Rapporto n. 194

An algorithm for maximum likelihood estimation of Zipf's Law

Daniele Felletti

Ottobre 2010

Dipartimento di Metodi Quantitativi per le Scienze Economiche ed Aziendali
Università degli Studi di Milano Bicocca
Via Bicocca degli Arcimboldi 8 - 20126 Milano - Italia
Tel +39/02/64483103 - Fax +39/2/64483105
Segreteria di redazione: Andrea Bertolini

An algorithm for maximum likelihood estimation of Zipf's law

Daniele Felletti

Keywords: Zipf law, power law, maximum likelihood, Riemann zeta function

Abstract

Zipf's law, the discrete version of the Power law, emerges in several areas of research, because it is the distribution that reflects the principle of scale invariance. Thus it is typically related to complex systems.

Here an algorithm is proposed to evaluate the parameter estimation of the Zipf's law that best fits a set of data according to the maximum likelihood principle.

1 Introduction

Zipf's law is the discrete counterpart of the power law ($Prob(x > \mu) = \frac{\beta}{\mu^{\alpha-1}}$). It appears a wide range of applications since it characterizes the scale invariance. Another case is when the percentage variation of the probability is linearly related to the percentage variation of the quantity: $\frac{dP}{P} = -\alpha\frac{dx}{x}$. Fields in physics, biology and computer science, but, in particular, in economics and social science show this behavior since complex networks and structured systems are characterized by power law distributions when they have grown up following some preferential attachment dynamics ([1],[2]). See ([6]) for a review on the subject.

From a statistical point of view it is characterized by a "fat tail": no matter α, it tends to 0 more slowly than an exponential, so rare events are more frequent than a Poisson distribution and much more frequent than a Gaussian distribution. When $\alpha \leq 3$ it has an infinite variance, when $\alpha \leq 2$ it neither has a mean value.

A log-log plot shows quite clearly when a power law is present, so, sometimes, a simple least-square estimation is performed to evaluate the model that best fits the data ([7]). Nevertheless this method may lead to inaccurate estimates ([3], [4], [5]). No further statistical considerations in support to one method against the other are reported. Neither the statistical significance (usually χ^2 test or Kolmogorov-Smirnov test) is considered.

To estimate the parameter of the Zipf's law maximizing the likelihood of a given set of data requires the solution of an equation involving the

Riemann's zeta function and its derivative. In this report an algorithm is proposed to numerically solve the equation: the function is bounded by two contiguous classes of functions and the zero is found with the desired approximation.

In the Appendix a C code for the algorithm is reported.

2 Zipf's law

Zipf's law is a one-parameter set of discrete probability distributions defined only on positive, integer numbers. Given the real parameter $\alpha > 1$, the probability of the event $x = n$ is

$$\text{Prob}(x = n) = C_\alpha \, n^{-\alpha} \quad , \quad n \in \mathbb{Z}^+$$

where C_α, the constant of normalization, is the reciprocal of the Riemann zeta function:

$$C_\alpha = \frac{1}{\sum\limits_{j=1}^{+\infty} \frac{1}{j^\alpha}} = \frac{1}{\zeta(\alpha)}$$

For future purposes, let

$$C'_\alpha = \frac{dC_\alpha}{d\alpha} - \frac{\sum\limits_{j=1}^{+\infty} j^{-\alpha} \log{(j)}}{\left(\sum\limits_{j=1}^{+\infty} j^{-\alpha}\right)^2} = -\frac{\zeta'(\alpha)}{\zeta^2(\alpha)}$$

3 Maximum likelihood estimation

Consider a set of K values independently drawn according to a Zipf's law. Let $f_1,\ f_2,\ \ldots,\ f_N$ the frequencies of $1,\ 2,\ \ldots,\ N$. So $K = \sum\limits_{j=1}^{N} f_j$. The probability of the set is

$$P = \prod_{j=1}^{N} \left(C_\alpha \, j^{-\alpha}\right)^{f_j} = C_\alpha^K \prod_{j=1}^{N} j^{-\alpha \, f_j}$$

The value of α that maximizes P maximizes $\log(P)$ too.

$$L = \log{(P)} = K \, \log{(C_\alpha)} - \alpha \sum_{j=1}^{N} f_j \, \log{(j)}$$

