Stat 471 Mathematical Model

Bao Nguyen, Adhiguna Pande, Khac Nguyen Nguyen, John He
Markov Mavericks

October 2023

Contents

1 Introduction **3**
1.1 Problem background 3
1.2 Restatement of the problem 3

2 Assumptions and Model Overview **3**
2.1 Assumptions 3
2.2 Model Framework 3
2.2.1 Model I: Voter Model 3
2.2.2 Model II: 4

3 Model Preparation **4**
3.1 Parameters 4
3.2 Data 4

4 Model I **5**
4.1 Normalized Distance 5
4.2 Transition Rates 5

5 Model II **6**
5.1 Linear Surface: Population density and Crime 6
5.2 Employment Rate 7
5.3 Final Result 8

6 Conclusion **8**

7 Bibliography **9**

8 Code **9**
8.1 Calculate parameters 9
8.2 Gradient Descent 10
8.3 Generating Data 12

1 Introduction

1.1 Problem background

In this project we will attempt to create a simulation for rent by changing different factors affecting rent prices.

1.2 Restatement of the problem

Given economic, social and environmental factors parameters, how can we simulate changing rent prices using Markov models and mathematical techniques?

2 Assumptions and Model Overview

2.1 Assumptions

- House types are uniform to ensure consistent analysis for each "particle" and limit disparities rising from having different house types
- Inflation is not accounted for

2.2 Model Framework

This paper will introduce two models which will be simulated simultaneously; one being a Markov model and the other based on the voter model.

2.2.1 Model I: Voter Model

City house prices interact with each other in a voter model where each average price change will have a probability of affecting average prices in other cities depending on physical proximity and size between cities.

Considering cities to be "particles," they each have the Markov property with 3 price states: going up, going down and staying the same with cutoffs.

Particles interact with each other, and during an interaction one particle adopts the opinion of others (i.e., one conforms to others). The model reaches an absorption state (where the entire population ends up holding the same opinion) at a rate proportional to the population size.

You can see this basic result with the default settings of the model shown here, but this implementation also includes some complicating factors - some of them idiosyncratic.

Each particle is initialised with a price state based on data such population density and CSI and employment rate, which will be introduced in Model II.

2.2.2 Model II:

Given data from Model I, we want to predict rent prices for the following year.

3 Model Preparation

3.1 Parameters

Parameter	**Definition**	**Units**
$R_t(i)$	Average rent of i^{th} site at time t	CAD (\$)
$X_t(j)$	State of site j at time t	$-1, 0, 1$
$\lambda_{i \to j}$	Rate at which site changes from state i to state j	$[0, 1]$
$\Delta_{i,j}$	Distance between site i and site j	km
$d(\Delta_{i,j})$	Weight/influence of site j on rate of site i	$[0, 1]$
P	Population density of i^{th} site	people/km^2
Q	Crime severity index at i^{th} site	$\mathbb{R}^+$
Z	Employment rate change at i^{th} site	%

Table 1: Table of Parameters for Model I

3.2 Data

GitHub for Data set
Statistics Canada. Table 35-10-0063-01 Crime severity index and weighted clearance rates, police services in British Columbia
Statistics Canada. Table 35-10-0190-01 Crime severity index and weighted clearance rates, police services in Alberta
Statistics Canada. Table 35-10-0061-01 Crime severity index and weighted clearance rates, police services in Saskatchewan
Statistics Canada. Table 35-10-0188-01 Crime severity index and weighted clearance rates, police services in Ontario
Statistics Canada. Table 35-10-0187-01 Crime severity index and weighted clearance rates, police services in Quebec
Statistics Canada. Table 14-10-0287-03 Labour force characteristics by province, monthly, seasonally adjusted
Statistics Canada. Table 17-10-0135-01 Population estimates, July 1, by census metropolitan area and census agglomeration, 2016 boundaries
Wikipedia-List of the largest municipalities in Canada by population
Statistics Canada. Table 34-10-0133-01 Canada Mortgage and Housing Corporation, average rents for areas with a population of 10,000 and over

4 Model I

4.1 Normalized Distance

For any city i, we want to find the transition rate for price changes. Firstly, we let d be the normalized distance between the current city i and a neighboring city j. Clearly, the further city i is from its neighbor j, the lower the probability city j's price changes will affect city i. Beyond a certain distance away, the price change probability will be negligible so we will only consider a city with neighbors within 100km.

$$d(\Delta_{i,j}) = \frac{1}{0.002\Delta_{i,j} - 1} + 2$$

Notice that when $x = 0$, $d(\Delta_{i,j}) = 1$. On the other hand, if $x = 100$, $d(\Delta_{i,j}) = 0.75$ and when $\Delta_{i,j} > 222.222$, $d(\Delta_{i,j}) < 0.2$, which is what we desire since the distance effect becomes negligible quickly.

