MLP - Credit Card Fraud Detection

Edgar Emilio González y González
emilio.gonzalez@galileo.edu.gt

Abstract

Los medios de pago han evolucionado junto con la tecnología, cuando escuchamos el término "tarjeta de crédito" lo primero que se nos viene a la mente es un plástico que cabe en nuestra billetera, que para poder utilizarlo lo tenemos que pasar por una máquina POS(Point of sale, por sus siglas en inglés) y que teniéndolo guardado está completamente seguro, pero, como ha evolucionado los métodos de pago así ha evolucionado el fraude en tarjeta de crédito, la tarjeta de un cliente puede estar en el lugar más seguro del planeta y aun así puede ser utilizada de forma fraudulenta y la tarjeta ni siquiera tuvo que ser expuesta a alguien. Este problema se da con el término utilizado en detección de fraude como "Carding", las personas que se dedican a realizara fraude por este medio tienen software especializado para generar números de tarjeta de crédito, ingresando el BIN de la tarjeta* y el programa esta devuelve números de tarjeta generados con la misma formula matemática que se utiliza para la creación de tarjetas de crédito.

1 Dataset

Se utiliza un dataset público(facilidado en una competencia de Kaggle) debido a que los bancos locales son estrictos con el manejo de su información, se intentó utilizar data ofuscada pero no se pudo extraer de la compañía. Este dataset contiene variables que viajan al momento de una transacción, como por ejemplo la categoría del comercio(MCC Code) o el tipo de lectura de la transacción(Banda Magnetica, Chip o Contactless

1.1 Variables

Comunmente en la detección de fraude se utilizan las siguientes variables, aunque en este dataset se encuentran con otros nombres debido a la confidencialidad:

* Marca de Tarjeta.
* BIN de Tarjeta.
* Tipo de Lectura.
* Tipo de Entrada a POS.
* Hora de Transacción.
* MCC del Comercio.
* Ubicación del comercio.
* Capacidad de la terminal(POS)
* Lugar de transacción(Internet, llamada, o en comercio).
* País de la Transacción.
* Monto de Transacción

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

Se presenta un resumen de las variables dadas, ya modificadas acorde a las necesidades:

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22	V23	V24	V25	V26	V27	V28	Amount	Class
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.363787	...	-0.018307	0.277838	-0.110474	0.066928	0.128539	-0.189115	0.133558	-0.021053	149.62	0
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.255425	...	-0.225775	-0.638672	0.101288	-0.339846	0.167170	0.125895	-0.008983	0.014724	2.69	0
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.514654	...	0.247998	0.771679	0.909412	-0.689281	-0.327642	-0.139097	-0.055353	-0.059752	378.66	0
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.387024	...	-0.108300	0.005274	-0.190321	-1.175575	0.647376	-0.221929	0.062723	0.061458	123.50	0
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.817739	...	-0.009431	0.798278	-0.137458	0.141267	-0.206010	0.502292	0.219422	0.215153	69.99	0

2 Models

El modelo principal utilizado es de Keras con optimizador Adam, adicional a este e utilizaron 3 distintos modelos de sklearn para ver cual de ellos performa mejor con nuetro problema y poder comparar entre sklearn y keras cual tiene mejor resultados y mejor tiempo de procesamiento, siempre en busqueda que nuestro MLP sea más eficiente y con mejores predicciones.

2.1 Algoritmo ADAM:

es una extensión del descenso de gradiente y un sucesor natural de técnicas como AdaGrad y RMSProp que adapta automáticamente una tasa de aprendizaje para cada variable de entrada para la función objetivo y suaviza aún más el proceso de búsqueda mediante el uso de una media móvil exponencialmente decreciente del gradiente para realizar actualizaciones a las variables.

Adam está diseñado para acelerar el proceso de optimización, p. Ej. disminuir el número de evaluaciones de funciones requeridas para alcanzar los óptimos, o para mejorar la capacidad del algoritmo de optimización, p. resultar en un mejor resultado final.

