Mawazo
Mostly technology with occasional sprinkling of other random thoughts

Fraudsters, Outliers and Big Data

Posted on January 2, 2012

3 Votes

Recently, I started working on Hadoop based solutions for fraud detection. Fraud detection is critical for many industries, including but not limited to financial, insurance and retail. Data mining is a key enabler in effective fraud detection.

In this and some following posts, I will cover commonly used data mining solutions for fraud detection. I also have an accompanying open source project on github called beymani where the implementations of these algorithms will be available.

> One key data mining concept in the context of fraud is outlier. An outlier is an object that is unlike most others in a data set.

An outlier record, e.g., fraudulent credit card transaction, has attribute values that are significantly different from typical attribute values. In data mining context, fraud detection translates to outlier detection in a data set. Finding outlier is like a "needle in haystack problem". Two data mining approaches that are applicable for outlier detection is supervised learning and unsupervised learning.

In real world fraud detection, the process is far more complex than simple application of some data mining techniques. In addition to data mining, the process may involve rule engine based processing, review by domain experts etc.

Simple Example

I will be referring to a simple example of a credit card transaction throughout this post. Each transaction in my simple model has the following attributes.

1. Time of the day for the transaction
2. Dollar amount
3. Product or service purchased

The data model may be simplistic but good enough to explore the different algorithms. In real world, the credit transaction will have many other attributes.

Supervised Learning

> With supervised learning we have at our disposal a data set where the class label of each record is known, e.g. whether a transaction is fraudulent or legitimate.

We build a predictive model using the data set, so that when a new record is encountered, we can predict it's label using the model built using the training data set. Some of the techniques that may be useful is Decision Tree, Bayesian filtering and Neural Network.

> For the predictive model to be robust and reliable, there should be enough records for the different class labels. Unfortunately, the number of outliers in a data set is likely to be very small.

By it's very definition, outliers occur very infrequently, which presents a dilemma in the application of supervised learning methods for fraud detection. My focus is on unsupervised learning methods, which is discussed next.

Unsupervised Learning

In unsupervised learning, we don't have any labels associated with the data. Instead, we try to discover interesting patterns. The classic example is clustering.

> As far as outlier detection, we are interested in finding those points that fall within certain patterns discovered through unsupervised learning. If using clustering, outliers are those objects that lie as far away as possible from the cluster centers in a multi dimensional feature space.

Outliers live far apart from it's neighbors and lie in a low density of the region of the multi dimensional space.All the techniques described below is based on this fundamental heuristic.

I don't know about you, but I generally don't buy expensive jewelery on line at 2 AM on a weekday. If someone does clustering analysis of my credit card transactions, such a transaction will be found to be away from all clusters of my credit card transactions and should be flagged as a potential fraudulent transaction.

Outlier detection based on unsupervised learning generally fall under the following categories. This list is not exhaustive by any means.

1. Clustering
2. Distribution Model
3. Proximity Analysis
4. Relative Density
5. Shared Nearest Neighbor
6. Entropy

Clustering

> Outliers can be detected as a by produc eep track of objects that are worst fit as far as cluster membership is concerned, while findin

Any object that has weak members to is potential outlier. If there is any small cluster far away from other clusters all clusters could also potentially be outliers. For example, there could be multip similar to each other and as a result form a

cluster.

> But sometimes it may be hard to tell whether they are fraudulent transactions or some new emergent buying behavior of a legitimate user.

It may not be easy to distinguish between the two cases. To make the final call. we may have to perform additional analysis and or subject the results to review by domain experts. I won't be pursuing any clustering based solution for my open source implementation.

Distribution Model

If we consider probability distribution of the data, the outlier will have a low probability of occurrence. The outliers could be defined in two ways. In the first approach, we consider outliers to be those that lie a certain multiplier of std deviation away from the mean as below.

> |x -m| > n * s where
> m = mean
> n = constant greater than 1
> s = standard deviation

If the distribution is assumed to be Gaussian, the x value meeting the criteria could easily be found. The second approach is based on percentile. We find some threshold value, so that the probability exceeding the absolute value of the threshold is some predefined probability as below. Any value higher than y or lower than -y is likely to be an outlier.

> p(|x-m| > y) = p
> where p = some predefined low value probability

With both approaches , all we are saying is that the points lying in the two tail ends of the distribution are likely to be outliers. I will be implementing a model based solution, except that my approach will be non parametric. Parametric model has several issues. The distribution model is assumed to be Gaussian. Moreover, presence of categorical attributes if any, makes things complex.

