CREDIT CARD TRANSACTIONS ANOMALY DETECTION
A PRACTICAL GUIDE ON MODELING CUSTOMER BEHAVIOR

I. Introduction
Most people think deep neural networks\(^1\) or any other modern machine learning techniques are capable of doing almost anything at least in specific single domains. Even if we consider this is a correct claim, the problem is we usually do not have access to enough data (specially labeled) to train a neural network and expect that magic. That is why most of the successful machine learning projects which catch our attention come from giant companies who have enough money to pay for data and hire hundreds of people to label the data. Now the question is, what can we do if we do not have that much money or labeled data, but still want to or have to use machine learning techniques? This technical white paper is the result of real experience on different projects in medium to large sized companies. Companies that just give us access to millions of transactions per hour and ask to find the anomalies!? Solving these kinds of problems is the real magic!

II. Knowledge usage
All we need to extract knowledge from any source of data is clustering, but the problem with using clustering alone is that it does not guarantee the obtained knowledge is what we are looking for or is understandable with our already trained mind at all. Our minds already have been trained and contain millions of patterns; these are the patterns that give meaning to our life, the reason we are looking for anomalies in credit card transactions is also related to some of these patterns! That is why we usually prefer using classification because, in that way, we push the knowledge extraction process - deep neural network - to find the solution the way our minds like it. Now we need to fill the gap of having no labeled data with something else, which is our previous knowledge, but not in form of labeling the data, because it has costs. So, we see in both cases we have to use our existing knowledge by either applying it during the process of data labeling or using the knowledge during the system design.

III. Credit card transactions
You get your first credit card, and after a year of using it, you decide to go on a vacation. You go to a different country, say from America to Europe; you think if you use your credit card, it might not get through or might get declined. However, nothing happens it works! Now the question is, does their fraud\(^2\) detection system works at all? Or it is that much intelligent that knows it is you, spending the money? There is no way they can make sure that it is you who is using the card in another country unless they track all your life event and know that you are going to Europe. It means even if someone steals your credit card and goes to Europe and uses it, bank’s fraud detection system cannot necessarily identify whether it is legitimate or not unless the thief changes your usage pattern. In fact, if the fraud detection system wants to catch the fraudulent transaction the moment it happens, the system gives you many false positive, so these - anomaly or fraud detection - systems usually wait to collect enough data and then when they enough evidence raise anomaly alarm.

---

\(^1\) Or recurrent neural network when temporal patterns are the problem.

\(^2\) “Fraud” is not an “anomaly”, but we use it here as like as anomaly. Frauds are usually complicated and already planned like money laundry.
We can analyze this problem from another point of view, the behavior of people in using their credit card is so diverse and different, at any time we might use the credit card in a way we have not used it before. Even if we think of processing of just four dimensions, "time", "location", "amount" and "usage category", the distribution of the card usage is still so sparse, that we cannot say with enough confidence that a single transaction is a fraud or not, unless it breaks the owner's behavior in a very wrong way. For example, the owner lives in America and never has used the card for buying a beer and has never used the card after 8:00 pm; then the card gets used to buy a beer in Europe at 12:00 am. That could be a good sign of anomaly and might worth raising the alarm.

IV. Behaviour of the owner
The key to catching anomalies in credit card usage is learning the behavior of the owner as much as possible. The more we learn from the owner's usage behavior and mindset the less false positive or false negative we have in our detection system. To continue talking about the problem let us consider we have that four dimension we talked about in the previous section:

1. Time
2. Location
3. Category (usage)
4. Store (type)
5. Method
6. Amount

So, every transaction is like (Time, Location, Amount, Category) or (t, l, a, c). Note that in this paper we are just trying to give practical guides to build a model for each customer's behavior and are not going to compare it with other customers or use any peer group analysis, demographics, breakpoint analysis, etc. Now the problem is if we had enough labeled data we would use some deep neural network to train a model, but since we do not have that kind of data, we need to use another method to catch the anomalies. Regardless of the way we may use to build the model, first we need to know the dimensions of the transactions better, don't forget because of the nature of the card transactions a simple clustering or even using of an autoencoder, does not give us an acceptable result.