Esto se logra calculando un tamaño de paso para cada parámetro de entrada que se optimiza. Es importante destacar que cada tamaño de paso se adapta automáticamente al rendimiento del proceso de búsqueda en función de los gradientes (derivadas parciales) encontrados para cada variable.

Algorithm: Generalized Adam

S0. Initialize $m_0 = 0$ and x_1

For $t = 1, \dots, T$, do

S1. $m_t = \beta_{1,t} m_{t-1} + \left(1 - \beta_{1,t}\right) g_t$

S2. $\hat{v}_t = h_t(g_1, g_2, \dots, g_t)$

S3. $x_{t+1} = x_t - \frac{\alpha_t m_t}{\sqrt{\hat{v}_t}}$

End

2.2 Algoritmo SGD:

es un método iterativo para optimizar una función objetivo con propiedades de suavidad adecuadas (por ejemplo, diferenciable o subdiferenciable). Puede considerarse como una aproximación estocástica de la optimización del descenso del gradiente, ya que reemplaza el gradiente real (calculado a partir del conjunto de datos completo) por una estimación del mismo (calculado a partir de un subconjunto de datos seleccionado al azar). Especialmente en problemas de optimización de alta dimensión, esto reduce la carga computacional, logrando iteraciones más rápidas en el comercio para una tasa de convergencia más baja. [1]

Si bien la idea básica detrás de la aproximación estocástica se remonta al algoritmo de Robbins-Monro de la década de 1950, el descenso de gradiente estocástico se ha convertido en un método de optimización importante en el aprendizaje automático.

Algorithm 1 Pseudo code of Stochastic Gradient Descent algorithm

1 *Enter initial values* (ϵ_k, θ)
2 ***while***
3 *Sampling* $\{x^{(1)}, \dots, x^{(m)}\}$ *from Dataset by using* $\{y^{(1)}, \dots, y^{(m)}\}$ *Labels*
4 $\hat{g} \leftarrow \frac{1}{m} \nabla_\theta \sum_i \mathrm{L}\left(f(x^{(i)}, \theta), y^{(i)}\right)$
5 $\theta \leftarrow \theta - \epsilon_k \hat{g}$
6 ***End while***

2.3 Algoritmo LBFGS:

Se trata de un método que hace un uso limitado de la memoria (usa mucha menos memoria que otros algoritmos para el mismo problema); L-BFGS viene de BFGS de memoria limitada. Permite obtener el mínimo de una función. Únicamente necesita la función y su gradiente, pero no la matriz Hessiana. L-BFGS, desarrollado por Jorge Nocedal es capaz de resolver funciones sin restricciones, mientras que la variante L-BFGS-B (Jorge Nocedal y Richard Byrd) puede resolver funciones con restricciones simples, los límites inferior y superior de esa variable) en sus parámetros. Si las restricciones son complejas otros métodos, como KNITRO, deben ser usados.