> I will be following a non parametric approach in my implementation. I won't be making any assumption about the Gaussian distribution of the data.

I will have one map reduce job to get the multivariate distribution of the data.With multi variate histogram, each bucket corresponds to bounded region in a multi dimensional attribute space.

Once I have the histogram, the focus will be on identifying those histogram buckets with smallest population through a second map reduce job. The criteria is that number of objects within a bucket should be smaller than a predefined percentage of of the total number of objects.

Essentially we are selecting objects with probability of occurrence below some threshold. Objects within those buckets are likely candidates to be outliers. We are using the second approach as described above. This

implementation is available in beymani.

Proximity Analysis

> Intuitively, we know that an outlier object will be far away from most of the objects in the multidimensional attribute space. The techniques under this category involves calculating distance between each pair of points.

So it's computationally intensive. For any given point we consider the distance to it's k nearest neighbor where k is a number that is pre defined. We can take either the distance to the k th nearest neighbors or the average distance to the k nearest neighbor. We can call this distance the outlier score.

> outlier score = distance to the k th nearest neighbor OR the average distance to the k nearest neighbor

We need one map reduce job to find pair wise distance and a second one to find the k nearest neighbor for a point. Both of these map reduce classes exist in my open source project sifarish. Essentially, we solve the same problem for a recommender system where we are interested in finding similar items. The final output is a record Id of an object and it's average distance to the nearest k neighbor or the distance of to k th nearest neighbor.

Once we sort the list by the descending order of the distance, the outliers will be at the beginning of the list. For this I use a sorter map reduce available in my open source project chombo. This project contains contains some generic map reduce classes and bunch of utility classes.

The proper selection of k can be tricky, if the k is too small, then the outlier score may be low if multiple outliers happen to lie close to each other, resulting in failure to detect to outlier, resulting in false negative. On the other hand if k is too big, then objects can be mis labeled as outlier, whereas in reality they belong to a small legitimate cluster, resulting in false positive.

Proximity based solution already exists in my open source project beymani on github.

Relative Density

> Relative density based approaches are similar to the proximity based methods, but perform better when the data contains regions of different density. It takes the local context around the object under consideration into account

By varying density we mean some clusters are densely packed and others not so. Density is defined as the inverse of the average distance to the nearest k neighbor. As density goes down the probability of an object being an outlier goes up.

> To properly handle regions of different density, we generally use relative density. which is defined as the ratio of the density of an object and the average density of it's nearest k neighbor.
> density(x) = 1 / (sum(dist(x,y)) / k)
> rel density(x) = density(x) / (sum(density(y)) / k)

In addition to the two map reduce jobs mentioned for proximity analysis, relative density computation will

require two additional map reduce jobs. This is the most computationally intensive among all the algorithms discussed so far. I will provide more details when I get around to implementing it.

Shared Nearest Neighbor

This approach is similar to the relative density method. It also takes into consideration, the local context. This approach also works better for higher dimensional data.

> First we find the nearest k neighbors for all objects. Next for any pair of objects that are in each others nearest neighbor list, we find the number common neighbors. Higher the number of common neighbors, greater the similarity between the the two objects.

For a given object we find the average similarity with the nearest k neighbors. Then we sort the objects by the ascending order of the average similarity. Potential outliers will appear at the beginning of the list.

Entropy

Fraudulent transactions increase the entropy of our data set. What do I mean by that?

> Entropy is measure of the homogeneity of data. For perfectly homogeneous data, the entropy is 0. Entropy goes up as the data become more random.

So when an oulier gets added to data, entropy goes up. In other words, an outlier makes a data set less homogeneous. Entropy is defined as

> entropy = sum (-p(x) log(p(x)) where p(x) = probability of x

Given a data set, we want to pick those objects whose removal cause the largest drop in entropy. This will require a map reduce which calculates entropy. We remove one object at a a time and calculate entropy of the remaining data set.

Standard deviation is also good measure of the homogeneity of data. However with a mixture of numerical and categorical attributes, there is no meaningful definition of standard deviation. Entropy does not pose any such problem. Another option is to use Gini Index defined as below.

> gini index = 1 – sum(p(x) * p(x))

Wrapping Up

The question that comes up is which technique to use. There is no clear answer. The essential characteristics of an outlier are as follows.

