A Nonparametric Estimation of the Cumulative Intensity Function for a Nonhomogeneous Poisson Process

Lawrence M. Leemis

February 25, 2002

A Scenario . . .

The year is 2020.

The place, some urban strip mall.

- Back in 2003, I cracked under the pressure of my computer science and math classes, and switched to philosophy.
- The job market for philosophy majors was not as good as I had hoped. Luckily, McDonalds had an opening ...
- After 17 years of slaving over McNuggets and Happy Meals, I finally had my potential noted, and was promoted to manager.
- My boss wants to cut down on employment costs, but he also wants to make sure customers are served quickly. How can he decide what number of employees he needs at different times during the day?
- He knows that I took lots of math and computer science courses back in the day, so he asks me if I can find a solution.
- It is too complicated a problem to solve explicitly (that is, I cannot do a single-variable optimization a la Calculus 112). But I could make a computer simulation of a typical day ...

A Scenario (cont'd)

- First things first. I need some real-world data, also known as "realizations." So I estimate how long it takes to make certain menu items, and how quickly the cashiers can fill certain types of orders. But how to model the arrival of customers?
- I decide to sit near the entrance for a working day (all 24 hours - it's the McDonald's of the future) and make a mark on a piece of paper every time a customer enters, so I'll know approximately how many people come in during an average day. I make lots and lots of slash marks on paper. (Years of working for McDonald's has deep-fried my brain.)
- But I have to keep in mind that that particular day may have been *National Eat at McDonald's Day*, and therefore my supposed average is far too high.
- In order to make sure my average has not been perturbed by uncharacteristic data, I need to take multiple "realizations", i.e. I need to combine data found over several days instead of using the data from just one day.
- I decide to take the next two weeks up taking observations of the entering customers to mitigate the effect of any corporate holidays. I ended up having 10080 customers in two weeks, giving an average of 720 customers every day at a rate of one every two minutes.
- When I decide to implement my simulation, I don't think it will be very representative to have a single customer arrive once every two minutes. So I decide to use a pseudo-random number generator to make variable arrival times in my simulation.

Making My Model

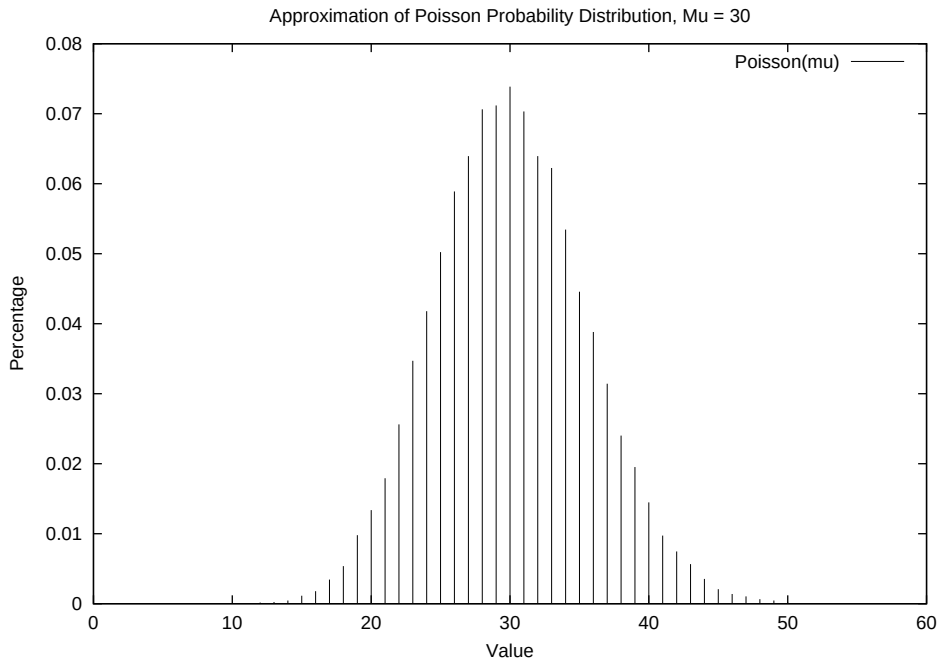
- Since I have an average of 720 customers every day, why not generate 720 random times throughout the day, sort them into chronological order, and use that for my simulation?
- Luckily, before I actually code that approach up, I take a look in my good ol' Simulation textbook and notice two bits of useful information:

1. First of all, in a given time period there is a $Poisson(\mu)$ probability distribution of the number of arrivals in that period, where μ is the average number of arrivals expected for that period.