Note that we can modify constants in this function to improve accuracy as needed.

4.2 Transition Rates

With d_i from above, we can compute the transition rates of city i based on its neighboring cities. Here we have three cases: increasing to no change, decreasing to no change and no change to

$$\lambda_{1\to 0}(j) = \frac{1}{\sum_j d_j} \sum_j \min(1 - X_t(j), 1) d_j$$

Here, we want the rate to

$$\min(1 - X_t(j), 1) = \begin{cases} 1, & \text{if } X_t(j) \neq 1 \\ 0, & \text{otherwise} \end{cases}$$

Similarly:

$$\lambda_{-1\to 0}(j) = \frac{1}{\sum_j d_j} \sum_j \min(1 + X_t(j), 1) d_j$$

$$\min(1 + X_t(j), 1) = \begin{cases} 1, & \text{if } X_t(j) \neq -1 \\ 0, & \text{otherwise} \end{cases}$$

For the last case going from state 0 to any other state:

$$\lambda_{0\to k}(j) = \frac{1}{\sum_j d_j} \sum_j d_j 1_{X_t(j)=k}$$

Adding up all three cases for transition rates equal 1 because the indicator function $1_{X_t(j)=0} + 1_{X_t(j)=1} + 1_{X_t(j)=-1} = 1$.

5 Model II

While plotting rental prices against various factors that potentially affect rent prices such as population density, crime severity and employment rate. Based on the plots, we found linear relation between rental price and factors such as population density and crime severity index but non-linear relation between rental prices and employment rate.

5.1 Linear Surface: Population density and Crime

In this section, we introduce rent price change model over one year with parameter - CSI(Crime Index Rate) and population density. We are using MLE (maximum likehood estimate) to find the coefficients for different parameter.

We start with the Gaussian-noise model:
1. The distribution P and Q are arbitrary (The model can work if they are non-random)
2. For $P = p$ and $Q = q$, let $R = ap + bq + c + \epsilon$, where ϵ is the noise
3. $\epsilon \sim N(0, \sigma^2)$, is independent of P and Q.

This model gives the conditional pdf of R given condition $P = p, Q = q$:

$$f(r|P = p, Q = q; a, b, c, \sigma^2)$$

Then, given the data set $(p_1, q_1), (p_2, q_2), ..., (p_n, q_n)$, the likelihood function is:

$$L(a_0, b_0, c_0, s^2) = \Pi_{i=1}^n f(r_i|p_i, q_i; a, b, c, \sigma^2) = \Pi_{i=1}^n \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(r_i-(ap_i+bq_i+c))^2}{2\sigma^2}}$$

Since the parameter are unknown to us, for any guess (a_0, b_0, c_0, s^2), we have the likelihood function:

$$\Pi_{i=1}^n f(r_i|p_i, q_i; a_0, b_0, c_0, s^2) = \Pi_{i=1}^n \frac{1}{\sqrt{2\pi s^2}} e^{-\frac{(r_i-(a_0p_i+b_0q_i+c_0))^2}{2s^2}}$$

Then we can find the log-likelihood function as:

$$l(a_0, b_0, c_0, s^2) = \ln(\Pi_{i=1}^n f(r_i|p_i, q_i; a_0, b_0, c_0, s^2))$$

$$= \sum_{i=1}^n \ln(f(r_i|p_i, q_i; a_0, b_0, c_0, s^2))$$

$$= -\frac{n}{2}\ln(2\pi) - n\ln(s) - \frac{1}{2s^2}\sum_{i=1}^n (r_i - (ap_i + bq_i + c))^2$$

In the method of maximum likelihood, we will find the parameter values which maximize the likelihood or more convenient, the log-likelihood:

$$\frac{\partial}{\partial a}\sum_i (r_i - f(p_i, p_i))^2 = 0 \rightarrow \sum_i (r_i - f(p_i, q_i))p_i = 0$$

$$\frac{\partial}{\partial b}\sum_i (r_i - f(p_i, q_i))^2 = 0 \rightarrow \sum_i (r_i - f(p_i, q_i))q_i = 0$$

$$\frac{\partial}{\partial c}\sum_i (r_i - f(p_i, q_i))^2 = 0 \rightarrow \sum_i r_i - f(p_i, q_i) = 0$$

where $f(p_i, q_i) = ap_i + bq_i + c$, then we have the matrix multiplication as follows:

$$\begin{bmatrix} \sum p_i^2 & \sum p_i q_i & \sum p_i \\ \sum p_i q_i & \sum q_i^2 & \sum q_i \\ \sum p_i & \sum q_i & n \end{bmatrix} \begin{pmatrix} a \\ b \\ c \end{pmatrix} = \begin{pmatrix} \sum r_i p_i \\ \sum r_i q_i \\ \sum r_i \end{pmatrix}$$

$$\text{Let } A = \begin{bmatrix} \sum p_i^2 & \sum p_i q_i & \sum p_i \\ \sum p_i q_i & \sum q_i^2 & \sum q_i \\ \sum p_i & \sum q_i & n \end{bmatrix} \text{ and } B = \begin{pmatrix} \sum r_i p_i \\ \sum r_i q_i \\ \sum r_i \end{pmatrix}$$