1. It occurs infrequently
2. It's far away from other objects

Distribution model based approaches are based on the first condition but not the second. The focus is on the temporal aspect of an object. Proximity based techniques rely on the second criteria, which is related to the features of an object. It can be made to satisfy the first condition by appropriately choosing the number of nearest neighbors. Implications of the two parameters are as below.

	Far away from others	Close to others
Rarely occurring	Very likely outliers	Potential outliers.
Not rarely occurring	Potential outliers	Not outliers

Objects that are not so rare but far from other objects will typically be part of a small cluster. It's ambiguous and it may be difficult to label them. The small cluster could represent a small set of outliers that are similar to each other or they could be legitimate objects.

Objects that rare but close to neighbors, will typically lie around a cluster boundary. It may be difficult to interpret them correctly.

For effective outlier detection, we need a combination of tenchniques based on the temporal and feature aspects of an object. We might use a model based approach to narrow down on rarely infrequently objects and then apply proximity analysis on them.

Going Forward

I will have some follow up posts, as I add these implementations to beymani. They will have details on Hadoop map reduce implementations. As I mentioned earlier, some implementations are already part of beymani

Related

Fraudsters are not Model Citizens
In "Big Data"

Real Time Fraud Detection with Sequence Mining
In "Big Data"

It's a lonely life for outliers
In "Big Data"

About Pranab

I am Pranab Ghosh, a software professional in the San Francisco Bay area. I manipulate bits and bytes for the good of living beings and the planet. I have worked with myriad of technologies and platforms in various business domains for early stage startups, large corporations and anything in between. I am an active blogger and open source contributor. I am passionate about technology and green and sustainable living. My technical interest areas are Big Data, Distributed Processing, NOSQL databases, Data Mining and Programming languages. I am fascinated by problems that don't have neat closed form solution.
View all posts by Pranab →

This entry was posted in Big Data, Data Mining, Fraud Detection and tagged fraud, Hadoop, outlier. Bookmark the permalink.

12 Responses to *Fraudsters, Outliers and Big Data*

Pingback: *Fraudsters are not Model Citizens | Mawazo*

Pingback: *It's a lonely life for outliers | Mawazo*

Pingback: *Relative Density and Outliers | Mawazo*

Jagaran *says:*
January 22, 2013 at 1:03 pm

I am not able to run the code.
Please let me know the steps ?

Reply

Pranab *says:*
January 22, 2013 at 1:08 pm

Jagaran
I have 3 implementations. I am not sure which one you want to run. I will create a tutorial document for the proximity based implementation with the detailed steps and check it into resource directory in github. In the mean time you can go through this post
http://pkghosh.wordpress.com/2012/06/18/its-a-lonely-life-for-outliers/

Reply

sundeep *says:*
February 11, 2013 at 9:15 am

Hi
I am doing a final year project on classifying against a fraudulent credit card transaction dataset using java. I am in the early stages of looking for an appropriate dataset (.xml or .csv or .arff files) containing legitimate and fraudulent data transactions which I plan to preform a range of classifications from a simple GUI based java program written on NetBeans. At the moment I am having difficulties finding a suitable dataset.
If you can give me any advice on ways of achieving this result or any changes to the initial idea that would be much appreciated.

In these early stages of the project, I need to find a dataset otherwise I cannot continue any further.
Thanx

Reply

Pranab *says:*
February 18, 2013 at 5:07 pm

You will find a script in my project beymani at this location. It generates credit card transaction data
https://github.com/pranab/beymani/blob/master/resource/cct.rb

Reply

madhu *says:*
March 28, 2013 at 8:11 am

can u tell me how to use my data into the program, coz it may b both in text and number form, so is there any technique to use it?

Reply

Pranab *says:*
March 28, 2013 at 9:47 am

Madhu
In the resource directory of the github project, you will find JSON meta data file. In there you define your meta data. including data type for each field. The tutorial files in the same directory might also be helpful

https://github.com/pranab/beymani/tree/master/resource

Reply

madhu *says:*
March 28, 2013 at 11:53 pm

Thank You , I was thinking to implement fraud detection using unsupervised learning through clustering of the abnormal transactions. As you suggested of time of transaction in your blog, in addition to it, I was thinking of some more parameters like number of transactions, time between two transactions that can also be regarded. Can you suggest me some more parameters? Thanx.

Reply

Pingback: *Real Time Fraud Detection with Sequence Mining | Mawazo*

Pingback: *Real Time Fraud Detection with Sequence Mining | Big Data Cloud*