   $Poisson(\mu)$, a discrete random variable, is defined functionally as

   $$f(x) = \frac{\mu^x}{x!}\, exp(-\mu) \qquad x = 0, 1, 2, \ldots \ .$$

   When represented graphically, it looks something like this:



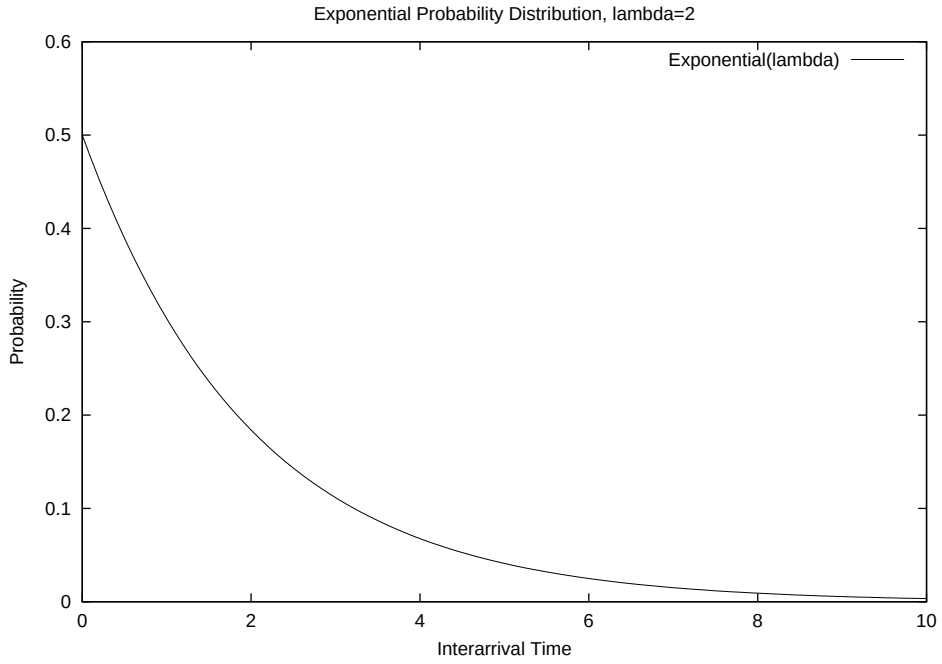
2. Secondly, random arrivals within an interval can be generated in order by summing a series of $Exponential(\nu)$ random variate calls, where ν

is the average expected interarrival time (in this case, two minutes).

$Exponential(\mu)$, a continuous random variable, is defined functionally as

$$f(x) = \frac{1}{\mu}\ \exp(-x/\mu) \qquad x > 0.$$

And the graphical representation of its probability density is something like this:



- When I'm making a lot of arrivals, I can save myself a lot of sorting time by using Exponential interarrivals instead of Uniform arrival times. So I go that way.

Generating Random Variates by Inversion

- If we have a continuous probability density function (pdf) $f(x)$, then the cumulative distribution function (cdf) is defined as

$$F(x) = \Pr(X \leq x) = \int_{t \leq x} f(t)dt.$$

  Note that if s is the smallest value s.t. $f(X) > 0$, and S is similarly the largest value, then the integral of $F(x)$ from s to S must always be equal to 1. More importantly, the cdf maps the range of possible values onto $(0, 1)$.

- Now we consider the inverse distribution function (idf) $F*(u)$. The idf is defined as the function s.t.

$$F^*(u) = x \text{ for the unique } x \text{ for which } F(x) = u.$$

  This function maps $(0, 1)$ onto the domain of the pdf. So we can use this function on a random value from a $Uniform(0, 1)$ call to generate a value in the pdf.

- The whole point of this creation of the idf is to allow us to quickly produce random variates the way the pdf would. For one thing, it only requires one call to $Random()$. And it also allows us to not have to calculate the pif (we only need the cif - the aforementioned cumulative intensity function).

