Progression overview

May 31, 2018

This is a progression overview of Outlier Factors for Device Profiling. All coding examples can be found at Github (not very tidy an the moment).

1 Mixture of Poissons

We will represent the data points with an online EM implementation of a number of Poisson distributions as described in [5].

Let a mixture model with

$$f(x, l_i) = \frac{(l_i)^x e^{-l_i}}{x!}$$

The mixture density of x:

$$p(x) = \sum_{k=1}^{K} \gamma_k f(x, l_k)$$

where K denotes the total number of mixtures available.

The values γ_k show the prior probability of a given observation being part of the mixture k.

This result can be also thought as following [2]:

Given discrete hidden variables:

$$z = (z_1, z_2, ..., z_k)$$

where $z_k \in \{0, 1\}$, $\sum_k z_k = 1$

Then

$$p(z_k = 1) = \gamma_k, p(z) = \prod_{k=1}^{K} \gamma_k^{z_k}$$

Then

$$p(x|z) = \prod_{k=1}^{K} f(x, l_k)^{z_k}$$

and so

$$p(x) = \sum_z p(z)p(x|z) = \sum_{k=1}^{K} \gamma_k f(x, l_k)$$

The likelihood of the data given the mixture, can then be described as:

$$L(\theta) = \prod_{i=1}^{N} (\sum_{k=1}^{K} \gamma_k f(x_i, l_k))$$

The goal will be to minimize the above likelihood, signifying a better representation of the dataset with the model. This can be achieved through an iterating algorithm, calculating at each step the best parameters of the madel to minimize this likelihood. This is known as EM algorithm.

There are different variations and implementations of this EM process. We will focus on an online version as described in [5, 4]. Data points are evaluated in batches and the parameters of the model are updated on the fly by an weighted average of the previous values of these parameters and the new calculated based on the new batch. These new parameter values, in the case of mixture of Poissons can be calculated as:

$$f(i|x_t, \theta^k) = \frac{\gamma_i^k f(x_t|i, \theta^k)}{\sum_{l=1}^m \gamma_l^k f(x_t|l, \theta^k)}$$

$$\gamma_i^{k+1} = \frac{1}{n} \sum_{t=1}^n f(i|x_t, \theta^k)$$

$$\lambda_i^{k+1} = \frac{\sum_{t=1}^n x_t f(i|x_t, \theta^k)}{\sum_{t=1}^n f(i|x_t, \theta^k)}$$

The above analysis can easily take place in a multidimensional space, by assuming individuality between the different dimensions.

2 Distribution

We will focus with the following dataset Los Alamos National Laboratorys corporate, internal computer network. A artificial generated dataset could be used as well, but the nature of the data should be closely taken into account.

The data used comprises of a collection of flows created at different timestamps from different users. In a first attempt we will only consider the number of flows and the bytes sent through these flows. We will also temporarily ignore time relations and drifts in the data.

To switch from irregular timestamps, events are bucketized based on time of the event.

The beginning of this dataset can be displayed in 1. As we can see most points denote a zero traffic for users in a given time.

Question: Poisson distributions on the initial dataset

I have not yet managed to represent the original dataset by the mixture of Poissons, but rather a log scaled version of it. This is due to the fact that some values are too large (e.g a flow may have a byte count of over 1e+7) and the Poisson mass function becomes too small.

This bucketizing of the data, introduces different time periods, which from now on will be noted as $epochs$. Our purpose is the evaluate the traffic of each individual host and the traffic originating from him, in comparison to that of others hosts.

As a result the whole dataset can be shown as in table 1. Each shell in this table represents a vector of size equal to the size of the feature's space.

We will then consider different approaches for this problem.

