertices of a trees), μ_x and σ_x are the mean and the standard deviation of the target metric ($\langle k^2 \rangle$ in one group; $\langle d \rangle$ in the other group)

Language	N	μ_n	σ_n	μ_x	σ_x
...	...	...	...	...	...
...	...	...	...	...	...
...	...	...	...	...	...

and

$$\frac{n}{8(n-1)} \langle k^2 \rangle + \frac{1}{2} \leq \langle d \rangle \leq n - 1 \tag{2}$$

You have to produce a table with the format of Table 1. This table might be necessary to interpret the results that are going to obtained in coming sections. You can adapt the script `summary_table.R` from session 2. The sample size, the mean and then standard deviation can be obtained with the functions `length(...)`, `mean(...)` and `sd(...)`.

3 Data analysis

```
> Catalan = read.table("./data/Catalan_dependency_tree_metrics.txt", header = FALSE)
> colnames(Catalan) = c("vertices","degree_2nd_moment", "mean_length")
> Catalan = Catalan[order(Catalan$vertices), ]
```

loads the information about a collection of dependency trees from sentence in Catalan and sorts the matrix rows by the number of vertices increasingly (that will help later when visualizing the plots).

3.1 Preliminary visualization

Consider the mean dependency length. A preliminary plot can be obtained with

```
> plot(Catalan$vertices, Catalan$mean_length,
       xlab = "vertices", ylab = "mean dependency length")
```

The same plot taking logs on both axes

```
> plot(log(Catalan$vertices), log(Catalan$mean_length),
       xlab = "log(vertices)", ylab = "log(mean dependency length)")
```

suggest a power-law dependency between mean length and number of vertices, in spite of the high dispersion. A clearer intuition about the underlying trend can be obtained by averaging mean lengths for a given number of vertices:

```
> mean_Catalan = aggregate(Catalan, list(Catalan$vertices), mean)
```

Now compare the plot obtained by

```
> plot(mean_Catalan$vertices, mean_Catalan$mean_length,
       xlab = "vertices", ylab = "mean mean dependency length")
```

with

```
> plot(log(mean_Catalan$vertices), log(mean_Catalan$mean_length),
       xlab = "log(vertices)", ylab = "log(mean mean dependency length)")
```

An intuition about how far the real scaling of $\langle d \rangle$ is from the a random linear arrangement can be obtained by adding the expected mean length in that case, which is $E[\langle d \rangle] = (n+1)/3$ [Ferrer-i-Cancho, 2004, Ferrer-i-Cancho, 2004], to the plots. Consider for instance a plot in double logarithmic scale with the averaged curve and the random linear arrangement expectation

```
> plot(log(Catalan$vertices), log(Catalan$mean_length),
       xlab = "vertices", ylab = "mean dependency length")
> lines(log(mean_Catalan$vertices),log(mean_Catalan$mean_length), col = "green")
> lines(log(mean_Catalan$vertices),log((mean_Catalan$vertices+1)/3), col = "red")
```

As for the scaling of $\langle k^2 \rangle$ a suitable null model are uniformly distributed random undirected trees. $f(n)$ can be estimated numerically by producing many of those trees for a given n. The Aldous-Brother algorithm allows one to generate uniformly random labelled spanning trees from a graph [Aldous, 1990, Broder, 1989]. Here we assume a complete graph as the source for the spanning trees. Indeed, it has been shown that

$$\langle k^2 \rangle = \left(1 - \frac{1}{n}\right)\left(5 - \frac{6}{n}\right) \tag{3}$$

for uniformly random labelled trees [Ferrer-i-Cancho, 2014].

An initial exploration of the scaling of $\langle k^2 \rangle$ can be performed with its expected value in uniformly random trees (Eq. 3) and the theoretical lower and upper bounds (Eq. 1):

```
> plot(Catalan$vertices, Catalan$degree_2nd_moment,
       xlab = "vertices", ylab = "degree 2nd moment")
> lines(mean_Catalan$vertices,mean_Catalan$degree_2nd_moment, col = "green")
```

```
> lines(Catalan$vertices,
       (1 - 1/Catalan$vertices)*(5 - 6/Catalan$vertices), col = "red")
> lines(Catalan$vertices,4-6/Catalan$vertices, col = "blue")
> lines(Catalan$vertices,Catalan$vertices-1, col = "blue")
```

The plots suggest that both $\langle d \rangle$ and $\langle k^2 \rangle$ grow sublinearly with n. The plot for $\langle d \rangle$ versus n suggests an almost power-law dependency. The functional dependency between these metrics and n will be investigated next.

3.2 The ensemble of models

We consider an ensemble of models

- $f(n) = (n/2)^b$ (model 1). This models is obtained applying the condition $f(2) = 1$ (satisfied both by $\langle d \rangle$ and $\langle k^2 \rangle$) to a more general model, i.e. $f(n) = an^b$. This leads to $a = 1/2^b$ and finally $f(n) = (n/2)^b$. The motivation of this model is that $\langle k^2 \rangle = \langle d \rangle = 1$ when $n = 2$.
- $f(n) = an^b$ (model 2), a power-law model.
- $f(n) = ae^{cn}$ (model 3), an exponential model.