Back to the Fast Food Joint

- So I create my simulation using *Exponential*() interarrival times, although in simulation terms it is referred to as a stationary (or homogeneous) Poisson arrival process.
- After creating my simulation, I test and modify the input parameters repeatedly to find a near-optimal solution.
- I give my boss the results of my testing, and assure him that I've done plenty of testing to make sure the results are correct.
- Unfortunately, when my boss implements my recommendations, everything goes haywire because my simulation suggests that there is equal demand throughout a day. So the night crews are the same size as the afternoon crews, and while bored workers are deep-frying the salt shakers at 2 AM, lines build up into the parking lot during the lunchtime rush as people wait to be served.
- Needless to say, my boss is not happy. But after enough grovelling and brown-nosing, he decides to give me one more chance to redeem myself.
- This time, you can be sure that the specification model is going to be more accurate.

Time for Nonhomogeneity

- Obviously a stationary Poisson process is not the answer! I need to factor in a changing arrival rate over the average day.
- When I look back at my Simulation book again, I notice that there are techniques for estimating a nonhomogeneous Poisson process. A NHPP is simply the generalization of a stationary Poisson process, in that it has its arrival rate $\lambda(t)$ as a function of time, rather than a constant value λ.
- The expected number of arrivals within an interval (a, b) is no longer a simple $\lambda(b-a)$ but instead $\int_a^b \lambda(\tau)d\tau$. When this interval begins at 0, it is an evaluation of the cumulative intensity function $\Lambda(t)$, so that we have

$$\Lambda(t) = \int_0^t \lambda(\tau)d\tau, \qquad t > 0.$$

- Of course, the expected value is only the average outcome, and generally not a possible value (in that it's a floating point number, while we can only have a discrete number of events). To find the probability of exactly n events occuring in the interval (a, b), we use the formula:

$$\frac{[\int_a^b \lambda(t)dt]^n \; e^{-\int_a^b \lambda(t)dt}}{n!}$$

Note that the sum of the probabilities of all possible values is equal to one (since by basic calculus, $\Sigma_{i=0}^{\infty} k^i/i! \; = \; e^k$).

Parametric vs. Nonparametric

- When I choose to approximate an NHPP, I have a choice between parametric models and nonparametric models. The difference between them is that parametric models incorporate one or more outside variables, which have values that are either chosen directly by the modeler or somehow supplied. Nonparametric models are generated solely from the data of our realizations.

- One common parametric technique is a Weibull process. Its mathematical definition is

$$\Lambda(t) = (\alpha t)^{\beta}, \quad t > 0.$$

  Note that α and β have to be supplied somehow from outside the data.

- A more complicated example is used by Lee, Wilson, and Crawford to describe weather events in the Arctic Sea.

$$\lambda(t) = \exp\left\{\sum_{i=0}^{m} \alpha_i t^i \; + \; \gamma \sin(\omega t \; + \; \phi)\right\}, \quad t > 0.$$

  This model has both a cyclic component (for the seasons) created by the sinusoidal part, and an overall trend controlled by the polynomial part.

- There are also plenty of nonparametric modeling techniques, ranging from the very simple to quite complex. Here is one that falls in the latter category, from Lewis and Schedler:

$$\hat{\lambda}(t; n, t_0) \; = \; \frac{1}{b(n)} \sum_{j=1}^{n} W\left(\frac{t - T_j}{b(n)}\right), \quad t > 0.$$

Please, Spare This Poor Brain of Mine!

- After 18 years at McDonald's, I really want a relatively simple way to approximate the cif (cumulative intensity function). Luckily, a copy of Dr. Leemis' paper has stuck to the back of my Simulation textbook.
- His nonparametric approximation function is based on linear approximations generated in between the n data points, and the interval endpoints 0 and S.
- The basic requirements of this estimator are:
  1. The intensity function must always be positive.
  2. I need to do k realizations on the time interval $(0, S]$. In my case, this is $(0, 24]$.
  3. I record the time of each event, and superimpose the observation times to get an ordering of them all as t_i's. (NB: We denote time 0 as t_0 and time S as t_{n+1}.)
  4. I record the total number of events over the k realizations as n.
- Once all that is done, we define the cif as follows:

$$\hat{\Lambda}(t) = \frac{in}{(n+1)k} + \left[\frac{n(t - t_{(i)})}{(n+1)k(t_{(i+1)} - t_{(i)})}\right], \qquad t_{(i)} < t \leq t_{(i+1)}; \quad i = 0, 1, 2, .$$

To gain some understanding into this complicated function, note that:

1. We are dividing by $n + 1$ in each fraction to account for the $n + 1$ intervals between t_0 and t_{n+1}.
2. $\Lambda(t_{n+1}) = n/k$, as expected (since n/k is the average number of events over the k realizations).

