Automatas Probabilisticos y Cadenas de Markov

Daniel Rojas — Felix Solano — Raul Mosquera

9 de noviembre de 2019

Índice

1. Introducción

En la década de 1960, se inicio el estudio de autómatas probabilísticos. Desde ese entonces, la mayoría de literatura sobre autómatas contiene secciones enteramente dedicadas a estos. Asimismo, son importantes para diversas áreas del conocimiento como Robótica, Inteligencias Artificial, Finanzas, etc. El presente proyecto se enfocara en autómatas probabilísticos orientados al industria financiera, específicamente en el área de evaluación del riesgo de un crédito. A continuación, se dará una definición mucho mas precisa de un autómata probabilístico.

Definiremos un autómata finito probabilístico (PFA) como una generalización del autómata finito determinista (AFD). En un AFD, un estado y una transición σ determinan el estado del autómata. La idea de un PFA se base en un comportamiento estocástico. Es decir, que para un estado y una entrada σ, el autómata puede moverse a cualquier estado s_i, siendo la probabilidad de que se llegue a s_i una función p_i,$(s_i,\ \sigma)$.

Un tipo de Autómata Finito Probabilístico son las cadenas de Markov. Las cadenas de Markov tienen la propiedad de que $s_{(n+1)}$ solo depende del estado anterior del sistema $s_{(n)}$. Los valores de las variables $(s_1,\ s_2,\ s_3,....s_n)$ no se pueden predecir exactamente, pero se puede se pueden hallar probabilidades para los distintos valores posibles de estas variables.

Para crear soluciones, en el área de gestión del riesgo crediticio de la industria financiera, se usan Modelos Ocultos de Markov (MOM), que son autómatas abstractos de estados finitos que permiten modelar procesos estocástico, donde la ocurrencia de los estados esta asociada con una distribución de probabilidad y donde las transiciones entre los estados están dadas por un conjunto de probabilidades llamadas probabilidades de transición de estados.

Para representar estas probabilidades, se crea una matriz de transición, donde para cada $i = 1, 2, ..., m$ existe un valor p_{ij} que representa su probabilidad de transición tal que: $T = [p_i j] =$

$$\begin{bmatrix} p_{11} & p_{12} & p_{13} & \cdots & p_{1m} \\ p_{21} & p_{22} & p_{23} & \cdots & p_{2m} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ p_{m1} & p_{m2} & p_{m3} & \cdots & p_{mm} \end{bmatrix}$$

2. Desarrollo

2.1. Definición del Problema

Las cadenas de Markov son una herramienta matemática importante en procesos estocásticos debido a la ya mencionada propiedad de Markov. La propiedad de Markov señala que algunas predicciones de estos procesos pueden ser simplificadas si se ve un estado futuro como independiente de los estados anteriores al presente.

2.1.1. Medición de Riesgo de un Crédito

El riesgo de un crédito esta definido como la posible perdida que sufre un acreedor debido al incumplimiento de un pago por parte de un deudor, y esta perdida es tomada en cuenta al momento en que una entidad financiera realiza un préstamo ya sea a una persona jurídica o a una persona natural.

Teniendo una matriz de transición podemos elevarla varias a la n. Con el objetivo de que los resultados de cada uno de los elementos converjan, de acuerdo a esta convergencia se puede categorizar los candidatos a un préstamo dividiéndolos por escalas y estimando su probabilidad de permanencia en dicha escala a través del tiempo. De esta manera, el acreedor puede minimizar su riesgo de perdida al momento de realizar un préstamo. Para determinar dicha clasificación se utiliza el siguiente proceso:

- Hallamos el autovalor de la matriz $T - I$ igualando $(T - I_m) = 0$ donde I_m es la matriz identidad de orden mxm. Y X es una columna de la matriz final a la cual converge la matriz de transición. Además es el autovalor de la matriz $T - I$.

- A través del método de eliminación Gaussiana hallamos la matriz triangular superior de la matriz $(T - I)$. Debido a que el autovector X a la misma vez es una columna de la matriz final la suma de sus elementos es 1. $(X_1 + X_2 + ... + X_m) = 1$

- Dado que la suma de las filas de $(T - I)$ es igual a 0, esta es linealmente dependiente y tiene al menos 1 solución distinta del vector cero. Al ser linealmente independiente una fila puede ser formada por las demás.

