Data Science Research at Aiden

Hugo Cisneros

June - September 2017

Trend change detection

The first part of my work was about investigating methods to detect trend changes in time series. One implementation of the methods described below has resulted in the smart insights, that automatically detect trend changes of a desired KPI.

The model is based on the extraction of a trend in the input data. Different methods were tested and evaluated for this purpose.

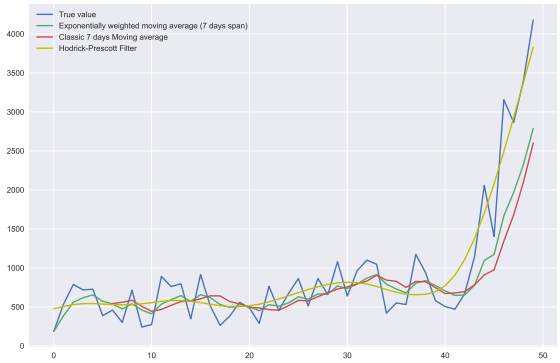


Figure 1: Comparison of different trending methods

Moving average and EW (exponentially weighted) moving average are two methods involving an average of past data to determine the trend. They are mostly equivalent for similar window parameters, although I have kept the latter because it is slightly more quickly sensitive to trend changes as shown on the graph.

The Hodrick-Prescott filter is obtained with radically different numerical techniques and is shown here mostly for purpose of clarity. It is not used in the final model because of several undesired properties (mainly the fact that it takes into account previous and future data and needs to be recalculated for every addition of data point).

After extracting the trend of the data and thus removing undesirable noise from the signal to be analyzed, a trend detection algorithm can be applied. The one I implemented is very simple and only involves linear regressions on the data. Two linear functions are fitted on the time periods to be analyzed (last week and the week before in the model). The slopes of those functions are then compared to determine whether the trend change is relevant or not.

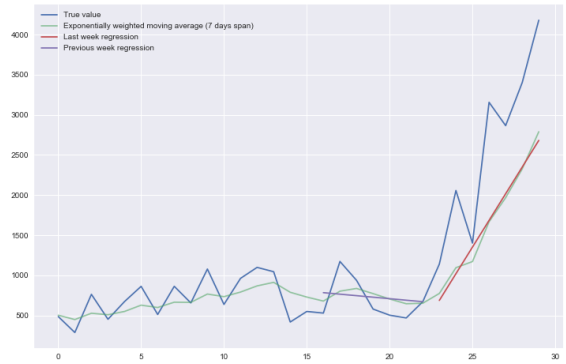


Figure 2: Illustration of the trend detection pre-processing

After empirical analysis of several trend thresholds, a 10 percentage points difference between the two slopes was chosen (For the example of the graph, the slopes could be -5% and 40%, resulting in a difference of 45% which is above the 10% threshold).

A possible improvement of the algorithm would be to implement a more complex and/or adaptive threshold system that could lead to more robust and potentially more accurate trend change detections.

Classification algorithms could be used with labeled data (i.e. sets of data points associated with the presence of a trend anomaly or not) and would probably yield more accurate and generalisable results.

Practical Notes: To implement this method with Facebook data, we had to deal with the windowed attribution system of actions (i.e., the fact that e.g. installs are attributed up until 28 days after the ad view) that made KPIs such as the CPI very unstable over time and with systematic uptrends for most recent data. To cope with this, the algorithm is actually performed on 1 day attributed data, which makes it much more consistent even for recent data. But it is based on the hypothesis that every unusual activity during the first day will have the same impact over the following 28 days of attribution.

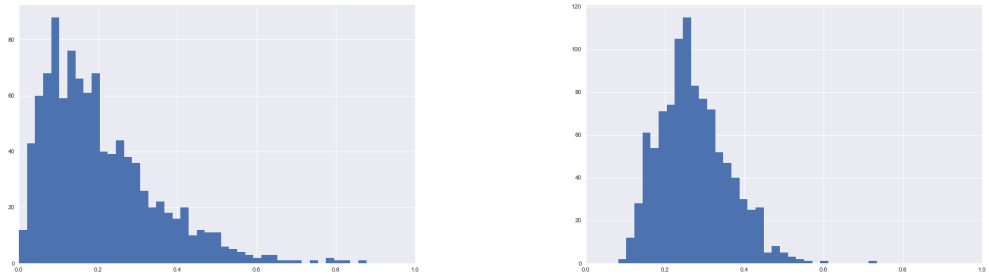
Outliers detection

I've explored some outliers detection methods on different kind of datasets. Detecting outliers in data implies recognizing and learning the "normality" of this data's distribution. This can be achieved through different methods and algorithms depending on the nature, dimensionality and distribution of the input.

