Machine learning algorithm and sampling methods evaluation on Credit Card Fraud Detection with focus on SVM Kernels

Author: Patrick Schneider

Date: 18.12.2018

1. General

1.1 Abstract

The growth of e-commerce increases the money transaction via electronic network which is designed for fast and easy money transaction. This creates a greater risk of credit card fraud which can happen by many types as by stolen cards. In this report I evaluate the Support Vector Machine (SVM) based method with multiple kernel involvement in comparison with other Machine learning algorithms. This dataset has been collected and analysed during a research collaboration of Worldline and the Machine Learning Group of ULB on big data mining and fraud detection. The dataset is highly unbalanced, the positive class (frauds) account for 0.172% of all transactions. Based on this data characteristic another step will be to analyse different sampling methods to improve the model performances. The results reflect the application of different SVM kernels, machine learning algorithms on several sampling algorithms in an experimental environment with a conclusion over the importance of choosing the right optimization metric.

1.2 Introduction

Machine learning vs. rule-based systems in fraud detection
The machine learning (ML) approach to fraud detection has received a lot of publicity in recent years and shifted industry interest from rule-based fraud detection systems to ML-based solutions.

Fraudulent activities in finance can be detected with rule-based approach by looking at on-surface and evident signals. Unusually, large transactions or the ones that happen in atypical locations are verified. Purely rule-based systems are using algorithms that perform several fraud manually written detection scenarios. Nowadays systems apply about many different rules on average to approve a transaction. They require adding/adjusting scenarios manually and can hardly detect implicit correlations. Further, rule-based systems often use legacy software that can hardly process the real-time data streams that are critical for the digital space.
ML-based fraud detection make use of subtle and hidden events in user behavior that may not be evident, but still speak for a fraudulent action. Machine learning allows for creating algorithms that process large datasets with many variables and help find hidden correlations between user behavior and the likelihood of fraudulent actions. So, the challenge for industry players is to implement real-time claim assessment and improve the accuracy of fraud detection.

General statistics
Credit card fraud is a wide-ranging term for fraud committed using or involving a payment card, such as a credit card as a fraudulent source in a transaction.
In the US 2015, the number of customers who experienced fraud hit a record 15.4 million people. Criminals stole about $6 billion from banks last year. A shift to the digital space opens new channels for financial services distribution. It also created a rich environment for criminals.[1]

In recent studies, identity crime in Australia costs upwards of $1.6 billion each year, with the majority lost by individuals through credit card fraud. More alarmingly, identity crime continues to be a key enabler of serious and organised crime, which is estimated to cost Australia around $15 billion annually.[2]

According to Javelin Strategy & Research, fraud also impacts banks that provide online payments service. For instance, 20 percent of customers change their banks after experiencing scams.[1]

The recent study shows that false declines make merchants lose about $118 billion per year while clients' loss is about $9 billion per year. It's the largest area for fraud in financial services. So fraud prevention is a strategic goal for banking and payments industries.

1.3 Previous Work

[3] Sitaram patel, Sunita Gond
This thesis proposes the SVM based method with multiple kernel involvement. The simulation result shows improvement in TP (true positive), TN (true negative) rate, & also decreases the FP (false positive) & FN (false negative) rate in their specific scenario for fraud detection.

[4] Andrea Dal Pozzolo, Olivier Caelen , Reid A. Johnson , Gianluca Bontempi
Study analytically and experimentally how undersampling affects the posterior probability of a machine learning model. Undersampling is a popular technique for unbalanced datasets to reduce the skew in class distributions.

[5] A. Srivastava & A. Kundu
Presented a HMM double embedded stochastic process with two hierarchy levels. It is a stochastic processes. A Hidden Markov Model has a finite set of states monitored by a set of transition probabilities. In a particular state, observation or an output generated according to an associated probability distribution. It is only the output & not the state that is visible to an external observer.

