**Investigations into Stochastic Time Delay Structures for Regression**

**Abstract**

Consider the typical univariate regression problem where the intention is to find

the relationship between x and y. When extended to time series, a number of time specific complexities arise. For the specific case where the input x affects the output y, with a random time delay in between, the estimated coefficients (or weights, predictions etc) are significantly attenuated. This attenuation occurs both in traditional statistical models and machine learning models, and for certain problems can be a serious limitation. The following document proposes techniques for handling this class of time series regression.

Random wait time (t) \*\*add symbol\*\*

x

y

As an example, let‘s take the management of blood sugar level in diabetes patients. In people with diabetes, the pancreas can't effectively regulate blood sugar levels. Therefore,

these levels must be controlled by insulin injections and a special diet. The challenge for many people, is that the relationship between the input (insulin) and output (blood sugar) is extremely complex. The effects of the insulin injection might be observed after 15, 20 or "t" minutes depending on a number of factors, many of which are unknown. Due to the stochastic nature of the time delays, the actual effect can't be easily determined. It’s difficult to differentiate the effect of the insulin injection from other factors and accurately determine how much one should take.

Inference for this type of problem is especially challenging. Typical regression models require a fixed alignment between cause and effect. Using standard methods, we'd need to assume that the effect occurs after some fixed time 't' which can be inferred from the data. However, if there is any uncertainty in the parameter 't' (t changes or is noisy) the resulting effect will be significantly attenuated.

Consider the simple example below where the effect of the input is 1, as shown in the image below. The observed input is given by the red line, the blue line is when the effect actually occurs. The first effect happens 1 time point after the input. The second effect happens at the same time as the input. A fixed time delay isn't valid in this case because the time shifts differ.

Chart, histogram

Description automatically generated

If we were to model the effect using a fixed time delay, the estimate would only be 0.5. This is because only one of the outputs is aligned. Obviously, this isn't ideal, we want the parameter estimates to be as close the real values as possible, regardless of any noise in the lag structures. One can mitigate the problem via time aggregations, however in complicated multivariate cases, this is just not feasible.

Therefore, we propose a regression model which can handle stochastic time delay structures. In order to do, we treat the stochastic time delay components as an ‘error’ in the time axis. Next, we find the maximum likelihood estimate, for a given set of parameters, considering the time error (t-axis) and the regression error (y-axis) simultaneously.

**Methodology**

We consider the problem as analogous to the typical error-in-variables (EIV) regression. Ordinary regression analyses (and machine learning models) define the loss function

with respect to errors in the y axis only. For EIV, errors are considered in both the y-axis and the x-axis **(reference).** This is useful when there are measurement errors in the

independent variable eg. because your physical measurements have some degree of random error. For our problem we assume that we have errors in the y-axis and the t-axis. That is, there are random prediction errors and random errors in the time domain.

Now, assume that there exists a time series x(t) and we want to determine to the functional relationship f to some other variable y. The series x(t) is constrained to be a stationary series with the a known value for where f(x(t)) = 0. That is, we know when the input series has no effect on the output (when it’s off). The series x(t) is also assumed to be a discrete time series with a one to one mapping between x(t) and f(x(t)). For clarity, we call all points where f(x(t)) != 0, 'non-zero impulses' because these points might have some effect on the output series. In addition, the input series should be sparse, in that there is a low density of ‘non-zero impulses’ in any region of the series.

Despite the existence of an observed x(t), due to noise in the time domain, we don’t know when the effects actually occur. We denote the time shifted series by x(t + tau), where tau is some shift in time for each non-zero impulse in the series.

We also assume that the set of taus, denoted ‘Tau’, is not a constant, but rather a random draw from some distribution (eg discrete gaussian, poisson).

**Tau ~ N(u,tsd) or Tau ~ Pois(lambda)**

and

**e ~ N(u, sd)**

Firstly, we take the input series x(t) and decompose it into its constituent non-zero impulse components.

So, if x(t) is a vector given by [0,0,1,0,1,0] then we decompose the vector into a matrix X(t) = [[0,0,1,0,0,0],[0,0,0,0,1,0]], where each impulse is treated separately.

**(insert image of X(t) matrix)**

Then the method should find the function f in the following relationship:

**y(t)= f(X(t + Tau)) + e**

where X(t + Tau) is the time shifted series and X(t) is the observed input.