- En la nueva matriz, luego de aplicar la eliminación Gaussiana, Por esta razón, al ultimo elemento X_n del vector X le damos cualquier valor numérico y hallamos los valores restantes para un vector X_i' que cumpla con la ecuación.

- Ya que la suma de elementos del vector X tiene que se igual a 1, definimos los elementos del vector de la siguiente forma: $X_i = \frac{X_i'}{\Sigma X_i'}$

  Este vector sera igual a todas las columnas de la matriz a la que converge T^n cuando n tiende a ∞.

3. Estado del Arte

El proceso estocástico que genera la dinámica de las migraciones crediticias puede ser representado mediante cadenas de Markov, pero este análisis no permite analizar adecuadamente las fuentes o factores que generan las fluctuaciones en las probabilidades de incumplimiento. En otras palabras, se podría mejorar el análisis para fines de supervisión financiera.

En 2016, Kipkoech utilizo una cadena multivariable de Markov para modelar el riesgo de crédito en prestamos a consumidores, generando la matriz de transición basándose en regresión logística acumulativa. En nuestro caso, utilizaremos una cadena de Markov regular para representar el modelo y observaciones generadas aleatoriamente para la construcción de la matriz de transición.

El sistema de puntaje de crédito que usaremos es el puntaje FICO, cuyos puntajes varían desde 300 hasta 850 puntos. **Mientras mayor sea el puntaje mayores serán las probabilidades de devolución del crédito sin problemas.** El puntaje FICO se divide en 6 secciones:

- 300-599: Probabilidades muy bajas.
- 600-649: Probabilidades bajas.
- 650-699: Probabilidades intermedias.
- 700-749: Probabilidades buenas.
- 749-799: Probabilidades muy buenas.
- 800-850: Probabilidades excelentes.

A partir de esto, generaremos la data que se usara creado perfiles de personas al azar, con un puntaje de crédito ficticio que ira variando cada seis meses por un periodo de 10 anos. Tomaremos consideraciones, como que el crecimiento o decrecimiento del puntaje no se aleje mucho de las posibilidades reales y que los porcentajes de población en cada rango se asemeje a los de los casos reales. Luego, observando la data y calculando las probabilidades de transición de cada rango se construirá la matriz de transición.

Para hallar la matriz de transición a largo plazo se puede tomar la matriz ya construida y a través del procedimiento antes mencionado calcular la potencia de la matriz un numero n de veces hasta obtener la que nos indique las probabilidades de cambio de estado a largo plazo.