I first worked with conversion rates between kpis on marketing data. Detecting outliers on such data seemed fairly natural since they have very well defined distributions.

One of the most widely used machine learning algorithms for probability distribution fitting is called the Gaussian Mixture method. The distribution is assumed to be a weighted sum of several Gaussian distributions and the exact parameters are computed with the expectation maximization algorithm that iterates through the training data.

The result of the Gaussian mixture algorithm is a model that can give the probability of appari-



(a) Distribution 1 (b) Distribution 2

Figure 3: Histograms of two simulated data distributions

tion of a given input with respect to the observed data. On the two figures 4 and 5 are plotted each distribution 3a and 3b, a sampled distribution from the fitted Gaussian mixture model to illustrate the accuracy of the model and the log probability of every point in the range [0, 1].

This learned distribution allows for the implementation of a classifying algorithm which will check for the log probability of a given input and treat it as outlier if it is below a certain threshold.

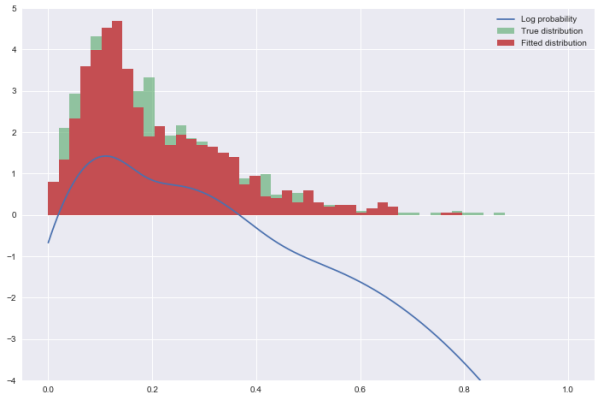


Figure 4: Gaussian mixture of 3 Gaussians fit for distribution of Figure 3a and log probability

This threshold can be determined empirically by choosing the outlier elements from the original distribution and by setting the threshold such as they would be indeed classified as outliers.

A limitation appears rapidly. For instance, the first distribution has its center of mass very close to zero and thus makes it impossible to detect small outliers close to zero since it would imply setting the threshold to approximately -0.6 and therefore classify a large chunk of the tail of the distribution as outlier.

This algorithm could nonetheless provide a robust and powerful method to detect outliers on a set of data with a well defined probability distribution. It can be applied to datasets of any dimension, and potentially very complex distributions since an arbitrary number of Gaussian can be used for fitting.

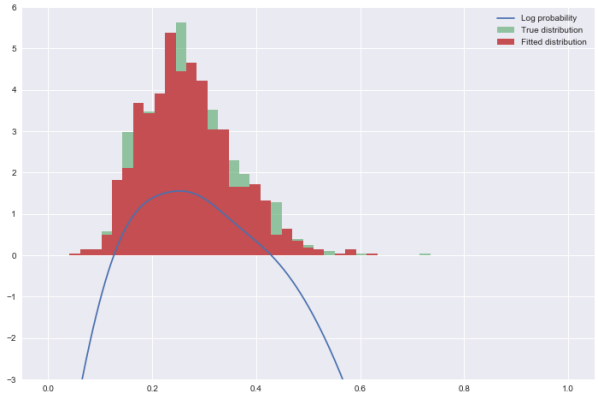


Figure 5: Gaussian mixture of 3 Gaussians fit for distribution of Figure 3b and log probability

The main obstacle to applying the method to non uniform data is the ability to extract the "explicable" component of this data, that is the part that can be explained and shouldn't be looked at for detecting outliers. Of course, the "trend" itself could be unexpected but isn't an outlier in the sense that by definition it is a somehow durable phenomenon. Once the trend component has been extracted from data, the same process as above can be applied and unlikely data points can be detected.