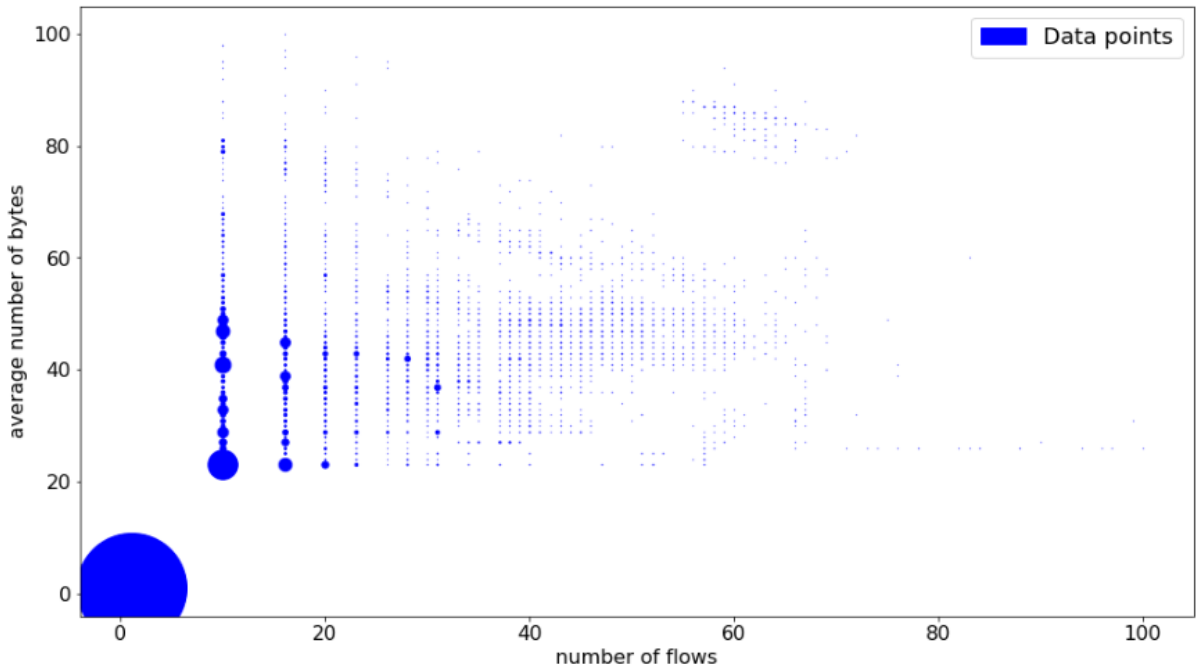

Figure 1: Distribution (the size of each dot denotes the number of points with the same value)

Hosts / Epochs	Epoch 1	Epoch 2	...	Epoch M
Host 1	r_{11}	r_{12}	...	r_{1M}
Host 2	r_{21}	r_{22}	...	r_{2M}
...	...	...	...	...
Host N	r_{N1}	r_{N2}	...	r_{NM}

Table 1: Data points after bucketizing the original data set for each individual host each individual epoch.

3 Algorithms

3.1 Pooled Poisson

The simplest approach one could follow would be representing the whole dataset with a single Poisson distribution. As already shown by a visual representation of the data, this could be extremely challenging.

In this case, the lambda value of this Poisson would simply be the average of the points across

notation	description
K	Number of Poisson distributions
N	Number of hosts
M	Number of epochs

Table 2: Major notations.

each feature space.

$$lambda_j = \frac{1}{N \times M} \sum_{i=1}^{N} \sum_{k=1}^{M} r_{ikj}$$

where J are the total number of features (only 2 in a first implementation, number of flows and average number of bytes).

We calculate the average log likelihood in this case with a new dataset. The average log likelihood is calculated equal to:

$$ALL\_Pooled\_Poisson = -13.91859$$

where ALL stands for average log likelihood.

3.2 Pooled data mixture of Poissons

In this case, we will represent the dataset with a mixture of Poisson distributions using an EM algorithm, as already described. An important parameter definitely is the value of K, denoting the number of Poisson distributions and also the initialization of them. We could use a number of approaches to tackle this:

- Greedy EM [6]: This algorithm begins with a single mixture and continues to add mixtures as long as the log likelihood of the data increases. Using a subsample, or a small number of epochs, we could use this method for an initialization of all of the Poisson's parameters (both the lambdas and the gammas). However this method has a poor complexity $O(Kn^2)$.
- We could use a variation of the kmeans++ algorithm. We initialize a fixed number of centers for the Poisson distributions. These centers are chosen randomly from the data points, depending on the distance on the closest to them already chosen center, according to a roulette mechanism.