3. All the times t_i have their $\Lambda(t_i) = (in)/((n+1)k)$. So in each interval (t_i, t_{i+1}), Λ increases by $n/((n+1)k)$, regardless of the interval length. So more closely packed data points will result in a steeper slope in those areas.

4. The second fraction in the equation is used in order to generate the proper linear line between any two t_i and t_{i+1}.

- A special case that must be considered is when two data values are equivalent, i.e. we recorded two events as happening at exactly the same time. In this case we must be slightly more explicit than above. So if $t_m = t_{m+1}$, we have

$$\hat{\Lambda}(t_{(m)}) \;=\; \hat{\Lambda}(t_{(m+1)}) \;=\; \frac{mn}{(n+1)k} \;\text{ and } \lim_{t \downarrow t_{(m+1)}} \;=\; \frac{(m+1)n}{(n+1)k}.$$

Some Empirical Evidence

- This kind of linear estimation works much better when k is very large. In fact, since both the estimator and the actual NHPP have the same mean for any t in (0,S], it is possible to prove that the estimator converges to to the actual NHPP as k goes to infinity. Intuitively, this means that as we take more and more realizations, we make better approximations.
- The $(1-\alpha)$ confidence interval for the linear estimator is as shown below:

$$\hat{\Lambda}(t) - z_{\alpha/2}\sqrt{\frac{\hat{\Lambda}(t)}{k}} < \Lambda(t) < \hat{\Lambda}(t) + z_{\alpha/2}\sqrt{\frac{\hat{\Lambda}(t)}{k}}$$

NB: $z_{\alpha/2}$ is formed from a call to $idfNormal(0, 1, 1-\alpha/2)$ if you are using Park's *rvms* library.

Variate Generation

- As we saw before, the inverse distribution function (idf) can be used to generate event times. We can use the event times from a unit Poisson process to generate the event times as follows:

  1. $i \leftarrow 1$,
  2. generate U_i from $U(0,1)$,
  3. $E_i \leftarrow -\ln(1-U_i)$,
  4. **while** $E_i < n/k$ **do**,
     $m \leftarrow \lfloor \frac{(n+1)kE_i}{n} \rfloor$,
     $T_i \leftarrow t(m) + [t_{(m+1)} - t(m)] \left(\frac{(n+1)kE_i}{n} - m \right)$,
     $i \leftarrow i+1$,
     generate U_i from $U(0,1)$,
     $E_i \leftarrow E_{i-1} - \ln(1-U_i)$,

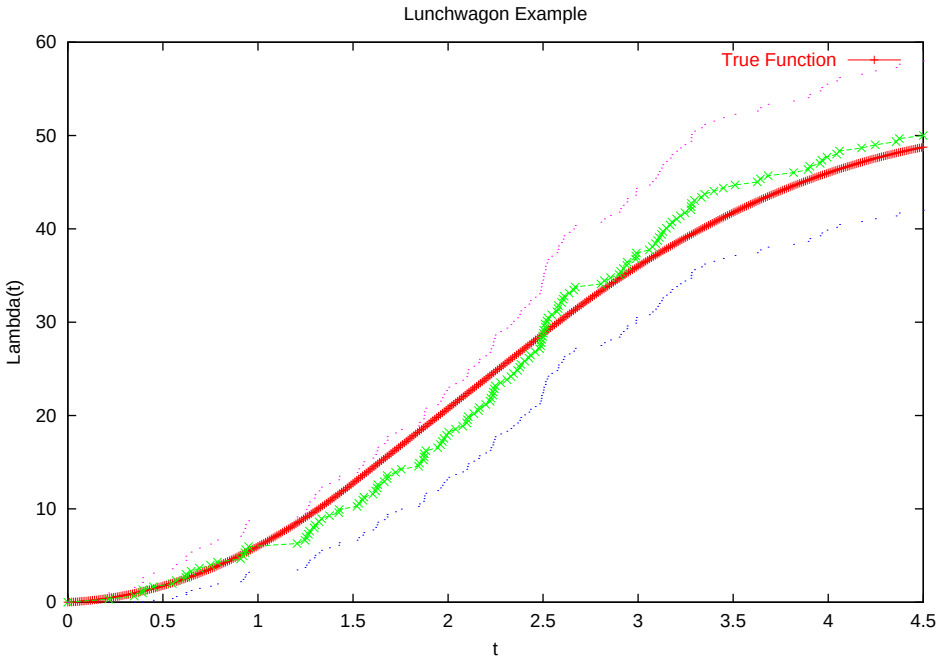
  end.