Additionally, it is convenient to consider a null model (model 0). Concerning the scaling of $\langle d \rangle$, a random linear arrangement of vertices gives $f(n) = n/3 + 1/3$ as the null model. Concerning the scaling of $\langle k^2 \rangle$, uniformly random labelled trees give $f(n) = \left(1 - \frac{1}{n}\right)\left(5 - \frac{6}{n}\right)$ as the null model. Notice that $f(n)$ depends exclusively on n in both null models (no extra parameters are needed).

The additive term of the random linear arrangement model suggests the models above should be generalized giving

- $f(n) = (n/2)^b + d$ (model 1+).
- $f(n) = an^b + d$ (model 2+).
- $f(n) = ae^{cn} + d$ (model 3+).

- Model 2+ and model 3+ turn the non-linear regression problem more complicated.
- For $\langle k^2 \rangle$, models 1-3 and 1+ are mandatory.
- Do not use models 2+ and 3+ for $\langle k^2 \rangle$ till you are requested to do so in Section 7.
- For $\langle d \rangle$, models 1-3, 1+ and 2+ are mandatory.
- Do not use model 3+ for $\langle d \rangle$ till you are requested to do so in Section 7.

4 Non-linear regression with R

We show how to fit model 2 to $\langle d \rangle$ as a function of n. The procedure for $\langle k^2 \rangle$ is the same but replacing `mean_length` by `degree_2nd_moment`. The non-linear regression can be invoked by means of the comment `nls(...)` through,

```
> a_initial = 4
> b_initial = 4
> nonlinear_model = nls(mean_length~a*vertices^b,data=Catalan,
  start = list(a = a_initial, b = b_initial), trace = TRUE)
```

where

- `mean_length~a*vertices^b` is the mathematical definition of the function to fit,
- `data=Catalan` indicates the data source,
- `start = list(a = a_initial, b = b_initial)` defines the initial values of the parameters,
- `trace = TRUE` indicates that the progress of the optimization algorithm must be shown.

We chose initial values for the parameters a and b arbitrarily. The initial values can be crucial to warrant that `nls(...)` is able to find a solution to the optimization problem or simply to increase the time efficiency of the non-linear regression. Try for instance `a_initial = 0` to see an example of failure of `nls(...)`. Good initial values for a and b can be obtained with a double logarithmic transformation, which gives

$$\log \langle d \rangle = b \log n + a', \tag{4}$$

where $a' = \log a$. A linear regression on that transformation allows one to obtain an initial value for b and initial value for a thanks to $a = e^{a'}$. This is what the next code does:

```
> linear_model = lm(log(mean_length)~log(vertices), Catalan)
> a_initial = exp(coef(linear_model)[1])
> b_initial = coef(linear_model)[2]
```

`lm(...)` is used to perform a non linear regression between $\log \langle d^2 \rangle$ and $\log n$. `coef(...)` retrieves the parameters of the linear model: `coef(...)[1]` contains the intercept and `coef(...)[2]` contains the slope.

We run the non-linear regression again with

```
> nonlinear_model = nls(mean_length~a*vertices^b,data=Catalan,
  start = list(a = a_initial, b = b_initial), trace = TRUE)
```

Notice the faster convergence of `nls(...)` with the new initial values of the parameters.

> Notice that a hidden assumption of the way we call `nls(...)` and non-linear regression in general is homocesdasticity. You have to check if this assumption holds. Plots of the original data can help but you may need to calculate the variance of points as a function of the number of vertices of the tree. In case that homocesdasticity does not hold we strongly suggest that `nls(...)` is feeded with the output of `aggregate(...)` and not the original data.

The RSS and the AIC of the nonlinear regression model can be retrieved, respectively, simply with

```
deviance(nonlinear_model)
```

and

```
AIC(nonlinear_model)
```

> It is important to be aware of a crucial difference between this lab session and an Session 2. Here we are following a nonlinear regression approach to curve fitting. Its goal is minimizing the error between the curve and the model (and the AIC is a function of that error). In contrast, session 2 was based on a maximum likelihood approach where there is no direct concern about that error. The AIC is computed differently depending on the approach.

s can be obtained through

```
> sqrt(deviance(nonlinear_model)/df.residual(nonlinear_model))
```

The parameters giving the best fit can be obtained through

```
coef(nonlinear_model)
```

`coef(nonlinear_model)[i]` contains the value of the i-th parameters for the best fit of the model ($i = 1$ for the 1st parameters). The mapping of parameter index to parameter is tricky and thus it is convenient that you use `coef(nonlinear_model)["a"]` to retrieve the value of a giving the best fit and `coef(nonlinear_model)["b"]` to retrieve the value of b giving the best fit.

Notice `nls(...)` cannot be used for models with no parameters. For instance, the RSS and s^2 for $\langle d \rangle = (n+1)/3$ can be calculated with

```
RSS <- sum((tree_metrics$mean_length-(tree_metrics$vertices+1)/3)^2)
```

Language	0	1	2	3	1+	2+	3+
...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...