Equating the derivative to zero, the following equation must be solved

$$\frac{dL}{d\alpha} = 0 \iff K \frac{C'_\alpha}{C_\alpha} - \sum_{j=1}^{N} f_j \, \log{(j)} = 0 \iff C'_\alpha - A \, C_\alpha = 0$$

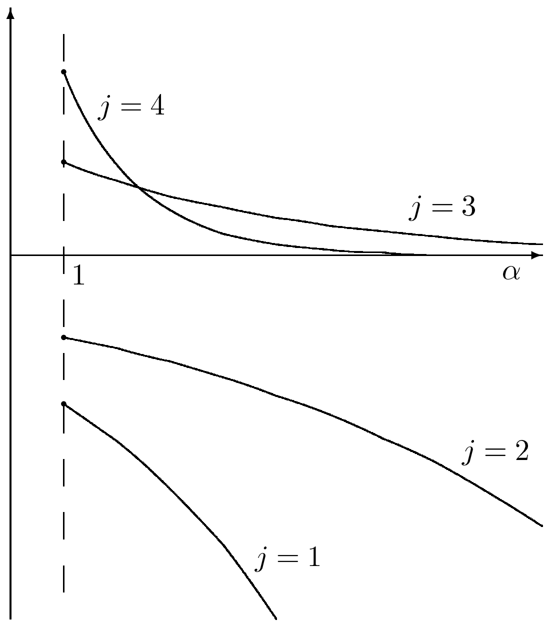

Figure 1: $g_j(\alpha)$ for several values of j (with $e^A = 2.5$).

where

$$A = \frac{1}{K} \sum_{j=1}^{N} f_j \log(j)$$

Note that

$$0 \le A \le \frac{1}{K} \sum_{j=1}^{N} f_j \log(N) = \log N$$

The optimal α must solve

$$\sum_{j=1}^{+\infty} j^{-\alpha} \log(j) - A \sum_{j=1}^{+\infty} j^{-\alpha} = 0 \iff \sum_{j=1}^{+\infty} j^{-\alpha} (\log(j) - A) = 0 \iff$$

$$\iff e^{-\alpha A} \sum_{j=1}^{+\infty} \left(\frac{j}{e^A}\right)^{-\alpha} \log\left(\frac{j}{e^A}\right) = 0 \iff \sum_{j=1}^{+\infty} \left(\frac{j}{e^A}\right)^{-\alpha} \log\left(\frac{j}{e^A}\right) = 0$$

Let

$$\begin{aligned} g_j(\alpha) &= \left(\tfrac{j}{e^A}\right)^{-\alpha} \log\left(\tfrac{j}{e^A}\right) \quad , \quad j \in \mathbf{Z}^+ \\ G(\alpha) &= \sum_{j=1}^{+\infty} \left(\tfrac{j}{e^A}\right)^{-\alpha} \log\left(\tfrac{j}{e^A}\right) = \sum_{j=1}^{+\infty} g_j(\alpha) \end{aligned}$$

- For every j: $g_j(\alpha)$ is a decreasing function with respect to α.

- For $j < e^A$: $g_j(\alpha)$ is negative for every α; $g_j(1) = \left(\frac{e^A}{j}\right) \log\left(\frac{j}{e^A}\right) < 0$; $\lim\limits_{\alpha \to +\infty} g_j(\alpha) = -\infty$.
- For $j > e^A$: $g_j(\alpha)$ is positive for every α; $g_j(1) = \left(\frac{e^A}{j}\right) \log\left(\frac{j}{e^A}\right) > 0$; $\lim\limits_{\alpha \to +\infty} g_j(\alpha) = 0$.

Consequently:

- $G(\alpha)$ exists for $\alpha > 1$.
- $G(\alpha)$ is a decreasing function with respect to α.
- $\lim\limits_{\alpha \to 1^+} G(\alpha) = +\infty$.
- $\lim\limits_{\alpha \to +\infty} G(\alpha) = -\infty$.

So there is just one value for α such that $G(\alpha) = 0$ and, observing the relation between $G(\alpha)$ and $P(\alpha)$, it maximizes the likelihood function.