V. “Time”
Time is the most crucial part of every transaction; it gives us at least the below extracted features:

1. The day of the week the customer goes for shop
2. The time of the day
3. If we divide a month into three parts, we can also determine if the customer uses the card in first or second or third part of the month

These are not the only features we can extract from a single time of a transaction. We might even be interested in quarter or month of the year, etc. But using these features definitely requires more training data so for this model let us just consider the above features. We still can extract more information from time. Take a look at this document “ON MEASURING ABNORMALITY”, it basically says that, to learn the behavior of a dynamic system, we need to update the built model periodically so, the time of the samples should have an equal effect on the built model, some of them have more effect some less. In credit card transactions, it means neither the very past nor the very recent transactions should be considered as customer's behavior seriously.

The reason for the very past transactions is that customers legitimate behavior might change, and the idea behind very recent transactions is that those transactions could potentially be fraud themselves. We use this idea when we want to calculate the score for a transaction to see how much it is legitimate.

VI. Underfitting and Overfitting
Before going forward, let us talk about the underfitting and overfitting, which we already have read about them in textbooks. Textbooks usually show us the error-complexity graphs and describe us in how we might come up with high bias or high variance model as the below picture. However, here we need some intuitive understanding of overfitting or underfitting, this is
important especially when we want to use different technologies in our model.

Let us consider the time of the transactions. As we saw, we extracted a feature called time of the day, now if we think of the time of the day as hh:mm or even hh the model is overfitted, the first one gives us like 1440 different daily time quanta while the second one gives us 24 quanta. Both of them give us overfitted model because people do not count on minutes or even hours for using their credit card. Perhaps the best choice is dividing a day into four time-intervals like morning (6:00 am - 12:00 pm), afternoon (12:00 pm - 6:00 pm), night (6:00 pm - 12:00 am), and midnight (12:00 am - 6:00 am). The over fit model gives us many false positive, or false negative results and these kinds of models need too much data to get trained. To analyze the time of the day we can also use clustering, we can run a clustering (like k-means) over the available time of the days and find out the time of day clusters each customer usually uses the credit cards. In this case, choosing a large number of k gives us an overfitted model.

Underfitting happens when we generalize too much so that our model cannot distinguish between legitimate or fraud transaction. Again, as for the time, if we do not consider the time of the day that transactions happen and limit ourselves to just day of the week we never know enough about the customer's daily time usage behavior. So, if someone steals the card and uses it early in the morning we never find it out.

VII. “Location”
We are going to use the same idea we used for the time for the location. Here we have more options, we can use the general already known the hierarchy of Country, Province, City, District, etc. or we can use some hierarchical clustering model based on geolocation information we might have access to. All we need to know is that how much the card owner is like to use the card in different parts of the city or, different cities of the country, or countries at all, so let us assume we extract the below information from the location field:

1. Country
2. Province
3. City
4. District

Note that we are not worried about having almost 200 countries or 4,500 cities in the world, nobody travels to that much countries or cities. Here we also have to consider the overfitting and underfitting problem too, so considering streets might give us an overfitted model while using just country and city might end to an underfitted model.

IIIX. “Category”, “Store” and “Method”
Defining the category of the usage is a problem for itself, which we are not going to get involved into that, but banks usually use simple classifications or NLP algorithms to define the category based on the transaction description. The result gives us labels like "bills & utilities", "commute", "education", "food & dining", "groceries", "health", … we just need to make sure again not using too many categories.


And finally, method is the way the customer uses the credit card; we can use simple categories as below. Note that in the first three methods the card is presented at the payment type while the in the next three ones is not:

1. Payment Terminal (POS)
2. Contactless
3. ATM
4. Internet purchase

Kamran Vatanabadi, December 2017
5. Other methods using CVV
6. Other methods without CVV

IX. “Amount”
Depending on the modeling method we want to use, we can use the amount as a continuous feature or do some quantization or clustering over it. As we said, the model should understand the mindset of the user, consider a credit card that belongs to a business which always gets used for purchases above 500 dollars, now buying a coffee with this card is a good sign of potential fraud. If we want to quantize amount for any reason, just note that with quantization we might lose crucial information. In the business credit card, if we choose 1,000 as a unit of quantization, we are going to have many false negatives.

X. Using “velocity”
We already talked about the features we can extract from the time of a transaction, there is something else we might be interested in using that, which is called velocity. It is a measure that shows how many times the card has been used in some unit of time or location. So, for example, the model might want to learn how many times a day, the owner uses the card or the time between to subsequent usages, or even the time between to subsequent card usages per distance between the locations or IP addresses. These are some high level extracted features that could let the model learns the detailed behavior of the usage. Note that if we do not use these features, we already have lost temporal information on card usage. So, for our model let us assume that we only are interested in using location velocity, which gives us the idea of how fast the user changes the location and uses the card.