For a simple implementation we will use a fixed number of centers K=8. With the parameters tuned from a training sample, we use a testing sample for the average log likelihood:

$$ALL = \frac{1}{N \times M} \sum_{i=1}^{N} \sum_{j=1}^{M} log(\sum_{k=1}^{K} \gamma_k f(r_{ij}, l_k))$$

In this case the result is:

$$ALL\_Pooled\_Mixture\_Poissons = -3.659311$$

We should note that the results arbitrary in contrast to the previous case. The parameters K, the batch size for the online EM and the initialization of the parameters, could lead to potentially different results. These results are however close to the minimum likelihood that can be achieved this way.

3.3 Per host mixture of Poissons

A more personalized approach could be a single mixture of Poissons per individual host. This case is even more sensible to parameter tuning. There is a large probability for over fitting.

We will use a parameter of $K = 3$ for representing the data points of each individual host. The average log likelihood in this case is:

$$ALL\_Host\_Specific\_Mixture\_Poissons = -3.261619$$

3.4 Pooled mixture of Poissons — MM 1

Similarly to case 3.2, each point is represented by a mixture of Poisson distribution on the pooled data. During this training process, the transitions between different clusters are represented with a transition matrix, of size $K \times K$, where the element t_{ij} show the probability that with a previous state i, the next state will be j. It is worth mentioning that this is a experimental probability and is derived through hard clustering. Every point is appointed to a single Poisson cluster and each row of this matrix is simply an average of these transitions.

Trough this process a Markov Model is generated. Each Markov Model is characterized by some parameters. These parameters are:

- An array π of size M denoting the prior probability for each state.
- An array a_{ij} of size $M \times M$ denoting the probability of transition from each state to the next one.
- An array b_{il} of size $M \times L$, where L the possible outcomes. Each value denotes the probability of each outcome given the current state.

In our case some adaptations have to be made. The prior probabilities π can be represented by the prior probabilities for each distribution γ. The array a_{ij} is the transition matrix. The same transition matrix also plays the role of the array b_{il}. Given a current state, a transition to a next state will also imply an observation of that state.

How to calculate the likelihood:

This is an important question. A simple solution would be the following:

1. Process a single data point x at a time.
2. For this point find the host it belongs to and through this the previous state of the host (that is the cluster of the last epoch for this host).
3. Find the probabilities $\gamma_i \times f(x, l_i)$ for each of the K distributions.
4. Based on these conclude which the next state will be.
5. Find the log likelihood:

$$log(\sum_{k=1}^{K}(transition\_matrix[previous\_state][k] * f(x, l_k)))$$

where the $transition\_matrix[previous\_state]$ denotes the probabilities of the next state based on the current one.

It is important to note that the cluster where each point belongs to, is computed merely based on the mixture of Poisson. On the other hand, while calculating the log likelihood the γ parameters are replaced by the transition probabilities.

The average log likelihood in this case is:

$$ALL\_Pooled\_Mixture\_Poissons\_MM1 = -3.55339$$

3.5 Pooled mixture of Poissons — Host specific MM 1

Similar to previous case but host specific Markov Model. In contrast to previous case, step 5 is replace with the following:

$$log(\sum_{k=1}^{K}(host\_specific\_transition\_matrix[previous\_state][k] * f(x, l_k)))$$

where the $host\_specific\_transition\_matrix[previous\_state]$ denotes the probabilities of the next state based on the current one, based on the past of each individual host.