In order to find the best estimate of f, we would like to find the function f() which maximises the joint likelihood of both Tau and e, the time-domain likelihood and the error likelihood respectively.

For simplicity, assume that the tau and error distributions are independent. To begin with, the values of each individual time shift (tau) are not known to us. In addition, the size of prediction error e can only be determined if each tau is known (because for each time shift there is a different prediction and prediction error). Therefore, our algorithm should find the best estimate of the individual time shifts from the data.

**Algorithm**

Firstly, we define a parametric form and some initial parameters to be estimated for the function f. For example, lets take the simple univariate linear model where the f(x(t)) is

parameterised by Beta, error standard deviation and the standard deviation of Tau. First, we initialise some starting values for each of these parameters.

Now we want to find the best possible time shift (tau) for each input impulse in X(t). It stands to reason that the best possible time shift would be one that is not too far away from the observed impulse and also gives the best possible prediction. In this example, we can get the prediction y by simply multiplying the shifted value by its parameter Beta. From there we can calculate the likelihood estimate for time shift plus the prediction error.

In principle, we can then try a number of values of tau (i.e. optimise) until we maximise the likelihood for this impulse.

However, we must also consider that the impulses in X(t) are not independent from each other. After shifting, its possible that two or more effects can occur simultaneously.

This could be particularly problematic if there are multiple impulses within a short period of time, or the impulses have a distributed effect over multiple time points. As an example consider the series x=[0,0,1,1,0,0] with Tau = [1,0] and Beta = 1. For this case, X(t + tau) = [[0,0,0,1,0,0],[0,0,0,1,0,0]] and the effect is therefore y = [0,0,0,2,0,0]. We need to consider the effects at the same time to accurately calculate the likelihood.

As shown above, if the impulses are not independent, then we need to consider time shifts (taus) at the same time. Therefore, we treat the problem of finding the best time shifts as a discrete optimisation problem.

For the optimisation step we utilise two assumptions to complete the optimisation. Firstly, smaller shifts are more likely than larger shifts (proportionate to the standard deviation of the tau distribution). This means that we should explore the space of smaller shifts more often than larger shifts. Secondly, impulses close to each other are more likely to be dependent on each other, than impulses further away.

Therefore, we intialise the optimisation algorithm with all parameters set to the mean of the tau distribution (eg zero for the zero mean gaussian). Then we randomly select a small number of impulses, the exact number could be specified manually as a hyperparameter, or inferred from the data by looking at the density of impulses in time. These impulses are randomly changed to a value drawn for the tau distribution (proposal vector), and the likelihood (both shift and prediction error) for the proposal is calculated. If the proposal likelihood is higher than the current maximum likelihood, then we update our initial estimate to the current best guess. This procedure is done in a loop, with the best guess of the taus improving over time.

Lastly, we optimise over the set of parameters Beta, error mean and Tau standard deviation, iteratively alternating between the time shift optimisation and the parameter optimisation. For the parameter optimisation, typical methods can be used such as gradient descent, genetic algorithms or annealing. In our implementation we have used the differential evolution algorithm from the **scipiy.optimize** package in python, due to its ability to handle noisy objective functions. In order to improve convergence, we also range standardize all input variables.

Through optimisation, we find the best fit for the values of Beta, error mean and Tau sd. The accuracy of the final parameter estimate is relative to the ratio of the y-axis error and the effect size B\*X(t). As e/B\*X(t) tends to infinity, the parameter estimate Beta tends to the standard linear regression coefficient.

On the other hand as e/B\*X(t) tends to zero, the estimated value of B tends to the correct value. The result is bounded in the worst case by standard regression estimates. Therefore, the method provides no guarantee on recovering the exact time shifts, instead only that we obtain an estimate equal to or better than standard regression.

**Limitations**

As the length of x(t) increases and more impulses are introduced, the size of the decomposed matrix X(t) becomes very large. We assume that

impulses further away from each other are functionally independent.. Therefore, the matrix X(t) is broken into a number of segments, where each segment is

composed of a smaller number of dependent impulses. Each of these segments can be optimised independently, in parallel, with little impact on the final estimates. This makes the inner optimisation procedure tractable for longer length sequences. To select the right split locations, we search the time series for locations with the lowest density of non-zero impulses.