To demonstrate the method with non priory bounded data, I will show an example of this algorithm applied to a generated dataset with a known underlying trend and a random noise. Let f be the function sampled to generate a dataset, defined by

$$f(x) = \left(\frac{x}{3} + 1\right)(0.5\sin(1.5x) + 3) + \mathcal{N}(0, 0.8)$$

Where $\mathcal{N}(\mu, \sigma)$ denotes the normal distribution of mean μ and standard deviation σ. An example sample for the function f is shown in the graph below:

Figure 6: Dataset generated with function f. The underlying trend is shown in green on the graph

The presence of outliers in a dataset is subject to debate, since the unlikeliness of a point depends of the subjective idea of what the dataset should look like. But in this particular generated data, they

are well defined by the points which random component cannot be explained by the normal distribution. In other terms, those points were very unlikely to appear given the underlying random distribution and therefore can be considered as outliers.

By removing the trend from the data and classifying as outliers elements from the Gaussian distribution having less than 2% probability of occurring, the outliers of Figure 7 are obtained.

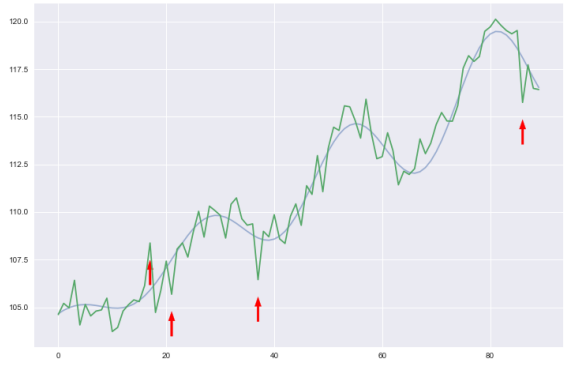


Figure 7: Outliers detection on dataset from Figure 6

Those outliers might not exactly be the ones a human agent would have classified as such, but they nonetheless follow a rather robust definition of what an outlier could be for this dataset.

In practise, the real trend of the data cannot be known precisely and needs to be estimated. It can be achieved by extracting the with methods similar to those described in *Trend change detection* section. For example, when estimating the trend with an exponentially weighted moving average, the outliers shown on figure 8 are detected.

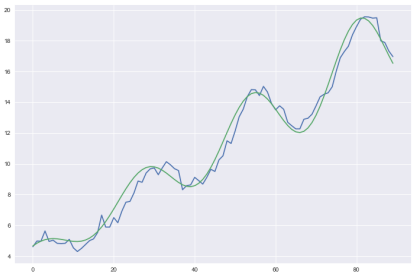
On Figure 8, it appears that some outliers weren't correctly detected with the estimated trend. It is also obvious (and this is a known property of the EW moving average because it takes into account all previous data, thus less information for the first points) that trend isn't well defined for the first data points. It happens here to correspond to the position of undetected outliers.

When the underlying random distribution isn't a Gaussian, it can simply be learned from data with a mixture of Gaussians for instance to be sure to detect truly unlikely data.

The algorithm works well for this particular dataset but cannot be bluntly applied to any time series without adjusting its parameters. For example, the smoothness of the moving average will have a great influence on what will be interpreted as an outlier or not. The most common way to proceed is to decompose the time series into trend and seasonality and to perform outlier detection on the residual data.

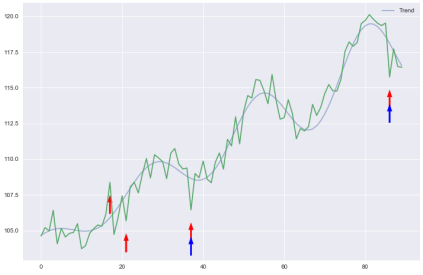
In the example dataset above, I assumed that the "seasonality" or periodic component was part of the underlying trend but more elaborate techniques for extracting the periodic component could yield better results by better isolating the strictly random part of the data.

In the end, it all comes to the same principle: learning the distribution of the "random" component of the data and deducing from this distribution the data points that were unlikely to appear. Of course lots of examples exist where this kind of techniques won't work perfectly and will yield false positives and false negatives but they could in some other cases provide a good enough way to efficiently detect outliers in time series.