The average log likelihood in this case is:

$$ALL\_Pooled\_Mixture\_Poissons\_Host\_Specific\_MM1 = -3.2533087$$

3.6 Pooled mixture of Poissons — Host specific MM 1 — cluster

At this point, relying on the last method we will attempt to cluster these transition matrices, in order to create clusters of similar behavior. We could use monte carlo techniques. A faster method to calculate an upper limit for the kullback-leibner dtstance, as described in [3], can be calculated as:

$$KL(P||Q) \leq \sum_{j=1}^{J} v_j(KL(a_j||\hat{a_j}) + KL(b_j||\hat{b_j}))$$

where v^T is a stationary distribution vector such that $v^T A = v^T$ and

$$\lim_{n\to\infty} \pi^T A^n = v^T$$

Question: How to find these stationary distributions

Most techniques resolve on finding the left eigenvector of the transition matrix with an eigenvalue of 1. However the second condition is perhaps not always satisfied. We will use this method from now on.

The problem with using the above method is that the kmeans alogirithm is not converging, in contrast to using the simple kl divergence for example.

A second important factor in the k-means is how to average values in order to compute the new centroids. The easiest method, as also described in [1], would be to simply take the mean of these point. This minimizes the expected Bregman divergence from the random vector.

We try this approach, using a number of clusters for the k-means. The log likelihood in this case would be:

$$log(\sum_{k=1}^{K}(centroid\_of\_host\_transition\_matrix[previous\_point][k] * f(x, l_k)))$$

The result in the same test data can be seen in table 3.

Question: How to calculate $KL(P||Q)$ where Q is zero and P is not zero

The simplest solution would be to simply ignore such cases.

An interesting method would be to compute the probability from both mass functions. That would be the following measure:

$$((P||(P+Q))/2 + (Q||(P+Q))/2)/2$$

This is only a slight improvement in comparison to case 3.4.

Number of clusters	log likelihood
1	-3.57562
2	-3.40931
3	-3.40948
4	-3.41231
5	-3.40604
6	-3.40674
8	-3.39985
10	-3.36155
15	-3.35945
20	-3.35488
25	-3.34879

Table 3: Log likelihood using the cluster centroids matrix.

Number of clusters	log likelihood
1	-3.26999
2	-3.24986
3	-3.24872
4	-3.24935
5	-3.24866
6	-3.24852
8	-3.24770
10	-3.24415
15	-3.24417
20	-3.24479
25	-3.35488

Table 4: Log likelihood using the average of the cluster centroids matrix and the individual host transition matrix.

Finally will we use a combination of the probabilities for the next state based on the current one, for the closest kmeans centroid and the host probabilities.

That is, based on a $previous_state$ the probabilities for the next state would be:

$$centroid\_of\_host\_transition\_matrix[previous\_point] \times \lambda + host\_specific\_transition\_matrix[previous\_state] \times (1 - \lambda)$$

We will use a parameter λ of 0.2, but this can be tuned later through cross validation. The results can be seen in table 4. Clustering this way does not seem to benefit in any way. Maybe consider a different clustering method, perhaps by measuring for each host the points in every cluster.

All the above results are taken with the following train and test sets. The first 1894 hosts are chosen. The train set composes of 1500 epochs, that is 1926×1500 data points. The test set comprises of the next 100 epochs for the same hosts.

References

[1] Banerjee, A., Merugu, S., Dhillon, I. S., and Ghosh, J. (2005). Clustering with bregman divergences. *J. Mach. Learn. Res.*, 6:1705–1749.

[2] Bishop, C. M. (2006). *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, Berlin, Heidelberg.

[3] Do, M. (2003). Fast approximation of kullback-leibler distance for dependence trees and hidden markov models. *IEEE Signal Processing Letters*, 10(4):115–118.

[4] Liang, P. and Klein, D. (2009). Online em for unsupervised models. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, NAACL '09, pages 611–619, Stroudsburg, PA, USA. Association for Computational Linguistics.

[5] Liu, Z., Almhana, J., Choulakian, V., and McGorman, R. (2006). Online EM algorithm for mixture with application to internet traffic modeling. *Computational Statistics & Data Analysis*, 50(4):1052–1071.

[6] Vlassis, N. and Likas, A. (2002). A greedy em algorithm for gaussian mixture learning. *Neural Processing Letters*, 15(1):77–87.