4 Approximation of $G(\alpha)$

Let

$$G_M(\alpha) = \sum_{j=1}^{M} g_j(\alpha)$$

$G_M(\alpha)$ is decreasing with respect to α (being the sum of decreasing functions) while it is decreasing with respect to M for $M < e^A$ and increasing for $M > e^A$: consider this last case.

$$G(\alpha) = \sum_{j=1}^{M-1} g_j(\alpha) + \sum_{j=M}^{+\infty} g_j(\alpha) = G_M(\alpha) + \sum_{j=M}^{+\infty} g_j(\alpha)$$

The following approximation (Figure 2) is useful

$$\int_{M}^{+\infty} \left(\frac{x}{e^A}\right)^{-\alpha} \log\left(\frac{x}{e^A}\right) dx < \sum_{j=M}^{+\infty} g_j(\alpha) < \int_{M-1}^{+\infty} \left(\frac{x}{e^A}\right)^{-\alpha} \log\left(\frac{x}{e^A}\right) dx \iff$$

$$\iff h(M, \alpha) < \sum_{j=M}^{+\infty} g_j(\alpha) < h(M-1, \alpha)$$

where

$$h(M, \alpha) = \frac{e^A}{\alpha - 1} \left(\frac{M}{e^A}\right)^{1-\alpha} \left(\log\left(\frac{M}{e^A}\right) + \frac{1}{\alpha - 1}\right)$$

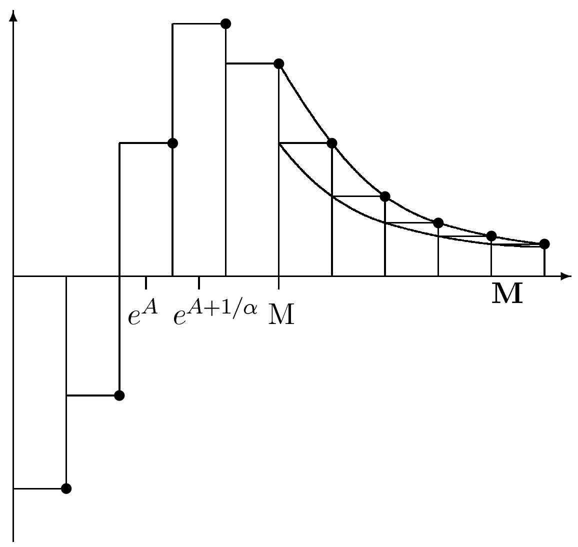

Figure 2: Approximation of $G(\alpha)$.

Since $M > e^A$, $h(M,\alpha)$ is positive and decreasing with respect to both M and α. Then

$$H_1(M,\alpha) = G_M(\alpha) + h(M,\alpha) \quad , \quad H_2(M,\alpha) = G_M(\alpha) + h(M-1,\alpha)$$

are decreasing functions with respect to α. Looking at Figure 2, it gets clear that, when $M > e^{A+\frac{1}{\alpha}}$, $H_1(M,\alpha)$ is an increasing function with respect to M, while $H_2(M,\alpha)$ is decreasing. In particular, this is true for any α when $M > e^{A+1}$.

The following identities are useful to speed up the algorithm:

$$\begin{aligned}
H_2(M,x) &= H_1(M,x) + h(M-1,x) - h(M,x)\\
H_1(M+1,x) &= H_1(M,x) + g_M(x) + h(M+1,x) - h(M,x)\\
H_2(M+1,x) &= H_2(M,x) + g_M(x) + h(M,x) - h(M-1,x)
\end{aligned}$$

5 The algorithm

Chosen the numerical precision ε, the goal is to find two values a and b and a not too large integer value M such that

- $H_1(M,a) \geq 0$ and $H_2(M,b) \leq 0$,
- $b - a < \varepsilon$ or $H_1(M,a) - H_2(M,b) < \varepsilon$.

Here follows the algorithm:

1. Set $M = \mathrm{Int}(e^{A+1}) + 1$ (note that $M \leq N\,e + 1$).

2. Choose an arbitrary starting value for a and repeat $a = \frac{1+a}{2}$ until the condition $Y = H_1(M, a) \geq 0$ is reached (i.e., keep on halving the interval $[1, a]$ until $H_1(M, a)$ gets positive).