[6] Y. Sahin & E. Duman
Demonstrates the advantages of applying the data mining techniques including Decision Trees & Support Vector Machine (SVM) to the credit card fraud detection problem for reducing the banks risk. The results show that the classifiers & other Decision Tree approaches outperform SVM approaches in solving the problem under investigation.

2. Own Work

2. 1 Theory

2.1.1 Support Vector Machine

The SVM is a machine learning tool based on statistical and mathematical foundations concerning generalization and optimization theory. It offers a technique for many aspects of data mining including classification, regression and outlier detection. SVM makes use of the Vapnik statistical learning theory and the intersection of kernel methods and maximum margin classifiers. SVMs have been successfully applied to many real-world problems such as face detection, intrusion detection, handwriting recognition, information extraction and others. SVM is a famous method due to its high generalization capability and its ability to handle high-dimensional input data.

Linearly separable case
In the linearly separable case, there exists on or more hyperplanes that may separate the two classes represented by the training data.

Non-linearly separable care
In the non-linearly separable case, it is not possible to find a linear hyperplane that separates all positive and negative examples. to solve this case, the margin maximization technique may be relaxed (c hyperparameter) by allowing some data points to fall on the wrong side of the margin, i.e. to allow a degree of error in the separation. Slack Variable are introduced to represent the error degree for each input data point.

Kernel trick
The kernel trick allows the computation of the vector product Φ(x_i)T Φ(y)= K(x,y), where K is a corresponding kernel function. Computing the vector products in the lower dimensional input space while solving the classification problem in the linearly separable feature space is a major advantage of SVMs using a kernel function. The dual problem then becomes to:

find α that maximizes $\sum_i \alpha - 1/2 * \sum_i \sum_j \alpha_i \alpha_j y_i y_j K(x_i, x_j)$

$\sum_{i=1}^{N} \alpha_i y_i = 0$, subject to $0 \le \alpha_i \le C, \quad \forall i$ and the resulting SVM takes the form:

$$f(x) = w^T \phi(x_i) + b = \sum_{i=1}^{N} \alpha_i y_i K(x_i, x) + b$$

The following kernel were evaluated in this project.

Kernel	Description
Linear	Basic linear Kernel
Linear weights	L2 Regularized Linear Support Vector Machines with Class Weights
Poly	Polynomial Kernel 2 Degree
RBF	Radial Basis Function Kernel
RBF Weights	RBF Weight is the same as RBF but also considered class weights
RBF cost	RBF cost is the same as Radial but sigest is run inside of each resampling loop.

2.1.2 Decision Tree

A decision tree is a Machine Learning algorithm able to fitting complex datasets and performing both classification and regression tasks. The idea behind a tree is to search for a pair of feature-value within the training set and split it in such a way that will generate the best two child subsets. The goal is to create branches and leafs based on an optimal splitting criteria (tree growing). Specifically, at every branch or

node, a conditional statement classifies the data point based on a fixed threshold in a specific variable. To make predictions, every new instance starts in the root node (top of the tree) and moves along the branches until it reaches a leaf node where no further branching is possible.

2.1.3 Random Forest (Bagging)

The Random Forest method introduces more randomness and diversity by applying the bagging method to the feature space. Instead of searching greedily for the best predictors to create branches, it randomly samples elements of the predictor space which means adding more diversity and reducing the variance of the trees at the cost of equal or higher bias. This process is also known as “feature bagging”.

2.1.4 XG boost (Boosting)

Boosting reduces variance, and also reduces bias. It reduces variance because it is using multiple models (bagging). It reduces bias by training the subsequent model by telling the model what errors the previous models made (the boosting part).
In contrast to bagging techniques like Random Forest, in which trees are grown to their maximum extent, boosting makes use of trees with fewer splits(shallow tree algorithm).

2.1.5 Sampling Methods

Down-sampling: Randomly subset all the classes in the training set so that their class frequencies match the least occuring class.

UP/Over-sampling: Randomly samples and replaces the minority class to be the same size as the majority class.