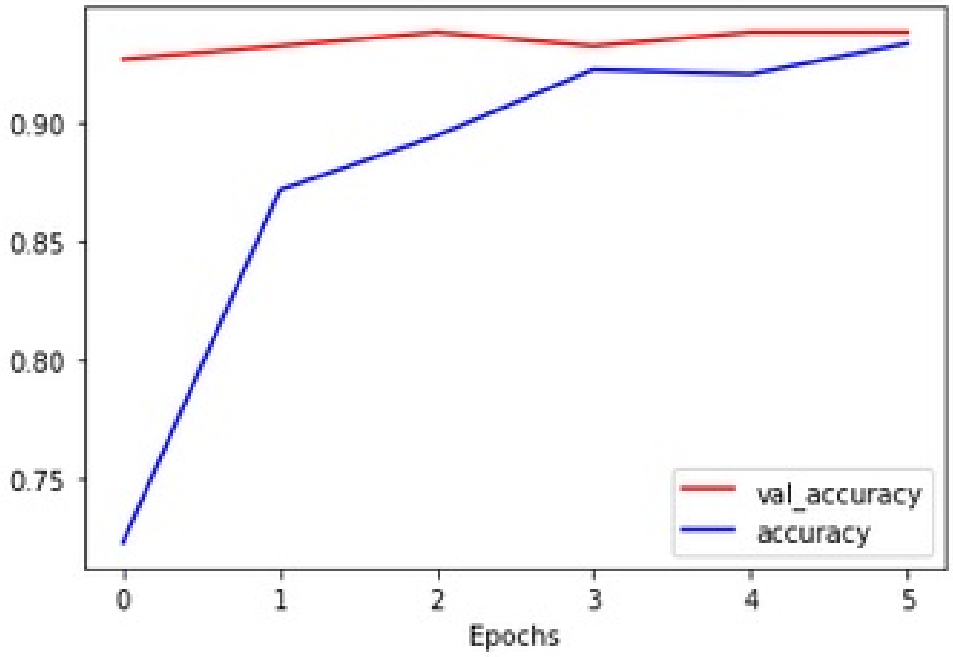
Para cada iteración el algoritmo busca una aproximación de la matriz Hessiana, concretamente de su inversa. Si la función tiene N variables, la matriz Hessiana tiene N 2 N2 elementos. Si N es grande, el tiempo necesario para calcular toda la matriz de forma exacta puede ser prohibitivo. Es por esto que se busca una aproximación

Algorithm 1 (L-BFGS two-loop recursion)

1. Given an input ∇f_k, set $q \leftarrow \nabla f_k$;
2. **for** $i = k-1, k-2, \dots, k-m$
3. $\quad \alpha_i \leftarrow \rho_i s_i^T q$ where $\rho_i = 1/(y_i^T s_i)$;
4. $\quad q \leftarrow q - \alpha_i y_i$;
5. **end (for)**
6. $r \leftarrow \gamma_k q$ where $\gamma_k = (s_{k-1}^T y_{k-1})/(y_{k-1}^T y_{k-1})$;
7. **for** $i = k-m, k-m+1, \dots, k-1$
8. $\quad \beta \leftarrow \rho_i y_i^T r$ where $\rho_i = 1/(y_i^T s_i)$;
9. $\quad r \leftarrow r + s_i(\alpha_i - \beta)$;
10. **end (for)**
11. **stop** with result $H_k \nabla f_k = r$.

3 Resultados

De las dos herramientas utilizadas tanto Keras como Sklearn MPL classifier, Keras tuvo por mucho mejores resultados que el segundo mencionado, tanto en el entrenamiento como en la validación, es importante tener en cuenta que los números de falsos positivos son muy bajos, lo que indica que nuestro modelo podría ser implementado en un sistemta transaccional sin afectar las necesidades del negocil:



4 Conclusiones

Para poder implementar un sistema que detenga transacciones "en vivo" en un banco se requiere por lo menos que 1 de cada 3 transacciones denegadas sean fraude(requerimientos de marcas Visa/Mastercard) por lo que nuestro MLP tiene muy buenos resultados y se podría implementar en un sistema bancario.

Adicional se puede concluir que la modificación de las variables para normalizarlas fue de gran ayuda en este dataset, en los bancos regularmente muchas variables son textos o fechas, lo que nos complica el análisis y nos puede crear mucho underfitting o tambien overfitting en nuestro análisis.

5 Referencias

VARMEDJA, Dejan, et al. Credit card fraud detection-machine learning methods. En 2019 18th International Symposium INFOTEH-JAHORINA (INFOTEH). IEEE, 2019. p. 1-5.

PILLAI, Thulasyammal Ramiah, et al. Credit card fraud detection using deep learning technique. En 2018 Fourth International Conference on Advances in Computing, Communication Automation (ICACCA). IEEE, 2018. p. 1-6.

ARUN, Gurumurthy Krishnamurthy; VENKATACHALAPATHY, Kaliyappan. Intelligent feature selection with social spider optimization based artificial neural network model for credit card fraud detection. IIOABJ, 2020, vol. 11, no 2, p. 85-91.