4. Código

A continuacion se presentara detalladamente el codigo que implementa nuestra solucion. Primero veremos el **codigo respectivo a la obtencion de la data.**

```
#include <bits/stdc++.h>
using namespace std;

#define FICO 6

class muestra{
public:
    int size; int min; int max;
    std::vector< int > data;
    std::vector< std::string > rankings;
    double frequencies[FICO];
    double percentages[FICO];

public:
    muestra(): min{300}, max{850}, size{0}, frequencies{0}, percentages{0} {};
    explicit muestra(int size) : size{size}, min{300}, max{850}, frequencies{0}, percentages{0}{};

    /* Se genera datos de puntaje aleatoriamente en un
     * rango entre 300 y 850 y se anade al vector data*/
    void generateData(){
        for(int i = 0; i < size; i++){
            int n = rand() %551 + 300;
            data.push_back(n);
        }
    }

    /* Dependiendo del rango donde se encuentre
     * el rango se asigna una categoria*/
    void calculateRanks(){
        string s;
        for(int i : data){
            if(i >= 300 && i < 600){
                s = "Muy bajas";
                rankings.push_back(s);
            }
            else if(i >= 600 && i < 650){
                s = "Bajas";
                rankings.push_back(s);
            }
            else if(i >= 650 && i < 700){
                s = "Intermedias";
                rankings.push_back(s);
            }
            else if(i >= 700 && i < 750){
                s = "Buenas";
                rankings.push_back(s);
            }
```

```
            else if(i >= 750 && i < 800){
                s = "Muy buenas";
                rankings.push_back(s);
            }
            else if(i >= 800 && i < 850){
                s = "Excelentes";
                rankings.push_back(s);
            }
        }
    }

    /* Calcula el porcentaje de acuerdo a la categoria */
    void calculatePercentage(){
        for(const auto & ranking : rankings){
            if(ranking == "Muy bajas"){ ++frequencies[0]; }
            else if(ranking == "Bajas"){ ++frequencies[1]; }
            else if(ranking == "Intermedias"){ ++frequencies[2]; }
            else if(ranking == "Buenas"){ ++frequencies[3]; }
            else if(ranking == "Muy buenas"){ ++frequencies[4]; }
            else if(ranking == "Excelentes"){ ++frequencies[5]; }
        }
        for(int i = 0; i < FICO; i++){
            percentages[i] = frequencies[i] / size * 100;
        }
    }

    /* Se genera una nueva muestra para calcular los
     * cambios la cual contiene los datos de la muestra
     * actual mas un numero al azar entre -100 y 100. Nos
     * seguramos de que esta suma nunca resulte en un
     * puntaje menor a 300 o mayor a 850*/
    muestra changes(){
        muestra newMuestra(size);
        for(int i : data){
            int change = (rand()%201)- 100;
            if(i + change < 300 || i + change > 850) {
                newMuestra.data.push_back(i);
            }
            else{
                newMuestra.data.push_back(i + change);
            }
        }
        newMuestra.calculateRanks();
        newMuestra.calculatePercentage();
        return newMuestra;
    }

    /* Muestra los porcentajes */
    void showPercentages(){
        for(int i = 0; i < FICO; i++){
            cout << "Categoria " << i << ": " << percentages[i] <<
"%" << endl;
        }
    }

    /* Muestra la data de cada persona */
    void showData(){
```

```
        for(int i = 0; i < static_cast<int>(data.size()); i++){
            cout << "Persona: " << i +1 << ": " <<  data.at(i) << endl;
        }
    }

    /* Muestra el ranking de cada persona (excelente, bueno, malo, etc) */
    void showRanks(){
        cout << rankings.size() << endl;
        for(int i = 0; i < static_cast<int>(rankings.size()); i++){
            cout << "Persona: " << i + 1  << ": " <<  rankings.at(i) << endl;
        }
    }
};
```