SMOTE-sampling: Synthetic Minority Over-sampling Technique: The oversample part takes a sample from the dataset, and consider its k nearest neighbors (in feature space). To create a synthetic data point, the vector between one of those k-neighbors and the current data point get chosen. Multiplying this vector by a random number between 0 and 1, to create the current data point and following the creation of the new synthetic data point.

ROSE-sampling: ROSE (Random Over-Sampling Examples) is a bootstrap-based technique which helps the task of binary classification in the presence of rare classes.

2.2. Experiments

The Experiments had a focus on 3 evaluation areas:

- Different machine learning algorithms (SVM, RF, XGB, GLM, DT) and their result
- Analysing different metrics and optimizing the algorithm based on it
- Effect of different sampling algorithm for class imbalances

Evaluation metrics:

- **True Positives:** Correctly Classified Fraud Transactions
- **False Positives:** Incorrectly Classified Fraud Transactions
- **True Negative:** Correctly Classified Non-Fraud Transactions
- **False Negative:** Incorrectly Classified Non-Fraud Transactions
- **Precision:** How precise is the model in detecting fraud transactions
- **Recall:** Is the amount of fraud cases our model is able to detect
- **AUC** (Area Under Curve): Represents how well the results are ranked. If you pick a positive e.g by random, and negative by random, AUC = probability that positive is ranked > negative example. Measure of quality of discrimination.
- **F1- Score:** Weighted average of precision recall
- **ROC curves:** Calculates sensitivity/specificity ratio.

Precision/Recall Tradeoff: The more precise the model, the less cases it will detect.

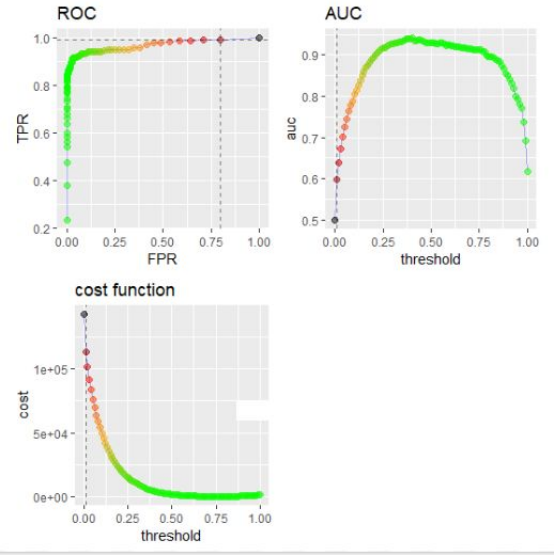
Accuracy score as a metric with imbalanced datasets this score will usually be high and misleading. Instead of the accuracy score many only communities recommend using the **f1-score, precision/recall score or confusion matrix.**

In this experiment, with keeping in mind the precision/recall tradeoff, I decided to create an own cost function. The calculation is based on

$$Cost = \sum FN * 10 + \sum FP$$

In this experiment the assumption is based on the cost of an undetected fraud case, that cost the bank 10 times more than blocking a legit transaction. This formula considered that after studies people that fell victim to a fraud case are 25% likely to switch the bank, while people that make a legit transaction that is being blocked might stay in the supermarked and

are not able to pay, which could lead them too to switch the bank too. Details about a proper cost function are not available, that is why this function ist just used as an experimental metric.



The above figure 1 shows an example of the Recall/precision tradeoff and the cost function, where the highest AUC didn't always represented the lowest cost. The details follow in this report.