Then, we can estimate the parameters as:

$$\begin{pmatrix} a \\ b \\ c \end{pmatrix} = A^{-1}B$$

We can then implement the above calculation using the calculate parameters

5.2 Employment Rate

Based on the generated graphs, we could not find a clear correlation between employment rate and rent prices. Hence, for the sake of this being an assignment, we modified the sigmoid function by adding parameters a and b to represent how employment rates affect rent price changes:

$$g(r|Z = z; a, b, \sigma^2) = \frac{a}{1 + e^{-bz}} - \frac{a}{2} + \epsilon$$

Then, applying the Gaussian model on the above function where $\epsilon \sim N(0, \sigma^2)$ and Z is employment rate, we have the likelihood:(for abbreviation we write $g(r|Z = z; a, b, \sigma^2) = g(r)$)

$$\Pi_{i=1}^n \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y_i - g(x_i))^2}{2\sigma^2}}$$

then we have the log-likelihood:

$$L(a, b, c, s^2) = \ln \Pi_{i=1}^n \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(r_i - g(z_i))^2}{2\sigma^2}} = \sum_{i=1}^n \ln(\frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(r_i - g(z_i))^2}{2\sigma^2}})$$

$$= n \ln(\frac{1}{\sqrt{2\pi\sigma^2}}) - \frac{1}{2\sigma^2}\sum_{i=1}^n (r_i - g(z_i))^2$$

In the method of maximum likelihood,(least square), we obtain:

$$\frac{\partial L}{\partial a} = 0 \to \sum_{i=1}^{n} (r_i - g(z_i))(\frac{1}{1+e^{-bz_i}} - \frac{1}{2}) = 0$$

$$\frac{\partial L}{\partial b} = 0 \to \sum_{i=1}^{n} (r_i - g(z_i))\frac{-z_i e^{-bz_i}}{(1+e^{-bz_i})^2} = 0$$

Since these derivatives were too complicated to determine analytic solutions, we used gradient descent instead to estimate a and b.

5.3 Final Result

Using functions g and f from above, we can create a new function

$$h(x_1, x_2, x_3) = ag(x_1) + bf(x_2, x_3) + \epsilon$$

where a and b are parameters $\epsilon \sim N(0, \sigma^2)$, We use the same MLE to obtain:

$$\frac{\partial L}{\partial a} = 0 \to \sum_{i=1}^{n} (y_i - ag(x_{1,i}) - bf(x_{2,i}, x_{3,i}))g(x_{1,i}) = 0$$

$$\frac{\partial L}{\partial b} = 0 \to \sum_{i=1}^{n} (y_i - ag(x_{1,i}) - bf(x_{2,i}, x_{3,i}))f(x_{1,i}) = 0$$

$$\to a = \frac{\sum_{i=1}^{n} (y_i g(x_{1,i}) - bg(x_{1,i})f(x_{2,i}, x_{3,i}))}{\sum_{i=1}^{n} g(x_{1,i})^2}$$

$$\to b = \frac{\sum_{i=1}^{n} (y_i g(x_{1,i}) - ag(x_{1,i})f(x_{2,i}, x_{3,i}))}{\sum_{i=1}^{n} f(x_{2,i}, x_{3,i})^2}$$

Note that the two equations above are the same. However, we can get another equation using expected value. We want to represent rental price with h, hence $a + b = 1$. Thus, we can calculate a and b since we have a consistent system of 2 equations and 2 variables.

6 Conclusion

Overall, we found a way to add a distance parameter to the voter model to demonstrate a particle's effect on surrounding particles. For the second model, there was slight linear correlation between rental price and factors such as population density and crime severity index.

We would like to integrate the models together. The voter model does not tell us how much rent changes when price increases or decreases. We believe that it is possible to use the second model to predict the amount of rent increase or decrease.

This model can be made more realistic over an extended period of time by adding in inflation rates and using percentage change in rent prices rather than raw change.