3. Choose an arbitrary starting value for b and repeat $b = 2b$ until the condition $y = H_2(M, b) \leq 0$ is reached (i.e., keep on doubling b until $H_2(M, b)$ gets negative).

4. If $b - a < \varepsilon$ or $Y - y < \varepsilon$
   - then stop and take $\alpha = \frac{a+b}{2}$.
   - else take $x = \frac{a+b}{2}$.

5. Increase M until one of the following conditions is reached:
   (a) $H_1(M, x) \geq 0$.
   (b) $H_2(M, x) \leq 0$.
   (c) $H_2(M, x) - H_1(M, x) < \varepsilon$.

6. According to the previous condition do:
   (a) $a = x$, $Y = H_1(M, x)$, $y = H_2(M, b)$. Go to step 4.
   (b) $b = x$, $y = H_2(M, x)$, $Y = H_1(M, a)$. Go to step 4.
   (c)
   - Set $x_1 = x$. Iterate either $a = \frac{a+x_1}{2}$ (if $H_1\left(M, \frac{a+x_1}{2}\right) > 0$) or $x_1 = \frac{a+x_1}{2}$ (if $H_1\left(M, \frac{a+x_1}{2}\right) < 0$) until $H_1(M, a) \leq H_1(M, x)$ is got.
   - Set $x_2 = x$. Iterate either $x_2 = \frac{x_2+b}{2}$ (if $H_2\left(M, \frac{x_2+b}{2}\right) > 0$) or $b = \frac{x_2+b}{2}$ (if $H_2\left(M, \frac{x_2+b}{2}\right) < 0$) until $H_2(M, b) \geq H_2(M, x)$ is got.
   - Stop and take $\alpha = \frac{a+b}{2}$.

The solution α has an accuracy $\frac{b-a}{2}$.

6 Some tests

The following results have been computed on a PC Pentium III 500MHz, ram 256MB, with OS Linux Ubuntu 2.6.17.

An accuracy of 10^{-5} has been chosen. Computational times are of order of tenths of seconds (neglecting the time spent for the data input and the computation of the parameter A).

First of all, the optimal value α^* versus the parameter $e^A \in [0, N]$ has been plot (Figure 3). It proves the quality of the algorithm.

A more interesting point is the stopping value of M. This value is related to the computational time, but it is not directly proportional since only the

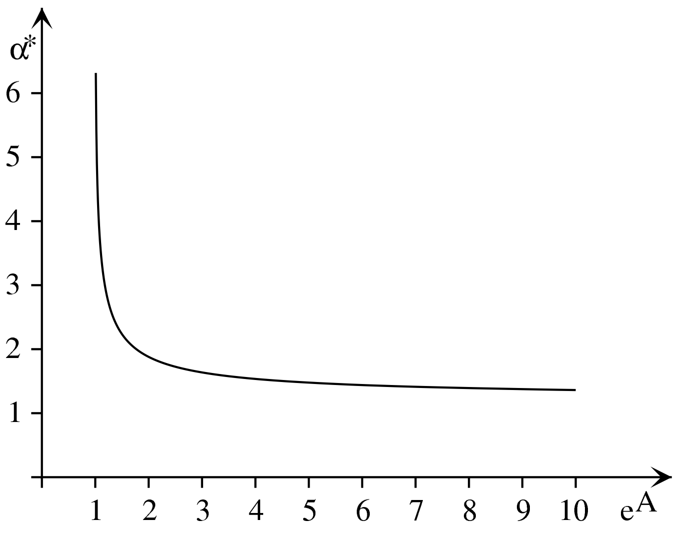

Figure 3: Optimal value α^* vs the parameter e^A.