2.2.1 Evaluation: SVM Kernel

Restricted training subset:

training method: CV 2
Sampling: Smote
Observations: 25.000
Features: All (30)

[train 20%, val 20%, test 60%]

Metric	precision	recall	accuracy	TNR	FNR	F1-score	own cost function FP*10+FN*1	own cost function FP*100+FN*1	AUC	Threshold	tp	tn	fp	fn
linear Weights	0,52	0,73	1,00	0,9989	0,27	0,61	803	6563	86,50	0,01	174	142003	163	64
Radial Weights	0,58	0,71	1,00	0,9991	0,29	0,64	813	7023	85,46	0,04	169	142043	123	69
Radial Weights(auc)	0,08	0,88	0,98	0,9820	0,12	0,14	2844	5454	93,01	0,01	209	139612	2554	29
Linear	0,83	0,78	1,00	0,9997	0,22	0,80	567	5337	88,85	0,01	185	142129	37	53
poly	0,86	0,79	1,00	0,9998	0,21	0,83	521	4931	89,70	0,1-0,15	189	142135	31	49
Radial(auc)	0,16	0,84	0,99	0,9923	0,16	0,26	1463	4793	91,84	0,01	201	141073	1093	37
Radial(cost)	0,82	0,77	1,00	0,9997	0,23	0,79	581	5441	88,64	0,04	184	142125	41	54

In the above figure 2, the result of the kernel evaluation can be found. The results were tuned on a validation set and evaluated on a test set. The table shows the results for the best AUC value on the specific Threshold. Note: The column "own cost function" represents the two cost functions for wrong classified cases.

Analysis result:

- The RBF (Weights) Kernel had the best performance on the AUC metric (column 9) as well as the best recall.
- While having the best AUC, the cost of 10*FP+FN was the highest (marked red)

- The Polynomial Kernel had the best score on F1, initial cost function and a good AUC value.
- The F1-score showed a strong correlation with the first cost function. The second cost function (with 100*FP+FN) showed no correlation with the F1-score.

Via grid search the following parameters where selected:

	LinearWeights	Linear	RadialWeights	Poly	Radial
cost	cost: 0.25	C = 1	C = 1	C = 0.25	C = 0.25
hyperparameter	gamma: 0.03333333		sigma = 0.05	degree = 2 scale = 0.01 offset = 1	sigma = 0.0296
number of support vectors	66	80	1252	103	772
Objective function Value		-569	-1.008	-122	-242
Training error		0,0008080	0,000211	0,000667	0,001861

Radial weight function had a high amount of support vector with 1252 compared to the linear SVM of 66.

Linear	Linear Weights	RBF	RBF Weights	Poly
23 sec	58 sec	53 sec	192 sec	59 sec

Further to note is the learning time were RBF (weights) used the longest with +3min and the linear kernel with 23 sec training time.

2.2.2 Evaluation: All prediction results AUC

The 2. experiment contained the evaluation of the different prediction algorithms. The entries represent the best outcome. Each entry can have its own threshold.

Method	Imbalanced	SMOTE result	cost based
GLM	0.91	0.87	0.87
Decision Tree	0.89	0.89	0.88
RF	0.92	0.94	0.91
SVM	0.85	0.93	0.91
XGB	0.91	0.94	0.92

Analysis result:

- On the unbalanced data set the best untuned algorithm was RF, while the linear SVM performed the worst.
- The best algorithm trained on the SMOTE balanced data set was XGB, RF and RBF SVM.

- The only surprising part was the GLM algorithm that had worse results with the balanced data set. This should be checked again or evaluated with different sampling methods (there might be very likely an error).

2.2.3 Sampling Methods on XGB

In this 3. experiment, the effects of different sampling methods were tested on the winning algorithm XGB. Each entry represents the best threshold for the case of best overall AUC and best AUC on cost function 10*FN+FP. The unbalanced AUC result for XGB was 0.913.

Method	AUC Score	AUC own cost function	(threshold)
unbalanced	0.913	0.913	(0.15)
up	0.93	0.910	(0.6)
down	0.90	0.87	(0.8)
smote	0.94	0.92	(0.95)
rose	0.900	0.900	(0.72)

Analysis result:

- SMOTE sampling gave the best result.
- DOWN sampling and ROSE decreased the score and total cost to the unbalanced data set. This result can be explained because the initial evaluation data set was restricted on 30k observations, where downsampling decreased the training observations even more.