XI. Selected and extracted features
Up to now, we prepared some features to build our model based on them, here is the summary:

1. dayOfWeek {Saturday, Sunday, Monday, ...}
2. timeOfDay {Morning, Afternoon, ...}
3. partOfMonth {First, Middle, Last}
4. country {Canada, Mexico, ...}
5. city {Toronto, Cambridge, ...}
6. district {Downtown, Lakeshore...}
7. category {Bills & Utilities, Education, ...}
8. store {Super market, Book store, ...}
9. method {POS, Contactless, ATM, ...}
10. amount (continues variable)
11. locationVelocity (continues variable)

Do not forget preparing the above data from the given transactions is a whole project for itself and when we get our features ready, we still can use any binary classification if we have labeled moreover kind of balanced positive and negative data. But if we do not, we have many options to build a model of credit card’s owner usage behavior that might even work better than using classification.

XII. Modeling options
We have 11 features for every transaction, two of them are continues numeric variable, and the rest are symbolic with cardinality between three up to 20 or even 50! So, it is evident that if we put the transactions in an 11 dimensions space, simple clustering does not give us a good result4, but still, we can use hierarchical clustering. Let us use more of our already known knowledge to build a better model. We assume the single velocity feature we introduced, contains all the temporal information we need for the model, if not we need to add more temporal features, do not forget we do not have access to labeled data. We have two different options, to deal with the transactions5:

1. Mixing up customers transactions together and building a shared model.
2. Building separate model for each customer.

---

3 We didn’t talk about it, but a comprehensive solution should also consider the IP address of the user. Either via the user’s device or any third-party terminal.

4 The general idea of anomaly detection using clustering is determining the cluster of the given sample; then if that cluster has a low population, then it could be an anomaly.

5 Choosing something between is also possible.
And, regardless of the selected options, we can choose one of the following models:

1. Updating the whole model periodically and use it just for prediction.
2. Doing every required process on the fly as a new transaction happens.

As we talked before, the problem with mixing up all customers' transactions is that we basically mix up hundreds of different mindset and behavior and build a generalized form which might have a significant distance from each customer. However, if we choose to do that, then using an autoencoder might be a useful method. When we choose the mixed transactions approach, automatically we have to choose the periodic updating of the model, because we have a massive number of transactions every moment.

XIII. Instance based learning
What happens if we decide to build a model for every single customer and then do the anomaly detection process on the fly? Why on the fly? Because in this scenario we are not facing massive number transactions for each model every hour, 1, 2, 3, maximum 30? Here are some pros ( √ ) and cons ( × ) of this approach:

1. √ Customers have their specific model describing their behavior. So, the model has a better chance to get fit and describe the customer's mindset.
2. √ Building or updating the model is so fast because we are dealing with just one customer's transaction.
3. √ An in-depth analytic could be done the moment a new transaction happens.
4. √ There is no need to have a massive and high-volume batch model updating process.
5. √ Instance-based learning models are often very accurate.
6. √ We just update or process models that their corresponding credit card are active.
7. × Since we don't use other customer's data, we need to wait for a while to collect enough data and learn the behavior of each customer.6
8. × We might need more disk space to store each customers' behavior data or processing power to update model or detect anomalies on the fly.

XIV. The learning and detection process
Any learning process requires comparison, and any comparison needs a method to calculate the distance between two objects, we usually use this in either clustering methods or classifications. For a moment to make things easier consider both velocity and amount in our 11-dimension space is also quantized values.

Suppose we have access to transactions for every customer and want to build a model for every one of them to describe their behavior, the only thing we know is that these transactions are mostly legitimate, if not all of them. First, we need to build the process of feature extraction that maps every raw transaction to our 11-dimension space. Every mapped transaction shows different behavior of the customer; one with high reported value on Friday nights show the customer usually have dinner out and another one might explain the customer usually pays utility bills at the end of the months. The key is learning enough from the customer so that if someone else uses the credit card, we distinguish any behavior change.