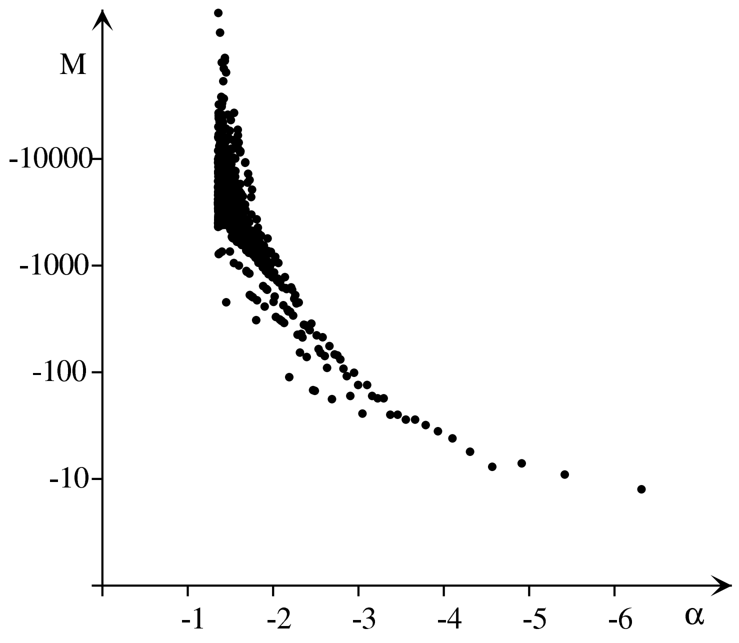

Figure 4: Stopping value of M vs α.

evaluation of the functions when the point α is changed (and M is large) is time expensive. The evaluation of the functions in the same point for increasing values of M is quite fast thanks to the recurrence formulas. This is the reason why the computational time remained acceptable.

Looking at Figure 4, the fast divergence of M as α tends to 1 is evident. What is interesting and deserve further investigations is the irregularity: consider the following "piece" of the plot

M	e^A	α^*
6839	9.67	1.35907
8395	9.68	1.35894
2827	9.69	1.3588
3779	9.70	1.35866
9395	9.71	1.35853
2331	9.72	1.3584
20029	9.73	1.35826
2465	9.74	1.35813
233536	9.75	1.358
2691	9.76	1.35786
4817	9.77	1.35773

When α is in this range, $M \sim 5000$ is usually required for a precision 10^{-5}, but the stopping value may be $M \sim 10000$, $M \sim 20000$ or even 200000! The irregularity remains for smaller values of the precision.

7 Conclusions

In this report a direct algorithm has been illustrated to evaluate the parameter in the Zipf's law that maximizes the likelihood of a given set of data. The evaluation of this parameter requires the solution of an equation involving the Riemann's zeta function and its derivative so that a numerical solution is required anyway.

The algorithm comes directly from the definition and makes use of simple upper and lower approximations of the goal function. The resulting, still rough, code is satisfactorily fast even on old computers even though it can surely be optimized after a deeper investigation of the reasons causing the strong irregularity in the parameter characterizing the approximation of the goal function.

Appendix

C code for the algorithm

The C code for the algorithm is reported here. It tests neither the parameters (the approximation and the starting guess) nor the number of input data. The functions do not check their input variables either, so, any error might lead to unpredictable results.

The main function is clearly "StimaML", which make use of the previously declared functions.

The main program, "as is", gets the data from the standard input by the function "LeggiDati" (which makes no check and stops as just as it receives a value smaller than one), calculates e^A and passes it to "StimaML".