(a) Estimated trend

(b) Residual data with limits at 2% probability for the fitted Gaussian



(c) Outlier detection with estimated trend (Red: "True" outliers, Blue: outliers with estimated trend)

Figure 8: Estimated trend and corresponding detected outliers for data of Figure 6

To apply this algorithm on real time data, a trade-off between the accepted lag of the results (numbers of days to let pass before processing data and looking for outliers) and the number of false positives must be made. A too short lag will result in a poorly estimated trend for rapidly changing data but for a too long lag, data will risk to be outdated when an outlier is detected.

On predicting spend

I've spent some time trying different models and architectures for predicting budget spend and haven't got very compelling results. I believe this is due to the fact that even though some patterns exist and can be easily predicted, this spend is for most part totally unpredictable, or at least that Facebook doesn't give enough data out of the box to do so.

The first most intuitive approach that can be explored is what is called auto-regression, that is trying to infer next data points with a linear regression over some features extracted from the previous data points. It allows to easily pick up seasonal patterns but won't be able to learn and/or predict changes in the trend except if they follow a pattern that exists in the data.

This auto-regressive model can be enhanced by adding external temporal information such as another metric (it could be the past number of impressions, installs, etc, for spend prediction) to try and pick up correlations between one another. This would work really well for situations where actions on a particular day have repercussions on the following days but that is not always the case.

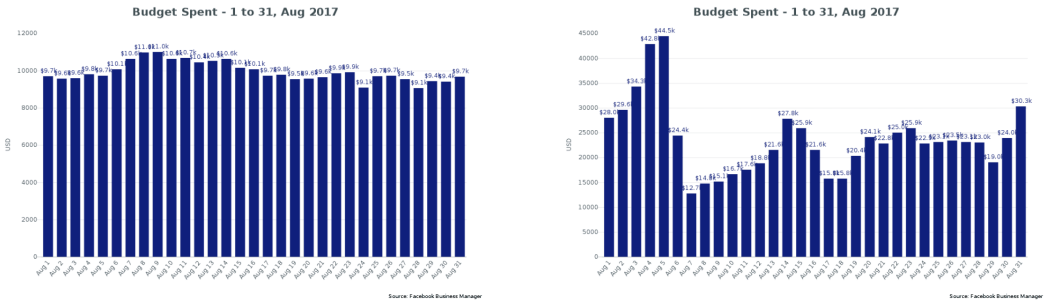
Auto-regression can also be improved by dividing a task into smaller ones: e.g. having one model

trying to predict an extracted trend component of the data and an other one the seasonal component. This allows one model to specialize in picking up repetitive patterns and working with bounded data while the other will perform auto-regression on the trend and extrapolate it.

I've started to implement a model that tries to predict the spend based on previous values. It splits the data into trend and seasonality and trains a simple linear regression on the trend component of the data and a more complex one involving Random Forest based regressions and Ridge regressions (regularized Linear regressions) for predicting the seasonal component of the data. This has yielded interesting results, allowing to predict the spend with 80% accuracy on average on testing data.

This is not bad in itself (although it could still mean wrongly estimating by $2,000 the spend for a $10,000 budget is to be expected from the model on a regular basis), but the lack of accuracy essentially comes from the fact that, in the particular case of Facebook, the user sets in advance a budget that will be the target for the time period. However, the effective amount spent for the same time period will depend on several exogenous factors (mainly the highest allowed bid, the bidding strategy, and of course, the same characteristics for every other people targeting the same audiences) which impact is hardly predictable.

For this reason, although patterns in spend data can be learned by machine learning models, they don't account for all the variations observed in spend evolution. Those effect might be more or less meaningful depending on the conditions of a particular account/campaign/..., as for instance accounts with automatically adjusted budgets seem to exhibit high regularities in their spend data. Figure 9 shows two examples of spend for two ad accounts: one having automatically managed budget, the other having a freely (to a certain extent probably) fluctuating spend.