Table 2: Do not forget to include results for model 0 here

	Model												
	1	2		2		1+		2+			3+		
Language	*b*	*a*	*b*	*a*	*c*	*b*	*d*	*a*	*b*	*d*	*a*	*c*	*d*
...	...	...	...	...	...	...	...	...	...	...	...	...	...

Table 3: Notice that Model 0 is no included here because it has no parameters.

```
n <- length(tree_metrics$vertices)
p <- 0
s <- sqrt(RSS/(n - p))
```

Finally, the corresponding AIC (assuming a non-linear regression model) can be calculated with [Ritz and Streibig, 2008, p. 105; Eq. 7.2]

```
AIC <- n*log(2*pi) + n*log(RSS/n) + n + 2*(p + 1)
```

5 Results

5.1 Model selection

For each metric, you have to prepare:

- Separate tables for s (residual standard error), AIC (Akaike information criterion), Δ (AIC differences) following the format of Table 2.
- Table with the values of the parameters of giving the best fit for each model following the format of Table 3.

5.2 Final visualization

For each language, you have to plot the empirical data and the curve for the best fit. Imagine that model 2 is the best model for the language `Catalan`. Then the data and the best fit can be plotted together with

```
> plot(log(Catalan$vertices), log(Catalan$mean_length),
```

```
       xlab = "log(vertices)", ylab = "log(mean dependency length)")
> lines(log(Catalan$vertices), log(fitted(nonlinear_model)), col = "green")
```

6 Deliverables

You have to prepare a report including the following sections (in this order): introduction, results, discussion and methods. Results includes all the tables and figures (the preliminary plots and the plots of the best model to the real data) and some guiding text. Methods should include any relevant methods not explained in this guide (for instance, decisions that you had to made and might have an influence on the results), initial values of the parameters used to call `nls()`, the techniques used to obtain those initial values and so on... The discussion should include a summary of the results and your interpretation. For instance, you should discuss

- If there is a significance difference between the fit of the functions from null hypotheses and that of alternative hypotheses.
- If the original data satisfy the assumption of homoscedasticity of the non-linear regression methods considered here. In case that it does not hold, you should explain how you have addressed it.
- Discuss if the function giving the best fit gives a reasonably good fit (e.g., checking visually that the best function provides a sufficiently good fit). Remember that the best function of an ensemble is not necessarily the best in absolute terms.
- The extent to which languages resemble or differ.

The discussion section should also include some conclusions.

Important rule: The lab session, and especially the report you have to hand in, are strictly individual work. Plagiarism will be prosecuted. Nevertheless, you are encouraged to ask the teacher as soon as possible if you think you don not understand what you are supposed to do, and also if you feel you are spending much more time than the rest of the group – sometimes a tiny error can be tricky to find and does not add much to your knowledge. Questions can be asked either in person or by email, and you will never be penalized by asking questions, no matter how stupid they look in retrospect.

To deliver: You must deliver the report explained above. The formats accepted for the report are, in principle, pdf, Word, OpenOffice, and Postscript. You also have to hand in the source code in `R`(or other languages) that you have used, including some minimal comments that can help the reader.

Procedure: Submit your work through the raco platform as a single zipped file.

Deadline: Work must be delivered within 2 weeks from the lab session you attend. Late deliveries risk being penalized or not accepted at all. If you anticipate problems with the deadline, please tell us as soon as possible.

7 Advanced exercises

- $\langle d \rangle$ group: add model 3+.
- $\langle k^2 \rangle$ group: add the models 2+ and 3+.
- Both groups: consider adding the following models:
  - $f(n) = an^b e^{cn}$ (model 4), a generalization of model 2 that is widely used in quantitative linguistics.
  - $f(n) = an^b e^{cn} + d$ (model 4+), a generalization of model 4 with an additive term.

The challenge of the advanced exercises may require a careful choice of the initial values of the parameters used to call `nls(...)`, tuning some parameters of `nls(...)` or using `aggregate(...)` to smooth the original dataset.

References

[Aldous, 1990] Aldous, D. (1990). The random walk construction of uniform spanning trees and uniform labelled trees. *SIAM J. Disc. Math.*, 3:450–465.

[Broder, 1989] Broder, A. (1989). Generating random spanning trees. In *Symp. Foundations of Computer Sci., IEEE*, pages 442–447, New York.

[Ferrer-i-Cancho, 2004] Ferrer-i-Cancho, R. (2004). Euclidean distance between syntactically linked words. *Physical Review E*, 70:056135.

[Ferrer-i-Cancho, 2013] Ferrer-i-Cancho, R. (2013). Hubiness, length, crossings and their relationships in dependency trees. *Glottometrics*, 25:1–21.

[Ferrer-i-Cancho, 2014] Ferrer-i-Cancho, R. (2014). A stronger null hypothesis for crossing dependencies. *http://arxiv.org/abs/1410.5485*.

[Ritz and Streibig, 2008] Ritz, C. and Streibig, J. C. (2008). *Nonlinear regression with R*. Springer, New York.