Finally, a note on problem constraints. The likelihood estimates are dependent on the inner (tau) optimisation procedure. There is no guarantee than the global maximum will be found during this process, particularly for sequences with a high density of non-zero impulses. For example, consider the example sequence x(t) = [0.5,0.6,0.4,1,2,1,0.3,0.2,0.5,0.8], where each impulse can shift between -2 and +2 positions. Ignoring shifts beyond the edge of the vector, the solutions space of Tau is 5^t options.

In cases such as these, the estimated likelihood is likely to be close to, but not exactly the same as the real value. As the density (in time) of impulses increases, the accuracy of the estimates decreases and therefore this model is most appropriate for sparse time series inputs.

**Experiments**

The following section shows performance on a simulated univariate example. Code to replicate the example can be found via the following link:

<https://github.com/aaron1rcl/tvs_regression/>

The simulated input signal (orange) has 17 non-zero impulses which have been drawn from the standard normal distribution. The system is off when x(t) =0. In other words, when x(t) = 0, f(x(t)) = 0.

The true shift distribution Tau is given by Tau ~ N(0, 2). There is one value of tau for each of the 17 non-zero impulses.

The green line below represents the shifted series, corresponding to the time at which the effect occurs. The blue line is the output ‘y’.

The values for the parameters were arbitrarily selected. The output y is defined by the following equation:

y = 2\*X(t + tau) + 6.5

**A picture containing graphical user interface

Description automatically generated**

The actual distribution of Tau is shown in image below:

Chart, bar chart

Description automatically generated

After fitting the model with the tvs regression. We obtain the following results:

|  |  |
| --- | --- |
| **TVS Regression Fit**  Chart  Description automatically generated | **OLS Regression Fit**  Chart, histogram  Description automatically generated |
| **Fit Convergence**  Chart, bar chart  Description automatically generated | **Scatterplot of x(t) vs y(t)**  Chart, scatter chart  Description automatically generated |

True Shifts vs Estimated Shifts

[-1,1,0,-1,0,-4,-2,-2,-1,-1,0,1,1,1,-1,1,0,3,1,0]

[-1,1,0,-1,0,-4,-2,0,-1,-1,0,1,1,1,-1,1,0,3,1,0]

**Estimated Parameters (Comparison):**

|  |  |  |  |
| --- | --- | --- | --- |
| **Parameter** | **TVS Regression** | **OLS Regression** | **True Value** |
| **Beta** | 1.96 | 0.26 | 2 |
| **Constant** | 6.48 | 6.62 | 6.5 |
| **Tau SD** | 1.51 | N/A | 1.47 |
| **Error SD** | 0.26 | 1.03 | 0.2 |

**Future Work**

In the univariate case, the utility of the described method is limited. However, we propose two extensions as future work. The first is to extend the method to multivariate regression. In principle, this requires only a small adjustment to the accommodate inference of multiple Beta values. The next extension could include the modelling of distributed lag structures. A distributed lag structure is where the effect occurs at some function of the lagged input values and where the effect can persist across a number of time points.

Through these two extensions we can begin to tackle a larger number of practical problems including the diabetes example described in the abstract.

In addition, by incorporating these elements we can begin to explore alternative forms of time series prediction. Essentially, existing prediction mechanisms predict the average across time shifts. Even though the loss in these models is equivalent to the proposed approach, the results can be nonsensical.

For example, let us consider a regression model to predict how much coffee someone drinks between 9am and 9:30am, at work. They set off every day in their car at exactly 8:30am and it takes about 30 minutes (+- tau) to get to work. The time shift tau could be caused by something as simple as variations in traffic.  
When they get to work, they always have a coffee. A typical regression might predict 0.5 coffees between 8:30 am and 9am and 0.5 coffees between 9am and 9:30am. Alternative prediction forms such as: 'When they get to work, they will have 1 coffee' might be more useful.

**Conclusion**

We have proposed a form of regression analysis for dealing with stochastic time delay problems. We have shown the feasibility of the approach and its performance on simulated data. The approach allows for better estimation and prediction when the input is affected by noise in the time domain. To our knowledge, the method is novel for this class of problem.