7 Bibliography

Three-State Majority-vote Model on Scale-Free Networks and the Unitary Relation for Critical Exponents

Method of Maximum Likelihood for Simple Linear Regression

8 Code

8.1 Calculate parameters

```
import numpy as np

def calculate_parameters(data):
    # data is a list of (p,q,r) triples
    n=len(data)

    # Calculating values of summation for p,q,r
    p_sum=sum([triple[0] for triple in data])
    q_sum=sum([triple[1] for triple in data])
    r_sum=sum([triple[2] for triple in data])

    # Calculating values of summation of squares for p,q,r
    p_sqsum=sum([triple[0]**2 for triple in data])
    q_sqsum=sum([triple[1]**2 for triple in data])
    r_sqsum=sum([triple[2]**2 for triple in data])

    # Calculting summation of prodcuts of pq, pr and qr
    pq_sum=0
    pr_sum=0
    qr_sum=0
    for i in range(0,n):
        pq_sum+=data[i][0]*data[i][1]
        pr_sum+=data[i][0]*data[i][2]
        qr_sum+=data[i][1]*data[i][2]

    # Initializing matrix A
    para_matrix=np.array([[p_sqsum,pq_sum,p_sum],[pq_sum,q_sqsum,q_sum],
                                                    [p_sum,q_sum,n]])

    # Finding inverse of matrix A
    inv=np.linalg.inv(para_matrix)

    # Initializing vector B
    z_vector=np.array([[pr_sum],[qr_sum],[r_sum]])

    # Multiply inverse of A with B to find vector of estimated parameters a,b,c
    res=inv.dot(z_vector)
    return res

def main():
    # We can use real data in place of [1,2,3],[2,5,6],[3,6,8],[4,7,9]
    calculate_parameters([[1,2,3],[2,5,6],[3,6,8],[4,7,9]])


if __name__ == '__main__':
    main()
```

8.2 Gradient Descent

```
from sklearn.linear_model import LinearRegression
import matplotlib.pyplot as plt
import numpy as np
import math

def Multivariable_Linear_Regression(X,y,learningrate, iterations):
    """ Find the multivarite regression model for the data set
         Parameters:
          X: independent variables matrix
          y: dependent variables matrix
          learningrate: learningrate of Gradient Descent
          iterations: the number of iterations
        Return value: the final theta vector and the plot of cost function
    """
    m = len(X)
    cost_lst = []
    a,c = 1.01,1.01#[a,b,c] #change this
    y_pred = np.zeros((m,1))
    for i in range(iterations):
        a -= learningrate * sigmoid_da(a,c,X,Y,sigma,m)
        c -= learningrate * sigmoid_dc(a,c,X,Y,sigma,m)

        #predict from X
        for j in range(m):
            #a,b,c = theta...
            y_pred[j][0] = sigmoid(X[j][0],a,c)

        cost_value = y_pred - Y
        #Calculate the loss for each training instance
        total = 0
        for j in range(m):
            total += abs(cost_value[j][0])
            #Calculate the cost function for each iteration
        cost_lst.append(total)
    plt.scatter(range(300000), cost_lst, color = 'red')
    plt.title('Cost function Graph')
    plt.xlabel('Number of iterations')
    plt.ylabel('Cost')
    return a,c

def sigmoid(a, c, x):
    return a*(1/(1+np.exp(-c*x)) - 1/2)

def sigmoid_da(a, c, x, y, sigma,n):
    sum = 0
    for i in range(n):
        sum += (y[i][0] - sigmoid(a, c, x[i][0]))*(1/(1+np.exp(-c*x[i][0])) - 1/
    sum *= -1/sigma**2
    return sum

def sigmoid_dc(a, c, x, y, sigma,n):
    sum = 0
    for i in range(n):
        sum += (y[i][0] - sigmoid(a, c, x[i][0]))*((-x[i][0]*np.exp(-c*x[i][0]))/
    sum *= -a/sigma**2
    return sum

def main():
    #X is list of employment_rate
    #Y is list of rental_price
    X = np.array([[1,3,5,7,9]])
    Y = np.array([[sigmoid(1,1,1),sigmoid(1,1,3),sigmoid(1,1,5),sigmoid(1,1,7),s
    print(sigmoid(1,1,1))
    sigma = np.std([1,3,5,7,9])
    #reshape
    X = np.reshape(X,(len(X[0]),1))
    Y = np.reshape(Y,(len(Y[0]),1)) print(Multivariable_Linear_Regression(x,y, 0.
    plt.show()
```

```
if __name__ == '__main__':
    main()
```

8.3 Generating Data

```
import numpy as np
import math

def simulate_data(mu, sigma, n):
    normal_sim = np.random.normal(loc = mu, scale = sigma, size = n)
    return normal_sim

def function(x,a,b):
    return a/(1+e^{-bx}) - a/2
    }

def inverse(x):
    return -ln(a/(y+ a/2) - 1) /b

def get_data(a,b):
    n = len(rental_price)
    sim = simulate_data(0, 1, n)
    data = []
    for sample in range(n):
        data.append(inverse(rental_price[sample] - sim[sample]))
def main():
    a = 1
    b = 1
    #change a,b to change the data
    rental_price = [1,3,5,7,9]
    #can use real rental price instead



    #rental_price = f(x) + normal distribution

if __name__ == '__main__':
    main()
```