(a) Automatically managed spend (b) Freely fluctuating spend

Figure 9: Two examples of spend data for differently managed accounts

Moreover, since every campaign or country in a single account might have very specific behavior/management, models often need to be trained on a specific and therefore limited in size dataset, which often yields cases of overfitting such as learning very specific events. One example of overfitting would be to learn the steps of Figure 10b or 10c as part of the prediction and thus predicting radical spend changes decisions that were probably made by a human agent for complex reasons. The extinction of the campaigns of Figure 10d might also be learned by such a model, although it is also likely to be a man-made decision with complex (or at least not reachable through Facebook) motivations.

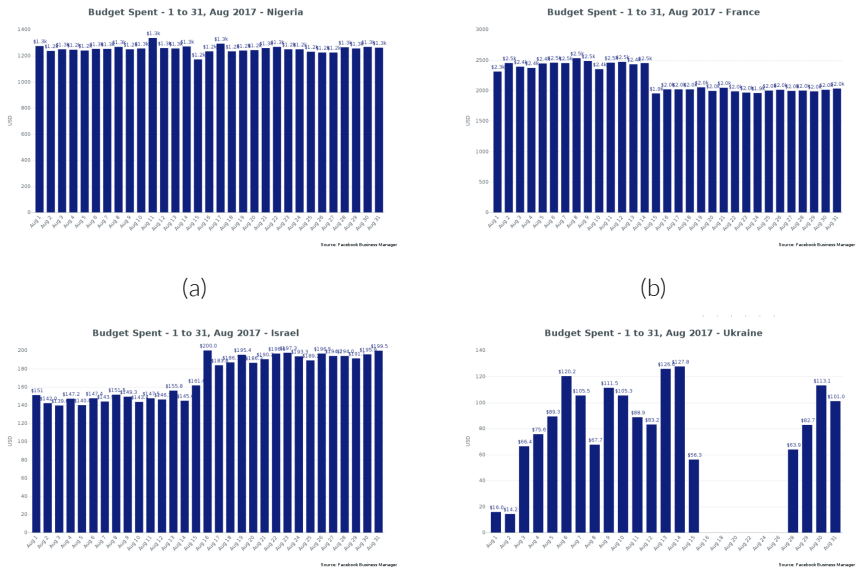


Figure 10: More examples of spend evolutions for different countries on one account

Further ideas

Clustering

I also carried out some experiments with data clustering, but didn't dig into it much because of lack of time and data. The initial idea was to cluster data from several campaigns/countries broken down by age and gender. Every point would correspond to a combination of breakdowns and clustering would be performed in the space of, for instance, 30 days of CPI/CPSU/Spend/etc data with the euclidean distance as clustering distance.

For a single country, the only results were information about the proximity of different genders in the same age range but once adding several of them, certain interesting patterns started to emerge: for instance the particularity of some countries compared to others or the proximity of some other countries (e.g. France was often close to United Kingdom, Spain to Italy).

To be complete, several types of distances should have been studied with more thoroughly engineered features in order to really detect specific behaviors and efficiently interpret the proximity/distance of countries and breakdown combinations. One other limitation of the model was the nature of the data itself: the breakdowns could have been much more granular, with data corresponding to specific audiences, publishers or any characteristic that could serve to label the data. This kind of categorical would allow for much more complex pattern recognition.

However, direct applications of those findings need some thinking since techniques such as clustering, which are part of the unsupervised learning family of algorithms, won't allow for easily automated and production-ready pipelines because they will often need a human agent to observe and interpret data after processing. However, they could yield interesting recommendation systems, with which countries, audiences or any other character is automatically associated with it's closest neighbors that are likely to behave similarly.

Computer Vision

As of today, this is more of a long shot but computer vision techniques also could be leveraged on ad creatives to yield all sort of results about CTR prediction or things like creative reporting and efficiency analysis. One recent paper from MIT AI research and Adobe, due to be presented at a conference in Toronto in October 2017 shows very interesting results for a model trained on visual attention data. They are able to predict the ”zones of attention” of a visual design and therefore guide the designer in choosing the right components, position, fonts, font sizes and images by updating in real time the amount of attention a component will trigger. An other application could be the learning of feature vectors encoding the creatives' characteristics, along with other categorical information about the ad in order to be used in Factorization Machines or more elaborate models based on it to predict their best placement and results.

There might be other papers like the one above for attention prediction, and there will certainly be more as the field of computer vision is starting to be mature enough to be applied to several previously untouched industries.