3. Discussion

3.1 Conclusion

In the kaggle online community the SVM algorithm received not to much attention for this non-competition dataset.
An initial baseline performance for the different algorithm gave the impression that the SVM performs not as good as the other algorithm. The selection and optimization of the right svm kernel made the SVM a viable choice for prediction.
The experiments showed that the performance of the RBF SVM kernel had an AUC performance of 93.43%, while only the Random Forest with 94.2% and XGB with 94.4% had a higher performance in this experimental environment.
Using SMOTE sampling on the dataset helped with the imbalance of the labels. By increasing the recall of the true positive cases the expected trade-off of the precision was visible.

To find a proper optimization of the trade-off, the defined cost function gave a good insight of a possible business metric. By creating a scenario of a cost measurement, the results of the algorithm changed slightly and showed that the performance in a real world scenario can not be singly evaluated against a F1-score/AUC/ROC. In a real world application, I consider the most important step of defining the right cost metric, or else the recall/precision trade-off optimization can be hardly justified.

3.2 Critical Assessment

Restricted training subset for comparison purpose:
The experiment of comparing the different algorithms was restricted on a subset. The benefit of RF and XGB is that they have a less computational intensive training time compared to SVM on such a subset and can be trained on a bigger training set in an appropriate time, which might lead to better performing models.
The sampling methods are performed on the restricted subset as well, where the down sampling could have been performed on a bigger training set to not lose so much information.
Own thoughts:
Initially my goal was to evaluate and optimize the SVM and other ML algorithms against each other. In the experimental state I noticed that an optimization based on AUC or other metric lacks a justification on my side and that's why the experiments lead to a more exhaustive process than expected. The cost evaluation gave me a better justification and showed me that the optimization task needs further research on my side and further tests with different optimization/loss functions.

3.3 Future work

With the gained experience of my first classification problem, combined with different sampling strategies I see the future work in the following areas:

Experiment 1: Prediction algorithm
Evaluation of the algorithms on individual training sets for each algorithm and selection of important PCA features. E.g. bigger training set for RF and XGB and smaller feature size.

Experiment 2: Sampling
The sampling algorithm should be tested based on their beneficial characteristic:

- Using DOWN-sampling on 80% of the data set for a balanced training set
- SMOTE together with edited nearest-neighbours (ENN). SMOTE can generate noisy samples by interpolating new points between marginal outliers and inliers. This issue might be improved by cleaning after over-sampling the resulted space.

Experiment 3: Evaluation of other algorithms
Hidden markov models are said to perform well on anomaly detection scenarios. An application could be evaluated for this, as well as deep learning models.

Experiment 4: Model ensemble methods
The results and characteristic of different algorithms can be evaluated in a more detailed way, where the combination of different models might lead to better overall performance.
Experiment 5: Other kernel method
There are many more Kernel methods that could be evaluated in detail based on training time and model performance.

References:

[1]-https://www.altexsoft.com/whitepapers/fraud-detection-how-machine-learning-systems-help-reveal-scams-in-fintech-healthcare-and-ecommerce/ [16.12.2018]
[2]-https://www.afp.gov.au/what-we-do/crime-types/fraud/identity-crime [16.12.2018]
[3] Supervised Machine (SVM) Learning for Credit Card Fraud Detection http://ijettjournal.org/volume-8/number-3/IJETT-V8P225.pdf [16.12.2018]
[4] Calibrating Probability with Undersampling for Unbalanced Classification https://www3.nd.edu/~rjohns15/content/papers/ssci2015_calibrating.pdf
[5] A. Srivastava & A. Kundu, “Credit card fraud detection using hidden markov model,” IEEE Transactions on Dependable & Secure Computing, vol. 5, no. 1, 2008 https://www.computer.org/csdl/trans/tq/2008/01/ttq2008010037-abs.html [16.12.2018]
[6] Y. Sahin & E. Duman, “Detecting credit card fraud by decision trees & support vector machines,” Proceeding of the International MultiConfrence of Engineers & Computer Scientist, vol. I, 2011