It finally prints to the standard output the parameter maximizing the likelihood, with the interval related to the numerical approximation chosen $\alpha \quad (d\alpha)$: $\alpha* \in [\alpha - d\alpha, \alpha + d\alpha]$.

```
#include <stdio.h>
#include <math.h>

double g(int m, double x, double c){
  /* The function  G_M(x) */
  int j;
  double y,z;
  y=0.0;
  j=1;
  while(j<m){
    z=(double)j/c;
    y=y+log(z)/pow(z,x);
    j++;
  };
  return y;
}

double gj(int m, double x, double c){
  /* The function  g_M(x) */
  double y,z;
  z=(double)m/c;
  y=log(z)/pow(z,x);
  return y;
}
```

```
double h(int m, double x, double c){
  /* The function h(M,x) */
  double y,z,k;
  z=(double)m/c;
  k=x-1.0;
  y=(((log(z)+1/k)/pow(z,k))*c)/k;
  return y;
}

double dh(int m, double x, double c){
  /* H_1(M,x) - H_2(M,x) */
  double y;
  y=h(m,x,c)-h(m-1,x,c);
  return y;
}

double h1(int m, double x, double c){
  /* H_1(M,x) = Lower approximation for G(x) */
  double y;
  y=g(m,x,c)+h(m,x,c);
  return y;
}

double h2(int m, double x, double c){
  /* H_2(M,x) = Upper approximation for G(x) */
  double y;
  y=g(m,x,c)+h(m-1,x,c);
  return y;
}

double StimaML(double c, double a0, double b0, double eps, double *dx){
  /* Given the parameter c=exp(A), the starting guess for a and b,
     the precision eps; it returns the optimal solution (the precision
     is in the variable *dx) */
  int m;
  double a,b,y,ya,yb,x,x1,x2,y1,y2;
  m=1+(int)(c*2.8);
  a=a0;
  ya=h1(m,a,c);
  while(ya<0.0){
    a=(1.0+a)*0.5;
    ya=h1(m,a,c);
  };
  b=b0;
```

```
yb=h2(m,b,c);
while(yb>0.0){
  b=2.0*b;
  yb=h2(m,b,c);
};
while(((b-a)>eps) && ((ya-yb)>eps)){
  x=(a+b)*0.5;
  y1=h1(m,x,c);
  y2=y1-dh(m,x,c);
  if(y1>0.0){
    a=x;
    ya=y1;
  }else if(y2<0.0){
    b=x;
    yb=y2;
  }else{
    while(((y2-y1)>eps) && (y1<0.0) && (y2>0.0)){
      y1=y1+gj(m,x,c)+dh(m+1,x,c);
      y2=y2+gj(m,x,c)+dh(m,x,c);
      ya=ya+gj(m,a,c)+dh(m+1,a,c);
      yb=yb+gj(m,b,c)+dh(m,b,c);
      m++;
    };
    if(y1>=0.0){
      a=x;
      ya=y1;
    }else if(y2<=0.0){
      b=x;
      yb=y2;
    }else{
      x1=x;
      x=(a+x1)*0.5;
      y=h1(m,x,c);
      while(ya>y2){
        if(y<0.0){
          x1=x;
        }else{
          a=x;
          ya=y;
        };
        x=(a+x1)*0.5;
        y=h1(m,x,c);
      };
      x2=x;
```

```
        x=(x2+b)*0.5;
        y=h2(m,x,c);
        while(yb<y1){
          if(y>0.0){
            x2=x;
          }else{
            b=x;
            yb=y;
          };
          x=(x2+b)*0.5;
          y=h2(m,x,c);
        };
      };
    };
  };
  x=(a+b)*0.5;
  *dx=(b-a)*0.5;
  return x;
}

double LeggiDati(){
  /* Reads data (until a value less than 1 is read)
     and returns the parameter c=exp(A) */
  int i,k;
  double a,c;
  k=0;
  a=0.0;
  scanf("%d",&i);
  while(i>0){
    k++;
    a=a+log((double)i);
    scanf("%d",&i);
  };
  a=a/(double)k;
  c=exp(a);
  return c;
}

main(){
  double c,x,dx;
  /* Arbitrary parameters */
  double a0=2,b0=2,eps=0.0001;
  c=LeggiDati();
  x=StimaML(c,a0,b0,eps,&dx);
  printf("x = %lg (%lg)\n",x,dx);
  return -1;
}
```

References

[1] A.L. Barabasi, R. Albert (1999) *Emergence of Scaling in Random Networks.* Science. 286:509-512.

[2] A.L. Barabasi (2005) *The origin of bursts and heavy tails in human dynamics.* Nature. 435:207-211.

[3] H. Bauke (2007) *Parameter estimation for power-law distributions by maximum likelihood methods.* The European Physical Journal B. 58(2):167-173.

[4] A. Clauset, C.R. Shalizi, M.E.J. Newman (2009) *Power-law distributions in empirical data.* SIAM Review. 51:661-703.

[5] M.L. Goldstein, S.A. Morris, G.G. Yen (2004) *Problems with fitting to the power-law distribution.* The European Physical Journal B. 41(2):255-258.

[6] M.E.J. Newman (2005) *Power laws, Pareto distributions and Zipf's law.* Contemporary Physics. 46(5):323-351.

[7] W.H. Press, S.A. Teukolsky, W.T. Vetterling, B.P.Flannery (1992) *Numerical recipes in C. The art of scientific computing.* Cambridge University Press, 2nd ed.