A continuacion se presentara la parte del codigo que muestra la obtencion de la matriz de transicion y como a traves de esta se obtiene la matriz final.

```
/* Se creo la clase repositorio que tiene como atributos una matriz de 20 muestras
 * una para cada semestra de los 10 anos. Y dos matrices de 6 x 6 que representan
 * la matriz de transicion y la matriz final*/
#include <bits/stdc++.h>
#include "muestra.h"

class repositorio{
public:
    muestra muestras[20];
    double matriz[6][6];
    double matriz_final[6][6];
    double matriz_autovalor[6][6];

    explicit repositorio(muestra muestras[20]): matriz{0}, matriz_final{0}, matriz_autovalor{0} {
        for(int i = 0; i < 20; ++i){ this->muestras[i] = muestras[i]; }
    }

    /* Se obtiene la matriz de transicion sumando el numero de trasiciones desde todos
     * los estados hasta todos los estados. Luego se divide la cantidad de transiciones
     * de un estado a otro entre la cantidad de transiciones de ese estado a cualquiera.
     * Asi se obtiene la probabilidad de cada transicion. */
    void matriz_transicion() {
        for (int i = 0; i < muestras->size -1 ; i++) {
            for (int j = 0; j < static_cast<int>(muestras[i].rankings.size())-1; j++) {
                if (muestras[i].rankings.at(j) == "Muy bajas") {
                    if (muestras[i + 1].rankings.at(j) == "Muy bajas") {
                        matriz[0][0]++;
                    } else if (muestras[i + 1].rankings.at(j) == "Bajas") {
```

```
                        matriz[1][0]++;
                    } else if (muestras[i + 1].rankings.at(j) == "Intermedias") {
                        matriz[2][0]++;
                    } else if (muestras[i + 1].rankings.at(j) == "Buenas") {
                        matriz[3][0]++;
                    } else if (muestras[i + 1].rankings.at(j) == "Muy buenas") {
                        matriz[4][0]++;
                    } else if (muestras[i + 1].rankings.at(j) == "Excelentes") {
                        matriz[5][0]++;
                    }
                } else if (muestras[i].rankings.at(j) == "Bajas") {
                    if (muestras[i + 1].rankings.at(j) == "Muy bajas") {
                        matriz[0][1]++;
                    } else if (muestras[i + 1].rankings.at(j) == "Bajas") {
                        matriz[1][1]++;
                    } else if (muestras[i + 1].rankings.at(j) == "Intermedias") {
                        matriz[2][1]++;
                    } else if (muestras[i + 1].rankings.at(j) == "Buenas") {
                        matriz[3][1]++;
                    } else if (muestras[i + 1].rankings.at(j) == "Muy buenas") {
                        matriz[4][1]++;
                    } else if (muestras[i + 1].rankings.at(j) == "Excelentes") {
                        matriz[5][1]++;
                    }
                } else if (muestras[i].rankings.at(j) == "Intermedias") {
                    if (muestras[i + 1].rankings.at(j) == "Muy bajas") {
                        matriz[0][2]++;
                    } else if (muestras[i + 1].rankings.at(j) == "Bajas") {
                        matriz[1][2]++;
                    } else if (muestras[i + 1].rankings.at(j) == "Intermedias") {
                        matriz[2][2]++;
                    } else if (muestras[i + 1].rankings.at(j) == "Buenas") {
                        matriz[3][2]++;
                    } else if (muestras[i + 1].rankings.at(j) == "Muy buenas") {
                        matriz[4][2]++;
                    } else if (muestras[i + 1].rankings.at(j) == "Excelentes") {
                        matriz[5][2]++;
                    }
                } else if (muestras[i].rankings.at(j) == "Buenas") {
```

```
                    if (muestras[i + 1].rankings.at(j) == "Muy bajas") {
                        matriz[0][3]++;
                    } else if (muestras[i + 1].rankings.at(j) == "Bajas") {
                        matriz[1][3]++;
                    } else if (muestras[i + 1].rankings.at(j) == "Intermedias") {
                        matriz[2][3]++;
                    } else if (muestras[i + 1].rankings.at(j) == "Buenas") {
                        matriz[3][3]++;
                    } else if (muestras[i + 1].rankings.at(j) == "Muy buenas") {
                        matriz[4][3]++;
                    } else if (muestras[i + 1].rankings.at(j) == "Excelentes") {
                        matriz[5][3]++;
                    }
                } else if (muestras[i].rankings.at(j) == "Muy buenas") {
                    if (muestras[i + 1].rankings.at(j) == "Muy bajas") {
                        matriz[0][4]++;
                    } else if (muestras[i + 1].rankings.at(j) == "Bajas") {
                        matriz[1][4]++;
                    } else if (muestras[i + 1].rankings.at(j) == "Intermedias") {
                        matriz[2][4]++;
                    } else if (muestras[i + 1].rankings.at(j) == "Buenas") {
                        matriz[3][4]++;
                    } else if (muestras[i + 1].rankings.at(j) == "Muy buenas") {
                        matriz[4][4]++;
                    } else if (muestras[i + 1].rankings.at(j) == "Excelentes") {
                        matriz[5][4]++;
                    }
                } else if (muestras[i].rankings.at(j) == "Excelentes") {
                    if (muestras[i + 1].rankings.at(j) == "Muy bajas") {
                        matriz[0][5]++;
                    } else if (muestras[i + 1].rankings.at(j) == "Bajas") {
                        matriz[1][5]++;
                    } else if (muestras[i + 1].rankings.at(j) == "Intermedias") {
                        matriz[2][5]++;
                    } else if (muestras[i + 1].rankings.at(j) == "Buenas") {
                        matriz[3][5]++;
                    } else if (muestras[i + 1].rankings.at(j) == "Muy buenas") {
                        matriz[4][5]++;
```

```
                    } else if (muestras[i + 1].rankings.at(j) == "Excelentes") {
                        matriz[5][5]++;
                    }
                }
            }
        }
        //showMatrix();
        cout << "--------------------------------" << endl;
        double suma[6];
        for(int j = 0; j < 6; ++j){
            suma[j] = 0;
            for(auto & i : matriz){ suma[j] += i[j]; }
        }
        for(int j = 0; j < 6; ++j){
            for(auto & i : matriz){ i[j] /= suma[j]; }
        }
    }

    /* Matriz de autovalores */
    void MatrizAutovalor(){
        for (int i = 0; i < 6; i++)
            for (int j = 0; j < 6; j++)
                matriz_autovalor[i][j]=matriz[i][j];
        for (int i = 0; i < 6; i++) {
            matriz_autovalor[i][i]=matriz_autovalor[i][i]-1;
        }

        for(auto & i : matriz_autovalor){
            for(double j : i) { cout << setw(10) << j << " "; }
            cout << endl;
        }
        cout<<"\n";
    }

    /* Mostrar matriz */
    void showMatrix(){
        for(auto & i : matriz){
            for(double j : i){ cout << setw(10) << j << " "; }
            cout << endl;
        }
        cout<<"\n";
    }
};
```