- Note that we can replace $(1-U_i)$ with U_i, although the monotonicity is reversed.

- As can be seen by looking at the "while" condition, the generation time of the algorithm is dependent on n/k instead of n. Therefore taking more realizations does not increase the run time of the algorithm, as it might for other approaches.

Another Food Example

- Here we look at an example presented in the paper that demonstrates a real-world application; in this case, it is the number of arrivals at a lunch-wagon over a 4.5 hour period of time at midday.



- The smooth line is the true function, while the one closest to is the linear estimator. The two outside, fuzziest lines are the 95% confidence intervals for the generated estimator.

- Just eyeballing the graph, it seems reasonable to suggest that the confidence interval does in fact contain the true function about 95% of the time.

Extensions

- So far we have never dealt with a case in which $\lambda(t) = 0$ for a given interval (the basic model requires that $\lambda(t) > 0$). But it is relatively simple to make adjustments for such a period, given the interval $(a, b]$ bounded by t_i and t_{i+1} on which the arrival rate is 0.
- We have (at least) two ways to account for these "zero periods". The first formula makes it so that the value of the zero interval is equal to $\Lambda((a+b)/2)$. This is:

$$\hat{\Lambda}(t) = \begin{cases} \frac{in}{(n+1)k} + \frac{n(a+b-2t_i)(t-t_i)}{2(n+1)k(a-t_i)(t_{(i+1)}-t_{(i)})}, & t_i < t \leq a, \\ \frac{n(a+b-2t_i+2i(t_{(i+1)}-t_i))}{2(n+1)k(t_{(i+1)}-t_i)}, & a < t \leq b, \\ \frac{(i+1)n}{(n+1)k} + \frac{n(2(t_{(i+1)}-t_i)-a-b+2t_i)(t-t_{(i+1)})}{2(n+1)k(t_{(i+1)}-b)(t_{(i+1)}-t_i)}, & b < t \leq t_{(i+1)}, \end{cases}$$

- A more tractable approach is just to assume that the line segments that touch the constant interval have the same slope. This approach gives the formula:

$$\hat{\Lambda}(t) = \begin{cases} \frac{in}{(n+1)k} + \frac{n(t-t_i)}{(n+1)k(t_{(i+1)}-t_{(i)}-b+a)}, & t_i < t \leq a, \\ \frac{in}{(n+1)k} + \frac{n(a-t_i)}{(n+1)k(t_{(i+1)}-t_{(i)}-b+a)}, & a < t \leq b, \\ \frac{(i+1)n}{(n+1)k} + \frac{n(t-t_{(i+1)})}{(n+1)k(t_{(i+1)}-t_{(i)}-b+a)}, & b < t \leq t_{(i+1)} \end{cases}$$

- These extensions are nearly equivalent if the zero interval $(a, b]$ is only a small part of the interval they are contained within. In some cases they can actually be equivalent (if the zero interval is centered within (t_i, t_{i+1})). Neither way is obviously "better" than the other, but since the second approach is slightly simpler it is probably the better choice in most situations.
- There are other extensions presented in the paper, but in the interests of time we will ignore them. Essentially they all focus on making the linear approximator more accurate for very specific types of cases.

The End

- The primary point to retain after this presentation is that when k (the nubmer of realizations) is large enough, we can get fairly useful and relatively simple results.
- After taking all this into account, I make a new model for McDonalds and get very good results. For my efforts, I get a \$1/hr raise, so it was all worth it!!