After mapping all the transactions to our multi-dimensional space, we have different instances for each behavior like in figure 2. As we see in figure 2, every behavioral pattern has a corresponding score. The simplest way to calculate the score is counting them as we observe any behavioral pattern, but as we said it is not a good idea, it is better to consider the time of the observation and consider that old and recent pattern should not be taken so seriously. So, instead of counting instances, we calculate the sum of a number - between 0 and 1 - that shows how much the observed pattern affects customers behavior .

---

6 We can solve this by using a peer group initial model.
7 Check this document “ON MEASURING ABNORMALITY”. 
Now the question is how we can calculate the distance of a new observed pattern ($T$) from the previously experienced ones. In figure 2, a new transaction causes observation of pattern $T_2$, precisely as behavior $B_4$, since we already know the score of the $B_4$, we just need an anomaly index function showing how much observing a $B_4$ pattern is an anomaly. However, what if the observed transaction gives us a new pattern that we have never experienced it before? Like $T_1$?

We should not forget that here, the distance function could not be like Euclidian distance because the dimensions are quantized and also are not necessarily numeric. So, the behavioral distance between the observed patterns $T_2$-$B_1$, $T_2$-$B_2$, and $T_2$-$B_3$ are all equal to $1/2$. Figure 2 also shows that the distance between $T_2$ and $B_4$ is zero, that means these two behaviors have nothing in common.

XV. Anomaly Index

One way to determine the anomaly index of a transaction is calculating the expected value of all similar behaviors as shown in 3.

$$distance(T, B_i) = 1 - \frac{\text{Number of equal features}}{\text{Total number of features}}$$

$$anomalyIndex(T) = \frac{\sum_i distance(T, B_i) \times S_i}{\sum S_i}$$

3- Calculating anomaly index for any given transaction

Here in (3), we calculate the distance between two different behavior as what they have in common over the total number of behavior's features. So, in our example with 11 features, if two behaviors have six common feature values, then the distance between them is $1-6/11=5/11$. If all of their feature values are equal, the distance is $1-11/11=0$, and of course, no equal feature values give a distance of $1-0/11=1$. But to calculate the overall behavior similarity or anomaly we need to calculate the expected value over the entire available behavior of the customer, as shown in (3).

Figure 4, shows how we can score each transaction based on the observing time. As you see, and we discussed, recent and very old transactions should not be considered as the behavior of the customer.

---

8 Here we have an implicite assumption that the risk or impact factor for all dimensions is equal.
Here we have to take care of velocity and amount. In our distance function, we assumed these two features are quantized like the others. If we do not want to quantize the amount and velocity, we have to make a change in the defined distance function in (3). The difficulty gets back to the numerator of the distance calculation fraction and has nothing to do with the denominator. Quantized values are either equal or not, 1 or 0, but for continuous real values we have to calculate an index showing the similarity between the two values. To deal with continuous values of velocity or amount we have many options, here is one example.

To deal with continuous values of velocity or amount we have many options, here is one example. Suppose we keep two individual lists for every quantized behavior entry like below:

<table>
<thead>
<tr>
<th>quantizedBehaviorId</th>
<th>score</th>
<th>amounts</th>
<th>velocity</th>
</tr>
</thead>
<tbody>
<tr>
<td>B1</td>
<td>S1</td>
<td>a1, a2, a3, ...</td>
<td>v1, v2, v3, ...</td>
</tr>
<tr>
<td>B2</td>
<td>S2</td>
<td>a1, a2, a3, ...</td>
<td>v1, v2, v3, ...</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>Bn</td>
<td>Sn</td>
<td>a1, a2, a3, ...</td>
<td>v1, v2, v3, ...</td>
</tr>
</tbody>
</table>

5- Sample data structure to keep and update behavior model.

The process of updating model is quite straightforward; we can do it every time a new transaction comes or every week or month. Here we implicitly assume that time does not affect the amount and velocity. We need to determine how much the new incoming amount or velocity is similar to this window’s history. We have many options; we can calculate z-index of the new amount or velocity and naturally if it is less than or equal to three, assume it scores 1, and if it is higher than three assume it scores 0. We can also calculate a fuzzy value between 0 and 1 to describe the similarity score.

If we update this table once in a while like every week or month, we can run a clustering over these two series of data and just keep their clusters’ centers and their population. Now when a new data comes, just look which cluster it belongs to, and the score is the cluster’s population percentage. So, as you see it is not a big deal to calculate the score for the non-quantized feature too.