Funcion utilizada para hallar la matriz final:

```
/* Funcion utilizada para hallar la matriz final*/
void GaussElimination(double a[6][6]) {
    int n,i,j,k; cout.precision(7); cout.setf(ios::fixed);
    n = 6; double x[n];

    for (i=0;i<n-1;i++)
        for (k=i+1;k<n;k++) {
            double t=a[k][i]/a[i][i];
            for (j=0;j<n;j++)
                a[k][j]=a[k][j]-t*a[i][j];
        }
```

```

    cout<<"\n\nLa matriz despues de la eliminacion gaussiana es la siguiente:\n\n\n";
    for (i=0;i<n;i++) {
        for (j=0;j<=n;j++)
            cout<<a[i][j]<<setw(16);
        cout<<"\n";
    }
    x[5]=3;
    for (i=n-2;i>=0;i--) {
        x[i]=a[i][n];
        for (j=i+1;j<n;j++)
            if (j!=i)
        x[i]=x[i]-a[i][j]*x[j];
        x[i]=x[i]/a[i][i];
    }
    double suma=0;
    for (i=0; i<n; i++) { suma+=x[i]; }
    for (i=0; i<n; i++) { x[i]=x[i]/suma; }

    cout<<"\nLos elementos del autovector son los siguientes \n";
    for (i=0;i<n;i++)
        cout<<x[i]<<endl;
}
```

Finalmente se ejecuta todo en el siguiente programa:

```
int main() {
    srand(time(nullptr));
    double m[6][6];
    int num_personas = 10;
    const int num_muestras = 20;
    muestra muestra1(num_personas);
    muestra muestras[num_muestras];

    muestra1.generateData();
    muestra1.calculateRanks();
    muestra1.calculatePercentage();
    cout << "\n";
    muestras[0] = muestra1;
    for(int i = 1; i < num_muestras; i++){
        muestras[i] = muestras[i-1].changes();
    }
    repositorio repositorio1(muestras);
    repositorio1.matriz_transicion();

    cout<<"\n\nLa matriz transicion es la siguiente:\n";
    repositorio1.showMatrix();
    cout<<"\n\nLa matriz diferencia entre la matriz transicion y la matriz identidad es la siguiente:\n";
    repositorio1.MatrizAutovalor();
    GaussElimination(repositorio1.matriz_autovalor);

    return 0;
}
```

5. Resultados y Conclusiones

A través del código mostrado, se pudo obtener las probabilidades de transición a cualquier estado en el largo plazo. El resultado obtenido fue el siguiente:

- Categoría 1: 0.5756038
- Categoría 2: 0.1197741
- Categoría 3: 0.0724008
- Categoría 4: 0.0643562
- Categoría 5: 0.0855645
- Categoría 6: 0.0822736

En conclusión, podemos observar que la mayoría de transiciones llegan a las categorías 1 y 2, lo cual muestra que los individuos están perdiendo puntaje FICO. A partir de esto se pues decir que una buena medida por parte de entidades seria aumentar intereses o ser mas rigurosos con la evaluación de un préstamo, ya que en este caso se puede predecir que sus usuarios baja mas de categoría de lo que suben.

6. Referencias

Kipkoech K. *A multivariate Markov Chain model for credit risk measurement and management.* University of Noirobi, School of Mathematics. November 2016.

Stoelinga M. *An introduction to probabilistic automata* Department of Computer Engineering University of California at Santa Cruz, CA 95064, USA.

D. Scon, L. Wallin and P. Wikstrom. *An introduction to Markov chains and their applications within finance*

M.O Rabin *Probabilist Automata* Hebrew University, Jerusalen and Massachuset Institute of Technology, Cambridge, Massachusets, 1963.