Business Report on Cloud Resource Planning

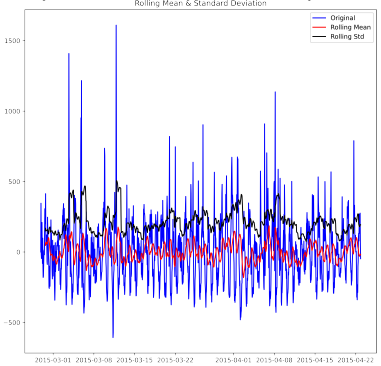
Thibaut Hurson - 05/03/2021

ABSTRACT

We are working on a webserver application that processes Amazon tweets, and which requires cloud resources. We would like to be able to forecast resource required for the next week time, at an hourly granularity. The cost of cloud resources is \$0.1 per tweet, and there is a user experience penalty equal to $p(\delta) = \$0.5\delta^2$, where δ is the number of resources we must urgently subscribe in case we are short. We are given a dataset that contains two months of Amazon tweets between 26/02/2015 and 22/04/2015, and we will forecast resources needed for the last week of data from 13/04/2015 until 19/04/2015. We will compare two techniques: An online gradient technique, and an Auto-Regressive Integrated Moving Average (ARIMA) forecast paired with a stochastic optimization approach.

1 ONLINE GRADIENT TECHNIQUE

Before even implementing a forcasting model, we need to verify that the time series is stationary. The reason why we verify time series stationarity is because we cannot build a time series model if the time series is not stationary. We use the rolling statistics plots along with Dickey-Fuller test results to verify it.



First, we can see that the rolling mean, and the rolling std are horizontal, but seems to have some trends. This give us a hint that the time series may be stationary. Thanks to Dickey-Fuller test, we obtain that the test statistic is smaller than the 5% critical values so we can say with 95% confidence that this is a stationary series. Moreover, as the p-value is inferior to the 5% threshold, the null hypothesis is rejected, meaning that the Dickey-Fuller test is verified. Thus, the time series is stationary. The figure above is actually obtained by substracting the rolling mean to the original time series. Doing so, enable us to obtain an even more stationary time series.

Let's derive the Online Gradient Ascent (OGA) algorithm. We suppose that y_t^π denotes the amount of cloud resources planned, and x_t the unknown demand. The reward function adds a quadratic penalty whenever the demand exceeds the planned resource:

$$f_t(y_t) = -0.1y_t - 0.5 * (x_t - y_t)^2 * 1_{x_t > y_t}$$

Let's compute the directional derivative:

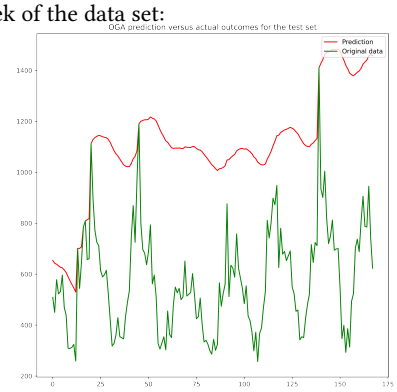
$$\frac{\partial f_t(y_t)}{\partial y_t} = -0.1 - (x_t - y_t) * 1_{x_t > y_t}$$

Then, the algorithm is resources planned in the following way:

$$y_{t+1} = (y_t + \eta((x_t - y_t)1_{x_t > y_t} - 0.1))^+$$

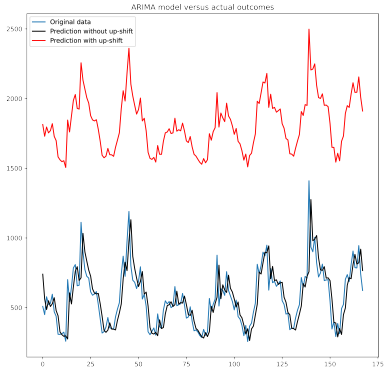
where η is the stepsize.

By using the latter equation, we obtain the following result for the last week of the data set:



We found out that we tend to allocate way more resources than necessary. When there is a spike, resources called out increases tremendously. Thus, for a well chosen eta (here eta = 1), most of the time, we have enough resource, however the amount of resources called up and unused is quite high. Hence, Online Gradient Ascend technique is not optimal in this particular scenario. Let's try to implement Auto-Regressive Integrated Moving Average (ARIMA) forecast to get better ressults.

2 ARIMA



We can see that the prediction with ARIMA is quite accurate. Yet, there is an often between predictions and actual outcomes, and predictions are often below actual outcomes. This can result in very high user experience penalty as seen above.

To counter those issues, we add a positive constant value. By doing so, we can counter times when there is a shift in cloud resource demand. We obtain the highest cumulative reward for a constant equal to 1240.

Adding a constant value to predictions is highly benificial for our reward function, the penalty is tremendously reduced, and we can see on the figure above that we manage to deal with shifts in cloud resource demand.