Another processing hint is that if we look at figure 4, we see that only in a specific time window we have non-zero scores, which can be a six months or 12 months or two years, etc. So, we do not need to process the whole available data if a new transaction comes or when we want to update the table (5).

Moreover, finally note that if for any reason, we want to give more value to some features or dimensions, like amount, then all we need to do is using a weight parameter for each feature and consider it in calculating the distance⁹, as shown in 6:

\[
\text{distance}(T, B_i) = 1 - \frac{\sum_{\text{equalFeatures}} \text{featureWeight}}{\sum_{\text{allFeatures}} \text{featureWeight}}
\]

6- Modified weighted distance function for the time when some dimensions are more important than the others.

XVI. Example

Before continuing with risk calculation, let us give an example which might give us a better understanding of the anomaly detection calculation.

Here in figure 6, we are modeling a customer’s behavior containing four same weighted features, F1, F2, F3 & F4. Processing training data gives us four different behavior with their corresponding scores shown in Table 1. Table 2, is the new observed customer’s behavior. Table 3 shows the similarity and distance calculation for each of the already observed behavior. So, as you see this new behavior is most similar to B4 and has nothing in common with B3; the overall gained score is also calculated there. Also, finally in Table 4, we calculate the anomaly index.

⁹ Models usually consider velocity the most crucial feature.
6- Simple behavior anomaly index calculation.

XVII. Choosing a threshold
The key to determining if the calculated anomaly index is high enough to raise the alarm or not is the threshold. In the given recent example, if we choose a threshold of 0.6 the transaction is an anomaly if we choose a threshold of 0.7, it is not, which one is correct? We can choose a static value for threshold, for a behavior model containing four features we might say: "We accept overall behavior similarity of %25", then it gives us a threshold of 0.75 for anomaly index. Note that this is not an adverse selection, just think about being happy might have many different signs, but if you see two or three of them, you label the observing person as happy. Keep in mind, choosing high similarity leads to low anomaly threshold and perhaps a high number of alarms unless the customer has a very consistent and limited behavior.

Since people's behavior are different, some like to explore different shops or cities and some not; it is better to calculate a specific threshold for every customer. Again, there are many options to estimate a better value for a threshold for each customer, here is an example. We implicitly accepted that all the transactions in the training window are legitimated, check figure 4, these transactions have a positive score. So, we can calculate the anomaly index for all of these transactions and choose the threshold based on their maximum, or if we want to be a bit cautious, choose it a bit lower\(^{10}\).

\[ \text{totalRisk} = \text{risk}(T) \times \text{cost}(T) \times \text{anomalyIndex}(T,M) \]

7- The total risk calculation

Note that in (7), \( \text{anomalyIndex} \) shows how much the behavior could be fraudulent, and it is a function of both model and transaction. The \( \text{cost} \) is a measure for the damages which we assume is just related to the transaction\(^{11}\), and the \( \text{risk} \) is a factor that defines how much the environment could be dangerous, so it also just is a function of the transaction. Using formula (7) to generate alarm reduces unnecessary alarms because

10 We can consider the threshold as: \( \mu + k \sigma \) for \( k = 3 \) or 4, ...

11 We can also consider cost as a function of both transaction and model, \( \text{cost} = f(T,M) \)
most of the time the transaction behavior might be high enough to pass the threshold, but if at the same time the risk and cost are low the total risk still stays low. Figure 8, gives an initial idea of how the anomaly detection system's modules.

I think the most important thing we - as engineers - should never forget is to be realistic and be able to distinguish the difference between science and engineering. We are working in the real world and real-world data, especially for medium and large companies, are not like what we commonly have access to when testing and learning these algorithms. The real world's requirement is much more complicated than those tests or doing some mathematics on the paper.

![Figure 8](image.png)

**Figure 8** - Simple idea of anomaly detection modules

**XIX. Conclusion**

What we tried to describe here was giving some ideas of how we can build an anomaly detection system for credit card fraud transaction using instance-based learning method to learn the behavioral patterns of each customer. The benefit of that is we do not mix up different generalize different customers mindset to build a model.

If you ask me what was the most important outcome of this discussion? I would say, "We cannot build a system that catches fraud transactions just by using a single machine learning algorithm, at least at this stage of machine learning techniques we know."

We also tried to give almost a comprehensive information of how you can build such a system, and the good thing about this system was that it needed no labeled data to work. However, still, we always can use and get the benefit of both supervised and unsupervised learning algorithm.